Package 'bnbc'

May 1, 2026

Version 1.35.0

Title Bandwise normalization and batch correction of Hi-C data

Description Tools to normalize (several) Hi-C data from replicates.

Depends R (>= 3.5.0), methods, BiocGenerics, SummarizedExperiment, GenomicRanges

Suggests BiocStyle, knitr, rmarkdown, RUnit, BSgenome.Hsapiens.UCSC.hg19

Imports Rcpp (>= 0.12.12), IRanges, rhdf5, data.table, Seqinfo, S4Vectors, matrixStats, preprocessCore, sva, parallel, EBImage, utils

LinkingTo Rcpp

VignetteBuilder knitr

License Artistic-2.0

URL https://github.com/hansenlab/bnbc

BugReports https://github.com/hansenlab/bnbc/issues

biocViews HiC, Preprocessing, Normalization, Software

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

git_branch devel

git_last_commit 3d959e0

git_last_commit_date 2026-04-28

Repository Bioconductor 3.24

Date/Publication 2026-05-01

Author Kipper Fletez-Brant [cre, aut], Kasper Daniel Hansen [aut]

Maintainer Kipper Fletez-Brant <cafletezbrant@gmail.com>

Contents

---

bnbc-package *Bandwise normalization and batch correction of Hi-C data*

Description

Tools to normalize (several) Hi-C data from replicates.

Details

The DESCRIPTION file: This package was not yet installed at build time.

Index: This package was not yet installed at build time.

The package implements the bnbc method for normalizing Hi-C data across samples. The name is short for band-wise normalization and batch correction. The main workhorse is the bnbc function. We recommend using smoothing and library size normalization first.

The package implements the `ContactGroup` class for storing multiple Hi-C contact matrices. This is most naturally done with one object per chromosome, which is ugly.

We also have functions for applying over a `ContactGroup` (`cgApply`) and working with matrix bands `band`, `getBandIdx`.

Author(s)

Kipper Fletez-Brant [cre, aut], Kasper Daniel Hansen [aut]

Maintainer: Kipper Fletez-Brant <cafletezbrant@gmail.com>

References

Fletez-Brant et al. *Distance-dependent between-sample normalization for Hi-C experiments*. In preparation.

See Also

`bnbc`, `ContactGroup`, `band`, `cgApply`.

Examples

```
data(cgEx)
batches <- colData(cgEx)$Batch
cgEx.cpm <- logCPM(cgEx)
cgEx.smooth <- boxSmoother(cgEx, 5, mc.cores=1)
cgEx.bnbc <- bnbc(cgEx.smooth, batches, 1e7, 4e4, bstart=2, nbands=4)
```

band *Get Band*

Description

Get or set band from matrix.

Usage

```
band(mat, band.no)
band(mat, band.no) <- value
```

Arguments

mat	A matrix.
band.no	Integer specifying which matrix band. band.no = 1 retrieves the main diagonal.
value	A scalar or vector equal in length to the matrix band.

Details

A matrix band is the set of elements in a matrix from a specific off-diagonal.

Value

A matrix band in the form of a vector.

See Also

getBandIdx

Examples

```
mat <- matrix(1:9, 3, 3)
band(mat, band.no = 2)
mat
band(mat, band.no = 2) <- c(9,10)
mat

data(cgEx)
tact.1 <- contacts(cgEx)[[1]]
b2 <- band(tact.1, 2)
band(tact.1, 2) <- b2
```

bnbc *Normalize Contact Matrices with BNBC*

Description

Applies BNBC method to normalize contact matrices.

Usage

```
bnbc(cg, batch, threshold = NULL, step = NULL, qn = TRUE, nbands
= NULL, mod = NULL, mean.only = FALSE, tol = 5, bstart = 2, verbose = TRUE)
```

Arguments

cg	A ContactGroup object.
batch	A single batch indicator variable.
threshold	The maximum distance interacting loci are allowed to be separated by.
step	The step size, or the number of bases a contact matrix cell represents.
qn	Whether to apply quantile normalization on each band matrix. Defaults to TRUE.
bstart	The first band to normalize. Defaults to 2.
nbands	The last band to normalize. Defaults to `nrow(cg) - 1`.
mod	A model matrix specifying which sample information is to be preserved by ComBat. Optional.
mean.only	Whether ComBat should not correct for batch effect in the variances of band matrix rows. Defaults to FALSE, which means variances are corrected. Set to TRUE if there is only one observation per batch.
tol	The number of significant digits for which the mean value of a band matrix must be greater than 0 to be processed by ComBat.
verbose	Should the function print progress?

