May 1st, 2022

Alignment of Prox-seq sequencing data

Hoang Van Phan

Tay Lab, The University of Chicago

Alignment workflow

The alignment pipeline is used to convert raw sequencing reads (fastq.gz files) into count data of PLA products. Java 8 is required to run the pipeline. The pipeline involves up to 6 steps.

1. Antibody barcode alignment: ReadAlignment10x, ReadAlignmentDropSeq or ReadAlignmentSmartSeq.
2. Merge cell barcode (optional): BuildCellBarcodes.
3. Cell barcode correction (optional): CellBarcodeCorrection.
4. Identify true cell barcodes using the knee plot (optional): ReadcountHistogram.
5. Merge UMI: UMIMerging.
6. Extract digital count: DigitalCount.

The pipeline is designed to work with 10x data (paired end reads), Drop-seq data (paired end reads), and Smart-seq2 data (single end reads). Below is an example usage of the alignment program, and detailed descriptions of each step.

Antibody barcode alignment

For the first step, we have to extract the antibody identities, cell barcodes, and UMIs from the sequencing reads. There are currently two options, depending on whether the Drop-seq or plate-based version of Prox-seq was used.

10x:

```
java -jar PLA_alignment.jar ReadAlignment10x \
R1=read1.fastq.gz \
R2=read2.fastq.gz \
AB_BC_LIST=barcode_cocktail.csv \
O=ReadAlignment_out.txt.gz \
SUMMARY=ReadAlignment_summary.txt \
HEADER=TRUE \
BASE_QUALITY=10 \
NUM_BELOW_BASE_QUALITY=1 \
BASE_QUALITY_START=1
```

Arguments:

R1 and R2: paths to the read 1 and read 2 files (fastq.gz format). Read 1 contains the cell barcodes and the UMIs.

AB_BC_LIST: path to the comma-separated file containing the protein names (first column) and their barcode sequences (second column). The protein names must NOT contain spaces, commas or colons.
Example: CD3,AGTCGACA

O: path to store the alignment results (txt.gz format).

SUMMARY (optional): path to store the summary file (txt format). The default is the current working directory.
HEADER (optional): indicates if ABfile contains a header row. The default is TRUE.
BASE_QUALITY (optional): the Phred quality score threshold for discarding reads. The default is 10.
NUM_BELOW_BASE_QUALITY (optional): the maximum number of bases in both reads 1 and 2 that is allowed to be below BASE_QUALITY. The default is 1.
BASE_QUALITY_START (optional): the number of bases from the start of read 2 to be discarded from the BASE_QUALITY filter. If 0, then the whole read 2 is considered by the filter. The default is 1.

Drop-seq:

```
java -jar PLA_alignment.jar ReadAlignmentDropSeq \
R1=read1.fastq.gz \
R2=read2.fastq.gz \
AB_BC_LIST=barcode_cocktail.csv \
O=ReadAlignment_out.txt.gz \
SUMMARY=ReadAlignment_summary.txt \
HEADER=TRUE \
BASE_QUALITY=10 \
NUM_BELOW_BASE_QUALITY=1 \
BASE_QUALITY_START=1
```

Arguments:

Same as the arguments for ReadAlignment10x.

Plate-based (plate-based like Smart-seq2):

```
java -jar PLA_alignment.jar ReadAlignmentSmartSeq \
R1_LIST=R1_LIST.csv \
AB_BC_LIST =barcode_cocktail.csv \
O=ReadAlignment_out.txt.gz \
SUMMARY=ReadAlignment_summary.txt \
HEADER=TRUE \
BASE_QUALITY=10 \
NUM_BELOW_BASE_QUALITY=1 \
BASE_QUALITY_START=1
```

Arguments:

R1_LIST: path to the csv file, each row contains the path a single cell read 1 file and its cell's ID, separated by comma. The csv file must NOT contain a header row. If the file is made with a text editor, it must end with a newline character.
Example: /path/to/cell_1_read1.fastq.gz,cell_1
AB_BC_LIST, O, SUMMARY, HEADER, BASE_QUALITY: identical to ReadAlignmentDropSeq.
NUM_BELOW_BASE_QUALITY: the maximum number of bases in both read 1 that is allowed to be below BASE_QUALITY. The default is 1.
BASE_QUALITY_START (optional): the number of bases from the start of read 1 to be discarded from the BASE_QUALITY filter. If 0, then the whole read 1 is considered by the filter. The default is 1.

Cell barcode correction

This step is only required for 10x or Drop-seq data. The user either supplies a list of “correct” reference cell barcodes (CellBarcodeCorrection), either provided by Drop-seq alignment tools (output of BAMTagHistogram function) or generated using BuildCellBarcodes.

