bcSeq

Fast Sequence Alignment for High-throughput shRNA and CRISPR Screens

Jiaxing Lin Jeremy Gresham Tongrong Wang So Young Kim
James Alvarez Jeffrey S. Damrauer Scott Floyd Joshua Granek
Andrew Allen Cliburn Chan Jichun Xie Kouros Owzar

2019-04-25

Outline

Introduction

This document provides comprehensive instructions and examples for using the `bcSeq` package to perform alignment of CRISPR or shRNA reads to a library of sequences. The alignment is based on a 'Trie' data structure, a tree like data structure for fast searching. The algorithm is implemented in C++, and ported to R by `Rcpp`.
Features of this package include:

1. $O(N\bar{m})$ computational complexity (where N is the number of sequence reads in the sample, and $\bar{m}$ is the average read length.)
2. Short sequences alignment that can be applied to barcode matching and similar problems.
3. Allows errors during the alignment. Supports both hamming and edit distance matching with constraints.
4. Alignment qualities are evaluated and ambiguous alignments are resolved using Bayes' classifier.
5. Support for user-defined matching probability models in evaluating the alignment quality.

Note: using the 'Trie' data structure only unique barcode sequences in the library are kept.

bcSeq_hamming

```
bcSeq_hamming(sampleFile, libFile, outFile, misMatch = 2, tMat = NULL,
    numThread = 4, count_only = TRUE, detail_info = FALSE)
```

bcSeq_hamming can be used for performing alignment using hamming distance.
Function arguments:

`sampleFile:` (string or DNAStringSet) If a string it is the sample filename and needs to be a fastq file.

`libFile:` (string or DNAStringSet) If a string it is the library filename, needs to be a fasta or fastq file. libFile and sampleFile must have the same type.

`outFile:` (string) output filename.

`misMatch:` (integer) the number of maximum mismatches or indels allowed in the alignment.

`tMat:` (two column dataframe) prior probability of a mismatch given a sequence. The first column is the prior sequence, the second column is the prior error probability. The default values of prior error probability for all different sequences are 1/3.

`numThread:` (integer) the number of threads for parallel computing. The default is 4.

bcSeq_hamming

Function arguments continued:

`count_only`: (bool) option for function returns (default is `TRUE`). If set to `FALSE`, returns a list contains a vector of read IDs, and a vector of barcode IDs, and an alignment probability matrix between all the reads and barcodes. The vectors of read/barcode IDs serve as the row and column names for the alignment probability matrix respectively. Examples of the probability matrix are provided in the example section. Note that the probability matrix is a sparse matrix.

`detail_info`: (bool) option for controlling function returns, default to be `FALSE`. If set to `TRUE`, a file contain read indexes and library indexes reads aligned will be created with filename `$(outFile).txt`.

bcSeq_hamming Return Values

Function returns:

`default:` A csv count table is created and written to user specified file. The .csv file contains two columns, the first column is the sequences of the barcodes, and the second columns is the number of reads that aligned to the barcodes. A list containing a vector of read IDs and barcode IDs is returned to R.

Extra optional returns:

`count_only = FALSE:` If set to `FALSE`, `bcSeq` will return a sparse matrix in the list containing the vectors of read IDs and barcode IDs. The rows of the matrix correspond to the vector of read IDs, and the columns correspond to the vector of barcode IDs.

bcSeq_edit

```
bcSeq_edit(sampleFile, libFile, outFile, misMatch = 2, tMat = NULL,
    numThread = 4, count_only = TRUE, userProb = NULL,
    gap_left = 2, ext_left = 1, gap_right = 2,
    ext_right = 1, pen_max = 5, detail_info = FALSE)
```

bcSeq_edit can be used for performing alignment using edit distance. Function

arguments for bcSeq_edit:

- `sampleFile`: (string or DNAStringSet) If a string it is the sample filename and needs to be a fastq file.
- `libFile`: (string or DNAStringSet) If a string it is the library filename, needs to be a fasta or fastq file. libFile and sampleFile must have the same type.
- `outFile`: (string) output filename.
- `misMatch`: (integer) the number of maximum mismatches allowed in the alignment.
- `tMat`: (two column dataframe) prior probability of a mismatch given a sequence. The first column is the prior sequence, the second column is the error rate. The default value for all prior sequences is 1/3.
- `numThread`: (integer) the number of threads for parallel computing, default to be 4.

