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

Margaret Antonio, Katherine McCoy, Tim van Opijnen, PhD

22 January 2017

Contents

1 Introduction

1.1 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. If you are unfamiliar with Tn-Seq or require a refresher, several scientific papers (1, 2, 3, 4) have been published with detailed explanations.

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

includegraphics[scale=0.87]SeqProdOverview.png

Figure 1. An overview of the two main approaches to Tn-Seq sample preparation, which both end in products able to be sequenced on an Illumina sequencer. A.) Shows the different steps in the MmeI Tn-Seq procedure, which uses MmeI digestion. It generates a PCR product of approximately 150bp, with two flanking sequences essential to enable 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 6nt 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.) Shows the randomly sheared approach where gDNA is randomly sheared, so that the product to be sequenced has a variable length. This means it is not clear 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.

After the reads described in Figure 1 are sequenced, they must be computationally analyzed (Fig 2). The first section of analysis is the "Data Prep", in which all non-genomic parts of the sequences are removed and the genomic data is mapped to discover the location of each read. The second section is the "Fitness Analysis", in which the differences in mapped genomic data between an initial time point and a second time point are used to calculate the fitness of various insertions then aggregate them by gene (If you're unfamiliar with the concept of a fitness calculation, see the papers linked at the start). Finally, the third section is "Comparative Analyses & Other Tools" - these optional analyses / tools let you compare and visualize your fitness data in various useful ways.

1.2 Workflow

Figure 2. An overview of the various steps of Tn-Seq Data Analysis. Data is trimmed of its non-genomic segments and mapped, has its genomic fitness calculated, and finally can be examined by an array of useful tools.

2 Part 1: Data Prep

2.1 The Files You'll Need

The data you'll need are as follow:

1. Sequencing reads in FASTQ format. These files should end in .fastq and look like this:

includegraphics[scale=0.5]fastq.jpg

2. Expansion factors, which are determined experimentally. You divide the number of bacteria you end with by the number you started with; to calculate this in vitro you could plate appropriate dilutions of your starting and cultures then measure their number of Colony Forming Units (CFUs), for instance. In vivo is a bit more complicated, and a reasonable estimation often works as well, but see this link for how you might experimentally determine in vivo growth rates and expansion factors.
3. Barcode files, if you used any to differentiate between your different conditions. Make these by hand; give each barcode a line which consists of whatever you'd like th name it, a tab, and its sequence. We strongly recommend putting your expansion factors in their relevant barcode names, for easy identification. These files should end in .txt and look like this:

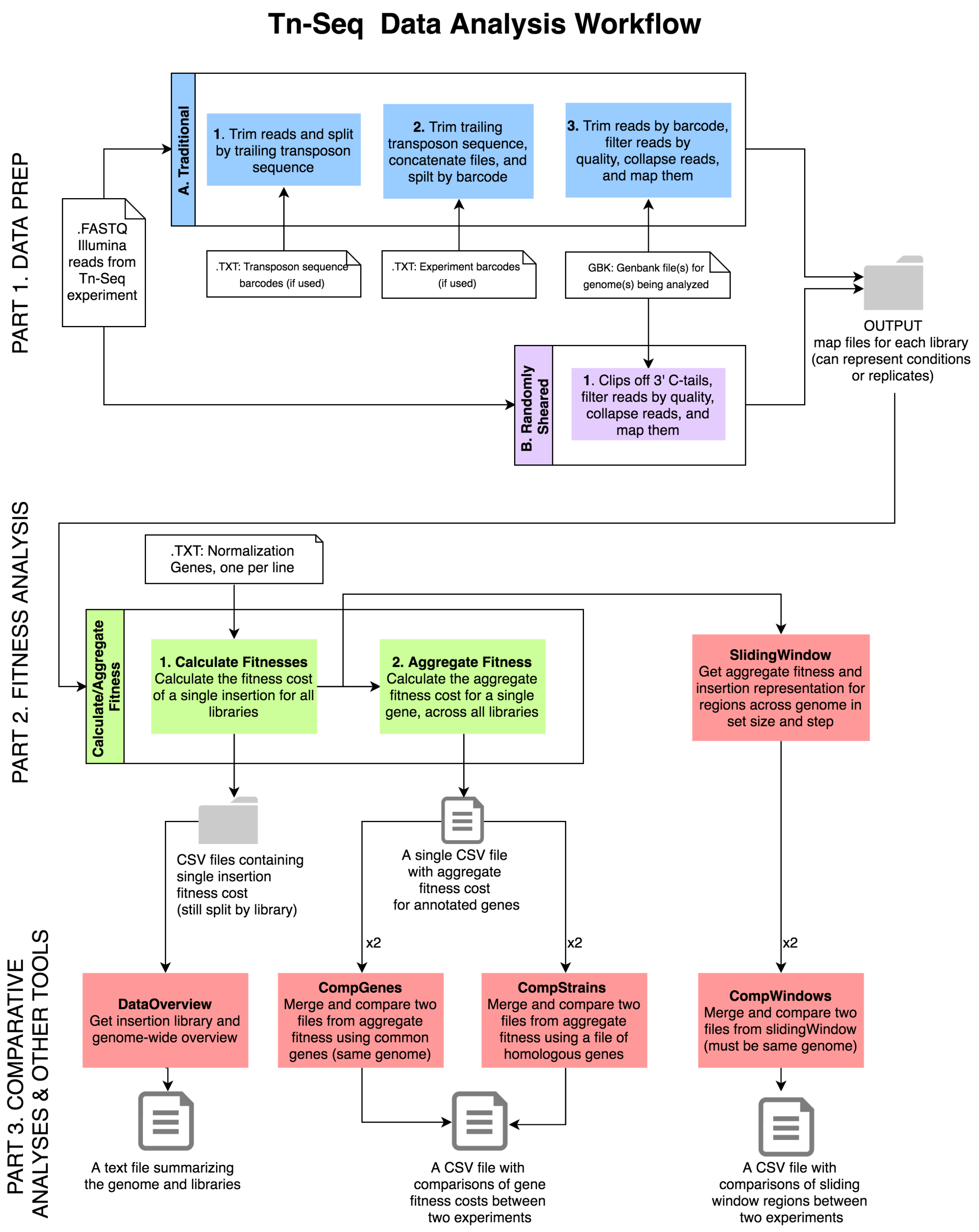
includegraphics[scale=0.5]barcode.png

4. If you are using the MmeI Tn-Seq approach, four additional barcode files which describe the end of the MmeI transposon sequence. (As shown in Fig 1, there is a variable region of adaptor DNA in each read, and these barcodes allow you to identify then trim it off.) They should end in .txt, have one line each, and are exactly as follows:

includegraphics[scale=0.5]transbc1.png

includegraphics[scale=0.5]transbc2.png

includegraphics[scale=0.5]transbc3.png



Tn-Seq Data Analysis Workflow
PART 1. DATA PREP
A. Traditional
1. Trim reads and split by trailing transposon sequence
2. Trim trailing transposon sequence, concatenate files, and spilt by barcode
3. Trim reads by barcode, filter reads by quality, collapse reads, and map them
.FASTQ Illumina reads from Tn-Seq experiment
.TXT: Transposon sequence barcodes (if used)
.TXT: Experiment barcodes (if used)
GBK: Genbank file(s) for genome(s) being analyzed
B. Randomly Sheared
1. 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 2. FITNESS ANALYSIS
.TXT: Normalization Genes, one per line
Calculate/Aggregate Fitness
1. Calculate Fitnesses Calculate the fitness cost of a single insertion for all libraries
2. Aggregate Fitness Calculate the aggregate fitness cost for a single gene, across all libraries
SlidingWindow Get aggregate fitness and insertion representation for regions across genome in set size and step
CSV files containing single insertion fitness cost (still split by library)
A single CSV file with aggregate fitness cost for annotated genes
x2
x2
x2
PART 3. COMPARATIVE ANALYSES & OTHER TOOLS
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
CompWindows 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 sliding window regions between two experiments

Figure 1: MAGenTA Workflow

includegraphics[scale=0.5]transbc4.png

5. Genbank and FASTQ files for each strain being analyzed. Both can be found on NCBI by searching for each strain under the Genome section. These files should end in .gbk and .fasta respectively, and look like this:

includegraphics[scale=0.5]gbk.png

includegraphics[scale=0.5]fasta.png

6. 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.

includegraphics[scale=0.5]normalization.png

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

2.2 Galaxy Toolshed

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 can all be found and installed through the Galaxy toolshed. You can find them by searching for their names in the toolshed:

- enhanced_bowtie_mapper
- fastq_collapser
- calculate_fitnesses
- aggregate_fitnesses

As described in Figure 1, there are three 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:

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, if you like.
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.

