fba

Release 0.0.12-49-ga1eca06-dirty

Jialei Duan

Mar 30, 2023

CONTENTS

1 Installation **3**

2 User Guide **5**
- 2.1 What is `fba`? . . . 5
- 2.2 Workflow . . . 5
- 2.3 Usage . . . 5

3 Tutorials **9**
- 3.1 CRISPR screening . . . 9
  - 3.1.1 10k A375 Cells Transduced with (1) Non-Target and (1) Target sgRNA . . . 9
  - 3.1.2 CROP-seq; 1:1:1 Mixture of DNMT3B, MBD1, and TET2 Knockout Cell Lines (HEK293T) . . . 22
  - 3.1.3 Direct-capture Perturb-seq; CRISPRi-based Screen of Unfolded Protein Response (UPR) Using 3' sgRNA-CR1[cs1] . . . 35
- 3.2 Cell surface protein labeling . . . 44
  - 3.2.1 CITE-seq; 8k Cord Blood Mononuclear Cells with 13 Antibodies . . . 44
  - 3.2.2 ASAP-seq; Multiplexed CRISPR Perturbations in Primary T Cells . . . 49
  - 3.2.3 1k Human PBMCs Stained with a Panel of TotalSeq B Antibodies . . . 64
- 3.3 ECCITE-seq . . . 70
  - 3.3.1 6k Single-cell Multimodal Readout of NIH-3T3, MyLa, Sez4 and PBMCs . . . 70
- 3.4 PHAGE-ATAC . . . 85
  - 3.4.1 PHAGE-ATAC; Anti-CD8 Phage Hashing Single-cell ATAC-seq Using CD8 T Cells from Four Human Donors . . . 85
- 3.5 CellPlex . . . 98
  - 3.5.1 10k 1:1 Mixture of Raji and Jurkat Cells Multiplexed . . . 98
  - 3.5.2 30k Mouse E18 Combined Cortex, Hippocampus and Subventricular Zone Nuclei Multiplexed 110
- 3.6 Cell hashing . . . 115
  - 3.6.1 Peripheral Blood Mononuclear Cells with 8 Antibodies . . . 115
- 3.7 MULTI-seq . . . 124
  - 3.7.1 15k HEK293 and 40k HMECs Multiplexed by Lipid- and Cholesterol-tagged Indices . . . 124
- 3.8 Targeted transcript enrichment . . . 132
  - 3.8.1 Hodgkin's Lymphoma, Dissociated Tumor: Targeted, Gene Signature Panel . . . 132
- 3.9 Pseudo-bulk . . . 136
  - 3.9.1 10k A375 Cells Transduced with (1) Non-Target and (1) Target sgRNA . . . 136
  - 3.9.2 10k 1:1 Mixture of Raji and Jurkat Cells Multiplexed . . . 139

4 Changelog **145**
- 4.1 **0.0.13 (Jan 1 2023)** . . . 145
- 4.2 **0.0.12 (Mar 9 2022)** . . . 145
- 4.3 **0.0.11 (Jun 17 2021)** . . . 145

5 Acknowledgements **147**

6 Quickstart **149**
6.1 Installation 149
6.2 Usage 149

7 Citation **151**

`fba`: a flexible and streamlined package for single-cell feature barcoding assays.

CHAPTER

ONE

INSTALLATION

`fba` can be installed with pip (stable version on PyPI):

```
$ pip install fba
```

Alternatively, if you're using Conda, you can run:

```
$ conda install -c bioconda fba
```

Or, you can grab the latest source code from GitHub:

```
$ pip install git+https://github.com/jlduan/fba.git
```

CHAPTER

TWO

USER GUIDE

2.1 What is fba?

fba is a flexible and streamlined toolbox for quality control, quantification, demultiplexing of various single-cell feature barcoding assays. It can be applied to customized feature barcoding specifications, including different CRISPR constructs or targeted enriched transcripts. fba allows users to customize a wide range of parameters for the quantification and demultiplexing process. fba also has a user-friendly quality control module, which is helpful in troubleshooting feature barcoding experiments.

2.2 Workflow

2.3 Usage

```
$ fba

usage: fba [-h]  ...

Tools for single-cell feature barcoding analysis

optional arguments:
-h, --help         show this help message and exit

functions:

   extract         extract cell and feature barcodes
   map             map enriched transcripts
   filter          filter extracted barcodes
   count           count feature barcodes per cell
   demultiplex     demultiplex cells based on feature abundance
   qc              quality control of feature barcoding assay
   kallisto_wrapper
                   deploy kallisto/bustools for feature barcoding
                   quantification
```

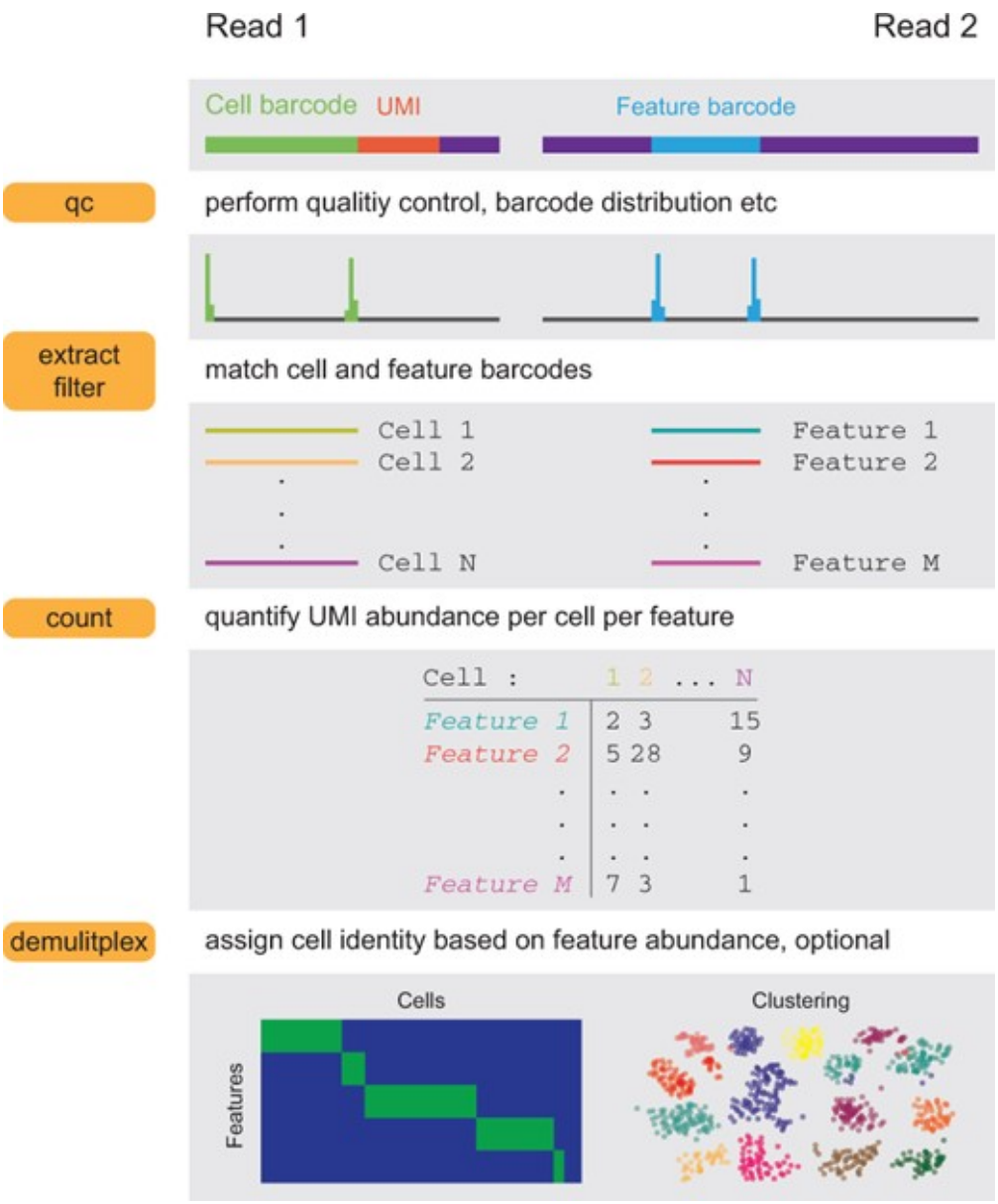


Fig. 1: The workflow of FBA: a flexible and streamlined package for single-cell feature barcoding assays.

- **extract**: extract cell and feature barcodes from paired fastq files. For single cell assays, read 1 typically contains cell partitioning and UMI information, while read 2 contains feature information.
- **map**: quantify enriched transcripts (through hybridization or PCR amplification) from parent single cell libraries. Read 1 contains cell partitioning and UMI information, while read 2 contains transcribed regions of enriched/targeted transcripts of interest. BWA (Li, H. 2013) or Bowtie2 (Langmead, B., et al. 2012) is used for read 2 alignment. The quantification (UMI deduplication) of enriched/targeted transcripts is powered by UMI-tools (Smith, T., et al. 2017).
- **filter**: filter extracted cell and feature barcodes (output of `extract` or `qc`). Additional fragment filter/selection can be applied through `-cb_seq` and/or `-fb_seq`.
- **count**: count UMIs per feature per cell (UMI deduplication), powered by UMI-tools (Smith, T., et al. 2017). The output of `extract` or `filter` is taken as input.
- **demultiplex**: demultiplex cells based on the abundance of features (matrix generated by `count` as input).
- **qc**: generate diagnostic information. If `-1` is omitted, bulk mode is enabled and only read 2 will be analyzed.
- **kallisto_wrapper**: deploy kallisto/bustools for feature barcoding quantification (just a wrapper) (Bray, N.L., et al. 2016).

CHAPTER

THREE

TUTORIALS

3.1 CRISPR screening

3.1.1 10k A375 Cells Transduced with (1) Non-Target and (1) Target sgRNA

Dataset: 10k A375 Cells Transduced with (1) Non-Target and (1) Target sgRNA, Dual Indexed

The detailed description of this dataset can be found here.

Preparation

Download fastq files.

```
$ wget https://cg.10xgenomics.com/samples/cell-exp/4.0.0/SC3_v3_NextGem_DI_CRISPR_10K/
↪SC3_v3_NextGem_DI_CRISPR_10K_fastqs.tar

$ tar xvf SC3_v3_NextGem_DI_PBMC_CSP_1K/SC3_v3_NextGem_DI_PBMC_CSP_1K_fastqs.tar
```

Combine reads of different lanes.

```
$ cat SC3_v3_NextGem_DI_CRISPR_10K_fastqs/SC3_v3_NextGem_DI_CRISPR_10K_crispr_fastqs/SC3_
↪v3_NextGem_DI_CRISPR_10K_crispr_S1_L00?_R1_001.fastq.gz > SC3_v3_NextGem_DI_CRISPR_10K_
↪crispr_S1_combined_R1_001.fastq.gz

$ cat SC3_v3_NextGem_DI_CRISPR_10K_fastqs/SC3_v3_NextGem_DI_CRISPR_10K_crispr_fastqs/SC3_
↪v3_NextGem_DI_CRISPR_10K_crispr_S1_L00?_R2_001.fastq.gz > SC3_v3_NextGem_DI_CRISPR_10K_
↪crispr_S1_combined_R2_001.fastq.gz
```

Download cell barcode info. These are the cell-associated barcodes in this single cell RNA-Seq library.

```
$ wget https://cf.10xgenomics.com/samples/cell-exp/4.0.0/SC3_v3_NextGem_DI_CRISPR_10K/
↪SC3_v3_NextGem_DI_CRISPR_10K_filtered_feature_bc_matrix.tar.gz

$ tar zxvf SC3_v3_NextGem_DI_CRISPR_10K_filtered_feature_bc_matrix.tar.gz
```

Inspect cell barcodes.

```
$ gzip -dc filtered_feature_bc_matrix/barcodes.tsv.gz | head

AAACCCACACATAGCT-1
AAACCCACATCATGAC-1
AAACCCAGTCATCGGC-1
AAACCCAGTCGCACAC-1
AAACCCAGTGCAACGA-1
AAACCCATCAAGCGTT-1
AAACCCATCAGATGCT-1
AAACCCATCATCTACT-1
AAACCCATCCTTATGT-1
AAACCCATCTCGGCTT-1
```

Prepare feature barcodes.

```
$ wget https://cf.10xgenomics.com/samples/cell-exp/4.0.0/SC3_v3_NextGem_DI_CRISPR_10K/
↪SC3_v3_NextGem_DI_CRISPR_10K_feature_ref.csv
```

Inspect feature barcode info.

```
$ cat SC3_v3_NextGem_DI_CRISPR_10K_feature_ref.csv

id,name,read,pattern,sequence,feature_type,target_gene_id,target_gene_name
RAB1A-2,RAB1A-2,R2,(BC)GTTTAAGAGCTAAGCTGGAA,GCCGGCGAACCAGGAAATA,CRISPR Guide Capture,
↪ENSG00000138069,RAB1A
NON_TARGET-1,NON_TARGET-1,R2,(BC)GTTTAAGAGCTAAGCTGGAA,AACGTGCTGACGATGCGGGC,CRISPR Guide␣
↪Capture,Non-Targeting,Non-Targeting
```

Clean up.

```
$ cut -d ',' -f1,5 SC3_v3_NextGem_DI_CRISPR_10K_feature_ref.csv | sed 's/,/\t/g' | tail -
↪2 > SC3_v3_NextGem_DI_CRISPR_10K_feature_ref.tsv

$ cat SC3_v3_NextGem_DI_CRISPR_10K_feature_ref.tsv

RAB1A-2 GCCGGCGAACCAGGAAATA
NON_TARGET-1    AACGTGCTGACGATGCGGGC
```

QC

The first 20,000 read pairs are sampled (set by `-n`, default `100,000`) for quality control. The `-t` option can be used to set the number of threads. By default, diagnostic results and plots are generated in the `qc` directory (set by `--output_directory`), and the full length of read 1 and read 2 are searched against reference cell and feature barcodes, respectively. The per base content of both read pairs and the distribution of matched barcode positions are summarized. Use `-r1_c` and/or `-r2_c` to limit the search range, and `-cb_n` and/or `-fb_n` to set the mismatch tolerance for cell and/or feature barcode matching (default `3`).

```
$ fba qc \
    -1 SC3_v3_NextGem_DI_CRISPR_10K_crispr_S1_combined_R1_001.fastq.gz \
    -2 SC3_v3_NextGem_DI_CRISPR_10K_crispr_S1_combined_R2_001.fastq.gz \
    -w filtered_feature_bc_matrix/barcodes.tsv.gz \
    -f SC3_v3_NextGem_DI_CRISPR_10K_feature_ref.tsv \
    -r1_c 0,16 \
    -n 20000
```

This library was constructed using the Chromium Next GEM Single Cell 3 Reagent Kits v3.1 (Dual Index) with Feature Barcode technology for CRISPR Screening and sequenced on an Illumina NovaSeq 6000. The first 16 bases of read 1 represent cell barcodes, and the following 12 bases represent UMIs. The base content plot indicates that the GC content of cell barcodes is evenly distributed. However, there is a slight T-enrichment in the UMIs.

Regarding read 2, the per base content analysis indicates that the first 31 bases are consistent and easily readable. These bases correspond to the Template Switch Oligo (TSO) sequence used in library construction. From base 32 onward, we have observed two distinct genotypes in the sampled reads.

The detailed `qc` results are stored in the `feature_barcoding_output.tsv.gz` file. The `matching_pos` columns indicate the matched positions on reads, while the `matching_description` columns indicate mismatches in the format of substitutions:insertions:deletions.

```
$ gzip -dc qc/feature_barcoding_output.tsv.gz | head

read1_seq       cell_barcode    cb_matching_pos cb_matching_description read2_seq      ␣
↪feature_barcode fb_matching_pos fb_matching_description
CNCCACACACGTGTTAatgagtactagc    CCTCACACACGTAGTT        0:15    2:0:1   ␣
↪AAGCAGTGGTATCAACGCAGAGTACATGGGATAGGTTTGGTCCTAGCCTTTCTATTAGCTCTTAGTAAGATTACACATGCAAGCATCCCC␣
↪   no_match        NA      NA
GNCGCGATCAGCATTActttttgtcaccc    GTCGCGAAGAGCATTA        0:16    3:0:0   ␣
↪AAGCAGTGGTATCAACGCAGAGTACATGGGGACTGTTGCTGGTGTGTACTTGCTAAGGTTTATGTCAGTTCAAGATTATAAGCCCCCCAG␣
↪   no_match        NA      NA
TNGGAAGGTAAGTGTAatcgagggaaca    TGGGAAGCAAAGTGTA        0:16    3:0:0   ␣
↪AAGCAGTGGTATCAACGCAGAGTACATGGGGGCCGGCGAACCAGGAAATAGTTTAAGAGCTAAGCTGGAAACAGCATAGCAAGTTTAAAT␣
↪   RAB1A-2_GCCGGCGAACCAGGAAATAG    31:51   0:0:0
CNCCCAAGTCGATAGGgagcgcaagcat    CCCAACTCACGATAGG        2:16    1:0:2   ␣
↪AAGCAGTGGTATCAACGCAGAGTACATGGGGGCCGGCGAACCAGGAAATAGTTTAAGAGCTAAGCTGGAAACAGCATAGCAAGTTTAAAT␣
↪   RAB1A-2_GCCGGCGAACCAGGAAATAG    31:51   0:0:0
CNCACTGCAAACGGTGggcgtaaatgag    CTCACTGGTAACGGTG        0:16    3:0:0   ␣
↪AAGCAGTGGTATCAACGCAGAGTACATGGGGGCCGGCGAACCAGGAAATAGTTTAAGAGCTAAGCTGGAAACAGCATAGCAAGTTTAAAT␣
↪   RAB1A-2_GCCGGCGAACCAGGAAATAG    31:51   0:0:0
ANCATCACAGGCGCTTgtcccactatat    AGCATCAGTGGCGCTT        0:16    3:0:0   ␣
↪AAGCAGTGGTATCAACGCAGAGTACATGGGGGCCGGCGAACCAGGAAATAGTTTAAGAGCTAAGCTGGAAACAGCATAGCAAGTTTAAAT␣
↪   RAB1A-2_GCCGGCGAACCAGGAAATAG    31:51   0:0:0
ANACGAACACTTTCATccaaaagaagtt    AAACGAAGTCTTTCAT        0:16    3:0:0   ␣
↪AAGCAGTGGTATCAACGCAGAGTACATGGGGGCCGGCGAACCAGGAAATAGTTTAAGAGCTAAGCTGGAAACAGCATAGCAAGTTTAAAT␣
↪   RAB1A-2_GCCGGCGAACCAGGAAATAG    31:51   0:0:0
ANCAACCAGTATCGTTgaaatcctggta    AACAACCTCTATCGTT        0:16    3:0:0   ␣
↪AAGCAGTGGTATCAACGCAGAGTACATGGGGAACGTGCTGACGATGCGGGCGTTTAAGAGCTAAGCTGGAAACAGCATAGCAAGTTTAAA␣
```

(continues on next page)

(continued from previous page)

```
↪   NON_TARGET-1_AACGTGCTGACGATGCGGGC       31:51   0:0:0
GNAGCCCGTACCACATgggcccagtatg    GAAGCCCCAACCACAT        0:16    3:0:0  ␣
↪AAGCAGTGGTATCAACGCAGAGTACATGGGGGCCGGCGAACCAGGAAATAGTTTAAGAGCTAAGCTGGAAACAGCATAGCAAGTTTAAAT␣
↪   RAB1A-2_GCCGGCGAACCAGGAAATAG    31:51   0:0:0
```

Barcode extraction

Although the length of the RAB1A-1 and RAB1A-2 feature barcodes differ by one base, they both start at the same position on read 2. To enable accurate feature barcode identification, we will include an extra downstream base (G) for the RAB1A-2 feature barcode to make their lengths equal.

```
$ cat SC3_v3_NextGem_DI_CRISPR_10K_feature_ref_edited.tsv

RAB1A-2 GCCGGCGAACCAGGAAATAG
NON_TARGET-1    AACGTGCTGACGATGCGGGC
```

The search ranges for barcode matching are set to `0,16` on read 1 and `31,51` on read 2. We allow for two mismatches for both cell and feature barcodes using the parameters `-cb_m` and `-cf_m`.

```
$ fba extract \
    -1 SC3_v3_NextGem_DI_CRISPR_10K_crispr_S1_combined_R1_001.fastq.gz \
    -2 SC3_v3_NextGem_DI_CRISPR_10K_crispr_S1_combined_R2_001.fastq.gz \
    -w filtered_feature_bc_matrix/barcodes.tsv.gz \
    -f SC3_v3_NextGem_DI_CRISPR_10K_feature_ref_edited.tsv \
    -o feature_barcoding_output.tsv.gz \
    -r1_c 0,16 \
    -r2_c 31,51 \
    -cb_m 2 \
    -fb_m 2
```

Preview of result.

```
$ gzip -dc feature_barcoding_output.tsv.gz  | head

read1_seq       cell_barcode    cb_num_mismatches       read2_seq       feature_barcode␣
↪fb_num_mismatches
GGCAGTCTCCGTTACTtatccagccttc    GGCAGTCTCGGTAACT        2       ␣
↪aagcagtggtatcaacgcagagtacatggggGCCGGCGAACCAGGAAATAGtttaagagctaagctggaaacagcatagcaagtttaaat␣
↪   RAB1A-2_GCCGGCGAACCAGGAAATAG     0
TTACGTTGTGAATCGGgtggggctcttc    TTACGTTCAGAATCGG        2       ␣
↪aagcagtggtatcaacgcagagtacatggggAACGTGCTGACGATGCGGGCgtttaagagctaagctggaaacagcatagcaagtttaaa␣
↪   NON_TARGET-1_AACGTGCTGACGATGCGGGC        0
TCGGGCAAGGATTGGTttctactcggaa    TCGGGCATCGATTGGT        2       ␣
↪aagcagtggtatcaacgcagagtacatgggaACGTGCTGACGATGCGGGCGtttaagagctaagctggaaacagcatagcaagtttaaat␣
↪   NON_TARGET-1_AACGTGCTGACGATGCGGGC        2
ACAACCACACATCTAGcggcatcatact    ACAACCAGTCATCTAG        2       ␣
↪aagcagtggtatcaacgcagagtacatggggCCGGCGAACCAGGAAATAGTttaagagctaagctggaaacagcatagcaagtttaaata␣
↪   RAB1A-2_GCCGGCGAACCAGGAAATAG     2
```

(continues on next page)

(continued from previous page)

```
AGACTCAAGTGCTAGAacagaactggtg    AGACTCATCTGCTAGA        2      ␣
↪aagcagtggtatcaacgcagagtacatggggAACGTGCTGACGATGCGGGCgtttaagagctaagctggaaacagcatagcaagtttaaa␣
↪   NON_TARGET-1_AACGTGCTGACGATGCGGGC        0
GAGTTGTTCGAACATTctgcccgacgtc    GAGTTGTAGGAACATT        2      ␣
↪aagcagtggtatcaacgcagagtacatggggAACGTGCTGACGATGCGGGCgtttaagagctaagctggaaacagcatagcaagtttaaa␣
↪   NON_TARGET-1_AACGTGCTGACGATGCGGGC        0
AGACTCAGTGGCACAAtgtcagaattca    AGACTCACAGGCACAA        2      ␣
↪aagcagtggtatcaacgcagagtacatggggGCCGGCGAACCAGGAAATAGtttaagagctaagctggaaacagcatagcaagtttaaat␣
↪   RAB1A-2_GCCGGCGAACCAGGAAATAG     0
TGCACGGAGGATAACCcgtgcacgtaca    TGCACGGTCGATAACC        2      ␣
↪aagcagtggtatcaacgcagagtacatggggGCCGGCGAACCAGGAAATAGtttaagagctaagctggaaacagcatagcaagtttaaat␣
↪   RAB1A-2_GCCGGCGAACCAGGAAATAG     0
CGTAGTAGTAACACGGaagagggaactg    CGTAGTAGTAACGCGA        2      ␣
↪aagcagtggtatcaacgcagagtacatggggAACGTGCTGACGATGCGGGCgtttaagagctaagctggaaacagcatagcaagtttaaa␣
↪   NON_TARGET-1_AACGTGCTGACGATGCGGGC        0
```

Result summary.

64.7% (93,795,979 out of 145,032,428) of total read pairs have valid cell and feature barcodes. Majority of fragments in this library have correct structure.

```
2021-02-15 01:51:59,262 - fba.__main__ - INFO - fba version: 0.0.7
2021-02-15 01:51:59,262 - fba.__main__ - INFO - Initiating logging ...
2021-02-15 01:51:59,262 - fba.__main__ - INFO - Python version: 3.7
2021-02-15 01:51:59,262 - fba.__main__ - INFO - Using extract subcommand ...
2021-02-15 01:51:59,276 - fba.levenshtein - INFO - Number of reference cell barcodes: 11,
↪791
2021-02-15 01:51:59,276 - fba.levenshtein - INFO - Number of reference feature barcodes:␣
↪2
2021-02-15 01:51:59,276 - fba.levenshtein - INFO - Read 1 coordinates to search: [0, 16)
2021-02-15 01:51:59,276 - fba.levenshtein - INFO - Read 2 coordinates to search: [31, 51)
2021-02-15 01:51:59,276 - fba.levenshtein - INFO - Cell barcode maximum number of␣
↪mismatches: 2
2021-02-15 01:51:59,276 - fba.levenshtein - INFO - Feature barcode maximum number of␣
↪mismatches: 2
2021-02-15 01:51:59,276 - fba.levenshtein - INFO - Read 1 maximum number of N allowed: 3
2021-02-15 01:51:59,276 - fba.levenshtein - INFO - Read 2 maximum number of N allowed: 3
2021-02-15 01:52:02,510 - fba.levenshtein - INFO - Matching ...
2021-02-15 02:20:39,807 - fba.levenshtein - INFO - Read pairs processed: 10,000,000
2021-02-15 02:49:04,142 - fba.levenshtein - INFO - Read pairs processed: 20,000,000
2021-02-15 03:17:27,422 - fba.levenshtein - INFO - Read pairs processed: 30,000,000
2021-02-15 03:45:54,615 - fba.levenshtein - INFO - Read pairs processed: 40,000,000
2021-02-15 04:14:23,049 - fba.levenshtein - INFO - Read pairs processed: 50,000,000
2021-02-15 04:42:49,377 - fba.levenshtein - INFO - Read pairs processed: 60,000,000
2021-02-15 05:11:15,736 - fba.levenshtein - INFO - Read pairs processed: 70,000,000
2021-02-15 05:39:43,011 - fba.levenshtein - INFO - Read pairs processed: 80,000,000
2021-02-15 06:08:09,940 - fba.levenshtein - INFO - Read pairs processed: 90,000,000
2021-02-15 06:36:39,658 - fba.levenshtein - INFO - Read pairs processed: 100,000,000
2021-02-15 07:05:08,115 - fba.levenshtein - INFO - Read pairs processed: 110,000,000
2021-02-15 07:33:32,101 - fba.levenshtein - INFO - Read pairs processed: 120,000,000
2021-02-15 08:02:01,233 - fba.levenshtein - INFO - Read pairs processed: 130,000,000
2021-02-15 08:30:29,660 - fba.levenshtein - INFO - Read pairs processed: 140,000,000
```

(continues on next page)

(continued from previous page)

```
2021-02-15 08:44:47,038 - fba.levenshtein - INFO - Number of read pairs processed: 145,
↪032,428
2021-02-15 08:44:47,038 - fba.levenshtein - INFO - Number of read pairs w/ valid␣
↪barcodes: 93,795,979
2021-02-15 08:44:47,153 - fba.__main__ - INFO - Done.
```

Matrix generation

Only fragments with correctly matched cell and feature barcodes are included, while fragments with UMI lengths less than the specified value are discarded. UMI removal is performed using UMI-tools (Smith, T., et al. 2017. Genome Res. 27, 491–499.), with the starting position on read 1 set by `-us` (default `16`) and the length set by `-ul` (default `12`). The UMI deduplication method can be set using `-ud` (default `directional`), and the UMI deduplication mismatch threshold can be specified using `-um` (default `1`).

The generated feature count matrix can be easily imported into well-established single cell analysis packages: Seurat and Scanpy.

```
$ fba count \
    -i feature_barcoding_output.tsv.gz \
    -o matrix_featurecount.csv.gz \
    -us 16 \
    -ul 12 \
    -um 1 \
    -ud directional
```

Result summary.

7.6% (7,145,799 out of 93,795,979) of read pairs with valid cell and feature barcodes are unique fragments. 4.9% (7,143,943 out of 145,032,428) of total sequenced read pairs contribute to the final matrix.

```
2020-10-20 04:47:32,738 - fba.__main__ - INFO - fba version: 0.0.7
2020-10-20 04:47:32,738 - fba.__main__ - INFO - Initiating logging ...
2020-10-20 04:47:32,738 - fba.__main__ - INFO - Python version: 3.7
2020-10-20 04:47:32,738 - fba.__main__ - INFO - Using count subcommand ...
2020-10-20 04:47:32,738 - fba.count - INFO - UMI-tools version: 1.0.1
2020-10-20 04:47:32,795 - fba.count - INFO - UMI starting position on read 1: 16
2020-10-20 04:47:32,795 - fba.count - INFO - UMI length: 12
2020-10-20 04:47:32,795 - fba.count - INFO - UMI-tools deduplication threshold: 1
2020-10-20 04:47:32,795 - fba.count - INFO - UMI-tools deduplication method: directional
2020-10-20 04:47:32,795 - fba.count - INFO - Header line: read1_seq cell_barcode cb_num_
↪mismatches read2_seq feature_barcode fb_num_mismatches
2020-10-20 04:51:50,886 - fba.count - INFO - Number of lines processed: 93,795,979
2020-10-20 04:51:50,893 - fba.count - INFO - Number of cell barcodes detected: 11,758
2020-10-20 04:51:50,894 - fba.count - INFO - Number of features detected: 2
2020-10-20 05:00:42,298 - fba.count - INFO - Total UMIs after deduplication: 7,145,799
2020-10-20 05:00:42,320 - fba.count - INFO - Median number of UMIs per cell: 477.0
2020-10-20 05:00:42,434 - fba.__main__ - INFO - Done.
```

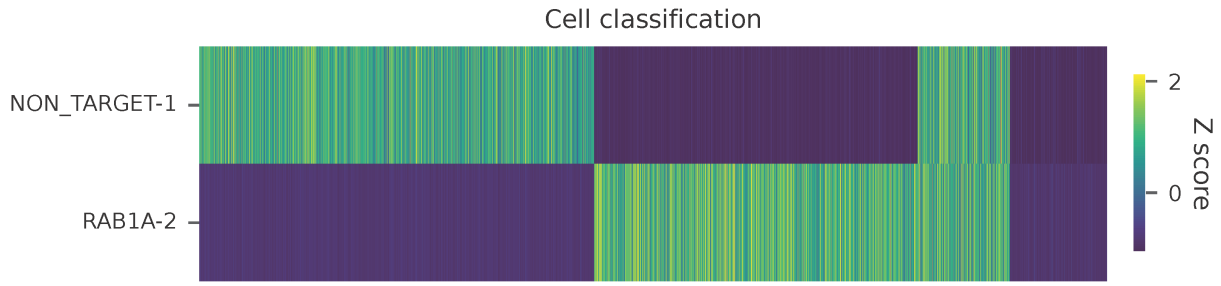
Demultiplexing

Negative binomial distribution

Cells are demultiplexed based on the feature count matrix using demultiplexing method 1 (set by `-dm`), which is implemented based on the method described by Stoeckius, M., et al. (2018) with some modifications. The output directory for demultiplexing is set by `--output_directory` (default `demultiplexed`). A cell identity matrix is generated, where 0 indicates negative and 1 indicates positive. To adjust the quantile threshold for demultiplexing, use `-q` (default `0.9999`). To generate visualization plots, set `-v`.

```
$ fba demultiplex \
    -i matrix_featurecount.csv.gz \
    --output_directory demultiplexed \
    -dm 1 \
    -q 0.75 \
    -v
```

Heatmap of the relative abundance of features (sgRNAs) across all cells. Each column represents a single cell.



t-SNE embedding of cells based on the abundance of features (sgRNAs, no transcriptome information used). Colors indicate the sgRNA status for each cell, as called by FBA.

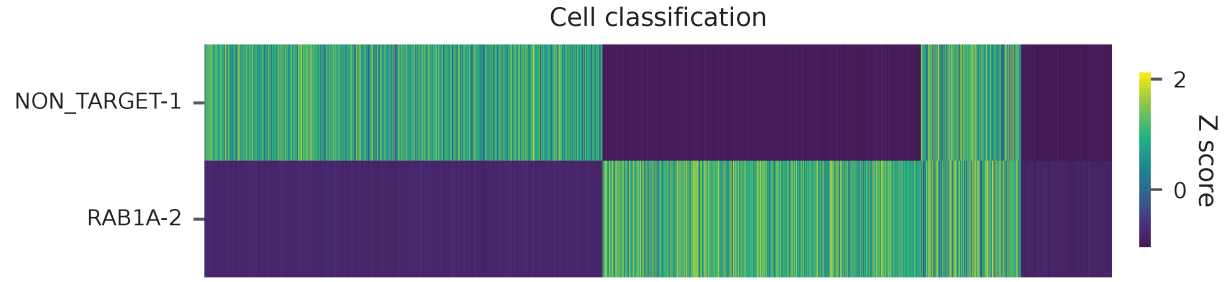
Gaussian mixture model

The implementation of demultiplexing method 2 (set by `-dm`) is inspired by the method described on the 10x Genomics' website. To set the probability threshold for demultiplexing, use `-p` (default `0.9`).

```
$ fba demultiplex \
    -i matrix_featurecount.csv.gz \
    -dm 2 \
    -v
```

```
2021-10-04 14:14:15,659 - fba.__main__ - INFO - fba version: 0.0.x
2021-10-04 14:14:15,659 - fba.__main__ - INFO - Initiating logging ...
2021-10-04 14:14:15,659 - fba.__main__ - INFO - Python version: 3.8
2021-10-04 14:14:15,659 - fba.__main__ - INFO - Using demultiplex subcommand ...
2021-10-04 14:14:36,166 - fba.__main__ - INFO - Skipping arguments: "-q/--quantile", "-
↪cm/--clustering_method"
2021-10-04 14:14:36,166 - fba.demultiplex - INFO - Output directory: demultiplexed
2021-10-04 14:14:36,166 - fba.demultiplex - INFO - Demultiplexing method: 2
2021-10-04 14:14:36,166 - fba.demultiplex - INFO - UMI normalization method: clr
2021-10-04 14:14:36,167 - fba.demultiplex - INFO - Visualization: On
2021-10-04 14:14:36,167 - fba.demultiplex - INFO - Visualization method: tsne
2021-10-04 14:14:36,167 - fba.demultiplex - INFO - Loading feature count matrix: matrix_
↪featurecount.csv.gz ...
2021-10-04 14:14:37,875 - fba.demultiplex - INFO - Number of cells: 11,758
2021-10-04 14:14:37,875 - fba.demultiplex - INFO - Number of positive cells for a␣
↪feature to be included: 200
2021-10-04 14:14:37,920 - fba.demultiplex - INFO - Number of features: 2 / 2 (after␣
↪filtering / original in the matrix)
2021-10-04 14:14:37,920 - fba.demultiplex - INFO - Features: NON_TARGET-1 RAB1A-2
2021-10-04 14:14:37,920 - fba.demultiplex - INFO - Total UMIs: 7,145,799 / 7,145,799
2021-10-04 14:14:37,942 - fba.demultiplex - INFO - Median number of UMIs per cell: 477.0␣
↪/ 477.0
2021-10-04 14:14:37,942 - fba.demultiplex - INFO - Demultiplexing ...
2021-10-04 14:14:38,418 - fba.demultiplex - INFO - Generating heatmap ...
2021-10-04 14:14:42,078 - fba.demultiplex - INFO - Embedding ...
2021-10-04 14:15:24,288 - fba.__main__ - INFO - Done.
```

Heatmap of the relative abundance of features (sgRNAs) across all cells. Each column represents a single cell.



t-SNE embedding of cells based on the abundance of features (sgRNAs, no transcriptome information used). Colors indicate the sgRNA status for each cell, as called by FBA.

UMI distribution and model fitting threshold:

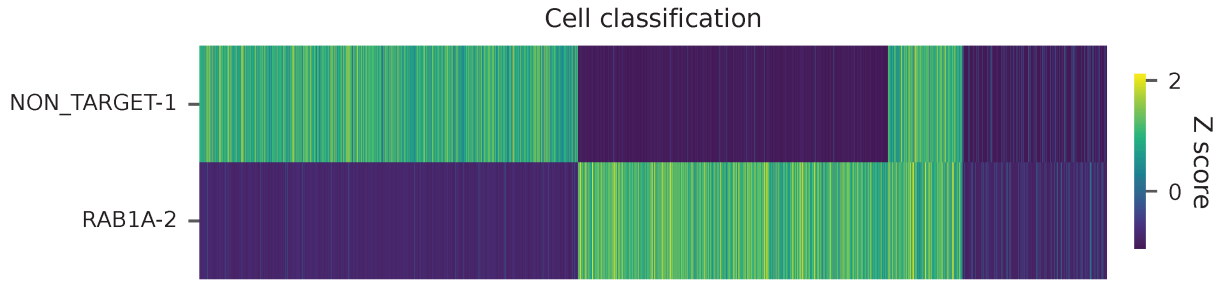
Poisson-Gaussian mixture model

The implementation of demultiplexing method `3` (set by `-dm`) is inspired by Replogle, M., et al. (2021). The probability threshold for demultiplexing can be set using `-p` (default `0.5`).

```
$ fba demultiplex \
    -i matrix_featurecount.csv.gz \
    -dm 3 \
    -v
```

```
2021-12-20 00:13:17,443 - fba.__main__ - INFO - fba version: 0.0.x
2021-12-20 00:13:17,443 - fba.__main__ - INFO - Initiating logging ...
2021-12-20 00:13:17,443 - fba.__main__ - INFO - Python version: 3.9
2021-12-20 00:13:17,443 - fba.__main__ - INFO - Using demultiplex subcommand ...
2021-12-20 00:13:19,774 - fba.__main__ - INFO - Skipping arguments: "-q/--quantile", "-
↪cm/--clustering_method"
2021-12-20 00:13:19,774 - fba.demultiplex - INFO - Output directory: demultiplexed
2021-12-20 00:13:19,774 - fba.demultiplex - INFO - Demultiplexing method: 3
2021-12-20 00:13:19,774 - fba.demultiplex - INFO - UMI normalization method: clr
2021-12-20 00:13:19,774 - fba.demultiplex - INFO - Visualization: On
2021-12-20 00:13:19,774 - fba.demultiplex - INFO - Visualization method: tsne
2021-12-20 00:13:19,774 - fba.demultiplex - INFO - Loading feature count matrix: matrix_
↪featurecount.csv.gz ...
2021-12-20 00:13:20,479 - fba.demultiplex - INFO - Number of cells: 11,758
2021-12-20 00:13:20,479 - fba.demultiplex - INFO - Number of positive cells for a␣
↪feature to be included: 200
2021-12-20 00:13:20,497 - fba.demultiplex - INFO - Number of features: 2 / 2 (after␣
↪filtering / original in the matrix)
2021-12-20 00:13:20,497 - fba.demultiplex - INFO - Features: NON_TARGET-1 RAB1A-2
2021-12-20 00:13:20,497 - fba.demultiplex - INFO - Total UMIs: 7,145,799 / 7,145,799
2021-12-20 00:13:20,506 - fba.demultiplex - INFO - Median number of UMIs per cell: 477.0␣
↪/ 477.0
2021-12-20 00:13:20,506 - fba.demultiplex - INFO - Demultiplexing ...
2021-12-20 00:13:21,930 - fba.demultiplex - INFO - Generating heatmap ...
2021-12-20 00:13:23,070 - fba.demultiplex - INFO - Embedding ...
2021-12-20 00:13:41,271 - fba.__main__ - INFO - Done.
```

Heatmap of the relative abundance of features (sgRNAs) across all cells. Each column represents a single cell.



t-SNE embedding of cells based on the abundance of features (sgRNAs, no transcriptome information used). Colors indicate the sgRNA status for each cell, as called by FBA.

UMI distribution and model fitting threshold:

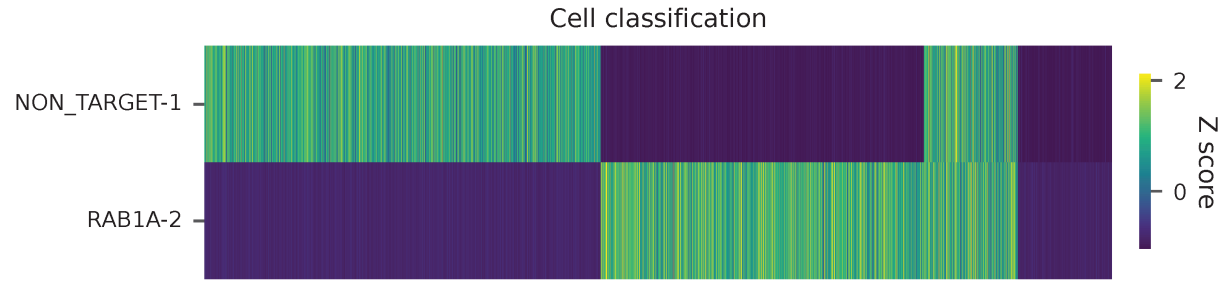
Kernel density estimation

CRISPR perturbations are demultiplexed based on the abundance of features using demultiplexing method 4, which is implemented with modifications to the method described in McGinnis, C., et al. (2019).

```
$ fba demultiplex \
    -i matrix_featurecount.csv.gz \
    -dm 4 \
    -v
```

```
2021-12-26 18:11:16,685 - fba.__main__ - INFO - fba version: 0.0.x
2021-12-26 18:11:16,685 - fba.__main__ - INFO - Initiating logging ...
2021-12-26 18:11:16,686 - fba.__main__ - INFO - Python version: 3.9
2021-12-26 18:11:16,686 - fba.__main__ - INFO - Using demultiplex subcommand ...
2021-12-26 18:11:19,633 - fba.__main__ - INFO - Skipping arguments: "-q/--quantile", "-
↪cm/--clustering_method", "-p/--prob"
2021-12-26 18:11:19,633 - fba.demultiplex - INFO - Output directory: demultiplexed
2021-12-26 18:11:19,633 - fba.demultiplex - INFO - Demultiplexing method: 4
2021-12-26 18:11:19,633 - fba.demultiplex - INFO - UMI normalization method: clr
2021-12-26 18:11:19,633 - fba.demultiplex - INFO - Visualization: On
2021-12-26 18:11:19,633 - fba.demultiplex - INFO - Visualization method: tsne
2021-12-26 18:11:19,633 - fba.demultiplex - INFO - Loading feature count matrix: matrix_
↪featurecount.csv.gz ...
2021-12-26 18:11:19,745 - fba.demultiplex - INFO - Number of cells: 11,758
2021-12-26 18:11:19,745 - fba.demultiplex - INFO - Number of positive cells for a␣
↪feature to be included: 200
2021-12-26 18:11:19,762 - fba.demultiplex - INFO - Number of features: 2 / 2 (after␣
↪filtering / original in the matrix)
2021-12-26 18:11:19,762 - fba.demultiplex - INFO - Features: NON_TARGET-1 RAB1A-2
2021-12-26 18:11:19,762 - fba.demultiplex - INFO - Total UMIs: 7,145,799 / 7,145,799
2021-12-26 18:11:19,771 - fba.demultiplex - INFO - Median number of UMIs per cell: 477.0␣
↪/ 477.0
2021-12-26 18:11:19,771 - fba.demultiplex - INFO - Demultiplexing ...
2021-12-26 18:11:22,049 - fba.demultiplex - INFO - Quantile cutoff: 18
2021-12-26 18:11:23,703 - fba.demultiplex - INFO - Generating heatmap ...
2021-12-26 18:11:24,911 - fba.demultiplex - INFO - Embedding ...
2021-12-26 18:11:44,219 - fba.__main__ - INFO - Done.
```

Heatmap of the relative abundance of features (sgRNAs) across all cells. Each column represents a single cell.



t-SNE embedding of cells based on the abundance of features (sgRNAs, no transcriptome information used). Colors indicate the sgRNA status for each cell, as called by FBA.

UMI distribution and model fitting threshold:

Knee point

Method 1

Cells are demultiplexed based on the abundance of features, specifically sgRNAs. Demultiplexing method `5-2019` is our previous implementation, which aims to identify perturbations in the cells by detecting an inflection point on the feature UMI saturation curve (Xie, S., et al. 2019).

```
$ fba demultiplex \
    -i matrix_featurecount.csv.gz \
    -dm 5-2019 \
    -v
```

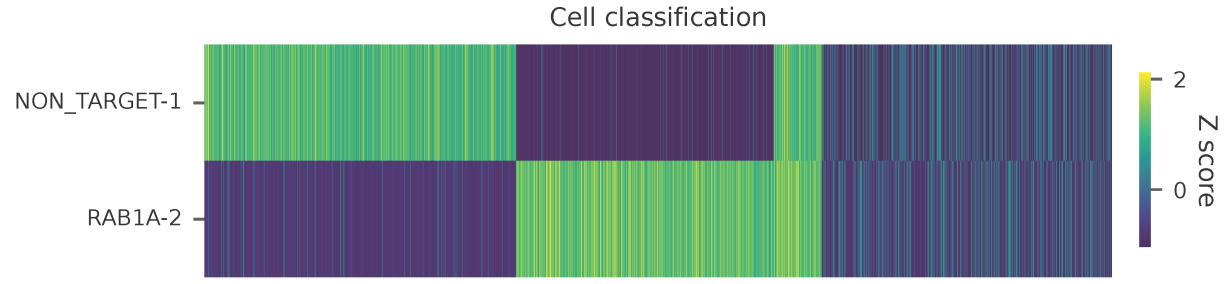
```
2022-01-02 13:57:51,792 - fba.__main__ - INFO - fba version: 0.0.x
2022-01-02 13:57:51,792 - fba.__main__ - INFO - Initiating logging ...
2022-01-02 13:57:51,792 - fba.__main__ - INFO - Python version: 3.9
2022-01-02 13:57:51,792 - fba.__main__ - INFO - Using demultiplex subcommand ...
2022-01-02 13:57:54,328 - fba.__main__ - INFO - Skipping arguments: "-q/--quantile", "-
↪cm/--clustering_method", "-p/--prob"
2022-01-02 13:57:54,329 - fba.demultiplex - INFO - Output directory: demultiplexed
2022-01-02 13:57:54,329 - fba.demultiplex - INFO - Demultiplexing method: 5-2019
2022-01-02 13:57:54,329 - fba.demultiplex - INFO - UMI normalization method: clr
2022-01-02 13:57:54,329 - fba.demultiplex - INFO - Visualization: On
2022-01-02 13:57:54,329 - fba.demultiplex - INFO - Visualization method: tsne
2022-01-02 13:57:54,329 - fba.demultiplex - INFO - Loading feature count matrix: raw/m2_
↪2020-10-20/matrix_featurecount.csv.gz ...
2022-01-02 13:57:54,444 - fba.demultiplex - INFO - Number of cells: 11,758
2022-01-02 13:57:54,444 - fba.demultiplex - INFO - Number of positive cells for a␣
↪feature to be included: 200
```

(continues on next page)

(continued from previous page)

```
2022-01-02 13:57:54,464 - fba.demultiplex - INFO - Number of features: 2 / 2 (after␣
↪filtering / original in the matrix)
2022-01-02 13:57:54,464 - fba.demultiplex - INFO - Features: NON_TARGET-1 RAB1A-2
2022-01-02 13:57:54,464 - fba.demultiplex - INFO - Total UMIs: 7,145,799 / 7,145,799
2022-01-02 13:57:54,474 - fba.demultiplex - INFO - Median number of UMIs per cell: 477.0␣
↪/ 477.0
2022-01-02 13:57:54,474 - fba.demultiplex - INFO - Demultiplexing ...
2022-01-02 13:57:55,509 - fba.demultiplex - INFO - Generating heatmap ...
2022-01-02 13:57:56,717 - fba.demultiplex - INFO - Embedding ...
2022-01-02 13:58:12,014 - fba.__main__ - INFO - Done.
```

Heatmap of the relative abundance of features (sgRNAs) across all cells. Each column represents a single cell.



t-SNE embedding of cells based on the abundance of features (sgRNAs, no transcriptome information used). Colors indicate the sgRNA status for each cell, as called by FBA.

UMI distribution and model fitting threshold:

Method 2

Cells are demultiplexed based on the abundance of features, specifically sgRNAs. Demultiplexing method 5 is implemented to use the local maxima on the difference curve to detemine the knee point on the UMI saturation curve.

```
$ fba demultiplex \
    -i matrix_featurecount.csv.gz \
    -dm 5 \
    -v
```

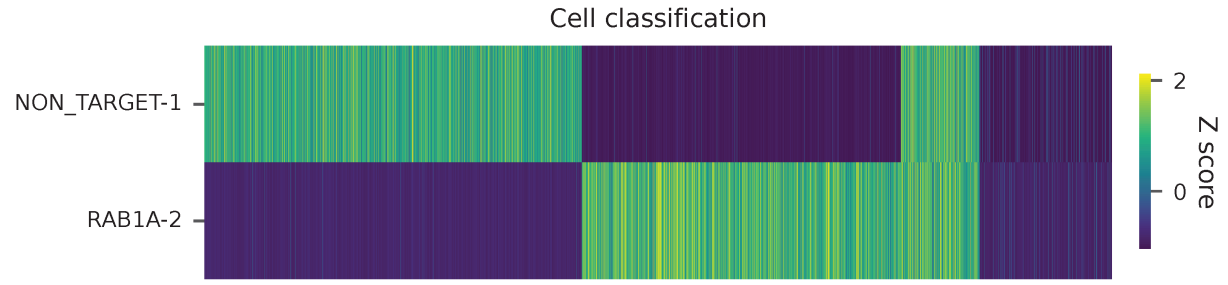
```
2021-12-29 22:55:34,303 - fba.__main__ - INFO - fba version: 0.0.x
2021-12-29 22:55:34,303 - fba.__main__ - INFO - Initiating logging ...
2021-12-29 22:55:34,303 - fba.__main__ - INFO - Python version: 3.9
2021-12-29 22:55:34,303 - fba.__main__ - INFO - Using demultiplex subcommand ...
2021-12-29 22:55:36,774 - fba.__main__ - INFO - Skipping arguments: "-q/--quantile", "-
```

(continues on next page)

(continued from previous page)

```
↪cm/--clustering_method", "-p/--prob"
2021-12-29 22:55:36,774 - fba.demultiplex - INFO - Output directory: demultiplexed
2021-12-29 22:55:36,774 - fba.demultiplex - INFO - Demultiplexing method: 5
2021-12-29 22:55:36,774 - fba.demultiplex - INFO - UMI normalization method: clr
2021-12-29 22:55:36,774 - fba.demultiplex - INFO - Visualization: On
2021-12-29 22:55:36,774 - fba.demultiplex - INFO - Visualization method: tsne
2021-12-29 22:55:36,774 - fba.demultiplex - INFO - Loading feature count matrix: matrix_
↪featurecount.csv.gz ...
2021-12-29 22:55:36,886 - fba.demultiplex - INFO - Number of cells: 11,758
2021-12-29 22:55:36,886 - fba.demultiplex - INFO - Number of positive cells for a␣
↪feature to be included: 200
2021-12-29 22:55:36,904 - fba.demultiplex - INFO - Number of features: 2 / 2 (after␣
↪filtering / original in the matrix)
2021-12-29 22:55:36,904 - fba.demultiplex - INFO - Features: NON_TARGET-1 RAB1A-2
2021-12-29 22:55:36,904 - fba.demultiplex - INFO - Total UMIs: 7,145,799 / 7,145,799
2021-12-29 22:55:36,913 - fba.demultiplex - INFO - Median number of UMIs per cell: 477.0␣
↪/ 477.0
2021-12-29 22:55:36,913 - fba.demultiplex - INFO - Demultiplexing ...
2021-12-29 22:55:37,415 - fba.demultiplex - INFO - Generating heatmap ...
2021-12-29 22:55:38,576 - fba.demultiplex - INFO - Embedding ...
2021-12-29 22:55:57,485 - fba.__main__ - INFO - Done.
```

Heatmap of the relative abundance of features (sgRNAs) across all cells. Each column represents a single cell.



t-SNE embedding of cells based on the abundance of features (sgRNAs, no transcriptome information used). Colors indicate the sgRNA status for each cell, as called by FBA.

UMI distribution and model fitting threshold:

3.1.2 CROP-seq; 1:1:1 Mixture of DNMT3B, MBD1, and TET2 Knockout Cell Lines (HEK293T)

Dataset: CROP-seq; 1:1:1 Mixture of DNMT3B, MBD1, and TET2 Knockout Cell Lines (HEK293T)

Datlinger, P., Rendeiro, A.F., Schmidl, C., Krausgruber, T., Traxler, P., Klughammer, J., Schuster, L.C., Kuchler, A., Alpar, D., and Bock, C. (2017). Pooled CRISPR screening with single-cell transcriptome readout. *Nat. Methods* **14**, 297–301.

Preparation

Download fastq files from European Nucleotide Archive.

```
$ curl -O ftp.sra.ebi.ac.uk/vol1/fastq/SRR516/009/SRR5163029/SRR5163029_1.fastq.gz

$ curl -O ftp.sra.ebi.ac.uk/vol1/fastq/SRR516/009/SRR5163029/SRR5163029_2.fastq.gz
```

Mapping

This dataset comprises single cell RNA-seq data obtained from HEK293T cell lines that had been knocked out for DNMT3B, MBD1, and TET2, separately and were mixed in equal proportions (Fig. 1h). The platform used for this dataset is Drop-seq, and more details about the original data processing can be found here. In brief, the raw data was initially processed with Drop-seq Tools v1.12 pipeline. The first 12 bases of read 1 are cell barcodes, followed by 8 bases of UMIs, while captured single cell transcripts are in read 2.

