Package: Chicago (via r-universe)

November 24, 2026

Type Package

Title CHiCAGO: Capture Hi-C Analysis of Genomic Organization

Version 1.42.0

Author Jonathan Cairns, Paula Freire Pritchett, Steven Wingett, Mikhail Spivakov

Maintainer Mikhail Spivakov <mikhail.spivakov@lms.mrc.ac.uk>

Description A pipeline for analysing Capture Hi-C data.

License Artistic-2.0

Depends R (>= 3.3.1), data.table

Imports matrixStats, MASS, Hmisc, Delaporte, methods, grDevices, graphics, stats, utils

Suggests argparser, BiocStyle, knitr, rmarkdown, PCHiCdata, testthat, GenomeInfoDb, Rsamtools, GenomicInteractions, GenomicRanges, IRanges, AnnotationHub

VignetteBuilder knitr

biocViews Epigenetics, HiC, Sequencing, Software

Config/pak/sysreqs cmake make libicu-dev libuv1-dev

Repository Bioconductor 3.24

Date/Publication 2026-11-23 22:36:22 UTC

RemoteUrl https://github.com/bioc/Chicago

RemoteRef RELEASE_3_24

RemoteSha 44f82eb0c2ce8b75c29b889271f06f7161eff80d

Contents

Chicago-package 2
cdUnitTest 3
chicagoData 4
chicagoPipeline 5
copyCD 6

defaultSettings . . . 7
estimateBrownianComponent . . . 10
estimateDistFun . . . 11
estimateTechnicalNoise . . . 12
exportResults . . . 13
getPvals . . . 15
getScores . . . 16
getSkOnly . . . 17
mergeSamples . . . 18
modifySettings . . . 19
normaliseBaits . . . 20
normaliseOtherEnds . . . 21
overlapFragWithFeatures . . . 23
peakEnrichment4Features . . . 24
plotBaits . . . 26
plotDistFun . . . 28
readAndMerge . . . 29
readSample . . . 30
setExperiment . . . 31

Index **32**

`Chicago-package` *CHiCAGO: Capture Hi-C Analysis of Genomic Organization*

Description

A pipeline for analysing Capture Hi-C data.

Details

To get started, please read the vignette: `vignette("Chicago")`

Author(s)

Jonathan Cairns, Paula Freire Pritchett, Steven Wingett, Mikhail Spivakov

Maintainer: Mikhail Spivakov - <spivakov@babraham.ac.uk>

cdUnitTest *ChicagoData object for unit testing*

Description

This data set is used for unit testing - it is too small to run all of the steps of CHiCAGO. For a toy data set that is large enough, please see the data package. (Note that cdUnitTest is a subset of those data.)

Usage

```
data("cdUnitTest")
```

Details

The data are derived from mouse ESCs. They are a subset of the object smESC (from the PCHiCdata package)

Value

A chicagoData object.

Source

Schoenfelder, S. et al. "The pluripotent regulatory circuitry connecting promoters to their long-range interacting elements." Genome research 25.4 (2015): 582-597.

See Also

smESC, chicagoData

Examples

```
data(cdUnitTest)

##modifications to cdUnitTest, ensuring it uses correct design directory
designDir <- file.path(system.file("extdata", package="Chicago"), "unitTestDesign")
cdUnitTest <- modifySettings(cd=cdUnitTest, designDir=designDir)

print(cdUnitTest)
```

chicagoData *The* chicagoData *class.*

Description

Constructor for the chicagoData class.

Usage

```
chicagoData(...)
```

Arguments

...	Arguments passed to new().

Details

While this function can be used to create a chicagoData object, most users will use the setExperiment function instead.

Value

A chicagoData object has three slots, accessed as follows:

* intData(cd) is a data.table (note: not a data.frame) that contains information about fragment pairs. * settings(cd) is a list of settings, usually set with the setExperiment() function. For more information about valid settings, please see defaultSettings. To modify the settings, use modifySettings. * params(cd) is a list of parameters. CHiCAGO estimates these automatically, as part of the pipeline.

