

# **CodonCode Aligner User Manual**

# Table of Contents

|  |                  |
|--|------------------|
| <b><u>About CodonCode Aligner.....</u></b>                 | <b><u>1</u></b>  |
| <u>System Requirements.....</u>                            | <u>1</u>         |
| <u>Licenses.....</u>                                       | <u>1</u>         |
| <b><u>Licenses for CodonCode Aligner.....</u></b>          | <b><u>3</u></b>  |
| <u>Demo Mode.....</u>                                      | <u>3</u>         |
| <u>Time-limited Trials.....</u>                            | <u>3</u>         |
| <u>Single-user Licenses.....</u>                           | <u>4</u>         |
| <u>Using Phred and Phrap from CodonCode Aligner.....</u>   | <u>4</u>         |
| <u>Replacement License Keys.....</u>                       | <u>5</u>         |
| <u>License Server Licenses.....</u>                        | <u>5</u>         |
| <u>Firewall ports for License Server use.....</u>          | <u>6</u>         |
| <b><u>CodonCode Aligner Features.....</u></b>              | <b><u>7</u></b>  |
| <u>Aligner Windows and Views.....</u>                      | <u>7</u>         |
| <u>The Main Window.....</u>                                | <u>7</u>         |
| <u>Project Window.....</u>                                 | <u>8</u>         |
| <u>Base View Window.....</u>                               | <u>8</u>         |
| <u>Trace Window.....</u>                                   | <u>9</u>         |
| <u>Quality View Window.....</u>                            | <u>10</u>        |
| <u>Contig View Window.....</u>                             | <u>10</u>        |
| <u>Restriction Map Window.....</u>                         | <u>12</u>        |
| <u>Customizable Toolbars.....</u>                          | <u>13</u>        |
| <u>Regions of Interest.....</u>                            | <u>13</u>        |
| <u>Take the Quick Tour!.....</u>                           | <u>13</u>        |
| <b><u>Usage Tips for CodonCode Aligner.....</u></b>        | <b><u>14</u></b> |
| <b><u>Quality Values In CodonCode Aligner.....</u></b>     | <b><u>15</u></b> |
| <u>Quality values explained.....</u>                       | <u>15</u>        |
| <u>Where quality values come from.....</u>                 | <u>15</u>        |
| <u>Artificial qualities.....</u>                           | <u>16</u>        |
| <u>Gap qualities.....</u>                                  | <u>16</u>        |
| <u>Viewing qualities.....</u>                              | <u>16</u>        |
| <u>Additional Reading.....</u>                             | <u>17</u>        |
| <b><u>Aligner Projects.....</u></b>                        | <b><u>18</u></b> |
| <b><u>Working with Aligner Projects.....</u></b>           | <b><u>20</u></b> |
| <u>Creating New Projects.....</u>                          | <u>20</u>        |
| <u>Opening Existing Projects.....</u>                      | <u>21</u>        |
| <u>Saving Projects.....</u>                                | <u>21</u>        |
| <u>Closing Projects.....</u>                               | <u>21</u>        |
| <b><u>Adding Sample Files to Aligner Projects.....</u></b> | <b><u>22</u></b> |
| <u>Opening Single Sample Files.....</u>                    | <u>22</u>        |
| <u>Adding Several Sample Files.....</u>                    | <u>22</u>        |
| <u>Adding a Subset of Samples.....</u>                     | <u>23</u>        |

# Table of Contents

|   |    |
|---|----|
| <b><u>Adding Sample Files to Aligner Projects</u></b>                             |    |
| <u>Adding Entire Folders of Sample Files</u>                                      | 24 |
| <u>Adding Entire Assemblies</u>   | 24 |
| <u>Importing Sequencher Projects (CAF Files)</u>                                  | 25 |
| <u>Importing Phrap Assemblies</u>   | 26 |
| <u>Tags in Phrap assemblies</u>   | 27 |
| <u>Import from GenBank</u>  | 27 |
| <u>Creating new text sequences</u>  | 27 |
| <u>Compatible File Formats</u>  | 28 |
| <u>Sample Names</u>   | 30 |
| <b><u>Organizing Samples And Contigs In Folders</u></b>                           | 31 |
| <u>Creating Folders</u>   | 31 |
| <u>Moving Samples and Contigs to Folders</u>                                      | 31 |
| <u>Renaming Folders</u>   | 31 |
| <u>Deleting Folders</u>   | 31 |
| <b><u>Removing Samples from an Aligner Project</u></b>                            | 33 |
| <b><u>Base Calling with PHRED</u></b>   | 34 |
| <u>How to call bases with PHRED from Aligner</u>                                  | 34 |
| <u>Prerequisites</u>  | 34 |
| <u>What Aligner does</u>  | 35 |
| <u>Editing the PHRED parameter file</u>   | 35 |
| <u>Base calling problems</u>  | 37 |
| <u>Cannot find base calling program</u>   | 37 |
| <u>Missing entries in the Phred parameter file</u>                                | 38 |
| <u>Problems reading the Phred parameter file</u>                                  | 39 |
| <u>Wrong command line parameters</u>  | 39 |
| <u>Problem running the workstation version of Phred</u>                           | 40 |
| <u>More about the Phred parameter file</u>  | 40 |
| <u>About Phred</u>  | 41 |
| <b><u>End Clipping</u></b>  | 43 |
| <u>End Clipping Parameters</u>  | 44 |
| <u>End Clipping Algorithms</u>  | 45 |
| <u>Method 1: Maximizing regions with error rates below a given threshold</u>      | 45 |
| <u>Method 2: Using separate criteria at the start and the end of the sequence</u> | 46 |
| <b><u>Trimming Vector Sequences</u></b>   | 47 |
| <b><u>Vector Library Files</u></b>  | 49 |
| <u>Using UniVec Library Files</u>   | 49 |
| <u>Using Custom Vector Files</u>  | 49 |
| <b><u>Assembly and Alignment</u></b>  | 51 |
| <u>Sequence Assembly</u>  | 51 |
| <u>NGS Assembly</u>   | 51 |

# Table of Contents

|  |           |
|--|-----------|
| <b><u>Assembly and Alignment</u></b>   |           |
| <u>Alignment with Clustal, Muscle, or based on protein translations (with MACSE)</u> | 52        |
| <u>Alignment to a Reference Sequence</u>   | 52        |
| <u>Alignment with Bowtie2</u>  | 53        |
| <b><u>Sequence Assembly</u></b>  | <b>54</b> |
| <u>Before assembling</u>   | 54        |
| <u>How to assemble</u>   | 54        |
| <u>Advanced assembly options</u>   | 54        |
| <u>Assemble with preprocessing</u>   | 55        |
| <u>Assemble in groups</u>  | 56        |
| <u>Assemble in groups by name part</u>   | 57        |
| <u>Assemble in groups by multiplex sequence tag</u>                                  | 59        |
| <u>Compare contigs to each other</u>   | 60        |
| <u>Assemble from scratch</u>   | 63        |
| <b><u>Sequence Assembly With Phrap</u></b>   | <b>64</b> |
| <u>How Aligner assembles using Phrap</u>   | 64        |
| <u>Things to Note for Phrap Assemblies</u>   | 64        |
| <u>About Phrap</u>   | 65        |
| <b><u>Alignments to a Reference Sequence</u></b>                                     | <b>66</b> |
| <u>Adding Samples to Alignments</u>  | 67        |
| <u>Advanced Alignments</u>   | 67        |
| <u>Align with Preprocessing</u>  | 68        |
| <u>Align to Reference in Groups</u>  | 69        |
| <u>Align to Reference from Scratch</u>   | 70        |
| <b><u>Contigs</u></b>  | <b>72</b> |
| <b><u>Unassembling Contigs</u></b>   | <b>73</b> |
| <b><u>Aligner Algorithms for Assembly and Alignments</u></b>                         | <b>74</b> |
| <u>Alignment to a Reference Sequence</u>   | 74        |
| <u>Assembly</u>  | 74        |
| <u>Consensus Calculation</u>   | 75        |
| <u>Quality-based Consensus</u>   | 75        |
| <u>Majority Consensus</u>  | 76        |
| <u>Inclusive Consensus</u>   | 77        |
| <u>Percentage-based Consensus</u>  | 77        |
| <u>Using the Reference Sequence as Consensus</u>                                     | 77        |
| <u>Rebuilding the Consensus Sequence</u>   | 78        |
| <u>Local, large gap, and end-to-end alignments</u>                                   | 78        |
| <b><u>NGS Data Analysis and Assembly</u></b>   | <b>80</b> |
| <u>Metagenomics</u>  | 80        |
| <u>Pre-processing tools</u>  | 80        |
| <u>NGS Assembly</u>  | 80        |



# Table of Contents

|  |     |
|--|-----|
| <b><u>NGS Data Analysis and Assembly</u></b>         |     |
| <u>NGS Alignments</u>                                | 80  |
| <b><u>Cluster Sequences</u></b>                      | 81  |
| <u>How Aligner Clusters</u>                          | 82  |
| <u>Clustering Result Files</u>                       | 82  |
| <u>Advanced Clustering Options</u>                   | 82  |
| <u>Limitations</u>                                   | 84  |
| <b><u>Identify Sequences</u></b>                     | 85  |
| <u>How to Identify Sequences</u>                     | 85  |
| <u>Query Sequences</u>                               | 85  |
| <u>Reference Sequences</u>                           | 85  |
| <u>Stringency and Options</u>                        | 86  |
| <u>Identify Sequences Results</u>                    | 89  |
| <u>Identify Sequences Algorithm</u>                  | 90  |
| <u>Limitations</u>                                   | 90  |
| <b><u>NGS Preprocessing Tools</u></b>                | 91  |
| <u>Trimming Reads and Removing Adapter Sequences</u> | 91  |
| <u>Error Correction with SparseAssembler</u>         | 95  |
| <u>Other NGS Tools</u>                               | 96  |
| <b><u>NGS Assembly</u></b>                           | 97  |
| <u>NGS Assembly with CodonCode Aligner</u>           | 97  |
| <u>NGS Assembly with Tadpole</u>                     | 98  |
| <u>NGS Assembly with SparseAssembler</u>             | 101 |
| <u>Other NGS Assembly Programs</u>                   | 101 |
| <u>Scaffolding</u>                                   | 101 |
| <b><u>Separate Alleles</u></b>                       | 103 |
| <u>How to Separate Alleles</u>                       | 103 |
| <u>Algorithm Details</u>                             | 103 |
| <u>Starting from Contigs</u>                         | 106 |
| <u>Limitations</u>                                   | 106 |
| <u>Files Generated</u>                               | 107 |
| <b><u>Alignments with Bowtie 2</u></b>               | 108 |
| <u>Viewing Bowtie2 alignments with Tablet</u>        | 112 |
| <b><u>Regions of Interest: Features</u></b>          | 115 |
| <u>Defining Features</u>                             | 115 |
| <u>Moving to Features</u>                            | 117 |
| <u>View, Print, Export, Import Features</u>          | 117 |
| <u>Find Common Features</u>                          | 117 |

# Table of Contents

|  |            |
|--|------------|
| <b><u>Finding Mutations</u></b>                      | <b>120</b> |
| <u>Prerequisites</u>                                 | 120        |
| <u>How To Find SNPs</u>                              | 120        |
| <u>Fixing Errors</u>                                 | 124        |
| <u>Tags for Finding Mutations</u>                    | 124        |
| <u>Defining the Coding Region</u>                    | 125        |
| <u>Numbering the Coding Sequence</u>                 | 126        |
| <u>Excluding Regions from Analysis</u>               | 129        |
| <u>How Aligner Finds SNPs</u>                        | 129        |
| <u>Limitations</u>                                   | 130        |
| <b><u>Heterozygous Insertions and Deletions</u></b>  | <b>132</b> |
| <u>Finding Heterozygous Indels</u>                   | 133        |
| <u>Hetero Indel Scores</u>                           | 134        |
| <u>False Negatives</u>                               | 134        |
| <u>False Positives</u>                               | 135        |
| <u>Splitting Heterozygous Indels</u>                 | 135        |
| <u>Processing Heterozygous Indels</u>                | 135        |
| <u>Limitations</u>                                   | 138        |
| <u>Processing Pre-Requisites and Limitations</u>     | 138        |
| <u>Interpreting Results</u>                          | 138        |
| <u>Acknowledgements</u>                              | 139        |
| <b><u>Methylation Analysis</u></b>                   | <b>140</b> |
| <u>Prerequisites</u>                                 | 140        |
| <u>Raw ABI Data</u>                                  | 140        |
| <u>Reference Sequence Alignments</u>                 | 140        |
| <u>How to Analyze Methylation</u>                    | 140        |
| <u>Viewing Raw Trace Data</u>                        | 141        |
| <u>Stretching raw data</u>                           | 142        |
| <u>Methylation Analysis Algorithm</u>                | 142        |
| <b><u>Primer Design</u></b>                          | <b>144</b> |
| <u>How to Pick Primers</u>                           | 144        |
| <u>Primer Design Parameters</u>                      | 144        |
| <u>PCR Parameters</u>                                | 146        |
| <u>Sequencing Parameters</u>                         | 146        |
| <u>Conditions &amp; Advanced Settings Parameters</u> | 147        |
| <u>Primer Results</u>                                | 148        |
| <u>Exporting Primers</u>                             | 152        |
| <u>Additional Information</u>                        | 152        |
| <b><u>RFLP Analysis</u></b>                          | <b>153</b> |
| <u>Prerequisites</u>                                 | 153        |
| <u>How to Analyze RFLPs</u>                          | 153        |
| <u>Samples with Different Lengths</u>                | 153        |
| <u>Analysis Options</u>                              | 153        |
| <u>Analysis Results</u>                              | 155        |

# Table of Contents

|  |            |
|--|------------|
| <b><u>Restriction Cloning</u></b> .....                                      | <b>157</b> |
| <b><u>Gibson Assembly</u></b> .....  | <b>160</b> |
| <b><u>Dot Plots</u></b> .....  | <b>164</b> |
| <b><u>Editing Samples</u></b> .....  | <b>168</b> |
| <b><u>Windows for Editing Samples</u></b> .....                              | <b>171</b> |
| <b><u>Cursor Positioning and Movement</u></b> .....                          | <b>172</b> |
| <u>Moving to Features, Ambiguities, Mismatches or Edited Bases</u> .....     | 172        |
| <b><u>Setting Base Numbers</u></b> .....                                     | <b>173</b> |
| <u>Rotating circular samples</u> .....                                       | 173        |
| <b><u>Synchronizing Sample Starts</u></b> .....                              | <b>174</b> |
| <b><u>Selecting Bases</u></b> .....  | <b>177</b> |
| <u>Selecting from sequence start to current cursor position</u> .....        | 177        |
| <u>Selecting from current cursor position to the end of a sequence</u> ..... | 177        |
| <u>Selecting all bases in a sequence</u> .....                               | 177        |
| <u>Selecting Ranges of Bases</u> .....                                       | 177        |
| <b><u>Selecting Samples and Contigs</u></b> .....                            | <b>179</b> |
| <b><u>Changing Bases</u></b> .....   | <b>180</b> |
| <u>Manual Editing</u> .....  | 180        |
| <u>Making Bases Lower or Upper Case</u> .....                                | 180        |
| <u>Automatic Edits</u> .....   | 180        |
| <u>Match Consensus</u> .....   | 181        |
| <u>Call Second Peaks Higher Than</u> .....                                   | 181        |
| <u>Change Ambiguities to Single Bases</u> .....                              | 182        |
| <u>Change Low Quality Bases to N</u> .....                                   | 182        |
| <u>Undo Auto Edits</u> .....   | 182        |
| <u>Change Bases Options</u> .....  | 182        |
| <b><u>Deleting Bases</u></b> .....   | <b>184</b> |
| <u>Using the Keyboard to Delete Bases</u> .....                              | 184        |
| <u>Menus for Deleting Bases</u> .....  | 184        |
| <b><u>Deleting Samples</u></b> .....   | <b>185</b> |
| <b><u>Moving Gaps and Samples</u></b> .....                                  | <b>186</b> |
| <u>Moving gaps in contigs</u> .....  | 186        |
| <u>Moving samples in contigs</u> .....                                       | 186        |
| <u>Moving samples to "Unassembled Samples"</u> .....                         | 186        |

# Table of Contents

|   |            |
|---|------------|
| <b><u>Inserting Gaps and Bases</u></b> .....              | <b>187</b> |
| <u>Adding Bases at the End of Reads</u> .....             | 187        |
| <b><u>Reverse Complementing</u></b> .....                 | <b>188</b> |
| <b><u>Editing Sample Information</u></b> .....            | <b>189</b> |
| <u>Changing Sample Names</u> .....                        | 189        |
| <u>Viewing Chromatogram Information</u> .....             | 190        |
| <u>Viewing Sample Tags</u> .....                          | 192        |
| <u>Confirming Tags</u> .....                              | 193        |
| <u>Viewing "Local" Tags</u> .....                         | 193        |
| <u>Adding Tags</u> .....                                  | 193        |
| <u>Adding Tags to All Sequences</u> .....                 | 194        |
| <u>Importing Annotation (Tags)</u> .....                  | 194        |
| <b><u>Saving Edits</u></b> .....                          | <b>196</b> |
| <b><u>Undo and Redo</u></b> .....                         | <b>197</b> |
| <b><u>Copy and Paste</u></b> .....                        | <b>198</b> |
| <u>Copy (selected sequence)</u> .....                     | 198        |
| <u>Paste</u> .....  | 198        |
| <b><u>Editing Contigs</u></b> .....                       | <b>199</b> |
| <b><u>Adding Samples to Contigs</u></b> .....             | <b>200</b> |
| <u>Duplicating samples</u> .....                          | 200        |
| <b><u>Merging Contigs</u></b> .....                       | <b>201</b> |
| <b><u>Removing Samples from Contigs</u></b> .....         | <b>202</b> |
| <b><u>Deleting Parts of Contigs</u></b> .....             | <b>204</b> |
| <b><u>Removing Consensus Gaps</u></b> .....               | <b>205</b> |
| <u>How to remove consensus gaps</u> .....                 | 205        |
| <b><u>Alignment Locations - Start and End</u></b> .....   | <b>207</b> |
| <b><u>Reverse Complementing</u></b> .....                 | <b>208</b> |
| <b><u>Splitting Contigs</u></b> .....                     | <b>209</b> |
| <b><u>Unassembling Contigs</u></b> .....                  | <b>212</b> |
| <b><u>Roundtrip Editing</u></b> .....                     | <b>213</b> |
| <u>How To Roundtrip Edit with CodonCode Aligner</u> ..... | 213        |
| <u>Limitations</u> .....                                  | 214        |

# Table of Contents

|  |            |
|--|------------|
| <b><u>Editing Contig Information</u></b> .....                   | <b>215</b> |
| <b><u>Search for Sequences</u></b> .....                         | <b>216</b> |
| <b><u>BLAST Searches</u></b> .....                               | <b>218</b> |
| <u>MegaBLAST</u> .....   | 218        |
| <u>Nucleotide (blastn)</u> .....                                 | 218        |
| <u>Translated (blastx)</u> .....                                 | 218        |
| <u>Translated (tblastx)</u> .....                                | 218        |
| <b><u>Exporting</u></b> .....                                    | <b>219</b> |
| <b><u>Export Project Summary</u></b> .....                       | <b>220</b> |
| <b><u>Exporting Samples</u></b> .....                            | <b>222</b> |
| <u>Export options for FASTA files</u> .....                      | 223        |
| <u>Exporting SCF files</u> .....                                 | 224        |
| <b><u>Exporting Consensus Sequences</u></b> .....                | <b>225</b> |
| <u>Export options for FASTA files</u> .....                      | 225        |
| <b><u>Exporting Assemblies</u></b> .....                         | <b>227</b> |
| <u>ACE Format Exports</u> .....                                  | 227        |
| <u>NEXUS/PAUP Format Exports</u> .....                           | 228        |
| <u>Phylip Format Exports</u> .....                               | 228        |
| <u>Other Formats for Exporting Assemblies</u> .....              | 229        |
| <b><u>Exporting Features</u></b> .....                           | <b>230</b> |
| <b><u>Exporting Aligner Projects in the Old Format</u></b> ..... | <b>232</b> |
| <b><u>Exporting Protein Translation</u></b> .....                | <b>233</b> |
| <b><u>Exporting Differences</u></b> .....                        | <b>234</b> |
| <b><u>Exporting Trees</u></b> .....                              | <b>236</b> |
| <b><u>Aligner Windows</u></b> .....                              | <b>237</b> |
| <u>The Project View Window</u> .....                             | 237        |
| <u>Selecting Samples and Contigs</u> .....                       | 237        |
| <u>Menu shortcuts: Buttons and Popup menus</u> .....             | 238        |
| <u>Project View Columns And Sorting</u> .....                    | 238        |
| <u>Status messages</u> .....                                     | 239        |
| <u>Colored Labels</u> .....                                      | 239        |
| <b><u>Trace View</u></b> .....                                   | <b>241</b> |
| <u>General Information</u> .....                                 | 241        |
| <u>Opening a Trace View</u> .....                                | 241        |

# Table of Contents

|  |     |
|--|-----|
| <b><u>Trace View</u></b>                               |     |
| <u>Scrolling and Scaling in Trace View</u>             | 241 |
| <u>Trace Sharpening</u>                                | 242 |
| <u>Colors and Highlighting</u>                         | 243 |
| <u>Automatic Trace Selection</u>                       | 243 |
| <u>Hiding Some Traces</u>                              | 244 |
| <u>Printing Traces</u>                                 | 245 |
| <b><u>Base View Window</u></b>                         | 247 |
| <b><u>Quality View Window</u></b>                      | 250 |
| <b><u>Contig View Window</u></b>                       | 251 |
| <u>Moving Around</u>                                   | 253 |
| <u>Contig Overview Panel</u>                           | 253 |
| <u>Coverage graph in the overview panel</u>            | 254 |
| <u>Navigating using the overview panel</u>             | 254 |
| <u>Changing the display of the sample arrows</u>       | 254 |
| <u>Contig Difference Table</u>                         | 256 |
| <u>Display options for the difference table</u>        | 257 |
| <u>Navigating using the difference table</u>           | 261 |
| <u>Aligned Bases and Consensus Protein Translation</u> | 261 |
| <u>Phylogenetic Trees</u>                              | 262 |
| <u>Split Contig From Tree</u>                          | 267 |
| <u>Building Trees for Selected Bases Only</u>          | 268 |
| <u>Masking Bases Matching the Consensus</u>            | 269 |
| <u>Zooming in the Contig View</u>                      | 270 |
| <u>Sorting Reads in the Contig View</u>                | 273 |
| <u>Automatic Trace Selection</u>                       | 274 |
| <u>Printing Contigs</u>                                | 274 |
| <b><u>Feature Window</u></b>                           | 276 |
| <b><u>Restriction Map View</u></b>                     | 278 |
| <u>Single Line Map</u>                                 | 278 |
| <u>Multi Line Map</u>                                  | 279 |
| <u>Text Map</u>  | 280 |
| <u>Virtual Gel</u>                                     | 280 |
| <u>Selecting Fragments</u>                             | 282 |
| <u>Restriction Enzymes in Aligner</u>                  | 284 |
| <b><u>Closing Windows</u></b>                          | 285 |
| <b><u>Scripting CodonCode Aligner</u></b>              | 286 |
| <b><u>Preferences and Settings</u></b>                 | 290 |

# Table of Contents

|   |            |
|---|------------|
| <b><u>Alignment Preferences</u></b> .....                       | <b>292</b> |
| <u>Algorithm</u> .....  | 293        |
| <u>Minimum Percent Identity</u> .....                           | 293        |
| <u>Minimum Overlap Length</u> .....                             | 293        |
| <u>Minimum Alignment Score</u> .....                            | 293        |
| <u>Maximum Unaligned End Overlap</u> .....                      | 294        |
| <u>Bandwidth (Maximum Gap Size)</u> .....                       | 294        |
| <u>Word Length</u> .....  | 295        |
| <u>Match scoring</u> .....                                      | 295        |
| <u>Clipping Uncovered Regions</u> .....                         | 295        |
| <u>Reverse-complement for external programs</u> .....           | 296        |
| <u>Restoring default parameters</u> .....                       | 296        |
| <b><u>Assembly Preferences</u></b> .....                        | <b>297</b> |
| <u>Algorithm</u> .....  | 298        |
| <u>Minimum Percent Identity</u> .....                           | 298        |
| <u>Minimum Overlap Length</u> .....                             | 298        |
| <u>Minimum Alignment Score</u> .....                            | 299        |
| <u>Maximum Unaligned End Overlap</u> .....                      | 299        |
| <u>Bandwidth (Maximum Gap Size)</u> .....                       | 299        |
| <u>Word Length</u> .....  | 300        |
| <u>Maximum Successive Failures</u> .....                        | 300        |
| <u>Match scoring</u> .....                                      | 301        |
| <u>Restoring default parameters</u> .....                       | 301        |
| <b><u>Base Calling Preferences</u></b> .....                    | <b>302</b> |
| <b><u>Base Color Preferences</u></b> .....                      | <b>304</b> |
| <u>Switching color schemes</u> .....                            | 305        |
| <u>Quality-based 3 color scheme</u> .....                       | 305        |
| <b><u>Continuous, quality-based background colors</u></b> ..... | <b>307</b> |
| <b><u>Base-specific colors</u></b> .....                        | <b>309</b> |
| <b><u>Translation based background colors</u></b> .....         | <b>313</b> |
| <b><u>Consensus Preferences</u></b> .....                       | <b>316</b> |
| <u>Consensus Method</u> .....                                   | 317        |
| <u>Consensus Gaps</u> .....                                     | 317        |
| <u>Quality Scores At Discrepancies And Edits</u> .....          | 318        |
| <u>Reference Sequence Alignments</u> .....                      | 318        |
| <u>External Consensus Sequences</u> .....                       | 318        |
| <u>Masking Low Coverage Regions</u> .....                       | 318        |
| <b><u>Clicking &amp; Scrolling Preferences</u></b> .....        | <b>320</b> |

# Table of Contents

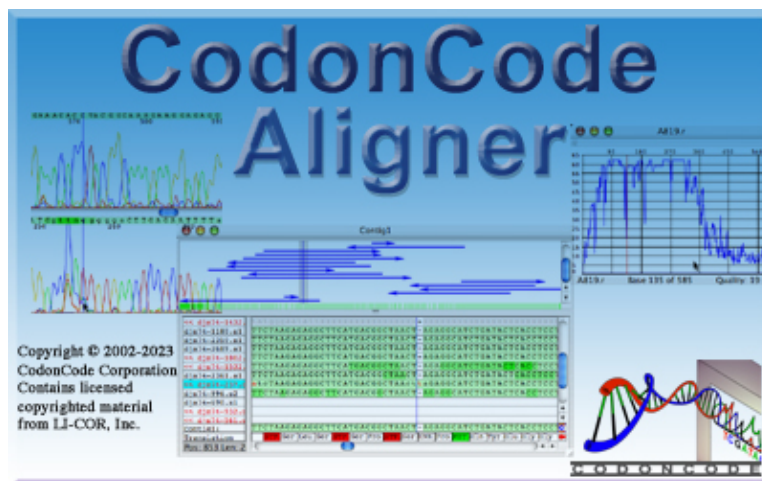
|  |            |
|--|------------|
| <b><u>End Clipping Preferences</u></b> .....   | <b>322</b> |
| <u>Automatically removing short and low-quality sequences after end clipping</u> ..... | 324        |
| <b><u>Feature Preferences</u></b> .....  | <b>325</b> |
| <u>Specifying subsets of tags</u> .....  | 326        |
| <b><u>Highlighting Preferences</u></b> .....   | <b>328</b> |
| <b><u>License Server Preferences</u></b> .....   | <b>330</b> |
| <b><u>Memory Preferences</u></b> .....   | <b>332</b> |
| <u>Memory on Windows</u> .....   | 333        |
| <u>Changing memory on OS X</u> .....   | 333        |
| <b><u>Mutation Detection Preferences</u></b> .....                                     | <b>335</b> |
| <u>Detection sensitivity</u> .....   | 336        |
| <u>Marking mutations</u> .....   | 336        |
| <u>Finding only homozygous mutations</u> .....   | 337        |
| <u>Finding indels</u> .....  | 337        |
| <b><u>Open &amp; Save Preferences</u></b> .....  | <b>339</b> |
| <b><u>Phrap Assembly Preferences</u></b> .....   | <b>341</b> |
| <b><u>Preference Options</u></b> .....   | <b>343</b> |
| <b><u>Printing Preferences</u></b> .....   | <b>345</b> |
| <u>Trace View Printing</u> .....   | 346        |
| <u>Contig View Printing</u> .....  | 346        |
| <b><u>Protein Translation Preferences</u></b> .....                                    | <b>348</b> |
| <u>Sequence Translation</u> .....  | 349        |
| <u>Consensus Translation</u> .....   | 350        |
| <b><u>Restriction Map Preferences</u></b> .....  | <b>352</b> |
| <u>Selecting Enzymes</u> .....   | 352        |
| <u>Restriction Map Options</u> .....   | 354        |
| <b><u>Sample Name Preferences</u></b> .....  | <b>358</b> |
| <u>Defining sample names</u> .....   | 358        |
| <u>Defining delimiters</u> .....   | 361        |
| <b><u>Startup Preferences</u></b> .....  | <b>364</b> |
| <u>The Startup Dialog</u> .....  | 365        |
| <b><u>Update Preferences</u></b> .....   | <b>366</b> |



# Table of Contents

|   |            |
|---|------------|
| <b><u>Toolbar Preferences</u></b> .....             | <b>368</b> |
| <b><u>Vector Trimming Preferences</u></b> .....     | <b>370</b> |
| <b><u>View Preferences</u></b> .....                | <b>372</b> |
| <b><u>Warning Preferences</u></b> .....             | <b>374</b> |
| <b><u>Window Placement Preferences</u></b> .....    | <b>376</b> |
| <b><u>CodonCode Aligner Help by Menu</u></b> .....  | <b>378</b> |
| <u>Aligner Menu (on Mac OS X only)</u> .....        | 378        |
| <u>File Menu</u> .....                              | 378        |
| <u>Edit Menu</u> .....                              | 378        |
| <u>Go Menu</u> .....                                | 379        |
| <u>Sample Menu</u> .....                            | 379        |
| <u>Contig Menu</u> .....                            | 380        |
| <u>Tools Menus</u> .....                            | 381        |
| <u>View Menu</u> .....                              | 381        |
| <u>Window Menu</u> .....                            | 382        |
| <u>Help Menu</u> .....                              | 382        |
| <b><u>Memory Requirements</u></b> .....             | <b>383</b> |
| <u>How Aligner memory on Windows is set</u> .....   | 383        |
| <u>How Aligner memory on Mac OS X is set</u> .....  | 384        |
| <b><u>CodonCode Aligner Release Notes</u></b> ..... | <b>385</b> |
| <b><u>Checking for Aligner Updates</u></b> .....    | <b>386</b> |
| <u>Visiting the Aligner Web Site</u> .....          | 386        |

# About CodonCode Aligner



Copyright 2002-2023 CodonCode Corporation. All Rights Reserved.

## For technical support:

e-mail: [support@codoncode.com](mailto:support@codoncode.com)

WWW: [www.codoncode.com](http://www.codoncode.com)

USA: (781) 686-1131

## System Requirements

**Mac OS:** Requires macOS version 10.10 or newer

**Windows:** Requires Windows XP, Vista, Windows 7, or newer and 2 GB RAM. 64-bit Windows is suggested for large projects and best performance.

**Other operating systems:** CodonCode Aligner is not available for other operating systems.

## Licenses

After installing, Aligner will start in **Demo** mode. Demo mode allows you to use Aligner for viewing sequence traces, as well as printing and editing unassembled traces. You can also open existing Aligner projects. However, you cannot print, save, or export projects that contain contigs in demo mode.

Demo mode also allows you to try out most of Aligner's functions, including end clipping, assembly, and so on; but again, you will not be able to save, import, export, or print after trying these functions in demo mode.

The first time that you use CodonCode Aligner, you will automatically receive a time-limited trial, which will enable you to use all functions for 30 days. For more information about how to get licenses for **regular** use, please check the "[Licenses for Aligner](#)" section.

## **Acknowledgements**

CodonCode Aligner contains licensed copyrighted material from LI-COR, Inc.

# Licenses for CodonCode Aligner

After installing, CodonCode Aligner will start up in [Demo](#) mode. The first time that you use CodonCode Aligner, you will automatically receive a time-limited trial, which will enable you to use all functions for 30 days.

When you are ready to purchase a license, you can choose between [single-user licenses](#) and [license server licenses](#).

All license keys other than the demo key are specific to a given computer, and will not work if transferred to a different computer. Certain changes on your computer, for example installing a new operating system or certain repairs, may also invalidate your license so that you will need to request a new license key.

## Demo Mode

In Demo mode, CodonCode Aligner works as a fully functional **trace viewer and editor**: you can open, view, edit, and print unassembled sequences. You can also save projects that contain only unassembled sequences, unless you have "processed" any of the sequences with Aligner's functions for end clipping, base calling, sequence assembly, and so on (Aligner will warn you before doing anything that will disable saving and printing).

You can also open existing Aligner projects in demo mode, for example the projects in the "Example Files" folder inside the "CodonCode Aligner" folder. However, if an opened project contains any contigs, you will not be able to save, import, export, or print in demo mode.

The demo mode also allows you to test most of Aligner's advanced functions, with the following exceptions:

- Base calling with Phred
- Assembly with Phrap (however, you can assemble using Aligner's own assembly algorithm)
- Finding mutations and heterozygous indels
- Processing of heterozygous insertions or deletions (indels)

If you would like to fully test Aligner with your own data, you can request a license key for a time-limited trial, as explained in the next section.

## Time-limited Trials

The first time you use CodonCode Aligner, you will automatically receive a time-limited trial that enables you to use all functions for 30 days. After the initial trial expires, Aligner will revert to Demo mode. You will still be able to open any existing projects you may have, but your ability to save or print will be limited, as described above.

If you need additional time to evaluate CodonCode Aligner, please contact us by email to support (at) codoncode.com. We will need to have your name and email address to send you a trial license (note that the email address cannot be an anonymous email address like xx@yahoo.com).

## Single-user Licenses

Single-user license keys that you purchased from CodonCode Corporation allow the use of CodonCode Aligner on a single computer. To enter a single-user license key that you received from CodonCode, start Aligner, and press the "Enter License" button in the startup dialog. If Aligner is already running, select "**License...**" from the "**Help**" menu, and then press the "**Enter New License**" button.

This will display the "Enter New License" dialog:



Enter the license string you received from CodonCode Corporation into the "License String" field, preferably by using copy and paste (*use the "Paste License" button, or the keyboard shortcut for pasting in Aligner - Command-V on OS X or Control-V on Windows*). Then, click the "Apply" button.

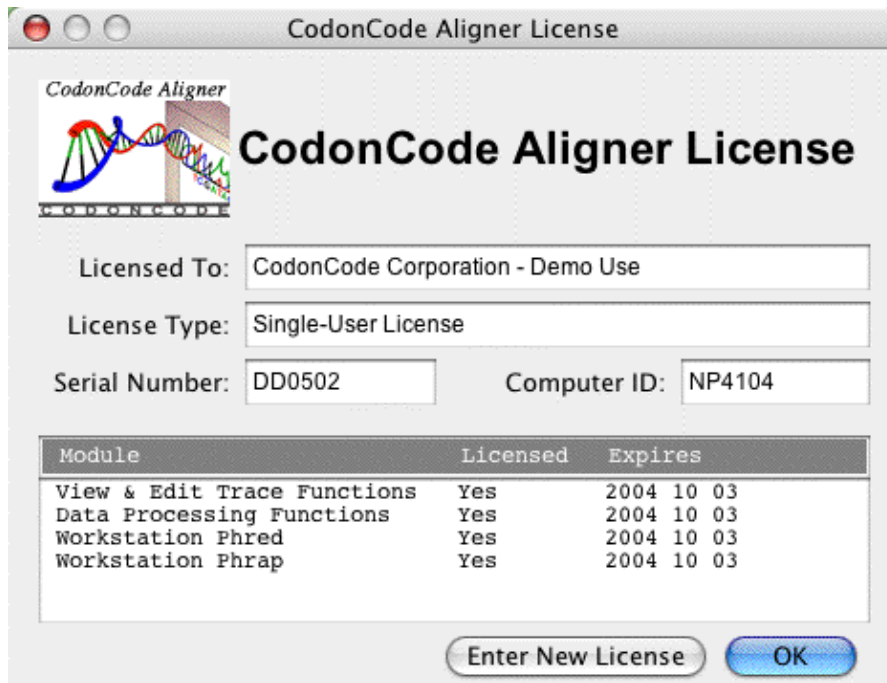
Licenses are bound to a single computer ID. If you try to install CodonCode Aligner on different computers and use the license there, the license will not be valid. A single user license allows you to install Aligner on one computer. If you plan to use your license on one of several computers, you need a [License Server License](#).

## Using Phred and Phrap from CodonCode Aligner

CodonCode Aligner allows you to use the base calling program Phred and the assembly program Phrap to base call and assemble your sequence data. Phred and Phrap are separate programs, and distributed by CodonCode Corporation through a distribution agreement with the University of Washington. Users at academic and non-profit institutions may use Phred and Phrap free of charge; users at for-profit institutions have to purchase separate licenses for Phred and Phrap.

When installing CodonCode Aligner, special "workstation" versions of Phred and Phrap are also installed. These workstation versions are identical to the regular Phred-Phrap versions, except that:

- the workstation versions of Phred and Phrap can only be run from CodonCode Aligner (not from the command line or scripts)
- they require a valid license for the "Workstation Phred" and "Workstation Phrap" modules, as shown in the license dialog:



Note that the "Licensed" column in the example above says "Yes" for all modules, indicating that you can use the workstation versions of Phred and Phrap that were installed with Aligner. Time-limited trial license will generally allow the (time-limited) use of workstation Phred and Phrap. Licenses purchased by academic users will also generally include licenses for workstation Phred and Phrap, since Phred and Phrap can be obtained by academic users free of charge. Licenses purchased by users at for-profit institutions, however, will not include permissions to use workstation Phred and Phrap, unless a separate license fee was paid.

If you have your own licensed copies of Phred and Phrap, you can use these from Aligner instead of the workstation versions (except when running in demo mode) by changing the [base calling preferences](#) or the [Phrap assembly preferences](#).

## Replacement License Keys

Under certain circumstances, you may need a **replacement license key**. This can happen if you replace your computer, if you install a new operating system, replace the hard drive, and similar circumstances. If you receive a replacement license, you must make sure that the original license is not used anymore. If Aligner detects that a replacement license and the original license are both used, Aligner will invalidate one or both of the licenses, and you may have to request another replacement license.

In general, you should not move the folder where CodonCode Aligner is installed after the installation. If you move the folder to a different location on the same computer, you may have to re-enter the license information.

## License Server Licenses

License Server licenses for CodonCode Aligner allow you to install CodonCode Aligner on an unlimited number of computers, but limit concurrent use to the number of licenses you purchased. Using License Server licenses requires installation of a separate program ("Aligner License Server") on a computer that acts as the license server.

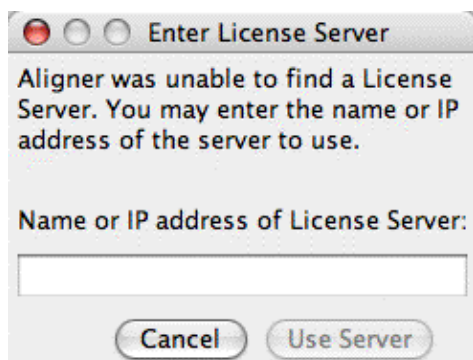
## CodonCode Aligner User Manual

If Aligner License Server is available for your laboratory or department, you can choose to use the Aligner License Server in the dialog that CodonCode Aligner displays when starting up:



If you click the **"Use License Server"** button, Aligner will attempt to locate a License Server for CodonCode Aligner on your local network. If a License Server is found, and a license is available, it will be checked out, and you can start using CodonCode Aligner. The license key will be returned so that someone else can use it when you quit CodonCode Aligner.

If CodonCode Aligner cannot find a License Server, the following dialog will be displayed:



If you know the computer name or the IP address for the license server computer, you can enter it here (if you do not know the name or IP address, ask your local system administrator).

## Firewall ports for License Server use

If Aligner cannot connect to the License Server, it may be because Aligner License Server is not running, or because a firewall is blocking the network traffic between your computer and the license server computer.

Please make sure that the License Server is running, and to allow communication on the following UDP **and** TCP ports: 123, 16030, 16031, 32156, 32157, 54643, and 54644. Communication must be enabled on both the License Server computer and on the computers where CodonCode Aligner is used.

# CodonCode Aligner Features

CodonCode Aligner is a versatile, powerful and easy-to-use DNA and RNA sequence assembler, aligner, and editor. Aligner imports a wide variety of sample file types into "projects". Aligner projects are collections of sample files and their resulting alignments or contigs.

Aligner works best with sequences that have chromatograms and quality scores, for example from SCF files generated with PHRED or ABI files that were base called with the KB base caller. However, you can also import text sequences without chromatograms, and trace sequences without quality scores (for example older ABI files, or ABI files analysed with the ABI base caller).

Aligner protects your original sequence data by making copies of the original sample data when files are imported. When you make edits in Aligner, you change only the copied data files.

## Aligner Windows and Views

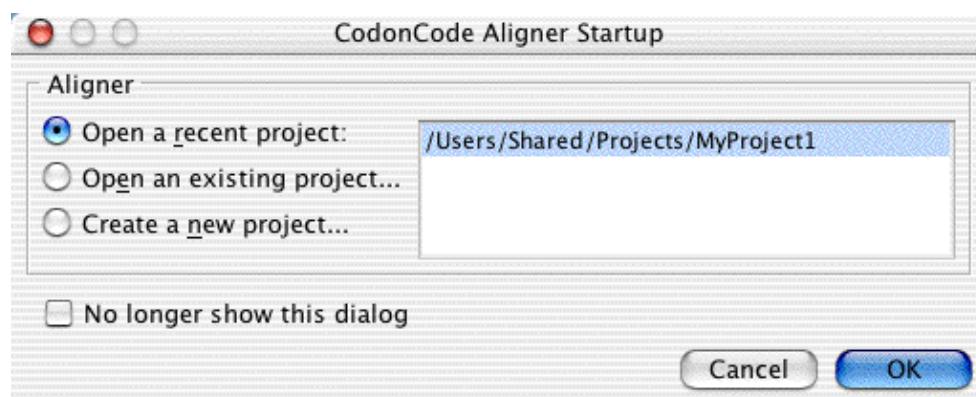
This section describes the most important Aligner windows (or "views") briefly. For more detailed descriptions, follow the links for each window, or read the sub-topics in the "Aligner Windows" help section.

### *The Main Window*

On **Windows**, opening CodonCode Aligner will open the main application window ("root window"), which contains the application menu and all other windows opened by Aligner.

Since CodonCode Aligner follows the common user interface standards for Windows and Mac OS X, there are some user interface differences between the Windows and OS X versions.

On **Mac OS X**, there is no such "root window". Instead, the main menu bar at the top of the screen is used. When you first start Aligner on Mac OS X, a startup dialog is displayed that allows you to open an existing project or create a new project:

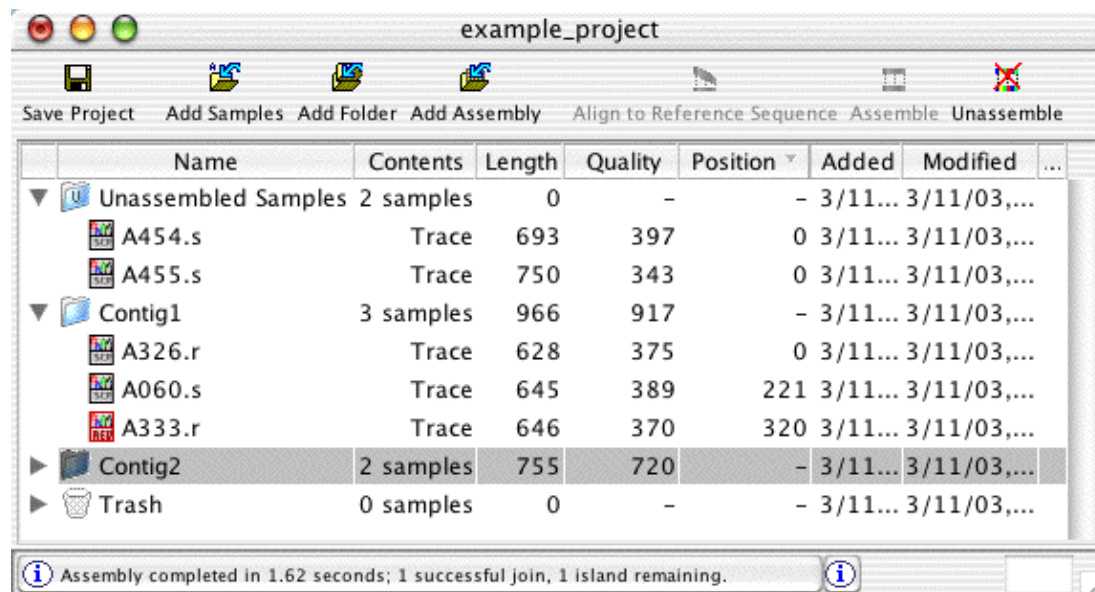




After selecting or creating a project, the project window described in the next section will be shown. *(If you hit cancel, or previously selected the "No longer show this dialog" checkbox, you will not see this window - the application will have only a menu bar without any windows. On Windows, the same dialog is also shown upon startup).*

## Project Window

To do anything in CodonCode Aligner, you must create a project, or open a previously created project. This will create a "Project Window", where the contents of the project are shown, similar to the way Finder or Explorer display contents of disks and folders:



In the project window, you can add samples, an entire folder of samples, or CodonCode Aligner projects and Phrap assemblies using the buttons on the top, or the corresponding items in the "File" menu. You can also add sequence files to a project by dropping the files onto the project view.

Newly imported samples will be in the "Unassembled Samples" folder.

Any contigs that are created or imported from other assemblies are shown as separate folders. You can also create "manual" folders to organize your samples and contigs.

In the project view, you can also select samples and contigs by clicking or shift-clicking, and then viewing the bases, traces, or contigs, assemble the samples, and so on.

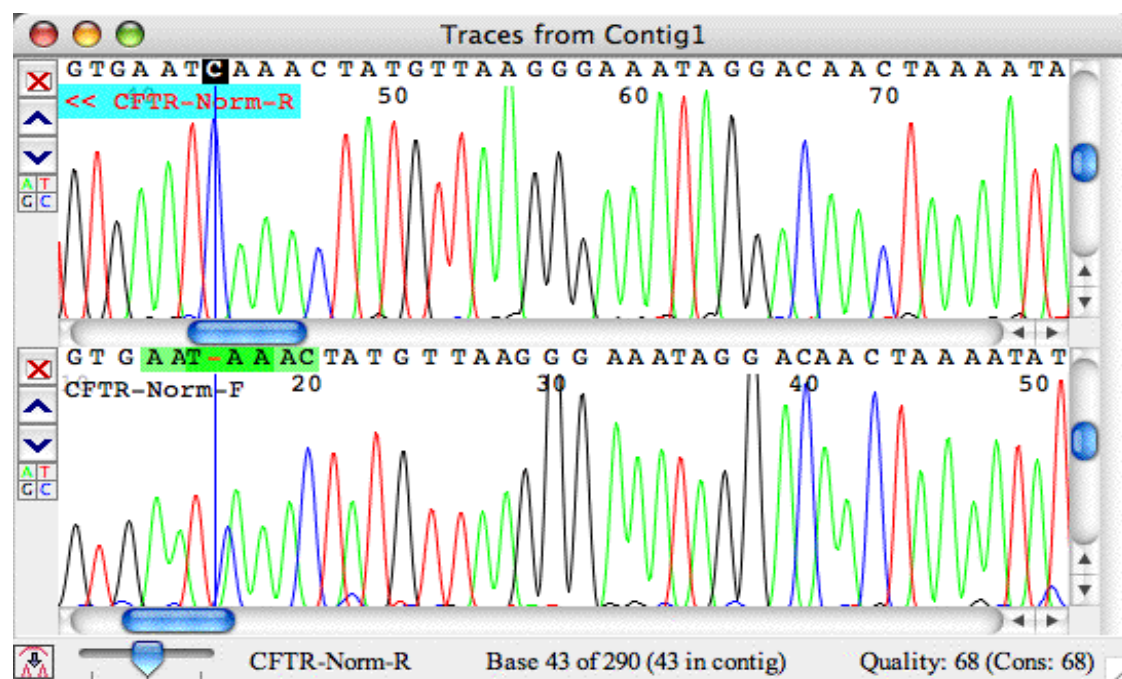
## Base View Window

The base view window shows all the bases for a sequence, as shown below:

As in the other windows that show bases, the quality scores for each base can be shown as a colored background. In the example above, base calls with qualities below 20 ("low quality bases", accuracy below 99%) are shown with a dark blue background, base calls with quality scores between 20 and 30 on light blue background, and base calls with qualities above 30 ("very high quality bases", accuracy above 99.9%) on a white background. You can change the colors and thresholds in the "Preferences" dialog.

## Trace Window

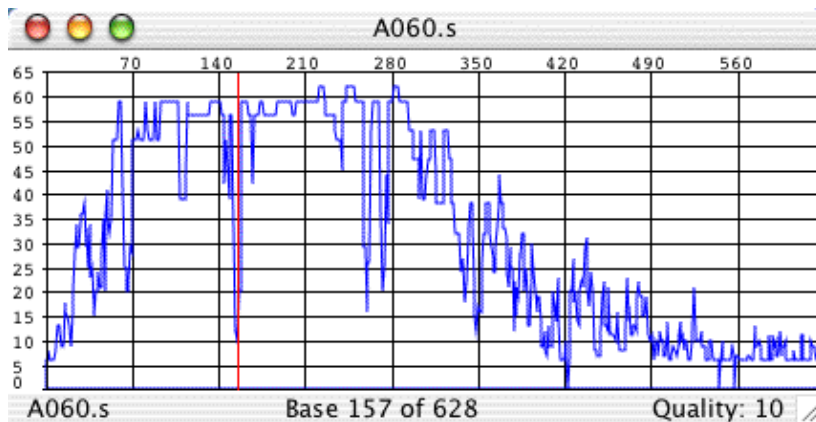
The trace window allows you to view and edit sequences that have chromatogram data, for example ABI and SCF files.



Each contig has its own trace window, which can show the traces for one or more sequences. If the traces for all sequences cannot be shown in the window at the same time, you can use the scroll bar on the right to scroll through the different sequences.

### Quality View Window

For sequences that have qualities, for example Phred-generated SCF files, you can get a quick overview of the quality by selecting the sample in the project window, and then choosing "Qualities" in the "View" menu. This will open up a window like this one:

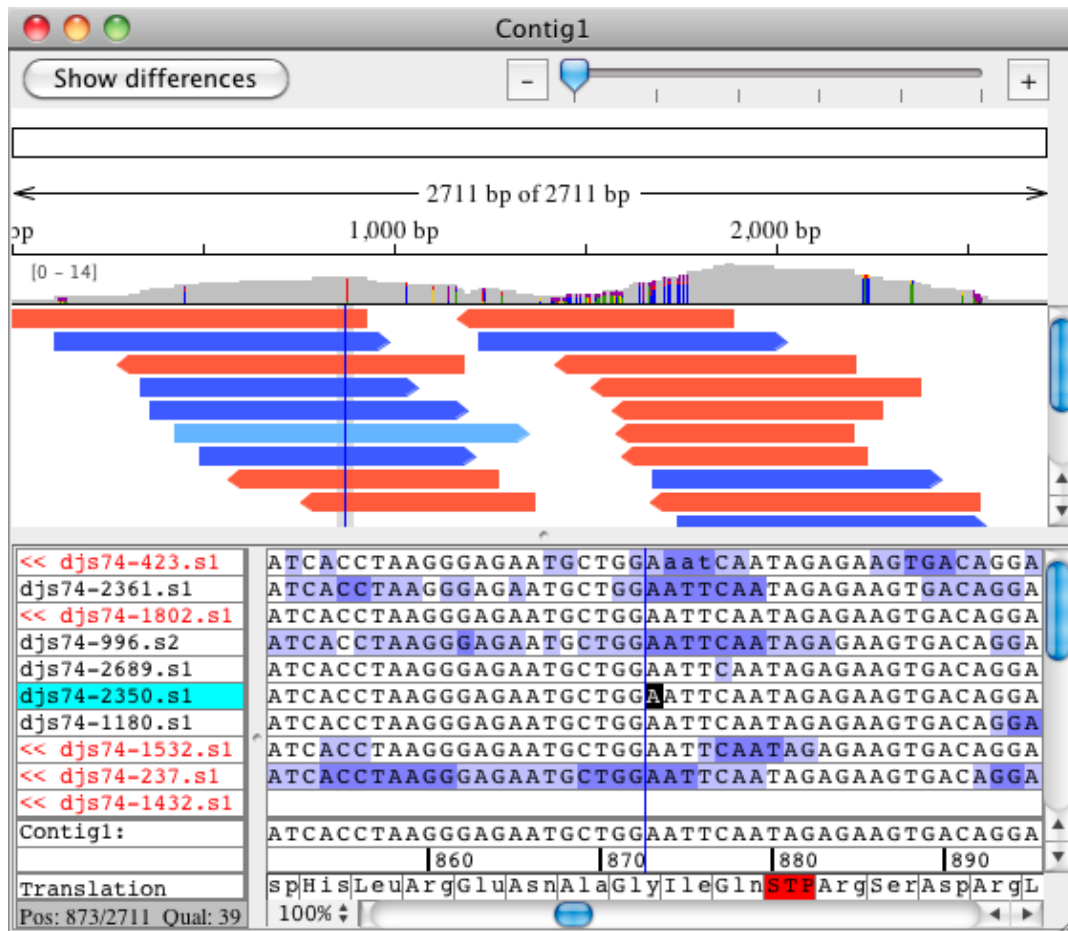


The picture shown above is pretty typical: a bit of low-quality sequence at the beginning, followed by rapidly increasing sequence quality and then several hundred bases where the quality is mostly very high, except for some short problem regions. If you open a trace view window for the same sequence, then clicking somewhere in the quality view will reposition the trace to the same region. This allows you to quickly check out problem regions in the sequence - try it out!

### Contig View Window

The contig window shows detailed information about contigs - a graphical overview about how the samples are aligned or a difference table for the alignment, the aligned bases, and the protein translation of the consensus sequence. It is also possible to show a [phylogenetic tree](#) to the left of the sample names.

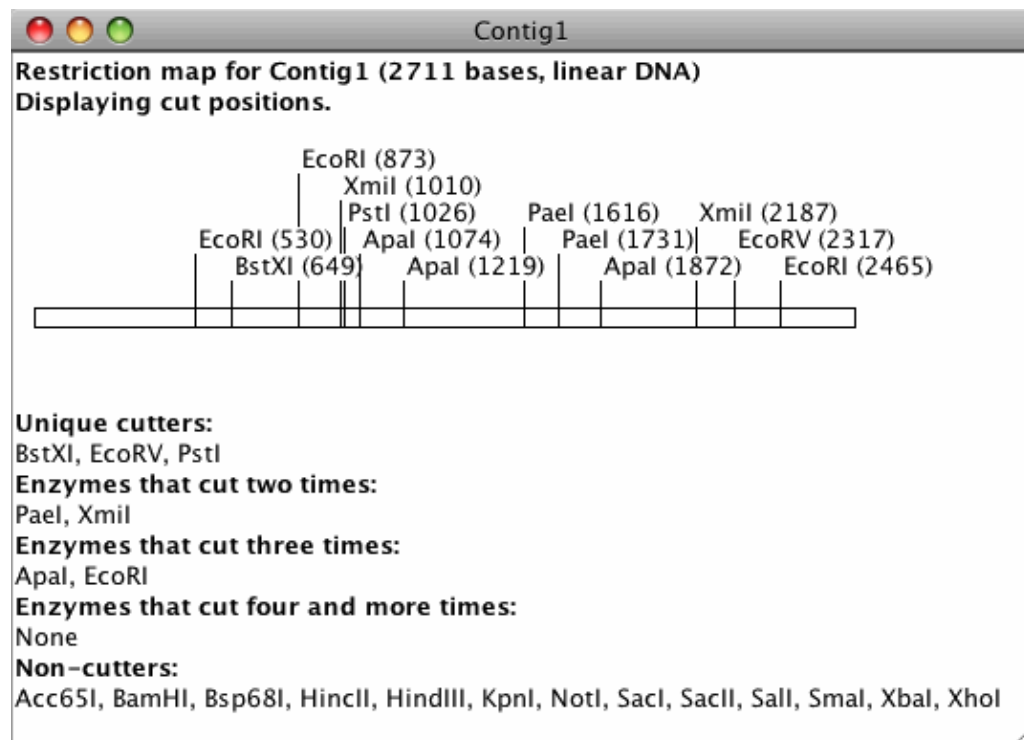
# CodonCode Aligner User Manual



You can edit sequences directly in the contig window, or select samples and then open trace, base, or quality windows for the sample. You can also set if and how the protein translation is shown using the "Protein translation" sub menu in the "View" menu.

## **Restriction Map Window**

The restriction map window shows a restriction map for the selected sample or `contigaligner_features.htm`.



You can change the display options and the enzymes used to generate the restriction map.

## Customizable Toolbars

All the views described above have the option of showing a customizable toolbar. The toolbar contains buttons which provide quick access to the different features in CodonCode Aligner without having to go through the menu. You can customize the toolbar for each view either through the [toolbar preferences](#) or directly through a popup menu from each toolbar. When using the popup menu, you can remove a selected button by choosing "**Remove Item**" from the popup. To add buttons to the toolbar using the popup menu, choose "**Customize Toolbar...**" from the popup, which will open the toolbar preferences for you. The toolbar buttons can be shown as text, as icons or as text & icons, which can also be set directly by using the popup menu or through the toolbar preferences.

You can hide the toolbar by selecting "**Toolbars > Hide Toolbars**" in the "**View**" menu. If you do not see the toolbar at the top, select "**Toolbars > Show Toolbars**" in the "**View**" menu to make it visible.

## Regions of Interest

One concept in CodonCode Aligner that deserves a closer look is the ability to define "regions of interest", more often called "features". Aligner lets you define what is of interest to you, for example low-quality consensus regions, discrepancies, mutation tags, and so on. You can then use this definition to get an overview of all the features in a contig or the entire project in a [feature view window](#), or to quickly [navigate](#) from feature to feature, or to [export features](#) to text files.

For more information about "region of interest" features, check the [feature help page](#).

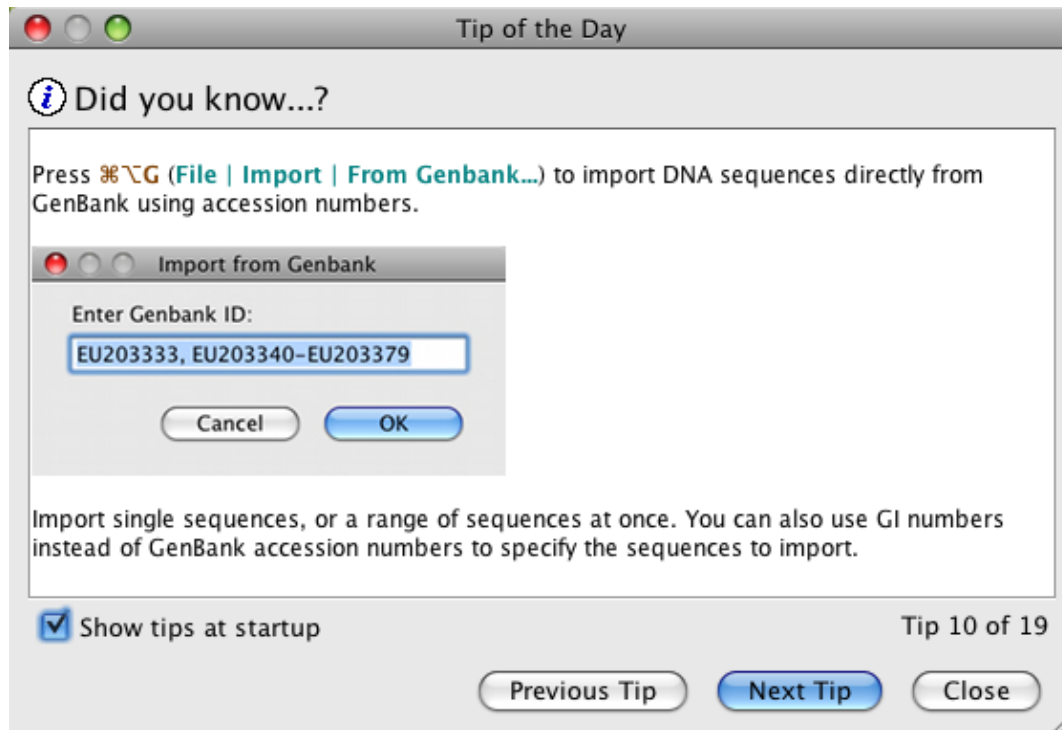
## Take the Quick Tour!

To quickly learn about the most important features in CodonCode Aligner, we suggest that you take the "Quick Tour", available online at <http://www.codoncode.com/aligner/quicktour/>. It takes about an hour to complete, but will probably save you quite a few hours, for example by showing you some very useful features that you otherwise might miss.

You can access the Quick Tour by selecting the "**Quick Tour**" menu item from the "**Help**" menu in CodonCode Aligner. This will open a new web browser window, and point it to the start of the quick tour. Alternatively, you can look at the Quick Tour in Adobe PDF format, which was installed as "QuickTour.pdf" in the directory where you installed CodonCode Aligner (typically "/Applications/CodonCode Aligner/" on OS X, and "C:\Program Files\CodonCode Aligner\" on Windows; you may not see the ".pdf" extension, depending on your system settings).

# Usage Tips for CodonCode Aligner

These tips are short explanations of features in CodonCode Aligner that are often overlooked. Tips are shown automatically when CodonCode Aligner starts. Here is an example:



If you do not want to see the tips every time at when CodonCode Aligner starts, simply uncheck the "Show tips at startup" checkbox in the "Tip of the Day" dialog.

You can also look at the tips by selecting the menu item "**Tip of the Day...**" from the "**Help**" menu.

# Quality Values In CodonCode Aligner

CodonCode Aligner was designed to make optimal use of base-specific quality values, both for pre-processing and for sequence assembly. Accurate, base-specific quality values that are linked to base calling accuracy were first introduced by the program [PHRED](#) (that's why they are often called "Phred qualities" or "Phred scores"). While you can do many things in Aligner even if your sequences do not have quality scores, we strongly suggest that you use sequences with quality scores.

For sequence traces that do not have qualities, Aligner allows you to call bases with PHRED to get PHRED base calls and quality scores, as described in the "[Base Calling with Phred](#)" section .

The following sections briefly [explain](#) the relation between quality scores and error probabilities; which files Aligner can [read](#) quality scores from; and how to [view](#) quality scores in Aligner.

## Quality values explained

Phred quality values represent the probability of error for each base call. The quality value  $q$  assigned to each base call is defined to be

$$q = -10 \times \log_{10}(p)$$

where  $p$  is the estimated error probability for the base call. As indicated below, a base call with a 1 in 100 probability of being incorrect will be assigned a quality value of 20.

| <u>Quality Value</u> | <u>Error Probability</u> |
|----------------------|--------------------------|
| 10                   | 1 in 10                  |
| 20                   | 1 in 100                 |
| 30                   | 1 in 1000                |
| 40                   | 1 in 10000               |
| 50                   | 1 in 100000              |

For additional details about Phred quality scores, you can read the articles listed [below](#).

## Where quality values come from

The first widely used program to produce highly accurate quality scores for each base call was the program [PHRED](#). Since then, a number of other base calling programs have been introduced and modified to also produce Phred-like quality scores. The two most common file types with quality scores are:

- SCF ("Standard Chromatogram Format") files produced by **PHRED**.
- ABI chromatogram files (.ab1 files) **processed with the KB base caller** (older ABI chromatogram files, and files produced with the ABI base caller, do not contain quality scores)

Chromatograms SCF format may or may not contain quality values, depending on which program generated the SCF files. SCF files generated by PHRED contain quality scores, but some other programs (like some versions of Sequencher) do not write quality information into SCF files.



When importing Phrap assemblies that were generated with quality scores, Aligner will read the quality information from the corresponding "PHD" files, regardless of whether or not the chromatogram files for the assembly were in ABI- or SCF format.

When reading sequences from FASTA files, Aligner will also read quality scores from ".qual" files - text files in FASTA-like format that contain quality scores. The name of the ".qual" file must be the same as the name of the FASTA file, with ".qual" appended (for example "seq.fasta" and "seq.fasta.qual"). The quality file must be in the same directory as the FASTA file.

Note that not all ABI chromatogram files have quality scores. Only ABI chromatograms processed with the KB base caller have quality scores; ABI chromatograms processed with the older ABI Basecaller do not have quality scores. For questions about the KB base caller, please read the "[KB Base Caller Frequently Asked Questions](#)" brochure.

When importing chromatogram sequences that do not have quality scores, Aligner will assign artificial quality scores to all bases, as described in the next section. You can identify sequences without quality scores quickly in the project view - the "Quality" column shows "0" as the number of high-quality bases for such sequences. To assign sequence quality scores to chromatograms without qualities, you can [base call](#) the samples with PHRED (which may require a separate license for PHRED).

## Artificial qualities

There may be times where you need to use sequences that do not have quality values, for example Genbank sequences. When importing such sequences into CodonCode Aligner, Aligner will assign artificial qualities to all bases in these sequences. The default value for artificial qualities is 15, but you can change this in the "[Open & save](#)" preferences.

If you import sequences into Aligner which have quality scores that seem artificial, for example because all scores are 0 or -1, or all scores have the same value, Aligner will also assign artificial quality scores.

## Gap qualities

Quality values are associated with base calls, so gaps do not really have quality values. However, since it is often convenient to have some quality value even for gaps, Aligner does assign qualities to gaps; the assigned quality is the average of the two bases on either side (with gap characters and edited bases being ignored). For bases at the start or end of the sequence that have only one neighboring base, the quality value of this base is used.

## Viewing qualities

In CodonCode Aligner, you can "see" quality values in a variety of ways:

- to get an **overview** of the quality in a sample, select the sample in the project view, and then choose "**Qualities**" in the "**View**" menu
- to get information about the quality **at any specific base**, you can click on the base in the base view window or the trace view window; the quality of the selected base is shown at the bottom of the window
- you can also choose to have the **background behind the bases** in the base view, trace view, and contig view windows indicate the quality, using either a continuous scale or a three-color scheme ("the

good, the bad, and the ugly" :-); you can set this in the "Base colors" preference panel.

## Additional Reading

R. Durbin and S. Dear, 1998 "Base Qualities Help Sequencing Software" *Genome Research*. 8: 161-162. ([Available](http://www.genome.org/content/vol8/issue3/) online at <http://www.genome.org/content/vol8/issue3/>)

B. Ewing B, L. Hillier L, M.C. Wendl and P. Green. "Base-calling of automated sequencer traces using Phred. I. Accuracy assessment." *Genome Research*. 8: 175-85. ([Available](http://www.genome.org/content/vol8/issue3/) online at <http://www.genome.org/content/vol8/issue3/>).

B. Ewing and P. Green, 1998 "Base-Calling of Automated Sequencer Traces Using Phred. II. Error Probabilities." *Genome Research*. 8: 186-198. ([Available](http://www.genome.org/content/vol8/issue3/) online at <http://www.genome.org/content/vol8/issue3/>)

P. Richterich, 1998. "Estimation of Errors in "Raw" DNA Sequences: A Validation Study" *Genome Research*. 8: 251-259. ([Available](http://www.genome.org/content/vol8/issue3/) online at <http://www.genome.org/content/vol8/issue3/>)

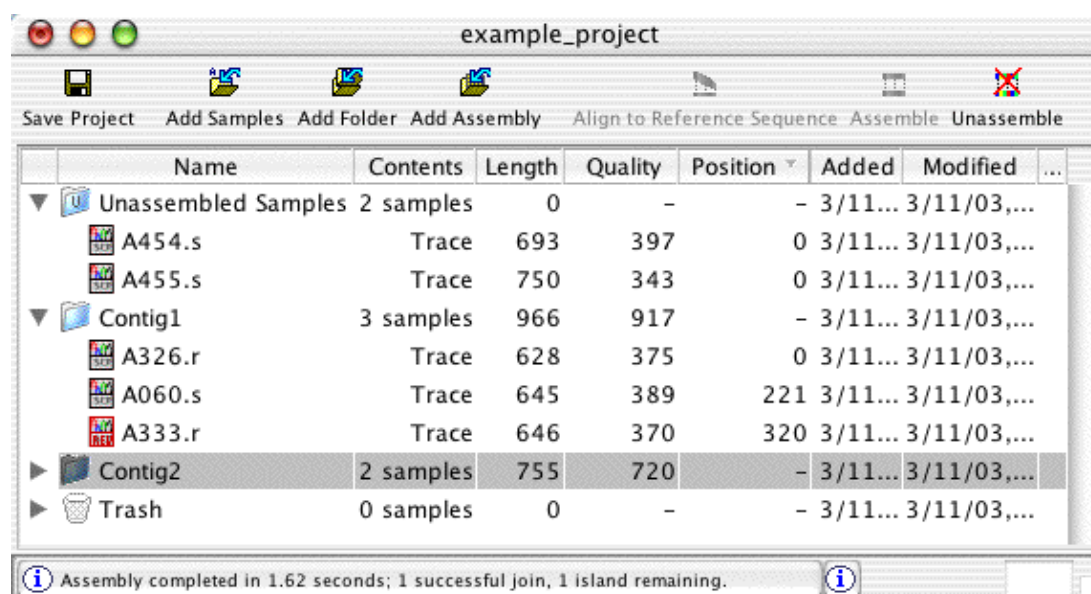
# Aligner Projects

An Aligner project contains all the information for a project you are working on - the samples you added, any contigs you may have assembled, and so on. Each project is contained in one file (ending in ".ccap"), which is created by CodonCode Aligner when you first save a project.

After an Aligner project is created, sample files can be added.

When alignments or assemblies are performed, the resulting consensus is stored in the project upon saving.

Only one project can be open at any point in time. The contents of a project are shown in the **project window**:



In the project window, samples are treated like files, and contigs are treated like folders (directories) in the Finder on MacOS or the Explorer on Windows. Two other folders also exist in every project: the "Unassembled Samples" folder, and the "Trash". Any samples that have been added to a project, but not yet assembled or deleted, get added to the "Unassembled Samples" folder. If you don't want a sample in your project anymore, you can move it to the Trash by selecting the sample in the project window, and then selecting "Move To Trash" in the "Edit" menu. If you later change your mind, you can move samples from the trash back to the "Unassembled Samples" folder.

Within the project view, you can also use **drag and drop** to move samples and even contigs around. You can drag samples from the "Unassembled Samples" folder to the "Trash", and the other way round; you can drag samples in contigs or entire contigs to the trash or to the "Unassembled Samples" folder; and you can add samples to existing contigs by dragging them from the "Unassembled Samples" folder onto the contig you want to add them to (of course, the samples then align with the contig; for more details, check the "[Assembly and Alignment](#)" help section).

Please note that CodonCode Aligner version 4.0 introduced a new project format. Older CodonCode Aligner versions saved projects in a separate folder that contained several files and sub-directories. CodonCode Aligner version 4.0 and newer saves projects in a single file with the file extension ".ccap" (which you may

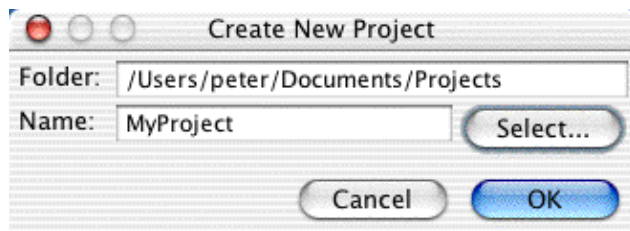
not see depending on your system settings). You can export Aligner projects in the old format using 'Export -> Aligner Project (Old Format)...' form the File menu.

# Working with Aligner Projects

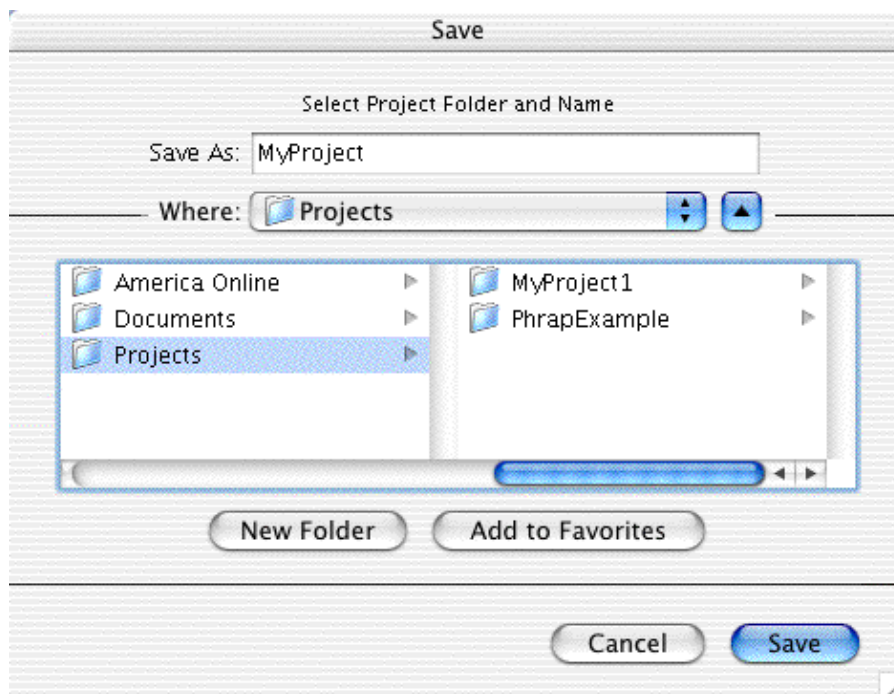
## Creating New Projects

To create a new project, choose **"New Project"** from Aligner's **"File"** menu. Aligner will create a new project, and show a project view window for it. The project will be empty, except for the "Trash" and the "Unassembled Samples" folders. You can specify where a project will be saved when you first save the new project.

If you prefer to set the project path when creating a new project, you can change the [Open & Save Preferences](#) accordingly. Now, when creating a new project, Aligner will show the "Create New Project" dialog:



The simplest way to set the project name and location is to click on the **"Select"** button. This will bring up a standard **"Save As.."** dialog:



After clicking "Ok", a new project window will be opened, which contains no samples, only the "Unassembled Samples" and the "Trash" folders.

## Opening Existing Projects

Choose **"Open..."** from the **"File"** menu to open a project you have already saved. Navigate to the directory your project is in, and select the project file. The project file has either a ".ccap" or a ".proj" extension (which you may not see, depending on your system settings). This will read all the contents of the project as it was last saved, and show the contents in a new project window. Projects created with CodonCode Aligner 4.0 or newer are saved in a new format and have the ".ccap" extension as opposed to older projects who have the ".proj" extension.

Alternatively, you can also use the **"Open Recent"** submenu in the **"File"** menu. This will show the projects you worked with recently. If a project is no longer available, for example because you deleted or renamed it's folder, then it will be grayed out in the **"Open Recent"** submenu.

## Saving Projects

Choose **"Save Project"** from the **File** menu to save your project (if the project is a new project for which you have not set a name and location before, you will be prompted to do so now).

To save a copy of an existing project under a different name and/or at a different location, choose **"Save Project As..."** from the **File** menu. The **"Save Project As..."** command will always create a new project file *(if you try to create a project with the same name as an existing project in this location, you will get a warning dialog)*.

**It's a good idea to save projects on a regular basis**, and especially before major processing steps like end clipping, vector trimming, or assembly.

If you are using CodonCode Aligner in **Demo** mode, you can only save projects that (a) do not contain any contigs, and (b) where you have not used any of the advanced Aligner functions like end clipping, sequence assembly, and so on.

## Closing Projects

Choose **"Close Project"** from the **File** menu to close the currently open project and all its windows. Aligner will first check if you made any changes to the project since the last time the project was saved. If any changes were made, Aligner will ask you if you want to save the changes before closing the project. If you answer "Don't save", all changes you have made since the last save will be lost.

# Adding Sample Files to Aligner Projects

CodonCode Aligner can import sequences from a variety of different [file formats](#), including chromatogram files in ABI and SCF format and text files in FASTA, FASTQ, SAM, Genbank, NBRF/PIR, PHD, or plain text format. Files in FASTA, FASTQ, SAM, Genbank, or NBRF/PIR format can contain multiple sequences per file.

You can add sequences to Aligner projects by **dragging and dropping** files onto Aligner project views, or by using one of several **"Open"** and **"Import"** options in the **"File"** menu:

1. by [opening single sample files](#)
2. by [adding samples from several files](#)
3. by [adding a subset of samples](#)
4. by [adding entire folders of samples](#)
5. by [adding an entire assembly or project](#)
6. by [importing from Genbank](#)
7. by [creating new text sequences](#)

Aligner will always create copies of the samples you import. Keep in mind, though, that these copies will be created only when you save the project, so save on a regular basis! *For files that have pointers to associated files, Aligner will also try to locate and import the related files (for example, PHD files and FASTA files may point to a chromatogram file in ABI or SCF format that contains the trace data).*

## Opening Single Sample Files

To add a single sample to a project, select **"Open..."** from the **"File"** menu. This will show the standard "Open" file dialog. Select the file that you want to add to your project, and click "OK". The file will be read, and the sample(s) in it will be added to the "Unassembled Samples" folder. Next, Aligner will open one or more views for the sample(s) you just added, based on your settings for which views to open when double-clicking on a sample in the [Double Clicking Preferences](#).

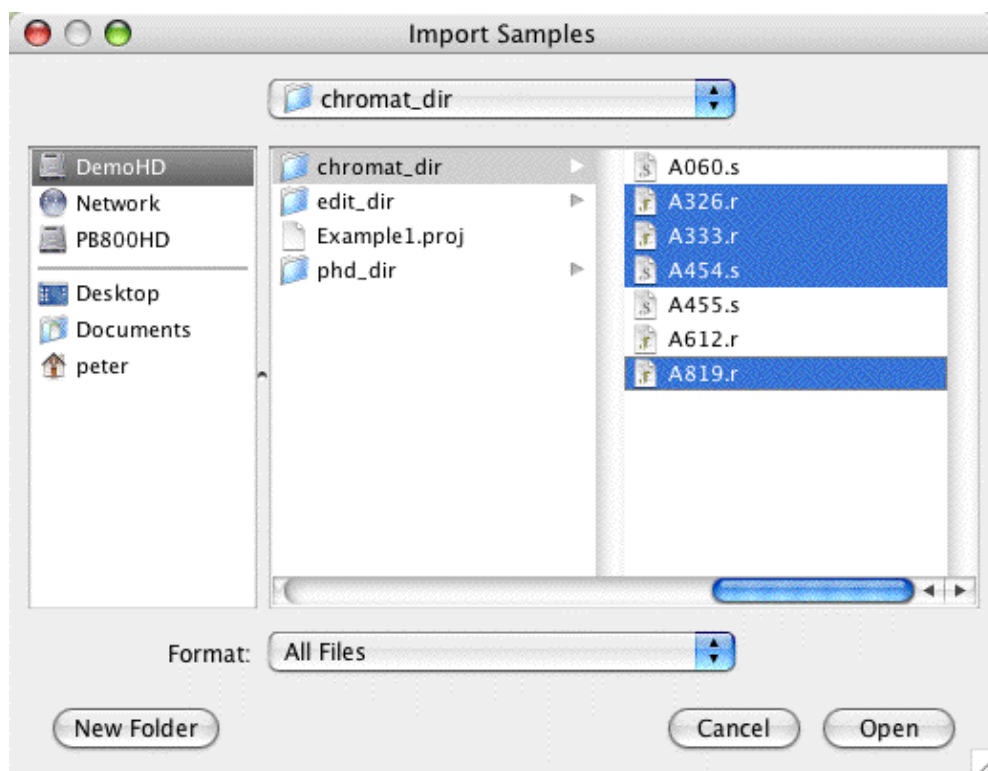
*You can also add files or folders to projects by drag and drop from Finder or Explorer windows onto Aligner project views.*

## Adding Several Sample Files

To add samples from a few files to a project, select **"Import"** -> **"Add Samples..."** from the **"File"** menu. This will show an "Open" file dialog. Select the files that you want to add to your project, and click "OK".

- To select several files in a row, click on the first file, and then keep the "shift" key pressed while clicking on the last file.
- To select several files that are not right next to each other, keep the "control" key (OS X: the "command" key) pressed while clicking on files.

The screen shot below shows an example of the "Import Samples" dialog with four selected files:

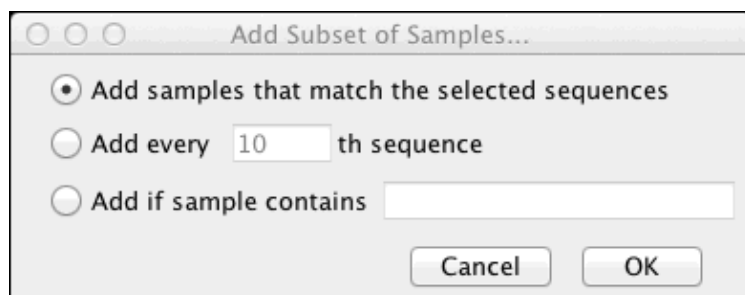


When you are done making your selection, press the "Open" button. CodonCode Aligner will read the files you selected, and add the samples to the "Unassembled Samples" folder.

If any problems were encountered during import, Aligner will tell you so by showing a dialog box. For example, if you already have a sequence with the same name as an imported sequence in the project, you will be given a choice to ignore the duplicated sequences, or to automatically rename them.

## Adding a Subset of Samples

To add only a subset of samples in a file to a project, select **"Import" -> "Add Subset of Samples..."** from the **"File"** menu. This will show the following options dialog:





You have three options for adding a subset of samples:

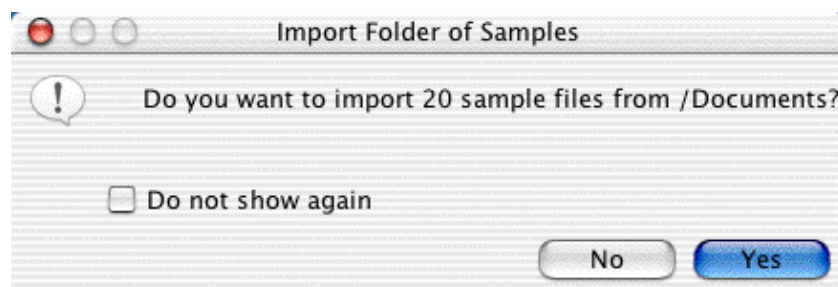
1. **"Add samples that match the selected sequences"**: This option will import only those samples from the file that match the selected samples (specifically, imported sequences must have at least five 16-base words in common with one of the selected sequences). *One example where this option can be used is to select only chloroplast sequences from a Next Gen sequencing experiment that contains whole plant DNA.*
2. **"Add every \_\_\_\_th sequence"**: If you select this radio button, only every  $n$ th sequence of the file will be imported, which can be useful to get an overview about the samples in your file.
3. **"Add if sample contains"** lets you specify a string that must be present in a sample - either in the name, or the sequence, or in the comments. Samples that do not contain the given string will not be imported.

Clicking **"OK"** will then show the "Import Samples" dialog where you can select the file that contains your samples. CodonCode Aligner will add the samples that match your subset criteria to the "Unassembled Samples" folder.

## Adding Entire Folders of Sample Files

To quickly add many sequence files to a project, it is easiest if you first gather all the files into one folder, and then import the entire folder. To import sequences from all files in a directory, select **"Import"** -> **"Add Folder..."** from the **"File"** menu. This will show the standard "Open" file dialog. Navigate to the directory you want to import, select any file in this directory, and then click **"OK"**.

Aligner will count the files in the directory, and then ask you if you really want to import all the files:



If you click **"Yes"**, Aligner will proceed to import sequences from all files in the directory. This may take a while if the directory contains many files! All imported sequences will be added to the "Unassembled Samples" folder.

**Note:** Aligner will try to be reasonable when importing all files in a folder, and not import some files that you probably did not really want to import. For example, Aligner will not import hidden files (like files where the name starts with a period); also, when importing an ABI runfolder, Aligner will import the .abi (or .ab1) files, but ignore all .abi.seq files if the corresponding .abi file was present.

## Adding Entire Assemblies

To add an entire assembly to your project, select **"Import"** -> **"Add Assembly..."** from the **"File"** menu. This will show the standard "Open" file dialog. Next, select the assembly file, which can be:

- an Aligner project file (*the file name must end in ".ccap" or ".proj"*)

- a "CAF" (Common Assembly Format) file produced by Sequencher (*the file name must end in ".caf"*)
- assemblies (ACE files) generated by Phrap (*the file name must end in ".ace", ".ace.1", ".ace.2", etc.*)

The assembly files **must have the correct extension** - either .ccap, .proj, .caf, or .ace; ACE files can also have a number appended, for example ".ace.1" (*Note that, depending on your system settings, you may not be able to see the extension*).

After you selected the assembly file, CodonCode Aligner will read the file, and any associated files like chromatogram files. If any of the samples or contigs in the imported assembly have the same names as files that are already in your project, Aligner will give you a choice to either rename the files, or to cancel the import.

## Importing Sequencher Projects (CAF Files)

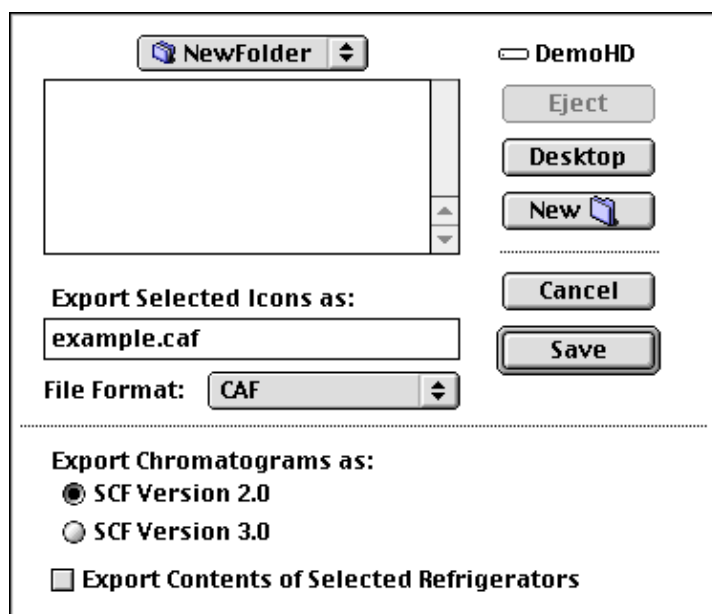
Many current and former users of the assembly program Sequencher have expressed the desire to import existing Sequencher projects into CodonCode Aligner. While CodonCode Aligner cannot read Sequencher project files directly, it is easy to export Sequencher projects in a file format that CodonCode Aligner can read, the "CAF" file format (for "Common Assembly Format").

## Exporting Sequencher Projects as CAF Files

To import a Sequencher project into CodonCode Aligner, you must first export the project from Sequencher in CAF format, as follows:

1. Open the project in Sequencher
2. Select the reads and contigs you want to export (use "Select All" from the "Select" menu to export the entire project)
3. Go to the "**File**" menu, and select "**Export => Selection As Subproject**". This will open a "Save" dialog.
4. Click on the "File Format:" pulldown menu, and select "CAF".
5. Use the "New folder" button or icon to create a new folder to export your project to.
6. Choose a name for the exported file. **Make sure that the name ends with ".caf"**.

Here is an example of what the dialog looks like with Sequencher 4.1 on Mac OS 9:



When you click "Save", Sequencer will export your selection to a text file in CAF format. In addition, Sequencer will create chromatogram files in SCF format for all reads that have traces (this is why we suggested to create a new folder in step 5 above).

You can now quit Sequencer, and import the CAF file you just exported into any open CodonCode Aligner project, using "Import => Add Assembly". Please keep a few things in mind when importing CAF files:

- Sequences in old Sequencer projects often do not have base-specific qualities, but many functions in CodonCode Aligner either require quality scores, or work better with quality scores. For unassembled samples that have chromatogram traces, you can use "[Call Bases](#)" to run Phred on the imported samples, and thereby get quality scores.
- Consensus sequences generated in Sequencer are not quality-based, and therefore likely to contain more errors and ambiguities than consensus sequences built in CodonCode Aligner. You can choose to have CodonCode Aligner re-build the consensus sequences when importing in the [consensus method preferences](#).
- CodonCode Aligner's support for CAF files is currently limited to CAF files exported by Sequencer; Aligner will generally not be able to read CAF files produced by other programs.

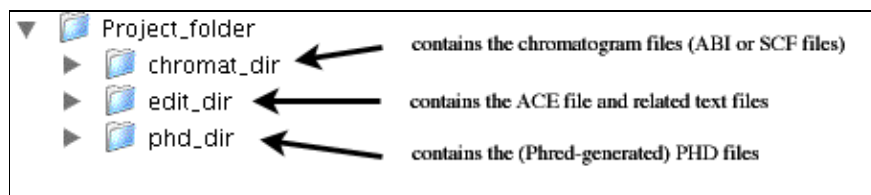
If you encounter any problems when importing CAF files into CodonCode Aligner, please send a description of your problem together with the CAF file to [support@codoncode.com](mailto:support@codoncode.com).

## Importing Phrap Assemblies

To import a Phrap assembly into CodonCode Aligner, select have the Phrap-generated ACE file in the file dialog you see after choosing **Import => Add Assembly** from the **File** menu (you must have a project open for this option to be available). In addition to the ACE file, you should also have the PHD and SCF files for the assembly, located in exactly the same directory structure as Consed would expect, as described below.

When importing Phrap assemblies, all the contigs that Phrap has created will remain intact, and detailed information like unaligned ends and tags will be reserved. Aligner will read the assembly information from the .ace file, and try to get additional information from the other files typically used with Phrap - the .PHD files in the "sister" folder "phd\_dir", and the chromatogram files in the "sister" folder "chromat\_dir". An

overview of the typical directory structure used with Phrap assemblies is as follows:



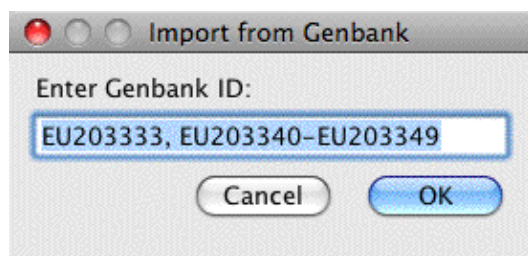
## Tags in Phrap assemblies

Phrap assemblies can contain tags, which are especially important when you are using PolyPhred. Aligner will import and preserve tags, and display tags based on your highlighting preferences. You can add notes to existing tags, and edit such notes. To do so, open a contig view, select a base that contains a tag, and then right-click (OS X: control-click) on it to display the popup menu. The first item in the popup menu will be "Display Tag..." (but only if you clicked on a base with a tag).

You can also [view](#) all tags for a sample in the ["Sample information" dialog](#), and [add tags](#).

## Import from GenBank

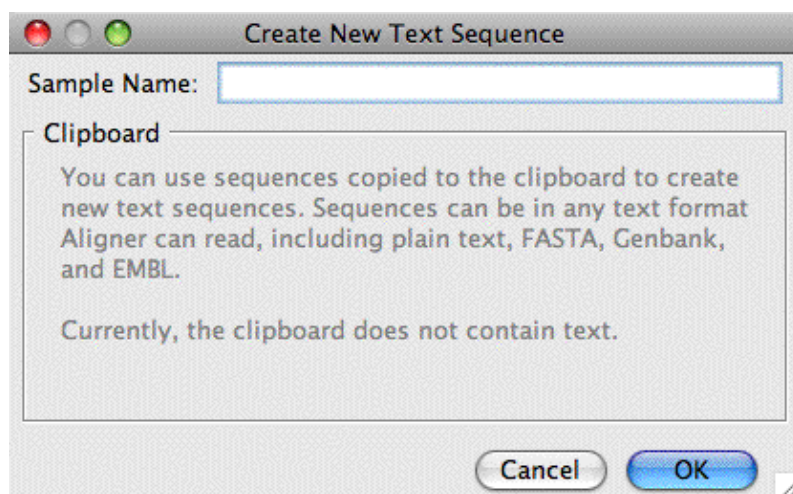
You can import DNA sequences directly from GenBank by using GenBank accession numbers (like EU203333) or GI numbers (like 166029874). To import one or more sequences from GenBank, go to the "File" menu, move to the "Import" submenu, and select "From Genbank...". In the dialog that pops up, enter the GenBank accession number(s) or GI number(s) you want to download. You can also use ranges, as shown in the next screen shot:



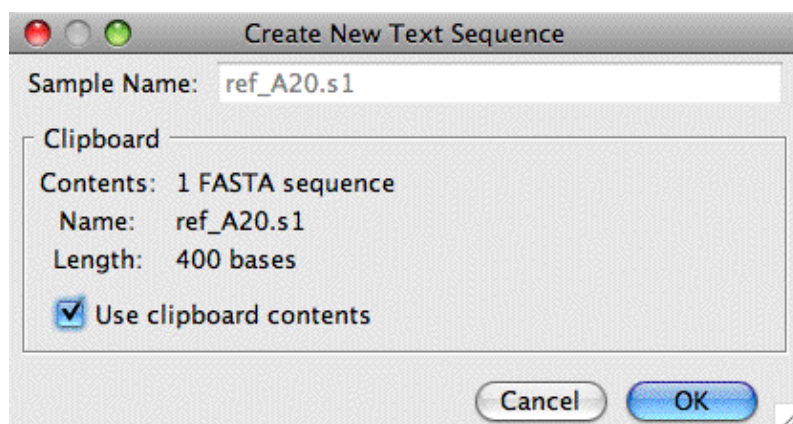
When you click "OK", CodonCode Aligner will connect to the NCBI web site, and try to import the selected sequence(s) as text sequences into your project. Of course, you need to have a project open, and you need to be connected to the internet. Also, the accession numbers or GI numbers you enter must be valid.

## Creating new text sequences

You can create new text sequences by selecting "New Text Sequence..." from the "File" menu (you need to have a project open to do this). This will show the following dialog:



One use of this option is to add new sequences from the contents of the clipboard - for example, sequences that you copied from a web browser. CodonCode Aligner will analyze the clipboard contents before showing the dialog; if the clipboard contains a text sequence in a format that CodonCode Aligner can read, the dialog will look like this:



When you select the checkbox labeled "Use clipboard contents", and then press "OK", CodonCode Aligner will create a new sequence, and open the base view for this sequence. If the clipboard contains multiple sequences in a format that allows multiple sequences per file (for example FASTA or Genbank format), Aligner can also create multiple new samples from the clipboard contents.

## Compatible File Formats

The following sample file types can be read by Aligner:

- **SCF files** - Standard Chromatogram Format, commonly available and generated by Phred, LI-COR software, and many other programs. This is the preferred format, especially if the SCF files were generated by Phred. It is also the format used by Aligner to store chromatogram information. (*Note: Sometimes, SCF files have the file extension .scf; this will be hidden in Windows even if you have your preferences set to display file extensions. However, Aligner will recognize this file extension.*)
- **ABI chromatogram files** - Files generated by systems from Applied Biosystems software. Typically have the extension .abi or .ab1 (but Phred-generated SCF files will typically have the same name and extension!). *Note that older ABI files may not contain base-specific quality scores; after importing*

*ABI files that do not have quality scores, you should first [base call](#) to get quality scores.*

- **FASTA files** - text files with one or more DNA sequences, where each new sequence starts with a line that starts with a '>' sign, followed by the name of the sequence.  
If the FASTA header line also contains the name of corresponding PHD or SCF files, Aligner will try to read these files, too.
- **FASTQ files** - text files with one or more DNA sequences and their qualities. A FASTQ file normally uses four lines per sequence. The first line starts with a '@' character and is followed by the name of the sequence. Line 2 contains the sequence. Line 3 begins with a '+' character. Line 4 encodes the quality values for the sequence in Line 2, and must contain the same number of symbols as letters in the sequence.
- **GenBank files** - text files exported from Genbank, and many other programs. Genbank files are often used for the reference sequence in re-sequencing and mutation detection projects.  
When reading Genbank files, Aligner will read all feature annotation, including the "/codon\_start" tag (this tag indicates the number of the base after the CDS start which start the first whole codon, and can either be 1, 2, or 3). *In Aligner, the "CDS" region is shown as a "codingSequence" tags, and the "/codon\_start" annotation as a "codonStart" tag.*
- **EMBL files** - text files exported from ENSEMBL and many other programs in EMBL format. Like Genbank files, EMBL files can be used for the reference sequence in re-sequencing and mutation detection projects. Aligner will read all feature annotation, similar to the way described for Genbank files.
- **GFF files** - text files with annotation information created by various programs. Typically, annotation from GFF files will be added as tags to samples with the names in the GFF file. If no sample with the name in the GFF file is in the project, and the GFF file contains a sequence in FASTA format, then new samples will be created from the FASTA sequence, and tags for the annotation in the GFF file will be added.
- **NBRF/PIR files** - text files in NBRF/PIR format, a simple format that can contain multiple sequences per file. Sequences in NBRF/PIR format may contain gaps in the sequences as well as at the start and end of sequences. When importing sequences from NBRF/PIR files, CodonCode Aligner will check if the imported sequences contain gaps. If the sequences contain gaps, and are all of the same length, Aligner will check the sequence names in the project to see if you are re-importing contigs that have been edited with an external sequence editor like MacClade, and offer the option to update existing contigs. For more information, read the ["Roundtrip Editing" help page](#).
- **PHD files** - text files that contain information about base calls and quality scores. PHD files are typically created by PHRED, but a number of other programs (including Aligner) can also write PHD files.  
PHD files typically contain the name of the chromatogram file they refer to, which Aligner will then also read to get the trace data.
- **Plain text format** - text files that have only DNA sequence without any annotation. Aligner will read all base letters ("ACGTUacgtu") and IUPAC ambiguity symbols, and ignore any spaces, line endings, and numbers. Plain text files cannot contain any "binary" characters; when creating text files in programs like Microsoft Word, make sure to save the files as "Plain text ", not in native formats (like "Microsoft Word Document (.doc)").
- **SFF files** - binary files from 454 sequencers that contain multiple sequences and experimental ("flowgram") data. Please note that CodonCode Aligner only offers **limited** support for 454 data. You can only import small 454 projects, but **not** entire runs with hundreds of thousands of sequences. Projects with about 20,000 to 40,000 samples from SFF files may be fine, provided your computer has at least 2 GB of memory installed. On OS X, working with many samples will require that you increase the amount of memory that CodonCode Aligner can use in the ["Memory" preferences](#).
- **SAM files** - text files that are used for storing large nucleotide sequence alignments and their information.

In addition to these sample files, CodonCode Aligner can read sequence assembly files in ".ace" format produced by Phrap or Consed, and assemblies in CAF format produced by Sequencher, as described above.

## Sample Names

When reading chromatogram files, Aligner will use the name of the file as a sample name. Since version 1.4.0, sample names can contain spaces, accented characters, and so on. However, if your sample names have "funny" characters like accents or umlaute, keep this in mind:

- The names may not be drawn correctly in all views (due to minor bugs in Java)
- On Windows, importing a folder of samples with special characters in their names may not work (Aligner may report that it cannot find some of the samples). Similar, adding samples to a project by drag and drop may not work if the sample name contains unusual characters. However, you should always be able to add samples using "Import -> Add Samples".
- When exporting samples, Aligner may replace unusual characters in the sample names with underscores (depending on your settings).

Some files contain the sample name in the file (e.g., FASTA files). When this is the case, the name specified in the file is used as the Aligner sample name. When a file contains multiple samples, the sample names **must** be specified in the file.

Within a project, **sample names must be unique**. If a sample name in a file already exists in the project, the new sample is renamed so that the sample name is unique. Samples are renamed by appending an underscore and a unique number (e.g., name\_1).

To **rename samples**, you can use the ["Sample Information" dialog](#). Alternatively, you can first select the sample in the project view, and then clicking on the sample name again (similar to the way you would rename files in Finder windows on OS X, or Explorer windows on Windows).



# Organizing Samples And Contigs In Folders

Aligner projects always contain the "Unassembled Samples" folder and the "Trash" folder. However, if your projects contain a lot of unassembled sequences or many contigs, you may want to organize the sequences a bit better. CodonCode Aligner allows you to create folders in projects, to which you can move samples and contigs by drag and drop or by using the "Move To..." menu item.

## Creating Folders

To create a new folder for organizing unassembled samples, select "**New Folder**" from the "**File**" menu. This will show a dialog where you can name the new folder.

Folders created this way can hold unassembled samples, contigs, or a mix of samples and contigs.

## Moving Samples and Contigs to Folders

You can move samples or contigs to and from folders in several ways:

- By **drag and drop**:  
Select the folders or samples you want to move in the project view, and drag & drop them onto the target folder.
- Using "**Move to...**" in the "**Edit**" menu:  
Select the samples you want to move, then choose to "**Move to...**" from the "**Edit**" menu. In the dialog that appears, select the target folder that you want to move the samples to, then press "OK".
- Using "**Move to...**" in the popup menu:  
Select the sample or samples you want to move in the project view, and then right-click (OS X: control-click) on one of the samples. From the popup menu, select "Move to..."

To move contigs from folders back to the same level as Unassembled Samples and the Trash, select the contig, choose "**Move to...**" from the popup or "**Edit**" menu, and then select "(top level)". You can also drag and drop contigs in folders onto empty space in the project view to see the same menu.

You cannot move unassembled samples to the top level, and the "(top level)" option will not be available if your selection contains unassembled samples.

## Renaming Folders

To rename a folder you created, click on the folder in the project view, wait a little bit, and click on it again. You will see when the name becomes editable (it may take a second or so). When you are done editing the name, press enter, or click on any other item in the project view.

If you change your mind about renaming the folder, you can press "escape" while you are still editing, or select "**Undo**" from the "**Edit**" menu right after you changed the name.

## Deleting Folders

To delete a folder that you created, first remove all of its contents, then:



## CodonCode Aligner User Manual

- Select the folder in the project view, then
- Choose "**Delete Folder**" from the "**File**" menu.

Only empty folders can be deleted. If a folder still contains samples or contigs, the "**Delete Folder**" menu item will be disabled. To delete a folder that contains samples, first move the samples to the trash, and then delete the folder.

You cannot delete the "Unassembled Samples" folder or the "Trash" folder.

# Removing Samples from an Aligner Project

To remove samples from an open project:

- Go to the project window
- Select the sample(s) you want to remove (*use shift-click to make continuous selections; use control-click (Windows) or command-click (OS X) to make discontinuous selections*)
- Select **"Move To Trash"** from the **"Edit"** menu (*or display the popup menu by right-clicking (OS X: control-clicking) , and choose **"Move To Trash"** from the pop-up menu*)

You can also select samples *in* contigs and move them to the trash. This will remove the samples from the contig. If removing a sample would leave a gap in a contig, thereby splitting a contig in two pieces, you would not be able to remove the sample (we plan to add the functionality to automatically split a contig in later versions...)

You can also move entire contigs to the trash this way. This will move the contig and all samples in it to the trash.

Moving samples to the trash does **not** delete them from your project. If you change your mind about a sample in the trash, you can select it, and then choose **"Move To Unassembled Samples"** (or "Move To...") from the **"Edit"** menu.

To permanently delete samples from you project, first move the samples to the trash, and then choose **"Empty Trash..."** from the **"File"** menu.

# Base Calling with PHRED

Aligner works best if sequences have quality scores - but what if you have sequence traces that do not have quality scores, for example ABI files? Well, you can simply run Phred to re-do the base calling and assign quality scores from within Aligner (Phred was developed by Phil Green and Brent Ewing at the University of Washington; [more about PHRED below](#)).

## How to call bases with PHRED from Aligner

To call bases for sequence traces with PHRED:

1. Select the samples in the project view.  
The sequences must be unassembled; you can also select the entire "Unassembled Samples" folder. If you want to call bases for samples that are already in a contig, you must first [unassemble](#) the contig.
2. Choose "Call Bases" from the "Sample" menu.

Depending on the number of samples you have chosen, base calling may take a while (up to a few seconds for each sample), and you will need to wait until it's done before you can do anything else.

## Prerequisites

To use base calling in Aligner, your sequences must have traces; any sequences from text files that do not have traces will be ignored.

Furthermore, base calling with PHRED requires that you have **PHRED installed** on the computer that you are running Aligner on. This can either be the workstation version of PHRED that is included with Aligner, or your own copy of PHRED.

### Using the workstation version of PHRED

When you install CodonCode Aligner, a workstation version of PHRED is installed in the "Phred-Phrap" folder inside the "CodonCode Aligner" folder. This workstation version can only be run from CodonCode Aligner, but is otherwise identical to PHRED. To use it, you must either have a valid trial license for Aligner, or a full (purchased) license that includes license permissions for the workstation version of PHRED.

CodonCode Aligner customers at academic and non-profit institutions can use the workstation version of PHRED free of charge; customers at for-profit institutions need to purchase a separate license for the workstation version of PHRED.

### Using your own copy of PHRED

If you already have a licensed copy of PHRED installed on the computer that you are using Aligner on, you can use this copy for base calling. First, though, you will need to tell Aligner where your copy of PHRED is installed; you can do this in the [base calling preferences](#).

Academic users who want to use their own copy of PHRED can obtain the source code for PHRED free of charge directly from the authors (with certain usage restrictions); please check [www.phrap.org](http://www.phrap.org) for details.

Users at for-profit organizations need to purchase a license for PHRED. Please check [www.phrap.com](http://www.phrap.com) for information about purchasing PHRED licenses from CodonCode.

### What Aligner does

In short, all that happens is that the base calls in the samples will be replaced with PHRED base calls, and PHRED's base-specific quality scores. But if you really want to know the details, here is what happens when you start base calling in Aligner:

1. Aligner will first create two temporary directories. These are created in the system's default temporary directory, for example in /tmp/ on OS X; the names of the directories will start with "bscl".
2. Aligner will write SCF files for each selected sample into the first folder, from which PHRED will read the data.
3. Aligner checks if all necessary entries are present in the Phred parameter file to process the traces you selected. *If any entries need to be added, Aligner will show a warning dialog, and then allow you to add the required entries using the Phred parameter file edit dialog as described [below](#).*
4. Next, Aligner will start PHRED, using the path to the program that you can define in the [base calling preferences](#) (the default is "workstation\_phred" in the "Phred-Phrap" folder inside the "CodonCode Aligner" folder). Aligner will also pass the location of the Phred parameter file, as defined in the [base calling preferences](#), to PHRED. Aligner will use the -id and -cd options to tell PHRED where the input files are, and where to write the result files to.
5. Aligner will wait for PHRED to finish. If you are base calling several hundred traces, this may be a good time for a coffee break. If it's just a few traces, it should take only a few seconds.
6. When Phred is done, Aligner will check to see if PHRED produced the expected number of result files; if not, Aligner will try to look at the PHRED output to find out what the problem was (see [below](#)). Aligner will also write the progress and error messages generated by PHRED into two files in the Aligner folder; the file names are "Basecalling\_errors.txt" for error messages and warning, and "Basecalling\_output.txt" for regular output (typically, the names of the files processed).
7. Aligner will then rename the samples that were base called, move them to the trash, and read the base calls from the new SCF files that PHRED has just created.
8. Finally, Aligner will delete all the temporary files it and PHRED have created. The temporary directories will be deleted when you exit Aligner.

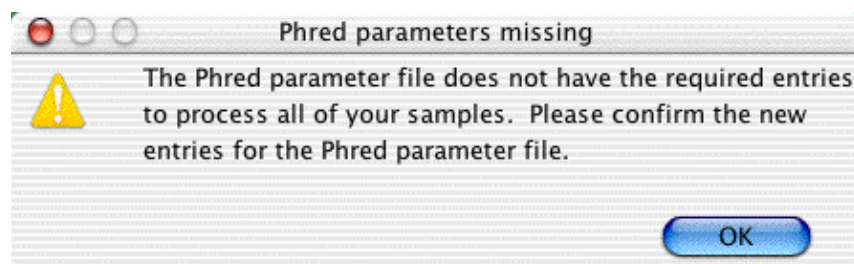
If anything went wrong, Aligner will leave the samples as they were; you will first need to correct the problem, and try to do the base calling again. The next section describes some of the most common problems and their solutions.

### Editing the PHRED parameter file

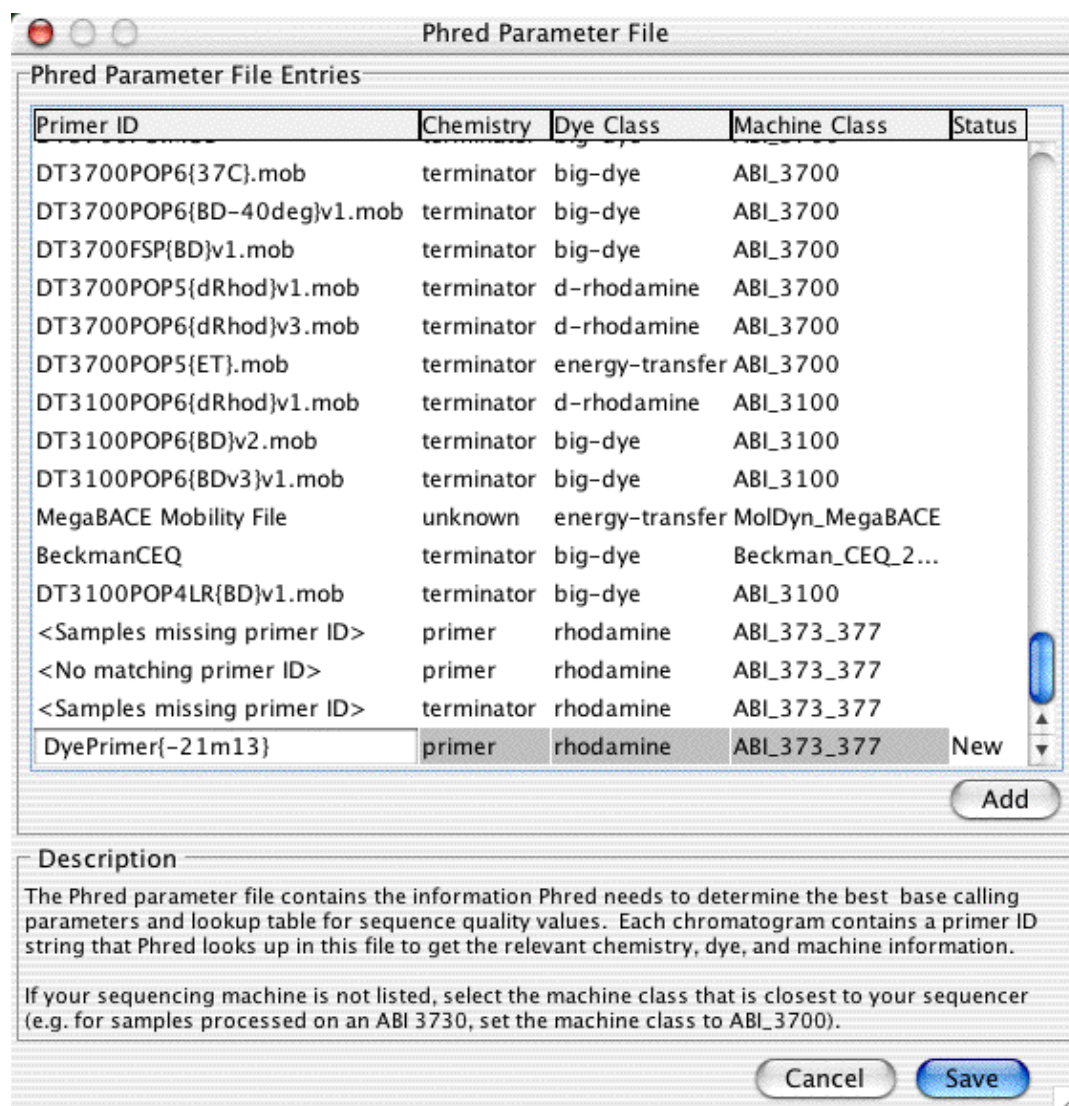
The first time you try to base call your own data with PHRED, you will probably need to add entries to the "Phred parameter file". PHRED uses this file to determine which sequencing chemistry, dye type, and sequencing machine were used to generate each individual sequence. PHRED does that by trying to match a "primer ID" string that is hidden in your chromatogram files with entries in the "Phred parameter file". Since sequencing vendors often come out with slightly changed chemistries that have a new "primer ID" string, chances are that you have to add one or more strings to the Phred parameter file.

