

# MAGenTA: Microbial Assessment by Genome-wide Tn-Seq Analysis

*Margaret Antonio, Katherine McCoy, Tim van Opijnen, PhD*

*25 January 2017*

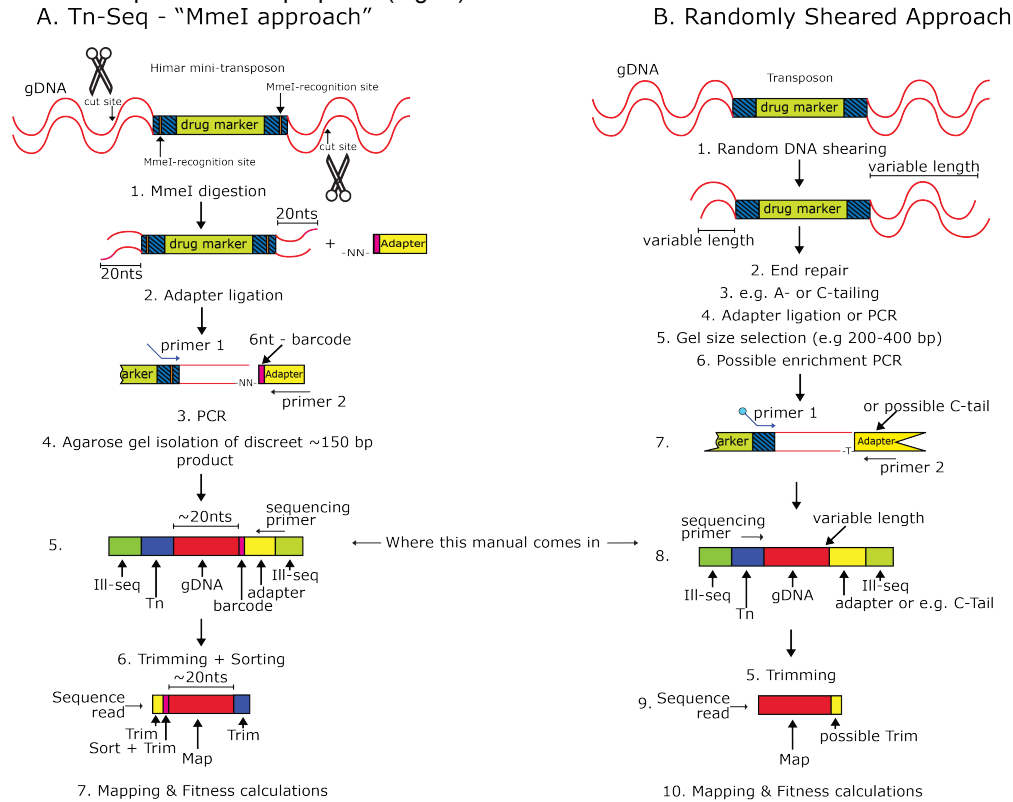
## Contents

<b>Introduction</b>	<b>2</b>
An Overview of Tn-Seq . . . . .	2
Workflow . . . . .	2
<b>Part A: Data Prep</b>	<b>4</b>
Input Files . . . . .	4
Galaxy Toolshed . . . . .	6
Mme1 Tn-Seq Prep . . . . .	6
Randomly Sheared Tn-Seq Prep . . . . .	8
<b>Part B: Fitness Analysis</b>	<b>10</b>
Calculate fitness effect of single mutations and over annotated genes . . . . .	10
RegionFitness: a non-gene-centric method for discovering regions important for organismal fitness . . . . .	10
<b>Part C: Comparative Analyses</b>	<b>12</b>
DataOverview: Characterizing the reference genome and genome-wide insertion data . . . . .	12
CompareGenes: Comparative gene analysis for the same organism . . . . .	14
CompareStrain: Comparative gene analysis for different organisms . . . . .	14
CompareRegions: Comparative analysis of small, unannotated regions . . . . .	14
SingleFitness: aggregate fitness information across multiple libraries for one weighted fitness value per insertion mutant . . . . .	15
<b>Performing Data pre-processing and MAGenTA analyses on the Command-line</b>	<b>15</b>
Data Prep: Fastx Toolkit and Bowtie . . . . .	15
Downloading command-line tools . . . . .	15
Script Usage . . . . .	15
<b>Data Visualization</b>	<b>21</b>
Multi-track Visualization of Tn-Seq data with Gviz . . . . .	21
Single Track Visualization of Tn-Seq Data . . . . .	25
Other Examples of Tn-Seq Data . . . . .	26

