# bhive Documentation

**Liguo Wang**

**Oct 13, 2020**

# Overview

Bhive is a python package for single cell RNA-seq data QC, analysis, and visualization.

# CHAPTER 1

# Installation

*bhive* is written in Python. Python3 is required to run all programs. Some programs also need R and R libraries to generate graphs.

## 1.1 Prerequisites

You need to install these tools if they are not available from your computer.

- Python3
- pip3
- R

## 1.2 R libraries required

- R library ggplot2
- R library cowplot
- R library pheatmap

**Note:** These R libraries will be automatically installed the first time they are used. Please manually install them if you encounter error like: Error in library(XYZ) : there is no package called 'XYZ'

## 1.3 Python packages required

- pandas
- numpy

- sklearn
- logomaker
- pysam

**Note:** Note: These Python packages will be automatically installed if you use pip3 to install bhive.

## 1.4 Install bhive

```
$ pip3 install bhive
or
$ pip3 install git+https://github.com/liguowang/bhive.git
```

## 1.5 Upgrade bhive

```
$ pip3 install bhive --upgrade
```

CHAPTER 2

# bhive Release history

## 2.1 Version 1.0.0

Initial release

CHAPTER 3

# Testing dataset

We use the same example dataset as used by 10X Genomics. Raw data (in BAM format) were downloaded from the NCBI Sequence Read Archive (SRA). The study was published in[1].

## 3.1 Download raw data

| File_name | GSM_accession | SRR_accession | File_size | Treat-ment | MD5 |
|---|---|---|---|---|---|
| C05.bam.1 | GSM3308718 | SRR7611046 | 70 Gb | normal | 97b87c87b539e69dad7dcb04e8f03132 |
| C07.bam.1 | GSM3308720 | SRR7611048 | 64 Gb | irradiated | 064669deb6be22e5f82fe58679f7e394 |

## 3.2 Convert BAM to FASTQ

Download `bamtofastq` from here. Convert BAM into FASTQ files.

```
$ bamtofastq C05.bam.1 normal_dat
$ bamtofastq C07.bam.1 irradiated_dat
```

After this step, you will get two subdirectories (`./normal_dat` and `./irradiated_dat`) under your current directory. And within `./normal_dat` and `./irradiated_dat`, there are subdirectories and fastq files, for example

```
$ cd ./normal_dat
$ tree
.
├── indepth_C05_MissingLibrary_1_HL5G3BBXX
```

(continues on next page)

[1] Ayyaz A, Kumar S, Sangiorgi B, Ghoshal B, Gosio J, Ouladan S, Fink M, Barutcu S, Trcka D, Shen J, Chan K, Wrana JL, Gregorieff A. Single-cell transcriptomes of the regenerating intestine reveal a revival stem cell. Nature. 2019 May;569(7754):121-125. doi: 10.1038/s41586-019-1154-y. Epub 2019 Apr 24. PMID: 31019301.

(continued from previous page)

```
│   ├── bamtofastq_S1_L003_I1_001.fastq.gz
│   ├── bamtofastq_S1_L003_I1_002.fastq.gz
│   ├── bamtofastq_S1_L003_R1_001.fastq.gz
│   ├── bamtofastq_S1_L003_R1_002.fastq.gz
│   ├── bamtofastq_S1_L003_R2_001.fastq.gz
│   ├── bamtofastq_S1_L003_R2_002.fastq.gz
│   ├── bamtofastq_S1_L004_I1_001.fastq.gz
│   ├── bamtofastq_S1_L004_I1_002.fastq.gz
│   ├── bamtofastq_S1_L004_I1_003.fastq.gz
│   ├── bamtofastq_S1_L004_I1_004.fastq.gz
│   ├── bamtofastq_S1_L004_I1_005.fastq.gz
│   ├── bamtofastq_S1_L004_I1_006.fastq.gz
│   ├── bamtofastq_S1_L004_R1_001.fastq.gz
│   ├── bamtofastq_S1_L004_R1_002.fastq.gz
│   ├── bamtofastq_S1_L004_R1_003.fastq.gz
│   ├── bamtofastq_S1_L004_R1_004.fastq.gz
│   ├── bamtofastq_S1_L004_R1_005.fastq.gz
│   ├── bamtofastq_S1_L004_R1_006.fastq.gz
│   ├── bamtofastq_S1_L004_R2_001.fastq.gz
│   ├── bamtofastq_S1_L004_R2_002.fastq.gz
│   ├── bamtofastq_S1_L004_R2_003.fastq.gz
│   ├── bamtofastq_S1_L004_R2_004.fastq.gz
│   ├── bamtofastq_S1_L004_R2_005.fastq.gz
│   └── bamtofastq_S1_L004_R2_006.fastq.gz
└── indepth_C05_MissingLibrary_1_HNNWNBBXX
    ├── bamtofastq_S1_L002_I1_001.fastq.gz
    ├── bamtofastq_S1_L002_I1_002.fastq.gz
    ├── bamtofastq_S1_L002_I1_003.fastq.gz
    ├── bamtofastq_S1_L002_I1_004.fastq.gz
    ├── bamtofastq_S1_L002_I1_005.fastq.gz
    ├── bamtofastq_S1_L002_R1_001.fastq.gz
    ├── bamtofastq_S1_L002_R1_002.fastq.gz
    ├── bamtofastq_S1_L002_R1_003.fastq.gz
    ├── bamtofastq_S1_L002_R1_004.fastq.gz
    ├── bamtofastq_S1_L002_R1_005.fastq.gz
    ├── bamtofastq_S1_L002_R2_001.fastq.gz
    ├── bamtofastq_S1_L002_R2_002.fastq.gz
    ├── bamtofastq_S1_L002_R2_003.fastq.gz
    ├── bamtofastq_S1_L002_R2_004.fastq.gz
    ├── bamtofastq_S1_L002_R2_005.fastq.gz
    ├── bamtofastq_S1_L003_I1_001.fastq.gz
    ├── bamtofastq_S1_L003_I1_002.fastq.gz
    ├── bamtofastq_S1_L003_R1_001.fastq.gz
    ├── bamtofastq_S1_L003_R1_002.fastq.gz
    ├── bamtofastq_S1_L003_R2_001.fastq.gz
    └── bamtofastq_S1_L003_R2_002.fastq.gz
```

## 3.3 Run CellRanger *count* workflow

Download `cellranger` and **Mouse reference dataset** from here

```
$ cellranger --version
cellranger 4.0.0
```

(continues on next page)

