Supplementary Methods

Details of *guideseq* pipeline design:

Sample demultiplexing: A pooled multi-sample sequencing run is demultiplexed into sample-specific read files based on sample-specific dual-indexed 8-nt barcodes.

PCR duplicate consolidation: Reads that share the same unique molecular index (UMI) barcode and the same first six bases of genomic sequence are presumed to originate from the same pre-PCR molecule and are thus consolidated into a single consensus read to improve quantitative interpretation of GUIDE-Seq read counts.

The sample demultiplexing and PCR duplicate consolidation steps are implemented in the *umi* python package (https://github.com/aryeelab/umi).

Read alignment: The demultiplexed, consolidated paired end reads are aligned to a reference genome using the BWA-MEM algorithm with default parameters[1].

Candidate site identification: The start mapping positions of reads amplified with the tag-specific primer (second of pair) are tabulated on a genome-wide basis. Start mapping positions are consolidated using a 10-bp sliding window, to group read start positions that may vary as a consequence of random indel mutagenesis prior to non-homologous mediated end-joining. Windows with reads mapping to both + and - strands, or to the same strand but amplified with both forward and reverse tag-specific primers, are flagged as sites of potential DSBs. This criteria ensures that two distinct molecular events are required to trigger positive site identification. 25 bp of reference sequence is retrieved on either side of the most frequently occuring start-mapping position in each flagged window. This distance is designed to capture the 17 bp and 6 bp sequences expected to flank a typical Cas9 DSB position. The retrieved

Nature Biotechnology doi:10.1038/nbt.3534

sequence is aligned to the intended target sequence using a Smith-Waterman local-alignment algorithm.

False positive filtering: Off-target cleavage sites with more than seven mismatches to the intended target sequence, or that are present in background controls, are filtered out using BEDtools[2]. The mismatch threshold was chosen based on the number of allowable mismatches before reads would begin to commonly align by chance[3] to a typical full-length *S. pyogenes* Cas9 target site.

Reporting: Identified off-targets, sorted by GUIDE-Seq read count are annotated in a final output table. The GUIDE-Seq read count is expected to scale approximately linearly with cleavage rates[3].

References for Supplementary Methods

1. Li, H. & Durbin, R. Fast and accurate short read alignment with Burrows-Wheeler transform. *Bioinformatics* **25**, 1754-1760 (2009).
2. Quinlan, A.R. & Hall, I.M. BEDTools: a flexible suite of utilities for comparing genomic features. *Bioinformatics* **26**, 841-842 (2010).
3. Tsai, S.Q. et al. GUIDE-seq enables genome-wide profiling of off-target cleavage by CRISPR-Cas nucleases. *Nat Biotechnol* **33**, 187-197 (2015).

Supplementary Note

***NOTE*: The following is a snapshot of the guideseq analysis package README information file captured on 19 January 2016. The most recent version of this document can be found at https://github.com/aryeelab/guideseq.**

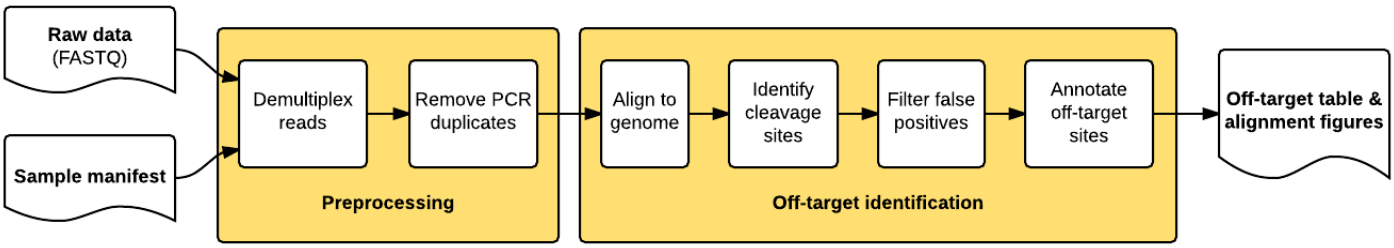
guideseq: The GUIDE-Seq Analysis Package

The guideseq package implements our data preprocessing and analysis pipeline for GUIDE-Seq data. It takes raw sequencing reads (FASTQ) and a parameter manifest file (.yaml) as input and produces a table of annotated off-target sites as output.

Table of Contents

Features

The package implements a pipeline consisting of a read preprocessing module followed by an off-target identification module. The preprocessing module takes raw reads (FASTQ) from a pooled multi-sample sequencing run as input. Reads are demultiplexed into sample-specific FASTQs and PCR duplicates are removed using unique molecular index (UMI) barcode information.