Fortunately, CodonCode Aligner makes adding new entries to the Phred parameter file relatively easy by providing a convenient editor for the Phred parameter file, and by making educated guesses for the entries that need to be added. Before running Phred, Aligner checks all samples which you selected for base calling to see if their "primer ID" string can be found in the Phred parameter file. If any entries are missing, you will see the

following dialog:



When you click "OK", Aligner will bring up a new **dialog that allows you to edit the Phred parameter file**. It shows all the current entries in the Phred parameter file, as well as one or more entries that need to be added:



At the bottom, you see a newly added line (in your case, this may be more than one line; you may have to scroll down to see the newly added entries). Here is a brief description:

- The first column contains the **primer ID** string that you need to add. Aligner already filled this out for you, so you should not need to change anything here.
- The second column contains the **chemistry**, which is "primer" for dye-labeled primers, and "terminator" for dye-labeled terminator chemistry. Most sequencing nowadays is done with dye-labeled terminators, but check with the supplier of your sequences or sequencing kits if in doubt.
- The third column contains details about which **class of fluorescent dyes** was used. Most current chemistries from ABI are "big-dye", while chemistries from Amersham Pharmacia are typically "energy-transfer".
- The fourth column lists the **kind of DNA sequencer** used. If you cannot find an exact match, use the machine that is most similar to your machine. For example, if you are using PHRED version 0.020425.c, use **"ABI\_3700" for data from ABI 3730 and ABI3730XL sequencers**, and use **"ABI\_3100" for data from ABI 310 sequencers**.

You can also add new entries by pressing the "Add" button, but that should usually not be necessary. After verifying the new entries, press the "Save" button to save your changes. *If you press "Cancel", your changes will not be saved, and you will most likely see the ["Missing entries"](#) error described [below](#).*

You could also use a text editor to edit the Phred parameter file, but unless you really know what you are doing, we strongly suggest that you use Aligner's build-in editor instead, as described above.

You can also check and edit the Phred parameter file by pressing the button labeled "Edit..." in the [Base Calling Preferences](#).

## Base calling problems

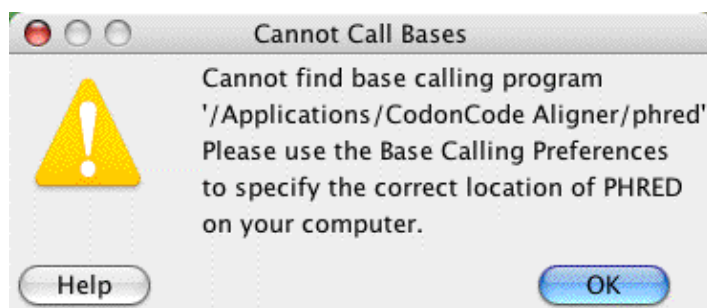
When using Phred for base calling, a number of things can go wrong. This section describes the most common problems, and how to solve them.

- [Cannot find the base calling program](#)
- [Missing entries in the Phred parameter file](#)
- [Problems reading the Phred parameter file](#)
- [Wrong command line parameters](#)
- [Problems running the workstation version of Phred](#)

Many of the problems are related to the Phred parameter file. This is a file used by Phred to determine which dye type, sequencing chemistry, and sequencing machine were used to a given trace. For more information, please read the ["About the Phred parameter file"](#) section below.

### Cannot find base calling program

If CodonCode Aligner cannot find the base calling program, Aligner will show the following error message:



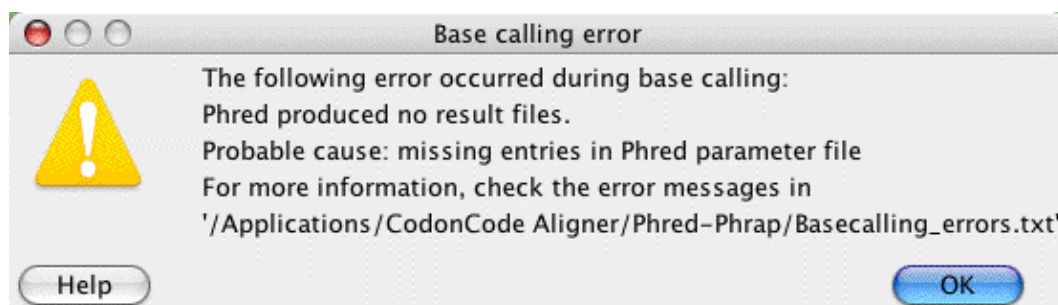
The name and location of the base calling program PHRED is defined in the base calling preferences; if the name specified there is not correct (for example because you moved, renamed, or deleted the program or the CodonCode Aligner folder), then Aligner cannot find Phred.

To solve this problem, do what the dialog says: use the [base calling preferences](#) to specify where exactly on your system Phred is installed. Detailed instructions are given [above](#) for the workstation version of Phred that is installed with CodonCode Aligner.

For more information, please read the [Prerequisites](#) section.

## Missing entries in the Phred parameter file

Another possible error message you may encounter looks like this:



This happens if Phred cannot find the "dye primer string" in the Phred parameter file (*since version 1.1.1 of Aligner, this should not happen anymore, since Aligner tries to help you in adding the missing entries before running PHRED, as described [above](#); however, you might still see this problem if you accidentally edited or misspelled a primer ID string, pressed "Cancel" in the Phred Parameter File dialog, or (against all our warning!) used a text editor to edit the Phred parameter file*).

Go ahead and open the file "Basecalling\_errors.txt" in Phred-Phrap directory inside the CodonCode Aligner directory. In there, you will see one or more entries like this:

```
SampleFileName.abi: unable to match primer ID string:    skipping chromatogram
unknown chemistry (DT3730POP7{BD}.mob) in chromat SampleFileName.abi
add a line of the form

"DT3730POP7{BD}.mob" <chemistry> <dye type> <machine type>

to the file /usr/local/genome/lib/phredpar.dat
```

To fix this problem, you need to add the line as requested to the Phred parameter file, as described [above](#).

At the top of the file, it tells you the known chemistries, dye types, and machines; pick the one that is closest to what was used for your sample. For example, the line you add may read:

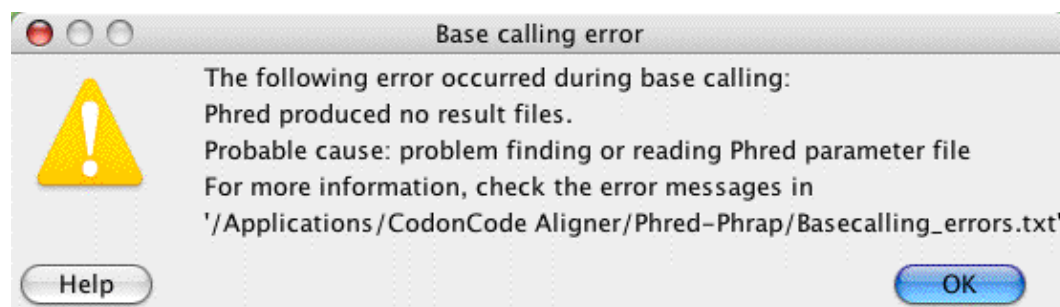
```
"DT3730POP7{BD}.mob" terminator big-dye ABI_3700
```

for the most commonly used sequencing chemistries from ABI 3700 or ABI 3730 sequencers.

Save the changed Phred parameter file (make sure to save it in a "Text only" format if using Word as the editor), and try the base calling again.

## Problems reading the Phred parameter file

If PHRED cannot find or read the Phred parameter file, you will see the following error message:

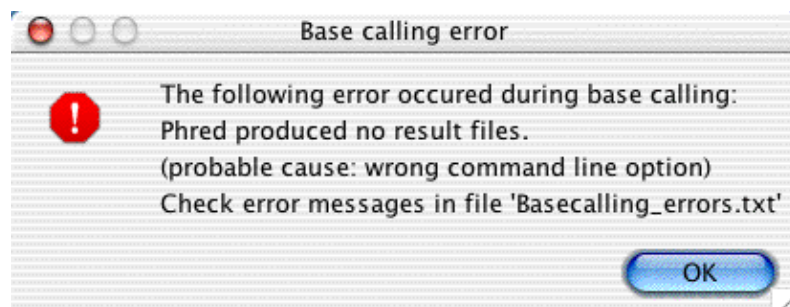


To fix this problem, you can do the following:

1. Make sure that the path to the Phred parameter file that is specified in the [base calling preferences](#) is correct, and that the Phred parameter file is readable.
2. Check the "Basecalling\_errors.txt" in the Aligner directory for hints about what is wrong.
3. Verify that the Phred parameter file is a plain text file, and not a binary format like RTF or Microsoft Word's ".doc" format.
4. Make sure that the file has the correct line endings, especially on OS X (due to its mixed origins, OS X files can have either Macintosh-style line endings or UNIX-style line ending; Phred requires UNIX line endings on OS X. You can use tools like [BBEdit Lite](#) or [LineBreak 2.2](#) to convert line endings, if needed.

## Wrong command line parameters

If you see the following error message:

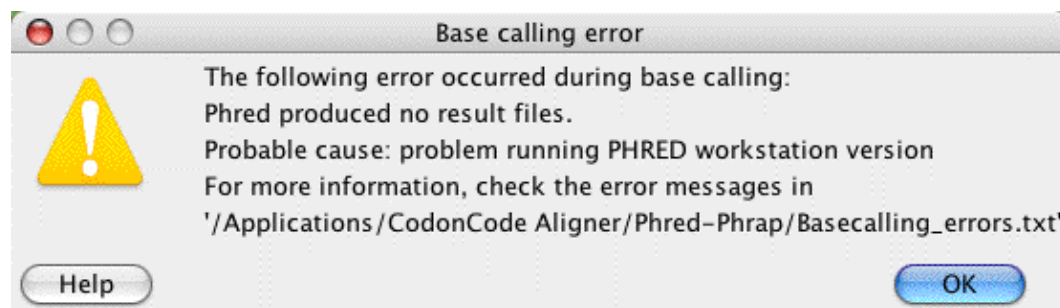




then you did not read the "for experts only" part in the Base Calling Preferences, did you? Or perhaps you just mistyped an option - go back to the [base calling preferences](#) and change the "Additional command line options". If you leave this line blank, as we suggest, you should not see this error message.

## Problem running the workstation version of Phred

When Aligner uses the workstation version of Phred, you may see the following error message:



Usually, you should not see this dialog - but if you see it, the first thing to try is to do the base calling again - it might work without problems the second time.

One thing that may cause this problem is renaming the Phred executable. Aligner looks at the program name to determine whether you are using the workstation version of Phred, or the regular version. If you have a regular (non-workstation) version of Phred, and you rename it to "workstation\_phred", you will see the dialog above.

If you did not rename the executable, and the problem persists, it's time to contact CodonCode support. Please send an email, and include the following as attachments:

- The "Basecalling\_errors.txt" file - the error dialog will tell you where exactly to find it
- The "Aligner\_errorlog.txt" file from the "CodonCode Aligner" directory
- The "status.log" file from the project directory
- A screen shot of the error message

Please send all this by email to [support@codoncode.com](mailto:support@codoncode.com).

## More about the Phred parameter file

The newest versions of PHRED (from the 2002 releases on) optimize the base calling for different sequencing chemistries, dye types, and machines. This requires that Phred can somehow determine the chemistry, dye type, and machine. Phred does this by extracting the "primer ID string" from the input files. The primer ID string looks something like this: "DyePrimer{-21m13}" or "ET\_MegaTerm" or "DT3700POP5{BD}v3.mob". PHRED looks up the primer ID string in the Phred parameter file, which tells it the dye type, chemistry, and machine through lines like this one:

```
"DT3700POP5{BD}v3.mob"      terminator      big-dye      ABI_3700
```

There are many different dye primer strings, and new ones get added all the time. If your sequence traces contain one that is not yet in the Phred parameter file, PHRED will refuse to call the bases in this trace, and generate an error message. In Aligner, this leads to the "[Missing entries in the Phred parameter file](#)" error described above. As described [above](#), you will need to [edit the Phred parameter file](#), and add a new line for

each new primer ID string. The possible entries are:

- for chemistry: primer, terminator, unknown
- for dyes : rhodamine, d-rhodamine, big-dye, energy-transfer, bodipy, unknown
- for machines : ABI\_373\_377, ABI\_3100, ABI\_3700, Beckman\_CEQ\_2000, LI-COR\_4000, MolDyn\_MegaBACE).

PHRED versions released in 2004 and later also support "ABI\_3730" as the machine type (at the time of writing, the new versions are still in beta testing, and have not yet been released). For more information, please check the [discussion groups](#) on CodonCode's support site.

In general, do not use the "unknown" tags. If your dye or machine is not listed, use the one most similar to it (for example ABI\_3700 for ABI 3730 XL sequencers if you are using PHRED version 0.020425.). The most common combination for ABI sequencing is terminator, big-dye, ABI\_3700.

The Phred parameter file can be in different locations on different systems. On Mac OS X, the default location is "/usr/local/genome/lib". On Windows, the default location is "/C:\Program Files\CodonCode\". To find out where the Phred parameter file is located, Phred checks the environment variable "PHRED\_PARAMETER\_FILE". This environment variable is temporarily set by Aligner when Aligner starts Phred, using the location of the Phred parameter file defined in the [base calling preferences](#)..

Additional tips for common problems with running Phred can also be found at <http://www.codoncode.com/support/faqs/phedpar.html>. If you want to, you can read more about Phred in the original Phred documentation, which is available at <http://www.codoncode.com/support/phred.doc.html>.

## About Phred

The base calling program Phred was developed by Phil Green and Brent Ewing at the University of Washington. Phred was widely used for base calling in the Human Genome Project, and still is often regarded as the standard for base calling. CodonCode Corporation distributes Phred executables for a variety of platforms, including Windows and Mac OS X, under license from the University of Washington.

Academic users can obtain the source code for Phred for restricted use directly from the authors at the University of Washington. For more information on this, please visit <http://www.phrap.org/>.

Trial versions of CodonCode Aligner may include time-limited trial versions of Phred. Any use of such trial versions are subjected to the license agreements displayed when you installed Aligner or Phred. In particular, you may not use the time-limited trial versions for any commercial purposes.

Please note that Phred is a command line program, not a typical Windows or Mac OS X application! This makes it easy to run Phred through scripts or from programs like CodonCode Aligner, but it means you cannot simply double-click on Phred and expect to see a graphical user interface.

You can read the original documentation for Phred at <http://www.codoncode.com/support/phred.doc.html>, and more about Phred in the following two articles:

- B. Ewing B, L. Hillier L, M.C. Wendl and P. Green. "Base-calling of automated sequencer traces using Phred. I. Accuracy assessment." *Genome Research*. 8: 175-85. ([Available](#) online at <http://www.genome.org/content/vol8/issue3/>).

## CodonCode Aligner User Manual

- B. Ewing and P. Green, 1998 "Base-Calling of Automated Sequencer Traces Using Phred. II. Error Probabilities." *Genome Research*. 8: 186-198. ([Available](http://www.genome.org/content/vol8/issue3/) online at <http://www.genome.org/content/vol8/issue3/>)

# End Clipping

Typically, sequence chromatograms have low-quality sequence at the beginning and at the end of the sequence. If you have sequence traces with quality values, you can use the quality values to automatically remove the low-quality sequence at the ends - a process called "end clipping" (or "end trimming").

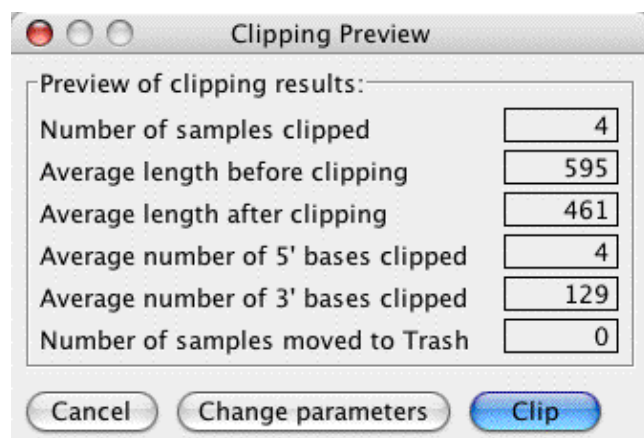
*If you plan to use Aligner's features to [detect and process heterozygous insertions and deletions](#), you should search for heterozygous indels **before** you end clip. Otherwise, end clipping will probably remove the parts of samples that have heterozygous insertions or deletions, and some heterozygous indels may not be detected. But if you first search for heterozygous indels, end clipping will leave the parts of sequences that have heterozygous indel tags unclipped.*

To end clip a set of sequences, do the following:

1. Select the sequences to be clipped in the project window. Typically, you can just select the "Unassembled Samples" folder to end clip all samples (only unassembled samples can be end clipped).
2. Select "Clip Ends..." from the sample menu. This will show the "Clipping preview" window that summarizes the clipping results (if you have selected many samples, it may take a few seconds for the dialog to show up).
3. If the end clipping results look ok to you, press the "Clip" button to apply the calculated clips. The low-quality bases at the end of each sequence will be removed, and any sequences that do not match the minimum quality criteria will be moved to the trash.

If you would like to change the settings for end clipping, press the "Change parameters" button instead of the "Clip" button. This will open a new window where you can change the end clip settings. After making your changes and selecting "OK", the clipping preview window will re-appear, showing the effects of the new parameters.

The clipping preview window is shown in the next picture:

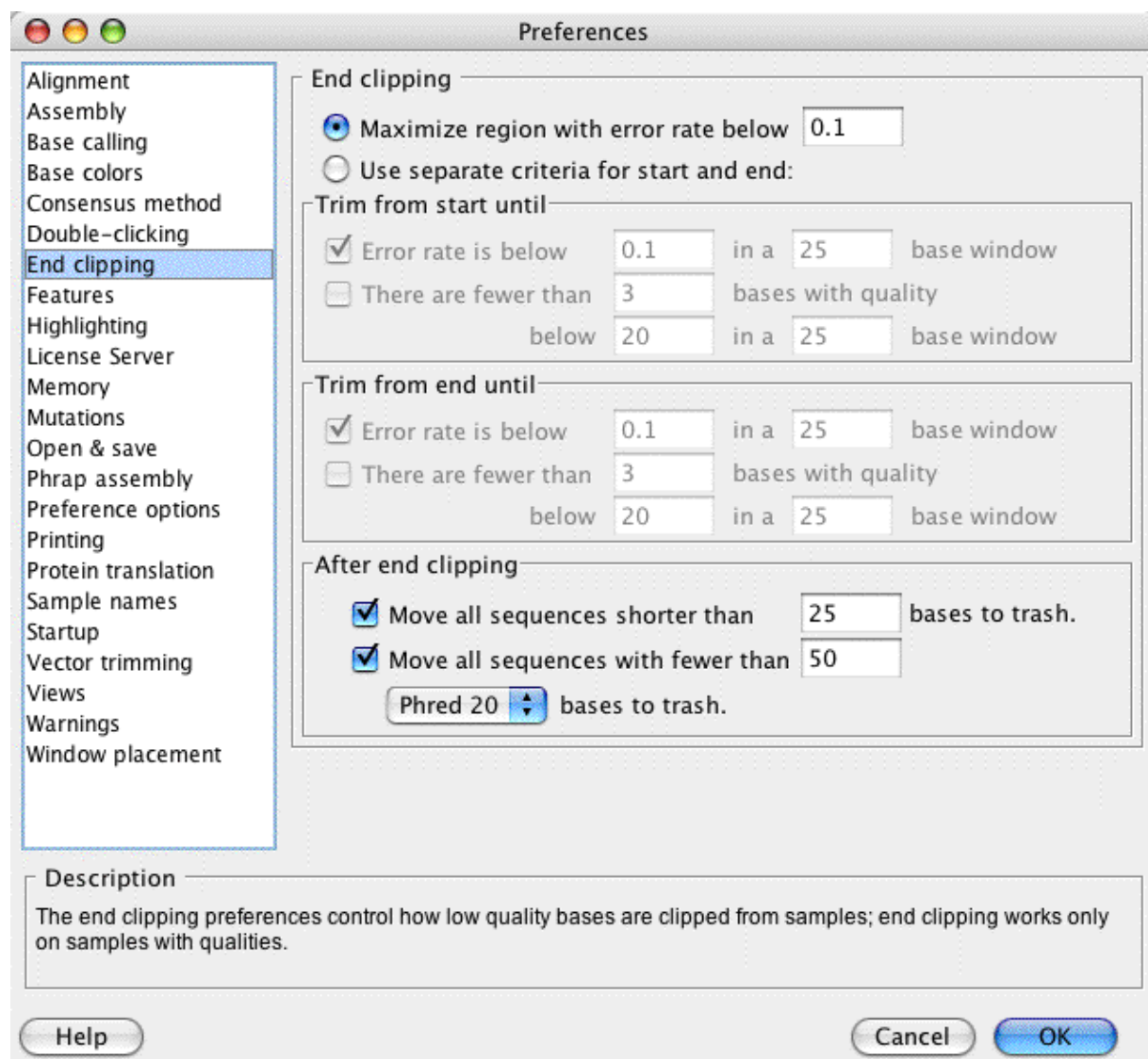


Please note that you do not have to end clip sequences before assembly or alignment - the assembly will typically work fine without end clipping. However, you can get cleaner and sometimes better assemblies by end clipping sequences before vector screening and assembly.

To each sequence that is clipped, Aligner will add a "processing" tag which states how many bases were clipped at the beginning and the end. You can see these tags in the [feature view window](#) if you have the processing tags included in your [definition of features](#), and in the [tag dialog](#) that is accessible from the [sample information dialog](#).

## End Clipping Parameters

You can change the stringency of the end clipping in the end clipping preferences. Either press the "Change parameters" button in the clipping preview window as described above, or select the "End clipping" preference panel in the "Preferences..." window (to see the Preferences window, select "Preferences" in the application menu on OS X, and "Preferences..." in the "Edit" menu on Windows). This will show the following window:



On the top, you have the choice between two different end clipping methods: a method that maximizes the region with an (estimated) error rate below the threshold you define, and a method that uses different criteria at the beginning and at the end of the reads. The first method is similar to the way Phred trims sequences with the "-trim\_alt" option. The second method gives you more options, and is (hopefully) a bit easier to understand. With typical parameters, both methods tend to give similar results, and remove the "junk" sequence at the end of chromatograms. The methods are [explained in more detail below](#).

You have the option to automatically identify "bad" sequences after the end clipping, and move those to the trash. Such bad sequences can be due to failed sequencing reactions, or any number of other problems. We suggest to move all sequences that are too short after clipping (for example, less than 25 or 100 bases) to the trash. A second widely used method to identify low-quality sequences is to count the number of bases with quality scores above 20 ("Phred20 bases"). With the settings shown in the picture above, any samples that, after the end clipping have fewer than 150 Phred20 bases, or are shorter than 200 bases, would be called bad and moved to the trash.

**We suggest that you experiment with the different parameters, and find a setting that works for your data.** For example, if you sequence short PCR products, you should definitely reduce the number of Phred20 bases required for a sequence to be kept, or perhaps uncheck this option.

Additional details about the end clipping parameters can be found on the ["End Clipping Preferences" help page](#).

## End Clipping Algorithms

This section briefly explains the different end clipping methods ("algorithms"). It is intended for the curious - you do not necessarily need to understand the end clipping algorithms to use them :-). All methods use the base-specific quality scores to find the low quality regions. For the end clipping to work correctly, the quality scores have to be reasonably accurate; however, they do not have to be perfect.

### Method 1: Maximizing regions with error rates below a given threshold

First, the quality scores at each base call are converted into estimated error rates. The quality scores are on a logarithmic scale - a quality of 10 corresponds to a 10% error rate, a score of 20 to 1%, 30 to 0.1%, and so on. Next, Aligner **finds the longest region** in the sequence where the following conditions are met:

- the average error rate over the entire region is below the (user-defined) threshold
- if the region would be expanded on either side, then the added sections would have an error rate that is **above** the threshold

The region found by this method can contain some lower-quality parts in the middle, where the estimated error rate is above the threshold. However, such regions will only be included if they are followed by a region that has a lower error rate, which drops the combined error rate over both regions below the threshold. Individual bases and short regions outside of the "good" region may also have error rates below the threshold - however, they are not included when they are flanked by higher error regions.

The "maximize region" method is very similar to the method used by the base calling program Phred with the "-trim\_alt" option, and based on the Mott algorithm for sequence trimming. It's a bit hard to understand, but works rather well.

## Method 2: Using separate criteria at the start and the end of the sequence

In this method, the clipping is done by coming in from the start and from the end of a sequence, and "chewing into" the sequence until a region that is "good" is encountered. There are two different ways to define what is a "good" region:

1. the average estimated error rate in a region (calculated from the quality scores), and
2. the number of "bad" bases (bases with a quality below the defined threshold).

You can also define the length of the region, and use different regions at the beginning and at the end of a sequence. At the beginning, where sequence quality improves rapidly, a short window (e.g. 20 bases) can make sense, while at the end, where the sequences quality drops slowly, a longer windows (e.g. 50 bases) can make sense. You can use either one of the two definitions, or you can use both together.

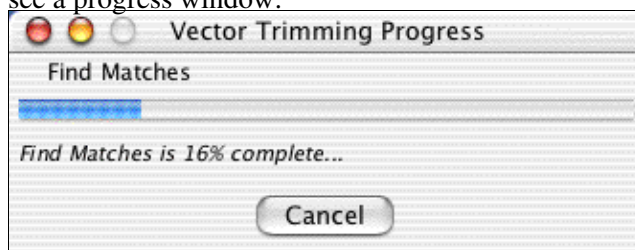
The advantages of this method are that (a) it is a bit easier to understand, and (b) it gives you more options to tune the trimming exactly like you want it.

# Trimming Vector Sequences

Vector sequences in your sample sequences can lead to incorrect assemblies, and therefore should be trimmed before assembly or alignment. *If you also plan to use end clipping to remove low-quality sequence, we strongly suggest that you do the end clipping before the vector trimming.*

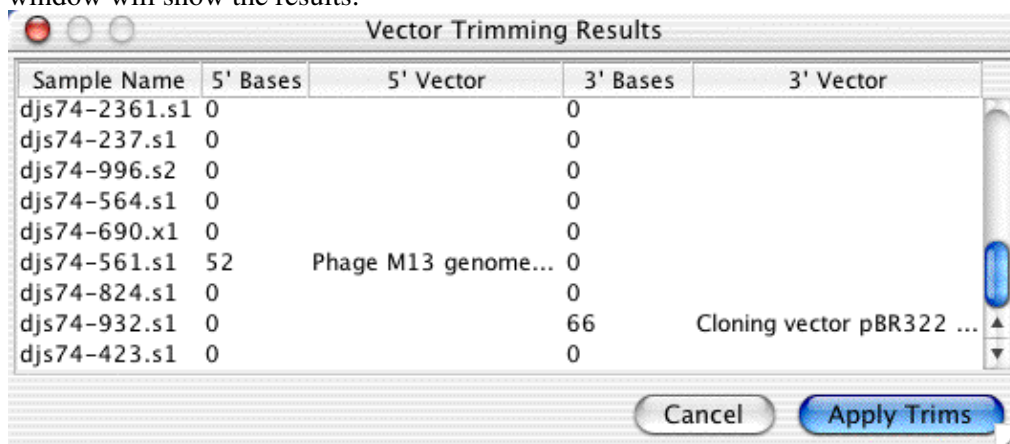
To remove vector contamination from samples, follow these steps:

1. Check your "Vector trimming" Preferences. Make sure that you have selected the cloning vector(s) that you used.
2. Since vector trimming cannot be undone, it's a good idea to save your project now.
3. In the Project window, select the samples that you want to screen. You can screen only unassembled samples, not samples in contigs. Typically, just select the "Unassembled Samples" folder to screen all unassembled samples.
4. Select "Trim Vector" in the "Sample" menu. Aligner will start to look for matches between the samples you selected in the project window, and the vectors you selected in the Preferences. You will see a progress window:



You can cancel the vector trimming at any time before it is complete.

5. After Aligner has found all vector matches that fit the criteria you defined in the Preferences, a new window will show the results:



| Sample Name   | 5' Bases | 5' Vector           | 3' Bases | 3' Vector                 |
|---------------|----------|---------------------|----------|---------------------------|
| djs74-2361.s1 | 0        |                     | 0        |                           |
| djs74-237.s1  | 0        |                     | 0        |                           |
| djs74-996.s2  | 0        |                     | 0        |                           |
| djs74-564.s1  | 0        |                     | 0        |                           |
| djs74-690.x1  | 0        |                     | 0        |                           |
| djs74-561.s1  | 52       | Phage M13 genome... | 0        |                           |
| djs74-824.s1  | 0        |                     | 0        |                           |
| djs74-932.s1  | 0        |                     | 66       | Cloning vector pBR322 ... |
| djs74-423.s1  | 0        |                     | 0        |                           |



## CodonCode Aligner User Manual

*(You will see a different dialog if no matches between your samples and the vector sequences were found).*

You can now choose to apply the trim results and remove the bases at the start or end of the samples that had vector matches; or you can press cancel and not apply the trim results (for example to try different parameters).

You can repeat the vector trimming, for example if you forgot to include a vector sequence the first time you trimmed.

To each sequence that has bases removed due to vector trimming, Aligner will add a "processing" tag which states how many bases were trimmed at the beginning and/or the end. You can see these tags in the [feature view window](#) if you have the processing tags included in your [definition of features](#), and in the [tag dialog](#) that is accessible from the [sample information dialog](#).

# Vector Library Files

Vector sequences for vector trimming can be read from several files that were supplied with CodonCode Aligner, or from your own ("custom") files. A "Vector" folder is created in the "Aligner Data" folder in the folder where CodonCode Aligner was installed (*on OS X, the default folder is "/Applications/CodonCode Aligner/"; on Windows, the default folder is "C:/Program Files/CodonCode Aligner/"*). Three files are installed in this vector folder: two UniVec files from NCBI, and an example custom vector file with some commonly used vector sequences.

To set up vector trimming, you:

1. Open the "Vector trimming" preference panel
2. Select the vector library file to use (typically from your own "Custom Vector" file)
3. Select the sequences you want to screen against

## Using UniVec Library Files

UniVec is a database created by the NCBI where redundant sub-sequences from vectors have been removed. UniVec also contains sequences for linkers, primers, and adapters that are commonly used in cloning. Two UniVec library files have been installed in your "Vector" folder inside your settings folder.

- UniVec.txt
- UniVec\_Core.txt

For more information about UniVec refer to:

<http://www.ncbi.nlm.nih.gov/VecScreen/UniVec.html>

<http://www.ncbi.nlm.nih.gov/VecScreen/Interpretation.html#Definitions>

The typical use of UniVec for vector screening would be to screen against all sequences in UniVec. In CodonCode Aligner, this is currently not supported - it would take a **really** long time. You can select several sequence fragments from UniVec to screen against, but **typically, it's a better idea to use a custom vector file**.

## Using Custom Vector Files

Custom vector files can contain any collection of vector, linker, etc. sequences that you want to screen against in FASTA format. An example custom vector file called "CustomVectors.txt" is installed in your "Vector" folder inside your Preferences folder. It includes some common vectors like BlueScript, M13, pBr, pGEM and pUC. Custom files can be kept in the same vector folder as the UniVec files, or in any folder that you choose. To add sequences you want to screen against, we suggest that you make a copy of the "CustomVectors.txt" file, and edit the copy. If you use linkers or PCR primers in your subcloning, it's a good idea to add the sequences of the linkers and primers to the custom vector file, and select them to be included when screening (in the "Vector trimming" preference panel).

When screening against short sequences like primers, however, keep in mind that short matches may not meet the minimum match criteria, and therefore not be masked. Minimum match criteria are discussed under "Vector screening" preferences.

## CodonCode Aligner User Manual

**Note:** When editing a custom vector file, please make sure that it is saved as a "text (.txt)" file. Aligner will not be able to read files in other formats, like Microsoft Word's ".doc" format.

**Custom vector files must be in FASTA format:** each sequence must have a line that has a "greater than" (>) sign, followed by the name. The file can have multiple entries; here is an example:

```
>gnl|uv|L09150.1:3248-3353 pUR291 cloning vector
CGCCGGTCGCTACCATTACCAGTTGGTCTGGTGTCTCGGGGATCCGTCGACCTGCAGCCAAGCTTATCGATG
ATAAGCTGTCAAACATGAGAATTCTTGAAGACGAAA
>gnl|uv|L09151.1:3249-3332 pUR292 cloning vector
GCCGGTCGCTACCATTACCAGTTGGTCTGGTGTCTAGGGGATCCGTCGACCTGCAGCCAAGCTTATCGATG
ATAAGCTGTCAAAC
>pUR278 cloning vector bases 3239-3335
CCAGCTGAGCGCCGGTCGCTACCATTACCAGTTGGTCTGGTGTCAAAAAGGGGATCCGTCGACTCTAGAA
AGCTTATCGATGATAAGCTGTCAAACA
```

The names for the entries are used when you select the vector sequences to screen against in the preferences - that's why you cannot use plain text vector sequence files.

# Assembly and Alignment

CodonCode Aligner offers a variety of functions to assemble and align DNA sequences:

- [Assembly](#)
- [NGS Assembly](#)
- [Alignment with Clustal or Muscle, or based on protein translations \(with MACSE\)](#)
- [Alignment to a reference sequence](#)
- [Alignment with Bowtie2](#)

## Sequence Assembly

[Assembly](#) assembles sequence fragments into a larger sequence by identifying overlaps between sample sequences. Samples that can be joined together are put into "contigs". Joins may fail because samples do not share overlaps that are long enough with other samples or contigs, or because the overlap contains too many discrepancies. Any sequences that cannot be put into contigs remain in the "Unassembled Samples" folder.

The result of the assembly can be one or more contigs, plus samples that remain in the "Unassembled Samples" folder. If none of the samples can be joined, no contigs will be formed.

You can assemble any mix of unassembled samples and previously assembled contigs. Contigs are assembled as they are, they are **not** dissolved first. However, the consensus sequence of contigs may change where new samples are added.

The "**Advanced Assembly**" submenu in the "**Contig**" menu offers several advanced options for sequence assembly. These include:

- [Assemble with pre-processing](#) - this enables you to pre-process unassembled samples automatically before assembly by base calling, end clipping, and/or vector trimming.
- [Assemble in groups](#) - this option lets you define how CodonCode Aligner should group samples based on their names; Aligner will then build separate contigs for each sample group.
- [Compare contigs](#) - this lets you build "Contigs of contigs", for example for phylogenetic studies. You can choose between three different algorithms - the built-in assembly algorithm, and the alignment programs Clustal and Muscle.
- [Assemble from scratch](#) - this option dissolves existing contigs before assembly.
- [Assemble with PHRAP](#) - With this option, you also have a choice between two assembly algorithms - the built-in assembly algorithm and the assembly program PHRAP (note that users at companies may have to pay a separate license fee to use PHRAP).

The "Assemble" functions in CodonCode Aligner are intended for classical sequencing projects, for example sequencing a single gene using ABI sequencing. For additional details and step-by-step instructions, check the "[Assembly](#)" page.

## NGS Assembly

CodonCode Aligner offers several tools for the [assembly of NGS sequence data](#) (for example Illumina reads). Since NGS data files typically contain millions of samples at high coverage, the assembly is done using external data files, and only the resulting contigs are imported into projects (in contrast to classical sequencing projects, where a much smaller number of sequences is imported into the projects before

assembly). If necessary, NGS data can be pre-processed, for example to remove sequencing adapters or trim low-quality sequence, using the [NGS pre-processing functions](#) in CodonCode Aligner.

For additional details and step-by-step instructions, check the "[NGS Assembly](#)" page.

## Alignment with Clustal, Muscle, or based on protein translations (with MACSE)

To facilitate the [alignment of multiple DNA sequences](#), for example from different species, CodonCode Aligner provides access to the commonly used sequence alignment programs Clustal and Muscle. For alignment of more divergent coding sequences, Aligner also supports the generation of amino-acid translation-based sequence alignments using the program MACSE.

A unique feature in CodonCode Aligner is that the alignments can be generated *directly* from existing contigs, without the need to export or duplicate consensus sequences. In ABI sequencing projects, this conserves the link to underlying sequence traces, and allows for efficient verification and editing of discrepancies from multi-sequence alignments.

For additional details and step-by-step instructions, check the "[Alignment](#)" page.

## Alignment to a Reference Sequence

[Alignment to a reference sequence](#) is used to determine differences between sequences and a known sequence. This requires that you first designate one sequence as the reference sequence. All other samples are then aligned to this sequence. If necessary, the samples are reverse-complemented before alignment.

Alignment results in one new contig that contains the reference sequence as well as any samples that could be aligned to it. If no samples could be aligned to the reference sequence with the current alignment criteria, no new contig will be formed, and all samples will remain in the "Unassembled Samples" folder.

The new contig will be limited to the length of the reference sequence, plus any gaps introduced during alignment.

When building the consensus sequence for contigs that result from alignments to a reference sequence, the [consensus preferences](#) determine how the reference sequence is considered. By default, the reference sequence is excluded from the consensus sequence. In this case, any regions where none of the samples overlap with the consensus sequence will result in gap characters in the consensus sequence. Alternatively, you can choose to use the reference sequence as the consensus sequence.

The "**Advanced Alignments**" submenu in the "**Contig**" menu offers several advanced options for aligning sequences to a reference sequence. These include:

- [Align with pre-processing](#) - this enables you to pre-process unassembled samples automatically before alignment by base calling, end clipping, and/or vector trimming.
- [Align in groups](#) - this option lets you define how CodonCode Aligner should group samples based on their names; Aligner will then build separate contigs for each sample group.
- [Align from scratch](#) - this option dissolves existing contigs before alignment.

**Limitations:** Currently, the implementation of alignments to reference sequences has several limitations. These include:

- The numbering in the aligned contig is not relative to the reference sequence numbering, and includes any gaps introduced in the consensus sequence. However, if you use the "Find mutations" function, the numbering used in the mutation tags will be relative to the reference sequence (and, if present, the coding sequence annotation of the reference sequence).
- With the default settings, aligning a cDNA sequence with several exons to a genomic sequence will typically fail, or give a bad alignment. However, you can change this by selecting the "Large Gap" algorithm in the [alignment preferences](#) (the "Large Gap" algorithm is specifically intended for cDNA to genomic alignments).

## Alignment with Bowtie2

For the alignment of many sequences to long reference sequences, CodonCode Aligner supports the use of [Bowtie2](#) directly from Aligner. For Bowtie2 alignments, the reference sequences can either be in the current CodonCode Aligner project, or in an external data file; the sequences to be aligned have to be in external files. The results of the alignment are imported into CodonCode Aligner as contigs; alternatively, the results can be viewed by the popular alignment viewing program Tablet (if it is installed on the computer).

For additional information about alignments with Bowtie2, check the ["Bowtie2"](#) page.

# Sequence Assembly

## Before assembling

Before performing an assembly in CodonCode Aligner, you'll need to create or open a project, and import the sample files you want to assemble. You also may want to pre-process your samples by end clipping and vector trimming (it's a good idea to save your project now, before starting the assembly).

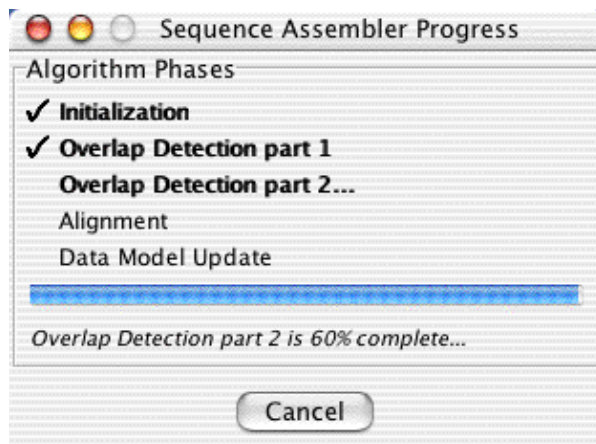
## How to assemble

- Go to the project view
- Select the samples and/or contigs that you want to assemble
- Choose "Assemble" from the "Contig" menu

Note:

- To make a **continuous selection**, keep the "shift" key pressed while clicking on samples or contigs
- To make a **discontinuous selection**, press the "control" key (on Windows) or the "command" key (on OS X) while clicking on samples or contigs

The assembly will start and show a progress window:



After the assembly is done, the progress window will disappear, and the project window will now show the newly formed contig(s) (unless no joins were successful, in which case you will see a dialog telling you so).

When CodonCode Aligner assembles pre-formed contigs, it looks for matches between the consensus sequences, and leaves the alignment of reads in the contig (mostly) unchanged. If contigs are merged with other reads or contigs, any necessary gaps will be added.

## Advanced assembly options

In addition to the simple assemblies described above, CodonCode Aligner also supports several "advanced" assembly options, including:

- [Assemble with preprocessing](#) - CodonCode Aligner can also automatically do common pre-processing steps like end clipping and vector trimming before assembly.

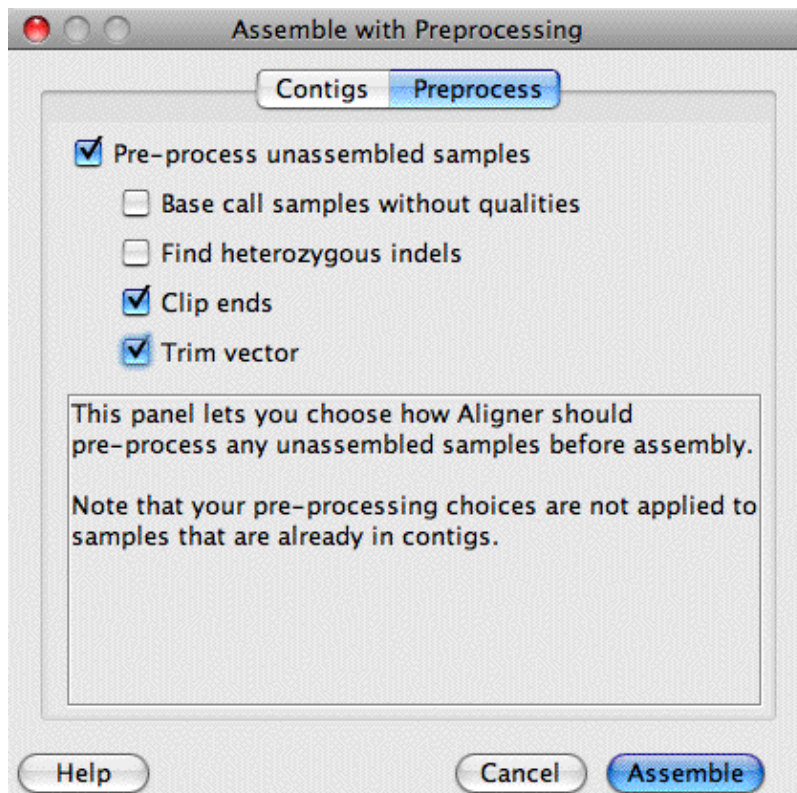
- [Assemble in groups](#) - with this option, CodonCode Aligner can automatically group samples based on their **names** or based on **multiplex sequence tags**, and form separate contigs for each group. This option can be a real time saver in phylogenetic and medical studies where you look at sequences from many species, isolates, or patients.
- [Compare contigs to each other](#) - this option allows you to compare several contigs, for example when studying genes from different species, isolates, or patients. The consensus sequences will be assembled into a new contig; to check out any differences, double-clicking on a sequence in this "contig of contigs" will bring you straight to the sequence traces. You can choose between the widely used alignment programs **ClustalW** and **muscle**, or use the built-in algorithm.
- [Assemble from scratch](#) - this will unassemble any existing contigs in your selection before starting the assembly. This option can be useful too if you want to undo manual introduction or movement of gaps, or to try different assembly parameters.
- [Assemble with PHRAP](#) - this option will use the assembly program PHRAP, rather than CodonCode Aligner's built-in methods, for sequence assembly. It can be useful for larger shotgun assemblies.

## Assemble with preprocessing

Assemble with preprocessing lets you do common pre-processing steps like end clipping and vector trimming before assembly.

1. Select the unassembled samples and contigs you want to work with in the project view.
2. Go to the "**Contig**" menu, move to the "**Advanced Assembly**" submenu, and select "**Assemble with Preprocessing**".
3. In the dialog that pops up, click on the "Preprocess" tab, and select the checkboxes you want to use.

Here is what the dialog looks like:





The checkbox at the top determines whether or not your samples will be pre-processed before assembly; the four lower check boxes let you pick the pre-processing steps that will be done. The choices are:

- **Base call samples without qualities:** Any samples that have chromatograms, but no base-specific quality scores, will be base called with PHRED. To use this option, you will need either a trial license or a purchased license; base calling is not enabled in demo mode. Additional information about base calling can be found at the "[Base calling](#)" help page.  
Please note one important difference to the normal "Call bases" menu choice: in automated pre-processing, samples that already have quality values, for example from the ABI KB basecaller or from calling bases on the sample before, will not be base called again.
- **Find heterozygous indels:** This option will look for potential heterozygous insertion/deletion (indel) mutations in the unassembled samples that have (a) chromatograms and (b) quality scores. If you are sequencing PCR products from genomic DNA that may contain heterozygous indels, you should check this option; if you are sequencing from cloned DNA, this option should not be checked. For more information, please read the "[Heterozygous insertions and deletions](#)" help page.
- **Clip ends:** This will remove low-quality sequence from samples that have chromatograms and quality scores. For more information, please read the "[End clipping](#)" help page.
- **Trim vector:** When checked, this option will identify and remove vector sequence contamination from the samples in your selection. You may need to set your [vector trimming preferences](#) before using this option. For additional information, please read the "[Vector trimming](#)" help page.

You will see progress dialogs for each of the steps performed after you click the "Assemble" button.

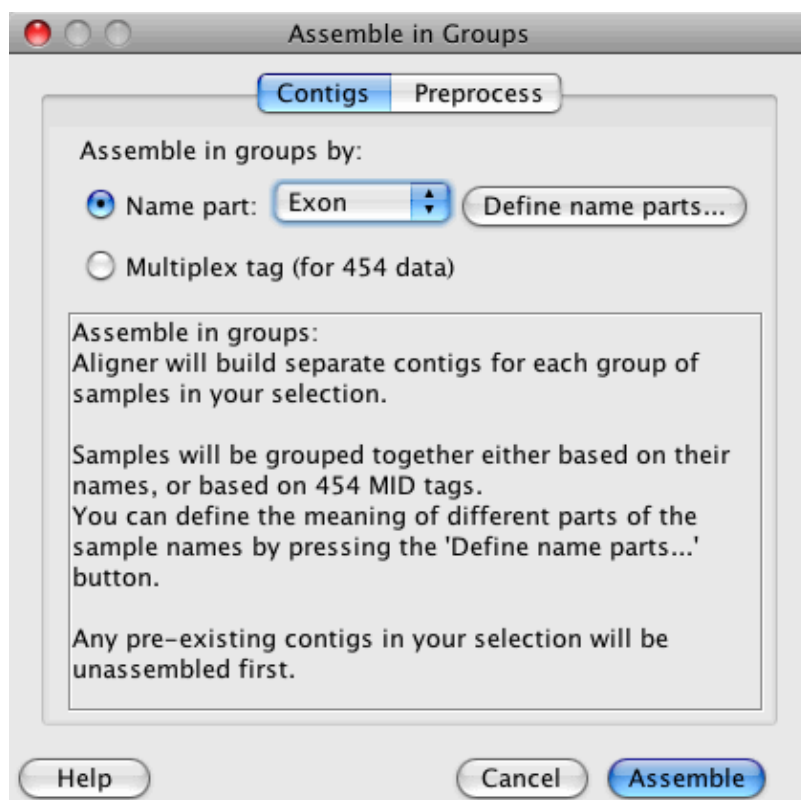
**Note that any samples that are already in contigs will not be pre-processed** - the preprocessing steps are only applied to unassembled samples in your selection.

## Assemble in groups

"Assemble in Groups" allows you to automatically build separate contigs for different sample groups, based on sample names or multiplex sequence tags (MID tags).

After selecting the samples you want to assemble in the project view, go to the "**Contig**" menu, move to "**Advanced Assembly**", and select "**Assemble in Groups...**".

This will display the following dialog:



Using the radio buttons at the top, select to assemble in groups by name part or by multiplex sequence tag.

## Assemble in groups by name part

You may want to use this option if you need to assemble contigs from a number of different sources, for example species, patients, isolates, and so on. To assemble in groups by name part:

1. Select the samples you want to assemble in the project view (you can also include contigs, which will be unassembled before building new contigs).
2. Go to the "**Contig**" menu, move to "**Advanced Assembly**", and select "**Assemble in Groups...**".
3. Click the "Define name parts..." button to define how sample names should be interpreted.
4. Select the name part to assemble by in the pulldown menu at the top.
5. Click the "Assemble" button.

You will see a progress dialog while CodonCode Aligner assembles separate contigs for each sample group in your selection. You can also automatically pre-process unassembled samples before the assembly in the "Preprocess" tab, as described above.

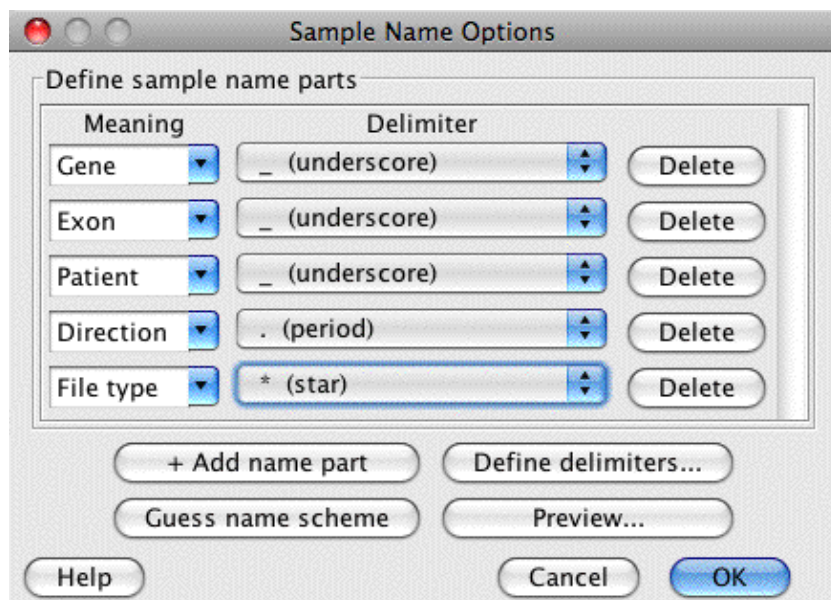
To use assemble in groups by name part, your samples must be named consistently. Typically, different parts of the name are separated by characters like underscores or periods.

Let us look at an example sample name, "EGFR\_exon19\_JJM\_F.abi". It consists of:

- The gene name ("EGFR")
- The exon ("exon19")
- A patient identifier ("JJM")
- The direction ("F" for forward)

- The file type extension ("abi")

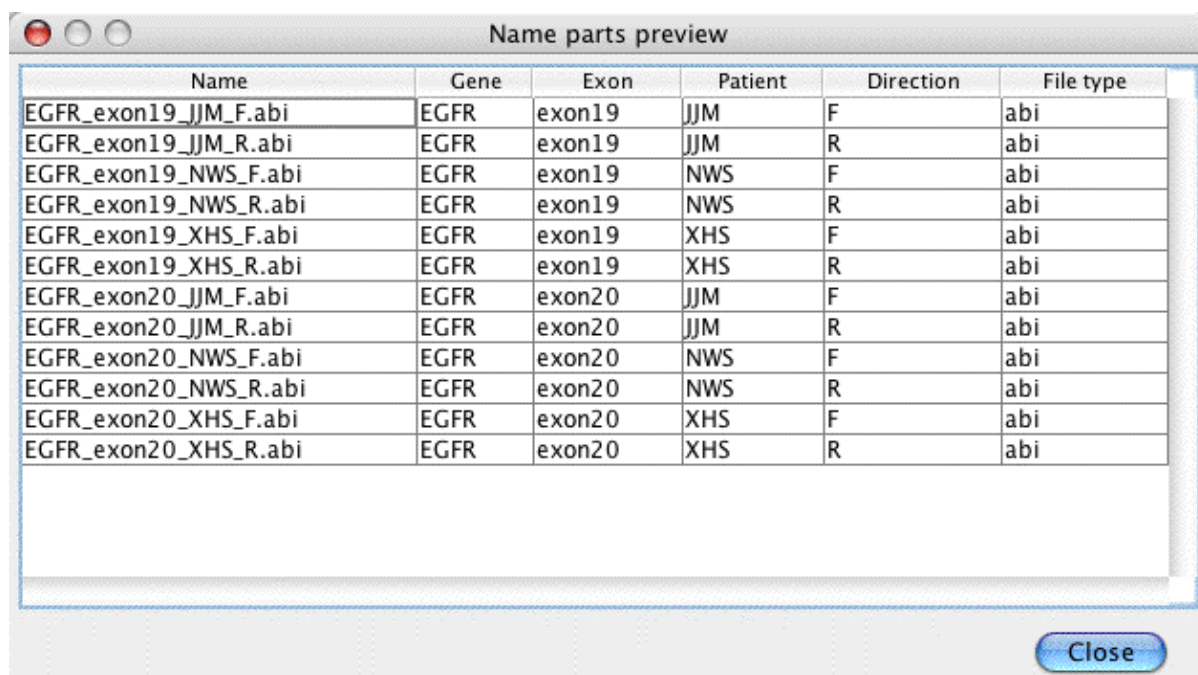
Most name parts are separated by underscores, except for the last two parts, which are separated by a period. Obviously, sample naming conventions will be different for different projects, so you will need to tell Aligner how to interpret ("parse") sample names. You can do this in the ["Sample names" preferences](#), or by clicking on the "Define name parts..." button in the "Assemble in Groups" dialog. For the example above, the definition would look like this:



Note that the last delimiter in this example does not matter, since the last part of the sample name is the file type.

You can define the name scheme manually by using the "Add name part" button and the "Meaning" and "Delimiter" pulldown menus, or use the "Guess name scheme" button to have CodonCode Aligner guess the name scheme for you. Guessing the name scheme will typically work if the sample name parts are separated by delimiters, and just one of the name parts varies.

To verify that the name part definition is correct, you can press the "Preview..." button; this will show the parsing of the currently selected samples in a separate window:



| Name                  | Gene | Exon   | Patient | Direction | File type |
|-----------------------|------|--------|---------|-----------|-----------|
| EGFR_exon19_JJM_F.abi | EGFR | exon19 | JJM     | F         | abi       |
| EGFR_exon19_JJM_R.abi | EGFR | exon19 | JJM     | R         | abi       |
| EGFR_exon19_NWS_F.abi | EGFR | exon19 | NWS     | F         | abi       |
| EGFR_exon19_NWS_R.abi | EGFR | exon19 | NWS     | R         | abi       |
| EGFR_exon19_XHS_F.abi | EGFR | exon19 | XHS     | F         | abi       |
| EGFR_exon19_XHS_R.abi | EGFR | exon19 | XHS     | R         | abi       |
| EGFR_exon20_JJM_F.abi | EGFR | exon20 | JJM     | F         | abi       |
| EGFR_exon20_JJM_R.abi | EGFR | exon20 | JJM     | R         | abi       |
| EGFR_exon20_NWS_F.abi | EGFR | exon20 | NWS     | F         | abi       |
| EGFR_exon20_NWS_R.abi | EGFR | exon20 | NWS     | R         | abi       |
| EGFR_exon20_XHS_F.abi | EGFR | exon20 | XHS     | F         | abi       |
| EGFR_exon20_XHS_R.abi | EGFR | exon20 | XHS     | R         | abi       |

After defining the name scheme, you can choose which name part Aligner should use to group samples, by selecting this name part in the "Assemble in Groups" dialog. In this example:

- Choosing "Gene" would try to assemble all reads into one contig
- Choosing "Exon" would try to assemble the genes into 2 contigs, one for exon 19 and one for exon 20.
- Choosing "Patient" would assemble the samples for each patient separately. Assuming that there is enough overlap between the exon 19 and exon 20 sequences, this would generate one contig for each of the three patients. Without sufficient overlap between the exon 19 and exon 20 sequences, the assembly would generate 2 separate contigs for each of the patients (for exon 19 and for exon 20), for a total of 6 contigs.
- Choosing "Direction" would assemble all the forward ("F") reads together, and all the reverse ("R") reads separately.

The contigs created will be named according to the name part used to group samples; for example, assembling by patient would create contigs called "JJM", "NWS", and "XHS". It is possible that the assembly will create more than one contig for each group, for example if some of the samples in a group do not overlap, or are too different from each other. If more than one contig per groups is created, the contigs will be named by adding numbers to the end, for example "JJM" and "JJM1".

CodonCode Aligner's name parsing scheme is very powerful and flexible; in addition to separator characters, you can also use fixed length name parts, and even "patterns". Additional information about defining name parts is available on the ["Sample names" preferences help page](#).

## Assemble in groups by multiplex sequence tag

When assembling 454 data, you can use "Assemble in groups" and multiplex (MID) tags to sub-divide your samples into smaller groups. CodonCode Aligner will look for the following multiplex tags in your samples:

|  |  |
|--|--|
|  |  |
|--|--|

| Multiplex Tag | Group Name |
|---------------|------------|
| ACGAGTGCGT    | MID1       |
| ACGCTCGACA    | MID2       |
| AGACGCACTC    | MID3       |
| AGCACTGTAG    | MID4       |
| ATCAGACACG    | MID5       |
| ATATCGCGAG    | MID6       |
| CGTGTCTCTA    | MID7       |
| CTCGCGTGTCT   | MID8       |
| TAGTATCAGC    | MID9       |
| TCTCTATGCG    | MID10      |
| TGATACGTCT    | MID11      |
| TACTGAGCTA    | MID12      |

Only the first 50 bases will be searched for exact matches to these tags. The samples which have the same multiplex tag will be grouped together. Samples that do not have any tag, or matches to more than 1 tag, will not be assembled. The names of the contig(s) formed for each group will start with the "Group name" as shown in the table above.

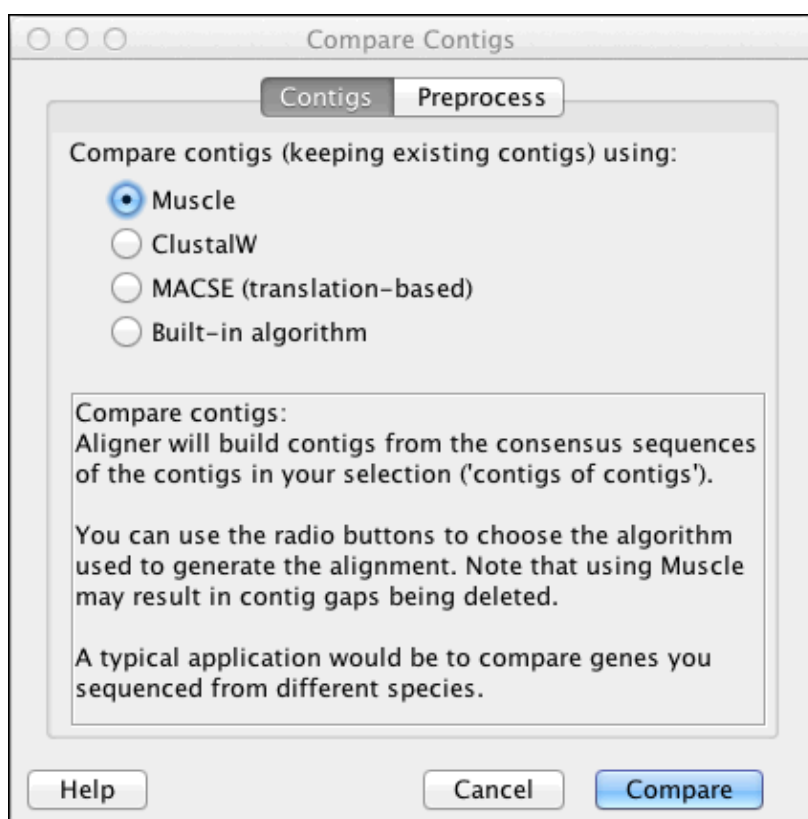
#### Some things worth noting for "Assemble in groups":

- You can also automatically pre-process unassembled samples before the assembly in the "Preprocess" tab as described in [Assemble with preprocessing](#).
- Assembling in groups will try to group and assembly all samples and contigs in your current selection. Any existing contigs in your selection will be unassembled before assembly.
- After creating contigs by "Assemble in Groups", you can [compare the contigs to each other](#) by building "contigs of contigs", as described in the next section.

## Compare contigs to each other

A common DNA sequencing application is to first sequence the genes from several sources, for example different species or patients, and to then compare (align) the consensus sequences to each other. CodonCode Aligner allows you to do this by creating "contigs of contigs", as follows:

- Assemble the contigs separately, for example generating one contig from several forward and reverse reads for each species.
- Select the contigs you want to compare in the project view. You can also include individual sequences, for example text sequences that you downloaded from sequence databases.
- Go to the "Contig" menu, move to "Advanced Assembly", and choose "Compare Contigs..." .
- A dialog opens that looks like this:



- Click on "**Compare**".

Aligner will start comparing the contigs to each other, showing you a progress dialog during the assembly. By default, CodonCode Aligner version 1.6 and newer will use the program **Muscle** to generate contigs of contigs. **Muscle** is a program that is widely used to generate alignments for phylogenetic studies; a copy of the program is included with the CodonCode Aligner distribution, and installed in the "Phred-Phrap" folder inside the CodonCode Aligner install folder.

To generate alignments with **Muscle**, CodonCode Aligner does the following:

1. Aligner examines your current selection to see if any contigs or samples need to be reverse-complemented before alignment, and reverse-complements these contigs and/or samples.
2. Aligner generates an input file for **Muscle**.
3. Aligner starts **Muscle**, and waits for the alignment to finish. During this time, a progress dialog is shown. Please note that alignments with **Muscle** can be slow, especially if you align many contigs or samples, and if your contigs are large.
4. When **Muscle** is finished, CodonCode Aligner analyzes the output generated by **Muscle**, and creates a new contig from the **Muscle** alignment. Aligner also calculates a consensus sequence for the alignment.

*Please note that muscle removes any existing gaps before performing alignments. Unless gaps are re-introduced at the same position during the alignment, CodonCode Aligner will have to delete the gaps in the contigs when importing the alignment results. This can, in turn, lead to the deletion of non-gap characters in samples at this contig position. Typically, these deleted bases are random errors. If you want to avoid such "automatic" edits, you can either use Clustal or the built-in algorithm to create the alignments, or you can change the [Consensus method preferences](#) to replace '-' in the consensus sequences with 'n'.*

## Algorithms for comparing contigs: Clustal, Muscle, MACSE, built-in

If you do not want to use **Muscle** to generate contigs of contigs, you have several alternative options: to use either the program **Clustal**, the program **MACSE**, or CodonCode Aligner's **built-in assembly algorithm** to form contigs of contigs. To do this, select the "ClustalW" or "Built-in algorithm" button in the "Compare contigs" dialog.

When you click on "Assemble" (and the "Compare" button is selected in the "Contigs" panel), CodonCode Aligner will use the algorithm you have chosen to create contigs of contigs. The following table gives an overview of some of the differences between the algorithms:

| <b>Muscle</b>   | <b>Clustal</b>  | <b>MACSE</b>   | <b>CodonCode Aligner's built-in algorithm</b>  |
|---|---|--|--|
| Developed and optimized for sequence alignments   | Developed and optimized for sequence alignments   | Developed for translation-aware sequence alignments  | Developed for "shotgun" sequence assemblies  |
| Always generates end-to-end alignments  | Always generates end-to-end alignments  | Always generates end-to-end alignments   | Uses local alignments that can include unaligned ends by default; can also generate end-to-end alignments and "large gap" alignments |
| Alignments formed generally include all input sequences   | Alignments formed generally include all input sequences   | Alignments formed generally include all input sequences  | Alignments may or may not include all input sequences, depending on sequence similarities and assembly parameters                    |
| Alignments can be better than Clustal alignments  | Can be slow, especially for alignments with many contigs and/or samples                               | Usually the slowest alignment program; typically gives the best placement of gaps when aligning coding sequences | Tends to be faster than Clustal or Muscle, but alignments can be sub-optimal   |
| Comparisons (alignments) can be generated for just samples, just contigs, or a mix of samples and contigs | Comparisons (alignments) can be generated just samples, just contigs, or a mix of samples and contigs | Comparisons (alignments) can be generated just samples, just contigs, or a mix of samples and contigs            | Comparisons must include at least one contig (for samples only, use "Assemble")  |
| Removes gaps from contigs before alignments (and may therefore remove bases in samples)                   | Keeps existing gaps   | Removes gaps from contigs before alignments (and may therefore remove bases in samples)                          | Keeps existing gaps  |

You can work with the new "contig of contigs" the same way you would with normal contigs. Double-clicking on a base for one of the contigs in the contig of contigs will open the trace views for this contig. You can open separate trace views for different contigs to quickly check any discrepancies.



To cite Muscle: Edgar, Robert C. (2004), *MUSCLE: multiple sequence alignment with high accuracy and high throughput*, *Nucleic Acids Research* 32(5), 1792-97. Muscle is available from <http://www.drive5.com/muscle>.

To cite ClustalW: Thompson, J.D., Higgins, D.G. and Gibson, T.J. (1994), *CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position specific gap penalties and weight matrix choice*, *Nucleic Acids Research*, 22(22):4673-4680. ClustalW is available from <http://www.clustal.org/>

To cite Clustal Omega: Sievers F, Wilm A, Dineen DG, Gibson TJ, Karplus K, Li W, Lopez R, McWilliam H, Remmert M, Soeding J, Thompson JD, Higgins DG (2011). *Fast, scalable generation of high-quality protein multiple sequence alignments using Clustal Omega*. *Molecular Systems Biology* 7:539. Clustal Omega is available from <http://www.clustal.org/>

To cite MACSE: Ranwez V, Harispe S, Delsuc F, Douzery EJP (2011) *MACSE: Multiple Alignment of Coding SEquences Accounting for Frameshifts and Stop Codons*. *PLoS ONE* 6(9): e22594. MACSE is available at <http://mbb.univ-montp2.fr/macse>

## Assemble from scratch

Sometimes, you may want to re-assemble a contig, for example after you did some editing and ended up messing up the alignment of reads. Or you may want to merge a contig with another contig and/or additional samples without preserving the existing alignment of reads in the contig.

To re-assemble one or more contigs from scratch:

- Go to the project view
- Select the contig, contigs, or contig(s) and sample(s) you want to assemble from scratch (*keep the shift-, control-, or command-key pressed to make continuous or discontinuous selections, as described [above](#)*)
- Go to the "**Contig**" menu, and move to "**Advanced Assembly...**"
- Select "**Assemble from scratch**")"

Aligner will first unassemble the contigs you selected, and then assemble all the samples in the contig(s) and any other samples you had selected. You will see a progress dialog while Aligner is assembling.

Note that the resulting number of contigs may be different from the initial number of contigs; if you want to just re-assemble contigs without merging the contig with other samples, make sure to just select one contig before choosing "Assemble from Scratch".

If your selection contained only unassembled samples, assemble from scratch will do exactly the same thing as "Assemble" would do.



# Sequence Assembly With Phrap

While CodonCode Aligner allows you to do assemblies with its own built-in assembly algorithm, Aligner also supports assembly with the assembly program Phrap (see [below for more information about Phrap](#)). Using Phrap to generate assemblies may give better results in large projects, for example for shotgun assemblies of BACs.

To assemble sequences with Phrap:

- Select the samples and/or contigs that you want to assemble in the project view.
- Go to the "**Contig**" menu, move to the "**Advanced Assembly**" submenu, and select "**Assemble with PHRAP**".

The assembly will start and show a progress window. Depending on the size of your selection, your depth of coverage, and the number of repeats, the assembly may take a while; assemblies with several hundred reads typically take a minute or a few minutes.

## How Aligner assembles using Phrap

Aligner uses Phrap to assemble your selection as follows:

1. Aligner checks if the Phrap program is indeed at the location specified in the "Phrap Assembly" preferences (for example, in the default location /usr/local/genome/bin on Mac OS X). If the Phrap program is missing, Aligner displays a warning and stops here.
2. Aligner exports the selected samples to a single sequence text file in FASTA format, and the corresponding qualities to a similar text file. If you selected any contig, then the samples in the contigs will be exported, not the contig sequence, since Phrap assumes that you always assemble "from scratch". The exported sequences will be without gaps.
3. Aligner now starts Phrap in a separate process, telling Phrap the name and location of the input files that were created in the previous step.
4. Aligner waits until Phrap is finished with the assembly. Aligner will display a progress dialog, but since it cannot be predicted how long Phrap assemblies will take, the progress window does **not** really indicate how far along the assembly is.
5. When Phrap is finished, Aligner will read the Phrap result file (the .ace file), and update the project based on the Phrap results. Any samples in your initial selection that Phrap did not include in contigs will be in the "Unassembled Samples" folder.

Please note that CodonCode Aligner is currently limited to projects with several hundred or at most a few thousand samples, even though PHRAP can handle much larger assemblies.

## Things to Note for Phrap Assemblies

While Phrap typically produces very good assemblies, Phrap also sometimes produces results that can be puzzling to the novice Phrap user. Some of these things are:

- Phrap disregards previously formed contigs, and always assembles "from scratch" (this is different from Aligner's assembly method, which leaves pre-existing contigs as they are, and looks for overlaps between the contig sequences).

- Phrap sometimes generates contigs with only one read in it. This happens if the sample has a significant overlap with other samples, but the overlap was not good enough to justify a merging. Aligner will unassemble such single-read contigs, and move the samples into the "Unassembled Samples" folder.
- For very large assemblies (or on computers with low amounts of memory), Phrap may run out of memory. Before running out of memory, Phrap may need to use virtual memory, which can slow Phrap down dramatically. In extreme cases (for example trying to assemble a bacterial genome on an underpowered computer), this can even lead to system crashes.
- Phrap tries to identify different copies of repeats during assembly, based on the quality scores of samples. This means that assemblies will be better if you have accurate quality scores (rather than just "dummy" qualities). If your samples have high-quality discrepancies, for example because you are trying to assemble samples with homozygous mutations, this may lead to a higher number of contigs than you would expect.
- Occasionally, contigs created by Phrap will contain reads that are not aligned to the consensus sequence. You can manually remove these reads in Aligner, using one of the different "Move To.." options.
- Like every assembly program out there, Phrap will occasionally produce incorrect assemblies. For example, multiple copies of highly identical repeats may all be "piled up" on top of each other. You can use Aligner's interactive features to move such misassembled reads to the trash or the "Unassembled Samples" folder.

## About Phrap

The assembly program Phrap was developed by Phil Green at the University of Washington. Phrap was widely used for sequence assembly in the Human Genome Project, and still is often regarded as the standard for sequence assembly. CodonCode Corporation distributes Phrap executables for a variety of platforms, including Windows and Mac OS X, under license from the University of Washington.

Academic users can obtain the source code for Phrap for restricted use directly from the authors at the University of Washington. For more information on this, please visit <http://www.phrap.org/>.

You can read the original documentation for Phrap at <http://www.codoncode.com/support/phrap.doc.html>. Please note that Phrap is a command line program, not a typical Windows or Mac OS X application! This makes it easy to run Phrap through scripts or from programs like CodonCode Aligner, but it means you cannot simply double-click on Phrap and expect to see a graphical user interface (there is none).

When CodonCode Aligner is installed, the installer also includes a special "workstation" version of PHRAP, which can be only from CodonCode Aligner. To run PHRAP from CodonCode Aligner, you need either a trial license, or a purchased license. Licenses purchased for academic use allow the use of PHRAP free of charge. Non-academic customers who want to use PHRAP must either install their own PHRAP executables, or pay a separate license fee for PHRAP.

# Alignments to a Reference Sequence

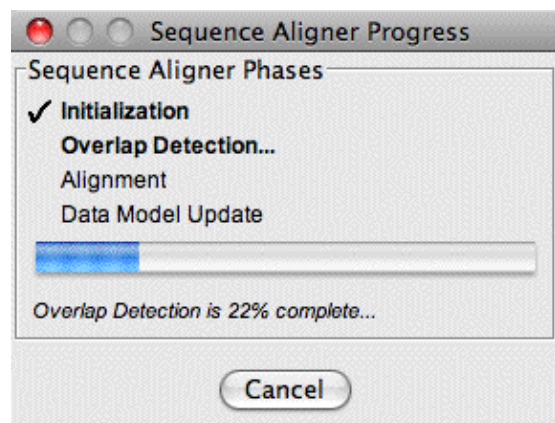
To perform an alignment to a reference sequence:

1. Open an existing project or create a new project (see [Creating Projects](#)).
2. Add the desired sample files (see [Adding Samples to a Project](#)).
3. Designate the **reference sequence**:
  - Select the reference sample in the project window
  - Choose "**Make Reference Sequence**" from the "**Sample**" menu
4. Select the reference sequence and the samples you want to align to it in the project view
5. Choose "**Align to Reference Sequence**" from the "**Contig**" menu

Note:

- To **select all unassembled samples**, simply click on the "Unassembled Samples" folder
- To make a **continuous selection**, keep the "**shift**" key pressed while clicking on the samples you want to select in the project view
- To make a **discontinuous selection**, press the "**control**" key (on Windows) or the "**command**" key (on OS X) while clicking on samples in the project view

Alignment begins immediately, and a progress window shows the status of the alignment.



When the alignment is done, you will see a newly formed contig in the project window. The new contig will contain a copy of the original reference sequence, leaving the original reference sequence unchanged.

Note that the new contig will be limited to the length of the reference sequence (plus any gaps introduced during the alignment). Any parts of sequences that extend beyond the start or end of the reference sequence will be marked as unaligned. You will not be able to see the overhanging, unaligned regions in the contig view, but you can see them in the trace views and base views.

The alignment is performed using the parameters defined by the [alignment preferences](#). Samples that do not meet the minimum criteria will not be aligned, and remain in the "Unassembled samples" folder.

Note that the alignment may be **clipped to the region covered by aligned samples**, depending on your choices in the [alignment preferences](#). The default setting is so that 100 bases to the left and the right of the first and last aligned base remain; if the aligned samples are inside a coding region that is annotated with a "codingSequence" tag in the reference sequence, then the clipping will be 100 bases to the right or left of the coding sequence region. You can choose the amount of bases left on each side, or choose not to trim the

alignment at all, in the [alignment preferences](#). If you work with very large reference sequences, for example genomic sequences for multi-exon genes, the automatic clipping after alignment can make working with the resulting contigs a lot easier, and also reduce the amount of memory and disk space needed for your projects.

If you wish to return the sample files to their unaligned state, select the alignment contig in the project view, and then choose [Unassemble](#) from the **Contig** menu. When unassembling alignments, the copy of the reference sequence that was included in the alignment will be moved to the trash, unless the original reference sequence was deleted or renamed.

You can also select more than one reference sequence for an alignment; CodonCode Aligner will align each sample to the reference sequence that it appears to be most similar to. This will typically work well as long as the reference sequences are sufficiently different (e.g. representing different genes), but may fail for reference sequences that are almost identical.

## Adding Samples to Alignments

To add new samples to existing alignments:

- Go to the project view
- Select the aligned contig, and the samples you want to add to it
- Go to the **"Contig"** menu, and choose **"Align to Reference Sequence"**.

This will start an alignment of the existing contig and the new samples.

Instead of using the menu, you can also use drag and drop in the project view to add unassembled samples to an existing alignment: select the samples you want to add, and then drag and drop them onto the contig that you want to add them to.

**Note:** All tags that had been added to the consensus sequence, including **tags** added by Aligner's mutation finding, **will be lost** when you add sequences to an existing alignment.

## Advanced Alignments

Several options for "advanced" alignments to reference sequences are available through the **"Advanced Alignments"** submenu in the **"Contig"** menu:

- [Align with Preprocessing](#) - allows you to automatically do common pre-processing steps like end clipping and vector trimming before alignment.
- [Align in Groups](#) - with this option, CodonCode Aligner can automatically group samples based on their **names**, and form separate contigs for each group. This option can be a real time saver in phylogenetic and medical studies where you look at sequences from many species, isolates, or patients. If you are working with 454 sequences, samples can also be grouped based on **multiplex sequence tags** (MID tags) at the start of the sequences.
- [Align from Scratch](#) - this will unassemble any existing contigs in your selection before starting the alignments. This option can be useful too if you want to undo manual introduction or movement of gaps, or to try different alignment parameters.

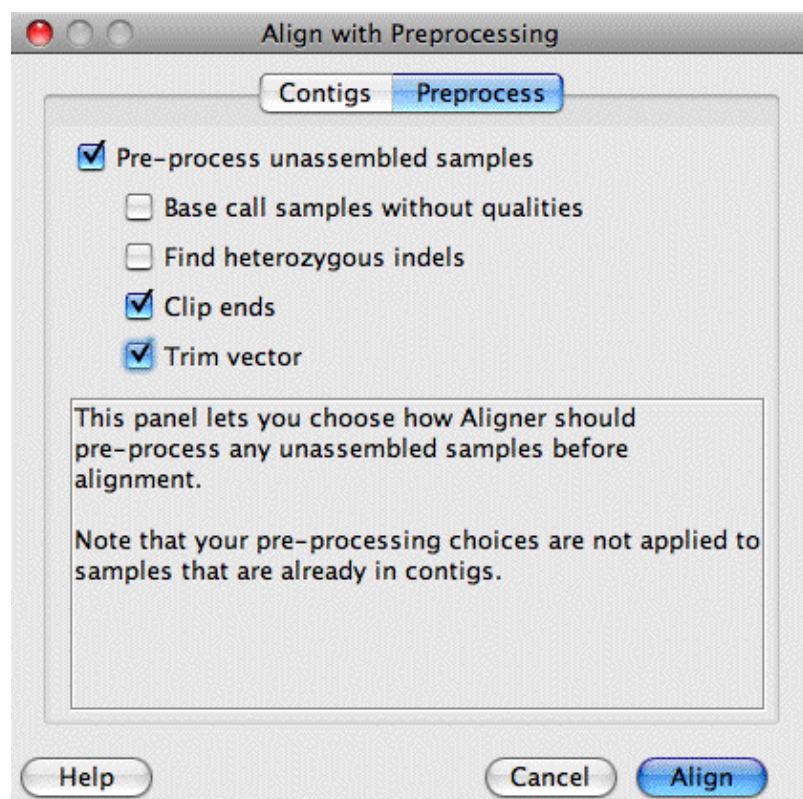
To use the advanced alignment options, select the samples you want to align, and then choose **"Advanced Alignments"** from the **"Contig"** menu. This will display a submenu where you can choose the different advanced options.

## Align with Preprocessing

Align with preprocessing lets you do common pre-processing steps like end clipping and vector trimming before aligning to a reference sequence.

1. Select the unassembled samples and contigs you want to work with in the project view.
2. Go to the "**Contig**" menu, move to the "**Advanced Alignments**" submenu, and select "**Align with Preprocessing**".
3. In the dialog that pops up, click on the "Preprocess" tab, and select the checkboxes you want to use.

Here is what the dialog looks like:



The checkbox at the top determines whether or not your samples will be pre-processed before assembly; the four lower check boxes let you pick the pre-processing steps that will be done. The choices are:

- **Base call samples without qualities:** Any samples that have chromatograms, but no base-specific quality scores, will be base called with PHRED. To use this option, you will need either a trial license or a purchased license; base calling is not enabled in demo mode. Additional information about base calling can be found at the "[Base calling](#)" help page.  
Please note that one important difference to the normal "Call bases" menu choice: in automated pre-processing, samples that already have quality values, for example from the ABI KB basecaller or from calling bases on the sample before, will not be base called again.
- **Find heterozygous indels:** This option will look for potential heterozygous insertion/deletion (indel) mutations in the unassembled samples that have (a) chromatograms and (b) quality scores. If you are sequencing PCR products from genomic DNA that may contain heterozygous indels, you should check this option; if you are sequencing from cloned DNA, this option should not be checked. For more information, please read the "[Heterozygous insertions and deletions](#)" help page.
- **Clip ends:** This will remove low-quality sequence from samples that have chromatograms and quality scores. For more information, please read the "[End clipping](#)" help page.
- **Trim vector:** When checked, this option will identify and remove vector sequence contamination from the samples in your selection. You may need to set your [vector trimming preferences](#) before using this option. For additional information, please read the "[Vector trimming](#)" help page.

You will see progress dialogs for each of the steps performed after you click the "Align" button.

Note that any samples that are already in contigs will **not** be pre-processed.

## Align to Reference in Groups

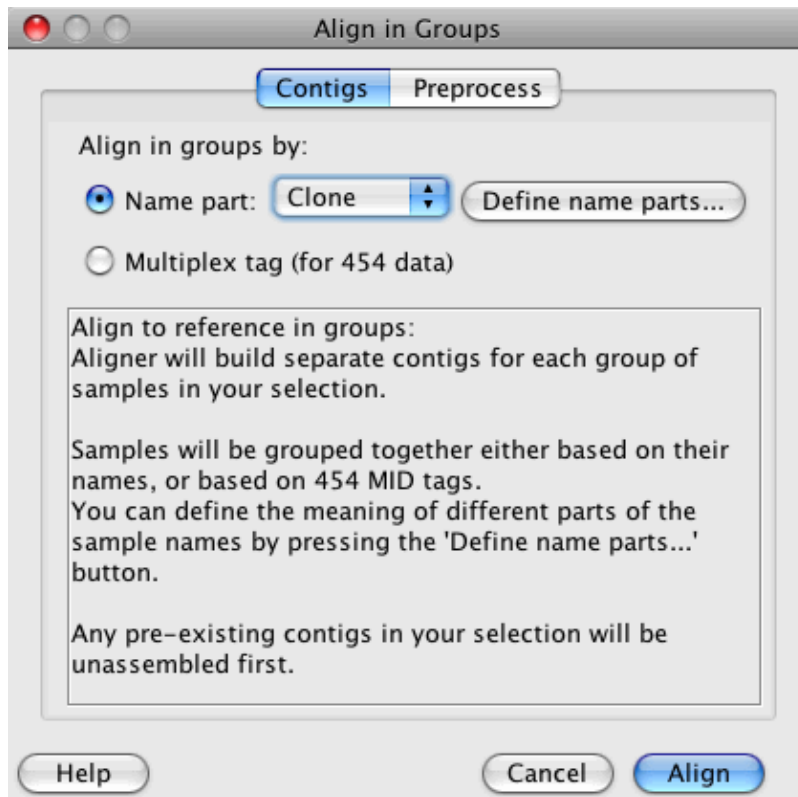
If you want to compare samples from a number of different sources (species, patients, isolates...) to one or more reference sequences, the "Align in Groups" option in CodonCode Aligner can simplify your work. "Align in Groups" uses sample names or multiplex sequence tags (MID tags) to group samples together, and to form separate contigs for each group. A copy of a reference sequence will be included in each contig.

To align to reference sequences in groups:

1. Go to the project view
2. Select the samples you want to align, and the reference sequences to use (*keep the shift-, control-, or command-key pressed to make continuous or discontinuous selections, as described [above](#)*)
3. Go to the "**Contig**" menu, move to "**Advanced Alignments**", and choose "**Align in Groups...**".

This will show the following dialog:





Using the radio buttons at the top, select to align in groups by name part or by multiplex sequence tag.

To align in groups **by name part**:

- After selecting the radio button "Name part", press the "Define name parts..." button to specify how CodonCode Aligner should interpret sample names.
- Select the sample section you want to use to group the samples, in the pulldown menu close to the top.
- For a description of how to use sample names to group samples, please see the ["Assemble in groups" help page](#).

To align 454 sequences in groups **by multiplex tag** (MID tag):

- Select the radio button "Multiplex tag (for 454 data)".
- For a description of the multiplex sequence tags and the resulting group names, please see the ["Assemble in groups by multiplex sequence tag" help page](#).

You can also choose to pre-process unassembled samples before alignment by clicking on the "Preprocess" tab, and then selecting the appropriate checkboxed, as described [above](#).

Click the "Align" button to start the alignment.

## Align to Reference from Scratch

Sometimes, you may want to re-assemble a contig, for example after you did some editing and ended up messing up the alignment of reads. Or you may want to merge a contig with another contig and/or additional samples without preserving the existing alignment of reads in the contig.

To re-align one or more contigs from scratch:

1. Go to the project view
2. Select the contig, contigs, or contig(s) and sample(s) you want to align from scratch (*keep the shift-, control-, or command-key pressed to make continuous or discontinuous selections, as described [above](#)*)
3. Go to the "**Contig**" menu, move to "**Advanced Alignments**", and choose "**Align from Scratch**"

Aligner will first unassemble the contigs you selected, and then align all the samples in the contig(s) and any other samples you had selected to your reference sequence(s). You will see a progress dialog while Aligner is aligning.

Note that the resulting number of contigs may be different from the initial number of contigs.

You can also choose "**Align to Reference from Scratch**" with just a contig selected, as long as your project contains a reference sequence.

If your selection contained only unassembled samples, align from scratch will do exactly the same thing as "Align to Reference Sequence" would (unless you also have [preprocessing](#) options selected).



Successful assemblies or alignments result in one or more contigs. Contigs are named "Contig1", "Contig2", and so on, and shown as folders in the project view.

You can edit samples in contigs, as described in the "[Editing Samples](#)" section, and you can edit contigs as described in the "[Editing Contigs](#)" section.

[illegible]

# Unassembling Contigs

To dissolve an existing contig and return the samples in it to the "Unassembled Samples" folder, select the contig in the project view, and choose "**Unassemble**" from the "**Contig**" menu (or click on the "Unassemble" button in the project window toolbar). Any gaps created by Aligner are removed, and samples that were reverse complemented are returned to their normal state.

You can also unassembled contigs by first selecting the contig in the project view, and then dragging & dropping the contig onto the "Unassembled Samples" folder. Before unassembling contigs this way, Aligner will show a warning dialog, asking you if you really want to unassemble the selected contig(s).

# Aligner Algorithms for Assembly and Alignments

CodonCode Aligner uses a fast, banded dynamic programming (Smith-Waterman) algorithm for pairwise alignments in both assembly and alignments. By default, CodonCode Aligner version 2.0.1 and newer will use "[end to end](#)" alignments; however, you can instead choose to have "[large gap](#)" or "[local](#)" alignments.

This section describes how Aligner generates assemblies and alignments to reference sequences.

## Alignment to a Reference Sequence

When aligning to a reference sequence, the selected samples are successively aligned to the reference sequence. Alignments that meet the stringency criteria set in the [alignment preferences](#) are accepted, those that do not meet the minimum criteria are rejected. If your selection contains more than one reference sequence, Aligner will align each sample to the reference sequence that shares the most "words" (typically 8-mers) first; alignments to other references will be tried only if this alignment is rejected. Typically, this will result in alignments to the reference sequence with the highest similarity; however, there can be exceptions to this rule.

When aligning to a reference sequence, any sequence parts that extend beyond the ends of the reference sequence are not included in the alignment.

## Assembly

For assembling contigs, the following simple "greedy" algorithm is used:

1. Find potential overlaps between samples by looking for shared 12-nucleotide "words" in the sequence.
2. Find the pair of samples that has the highest number of shared words.
3. Perform a pairwise alignment between the two samples (or contigs in later cycles).
4. If the alignment is good enough, keep it as a new contig, and calculate the consensus sequence; otherwise, reject the merger, and leave the two samples separate.
5. Find the next pair of sequences, looking again for the highest number of matching words. If a sample is in a contig, use the consensus sequence for the contig. If the two samples are already in the same contig, get the next pair.
6. Go back to step 3., and continue the pairwise joins until all possible joins have been tried, or until the maximum number of merge failures in a row has occurred.

The algorithm typically performs well enough for projects containing up to several hundred reads (possibly more, if you have enough memory available and a fast computer). However, it has a few weaknesses, most of which are typical for such greedy algorithms:

- If a project contains multiple copies of a repeat with high identity, there is a good chance that they are mis-assembled (but Alu sequences are generally not a problem)
- Samples have to share at least one perfect word match to be considered for joining. This means very short overlaps, or short overlaps with errors or ambiguities, may not be found. The default word length is 12 bases, but this can be changed in the [assembly preferences](#).
- Information from double-ended sequencing or other ordering information is not used to generate the assembly, which can lead to incorrect assemblies, especially in larger projects containing high-identity repeats.

You can adjust the stringency requirements for successful mergers in the [Assembly preference panel](#). (Note that the assembly preferences apply only to assemblies generated with the built-in algorithm, not for assemblies with PHRAP or contig comparisons generated with ClustalW or muscle).

## Consensus Calculation

CodonCode Aligner offers five methods for determining the contig ("consensus") sequence:

1. [Quality-based consensus sequences](#). This is the preferred method for most assembly projects, since it gives the most accurate consensus sequence. It requires that sequences have [base-specific quality scores](#), for example from [base calling](#) with Phred, which can be done directly from Aligner.
2. [Majority consensus](#). The majority consensus simply counts all bases at a given position. If one base (or gap) are more than 50% of all calls, the base (or gap) is used as the consensus; otherwise, an ambiguity code for the bases present is used (this description is a bit simplified, more details [below](#)).
3. [Inclusive consensus](#). The inclusive consensus considers all aligned bases at a given position, and uses the [IUPAC ambiguity code](#) that represents all bases present at a given location.
4. [Percentage consensus](#). The percentage consensus considers all bases at a given position that occur at least as often as the set percentage. A base that occurs less often than the set percentage at a specific position, is ignored when building the consensus.
5. [Using the reference sequence as the consensus](#). For contigs that were created by aligning to a reference sequence rather than by assembling, you can choose to have the reference sequence be used as the consensus sequence. This can be useful in mutation detection and clone verification projects.

You can set which consensus method is used in the [consensus preferences](#).

## Quality-based Consensus

To determine the consensus base and quality score for assemblies where the samples have qualities, CodonCode Aligner tries to emulate what human "finishers" typically do: Aligner identifies the highest quality base at each position, taking into account confirmations by reads in the opposite direction. Slightly simplified, here is the algorithm in detail:

1. For each possible nucleotide (A, G, C, T), find the sample with this base at the given position that has the highest quality. Take this score as the initial consensus score for the nucleotide (use 0 if no sample has this nucleotide).
2. Find the highest quality confirming base from a read in opposite direction, and add the score of the confirming base.
3. Optionally, find the highest quality discrepant base, and subtract the score. *This subtraction can be turned on or off in the "[Consensus](#)" preferences. The default setting is to not subtract the discrepant score.*
4. Calculate the score for a gap in the consensus the same way, using gaps at this position in the individual samples. The quality scores of gaps in samples are taken as the average of the two neighboring bases.
5. Pick the nucleotide (or the gap character) that has the highest quality. Assign the calculated confirmed quality score to the consensus sequence. The maximum quality score assigned to a consensus base is 90.

Several points to remember when trying to understand this algorithm:

- The base quality scores are error probabilities on a logarithmic scale; therefore, the error probabilities can simply be added *as long as the probabilities are independent*.
- Error probabilities for confirming reads in the same direction are **NOT** independent, since most sequencing errors are due to systematic problems like polymerase stops, GC compressions, etc. Therefore, qualities of confirming reads in the same direction **cannot** simply be added.
- Arguments can be made for and against subtracting the quality scores of discrepant bases. Some scientists feel that having a discrepant base should make a difference in the consensus quality, while others use statistical arguments why qualities of discrepancies should not be subtracted. You can decide for yourself, and change this in the "Preferences" (*if you have a confirmed high-quality base and a low-quality random discrepancy, the difference in the consensus quality will be zero or very small, anyway*).

The algorithm used by Aligner is similar to the algorithm used by the assembly program Phrap, but it is not identical. Phrap's algorithm is more complicated, and uses (for example) "confirmed sequence segments" rather than individual bases. However, the quality scores assigned by Aligner will often (but not always) be identical to the scores Phrap would assign to the same assembly.

## Majority Consensus

The majority method will be used to make the consensus sequence if you select "Majority" as the consensus method in the [consensus preferences](#) (unless the contig is an alignment, and you selected to use the reference sequence to build the consensus).

**The short description:** Basically, Aligner will use the most common base at each position as the consensus, unless no base accounts for more than 50% of the base calls there, in which case an ambiguity code will be used. Well, this is actually a bit simplified - here is how the majority consensus is determined in detail:

### The long description:

First, Aligner looks at all the bases and gap characters in the aligned parts of all samples at this consensus position. If more than 50% of the samples have a gap here, the consensus will be a gap. Otherwise, the gaps will be ignored for the following analysis.

If none of the base calls are ambiguities here, the rest is simple. If one base (A, G, C, or T) accounts for more than 50% of all non-gap base calls here, it is used as the consensus base. If no base accounts for more than 50%, a IUPAC ambiguity character is used, based on all the base calls here. The IUPAC ambiguity codes are:

| Ambiguity Code | Bases            |
|----------------|------------------|
| M              | A or C           |
| R              | A or G           |
| W              | A or T           |
| S              | C or G           |
| Y              | C or T           |
| K              | G or T           |
| V              | A or C or G      |
| H              | A or C or T      |
| D              | A or G or T      |
| B              | C or G or T      |
| N              | G or A or T or C |

If at least one of the base calls in the samples at this position already is an ambiguity code, determining the majority consensus is done as follows:

Each regular base call here gets a score of two. For ambiguity codes, each base represented by the code gets a score of one (for example, at a "M" call, both A and C would get a score of one). The scores for all samples that have aligned bases at this position are summed up. Now, if one of the four possible bases (A, G, C, or T) got more than 50% of the total score, this base is used as a consensus. Otherwise, an ambiguity code that represents all base calls at this position is used.

For example, if there are just two samples, one "A" and one "W" (meaning A or T), then "A" will be used for the consensus. If one base is "A" and the other "B", then "N" will be used for the consensus.

For sequencing projects where the goal is to determine the correct sequence of a gene, a quality-based consensus sequence will generally be better than a majority-based sequence. For example, if a region is covered by only two samples, the majority consensus will contain ambiguity codes at all discrepancies, while the quality-based consensus will pick the higher quality base, which is most often the correct one. However, a majority-based consensus can make sense for other types of sequencing; one example is genotyping a number of samples, where the majority sequence will show the most common allele.

### Inclusive Consensus

The inclusive consensus method looks at all aligned bases at each position, and generally uses the [IUPAC ambiguity code](#) that represents all bases present at a given location. If the majority of bases at a location are gaps, then a gap character will be used.

### Percentage-based Consensus

The percentage-based consensus method is similar to the inclusive method, except that you can set a threshold percentage for inclusion of bases in the consensus. An example where this method can be useful is the sequencing of many isolates where you want to build a consensus sequence that includes common variants, but excludes rare mutations. The percentage threshold can be set in the [consensus preferences](#) (when the percentage-based consensus method is selected). You can set different thresholds for regular contigs and for contigs of contigs.

### Using the Reference Sequence as Consensus

When you are comparing samples to a known reference sequence, you may not be interested in building a consensus sequence - you just want to know where your samples differ from the reference. Aligner supports this by letting you use the reference sequence as the consensus sequence. To use the reference sequence as the consensus sequence:

1. In the [consensus preferences](#), select "Use the reference sequence as the consensus sequence", then click "OK".
2. Create your contig by "**Align to Reference Sequence**", not by "Assemble" or "Assemble with Phrap".

Note that assembled contigs may contain the reference sequence - that alone does not mean that the contig is an alignment to a reference sequence. Only contigs that were created by "**Align to Reference Sequence**" can be alignments.

Note that if you merge an aligned contig with other contigs by using "**Assemble**" or "**Assemble with Options...**", the contig will change from an alignment to a regular (assembled rather than aligned) contig; however, you can add unassembled samples to existing alignments without changing the contig to a regular (assembled) contig.

*Tips for dealing with alignments:*

- *To find out if a contig is an alignment, look at its icon. Alignments icons have two vertical lines in the folder. Alternatively, you can check the contig information dialog. To see it, go to the project view, select "**Contig Information**" from the "**Contig**" menu.*
- *To convert a contig that contains a reference sequence to an alignment, you can select the contig in the project view, and then select "Align with options" from the "Contig" menu; in the dialog that follows, choose "Align to reference from scratch". However, the resulting contig may be different from the initial contig; for example, not all reads may have been added - some reads may have been moved to the "Unassembled Samples" folder.*

## Rebuilding the Consensus Sequence

You can rebuild the consensus sequence for any contig by selecting the contig, and then choosing "**Rebuild Consensus**" from the "**Contig**" menu. In general, you will not need to use this option, since CodonCode Aligner automatically calculates the consensus sequence, and updates it when you edit any sequence in a contig.

However, if you change the consensus method, and then open a previously saved project, using "**Rebuild Consensus**" may be advisable, since existing consensus sequences are **not** re-calculated when opening projects by default.

If you would like CodonCode Aligner to re-build consensus sequences when you open projects, go to the [consensus preferences](#), and select the radio button "Rebuild external consensus on import" in the "External Consensus" section.

## Local, large gap, and end-to-end alignments

By default, CodonCode Aligner version 2.0.1 and newer will perform "**end to end**" alignments. Assemblies and alignments generated this way will always end at the end of one sequence, not before. In contrast to the the local alignment algorithm, end-to-end alignments will never generate dangling (unaligned) ends. When using this algorithm, samples should always be end clipped, and if necessary also vector trimmed.

When using the "**Local alignment**" algorithm, Aligner automatically uses the maximum amount of each sequence, without requiring end clipping before assembly. When using local alignments, the alignment only extends as far as the alignment scores improve. This means that low-quality ends of sequences, where high error rates would reduce the alignment score, are left unaligned, or "dangling".

Unaligned parts of sequences are shown on light gray background in the contig view; the alignment end points can manually be changed using "Mark > Start Alignment Location" and "Mark > End Alignment Location" in the "Sample" menu.

*The local alignment algorithm was used by default in CodonCode Aligner version 1.6.3 and older. If you started using CodonCode Aligner before version 2.0.1, the local alignment algorithm is probably still being used, unless you changed your assembly and alignment preferences.*

When working with samples that have large insertions or deletions, or if you are trying to align **cDNA to genomic DNA**, you should change the alignment algorithm to "**Large gap**" alignment in the [assembly preferences](#) and/or the [alignment preferences](#). This will allow the alignment of multiple parts in one sequence to different sections in a second sequence - for example exons in a cDNA sequence to the corresponding regions in a genomic DNA.



# NGS Data Analysis and Assembly

CodonCode Aligner provides a number of functions to analyze, process, and assemble NGS sequencing data, using a mix of external tools and proprietary algorithms. In contrast to "classical" (smaller) sequencing projects, where all data are imported into projects before analysis, NGS methods typically work with external data files. Depending on the particular method, the analysis results may either be a new set of external data files (for example when trimming NGS sequences), or they are imported into CodonCode Aligner (e.g. contig sequences from assemblies).

## Metagenomics

- [Clustering sequences](#)
- [Identify sequences](#)

## Pre-processing tools

- [Adapter removal and read trimming with bbdduk2](#)
- [Error correction with SparseAssembler](#)

## NGS Assembly

- [CodonCode NGS Assembler](#)
- [Tadpole](#)
- [SparseAssembler](#)

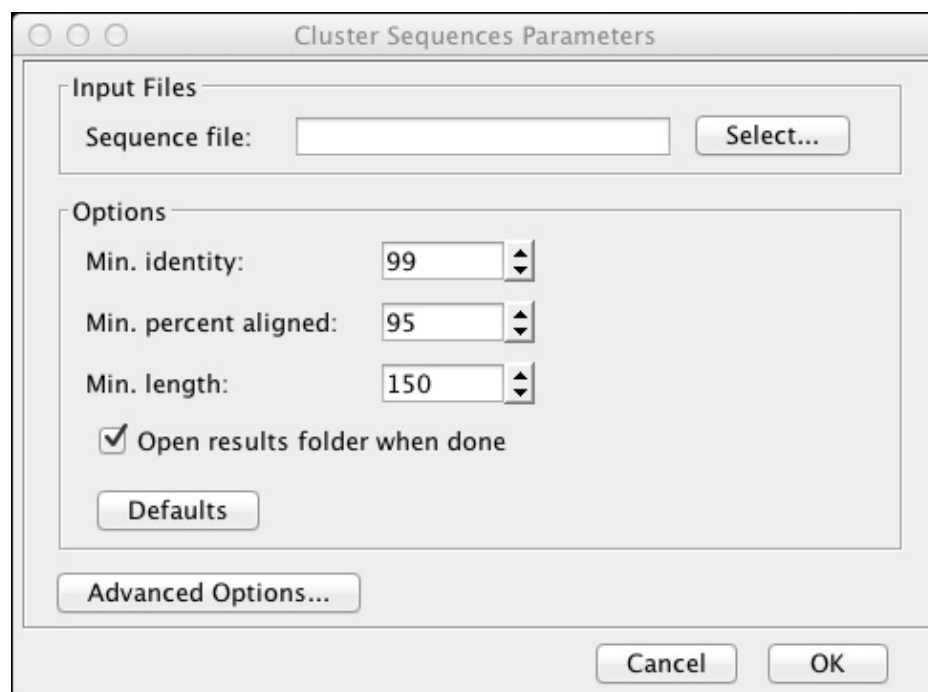
## NGS Alignments

- [Alignments with Bowtie 2](#)

*Please note that NGS analysis and assembly tools often require very large amounts of processing power and memory. The NGS tools in CodonCode Aligner are primarily intended for smaller NGS projects; large projects may require the use of dedicated processing clusters and other software.*

# Cluster Sequences

Clustering nearly identical sequences is often performed in metagenomics. To cluster sequences, go to the "Tools" menu, and select "Cluster Sequences...". This will show the following dialog:



Press the "Select..." button to choose a sequence file in FASTA format.

The "**Min. identity:**" option sets the stringency for the clustering. Sequences must have at least the given percent identity to the cluster center sequence to be included in a cluster.

The "**Min. percent aligned:**" option defines what fraction of the query or reference sequence (whichever is shorter) must be aligned. Query sequences generated by some sequencing technologies often contain errors or short contaminating adapter sequences at the end, which can reduce the identity score if end-to-end alignments are used. For this reason, CodonCode Aligner uses local alignments by default, which automatically exclude low-quality and contaminating sequences at the start and end of sequences. The default settings of 95% allows up to 5% of the sequence at the start and/or end to be unaligned. If you change the "Min. percent aligned" number to 100, CodonCode Aligner will perform global (end-to-end) alignment instead of local alignments.

The "**Min. length**" option lets you define the minimum length of the alignment. Sequences shorter than the minimum length will not be considered, and alignments that are shorter will be ignored.

The results of "Cluster Sequences" are written to several files (described in the next section) which are placed in a newly created folder. When done, Aligner will open a Finder or Explorer window to show the contents of this folder, unless the "**Open results folder when done**" checkbox is not checked.

Additional options for clustering sequences can be accessed by pressing the "**Advanced Options...**" button, which is described [below](#).

## How Aligner Clusters

CodonCode Aligner analyzes the sequences to cluster to find shared words (kmers). Starting with a "center sequence" for a cluster, any sequences with a sufficient number of shared words are compared to it using a banded alignment algorithm. Sequences where the alignment meets the minimum stringency criteria are added to the cluster; other sequences are set aside for later consideration.

The result of a clustering approach as described above depends on the order in which the sequences are clustered. To maximize the size of the clusters, CodonCode Aligner uses an additional step to find the initial cluster centers. Slightly simplified, CodonCode Aligner develops an initial estimate of possible cluster sizes for each sequence by counting the number of other sequences which have at least a given number of shared kmers (this threshold depends on minimum identity and sequence lengths). Cluster building will start with the sequence with the largest cluster size estimate, considering those sequences that have a minimum number of shared kmers. After completing the alignments for the first cluster, the next cluster center sequence will be the sequence with the highest cluster size estimate that was not previously considered for inclusion in a cluster, and so on. After a first round of clustering attempts, any remaining unclustered sequences will then be compared to the clusters from the first round; if a sequence does not match any existing cluster, it will be used as a new cluster center sequence.

Test results have shown that the additional step to find the "best" cluster centers as outlined above can yield substantially larger clusters than simpler approaches. However, the extra computations required make this approach slower, and consume more memory. Therefore, CodonCode Aligner also supports two simpler clustering algorithms, "iterative" and "kmers only", which can be chosen in the "**Advanced Options...**".

## Clustering Result Files

The results of "Cluster Sequences" are saved to several files inside a newly created folder. This folder will be created in "CodonCode/Cluster\_Sequences" in your Documents folder. The files created are:

**Cluster sequences\_log.txt** - a text file that shows the input files and parameters used and a summary of the clustering results.

**clusters.txt** - a text file with one line for each sequence, showing its assigned cluster, length, and name.

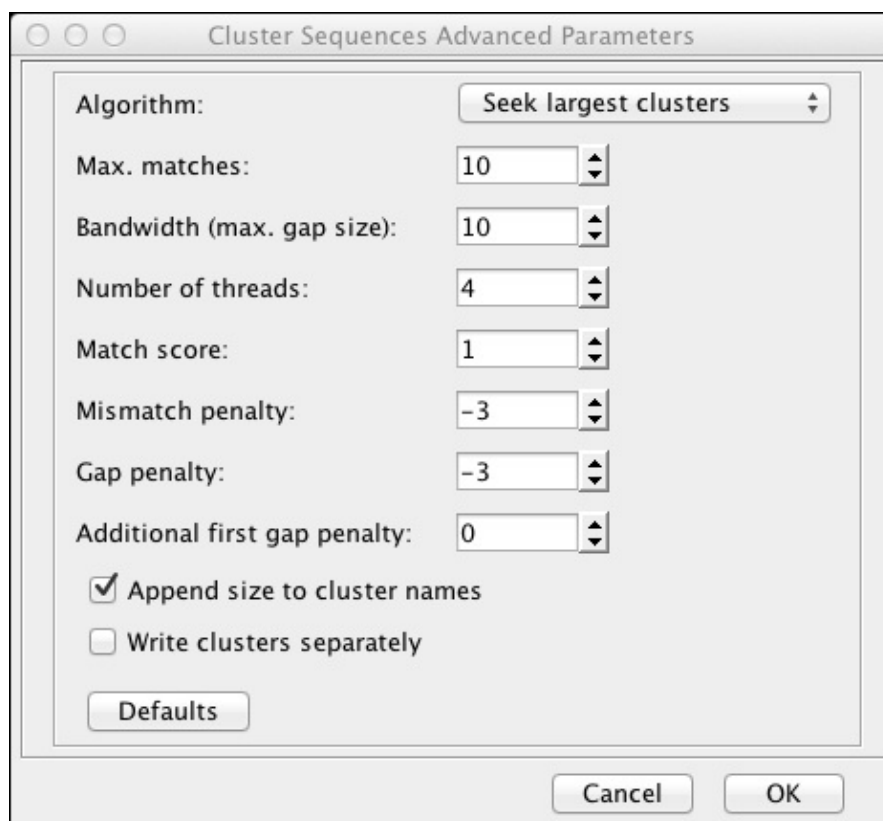
**clusterSeqs.fa** - a FASTA file with the cluster center sequences.

**singletons.fa** - a FASTA file with sequences that were not assigned to any center .

If the relevant checkboxes in the "Advanced Options" are selected, Identify Sequences can also create additional FASTA files for each cluster. which contains the sequences in it.

## Advanced Clustering Options

Pressing the "Advanced Options..." button in the "Cluster Sequences" dialog will show the following dialog:



**"Algorithm"** has three options for how Aligner clusters sequences. The default "Seek largest clusters" is described above. The "Incremental alignments" option omits the first step of finding the best cluster centers, and instead builds clusters starting with the longest sequences first. This method is usually faster, but produces smaller clusters. The third option "kmer counts only" also omits the banded alignment step, and build clusters based only on the number of shared words between two sequences. This method is fastest, but typically produces the lowest quality clusters.

**"Bandwidth (max. gap size)"** is the half width of matrix generated during banded alignments. If query sequences have more than the given number of insertions or deletions in a row, the optimal alignment may not be found. Increasing this number can make sense when dealing with long sequences that have multiple insertions or deletions. Larger numbers will lead to longer run times.

**"Number of threads"** controls the maximum number of threads used to cluster sequences. Lower numbers will lead to longer run times. However, only parts of the clustering are multi-threaded.

The next four rows determine the scoring parameters for the banded alignment.

**"Append size to cluster names"** controls whether or not the size of clusters will be appended in the cluster center result file.

**"Write clusters separately"** enables the creation of separate files with the (unaligned) sequences in each cluster.

## Limitations

## CodonCode Aligner User Manual

The "Cluster Sequences" algorithm in CodonCode Aligner currently is designed for sequence files with tens of thousands to a few hundred thousand sequences, typically shorter than one or two kb. Larger sequences files may require unusually large amounts of RAM and very long processing times, or analysis may fail.

# Identify Sequences

The "Identify Sequences" function in CodonCode Aligner is intended for metagenomics projects, for example to identify species in an environmental sample by comparing 18S RNA sequences against the SILVA database. CodonCode Aligner can identify the most similar sequences in the reference database to the query sequence, and summarize the results by taxonomy.

## How to Identify Sequences

To identify sequences, you need two files:

1. The sequences you want to analyze (the query sequences), for example a FASTA file with PacBio CCS sequences.
2. The reference sequences that your query sequences will be compared to, for example FASTA files from the SILVA RNA database.

Query and reference sequences files can be compressed with gzip or bzip2.

## Query Sequences

All query sequences should be in a single file in FASTA or FASTQ format. Sequences should be adequately pre-processed, for example to remove low quality reads and part and vector sequences. Longer reads, for example PacBio CCS sequences or merged overlapping paired-end reads, will give more accurate results than shorter reads.