(continued from previous page)

```
# run cellranger for normal sample
$ cd ./normal_dat
$ cellranger count  --id=normal        --transcriptome=/XYZ/CellRanger/refdata-gex-
↪mm10-2020-A  --fastqs=./indepth_C05_MissingLibrary_1_HL5G3BBXX,./indepth_C05_
↪MissingLibrary_1_HNNWNBBXX

# run cellranger for irradiated sample
$ cd ./irradiated_dat
$ cellranger count  --id=irradiated    --transcriptome=/XYZ/CellRanger/refdata-gex-
↪mm10-2020-A  --fastqs=./indepth_C07_MissingLibrary_1_HL5G3BBXX,./indepth_C07_
↪MissingLibrary_1_HNNWNBBXX
```

After each `cellranger count` workflow is finished successfully. Subdirectories `normal` and `irradiated` will be created, which contain the cellranger outputs. For example,

```
$ cd normal
$ ls -F
_cmdline     _invocation  _mrosource     _perf                _tags       _vdrkill
_filelist    _jobmode    normal.mri.tgz  SC_RNA_COUNTER_CS/  _timestamp  _versions
_finalstate  _log        outs/           _sitecheck          _uuid
```

**Note:** Replace /XYZ/ with the actual path on your system.

## 3.4 Run CellRanger *aggr* workflow

First, make the `library.csv` file. This CSV file has two columns which define the **ID** and the location of the **molecule_info.h5** file from each run.

```
$ cat  library.csv

library_id,molecule_h5
normal,/ABC/normal_dat/normal/outs/molecule_info.h5
irradiated,/ABC/irradiated_dat/irradiated/outs/molecule_info.h5
```

**Note:** Replace /ABC/ with the actual path on your system.

Then, run `cellranger aggr` workflow. The `cellranger aggr` workflow aggregates outputs from multiple runs of the `cellranger count` workflow

```
$ cellranger aggr --id=aggr --csv=libraries.csv
```

After each `cellranger aggr` workflow is finished successfully. A subdirectory `aggr` will be created, which contain the cellranger outputs. For example,

```
$ cd aggr
$ ls -F
aggr.mri.tgz  _finalstate  _log        _perf                 _tags       _vdrkill
_cmdline      _invocation  _mrosource  SC_RNA_AGGREGATOR_CS/  _timestamp  _versions
_filelist     _jobmode    outs/       _sitecheck             _uuid
```

## 3.5 References

# CHAPTER 4

# BC_edit_matrix.py

## 4.1 Description

This program generates heatmaps to visualize the **positions** (X-axis), type of edits/corrections (Y-axis, such as "C" to "T") and **frequencies** (color) of error-corrected nucleotides in cell barcodes and UMIs.

## 4.2 Options

**--version** show program's version number and exit

**-h, --help** show this help message and exit

**-i IN_FILE, --infile=IN_FILE** Input file in BAM foramt.

**-o OUT_FILE, --outfile=OUT_FILE** The prefix of output files.

**--limit=READS_NUM** Number of alignments to process. default=none

**--cr-tag=CR_TAG** Tag of cellular barcode reported by the sequencer in BAM file. default='CR'

**--cb-tag=CB_TAG** Tag of error-corrected cellular barcode in BAM file. default='CB'

**--ur-tag=UR_TAG** Tag of UMI reported by the sequencer in BAM file. default='UR'

**--ub-tag=UB_TAG** Tag of error-corrected UMI in BAM file. default='UB'

**--cell-width=CELL_WIDTH** Points of cell width in the heatmap. default=15

**--cell-height=CELL_HEIGHT** Points of cell height in the heatmap. default=10

**--font-size=FONT_SIZE** Font size. If –display-num was set, fontsize_number = 0.8 * font_size. default=8

**--angle=COL_ANGLE** The angle (must be 0, 45, 90, 270, 315) of column text lables under the heatmap. default=45

**--text-color=TEXT_COLOR** The color of numbers in each cell. default=black

**--file-type=FILE_TYPE** The file type of heatmap. Choose one of 'pdf', 'png', 'tiff', 'bmp', 'jpeg'. default=pdf

**--verbose** If set, detailed running information is printed to screen.

**--no-num** If set, will not print numerical values to cells. default=False

## 4.3 Input file format

BAM file with the following tags:

- CB : cellular barcode sequence that is error-corrected
- CR : cellular barcode sequence as reported by the sequencer.
- UB : molecular barcode sequence that is error-corrected
- UR : molecular barcode sequence as reported by the sequencer.

## 4.4 Example (Visualize sample barcode)

```
$ python3 BC_edit_matrix.py -i normal_possorted_genome_bam.bam --limit 5000000 -o␣
↪output

2020-09-30 08:59:21 [INFO]  Reading BAM file "normal_possorted_genome_bam.bam" ...
2020-09-30 09:00:03 [INFO]  Total alignments processed: 5000000
2020-09-30 09:00:03 [INFO]  Number of alignmenets with <cell barcode> kept AS IS:␣
↪4876615
2020-09-30 09:00:03 [INFO]  Number of alignmenets wiht <cell barcode> edited: 47377
2020-09-30 09:00:03 [INFO]  Number of alignmenets with <cell barcode> missing: 76008
2020-09-30 09:00:03 [INFO]  Number of alignmenets with UMI kept AS IS: 4973597
2020-09-30 09:00:03 [INFO]  Number of alignmenets wiht UMI edited: 24842
2020-09-30 09:00:03 [INFO]  Number of alignmenets with UMI missing: 1561
2020-09-30 09:00:03 [INFO]  Writing cell barcode frequencies to "output.CB_freq.tsv"
2020-09-30 09:00:03 [INFO]  Writing UMI frequencies to "output.UMI_freq.tsv"
2020-09-30 09:00:04 [INFO]  Writing the nucleotide editing matrix (count) of cell␣
↪barcode to "output.CB_edits_count.csv"
2020-09-30 09:00:04 [INFO]  Writing the nucleotide editing matrix of molecular␣
↪barcode (UMI) to "output.UMI_edits_count.csv"
2020-09-30 09:00:04 [INFO]  Writing R code to "output.CB_edits_heatmap.r"
2020-09-30 09:00:04 [INFO]  Displayed numerical values on heatmap
2020-09-30 09:00:04 [INFO]  Numbers will be displayed on log2 scale
2020-09-30 09:00:04 [INFO]  Running R script file "output.CB_edits_heatmap.r"
Loading required package: Matrix
Loading required package: SPAtest
Loading required package: pheatmap
2020-09-30 09:00:07 [INFO]  Writing R code to "output.UMI_edits_heatmap.r"
2020-09-30 09:00:07 [INFO]  Displayed numerical values on heatmap
2020-09-30 09:00:07 [INFO]  Numbers will be displayed on log2 scale
2020-09-30 09:00:07 [INFO]  Running R script file "output.UMI_edits_heatmap.r"
Loading required package: Matrix
Loading required package: SPAtest
Loading required package: pheatmap
```

