

# MacVector 17.5

for Mac OS X

## Using NGS Data to Map Transposon Insertion Locations

*MacVector, Inc.*  
Software for Scientists

## Copyright statement

Copyright **MacVector, Inc**, 2020. All rights reserved.

This document contains proprietary information of **MacVector, Inc** and its licensors. It is their exclusive property. It may not be reproduced or transmitted, in whole or in part, without written agreement from **MacVector, Inc**.

The software described in this document is furnished under a license agreement, a copy of which is packaged with the software. The software may not be used or copied except as provided in the license agreement.

**MacVector, Inc** reserves the right to make changes, without notice, both to this publication and to the product it describes. Information concerning products not manufactured or distributed by **MacVector, Inc** is provided without warranty or representation of any kind, and **MacVector, Inc** will not be liable for any damages.

This version of the Transposon Insertion Mapping tutorial was published in January 2020.

## Contents

<b>CONTENTS</b>	<b>3</b>
<b>INTRODUCTION</b>	<b>4</b>
<b>SAMPLE FILES</b>	<b>4</b>
<b>TUTORIAL</b>	<b>4</b>
Filter for Transposon-Containing Reads	4
Assembling a Transposon Consensus	8
Identify and Save Reads for Each Genomic Insertion	11
Assemble Filtered Sets of Reads	16
Identifying Transposition Sites on the Genome	19
What Genes Have the Transpositions Interrupted?	22
Shortcuts!	24

## Introduction

One common approach to investigating gene function is to randomly generate knockout mutations and screen for desired phenotypes. In bacteria, this is frequently done using transposons, typically carrying a selectable marker so that only those clones that take up the transposon can survive. With the advent of high throughput Next Generation Sequencing (NGS), it is often easiest and most cost-effective to simply sequence the entire genome of each isolate and then determine via software where in the genome the transposon insertion has occurred.

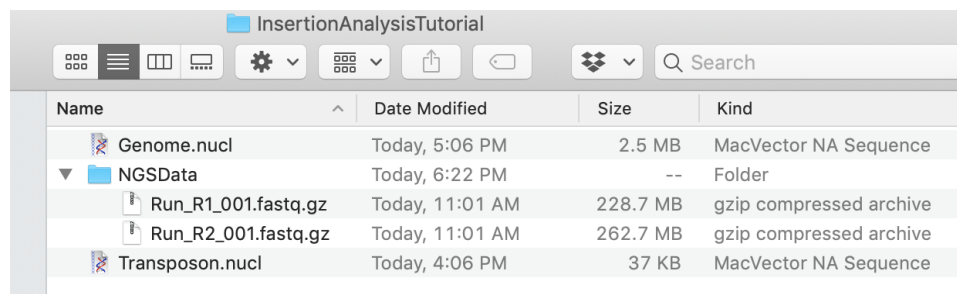
One trick to make it even cheaper to perform this analysis is to pool multiple clones into a single sequencing reaction and then use software to identify the different insertion locations. This tutorial demonstrates how you can use MacVector to identify the insertion locations of five individual transposition events pooled into a single NGS experiment.

## Sample Files

This tutorial uses four files;

- (a) A complete annotated bacterial genome (`Genome.nucl`)
- (b) The sequence of the transposon used in the mutagenesis (`Transposon.nucl`)
- (c) A pair of Illumina MiSeq gzipped NGS data files resulting from the sequencing of 5 individual transposon isolate clones (`Run_R1_001.fastq.gz` and `Run_R2_001.fastq.gz`)

For the purposes of this tutorial, we have organized these into a minimal folder hierarchy as follows;



Name	Date Modified	Size	Kind
Genome.nucl	Today, 5:06 PM	2.5 MB	MacVector NA Sequence
NGSData	Today, 6:22 PM	--	Folder
Run_R1_001.fastq.gz	Today, 11:01 AM	228.7 MB	gzip compressed archive
Run_R2_001.fastq.gz	Today, 11:01 AM	262.7 MB	gzip compressed archive
Transposon.nucl	Today, 4:06 PM	37 KB	MacVector NA Sequence

The files can be downloaded using this link;

<https://macvector.net/insertionanalysisistutorialdata.zip>

Once downloaded, extract the data and move the enclosing folder to a suitable location on your hard drive.

## Tutorial

### Filter for Transposon-Containing Reads

There is no need for us to assemble entire genomes for this analysis (though MacVector can do that if necessary). All we need are those reads that cross the junction between transposon and genome. If we can filter for those reads that match the transposon, then some of them will be at the ends and will cross into genomic sequence. We **could** search specifically for reads that match the ends of the transposon, but realistically there is no need to do that.

We can accomplish this using the **Database | Align to Folder** function in MacVector. This will scan folders on your file system containing sequences (including any reads in gzipped fasta and fastq files) searching for matches to an input sequence. You can then retrieve the matching sequences/reads of interest into a much smaller fasta/fastq file. One really nice thing about the function is that it is paired-end read aware, so if one read of a pair matches the query sequence, the other read of the pair will also be retrieved. For this experiment, if one of the pair lies within the transposon and the other lies outside, both will be retrieved, allowing us to pull out adjacent genome sequence even if neither read crossed the junction.

The first step is to open the `Transposon.nucl` sequence and invoke **Database | Align to Folder**.

There are a number of settings here that you need to change from the defaults;

**Search Folder:** click on **Choose** and select the folder containing your data.

**Folder contains paired-end reads:** this NGS data is paired-end, and selecting this helps tremendously with the analysis.

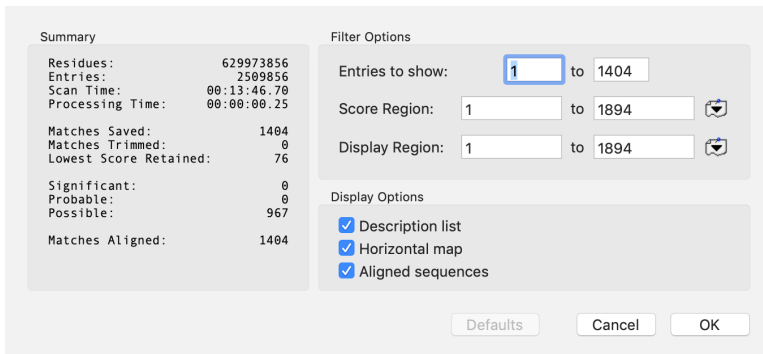
**Hash Value:** Its usually good to set this to the maximum unless you have less than 4 GB RAM or if your NGS data is particularly noisy (e.g. PacBio or Nanopore). Setting it to 12 means there must be at least one run of 12 perfect matches between the query and a read before MacVector will initiate an alignment.

**Scores to Keep:** You want to keep as many hits as you can, so set it to at least 10,000.

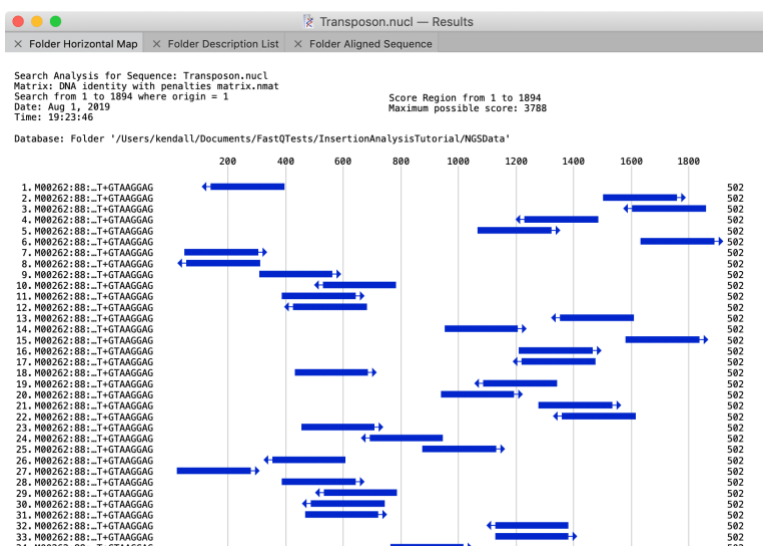
**Scoring Matrix:** For this type of analysis, you would expect very high identity between reads and the transposon. By using the `DNA Identity with Penalties.nmat` matrix, this will give priority to reads with short perfect matches at the ends of the query sequence (i.e. the junction reads we really want) versus those that have weak end-to-end similarity elsewhere in the query.

When you click **OK**, the search will run. The length of time this takes is dependent primarily on the size of your NGS data set and to some extent on the size of the query sequence and the speed of your machine. However, this is one case where the amount of RAM on your machine makes little difference as the reads are streamed into memory a few at a time and thus use very little RAM. In this experiment, the fastq files each contain 1.28 million Illumina MiSeq reads of

251 nt each and the search transposon is just under 2,000 bp. On a relatively high end 2018 i9 MacBook Pro with 32 GB RAM, the search completes in under 15 minutes;

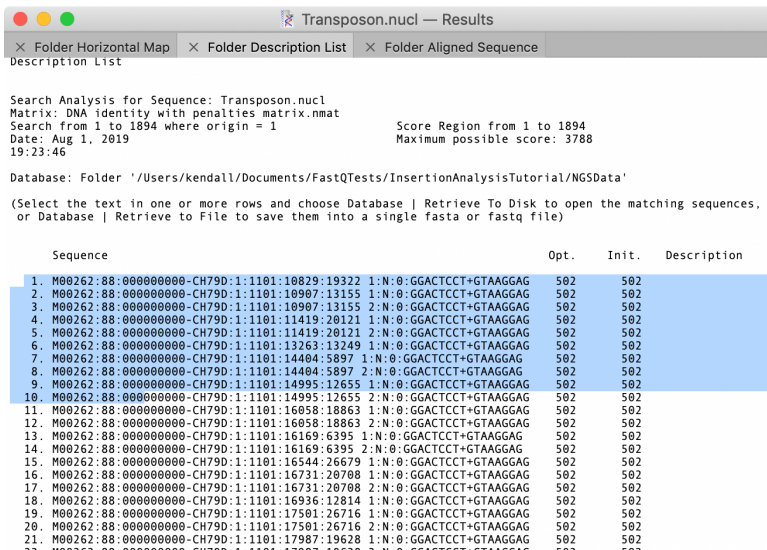


We want to see all of the **Display Options**, so check all the boxes.