After the data prep, everything else is done in the same way. The fitness analysis is done with the Calculate / Aggregate Fitness Workflow, and the Comparative Analyses & Other Tools can then be used to sort through and visualize your fitness data.

2.3 MmeI 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, concatinates 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.

2.3.1 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". You should see a workflow with these steps:

includegraphics[scale=0.5]StandPrep1.png

2. Select your FASTQ file under "this dataset" in "Step 1: Trim". The default settings will trim of the first 9 and the last 17nts.
3. Select your 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 your experimental barcodes, just transposon ends the tool searches for like barcodes. 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.
4. Press "Run workflow", and wait for the workflow to finish running.

2.3.2 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". You should see a workflow with these steps:

includegraphics[scale=0.5]StandPrep2.png

2. Select your 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 your 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 your 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 your 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 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.
6. Press "Run workflow", and wait for the workflow to finish running.
7. 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:

includegraphics[scale=1.0]opbutton.png

2.3.3 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". You should see a workflow with these steps:

includegraphics[scale=0.5]StandPrep3.png

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

2.4 Randomly Sheared Tn-Seq Prep

Download Galaxy Workflow

As described in Fig. 1 B, another method to prepare your 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 your reads were prepared in this manner but also had barcodes, you should use the barcode splitter tool to separate them by barcode first.

1. 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:

includegraphics[scale=1.0]opbutton.png

2. 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:

includegraphics[scale=0.5]Random.png

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

2.5 Tn-Seq Fitness Analysis

Download Fitness Workflow

This workflow first calculates fitnesses for each insertion in the genome, then aggregates them by gene. You can also use the aggregate tool 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 you want to analyze.

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

includegraphics[scale=0.5]CalcAgg.png

2. In “Step 1: Calculate Fitness”, select your 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 to your time 2 sample’s expansion factor, in that same step.
4. In “Step 2: Aggregate”, Select your GenBank reference genome under the corresponding header.
5. In “Step 2: Aggregate”, 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.
6. Press “Run workflow”, and wait for the workflow to finish running.
7. Once this is done, if you have data from different libraries 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 a custom blank count?” as shown below. For the custom blank count, simply average the blank counts of each fitness

file you're using, 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"!

2.6 Analysis via Command Line

If you know a bit of coding, and would rather run our tools through the command line, you may do this. We don't necessarily recommend it, just because it takes much more of the user's time, as jobs must be entered manually via the command line and don't automatically flow into each other like they do in Galaxy. The steps are as follows:

2.6.1 The Data Prep

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 your 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 (Wwether 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)

2.6.2 Fitness Analysis

1. Download our CalculateFitness tool and our AggregateFitness tool.
2. For every pair of t1/t2 reads, calculate the fitnesses of insertion locations using the calc_fitness.py. Usage as follows:

includegraphics[scale=0.4]CalcOpts.png

3. For every csv file calculate_fitness outputs with the fitnesses of insertion locations, calculate the fitnesses of genes with aggregate.py. Usage as follows:

includegraphics[scale=0.4]AggOpts.png

4. As described in the aggregate.py usage, you may also input multiple insertion location fitness files, and so make "super aggregate" files that incorporate the data from all your runs. This is highly recommended.

3 Comparative Analysis Tools

3.1 Comparative Analysis Tool Summary

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?

Tool	Function	Research Questions Addressed
SlidingWindow	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?
CompGene	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?
CompStrains	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?
CompWindows	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?

3.2 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
```

Usage

```
perl dataOverview.pl -d inputs/ -f genome.fasta -r genome.gbk
```

```
REQUIRED
-d  Directory containing all input files (results files from calc fitness script)
    OR In the command line (without a flag), input the file names
-f  Filename for genome sequence, in fasta format
-r  Filename for genome annotation, in GenBank format

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

Output

The DataOverview tool outputs a summary of 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%

3.3 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 applciation, such as Excel. However, GeneCompare can perform it in less than five seconds for 2000 gene.