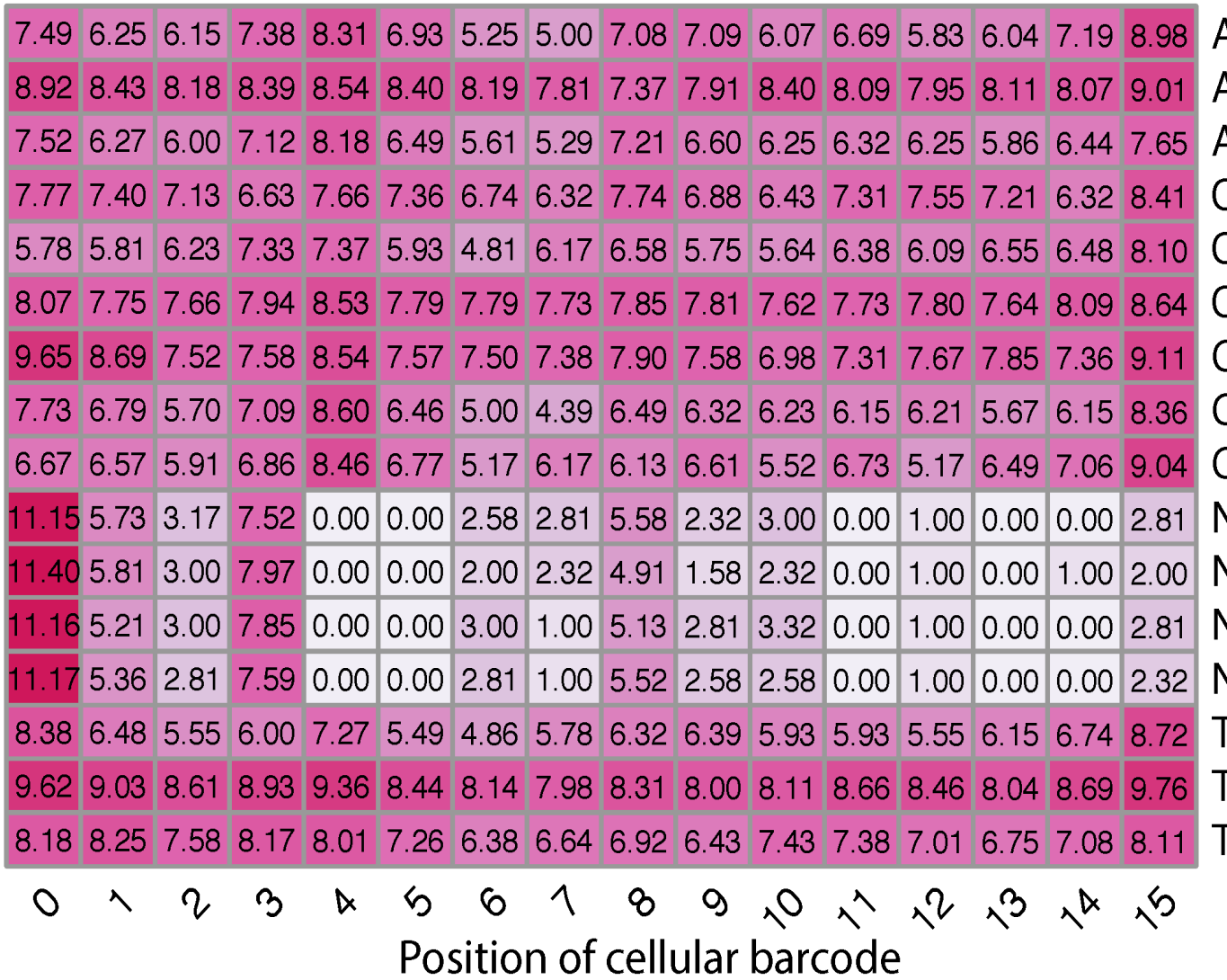
## 4.5 out put files

- output.CB_edits_count.csv : editing matrix of cellular barcodes in CSV format.
- output.CB_freq.tsv : corrected cell barcodes and their frequencies.
- output.CB_edits_heatmap.pdf : heatmap showing the positions, types and frequencies of nucleotides that have been corrected.
- output.CB_edits_heatmap.r : R script for the above heatmap.
-
- output.UMI_edits_count.csv : editing matrix of UMIs in CSV format.
- output.UMI_freq.tsv : corrected UMIs and their frequencies.
- output.UMI_edits_heatmap.pdf : heatmap showing the positions, types and frequencies of nucleotides that have been corrected.
- output.UMI_edits_heatmap.r : R script for the above heatmap.

Three files were generated.

- I1.count_matrix.csv
- I1.logo.pdf
- I1logo.mean_centered.pdf

output.CB_edits_heatmap.pdf




output.UMI_edits_heatmap.pdf

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 6.27 | 5.88 | 5.81 | 8.47 | 8.47 | 5.39 | 7.19 | 5.21 | 5.32 | 5.58 | A:C |
| 7.75 | 7.22 | 6.94 | 8.69 | 8.84 | 7.22 | 7.92 | 7.43 | 7.02 | 6.57 | A:G |
| 6.09 | 5.52 | 5.67 | 8.35 | 8.36 | 5.43 | 6.70 | 6.43 | 7.83 | 9.35 | A:T |
| 6.58 | 6.89 | 6.29 | 8.42 | 8.14 | 6.29 | 6.79 | 5.43 | 5.21 | 6.17 | C:A |
| 5.73 | 5.39 | 5.04 | 8.11 | 7.97 | 5.21 | 6.34 | 4.91 | 5.58 | 5.73 | C:G |
| 6.92 | 6.74 | 7.17 | 8.65 | 8.44 | 7.02 | 7.30 | 7.01 | 7.52 | 8.79 | C:T |
| 6.94 | 6.19 | 6.66 | 8.60 | 8.59 | 6.52 | 7.21 | 6.23 | 6.61 | 6.04 | G:A |
| 6.04 | 4.52 | 5.61 | 8.22 | 8.14 | 5.21 | 6.34 | 4.81 | 3.70 | 4.58 | G:C |
| 4.95 | 5.39 | 5.09 | 8.38 | 8.43 | 5.64 | 6.78 | 6.07 | 7.00 | 8.53 | G:T |
| 6.30 | 5.17 | 4.70 | 8.10 | 8.03 | 6.15 | 7.34 | 8.08 | 9.95 | 11.79 | T:A |
| 7.19 | 7.06 | 6.64 | 8.35 | 7.83 | 6.43 | 6.74 | 7.13 | 8.41 | 10.05 | T:C |
| 5.46 | 5.49 | 5.43 | 7.89 | 7.58 | 6.09 | 6.17 | 6.93 | 8.76 | 10.77 | T:G |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | |

Position of UMI

CHAPTER 5

# cell_calling.py

## 5.1 Description

Call cells from background.

- First, calculates the "read count" and "UMI count" for each barcode (i.e. cell barcode),and generates the barcode rank plot and density plot.
- Then, using the Bayesian Gaussian Mixture Model (BGMM) to classify barcodes into "cell-associated" and "background-associated".

