PinAPL-Py Tutorial

NOTE: The analysis of sequencing data requires high performance computational resources. For large sequencing data files (≥10M reads), this workflow is recommended to run with more than 8GB of RAM.

Table of Contents

1. Quick Start

0. **Install docker** on your machine: https://docs.docker.com/engine/installation/

1. **Create a working directory** and create both a /Data and /Library subfolder inside it

2. **Copy your read files** (fastq.gz) to the /Data folder.

3. **Rename your read files.** To be recognized and correctly interpreted, the fastq.gz file names of control replicates need to start with *"Control_R1_..."*, *"Control_R2_...."* etc. Similarly, file names of treatment replicates (e.g. treatmentX) need to start with *"TreatmentX_R1_..."*, *"TreatmentX_R2_..."* etc. For more information on filename requirements see section 2.

4. **Copy the library file** (tsv) to the /Library folder. The library file is a tab delimited file of all sgRNAs in the library. Columns should be 1: gene_ID, 2: sgRNA_ID, 3: sequence. If you work with the GECKO_v2 library, you can download this file from GitHub

5. **Download the default configuration file** and copy it to the working directory. The file (configuration.yaml) can be found on GitHub

6. **Edit the configuration file**. *For GECKO_v2 enrichment screens, you can leave everything at default.*

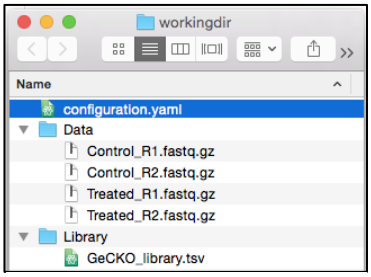

Figure 1: Working directory content before starting PinAPL-Py

7. **Start PinAPL-Py** from the working directory using Terminal (Windows: Docker Quickstart Terminal). *(Note: On Linux, you may need to execute as super user using "sudo" before the command).*

```
docker run -v $PWD:/workingdir oncogx/pinaplpy_docker PinAPL.py
```

The workflow will create several output folders. For interpretation of the results, refer to Section 3 of the tutorial.

2. Instructions

0. Install the Docker software on your machine

   Follow https://docs.docker.com/engine/installation/ to install Docker on your operating system. For Windows users (including **Win 10 Pro**): follow the links for the **Docker Toolbox**.

   To validate successful installation, open a Terminal window (Linux and MacOS) or the Docker QuickStart Terminal (Windows) and type **docker** to see a list of available options.

1. Download the PinAPL-Py docker image (~2 GB)

   Open the Terminal (Linux, MacOS) / Docker QuickStart Terminal (Windows). Type

   ```
   docker pull oncogx/pinaplpy_docker
   ```

2. Prepare a working directory

   Make a **new folder (e.g. "workingdir")** on your machine, preferably on the Desktop. Choosing a different location is fine, but you need to be able to navigate to it using the command line (step 7). Avoid spaces in the path to the name of your working directory.

3. Prepare a Data subfolder

   Make a subfolder "/Data" in your working directory and copy your read files into it. Your read files need to be in **zipped fastq format (.fastq.gz)**. If yours are in a different format, contact your sequencing facility and ask for .fastq.gz format instead.

   NOTE: You can download a test dataset from GitHub

4. Rename your filenames

   Have each filename start with a **prefix** that specifies condition and replicate number. Let it follow the format

   <samplename>_R< number>_

   For example, if your file represents replicate 1 of a treatment with Treatment A, the filename would start with **TreatmentA_R1_**. You can choose any name instead of "TreatmentA" as long as it consists of letters, numbers and hyphen ("-"). Avoid other special characters, underscores or spaces.

   For your control replicates, choose **Control_R1_..., Control_R2_...** etc. Edit the configuration file (see step 6) if you want to choose a different name for your controls.

   It is acceptable to have prefixes like TreatmentA_R1a, TreatmentA_R1b (e.g. for technical replicates).