# Introduction

## An Overview of Tn-Seq

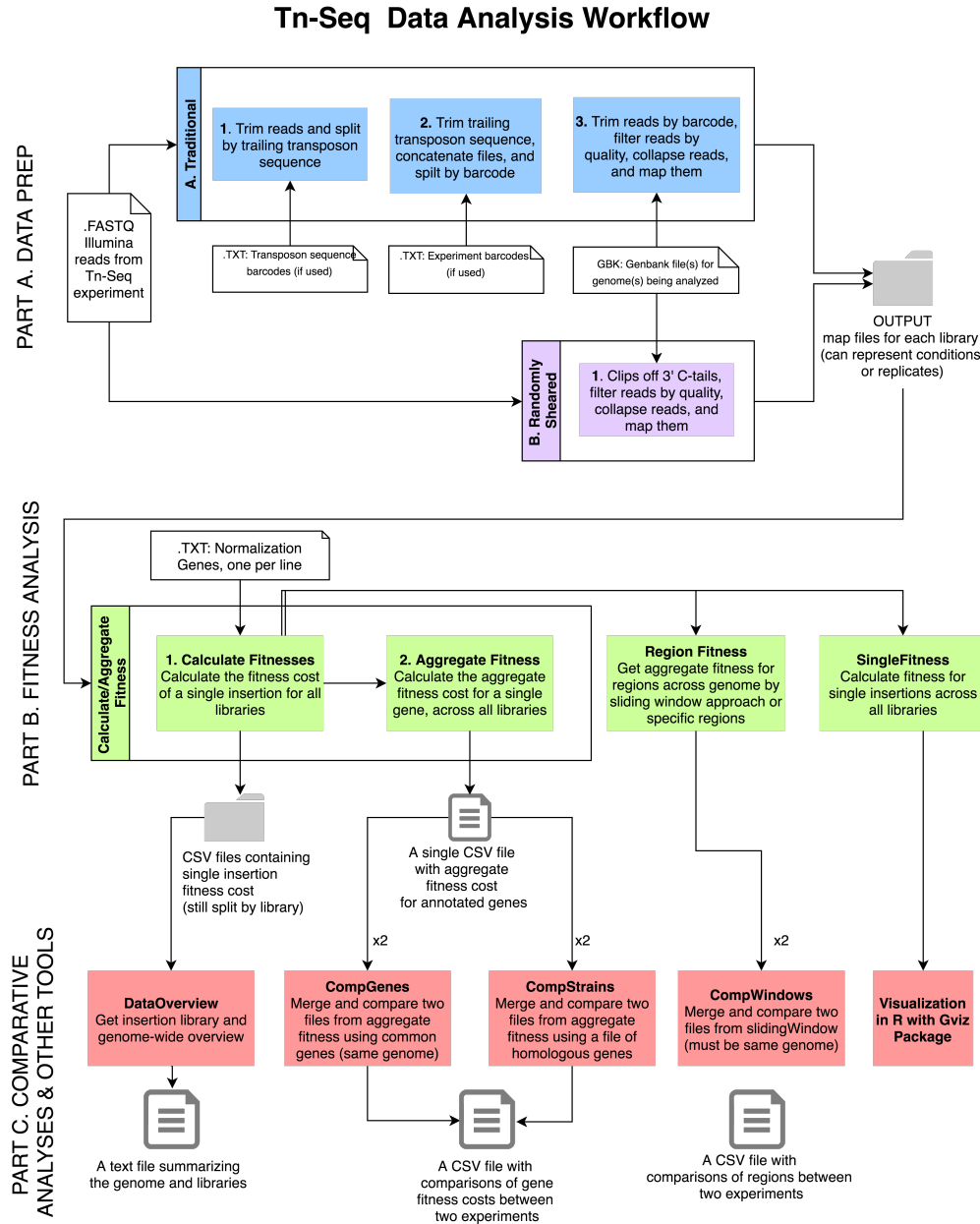
Tn-Seq is a method of determining quantitative fitnesses of bacterial genes. This manual is not meant to describe it in full, only how to analyze the sequencing data it produces. For more information on Tn-Seq, several papers [1](#), [2](#), [3](#), [4](#) have been published with detailed explanations. To begin, the tools and steps used to process the sequence data depend on how the sequences were prepared (Fig. 1).



**Figure 1. Two approaches to Tn-Seq sample preparation.** (A) The [MmeI Tn-Seq procedure](#) uses MmeI digestion to generate a PCR product of approximately 150bp, with two flanking sequences necessary for Illumina 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 six nucleotide barcode to enable multiplexing (in pink). Note that in this approach sequencing starts from the sequencing primer on the right then reads into the barcode, the gDNA, and finally the transposon. (B) Alternatively, gDNA is randomly sheared so that the product to be sequenced has a variable length. It is unclear how large the gDNA fragment is; sequencing 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.

## Workflow

Following sequencing of reads from either approach in Figure 1, the reads are processed, fitness costs of single mutations are calculated, and comparative analyses can be performed (Fig. 2).