Function arguments continued:

`count_only`: (bool) option for function returns (default is `TRUE`). If set to `FALSE`, returns a list contains a vector of read IDs, and a vector of barcode IDs, and an alignment probability matrix between all the reads and barcodes. The vectors of read/barcode IDs serve as the row and column names for the alignment probability matrix respectively. Examples of the probability matrix are provided in the example section. Note that the probability matrix is a sparse matrix.

`userProb`: (function) an vectorized R function taking three arguments: `userProb(max_pen, prob, pen_val)`, `max_pen` is the maximum penalty allowed, `prob` is a vector containing the combined match/misMatch probabilities for each unique alignment between a given read and barcode, `pen_val` is a vector containing total penalties for the reads and barcodes in the same order. `userProb` is a way for the user to control which alignments are considered. The default value is `NULL`, indicating the alignment probabilities are to be computed following the model presented in the Alignment Probability Model section and a comprehensive example is given there.

`detail_info`: (bool) option for controlling function returns, default to be `FALSE`. If set to `TRUE`, a file contain read indexes and library indexes reads aligned will be created with filename `$(outFile).txt`. Not avaliable for user-defined probablity model case.

bcSeq_edit function

Extra function arguments for tuning alignment based on edit distance:

- `gap_left`: (double) Penalty score for deleting a base for the reads.
- `ext_left`: (double) Penalty score for extending a deletion of base for the reads.
- `gap_right`: (double) Penalty score for inserting a base
- `ext_right`: (double) Penalty score for extending an insertion
- `pen_max`: (double) Maximum penalty allowed for any alignment

bcSeq_edit Return Values

Function returns:

`default:` A csv count table is created and written to user specified file. The .csv file contains two columns, the first column is the sequences of the barcodes, and the second columns is the number of reads that aligned to the barcodes. A list containing a vector of read IDs and barcode IDs is returned to R.

Extra optional returns:

`count_only = FALSE:` If set to `FALSE`, `bcSeq` will return a sparse matrix in the list containing the vectors of read IDs and barcode IDs. The rows of the matrix correspond to the vector of read IDs, and the columns correspond to the vector of barcode IDs.

Example Library and Read simulation

Generating a library fasta file:

```
lFName    <- "./libFile.fasta"
bases     <- c(rep('A', 4), rep('C',4), rep('G',4), rep('T',4))
numOfBars <- 7
Barcodes  <- rep(NA, numOfBars*2)
for (i in 1:numOfBars){
    Barcodes[2*i-1] <- paste0(">barcode_ID: ", i)
    Barcodes[2*i]   <- paste(sample(bases, length(bases)), collapse = '')
}
write(Barcodes, lFName)
```

Generating a read fastq file:

```
rFName     <- "./readFile.fastq"
numOfReads <- 8
Reads      <- rep(NA, numOfReads*4)
for (i in 1:numOfReads){
    Reads[4*i-3] <- paste0("@read_ID_",i)
    Reads[4*i-2] <- Barcodes[2*sample(1:numOfBars,1,
        replace=TRUE, prob=seq(1:numOfBars))]
    Reads[4*i-1] <- "+"
    Reads[4*i]   <- paste(rawToChar(as.raw(
        33+sample(20:30, length(bases),replace=TRUE))),
        collapse='')
}
write(Reads, rFName)
```

Alignment by bcSeq_hamming

Alignment using default mapping probability and output

```
library(Matrix)
library(bcSeq)
ReadFile <- "./readFile.fastq"
BarFile  <- "./libFile.fasta"
outFile  <- "./count.csv"
```

```
res <- bcSeq_hamming(ReadFile, BarFile, outFile, misMatch = 2,
    tMat = NULL, numThread = 4, count_only = TRUE )
res <- read.csv(outFile, header=FALSE)
```

The function writes the read counts to a .csv file, the file name can be set through argument `outFile`. There are two columns in the output .csv file, the first column is the barcode sequence, and the second column is the corresponding number of reads aligned to the barcode.