We don't care about the **Folder Horizontal Map** tab too much – it gives an overview of the locations of the hits. As expected, the best hits land squarely in the middle of the transposon.

You can scroll through the **Folder Aligned Sequences** tab – it shows the text alignment of each read with the transposon. This can be a lot of text. As you explore you will see a number of reads that match the 5' end of the transposon and clearly extend to the left of the sequence. Those are the reads that we really want, but we don't have to specifically search them out at this point. What is most interesting is the **Folder Description List** tab;



While this looks like a plain text window, it is quite interactive. If you select one or more lines (even just a partial line as shown above) MacVector knows that you are interested in those sequences and will activate a number of menu items in the **Database** menu;

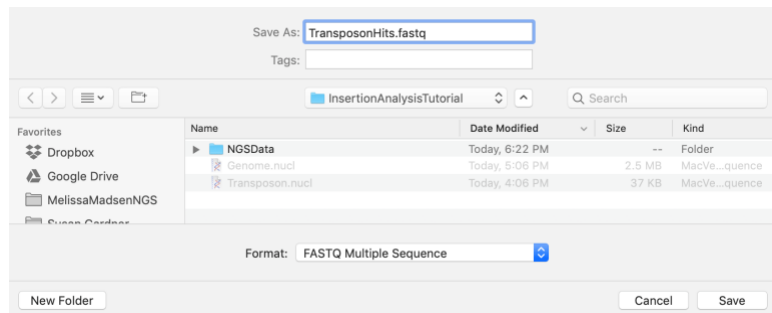
**Retrieve and Open** – this will retrieve the individual sequences or reads and open each in a separate window. Not recommended for more than 10 or so sequences!

**Retrieve to Disk** – retrieves and saves each sequence or read as a separate file in a folder you can designate.

**Retrieve to File** – retrieves and saves each sequence or read into a single fasta or fastq file.

The great thing about **Retrieve to File** is that it is paired-end read aware, so in cases like this it will actually save the reads into a pair of files that have “-1” and “-2” added to the filename.

You can optionally carefully scroll down the **Description List**, finding just those files that you want to retrieve, but for our purposes, the easiest thing is to choose **Edit | Select All**, followed by **Database | Retrieve to File** and give the file(s) a suitable format and name;



When you press **Save**, you’ll find that two files have been created – these are a matched pair of files where each entry in “-1” has its pair in “-2”. Many algorithms (*Bowtie*, *Velvet*, *SPAdes*) can take advantage of the paired reads in the files to generate better alignments.

Name	Date Modified	Size	Kind
Genome.nucl	Today, 5:06 PM	2.5 MB	MacVector NA Sequence
NGSDData	Today, 6:22 PM	--	Folder
Run_R1_001.fastq.gz	Today, 11:01 AM	228.7 MB	gzip compressed archive
Run_R2_001.fastq.gz	Today, 11:01 AM	262.7 MB	gzip compressed archive
Transposon.nucl	Today, 4:06 PM	37 KB	MacVector NA Sequence
TransposonHits-1.fastq	Today, 8:21 PM	464 KB	FASTQ Sequence
TransposonHits-2.fastq	Today, 8:21 PM	464 KB	FASTQ Sequence

## Assembling a Transposon Consensus

Now that we have a filtered collection of reads that match the transposon, we need a strategy to identify the reads that overlap the insertion locations in the genome. The easiest way to do this is to simply assemble all of the reads – the transposon should assemble very cleanly, but adjacent to the ends should be a mix of the five genome insertion locations. Those areas should have a lot of misaligned sequences which we can use to identify the genomic flanking sequences where the transposon inserted.

First, we create a **File | New | Assembly Project** (you'll need to have the *Assembler* module for this). Then we have two options for adding the `Transposon Hits`;

- Add Reads:** If you have a reasonable number of hits (this data set has over 700 reads in each file) you should click on the **Add Reads** toolbar button – the reads will be added as references to disk files;

Name	Status	Length	#	ClipL	ClipR	Start	Stop	Definition
TransposonHits-1.fastq	Illumina Paired-e...	251	722					/Users/kendall/Documents/FastQTe...
TransposonHits-2.fastq	Illumina Paired-e...	251	722					/Users/kendall/Documents/FastQTe...

NGS-based assembly algorithms such as *Velvet* and *SPAdes* work better with pairs of files, rather than individual sequences. However, they do require a reasonable coverage level (10x or more) so, if you have less than that, you may prefer to import the data as individual sequences;

- Add Seqs:** This will import the reads as individual sequences, as long as there are less than 5,000 sequences in each file, otherwise they will appear as disk-based references. When you have very few reads/sequences, it's typically better to use *phrap* as an assembly algorithm. While the MacVector implementation of *phrap* can use references to fasta/fastq files, it's often easier and cleaner to see the individual reads, particularly when you have less than 50 reads to assemble (more on that later) or if you have a mix of long contigs and short "patch" sequences to assemble.



Name	Status	Length	#	ClipL	ClipR	Start
M00262.88-000000000-CH79D:1:1101:2521:14656,1:N:0:GGACTCCT+GTAAGGAG		251	1	1	251	
M00262.88-000000000-CH79D:1:1101:2521:14656,2:N:0:GGACTCCT+GTAAGGAG		251	1	1	251	
M00262.88-000000000-CH79D:1:1101:5394:19255,1:N:0:GGACTCCT+GTAAGGAG		251	1	1	251	
M00262.88-000000000-CH79D:1:1101:5394:19255,2:N:0:GGACTCCT+GTAAGGAG		251	1	1	251	
M00262.88-000000000-CH79D:1:1101:7172:18753,1:N:0:GGACTCCT+GTAAGGAG		251	1	1	251	
M00262.88-000000000-CH79D:1:1101:7172:18753,2:N:0:GGACTCCT+GTAAGGAG		251	1	1	251	
M00262.88-000000000-CH79D:1:1101:7912:4325,1:N:0:GGACTCCT+GTAAGGAG		251	1	1	251	
M00262.88-000000000-CH79D:1:1101:7912:4325,2:N:0:GGACTCCT+GTAAGGAG		251	1	1	251	
M00262.88-000000000-CH79D:1:1101:8185:18642,1:N:0:GGACTCCT+GTAAGGAG		251	1	1	251	
M00262.88-000000000-CH79D:1:1101:8185:18642,2:N:0:GGACTCCT+GTAAGGAG		251	1	1	251	
M00262.88-000000000-CH79D:1:1101:8262:16215,1:N:0:GGACTCCT+GTAAGGAG		251	1	1	251	
M00262.88-000000000-CH79D:1:1101:8262:16215,2:N:0:GGACTCCT+GTAAGGAG		251	1	1	251	
M00262.88-000000000-CH79D:1:1101:8701:24218,1:N:0:GGACTCCT+GTAAGGAG		251	1	1	251	
M00262.88-000000000-CH79D:1:1101:8701:24218,2:N:0:GGACTCCT+GTAAGGAG		251	1	1	251	
M00262.88-000000000-CH79D:1:1101:10034:10028,1:N:0:GGACTCCT+GTAAGGAG		251	1	1	251	
M00262.88-000000000-CH79D:1:1101:10034:10028,2:N:0:GGACTCCT+GTAAGGAG		251	1	1	251	
M00262.88-000000000-CH79D:1:1101:10413:27275,1:N:0:GGACTCCT+GTAAGGAG		251	1	1	251	
M00262.88-000000000-CH79D:1:1101:10413:27275,2:N:0:GGACTCCT+GTAAGGAG		251	1	1	251	
M00262.88-000000000-CH79D:1:1101:10829:19322,1:N:0:GGACTCCT+GTAAGGAG		251	1	1	251	
M00262.88-000000000-CH79D:1:1101:10829:19322,2:N:0:GGACTCCT+GTAAGGAG		251	1	1	251	
M00262.88-000000000-CH79D:1:1101:10907:13155,1:N:0:GGACTCCT+GTAAGGAG		251	1	1	251	
M00262.88-000000000-CH79D:1:1101:10907:13155,2:N:0:GGACTCCT+GTAAGGAG		251	1	1	251	
M00262.88-000000000-CH79D:1:1101:11317:24678,1:N:0:GGACTCCT+GTAAGGAG		251	1	1	251	
M00262.88-000000000-CH79D:1:1101:11317:24678,2:N:0:GGACTCCT+GTAAGGAG		251	1	1	251	
M00262.88-000000000-CH79D:1:1101:11419:20121,1:N:0:GGACTCCT+GTAAGGAG		251	1	1	251	
M00262.88-000000000-CH79D:1:1101:11419:20121,2:N:0:GGACTCCT+GTAAGGAG		251	1	1	251	
M00262.88-000000000-CH79D:1:1101:11419:20121,3:N:0:GGACTCCT+GTAAGGAG		251	1	1	251	
M00262.88-000000000-CH79D:1:1101:13263:13249,1:N:0:GGACTCCT+GTAAGGAG		251	1	1	251	
M00262.88-000000000-CH79D:1:1101:13263:13249,2:N:0:GGACTCCT+GTAAGGAG		251	1	1	251	
M00262.88-000000000-CH79D:1:1101:13527:18762,1:N:0:GGACTCCT+GTAAGGAG		251	1	1	251	
M00262.88-000000000-CH79D:1:1101:13527:18762,2:N:0:GGACTCCT+GTAAGGAG		251	1	1	251	
M00262.88-000000000-CH79D:1:1101:13971:6683,1:N:0:GGACTCCT+GTAAGGAG		251	1	1	251	
M00262.88-000000000-CH79D:1:1101:13971:6683,2:N:0:GGACTCCT+GTAAGGAG		251	1	1	251	

For this data, with 2 x 700 reads, we can import as “Reads” and assemble using *SPAdes*. So, we select the two `fastq.gz` files and click on the **SPAdes** toolbar item.