**Figure 2. An overview of the main steps of Tn-Seq Data Pre-processing and MAGenTA Analyses.** In Part A: “Data Prep”, all non-genomic parts of the sequences are removed and the remaining sequence is mapped to the reference genome. In Part B: “Fitness Analysis”, the differences in mapped reads between an initial time point and a second time point are used to calculate the fitness of single insertions, which can then be aggregated over regions in a gene-centric or non-gene-centric analysis. Finally, Part C: “Comparative Analyses & Other Tools” includes optional analyses for comparing and visualizing fitness data.

## Input Files

1. **Sequencing reads** in **FASTQ format**. These files should end in .fastq and look like [this](#):

2. **Expansion factors**, experimentally determined integers representing expansion of the bacterial populations under study. One method of determining the expansion factor is to divide the number of bacteria at the end of an experiment by the number at the beginning of the experiment. In vitro, assessing bacteria population size is conventionally performed by counting colony forming units (CFUs) in serial dilutions. A method for in vivo determination of expansion factor is discussed [here](#).
3. **Barcodes**, listed in a text (.txt) file, formatted as follows: one barcode per line with the barcode name and sequence tab-delimited. Barcodes are often only used to differentiate between different libraries or experimental conditions. This file can be created in any plain text editor, with one barcode per line: an arbitrary barcode name, a tab, and the barcode sequence. It is recommended that the barcode names be descriptive, including library number, expansion factor, and experimental condition for better identification. A sample barcode.txt file:

4. **Additional barcodes for the Mmel approach.** If using the Mmel Tn-Seq approach, four additional barcode files which describe the end of the Mmel transposon sequence are required. As shown below, there is a variable region of adaptor DNA in each read, which is identified in the barcode for trimming. As in the previous barcode file, this should be in tab-delimited format, ending in .txt, with one barcode per line, per file. An example of four such barcode files are shown [here](#). Each of the barcode files contain one of the following lines:

4

Note that even though the Galaxy and Command-line tools we describe in this manual can open and read these files, typically if one just clicks the file to open it, the computer cannot automatically detect which program to use to open fasta and genbank files. To view the file before using it, simply change the file extension from “.gbk” or “.fasta” to “.txt”. This way a regular text editor will open it.

**An example of the first five lines and a few lines from the first annotated gene of a typical genbank file are shown:**

```
LOCUS      NC_012469                2112148 bp    DNA      circular CON 10-JUN-2013
DEFINITION Streptococcus pneumoniae Taiwan19F-14 chromosome, complete genome.
ACCESSION  NC_012469
VERSION    NC_012469.1  GI:225860012
DBLINK     Project: 59119
```

...

```
source      1..2112148
            /organism="Streptococcus pneumoniae Taiwan19F-14"
            /mol_type="genomic DNA"
            /strain="Taiwan19F-14"
gene        197..1558
            /gene="dnaA"
            /locus_tag="SPT_0001"
            /db_xref="GeneID:7686582"
CDS         197..1558
            /gene="dnaA"
            /locus_tag="SPT_0001"
            /note="binds to the dnaA-box as an ATP-bound complex"
            /codon_start=1
            /transl_table=11
            /product="chromosomal replication initiation protein"
            /protein_id="YP_002741522.1"
            /db_xref="GI:225860013"
            /db_xref="GeneID:7686582"
            /translation="MKEKQFWNRILEFAQERLTRSMYDFYAIQAELIKVEENVATIFL
PRSEMEMVWEKQLKDIIVVAGFEIYDAEITPHYIFTKPQDTTSSQVEEATNLTLYDYS
PKLVSIPYSDTGLKEKYTFDNFIQGDGNVWAVSAALAVSEDLALTYNPLFIYGGPGLG"
```

**An example of the beginning of the beginning of the *Streptococcus pneumoniae* R6 genome in FASTA format is as follows:**

```
>NC_003098.1 Streptococcus pneumoniae R6 chromosome, complete genome
TTGAAAGAAAAACAATTTTGAATCGTATATTAGAATTTGCACAAGAAAGACTGACTCGATCCATGTATGATTTCTATGC
TATTCAAGCTGAAGTTATCAAGGTAGAGGAAAATGTTGCCACTATATTTCTACCTCGCTCTGAAATGGAAATGGTCTGGG
AAAAACAATAAAAGATATTATTGTAGTAGCTGGTTTTGAAATTTATGACGCTGAAATAACTCCCCACTATATTTTCACC
```

6. **List of normalization genes** for the organism under study. Normalization genes are genes that do not affect organismal fitness when disrupted. Such genes are generally transposon genes, pseudogenes, and degenerate genes. Although a normalization gene file is not necessary for running Tn-Seq analysis tools, if the normalization genes are known, they should be listed, one per line in a text file, ending in .txt. For example, a portion of the

normalization file for *Streptococcus pneumoniae* TIGR4 is shown as follows, with genes listed by their gene id:

```
SP_0130
SP_0131
SP_0362
SP_0364
SP_0365
```