Alignment using bcSeq_hamming with Optional Return

The package also provides an option for the user to output more detailed information for the alignments. The current version can return the alignment probability between the reads and the barcodes by setting argument `count_only` to "FALSE".

```
outFile  <- "./count2.csv"
bcSeq_hamming(ReadFile, BarFile, outFile, misMatch = 2, tMat = NULL,
    numThread = 4,count_only=FALSE )
```

Alignment by bcSeq_edit

Alignment using default mapping probability and output format

```
res <- bcSeq_edit(ReadFile, BarFile, outFile, misMatch = 2,
    tMat = NULL, numThread = 4, count_only = TRUE,
    gap_left = 2, ext_left = 1, gap_right = 2, ext_right = 1,
    pen_max = 7)
res <- read.csv(outFile, header=FALSE)
res[1:3,]
```

The function writes the read counts to a .csv file, the file name can be set through argument `outFile`. There are two columns in the output .csv file, the first column is the barcode sequence, and the second column is the corresponding number of reads aligned to the barcode.

Alignment using bcSeq_edit with Optional Return

The package also provides an option for the user to output more detailed information for the alignments. The current version can return the alignment probability between the reads and the barcodes by setting argument `count_only` to "FALSE".

```
outFile  <- "./count2.csv"
```

```
bcSeq_edit(ReadFile, BarFile, outFile, misMatch = 2, tMat = NULL,
    numThread = 4, count_only = FALSE, gap_left = 2, ext_left = 1,
    gap_right = 2, ext_right = 1, pen_max = 5)
```

General Alignment Probability Model in bcSeq

Let $\mathbb{P}(\tilde{\boldsymbol{r}}^k|\boldsymbol{r}) := \mathbb{P}(\tilde{R} = \tilde{\boldsymbol{r}}^k|R = \boldsymbol{r})$ denote the probability that the read originated from barcode k, with the corresponding sequence $\tilde{\boldsymbol{r}}^k$, given that the sequence of the observed read is $\boldsymbol{r} = [r_1, \ldots, r_L]$. bcSeq models the joint probability distribution of the originating reads conditional on the corresponding observed reads, $\mathbb{P}(\tilde{\boldsymbol{r}}^k|\boldsymbol{r})$, under the assumption of conditional independence, as

$$\begin{aligned}
\mathbb{P}(\tilde{\boldsymbol{r}}^k|\boldsymbol{r}) &= \mathbb{P}(\tilde{R} = \tilde{\boldsymbol{r}}^k|R = \boldsymbol{r}) \\
&= \mathbb{P}(\tilde{R} = [\tilde{r}_1, \ldots, \tilde{r}_L]|R = [r_1, \ldots, r_L]) \\
&= \prod_{i=1}^{L} \mathbb{P}(\tilde{R}_j = \tilde{r}_j|R = [r_1, \ldots, r_L]) \\
&= \prod_{i=1}^{L} \mathbb{P}(\tilde{R}_j = \tilde{r}_j|R_j = r_j).
\end{aligned}$$

The marginal conditional probability is modeled as

$$\mathbb{P}(\tilde{R}_j = \tilde{r}_j|R_j = r_j) = \begin{cases} 1 - \epsilon_j, & \tilde{r}_j = r_j \\ \frac{\epsilon_j}{3} & \tilde{r}_j \neq r_j, \end{cases}$$

where $\epsilon_j = 10^{-q_j/10}$ is the base-calling error probability corresponding to the observed Phred score q_j.

