Package 'bcSeq'

November 4, 2026

Type Package

Title Fast Sequence Mapping in High-Throughput shRNA and CRISPR Screens

Version 1.34.0

Date 2019-07-18

Author Jiaxing Lin [aut, cre],
Jeremy Gresham [aut],
Jichun Xie [aut],
Kouros Owzar [aut],
Tongrong Wang [ctb],
So Young Kim [ctb],
James Alvarez [ctb],
Jeffrey S. Damrauer [ctb],
Scott Floyd [ctb],
Joshua Granek [ctb],
Andrew Allen [ctb],
Cliburn Chan [ctb]

Maintainer Jiaxing Lin <jiaxing.lin@duke.edu>

Description This Rcpp-based package implements a highly efficient data structure and algorithm for performing alignment of short reads from CRISPR or shRNA screens to reference barcode library. Sequencing error are considered and matching qualities are evaluated based on Phred scores. A Bayes' classifier is employed to predict the originating barcode of a read. The package supports provision of user-defined probability models for evaluating matching qualities. The package also supports multi-threading.

License GPL (>= 2)

biocViews ImmunoOncology, Alignment, CRISPR, Sequencing, SequenceMatching, MultipleSequenceAlignment, Software, ATACSeq

URL https://github.com/jl354/bcSeq

Imports Rcpp (>= 0.12.12), Matrix, Biostrings

Depends R (>= 3.4.0)

LinkingTo Rcpp, Matrix

BugReports https://support.bioconductor.org

Suggests knitr

VignetteBuilder knitr

BuildVignettes yes

NeedsCompilation yes

git_url https://git.bioconductor.org/packages/bcSeq

git_branch RELEASE_3_23

git_last_commit c800d32

git_last_commit_date 2026-04-28

Repository Bioconductor 3.23

Date/Publication 2026-11-04

Contents

bcSeq-package 2
bcSeq_edit 4
bcSeq_hamming 7
trimRead 9
uniqueBar 10

Index **11**

`bcSeq-package` *Fast Sequence Alignment for High-throughput shRNA and CRISPR Screens*

Description

This Rcpp-based package implements highly efficient data structure and algorithms for performing the alignment of short 'CRISPR' or shRNA screens reads to library barcodes based on user specified mismatch, insert and delete tolerance. Matching qualities are then evaluated based on Phred score. A Bayers' classifier is employed to determine the originating barcode of a read. We support user-defined probability model for evaluting matching qualities as well as flexible output.The alignment also support multiple-thread to reduce the processing time in the C++ implementation.

Details

Package:	bcSeq
Type:	Package
Version:	1.5.9
Date:	2019-04-23
License:	GPL-3

Please see the example function calls below, or refer to the individual function documentation or the included vignette for more information. The package vignette serves as a tutorial for using this package.

Author(s)

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

Maintainer: Jiaxing Lin <jiaxing.lin@duke.edu>

See Also

Rcpp

Examples

```
#### Generate barcodes
lFName    <- "./libFile.fasta"
bases     <- c(rep('A', 4), rep('C',4), rep('G',4), rep('T',4))
numOfBars <- 15
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)

#### Generate reads and phred score
rFName     <- "./readFile.fastq"
numOfReads <- 100
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)

#### perform alignment
outFile  <- "./counthamming.csv"
#res <- bcSeq_hamming(rFName, lFName, outFile, misMatch = 2,
#    tMat = NULL, numThread = 4, count_only = TRUE)
outFile  <- "./countedit.csv"
#res <- bcSeq_edit(rFName, lFName, 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, userProb = NULL)
```

---

bcSeq_edit *Function to perform reads to barcode alignment tolerating edit type error distance.*

Description

This a function for aligning CRISPR barcode reads to library, or similar problems using edit distance to evaluate the distance between a read and a barcode for the error toleration.