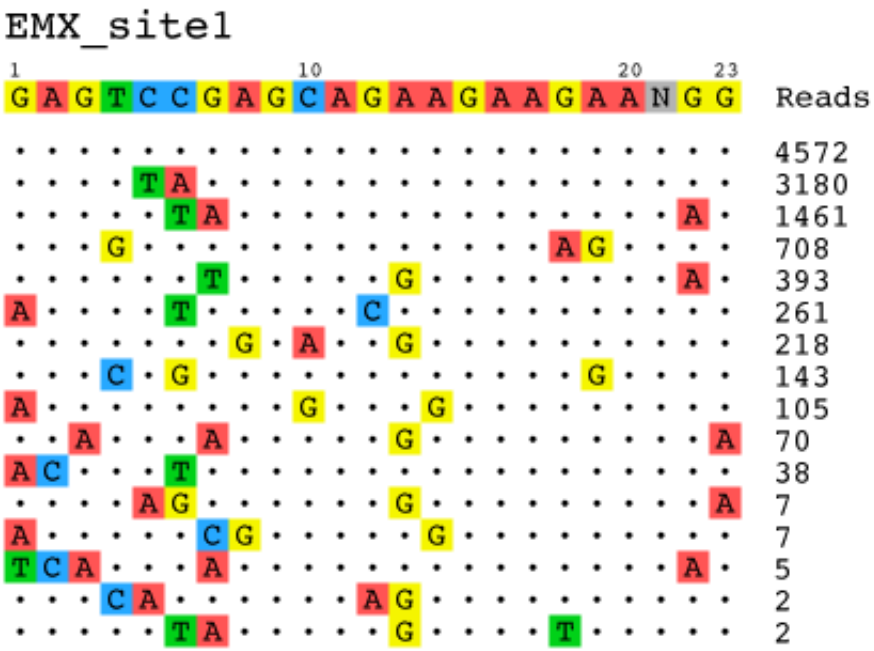


The individual pipeline steps are:

1. **Sample demultiplexing**: A pooled multi-sample sequencing run is demultiplexed into sample-specific read files based on sample-specific dual-indexed barcodes
2. **PCR Duplicate Consolidation**:Reads that share the same UMI and the same first six bases of genomic sequence are presumed to originate from the same pre-PCR molecule and are thus consolidated into a single consensus read to improve quantitative interpretation of GUIDE-Seq read counts.
3. **Read Alignment**: The demultiplexed, consolidated paired end reads are aligned to a reference genome using the BWA-MEM algorithm with default parameters (Li. H, 2009).
4. **Candidate Site Identification**: The start mapping positions of the read amplified with the tag-specific

primer (second of pair) are tabulated on a genome-wide basis. Start mapping positions are consolidated using a 10-bp sliding window. Windows with reads mapping to both + and - strands, or to the same strand but amplified with both forward and reverse tag-specific primers, are flagged as sites of potential DSBs. 25 bp of reference sequence is retrieved on either side of the most frequently occuring start-mapping position in each flagged window. The retrieved sequence is aligned to the intended target sequence using a Smith-Waterman local-alignment algorithm.

5. **False positive filtering**: Off-target cleavage sites with more than six mismatches to the intended target sequence, or that are present in background controls, are filtered out.
6. **Reporting**: Identified off-targets, sorted by GUIDE-Seq read count are annotated in a final output table. The GUIDE-Seq read count is expected to scale approximately linearly with cleavage rates (Tsai et al., *Nat Biotechnol.* 2015).
7. **Visualization**: Alignment of detected off-target sites is visualized via a color-coded sequence grid, as seen below:



Dependencies

- Python (2.7)
- Reference genome fasta file (Example)
- `bwa` alignment tool
- `bedtools` genome arithmetic utility

Getting Set Up

Install Dependencies

To run guideseq, you must first install all necessary dependencies:

- **Python 2.7**: If a version does not come bundled with your operating system, we recommend the Anaconda scientific Python package.
- **Burrows-Wheeler Aligner (bwa)**: You can either install bwa with a package manager (e.g. `brew` on OSX or `apt-get` on Ubuntu/Debian), or you can download it from the project page and compile it from source.
- **Bedtools**: You can either install bwa with a package manager (e.g. `brew` or `apt-get`), or you can download it from the project page and compile it from source.