   IMPORTANT: PinAPL-Py requires **at least 2 control replicates.**

5. Prepare a Library subfolder

   Make a subfolder "/Library" in your working directory and copy your library spreadsheet into it. Make sure your spreadsheet has **three columns: gene name | sgRNA name | sequence** (Column header names can be changed). Refer to the GeCKO library file on GitHub as a layout example. **Save the spreadsheet as a tab-delimited text file with ending .tsv (="tab-separated values")!**

   NOTE: If you're running the test dataset, you need to use the GeCKO library file from GitHub

6. Edit the configuration file

Download **configuration.yaml** from GitHub and place it in your working directory.

Open the file using a text editor (e.g. TextEdit/Gedit on MacOS/Linux; **Notepad++** on Windows). While most parameters can be left at default, it is important to check the following parameters:

- *ScreenType*: Specify if your screen is an enrichment or depletion (dropout) screen
- *LibFilename:* Enter the filename of your library filename (Step 5)
- *seq_5_end:* Enter the adapter sequence that lies 5' of the sgRNA in your reads (see section 5 for an illustration)
- *seq_3_end:* Enter the adapter sequence that lies 3' of the sgRNA in your reads (see section 5 for an illustration)
- *CtrlPrefix:* Enter the name that specifies the prefix of your control samples *(e.g. if your control files are Control_R1_..., Control_R2,… etc, enter 'Control_')*
- *NonTargetPrefix:* Enter the ID prefix of non-targeting sgRNAs in your library *(e.g. 'NonTargeting'. Leave unchanged if you are working with GeCKOv2 or if no such sgRNAs are present.)*
- *sgRNAsPerGene:* Enter the number of sgRNAs that target each gene *(e.g. 6 in case of the GeCKOv2 library)*

For a description of the remaining parameters see Section 5.

7. Navigate to your working directory

   In the Terminal (Linux, MacOS) / Docker QuickStart Terminal (Windows) use **cd** to navigate into your working directory. E.g. if your working directory is on the desktop, type

   ```
   cd Desktop/workingdir
   ```

   If your working directory is in a different location, replace the path to it accordingly. To help you navigate, you can type **ls** to see the files and subfolders of your current locaction.

8. Choose a name for the PinAPL-Py container (e.g. pinaplpy_test) and type the command below (without line breaks!). This creates a container (i.e. a virtual server that runs the PinAPL-Py workflow) and links it to your working directory. Make sure the working directory follows the structure shown in Fig. 1 (Section 1).

   ```
   docker run -t -i --name pinaplpy_test -v $PWD:/workingdir
   oncogx/pinaplpy_docker
   ```

   This way, the output generated by PinAPL-Py will be automatically made available in your working directory.

   NOTE: You can choose any name instead of *pinaplpy_test* (avoid spaces, thouth!). Keep track of which container name you chose for which working directory, so you can repeat your analysis at a later point if needed.

9. Start the workflow by typing

   ```
   PinAPL.py
   ```

   All read files in your Data directory will be automatically processed. After completion, your working directory will contain two additional subfolders /Alignments and /Analysis that are described in detail below (Section 3).

   NOTE: For running individual parts of the PinAPL-Py workflow (for example, if you only want to check coverage of your library), refer to Section 4

   After the workflow has completed, you can type **exit** to exit the container.

3. Description of the PinAPL-Py output

NOTE: Spreadsheets (.tsv) are best viewed in Excel (or similar). Text files (.txt) are best viewed in TextEdit/Gedit (MacOS/Linux) or Notepad++ (Windows) !

/Data:

For each .fastq.gz file PinAPL-Py generates a .fastq.gz file with the 5' and 3' adapters removed (parameters 'seq_5_end'/'seq_3_end', see Section 5). This filename has the suffix '_cut.fastq' and is used as input for Bowtie2 alignment. By default, this file is deleted after completion of the workflow. The behavior can be changed in the configuration file (for troubleshooting).

/Library:

PinAPL-Py calls Bowtie2 to generate an index for the library provided. The files for this index are stored in the subfolder /Library/Bowtie2_Index. PinAPL-Py will check the /Library folder for the presence of this index folder during the run, so index generation is only required when working with a library for the first time.

/Alignments

For each sample, PinAPL-Py stores a spreadsheet listing which sgRNA_ID is matched to each read, together with the mapping quality (file '…_bw2output.tsv'). The full Bowtie2 output is saved as a SAM file (file '…_bw2output.sam'). Alignments are extracted from this file and saved separately as text (file '…_SAM_alignments.txt'). By default, the full Bowtie2 output is converted from SAM to BAM, and the textfile containing the extracted alignments is compressed to tar.gz.