## Galaxy Toolshed

The [Galaxy Project](#) is a web-based platform for biomedical research tools. A tutorial on how to set up a Galaxy environment can be found [here](#), how to upload files is described [here](#), and our custom tools can all be found and installed through the [Galaxy toolshed](#). Currently the custom tools in this pipeline that are available in the Galaxy Toolshed are:

- enhanced\_bowtie\_mapper
- fastq\_collapser
- calculate\_fitnesses
- aggregate\_fitnesses

As described in Figure 1, there are three main parts to the analysis. Part 1: Data Prep varies depending on the type of input data. If experimental samples were prepared using the Mmel Tn-Seq method, then the corresponding data preparation procedure should be followed. An alternative processing workflow is described for the random shearing method. In these two cases and if samples were prepared through another method, the general outline of the workflow is as follows and can be customized depending on the method used. Create a custom workflow using Galaxy's [workflow editor](#):

1. Split data by experimental condition barcodes, if any, using the fastx barcode splitter.
2. Remove all non-genomic DNA using a tool like the fastq trimmer.
3. Filter reads by quality using the fastq quality splitter
4. Map reads in Bowtie. 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.

All output results of Part 1: Data Prep can be used in Part 2 and Part 3 regardless of whether the Mmel or Random Shearing methods were used.

## Mmel Tn-Seq Prep

Download Galaxy Workflows: [Part 1](#), [Part 2](#), and [Part 3](#)

This prep procedure assumes a 50nt read, as described in Fig. 1 A), with its first 9 nucleotides from an adaptor, the next six from its barcode, the next 15 to 18 from genomic DNA, and the last 20 to 23 from its transposon. In Part 1 it trims the first 9 and last 17, 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 genome provided.

## Part 1: Trim & split by trailing transposon sequence

1. Navigate to the workflow tab, click on the “Standard Tn-Seq Prep Pt.1” workflow, and select “Run”. A workflow should appear with the following steps:

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

[Expand All](#)[Collapse](#)

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

Run workflow

2. Select the FASTQ file under “this dataset” in “Step 1: Trim”. The default settings will trim of the first 9 and the last 17nts.
3. Select the four transposon sequence barcode files in Steps 2 through 5, one per step. Note that these transposon-endings are entered as “barcodes” in the workflow, but they are not the experimental barcodes, just transposon ends the tool searches for like barcodes. They can go in any order, 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.
4. Press “Run workflow”, and wait for the workflow to finish running.

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

1. Navigate to the workflow tab, click on the “Standard Tn-Seq Prep Pt.2” workflow, and select “Run”. A workflow should appear with the following 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)

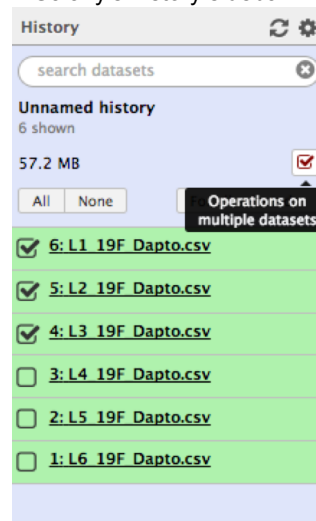
Step 6: Barcode Splitter (version 1.1)

☐ Send results to a new history

Run workflow

2. Select the trim4 barcode-split file under “this dataset” for “Step 1: Trim” (the step with “-4” under “Remove everything from this position to the end”).
3. Select the trim3 barcode-split file under “this dataset” for “Step 2: Trim” (the step with “-3” under “Remove everything from this position to the end”).
4. Select the trim2 barcode-split file under “this dataset” for “Step 3: Trim” (the step with “-1” under “Remove everything from this position to the end”).

5. Select the trim1 barcode-split file under “this dataset” for “Step 4: Trim” (the step with “-2” under “Remove everything from this position to the end”).
6. Select the barcode file under “Barcodes to use” in “Step 6: Barcode Splitter”. These should be the barcodes identifying which experimental condition each read is from.
7. Press “Run workflow”, and wait for the workflow to finish running.
8. Refresh the history and make a list out of the resulting files to send into the next workflow, via the “operations on multiple datasets” button located in Galaxy’s history sidebar.



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

1. Navigate to the workflow tab, click on the “Standard Tn-Seq Prep Pt.3” workflow, and select “Run”. A workflow should appear with the following steps:



2. Under “Step 5: Map with Bowtie for Illumina”, select the fasta genome under “Select the reference genome”.
3. Press “Run workflow”, and wait for the workflow to finish running.