For both bwa and bedtools, make sure you know the path to the respective executables, as they need to be specified in the pipeline manifest file.

Download Reference Genome

The guideseq package requires a reference genome for read mapping. You can use any genome of your choosing, but for all of our testing and original GUIDE-seq analyses (Tsai et al. *Nature Biotechnol* 2015) we use hg19 (download). Be sure to (g)unzip the FASTA file before use if it is compressed.

Download and Set Up guideseq

Once all dependencies are installed, there are a few easy steps to download and set up the guideseq package:

1. Obtain a copy of the guideseq package source code. You can either download and unzip the latest source from the github release page, or you use git to clone the repository by running `git clone --recursive https://github.com/aryeelab/guideseq.git`
2. Install guideseq dependencies by entering the guideseq directory and running `pip install -r requirements.txt`

Once all guideseq dependencies are installed, you will be ready to start using guideseq.

Configuring a MiSeq to Output Index Reads

The guideseq package requires index reads from the MiSeq sequencing run for read consolidation. The default MiSeq Reporter settings do not generate index (I1, I2) reads. This feature can be enabled by adding the line

```
<add key="CreateFastqForIndexReads" value="1">
```

to the `Miseq Reporter.exe.config` file located in the Miseq Reporter installation folder. The default

installation folder is `C:\Illumina\MiSeqReporter`. After modifying the config file it should look like this:

```
<appSettings>
    ... [LEAVE EXISTING LINES UNCHANGED] ...
    <add key="CreateFastqForIndexReads" value="1">
</appSettings>
```

The MiSeq Reporter service needs to be restarted for the change to take effect. Future runs of the GenerateFASTQ workflow (and probably other workflows) will generate I1 and I2 reads in addition to R1 and R2. All four of these reads files will be needed for guideseq analysis.

See page 29 of the Miseq Reporter User Guide for further instructions.

Running the Full Analysis Pipeline

Quickstart

To run the full guideseq analysis pipeline, you must first create a manifest YAML file that describes all pipeline inputs. Once you have done so, you can simply run

```
python /path/to/guideseq.py all -m /path/to/manifest.yaml
```

to run the entire pipeline. Below are specific instructions detailing how to write the manifest file.

If you wish to run an example on our abridged test data, you can simply run

```
python guideseq/guideseq.py all -m test/test_manifest.yaml
```

from the guideseq root directory. The `test_manifest` assumes that both the `bwa` and `bedtools` executables are in your system PATH. You will see the pipeline results outputted to the `test/output` folder.

Writing A Manifest File

When running the end-to-end analysis functionality of the guideseq package, a number of inputs are required. To simplify the formatting of these inputs and to encourage reproducibility, these parameters are inputted into the pipeline via a manifest formatted as a YAML file. YAML files allow easy-to-read specification of key-value pairs. This allows us to easily specify our parameters. The following fields are required in the manifest:

- `reference_genome`: The absolute path to the reference genome FASTA file.

- `output_folder` : The absolute path to the folder in which all pipeline outputs will be saved.
- `bwa` : The absolute path to the `bwa` executable
- `bedtools` : The absolute path to the `bedtools` executable
- `undemultiplexed` : The absolute paths to the undemultiplexed paired end sequencing files. The required parameters are:
    - `forward` : The absolute path to the FASTQ file containing the forward reads.
    - `reverse` : The absolute path to the FASTQ file containing the reverse reads.
    - `index1` : The absolute path to the FASTQ file containing the forward index reads.
    - `index2` : The absolute path to the FASTQ file containing the reverse index reads.

An example `undemultiplexed` field:

```
undemultiplexed:
    forward: ../test/data/undemux.r1.fastq.gz
    reverse: ../test/data/undemux.r2.fastq.gz
    index1: ../test/data/undemux.i1.fastq.gz
    index2: ../test/data/undemux.i2.fastq.gz
```

- `samples` : A nested field containing the details of each sample. At least two samples must be specified: a "control" sample (to be used to filter out background off-target sites) and at least one treatment sample. The required parameters are:
    - `target` : The sample targetsites
    - `barcode1` : The forward barcode
    - `barcode2` : The reverse barcode
    - `description` : A description of the sample

An example `samples` field:

```
samples:
    control:
        target:
        barcode1: CTCTCTAC
        barcode2: CTCTCTAT
        description: Control

    [SAMPLENAME]:
        target: GAGTCCGAGCAGAAGAAGAANGG
        barcode1: TAGGCATG
        barcode2: TAGATCGC
        description: EMX1
```