/Analysis:

/QC:

This folder stores various output from the individual PinAPL-Py scripts that is interesting for both quality control and exploratory analysis.

- *cutadapt.txt:* This file contains the log output from cutadapt (Martin, 2011) providing information about the read clipping process. Refer to the cutadapt manual for a detailed description.
- *Bowtie2_AlignmentScores.png:* This contains a histogram showing the distributions the primary (best) and secondary (second-best) alignment scores achieved for each read. The matched and discarded read fractions are marked. For more information about alignment scores, refer to the Bowtie2 manual.
- *Bowtie2_MappingQuality_Hist.png:* This contains a histogram of the mapping quality of the Bowtie2 alignment protocol. For more information about mapping quality, refer to the Bowtie2 manual.
- *Bowtie2_Results.txt:* Text file listing statistics of the Bowtie2 alignment.
- **GuideCounts.txt / *GeneCounts.txt:* Text files listing the raw read counts for each sgRNA or each gene in the library, respectively.
- **GuideCounts_0.txt / *GeneCounts_0.txt:* Text files listing the counts for each sgRNA or each gene in the library, respectively, normalized to a standard (1M reads by default, see the configuration file).
- **GuideCounts_0_sorted.txt:* Text file listing the normalized read counts for each sgRNA, sorting by either the highest or lowest counts, for enrichment or depletion screens, respectively.

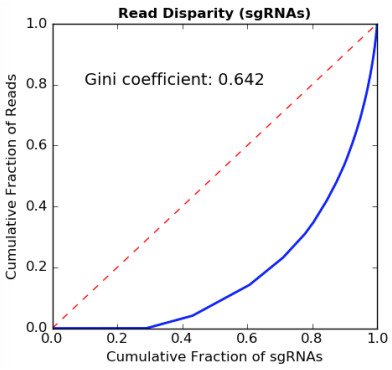

Figure 2: example of a Lorenz curve generated by PinAPL-Py

- *ReadCounts_Distribution.png:* Boxplots (outliers removed) and histograms for the normalized read counts per sgRNA or gene, respectively.
- *ReadCounts_LorenzCurve.png:* Lorenz curves and Gini coefficients for both normalized read counts per sgRNA and gene, respectively. The Lorenz curve (Figure 2) provides information about the disparity of read counts across the sample, and thus serves as an indicator of the quantity of selection in a sample. A perfectly even distribution of reads counts results in a diagonal curve (Gini coeff = 0) while an extreme uneven distribution (all reads concentrated on a single sgRNA) results in a flat curve (Gini coeff = 1).
- *ReadCounts_Statistics.txt:* Text file summarizing statistics of read count distributions of both sgRNAs and genes.

/Control:

This folder contains a tab-delimited text file listing the normalized read counts for all control samples, with their mean, sample variance and variance as estimated from negative binomial regression (Robinson and Smyth, 2007). The scatterplots show the over-dispersion in the read counts (by how much does count variance exceed the mean) and the linear regression used to estimate sample variances following a negative binomial model.

/Candidate_Lists:

This folder contains tab-delimited spreadsheets (**sgRNAList.tsv*) for each (non-control) sample listing the normalized read count for each sgRNA, the mean read count from the controls, the standard deviation of read counts from the controls, the fold-change in read counts relative to control, the p-value, and the (FDR-corrected) significance of enrichment/depletion. Another file (**GeneList.tsv*) lists significantly enriched/depleted genes, as measured using a gene ranking method. PinAPL-Py implements three gene ranking strategies, selected in the configuration file: aRRA (Li *et al.*, 2014), STARS (Doench *et al.*, 2016) and ES (Subramanian *et al.*, 2005).

/sgRNA_Efficiency:

This folder provides barplots showing the number of sgRNAs with significant counts per gene (among all genes with at least 1 significant sgRNA normalized count).

/ScatterPlots:

This folder contains scatterplots of normalized sgRNA log counts of control versus sample as well each pair of treatment replicates against each other. Scatterplots that were generated by manual running the “PlotSample” script are stored in this folder as well (see Section 4).