Author(s)

Mikhail Spivakov, Jonathan Cairns, Paula Freire Pritchett

See Also

setExperiment, defaultSettings

Examples

```
cd <- chicagoData()
```

chicagoPipeline *CHiCAGO pipeline function*

Description

This function runs data through the CHiCAGO pipeline.

Usage

```
chicagoPipeline(cd, outprefix = NULL, printMemory = FALSE)
```

Arguments

cd	A chicagoData object.
outprefix	NULL, or a character string. If NULL, diagnostic plots are outputted to the current plotting device. If a character string, then pdfs will be generated for a series of diagnostic plots, in files of form "[outprefix]_[plotname].pdf". For example, outprefix="experiment1" leads to files experiment1_oeNorm.pdf, etc...
printMemory	Set to TRUE for memory diagnostics.

Details

This pipeline runs the following functions in order:

- normaliseBaits
- normaliseOtherEnds
- estimateTechnicalNoise
- estimateDistFun
- estimateBrownianNoise
- getPvals
- getScores

It does not export the output. Use exportResults for this.

Value

An object of class chicagoData.

Warning

The object intData(cd) is updated by reference. Thus, intData(cd) will be altered. See vignette for further information.

Author(s)

Mikhail Spivakov, Jonathan Cairns, Paula Freire Pritchett

See Also

exportResults

Examples

```
##Read in some raw data
filesDir <- file.path(system.file("extdata", package="Chicago"), "unitTestData")
file <- file.path(filesDir, dir(filesDir))[1]
print(file) ##we will read in this file

designDir <- file.path(system.file("extdata", package="Chicago"), "unitTestDesign")

##Add a setting specific to the unit test data! Do not use in practice!
if(!interactive()) {
  settings <- list(brownianNoise.samples=1)
} else {
  settings <- NULL
}

cd <- setExperiment(designDir=designDir, settings=settings)
cd <- readAndMerge(file, cd)
```

---

copyCD *Copy chicagoData object*

Description

Copies a chicagoData object. (Failing to use this function may mean that an object is updated by reference when its 'copy' is altered.)

Usage

```
copyCD(cd)
```

Arguments

cd	chicagoData object.

Value

chicagoData object.

Author(s)

Jonathan Cairns

Examples

```
data(cdUnitTest)
x <- copyCD(cdUnitTest)
```

`defaultSettings` *Default CHiCAGO settings*

Description

A function that gives the default settings used for a CHiCAGO experiment.

IMPORTANT: from version 1.13, the following parameters are set based on the values in .npb file header and checked for consistency with the headers of .npbp and .poe files and custom-defined settings. They should therefore be provided to the makeDesignFiles.py script, which needs to be rerun if they need to be modified:

rmapfile (only the basename is checked; inconsistent baitmapfile will only generate a warning for compatibility with publicly released older designs), minFragLen, maxFragLen, binsize, removeAdjacent, adjBait2bait.

Usage

```
defaultSettings()
```

Value

A list of the following settings:

`rmapfile` Default: NA. The location of the restriction map file; see the vignette for a description of what this file should contain.

`baitmapfile` Default: NA. The location of the bait map file; see the vignette for a description of what this file should contain.

`nperbinfile` Default: NA. See vignette.

`nbaitsperbinfile` Default: NA. See vignette.

`proxOEfile` Default: NA. See vignette.

`Ncol` Default: "N". The column in intData(cd) that contains the number of reads.

`baitmapFragIDcol` Default: 4. In the bait map file, the number of the column that specifies the fragment ID of each bait.

`baitmapGeneIDcol` Default: 5. In the bait map file, the number of the column that specifies which gene(s) are on each fragment.

`maxLBrownEst` Default: 1500000. The distance range to be used for estimating the Brownian component of the null model The parameter setting should approximately reflect the maximum distance, at which the power-law distance dependence is still observable.

`minFragLen` Default: 150. (See maxFragLen.)

`maxFragLen` Default: 40000. minFragLen and maxFragLen correspond to the limits within which we observed no clear dependence between fragment length and the numbers of reads mapping to these fragments in HindIII PCHiC data.