Read pre-processing

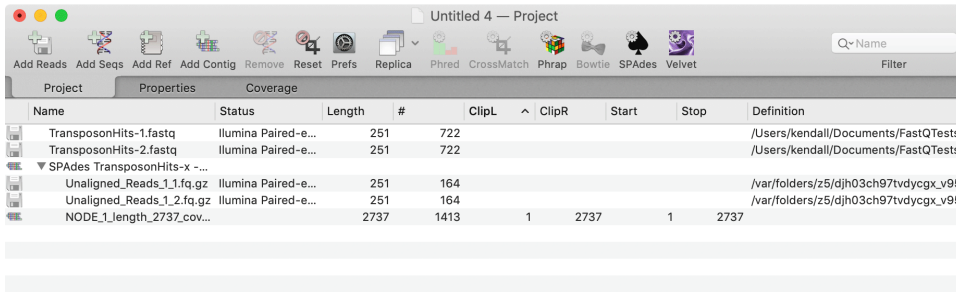
- Discard reads less than  nt
- Trim ends with quality less than
- Trim N's from ends
- Discard short reads that contain any N's

SPAdes Options

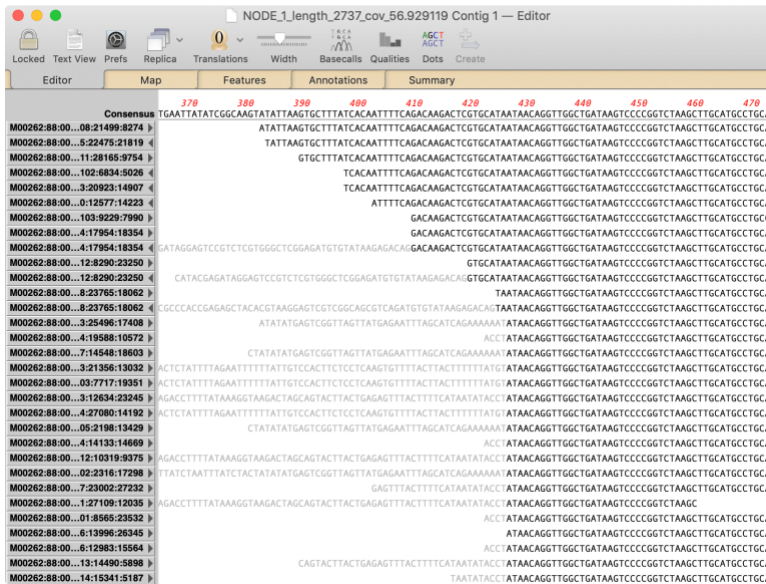
- Override coverage cutoff:
- Use custom K-MER:
- Threads:  Enter odd values less than 128 in ascending order, separated by commas (e.g. 21,33,55).
- Use "careful" mode with MismatchCorrector (slow)
- Generate Alignments Using Bowtie
- Threads:

? Defaults Cancel OK

The defaults are basically fine, but click on the **Generate Alignments Using Bowtie** checkbox so we get to see the actual reads aligned to the consensus. When you click **OK**, the *SPAdes* alignment will run. With a relatively small number of reads like this (2 x 700) that assemble into a short (~3 kb) sequence, the assembly only takes about 30 seconds. When complete, a new *SPAdes* job appears in the *Assembly Project*;

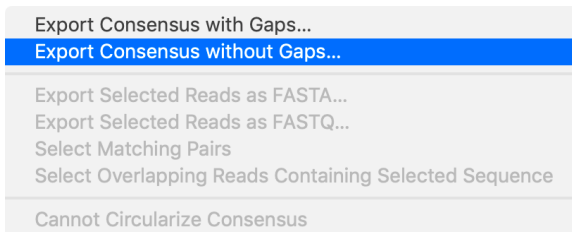


SPAdes contigs are always named using “NODE\_X\_...” nomenclature. In this case we have a single contig. If we double-click on the NODE\_1\_... item, a contig window appears;



If you scroll through the contig, you will eventually reach a region where many of the reads have greyed out areas to the left of a cutoff point, where to the right they are in bold text, as shown above. This is the left end of the contig - the greyed-out sequences should represent the flanking regions of the other insertion sites. The bold sequences at the top of the screenshot to the left of the inflection location also represent genomic flanking sequence – the SPAdes assembler will randomly choose one of the flanking sequences (typically the one with the most reads) and use that as the consensus.

While we could directly use this contig to determine the various flanking genomic regions of the transposon insertions, we will save the consensus sequence, then use MacVector’s **Align to Reference** functions for further analysis. If you right-click (<ctrl>-click on a trackpad) a popup menu appears;

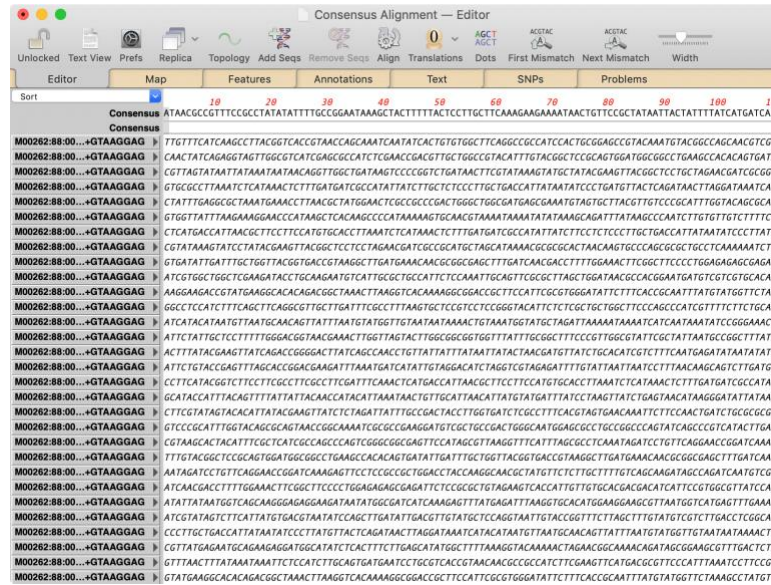


Choose **Export Consensus without Gaps** and choose a suitable filename to save the consensus sequence (e.g. Consensus.nucl).

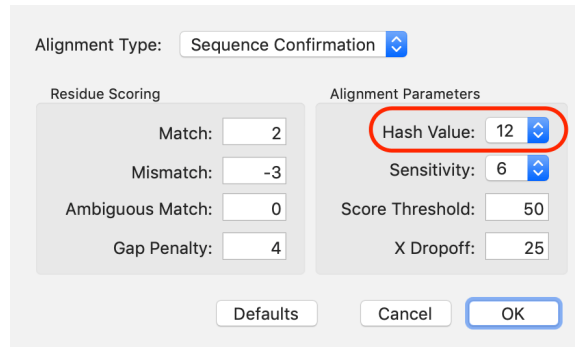
## Identify and Save Reads for Each Genomic Insertion

For the next step we will align all of the transposon-containing reads (and their pairs) to the saved consensus sequence, then examine the junction between the transposon and genomic sequence to pull out reads representing each of the five insertion sites.

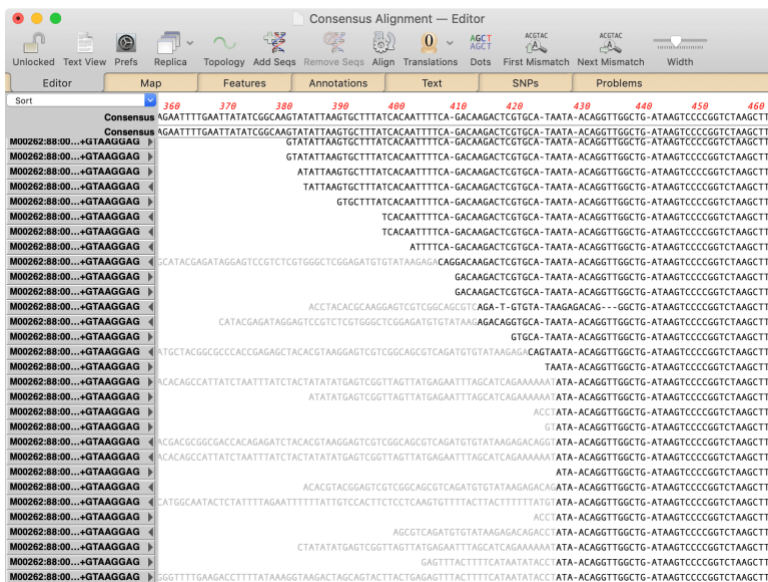
First, open the saved Consensus.nucl then choose **Analyze | Align to Reference**. Click on the **Add Seqs** toolbar button and select the TransposonHits-1.fastq and TransposonHits-2.fastq files. The **Editor** tab should look something like this;



Now click on the **Align** toolbar button and set up the alignment parameters something like this – the only parameter I’ve changed is the **Hash Value** to help speed up the alignment;



After you click **OK**, the alignment should take just a few seconds. If you scroll through the alignment, you should eventually reach an area where many of the reads abruptly do not match the reference and the left ends of the reads are shown greyed out (you may have scroll downwards as well as rightwards to see this);



Before proceeding further, you may want to save the alignment so that you can go back to it later. Choose **File | Save As...** and save the alignment with a suitable filename.

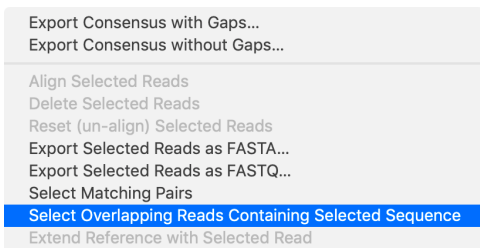
The reads towards the top of the screenshot are those that SPAdes used to create its preferred consensus and so align perfectly with the reference. These likely represent the insertion event that was most populous in the original pool of five, or that sequenced the most efficiently. In any event, we would like to select just those reads and save them into a separate file. Luckily, MacVector provides a simple way of doing this.

First, choose one of the reads that has perfect identity across the junction and select a few residues, ending one or two residues into the junction e.g.