## Reference Sequences

The reference sequences to compare to must be present as a single file in FASTA format. For ribosomal RNA projects, the SILVA database (<https://www.arb-silva.de>) is commonly used; reference sequence files in FASTA format can be downloaded from [www.arb-silva.de/no\\_cache/download/archive/current/Exports/](http://www.arb-silva.de/no_cache/download/archive/current/Exports/). Reference sequences should be truncated to the gene of interest, and not include alignment gaps. For example, the recommended reference sequence file for 16S and 18S RNA projects in SILVA release 132 is available at [https://www.arb-silva.de/fileadmin/silva\\_databases/current/Exports/SILVA\\_132\\_SSURef\\_Nr99\\_tax\\_silva\\_trunc.fasta](https://www.arb-silva.de/fileadmin/silva_databases/current/Exports/SILVA_132_SSURef_Nr99_tax_silva_trunc.fasta). Please note that use of the SILVA database is subject to the SILVA Terms of Use, available at <https://www.arb-silva.de/silva-license-information>. Commercial and non-academic users may need to purchase a license to use SILVA files.

It is also possible to use reference sequence files from other sources, for example COI sequences from the Barcode of Life project ([www.boldsystems.org](http://www.boldsystems.org)). CodonCode Aligner will automatically extract taxonomy information from the name lines in FASTA if the information is provided in one of the following formats:

**SILVA-format:** the sequence name is followed by a space, and then taxonomic information, with different taxa separated by semicolons. Example:

```
>AEKE02012114.67235.70246 Bacteria;Cyanobacteria;Oxyphotobacteria;Chloroplast;Solanum lycopersicum
```

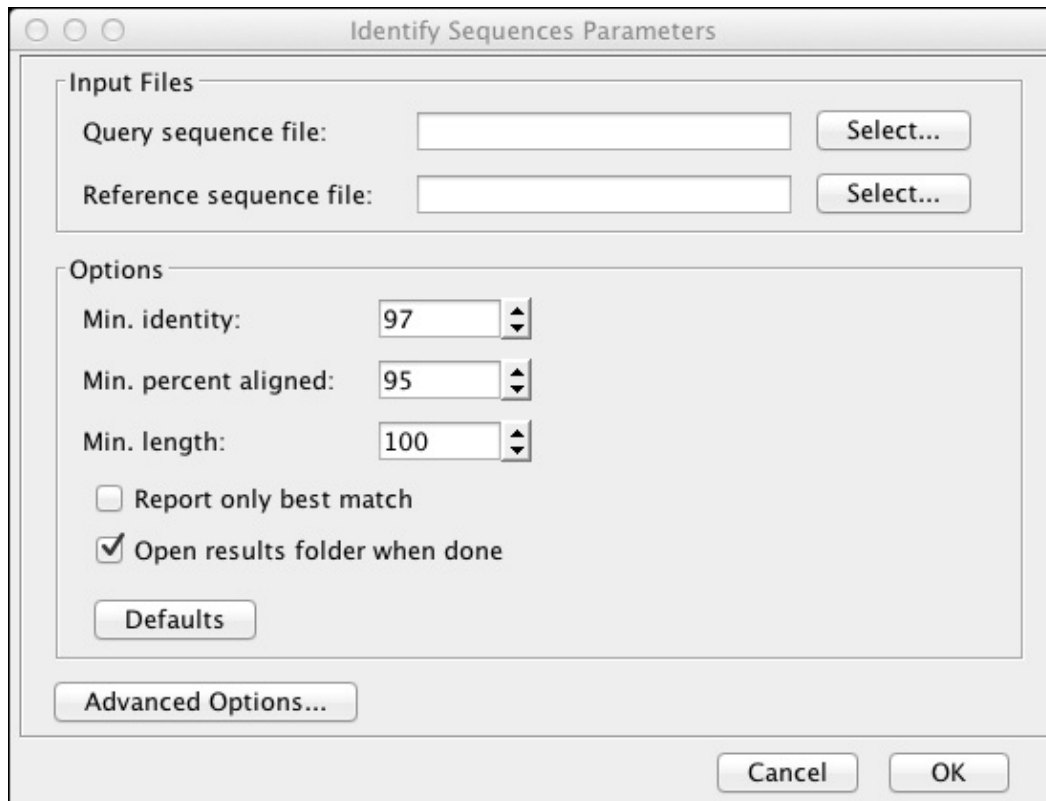
**BOLD-format:** the sequence name is followed by a single taxon name, the marker name, and (optionally) additional information; fields are separated by a vertical line. example:

```
>ABMC023-05|Antilocapra americana|COI-5P|JF443189
```

Name lines must contain at least two semicolons or vertical lines for the taxonomy format to be recognized. All entries in the fasta file must follow the same format. If the name lines in the FASTA file do not contain taxonomy information in one of the two formats described above, results will be summarized by reference sequence instead of by taxonomy.

## Stringency and Options

To start identifying sequences, go to the "Tool" menu, and select "Identify Sequences..." from the "NGS Tools" submenu. This will display the following dialog:



The dialog box is titled "Identify Sequences Parameters". It contains two main sections: "Input Files" and "Options".

**Input Files:**

- Query sequence file: [Text Field] [Select...]
- Reference sequence file: [Text Field] [Select...]

**Options:**

- Min. identity: [97] [Up/Down Arrow]
- Min. percent aligned: [95] [Up/Down Arrow]
- Min. length: [100] [Up/Down Arrow]
- ☐ Report only best match
- ☒ Open results folder when done
- [Defaults]
- [Advanced Options...]

At the bottom right are [Cancel] and [OK] buttons.

Click on the "Select..." buttons to define the query and reference sequence files. In the "**Min. identity**" field, set the required minimum identity (in percent) of the alignment between the query and the reference sequences. Alignments below the given threshold will be ignored.

The "**Min. percent aligned:**" option defines what fraction of the query or reference sequence (whichever is shorter) must be aligned. Query sequences generated by some sequencing technologies often contain errors or short contaminating adapter sequences at the end, which can reduce the identity score if end-to-end alignments are used. For this reason, CodonCode Aligner uses local alignments by default, which automatically exclude low-quality and contaminating sequences at the start and end of sequences. The default settings of 95% allows up to 5% of the sequence at the start and/or end to be unaligned. If you change the "Min. percent aligned" number to 100, CodonCode Aligner will perform global (end-to-end) alignment instead of local alignments.

The "**Min. length**" option lets you define the minimum length of the alignment. Sequences shorter than the minimum length will not be considered, and alignments that are shorter will be ignored.

"**Report only best match**" allows you to restrict the output of alignments to just one alignment per query sequence. If unchecked (the default), CodonCode Aligner will list all the alignments that it considered which match the stringency criteria.

The results of "Identify Sequences" are written to several files (described in the next section) which are placed in a newly created folder. When done, Aligner will open a Finder or Explorer window to show the contents of this folder, unless the "**Open results folder when done**" checkbox is not checked.

Additional options for identifying sequences can be accessed by pressing the "**Advanced Options...**" button, which will show the following dialog:



Identify Sequences Advanced Parameters

Max. matches: 10

Bandwidth (max. gap size): 10

Taxonomy levels to report: 12

Number of threads: 4

☐ Trim query sequence names

☐ Save sequences with matches

☐ Save sequences without matches

☐ Log rejected matches

Match score: 1

Mismatch penalty: -3

Gap penalty: -3

Additional first gap penalty: 0

Defaults

Cancel OK

**"Max. matches"** is the maximum number of alignments that will be tried per query sequence. Larger numbers will lead to longer run times.

**"Bandwidth (max. gap size)"** is the half width of matrix generated during banded alignments. If query sequences have more than the given number of insertions or deletions in a row, the optimal alignment may not be found. Increasing this number can make sense when dealing with long sequences that have multiple insertions or deletions. Larger numbers will lead to longer run times.

**"Taxonomy levels to report"** lets you control the number of levels in the taxonomy results files.

**"Number of threads"** controls the maximum number of threads used to identify sequences. Lower numbers will lead to longer run times.

**"Trim query sequence names"**: when checked, the names of query sequences will be shortened to include only the section to the first space character. If unchecked, the entire content of the FASTA name lines will be used as the sequence name.

**"Save sequences with matches"** and **"Save sequences without matches"** control whether or not Aligner will generate new FASTA files that include only the query sequences that have respectively do not have matches to the reference sequences.

**"Log rejected matches"** lets you generate a separate file with alignment summaries for sequences that only have matches which did not meet the minimum criteria.

The final four rows determine the scoring parameters for the banded alignment.

## Identify Sequences Results

The results of "Identify Sequences" are saved to several files inside a newly created folder. This folder will be created in "CodonCode/Identify\_Sequences" in your Documents folder. The files created are:

**Identify\_sequences\_log.txt** - a text file that shows the input files and parameters used, logged progress messages, and a very brief summary of the results.

**Identify\_sequences\_alignments.tsv** - a tab-separated file with information about the alignments that meet the minimum identity criteria.

**Identify\_sequences\_summary.tsv** - a tab-separated file that summarizes the alignment results by taxon, organized by taxonomic level

**Identify\_sequences\_hierarchy.tsv** - a tab-separated file that summarizes the alignment results by taxon, organized hierarchically

If the relevant checkboxes in the "Advanced Options" are selected, Identify Sequences can also create additional FASTA files for query sequences with and without matches, and a file with alignment information for query sequences that only have alignments that did not meet the minimum criteria.

## Identify Sequences Algorithm

To identify matching sequences in a reference database, CodonCode Aligner first identifies potential candidates by looking at shared kmers, 11 base long nucleotide sequences. The minimum number of shared kmers is calculated based on sequence length and minimum identity, with an absolute minimum of 2 shared kmers. Candidate reference sequences are then sorted by the number of shared kmers, and compared to the query sequences using a banded pairwise alignment. By default, up to 10 alignments are calculated per query sequence; this number can be changed in the "Advanced Options". Alignments that match the minimum requirements are then sorted by alignment score, and logged to the results file.

For taxonomic summaries, the taxonomic information is extracted and parsed from the name lines in the FASTA file for the reference sequences. If the name lines do not contain taxon information as described in the "Reference Sequences" section, summary reports will be based on reference sequence names instead.

## Limitations

The "Identify Sequences" algorithm in CodonCode Aligner has been developed and tested with PacBio CCS sequences, which provide longer read lengths and lower sample numbers than, for example, Illumina sequencing. Memory use and run times are proportional to the size of the reference data base and the query sequence files; processing query files with millions of sequences may not be possible, or extremely slow, on typical computers.

# NGS Preprocessing Tools

CodonCode Aligner can use external programs to pre-process NGS data, for example Illumina sequences. Pre-processing before assembly can improve the assembly quality dramatically. The following pre-processing functions are included with the CodonCode Aligner installation:

- [Adapter removal and read trimming](#) with bbdduk2
- [Error correction](#) with SparseAssembler

NGS tools generally work with external data files (typically in FASTQ format), and write the processing results to new files. Which of the tools should be used depends on the nature of the input data: for high coverage projects with good data quality, removing adapter sequences and quality-based trimming can be sufficient; for data with lower coverage or higher error rates, error correction may be required, possibly in addition to adapter removal and trimming.

*Please note that NGS error correction and assembly in CodonCode Aligner is currently **only intended for small NGS projects, for example bacterial genomes**.*

## Trimming Reads and Removing Adapter Sequences

Removing low-quality sequences and adapter contaminations (for example from long sequences extending through short inserts) is often essential to obtain good quality assemblies. For example, failure to remove adapter sequences from "read-through" of short insert sequences can increase the number of contigs dramatically, and reduce average contigs lengths by tenfold or more.

CodonCode Aligner includes the program [BBduk2](#) for quality and adapter trimming, and provides a graphical user interface to simplify its use. BBDuk2 is part of the "BBTools" package developed at the Joint Genome Institute (<http://jgi.doe.gov/data-and-tools/bbtools/>), and offers a variety of functions for quality trimming, adapter removal, GC and entropy filtering, histogram generation, and more.

To trim NGS sequences with BBDuk2, go to the "**Tools**" menu, select the "**NGS Tools**" submenu, and choose "**Trim Reads with BBDuk2**". This will open a dialog where you can specify the input file(s) and trim options:

Trim Reads With BBDuk2 Parameters

**Input Files**

Input file:

2nd file (for paired files):

**Output Files**

Save trimmed reads to:

2nd results file (for paired reads):

Singletons file (for paired reads):

**Adapter Removal Options**

☒ Remove right adapters; adapter file:

☐ Remove left adapters; adapter file:

Min. adapter match at read tips:

**Quality Trim Options**

Quality trim mode:

Minimum trim quality:

Minimum average quality:

Press the "Select" button at the top to choose the file you want to process. If you have paired end reads in 2 separate files, define the second file in the second line. You can also select where the trimmed sequences will be saved in the "Output Files" section. If you do not specify a location for the result files, a single result file will be created in the folder off the original sequence file, using the name of the sequence file with "\_BBDuk2" appended before the extension.

**Adapter trimming** can be done independently on the right and left side of sequences against all sequences in a FASTA-format file. One such file from the BBTools distribution that includes standard Illumina adapter sequences is included with the CodonCode Aligner distribution, and will be shown in the "adapter file:" fields when you use BBDuk2 the first time. If you use adapter sequences that are not included in this file, you can either add them to the file, or change the adapter file(s) used.

**Quality trimming** can be done from both ends, just the left or just the right end, or using a window-maximizing algorithm, in the "Quality trim mode:" pulldown. You can also choose to not quality trim sequences. The "Minimum trim quality" number determines the trimming stringency; higher numbers mean higher stringency and more removed bases. If you want to discard sequences that still are low quality after trimming, you can use the "Minimum average quality" box to define a threshold; sequences with average qualities below this threshold will be discarded.

The dialog shown above will often be adequate for read trimming. If you need additional control over trimming parameters, or want to create additional output files (e.g. for quality histograms), click the "**Advanced Options...**" button. This will open a second dialog with "expert" options:

Trim Reads With BBDuk2 Advanced Parameters

**General Options**

Save progress messages to:

k-mer size:

# reads to process (-1 means all):

☐ **Output Options**

☐ **Statistics Output**

☐ **Trimming Options**

☐ **Processing Options**

☒ **Speed and Memory Options**

Number of threads:

☐ Preallocate memory

Monitor process:

minrskip:

maxrskip:

rskip:

qskip:

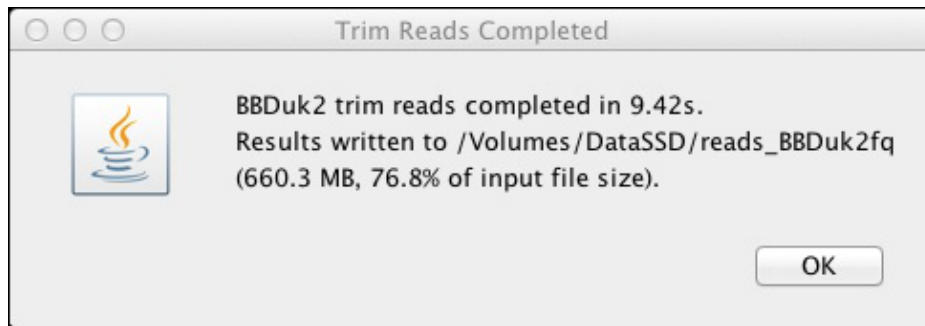
speed:

☐ **Entropy/Complexity Options**

The advanced options are divided into several groups, which can be expanded or hidden by clicking on the little arrows before the group names. In the example above, the "Speed and Memory Options" are shown expanded, while "Output Options" and most other groups are folded (hidden).

Hovering over an option with the mouse pointer will show a tooltip that gives additional information about the option. *Options that are shown only by the name of the command line option (like "minrskip" in the example above) are intended only for expert use - if you don't know what the option is for, you probably should not change it!*

After clicking "OK" in the dialogs, CodonCode Aligner will start BBDuk2, and show progress messages that BBDuk2 generates. When BBDuk2 finishes, CodonCode Aligner will show a summary dialog:



If the trimming encounters any problems like corrupt input files, invalid parameters, or insufficient memory, you can check the log file for additional information. The default location of the log file is in your "Documents" folder in a sub-directory "CodonCode/BBDuk2". Please note that this log file will be overwritten every time you trim reads; if you want to keep a log file, you can specify a different name and location in the "Advanced Options dialog.

## Error Correction with SparseAssembler

Error correction reduces the number of random errors in the sequence data, and can result in better assemblies with fewer contigs and errors. One program for correcting errors in NGS reads, SparseAssembler (Ye et al. BMC Bioinformatics 2012, 13(Suppl 6):S1), is included in the CodonCode Aligner installation. *Please note that NGS error correction with SparseAssembler requires a 64-bit operating system, and is not available on 32-bit Windows.*

To error NGS data with SparseAssembler in CodonCode Aligner, create a new project, go to the "**Tools**" menu, select the "**NGS Tools**" submenu, and then "**Error Correct with SparseAssembler...**". This will open a dialog where you can select the input files, and adjust parameters. Input files must be in FASTQ format. Reads from pair-end and mate-pair sequencing should be in separate files (all reads from one end in one file, and all reads from the other end in a second file, with exactly the same read order).

CodonCode Aligner will then start SparseAssembler, and display the progress output generated during error correction in a dialog. SparseAssembler will create new files with the error-corrected read sequences, which will be placed in a new folder. This folder is located in the "CodonCode/SparseAssembler\_Denoiser" folder inside your Documents folder. SparseAssembler will create a new file for input file by added "Denoised\_" to the start of the file name. If the input sequences contained read pairs in two separate files, SparseAssembler will also create a third file named "Denoised\_Single\_" followed by the input file name which contains sequences where one of the two paired sequences was rejected during error correction (for example because of low quality).



## Other NGS Tools

CodonCode Aligner also supports the use of other NGS tool programs, for example for error correction, through a simple, XML-file based mechanism. To add another program, a new folder for the program must be created inside the "NGSTools" folder in the "HelperPrograms" folder where CodonCode Aligner is installed. The program executable is placed in this folder, and the parameters for the program are defined by creating an XML file in this folder.

If you are interested in using a different program for NGS data error correction in CodonCode Aligner, please contact CodonCode Corporation's support team.

# NGS Assembly

CodonCode Aligner provides the following functions for assembly of NGS data (e.g. Illumina reads):

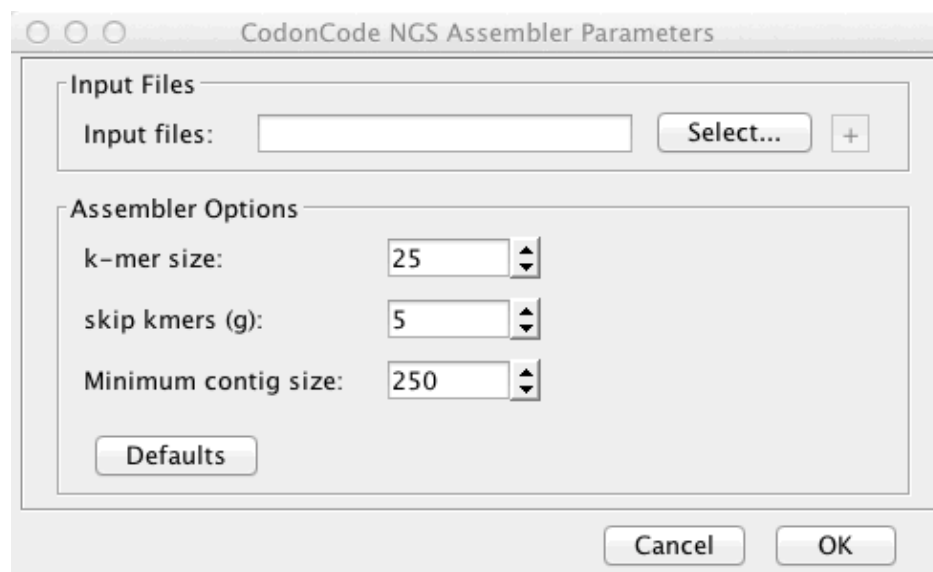
- [Assembly with the CodonCode NGS Assembler](#)
- [Assembly with Tadpole](#)
- [Assembly with SparseAssembler](#) (on 64-bit operating systems only)
- [Scaffolding](#)

All of these functions are currently intended only for **small** sequencing project like bacterial genomes. Before assembly, NGS sequence data should be pre-processed: trimmed and, if necessary, error corrected. Using "raw" NGS data will often result in a large number of very small contigs. Pre-processing tools are available in CodonCode Aligner through the "[NGS Tools](#)" submenu in the "Tools" menu.

## NGS Assembly with CodonCode Aligner

To assemble NGS data in CodonCode Aligner using CodonCode Aligner's built-in algorithm:

- Create a new project.
- Go to the "Tools" menu, and select "**CodonCode NGS Assembler...**" from the "**Assemble NGS Data**" submenu (*on 32-bit versions of Windows, select "Assemble NGS Data..." from the "Tools menu*).
- This will open a dialog where you can specify the NGS data files and assembly parameters:



Press the "Select" button to choose your input file in FASTA or FASTQ format. If you have several input files, use the little "+" button next to the "Select" button. You can also specify the following parameters for the assembly:

- **K-mer size:** The word size used for the assembly (default: 25; maximum: 31)
- **Skip k-mers:** The distance between k-mers used for the assembly. Larger numbers will reduce memory requirements for the assembly, but can lead to more contigs.
- **Minimum contig size:** Any contigs generated by the assembly shorter than the given size will be ignored. This number should be larger than the typical read length.

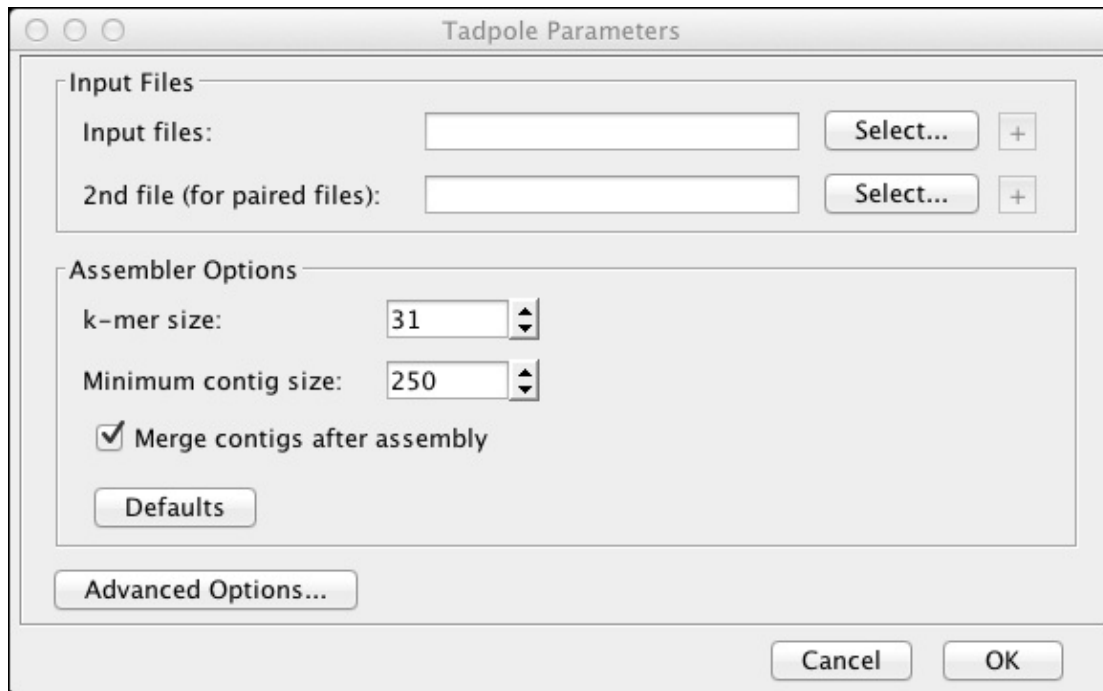
After pressing the "OK" button, CodonCode Aligner will start the assembly, and display some progress messages in a dialog. Assemblies of bacterial genomes typically take several minutes or longer, depending on the size of the data set. When the assembly is complete, the resulting contigs will be imported.

CodonCode Aligner's built-in **NGS assembly algorithm** uses a Debrujin-graph based approach that first builds "unitigs" using k-mers that are distributed evenly, following an approach similar to SparseAssembler. After building unitigs, CodonCode Aligner uses additional information from the entire reads to resolve ambiguous branches between unitigs, thereby creating larger contigs. The resulting contigs are typically larger than those created by SparseAssembler. Typical contig sizes depend on read length; best results are obtained with longer reads (e.g. 250 bases).

## NGS Assembly with Tadpole

Tadpole is a fast NGS assembler that is part of the [BBTools](#) package from the Joint Genome Institute. Compared to the CodonCode NGS assembler, Tadpole is more aggressive in resolving branch points. This can result in assemblies with fewer and larger contigs, but also in more misassemblies (false joints). Tadpole can also use k-mer sizes larger than 31, the current maximum for the CodonCode NGS assembler.

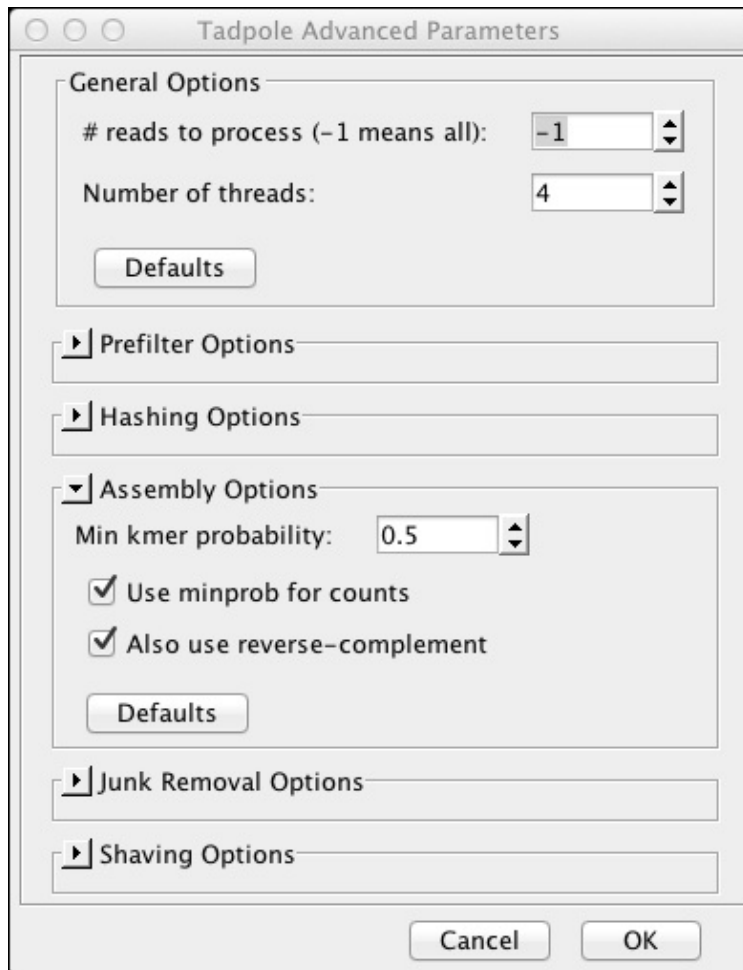
To assemble NGS data with Tadpole in CodonCode Aligner, create a new project, go to the "**Tools**" menu, and select "**Tadpole...**" from the "**Assemble NGS Data**" submenu. This will open a dialog where you can specify the input files and assembly parameters:



Press the "Select..." buttons on top to choose the data files in FASTQ or FASTA format to assemble. If you have paired-end reads in separate files, define the second file for each pair of files on the second line.

The "Merge contigs after assembly" option can be enabled to use part of the functionality in the CodonCode NGS Assembler to reduce the total number of contigs, and increase contig sizes, after the initial assembly with Tadpole. This function uses information from entire reads to bridge some otherwise unresolved branch points, specifically branches caused by repetitive sequences that are as least as long as the chosen k-mer size, but shorter than the typical read length. The effect of using this function depends on the genome on the read length; for bacterial genomes with frequent very short repeats and read length of 250, the reduction in contig numbers and concurrent increase in N50-values can be 10-fold or higher.

You can define additional "expert" options for Tadpole by pressing the "Advanced Options..." button, which will bring up a second dialog:



The advanced options are grouped and shown in panels that can be expanded and collapsed by clicking on the little triangle in front of the group name.

After defining the input files and options, click "OK" to start the assembly with Tadpole. CodonCode Aligner will open a progress dialog that shows the progress messages from Tadpole while the assembly is running, and will import the resulting contigs when the assembly is finished.

## NGS Assembly with SparseAssembler

To assemble NGS data with SparseAssembler in CodonCode Aligner (available only on OS X or 64-bit Windows):

1. Create a new project.
2. Go to the "Tools" menu, and select "SparseAssembler..." from the "Assemble NGS Data" submenu.
3. In the dialog that opens, select the input files for the assembly, and adjust assembly parameters.
4. Press "OK" to start the assembly. Assemblies will typically take several minutes or longer, depending on the amount of the data.

Input files must be in FASTA or FASTQ format. Reads from pair-end and mate-pair sequencing should be in separate files (all reads from one end in one file, and all reads from the other end in a second file, with exactly the same read order).

While running SparseAssembler, CodonCode Aligner will display a dialog that shows the progress output generated. When the assembly is complete, the resulting contigs will be imported as text sequences into the project. *By default, only contigs with a minimum length of 500 bases are imported; the minimum length can be changed by editing "MinContigSize" parameter in the "SparseAssembler.xml" file, which is located in HelperPrograms/ExternalAssemblers/SparseAssembler in the CodonCode Aligner directory.*

If mate-pair sequences ("outward paired reads") were included in the input, CodonCode Aligner will use the mate-pair information to find links between contigs, and form scaffolds. It does this by mapping reads to the contig sequences with Bowtie2; analyzing the mapping information for mate pair links between contigs; and looking for sequence overlaps between linked contigs. The resulting scaffolds will be imported as contigs into the project.

## Other NGS Assembly Programs

CodonCode Aligner also supports the use of other NGS assembly programs through a simple, XML-file based mechanism. To add another assembler, a new folder for the program must be created inside the "ExternalAssemblers" folder in the "HelperPrograms" folder where CodonCode Aligner is installed. The program executable is placed in this folder, and the parameters for the program are defined by creating an XML file in this folder.

If you are interested in using a different assembly program in CodonCode Aligner, please contact CodonCode Corporation's support team.

## Scaffolding

For NGS sequencing projects that include mate-pair sequences, CodonCode Aligner can form scaffolds of contigs by analyzing the linking information provided by the mate pairs. To create scaffolds:

## CodonCode Aligner User Manual

1. Create a new project, or open an existing project.
2. Go to the "**Tools**" menu, and select "**Scaffold NGS Contigs...**".
3. A dialog will be shown where you can define a pair of files with the sequence information for the reads (one file for each end). Scaffolds can be built either for contig sequences in a file, or for sequences that are already in your project, and selected before starting the scaffolding.
4. Press the "OK" button.

CodonCode Aligner will then map the reads to the contig sequences using Bowtie2. Information from mate pairs where both ends can be mapped unambiguously to one contig is then used to find links between contigs, and to create scaffolds. Contigs that contain highly identical repeat sequences, for example rRNA sequences, will typically remain as single contigs after this step. Scaffolds will then be imported as contigs into CodonCode Aligner.

# Separate Alleles

The "Separate Alleles" function in CodonCode Aligner is intended identify linkages between alleles in small NGS sequencing projects from diploid organism. A typical project might start with a known sequence for a small genomic region of several kb, and a Blast search of a sequencing project in the NCBI Sequence Read Archive to find reads for the genomic region. Such a project would typically contain several hundred to thousand NGS sequence reads.

## How to Separate Alleles

To separate alleles in CodonCode Aligner, you can either start with a reference sequence and a collection of NGS reads, or with a contig that includes a reference sequence. To start from unassembled reads, follow these steps:

- Create a new project.
- Import the reference sequence and the reads, and select them.
- Go to the **"Tools"** menu, and select **"Separate Alleles"**.
- Wait until CodonCode Aligner finishes the analysis, and shows a summary of the results.

The end result of the analysis is an alignment of the contigs for the linked haplotypes to the reference sequence. If all SNPs could be linked, the alignment will contain only two contigs, one for each haplotype. If the linkages do *not* extend over the entire contig, for example because longer regions do not contain SNPs, then the alignment will contain multiple contigs that each cover only a part of the reference sequence. The sequences for the individual alleles or allele fragments can be exported using the "Export Consensus Sequences" option from the "File" menu.

## Algorithm Details

When separating alleles, CodonCode Aligner performs the following steps:

### 1. Identify mate pairs.

Aligner analyses the reads in the selection to identify mate pairs - typically the forward and reverse sequences from single inserts. In general, the reads for a pair should and in the numbers 1 and 2, preceded by either a forward slash ('/'), a period ('.'), or an underscored. In addition, Aligner supports read pairs where both reads have identical names; when such reads are imported, the second read name is modified by appending "\_1".

The reference sequence can be identified automatically if it is significantly longer than the reads; alternatively, references sequences can be designated manually by using the "Make Reference Sequence" item in the "Sample" menu.

### 2. Align read pairs to the reference sequence using Bowtie2.

The read pairs are aligned to the reference sequence using Bowtie2. The Bowtie2 parameters used have been optimized (which includes end-to-end alignments and modified gap penalties of (5,3) for reads and (9,3) for reference gaps). The resulting contig is imported and named "Unphased".

### 3. Find SNPs.

To identify single nucleotide differences (SNPs, which in this context include insertions and deletions) between the alleles, Aligner examines the reads at each position in the contig. To qualify as a SNP for further analysis, differences that are seen in just one read, or fewer than 10% of the reads at a given position, are ignored; in other words, SNPs must be present in at least two reads, and at least 10% of the reads.



**4. Find linked SNPs, assigning reads to alleles.**

In this step, the two observed bases at each SNP are assigned to one of two different alleles, using a simple iterative-greedy algorithm. Initially, the read pair that contains the highest number of SNPs is chosen as a starting point. The bases observed at the SNPs in this read pair are assigned to allele 1, and the bases not observed are assigned to allele 2. Next, the sequence with the highest number of SNPs that have already been assigned to an allele is identified, and assigned to either allele 1 or allele 2, depending on the observed bases at the SNPs. If both alleles are observed the same number of times in a read pair, which can happen due to sequencing errors or alignment errors, the pair is removed from further analysis. When an assignable read pair includes a SNP that has not been previously assigned to an allele, the observed base is assigned the allele number of the read, thereby extending the length of the linked haplotype regions. When no more read pairs that overlap the currently defined region can be found, a new starting read pair is identified, which creates a new "group" of two linked alleles, followed by extension as described above. This is repeated until all read pairs with SNPs are assigned to an allele. If all SNPs can be successfully linked, the result will be one group of two linked alleles that represent the estimate of the underlying haplotypes. If there are any gaps in the linkage, two or more groups of regions with 2 linked haplotypes will be created.

**5. Create allele-specific contigs.**

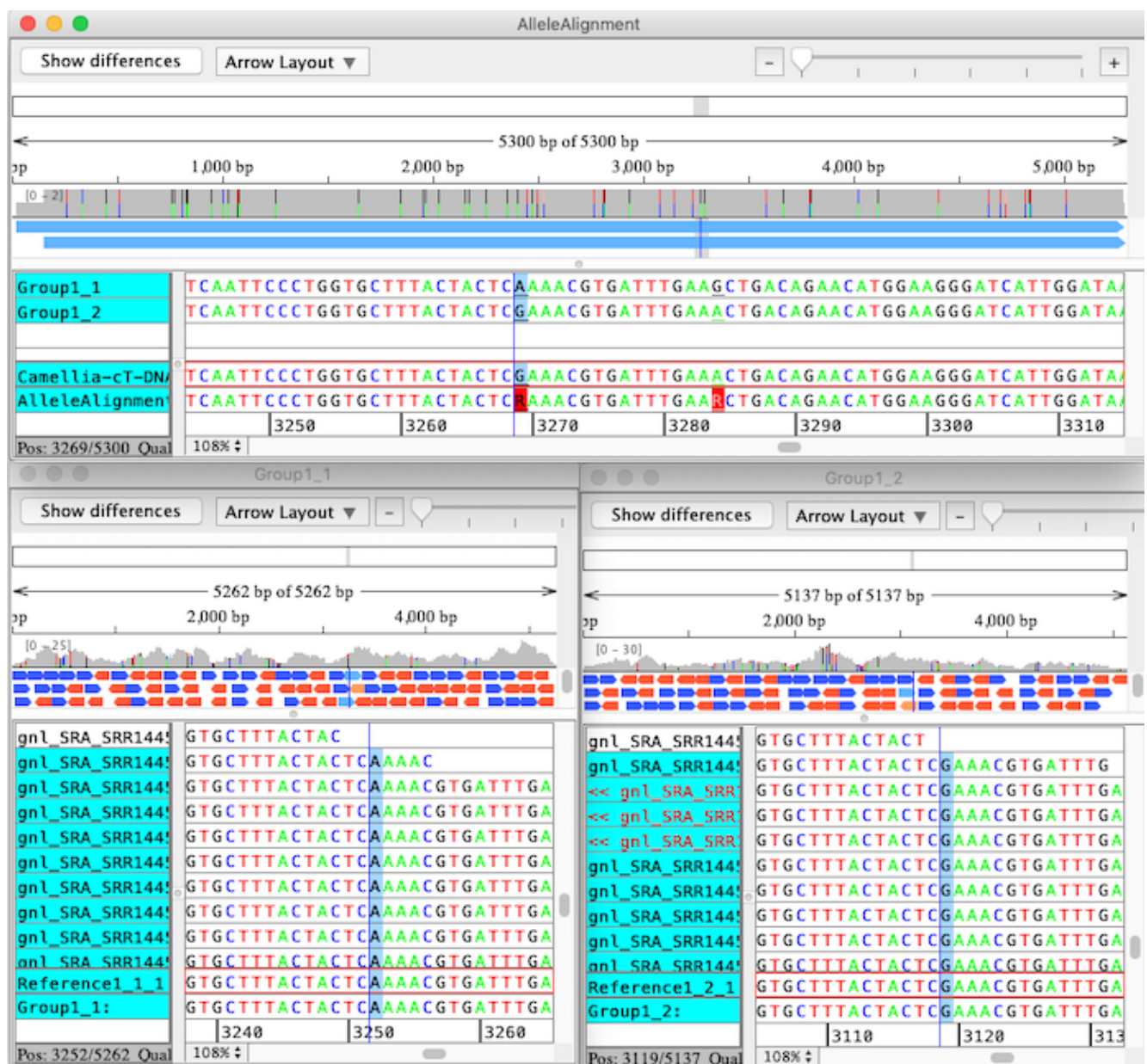
The goal of this step is to create haplotype-specific contigs, using the linked read pairs from the previous step. Similar to step 2, these contigs are generated by aligning read pairs to a reference sequence with Bowtie2, but here, the reference sequences are modified at all SNP positions to reflect the deducted haplotype. In the absence of all errors, this alignment would be trivial, with all reads matching the reference sequence perfectly. With real world data, however, the alignment will show random sequencing errors in the reads, and possibly also alignment errors, and errors due to analysis failures in previous steps.

The contigs created during this stage are named "Group1\_1" and "Group1\_2" for the first set of linked alleles. If not all SNPs can be linked, additional contigs named "Group2\_1", "Group2\_2", and so on are created.

**6. Align allele-specific contigs to reference sequence**

The final step is to align all contigs from the previous step to the original consensus sequence, using CodonCode Aligner's "Compare Contigs" function using the built-in algorithm. The resulting "contig of contigs" is named "AlleleAlignment", and shows the alignment of the allele-specific contigs to the reference sequence (which is ignored when the consensus sequence is calculated).

The screen shot below shows an example of the end results, with the contig view for the final contig on top, and the contig views for the two allele-specific contigs below:



Contig views for the child contigs (the two alleles specific contigs) can be opened by double-clicking on a contig in the parent contig view. The placement of the child contig views relative to the parent can be controlled using the "Window Placement" preferences.

## Starting from Contigs

In the description above, separating alleles started from unassembled reads. However, it is also possible to start from existing alignments to a reference sequence. Typically, such alignments should be generated using Bowtie2, although this is not an absolute requirement. One reason to start with a pre-existing contig is that this allows the editing of a contig to remove misalignments before further processing. Such misalignments can frequently happen near the start or end of reads, especially if one of the two alleles has insertions or deletions relative to the reference sequence. In such cases, alignment approaches with "general" scoring matrices often cannot produce an accurate alignment near the end, since the penalties from gaps or nucleotide differences exceed any positive scores from aligned bases. Such regions may introduce errors in the analysis; however, they can often easily be edited by hand, or simply clipped off.

When the analysis of the final results indicates that errors may be due to misalignments, one possible way to correct these errors is by manual editing of the "Unphased" contig from step 2 above, and then select the edited contig and choose "Separate Alleles" again.

## Limitations

Separate Alleles is a new function in CodonCode Aligner, with a number of significant limitations. Any feedback for future improvements is welcome. Some of the known limitations include:

- **Project size limitations:** Currently, only projects with several hundred to a few thousand reads should be analyzed. The analysis is likely to fail and/or take a very long time for larger projects.
- **Mate pairs required:** Paired end reads (mate pairs), for example from Illumina reads, are required, and must follow one of the naming schemes described above.
- **Wrong links:** Some of the links between SNPs may be incorrect due to sequencing errors, alignment errors, or other factors. The current implementation of the "phasing" algorithm is simple, and more prone to errors than some more sophisticated algorithms that have been described in the scientific literature.
- **Missing links:** The algorithm may miss existing links between SNPs for a number of reasons, which include low coverage for one or both alleles, sequencing errors, and misalignments.
- **Missed large-scale variations:** The algorithm can handle only single nucleotide polymorphisms and relatively short insertions and deletions, but not larger indels or rearrangements.
- **Diploid regions only:** The analysis is limited to diploid organisms, where both haplotypes are represented in roughly similar numbers in the reads. Data from where more than two haplotypes are present, or where coverage of the alleles is very different, cannot be analyzed.
- **Side effect of settings changes and edits:** Changing the "Consensus Method" preferences in CodonCode Aligner may have unexpected and undesirable consequences for contigs that result from the "Separate Alleles" method, since any changes will trigger a rebuilding of the consensus sequences in the currently open project. Similarly, editing sequences in contig often require a partial or complete rebuilding of the consensus sequence, which may affect larger regions in a contig, especially when gaps are added or removed, or samples are removed. If the current consensus method settings differ from the settings used by the "Separate Alleles" function, this can lead to unexpected changes, and to consensus sequences that are "pieced together" using *different* consensus methods. This can be corrected by making sure that the desired consensus method is selected in the preference, and by using the "Rebuild Consensus" function from the "Contig" menu.

## Files Generated

The "Separate Alleles" function generates a number of files, which are stored in time stamped folders inside the "Alleles" folder in the "CodonCode" folder in your documents folder. These files include:

- UnphasedMates1/2.fasta: Input files for the initial Bowtie2 alignment that creates the "Unphased" contig.
- GroupX\_YMates1/2.fasta and Re: Input files for generating the allele-specific contigs with Bowtie2.
- Reference1\_1/2.fasta: Reference sequence files with allele-specific changes, used to generate the allele-specific contigs. Note that the consensus sequences for these contigs may or may not differ from the reference sequences!
- Alleles.csv: A comma separated file that shows the positions of SNPs used during the analysis, and the assignment of alleles to the read pairs.
- ProblemSNPs.txt: A text file that lists SNPs which may be "false" SNPs due to sequencing or alignment errors. If, for example, two read pairs have the same random error at a given position, the error base may be falsely identified as a SNP. When identifying linked SNPs, these false SNPs can lead to read pairs which cannot be assigned to an allele because they have the same number of bases classified as "allele 1" and "allele 2". When such read pairs are encountered during analysis, the positions of the SNPs in these read pairs are listing in the ProblemSNPs file to facilitate further analysis.

Files and folders in the "Alleles" folder that are not needed for further analysis should be deleted by the user.

# Alignments with Bowtie 2

Bowtie 2 is a fast and memory-efficient tool for aligning sequencing reads to long reference sequences. Bowtie2 is available from <http://bowtie-bio.sourceforge.net/bowtie2/>; executable versions of Bowtie 2 are installed into the "HelperPrograms" folder when you install CodonCode Aligner 5.0 or newer.

Bowtie 2 can be run directly from CodonCode Aligner. Since Bowtie 2 is intended for large data sets, it is generally used with *external* data files. In other words, the sequences to be aligned are not imported into a CodonCode Aligner project first. Instead, the alignment results are imported into CodonCode Aligner projects.

Please note that CodonCode Aligner is currently not intended for working with large genomes or very large datasets. While Bowtie2 can align large data sets to entire genomes, importing the results into CodonCode Aligner will typically not be possible. On most computers, CodonCode Aligner can work with alignments of several hundred thousand short reads to single chromosome; on computers with large amount of memory (RAM), working with several million reads may be possible.

To perform an alignment with Bowtie2:

1. Open an existing project or create a new project (see [Creating Projects](#)).
2. Choose "**Align with Bowtie2...**" from the "**Contig**" menu

This will display the following dialog:

**Bowtie2 Alignment**

**Align to:**

- ☐ Selected sequence
- ☒ Sequences from file
- ☐ Existing Bowtie2 index

**Sequences to align:**

First sequence file to align:

Second sequence file (for paired end alignments):

**Options:**

- ☒ Local alignments ☐ End-to-end alignments
- ☒ Exclude unaligned reads

**Paired end options:**

Insert size range: from  to

- ☐ Exclude unpaired ends
- ☐ Exclude pairs with wrong insert size or orientation

The "**Align to:**" section at the top lets you select the reference sequence(s) that you want to align to. Typically, the reference sequence will be in a file in FASTA format; you can select this file after pressing the "Select..." button at the top. If you have already imported the reference sequence(s) into your CodonCode Aligner project, and select the sequence(s) before selecting "**Align with Bowtie2...**", the "Selected sequence" radio button will be enabled.

The third option for the reference sequence is to align against an existing Bowtie2 index. To use an existing index, place all the bowtie2 index files inside a folder, and place this folder into this directory: CodonCode/bowtie2/Index in your "Documents" folder.

The "**Sequences to align:**" section in the middle lets you specify one or two sequence files that should be aligned to the reference sequence. These files must be in a format that Bowtie2 can read (typically FASTQ or FASTA). Two files are used for paired-end sequences, where each file contains one of the two ends; both sequence files must have exactly the same number of sequences in the same order.

The "**Options:**" section lets you choose options for the Bowtie2 alignment. The first option is whether bowtie2 should generate local alignments, where some bases at the start and end of reads may remain unaligned; or end-to-end alignments. The next option lets you choose whether or not reads that do not align to the reference sequence should be imported.

The "Paired end options:" section applies only if you are aligning paired-end reads (separated into two files, specified in the "Sequences to align:" section). Make sure to specify the correct size range of your inserts in the given boxes, and then choose whether or not you want to exclude reads where only one of the two "mates" align, and whether to exclude pairs that have the wrong insert size or read orientation ("discordant" pairs).

You can specify additional parameters for the Bowtie2 alignment by pressing the "**More options...**" button. This will bring up the following dialog:

Advanced Bowtie2 Options

Advanced Bowtie2 options

# of cores to use: 1

☐ Keep index files

☐ Keep result (.sam) file

5' bases to trim: 0

3' bases to trim: 0

Speed option: default

Reads to skip at start: 0

# reads to align: all (default)

Match bonus: 2

Mismatch penalty: 6,2

Penalty at Ns: 1

Read gap penalty: 5,3

Reference gap penalty: 5,3

Min. score: default

Max. ambiguities: default

Max. # of pads: 15

Gap distance from ends: 4

Mate orientation: --fr

Quality type: --phred33

☐ Ignore qualities

Defaults

Cancel OK



The first option lets you choose **how many CPUs or cores** Bowtie2 should use for the alignment. If your computer has multiple CPUs or cores, selecting more cores can reduced the time needed to create the alignment significantly. The number of cores you can choose is limited to the number of cores that your computer has.

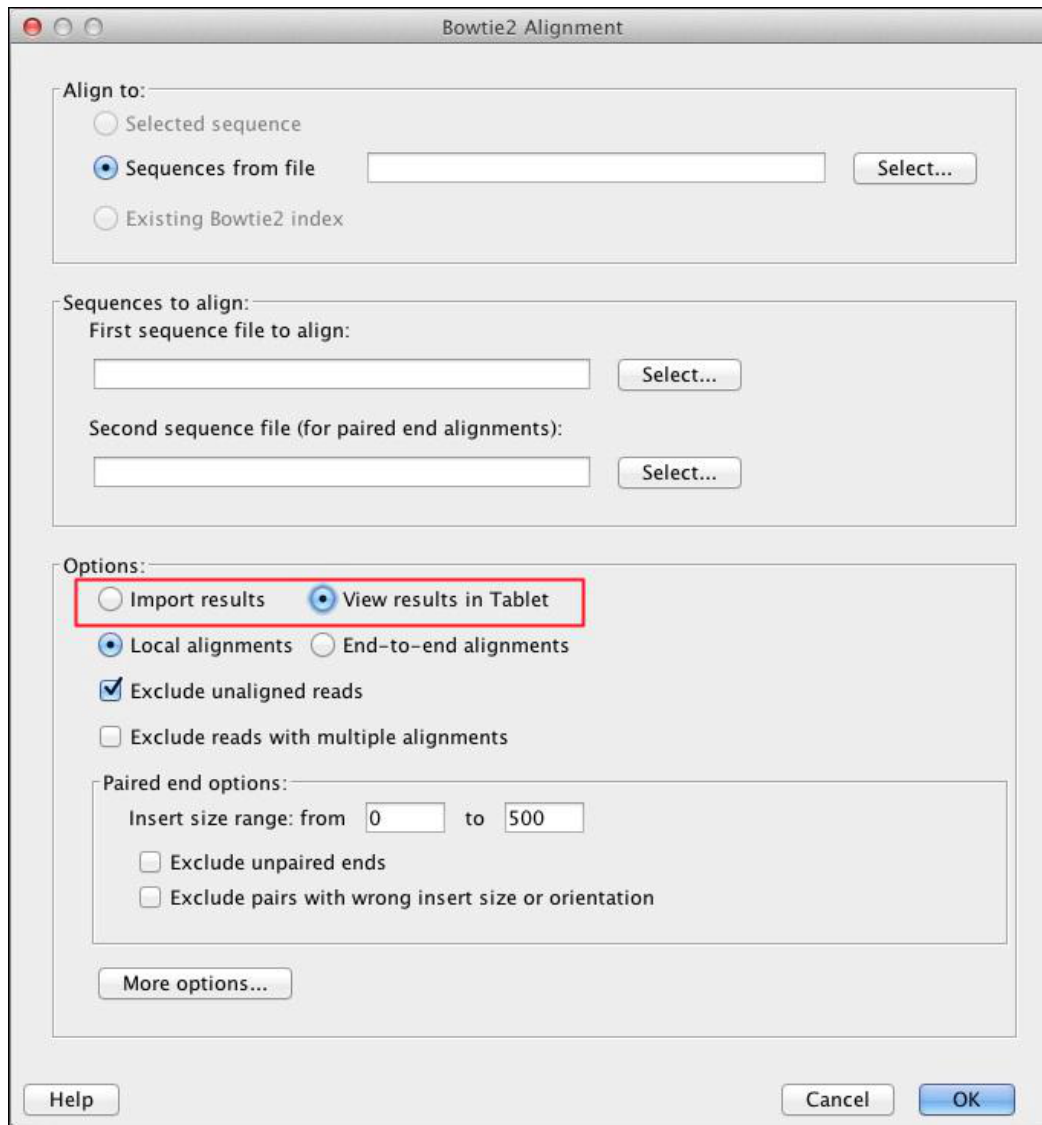
The second option allows you to **keep the index files** that Bowtie2 creates from your reference sequence. If selected, a new folder with the name of your reference sequence will be created in a folder called "Index", which in turn is inside the "bowtie2" folder in the "CodonCode" folder in your "Documents" folder. Keeping index files for future use can be especially useful when working with large reference sequences, where the index file creation may take a long time. Existing index files will be listed in the top section ("Existing Bowtie2 index") the next time you start a Bowtie2 alignment from CodonCode Aligner. If the "Keep the index file" option is not checked, any newly created index files will be deleted after the Bowtie2 alignment is complete.

The third option lets you choose to **keep the alignment result file (the .sam file)** that Bowtie2 produces. If this checkbox is checked, the file will be stored in a folder called "Results", which in turn is inside the "bowtie2" folder in the "CodonCode" folder in your "Documents" folder. If this checkbox is not selected, the file will be deleted after CodonCode Aligner imported it. However, if CodonCode Aligner cannot import the result file (for example because of insufficient memory), you will be asked if you want to keep this file.

The additional options in this dialog let you fine-tune how Bowtie2 creates the alignment. Most of these options are intended for expert use only. For additional information about these options, please check the Bowtie2 documentation in the "HelperPrograms" folder inside the folder where CodonCode Aligner is installed, or at <http://bowtie-bio.sourceforge.net/bowtie2/>. Hovering over an option with the mouse will show a tooltip text that shows the corresponding command line option.

## Viewing Bowtie2 alignments with Tablet

The program Tablet (<https://ics.hutton.ac.uk/tablet/>) is a popular viewer for NGS alignments and assemblies. CodonCode Aligner can let you view Bowtie2 alignments directly with Tablet rather than importing the assemblies, as long as you have Tablet installed at the default install location on your system (in /Applications on OS X, and "Program Files/Tablet/" on Windows). If CodonCode Aligner can find the Tablet executable at the default location, the Bowtie2 dialog will have a pair of radio buttons that let you determine how to view the alignment results:



## CodonCode Aligner User Manual

If you do not see the option to view results in Tablet, CodonCode Aligner did not find the Tablet program at the expected location.

When you choose to view results in Tablet, the alignment results generated by Bowtie2 are saved in a ".sam" format file in the folder "CodonCode/Bowtie2/Results" in you "Documents" folder.

# Regions of Interest: Features

When looking at sequences or contigs, you often may want to not look at the entire sequence, but rather move quickly to specific "regions of interest". Of course, what is interesting to you will depend on the kind of project you are doing. For example, if you are assembling a shotgun sequencing project, you may be interested in regions of low coverage or low consensus quality; but in a mutation analysis project, you may instead be interested only in potential homo- or heterozygous mutations; and when working with single sequences, you may be interested in annotation of coding regions.

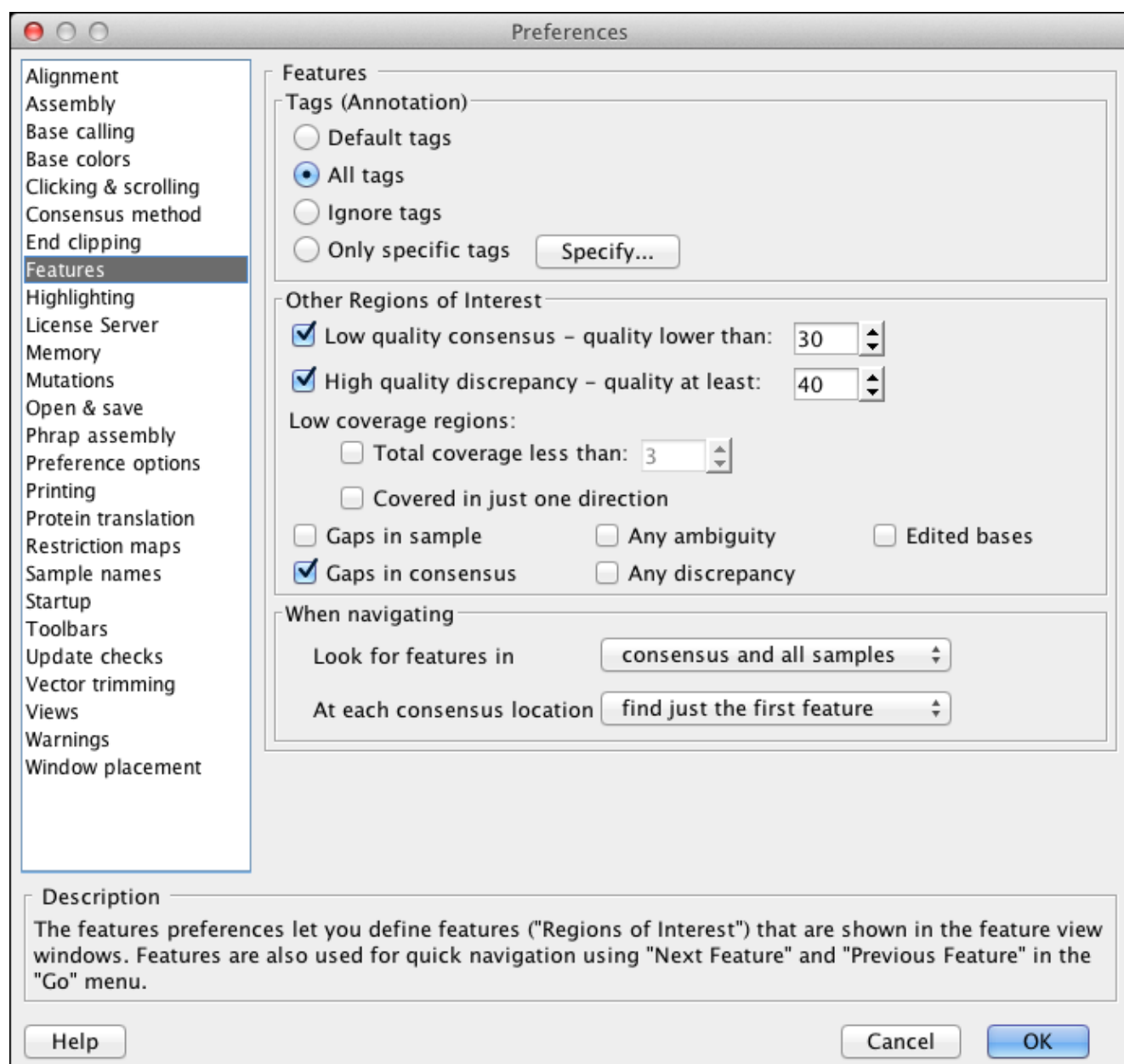
To enable you to quickly find and navigate to *your* regions of interest, CodonCode Aligner allows you to define "features". You can define a variety of different things as features, for example gaps in samples or contigs, discrepancies, low coverage regions, or tags (annotation) that have been added by programs like PolyPhred, or were imported from Genbank files. After defining features, you can:

- view all features in a separate window (the "feature view")
- quickly move to the next or previous feature in a contig
- export features to text files for later analysis

In addition, CodonCode Aligner can display annotation graphically in base view windows, and find features that are common to cloning vectors through the "Find Common Features" function.

## Defining Features

To define feature criteria, select "**Define Features...**" from the "**Go**" menu in CodonCode Aligner. This will open the preferences dialog with the "Features" panel selected:



You can change the definition of features here by using the various buttons and text fields. For a more detailed explanation, check the [feature preference help page](#).

## Moving to Features

You can quickly move from feature to feature in contig views by using the "**Next Feature**" and "**Previous Feature**" menu items in the "**Go**" menu.

To try this out, open a contig view, for example for the contig in the "Example1" project (in the "Example" folder in the CodonCode Aligner folder). Then, select the "**Go**" menu. The item we want to use is "**Next Feature**". Notice the keyboard shortcut shown in the menu - it is **Command-Right** arrow on OS X, and **Control-Right** arrow on Windows. Similar, the shortcut for "Previous Feature" is Command-Left arrow respectively Control-Left arrow.

Try out the "Next Feature" keyboard shortcut out from the contig view. You will notice that the cursor and selection in the contig view move to the right, sometimes by a few bases, sometimes by many bases. To see what kind of feature you just navigated to, look at the project view window. The status panel at the bottom shows a brief description of the feature (in our example, "Feature: Low Quality Consensus" twice in a row, then "Feature: Discrepancy").

Use the keyboard shortcuts to move forward and backward between features. You can also try changing the definition of features in the preferences.

## View, Print, Export, Import Features

To get an overview of features in one or more contigs, select the contig(s) in the project view, and then choose "**Feature View**" from the "**View**" menu. This will open a feature view window, showing a table of all features in the contig or contigs. You can double-click on any feature to take a closer look at the feature, or print the feature view; for more information, please read the [feature view help page](#).

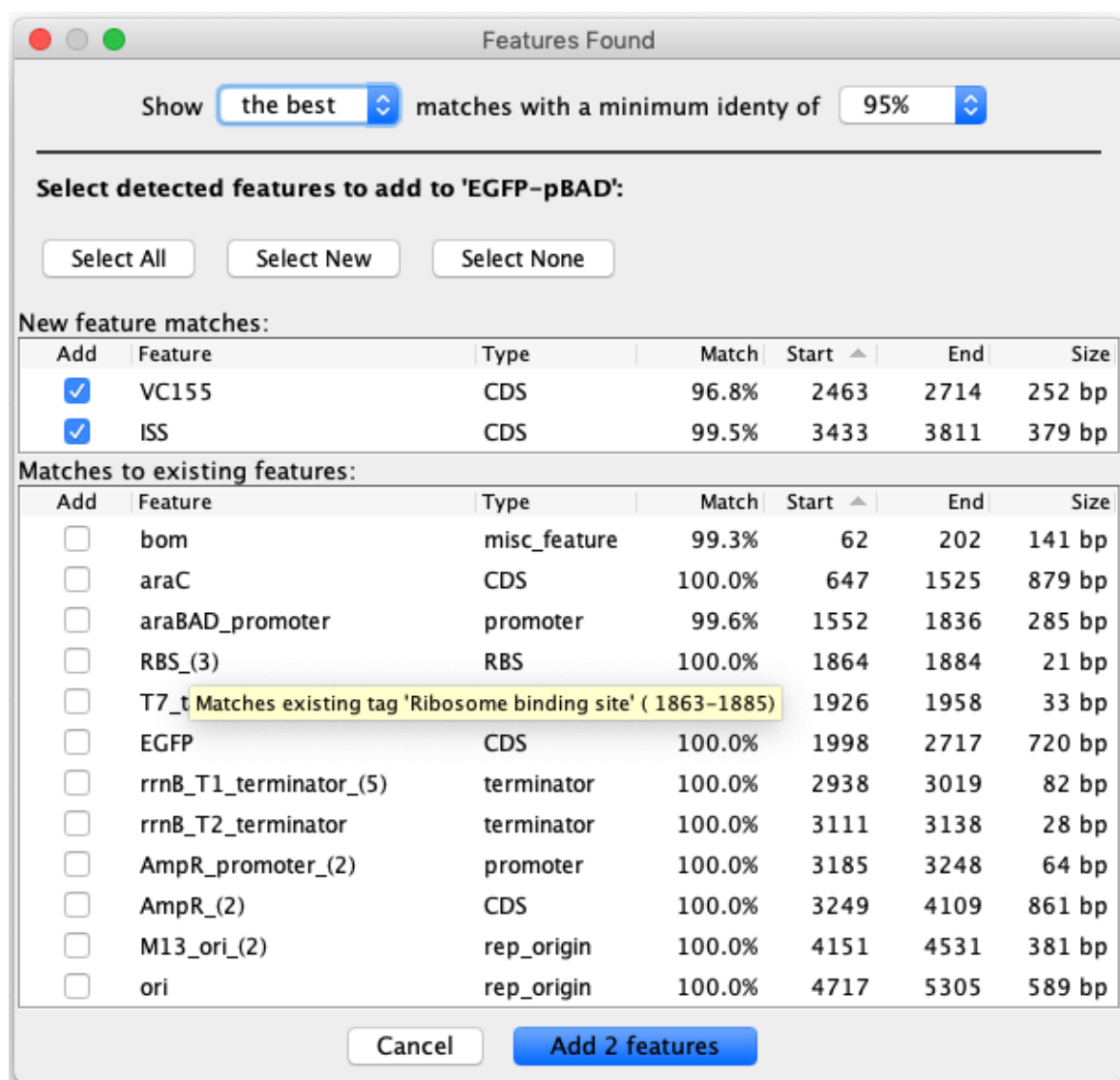
You can also export features from a selection of contigs or all contigs in a project to text files, as described on the "[Exporting features](#)" help page.

You can import annotation features as tags from files in GFF format, as explained on the "[Importing Annotation \(Tags\)](#)" help section.

## Find Common Features

To annotate DNA sequences, for example recombinant plasmids, CodonCode Aligner can identify functional elements that are commonly found in plasmids ("common features"), and add corresponding tags to the sequence.

To find common features, select the sample, and then choose "**Find Common Features...**" from the "**Sample**" menu. Aligner will compare the sequence of your sample to the sequence from known common features, and show the results in a dialog:



When matches are found they are displayed in one or two tables: new matches on top, and matches to features that are already annotated in the sequence below. Hovering over a feature that already is annotated with the mouse shows a the name and position of the existing match. To characterize a feature as "existing", the found feature must be of the same type as the existing feature, and one of the two features must contain at least 60% of the region covered by the other feature.

The common features that CodonCode Aligner searches for are defined in a file that is installed in the "Features" folder inside the "Aligner Data" folder where CodonCode Aligner is installed. The file is based on the features used by the program pLAnnotate (<https://github.com/barricklab/pLAnnotate>), and contains about 1300 entries. For many features, multiple similar entries are present in the file. By default, CodonCode Aligner will only show the match with the highest identity (the best match). The pulldown at the top allows you to see all matches found that match the given minimum identity by selecting "all" instead of "the best".

To add tags for the found common features to the sample, select the features you want to add using the checkboxes and "Select" buttons, and then press the "Add" button.

You can also include your own "custom" features in the search for common features by adding sequence files to the "Features" folder inside the "Aligner Data" folder where CodonCode Aligner is installed. Files must be in EMBL or Genbank format, and contain correctly formatted annotation entries.



# Finding Mutations

When working with sequence traces from genomic PCR sequencing, CodonCode Aligner can help you find and analyze homozygous and heterozygous mutations ("SNPs"), both point mutations and insertions and deletions (*of course, you can also look for homozygous mutations by simply defining discrepancies and gaps as features*).

## Prerequisites

To find point mutations, the following pre-requisites must be met:

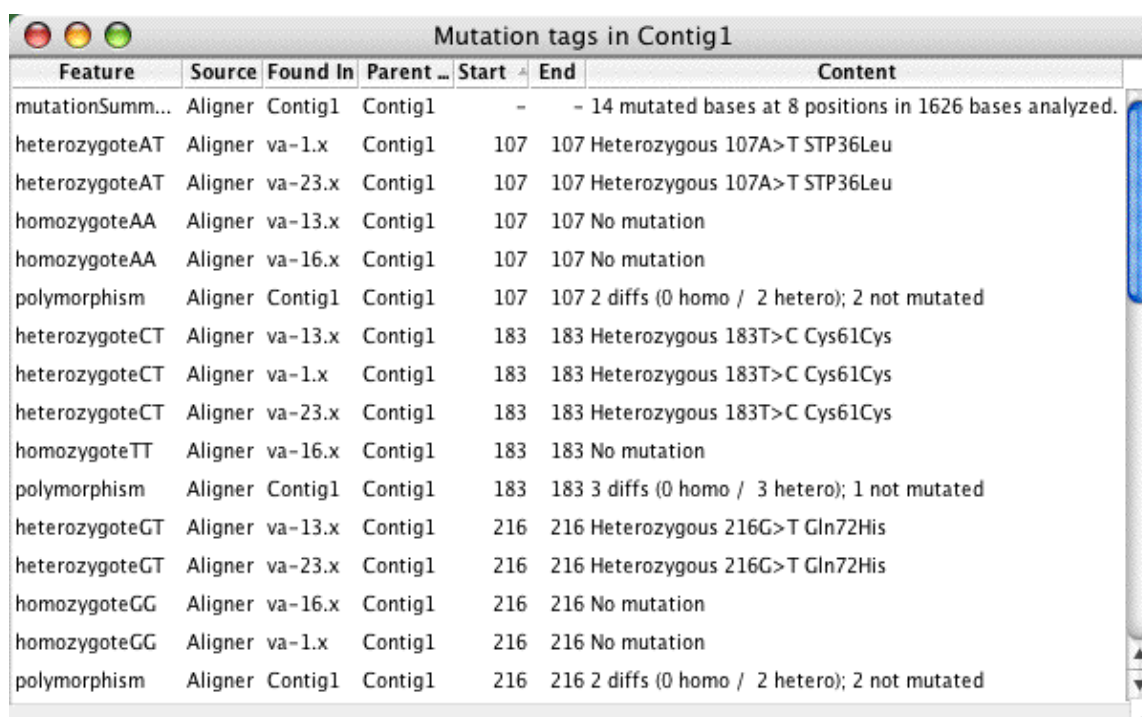
- Any **sequences** to be analyzed **that have chromatograms must have base-specific quality scores**. You can use the [base calling](#) with PHRED in Aligner to get quality scores, if needed. Sequences without traces (text sequences) do not need to have quality scores; if they have quality scores, the quality scores will be used to ignore low-quality sequence at the ends.
- The **sequences must be in a contig**. The contig can be generated with Aligner by assembling or aligning to a reference sequence, or it can be from importing an assembly. Only samples that have sequence traces as well as base-specific quality scores can be analyzed.

Detection of point mutations in Aligner is intended for re-sequencing projects only. The sequences to be analyzed should have the same sequencing chemistry and primers; mixing different chemistries, for example dye primer and dye terminator sequencing, will likely result in many false identifications.

## How To Find SNPs

- Go to the project view
- Select the contig or contigs you want to analyze
- Choose "**Find Mutations**" from the "**Contig**" menu

Finding mutations should take only a few seconds for small projects; Aligner will show a progress dialog, which allows you to cancel if things take too long. When the finding of heterozygous indels is done, Aligner will open a new window that shows the analysis results in a table like the one shown below:

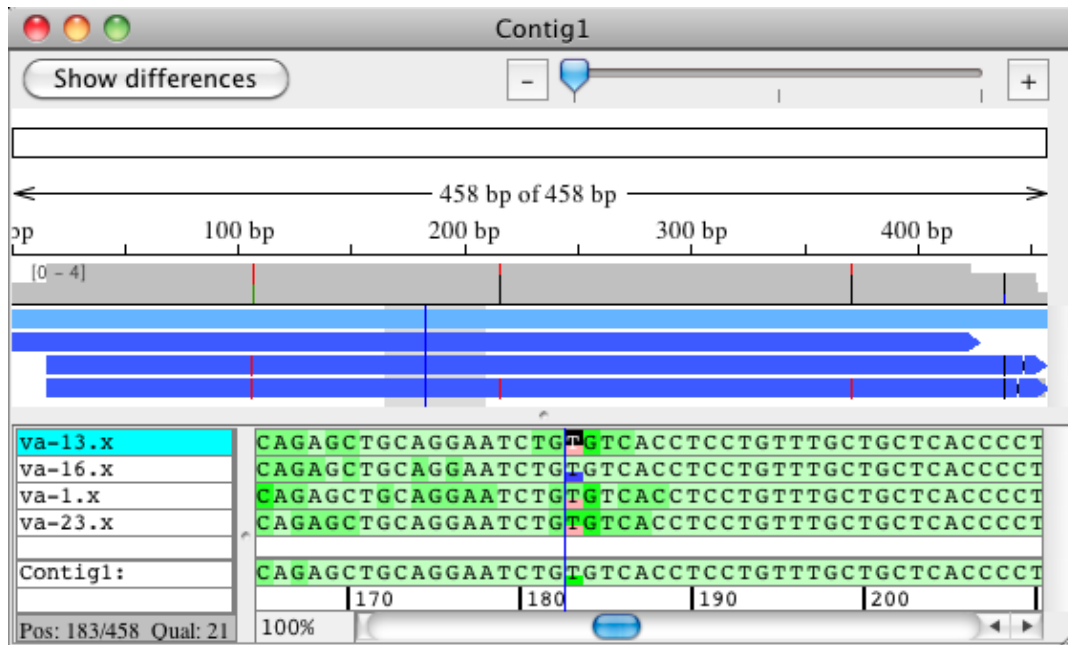


| Feature         | Source  | Found In | Parent ... | Start | End | Content   |
|-----------------|---------|----------|------------|-------|-----|---|
| mutationSumm... | Aligner | Contig1  | Contig1    | -     | -   | 14 mutated bases at 8 positions in 1626 bases analyzed. |
| heterozygoteAT  | Aligner | va-1.x   | Contig1    | 107   | 107 | Heterozygous 107A>T STP36Leu                            |
| heterozygoteAT  | Aligner | va-23.x  | Contig1    | 107   | 107 | Heterozygous 107A>T STP36Leu                            |
| homozygoteAA    | Aligner | va-13.x  | Contig1    | 107   | 107 | No mutation   |
| homozygoteAA    | Aligner | va-16.x  | Contig1    | 107   | 107 | No mutation   |
| polymorphism    | Aligner | Contig1  | Contig1    | 107   | 107 | 2 diffs (0 homo / 2 hetero); 2 not mutated              |
| heterozygoteCT  | Aligner | va-13.x  | Contig1    | 183   | 183 | Heterozygous 183T>C Cys61Cys                            |
| heterozygoteCT  | Aligner | va-1.x   | Contig1    | 183   | 183 | Heterozygous 183T>C Cys61Cys                            |
| heterozygoteCT  | Aligner | va-23.x  | Contig1    | 183   | 183 | Heterozygous 183T>C Cys61Cys                            |
| homozygoteTT    | Aligner | va-16.x  | Contig1    | 183   | 183 | No mutation   |
| polymorphism    | Aligner | Contig1  | Contig1    | 183   | 183 | 3 diffs (0 homo / 3 hetero); 1 not mutated              |
| heterozygoteGT  | Aligner | va-13.x  | Contig1    | 216   | 216 | Heterozygous 216G>T Gln72His                            |
| heterozygoteGT  | Aligner | va-23.x  | Contig1    | 216   | 216 | Heterozygous 216G>T Gln72His                            |
| homozygoteGG    | Aligner | va-16.x  | Contig1    | 216   | 216 | No mutation   |
| homozygoteGG    | Aligner | va-1.x   | Contig1    | 216   | 216 | No mutation   |
| polymorphism    | Aligner | Contig1  | Contig1    | 216   | 216 | 2 diffs (0 homo / 2 hetero); 2 not mutated              |

The table shows the characterization of samples at each consensus base where Aligner found at least one mutated base; Aligner has added tags at these points to each sample that describe Aligner's classification. For example, at consensus base 183 in the table above, Aligner classified three samples as heterozygous C-T mixes, and one sample as homozygous T-T.

Use the [mutation preferences](#) to adjust the mutation detection sensitivity, whether Aligner should add tags to samples that are not mutated, and so on. You can [use tags](#), for example a "codingSequence" tag assigned to the reference sequence or "dontGenotype" tags, to fine-tune which regions Aligner analyzes, and how the effect of mutations is described in the "Content" field; this is described in detail [below](#).

To take a close look at the results, you can double-click on any entry in the table. This will bring up the views you have chosen in the [double-click preferences](#) - typically the contig view and the trace view. A screen shot of the contig view for the example above is shown below:



The tags added by Aligner are shown as blue and pink boxes - blue boxes indicating homozygous bases, and pink boxes indicating heterozygous bases (unless you changed the display of tags in the [highlighting preferences](#) to something other than "box").

The corresponding trace view for the four samples is shown below:



Note that all three traces that were characterized as heterozygous C-T show two peaks, a blue C peak and a red T peak (your colors may be different, depending on your [base color preferences](#)). Also note that the T peak in the heterozygous samples is only about half the height of the T peak in the homozygous sample at the top. Aligner uses these two pieces of information, the secondary peak and the reduction in peak intensity, to identify heterozygous bases.

To get more information about a tag, you can right-click (OS X: control-click) on the base with the tag, and select **"Edit Tag ..."** from the popup menu (the menu text will show the type of the tag selected, for example **"Edit Tag 'heterozygoteCT'..."**).

Selecting the **"Edit Tag ..."** popup menu item will display the following tag dialog:

**Edit tag for va-1.x**

**Tag Details**

Label: heterozygoteCT

Type: heterozygoteCT

Program: Aligner

Start: 168

End: 168

Date: Sep 24, 2019 13:07:40

Color:

Notes: Heterozygous 183T>C Cys61Cys

☐ Confirmed

Cancel OK

You can quickly confirm a tag by clicking on the "Confirmed" checkbox, delete the tag, or add your comments in the "Notes" text area.

## Fixing Errors

There will be cases where you disagree with the classification made by CodonCode Aligner, and want to change it. Depending on the circumstances, this is what you can do:

- **False positives:**

If you think that a base is not mutated, you can change the tag to a "false positive" tag. The fastest way to do this is by using the popup menu from the trace view or contig view. Right-click (OS X: control-click) on the base with the wrong tag, and then choose "**Mark heterozygoteAC as False Positive**" from the popup menu. If you did click on a base with a mutation tag, this will be the second item in the popup menu. The actual text will differ a bit, depending on the original classification. False positive tags get preserved when you re-do the mutation detection.

- **Wrong classifications:**

If a base is mutated, but the classification by Aligner is incorrect, you can change the tag in a two-step process. First, bring up the tag information dialog, as described just above (right-click resp. control-click on the base, select "**Show tag...**" from the popup menu). In the tag dialog, click on the "**Change...**" button. This will bring up another dialog where you can choose the correct tag from a pulldown menu. This allows you to (for example) change a tag from "heterozygote CT" to "homozygoteCC".

When you change a tag this way, Aligner will update the notation that shows the amino acid effect in the "Notes" section of the tag (unless you have annotated a coding region, and the mutation is in a non-coding section of your sample).

- **Missed mutations:** If Aligner missed a mutation, you can manually add a tag that describes the mutation. In a trace view or contig view, right-click (OS X: control-click) on the mutated base, and then choose "**Add Tag...**" from the popup menu. This will bring up the "Add tag" dialog (an example is shown a bit further below). Select the tag type for your mutation from the pullup menu near the top - for example "heterozygoteCT" or "homozygoteAA". Aligner will show the amino-acid effect in the "Notes" section. Click "OK" when you're done.

If Aligner misses a lot of mutations in your project, or if finding *all* mutated bases is important to you, you may want to make sure that you have set the mutation detection sensitivity to "High" in the [mutation detection preferences](#).

The three actions described above ( Show tag, Add tag, and Mark as False Positive ) are also available from the "**Sample => Tag**" menu - but usually, using the popup menu as described will be faster and easier.

Please note that any mutation tags that you change or add, except for "False Positive" tags, are usually deleted when you repeat "Find Mutations" for the same contig.

## Tags for Finding Mutations

To fine-tune mutation detection, you can use tags to:

- [Define the coding region](#) in your consensus or reference sequence
- [Exclude regions](#) in samples or the contig from mutation finding

## Defining the Coding Region

You probably noticed that the tag content contains a summary of the mutation which also indicates the amino acid change ("183 T>C Cys61Cys" in the example above). Unless you give Aligner more information, this annotation will be relative to the first base in the consensus sequence. However, the coordinates used will be **ungapped** (after removing any gaps from the consensus), while other coordinates shown by Aligner are typically **gapped** (they include the count of gaps in the consensus). In the example above, the gapped and ungapped coordinates are identical, since the consensus does not contain any gaps.

The amino acid change (or lack of change) is shown for the first of the three possible forward reading frames. In your own analysis, that may not be what you want - your coding sequence may start at base 2 or 3, or even further in, for example because you included a part of the sequence before the start codon in your reference sequence.

To define where the coding region is, you can add a "codingSequence" tags to the consensus or to your reference sequence.

If you have a reference sequence, you should:

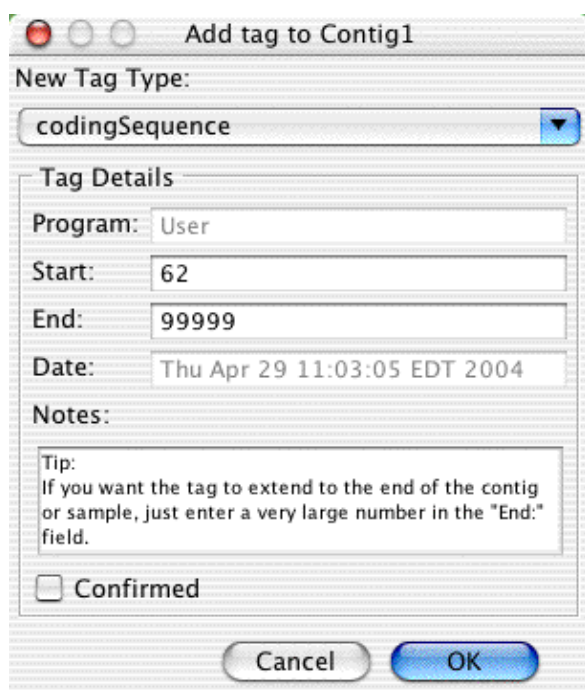
- add "codingSequence" tags to the reference sequence (not to the consensus sequence).
- create your contigs by [aligning to a reference sequence](#), not by assembling (you can use the "Contig information" menu item in the "Contig" menu to verify this if in doubt).

For contigs formed by alignments to reference sequences, the coding sequence tags will always be taken from the reference sequence, not from the coding sequence. If your original sequences were in Genbank or EMBL format that had coding sequence annotation, this annotation will be used. Otherwise, you can add coding sequence tags manually.

### To add a "codingSequence" tag:

- Open a contig view for the contig you are working with (you can also use a base view)
- Select the first base of your coding sequence in the reference sequence (or the consensus sequence)
- Right-click (OS X: control-click) to show the popup menu, and select "**Add Tag..**" (or select "**Add Tag..**" from the "**Tag**" sub-menu in the "**Sample**" menu)

This will bring up the "Add tag" dialog:



Select "codingSequence" as the tag type from the pull down menu at the top. Then enter the end coordinate of your coding sequence at the bottom (you could also select the entire coding sequence before choosing "Add Tag..."). When everything looks right, press "OK".

If the first base of your coding sequence is NOT the first base of a codon, you can also add a "codonStart" tag; the codonStart tag must be within two bases of the codingSequence start, though.

If your reference sequence was originally read from a file in Genbank format, and contained a simple "CDS" annotation, Aligner will use this to automatically create "codingSequence" and "codonStart" tags. However, any "CDS" tags in Genbank sequences that point to other sequences or join multiple sequences will be ignored.

You can add as many "codingSequence" tags as you like. You can add the "codonStart" tag only to one of the first three bases in the first codingSequence; if you try to place it anywhere else, it will be ignored.

After defining your coding region, Aligner will use this information next time you choose "Find Mutations" for this contig. There are a few things to keep in mind, though:

- The content of mutation tags is not updated when you change your reference sequence or consensus (you can simply find mutations again to get the current results)
- If both the reference sequence and the consensus have a "codingSequence" tag, the tag from the consensus will be used
- The "codingSequence" tag from the reference sequence will only be used if the contig you are analyzing is an alignment, not an assembly

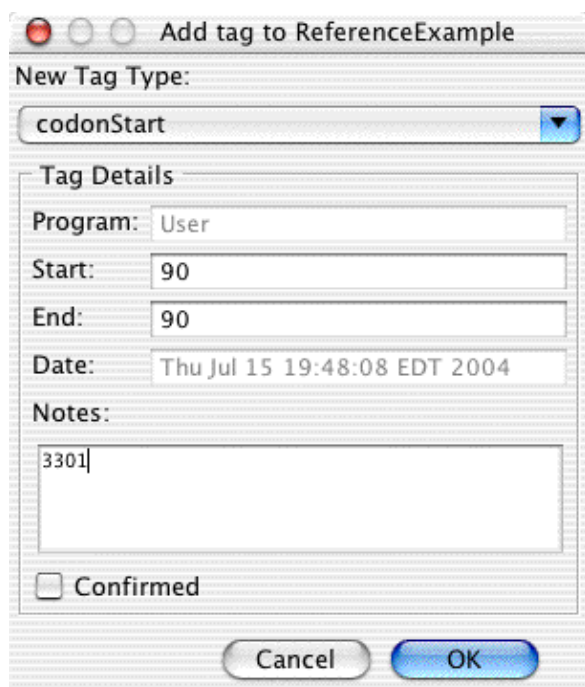
## Numbering the Coding Sequence

In mutation analysis projects, you will often want to assign a base number to the first base in your coding sequence - for example because it is not the first exon in a gene. In Aligner, you can do this by adding a "codonStart" tag, and entering the corresponding nucleotide number in the "Notes" field. The codonStart tag



identifies the first base of the first complete codon, and therefore must be assigned to any of the first three bases in the first coding region (see [Notes](#) below).

Here is an example of what the "Add Tag" dialog looks like in this case:

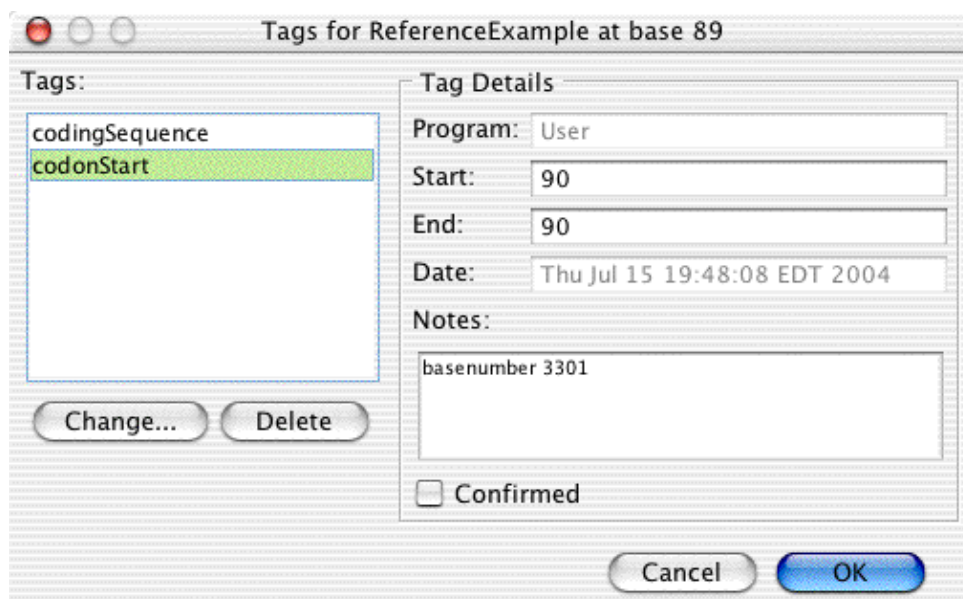


The dialog box is titled "Add tag to ReferenceExample". It contains a "New Tag Type:" dropdown menu with "codonStart" selected. Below this is a "Tag Details" section with the following fields:

- Program: User
- Start: 90
- End: 90
- Date: Thu Jul 15 19:48:08 EDT 2004
- Notes: 3301

At the bottom of the "Tag Details" section is a checkbox labeled "Confirmed" which is currently unchecked. At the very bottom of the dialog are "Cancel" and "OK" buttons.

If you prefer, you can write "basenumber 3301" (or your number, of course!), or "basenumber=3301" in the "Notes:" field. You do not have to enter this information when adding the tag, you can do it later by editing the tag in the "Show tag" dialog:



The dialog box is titled "Tags for ReferenceExample at base 89". It has a "Tags:" list on the left containing "codingSequence" and "codonStart", with "codonStart" selected. Below the list are "Change..." and "Delete" buttons. To the right is a "Tag Details" section with the following fields:

- Program: User
- Start: 90
- End: 90
- Date: Thu Jul 15 19:48:08 EDT 2004
- Notes: basenumber 3301

At the bottom of the "Tag Details" section is a checkbox labeled "Confirmed" which is currently unchecked. At the very bottom of the dialog are "Cancel" and "OK" buttons.



## CodonCode Aligner User Manual

The number you set this way for the codonStart tag is the nucleotide number at this position.

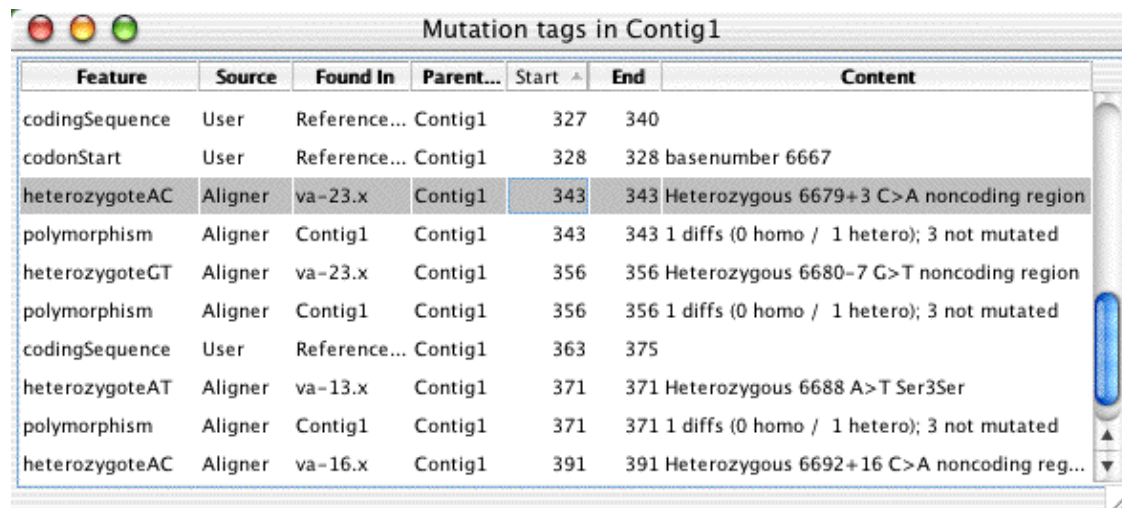
**Note:** Keep in mind that the codonStart tag identifies the start of the first **complete** codon in this exon. Therefore:

1. You **must** add "codingSequence" tags **before** adding the "codonStart" tag.
2. The codonStart tag must be in the first three bases of a "codingSequence" tag.
3. You can add **only one** codonStart tag per gene in your reference sequence.
4. If your reference sequence has several exons, the codonStart tag must be added to the first exon in the reference sequence. For example, if your reference sequence contains exons 4 and 5, add the codonStart tag to exon 4.
5. When specifying a base number in the codonStart tag, the number refers to the **nucleotide in the cDNA sequence**. Therefore, **the number must give a remainder of 1 when divided by 3** (e.g. 1, 4, 7, 10, ..., 3301, 3304,...).

Aligner will show a warning dialog when any one of these conditions is not met.

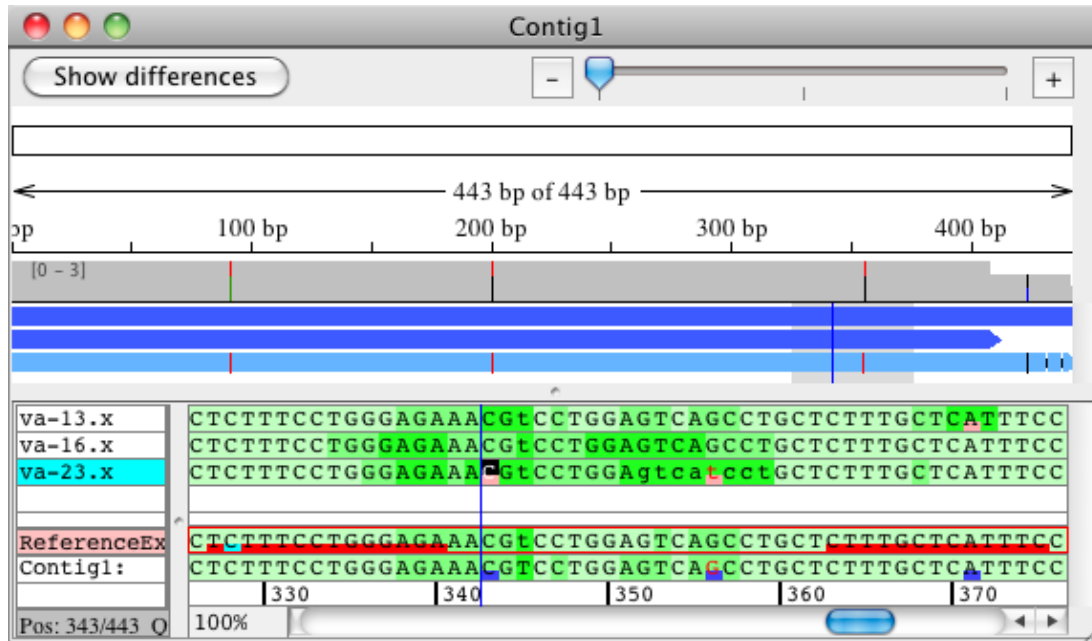
The **easiest way** to get the correct codingSequence and codonStart tags is by **importing the reference sequence from a text file in Genbank format**. CodonCode Aligner will use the "CDS" features in the Genbank file and add corresponding codingSequence and codonStart tags.

If you have more than one coding sequence region defined, Aligner will automatically keep track of the reading frame as the base numbering; mutations in introns will be annotated using the standard "n+x" or "n-y" numbering. Here is an example:



| Feature        | Source  | Found In     | Parent... | Start | End | Content                                    |
|----------------|---------|--------------|-----------|-------|-----|--|
| codingSequence | User    | Reference... | Contig1   | 327   | 340 |  |
| codonStart     | User    | Reference... | Contig1   | 328   | 328 | basenumber 6667                            |
| heterozygoteAC | Aligner | va-23.x      | Contig1   | 343   | 343 | Heterozygous 6679+3 C>A noncoding region   |
| polymorphism   | Aligner | Contig1      | Contig1   | 343   | 343 | 1 diffs (0 homo / 1 hetero); 3 not mutated |
| heterozygoteGT | Aligner | va-23.x      | Contig1   | 356   | 356 | Heterozygous 6680-7 G>T noncoding region   |
| polymorphism   | Aligner | Contig1      | Contig1   | 356   | 356 | 1 diffs (0 homo / 1 hetero); 3 not mutated |
| codingSequence | User    | Reference... | Contig1   | 363   | 375 |  |
| heterozygoteAT | Aligner | va-13.x      | Contig1   | 371   | 371 | Heterozygous 6688 A>T Ser3Ser              |
| polymorphism   | Aligner | Contig1      | Contig1   | 371   | 371 | 1 diffs (0 homo / 1 hetero); 3 not mutated |
| heterozygoteAC | Aligner | va-16.x      | Contig1   | 391   | 391 | Heterozygous 6692+16 C>A noncoding reg...  |

The corresponding contig view of this region shows the coding sequence regions (red boxes), the codonStart tag (light blue box), and the mutations found (pink boxes in the samples, dark blue boxes in the consensus):



## Excluding Regions from Analysis

You can exclude regions from being analyzed when Aligner finds mutations by adding "dontGenotype" tags. You can add "dontGenotype" tags to:

- Individual samples - to exclude tagged regions in the sample only
- The consensus sequence - to exclude the tagged region in all samples
- The reference sequence - to exclude the tagged region in all samples if (a) the analyzed contig is an alignment to the reference sequence, and (b) the reference sequence is used as the consensus sequence (as defined in the [consensus method preferences](#))

Adding tags to a region in a sample can be useful to skip regions with sequence artifacts which can throw off Aligner's analysis. Adding tags to the consensus sequence or the reference sequence in alignments can be useful to limit the analysis to a region of interest, or to avoid regions where all sample traces have artifacts.

How to add tags is described [above](#) (of course, you need to choose "dontGenotype" as the tag type to exclude regions from analysis).

## How Aligner Finds SNPs

When searching for heterozygous point mutations, CodonCode Aligner does the following:

1. Each sequence is analyzed for low-quality sequence at the start and at the end. Regions that fall below the threshold set in the [mutation detection preferences](#) are marked with a "dataNeeded" tag, and ignored in the subsequent analysis.
2. Aligner looks for heterozygous insertion/deletion tags ("heterozygoteIndel") in each sequence, and excludes regions identified as heterozygote indels. Note that the tags need to extend to the start or to the end of the sample; tags that do not extend to the start are assumed to go to the end. (*This step is*

*omitted if "Look for heterozygous indels" is deselected in the [mutation detection preferences](#), or when only looking for homozygous mutations).*

3. Aligner loops through all the consensus bases. In all samples that have aligned bases at a given position, Aligner examines the traces in each direction. Aligner looks for both secondary peaks and for drops in intensity that indicate a heterozygous base. For text samples that do not have chromatograms, Aligner just compares the base to the consensus base.
4. When a potential heterozygous base is found, Aligner examines the peaks and secondary peaks in the other samples at this position (and in the same direction) to see if the secondary peak may be due to random "noise". This "noise filter" can be turned off in the [mutation detection preferences](#).
5. Aligner classifies each base as homozygous or heterozygous, based on the height of the secondary peak and the intensity drop. Note that peaks may be classified as heterozygous even if there is no clear intensity drop (for example if all bases at this position are heterozygous).
6. If any of the samples at a given consensus position is classified as having a heterozygous base, tags are added to all analyzed samples at this position (unless "Add tags only to mutated bases" is selected in the [mutation detection preferences](#).)

The sensitivity of the detection can be adjusted in the [mutation detection preferences](#). Keep in mind, however, that Aligner may classify some bases incorrectly, and miss some heterozygous bases, regardless of the setting.

## Limitations

**CodonCode Aligner's detection of heterozygous point mutations is intended for research use only.** It has not been verified for any diagnostic or clinical applications. Aligner will miss certain mutation, and incorrectly classify others. False-positive and false-negative rates depend on the current sensitivity settings, but are never expected to be zero. All results obtained with Aligner should be checked by a qualified scientist.

The following is an incomplete list of some known limitations and suggestions:

- Identification of heterozygous point mutations requires sequencing reactions generated from PCR products from heterozygous template DNA.
- Random peaks in sequence traces may cause mis-classifications (you can add a "dontGenotype" tag in such regions, so that Aligner will ignore it the next time you choose "Find Mutations")
- Aligner relies on peak patterns that are very similar between the different samples in an analysis. Any experimental changes, including, but not limited to, the use of different sequencing primers, kits, enzymes, dyes, sequencing machines, and running conditions may result in peak pattern variations that cause increased error rates.
- The base calls need to be correct in the analyzed regions; at heterozygous bases, one of the two bases must be called correctly, or the correct [IUPAC ambiguity](#) code must be used. Homozygous bases that have ambiguity base calls are likely to be classified incorrectly.
- Vector sequences in the samples may cause analysis errors.
- Sequence quality must be high, and base-specific quality scores must be reasonably accurate (but need not be "perfect").
- Mistakes are most likely in low-quality parts of samples, and may affect other samples at this position. Mistakes at the beginning and end of samples can be reduced by increasing the minimum data quality requirement at the start and end in the [mutation detection preferences](#).
- If any samples contain heterozygous insertions or deletions, they should be identified before searching for heterozygous point mutations (for example by checking the "Look for heterozygous indels" check box in [mutation detection preferences](#)). Otherwise, many false classifications may result, especially if the minimum quality at start and end is set to low values.
- You should always adjust the [mutation detection preferences](#) to your specific needs.

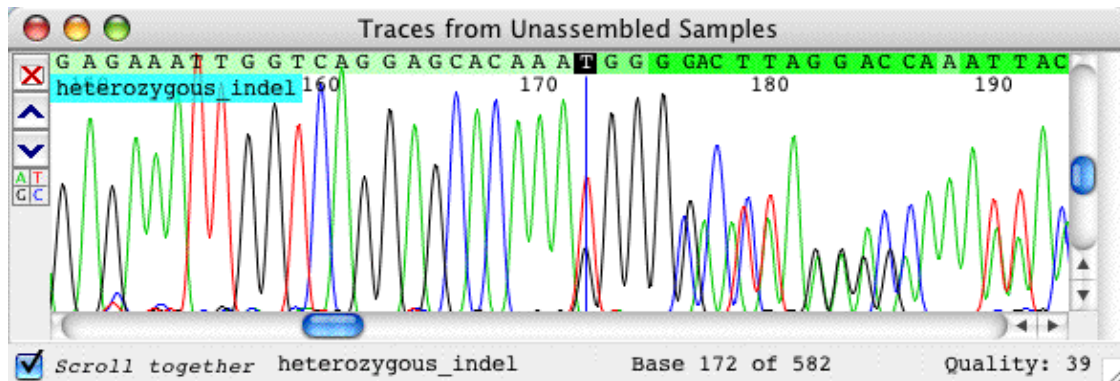
## CodonCode Aligner User Manual

- All results obtained with Aligner should be checked by a qualified scientist.

If you find any examples where Aligner's classification seems to be incorrect (and is not due to low-quality data or similar potential causes described above), please let us know by sending an email with your data attached as a Stufit or Zip archive to [support@codoncode.com](mailto:support@codoncode.com).

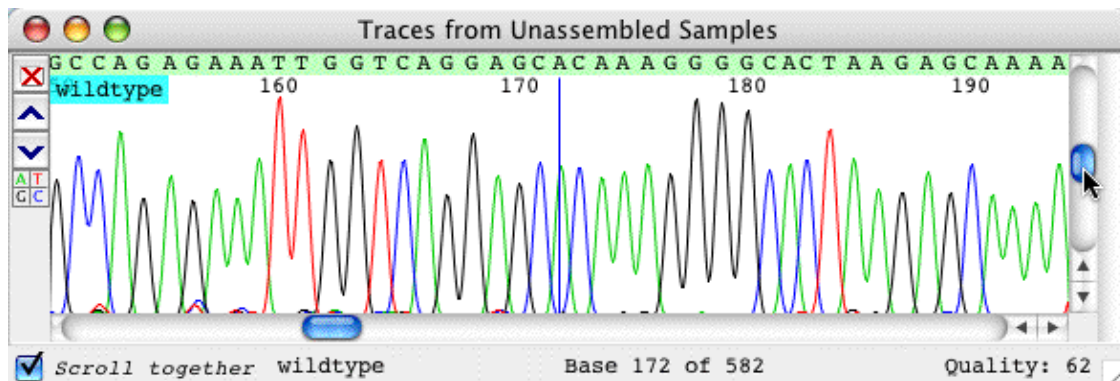
# Heterozygous Insertions and Deletions

When sequencing PCR products from genomic DNA for mutation analysis, you will sometimes encounter traces that look like this:



Up to base 174, you have a nice clean sequence; but from base 172 on, you see double peaks at many (but not all) locations. Many of the peaks are also less tall than the peaks before base 172. This is typically caused by an insertion or deletion ("indel") in one of two chromosomes that you sequenced - by a "heterozygous indel".

Let us look at a homozygous "wild type" sequence at this location:



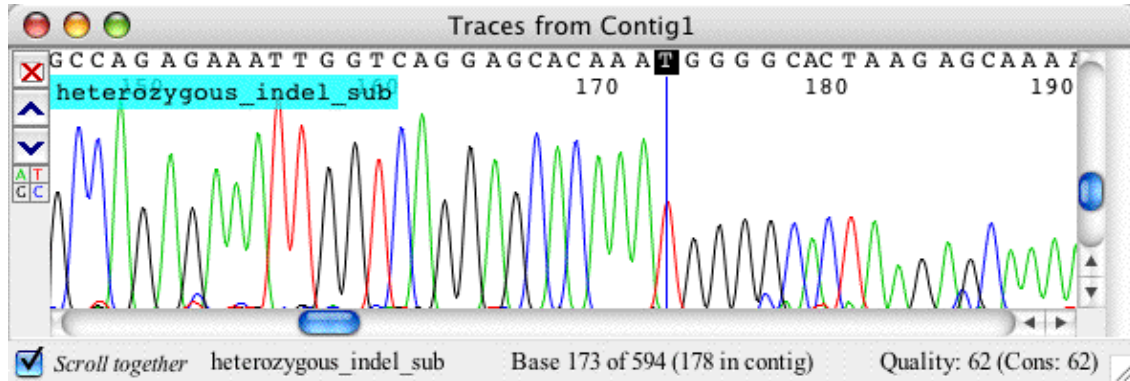
If you compare the wild type sequence to the indel sequence above, you will notice the additional red (T) peak at base 172. You also can see that the first yellow G peak is only about 50% of the intensity in the wild type trace, followed by 3 G peaks of regular height, and then an extra G peak that is half as high as the three preceding G peaks. From this, we can conclude that the indel event here is a one-base 'T' insertion on one of the two chromosomes sequenced.

Analyzing heterozygous indel traces this way is possible, but it can be tedious and error prone, especially if the insertions or deletions are longer than one or a few bases. CodonCode Aligner provides two functions that can help you in analyzing heterozygous insertions and deletions:

- The "[Find Heterozygous Indels](#)" function in the "Sample" menu can automatically identify potential heterozygous insertions and deletions.
- The "[Process Heterozygous Indels](#)" function can help you see the sequence of the mutated allele. CodonCode Aligner uses two algorithms to deduce the sequence of the heterozygous indel. The first algorithm looks at secondary bands and the reference (or consensus) sequence and replaces the

basecalls in the indel region with the likely sequence of the second allele. The second algorithm creates a new "subtracted" artificial sample by subtracting a scaled version of the wild type sequence, base calling the new "subtracted" trace with Phred. After this, both sequences are re-aligned or re-assembled with the rest of the original contig.

Here is an example of a "subtracted" sequence that was created by CodonCode Aligner from the heterozygous indel trace shown above:



In the processed sequence, where the allele that corresponds to the wild type sequence has been subtracted, it is very easy to see the one base "T" insertion.

To someone experienced in analyzing sequence traces with heterozygous indels, this example may appear trivial - after all, we could determine the mutation quickly by looking at the original trace. However, CodonCode Aligner's algorithm will work as well even for large insertions or deletions, which otherwise can be very hard to analyze. There are, of course, some [limitations](#), which are described [below](#).

## Finding Heterozygous Indels

CodonCode Aligner's "Find Heterozygous Indel" function analyses sequence chromatograms for characteristics that are typical for heterozygous insertions and deletions. In traces where Aligner finds a putative heterozygous indel, Aligner adds a tag that extends from the start of the putative indel to the end of the sequence.

To find heterozygous indels, do the following:

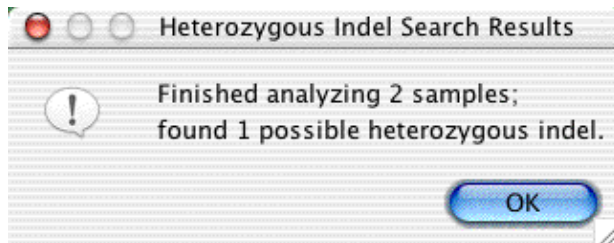
- Select the traces you want to analyze in the project view. Traces can be unassembled or in contigs, but must have base-specific quality scores (run base calling with Phred first if needed).
- Heterozygous indel detection works best if the samples have not been end clipped. End clipping will typically remove the parts of sequences that have heterozygous indels. While Aligner tries to compensate for this, the detection rate for samples that have not been end-clipped is slightly higher than that for end-clipped samples. *If your samples have been end clipped, you can simply re-do the base calling to get end-to-end sequences.*
- Go to the "Sample" menu, and select "**Find Heterozygous Indels**".

Aligner will show a progress bar while looking at your traces one-by-one. The analysis may take a few seconds per trace.

Aligner will show the results for each sequence analyzed in the status area, and add a "heterozygoteIndel" tag



to each sequence where it finds a putative heterozygous insertion/deletion. When all samples have been analyzed, Aligner will show a dialog that summarizes the results:



If any indels were found, Aligner will open a new window that shows information about these indels, similar to a feature view window:

| Feature           | Source  | Found In           | Parent Contig       | Start | End | Content   |
|-------------------|---------|--------------------|---------------------|-------|-----|-----------|
| heterozygoteIndel | Aligner | heterozygous_indel | Unassembled Samples | 172   | 582 | Score: 29 |

Double-clicking on any line in the table above will open a view for this sample so that you can verify the results (typically, the trace view will be opened, unless you have changed the [double clicking preferences](#)).

To try this out with an example project that is included with CodonCode Aligner, you can open the "Hetero\_indel" project in the "Example Files" folder in the directory where you installed CodonCode Aligner.

## Hetero Indel Scores

All heterozygous indels found by CodonCode Aligner will receive a score that is shown in the "Content" column. Scores assigned range from 11 to about 35, with higher scores indicating higher confidence. Indels with a score below 15 are more likely to be false positives; the proportion of false positives in indels with scores above 20 is substantially lower. *Indel scores are somewhat similar to Phred quality scores; however, unlike Phred quality scores, indel scores are **not accurately linked to error probabilities**.*

## False Negatives

The indel finding algorithm may sometimes miss real mutations, especially if they are very close to the start or end of a sequence, or if the sequence before the indel is of low quality. In this case, you can **add a "heterozygoteIndel" tag by hand**, as follows:

- In a trace view window, select the base where the heterozygous indel starts (for example, base 175 in the "heterozygous indel" sample in the example project)
- Bring up the popup menu by right-clicking on this base on Windows, or Control-clicking on OS X.
- Select "Add Tag..." from the popup menu. This will open a new "Add Tag" dialog.
- In the "Add Tag" dialog, select "heterozygoteIndel" as the tag type from the pull down menu at the top. You can also enter "9999" in the "End" text field if you would like the tag to cover the entire rest of the sample, instead of just one base. Then, click "OK".

You should now see the tag displayed in the trace view for this sample (unless you changed your [highlighting preferences](#) to not show tags).

## False Positives

Occasionally, the indel finding may incorrectly identify artifacts in sequencing traces as heterozygous indels. This can happen when the sequence quality suddenly drops dramatically in a sequence, for example due to "polymerase stutter" after poly-A runs. Also, please keep in mind that the heterozygous indel finding is intended only for sequences from genomic PCR.

You can easily **remove incorrect heterozygous indel tags** as follows:

- Open a trace view that shows the indel tag (the easiest way to do this is to double-click on the line describing the indel tag in the report view shown above or in a feature view)
- Right-click (OS X: control-click) on any base with the indel tag to bring up the popup menu
- Select "Show local tags" from the popup menu to open the tag dialog
- In the tag dialog, select the "heterzygousIndel" tag, then press the "Delete" button, followed by the "OK" button

## Splitting Heterozygous Indels

You can split heterozygous indels into two new "pseudoallele" samples as follows:

- Select the sample(s) with a heterozygous indel tag.
- Go to the "**Sample**" menu, and select "**Split Heterozygous Indels**".

For each sample with a heterozygous indel, two new samples will be created in the "Unassembled Samples" folder. CodonCode Aligner will try to separate the traces, starting at the indel site, into a longer and a shorter pseudo-allele. This often works reasonably well, but please keep these limitations in mind:

- Splitting will work only for heterozygous indels up to about 25 bases, and only if the indel is in a region where peaks are reasonably well separated. For other indels, the sample traces that result from the splitting will typically have missing peaks or lots of double peaks.
- The separated pseudo-alleles are (currently) only intended for indel size estimates and manual verification. Specifically, any differences in peaks between the shorter and the longer allele after the indel site may or may not be real; even real differences may be attributed to the wrong allele.

## Processing Heterozygous Indels

To analyze ("process") heterozygous indels in Aligner, the samples to be analyzed need to have a "heterozygoteIndel" tag. These tags can be added by CodonCode Aligner or manually, as described in the preceding section.