A Full Manifest File Example

Below is an example of a full manifest file. Feel free to copy it and replace the parameters with your own experiment data. Remember that you can input more than just one treatment sample (e.g. the "EMX1" data below).

```
reference_genome: test/test_genome.fa
output_folder: test/output

bwa: bwa
bedtools: bedtools

demultiplex_min_reads: 1000

undemultiplexed:
    forward: test/data/undemultiplexed/undemux.r1.fastq.gz
    reverse: test/data/undemultiplexed/undemux.r2.fastq.gz
    index1: test/data/undemultiplexed/undemux.i1.fastq.gz
    index2: test/data/undemultiplexed/undemux.i2.fastq.gz

samples:
    control:
        target:
        barcode1: CTCTCTAC
        barcode2: CTCTCTAT
        description: Control

    EMX1:
        target: GAGTCCGAGCAGAAGAAGAANGG
        barcode1: TAGGCATG
        barcode2: TAGATCGC
        description: EMX_site1
```

Pipeline Output

When running the full pipeline, the results of each step are outputted to the `output_folder` in a separate folder for each step. The output folders and their respective contents are as follows:

Output Folders

- `output_folder/demultiplexed` : Contains the four undemultiplexed reads files (forward, reverse, index1, index2) for each sample.
- `output_folder/umitagged` : Contains the two umitgged reads files (forward, reverse) for each

sample.

- `output_folder/consolidated` : Contains the two consolidated reads files (forward, reverse) for each sample.
- `output_folder/aligned` : Contains an alignment `.sam` file for each sample.
- `output_folder/identified` : Contains a tab-delimited `.txt` file for each sample with an identified off-target in each row.
- `output_folder/filtered` : Contains a tab-delimited `.txt` file for each sample containing the identified DSBs that are background sites (not off-targets)
- `output_folder/visualization` : Contains a `.svg` vector image representing an alignment of all detected off-targets to the targetsite for each sample.

The final detected off-target sites are placed in the `output_folder/identified` folder, with one `.txt` file for each sample specified in the manifest. The fields that are populated in each row of these off-target files are specified below:

Output Off-Targets `.txt` Fields:

- `BED Chromosome` : Window chromosome
- `BED Min.Position` : Window 0-based start position
- `BED Max.Position` : Window 0-based end position
- `BED Name` : Name of window
- `Filename` : The name of the current `.SAM` file used in analysis.
- `WindowIndex` : Index number of window
- `Chromosome` : Chromosome corresponding to position with maximum reads in window (matches `BED Chromosome`)
- `Position` : Position with maximum number of reads in window
- `Sequence` : The window sequence, starting 25 bp upstream and ending 25 bp downstream of `Chromosome:Position`
- `+.mi` : Number of forward reads with distinct molecular indices
- `-.mi` : Number of reverse reads with distinct molecular indices
- `bi.sum.mi` : Sum of the `+.mi` and `-.mi` fields (GUIDE-seq Read Count)
- `bi.geometric_mean.mi` : Geometric mean of the `+.mi` and `-.mi` fields
- `+.total` : Total number of forward mapping reads
- `-.total` : Total number of reverse mapping reads
- `total.sum` : Sum of `+.total` and `-.total` fields
- `total.geometric_mean` : Geometric mean of the `+.total` and `-.total` fields
- `primer1.mi` : Number of reads amplified by forward primer with distinct molecular indices
- `primer2.mi` : Number of reads amplified by reverse primer with distinct molecular indices
- `primer.geometric_mean` : Geometric mean of the `primer1.mi` and `primer2.mi` fields

- `position.stdev` : Standard deviation of positions within genomic window
- `Off-Target Sequence` : Off-target sequence derived from genome reference
- `Mismatches` : Number of mismatches between the intended target sequence and the off-target sequence
- `Length` : Length of the target sequence
- `BED off-target Chromosome` : Off-target chromosome
- `BED off-target start` : Off-target 0-based start position
- `BED off-target end` : Off-target 0-based end position
- `BED off-target name` : Off-target name
- `BED Score` : Field to conform to standard BED format
- `Strand` : Indicates the strand of detected off-target site. `+` for forward strand and `-` for reverse strand
- `Cells` : Cell type
- `Target site` : Targetsite name
- `Target Sequence` : Intended target site sequence (including PAM)

The key fields for interpreting this output and identifying off-target sites are:
`BED off-target Chromosome`, `BED off-target start`, `BED off-target end`, `BED off-target name`, `BED off-target strand`, `Off-Target Sequence`, `bi.sum.mi`

Output Visualizations

The outputted visualizations are in the `.svg` vector format, which is an open image standard that can be viewed in any modern web browser (e.g. Google Chrome, Apple Safari, Mozilla Firefox), and can be viewed and edited in any vector editing application (e.g. Adobe Illustrator). Because the output visualizations are vector images, they can be scaled up or down infinitely without a loss in quality, and can also be edited as shapes with ease. This makes the images produced by the guideseq package ideal for posters, presentations, and papers.