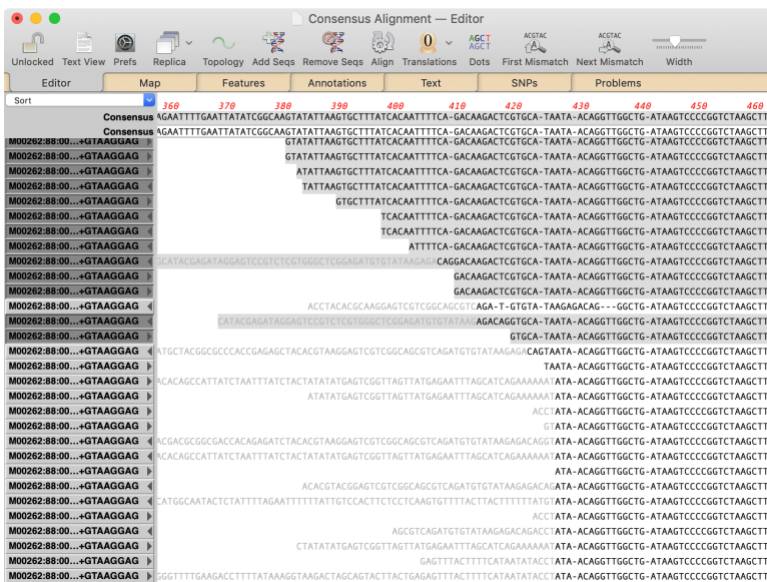
```

TTTATCACAAATTTTCA-GACAAGACTCGTGCA-TAATA-ACAGTTGGCTG-ATA
TTTATCACAAATTTTCA-GACAAGACTCGTGCA-TAATA-ACAGTTGGCTG-ATA
    TCACAATTTTCA-GACAAGACTCGTGCA-TAATA-ACAGTTGGCTG-ATA
    TCACAATTTTCA-GACAAGACTCGTGCA-TAATA-ACAGTTGGCTG-ATA
        ATTTTCA-GACAAGACTCGTGCA-TAATA-ACAGTTGGCTG-ATA
ATGTGTATAAGAGACAGGACAAGACTCGTGCA-TAATA-ACAGTTGGCTG-ATA
    GACAAGACTCGTGCA-TAATA-ACAGTTGGCTG-ATA
    GACAAGACTCGTGCA-TAATA-ACAGTTGGCTG-ATA
CAAGGAGTCGTCGGCAGCGTCAGA-T-GTGA-TAAGAGACAG--GGCTG-ATA
GGGCTCGGAGATGTGTATAAGACAGGTCGCA-TAATA-ACAGTTGGCTG-ATA
    GTGCA-TAATA-ACAGTTGGCTG-ATA
TCGTCGGCAGCGTCAGATGTGTATAAGAGACAGTAATA-ACAGTTGGCTG-ATA
    TAATA-ACAGTTGGCTG-ATA
TCGGTTAGTTATGAGAATTTAGCATCAGAAAAAATA-ACAGTTGGCTG-ATA
TCGGTTAGTTATGAGAATTTAGCATCAGAAAAAATA-ACAGTTGGCTG-ATA
    ACCTATA-ACAGTTGGCTG-ATA
    GTATA-ACAGTTGGCTG-ATA
    
```

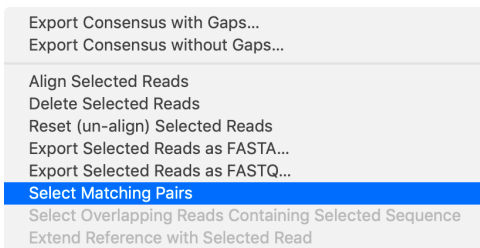
In this case I have selected a total of 7 residues (ignoring the gap) where the last two are the “AT” that appear to represent the left end of the transposon. Now we’d like to select all of the other reads that contain the same sequence aligned in the same position. Right-click (<ctrl>-click with a trackpad) and choose the **Select Overlapping Reads Containing Selected Sequence** in the context-sensitive menu;



Immediately, all of the reads containing that sequence at that position become selected;

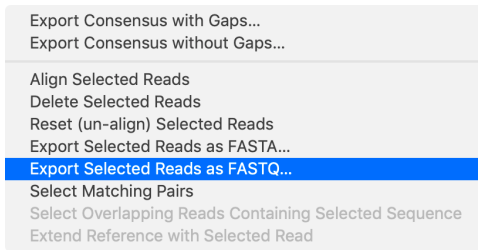


Now, remember, we started with paired-end reads. Depending on orientation, some of the reads will have a pair lying further within the transposon, while others will have a partner lying further into the genomic sequence. It would be nice to select those too. MacVector to the rescue! Being careful not to change the selections, again right-click to bring up the context-sensitive menu. This time choose **Select Matching Pairs**.

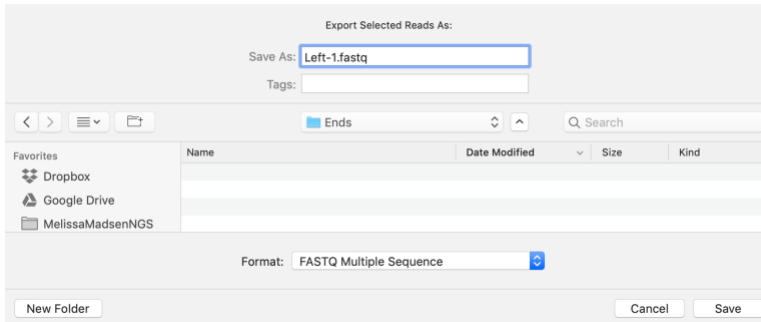


You may not see a visible change on screen if none of the matching pairs are currently visible. However, the appropriate pairs will be selected, even if they are not visible and even if they did not align to the reference.

Finally, we would like to save the selected reads to a fasta or fastq file. Generally, we want to save in the same format we started with, in this case fastq. There are two ways to do this. One way is to use the context sensitive menu again, this time choosing **Export Selected Reads as FASTQ**.

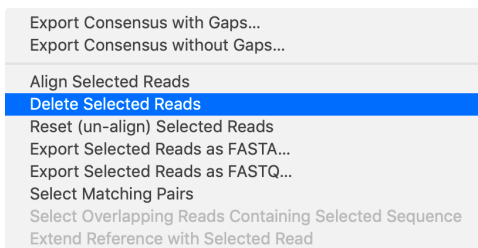


Choose a suitable name for the fastq file. Here, I've created a new `Ends` folder and have called the file `Left-1.fastq` as this is the first set of junction reads from the left end of the transposon;



You can also save the reads using the main **File | Export Selected Reads As...** menu item and choosing the file format in the export dialog.

Finally, now that we have saved the reads, we are really not interested in those particular anymore. To avoid confusion when we start looking at the other junction reads, let's just delete them. Assuming you still have them selected, this can also be done with the context-sensitive menu;



You can, of course, just press the `<delete>` key to accomplish the same thing.

Now we are ready to repeat this with the other sets of reads. If you look carefully at the greyed-out residues, you will see certain patterns repeated. For example, the following reads clearly fall into two groups

```

>ATTTAGCATCAGAAAAAATA-A(
TTACTTTTCATAATATACCTATA-A(
TTACTTTTCATAATATACCTATA-A(
>ATTTAGCATCAGAAAAAATA-A(
>ATTTAGCATCAGAAAAAATA-A(
TTACTTTTCATAATATACCTATA-A(
TTACTTTTCATAATATACCTATA-A(
TTACTTTTCATAATATACCTATA-A(
>ATTTAGCATCAGAAAAAATA-A(
>ATTTAGCATCAGAAAAAATA-A(
    
```

One set has the sequence `ATTTAGCATCAGAAAAAAT` before the **ATA-A** bold residues and the other has `TACTTTTCATAATATACC`. We can use the same principle as before to select the two distinct sets of reads. Note that MacVector doesn't care about the fact that some of the residues are greyed out, it will treat them just the same as the bold residues. So, lets select a few residues across the junction of the first of the sequence motifs. The exact length of selection is up to you. You want to select enough residues to ensure that only those reads that have a unique sequence are selected, but not so many that you miss some of the reads that just have short extensions across the junction. In this case I selected 8 residues, including the final **AT**.

```

\TTTAGCATCAGAAAAAATATA-AC
TACTTTTCATAATATACCTATA-AC
TACTTTTCATAATATACCTATA-AC
\TTTAGCATCAGAAAAAATATA-AC
\TTTAGCATCAGAAAAAATATA-AC
TACTTTTCATAATATACCTATA-AC
TACTTTTCATAATATACCTATA-AC
TACTTTTCATAATATACCTATA-AC
\TTTAGCATCAGAAAAAATATA-AC

```

Again, we can use the context-sensitive menu items **Select Overlapping Reads Containing Selected Sequence, Select Matching Pairs, Export Selected Reads as FASTQ** and **Delete Selected Reads** to save the pairs of reads containing the selection and to remove them from the alignment. In this case, I called the resulting file `Left-2.fastq`.

Next we repeat the procedure with the `TACTTTTCATAATATACC` sequence, calling those hits `Left-3.fastq`.

That should, in theory, leave us with just two remaining sets of insertion reads. In fact, there do appear to be two distinct sets of reads remaining;

```

CAAGTGTTTACTTACTTTTTTATGTATA-
CAAGTGTTTACTTACTTTTTTATGTATA-
CAAGTGTTTACTTACTTTTTTATGTATA-
      ATA-
      AAAGATA-
      ACCTATA-
CAAGCATAAACCTTGACAAAAAACAGTA-
CAAGCATAAACCTTGACAAAAAACAGTA-
CAAGCATAAACCTTGACAAAAAACAGTA-
CAAGCATAAACCTTGACAAAAAACAGTA-
CAAGCATAAACCTTGACAAAAAACAGTA-

```

Again, we can repeat the same procedure as above, saving the selected reads as `Left-4.fastq` and `Left-5.fastq`. When those reads have been removed from the alignment, our junction looks like this;