### Randomly Sheared Tn-Seq Prep

#### Download Galaxy Workflow

As described in Fig. 1 B, another method to prepare a Tn-Seq sample involves DNA shearing and ligating on C-tails. This prep assumes a 50nt read of genomic DNA followed by a C-tail, of variable length due to the randomness of the shearing. It removes the C-tail from the end with Cutadapt, grooms the reads, filters them for quality, collapses them,



and maps them. If the reads were prepared in this manner but also had barcodes, the barcode splitter tool should be used to separate them by barcode first.

1. Use the “operations on multiple datasets” button (located in Galaxy’s history sidebar) to make a list out of all the sequence files to be analyzed.
2. Navigate to the workflow tab, click on the “Randomly Sheared Data Prep” workflow, and select “Run”. A workflow should appear with the following steps:

#### Running workflow "Randomly Sheared Data Prep"

[Expand All](#)[Collapse](#)

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

Run workflow

3. Enter the list of files under “Step 1: Input dataset collection”
4. At “Step 6: Map with Bowtie for Illumina”, select the fasta genome under “Select the reference genome”. Make sure this is the [fasta](#) file not the [genbank](#) file.
5. Press “Run workflow”, and wait for the workflow to finish running

## Part B: Fitness Analysis

### Calculate fitness effect of single mutations and over annotated genes

#### [Download Fitness Workflow](#)

This workflow first calculates fitnesses for each insertion in the genome, then aggregates them by gene. The aggregate tool can also be used on multiple Calculate Fitness output files to make a 'super' aggregate file, which calculates fitness of genes by averaging fitness of all insertions across multiple libraries. Note that this workflow cannot be run in batch, because each run requires a unique expansion read, so repeat the below steps for each time 2 condition to be analyzed.

1. Navigate to the workflow tab, click on the "Calculate / Aggregate Fitness" workflow, and select "Run". A workflow should appear with the following steps:



Step 1: Calculate Fitness (version 1.0.0)

Step 2: Aggregate (version 1.0.0)

☐ Send results to a new history

Run workflow

2. In "Step 1: Calculate Fitness", select the map files from t1, map files from t2, GenBank reference genome, and normalization genes under the corresponding headers.
3. In "Step 1: Calculate Fitness", click the edit symbol under "Expansion Factor" and edit that value with the time 2 sample's expansion factor, in that same step.
4. In "Step 2: Aggregate", Select the GenBank reference genome under the corresponding header.
5. In "Step 2: Aggregate", optionally specify whether to mark certain genes (for example, normalization genes, for reference). Choose "Yes" under "Mark certain genes?". Then choose the normalization genes file or a similarly formatted file as input.
6. Press "Run workflow", and wait for the workflow to finish running.
7. The output data is from different libraries under the same condition, but can be aggregated across gene regions (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 a custom blank count?" as shown below. For the custom blank count, simply average the blank counts of each fitness file, which can be found in its corresponding calc fitness txt file output. Then enter all the fitness files from the same condition, and press "Execute"!

### RegionFitness: 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. At the same time, it has been shown that non-coding RNAs have an important role in virulence and other functions. Therefore, to enable discovery of such regions that may have an important fitness value to the organism, 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 inter-genic 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 can help determine these factors.

**Method** In each regional profile, the aggregate fitness cost for insertions and the insertion representation in that region is calculated. To assess insertion representation, we follow the method used by Zhang et. al.14. 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. In the future, a function to identify promoter regions based on -10 and -35 box sequences and domain regions will be useful.

SlidingWindow performs the fitness and insertion representation assessments in approximately 8 minutes for a two-thousand base pair genome with ~70% TA site saturation. For the Tn-Seq data from two strains of *Streptococcus pneumoniae* grown in two conditions, 211,151 regions (300 bp) across the 2,112,148 bp genome were profiled using this approach and later used for comparative analyses.

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

CSV Fitness File(s)

No data dataset available.

Additional csv fitness file(s)

Insert 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

Execute

## Part C: Comparative Analyses

**Table 1. A summary of analysis tools in MAGenTA.**

Tool	Function	Research Questions Addressed
DataOverview	Produces a summary file of information about TA sites and insertions in the genome, libraries, and gene/intergenic regions	What is the TA and insertion distribution in the genome for genes and non-gene regions?
RegionFitness	Scans the genome data with set window size and step to perform aggregate fitness and essentiality assessments at each window.	Which unannotated (intergenic or gene domain) regions are important or essential?
CustomWindow	Given a list of start and end coordinates, will return aggregate fitness and essentiality test results. Useful for recalculating info for merged regions	What is the importance or essentiality of specific regions of interest?
CompareGenes	Creates a gene comparison file with significant stats for two Tn-Seq experiments of the same ref genome. Need aggregate output	How can we compare genes for a given strain in two different conditions? Which genes are conditionally important?
CompareStrains	Creates a gene comparison file with significant stats for two Tn-Seq experiments of different strains (genomes). Need conversion file and aggregate output.	How can we compare homologous genes for two strains? Which genes have a strain-dependent importance?
CompareRegions	Creates a comparison file with significance statistics for two Tn-Seq experiments of different strains (genomes). Need sliding window output	How can we compare intergenic or gene domain regions for the same strain in two conditions?