Running Analysis Steps Individually

In addition to end-to-end pipeline analysis functionality, the guideseq package also allows for every step fo the analysis to be run individually. Here we have detailed the required inputs and expected outputs of each step. For each step, we have included a "runnable example" command that can be executed from the guideseq root directory to run that step on the included sample data. These "runnable example" snippets put their output in the `test/output` folder.

`demultiplex` Pooled Multi-Sample Sequencing (Manifest Required)

- **Functionality**: Given undemultiplexed sequence files and sample barcodes specified in the manifest, output the demultiplexed sample-specific reads in FASTQ format. The forward, reverse, and two index files for each sample in the manifest are outputted to the `output_folder/consolidated` folder.
- **Required Parameters**:
  - `-m or --manifest` : Specify the path to the manifest YAML file
- **Runnable Example**:
  - `python guideseq/guideseq.py demultiplex -m test/test_manifest.yaml`

`umitag` Reads

- **Functionality**: Given the demultiplexed files in the folder `output_folder/undemultiplexed` (where `output_folder` is specified in the manifest), 'tag' the reads by adding the UMI barcode sequence to the FASTQ read name header in preparation for the subsequent PCR duplicate read consolidation step. The forward and reverse files for each sample in the manifest are outputted to the `output_folder/umitagged` folder.
- **Required Parameters**:
  - `--read1` : Path to the forward demultiplexed reads file (FASTQ)
  - `--read2` : Path to the reverse demultiplexed reads file (FASTQ)
  - `--index1` : Path to the index1 demultiplexed reads file (FASTQ)
  - `--index2` : Path to the index2 demultiplexed reads file (FASTQ)
  - `--outfolder` : Path to the folder in which the output files will be saved
- **Runnable Example**:

```
python guideseq/guideseq.py umitag --read1 test/data/demultiplexed/EMX1.r1.fastq
\
--read2 test/data/demultiplexed/EMX1.r2.fastq \
--index1 test/data/demultiplexed/EMX1.i1.fastq \
--index2 test/data/demultiplexed/EMX1.i2.fastq \
--outfolder test/output/
```

`consolidate` PCR Duplicates

- **Functionality**: Given undemultiplexed sequence files and sample barcodes specified in the manifest, output the consolidated forward and reversed reads to the `outfolder` .
- **Required Parameters**:
  - `--read1` : Path to the forward umitagged reads file (FASTQ)

- `--read2` : Path to the reverse umitagged reads file (FASTQ)
- `--outfolder` : Path to the folder in which the output files will be saved

- **Optional Parameters**:

  - `--min_quality` : The minimum quality of a read for it to be considered in the consolidation
  - `--min_frequency` : The minimum frequency of a read for the position to be consolidated

- **Runnable Example**:

```
python guideseq/guideseq.py consolidate --read1 test/data/umitagged/EMX1.r1.umita
gged.fastq \
 --read2 test/data/umitagged/EMX1.r2.umitagged.fastq \
 --outfolder test/output/
```

`align` Sites to Genome

- **Functionality**: Given the consolidated forward and reverse reads, execute a paired-end mapping of the sequences to the reference genome using the `bwa` package. Outputs an alignment `.sam` file to the `outfolder`.
- **Required Parameters**:

  - `--bwa` : Path to the `bwa` executable
  - `--genome` : Path to the reference genome FASTA file
  - `--read1` : Path to the consolidated forward read FASTQ file
  - `--read2` : Path to the consolidated reverse read FASTQ file
  - `--outfolder` : Path to the folder in which the output files will be saved

- **Runnable Example**:

```
python guideseq/guideseq.py align --bwa bwa --genome test/test_genome.fa\
 --read1 test/data/consolidated/EMX1.r1.consolidated.fastq\
 --read2 test/data/consolidated/EMX1.r2.consolidated.fastq\
 --outfolder test/output/
```

`identify` Off-target Site Candidates

- **Functionality**: Given the alignment samfile for a given site, a reference genome, and a target sequence, output a tab-delimited `.txt` file containing the identified off-target sites.
- **Required Parameters**:

- `--aligned` : Path to the site-specific alignment `.sam` file.
- `--genome` : Path to the reference genome FASTA file.
- `--outfolder` : Path to the folder in which the output files will be saved.
- `--target_sequence` : The sequence targeted in the sample (blank for control sample)

- **Optional Parameters**:

  - `--description` : Specify additional information about the sample.