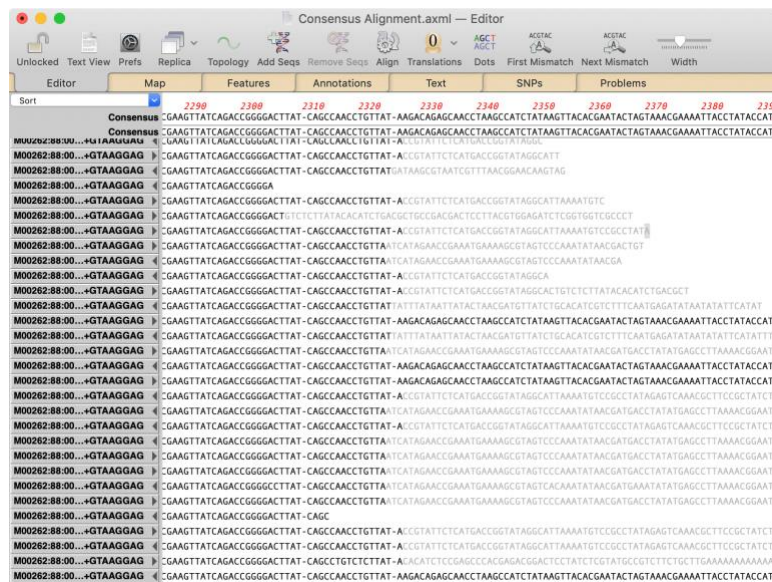
```

;GAGTCGTCGGCACGTCAGA-T-GTGTA-TAAGAGACAG---GGCTG-AT.
TCGGCAGCGTCAGTGTGTATAAGAGACAGTAATA-ACAGGTTGGCTG-AT.
      TAATA-ACAGGTTGGCTG-AT.
      ACCTATA-ACAGGTTGGCTG-AT.
      GTATA-ACAGGTTGGCTG-AT.
TCGGCAGCGTCAGTGTGTATAAGAGACAGGTAATA-ACAGGTTGGCTG-AT.
      ATA-ACAGGTTGGCTG-AT.
;GTGGCAGCGTCGATGTGTATAAGAGACAGATA-ACAGGTTGGCTG-AT.
      ACCTATA-ACAGGTTGGCTG-AT.
      AGCGTCAGATTGTATAAGAGACAGACCTATA-ACAGGTTGGCTG-AT.
      ACCTATA-ACAGGTTGGCTG-AT.
      ATA-ACAGGTTGGCTG-AT.
      AAAGATA-ACAGGTTGGCTG-AT.
      ACCTATA-ACAGGTTGGCTG-AT.
      GTA-ACAGGTTGGCTG-AT.
      GGCTG-AT.

```

Finally, we can see there is no significant commonality in the remaining reads. Some have short greyed out regions and thus were not selected by the **Select Overlapping Reads Containing Selected Sequence**, whereas the others are likely sequencing errors. At this stage we can be confident we have found the 5 sets of junction reads at this end.

Now we need to turn our attention to the right end. Again, after scrolling around, we can clearly see the greyed out reads at the junction. Note, that to help you view reads when scrolling around, if you click on a residue in the reference or consensus sequences, the display will scroll vertically to bring the nearest overlapping read into view.

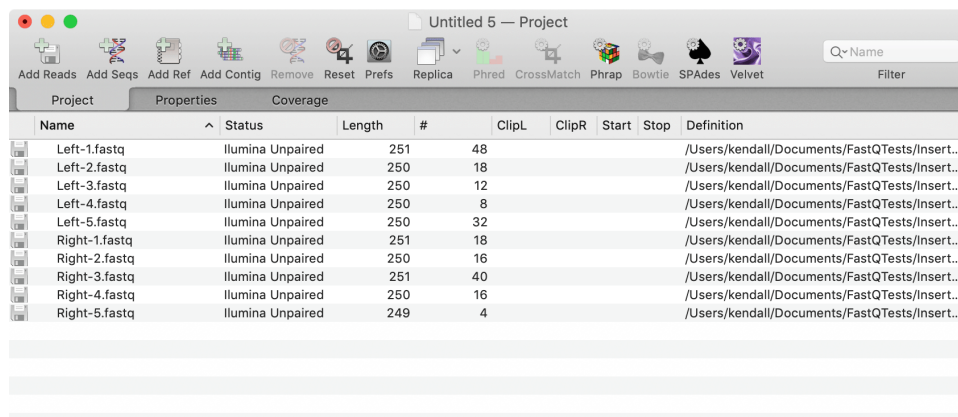


Now we just have to repeat the same analysis as we did for the left end, but calling our read collections `Right-1.fastq` through `Right-5.fastq`. When finished, close the alignment window, but don't save – we don't want to replace the original that had all the reads aligned before we started removing reads we had already saved to disk.

### Assemble Filtered Sets of Reads

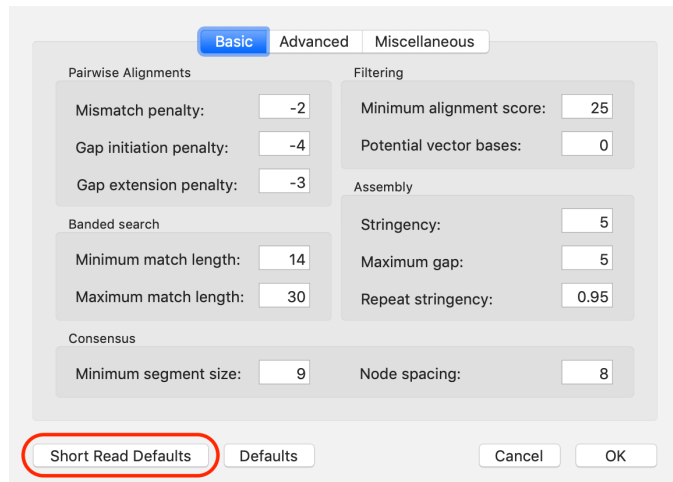
The most comprehensive way to proceed now is to separately assemble each of the 10 sets of reads we have created and then save the consensus from each of those to be aligned to the reference genome to reveal the exact locations of the transposon insertions.

Create a new **File | New | Assembly Project**, click on the **Add Reads** toolbar button and select the 10 sets of saved fastq files.

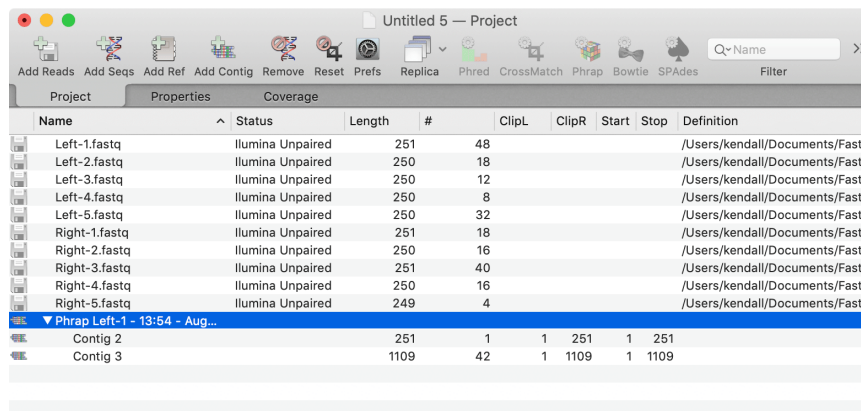




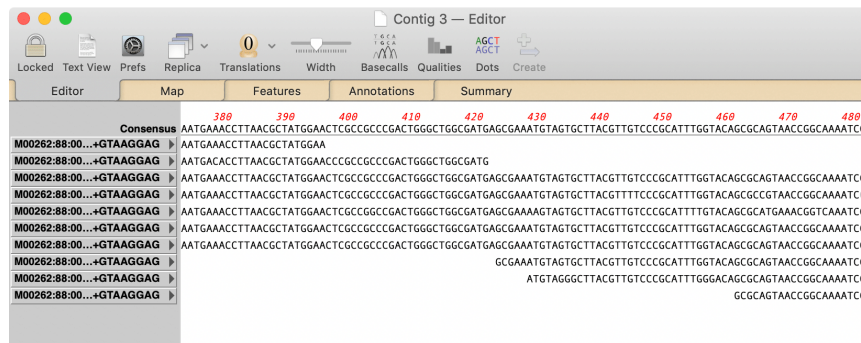
Note that some of the fastq files have very few sequences in them - just 8 for `Left-4.fastq` and only 4 for `Right-5.fastq`. These are far too few for typical NGS short-read assemblers such as *SPAdes* and *Velvet* to assemble. However, *phrap* can easily assemble small numbers of reads as long as they share some overlap and, in addition, will take into account the quality scores in the fastq files to generate better alignments. We want to assemble each set of reads independently, so first select `Left-1.fastq`, then click on the **Phrap** toolbar button.



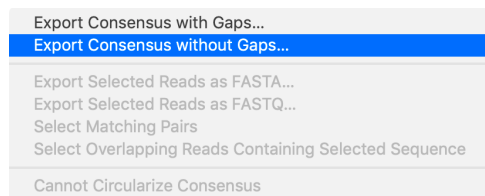
Click on the **Short Read Defaults** button to set up optimal parameters for this alignment, then click **OK**. After a short delay (2-3 seconds) a new *Phrap* job object appears in the project window.



You can see that two contigs were created, but one only contains a single read (the # column) and the other is much larger and contains most of the input reads. Double-click on `Contig 3` to open up the *Contig Editor*.



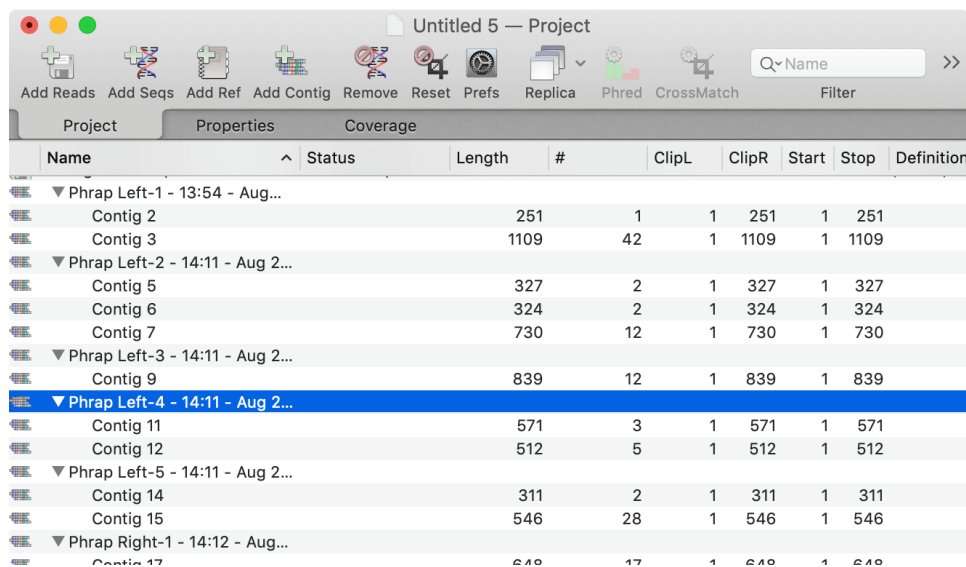
As you scroll through you can see that the reads have assembled very nicely. You can save the consensus by selecting the **File | Export Consensus As...** menu item or using the right-click context menu.



