MAGenTA - Microbial Assessment by Genome-Wide Tn-Seq Analysis - Manual

Katherine Maia McCoy[1,2,4], Margaret L. Antonio[1,3,4], & Tim van Opijnen[1,5]

updated March 14, 2017

1 Boston College, Biology Department, 140 Commonwealth Ave, Higgins Hall 420, Chestnut Hill, MA 02467.

2 Current Address: MCB Graduate Program, Geisel School of Medicine, 7560 Remsen Building, Room 239, Hanover, NH 03755

3 Current Address: Biomedical Informatics Graduate Program, Stanford University School of Medicine, 1265 Welch Road, MSOB, X-215, MC 5479, Stanford CA 94305-5479

4 Equal 1st author contribution

5 Corresponding author: vanopijn@bc.edu

Contents

1 An Overview of Tn-Seq

This manual is not meant to describe Tn-Seq in full, only how to analyze data generated by Tn-Seq. If you are unclear on how Tn-Seq itself works, several scientific papers (1, 2, 3, 4) have been published with detailed explanations. Below in **Figure 1.** we illustrate an overview of the different types of approaches, the different types of sequencing products and reads these produce and how this translates into different processing requirements. This manual will help you to process data from either of these methods and with a little imagination probably any other variation you may use.

To begin, the tools and steps you must use on your data depend on how your sequencing samples were prepared. We review sample preparation in **Figure 1.**

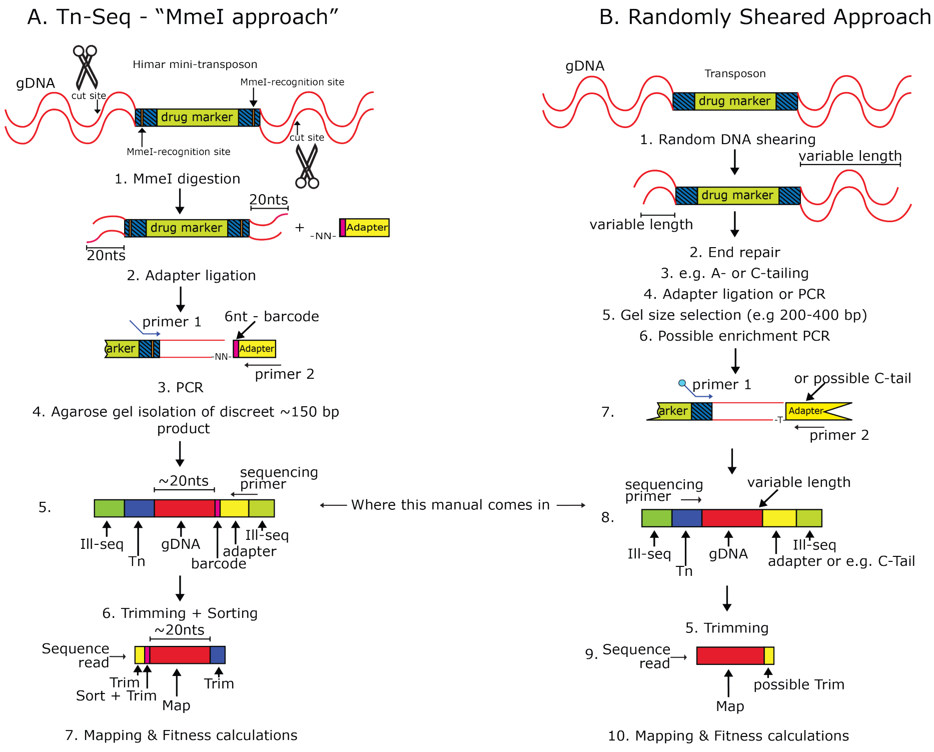


Figure 1. An overview of some of the main steps in the Tn-Seq sample preparation procedure that results in products that can be sequenced on an Illumina sequencer, the sequencing reads that come of the instrument and the way these can be processed. **A.)** Shows the different steps in the Tn-Seq procedure that uses "the MmeI approach" (see this article). Important here is that a PCR product is generated that has a length of approximately 150bp (step 3.). This product has two flanking Illumina sequences (which are essential to enable sequencing; in green), a small piece of the transposon sequence (in blue), approximately 20nts of bacterial genomic DNA (gDNA; in red), and the adapter sequence (in yellow), which also contains a 6nt barcode to enable multiplexing (in pink). Note that in this approach sequencing starts from the side of the adapter (sequencing primer), reads into the barcode, the gDNA and finally the transposon. For mapping purposes only the gDNA is needed and thus to enable this we need to trim off the transposon sequence, the adapter sequence, sort samples by their barcode and finally trim off the barcode. **B.)** In alternative approaches the product that gets sequenced has a variable length for instance because the gDNA gets randomly sheared. This means it is not exactly clear how large the gDNA fragment is that is flanked by the Illumina sequences (step 8.). Therefore sequencing often starts from the transposon side, reads into the gDNA and depending on the size of the gDNA may read into the adapter or C-tail (depending on the method used). For mapping purposes only the gDNA is needed and thus to enable this you often only need to trim off the adapter/C-tail sequence. However, if any more trimming or sorting is required it will be easy to implement with the tools described here.

After the reads described in Figure 1 are sequenced, they must be computationally analyzed, which is what we describe in detail in this manual. For clarity the analysis is roughly split into the following four parts (Figure 2):

Part A: Data Preparation

Tn-Seq data typically starts out as a huge file of reads, where each read contains genomic data and some non-genomic data. Potential non-genomic data includes parts of the transposon, barcodes, adaptors, etc. – anything that isn't a part of the original organism's genome. The data may also include reads from many different libraries and experimental times and conditions, with each unique combination specified by a barcode.

The Tn-Seq analysis tools calculate fitness by comparing mapped genomic data from a specific library at time 1 (T1) to mapped genomic data from that same library under an experimental condition at time 2 (T2). This means before fitness can be calculated, data must be separated by its barcodes, trimmed of all non-genomic portions, and mapped to the genome. Thus any Tn-Seq analysis has two main parts – prepping the data to be analyzed and analyzing it. The following data preps have premade workflows:

1. ”MmeI” Data Prep: for preparing data produced by the van Opijnen lab's standard Tn-Seq protocol (”MmeI approach”), as originally described here.

2. ”Randomly Sheared” Data Prep: for preparing data produced by a random shearing method, as described in papers like this one.

And if your data doesn't fit either of the above parameters, you can use a combination of our custom tools and the myriad of in-built Galaxy tools to create your own prep work flow. Details on this are further explained both in the Galaxy section and the Command Line / R section, depending on which approach you take.

Part B: Fitness Analysis

In this part fitness is calculated by making use of the change in number of reads from T1 to T2 and the growth achieved during the experiment (expansion factor). Each individual fitness for each insertion can finally be aggregated (e.g. by gene or genetic region) to obtain a single fitness value (e.g. per gene) and standard deviation, enabling further statistical analyses.

Part C: Exploring Fitness Values

An optional set of analyses and tools that let you compare your fitness data in various useful ways and prepare it for part D.

Part D: Data Visualization

An optional part which lets you visualize your data.

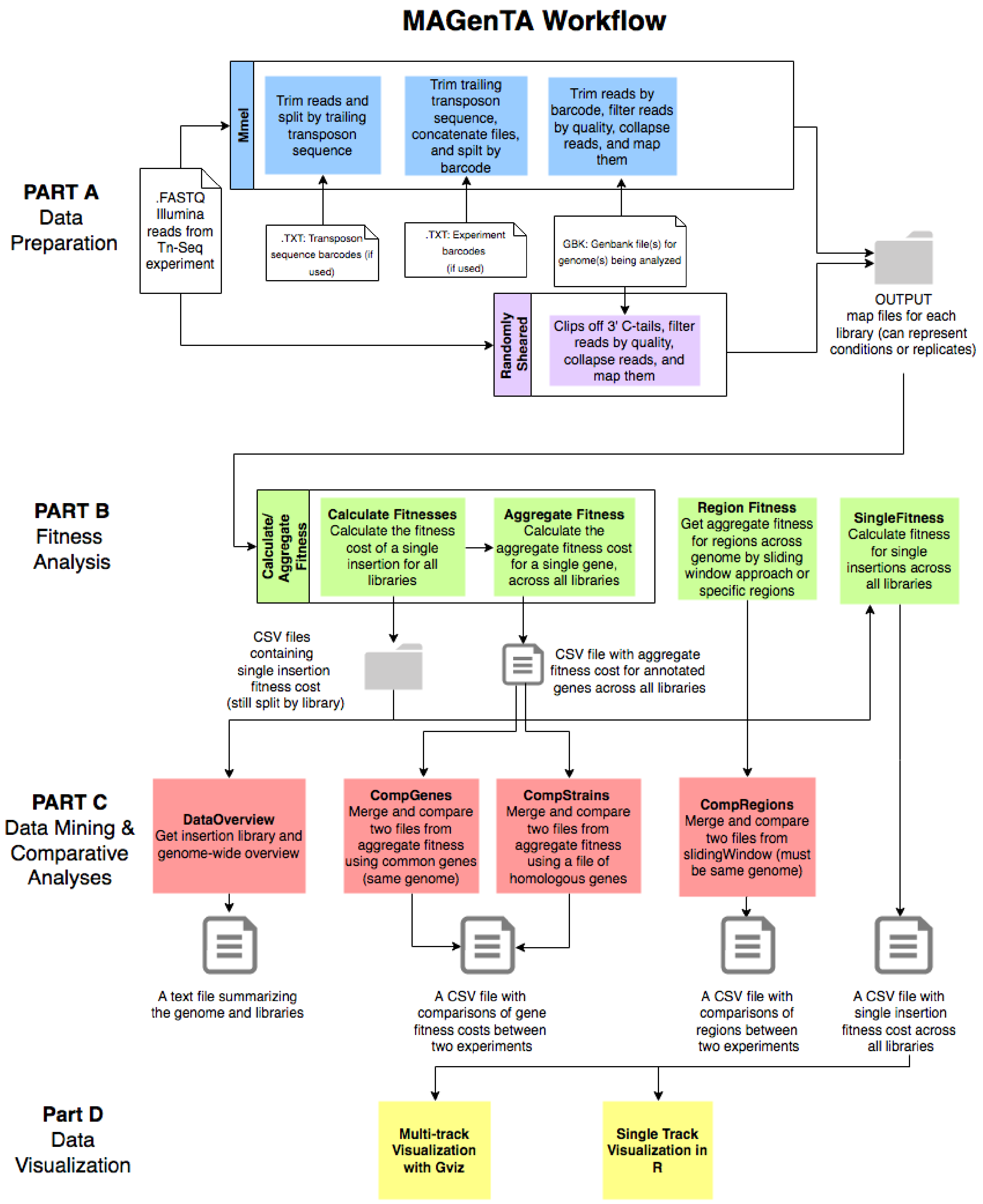
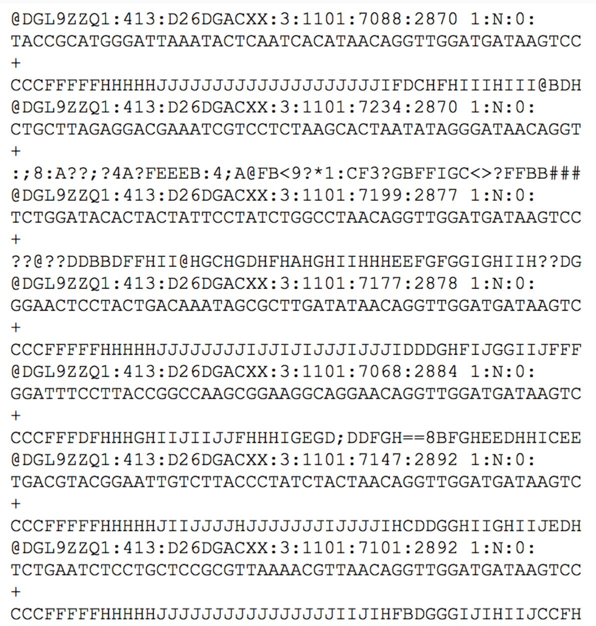

MAGenTA Workflow
PART A
Data Preparation
.FASTQ Illumina reads from Tn-Seq experiment
MmeI
Trim reads and split by trailing transposon sequence
Trim trailing transposon sequence, concatenate files, and spilt by barcode
Trim reads by barcode, filter reads by quality, collapse reads, and map them
.TXT: Transposon sequence barcodes (if used)
.TXT: Experiment barcodes (if used)
GBK: Genbank file(s) for genome(s) being analyzed
Randomly Sheared
Clips off 3' C-tails, filter reads by quality, collapse reads, and map them
OUTPUT
map files for each library (can represent conditions or replicates)
PART B
Fitness Analysis
Calculate/ Aggregate Fitness
Calculate Fitnesses
Calculate the fitness cost of a single insertion for all libraries
Aggregate Fitness
Calculate the aggregate fitness cost for a single gene, across all libraries
Region Fitness
Get aggregate fitness for regions across genome by sliding window approach or specific regions
SingleFitness
Calculate fitness for single insertions across all libraries
CSV files containing single insertion fitness cost (still split by library)
CSV file with aggregate fitness cost for annotated genes across all libraries
PART C
Data Mining & Comparative Analyses
DataOverview
Get insertion library and genome-wide overview
CompGenes
Merge and compare two files from aggregate fitness using common genes (same genome)
CompStrains
Merge and compare two files from aggregate fitness using a file of homologous genes
CompRegions
Merge and compare two files from slidingWindow (must be same genome)
A text file summarizing the genome and libraries
A CSV file with comparisons of gene fitness costs between two experiments
A CSV file with comparisons of regions between two experiments
A CSV file with single insertion fitness cost across all libraries
Part D
Data Visualization
Multi-track Visualization with Gviz
Single Track Visualization in R

Figure 2. An overview of the various steps of Tn-Seq Data Analysis. Data is trimmed of its non-genomic segments and mapped (A), has its genomic fitness calculated (B), can have fitness compared (C), and is visualized (D) by an array of useful tools.