**Options:**

**--version** show program's version number and exit

**-h, --help** show this help message and exit

**-i IN_FILE, --infile=IN_FILE** Input file in BAM foramt.

**-o OUT_FILE, --outfile=OUT_FILE** The prefix of output files.

**--cb-tag=CB_TAG** Tag of error-corrected cellular barcode in BAM file. default='CB'

**--umi-tag=UMI_TAG** Tag of error-corrected UMI in BAM file. default='UB'

**--cb-num=CB_LIMIT** Maximum cell barcodes (ranked by associated UMI frequency) analysed. default=100000

**--min-read-count=MIN_READS** The minimum number of reads to filter out cell barcode. default=200

**-r, --report** If set, generates report file for mixture models. default=False

**-s RANDOM_STATE, --seed=RANDOM_STATE** The seed used by the random number generator. default=0

**--prob-cut=PROBABILITY_CUTOFF** The probabiilty cutoff [0.5, 1] to assign cell barcode to the "cell" or the "background" component. default=0.5

**--verbose** If set, print detailed information for debugging.

## 5.2 Example

```
$ python3 cell_barcode.py -i possorted_genome_confident.bam -o output

2020-10-05 02:44:42 [INFO]  Top 100000 cell barcodes (ranked by associated UMI␣
↪frequency) will be analyzed.
2020-10-05 02:44:42 [INFO]  Only count UMIs for cell barcodes with more than 200␣
↪reads.
2020-10-05 02:44:42 [INFO]  Reading BAM file "possorted_genome_confident.bam". Count␣
↪reads for each cell barcode ...
2020-10-05 02:59:37 [INFO]  Total 98839578 alignments processed
2020-10-05 02:59:37 [INFO]  Filtering cell barcodes ...
2020-10-05 02:59:37 [INFO]  Total cell barcode: 856517
2020-10-05 02:59:37 [INFO]  Cell barcode with more than 200 reads: 8204
2020-10-05 02:59:37 [INFO]  Reading BAM file "possorted_genome_confident.bam". Count␣
↪UMIs for each cell barcode ...
2020-10-05 03:15:33 [INFO]  Total 90458232 alignments processed
2020-10-05 03:15:33 [INFO]  Writing cell barcodes' reads and UMI frequencies to
↪"output.Read_UMI_freq.tsv"
2020-10-05 03:15:33 [INFO]  Done.
2020-10-05 03:15:38 [INFO]  Read output.Read_UMI_freq.tsv to build Bayesian Gaussian␣
↪Mixture Model (BGMM)...
2020-10-05 03:15:38 [INFO]  Reading input file: "output.Read_UMI_freq.tsv"
2020-10-05 03:15:38 [INFO]     Total analyzed barcodes: 8204
2020-10-05 03:15:38 [INFO]  Build BGMM ...
2020-10-05 03:15:38 [INFO]  Building Bayesian Gaussian Mixture model for subject: UMI_
↪count ...
2020-10-05 03:15:40 [INFO]  Building Bayesian Gaussian Mixture model for subject:␣
↪read_count ...
2020-10-05 03:15:40 [INFO]  Classify cell barcode using the BGMM models ...
2020-10-05 03:15:40 [INFO]  Writing to "output.UMI_count_classification.txt" ...
2020-10-05 03:15:40 [INFO]  Writing to "output.read_count_classification.txt" ...
2020-10-05 03:15:40 [INFO]  Writing R script to "output.Read_UMI_freq.r"
```

## 5.3 Output files

1. **output.read_count_classification.txt** and **output.UMI_count_classification.txt**. Classify cell barcodes according to read (or UMI) count.

- Column-1 : The barcode sequence.
- Column-2 : Read/UMI count in log10 scale.
- Column-3 : The probability that this barcode belonging to "background" group.
- Column-4 : The probability that this barcode belonging to "cell" group.
- Column-5 : Assigned lable ("cell", "unknown" or "background").

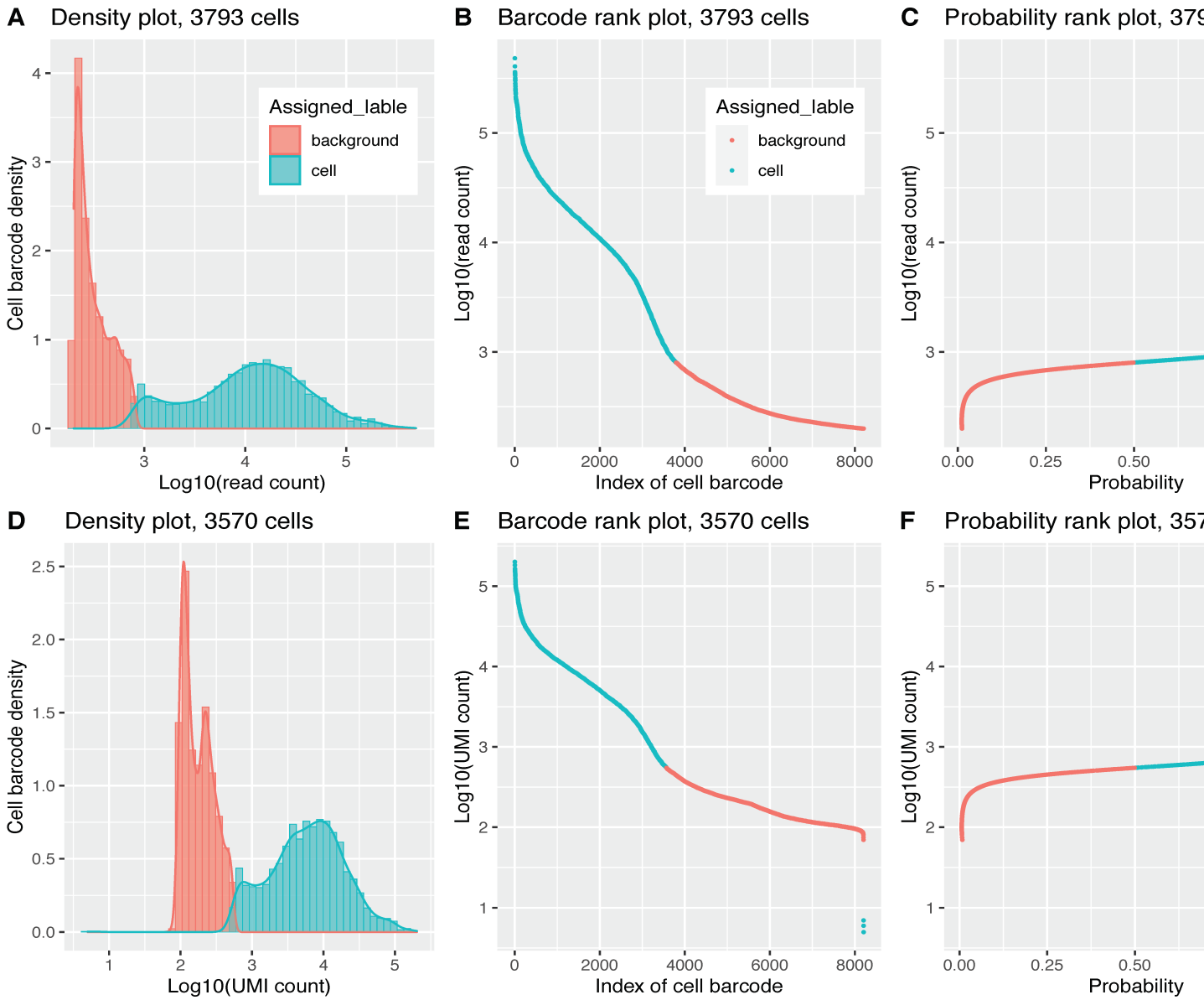
| Barcode | log10_count | background_prob | cell_prob | Lable |
|---|---|---|---|---|
| AACAACCCAGTTCTAG | 5.68488933 | 1.03E-73 | 1 | cell |
| CTATCCGCATGGATCT | 5.61115166 | 2.05E-70 | 1 | cell |
| AATCGTGTCTTTGATC | 5.55137314 | 8.49E-68 | 1 | cell |
| ... | | | | |
| GAACGTTTCGGCAGTC | 2.30103 | 0.98811857 | 0.01188143 | background |
| CATCCACAGGCGAAGG | 2.32633586 | 0.98862923 | 0.01137077 | background |
| TCACGGGGTGATATAG | 2.31175386 | 0.98836594 | 0.01163406 | background |