I saved the consensus as `Left-1.nucl`.

Now close the *Contig Editor* window and repeat for each of the `.fastq` input files. You may find it easier to run all the *phrap* assemblies first and to then go through each *phrap* job in turn, exporting the consensus.

Not all of the *phrap* assemblies generate clear “major” contigs. *Phrap Left-4* generates two contigs of similar size, one with 3 reads and one with 5 reads. We’ll save both of these contigs, with the names `Left-4a.nucl` and `Left4b.nucl`



Note that the procedure above represents the slow and methodical way of generating the consensus sequence for each filtered set of reads. In reality, you can save a lot of time by selecting all of the *phrap* result objects...

Name	Status	Length	#	ClipL	ClipR	Start	Stop	Defini
Right-4.fastq	Illumina Unpaired	250	16					/Users
Right-5.fastq	Illumina Unpaired	249	4					/Users
▼ Phrap Left-1 - 13:54 - Aug...								
Contig 2		251	1	1	251	1	251	
Contig 3		1109	42	1	1109	1	1109	
▼ Phrap Left-2 - 14:11 - Aug 2...								
Contig 5		327	2	1	327	1	327	
Contig 6		324	2	1	324	1	324	
Contig 7		730	12	1	730	1	730	
▼ Phrap Left-3 - 14:11 - Aug 2...								
Contig 9		839	12	1	839	1	839	
▼ Phrap Left-4 - 14:11 - Aug 2...								
Contig 11		571	3	1	571	1	571	
Contig 12		512	5	1	512	1	512	
▼ Phrap Left-5 - 14:11 - Aug 2...								
Contig 14		311	2	1	311	1	311	
Contig 15		546	28	1	546	1	546	
▼ Phrap Right-1 - 14:12 - Aug...								
Contig 17		648	17	1	648	1	648	
▼ Phrap Right-2 - 14:12 - Aug...								
Contig 19		250	1	1	250	1	250	
Contig 20		250	1	1	250	1	250	
Contig 21		612	13	1	612	1	612	
▼ Phrap Right-3 - 14:12 - Aug...								
Contig 23		847	35	1	847	1	847	
▼ Phrap Right-4 - 14:12 - Aug...								
Contig 25		882	16	1	882	1	882	
▼ Phrap Right-5 - 14:12 - Aug...								
Contig 27		372	4	1	372	1	372	

... and then choosing **File | Export Selected Contigs To...** which will prompt you for a folder to save the contigs into (I created a new empty folder), and will then save the consensus from each contig into that folder using the name of the contig e.g. Contig 2.nucl, Contig 3.nucl etc. You will see in the **Shortcuts** section, where we align the consensus sequences to the genome to identify the transposon locations, that either way will generate the same results.

### Identifying Transposon Sites on the Genome

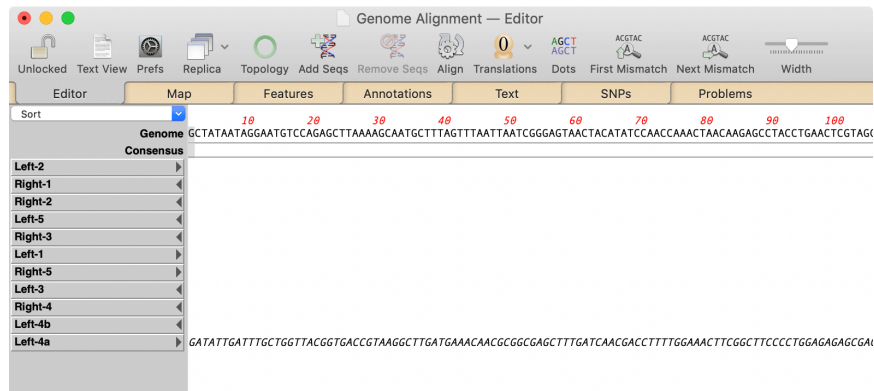
Now we can use the *Align to Reference* function to align the left and right consensus sequences to the target genome. In theory, only the section of each consensus representing the genome sequence should align, with any transposon sequence remaining unaligned and greyed out. In addition, if all has worked correctly, each alignment site should have a pair of consensus sequences aligning, one for each side of the transposon location.

First, we should open `Genome.nucl`, then choose **Analyze | Align to Reference**. Make sure you have the **Editor** tab selected, click on the **Add Seqs** toolbar button and select all the consensus sequences you created in the previous section.

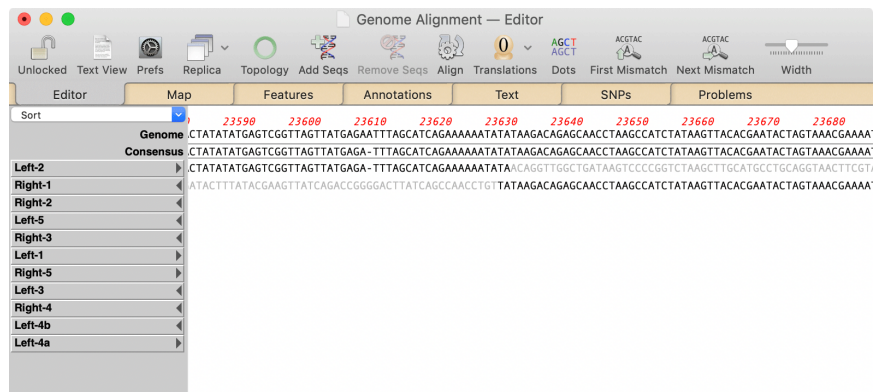
Sort	Genome
	GCTATAATAGGAATGTCAGAGCTTAAAGCAATGCTTTGATTTAATTAATCGGGAGTAACATATCAACCAAACTAACCAAGCCCTACCTGAACCTGCTAGC
<b>Consensus</b>	
Left-1	AAGCCAc aCAGTATTGATTTGCTGGTTACGgTGACCGTAAGGCTTGATGAACAACGGCGGAGCTTTGATcACGACCTTTTGGAACTTCGGCtLTC
Left-2	ATTTaATCCAAATGGGGAAATAGTCATCTATGCCGacTGCTAAATGGATAGCTTAACCAAAAAATTAGCTgTAACCCACCTATAAATGTAGGGTCACTaT
Left-3	aGATTCACAGCTCTTAAAGCATATAATATCTGTCTCTGTGTTGCTGAATAGAAACATCATCTGAAGCCATACCGTAAGCTTTTTCAAGCTCTTGATCTTAA
Left-4a	GATATTGATTTGCTGGTTACGGTACCGTAAGGCTTGATGAACAACGGCGGAGCTTTGATcAACGACCTTTTGGAAACTTCGGCTTCGGAGAGCGGA
Left-4b	gTTATAGACGGTACAAATGGGGTTAAGCTCTCGATTTTCCCACTAAAGCCGAGCATGATTTTACCATCGCCCTTAAAGTTAGAGAGACTGGCAAGGATG
Left-5	ATATTAGGTACACTAAACCAAGCATGATTATGTGGTATTTAAGAANGGAACCCATAAGCTCACAAGCCCAATAAAAGTGAACGTAATAAATAATAA
Right-1	GATTCACCTGGCAATAGCTTGCTTTTCCCTAACATTCTAGCTTCTCTACTCTCAATAGTTTATCTCTAAATCCAAATACTAAATTTCTTGCAAGTTCT
Right-2	ACTTAGGATaAaTcATAcTAAGTTAATGCAACAGTATTAAATGATGGTGTaATAAATAAACTGTAATGGTATGCTAGATAAAAATAAATCACTCAAT
Right-3	GCCTTTTGTGACCTTAAGTTGGCCGTGTGTGCCCTCATACGGCTCTCTTCGCTTCGCTTCGATTTCAAACTATGACCAATTAAGCTCTCTCCATGT
Right-4	gtGcGCcttAaaTCTcATAAAcTctttGATgATGcCaAtATTATCTCTCTCCCTGCTGACCAATTATAAATCCCTTATGTTACTCAGATAACTAGGATAA
Right-5	GCCGTATAATAACAAATCTAAGGTAAGCAAAAAATTTATCTTGAAGATGATGATAACGGCAGTAATGCTTGTTCGATATAATAAGCCTATTTCTAGTAAAG

Click on the **Align** toolbar button – the default parameters are fine for this, but if you find the alignment taking some time, you can try increasing the **Hash Value** to 12.

When the alignment completes, you should see that all of the sequences aligned except for Left-4a. Chances are this was created from internal transposon sequence that is not present in the genome.

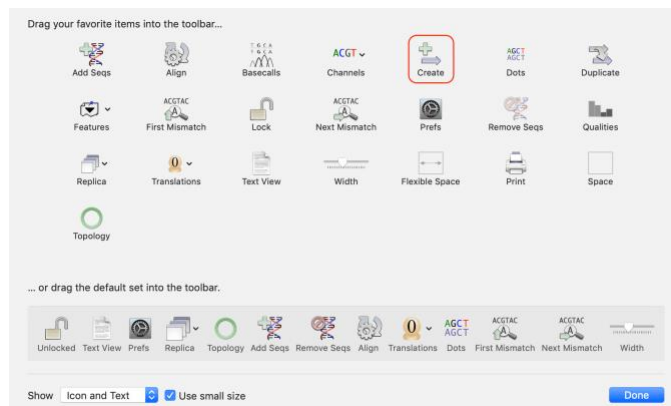


If you click on the Left-2 name button, the display will scroll to show where that sequence aligns. If you continue scrolling, you should eventually reach the location where Left-2 and Right-1 diverge.