User defined Alignment Probability

bcSeq provides two approaches for the user to defined their own alignment probability

- User-defined Marginal Conditional Probability
- User-defined $\mathbb{P}(\tilde{R}_j = \tilde{r}_j | R_j = r_j)$

User-defined Marginal Conditional Probability

Redefine marginal conditional probability:

$$\begin{aligned}
&\mathbb{P}(\tilde{R}_j = \tilde{r}_j | R_j = r_j) \\
\equiv &\mathbb{P}(\tilde{R}_j = \tilde{r}_j | R_j = [r_{j-n}, \ldots, r_j]) \\
= &\begin{cases} 1 - \epsilon_j, & \tilde{r}_j = r_j \\ f(\epsilon_j)_{r_{j-n},\ldots,r_j} & \tilde{r}_j \neq r_j, \end{cases}
\end{aligned}$$

This definition assumes that sequences before position j has influence on the error rate. The option can be activated by providing a dataframe to the argument `tMat`. The first column of this dataframe is a vector of sequences $[r_{j-n}, \ldots, r_j]$ and the second column is the corresponding value of $f(\epsilon_j)_{[r_{j-n},\ldots,r_j]}$. If a sequence is not in the dataframe the default values $\frac{\epsilon_j}{3}$ are used.

User-defined $\mathbb{P}(\tilde{R}_j = \tilde{r}_j | R_j = r_j)$

Customized $\mathbb{P}(\tilde{R}_j = \tilde{r}_j | R_j = r_j)$ are also supported by bcSeq for bcSeq_edit. The user can define a function `userProb` to compute the alignment score. `userProb` is a vectorized R function taking three arguments: `userProb(max_pen, prob, pen_val)`, `max_pen` is the maximum penalty allowed, `prob` is a vector containing the combined match/misMatch probabilities for each unique alignment between a given read and barcode, `pen_val` is a vector containing total penalties for the reads and barcodes in the same order. `userProb` is a way for the user to control which alignments are considered.

```
customizeP <- function(max_pen, prob, pen_val)
{
    prob * (1 - log(2) + log(1 + max_pen / (max_pen + pen_val) ) )
}
```

```
bcSeq_edit(sampleFile, libFile, outFile, misMatch = 2, tMat = NULL,
    numThread = 4, count_only = TRUE, userProb = comstomizeP,
    gap_left = 2, ext_left = 1, gap_right = 2,
    ext_right = 1, pen_max = 5)
```

Note: the using of user-defined $\mathbb{P}(\tilde{R}_j = \tilde{r}_j | R_j = r_j)$ can add extra computation time. As defined here, `customizeP` is equivalent to the default model.

The customized `userProb` function can also be defined using sounceCpp in a non-vectorized fashion. The function signature is Rcpp::NumericVector(double, Rcpp::NumericVector, Rcpp::NumericVector).

```
library(Rcpp)
sourceCpp(code='
#include<Rcpp.h>
using namespace Rcpp;
// [[Rcpp::export]]
NumericVector cpp_fun(double m, NumericVector prob, NumericVector pen){
    NumericVector ret;
    for(int i = 0; i < prob.size(); ++i){
        ret.push_back(prob[i] * (1 - log(2) + log(1 + pen[i]/(m + pen[i]))));
    }
    return ret;
}')
```

```
bcSeq_edit(sampleFile, libFile, outFile, misMatch = 2, tMat = NULL,
    numThread = 4, count_only = TRUE, userProb = cpp_fun,
    gap_left = 2, ext_left = 1, gap_right = 2,
    ext_right = 1, pen_max = 5)
```

Practical Alignment Example

The sequencing library used in the benchmarking analysis can be downloaded from the MAGeCK website:
`ftp://ftp.sra.ebi.ac.uk/vol1/fastq/ERR376/ERR376998/ERR376998.fastq.gz`
The reference barcode library can also be downloaded from Kosuke Yusa laboratory:
`https://www.nature.com/nbt/journal/v32/n3/extref/nbt.2800-S7.xlsx`

```
get read and library file
wget ftp://ftp.sra.ebi.ac.uk/vol1/fastq/ERR376/ERR376998/ERR376998.fastq.gz
-P ./src/
wget https://www.nature.com/nbt/journal/v32/n3/extref/nbt.2800-S7.xlsx
-P ./src
```