Details

Normalization and batch correction is performed in a band-wise manner, correcting all samples' observations of one matrix off-diagonal (which we refer to as a matrix "band") at a time. For each matrix band, we collect all samples' observations into a single matrix. We then apply quantile normalization to ensure distributional similarity across samples. Finally, we perform batch effect correction using ComBat on this matrix. Each samples' matrix band is then replaced with its corrected version. We refer to this process of Band-Wise Normalization and Batch Correction as BNBC.

This function applies BNBC to the set of contact matrices and returns a ContactGroup object with matrix bands `bstart:nbands` corrected. For those rows in the matrix bands which cannot be corrected we set all elements to 0.

Very high bands contain little data in Hi-C experiments, and we don't recommend to analyze those or apply this function to high bands, see the `nbands` argument to the function.

We recommend performing bnbc on contact matrices which have been converted to log-CPM and smoothed, see the example.

Value

A ContactGroup object for which matrix bands `bstart:nbands` have had BNBC applied.

References

Johnson, W.E., Li, C. and Rabinovic, A. *Adjusting batch effects in microarray expression data using empirical Bayes methods.* Biostatistics 2007, 8:118-127. doi:10.1093/biostatistics/kxj037

Fletez-Brant et al. *Distance-dependent between-sample normalization for Hi-C experiments*. In preparation.

See Also

ContactGroup, logCPM, boxSmoother, band

Examples

```
data(cgEx)
batches <- colData(cgEx)$Batch
cgEx.cpm <- logCPM(cgEx)
cgEx.smooth <- boxSmoother(cgEx, 5, mc.cores=1)
cgEx.bnbc <- bnbc(cgEx.smooth, batches, 1e7, 4e4, bstart=2, nbands=4)
```

---

cgApply *Apply-type methods*

Description

These functions are apply-type functions for ContactGroup objects.

Usage

```
cgApply(cg, FUN, mc.cores=1, ...)
cgBandApply(cg, FUN, nbands=NULL, mc.cores=1, bstart=2, ...)
```

Arguments

`cg`	A ContactGroup object.
`FUN`	A function to be applied. For `cgApply` this function should operate on a square matrix. For `cgBandApply` this function should operate on a band, ie. a vector (see band).
`mc.cores`	The number of cores to be used. Defaults to 1
`bstart`	The first band to apply a function to. Defaults to 2. Only applicable to `cgBandApply`.
`nbands`	The last band to apply a function to. Default is `nrow(cg) - 1`. Only applicable to `cgBandApply`.
`...`	Passed to `mclapply`.

Details

These methods make it easy to apply functions to either all contact matrices or a set of bands in all contact matrices. Both methods accept a function FUN. For cgApply, the first argument should be cg, the contact group itself. For cgBandApply, the first argument should also be cg, and the second argument should be a specific band number. Additionally, the bands to be iterated are specified through bstart:nbands: bstart indicates the starting band, and nbands indicates the last band.

Value

For cgApply, a ContactGroup object. For cgBandApply, a list whose elements are the returned value of FUN.

See Also

ContactGroup, getBandMatrix, band

Examples

```
data(cgEx)
cgEx.1 <- cgApply(cgEx, FUN=function(xx){ xx + 1 })
band.matrix.list <- cgBandApply(cgEx, FUN=getBandMatrix, bstart=2, nbands=5)
```

---

cgEx *Sample chr22 Data*

Description

This is a sample ContactGroup object representing observations on chr22 from 3 1k Genomes trios' lymphoblastoid cell lines (LCL). colData(cgEx) gives the cell line name (CellLine), the ethnicity of the individual (Population), the family (Family), the gender (Gender), the relationship of the individuals within a trio (Role), the replicate number (Tech) and each sample's batch (Batch).

These data were generated by the dilution Hi-C method using HindIII (Lieberman-Aiden et al.). Hi-C contact matrices were generated by tiling the genome into 40kb bins and counting the number of interactions between bins.

These data have undergone no preprocessing.

Format

The data is an object of class ContactGroup.

Source

Raw data are available from the 4D nucleome data portal (https://data.4dnucleome.org) under accessions 4DNESYUYFD6H, 4DNESVKLYDOH, 4DNESHGL976U, 4DNESJ1VX52C, 4DNESI2UKI7P, 4DNESTAPSPUC, 4DNES4GSP9S4, 4DNESJIYRA44, 4DNESE3ICNE1.

References

Lieberman-Aiden, E, et al. *Comprehensive mapping of long-range interactions reveals folding principles of the human genome*. Science 2009, 326:289-293. doi:10.1126/science.1181369

See Also

ContactGroup

ContactGroup-class *Class* "ContactGroup"

Description

The ContactGroup class represents a collection of contact matrices which are observations from different samples on the same set of genomic loci.