/Heatmap:

This folder contains a heatmap of sgRNA log counts used to perform unsupervised clustering of the samples in your dataset. By default, the 100 sgRNAs showing the highest variance across the dataset are chosen (this behavior can be adjusted in the configuration file). The list of sgRNAs included is saved as a separate spreadsheet.

4. Manually running individual PinAPL-Py steps

In addition to running the entire PinAPL-Py workflow (preferred option), users may be interested in repeating individual steps for troubleshooting or optimization. Typical reasons include:

1. No enrichment analysis is desired. For example, samples are only processed to check quality and coverage.
2. Individual genes of interest need to be highlighted in a scatterplot
3. Certain other steps are to be repeated with different parameter settings (e.g. gene ranking method) without having to re-run the complete workflow. Note that each step relies on data generated in preceding steps in the workflow as well as on the configuration file, so when re-running steps, make sure all previous steps did successfully finish and the data is present in the expected folder.

4.1. Working with containers and scripts

4.1.1 Resuming a previously closed PinAPL-Py container session

In order to resume a previously closed session, you need to first restart and then re-attach the PinAPL-Py container that you created when linking your working directory (see Section 2, Step 8). Assuming the name of the container you chose was pinaplpy_test, open the Terminal (Linux/MacOS) / Docker QuickStart Terminal (Windows) and type

```
docker start pinaplpy_test
```

This restarts the previously closed container. The name of the container (pinaplpy_test) should show up in the terminal. Next, in order to attach the container, type

```
docker attach pinaplpy_test
```

Depending on your Terminal session, you might have to hit ENTER twice (!) after this command. You are now in the container and can run individual PinAPL-Py scripts.

For more information about handling containers and images, refer to Section 4.1.3

4.1.2 Running individual scripts in the PinAPL-Py workflow

script00: BuildLibraryIndex

PinAPL-Py uses the Bowtie2 aligner to align reads to the library of sgRNA sequences. This script translates the provided library spreadsheet into .fasta format and builds an index for the library. For more information about library indexing refer to the Bowtie2 manual.

To run this script, type

```
BuildLibraryIndex.py
```

The output is stored in the /Library folder.

script01: AlignReads

After the library has been indexed, the reads (.fastq.gz file) from the specified sample are aligned to the reference library using Bowtie2 (Langmead and Salzberg, 2012). Before alignment, the script calls the Cutadapt program (Martin, 2011) in order to clip off 5' and 3’ PCR-adapter sequences and isolate the sgRNA sequence within the read. Alignment statistics are provided, such as read depth, mapping quality and fractions of successful and failed alignments. Reads with failed alignments are discarded. Reads that cannot be unambiguously mapped to a single library entry are discarded as well (The ambiguity tolerance level can be specified in the configuration file). After alignment, reads are counted, and counts are normalized.

To run this script, type

```
AlignReads.py <sample>
```

where <sample> is the sample prefix (e.g. Treatment_R1). The output is stored in the /Alignments folder.

script02: AnalyzeCounts

This script analyzes the distribution of read counts in the sample, and produces plots and text files on count statistics.

To run this script, type

```
AnalyzeCounts.py <sample>
```

where <sample> is the sample prefix (e.g. TreatmentA_R1). The output is stored in the /Analysis/QC folder.

script03: AnalyzeControl

This script analyzes estimates the variance in the control samples, to carry out significance test on sgRNAs enrichment/depletion in the treatment samples. Read counts are assumed to follow a negative binomial distribution as motivated previously (Robinson and Smyth, 2007). Control samples are automatically identified by the algorithm through their sample name definition in the configuration file. The scripts automatically processes all control samples in the dataset (An arbitrary number of control samples ≥ 2 can be provided).

To run this script, type

```
AnalyzeControl.py
```

The output is stored in the /Analysis/Control folder.

script04: ListCandidateGuides

This script generates a spreadsheet for each treatment sample listing the normalized sgRNA read counts together with their fold enrichment / depletion and level of significance.

To run this script, type

```
ListCandidateGuides.py <sample>
```

where <sample> is the sample prefix (e.g. TreatmentA_R1). The output is stored in the /Analysis/Candidates_Lists folder.

script05: ListCandidateGenes

Based on the sgRNA enrichment/depletion in the previous script, this script generates a spreadsheet for each treatment sample listing the results from a gene ranking analysis, using one of three metrics (ES, a-RRA or STARS). Statistical significance is estimated using permutations. In addition, the script provides barplots indicating the on-target efficiency of the library (i.e. the number of significant sgRNAs per gene).