To perform the downstream analysis, I reprocessed the raw reads to obtain cell-associated barcodes. The raw reads were trimmed using Trim Galore! and then mapped to the human reference genome refdata-gex-GRCh38-2020-A using STAR (version 2.7.8a). The plasmid CROPseq-Guide-Puro sequence was not included in the reference.

Table 1: Mapping statistics

Number of Read Pairs	227,621,653
Number of Read Pairs, After Trimming	201,527,916
Reads Mapped to Genome: Unique+Multiple	0.7654
Reads Mapped to Genome: Unique	0.696169
Estimated Number of Cells	1,199
Fraction of Reads in Cells	0.479281
Mean Reads per Cell	45,290
Median Reads per Cell	16,329
Mean UMI per Cell	2,391
Median UMI per Cell	1,118
Mean Genes per Cell	1,042
Median Genes per Cell	716
Total Genes Detected	16,068

Inspect cell barcodes.

```
$ cat cell_barcodes.txt

AACGGGCATGGG
AACTGGGCATGG
AAGACAGCGTGT
AAGGGCGTACTC
AATAAATACAAA
AATAAATACAAC
AATAAATACAAG
AATAAATACAAT
AATCAATCGCAA
AATCAATCGCAC
```

Prepare feature barcodes. sgRNA sequences can be found in Supplementary Table 1.

```
$ cat feature_barcodes.tsv

DNMT3B  CAGGATTGGGGGCGAGTCGG
MBD1    ATAGGTGTCTGAGCGTCCAC
TET2    CAGGACTCACACGACTATTC
```

Approach A

As sgRNAs are captured together with other transcripts through polyA tails, their locations on read 2 may vary (it should be noted that this is not an sgRNA enrichment library). To expedite processing, we first screened reads that contained the constant sequence (`GACGAAACACCG`) upstream of sgRNAs using cutadapt (version 3.7).

```
$ cutadapt \
    --cores 0 \
    --front GACGAAACACCG \
    --length 25 \
    --minimum-length 25:25 \
    --trimmed-only \
    --output read_2_trimmed.fq.gz --paired-output read_1_trimmed.fq.gz \
    ../SRR5163029_2.fastq.gz ../SRR5163029_1.fastq.gz
```

Preview the filtering result: 1,429,437 out of 227,621,653 (0.6%) read pairs are kept for sgRNA identification.

```
== Read fate breakdown ==
Pairs that were too short:                25,972 (0.0%)
Pairs discarded as untrimmed:        226,166,244 (99.4%)
Pairs written (passing filters):       1,429,437 (0.6%)
```

QC

The first 100,000 read pairs are sampled (default, set by `-n`) for quality control. By default, diagnostic results and plots are generated in the `qc` directory (set by `--output_directory`), and the full length of read 1 and read 2 are searched against reference cell and feature barcodes, respectively. The per base content of both read pairs and the distribution of matched barcode positions are summarized. Use `-r1_c` and/or `-r2_c` to limit the search range, and `-cb_n` and/or `-fb_n` to set the mismatch tolerance for cell and/or feature barcode matching (default 3).

```
$ fba qc \
    -1 read_1_trimmed.fq.gz \
    -2 read_2_trimmed.fq.gz \
    -w cell_barcodes.txt \
    -f feature_barcodes.tsv \
    -r1_c 0,12
```

This library was constructed using the Drop-seq platform, where the first 12 bases correspond to cell barcodes, and the following 8 bases represent UMIs. Based on the base content plot, the GC content of the cell barcodes is evenly distributed. However, the UMIs show a slight T-enrichment.

Regarding read 2, the GC content of sgRNAs is uniformly distributed. It should be noted that the first 20 bases correspond to the sgRNA sequences.

The detailed `qc` results are stored in the `feature_barcoding_output.tsv.gz` file. The `matching_pos` columns indicate the matched positions on reads, while the `matching_description` columns indicate mismatches in the format of substitutions:insertions:deletions.

```
$ zcat feature_barcoding_output.tsv.gz | grep -v no_match | head

read1_seq       cell_barcode    cb_matching_pos cb_matching_description read2_seq      ␣
↪feature_barcode fb_matching_pos fb_matching_description
GCTGCATAGTCGgggggggatttttt      TTCATAGCTCCG    2:12    1:0:2   ␣
↪CAGGACTCACACGACTATTCGTTTT      TET2_CAGGACTCACACGACTATTC        0:20    0:0:0
GTTGCTCCTCACggtgatttttttt       GTTCCCTCCCAC    0:12    1:1:1   ␣
↪CAGGACTCACACGACTATTCGTTTT      TET2_CAGGACTCACACGACTATTC        0:20    0:0:0
TAATGTTTAGGGagggcgctttttt       TAATGTTTAGGG    0:12    0:0:0   ␣
↪ATAGGTGTCTGAGCGTCCACGTTTT      MBD1_ATAGGTGTCTGAGCGTCCAC        0:20    0:0:0
TCTTCCACTACCggtatgacttttt       TCTTCCACTACC    0:12    0:0:0   ␣
↪CAGGATTGGGGGCGAGTCGGGTTTT      DNMT3B_CAGGATTGGGGGCGAGTCGG      0:20    0:0:0
GGAATGCCTTGAgtatacttttttt       GGAATGCCTTGA    0:12    0:0:0   ␣
↪CAGGACTCACACGACTATTCGTTTT      TET2_CAGGACTCACACGACTATTC        0:20    0:0:0
GCGATCACAATGtaatagatttttt       GCGATCACAATG    0:12    0:0:0   ␣
↪CAGGATTGGGGGCGAGTCGGGTTTT      DNMT3B_CAGGATTGGGGGCGAGTCGG      0:20    0:0:0
CGCCGTCGGACAcgaatccttttt        CCGTAGCGGGCA    2:12    1:0:2   ␣
↪ATAGGTGTCTGAGCGTCCACGTTTT      MBD1_ATAGGTGTCTGAGCGTCCAC        0:20    0:0:0
CCGTCCTAGTTGatcccagttttt        CCGTCCTAGTTG    0:12    0:0:0   ␣
↪CAGGACTCACACGACTATTCGTTTT      TET2_CAGGACTCACACGACTATTC        0:20    0:0:0
ATTGTTCCATCTgtcggctttttt        ACTGTTTGATCT    0:12    3:0:0   ␣
↪ATAGGTGTCTGAGCGTCCACGTTTT      MBD1_ATAGGTGTCTGAGCGTCCAC        0:20    0:0:0
```

Barcode extraction

Search ranges are set to `0,12` on read 1 and `0,20` on read 2. One mismatch for cell and feature barcodes (`-cb_m`, `-cf_m`) are allowed.

```
$ fba extract \
    -1 read_1_trimmed.fq.gz \
    -2 read_2_trimmed.fq.gz \
    -w cell_barcodes.txt \
    -f feature_barcodes.tsv \
    -o feature_barcoding_output.tsv.gz \
    -r1_c 0,12 \
    -r2_c 0,20 \
    -cb_m 1 \
    -fb_m 1
```

Preview of result.

```
$ gzip -dc feature_barcoding_output.tsv.gz | head

read1_seq       cell_barcode    cb_num_mismatches       read2_seq       feature_barcode␣
↪fb_num_mismatches
TAATGTTTAGGGagggcgctttttt       TAATGTTTAGGG    0       ATAGGTGTCTGAGCGTCCACgtttt      ␣
↪MBD1_ATAGGTGTCTGAGCGTCCAC      0
TCTTCCACTACCggtatgacttttt       TCTTCCACTACC    0       CAGGATTGGGGGCGAGTCGGgtttt      ␣
↪DNMT3B_CAGGATTGGGGGCGAGTCGG    0
GGAATGCCTTGAgtatacttttttt       GGAATGCCTTGA    0       CAGGACTCACACGACTATTCgtttt       ␣
↪TET2_CAGGACTCACACGACTATTC      0
GCGATCACAATGtaatagatttttt       GCGATCACAATG    0       CAGGATTGGGGGCGAGTCGGgtttt      ␣
↪DNMT3B_CAGGATTGGGGGCGAGTCGG    0
CCGTCCTAGTTGatcccagttttt        CCGTCCTAGTTG    0       CAGGACTCACACGACTATTCgtttt       ␣
↪TET2_CAGGACTCACACGACTATTC      0
ATTATATGTGAGcagacttttttt        ATTATATGTGAG    0       ATAGGTGTCTGAGCGTCCACgtttt       ␣
↪MBD1_ATAGGTGTCTGAGCGTCCAC      0
TTTCAGTATTGGggcgaatttttt        TTTCAGTATTGG    0       ATAGGTGTCTGAGCGTCCACgtttt       ␣
↪MBD1_ATAGGTGTCTGAGCGTCCAC      0
GTTCCCTCCCAAacatgagttttt        GTTCCCTCCCAA    0       CAGGATTGGGGGCGAGTCGGgtttt       ␣
↪DNMT3B_CAGGATTGGGGGCGAGTCGG    0
GCTCCGCTTTTAactcaagttttt        GCTCCGCTTTTA    0       CAGGATTGGGGCCGAGTCGGgactt       ␣
↪DNMT3B_CAGGATTGGGGGCGAGTCGG    1
```

Result summary.

9,213 out of 1,429,437 read pairs have valid cell and feature barcodes.

```
2022-03-07 16:11:53,295 - fba.__main__ - INFO - fba version: 0.0.x
2022-03-07 16:11:53,295 - fba.__main__ - INFO - Initiating logging ...
2022-03-07 16:11:53,295 - fba.__main__ - INFO - Python version: 3.10
2022-03-07 16:11:53,295 - fba.__main__ - INFO - Using extract subcommand ...
2022-03-07 16:11:53,310 - fba.levenshtein - INFO - Number of reference cell barcodes: 1,
↪199
```

(continues on next page)

(continued from previous page)

```
2022-03-07 16:11:53,310 - fba.levenshtein - INFO - Number of reference feature barcodes:␣
↪3
2022-03-07 16:11:53,310 - fba.levenshtein - INFO - Read 1 coordinates to search: [0, 12)
2022-03-07 16:11:53,310 - fba.levenshtein - INFO - Read 2 coordinates to search: [0, 20)
2022-03-07 16:11:53,310 - fba.levenshtein - INFO - Cell barcode maximum number of␣
↪mismatches: 1
2022-03-07 16:11:53,310 - fba.levenshtein - INFO - Feature barcode maximum number of␣
↪mismatches: 1
2022-03-07 16:11:53,312 - fba.levenshtein - INFO - Read 1 maximum number of N allowed: 3
2022-03-07 16:11:53,312 - fba.levenshtein - INFO - Read 2 maximum number of N allowed: 3
2022-03-07 16:11:53,337 - fba.levenshtein - INFO - Matching ...
2022-03-07 16:12:13,951 - fba.levenshtein - INFO - Number of read pairs processed: 1,429,
↪437
2022-03-07 16:12:13,952 - fba.levenshtein - INFO - Number of read pairs w/ valid␣
↪barcodes: 9,213
2022-03-07 16:12:13,954 - fba.__main__ - INFO - Done.
```

Matrix generation

Only fragments with correctly matched cell and feature barcodes are included, while fragments with UMI lengths less than the specified value are discarded. UMI removal is performed using UMI-tools (Smith, T., et al. 2017. Genome Res. 27, 491–499.), with the starting position on read 1 set by `-us` (default `16`) and the length set by `-ul` (default `12`). The UMI deduplication method can be set using `-ud` (default `directional`), and the UMI deduplication mismatch threshold can be specified using `-um` (default `1`).

The generated feature count matrix can be easily imported into well-established single cell analysis packages such as Seurat and Scanpy.

```
$ fba count \
    -i feature_barcoding_output.tsv.gz \
    -o matrix_featurecount.csv.gz \
    -us 12 \
    -ul 8
```

Result summary.

```
2022-03-08 13:43:27,499 - fba.__main__ - INFO - fba version: 0.0.x
2022-03-08 13:43:27,499 - fba.__main__ - INFO - Initiating logging ...
2022-03-08 13:43:27,499 - fba.__main__ - INFO - Python version: 3.9
2022-03-08 13:43:27,499 - fba.__main__ - INFO - Using count subcommand ...
2022-03-08 13:43:28,183 - fba.count - INFO - UMI-tools version: 1.1.1
2022-03-08 13:43:28,184 - fba.count - INFO - UMI starting position on read 1: 12
2022-03-08 13:43:28,184 - fba.count - INFO - UMI length: 8
2022-03-08 13:43:28,184 - fba.count - INFO - UMI-tools deduplication threshold: 1
2022-03-08 13:43:28,184 - fba.count - INFO - UMI-tools deduplication method: directional
2022-03-08 13:43:28,184 - fba.count - INFO - Header line: read1_seq cell_barcode cb_num_
↪mismatches read2_seq feature_barcode fb_num_mismatches
2022-03-08 13:43:28,194 - fba.count - INFO - Number of lines processed: 9,213
```

(continues on next page)

(continued from previous page)

```
2022-03-08 13:43:28,194 - fba.count - INFO - Number of cell barcodes detected: 420
2022-03-08 13:43:28,194 - fba.count - INFO - Number of features detected: 3
2022-03-08 13:43:28,194 - fba.count - INFO - UMI deduplicating ...
2022-03-08 13:43:28,202 - fba.count - INFO - Total UMIs after deduplication: 1,089
2022-03-08 13:43:28,202 - fba.count - INFO - Median number of UMIs per cell: 1.0
2022-03-08 13:43:28,204 - fba.__main__ - INFO - Done.
```

Demultiplexing

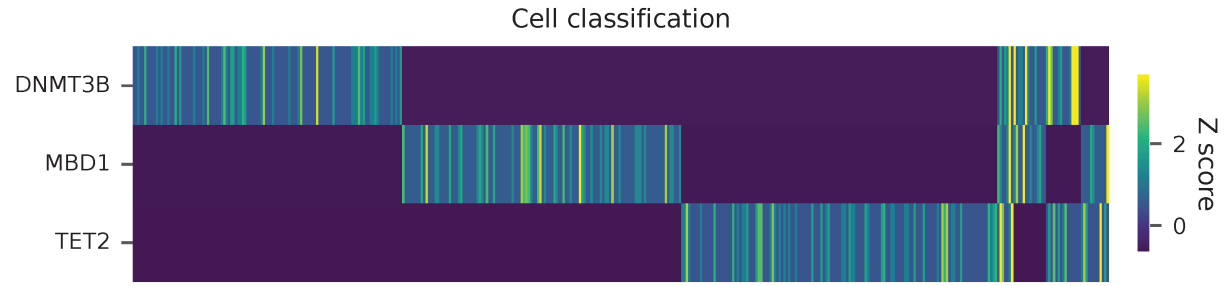
Gaussian mixture model

The implementation of demultiplexing method 2 (set by `-dm`) is inspired by the method described on the 10x Genomics' website. To set the probability threshold for demultiplexing, use `-p` (default `0.9`). To specify the minimum number of positive cells for a given feature to be considered during demultiplexing, use `-nc` (default `200`).

```
$ fba demultiplex \
    -i matrix_featurecount.csv.gz \
    -dm 2 \
    -v \
    -nc 0
```

```
2022-03-07 19:57:14,925 - fba.__main__ - INFO - fba version: 0.0.x
2022-03-07 19:57:14,925 - fba.__main__ - INFO - Initiating logging ...
2022-03-07 19:57:14,925 - fba.__main__ - INFO - Python version: 3.9
2022-03-07 19:57:14,925 - fba.__main__ - INFO - Using demultiplex subcommand ...
2022-03-07 19:57:17,564 - fba.__main__ - INFO - Skipping arguments: "-q/--quantile", "-
↪cm/--clustering_method"
2022-03-07 19:57:17,564 - fba.demultiplex - INFO - Output directory: demultiplexed_gm
2022-03-07 19:57:17,564 - fba.demultiplex - INFO - Demultiplexing method: 2
2022-03-07 19:57:17,564 - fba.demultiplex - INFO - UMI normalization method: clr
2022-03-07 19:57:17,564 - fba.demultiplex - INFO - Visualization: On
2022-03-07 19:57:17,564 - fba.demultiplex - INFO - Visualization method: tsne
2022-03-07 19:57:17,564 - fba.demultiplex - INFO - Loading feature count matrix: matrix_
↪featurecount.csv.gz ...
2022-03-07 19:57:17,571 - fba.demultiplex - INFO - Number of cells: 420
2022-03-07 19:57:17,571 - fba.demultiplex - INFO - Number of positive cells for a␣
↪feature to be included: 0
2022-03-07 19:57:17,572 - fba.demultiplex - INFO - Number of features: 3 / 3 (after␣
↪filtering / original in the matrix)
2022-03-07 19:57:17,572 - fba.demultiplex - INFO - Features: DNMT3B MBD1 TET2
2022-03-07 19:57:17,572 - fba.demultiplex - INFO - Total UMIs: 1,081 / 1,081
2022-03-07 19:57:17,573 - fba.demultiplex - INFO - Median number of UMIs per cell: 1.0 /␣
↪1.0
2022-03-07 19:57:17,573 - fba.demultiplex - INFO - Demultiplexing ...
2022-03-07 19:57:18,277 - fba.demultiplex - INFO - Generating heatmap ...
2022-03-07 19:57:18,423 - fba.demultiplex - INFO - Embedding ...
2022-03-07 19:57:21,922 - fba.__main__ - INFO - Done.
```

Heatmap of the relative abundance of features (sgRNAs) across all cells. Each column represents a single cell.



Preview the demultiplexing result: the numbers of singlets and multiplets.

```
In [1]: import pandas as pd

In [2]: m = pd.read_csv("demultiplexed/matrix_cell_identity.csv.gz", index_col=0)

In [3]: m.loc[:, m.sum(axis=0) == 1].sum(axis=1)
Out[3]:
DNMT3B    141
MBD1      150
TET2      158
dtype: int64

In [4]: sum(m.sum(axis=0) > 1)
Out[4]: 74
```

Knee point

Cells are demultiplexed according to the abundance of features, specifically sgRNAs. Demultiplexing method 5 is implemented to use the local maxima on the difference curve to detemine the knee point on the UMI saturation curve.

```
$ fba demultiplex \
    -i matrix_featurecount.csv.gz \
    -dm 5 \
    -v \
    -nc 0
```

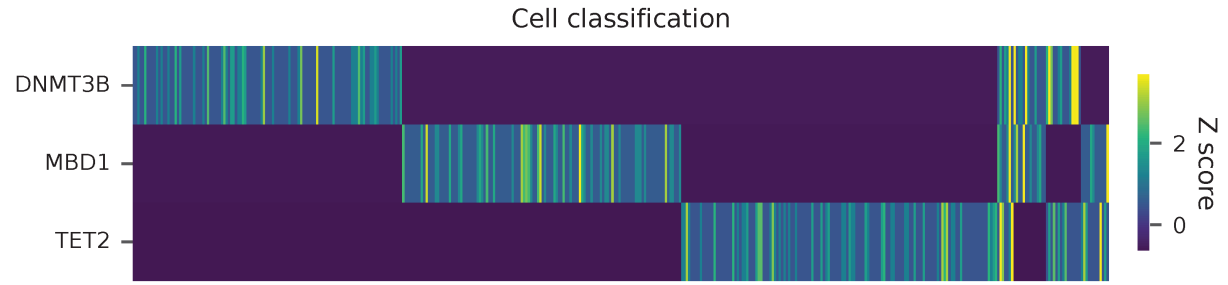
```
2022-03-05 01:52:38,900 - fba.__main__ - INFO - fba version: 0.0.x
2022-03-05 01:52:38,900 - fba.__main__ - INFO - Initiating logging ...
2022-03-05 01:52:38,900 - fba.__main__ - INFO - Python version: 3.9
2022-03-05 01:52:38,900 - fba.__main__ - INFO - Using demultiplex subcommand ...
2022-03-05 01:52:41,396 - fba.__main__ - INFO - Skipping arguments: "-q/--quantile", "-
↪cm/--clustering_method", "-p/--prob"
2022-03-05 01:52:41,396 - fba.demultiplex - INFO - Output directory: demultiplexed
2022-03-05 01:52:41,396 - fba.demultiplex - INFO - Demultiplexing method: 5
2022-03-05 01:52:41,396 - fba.demultiplex - INFO - UMI normalization method: clr
```

(continues on next page)

(continued from previous page)

```
2022-03-05 01:52:41,396 - fba.demultiplex - INFO - Visualization: On
2022-03-05 01:52:41,396 - fba.demultiplex - INFO - Visualization method: tsne
2022-03-05 01:52:41,396 - fba.demultiplex - INFO - Loading feature count matrix: matrix_
↪featurecount.csv.gz ...
2022-03-05 01:52:41,403 - fba.demultiplex - INFO - Number of cells: 523
2022-03-05 01:52:41,403 - fba.demultiplex - INFO - Number of positive cells for a␣
↪feature to be included: 0
2022-03-05 01:52:41,404 - fba.demultiplex - INFO - Number of features: 3 / 3 (after␣
↪filtering / original in the matrix)
2022-03-05 01:52:41,404 - fba.demultiplex - INFO - Features: DNMT3B MBD1 TET2
2022-03-05 01:52:41,404 - fba.demultiplex - INFO - Total UMIs: 1,364 / 1,364
2022-03-05 01:52:41,405 - fba.demultiplex - INFO - Median number of UMIs per cell: 1.0 /␣
↪1.0
2022-03-05 01:52:41,405 - fba.demultiplex - INFO - Demultiplexing ...
2022-03-05 01:52:41,810 - fba.demultiplex - INFO - Generating heatmap ...
2022-03-05 01:52:41,979 - fba.demultiplex - INFO - Embedding ...
2022-03-05 01:52:44,840 - fba.__main__ - INFO - Done.
```

Heatmap of the relative abundance of features (sgRNAs) across all cells. Each column represents a single cell.



Preview the demultiplexing result: the numbers of singlets and multiplets.

```
In [1]: import pandas as pd

In [2]: m = pd.read_csv("demultiplexed/matrix_cell_identity.csv.gz", index_col=0)

In [3]: m.loc[:, m.sum(axis=0) == 1].sum(axis=1)
Out[3]:
DNMT3B    141
MBD1      150
TET2      158
dtype: int64

In [4]: sum(m.sum(axis=0) > 1)
Out[4]: 74
```

Approach B

Rather than pre-filtering read 2 for the constant upstream region of sgRNA, we conduct a search for sgRNAs throughout the entire read 2. Although this approach may take longer, it provides more comprehensive results. To optimize speed, we suggest splitting fastq files and running them on different nodes simultaneously.

Barcode extraction

The CROPseq-Guide-Puro transcripts captured by Drop-seq beads in single cell RNA-seq library contain sgRNA sequences. As there are no secondary libraries for sgRNA enrichment, we need to extract the sgRNA sequences from read 2. However, the locations of sgRNAs on read 2 vary because the transcripts are captured by polyA tails. Therefore, we use the `qc` mode for sgRNA extraction, which is capable of handling the variable locations of sgRNAs on read 2.

To specify the number of reads to analyze, use `--num_reads`, where `None` means all reads. To set the number of threads, use `-t`. By default, the diagnostic results and plots are generated in the `qc` directory (set by `--output_directory`), and the full length of read 1 and read 2 are searched against reference cell and feature barcodes, respectively. The per base content of both read pairs and the distribution of matched barcode positions are summarized. To limit the search range for read 1 and/or read 2, use `-r1_c` and/or `-r2_c`, respectively. Use `-cb_n` and/or `-fb_n` to set the mismatch tolerance for cell and/or feature barcode matching (default `3`).

```
$ fba qc \
    -1 SRR5163029_1.fastq.gz \
    -2 SRR5163029_2.fastq.gz \
    -w cell_barcodes.txt \
    -f feature_barcodes.tsv \
    -cb_m 1 \
    -fb_m 1 \
    -cb_n 15 \
    -fb_n 15 \
    -r1_c 0,12 \
    -t $SLURM_CPUS_ON_NODE \
    --num_reads None
```

The detailed `qc` results are stored in the `feature_barcoding_output.tsv.gz` file. The `matching_pos` columns indicate the matched positions on reads, while the `matching_description` columns indicate mismatches in the format of substitutions:insertions:deletions.

```
$ gzip -dc qc/feature_barcoding_output.tsv.gz | head

read1_seq       cell_barcode    cb_matching_pos cb_matching_description read2_seq     ␣
↪feature_barcode fb_matching_pos fb_matching_description
TTTAGGATCGTTtgatgtattttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttt
↪no_match        NA      NA     ␣
↪ATTAAAAATATTGTGGCAGGAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
↪   NA      NA      NA
GTCGAAACTCTTaacgggattttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttt
↪no_match        NA      NA     ␣
↪TTATAATGGTTACAAATAAAGCAATAGCATCACAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
↪   NA      NA      NA
GTTTACGTGTTCatgggcgatttttttttttttttttttttttttttttaaaaaagttaaaagggggcccgtgggggacaaatagagggcctagagttcca
↪GTTTACGTGTTC    0:12    0:0:0  ␣
↪GGAGTACGGAGAATTCTATAAGAGCTTGACCAATGACTGGGAAGATCACTTGGCAGTGAAGCATTTTTCAGTTGAAGGACAGTTGGAATTCAGAGCCCTTCT
↪   no_match        NA      NA
```

(continues on next page)

(continued from previous page)

```
CCGTCCTAGTTGgtgtatatttttttgttttttttttttttttcaccgggtcagagctgcccctaagtaccacgtcccgtcccacctttatcggacctcggcc
↪CCGTCCTAGTTG    0:12    0:0:0  ␣
↪AATTAAGTCTCGTAAAGAACGAGAAGCTGAACTTGGACCTAGGGCAACCGACTTCACCAATGTTTACAGCGAGAATCTTGGTGACGACGTGGATGATGAGC
↪   no_match        NA      NA
TTTCAGTATTGGggcgaatttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttggctagtttttttgtggttttttgcttttggttctctcgt
↪TTTCAGTATTGG    0:12    0:0:0  ␣
↪GGACGAAACACCGATAGGTGTCTGAGCGTCCACGTTTTAGAGCTAGAAATAGCAAGTTAAAATAAGGCTAGTCCGTTATCAACTTGAAAAAGTGGCACCGAG
↪MBD1_ATAGGTGTCTGAGCGTCCAC       13:33 0:0:0
CTAGGTACCACTagacagtttttttttttttttttttttttttttttttttttttttttttttttttttttctctatgtgtgctttttttggctttagtctgtgggtccctag
↪CTAGGTACCACT    0:12    0:0:0  ␣
↪TTCTTGGGTAGTTTGCAGTTTTTAAAATTATGTTTTAAAATGGACTATCATATGCTTACCGTAACTTGAAAGTATTTCGATTTCTTGGCTTTATATATCTTG
↪MBD1_ATAGGTGTCTGAGCGTCCAC       121:1410:0:0
TCTTCCACTACCgtcccgtctttttttttttttttttttttttttttttttttttttttttttttttctttatgtcagtttttttgtgctttagtattgggttcccttgttt
↪TCTTCCACTACC    0:12    0:0:0  ␣
↪ACCGATAGGTGTCTGAGCGTCCACGTTTTAGAGCTAGAAATAGCAAGTTAAAATAAGGCTAGTCCGTTATCAACTTGAAAAAGTGGCACCGAGTCGGTGCT
↪MBD1_ATAGGTGTCTGAGCGTCCAC       4:24  0:0:0
CTTAATTTGGTGgggaagatttttttttttttttttttttttttttttttttaagtactttaagtaagctttttttaggctttagccgtgggttcccctgttagcccg
↪CTTAATTTGGTG    0:12    0:0:0  ␣
↪CCGTAACTTGAAAGTATTTCGATTTCTTGGCTTTATATATCTTGTGGAAAGGACGAAACACCGCAGGACTCACACGACTCTTCGTTTTAGAGCTAGCAATAG
↪TET2_CAGGACTCACACGACTATTC       63:83 1:0:0
TCGTACATACGGtggtttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttgttttttttttttttttgttttttttttgtgtccttt
↪TCGTACATACGG    0:12    0:0:0  ␣
↪ATCATATGCTTACCGTAACTTGAAAGTATTTCGATTTCTTGGCTTTATATATCTTGTGGAAAGGACGAAACACCGCAGGATTGGGGGCGAGTCGGGTTTTAC
↪DNMT3B_CAGGATTGGGGGCGAGTCGG     75:95 0:0:0
```

Matrix generation

Only fragments with correctly matched cell and feature barcodes are included, while fragments with UMI lengths less than the specified value are discarded. UMI removal is performed using UMI-tools (Smith, T., et al. 2017. Genome Res. 27, 491–499.), with the starting position on read 1 set by `-us` (default `16`) and the length set by `-ul` (default `12`). The UMI deduplication method can be set using `-ud` (default `directional`), and the UMI deduplication mismatch threshold can be specified using `-um` (default `1`).

The generated feature count matrix can be easily imported into well-established single cell analysis packages: Seurat and Scanpy.

```
$ fba count \
    -i feature_barcoding_output.tsv.gz \
    -o matrix_featurecount.csv.gz \
    -us 12 \
    -ul 8
```

Result summary.

11.76 % (1,364 out of 11,597) of read pairs with valid cell and feature barcodes are unique fragments.

```
2022-03-04 23:18:27,501 - fba.__main__ - INFO - fba version: 0.0.x
2022-03-04 23:18:27,501 - fba.__main__ - INFO - Initiating logging ...
```

(continues on next page)

(continued from previous page)

```
2022-03-04 23:18:27,501 - fba.__main__ - INFO - Python version: 3.10
2022-03-04 23:18:27,501 - fba.__main__ - INFO - Using count subcommand ...
2022-03-04 23:18:31,494 - fba.count - INFO - UMI-tools version: 1.1.2
2022-03-04 23:18:31,495 - fba.count - INFO - UMI start position on read 1 auto-detected,␣
↪overriding -us
2022-03-04 23:18:31,495 - fba.count - INFO - UMI length: 8
2022-03-04 23:18:31,496 - fba.count - INFO - UMI-tools deduplication threshold: 1
2022-03-04 23:18:31,496 - fba.count - INFO - UMI-tools deduplication method: directional
2022-03-04 23:18:31,496 - fba.count - INFO - Header line: read1_seq cell_barcode cb_
↪matching_pos cb_matching_description read2_seq feature_barcode fb_matching_pos fb_
↪matching_description
2022-03-04 23:18:31,581 - fba.count - INFO - Number of lines processed: 11,597
2022-03-04 23:18:31,581 - fba.count - INFO - Number of cell barcodes detected: 523
2022-03-04 23:18:31,582 - fba.count - INFO - Number of features detected: 3
2022-03-04 23:18:31,608 - fba.count - INFO - Total UMIs after deduplication: 1,364
2022-03-04 23:18:31,609 - fba.count - INFO - Median number of UMIs per cell: 1.0
2022-03-04 23:18:31,615 - fba.__main__ - INFO - Done.
```

Demultiplexing

Gaussian mixture model

The implementation of demultiplexing method `2` (set by `-dm`) is inspired by the method described on the 10x Genomics' website. To set the probability threshold for demultiplexing, use `-p` (default `0.9`). To specify the minimum number of positive cells for a given feature to be considered during demultiplexing, use `-nc` (default `200`).

```
$ fba demultiplex \
    -i matrix_featurecount.csv.gz \
    -dm 2 \
    -v \
    -nc 0
```

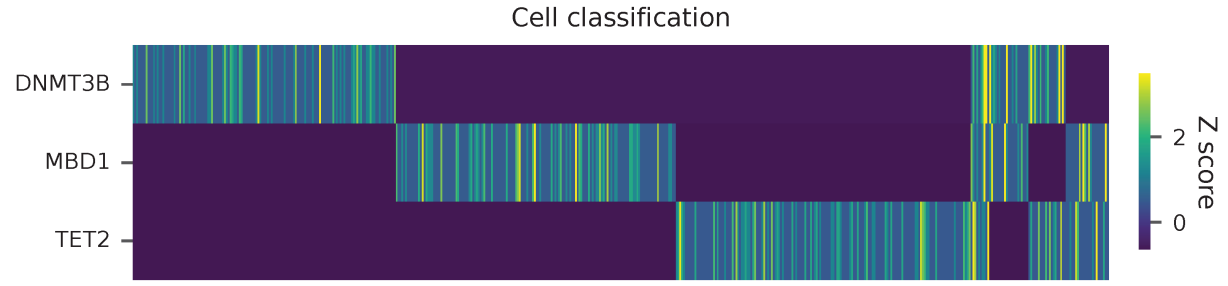
```
2022-03-04 23:19:05,218 - fba.__main__ - INFO - fba version: 0.0.x
2022-03-04 23:19:05,219 - fba.__main__ - INFO - Initiating logging ...
2022-03-04 23:19:05,219 - fba.__main__ - INFO - Python version: 3.10
2022-03-04 23:19:05,219 - fba.__main__ - INFO - Using demultiplex subcommand ...
2022-03-04 23:19:15,199 - fba.__main__ - INFO - Skipping arguments: "-q/--quantile", "-
↪cm/--clustering_method"
2022-03-04 23:19:15,200 - fba.demultiplex - INFO - Output directory: demultiplexed
2022-03-04 23:19:15,201 - fba.demultiplex - INFO - Demultiplexing method: 2
2022-03-04 23:19:15,201 - fba.demultiplex - INFO - UMI normalization method: clr
2022-03-04 23:19:15,201 - fba.demultiplex - INFO - Visualization: On
2022-03-04 23:19:15,201 - fba.demultiplex - INFO - Visualization method: tsne
2022-03-04 23:19:15,201 - fba.demultiplex - INFO - Loading feature count matrix: matrix_
↪featurecount.csv.gz ...
2022-03-04 23:19:15,219 - fba.demultiplex - INFO - Number of cells: 523
```

(continues on next page)

(continued from previous page)

```
2022-03-04 23:19:15,219 - fba.demultiplex - INFO - Number of positive cells for a␣
↪feature to be included: 0
2022-03-04 23:19:15,222 - fba.demultiplex - INFO - Number of features: 3 / 3 (after␣
↪filtering / original in the matrix)
2022-03-04 23:19:15,222 - fba.demultiplex - INFO - Features: DNMT3B MBD1 TET2
2022-03-04 23:19:15,222 - fba.demultiplex - INFO - Total UMIs: 1,364 / 1,364
2022-03-04 23:19:15,223 - fba.demultiplex - INFO - Median number of UMIs per cell: 1.0 /␣
↪1.0
2022-03-04 23:19:15,223 - fba.demultiplex - INFO - Demultiplexing ...
2022-03-04 23:19:17,319 - fba.demultiplex - INFO - Generating heatmap ...
2022-03-04 23:19:17,784 - fba.demultiplex - INFO - Embedding ...
2022-03-04 23:19:32,256 - fba.__main__ - INFO - Done.
```

Heatmap of the relative abundance of features (sgRNAs) across all cells. Each column represents a single cell.



Preview the demultiplexing result: the numbers of singlets and multiplets.

```
In [1]: import pandas as pd

In [2]: m = pd.read_csv("demultiplexed/matrix_cell_identity.csv.gz", index_col=0)

In [3]: m.loc[:, m.sum(axis=0) == 1].sum(axis=1)
Out[3]:
DNMT3B    141
MBD1      150
TET2      158
dtype: int64

In [4]: sum(m.sum(axis=0) > 1)
Out[4]: 74
```

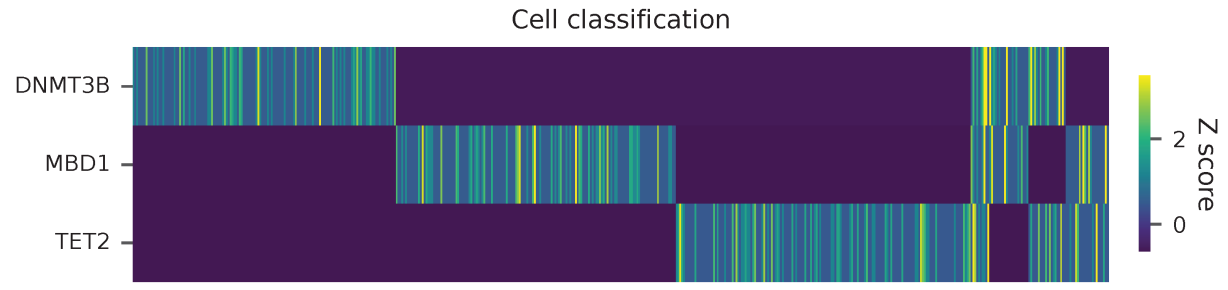
Knee point

Cells are demultiplexed according to the abundance of features, specifically sgRNAs. Demultiplexing method 5 is implemented to use the local maxima on the difference curve to detemine the knee point on the UMI saturation curve.

```
$ fba demultiplex \
    -i matrix_featurecount.csv.gz \
    -dm 5 \
    -v \
    -nc 0
```

```
2022-03-05 01:52:38,900 - fba.__main__ - INFO - fba version: 0.0.x
2022-03-05 01:52:38,900 - fba.__main__ - INFO - Initiating logging ...
2022-03-05 01:52:38,900 - fba.__main__ - INFO - Python version: 3.9
2022-03-05 01:52:38,900 - fba.__main__ - INFO - Using demultiplex subcommand ...
2022-03-05 01:52:41,396 - fba.__main__ - INFO - Skipping arguments: "-q/--quantile", "-
↪cm/--clustering_method", "-p/--prob"
2022-03-05 01:52:41,396 - fba.demultiplex - INFO - Output directory: demultiplexed
2022-03-05 01:52:41,396 - fba.demultiplex - INFO - Demultiplexing method: 5
2022-03-05 01:52:41,396 - fba.demultiplex - INFO - UMI normalization method: clr
2022-03-05 01:52:41,396 - fba.demultiplex - INFO - Visualization: On
2022-03-05 01:52:41,396 - fba.demultiplex - INFO - Visualization method: tsne
2022-03-05 01:52:41,396 - fba.demultiplex - INFO - Loading feature count matrix: matrix_
↪featurecount.csv.gz ...
2022-03-05 01:52:41,403 - fba.demultiplex - INFO - Number of cells: 523
2022-03-05 01:52:41,403 - fba.demultiplex - INFO - Number of positive cells for a␣
↪feature to be included: 0
2022-03-05 01:52:41,404 - fba.demultiplex - INFO - Number of features: 3 / 3 (after␣
↪filtering / original in the matrix)
2022-03-05 01:52:41,404 - fba.demultiplex - INFO - Features: DNMT3B MBD1 TET2
2022-03-05 01:52:41,404 - fba.demultiplex - INFO - Total UMIs: 1,364 / 1,364
2022-03-05 01:52:41,405 - fba.demultiplex - INFO - Median number of UMIs per cell: 1.0 /␣
↪1.0
2022-03-05 01:52:41,405 - fba.demultiplex - INFO - Demultiplexing ...
2022-03-05 01:52:41,810 - fba.demultiplex - INFO - Generating heatmap ...
2022-03-05 01:52:41,979 - fba.demultiplex - INFO - Embedding ...
2022-03-05 01:52:44,840 - fba.__main__ - INFO - Done.
```

Heatmap of the relative abundance of features (sgRNAs) across all cells. Each column represents a single cell.



Preview the demultiplexing result: the numbers of singlets and multiplets.

```
In [1]: import pandas as pd

In [2]: m = pd.read_csv("demultiplexed/matrix_cell_identity.csv.gz", index_col=0)

In [3]: m.loc[:, m.sum(axis=0) == 1].sum(axis=1)
Out[3]:
DNMT3B    141
MBD1      150
TET2      158
dtype: int64

In [4]: sum(m.sum(axis=0) > 1)
Out[4]: 74
```

UMI distribution and knee point detection:

3.1.3 Direct-capture Perturb-seq; CRISPRi-based Screen of Unfolded Protein Response (UPR) Using 3' sgRNA-CR1^{cs1}

Dataset: Direct-capture Perturb-seq (sgRNA-CR1^{cs1})

Replogle, J.M., Norman, T.M., Xu, A., Hussmann, J.A., Chen, J., Cogan, J.Z., Meer, E.J., Terry, J.M., Riordan, D.P., Srinivas, N., et al. (2020). Combinatorial single-cell CRISPR screens by direct guide RNA capture and targeted sequencing. *Nat. Biotechnol.* **38**, 954–961.

Preparation

Download fastq files from Gene Expression Omnibus.

```
$ cat SRR11214033_2.fastq.gz SRR11214034_2.fastq.gz SRR11214035_2.fastq.gz SRR11214036_2.
↪fastq.gz > GSM4367980_1.fq.gz

$ cat SRR11214033_3.fastq.gz SRR11214034_3.fastq.gz SRR11214035_3.fastq.gz SRR11214036_3.
↪fastq.gz > GSM4367980_2.fq.gz
```

Prepare cell barcodes.

```
$ wget https://ftp.ncbi.nlm.nih.gov/geo/samples/GSM4367nnn/GSM4367979/suppl/GSM4367979_
↪exp1-5.barcodes.tsv.gz
```

(continues on next page)

(continued from previous page)

```
$ gzip -dc GSM4367979_exp1-5.barcodes.tsv.gz | grep '\-4' > cell_barcodes.txt
```

Inspect cell barcodes (Fig. 2d, middle column).

```
$ wc -l cell_barcodes.txt

8727

$ head cell_barcodes.txt

AAACCCAAGAGGGCGA-4
AAACCCACACCCTCTA-4
AAACCCACATAGATGA-4
AAACCCACATATCGGT-4
AAACCCATCATGAGTC-4
AAACCCATCCGGTAAT-4
AAACCCATCCTGCCAT-4
AAACCCATCTCACCCA-4
AAACGAAAGCCTGACC-4
AAACGAAAGTGCCGAA-4
```

Prepare feature barcodes. sgRNA sequences can be found in Supplementary Table 2 and are truncated to equal length.

```
$ cat feature_barcodes_UPR_edited.tsv

NegCtrl2        GCGATGGGGGGGTGGGTAG
NegCtrl3        GACGACTAGTTAGGCGTGT
DDRGK1  GCGGTCCACAAAGGCTCAG
UFL1    GTGACTCGCAGTAGACGCG
UFM1    GCGGTAAGCAAACACTTAC
SEC61G  GCTCCAGTGCTACGTGTCC
SEC61A1 GCTGTGCAGTGGAACGCGC
SRP68   GAGAAGCAGGTCCCAGGCG
SRPRB   GGCCACCCGGCGCGAGTCC
SRPR    GGCGAACGCGGCCTGAATT
SRP72   GCCTCCAAGATGGCGAGCG
TIMM23  GAAGTAGGCGCTGGCAACG
ATP5B   GAGTCTCCGCAAGGCCCCG
MRPL39  GTCTGGCTGGTCGCACCCG
TMED2   GTGAGGCCGAAGCCAGGAC
TMED10  GAGACTCGTTCACCACCGA
CARS    GAGCCATGGCAGATTCCTC
HARS    GCTCAAGTGGACAGCCGGG
QARS    GCGCGCTCAGTGAGAGGAA
TTI1    GAAGGCTGGAAGACGAGGT
TELO2   GCCGCGGAGACCCGCCCCA
TTI2    GTCCGGATCCTGTTAGACA
TMEM167A        GCAGCCACATCACCCTTCC
YIPF5   GGGTGCAGGGGACCGCGTC
SCYL1   GGCCGGAGGACCCGGAGCT
IER3IP1 GGGGCCCCATCGGCTTCCG
```

(continues on next page)

(continued from previous page)

```
DDOST   GTGGGTCCTTCGGCAGGAG
DAD1    GACCTTGCGTGCAGTTATG
OST4    GGCTTGTTCGCTGGTGGCG
EIF2B4  GCTGAGGGCGATGGCTGCT
EIF2B2  GTAGCTGCCTTCAGCCTTC
EIF2B3  GCCATTGGGCTGTCAGTCA
```

First we screen reads that have the constant sequence (`GTACATGGGG`) upstream of sgRNAs on read 2 (cutadapt, version 3.7).

```
$ cutadapt \
    --cores 0 \
    --front GTACATGGGG \
    --length 50 \
    --minimum-length 50:26 \
    --trimmed-only \
    --output read_2_trimmed.fq.gz --paired-output read_1_trimmed.fq.gz \
    GSM4367980_2.fq.gz GSM4367980_1.fq.gz
```

Preview the filtering result: 104,375,315 out of 404,963,129 (25.8%) read pairs are kept for sgRNA identification.

```
== Read fate breakdown ==
Pairs that were too short:             377,332 (0.1%)
Pairs discarded as untrimmed:      300,210,482 (74.1%)
Pairs written (passing filters):   104,375,315 (25.8%)
```

QC

The first 200,000 read pairs are sampled (set by `-n`, default `100,000`) for quality control. The `-t` option can be used to set the number of threads. By default, diagnostic results and plots are generated in the `qc` directory (set by `--output_directory`), and the full length of read 1 and read 2 are searched against reference cell and feature barcodes, respectively. The per base content of both read pairs and the distribution of matched barcode positions are summarized. Use `-r1_c` and/or `-r2_c` to limit the search range, and `-cb_n` and/or `-fb_n` to set the mismatch tolerance for cell and/or feature barcode matching (default 3).

```
$ fba qc \
    -1 read_1_trimmed.fq.gz \
    -2 read_2_trimmed.fq.gz \
    -w cell_barcodes.txt \
    -f feature_barcodes_UPR_edited.tsv \
    -r1_c 0,16 \
    -n 200000
```

This library is built using the Chromium Single Cell 3' Solution v3 and sequenced on Illumina NovaSeq 6000. The first 16 bases are cell barcodes and the following 10 bases are UMIs. Based on the base content plot, the GC content of cell barcodes are quite even. The UMIs are slightly T enriched.

As for read 2, based on the per base content, it suggests that read 2 is slightly A enriched.

The detailed `qc` results are stored in the `feature_barcoding_output.tsv.gz` file. The `matching_pos` columns indicate the matched positions on reads, while the `matching_description` columns indicate mismatches in the format of substitutions:insertions:deletions.

```
$ gzip -dc feature_barcoding_output.tsv.gz | grep -v no_ | head

read1_seq       cell_barcode    cb_matching_pos cb_matching_description read2_seq
↪feature_barcode fb_matching_pos fb_matching_description
GTGTCCTGTCGCGCATaggacttccg      GTGTCCTCACGCGCAT        0:16    2:0:0
↪GTGACTCGCAGTAGACGCGGGTTTAAGAGCTAAGCTGGAAACAGCATAGC      UFL1_GTGACTCGCAGTAGACGCG
↪ 0:19    0:0:0
CGGAGAAAGACCTGTCggtatgggac      CGGAGAATCACCTGTC        0:16    2:0:0
↪GGCTTGTTCGCTGGTGGCGTGTTTAAGAGCTAAGCTGGAAACAGCATAGC      OST4_GGCTTGTTCGCTGGTGGCG
↪ 0:19    0:0:0
TAGTGCAGTGCATGCCccgaatgttt      TAGTGCAGTGGTATGG        0:15    2:0:1
↪GGGCCGGAGGACCCGGAGCTAGTTTAAGAGCTAAGCTGGAAACAGCATAG      SCYL1_GGCCGGAGGACCCGGAGCT
↪ 1:20    0:0:0
GTCATCCGTTGACTACgggggccact      ATCCTATGTTGACTAC        3:16    0:0:3
↪GGCTTGTTCGCTGGTGGCGTGTTTAAGAGCTAAGCTGGAAACAGCATAGC      OST4_GGCTTGTTCGCTGGTGGCG
↪ 0:19    0:0:0
ATCGCCTCAAGGATATttcagattaa      TCGACCTCAAGAATGT        1:16    2:0:1
↪GAGCCATGGCAGATTCCTCCGTTTAAGAGCTAAGCTGGAAACAGCATAGC      CARS_GAGCCATGGCAGATTCCTC
↪ 0:19    0:0:0
GTTGCGGGTCGCCACAgtacatactt      GTTGCGGCACGCCACA        0:16    2:0:0
↪GCGGTCCACAAAGGCTCAGAGTTTAAGAGCTAAGCTGGAAACAGCATAGC      DDRGK1_GCGGTCCACAAAGGCTCAG
↪ 0:19    0:0:0
TGCATGATCGTGATCGtggagaaagt      AGTGATCAGGTGATCG        3:16    0:0:3
↪CTCCAGTGCTACGTGTCCCGTTTAAGAGCTAAGCTGGAAACAGCATAGCA      SEC61G_GCTCCAGTGCTACGTGTCC
↪ 0:18    0:0:1
CTCCCAAAGCCGTGTTcatcgatatt      CTCCCAAAGACCTTTG        0:14    1:0:2
↪GTGACTCGCAGTAGACGCGGGTTTAAGAGCTAAGCTGGAAACAGCATAGC      UFL1_GTGACTCGCAGTAGACGCG
↪ 0:19    0:0:0
TTTGGTTGTCGACAGAttacgcgttt      TTGGTTTGTCGCACAC        1:15    1:0:2
↪GACGACTAGTTAGGCGTGTAGTTTAAGAGCTAAGCTGGAAACAGCATAGC      NegCtrl3_GACGACTAGTTAGGCGTGT
↪ 0:19    0:0:0
```

Barcode extraction

Search ranges are set to `0,16` on read 1 and `0,19` on read 2. Two mismatches for cell and feature barcodes (`-cb_m`, `-cf_m`) are allowed.

```
$ fba extract \
    -1 read_1_trimmed.fq.gz \
    -2 read_2_trimmed.fq.gz \
    -w cell_barcodes.txt \
    -f feature_barcodes_UPR_edited.tsv \
    -o feature_barcoding_output.tsv.gz \
    -r1_c 0,16 \
    -r2_c 0,19 \
    -cb_m 2 \
    -fb_m 2
```

Preview of result.

```
$ gzip -dc feature_barcoding_output.tsv.gz | head

read1_seq       cell_barcode    cb_num_mismatches       read2_seq       feature_barcode␣
↪fb_num_mismatches
GTGTCCTGTCGCGCATaggacttccg      GTGTCCTCACGCGCAT        2       ␣
↪GTGACTCGCAGTAGACGCGggtttaagagctaagctggaaacagcatagc     UFL1_GTGACTCGCAGTAGACGCG        0
CGGAGAAAGACCTGTCggtatgggac      CGGAGAATCACCTGTC        2       ␣
↪GGCTTGTTCGCTGGTGGCGtgtttaagagctaagctggaaacagcatagc     OST4_GGCTTGTTCGCTGGTGGCG        0
GTTGCGGGTCGCCACAgtacatactt      GTTGCGGCACGCCACA        2       ␣
↪GCGGTCCACAAAGGCTCAGagtttaagagctaagctggaaacagcatagc     DDRGK1_GCGGTCCACAAAGGCTCAG      0
TTTGGTTGTCGACAGAttacgcgttt      TTTGGTTCACGACAGA        2       ␣
↪GACGACTAGTTAGGCGTGTagtttaagagctaagctggaaacagcatagc     NegCtrl3_GACGACTAGTTAGGCGTGT    0
TCGTGGGAGGGAAACGcatggtcgaa      TCGTGGGTCGGAAACG        2       ␣
↪GTCTGGCTGGTCGCACCCGggtttaagagctaagctggaaacagcatagc     MRPL39_GTCTGGCTGGTCGCACCCG      0
ACAGCCGTCTTGCTCAtttaacaggc      ACAGCCGAGTTGCTCA        2       ␣
↪GACCTTGCGTGCAGTTATGtgtttaagagctaagctggaaacagcatagc     DAD1_GACCTTGCGTGCAGTTATG        0
CCGTAGGAGTGCGGCAgccgagcaac      CCGTAGGTCTGCGGCA        2       ␣
↪GACGACTAGTTAGGCGTGTagtttaagagctaagctggaaacagcatagc     NegCtrl3_GACGACTAGTTAGGCGTGT    0
GTCTCACTCAGGACTCtatccatcca      GTCTCACAGAGGACTC        2       ␣
↪GCGAACGCGGCCTGAATTCcgtttaagagctaagctggaaacagcatagc     SRPR_GGCGAACGCGGCCTGAATT        2
AAGACAACATTCGCTCtctaactgca      AAGACAAGTTTCGCTC        2       ␣
↪GAGTCTCCGCAAGGCCCCGggtttaagagctaagctggaaacagcatagc     ATP5B_GAGTCTCCGCAAGGCCCCG       0
```

Result summary.