Usage

```
perl compGenes.pl <options> <aggregate1.csv> <aggregate2.csv>
```

```
REQUIRED
-d  Directory containing all input files (results files from calc fitness script)
    OR In the command line (without a flag), input the file names

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

3.4 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> -c conversion.csv [aggregateFile1.csv aggregateFile2.csv OR -d directory/]
```

```
REQUIRED
-d  Directory containing all input files (results files from calc fitness script)
    OR In the command line (without a flag), input the file names
-c  Tab delimited file with homologous genes

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

3.5 SlidingWindow: Discovering important unannotated regions

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 tools 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 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.

Usage

```
perl slidingWindow.pl -d inputs/ -o slidWind_output/ -f genome.fasta -r genome.gbk
```

```
  REQUIRED:
  -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:
  -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
      Default=sw_out/
  -w  Do weighted average for fitness per insertion
  -wc  Integer value for weight ceiling. Default=50
```

3.6 WindowCompare: Comparative analysis of small, unannotated regions

WindowCompare takes results from the SlidingWindow 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.

Usage

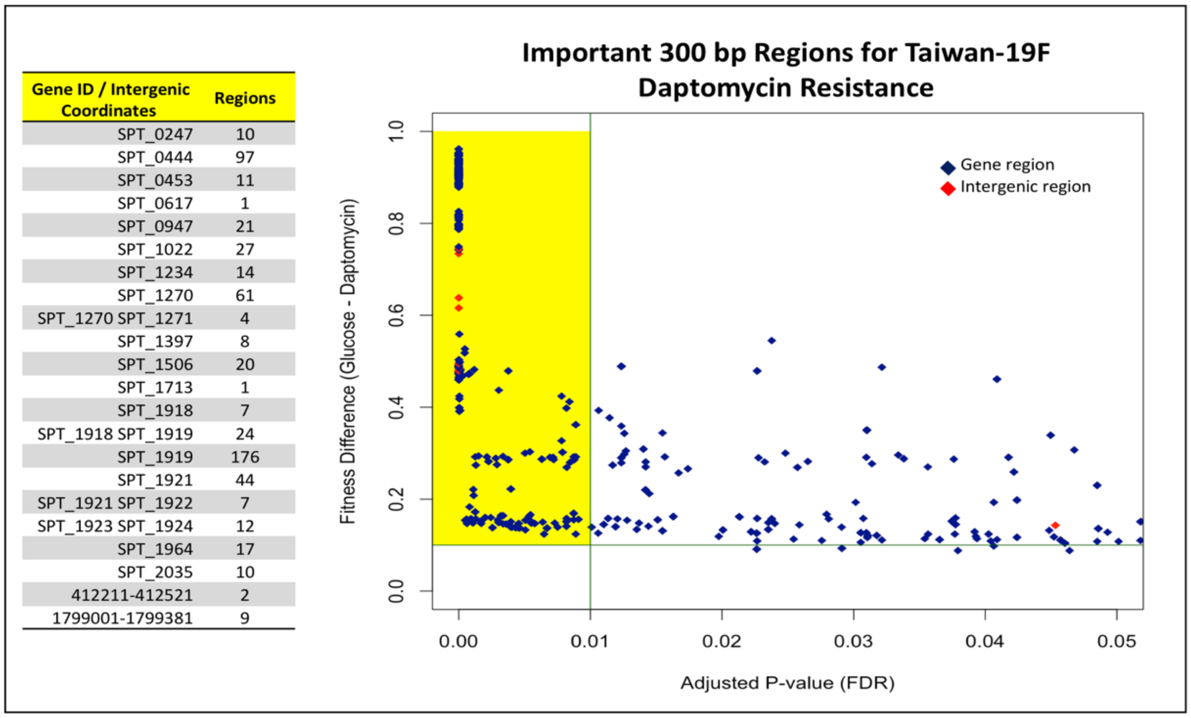
```
perl compWindows.pl <options> [slidingWindows1.csv slidingWindows2.csv OR -d directory]
```

```
REQUIRED
-d  Directory containing all input files (results files from slidingWindow)
    OR In the command line (without a flag), input the file names

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

Example analysis of sliding window output

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.



Gene ID / Intergenic Coordinates	Regions
SPT_0247	10
SPT_0444	97
SPT_0453	11
SPT_0617	1
SPT_0947	21
SPT_1022	27
SPT_1234	14
SPT_1270	61
SPT_1270 SPT_1271	4
SPT_1397	8
SPT_1506	20
SPT_1713	1
SPT_1918	7
SPT_1918 SPT_1919	24
SPT_1919	176
SPT_1921	44
SPT_1921 SPT_1922	7
SPT_1923 SPT_1924	12
SPT_1964	17
SPT_2035	10
412211-412521	2
1799001-1799381	9

Figure 3. The SlidingWindow tool identifies small intergenic and gene regions conditionally important for *Streptococcus pneumoniae* Taiwan-19F daptomycin resistance. The sliding window analysis assessed the aggregate fitness cost of mutations in 211,151 windows (size=300 bp) every 10 bp of the genome. 583 regions were found to be important based on a fitness difference (glucose-daptomycin) > 0.1 and an adjusted p-value <0.01 (highlighted in yellow). These regions fell into 20 genes and two intergenic regions.