2. output.Read_UMI_freq.tsv

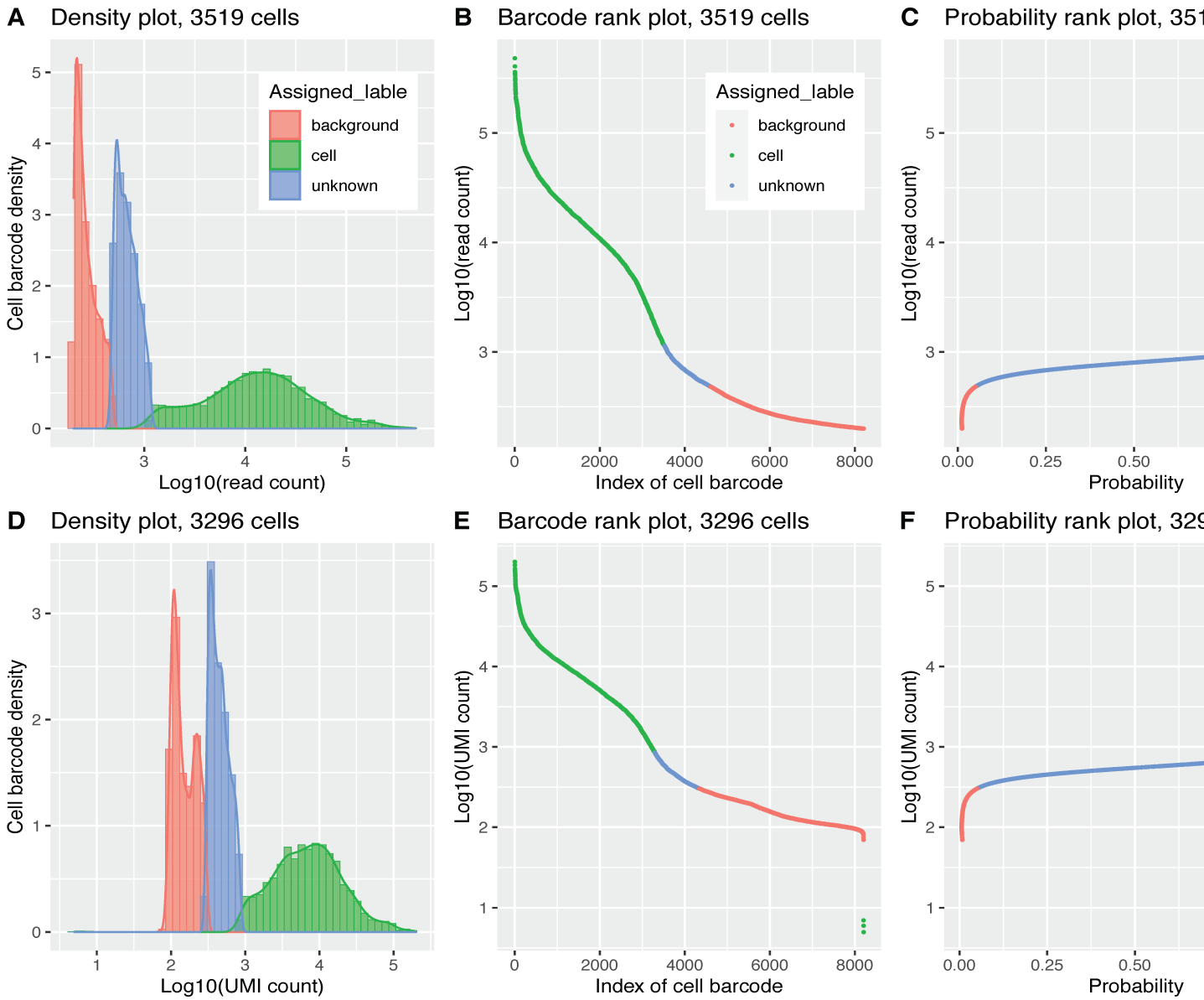
| Serial# | Cell_barcode | read_count | UMI_count |
|---|---|---|---|
| 1 | AACAACCCAGTTCTAG | 484049 | 202124 |
| 2 | CTATCCGCATGGATCT | 408462 | 184155 |
| 3 | AATCGTGTCTTTGATC | 355937 | 165832 |
| 4 | CTTCTCTGTTGTCCCT | 349352 | 159697 |
| 5 | GAGAAATCATGACACT | 362654 | 153856 |
| ... | | | |

3. output.Read_UMI_freq.r

Generate figure as below. Panels A, B, C were generated from **read count**, and panels D, E, F were generated from **UMI count**. (A) and (D) Density plots of cell barcodes that have been classified into **background** (red) and **cells** (blue). (B) and (E) Barcode rank plots of all cell barcodes. (C) and (F) Probability rank plots of all cell barcodes.




**Note:** All the cell barcodes were classified into **two** groups, because the default `--prob-cut` is 0.5. In this Scenario, each barcode was either classified into **cell-associated** group (when prob >= 0.5) or **background-associated** group (when prob < 0.5). If the `--prob-cut` is set to 0.95, cell barcodes will be classified into three groups: "cell-associated" (when cell_prob >= 0.95), "background-associated" (when background_prob >= 0.95) and "unknown" (cell_prob < 0.9 and background_prob < 0.95). The "unknown" group might contains cells with low RNA content. Please see figure below.