In addition, the sample to be analyzed and the corresponding "wild type" trace must be in the same contig before heterozygous indels can be processed, since CodonCode Aligner uses the alignment information when subtracting the wild type trace.

To illustrate the entire process of analyzing heterozygous indels, let us process the "Hetero\_indel" example project that comes with CodonCode Aligner:



## CodonCode Aligner User Manual

- Open the "Hetero\_indel" example project (in the "Example Files" directory inside the directory where you installed CodonCode Aligner).
- Select the sample named "heterozygous\_indel".
- Go to the "**Sample**" menu, and select "**Find Heterozygous Indels**".
- When Aligner is done with the previous step, select the "Unassembled Samples" folder in the project view.
- Go to the "**Contig**" menu, and select "**Assemble**".
- When the assembly is done, select the contig "Contig1" in the project view. Alternatively, select the two samples in Contig1 in the project view.
- Go to the "**Contig**" menu, and select "**Process Heterozygous Indels**".

You will see a progress dialog appear, which shows that CodonCode Aligner does the following steps:

- Aligner finds all samples that have a "heterozygoteIndel" tag, and then identifies "wild type" traces for each (since we have only 2 traces here, this is trivial).
- Aligner replaces the base calls in the indel region as follows:
  1. Aligner looks for secondary peaks in the indel region, and replaces the base calls with ambiguity codes for the the two bases.
  2. Aligner compares the ambiguity codes to the consensus sequence, and removes the consensus base call. This leaves the base call that (often) corresponds to the base in the mutated allele.
- Aligner creates a new artificial sequence by subtracting the wild type trace from the heterozygous indel trace, starting at the start of the "heterozygoteIndel" tag. The wild type traces are scaled vertically and horizontally as needed, with the goal to subtract the trace that is due to the unmutated allele, and leaving the mutated allele.
- Next, Aligner base calls the newly created trace or traces with Phred (which requires that you have a trial license or a full license).
- When Phred is done with base calling, Aligner moves the initial subtracted trace which still has the original base calls to the trash, and imports the subtracted trace with the new Phred base calls.
- Finally, Aligner re-aligns or re-assembles the original contig, adding the newly created subtracted traces.

Here is what the final result looks like:

## CodonCode Aligner User Manual

**Hetero\_indel**

Save Project Add Samples Add Folder Add Assembly Align to Reference Assemble Unassemble

| Name                  | Contents  | Length | Quality | Position   | Added | Mo... | C... |
|-----------------------|-----------|--------|---------|------------|-------|-------|------|
| Unassembled Samples   | 0 samples | 0      | 0       | - 4/28...  | 3/... |       |      |
| Contig1               | 3 samples | 595    | 0       | - 3/16...  | 3/... |       |      |
| wildtype              | Trace     | 592    | 510     | 0 9/26...  | 3/... |       |      |
| heterozygous_indel    | Trace     | 590    | 155     | 5 9/26...  | 3/... |       |      |
| heterozygous_indel... | Trace     | 580    | 436     | 12 3/16... | 3/... |       |      |
| Trash                 | 1 samples | 0      | 0       | - 4/28...  | 3/... |       |      |

Processing heterozygous indels completed.

**Contig1**

heterozygous\_indel  
heterozygous\_indel\_sub  
wildtype

Contig1:  
Pos: 178/595

CAGAGAAATTGGTCAGGAGCACAAA-~~GGGG~~gACctAagagcAAA  
CAGAGAAATTGGTCAGGAGCACAAA-~~GGGG~~GC-CTAAGAGCAAA  
CAGAGAAATTGGTCAGGAGCACAAA-~~GGGG~~CACTAAGAGCAAA  
CAGAGAAATTGGTCAGGAGCACAAA-~~GGGG~~CACTAAGAGCAAA

**Traces from Contig1**

AGAAATTGGTCAGGAGCACAAA-~~GGGG~~gACctAagagcAAAatAa  
heterozygous\_indel 170 180 190  
AGAAATTGGTCAGGAGCACAAA-~~GGGG~~CACTAAGAGCAAAATAAC  
wildtype 170 180 190 200  
AGAAATTGGTCAGGAGCACAAA-~~GGGG~~GC-CTAAGAGCAAAATAAC  
heterozygous\_indel\_sub 160 170 180

✓ Scroll together heterozygous\_indel Base 173 of 590 (178 in contig) Quality: 39

The newly created subtracted sample is called "heterozygous\_indel\_sub". Note that (a) its sequence after the single 'T' insertion is identical to the wild type (unlike the original sample, where the extra peaks led to extra base calls), and that (b) the quality scores of the sequence after the T insertion are higher, as indicated by the lighter background colors. Before getting too excited, however, please read the next section!

## Limitations

While we believe that CodonCode Aligner's functions for analyzing heterozygous insertions and deletions can be a valuable tool, it is important to keep a number of limitations in mind - **limitations regarding the performance** of the algorithms, and **limitations regarding the interpretation of the results**.

### Processing Pre-Requisites and Limitations

- **Finding heterozygous indels requires base-specific quality scores.** The algorithm will not work on sequences that do not have quality scores. Most testing so far has been done on sequences processed with the base calling program PHRED.
- **The analysis will work only in regions of reasonably high quality, and not close to the start or end of sequence traces.** Both the finding and the subtraction step require that sequence quality before the heterozygous indel is reasonably high; for example, multiple peaks need to be well resolved, and the indel cannot be closer than approximately 50 bases to the start or end of the sequence. In addition, indels in or after long mono- or dinucleotide repeats will generally not be found (since such repeats often result in "polymerase stutter", which looks very similar to heterozygous indels).
- **Trace subtraction requires a sequence without heterozygous indels.** If all your sequences have heterozygous indels, the subtraction does not work. Ideally, the wild type sequence should not have any heterozygous mutations; however, isolated heterozygous point mutations may be tolerable.
- **The "wild type" sequence must be the same direction and chemistry, and preferably use the same primer.** The subtraction step relies on the peak patterns being very similar in the two sequences. This is generally not the case for sequences produced with different sequencing chemistries or sequenced from different strands.
- **The wild type sequence must be largely identical to one of the two alleles in the mutated sample.** This is typically the case when analyzing sequences from human samples; however, it may not be the case when analyzing regions with a high degree of lengths polymorphisms, for example intronic sequences from different species.
- **The analysis may fail in obvious or non-obvious ways.** While the trace subtraction should always produce a new "subtracted" sequence, the resulting trace may not correctly show the presumed mutated allele. In obvious cases, the subtracted sequence will have peak patterns that are clearly incorrect, and low quality scores. In less obvious cases, the subtracted sequence may look reasonable, but individual peaks and corresponding bases may be missing. Therefore, it is essential to closely look at all three sequence traces together - the indel sequence, the wild type, and the subtracted sequence.

If you happen to have examples of where the algorithm did not work as expected, we would certainly appreciate if you could send us your trace files so that we can continue to optimize CodonCode Aligner's algorithms.

## Interpreting Results

When interpreting the results of heterozygous indel processing done with CodonCode Aligner, always keep in mind that CodonCode Aligner is only a tool that helps you to come to conclusions. You will need to use caution whenever interpreting what you see. Some specific things to keep in mind:

- The subtracted sequence you see is **not** a real allele sequence. For example, any mutations you see after the indel site could be on either allele, or on both alleles.
- The base-specific **quality scores** close to and after the indel site in the subtracted sequence should be used **for guidance only**. They do **not** have the same mathematical accuracy as quality scores for "real" sequences. However, we believe that you can use the quality scores to quickly get an idea if the subtraction worked or not.
- The algorithms for analyzing heterozygous indels have been developed and tested only on a limited number of sequences, and have not been experimentally validated.

This is by no means a complete list, so be cautious! CodonCode Aligner is intended for research use only, and has never been validated for any clinical or medical applications. You should **not** base any medical decisions on any results obtained with CodonCode Aligner.

## Acknowledgements

The sequences used for the examples shown herein are from the PolyPhred examples by Dr. Deborah Nickerson's group, available at <http://droog.mbt.washington.edu/example/csft2.tar.gz>. The names of the sequences used from this collection were changed for illustrative purposes. We thank Dr. Nickerson and all others who have made traces with heterozygous indels available to us. The use of trace subtraction to identify heterozygous (point) mutations was originally described by Staden, Rada, and Bonfield and implemented in the program TraceDiff (Bonfield,JK, Rada,C and Staden,R,. 1998, Nucl. Acids Res. 26, 3404-3409).

Early work on the algorithms for analyzing heterozygous insertions and deletions used in CodonCode Aligner was funded by the National Cancer Institute (SBIR grant 1R43CA83384-01).

# Methylation Analysis

CodonCode Aligner can be used to analyze methylation of cytosines after bisulfite modification, PCR, and capillary sequencing. In the sequencing traces, non-methylated Cs will be converted to Ts, while methylated Cs in CGs will remain as Cs. If the original template DNA was partially methylated, mixed C + T peaks will be observed at methylation sites. Often, the reverse strand is sequenced, so that Gs are converted to As at non-methylated sites, and remain as Gs at fully methylated sites.

*Please note that methylation analysis is a new feature in CodonCode Aligner that has been developed with only a limited set of test data. Performance may vary for your data. If you have data where the results seem questionable, please contact CodonCode's support team.*

## Prerequisites

### Raw ABI Data

To analyze methylation in bisulfite sequencing traces, CodonCode Aligner examines the "raw" ABI data. For other purposes, the "processed" ABI trace data will generally be used. When converting raw data to processed data, the ABI software adjusts traces to even out spacing between peaks and to adjust the peak intensities in all four lanes to similar levels. However, these adjustments can artificially increase the peak heights in lanes with overall low intensities, like the C lane in bisulfite sequencing; this, in turn, can lead to large errors when trying to calculate methylation.

By default, CodonCode Aligner (and similar programs) ignores and discards "raw" ABI data, since they significantly increase memory requirements and file size. For methylation projects, you need to change this behaviour in the "Open and Save" preferences, as described below. You may then need to re-import the original ABI sequences, or re-create the project.

### Reference Sequence Alignments

For methylation analysis, CodonCode Aligner **requires a reference sequence where all Cs have been converted to Ts**, except for potentially methylated Cs in CG-dinucleotides. Only contigs that have been created by alignment to such a reference sequence can be analyzed. CodonCode Aligner provides a function to automatically convert Cs to Gs, except in CGs, as described below.

## How to Analyze Methylation

Before analyzing methylation for the first time, you need to tell CodonCode Aligner to read and save ABI raw data:

1. Open the CodonCode Aligner preferences (in the "**Edit**" menu on Windows, and the "**CodonCode Aligner**" menu on OS X).
2. Select "**Open & save**" on the left panel.
3. Make sure the checkbox labeled "**Save ABI raw data (for methylation analysis)**" is checked.
4. Click "**OK**" to close the preferences.

To analyze methylation in a set of ABI chromatograms, follow these steps:

1. Create a new CodonCode Aligner project.
2. Import the reference sequence you want to use (typically from a text file in FASTA or similar format).
3. Select the sequence you imported, and choose **"Make Reference Sequence"** from the **"Sample"** menu.
4. Select the reference sequence in the project view (or select all bases in the base view).
5. Go to the **"Edit"** menu, click on **"Change Bases"**, and select **"Change Cs to Ts (except CGs)"**. You should see a message telling you how many bases were converted.
6. Import your ABI files. *Note that the files must be in .abi or .abi format. The commonly used SCF format does not contain raw data!*
7. Select the imported ABI sequences in the project view, go to the **"Contig"** menu, and select **"Align to Reference Sequence"**.
8. When the alignment is complete, choose the resulting contig, go to the **"Contig"** menu, and select **"Analyze Methylation"**.

CodonCode Aligner will analyze the methylation in the contig, which can take several seconds per sample. When done, a result view will be displayed:

| Methylation results in Contig1 (143) |                 |          |             |       |     |  |
|--------------------------------------|-----------------|----------|-------------|-------|-----|--|
| Feature                              | Source          | Found In | Parent C... | Start | End | Content                                    |
| methylation                          | Aligner Sample3 | Contig1  |             | 135   | 135 | 0.946, G 3421-64=3357, A 371-178=193, 128  |
| methylation                          | Aligner Sample1 | Contig1  |             | 135   | 135 | 0.124, G 114-38=76, A 634-95=539, 128      |
| methylation                          | Aligner Sample2 | Contig1  |             | 135   | 135 | 0.242, G 884-112=772, A 2621-202=2419, 128 |
| methylationRange                     | Aligner Contig1 | Contig1  |             | 135   | 135 | Methylation range: 0.124 - 0.946           |
| methylation                          | Aligner Sample3 | Contig1  |             | 144   | 144 | 0.933, G 3070-50=3020, A 401-183=218, 137  |
| methylation                          | Aligner Sample1 | Contig1  |             | 144   | 144 | 0.075, G 75-36=39, A 551-71=480, 137       |
| methylation                          | Aligner Sample2 | Contig1  |             | 144   | 144 | 0.253, G 779-108=671, A 2237-254=1983, 137 |
| methylationRange                     | Aligner Contig1 | Contig1  |             | 144   | 144 | Methylation range: 0.075 - 0.933           |

In this example, the result window shows regions that were of low quality and therefore not analyzed at the top. Below are methylation results for individual samples, and a summary for a given contig base that shows the observed methylation range. The numbers in the "Content" column are as follows:

1. The calculated methylation ratio.
2. The intensities of the raw data peaks in the analyzed region for the two analyzed lanes (C and T, or G and A). The lane is shown first, followed by the raw intensity, the estimated background at the peak location, and the net intensity (raw - background).
3. The *ungapped* contig coordinates of the analyzed base.

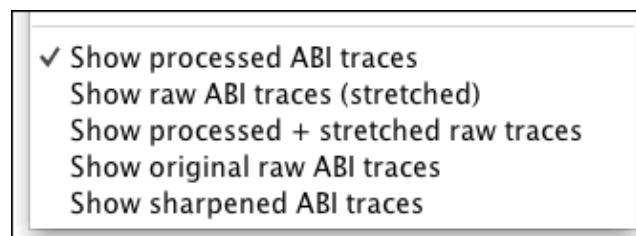
Double-clicking on a row in the table will open the related contig view and/or trace view, depending on the current [double clicking preferences](#).

## Viewing Raw Trace Data

When viewing sequence traces, CodonCode Aligner will by default show the "processed" data that are the final result of the manufacturer's image processing software. However, the processing software tries to even out intensities between lanes (among other things to make the data look pretty). This can lead to strongly enhanced peaks in the "missing" lane (C or G), which therefore cannot be used to calculate methylation. Therefore, CodonCode Aligner used the "raw" ABI data for methylation analysis, and provides options to

view the raw trace data.

When viewing traces, right-clicking in the trace view will bring up a popup menu where you can choose what to view:



For analyzing methylation, you should generally use the second option ("Show raw ABI traces (stretched)") or the third option ("Show processed + stretched raw traces"). The next section explains how CodonCode Aligner stretches raw traces.

*In the trace view, it is also possible to display the intensities of the traces by hovering with the mouse while pressing the shift key. After a short delay, CodonCode Aligner will display the intensities for the four lanes. If both processed and stretched raw traces are currently displayed, the value for both traces sets will be shown.*

## Stretching raw data

When comparing "raw" ABI data to the "processed" ABI chromatograms, one problem is that the base positions in the processed traces can be very different from the raw data. Often, the raw data have many additional data points that have been clipped off in the processed data; furthermore, peak spacing in the raw data can be very different from the processed data, since the ABI processing software tries to even out the spacing between peaks.

To enable the comparison between raw and processed ABI data, CodonCode Aligner tries to clip and stretch the raw data so that the base positions roughly match the base positions in the processed data. There are some important differences, however:

- Data are only stretched horizontally, not vertically (intensities are *not* changed)
- Shift between the lanes is only partially corrected
- Stretching is estimated, and will not be exact. This is especially true near the start of a trace; therefore, bases near the start of the trace will be excluded from methylation analysis.

For some sequence traces, the raw data stretching algorithm fails, and such traces cannot be used for methylation analysis. Such failures may be caused by a number of factors, which include poor quality; low raw data intensity; high background levels; and unusual processing by the ABI sequence analysis software. In general, traces created with old ABI sequencing chemistries (e.g. POP6 or dye primer data) or with non-ABI sequencing machines cannot be analyzed.

*If you have data where the stretching did not work properly, but none of the above factors applies, please contact CodonCode's support team.*

## Methylation Analysis Algorithm

## CodonCode Aligner User Manual

To calculate methylation, CodonCode Aligner examines peaks near the base positions where the reference sequence contains Cs (or Gs if the reference sequence was reverse-complemented). *To limit the analysis to only CG-sites, it is important to convert other Cs in the reference sequence to Ts, as described in the step-by-step instructions above.* Please note that any errors in the base calling or the alignment may lead to errors in the estimated methylation ratios. To minimize errors from low data quality, CodonCode Aligner will try to identify poor quality regions at the start and end of sequences; such regions will be marked with a "low quality" tag, and ignored in methylation analysis.

The first step in methylation analysis is to clip and horizontally stretch the raw traces so that peak positions roughly match the positions in the processed traces. This step is computing-intensive and may require several seconds per sample.

Next, CodonCode Aligner looks for the maximum peaks intensities in the C and T lanes (or G and A lanes when analyzing the reverse complement). Background levels near the peak positions are estimated, and subtracted from the peak intensities.

Finally, CodonCode Aligner calculates methylation as:  $M = C / (C+T)$

Since most sequence traces contain some degree of uncorrected background noise from non-specific peaks, the calculated methylation is often slightly higher than 0.0 even at fully unmethylated sites, and below 1.0 at fully methylated site. In general, the estimated methylation levels should be assumed to be inaccurate by 5 to 10% (e.g. a level of 0.5 may well be 0.45 or 0.55, or even 0.4 or 0.6). However, the calculated methylation can be useful to identify changes in methylation patterns between different samples. If absolute methylation results are desired, reference samples of fully methylated and fully unmethylated DNA should be included, and used for post-analysis correction of the calculated methylation values.



# Primer Design

CodonCode Aligner allows you to design primers for PCR, cloning and sequencing. Based on your selected sequence and the chosen primer design options, primers are picked using [Primer3](#). Results are then displayed and you can choose which primers you would like to import into CodonCode Aligner.

## How to Pick Primers

To design primers for one of your sequences, follow these steps:

1. Go to the project view.
2. Select the sample or contig that you want to design primers for.
3. Choose "**Design Primers**" from the "**Tools**" menu.
4. The "Primer Design" dialog is displayed where you can adjust primer picking parameters. Click "**Pick Primers**" to start finding primers for the parameters you selected.
5. If primers are found they are displayed in the "Primer Result" dialog. Select the primers that you want to import as new primer sequences in your project, then click the "**Create Primers**" button.
6. The selected primers can be found in the folder called "**Primer**" in your project.

When finding primers, CodonCode Aligner uses Primer3. Primer picking parameters can be set in the "Primer Design" dialog (see section "[Primer Design Parameters](#)"). You can design PCR, sequencing and cloning primers. Only primers that you select in the displayed primer results and choose to create, are imported in your Aligner project. Information about imported primers (like Tm, GC%, etc.) can be seen for each primer in the sample information dialog (for more details see section "[Primer Results](#)"). The template sequence that was used to pick primers for, will have feature tags that show primer annotations. Primer information can be exported for ordering primers (see "[Exporting Primers](#)").

## Primer Design Parameters

When designing primers in CodonCode Aligner, the first dialog you see lets you choose your primer design parameters:

**Primer Design**

Design primers for: ☒ PCR ☐ Sequencing

☒ Forward Primer ☒ Reverse Primer

Target region: 51 to 1610

Primer length: 18 to 22

T<sub>m</sub> (°C): 57 to 62

GC%: 20 to 80

**PCR Primer Details**

☒ Choose optimal primer location within 50 bp of target

☐ Primers end exactly at region to amplify

☐ Add Hybridization Probe

Number of primer pairs to show: 5

**Conditions**

**Advanced Settings**

Based on your selection if you want to design primers for "**PCR**" or for "**Sequencing**", the details for this dialog will change. Cloning primers can be created by selecting "PCR" and choosing "Primers end exactly at the region to amplify" in the PCR primer details.

When the dialog is first shown, the **target region** is either set to the whole sample (minus the range to pick primers in), or to the selected bases of the sequence template.

The maximum **primer length** is 35 bases. The optimal primer length and optimal T<sub>m</sub> are always the median value of minimum and maximum.

All values except for the target region are remembered. Original values can be restored at any time by pressing the "**Default**" button at the bottom of the dialog.

## PCR Parameters

When designing PCR primers you have the following options as can be seen in the image above:

- Choose primers within a certain range of your target region, or pick primers exactly at the ends of the target region.
- Add a hybridization probe. You can set the parameters for the probes by clicking on the "Set Probe Characteristics" button.
- Choose the number of primer pairs to show.

If you choose to pick **primers in a certain range**, please note that primers are picked *outside* of your specified target region. Therefore you need to make sure that the target region is set so that there is enough space for primers to be picked (e.g. setting the target region to 10-90 for a 100 bp template will only work for primers with a length of 9 bp or smaller).

If you select the radio button "**Primers end exactly at region to amplify**", primers will be fixed at the ends *inside* the target region. You want to use this setting to design *cloning primers*.

## Sequencing Parameters

For designing sequencing primers, the sequencing specific parameters look like this:

| Sequencing Primer Details                    |   |    |
|--|---|----|
| Distance between primers on same strand      | <input style="width: 50px;" type="text" value="500"/> | bp |
| Distance between forward and reverse primers | <input style="width: 50px;" type="text" value="250"/> | bp |
| Choose optimal primer location within        | <input style="width: 50px;" type="text" value="20"/>  | bp |
| Distance between first primer and target     | <input style="width: 50px;" type="text" value="50"/>  | bp |

The distance between primers on the same strand ("Spacing"), is the space from the 3'end of the primer to the 3'end of the next primer on the same strand.

The distance between forward and reverse primer is also called "Interval" and means the space from the 3'end of the primer to the 3'end of the next primer on the reverse strand.

Named "Accuracy" in Primer3, the optimal location in a certain range is the space from the calculated position of the 3'end to both sides in which Primer3 picks the best primer.

The distance between the first primer and the target ("Lead") defines the space from the 3'end of the primer to the point where the trace signals are readable.

## Conditions & Advanced Settings Parameters

The conditions section allows you to set a mispriming library, the reaction concentrations, and the formulas used to calculate the melting temperature and the salt correction:

**Conditions**

Mispriming / Repeat Library:

**Concentrations**

Monovalent salt:  mM    Divalent salt:  mM

Annealing primer:  nM    dNTPs:  mM

Melting temperature formula:

Salt correction formula:

**Mispriming / repeat libraries** are files that contain sequences that the primers should not bind to. These files are located in the folder "CodonCode Aligner/HelperPrograms/Primer3/Libraries". If you add libraries, please note that the library file must be in FASTA format, where each sequence entry must begin with an "id line" that starts with '>'. The contents of the id line is "slightly restricted" in that it should not contain any asterisk (\*).

The **formulas for melting temperature and salt correction** can also be set in the "Conditions" section. The recommended formulas by Primer3 for  $t_m$  and salt correction is "[SantaLucia 1998](#)". "Breslauer et al. 1986" as formula for the melting temperature, and "Schildkraut and Lifson 1965" as formula for the salt correction, are formulas that were used in older versions of Primer3 and allow backward compatibility. The salt correction formula can also be calculated using the formula "Owczarzy et. 2004" as described in the paper [Owczarzy, R., Moreira, B.G., You, Y., Behlke, M.A., and Walder, J.A. (2008)].

The **concentration** settings are used to calculate the oligo and primer melting temperature during the reaction.

▼ **Advanced Settings**

**Primers**   **Pairs**

|                         |                                 |              |                                |
|-------------------------|---------------------------------|--------------|--------------------------------|
| Max 3' End Stability:   | <input type="text" value="9"/>  | Max PolyX:   | <input type="text" value="5"/> |
| Max Self Compl. Any Tm: | <input type="text" value="47"/> | GC Clamp:    | <input type="text" value="0"/> |
| Max Self Compl. End Tm: | <input type="text" value="47"/> | Max N Bases: | <input type="text" value="0"/> |
| Max Hairpin Tm:         | <input type="text" value="47"/> | Max GC End:  | <input type="text" value="5"/> |
| Max Library Mispriming: | <input type="text" value="12"/> |              |                                |

The advanced settings section contains primer and primer pair characteristics. Most are self explanatory. "**Max PolyX**" is the maximum allowable length of a mononucleotide repeat (e.g. "AAAAA"), "**GC Clamp**" is the required number of consecutive Gs and Cs at the 3' end of a primer, "**Max GC End**" is the maximum number of Gs or Cs allowed in the last five 3' bases of a primer. "**Max 3' End Stability**" is the maximum stability for the last five 3' bases of a primer -bigger numbers mean more stable 3' ends.

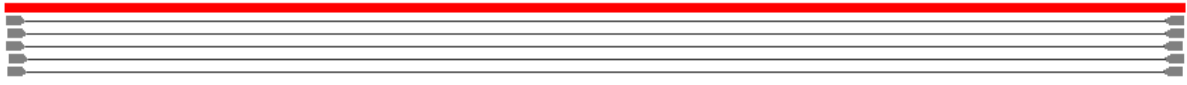
Selecting the tab "Pairs" at the top of this section will show primer characteristics for primer pairs.

## Primer Results

After clicking the "Pick Primers" button in the primer design dialog, CodonCode Aligner will run Primer3 and then show you the results:

# CodonCode Aligner User Manual

Primer Results for Contig1



Select the primers to import:

☒ Primer Pair 1

| Name     | Sequence             | Start | Length | Tm    | GC%   | Self | Any  | Hairpin | 3'Stability |
|----------|----------------------|-------|--------|-------|-------|------|------|---------|-------------|
| Primer1F | GATCGTCTCTCCTCCCCTCA | 4     | 20     | 59.82 | 60.00 | 0.00 | 0.00 | 0.00    | 3.86        |
| Primer1R | GGGGATCGTATGCCATTCT  | 1660  | 20     | 57.43 | 50.00 | 0.00 | 7.48 | 43.48   | 2.52        |

Product size: 1657 Pair Any: 0.0 Pair End: 0.0

☒ Primer Pair 2

| Name     | Sequence             | Start | Length | Tm    | GC%   | Self | Any  | Hairpin | 3'Stability |
|----------|----------------------|-------|--------|-------|-------|------|------|---------|-------------|
| Primer2F | ATCGTCTCTCCTCCCCTCAC | 5     | 20     | 60.11 | 60.00 | 0.00 | 0.00 | 0.00    | 3.51        |
| Primer2R | GGGGATCGTATGCCATTCT  | 1660  | 20     | 57.43 | 50.00 | 0.00 | 7.48 | 43.48   | 2.52        |

Product size: 1656 Pair Any: 0.0 Pair End: 0.0

☒ Primer Pair 3

| Name     | Sequence             | Start | Length | Tm    | GC%   | Self | Any  | Hairpin | 3'Stability |
|----------|----------------------|-------|--------|-------|-------|------|------|---------|-------------|
| Primer3F | GATCGTCTCTCCTCCCCTCA | 4     | 20     | 59.82 | 60.00 | 0.00 | 0.00 | 0.00    | 3.86        |
| Primer3R | GGGGATCGTATGCCATTCTT | 1660  | 21     | 58.14 | 47.62 | 0.00 | 7.48 | 43.48   | 2.52        |

Product size: 1657 Pair Any: 0.0 Pair End: 0.0

☐ Primer Pair 4

| Name     | Sequence             | Start | Length | Tm    | GC%   | Self | Any  | Hairpin | 3'Stability |
|----------|----------------------|-------|--------|-------|-------|------|------|---------|-------------|
| Primer4F | GTCTCTCCTCCCCTCACCTC | 8     | 20     | 60.40 | 65.00 | 0.00 | 0.00 | 0.00    | 3.85        |
| Primer4R | GGGGATCGTATGCCATTCT  | 1660  | 20     | 57.43 | 50.00 | 0.00 | 7.48 | 43.48   | 2.52        |

Product size: 1653 Pair Any: 0.0 Pair End: 0.0

☐ Primer Pair 5

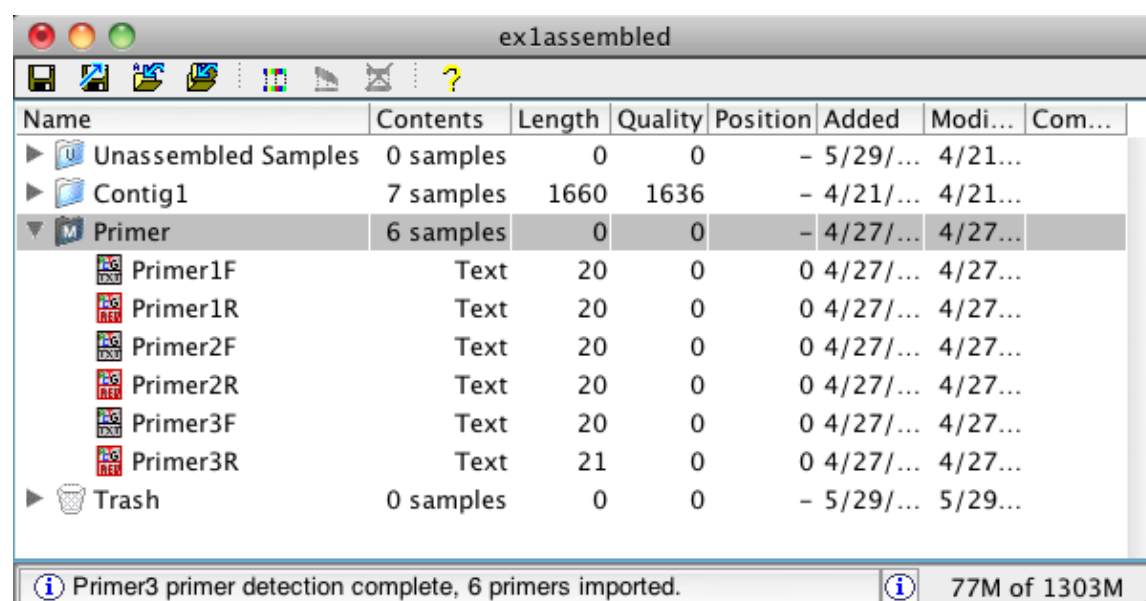
| Name     | Sequence             | Start | Length | Tm    | GC%   | Self | Any  | Hairpin | 3'Stability |
|----------|----------------------|-------|--------|-------|-------|------|------|---------|-------------|
| Primer5F | ATCGTCTCTCCTCCCCTCAC | 5     | 20     | 60.11 | 60.00 | 0.00 | 0.00 | 0.00    | 3.51        |

To import the primers into your Aligner project, **select** the primers you want to import and click the **"Create Primers"** button.

The results for primer pairs as shown above include information about each pair and some statistics at the bottom which tells you how many primers were considered and why primers might have been rejected. Primer results for sequencing primers look a little different since there are no primer pairs: all forward and all reverse primers are listed together. An overview showing where primers are located on your template sample is also displayed.

When no primers are found, Aligner displays a dialog that should show you why no primers were found. The most likely cause is that your settings were too strict in which case looking at the statistics in the dialog might give you a good idea which parameters you can change to get results.

If you have primers and choose to import them, new primers are put into a **folder called "Primer"** in your Aligner project. The folder is created if it does not already exist:



**Primer characteristics** for each primer are displayed in the sample information dialog (select the primer and choose **"Sample Information..."** from the **"Sample"** menu):

Sample Information: Primer1F

Name:

20 bp, 60.0% GC (2 A, 10 C, 2 G, 6 T)

Comments:

Primer Information:

Primer template: Contig1  
 Penalty: 0.316029  
 Interval: 4 -> 23  
 Length: 20  
 Direction: forward  
 TM: 59.816  
 GC%: 60.0  
 ANY: 0.0

☒ Allow manual edits

The sequence that the primers were created for will have a **primer annotation** tag added to its sequence for each imported primer:

Contig1

1 GGGGATCGTC TCTCCTCCCC TCACCTCCCC GCCCC TTTCT ACTGCAGCCA

51 TTAGGGGAGC CCTGTTCTG TGC

101 AAAGTCCATT TCCTCTGCC TGC

151 AGCTCCTACC TGTGCTTCAG ATC

201 AGCACACCCT CACCTCCCTG CTC

251 ACAGAATCAT ATGTCTGTTG GTA

301 GTACTTATCT GATTATTTT GTG

Contig1 Base 23 of 37

Context Menu:

- Show Tag: primerAnnotation ...
- Add Tag
- Display tag information
- View Contig
- View Bases
- View Traces
- View Qualities
- View Features
- View Restriction Map



Details about each primer can be viewed by showing the primer annotation tag (right click on the tag and select "**Show Tag: primerAnnotation ...**").

## Exporting Primers

For ordering primers you can export them into a simple csv (comma separated value) format. The exported file contains one line for each sample which consists of the primer name and the sequence.

You can export primers by choosing "**Export -> Samples...**" from the "**File**" menu. In the upcoming export dialog choose "**Primer - CSV**" as the format.

Note: All sequences in the folder called "Primer" will be treated as primers and will be exported.

## Additional Information

CodonCode Aligner runs Primer3 to pick primers.

Primer3: Copyright (c) 1996,1997,1998,1999,2000,2001,2004,2006,2007,2008,2009,2010 2011,2012,2013 Whitehead Institute for Biomedical Research, Steve Rozen (<http://purl.com/STEVEROZEN/>), Andreas Untergasser, Mado Remm, Triinu Koressaar and Helen Skaletsky. All rights reserved.

If you use primer design in CodonCode Aligner, please cite the use of Primer3 in publications as:

Untergasser A, Cutcutache I, Koressaar T, Ye J, Faircloth BC, Remm M and Rozen SG.  
Primer3--new capabilities and interfaces.  
Nucleic Acids Res. 2012 Aug 1;40(15):e115.

The paper is available at <http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3424584/>

Source code available at <http://sourceforge.net/projects/primer3/>.

Additional information and documentation about Primer3 and many of the parameters available for primer design, can be found at: [http://primer3.ut.ee/primer3web\\_help.htm](http://primer3.ut.ee/primer3web_help.htm). Please note that some of the parameters available in CodonCode Aligner may differ from the parameters found in the Primer3 online help.

# RFLP Analysis

In CodonCode Aligner you can simulate a restriction fragment length polymorphism (RFLP) analysis for your samples. You can use the RFLP tool to identify the most suitable restriction enzyme for your analysis, or simply to verify and print the expected gel. Aligner's RFLP analysis makes it possible to automatically identify the enzymes that produce different cut patterns from a large number of enzymes, and even to pick the enzyme that produces the most different cut pattern between your sequences. Results can be displayed in a virtual gel, as aligned samples with cut positions, or as table with fragment sizes. All results can be printed.

## Prerequisites

To run a RFLP analysis, the following conditions for the sequences you want to compare must be met:

- The sequences have to be in a contig. You can make a contig by creating either an [assembly or an alignment](#) for your sequences.
- All sequences in the contig need to overlap.

## How to Analyze RFLPs

To start the analysis, go to the project view and select the contig you want to analyze. Then choose "**RFLP Analysis...**" from the "**Tools**" menu.

## Samples with Different Lengths

If the samples in your contig do have different length, the samples will need to be adjusted to the same length for the analysis. The RFLP analysis tool gives you the option to shorten or extend your samples. If you choose to shorten, each sample will be shortened to the common overlapping region that all samples share. If you choose to extend the samples, each sample will be extended to the largest sample using consensus bases. The sample lengths are only changed temporarily for the analysis, and you will **not** see any changes to the samples in your project.

## Analysis Options

If all your samples have the same length or were adjusted in the previous step, you can now choose several options in the following dialog:

**RFLP Analysis**

Select enzymes to consider

☒ Use saved list: Popular Enzymes

☐ Use 5 selected enzymes: BamHI, EcoRI, MboI, ...

Specify enzymes...

Display results as

☒ Virtual gel ☐ Aligned samples ☐ Text

Marker: PCR Marker

Select enzymes for display

☐ Best enzyme only

☒ 3 best enzymes

☐ All with different cut patterns

☐ All enzymes

Cancel RFLP Analysis

The top section allows you to **select the enzymes that should be considered** for the analysis. You can either use a pre-saved enzyme list, or select enzymes explicitly. To use specific enzymes, select the **"Use selected enzymes"** radio button and click on the **"Specify enzymes..."** button, which will open a dialog that allows you to choose your enzymes.

In the **"Display results as"** options, you can choose how your results should be displayed. Options are a **virtual gel**, as an **aligned samples** map displaying cut sites, or as **text** in table form displaying the resulting fragment lengths.

The bottom section called **"Select enzymes for display"** lets you select which enzymes should be used when displaying your results. This option allows you for example to choose a large number of enzymes in the top section (e.g. all enzymes you have available in your lab), but display only results for those enzymes that produce the best (i.e. most different) cut pattern for your samples. The options on **which enzymes to display** are:

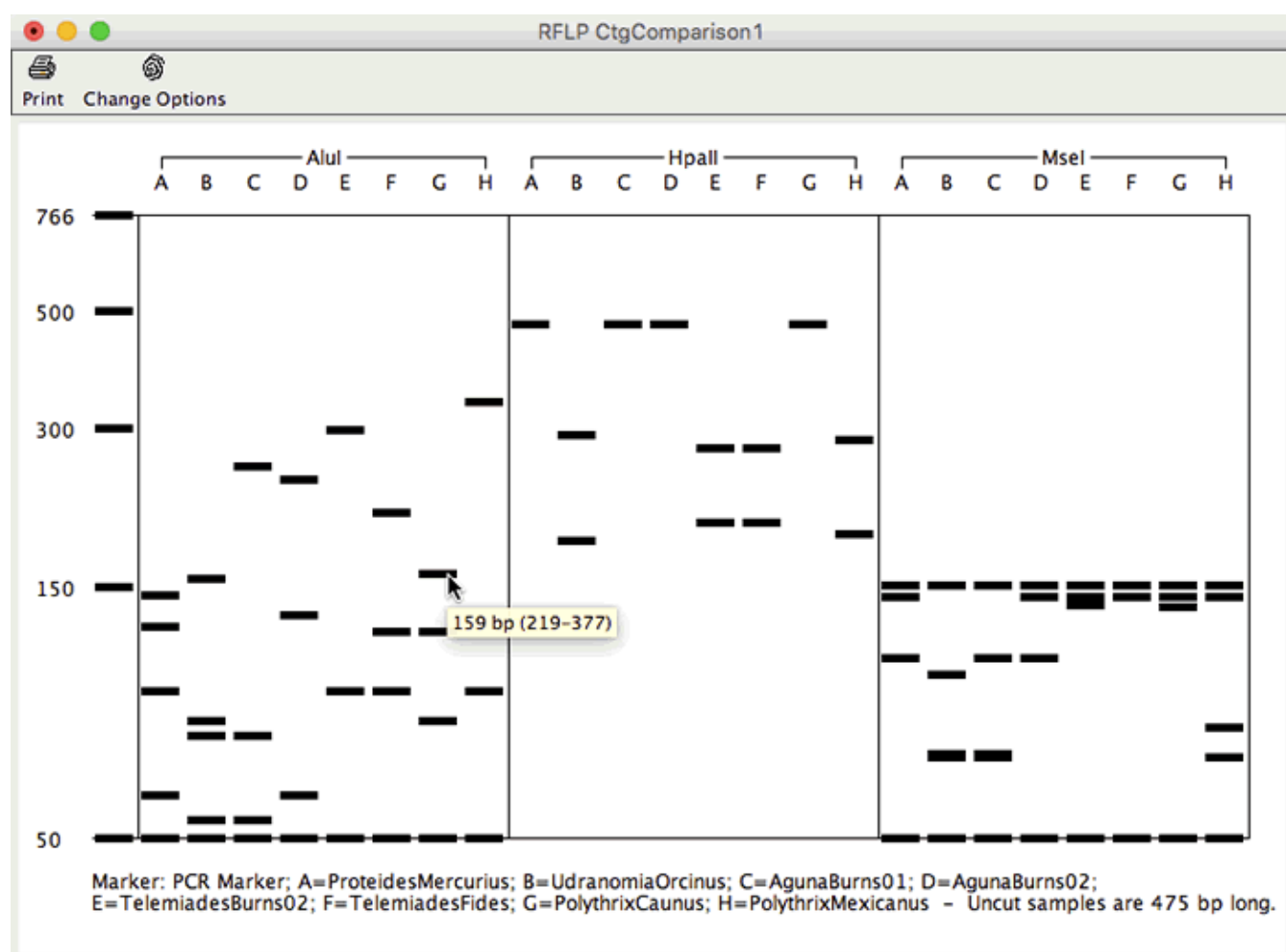
- **Best enzyme only:** The results are generated with only one enzyme - the one that produces the most different cut pattern.
- **3 best enzymes:** The 3 enzymes that produce the most different cut patterns are used for generating the resulting gel, aligned map, or table.
- **All with different cut patterns:** Shows results for all enzymes that cut at least one of your samples differently than the other samples.
- **All enzymes:** This will show all enzymes you selected in the top section. Note that if you are displaying the results as aligned samples only cut positions are displayed, so non-cutting enzymes are not included.

"Best" enzyme means the one that produces the most different cut pattern. For example if you compare 5 samples, and the first enzyme cuts all 5 samples differently, and the second enzyme cuts 4 samples the same way and 1 sample differently, then the first enzyme would be considered better. If the same number of samples are cut differently, then the enzyme that produces the most different fragment lengths will be considered better. If several enzymes are considered equally good, then the enzyme that comes first in alphabetical order will be used.

## Analysis Results

Results can be displayed as a virtual gel, aligned sequences, or text. The gel has one lane per sample and enzyme, arranged in groups by enzyme. The aligned samples are displayed below each other and show color coded cut positions, and the text consists of a table listing the fragment lengths for each sample and enzyme. Results can be printed.

An RFLP analysis for a contig comparison of 8 different species, displayed as gel and showing the best 3 enzymes, could look like this:



## CodonCode Aligner User Manual

In the example above, the enzyme AluI produces a different cut pattern for every one of the sequences, and enzymes HpaII and MseI cut differently for only some of the samples. Mouse overs show the fragment length and base numbers. Sample names are abbreviated with letters at the top of the gel, and a legend at the bottom maps each letter to the sample name. The information at the bottom of the gel also includes the marker and the length of the uncut sequences.

The options for each analysis can be changed using either the toolbar button or popup menu item called "Change Options". This allows you to change all parameters for the analysis, including enzymes used, display style, and which enzymes are displayed in the results.

Note that changing the contig or samples for an open RFLP result window does **not** update the RFLP results according to sample changes. Any changes will result in a red cross being drawn across the window to show that the result is no longer valid and was created for different data.

# Restriction Cloning

Simulate restriction cloning to visualize and plan the steps of your cloning process.

To start the cloning wizard, select **Restriction Cloning...** from the **Tools** menu:

The screenshot displays the "Restriction cloning" software interface. At the top, there are tabs for "Vector", "Insert", and "Product". The main window shows a circular plasmid map with various restriction enzyme sites labeled, including XhoI, HindIII, EcoRI, BamHI, NdeI, NcoI, XbaI, SphI, PvuI, XmaI, SmaI, ClaI, NruI, BstEII, EcoRV, HpaI, and BglI. The map also indicates the positions of the lacZ, KanR, ori, and p3000 regions. On the right side, the "Vector:" dropdown is set to "pET-28a(+)", and the "Cut with:" dropdowns are set to "XhoI" and "HindIII". The "Fragment to use:" section shows a selected fragment of 5354 bp (XhoI (5208) - HindIII (5192)) and a 15 bp fragment (HindIII (5193) - XhoI (5207)). At the bottom, the cloning process is summarized: a "Vector 5354 bp" (HindIII (5192) - XhoI (5208)) is combined with an "Insert 1520 bp" (HindIII (1521) - XhoI (2)) to produce a "Product 6874 bp" (HindIII - XhoI). The sequence context around the restriction sites is shown: Vector: ...aca TCGAgca... / ...tgtTCGA cgt...; Insert: AGCTtac...ccc / atg...gggAGCT; Product: ...acaAGCTtac...cccTCGAgca... / ...tgtTCGAatg...gggAGCTcgt... A green box on the right indicates "Ready to clone Product: 6874 bp". At the bottom right, there is a "Product name:" field containing "CloningProduct" and a "Clone" button.

## CodonCode Aligner User Manual

The "Vector", "Insert" and "Product" buttons at the top allow you to switch between them. Sequences and enzymes to use can be selected in the drop down boxes at the top right. Shift-clicking on any two enzymes will also pick the enzymes and highlight your chosen region. The bottom section displays the enzyme overhangs and how vector and insert fit together. The insert can be flipped by using the "<" or ">" buttons.

The map has multiple customizable options that can be found in the side toolbar on the left side of the map. You can choose specific features and enzymes to display. It is also possible to show enzymes based on number of cuts, for example showing only unique cutters to make picking enzymes for your cloning experiment easier. The map can be shown as circular or linear map, or you can view the bases instead:

The screenshot displays the "Restriction cloning" window in CodonCode Aligner. The interface is divided into several sections:

- Top Bar:** Contains buttons for "Vector", "Insert", and "Product".
- Left Panel:** A vertical toolbar with icons for map navigation and display options.
- Main Map:** A linear map showing the DNA sequence with various features and enzyme sites. Key features include:
  - Vector:** T7 promoter, lac operator, ribos., 6xHis, thrombin site, T7 tag, MCS, HindIII, XhoI.
  - Insert:** vaccine, ORF10, Nucleocapsid.
- Right Panel:** Metadata for the Vector and Insert.
  - Vector:** Source: pET-28a(+), Fragment: XhoI (5208) – HindIII (5192), Length: 5354 bp.
  - Insert:** Source: vaccine, Fragment: XhoI (2) – HindIII (1521), Length: 1520 bp.
- Bottom Section:** A diagram showing the assembly of the product. It includes the Vector (5354 bp) and Insert (1520 bp) with their respective enzyme sites (HindIII and XhoI) and the resulting Product (6874 bp). The sequence context around the insertion site is shown: ...acaTCGAgca... and ...tgtTCGA... for the vector, and AGCTtac...ccc and atg...gggAGCT for the insert.
- Bottom Right:** A green box indicating "Ready to clone Product: 6874 bp". Below it, there is a field for "Product name: CloningProduct" and a "Clone" button.

## CodonCode Aligner User Manual

The screen shot above shows the bases of the cloning product to verify the reading frame for the vaccine insert. Start and stop codons can be highlighted with red and green colored background for the bases which makes spotting the expected start codon (or unexpected stop codons) easy. All features and cut sites are also displayed when viewing the bases, and mouse overs show details for them.

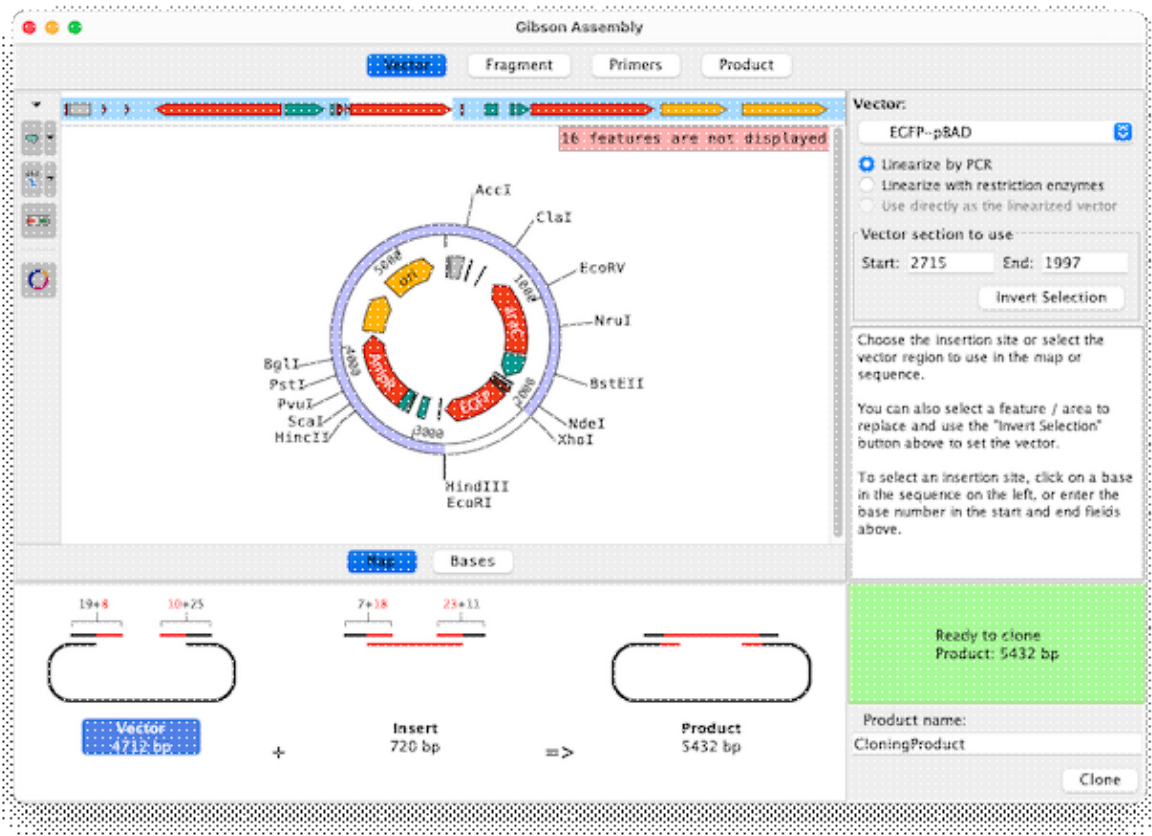
You can name your cloning product in the bottom right corner of the view. Clicking on the "**Clone**" button will create the cloned sequence in your project.



# Gibson Assembly

Simulate Gibson Assembly with one or multiple inserts to visualize and plan the steps of your cloning process.

To start the cloning wizard, select **Gibson Assembly...** from the **Tools** menu.



The "**Vector**", "**Fragment**", "**Primer**", and "**Product**" buttons at the top allow you to switch between them.

The bottom section shows how vector and insert(s) will form the assembly product, as well as the primers when cloning with just one fragment.

Sequences to use can be selected in the drop down box on the right. To linearize the vector, you have the option to either "**Linearize by PCR**", "**Linearize with restriction enzymes**", or to "**Use directly as the linearized vector**". The option to use the vector directly as is, is only available if the vector is already a linear sequence.

Clicking on a specific feature will select all bases for this feature. The "**Invert Selection**" button makes it easy to flip the selection, which was used in the example above to select all but the "EGFP" section of the vector.

You can name your cloning product in the bottom right corner of the view. Clicking on the "**Clone**" button will create a folder in your project that contains the cloned sequence and any other sequences you chose to create as part of your cloning experiment. Which sequences are added can be chosen in the "**Product**" tab. You can **create** and / or **export** the linearized vector, fragment(s) and the primers as sequences in your project by selecting the matching check boxes.

In the "**Fragment**" part, you can add up to 15 fragments using the "**Add**" button. Use the left and right arrow buttons or the drop down box to view and change the different fragments. The "**Order**" button allows you to change the order of the fragments for your cloning project. The direction of each fragment can be changed by selecting the matching direction button on the right side of the view when the fragment is shown. The option to use a fragment directly as linearized fragments is only available if your cloning product contains only one fragment, and the vector is created with PCR, and the fragment is a linear sequence.

The "**Primer**" tab shows the primers for this cloning experiment. If this section is shown the first time, or if you click on the button "Pick overlapping PCR primers..." on the top right, the primer picking dialog is shown:

**Pick Overlapping PCR Primers**

Regions to amplify:

Vector: 2,715 .. 1,997 = 4,712 bp

Fragment1: 979 .. 1,698 = 720 bp

Target Tm for PCR primers: 60 °C

Overlapping ends:

☐ 40 overlap bases

☒ 15 to 25 overlap bases with minimum Tm of 48 °C

Add overlap to: both vector and fragment primers

Cancel Pick Primers

You can set the  $T_m$  for the PCR primers as well as the overlap region. For simple Gibson assembly reactions overlapping ends from 15 to 25 bases and a  $T_m$  of 48°C or more is recommended. Large or multiple fragments may require longer overlaps for a successful reaction.

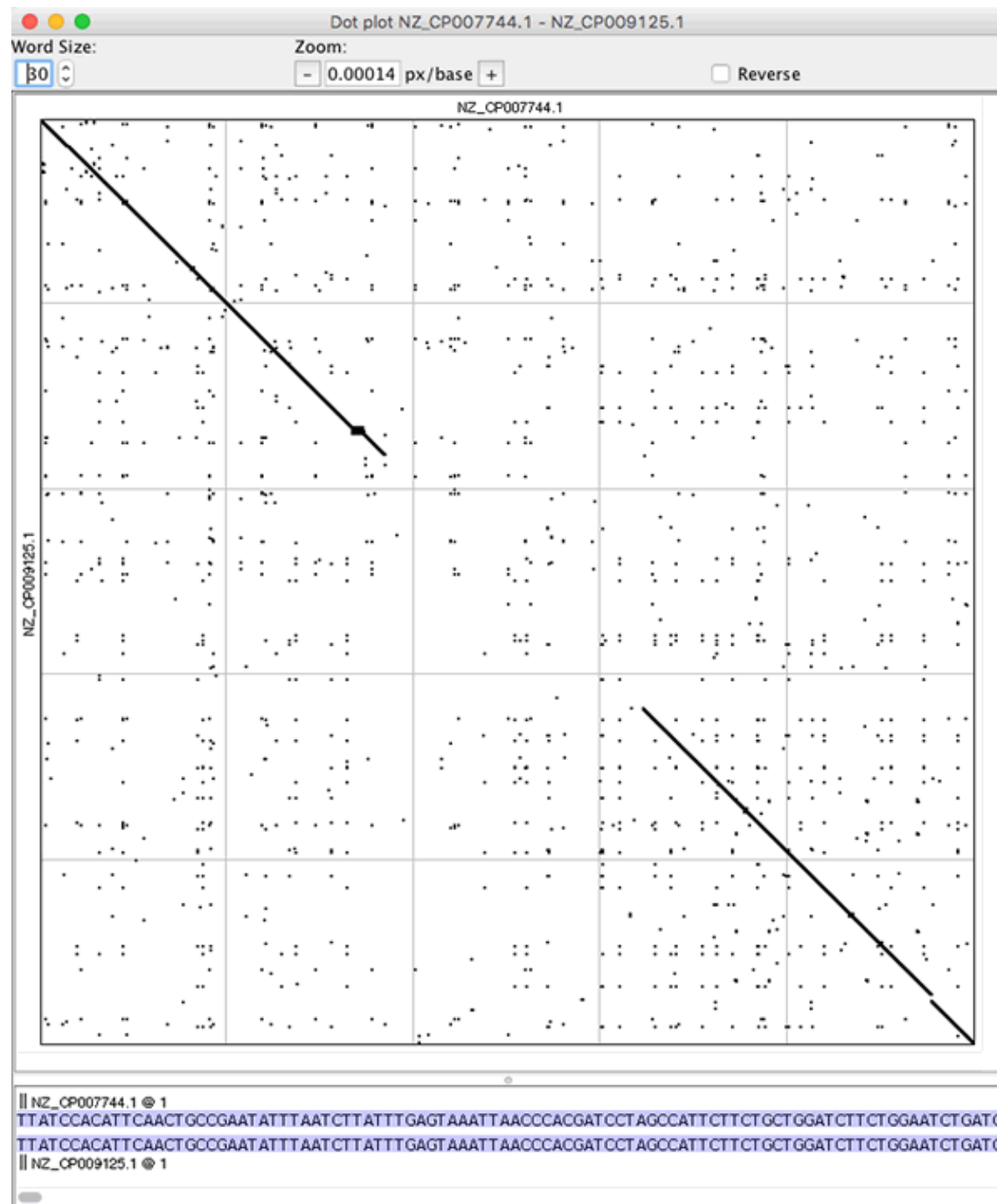
Depending on your cloning experiment you can choose to add the primers either to the fragment or vector only, or to both, the vector and fragment.

The "**Primer**" section shows all of the primers for your cloning experiment in a table that also allows you to edit the primers manually. This can be useful to add spacers or adjust the reading frame if necessary.

# Dot Plots

Dot plots are a graphical view of the similarities between sequences. CodonCode Aligner can show dot plots that display shared words (k-mers) of a user-selectable length, which allows fast generation of dot plots even for large sequences.

To display a dot plot for selected samples or contigs, choose **Dot Plot** from the **Tools** menu :



A dot plot can be shown for one or two samples, but you can also generate and show several dot plots at once by selecting multiple samples. You will be given the option to choose which of the selected samples should be displayed horizontally and which vertically. Each horizontal sample will be compared to each vertical sample. This allows you, for example, to compare dot plots for several partial sequences to a reference sequence:

Dot plot DM1-3-516-R44 ...



## CodonCode Aligner User Manual

The bottom of the dot plot window shows the aligned bases for the selected base positions in a plot above. Matching bases are shown with a light blue background. The selected position is highlighted by blue crosshairs in the dot plot. The crosshairs can be set by clicking on a plot with the mouse and by using the arrow keys.

Mouse overs are displayed for the sample names and base positions in each dot plot.

At the top of the dot plot window you can change the word size, the zoom, and if the reverse complement of the vertical sequence should be included in the plot. The word size is the word length used when finding matching positions. Increase the word size to get rid of unwanted noise, and reduce it to see more matches. Generally it also makes sense to use a larger word size for longer sequences. You can zoom in and out with the + and - buttons at the top. The current zoom level is displayed in pixels per base.

The dot plot window can be printed using the "File -> Print..." menu item, or the key board shortcut "control P" on Windows and "Command P" on Mac OS X.



# Editing Samples

CodonCode Aligner allows you to edit your sequences - but before you start editing, remember that

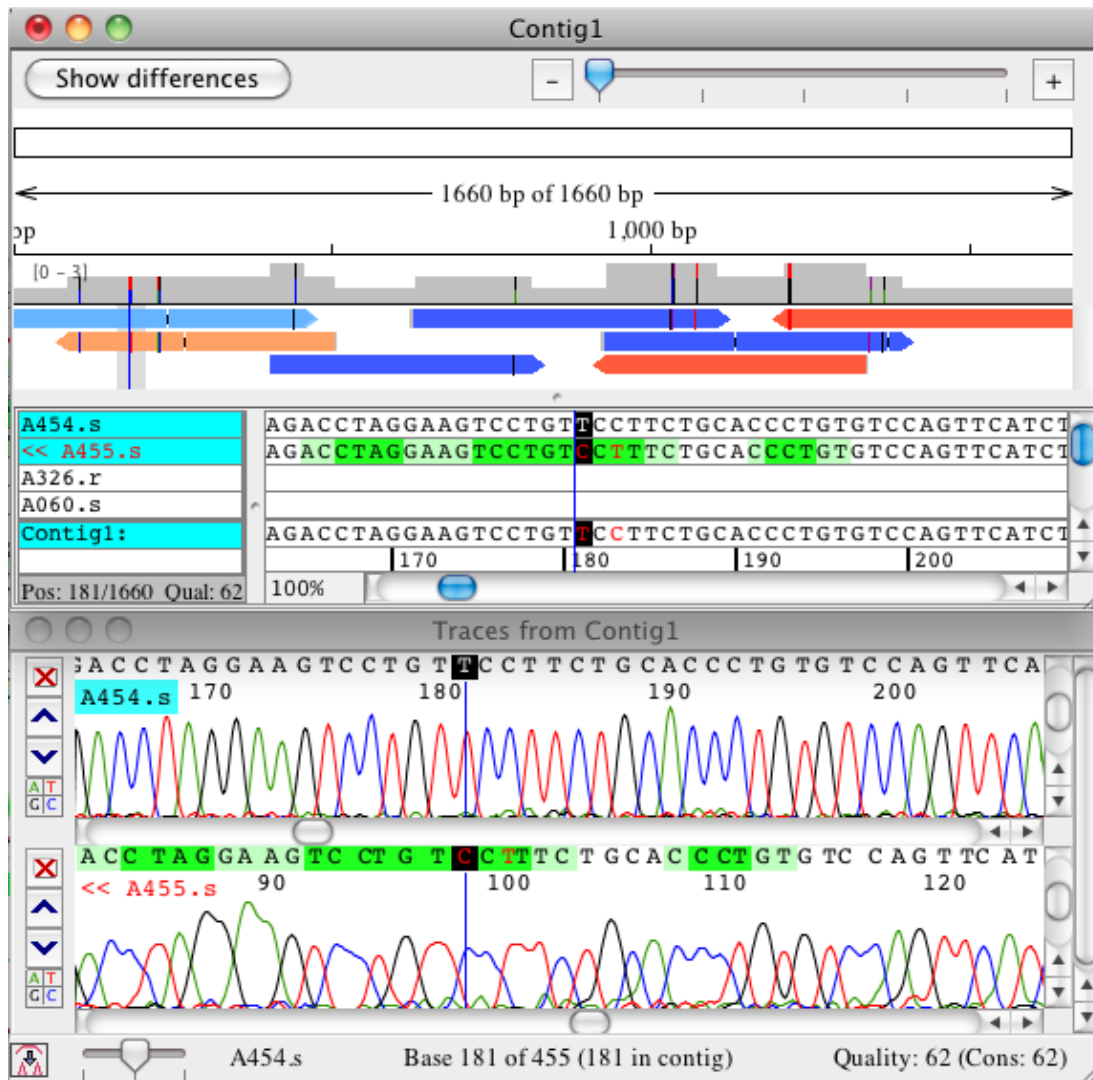
## You may not need to edit!

This is because Aligner will examine the quality scores of all aligned bases at each contig position when building the consensus - in other words, Aligner typically builds a **quality-based consensus**, rather than a majority-based consensus.

In the days before Phred quality scores, editing discrepancies was important: consensus sequences were always majority-based, and the only way to make sure that the consensus sequence was indeed correct was to look at the discrepancies. Typically, scientists would simply correct any wrong base calls, so that they would not have to look at the same region again.

With samples that have Phred (or Phred-like) quality scores and sequence assembly algorithms that use the quality scores, a lot of this editing is not necessary anymore. Typically, wrong bases will have lower quality scores, and the correct "consensus" can be determined automatically by just looking for the highest-quality base at any position. The low-quality discrepancies can often safely be ignored.

Here is a typical example - a region covered by two sequences which differ at several places:



At the cursor position, Aligner chose the 'T' from sequence 'A454.s' as the consensus, since it is of much higher quality than the 'C' in sequence 'A455.s' (the higher quality is indicated by the lighter background). You can also see one other discrepancy in this region; in each case, the sequence from 'A454.s' is high quality and correct, and therefore chosen as the consensus sequence.

If you would look at the same region in an older assembly program that only supports a majority-based consensus, the consensus base at all different locations would be an ambiguity, or the single called base at places where one sequence has a gap character. To get a clean consensus, you would have to look at all different regions, and edit every time, to come to the same end result.

Also note that Aligner uses the quality scores to estimate how likely it is that the consensus quality is correct (this can be done with rather good accuracy since the quality scores are linked to error probabilities). In the region shown above, the estimated consensus quality is very high, since one of the two sequences is of very high quality through the entire region. When editing contigs with Aligner, you can define quality thresholds for the consensus sequence in the [Feature preferences](#), and use this to [very quickly move](#) to regions that need your attention.

To build a quality-based consensus, two prerequisites need to be met:

1. The sequences must have base-specific quality scores (if you import sequence trace files that do not have quality scores, you should first do the "[Base calling](#)" with Phred to get quality scores).
2. You must have selected to build a quality-based consensus in the [Consensus preferences](#) (this is the default setting when you install Aligner).

In addition to the quality-based consensus, there are two other factors that enable you to get cleaner assemblies with much less need for "contig editing" are:

- The ability to automatically clip low-quality sequence from the end of reads, thereby reducing the total number of discrepancies.
- The use of local alignments (as opposed to end-to-end alignments); regions with high error rates tend to end up in the unaligned ends, and can also be ignored.

Of course, there will be times where contig editing is necessary, and CodonCode Aligner does provide many editing functions that are described on the next pages.

# Windows for Editing Samples

You can edit sample sequences in several different views (windows) - the [base view](#), the [trace view](#), and (for assembled or aligned samples) the [contig view](#). The only views that do not allow editing are the views that do not display the bases - the [project view](#), the [quality view](#), and the [feature view](#).

In general, **we suggest to do most editing in the trace view window**, simply because you can see the underlying data while editing. One possible exception of this rule is editing sequences that are part of contigs - sometimes it may make sense to edit in the contig view (but having a trace view open that shows the sequence is still a good idea!).

All windows are linked. If you move the cursor or make a selection in a contig window, Aligner will scroll to the same position in any open base view and trace view windows for that sample. Likewise, when editing base calls, changes are immediately made in all other open views for that sample.

# Cursor Positioning and Movement

The cursor is positioned by clicking at the desired position in a sample or the consensus. The cursor position is indicated by a vertical cursor position line displayed across all samples and the consensus.

## Moving to Features, Ambiguities, Mismatches or Edited Bases

The **Go** menu has selections for quickly moving through the sequence:

- **Next or Previous Feature:** Moves to the next or previous feature (or "region of interest"). Features can be discrepancies, low-quality consensus bases, low coverage regions, or a number of other things that may be of interest to you. You can define *your* regions of interest in the [feature preferences](#).  
*Tip: the fastest way to navigate to the next or previous feature is by using the **keyboard shortcuts** : On Windows, use Control-Right Arrow and Control-Left Arrow; on OS X, use Command-Right Arrow and Command-Left Arrow*
- **Next or Previous High Quality Mismatch:** Moves to the next or previous high-quality base that disagrees with the consensus
- **Next or Previous Low Quality Consensus:** Moves to the next or previous place where the consensus quality is low.
- **Next or Previous Ambiguity:** Selects the next or previous ambiguity in the selected sample or consensus.
- **Next or Previous Mismatch:** Moves the cursor to the next or previous mismatch in the selected contig.
- **Next or Previous Edited Base:** Selects the next or previous edited base in the selected sample.
- **Base Number:** Displays a [dialog](#) where you specify a specific base number in the selected sample or consensus.
- **First Aligned Base:** Moves the cursor to the first aligned base in the current sample.
- **Last Aligned Base:** Moves the cursor to the last aligned base in the current sample.

In addition to the menu items in the "**Go**" menu and their keyboard shortcuts, you can also use the "home" key to go to the first base of a sequence, and the "end" key to go to the last base of a sequence.

# Setting Base Numbers

Usually, base numbers shown in CodonCode Aligner are relative to the first base in a sample or contig, and gaps are included in the count. Sometimes, you may want to change the numbering, for example to indicate the start of a region that you are interested in. You can set the base numbering for any sample or contig as follows:

1. Select the base that you want to assign a number to (you can do this in a base view, trace view, or contig view).
2. Go to the "**Sample**" menu, and select "**Set Base Number...**". This will open a dialog.
3. Enter a base number (greater than 0), and press "OK".

When setting base numbers for contigs, please note that adding additional samples to an existing contig will re-built the contig, and the base numbering will be reset (so the first base in the contig is base 1).

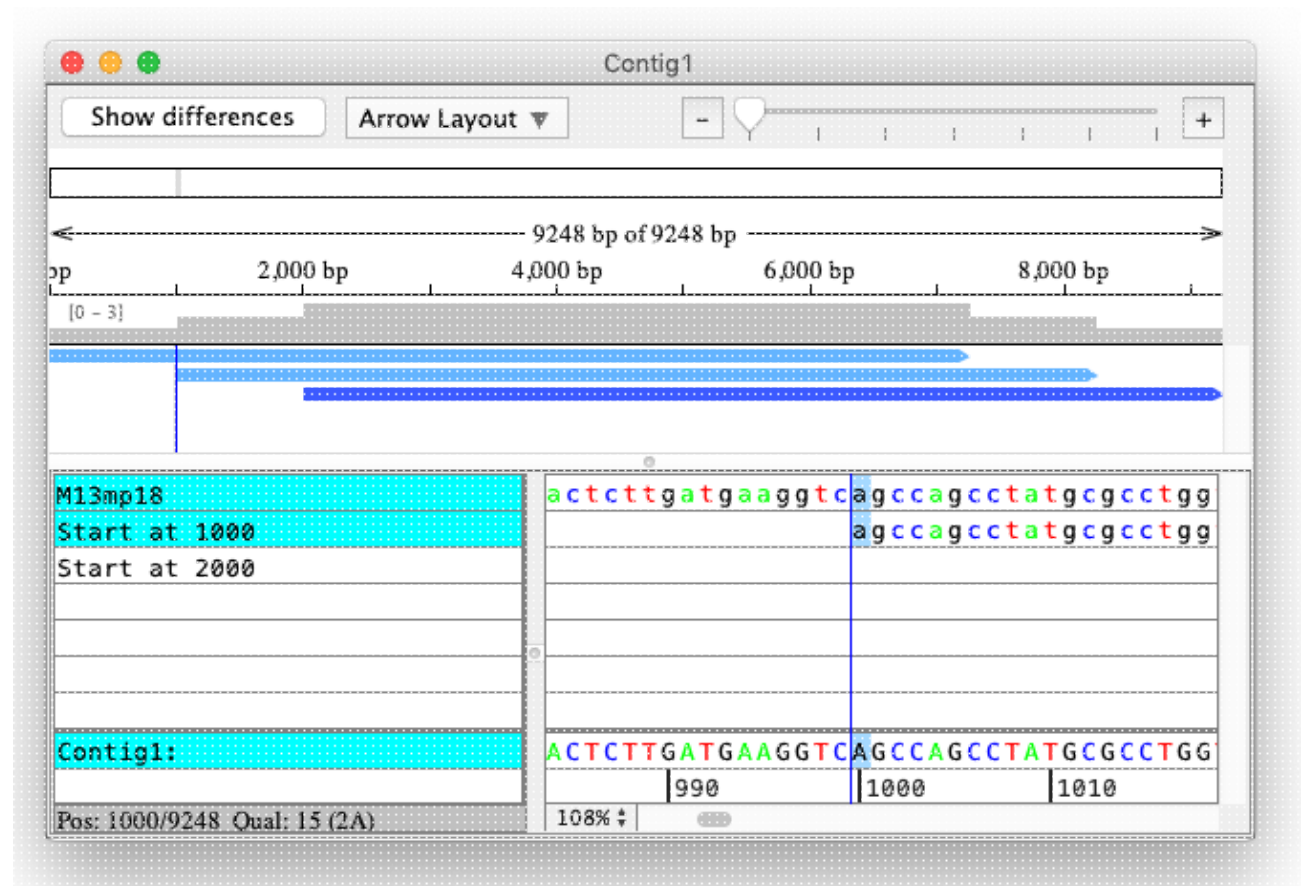
## Rotating circular samples

For samples that are circular and unassembled, the "**Set Base Number...**" function can be used to effectively "rotate" the sample, changing the first base that is displayed in the base view. Selecting any base, and then setting the base number to 1, will rotate the sequence so that the selected base is now the first base; any bases that were previously before this base are now added to the end of the sequence.

Setting the base number of a selected base in a circular, unassembled sample to a number other than 1 will show a dialog asking whether or not the sequence should be rotated. If the user selects to not rotate the sample, only the numbering of bases will be changed. This also applies for circular samples that are in contigs, and which therefore cannot be rotated.

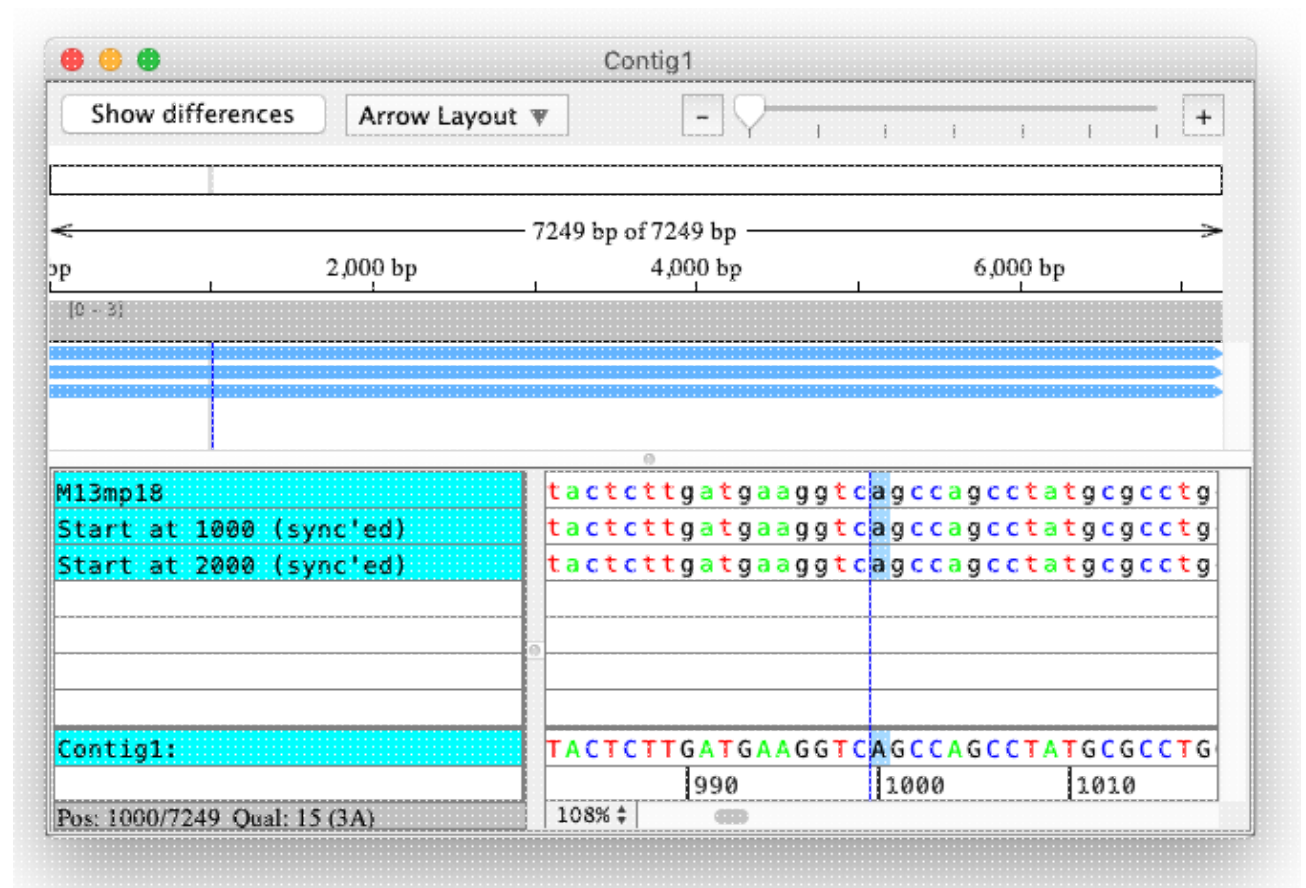
# Synchronizing Sample Starts

For better alignment of circular DNA samples, for example cloning constructs, CodonCode Aligner can "synchronize" the starts of circular samples. Since circular DNA does not have a "naturally" defined start, the first base is often selected randomly, for example when the sequence of a construct is determined by a sequencing service or assembling. When aligning several such samples that have different start bases, the overlap between samples will be only partial, with parts of samples overhanging at the start or end. This is illustrated in the following image:



## CodonCode Aligner User Manual

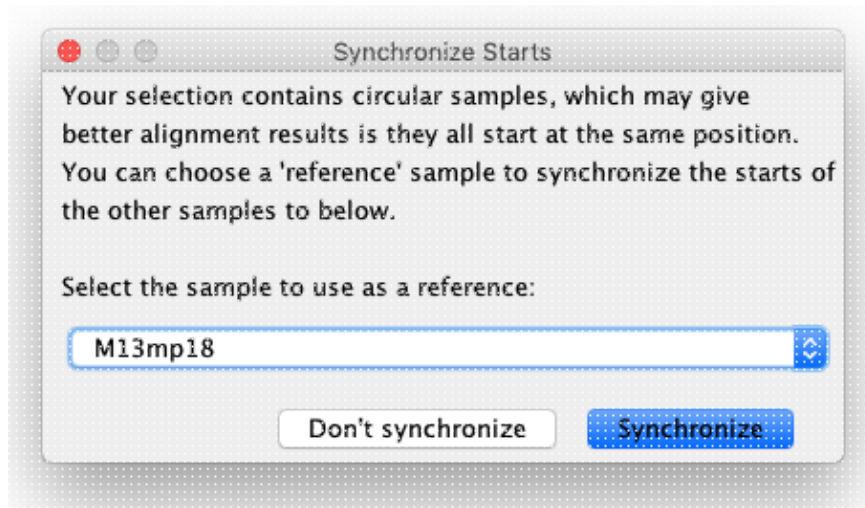
After synchronizing the starts of the samples, a better alignment over the entire sequences can be generated:





To synchronize the starts of sequence, select several samples. The samples to be synchronized need to be circular and unassembled; you can also include one linear sequence to serve as a "reference", to which the starts of the circular samples will be aligned. If no linear sequence is included, CodonCode Aligner will show a dialog where you can choose one of the samples in your selection as the reference.

When trying to assemble or align circular sequences in CodonCode Aligner, and all the selected sequences are unassembled, circular samples, Aligner will check the samples to see if their starts have already been synchronized. For samples that do not appear to have synchronized starts, CodonCode Aligner will show a dialog offering to synchronize the starts, where you can choose the reference sequence for the synchronization:



If you press the "Synchronize" button, CodonCode Aligner will try to align the starts of the circular samples in your selection before proceeding with the assembly or alignments.

# Selecting Bases

To select bases, move the mouse cursor over the first base you wish to select (in the base view, contig view, or trace view). Click and hold down the mouse button, drag the cursor horizontally over the base(s) and release the mouse button after dragging the cursor over the last base. *This can be a bit slow for large selections; if you want to select all bases to the start or end of the sequence, it can be a lot faster to use one of the options described below, so please read on!*

Selected bases are displayed with a different background color than surrounding bases. If the selection was made in the consensus, the selection is also shown in all samples that are part of the consensus.

## Selecting from sequence start to current cursor position

To select all bases from the start of the sequence to the current cursor position, choose "**Select from Start to Here**" from the "**Edit**" menu. The selection will include any unaligned bases at the start of the sequence (up to the cursor position).

You can accomplish the same thing without having to use the menus by keeping the "shift" key pressed while pressing the "home" key.

## Selecting from current cursor position to the end of a sequence

To select all bases from the the current cursor position to the end of the sequence, choose "**Select from Here to End**" from the "**Edit**" menu. The selection will include any unaligned bases at the end of the sequence (up to the cursor position).

You can accomplish the same thing without having to use the menus by keeping the "shift" key pressed while pressing the "end" key.

## Selecting all bases in a sequence

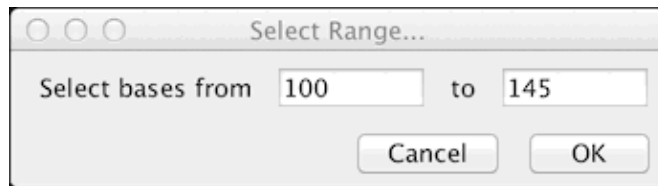
To select all bases in a sequence, choose "**Select All**" from the "**Edit**" menu. Alternatively, you can use the keyboard shortcut - Control-A on Windows, and Command-A on OS X (keep the control- respectively command-key pressed while pressing 'A').

The selection will include any unaligned bases at the start and at the end of the sequence.

If the project view is the active view, then "**Select All**" will select all contigs and samples shown in the project view. The "Unassembled Samples" and "Trash" folders will also be included, unless they are empty.

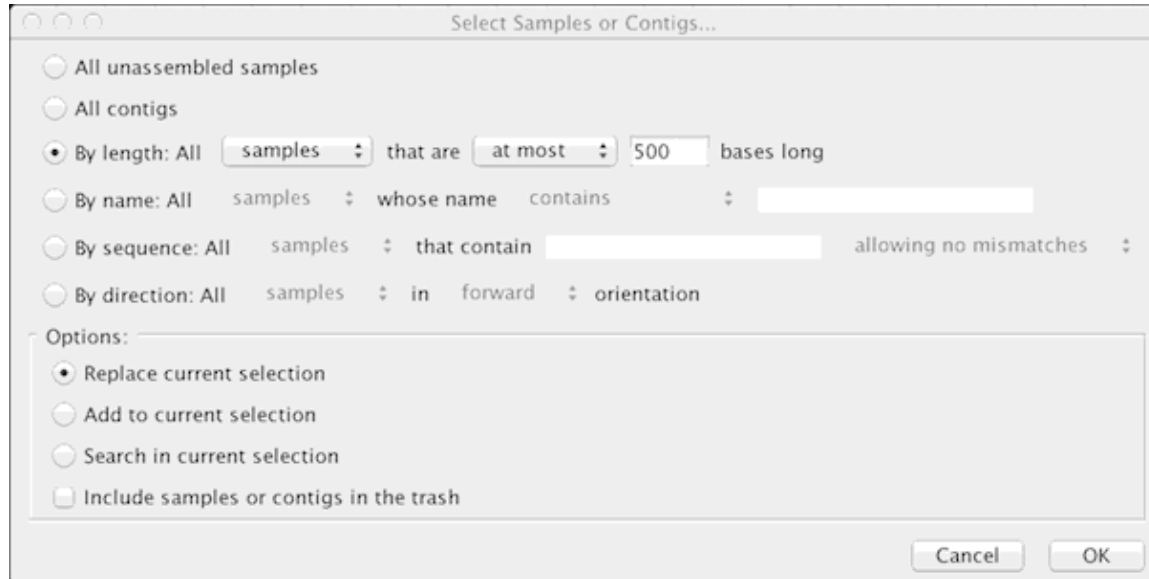
## Selecting Ranges of Bases

To select a range of bases by number, choose "**Select Range...**" from the "**Edit**" menu. This will display a dialog where you can enter the selection range:



# Selecting Samples and Contigs

To select samples or contigs, or to modify a current selection, based on sample name, length, sequence, or direction, choose "Select Samples or Contigs..." from the "Edit" menu. This will display the following dialog:



The options in the top half of the dialog determine the search criteria. The options in the bottom of the dialog determine if the search results should replace whatever is currently selected; be added to the current selection; or be searched only in the current selection. By using "Select Samples of Contigs" several times in a row, this allows refinement of selections by multiple search criteria.

# Changing Bases

CodonCode Aligner supports [manual editing](#) in all views that show bases - the base view, trace view, and contig view.

In addition, CodonCode Aligner provides several functions for [automatic editing](#), and to [change bases to lower or upper case](#).

## Manual Editing

To change a single base, select the base and type a new base letter. Letters for ambiguity codes can also be entered.

Be careful if more than one base is selected before you start typing: all the selected bases will be replaced with a single base, which may not be what you want.

After changing a base, Aligner will automatically move the selection one base to the right. This allows you to quickly edit a number of bases in a row.

The consensus sequence is automatically updated with any changes, if such changes introduce a change in the consensus sequence. Edits made in one View are automatically updated in other Views of the same sample.

If you **edit a consensus base**, then the change will be applied to all samples that have aligned bases at this position - so once again, be careful!

You can also [delete bases](#) by using the "delete" and "backspace" keys, and [insert new bases](#) by pressing the "space" or "shift-space" keys, as described in detail on other pages.

## Making Bases Lower or Upper Case

Usually, bases in CodonCode Aligner are shown in upper case. Sometimes, you may want to change the case, for example to indicate a region that you are interested in. You can change bases to lower or upper case as follows:

1. Select the bases that you want to change (you can do this in a base view, trace view, or contig view).
2. To make these base lower case, go to the **"Edit"** menu, and select **"Make Lower Case"**. To make base upper case, go to the **"Edit"** menu, and select **"Make Upper Case"**.

This will change any selected bases to the chosen case. Upper and lower case regions in contigs are preserved when adding new samples to a contig, and when exporting sequences to text files.

*Note that editing bases in CodonCode Aligner will make bases in samples lower case, unless you press the "shift" key while editing. By definition, lower case edited bases indicate lower-quality (confidence) edits, while upper-case bases indicate higher-quality edits.*

## Automatic Edits

CodonCode Aligner offers several functions to automatically change bases, for example to call heterozygous bases, or to convert low-quality bases to N. To auto-edit bases, select the bases you want to edit, go to the "Edit" menu, and choose "Change Bases". This will show a submenu that allows you to choose the automatic edit you want to apply to your selection. Available auto-edits are:

- [Match Consensus](#)
- [Call Secondary Peaks](#)
- [Remove Ambiguities](#)
- [Change Low Qual Bases](#)
- [Undo Auto Edits](#)

For some of these auto-edits, you can set thresholds by selecting "Change Bases Options..." from the "Change Bases" menu, which will display the Change Bases Options dialog.

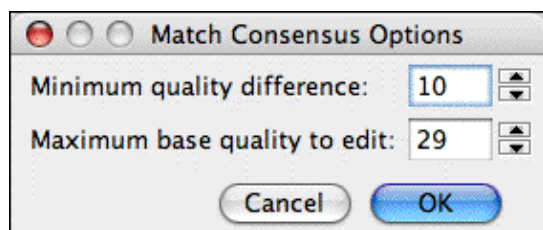
All auto-edits will be done on the currently selected base or bases. Please note:

- When you select consensus bases in a *contig view*, the edits will affect *all samples at this position*.
- When you select samples in the *project view*, the changes will be done to *all bases in the selected sample(s)*.

### Match Consensus

The "Match consensus" option will automatically change bases to match the consensus of the contig they are in (it will have no effect on unassembled samples). One example where this option is useful is changing bases in "dye blobs" in DNA sequences, where many bases are wrong because of artificial strong peaks.

This option is intended for low-quality sequence, and will not change high-quality bases. You can change the thresholds in the "[Change Bases Option...](#)" dialog by clicking on the "Options..." button next to "Match consensus". This will show the following dialog:



The number at the top is the minimum difference in sequence quality between the consensus and the base in question. The default setting of 10 means that bases will only be edited if the consensus sequence quality is at least 10 higher, corresponding to a 90% or higher probability that the edit is correct (assuming that this is a random error).

The number at the bottom determines the highest quality a base can have to be edited; with the default setting, any base with a quality score of 30 or higher will not be edited.

### Call Second Peaks Higher Than...

The "Call second peaks" option allows you to convert all bases with a significant secondary peak to ambiguities. You can set the threshold of secondary peaks, relative to the intensity of the first peak, in the "[Change Bases Option...](#)" dialog (the default is 25%).

This method can sometimes be useful in mutation detection. Please note, however, that it is much simpler than the methods used in CodonCode Aligner's "Find mutations" function, and therefore much more prone to both false-positives and false negatives. For example, the "Find mutations" function compares peaks at a given position to other peaks in other samples at this position, and can therefore detect heterozygous bases from a drop in peak intensity, even if the secondary peak is very small.

## Change Ambiguities to Single Bases

This is the reverse of the previous function - any bases with ambiguity codes will be converted to regular bases (A,C,G, or T), according to the highest peak at the base position. This option is useful if you have sequences that contain ambiguity codes, but you know that there are no heterozygous bases (for example because you sequenced a clone, not a PCR product).

## Change Low Quality Bases to N

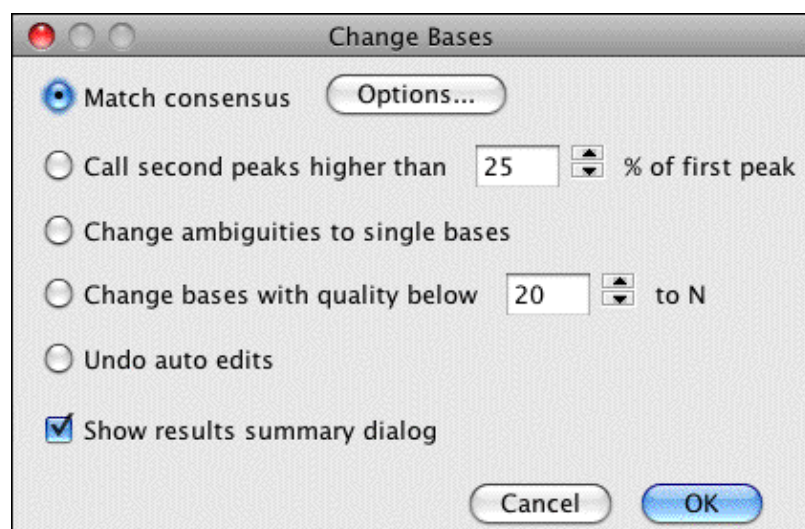
This option allows you to change all bases with a quality below a given threshold to 'N'. The default threshold is 20; you can change it in the "[Change Bases Option...](#)" dialog.

## Undo Auto Edits

This option will undo all previous auto-edits (any of the options described above) for your current selection, and convert bases back to the base calls before the first auto edit. To function properly, this requires that the "auto edit" tags that Aligner added originally are still present. As long as your samples stay in the same project, this is usually not a problem. However, if you export samples and then re-import them, the tags will typically be lost, and undoing auto-edits with this option will not be possible.

## Change Bases Options

The "Change Bases" dialog shows all options available to auto edit bases. It allows to set certain options that are used when doing bases edits (for example changing only bases below a certain, user-settable quality to N). It also enables you to show or hide a result dialog for the auto edits. To show the options dialog for changing bases, go to the **Edit** menu, select **Change Bases**, and choose **Change Bases Options...**. This will show the following dialog:



## CodonCode Aligner User Manual

Use the radio buttons to choose how you want to change bases (see below). You can also choose whether or not you want to see a dialog summarizing the changes.

CodonCode Aligner will add tags to all auto-edited bases. You can undo all changes right after you made them, or later with the "Undo auto edits" selection in this dialog.



# Deleting Bases

## Using the Keyboard to Delete Bases

**Backspace Key:** Deletes the currently selected base or bases; then, for sequences in contigs, the bases to the left of the cursor are shifted to the right.

**Delete Key:** Deletes the currently selected base or bases; then, for sequences in contigs, the bases to the right of the cursor are shifted to the left.

*Note: If the consensus is selected, Delete or Backspace deletes not only the base(s) in the consensus, but bases at that position in all samples having sequence at that position.*

*OS X Users: On Macintosh keyboards, the Backspace key is actually labeled "delete", not Backspace! It is part of the main key group. The "Delete" key is labeled "del" (above an "X" in a box), and usually is part of a small group of keys that include the "page up" and "page down" keys.*

## Menus for Deleting Bases

Instead of using the keyboard shortcuts for deleting bases, you can also go to the "**Sample**" menu, select the "**Delete**" submenu, and then choose one of the following items:

**Selection - Fill from Left:** Same as Backspace key.

**Selection - Fill from Right:** Same as Delete key.

**From Sample Start:** Deletes all bases to the left of the cursor for the selected sample.

**To Sample End:** Deletes all bases to the right of the cursor for the selected sample.

# Deleting Samples

Deleting samples in Aligner is a two-step process: first, you move samples to the "Trash" folder, and then, you empty the trash. To move a sample to the trash, select the sample (for example in the project view), and then choose "**Move to Trash**" from the "**Edit**" menu. For samples in contigs, some special considerations apply, as described in the "[Editing Contigs](#)" section.

Samples in the trash can be re-used by selecting them in the project view, and then choosing " **Move to Unassembled Samples**" from the "**Edit**" menu.

To permanently delete samples from the trash and remove them from your project, choose "**Empty Trash...**" from the "**File**" menu. The corresponding sample files in your project folder will be deleted when you save the project the next time.

# Moving Gaps and Samples

Sometimes, you may find that some of the gaps introduced during assembly are not quite at the right positions, and should be moved around a bit; aligning gaps in all reads may even enable you to remove an entire column of gap characters.

## Moving gaps in contigs

To move gaps, select the gaps in the contig view (or in the base or trace view), and then select "**Move Gap Left**" or "**Move Gap Right**" from the "**Sample**" menu. You can also use the keyboard shortcuts:

- Option-F5 for "**Move Gap Left**"
- Option-F6 for "**Move Gap Right**"

(Due to a Java bug on OS X, the keyboard shortcuts are not always shown in the menus.)

In the contig view, you can also move gaps around by drag and drop: select the gaps; then click on them again, but keep the mouse pressed and move the cursor to where you want the gaps to be; then release the mouse.

Contig gaps at positions where one or more samples have bases that are not gaps, can only be moved if the bases are deleted. CodonCode Aligner will show a warning dialog before deleting bases to move gaps the first time.

## Moving samples in contigs

You can also move entire sequences one base to the left or to the right within a contig by selecting the sequence, and then choosing "**Move Sequence Left**" or "**Move Sequence Right**" from the "**Sample**" menu.

## Moving samples to "Unassembled Samples"

You can move samples from contigs or from the "Trash" folder to the "Unassembled Samples" folder by selecting the sample(s), and then choosing "**Move to Unassembled Samples**" from the "**Edit**" menu. For samples in contigs, some special considerations apply, as described in the "[Editing Contigs](#)" section.

In the project view, moving samples to the Trash or Unassembled Samples can also be done using drag and drop.

# Inserting Gaps and Bases

To **insert gaps**:

- Position the cursor at the base after which you want to insert a gap
- Press the space bar (or select "**Insert gap => Shift Bases Right**" from the "**Sample**" menu).

A new gap will be inserted before the current cursor position. If the sample is part of a contig, the bases *after* the new gap will be shifted to the *right*.

One exception to this rule applies when the last base of a sample is selected: in this case, the gap will be inserted *after* the current base, so that you can [add bases to the end of reads](#).

If you want to insert a gap and **shift the bases before the gap to the left** (instead of the bases *after* it to the *right*), select "**Insert gap => Shift Bases Left**" from the "**Sample**" menu. Alternatively, use the keyboard shortcut Shift-Space (keep the shift key pressed while pressing the space bar).

If you are working in the contig view and have a consensus base selected, gaps will be inserted in the consensus and in all sequences at this position.

To **insert a base**:

- insert a gap character as described above
- then type the letter of the base you wanted to insert.

Since the gap you just inserted was selected after you inserted the gap, you just replaced it with the character you typed.

## Adding Bases at the End of Reads

To add bases to the end of a read:

1. Select the last base of the sample
2. Press the space bar to insert a gap
3. Press the letter of the base you want to add to change the gap to the base
4. Repeat steps 2 and 3 for each base you want to add.

The gaps and bases that are added this way will not necessarily line up exactly with the peaks - instead, they will be spaced evenly, with the same spacing as the last 20 or so bases before the last base. This is usually ok if you need to add just a few bases. If you need to add many bases, you may be better off repeating the [Base Calling](#) for this sample (if the sample is in a contig, you first have to move it to Unassembled Samples).

You cannot directly insert a gap or a base right before the last base in a sample, but there is an easy workaround: just insert a gap after the last base, and then choose "**Move => Gap Left**" from the "**Sample**" menu.

***Note:** Aligner is not intended as an editor where you enter sequence by hand; there are plenty of other programs out there that allow you to do that. Aligner is intended to be used with sequence traces, preferably traces with Phred- or Phred-like qualities.*

# Reverse Complementing

To reverse complement a sample or contig:

- Go to the project view
- Select the contig (or a sample in the "Unassembled Samples" folder)
- Choose "**Reverse Complement**" from the "**Edit**" menu

In most views, you will be able to identify samples that have been reverse-complemented by the "<<" prefix before the name, and by the fact that the name is drawn in red. In the project view, you can identify reverse-complemented samples by the icon (it has red rather than black borders).

Reverse-complementing of unassembled samples is possible, but usually not necessary, since samples will be reverse-complemented as needed during assembly and alignment.

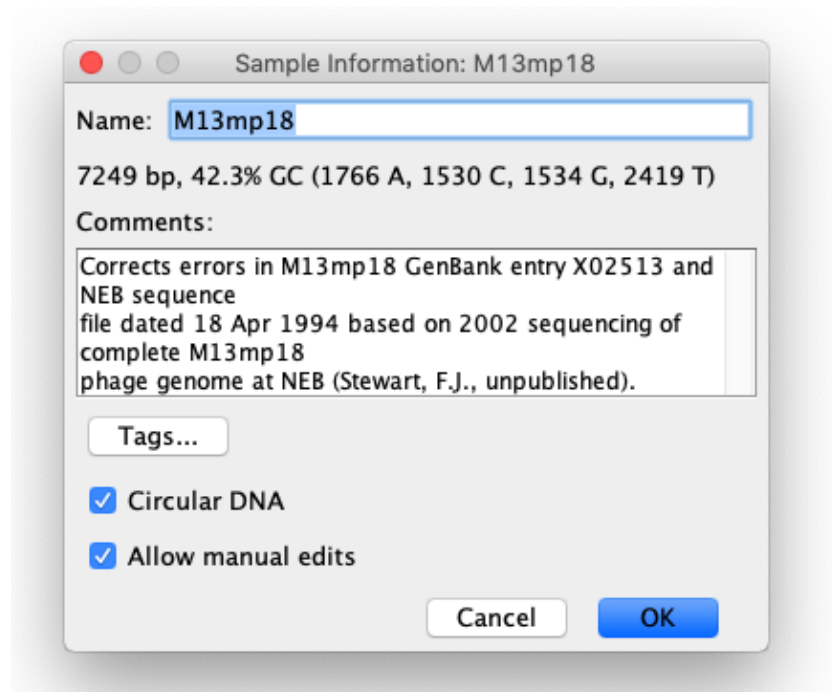
You cannot reverse-complement samples in contigs directly, since that would destroy the alignment to the consensus. If you select "**Reverse Complement**" when a contig view is in the foreground, the entire contig will be reverse-complemented.

# Editing Sample Information

To view information about a sample, or to change a sample name:

- Select a single sample in the project view (or in trace view or contig view)
- Go to the choosing "**Sample**" menu, and select "**Sample Information**"  
(or use the keyboard shortcut: control-I on Windows, command-I on OS X)

This will show the sample information dialog:



You can edit the sample name in the "**Name**" field, and add or change remarks about the sample in the comments field. The length and base composition is shown below the name.

The "**Circular DNA**" checkbox indicates if a sequence is circular (for example a plasmid cloning vector) or linear. It will be set automatically for circular sequences that are imported from files in Genbank or EMBL format. For files imported from other formats that do not contain information that a sample is circular, for example FASTA files, you can manually define a sequences as circular by checking this checkbox.

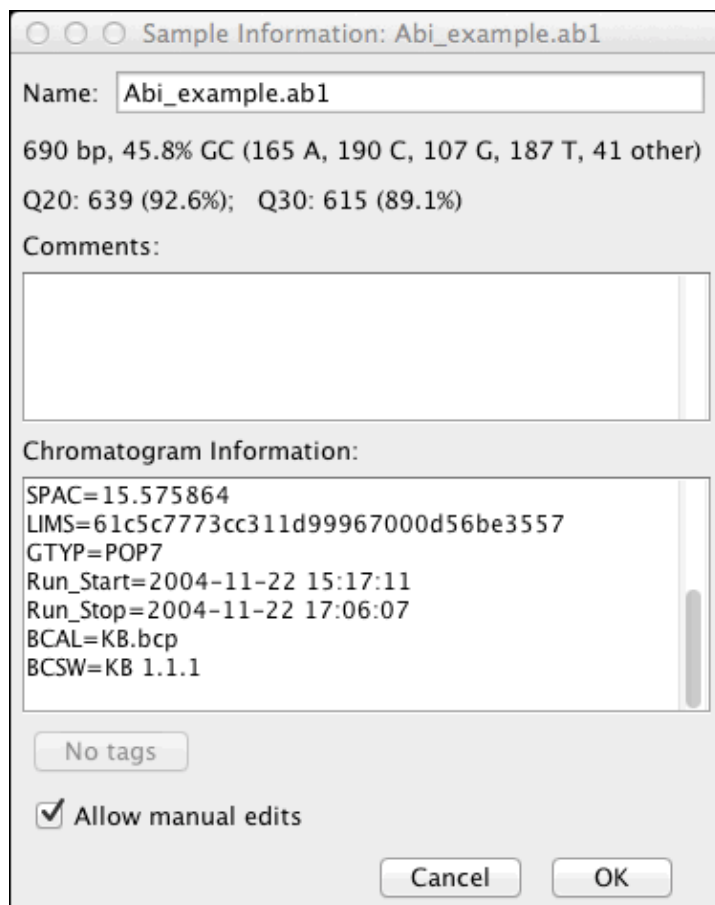
You can also designate a sample file as "read only" by unchecking the "**Allow Manual Edits**" checkbox.

## Changing Sample Names

You can **change the sample name** by editing the "Name:" field. Note that we do not recommend using spaces or other "funny" characters in the sample names, since such characters can cause problems when using these sample files in other programs.

## Viewing Chromatogram Information

For samples that have chromatograms, additional information about the chromatogram will be shown in the sample information dialog:



The image shows a 'Sample Information' dialog box for a file named 'Abi\_example.ab1'. The dialog has a title bar with three window control buttons and the text 'Sample Information: Abi\_example.ab1'. Inside, there is a 'Name:' label followed by a text field containing 'Abi\_example.ab1'. Below this, the sequence statistics are displayed: '690 bp, 45.8% GC (165 A, 190 C, 107 G, 187 T, 41 other)' and 'Q20: 639 (92.6%); Q30: 615 (89.1%)'. A 'Comments:' label is followed by a large, empty text area. Below the comments area is a section titled 'Chromatogram Information:' which contains a scrollable text area with the following data: 'SPAC=15.575864', 'LIMS=61c5c7773cc311d99967000d56be3557', 'GTYP=POP7', 'Run\_Start=2004-11-22 15:17:11', 'Run\_Stop=2004-11-22 17:06:07', 'BCAL=KB.bcp', and 'BCSW=KB 1.1.1'. At the bottom of the dialog, there is a 'No tags' button, a checked checkbox labeled 'Allow manual edits', and 'Cancel' and 'OK' buttons.

Sample Information: Abi\_example.ab1

Name: Abi\_example.ab1

690 bp, 45.8% GC (165 A, 190 C, 107 G, 187 T, 41 other)  
Q20: 639 (92.6%); Q30: 615 (89.1%)

Comments:

Chromatogram Information:

SPAC=15.575864  
LIMS=61c5c7773cc311d99967000d56be3557  
GTYP=POP7  
Run\_Start=2004-11-22 15:17:11  
Run\_Stop=2004-11-22 17:06:07  
BCAL=KB.bcp  
BCSW=KB 1.1.1

No tags

☒ Allow manual edits

Cancel OK

## CodonCode Aligner User Manual

The example above shows information extracted from an ABI file. Note that the last two lines show which base caller was used for this sample. In this case, the KB base caller, which assigns quality scores to each base, was used (the newer replacement of the old ABI base caller, which did not assign quality scores). For sequences like this one that have valid quality scores, a quality summary is given near the top, below the base composition line.

For chromatograms that were base called with PHRED, the last lines in the "Chromatogram Information" will still point to the KB or ABI base caller. But if you **scroll through the chromatogram information**, you can see entries added by PHRED:

Sample Information: Phred\_example

Name:

661 bp, 47.2% GC (166 A, 197 C, 115 G, 183 T)  
Q20: 629 (95.2%); Q30: 608 (92.0%)

Comments:

Chromatogram Information:

SIGN=A=500,C=719,G=278,T=699  
SPAC= 15.66  
PRIM=0  
MACH=Geenie2-16110-028  
DYEP=KB\_3730\_POP7\_BDTv3.mob  
TPSW=  
NAME=ABPS002-04F  
LANE= 2

☒ Allow manual edits

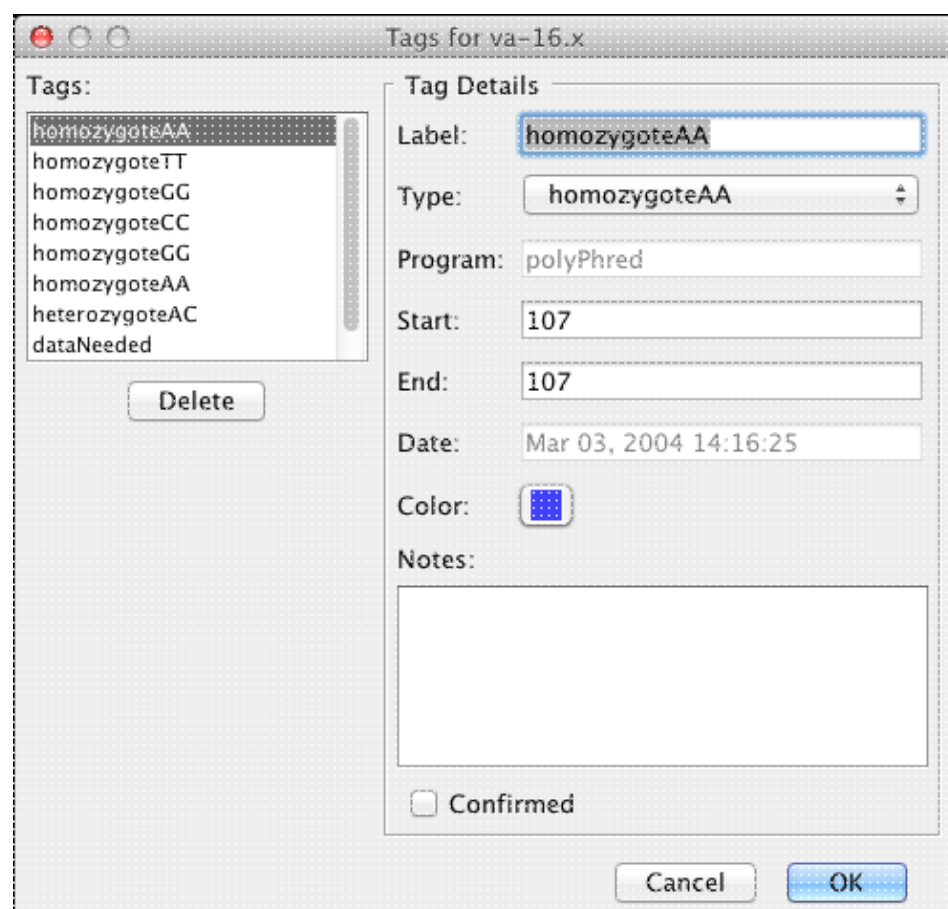


In the example above, you see an entry labeled "CONV" that point to PHRED, and list the version of PHRED that was used.

The chromatogram information is read from ABI files, or from the comment field in SCF files. You cannot edit this information.

## Viewing Sample Tags

Samples may also contain tags which have been added by users, other programs like PolyPhred, or CodonCode Aligner during steps like end clipping or vector trimming. If a sample contains tags, you can click on the **"Tags..."** button to display the tag dialog:



*(You can also see all tags for a sample by selecting the sample in the project view, going to the "**Sample**" menu, and selecting "**Show All Tags...**" from the "**Tags**" sub-menu).*

You can select a tag on the left to view details about the tag on the right. You can change the start and end position of tags, and add comments about the tag in the "Notes:" text field. You can also delete tags. Changes will be saved after you press the "OK" button.

### Confirming Tags

One common use of the tag "notes" is to mark tags as confirmed after looking at the trace data. You could do this by typing "Confirmed" in the note box, or simply by clicking on the "Confirmed" checkbox. Clicking on the checkbox when it is unchecked will add the word "Confirmed" at the beginning of the text; clicking it when it is checked will remove **all** occurrences of the word "confirmed" from the check box.

**Tip:** *If you find a tag that is wrong, you can either delete the tag, or you could mark it in the "Notes" section. We suggest that you do **not** write "Not confirmed", but rather use different words like "Disagree" or "Incorrect".*

The contents of the "Notes" field are shown in the "Content" column in the Feature View, and when [exporting features](#) to text files.

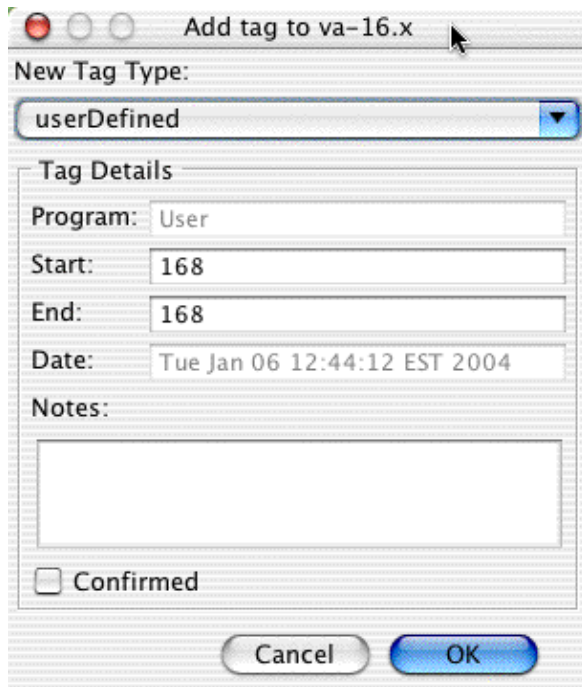
### Viewing "Local" Tags

You can also view **tags at specific locations** in a sample from trace view, base view, and contig view windows. Select a base that contains a tag, and then right-click (OS X: control-click) on it to display the popup menu. The first item in the popup menu will be "Display Tag..." (but only if you clicked on a base with a tag). This will bring up the tag dialog as above, except that only tags at this position in the sample will be shown.

*(Instead of using the popup menu, you can also go to the "**Sample**" menu, and select "**Show Local Tags...**" from the "**Tags**" sub-menu).*

### Adding Tags

You can also add your own tags to any base or range of bases in a sample. Open a view that shows the bases (a trace view, base view, or contig view), select the bases you want to tag, and then right-click (OS X: control-click) on it to display the popup menu. The first or second menu item in the popup menu will be "Add Tag...". Selecting it will bring up the following dialog:



The default type of the new tag is "userDefined". You can change this to any of the other pre-defined tag types using the pull down menu at the top. You can also edit the start and end position, and add comments about the tag in the "Notes:" text field. If you want a tag to extend to the start of a sequence, just enter "0" in the "Start:" field. If you want a tag to go to the end of the sample, enter a large number like "9999" in the "End" text field. Changes will be saved when you press the "OK" button.

*(Instead of using the popup menu to add tags, you can also go to the "**Sample**" menu, and select "**Add Tag...**" from the "**Tags**" sub-menu).*

## Adding Tags to All Sequences

Tags can also be added to all sequences in a contig at the same time. Open the contig view that shows the contig, select the bases you want to tag in the consensus sequence, and then right-click (OS X: control-click) on it to display the popup menu. The second menu item in the popup menu will be "Add Tag to All...". Selecting it will bring up the same dialog as when adding a tag to one sample.

*(Instead of using the popup menu to add tags to all sequences at the same position, you can also go to the "**Sample**" menu, and select "**Add Tag to All...**" from the "**Tags**" sub-menu).*

## Importing Annotation (Tags)

To add annotation created by other programs to samples in CodonCode Aligner, you can import annotation files in GFF format. Annotation will be added as tags to samples in the project where the names match the names in the GFF format. When no samples can be found in the project that match the name(s) in the GFF file, and the GFF file contains sequence information in GFF format, then new samples will be created from the sequences in the GFF files, and tags for the annotation will be added to these new samples.

Please note that sample names with spaces or special characters may cause problems when importing annotation, since some programs may not support such names, and special character may be lost or changed,

or sample names may be truncated.