Figure 2: alt text

3.7 SingleVal: single Fitness or insertion count per insertion as for visualization inptut (IGV or GvisTnSeq)

Usage

```
perl singleVal.pl -d inputs/ -v count -n NC_XXXOX
```

```
REQUIRED
-d  Directory containing all input files (results files from slidingWindow)
    OR In the command line (without a flag), input the file names

OPTIONAL
-h  Print usage and exits program
-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.wig
```

4 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.

4.1 Multi-track Visualization of Tn-Seq data with Gviz

4.1.1 Installing Gviz and dependency packages

```r
# 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)
```

4.1.2 Input Files

```r
# 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"
```

4.1.3 Global Gviz parameters

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

4.1.4 Gviz Tracks

4.1.4.1 Genome Axis Track

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

4.1.4.2 Sequence Track

```r
# 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)
```

4.1.4.3 Genome Annotation Track

```r
# 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))
```

```r
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")
```

4.1.4.4 Data Track

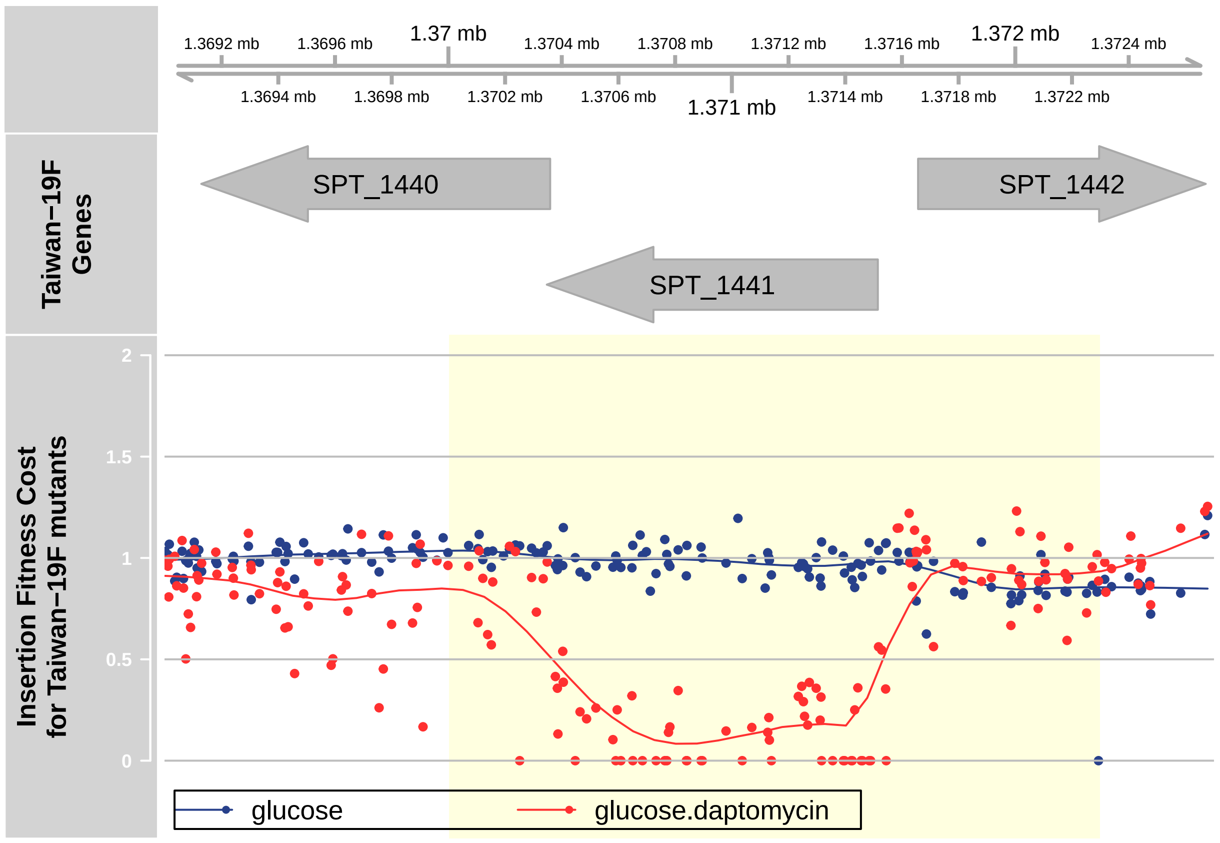
```r
# MAGenTA input: single aggregate fitness per insertion site, from SingleFitness tool
# (ctrlSingleFitFile)

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)
# 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","l"), col.baseline="gray",
                    baseline=c(0,.5,1,1.5,2), col = c("royalblue4","firebrick1"),
                    legend = TRUE, name="Insertion Fitness Cost\nfor Taiwan-19F mutants",
                    fontcolor.legend = "black",
                    fontsize.legend = 12, box.legend = TRUE)
```