A
Density plot, 3519 cells
Assigned_lable
background
cell
unknown
Cell barcode density
Log10(read count)
B
Barcode rank plot, 3519 cells
Assigned_lable
background
cell
unknown
Log10(read count)
Index of cell barcode
C
Probability rank plot, 351
Log10(read count)
Probability
D
Density plot, 3296 cells
Cell barcode density
Log10(UMI count)
E
Barcode rank plot, 3296 cells
Log10(UMI count)
Index of cell barcode
F
Probability rank plot, 329
Log10(UMI count)
Probability

CHAPTER 6

# map_stat.py

## 6.1 Description

Report reads mapping statistics

**Options:**

**--version** show program's version number and exit

**-h, --help** show this help message and exit

**-i BAM_FILE, --infile=BAM_FILE** Input file in BAM foramt. Must have BAM alignment tags indicated below.

**--cb-tag=CB_TAG** BAM alignment tag. Used to indicate error-corrected cellular barcode. default='CB'

**--re-tag=RE_TAG** BAM alignment tag. Used to indicate the region type of the alignment (E = exonic, N = intronic, I = intergenic). default='RE'

**--tx-tag=TX_TAG** BAM alignment tag. Used to indicate reads aligned to the same strand as the annotated transcripts. default='TX'

**--an-tag=AN_TAG** BAM alignment tag. Used to indicate reads aligned to the antisense strand of the annotated transcripts. default='AN'

**--umi-tag=UMI_TAG** BAM alignment tag. Used to indicat the error-corrected UMI. default='UB'

**--xf-tag=XF_TAG** BAM alignment tag. Used to indicate reads confidently mapped to the feature. default='xf'

**--chrM-id=MIT_CONTIG_NAME** The name of mitochondrial chromosome in BAM file. default='chrM'

**--verbose** Logical to determine if detailed running information is printed to screen.

## 6.2 Example

```
$ python3 map_stat2.py -i normal_possorted_genome_bam.bam

2020-10-08 10:06:41 [INFO]  Reading BAM file "normal_possorted_genome_bam.bam" ...
2020-10-08 10:06:41 [INFO]  Processing "chr1" ...
2020-10-08 10:11:50 [INFO]  Processed 24729033 alignments mapped to: "chr1"
2020-10-08 10:12:16 [INFO]  Processing "chr10" ...
2020-10-08 10:17:57 [INFO]  Processed 26004376 alignments mapped to: "chr10"
2020-10-08 10:18:27 [INFO]  Processing "chr11" ...
2020-10-08 10:26:59 [INFO]  Processed 37558210 alignments mapped to: "chr11"
...
...

Total_alignments: 589060389
└--Confident_alignments: 443330914

Total_mapped_reads:     589060389
|--Non_confidently_mapped_reads:        145729475       (24.74%)
└--Confidently_mapped_reads:    443330914       (75.26%)
   |--Reads_with_PCR_duplicates:        327447641       (73.86%)
   └--Reads_no_PCR_duplicates:  115883273       (26.14%)

   |--Reads_map_to_forward(Waston)_strand:      259474203       (58.53%)
   └--Reads_map_to_Reverse(Crick)_strand:       183856711       (41.47%)

   |--Reads_map_to_sense_strand:        443330914       (100.00%)
   └--Reads_map_to_antisense_strand:    0       (0.00%)
   └--Other:    0       (0.00%)

   |--Reads_map_to_exons:       443330914       (100.00%)
   └--Reads_map_to_introns:     0       (0.00%)
   └--Reads_map_to_intergenic:  0       (0.00%)
   └--Other:    0       (0.00%)

   |--Reads_with_Error-Corrected_barcode:       437707874       (98.73%)
   └--Reads_no_Error-Corrected_barcode: 5623040 (1.27%)

   |--Reads_with_Error-Corrected_UMI:   443184634       (99.97%)
   └--Reads_no_Error-Corrected_UMI:     146280  (0.03%)

   |--Reads_map_to_mitochonrial_genome: 56744099        (12.80%)
   └--Reads_map_to_nuclear_genome:      386586815       (87.20%)

   |--Map_consecutively:        242755968       (54.76%)
   |--Map_with_clipping:        49473035        (11.16%)
   |--Map_with_splicing:        115086767       (25.96%)
   |--Map_with_splicing_and_clipping:   19346122        (4.36%)
   └--Others:   16669022        (3.76%)
```

**Note:** Except the header section, each row in a BAM file represents an *alignment*. One read can have multiple alignments.

CHAPTER 7

# seq_logo.py

## 7.1 Description

This program generates a DNA sequence logo from fasta or fastq format file. It is useful to visualize the nucleotide compositions of sample barcodes, cell barcodes and molecular barcodes (UMI).

## 7.2 Options

**--version** show program's version number and exit

**-h, --help** show this help message and exit

**-i IN_FILE, --infile=IN_FILE** Input DNA sequence file in FASTQ, FASTA or pure sequence format. All sequences must be the same length. This file can be plain text or compressed format (".gz", ".Z",".z",".bz", ".bz2", ".bzip2").

**-o OUT_FILE, --outfile=OUT_FILE** The prefix of output files.

**--iformat=IN_FORMAT** The format of input file. Must be 'fq' or 'fa'. defualt='fq'

**--oformat=OUT_FORMAT** The format of output logo file. Must be 'pdf', 'png' or 'svg'. defualt='pdf'

**-n MAX_SEQ, --nseq-limit=MAX_SEQ** Only process this many sequences and stop. default=none (generate logo fromALL sequences).

**--font-name=FONT_NAME** The font of logo characters. For a list of valid font names, run logomaker.list_font_names().default='sans'

**--stack-order=STACK_ORDER** Must be 'big_on_top', 'small_on_top', or 'fixed'. 'big_on_top' : nucleotide with the highestfrequency will be on the top; 'small_on_top' : nucleotide with the lowest frequency will be on thetop; 'fixed' : nucleotides from top to bottom are in the same order as characters appear in thedata frame. default='big_on_top'

**--flip-below** If set, characters below the X-axis (which correspond to negative values in the matrix)will be flipped upside down. default=False

**--shade-below=SHADE_BELOW** The amount of shading to use for characters drawn below the X-axis. 0 <= shade_below <= 1.Larger values correspond to more shading. default=0.0

**--fade-below=FADE_BELOW** The amount of fading to use for characters drawn below the X-axis. 0 <= shade_below <= 1. Larger values correspond to more fading. default=0.0

**--excludeN** If set, exclude all DNA sequences containing "N".

**--highlight-start=HI_START** Highlight logo from this position. Must be within [0, sequence_length-1].default=none (no highlight)