# Saving Edits

Edits to the consensus sequence or sample sequences are not saved until you save the project by selecting "**Save Project**" in the "**File**" menu, or using the corresponding toolbar buttons or keyboard shortcut (Command\_S on OS X, Control-S on Windows).

If you are experimenting and think you might want to go back to a previous state that your project was in, you should save the project under a new name by going to the "**File**" menu and selecting "**Save Project As...**". This will save all data for your current project to a new location.

You can change the name of individual samples, which will then save them under a different name the next time you save the project, by selecting the sample and choosing "**Sample Information**" from the "**Sample**" menu, and editing the sample name. For more information, please read the "[Sample Information Dialog](#)" help page.

# Undo and Redo

In the current version of Aligner, only limited support for undoing and re-doing is provided. Most actions, for example adding samples, assembling, unassembling, endclipping, and vector screening, cannot be undone. Therefore, please **save your project frequently!**

You can, however, undo simple edits. Just choose "**Undo**" from the "**Edit**" menu to undo an edit, and restore the sequence to the state it was before the edit. If you change your mind, you can choose "**Redo**" from the "**Edit**" menu to re-do the edit that you just undid. If you did anything else that is not undoable, "**Undo**" from the "**Edit**" menu will not be available (grayed out).

We plan to add Undo and Redo support for more actions in future Aligner releases. If undo support for a particular feature is important to you, please let us know about it!

# Copy and Paste

## Copy (selected sequence)

If your current selection is a single sample or contig, or part of a single sample, you can copy sequences and parts of sequences by selecting "**Copy (selected sequence)**" in the "**Edit**" menu.

If the active window is a **trace view**, **base view**, or **contig view**, then the currently selected bases will be copied to the clip board.

If the currently active window is the **project view**, and a single **sample** is selected, then the entire sequence of this sample will be copied to the clip board. If you have selected a single **contig** in the project view, then the consensus sequence for this contig will be copied. If you have more than one sample or contig selected, copy will not be available.

## Paste

The "**Paste**" menu item in the "**Edit**" menu is disabled, since Aligner is not intended to be used as a text editor. You can, however, paste any sequence that you copied (see above) into programs outside of Aligner (for example BLAST web pages).

Please note that **any gaps in sequences will be removed** in the copied sequence, since most programs or web pages you might paste the sequence into will expect ungapped sequences. If you need to copy and paste gapped sequences, please let us know!

There are also some dialogs that support paste through the keyboard shortcuts (control-V on Windows, command-V on OS X). The most notable is the "**Search Sequence...**" dialog, accessible from the "**Go**" menu.

# Editing Contigs

CodonCode Aligner lets you manipulate contigs in a number of different ways. You can edit the samples in a contig, as described in the previous section. Several other common tasks with contigs are:

- [Adding new samples to existing contigs](#)
- [Merging contigs](#)
- [Removing samples from contigs](#)
- [Reverse-complementing contigs](#)
- [Splitting contigs](#)
- [Unassembling contigs](#)
- [Editing contig information](#)



# Adding Samples to Contigs

To add new samples to an existing contig, you need to first add the samples to the project, and do any pre-processing steps like base calling or end clipping that you want to do. Then, to add the sample to an existing contig:

1. Go to the project view
2. Select both the contig and the samples that you want to add to it  
(to make continuous selections, use *shift-click*; to make discontinuous selections, use *control-click* on Windows and *command-click* on OS X)
3. Go to the "**Contig**" menu
4. Choose "**Assemble**" (or, if your contig is an alignment to a reference sequence, choose "**Align To Reference Sequence**")

Aligner will try to merge the samples with the contig. If samples share an overlap that meets the minimum criteria you defined, they will be added to the contig. Samples that do not overlap the contig, or where the overlap is not good enough, will remain in the Unassembled Samples folder.

You can also choose more than one contig and one or more samples - for example two neighboring contigs, and some finishing reads that bridge the gap between the contigs.

Alternatively, you can use **drag and drop** in the project view to add samples to contigs. First, select the samples and/or contigs you want to add in the project view, and then drag and drop them onto the contig to which you want to add them (this "target" contig cannot be in the initial selection). Once you release the mouse, CodonCode Aligner will start the assembly of the samples and contig(s).

When you use the "**Assemble**" or "**Align To Reference Sequence**" menu with existing contigs, the arrangement of the samples in the contig will remain the same; changes are limited to any gaps that need to be introduced, re-building of the consensus sequence, and possibly changing the extend of unaligned regions at the end of samples. In addition, the contig may be reverse-complemented during assembly.

If you do **not** care about keeping the current arrangement of the samples in a contig, you can choose "**Assemble from Scratch**" or "**Align To Reference from Scratch**" instead. This will first unassemble the existing contigs, and then re-assemble or re-align all reads that were in the contig(s), plus any other reads you have selected.

## Duplicating samples

You can duplicate samples in a project, and create text sequences from consensus sequences, as follows:

1. Select the sample(s) or contig(s) you want to duplicate.
2. Go to the "**File**" menu.
3. Select "**Duplicate**".

This will create a copy of every selected sample, and a new text sequence for every selected contig.

Before using this option to align consensus sequences, however, please look at the help for "[Compare contigs](#)" - in CodonCode Aligner, you can compare contigs directly to each other, and keep the relation to the sequences and their traces in a contig intact!

# Merging Contigs

To merge two or more contigs:

1. Go to the project view
2. Select the contigs you want to merge (you can also select reads in "Unassembled Samples")
3. Choose "**Assemble**" from the "**Contig**" menu.

Aligner will look for overlaps between the contigs, and merge the contigs if the overlap meets the minimum assembly criteria. If the overlap between the contigs is not good enough, for example because it is very short or has a high discrepancy rate, Aligner will reject the merger, and leave the contigs as they were.

When you use "**Assemble**" with existing contigs, Aligner will compare the consensus sequences of the two contigs to look for an overlap. The relative arrangement of the samples in each contig will remain unchanged. Changes are limited to any gaps that need to be introduced, reverse-complementing (if needed), re-building of the consensus sequence, and possibly changing the extend of unaligned regions at the end of samples.

There are two alternatives to using the "**Assemble**" menu item: using "**Assemble with Phrap**" and "**Assemble from Scratch**". Both options will first unassemble the existing contigs, and then assemble the existing reads, using either Phrap or CodonCode Aligner's build-in assembly algorithm to build the contigs (for-profit users need to have a separate license for Phrap). The potential drawback of both of these options is that any edits made by moving gaps or samples around will be lost (but other edits, like change base calls or removed ends, will of course remain).

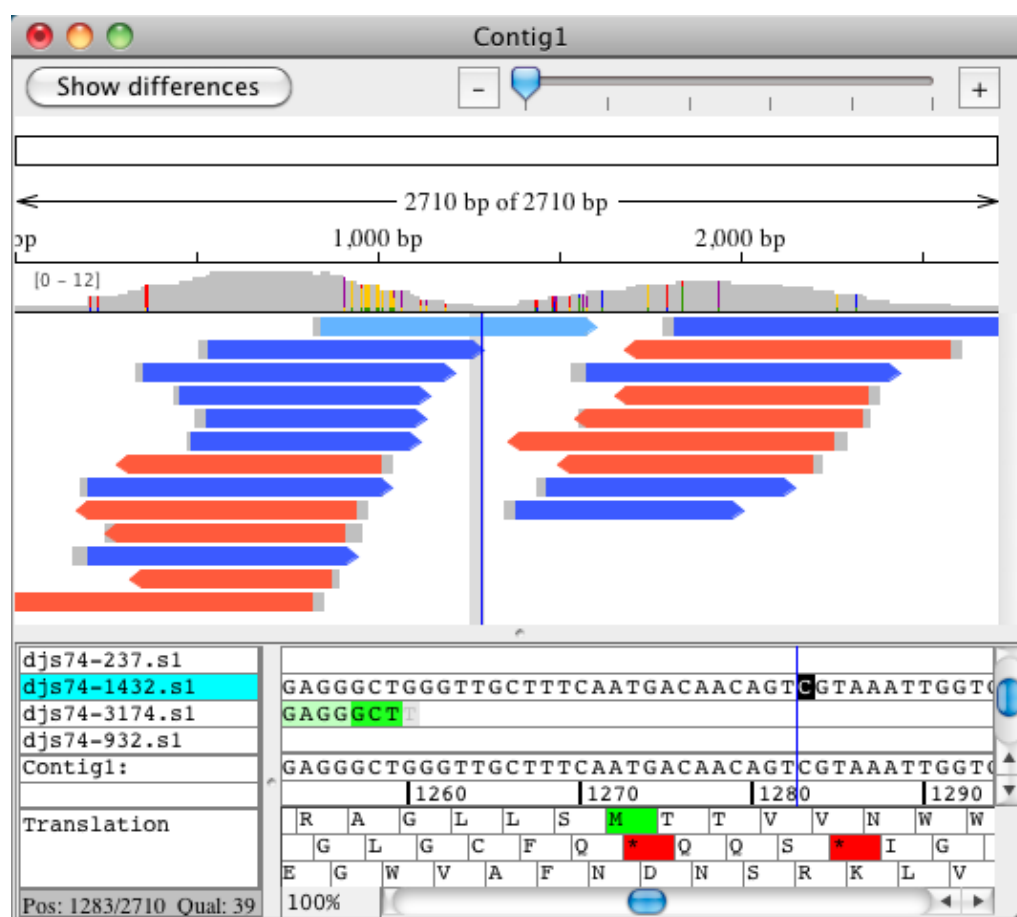
# Removing Samples from Contigs

Occasionally, Aligner may put a sample into a contig where it does not belong, or you may want to remove a read from a contig for other reasons.

To remove a sample from a contig, select the sample in the contig view by clicking on its name or a base in it. Then, choose either **"Move to Unassembled Samples"** or **"Move to Trash"** from the **"Edit"** menu.

Before removing the sample from the contig, Aligner first checks if removing the sample would introduce any gaps in the contig. This happens if there are reads to both sides of the sample, and the sample is the only sample that covers one or more bases of the consensus sequence. In this case, removing the sample would in effect split the contig into two pieces, and (at least currently) Aligner will refuse to do so. If splitting the contig is what you wanted to do, you will first have to [split the contig](#) yourself, and then remove the sample from the new contigs formed after splitting.

An example where removing a read would split a contig into two is shown below:



Here, removing the selected sample (djs74-1432.s1) would split the contig into two parts. However, the other samples could be removed without problems, since they only cover regions that are also covered by other reads.

# Deleting Parts of Contigs

To delete all bases from the current cursor position to the start of a contig:

1. Select a contig base in the contig view
2. Go to the "**Contig**" menu
3. Go to the "**Delete**" submenu
4. Select "**From Contig Start**" to delete all bases to the start of the contig,  
or select "**To Contig End**" to delete all bases to the end of the contig

If any samples are completely within the removed contig region, these samples will be moved to the trash.

If the menu items are disabled, you probably do not have a contig base selected. You may have clicked on a sample accidentally, or perhaps a view other than the contig view is in the foreground.

***Note:** Deleting from the contig start or to the contig end **cannot be undone**, so you may want to save your project first, or to save a copy of your project using "**Save As**" in the "**File**" menu.*

# Removing Consensus Gaps

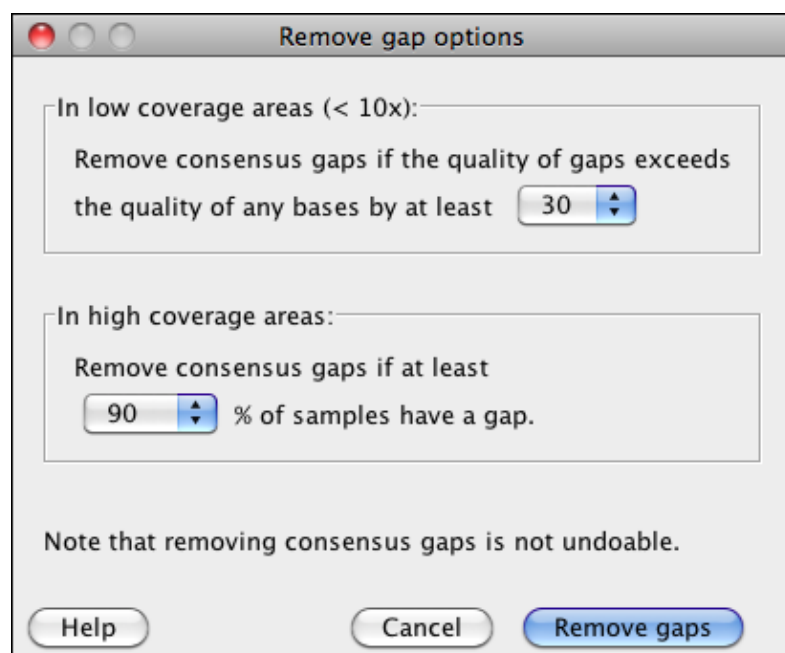
Gaps in the consensus sequence are caused by insertions in reads, which are often due to random sequencing errors. Often, consensus gaps can be simply ignored, but at other times, it is desirable to remove consensus gaps. Manual removal of consensus gaps is always possible (select the gap in the consensus sequence, and press the "delete" key), but can be impractical for larger projects.

CodonCode Aligner provides a functions to automatically remove most consensus gaps by editing all aligned samples at consensus gap positions. Since this will remove bases in the samples that caused the gap, there are safeguards based on coverage and sequence quality, as explained below.

## How to remove consensus gaps

To remove consensus gaps:

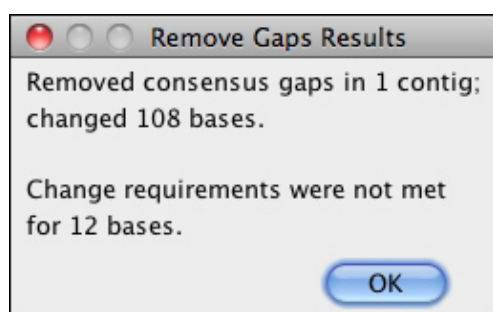
1. Select a contig in the project view, or a consensus region in the contig view.
2. Choose "**Remove Consensus Gaps**" from the "**Contig**" menu. This will show the following dialog:



The dialog lets you set the safeguard thresholds.

- In **low-coverage areas** (less than 10 x coverage), CodonCode Aligner will remove consensus gaps only if the quality of any basis in samples at a given consensus position is lower than the quality of the highest-quality gaps at this position; the required difference can be set to values between 10 and 99 (default: 30). *Note that choosing a value of 99 means that no gaps will be removed in low-coverage areas.*
- In **high-coverage areas** (10x or higher coverage), the decision whether a consensus gap should be removed will be based on how many of the aligned samples have a gap at this position. The value can be set to between 50% and 100% (default: 90%).

After setting the thresholds, click on the "Remove gaps" button to remove consensus gaps in the selected contig(s) or contig region. CodonCode Aligner will perform the necessary edits, and show a dialog summarizing the results:



Note that removing consensus gaps is not undoable! We suggest that you save a copy of your project first.

# Alignment Locations - Start and End

During contig building by assembly or alignment, Aligner automatically determines the useful parts of reads. For each sample, regions at the start and end with high levels of discrepancies to other reads will remain unaligned. These regions are typically shown dimmed in the contig view. Bases in the unaligned regions are not considered when determining the consensus sequence. Unaligned regions are not aligned, meaning that Aligner will not introduce gaps in unaligned regions.

You can change the start and end of sample alignments through two menu items in the "**Sample**" menu: "**Mark Start Alignment Location**" and "**Mark End Alignment Location**". If you use one of these options to extend the aligned region of a sample, you will have to introduce any gaps required for proper alignment by hand.

If you import Phrap assemblies, please note that unaligned regions of samples may extend beyond the ends of a contig on both sides (the start and the end).



# Reverse Complementing

To reverse complement a sample or contig:

- Go to the project view
- Select the contig (or a sample in the "Unassembled Samples" folder)
- Choose "**Reverse Complement**" from the "**Edit**" menu

In most views, you will be able to identify samples that have been reverse-complemented by the "<<" prefix before the name, and by the fact that the name is drawn in red. In the project view, you can identify reverse-complemented samples by the icon (it has red rather than black borders).

Reverse-complementing of unassembled samples is possible, but usually not necessary, since samples will be reverse-complemented as needed during assembly and alignment.

You cannot reverse-complement samples in contigs directly, since that would destroy the alignment to the consensus. If you select "**Reverse Complement**" when a contig view is in the foreground, the entire contig will be reverse-complemented.

# Splitting Contigs

Like other assemblers that use a "greedy" assembly algorithm, CodonCode Aligner will occasionally misassemble contigs. You can manually split misassembled contigs into two parts as follows:

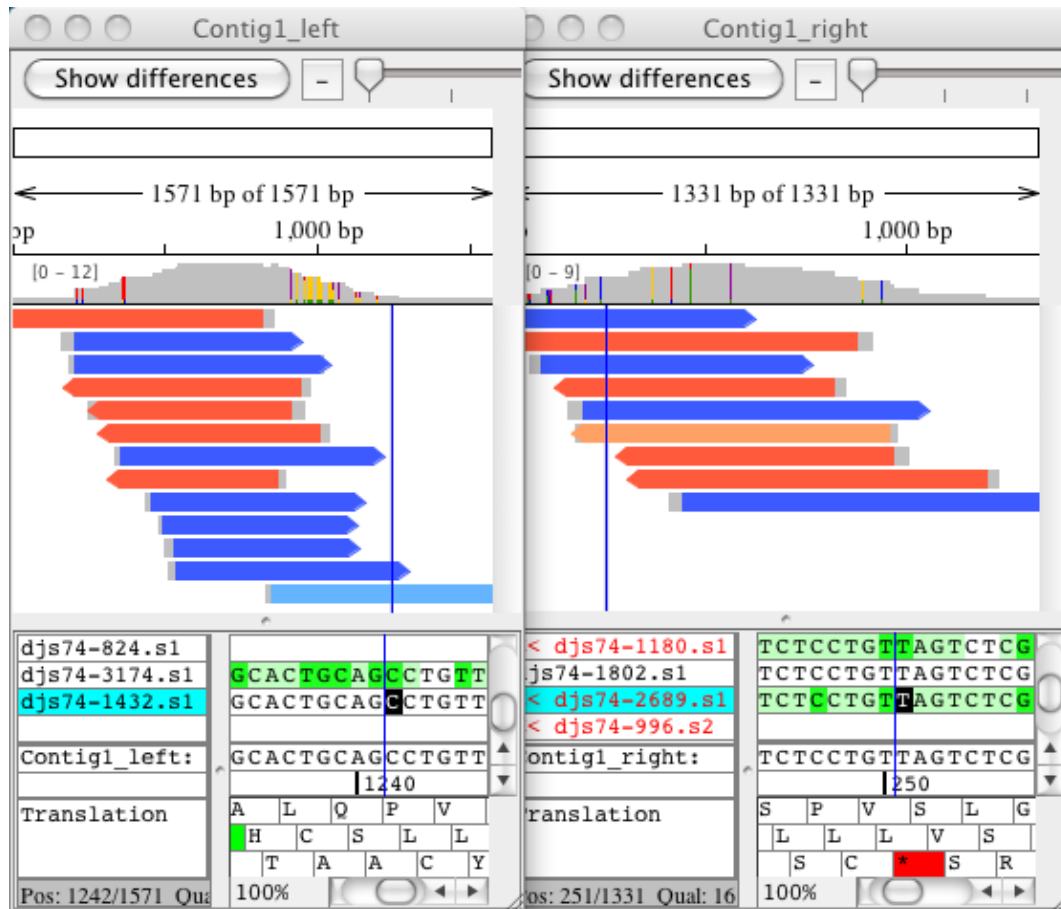
- Open the contig view for the contig in question
- Set the cursor to the position where you want to split up the contig
- Select **"Split contig"** from the **"Contig"** menu

Aligner will create two new contigs - one containing the samples that were to the left of the point where you split the contig, and one containing the samples that were to the right. For any samples that had aligned bases at the position of the split, Aligner will put this either into the new right or the new left contig; the right contig will be chosen for samples that had more aligned bases to the right of the split point, and the left contig for samples that had more aligned bases to the left of it.

Here's an example of a contig before splitting:



After splitting at the position indicated (1283), the two new contigs look like this:



Note that the sample djs74-1432.s1 ended up in the leftt contig.

# Unassembling Contigs

To unassemble one or more contigs:

- Go to the project view
- Select the contig(s)
- Choose "**Unassemble**" from the "**Contig**" menu

All samples in the contig will be put into the "Unassembled Samples" folder, and any gaps in the samples will be removed. Any samples that were reverse-complemented in the contigs will be returned to the original (not reverse-complemented) state.

# Roundtrip Editing

While CodonCode Aligner supports manual editing of contigs, Aligner does not provide all the functionality of programs that were specifically developed to edit sequence alignments. However, CodonCode Aligner supports "Roundtrip Editing" - exporting existing contigs for editing in external programs (like MacClade or Mesquite), and re-importing the manually edited contigs. The re-imported contigs maintain the connection to the original chromatograms, so you can quickly check any discrepancies by looking at the original sequence traces.

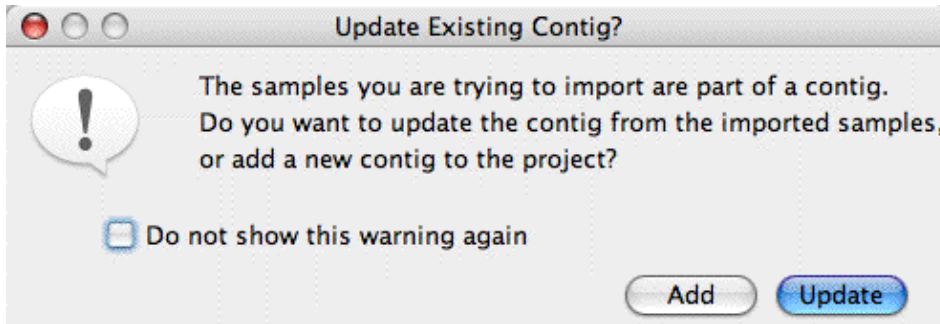
A typical example where roundtrip editing can make sense is a phylogenetic project, where you go through the following steps:

- Create a project in CodonCode Aligner with sequence traces from a number of different species (isolates, ...)
- Create separate contigs for each species (for example using "[Assemble in Groups](#)")
- Edit the initial contigs
- Create an alignment of the contig sequences (a "contig of contigs") with Clustal W from within CodonCode Aligner (using "[Compare Contigs...](#)" in "[Advanced Assembly](#)")
- Export the contig of contigs for editing in MacClade (or similar)
- Use MacClade to move gaps around
- Export the edited assembly from MacClade
- Re-import the edited assembly into CodonCode Aligner, thereby updating the position of gaps in the contig of contigs
- Verify and edit discrepancies between the species in CodonCode Aligner; you can quickly go back to the original sequence traces by double-clicking in the contig view for the contig of contigs
- Export the final assembly in PIR format (or similar) for further analysis

## How To Roundtrip Edit with CodonCode Aligner

Here's how to roundtrip edit in CodonCode Aligner. The starting point assumes you have a project where you already have a contig (or contig of contigs) that you would like to edit with an external editor like MacClade.

1. **Check names** - before exporting, check the names of the contigs you want to export. Many export formats and external editors restrict how long names can be, and which characters can be used in names. To be safe, use short names with only numbers and letters. Names of 10 or fewer characters are safest, although up to 30 characters may also work.
2. Select the contig(s) to export in the project view.
3. Go to the **File** menu, and select **Export > Assembly**. From the format pulldown menu, select a format that works for the program you want to use, for example "Interleaved NEXUS/PAUP", and then export the file.
4. Import the file you just created in your favorite sequence editor, for example MacClade.
5. Edit the alignment, but limit your edits to moving gaps.
6. Export the edited alignment in NBRF/PIR format (with gaps).
7. Open the NBRF/PIR file in CodonCode Aligner, using drag and drop or **File > Open Sample....** You should see the following dialog:



Click on "Update". Aligner will import the sequences, and update the existing contig to reflect your edits.

## Limitations

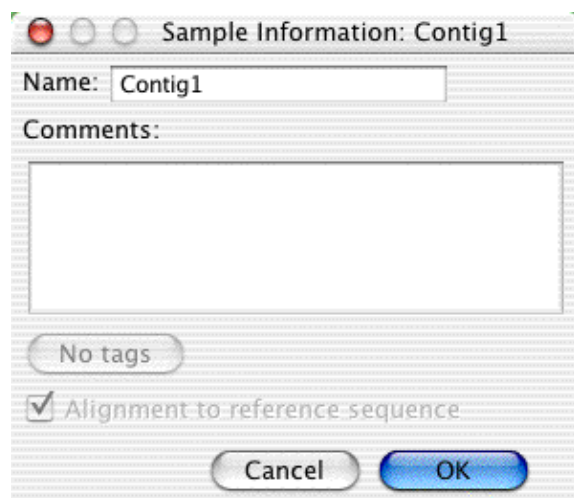
Currently, this "Roundtrip Editing" functionality is limited. You can move gaps around in external editing programs, but you cannot change or delete any bases. Here is a list of additional restrictions:

- Updating existing contigs will work only when importing files in NBRF/PIR format
- All sequences in the file must be exactly the same length, including leading and trailing gaps (*note that some programs (e.g. BioEdit) may not always make sequences of equal length when exporting in NBRF/PIR format*)
- The names of the samples in the imported file must be identical to (or sufficiently similar to) the names already in the project. *Note that some programs do not allow spaces, dashes, or other "funny" characters in sequence names, and that many programs and file formats restrict the length of sequence names.*
- The only edits currently supported are gap movements; if you edit, insert, or delete bases, the updating will fail.
- If the exported contig had unaligned (dangling) ends, the updating will likely fail. *One way to create dangling ends is to generate contigs of contigs using the built-in algorithm, and then remove samples or contigs at the beginning or end of the contig. This cannot happen if the contig of contigs is generated with ClustalW (since ClustalW creates end-to-end alignments).*

We plan to remove some of these restrictions in future releases.

# Editing Contig Information

You can rename contigs, and add comments, by selecting the contig in the project view, and then choosing "Contig Information" in the "Contig" menu. This will bring up the following dialog box:



You can change the name in the "Name:" textfield at the top. You can also add remarks about the contig in the larger text area below; these will be shown in the "Comments" column in the project view.

If a contig has tags (for example "polymorphism" tags added from finding mutations), the button that is labeled "No tags" in the image above will be labeled "Tags...", and become active. Clicking on it will bring up a tag dialog that shows all tags for the consensus sequence.

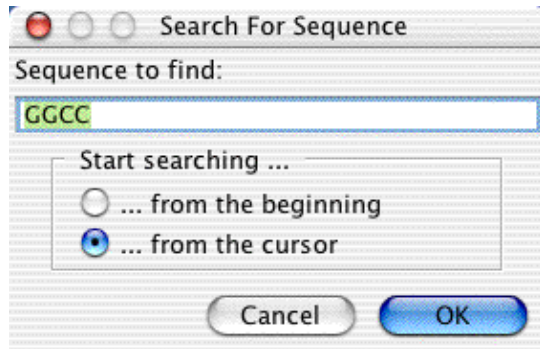
The checkbox at the bottom indicates whether a contig is an alignment to a reference sequence (as in the image above), or a regular assembly.

*Please note that contig names and remarks, as well as tags added to a consensus sequence, will change or get lost when you later unassemble the contig or merge it with other contigs or unassembled reads.*



# Search for Sequences

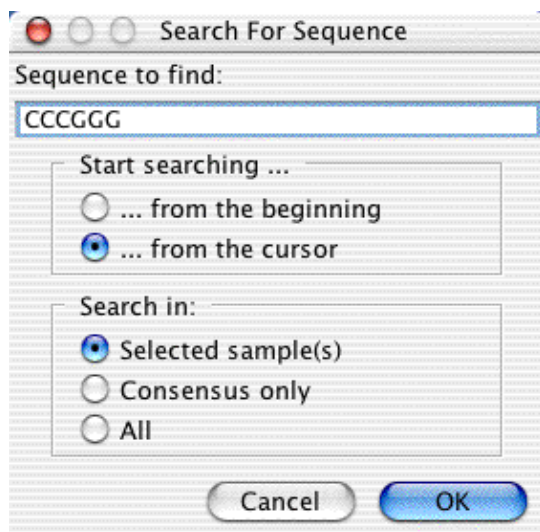
Aligner allows you to look for short sequences within your samples and contigs. First, select the sample or contig that you want to look in, and open a base view, trace view, or contig view for it. Then, selected **"Search Sequence..."** from the **"Go"** menu. This will display the following dialog:



You type the sequence to find in the text box at the top, and then press the "OK" button. The sequence you can look for may contain the four bases A,G,C, and T, as well as ambiguity codes; gaps are ignored. If you search for ambiguity codes, note that you will get a match only if the ambiguity code is found (for example, 'AGTN' will match 'AGTN', but not 'AGTC').

If Aligner finds your search string, it will position the cursor to the first occurrence; if Aligner cannot find the sequence, it will tell you so.

If you started the search from a contig window, the dialog will be slightly different:



## CodonCode Aligner User Manual

When searching in contigs, you can specify where you want to look for your search sequence - only in the currently selected sample(s), only in the consensus, or in all sequences and the consensus.

**Note:** You can repeat any search by choosing **Search Again** from the **Go** menu.

# BLAST Searches

To start a BLAST search from CodonCode Aligner:

- Select the sample(s) or contig that you want to BLAST in the project view, **or**: select the bases you want to search in a contig view, trace view, or base view
- Go to the "Go" menu, and select one of the options from the "**BLAST Search**" submenu.

Aligner will open web browser page for the NCBI BLAST server, and paste the selected sequence into the "Search" field. You can now change the search options or database to search against on this web page, and start the BLAST search. (If your security settings are very strict so that Aligner is not allowed to open a browser page, nothing may happen - you will need to change the security settings first).

For all BLAST searches, your selection can include more than one sequence. However, the total length is limited to about 8 kb. For BLAST searches with sequences longer than 8 kb, you can export the sequence(s) in FASTA format, and load the exported file on the [BLAST search web page](#).

## MegaBLAST

[MegaBLAST](#) is a BLAST version that has been optimized for aligning sequences that differ only slightly, for example because of sequencing errors. It can be up to 10 times faster than other BLAST versions.

## Nucleotide (blastn)

The "Nucleotide (blastn)" option can be used to initiate a comparison of nucleotide sequences against nucleotide databases.

## Translated (blastx)

The "Translated (blastx)" option can be used to initiate a comparison of a protein translation of your nucleotide sequences against protein databases.

## Translated (tblastx)

The "Translated (tblastx)" option can be used to initiate a comparison of a protein translation of your nucleotide sequences against a protein translation of the nucleotide databases.

For more information about BLAST, please visit <https://blast.ncbi.nlm.nih.gov/Blast.cgi/>.

### Using other BLAST servers

By default, CodonCode Aligner uses the BLAST servers at the NCBI (National Center for Biotechnology Information). However, it is possible to use other BLAST servers instead, for example private servers on your local network, by adding an entry for BLAST\_URL in the CodonCode Aligner preferences file. Please contact CodonCode Corporation if you are interested in this and need additional information.

# Exporting

Sooner or later, you will probably want to use the data you see with Aligner with other programs. For example, you may want to use BLAST to compare the consensus sequences to nucleotide databases, or you may want to export read names and reads lengths for importing into spreadsheets or databases.

You can export the key data in Aligner projects using the various sub-menus in the "**Export**" menu in the "**File**" menu. To export samples or consensus sequences, first select what you want to export in the project view. Then, choose one of the following export options:

- [Export Project Summary...](#)
- [Export Samples...](#)
- [Export Consensus Sequences...](#)
- [Export Assembly...](#)
- [Export Aligner Project \(Old Format\)...](#)
- [Export Features...](#)
- [Export Protein Translation...](#)
- [Export Differences...](#)
- [Export Trees...](#)

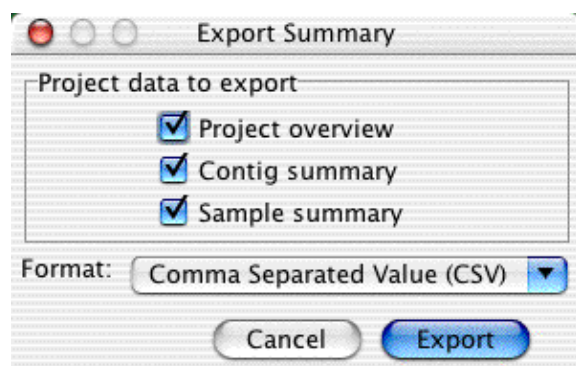
Aligner will then show a dialog box, where you can set options like file formats. Next, Aligner will present a standard "Save As" dialog where you can choose the folder for your exported files and (if you are exporting to single files) the file names.

If you are exporting multiple items to individual files, Aligner will automatically determine the file names and extensions. If any files with the same name already exist in the folder where you want to save your files, Aligner will warn you that the existing files will be replaced.

# Export Project Summary

This allows you to export project data like sample names, read lengths, and so on to text files that can be imported into external spreadsheet programs or databases. The information exported will be the information you see in the project view window, and (optionally) summary information about your project.

To export project summary information, go to the **"File"** menu, and select **"Project Summary..."** from the **"Export"** sub-menu. This will show the following dialog:



When you press the "Export" button, a standard "Save" dialog appears, where you can choose the name and location of the exported file. The file is a text file, and can be imported into word processors or spread sheets like Microsoft Excel (*you may need to select "All Files" in the "Open" dialog to be able to import the file*).

Here is an example of what the exported data look like after importing into Microsoft Excel:

| PolyPhred_summary.csv |                         |   |               |         |           |                      |
|-----------------------|-------------------------|---|---------------|---------|-----------|----------------------|
|                       | A                       | B   | C             | D       | E         | F                    |
| 1                     | Project Name            | PolyPhred.proj                              |               |         |           |                      |
| 2                     | Date and Time of export | 11/21/03 10:01                              |               |         |           |                      |
| 3                     | Project location        | /Users/Shared/Documents/Projects/PolyPhred/ |               |         |           |                      |
| 4                     | Contig count            | 2   |               |         |           |                      |
| 5                     | Sample count            | 10  |               |         |           |                      |
| 6                     | Total sample bases      | 9018  |               |         |           |                      |
| 7                     | Average sample length   | 901   |               |         |           |                      |
| 8                     | Average sample quality  | 387   |               |         |           |                      |
| 9                     | Average contig length   | 597   |               |         |           |                      |
| 10                    | Average contig quality  | 0   |               |         |           |                      |
| 11                    |                         |   |               |         |           |                      |
| 12                    | Contig Name             | Number of Samples                           | Contig Length | Quality | Coverage  | Estimated Error Rate |
| 13                    | Unassembled Samples     | 0   | 0             | 0       | 0         | 0.00E+00             |
| 14                    | Contig1                 | 4   | 473           | 0       | 3.9       | 2.20E-02             |
| 15                    | Contig2                 | 6   | 722           | 0       | 4         | 3.55E-01             |
| 16                    | Trash                   | 0   | 0             | 0       | 0         | 0.00E+00             |
| 17                    |                         |   |               |         |           |                      |
| 18                    | Sample Name             | Sample Location                             | Sample Length | Quality | Direction | Contig Offset        |
| 19                    | va-1.x                  | Contig1                                     | 1058          | 372     | Fwd       | 0                    |
| 20                    | va-13.x                 | Contig1                                     | 1020          | 406     | Fwd       | 0                    |
| 21                    | va-23.x                 | Contig1                                     | 1240          | 354     | Fwd       | -1                   |
| 22                    | va-16.x                 | Contig1                                     | 1025          | 382     | Fwd       | 1                    |
| 23                    | ca-9.r                  | Contig2                                     | 869           | 415     | Rev       | -396                 |
| 24                    | ca-22.r                 | Contig2                                     | 860           | 400     | Rev       | -385                 |
| 25                    | ca-22.s                 | Contig2                                     | 765           | 305     | Fwd       | 18                   |
| 26                    | ca-9.s                  | Contig2                                     | 742           | 394     | Fwd       | 19                   |
| 27                    | ca-21.s                 | Contig2                                     | 745           | 406     | Fwd       | 18                   |
| 28                    | ca-23.s                 | Contig2                                     | 694           | 437     | Fwd       | 21                   |
| 29                    |                         |   |               |         |           |                      |

PolyPhred\_summary.csv

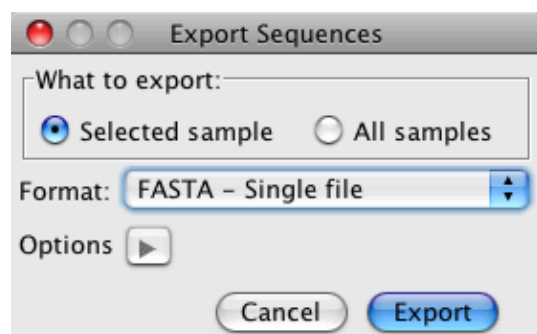
Ready Sum=0

# Exporting Samples

You can export samples to text files in FASTA format, or to chromatogram files in SCF ("Standard Chromatogram Format") format, as follows:

- Go to the project view
- Select the samples you want to export (*use shift-click to make continuous selections, and control-click (OS X: command-click) to make discontinuous selections*)
- Choose **"Export => Samples"** in the **"File"** menu.

This will display the following dialog:



Using the radio buttons at the top, select to export just the selected samples, or all samples in the project. The Format pull-down menu gives you the following format choices:

- **Single FASTA file:** This will generate a single text file in FASTA format which contains all the exported sequences. You can specify the name and location of the file in a "Save As" dialog box that will be shown when you click the "Export" button.
- **Individual FASTA files:** This option allows you to create a separate file for each sample. Again, this is a text file in FASTA format; each file contains exactly one sequence. You can choose the folder where the files are created in a "Save As" dialog that will be shown once you click on the "Export" button. All files will be created in the same folder; the names of the files will be the sample name, with ".fasta" appended.
- **Genbank files:** Files in Genbank format contain the current bases and any annotation (tags). You can choose the folder where the files are created in a "Save As" dialog that will be shown once you click on the "Export" button. All files will be created in the same folder; the names of the files will be the sample name.
- **EMBL files:** Files in EMBL format contain the current bases and any annotation (tags). You can choose the folder where the files are created in a "Save As" dialog that will be shown once you click on the "Export" button. All files will be created in the same folder; the names of the files will be the sample name.
- **SCF files:** SCF files contain the current base calls as well as the chromatogram data, and allow you to import the sequences into programs that support the standard SCF format. You can choose the folder where the files are created in a "Save As" dialog that will be shown once you click on the "Export" button. All files will be created in the same folder; the names of the files will be the sample name.
- **Single FASTQ file:** This will generate a single text file in FASTQ format which contains all the exported sequences and their quality scores. You can specify the name and location of the file in a "Save As" dialog box that will be shown when you click the "Export" button.

## Export options for FASTA files

If you choose to export a single FASTA file or individual FASTA files, you can set several options. To see these options, press the Options expand button (the triangle next to "Options").

The dialog will then look like this:



The first option allows you to automatically **replace "problem" characters** in your sample names. If you check the box "Replace problem characters in names", potentially problematic characters such as spaces or colons (':') will be replaced by underscores ('\_'). If the box is not checked, the sample names are exported as they are. In this case, importing the file into other programs might be problematic if samples contain "problem" characters.

The second option allows you to **append the sample comments** to the header of the FASTA file. The comments from the sample information of each sample in CodonCode Aligner will be appended to the first line of the FASTA file.

By checking the box labeled "**Include gaps in FASTA files**", you can include gap characters in the output. If the box is not checked, the exported sequences will be ungapped.

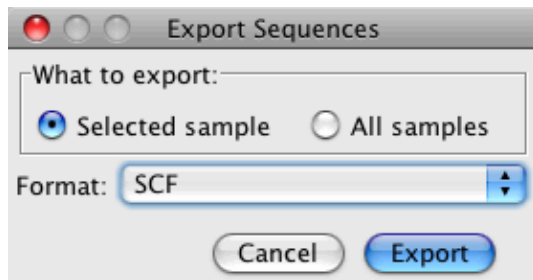
If you check the box "**Write FASTA quality files**", a quality file in FASTA format will also be created. If you are exporting a single FASTA file, this file will be in the same folder as the FASTA file. The name of the quality file will be the name of the FASTA file, with ".qual" appended at the end of the name. If you are exporting individual FASTA files for each sample, a separate quality file in FASTA format will be created for each sample.

When the "**Format sequences**" box is checked, longer sequences will be split up into multiple lines, with 50 bases per line. This can be a problem for some older software, which expects the sequence to be in a single line; to write the entire sequence into a single line in the FASTA file, simply uncheck the "**Format sequences**" box.



## Exporting SCF files

If you select SCF files as export format, the export dialog will look like this:



Note that there are no export options for SCF files. A separate file will be created for each sample that has traces.

# Exporting Consensus Sequences

To export the consensus sequences of contigs, select the contigs of interest in the project view, and then choose "Export Consensus Sequences..." from the "File" menu. This will bring up the following dialog:



Using the radio buttons at the top, select to export the consensus sequences for just the selected contigs, or all contigs in the project. The "Format" pull-down menu gives you the following format choices:

- **Single FASTA file:** This will generate a single text file in FASTA format which contains all the exported consensus sequences. You can specify the name and location of the file in a "Save As" dialog box that will be shown when you click the "Export" button.
- **Individual FASTA files:** This option allows you to create a separate file for each consensus sequence. Again, this is a text file in FASTA format; each file contains exactly one sequence. You can choose the folder where the files are created in a "Save As" dialog that will be shown once you click on the "Export" button. All files will be created in the same folder; the names of the files will be the sample name, with ".fasta" appended. Note that this option is only available if you export all consensus sequences.
- **Genbank files:** Files in Genbank format contain the current bases and any annotation (tags). You can choose the folder where the files are created in a "Save As" dialog that will be shown once you click on the "Export" button. All files will be created in the same folder; the names of the files will be the sample name.
- **EMBL files:** Files in EMBL format contain the current bases and any annotation (tags). You can choose the folder where the files are created in a "Save As" dialog that will be shown once you click on the "Export" button. All files will be created in the same folder; the names of the files will be the sample name.
- **Haplotypes as FASTA files:** This will export the consensus sequences as unphased pseudo-haplotypes to a single file. The file will contain two entries for each consensus sequence. These two entries will be identical *except* at positions where a consensus sequence contains a 2-base ambiguity code. For example, if the the consensus sequence contains a 'Y', one of the two entries will have a 'C' at this position, and the other entry will have a 'T'.
- **FASTQ files:** This will generate a single text file in FASTQ format which contains all the exported consensus sequences and their qualities. You can specify the name and location of the file in a "Save As" dialog box that will be shown when you click the "Export" button.

## Export options for FASTA files

If you choose to export as FASTA files, you can set several options. To see these options, press the Options expand button (the triangle next to "Options").

The dialog will then look like this:



The first option allows you to automatically **replace "problem" characters** in your contig names. If you check the box "Replace problem characters in names", potentially problematic characters such as spaces or colons (':') will be replaced by underscores ('\_'). If the box is not checked, the contig names are exported as they are. In this case, importing the file into other programs might be problematic if contigs contain "problem" characters.

The second option allows you to **append the number of samples in contigs**. The number of samples in a contig will be added to the first line of the FASTA file.

The third option allows you to **append the sample (or contig) comments** to the header of the FASTA file. The comments from the sample information of each sample in CodonCode Aligner will be appended to the first line of the FASTA file. If comments extend over several lines, additional comment lines (starting with ";") will be added after the first header line.

By checking the box labeled **"Include gaps in FASTA files"**, you can include gap characters in the output. If the box is not checked, the exported sequences will be ungapped.

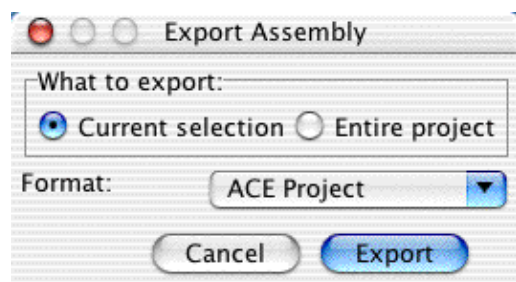
If you check the box **"Write FASTA quality files"**, a quality file in FASTA format will also be created. If you are exporting a single FASTA file, this file will be in the same folder as the FASTA file. The name of the quality file will be the name of the FASTA file, with ".qual" appended at the end of the name. If you are exporting individual FASTA files for each contig, a separate quality file in FASTA format will be created for each sample.

When the **"Format sequences"** box is checked, longer sequences will be split up into multiple lines, with 50 bases per line. This can be a problem for some older software, which expects the sequence to be in a single line; to write the entire sequence into a single line in the FASTA file, simply uncheck the **"Format sequences"** box.

# Exporting Assemblies

You can export Aligner projects for importing into other programs, for example the contig editor Consed. You can export entire Aligner projects, or parts of Aligner projects, for example single contigs.

First, select the contig(s) you want to export in the project window. Then, choose "**Export Assembly...**" from the "**File**" menu. This will bring up the following dialog:



You can choose to export the current selection or the entire project. Currently, only two formats are supported:

- the "[ACE](#)" format that is used by the Phred-Phrap-Consed package, and also supported by other contig editors
- the "[NEXUS/PAUP](#)" format, which is used by many phylogenetic analysis programs
- the "[Phylip](#)" format, another format often used by phylogenetic analysis programs

Please note that many formats may not really export the entire selection or project. For example, the ACE file generated when exporting in the ACE file format will contain only contigs, but not unassembled reads or samples in the trash.

## ACE Format Exports

The ACE format for exporting projects is modeled after the format used by the Phred-Phrap-Consed package, which is also supported by other sequence editors. Exporting will create three directories, as illustrated below:



The folder "edit\_dir" contains a single file, the ".ace" file. This is a text file that contains the information about the assembly, for example the number of contigs and the consensus sequences.

The second folder, "chromat\_dir", contains the chromatogram files format for the exported traces. The trace files are in SCF ("Standard Chromatogram Format") format.

The third folder, called "phd\_dir", contains "PHD" files for each sample. The PDH files are text files which contain the base calls, qualities, and additional information like tags.

You should be able to directly open the exported project in Consed, and (hopefully) in other contig editors that support the ACE file format. However, if you transfer the exported files to a different operating system, make sure that the files are transferred correctly. The chromatogram files must be transferred as binary files, and the other files must be transferred as text files. Incorrectly transferred files will likely cause problems when you try to open them.

## NEXUS/PAUP Format Exports

The NEXUS/PAUP format allows you to export projects for phylogenetic analysis programs like MacClade or PAUP. Only contigs will be exported, with a single file being created for each contig. Note that the exported files only contain information about the bases and gaps, but not about associated chromatograms or sequence qualities.

You have a choice of exporting in either "interleaved" or "sequential" format. Different phylogenetic programs have different restrictions on which kind of files they can read, so you may need to try both to see which one works for the program you plan to use.

If you have sample names that are very long and/or contain spaces or other "unusual" characters, it may be necessary to truncate or change the name of the samples in the exported files. Again, different programs have different restrictions. If sample names may to be changed in the exported files, Aligner will present you with a list of choices on how to change the sample names.

## Phylip Format Exports

The Phylip format allows you to export projects for phylogenetic analysis programs like MacClade or PAUP. Only contigs will be exported, with a single file being created for each contig. Note that the exported files only contain information about the bases and gaps, but not about associated chromatograms or sequence qualities.

You have a choice of exporting in either "interleaved" or "sequential" format. Different phylogenetic programs have different restrictions on which kind of files they can read, so you may need to try both to see which one works for the program you plan to use.

If you have sample names that are very long and/or contain spaces or other "unusual" characters, it may be necessary to truncate or change the name of the samples in the exported files. Again, different programs have different restrictions. If sample names may to be changed in the exported files, Aligner will present you with a list of choices on how to change the sample names.

## Other Formats for Exporting Assemblies

We plan to add support for other formats in the future. If you need a specific format, please let us know the format and the program that you need it for.

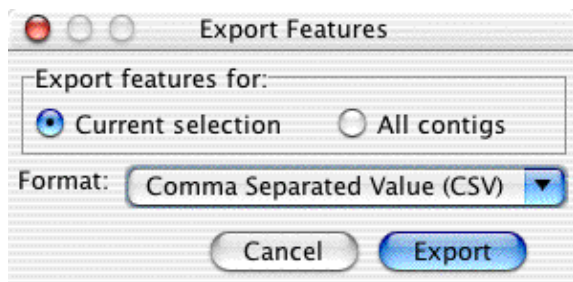
# Exporting Features

In Aligner, "Features" are regions which, for one reason or another, deserve special attention. They can include discrepancies, regions of low coverage, bases with tags, or a number or other criteria which you can define in the [feature preferences](#).

To export features for a set of contigs:

- Go to the contig view
- Select the contig(s) for which you want to export features (or, if you want to export features for all contigs, select any contig)
- Choose **"Export Features"** in the **"File"** menu.

A dialog will appear where you can specify details about the output format:

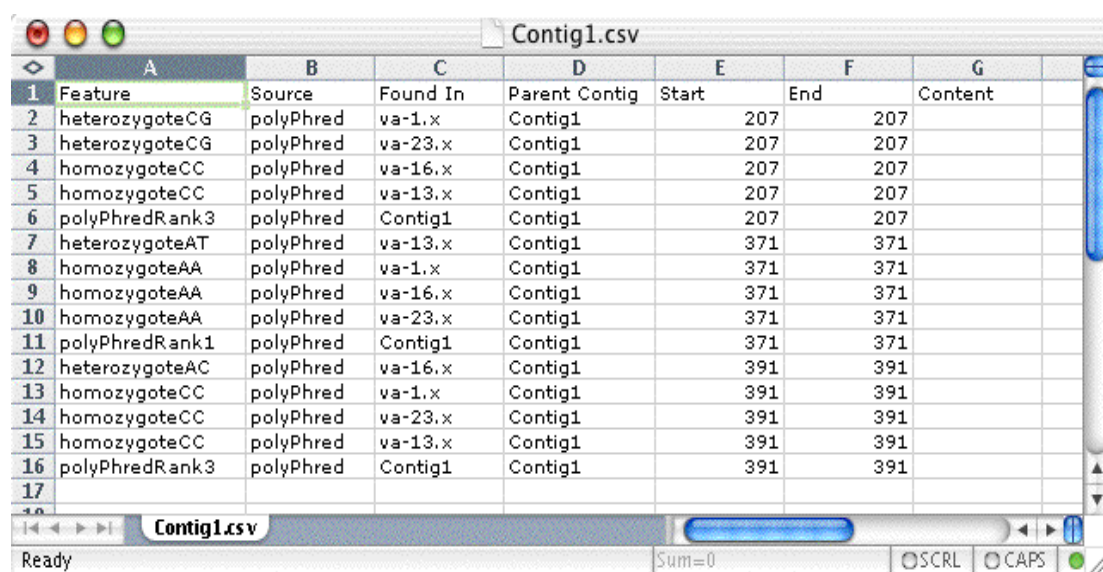


Use the radio button at the top to select whether you want to export only the selected sample or all samples in the contig. Then, use the pull-down menu to select the file format, and click on the "Export" button. Next, Aligner will show you a "Save As" dialog where you can select the location of the exported file or files. The exported files are text files, and can be opened with text editor or spread sheet programs.

One thing to note when exporting features is that CodonCode Aligner will always export all features in the consensus sequence and in all samples in a given contig. This is true even if you used the [Feature preferences](#) to specify to look for features only in the consensus sequences or the samples or the current selection when navigating (this selection does apply only for navigating, not for feature views or exports).

Here is an example of what exported features from the "PolyPhred\_example" project can look like after importing an exported feature file into Microsoft Excel:

## CodonCode Aligner User Manual



|    | A              | B         | C        | D             | E     | F   | G       |
|----|----------------|-----------|----------|---------------|-------|-----|---------|
|    | Feature        | Source    | Found In | Parent Contig | Start | End | Content |
| 2  | heterozygoteCG | polyPhred | va-1.x   | Contig1       | 207   | 207 |         |
| 3  | heterozygoteCG | polyPhred | va-23.x  | Contig1       | 207   | 207 |         |
| 4  | homozygoteCC   | polyPhred | va-16.x  | Contig1       | 207   | 207 |         |
| 5  | homozygoteCC   | polyPhred | va-13.x  | Contig1       | 207   | 207 |         |
| 6  | polyPhredRank3 | polyPhred | Contig1  | Contig1       | 207   | 207 |         |
| 7  | heterozygoteAT | polyPhred | va-13.x  | Contig1       | 371   | 371 |         |
| 8  | homozygoteAA   | polyPhred | va-1.x   | Contig1       | 371   | 371 |         |
| 9  | homozygoteAA   | polyPhred | va-16.x  | Contig1       | 371   | 371 |         |
| 10 | homozygoteAA   | polyPhred | va-23.x  | Contig1       | 371   | 371 |         |
| 11 | polyPhredRank1 | polyPhred | Contig1  | Contig1       | 371   | 371 |         |
| 12 | heterozygoteAC | polyPhred | va-16.x  | Contig1       | 391   | 391 |         |
| 13 | homozygoteCC   | polyPhred | va-1.x   | Contig1       | 391   | 391 |         |
| 14 | homozygoteCC   | polyPhred | va-23.x  | Contig1       | 391   | 391 |         |
| 15 | homozygoteCC   | polyPhred | va-13.x  | Contig1       | 391   | 391 |         |
| 16 | polyPhredRank3 | polyPhred | Contig1  | Contig1       | 391   | 391 |         |
| 17 |                |           |          |               |       |     |         |

Contig1.csv

Ready Sum=0 SCRL CAPS



# Exporting Aligner Projects in the Old Format

Older CodonCode Aligner versions (before 4.0) cannot read Aligner project files create by version 4.0 and newer. If you or a collaborator needs to open a CodonCode Aligner project with an older version of CodonCode Aligner, you need to export the project in the old format.

To export projects in the old format, choose "**Export => Aligner Project (Old Format)**" in the "**File**" menu.

This will create a new folder with the project name that contains a ".proj" file and several sub-directories. The project folder and all its contents (and not just the ".proj" file) constitute a project in the fold format.

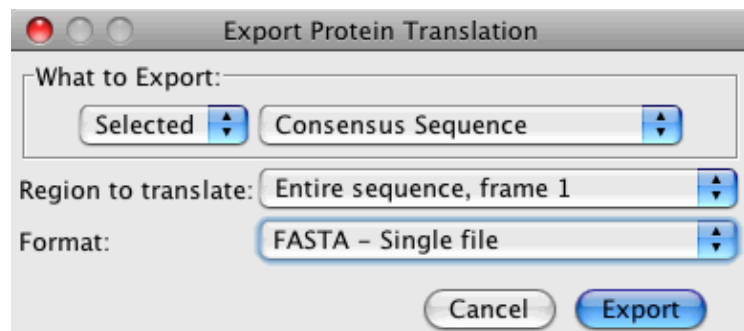
*Please be aware that the old project format is not intended for large projects with many samples, for example Next Gen projects! The old format creates at least two files for every sample in your project, and needs significantly more disk space than the newer ".ccap" format.*

# Exporting Protein Translation

You can export protein translations to text files in FASTA format as follows:

- Go to the project view
- Select the sequences you want to export protein translations for (*use shift-click to make continuous selections, and control-click (OS X: command-click) to make discontinuous selections*)
- Choose "**Export => Protein Translations...**" in the "**File**" menu.

A dialog will appear where you can specify details about the output format:



In the "**What to Export**" section at the top, you can specify what part of the project you want to export protein translations for. Use the left pull-down to select whether you want to export only the current selection or all samples and/or contigs. The second pull-down on the right allows you to select if you want to export samples, consensus sequences or both.

You can choose between different **regions to translate** using the pull-down in the middle:

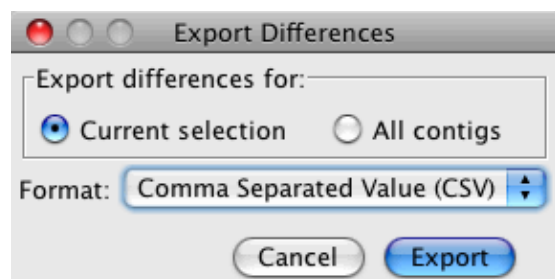
- The entire sequence, frame 1, 2, or 3
- The entire sequence, forward 3 frames
- The entire sequence, all 6 frames
- The selected bases
- The defined coding regions

The **Format** pull-down at the bottom gives you the format choice of a single FASTA file and one FASTA file per sample.

# Exporting Differences

Exporting Differences allows you to export the [Difference Table](#) you can see in the contig view in Aligner. The information exported will be the information you see in the overview panel of the contig view window if you show the differences there. This can exclude Ns, non-gaps or differences with a quality higher than the set threshold, depending on the filters you use when showing the difference table in the contig view.

To export information about the differences in your contigs, go to the "File" menu, and select "Differences..." from the "Export" sub-menu. This will show the following dialog:



Use the radio button at the top to select whether you want to export only the selected contig(s) or all contigs in your project. Then, use the pull-down menu to select the file format, and click on the "Export" button. Next, Aligner will show you a "Save As" dialog where you can select the location of the exported file. The exported file is a text file, and can be opened with text editor or spread sheet programs.

Here is an example of what exported differences can look like after importing an exported differences file into OpenOffice:

| Contig1.csv - OpenOffice.org Calc |                 |                        |       |       |       |       |       |                     |
|-----------------------------------|-----------------|------------------------|-------|-------|-------|-------|-------|---------------------|
|                                   | A               | B                      | C     | D     | E     | F     | G     | H                   |
| 1                                 | Project:        | PolyPhred_example.proj |       |       |       |       |       |                     |
| 2                                 | Exported:       | 20 Mar 2009 10:33      |       |       |       |       |       |                     |
| 3                                 | Excluded:       | Nothing                |       |       |       |       |       |                     |
| 4                                 |                 |                        |       |       |       |       |       |                     |
| 5                                 | Differences in: | Contig1                |       |       |       |       |       |                     |
| 6                                 | Position        | 107                    | 183   | 222   | 363   | 386   | 406   | Total # differences |
| 7                                 | Consensus       | A                      | T     | C     | T     | A     | C     |                     |
| 8                                 | va-13.x         | W                      | Y     | S     | T     | W     | C     | 4                   |
| 9                                 | va-16.x         | A                      | T     | C     | W     | A     | M     | 2                   |
| 10                                | va-1.x          | W                      | Y     | S     | T     | A     | C     | 3                   |
| 11                                | va-23.x         | W                      | Y     | S     | T     | A     | C     | 3                   |
| 12                                | Summary:        | 1A,3W                  | 1T,3Y | 1C,3S | 3T,1W | 3A,1W | 3C,1M | 12                  |
| 13                                |                 |                        |       |       |       |       |       |                     |

## CodonCode Aligner User Manual

Please note that columns and rows will be automatically swapped if you have more than 256 columns and more columns than rows. This is because all spreadsheet programs have a limit on the number of columns they can handle; the limit of 256 was chosen for old Excel versions.

You can adjust the threshold for the automatic row-column swapping as follows:

1. Create a text file that contains just a single line like the following one:  
ExportDiffsTransposeColMin=64000  
The file has to be a pure text file (not a Word document or similar), and the extension has to be ".alpref".
2. Drag and drop the file onto an Aligner project view (or use "Open.." to open the file).
3. Export export again.

The example above will allow you to export up to 64000 columns before CodonCode Aligner would automatically swap rows and columns; you can use different numbers if necessary. If you need further assistance, please contact us.

# Exporting Trees

If your contigs have [phylogenetic trees](#), you can export the trees in Newick format (you can build phylogenetic trees for contigs and contigs of contigs with the "**Built Tree...**" menu item in the "**Contig**" menu).

To export phylogenetic trees:

- Go to the project view
- Select the contigs you want to export (*use shift-click to make continuous selections, and control-click (OS X: command-click) to make discontinuous selections*)
- Choose "**Export => Trees**" in the "**File**" menu.

If none of your selected contigs have phylogenetic trees, you will not be able to export trees. To export trees, at least one of your selected contigs has to have a tree, which can be built using "Build Tree..." from the "Contig" menu. If you have any trees in your project and choose to export them, you will see the following dialog:



Use the radio button at the top to select whether you want to export only the selected contig(s) or all contigs in your project. The file format used for exporting trees is Newick format. When you click on the "Export" button, Aligner will show you a "Save As" dialog where you can select the location of the exported file. The exported file has the extension ".nwk" (for Newick), and can be opened in tree visualization software such as FigTree.

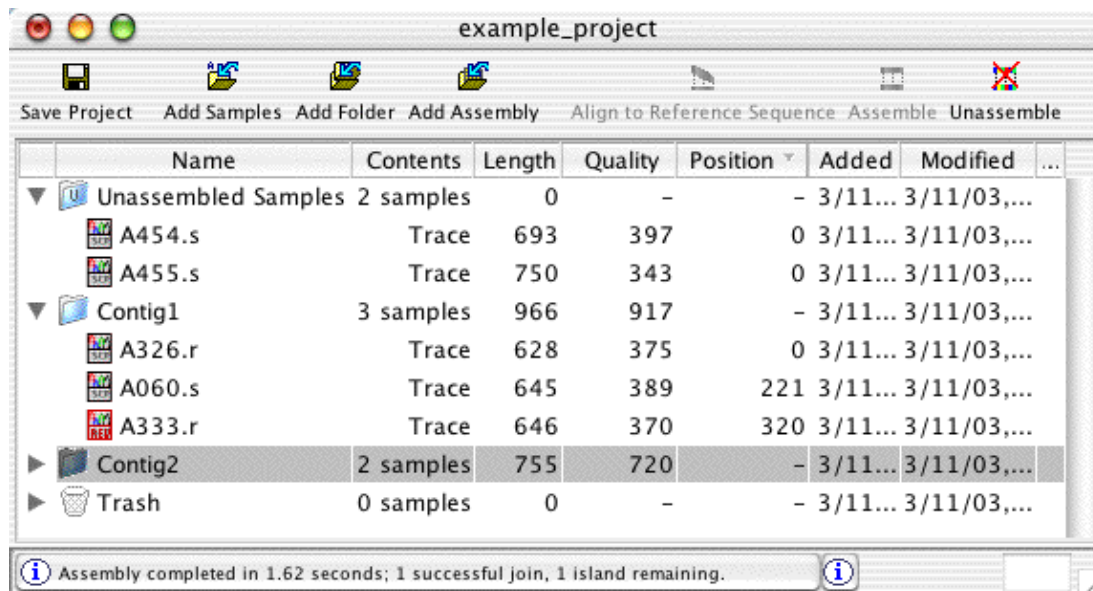
If you have one or more invalid trees in your selection of trees to export (trees are marked as invalid due to contig edits after the tree was generated), you will see a dialog asking you if you would still like to export the selected trees. You have the option of canceling the export, or exporting the trees even though they do not represent your current contigs.

Please note that sample names with "funny" characters (such as spaces, colons, or brackets) will be enclosed in single quotes. Some tree visualization software may not be able to read the exported Newick files if your sample names contain any of the following characters: single quotes, double quotes, brackets, colons, semicolons, whitespaces or commas. In this case, please make sure to use a newer program such as FigTree.

# Aligner Windows

## The Project View Window

The project window is the main window in CodonCode Aligner - most actions are started by selecting samples or contigs in the project window, and then choosing a menu option or one of the buttons at the top of the project window:



The project window is organized similar to the familiar list view in the Macintosh Finder, or the detail view in Windows Explorer. Every project has two folders:

- The "Unassembled Samples" folder contains samples newly added to a project, and not yet assembled or moved to the trash.
- The "Trash" folder, which contains samples marked for deletion, but not yet removed from the project.

In addition, projects that have been assembled or aligned will have one folder for each contig formed. You can **expand or condense folders** by clicking on the little triangle on the left of each folder. In the example above, the "Unassembled Samples" folder and Contig1 are expanded, showing details about the samples in Unassembled Samples and in Contig1.

## Selecting Samples and Contigs

For most things you do in CodonCode Aligner, you start by selecting the samples or contigs you want to do something with, for example vector screen or assemble. You can:

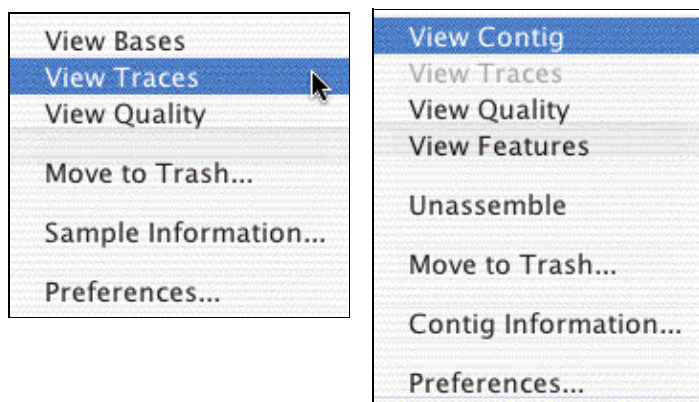
- **click** on a sample or contig to select the item
- **shift-click** to make a continuous selection (press on the first item, then press shift while clicking on the last item you want to select)
- **control-click** (OS X: **command-click**) to make a discontinuous selection

**Your current selection determines which menu items and buttons are active.** For example, you need to have at least two samples or contigs selected before you can choose "Assemble" from the "Contig" menu, or the "Assemble" button at the top of the window; the "Unassemble" menu item and button require that your selection contains only contigs; and so on. If a menu item is not available (and the button is "grayed out"), you have not selected the item or items required for this action.

## Menu shortcuts: Buttons and Popup menus

The most commonly used actions are available as buttons on the top of the project view, and as popup menus. You can hide the buttons by selecting "**Toolbars > Hide Toolbars**" in the "**View**" menu. If you do not see the toolbar at the top, select "**Toolbars > Show Toolbars**" in the "**View**" menu to make them visible.

If you press the right mouse button on Windows, or control-click on Mac OS X, in the project window, a **popup menu** with the most commonly used actions will be displayed. The content of the popup menu will depend on what you currently have selected - just try it out! Two example popup menus are shown below:



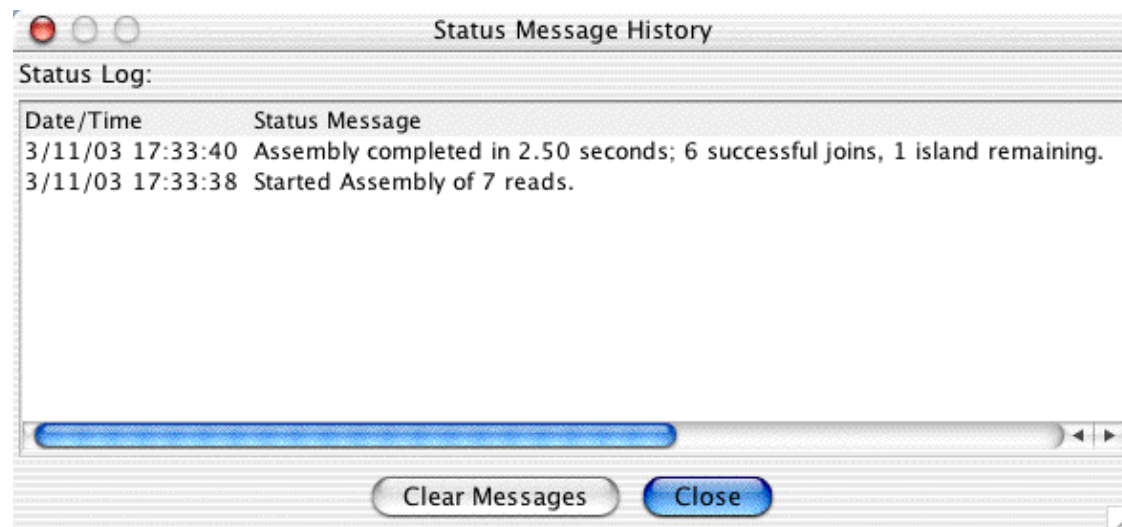
## Project View Columns And Sorting

The project view contains a number of columns, most of which are pretty self-explanatory. You can **click on the column headers to sort** the project view; however, the "Unassembled Samples" folder will always be on the top, and the "Trash" folder at the bottom. Here are descriptions of the columns, from left to right:

- the **Triangles** in the left-most column let you expand or condense contigs and other "folders"
- the **Name** column shows the name of the folder or sample; you can change sample and contig names through the "Sample Information" and "Contig Information" dialogs, accessible through the "Sample" and "Contig" menus
- the **Contents** column tells you details about what is in a folder or contig, or about the samples
- the **Length** column shows the length of a contig or sample
- the **Quality** column displays the number of bases with a quality score of 20 or above ("Phred20 bases") for samples and contigs that have qualities
- the **Position** column shows the position of the first base of a sample within a contig (only for samples in contigs, not for unassembled samples or samples in the trash)
- the **Added** column shows the date a sample was added to the project
- the **Modified** column shows the date a sample was last edited or otherwise modified
- the **Comments** column shows and comments you (or a program) may have added to the sample

## Status messages

The area at the bottom of the project view window is used by Aligner to display messages - status messages, warning, and error messages. If you click on this area, the "Message history" dialog will be displayed, which shows all previous messages:

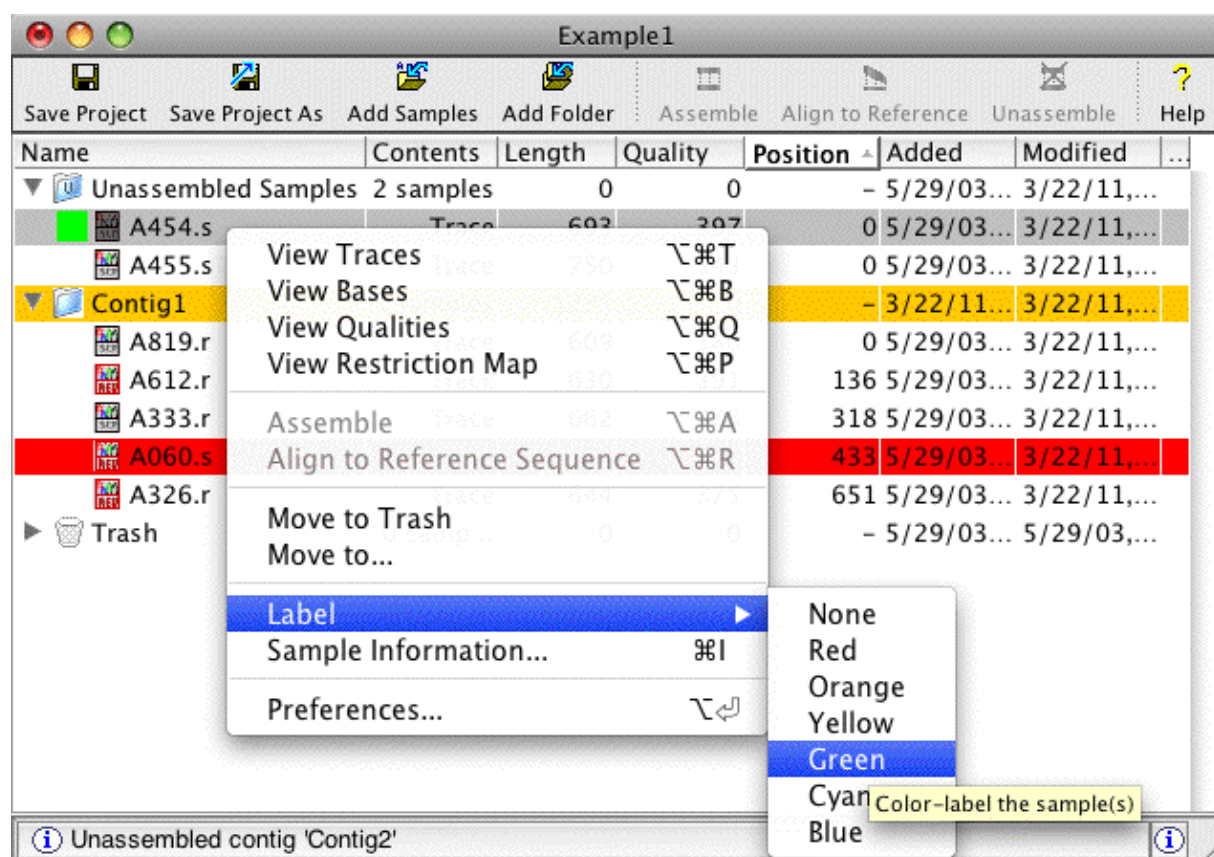


You will have to close this window before you can do anything else in Aligner. You can press the "Clear Messages" button to erase all old messages.

## Colored Labels

You can highlight sequences in the project view by using colored labels. Select the sample or contig you want to highlight, and go to "**Label**" in the "**Sample**" menu. Then choose the color you want to label the sample with from the "Label" submenu. You can also label the sequences in the project view by using the popup menu:

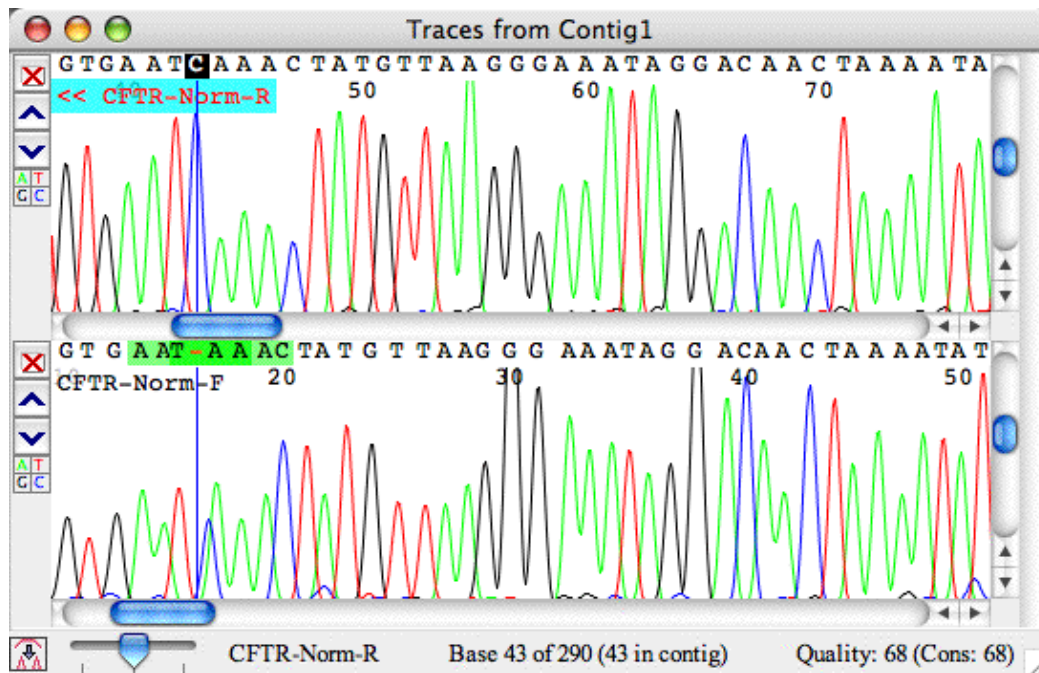




# Trace View

## General Information

In Trace View, you can view and edit samples that have trace data. Files with trace data that can be opened in a Trace View include standard chromatogram files (\*.scf) from LI-COR and other manufacturers, and Applied Biosystems ABI trace files (\*.abi, \*.ab1).



Each contig has one trace view window, in which all selected traces for this contig will be shown. The same is true for the "Unassembled Samples" folder - it has its own trace view window. The number of traces in a trace view window is limited only by available memory and operating system limitations; however, displaying many traces may be slow.

## Opening a Trace View

A trace view for a sample that has trace data associated with it can be opened by:

- Selecting the sample, then choosing **Trace View** from the **View** menu.
- Right clicking on the sample and choosing **Trace View**.
- Double-clicking on the sample (if you selected to open a trace view in the [double clicking preferences](#)).

## Scrolling and Scaling in Trace View

The vertical sliders at the right side of each trace let you scale a trace vertically to make the peaks larger or smaller.

You can use the sliders at the bottom of each sample to scroll a trace. Generally, all traces will be scrolled together when you scroll one of the traces. *It is possible to scroll traces individually; if you need to us this function, please contact CodonCode Corporation's support team for instructions.*

If the trace view window contains more traces than currently fit on the screen, the vertical slider on the right allows you to scroll between the traces. You can remove traces from the trace view window by clicking on the read "X" button on the left. You can move individual traces up and down by clicking on the up and down arrows on the left.

If you have a mouse with a scroll wheel, you can also use the scroll wheel to move around in the trace view window:

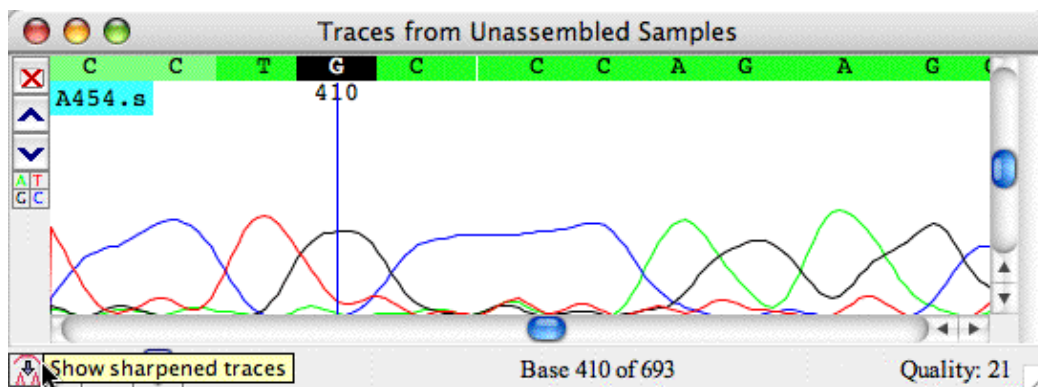
- when the mouse pointer is positioned over a trace, moving the scroll up or down will move horizontally (to the right or to the left)
- to scale a trace view **vertically**, keep the **control key** pressed while scrolling with the mouse wheel: this has the same effect as using the vertical scroll bar, making peaks larger or smaller
- to scroll between traces if the trace view window contains more traces than currently fit on the screen, move the mouse pointer over the vertical slider on the right allows, and then use the scroll wheel

You can change the default height of trace view panels in the "[Views](#)" preferences.

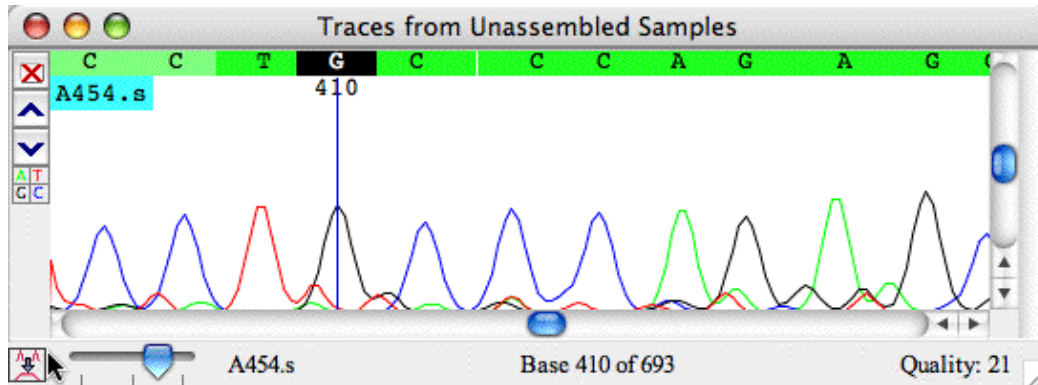
To scale the traces horizontally, use the slider in the bottom left corner of the trace view.\

## Trace Sharpening

If you are working with traces where the peaks overlap and are badly resolved, CodonCode Aligner's trace sharpening can help you to get a better look at your data. Here is an example of a badly resolved region in a sequence:



When you click on the little button in the lower right corner, CodonCode Aligner will try to sharpen the peaks in the displayed traces; the result looks like this:



In this example, it is much easier to see that there are indeed 3 Cs after the G, not 2 or 4. Looking at the sharpened traces can be a useful tool when working with less-than-perfect data.

To go back to the original traces, just click the little button in the bottom left corner again. You will notice that showing the sharpened traces may take a few seconds the first time, especially if the trace view shows several traces. This is because the trace sharpening is rather computation intensive.

When looking at the sharpened traces, keep in mind that you are looking at a mathematical artifact, which occasionally may give an incorrect impression - so always look back at the original traces, too, and use your scientific judgement.

## Colors and Highlighting

The colors in the Trace view are determined by your [color preferences](#); in the example above, a quality-based 3-color scheme was used. You [highlighting preferences](#) determine how discrepancies, edits, and tags are displayed. For more information, please check the "[Preferences](#)" help section.

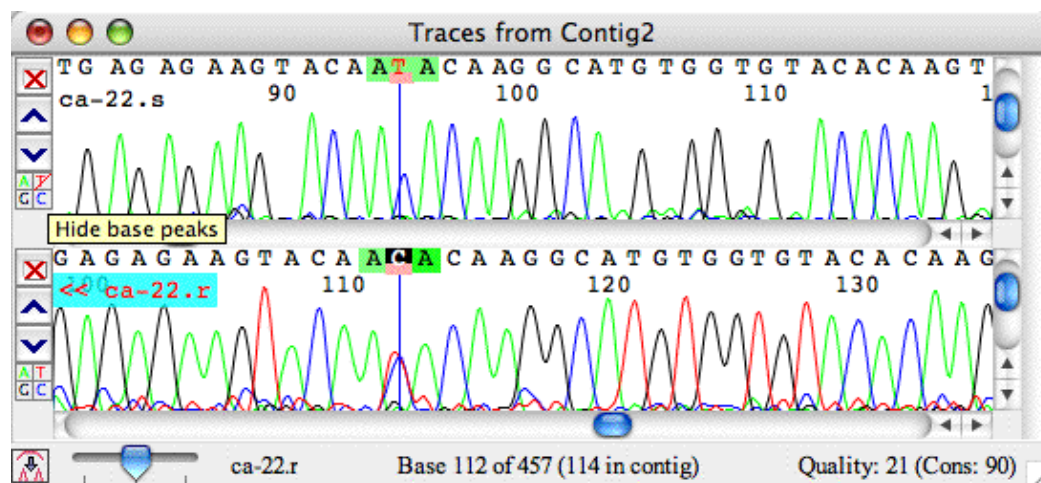
## Automatic Trace Selection

For trace views that show samples in **contigs**, CodonCode Aligner can automatically pick traces to display while you are moving around in a contig. You can turn this option on or off by selecting "**Auto Select Traces**" in the "**View**" menu. You can also set the number of traces to be displayed in the "[Views](#)" preferences.

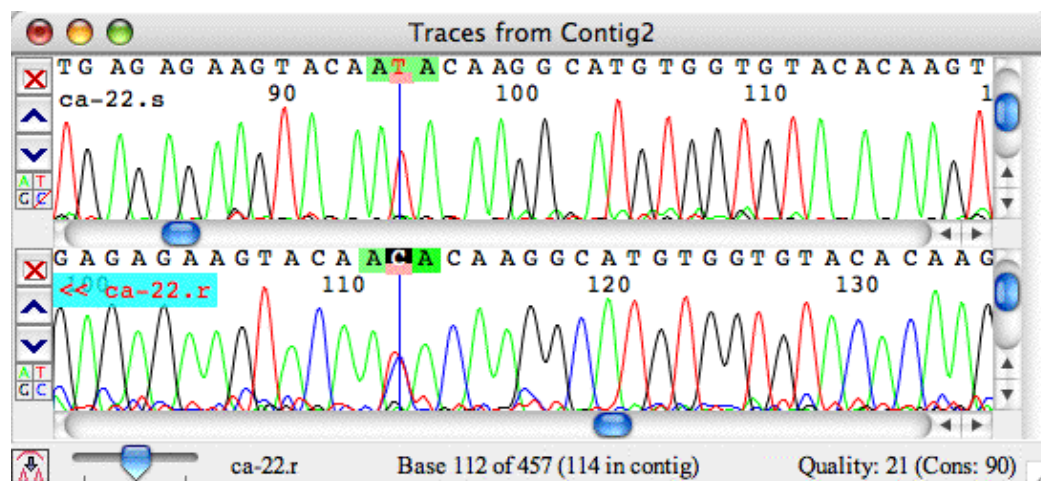
**For automatic trace selection to work, you need to first open both the contig view and the corresponding trace view** (for example by double-clicking in the overview panel in the contig view). Once both views are open, Aligner will automatically add and remove traces when you go to a new location in a contig, for example by clicking in the overview panel or by moving the cursor in the bases panel. The trace view window will grow larger and smaller, depending on the number of traces shown and the available space on your screen. If you want to see more than 2 or 3 traces, you may want to change the height of the trace panels to "Small" or "Tiny" in the "[Views](#)" preferences.

## Hiding Some Traces

You can hide traces for one or more bases by clicking on little colored boxes with the corresponding letter of the trace on the left side. This can be very useful when analyzing heterozygous mutations, where a peak for one base may obscure a second peak. Below is an example - the same traces as above, but the 'T' lane is hidden:



The red line through the 'T' box at the left indicates that the T lane is hidden. Clicking on it again will show the T lane, and clicking on the 'C' will hide the C lane, as shown below:



Hiding lanes will affect only one trace, and only until it is closed; re-opening the trace will show all traces again.

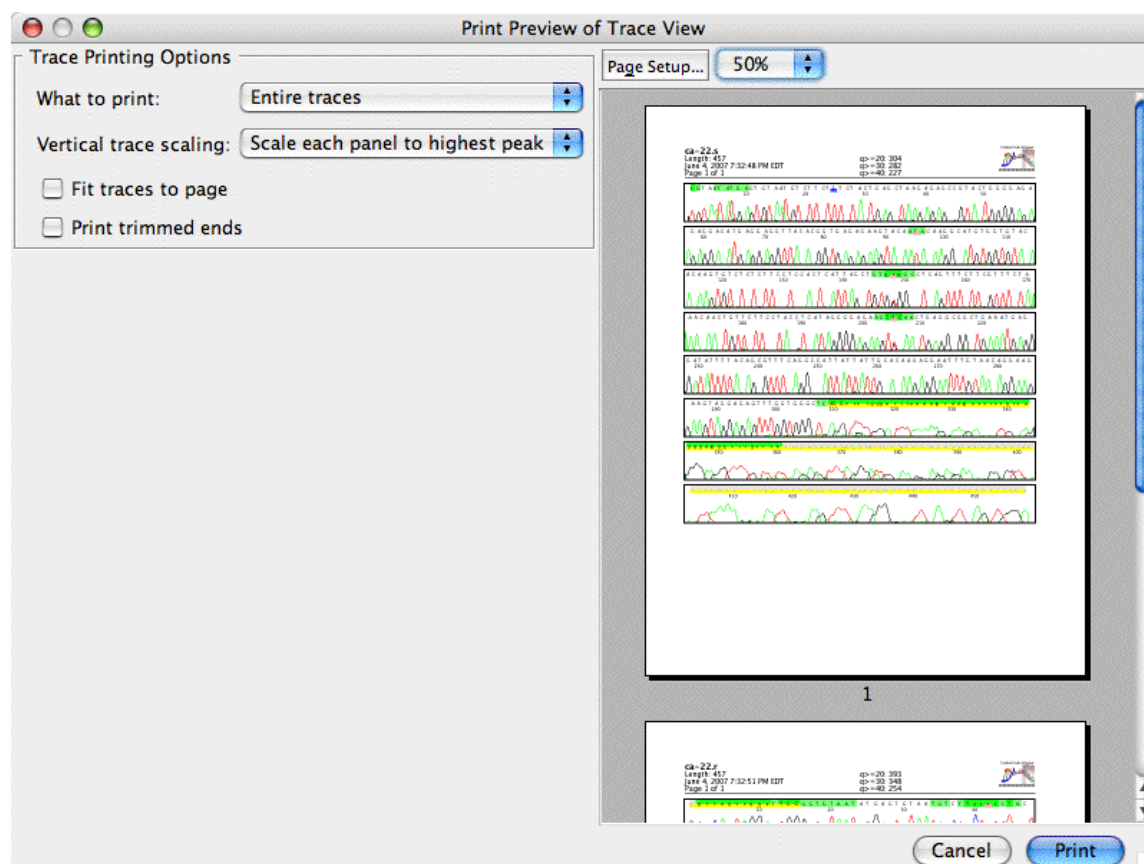
You can also press the shift-key and/or the alt-key while clicking in the base boxes to hide more than just one trace:

- **Shift-click** to hide all traces except the one for the base you are clicking on in the current sample; shift-click again to show all traces again
- **Alt-click** to hide the trace for this base in **all** samples in the trace view window; alt-click again to show the trace in all windows again (OS X users can also use the command key instead of the alt key)
- **Alt-Shift-click** to hide all other traces in all samples. Repeat to show all traces in all samples again. Try it out!

## Printing Traces

To print the traces that are currently in your trace view, first select the trace view, and then choose "**Print**" from the "**File**" menu (or use the keyboard shortcut Control-P on Windows, Command-P on OS X).

This will show the a print preview dialog like the one below:





## CodonCode Aligner User Manual

On the left, you can choose whether to print the entire traces or just what you see on the screen, how to scale traces, and a few more options. On the right side, you get a preview of the print results with your current settings. The different options are described in detail on the "[Printing Preferences](#)" page.

# Base View Window

Choose **Base View** from the **View** menu to display the bases for a sample or consensus:

AF033819

Print Colors Bases<>Transl. Next Frame Help

1 ggtctctctggttagaccagatctgagcctgggagctctctggctaactaggaaccactgctt  
66 aagcctcaataaagcttgcttgagtgcctcaagtagtggtgcccgtctgttggtgactctgg  
regul.. 5' UTR  
HaeII  
131 taactagagatccctcagacccttttagtcagtggtgaaaatctctagcagtgggcgccgaacag  
5' UTR primer binding ..  
196 ggacctgaaagcgaaggggaaaccagaggagctctctcgacgcaggactcggcttgctgaagcgc  
261 gcacggcaagaggcgaggggcggcgactggtgagtacgccaaaaatttgactagcggaggctag  
ClaI  
326 aaggagagagatgggtgctgagagcgtcagtattaagcgggggagaattagatcgatgggaaaaaa  
gag mature peptide  
391 ttcggttaaggccagggggaaagaaaaaatataaattaaaacatatagtatgggcaagcaggag  
gag mature peptide  
Map Bases  
AF033819 1503 bases (336-1838) of 9181



## CodonCode Aligner User Manual

You can edit sequences in the base view if they are editable; but for sequences with traces, it's generally a better idea to edit in the trace view. The bases are displayed according to your [color preferences](#).

At the bottom of the base view window, you see the name of the sample, the current cursor position, and (if the sample has qualities) the quality at the cursor.

The toolbar on the left side lets you choose display options. The top button turns the display of annotation tags (features) on or off. The pulldown on the right side lets you choose which features are shown. When the display of features is deselected, features may still be indicated by differently drawn bases, for example boxes or underlining, as determined by the "Highlighting" preferences.

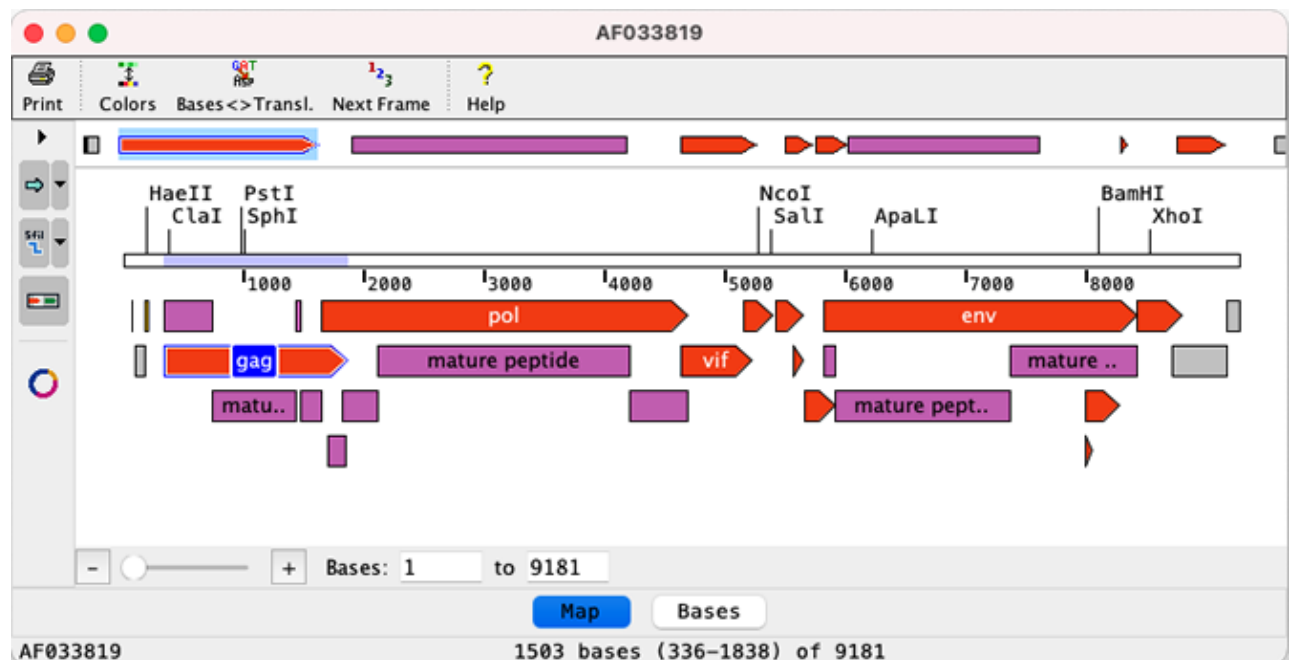
The middle button lets you turn the display of restriction enzyme cut sites on or off. The pulldown menu on the right side of the button shows a number of different options which enzyme sites should be shown.

The third button controls the display of the "mini map" above the bases. The mini map shows an overview of features in the entire sequence. Where multiple features overlap, only one feature will be shown in the mini map, with preference given to coding sequence tags.

The fourth button allows you to group bases. You can for example choose to show bases in groups of three to match amino acids.

Clicking on a feature will select the feature and the bases that it covers. Shift-clicking will extend the selection. You can select a restriction fragment by first clicking on the restriction enzyme name above the cut site, and then shift-clicking on a second restriction enzyme name.

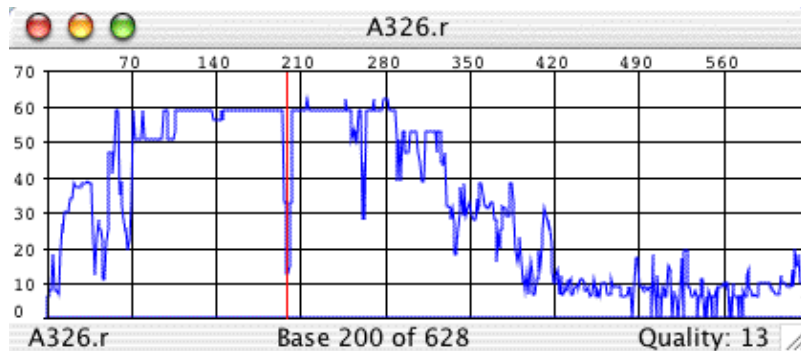
The buttons "**Map**" and "**Bases**" at the bottom of the view make it possible to switch between showing the bases and a graphical overview (the map) of the sequence:



Maps can be shown as linear or circular maps, which can be changed by using the fourth button in the side toolbar on the left. Linear maps allow zooming. You can use the slider, the "+" and "-" buttons, or enter the range of bases to show below the map.

# Quality View Window

Choose **Quality View** from the **View** menu to display a graph of the quality values for the selected sample or consensus.

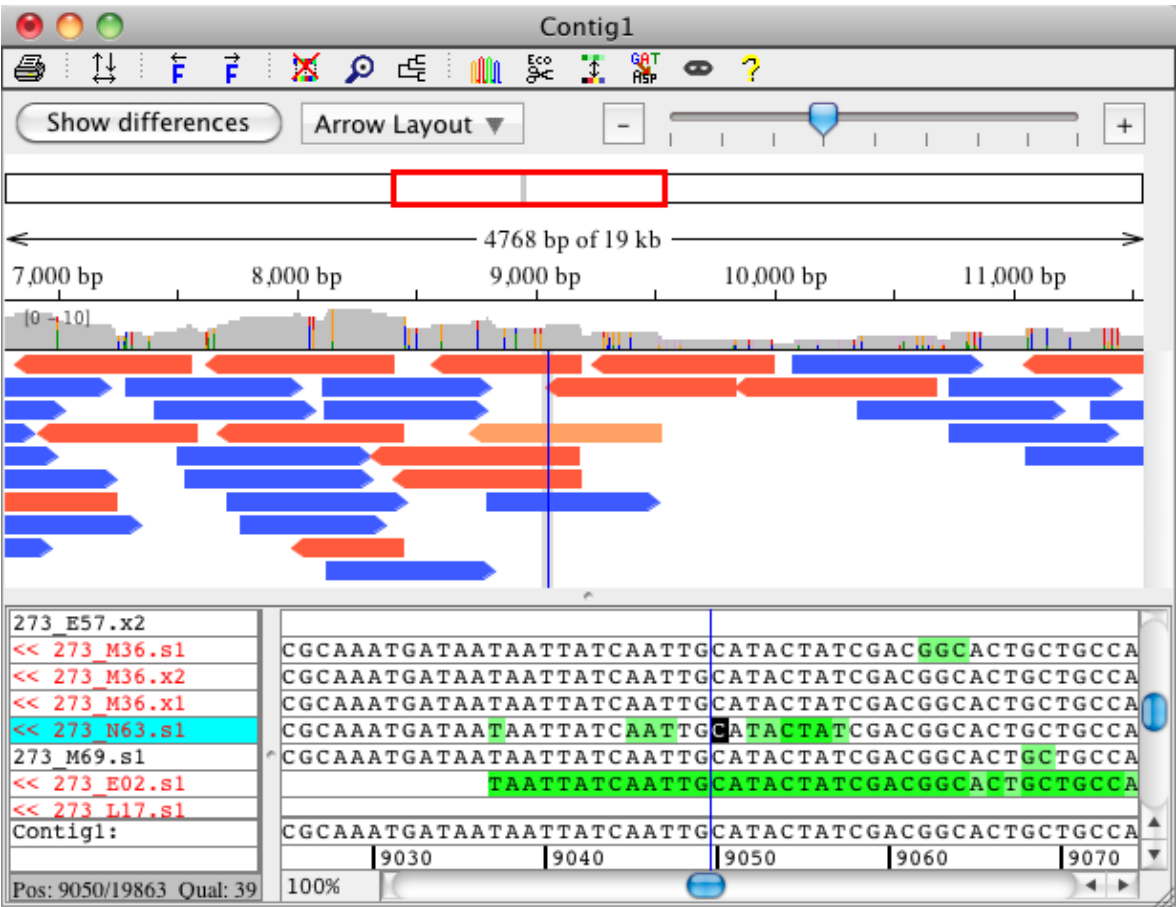


The **Quality View** graphs [quality value](#) (Y-axis) versus base position (X-axis) for all base calls. The quality graph can be enlarged or reduced by resizing the Quality View window (click and drag a corner of the window).

Clicking in the quality view window will move the cursor in all views for this sample. This can be useful to quickly check out problem regions, like the low quality region near base 200 in the example above. Just open a trace view and a quality view for a sample, and click at the low-quality regions in the quality view; then look at this region in the trace view.

# Contig View Window

Choose **Contig View** from the **View** menu to display a window showing an overview or the differences of the alignment and the aligned bases for a contig:



## CodonCode Aligner User Manual

The upper half of the contig can either show a **graphical overview** or a **difference table** of the contig. The graphical overview is shown in the example above. If you press the button "Show differences", you will see the difference table for the same alignment:

The screenshot shows the 'Contig2' window in CodonCode Aligner. The top menu bar includes: Print, Reverse, Build Tree, View Traces, Colors, Bases<>Transl., Next Frame, Mask Matches, and Help. Below the menu bar are three buttons: 'Show overview', 'Change size', and 'Exclude'. The main area is a grid showing the alignment of various contigs. The columns are labeled with contig IDs: 5345, 5349, 5352, 5382, 5387, 5422, 5462, 5467, 5474, 5475, 5516, 5530, 5564, 5565, 5818, 5819, and 5. The rows show the sequence of bases (A, C, G, T, -) for each contig. The bottom panel shows a list of contigs on the left and their corresponding sequence alignments on the right. The contigs listed are: 273\_N35.x2, 273\_N35.s1, << 273\_L95.s1, << 273\_N86.s1, << 273\_E77.s1, 273\_M77.s1, << 273\_N64.s1, 273\_M17.s1, and Contig2:. The sequence alignments are shown in a color-coded format. The bottom status bar shows 'Pos: 5467/9989 Qual: 100%'.

You can switch between the difference table and the graphical overview by pressing the button "**Show differences**" if you are looking at the overview, or the button "**Show overview**" if you are looking at the differences.

## Moving Around

You can move around in the aligned bases panel by using the scroll bar at the bottom, or by using the keyboard. When moving around in a contig view, the gray rectangle in the contig overview panel moves, indicating that a different part of the contig is shown; but the blue line for the cursor position does not change until you also click on a base in the aligned base panel.

Using the keyboard, you can move around as follows:

- Move around with the right and left arrow keys; you move one base at a time.
- Press the "page up" and "page down" keys a few times; this moves you by a full screen forward or backwards.
- Try the "home" and "end" keys; they move you to the beginning respectively end of the selected sequence. If the consensus sequence is selected, you go to the beginning and end of the contig; if you select a read, you move to the beginning or end of the read.

If you have a mouse with a scroll wheel, you can also use the scroll wheel to move around in a contig. Moving the scroll down will move forward in a contig, moving up will move backwards. If you have a contig with many reads in it, then pressing the "control" key while scrolling with the mouse wheel will move the read list up and down.

## Contig Overview Panel

Some things worth noting for the **graphical overview** of the contig (the "overview panel") in the upper half of the contig view:

- The zoom scale at the top right corner allows zooming in and out
- The box below shows which region of the contig is currently displayed in the coverage and arrows panels below
- The coverage graph shows the coverage at each position, as well as color-highlighted differences with a settable threshold for the allele frequency (the default is 0.1)
- The arrows panel shows the location and orientation of individual samples in the contig (*unless you are working with very large contigs and have zoomed out to show a large region*)
- When zoomed in, the sample arrows show differences to the consensus sequence as colored bars
- Moving the mouse over an arrow will display some information about the read (the name, direction, alignment start location, current cursor location, base and quality at this position for the sample base, and the consensus base)
- The sample arrows can be displayed in a "stacked" or "packed" layout and you can color them by direction.
- Unaligned ("dangling") ends of reads are shown in a light gray color (in projects where ends were clipped before assembly, samples will typically have no or just short unaligned ends)
- A vertical blue line shows the current cursor position in the contig
- The currently selected sample or samples are drawn in a lighter color (in the example at the top of this section, the orange read in the middle of the overview panel is selected)
- You can click in the overview panel to move around in a contig, as explained in the section "Navigating using the overview panel" below

## Coverage graph in the overview panel

The coverage graph in the overview panel shows the coverage at each position in your contig. The displayed coverage range is shown on the left side of the coverage panel (e.g. "[0-10]"). The coverage panel shows differences between the samples as colored bars that represent the amount of bases in base colors. To control which differences are displayed, the allele frequency threshold is settable through the pop-up menu in the coverage panel.

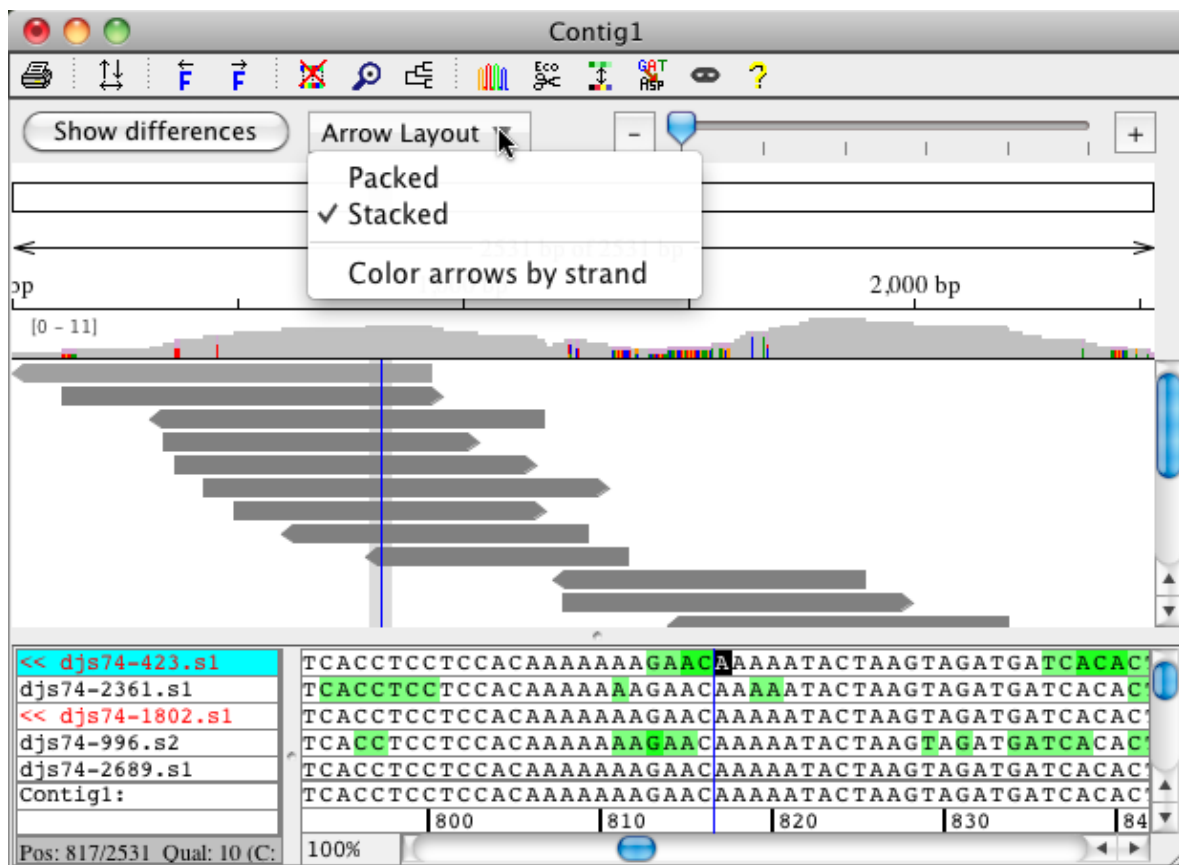


## Navigating using the overview panel

You can **click** in the overview panel to move the cursor. If you click on an arrow, the corresponding read will be selected, and selected read and base will be shown in the lower "aligned bases" panel. If you click somewhere else (in the white spaces), the corresponding consensus bases will be selected (you may have to use the vertical scroll bar to see bases for aligned reads at this location). You can also select the consensus by clicking in the coverage panel or on the ruler displaying the base pair numbers. Another way to move around in the overview panel is by clicking on the top bar representing the contig or moving the red bar which highlights the currently displayed section of the contig. This allows you to move around in the overview panel without updating the selection in the aligned bases panel. Holding the left mouse button pressed and moving the mouse to the left or right after clicking in the overview panel will update the overview display the same way.

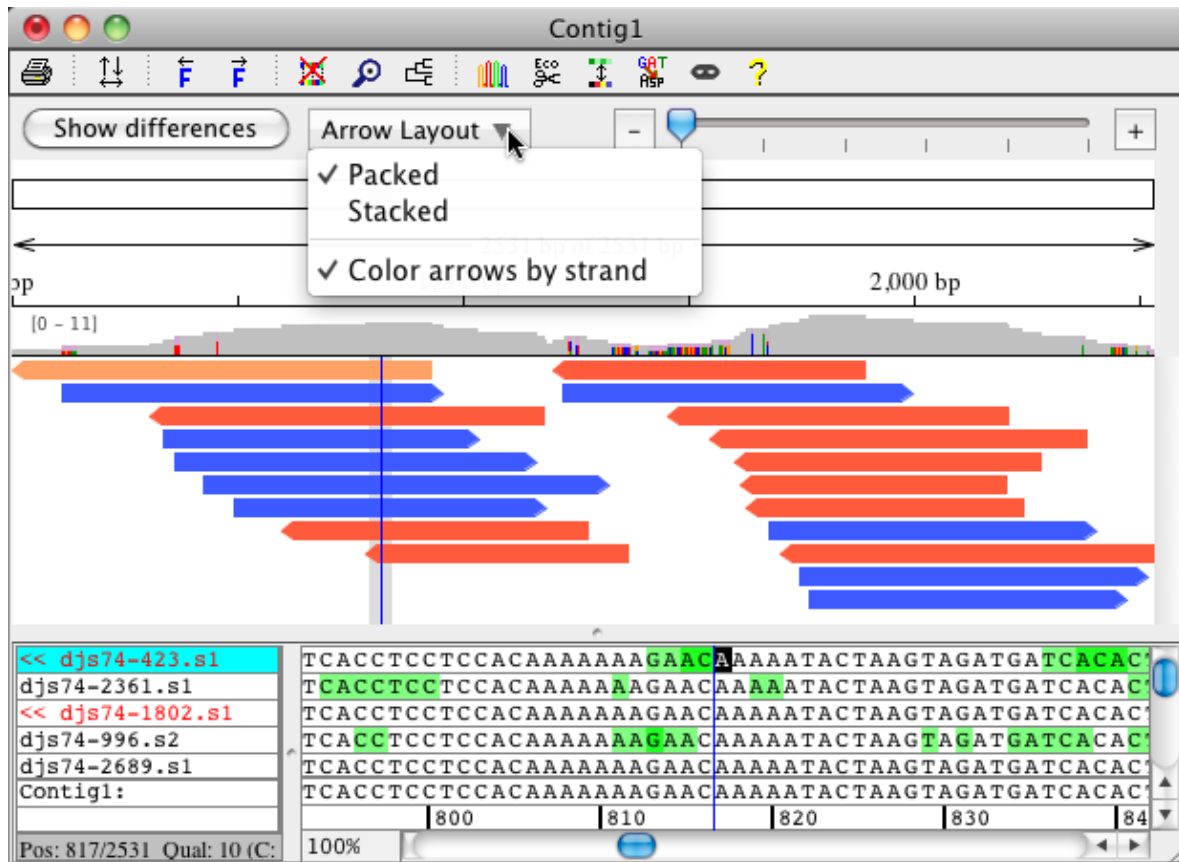
## Changing the display of the sample arrows

The sample arrows in the overview can be displayed "**stacked**" or "**packed**". The stacked layout stacks the samples one under the other. This layout is great for small projects or if you sort your samples:



The packed layout packs the samples in the overview so you can see more of your sample arrows at once:





This layout is very useful for larger projects. The packed layout dismisses the sort order of the samples to pack them - if you want to see your samples in the same order in the bases panel and in the arrow panel use the stacked layout.

The arrow layout can be changed through the pop-up menu in the graphic overview or by using the "**Arrow Layout**" button above it. In addition to the sample layout you can also change if you want to view the **arrows colored by strand** or not.