52,352,330 out of 104,375,315 read pairs have valid cell and feature barcodes.

```
2022-03-06 03:44:57,488 - fba.__main__ - INFO - fba version: 0.0.x
2022-03-06 03:44:57,488 - fba.__main__ - INFO - Initiating logging ...
2022-03-06 03:44:57,489 - fba.__main__ - INFO - Python version: 3.10
2022-03-06 03:44:57,489 - fba.__main__ - INFO - Using extract subcommand ...
2022-03-06 03:44:57,504 - fba.levenshtein - INFO - Number of reference cell barcodes: 8,
↪727
2022-03-06 03:44:57,504 - fba.levenshtein - INFO - Number of reference feature barcodes:␣
↪32
2022-03-06 03:44:57,504 - fba.levenshtein - INFO - Read 1 coordinates to search: [0, 16)
```

(continues on next page)

(continued from previous page)

```
2022-03-06 03:44:57,504 - fba.levenshtein - INFO - Read 2 coordinates to search: [0, 19)
2022-03-06 03:44:57,504 - fba.levenshtein - INFO - Cell barcode maximum number of␣
↪mismatches: 2
2022-03-06 03:44:57,504 - fba.levenshtein - INFO - Feature barcode maximum number of␣
↪mismatches: 2
2022-03-06 03:44:57,504 - fba.levenshtein - INFO - Read 1 maximum number of N allowed: 3
2022-03-06 03:44:57,504 - fba.levenshtein - INFO - Read 2 maximum number of N allowed: 3
2022-03-06 03:44:58,965 - fba.levenshtein - INFO - Matching ...
2022-03-06 04:02:50,201 - fba.levenshtein - INFO - Read pairs processed: 10,000,000
2022-03-06 04:21:17,938 - fba.levenshtein - INFO - Read pairs processed: 20,000,000
2022-03-06 04:40:47,371 - fba.levenshtein - INFO - Read pairs processed: 30,000,000
2022-03-06 05:00:15,184 - fba.levenshtein - INFO - Read pairs processed: 40,000,000
2022-03-06 05:19:43,813 - fba.levenshtein - INFO - Read pairs processed: 50,000,000
2022-03-06 05:39:14,583 - fba.levenshtein - INFO - Read pairs processed: 60,000,000
2022-03-06 05:58:41,750 - fba.levenshtein - INFO - Read pairs processed: 70,000,000
2022-03-06 06:18:09,714 - fba.levenshtein - INFO - Read pairs processed: 80,000,000
2022-03-06 06:37:33,602 - fba.levenshtein - INFO - Read pairs processed: 90,000,000
2022-03-06 06:56:58,484 - fba.levenshtein - INFO - Read pairs processed: 100,000,000
2022-03-06 07:05:24,748 - fba.levenshtein - INFO - Number of read pairs processed: 104,
↪375,315
2022-03-06 07:05:24,771 - fba.levenshtein - INFO - Number of read pairs w/ valid␣
↪barcodes: 52,352,330
2022-03-06 07:05:24,834 - fba.__main__ - INFO - Done.
```

Matrix generation

Only fragments with correctly matched cell and feature barcodes are included, while fragments with UMI lengths less than the specified value are discarded. UMI removal is performed using UMI-tools (Smith, T., et al. 2017. Genome Res. 27, 491–499.), with the starting position on read 1 set by `-us` (default `16`) and the length set by `-ul` (default `12`). The UMI deduplication method can be set using `-ud` (default `directional`), and the UMI deduplication mismatch threshold can be specified using `-um` (default `1`).

The generated feature count matrix can be easily imported into well-established single cell analysis packages: Seurat and Scanpy.

```
$ fba count \
    -i feature_barcoding_output.tsv.gz \
    -o matrix_featurecount.csv.gz \
    -us 16 \
    -ul 10
```

Result summary.

10.7% (5,581,448 out of 52,352,330) of read pairs with valid cell and feature barcodes are unique fragments. 1.4% (5,581,448 out of 404,963,129) of total sequenced read pairs contribute to the final matrix. The meidan number of UMIs of sgRNA per cell is 413.0.

```
2022-03-06 07:05:24,972 - fba.__main__ - INFO - fba version: 0.0.x
2022-03-06 07:05:24,973 - fba.__main__ - INFO - Initiating logging ...
```

(continues on next page)

(continued from previous page)

```
2022-03-06 07:05:24,973 - fba.__main__ - INFO - Python version: 3.10
2022-03-06 07:05:24,973 - fba.__main__ - INFO - Using count subcommand ...
2022-03-06 07:05:26,523 - fba.count - INFO - UMI-tools version: 1.1.2
2022-03-06 07:05:26,525 - fba.count - INFO - UMI starting position on read 1: 16
2022-03-06 07:05:26,526 - fba.count - INFO - UMI length: 10
2022-03-06 07:05:26,526 - fba.count - INFO - UMI-tools deduplication threshold: 1
2022-03-06 07:05:26,526 - fba.count - INFO - UMI-tools deduplication method: directional
2022-03-06 07:05:26,526 - fba.count - INFO - Header line: read1_seq cell_barcode cb_num_
↪mismatches read2_seq feature_barcode fb_num_mismatches
2022-03-06 07:06:52,385 - fba.count - INFO - Number of lines processed: 52,352,330
2022-03-06 07:06:52,394 - fba.count - INFO - Number of cell barcodes detected: 8,670
2022-03-06 07:06:52,394 - fba.count - INFO - Number of features detected: 32
2022-03-06 07:11:58,774 - fba.count - INFO - Total UMIs after deduplication: 5,581,448
2022-03-06 07:11:58,776 - fba.count - INFO - Median number of UMIs per cell: 413.0
2022-03-06 07:11:59,297 - fba.__main__ - INFO - Done.
```

Demultiplexing

Poisson-Gaussian mixture model

The implementation of demultiplexing method `3` (set by `-dm`) is inspired by Replogle, M., et al. (2021). To set the probability threshold for demultiplexing, use `-p` (default `0.9`). To specify the minimum number of positive cells for a given feature to be considered during demultiplexing, use `-nc` (default `200`).

```
$ fba demultiplex \
    -i matrix_featurecount.csv.gz \
    -dm 3 \
    -v \
    -nc 0
```

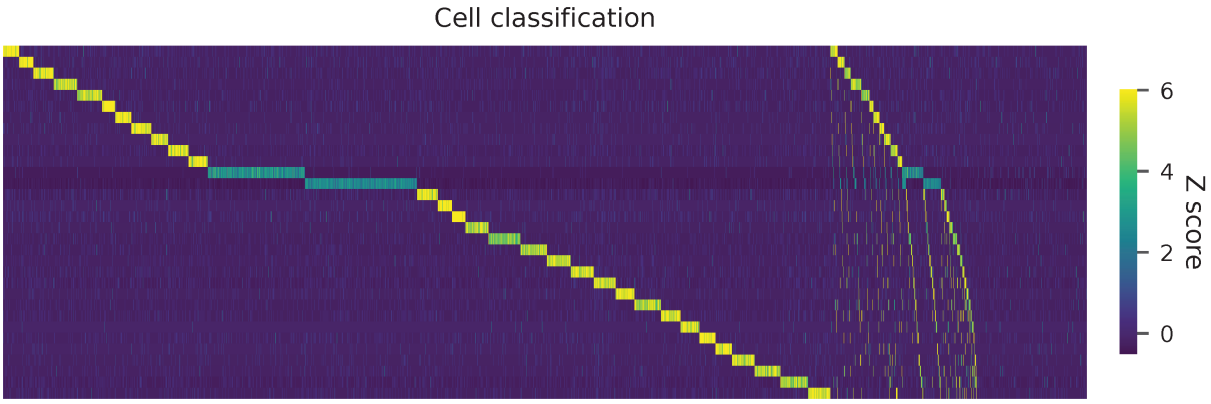
```
2022-03-06 14:07:02,328 - fba.__main__ - INFO - fba version: 0.0.x
2022-03-06 14:07:02,328 - fba.__main__ - INFO - Initiating logging ...
2022-03-06 14:07:02,328 - fba.__main__ - INFO - Python version: 3.9
2022-03-06 14:07:02,328 - fba.__main__ - INFO - Using demultiplex subcommand ...
2022-03-06 14:07:04,814 - fba.__main__ - INFO - Skipping arguments: "-q/--quantile", "-
↪cm/--clustering_method"
2022-03-06 14:07:04,814 - fba.demultiplex - INFO - Output directory: demultiplexed
2022-03-06 14:07:04,814 - fba.demultiplex - INFO - Demultiplexing method: 3
2022-03-06 14:07:04,814 - fba.demultiplex - INFO - UMI normalization method: clr
2022-03-06 14:07:04,814 - fba.demultiplex - INFO - Visualization: On
2022-03-06 14:07:04,814 - fba.demultiplex - INFO - Visualization method: tsne
2022-03-06 14:07:04,814 - fba.demultiplex - INFO - Loading feature count matrix: matrix_
↪featurecount.csv.gz ...
2022-03-06 14:07:04,902 - fba.demultiplex - INFO - Number of cells: 8,670
2022-03-06 14:07:04,902 - fba.demultiplex - INFO - Number of positive cells for a␣
↪feature to be included: 0
2022-03-06 14:07:04,916 - fba.demultiplex - INFO - Number of features: 32 / 32 (after␣
```

(continues on next page)

(continued from previous page)

```
↪filtering / original in the matrix)
2022-03-06 14:07:04,935 - fba.demultiplex - INFO - Features: ATP5B CARS DAD1 DDOST␣
↪DDRGK1 EIF2B2 EIF2B3 EIF2B4 HARS IER3IP1 MRPL39 NegCtrl2 NegCtrl3 OST4 QARS SCYL1␣
↪SEC61A1 SEC61G SRP68 SRP72 SRPRB SRPR TELO2 TIMM23 TMED10 TMED2 TMEM167A TTI1 TTI2␣
↪UFL1 UFM1 YIPF5
2022-03-06 14:07:04,935 - fba.demultiplex - INFO - Total UMIs: 5,581,448 / 5,581,448
2022-03-06 14:07:04,943 - fba.demultiplex - INFO - Median number of UMIs per cell: 413.0␣
↪/ 413.0
2022-03-06 14:07:04,957 - fba.demultiplex - INFO - Demultiplexing ...
2022-03-06 14:07:47,811 - fba.demultiplex - INFO - Generating heatmap ...
2022-03-06 14:07:57,340 - fba.demultiplex - INFO - Embedding ...
2022-03-06 14:08:12,180 - fba.__main__ - INFO - Done.
```

Heatmap of the relative abundance of features (sgRNAs) across all cells. Each column represents a single cell.



Preview the demultiplexing result (Fig. 2d, middle column): the numbers of singlets, multiplets and negatives are 6,618 (76.3%), 1,171 (13.5%), and 881 (10.1%), respectively.

```
In [1]: import pandas as pd

In [2]: m = pd.read_csv("demultiplexed/matrix_cell_identity.csv.gz", index_col=0)

In [3]: m.loc[:, m.sum(axis=0) == 1].sum(axis=1)
Out[3]:
ATP5B       130
CARS        115
DAD1        163
DDOST       186
DDRGK1      200
EIF2B2      104
EIF2B3      129
EIF2B4      160
HARS        136
IER3IP1     162
MRPL39      153
NegCtrl2    777
NegCtrl3    898
OST4        167
```

(continues on next page)

(continued from previous page)

```
QARS         113
SCYL1        108
SEC61A1      183
SEC61G       256
SRP68        212
SRP72        191
SRPRB        184
SRPR         175
TELO2        152
TIMM23       211
TMED10       156
TMED2        152
TMEM167A     127
TTI1         132
TTI2         183
UFL1         204
UFM1         223
YIPF5        176
dtype: int64

In [4]: sum(m.sum(axis=0) == 1)
Out[4]: 6618

In [5]: sum(m.sum(axis=0) > 1)
Out[5]: 1171

In [6]: sum(m.sum(axis=0) == 0)
Out[6]: 881

In [7]: m.shape
Out[7]: (32, 8670)
```

t-SNE embedding of cells based on the abundance of features (sgRNAs, no transcriptome information used). Colors indicate the sgRNA status for each cell, as called by FBA.

- *10k A375 Cells Transduced with (1) Non-Target and (1) Target sgRNA, Dual Indexed*
- *CROP-seq; 1:1:1 Mixture of DNMT3B, MBD1, and TET2 Knockout Cell Lines (HEK293T)*
- *Direct-capture Perturb-seq; CRISPRi-based Screen of Unfolded Protein Response (UPR) Using 3' sgRNA-CR1cs1*

3.2 Cell surface protein labeling

3.2.1 CITE-seq; 8k Cord Blood Mononuclear Cells with 13 Antibodies

Dataset: CITE-seq

Stoeckius, M., Hafemeister, C., Stephenson, W., Houck-Loomis, B., Chattopadhyay, P.K., Swerdlow, H., Satija, R., and Smibert, P. (2017). Simultaneous epitope and transcriptome measurement in single cells. *Nat. Methods* **14**, 865–868.

Preparation

Download fastq files from European Nucleotide Archive.

```
$ curl -O ftp.sra.ebi.ac.uk/vol1/fastq/SRR580/000/SRR5808750/SRR5808750_1.fastq.gz

$ curl -O ftp.sra.ebi.ac.uk/vol1/fastq/SRR580/000/SRR5808750/SRR5808750_2.fastq.gz
```

Download cell barcode info from Gene Expression Omnibus. These are the cell-associated barcodes in this single cell RNA-Seq library.

```
$ wget https://ftp.ncbi.nlm.nih.gov/geo/series/GSE100nnn/GSE100866/suppl/GSE100866_CBMC_
↪8K_13AB_10X-ADT_umi.csv.gz

$ gzip -dc GSE100866_CBMC_8K_13AB_10X-ADT_umi.csv.gz | head -1 | sed 's/,/\n/g' | awk 'NF
↪' > cell_barcodes.txt
```

Inspect cell barcodes.

```
$ head cell_barcodes.txt

CTGTTTACACCGCTAG
CTCTACGGTGTGGCTC
AGCAGCCAGGCTCATT
GAATAAGAGATCCCAT
GTGCATAGTCATGCAT
TACACGACACATCCGG
TCATTTGGTGTGAAAT
GGGCACTGTGAAGGCT
AACACGTCATTAACCG
CACATAGCAATGCCAT
```

Prepare feature barcodes (antibody-oligo sequences, from the online methods section of the paper).

```
$ cat feature_barcodes.tsv

v2_BC1  ATCACG
v2_BC2  CGATGT
v2_BC3  TTAGGC
v2_BC4  TGACCA
```

(continues on next page)

(continued from previous page)

```
v2_BC5  ACAGTG
v2_BC6  GCCAAT
v2_BC8  ACTTGA
v2_BC9  GATCAG
v2_BC10 TAGCTT
v2_BC11 GGCTAC
v2_BC12 CTTGTA
v2_BC13 AGTCAA
v2_BC14 AGTTCC
```

QC

The first 100,000 read pairs are sampled (default, set by `-n`) for quality control. By default, diagnostic results and plots are generated in the `qc` directory (set by `--output_directory`), and the full length of read 1 and read 2 are searched against reference cell and feature barcodes, respectively. The per base content of both read pairs and the distribution of matched barcode positions are summarized. Use `-r1_c` and/or `-r2_c` to limit the search range, and `-cb_n` and/or `-fb_n` to set the mismatch tolerance for cell and/or feature barcode matching (default `3`).

```
$ fba qc \
    -1 SRR5808750_1.fastq.gz \
    -2 SRR5808750_2.fastq.gz \
    -w cell_barcodes.txt \
    -f feature_barcodes.tsv \
    --output_directory qc
```

This library was constructed using the Chromium Single Cell 3' Reagent Kits (v2 Chemistry). The first 16 bases represent cell barcodes and the next 9 bases are UMIs (Read 1 has a length of 25). The base content plot shows that the GC content of cell barcodes is evenly distributed. Additionally, the UMIs are slightly enriched in G.

Based on the per base content analysis of read 2, it appears that bases 0-5 correspond to the feature barcodes, as evidenced by the distribution of matched barcode positions.

The detailed `qc` results are stored in the `feature_barcoding_output.tsv.gz` file. The `matching_pos` columns indicate the matched positions on reads, while the `matching_description` columns indicate mismatches in the format of substitutions:insertions:deletions.

```
$ gzip -dc qc/feature_barcoding_output.tsv.gz | head

read1_seq       cell_barcode    cb_matching_pos cb_matching_description read2_seq      ␣
↪feature_barcode fb_matching_pos fb_matching_description
CGTAGCGGTTAGTGGGGGGTGGCGC       CGTAGCGGTTAGTGGG        0:16    0:0:0   ␣
↪CCCAGGGCCTCGTGGCCGGAGAAACTGCCCA v2_BC1_ATCACG   0:6     3:0:0
```

(continues on next page)

(continued from previous page)

```
ACACTGAGTCCGAAGACTCGTTTGA       ACTGAGTCAGTACACT        2:18    3:0:0  ␣
↪GATCAGCAAAAAAAAAAAAAAAAAAAAAAAA v2_BC9_GATCAG   0:6     0:0:0
TTCTTAGGTGGCCCTATCCAGAGAG       GCTCCTATCAGAGACG        10:25   0:1:2  ␣
↪ACAGTGCAAAAAAAAAAAAAAAAAAAAAAAA v2_BC5_ACAGTG   0:6     0:0:0
GATCAGTGTCTAAAGATCCGGGCGC       GATCAGTGTCGAAAGC        0:16    2:0:0  ␣
↪AGTCAATAAAAAAGAAAAAAAAAAAAAAAAA v2_BC13_AGTCAA  0:6     0:0:0
AACTGGTTCTGGTGTAGATGAATCA       CTGGTCTAGAGTAATC        8:24    3:0:0  ␣
↪ATCACGGAAAAAAAAAAAAAAAAAAAAAAAA v2_BC1_ATCACG   0:6     0:0:0
TCAGGATGTTGATTCGACAGTGAAA       AGGGATGTCTGATTCT        2:16    1:0:2  ␣
↪GATCAGGAAAAAAAAAAAAAAAAAAAAAAAA v2_BC9_GATCAG   0:6     0:0:0
CTCGAGGAGCCACCTGGAGATACAG       CTCCTAGAGCCACCTG        0:16    3:0:0  ␣
↪ACAGTGCAAAAAAAAAAAAAAAAAAAAAAAA v2_BC5_ACAGTG   0:6     0:0:0
GACCTGGTCTTAGCCCCCCAGATCA       GACCTGGTCTTAACCT        0:16    2:0:0  ␣
↪GCCAATCAAAAAAAAAAAAAAAAAAAAAAAA v2_BC6_GCCAAT   0:6     0:0:0
NCGGCCAGGGTGTAGCACGGATTGC       CTGATAGCAGGGATTG        9:24    2:0:1  ␣
↪ACAGTGCAAAAAAAAAAAAAAAAAAAAAAAA v2_BC5_ACAGTG   0:6     0:0:0
```

Barcode extraction

The lengths of cell and feature barcodes are all identical (16 and 6, respectively). And based on the `qc` results, the distributions of starting and ending positions of cell and feature barcodes are very uniform. Search ranges are set to `0,16` on read 1 and `0,6` on read 2. One mismatch for cell and feature barcodes (`-cb_m`, `-cf_m`) are allowed. And by default, three ambiguous nucleotides (Ns) for read 1 and read 2 (`-cb_n`, `-cf_n`) are allowed.

```
$ fba extract \
    -1 SRR5808750_1.fastq.gz \
    -2 SRR5808750_2.fastq.gz \
    -w cell_barcodes.txt \
    -f feature_barcodes.tsv \
    -o feature_barcoding_output.tsv.gz \
    -r1_c 0,16 \
    -r2_c 0,6 \
    -cb_m 1 \
    -fb_m 1 \
    -cb_n 3 \
    -fb_n 3
```

Preview of result.

```
$ gzip -dc feature_barcoding_output.tsv.gz | head

read1_seq       cell_barcode    cb_num_mismatches       read2_seq       feature_barcode␣
↪fb_num_mismatches
GGCGTGTGTCCATGATtcatgtatg       GGCGTGTGTCCATGAT        0       ␣
↪ACAGTGcaaaaaaaaaaaaaaaaaaaaaaaa v2_BC5_ACAGTG   0
CGACCTTCATAGACTCtacctcgcg       CGACCTTCATAGACTC        0       ␣
↪AGTCAAgaaaaaaaaaaaaaaaaaaaaaaaa v2_BC13_AGTCAA  0
CTGATCCTCAATAAGGtcgtttgga       CTGATCCTCAATAAGG        0       ␣
```

(continues on next page)

(continued from previous page)

```
↪ACAGTGgaaaaaaaaaaaaaaaaaaaaaaaa v2_BC5_ACAGTG   0
TTGACTTTCACGACTAagtttgggg       TTGACTTTCACGACTA        0       ␣
↪AGTCAAtaaaaaaaaaaaaaaaaaaaaaaaa v2_BC13_AGTCAA  0
CGGAGTCAGGAGCGTTatccgtaat       CGGAGTCAGGAGCGTT        0       ␣
↪ACAGTGgaaaaaaaaaaaaaaaaaaaaaaaa v2_BC5_ACAGTG   0
TTTGGTTGTAGAGCTGgggcaagta       TTTGGTTGTAGAGCTG        0       ␣
↪ACAGTGcaaaaaaaaaaaaaaaaaaaaaaaa v2_BC5_ACAGTG   0
ACCCACTAGACCCACCgaaccttta       ACCCACTAGACCCACC        0       ␣
↪GCCAATtaaaaaaaaaaaaaaaaaaaaaaaa v2_BC6_GCCAAT   0
CGGGTCATCTGCGACGgccttttttt      CGGGTCATCTGCGACG        0       ␣
↪ACAGTGtaaaaaaaaaaaaaaaaaaaaaaaa v2_BC5_ACAGTG   0
CACATTTGTCATCCCTaccatccgc       CACATTTGTCATCCCT        0       ␣
↪ATCACGcaaaaaaaaaaaaaaaaaaaaaaaa v2_BC1_ATCACG   0
```

Result summary.

30.4% (63,063,944 out of 207,724,395) of total read pairs have valid cell and feature barcodes.

```
2021-02-17 23:47:41,923 - fba.__main__ - INFO - fba version: 0.0.7
2021-02-17 23:47:41,923 - fba.__main__ - INFO - Initiating logging ...
2021-02-17 23:47:41,923 - fba.__main__ - INFO - Python version: 3.7
2021-02-17 23:47:41,923 - fba.__main__ - INFO - Using extract subcommand ...
2021-02-17 23:47:41,928 - fba.levenshtein - INFO - Number of reference cell barcodes: 8,
↪617
2021-02-17 23:47:41,928 - fba.levenshtein - INFO - Number of reference feature barcodes:␣
↪13
2021-02-17 23:47:41,928 - fba.levenshtein - INFO - Read 1 coordinates to search: [0, 16)
2021-02-17 23:47:41,928 - fba.levenshtein - INFO - Read 2 coordinates to search: [0, 6)
2021-02-17 23:47:41,928 - fba.levenshtein - INFO - Cell barcode maximum number of␣
↪mismatches: 1
2021-02-17 23:47:41,928 - fba.levenshtein - INFO - Feature barcode maximum number of␣
↪mismatches: 1
2021-02-17 23:47:41,928 - fba.levenshtein - INFO - Read 1 maximum number of N allowed: 3
2021-02-17 23:47:41,928 - fba.levenshtein - INFO - Read 2 maximum number of N allowed: 3
2021-02-17 23:47:42,231 - fba.levenshtein - INFO - Matching ...
2021-02-17 23:51:17,514 - fba.levenshtein - INFO - Read pairs processed: 10,000,000
2021-02-17 23:54:52,641 - fba.levenshtein - INFO - Read pairs processed: 20,000,000
2021-02-17 23:58:27,676 - fba.levenshtein - INFO - Read pairs processed: 30,000,000
2021-02-18 00:02:02,380 - fba.levenshtein - INFO - Read pairs processed: 40,000,000
2021-02-18 00:05:36,809 - fba.levenshtein - INFO - Read pairs processed: 50,000,000
2021-02-18 00:09:11,444 - fba.levenshtein - INFO - Read pairs processed: 60,000,000
2021-02-18 00:12:45,146 - fba.levenshtein - INFO - Read pairs processed: 70,000,000
2021-02-18 00:16:20,131 - fba.levenshtein - INFO - Read pairs processed: 80,000,000
2021-02-18 00:19:54,980 - fba.levenshtein - INFO - Read pairs processed: 90,000,000
2021-02-18 00:23:29,696 - fba.levenshtein - INFO - Read pairs processed: 100,000,000
2021-02-18 00:27:04,208 - fba.levenshtein - INFO - Read pairs processed: 110,000,000
2021-02-18 00:30:38,402 - fba.levenshtein - INFO - Read pairs processed: 120,000,000
2021-02-18 00:34:11,917 - fba.levenshtein - INFO - Read pairs processed: 130,000,000
2021-02-18 00:37:44,939 - fba.levenshtein - INFO - Read pairs processed: 140,000,000
2021-02-18 00:41:18,752 - fba.levenshtein - INFO - Read pairs processed: 150,000,000
2021-02-18 00:44:51,673 - fba.levenshtein - INFO - Read pairs processed: 160,000,000
2021-02-18 00:48:25,440 - fba.levenshtein - INFO - Read pairs processed: 170,000,000
```

(continues on next page)

(continued from previous page)

```
2021-02-18 00:51:58,766 - fba.levenshtein - INFO - Read pairs processed: 180,000,000
2021-02-18 00:55:32,141 - fba.levenshtein - INFO - Read pairs processed: 190,000,000
2021-02-18 00:59:05,690 - fba.levenshtein - INFO - Read pairs processed: 200,000,000
2021-02-18 01:01:50,228 - fba.levenshtein - INFO - Number of read pairs processed: 207,
↪724,395
2021-02-18 01:01:50,228 - fba.levenshtein - INFO - Number of read pairs w/ valid␣
↪barcodes: 63,063,944
2021-02-18 01:01:50,249 - fba.__main__ - INFO - Done.
```

Matrix generation

Only fragments with correctly matched cell and feature barcodes are included, while fragments with UMI lengths less than the specified value are discarded. UMI removal is performed using UMI-tools (Smith, T., et al. 2017. Genome Res. 27, 491–499.), with the starting position on read 1 set by `-us` (default `16`) and the length set by `-ul` (default `12`). The UMI deduplication method can be set using `-ud` (default `directional`), and the UMI deduplication mismatch threshold can be specified using `-um` (default `1`).

The generated feature count matrix can be easily imported into well-established single cell analysis packages such as Seurat and Scanpy.

```
$ fba count \
    -i feature_barcoding_output.tsv.gz \
    -o matrix_featurecount.csv.gz \
    -us 16 \
    -ul 9 \
    -um 1 \
    -ud directional
```

Result summary.

54.8% (34,574,243 out of 63,063,944) of read pairs with valid cell and feature barcodes are unique fragments. 16.6% (34,574,243 out of 207,724,395) of total sequenced read pairs contribute to the final matrix.

```
2021-02-18 01:16:22,447 - fba.__main__ - INFO - fba version: 0.0.7
2021-02-18 01:16:22,447 - fba.__main__ - INFO - Initiating logging ...
2021-02-18 01:16:22,447 - fba.__main__ - INFO - Python version: 3.7
2021-02-18 01:16:22,447 - fba.__main__ - INFO - Using count subcommand ...
2021-02-18 01:16:22,447 - fba.count - INFO - UMI-tools version: 1.1.1
2021-02-18 01:16:22,450 - fba.count - INFO - UMI starting position on read 1: 16
2021-02-18 01:16:22,450 - fba.count - INFO - UMI length: 9
2021-02-18 01:16:22,450 - fba.count - INFO - UMI-tools deduplication threshold: 1
2021-02-18 01:16:22,450 - fba.count - INFO - UMI-tools deduplication method: directional
2021-02-18 01:16:22,450 - fba.count - INFO - Header line: read1_seq cell_barcode cb_num_
↪mismatches read2_seq feature_barcode fb_num_mismatches
2021-02-18 01:18:58,245 - fba.count - INFO - Number of lines processed: 63,063,944
2021-02-18 01:18:58,260 - fba.count - INFO - Number of cell barcodes detected: 8,617
2021-02-18 01:18:58,261 - fba.count - INFO - Number of features detected: 13
2021-02-18 08:12:46,216 - fba.count - INFO - Total UMIs after deduplication: 34,574,243
```

(continues on next page)

(continued from previous page)

```
2021-02-18 08:12:46,244 - fba.count - INFO - Median number of UMIs per cell: 3,816.0
2021-02-18 08:12:46,435 - fba.__main__ - INFO - Done.
```

3.2.2 ASAP-seq; Multiplexed CRISPR Perturbations in Primary T Cells

Dataset: ASAP-seq: Multiplexed CRISPR Perturbations in Primary T Cells

Mimitou, E.P., Lareau, C.A., Chen, K.Y., Zorzetto-Fernandes, A.L., Hao, Y., Takeshima, Y., Luo, W., Huang, T.-S., Yeung, B.Z., Papalexi, E., et al. (2021). Scalable, multimodal profiling of chromatin accessibility, gene expression and protein levels in single cells. *Nat. Biotechnol.* **39**, 1246–1258.

Preparation

Download fastq files from Gene Expression Omnibus.

```
$ ls -l

SRR12476627_1.fastq.gz
SRR12476627_2.fastq.gz
SRR12476627_3.fastq.gz
SRR12476628_1.fastq.gz
SRR12476628_2.fastq.gz
SRR12476628_3.fastq.gz
SRR12476630_1.fastq.gz
SRR12476630_2.fastq.gz
SRR12476630_3.fastq.gz
SRR12476631_1.fastq.gz
SRR12476631_2.fastq.gz
SRR12476631_3.fastq.gz
```

Hashtag oligos

```
$ cat SRR12476630_2.fastq.gz SRR12476631_2.fastq.gz > hto_read_2.fq.gz

$ cat SRR12476630_3.fastq.gz SRR12476631_3.fastq.gz > hto_read_3.fq.gz
```

Antibody-derived tags

```
$ cat SRR12476627_2.fastq.gz SRR12476628_2.fastq.gz > adt_read_2.fq.gz

$ cat SRR12476627_3.fastq.gz SRR12476628_3.fastq.gz > adt_read_3.fq.gz
```

Download cell barcode info from the manuscript's wonderful GitHub repository. We need to get the cell-associated barcodes of this single-cell ATAC-Seq library.

```
$ wget https://github.com/caleblareau/asap_reproducibility/blob/master/CD4_CRISPR_
↪asapseq/data/filtered_peak_bc_matrix.h5

$ ls -l

filtered_peak_bc_matrix.h5
```

Retrieve cell-associated barcodes from this downloaded h5 file generated by 10x Genomics' Cell Ranger ATAC.

```
R version 4.1.3 (2022-03-10) -- "One Push-Up"
Platform: aarch64-apple-darwin21.3.0 (64-bit)

r$> h5f <- rhdf5::H5Fopen(name = "filtered_peak_bc_matrix.h5")

r$> cell_barcodes <- h5f$matrix$barcodes

r$> rhdf5::H5Fclose(h5f)

r$> write.table(
        x = cell_barcodes,
        file = "cell_barcodes.txt",
        col.names = FALSE,
        row.names = FALSE,
        quote = FALSE
    )
```

Inspect cell barcodes.

```
$ head cell_barcodes.txt

AAACGAAAGCTCGTTA-1
AAACGAAAGCTGAGGT-1
AAACGAAAGGTGAACC-1
AAACGAACAACATAAG-1
AAACGAACAATAGCGG-1
AAACGAACAATCCATG-1
AAACGAACACGTTAGT-1
AAACGAACAGAGATGC-1
AAACGAACAGCGTAGA-1
AAACGAAGTAAACGAT-1
```

Prepare feature barcodes (hashtag oligos, HTOs) from sheet 'Supplementary_1_Perturbation' in the supplementary table file.

```
$ cat feature_barcodes_HTO.tsv

anti-human_hashtag_1    GTCAACTCTTTAGCG
anti-human_hashtag_2    TGATGGCCTATTGGG
anti-human_hashtag_3    TTCCGCCTCTCTTTG
anti-human_hashtag_4    AGTAAGTTCAGCGTA
anti-human_hashtag_5    AAGTATCGTTTCGCA
anti-human_hashtag_12   TAACGACCAGCCATA
anti-human_hashtag_13   AAATCTCTCAGGCTC
```

Prepare feature barcodes (antibody-derived tags, ADTs) from sheet 'Supplementary_1_Hashing' in the supplementary table file.

```
$ cat feature_barcodes_ADT.tsv

UCHT1   CTCATTGTAACTCCT
RPA-T4  TGTTCCCGCTCAACT
SK1     GCGCAACTTGATGAT
HI100   TCAATCCTTCCGCTT
UCHL1   CTCCGAATCATGTTG
EH12.2H7        ACAGCGCCGTATTTA
A019D5  GTGTGTTGTCCTATG
DX2     CCAGCTCATTAGAGC
G043H7  AGTTCAGTCAACCGA
WM59    ACCTTTATGCCACGG
BC96    TTTGTCCTGTACGCC
QA17A04 AACTCCCTATGGAGG
FN50    GTCTCTTGGCTTAAA
O323    GCACTCCTGCATGTA
C398.4A CGCGCACCCATTAAA
BJ18    AATCCTTCCGAATGT
A15153G TTGCTTACCGCCAGA
CD7-6B7 TGGATTCCCGGACTT
Ber-ACT35_(ACT35)       AACCCACCGTTGTTA
DREG-56 GTCCCTGCAACTTGA
A1      TTACCTGGTATCCGT
4B4-1   CAGTAAGTTCGGGAC
108-17  ACCTTTCGACACTCG
CD28.2  TGAGAACGACCCTAA
HIT2    TGTACCCGCTTGTGA
AD2     CAGTTCCTCAGTTCG
HP-3G10 GTACGCAGTCCTTCT
HIB19   CTGGGCAATTACTCG
2D1     TCCCTTGCGATTTAC
M5E2    TCTCAGACCTCCGTA
5.1H11  TCCTTTCCTGATAGG
3G8     AAGTTCACTCTTTGC
IP26    CGTAACGTAGAGCGA
F38-2E2 TGTCCTACCCAACTT
J252D4  AATTCAACCGTCGCC
S-HCL-3 TACGCCTATAACTTG
MOPC-21 GCCGGACGACATTAA
```

Cell hashing

QC

The first 100,000 read pairs are sampled (default, set by `-n`) for quality control. By default, diagnostic results and plots are generated in the `qc` directory (set by `--output_directory`), and the full length of read 1 and read 2 are searched against reference cell and feature barcodes, respectively. The per base content of both read pairs and the distribution of matched barcode positions are summarized. Use `-r1_c` and/or `-r2_c` to limit the search range, and `-cb_n` and/or `-fb_n` to set the mismatch tolerance for cell and/or feature barcode matching (default 3).

This library was constructed using 10x Genomics' Chromium Single Cell ATAC Reagent Kits, where the 16-base pair cell barcode is sequenced during the i5 index read. In Cell Ranger ATAC, the raw 16 bp sequences may be transformed into their reverse-complement counterparts as cell barcodes in the outputs. To achieve the same result in `fba`, use `-cb_rc` to reverse-complement the cell barcode sequences.

```
$ fba qc \
    -1 hto_read_2.fq.gz \
    -2 hto_read_3.fq.gz \
    -w cell_barcodes.txt \
    -f feature_barcodes_HTO.tsv \
    --output_directory qc \
    -cb_rc
```

This library was constructed using the Chromium Single Cell ATAC Reagent Kits and sequenced on the Illumina NextSeq 550 platform. The base composition analysis reveals that the cell barcodes in read 2 are enriched for A bases.

The per base content of read 3 suggests that the feature barcodes are located in bases 0-14, as indicated by the distribution of matched barcode positions (See the fba `qc` results for more details).

The detailed `qc` results are stored in the `feature_barcoding_output.tsv.gz` file. The `matching_pos` columns indicate the matched positions on reads, while the `matching_description` columns indicate mismatches in the format of substitutions:insertions:deletions.

```
$ gzip -dc feature_barcoding_output.tsv.gz | grep -v no | head

read1_seq       cell_barcode    cb_matching_pos cb_matching_description read2_seq      ␣
↪feature_barcode fb_matching_pos fb_matching_description
TTTAAGCTGCCTAACA        TTGTTAGCTGCCCAAC        0:15    2:0:1   ␣
↪TAACGACCAGCCATNNNANAANNANNANNNANNNNANNNNNANNNNNNNANNNNNNNNNNNNNNNNANNNN        anti-
↪human_hashtag_12_TAACGACCAGCCATA   0:15       1:0:0
AGAACGCGAAAAGGTT        AGAACGCGAGTACGTT        0:16    3:0:0   ␣
↪TGATGACCTATTGGNNNAAAANNANNANNNANNNNANANNNANNNANNNANNNNNNANNNNNNNNNNNANNAN        anti-
↪human_hashtag_2_TGATGGCCTATTGGG    0:15       2:0:0
TGAGACTTGGCAGGAT        TGAGACTTGGCAGGAT        0:16    0:0:0   ␣
↪TTTCGCCTTTCTTTGNAAAAANNANNAANAANNANANANAAANNAAANNAANANNAAAAANNANNNNAANAN        anti-
↪human_hashtag_3_TTCCGCCTCTCTTTG    0:15       2:0:0
ATTTATTGACGCAAAG        CTTATTGTGCGCAAAG        1:16    2:0:1   ␣
↪TTCCACCTCTCTTTGNAAAAAANAANAAAAANAANAAANAAANAAAANAAAAANNAAAAAANAANNNAAAAN        anti-
↪human_hashtag_3_TTCCGCCTCTCTTTG    0:15       1:0:0
CGCCCTTCTGGGTAGT        CGCCCTTCTGGGTAGT        0:16    0:0:0   ␣
```

(continues on next page)

(continued from previous page)

```
↪AAGTATCGTTTCGCATAAAAAAAAANAAAAANAANAAANAAAAAAAAAAAAAAAAAAAAAAAAANAAANNAAAAN        anti-
↪human_hashtag_5_AAGTATCGTTTCGCA    0:15       0:0:0
TACCTCGACCTGGAAG        TACCTCGACCTGGAAG        0:16    0:0:0  ␣
↪ATCAACTCTTTAGCGCAAAAAAAAAAAAAAAAAAAAAAAANAAAAAAAAAAAAAAAAAAAAAAAAANAAAANAAAAN        anti-
↪human_hashtag_1_GTCAACTCTTTAGCG    0:15       1:0:0
ACCACCCCCAACACCC        TACCACCACCCTAACA        0:13    0:0:3  ␣
↪ATCAACTCTTTAACATAAAAAAAAAAAAAAAAAAAAAAAANAAAAAAAAAAAAAAAAAAAAAAAAANAAAAAAAAAA        anti-
↪human_hashtag_1_GTCAACTCTTTAGCG    0:15       3:0:0
GCTAACTGATTCGGGC        AACTCGATGTCGGGCT        3:16    0:0:3  ␣
↪TAACAACCAGCCATAGAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAACAAAAAAAAAAANAAAAAAAAAA        anti-
↪human_hashtag_12_TAACGACCAGCCATA   0:15       1:0:0
TTTGCGGCTCTCCTAC        TTTGCGGCTCATGCAT        0:14    1:0:2  ␣
↪TTCCACCTCTCTTTGCAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAANAAAAAAAAAA        anti-
↪human_hashtag_3_TTCCGCCTCTCTTTG    0:15       1:0:0
```

Barcode extraction

The lengths of cell and feature barcodes are all identical (16 and 15, respectively). And based on the `qc` results, the distributions of starting and ending positions of cell and feature barcodes are very uniform. Search ranges are set to `0,16` on read 2 and `0,15` on read 3. One mismatch for cell and feature barcodes (`-cb_m`, `-cf_m`) are allowed. And by default, three ambiguous nucleotides (Ns) for read 1 and read 2 (`-cb_n`, `-cf_n`) are allowed. Use `-cb_rc` to reverse-complement cell barcode sequences.

```
$ fba extract \
    -1 hto_read_2.fq.gz \
    -2 hto_read_3.fq.gz \
    -w cell_barcodes.txt \
    -f feature_barcodes_HTO.tsv \
    -o feature_barcoding_output_HTO.tsv.gz \
    -r1_c 0,16 \
    -r2_c 0,15 \
    -cb_m 1 \
    -fb_m 1 \
    -cb_n 3 \
    -fb_n 3 \
    -cb_rc
```

Preview of result.

```
$ gzip -dc feature_barcoding_output_HTO.tsv.gz | head

read1_seq       cell_barcode    cb_num_mismatches       read2_seq       feature_barcode␣
↪fb_num_mismatches
AATAACCGACAGGTGA        AATCACCGACAGGTGA        1       ␣
↪ATCAACTCTTTAGCGtaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaanaaaaaaaaaa        anti-
↪human_hashtag_1_GTCAACTCTTTAGCG    1
TGCAGTATGCCTCGTA        TGCAGTATGCCTCGTT        1       ␣
```

(continues on next page)

(continued from previous page)

```
↪TAATGGCCTATTGGGgaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaacacccaaaaaaaaaaaaaaa        anti-
↪human_hashtag_2_TGATGGCCTATTGGG    1
TCGCGGTGAGCTTACA        TCGCGGTGAGCTTACA        0       ␣
↪TTCCGCCTCTCTTTGcaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa        anti-
↪human_hashtag_3_TTCCGCCTCTCTTTG    0
AACTAGCACTATTGCG        AACTAGCACTATTGCG        0       ␣
↪AAGTATCGTTTCGCAcaaaaaaaaaaaaaaaaaaaaaaaaaataacacttaaaaataaaaaaaaaaaacccaa        anti-
↪human_hashtag_5_AAGTATCGTTTCGCA    0
TGCAATGTGGGGTTCC        TGCAATGTGGGGTTCC        0       ␣
↪TTCCGCCTCTCTTTGaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa        anti-
↪human_hashtag_3_TTCCGCCTCTCTTTG    0
TGGATAGCTATCTGTG        TGGATAGCTATCTGTG        0       ␣
↪AAGTATCGTTTCGCAcaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaacccaaaaaaaaaaaaaaa        anti-
↪human_hashtag_5_AAGTATCGTTTCGCA    0
AGCAGAGACATCCTAG        AGCAGAGACATCCTAG        0       ␣
↪TTCCGCCTCTCTTTGaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa        anti-
↪human_hashtag_3_TTCCGCCTCTCTTTG    0
CTTAATCTGTGTTGTG        CTTAATCTGTGTTGTG        0       ␣
↪AAGTATCGTTTCGCAcaaaaaaaaaaaaaggtgtattactgtctcttatacacatctgacgctgccgacgact        anti-
↪human_hashtag_5_AAGTATCGTTTCGCA    0
GTTTCATTGTGGCATT        GTTTCATTGTGGCATT        0       ␣
↪AATAAGTTCAGCGTAgaaaaaaaaaaaaaaaaaaaaaaaaaaaatttaaatttgaattaaaataaaaaaaaata        anti-
↪human_hashtag_4_AGTAAGTTCAGCGTA    1
```

Result summary.

72.4% (22,820,698 out of 31,512,084) of total read pairs have valid cell and feature barcodes.

```
2022-03-16 00:14:08,601 - fba.__main__ - INFO - fba version: 0.0.x
2022-03-16 00:14:08,601 - fba.__main__ - INFO - Initiating logging ...
2022-03-16 00:14:08,601 - fba.__main__ - INFO - Python version: 3.10
2022-03-16 00:14:08,601 - fba.__main__ - INFO - Using extract subcommand ...
2022-03-16 00:14:08,635 - fba.levenshtein - INFO - Number of reference cell barcodes: 9,
↪151
2022-03-16 00:14:08,635 - fba.levenshtein - INFO - Number of reference feature barcodes:␣
↪7
2022-03-16 00:14:08,635 - fba.levenshtein - INFO - Read 1 coordinates to search: [0, 16)
2022-03-16 00:14:08,635 - fba.levenshtein - INFO - Read 2 coordinates to search: [0, 15)
2022-03-16 00:14:08,635 - fba.levenshtein - INFO - Cell barcode maximum number of␣
↪mismatches: 1
2022-03-16 00:14:08,635 - fba.levenshtein - INFO - Feature barcode maximum number of␣
↪mismatches: 1
2022-03-16 00:14:08,635 - fba.levenshtein - INFO - Read 1 maximum number of N allowed: 3
2022-03-16 00:14:08,635 - fba.levenshtein - INFO - Read 2 maximum number of N allowed: 3
2022-03-16 00:14:08,871 - fba.levenshtein - INFO - Matching ...
2022-03-16 00:19:01,333 - fba.levenshtein - INFO - Read pairs processed: 10,000,000
2022-03-16 00:23:44,891 - fba.levenshtein - INFO - Read pairs processed: 20,000,000
2022-03-16 00:28:29,304 - fba.levenshtein - INFO - Read pairs processed: 30,000,000
2022-03-16 00:29:12,889 - fba.levenshtein - INFO - Number of read pairs processed: 31,
↪512,084
2022-03-16 00:29:12,890 - fba.levenshtein - INFO - Number of read pairs w/ valid␣
↪barcodes: 22,820,698
2022-03-16 00:29:12,902 - fba.__main__ - INFO - Done.
```

Matrix generation

Only fragments with correctly matched cell and feature barcodes are included. Use `-ul` to set the UMI length (default 12). Setting to `0` means no UMIs and read counts are summarized instead. Use `-cb_rc` to reverse-complement cell barcode sequences in the output matrix if needed.

The generated feature count matrix can be easily imported into well-established single cell analysis packages such as Seurat and Scanpy.

```
$ fba count \
    -i feature_barcoding_output_HTO.tsv.gz \
    -o matrix_featurecount_HTO.csv.gz \
    -ul 0
```

Result summary.

The median number of reads per cell of this HTO library is 1,893.0.

```
2022-03-16 00:29:13,026 - fba.__main__ - INFO - fba version: 0.0.x
2022-03-16 00:29:13,026 - fba.__main__ - INFO - Initiating logging ...
2022-03-16 00:29:13,026 - fba.__main__ - INFO - Python version: 3.10
2022-03-16 00:29:13,026 - fba.__main__ - INFO - Using count subcommand ...
2022-03-16 00:29:14,943 - fba.count - INFO - UMI-tools version: 1.1.2
2022-03-16 00:29:14,950 - fba.count - INFO - UMI length set to 0, ignoring UMI␣
↪information. Skipping arguments: "-us/--umi_start".
2022-03-16 00:29:14,950 - fba.count - INFO - Header: read1_seq cell_barcode cb_num_
↪mismatches read2_seq feature_barcode fb_num_mismatches
2022-03-16 00:29:52,721 - fba.count - INFO - Number of read pairs processed: 22,820,698
2022-03-16 00:29:52,730 - fba.count - INFO - Number of cell barcodes detected: 9,151
2022-03-16 00:29:52,730 - fba.count - INFO - Number of features detected: 7
2022-03-16 00:29:52,730 - fba.count - INFO - Counting ...
2022-03-16 00:29:52,940 - fba.count - INFO - Total reads: 22,820,698
2022-03-16 00:29:52,941 - fba.count - INFO - Median number of reads per cell: 1,893.0
2022-03-16 00:29:53,099 - fba.__main__ - INFO - Done.
```

```
In [1]: import pandas as pd

In [2]: m = pd.read_csv("matrix_featurecount.csv.gz", index_col=0)

In [3]: m.sum(axis=1)
Out[3]:
anti-human_hashtag_12_TAACGACCAGCCATA    4402031
anti-human_hashtag_13_AAATCTCTCAGGCTC    2225016
anti-human_hashtag_1_GTCAACTCTTTAGCG     4107376
anti-human_hashtag_2_TGATGGCCTATTGGG     2672503
anti-human_hashtag_3_TTCCGCCTCTCTTTG     2469687
anti-human_hashtag_4_AGTAAGTTCAGCGTA     3172034
anti-human_hashtag_5_AAGTATCGTTTCGCA     3772051
dtype: int64

In [4]: m1 = m.loc[
```

(continues on next page)

(continued from previous page)

```
...:     [
...:         "anti-human_hashtag_1_GTCAACTCTTTAGCG",
...:         "anti-human_hashtag_2_TGATGGCCTATTGGG",
...:         "anti-human_hashtag_3_TTCCGCCTCTCTTTG",
...:         "anti-human_hashtag_4_AGTAAGTTCAGCGTA",
...:         "anti-human_hashtag_5_AAGTATCGTTTCGCA",
...:     ],
...:     :,
...: ]

In [5]: m1.to_csv(path_or_buf="matrix_featurecount_HTO_1-5.csv.gz",
                  compression="infer")

In [6]: m2 = m.loc[[
...:     "anti-human_hashtag_12_TAACGACCAGCCATA",
...:     "anti-human_hashtag_13_AAATCTCTCAGGCTC"
...: ], :]

In [7]: m2.to_csv(path_or_buf="matrix_featurecount_HTO_12-13.csv.gz",
                  compression="infer")
```

Demultiplexing

Gaussian mixture model

The implementation of demultiplexing method `2` (set by `-dm`) is inspired by the method described on the 10x Genomics' website. To set the probability threshold for demultiplexing, use `-p` (default `0.9`). To generate visualization plots, set `-v`.

```
$ fba demultiplex \
    -i matrix_featurecount_HTO_1-5.csv.gz \
    -dm 2 \
    -v
```

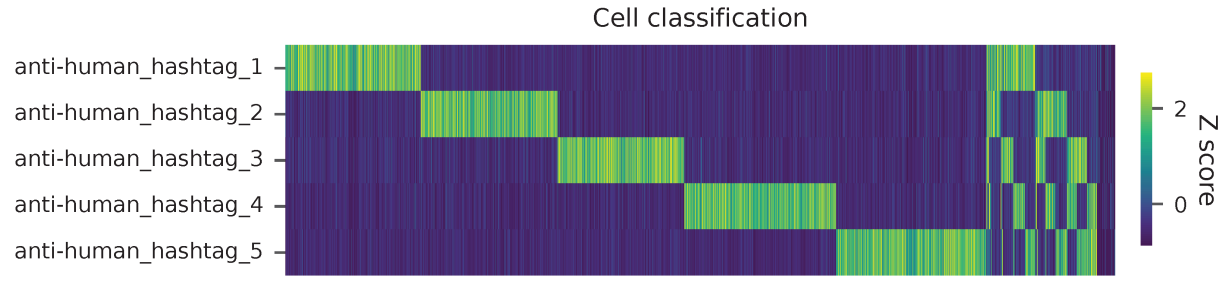
```
2022-03-16 00:38:18,749 - fba.__main__ - INFO - fba version: 0.0.x
2022-03-16 00:38:18,749 - fba.__main__ - INFO - Initiating logging ...
2022-03-16 00:38:18,749 - fba.__main__ - INFO - Python version: 3.9
2022-03-16 00:38:18,749 - fba.__main__ - INFO - Using demultiplex subcommand ...
2022-03-16 00:38:21,709 - fba.__main__ - INFO - Skipping arguments: "-q/--quantile", "-
↪cm/--clustering_method"
2022-03-16 00:38:21,709 - fba.demultiplex - INFO - Output directory: demultiplexed
2022-03-16 00:38:21,709 - fba.demultiplex - INFO - Demultiplexing method: 2
2022-03-16 00:38:21,709 - fba.demultiplex - INFO - UMI normalization method: clr
2022-03-16 00:38:21,709 - fba.demultiplex - INFO - Visualization: On
2022-03-16 00:38:21,709 - fba.demultiplex - INFO - Visualization method: tsne
2022-03-16 00:38:21,709 - fba.demultiplex - INFO - Loading feature count matrix: matrix_
```

(continues on next page)

(continued from previous page)

```
↪featurecount_HTO_1-5.csv.gz ...
2022-03-16 00:38:21,796 - fba.demultiplex - INFO - Number of cells: 9,151
2022-03-16 00:38:21,796 - fba.demultiplex - INFO - Number of positive cells for a␣
↪feature to be included: 200
2022-03-16 00:38:21,810 - fba.demultiplex - INFO - Number of features: 5 / 5 (after␣
↪filtering / original in the matrix)
2022-03-16 00:38:21,810 - fba.demultiplex - INFO - Features: anti-human_hashtag_1 anti-
↪human_hashtag_2 anti-human_hashtag_3 anti-human_hashtag_4 anti-human_hashtag_5
2022-03-16 00:38:21,810 - fba.demultiplex - INFO - Total UMIs/reads: 16,193,651 / 16,193,
↪651
2022-03-16 00:38:21,817 - fba.demultiplex - INFO - Median number of UMIs/reads per cell:␣
↪1,326.0 / 1,326.0
2022-03-16 00:38:21,817 - fba.demultiplex - INFO - Demultiplexing ...
2022-03-16 00:38:24,130 - fba.demultiplex - INFO - Generating heatmap ...
2022-03-16 00:38:26,376 - fba.demultiplex - INFO - Embedding ...
2022-03-16 00:38:43,503 - fba.__main__ - INFO - Done.
```

Heatmap of the relative abundance of features (hashtag oligos, HTOs) across all cells. Each column represents a single cell. This is a re-creation of Extended Data Fig. 6b in Mimitou, E.P., et al. (2021).