**--highlight-end=HI_END** Highlight logo to this position. Must be within [0, len(logo)-1].default=none (no highlight)

**--verbose** If set, print detailed information for debugging.

## 7.3 Input file format

FASTQ format

```
@K00316:386:HHMKGBBXY:2:1101:28595:1314 1:N:0:NGTTTACT
AGAGCCCTCTATTCGTATAAGTTTTCAT
+
AAFFFJJJJJJJJJJJJJJJJJJJJJJJ
@K00316:386:HHMKGBBXY:2:1101:29021:1314 1:N:0:NGTTTACT
CCGGTGATCTATGTGGATAGGTAATTGA
+
A<AFFF-F<J7<JAAJFA<FJFAFFJJ<
@K00316:386:HHMKGBBXY:2:1101:29143:1314 1:N:0:NGTTTACT
CTTCGGTGTCTTGCTCACGAACAGCTAT
+
AAFFFJJJJJJJJJJJJJJFJJJJJJJJ
...
```

FASTA format

```
>seq_1
AGAGCCCTCTATTCGTATAAGTTTTCAT
>seq_2
CCGGTGATCTATGTGGATAGGTAATTGA
>seq_3
CTTCGGTGTCTTGCTCACGAACAGCTAT
...
```

Sequence format

```
AGAGCCCTCTATTCGTATAAGTTTTCAT
CCGGTGATCTATGTGGATAGGTAATTGA
CTTCGGTGTCTTGCTCACGAACAGCTAT
...
```

## 7.4 Example (Visualize sample barcode)

After `cellranger mkfastq`, three fastq.gz files will be produced: I1, R1 and R2.

- **I1 fastq file** contains the 8 bp **sample barcode**. **sample barcode** is used to separate reads into different samples.
- **R1 fastq file** contains the 16bp **cell barcode** + 10 bp UMI. **Cell barcode** is used to assign reads/UMIs to different cells. UMI is used to remove PCR duplicates.
- **R2 fastq file** contains the real RNAseq reads.

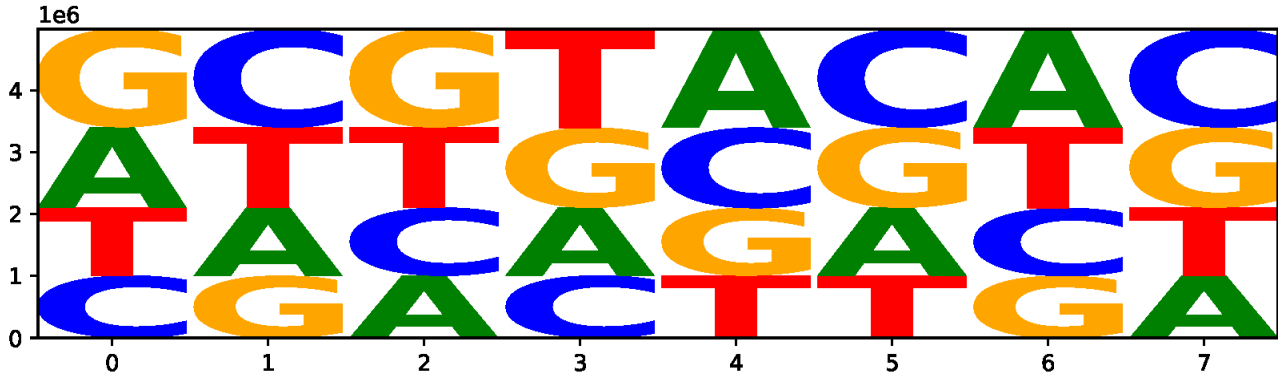
```
#exclude barcode with "N"
$python3 seq_logo.py -i ../normal_dat/indepth_C05_MissingLibrary_1_HL5G3BBXX/
↪bamtofastq_S1_L003_I1_001.fastq.gz --excludeN -n 5000000 -o I1

2020-09-29 01:54:41 [INFO]  Reading FASTQ file "../normal_dat/indepth_C05_
↪MissingLibrary_1_HL5G3BBXX/bamtofastq_S1_L003_I1_001.fastq.gz" ...
2020-09-29 01:55:12 [INFO]  5000000 sequences finished
2020-09-29 01:55:12 [INFO]  Make data frame from dict of dict ...
2020-09-29 01:55:12 [INFO]  Filling NA as zero ...
2020-09-29 01:55:12 [INFO]  Making logo ...
2020-09-29 01:55:12 [INFO]  'N' will be excluded.
2020-09-29 01:55:12 [INFO]  Mean-centered logo saved to "I1.logo_mean_centered.pdf".
2020-09-29 01:55:13 [INFO]  Logo saved to "I1.logo.pdf".
```

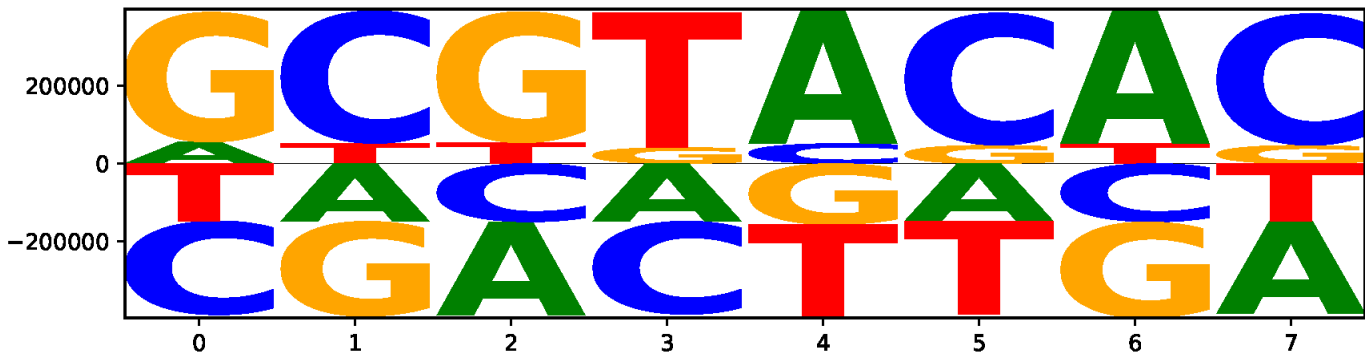
Three files were generated.