Usage

```
ContactGroup(rowData, contacts, colData)
```

Arguments

rowData	Object of class GenomicRanges equal in length to the number of rows/columns in contact matrices.
contacts	Object of class list that contains all contact matrices.
colData	Object of class DataFrame containing sample-level information.

Details

The ContactGroup class contains a set of contact matrices in the slot 'contacts'. All matrices are required to be of the same dimensionality. 'ContactGroup()' expects a list of symmetric matrices to be passed to the contstructor. Data about these contact matrices is held in two other slots. Data about the genomic loci represented in the ContactGroup is found in the 'rowData' slot as a GenomicRanges objects, and sample-level information is located in the 'colData' slot as a DataFrame.

Value

A ContactGroup object.

Methods

In the code snippets below, x is a ContactGroup object.

[`signature(x = "ContactGroup", i = "ANY", j = "ANY", drop = "ANY")`: Allows for subsetting the contact matrices through use of `i` or of samples through `j`.

colData `signature(x = "ContactGroup")`: Get sample-level information about samples in `x`

colData<- `signature(x = "ContactGroup", value = "DataFrame")`: Set sample-level information about samples in `x`. `value` is expected to be a `DataFrame` object.

dim `signature(x = "ContactGroup")`: Obtain the dimensions of a ContactGroup. Returns 2 values: one representing the number of bins in the contact matrices and another representing the number of samples.

rowData `signature(x = "ContactGroup")`: Get the GenomicRanges object describing the loci in the ContactGroup. `value` is expected to be a GenomicRanges object.

rowData<- `signature(x = "ContactGroup")`: Set the GenomicRanges object describing the bins in the ContactGroup. `value` is expected to be a GenomicRanges object.

show `signature(object = "ContactGroup")`: Method to display summary information about a ContactGroup: the number of bins, the width of the bins and the number of samples.

librarySize `signature(x = "ContactGroup")`: Method to compute the library size of each contact matrix in `x`. Library size is defined to be the sum of the upper triangle of a contact matrix.

logCPM `signature(x = "ContactGroup")`: Method to transform each contact matrix to logCPM scale.

Utilities

contacts `contacts(x)`, `contacts(x) <- value`: Method to extract the list of contact matrices from a ContactGroup. `value` is expected to be a list object.

distanceIdx `signature(cg = "ContactGroup", threshold="ANY", step="ANY")`: Method to identify which matrix bands are no more than `threshold` bins apart, where each bin represents `step` base pairs.

References

Law, C.W., Chen, Y., Shi, W. and Smyth G.K. *voom: Precision weights unlock linear model analysis tools for RNA-seq read counts.* Genome Biology 2014, 15:R29. doi:10.1186/gb2014152r29.

Examples

```
data(cgEx)

cgEx[1,]
cgEx[,1]

cd <- colData(cgEx)
colData(cgEx) <- cd

gr <- rowData(cgEx)
rowData(cgEx) <- gr

cgEx

cl <- contacts(cgEx)
contacts(cgEx) <- cl

d.idx <- distanceIdx(cgEx, 1e7, 4e4)

libs <- librarySize(cgEx)

cgEx.cpm <- logCPM(cgEx)

## below, upper.mats.list is a list of upper triangular matrices
## SampleData is a DataFrame of sample data and LociData is a GenomicRanges objec
## Not run:
  MatsList <- lapply(upper.mats.list, function(M) M[lower.tri(M)] = M[upper.tri(M)])
  cg <- ContactGroup(LociData, MatsList, SampleData)

## End(Not run)
```

cooler_methods *Methods for manipulating cooler files*

Description

These are a set of methods for working with data in cooler file format.

Usage

```
getChrIdx(chr.length, chr, step)
getChrCGFromCools(files, chr, step, index.gr, work.dir, exp.name,
                  coldata, norm.factor=NULL)
cg2bedgraph2(cg, out.dir, prefix)
```

Arguments

chr.length	The length of a chromosome.
step	The resolution of the data inside the cooler file.
files	A vector of cooler file names.
chr	The target chromosome to be read.
index.gr	A GRanges object, can be output from getChrIdx.
work.dir	Directory for saving temporary files.
exp.name	The name of the experiment, will be appended all output file names.
coldata	A data.frame or DataFrame of metadata for the ContactGroup object.
cg	A ContactGroup object.
out.dir	A directory in which individual bedgraph2 (BG2) files are to be written.
prefix	A prefix for all output files; e.g. "treatment_study_".
norm.factor	The normalization factor

Details

These methods allow for the normalization of cooler files. Users must create their own index, for which we provide getChrIdx, which is an input into getChrCGFromCools. getChrCGFromCools reads multi-resolution cooler (mcool) files directly via rhdf5 and returns a ContactGroup object. Users can then follow the standard pipeline, and save their data in bedgraph2 (BG2) format using cg2bedgraph2. Note that getChrCGFromCools expects multiple resolutions in the cooler file (mcool format).