Usage

```
bcSeq_edit(sampleFile, libFile, outFile, misMatch = 2, tMat =
NULL, numThread = 4, count_only = TRUE, gap_left = 3,
ext_left = 1, gap_right = 3, ext_right = 1, pen_max =
6, userProb = NULL, detail_info = FALSE)
```

Arguments

sampleFile	(string) sample filename, needs to be a fastq file.
libFile	(string) library filename, needs to be a fasta or fastq file.
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 error rate. The default value for all prior sequences is 1/3.
numThread	(integer) the number of threads for parallel computing, default 4.
count_only	(bool) option for controlling function returns, default to be TRUE. If set to FALSE, a list contains a alignment probability matrix between all the reads and barcodes, a read IDs vector, and barcode IDs vector will be returned. The row of the matrix is corresponding to the read IDs and the column of the matrix is associated with the barcode IDs. Examples of the probability matrix are provided in the vignettes file.
gap_left	(double) Penalty score for delete a base for the reads.
ext_left	(double) Penalty score for extending deletion of base for the reads.
gap_right	(double) Penalty score for delete a base for the barcodes.
ext_right	(double) Penalty score for extending deletion of base for the barcodes.
pen_max	(double) Max penalty allowed for a alignment.

`userProb` (function) a function to compute the alignment probability with 3 arguments `userProb(max_pen, prob, pen_val)`, `max_pen` is the max penalty allowed, `prob` is a vector the probability for match and mismatch part between a read and a barcode for all the for all the possible alignment forms (since there are multiple way to align a read to a barcode for same edit distance), `pen_val` is a vector for the value of penalty for all the possible alignment forms between a read and a barcode. The purpose of `userProb(max_pen, prob, pen_val)` is to provide a way to determine the alignment pattern and probability.

`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 file name `\$(outFile).txt`. Not available for user-defined probability model case.

Value

`default` No objects are returned to R, instead, a csv count table is created and written to files. 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.

`count_only = FALSE`
If set to `FALSE`, bcSeq will return list contains a sparse matrix for alignment probabilities, a read IDs vector, and a barcode IDs vector. The rows of the matrix are corresponding to the read IDs vector, and the columns is associated with the barcode IDs vector.

Note

The user need to perform the removing of any adapter sequence before and after the barcode for the fastq file.

Examples

```
#### Generate barcodes
lFName    <- "./libFile.fasta"
bases     <- c(rep('A', 4), rep('C',4), rep('G',4), rep('T',4))
numOfBars <- 20
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)

#### Generate reads and phred score
rFName     <- "./readFile.fastq"
numOfReads <- 800
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)

#### perform alignment
outFile  <- "./count_edit.csv"
#res <- bcSeq_edit(rFName, lFName, 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 = 7)

#### The user defined probability model function is a modeling
#### of the alignment probability and the penalty encounted
#### due to the edit distance between a read and a candidata
#### barcode. The user define probability model function serves
#### as a combined evaluation of the alignment quality by
#### considering both the alignment probability and the edit
#### distance penalty.

#### the user defined function has three arguements
#### (1) val: a double vector indicates the alignment probabilities
####          between a read and its candidate barcodes based on
####          edit distance.
#### (2) pens: a double vector indicates the penalties of edit distance
####          between a read and its candidate barcodes for the alignment
####          based on edit distance.
#### (3) m: a double scalor indicating the weight for alignment
####        probabilities and edit distance penalty to determine
####        the final alignment quality that can be used as
####        as clasifier.

#### User can also constuct a more complexed model by only keeping the
#### function signature.

#### Example function in R
useP <-function(m, val, pens) { val * (1 - log(2) + log(1 + m / (m + pens) ) ) }

#### Example function in C++(can be ported to R using Rcpp packages)
#library(Rcpp)
#cppFunction(
#'NumericVector cpp_fun(int m, NumericVector val, NumericVector pens) {
#    int n = val.size();
#    NumericVector out(n);
#    for(int i = 0; i < n; ++i) {
#        out[i] = val[i] * (1 - log(2) +
#        log(1 + m / (m + pens[i]) ) );
#    }
#    return out;
#}')
```

```
outFile  <- "./count_edit_2.csv"
#res <- bcSeq_edit(rFName, lFName, outFile, misMatch = 2,
#    tMat = NULL, numThread = 4, count_only = TRUE, userProb = useP,
#    gap_left = 2, ext_left = 1, gap_right = 2, ext_right = 1,
#    pen_max = 7)
```

---

bcSeq_hamming *Function to perform reads to barcode alignment tolerating hamming type error distance*

Description

This a function for aligning CRISPR barcode reads to library, or similar problems using hamming distance to evaluate the distance between a read and a barcode for the error toleration.