- I1.count_matrix.csv
- I1.logo.pdf
- I1logo.mean_centered.pdf

I1.logo.pdf




I1logo.mean_centered.pdf




## 7.5 Example (Visualize cell barcode and UMI)

Sequences in R1 fastq file contains cell barcode (first 16 nt) and UMI (last 10 nt)

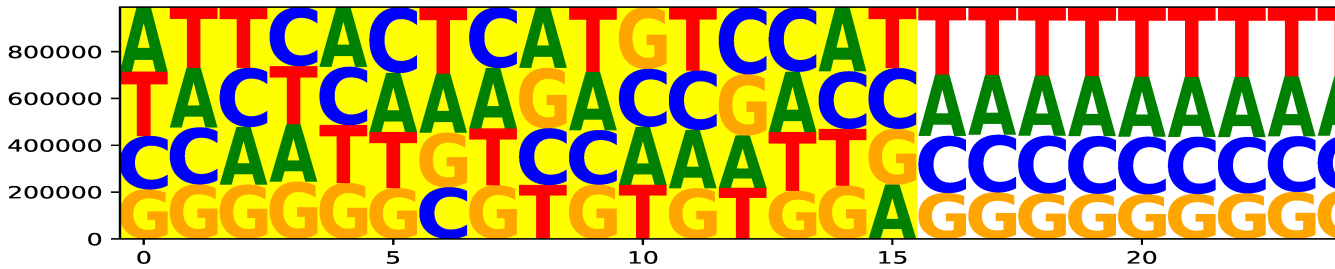
```
python3 seq_logo.py -i ../normal_dat/indepth_C05_MissingLibrary_1_HL5G3BBXX/
↪bamtofastq_S1_L004_I1_001.fastq.gz --excludeN -n 5000000 --highlight-start 0 --
↪highlight-end 15 -o R1
2020-09-29 03:49:09 [INFO]  Reading FASTQ file "../normal_dat/indepth_C05_
↪MissingLibrary_1_HL5G3BBXX/bamtofastq_S1_L004_R1_001.fastq.gz" ...
2020-09-29 03:49:53 [INFO]  5000000 sequences finished
2020-09-29 03:49:53 [INFO]  Make data frame from dict of dict ...
2020-09-29 03:49:53 [INFO]  Filling NA as zero ...
2020-09-29 03:49:53 [INFO]  Making logo ...
2020-09-29 03:49:53 [INFO]  'N' will be excluded.
2020-09-29 03:49:53 [INFO]  Mean-centered logo saved to "R1.logo_mean_centered.pdf".
2020-09-29 03:49:55 [INFO]  Highlight logo from 0 to 15
2020-09-29 03:49:55 [INFO]  Logo saved to "R1.logo.pdf".
2020-09-29 03:49:56 [INFO]  Highlight logo from 0 to 15
```

**Output**

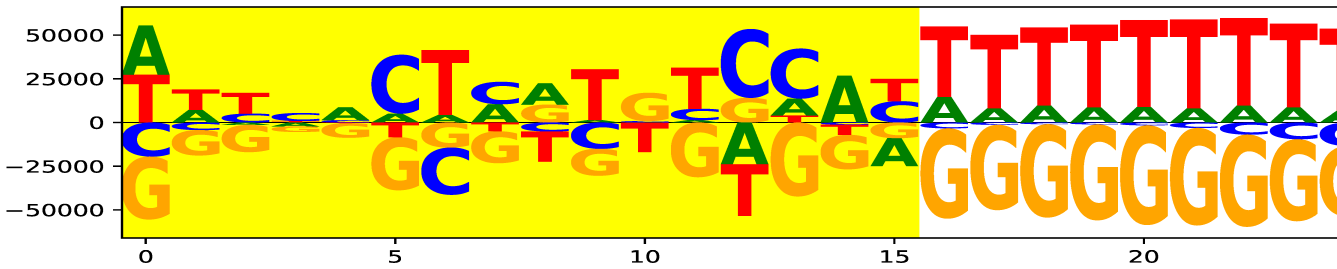
Three files were generated.

- R1.count_matrix.csv
- R1.logo.pdf
- R1logo.mean_centered.pdf

R1.logo.pdf (highlighted is the logo of cell barcode, un-highlighted is the logo of UMI)




R1.logo.mean_centered.pdf (highlighted is the logo of cell barcode, un-highlighted is the logo of UMI)


50000
25000
0
−25000
−50000
0
5
10
15
20

CHAPTER 8

# seq_qual.py

## 8.1 Description

This program generates heatmap from a FASTQ file to visualize the sequencing quality.

## 8.2 Options

**--version** show program's version number and exit

**-h, --help** show this help message and exit

**-i IN_FILE, --infile=IN_FILE** Input file in FASTQ (https://en.wikipedia.org/wiki/FASTQ_format#) format.

**-o OUT_FILE, --outfile=OUT_FILE** The prefix of output files.

**-n MAX_SEQ, --nseq-limit=MAX_SEQ** Only process this many sequences and stop. default=none (generate logo from ALL sequences).

**--cell-width=CELL_WIDTH** Cell width (in points) of the heatmap. default=12

**--cell-height=CELL_HEIGHT** Cell height (in points) of the heatmap. default=10

**--font-size=FONT_SIZE** Font size in points. If –display-num was set, fontsize_number = 0.8 * font_size. default=6

**--angle=COL_ANGLE** The angle (must be 0, 45, 90, 270, 315) of column text lables under the heatmap. default=45

**--text-color=TEXT_COLOR** The color of numbers in each cell. default=black

**--file-type=FILE_TYPE** The file type of heatmap. Choose one of 'pdf', 'png', 'tiff', 'bmp', 'jpeg'. default=pdf

**--no-num** if set, will not print numerical values to cells. default=False

**--verbose** If set, will produce detailed information for debugging.

## 8.3 Example

```
$ python3 seq_qual.py -i  ../normal_dat/indepth_C05_MissingLibrary_1_HL5G3BBXX/
↪bamtofastq_S1_L004_R1_001.fastq.gz -n  5000000 -o R1_qual

2020-09-29 04:34:40 [INFO]  Reading FASTQ file "../normal_dat/indepth_C05_
↪MissingLibrary_1_HL5G3BBXX/bamtofastq_S1_L004_R1_001.fastq.gz" ...
2020-09-29 04:35:30 [INFO]  5000000 quality sequences finished
2020-09-29 04:35:30 [INFO]  Make data frame from dict of dict ...
2020-09-29 04:35:30 [INFO]  Filling NA as zero ...
2020-09-29 04:35:30 [INFO]  Writing R code to "R1_qual.qual_heatmap.r"
2020-09-29 04:35:30 [INFO]  Displayed numerical values on heatmap
2020-09-29 04:35:30 [INFO]  Running R script file "R1_qual.qual_heatmap.r"
Loading required package: Matrix
Loading required package: SPAtest
Loading required package: pheatmap
```

## 8.4 Output files

- R1_qual.qual_count.csv
- R1_qual.qual_heatmap.pdf
- R1_qual.qual_heatmap.r
- R1_qual.qual_percent.csv

R1_qual.qual_heatmap.pdf

# CHAPTER 9

# LICENSE

bhive is distributed under **The MIT License**

# CHAPTER 10

# Reference

Unpublished.