2 The Files You'll Need

For the data analysis you will need the following (N.B. if you don't know what the different file formats that are mentioned below should look like, there are screenshots of different file-types sprinkled throughout the manual and examples can be found at our GitHub account):

- **Sequencing reads** in fastq format. These files should end in .fastq and look like this:

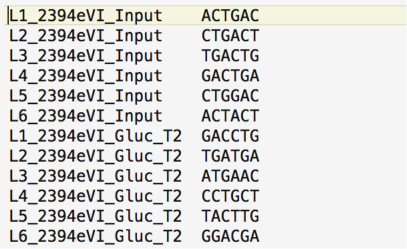


```
@DGL9ZZQ1:413:D26DGACXX:3:1101:7088:2870 1:N:0:
TACCGCATGGGATTAAATACTCAATCACATAACAGGTTGGATGATAAGTCC
+
CCCFFFFFHHHHHJJJJJJJJJJJJJJJJJJJJIFDCHFHIIIHIII@BDH
@DGL9ZZQ1:413:D26DGACXX:3:1101:7234:2870 1:N:0:
CTGCTTAGAGGACGAAATCGTCCTCTAAGCACTAATATAGGGATAACAGGT
+
:;8:A??;?4A?FEEEB:4;A@FB<9?*1:CF3?GBFFIGC<>?FFBB###
@DGL9ZZQ1:413:D26DGACXX:3:1101:7199:2877 1:N:0:
TCTGGATACACTACTATTCCTATCTGGCCTAACAGGTTGGATGATAAGTCC
+
??@??DDBBDFFHII@HGCHGDHFHAHGHIIHHHEEFGFGGIGHIIH??DG
@DGL9ZZQ1:413:D26DGACXX:3:1101:7177:2878 1:N:0:
GGAACTCCTACTGACAAATAGCGCTTGATATAACAGGTTGGATGATAAGTC
+
CCCFFFFFHHHHHJJJJJJJIJJIJIJJJIJJJIDDDGHFIJGGIIJFFF
@DGL9ZZQ1:413:D26DGACXX:3:1101:7068:2884 1:N:0:
GGATTTCCTTACCGGCCAAGCGGAAGGCAGGAACAGGTTGGATGATAAGTC
+
CCCFFFDFHHHGHIIJIIJJFHHHIGEGD;DDFGH==8BFGHEEDHHICEE
@DGL9ZZQ1:413:D26DGACXX:3:1101:7147:2892 1:N:0:
TGACGTACGGAATTGTCTTACCCTATCTACTAACAGGTTGGATGATAAGTC
+
CCCFFFFFHHHHHJIIJJJJHJJJJJJJIJJJJIHCDDGGHIIGHIIJEDH
@DGL9ZZQ1:413:D26DGACXX:3:1101:7101:2892 1:N:0:
TCTGAATCTCCTGCTCCGCGTTAAAACGTTAACAGGTTGGATGATAAGTCC
+
CCCFFFFFHHHHHJJJJJJJJJJJJJJJJJIIJIHFBDGGGIJIHIIJCCFH
```

- **Expansion factors**, which are determined experimentally. For *in vitro* experiments you determine the number of bacteria you start the experiment with for instance by plating appropriate dilutions of your starting culture (T1) and counting the number of Colony Forming Units (CFUs). Next, you repeat this for the end of your experiment (T2) and divide the number of CFUs at T2 over those at T1.

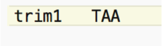
  For instance if you end with 100 million CFUs/ml at T2 and started with 1 million CFUs/ml at T1, your expansion factor would be 100. For *in vivo* experiments it may be a bit more complicated, but a reasonable estimation often works as well, but see here how you can experimentally determine *in vivo* growth rates and expansion factors. You should know that the expansion factor is used to normalize the data for each library. So if one library has grown longer then another your expansion factor will be larger for the latter one. Without normalization you would have a very difficult time aggregate libraries or compare experiments, however by using expansion factors and our "fitness formula" this does become possible. Importantly, by using this approach you calculate the growth rate for each insertion, thereby making your data independent of the time it was sampled. See for a more detailed explanation the following papers (1, 2, 3, 4).

- Barcode files, if you used any barcodes to differentiate between e.g. different conditions. You can make this file in any plain text editor; give each barcode a line which consists of whatever you'd like to name it, a tab, and its sequence. We strongly recommend making barcode names descriptive so that they include the sample's library number, and experimental condition, for easy identification. These files should end in .txt and look like this:

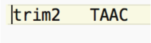


```
L1_2394eVI_Input     ACTGAC
L2_2394eVI_Input     CTGACT
L3_2394eVI_Input     TGACTG
L4_2394eVI_Input     GACTGA
L5_2394eVI_Input     CTGGAC
L6_2394eVI_Input     ACTACT
L1_2394eVI_Gluc_T2   GACCTG
L2_2394eVI_Gluc_T2   TGATGA
L3_2394eVI_Gluc_T2   ATGAAC
L4_2394eVI_Gluc_T2   CCTGCT
L5_2394eVI_Gluc_T2   TACTTG
L6_2394eVI_Gluc_T2   GGACGA
```

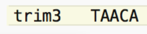
- **Four variable-MmeI-ending files**, however, this is only necessary if you're working with reads created by cutting your transposon with MmeI, which is the wet lab method used by the van Opijnen lab (for instance see 1) and Figure 1. These are formatted the same way as the barcode files, but instead of containing barcodes they describe the exact sequence of the end of your transposon. These files are required for using the barcode_splitter tool ”off label” to split reads by their length of remaining transposon DNA. These files can be found on our github but you can also easily make them yourself; they should end in .txt, have one line each, and are exactly as follows:



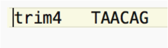
```
trim1   TAA
```



```
trim2   TAAC
```

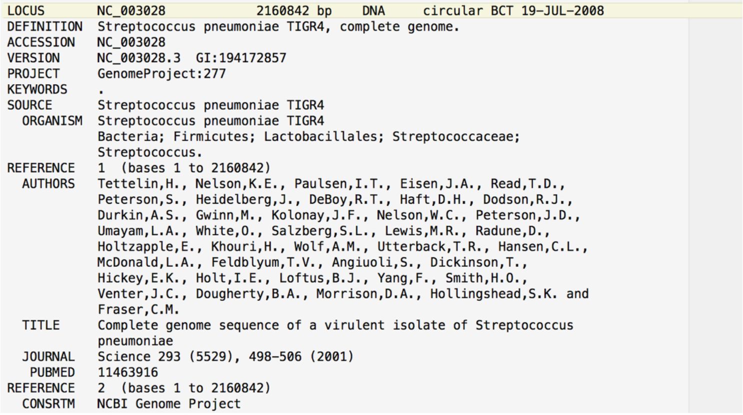


```
trim3   TAACA
```

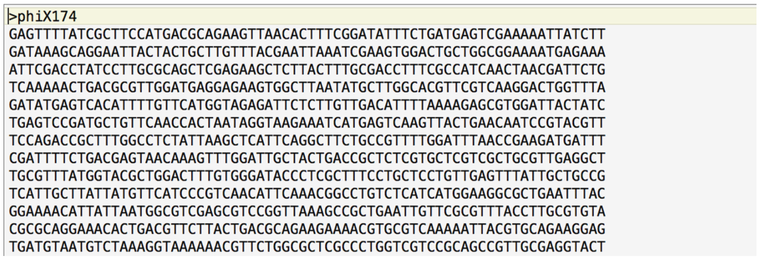


```
trim4   TAACAG
```

- **Genbank files / fasta genome sequences** for each strain you're analyzing, which can be found on NCBI by searching for each strain under the Genome section. These files should respectively end in .gbk and .fasta, and look like these examples:

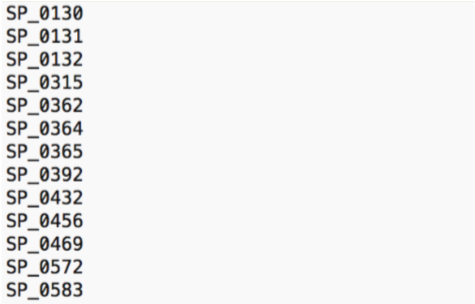


```
LOCUS       NC_003028            2160842 bp    DNA     circular BCT 19-JUL-2008
DEFINITION  Streptococcus pneumoniae TIGR4, complete genome.
ACCESSION   NC_003028
VERSION     NC_003028.3  GI:194172857
PROJECT     GenomeProject:277
KEYWORDS    .
SOURCE      Streptococcus pneumoniae TIGR4
  ORGANISM  Streptococcus pneumoniae TIGR4
            Bacteria; Firmicutes; Lactobacillales; Streptococcaceae;
            Streptococcus.
REFERENCE   1  (bases 1 to 2160842)
  AUTHORS   Tettelin,H., Nelson,K.E., Paulsen,I.T., Eisen,J.A., Read,T.D.,
            Peterson,S., Heidelberg,J., DeBoy,R.T., Haft,D.H., Dodson,R.J.,
            Durkin,A.S., Gwinn,M., Kolonay,J.F., Nelson,W.C., Peterson,J.D.,
            Umayam,L.A., White,O., Salzberg,S.L., Lewis,M.R., Radune,D.,
            Holtzapple,E., Khouri,H., Wolf,A.M., Utterback,T.R., Hansen,C.L.,
            McDonald,L.A., Feldblyum,T.V., Angiuoli,S., Dickinson,T.,
            Hickey,E.K., Holt,I.E., Loftus,B.J., Yang,F., Smith,H.O.,
            Venter,J.C., Dougherty,B.A., Morrison,D.A., Hollingshead,S.K. and
            Fraser,C.M.
  TITLE     Complete genome sequence of a virulent isolate of Streptococcus
            pneumoniae
  JOURNAL   Science 293 (5529), 498-506 (2001)
   PUBMED   11463916
REFERENCE   2  (bases 1 to 2160842)
  CONSRTM   NCBI Genome Project
```



```
>phiX174
GAGTTTTATCGCTTCCATGACGCAGAAGTTAACACTTTCGGATATTTCTGATGAGTCGAAAAATTATCTT
GATAAAGCAGGAATTACTACTGCTTGTTTACGAATTAAATCGAAGTGGACTGCTGGCGGAAAATGAGAAA
ATTCGACCTATCCTTGCGCAGCTCGAGAAGCTCTTACTTTGCGACCTTTCGCCATCAACTAACGATTCTG
TCAAAAACTGACGCGTTGGATGAGGAGAAGTGGCTTAATATGCTTGGCACGTTCGTCAAGGACTGGTTTA
GATATGAGTCACATTTTGTTCATGGTAGAGATTCTCTTGTTGACATTTTAAAAGAGCGTGGATTACTATC
TGAGTCCGATGCTGTTCAACCACTAATAGGTAAGAAATCATGAGTCAAGTTACTGAACAATCCGTACGTT
TCCAGACCGCTTTGGCCTCTATTAAGCTCATTCAGGCTTCTGCCGTTTTGGATTTAACCGAAGATGATTT
CGATTTTCTGACGAGTAACAAAGTTTGGATTGCTACTGACCGCTCTCGTGCTCGTCGCTGCGTTGAGGCT
TGCGTTTATGGTACGCTGGACTTTGTGGGATACCCTCGCTTTCCTGCTCCTGTTGAGTTTATTGCTGCCG
TCATTGCTTATTATGTTCATCCCGTCAACATTCAAACGGCCTGTCTCATCATGGAAGGCGCTGAATTTAC
GGAAAACATTATTAATGGCGTCGAGCGTCCGGTTAAAGCCGCTGAATTGTTCGCGTTTACCTTGCGTGTA
CGCGCAGGAAACACTGACGTTCTTACTGACGCAGAAGAAAACGTGCGTCAAAAATTACGTGCAGAAGGAG
TGATGTAATGTCTAAAGGTAAAAAACGTTCTGGCGCTCGCCCTGGTCGTCCGCAGCCGTTGCGAGGTACT
```

- **Normalization files** for each strain, which are .txt files containing all ”normalization genes”, one per line. Normalization genes are those that should have no effect on fitness when disrupted, and you can generate normalization files yourself by looking through a strain's genbank file and recording all transposon genes, pseudogenes, and degenerate genes - or generally include any gene that you know does not have an effect on fitness. However, this file is not essential and if it is not available or you don't want to compile it, it is often OK to run the analysis without it. These files should end in .txt.



```
SP_0130
SP_0131
SP_0132
SP_0315
SP_0362
SP_0364
SP_0365
SP_0392
SP_0432
SP_0456
SP_0469
SP_0572
SP_0583
```

Finally, if you'd rather run our tools manually on your own computer rather than automatically through Galaxy, you can skip down to command line / R instructions here. Otherwise, continue on for the Galaxy instructions.