Usage

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

Arguments

sampleFile	(string) sample filename, needs to be a fastq file
libFile	(string) library filename, needs to be a fasta or fastq file.
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 error rate. The default value for all prior sequences is 1/3.
numThread	(integer) the number of threads for parallel computing, default 4.
count_only	(bool) option for function returns, default to be TRUE. If set to FALSE, a list contains a alignment probability matrix between all the reads and barcodes, a read IDs vector, and barcode IDs vector will be returned. The row of the matrix is corresponding to the read IDs and the column of the matrix is associated with the barcode IDs. Examples of the probability matrix are provided in the vignettes files.
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 file name \$(outFile).txt.

Value

`default` No objects are returned to R, instead, a csv count table is created and written to files. 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.

`count_only = FALSE`
If set to `FALSE`, bcSeq will return list contains a sparse matrix for alignment probabilities, a read IDs vector, and a barcode IDs vector. The rows of the matrix are corresponding to the read IDs vector, and the columns is associated with the barcode IDs vector.

Note

The user need to perform the removing of any adaptor sequence before and after the barcode for the fastq file.

Examples

```
#### Generate barcodes
lFName    <- "./libFile.fasta"
bases     <- c(rep('A', 4), rep('C',4), rep('G',4), rep('T',4))
numOfBars <- 40
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)

#### Generate reads and phred score
rFName     <- "./readFile.fastq"
numOfReads <- 800
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)

#### perform alignment
outFile  <- "./count_hamming.csv"
#res <- bcSeq_hamming(rFName, lFName, outFile, misMatch = 2,
#    tMat = NULL, numThread = 1, count_only = TRUE)
```

trimRead *Tool function to trim adaptor for read sequences.*

Description

This a function for triming reads with adaptor. The sequencing within [start, end] will be written to the output file.

Usage

```
trimRead(inputFile,outputFile, start, end)
```

Arguments

inputFile	(string) filename for the library sequences, needs to be a fasta or fastq file.
outputFile	(string) output filename.
start	(integer) starting position.
end	(integer) ending position.

Value

default	No objects are returned to R

Examples

```
#### Generate barcodes
lFName    <- "./libFile.fasta"
bases     <- c(rep('A', 4), rep('C',4), rep('G',4), rep('T',4))
numOfBars <- 40
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)

#### Generate reads and phred score
rFName     <- "./readFile.fastq"
numOfReads <- 800
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)

#### perform alignment
outFile  <- "./readFile_trimReaded.fastq"
trimRead(rFName, outFile, 5,15)
```

---

uniqueBar *Tool function to obtain unique barcode sequences from a library .*

Description

This a function for removing the duplicated barcodes in the library file that will be used for the bcSeq alignment.

Usage

```
uniqueBar(inputFile,outputFile)
```

Arguments

inputFile	(string) filename for the library sequences, needs to be a fasta or fastq file.
outputFile	(string) output filename.

Value

default	No objects are returned to R

Examples

```
#### Generate barcodes
lFName    <- "./libFile.fasta"
bases     <- c(rep('A', 4), rep('C',4), rep('G',4), rep('T',4))
numOfBars <- 20
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 = '')
}
Barcodes <- rbind(Barcodes, Barcodes)
write(Barcodes, lFName)

outFile  <- "./libFile_unique.fasta"
uniqueBar(lFName, outFile)
```

Index

∗ **package**
bcSeq-package, 2

bcSeq-package, 2
bcSeq_edit, 4
bcSeq_hamming, 7

Rcpp, *3*

trimRead, 9

uniqueBar, 10