- **Runnable Example**:

```
python guideseq/guideseq.py identify --aligned test/data/aligned/EMX1.sam\
 --genome test/test_genome.fa --outfolder test/output/\
 --target_sequence GAGTCCGAGCAGAAGAAGAANGG --description EMX1
```

`filter` Background DSB Sites

- **Functionality**: Given the identified site `.txt` files for a treatment and control samples, output a `.txt` file in the same format as outputted by the `identify` step containing the sites filtered out as false-positives.
- **Required Parameters**:

  - `--bedtools` : Path to the `bedtools` executable
  - `--identified` : Path to the `.txt` file outputted by the `identify` step for a treatment sample.
  - `--background` : Path to the `.txt` file outputted by the `identify` step for a control sample.
  - `--outfolder` : Path to the folder in which the output files will be saved.

- **Runnable Example**:

```
python guideseq/guideseq.py filter --bedtools bedtools\
 --identified test/data/identified/EMX1_identifiedOfftargets.txt\
 --background test/data/identified/control_identifiedOfftargets.txt\
 --outfolder test/output/
```

`visualize` Detected Off-Target Sites

- **Functionality**: Given an identified off-target sites `.txt` file, output an alignment visualization of the off-target sites.
- **Required Parameters**:

- `--infile` : Path to the input `.txt.` off-targets file
- `--outfolder` : Path to the outputted folder containing the outputted `.svg` graphic

- **Optional Parameters**:
  - `--title` : Specify the title of the visualization, to be printed at the top of the graphic. Useful for posters and presentations.

- **Runnable Example**:

```
python guideseq/guideseq.py visualize --infile test/data/identified/EMX1_identifi
edOfftargets.txt\
 --outfolder test/output/ --title EMX1
```

Testing the guideseq Package

In the spirit of Test-Driven Development, we have written end-to-end tests for each step of the pipeline. These can be used to ensure that the software is running with expected functionality.

NOTE: Due to differences in sorting between different versions of the `bwa` package, you must be using `bwa v0.7.9a` for these tests to work. We also recommend that you use `bedtools v2.25.0` when running these tests for consistency's sake.

Single-Step Regression Tests

For ongoing testing and development, we have created an abridged set of input data and expected output data for each step of the pipeline. This way, changes to the pipeline can be quickly tested for feature regression.

To run these tests, you must first install the `nose` testing Python package.

```
pip install nose
```

Then, from the guideseq root directory, simply run

```
nosetests
```

and the regression tests for each pipeline step will be run.

Full Large Test

If you have more time, we have prepared a bash script that downloads and compiles all dependencies from

source, downloads a fresh reference genome and a full GUIDE-seq sequencing data run, and performs a full test of the entire pipeline. This test takes a long time, but we require that it be run before we tag a new release.

To run the full large test, enter the `guideseq/test` folder and run

```
./large_test.sh
```

Then, sit back and watch the full large testing process unfold automatically in the terminal.

Manual Testing

If you wish to run a full GUIDE-Seq dataset through the analysis pipeline, you may find it and a test manifest (to be altered depending on your dependency locations) here:

```
http://aryee.mgh.harvard.edu/guideseq/data/guideseq_test_fastq.zip
```

which should be used with the following reference genome:

```
http://www.broadinstitute.org/ftp/pub/seq/references/Homo_sapiens_assembly19.fasta
```

Frequently Asked Questions

How do I Run the Pipeline with Demultiplexed Data?

If you already have demultiplexed data, you can run the pipeline on the data by running each step after demultiplexing individually, as described in the "Running Analysis Steps Individually" section above. Be sure to run the individual steps in the following orders:

- `umitag`
- `consolidate`
- `align`
- `identify`
- `filter`
- `visualize`

Can I analyze data without UMIs?

Yes. If your reads do not have UMIs, you can run the pipeline on previously demultiplexed data as described in the "Running Analysis Steps Individually" section above, starting with the `align` step. **Note that we have**

not analyzed such data ourselves! We suspect that PCR duplication bias may affect the quantitative interpretion of GUIDE-Seq read counts, but have not explored this.