Preview the demultiplexing result: the numbers of singlets, multiplets and negatives are 7,728 (84.4%), 1,224 (13.4%), and 199 (2.2%), respectively.

```
In [1]: import pandas as pd

In [2]: m = pd.read_csv("demultiplexed/matrix_cell_identity.csv.gz", index_col=0)

In [3]: m.loc[:, m.sum(axis=0) == 1].sum(axis=1)
Out[3]:
anti-human_hashtag_1    1493
anti-human_hashtag_2    1511
anti-human_hashtag_3    1395
anti-human_hashtag_4    1675
anti-human_hashtag_5    1654
dtype: int64

In [4]: sum(m.sum(axis=0) == 1)
Out[4]: 7728

In [5]: sum(m.sum(axis=0) > 1)
Out[5]: 1224
```

(continues on next page)

(continued from previous page)

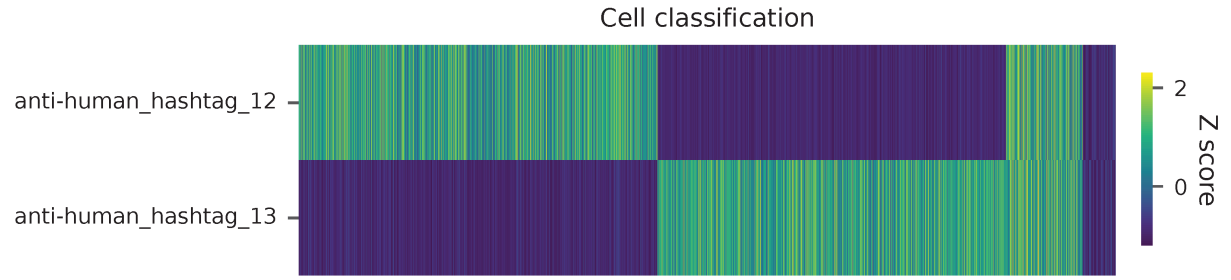
```
In [6]: sum(m.sum(axis=0) == 0)
Out[6]: 199

In [7]: m.shape
Out[7]: (5, 9151)
```

t-SNE embedding of cells based on the abundance of features (phage-derived tags, no transcriptome information used). Colors indicate the hashtag status for each cell, as called by FBA.

```
$ fba demultiplex \
    -i matrix_featurecount_HTO_12-13.csv.gz \
    -dm 2 \
    -v
```

```
2022-03-16 00:39:44,380 - fba.__main__ - INFO - Initiating logging ...
2022-03-16 00:39:44,380 - fba.__main__ - INFO - Python version: 3.9
2022-03-16 00:39:44,380 - fba.__main__ - INFO - Using demultiplex subcommand ...
2022-03-16 00:39:47,238 - fba.__main__ - INFO - Skipping arguments: "-q/--quantile", "-
↪cm/--clustering_method"
2022-03-16 00:39:47,238 - fba.demultiplex - INFO - Output directory: demultiplexed
2022-03-16 00:39:47,238 - fba.demultiplex - INFO - Demultiplexing method: 2
2022-03-16 00:39:47,238 - fba.demultiplex - INFO - UMI normalization method: clr
2022-03-16 00:39:47,238 - fba.demultiplex - INFO - Visualization: On
2022-03-16 00:39:47,238 - fba.demultiplex - INFO - Visualization method: tsne
2022-03-16 00:39:47,238 - fba.demultiplex - INFO - Loading feature count matrix: matrix_
↪featurecount_HTO_12-13.csv.gz ...
2022-03-16 00:39:47,329 - fba.demultiplex - INFO - Number of cells: 9,151
2022-03-16 00:39:47,341 - fba.demultiplex - INFO - Number of positive cells for a␣
↪feature to be included: 200
2022-03-16 00:39:47,355 - fba.demultiplex - INFO - Number of features: 2 / 2 (after␣
↪filtering / original in the matrix)
2022-03-16 00:39:47,363 - fba.demultiplex - INFO - Features: anti-human_hashtag_12 anti-
↪human_hashtag_13
2022-03-16 00:39:47,363 - fba.demultiplex - INFO - Total UMIs/reads: 6,627,047 / 6,627,
↪047
2022-03-16 00:39:47,370 - fba.demultiplex - INFO - Median number of UMIs/reads per cell:␣
↪559.0 / 559.0
2022-03-16 00:39:47,370 - fba.demultiplex - INFO - Demultiplexing ...
2022-03-16 00:39:48,484 - fba.demultiplex - INFO - Generating heatmap ...
2022-03-16 00:39:49,412 - fba.demultiplex - INFO - Embedding ...
2022-03-16 00:40:06,551 - fba.__main__ - INFO - Done.
```

Heatmap of the relative abundance of features (hashtag oligos, HTOs) across all cells. Each column represents a single cell. This is a re-creation of Extended Data Fig. 6b in Mimitou, E.P., et al. (2021).



Preview the demultiplexing result: the numbers of singlets, multiplets and negatives are 7,924 (86.6%), 856 (9.4%), and 371 (4.1%), respectively.

```
In [1]: import pandas as pd

In [2]: m = pd.read_csv("demultiplexed/matrix_cell_identity.csv.gz", index_col=0)

In [3]: m.loc[:, m.sum(axis=0) == 1].sum(axis=1)
Out[3]:
anti-human_hashtag_12    4018
anti-human_hashtag_13    3906
dtype: int64

In [4]: [sum(m.sum(axis=0) == i) for i in (1, 2, 0)]
Out[4]: [7924, 856, 371]

In [5]: m.shape
Out[5]: (2, 9151)
```

t-SNE embedding of cells based on the abundance of features (phage-derived tags, no transcriptome information used). Colors indicate the hashtag status for each cell, as called by FBA.

CITE-seq

QC

Same as the HTO library, sample the first 10,000 (set by `-n`) read pairs for quality control.

```
$ fba qc \
    -1 adt_read_2.fq.gz \
    -2 adt_read_3.fq.gz \
    -w cell_barcodes.txt \
    -f feature_barcodes_HTO.tsv \
    --output_directory qc \
    -cb_rc
```

Cell barcodes are A-rich.

As for read 3, based on the per base content, it suggests that bases 0-14 are actually our feature barcodes (See the distribution of matched barcode positions on read 3).

The detailed qc results are stored in `feature_barcoding_output.tsv.gz` file. `matching_pos` columns indicate the matched positions on reads. `matching_description` columns indicate mismatches in substitutions:insertions:deletions format.

```
$ gzip -dc feature_barcoding_output.tsv.gz | grep -v no | head

read1_seq       cell_barcode    cb_matching_pos cb_matching_description read2_seq      ␣
↪feature_barcode fb_matching_pos fb_matching_description
CCTAAGAAAAAGCTGC        CCTAAGAACAAGACAT        0:14    1:0:2  ␣
↪AATCCTTCCGAATNNNNNNANNNNNNANNNANNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN        BJ18_
↪AATCCTTCCGAATGT    0:15    2:0:0
TTATCCAAGGACTGTT        TCCCACATGGACTGTT        3:16    0:0:3  ␣
↪TATTCCCGCTCAANNNNNNANNNNNNANNNANNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN        RPA-T4_
↪TGTTCCCGCTCAACT  0:15    3:0:0
GAGAAGAGACCGATTA        AGGAAGAGACCCGAGT        1:15    1:0:2  ␣
↪CTCATTGTAACTCCNNNANAANNANNANNNANNNNANNNNNANNNNNNNANNNNNNNNNNNNNNNANNNN        UCHT1_
↪CTCATTGTAACTCCT   0:15    1:0:0
ATACATCATAAACAAA        ATACATCCTTAACGAA        0:16    3:0:0  ␣
↪ATCCTTCCGAATGTNNNAAAANNANNANNAANNNNANANNNANNNANNNANNNNNNNANNNNNANNNNNANNAN        BJ18_
↪AATCCTTCCGAATGT    0:14    0:0:1
GATCATGTGCTTTGTT        TGAACTGTGCTTTGTT        0:16    1:1:1  ␣
↪CACACACCCATTAAANAAAAANNANNANNAANNNNANANNNANNNANNNAANNNNNAAAANNANNNNNAANAN        C398.
↪4A_CGCGCACCCATTAAA 0:15    2:0:0
AACACTCGACAGGTGA        AACACTCGAGAAGGCT        0:14    1:0:2  ␣
↪ATCCCTGCAACTTGANAAAAANNANNAANAANNANANANNNANNNANNNAANANNNAAAANNANNNNNAANAN        DREG-
↪56_GTCCCTGCAACTTGA 0:15    1:0:0
ACTGTTCGACTACTCC        AGTGTTCGACTATCTT        0:15    2:0:1  ␣
↪TCCCTTGCGATTTACNAAAAAANAANAANAANAANAAANAAANCAAANNAANANNAAAAAANNANNNNNAANAN        2D1_
↪TCCCTTGCGATTTAC     0:15    0:0:0
ACTTATCTGTGCATCA        ACTTATCTGTGCATCA        0:16    0:0:0  ␣
↪ACCTTTATGCCACGGNAAAAAANAANAAAAAANAANAAANAAANAAAANTTTAANNAAAAAANAANNNAAAAN        WM59_
↪ACCTTTATGCCACGG    0:15    0:0:0
GAGCATAATTTAGCAG        GAGCATACTTGAGCAG        0:16    2:0:0  ␣
↪CAATAAGTTCGGGACCAAAAAAAAANAAAAANAANAAANAAANAAAAAAAAAAANAAAAAAANAAANNAAAAN        4B4-1_
↪CAGTAAGTTCGGGAC   0:15    1:0:0
```

Barcode extraction

The lengths of cell and feature barcodes are all identical (16 and 15, respectively). And based on the `qc` results, the distributions of starting and ending positions of cell and feature barcodes are very uniform. Search ranges are set to `0,16` on read 2 and `0,15` on read 3. One mismatch for cell and feature barcodes (`-cb_m`, `-cf_m`) are allowed. And by default, three ambiguous nucleotides (Ns) for read 1 and read 2 (`-cb_n`, `-cf_n`) are allowed. Use `-cb_rc` to reverse-complement cell barcode sequences for processing.

```
$ fba extract \
    -1 adt_read_2.fq.gz \
    -2 adt_read_3.fq.gz \
    -w cell_barcodes.txt \
    -f feature_barcodes_ADT.tsv \
    -o feature_barcoding_output_ADT.tsv.gz \
    -r1_c 0,16 \
    -r2_c 0,15 \
    -cb_m 1 \
    -fb_m 1 \
    -cb_n 3 \
    -fb_n 3 \
    -cb_rc
```

Preview of result.

```
$ gzip -dc feature_barcoding_output_ADT.tsv.gz | head

read1_seq       cell_barcode    cb_num_mismatches       read2_seq       feature_barcode␣
↪fb_num_mismatches
CAAGAAATGCATTCAG        CAAGAAATGCATTCAG        0       ␣
↪TATACCCGCTTGTGAtaaaaaaaaaaaaaaaaaaaaaaaanaaaaaaaaaaaaaaaaaaaaaaaanaaaaaaaaaa        HIT2_
↪TGTACCCGCTTGTGA    1
GTGGCTGTGTTTGTCT        GTGGCTGTGTTTGTCT        0       ␣
↪TAGATTCCCGGACTTgaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaanaaaaaaaaaa        CD7-
↪6B7_TGGATTCCCGGACTT 1
ACTGATTCTGCCCTAG        ACTGATTCTGCCCTAG        0       ␣
↪TAGATTCCCGGACTTtaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaanaaaaaaaaaa        CD7-
↪6B7_TGGATTCCCGGACTT 1
CCTGCTATGCTGCGGT        CCTGCTATGCTGCGGT        0       ␣
↪TAGATTCCCGGACTTgaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaanaaaaaaaaaa        CD7-
↪6B7_TGGATTCCCGGACTT 1
TGTAGTTTGGATAGCG        GGTAGTTTGGATAGCG        1       ␣
↪TTTATCCTGTACGCCtaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaanaaaaaaaaaa        BC96_
↪TTTGTCCTGTACGCC    1
TCCTTAAACCATCCTC        TCCTTAAACCATCCTC        0       ␣
↪AATCCTTCCGAATGTtaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaanaaaaaaaaaa        BJ18_
↪AATCCTTCCGAATGT    0
TTCCCATCTGGAATAT        TTCCCATCTGGAATAT        0       ␣
↪CTCATTGTAACTCCTcaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa        UCHT1_
↪CTCATTGTAACTCCT   0
ACTCAGAACGAATTGG        ACTCAGAACGAATTGA        1       ␣
↪TATTCCCGCTCAACTcaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa        RPA-T4_
↪TGTTCCCGCTCAACT  1
CAGGACCTGCGCACTG        CAGGACCTGCGCACTG        0       ␣
```

(continues on next page)

(continued from previous page)

```
↪TATTCCCGCTCAACTcaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa        RPA-T4_
↪TGTTCCCGCTCAACT  1
```

Result summary.

51.3% (27,719,537 out of 54,024,324) of total read pairs have valid cell and feature barcodes.

```
2022-03-15 23:43:13,501 - fba.__main__ - INFO - fba version: 0.0.x
2022-03-15 23:43:13,501 - fba.__main__ - INFO - Initiating logging ...
2022-03-15 23:43:13,501 - fba.__main__ - INFO - Python version: 3.10
2022-03-15 23:43:13,501 - fba.__main__ - INFO - Using extract subcommand ...
2022-03-15 23:43:13,562 - fba.levenshtein - INFO - Number of reference cell barcodes: 9,
↪151
2022-03-15 23:43:13,562 - fba.levenshtein - INFO - Number of reference feature barcodes:␣
↪37
2022-03-15 23:43:13,562 - fba.levenshtein - INFO - Read 1 coordinates to search: [0, 16)
2022-03-15 23:43:13,562 - fba.levenshtein - INFO - Read 2 coordinates to search: [0, 15)
2022-03-15 23:43:13,562 - fba.levenshtein - INFO - Cell barcode maximum number of␣
↪mismatches: 1
2022-03-15 23:43:13,562 - fba.levenshtein - INFO - Feature barcode maximum number of␣
↪mismatches: 1
2022-03-15 23:43:13,562 - fba.levenshtein - INFO - Read 1 maximum number of N allowed: 3
2022-03-15 23:43:13,562 - fba.levenshtein - INFO - Read 2 maximum number of N allowed: 3
2022-03-15 23:43:13,798 - fba.levenshtein - INFO - Matching ...
2022-03-15 23:47:34,902 - fba.levenshtein - INFO - Read pairs processed: 10,000,000
2022-03-15 23:51:53,692 - fba.levenshtein - INFO - Read pairs processed: 20,000,000
2022-03-15 23:56:13,885 - fba.levenshtein - INFO - Read pairs processed: 30,000,000
2022-03-16 00:00:31,902 - fba.levenshtein - INFO - Read pairs processed: 40,000,000
2022-03-16 00:04:52,531 - fba.levenshtein - INFO - Read pairs processed: 50,000,000
2022-03-16 00:06:37,721 - fba.levenshtein - INFO - Number of read pairs processed: 54,
↪024,324
2022-03-16 00:06:37,722 - fba.levenshtein - INFO - Number of read pairs w/ valid␣
↪barcodes: 27,719,537
2022-03-16 00:06:37,734 - fba.__main__ - INFO - Done.
```

Matrix generation

Only fragments with correctly matched cell and feature barcodes are included. Use `-ul` to set the UMI length (default 12). Setting to `0` means no UMIs and read counts are summarized instead. Use `-cb_rc` to reverse-complement cell barcode sequences in the output matrix if needed.

```
$ fba count \
    -i feature_barcoding_output_ADT.tsv.gz \
    -o matrix_featurecount_ADT.csv.gz \
    -ul 0
```

Result summary.

The median number of reads per cell of this ADT library is 2,645.0.

```
2022-03-16 00:14:03,746 - fba.__main__ - INFO - fba version: 0.0.x
2022-03-16 00:14:03,746 - fba.__main__ - INFO - Initiating logging ...
2022-03-16 00:14:03,746 - fba.__main__ - INFO - Python version: 3.10
2022-03-16 00:14:03,746 - fba.__main__ - INFO - Using count subcommand ...
2022-03-16 00:14:05,873 - fba.count - INFO - UMI-tools version: 1.1.2
2022-03-16 00:14:05,881 - fba.count - INFO - UMI length set to 0, ignoring UMI␣
↪information. Skipping arguments: "-us/--umi_start".
2022-03-16 00:14:05,881 - fba.count - INFO - Header: read1_seq cell_barcode cb_num_
↪mismatches read2_seq feature_barcode fb_num_mismatches
2022-03-16 00:14:50,518 - fba.count - INFO - Number of read pairs processed: 27,719,537
2022-03-16 00:14:50,549 - fba.count - INFO - Number of cell barcodes detected: 9,151
2022-03-16 00:14:50,549 - fba.count - INFO - Number of features detected: 37
2022-03-16 00:14:50,549 - fba.count - INFO - Counting ...
2022-03-16 00:14:50,901 - fba.count - INFO - Total reads: 27,719,537
2022-03-16 00:14:50,903 - fba.count - INFO - Median number of reads per cell: 2,645.0
2022-03-16 00:14:51,456 - fba.__main__ - INFO - Done.
```

On average, approximately 25 ADTs (proteins) are detected per cell.

```
In [1]: import pandas as pd

In [2]: m = pd.read_csv("matrix_featurecount.csv.gz", index_col=0)

In [3]: m.shape
Out[3]: (37, 9151)

In [4]: print(m.sum(axis=1).sort_values(ascending=False).to_string())

RPA-T4_TGTTCCCGCTCAACT                4373383
BC96_TTTGTCCTGTACGCC                  3770330
CD7-6B7_TGGATTCCCGGACTT               3656097
FN50_GTCTCTTGGCTTAAA                  2477291
BJ18_AATCCTTCCGAATGT                  2448859
2D1_TCCCTTGCGATTTAC                   2192495
C398.4A_CGCGCACCCATTAAA               2186772
HIT2_TGTACCCGCTTGTGA                  1864653
UCHT1_CTCATTGTAACTCCT                 1243612
Ber-ACT35_(ACT35)_AACCCACCGTTGTTA      666300
DX2_CCAGCTCATTAGAGC                    606868
EH12.2H7_ACAGCGCCGTATTTA               510931
O323_GCACTCCTGCATGTA                   396037
108-17_ACCTTTCGACACTCG                 299575
F38-2E2_TGTCCTACCCAACTT                230863
WM59_ACCTTTATGCCACGG                   123346
A1_TTACCTGGTATCCGT                     117428
UCHL1_CTCCGAATCATGTTG                  111140
4B4-1_CAGTAAGTTCGGGAC                   90803
DREG-56_GTCCCTGCAACTTGA                 63105
G043H7_AGTTCAGTCAACCGA                  55890
S-HCL-3_TACGCCTATAACTTG                 50507
IP26_CGTAACGTAGAGCGA                    49413
CD28.2_TGAGAACGACCCTAA                  27709
HI100_TCAATCCTTCCGCTT                   26717
```

(continues on next page)

(continued from previous page)

```
SK1_GCGCAACTTGATGAT                    12505
A019D5_GTGTGTTGTCCTATG                 10286
HIB19_CTGGGCAATTACTCG                   8570
5.1H11_TCCTTTCCTGATAGG                  8190
A15153G_TTGCTTACCGCCAGA                 7384
HP-3G10_GTACGCAGTCCTTCT                 7150
3G8_AAGTTCACTCTTTGC                     6133
M5E2_TCTCAGACCTCCGTA                    6059
AD2_CAGTTCCTCAGTTCG                     5723
J252D4_AATTCAACCGTCGCC                  4252
MOPC-21_GCCGGACGACATTAA                 1807
QA17A04_AACTCCCTATGGAGG                 1354

In [5]: np.median(m.sum(axis=0))
Out[5]: 2645.0

In [6]: np.median((m > 0).sum(axis=0))
Out[6]: 25.0
```

3.2.3 1k Human PBMCs Stained with a Panel of TotalSeq B Antibodies

Dataset: 1k Human PBMCs Stained with a Panel of TotalSeq B Antibodies, Dual Indexed

The detailed description of this dataset can be found here.

Preparation

Download fastq files.

```
$ wget https://cf.10xgenomics.com/samples/cell-exp/4.0.0/SC3_v3_NextGem_DI_PBMC_CSP_1K/
↪SC3_v3_NextGem_DI_PBMC_CSP_1K_fastqs.tar

$ tar xvf SC3_v3_NextGem_DI_PBMC_CSP_1K/SC3_v3_NextGem_DI_PBMC_CSP_1K_fastqs.tar
```

Combine reads of different lanes.

```
$ cat SC3_v3_NextGem_DI_CSP-Labeled_PBMCs_1K_fastqs/SC3_v3_NextGem_DI_CSP-Labeled_PBMCs_
↪1K_antibody_fastqs/SC3_v3_NextGem_DI_CSP-Labeled_PBMCs_1K_antibody_S1_L00?_R1_001.
↪fastq.gz > SC3_v3_NextGem_DI_CSP-Labeled_PBMCs_1K_antibody_S1_combined_R1.fastq.gz

$ cat SC3_v3_NextGem_DI_CSP-Labeled_PBMCs_1K_fastqs/SC3_v3_NextGem_DI_CSP-Labeled_PBMCs_
↪1K_antibody_fastqs/SC3_v3_NextGem_DI_CSP-Labeled_PBMCs_1K_antibody_S1_L00?_R2_001.
↪fastq.gz > SC3_v3_NextGem_DI_CSP-Labeled_PBMCs_1K_antibody_S1_combined_R2.fastq.gz
```

Download cell barcode info. These are the cell-associated barcodes in this single cell RNA-Seq library.

```
$ wget https://cf.10xgenomics.com/samples/cell-exp/4.0.0/SC3_v3_NextGem_DI_PBMC_CSP_1K/
↪SC3_v3_NextGem_DI_PBMC_CSP_1K_filtered_feature_bc_matrix.tar.gz

$ tar zxvf SC3_v3_NextGem_DI_PBMC_CSP_1K_filtered_feature_bc_matrix.tar.gz
```

Inspect cell barcodes.

```
$ gzip -dc filtered_feature_bc_matrix/barcodes.tsv.gz | head

AAACCCAGTACCAGAG-1
AAACGCTTCGGTCTGG-1
AAACGCTTCGTTGCCT-1
AAAGAACAGGAACTCG-1
AAAGGATAGGTCGAGT-1
AAAGGGCCAAGACGGT-1
AAAGGGCCATCCTCAC-1
AAAGGTATCGTGGACC-1
AAAGTCCGTCGCGTTG-1
AAAGTCCTCCTTATCA-1
```

Prepare feature barcodes.

```
$ wget https://cf.10xgenomics.com/samples/cell-exp/4.0.0/SC3_v3_NextGem_DI_PBMC_CSP_1K/
↪SC3_v3_NextGem_DI_PBMC_CSP_1K_feature_ref.csv
```

Inspect feature barcode info.

```
$ head SC3_v3_NextGem_DI_PBMC_CSP_1K_feature_ref.csv

id,name,read,pattern,sequence,feature_type
CD3,CD3,R2,^NNNNNNNNNN(BC)NNNNNNNNN,CTCATTGTAACTCCT,Antibody Capture
CD4,CD4,R2,^NNNNNNNNNN(BC)NNNNNNNNN,TGTTCCCGCTCAACT,Antibody Capture
CD8a,CD8a,R2,^NNNNNNNNNN(BC)NNNNNNNNN,GCTGCGCTTTCCATT,Antibody Capture
CD11b,CD11b,R2,^NNNNNNNNNN(BC)NNNNNNNNN,GACAAGTGATCTGCA,Antibody Capture
CD14,CD14,R2,^NNNNNNNNNN(BC)NNNNNNNNN,TCTCAGACCTCCGTA,Antibody Capture
CD15,CD15,R2,^NNNNNNNNNN(BC)NNNNNNNNN,TCACCAGTACCTAGT,Antibody Capture
CD16,CD16,R2,^NNNNNNNNNN(BC)NNNNNNNNN,AAGTTCACTCTTTGC,Antibody Capture
CD19,CD19,R2,^NNNNNNNNNN(BC)NNNNNNNNN,CTGGGCAATTACTCG,Antibody Capture
CD20,CD20,R2,^NNNNNNNNNN(BC)NNNNNNNNN,TTCTGGGTCCCTAGA,Antibody Capture
```

Clean up.

```
$ wc -l SC3_v3_NextGem_DI_PBMC_CSP_1K_feature_ref.csv

33 SC3_v3_NextGem_DI_PBMC_CSP_1K_feature_ref.csv
```

```
$ cut -d',' -f1,5 SC3_v3_NextGem_DI_PBMC_CSP_1K_feature_ref.csv | tail -32 | sed 's/,/\t/
↪g' > feature_barcode_ref.tsv

$ head feature_barcode_ref.tsv

CD3     CTCATTGTAACTCCT
CD4     TGTTCCCGCTCAACT
```

(continues on next page)

(continued from previous page)

```
CD8a    GCTGCGCTTTCCATT
CD11b   GACAAGTGATCTGCA
CD14    TCTCAGACCTCCGTA
CD15    TCACCAGTACCTAGT
CD16    AAGTTCACTCTTTGC
CD19    CTGGGCAATTACTCG
CD20    TTCTGGGTCCCTAGA
CD25    TTTGTCCTGTACGCC
```

QC

The first 20,000 read pairs are sampled (set by `-n`, default `100,000`) for quality control. The `-t` option can be used to set the number of threads. By default, diagnostic results and plots are generated in the `qc` directory (set by `--output_directory`), and the full length of read 1 and read 2 are searched against reference cell and feature barcodes, respectively. The per base content of both read pairs and the distribution of matched barcode positions are summarized. Use `-r1_c` and/or `-r2_c` to limit the search range, and `-cb_n` and/or `-fb_n` to set the mismatch tolerance for cell and/or feature barcode matching (default `3`).

```
$ fba qc \
    -1 SC3_v3_NextGem_DI_CSP-Labeled_PBMCs_1K_antibody_S1_combined_R1_001.fastq.gz \
    -2 SC3_v3_NextGem_DI_CSP-Labeled_PBMCs_1K_antibody_S1_combined_R2_001.fastq.gz \
    -w filtered_feature_bc_matrix/barcodes.tsv.gz \
    -f SC3_v3_NextGem_DI_PBMC_CSP_1K_feature_ref.tsv \
    --output_directory qc \
    -n 20000
```

This library was constructed using the Chromium Next GEM Single Cell 3 Reagent Kits v3.1 (Dual Index) with Feature Barcode technology for Cell Surface Protein, and was sequenced on the Illumina NovaSeq 6000 platform. The first 16 bases of read 1 correspond to the cell barcodes, followed by 12 bases for UMIs. The base content plot reveals that the GC content of the cell barcodes is evenly distributed, while the UMIs show a slight enrichment for the nucleotide T.

Regarding read 2, the per base content analysis suggests that for the sampled reads, bases 0-9, 25-33, and 56-83 exhibit a balanced GC content, which indicates that their sequences are likely random at the library level. However, bases 34-55 and 84-89 appear to be constant sequences, and we can almost read the bases in these regions. The bases 10-24 are less random but also not constant, as they correspond to our feature barcodes, which is evident from the distribution of matched barcode positions on read 2.

The fragment structure inferred from the `qc` results of this feature barcoding library matches the design as specified on Page 3, Table 1 (Note that this is an example dataset from 10x Genomics). The sequence of bases 34-56 is referred to as "Capture Sequence 1" on the beads.

The detailed `qc` results are stored in the `feature_barcoding_output.tsv.gz` file. The `matching_pos` columns indicate the matched positions on reads, while the `matching_description` columns indicate mismatches in the format of substitutions:insertions:deletions.

```
$ gzip -dc qc/feature_barcoding_output.tsv.gz | head

read1_seq       cell_barcode    cb_matching_pos cb_matching_description read2_seq      ␣
↪feature_barcode fb_matching_pos fb_matching_description
GNAGGTTTCGTCGACACGGGTATGGCCA    GTAGGTTAGGTCGACA        0:16    3:0:0   ␣
↪GGAACGACGATCTCAGACCTCCGTAACGAACGTAGCTTTAAGGCCGGTCCTAGCAATGGCCATACCCGTGTCGACGAAACCTACCTGTCT␣
↪     CD14_TCTCAGACCTCCGTA    10:25   0:0:0
TCGTGGGCAAGATTGAGTAAAAATTCAG    TCGTGGGCAAACTAGA        0:16    3:0:0   ␣
↪CAATCTACCGCTGGGCAATTACTCGCCCGATGTGGCTTTAAGGCCGGTCCTAGCAACTGAATTTTTACTCAATCTTGCCCACGACTGTCT␣
↪     CD19_CTGGGCAATTACTCG    10:25   0:0:0
TCCGGGATCGTGGCTATGTGTTCGAGGG    no_match        NA      NA      ␣
↪TGGGTCAATTCTCATTGTAACTCCTTCCCTCAAAGCTTTAAGGCCGGTCCTAGCAACCCTCGAACACATAGCCACGATCCCGGACTGTCT␣
↪     NA      NA      NA
GATTGGTGTGTCTCTCGGTGCCCAAAAT    no_match        NA      NA      ␣
↪TACACTGACAGCCGGACGACATTAAACGGAAGCCGCTTTAAGGCCGGTCCTAGCAAATTTTGGGCACCGAGAGACACACCAATCCTGTCT␣
↪     NA      NA      NA
TGCTCGTAGTACAGGTAAGCGTGAAGCA    CATCGTCGTACAGGTG        2:17    2:0:1   ␣
↪GCCCACAACACTCCGAATCATGTTGACGGGGTTCGCTTTAAGGCCGGTCCTAGCAATGCTTCACGCTTACCTGTACTACGAGCACTGTCG␣
↪     CD45RO_CTCCGAATCATGTTG   10:25   0:0:0
AGATGAAAGGGAGTAGGGTGCGGGTTAT    AGGGAGTAGGAGGGTG        7:22    2:0:1   ␣
↪TGCGTTTCTTCACCAGTACCTAGTCACACGTCAGCTTTAAGGCCGGCCCTAGCAAATAACCCGCACCCTACTCCCTTTCATCTCTGTCTC␣
↪     CD15_TCACCAGTACCTAGT    9:24    0:0:0
CCTCCTCAGCTCTGCATATGAGCGAATT    no_match        NA      NA      ␣
↪AAGCAATGCTTCACCAGTACCTAGTCGTTCGTGTGCTTAAGGCCGGTCCTAGCAAAATTCGCTCATATGCAGAGCTGAGGAGGCTGTCTC␣
↪     NA      NA      NA
TGATCTTAGAACACGTCAGGGTCCTGAA    TGATCTTTCAACACGT        0:16    2:0:0   ␣
↪GGGGGGGGGGGGGGGGAGGGGGCCGGAAAAGAACCCCGAGAGGCCAGCGCCAAACAAAAAAGAACAAAAAAGAGGAAAAAAAAAAAAAAA␣
↪     no_match        NA      NA
GGGCTACAGGACGCTGGTTTCATTTTTT    CTGGTCTTCATTGTTC        13:28   2:0:1   ␣
↪CCTTAATCAACTCATTGTAACTCCTGTTCCCACAGCTTTAAGGCCGGTCCTAGCAAAAAAAATGAAACCAGCGTCCTGTAGCCCCTGTCT␣
↪     CD3_CTCATTGTAACTCCT      10:25   0:0:0
```

Barcode extraction

The lengths of cell and feature barcodes are all identical (16 and 15, respectively). And based on the `qc` results, the distributions of starting and ending positions of cell and feature barcodes are very uniform. Search ranges are set to `0,16` on read 1 and `10,25` on read 2. Two mismatches for cell and feature barcodes (`-cb_m`, `-cf_m`) are allowed. And by default, three ambiguous nucleotides (Ns) for read 1 and read2 (`-cb_n`, `-cf_n`) are allowed.

```
$ fba extract \
    -1 SC3_v3_NextGem_DI_CSP-Labeled_PBMCs_1K_antibody_S1_combined_R1_001.fastq.gz \
    -2 SC3_v3_NextGem_DI_CSP-Labeled_PBMCs_1K_antibody_S1_combined_R2_001.fastq.gz \
    -w filtered_feature_bc_matrix/barcodes.tsv.gz \
    -f SC3_v3_NextGem_DI_PBMC_CSP_1K_feature_ref.tsv \
    -o feature_barcoding_output.tsv.gz \
    -r1_c 0,16 \
    -r2_c 10,25 \
    -cb_m 2 \
```

(continues on next page)

(continued from previous page)

```
    -fb_m 2
```

Preview of result.

```
$ gzip -dc feature_barcoding_output.tsv.gz | head

read1_seq       cell_barcode    cb_num_mismatches       read2_seq       feature_barcode␣
↪fb_num_mismatches
TCGTGGGCAAGATTGAgtaaaaattcag    TCGTGGGGTAGATTGA        2       ␣
↪caatctaccgCTGGGCAATTACTCGcccgatgtggctttaaggccggtcctagcaactgaatttttactcaatcttgcccacgactgtct␣
↪    CD19_CTGGGCAATTACTCG     0
TGTCCACTCTAGGGTCcagaaatcgcag    TGTCCACAGTAGGGTC        2       ␣
↪gtggccgtgtTCAATCCTTCCGCTTcgtcgttctgctttaaggccggtcctagcaagctgcgatttctggaccctagagtggacactgtc␣
↪    CD45RA_TCAATCCTTCCGCTT   0
TTTCGATAGTGTCAGAtttacttatggt    TTTCGATTCTGTCAGA        2       ␣
↪caaagtcctaTCTCAGACCTCCGTAaagcatgtggctttaaggccggtcctagcaaaccataagtaaatctgacactatcgaaactgtct␣
↪    CD14_TCTCAGACCTCCGTA     0
ACCATTTCAGAGTAGCaaaaccgttggg    ACCATTTGTGAGTAGC        2       ␣
↪acacgtggcgTGTTCCCGCTCAACTgtgcttcaagctttaaggccggtcctagcaacccaacggttttgctactctgaaatggtctgtct␣
↪    CD4_TGTTCCCGCTCAACT      0
CATGCGGAGCACAGCGctagttacaaac    CATGCGGTCCACAGCG        2       ␣
↪ctgagcaggaTCTCAGACCTCCGTAatgggttaagctttaaggccggtcctagcaagtttgtaactagcgctgtgctccgcatgctgtct␣
↪    CD14_TCTCAGACCTCCGTA     0
TGCCGAGCAACGTAGGgcgtaattagcg    TGCCGAGGTACGTAGG        2       ␣
↪aggagacccgTCTCAGACCTCCGTAagtacccgagctttaaggccggtcctagcaacgctaattacgccctacgttgctcggcactgtct␣
↪    CD14_TCTCAGACCTCCGTA     0
ATTCCATGTCTCTCGTcgtctaactccc    ATTCCATCACTCTCGT        2       ␣
↪atgcagagtgCTCATTGTAACTCCTccgtttgacgctttaaggccggtcctagcaagggagttagacgacgagagacatggaatctgtct␣
↪    CD3_CTCATTGTAACTCCT      0
GGCAGTCGTAAGGTTAtgcaccacacga    GGCAGTCCAAAGGTTA        2       ␣
↪gcaacatggtTCTCAGACCTCCGTAgcatgttaggctttaaggccggtcctagcaatcgtgtggtgcataaccttacgactgccctgtct␣
↪    CD14_TCTCAGACCTCCGTA     0
GATGGAGGTGAGCTAGaaatgccaagtt    GATGGAGCAGAGCTAG        2       ␣
↪acacaatgaaTGTTCCCGCTCAACTtaccggggtgctttaaggccggtcctagcaaaacttggcatttctagctcacctccatcctgtct␣
↪    CD4_TGTTCCCGCTCAACT      0
```

Result summary.

59.8% (4,607,787 out of 7,704,799) of total read pairs have valid cell and feature barcodes.

```
2021-02-17 23:37:41,353 - fba.__main__ - INFO - fba version: 0.0.7
2021-02-17 23:37:41,353 - fba.__main__ - INFO - Initiating logging ...
2021-02-17 23:37:41,353 - fba.__main__ - INFO - Python version: 3.7
2021-02-17 23:37:41,353 - fba.__main__ - INFO - Using extract subcommand ...
2021-02-17 23:37:41,356 - fba.levenshtein - INFO - Number of reference cell barcodes: 1,
↪200
2021-02-17 23:37:41,356 - fba.levenshtein - INFO - Number of reference feature barcodes:␣
↪32
2021-02-17 23:37:41,356 - fba.levenshtein - INFO - Read 1 coordinates to search: [0, 16)
2021-02-17 23:37:41,356 - fba.levenshtein - INFO - Read 2 coordinates to search: [10, 25)
2021-02-17 23:37:41,356 - fba.levenshtein - INFO - Cell barcode maximum number of␣
↪mismatches: 2
2021-02-17 23:37:41,356 - fba.levenshtein - INFO - Feature barcode maximum number of␣
```

(continues on next page)

(continued from previous page)

```
↪mismatches: 2
2021-02-17 23:37:41,356 - fba.levenshtein - INFO - Read 1 maximum number of N allowed: 3
2021-02-17 23:37:41,356 - fba.levenshtein - INFO - Read 2 maximum number of N allowed: 3
2021-02-17 23:37:41,634 - fba.levenshtein - INFO - Matching ...
2021-02-17 23:53:22,264 - fba.levenshtein - INFO - Number of read pairs processed: 7,704,
↪799
2021-02-17 23:53:22,264 - fba.levenshtein - INFO - Number of read pairs w/ valid␣
↪barcodes: 4,607,787
2021-02-17 23:53:22,279 - fba.__main__ - INFO - Done.
```

Matrix generation

Only fragments with correctly matched cell and feature barcodes are included, while fragments with UMI lengths less than the specified value are discarded. UMI removal is performed using UMI-tools (Smith, T., et al. 2017. Genome Res. 27, 491–499.), with the starting position on read 1 set by `-us` (default 16) and the length set by `-ul` (default 12). The UMI deduplication method can be set using `-ud` (default `directional`), and the UMI deduplication mismatch threshold can be specified using `-um` (default 1).

The generated feature count matrix can be easily imported into well-established single cell analysis packages such as Seurat and Scanpy.

```
$ fba count \
    -i feature_barcoding_output.tsv.gz \
    -o matrix_featurecount.csv.gz \
    -us 16 \
    -ul 12 \
    -um 1 \
    -ud directional
```

Result summary.

69.8% (3,214,503 out of 4,607,787) of read pairs with valid cell and feature barcodes are unique fragments. 41.7% (3,214,503 out of 7,704,799) of total sequenced read pairs contribute to the final matrix.

```
2021-02-17 23:53:36,024 - fba.__main__ - INFO - fba version: 0.0.7
2021-02-17 23:53:36,024 - fba.__main__ - INFO - Initiating logging ...
2021-02-17 23:53:36,024 - fba.__main__ - INFO - Python version: 3.7
2021-02-17 23:53:36,024 - fba.__main__ - INFO - Using count subcommand ...
2021-02-17 23:53:36,024 - fba.count - INFO - UMI-tools version: 1.1.1
2021-02-17 23:53:36,027 - fba.count - INFO - UMI starting position on read 1: 16
2021-02-17 23:53:36,027 - fba.count - INFO - UMI length: 12
2021-02-17 23:53:36,027 - fba.count - INFO - UMI-tools deduplication threshold: 1
2021-02-17 23:53:36,027 - fba.count - INFO - UMI-tools deduplication method: directional
2021-02-17 23:53:36,027 - fba.count - INFO - Header line: read1_seq cell_barcode cb_num_
↪mismatches read2_seq feature_barcode fb_num_mismatches
2021-02-17 23:53:49,419 - fba.count - INFO - Number of lines processed: 4,607,787
2021-02-17 23:53:49,422 - fba.count - INFO - Number of cell barcodes detected: 1,199
2021-02-17 23:53:49,422 - fba.count - INFO - Number of features detected: 30
```

(continues on next page)

(continued from previous page)

```
2021-02-17 23:55:18,907 - fba.count - INFO - Total UMIs after deduplication: 3,214,503
2021-02-17 23:55:18,910 - fba.count - INFO - Median number of UMIs per cell: 2,564.0
2021-02-17 23:55:18,944 - fba.__main__ - INFO - Done.
```

- *CITE-seq; 8k Cord Blood Mononuclear Cells with 13 Antibodies*
- *ASAP-seq; Multiplexed CRISPR Perturbations in Primary T Cells*
- *1k Human PBMCs Stained with a Panel of TotalSeq B Antibodies, Dual Indexed*

3.3 ECCITE-seq

3.3.1 6k Single-cell Multimodal Readout of NIH-3T3, MyLa, Sez4 and PBMCs

Dataset: ECCITE-seq

Mimitou, E.P., Cheng, A., Montalbano, A., Hao, S., Stoeckius, M., Legut, M., Roush, T., Herrera, A., Papalexi, E., Ouyang, Z., et al. (2019). Multiplexed detection of proteins, transcriptomes, clonotypes and CRISPR perturbations in single cells. *Nat. Methods* **16**, 409–412.

Preparation

Download fastq files from European Nucleotide Archive.

Hashtag (Cell hashing):

```
$ curl -O ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR855/007/SRR8550947/SRR8550947_1.fastq.gz

$ curl -O ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR855/007/SRR8550947/SRR8550947_2.fastq.gz
```

Protein-tag (CITE-seq):

```
$ curl -O ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR855/006/SRR8550946/SRR8550946_1.fastq.gz

$ curl -O ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR855/006/SRR8550946/SRR8550946_2.fastq.gz
```

Guide-tag (sgRNAs):

```
$ curl -O ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR855/008/SRR8550948/SRR8550948_1.fastq.gz

$ curl -O ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR855/008/SRR8550948/SRR8550948_2.fastq.gz
```

Download pre-processed transcriptome matrix from Gene Expression Omnibus. We will need the cell-associated barcodes, which are determined by the transcriptomes.

```
$ wget https://ftp.ncbi.nlm.nih.gov/geo/samples/GSM3596nnn/GSM3596084/suppl/GSM3596084_
↪mx-cDNA.txt.gz
```

Inspect cell barcodes:

```
$ gzip -dc GSM3596084_mx-cDNA.txt.gz | head -1 | sed 's/"//g' | sed 's/ /\n/g' | sort |␣
↪grep -B1 1

CAGATCACACGTAAGG
CAGATCACACGTAAGG.1
--
CGCTATCAGCCGCCTA
CGCTATCAGCCGCCTA.1
--
TTTGCGCCAGTTCATG
TTTGCGCCAGTTCATG.1
```

It seems there are 3 colliding barcodes. We will use the first ones.

```
$ gzip -dc GSM3596084_mx-cDNA.txt.gz | head -1 | sed 's/"//g' | sed 's/ /\n/g' | sort |␣
↪grep -v 1 > cell_barcodes.txt

$ head cell_barcodes.txt

AAACCTGAGTGGTAGC
AAACCTGAGTGTTTGC
AAACCTGAGTTAGGTA
AAACCTGCAAGTTGTC
AAACCTGCAATTCCTT
AAACCTGCACAGACTT
AAACCTGCACATCCGG
AAACCTGCACGGTAAG
AAACCTGCAGACACTT
AAACCTGCATCCGCGA
```

Hashtag

Preparation

Prepare feature barcodes (hashtag-oligo sequences, from Supplementary Table 4 and 5, legend of Supplementary Figure 1):

> NIH-3T3 cells were split into 7 tubes and stained with 7 barcoded hashing antibodies (Hashtag-A to Hashtag-G), followed by washing and pooling. MyLa, Sez4 and PBMCs were stained with Hashtag_1, Hashtag_2 and Hashtag_3 respectively.

```
$ cat feature_barcodes_hashtag.tsv

Hashtag_1       ACATGTTACCGT
Hashtag_2       AGCTTACTATCC
Hashtag_3       TATCACATCGGT
Hashtag_A       AGGACCATCCAA
Hashtag_B       TCGATAATGCGA
Hashtag_C       GAGGCTGAGCTA
Hashtag_D       GTGTGACGTATT
Hashtag_E       ACTGTCTAACGG
Hashtag_F       CACATAATGACG
Hashtag_G       TAACGACGTGGT
```

QC

The first 100,000 read pairs are sampled (default, set by `-n`) for quality control. The `-t` option can be used to set the number of threads. By default, diagnostic results and plots are generated in the `qc` directory (set by `--output_directory`), and the full length of read 1 and read 2 are searched against reference cell and feature barcodes, respectively. The per base content of both read pairs and the distribution of matched barcode positions are summarized. Use `-r1_c` and/or `-r2_c` to limit the search range, and `-cb_n` and/or `-fb_n` to set the mismatch tolerance for cell and/or feature barcode matching (default 3).

```
$ fba qc \
    -1 SRR8550947_1.fastq.gz \
    -2 SRR8550947_2.fastq.gz \
    -w cell_barcodes.txt \
    -f feature_barcodes_hashtag.tsv \
    --output_directory qc
```

```
2022-01-08 16:32:43,465 - fba.__main__ - INFO - fba version: 0.0.x
2022-01-08 16:32:43,465 - fba.__main__ - INFO - Initiating logging ...
2022-01-08 16:32:43,465 - fba.__main__ - INFO - Python version: 3.7
2022-01-08 16:32:43,465 - fba.__main__ - INFO - Using qc subcommand ...
2022-01-08 16:32:44,108 - fba.qc - INFO - Summarizing per base read content ...
2022-01-08 16:32:44,108 - fba.qc - INFO - Number of read pairs to analyze: 100,000
2022-01-08 16:32:44,108 - fba.qc - INFO - Output directory: qc
2022-01-08 16:32:44,429 - fba.qc - INFO - Number of reads processed: 100,000
2022-01-08 16:32:47,187 - fba.regex - INFO - regex version: 2.5.91
2022-01-08 16:32:47,192 - fba.regex - INFO - Number of reference cell barcodes: 6,871
2022-01-08 16:32:47,192 - fba.regex - INFO - Number of reference feature barcodes: 10
2022-01-08 16:32:47,192 - fba.regex - INFO - Cell barcode maximum number of mismatches: 3
2022-01-08 16:32:47,192 - fba.regex - INFO - Feature barcode maximum number of␣
↪mismatches: 3
2022-01-08 16:32:47,192 - fba.regex - INFO - Read 1 maximum number of N allowed: inf
2022-01-08 16:32:47,192 - fba.regex - INFO - Read 2 maximum number of N allowed: inf
2022-01-08 16:32:47,192 - fba.regex - INFO - Number of read pairs to analyze: 100,000
2022-01-08 16:32:48,448 - fba.regex - INFO - Number of threads: 72
2022-01-08 16:32:48,448 - fba.regex - INFO - Chunk size: 50,000
2022-01-08 16:32:48,449 - fba.regex - INFO - Matching ...
2022-01-08 16:33:55,391 - fba.regex - INFO - Read pairs processed: 50,000
2022-01-08 16:35:02,689 - fba.regex - INFO - Read pairs processed: 100,000
```

(continues on next page)

(continued from previous page)

```
2022-01-08 16:35:04,179 - fba.qc - INFO - Summarizing barcode coordinates ...
2022-01-08 16:35:04,179 - fba.qc - INFO - Output directory: qc
2022-01-08 16:35:05,431 - fba.__main__ - INFO - Done.
```

For read 1, the first 16 bases represent cell barcodes, and the following 10 bases are UMIs (the read 1 length is 26). According to the base content plot, the GC content of cell barcodes is relatively uniform. However, UMIs are slightly enriched with T bases.

As for read 2, the per base content suggests that bases 0-12 correspond to our feature barcodes (refer to the distribution of matched barcode positions on read 2).

The detailed `qc` results are stored in the `feature_barcoding_output.tsv.gz` file. The `matching_pos` columns indicate the matched positions on reads, while the `matching_description` columns indicate mismatches in the format of substitutions:insertions:deletions.

```
$ gzip -dc qc/feature_barcoding_output.tsv.gz | head

read1_seq       cell_barcode    cb_matching_pos cb_matching_description read2_seq      ␣
↪feature_barcode fb_matching_pos fb_matching_description
NGACGGCGTGTGACGAACGCGCGCCT      GACGCGTGTGCGAAAC        1:18    0:2:1   ␣
↪ACATGTTACCGTCCCATATAAGAAAAGGCGCGCGTTCGT Hashtag_1_ACATGTTACCGT  0:12    0:0:0
NCTACACCACGGTAGAGACCTAGGTC      CACCACTGTGAGTGAC        4:19    2:0:1   ␣
↪AGGACCATCCAACCCATATAAGAAAGACCTAGGTCTCTA Hashtag_A_AGGACCATCCAA  0:12    0:0:0
GCAAACTAGATGGCGTCGACGCTTAG      GCAAACTAGATGGCGT        0:16    0:0:0   ␣
↪AGCTTACTATCCCCCATATAGAAGCTAAGCGTCGACGCC Hashtag_2_AGCTTACTATCC  0:12    0:0:0
CCTTCGAAGTGCCATTCTTTCACTGG      CCTTCGAAGTGCCATT        0:16    0:0:0   ␣
↪TATCACATCGGTCCCATATAAGAAACCAGTGAAAGAATG Hashtag_3_TATCACATCGGT  0:12    0:0:0
NGATCTGGTATGAAACGATCAGGTCA      AGATCTGGTATGAAAC        0:16    1:0:0   ␣
↪AGCTTACTATCCCCCATATAAGAAATGACCTGATCGTTT Hashtag_2_AGCTTACTATCC  0:12    0:0:0
NTCGGGATCTGTGCAAATCGGGTAGT      CTCGGGATCTGTGCAA        0:16    1:0:0   ␣
↪AGCTTACTATCCCCCATATAAGAAACTACCCGATTTGCA Hashtag_2_AGCTTACTATCC  0:12    0:0:0
GCGCGATGTACTTAGCTGCGTAGGTG      GCGCGATGTACTTAGC        0:16    0:0:0   ␣
↪TATCACATCGGTCCCATATAAGAAACACCTACGCAGCTA Hashtag_3_TATCACATCGGT  0:12    0:0:0
AACCGCGCACACAGAGCGTTTGGCCG      AACCGCGCACACAGAG        0:16    0:0:0   ␣
↪ACATGTTACCGTCCCATATAAGAAACGGCCAAACGCTCT Hashtag_1_ACATGTTACCGT  0:12    0:0:0
NAATGAACATGCGCACACGATAGTTT      no_match        NA      NA      ␣
↪TATCACATCGGTCCCATATAAGAAAAAACTATCGTGTGC NA      NA      NA
```

Barcode extraction

Both the cell and feature barcodes have identical lengths of 16 and 12, respectively. The `qc` results show a very uniform distribution of the starting and ending positions of the barcodes. The search range for read 1 is set to `0,16`, and for read 2, it is set to `0,12`. A single mismatch (`-cb_m`, `-cf_m`) is permitted for the cell and feature barcodes. Additionally, three ambiguous nucleotides (Ns) are allowed by default for both read 1 and read 2 (`-cb_n`, `-cf_n`).

```
$ fba extract \
    -1 SRR8550947_1.fastq.gz \
```

(continues on next page)

(continued from previous page)

```
    -2 SRR8550947_2.fastq.gz \
    -w cell_barcodes.txt \
    -f feature_barcodes_hashtag.tsv \
    -o feature_barcoding_output.tsv.gz \
    -r1_c 0,16 \
    -r2_c 0,12 \
    -cb_m 1 \
    -fb_m 1 \
    -cb_n 3 \
    -fb_n 3
```

Preview of result.

```
$ gzip -dc feature_barcoding_output.tsv.gz | head

read1_seq       cell_barcode    cb_num_mismatches       read2_seq       feature_barcode␣
↪fb_num_mismatches
NGACGGCGTGTGACGAacgcgcgcct      TGACGGCGTGTGACGA        1       ␣
↪ACATGTTACCGTcccatataagaaaaggcgcgcgttcgt Hashtag_1_ACATGTTACCGT  0
NCTACACCACGGTAGAgacctaggtc      CCTACACCACGGTAGA        1       ␣
↪AGGACCATCCAAcccatataagaaagacctaggtctcta Hashtag_A_AGGACCATCCAA  0
GCAAACTAGATGGCGTcgacgcttag      GCAAACTAGATGGCGT        0       ␣
↪AGCTTACTATCCccccatatagaagctaagcgtcgacgcc Hashtag_2_AGCTTACTATCC  0
CCTTCGAAGTGCCATTctttcactgg      CCTTCGAAGTGCCATT        0       ␣
↪TATCACATCGGTcccatataagaaaccagtgaaagaatg Hashtag_3_TATCACATCGGT  0
NGATCTGGTATGAAACgatcaggtca      AGATCTGGTATGAAAC        1       ␣
↪AGCTTACTATCCccccatataagaaatgacctgatcgttt Hashtag_2_AGCTTACTATCC  0
NTCGGGATCTGTGCAAatcgggtagt      CTCGGGATCTGTGCAA        1       ␣
↪AGCTTACTATCCccccatataagaaactacccgatttgca Hashtag_2_AGCTTACTATCC  0
GCGCGATGTACTTAGCtgcgtaggtg      GCGCGATGTACTTAGC        0       ␣
↪TATCACATCGGTcccatataagaaacacctacgcagcta Hashtag_3_TATCACATCGGT  0
AACCGCGCACACAGAGcgtttggccg      AACCGCGCACACAGAG        0       ␣
↪ACATGTTACCGTcccatataagaaacggccaaacgctct Hashtag_1_ACATGTTACCGT  0
TCAGATGAGAATGTTGgtggggcttc      TCAGATGAGAATGTTG        0       ␣
↪TATCACATCGGTcccatataagaaagaagccccaccaac Hashtag_3_TATCACATCGGT  0
```

Result summary.

57.0% (4,897,995 out of 8,591,807) of total read pairs have valid cell and feature barcodes.

```
2022-01-08 16:35:05,778 - fba.__main__ - INFO - fba version: 0.0.x
2022-01-08 16:35:05,778 - fba.__main__ - INFO - Initiating logging ...
2022-01-08 16:35:05,778 - fba.__main__ - INFO - Python version: 3.7
2022-01-08 16:35:05,778 - fba.__main__ - INFO - Using extract subcommand ...
2022-01-08 16:35:05,791 - fba.levenshtein - INFO - Number of reference cell barcodes: 6,
↪871
2022-01-08 16:35:05,791 - fba.levenshtein - INFO - Number of reference feature barcodes:␣
↪10
2022-01-08 16:35:05,791 - fba.levenshtein - INFO - Read 1 coordinates to search: [0, 16)
2022-01-08 16:35:05,791 - fba.levenshtein - INFO - Read 2 coordinates to search: [0, 12)
2022-01-08 16:35:05,791 - fba.levenshtein - INFO - Cell barcode maximum number of␣
↪mismatches: 1
2022-01-08 16:35:05,792 - fba.levenshtein - INFO - Feature barcode maximum number of␣
```

(continues on next page)

(continued from previous page)

```
↪mismatches: 1
2022-01-08 16:35:05,792 - fba.levenshtein - INFO - Read 1 maximum number of N allowed: 3
2022-01-08 16:35:05,792 - fba.levenshtein - INFO - Read 2 maximum number of N allowed: 3
2022-01-08 16:35:05,984 - fba.levenshtein - INFO - Matching ...
2022-01-08 16:38:39,570 - fba.levenshtein - INFO - Number of read pairs processed: 8,591,
↪807
2022-01-08 16:38:39,572 - fba.levenshtein - INFO - Number of read pairs w/ valid␣
↪barcodes: 4,897,995
2022-01-08 16:38:39,582 - fba.__main__ - INFO - Done.
```

Matrix generation

Only fragments with correctly matched cell and feature barcodes are included, while fragments with UMI lengths less than the specified value are discarded. UMI removal is performed using UMI-tools (Smith, T., et al. 2017. Genome Res. 27, 491–499.), with the starting position on read 1 set by `-us` (default 16) and the length set by `-ul` (default 12). The UMI deduplication method can be set using `-ud` (default `directional`), and the UMI deduplication mismatch threshold can be specified using `-um` (default 1).