### **DataOverview: Characterizing the reference genome and genome-wide insertion data**

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. The input data are queried for TA sites and insertions with respect to

1. The genome
2. Open reading frames denoted by start and stop codons
3. Annotated genes from the Genbank record
4. Intergenic regions

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

csv fitness file

No data dataset available.

Additional csv fitness file(s)

+

Insert Additional csv fitness file(s)

Fasta file

No fasta dataset available.

GenBank reference genome

No data dataset available.

Cutoff

10

Weight ceiling

50

✓ Execute

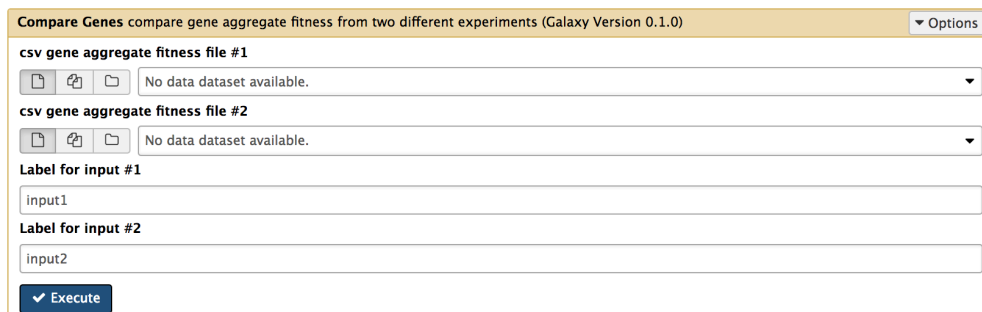
## Sample Output

**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%

## CompareGenes: Comparative gene analysis for the same organism

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, GeneCompare can perform it in less than five seconds for 2000 gene.



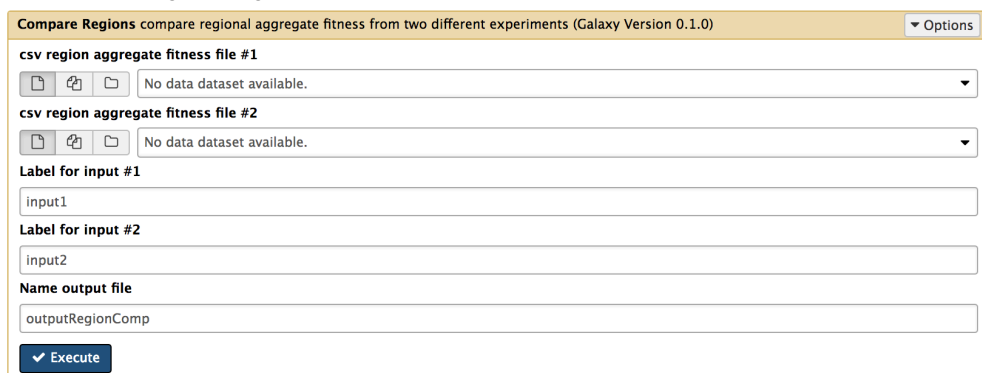
## CompareStrain: Comparative gene analysis for different organisms

The CompareStrain 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.

## CompareRegions: Comparative analysis of small, unannotated regions

CompareRegions takes results from the RegionFitness tool and performs a similar comparative analysis as done on genes by GeneCompare and StrainCompare for genes. Since start and end coordinates define windows (or regions), only comparisons between datasets from the same organism (same genome) can be performed.

In the example shown, over 200,000 300-bp windows for Tn-Seq data from Taiwan-19F grown in glucose and the presence of daptomycin are assessed for conditionally important regions using the adjusted p-value (FDR) and fitness difference between the two conditions. As shown in the yellow highlighted area, 583 regions met the criteria for having adjusted p-values below a critical value of 0.01 and a fitness difference greater than 0.1. Distributions of regions in 20 unique genes and two intergenic regions are shown in the table.