3 Analysis via Galaxy

If you have never used Galaxy or our tools before, an easy tutorial on how to set Galaxy up can be found here, how to upload files is described here, and our custom tools for data pre-processing and the traditional Tn-Seq fitness analysis (caluclate fitness and aggregate fitness) can all be found and installed through the Galaxy toolshed. You can find them by searching for their names in the toolshed. Tool IDs and the repositories in which they are located are also listed.

- Enhanced Bowtie Mapper (id: bowtie_mapper, repository: enhanced_bowtie_mapper)
- FASTQ Collapser (id: fastq collapser, repository: fastq_collapser)
- Calculate Fitness (id: calc_fitness, repository: calculate_fitness)
- Aggregate (id: aggregate, repository: aggregate_fitness)
- Region Fitness (id: regionFitness, repository: regionfitness)
- Single Fitness (id: singleFitness, repository: singlefitness)
- Data Overview (id: dataOverview, repository: dataoverview)
- Compare Regions (id: compRegions, repository: compregions)
- Compare Genes (id: compGenes, repository: compgenes)

As described in Figure 2, there are four main parts to the analysis. The first, the Data Prep, varies depending on your type of data. If you prepared your samples with the MmeI Tn-Seq method, you'll use the MmeI Tn-Seq Prep. If you prepared your samples with a random shearing method, you'll use the Randomly Sheared Tn-Seq Prep. If you prepared your samples some other way, you can follow this generalized data prep protocol for making your own workflow via Galaxy's workflow editor:

- Split data by experimental condition barcodes, if any, using the fastx barcode splitter.
- Remove all non-genomic DNA using a tool like the fastq trimmer.
- Filter reads by quality using the fastq quality splitter, if you like.
- Map reads with Bowtie1. We recommend using most standard flags, with the following modifications: use best, try hard, discard reads mapping to multiple locations, and have the output be in map format (as the calc_fit tool only takes inputs in this format). The number of allowed mismatches can be fiddled with, but should be relatively low to prevent false mapping.

After the data prep, everything else is done in the same way. The fitness analysis is done with the Calculate / Aggregate Fitness Workflow, and our set of optional Comparative Analyses & Other Tools can then be used to sort through your fitness data. There are also two optional visualization tools, but these are not available in Galaxy because Galaxy doesn't support R studio at the moment; if you want to use these you can follow their instructions under the Command Line / R section after running everything else in Galaxy.

3.A Data Prep

3.A.i MmeI Tn-Seq Prep

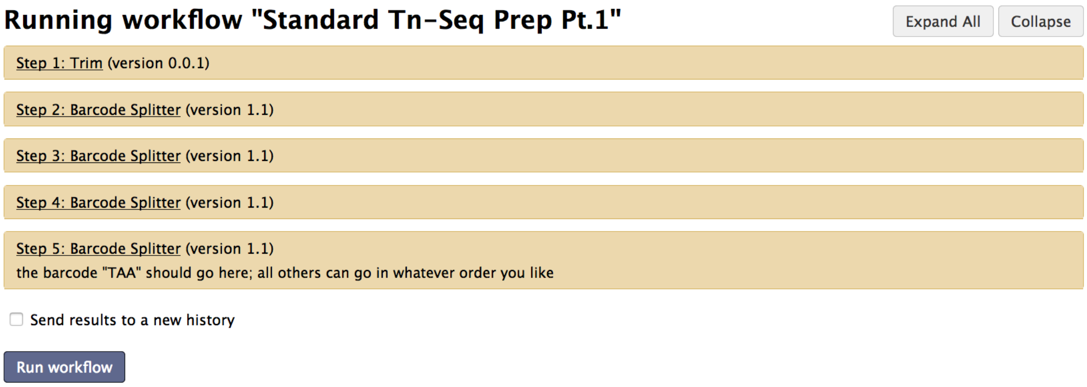
Download links: Pt 1, Pt 2, and Pt 3

This prep assumes a 50nt read, as described in Fig. 1 A), with the first 9 nucleotides from the adapter, the next six from its barcode, the next 15 to 18 from genomic DNA, and the last 20 to 23 from its transposon. The length of the genomic DNA turns out to be slightly variable because we found, to our surprise, that MmeI doesn't always cut exactly 20 bases downstream of its recognition site; it varies slightly so that the number of transposon nucleotides vary between 20 and 23. This in turn means that the bacterial gDNA sandwiched between the transposon and the adapter sequences also varies in length, making it impossible to automatically trim each read. Instead we split by the length and sequence of the trailing transposon sequences that a read can have (TAA, TAAC, TAACA, or TAACAG) and place them in four different files named according to the length the ending will be trimmed by later (e.g. Trim1, Trim2, Trim3. Trim4).

More specifically: in Part 1 this prep trims the first 9 and last 17 nucleotides, then splits by the potential segments of transposon DNA (TAA, TAAC, TAACA, or TAACAG) left on its end using the barcode splitter. In part 2 it trims that remaining transposon DNA, concatenates the files back into one, and splits by experimental barcode with the barcode splitter. In Part 3 it trims the barcodes from the reads and puts the barcode names in the file names so that they're labeled by experimental condition, filters the reads, collapses them, and maps them to the provided genome.

Part 1: trim & split by trailing transposon sequence

- Navigate to the workflow tab, click on the "Standard Tn-Seq Prep Pt.1" workflow, and select "Run". You should see a workflow with these steps:



Running workflow "Standard Tn-Seq Prep Pt.1"

Step 1: Trim (version 0.0.1)

Step 2: Barcode Splitter (version 1.1)

Step 3: Barcode Splitter (version 1.1)

Step 4: Barcode Splitter (version 1.1)

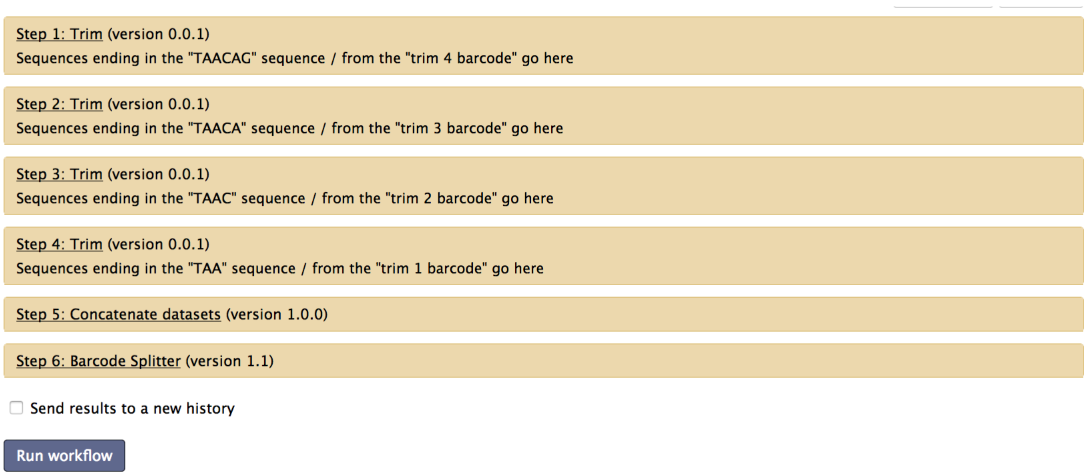
Step 5: Barcode Splitter (version 1.1)
the barcode "TAA" should go here; all others can go in whatever order you like

☐ Send results to a new history

- Select your FASTQ file under "this dataset" in "Step 1: Trim". The default settings will trim of the first 9 and the last 17nts.
- Select your four variable-MmeI-ending files in Steps 2 through 5, one per step. Note that these transposon-endings are entered as "barcodes" in the workflow even though they're not real barcodes, because we are using the barcode splitter tool off label to identify trailing transposon DNA sequences. They can go in whichever order you wish, except that the "trim1" barcode (finds sequences ending in TAA) should go under Step 5, which allows for 0 mismatches, because it is only 3 bp long so even a single mismatch is too loose a constraint.
- Press "Run workflow", and wait for the workflow to finish running.

Part 2: trim trailing transposon sequence, concatenate files together, and split by barcode

- Navigate to the workflow tab, click on the “Standard Tn-Seq Prep Pt.2” workflow, and select “Run”. You should see a workflow with these steps:



Step 1: Trim (version 0.0.1)
Sequences ending in the "TAACAG" sequence / from the "trim 4 barcode" go here

Step 2: Trim (version 0.0.1)
Sequences ending in the "TAACA" sequence / from the "trim 3 barcode" go here

Step 3: Trim (version 0.0.1)
Sequences ending in the "TAAC" sequence / from the "trim 2 barcode" go here

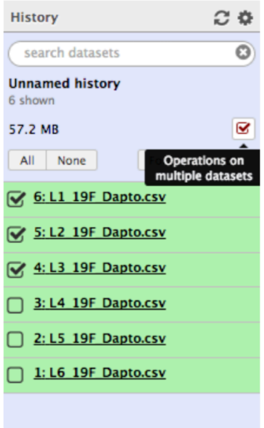
Step 4: Trim (version 0.0.1)
Sequences ending in the "TAA" sequence / from the "trim 1 barcode" go here

Step 5: Concatenate datasets (version 1.0.0)

Step 6: Barcode Splitter (version 1.1)

□ Send results to a new history

- In this workflow each of four output files from the last workflow is entered separately, in order to trim off the right number of nucleotides. It is thus essential you select the correct file for each ”Trim” step as follows:
- Select your trim4 variable-MmeI-ending split file under “this dataset” for “Step 1: Trim” (the step with “-4” under “Remove everything from this position to the end”).
- Select your trim3 variable-MmeI-ending split file under “this dataset” for “Step 2: Trim” (the step with “-3” under “Remove everything from this position to the end”).
- Select your trim2 variable-MmeI-ending split file under “this dataset” for “Step 3: Trim” (the step with “-1” under “Remove everything from this position to the end”).
- Select your trim1 variable-MmeI-ending split file under “this dataset” for “Step 4: Trim” (the step with “-2” under “Remove everything from this position to the end”).
- Next, select your barcode file under “Barcodes to use” in “Step 6: Barcode Splitter”. These should be the barcodes identifying which experimental condition each read is from.
- Then press “Run workflow”, and wait for the workflow to finish running.
- Finally, refresh your history and make a list out of the resulting files to send into the next workflow, via the “operations on multiple datasets” button. The button looks like this and is in Galaxy’s history sidebar:



History

search datasets

Unnamed history
6 shown

57.2 MB ☑

All | None

Operations on multiple datasets

☑ 6: L1 19F Dapto.csv

☑ 5: L2 19F Dapto.csv

☑ 4: L3 19F Dapto.csv

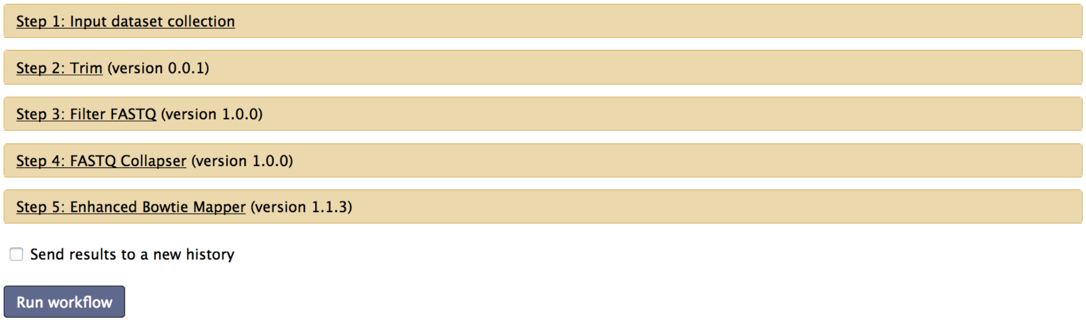
□ 3: L4 19F Dapto.csv

□ 2: L5 19F Dapto.csv

□ 1: L6 19F Dapto.csv

Part 3: trim barcode, filter reads by quality, collapse reads, and map them

- Navigate to the workflow tab, click on the "Standard Tn-Seq Prep Pt.3" workflow, and select "Run". You should see a workflow with these steps:



Step 1: Input dataset collection

Step 2: Trim (version 0.0.1)

Step 3: Filter FASTQ (version 1.0.0)

Step 4: FASTQ Collapser (version 1.0.0)

Step 5: Enhanced Bowtie Mapper (version 1.1.3)

□ Send results to a new history

Run workflow

- Under Step 1, input the dataset you made out of the last workflow's output.
- Under Step 2, reads are trimmed by our standard barcode length, which is 6 nucleotides. If your barcode length is different, change the amount trimmed to that under "Trim from the beginning up to this position".
- Under Step 3, reads are filtered by quality with a relatively low quality filter of 8, which is fine here because the mapping step below is very stringent. However, you can raise this if you like, under "Minimum quality score".
- Under step 4, identical reads are first collapsed into a 'virtual' single read. This collapsing of reads speeds up the mapping (i.e. it doesn't have to map the same read over and over), however, it is important to note that the quantitative information of each collapsed read is maintained. This means we still know how many reads are collapsed into one, which is essential information in order to maintain the ability to later obtain a quantitative read-out in the shape of fitness or if your prefer a simple ratio. You don't need to change anything here, we just provide this short explanation so you understand what's happening here.
- Under "Step 5: Map with Bowtie for Illumina", select your fasta genome under "Select the reference genome". Make sure this file is in ".fa" format, and do not use the ".gbk" file here.
- Under "Bowtie settings to use", the rest of the options are currently set to how the van Opijnen Lab normally runs this mapping step. But if you have good reason to change mapping settings, do that here. Right now it will allow for one mismatch, only map a read if it can be uniquely placed at one location in the genome, and will "try hard" to find the best mapping locations. Especially the number of allowed mismatches can be fiddled with, but

should be relatively low to prevent false-positives with respect to mapping.