The generated feature count matrix can be easily imported into well-established single cell analysis packages: Seurat and Scanpy.

```
$ fba count \
    -i feature_barcoding_output.tsv.gz \
    -o matrix_featurecount.csv.gz \
    -us 16 \
    -ul 10 \
    -um 1 \
    -ud directional
```

Result summary.

31.3% (1,531,088 out of 4,897,995) of read pairs with valid cell and feature barcodes are unique fragments. 17.8% (1,531,088 out of 8,591,807) of total sequenced read pairs contribute to the final matrix with an average of 55 UMIs per cell.

```
2022-01-08 16:41:49,871 - fba.__main__ - INFO - fba version: 0.0.x
2022-01-08 16:41:49,871 - fba.__main__ - INFO - Initiating logging ...
2022-01-08 16:41:49,871 - fba.__main__ - INFO - Python version: 3.7
2022-01-08 16:41:49,871 - fba.__main__ - INFO - Using demultiplex subcommand ...
2022-01-08 16:42:01,202 - fba.__main__ - INFO - Skipping arguments: "-p/--prob"
2022-01-08 16:42:01,203 - fba.demultiplex - INFO - Output directory: demultiplexed
2022-01-08 16:42:01,203 - fba.demultiplex - INFO - Demultiplexing method: 1
2022-01-08 16:42:01,203 - fba.demultiplex - INFO - UMI normalization method: clr
2022-01-08 16:42:01,203 - fba.demultiplex - INFO - Visualization: On
2022-01-08 16:42:01,203 - fba.demultiplex - INFO - Visualization method: tsne
2022-01-08 16:42:01,203 - fba.demultiplex - INFO - Loading feature count matrix: matrix_
↪featurecount.csv.gz ...
2022-01-08 16:42:01,479 - fba.demultiplex - INFO - Number of cells: 6,871
2022-01-08 16:42:01,479 - fba.demultiplex - INFO - Number of positive cells for a␣
↪feature to be included: 200
2022-01-08 16:42:01,498 - fba.demultiplex - INFO - Number of features: 10 / 10 (after␣
↪filtering / original in the matrix)
```

(continues on next page)

(continued from previous page)

```
2022-01-08 16:42:01,498 - fba.demultiplex - INFO - Features: Hashtag_1 Hashtag_2 Hashtag_
↪3 Hashtag_A Hashtag_B Hashtag_C Hashtag_D Hashtag_E Hashtag_F Hashtag_G
2022-01-08 16:42:01,499 - fba.demultiplex - INFO - Total UMIs: 1,531,088 / 1,531,088
2022-01-08 16:42:01,507 - fba.demultiplex - INFO - Median number of UMIs per cell: 55.0 /
↪ 55.0
2022-01-08 16:42:01,507 - fba.demultiplex - INFO - Demultiplexing ...
2022-01-08 16:44:11,296 - fba.demultiplex - INFO - Generating heatmap ...
2022-01-08 16:44:17,314 - fba.demultiplex - INFO - Embedding ...
2022-01-08 16:44:28,444 - fba.__main__ - INFO - Done.
```

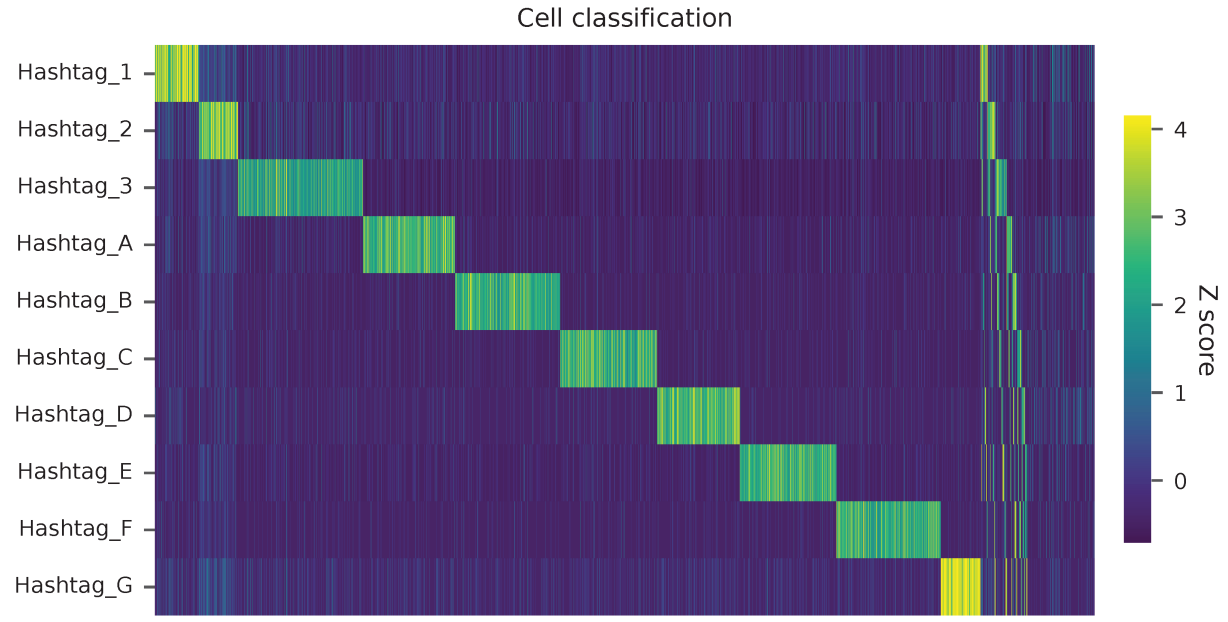
Demultiplexing

Cells are classified based on the abundance of features (hashtags, no transcriptome information used). Demultiplexing method `1` (set by `-dm`) is implemented based on the method described in Stoeckius, M., et al. (2018) with some modifications. A cell identity matrix is generated in the output directory (default `demultiplexed`, set by `--output_directory`): 0 means negative, 1 means positive. To adjust the quantile threshold for demultiplexing, use `-q` (default `0.9999`). To generate visualization plots, set `-v`.

```
$ fba demultiplex \
    -i matrix_featurecount.csv.gz \
    --output_directory demultiplexed \
    -dm 1 \
    -v
```

```
2022-01-08 16:41:49,871 - fba.__main__ - INFO - fba version: 0.0.x
2022-01-08 16:41:49,871 - fba.__main__ - INFO - Initiating logging ...
2022-01-08 16:41:49,871 - fba.__main__ - INFO - Python version: 3.7
2022-01-08 16:41:49,871 - fba.__main__ - INFO - Using demultiplex subcommand ...
2022-01-08 16:42:01,202 - fba.__main__ - INFO - Skipping arguments: "-p/--prob"
2022-01-08 16:42:01,203 - fba.demultiplex - INFO - Output directory: demultiplexed
2022-01-08 16:42:01,203 - fba.demultiplex - INFO - Demultiplexing method: 1
2022-01-08 16:42:01,203 - fba.demultiplex - INFO - UMI normalization method: clr
2022-01-08 16:42:01,203 - fba.demultiplex - INFO - Visualization: On
2022-01-08 16:42:01,203 - fba.demultiplex - INFO - Visualization method: tsne
2022-01-08 16:42:01,203 - fba.demultiplex - INFO - Loading feature count matrix: matrix_
↪featurecount.csv.gz ...
2022-01-08 16:42:01,479 - fba.demultiplex - INFO - Number of cells: 6,871
2022-01-08 16:42:01,479 - fba.demultiplex - INFO - Number of positive cells for a␣
↪feature to be included: 200
2022-01-08 16:42:01,498 - fba.demultiplex - INFO - Number of features: 10 / 10 (after␣
↪filtering / original in the matrix)
2022-01-08 16:42:01,498 - fba.demultiplex - INFO - Features: Hashtag_1 Hashtag_2 Hashtag_
↪3 Hashtag_A Hashtag_B Hashtag_C Hashtag_D Hashtag_E Hashtag_F Hashtag_G
2022-01-08 16:42:01,499 - fba.demultiplex - INFO - Total UMIs: 1,531,088 / 1,531,088
2022-01-08 16:42:01,507 - fba.demultiplex - INFO - Median number of UMIs per cell: 55.0 /
↪ 55.0
2022-01-08 16:42:01,507 - fba.demultiplex - INFO - Demultiplexing ...
2022-01-08 16:44:11,296 - fba.demultiplex - INFO - Generating heatmap ...
2022-01-08 16:44:17,314 - fba.demultiplex - INFO - Embedding ...
2022-01-08 16:44:28,444 - fba.__main__ - INFO - Done.
```

Heatmap of the relative abundance of features (hashtags) across all cells. Each column represents a single cell.



t-SNE embedding of cells based on the abundance of features (hashtags, no transcriptome information used). Colors indicate the hashtag status for each cell, as called by FBA.

Preview the demultiplexing result: the numbers of singlets, multiplets and negative cells. In summary, the numbers of MyLa, Sez4, PBMCs and NIH-3T3 cells demultiplexed are 324, 283, 914 and 4,518 respectively.

```
In [1]: import pandas as pd

In [2]: m = pd.read_csv("demultiplexed/matrix_cell_identity.csv.gz", index_col=0)

In [3]: m.loc[:, m.sum(axis=0) == 1].sum(axis=1)
Out[3]:
Hashtag_1    324
Hashtag_2    283
Hashtag_3    914
Hashtag_A    673
Hashtag_B    771
Hashtag_C    709
Hashtag_D    603
Hashtag_E    707
Hashtag_F    764
Hashtag_G    291
dtype: int64

In [4]: sum(m.sum(axis=0) > 1)
Out[4]: 341

In [5]: sum(m.sum(axis=0) == 0)
Out[5]: 491
```

Protein-tag

Preparation

Prepare feature barcodes (protein-tag sequences, from Supplementary Table 3, legend of Supplementary Figure 1):

> All cells were stained with a mix of anti-human CD29 and anti-mouse CD29 antibodies.

```
$ cat feature_barcodes_CD29.tsv

hCD29   AATAGCGGAGCC
mCD29   CGAAGACCAAGA
```

QC

```
$ fba qc \
    -1 SRR8550946_1.fastq.gz \
    -2 SRR8550946_2.fastq.gz \
    -w cell_barcodes.txt \
    -f feature_barcodes_CD29.tsv \
    --output_directory qc
```

```
2022-01-08 12:29:00,323 - fba.__main__ - INFO - fba version: 0.0.x
2022-01-08 12:29:00,323 - fba.__main__ - INFO - Initiating logging ...
2022-01-08 12:29:00,323 - fba.__main__ - INFO - Python version: 3.7
2022-01-08 12:29:00,323 - fba.__main__ - INFO - Using qc subcommand ...
2022-01-08 12:29:00,896 - fba.qc - INFO - Summarizing per base read content ...
2022-01-08 12:29:00,896 - fba.qc - INFO - Number of read pairs to analyze: 100,000
2022-01-08 12:29:00,896 - fba.qc - INFO - Output directory: qc
2022-01-08 12:29:01,119 - fba.qc - INFO - Number of reads processed: 100,000
2022-01-08 12:29:03,848 - fba.regex - INFO - regex version: 2.5.91
2022-01-08 12:29:03,852 - fba.regex - INFO - Number of reference cell barcodes: 6,871
2022-01-08 12:29:03,852 - fba.regex - INFO - Number of reference feature barcodes: 2
2022-01-08 12:29:03,852 - fba.regex - INFO - Cell barcode maximum number of mismatches: 3
2022-01-08 12:29:03,852 - fba.regex - INFO - Feature barcode maximum number of␣
↪mismatches: 3
2022-01-08 12:29:03,852 - fba.regex - INFO - Read 1 maximum number of N allowed: inf
2022-01-08 12:29:03,852 - fba.regex - INFO - Read 2 maximum number of N allowed: inf
2022-01-08 12:29:03,852 - fba.regex - INFO - Number of read pairs to analyze: 100,000
2022-01-08 12:29:05,062 - fba.regex - INFO - Number of threads: 72
2022-01-08 12:29:05,062 - fba.regex - INFO - Chunk size: 50,000
2022-01-08 12:29:05,062 - fba.regex - INFO - Matching ...
2022-01-08 12:31:07,626 - fba.regex - INFO - Read pairs processed: 50,000
2022-01-08 12:33:11,971 - fba.regex - INFO - Read pairs processed: 100,000
2022-01-08 12:33:13,341 - fba.qc - INFO - Summarizing barcode coordinates ...
2022-01-08 12:33:13,341 - fba.qc - INFO - Output directory: qc
2022-01-08 12:33:14,203 - fba.__main__ - INFO - Done.
```

The per base content analysis of read 2 suggests that bases 0-12 correspond to our feature barcodes, as indicated by the distribution of matched barcode positions (see fba `qc` results).

The detailed `qc` results are stored in the `feature_barcoding_output.tsv.gz` file. The `matching_pos` columns indicate the matched positions on reads, while the `matching_description` columns indicate mismatches in the format of substitutions:insertions:deletions.

```
$ gzip -dc feature_barcoding_output.tsv.gz | head -20

read1_seq       cell_barcode    cb_matching_pos cb_matching_description read2_seq      ␣
↪feature_barcode fb_matching_pos fb_matching_description
NAGCCGATCACGCGGTCTGGTGGGCA      CAGCCGATCACGCGGT        0:16    1:0:0   ␣
↪AATTCCGTCAGATGACCCATATAAGAAATGCCCACCAGA no_match        NA      NA
NACACAAGTCTCCCTAGGCCTGTGAC      CAAGATCTCCCTTGTG        4:18    1:0:2   ␣
↪AATTCCGTCAGATGACCCATATAAGAAAGTCACAGGCCT no_match        NA      NA
TGTATTCAGGAGTCTGAATTGTAATA      CTCGAGGAGATCTGAA        4:18    1:0:2   ␣
↪AATTCCGTCAGATGACCCATATAAGAAATATTACAATTC no_match        NA      NA
NCGTCAAGTGCCTGGTGCTCCTGTAT      ACGTCAAGTGCCTGGT        0:16    1:0:0   ␣
↪CGAAGACCAAGACCCATATAAGAAAATACAGGAGCACCA mCD29_CGAAGACCAAGA      0:12    0:0:0
CTAGAGTAGATCGATACGCGGATGGT      CTAGAGTAGATCTGAA        0:15    1:0:1   ␣
↪CGAAGACCAAGACCCATATAAGAAAACCATCCGCGTATC mCD29_CGAAGACCAAGA      0:12    0:0:0
NATCGGGGTCGAACAGGGAGCGTCAG      AACCGCGAGCGTCAAG        11:26   2:0:1   ␣
↪AATTCCGTCAGATGACCCATATAAGAAACTGACGCTCCC no_match        NA      NA
AAAGCAAAGACAAGCCAGTATTTACG      ACACCAAGTCCAGTAT        7:21    1:0:2   ␣
↪AATTCCGTCAGATGACCCATATAAGAAACGTAAATACTG no_match        NA      NA
AAGCCGCGTCTCAACAACAGACTACG      AAGCCGCGTCTCAACA        0:16    0:0:0   ␣
↪AATTCCGTCAGATGACCCATATAAGAAACGTAGTCTGTT no_match        NA      NA
ACGGAGAAGCGCCTCACTCTATCTTC      ATCCGAAAGCGCCTCA        0:16    1:1:1   ␣
↪AATTCCGTCAGATGACCCATATAAGAAAGAAGATAGAGT no_match        NA      NA
TCGTACCCACCATCCTACACCGGCAC      CGGAGTCCACCATCCT        1:16    2:0:1   ␣
↪AATTCCGTCAGATGACCCATATAAGAAAAGTGCCGGTGT no_match        NA      NA
ACAGCTACAGTATGCTTAAAAACAGG      ACAGCTAAGTACTTGC        0:15    0:1:2   ␣
↪AATTCCGTCAGATGACCCATATAAGAAACCTGTTTTTAA no_match        NA      NA
NTAAGACGTCTAAACCGAGCTGGCAC      CAGCAGCGTCTAAACC        2:16    1:0:2   ␣
↪AATTCCGTCAGATGACCCATATAAGAAAGTGCCAGCTCG no_match        NA      NA
TCTCTAATCCAGTATGCCTCTCTTGA      AATCCAGTCCGCATCT        5:21    3:0:0   ␣
↪AATTCCGTCAGATGACCCATATAAGAAATCAAGAGAGGC mCD29_CGAAGACCAAGA      23:34   2:0:1
GACAGAGTCGCATGATTAAAAATCAA      ACAGCTATCGCATGAT        1:16    2:0:1   ␣
↪AATTCCGTCAGATGACCCATATAAGAAATTGATTTTTAA no_match        NA      NA
NTGAGGGTCTCCAACCGCTTTCTAAT      GATCGCGTCTCCAACC        2:16    1:0:2   ␣
↪AATTCCGTCAGATGACCCATATAAGAAAATTAGAAAGCG no_match        NA      NA
NAGCTGGTCGCCAAATACGTATAACT      GCCAAATTCGATAGAA        9:24    2:0:1   ␣
↪AATTCCGTCAGATGACCCATATAAGAAAAGTTATACGTA no_match        NA      NA
GTGCGGTAGTGTTGAAGGTTTATAAT      AGTAGTCAGAAGGTTT        7:21    1:0:2   ␣
↪AATTCCGTCAGATGACCCATATAAGAAAATTATAAACCT no_match        NA      NA
NGAGCACGTGCCTGTGCTACTAGTAC      ACGCCAGGTGCCTGTG        2:16    1:0:2   ␣
↪AATTCCGTCAGATGACCCATATAAGAAAGTACTAGTAGC no_match        NA      NA
CGGGTCAAGACACTAAAAAACCTGCT      ACACTGACAAACTGCT        9:26    2:1:0   ␣
↪AATTCCGTCAGATGACCCATATAAGAAAAGCAGGTTTTT no_match        NA      NA
```

Barcode extraction

```
$ fba extract \
    -1 SRR8550946_1.fastq.gz \
    -2 SRR8550946_2.fastq.gz \
    -w cell_barcodes.txt \
    -f feature_barcodes_CD29.tsv \
    -o feature_barcoding_output.tsv.gz \
    -r1_c 0,16 \
    -r2_c 0,12 \
    -cb_m 2
```

Preview of result.

```
$ gzip -dc feature_barcoding_output.tsv.gz | head

read1_seq       cell_barcode    cb_num_mismatches       read2_seq       feature_barcode␣
↪fb_num_mismatches
NCGTCAAGTGCCTGGTgctcctgtat      ACGTCAAGTGCCTGGT        1       ␣
↪CGAAGACCAAGAcccatataagaaaatacaggagcacca mCD29_CGAAGACCAAGA      0
CTAGAGTAGATCGATAcgcggatggt      CTAGAGTAGATCTGAA        2       ␣
↪CGAAGACCAAGAcccatataagaaaaccatccgcgtatc mCD29_CGAAGACCAAGA      0
GGAAAGCCAATCCGATatcccgtatc      GGAAAGCCAATCCGAT        0       ␣
↪CGAAGACCAAGAcccatataagaaagatacgggatatcg mCD29_CGAAGACCAAGA      0
GCAAACTCAAACAACAaaccttaagg      GCAAACTCAAACAACA        0       ␣
↪CGAAGACCAAGAcccatataagaaaccttaaggtttgtt mCD29_CGAAGACCAAGA      0
GTTACAGGTCTCCACTaatagaaggg      GTTACAGGTCTCCACT        0       ␣
↪AATAGCGGAGCCcccatataagaaacccttctattagtg hCD29_AATAGCGGAGCC      0
CGGACACAGGGCTTCCaaagttttag      CGGACACAGGGCTTCC        0       ␣
↪AATAGCGGAGCCcccatataagaaactaaaactttggaa hCD29_AATAGCGGAGCC      0
TACGGATTCACCACCTcaccctcttg      TACGGATTCACCACCT        0       ␣
↪CGAAGACCAAGAcccatataagaaacaagagggtgaggt mCD29_CGAAGACCAAGA      0
GCTTCCAGTTCCCTTGcagacaagag      GCTTCCAGTTCCCTTG        0       ␣
↪CGAAGACCAAGAcccatataagaaactcttgtctgcaag mCD29_CGAAGACCAAGA      0
CTGCCTAGTGAAATCAatggggaggc      CTGCCTAGTGAAATCA        0       ␣
↪CGAAGACCAAGAcccatataagaaagcctccccattgat mCD29_CGAAGACCAAGA      0
```

Result summary.

5.9% (256,759 out of 4,372,604) of total read pairs have valid cell and feature barcodes.

```
2022-01-08 12:33:14,547 - fba.__main__ - INFO - fba version: 0.0.x
2022-01-08 12:33:14,547 - fba.__main__ - INFO - Initiating logging ...
2022-01-08 12:33:14,547 - fba.__main__ - INFO - Python version: 3.7
2022-01-08 12:33:14,547 - fba.__main__ - INFO - Using extract subcommand ...
2022-01-08 12:33:14,561 - fba.levenshtein - INFO - Number of reference cell barcodes: 6,
↪871
2022-01-08 12:33:14,561 - fba.levenshtein - INFO - Number of reference feature barcodes:␣
↪2
2022-01-08 12:33:14,561 - fba.levenshtein - INFO - Read 1 coordinates to search: [0, 16)
2022-01-08 12:33:14,561 - fba.levenshtein - INFO - Read 2 coordinates to search: [0, 12)
2022-01-08 12:33:14,561 - fba.levenshtein - INFO - Cell barcode maximum number of␣
↪mismatches: 2
```

(continues on next page)

(continued from previous page)

```
2022-01-08 12:33:14,561 - fba.levenshtein - INFO - Feature barcode maximum number of␣
↪mismatches: 1
2022-01-08 12:33:14,561 - fba.levenshtein - INFO - Read 1 maximum number of N allowed: 3
2022-01-08 12:33:14,561 - fba.levenshtein - INFO - Read 2 maximum number of N allowed: 3
2022-01-08 12:33:15,856 - fba.levenshtein - INFO - Matching ...
2022-01-08 12:37:37,930 - fba.levenshtein - INFO - Number of read pairs processed: 4,372,
↪604
2022-01-08 12:37:37,931 - fba.levenshtein - INFO - Number of read pairs w/ valid␣
↪barcodes: 256,759
2022-01-08 12:37:37,985 - fba.__main__ - INFO - Done.
```

Matrix generation

```
$ fba count \
    -i feature_barcoding_output.tsv.gz \
    -o matrix_featurecount.csv.gz \
    -us 16 \
    -ul 10 \
    -um 1 \
    -ud directional
```

Result summary.

96.2% (246,996 out of 256,759) of read pairs with valid cell and feature barcodes are unique fragments. 5.6% (246,996 out of 4,372,604) of total sequenced read pairs contribute to the final matrix with an average of 29 UMIs per cell.

```
2022-01-08 12:37:38,233 - fba.__main__ - INFO - fba version: 0.0.x
2022-01-08 12:37:38,233 - fba.__main__ - INFO - Initiating logging ...
2022-01-08 12:37:38,233 - fba.__main__ - INFO - Python version: 3.7
2022-01-08 12:37:38,233 - fba.__main__ - INFO - Using count subcommand ...
2022-01-08 12:37:39,087 - fba.count - INFO - UMI-tools version: 1.1.1
2022-01-08 12:37:39,090 - fba.count - INFO - UMI starting position on read 1: 16
2022-01-08 12:37:39,090 - fba.count - INFO - UMI length: 10
2022-01-08 12:37:39,090 - fba.count - INFO - UMI-tools deduplication threshold: 1
2022-01-08 12:37:39,090 - fba.count - INFO - UMI-tools deduplication method: directional
2022-01-08 12:37:39,090 - fba.count - INFO - Header line: read1_seq cell_barcode cb_num_
↪mismatches read2_seq feature_barcode fb_num_mismatches
2022-01-08 12:37:39,587 - fba.count - INFO - Number of lines processed: 256,759
2022-01-08 12:37:39,590 - fba.count - INFO - Number of cell barcodes detected: 6,871
2022-01-08 12:37:39,590 - fba.count - INFO - Number of features detected: 2
2022-01-08 12:37:40,917 - fba.count - INFO - Total UMIs after deduplication: 246,996
2022-01-08 12:37:40,926 - fba.count - INFO - Median number of UMIs per cell: 29.0
2022-01-08 12:37:40,983 - fba.__main__ - INFO - Done.
```

t-SNE embedding of cells based on the abundance of features (hashtags, no transcriptome information used). Colors indicate the hashtag status for each cell, as called by FBA, and the abundance of protein tags. This is a re-creation of Fig. 1c in Mimitou, E.P., et al. (2019) (The embedding is based on hashtags, not the transcriptomes).

Guide-tag

Preparation

Prepare feature barcodes (guide-tag sequences, from Supplementary Table 2, Supplementary Figure 1c).

```
$ cat feature_barcodes_guide-tag.tsv

mNT1    CGCGGAGCCGAATACCTCG
mNT2    CGTCGAACCTCCGTGAAAG
mNT3    ATCGAGCCGAACTGCAACT
mNT4    AAGGCGTTCGCCTTACACG
mNT5    GACATTTAGTACCCGGAGT
mNT6    CTCGTTCCCTAACGGCGCG
mNT7    CCCGTAGACGGTCGAACAA
mNT8    CCATATCGCACCCGATGGG
mNT9    TTACTAGCAGGTGACGCCC
mNT10   AATACGTTGCGAGTAGAAG
```

QC

```
$ fba qc \
    -1 SRR8550948_1.fastq.gz \
    -2 SRR8550948_2.fastq.gz \
    -w cell_barcodes.txt \
    -f feature_barcodes_guide-tag.tsv \
    --output_directory qc
```

```
2022-01-08 14:39:46,616 - fba.__main__ - INFO - fba version: 0.0.x
2022-01-08 14:39:46,616 - fba.__main__ - INFO - Initiating logging ...
2022-01-08 14:39:46,617 - fba.__main__ - INFO - Python version: 3.7
2022-01-08 14:39:46,617 - fba.__main__ - INFO - Using qc subcommand ...
2022-01-08 14:39:47,209 - fba.qc - INFO - Summarizing per base read content ...
2022-01-08 14:39:47,209 - fba.qc - INFO - Number of read pairs to analyze: 100,000
2022-01-08 14:39:47,209 - fba.qc - INFO - Output directory: qc
2022-01-08 14:39:47,428 - fba.qc - INFO - Number of reads processed: 100,000
2022-01-08 14:39:50,099 - fba.regex - INFO - regex version: 2.5.91
2022-01-08 14:39:50,103 - fba.regex - INFO - Number of reference cell barcodes: 6,871
2022-01-08 14:39:50,103 - fba.regex - INFO - Number of reference feature barcodes: 10
2022-01-08 14:39:50,103 - fba.regex - INFO - Cell barcode maximum number of mismatches: 3
2022-01-08 14:39:50,103 - fba.regex - INFO - Feature barcode maximum number of␣
↪mismatches: 3
2022-01-08 14:39:50,103 - fba.regex - INFO - Read 1 maximum number of N allowed: inf
2022-01-08 14:39:50,103 - fba.regex - INFO - Read 2 maximum number of N allowed: inf
2022-01-08 14:39:50,103 - fba.regex - INFO - Number of read pairs to analyze: 100,000
2022-01-08 14:39:51,295 - fba.regex - INFO - Number of threads: 72
2022-01-08 14:39:51,296 - fba.regex - INFO - Chunk size: 50,000
2022-01-08 14:39:51,296 - fba.regex - INFO - Matching ...
2022-01-08 14:40:49,770 - fba.regex - INFO - Read pairs processed: 50,000
2022-01-08 14:41:50,088 - fba.regex - INFO - Read pairs processed: 100,000
2022-01-08 14:41:51,366 - fba.qc - INFO - Summarizing barcode coordinates ...
```

(continues on next page)

(continued from previous page)

```
2022-01-08 14:41:51,366 - fba.qc - INFO - Output directory: qc
2022-01-08 14:41:52,542 - fba.__main__ - INFO - Done.
```

As for read 2, the per base content suggests that bases 15-34 correspond to our feature barcodes (see the distribution of matched barcode positions on read 2).

The detailed `qc` results are stored in the `feature_barcoding_output.tsv.gz` file. The `matching_pos` columns indicate the matched positions on reads, while the `matching_description` columns indicate mismatches in the format of substitutions:insertions:deletions.

```
$ gzip -dc feature_barcoding_output.tsv.gz | head

read1_seq       cell_barcode    cb_matching_pos cb_matching_description read2_seq      ␣
↪feature_barcode fb_matching_pos fb_matching_description
NTAAGAGGTCTGCAATCTATATGCAA      AGCGGTCCATGCAATC        3:17    1:0:2   ␣
↪TTCTAGCTCTAAAACCCCGTAGACGGTCGAACAATCCCC mNT7_CCCGTAGACGGTCGAACAA        15:34   0:0:0
ATCCACCGTCATATCGACATGCCACA       ATCCACCGTCATATCG        0:16    0:0:0   ␣
↪TTCTAGCTCTAAAACCCCGTAGACGGTCGAACAATCCCC mNT7_CCCGTAGACGGTCGAACAA        15:34   0:0:0
TGCCAAACACTGAAGGATGTCGCCAC       ACACTGAAGGATGTAT        6:22    2:0:0   ␣
↪TTCTAGCTCTAAAACGACATTTAGTACCCGGAGTCCCCA mNT5_GACATTTAGTACCCGGAGT        15:34   0:0:0
NTCGAAGAGGGTATCGTGAAGTGCTT      AGGGATGGTGAAGGCT        7:25    1:2:0   ␣
↪TTCTAGCTCTAAAACCCATATCGCACCCGATGGGTCCCC mNT8_CCATATCGCACCCGATGGG        15:34   0:0:0
CCTTACGAGTGGACGTGCAGTCAGGT      CCTTACGAGTGGACGT        0:16    0:0:0   ␣
↪TTCTAGCTCTAAAACCTCGTTCCCTAACGGCGCGGCCCA mNT6_CTCGTTCCCTAACGGCGCG        15:34   0:0:0
GGTACAGGATCGCAAACGCGCAAATT      ACACTGATCGCAAACT        3:18    2:0:1   ␣
↪AATTCCGTCAGATGACCCATATAAGAAATTTGCGCGTTT no_match        NA      NA
NAAATGATCCAAACTGTAAGGGAAGC      AAACCTGTCCAAACTG        1:16    0:1:2   ␣
↪TTCTAGCTCTAAAACGACATTTAGTACCCGGAGTCCCCA mNT5_GACATTTAGTACCCGGAGT        15:34   0:0:0
NCACATAAGGAGTTGCGCAACCGCGA      CACATAGAGTTCGCGC        1:18    0:2:1   ␣
↪TTCTAGCTCTAAAACCCATATCGCACCCGATGGGTCCCC mNT8_CCATATCGCACCCGATGGG        15:34   0:0:0
AGGGTGAAGCGCTTATTAATCGAAGG      AGGGTGAAGCGATGAC        0:16    3:0:0   ␣
↪AATTCCGTCAGATGACCCATATAAGAAACCTTCGATTAA no_match        NA      NA
```

Barcode extraction

```
$ fba extract \
    -1 SRR8550948_1.fastq.gz \
    -2 SRR8550948_2.fastq.gz \
    -w cell_barcodes.txt \
    -f feature_barcodes_guide-tag.tsv \
    -o feature_barcoding_output.tsv.gz \
    -r1_c 0,16 \
    -r2_c 15,34 \
    -cb_m 2
```

Result summary.

63.4% (24,425,023 out of 38,537,829) of total read pairs have valid cell and feature barcodes.

```
2022-01-08 14:41:52,877 - fba.__main__ - INFO - fba version: 0.0.x
2022-01-08 14:41:52,877 - fba.__main__ - INFO - Initiating logging ...
2022-01-08 14:41:52,877 - fba.__main__ - INFO - Python version: 3.7
2022-01-08 14:41:52,877 - fba.__main__ - INFO - Using extract subcommand ...
2022-01-08 14:41:52,894 - fba.levenshtein - INFO - Number of reference cell barcodes: 6,
↪871
2022-01-08 14:41:52,895 - fba.levenshtein - INFO - Number of reference feature barcodes:␣
↪10
2022-01-08 14:41:52,895 - fba.levenshtein - INFO - Read 1 coordinates to search: [0, 16)
2022-01-08 14:41:52,895 - fba.levenshtein - INFO - Read 2 coordinates to search: [15, 34)
2022-01-08 14:41:52,895 - fba.levenshtein - INFO - Cell barcode maximum number of␣
↪mismatches: 2
2022-01-08 14:41:52,895 - fba.levenshtein - INFO - Feature barcode maximum number of␣
↪mismatches: 1
2022-01-08 14:41:52,895 - fba.levenshtein - INFO - Read 1 maximum number of N allowed: 3
2022-01-08 14:41:52,895 - fba.levenshtein - INFO - Read 2 maximum number of N allowed: 3
2022-01-08 14:41:54,222 - fba.levenshtein - INFO - Matching ...
2022-01-08 14:54:23,469 - fba.levenshtein - INFO - Read pairs processed: 10,000,000
2022-01-08 15:06:57,696 - fba.levenshtein - INFO - Read pairs processed: 20,000,000
2022-01-08 15:19:24,990 - fba.levenshtein - INFO - Read pairs processed: 30,000,000
2022-01-08 15:30:09,131 - fba.levenshtein - INFO - Number of read pairs processed: 38,
↪537,829
2022-01-08 15:30:09,131 - fba.levenshtein - INFO - Number of read pairs w/ valid␣
↪barcodes: 24,425,023
2022-01-08 15:30:09,188 - fba.__main__ - INFO - Done.
```

Matrix generation

```
$ fba count \
    -i feature_barcoding_output.tsv.gz \
    -o matrix_featurecount.csv.gz \
    -us 16 \
    -ul 10 \
    -um 1 \
    -ud directional
```

Result summary.

4.3% (1,050,888 out of 24,425,023) of read pairs with valid cell and feature barcodes are unique fragments. 2.7% (1,050,888 out of 38,537,829) of total sequenced read pairs contribute to the final matrix with an average of 102 UMIs per cell.

```
2022-01-08 15:30:09,447 - fba.__main__ - INFO - fba version: 0.0.x
2022-01-08 15:30:09,447 - fba.__main__ - INFO - Initiating logging ...
2022-01-08 15:30:09,447 - fba.__main__ - INFO - Python version: 3.7
2022-01-08 15:30:09,447 - fba.__main__ - INFO - Using count subcommand ...
2022-01-08 15:30:10,313 - fba.count - INFO - UMI-tools version: 1.1.1
2022-01-08 15:30:10,316 - fba.count - INFO - UMI starting position on read 1: 16
2022-01-08 15:30:10,316 - fba.count - INFO - UMI length: 10
2022-01-08 15:30:10,316 - fba.count - INFO - UMI-tools deduplication threshold: 1
2022-01-08 15:30:10,316 - fba.count - INFO - UMI-tools deduplication method: directional
```

(continues on next page)

(continued from previous page)

```
2022-01-08 15:30:10,316 - fba.count - INFO - Header line: read1_seq cell_barcode cb_num_
↪mismatches read2_seq feature_barcode fb_num_mismatches
2022-01-08 15:30:53,780 - fba.count - INFO - Number of lines processed: 24,425,023
2022-01-08 15:30:53,785 - fba.count - INFO - Number of cell barcodes detected: 6,867
2022-01-08 15:30:53,785 - fba.count - INFO - Number of features detected: 10
2022-01-08 15:31:18,853 - fba.count - INFO - Total UMIs after deduplication: 1,050,888
2022-01-08 15:31:18,862 - fba.count - INFO - Median number of UMIs per cell: 102.0
2022-01-08 15:31:19,003 - fba.__main__ - INFO - Done.
```

t-SNE embedding of cells based on the abundance of features (hashtags, no transcriptome information used). Colors indicate the guide tag abundance for each cell, as caculated by FBA. This is a re-creation of Fig. 1c (iv) in Mimitou, E.P., et al. (2019) (The embedding is based on hashtags, not the transcriptomes).

- *6k Single-cell Multimodal Readout of NIH-3T3, MyLa, Sez4 and PBMCs*

3.4 PHAGE-ATAC

3.4.1 PHAGE-ATAC; Anti-CD8 Phage Hashing Single-cell ATAC-seq Using CD8 T Cells from Four Human Donors

Dataset: PHAGE-ATAC: four anti-CD8 phage hashtags and a subsequent hashing experiment using CD8 T cells from four human donors

Fiskin, E., Lareau, C.A., Ludwig, L.S., Eraslan, G., Liu, F., Ring, A.M., Xavier, R.J., and Regev, A. (2021). Single-cell profiling of proteins and chromatin accessibility using PHAGE-ATAC. *Nat. Biotechnol.* **40**, 374–381.

Preparation

Download fastq files from Gene Expression Omnibus.

Inspect fastq files (This is a phage-derived tag library built on top of the single-cell ATAC-seq library, we will need all 3 end reads).

```
$ zcat SRR12588752_1.fastq.gz | head

@SRR12588752.1 NB501583:726:HMCKKBGXF:1:11101:16677:1031 length=34
CAGCTNTTCCTGCGCTCGACACCGTTACTTGTGT
+SRR12588752.1 NB501583:726:HMCKKBGXF:1:11101:16677:1031 length=34
AAAAA#EEEEEEEEEEEEEEEEEEEEEEEEEE
@SRR12588752.2 NB501583:726:HMCKKBGXF:1:11101:14655:1033 length=34
CAGCTNTTCCTGCGCTGCTTACCGTAACTTGTGT
+SRR12588752.2 NB501583:726:HMCKKBGXF:1:11101:14655:1033 length=34
```

(continues on next page)

(continued from previous page)

```
AAAAA#EEEEEEEAEEEEEEEEEEEEEEEEEE
@SRR12588752.3 NB501583:726:HMCKKBGXF:1:11101:17414:1034 length=34
CAGCTNTTCCTGCGCTCGACACCGTTACTTGTGT

$ zcat SRR12588752_2.fastq.gz | head

@SRR12588752.1 NB501583:726:HMCKKBGXF:1:11101:16677:1031 length=16
NNNNAGNNNNNATGNN
+SRR12588752.1 NB501583:726:HMCKKBGXF:1:11101:16677:1031 length=16
####AE#####EEE##
@SRR12588752.2 NB501583:726:HMCKKBGXF:1:11101:14655:1033 length=16
NNAACTNNNNNAAAAN
+SRR12588752.2 NB501583:726:HMCKKBGXF:1:11101:14655:1033 length=16
##//A6#####////#
@SRR12588752.3 NB501583:726:HMCKKBGXF:1:11101:17414:1034 length=16
NNTCAGTNNNTCAGGN

$ zcat SRR12588752_3.fastq.gz | head

@SRR12588752.1 NB501583:726:HMCKKBGXF:1:11101:16677:1031 length=34
GATACCGNGGNNNNNNNNNNNNNNNNNNNNNN
+SRR12588752.1 NB501583:726:HMCKKBGXF:1:11101:16677:1031 length=34
AAAAAEE#EE########################
@SRR12588752.2 NB501583:726:HMCKKBGXF:1:11101:14655:1033 length=34
GATACCGCGGTGTNTNANNNCNNNNNAGNNNCNN
+SRR12588752.2 NB501583:726:HMCKKBGXF:1:11101:14655:1033 length=34
AAAAAEEEEEEEE#E#E###E#####EE###E##
@SRR12588752.3 NB501583:726:HMCKKBGXF:1:11101:17414:1034 length=34
GATACCGCGGTGTATNANNNGNNNNNAANNNGNN
```

Prepare cell barcodes (downloaded from the manuscript's GitHub repository). These are the cell-associated barcodes, they are generated based on the single-cell ATAC-seq library.

```
$ wget https://raw.githubusercontent.com/evgenijfiskin/phage-atac/master/cd8_hashing/
↪data/barcodes.tsv
```

Inspect cell barcodes.

```
$ wc -l barcodes.tsv

8366

$ head barcodes.tsv

AAACGAAAGACCATAA-1
AAACGAAAGGAGGCGA-1
AAACGAAAGGCGATTG-1
AAACGAACAAGAGATT-1
AAACGAACACAGTTAC-1
AAACGAACACTGATTG-1
AAACGAACACTTACAG-1
AAACGAACAGCAGGTA-1
```

(continues on next page)

(continued from previous page)

```
AAACGAACAGCGTACC-1
AAACGAAGTTGCAGAG-1
```

Prepare feature barcodes. CDR3 barcode sequences (phage-derived tags, PDTs) can be found in Supplementary Table 4 and are truncated to only keep the variable parts.

Table 2: CDR3 barcode sequences

CD8Nb PH-A	GATACCGCGGTGTATTATTGCGCAAAGGACGCGG
CD8Nb PH-B	GATACCGCGGTGTATTATTGCGCTAAAGACGCGG
CD8Nb PH-C	CAGCTCTTCCTGCGCTGCTTACCGTAACTTGTGT
CD8Nb PH-D	CAGCTCTTCCTGCGCTGCTTACAGTGACCTGTGT

```
$ cat feature_barcodes_R3_truncated.tsv

CD8Nb_PH-A      CAAAGGACGCGG
CD8Nb_PH-B      CTAAAGACGCGG

$ cat feature_barcodes_R1_truncated.tsv

CD8Nb_PH-C      CGTAACTTGTGT
CD8Nb_PH-D      AGTGACCTGTGT
```

First, we screened reads for the constant sequence (`GATACCGCGGTGTATTATTGCG`) at the beginning of the CDR3 barcode sequences on read 3 using cutadapt (version 3.7).

```
$ cutadapt \
    --cores 0 \
    --front GATACCGCGGTGTATTATTGCG \
    --minimum-length 12:16 \
    --trimmed-only \
    --output SRR12588752_3_trimmed.fq.gz --paired-output SRR12588752_2_trimmed.fq.gz \
    SRR12588752_3.fastq.gz SRR12588752_2.fastq.gz
```

Preview the filtering result: 51,140,637 out of 54,274,791 (94.2%) read pairs are kept for phage-derived tag (PDT) identification.

```
== Read fate breakdown ==
Pairs that were too short:             652,917 (1.2%)
Pairs discarded as untrimmed:        2,481,237 (4.6%)
Pairs written (passing filters):    51,140,637 (94.2%)
```

Then, for read 1 (`CAGCTCTTCCTGCGCTGCTTAC`).

```
$ cutadapt \
    --cores 0 \
    --front CAGCTCTTCCTGCGCTGCTTAC \
```

(continues on next page)

(continued from previous page)

```
    --minimum-length 12:16 \
    --trimmed-only \
    --output SRR12588752_1_trimmed.fq.gz --paired-output SRR12588752_2_trimmed.fq.gz \
    SRR12588752_1.fastq.gz SRR12588752_2.fastq.gz
```

Preview the filtering result: 25,988,762 out of 54,274,791 (47.9%) read pairs are kept for phage-derived tag (PDT) identification.

```
== Read fate breakdown ==
Pairs that were too short:               16,644 (0.0%)
Pairs discarded as untrimmed:        28,269,385 (52.1%)
Pairs written (passing filters):     25,988,762 (47.9%)
```

QC

The first 10,000 read pairs are sampled (set by `-n`, default `100,000`) for quality control. The `-t` option can be used to set the number of threads. By default, diagnostic results and plots are generated in the `qc` directory (set by `--output_directory`), and the full length of read 1 and read 2 are searched against reference cell and feature barcodes, respectively. The per base content of both read pairs and the distribution of matched barcode positions are summarized. Use `-r1_c` and/or `-r2_c` to limit the search range, and `-cb_n` and/or `-fb_n` to set the mismatch tolerance for cell and/or feature barcode matching (default `3`).

This library was constructed using 10x Genomics' Chromium Single Cell ATAC Reagent Kits, where the 16-base pair cell barcode is sequenced during the i5 index read. In Cell Ranger ATAC, the raw 16 bp sequences may be transformed into their reverse-complement counterparts as cell barcodes in the outputs. To achieve the same result in `fba`, use `-cb_rc` to reverse-complement the cell barcode sequences.

To achieve the same result in fba,

R3

```
$ fba qc \
    -1 SRR12588752_2_trimmed.fq.gz \
    -2 SRR12588752_3_trimmed.fq.gz \
    -w barcodes.tsv \
    -f feature_barcodes_R3_truncated.txt \
    -cb_rc \
    -n 10000
```

This library is constructed using the Chromium Single Cell ATAC Reagent Kits and sequenced on Illumina NextSeq 500. The GC content of cell barcodes (read 2) are quite even.

As for read 3, based on the per base content, it suggests low complexity.

The detailed qc results are stored in the feature_barcoding_output.tsv.gz file. The matching_pos columns indicate the matched positions on reads, while the matching_description columns indicate mismatches in the format of substitutions:insertions:deletions.

```
$ gzip -dc feature_barcoding_output.tsv.gz | head

read1_seq       cell_barcode    cb_num_mismatches       read2_seq       feature_barcode␣
↪fb_num_mismatches
NTGTTGCTGGTTAGAA        CTGTTGCTGGTTAGAA        1       CAAAGGACGCGG    CD8Nb_PH-A_
↪CAAAGGACGCGG 0
NTCGACCGATTGCGTA        GTCGACCGATTGCGTA        1       CTAAAGACGCGG    CD8Nb_PH-B_
↪CTAAAGACGCGG 0
GCCGAACTGTTAGAAG        GCCGAACTGTTAGAAG        0       CAAAGGACGCGG    CD8Nb_PH-A_
↪CAAAGGACGCGG 0
TGAGCGCACACCTTGA        TGAGCGCACACCTTGA        0       CAAAGGACGCGG    CD8Nb_PH-A_
↪CAAAGGACGCGG 0
AATTCTGCTTGGCTGC        AATTCTGCTTGGCTGC        0       CAAAGGACGCGG    CD8Nb_PH-A_
↪CAAAGGACGCGG 0
GGAATGGTGACCGTGC        GGAATGGTGACCGTGC        0       CAAAGGACGCGG    CD8Nb_PH-A_
↪CAAAGGACGCGG 0
AGGAATTGATTCGCCT        AGGAATTGATTCGCCT        0       CAAAGGACGCGG    CD8Nb_PH-A_
↪CAAAGGACGCGG 0
CCAAGTTGATAATAGG        CCAAGTTGATAATAGG        0       CTAAAGACGCGG    CD8Nb_PH-B_
↪CTAAAGACGCGG 0
CCGCAAGTGAATCCAC        CCGCAAGTGAATCCAC        0       CAAAGGACGCGG    CD8Nb_PH-A_
↪CAAAGGACGCGG 0
```

R1

```
$ fba qc \
    -1 SRR12588752_2_trimmed.fq.gz \
    -2 SRR12588752_1_trimmed.fq.gz \
    -w barcodes.tsv \
    -f feature_barcodes_R1_truncated.txt \
    -cb_rc \
    -n 10000
```

For read 1, the per base content plot suggests low complexity, with a high proportion of constant bases at the beginning of the reads.

The detailed qc results are stored in the feature_barcoding_output.tsv.gz file. The matching_pos columns indicate the matched positions on reads, while the matching_description columns indicate mismatches in the format of substitutions:insertions:deletions.

```
$ gzip -dc feature_barcoding_output.tsv.gz | head

read1_seq       cell_barcode    cb_num_mismatches       read2_seq       feature_barcode␣
↪fb_num_mismatches
NCTCGGGACGTCTGGC        ACTCGGGACGTCTGGC        1       AGTGACCTGTGT    CD8Nb_PH-D_
↪AGTGACCTGTGT 0
NCTAAGACTTTATGGC        GCTAAGACTTTATGGC        1       AGTGACCTGTGT    CD8Nb_PH-D_
```

(continues on next page)

(continued from previous page)

```
↪AGTGACCTGTGT 0
NACGGAAGATCGTAAC        CACGGAAGATCGTAAC        1       AGTGACCTGTGT    CD8Nb_PH-D_
↪AGTGACCTGTGT 0
NTGTTGTGAGTCCCGA        GTGTTGTGAGTCCCGA        1       AGTGACCTGTGT    CD8Nb_PH-D_
↪AGTGACCTGTGT 0
CCTCCTGCTATCAGGG        CCTCCTGCTATCAGGG        0       AGTGACCTGTGT    CD8Nb_PH-D_
↪AGTGACCTGTGT 0
GTTGATTCTCGAAGCA        GTTGATTCTCGAAGCA        0       AGTGACCTGTGT    CD8Nb_PH-D_
↪AGTGACCTGTGT 0
TGGTTAGACTCCGTAA        TGGTTAGACTCCGTAA        0       AGTGACCTGTGT    CD8Nb_PH-D_
↪AGTGACCTGTGT 0
GCCTCTTGACTGGGTC        GCCTCTTGACTGGGTC        0       CGTAACTTGTGT    CD8Nb_PH-C_
↪CGTAACTTGTGT 0
AGGTAGCGAGAGTAAT        AGGTAGCGAGAGTAAT        0       AGTGACCTGTGT    CD8Nb_PH-D_
↪AGTGACCTGTGT 0
```

Barcode extraction

R3

Search ranges are set to `0,16` on read 2 and `0,12` on read 3. One mismatch for cell and feature barcodes (`-cb_m`, `-cf_m`) are allowed. Use `-cb_rc` to reverse-complement cell barcode sequences in the output if needed.

```
$ fba extract \
    -1 SRR12588752_2_trimmed.fq.gz \
    -2 SRR12588752_3_trimmed.fq.gz \
    -w barcodes.tsv \
    -f feature_barcodes_R3_truncated.txt \
    -o feature_barcoding_output_R3.tsv.gz \
    -r1_c 0,16 \
    -r2_c 0,12 \
    -cb_m 1 \
    -fb_m 1 \
    -cb_rc
```

Preview of result.

```
$ gzip -dc feature_barcoding_output_R3.tsv.gz | head

read1_seq       cell_barcode    cb_num_mismatches       read2_seq       feature_barcode␣
↪fb_num_mismatches
NTGTTGCTGGTTAGAA        CTGTTGCTGGTTAGAA        1       CAAAGGACGCGG    CD8Nb_PH-A_
↪CAAAGGACGCGG 0
NTCGACCGATTGCGTA        GTCGACCGATTGCGTA        1       CTAAAGACGCGG    CD8Nb_PH-B_
↪CTAAAGACGCGG 0
GCCGAACTGTTAGAAG        GCCGAACTGTTAGAAG        0       CAAAGGACGCGG    CD8Nb_PH-A_
↪CAAAGGACGCGG 0
```

(continues on next page)

(continued from previous page)

```
TGAGCGCACACCTTGA        TGAGCGCACACCTTGA        0       CAAAGGACGCGG    CD8Nb_PH-A_
↪CAAAGGACGCGG 0
AATTCTGCTTGGCTGC        AATTCTGCTTGGCTGC        0       CAAAGGACGCGG    CD8Nb_PH-A_
↪CAAAGGACGCGG 0
GGAATGGTGACCGTGC        GGAATGGTGACCGTGC        0       CAAAGGACGCGG    CD8Nb_PH-A_
↪CAAAGGACGCGG 0
AGGAATTGATTCGCCT        AGGAATTGATTCGCCT        0       CAAAGGACGCGG    CD8Nb_PH-A_
↪CAAAGGACGCGG 0
CCAAGTTGATAATAGG        CCAAGTTGATAATAGG        0       CTAAAGACGCGG    CD8Nb_PH-B_
↪CTAAAGACGCGG 0
CCGCAAGTGAATCCAC        CCGCAAGTGAATCCAC        0       CAAAGGACGCGG    CD8Nb_PH-A_
↪CAAAGGACGCGG 0
```

Result summary.