## Contig Difference Table

Some things worth noting for the **difference table** of the contig, which can be shown in the upper half of the contig view:

- You can move around in the difference table by using the scroll bar at the bottom and on the right side of the difference table
- The difference table lists all samples of your alignment and shows their names in the left column of the difference table
- The right column of the table contains the number of all differences for the sample in this row
- The top row of the difference table shows the position of the differences in this column in the contig and the contig base at this position
- The bottom row of the difference table displays a summary of the differences in all samples at this position
- The currently selected sample or samples are shown with a blue background in the table; the selected base has a black background

## Display options for the difference table

The toolbar above the difference table allows you to switch back to the overview, but also to **change the size** of the difference table and to **use filters** for choosing which differences are displayed.

**Changing the size of the difference table** allows you to either look at an overview of the differences, or to look at the differences in a way that you can see the single base differences. You can change the size of the difference table by pressing the "Change size" button. The overview of the differences is a condensed version of the difference table, and can be highlighted by colors as shown in the example below:

The screenshot shows the 'Contig1' window in CodonCode Aligner. The toolbar includes buttons for 'Show overview', 'Change size', and 'Exclude'. The main area displays a difference table with columns for positions and consensus bases (G, C, T, A). The table is color-coded to show differences between sequences. A summary row at the bottom provides counts for each base. Below the table, a list of sequence files is shown, including 'djs74-2350.s1', 'djs74-1180.s1', and others. The bottom status bar shows the current position as 1305/2711 and a 108% zoom level.

You have the option to **exclude certain differences** by using the difference table filters. The filters can be changed by clicking on the "Exclude" button above the difference table:

The screenshot shows the CodonCode Aligner interface with the 'Contig1' window. The 'Exclude' menu is open, displaying the following options:

- Show all columns and rows
- Exclude Ns
- ✓ Exclude non-gaps
- Exclude high consensus quality
- Exclude low frequency changes
- Hide rows without changes
- Set filters and thresholds...

The background shows a difference table with columns for positions and consensus, and rows for individual sequences. The 'Summary' section at the bottom left shows the following data:

| Summary: | 1G | 1C | 2T | 4T | 7T | 7T | 7C | 6G | 6C | 5C | 6C | 5- | 2A | 3A | 3A | 2G | 5T | 5- | 7- | 10- | 1... | 1... | 1... |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-----|------|------|------|
|          | 1- | 1- | 3- | 1- | 1- | 1- | 1- | 1- | 1- | 2- | 1- | 1C | 1- | 1- | 1- | 1X | 1X | 1X | 3X | 1X  | 1X   | 1X   | 1X   |

The bottom panel shows a sequence alignment view with the following sequence:

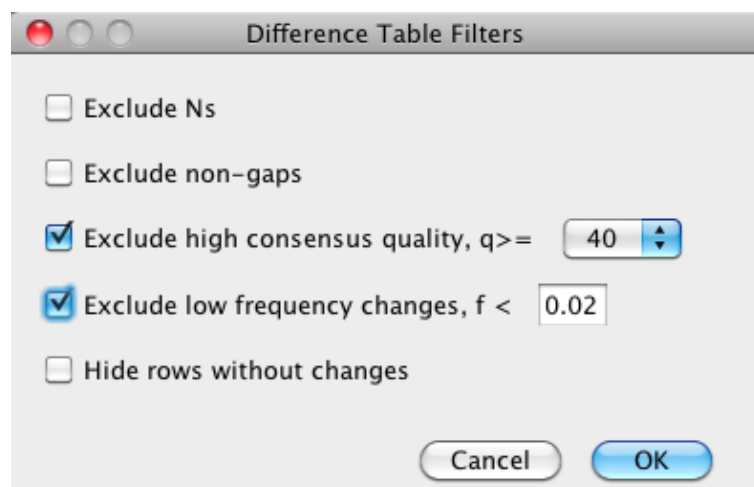
```

djs74-2350.s1 g a t a c T G A T G T A T C T T G G G C C A C A G T G c - a g A C A T T G T C A G c a
djs74-1180.s1
<< djs74-1532
<< djs74-237.
<< djs74-1432
djs74-2931.s1
<< djs74-3174
Contig1:
1280 1290 1300 1310 13
Pos: 1305/2711 Qu 108%
  
```

There are five types of filters you can use to look at the difference table:

1. **Exclude Ns:** You can exclude all differences that contain a "N" by selecting this option. The resulting difference table will omit any columns where the only different bases are "N".
2. **Exclude non-gaps:** By checking the "Non-gaps" option, you can exclude all differences that do not contain any gaps. The resulting difference table will only show differences that have a gap in at least one of the samples at a certain position.
3. **Exclude high consensus quality:** Checking this filter excludes all differences with a *consensus quality* greater than the one set in the filter dialog. You can set the quality by clicking on "Set filters and thresholds" which will show the filter dialog. You have the option to set the quality to 20, 30, 40, 50 or 60. For example excluding all differences with a quality greater than 40 will only show differences at places of the alignment where the consensus quality is less than 40.
4. **Exclude low frequency changes:** If this filter is selected your difference table will exclude all differences with a frequency below the one set in the filter dialog. You can set the frequency by clicking on "Set filters and thresholds" which will show the filter dialog. For example if you set the frequency to 0.1 and select the filter, your difference table would only show differences that occur in 10% or more of your samples.
5. **Hide rows without changes:** This filter allows you to omit all samples from your difference table that do not have any differences compared to the consensus after considering all other filters. Without checking this filter if one of the samples has a difference at a position, your difference table will include the base for each sample at this position. Checkign the filter will only show the samples that have the difference.

"**Show all columns and rows**" will cancel all filters and show the difference table with all differences for this contig. Clicking on "**Set filters and thresholds**" opens a dialog that allows you to set the thresholds for some filters. Here you can set the quality threshold for the filter that shows only differences below a certain consensus quality, and you can set the frequency cutoff for the filter that allows you to only show differences above a certain frequency:



## CodonCode Aligner User Manual

The example below shows a difference table where all filters were used. The resulting difference table does not contain any columns where the only differences are Ns; all of the shown differences have a gap; the table only includes differences that have a consensus quality of less than 60; all differences shown occur in at least 2% of the samples; and rows without changes relative to the consensus are not shown.

**Contig1**

Print Reverse Build Tree View Traces Colors Bases <> Transl. Next Frame Mask Matches Help

Show overview Change size Exclude

Show all columns and rows

- ✓ Exclude Ns
- ✓ Exclude non-gaps
- ✓ Exclude high consensus quality
- ✓ Exclude low frequency changes
- ✓ Hide rows without changes

Set filters and thresholds...

| Position  | 241 | 268 | 972 | 973 | 1131 | 1328 | Total |
|-----------|-----|-----|-----|-----|------|------|-------|
| Consensus | C   | G   | T   | T   | A    | T    |       |
| A454.s    | -   | G   |     |     |      |      | 2     |
| A455.s    | C   | -   |     |     |      |      | 8     |
| A612.r    |     |     | T   | T   | -    | T    | 5     |
| A333.r    |     |     | -   | -   | A    | T    | 2     |
| A819.r    |     |     |     |     |      | -    | 3     |
| Summary:  | 1C  | 1G  | 2T  | 2T  | 1A   | 2T   | 24    |
|           | 1-  | 1-  | 1-  | 1-  | 1-   | 1-   |       |

<< A455.s

A326.r

A060.s

A612.r

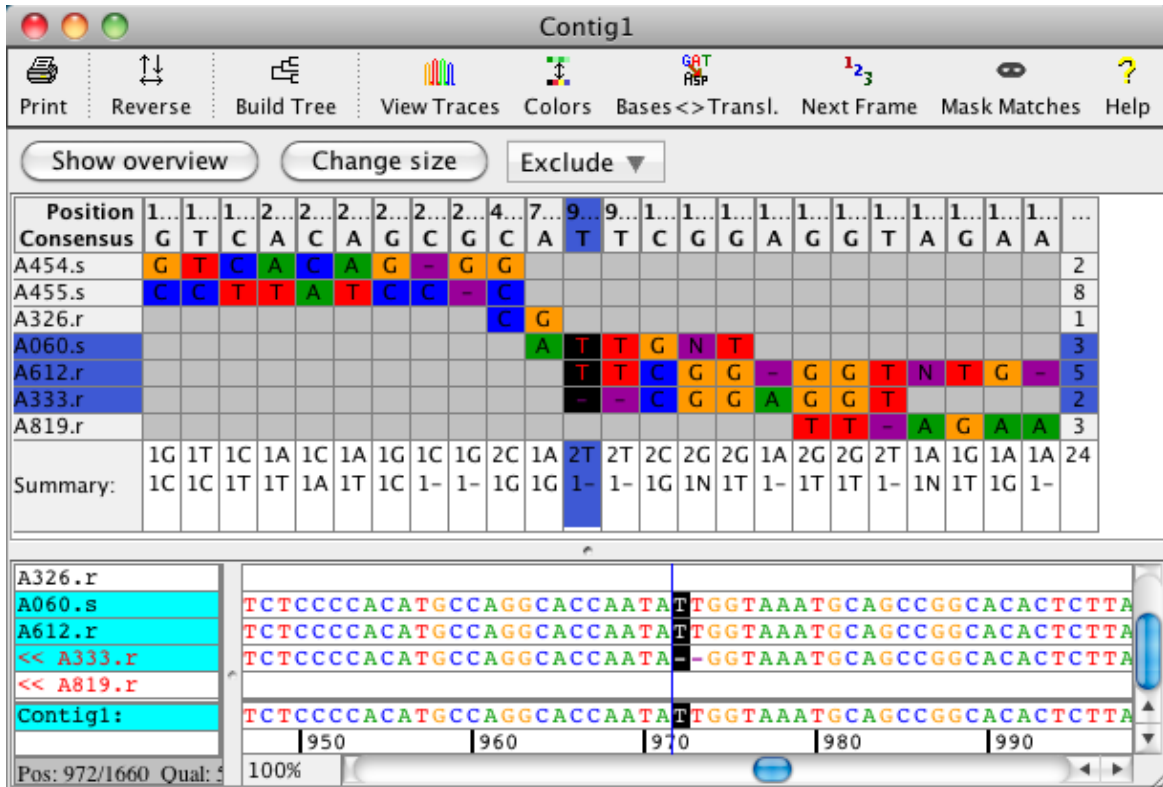
<< A333.r

<< A819.r

Contig1:

Pos: 972/1660 Qual: 5 100%

Here in comparison the difference table for the same contig showing all columns and rows:



## Navigating using the difference table

You can **click** in the difference table to move the cursor, for example to examine a specific difference.

The corresponding read will be selected, and the selected read and base will be shown in the lower "aligned bases" panel. If you click in the top row, the corresponding consensus base will be selected, which will also select all bases in all samples at this position (see picture above).

## Aligned Bases and Consensus Protein Translation

The lower half of the contig view shows the **aligned bases** for the samples in a contig. It can also show the **protein translation** of the consensus sequence, depending on how your preferences are set. You can choose to show no translation, one frame, three frames, six frames or the translation for an annotated coding region in the [protein translation preferences](#) and in the "View" menu. The cursor position is indicated by a black line, and the currently selected base or bases by a different background color (in the example above by black background). You can use the the cursor keys and the horizontal scroll bar at the bottom to move around in a contig.


You can zoom in and out using the zoom popup control at the bottom left of the base panel, or by changing the font size for the contig view in the [view preferences](#).

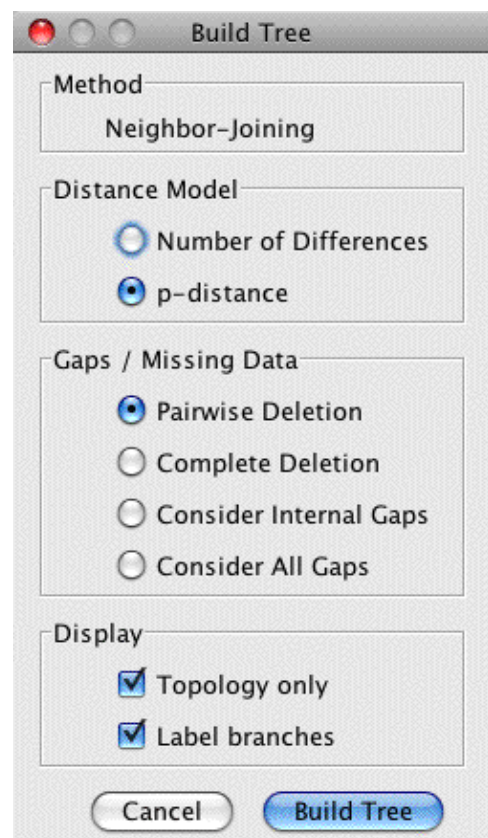
The bases are displayed according to your [color preferences](#). In the picture above, a quality-based three-color scheme was used.

You can **resize** the panels by dragging the bar between the panels.

The names of the samples are shown on the left side of the aligned bases panel. Reverse-complemented samples are indicated by "<<" before the name, and by **red** sample names. You can **select** samples by clicking on the sample.

## Phylogenetic Trees

The contig view can also display a **phylogenetic tree** for its contig to the left of the sample names. You can generate the tree by going to the "**Contig**" menu and selecting "**Build Tree...**", or by using the toolbar button  "**Build Tree**". This will show a dialog with different options of how to build the tree:



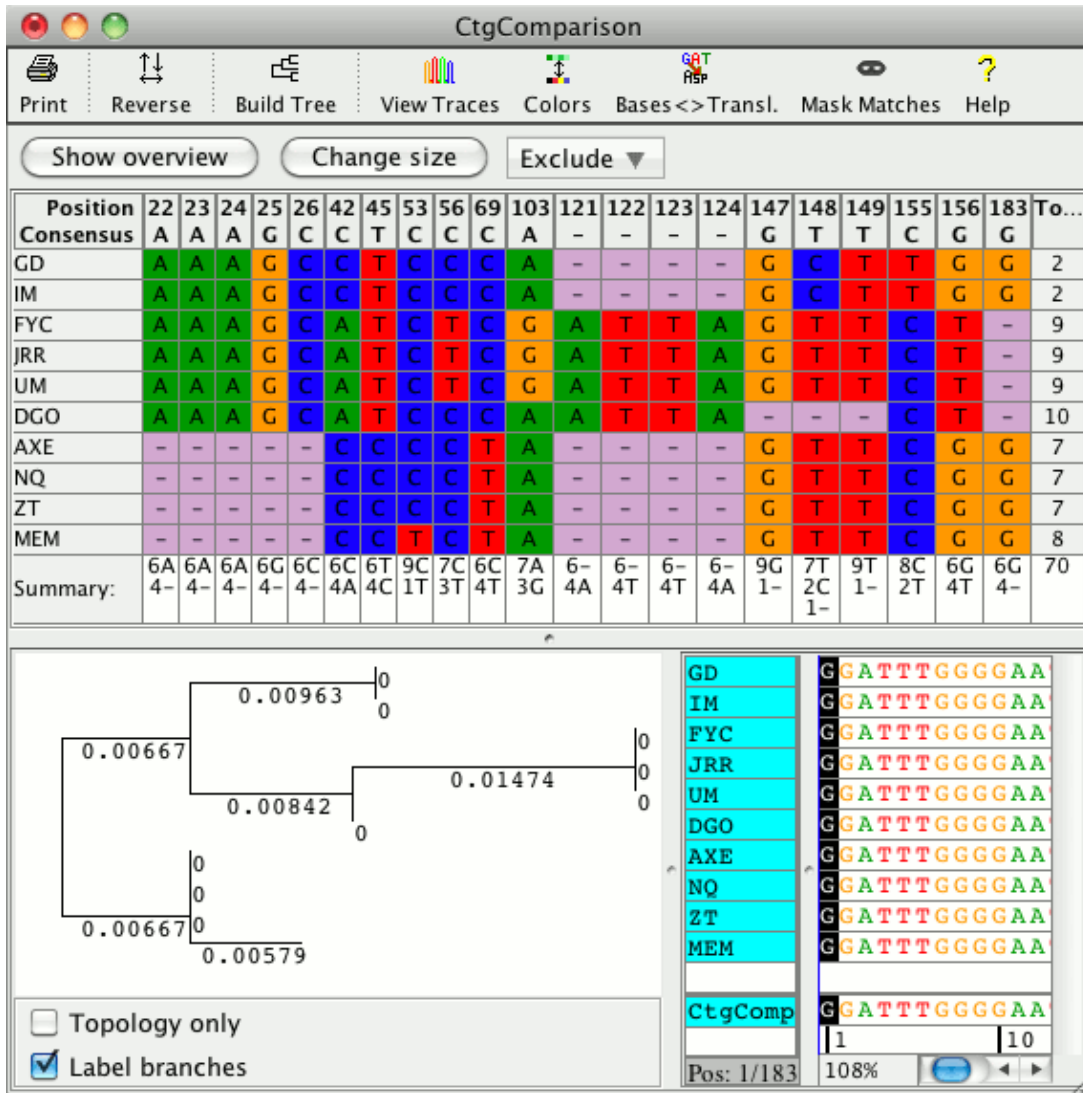
CodonCode Aligner allows you to build a Neighbor-Joining tree with the following options:

1. **Distance Model:** The distance "**Number of Differences**" is the number of sites at which two sequences differ. Please note that we suggest using the complete deletion method with this distance model, since the pairwise deletion method does not normalize the number of differences if you have gaps in your alignment. The "**p-distance**" model uses the proportional distance between different nucleotide sites and the number of compared sites in two samples.
2. **Gaps / Missing Data:** The "**Pairwise Deletion**" method removes all sites that contain gaps or missing data from each sequence pair during the analysis as needed. When selecting "**Complete Deletion**", these sites are removed prior to the analysis from the whole sequence alignment. "**Consider Internal Gaps**" considers all internal gaps as differences and removes only external gaps. Choosing "**Consider All Gaps**" includes all gaps and missing data as differences when building the tree. When including gaps, note that a difference between a gap and a base has the same value as a difference between two non-matching bases.
3. **Display options:** The check box "**Topology only**" ignores the branch lengths when checked, showing only the relationship between samples; or if unchecked, displays a tree where the branch lengths are proportional to the calculated evolutionary time between the sequences. By clicking on the check box "**Label branches**", you can switch between showing and hiding the distance labels on the branches.

By clicking on the "Build Tree" button in the dialog above, you will generate a tree for the currently selected contig. The calculated Neighbor-Joining tree is then displayed in the contig view as shown in the example below:



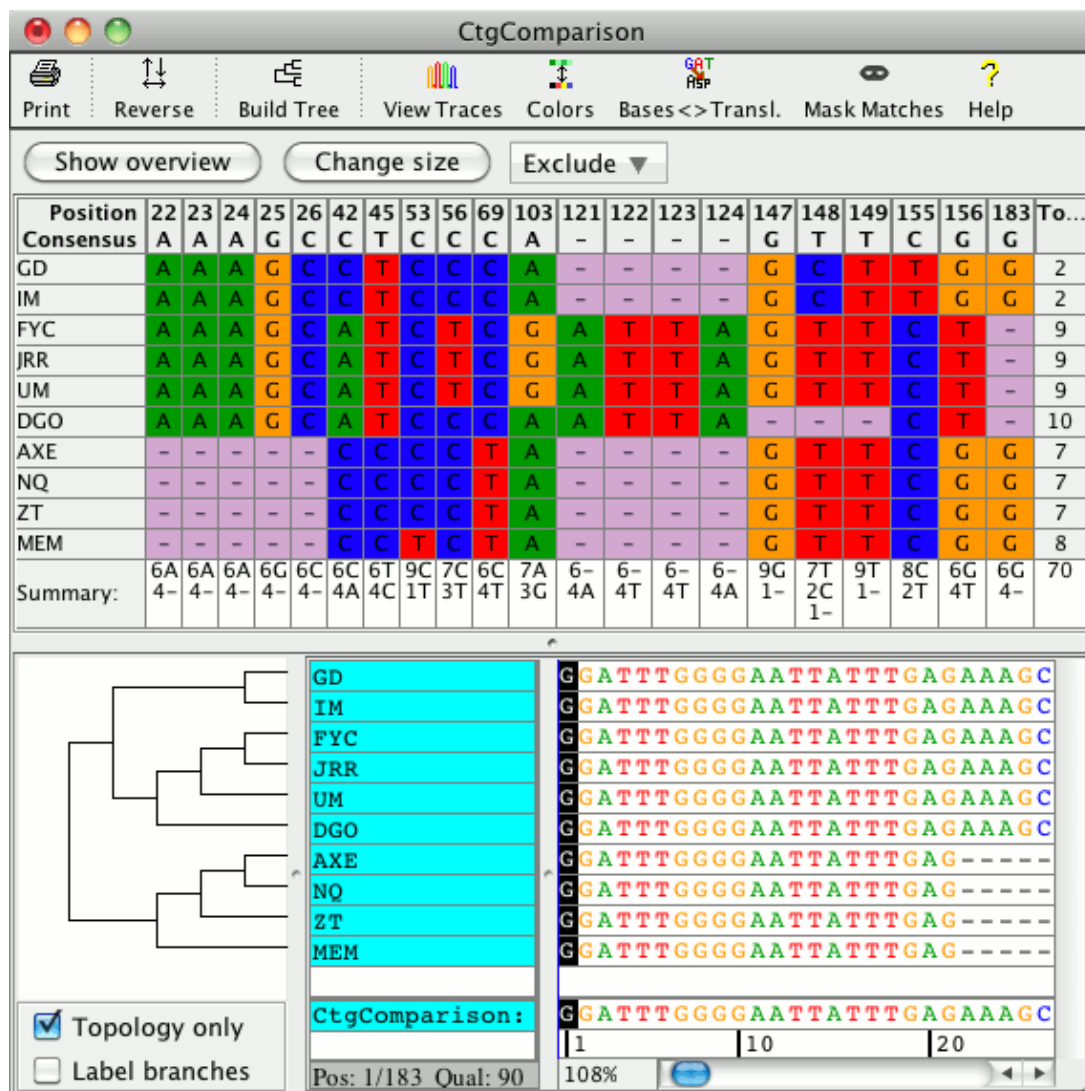
# CodonCode Aligner User Manual



# CodonCode Aligner User Manual

The part of the contig view that is showing the tree can be resized by pressing and holding down the left mouse button on top of the gray bar on the right side of the tree and then dragging the mouse to the left or right.

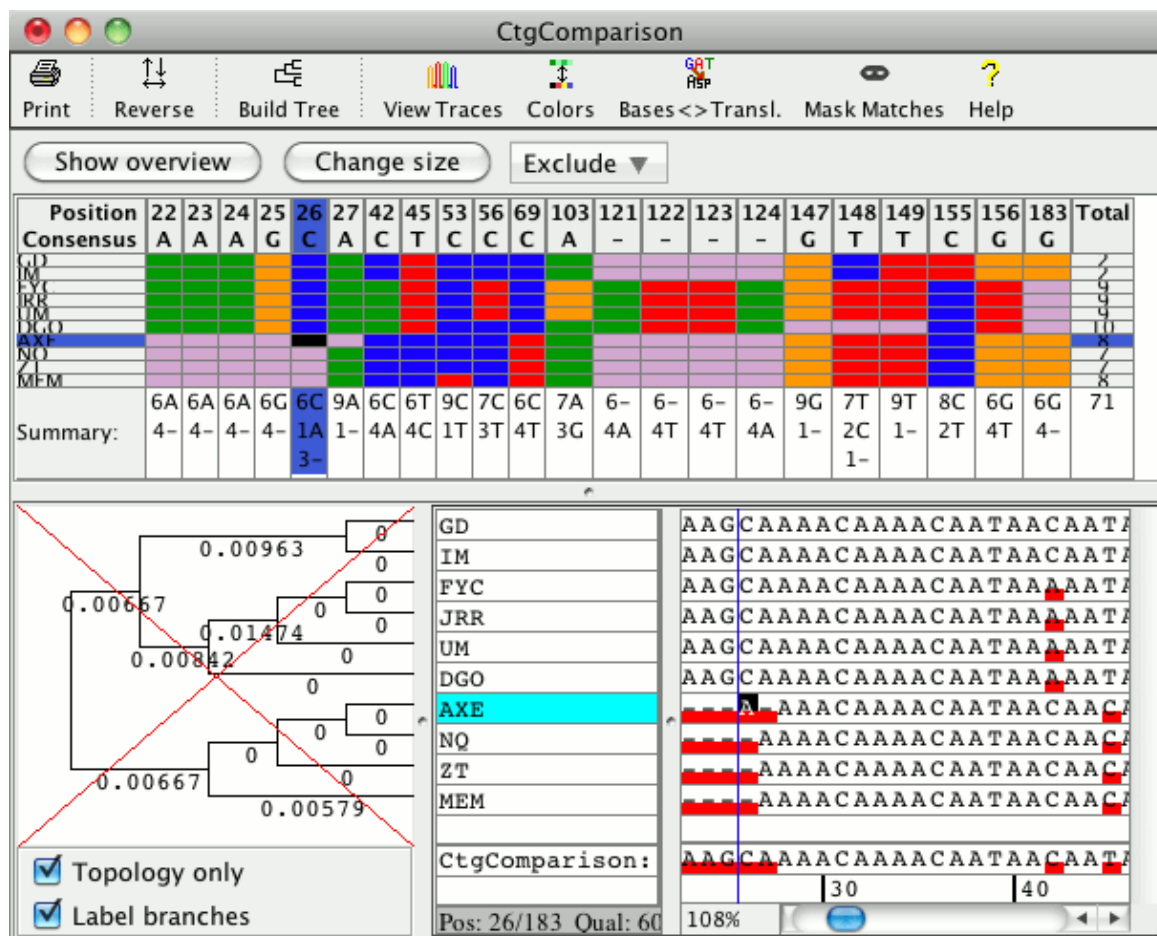
By clicking on the check boxes below the tree, you can switch between showing the tree branch length proportional to the calculated evolutionary time between organisms (as shown above), or the topology only. The distance labels shown on the branches can also be turned on and off. A tree showing only the topology between the organisms could look like this:



## CodonCode Aligner User Manual

Please note that Neighbor-Joining trees can only be built if you have three or more samples in your contig (not counting the reference sequence), and if all samples overlap by at least 20 bases.

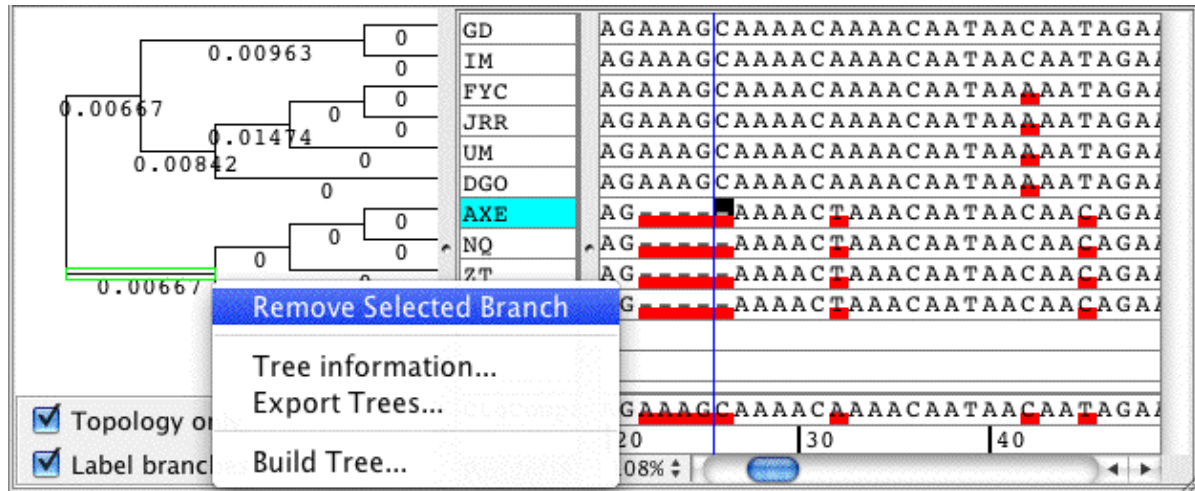
Phylogenetic trees can become **"invalid"** if you change the contig that was used to build the tree. Actions that will invalidate a phylogenetic tree include for example base edits, changing the start or end position of the alignment, moving gaps and deleting bases. An invalid tree is shown with a red cross drawn through it as shown in the example below.



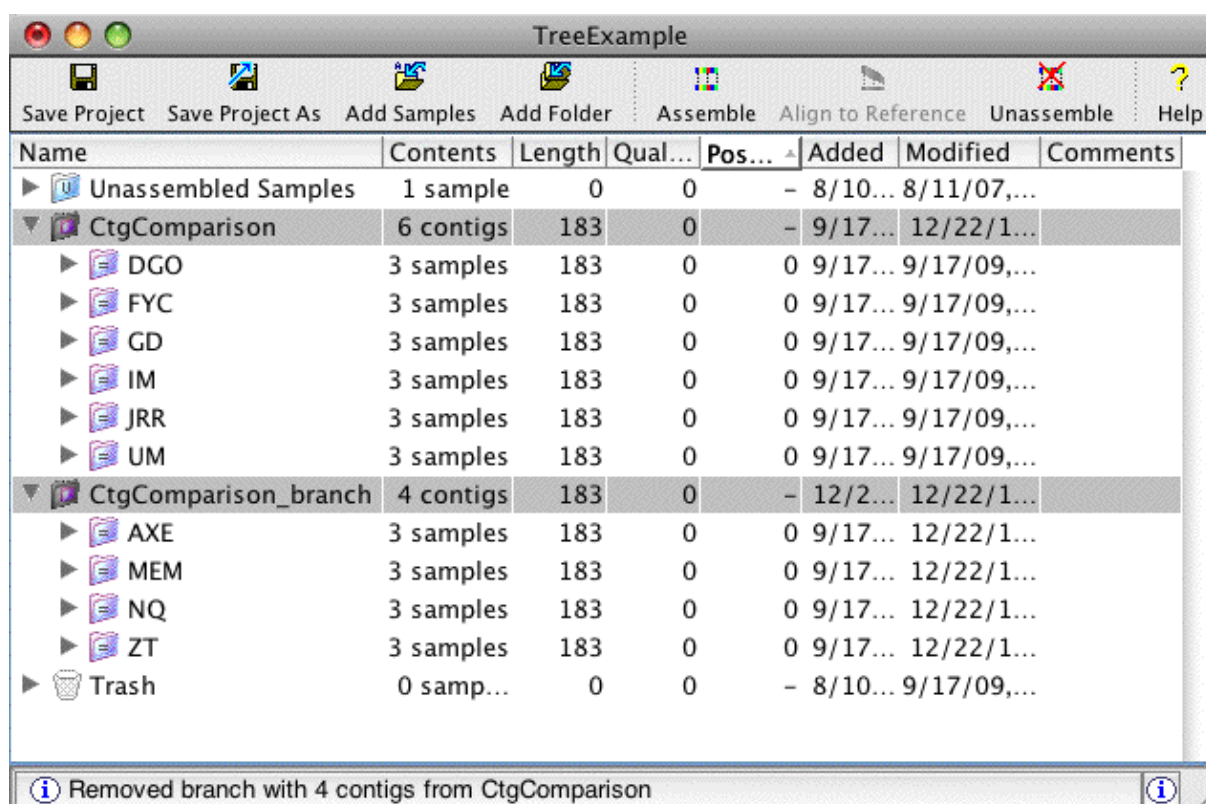
If a tree is invalid can also be seen in its tooltip when moving the mouse over the tree. Building a tree again, will generate a new, valid tree, and delete the old tree.

## Split Contig From Tree

If you have a phylogenetic tree for your contig, you can split the contig using the tree. For example if you want a separate contig for a part of the tree where the samples have many differences compared to the rest, you can use this feature to split the original contig. To split a contig using its phylogenetic tree, right-click on the branch of the tree where you want to split the contig. This will open a popup menu as shown in the example below. From the popup menu choose **"Remove Selected Branch"**.

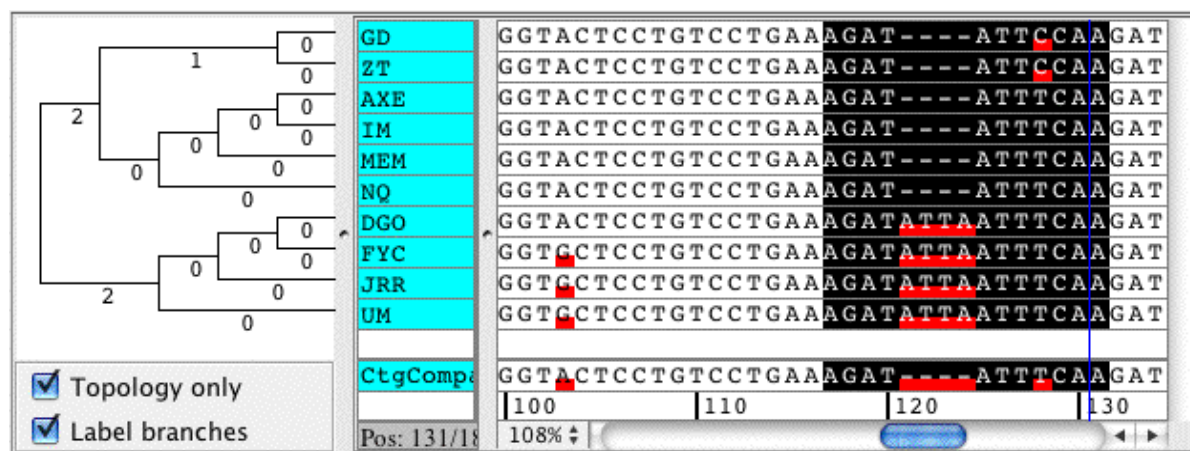


This will split your contig into two contigs. One contig will contain the sequences that are part of the selected tree branch, the other new contig will have all the other sequences. In this example, the bottom four sequences will be moved to the new contig:



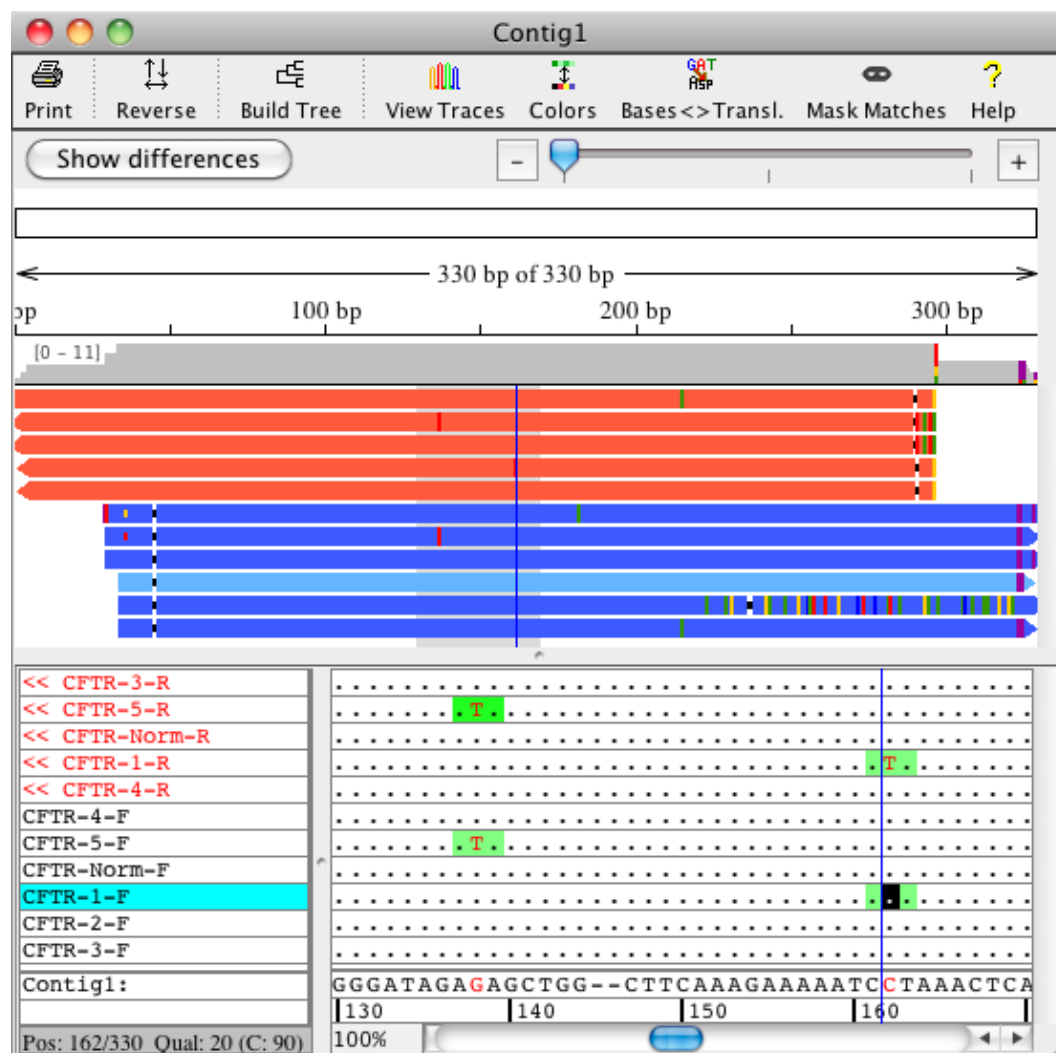
## Building Trees for Selected Bases Only

You can also build a phylogenetic tree for the selected bases only (instead of for the whole contig). Select the bases you want to use in the consensus of the contig view, then choose **"Build Tree for Selected Bases..."** from the **"Contig"** menu. The options dialog that appears is the same as when building a tree for all bases in a contig. The resulting Neighbor-Joining tree takes only the selected bases into account when building the tree. The example below shows a phylogenetic tree built for the selected bases only, that shows the number of differences and was built including all internal gaps as differences:



## Masking Bases Matching the Consensus

One way to focus on bases that differ from the consensus sequence is by going to the "**View**" menu, and selecting "**Mask Bases Matching Consensus**". If checked, any bases in aligned regions that are identical to the consensus sequences are replaced by dots; only bases that differ from the consensus sequence, and the consensus sequence itself, are shown as letters. Here is an example:

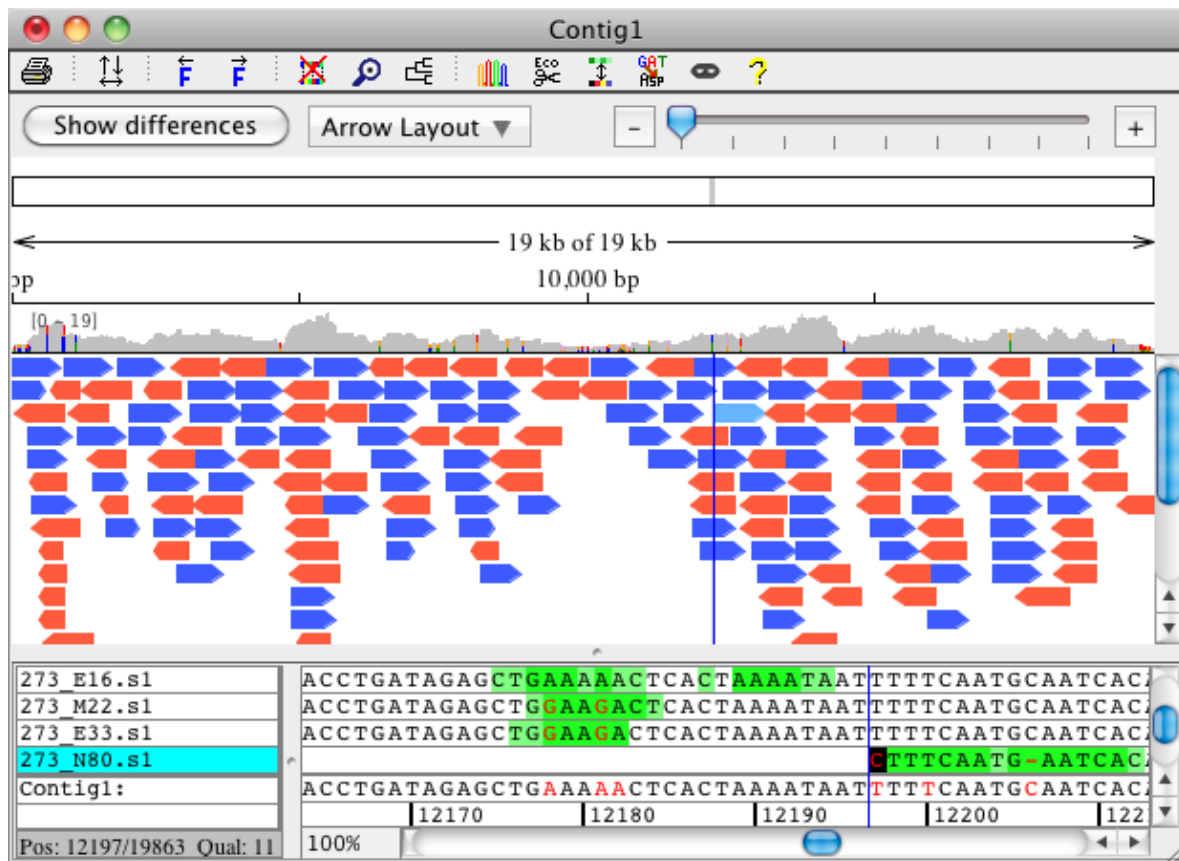


To revert back to the normal display where all bases are drawn, just select "**Mask Bases Matching Consensus**" (or the "Mask Matches" toolbar button) again.

## Zooming in the Contig View

You can zoom in the aligned bases panel at the bottom of the contig view as well as in the overview panel shown at the top.

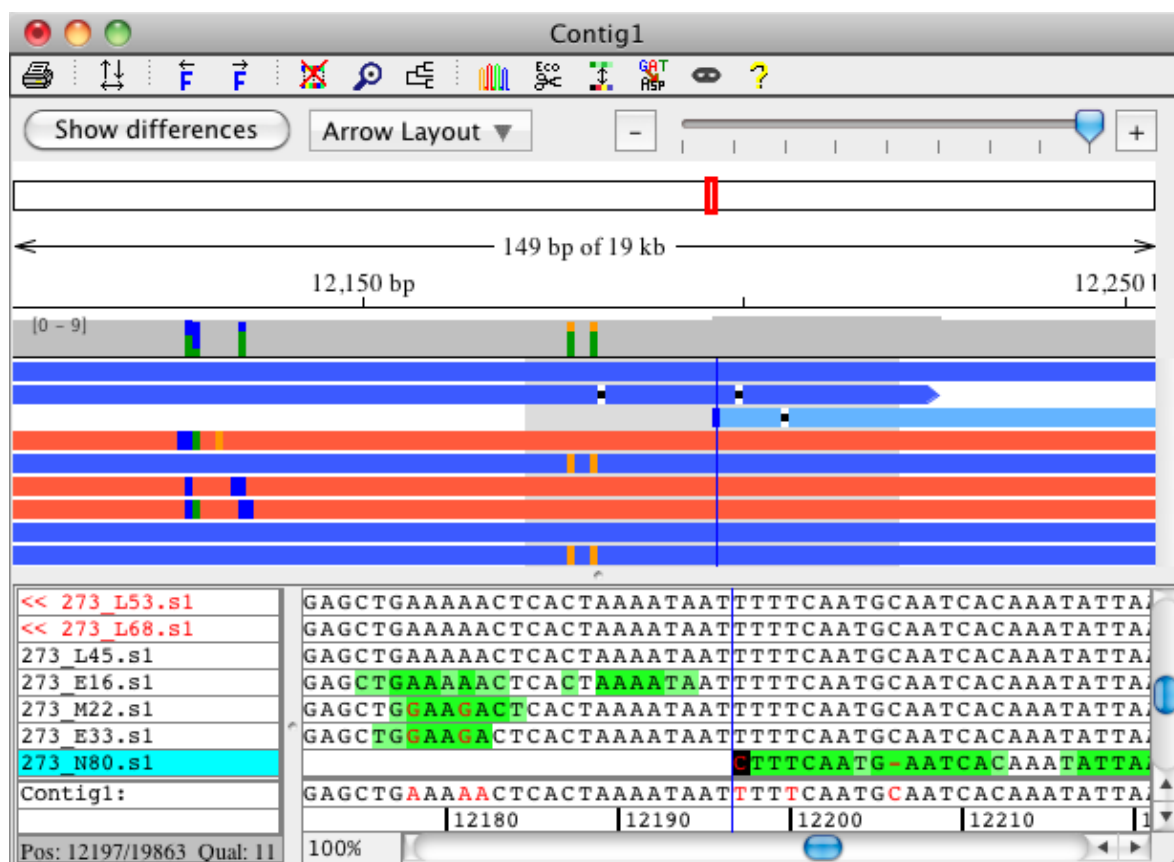
Zooming in the overview panel at the top can be done by using the zoom scale at the top right corner of the contig view. When zoomed out all the way, the overview shows the complete contig as shown in the example below:





## CodonCode Aligner User Manual

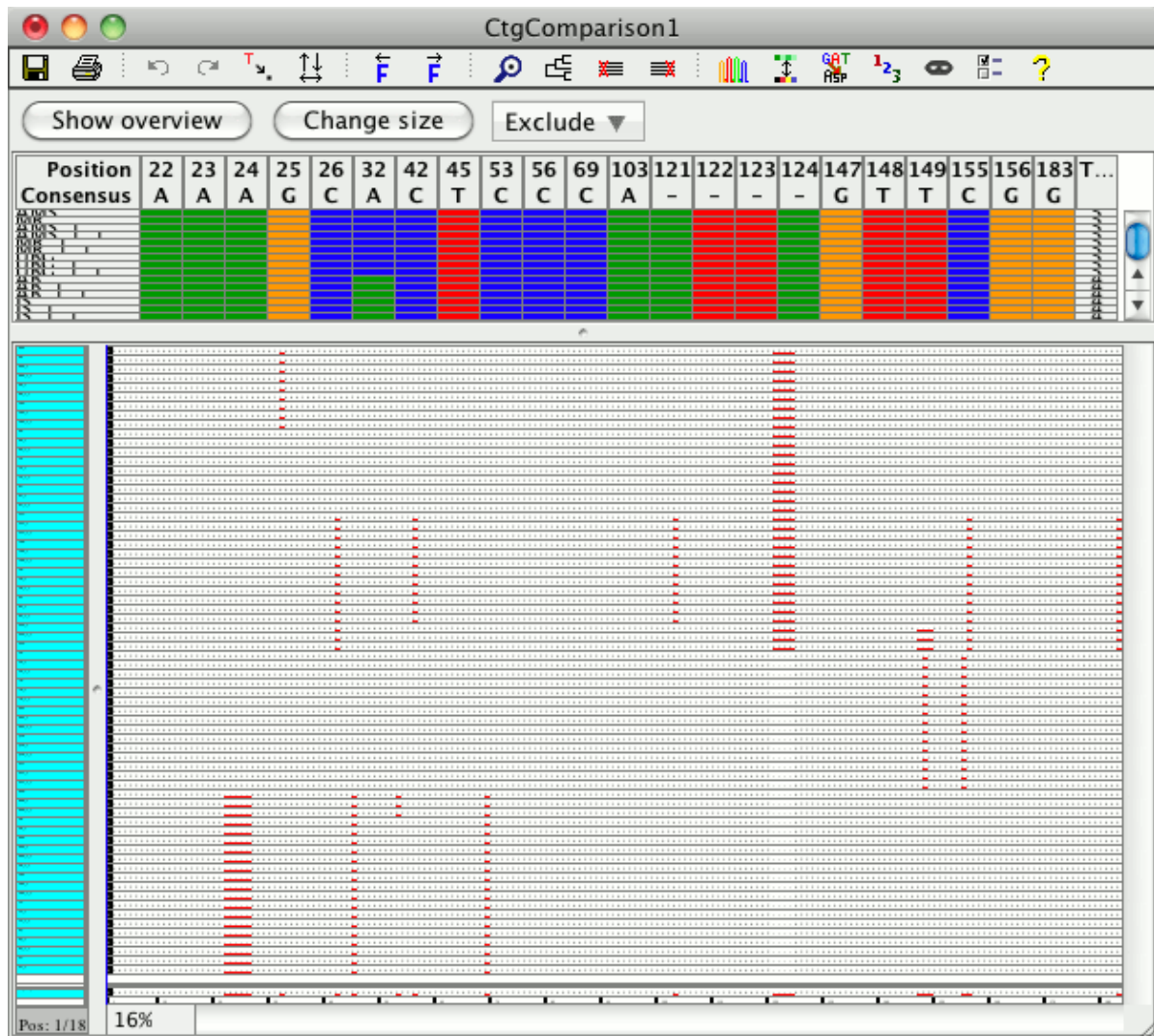
Zooming in allows you to see and navigate to the colored differences in the sample arrows (bases are shown as colored rectangles in the base color, gaps are shown as black box on white background). The red box on the rectangle at the top of the overview highlights which region is displayed as sample arrows. Below that, you can see how many bases are displayed out of the whole contig and which region:





## CodonCode Aligner User Manual

To see regions with few differences more quickly, you can zoom in and out on the aligned bases in the contig view. If you also set your [highlighting preferences](#) to show a box for ambiguities and discrepancies, you can zoom out to easily spot very different and preserved regions. An example like this is shown below:



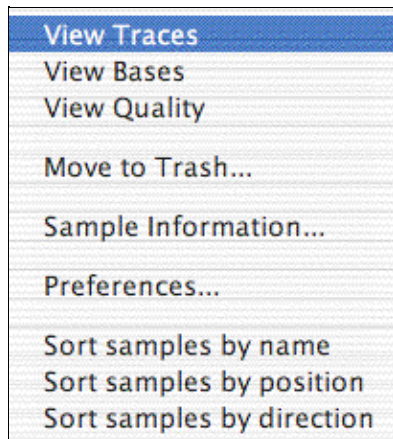
The contig view above uses a zoom level of 16%, which can be set in the bottom left corner of the aligned bases panel in the contig view. You can also use the [view preferences](#) to change the zoom level by setting the font size. The zoom level of 16% is equivalent to a font size of 2.

## Sorting Reads in the Contig View

In the contig view, samples can be sorted in one of several ways:

- by position in the contig (the default)
- by sample name
- by direction (forward or reverse) and then by position in the contig
- by the user

The default sort method for samples in the contig view can be set in the ["View" preferences](#). If you change the sort method in the view preferences, the new setting will affect only contig view windows opened after the change. To change the sorting of reads in an open contig view window, right-click (OS X: command-click) anywhere in the "aligned bases" panel to display a popup menu like this:



Select any of the "Sort samples by ..." options at the bottom of the menu, and the reads in this contig view will be re-sorted.

Typically, sorting samples by position will be the best way to sort samples. For re-sequencing and mutation detection projects, sorting by name may work well if you followed a naming scheme that uses similar names for forward- and reverse reads, so that sorting by name will have the forward- and reverse reads right underneath each other.

You can also **manually** sort the samples by selecting and dragging the sample names up or down. After manually sorting samples in a contig, Aligner will keep your manual sort order for this contig until you select one of the sort options in the popup menu; changing the sorting in the contig view preferences will **not** change the sorting of contigs that have been sorted manually.

Note that the overview panel displaying the samples as arrows only shows the correct sort order if you display your samples in a "stacked" layout. You can change the arrow layout by using the "Arrow Layout" button above the arrow panel, or through the pop-up menu by right-clicking in the arrow panel.

## Automatic Trace Selection

When checking or editing a contig, you will often want to look at a few traces at a given position. CodonCode Aligner can automatically pick traces to display while you are moving around in a contig. You can turn this option on or off by selecting "**Auto Select Traces**" in the "**View**" menu. You can also set the number of traces to be displayed in the "[Views](#)" preferences.

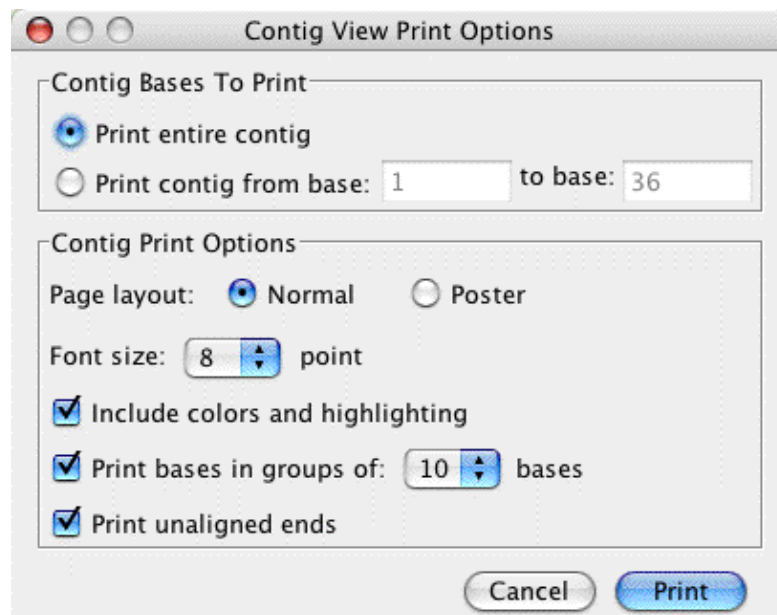
**For automatic trace selection to work, you need to first open both the contig view and the corresponding trace view** (for example by double-clicking in the overview panel in the contig view). Once both views are open, Aligner will automatically add and remove traces when you go to a new location in a contig, for example by clicking in the overview panel or by moving the cursor in the bases panel. The trace view window will grow larger and smaller, depending on the number of traces shown and the available space on your screen. If you want to see more than 2 or 3 traces, you may want to change the height of the trace panels to "Small" or "Tiny" in the "[Views](#)" preferences.

## Printing Contigs

To print a contig:

- open and select the contig view for the contig you want to print
- then, choose "**Print**" from the "**File**" menu (or use the keyboard shortcut Control-P on Windows, Command-P on OS X)

This will show the following option dialog:



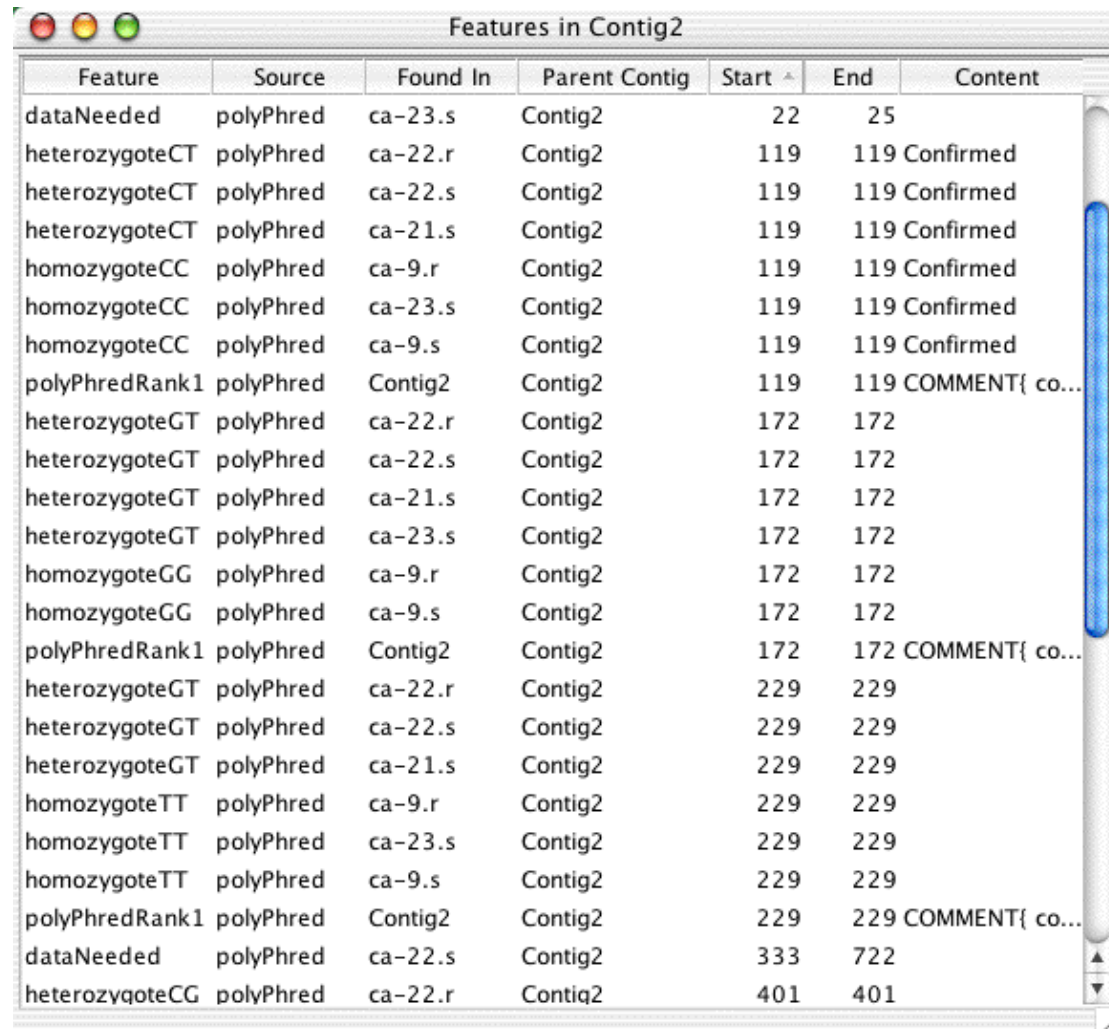
## CodonCode Aligner User Manual

The top section of the dialog lets you choose how much of the contig to print - either the entire contig, or just a section of the contig. Aligner automatically fills in the first and last bases visible in the contig view as the range for printing, but you can edit both numbers to print any section of your contig.

The options in the lower half are described in detail in the [Printing Preferences](#).

# Feature Window

Select a contig in the project view, and then choose **Feature View** from the **View** menu to display a window showing features for a contig:



The screenshot shows a window titled "Features in Contig2" with a table of genomic features. The table has seven columns: Feature, Source, Found In, Parent Contig, Start, End, and Content. The features are listed in rows, with some having additional information in the Content column, such as "Confirmed" or "COMMENT{ co...".

| Feature        | Source    | Found In | Parent Contig | Start | End | Content        |
|----------------|-----------|----------|---------------|-------|-----|----------------|
| dataNeeded     | polyPhred | ca-23.s  | Contig2       | 22    | 25  |                |
| heterozygoteCT | polyPhred | ca-22.r  | Contig2       | 119   | 119 | Confirmed      |
| heterozygoteCT | polyPhred | ca-22.s  | Contig2       | 119   | 119 | Confirmed      |
| heterozygoteCT | polyPhred | ca-21.s  | Contig2       | 119   | 119 | Confirmed      |
| homozygoteCC   | polyPhred | ca-9.r   | Contig2       | 119   | 119 | Confirmed      |
| homozygoteCC   | polyPhred | ca-23.s  | Contig2       | 119   | 119 | Confirmed      |
| homozygoteCC   | polyPhred | ca-9.s   | Contig2       | 119   | 119 | Confirmed      |
| polyPhredRank1 | polyPhred | Contig2  | Contig2       | 119   | 119 | COMMENT{ co... |
| heterozygoteGT | polyPhred | ca-22.r  | Contig2       | 172   | 172 |                |
| heterozygoteGT | polyPhred | ca-22.s  | Contig2       | 172   | 172 |                |
| heterozygoteGT | polyPhred | ca-21.s  | Contig2       | 172   | 172 |                |
| heterozygoteGT | polyPhred | ca-23.s  | Contig2       | 172   | 172 |                |
| homozygoteGG   | polyPhred | ca-9.r   | Contig2       | 172   | 172 |                |
| homozygoteGG   | polyPhred | ca-9.s   | Contig2       | 172   | 172 |                |
| polyPhredRank1 | polyPhred | Contig2  | Contig2       | 172   | 172 | COMMENT{ co... |
| heterozygoteGT | polyPhred | ca-22.r  | Contig2       | 229   | 229 |                |
| heterozygoteGT | polyPhred | ca-22.s  | Contig2       | 229   | 229 |                |
| heterozygoteGT | polyPhred | ca-21.s  | Contig2       | 229   | 229 |                |
| homozygoteTT   | polyPhred | ca-9.r   | Contig2       | 229   | 229 |                |
| homozygoteTT   | polyPhred | ca-23.s  | Contig2       | 229   | 229 |                |
| homozygoteTT   | polyPhred | ca-9.s   | Contig2       | 229   | 229 |                |
| polyPhredRank1 | polyPhred | Contig2  | Contig2       | 229   | 229 | COMMENT{ co... |
| dataNeeded     | polyPhred | ca-22.s  | Contig2       | 333   | 722 |                |
| heterozygoteCG | polyPhred | ca-22.r  | Contig2       | 401   | 401 |                |

A feature view window shows the features for one or more contigs, depending on your selection when you opened the feature view. You can also look at the features for all samples in the "Unassembled Samples" folder by selecting it in the project view, and then opening a feature view.

You can determine which features are shown in the [feature preferences](#). In the example above, low quality consensus regions and tags are shown.

You can **sort** the table by clicking on the column headers. You can also print and export features through the corresponding file menu items (you may have to switch to the project view first, and select the contig or contigs you want to print or export in the project view).

You can **double-click** on any item in the feature view to take a closer look at the feature. This will open the views as defined in the [double-clicking preferences](#) - typically the trace view for the sample, and the contig view for the corresponding contig (unless you have changed the preferences).

The "Start" and "End" numbers are the positions of the feature within the contig (gaps are included in the count). If you changed the numbering of the consensus sequence by using "[Set Base Number...](#)", this will be reflected in the start and end positions.

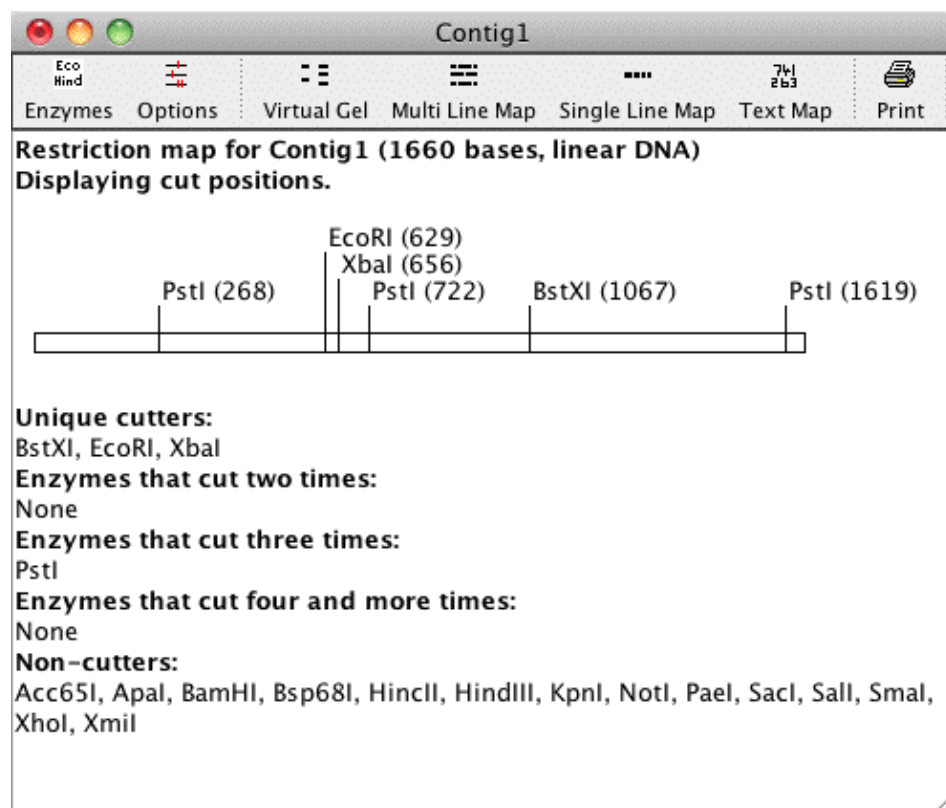
# Restriction Map View

Choose **Restriction Map** from the **View** menu to display a window showing the restriction map for the selected sample or contig.

A restriction map can be displayed in four different map styles: **Single Line Map**, **Multiple Line Map**, **Text Map** and **Virtual Gel**.

## Single Line Map

The **single line map** is shown below:

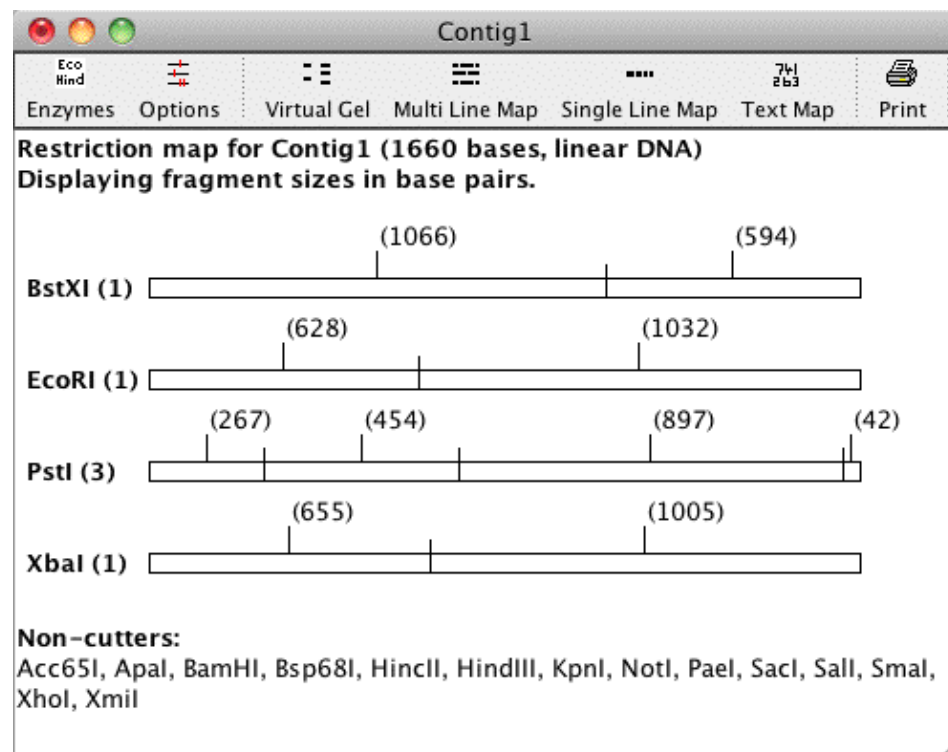


In the example above, you see a single line map for linear DNA that displays **cut positions**. The cut positions are shown behind the enzyme names. A **summary** of the cutters and non-cutters is shown at the bottom of the map.

You can change how the map is displayed and what enzymes to use in the [restriction map preferences](#). You can open the preferences through the icons in the toolbar of the restriction map view.

## Multi Line Map

The **multi line map** for the same example is shown below:



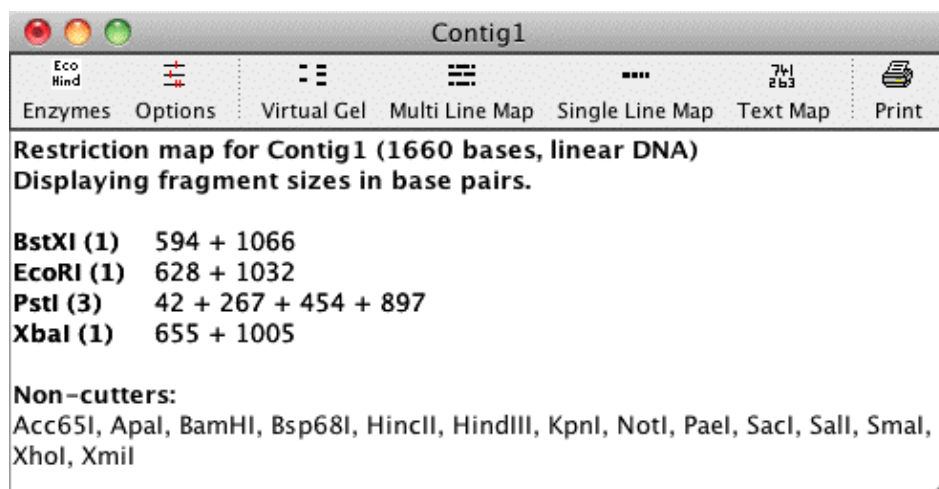


The multiple line map displays a separate graph for each enzyme. The number after the enzyme name tells you how often this enzyme cuts. In the screen shot above, **fragment sizes** (rather than cut positions as in the first screen shot) are displayed. The fragment sizes are displayed in base pairs in the middle of each fragment. For example, BstXI cuts one time, generating two fragments with 1073 bp and 600 bp.

At the bottom of the map a list of all enzymes that do not cut in "Contig1" is given. It is possible to show only cutting enzymes, only non-cutting enzymes, both, or no summary at all. You can choose what to display in the [restriction map preferences](#).

## Text Map

The **text map** for the same example, displaying fragment sizes, is shown below:

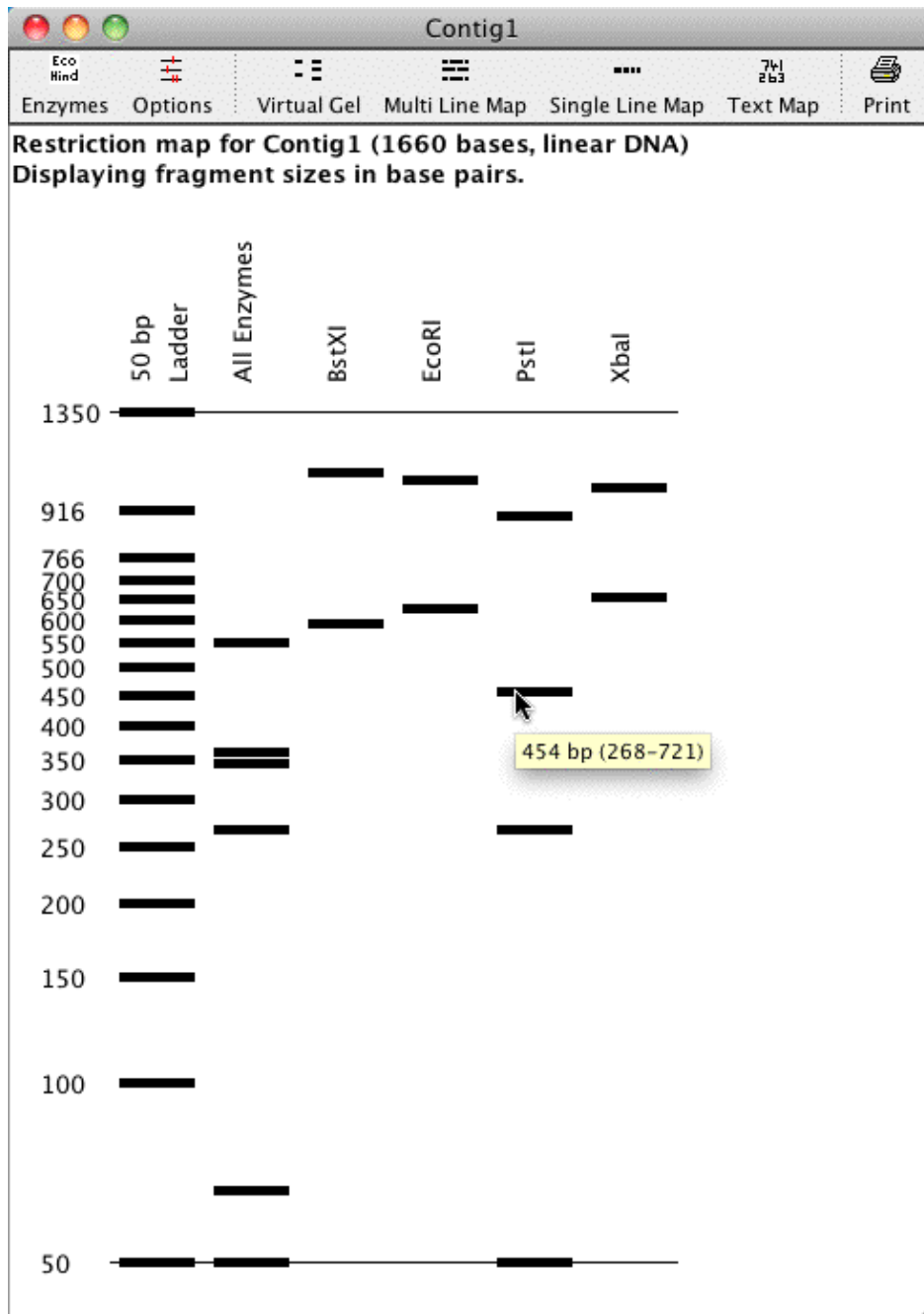


The text map displays the cut results for each enzyme separately like the multi line map. In the screen shot above, you see the fragment sizes generated from the contig by each enzyme.

The restriction maps can be **printed**, and the text map and the summary of each map can be **copied** using keyboard shortcuts. To copy a restriction map, select the part you want to copy and use the keyboard shortcut "Apple + C" on OSX and "Ctrl + C" on Windows to copy, and the keyboard shortcut "Apple + V" on OSX and "Ctrl + V" on Windows to paste.

## Virtual Gel

The **virtual gel** for the same example, but not showing non-cutters, looks like this:

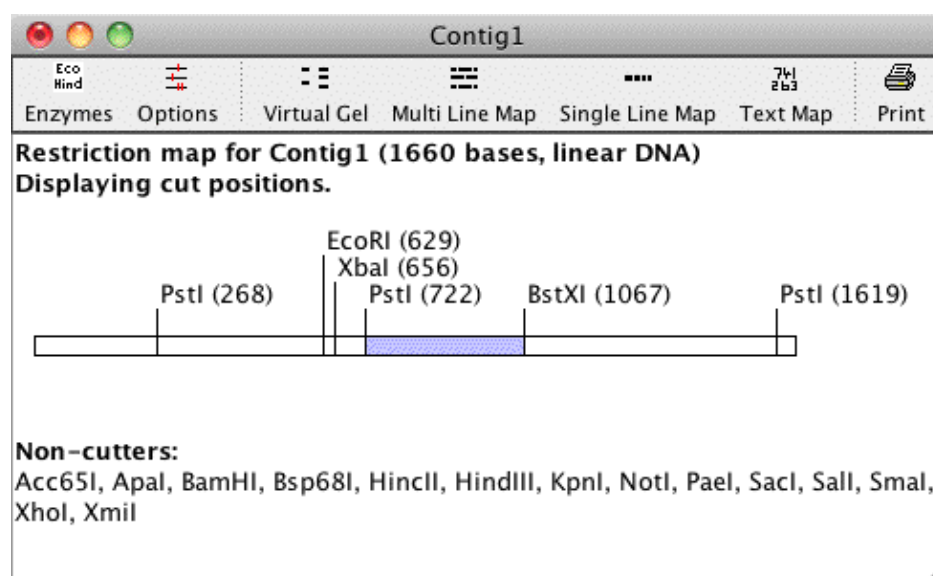


The virtual gel shows the fragments in bp on a gel. The first lane displays the chosen marker. In this example the second lane shows a digest for all enzymes of this sample. The other lanes represent the sample being cut by one enzyme each. Mouse overs show the size, start and end base of the fragment.

The restriction maps can be **printed**, and the text map and the summary of each map can be **copied** using keyboard shortcuts. To copy a restriction map, select the part you want to copy and use the keyboard shortcut "Apple + C" on OSX and "Ctrl + C" on Windows to copy, and the keyboard shortcut "Apple + V" on OSX and "Ctrl + V" on Windows to paste.

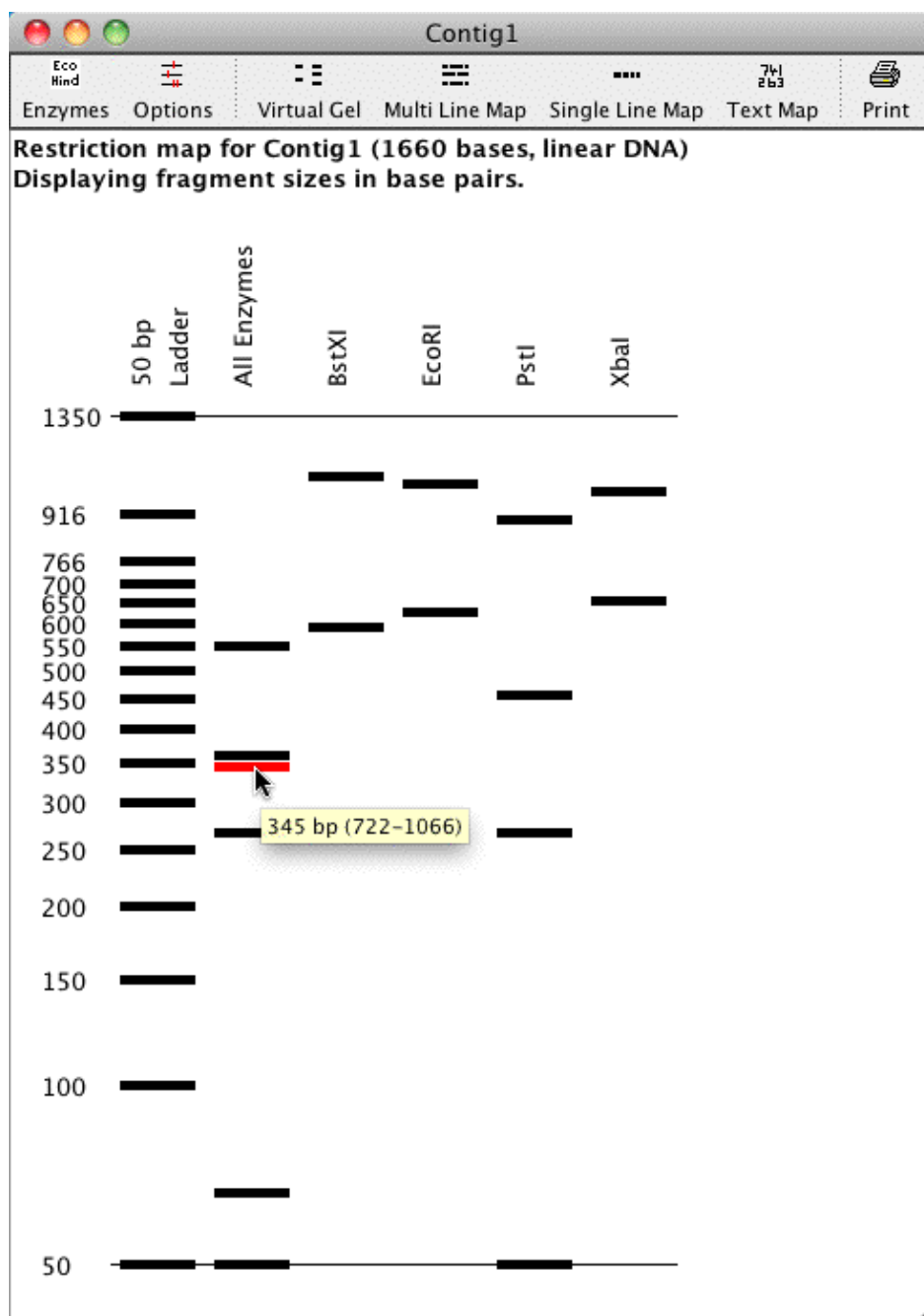
## Selecting Fragments

The graphical restriction map views (single, multi line map, and gel) allow you to **select** part of the sample shown in the map by clicking on a fragment. The selected part of the sample is highlighted in blue in the single and multi line map and in red in the virtual gel:



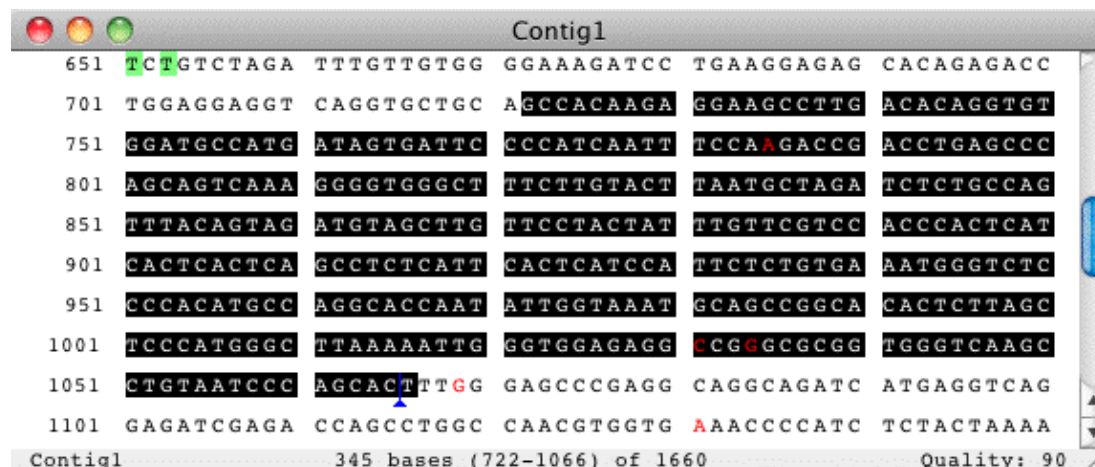
In the screen shot above, the fragment between the two enzymes PstI cutting at position 722 and BstXI cutting at 1067, is selected.

Here is a screen shot of the same fragment being selected in the virtual gel:



The single and multi line map also allow **continuous selections** by holding the Shift key pressed while selecting regions in the view with your mouse.

The **selection updates accordingly in the other views**, like the base view, the contig view and the trace view. The region selected in the screen shot above (from base 722 to base 1066), is now selected in the base view for this contig:



The selection also works the other way round: When you select some bases in the base view, trace view or contig view, they will be highlighted blue in the single and multi line map.

## Restriction Enzymes in Aligner

The restriction enzymes used in Aligner are read in from a file.

This **file** is a text file called "**rebase\_bairoch.txt**" and is located in the folder "Aligner Data" in the "CodonCode Aligner" folder.

The enzymes in this file are listed in a certain format compatible with known formats such as the ones from EMBL, PROSITE and SWISS-PROT.

The file was obtained from REBASE (RestrictionEnzyme dataBASE) [<http://rebase.neb.com>]. The rebase file used is the file in rebase format #19, also called "Bairoch format". The current file and a help file describing its structure can be downloaded from <http://rebase.neb.com/rebase/rebase.f19.html>

Aligner uses only the type II restriction enzymes that have known recognition sites from this file.

You can edit or update the file used by Aligner. However, if you decide to modify the file it is a good idea to first make a copy of it. This way you have a working enzyme file in case something unexpected happens when editing the file. You can open and edit the file with a text editor. When the file is open the version of the file is shown in the first line, and the date of this version a few lines below. To add your own enzymes to the file, please make sure to use the right format.

# Closing Windows

You can close regular Aligner windows, for example the trace view and project view windows, either by clicking on the close button at the top of the window, or by choosing "**Close**" from the "**Window**" menu. Aligner dialog windows, for example the preference window, can be closed by clicking on one of the buttons at the bottom (often labeled "Cancel" and "OK"). If a dialog window is active, you need to close it before you can do anything else.

The option "**Close Others**" in the "**Window**" menu will close all windows except the project view window *and* the currently active window.

If you **close the project view**, Aligner will check if the project has been modified since the last save. If the project has unsaved changes, Aligner will ask you if you want to save the changes first. Closing the project window will also close all open windows that belong to the project.

If you are running Aligner on a Microsoft Windows operating system, you also have a main ("root") window which contains all other windows. Closing the main window will exit Aligner (again, with the option to save unsaved changes in open projects).

On Mac OS X, there is no main window. You have to choose "Quit" in the "Aligner" menu to exit Aligner.

# Scripting CodonCode Aligner

Some functions in CodonCode Aligner can be executed from scripts - text files that contain a collection of commands, and have the extension ".alscpt". Scripts can be opened using "Open..." from the "File" menu, or by drag & drop.

Some commands allow you to specify files or folders, for example for importing. Please note that any file or folder names that contains spaces must be surrounded with quotes. If the file or folder names contain quotes, change the quotes in the names to double quotes. For example, a file `/Users/Shared/Name with "quote"` should be written as `/Users/Shared/Name with ""quote"`.

In scripts, the same logic as when using CodonCode Aligner normally applies. Specifically, to do something like end clipping or assembling, you must first select the samples or contigs to work with. Here is an overview of the available commands:

| <b>File menu</b>            |  |  |
|-----------------------------|--|--|
| <b>Command</b>              | <b>Parameter</b>   | <b>Comments</b>                        |
| newProject                  | <i>none</i>  | Creates a new project                  |
| openProject                 | full path to project file (e.g. <code>"/Users/Shared/My Project/My Project.proj"</code> )  | Opens an existing project              |
| importFolder                | full path to folder  | Imports a folder of samples            |
| importProject               | full path to project file  | Imports a project                      |
| importSample                | full path to sample file   | Imports a sample                       |
| exportConsensus             | <i>Required:</i><br>file=full path to sample file<br><i>Optional:</i><br>exportAll=true or false<br>writeQuals=true or false<br>gapped=true or false<br>format=Fasta or Haplotypes | Exports consensus sequences            |
| closeProject                | <i>none</i>  | Closes a project                       |
| saveProject                 | <i>none</i>  | Saves a project                        |
| saveProjectAs               | full path to project file (e.g. <code>"/Users/Shared/My Project/My Project.proj"</code> )  | Save the project at the given location |
| emptyTrash                  | <i>none</i>  | Empties the trash                      |
| <b>Selecting</b>            |  |  |
| select                      | sample or contig name  | Selects the sample with the given name |
| selectAll                   | <i>none</i>  | Selects everything in the current view |
| selectAllUnassembledSamples | <i>none</i>  | Selects all unassembled samples        |
| selectAllContigs            | <i>none</i>  | Selects all contigs                    |
| unselect                    | sample or contig name  |  |

## CodonCode Aligner User Manual

|                        |  |   |
|------------------------|--|---|
|                        |  | Unselects the sample or contig with the given name  |
| <b>Sample menu</b>     |  |   |
| callBases              | <i>none</i>  | Calls the bases   |
| clipEnds               | <i>none</i>  | End clips the currently selected samples  |
| trimVector             | <i>none</i>  | Vector trims the currently selected samples   |
| makeRefSeq             | sample or contig name  | Makes the named sample a reference sequence   |
| findHeteroIndels       | <i>none</i>  | Finds heterozygous indels in the selected samples   |
| <b>Contig menu</b>     |  |   |
| setAssemblyParameters  | <i>At least one of these:</i><br>algorithm=local or large or end<br>minPercentIdentity=50 .. 100 ( <i>a number between 50 and 100</i> )<br>minOverlapLenth=10 .. 500<br>minScore=10 .. 500<br>maxUnalignedEndOverlap=0.0 .. 100.0<br>bandWidth=10 .. 100<br>wordLength=6 .. 24<br>maxSuccessiveFailures=10 .. 5000<br>matchScore=1 .. 19<br>mismatchPenalty=-1 .. -19<br>gapPenalty=0 .. -19<br>additionalFirstGapPenalty=0 .. -19 | Sets assembly parameters for subsequent assemblies  |
| assemble               | <i>none</i>  | Assembles the selected samples and contigs  |
| assembleFromScratch    | <i>none</i>  | Assembles from scratch  |
| assembleByName         | <i>none</i>  | Assembles in groups (by name)   |
| compareContigs         | Clustal or muscle (optional)   | Compares contigs to each other, using Clustal or muscle (if specified), otherwise the current default |
| assembleUsingPhrap     | <i>none</i>  | Assembles contigs with PHRAP  |
| setAlignmentParameters | <i>At least one of these:</i><br>algorithm=local or large or end<br>minPercentIdentity=50 .. 100 ( <i>a number between 50 and 100</i> )<br>minOverlapLenth=10 .. 500<br>minScore=10 .. 500<br>maxUnalignedEndOverlap=0.0 .. 100.0<br>bandWidth=10 .. 100<br>wordLength=6 .. 24   | Sets alignment parameters for subsequent alignments to a reference sequence                           |



## CodonCode Aligner User Manual

|                              |  |   |
|------------------------------|--|---|
|                              | matchScore=1 .. 19<br>mismatchPenalty=-1 .. -19<br>gapPenalty=0 .. -19<br>additionalFirstGapPenalty=0 .. -19 |   |
| alignToReference             | <i>none</i>  | Aligns selected samples to a reference sequence                     |
| alignByName                  | <i>none</i>  | Aligns to reference sequence by name (in groups)                    |
| findMutations                | <i>none</i>  | Finds mutations in the selected contigs                             |
| buildTree                    | <i>Optional:</i><br>distanceMethod=number or p-distance<br>pairwise=true or false                            | Build a phylogenetic tree for the selected contig                   |
| <b>Editing &amp; Go menu</b> |  |   |
| gotoBase                     | Base number  | Move to the given base  |
| search                       | Sequence to find   | Find the given sequence   |
| deleteFromContigStart        | <i>none</i>  | Deletes all bases from the contig start to the current base         |
| deleteToContigEnd            | <i>none</i>  | Deleted all bases from the current position to the contig end       |
| callSecondaryPeaks           | threshold=1..99  | Call secondary peaks  |
| changeLowQualToN             | threshold=0..90  | Change low quality bases to N                                       |
| changeAmbigsToSingleBases    | <i>none</i>  | Change ambiguities to single bases                                  |
| <b>View menu</b>             |  |   |
| openBaseView                 | sample or contig name  | Open a base view  |
| openContigView               | contig name  | Open a contig view for the given contig                             |
| openTraceView                | sample name  | Open a trace view for the given sample                              |
| hideScriptWindow             | <i>none</i>  | Hide the script window  |
| <b>Other commands</b>        |  |   |
| setComments                  | any text to be used as comment<br>(leave empty to erase existing comments)                                   | Sets the comments for the currently selected samples and/or contigs |
| showMessage                  | message text   | Show the given message  |
| logProgress                  | full path to file where progress messages should be logged (e.g. "/Users/Shared/Script_progress.txt")        | Log progress messages to the given file                             |
| abortOnError                 | <i>Optional:</i><br>true (default) or false  | Set whether to abort scripts when errors occur                      |

Note that most script commands require a selection that is valid for the command; some commands also require that the appropriate view is open (for example the contig view for "deleteFromContigStart").

## CodonCode Aligner User Manual

Scripts can be added to the "Scripts" menu in CodonCode Aligner by putting them in certain directories:

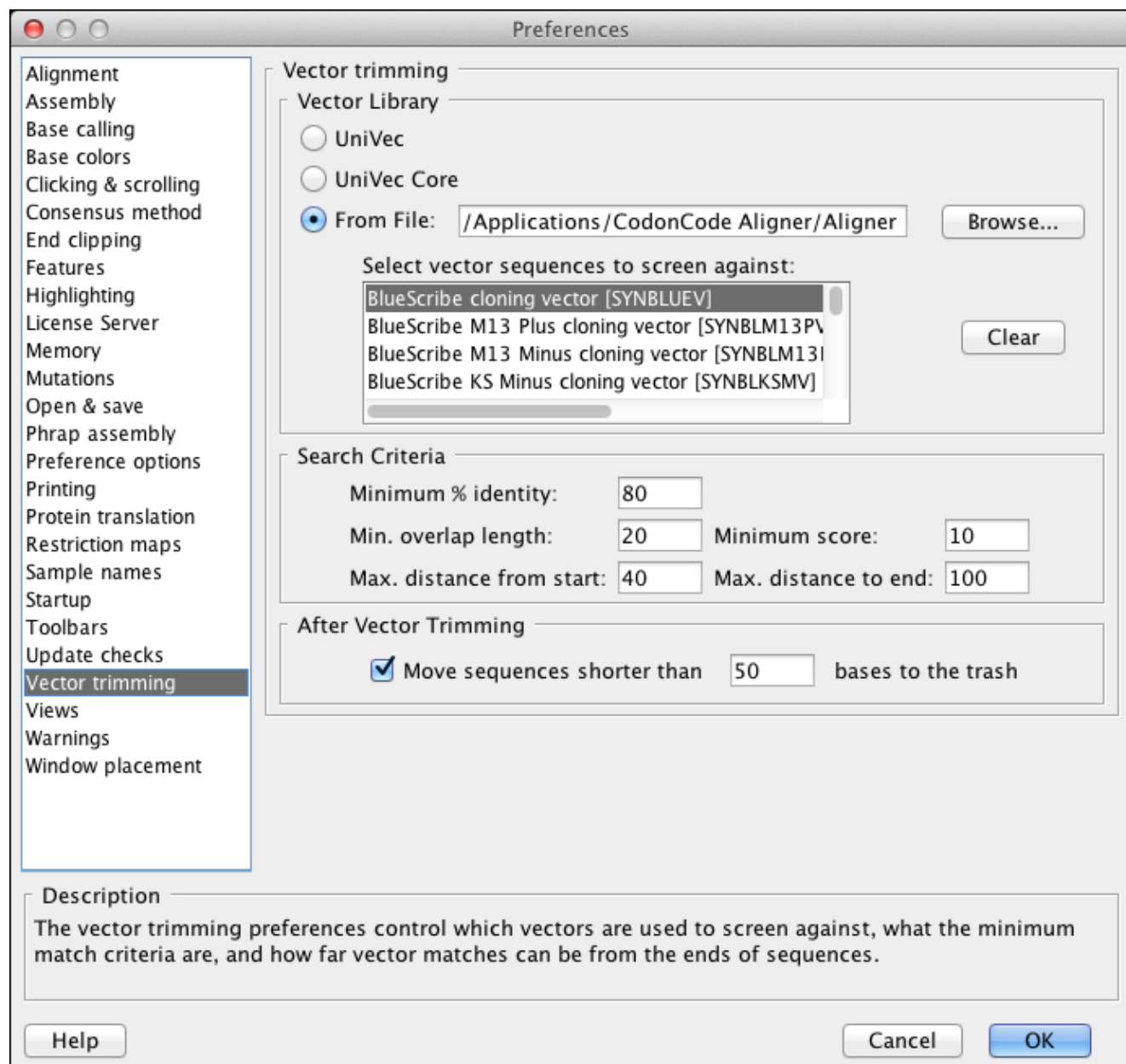
- On **OS X**, in "/Library/Scripts/CodonCode Aligner/" (for all users) and in "~/Library/Scripts/CodonCode Aligner/" (the Library folder in your home folder, for "private" scripts)
- On **Windows**, in a "CodonCode Aligner Scripts" folder in the "My Documents" or the "Shared Documents" folder.

# Preferences and Settings

You can customize many aspects of Aligner in the "Preferences" dialog. To display the Preference dialog:

- On Mac OS X, select "**P**references" in the "**C**odon**C**ode **A**ligner" menu, or press Option-return
- On Windows, select "**P**references" in the "**E**dit" menu, or press Alt-Enter

This will show the preference window, which looks like this:



## CodonCode Aligner User Manual

On the left side, you choose which specific preferences you want to edit. In the image above, "Vector trimming" is selected. The right side changes according to the selection on the left side. You can edit your settings on the left. Any changes you make are only saved when you press "OK" (or hit the return key). If you click "Cancel" (or press the escape key), no changes to the settings will be made.

If you make any invalid entries in one of the fields, Aligner will show a warning that indicates the problem, and reset the field to the previous value.

By default, the Preferences are specific for each user. However, Aligner offers the option to share preferences between different users; detailed instructions are available on the ["Preference options" help page](#).

# Alignment Preferences

You can specify parameters for alignments to reference sequences in the Alignment Preferences:

**Preferences**

**Alignment**

Algorithm: End to end alignments

Min. percent identity: 80.0

Min. overlap length: 30

Min. score: 20

Max. unaligned end overlap: 90.0

Bandwidth (max. gap size): 30

Word length: 8

Match score: 1

Mismatch penalty: -2

Gap penalty: -2

Additional first gap penalty: -3

Uncovered reference sequence: Leave as is

☒ Reverse-complement for external programs as needed

Defaults

**Description**

The alignment preferences control settings for alignments to reference sequences (except for 'Reverse-complement for external programs as needed', which affects alignments with Clustal, Muscle, or MACSE).

Help Cancel OK

During the alignment process, the sequences are aligned successively against the reference sequence. Each alignment is evaluated against the match criteria defined in the alignment preferences. Only if the alignment meets all the criteria, the sample will be added. If the alignment does not match any one of the criteria, it will not be added, and remain in the "Unassembled Samples" folder. The parameters are similar to the parameters for sequence assembly, but you can assign different values for assembly and alignment parameters. The meaning of each parameter is discussed in the next sections.

Please note that the alignment parameters will **not** be used when comparing contigs (from the "Assemble with Options..." dialog).

## Algorithm

The algorithm pulldown lets you choose how CodonCode Aligner compares sequences during alignment, with the following options:

- **Local alignments:** When this algorithm is used, Aligner uses local alignments. This means the start and the end of sequences is not necessarily included in the alignment - the alignments stop when the alignment score would not improve anymore. This can be due to (for example) too many discrepancies, or unremoved vector sequences. The resulting unaligned ("dangling") ends are shown on gray background in the contig view, base view, and trace view.  
*This was the default algorithm for CodonCode Aligner version 1.6.3 and older.*
- **Large gap alignments:** This algorithm is typically used when aligning cDNA to genomic DNA. It allows for large gaps in between alignments, without penalizing the large gaps. The large gap algorithm can also be useful when analyzing samples with large insertions or deletions.
- **End to end alignment:** When this algorithm is used, alignments always include the entire sequences (*this method is the default algorithm in other programs, for example Sequencher*). When using this algorithm, it is important that samples have been end clipped and vector trimmed.  
*This is the default algorithm for CodonCode Aligner version 2.0.1 and newer. However, if you used older versions of CodonCode Aligner before, you may need to manually select this algorithm to use it.*

## Minimum Percent Identity

This is the minimum percentage of identical bases in the aligned region. The default parameter of 80% is relatively relaxed; you may want to use a more stringent setting for your projects, especially if you did use end clipping before the alignment.

Be careful about setting this value to the 100%: only samples that fully agree with the reference sequence will be aligned, samples with even a single discrepancy will not be aligned.

## Minimum Overlap Length

This is the minimum length of the aligned region. If the aligned region is shorter than the value you set here (with 30 being the default), alignments will be rejected, and samples will remain in the "Unassembled Samples" folder.

## Minimum Alignment Score

This parameter is similar to the "Minimum Overlap Length", but it takes discrepancies into account. Scores will be scaled so that a match gives a score of 1 - for each matching base in the aligned region, a score of +1

will be added. With the default settings, a score of -2 will be subtracted for each mismatch; for single base insertions or deletions, a score of -5 will be subtracted (-3 gap introduction penalty and -2 gap penalty; additional gaps in the same run lead to a subtraction of -2 per base).

In general, your minimum alignment score should be lower than your minimum overlap length to allow for some level of discrepancies between the sequences.

## Maximum Unaligned End Overlap

*This parameter is **only** effective when the "Local Alignments" algorithm is used.* After doing an alignment, Aligner looks at the unaligned ("dangling") ends of both reads. Since Aligner does local alignments, two aligned sequences may have unaligned bases at the same end. This will happen, for example, when aligning two different copies of a repeat sequence, or if one of the samples is a chimeric clone. It can also happen if the sequence of one or both clones has a very high error rate, as typically seen towards the end of reads that have not been end clipped.

To illustrate how this parameter is calculated, consider the following diagram:

|   |
|---|
| 123456789012345678901234567890              |
| tctctctctcAGCGATCAATaaaaatttt               |
|   |
| gggggAGCGATCAATggggccggggccggggccgg         |
| 12345                  12345678901234512345 |

The sequence at the top has 10 unaligned bases at the start and end, and 10 aligned bases in the middle. The bottom sequence has 5 unaligned bases at the start and 20 unaligned bases at the end. Between the two sequences, there are 15 additional bases that could possibly have been aligned - the 5 unaligned bases at the start of the bottom sequence, and the 10 bases at the start of the top sequence.

Aligner calculated the relative amount of unaligned sequence that could have been aligned by dividing the overlapping bases in the unaligned ends by the length of the shorter sequence. In our example, this is 15 / 30 (the length of the top sequence), or 50%. With the default setting of 70% for the "Maximum Unaligned End Overlap", our example would have passed, at least for this parameter.

You may need to adjust this value, depending on the kind of project you are doing. If you aligned cDNA sequences to genomic DNA, use values of or near 100%, since large stretches of exons may be unaligned. But if you expect your samples to match end-to-end, and pre-process your sequences with end clipping and vector trimming, you can use lower values to reduce the chance that different copies of repeats will be incorrectly assembled together.

## Bandwidth (Maximum Gap Size)

The bandwidth parameter allows you to set the half width of the diagonal used during the banded alignment. This has an effect on the maximum size of gaps (insertions or deletions in one sample) that can still be aligned. A bit simplified, if one sample has an insertion or deletion that is larger than the "bandwidth" number, the alignment will typically stop at the insertion/deletion, and the rest of the sample will be unaligned. If the insertion or deletion is shorter than the bandwidth, the alignment will continue after introducing the necessary number or gaps in one sequence, as long as the aligned parts after the gaps is long enough (the aligned regions before and after the gaps must be at least 1 and 1/2 times as long as the number of gaps, and longer for any mismatches or ambiguities).

The discussion above is a bit simplified - in reality, what counts is the total number of gaps in one sequence, minus the total number of gaps in the other sequence, at any position, since the alignment uses a banded Needleman-Wunsch algorithm.

The bandwidth parameter has an impact on the alignment speed - larger values mean slower alignments. For large projects, you may want to reduce the bandwidth value; for projects where you know that you have larger insertions and deletions, you may want to increase it. Note, however, that increasing the bandwidth will typically not be enough to extend alignments through very large gaps, like large introns. Support for "Large gap alignments" will be added to later versions of Aligner.

*The bandwidth parameter does not apply to large gap alignments; in large gap alignments, gaps between aligned parts can extend for thousands of bases.*

## Word Length

The "word length" parameter determines the size of "words" that CodonCode Aligner uses when looking for potential overlaps between sequences. Only sequence pairs that have perfect matches of at least this length will be considered for merging.

If you are trying to align sequences with high error or mutation rates, reducing the word length may help to get samples aligned. For very large projects or projects with many repeat sequences, larger numbers may give better results.

The impact of the word length setting on the alignment speed depends on the size of the project; for large projects, larger word length values can lead to faster alignments.

## Match scoring

The next four alignment parameters determine how matches are scored. The "Match score" is used when two aligned nucleotides are identical; the "Mismatch penalty" when two base calls are different. The "Gap penalty" and the "Additional first gap penalty" is used when one of the two sequences has a deletion relative to the other sequence. For single base deletions, the penalty score will be the sum of the gap penalty and the additional first gap penalty; for additional deleted bases (multiple gaps in a row), the penalty will be just the gap penalty.

You can change the scoring within limits (scores from 1 to 19 for matches, and penalties of -1 to -19). In general, we suggest that only experts change the match scores and penalties.

## Clipping Uncovered Regions

When working with large reference sequences, for example genomic sequences for multi-exon genes, CodonCode Aligner can automatically trim the alignment to the covered region. For example, you may use "Align in Groups" to generate separate contigs for each of the exons for a gene with many exons, where the reference sequence may be 100,000 bases long. Clipping uncovered regions can reduce the size of each exon alignment to just the size of the exon, optionally leaving 50, 100, or 250 bases additional sequence on each side.

Clipping is done after alignment based on (a) coverage by samples other than the reference sequence, and (b) coding sequence annotation of the reference sequence. For example, when clipping to +100 bases, and all



your sample align inside the coding region, the resulting contig will be clipped so that 100 bases before the start and after the exon of the exon region remain. If the alignment of samples extends to before the start of the coding sequence, then the clipping will be moved accordingly, leaving 100 bases before the first aligned base.

If the clipped reference sequence contains multiple exons and a valid "codonStart" tag, the "codonStart" tag will be moved as needed. For example, if the first two exons are removed when clipping, a "codonStart" tag will be added to the third exon. It will be placed on the first, second, or third base, depending on where the first complete codon starts; and it will contain the correct base number for this base, so that the mutation annotation after using "Find mutations" will be correct.

If you do not want to clip alignments, select "Leave as is" from the pulldown menu.

## Reverse-complement for external programs

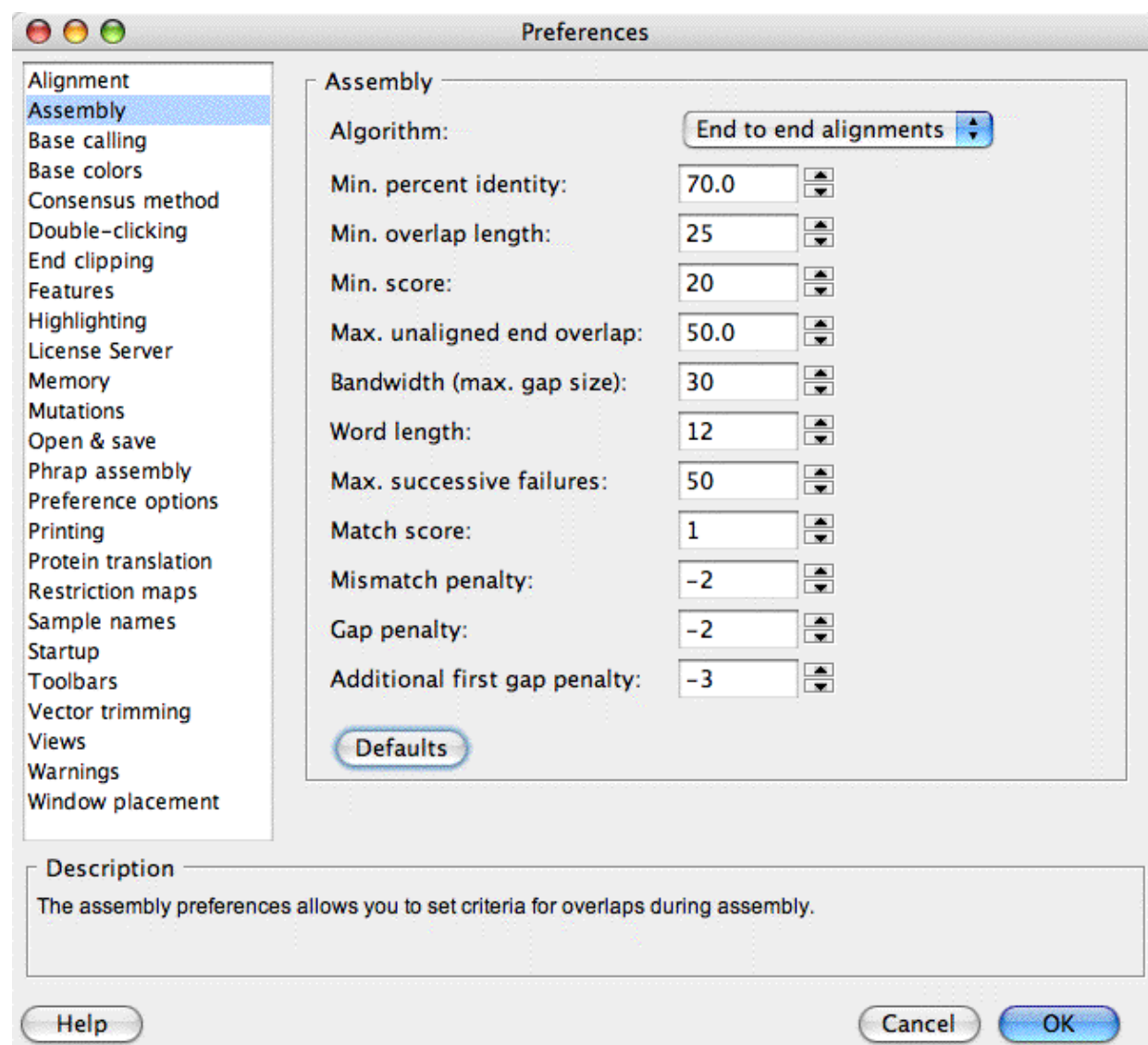
This parameter applies *only* to alignments generated with external programs like Clustal, Muscle, and MACSE when the "Align with Clustal", "Align with Muscle", "Align using translations", or "Compare contigs" menu options are used. If selected, CodonCode Aligner will examine all selected sequences before alignment, and reverse-complement sequences as needed so that all sequences are oriented in the same direction. The determination whether or not sequences should be reverse-complemented is based on a shared-word count with the longest sequence in the selection, and therefore may occasionally be incorrect. If this happens, unselected the "Reverse-complement for external programs as needed" checkbox, and use the "Reverse-complement" option from the "Edit" menu to manually reverse-complement any sequences that are in the "wrong" orientation.

## Restoring default parameters

To restore the default parameters, click on the "Defaults" button near the bottom. This will reset all parameters to the choices shown in the screen shot above.

# Assembly Preferences

The assembly preferences allow you to specify parameters for sequence assembly:



During the assembly process, the sequences or contigs are aligned successively against each other. Each alignment is evaluated against the match criteria defined in the alignment preferences. Only if the alignment meets all the criteria, the samples or contigs will be merged. If the alignment does not match any one of the criteria, the merger will be rejected. The parameters are similar to the parameters for sequence alignment, but you can assign different values for assembly and alignment parameters. The meaning of each parameter is discussed in the next sections.

Please note that the assembly parameters **will not be used when comparing contigs with ClustalW or muscle**, or when assembling with Phrap. When comparing contigs, the assembly preferences will only be used when the built-in algorithm is selected in the "Algorithm" panel of the "Assemble with Options" dialog.

## Algorithm

The algorithm pulldown lets you choose how CodonCode Aligner compares sequences during assembly, with the following options:

- **Local alignments:** When this algorithm is used, Aligner uses local alignments (*this method is also used when assembling using Phrap*). This means the start and the end of sequences is not necessarily included in the alignment - the alignments stop when the alignment score would not improve anymore. This can be due to (for example) too many discrepancies, or unremoved vector sequences. The resulting unaligned ("dangling") ends are shown on gray background in the contig view, base view, and trace view.  
*This was the default algorithm for CodonCode Aligner version 1.6.3 and older.*
- **Large gap alignments:** This algorithm is typically used when aligning cDNA to genomic DNA. It allows for large gaps in between alignments, without penalizing the large gaps. The large gap algorithm can also be useful when analyzing samples with large insertions or deletions.
- **End to end alignment:** When this algorithm is used, alignments always include the entire sequences. When using this algorithm, it is important that samples have been end clipped (and possibly also vector trimmed).  
*This is the default algorithm for CodonCode Aligner version 2.0.1 and newer. However, if you used older versions of CodonCode Aligner before, you may need to manually select this algorithm to use it.*

## Minimum Percent Identity

This is the minimum percentage of identical bases in the aligned region. The default parameter of 70% is relatively relaxed; you may want to use a more stringent setting for your projects, especially if you did use end clipping before the alignment.

Be careful about setting this value to 100%: only samples that fully match each other in the overlapping regions will be assembled, samples with even a single discrepancy will not be aligned.

## Minimum Overlap Length

This is the minimum length of the aligned region. If the aligned region is shorter than the value you set here (with 25 being the default), alignments will be rejected, and samples will remain in the "Unassembled Samples" folder.