CellBarcodeCorrection

```
java -jar PLA_alignment.jar CellBarcodeCorrection \
I=ReadAlignment_out.txt.gz \
O=cellbarcode_out.txt.gz \
MODE=10X \
CELL_BC_LIST=referencebarcodes.txt.gz \
SUFFIX=NONE \
READCOUNT_CUTOFF=100 \
SUMMARY=cellbarcode_summary.txt \
HEADER=TRUE
```

Arguments:

- I: path to alignment results (output of ReadAlignmentDropSeq).
- O: path to store output file with cell barcode corrected (txt.gz format).
- MODE: whether alignment is being done for 10x or Drop-seq pipeline. Only accepts 10X or DROPSEQ values.
- CELL_BC_LIST: path to a tab-separated list of cell barcodes (second column) and their read counts (first column). This is the file produced by Drop-seq alignment tools (output of BAMTagHistogram function), by the function BuildCellBarcodes below, or by cellranger count function (barcodes.tsv.gz file in either ./outs/filtered_feature_bc_matrix/ or ./outs/raw_feature_bc_matrix/).
- SUFFIX (optional): suffix of each cell barcode, only used for 10X mode (eg, SUFFIX=-1 means cell barcode is AGTC-1). The default is NONE (ie, no suffix is added to the cell barcode)
- READCOUNT_CUTOFF (optional): the minimum read count for a reference cell barcode to be used. The default is 100.
- SUMMARY (optional): path to store summary file. The default is the current working directory.
- HEADER (optional): indicates if CELL_BC_LIST has a header row to be discarded. The default is TRUE.

BuildCellBarcodes

```
java -jar PLA_alignment.jar BuildCellBarcodes \
I=ReadAlignment_out.txt.gz \
O=cellbarcode_out.txt.gz \
SUMMARY=cellbarcode_summary.txt
```

Arguments:

- I: path to alignment results (output of ReadAlignmentDropSeq).
- O: path to store output file with cell barcode corrected.
- SUMMARY (optional): path to store summary file. The default is the current working directory.

This function builds a list of reference cell barcodes by merging all cell barcodes that are 1 Hamming distance apart, and keep the cell barcode with the highest read count. If a cell barcode matches with more than 2 cell barcodes at 1 Hamming distance, then this barcode is discarded.
NOTE: the output file is tab-separated, and has a header row just like the output of BAMTagHistogram from Drop-seq alignment tools.

Knee plot

This step is only optionally used for Drop-seq data. Here, we export the read count of each single-cell barcode.

```
java -jar PLA_alignment.jar ReadcountHistogram \
I=cellbarcode_out.txt.gz \
O=cellbarcode_readcounts.txt.gz
```

Arguments:

I: path to the cell barcode corrected file.

O: path to export the tab-separated cell barcodes (second column) and their corresponding read counts (first column) (txt.gz format).

Then, we can use the output cellbarcode_readcounts.txt.gz to plot the knee plot to identify the good single-cell barcodes (similar to Drop-seq tools).

UMI merging

Now, we have to merge PLA products coming from the same single cell with similar UMI sequences together. Here, reads with duplicated UMIs, and reads with UMIs that are 1 Hamming distance from another UMI with a higher read count are discarded. This step is performed for both Drop-seq and plate-based versions.

10x:

```
java -jar PLA_alignment.jar UMIMerging \
I=cellbarcode_out.txt.gz \
O=umi_out.txt.gz \
SUMMARY=umi_summary.txt
```

Drop-seq:

```
java -jar PLA_alignment.jar UMIMerging \
I=cellbarcode_out.txt.gz \
O=umi_out.txt.gz \
SUMMARY=umi_summary.txt
```

Plate-based:

```
java -jar PLA_alignment.jar UMIMerging \
I= ReadAlignment_out.txt.gz \
O=umi_out.txt.gz \
SUMMARY=umi_summary.txt
```

Arguments:

I: path to the input file. For Drop-seq experiments, use the output of CellBarcodeCorrection. For plate-based experiments, use the output of ReadAlignmentSmartSeq.

O: path to export the UMI merging result (txt.gz format).

SUMMARY (optional): path to store summary file. The default is the current working directory.

Export to digital count matrix

Finally, we export a count matrix containing the UMI counts of all single-cell PLA products. The count matrix format is PLA product by single cells.

10x:

```
java -jar PLA_alignment.jar DigitalCount \
I=umi_out.txt.gz \
O=count_matrix.txt.gz \
CELL_BC_LIST=NONE \
SUMMARY=count_summary.txt \
DUPLICATE_EXPORT=DigitalCountduplicate_export.txt.gz \
REMOVE_DUPLICATE=TRUE
```

Drop-seq:

```
java -jar PLA_alignment.jar DigitalCount \
I=umi_out.txt.gz \
O=count_matrix.txt.gz \
CELL_BC_LIST=knee_barcodes.txt \
HEADER=FALSE \
SUMMARY=count_summary.txt \
DUPLICATE_EXPORT=duplicate_export.txt.gz \
REMOVE_DUPLICATE=TRUE
```

Plate-based:

```
java -jar PLA_alignment.jar DigitalCount \
I=umi_out.txt.gz \
O=count_matrix.txt.gz \
CELL_BC_LIST=NONE \
SUMMARY=count_summary.txt \
DUPLICATE_EXPORT=duplicate_export.txt.gz \
REMOVE_DUPLICATE=TRUE
```

Arguments:

I: path to the UMI merging result.

O: path to store the tab-separated digital count matrix. Rows are PLA products, columns are single-cell barcodes (txt.gz format).

CELL_BC_LIST (optional): a list of chosen cell barcodes (from knee plot) (txt format). If NONE (default), export all available cell barcodes.

HEADER (optional): whether the CELL_BC_LIST has a header row. The default is FALSE.

SUMMARY (optional): path to store the summary file. The default is the current working directory.

DUPLICATE_EXPORT (optional): path to store the list of duplicated PLA products across single cells. These are the PLA products that have the same UMI in more than one cell. The default is the current working directory.

REMOVE_DUPLICATE (optional): whether to remove duplicated PLA products across single cells. The default is FALSE.