10,543,901 out of 51,140,637 read pairs have valid cell and feature barcodes.

```
2022-03-13 00:13:02,564 - fba.__main__ - INFO - fba version: 0.0.12
2022-03-13 00:13:02,564 - fba.__main__ - INFO - Initiating logging ...
2022-03-13 00:13:02,564 - fba.__main__ - INFO - Python version: 3.10
2022-03-13 00:13:02,564 - fba.__main__ - INFO - Using extract subcommand ...
2022-03-13 00:13:02,589 - fba.levenshtein - INFO - Number of reference cell barcodes: 8,
↪366
2022-03-13 00:13:02,590 - fba.levenshtein - INFO - Number of reference feature barcodes:␣
↪2
2022-03-13 00:13:02,590 - fba.levenshtein - INFO - Read 1 coordinates to search: [0, 16)
2022-03-13 00:13:02,590 - fba.levenshtein - INFO - Read 2 coordinates to search: [0, 12)
2022-03-13 00:13:02,590 - fba.levenshtein - INFO - Cell barcode maximum number of␣
↪mismatches: 1
2022-03-13 00:13:02,590 - fba.levenshtein - INFO - Feature barcode maximum number of␣
↪mismatches: 1
2022-03-13 00:13:02,590 - fba.levenshtein - INFO - Read 1 maximum number of N allowed: 3
2022-03-13 00:13:02,590 - fba.levenshtein - INFO - Read 2 maximum number of N allowed: 3
2022-03-13 00:13:02,809 - fba.levenshtein - INFO - Matching ...
2022-03-13 00:16:00,978 - fba.levenshtein - INFO - Read pairs processed: 10,000,000
2022-03-13 00:18:58,488 - fba.levenshtein - INFO - Read pairs processed: 20,000,000
2022-03-13 00:21:55,956 - fba.levenshtein - INFO - Read pairs processed: 30,000,000
2022-03-13 00:24:53,698 - fba.levenshtein - INFO - Read pairs processed: 40,000,000
2022-03-13 00:27:51,819 - fba.levenshtein - INFO - Read pairs processed: 50,000,000
2022-03-13 00:28:12,045 - fba.levenshtein - INFO - Number of read pairs processed: 51,
↪140,637
2022-03-13 00:28:12,045 - fba.levenshtein - INFO - Number of read pairs w/ valid␣
↪barcodes: 10,543,901
2022-03-13 00:28:12,060 - fba.__main__ - INFO - Done.
```

R1

Search ranges are set to `0,16` on read 2 and `0,12` on read 1. One mismatch for cell and feature barcodes (`-cb_m`, `-cf_m`) are allowed. Use `-cb_rc` to reverse-complement cell barcode sequences in the output if needed.

```
$ fba extract \
    -1 SRR12588752_2_trimmed.fq.gz \
    -2 SRR12588752_1_trimmed.fq.gz \
    -w barcodes.tsv \
    -f feature_barcodes_R1_truncated.txt \
    -o feature_barcoding_output_R1.tsv.gz \
    -r1_c 0,16 \
    -r2_c 0,12 \
    -cb_m 1 \
    -fb_m 1 \
    -cb_rc
```

Preview of result.

```
$ gzip -dc feature_barcoding_output_R1.tsv.gz | head

read1_seq       cell_barcode    cb_num_mismatches       read2_seq       feature_barcode␣
↪fb_num_mismatches
NCTCGGGACGTCTGGC        ACTCGGGACGTCTGGC        1       AGTGACCTGTGT    CD8Nb_PH-D_
↪AGTGACCTGTGT 0
NCTAAGACTTTATGGC        GCTAAGACTTTATGGC        1       AGTGACCTGTGT    CD8Nb_PH-D_
↪AGTGACCTGTGT 0
NACGGAAGATCGTAAC        CACGGAAGATCGTAAC        1       AGTGACCTGTGT    CD8Nb_PH-D_
↪AGTGACCTGTGT 0
NTGTTGTGAGTCCCGA        GTGTTGTGAGTCCCGA        1       AGTGACCTGTGT    CD8Nb_PH-D_
↪AGTGACCTGTGT 0
CCTCCTGCTATCAGGG        CCTCCTGCTATCAGGG        0       AGTGACCTGTGT    CD8Nb_PH-D_
↪AGTGACCTGTGT 0
GTTGATTCTCGAAGCA        GTTGATTCTCGAAGCA        0       AGTGACCTGTGT    CD8Nb_PH-D_
↪AGTGACCTGTGT 0
TGGTTAGACTCCGTAA        TGGTTAGACTCCGTAA        0       AGTGACCTGTGT    CD8Nb_PH-D_
↪AGTGACCTGTGT 0
GCCTCTTGACTGGGTC        GCCTCTTGACTGGGTC        0       CGTAACTTGTGT    CD8Nb_PH-C_
↪CGTAACTTGTGT 0
AGGTAGCGAGAGTAAT        AGGTAGCGAGAGTAAT        0       AGTGACCTGTGT    CD8Nb_PH-D_
↪AGTGACCTGTGT 0
```

Result summary.

11,128,546 out of 25,988,762 read pairs have valid cell and feature barcodes.

```
2022-03-12 23:29:33,460 - fba.__main__ - INFO - fba version: 0.0.12
2022-03-12 23:29:33,460 - fba.__main__ - INFO - Initiating logging ...
2022-03-12 23:29:33,460 - fba.__main__ - INFO - Python version: 3.10
2022-03-12 23:29:33,460 - fba.__main__ - INFO - Using extract subcommand ...
2022-03-12 23:29:33,488 - fba.levenshtein - INFO - Number of reference cell barcodes: 8,
↪366
2022-03-12 23:29:33,488 - fba.levenshtein - INFO - Number of reference feature barcodes:␣
↪2
```

(continues on next page)

(continued from previous page)

```
2022-03-12 23:29:33,488 - fba.levenshtein - INFO - Read 1 coordinates to search: [0, 16)
2022-03-12 23:29:33,488 - fba.levenshtein - INFO - Read 2 coordinates to search: [0, 12)
2022-03-12 23:29:33,488 - fba.levenshtein - INFO - Cell barcode maximum number of␣
↪mismatches: 1
2022-03-12 23:29:33,488 - fba.levenshtein - INFO - Feature barcode maximum number of␣
↪mismatches: 1
2022-03-12 23:29:33,488 - fba.levenshtein - INFO - Read 1 maximum number of N allowed: 3
2022-03-12 23:29:33,488 - fba.levenshtein - INFO - Read 2 maximum number of N allowed: 3
2022-03-12 23:29:33,707 - fba.levenshtein - INFO - Matching ...
2022-03-12 23:33:10,471 - fba.levenshtein - INFO - Read pairs processed: 10,000,000
2022-03-12 23:36:47,019 - fba.levenshtein - INFO - Read pairs processed: 20,000,000
2022-03-12 23:38:56,544 - fba.levenshtein - INFO - Number of read pairs processed: 25,
↪988,762
2022-03-12 23:38:56,544 - fba.levenshtein - INFO - Number of read pairs w/ valid␣
↪barcodes: 11,128,546
2022-03-12 23:38:56,558 - fba.__main__ - INFO - Done.
```

Matrix generation

Only fragments with correctly matched cell and feature barcodes are included. Use `-ul` to set the UMI length (default 12). Setting to `0` means no UMIs and read counts are summarized instead. Use `-cb_rc` to reverse-complement cell barcode sequences in the output matrix if needed. The generated feature count matrix can be easily imported into well-established single cell analysis packages such as Seurat and Scanpy.

```
$ fba count \
    -i feature_barcoding_output_R1.tsv.gz \
    -i feature_barcoding_output_R3.tsv.gz \
    -o matrix_featurecount.csv.gz \
    -ul 0
```

Result summary.

39.9 % (21,672,447 out of 54,274,791) of total read pairs have valid cell and feature barcodes. The median number of reads per cell for this phage-derived tag library is 2,261.0.

```
2022-03-13 00:36:01,502 - fba.__main__ - INFO - fba version: 0.0.12
2022-03-13 00:36:01,502 - fba.__main__ - INFO - Initiating logging ...
2022-03-13 00:36:01,502 - fba.__main__ - INFO - Python version: 3.9
2022-03-13 00:36:01,502 - fba.__main__ - INFO - Using count subcommand ...
2022-03-13 00:36:02,348 - fba.count - INFO - UMI-tools version: 1.1.1
2022-03-13 00:36:02,348 - fba.count - INFO - UMI length set to 0, ignoring UMI␣
↪information. Skipping arguments: "-us/--umi_start".
2022-03-13 00:36:02,348 - fba.count - INFO - Header: read1_seq cell_barcode cb_num_
↪mismatches read2_seq feature_barcode fb_num_mismatches
2022-03-13 00:36:20,914 - fba.count - INFO - Number of read pairs processed: 21,672,447
2022-03-13 00:36:20,917 - fba.count - INFO - Number of cell barcodes detected: 8,366
2022-03-13 00:36:20,917 - fba.count - INFO - Number of features detected: 4
2022-03-13 00:36:20,917 - fba.count - INFO - Counting ...
```

(continues on next page)

(continued from previous page)

```
2022-03-13 00:36:21,009 - fba.count - INFO - Total reads: 21,672,447
2022-03-13 00:36:21,016 - fba.count - INFO - Median number of reads per cell: 2,261.0
2022-03-13 00:36:21,103 - fba.__main__ - INFO - Done.
```

Demultiplexing

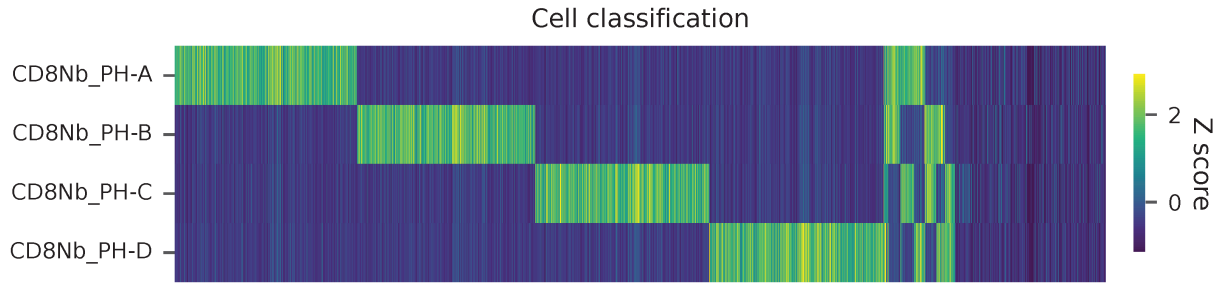
Negative binomial distribution

Cells are demultiplexed based on the feature count matrix using demultiplexing method 1 (set by `-dm`), which is implemented based on the method described by Stoeckius, M., et al. (2018) with some modifications. The output directory for demultiplexing is set by `--output_directory` (default `demultiplexed`). A cell identity matrix is generated, where 0 indicates negative and 1 indicates positive. To adjust the quantile threshold for demultiplexing, use `-q` (default `0.9999`). To generate visualization plots, set `-v`.

```
$ fba demultiplex \
    -i matrix_featurecount.csv.gz \
    -q 0.99 \
    -v
```

```
2022-03-13 00:47:41,569 - fba.__main__ - INFO - fba version: 0.0.12
2022-03-13 00:47:41,569 - fba.__main__ - INFO - Initiating logging ...
2022-03-13 00:47:41,569 - fba.__main__ - INFO - Python version: 3.10
2022-03-13 00:47:41,569 - fba.__main__ - INFO - Using demultiplex subcommand ...
2022-03-13 00:47:49,145 - fba.__main__ - INFO - Skipping arguments: "-p/--prob"
2022-03-13 00:47:49,145 - fba.demultiplex - INFO - Output directory: demultiplexed
2022-03-13 00:47:49,145 - fba.demultiplex - INFO - Demultiplexing method: 1
2022-03-13 00:47:49,146 - fba.demultiplex - INFO - UMI normalization method: clr
2022-03-13 00:47:49,146 - fba.demultiplex - INFO - Visualization: On
2022-03-13 00:47:49,146 - fba.demultiplex - INFO - Visualization method: tsne
2022-03-13 00:47:49,146 - fba.demultiplex - INFO - Loading feature count matrix: matrix_
↪featurecount.csv.gz ...
2022-03-13 00:47:49,324 - fba.demultiplex - INFO - Number of cells: 8,366
2022-03-13 00:47:49,324 - fba.demultiplex - INFO - Number of positive cells for a␣
↪feature to be included: 200
2022-03-13 00:47:49,327 - fba.demultiplex - INFO - Number of features: 4 / 4 (after␣
↪filtering / original in the matrix)
2022-03-13 00:47:49,327 - fba.demultiplex - INFO - Features: CD8Nb_PH-A CD8Nb_PH-B CD8Nb_
↪PH-C CD8Nb_PH-D
2022-03-13 00:47:49,327 - fba.demultiplex - INFO - Total UMIs/reads: 21,672,447 / 21,672,
↪447
2022-03-13 00:47:49,328 - fba.demultiplex - INFO - Median number of UMIs/reads per cell:␣
↪2,261.0 / 2,261.0
2022-03-13 00:47:49,328 - fba.demultiplex - INFO - Demultiplexing ...
2022-03-13 00:48:53,685 - fba.demultiplex - INFO - Generating heatmap ...
2022-03-13 00:48:59,759 - fba.demultiplex - INFO - Embedding ...
2022-03-13 00:49:20,128 - fba.__main__ - INFO - Done.
```

Heatmap of the relative abundance of features (phage-derived tags, PDTs) across all cells. Each column represents a single cell. This is a re-creation of Fig. 3b in Fiskin, E., et al. (2021).



Preview the demultiplexing result: the numbers of singlets, multiplets and negatives are 6,373 (76.2%), 639 (7.6%), and 1,354 (16.2%), respectively.

```
In [1]: import pandas as pd

In [2]: m = pd.read_csv("demultiplexed/matrix_cell_identity.csv.gz", index_col=0)

In [3]: m.loc[:, m.sum(axis=0) == 1].sum(axis=1)
Out[3]:
CD8Nb_PH-A    1640
CD8Nb_PH-B    1601
CD8Nb_PH-C    1564
CD8Nb_PH-D    1568
dtype: int64

In [4]: sum(m.sum(axis=0) == 1)
Out[4]: 6373

In [5]: sum(m.sum(axis=0) > 1)
Out[5]: 639

In [6]: sum(m.sum(axis=0) == 0)
Out[6]: 1354

In [7]: m.shape
Out[7]: (4, 8366)
```

t-SNE embedding of cells based on the abundance of features (phage-derived tags, no transcriptome information used). Colors indicate the hashtag status for each cell, as called by FBA. This is a re-creation of Fig. 3d in Fiskin, E., et al. (2021).

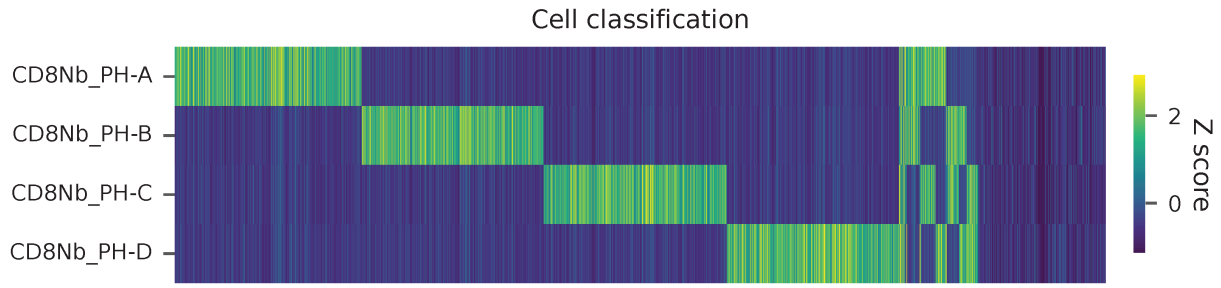
Gaussian mixture model

The implementation of demultiplexing method `2` (set by `-dm`) is inspired by the method described on the 10x Genomics' website. To set the probability threshold for demultiplexing, use `-p` (default `0.9`). To generate visualization plots, set `-v`.

```
$ fba demultiplex \
    -i matrix_featurecount.csv.gz \
    -dm 2 \
    -v
```

```
2022-03-13 11:27:47,035 - fba.__main__ - INFO - fba version: 0.0.12
2022-03-13 11:27:47,035 - fba.__main__ - INFO - Initiating logging ...
2022-03-13 11:27:47,035 - fba.__main__ - INFO - Python version: 3.9
2022-03-13 11:27:47,035 - fba.__main__ - INFO - Using demultiplex subcommand ...
2022-03-13 11:27:49,515 - fba.__main__ - INFO - Skipping arguments: "-q/--quantile", "-
↪cm/--clustering_method"
2022-03-13 11:27:49,515 - fba.demultiplex - INFO - Output directory: demultiplexed
2022-03-13 11:27:49,515 - fba.demultiplex - INFO - Demultiplexing method: 2
2022-03-13 11:27:49,515 - fba.demultiplex - INFO - UMI normalization method: clr
2022-03-13 11:27:49,515 - fba.demultiplex - INFO - Visualization: On
2022-03-13 11:27:49,515 - fba.demultiplex - INFO - Visualization method: tsne
2022-03-13 11:27:49,515 - fba.demultiplex - INFO - Loading feature count matrix: matrix_
↪featurecount.csv.gz ...
2022-03-13 11:27:49,595 - fba.demultiplex - INFO - Number of cells: 8,366
2022-03-13 11:27:49,595 - fba.demultiplex - INFO - Number of positive cells for a␣
↪feature to be included: 200
2022-03-13 11:27:49,607 - fba.demultiplex - INFO - Number of features: 4 / 4 (after␣
↪filtering / original in the matrix)
2022-03-13 11:27:49,607 - fba.demultiplex - INFO - Features: CD8Nb_PH-A CD8Nb_PH-B CD8Nb_
↪PH-C CD8Nb_PH-D
2022-03-13 11:27:49,607 - fba.demultiplex - INFO - Total UMIs: 21,672,447 / 21,672,447
2022-03-13 11:27:49,614 - fba.demultiplex - INFO - Median number of UMIs/reads per cell:␣
↪2,261.0 / 2,261.0
2022-03-13 11:27:49,614 - fba.demultiplex - INFO - Demultiplexing ...
2022-03-13 11:27:51,392 - fba.demultiplex - INFO - Generating heatmap ...
2022-03-13 11:27:53,158 - fba.demultiplex - INFO - Embedding ...
2022-03-13 11:28:08,072 - fba.__main__ - INFO - Done.
```

Heatmap of the relative abundance of features (phage-derived tags, PDTs) across all cells. Each column represents a single cell. This is a re-creation of Fig. 3b in Fiskin, E., et al. (2021).



Preview the demultiplexing result: the numbers of singlets, multiplets and negatives are 6,510 (77.8%), 709 (8.5%),

and 1,147 (13.7%), respectively.

```
In [1]: import pandas as pd

In [2]: m = pd.read_csv("demultiplexed/matrix_cell_identity.csv.gz", index_col=0)

In [3]: m.loc[:, m.sum(axis=0) == 1].sum(axis=1)
Out[3]:
CD8Nb_PH-A    1680
CD8Nb_PH-B    1637
CD8Nb_PH-C    1646
CD8Nb_PH-D    1547
dtype: int64

In [4]: sum(m.sum(axis=0) == 1)
Out[4]: 6510

In [5]: sum(m.sum(axis=0) > 1)
Out[5]: 709

In [6]: sum(m.sum(axis=0) == 0)
Out[6]: 1147

In [7]: m.shape
Out[7]: (4, 8366)
```

t-SNE embedding of cells based on the abundance of features (phage-derived tags, no transcriptome information used). Colors indicate the hashtag status for each cell, as called by FBA. This is a re-creation of Fig. 3d in Fiskin, E., et al. (2021).

Read distribution and model fitting threshold:

- *PHAGE-ATAC; Anti-CD8 Phage Hashing Single-cell ATAC-seq Using CD8 T Cells from Four Human Donors*

3.5 CellPlex

3.5.1 10k 1:1 Mixture of Raji and Jurkat Cells Multiplexed

Dataset: 10k 1:1 Mixture of Raji and Jurkat Cells Multiplexed, 2 CMOs

The detailed description of this dataset can be found here.

Preparation

Download fastq files.

```
$ wget https://s3-us-west-2.amazonaws.com/10x.files/samples/cell-exp/6.0.0/SC3_v3_
↪NextGem_DI_CellPlex_Jurkat_Raji_10K_Multiplex/SC3_v3_NextGem_DI_CellPlex_Jurkat_Raji_
↪10K_Multiplex_fastqs.tar

$ tar xvf SC3_v3_NextGem_DI_CellPlex_Jurkat_Raji_10K_Multiplex_fastqs.tar
```

Combine reads of different lanes.

```
$ cat SC3_v3_NextGem_DI_CellPlex_Jurkat_Raji_10K/SC3_v3_NextGem_DI_CellPlex_Jurkat_Raji_
↪10K_?_multiplexing_capture/SC3_v3_NextGem_DI_CellPlex_Jurkat_Raji_10K_?_multiplexing_
↪capture_S1_L00?_R1_001.fastq.gz > SC3_v3_NextGem_DI_CellPlex_Jurkat_Raji_10K_1_
↪multiplexing_capture_S1_combined_R1_001.fastq.gz

$ cat SC3_v3_NextGem_DI_CellPlex_Jurkat_Raji_10K/SC3_v3_NextGem_DI_CellPlex_Jurkat_Raji_
↪10K_?_multiplexing_capture/SC3_v3_NextGem_DI_CellPlex_Jurkat_Raji_10K_?_multiplexing_
↪capture_S1_L00?_R2_001.fastq.gz > SC3_v3_NextGem_DI_CellPlex_Jurkat_Raji_10K_1_
↪multiplexing_capture_S1_combined_R2_001.fastq.gz
```

Download cell barcode info. These are the cell-associated barcodes in this single cell RNA-Seq library (determined by the number of transcriptomic UMIs captured per barcode).

```
$ wget https://cf.10xgenomics.com/samples/cell-exp/6.0.0/SC3_v3_NextGem_DI_CellPlex_
↪Jurkat_Raji_10K_Multiplex/SC3_v3_NextGem_DI_CellPlex_Jurkat_Raji_10K_Multiplex_
↪multiplexing_analysis_assignment_confidence_table.csv

$ cut -d ',' -f4 SC3_v3_NextGem_DI_CellPlex_Jurkat_Raji_10K_Multiplex_multiplexing_
↪analysis_assignment_confidence_table.csv | sed 's/-1//g' > cell_barcodes.txt
```

Inspect cell barcodes.

```
$ head cell_barcodes.txt

AAACCCAAGAGTGTGC
AAACCCAAGATGCTTC
AAACCCAAGGTACAGC
AAACCCAAGTAGCTCT
AAACCCACACGGATCC
```

(continues on next page)

(continued from previous page)

```
AAACCCACAGACCCGT
AAACCCACATTCCTAT
AAACCCAGTACTCCGG
AAACCCATCATGCGGC
AAACCCATCGACGCTG
```

Prepare feature barcodes (cell multiplexing oligos, CMOs).

```
$ wget https://cf.10xgenomics.com/samples/cell-exp/6.0.0/SC3_v3_NextGem_DI_CellPlex_
↪Jurkat_Raji_10K_Multiplex/SC3_v3_NextGem_DI_CellPlex_Jurkat_Raji_10K_Multiplex_count_
↪feature_reference.csv

$ cut -d',' -f1,5 SC3_v3_NextGem_DI_CellPlex_Jurkat_Raji_10K_Multiplex_count_feature_
↪reference.csv | sed 's/,/\t/g' | grep ^C > feature_barcodes.txt
```

Inspect feature barcodes.

```
$ cat feature_barcodes.tsv

CMO301  ATGAGGAATTCCTGC
CMO302  CATGCCAATAGAGCG
CMO303  CCGTCGTCCAAGCAT
CMO304  AACGTTAATCACTCA
CMO305  CGCGATATGGTCGGA
CMO306  AAGATGAGGTCTGTG
CMO307  AAGCTCGTTGGAAGA
CMO308  CGGATTCCACATCAT
CMO309  GTTGATCTATAACAG
CMO310  GCAGGAGGTATCAAT
CMO311  GAATCGTGATTCTTC
CMO312  ACATGGTCAACGCTG
```

QC

The first `100,000` read pairs are sampled (default, set by `-n`) for quality control. The `-t` option can be used to set the number of threads. By default, diagnostic results and plots are generated in the `qc` directory (set by `--output_directory`), and the full length of read 1 and read 2 are searched against reference cell and feature barcodes, respectively. The per base content of both read pairs and the distribution of matched barcode positions are summarized. Use `-r1_c` and/or `-r2_c` to limit the search range, and `-cb_n` and/or `-fb_n` to set the mismatch tolerance for cell and/or feature barcode matching (default 3).

```
$ fba qc \
    -1 ../SC3_v3_NextGem_DI_CellPlex_Jurkat_Raji_10K_1_multiplexing_capture_S1_combined_
↪R1_001.fastq.gz \
    -2 ../SC3_v3_NextGem_DI_CellPlex_Jurkat_Raji_10K_1_multiplexing_capture_S1_combined_
↪R2_001.fastq.gz \
    -w cell_barcodes.txt \
    -f feature_barcodes.tsv \
```

(continues on next page)

(continued from previous page)

```
    --output_directory qc \
    -r1_c 0,16
```

This library was constructed using the Chromium Next GEM Single Cell 3 Reagent Kits v3.1 (Dual Index) with Feature Barcode technology for Cell Multiplexing and sequenced on an Illumina NovaSeq 6000. The first 16 bases of read 1 represent cell barcodes, and the following 12 bases represent UMIs. The base content plot indicates that the GC content of cell barcodes is evenly distributed. However, there is a slight T-enrichment in the UMIs.

In reference to read 2, the per base content indicates that bases 0-14 represent feature barcodes (CMOs, at 15 bp). Bases 15-36 are constant and comprise Capture Sequence 2, which can be read with high accuracy (GCTCACCTATTAGCGGCTAAGG). The succeeding 12 bases represent UMIs, followed by 16-base cell barcodes. Bases 37-54 correspond to the reverse complement of read 1. While the CellPlex library is relatively small, read 2 has also sequenced through a portion of the Nextera Read 1 sequencing primer, which is constant and encompasses bases 55-79. In actuality, read 1 is unnecessary since read 2 contains all the necessary information for demultiplexing, including cell barcodes, UMIs, and CMOs. Theoretically, we can further enhance the accuracy of demultiplexing by utilizing the cell barcodes and UMIs on both reads to account for PCR and sequencing errors.

Most of the reads have the correct structure.

The detailed `qc` results are stored in `feature_barcoding_output.tsv.gz` file. `matching_pos` columns indicate the matched positions on reads. `matching_description` columns indicate mismatches in substitutions:insertions:deletions format.

```
$ gzip -dc qc/feature_barcoding_output.tsv.gz | head

read1_seq       cell_barcode    cb_matching_pos cb_matching_description read2_seq      ␣
↪feature_barcode fb_matching_pos fb_matching_description
AAGCGTTAGAGTCTTTggtatttttatt    AAGCGTTAGAAGCCTG        0:15    2:0:1   ␣
↪ATGAGGAATTCCTGCGCTCACCTATTAGCGGCTAAGGAATAAAACTACCAAAGACTCTAACGCTTCTGTCTCTTATACACATCTGACGCT␣
↪     CMO301_ATGAGGAATTCCTGC     0:15    0:0:0
ATCTCTACAACCCACGctttattgttta    ATCTCTAGTACCCACG        0:16    2:0:0   ␣
↪ATGAGGAATTCCTGCGCTCACCTATTAGCGGCTAAGGTAAACAATAAAGCGTGGGTTGTAGAGATCTGTCTCTTATACACATCTGACGCT␣
↪     CMO301_ATGAGGAATTCCTGC     0:15    0:0:0
TGCTTCGAGCATGATGttctgagccggt    TGCTTCGAGATTGAGT        0:15    2:0:1   ␣
↪CATGCCAATAGAGCGGCTCACCTATTAGCGGCTAAGGACCGGCTCAGAACATCATGCTCGAAGCACTGTCTCTTATACACATCTGACGCT␣
↪     CMO302_CATGCCAATAGAGCG     0:15    0:0:0
CGGGACTGTAGTATAGacctaatttttcc    CGGGACTGTAAGCAAT        0:14    1:0:2   ␣
↪CATGCCAATAGAGCGGCTCACCTATTAGCGGCTAAGGGGAAAATTAGGTCTATACTACAGTCCCGCTGTCTCTTATACACATCTGACGCT␣
↪     CMO302_CATGCCAATAGAGCG     0:15    0:0:0
TCACTCGCAATTTCGGaacatggacatc    TCACTCGCACCATTCC        0:14    1:0:2   ␣
↪ATGAGGAATTCCTGCGCTCACCTATTAGCGGCTAAGGGAAGTCCATGTTCCGAAATTGCGAGTGACTGTCTCTTATACACATCTGACGCT␣
↪     CMO301_ATGAGGAATTCCTGC     0:15    0:0:0
AGTTAGCAGACGTAGTgccttaatttgg    AGTTAGCAGAGCATTA        0:14    1:0:2   ␣
↪CATGCCAATAGAGCGGCTCACCTATTAGCGGCTAAGGCCAAATTAAGGCACTACGTCTGCTAACTCTGTCTCTTATACACATCTGACGCT␣
```

(continues on next page)

(continued from previous page)

```
↪      CMO302_CATGCCAATAGAGCG      0:15    0:0:0
GTCCCATTCTAAACGTtgagtacgagcg    CATCCCATCCTAAACG        0:15    2:0:1   ␣
↪CATGCCAATAGAGCGGCTCACCTATTAGCGGCTAAGGCGCTCGTACTCAACGTTTAGAATGGGACCTGTCTCTTATACACATCTGACGCT␣
↪      CMO302_CATGCCAATAGAGCG      0:15    0:0:0
CAGAGCCCAATAGGGCcaccctcttaac    CAGAGCCGTATAGGGC        0:16    2:0:0   ␣
↪ATGAGGAATTCCTGCGCTCACCTATTAGCGGCTAAGGGTTAAGAGGGTGGCCCTATTGGGCTCTGCTGTCTCTTATACACATCTGACGCT␣
↪      CMO301_ATGAGGAATTCCTGC      0:15    0:0:0
AACCCAATCAGTTGTAggatattcacct    AACCCAACAGCATTGT        0:15    0:1:2   ␣
↪ATGAGGAATTCCTGCGCTCACCTATTAGCGGCTAAGGAGGTGAATATCCTACAACTGATTGGGTTCTGTCTCTTATACACATCTGACGCT␣
↪      CMO301_ATGAGGAATTCCTGC      0:15    0:0:0
```

Barcode extraction

Both cell and feature barcodes (CMOs) have the same length of 16 and 15, respectively. The starting and ending positions of these barcodes are uniformly distributed according to the `qc` results. The read 1 search range is set from 0 to 16, while the read 2 search range is set from 0 to 15. There is a tolerance of two mismatches allowed for the cell barcode and one for the feature barcode (set by `-cb_m` and `-cf_m`). The default setting allows for three ambiguous nucleotides (Ns) in both read 1 and read 2 (set by `-cb_n` and `-cf_n`).

```
$ fba extract \
    -1 ../SC3_v3_NextGem_DI_CellPlex_Jurkat_Raji_10K_1_multiplexing_capture_S1_combined_
↪R1_001.fastq.gz \
    -2 ../SC3_v3_NextGem_DI_CellPlex_Jurkat_Raji_10K_1_multiplexing_capture_S1_combined_
↪R2_001.fastq.gz \
    -w cell_barcodes.txt \
    -f feature_barcodes.tsv \
    -o feature_barcoding_output.tsv.gz \
    -r1_c 0,16 \
    -r2_c 0,15 \
    -cb_m 2 \
    -fb_m 1 \
    -cb_n 3 \
    -fb_n 3
```

Preview of result.

```
$ gzip -dc feature_barcoding_output.tsv.gz | head

read1_seq       cell_barcode    cb_num_mismatches       read2_seq       feature_barcode␣
↪fb_num_mismatches
AAGCGTTAGAGTCTTTggtatttttatt    AAGCGTTTCAGTCTTT        2       ␣
↪ATGAGGAATTCCTGCgctcacctattagcggctaaggaataaaactaccaaagactctaacgcttctgtctcttatacacatctgacgct␣
↪      CMO301_ATGAGGAATTCCTGC  0
ATCTCTACAACCCACGctttattgttta    ATCTCTAGTACCCACG        2       ␣
↪ATGAGGAATTCCTGCgctcacctattagcggctaaggtaaacaataaagcgtgggttgtagagatctgtctcttatacacatctgacgct␣
↪      CMO301_ATGAGGAATTCCTGC  0
TCACTCGCAATTTCGGaacatggacatc    TCACTCGCAGTTTCAG        2       ␣
↪ATGAGGAATTCCTGCgctcacctattagcggctaagggaagtccatgttccgaaattgcgagtgactgtctcttatacacatctgacgct␣
```

(continues on next page)

(continued from previous page)

```
↪     CMO301_ATGAGGAATTCCTGC  0
CAGAGCCCAATAGGGCcaccctcttaac    CAGAGCCGTATAGGGC        2       ␣
↪ATGAGGAATTCCTGCgctcacctattagcggctaagggttaagagggtggccctattgggctctgctgtctcttatacacatctgacgct␣
↪     CMO301_ATGAGGAATTCCTGC  0
TGAGGGACATGCCAATcattttgaattt    TGAGGGAGTTGCCAAT        2       ␣
↪ATGAGGAATTCCTGCgctcacctattagcggctaaggaaattcaaaatgattggcatgtccctcactgtctcttatacacatctgacgct␣
↪     CMO301_ATGAGGAATTCCTGC  0
CAGGGCTGTGCATGCCgcttaaacagca    CAGGGCTCAGCATGCC        2       ␣
↪ATGAGGAATTCCTGCgctcacctattagcggctaaggtgctgtttaagcggcatgcacagccctgctgtctcttatacacatctgacgct␣
↪     CMO301_ATGAGGAATTCCTGC  0
TCGGGTGTCCGACATGactctagtacat    TCGGGTGAGCGACATG        2       ␣
↪ATGAGGAATTCCTGCgctcacctattagcggctaaggatgtactagagtcatgtcggacacccgactgtctcttatacacatctgacgct␣
↪     CMO301_ATGAGGAATTCCTGC  0
TCGAAGTGTCAAAGTAgtaaaaggtacc    TCGAAGTCACAAAGTA        2       ␣
↪ATGAGGAATTCCTGCgctcacctattagcggctaagggagaagtcccaatactttgacgctcacctattagcggctaaggggtacctttt␣
↪     CMO301_ATGAGGAATTCCTGC  0
GTCATCCAGTGAGAGGtcagtgacacct    GTCATCCAGAGAGCGG        2       ␣
↪ATGAGGAATTCCTGCgctcacctattagcggctaaggaggtgtcactgacctctcactggatgacctgtctcttatacacatctgacgct␣
↪     CMO301_ATGAGGAATTCCTGC  0
```

Result summary.

63.98% (138,246,914 out of 216,070,514) of total read pairs have valid cell and feature barcodes. Majority of the fragments in this library have the correct structure.

```
2021-09-30 02:00:26,049 - fba.__main__ - INFO - fba version: 0.0.x
2021-09-30 02:00:26,049 - fba.__main__ - INFO - Initiating logging ...
2021-09-30 02:00:26,049 - fba.__main__ - INFO - Python version: 3.7
2021-09-30 02:00:26,049 - fba.__main__ - INFO - Using extract subcommand ...
2021-09-30 02:00:26,075 - fba.levenshtein - INFO - Number of reference cell barcodes: 13,
↪615
2021-09-30 02:00:26,075 - fba.levenshtein - INFO - Number of reference feature barcodes:␣
↪12
2021-09-30 02:00:26,075 - fba.levenshtein - INFO - Read 1 coordinates to search: [0, 16)
2021-09-30 02:00:26,075 - fba.levenshtein - INFO - Read 2 coordinates to search: [0, 15)
2021-09-30 02:00:26,075 - fba.levenshtein - INFO - Cell barcode maximum number of␣
↪mismatches: 2
2021-09-30 02:00:26,075 - fba.levenshtein - INFO - Feature barcode maximum number of␣
↪mismatches: 1
2021-09-30 02:00:26,075 - fba.levenshtein - INFO - Read 1 maximum number of N allowed: 3
2021-09-30 02:00:26,075 - fba.levenshtein - INFO - Read 2 maximum number of N allowed: 3
2021-09-30 02:00:29,258 - fba.levenshtein - INFO - Matching ...
2021-09-30 02:16:48,398 - fba.levenshtein - INFO - Read pairs processed: 10,000,000
2021-09-30 02:33:07,679 - fba.levenshtein - INFO - Read pairs processed: 20,000,000
2021-09-30 02:49:32,978 - fba.levenshtein - INFO - Read pairs processed: 30,000,000
2021-09-30 03:05:53,492 - fba.levenshtein - INFO - Read pairs processed: 40,000,000
2021-09-30 03:22:08,512 - fba.levenshtein - INFO - Read pairs processed: 50,000,000
2021-09-30 03:38:20,583 - fba.levenshtein - INFO - Read pairs processed: 60,000,000
2021-09-30 03:54:33,108 - fba.levenshtein - INFO - Read pairs processed: 70,000,000
2021-09-30 04:10:45,824 - fba.levenshtein - INFO - Read pairs processed: 80,000,000
2021-09-30 04:26:57,385 - fba.levenshtein - INFO - Read pairs processed: 90,000,000
2021-09-30 04:43:13,387 - fba.levenshtein - INFO - Read pairs processed: 100,000,000
```

(continues on next page)

(continued from previous page)

```
2021-09-30 04:59:37,730 - fba.levenshtein - INFO - Read pairs processed: 110,000,000
2021-09-30 05:15:57,226 - fba.levenshtein - INFO - Read pairs processed: 120,000,000
2021-09-30 05:32:16,897 - fba.levenshtein - INFO - Read pairs processed: 130,000,000
2021-09-30 05:48:34,670 - fba.levenshtein - INFO - Read pairs processed: 140,000,000
2021-09-30 06:04:55,040 - fba.levenshtein - INFO - Read pairs processed: 150,000,000
2021-09-30 06:21:12,282 - fba.levenshtein - INFO - Read pairs processed: 160,000,000
2021-09-30 06:37:28,322 - fba.levenshtein - INFO - Read pairs processed: 170,000,000
2021-09-30 06:53:47,355 - fba.levenshtein - INFO - Read pairs processed: 180,000,000
2021-09-30 07:10:10,017 - fba.levenshtein - INFO - Read pairs processed: 190,000,000
2021-09-30 07:26:29,370 - fba.levenshtein - INFO - Read pairs processed: 200,000,000
2021-09-30 07:42:51,320 - fba.levenshtein - INFO - Read pairs processed: 210,000,000
2021-09-30 07:52:47,851 - fba.levenshtein - INFO - Number of read pairs processed: 216,
↪070,514
2021-09-30 07:52:47,851 - fba.levenshtein - INFO - Number of read pairs w/ valid␣
↪barcodes: 138,246,914
2021-09-30 07:52:47,970 - fba.__main__ - INFO - Done.
```

Matrix generation

Only fragments with correctly matched cell and feature barcodes are included, while fragments with UMI lengths less than the specified value are discarded. UMI removal is performed using UMI-tools (Smith, T., et al. 2017. Genome Res. 27, 491–499.), with the starting position on read 1 set by `-us` (default `16`) and the length set by `-ul` (default `12`). The UMI deduplication method can be set using `-ud` (default `directional`), and the UMI deduplication mismatch threshold can be specified using `-um` (default `1`).

The generated feature count matrix can be easily imported into well-established single cell analysis packages such as Seruat and Scanpy.

```
$ fba count \
    -i feature_barcoding_output.tsv.gz \
    -o matrix_featurecount.csv.gz \
    -us 16 \
    -ul 12 \
    -um 1 \
    -ud directional
```

Result summary.

88.00% (121,661,177 out of 138,246,914) of read pairs with valid cell and feature barcodes are unique fragments. 56.31% (121,661,177 out of 216,070,514) of total sequenced read pairs contribute to the final matrix.

```
2021-09-30 07:52:48,076 - fba.__main__ - INFO - fba version: 0.0.x
2021-09-30 07:52:48,076 - fba.__main__ - INFO - Initiating logging ...
2021-09-30 07:52:48,076 - fba.__main__ - INFO - Python version: 3.7
2021-09-30 07:52:48,076 - fba.__main__ - INFO - Using count subcommand ...
2021-09-30 07:52:49,463 - fba.count - INFO - UMI-tools version: 1.1.1
2021-09-30 07:52:49,466 - fba.count - INFO - UMI starting position on read 1: 16
2021-09-30 07:52:49,466 - fba.count - INFO - UMI length: 12
```

(continues on next page)

(continued from previous page)

```
2021-09-30 07:52:49,467 - fba.count - INFO - UMI-tools deduplication threshold: 1
2021-09-30 07:52:49,467 - fba.count - INFO - UMI-tools deduplication method: directional
2021-09-30 07:52:49,467 - fba.count - INFO - Header line: read1_seq cell_barcode cb_num_
↪mismatches read2_seq feature_barcode fb_num_mismatches
2021-09-30 07:58:54,696 - fba.count - INFO - Number of lines processed: 138,246,914
2021-09-30 07:58:54,707 - fba.count - INFO - Number of cell barcodes detected: 13,612
2021-09-30 07:58:54,707 - fba.count - INFO - Number of features detected: 12
2021-09-30 18:31:30,172 - fba.count - INFO - Total UMIs after deduplication: 121,661,177
2021-09-30 18:31:30,208 - fba.count - INFO - Median number of UMIs per cell: 7,663.5
2021-09-30 18:31:30,457 - fba.__main__ - INFO - Done.
```

Demultiplexing

Inspect feature count matrix.

```
In [1]: import pandas as pd

In [2]: m = pd.read_csv("matrix_featurecount.csv.gz", index_col=0)

In [3]: m.sum(axis=1)
Out[3]:
CMO301_ATGAGGAATTCCTGC    81595732
CMO302_CATGCCAATAGAGCG    39999656
CMO303_CCGTCGTCCAAGCAT        1719
CMO304_AACGTTAATCACTCA         973
CMO305_CGCGATATGGTCGGA         167
CMO306_AAGATGAGGTCTGTG         563
CMO307_AAGCTCGTTGGAAGA         757
CMO308_CGGATTCCACATCAT       57738
CMO309_GTTGATCTATAACAG        2767
CMO310_GCAGGAGGTATCAAT         236
CMO311_GAATCGTGATTCTTC         166
CMO312_ACATGGTCAACGCTG         703
dtype: int64

In [4]: m = m.loc[["CMO301_ATGAGGAATTCCTGC", "CMO302_CATGCCAATAGAGCG"], :]

In [5]: m.to_csv(path_or_buf="matrix_featurecount_filtered.csv.gz", compression="infer")
```

CMO301_ATGAGGAATTCCTGC and CMO302_CATGCCAATAGAGCG have the highest UMI counts and were the CMOs used in this experiment."

Gaussian mixture model

Cells are demulitplexed based on the feature count matrix (CMO abundance). Demultiplexing method 2 (set by `-dm`) is inspired by the method described on 10x Genomics' website. A cell identity matrix is generated in the output directory: 0 means negative, 1 means positive. To set normalization method, use `-nm` (default `clr`). To set the probability threshold for demultiplexing, use `-p` (default `0.9`). To generate visualization plots, set `-v`. To choose visualization method, use `-vm` (default `tsne`).

```
$ fba demultiplex \
    -i matrix_featurecount_filtered.csv.gz \
    --output_directory demultiplexed \
    -dm 2 \
    -v \
    -vm umap
```

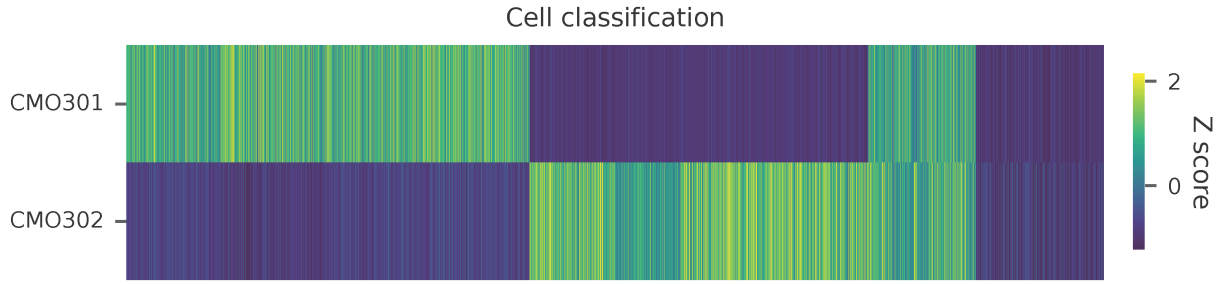
```
2021-10-01 23:07:30,925 - fba.__main__ - INFO - fba version: 0.0.x
2021-10-01 23:07:30,925 - fba.__main__ - INFO - Initiating logging ...
2021-10-01 23:07:30,925 - fba.__main__ - INFO - Python version: 3.7
2021-10-01 23:07:30,925 - fba.__main__ - INFO - Using demultiplex subcommand ...
2021-10-01 23:07:45,559 - fba.__main__ - INFO - Skipping arguments: "-q/--quantile", "-
↪cm/--clustering_method"
2021-10-01 23:07:45,560 - fba.demultiplex - INFO - Output directory: demultiplexed
2021-10-01 23:07:45,560 - fba.demultiplex - INFO - Demultiplexing method: 2
2021-10-01 23:07:45,560 - fba.demultiplex - INFO - UMI normalization method: clr
2021-10-01 23:07:45,560 - fba.demultiplex - INFO - Visualization: On
2021-10-01 23:07:45,560 - fba.demultiplex - INFO - Visualization method: umap
2021-10-01 23:07:45,560 - fba.demultiplex - INFO - Loading feature count matrix: matrix_
↪featurecount_filtered.csv.gz ...
2021-10-01 23:07:46,353 - fba.demultiplex - INFO - Number of cells: 13,612
2021-10-01 23:07:46,353 - fba.demultiplex - INFO - Number of positive cells for a␣
↪feature to be included: 200
2021-10-01 23:07:46,400 - fba.demultiplex - INFO - Number of features: 2 / 2 (after␣
↪filtering / original in the matrix)
2021-10-01 23:07:46,400 - fba.demultiplex - INFO - Features: CMO301 CMO302
2021-10-01 23:07:46,401 - fba.demultiplex - INFO - Total UMIs: 121,595,388 / 121,595,388
2021-10-01 23:07:46,423 - fba.demultiplex - INFO - Median number of UMIs per cell: 7,659.
↪0 / 7,659.0
2021-10-01 23:07:46,423 - fba.demultiplex - INFO - Demultiplexing ...
2021-10-01 23:07:47,160 - fba.demultiplex - INFO - Generating heatmap ...
2021-10-01 23:07:52,192 - fba.demultiplex - INFO - Embedding ...
UMAP(dens_frac=0.0, dens_lambda=0.0, n_neighbors=10, random_state=42,
    verbose=True)
Construct fuzzy simplicial set
Fri Oct  1 23:07:52 2021 Finding Nearest Neighbors
Fri Oct  1 23:07:52 2021 Building RP forest with 10 trees
Fri Oct  1 23:07:53 2021 NN descent for 14 iterations
        1  /  14
        2  /  14
        Stopping threshold met -- exiting after 2 iterations
Fri Oct  1 23:08:08 2021 Finished Nearest Neighbor Search
Fri Oct  1 23:08:10 2021 Construct embedding
        completed  0  /  200 epochs
        completed  20  /  200 epochs
```

(continues on next page)

(continued from previous page)

```
        completed  40  /  200 epochs
        completed  60  /  200 epochs
        completed  80  /  200 epochs
        completed  100  /  200 epochs
        completed  120  /  200 epochs
        completed  140  /  200 epochs
        completed  160  /  200 epochs
        completed  180  /  200 epochs
Fri Oct  1 23:08:21 2021 Finished embedding
2021-10-01 23:08:22,267 - fba.__main__ - INFO - Done.
```

Heatmap of the relative abundance of features (CMOs) across all cells. Each column represents a single cell.



UMAP embedding of cells based on the abundance of features (CMOs, no transcriptome information used). Colors indicate the CMO status for each cell, as called by FBA.

Preview the demultiplexing result: the numbers of singlets (5,614 + 4,712), multiplets (1,505) and negative cells (1,781).

```
In [1]: import pandas as pd

In [2]: m = pd.read_csv("demultiplexed/matrix_cell_identity.csv.gz", index_col=0)

In [3]: m.loc[:, m.sum(axis=0) == 1].sum(axis=1)
Out[3]:
CMO301    5614
CMO302    4712
dtype: int64

In [4]: [sum(m.sum(axis=0) == i) for i in (2, 0)]
Out[4]: [1505, 1781]
```

Knee point

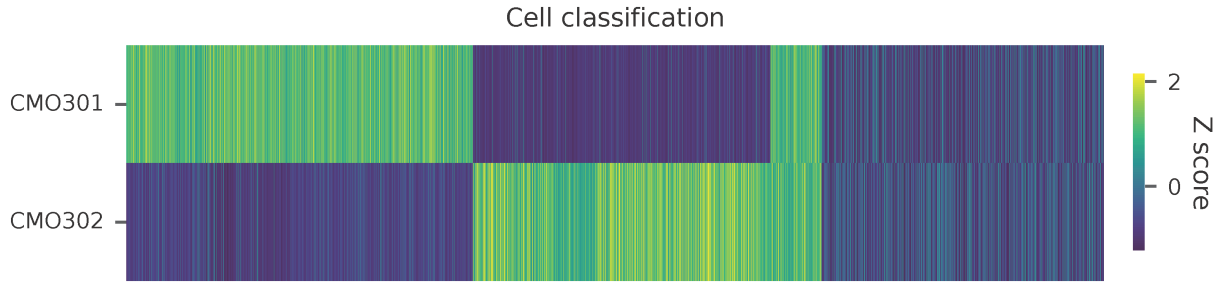
Method 1

Cells are demultiplexed based on the abundance of features, specifically CMOs. Demultiplexing method `5-2019` is our previous implementation, which aims to identify perturbations in the cells by detecting an inflection point on the feature UMI saturation curve (Xie, S., et al. 2019).

```
$ fba demultiplex \
    -i matrix_featurecount.csv.gz \
    -dm 5-2019 \
    -v
```

```
2022-01-02 13:52:10,698 - fba.__main__ - INFO - fba version: 0.0.x
2022-01-02 13:52:10,698 - fba.__main__ - INFO - Initiating logging ...
2022-01-02 13:52:10,698 - fba.__main__ - INFO - Python version: 3.9
2022-01-02 13:52:10,698 - fba.__main__ - INFO - Using demultiplex subcommand ...
2022-01-02 13:52:13,179 - fba.__main__ - INFO - Skipping arguments: "-q/--quantile", "-
↪cm/--clustering_method", "-p/--prob"
2022-01-02 13:52:13,179 - fba.demultiplex - INFO - Output directory: demultiplexed
2022-01-02 13:52:13,179 - fba.demultiplex - INFO - Demultiplexing method: 5-2019
2022-01-02 13:52:13,179 - fba.demultiplex - INFO - UMI normalization method: clr
2022-01-02 13:52:13,179 - fba.demultiplex - INFO - Visualization: On
2022-01-02 13:52:13,179 - fba.demultiplex - INFO - Visualization method: tsne
2022-01-02 13:52:13,179 - fba.demultiplex - INFO - Loading feature count matrix: matrix_
↪featurecount_filtered.csv.gz ...
2022-01-02 13:52:13,308 - fba.demultiplex - INFO - Number of cells: 13,612
2022-01-02 13:52:13,308 - fba.demultiplex - INFO - Number of positive cells for a␣
↪feature to be included: 200
2022-01-02 13:52:13,328 - fba.demultiplex - INFO - Number of features: 2 / 2 (after␣
↪filtering / original in the matrix)
2022-01-02 13:52:13,328 - fba.demultiplex - INFO - Features: CMO301 CMO302
2022-01-02 13:52:13,328 - fba.demultiplex - INFO - Total UMIs: 121,595,388 / 121,595,388
2022-01-02 13:52:13,338 - fba.demultiplex - INFO - Median number of UMIs per cell: 7,659.
↪0 / 7,659.0
2022-01-02 13:52:13,338 - fba.demultiplex - INFO - Demultiplexing ...
2022-01-02 13:52:14,446 - fba.demultiplex - INFO - Generating heatmap ...
2022-01-02 13:52:15,784 - fba.demultiplex - INFO - Embedding ...
2022-01-02 13:52:32,821 - fba.__main__ - INFO - Done.
```

Heatmap of the relative abundance of features (CMOs) across all cells. Each column represents a single cell.



t-SNE embedding of cells based on the abundance of features (CMOs, no transcriptome information used). Colors

indicate the CMO status for each cell, as called by FBA.