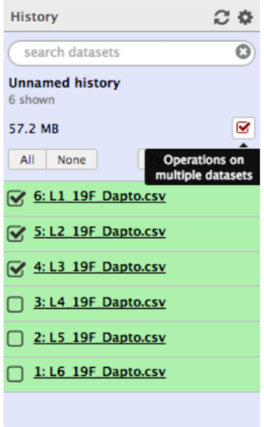
- Press “Run workflow”, and wait for the workflow to finish running.

3.A.ii Randomly Sheared Tn-Seq Prep

Download link

As described in Fig. 1 B, another method to prepare your Tn-Seq sample involves DNA shearing or fragmentation, followed by C-tailing or A-tailing and ligating on an adapter. This prep is designed for that method, and assumes a 50nt read of genomic DNA followed by an adapter of repeating C tail of variable length due to the randomness of the shearing. With Cutadapt the specific sequence can be removed, followed by read grooming, filtering for quality, collapsing, and mapping. If your reads also have barcodes, you should use the barcode splitter tool to separate them by barcode first.

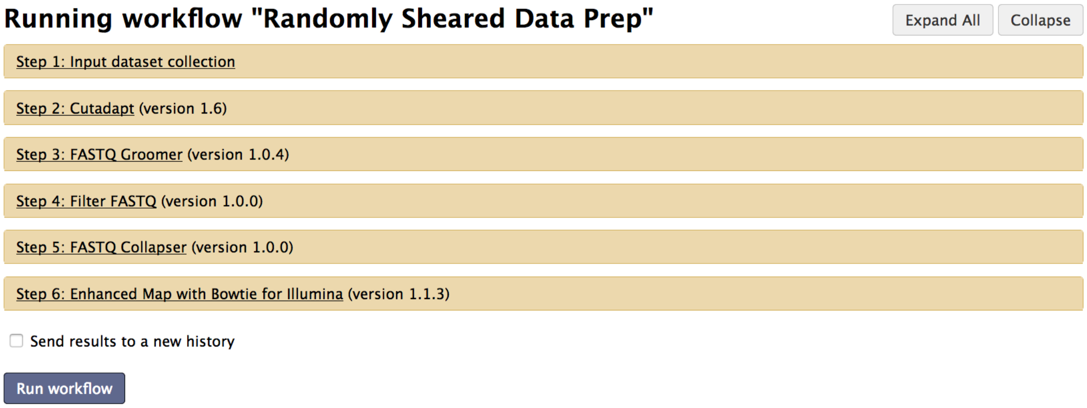
- First make a list out of all the sequence files you want to analyze, via the "operations on multiple datasets" button. The button looks like this and is in Galaxy's history sidebar:



Unnamed history
6 shown
57.2 MB

☑ 6: L1_19F_Dapto.csv
☑ 5: L2_19F_Dapto.csv
☑ 4: L3_19F_Dapto.csv
□ 3: L4_19F_Dapto.csv
□ 2: L5_19F_Dapto.csv
□ 1: L6_19F_Dapto.csv

- Next navigate to the workflow tab, click on the "Randomly Sheared Data Prep" workflow, and select "Run". You should see a workflow with these steps:



Running workflow "Randomly Sheared Data Prep"

Step 1: Input dataset collection

Step 2: Cutadapt (version 1.6)

Step 3: FASTQ Groomer (version 1.0.4)

Step 4: Filter FASTQ (version 1.0.0)

Step 5: FASTQ Collapser (version 1.0.0)

Step 6: Enhanced Map with Bowtie for Illumina (version 1.1.3)

□ Send results to a new history

- Enter the list of your files under "Step 1: Input dataset collection"
- If you used an A-tail method, click "Step2: Cutadapt" to expand it and replace the string of "C"s with the adapter sequence. Otherwise if you used a C-tail, leave it as it is.
- Click "Step 6: Map with Bowtie for Illumina" to expand it, and select your fasta genome at "Select the reference genome". Make sure this file is in ".fa" format, and do not use the ".gbk" file here.

- Under ”Bowtie settings to use”, the options are currently set to how the van Opijnen Lab normally runs this mapping step. But if you have good reason to change mapping settings, do that here. Right now it will allow for one mismatch, only map a read if it can be uniquely placed at one location in the genome, and ”try hard” to find the best mapping locations. The number of allowed mismatches particularly can be fiddled with, but should be relatively low to prevent false positives.
- Press “Run workflow”, and wait for the workflow to finish running.

3.B Fitness Analysis

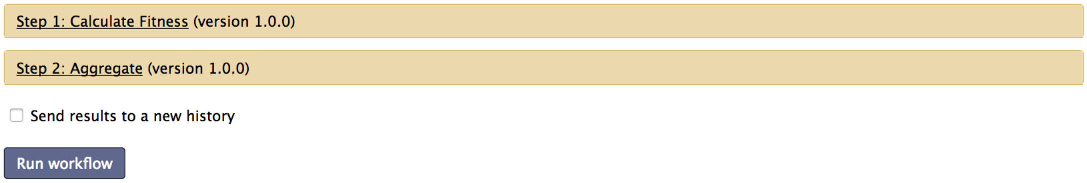
3.B.i Calculate_Fitness and Aggregate_Fitness

Download link

This workflow first calculates fitness for each insertion in the genome, then aggregates them by gene (or specified genetic region). If you have different libraries or replicates you will want to do this for each library. You can then use the aggregate tool on multiple Calculate Fitness output files from the same condition (but different libraries or replicates) to make 'super' aggregate files. In a super aggregate file fitness is calculated for each gene by averaging over all insertions across multiple libraries. This latter option thus not only captures variation that arises from multiple insertions in a specific region but also captures the variation that may arise from performing multiple replicate experiments.

Note that this workflow cannot be run in batch, because each run requires a unique ”expansion factor”, so you have to repeat the below steps for each time 2 (T2) measurement you performed.

- Navigate to the workflow tab, click on the “Calculate / Aggregate Fitness” workflow, and select “Run”. You should see a workflow with these steps:



Step 1: Calculate Fitness (version 1.0.0)

Step 2: Aggregate (version 1.0.0)

☐ Send results to a new history

- In “Step 1: Calculate Fitness” select your map file from T1 and map file from T2 (these are bowtie mapfiles containing the mapped reads from T1 and T2 respectively), the GenBank reference genome (.gbk file), and potentially the gene normalization file.
- If you expect to have reads from both sides flanking your transposon choose ”both”.
- Click the edit symbol under “Expansion Factor” and edit the value to match the expansion factor.
- Cutoff 1. This cutoff works on all insertions and ignores insertions with a read count less then the number set. Importantly, insertions with a low number of reads can skew the data.
- Cutoff 2. This cutoff only works on the normalization genes when calculating a normalization factor, and ignores insertions with a read count less then the number set. This is also to avoid skewing your data.
- Exclude first %, will exclude insertions that are located in the 5' part of a genetic region (e.g. gene). For instance the value ”0.1” will ignore any insertion located in the first 10% of a gene.
- Exclude last %, will exclude insertions that are located in the 3' part of a genetic region (e.g. gene). For instance the value ”0.1” will ignore any insertion located in the last 10% of a gene.
- Maximum weight of a gene in normalization calculations. Insertions with a low number of reads can skew the data (i.e. small changes from T1 to T2 can have disproportionate effects), therefore we give insertions with more reads a higher weight, however we can set a limit to what we consider a number of reads above which we find anything trustworthy. The default level is set to 75, which means that any insertion with more then 75 reads is as trustworthy as an insertion can get, lower then 75 the trustworthiness slowly declines. If your data contains many reads this value will have little effect on the outcome, however if you have a low number of reads this number

can become more important and have a higher impact. Note that this weight only affects the trustworthiness of the normalization genes, below we'll set the weight for all the other insertions.

- Multiply fitness scores by a certain value. This enables you to multiply each fitness value with a specific value. This could be useful if you are building a Genetic Interaction Map, and your query gene has a fitness lower then 1. For more info about this for instance see the following papers (1, 2, 3, 4).
- Set reads manually. We don't recommend using this function. It's more a remnant then anything else.

This step in the workflow will create three output files: **1.** A .csv file containing the fitness values calculated for each insertion (position). **2.** A .txt file containing the bottleneck value. **3.** A .wig file that can be used for visualization of each insertion and its corresponding fitness (see the visualization section below).

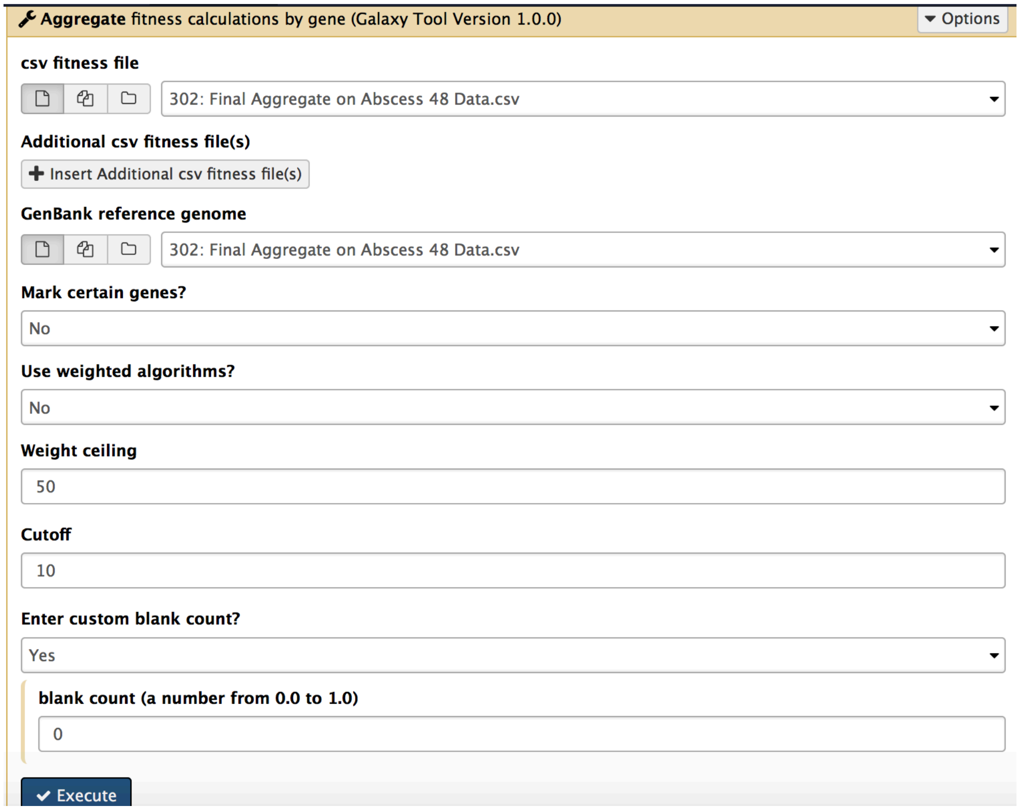
The bottleneck value is very important under certain circumstances, because it can estimate the bottleneck (random loss of insertions) your experiment may have gone through and correct your fitness calculations by removing its influence. In *in vitro* experiments a bottleneck may not, and probably should not, occur. However, in *in vivo* experiments a bottleneck does often occur. This means that you may start with a certain amount of insertion mutants but randomly loose a certain percentage due to reasons that have nothing to do with low fitness.