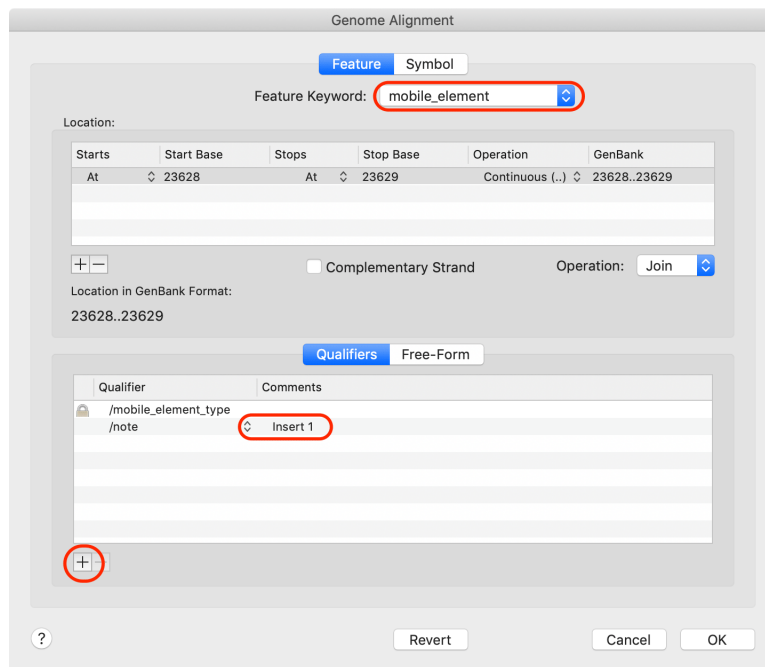
You can see that the two sequences represent the two genomic flanking regions of this insertion site. Note they both share a TA overlap – a feature of the transposon characteristic of this transposon.

Select the shared TA residues in the Genome reference sequence. Let's create a feature to highlight this insertion site. By default, the *Align to Reference* Editor tab does not have a **Create** toolbar button, but we can add one easily enough. Right-click (or **ctrl**-click) on any grey space on the toolbar and a context sensitive menu will appear. Choose **Customize Toolbar...** to bring up a sheet with options;



Drag the **Create** button (outlined above) onto the **Editor** tab toolbar and click **Done**.

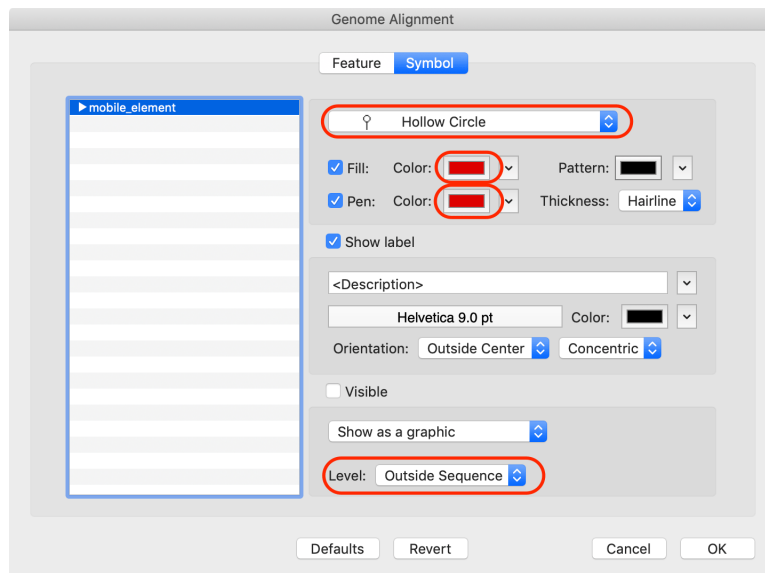
Now, with the **TA** residues selected, we can click on the **Create** toolbar button to bring up the feature creation/editor dialog.



Normally, I would recommend that you use `misc_feature` for annotating something like this, but the `Genome.nucl` sequence already has quite a lot of `misc_feature` annotations, so, to avoid our new feature getting lost in the midst of so many annotations, let's use `mobile_element` instead.

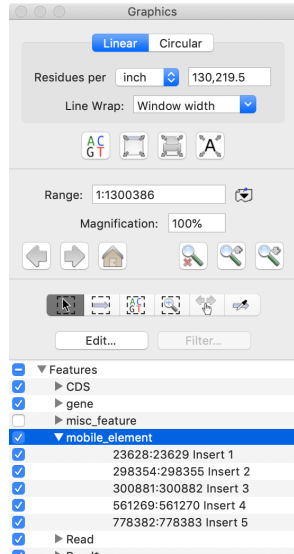
Also click on the “+” button (circled) to add a `/note` with a suitable name for our transposition location (e.g. “Insert 1”).

Click on the **Symbol** tab and let's add some eye-catching graphics for this;



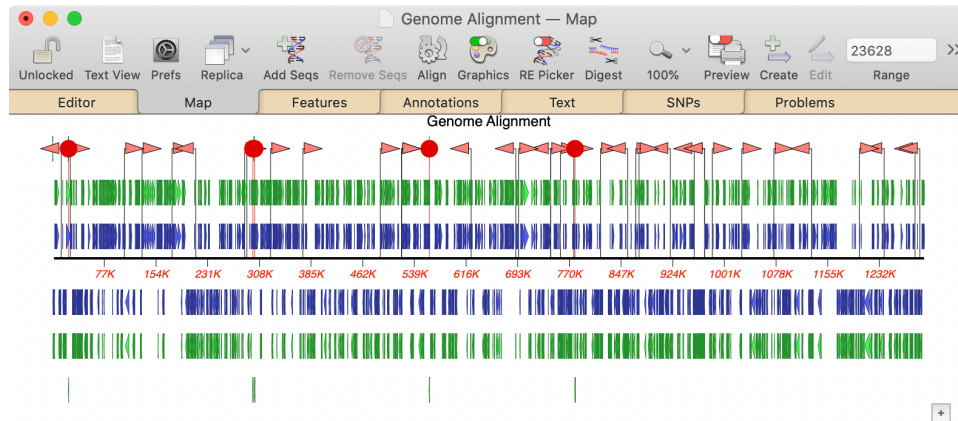
This will create a nice bright red “lollipop” at the insertion location.

Repeat with the other 4 insertion locations. As expected, at least for this data set, the 5 pairs nicely flank the 5 insertion sites. Note that to save some effort setting the symbols, you can create all `mobile_element` features without changing the default symbol, switch to the **Map** tab, then double-click on the `mobile_element` entry in the **Features** tree list so that you can change all of the feature symbols at one time.

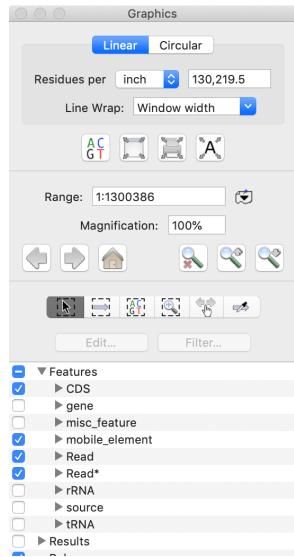


### What Genes Have the Transpositions Interrupted?

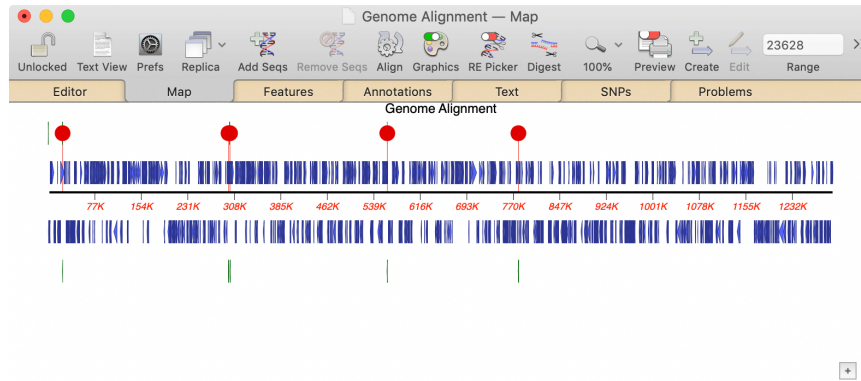
If we look at the **Map** tab, by default it's a bit of a mess with all of the different layers of features turned on;



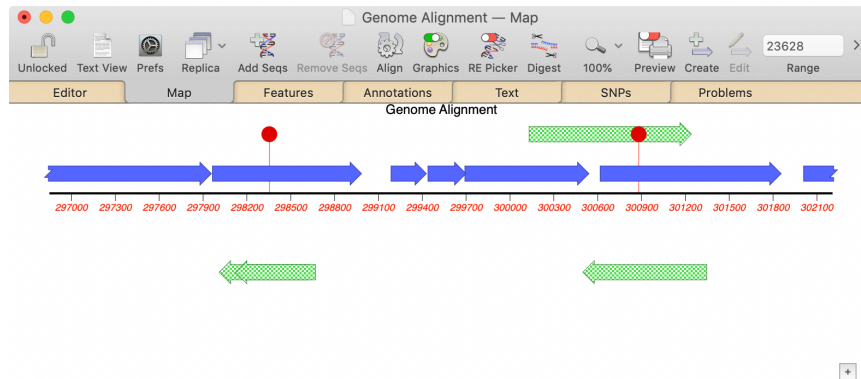
You can just about make out the red lollipops we added as flags for the insertion locations. This genome was downloaded directly from GenBank and has the standard default MacVector feature appearance – CDS features in blue, gene features in green and various RNA features as lines with arrowheads. In general, CDS features have the most extensive annotation when GenBank submissions are run through the Prokaryotic Genome Annotation Pipeline, so let's turn off the `gene` and RNA features, just leaving the `CDS`, `mobile_element` and `Read` features using the *Graphics Palette*;



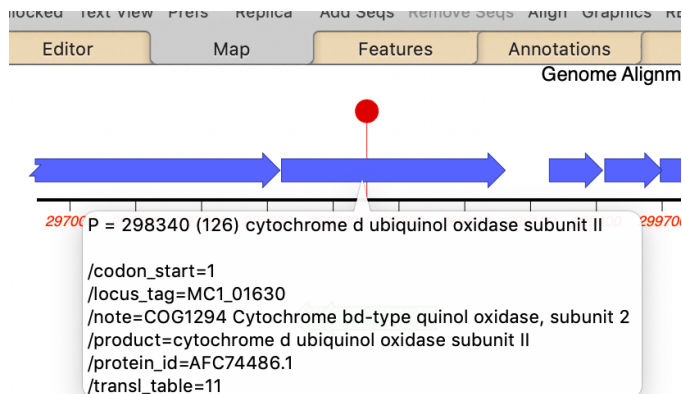
This cleans up the display tremendously and we can now clearly see our “lollipops” along with the green lines that are actually the locations of our aligned consensus sequences.