UMI distribution and knee point detection:

Preview the demultiplexing result: the numbers of singlets, multiplets and negative cells.

```
In [1]: import pandas as pd

In [2]: m = pd.read_csv("demultiplexed/matrix_cell_identity.csv.gz", index_col=0)

In [3]: m.loc[:, m.sum(axis=0) == 1].sum(axis=1)
Out[3]:
CMO301    4824
CMO302    4147
dtype: int64

In [4]: [sum(m.sum(axis=0) == i) for i in (2, 0)]
Out[4]: [716, 3925]
```

Method 2

Cells are demultiplexed based on the abundance of features, specifically CMOs. Demultiplexing method 5 is implemented to use the local maxima on the difference curve to detemine the knee point on the UMI saturation curve.

```
$ fba demultiplex \
    -i matrix_featurecount.csv.gz \
    -dm 5 \
    -v
```

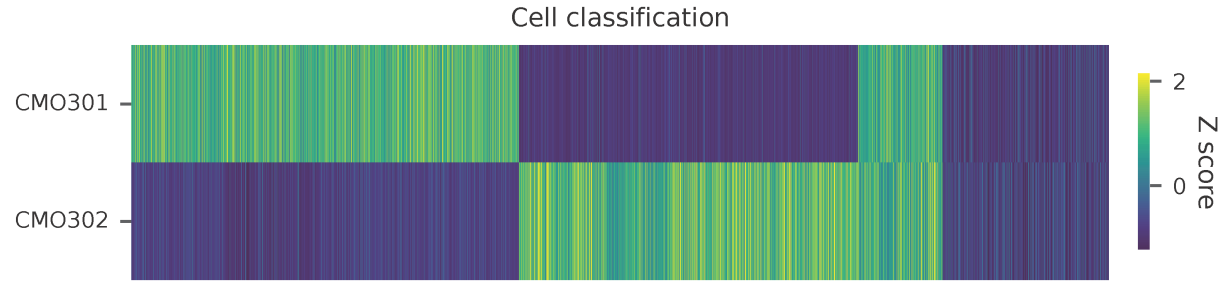
```
2022-01-02 13:53:33,250 - fba.__main__ - INFO - fba version: 0.0.x
2022-01-02 13:53:33,250 - fba.__main__ - INFO - Initiating logging ...
2022-01-02 13:53:33,250 - fba.__main__ - INFO - Python version: 3.9
2022-01-02 13:53:33,250 - fba.__main__ - INFO - Using demultiplex subcommand ...
2022-01-02 13:53:35,723 - fba.__main__ - INFO - Skipping arguments: "-q/--quantile", "-
↪cm/--clustering_method", "-p/--prob"
2022-01-02 13:53:35,723 - fba.demultiplex - INFO - Output directory: demultiplexed
2022-01-02 13:53:35,723 - fba.demultiplex - INFO - Demultiplexing method: 5
2022-01-02 13:53:35,723 - fba.demultiplex - INFO - UMI normalization method: clr
2022-01-02 13:53:35,723 - fba.demultiplex - INFO - Visualization: On
2022-01-02 13:53:35,724 - fba.demultiplex - INFO - Visualization method: tsne
2022-01-02 13:53:35,724 - fba.demultiplex - INFO - Loading feature count matrix: matrix_
↪featurecount_filtered.csv.gz ...
2022-01-02 13:53:35,852 - fba.demultiplex - INFO - Number of cells: 13,612
2022-01-02 13:53:35,852 - fba.demultiplex - INFO - Number of positive cells for a␣
↪feature to be included: 200
```

(continues on next page)

(continued from previous page)

```
2022-01-02 13:53:35,872 - fba.demultiplex - INFO - Number of features: 2 / 2 (after␣
↪filtering / original in the matrix)
2022-01-02 13:53:35,872 - fba.demultiplex - INFO - Features: CMO301 CMO302
2022-01-02 13:53:35,872 - fba.demultiplex - INFO - Total UMIs: 121,595,388 / 121,595,388
2022-01-02 13:53:35,883 - fba.demultiplex - INFO - Median number of UMIs per cell: 7,659.
↪0 / 7,659.0
2022-01-02 13:53:35,883 - fba.demultiplex - INFO - Demultiplexing ...
2022-01-02 13:53:36,435 - fba.demultiplex - INFO - Generating heatmap ...
2022-01-02 13:53:37,779 - fba.demultiplex - INFO - Embedding ...
2022-01-02 13:53:56,162 - fba.__main__ - INFO - Done.
```

Heatmap of the relative abundance of features (CMOs) across all cells. Each column represents a single cell.



t-SNE embedding of cells based on the abundance of features (CMOs, no transcriptome information used). Colors indicate the CMO status for each cell, as called by FBA.

UMI distribution and knee point detection:

Preview the demultiplexing result: the numbers of singlets (5,396 + 4,726), multiplets (1,178) and negative cells (2,312).

```
In [1]: import pandas as pd

In [2]: m = pd.read_csv("demultiplexed/matrix_cell_identity.csv.gz", index_col=0)

In [3]: m.loc[:, m.sum(axis=0) == 1].sum(axis=1)
Out[3]:
CMO301    5396
CMO302    4726
dtype: int64

In [4]: [sum(m.sum(axis=0) == i) for i in (2, 0)]
Out[4]: [1178, 2312]
```

3.5.2 30k Mouse E18 Combined Cortex, Hippocampus and Subventricular Zone Nuclei Multiplexed

Dataset: 30k Mouse E18 Combined Cortex, Hippocampus and Subventricular Zone Nuclei Multiplexed, 12 CMOs

The detailed description of this dataset can be found here.

Note: The processing of QC, barcode extraction and matrix generation is similar to *10k 1:1 Mixture of Raji and Jurkat Cells Multiplexed, 2 CMOs*.

Demultiplexing

In summary, 799,808,703 of total 1,238,424,843 read pairs have the valid structure (`-cb_m 2`, `-fb_m 1`). The average UMIs per cell is 22,962.0 for this feature barcode library.

Inspect feature count matrix.

```
In [1]: import pandas as pd

In [2]: m = pd.read_csv("matrix_featurecount.csv.gz", index_col=0)

In [3]: m.shape
Out[3]: (12, 59342)

In [4]: m.sum(axis=1)
Out[4]:
CMO301_ATGAGGAATTCCTGC     90849728
CMO302_CATGCCAATAGAGCG     73859662
CMO303_CCGTCGTCCAAGCAT     94388625
CMO304_AACGTTAATCACTCA    112458502
CMO305_CGCGATATGGTCGGA     57147183
CMO306_AAGATGAGGTCTGTG     67979145
CMO307_AAGCTCGTTGGAAGA    191026854
CMO308_CGGATTCCACATCAT     97445798
CMO309_GTTGATCTATAACAG    189936250
CMO310_GCAGGAGGTATCAAT    130990984
CMO311_GAATCGTGATTCTTC    131131183
CMO312_ACATGGTCAACGCTG     55103315
dtype: int64

In [5]: (m > 0).sum(axis=1)
Out[5]:
CMO301_ATGAGGAATTCCTGC    59128
CMO302_CATGCCAATAGAGCG    59088
CMO303_CCGTCGTCCAAGCAT    59096
CMO304_AACGTTAATCACTCA    59194
CMO305_CGCGATATGGTCGGA    58794
CMO306_AAGATGAGGTCTGTG    58932
```

(continues on next page)

(continued from previous page)

```
CMO307_AAGCTCGTTGGAAGA    59306
CMO308_CGGATTCCACATCAT    59078
CMO309_GTTGATCTATAACAG    59292
CMO310_GCAGGAGGTATCAAT    59202
CMO311_GAATCGTGATTCTTC    59204
CMO312_ACATGGTCAACGCTG    58836
dtype: int64
```

Gaussian mixture model

Cells are demulitplexed based on the feature count matrix (CMO abundance). Demultiplexing method 2 (set by `-dm`) is inspired by the method described on 10x Genomics' website. A cell identity matrix is generated in the output directory: 0 means negative, 1 means positive. To set normalization method, use `-nm` (default `clr`). To set the probability threshold for demultiplexing, use `-p` (default `0.9`). To generate visualization plots, set `-v`. To choose visualization method, use `-vm` (default `tsne`).

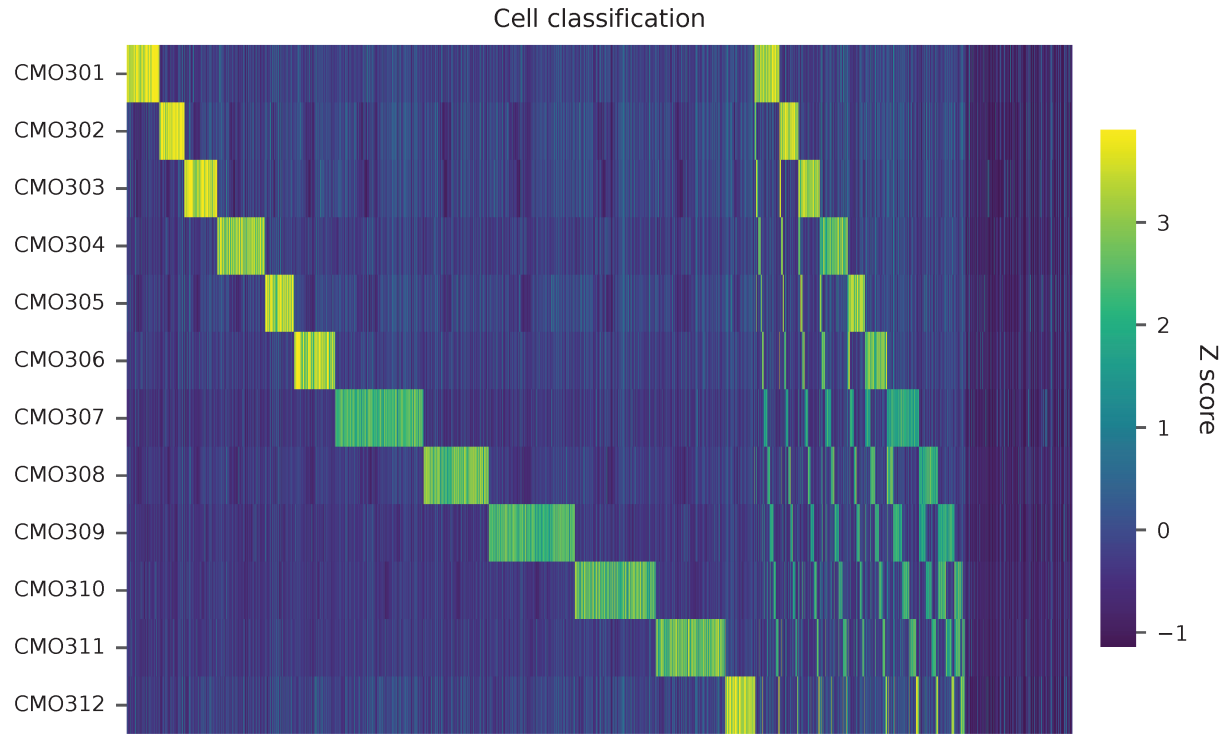
```
$ fba demultiplex \
    -i matrix_featurecount.csv.gz \
    --output_directory demultiplexed \
    -dm 2 \
    -v
```

```
2021-10-05 20:45:58,069 - fba.__main__ - INFO - fba version: 0.0.x
2021-10-05 20:45:58,069 - fba.__main__ - INFO - Initiating logging ...
2021-10-05 20:45:58,069 - fba.__main__ - INFO - Python version: 3.8
2021-10-05 20:45:58,069 - fba.__main__ - INFO - Using demultiplex subcommand ...
2021-10-05 20:46:17,903 - fba.__main__ - INFO - Skipping arguments: "-q/--quantile", "-
↪cm/--clustering_method"
2021-10-05 20:46:17,903 - fba.demultiplex - INFO - Output directory: demultiplexed
2021-10-05 20:46:17,903 - fba.demultiplex - INFO - Demultiplexing method: 2
2021-10-05 20:46:17,903 - fba.demultiplex - INFO - UMI normalization method: clr
2021-10-05 20:46:17,903 - fba.demultiplex - INFO - Visualization: On
2021-10-05 20:46:17,903 - fba.demultiplex - INFO - Visualization method: tsne
2021-10-05 20:46:17,903 - fba.demultiplex - INFO - Loading feature count matrix: matrix_
↪featurecount.csv.gz ...
2021-10-05 20:46:27,051 - fba.demultiplex - INFO - Number of cells: 31,171
2021-10-05 20:46:27,052 - fba.demultiplex - INFO - Number of positive cells for a␣
↪feature to be included: 200
2021-10-05 20:46:27,163 - fba.demultiplex - INFO - Number of features: 12 / 12 (after␣
↪filtering / original in the matrix)
2021-10-05 20:46:27,163 - fba.demultiplex - INFO - Features: CMO301 CMO302 CMO303 CMO304␣
↪CMO305 CMO306 CMO307 CMO308 CMO309 CMO310 CMO311 CMO312
2021-10-05 20:46:27,164 - fba.demultiplex - INFO - Total UMIs: 713,913,321 / 713,913,321
2021-10-05 20:46:27,218 - fba.demultiplex - INFO - Median number of UMIs per cell: 22,
↪962.0 / 22,962.0
2021-10-05 20:46:27,218 - fba.demultiplex - INFO - Demultiplexing ...
2021-10-05 20:46:29,001 - fba.demultiplex - INFO - Generating heatmap ...
2021-10-05 20:47:17,305 - fba.demultiplex - INFO - Embedding ...
2021-10-05 20:49:27,083 - fba.__main__ - INFO - Done.
```

According to the description of this dataset:

The four E18 mouse nuclei samples were multiplexed at equal proportions with 3 CMOs per nuclei sample, resulting in a pooled sample labeled with 12 CMOs. Nuclei from the non-multiplexed sample were used as one of the four sample types composing the multiplexed sample.

Heatmap of the relative abundance of features (CMOs) across all cells. Each column represents a single cell. Multiplets have more than one CMOs.



t-SNE embedding of cells based on the abundance of features (CMOs, no transcriptome information used). Colors indicate the CMO status for each cell, as called by FBA. Twelve singlet clusters and cross-oligo multiplet clusters are clearly present.

Preview the demultiplexing result: the numbers of singlets.

```
In [1]: import pandas as pd

In [2]: m = pd.read_csv("demultiplexed/matrix_cell_identity.csv.gz", index_col=0)

In [3]: m.loc[:, m.sum(axis=0) == 1].sum(axis=1)
Out[3]:
CMO301    1078
CMO302     824
CMO303    1085
CMO304    1575
CMO305     959
CMO306    1362
CMO307    2912
CMO308    2144
CMO309    2841
CMO310    2675
```

(continues on next page)

(continued from previous page)

```
CMO311    2292
CMO312     951
dtype: int64
```

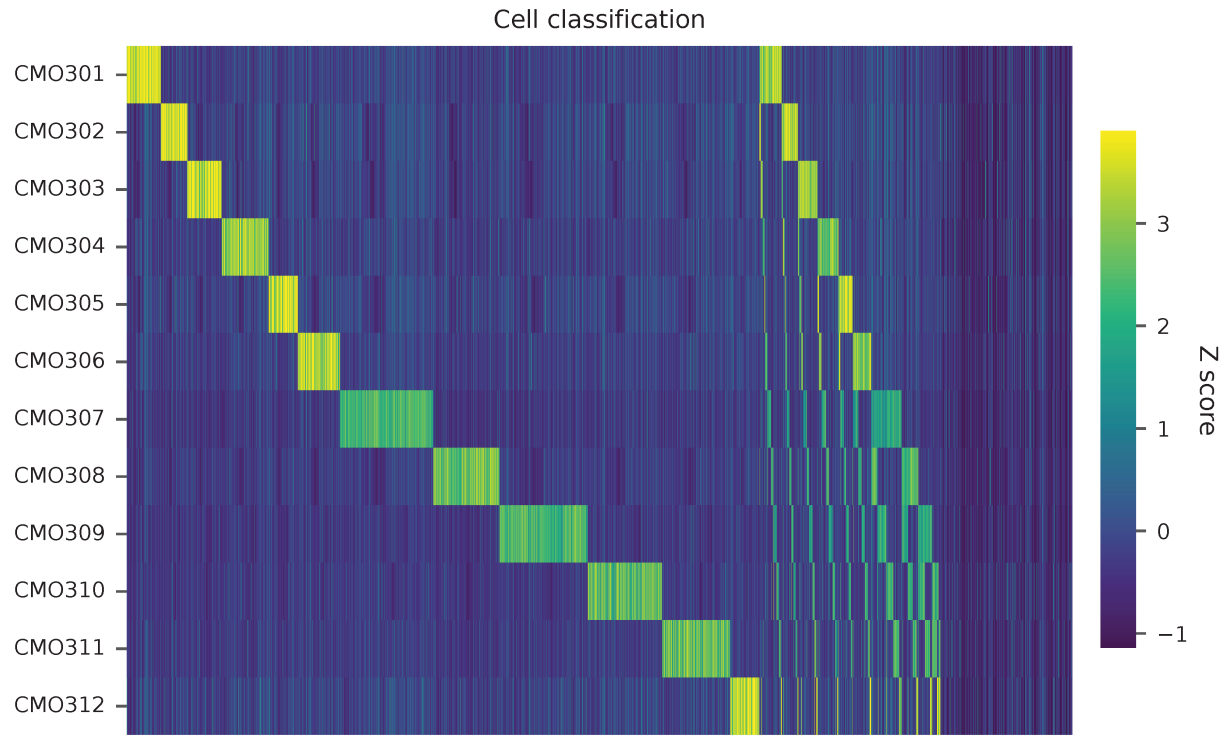
Kernel density estimation

Cells are demulitplexed based on the feature count matrix (CMO abundance) using demultiplexing method 4, which is implemented with modifications to the method described in McGinnis, C., et al. (2019). A cell identity matrix is generated in the output directory: 0 means negative, 1 means positive. To generate visualization plots, set `-v`.

```
$ fba demultiplex \
    -i matrix_featurecount.csv.gz \
    -dm 4 \
    -v
```

```
2021-12-27 12:03:15,693 - fba.__main__ - INFO - fba version: 0.0.x
2021-12-27 12:03:15,693 - fba.__main__ - INFO - Initiating logging ...
2021-12-27 12:03:15,693 - fba.__main__ - INFO - Python version: 3.9
2021-12-27 12:03:15,693 - fba.__main__ - INFO - Using demultiplex subcommand ...
2021-12-27 12:03:18,145 - fba.__main__ - INFO - Skipping arguments: "-q/--quantile", "-
↪cm/--clustering_method", "-p/--prob"
2021-12-27 12:03:18,145 - fba.demultiplex - INFO - Output directory: demultiplexed
2021-12-27 12:03:18,145 - fba.demultiplex - INFO - Demultiplexing method: 4
2021-12-27 12:03:18,145 - fba.demultiplex - INFO - UMI normalization method: clr
2021-12-27 12:03:18,145 - fba.demultiplex - INFO - Visualization: On
2021-12-27 12:03:18,145 - fba.demultiplex - INFO - Visualization method: tsne
2021-12-27 12:03:18,145 - fba.demultiplex - INFO - Loading feature count matrix: matrix_
↪featurecount.csv.gz ...
2021-12-27 12:03:18,453 - fba.demultiplex - INFO - Number of cells: 31,171
2021-12-27 12:03:18,453 - fba.demultiplex - INFO - Number of positive cells for a␣
↪feature to be included: 200
2021-12-27 12:03:18,499 - fba.demultiplex - INFO - Number of features: 12 / 12 (after␣
↪filtering / original in the matrix)
2021-12-27 12:03:18,499 - fba.demultiplex - INFO - Features: CMO301 CMO302 CMO303 CMO304␣
↪CMO305 CMO306 CMO307 CMO308 CMO309 CMO310 CMO311 CMO312
2021-12-27 12:03:18,499 - fba.demultiplex - INFO - Total UMIs: 713,913,321 / 713,913,321
2021-12-27 12:03:18,523 - fba.demultiplex - INFO - Median number of UMIs per cell: 22,
↪962.0 / 22,962.0
2021-12-27 12:03:18,523 - fba.demultiplex - INFO - Demultiplexing ...
2021-12-27 12:03:39,128 - fba.demultiplex - INFO - Quantile cutoff: 49
2021-12-27 12:03:51,501 - fba.demultiplex - INFO - Generating heatmap ...
2021-12-27 12:04:07,664 - fba.demultiplex - INFO - Embedding ...
2021-12-27 12:04:56,977 - fba.__main__ - INFO - Done.
```

Heatmap of relative abundance of feature across all cells. Each column represents a single cell.



t-SNE embedding of cells based on the abundance of features (no transcriptome information used). Colors indicate the sgRNA status for each cell, as called by FBA.

Preview the demultiplexing result: the numbers of singlets.

```
In [1]: import pandas as pd

In [2]: m = pd.read_csv("demultiplexed/matrix_cell_identity.csv.gz", index_col=0)

In [3]: m.loc[:, m.sum(axis=0) == 1].sum(axis=1)
Out[3]:
CMO301    1127
CMO302     872
CMO303    1124
CMO304    1562
CMO305     950
CMO306    1386
CMO307    3085
CMO308    2187
CMO309    2914
CMO310    2452
CMO311    2248
CMO312     950
dtype: int64
```

- *10k 1:1 Mixture of Raji and Jurkat Cells Multiplexed, 2 CMOs*
- *30k Mouse E18 Combined Cortex, Hippocampus and Subventricular Zone Nuclei Multiplexed, 12 CMOs*

3.6 Cell hashing

3.6.1 Peripheral Blood Mononuclear Cells with 8 Antibodies

Dataset: Cell hashing

Stoeckius, M., Zheng, S., Houck-Loomis, B., Hao, S., Yeung, B.Z., Mauck, W.M., 3rd, Smibert, P., and Satija, R. (2018). Cell Hashing with barcoded antibodies enables multiplexing and doublet detection for single cell genomics. *Genome Biol.* **19**, 224.

Preparation

Download fastq files from European Nucleotide Archive.

```
$ wget ftp.sra.ebi.ac.uk/vol1/fastq/SRR828/007/SRR8281307/SRR8281307_1.fastq.gz

$ wget ftp.sra.ebi.ac.uk/vol1/fastq/SRR828/007/SRR8281307/SRR8281307_2.fastq.gz
```

Download cell barcode info.

These are the cell-associated barcodes in this single cell RNA-Seq library (determined by the number of transcriptomic UMIs captured per barcode).

```
$ curl -O https://ftp.ncbi.nlm.nih.gov/geo/samples/GSM2895nnn/GSM2895283/suppl/
↪GSM2895283_Hashtag-HTO-count.csv.gz

$ gzip -dc GSM2895283_Hashtag-HTO-count.csv.gz | head -1 | sed 's/,/\n/g' | grep -v '^$'␣
↪> cell_barcodes.txt
```

Inspect cell barcodes.

```
$ head cell_barcodes.txt

GGCGACTAGAGGACGG
CATCAAGGTCTTGTCC
AAACCTGAGTGATCGG
TGAGGGAGTACTTAGC
CCTAAAGAGATGTGGC
AGACGTTTCAGCCTAA
TGGGAAGCAACACCCG
CGATTGATCTTCGGTC
CATCGAAGTGATGCCC
TCAGATGCACGAGAGT
```

Prepare feature barcodes (hashtag-oligos, HTO).

```
$ gzip -dc GSM2895283_Hashtag-HTO-count.csv.gz | cut -d ',' -f1 | grep Batch | gsed 's/-/
↪\t/g' > feature_barcodes.tsv
```

Inspect feature barcodes.

```
$ cat feature_barcodes.tsv

BatchA  AGGACCATCCAA
BatchB  ACATGTTACCGT
BatchC  AGCTTACTATCC
BatchD  TCGATAATGCGA
BatchE  GAGGCTGAGCTA
BatchF  GTGTGACGTATT
BatchG  ACTGTCTAACGG
BatchH  TATCACATCGGT
```

QC

The first 20,000 read pairs are sampled (default `100,000`, set by `-n`,) for quality control. The `-t` option can be used to set the number of threads. By default, diagnostic results and plots are generated in the `qc` directory (set by `--output_directory`), and the full length of read 1 and read 2 are searched against reference cell and feature barcodes, respectively. The per base content of both read pairs and the distribution of matched barcode positions are summarized. Use `-r1_c` and/or `-r2_c` to limit the search range, and `-cb_n` and/or `-fb_n` to set the mismatch tolerance for cell and/or feature barcode matching (default 3).

```
$ fba qc \
    -1 SRR8281307_1.fastq.gz \
    -2 SRR8281307_2.fastq.gz \
    -w cell_barcodes.txt \
    -f feature_barcodes.tsv \
    --output_directory qc \
    -n 20000
```

This library was constructed using Chromium Single Cell 3' Reagent Kits (v2 Chemistry). The first 16 bases correspond to cell barcodes and the following 10 bases correspond to UMIs on read 1. Based on the base content plot, both the cell barcode and UMI sequences exhibit even GC content. A poly-A tail starts at base 26.

As for read 2, based on the per base content, it suggests that bases 0-11 exhibit relatively balanced GC content for the reads that we have sampled. Starting from base 12, there is a poly-A tail. Bases 0-11 are hashtag oligo sequences, and the majority of the reads appear to have the expected structure.

The detailed `qc` results are stored in `feature_barcoding_output.tsv.gz` file. `matching_pos` columns indicate the matched positions on reads. `matching_description` columns indicate mismatches in substitutions:insertions:deletions format.

```
$ gzip -dc qc/feature_barcoding_output.tsv.gz | head

read1_seq       cell_barcode    cb_matching_pos cb_matching_description read2_seq      ␣
↪feature_barcode fb_matching_pos fb_matching_description
NTCCGAACATATGAGAGCAATAGTCGTTT   CGAACATGTAAGAGAG        3:17    1:0:2  ␣
↪NCATGTTACCGTGAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAACAGCAATTGTCACTTATAGGAGGAGAAGAAGGGAAGGGGGGGGGGGGGGGAAA␣
↪    BatchB_ACATGTTACCGT     0:12    1:0:0
NAACGGATCCACGAATGAAGGACGCCTTT   TACGGTATCCACGAAT        1:16    1:0:1  ␣
↪NNGNNAATGCGAGAAAAAAAAAAAAAAAAAAAAAAAAAAAAGGGGCGCTCTCTTCGGGGGGGCGGGGAGAGCGAAGGAGGGGGGGGGGGGGGAAGGAG␣
↪    no_match        NA      NA
NGGCCAGTCTTCAACTGTTAACACTATTT   GTCCTCAAGCTGTCTA        6:20    1:0:2  ␣
↪NNNNNNNNNNNNAAANNAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAGGTTTAAAAAGTGAAAGAGGGACAAAACGGGAAAAACGGGGGTGGGGAAAA␣
↪    no_match        NA      NA
NATCCAGCAATACGCTTTCCACGACATTT   ATCCACCCATACGCTA        1:17    3:0:0  ␣
↪NNNNNNNNNNNNAAANNAAAAAAAAAAAAAAAAAAAAAAAGTGGGGGGAAAGCGGTTTTGGGAGATAAAACGAAAAAGCGGCGGGGGGGGGAAAAAGGTGA␣
↪    no_match        NA      NA
NTGCGATAGACACTAAGAGGAGTTCATTT   CGCGGTAAGACACTAA        1:16    2:0:1  ␣
↪NCGATAATGCGACAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAACCCCCTTTGTTTTTATCGTAAAGATGGGAAGGGGGCGGTGGAGGGAAAAA␣
↪    BatchD_TCGATAATGCGA     0:12    1:0:0
NTGATCCAGAAGGTGAGGGAGGCTGATTT   AGATTGCGTGAGGGAG        7:21    1:0:2  ␣
↪NNNNNNNNNNNNNAANNAAAAAAAAAAAAAAAAAAAATCACCCCCCCCCCCCCTTTTGGTTCAAAAACGGAAAAAGCGCCGCGGGGGGAAAGAGTGTAAAT␣
↪    no_match        NA      NA
NTGGGTCAGGCCGAATTGAAGGGATGTTT   GAAATGAAGTGAAGTT        12:28   3:0:0  ␣
↪NNNNNNCTATCCAAAANNAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAACCCCTTCAATTGGCCCAGACCCAACACTCGAAGGGCCGGCTGGCAGCAAA␣
↪    no_match        NA      NA
NGAGAAGTCTCGATGAATCTAGCCGCTTT   CGATTGAAGCTAGCCC        10:25   2:0:1  ␣
↪NNNNNNNNNCTNCAAANNAAAAAAAAAAAAAAAAATAAAAAAAACGGGCTGATCCCAAGCAGACGTCACAAAGAAGCGAGAGAGTGGGATTGAGAAAAAGA␣
↪    no_match        NA      NA
NCACGGAGTTCCCTTGCCAATGTAGTTTT   AGGGAGTTCGTTTGCC        2:18    3:0:0  ␣
↪NGCTTACTATCCTAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAATATGGGGGGGGGGAATCGGGGGGGAGGGGAAAGGGGGGGTGGGGGAAAAAAGA␣
↪    BatchC_AGCTTACTATCC     0:12    1:0:0
```

Barcode extraction

The lengths of cell and feature barcodes (hashtags) are all identical (16 and 12, respectively). And based on the `qc` results, the distributions of starting and ending positions of cell and feature barcodes are very uniform. Search ranges are set to `0,16` on read 1 and `0,12` on read 2. One mismatch for cell and feature barcodes (`-cb_m`, `-cf_m`) are allowed. By default, three ambiguous nucleotides (Ns) for read 1 and read2 (`-cb_n`, `-cf_n`) are allowed.

```
$ fba extract \
    -1 SRR8281307_1.fastq.gz \
    -2 SRR8281307_2.fastq.gz \
    -w cell_barcodes.txt \
    -f feature_barcodes.tsv \
    -o feature_barcoding_output.tsv.gz \
    -r1_coords 0,16 \
    -r2_coords 0,12 \
    -cb_m 1 \
```

(continues on next page)

(continued from previous page)

```
    -fb_m 1 \
    -cb_n 3 \
    -fb_n 3
```

Preview of result.

```
$ gzip -dc feature_barcoding_output.tsv.gz | head

read1_seq       cell_barcode    cb_num_mismatches       read2_seq       feature_barcode␣
↪fb_num_mismatches
NTCCGAACATATGAGAgcaatagtcgttt   ATCCGAACATATGAGA        1       ␣
↪NCATGTTACCGTgaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaacagcaattgtcacttataggaggagaagaagggaagggggggggggggggaaa␣
↪   BatchB_ACATGTTACCGT     1
NTGCGATAGACACTAAgaggagttcattt   ATGCGATAGACACTAA        1       ␣
↪NCGATAATGCGAcaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaccccctttgtttttatcgtaaagatgggaaggggcggtggagggaaaaa␣
↪   BatchD_TCGATAATGCGA     1
NCACGGAGTTCCCTTGccaatgtagtttt   CCACGGAGTTCCCTTG        1       ␣
↪NGCTTACTATCCtaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaatatgggggggggggaatcgggggggaggggaaaggggggggtgggggaaaaaaga␣
↪   BatchC_AGCTTACTATCC     1
NGGGATGCAGCTTAACcgggcatcgcttt   AGGGATGCAGCTTAAC        1       ␣
↪NCATGTTACCGTcaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaatgaaatggaagtaggggtgtccctagtctgtagaagcggcgactggggaaatgtat␣
↪   BatchB_ACATGTTACCGT     1
NTTGTCACATACGCTAcgagcctgcattt   TTTGTCACATACGCTA        1       ␣
↪NATCACATCGGTtaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaagaaggccgggggggggggggaaaaaaaaaaaaaaaaaggcggggtggggagagagtga␣
↪   BatchH_TATCACATCGGT     1
NGCTCTCGTTCCACGGaggttatcggttt   AGCTCTCGTTCCACGG        1       ␣
↪NCTGTCTAACGGgaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaccccccgggggaggggaaaaaaagcaggaaaagcgccatgggggaaaaaaaaa␣
↪   BatchG_ACTGTCTAACGG     1
GATCTAGCAATGTTGCcaaccatttttttt   GATCTAGCAATGTTGC        0       ␣
↪AGGACCATCCAAgaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaagatggaggaacttggttagaacagaaggaggaggggtgggggggggaa␣
↪   BatchA_AGGACCATCCAA     0
NTTGCGCCATGGTCATagtaacaagattt   TTTGCGCCATGGTCAT        1       ␣
↪NCATGTTACCGTcaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaatcttttttctttgccctgggcgaaaaagatgggaggaggggggggggggggaaagggt␣
↪   BatchB_ACATGTTACCGT     1
CGCGGTAAGACACTAAcggccgtggtttt   CGCGGTAAGACACTAA        0       ␣
↪TATCACATCGGTtaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaacccgggcgggtggggttttacgaggaaggggagcaggggggggtggaggaaaaaaa␣
↪   BatchH_TATCACATCGGT     0
```

Result summary.

91.5% (67,916,430 out of 74,219,921) of total read pairs have valid cell and feature barcodes. Majority of the fragments in this library have the correct structure.

```
2021-02-17 16:16:13,003 - fba.__main__ - INFO - fba version: 0.0.7
2021-02-17 16:16:13,003 - fba.__main__ - INFO - Initiating logging ...
2021-02-17 16:16:13,003 - fba.__main__ - INFO - Python version: 3.7
2021-02-17 16:16:13,003 - fba.__main__ - INFO - Using extract subcommand ...
2021-02-17 16:16:13,026 - fba.levenshtein - INFO - Number of reference cell barcodes: 65,
↪000
2021-02-17 16:16:13,027 - fba.levenshtein - INFO - Number of reference feature barcodes:␣
↪8
2021-02-17 16:16:13,027 - fba.levenshtein - INFO - Read 1 coordinates to search: [0, 16)
2021-02-17 16:16:13,027 - fba.levenshtein - INFO - Read 2 coordinates to search: [0, 12)
```

(continues on next page)

(continued from previous page)

```
2021-02-17 16:16:13,027 - fba.levenshtein - INFO - Cell barcode maximum number of␣
↪mismatches: 1
2021-02-17 16:16:13,027 - fba.levenshtein - INFO - Feature barcode maximum number of␣
↪mismatches: 1
2021-02-17 16:16:13,027 - fba.levenshtein - INFO - Read 1 maximum number of N allowed: 3
2021-02-17 16:16:13,027 - fba.levenshtein - INFO - Read 2 maximum number of N allowed: 3
2021-02-17 16:16:15,500 - fba.levenshtein - INFO - Matching ...
2021-02-17 16:28:20,306 - fba.levenshtein - INFO - Read pairs processed: 10,000,000
2021-02-17 16:40:24,344 - fba.levenshtein - INFO - Read pairs processed: 20,000,000
2021-02-17 16:52:14,506 - fba.levenshtein - INFO - Read pairs processed: 30,000,000
2021-02-17 17:04:04,292 - fba.levenshtein - INFO - Read pairs processed: 40,000,000
2021-02-17 17:15:52,792 - fba.levenshtein - INFO - Read pairs processed: 50,000,000
2021-02-17 17:27:43,975 - fba.levenshtein - INFO - Read pairs processed: 60,000,000
2021-02-17 17:39:35,941 - fba.levenshtein - INFO - Read pairs processed: 70,000,000
2021-02-17 17:44:36,162 - fba.levenshtein - INFO - Number of read pairs processed: 74,
↪219,921
2021-02-17 17:44:36,162 - fba.levenshtein - INFO - Number of read pairs w/ valid␣
↪barcodes: 67,916,430
2021-02-17 17:44:36,264 - fba.__main__ - INFO - Done.
```

Matrix generation

Only fragments with correctly matched cell and feature barcodes are included, while fragments with UMI lengths less than the specified value are discarded. UMI removal is performed using UMI-tools (Smith, T., et al. 2017. Genome Res. 27, 491–499.), with the starting position on read 1 set by `-us` (default `16`) and the length set by `-ul` (default `12`). The UMI deduplication method can be set using `-ud` (default `directional`), and the UMI deduplication mismatch threshold can be specified using `-um` (default `1`).

The generated feature count matrix can be easily imported into well-established single cell analysis packages: Seurat and Scanpy.

```
$ fba count \
    -i feature_barcoding_output.tsv.gz \
    -o matrix_featurecount.csv.gz \
    -us 16 \
    -ul 10 \
    -um 1 \
    -ud directional
```

Result summary.

25.1% (17,022,991 out of 67,916,430) of read pairs with valid cell and feature barcodes are unique fragments. 22.9% (17,022,991 out of 74,219,921) of total sequenced read pairs contribute to the final matrix.

```
2021-02-17 17:44:43,315 - fba.__main__ - INFO - fba version: 0.0.7
2021-02-17 17:44:43,315 - fba.__main__ - INFO - Initiating logging ...
2021-02-17 17:44:43,315 - fba.__main__ - INFO - Python version: 3.7
2021-02-17 17:44:43,315 - fba.__main__ - INFO - Using count subcommand ...
```

(continues on next page)

(continued from previous page)

```
2021-02-17 17:44:43,315 - fba.count - INFO - UMI-tools version: 1.1.1
2021-02-17 17:44:43,318 - fba.count - INFO - UMI starting position on read 1: 16
2021-02-17 17:44:43,318 - fba.count - INFO - UMI length: 10
2021-02-17 17:44:43,318 - fba.count - INFO - UMI-tools deduplication threshold: 1
2021-02-17 17:44:43,318 - fba.count - INFO - UMI-tools deduplication method: directional
2021-02-17 17:44:43,318 - fba.count - INFO - Header line: read1_seq cell_barcode cb_num_
↪mismatches read2_seq feature_barcode fb_num_mismatches
2021-02-17 17:48:32,866 - fba.count - INFO - Number of lines processed: 67,916,430
2021-02-17 17:48:33,127 - fba.count - INFO - Number of cell barcodes detected: 64,998
2021-02-17 17:48:33,127 - fba.count - INFO - Number of features detected: 8
2021-02-17 18:01:15,176 - fba.count - INFO - Total UMIs after deduplication: 17,022,991
2021-02-17 18:01:15,298 - fba.count - INFO - Median number of UMIs per cell: 63.0
2021-02-17 18:01:16,924 - fba.__main__ - INFO - Done.
```

Demultiplexing

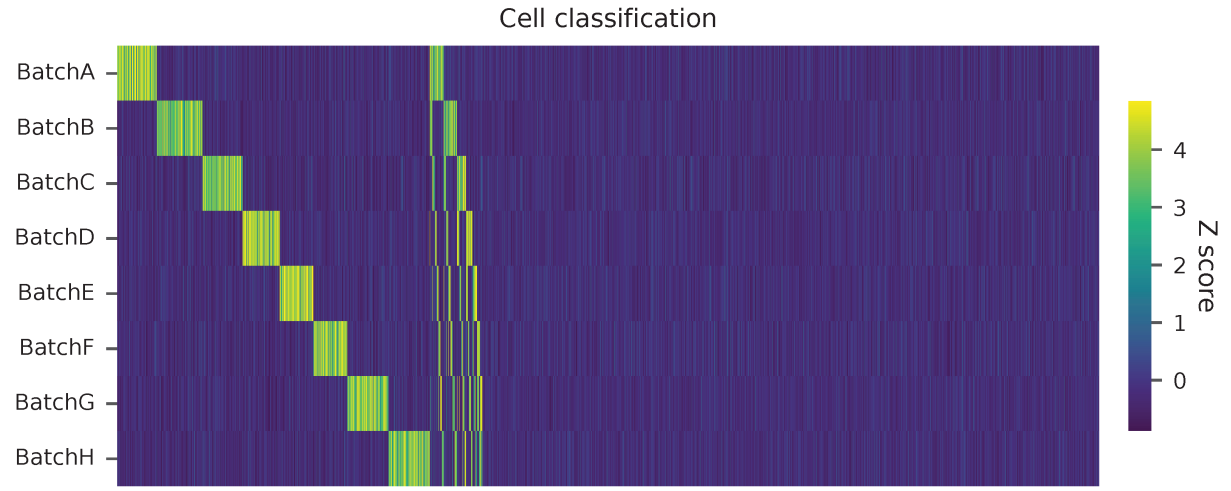
Negative binomial distribution

Cells are classified based on the abundance of features (HTOs, no transcriptome information used). Demultiplexing method 1 (set by `-dm`) is implemented based on the method described in Stoeckius, M., et al. (2018) with some modifications. A cell identity matrix is generated in the output directory (set by `--output_directory`, default `demultiplexed`): 0 means negative, 1 means positive. To adjust the quantile threshold for demultiplexing, use `-q` (Default `0.9999`). To generate visualization plots, set `-v`.

```
$ fba demultiplex \
    -i matrix_featurecount.csv.gz \
    --output_directory demultiplexed \
    -dm 1 \
    -v
```

```
2021-02-18 01:27:19,172 - fba.__main__ - INFO - fba version: 0.0.7
2021-02-18 01:27:19,173 - fba.__main__ - INFO - Initiating logging ...
2021-02-18 01:27:19,173 - fba.__main__ - INFO - Python version: 3.7
2021-02-18 01:27:19,173 - fba.__main__ - INFO - Using demultiplex subcommand ...
2021-02-18 01:27:19,177 - fba.demultiplex - INFO - Output directory: demultiplexed
2021-02-18 01:27:19,177 - fba.demultiplex - INFO - Loading feature count matrix: matrix_
↪featurecount.csv.gz ...
2021-02-18 01:27:22,932 - fba.demultiplex - INFO - Number of cells: 64,998
2021-02-18 01:27:22,932 - fba.demultiplex - INFO - Number of features: 8
2021-02-18 01:27:22,932 - fba.demultiplex - INFO - Total UMIs: 17,021,991
2021-02-18 01:27:23,029 - fba.demultiplex - INFO - Median number of UMIs per cell: 63.0
2021-02-18 01:27:23,029 - fba.demultiplex - INFO - Demultiplexing ...
2021-02-18 03:19:27,245 - fba.demultiplex - INFO - Generating heatmap ...
2021-02-18 03:20:37,827 - fba.demultiplex - INFO - Embedding ...
2021-02-18 03:21:21,120 - fba.__main__ - INFO - Done.
```

Heatmap of the relative abundance of features (HTOs) across all cells. Each column represents a single cell. This is a re-creation of Fig. 1c in Stoeckius, M., et al. (2018).



t-SNE embedding of cells based on the abundance of features (HTOs, no transcriptome information used). Colors indicate the HTO status for each cell, as called by FBA. This is a re-creation of Fig. 1d in Stoeckius, M., et al. (2018).

Preview the demultiplexing result: the numbers of singlets. The result in Stoeckius, M., et al. (2018) can be found in Additional file 3.

```
In [1]: import pandas as pd

In [2]: m = pd.read_csv("demultiplexed/matrix_cell_identity.csv.gz", index_col=0)

In [3]: m.loc[:, m.sum(axis=0) == 1].sum(axis=1)
Out[3]:
BatchA    2637
BatchB    3019
BatchC    2666
BatchD    2441
BatchE    2242
BatchF    2234
BatchG    2747
BatchH    2719
dtype: int64
```

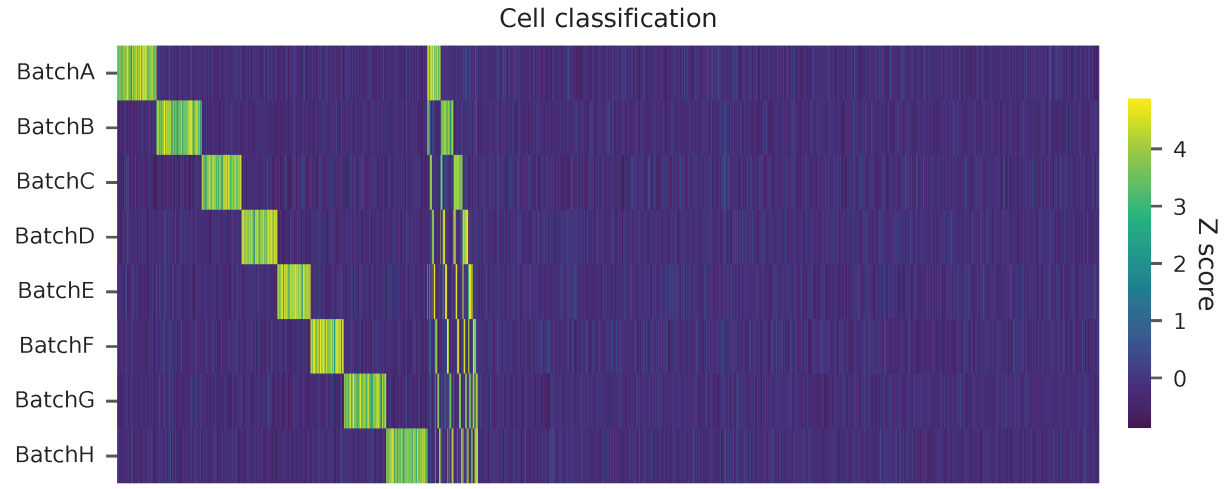
Gaussian mixture model

Alternatively, cells can be demultiplexed using gaussian mixture model. The implementation of demultiplexing method `2` (set by `-dm`) is inspired by the method described on *10x Genomics' website*. To set the probability threshold for demultiplexing, use `-p` (default `0.9`). To generate visualization plots, set -v.

```
$ fba demultiplex \
    -i matrix_featurecount.csv.gz \
    -dm 2 \
    -v
```

```
2021-12-27 11:37:31,026 - fba.__main__ - INFO - fba version: 0.0.x
2021-12-27 11:37:31,026 - fba.__main__ - INFO - Initiating logging ...
2021-12-27 11:37:31,026 - fba.__main__ - INFO - Python version: 3.9
2021-12-27 11:37:31,026 - fba.__main__ - INFO - Using demultiplex subcommand ...
2021-12-27 11:37:33,496 - fba.__main__ - INFO - Skipping arguments: "-q/--quantile", "-
↪cm/--clustering_method"
2021-12-27 11:37:33,496 - fba.demultiplex - INFO - Output directory: demultiplexed
2021-12-27 11:37:33,496 - fba.demultiplex - INFO - Demultiplexing method: 2
2021-12-27 11:37:33,496 - fba.demultiplex - INFO - UMI normalization method: clr
2021-12-27 11:37:33,496 - fba.demultiplex - INFO - Visualization: On
2021-12-27 11:37:33,496 - fba.demultiplex - INFO - Visualization method: tsne
2021-12-27 11:37:33,496 - fba.demultiplex - INFO - Loading feature count matrix: matrix_
↪featurecount.csv.gz ...
2021-12-27 11:37:34,111 - fba.demultiplex - INFO - Number of cells: 64,998
2021-12-27 11:37:34,111 - fba.demultiplex - INFO - Number of positive cells for a␣
↪feature to be included: 200
2021-12-27 11:37:34,205 - fba.demultiplex - INFO - Number of features: 8 / 8 (after␣
↪filtering / original in the matrix)
2021-12-27 11:37:34,205 - fba.demultiplex - INFO - Features: BatchA BatchB BatchC BatchD␣
↪BatchE BatchF BatchG BatchH
2021-12-27 11:37:34,206 - fba.demultiplex - INFO - Total UMIs: 17,021,991 / 17,021,991
2021-12-27 11:37:34,254 - fba.demultiplex - INFO - Median number of UMIs per cell: 63.0 /
↪ 63.0
2021-12-27 11:37:34,254 - fba.demultiplex - INFO - Demultiplexing ...
2021-12-27 11:37:48,810 - fba.demultiplex - INFO - Generating heatmap ...
2021-12-27 11:38:10,642 - fba.demultiplex - INFO - Embedding ...
2021-12-27 11:38:54,942 - fba.__main__ - INFO - Done.
```

Heatmap of the relative abundance of features (HTOs) across all cells. Each column represents a single cell.



t-SNE embedding of cells based on the abundance of features (HTOs, no transcriptome information used). Colors indicate the HTO status for each cell, as called by FBA. This is a re-creation of Fig. 1d in Stoeckius, M., et al. (2018).

Preview the demultiplexing result: the numbers of singlets.

```
In [1]: import pandas as pd

In [2]: m = pd.read_csv("demultiplexed/matrix_cell_identity.csv.gz", index_col=0)

In [3]: m.loc[:, m.sum(axis=0) == 1].sum(axis=1)
Out[3]:
BatchA    2618
BatchB    2979
BatchC    2648
BatchD    2368
BatchE    2198
BatchF    2201
BatchG    2810
BatchH    2721
dtype: int64
```

- *Peripheral Blood Mononuclear Cells with 8 Antibodies*

3.7 MULTI-seq

3.7.1 15k HEK293 and 40k HMECs Multiplexed by Lipid- and Cholesterol-tagged Indices

Dataset: MULTI-seq: sample multiplexing for single-cell RNA sequencing using lipid-tagged indices

McGinnis, C.S., Patterson, D.M., Winkler, J., Conrad, D.N., Hein, M.Y., Srivastava, V., Hu, J.L., Murrow, L.M., Weissman, J.S., Werb, Z., et al. (2019). MULTI-seq: sample multiplexing for single-cell RNA sequencing using lipid-tagged indices. *Nat. Methods* **16**, 619–626.

Preparation

Download pre-processed feature count matrix files from Gene Expression Omnibus.

```
$ wget https://ftp.ncbi.nlm.nih.gov/geo/series/GSE129nnn/GSE129578/suppl/GSE129578_
↪processed_data_files.csv.tar.gz

$ tar zxvf GSE129578_processed_data_files.csv.tar.gz

$ ls -1

GSE129578_processed_data_files.csv.tar.gz
HMEC_orig_MULTI_matrix.csv
HMEC_techrep_MULTI_matrix.csv
PDX_MULTI_matrix.csv
POC_MULTI_matrix.csv
POC_nuc_MULTI_matrix.csv
```

15k human embryonic kidney 293 cells (HEK293)

Pre-processing

Inspect feature count matrix.

```
$ head POC_MULTI_matrix.csv

CellID,LaneID,Bar1,Bar2,Bar3,nUMI_Bar
AAACCTGAGAAGGTTT-1,LMO,1,35,0,36
AAACCTGAGATGTTAG-1,LMO,3,1,4,8
AAACCTGAGGAATGGA-1,LMO,257,0,0,257
AAACCTGAGGGATCTG-1,LMO,2,0,1,3
AAACCTGAGGTAAACT-1,LMO,98,0,0,98
AAACCTGAGTGTTAGA-1,LMO,87,0,0,87
```

(continues on next page)

(continued from previous page)

```
AAACCTGCACACAGAG-1,LMO,114,1,2,117
AAACCTGCACAGCCCA-1,LMO,1,1,25,27
AAACCTGCATCCAACA-1,LMO,4,8,0,12
```

Pre-process feature count matrix.

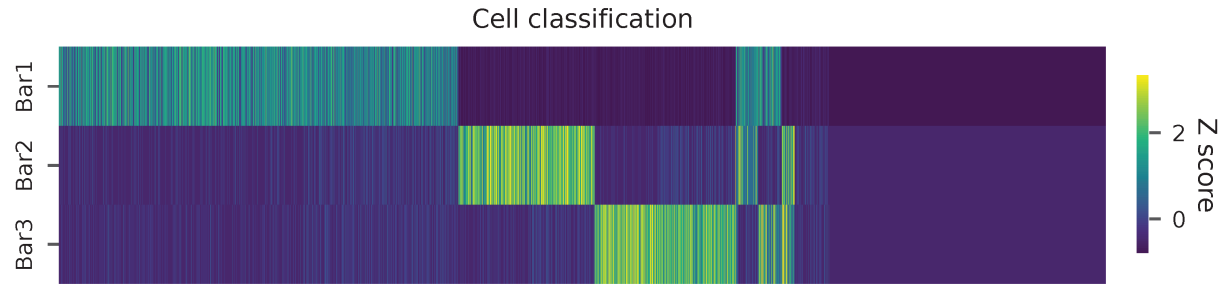
```
In [1]: import numpy as np

In [2]: import pandas as pd

In [3]: m = pd.read_csv(filepath_or_buffer="POC_MULTI_matrix.csv",
                        index_col=0)

In [4]: m.iloc[1:5, 1:5]
Out[4]:
                    Bar1  Bar2  Bar3  nUMI_Bar
CellID
AAACCTGAGATGTTAG-1     3     1     4         8
AAACCTGAGGAATGGA-1   257     0     0       257
AAACCTGAGGGATCTG-1     2     0     1         3
AAACCTGAGGTAAACT-1    98     0     0        98

In [5]: m = m.T.drop(["LaneID", "nUMI_Bar"])

In [6]: m.to_csv(path_or_buf="matrix_featurecount_POC_MULTI.csv.gz",
                 compression="infer")

In [7]: m.sum(axis=0)
Out[7]:
CellID
AAACCTGAGAAGGTTT-1     36
AAACCTGAGATGTTAG-1      8
AAACCTGAGGAATGGA-1    257
AAACCTGAGGGATCTG-1      3
AAACCTGAGGTAAACT-1     98
                      ...
TTTGTCACAAGCCTAT-3      0
TTTGTCAGTATAGTAG-3      0
TTTGTCAGTCTGATCA-3      0
TTTGTCAGTGCGCTTG-3      0
TTTGTCAGTGGTCCGT-3      0
Length: 15482, dtype: object

In [8]: np.median(m.sum(axis=0))
Out[8]: 20.0
```

Demultiplexing

Cells are demultiplexed based on the abundance of features. Demultiplexing method 4 is implemented based on the method described in McGinnis, C., et al. (2019) with some modifications. A cell identity matrix is generated in the output directory: 0 means negative, 1 means positive. To generate visualization plots, set -v.

```
$ fba demultiplex -i matrix_featurecount_POC_MULTI.csv.gz -dm 4 -v

2021-12-20 14:54:45,248 - fba.__main__ - INFO - fba version: 0.0.x
2021-12-20 14:54:45,248 - fba.__main__ - INFO - Initiating logging ...
2021-12-20 14:54:45,248 - fba.__main__ - INFO - Python version: 3.9
2021-12-20 14:54:45,249 - fba.__main__ - INFO - Using demultiplex subcommand ...
2021-12-20 14:54:47,474 - fba.__main__ - INFO - Skipping arguments: "-q/--quantile", "-
↪cm/--clustering_method", "-p/--prob"
2021-12-20 14:54:47,474 - fba.demultiplex - INFO - Output directory: demultiplexed
2021-12-20 14:54:47,474 - fba.demultiplex - INFO - Demultiplexing method: 4
2021-12-20 14:54:47,474 - fba.demultiplex - INFO - UMI normalization method: clr
2021-12-20 14:54:47,474 - fba.demultiplex - INFO - Visualization: On
2021-12-20 14:54:47,474 - fba.demultiplex - INFO - Visualization method: tsne
2021-12-20 14:54:47,474 - fba.demultiplex - INFO - Loading feature count matrix: matrix_
↪featurecount_POC_MULTI.csv.gz ...
2021-12-20 14:54:48,677 - fba.demultiplex - INFO - Number of cells: 15,482
2021-12-20 14:54:48,677 - fba.demultiplex - INFO - Number of positive cells for a␣
↪feature to be included: 200
2021-12-20 14:54:48,701 - fba.demultiplex - INFO - Number of features: 3 / 3 (after␣
↪filtering / original in the matrix)
2021-12-20 14:54:48,701 - fba.demultiplex - INFO - Features: Bar1 Bar2 Bar3
2021-12-20 14:54:48,701 - fba.demultiplex - INFO - Total UMIs: 705,913 / 705,913
2021-12-20 14:54:48,713 - fba.demultiplex - INFO - Median number of UMIs per cell: 20.0 /
↪ 20.0
2021-12-20 14:54:48,713 - fba.demultiplex - INFO - Demultiplexing ...
2021-12-20 14:54:52,347 - fba.demultiplex - INFO - Generating heatmap ...
2021-12-20 14:54:54,168 - fba.demultiplex - INFO - Embedding ...
2021-12-20 14:55:12,277 - fba.__main__ - INFO - Done.
```

Heatmap of the relative abundance of features across all cells. Each column represents a single cell.