To calculate the bottleneck we simply add up the number of insertions we started with (T1), and the number of insertions we end with (i.e. insertions that we have reads for at T2). Normally we can assume that most insertions will have no effect on fitness and therefore by dividing the number of insertions at T2 over T1 tells us the percentage of insertions we are left with, and thus also how much we lost (also see next paragraph for explanation of the assumption). For instance if we start with 10,000 insertions and we end up with 5,000, we lost 50% of our insertions during the experiment. To correct for this we remove 50% of the insertions with fitness 0 (those insertions that disappeared at T2) from each specific genetic region (e.g. gene). For instance if a gene had 10 insertions at T1, and at T2 five insertions have reads and thus a fitness >0, and 5 have no reads and thus have a fitness of 0, we remove 50% of the insertions but only if their fitness is 0, which means in this case we remove the five insertions with a fitness of 0. Another example, if a gene had 10 insertions at T1, and at T2 three insertions have reads and thus a fitness >0, and seven have no insertions and thus have a fitness of 0, we remove 50% of the insertions but only if they are 0, which means in this case we still remove five insertions with a fitness of 0. Therefore the average fitness is made up of five fitness values, of which three have a fitness >0 and 2 have a fitness of 0. For more on this read these papers (2, 3,).

N.B. We often calculate the bottleneck in two ways. The first way takes into account all insertions in the population (as described above), however if you have listed a set of normalization genes the bottleneck is also calculated by only considering the insertions in these genes. As described, the normalization genes are considered to never have an effect on fitness. Insertions in these genes should thus be neutral. And thus if these insertions disappear from your experiment it is indicative of a bottleneck. By dividing the insertions in the normalization genes from T2 over T1 will therefore also tell you the size of the bottleneck. It turns out that in a vast majority of the experiments the van Opijnen Lab has performed that the same bottleneck size is obtained when it is calculated over the complete population of insertions or only over the insertions in the normalization genes, which is because in general most insertions will not have an effect on fitness.

- In “Step 2: Aggregate”, Select your GenBank reference genome under the corresponding header.
- If you'd like to mark certain genes (like say, normalization genes, for reference) chose “Yes” under “Mark certain genes?”. Then chose your normalization genes file or a similarly formatted file as input. The only thing that this will do is create a column in your output file and list an ”M” if a gene was used for normalization. If you don't know what we mean with normalization see page 2 for an explanation.
- If you'd like to use a weighted algorithm, which we recommend, leave that option on ”yes”. The weighted fitness algorithm gives slightly more importance to insertions composed of many reads. The number is specified below by the ”Weight ceiling”.

- If you'd like to change the weight ceiling, you can; the default is a value that generally works well. Insertions with a low number of reads can skew the data (i.e. small changes from T1 to T2 can have disproportionate effects), therefore we give insertions with more reads a higher weight, however we can set a limit to what we consider a number of reads above which we find anything trustworthy. The default level is set to 50, which means that any insertion with more then 50 reads is as trustworthy as an insertion can get, lower then 50 the trustworthiness slowly declines. If your data contains many reads this value will have little effect on the outcome, however if you have a low number of reads this number can become more important and have a higher impact. Note that the weight above affects the trustworthiness of the normalization genes, while here we set the weight for all the other insertions.
- If you'd like to change cutoff 3, you can; it is 10 by default and ignores any insertion that has less then the specified number of reads.
- Press "Run workflow", and wait for the workflow to finish running.
- Once this is done and if you have data from different libraries or replicates under the same condition, you can aggregate that data all together (recommended). Do this by navigating to the "Aggregate" tool and using the option to "Insert additional CSV fitness files" as well as selecting "Yes" under "Enter custom bottleneck correction value?" as shown below. For the custom bottleneck value, simply average the bottleneck values of each fitness file you're using, which can be found in their corresponding calc fitness txt file output. Then enter all the fitness files from the same condition, and press "Execute"!



Aggregate fitness calculations by gene (Galaxy Tool Version 1.0.0)

csv fitness file: 302: Final Aggregate on Abscess 48 Data.csv

Additional csv fitness file(s)

GenBank reference genome: 302: Final Aggregate on Abscess 48 Data.csv

Mark certain genes?: No

Use weighted algorithms?: No

Weight ceiling: 50

Cutoff: 10

Enter custom blank count?: Yes

blank count (a number from 0.0 to 1.0): 0

OPTIONAL ANALYSES: FITNESS AND COMPARATIVE

The following fitness analyses (singleFitness and regionFitness) as well as all of the comparative analyses in the following sections are considered optional analyses, beyond the traditional Tn-Seq analysis pipeline, which includes pre-processing steps, calculating fitness for mutants in libraries, and calculating aggregate fitness for genes across multiple libraries.

Optional fitness analysis tools begin here. They require several Perl modules, all of which are available in the public Galaxy toolshed and should automatically install when magenta tools are installed from the public Galaxy toolshed.

3.B.ii SingleFitness: single fitness per insertion

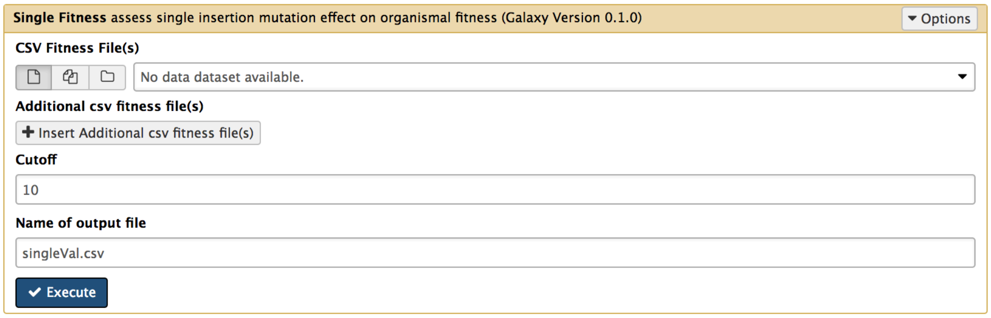
The SingleFitness tool aggregates fitness information across multiple libraries for one weighted fitness value per insertion. The method for calculating the weighted fitness score associated with a single insertion is similar to the weighted fitness score calculated for a gene in aggregate_fitness. In the end, a single fitness value is associated with a single mutation in the genome of the organism. This single value could represent multiple single insertion mutants with a library and across libraries.

Required input files:

- CSV Fitness Files outputted by Calc_fitness

Click here for the command-line implementation of SingleFitness and for more information on option inputs and outputs.

Galaxy Interface for SingleFitness tool:



Single Fitness assess single insertion mutation effect on organismal fitness (Galaxy Version 0.1.0)

CSV Fitness File(s)

No data dataset available.

Additional csv fitness file(s)

Cutoff

10

Name of output file

singleVal.csv

3.B.iii RegionFitness: non-gene-centric aggregate fitness

The RegionFitness tool is a non-gene-centric method for discovering regions important for organismal fitness. The method for calculating the aggregate fitness value of a gene relies on the genomic start and end coordinates of the region, which is retrievable from the organism's Genbank record. However, not all official genome records are complete or well annotated beyond genes. Therefore, to enable discovery of regions that may contain non-coding RNA's or other genetic elements that have a significant effect on fitness; this tool uses a sliding window approach to mine Tn-Seq data in a non-gene-centric manner.

Determining window size and step The entire genome and ordered Tn-Seq data is scanned by performing assessments of regions at a set size (base pair length of window) and step (base pairs between the start of each window). Based on the genome used, typically a window size between 250 bp and 500 bp is small enough to pick up intergenic regions or gene domains but large enough to contain enough TA sites and insertions. However, this can vary depending on organism and insertion density. The DataOverview tool described below can help determine these factors.

Method In each regional profile, aggregate fitness for insertions and the insertion representation in that region is calculated. To assess insertion representation, we follow the method used by Zhang et. al. For a given region, with x insertions and n TA sites, 10,000 random sets of n TA sites are selected and the average of insertions over TA sites is calculated for each. The 10,000 resulting means create a null distribution. The p-value can be derived for the original region of interest by ranking it on the distribution for x/n. A statistically underrepresented number of insertions for a region will be significant if the p-value is below the critical value. Gene annotations are also recorded if they are contained in the region.

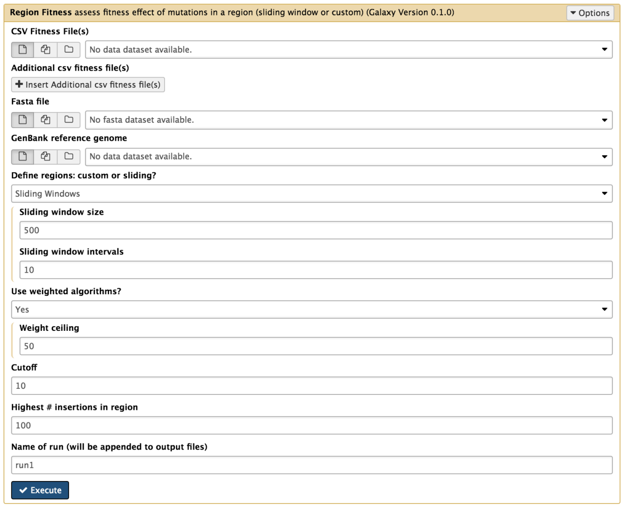
Galaxy Usage The Region Fitness tool requires the same inputs as AggregateFitness, with additional options for specifying the window size and step for the sliding window approach, or simply a tab-delimited file with start and end coordinates for custom regions. The input csv files will come from the output of CalculateFitness. All of these options and inputs are described in the Galaxy tool interface and in the command-line usage instructions.

Required input files:

- CSV Fitness File(s) output from Calc_fitness
- Fasta file of the organism's genome sequence
- Genbank file for the organism's annotated genome

For more information on required and optional inputs for this tool, see the documentation for the RegionFitness command-line implementation or view the help section in the RegionFitness Galaxy tool.

Galaxy Interface for RegionFitness tool:



Region Fitness assess fitness effect of mutations in a region (sliding window or custom) (Galaxy Version 0.1.0)

CSV Fitness File(s): No data dataset available.

Additional csv fitness file(s)

Fasta file: No fasta dataset available.

GenBank reference genome: No data dataset available.

Define regions: custom or sliding?: Sliding Windows

Sliding window size: 500

Sliding window intervals: 10

Use weighted algorithms?: Yes

Weight ceiling: 50

Cutoff: 10

Highest # insertions in region: 100

Name of run (will be appended to output files): run1

3.C Data Mining and Comparative Analyses

Table 1. A summary of data mining and comparative analysis tools in MAGenTA.

Tool	Function	Research Questions Addressed
Data Overview	Produces a summary file of information about TA sites and insertions in the genome, libraries, and gene/intergenic regions. Requires calc_fitness output files as inputs.	What is the TA and insertion distribution in the genome for genes and non-gene regions?
Compare Genes	Creates a gene comparison file with a significance test for the difference in fitness for each gene for two Tn-Seq experiments of the same ref genome. Requires aggregate_fitness output	How can we compare genes for a given strain in two different conditions? Which genes are conditionally important?
Compare Strains	Creates a gene comparison file with significant stats for two Tn-Seq experiments of different strains (genomes). Requires a file of homologous genes aggregate_fitness output files as inputs.	How can we compare homologous genes for two strains? Which genes have a strain-dependent importance?
Compare Regions	Creates a comparison file with significance statistics for two Tn-Seq experiments of different strains (genomes). Requires RegionFitness output files as input.	How can we compare intergenic or gene domain regions for the same strain in two conditions?

3.C.i DataOverview: dataset and genome summary

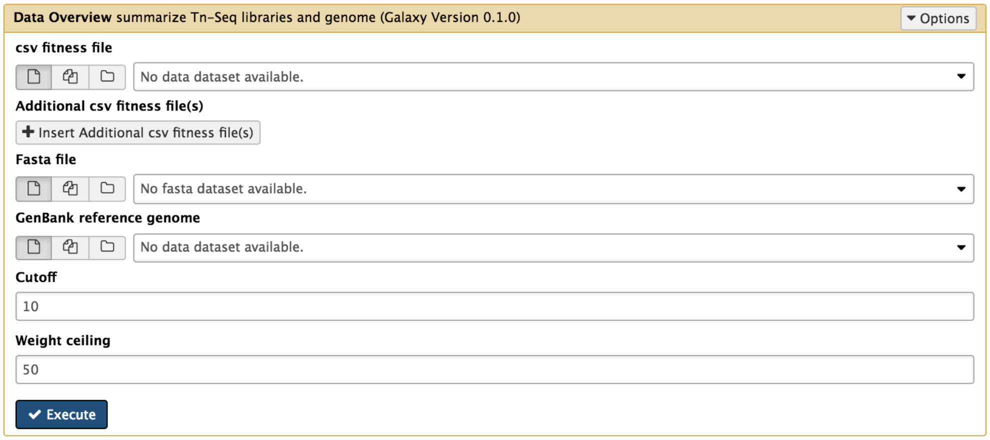
Prior to Tn-seq data analysis it is important to understand the genome-wide scale and context of the data. Large differences in sequencing coverage, insertion representation, and GC content between genomes can affect the robustness of the comparative analyses performed by MAGenTA. Tn-Seq is also generally performed with replicate libraries, where differences (if they exist) would be important to know. The DataOverview tool produces a summary of genome-wide insertion representation and inherent genome characteristics shown in Table 2. The input data are queried for TA sites and insertions with respect to the genome, open reading frames denoted by start and stop codons, annotated genes from the Genbank record, and intergenic regions

Required input files:

- CSV Fitness File(s) output from Calc_fitness
- Fasta file of the organism's genome sequence
- Genbank file for the organism's annotated genome

For more information on required and optional inputs for this tool, see the documentation for the DataOverview command-line implementation or view the help section in the DataOverview Galaxy tool.