**SingleFitness: aggregate fitness information across multiple libraries for one weighted fitness value per insertion mutant**

## Performing Data pre-processing and MAGenTA analyses on the Command-line

### Data Prep: Fastx Toolkit and Bowtie

1. Download and install the [FASTX Toolkit](#) and [Bowtie](#). For more on the usage of the fastx\_toolkit see its [manual](#), and for more on Bowtie see its [manual](#).
2. 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 the data.
3. Filter reads by quality (e.g. a minimum quality of 8) using fastq quality filter.
4. Collapse reads using fastq collapser.
5. Map reads using Bowtie. 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)

### Downloading command-line tools

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

```
git clone https://github.com/vanOpijnenLab/magenta-p2.git
```

Or [download the MAGenTA project repository](#)

### Script Usage

**CalculateFitness: calculate fitness of single insertion mutants using read counts at two time points**

#### Usage

```
python calc_fitness.py <options>
```

#### Required Inputs

Flag	Description
ref	The name of the reference genome file, in GenBank format

Flag	Description
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)
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)
maxweight	The maximum weight a transposon 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 because may not affect gene function.
el	Exclude insertions in the last N amount (%) of the gene—considering truncation may not affect gene function.
wig	Create a wiggle file for viewing in a genome browser. Provide a filename.
uncol	Use if reads were uncollapsed when mapped.

### AggregateFitness: calculate aggregate fitness cost over genes

#### Usage

```
python aggregate.py -o <output file> <options> <inputs or -d directory>
```

#### Required Inputs

Flag	Description
o	Output file for aggregated data

#### 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, which will be discounted for all genes (for example, 20% would be entered as 0.20; default 0.0)
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)

## DataOverview

### Usage

```
perl dataOverview.pl <otptions> <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)
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

## GeneCompare: Comparative gene analysis for the same organism

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, GeneCompare can perform it in less than five seconds for 2000 gene.

### 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.

### StrainCompare: Comparative gene analysis for different organisms

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

Flag	Description
l	Labels for input files. Two strings, comma separated (i.e. -l expt1,expt2). Order should match file order. Default: input filenames.

## Region Fitness

### Usage

```
perl slidingWindow.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)
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
wc	Integer value for weight ceiling. Default=50

## CompareRegions: Comparative analysis of small, unannotated regions

### Usage

```
perl compWindows.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

**SingleFitness: single Fitness or insertion count per insertion as for visualization input (IGV or Gviz)**

### Usage

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

### Required Inputs

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

### Optional Inputs

Flag	Description
h	Print usage and exits program
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.
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

## Data Visualization

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 as follows.

### 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

```
# To facilitate downstream function usage,  
# specifying file paths in the beginning of a  
# session  
  
# Set working directory  
# setwd('~Documents/lab/tvolab/magenta/')  
  
# Specify input files  
  
# Fasta file for the reference genome, available on  
# NCBI genomes  
genomeFile = "~/Documents/lab/tvolab/magenta/genome/19f/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 = "~/Documents/lab/tvolab/magenta/genome/19f/19F_012469_genes.txt"

# Single fitness files are created by the SingleFit
# tool (Galaxy or command-line) using input library
# csv files from the CalculateFitness tools Here, a
# control (glucose) file and experimental (glucose
# + daptomycin) are used
ctrlSingleFitFile = "~/Documents/lab/tvolab/magenta/data/dapto-19ft4/19fglucSingleFit.csv"
exptSingleFitFile = "~/Documents/lab/tvolab/magenta/data/dapto-19ft4/19fdaptoSingleFit.csv"

```

### Global Gviz parameters