To run this script, type

```
ListCandidateGenes.py <sample>
```

where <sample> is the sample prefix (e.g. TreatmentA_R1). The output is stored in the /Analysis/Candidates_Lists folder.

script06: PlotSample

This script draws a scatterplot of normalized sgRNA log counts of the control average versus the specified treatment sample. sgRNA counts with significant enrichment/depletion are marked in green. Non-targeting sgRNAs are marked in purple.

To run this script, type

```
PlotSample.py <sample>
```

where <sample> is the sample prefix (e.g. TreatmentA_R1). The output is stored in the /Analysis/ScatterPlots folder.

script07: PlotReplicates

This script draws a scatterplot of normalized sgRNA log counts from two treatment replicates against each other. The script reports both Pearson and Spearman correlation coefficients and the significance of correlation (Figure 3).

To run this script, type

```
PlotReplicates.py <replicate1>
                  <replicate2>
```

where <replicate 1> and <replicate 2> are the sample prefixes (e.g. TreatmentA_R1 and TreatmentA_R2). The output is stored in the /Analysis/ScatterPlots folder.

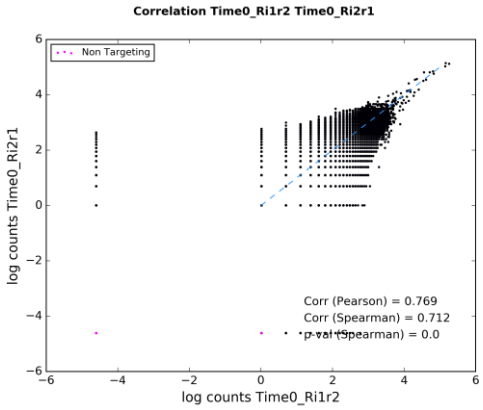

Figure 3: Example of a correlation scatterplot.

script08: PlotHeatmap

This script draws a heatmap of log sgRNA counts and performs a clustering analysis of all samples in the dataset using the heatmap.2 function in the R Bioconductor package. By default, the N (by default, N=100) sgRNAs displaying the highest variance across the dataset are taken into account (The number N is specified in the configuration file). Alternatively, for each sample the N sgRNAs with the highest counts are chosen and being combined in a joint list of sgRNAs which is taken as the basis for the clustering analysis.

To run this script, type

```
PlotHeatmap.py
```

The output is stored in the /Analysis/Heatmap folder.

4.1.3 Managing PinAPL-Py containers

When finished with the analysis, type **exit** to exit and close the container that you have been working on. You can use the **start** and **attach** commands (see 4.1.1) to resume the session at any point later.

You can work on multiple CRISPR screen datasets separately by preparing separate working directories (e.g. screen1 and screen2) and by following the instructions in Section 2 for both directories separately. **IMPORTANT**: when linking the PinAPL-Py container (Section 2, step 8) make sure to use separate container names (e.g. pinaplpy_screen1 and pinaplpy_screen2, respectively).

In case you want to detach a container while the analysis is still running, type **Ctrl+p Ctrl+q**

To get a list of all containers created, type **docker ps -a**

To delete a container (after it has been stopped or exited), type **docker rm <container name>**

To delete the PinAPL-Py image, first type **docker images**, then type **docker rmi <image_ID>** and enter (or copy/paste) the ID of the image shown from the "docker images" command.

Type **docker** to get a full list of available Docker commands and refer to Docker documentation for

more options.

4.2. Applications

4.2.1 Checking read quality and coverage

If you only wish to check read quality and coverage, perform steps 1-8 from Section 2, but do not launch PinAPL.py. Instead:

1. Execute

```
BuildLibraryIndex
```

NOTE: This command only needs to be done once!

2. Execute

```
AlignReads.py <sample>
```

Where <sample> is the prefix of your reads file in the /Data directory (e.g. “Control_R1”).

3: Execute

```
AnalyzeCounts.py <sample>
```

Refer to Section 3 for description of the results.