Galaxy Interface for DataOverview tool:



Data Overview summarize Tn-Seq libraries and genome (Galaxy Version 0.1.0)

csv fitness file

No data dataset available.

Additional csv fitness file(s)

Fasta file

No fasta dataset available.

GenBank reference genome

No data dataset available.

Cutoff

10

Weight ceiling

50

Table 2. The DataOverview tool outputs genome and Tn-Seq related summary information. The summary includes genome and gene-wide insertion representation for each library, including how many insertions would be filtered out given a cutoff requirement for the number of reads representing each insertion.

	Strain1	Strain2
Genbank Record	NC_003028	NC_0012469
Genome Size (bp)	2160842	2112148
GC content	39.70%	39.77%
Number of Genes	2390	2204
Total number of TA sites	141459	137693
Genome coverage by TA sites	6.55%	6.52%
Largest gap between TA sites (bp)	252	206
Genome coverage by insertions	1.33%	3.68%
TA site coverage by insertions	32.80%	73.12%
TA site coverage by filtered insertions*	20.28%	56.45%
Largest gap between insertions (bp)	14996	6708
Smallest ORF (bp)	6	6
Largest ORF (bp)	220	220
Max number TA sites in an ORF	10	10
Number of ORFs without TA sites	6997	6701
Average gene length	803	834
Minimum gene length	30	65
Max gene length	14330	6701
Average insertions in a gene	9.84	28.4
Minimum insertions in a gene	0	0
Maximum insertions in a gene	237	420
Genes with insertions	65%	77%
Number of genes without insertions	564	257
Insertions outside of annotated genes	15773	43266
Insertions in intergenic regions**	54.98%	55.66%

3.C.ii CompareGenes: compare genes for identical strains

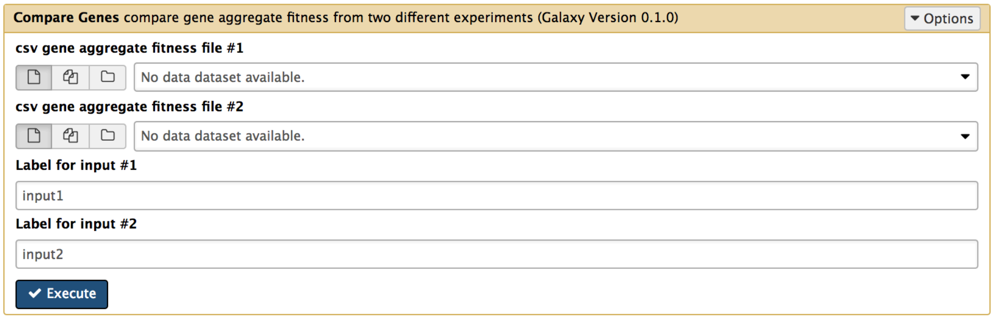
Comparative gene analysis for two experiments of the same organism (i.e. One strain tested in Glucose and in Glucose+Antibiotic) involves merging two output files from the fitness script, then performing a t-test and calculating average fitness difference for each gene. This process can be performed in a spreadsheet application, such as Excel. However, CompareGenes can perform it in less than three seconds for 2000 genes.

Required input files:

- Two (2) CSV Aggregate (Gene) Fitness files outputted by Aggregate_fitness

Labels for each file can be specified to be appended to the respective columns in the merged output file and to name the resulting output file. Click here for the command-line implementation of CompareGenes.

Galaxy Interface for CompareGenes tool:



Compare Genes compare gene aggregate fitness from two different experiments (Galaxy Version 0.1.0)

csv gene aggregate fitness file #1

No data dataset available.

csv gene aggregate fitness file #2

No data dataset available.

Label for input #1

input1

Label for input #2

input2

3.C.iii CompareStrain: compare genes for different strains

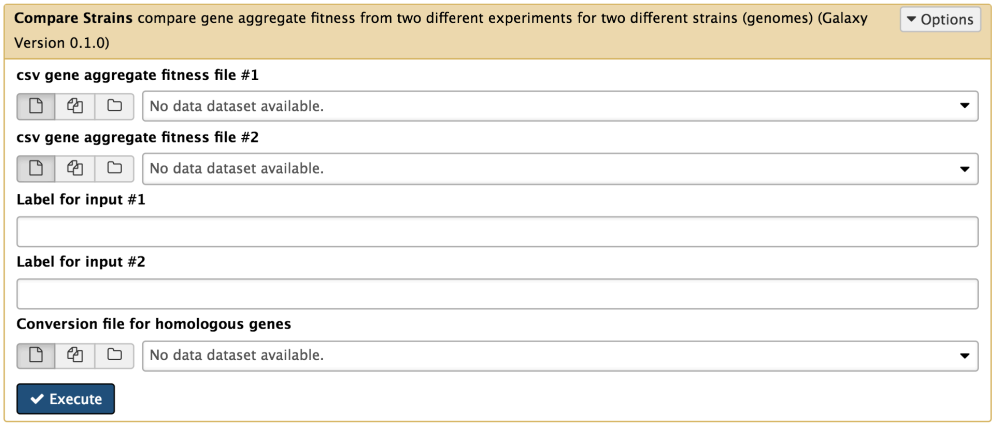
The CompareStrain tool, a variation of CompareGenes, is specifically for gene comparisons across two experiments from different organisms/strains. Since gene identification numbers will be different at the organism/strain level, a file containing the homologous genes is required. The same output as CompareGenes will be produced.

Required input files:

- Two (2) Aggregate (Gene) Fitness files (CSV) output from Aggregate_fitness
- A tab-delimited text (.txt) file containing the corresponding gene homologs for the two strains. One homology set per line, each gene ID separated by a tab. So that one column of the file will have all of the gene IDs for one strain and the second column will contain the corresponding homologs for those genes for the other strain.

Click here for the command-line implementation of CompareStrains.

Galaxy Interface for CompareStrains tool:



Compare Strains compare gene aggregate fitness from two different experiments for two different strains (genomes) (Galaxy Version 0.1.0)

csv gene aggregate fitness file #1

No data dataset available.

csv gene aggregate fitness file #2

No data dataset available.

Label for input #1

Label for input #2

Conversion file for homologous genes

No data dataset available.

3.C.iv CompareRegions: compare any region for identical strains

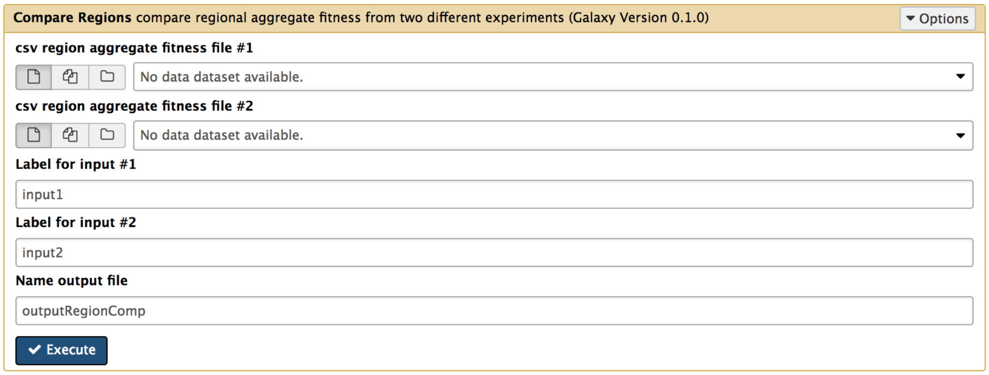
CompareRegions takes results from the RegionFitness tool and performs a similar comparative analysis as done on genes by CompareGenes and CompareStrains. Since genomic start and end coordinates define windows (or regions), only comparisons between datasets from the same organism (same genome) make sense.

Required input files:

- Two (2) files output from RegionFitness

Labels for each file can be specified to be appended to the respective columns in the merged output file and to name the resulting output file. Click here for the command-line implementation of CompareRegions.

Galaxy Interface for CompareRegions tool:



Compare Regions compare regional aggregate fitness from two different experiments (Galaxy Version 0.1.0)

csv region aggregate fitness file #1: No data dataset available.

csv region aggregate fitness file #2: No data dataset available.

Label for input #1: input1

Label for input #2: input2

Name output file: outputRegionComp

4 Analysis via Command Line / R

If you know a bit of coding, and would rather run our tools through the command line, you can do so as follows:

4.A Data Prep

- Download and install the fastx_toolkit and bowtie1. For more on the usage of the fastx_toolkit see its manual, and for more on Bowtie see its manual (Note that we recommend using Bowtie 1).
- Split data by experimental condition (i.e. barcodes) with fastx barcode splitter and remove all non-genomic DNA (e.g. barcodes) with fastx trimmer. These steps may be done in whatever order best suits your data.
- Filter reads by quality (e.g. a minimum quality of 8) using fastq quality filter.
- Collapse reads using fastq collapser.
- Map reads using Bowtie 1. We recommend using the following flags: -n 1 (maximum 1 mismatch) –best (whether or not to make Bowtie guarantee that reported singleton alignments are 'best' in terms of stratum and quality) -y (try hard) and -m 1 (supress all alignments if more than one can be found)

4.B Fitness Analysis

4.B.i Downloading command-line tools

Command-line tools for MAGenTA are open source and available through GitHub

```
git clone https://github.com/vanOpijnenLab/magenta.git
```

Or download the MAGenTA project repository

4.B.ii Calc_Fitness

Usage

```
python calc_fitness.py <options> -t1 <t1.map> -t2 <t2.map> -out <outfile.csv>
```

For every pair of T1/T2 reads, calculate fitness for each insertion location using calc_fitness.py. Fitness is calculated as described in the 2009 Tn-Seq paper under the ”Fitness Calculations” section.

Required Inputs

Flag	Description
ref	The name of the reference genome file, in GenBank format
t1	The name of the bowtie mapfile from time point 1
t2	The name of the bowtie mapfile from time point 2
out	Name of a file to enter the .csv output

Optional Inputs

Flag	Description
expansion	Expansion factor (default: 250)

Flag	Description
reads1	The number of reads to be used to calculate the correction factor for time 0 (default counted from bowtie output)
reads2	The number of reads to be used to calculate the correction factor for time 1 (default counted from bowtie output)
cutoff1	Discard any positions where the average of counted transcripts at time 0 and time 1 is below this number (default 0)
cutoff2	Discard any positions within the normalization genes where the average of counted transcripts at time 0 and time 1 is below this number (default 10)
strand	Use only the specified strand (+ or -) when counting transcripts (default: both)
normalize	A file that contains a list of genes that should have a fitness of 1 - used for normalization and bottleneck calculations.
b	Calculate bottleneck value (the percentage of insertions randomly lost) from all genes (rather than only normalization genes). Bottleneck values are used for normalization of the data in Aggregate_Fitness. If you don't input a file of normalization genes make sure to chose this flag.
maxweight	The maximum weight a gene can have in normalization calculations
multiply	Multiply all fitness scores by a certain value (e.g., the fitness of a knockout).
ef	Exclude insertions that occur in the first N amount (%) of gene.
el	Exclude insertions in the last N amount (%) of the gene.
wig	Create a wiggle file for viewing in a genome browser. Provide a filename.
uncol	Use if reads were uncollapsed when mapped.

4.B.iii Aggregate_Fitness

Usage

This takes in csv file(s) outputted by Calculate_Fitness and containing individual insertion fitnesses, and aggregates the fitness by gene. As described in the aggregate.py usage, you can input multiple insertion location fitness files, which will generate a ”super aggregate” file that incorporates the data from all your libraries and or replicates. This is highly recommended.

```
python aggregate.py -out <output.csv> <options> <fitness1.csv fitness2.csv fitness3.csv ...>
```

Required Inputs

Flag	Description
out	Name of a file to enter the .csv output calcfitness csv output files without a flag

Optional Inputs

Flag	Description
c	Check for missing genes in the data set - provide a reference genome in Genbank format. Missing genes will be sent to stdout
m	Place a mark in an extra column for this set of genes. Provide a file with a list of genes seperated by newlines
x	Cutoff: Don't include fitness scores with average counts (c1+c2)/2 < x (default: 10)
b	Bottleneck value: The percentage of insertions randomly lost, calculated upstream by Calculate_Fitness, which will be discounted for all genes (for example, 20% would be entered as 0.20. If you are inputting multiple fitness files then average their bottleneck values and input that. Default 0.0)

Flag	Description
f	An in-between file carrying information on the blank count found from calc_fitness or consol_fitness, one of two ways to pass a blank count to this script
w	Use weighted algorithm in calculations
l	Weight ceiling: maximum value to use as a weight (default: 999,999)

OPTIONAL ANALYSES: FITNESS AND COMPARATIVE

The following fitness analyses (singleFitness and regionFitness) as well as all of the comparative analyses in the following sections are considered optional analyses, beyond the traditional Tn-Seq analysis pipeline, which includes pre-processing steps, calculating fitness for mutants in libraries, and calculating aggregate fitness for genes across multiple libraries. **The following tools require the following Perl modules to be installed:**

- Bio::SeqIO
- Getopt::Long
- Data::Dumper
- Statistics::Distributions
- Data::Random
- List::BinarySearch

To install the above modules, follow the instructions discussed in detail here and summarized below :

1. Enter the computer's Terminal emulator
2. Install cpanm by typing in this command: **cpan App::cpanminus**
3. Install each module above by typing in this command **cpanm Module::Name**, substituting Module::Name for the module listed above (i.e. Bio::SeqIO)

4.B.iv SingleFitness

The SingleFitness tool aggregates fitness information across multiple libraries for one weighted fitness value per insertion. The method for calculating the weighted fitness score associated with a single insertion is similar to the weighted fitness score calculated for a gene in aggregate_fitness. In the end, a single fitness value is associated with a single gene or genetic region in the genome. This single value can represent multiple insertion mutants coming from a single library or from multiple libraries or replicates.