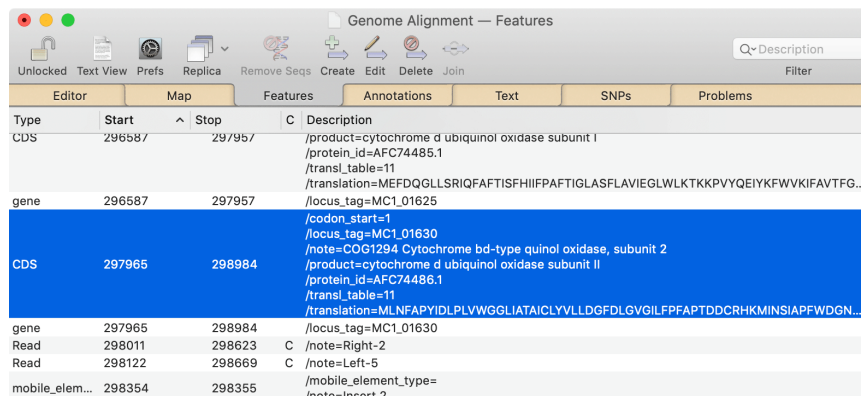
The second “lollipop” is actually two sites very close to each other – we can see that more clearly if we click to the left of the graphic, hold down the mouse and drag across a few pixels to “zoom” into the sequence. Note that at any time when zooming, you can double-click to get back to the “fit to window” scaling. You can also use the up and down arrow keys to zoom in/out 2x with each press and use the left/right arrow keys to nudge the zoomed region left or right.



You can clearly see that both of these transpositions have inserted into a CDS feature. If you mouse over the blue arrow representing the feature, a tooltip appears with additional information about the feature.



In addition, if you click on the blue graphic to select it, then switch to the **Features** tab, the feature is selected and scrolls into view;



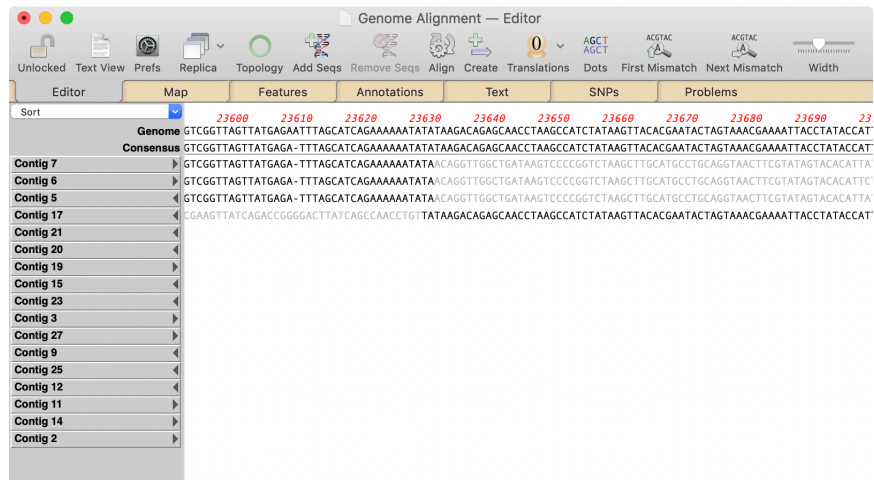
## Shortcuts!

The strategy outlined in the previous sections is very thorough and walks you carefully through each step. It should give you a good idea of the different functions in MacVector that can help you analyze this sort of data and is particularly applicable for small data sets where there may be very few reads that actually cross the transposon/genome boundaries. But, once you understand the concepts, you can often take shortcuts and still end up with the exact same results for less effort. Here are a couple of examples.

### Export All Contigs at Once

As described above, once you have used *phrap* to assemble contigs for each of the filtered read sets, rather than export each consensus separately, you can just save them all into a folder. Then, you can run an *Align to Reference* of all the contigs against the genome in the same way as we did with the individually named contigs;

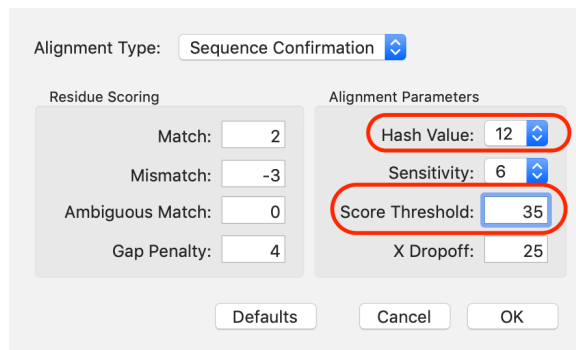




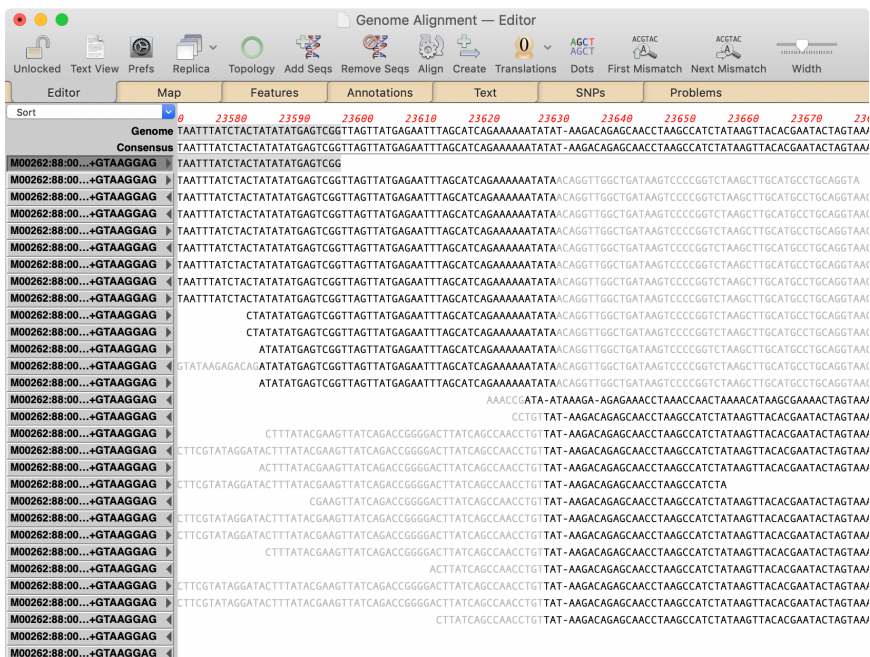
In this case, there are more contigs that do not align to the genome (likely pure transposon sequences) and some locations, such as above, where several contigs align either side of an insertion site. But the end result is the same – greyed out transposon sequences either side of an **AT** insertion point.

### Directly Align Raw Reads

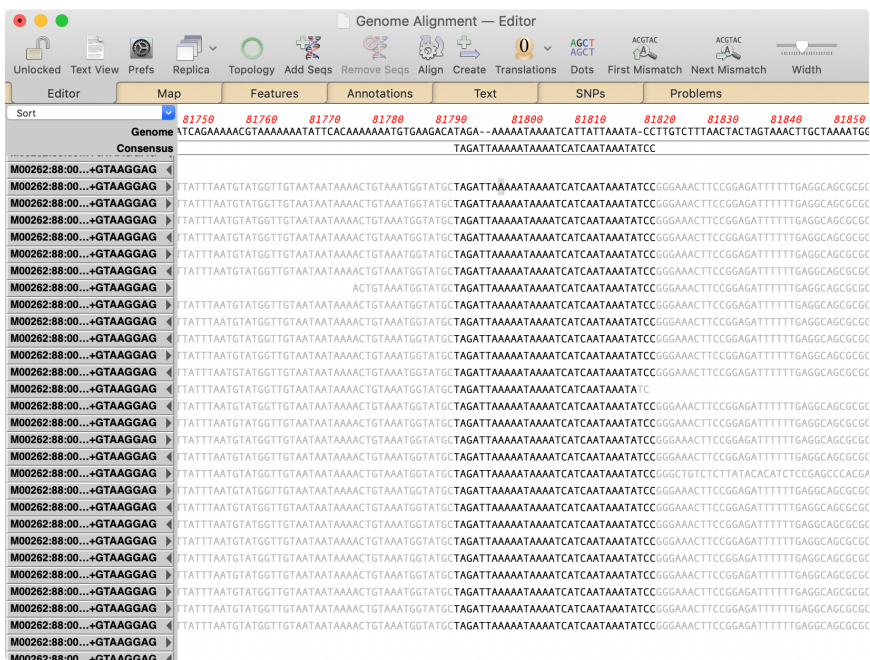
An even simpler approach is to completely bypass the read selection and contig assembly steps and just align all of the transposon containing reads from the *Align to Folder* search results directly against the genome. In some cases, this can work well – however it is reliant on having reads that cross the transposon junctions with enough adjacent target sequence present to align to the reference genome. Accordingly, you may want to adjust the alignment parameters a little. Here I've added the *Align to Folder* hits from the initial transposon search to a `Genome.nucl` *Align to Reference* and clicked on the **Align** toolbar button.



The main change from the defaults is to lower the **Score Threshold** to 35. As a **Match** is worth a score of 2, this means you need 18 perfect matches, for a total score of 36, in order for the read to align. You can try dropping this down to an even lower value to handle even shorter overlaps, but anything below 20 (i.e. 10 perfect matches) is likely to accept spurious matches. Note that I also increased the hash value to 12 for speed, though unless you have many thousands of reads to process, this is likely unnecessary.



After clicking on the first aligned read and scrolling along, you can clearly see that this simple approach works extremely well to identify the first insertion location, avoiding the need to work through all of the steps outlined above. But this may not always be the case. There can also be confusing areas like this one below, which is nowhere near any of the insertion sites but presumably reflects a region that has random short identity with the transposon sequence;



However, with this relatively clean set of data, all five insertion sites show up clearly and you just have to walk through a number of obviously incorrect spurious alignments to find the important alignments.