Repeat Steps 2&3 for all samples you wish to analyze.

4.2.2 Highlight individual genes in the scatterplot

If you wish to highlight a particular gene in the scatterplot (see Section 3), execute the following command:

```
PlotSample.py <sample> <gene>
```

where <sampe> is the prefix of your reads file (e.g. “ToxinA_R1”) and <gene> is your gene of interest (e.g. HBEGF). The output is stored in the /Analysis/ScatterPlots folder.

NOTE: It is recommended to first complete one full run of PinAPL-Py before executing this command.

5. Parameters in the configuration file

5.1. PROJECT PARAMETERS:

These parameters are most likely to require adjustment for your dataset.

`ScreenType: 'enrichment'/'depletion'`

This specifies the type of screen. Choose between 'enrichment' and 'depletion'

`LibFilename: 'GeCKOv2_library.tsv'`

This specifies the library filename. Make sure your library is saved as a tab-delimited text file ending in .tsv. Leave unchanged for GeCKO v2

`seq_5_end: 'TCTTGTGGAAAGGACGAAACACCG'`

This specifies the 5' adapter in front of the 20bp sgRNA region. Specification of the sequence is required for efficient alignment. Leave unchanged for GeCKO v2

`seq_3_end: 'GTTTTAGAGCTAGAAATAGCAAGTT'`

This specifies the 3' adapter in front of the 20bp sgRNA region. Specification of the sequence is required for efficient alignment. Leave unchanged for GeCKO v2

Figure 4: Read structure with adapters.

`CtrlPrefix: 'Control_'`

This specifies the prefix that defines your control samples.

`NonTargetPrefix: 'NonTargeting'`

This specifies the name of non-targeting sgRNAs in the library. Enter the entire name that is shared among all non-targeting sgRNA IDs. Leave at 'NonTargeting' if no non-targeting sgRNAs are present in the library.

`sgRNAsPerGene: 6`

Specifies the number of sgRNAs targeting a single gene. Note: This currently does not pertain to non-targeting sgRNAs and microRNAs!

5.2. ANALYSIS OPTIONS:

These parameters give a few options for analysis and processing of output.

`AlnOutput: 'Compress'/'Keep'/'Delete'`

This specifies how the raw alignment output (.SAM files) is to be processed after completion.

`keepCutReads: True/False`

This specifies if you would like to keep the file with the 5' adaptors removed. This usually would only be interesting for troubleshooting.

`VarEst: 'model'/'sample'`

This specifies the method of read count variance estimation in the control samples. Choose between 'model' for estimation from a negative binomial model or 'sample' for estimating from the raw sample variance.

`GeneMetric: 'ES'/'aRRA'/'STARS'`

Metric for gene ranking. Choose between 'aRRA' (Li et al. 2014), 'ES' (Subramanian et al. 2005), or STARS (Doench et al., 2016). 'ES' significantly increases computation time.

`scatter_annotate: True/False`

This specifies if the IDs of individual sgRNAs are to be displayed when running the PlotSamples.py script with specification of a gene of interest.

`ClusterBy: 'variance'/'model'`

This specifies whether the clustering analysis is to be based on the most variable sgRNAs ('variance') or the most enriched/depleted sgRNAs in individual samples ('counts').

`TopN: 100`

This specifies the number of most variable sgRNAs in the dataset to be taken into account for clustering analysis (if ClusterBy: 'variance'). In case ClusterBy is set to 'counts' it specifies the number of most enriched/depleted sgRNAs to be taken in each sample.

5.3. TECHNICAL PARAMETERS:

These represent more technical parameters.

`CutErrorTol: 0.25`

This defines the allowed error rate when identifying the 5' and 3' adapters. Refer to the cutadapt website to learn more.

`R_min: 10`

This specifies the minimal allowed read length after cutting the 5' adapter. Reads with length shorter than R_min after cutting the adaptor will be discarded. This is useful to sort out incomplete PCR products.

`L_bw: 11`

`N_bw: 1`

`i_bw: 'S,1,0.75'`

These are technical parameters for the alignment protocol. Refer to the Bowtie2 manual to learn more.