Usage

```
perl singleFitness.pl <options> <inputs or -d directory>
```

Required Inputs

Flag	Description
d	Directory containing all input files OR individual input filenames(s) in the command line (without a flag) from Calc_fitness output

Optional Inputs

Flag	Description
h	Print usage and exits program
o	Filename for output. Default: compFile1File2.csv

Flag	Description
l	Labels for input files. Two strings, comma separated (i.e. -l expt1,expt2). Order should match file order. Default: input filenames.
v	String value for output: ‘fit’ for fitness OR ‘count’ for count print. Default: “fit” for fitness
n	Name of the reference genome, to be included in the wig header. Default: genome
o	Output file for comparison data. Default: singleVal.csv

4.B.v RegionFitness

Usage

```
perl regionFitness.pl <options> <inputs or -d directory>
```

The RegionFitness tool is a non-gene-centric method for discovering regions important for organismal fitness. The method for calculating the aggregate fitness value of a gene relies on the genomic start and end coordinates of the region, which is retrievable from the organism’s Genbank record. However, not all official genome records are complete or well annotated beyond genes. Therefore, to enable discovery of regions that may contain non-coding RNA’s or other genetic elements that have a significant effect on fitness; this tool uses a sliding window approach to mine Tn-Seq data in a non-gene-centric manner.

Determining window size and step The entire genome and ordered Tn-Seq data is scanned by performing assessments of regions at a set size (base pair length of window) and step (base pairs between the start of each window). Based on the genome used, typically a window size between 250 bp and 500 bp is small enough to pick up intergenic regions or gene domains but large enough to contain enough TA sites and insertions. However, this can vary depending on organism and insertion density. The DataOverview tool described below can help determine these factors.

Method In each regional profile, aggregate fitness for insertions and the insertion representation in that region is calculated. To assess insertion representation, we follow the method used by Zhang et. al. For a given region, with x insertions and n TA sites, 10,000 random sets of n TA sites are selected and the average of insertions over TA sites is calculated for each. The 10,000 resulting means create a null distribution. The p-value can be derived for the original region of interest by ranking it on the distribution for x/n. A statistically underrepresented number of insertions for a region will be significant if the p-value is below the critical value. Gene annotations are also recorded if they are contained in the region.

Required Inputs

Flag	Description
d	Directory containing all input files OR in the command line without a flag input filename(s) (output files from Calc_fitness)
f	Filename for genome sequence in Fasta format
r	Filename for genome annotation in Genbank format

Optional Inputs

Flag	Description
h	Print usage
size	The size of the sliding window. Default=50
step	The window spacing. Default=10
x	Exclude values with avg. counts less than x where (c1+c2)/2<x. Default=15
log	Send all output to a log file instead of the terminal
max	Expected max number of TA sites in a window. Used for creating null distribution library. Default=100
o	Specify name of new directory for all output files
w	Do weighted average for fitness per insertion

Flag	Description
wc	Integer value for weight ceiling. Default=50

4.C Data Mining and Comparative Analyses

4.C.i DataOverview

Usage

```
perl dataOverview.pl <otpions> <inputs or -d directory>
```

Prior to Tn-seq data analysis it is important to understand the genome-wide scale and context of the data. Large differences in sequencing coverage, insertion representation, and GC content between genomes can affect the robustness of the comparative analyses performed by MAGenTA. Tn-Seq is also generally performed with replicate libraries, where differences (if they exist) would be important to know. The DataOverview tool produces a summary of genome-wide insertion representation and inherent genome characteristics shown in Table 2. The input data are queried for TA sites and insertions with respect to the genome, open reading frames denoted by start and stop codons, annotated genes from the Genbank record, and intergenic regions

	Strain1	Strain2
Genbank Record	NC_003028	NC_0012469
Genome Size (bp)	2160842	2112148
GC content	39.70%	39.77%
Number of Genes	2390	2204
Total number of TA sites	141459	137693
Genome coverage by TA sites	6.55%	6.52%
Largest gap between TA sites (bp)	252	206
Genome coverage by insertions	1.33%	3.68%
TA site coverage by insertions	32.80%	73.12%
TA site coverage by filtered insertions*	20.28%	56.45%
Largest gap between insertions (bp)	14996	6708
Smallest ORF (bp)	6	6
Largest ORF (bp)	220	220
Max number TA sites in an ORF	10	10
Number of ORFs without TA sites	6997	6701
Average gene length	803	834
Minimum gene length	30	65
Max gene length	14330	6701
Average insertions in a gene	9.84	28.4
Minimum insertions in a gene	0	0
Maximum insertions in a gene	237	420
Genes with insertions	65%	77%
Number of genes without insertions	564	257
Insertions outside of annotated genes	15773	43266
Insertions in intergenic regions**	54.98%	55.66%

Required Inputs

Flag	Description
d	Directory containing all input files (output files from calcFitness tool) OR In the command line (without a flag) input filename(s)

Flag	Description
f	Filename for genome sequence in fasta format
r	Filename for genome annotation in GenBank format

Optional Inputs

Flag	Description
h	Print usage
o	Filename for output. Default: standard output
c	Cutoff average(c1+c2)>c. Default: 15

4.C.ii CompareGenes

Comparative gene analysis for two experiments of the same organism (i.e. One strain tested in Glucose and in Glucose+Antibiotic) involves merging two output files from the fitness script, then performing a t-test and calculating the average fitness difference for each gene. This process can be performed in a spreadsheet application, such as Excel. However, GeneCompare can perform it in less than three seconds for 2000 genes.

Usage

```
perl compGenes.pl <options> <inputs or -d directory>
```

Required Inputs

Flag	Description
d	Directory containing all input files (output files from calcFitness tool) OR In the command line (without a flag) input filename(s)
c	Tab delimited file with homologous genes, one gene homolog pair per line

Optional Inputs

Flag	Description
h	Print usage
o	Filename for output. Default: compFile1File2.csv
l	Labels for input files. Two strings, comma separated (i.e. -l expt1,expt2). Order should match file order. Default: input filenames.

4.C.iii CompareStrains

The StrainCompare tool, a variation of GeneCompare, is specifically for gene comparisons across two experiments from different organisms/strains. Since gene identification numbers will be different at the organism/strain level, a file containing the homologous genes is required. The same output as GeneCompare will be produced and can be used in downstream analysis tools. An automated method for extracting homologous genes will be a useful future improvement to this tool.

Usage

```
perl compStrains.pl <options> <inputs or -d directory>
```

Required Inputs

Flag	Description
d	Directory containing all input files (output files from calcFitness tool) OR In the command line (without a flag) input filename(s)
c	Tab delimited file with homologous genes, one gene homolog pair per line

Optional Inputs

Flag	Description
h	Print usage
o	Filename for output. Default: compFile1File2.csv
l	Labels for input files. Two strings, comma separated (i.e. -l expt1,expt2). Order should match file order. Default: input filenames.

4.C.iv CompareRegions

CompareRegions takes results from the RegionFitness tool and performs a similar comparative analysis as done on genes by CompareGenes and CompareStrains. Since genomic start and end coordinates define windows (or regions), only comparisons between datasets from the same organism (same genome) make sense.

Usage

```
perl compRegions.pl <options> <inputs or -d directory>
```

Required Inputs

Flag	Description
d	Directory containing all input files (output files from calcFitness tool) OR In the command line (without a flag) input filename(s)

Optional Inputs

Flag	Description
h	Print usage
l	Labels for input files. Two strings, comma separated (i.e. -l expt1,expt2). Order should match file order. Default: input filenames.
o	Filename for output. Default: label1label2.csv

4.D Data Visualization in R

Fitness information is a central piece of data in Tn-Seq analysis. This data makes most sense in the context of genome-wide information including annotations (i.e. genes and non-coding RNAs), base information to identify possible TA insertion sites, and genome coordinates. Therefore, multi-track visualization methods are preferred for Tn-Seq data. MAGenTA output files can be formatted for multi-track visualization in GViz, an R package available through Bioconductor. A simpler option, single-track visualization is also possible in R. Both are demonstrated below.

Required input files for multi-track visualization in Gviz:

- 1-2 Output files from SingleFitness (galaxy or command-line)
- Fasta file of the organism's genome sequence

- Genbank file for the organism’s annotated genome

Required input files for single track visualization in R:

- 1 Output files from SingleFitness (galaxy or command-line)

4.D.i Multi-track Visualization of Tn-Seq data with Gviz

Installing Gviz and dependency packages

```
# During installation step, allow updates or installing of
# dependency packages if prompted

# Run these two lines to check if necessary packages are
# installed If not, they will be installed.
packages <- c("Biostrings", "seqinr", "Gviz")
if (length(setdiff(packages, rownames(installed.packages()))) >
    0) {
    install.packages(setdiff(packages, rownames(installed.packages())))
}
# Load libraries for installed packages
library(seqinr)
library(Biostrings)
library(Gviz)
```

Input Files

```
# Set working directory
# setwd('~/Documents/lab/tvolab/magenta/')

# Specify file paths in the beginning of a session

# Fasta file for the reference genome, available on NCBI genomes
genomeFile = "19F_012469.fasta"

# Gene Features formatted for Gviz track from a Genbank file using the
# GetGeneFeatures tool
# Format: tab-delimited text file with gene_id, gene_name, start, end , strand,
#  and function fields
geneFile = "19F_012469_genes.txt"

# Single fitness files are created by the SingleFitness tool (Galaxy or command-line)
#  using input library csv files from the CalculateFitness tool
#  Here, a control (glucose) file and experimental (glucose  + daptomycin) are used
ctrlSingleFitFile = "19fglucSingleFit.csv"
exptSingleFitFile = "19fdaptoSingleFit.csv"
```

```
# so Gviz doesn't look for NC_003028 in the UCSC site
options(ucscChromosomeNames = FALSE)
```

Genome Axis Track

```
# MAGenTA input: None Number coordinates for genome
gTrack <- GenomeAxisTrack(littleTicks = TRUE)
```

Sequence Track

```
# MAGenTA input: FASTA file (genomeFile).  Track for sequence (ATGC) in color
fcol <- c(A = "darkorange", C = "yellow", T = "darkred", G = "darkgreen")
sTrack <- SequenceTrack(genomeFile, name = "Sequence", fontcolor = fcol)
```