## Minimum Alignment Score

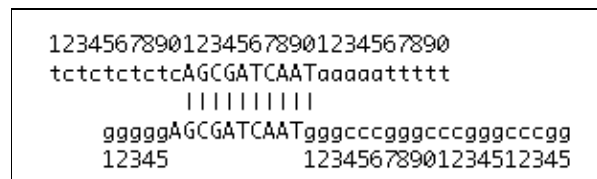
This parameter is similar to the "Minimum Overlap Length", but it takes discrepancies into account. Scores will be scaled so that a match gives a score of 1 - for each matching base in the aligned region, a score of +1 will be added. With the default settings, a score of -2 will be subtracted for each mismatch; for single base insertions or deletions, a score of -5 will be subtracted (-3 gap introduction penalty and -2 gap penalty; additional gaps in the same run lead to a subtraction of -2 per base).

In general, your minimum alignment score should be lower than your minimum overlap length to allow for some level of discrepancies between the sequences.

## Maximum Unaligned End Overlap

This parameter is perhaps the hardest to understand. After doing an alignment, Aligner looks at the unaligned ("dangling") ends of both reads. Since Aligner does local alignments, two aligned sequences may have unaligned bases at the same end. This will happen, for example, when aligning two different copies of a repeat sequence, or if one of the samples is a chimeric clone. It can also happen if the sequence of one or both clones has a very high error rate, as typically seen towards the end of reads that have not been end clipped.

To illustrate how this parameter is calculated, consider the following diagram:



The sequence at the top has 10 unaligned bases at the start and end, and 10 aligned bases in the middle. The bottom sequence has 5 unaligned bases at the start and 20 unaligned bases at the end. Between the two sequences, there are 15 additional bases that could possibly have been aligned - the 5 unaligned bases at the start of the bottom sequence, and the 10 bases at the start of the top sequence.

Aligner calculated the relative amount of unaligned sequence that could have been aligned by dividing the overlapping bases in the unaligned ends by the length of the shorter sequence. In our example, this is 15 / 30 (the length of the top sequence), or 50%. With the default setting of 70% for the "Maximum Unaligned End Overlap", our example would have passed, at least for this parameter.

You may need to adjust this value, depending on the kind of project you are doing. If you aligned cDNA sequences to genomic DNA, use values of or near 100%, since large stretches of exons may be unaligned. But if you expect your samples to match end-to-end, and pre-processed your sequence with end clipping and vector trimming, you can use lower values to reduce the chance that different copies of repeats will be incorrectly assembled together.

## Bandwidth (Maximum Gap Size)

The bandwidth parameter allows you to set the half width of the diagonal used during the banded alignment. This has an effect on the maximum size of gaps (insertions or deletions in one sample) that can still be aligned. A bit simplified, if one sample has an insertion or deletion that is larger than the "bandwidth" number, the alignment will typically stop at the insertion/deletion, and the rest of the sample will be

unaligned. If the insertion or deletion is shorter than the bandwidth, the alignment will continue after introducing the necessary number of gaps in one sequence, as long as the aligned parts after the gaps are long enough (the aligned regions before and after the gaps must be at least 1 and 1/2 times as long as the number of gaps, and longer for any mismatches or ambiguities).

The discussion above is a bit simplified - in reality, what counts is the total number of gaps in one sequence, minus the total number of gaps in the other sequence, at any position, since the alignment uses a banded Needleman-Wunsch algorithm.

The bandwidth parameter has an impact on the assembly speed - larger values mean slower assemblies. For large projects, you may want to reduce the bandwidth value; for projects where you know that you have larger insertions and deletions, you may want to increase it. Note, however, that increasing the bandwidth will typically not be enough to extend alignments through very large gaps, like large introns. Support for "Large gap alignments" will be added to later versions of Aligner.

*The bandwidth parameter is ignored when the "large gap alignments" algorithm is used for assembly.*

## Word Length

The "word length" parameter determines the size of "words" that CodonCode Aligner uses when looking for potential overlaps between sequences. Only sequence pairs that have perfect matches of at least this length will be considered for merging.

If you are trying to assemble sequences with high error or mutation rates, reducing the word length may help to get samples aligned. For very large projects or projects with many repeat sequences, larger numbers may give faster assemblies and better results.

The impact of the word length setting on the assembly speed depends on the size of the project; for large projects, larger word length values can lead to faster alignments.

## Maximum Successive Failures

This parameter is only relevant for larger assemblies with tens or hundreds of reads. It can be used to limit how long Aligner will try to merge contigs for very large projects, with larger numbers meaning more tries and longer assembly times, but potentially fewer contigs.

In detail: Aligner will initially look for overlaps between all samples, and then use this map of overlaps to merge samples into larger and larger contigs. If two samples are already in contigs, Aligner will try to merge the contigs. If the merger is rejected, and Aligner later finds two other overlapping samples in the same contigs, Aligner will try to merge the contigs again. If the contigs have changed in the meantime, this can make sense, since the merger may now work. For very large projects with many repeats, however, this may mean that Aligner tries the same mergers many times. The "Maximum Successive Failures" parameter can be used to stop such fruitless efforts - after Aligner has tried many times (default: 50) in a row to merge contigs without success, the assembly will stop.

*Yes, we do realize that the assembly algorithm could be made smarter here, and we may modify it in the future - but we think that most users will not be doing such large assemblies with Aligner, anyway. There are other assemblers out there that can handle large assemblies, for example Phrap - and you can import Phrap assemblies into Aligner for editing.*

## Match scoring

The last four alignment parameters determine how matches are scored. The "Match score" is used when two aligned nucleotides are identical; the "Mismatch penalty" when two base calls are different. The "Gap penalty" and the "Additional first gap penalty" is used when one of the two sequences has a deletion relative to the other sequence. For single base deletions, the penalty score will be the sum of the gap penalty and the additional first gap penalty; for additional deleted bases (multiple gaps in a row), the penalty will be just the gap penalty.

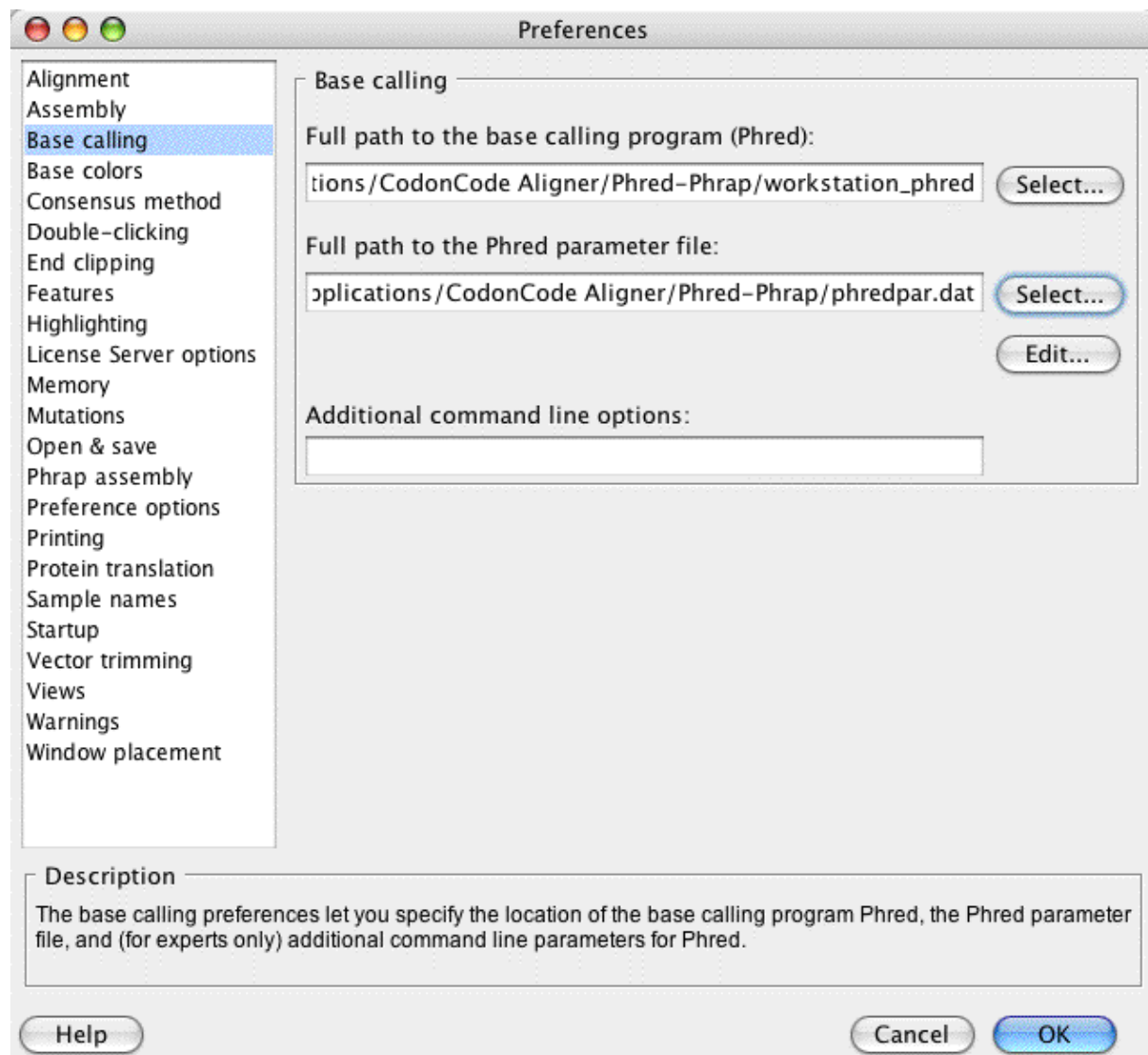
You can change the scoring within limits (scores from 1 to 19 for matches, and penalties of -1 to -19). In general, we suggest that only experts change the match scores and penalties.

## Restoring default parameters

To restore the default parameters, click on the "Defaults" button near the bottom. This will reset all parameters to the choices shown in the screen shot above.

# Base Calling Preferences

In the base calling preferences, you can specify details needed for running the base calling program Phred (to use base calling in Aligner, **you need to have Phred installed on the system that you are running Aligner on**, of course):



## CodonCode Aligner User Manual

You specify the location and name of the base calling program in the upper text field. Currently, Phred is the only base calling program that is supported by Aligner.

In the second text field, you specify the location of the Phred parameter file, which is typically called "phredpar.dat". For more information about the Phred parameter file, please check the [base calling help page](#). It also gives you some more details about how to [edit](#) the Phred parameter file (which you can do by pressing the "Edit..." button).

If you installed Phred with Aligner, the path to Phred and to the Phred parameter file should be correct, and not need any changes. However, if you use your own installation of Phred, you may need to specify the location of both files on your system. The default location and names are:

- On OSX:  
/Application/CodonCode Aligner/Phred-Phrap/workstation\_phred  
/Application/CodonCode Aligner/Phred-Phrap/phredpar.dat
- On Windows:  
C:\Program Files\CodonCode Aligner\Phred-Phrap\workstation\_phred.exe  
C:\Program Files\CodonCode Aligner\Phred-Phrap\phredpar.dat

Note that since Aligner version 1.2.5 beta 5, Aligner will use the workstation versions of Phred (and Phrap) by default. The workstation version of Phred can only be run from CodonCode Aligner, and requires either:

- a valid trial license, or
- a valid full (purchased) license that includes a license for the "Workstation Phred" module. Licenses purchased by academic customers automatically include a license for the "Workstation Phred" module free of charge; users at for-profit organizations need to pay a separate license fee for the workstation version of Phred.

If you started using CodonCode Aligner with a version before 1.2.5 beta 5, and would like to use the workstation version of Phred, please:

1. Check if your license includes a license for "Workstation Phred" (go to the "Help" menu, select "License...", and check the module permissions in the license dialog).
2. Click on the upper "Select..." (or "Browse...") button in the base calling preferences, and navigate to the workstation\_phred executable.
3. Select "workstation\_phred", and then click "OK" in the file dialog and in the preference dialog.

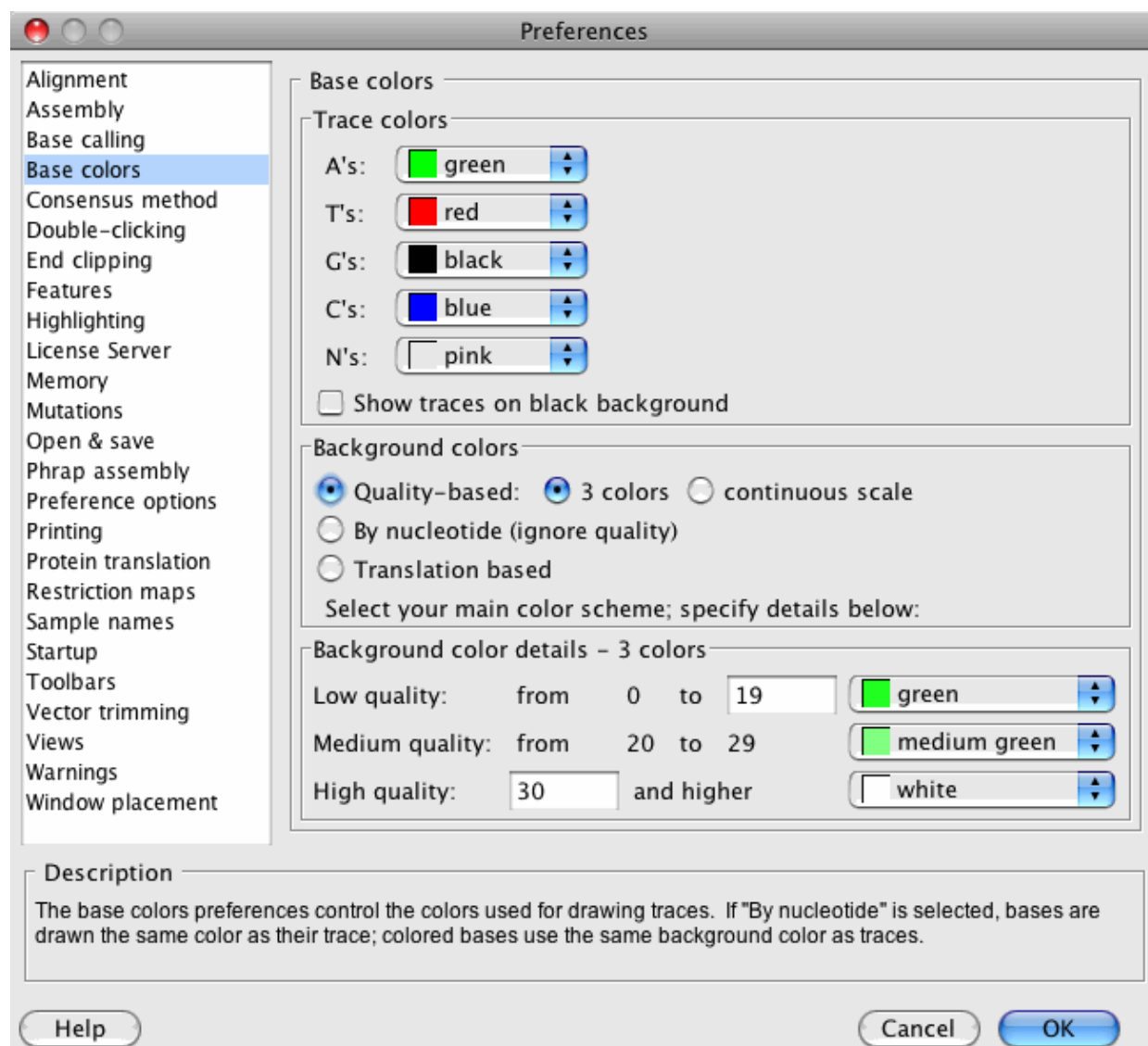
One note of caution: CodonCode Aligner uses the location and name of the Phred executable to determine whether you are using the workstation version or the regular version (both versions perform identically, but the workstation version can only be run from Aligner with a valid license). Aligner will assume that you are using the workstation version if it finds "workstation\_phred" anywhere in the name, which includes the directories. Therefore, **please do not change the name of the Phred executable** installed by Aligner. If you are using your own copy of Phred rather than the workstation version, please make sure to **not** call it "workstation\_phred", or put into a folder hierarchy that contains "workstation\_phred"!

In the bottom text field, you can specify additional command line options for Phred. In general, please leave this line blank - **only experts who really know what they are doing should specify command line options** here! Also, note that Aligner uses the -id and the -cd options to specify input- and output-directories for Phred; for more details, please check the [base calling help page](#).



# Base Color Preferences

CodonCode Aligner can use several different color schemes to display bases. You can control and fine-tune these schemes in the Base Color Preference dialog:



At the top, you can set the colors used to draw the traces (and, if you are using the "By nucleotide" color scheme, for the bases).

Usually, traces will be shown on a white background. If you prefer to see traces on a black background, make sure the checkbox "Show traces on black background" is selected.

In the middle, the "Background colors" panel allows you to choose between different color schemes:

1. **Quality-based background colors.** Here, you can also choose between a three-color scheme for low, medium, and high quality bases (similar to Sequencher), and a continuous color scheme with darker backgrounds for low qualities and lighter backgrounds for higher quality (similar to Consed).
2. **By nucleotide:** colors that vary by base (and ignore base quality).
3. **Translation-based:** Background colors that are based on the protein translation of the sequences, so that blocks of three bases will have the same color, which will depend on the amino acid translation for these three bases.

## Switching color schemes

A quick way to switch between color schemes is to go to the "**View**" menu, choose "**Sequence Colors**" and select the color scheme you want to use.

Another fast option is to use the toolbar button  "**Colors**". Pressing this button will switch between the different color schemes in the following order:

1. Quality-based 3 color scheme
2. Quality-based continuous scheme
3. Base-specific colors
4. Protein translation based background colors

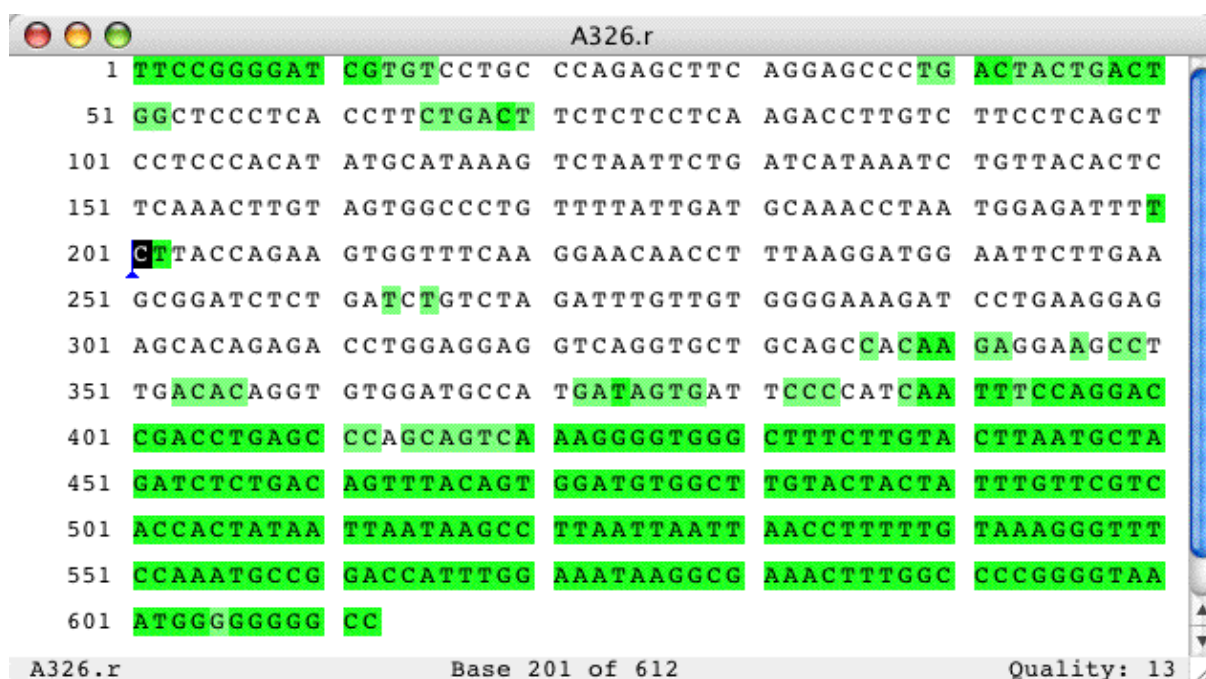
Then, the circle starts over again - after translation-based background colors comes the quality-base 3-color scheme again.

If the toolbar button is not showing, you can change the toolbar settings in the [Toolbar Preferences](#).

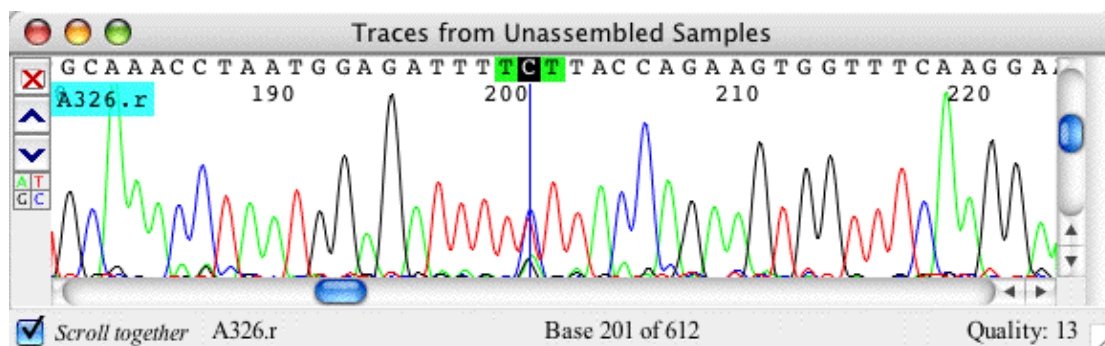
The View menu and the corresponding toolbar icon provide a fast way to switch between the color schemes. For full control, for example to change colors or threshold, use the Base Color preferences.

## Quality-based 3 color scheme

The picture above shows the settings for the 3 color scheme. In this scheme, the background behind the bases is shown in different colors for "low quality" bases, "medium quality" bases, and "High quality" bases. You can set the ranges for low- and high-quality bases (with medium quality being the range in between) by changing the numbers in the text boxes. The most commonly used numbers are 0 to 19 for low quality, and 20-29 for medium quality, and 30 and higher for high quality. You can also assign background colors for each of these ranges. We suggest to use darker colors for lower qualities, and lighter colors for high quality bases. These colors will be used to draw the bases in the base view, contig view, and the trace view windows. Here is an example of the base view with this setting:



Notice that there is a short stretch of low-quality bases at the start of the sequence, and a longer stretch at the end, which is a typical picture. There is also a short stretch of low quality bases near base 200; looking at the trace view, you can see that this is due to multiple peaks at this location:

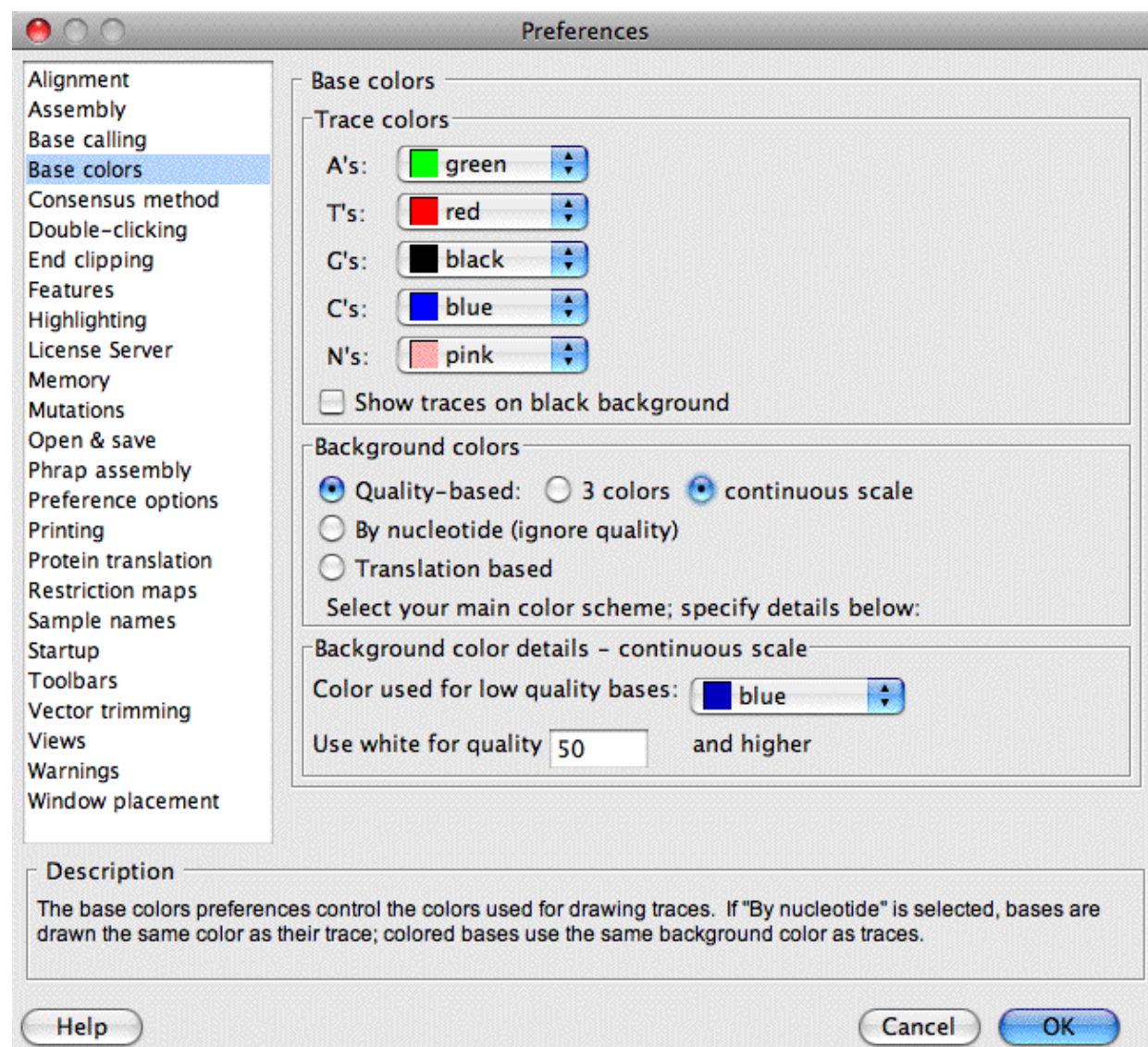


(Note that Phred typically lowers the quality scores of three bases, not just one base, in problem areas, since neighboring bases are also more likely to be wrong).

The 3-color scheme is often useful to get a general idea of the sequence quality. If you want more detailed color shading, select the "Quality-based: continuous scale" color scheme.

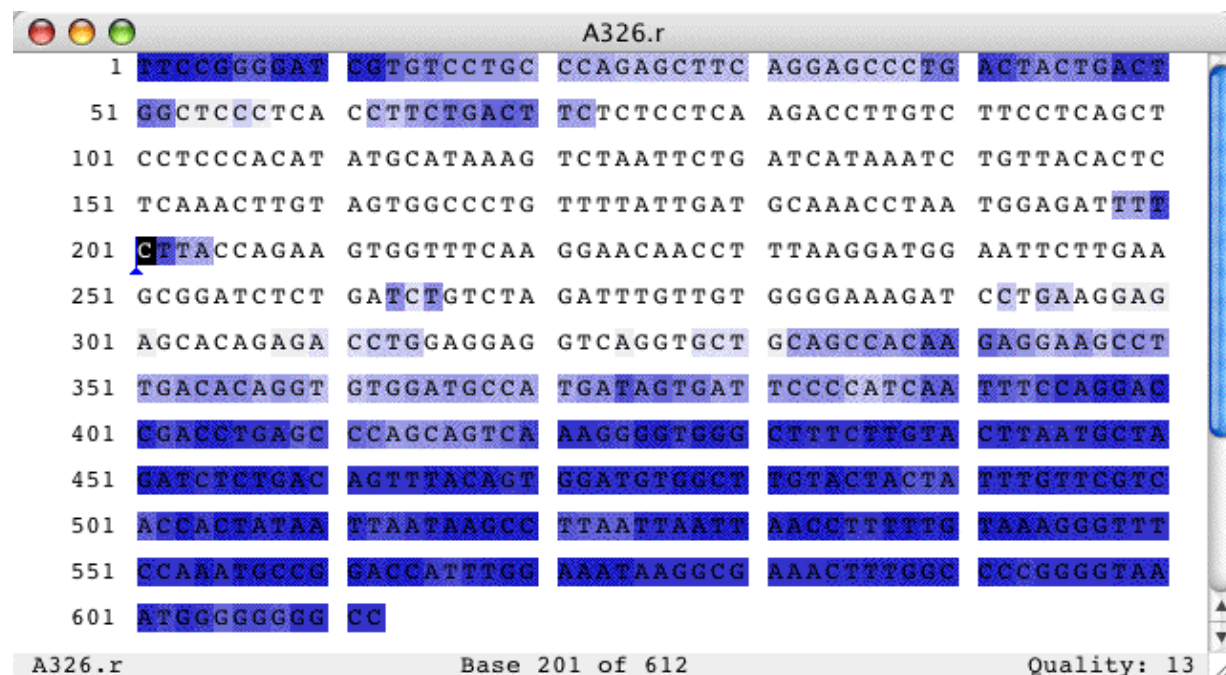
# Continuous, quality-based background colors

In the continuous color scheme, you can select two settings: the background color for low-quality bases, and the minimum quality for which to use a white background.



## CodonCode Aligner User Manual

With the settings shown above, bases with a quality of 0 will be shown on blue background, and bases with qualities of 60 and higher on white background. Bases in between will be shown on varying shades of blue, with darker blues for lower qualities:

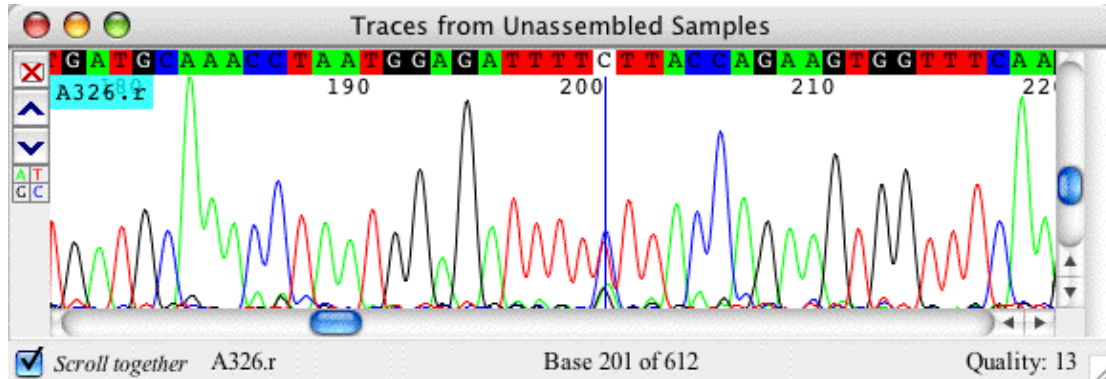


Note that both quality-based color schemes require samples that have Phred qualities (or Phred-like qualities). Aligner will work best with sequences that have such qualities. If you do not have qualities, you can use the base-specific color scheme.

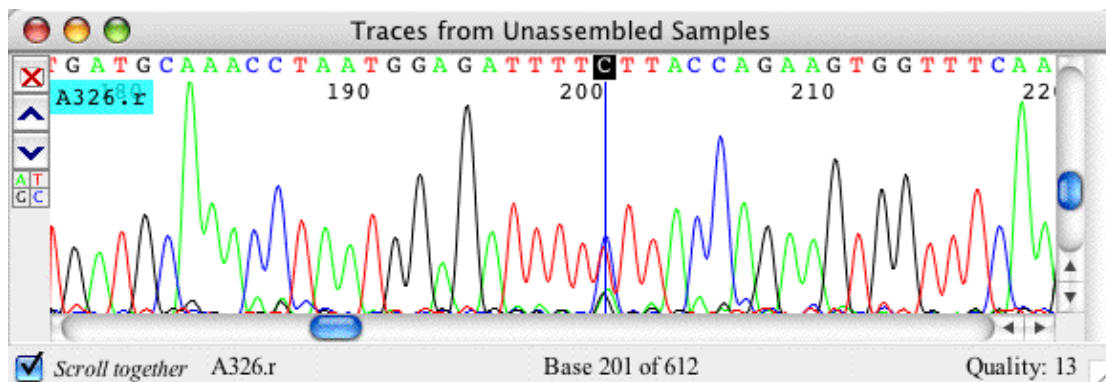


# Base-specific colors

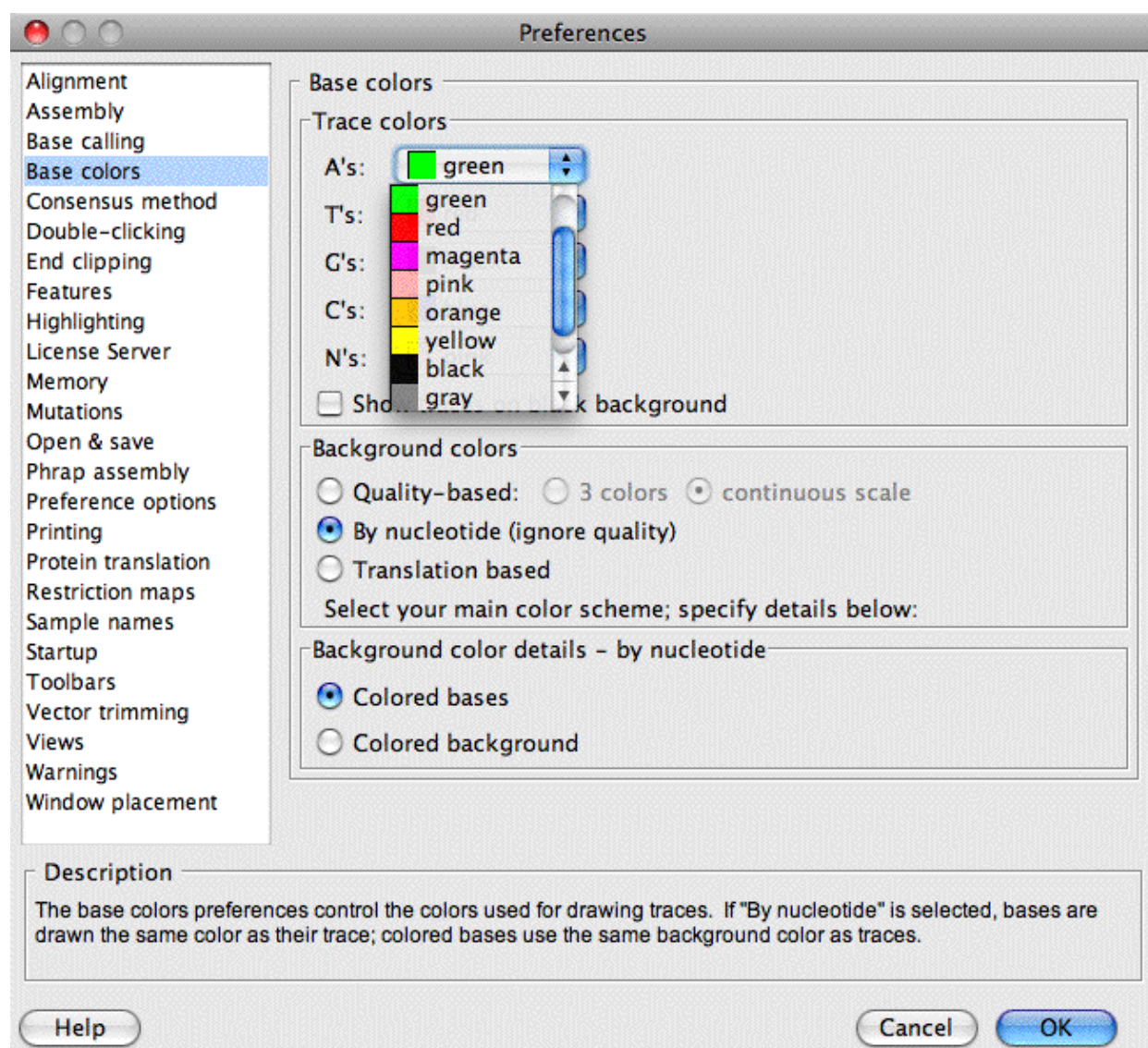
The third color scheme ignores sequence qualities, and chooses the colors from the bases. You can either have bases drawn on colored background, or colored bases on white background. Here's an example of a trace on a colored background:



The same trace with colored bases:

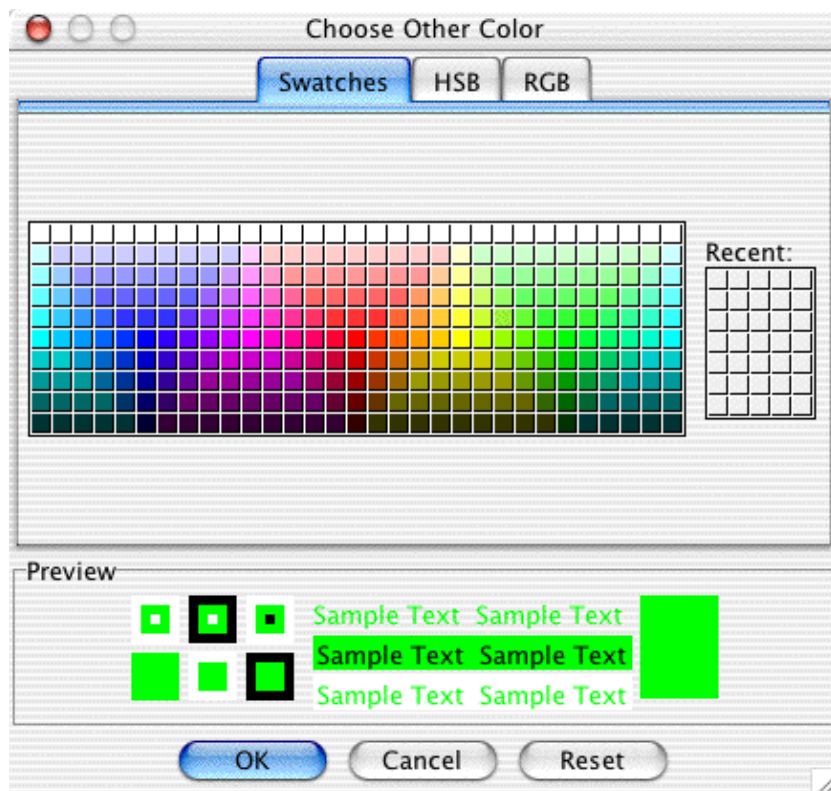


The base-specific color scheme makes most sense when samples do not have qualities, or sometimes when looking for differences in aligned or assembles samples. You can assign the color for each base in the top section of the "Base colors" preference panel, as shown below:



To assign a color to a specific base, use the drop down box to the right of the base. As always, any changes will only be saved when you click "OK". *Note that the colors chosen will also be used to draw the sequence traces, regardless of the background color scheme.*

You can pick one of the pre-defined colors, or click on "Other..." to choose a different color. This will bring up the following "color chooser" dialog:

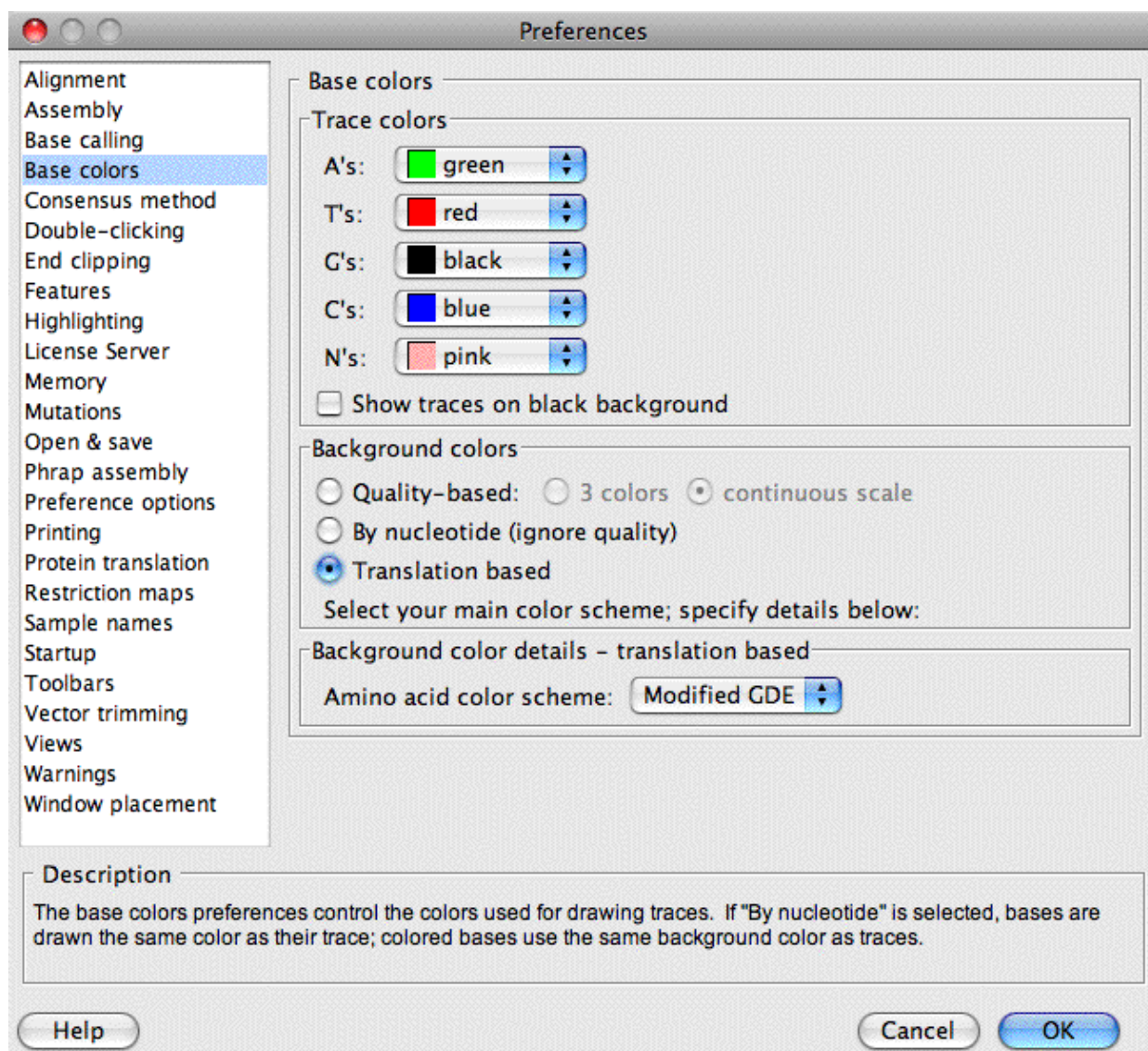




Select the new color, then press "OK" to use this color, or "Cancel".

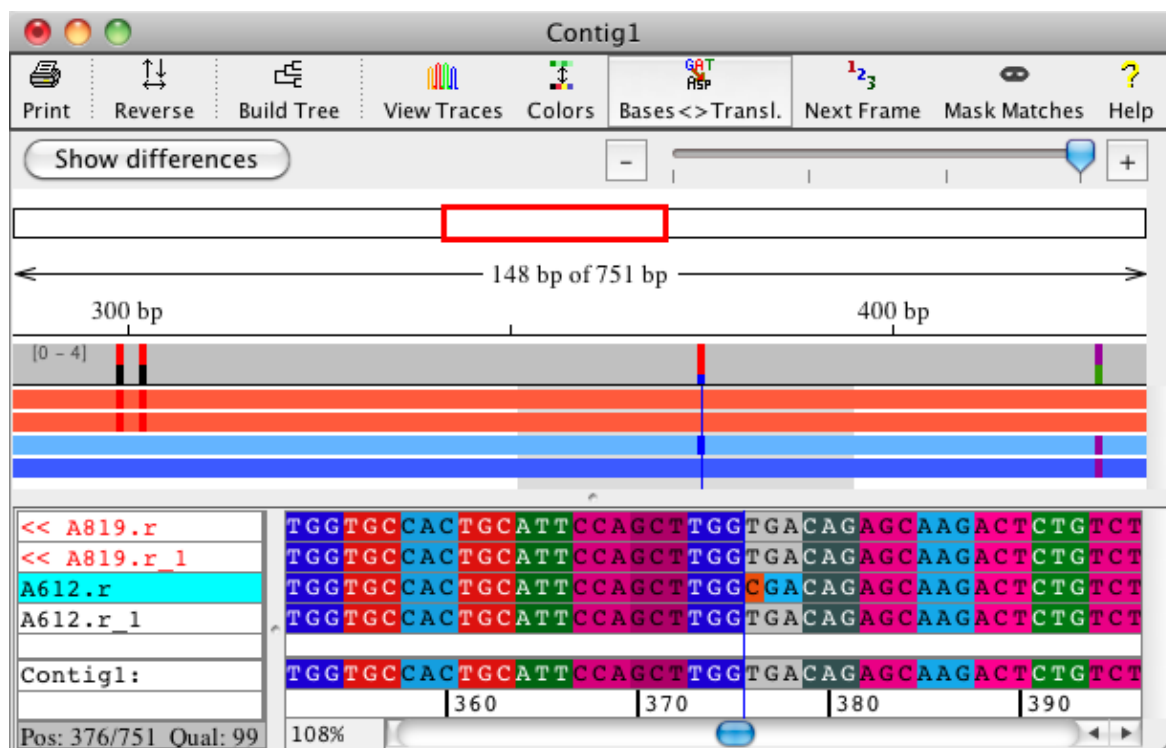
# Translation based background colors

To draw based on a background that indicates the amino acid translation, select the "Translation based" radio button in the "Base colors" preferences:



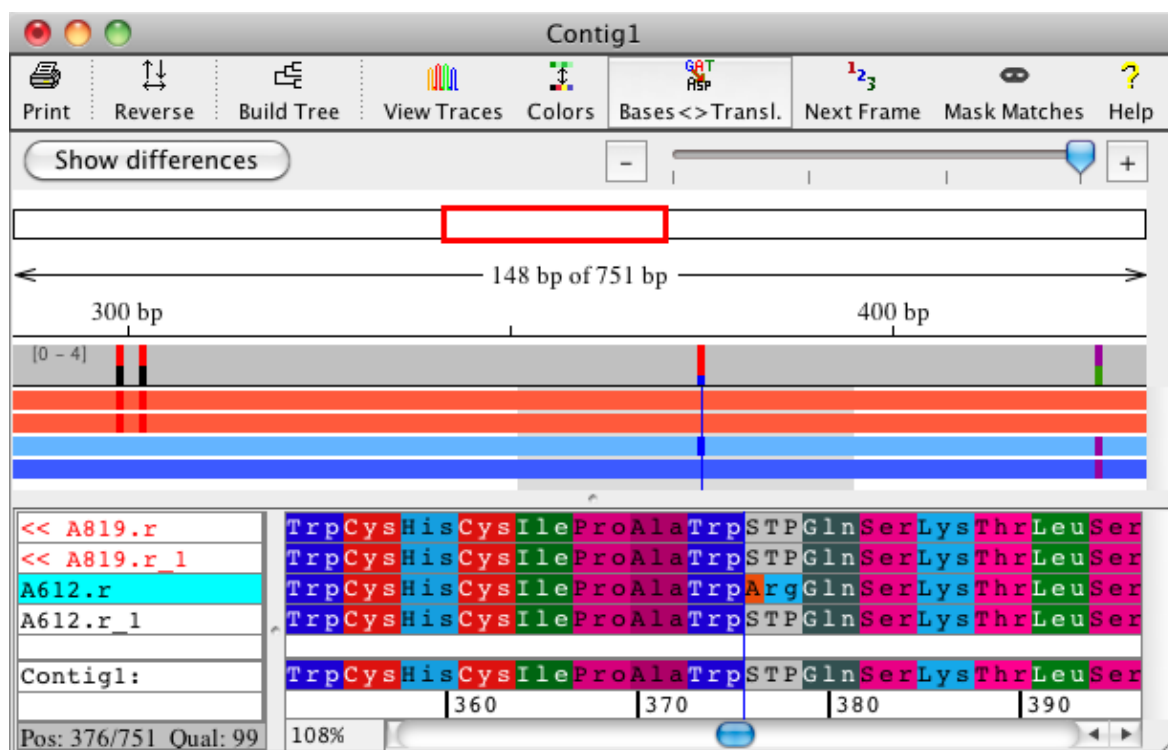
## CodonCode Aligner User Manual

This scheme will use background colors that are based on the protein translation of the sequences, so that blocks of three bases will have the same color, which will depend on the amino acid translation for these three bases. Here is an example:



Using translation-based background colors makes it easy to spot mutations that cause amino acid changes. The reading frame that is used can be changed in the "Sequence Translation" submenu in the "View" menu.

Translation-based background colors can be used when showing bases as above, or when showing amino-acid translations, as shown below:



You can switch between showing bases and showing translations through the "**Sequence Translation**" submenu in the "**View**" menu.

The background colors in the default amino acid color scheme are similar to the "GDE" color scheme as used by the sequence editor Se-AL. However, the scheme has been modified so that similar amino acids use similar, but slightly different colors. You can choose to use the GDE color scheme, where similar amino acids lead to identical background colors, in the "Amino acid colors" pulldown menu.

# Consensus Preferences

The consensus preferences allow you to specify how the contig ("consensus") sequences are calculated:

Preferences

Alignment  
Assembly  
Base calling  
Base colors  
Clicking & scrolling  
**Consensus method**  
End clipping  
Features  
Highlighting  
License Server  
Memory  
Mutations  
Open & save  
Phrap assembly  
Preference options  
Printing  
Protein translation  
Restriction maps  
Sample names  
Startup  
Toolbars  
Update checks  
Vector trimming  
Views  
Warnings  
Window placement

**Consensus method**

For regular contigs: Quality-based

For contigs of contigs: Percentage min. %: 25.0

For very large (NGS) contigs: Majority

☐ Replace '-' in consensus with 'n': if any sample has bases

**Quality based consensus determination**

Subtract quality scores of mismatches? ☒ Ignore ☐ Subtract

Default qualities for:  
Low-quality edits 10 High-quality edits 30

**Align to reference sequence consensus determination**

☐ Always use reference sequence as the consensus  
☐ Use reference sequence as the consensus at low coverage  
☒ Exclude reference sequence when building consensus  
In uncovered regions, use - as consensus

**External Consensus**

☒ Conserve external consensus  
☐ Rebuild external consensus on import

**Low coverage regions**

☐ If coverage is less than 2, use N

**Description**

The consensus method preferences control how the consensus sequence is formed.

Help Cancel OK

## Consensus Method

In the section "Consensus method" at the top, you choose whether Aligner will use a quality-based consensus, a majority consensus, an inclusive consensus, or a percentage consensus. For the percentage consensus, you can set the minimum percent that a base has to occur to be included in the consensus. You can set different consensus methods for regular contigs and for contigs of contigs. The default setting in CodonCode Aligner is to use a quality-based consensus for both, regular contigs and contigs of contigs.

For chromatogram-based projects, we strongly suggest using a **quality-based consensus**. A quality-based consensus mimics what a human "contig editor" would do to determine the correct consensus sequence: Aligner looks for the highest quality sample at each position, taking into consideration confirmation by samples in reverse direction, as well as disagreements.

One exception where you may not want to use a quality-based consensus are **re-sequencing projects**, where you align your sequences to a known reference sequence. In such projects, you may want to use the **majority consensus**, so you can see the most common base at each position in the consensus sequence (for details about how Aligner builds the majority consensus, check the "[Algorithms](#)" section). Alternatively, you can use a **percentage-based consensus**, which lets you define a cutoff frequency - only bases above this frequency will be included when building the consensus. This allows you to ignore rare mistakes or mutants. Using a percentage-based consensus often makes sense for "contigs of contigs" that were generated with Aligner's "Compare Contigs" function.

For NGS-type projects, which CodonCode Aligner recognizes by deep coverage and the absence of chromatograms, using a majority consensus typically gives the best results.

## Consensus Gaps

By default, CodonCode Aligner will use a '-' character (minus sign) to indicate gaps in the consensus sequence. However, if you plan to align contigs with the program MUSCLE using CodonCode Aligner's "Compare Contigs" function, this can sometimes cause problems, since MUSCLE will remove all gaps before alignment. When aligning consensus sequences, this can lead to the deletion of bases in samples when importing the alignment results.

Similar problems might occur with other analysis programs if you export consensus sequences with gaps.

To avoid such problems, CodonCode Aligner enables you to replace '-' characters in consensus sequences with 'n' characters. To use this option, make sure the checkbox labeled **"Replace '-' in consensus with 'n'"** is checked. You have two choices when to replace '-' with 'n':

1. **If any sample has bases.** This option will use 'n' in the consensus if any of the aligned samples at this position has a non-gap base. But if all samples have a gap at this position, the '-' character will be used in the consensus.
2. **Always.** This option will replace all '-' characters in the consensus with 'n'.

The first option can be useful when working with aligned contigs ("contigs of contigs") when some, but not all, of the aligned contigs have insertions. The contigs with insertions will have 'n' in the consensus, while the other contigs will have a '-'

The second option (always use 'n') makes sure that consensus sequences do not have any gap characters ('-') at all. One example where this option can be useful is if you are working with contigs that contain samples with

heterozygous insertions or deletions, where the longer allele leads to the introduction of gaps in the consensus.

When you change any consensus method preferences and then click the "OK" button, the consensus sequences for all contigs in the currently open project will be re-calculated.

## Quality Scores At Discrepancies And Edits

You can choose whether or not you want Aligner to **subtract the quality scores** of discrepant bases when calculating the consensus quality score. In general, this is not necessary, since discrepancies are typically errors that do not influence the correctness of the consensus base; however, you can tell Aligner that you think differently, and subtract the quality scores of discrepant bases.

You can also set the quality values used for edited bases when calculating consensus quality. Internally, Aligner uses qualities of 98 for "low quality" edits, and 99 for "high quality" edits. This follows conventions used by the contig editing programs Gap4 and Consed, and it allows Aligner to take edited bases into account when it matters, for example when counting Phred20 bases. When calculating the consensus quality, Aligner will substitute the quality scores with the scores you specify here; the default values are 10 for low quality edits, and 30 for high quality edits.

## Reference Sequence Alignments

When building the reference sequence for contigs that were generated by aligning to a reference sequence, the default behaviour is to ignore the reference sequence when building the consensus. However, instead of building a consensus using one of the consensus methods from the top section, Aligner can also **use the reference sequence as the consensus** sequence for contigs that were generated by alignment to a reference sequence. To do this, check the box labeled "Use the reference sequence as the consensus".

You can also ignore the reference sequence completely when building the consensus. To do this, select the **"Exclude reference sequence when building consensus"** radio button. If you exclude the reference sequence when building the consensus, also choose one of the characters 'X', 'N', or '-' from the pull down box, that will be used as the consensus at uncovered regions.

A third option for building the consensus sequence in aligned contigs is to **"Use reference sequence as consensus sequence at low coverage"**. This will use the reference sequence in all areas with less than 2-fold coverage, and in areas with 2-fold coverage when the two reads differ. This option can be useful to minimize errors in the consensus sequence due to random sequencing errors.

## External Consensus Sequences

The section "External Consensus" only concerns the consensus of imported assemblies. You can choose whether to keep the consensus sequence when you import entire assemblies, or to re-calculate the consensus based on the current settings. Even when you decide to keep the imported consensus sequence, however, Aligner will re-calculate the consensus sequence when you later change a contig, for example by editing bases or removing samples.

## Masking Low Coverage Regions

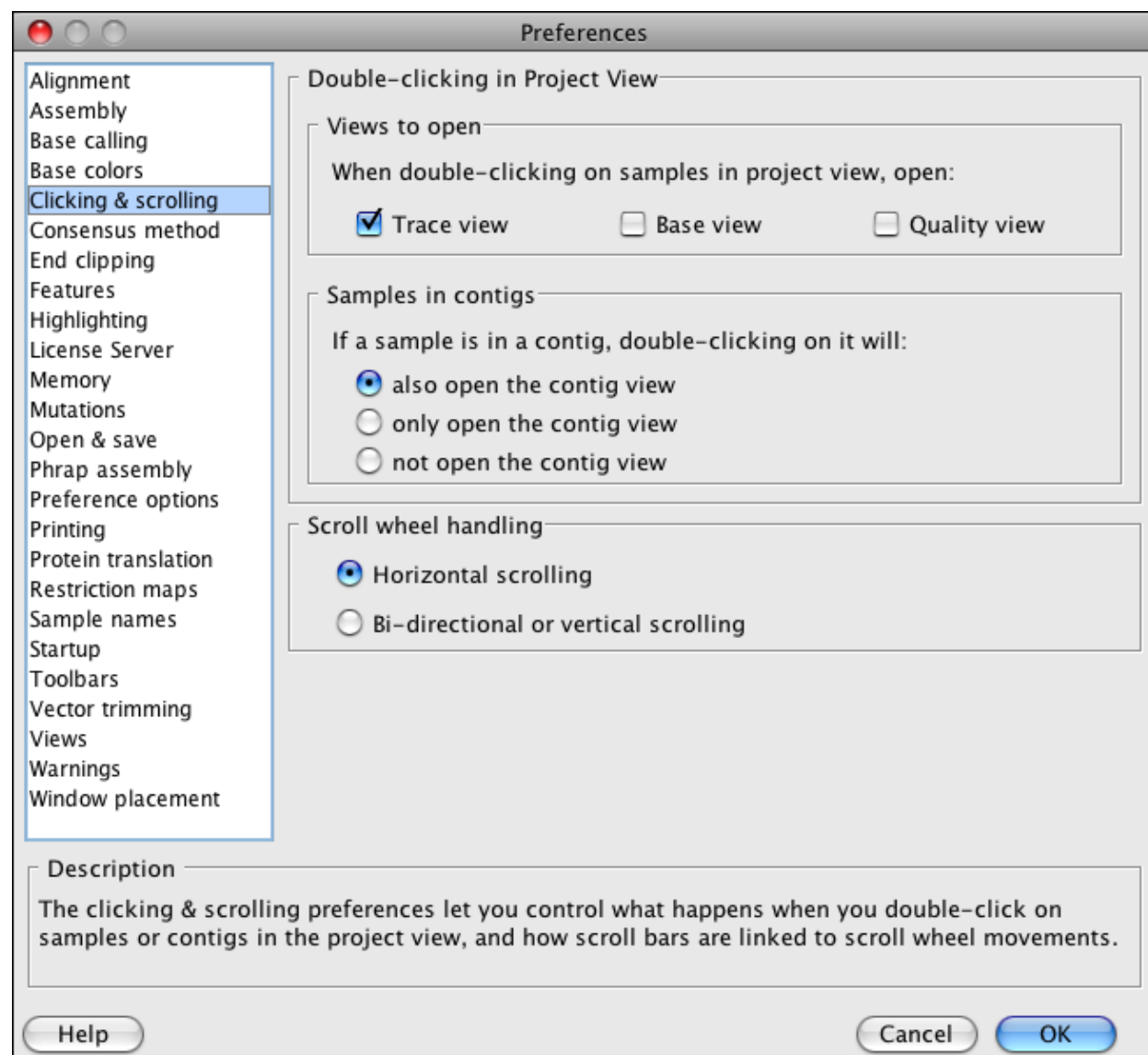
The section at the bottom called "Low coverage regions" allows you to automatically **set the consensus bases in regions with low coverage** to the chosen character. You can specify in which regions you want to replace

the consensus bases with the masking character, by setting the minimum number of coverage needed to show the original consensus base. If the coverage is less than the set number, the consensus base at this position will be replaced with the chosen character. Please note that the default behavior in CodonCode Aligner is to not use this option.



# Clicking & Scrolling Preferences

The clicking & scrolling preferences allow you to choose what happens if you double-click on one or more samples in the project view windows, and to set how your scroll wheels are used:



The section "Double-clicking in Project View" allows you to choose which views (windows) Aligner should open in response to double-clicks in the Project View. If the samples you selected are in contigs, you can also choose if Aligner should open the corresponding contig view in addition to the other views selected above (such as the trace view), or just open the corresponding contig view, or not open the contig view at all.

In the section "Scroll wheel handling", you can set how the scroll bars are linked to your scroll wheel movements. "Horizontal scrolling" uses the scroll wheel to scroll horizontally. If this option is selected, you can scroll left and right in the contig view by using your scroll wheel. If your mouse supports bi-directional scrolling (e.g. the Magic Mouse or Mighty Mouse from Apple), or you would like to scroll vertically in the contig view, then select the option "Bi-directional or vertical scrolling".

# End Clipping Preferences

The end clipping preferences allow you to specify how Aligner will remove low-quality sequence from the ends of samples, and to set up minimum quality criteria after end clipping:

The screenshot shows a 'Preferences' dialog box with a sidebar on the left containing a list of settings categories. The 'End clipping' category is selected and highlighted in blue. The main area of the dialog is titled 'End clipping' and contains three sections: 'End clipping', 'Trim from start until', and 'Trim from end until'. The 'End clipping' section has two radio buttons: 'Maximize region with error rate below' (selected) with a value of '0.1', and 'Use separate criteria for start and end:'. The 'Trim from start until' section has two checkboxes: 'Error rate is below' (checked) with a value of '0.1' in a '25' base window, and 'There are fewer than' (unchecked) with a value of '3' bases with quality below '20' in a '25' base window. The 'Trim from end until' section has identical settings. The 'After end clipping' section has two checked checkboxes: 'Move all sequences shorter than' with a value of '25' bases to trash, and 'Move all sequences with fewer than' with a value of '50' bases to trash. Below these is a 'Phred 20' button with a dropdown arrow. At the bottom of the dialog is a 'Description' section with the text: 'The end clipping preferences control how low quality bases are clipped from samples; end clipping works only on samples with qualities.' At the very bottom are three buttons: 'Help', 'Cancel', and 'OK'.

**Preferences**

**End clipping**

☒ Maximize region with error rate below

☐ Use separate criteria for start and end:

**Trim from start until**

☒ Error rate is below  in a  base window

☐ There are fewer than  bases with quality below  in a  base window

**Trim from end until**

☒ Error rate is below  in a  base window

☐ There are fewer than  bases with quality below  in a  base window

**After end clipping**

☒ Move all sequences shorter than  bases to trash.

☒ Move all sequences with fewer than  bases to trash.

Phred 20

**Description**

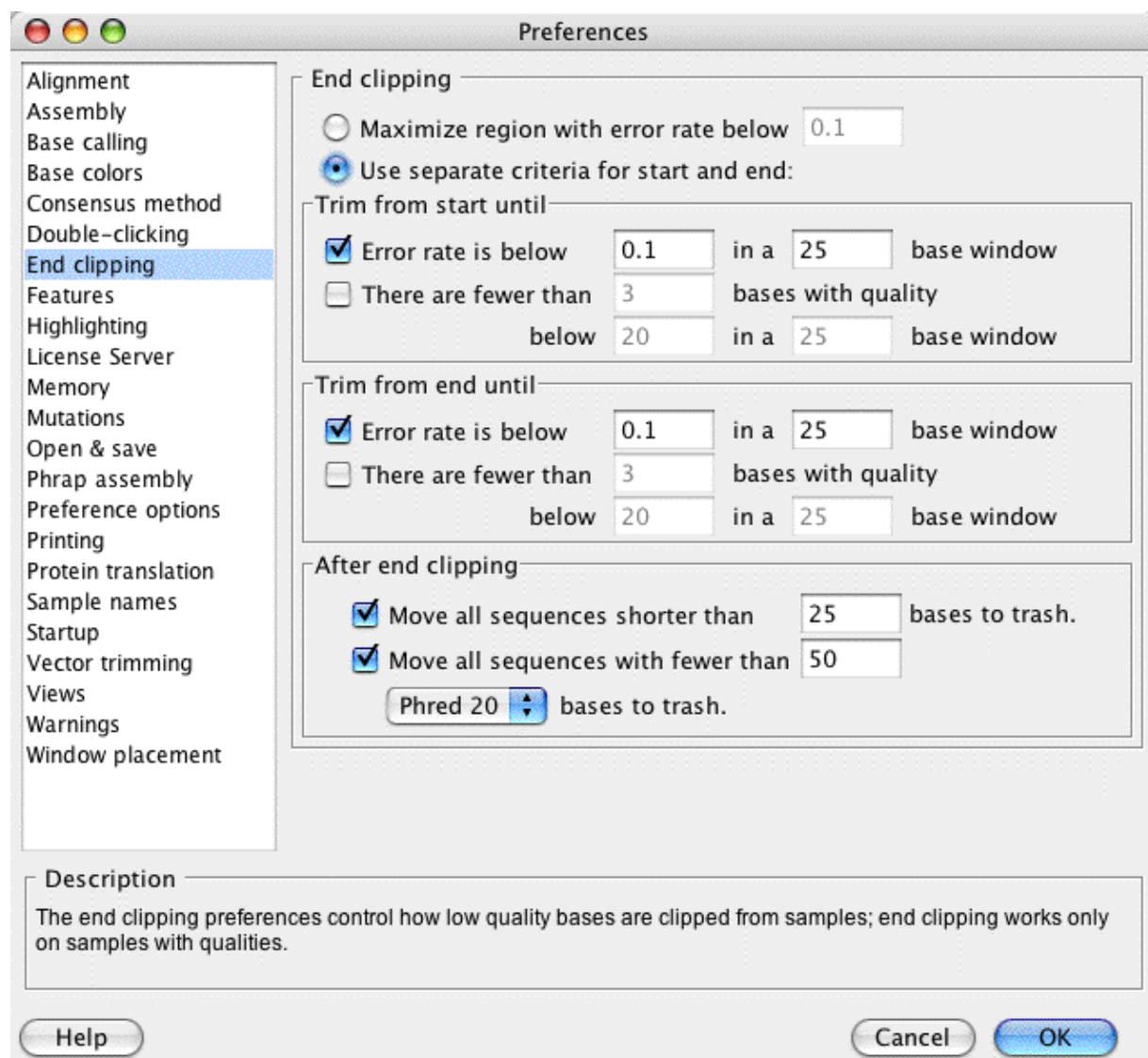
The end clipping preferences control how low quality bases are clipped from samples; end clipping works only on samples with qualities.

On the top, you have the choice between two different end clipping methods:

1. a method that maximizes the region with an (estimated) error rate below the threshold you define, and
2. a method that uses different criteria at the beginning at the end of the reads.

The first method is similar to the way Phred trims sequences with the "-trim\_alt" option. The second method gives you more options, and is (hopefully) a bit easier to understand. With typical parameters, both methods tend to give similar results, and remove the "junk" sequence at the end of chromatograms. The methods are explained in more detail on the ["End clipping algorithms" help page](#).

If you choose the first method, you only have to specify one parameter, which is the maximum error rate for the clipped region. If you select the second trim method, you can define the end clipping stringency in more detail, as illustrated below:



You can choose to trim from the start until the estimated error rate drops below the cutoff you choose, and also specify the length of the "window" over which to calculate the expected error rate. Alternatively, you can clip until there are only very few low-quality bases in a window, specifying the window length, the maximum number of low quality bases, and what "low quality" means to you (typically, Phred scores lower than 20). You can also combine these two measures, or choose neither one if you do not want to end clip at the start.

The same applies to clipping from the end; you can choose different numbers at the end of sequences. For example, it often makes sense to use longer windows at the end than at the start, since the quality rapidly improves at the start of sequences, but only slowly deteriorates at the end.

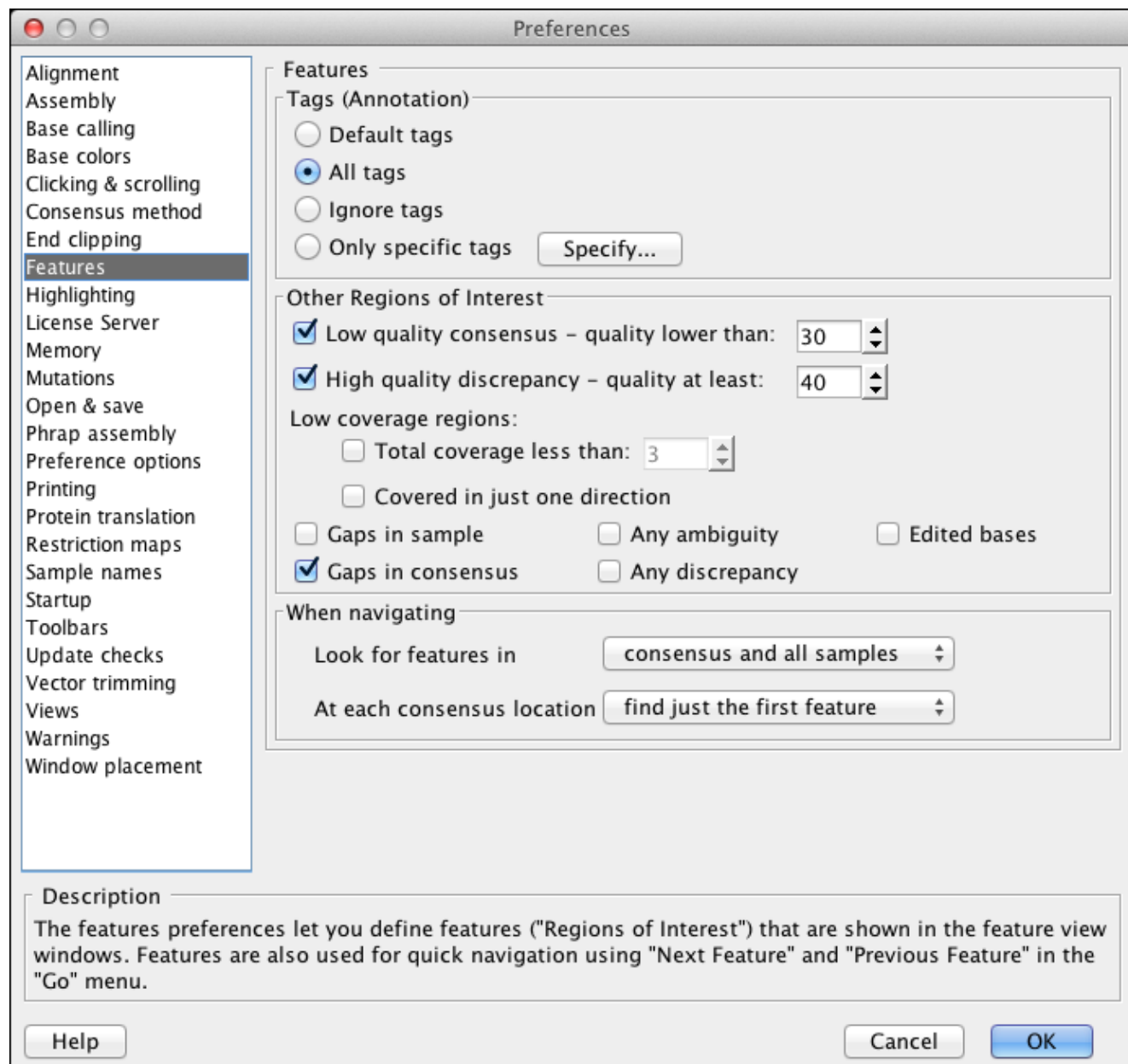
## Automatically removing short and low-quality sequences after end clipping

You have the option to automatically identify "bad" sequences after the end clipping, and move those to the trash. Such bad sequences can be due to failed sequencing reactions, or any number of other problems. We suggest to move all sequences that are too short (for example, less than 25 or 100 bases) after clipping to the trash. A second widely used method to identify low-quality sequences is to count the number of bases with quality scores above 20 ("Phred 20 bases"). With the settings shown in the picture above, any samples that have fewer than 350 Phred 20 bases after the end clipping would be called bad and moved to the trash.

*We suggest that you experiment with the different parameters for end clipping, and find a setting that works for your data. For example, if you sequence short PCR products, you should definitely reduce the number of Phred 20 bases required for a sequence to be kept, or perhaps uncheck this option. The results of end clipping are not saved until you save your project - you can use this to try out different parameters. Just close the project without saving, open it again, and end clip with different settings, until you found settings that feel "right" to you.*

# Feature Preferences

Aligner allows you to define criteria for places that you want to examine more closely, called "Features". Your definition of features can include low-quality consensus bases, low coverage regions, gaps, ambiguities, and tags; you make your choices in the "Features" preferences:

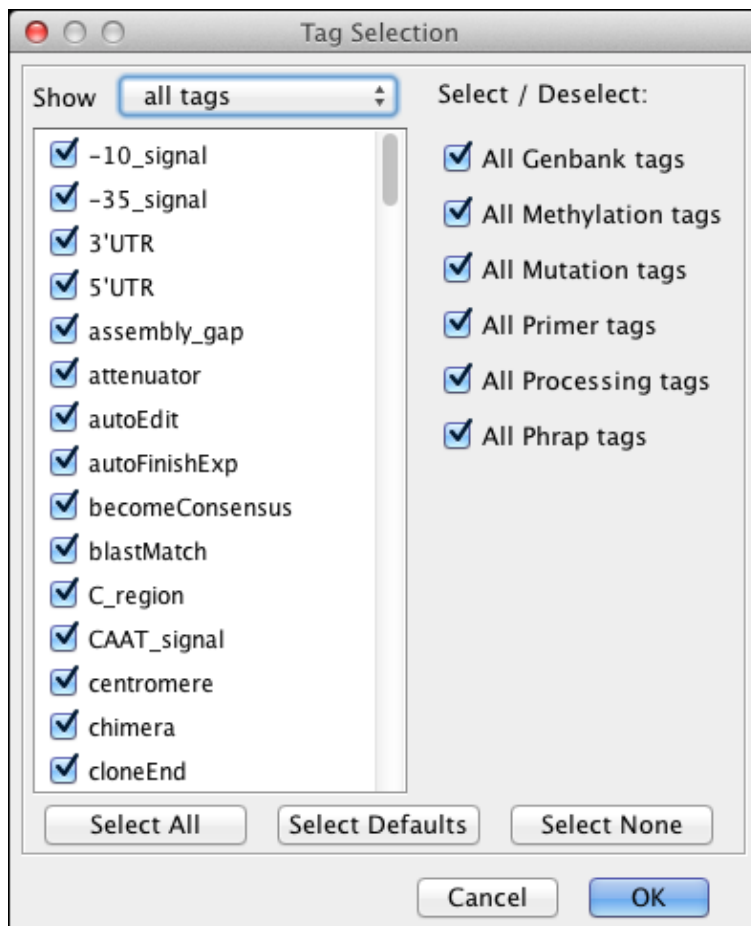


In the top panel, you define the criteria of features (interesting regions). Any place where one or more of the selected criteria are met will become a feature. This definition of features will be used by the "Feature View" window for contigs, and when navigating in a contig view using "**Next Feature**" and "**Previous Feature**" in the "**Go**" menu.

In the lower panel, you can define where Aligner should look for features when navigating - in the consensus sequence and all samples, or just a sub-set of the consensus or samples. This selection is used only for navigating - the feature view windows will always show all features for the entire contig, both in the consensus and in all samples.

## Specifying subsets of tags

Tags contain annotation, for example from Genbank or EMBL files. A number of different programs also add tags to regions of sequences. For example, the assembly program Phrap adds tags for compressions and for regions that match regions in other contigs, and PolyPhred uses tags to mark possible mutations it has identified. You may want to use Aligner's feature view or feature navigation to look at only some of these tags - for example, only tags from Genbank files, but not "processing" tags added by CodonCode Aligner. To do this, click on the "**Specify**" button in the Feature preference dialog. This will bring up the following dialog:

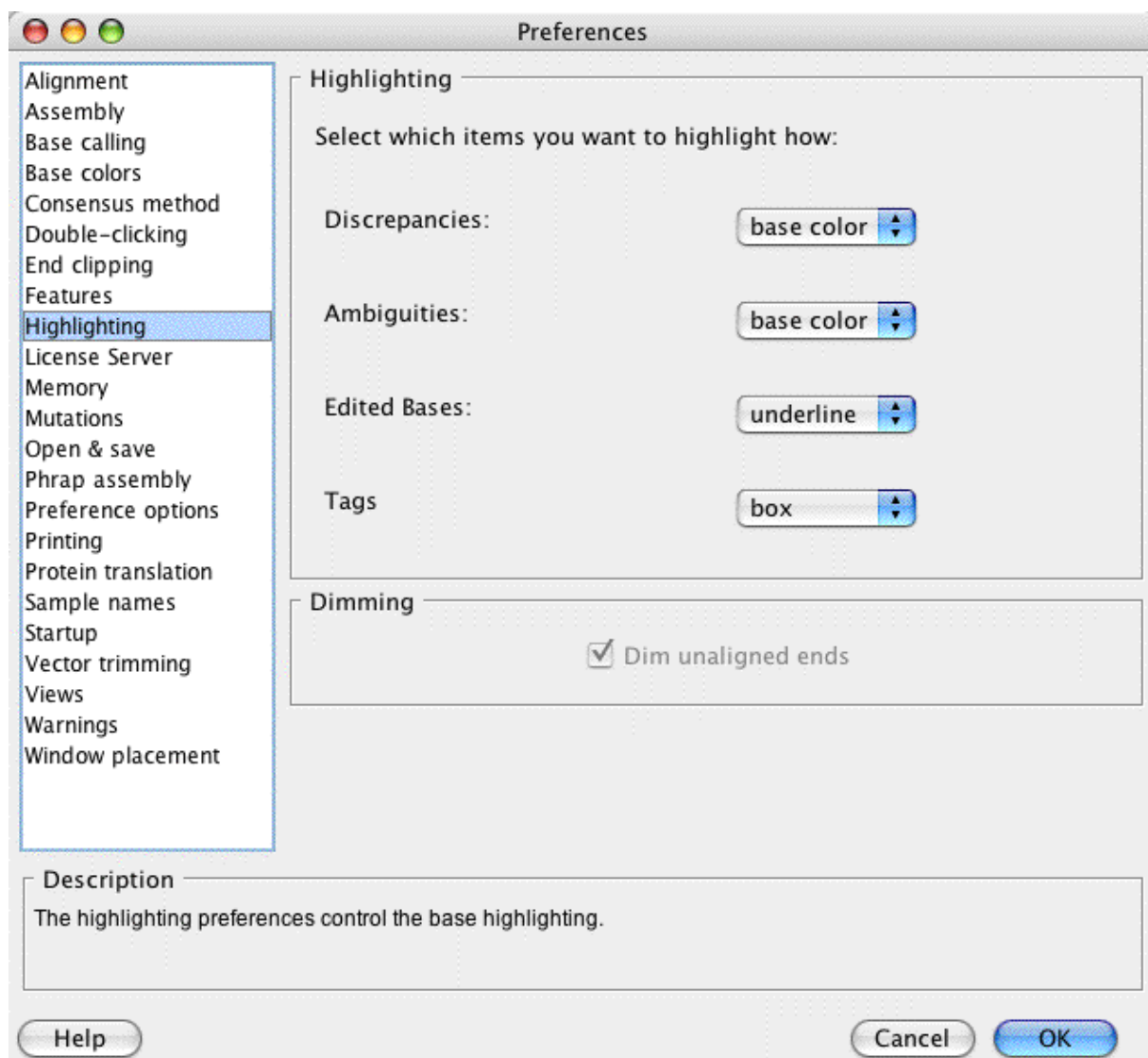


You can use the buttons and checkboxes to select or unselect groups of tags. You can fine-tune your selection in the list of tags on the right. To make or change **discontinuous selections**, use Control-click on Windows and Command-Click on OS X. Your selections will take effect when you click "OK" in the tag selection dialog, and then in the Preference dialog.



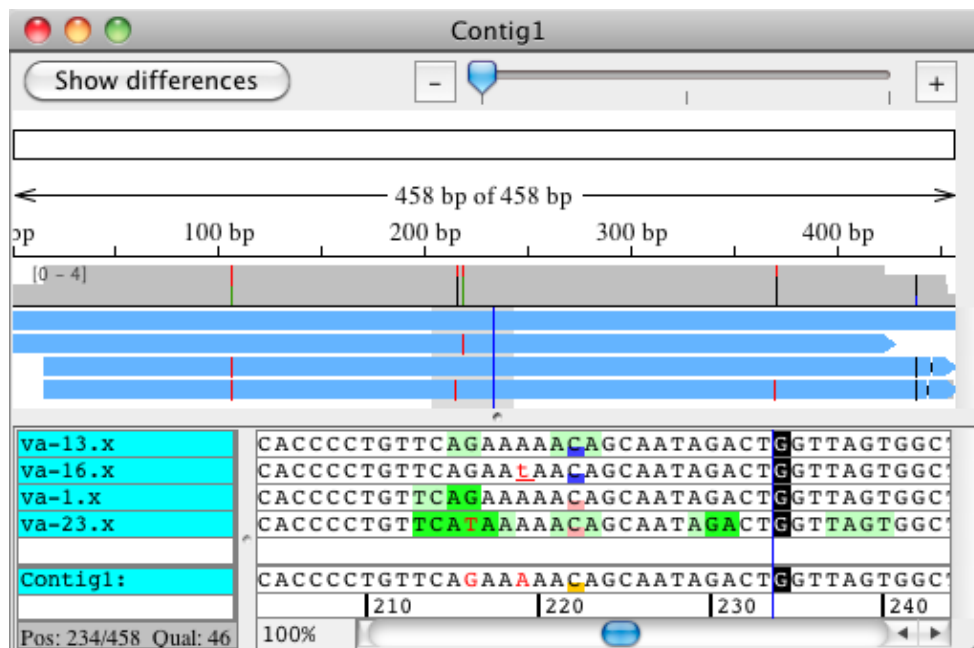
# Highlighting Preferences

You can select how Aligner highlights discrepancies, ambiguities, edited bases, and tags in the highlighting preferences:



## CodonCode Aligner User Manual

By default, Aligner will highlight discrepancies and ambiguities by using a different background or foreground color (depending on the color scheme and settings); edited bases by underlining the base call; and tags by a colored box that covers the bottom of the base. An example is shown below:

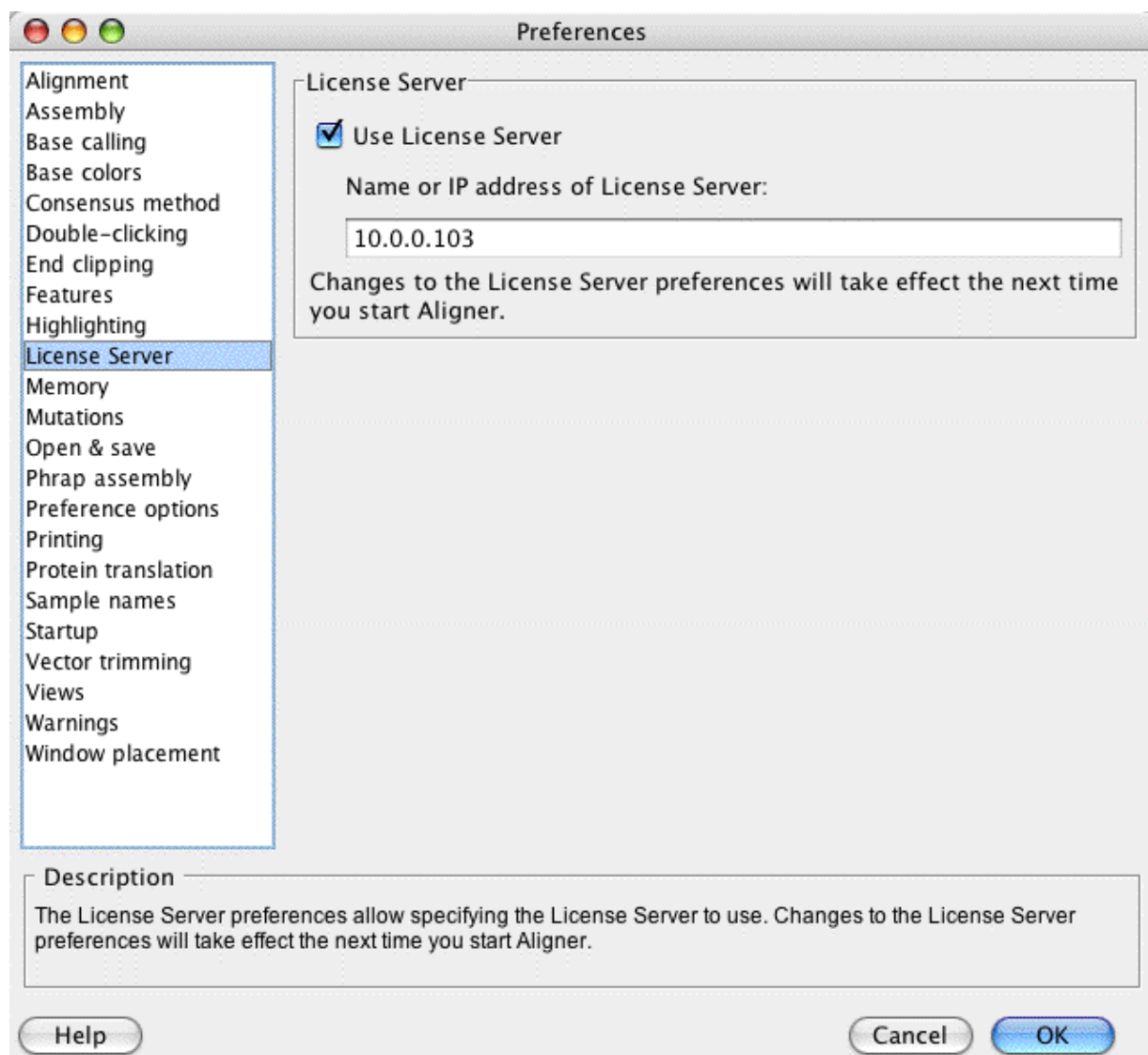


Here, discrepant bases (the two T's near the middle) are shown in red. The lower-case "T" in sample va-16.x was edited, as shown by the underline. The C's in the center have two different kinds of tags, as shown by the pink and blue boxes.

You can change these settings for each of the four categories by choosing an option in the pulldown menu next to it. Changes will take effect when you press "OK".

# License Server Preferences

The License Server preferences allow you set whether CodonCode Aligner should use Aligner License Server, and the name or address of the License Server computer used. Most users will not need to use this preference panel, since CodonCode Aligner can automatically find a local License Server to use when starting up.



## CodonCode Aligner User Manual

However, you may need to specify the address of the License Server computer here if Aligner cannot automatically find the correct License Server . This can happen if:

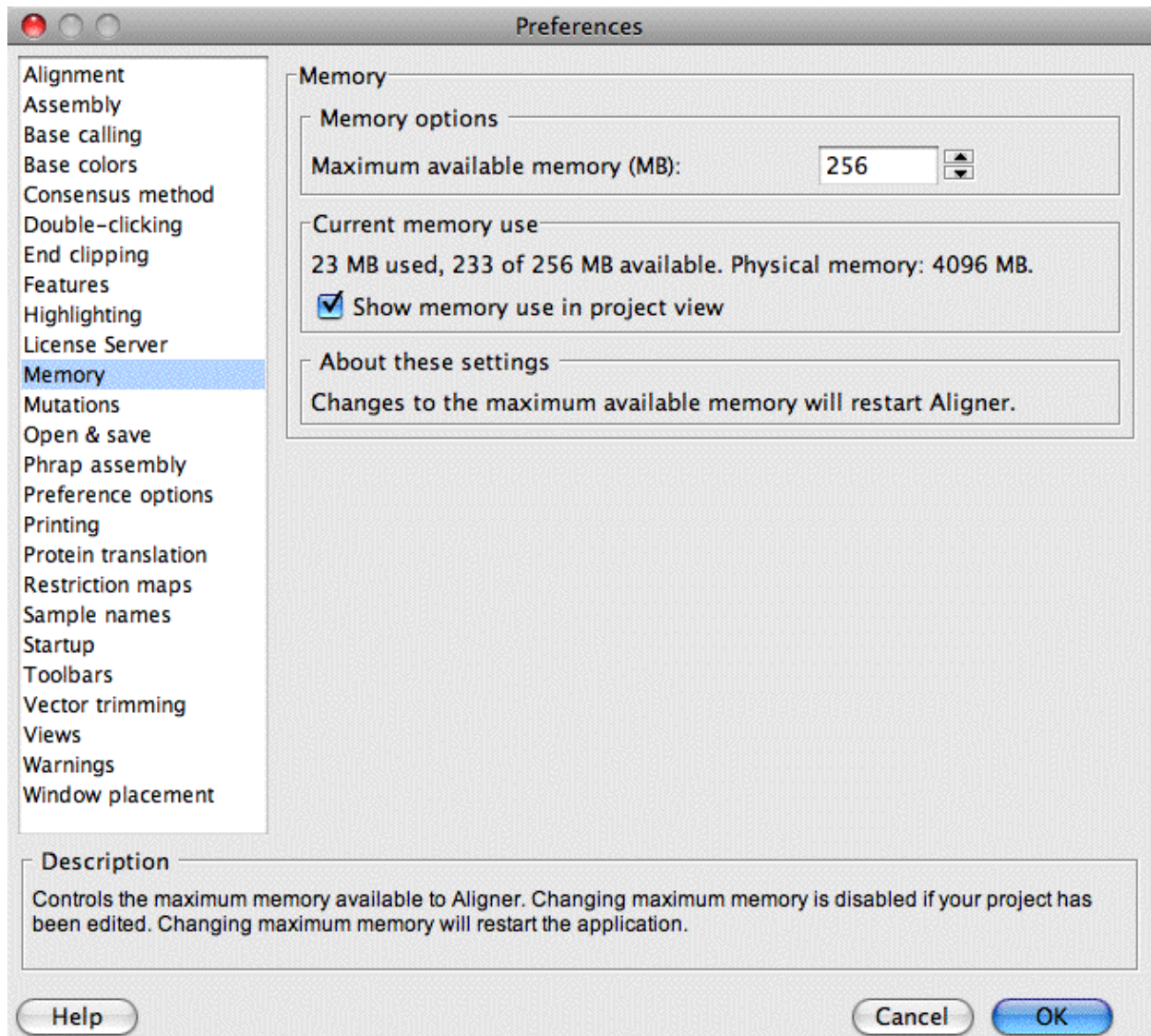
- The License Server is on a different "subnet" (for example in a different department).
- A firewall between (or on) your computer and the License Server computer blocks the automatic License Server detection.
- CodonCode Aligner finds a License Server different from the one you want to use (for example the one from the neighboring lab).
- You are switching from a single-user license to using the License Server .

You can specify the License Server address either as computer name (for example "mylicenseserver.myuniversity.edu") or as an IP address (for example "10.123.012.20"). If you do not know the name or address to use, please contact your local system administrator.

Any changes made will take effect the next time you start Aligner after quitting your current Aligner session.

# Memory Preferences

The memory preferences allow you to see how much memory CodonCode Aligner is currently using. On Mac OS X, it also allows you to set the memory available to Aligner:



The "Memory options" panel at the top shows you the maximum memory available to CodonCode Aligner.

The panel in the middle shows you how much memory CodonCode Aligner is currently using, and how much memory is available. It also shows the amount of physical memory (built-in RAM) for your computer.

If the check box "Show memory use in project view" is checked, CodonCode Aligner will show the current memory use in the bottom right corner of the project view window.

## Memory on Windows

On **Windows**, CodonCode Aligner will automatically try to get as much memory as is needed, with the following limitations:

- On **32-bit versions of Windows**, the maximum memory available to CodonCode Aligner is limited to 1 GB.
- On **64-bit versions of Windows**, the maximum memory available to CodonCode Aligner is limited by the available physical memory (RAM). This number may be lower than the total amount of RAM because of memory use by other open applications.

CodonCode Aligner may use "virtual" memory if needed, so the number shown as "available" may be larger than the amount of free physical memory. Please note, however, that using virtual memory may lead to substantially slower performance, since memory must be "swapped" to and from the hard disk. This will happen when the amount of memory used is larger than the amount of physical memory; it can also happen earlier if you have other applications running. **The slowdown due to "swapping" can be dramatic - 10-500 fold slowdowns**, and pauses of several seconds where nothing seems to happen, are common. If you work with large projects that need a lot of memory, you may want to consider installing more memory (RAM), and quit other applications before starting CodonCode Aligner.

With 32-bit versions of CodonCode Aligner, it is possible that the application fails to start if not enough memory is available (CodonCode Aligner needs access to a *contiguous* block of 1 GB memory). This is most likely to happen if virtual memory is turned off (the page file size is set to 0). If CodonCode Aligner fails to start, quit other open applications, and try again. If the problem persists, please contact CodonCode Corporation.

## Changing memory on OS X

If you are working with large projects on OS X, and Aligner runs out of memory, you can try to increase the memory Aligner can use using this panel.

Changing the memory requires that CodonCode Aligner restarts. Therefore, you can not change the memory if a project has been edited; you need to save your changes first. If your project has unsaved changes, this preference panel will be disabled.

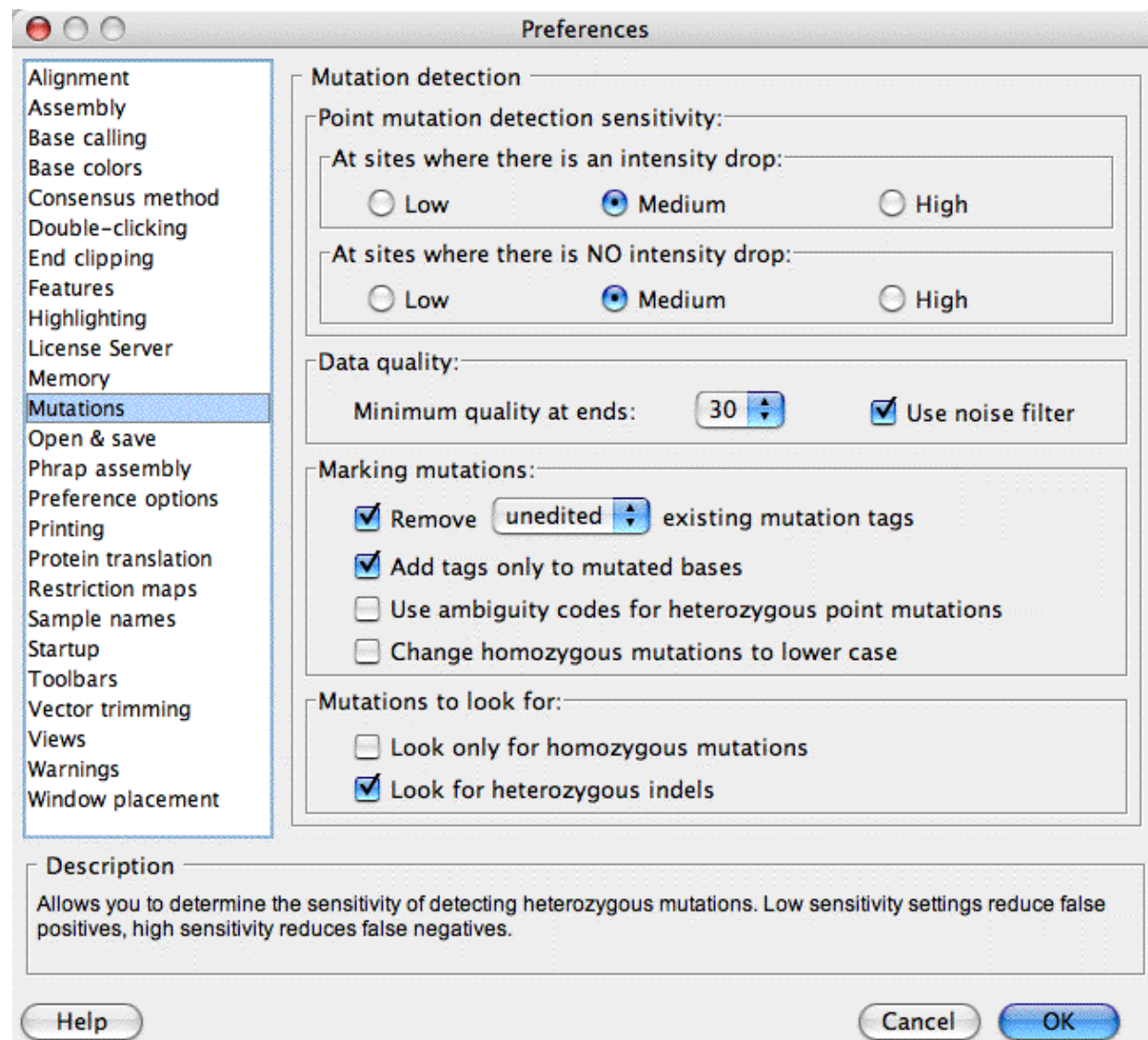
The maximum amount of memory you can set here is determined by the amount of physical memory (RAM) installed in your computer. On OS X, Aligner will determine this amount, and limit the memory you can set to the installed memory.

If you are working on a computer with a limited amount of physical memory (e.g. 2 GB or less), or if you have a number of other applications open, you may experience slow-downs due to "swapping", similar to what is described above in the "Memory on Windows" section. In this case, closing other applications and

installing more memory can lead to dramatic performance improvements.

# Mutation Detection Preferences

To change the mutation preferences, select "**Preferences**" from the "**Edit**" menu on Windows, or from the "**CodonCode Aligner**" menu on OS X, and then click on "Mutations" on the left panel. The mutation preferences dialog looks like this:





## Detection sensitivity

At the top, you can set the **sensitivity** for detection of heterozygous point mutations. The first set of radio buttons determines the sensitivity at places where a secondary peak is accompanied by a drop in intensity of the primary peak, compared to other samples at this position. The second row of radio buttons applies to places where there is no clear drop of intensities, for example because all samples at this position are heterozygous.

There is a trade-off between sensitivity and accuracy. With detection sensitivity set to "**low**", most or all point mutations identified by Aligner will be "real" - in other words, the false-positive rate will be low. However, Aligner may miss some heterozygous point mutations, for example if the secondary peak is weak. On the other hand, a "**high**" sensitivity setting will miss few, if any, heterozygous point mutations, but may incorrectly identify some secondary peaks as heterozygous point mutations when they are really caused by "noise" in the sequence traces.

Your choices in the "**Data quality**" panel also affect the false-positive to false-negative balance. At the start and end of sequence traces, secondary noise peaks are most common, which can lead to false positives. Aligner tries to reduce this problem by examining sequence quality at the start and end of each trace, and excluding low-quality regions. You can determine the stringency of this step by setting the "**Minimum quality at start and end**". The value chosen there is the sequence quality ("Phred quality") required before Aligner starts analyzing; choosing higher values, e.g. 40, will give fewer errors near the end of sequences. *You can see which regions Aligner excluded from analysis by looking at the "dataNeeded" tags, which are shown as yellow boxes (depending on your [highlighting preferences](#)).*

In general, the "**medium**" sensitivity settings should work reasonably well; however, you should adapt the settings to the particular needs of your project and the quality of your sequence data. If identifying every single potential point mutation is important, select "high" in the sensitivity check boxes, and uncheck the "Use noise filter" check box. We strongly suggest that you try the different settings with your own data to find out which setting works best for you.

Note that most detection sensitivity settings only apply to samples that have chromatograms; for text samples, the bases can be analyzed as they are. The only exception is the "Data quality" section - low-quality sequence at the ends of text sequences will also be ignored if the text sequences have quality scores.

## Marking mutations

The section labeled "Marking mutations" lets you fine-tune how CodonCode Aligner marks mutations it finds. The main method Aligner uses if to add tags to mutated bases. To facilitate analysis with other programs, however, Aligner can also convert heterozygous mutations to ambiguities (e.g. using "R" for heterozygous A+G), and change the case of homozygous mutations to lower-case (e.g. changing "A" to "a").

Aligner will add mutation tags at bases where it finds a putative heterozygous or homozygous mutation - except at bases where mutation tags have been edited or confirmed by the user, and kept between successive rounds of mutation detection, as explained in the next paragraph.

If you perform mutation finding more than once, you should check the "**Remove existing mutation tags**" checkbox. If this checkbox is checked, Aligner will remove existing mutation tags in the samples that you have selected before finding mutations. You can either remove only unedited tags or all tags. If "unedited" is selected, any tags edited or added by users (and not the program) will remain; this includes confirmed tags, and tags where the tag type was changed, for example from homozygous to heterozygous. When removing

tags from previous mutation detection rounds, Aligner will also undo automatic edits (like changes to ambiguities) done during previous "Find mutation" cycles.

If the **"Add tags only to mutated bases"** checkbox is checked, Aligner will add tags only to bases that differ from the consensus sequence. This is useful if, for example, you are looking for rare mutations (SNPs). In other cases, for example when genotyping, you may want to add a tag to all samples at each consensus position where a mutation is found; to do so, make sure the **"Add tags only to mutated bases"** checkbox is **not** checked. Then, Aligner will add tags that characterize the base at a given consensus position to all samples, even if just one sample out of a hundred differs from the consensus base.

The last two checkboxes in the "Marking mutations" section determine if Aligner will change base calls at homo- or heterozygous mutations. The default behaviour is that Aligner will leave the bases unchanged, and only add tags at mutated bases. If you want to export your data for subsequent analysis with other programs (for example the population genetics program [Arlequin](#)), you may need to have ambiguity codes at heterozygous bases. If you mark the checkbox **"Use ambiguity codes for heterozygous point mutations"**, Aligner will change the call for all heterozygous point mutations it identifies to [IUPAC ambiguity codes](#).

You may also need to be able to identify homozygous mutations in exported sequences. If you mark the checkbox **"Change homozygous mutations to lower case"**, Aligner will change the base calls of all homozygous mutations it identifies to lower case (e.g. from "A" to "a").

**Please note:** the changed base calls and the tags are currently not linked - if you would want to manually change a base that Aligner identified incorrectly as a heterozygote, you will need to change **both** the base **and** the tag.

## Finding only homozygous mutations

In some projects, you may know that you do not have any heterozygous mutations - for example when you are sequencing clones rather than PCR products. When looking for mutations in such project, make sure that the checkbox "look only for homozygous mutations" is checked.

Of course, when looking for homozygous mutations, you could just look for any discrepancies rather than using Aligner's mutation finding. However, there are several reasons to use Aligner's "Find Mutations":

- Aligner can automatically ignore the low-quality regions at the end (as defined in the "Data quality" pulldown)
- Aligner adds mutation tags, which can be printed and exported for analysis in other programs like Microsoft Excel

## Finding indels

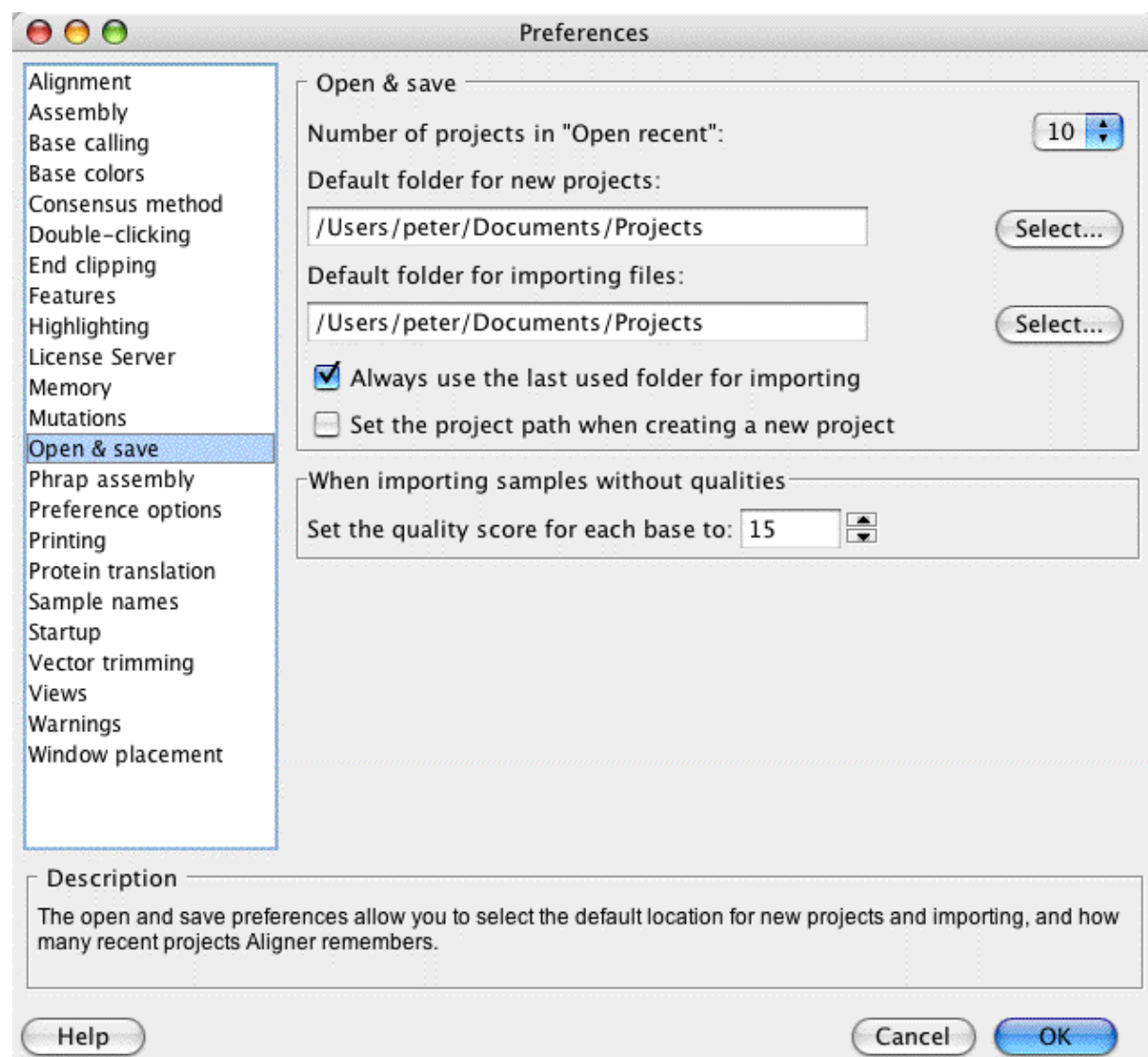
When looking for mutations, you have the option to also look for heterozygous insertions and deletions by checking **"Look for heterozygous indels"**. If you have already searched for heterozygous insertions and deletions (for example before end clipping), you can uncheck this checkbox, and the mutation finding will be a bit faster.

Keep in mind that **it is generally recommended to search from heterozygous indels before end clipping**, since end clipping may clip the entire heterozygous part.



# Open & Save Preferences

In the "Open & save" preferences, you can set what Aligner does when opening and saving projects:



Specifically, you can:

- determine how many recent projects CodonCode Aligner remembers and displays in the "Open recent" menu
- set default folders for saving your projects
- set default folders for importing
- tell Aligner to remember the last location from where you imported files, so that Aligner will return to this folder the next time you choose an "Import" command ("Add Samples", "Add Folder", or "Add Assembly")
- specify if you want to set the name and path of a project right away when creating new projects (if not, the name and location will be set the first time you save a project)
- set the default quality that will be assigned to sequences without qualities when they are imported

The default quality for sequences without qualities you set here will be applied to any sequences imported later where Aligner determines that the sequence has no qualities, or has artificial qualities. This value is used whenever:

- you import sequences from text files (for example FASTA files) that do not have qualities
- you import sequences from files like PHD or SCF files that have qualities, but Aligner determines that the qualities are artificial because:
  - all qualities are the 0 or -1 (for example, if an SCF file was created from an ABI file without qualities)
  - all qualities have exactly the same value, and the sequence is at least 15 bases long (for example an artificial SCF file created with tools like `mktrace` or `fasta2Phd.perl`)
  - a sequence from a PDF file was previously identified as having artificial sequences by Aligner, and has a corresponding tag in the PHD file

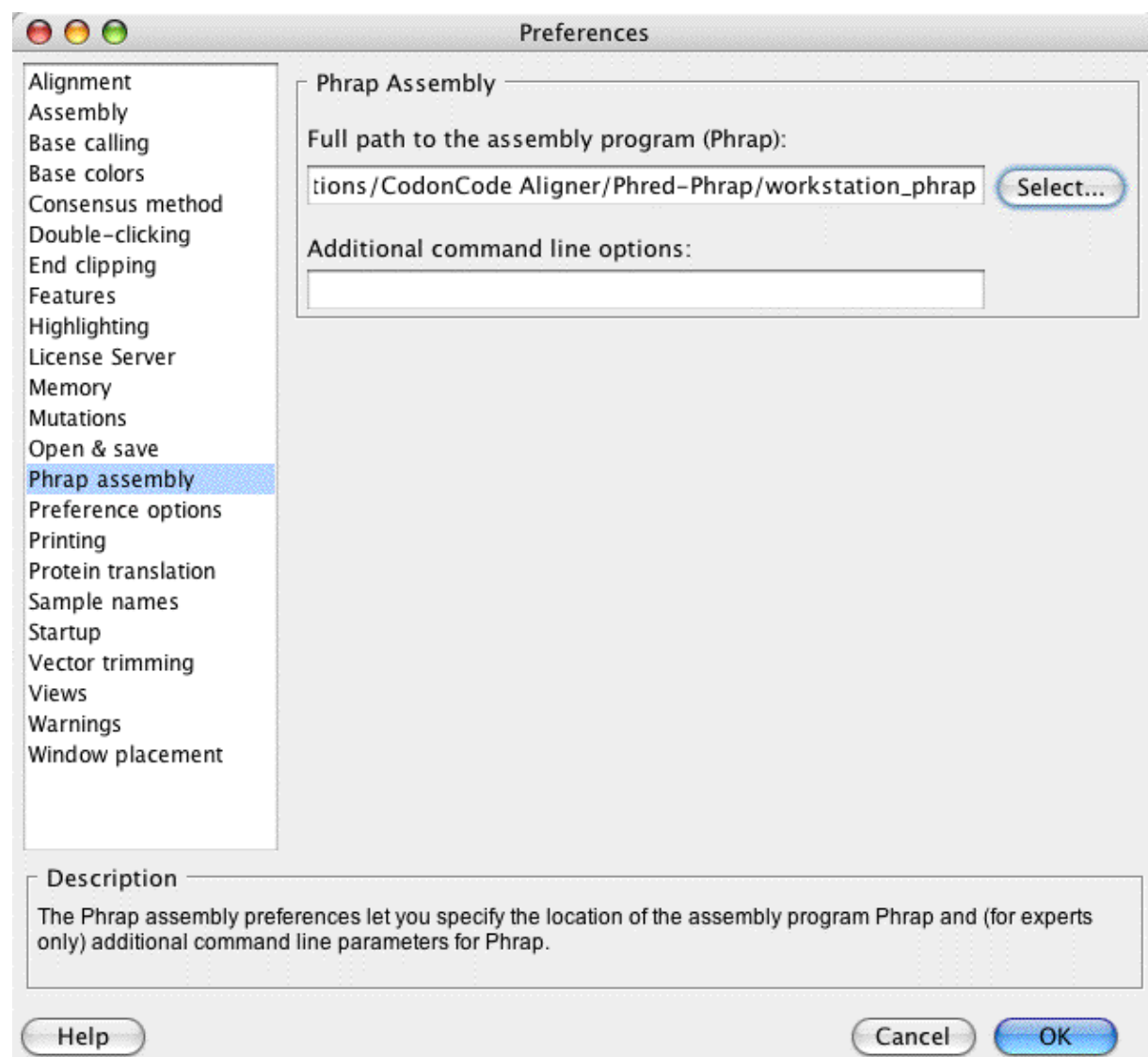
Changing this value will **not** affect any sequences you previously imported. If you want to change the assigned artificial quality values for a sequence in your project, you currently have to remove the sequence, and then import it again.