These parameters need to be modified when using a restriction enzyme with a different cutting frequency (such as a 4-cutter) and can also be verified by users with their datasets in each individual case. However, we note that the fragment-level scaling factors (s_i and s_j) generally incorporate the effects of fragment size, so this filtering step only aims to remove the strongest bias.

`minNPerBait` Default: 250. Minimum number of reads that a bait has to accumulate to be included in the analysis.

Reasonable numbers of per-bait reads are required for robust parameter estimation. If this value is too low, the confidence of interaction calling is reduced. If too high, too many baits may be unreasonably excluded from the analysis. If it is desirable to include baits below this threshold, we recommend decreasing this parameter and then visually examining the result bait profiles (for example, using plotBaits()).

`binsize` Default: 20000. The bin size (in bases) used when estimating the Brownian collision parameters.

The bin size should, on average, include several (~4-5) restriction fragments to increase the robustness of parameter estimation. However, using too large bins will reduce the precision of distance function estimation. Therefore, this value needs to be changed if using an enzyme with a different cutting frequency (such as a 4-cutter).

`removeAdjacent` Default: TRUE. Should fragments adjacent to baits be removed from analysis?

We remove fragments adjacent to baits by default, as the corresponding ligation products are indistinguishable from incomplete digestion. This setting however may be set to FALSE if the rmap and baitmap files represent bins over multiple fragments as opposed to fragment-level data (e.g., to address sparsity issues with low-coverage experiments).

`adjBait2bait` Default: TRUE. Should baited fragments be treated separately? Baited fragments are treated separately from the rest in estimating other end-level scaling factors (si) and technical noise levels. It is a free parameter mainly for development purposes, and we do not recommend changing it.

`tlb.filterTopPercent`

Default: 0.01. Top percent of fragments with respect to accumulated trans-counts to be filtered out in the binning procedure.

Other ends are pooled together when calculating their scaling factors and as part of technical noise estimation. Binning is performed by quantile, and for the most extreme outliers this approach is not going to be adequate. Increasing this value may potentially make the estimation for the highest-count bin more robust, but will exclude additional other ends from the analysis.

`tlb.minProxOEPerBin`

Default: 50000. Minimum pool size (i.e. minimum number of other ends per pool), used when pooling other ends together based on trans-counts.

If this parameter is set too small, then estimates will be imprecise due to sparsity issues. If this parameter is set too large, then the model becomes inflexible and

so the model fit is hindered. This parameter could be decreased in a dataset that has been sequenced to an extremely high depth. Alternatively, it may need to be decreased out of necessity, in a dataset with very few other ends - for example, the vignette decreases this setting to process the PCHiCdata package data (since these data sets span only a small subset of the genome, in each case).

`tlb.minProxB2BPerBin`
: Default: 2500. Minimum pool size, used when pooling other ends together (bait-to-bait interactions only). (See previous entry, tlb.minProxOEPerBin, for advice on setting parameter.)

`techNoise.minBaitsPerBin`
: Default: 1000. Minimum pool size, used when pooling baits together based on accumulated trans-counts. (See tlb.minProxOEPerBin for advice on setting parameter.)

`brownianNoise.samples`
: Default: 5. Number of times subsampling occurs when estimating the Brownian collision dispersion.

  Dispersion estimation from a subset of baits has an error attached. Averaging over multiple subsamples allows us to decrease this error. Increasing this number improves the precision of dispersion estimation at the expense of greater runtime.

`brownianNoise.subset`
: Default: 1000. Number of baits sampled from when estimating the Brownian noise dispersion. If set to NA, then all baits are used.

  Estimating dispersion from the entire dataset usually requires a prohibitively large amount of memory. A subset is chosen that is large enough to get a reasonably precise estimate of the dispersion, but small enough to stay in memory. A user with excess memory may wish to increase this number to further improve the estimate's precision.

`brownianNoise.seed`
: Default: NA. If not NA, then `brownianNoise.seed` is used as the random number generator seed when subsampling baits. Set this to make your analysis reproducible.