Value

For getChrIdx a GRanges object with coordinates for each bin. For getChrCGFromCools, a ContactGroup object. There is nothing returned by cg2bedgraph2.

See Also

ContactGroup

Examples

```
## Not run:
coolerDir <- system.file("cooler", package = "bnbc")
cools <- list.files(coolerDir, pattern="cool$", full.names=TRUE)

step <- 4e4

ixns <- bnbc:::getGenomeIdx(seqlengths(BSgenome.Hsapiens.UCSC.hg19)["chr22"], step)

data(cgEx)
cool.cg <- bnbc:::getChrCGFromCools(files = cools,
chr = "chr22",
step=step,
index.gr=ixns,
work.dir="tmp.dir",
exp.name="example_case",
colData = colData(cgEx)[1:2,])
all.equal(contacts(cgEx)[[1]], contacts(cool.cg)[[1]])

## End(Not run)
```

---

getBandIdx *Get Band Indices*

Description

Get the indices corresponding to a matrix band.

Usage

```
getBandIdx(n, band.no)
```

Arguments

n	The number of rows/columns of a contact matrix
band.no	Integer specifying which matrix band. band.no = 1 retrieves the main diagonal.

Details

This function is used in subsetting contact matrices, primarily in getBandMatrix. However, users wishing to extract band matrices directly may find this useful

Value

A matrix with 2 columns and as many rows as entries in the matrix band.

See Also

ContactGroup, getBandMatrix, band

Examples

```
data(cgEx)
b2.idx <- getBandIdx(nrow(cgEx), 2)
```

getBandMatrix *Get Band Matrix*

Description

Get band matrix from ContactGroup.

Usage

```
getBandMatrix(cg, band.no=1)
```

Arguments

cg	A ContactGroup object.
band.no	Integer specifying which matrix band. band.no = 1 retrieves the main diagonal.

Details

A band matrix is a matrix whose columns are the band.no-th off-diagonal of each sample's contact matrix. If there are k samples and matrix band band.no has r entries, then the returned band matrix is of dimension r x k.

Value

A matrix with one column per sample in the ContactGroup and number of rows equal to the length of the matrix band.

See Also

ContactGroup, getBandIdx, band

Examples

```
data(cgEx)
b2 <- getBandMatrix(cgEx, 2)
```

groupZeros *Group Zero Operations*

Description

These functions find and remove rows in the set of contact matrices for which all elements in the row are 0 in all samples.

Usage

```
getGroupZeros(cg)
dropGroupZeros(cg, g0s)
```

Arguments

cg	A ContactGroup object.
g0s	A list of elements identified as group zeros.

Details

Group zeros are those rows for which all elements of the row in all samples are 0. These can impact estimation of features such as A/B compartment status and so should be removed for many analyses.

Value

A ContactGroup object with the group zeros removed from all rows in all contact matrices.

See Also

ContactGroup

Examples

```
data(cgEx)
g0s <- getGroupZeros(cgEx)
cgEx <- dropGroupZeros(cgEx, g0s)
```

---

smoothing *Smoothing Operations*

Description

These functions apply a smoothing kernel to all contact matrices in a ContactGroup object.

Usage

```
boxSmoother(cg, h, mc.cores)
gaussSmoother(cg, radius, sigma, mc.cores)
```

Arguments

cg	A ContactGroup object.
h	The desired smoother radius. Only applies to box smoother. This is an integer.
radius	The desired smoother width. Only applies to Gaussian smoother. This is an integer.
sigma	The desired smoother standard deviation. Only applies to Gaussian smoother. This is a positive number.
mc.cores	The number of cores to be used.

Details

boxSmoother applies a square smoothing kernel of radius h to all contact matrices in a ContactGroup object. Specifying radius h implies that the width of the kernel is $2 * h + 1$ matrix cells.

gaussSmoother applies a square Gaussain smoothing kernel of width radius with standard deviation sigma to all contact matrices in a ContactGroup object.

Value

A ContactGroup object is returned that contains the smoothed matrices.

See Also

ContactGroup

Examples

```
data(cgEx)
cgEx.smooth <- boxSmoother(cgEx, h=5, mc.cores=1)
cgEx.smooth <- gaussSmoother(cgEx, radius=3, sigma=0.5, mc.cores=1)
```

Index