4.1.4.5 Plot Tracks

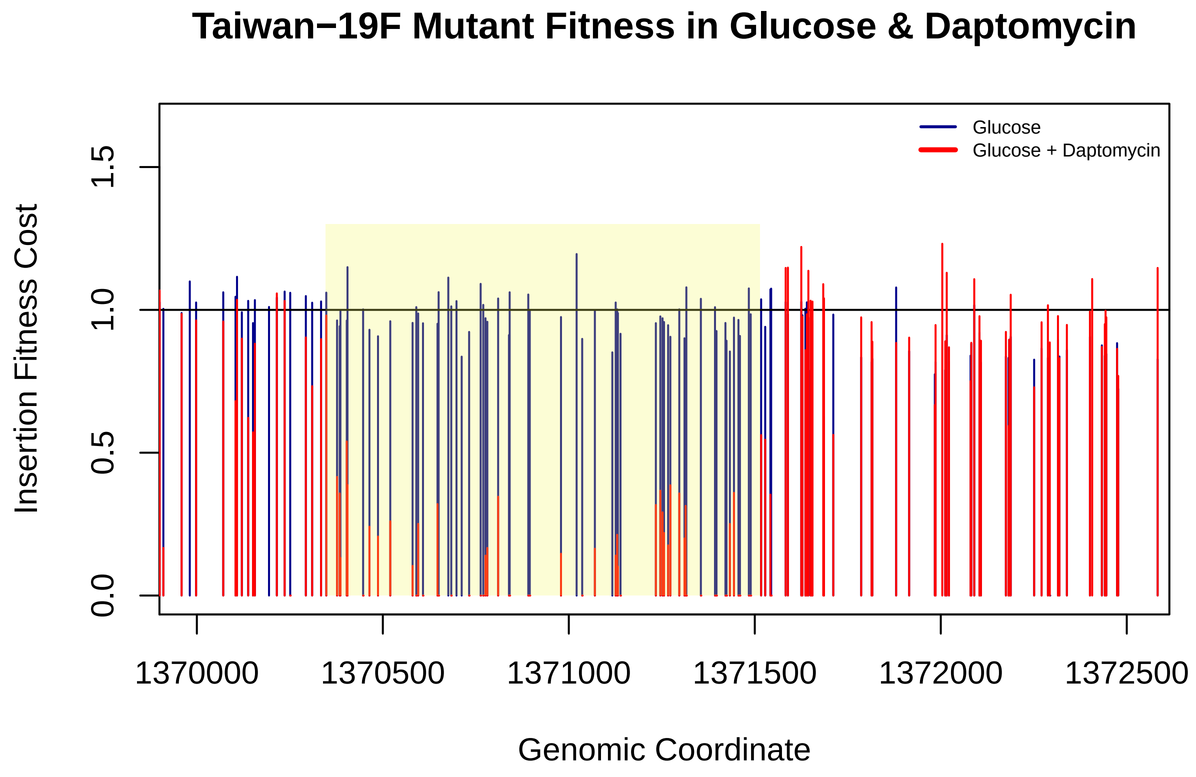
```r
# Specify viewing window start and end coordinates
fromCoord = 1370000
toCoord = 1372300
# Change surrounding flanking region
x=1000
y=400
```

```
#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 = .8, fontfamily = "sans")
```



4.2 Single Track Visualization of Tn-Seq Data

```
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 = "Taiwan-19F Mutant Fitness in Glucose & Daptomycin",
       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
Glucose
Glucose + Daptomycin
Insertion Fitness Cost
Genomic Coordinate