`baitIDcol`
: Default: "baitID". The name of the baitID column in intData(cd).

`otherEndIDcol`
: Default: "otherEndID". The name of the otherEndID column in intData(cd).

`otherEndLencol`
: Default: "otherEndLen". The name of the column in intData(cd) that contains the lengths of the other end fragments.

`distcol`
: Default: "distSign". The name of the column in intData(cd) that contains the genomic distance that an interaction spans.

`weightAlpha`
: Default: 34.1157346557331. This, and the following parameters, are used in the p-value weighting procedure.

`weightBeta`
: Default: -2.58688050486759

`weightGamma`
: Default: -17.1347845819659

`weightDelta`
: Default: -7.07609245521541

Author(s)

Mikhail Spivakov, Jonathan Cairns, Paula Freire Pritchett

See Also

setExperiment, modifySettings

Examples

```
s <- defaultSettings()
print(s)
```

---

estimateBrownianComponent

Estimate Brownian background component.

Description

Estimates the dispersion, and adds a a Bmean column giving the expected number of Brownian reads.

Usually, the dispersion is not calculated on the full dataset - rather, a subsample of baits is taken, and the dispersion is calculated on that. The number of baits used is taken from brownianNoise.subset (with an NA value meaning that the entire dataset is used, and no subsampling is performed).

(Note that the alias estimateBrownianNoise() is provided for back-compatibility.)

Usage

```
estimateBrownianNoise(cd)
```

Arguments

cd	A chicagoData object.

Value

An object of class chicagoData.

Warning

The object intData(x) is updated by reference. Thus, intData(cd) will be altered. See vignette for further information.

Author(s)

Mikhail Spivakov, Jonathan Cairns, Paula Freire Pritchett

See Also

chicagoPipeline

Examples

```
data(cdUnitTest)

##modifications to cdUnitTest, ensuring it uses correct design directory
designDir <- file.path(system.file("extdata", package="Chicago"), "unitTestDesign")
cdUnitTest <- modifySettings(cd=cdUnitTest, designDir=designDir)

##make cdUnitTest use the full subset of baits
cdUnitTest <- modifySettings(cd=cdUnitTest, settings=list(brownianNoise.subset=NA))

cdUnitTest <- estimateBrownianComponent(cdUnitTest)
```

---

estimateDistFun *Estimate the Distance Function*

Description

Estimates the function that models how the expected number of counts decreases with increasing distance.

Usage

```
estimateDistFun(cd, method = "cubic", plot = TRUE, outfile = NULL)
```

Arguments

cd	A chicagoData object.
method	Choice of method: "cubic" is currently the only allowed option, which fits a cubic function with linear extrapolation, on a log-log scale.
plot	Output a diagnostic plot.
outfile	If NULL, plot to current device. Otherwise, plot to the .pdf file outfile.

Details

By default, we look in 75 distance bins, and a cubic fit is used. For distances that lie outside of the bin boundaries, it is assumed that the function is log-linear, with continuity of f and its first derivative on the log-scale.

Value

An object of class chicagoData, with the parameters of the distance function present as params(cd)$distFunParams.

Author(s)

Mikhail Spivakov, Jonathan Cairns, Paula Freire Pritchett

See Also

chicagoPipeline, plotDistFun

Examples

```
data(cdUnitTest)
estimateDistFun(cdUnitTest)
```

---

estimateTechnicalNoise *Estimate Technical Noise*

Description

Calculates the expected technical noise based on trans read pairs.

Usage

```
estimateTechnicalNoise(cd, plot = TRUE, outfile = NULL)
```

Arguments

cd	A chicagoData object.
plot	Logical - if TRUE, then output a diagnostic plot.
outfile	NULL, or a character string. If NULL, the diagnostic plot is outputted to the current plotting device. If a character string, e.g. outfile="tech.pdf", then the plot will be outputted to that file.

Value

An object of class chicagoData, with additional columns "tlb", "tblb", "Tmean".

Warning

The object intData(cd) is updated by reference. Thus, intData(cd) will be altered. See vignette for further information.