Preview the demultiplexing result: the numbers of singlets, multiplets and negative cells.

```
In [1]:  import pandas as pd

In [2]: m = pd.read_csv("demultiplexed/matrix_cell_identity.csv.gz", index_col=0)
```

(continues on next page)

(continued from previous page)

```
In [3]: m.loc[:, m.sum(axis=0) == 1].sum(axis=1)
Out[3]:
Bar1    5909
Bar2    2016
Bar3    2083
dtype: int64

In [4]: sum(m.sum(axis=0) > 1)
Out[4]: 875

In [5]: sum(m.sum(axis=0) == 0)
Out[5]: 4599
```

40k primary human mammary epithelial cells (HMECs)

Pre-processing

Inspect feature count matrix.

```
$ head HMEC_orig_MULTI_matrix.csv

CellID,Bar1,Bar2,Bar3,Bar4,Bar5,Bar6,Bar7,Bar8,Bar9,Bar10,Bar11,Bar13,Bar15,Bar18,Bar20,
↪Bar22,Bar23,Bar24,Bar25,Bar27,Bar28,Bar29,Bar31,Bar32,Bar33,Bar34,Bar35,Bar36,Bar37,
↪Bar39,Bar40,Bar41,Bar42,Bar43,Bar44,Bar45,Bar46,Bar47,Bar48,Bar49,Bar51,Bar53,Bar54,
↪Bar55,Bar58,Bar59,Bar60,Bar61,Bar63,Bar65,Bar66,Bar67,Bar69,Bar70,Bar71,Bar72,Bar73,
↪Bar75,Bar76,Bar77,Bar78,Bar80,Bar81,Bar82,Bar83,Bar84,Bar85,Bar88,Bar89,Bar90,Bar91,
↪Bar92,Bar93,Bar94,Bar95,Bar96,nUMI_Bar
AAACCTGAGAAACGAG-1,1,3,5,23,1,3,0,1,6,11,4,4655,7,2,1,3,6,4,0,1,4,4,0,2,3,3,2,0,2,6,4,3,
↪1,0,2,3,5,5,4,0,3,2,1,0,0,2,1,1,1,0,2,0,7,7,2,2,3,2,15,35,0,0,3,9,4,1,3,3,1,1,0,0,2,0,
↪2,0,4907
AAACCTGAGAATGTGT-1,1,1,2,12,3,4,0,0,6,6,7,2,5,2,0,4,198,2,4,9,11,6,0,1,2,4,2,0,1,0,4,0,0,
↪0,0,2,6,0,2,1,8,2,0,0,0,0,0,1,0,89,14,0,2,41,1,2,2,3,6,3,4,0,3,3,1,1,0,3,0,0,1,0,1,8,1,
↪2,539
AAACCTGAGACCCACC-1,0,0,4,6,2,2,8,0,2,5,1,0,6,1,0,4,492,2,0,1,2,3,1,1,2,0,3,0,1,0,2,2,0,1,
↪1,1,3,2,1,0,1,1,0,0,0,2,0,2,0,0,1,0,0,0,1,2,0,0,1,1,0,0,2,3,0,1,0,0,76,0,1,1,0,1,3,3,
↪671
AAACCTGAGGAACTGC-1,1,0,3,7,4,5,2,0,5,5,5,1,2,1,2,2550,3,4,0,4,5,62,1,4,6,2,4,1,2,0,6,37,
↪8,0,2,1,34,2,1,0,3,1,0,0,0,1,1,0,2,4,3,0,3,0,5,0,3,12,4,3,3,0,4,5,4,0,2,1,2,11,1,0,3,0,
↪2,0,2866
AAACCTGCAACTGGCC-1,0,1,3,3,4,5,0,1,1,11,4,1,6,1,3,25,8,1,1,8,6,9,2,6,2,3,2,1,1,4,3,1,4,1,
↪6,1,29,4,9,0,7,21,1,0,0,2,1,1,3,1,1,2,2,5,5,26,6,3,13,15,4,2,3,4,18,1,0,2,1,3,0,1,4,20,
↪1,0,372
AAACCTGCAGATTGCT-1,1,0,2,5,2,4,1,1,8,6,4,0,4,0,1,7,2,4,0,2,1,8,0,0,15,3,2,4,1,2,3,3,1,2,
↪0,3,3,14,3,0,2,3,0,0,0,0,2,515,0,2,2,0,1,1,2,2,3,3,5,0,1,2,1,3,11,0,3,3,0,0,0,1,1,0,4,
↪3,715
AAACCTGCAGGGCATA-1,0,4,11,10,3,4,4,2,4,23,4,2,11,3,1,17,6,4,8,7,6,16,2,3,23,4,6,5,3,8,4,
↪4,5,2,10,7,29,3,3,0,4,2,3,0,0,1,0,7,0,3,2,2,1,6,6,0,0,9,12,7,2398,3,6,6,9,6,1,8,9,4,1,
```

(continues on next page)

(continued from previous page)

```
↪758,4,8,7,0,3570
AAACCTGCATACGCCG-1,0,2,2,11,1,2,1,0,6,7,1,0,3,1655,1,3,1,3,2,10,7,5,0,0,1,1,2,0,1,1,4,1,
↪10,0,0,2,4,2,0,0,0,0,2,1,1,1,1,0,0,5,0,2,4,0,7,1,4,1,4,2,3,2,1,2,3,1,2,6,2,2,0,1,1,59,
↪3,0,1910
AAACCTGCATCACAAC-1,0,1,10,4,1,7,0,0,3,5,0,1,2,1,207,10,5,2,3,3,8,3,1,1,3,1,4,2,4,0,1,1,5,
↪0,1,0,5,3,12,0,3,0,1,1,2,1,1,3,2,2,0,0,0,1,0,1,2,3,6,2,4,0,1,6,2,1,3,1,4,0,1,1,6,4,2,0,
↪390
```

Pre-process feature count matrix.

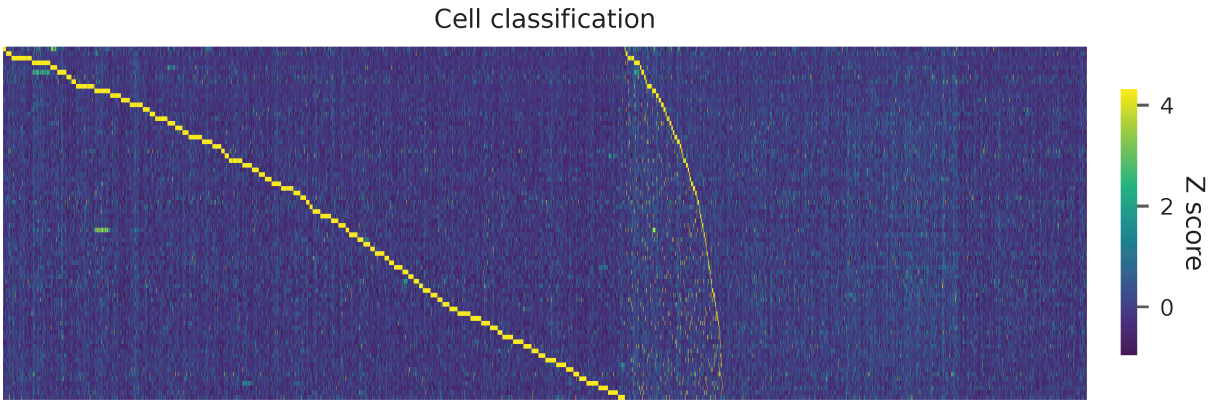
```
In [1]: import numpy as np

In [2]: import pandas as pd

In [3]: m = pd.read_csv(filepath_or_buffer="HMEC_orig_MULTI_matrix.csv",
                        index_col=0)

In [4]: m = m.T.drop(['nUMI_Bar'])

In [5]: m.to_csv(path_or_buf="matrix_featurecount_HMEC_MULTI.csv.gz", compression="infer
↪")

In [6]: m.sum(axis=0)
Out[6]:
CellID
AAACCTGAGAAACGAG-1     4905
AAACCTGAGAATGTGT-1      512
AAACCTGAGACCCACC-1      665
AAACCTGAGGAACTGC-1     2865
AAACCTGCAACTGGCC-1      360
                       ...
TTTGTCATCGAATGGG-3     3436
TTTGTCATCGGAGCAA-3      662
TTTGTCATCGGATGTT-3      152
TTTGTCATCTGATTCT-3    27223
TTTGTCATCTGCCAGG-3      256
Length: 40009, dtype: int64

In [7]: np.median(m.sum(axis=0))
Out[7]: 1241.0

In [8]: m.shape
Out[8]: (76, 40009)
```

Demultiplexing

Cells are demultiplexed based on the abundance of features. Demultiplexing method 4 is implemented based on the method described in McGinnis, C., et al. (2019) with some modifications. A cell identity matrix is generated in the output directory: 0 means negative, 1 means positive. To generate visualization plots, set -v.

```
$ fba demultiplex -i matrix_featurecount_HMEC_MULTI.csv.gz -dm 4 -v

2021-12-20 16:31:12,889 - fba.__main__ - INFO - fba version: 0.0.x
2021-12-20 16:31:12,889 - fba.__main__ - INFO - Initiating logging ...
2021-12-20 16:31:12,889 - fba.__main__ - INFO - Python version: 3.9
2021-12-20 16:31:12,889 - fba.__main__ - INFO - Using demultiplex subcommand ...
2021-12-20 16:31:15,503 - fba.__main__ - INFO - Skipping arguments: "-q/--quantile", "-
↪cm/--clustering_method", "-p/--prob"
2021-12-20 16:31:15,503 - fba.demultiplex - INFO - Output directory: demultiplexed
2021-12-20 16:31:15,503 - fba.demultiplex - INFO - Demultiplexing method: 4
2021-12-20 16:31:15,503 - fba.demultiplex - INFO - UMI normalization method: clr
2021-12-20 16:31:15,503 - fba.demultiplex - INFO - Visualization: On
2021-12-20 16:31:15,503 - fba.demultiplex - INFO - Visualization method: tsne
2021-12-20 16:31:15,503 - fba.demultiplex - INFO - Loading feature count matrix: matrix_
↪featurecount_HMEC_MULTI.csv.gz ...
2021-12-20 16:31:23,363 - fba.demultiplex - INFO - Number of cells: 40,009
2021-12-20 16:31:23,363 - fba.demultiplex - INFO - Number of positive cells for a␣
↪feature to be included: 200
2021-12-20 16:31:23,430 - fba.demultiplex - INFO - Number of features: 76 / 76 (after␣
↪filtering / original in the matrix)
2021-12-20 16:31:23,430 - fba.demultiplex - INFO - Features: Bar1 Bar2 Bar3 Bar4 Bar5␣
↪Bar6 Bar7 Bar8 Bar9 Bar10 Bar11 Bar13 Bar15 Bar18 Bar20 Bar22 Bar23 Bar24 Bar25 Bar27␣
↪Bar28 Bar29 Bar31 Bar32 Bar33 Bar34 Bar35 Bar36 Bar37 Bar39 Bar40 Bar41 Bar42 Bar43␣
↪Bar44 Bar45 Bar46 Bar47 Bar48 Bar49 Bar51 Bar53 Bar54 Bar55 Bar58 Bar59 Bar60 Bar61␣
↪Bar63 Bar65 Bar66 Bar67 Bar69 Bar70 Bar71 Bar72 Bar73 Bar75 Bar76 Bar77 Bar78 Bar80␣
↪Bar81 Bar82 Bar83 Bar84 Bar85 Bar88 Bar89 Bar90 Bar91 Bar92 Bar93 Bar94 Bar95 Bar96
2021-12-20 16:31:23,432 - fba.demultiplex - INFO - Total UMIs: 154,135,306 / 154,135,306
2021-12-20 16:31:23,462 - fba.demultiplex - INFO - Median number of UMIs per cell: 1,241.
↪0 / 1,241.0
2021-12-20 16:31:23,462 - fba.demultiplex - INFO - Demultiplexing ...
2021-12-20 16:33:51,278 - fba.demultiplex - INFO - Generating heatmap ...
2021-12-20 16:35:49,956 - fba.demultiplex - INFO - Embedding ...
2021-12-20 16:36:58,791 - fba.__main__ - INFO - Done.
```

Heatmap of the relative abundance of features across all cells. Each column represents a single cell.



t-SNE embedding of cells based on the abundance of features (no transcriptome information used). Colors indicate the index status for each cell, as called by FBA. This is a re-creation of Fig. 2a in McGinnis, C., et al. (2019).

Preview the demultiplexing result: the numbers of singlets, multiplets and negative cells.

```
In [1]: import numpy as np

In [2]: import pandas as pd

In [3]: pd.options.display.max_rows = 999

In [4]: m = pd.read_csv("demultiplexed/matrix_cell_identity.csv.gz", index_col=0)

In [5]: m.loc[:, m.sum(axis=0) == 1].sum(axis=1)
Out[5]:
Bar1     121
Bar2     201
Bar3     745
Bar4     681
Bar5     257
Bar6     346
Bar7     181
Bar8     169
Bar9     687
Bar10    572
Bar11    395
Bar13    334
Bar15    472
Bar18    262
Bar20    195
Bar22    456
Bar23    279
Bar24    276
Bar25    226
Bar27    503
Bar28    378
Bar29    324
Bar31    123
```

(continues on next page)

(continued from previous page)

```
Bar32    165
Bar33    488
Bar34    358
Bar35    260
Bar36    222
Bar37    247
Bar39    348
Bar40    453
Bar41    252
Bar42    215
Bar43    134
Bar44    103
Bar45    297
Bar46    391
Bar47    268
Bar48    276
Bar49    122
Bar51    316
Bar53    211
Bar54    254
Bar55    164
Bar58    356
Bar59    172
Bar60    182
Bar61    321
Bar63    202
Bar65    207
Bar66    200
Bar67    130
Bar69    286
Bar70    258
Bar71    184
Bar72    257
Bar73    299
Bar75    478
Bar76    396
Bar77    328
Bar78    286
Bar80    174
Bar81    387
Bar82    391
Bar83    294
Bar84    250
Bar85    263
Bar88    400
Bar89    360
Bar90    277
Bar91    213
Bar92    259
Bar93    452
Bar94    354
Bar95    366
```

(continues on next page)

(continued from previous page)

```
Bar96    255
dtype: int64

In [6]: np.median(m.loc[:, m.sum(axis=0) == 1].sum(axis=1))
Out[6]: 276.0

In [7]: sum(m.sum(axis=0) > 1)
Out[7]: 3597

In [8]: sum(m.sum(axis=0) == 0)
Out[8]: 13448
```

- *15k HEK293 and 40k HMECs Multiplexed by Lipid- and Cholesterol-tagged Indices*

3.8 Targeted transcript enrichment

3.8.1 Hodgkin's Lymphoma, Dissociated Tumor: Targeted, Gene Signature Panel

Dataset: Hodgkin's Lymphoma, Dissociated Tumor: Targeted, Gene Signature Panel

The detailed description of this dataset can be found here.

Preparation

Download fastq files.

```
$ wget https://cf.10xgenomics.com/samples/cell-exp/4.0.0/Targeted_NGSC3_DI_
↪HodgkinsLymphoma_GeneSignature/Targeted_NGSC3_DI_HodgkinsLymphoma_GeneSignature_fastqs.
↪tar

$ tar xvf Targeted_NGSC3_DI_HodgkinsLymphoma_GeneSignature_fastqs.tar
```

Download cell barcode info.

These are the cell-associated barcodes determined in the parent Hodgkin's Lymphoma, Dissociated Tumor: Whole Transcriptome Analysis dataset.

```
$ wget https://cf.10xgenomics.com/samples/cell-exp/4.0.0/Parent_NGSC3_DI_
↪HodgkinsLymphoma/Parent_NGSC3_DI_HodgkinsLymphoma_filtered_feature_bc_matrix.tar.gz

$ tar zxvf Parent_NGSC3_DI_HodgkinsLymphoma_filtered_feature_bc_matrix.tar.gz
```

Inspect cell barcodes.

```
$ gzip -dc filtered_feature_bc_matrix/barcodes.tsv.gz | head

AAACCCACAGGTTCGC-1
AAACCCACATCGATCA-1
AAACCCACATTGAGCT-1
AAACCCAGTATTTCCT-1
AAACGAACACGGTCTG-1
AAACGAAGTGGGTATG-1
AAACGAATCCTTATCA-1
AAACGCTAGTCTTCGA-1
AAACGCTGTTCTTGTT-1
AAAGAACAGATTCGAA-1
```

Prepare feature sequences.

The `map` subcommand is designed to process secondary libraries created on top of the (whole) transcriptome assays. These libraries enrich transcripts of interest through hybridization or PCR amplification, allowing for potentially more sensitive detection. The transcripts may be from endogenous genes or ectopic constructs, such as eGFP, as long as they were captured in the parent libraries.

In this tutorial, we use the 10x Genomics' "Targeted Gene Expression" library as an example. The bait sequences serve as references for read 2 mapping. Ideally, the reference sequences should include all possible captured transcribed regions.

Read 1 contains cell barcodes and UMIs, while read 2 should contain the captured transcribed regions. The feature references should only include transcribed parts that are non-overlapping and have no introns for endogenous genes.

```
$ wget https://cf.10xgenomics.com/samples/cell-exp/4.0.0/Targeted_NGSC3_DI_
↪HodgkinsLymphoma_GeneSignature/Targeted_NGSC3_DI_HodgkinsLymphoma_GeneSignature_target_
↪panel.csv
```

Inspect feature reference info.

```
$ head Targeted_NGSC3_DI_HodgkinsLymphoma_GeneSignature_target_panel.csv

#panel_name=Human Gene Signature Panel
#panel_type=predesigned
#reference_genome=GRCh38
#reference_version=2020-A
#target_panel_file_format=1.0
gene_id,bait_seq,bait_id
ENSG00000000003,
↪AGTTGTGGACGCTCGTAAGTTTTCGGCAGTTTCCGGGGAGACTCGGGGACTCCGCGTCTCGCTCTCTGTGTTCCAATCGCCCGGTGCGGTGGTGCAGGGTC
↪ENSG00000000003|TSPAN6|1
ENSG00000000003,
↪CCCGTCTCGGAGACTGCAGACTAAACCAGTCATTACTTGTTTCAAGAGCGTTCTGCTAATCTACACTTTTATTTTCTGGATCACTGGCGTTATCCTTCTTG
↪ENSG00000000003|TSPAN6|2
ENSG00000000003,
↪GGTGAGCCTGGAGAATTACTTTTCTCTTTTAAATGAGAAGGCCACCAATGTCCCCTTCGTGCTCATTGCTACTGGTACCGTCATTATTCTTTTGGGCACCT
↪ENSG00000000003|TSPAN6|3
ENSG00000000003,
↪CCGAGCTTCTGCATGGATGCTAAAACTGTATGCAATGTTTCTGACTCTCGTTTTTTTGGTCGAACTGGTCGCTGCCATCGTAGGATTTGTTTTCAGACATG
↪ENSG00000000003|TSPAN6|4
```

Re-format.

```
$ grep -v '#' Targeted_NGSC3_DI_HodgkinsLymphoma_GeneSignature_target_panel.csv | wc -l

53720

$ cut -d',' -f1,2 Targeted_NGSC3_DI_HodgkinsLymphoma_GeneSignature_target_panel.csv |␣
↪gsed 's/,/\t/g' | grep -v '#' | head -53719 > Targeted_NGSC3_DI_HodgkinsLymphoma_
↪GeneSignature_target_panel.tsv

$ head Targeted_NGSC3_DI_HodgkinsLymphoma_GeneSignature_target_panel.tsv

ENSG00000000003␣
↪AGTTGTGGACGCTCGTAAGTTTTCGGCAGTTTCCGGGGAGACTCGGGGACTCCGCGTCTCGCTCTCTGTGTTCCAATCGCCCGGTGCGGTGGTGCAGGGTC
ENSG00000000003␣
↪CCCGTCTCGGAGACTGCAGACTAAACCAGTCATTACTTGTTTCAAGAGCGTTCTGCTAATCTACACTTTTATTTTCTGGATCACTGGCGTTATCCTTCTTG
ENSG00000000003␣
↪GGTGAGCCTGGAGAATTACTTTTCTCTTTTAAATGAGAAGGCCACCAATGTCCCCTTCGTGCTCATTGCTACTGGTACCGTCATTATTCTTTTGGGCACCT
ENSG00000000003␣
↪CCGAGCTTCTGCATGGATGCTAAAACTGTATGCAATGTTTCTGACTCTCGTTTTTTTGGTCGAACTGGTCGCTGCCATCGTAGGATTTGTTTTCAGACATG
ENSG00000000003␣
↪GAATAATTATGAGAAGGCTTTGAAGCAGTATAACTCTACAGGAGATTATAGAAGCCATGCAGTAGACAAGATCCAAAATACGTTGCATTGTTGTGGTGTCA
ENSG00000000003␣
↪AGATACTAATTATTACTCAGAAAAAGGATTTCCTAAGAGTTGCTGTAAACTTGAAGATTGTACTCCACAGAGAGATGCAGACAAAGTAAACAATGAAGGTT
ENSG00000000003␣
↪CATTATAGAGTCAGAAATGGGAGTCGTTGCAGGAATTTCCTTTGGAGTTGCTTGCTTCCAACTGATTGGAATCTTTCTCGCCTACTGCCTCTCTCGTGCCA
ENSG00000000003␣
↪GATAGTGTAACCCAATGTATCTGTGGGCCTATTCCTCTCTACCTTTAAGGACATTTAGGGTCCCCCCCTGTGAATTAGAAAGTTGCTTGGCTGGAGAACTGA
ENSG00000000003␣
↪ACCAAAAAACTACACCAGTAGGTTGATTCAATCAAGATGTATGTAGACCTAAAACTACACCAATAGGCTGATTCAATCAAGATCCGTGCTCGCAGTGGGCT
ENSG00000000003␣
↪TTTGCTATGTTCTAAGTCCACCTTCTATCCCATTCATGTTAGATCGTTGAAACCCTGTATCCCTCTGAAACACTGGAAGAGCTAGTAAATTGTAAATGAAG
```

Matrix generation

To begin with, we search all read 1 sequences against a set of reference cell-associated barcodes. You can adjust the search range with `-r1_c` (default `0,16`), and the mismatching threshold with `-cb_m` (default `1`). Next, we map read 2 sequences with correct cell barcodes (found in read 1) to provided reference sequences using an aligner such as `bwa` (default, set with `-al`, Li, H. (2013). arXiv:1303.3997.) or `bowtie2` (Langmead, B., and Salzberg, S.L. (2012). Nat. Methods 9, 357–359.). We keep only alignments with a mapping quality above a threshold, which you can set with `--mapq` (default `10`), for downstream feature counting.

For UMI deduplication, we use the UMI-tools package (Smith, T., Heger, A., and Sudbery, I. (2017). Genome Res. 27, 491–499.). You can specify the UMI starting position on read 1 with `-us` (default `16`), and the UMI length with `-ul` (default `12`). Fragments with a UMI length less than this value are discarded. The UMI deduplication method is set with `-ud` (default `directional`), and the UMI deduplication mismatch threshold is set with `-um` (default `1`).

Finally, the resulting feature count matrix can be easily imported into well-established single cell analysis packages such as Seurat and Scanpy.

```
$ fba map \
    -1 Targeted_NGSC3_DI_HodgkinsLymphoma_GeneSignature_fastqs/Targeted_NGSC3_DI_
↪HodgkinsLymphoma_GeneSignature_S1_L003_R1_001.fastq.gz \
    -2 Targeted_NGSC3_DI_HodgkinsLymphoma_GeneSignature_fastqs/Targeted_NGSC3_DI_
↪HodgkinsLymphoma_GeneSignature_S1_L003_R2_001.fastq.gz \
    -w filtered_feature_bc_matrix/barcodes.tsv.gz \
    -f Targeted_NGSC3_DI_HodgkinsLymphoma_GeneSignature_target_panel.tsv \
    -o matrix_featurecount.csv.gz \
    -r1_c 0,16 \
    -cb_m 1 \
    -al bwa \
    --mapq 10 \
    -us 16 \
    -ul 12 \
    -um 1 \
    -ud directional \
    --output_directory barcode_mapping
```

Result summary.

7.67% of total read pairs (2,405,998 of 31,372,024) contribute to the final expression matrix after UMI deduplication. Sequenced quite deep.

```
2021-02-17 23:33:59,615 - fba.__main__ - INFO - fba version: 0.0.7
2021-02-17 23:33:59,615 - fba.__main__ - INFO - Initiating logging ...
2021-02-17 23:33:59,615 - fba.__main__ - INFO - Python version: 3.7
2021-02-17 23:33:59,615 - fba.__main__ - INFO - Using map subcommand ...
2021-02-17 23:33:59,863 - fba.map - INFO - bwa version: 0.7.17
2021-02-17 23:34:02,116 - fba.map - INFO - samtools version: 1.3
2021-02-17 23:34:02,145 - fba.map - INFO - Number of reference cell barcodes: 3,394
2021-02-17 23:34:02,145 - fba.map - INFO - Read 1 coordinates to search: [0, 16)
2021-02-17 23:34:02,145 - fba.map - INFO - Cell barcode maximum number of mismatches: 1
2021-02-17 23:34:02,145 - fba.map - INFO - Read 1 maximum number of N allowed: 3
2021-02-17 23:34:02,145 - fba.map - INFO - Matching cell barcodes, read 1 ...
2021-02-17 23:47:07,994 - fba.map - INFO - number of read pairs processed: 31,372,024
2021-02-17 23:47:07,995 - fba.map - INFO - Number of read pairs w/ valid cell barcodes:␣
↪28,336,049
2021-02-17 23:47:08,024 - fba.map - INFO - Number of reference features: 1,142
2021-02-17 23:47:08,024 - fba.map - INFO - Number of threads: 56
2021-02-17 23:47:08,024 - fba.map - INFO - Aligning read 2 ...
2021-02-17 23:52:34,225 - fba.map - INFO -
[M::bwa_idx_load_from_disk] read 0 ALT contigs
[M::process] read 6222224 sequences (560000160 bp)...
[M::mem_process_seqs] Processed 6222224 reads in 1411.678 CPU sec, 87.647 real sec
[M::process] read 6222224 sequences (560000160 bp)...
[M::mem_process_seqs] Processed 6222224 reads in 484.034 CPU sec, 11.666 real sec
[M::process] read 6222224 sequences (560000160 bp)...
[M::mem_process_seqs] Processed 6222224 reads in 487.450 CPU sec, 11.070 real sec
[M::process] read 6222224 sequences (560000160 bp)...
[M::mem_process_seqs] Processed 6222224 reads in 457.438 CPU sec, 8.857 real sec
[M::process] read 3447153 sequences (310243770 bp)...
[M::mem_process_seqs] Processed 3447153 reads in 273.114 CPU sec, 8.418 real sec
[main] Version: 0.7.17-r1198-dirty
```

(continues on next page)

(continued from previous page)

```
[main] CMD: /home2/s166631/bin/bwa mem -t 56 -C barcode_mapping/feature_ref.fasta␣
↪barcode_mapping/modified.fq.gz
[main] Real time: 187.399 sec; CPU: 3189.278 sec
2021-02-17 23:52:34,226 - fba.map - INFO - Generating matrix (UMI deduplication) ...
2021-02-17 23:52:34,226 - fba.map - INFO - UMI-tools version: 1.1.1
2021-02-17 23:52:34,226 - fba.map - INFO - Mapping quality threshold: 10
2021-02-17 23:52:34,226 - fba.map - INFO - UMI starting position on read 1: 16
2021-02-17 23:52:34,226 - fba.map - INFO - UMI length: 12
2021-02-17 23:52:34,226 - fba.map - INFO - UMI-tools deduplication threshold: 1
2021-02-17 23:52:34,226 - fba.map - INFO - UMI-tools deduplication method: directional
2021-02-17 23:54:06,700 - fba.map - INFO - Number of cell barcodes detected: 3,379
2021-02-17 23:54:06,700 - fba.map - INFO - Number of features detected: 1,129
2021-02-17 23:54:06,704 - fba.map - INFO - Total UMIs after deduplication: 2,405,998
2021-02-17 23:54:06,713 - fba.map - INFO - Median number of UMIs per cell: 507.0
2021-02-17 23:54:11,085 - fba.__main__ - INFO - Done.
```

- *Hodgkin's Lymphoma, Dissociated Tumor: Targeted, Gene Signature Panel*

3.9 Pseudo-bulk

3.9.1 10k A375 Cells Transduced with (1) Non-Target and (1) Target sgRNA

Dataset: 10k A375 Cells Transduced with (1) Non-Target and (1) Target sgRNA, Dual Indexed

The detailed description of this dataset can be found here.

Preparation

Fastq files and feature barcodes are prepared as described *here*.

QC

Threshold: one mismatch

When running the `qc` subcommand, omitting the `-1` (read 1) option activates bulk mode, which is useful for designing and testing feature barcoding assays prior to conducting single cell experiments. For example, bulk mode can be used to estimate: 1) the number of reads with valid feature barcodes, which can indicate primer specificity and suggest the number of reads required for sequencing; and 2) the distribution of feature barcodes, which reflects the biological aspect of the assay design.

To specify read 2 and feature barcodes, use `-2` and `-f` options, respectively. The search range for read 2 can be controlled with `-r2_c`. In this example, a single mismatch is allowed for feature barcode matching, set by `-fb_m`. Use `-n` to specify the number of reads to analyze, with `None` indicating that all reads in the fastq file should be analyzed. By default, the distribution of detected feature barcodes is summarized in `qc/feature_barcode_frequency.csv`.

```
$ fba qc \
    -2 SC3_v3_NextGem_DI_CRISPR_10K_crispr_S1_combined_R2_001.fastq.gz \
    -f SC3_v3_NextGem_DI_CRISPR_10K_feature_ref_edited.tsv \
    -r2_c 31,51 \
    -fb_m 1 \
    -n None
```

The content of `qc/feature_barcode_frequency.csv`.

feature barcode	num_reads	percentage
NON_TARGET-1_AACGTGCTGACGATGCGGGC	59,310,228	0.6979885931379053
RAB1A-2_GCCGGCGAACCAGGAAATAG	25,662,834	0.3020114068620947

Result summary.

58.59% (84,973,062 / 145,032,428) of reads have valid feature barcodes (sgRNAs). Although the valid read ratio is 1.663170816 (NON_TARGET-1_AACGTGCTGACGATGCGGGC / RAB1A-2_GCCGGCGAACCAGGAAATAG; 59,310,228 / 25,662,834), cells transduced with them separately are mixed at 1: 1 ratio. See *here* for more details.

```
2021-02-17 16:12:35,393 - fba.__main__ - INFO - fba version: 0.0.7
2021-02-17 16:12:35,393 - fba.__main__ - INFO - Initiating logging ...
2021-02-17 16:12:35,393 - fba.__main__ - INFO - Python version: 3.7
2021-02-17 16:12:35,393 - fba.__main__ - INFO - Using qc subcommand ...
2021-02-17 16:12:35,394 - fba.__main__ - INFO - Bulk mode enabled: only feature barcodes␣
↪on reads 2 are analyzed
2021-02-17 16:12:35,394 - fba.__main__ - INFO - Skipping arguments: "-w/--whitelist", "-
↪cb_m/--cb_mismatches", "-r1_c/--read1_coordinate"
2021-02-17 16:12:35,395 - fba.qc - INFO - Number of reference feature barcodes: 2
2021-02-17 16:12:35,395 - fba.qc - INFO - Read 2 coordinates to search: [31, 51)
2021-02-17 16:12:35,395 - fba.qc - INFO - Feature barcode maximum number of mismatches: 1
2021-02-17 16:12:35,395 - fba.qc - INFO - Read 2 maximum number of N allowed: inf
2021-02-17 16:12:35,395 - fba.qc - INFO - Number of read pairs to analyze: all
2021-02-17 16:12:35,395 - fba.qc - INFO - Matching ...
2021-02-17 16:15:07,684 - fba.qc - INFO - Reads processed: 10,000,000
2021-02-17 16:17:39,083 - fba.qc - INFO - Reads processed: 20,000,000
2021-02-17 16:20:09,116 - fba.qc - INFO - Reads processed: 30,000,000
2021-02-17 16:22:38,981 - fba.qc - INFO - Reads processed: 40,000,000
2021-02-17 16:25:11,671 - fba.qc - INFO - Reads processed: 50,000,000
```

(continues on next page)

(continued from previous page)

```
2021-02-17 16:27:44,790 - fba.qc - INFO - Reads processed: 60,000,000
2021-02-17 16:30:18,110 - fba.qc - INFO - Reads processed: 70,000,000
2021-02-17 16:32:51,391 - fba.qc - INFO - Reads processed: 80,000,000
2021-02-17 16:35:24,625 - fba.qc - INFO - Reads processed: 90,000,000
2021-02-17 16:37:57,678 - fba.qc - INFO - Reads processed: 100,000,000
2021-02-17 16:40:30,706 - fba.qc - INFO - Reads processed: 110,000,000
2021-02-17 16:43:03,867 - fba.qc - INFO - Reads processed: 120,000,000
2021-02-17 16:45:37,197 - fba.qc - INFO - Reads processed: 130,000,000
2021-02-17 16:48:10,511 - fba.qc - INFO - Reads processed: 140,000,000
2021-02-17 16:49:27,662 - fba.qc - INFO - Number of reads processed: 145,032,428
2021-02-17 16:49:27,663 - fba.qc - INFO - Number of reads w/ valid feature barcodes: 84,
↪973,062
2021-02-17 16:49:27,664 - fba.__main__ - INFO - Output file: qc/feature_barcode_
↪frequency.csv
2021-02-17 16:49:27,689 - fba.__main__ - INFO - Done.
```

Threshold: two mismatches

Let's relax the threshold to allow 2 mismatches for feature barcode matching (set by `-fb_m`).

```
$ fba qc \
    -2 SC3_v3_NextGem_DI_CRISPR_10K_crispr_S1_combined_R2_001.fastq.gz \
    -f SC3_v3_NextGem_DI_CRISPR_10K_feature_ref_edited.tsv \
    -r2_c 31,51 \
    -fb_m 2 \
    -n None
```

The content of `qc/feature_barcode_frequency.csv`.

feature barcode	num_reads	percentage
NON_TARGET-1_AACGTGCTGACGATGCGGGC	66,334,740	0.6613115326075217
RAB1A-2_GCCGGCGAACCAGGAAATAG	33,973,113	0.3386884673924782

Result summary.

69.16% (100,307,853 / 145,032,428) of reads have valid feature barcodes.

```
2021-02-17 16:12:00,407 - fba.__main__ - INFO - fba version: 0.0.7
2021-02-17 16:12:00,407 - fba.__main__ - INFO - Initiating logging ...
2021-02-17 16:12:00,408 - fba.__main__ - INFO - Python version: 3.7
2021-02-17 16:12:00,408 - fba.__main__ - INFO - Using qc subcommand ...
2021-02-17 16:12:00,408 - fba.__main__ - INFO - Bulk mode enabled: only feature barcodes␣
↪on reads 2 are analyzed
2021-02-17 16:12:00,408 - fba.__main__ - INFO - Skipping arguments: "-w/--whitelist", "-
↪cb_m/--cb_mismatches", "-r1_c/--read1_coordinate"
2021-02-17 16:12:00,426 - fba.qc - INFO - Number of reference feature barcodes: 2
2021-02-17 16:12:00,426 - fba.qc - INFO - Read 2 coordinates to search: [31, 51)
```

(continues on next page)

(continued from previous page)

```
2021-02-17 16:12:00,426 - fba.qc - INFO - Feature barcode maximum number of mismatches: 2
2021-02-17 16:12:00,426 - fba.qc - INFO - Read 2 maximum number of N allowed: inf
2021-02-17 16:12:00,426 - fba.qc - INFO - Number of read pairs to analyze: all
2021-02-17 16:12:00,426 - fba.qc - INFO - Matching ...
2021-02-17 16:28:02,710 - fba.qc - INFO - Reads processed: 10,000,000
2021-02-17 16:44:07,554 - fba.qc - INFO - Reads processed: 20,000,000
2021-02-17 17:00:13,431 - fba.qc - INFO - Reads processed: 30,000,000
2021-02-17 17:16:17,034 - fba.qc - INFO - Reads processed: 40,000,000
2021-02-17 17:32:21,635 - fba.qc - INFO - Reads processed: 50,000,000
2021-02-17 17:48:26,948 - fba.qc - INFO - Reads processed: 60,000,000
2021-02-17 18:04:31,050 - fba.qc - INFO - Reads processed: 70,000,000
2021-02-17 18:20:34,413 - fba.qc - INFO - Reads processed: 80,000,000
2021-02-17 18:36:38,778 - fba.qc - INFO - Reads processed: 90,000,000
2021-02-17 18:52:44,033 - fba.qc - INFO - Reads processed: 100,000,000
2021-02-17 19:08:49,500 - fba.qc - INFO - Reads processed: 110,000,000
2021-02-17 19:24:56,356 - fba.qc - INFO - Reads processed: 120,000,000
2021-02-17 19:41:02,072 - fba.qc - INFO - Reads processed: 130,000,000
2021-02-17 19:57:09,967 - fba.qc - INFO - Reads processed: 140,000,000
2021-02-17 20:05:15,665 - fba.qc - INFO - Number of reads processed: 145,032,428
2021-02-17 20:05:15,666 - fba.qc - INFO - Number of reads w/ valid feature barcodes: 100,
↪307,853
2021-02-17 20:05:15,667 - fba.__main__ - INFO - Output file: qc/feature_barcode_
↪frequency.csv
2021-02-17 20:05:15,701 - fba.__main__ - INFO - Done.
```

3.9.2 10k 1:1 Mixture of Raji and Jurkat Cells Multiplexed

Dataset: 10k 1:1 Mixture of Raji and Jurkat Cells Multiplexed, 2 CMOs

The detailed description of this dataset can be found here.

Preparation

Fastq files and feature barcodes are prepared as described *here*.

QC

Threshold: one mismatch

When running the `qc` subcommand, omitting the `-1` (read 1) option activates bulk mode, which is useful for designing and testing feature barcoding assays prior to conducting single cell experiments. For example, bulk mode can be used to estimate: 1) the number of reads with valid feature barcodes, which can indicate primer specificity and suggest the number of reads required for sequencing; and 2) the distribution of feature barcodes, which reflects the biological aspect of the assay design.

To specify read 2 and feature barcodes, use `-2` and `-f` options, respectively. The search range for read 2 can be controlled with `-r2_c`. In this example, a single mismatch is allowed for feature barcode matching, set by `-fb_m`. Use `-n` to specify the number of reads to analyze, with `None` indicating that all reads in the fastq file should be analyzed. By default, the distribution of detected feature barcodes is summarized in `qc/feature_barcode_frequency.csv`.

```
$ fba qc \
    -2 SC3_v3_NextGem_DI_CellPlex_Jurkat_Raji_10K_1_multiplexing_capture_S1_combined_R2_
↪001.fastq.gz \
    -f SC3_v3_NextGem_DI_CRISPR_10K_feature_ref.tsv \
    -r2_c 0,15 \
    -fb_m 1 \
    -n None
```

The content of `qc/feature_barcode_frequency.csv`.

feature barcode	num_reads	percentage
CMO301_ATGAGGAATTCCTGC	132435325	0.62351149
CMO302_CATGCCAATAGAGCG	79628216	0.37489323
CMO308_CGGATTCCACATCAT	320078	0.00150694
CMO309_GTTGATCTATAACAG	6445	3.03E-05
CMO303_CCGTCGTCCAAGCAT	4047	1.91E-05
CMO304_AACGTTAATCACTCA	2199	1.04E-05
CMO307_AAGCTCGTTGGAAGA	1735	8.17E-06
CMO312_ACATGGTCAACGCTG	1508	7.10E-06
CMO306_AAGATGAGGTCTGTG	1323	6.23E-06
CMO310_GCAGGAGGTATCAAT	532	2.50E-06
CMO311_GAATCGTGATTCTTC	502	2.36E-06
CMO305_CGCGATATGGTCGGA	472	2.22E-06

Result summary.

98.3% (212,402,382 / 216,070,514) of reads have valid feature barcodes. CMO301_ATGAGGAATTCCTGC and CMO302_CATGCCAATAGAGCG are the most abundant CMOs. They account for all most all of the valid reads. Although the valid read ratio is 1.663171 (132,435,325 / 79,628,216), cells labeled with them separately are mixed at 1: 1 ratio. See *here* for more details.

```
2021-10-02 02:02:31,092 - fba.__main__ - INFO - fba version: 0.0.11
2021-10-02 02:02:31,092 - fba.__main__ - INFO - Initiating logging ...
2021-10-02 02:02:31,092 - fba.__main__ - INFO - Python version: 3.7
2021-10-02 02:02:31,092 - fba.__main__ - INFO - Using qc subcommand ...
2021-10-02 02:02:31,873 - fba.__main__ - INFO - Bulk mode enabled: only feature barcodes␣
↪on reads 2 are analyzed
2021-10-02 02:02:31,873 - fba.__main__ - INFO - Skipping arguments: "-w/--whitelist", "-
```

(continues on next page)

(continued from previous page)

```
↪cb_m/--cb_mismatches", "-r1_c/--read1_coordinate"
2021-10-02 02:02:31,875 - fba.qc - INFO - Number of reference feature barcodes: 12
2021-10-02 02:02:31,875 - fba.qc - INFO - Read 2 coordinates to search: [0, 15)
2021-10-02 02:02:31,875 - fba.qc - INFO - Feature barcode maximum number of mismatches: 1
2021-10-02 02:02:31,875 - fba.qc - INFO - Read 2 maximum number of N allowed: inf
2021-10-02 02:02:31,875 - fba.qc - INFO - Number of read pairs to analyze: all
2021-10-02 02:02:31,875 - fba.qc - INFO - Matching ...
2021-10-02 02:04:19,871 - fba.qc - INFO - Reads processed: 10,000,000
2021-10-02 02:06:06,844 - fba.qc - INFO - Reads processed: 20,000,000
2021-10-02 02:07:53,987 - fba.qc - INFO - Reads processed: 30,000,000
2021-10-02 02:09:40,854 - fba.qc - INFO - Reads processed: 40,000,000
2021-10-02 02:11:27,502 - fba.qc - INFO - Reads processed: 50,000,000
2021-10-02 02:13:14,277 - fba.qc - INFO - Reads processed: 60,000,000
2021-10-02 02:15:02,641 - fba.qc - INFO - Reads processed: 70,000,000
2021-10-02 02:16:51,149 - fba.qc - INFO - Reads processed: 80,000,000
2021-10-02 02:18:40,463 - fba.qc - INFO - Reads processed: 90,000,000
2021-10-02 02:20:30,099 - fba.qc - INFO - Reads processed: 100,000,000
2021-10-02 02:22:19,651 - fba.qc - INFO - Reads processed: 110,000,000
2021-10-02 02:24:09,364 - fba.qc - INFO - Reads processed: 120,000,000
2021-10-02 02:25:59,016 - fba.qc - INFO - Reads processed: 130,000,000
2021-10-02 02:27:48,634 - fba.qc - INFO - Reads processed: 140,000,000
2021-10-02 02:29:38,323 - fba.qc - INFO - Reads processed: 150,000,000
2021-10-02 02:31:28,018 - fba.qc - INFO - Reads processed: 160,000,000
2021-10-02 02:33:17,585 - fba.qc - INFO - Reads processed: 170,000,000
2021-10-02 02:35:07,168 - fba.qc - INFO - Reads processed: 180,000,000
2021-10-02 02:36:56,770 - fba.qc - INFO - Reads processed: 190,000,000
2021-10-02 02:38:46,487 - fba.qc - INFO - Reads processed: 200,000,000
2021-10-02 02:40:36,129 - fba.qc - INFO - Reads processed: 210,000,000
2021-10-02 02:41:42,628 - fba.qc - INFO - Number of reads processed: 216,070,514
2021-10-02 02:41:42,628 - fba.qc - INFO - Number of reads w/ valid feature barcodes: 212,
↪402,382
2021-10-02 02:41:42,629 - fba.__main__ - INFO - Output file: qc/feature_barcode_
↪frequency.csv
2021-10-02 02:41:42,645 - fba.__main__ - INFO - Done.
```

Threshold: two mismatches

Let's relax the threshold to allow 2 mismatches for feature barcode matching (set by `-fb_m`).

```
$ fba qc \
    -2 SC3_v3_NextGem_DI_CellPlex_Jurkat_Raji_10K_1_multiplexing_capture_S1_combined_R2_
↪001.fastq.gz \
    -f SC3_v3_NextGem_DI_CRISPR_10K_feature_ref.tsv \
    -r2_c 0,15 \
    -fb_m 2 \
    -n None
```

The content of `qc/feature_barcode_frequency.csv`.

feature barcode	num_reads	percentage
CMO301_ATGAGGAATTCCTGC	133957542	0.624153341
CMO302_CATGCCAATAGAGCG	80322629	0.374250203
CMO308_CGGATTCCACATCAT	323662	0.00150805
CMO309_GTTGATCTATAACAG	6498	3.03E-05
CMO303_CCGTCGTCCAAGCAT	4091	1.91E-05
CMO304_AACGTTAATCACTCA	2225	1.04E-05
CMO307_AAGCTCGTTGGAAGA	1751	8.16E-06
CMO312_ACATGGTCAACGCTG	1535	7.15E-06
CMO306_AAGATGAGGTCTGTG	1351	6.29E-06
CMO310_GCAGGAGGTATCAAT	539	2.51E-06
CMO311_GAATCGTGATTCTTC	507	2.36E-06
CMO305_CGCGATATGGTCGGA	477	2.22E-06

Result summary.

99.33% (214,622,807 / 216,070,514) of reads have valid feature barcodes.

```
2021-10-02 02:02:31,268 - fba.__main__ - INFO - fba version: 0.0.11
2021-10-02 02:02:31,268 - fba.__main__ - INFO - Initiating logging ...
2021-10-02 02:02:31,268 - fba.__main__ - INFO - Python version: 3.7
2021-10-02 02:02:31,268 - fba.__main__ - INFO - Using qc subcommand ...
2021-10-02 02:02:32,021 - fba.__main__ - INFO - Bulk mode enabled: only feature barcodes␣
↪on reads 2 are analyzed
2021-10-02 02:02:32,021 - fba.__main__ - INFO - Skipping arguments: "-w/--whitelist", "-
↪cb_m/--cb_mismatches", "-r1_c/--read1_coordinate"
2021-10-02 02:02:32,025 - fba.qc - INFO - Number of reference feature barcodes: 12
2021-10-02 02:02:32,025 - fba.qc - INFO - Read 2 coordinates to search: [0, 15)
2021-10-02 02:02:32,026 - fba.qc - INFO - Feature barcode maximum number of mismatches: 2
2021-10-02 02:02:32,026 - fba.qc - INFO - Read 2 maximum number of N allowed: inf
2021-10-02 02:02:32,026 - fba.qc - INFO - Number of read pairs to analyze: all
2021-10-02 02:02:32,026 - fba.qc - INFO - Matching ...
2021-10-02 02:13:36,407 - fba.qc - INFO - Reads processed: 10,000,000
2021-10-02 02:24:40,718 - fba.qc - INFO - Reads processed: 20,000,000
2021-10-02 02:35:43,572 - fba.qc - INFO - Reads processed: 30,000,000
2021-10-02 02:46:45,598 - fba.qc - INFO - Reads processed: 40,000,000
2021-10-02 02:57:47,743 - fba.qc - INFO - Reads processed: 50,000,000
2021-10-02 03:08:49,904 - fba.qc - INFO - Reads processed: 60,000,000
2021-10-02 03:19:52,124 - fba.qc - INFO - Reads processed: 70,000,000
2021-10-02 03:30:54,289 - fba.qc - INFO - Reads processed: 80,000,000
2021-10-02 03:41:56,459 - fba.qc - INFO - Reads processed: 90,000,000
2021-10-02 03:53:01,896 - fba.qc - INFO - Reads processed: 100,000,000
2021-10-02 04:04:07,940 - fba.qc - INFO - Reads processed: 110,000,000
2021-10-02 04:15:13,882 - fba.qc - INFO - Reads processed: 120,000,000
2021-10-02 04:26:19,716 - fba.qc - INFO - Reads processed: 130,000,000
2021-10-02 04:37:25,780 - fba.qc - INFO - Reads processed: 140,000,000
2021-10-02 04:48:31,630 - fba.qc - INFO - Reads processed: 150,000,000
2021-10-02 04:59:36,756 - fba.qc - INFO - Reads processed: 160,000,000
2021-10-02 05:10:42,247 - fba.qc - INFO - Reads processed: 170,000,000
2021-10-02 05:21:47,635 - fba.qc - INFO - Reads processed: 180,000,000
2021-10-02 05:32:53,151 - fba.qc - INFO - Reads processed: 190,000,000
2021-10-02 05:43:58,739 - fba.qc - INFO - Reads processed: 200,000,000
```

(continues on next page)

(continued from previous page)

```
2021-10-02 05:55:04,397 - fba.qc - INFO - Reads processed: 210,000,000
2021-10-02 06:01:48,423 - fba.qc - INFO - Number of reads processed: 216,070,514
2021-10-02 06:01:48,424 - fba.qc - INFO - Number of reads w/ valid feature barcodes: 214,
↪622,807
2021-10-02 06:01:48,425 - fba.__main__ - INFO - Output file: qc/feature_barcode_
↪frequency.csv
2021-10-02 06:01:48,442 - fba.__main__ - INFO - Done.
```

- *10k A375 Cells Transduced with (1) Non-Target and (1) Target sgRNA, Dual Indexed*
- *10k 1:1 Mixture of Raji and Jurkat Cells Multiplexed, 2 CMOs*
- CRISPR screening
  - *10k A375 Cells Transduced with (1) Non-Target and (1) Target sgRNA, Dual Indexed*
  - *CROP-seq; 1:1:1 Mixture of DNMT3B, MBD1, and TET2 Knockout Cell Lines (HEK293T)*
  - *Direct-capture Perturb-seq; CRISPRi-based Screen of Unfolded Protein Response (UPR) Using 3' sgRNA-CR1cs1*
- Cell surface protein labeling
  - *CITE-seq; 8k Cord Blood Mononuclear Cells with 13 Antibodies*
  - *ASAP-seq; Multiplexed CRISPR Perturbations in Primary T Cells*
  - *1k Human PBMCs Stained with a Panel of TotalSeq B Antibodies, Dual Indexed*
- ECCITE-seq
  - *6k Single-cell Multimodal Readout of NIH-3T3, MyLa, Sez4 and PBMCs*
- PHAGE-ATAC
  - *PHAGE-ATAC; Anti-CD8 Phage Hashing Single-cell ATAC-seq Using CD8 T Cells from Four Human Donors*
- CellPlex
  - *10k 1:1 Mixture of Raji and Jurkat Cells Multiplexed, 2 CMOs*
  - *30k Mouse E18 Combined Cortex, Hippocampus and Subventricular Zone Nuclei Multiplexed, 12 CMOs*
- Cell hashing
  - *Peripheral Blood Mononuclear Cells with 8 Antibodies*
- MULTI-seq
  - *15k HEK293 and 40k HMECs Multiplexed by Lipid- and Cholesterol-tagged Indices*
- Targeted transcript enrichment
  - *Hodgkin's Lymphoma, Dissociated Tumor: Targeted, Gene Signature Panel*
- Pseudo-bulk
  - *10k A375 Cells Transduced with (1) Non-Target and (1) Target sgRNA, Dual Indexed*
  - *10k 1:1 Mixture of Raji and Jurkat Cells Multiplexed, 2 CMOs*

CHAPTER

FOUR

CHANGELOG

4.1 0.0.13 (Jan 1 2023)

Added

- Multiple `-i/--input` flags support in `count` module
- Better single-cell ATAC-seq support
  - Setting `-ul/--umi_length` to 0 to disable UMI deduplication in `count` module
  - Option `-cb_rc/--cell_barcode_reverse_complement` to convert the input cell barcode sequences to reverse-complement in `regex`, `extract` and `count` modules

4.2 0.0.12 (Mar 9 2022)

Added

- Non-uniform read lengths support in `qc` module

Fixed

- Readability of the `qc` results
- Matplotlib `FixedFormatter` warning

4.3 0.0.11 (Jun 17 2021)

Changed

- Rename options `-r1_coords` and `-r2_coords` to `-r1_c` and `-r2_c`, respectively.
- Improve logging information.

CHAPTER

FIVE

ACKNOWLEDGEMENTS

`fba` is inspired by several outstanding projects:

- CITE-seq-Count[1]
- TinyFastSS
- dnaio

——— .

[1] Stoeckius, M., Zheng, S., Houck-Loomis, B., Hao, S., Yeung, B.Z., Mauck, W.M., 3rd, Smibert, P., and Satija, R. (2018). Cell Hashing with barcoded antibodies enables multiplexing and doublet detection for single cell genomics. *Genome Biol.* **19**, 224. DOI: 10.1186/s13059-018-1603-1

CHAPTER

SIX

QUICKSTART

6.1 Installation

```
$ pip install fba
```

6.2 Usage

```
$ fba

usage: fba [-h]  ...

Tools for single-cell feature barcoding analysis

optional arguments:
-h, --help         show this help message and exit

functions:

   extract         extract cell and feature barcodes
   map             map enriched transcripts
   filter          filter extracted barcodes
   count           count feature barcodes per cell
   demultiplex     demultiplex cells based on feature abundance
   qc              quality control of feature barcoding assay
   kallisto_wrapper
                   deploy kallisto/bustools for feature barcoding
                   quantification
```

CHAPTER

SEVEN

CITATION

Jialei Duan, Gary C Hon, **FBA: feature barcoding analysis for single cell RNA-Seq**, *Bioinformatics*, Volume 37, Issue 22, 15 November 2021, Pages 4266–4268. DOI: 10.1093/bioinformatics/btab375. PMID: 33999185.