Genome Annotation Track

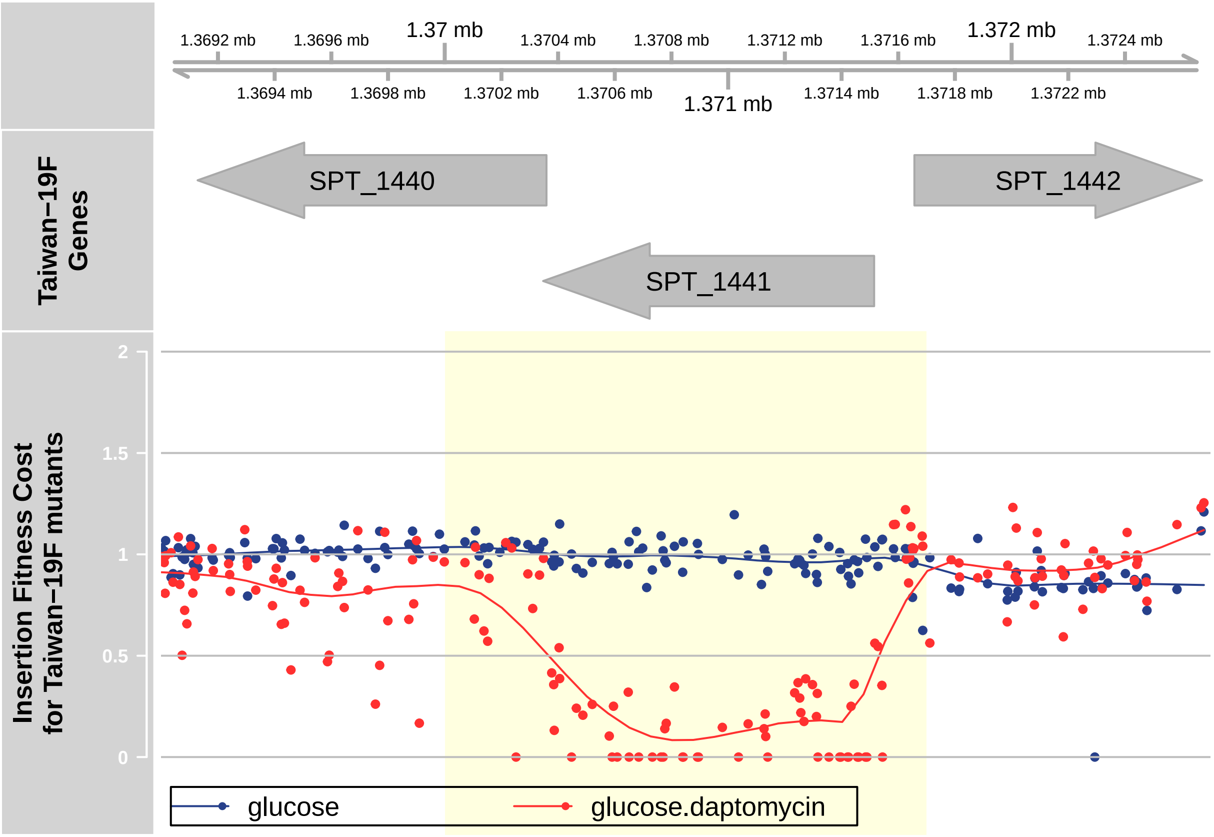
```
# MAGenTA input: A formatted file containing gene id, start, end coordinates, etc.
# from Genbank file using script getCoordsGBK.py (geneFile) adds gene ids

# Make sure coordinate columns 2 and 3 are numeric.  If not then reassign as numeric:
# column is.numeric(genes[,2])

geneDF <- as.data.frame(read.table(file = geneFile,
    header = TRUE))
anTrack <- AnnotationTrack(start = geneDF$start, end = geneDF$end,
    chromosome = "genome", strand = geneDF$strand,
    id = geneDF$id, showFeatureId = TRUE, name = "Taiwan-19F\nGenes",
    fill = "gray", fontcolor.item = "black")
```

Data Track

```
# MAGenTA input: single aggregate fitness per insertion site, from SingleFitness tool
makeJointAggData <- function(file1, file2) {
    # Adjust file format
    insertFit <- read.csv(file1)
    insertFit$seqnames = "genome"
    insertFit$start = insertFit$pos
    insertFit$end = insertFit$pos + 1
    keepCols <- c("seqnames", "start", "end", "fitness")
    insertFit <- insertFit[keepCols]
    colnames(insertFit) <- c("seqnames", "start", "end", "control")
    insertFit2 <- read.csv(file2)
    insertFit2$seqnames = "genome"
    insertFit2$start = insertFit2$pos
    insertFit2$end = insertFit2$pos + 1
    keepCols <- c("seqnames", "start", "end", "fitness")
    insertFit2 <- insertFit2[keepCols]
    merged <- merge(insertFit, insertFit2, by = c("seqnames",
        "start", "end"), all = TRUE)
    colnames(merged) <- c("seqnames", "start", "end", "glucose",
        "glucose.daptomycin")
    insertData <- makeGRangesFromDataFrame(merged, keep.extra.columns = TRUE,
        ignore.strand = TRUE, seqinfo = NULL, seqnames.field = "seqnames",
        start.field = "start", starts.in.df.are.0based = FALSE)
    return(insertData)
}
jointInsertData <- makeJointAggData(ctrlSingleFitFile, exptSingleFitFile)
count = length(jointInsertData)
trackTitle = "Insertion Fitness Cost\nfor Taiwan-19F mutants"
# Make track of insertion fitness. Histogram is used here, but other options are available.
# See Gviz documentation.
colors = c("royalblue4", "firebrick1")
aggTrack <- DataTrack(jointInsertData, chromosome = "genome",
    groups = c("glucose", "glucose.daptomycin"), ylim = c(0,
        2), type = c("p"), col.baseline = "gray", baseline = c(0,
        0.5, 1, 1.5, 2), col = c("royalblue4", "firebrick1"),
    legend = TRUE, name = trackTitle, fontcolor.legend = "black",
    fontsize.legend = 12, box.legend = TRUE)
```

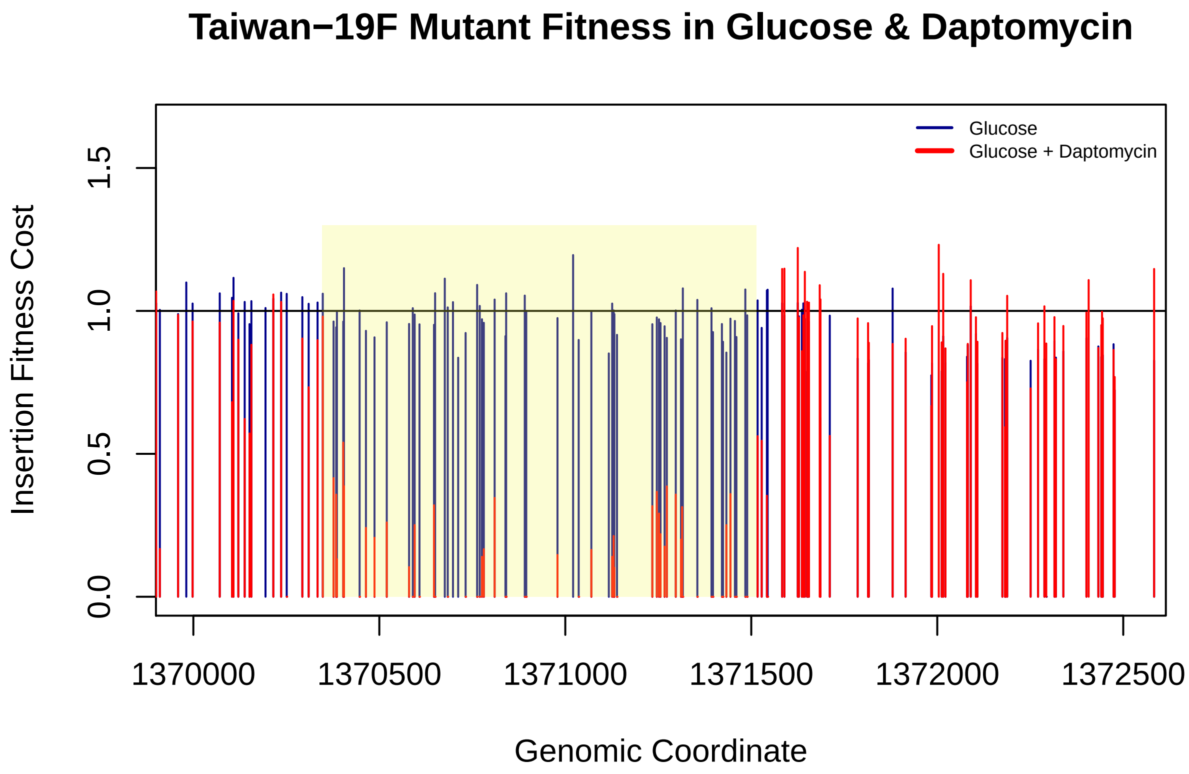
Plot Tracks

```
# Specify viewing window start and end coordinates
fromCoord = 1370000
toCoord = 1371700
# Change surrounding flanking region
x = 1000
y = 1000

# Highlight Track
ht <- HighlightTrack(trackList = list(aggTrack), fill = "lightyellow",
    col = "transparent", start = fromCoord, end = toCoord,  chromosome = "genome")

# Plot all Tracks
plotTracks(list(gTrack, anTrack, ht), chromosome = "genome",
    background.title = "lightgray", fontcolor = "black",
    fontsize = 10, from = fromCoord - x, to = toCoord +
        y, type = c("p", "smooth"), fontcol = "black",
    col.title = "black", axis.col = "black", fontface.main = 0.8,  fontfamily = "sans")
```



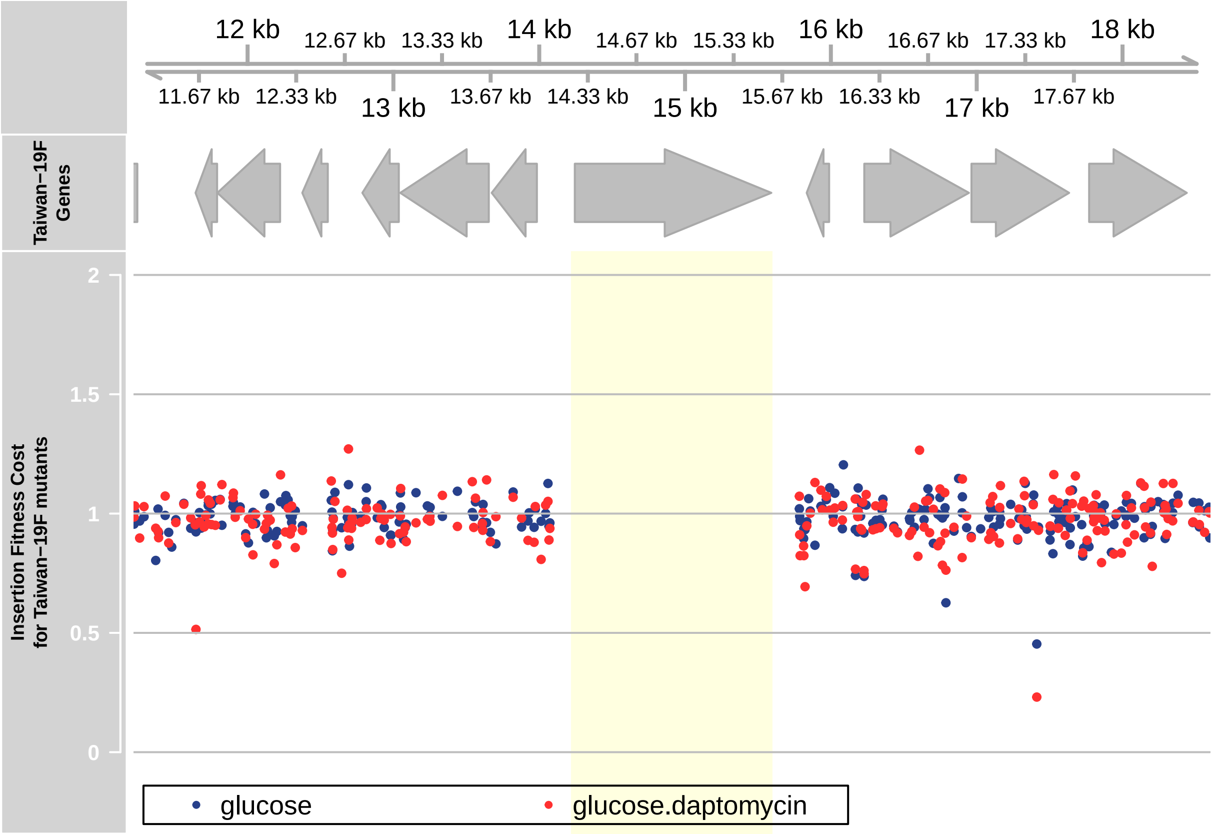
4.D.ii Single Track Visualization of Tn-Seq Data

```
mainTitle = "Taiwan-19F Mutant Fitness in Glucose & Daptomycin"

visSingleFit <- function(ctrlFile, exptFile, x1, x2, h1, h2) {
    ctrl <- read.csv(ctrlFile)
    expt <- read.csv(exptFile)
    plot(ctrl$pos, ctrl$fitness, xlab = "Genomic Coordinate",
        ylab = "Insertion Fitness Cost", main = mainTitle, type = "h",
        xlim = c(x1, x2), col = "darkblue")
    legend("topright", 1.35, c("Glucose", "Glucose + Daptomycin"),
        lty = c(1, 1), lwd = c(1.5, 2.5), col = c("darkblue",
            "red"), bty = "n", cex = 0.6)
    abline(h = 1, col = "black")
    lines(expt$pos, expt$fitness, col = "red", type = "h")
    rect(h1, 0, h2, 1.3, col = "#F4FA5840", border = NA)
}
visSingleFit(ctrlSingleFitFile, exptSingleFitFile, 1370000, 1372514,
    1370347, 1371514)
```



4.D.iii Other Examples of Tn-Seq Data

Region with underrepresented number of insertions


12 kb
12.67 kb
13.33 kb
14 kb
14.67 kb
15.33 kb
16 kb
16.67 kb
17.33 kb
18 kb
11.67 kb
12.33 kb
13 kb
13.67 kb
14.33 kb
15 kb
15.67 kb
16.33 kb
17 kb
17.67 kb
Taiwan−19F Genes
Insertion Fitness Cost for Taiwan−19F mutants
2
1.5
1
0.5
0
● glucose
● glucose.daptomycin