Author(s)

Mikhail Spivakov, Jonathan Cairns, Paula Freire Pritchett

See Also

chicagoPipeline

Examples

```
data(cdUnitTest)

##modifications to cdUnitTest, ensuring it uses correct design directory
designDir <- file.path(system.file("extdata", package="Chicago"), "unitTestDesign")
cdUnitTest <- modifySettings(cd=cdUnitTest, designDir=designDir)

cdUnitTest <- estimateTechnicalNoise(cdUnitTest)
```

`exportResults` *Export Results*

Description

Export the results from a `chicagoData` object to disk, or to a `GenomicInteractions` object.

Usage

```
exportResults(cd, outfileprefix, scoreCol = "score", cutoff = 5, b2bcutoff = NULL,
        format = c("seqMonk", "interBed", "washU_text"),
        order = c("position", "score")[1], removeMT=TRUE)
exportToGI(cd, scoreCol="score", cutoff=5, b2bcutoff=NULL,
                      order=c("position", "score")[1], removeMT=TRUE)
```

Arguments

`cd`	A `chicagoData` object.
`outfileprefix`	A character string that forms the prefix for each output file.
`scoreCol`	The column of `intData(cd)` that contains the score.
`cutoff`	The score cutoff.
`b2bcutoff`	If desired, an alternative score cutoff for bait-to-bait interactions.
`format`	The file format(s) to output. If a multiple formats are supplied as a vector, then all of these formats will be outputted. Supported formats are: "seqMonk", "interBed", "washU_text" and, for advanced users, "washU_track".
`order`	Should output be ordered by position or score?
`removeMT`	Logical. If `TRUE`, remove any interactions involving mitochondrial DNA from the output.

Details

Important notes on the washU formats: Most users will prefer "washU_text" output to "washU_track" output. The "washU_text" output can be uploaded to the washU browser directly. To do this, open the browser, select "Add custom tracks", and use the "Got text files instead? Upload them from your computer" link near the bottom of the page.

The "washU_track" output needs to be hosted elsewhere. You can then link the browser to the data via the "Interaction - pairwise interaction" button on the "Add custom tracks" page.

If you get the warning "WashU Browser track format could not be finalized due to absence of bgzip or tabix", this could be because you have not installed SAMtools and htslib. You can check with system2("tabix") and system2("bgzip"). Sometimes RStudio has issues with reading `$PATH` - you can check this with `system2("echo", "$PATH")`. Consider running the command in R, outside of RStudio, to fix this problem.

If all else fails, and you need "washU_track" output, then you can manually perform the final steps yourself by running: `bgzip <outfileprefix>_washU_track.txt` and `tabix -p bed <outfileprefix>.txt.gz`.

Value

`exportResults()`: NULL.

`exportToGI()`: a GenomicInteractions object. Anchor one is the bait, anchor two is the other end.

Author(s)

Mikhail Spivakov, Jonathan Cairns, Paula Freire Pritchett

See Also

chicagoPipeline

Examples

```
data(cdUnitTest)

##modifications to cdUnitTest, ensuring it uses correct design directory
designDir <- file.path(system.file("extdata", package="Chicago"), "unitTestDesign")
cdUnitTest <- modifySettings(cd=cdUnitTest, designDir=designDir)

##create a temporary directory, export output there
tempDirectory <- tempdir()
print(tempDirectory)
exportResults(cdUnitTest, outfileprefix = file.path(tempDirectory, "unitTestOutput"))

GI <- exportToGI(cdUnitTest)
```

getPvals *Get P-values*

Description

Based on a Delaporte model, calculate the P-value associated with each observation.

Usage

```
getPvals(cd)
```

Arguments

cd	A chicagoData object.

Details

The parameters for the Delaporte distribution are obtained as follows: the NB mean from the column intData(cd)$Bmean, the Poisson mean from the column intData(cd)$Tmean, and the dispersion from params(cd)$dispersion.

Value

An object of class chicagoData, with new column log.p.

Warning

The object intData(cd) is updated by reference. Thus, intData(cd) will be altered. See vignette for further information.