*What are artificial qualities good for?*

Good question. They cannot be used for end clipping, since end clipping requires real quality scores assigned by programs like Phred. However, the qualities are used to determine the consensus sequence during assembly and alignment to a reference sequence. In general, you should use low artificial scores (like the default value of 15). However, if you want to make sure that a sequence without real qualities is used to contribute more to the consensus, you can assign higher values, all the way up to 90.

# Phrap Assembly Preferences

In the Phrap assembly preferences, you can specify details needed for assembling samples with [Phrap](#):



## CodonCode Aligner User Manual

You specify the location and name of Phrap assembly program in the upper text field.

If you want to use the workstation version of Phrap that was installed with Aligner, the path to Phrap should be correct, and not need any changes. However, if you use your own installation of Phrap, you may need to specify the location of the file on your system.

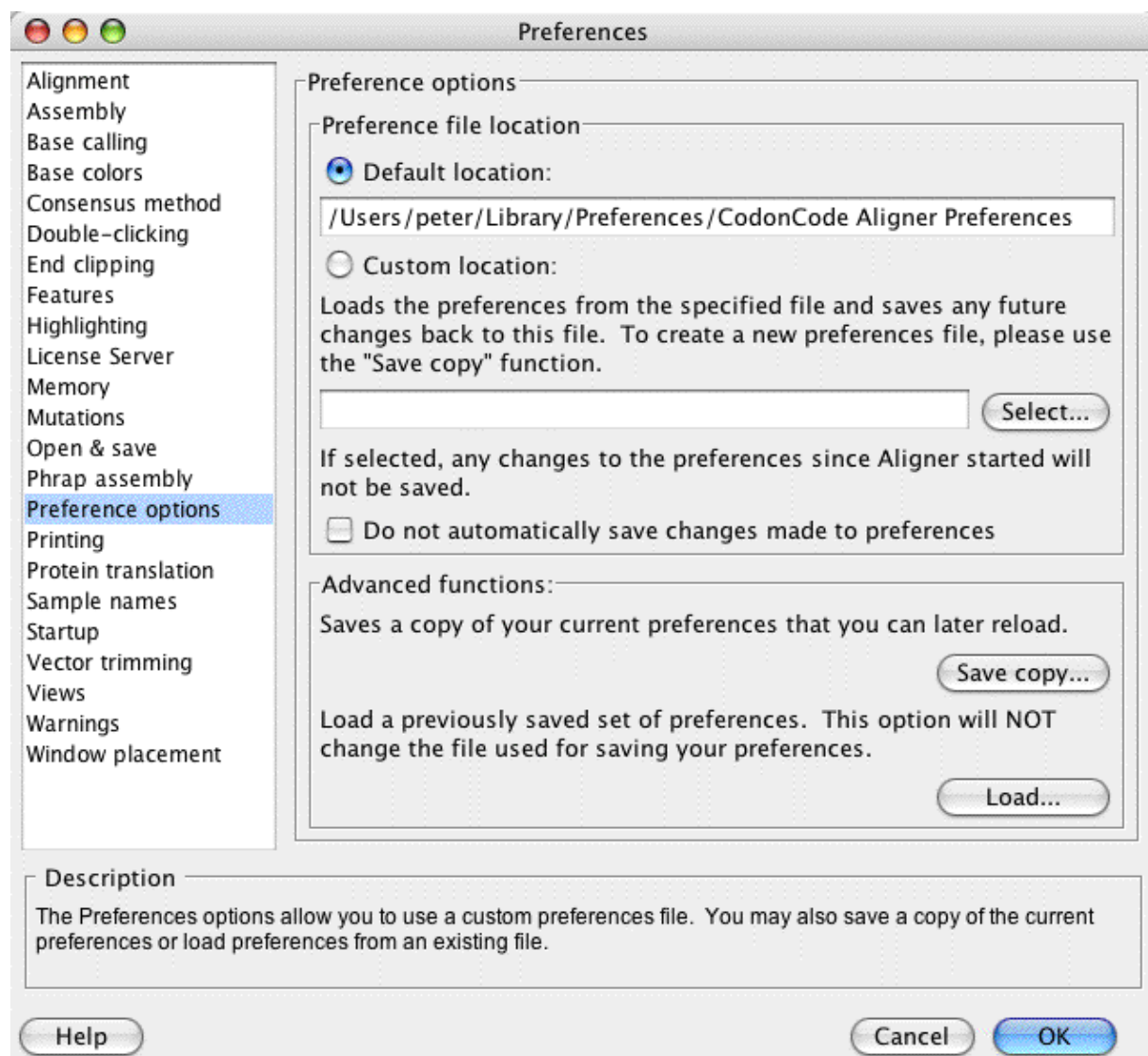
Please note that using Phrap from Aligner requires that you have a trial license or a purchased license; Phrap use is not enabled in demo mode. Academic users who purchased a license for CodonCode Aligner can use the workstation version of Phrap free of charge for academic research; users at companies will have to purchase a separate license to use Phrap.

In the bottom text field, you can specify additional command line options for Phrap. In general, please leave this line blank - **only experts who really know what they are doing should specify command line options** here!

For more information about assembling with Phrap, please read the "[Sequence Assembly With Phrap](#)" section.

# Preference Options

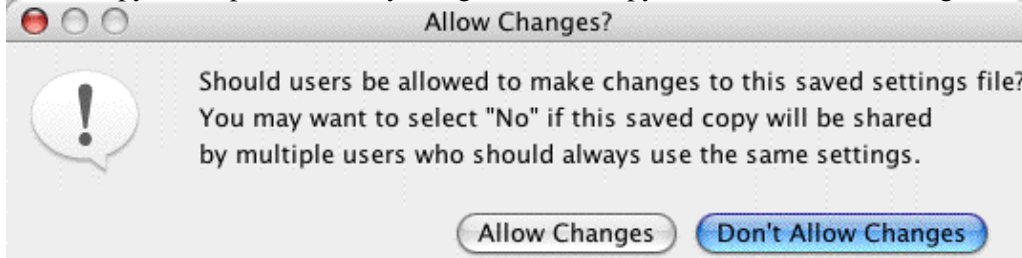
CodonCode Aligner offers the option to save and load copies of Aligner's preferences in the "Preference options" panel of the Preferences dialog:





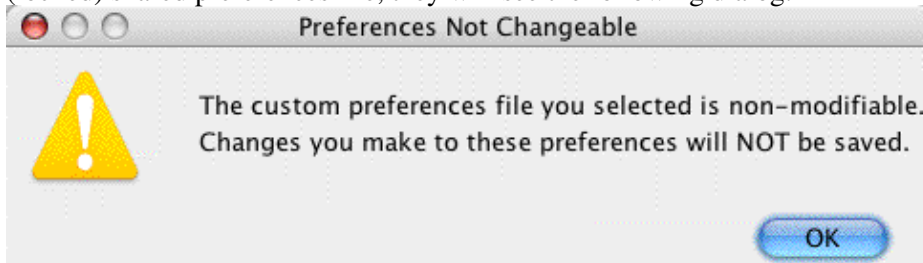
An example of typical use is a group of **several scientists who want to share a "locked" set of preferences**, so that all analyses will be performed with the same settings. This can be done as follows:

1. Change all preferences to the desired settings.
2. Open the Preferences dialog, and select "Preference options" on the left.
3. Save a copy of the preferences by using the "Save copy..." button. The following dialog will appear:



If you want to protect the saved file against changes, press the "Don't Allow Changes" button. A "Save As.." dialog will be shown next, where you can choose the name and location the preferences are saved to. Choose a folder that all users can access, for example on a file server or shared disk.

4. Each user that wants to use the shared preferences now has to choose this file by starting CodonCode Aligner, opening the Preference options dialog, and clicking the "Select..." button. After selection the (locked) shared preferences file, they will see the following dialog:



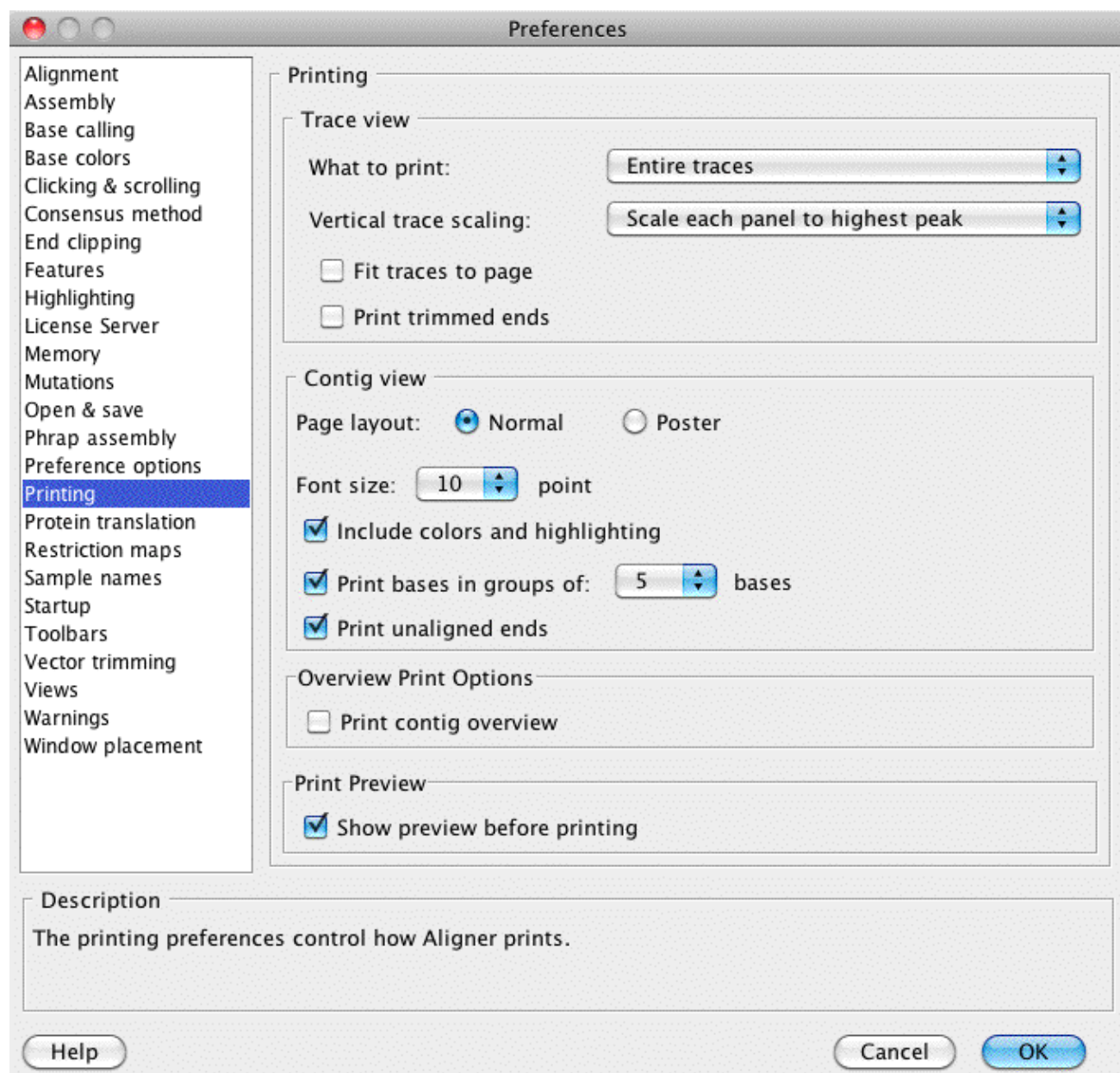
After clicking "OK" in this dialog and then also in the Preferences dialog, the shared preferences will be used. Note that each user can still change settings while Aligner is running; however, such changes will not be saved when Aligner quits.

To change back to separate preferences, a user can select the "Default location" button.

# Printing Preferences

The Printing Preferences allows you to determine how different windows will be printed.

To see the printing preferences, select "**Preferences**" from the "**Edit**" menu on Windows, respectively the "**CodonCode Aligner**" menu on OS X. On the left side of the preference dialog, click on "Printing"; your dialog should now look like this:



Currently, you can only set options for [trace view printing](#) and [contig view printing](#). If you would like more options for printing other views, please let us know!

## Trace View Printing

**What to print** when printing traces is determined in the first pull-down menu. Your options are:

- **Entire traces** - this will print the entire sequence trace for each sample currently in the trace view. Each sample will be printed using several panels on one or more pages before the next sample is printed.
- **Only visible regions** - this will print just the part of the traces that is currently showing on the screen. If your trace view window contains multiple traces, one panel for each trace will be printed, similar to the way the sequences are shown on the screen.

The **vertical trace scaling** lets you select how the traces are scaled when printed:

- **"Use Trace view scaling"** will print traces using a constant scale factor for all panels in a sample; the scale factor used will be the one currently used for this sample in the trace view.
- **"Scale each panel to highest peak"** will calculate a separate scale factor for each panel in each sequence, based on the highest peak in this section of the sequence. This will give you a nicer looking printout (more even peak intensities) when printing entire sequences.
- **"Equalize traces for each panel"** is useful if the intensities for the four bases are very different in your samples. This will calculate separate scaling factors for each of the four traces in each panel, based on the highest peak in this section of the trace.

If you select **"Fit traces on one page"**, Aligner will try to fit everything onto one page, scaling traces as needed. If you chose to print "Entire traces", each sample will be printed onto a separate page.

If the **"Fit traces on one page"** checkbox is not checked, the height of each panel will be determined by the height of the trace view windows defined in your [View Preferences](#) when printing entire traces, or by the height of the panels on the screen when printing only visible regions.

Usually, only the trace between the first called base and the last called base is printed. If you want to also print the trace before the first base and after the last base, make sure **"Print trimmed ends"** is checked.

## Contig View Printing

The **"Page Layout"** used when printing is determined by the first set of radio buttons (*note that this is **not** the same thing as selecting the page orientation (i.e. "Portrait" or "Landscape") in the "Page Setup" dialog!*). The page layout options for contig printing are:

- **Normal** – Prints the sample and contig names with as much of the sequences as will fit across the page. If space permits, additional rows with sample and contig names followed by the sequences will be included on the page.
- **Poster** - Prints the sample and contig names with as much of the consensus sequence as will fit across the page. Continues printing the sequences across additional pages, *without* including the sample or contig names on those pages.

The **"Font size"** pull-down menu selects the size of the font used for printing the contig.

If the “**Include colors and highlighting**” checkbox is checked, the contig will be printed as it appears in the contig view, including colored bases, colored backgrounds, and other highlights. If this box is unchecked, then the bases will be printed as black letters on a white background, ignoring all highlights other than underlining.

To print sequences grouped every 3, 5, or 10 bases, with spaces in between, check the box labeled “**Print bases in groups of # bases**”. If this box is not checked, bases will be printed continuously across the page.

Usually, unaligned ends of samples are printed with the contig. If you do not want to print the unaligned ends, uncheck the “**Print unaligned ends**” checkbox.

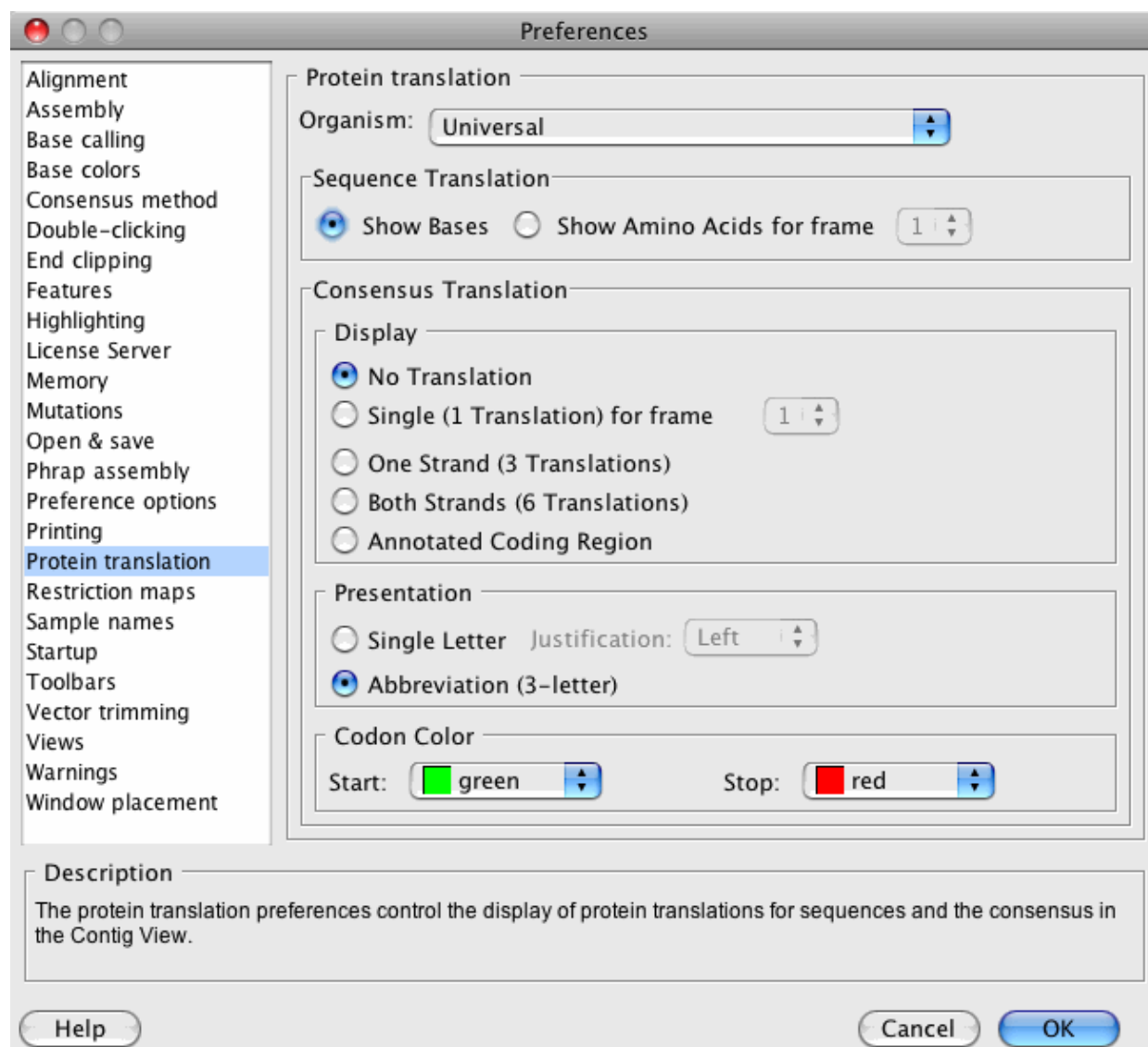
# Protein Translation Preferences

The organism for which the sequence translation is being performed can be selected from the **Organism** list.

In the "**Sequence Translation**" section, you can choose if you want to show the sequences as bases, or as translated as amino acids; this affects the contig, base, and trace views..

The "**Consensus Translation**" section allows you to choose settings for the protein translation of the consensus sequence in the contig view.

Most of these settings can also be selected from the "View" menu.

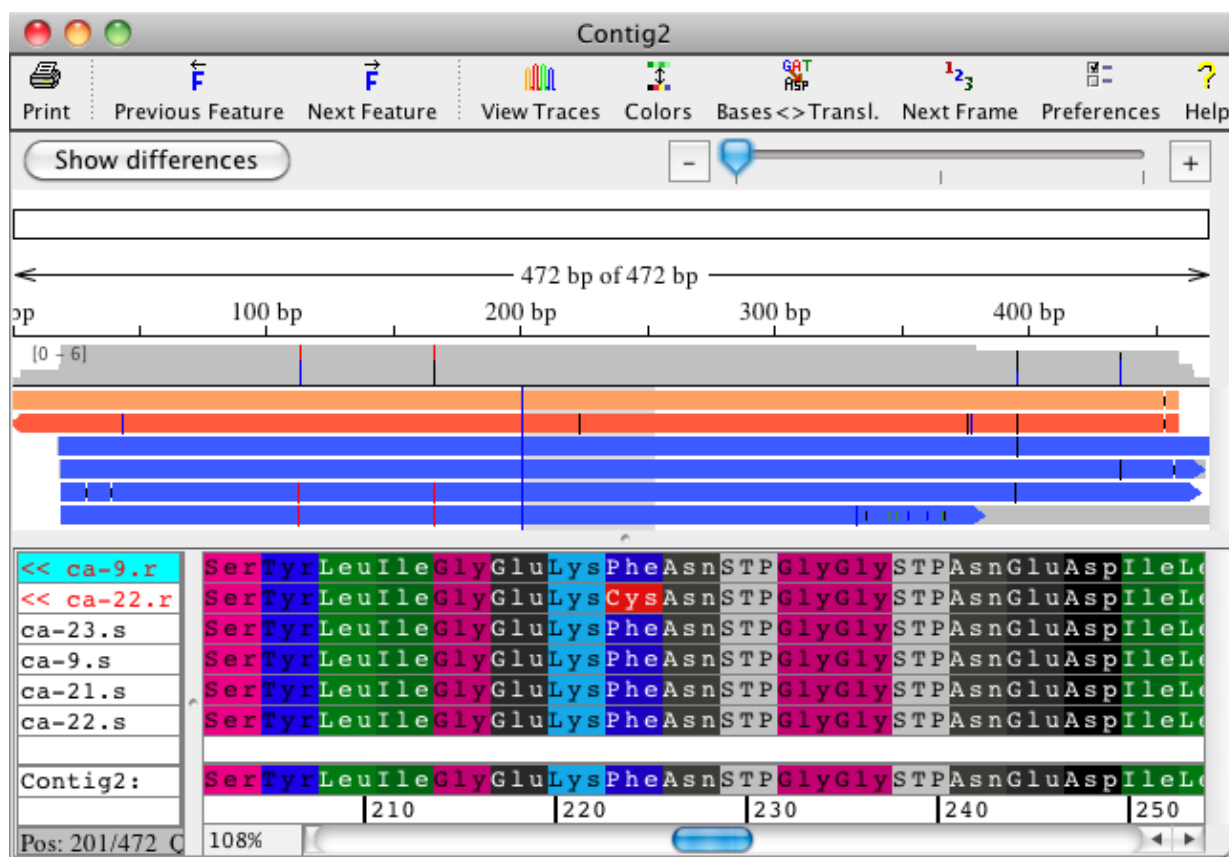


## Sequence Translation


The "Sequence Translation" section of the Protein translation preferences controls if you see bases or amino acids when you look at your sequences. For the amino acid translation, you can choose one of the three possible frames. Changes to which frame to use are only visible if you are showing amino acids and not bases.

The sequence translation preferences control the display of your sequences in Contig View, Base View and Trace View. You can also access these options through the "**Sequence Translation**" submenu in the "**View**" menu.

Here is an example of a Contig View when showing amino acids for sequences:



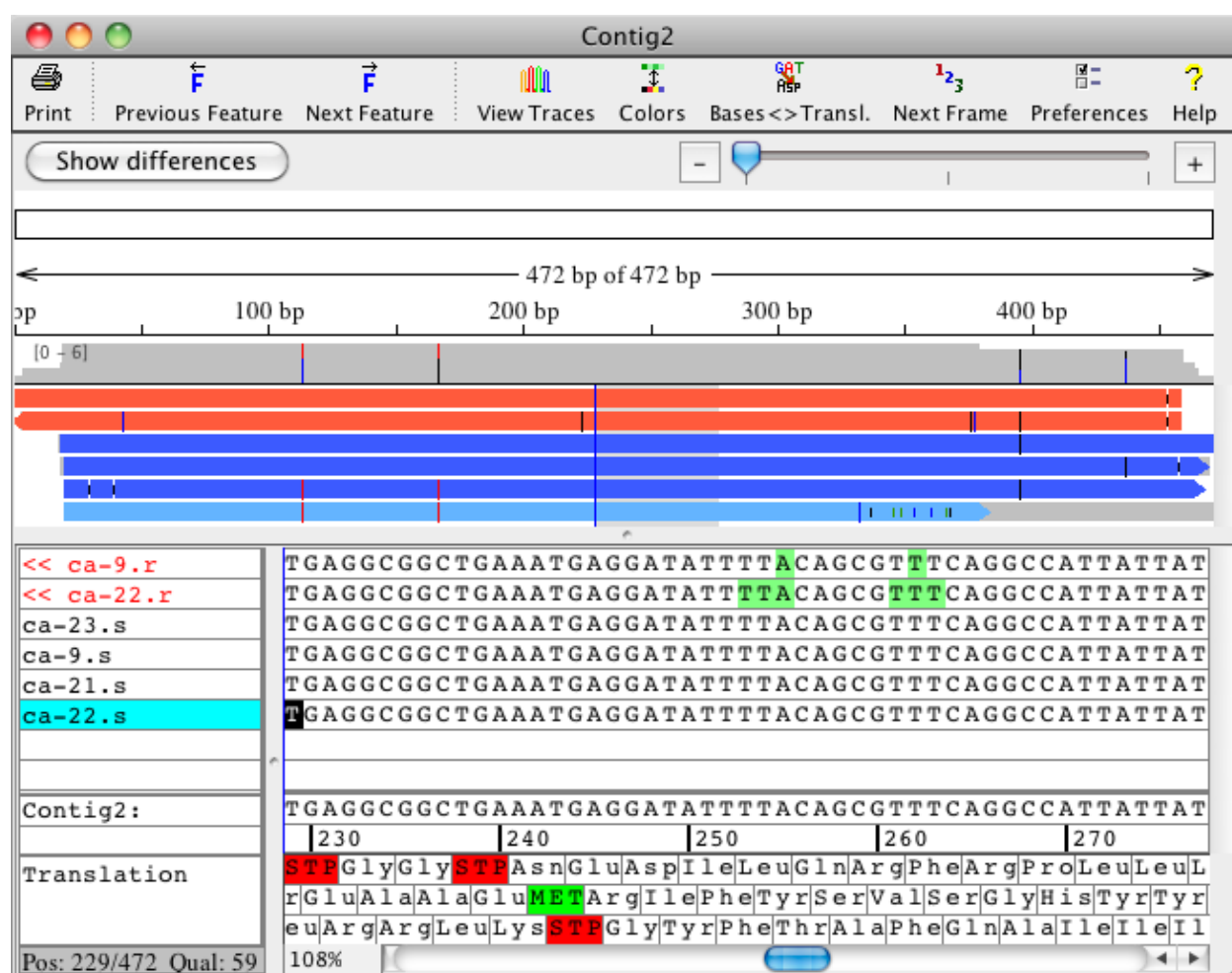
In the picture above, the sequences are drawn by using a translation-based background; this background scheme can be selected from the ["Base Color" preferences](#).

To change the frame used for the translation, you can either use the pulldown menu next to the "Show amino acids for frame" radio button, or use the **"Sequence Translation"** submenu in the **"View"** menu, or use the **"Next Frame"**  toolbar button.

*The translations shown will always use a "fixed" frame, where gap characters in sequences are treated the same way as bases. This is different from the way gaps are treated in the consensus translation described in the next section.*

## Consensus Translation

The consensus translation preferences control the display of the protein translation only for the consensus sequence, which are only visible in the **Contig View**. Here is an example where the consensus translation for all three forward reading frames is shown:



Which consensus translation(s) are shown depends on the selection in the "**Display**" section. You can choose between no translation (the default), the translation of one reading frame, all three forward frames, all six forward and reverse reading frames, and annotated coding regions.

To see the **translation of annotated coding regions**, the reference sequence (for alignments to a reference sequence) or the consensus sequence (for other contigs) must contain "codingSequence" tags that describe which regions are coding. Typically, this display is used together with reference sequences imported from Genbank or EMBL-formatted files; CodonCode Aligner will use the coding sequence annotation from these files if it is present.

Amino acid names are commonly represented by a single letter (A = Alanine) or an abbreviation (Gly = Glycine). The **Single Letter** and **Abbreviation** choices let you set your preference for the consensus protein translation.

The color of start and stop codons in the consensus translation can be changed to make them easier to see. Use the **Start** and **Stop** color lists to select the colors you want.

*For the consensus translation, any gaps in the consensus sequence are ignored in the translation. This is different from the translation of individual sequences described in the previous section.*



# Restriction Map Preferences

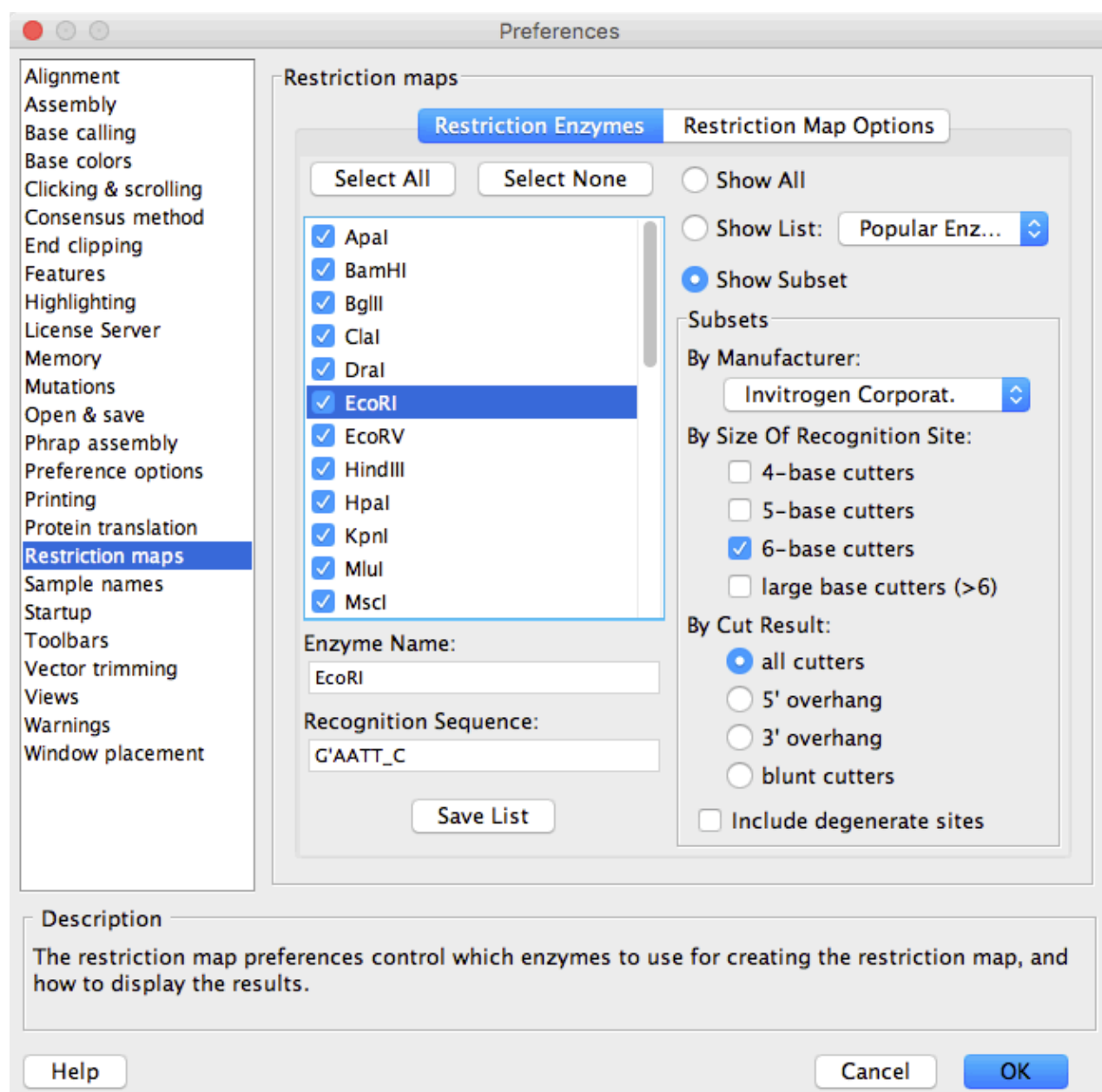
In the restriction map preferences, you specify which enzymes to use for generating restriction maps, and how to display the results.

To change the restriction map preferences select "**Preferences**" from the "**Edit**" menu on Windows, or from the "**CodonCode Aligner**" menu on OS X, and then click on "Restriction maps" on the left panel. You can also access the restriction map preferences from the [restriction map view](#) by clicking on the restriction map icons in the toolbar.

The restriction map preferences are displayed in two different tabs. One tab allows you to change the restriction map options, the other tab allows you to select enzymes for the digest. The tabs are shown at the top of the restriction map preferences and you can switch from one to the other by clicking on the tab.

## Selecting Enzymes

The "Restriction Enzymes" tab in the restriction map preferences allows you to select enzymes for the digest:



The left side of these preferences contains a list of enzymes. This list reflects a set of enzymes that can be chosen on the right side of this view. The three radio buttons on the top right side allow you to show either all enzymes, a pre-saved list, or only a subset of enzymes in the index on the left side. If the "Show Subset" radio button is selected, you can choose your subset using the preferences in the "Subsets" section:

- You can select a subset by **manufacturer**. In the example above only enzymes from Invitrogen Corporation are shown in the index on the left side.
- You have the option to choose a subset of enzymes by the **size of their recognition site**. Above, only 6-base cutters are included in this subset.
- The **cut result** can also be used to specify a specific subset. For example you can include only enzymes that generate a 5' overhang in your subset.
- The "**Include degenerate sites**" checkbox allows you to in- or exclude enzymes that have one or more ambiguities in their recognition site.

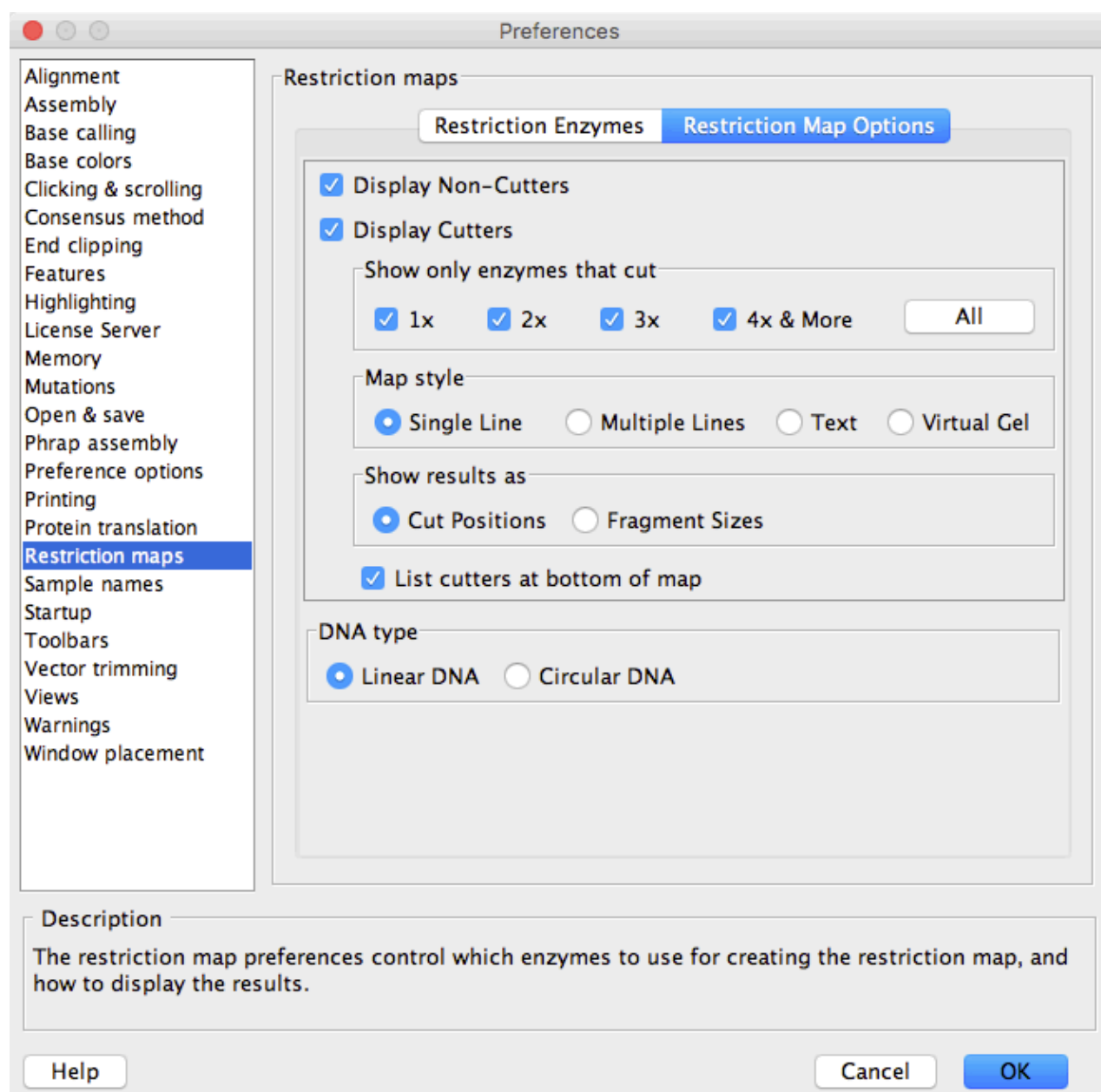
The index of enzymes on the left side updates according to the selection for the subsets. You can select or unselect enzymes in the index through the checkboxes in front of the enzymes, and by using the "Select All" and "Select None" buttons above the index.

The enzyme name and recognition sequence for the enzyme currently selected are shown below the index.

The button "**Save List**" at the bottom left lets you save the selected enzymes above as an enzyme list. The enzymes saved in it can be chosen again by selecting the "**Show List**" radio button on the top right, and selecting the saved list from the drop down menu.

## Restriction Map Options

The "Restriction Map Options" tab in the restriction map preferences allows you to change how the map is displayed, and which DNA type to use:

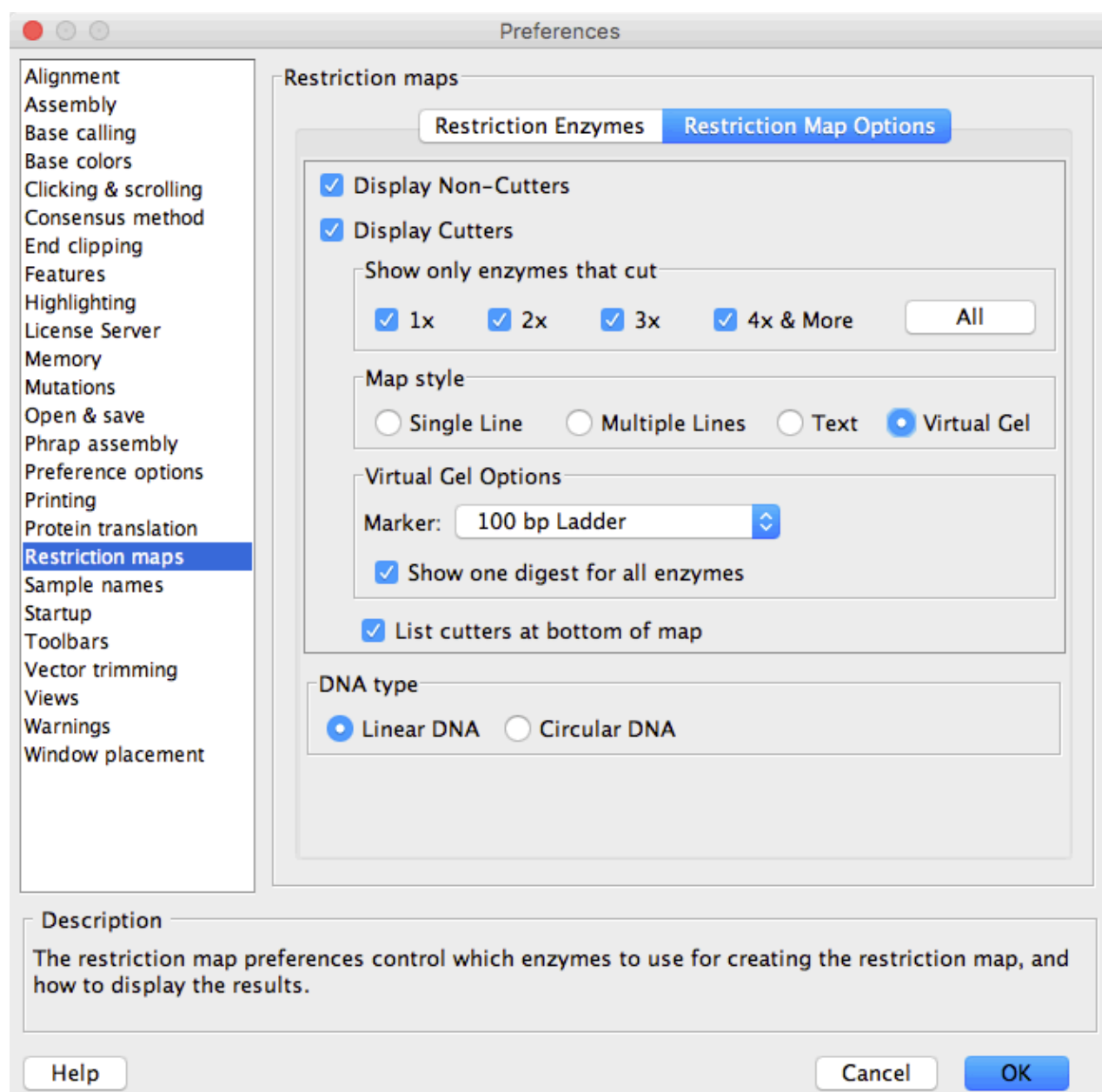


In the first section, you can select if **cutting and/or non-cutting enzymes** should be displayed. If you choose to display cutters, you can set how to show them and the cut results in the restriction map:

- You have the option to show only those enzymes that **cut a specific amount of times**. For example, the restriction map will show only unique cutters if you have only the "1x" checkbox selected.
- You can set the **map style**:
  - ◆ A "Single Line" map displays a graphical map where all enzymes that cut are shown together (on a single line).
  - ◆ The "Multiple Lines" map shows a separate graph for each enzyme that cuts.
  - ◆ The "Text" map lists the cut results in text format.
  - ◆ The "Virtual Gel" shows a simulated gel with one lane for each enzyme.
- The **results** for the single line, multi line and text map can either be displayed as "Cut Positions" or as "Fragment Sizes". The cut positions and fragment sizes are shown in base pairs.
- You also have the option to **list the cutters** in a summary at the bottom of the map. The summary shows the cutters sorted by the number of cuts. A summary of non-cutters is always listed if the "Display Non-Cutters" checkbox is selected.

In the second section you set the **DNA type** used for the digest to be either linear or circular DNA. This can affect your results for fragment sizes and cut positions shown in the map.

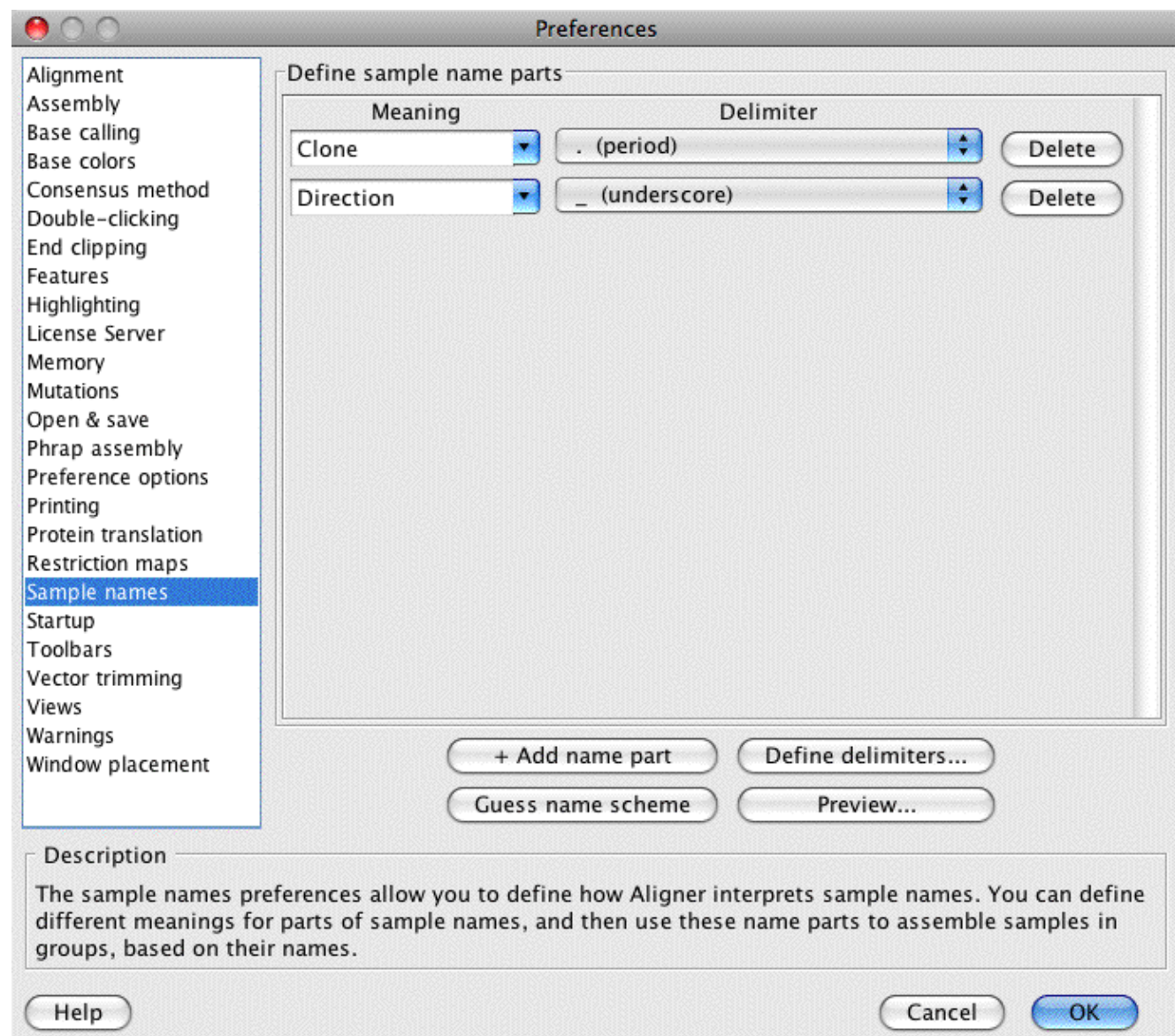
For the virtual gel you can also choose the marker to use in the gel, and show one lane with a digest for all selected enzymes. Fragments in the gel are always shown in base pairs:



# Sample Name Preferences

## Defining sample names

CodonCode Aligner can interpret ("parse") sample names to automatically group samples, and assemble samples in groups. You can define how sample names should be parsed in the Sample name preferences:



## CodonCode Aligner User Manual

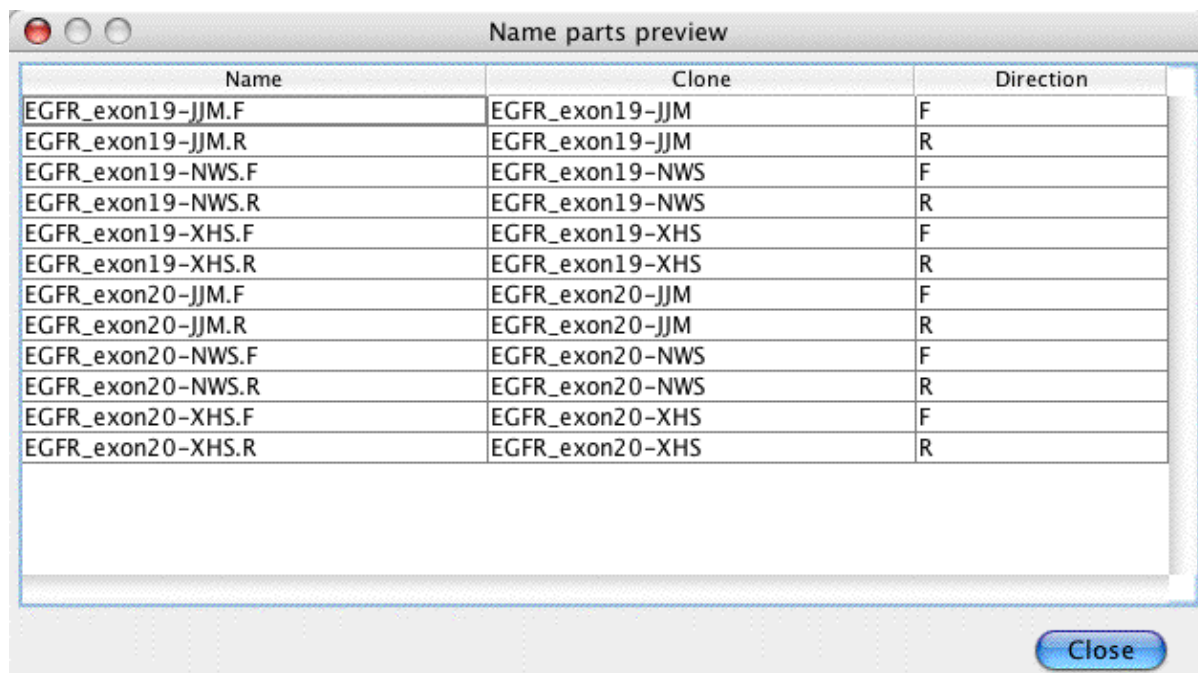
In the screen shot above, Aligner would interpret samples as follows:

- Everything from the beginning of a sample name to the first period ('.') would be interpreted as the clone name.
- The part after the first period up to the next underscore would be interpreted as the direction (typically, 'F' would indicate forward reads and 'R' reverse reads).
- Anything that follows after the underscore will simply be ignored.

You change the meaning of a name part, and the delimiter that indicates the end of a name part, using the respective combo boxes. You can add more name parts using the "+ Add name part" button, and delete name parts using the "**Delete**" button next to the part you want to delete.

Instead of defining your name scheme manually, you can use the "**Guess name scheme**" button to have CodonCode Aligner guess how to interpret the sample names in your project. Guessing the name scheme will work only if your name parts are separated by delimiters (like underscores, dashes, or periods). You can, of course, always use the guessed name scheme as a starting point, and modify it as needed.

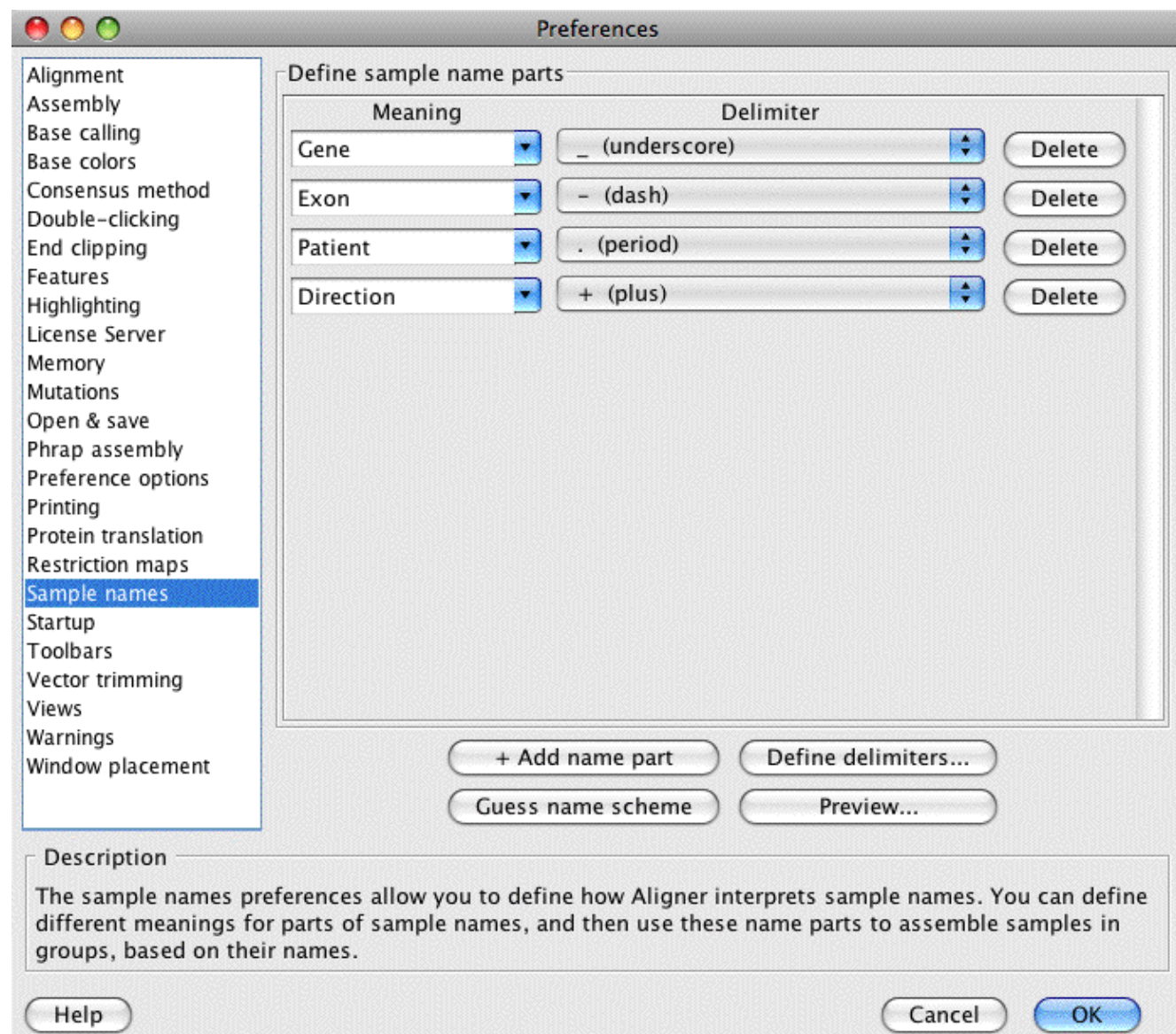
You can view how sample names will be interpreted by clicking on the "**Preview...**" button. This will bring up a preview dialog:

A screenshot of a 'Name parts preview' dialog box. It features a table with three columns: 'Name', 'Clone', and 'Direction'. The table lists 14 sample names, each split into its clone and direction components. For example, 'EGFR\_exon19-JJM.F' is split into 'EGFR\_exon19-JJM' and 'F'. The dialog has a 'Close' button at the bottom right.

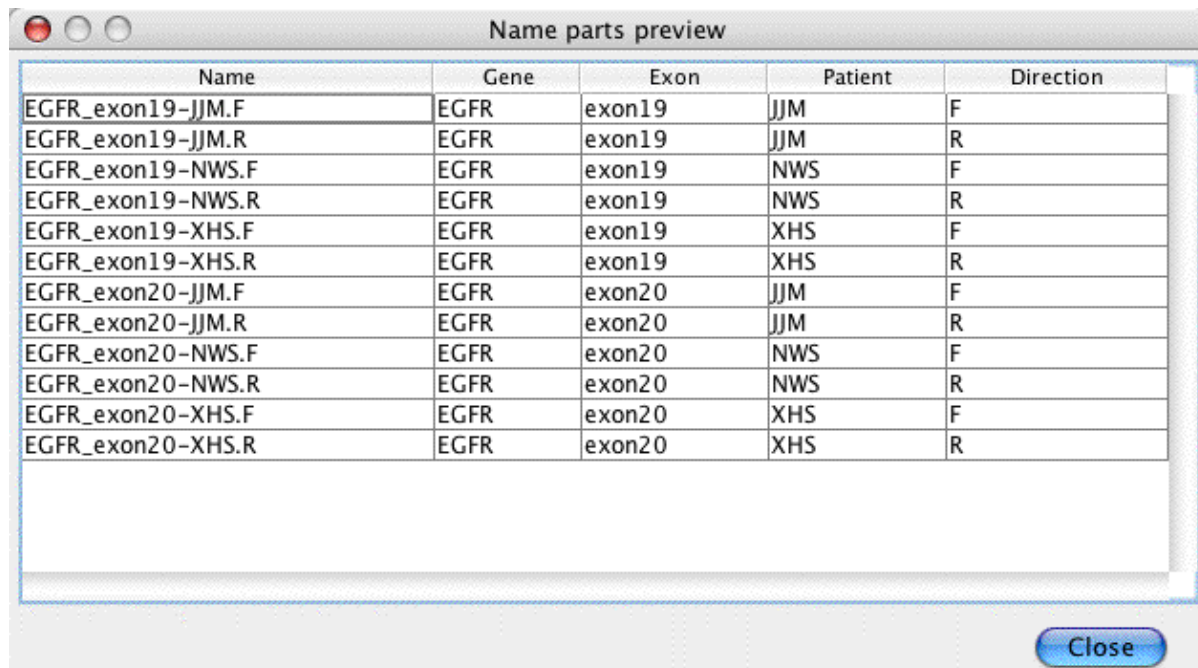
| Name              | Clone           | Direction |
|-------------------|-----------------|-----------|
| EGFR_exon19-JJM.F | EGFR_exon19-JJM | F         |
| EGFR_exon19-JJM.R | EGFR_exon19-JJM | R         |
| EGFR_exon19-NWS.F | EGFR_exon19-NWS | F         |
| EGFR_exon19-NWS.R | EGFR_exon19-NWS | R         |
| EGFR_exon19-XHS.F | EGFR_exon19-XHS | F         |
| EGFR_exon19-XHS.R | EGFR_exon19-XHS | R         |
| EGFR_exon20-JJM.F | EGFR_exon20-JJM | F         |
| EGFR_exon20-JJM.R | EGFR_exon20-JJM | R         |
| EGFR_exon20-NWS.F | EGFR_exon20-NWS | F         |
| EGFR_exon20-NWS.R | EGFR_exon20-NWS | R         |
| EGFR_exon20-XHS.F | EGFR_exon20-XHS | F         |
| EGFR_exon20-XHS.R | EGFR_exon20-XHS | R         |



The default name scheme is clearly not a good one for these clones. It seems that the name starts with the name of the gene (EGFR), followed by the name of the exon and a sample or patient identifier. If we change the name parts definition to look like this:



and then press "**Preview...**" again, we get:



| Name              | Gene | Exon   | Patient | Direction |
|-------------------|------|--------|---------|-----------|
| EGFR_exon19-JJM.F | EGFR | exon19 | JJM     | F         |
| EGFR_exon19-JJM.R | EGFR | exon19 | JJM     | R         |
| EGFR_exon19-NWS.F | EGFR | exon19 | NWS     | F         |
| EGFR_exon19-NWS.R | EGFR | exon19 | NWS     | R         |
| EGFR_exon19-XHS.F | EGFR | exon19 | XHS     | F         |
| EGFR_exon19-XHS.R | EGFR | exon19 | XHS     | R         |
| EGFR_exon20-JJM.F | EGFR | exon20 | JJM     | F         |
| EGFR_exon20-JJM.R | EGFR | exon20 | JJM     | R         |
| EGFR_exon20-NWS.F | EGFR | exon20 | NWS     | F         |
| EGFR_exon20-NWS.R | EGFR | exon20 | NWS     | R         |
| EGFR_exon20-XHS.F | EGFR | exon20 | XHS     | F         |
| EGFR_exon20-XHS.R | EGFR | exon20 | XHS     | R         |

Close

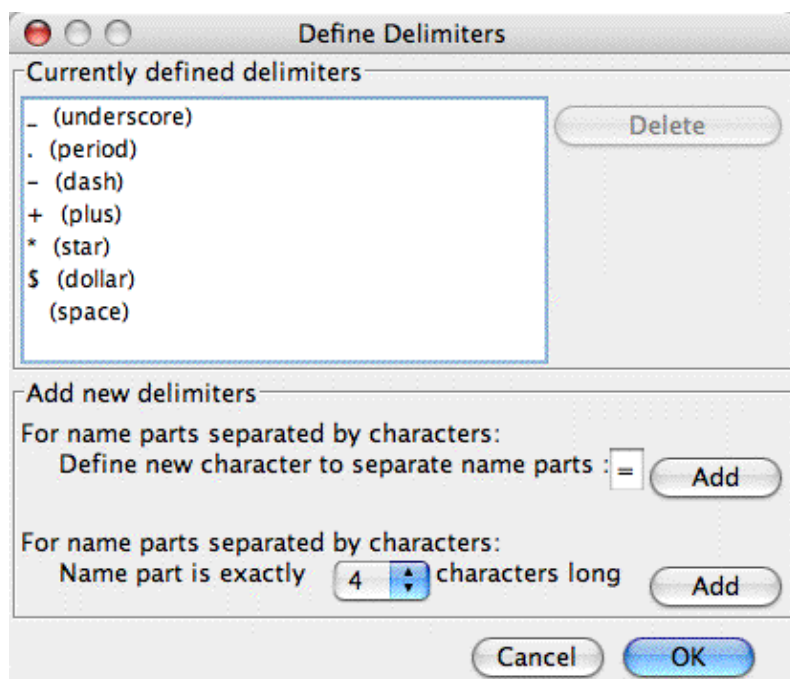
Currently, the only time where CodonCode Aligner uses the name part definition is when [assembling by group](#) (available through "**Assemble with Options...**" in the "**Contig**" menu). When assembling by groups, Aligner can use any of the name parts you define to group samples together for forming contigs - Aligner will try to assemble only samples that belong to the same group. For additional information, please read the [assemble in groups](#) help.

## Defining delimiters

CodonCode Aligner offers two different ways to parse sample names:

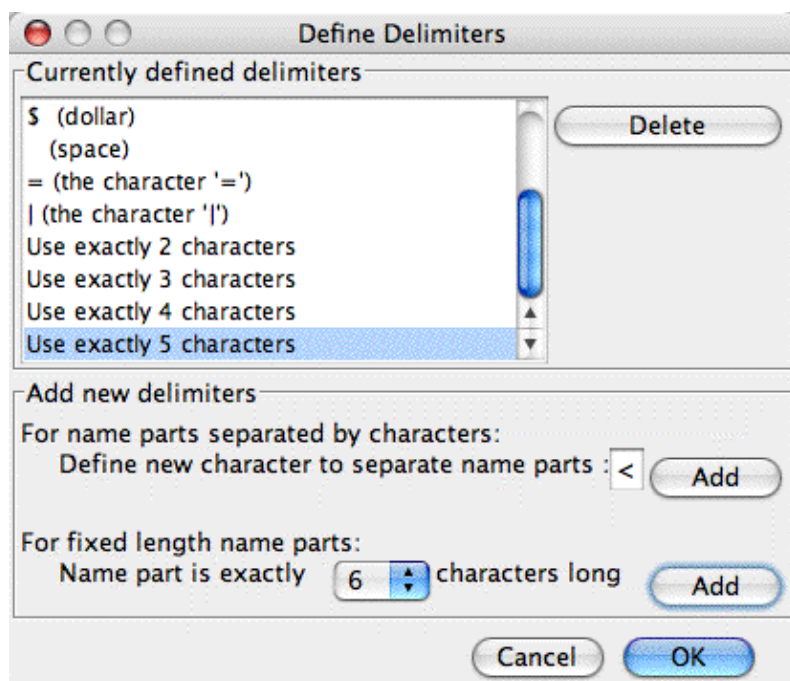
- Using delimiters to separate name parts (as shown in the example above)
- Using a fixed number of characters for a name part

Aligner has several pre-defined delimiter characters; if you want to use a different character as a delimiter, or if you want to use a fixed number of characters, click on the "**Define delimiters...**" button in the sample name preferences (or the "Sample Name Options" dialog from "Assemble With Options"). This will show the following dialog:



To add a new character delimiter, type the character in the text field after "Define new character to separate name parts", and then press the "Add" button next to it.

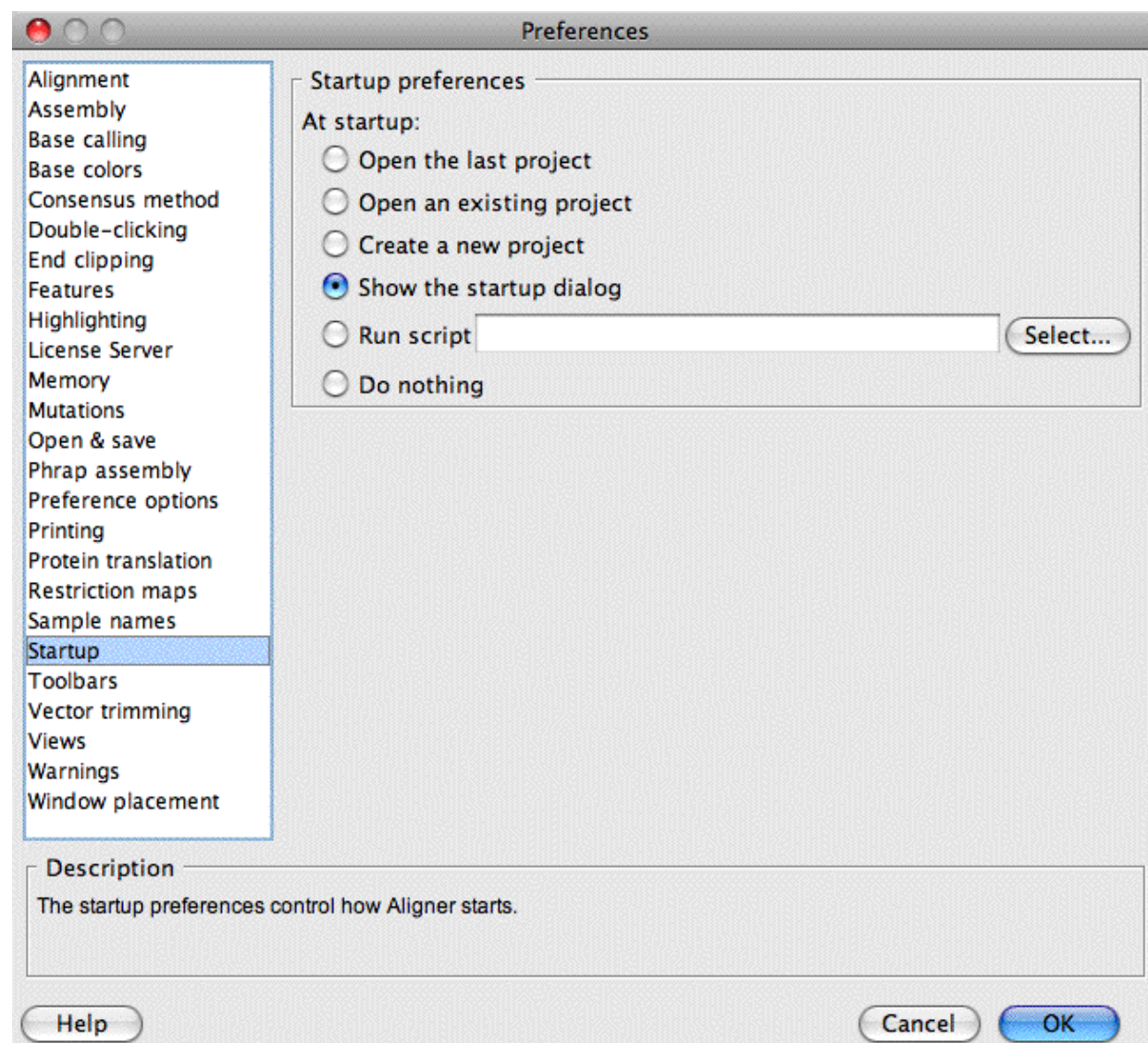
To use fixed length name parts, select the length of your name part in the pulldown menu near the bottom, and then click the "Add" button to the right of it. Repeat this for different lengths, as needed. The screen shot below shows an example of several custom defined delimiters:



After clicking "OK", you will be able to use the newly defined delimiters in the sample name preferences.

# Startup Preferences

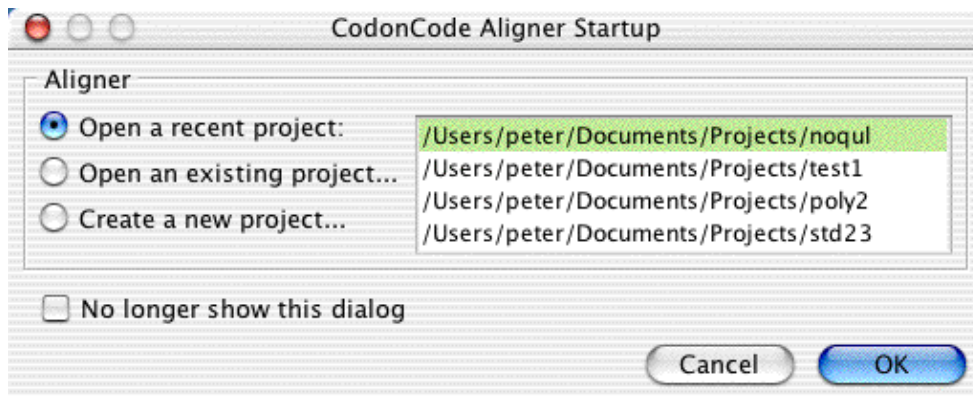
The "Startup Preferences" allow you to control what happens when you start CodonCode Aligner. Aligner can automatically open the last project you worked on, or present dialogs to open existing projects or create a new project, or show the "Startup Wizard" dialog, or run the specified script. The following picture shows the startup preference dialog:



If you select "Do nothing", Aligner will start, but not open any project or show any dialogs. On Windows, you will see an empty Aligner window. On Mac OS X, the only thing you see will be a change in the menu bar; Aligner will not open any windows. You will need to select "Open Project...", "New Project...", or "Open Recent..." before you can do anything else (other than looking at the online help).

## The Startup Dialog

If you choose "Show the startup dialog", the following window will be shown when Aligner starts:



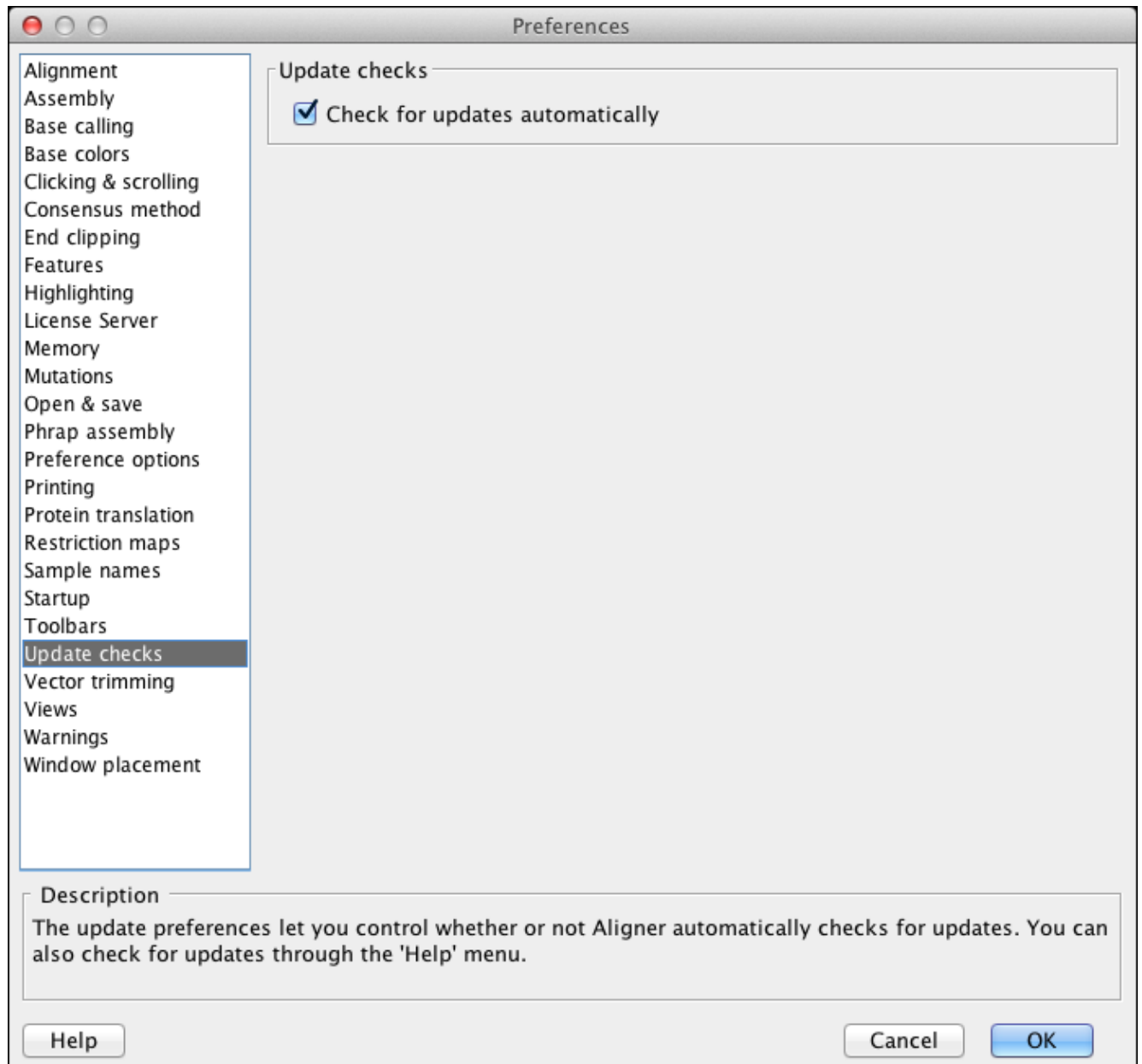
This allows you to quickly open one of *the* projects you worked on recently, or to open another project, or to create a new project.

***Note:** The first time you start CodonCode Aligner, the list of recent projects will be empty. The list will also be empty when you clear the recent project list by choosing "Clear Menu" in the "Open Recent" submenu in the "File" menu. The currently open project and any projects that are currently unavailable will be unavailable ("grayed out") in the "Open Recent" menu.*



# Update Preferences

You can control whether CodonCode Aligner automatically checks for updates in the Update Preference dialog:



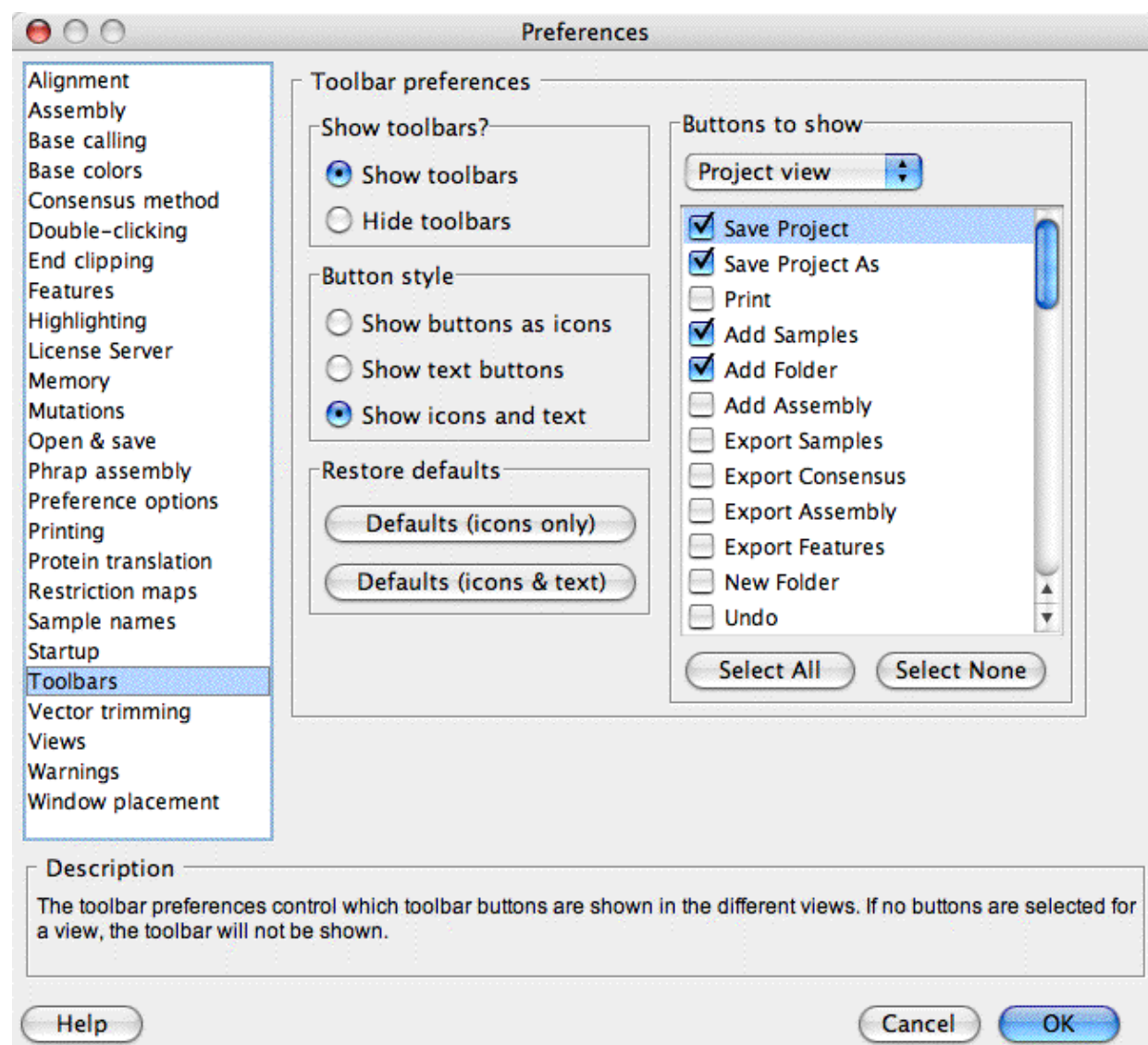
## CodonCode Aligner User Manual

When the checkbox is selected, CodonCode Aligner automatically checks for updates when you start Aligner. Aligner will let you know if a newer version is available, or if there are any problems during the update check. If your version is current (which should be the case most of the time you start Aligner), you will not be notified.



# Toolbar Preferences

The toolbar preferences allow you to customize toolbars for the different views in CodonCode Aligner:



## CodonCode Aligner User Manual

On the left side, you can choose whether or not to show toolbars for all windows, and if toolbars should be shown as buttons, text, or buttons and text. You can also restore the default settings by clicking on one of the "Defaults" buttons. The "Defaults (icons only)" button will restore the initial buttons settings on Windows, where buttons are shown as icons only. The "Defaults (icons & text)" button will restore the initial buttons settings on Mac OS X, where toolbar buttons are by convention shown as icons and text, and therefore need more space. However, you can use either button on both operating systems.

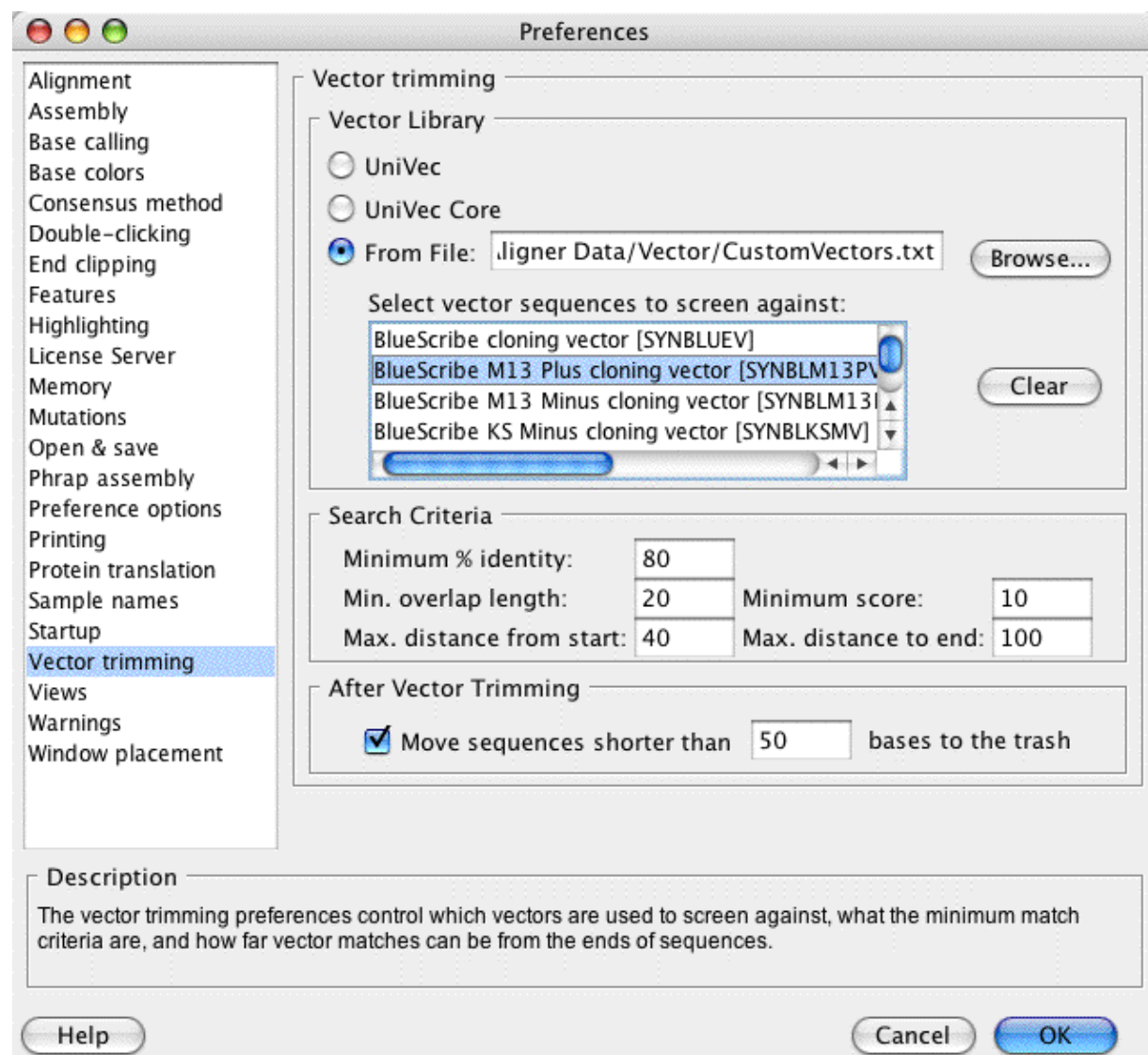
On the right side, you can select which buttons will be shown for the different views. Select the view to customize at the top, then check or uncheck the items you want to see or not see in the toolbar.

If you want to see toolbars in some views, but not in other views, you can simply use the "Select None" button to de-select all buttons for the views where you do not want to see the toolbars.

You can also customize your toolbars directly through the toolbar popup menus in each view.

# Vector Trimming Preferences

In the vector trimming preferences, you specify which vector sequences Aligner screens against, and what the minimum criteria for a vector match are:



To setup or change your vector screening parameters, follow these steps:

1. Choose the "vector library" by pressing one of the radio buttons on the top (the vector library is a collection of vector sequences in FASTA format). It's a good idea to **select your own vector sequence file** by pressing the "Browse..." button. Vector sequence files must be in FASTA format, and can contain one or many sequences.
2. Choose the vector sequences to screen against in the scroll pane below. You can select multiple vector sequences by pressing the shift-key while clicking on a vector name, or the command- or control-key for discontinuous selections. In general, you should select only one or a few vector sequences to screen against.
3. Set the search criteria for the stringency and sensitivity you desire. The settings shown in the image above will work fine for most projects.
4. If you want Aligner to automatically move sequences to the trash if they are too short after vector screening, check the check box at the bottom, and enter a minimum sequence length (for example 100).

When Aligner finds a match between a sample sequence and a vector sequence, it first looks at the minimum match criteria you defined. If the match is not good enough (for example because it is too short), then it will be ignored.

Two parameters that need explanation are the "**Max. distance from start**" and "**Max. distance from end**" numbers. To understand the meaning of these numbers, keep in mind that matches between sample sequences and vector sequences are "local" matches - that is, the matches do not necessarily extend to the beginning or the end of the sample. If there is low-quality sequence at the beginning or end of your samples, errors in these parts will often lead to alignments that start a few bases into the sequence, and/or do not extend all the way to the end. Aligner uses the "Max. distance" parameters to determine how far from the start or end of a sample a match can be. For example, if a match starts 35 bases from the start of the sample, and the maximum distance from the start is set to 50, then Aligner will trim the first unaligned 35 bases, plus the bases that actually match. But if the maximum distance from the start was set to less than 35, Aligner would ignore this match, and not trim.

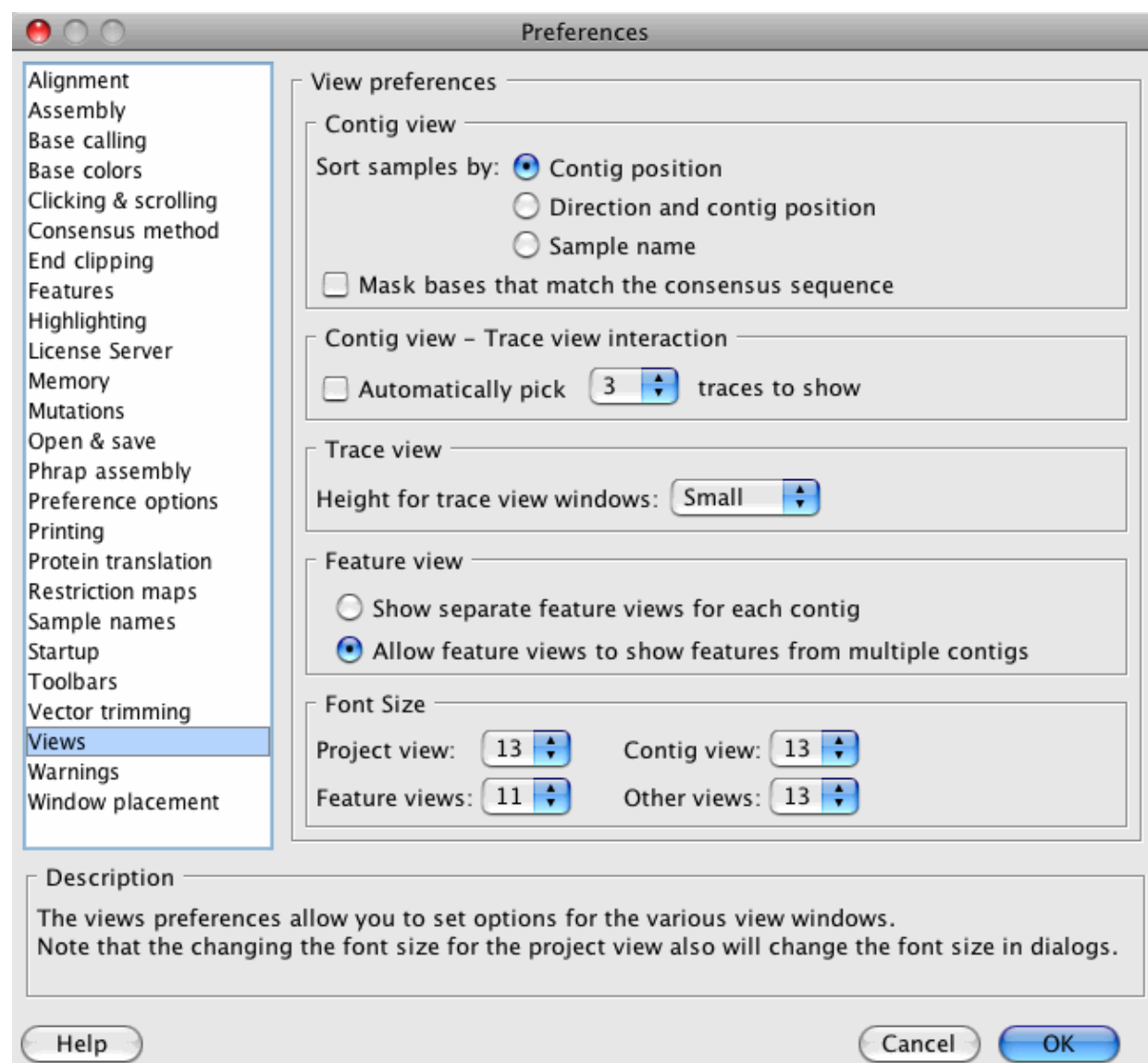
If you use end clipping before vector trimming, you can use low numbers for the "max. distance" parameters, for example 20 and 50. With samples that are not end clipped first, higher numbers (like 50 for the start and 150 - 250 for the end) may make more sense.

The alignment score is calculated based on the number of matches, discrepancies, and insertions/ deletions in a match. Each matching bases gives a score of +1, while mismatches and insertions/deletions reduce the score. Therefore, your minimum score should be lower than the minimum overlap length.

Keep in mind that there is always a trade-off between sensitivity and specificity. If you use very stringent match criteria, you are likely to miss some vector matches; if you use very loose criteria, you may end up trimming sequences from random hits.

# View Preferences

The view preferences allow you to set options for the different views:



In the "**Contig view**" panel at the top, you choose how samples should be sorted in contig views. For most projects, the default of sorting samples **by position** in contigs makes the most sense. Samples will be sorted in ascending order, based on their first aligned base in a contig.

Sometimes, sorting the samples **by name** is more desirable. One example is a mutation detection project where your forward- and reverse reads have similar names (like "Sample123.f" for forward and "Sample123.r" for reverse), and where you would like to have the forward- and reverse-reads right underneath each other.

If you select "By direction and contig position", all reads in forward direction will first be sorted by position, and then, all reads in reverse direction will be sorted.

Changing these preferences here will **not** affect any open contig view windows; it will also **not** affect contigs where you previously sorted samples manually by dragging read names up or down. To change the sorting in open [contig view windows](#) (or to quickly try out the different sort options), open a contig view, and right-click (command-click on OS X) in the "aligned bases" panel to display the contig view [popup](#) menu, then choose "Sort samples by..."

In the "**Contig view-Trace View interaction**" panel, you can choose what happens when you have the contig view and the trace view for a contig open. If the check box before "Automatically pick 3 traces to show" is checked, Aligner will try to pick 3 traces to show every time you move around in a contig. However, please note that Aligner will **not** automatically open the trace view for you. You can choose how many traces Aligner should pick - we suggest to use smaller numbers for faster performance.

In the "**Trace view**" panel, you can set the default height of traces in trace view windows - try it out!

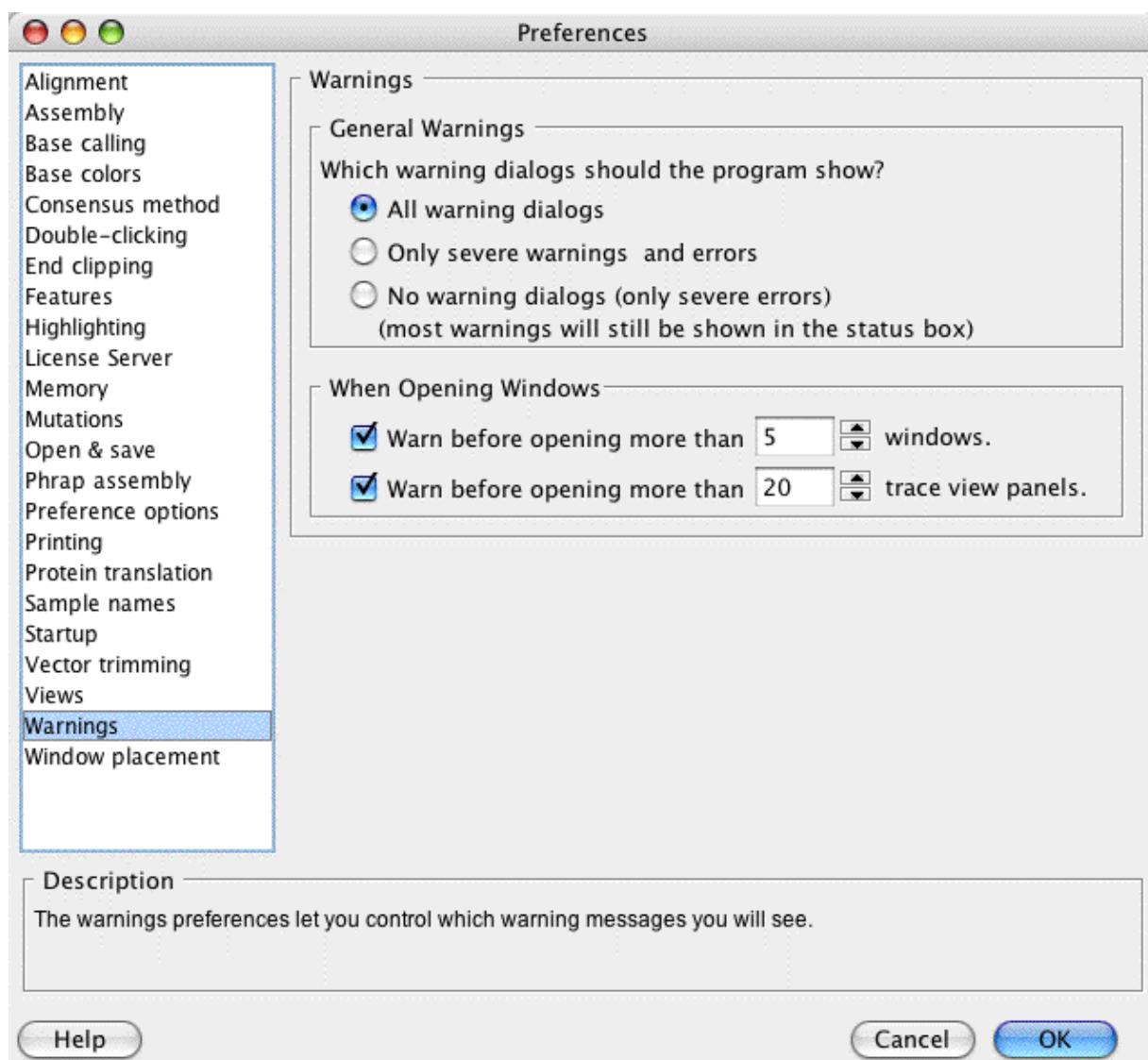
In the "**Feature view**" panel, you can determine how feature views will behave. Since Aligner version 1.1.2, feature views can show the features for more than one contig. For example, you may be interested in viewing all PolyPhred tags in all of your contigs in a single table. You can do this by selecting all contigs, and then selecting "Feature View" in the "View" menu. If you have "Allow feature views to show features from multiple contigs" checked, you will get just one window. But if you have "Show separate feature views for each contig" checked, you will get an individual window for each contig - you guessed that much, didn't you :-)

There's just one thing to notice when you allow feature views to show features for multiple contigs: any feature will be shown in only one window. This means that Aligner may sometimes close older feature views. If, for example, you have a feature view for "Contig1" and "Contig2" open, and then open a feature view for "Contig2" and "Contig3", Aligner will first close the older feature view, and then show you the new feature view for "Contig2" and "Contig3".

In the "**Font size**" panel, you can set the font size used in the different views. The project view settings affect only the project view; the "feature views" settings affect feature views and mutation report windows; the "contig view" settings affect only the contig view; and the "other views" setting affects the trace views, base views, and other views.

# Warning Preferences

You can control how many warnings CodonCode Aligner shows in the Warning Preference dialog:



The upper section affects most warnings that Aligner can show. For new users, we suggest that you display all warning dialogs, as shown above. Once you are familiar with Aligner, you can choose to see only the more severe error messages and warning dialogs by selecting the checkbox in the second row. If you really know what you are doing and do not want to see any warning dialogs, select the checkbox in the third row.

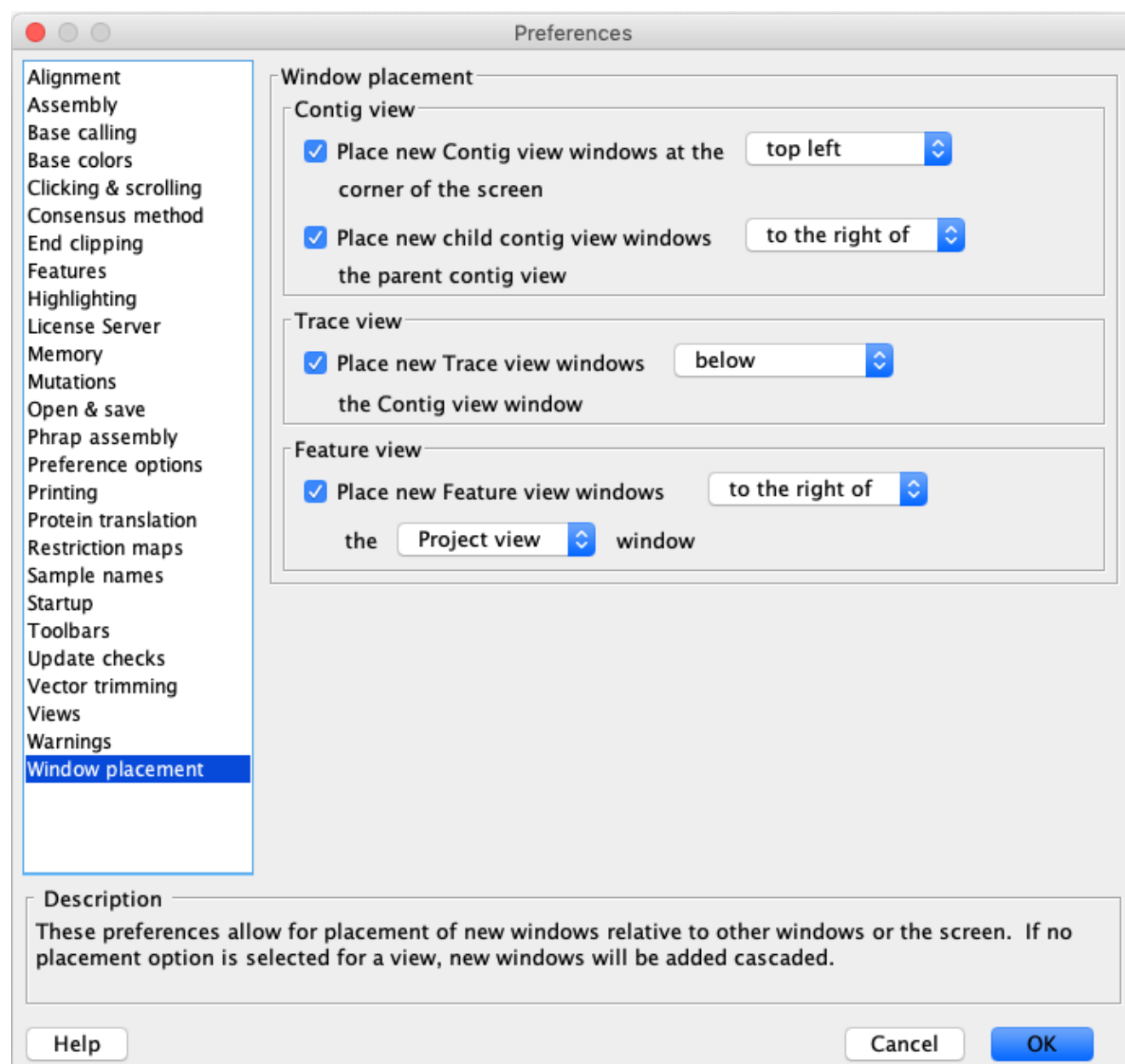
In the lower section, you can choose if Aligner should warn you before opening many windows (for example because you accidentally double-clicked on the "Unassembled Reads" folder, which contains many samples). You can also set the threshold for how many windows can be opened before Aligner shows a warning. *Please note that the warning dialog will not be shown if you turned off all warnings in the "Warnings" preferences.*

Some warning dialogs also allow you to turn off the optional warnings, which has the same effect as selecting "Only severe warnings and errors". *Some dialogs can be turned off individually, while others affect all optional warnings.*



# Window Placement Preferences

You can control how windows are arranged in CodonCode Aligner in the window placement preferences:



## CodonCode Aligner User Manual

Using the check boxes and drop-down menus, you can select where Aligner places contig views, trace views, and feature views when opening new windows.

For contig views, you can select any of the four corners of the screen (or the "root" window on Windows).

"Child" contigs in contigs of contigs (created by Aligner's "Compare Contigs" function) can be placed relative to the "parent" contig view, using the settings in the second line. This requires that the parent contig view is open.

For trace views, you can select where to place trace views relative to the contig view; common choices are to put trace views below or to the right of the corresponding contig view.

For feature views, you can select where to put newly opened feature view windows relative to the project view or relative to the corresponding contig view.

If the check boxes are unchecked, or if a window used for relative placements are not shown, Aligner will add windows cascaded, so that each new window will be a bit lower and to the left of the window that was opened last.

# CodonCode Aligner Help by Menu

[File](#) - [Edit](#) - [Go](#) - [Sample](#) - [Contig](#) - [Tools](#) - [View](#) - [Window](#) - [Help](#)

## Aligner Menu (on Mac OS X only)

[About Aligner..](#)

[Preferences...](#)

[Quit](#)

## File Menu

[New Project](#)

[New Text Sequence...](#)

[Open...](#)

[Open Recent](#)

[Close Project](#)

[Save Project](#)

[Save Project As...](#)

[Import](#)

[Add Samples](#)

[Add Subset of Samples...](#)

[Add Folder](#)

[Add Assembly](#)

[From Genbank...](#)

[Export](#)

[Export Project Summary...](#)

[Export Samples...](#)

[Export Consensus Sequences...](#)

[Export Assembly...](#)

[Aligner Project \(Old Format\)...](#)

[Export Features...](#)

[Export Protein Translation...](#)

[Export Differences...](#)

[Export Trees...](#)

[New Folder...](#)

[Delete Folder...](#)

[Duplicate Samples](#)

[Exit \(on Windows\)](#)

## Edit Menu

[Undo](#)

[Redo](#)

[Copy \(selected sequence\)](#)

[Paste](#)

[Select from Start to Here](#)

[Select from Here to End](#)

- [Select All](#)
- [Select Range](#)
- [Select Samples or Contigs](#)
- [Make Lower Case](#)
- [Make Upper Case](#)
- [Change Bases](#)
  - [Match Consensus](#)
  - [Call Second Peaks](#)
  - [Remove Ambiguities](#)
  - [Change Low Qual Bases](#)
  - [Undo Auto Edits](#)
  - [Change Bases Options...](#)
- [Reverse complement](#)
- [Move to Unassembled Samples](#)
- [Move to Trash](#)
- [Preferences](#) (Windows only)

## Go Menu

- [Define Features...](#)
- [Next Feature](#)
  - Next
    - [High Quality Mismatch](#)
    - [Low Quality Consensus](#)
    - [Ambiguity](#)
    - [Mismatch](#)
    - [Edited Base](#)
- [Previous Feature](#)
  - Previous
    - [High Quality Mismatch](#)
    - [Low Quality Consensus](#)
    - [Ambiguity](#)
    - [Mismatch](#)
    - [Edited Base](#)
- [Base Number...](#)
- [First Aligned Base](#)
- [Last Aligned Base](#)
- [Search Sequence...](#)
- [Search Again](#)
- [BLAST Search](#)
  - [MegaBLAST](#)
  - [Nucleotide \(blastn\)](#)
  - [Translated \(blastx\)](#)
  - [Translated \(tblastx\)](#)

## Sample Menu

- [Call Bases](#)
- [Find Heterozygous Indels](#)
- [Split Heterozygous Indels](#)

- [Clip Ends...](#)
- [Trim Vector...](#)
- [Insert gap](#)
  - [Shift Bases Left](#)
  - [Shift Bases Right](#)
- [Delete](#)
  - [Selection - Fill from Left](#)
  - [Selection - Fill from Right](#)
  - [From Sample Start](#)
  - [To Sample End](#)
- [Move](#)
  - [Gap Left](#)
  - [Gap Right](#)
  - [Sequence Left](#)
  - [Sequence Right](#)
- [Mark](#)
  - [Start Alignment Location](#)
  - [End Alignment Location](#)
- [Tag](#)
  - [Show Local Tags...](#)
  - [Confirm Tag](#)
  - [Mark Tag False Positive](#)
  - [Show All Tags...](#)
  - [Add Tag...](#)
  - [Add Tag to All...](#)
- [Find Common Features...](#)
- [Set Base Number...](#)
- [Synchronize Starts...](#)
- [Make Reference Sequence](#)
- [Label](#)
- [Sample Information...](#)

## Contig Menu

- [Assemble](#)
- [Advanced Assembly](#)
  - [Assemble with Preprocessing...](#)
  - [Assemble in Groups...](#)
  - [Compare Contigs...](#)
  - [Assemble from Scratch](#)
  - [Assemble with Phrap](#)
- [Align with Clustal](#)
- [Align with Muscle](#)
- [Align using Translation \(with MACSE\)](#)
- [Align to Reference Sequence](#)
- [Advanced Reference Alignments](#)
  - [Align with Preprocessing...](#)
  - [Align in Groups...](#)
  - [Align from Scratch](#)
- [Unassemble](#)

- [Rebuild Consensus](#)
- [Find Mutations](#)
- [Process Heterozygous Indels](#)
- [Analyze Methylation](#)
- [Build Tree...](#)
- [Build Tree for Selected Bases...](#)
- [Delete](#)
  - [From Contig Start](#)
  - [To Contig End](#)
- [Remove Consensus Gaps...](#)
- [Split Contig](#)
- [Contig Information...](#)

## Tools Menus

- [Design Primers...](#)
- [RFLP Analysis...](#)
- [Restriction Cloning...](#)
- [Gibson Assembly...](#)
- [Dot Plot](#)
- [NGS Tools](#)
  - [Cluster Sequences...](#)
  - [Identify Sequences...](#)
  - [Trim Reads with BBDuk2...](#)
  - [Error Correct with SparseAssembler..](#)
- [Assemble NGS Data](#)
  - [CodonCode NGS Assembler...](#)
  - [Tadpole...](#)
  - [SparseAssembler...](#)
- [Separate Alleles...](#)
- [Align with Bowtie2...](#)
- [Script](#)
  - [About the Script Menu](#)

## View Menu

- [Contig](#)
- [Traces](#)
- [Bases](#)
- [Qualities](#)
- [Features](#)
- [Restriction Map](#)
- [Select Enzymes...](#)
- [Restriction Map Options...](#)
- [Mask Bases Matching Consensus](#)
- [Auto Select Traces](#)
- [Sequence Colors](#)
  - [Quality - 3 Colors](#)
  - [Quality - Continuous](#)
  - [By Base](#)

- [Translation Based](#)
- [Sequence Translation](#)
- [Show Bases](#)
- [Show Amino Acids](#)
- [Amino acids - frame 1](#)
- [Amino acids - frame 2](#)
- [Amino acids - frame 3](#)
- [Consensus Translation](#)
- [None](#)
- [Frame 1](#)
- [Frame 2](#)
- [Frame 3](#)
- [Frame Forward 3 Frames](#)
- [All 6 Frames](#)
- [Annotated Coding Regions](#)
- [Preferences...](#)
- [Toolbars](#)
- [Show Toolbars](#)
- [Hide Toolbars](#)

## Window Menu

- [Close](#)
- [Close Others](#)

## Help Menu

- [Aligner Help...](#)
- [Quick Tour](#)
- [Tip of the Day...](#)
- [License...](#)
- [Aligner Web Site](#)
- [Check for updates...](#)

# Memory Requirements

CodonCode Aligner is currently intended for projects with up to several thousand chromatograms or several hundred thousand short reads. On computers with sufficient RAM (4 GB or more), CodonCode Aligner can handle projects with up to several thousand ABI reads (each about 500-1000 bases long), or several hundred thousand short "next gen" reads (300 bases or shorter).

The size of projects that CodonCode Aligner can handle is determined by the amount of memory available to Aligner. Details depend on the operation system used:

- On **Windows**, CodonCode Aligner will automatically try to use as much memory as needed, up to the amount of available physical memory. However, on 32-bit versions of Windows (or when running 32-bit versions of CodonCode Aligner on 64-bit Windows), the maximum amount of memory is limited to 1 GB.
- On **Mac OS X**, the maximum memory available to CodonCode Aligner is initially limited to 1024 MB. This is sufficient for projects with several thousand reads. You can **increase** the memory available to Aligner using the [memory preferences](#), and may need to do so if working with large projects.

The amount of memory that you can assign to CodonCode Aligner is limited only by the amount of built-in RAM. Please note that the memory available to Aligner should not exceed the amount of installed memory (RAM) on your computer; if you try to assign more memory, CodonCode Aligner may not start, or behave erratically.

If you are working on a computer with a limited amount of RAM, you may get better performance by quitting other open applications, and by reducing the amount of memory available to CodonCode Aligner.

You can always check how much memory CodonCode Aligner is currently using in the [memory preferences](#).

## How Aligner memory on Windows is set

On Windows, CodonCode Aligner will automatically try to use as much memory as needed. The memory available to CodonCode Aligner will be limited by:

- how much physical memory (RAM) is installed;
- how much memory is used by other open applications;
- on 32-bit versions of Windows, by limitations of the operating system.

On 32-bit versions of Windows, the maximum amount of memory that CodonCode Aligner can use is 1 GB(*the 1 GB limitation also applies when running 32-bit versions of CodonCode Aligner on 64-bit Windows*). On 64-bit Windows, CodonCode Aligner 4.2 and newer can use up to the total amount of available physical memory installed on the computer.

Other running applications will also require some memory, which may limit the memory available to CodonCode Aligner. Running other applications can also lead to "paging", where contents of the memory have to be written to and/or read from the hard disk; this can slow all applications down dramatically. If this happens, closing other open applications or installing more memory may help.



## How Aligner memory on Mac OS X is set

On OS X, the maximum memory is set in the file "CodonCode Aligner.cfg", which is inside the CodonCode Aligner package. When you change the memory settings for CodonCode Aligner in the [memory preferences](#), Aligner will edit this file, and then restart itself so that the new memory settings take effect.

**Please always use the [memory preferences](#) to change the memory available to Aligner on OS X.** Please do **not** change the memory available to Aligner by editing the "CodonCode Aligner.cfg" file directly - if you make any mistakes, Aligner may not be able to start, or may behave erratically.

# CodonCode Aligner Release Notes

The release notes for the current version of CodonCode Aligner can be found in a separate file, called "ReleaseNotes.txt", in the Aligner directory.

# Checking for Aligner Updates

You can check if your version of CodonCode Aligner is current by selection "**Check for Updates**" in the "**Help**" menu. Aligner will then check CodonCode's web site to see if your version is current; therefore, this requires that you have an active internet connection. If you are not currently connected to the internet, you will see an error message "Unable to verify the current Aligner version". The same message can also result from other problems, for example if CodonCode's web site is down (which hopefully will be rare :-).

When you start Aligner, Aligner also automatically tries to check for updates. Aligner will let you know if a newer version is available, or if there are any problems during the update check. If your version is current (which should be the case most of the time you start Aligner), you will not be notified. The automatic update checking can be turned off in the [update preferences](#).

Updates may be free or charge, or they may require the payment of additional license fees, depending on when you purchased your Aligner license, and the terms of your license.

## Visiting the Aligner Web Site

You can visit the Aligner web site by selecting "**Aligner Web Site**" from the "**Help**" menu. This will open a browser window and point it to the CodonCode Aligner web site - the starting point for downloading updates, requesting trial licenses, and the latest Aligner news. The address of the Aligner web site is <http://www.codoncode.com/aligner/>.