Process library data to fasta format

```
library(gdata)
x <- read.xls("./nbt.2800-S7.xlsx")
fName <- "./libgRNA.fasta"
size <- nrow(x)
for(i in 1:size){
    cat(">seq ID","\n",file=fName, append=TRUE)
    cat(as.character(x$gRNA.sequence[i]),"\n", file=fName, append=TRUE)}
fName <- "./libgRNA.csv"
size <- nrow(x)
for(i in 1:size){
    cat(as.character(x$gRNA.sequence[i]),"\n", file=fName, append=TRUE)}
```

Practical Alignment Example

Unzip the reads and trim the adaptor

`gunzip ERR376998.fastq.gz`

Ad hoc C++ function for trimming adaptor (`trimMAGeCK.cpp`)

```
#include <iostream>
#include <fstream>
using namespace std;
int main(int argc, char *argv[]){
    int length = 0;
    string line;
    ifstream myfile (argv[1]);
    ofstream outfile (argv[2]);
    if (myfile.is_open()){
        while ( getline (myfile,line) ){
            outfile << line <<endl;
            getline (myfile,line);
            length = line.length() - 31;
            outfile << line.substr(23, length) <<endl;
            getline (myfile,line);
            outfile << line <<endl;
            getline (myfile,line);
            outfile << line.substr(23, length) <<endl;}
        myfile.close();}
    return 0;
}
```

Practical Alignment Example

Ad hoc C++ function to remove duplicated barcodes (`uniqueBar.cpp`)

```
#include <iostream>
#include <fstream>
#include <unordered_set>
using namespace std;
int main(int argc, char *argv[]){
    int length = 0; string line;
    std::unordered_set<std::string> myset;
    ifstream myfile (argv[1]); ofstream outfile (argv[2]);
    if (myfile.is_open()){
        while ( getline (myfile,line) )
            getline (myfile,line); myset.insert(line);
        myfile.close();}
    for(const std::string& x: myset)
        outfile << ">fake␣ID␣"<<x<< endl<<x<<endl;
    return 0;
}
```

Practical Alignment Example

Compile trim code and trim adoptor

```
g++ -std=c++11 trimMAGeCK.cpp -o trim.exe
./trim.exe ERR376998.fastq ERR376998_trimed.fastq
```

Remove duplicated barcodes

```
g++ -std=c++11 uniqueBar.cpp -o uniBar.exe
./uniBar.exe ./libgRNA.fasta ./libgRNAUni.fasta
mv libgRNAUni.fasta libgRNA.fasta
```

Perform alignment

```
library(bcSeq)
readFileName  <- "ERR376998_trimed.fastq"
libFileName   <- "libgRNA.fasta"
alignedFile   <- "SampleAligned.txt"
bcSeq_hamming(readFileName, libFileName, alignedFile, misMatch = 2,
    tMat = NULL, numThread = 4, count_only = TRUE)
```

Session Information

- R version 4.2.1 (2022-06-23), `aarch64-apple-darwin20`
- Running under: `macOS Ventura 13.0`
- Matrix products: default
- BLAS: `/Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/lib/libRblas.0`
- LAPACK: `/Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/lib/libRlapack`
- Base packages: base, datasets, grDevices, graphics, methods, stats, utils
- Other packages: Matrix 1.4-1, Rcpp 1.0.9, bcSeq 1.20.0, knitr 1.39
- Loaded via a namespace (and not attached): BiocGenerics 0.44.0, Biostrings 2.66.0, GenomeInfoDb 1.34.2, GenomeInfoDbData 1.2.8, IRanges 2.32.0, RCurl 1.98-1.7, S4Vectors 0.36.0, XVector 0.38.0, bitops 1.0-7, compiler 4.2.1, crayon 1.5.1, evaluate 0.15, grid 4.2.1, highr 0.9, lattice 0.20-45, magrittr 2.0.3, stats4 4.2.1, stringi 1.7.8, stringr 1.4.0, tools 4.2.1, xfun 0.31, zlibbioc 1.44.0

```
## [1] "Start Time Tue Nov  8 00:51:50 2022"
## [1] "End Time   Tue Nov  8 00:51:59 2022"
```