`Theta: 2`

This specifies the minimum required difference between primary and secondary alignment score in order to consider the read successfully matched. Reads with lower difference between primary and secondary alignment will be discarded. Increase Theta to increase the stringency of the alignment. Decrease Theta to increase sensitivity. Refer to the Bowtie2 manual to learn more about alignment scores.

`N0: 1000000`

This number specifies what read depth the counts are to be normalized to.

`max_q: 95`

This specifies the maximum quantile to be plotted in count distribution histograms.

`alpha: 0.01`

This specifies the critical level for single tests of sgRNA enrichment/depletion significance (FWER).

`pcorr: 'fdr_bh'`

This specifies the method for correction of the critical significance level in a multiple-testing situation. Refer to the statsmodels website to learn more.

`Np_ES/Np_aRRA/Np_STARS: 100`

This specifies the number of permutations to run to estimate p-values of gene ranking when using any of the gene metric options. Refer to the cited literature to learn more.

5.4. VISUALIZATION PARAMETERS:

These control some aspects of graphical display.

`delta_s: 0.01`

This specifies by how much normalized read counts are to be shifted before log transformation in the scatterplots produced by PlotSamples.py and PlotReplicates.py.

`delta_p: 1`

This specifies by how much normalized read counts are to be shifted before log transformation in the heatmap produced by PlotHeatmap.py.

`dpi: 300`

Resolution of PNG graphics.

`dotsize: 10`

Dot size in scatterplots.

5.5. DIRECTORIES

These specify the folders in the PinAPL-Py image. Do not change these settings!

```
WorkingDir: '/workingdir/'
DataDir: '/workingdir/Data/'
LibDir: '/workingdir/Library/'
IndexDir: '/workingdir/Library/Bowtie2_Index/'
ScriptsDir: '/opt/PinAPL-Py/'
AnalysisDir: '/workingdir/Analysis/'
AlignDir: '/workingdir/Alignments/'
QCDir: '/workingdir/Analysis/QC/'
bw2Dir: '/usr/bin/'
CutAdaptDir: '/root/.local/bin/'
STARSDir: '/opt/PinAPL-Py/STARS_mod/'
```

5.6. SCRIPT FILENAMES

These specify the names of the individual scripts in PinAPL-Py. Do not change these settings!

```
script00: 'BuildLibraryIndex'
script01: 'AlignReads'
script02: 'AnalyzeCounts'
script03: 'AnalyzeControl'
script04: 'ListCandidateGuides'
script05: 'ListCandidateGenes'
script06: 'PlotSample'
script07: 'PlotReplicates'
script08: 'PlotHeatmap'
```

6. Dependencies

The PinAPL-Py workflow relies on the following tools and packages. Note that all packages are listed for information only as they are included in the PinAPL-Py Docker image

- Cutadapt 1.11
- Bowtie2 2.2.8
- Python 2.7.6: numpy, yaml, scipy, statsmodels, pandas, matplotlib, joblib, multiprocessing
- R 3.2.3: gplots, heatmap.2

7. Troubleshooting

Unrecognized commands

Please type all docker commands into the terminal/Docker Quickstart Terminal. Do not copy/paste from the tutorial! Some special characters like hyphens might not be interpreted correctly by the terminal (although they look correct).

Problem connecting to the repository

If running into an error “connection reset by user” or similar when executing the docker pull command (Section 1, Step 1), this is likely caused by temporal inability to connect to the D. Wait a few minutes and keep trying.

Problem viewing tab-delimited files

The best way to open .tsv files is by using a spreadsheet processor like Excel. When you open the file from within Excel, an import wizard screen should pop up. Make sure to mark “Delimited” and “Tab” (on the next screen). Then the data should be correctly formatted into separate columns.

All other .txt files produced by PinAPL-Py are best viewed using Notepad++/TextEdit (Windows/macOS)

Out of memory

Large sequence data files (>= 10 mio reads per file) require considerable RAM (at least 8 GB). If PinAPL-Py crashes with the message “Killed”, your machine has insufficient RAM. Windows might not allow a single program to use all available RAM on your computer, in which case the program might crash even though there is sufficient RAM installed. Installing more RAM or switching to a different workstation might be required.

Path to working directory

Avoid spaces and special characters in the path of your working directory (except underscore). They might not be correctly recognized by docker. To be safe, create your working directory directly on your desktop.