```

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

```

### Gviz Tracks

#### 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 or other annotation information

# Make sure coordinate columns 2 and 3 are numeric.
# If not then reassign as numerically converted
# 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 Note: Here histogram is
# used, 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)

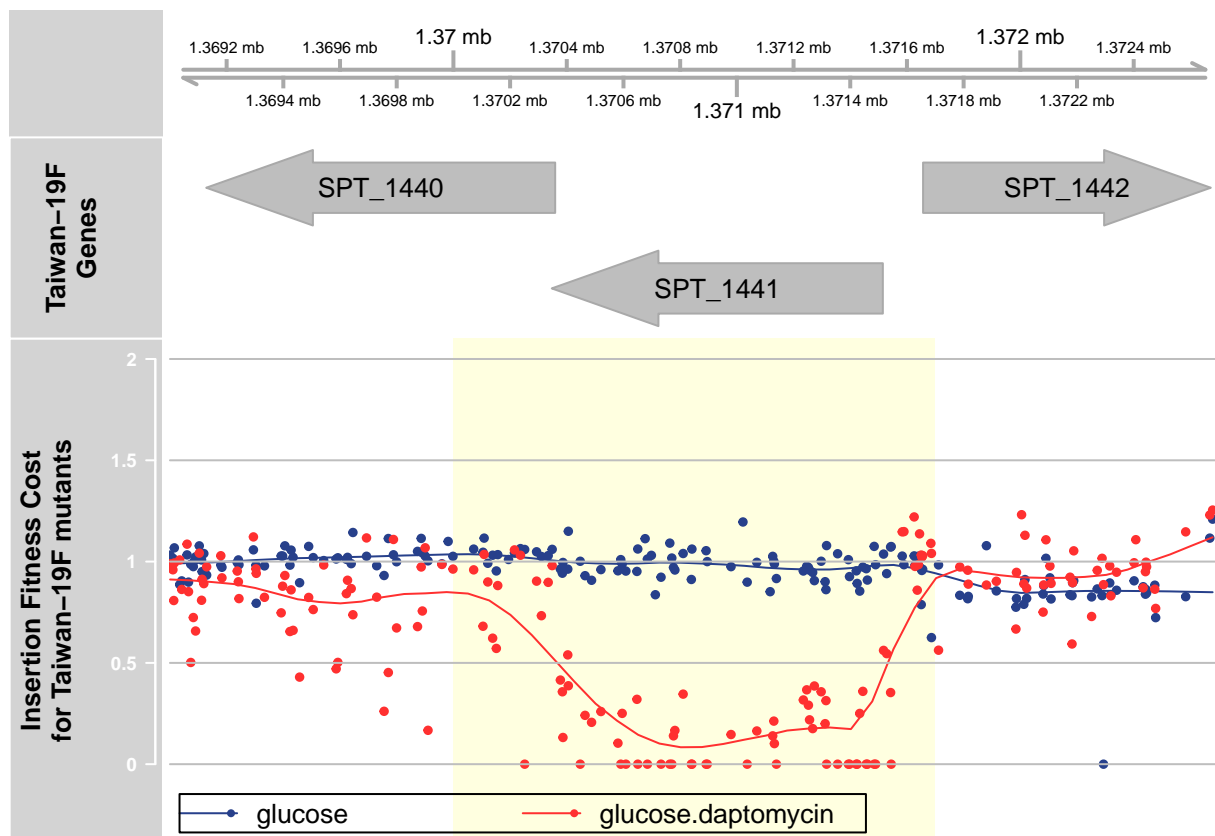
```

## 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")
```





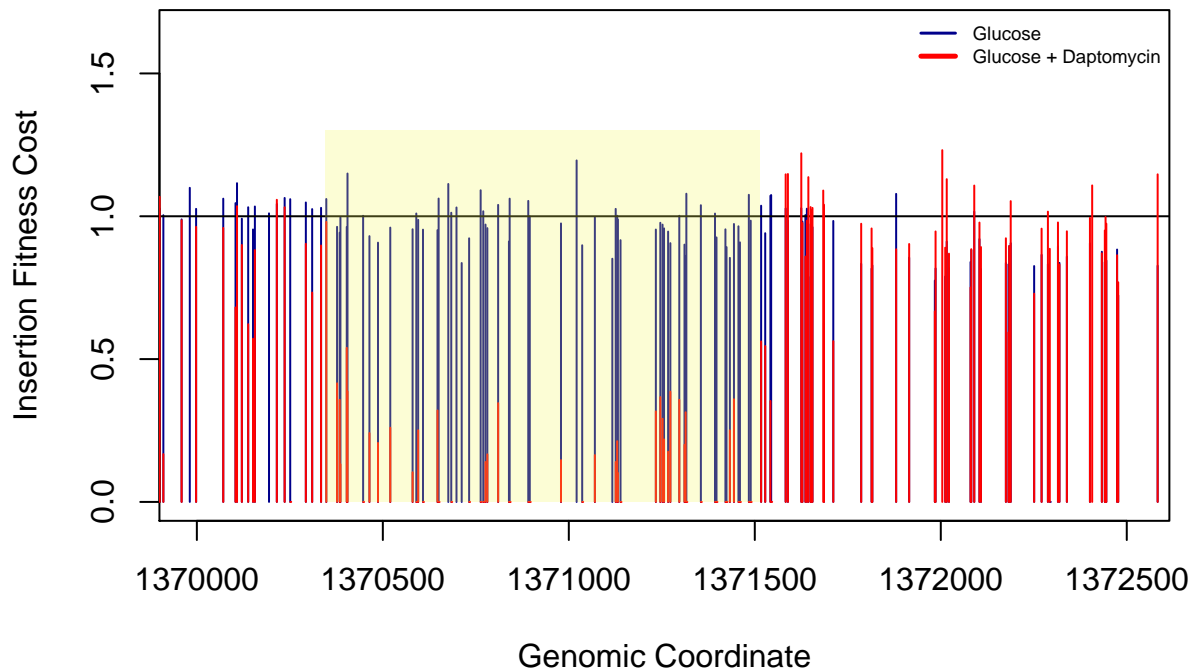
## Single Track Visualization of Tn-Seq Data

```
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)
```

### Taiwan-19F Mutant Fitness in Glucose & Daptomycin



## Other Examples of Tn-Seq Data

### Region with underrepresented number of insertions