Author(s)

Mikhail Spivakov, Jonathan Cairns, Paula Freire Pritchett

See Also

chicagoPipeline

Examples

```
data(cdUnitTest)
cdUnitTest <- getPvals(cdUnitTest)
```

getScores *Get CHiCAGO scores.*

Description

Converts p-values into a CHiCAGO score, using p-value weighting.

Usage

```
getScores(cd, method = "weightedRelative",
      includeTrans = TRUE, plot = TRUE, outfile = NULL)
```

Arguments

cd	A chicagoData object.
method	Either "weightedRelative" (recommended), or "unweighted".
includeTrans	If FALSE, trans interactions are discounted.
plot	Plot a diagnostic plot.
outfile	A string containing a .pdf file location to write to.

Details

Weighting is performed using the parameters weightAlpha, weightBeta, weightGamma, weightDelta. Briefly, this function calculates weights w that decrease with increasing distance. Then, we construct weighted p-values p/w. As a result, the significance of long-range interactions is upweighted, and the significance of short-range interactions is downweighted.

Finally, the output score is calculated as -log(p/w) - log(w_max), where w_max is the highest attainable weight, and provided the score is positive (otherwise it is set to 0).

Please see the CHiCAGO paper and its supplementary for full details.

Value

An object of class chicagoData.

Warning

The object intData(cd) is updated by reference. Thus, intData(cd) will be altered. See vignette for further information.

Author(s)

Jonathan Cairns

References

Genovese, C. R., Roeder, K., and Wasserman, L. (2006). False discovery control with p-value weighting. Biometrika, 93, 509-524. doi:10.1093/biomet/93.3.509

See Also

chicagoPipeline

Examples

```
data(cdUnitTest)

##modifications to cdUnitTest, ensuring it uses correct design directory
designDir <- file.path(system.file("extdata", package="Chicago"), "unitTestDesign")
cdUnitTest <- modifySettings(cd=cdUnitTest, designDir=designDir)

cdUnitTest <- getScores(cdUnitTest)
```

---

getSkOnly *Get S_k factors from multiple replicates*

Description

Finds s_k scaling factors for a (potentially large) number of samples. Typically, these factors are used as library size factors in some sort of differential count algorithm (DESeq, EdgeR, baySeq, ...) to find differential binding events between samples.

Usage

```
getSkOnly(files, cd)
```

Arguments

files	Character vector containing the locations of the .chinput files to read in.
cd	A blank chicagoData object for reference, usually created with setExperiment.

Value

Numeric vector of s_k factors.

Author(s)

Jonathan Cairns, Paula Freire Pritchett, Mikhail Spivakov

See Also

readAndMerge

Examples

```
filesDir <- file.path(system.file("extdata", package="Chicago"), "unitTestData")
files <- file.path(filesDir, dir(filesDir))
print(files) ##we will read in and merge these files

designDir <- file.path(system.file("extdata", package="Chicago"), "unitTestDesign")

cd <- setExperiment(designDir=designDir)
s_k <- getSkOnly(files, cd)
```

`mergeSamples` *Merge samples together.*

Description

Merge a number of `chicagoData` objects together, summarising their counts into a normalised value.

Usage

```
mergeSamples(cdl, normalise = TRUE, NcolOut = "N",
    NcolNormPrefix = "NNorm", mergeMethod = c("weightedMean", "mean")[1], repNormCounts = (mergeMethod
```

Arguments

`cdl`	A `list` of `chicagoData` objects.
`normalise`	If `TRUE`, use a normalisation procedure, specified by `mergeMethod`, to arrive at a normalised count. If `FALSE`, take the mean number of reads.
`NcolOut`	The column to store the normalised counts in.
`NcolNormPrefix`	Each sample gains a normalised count column, that begins with this prefix.
`mergeMethod`	If `mergeMethod == "weightedMean"`, then NcolOut is the weighted mean of the sample-wise counts adjusted by the samples' respective scaling factors `s_k`. If `mergeMethod == "mean"`, then sample-specific counts are first normalised by dividing by s_k, and NcolOut is computed as the mean of these normalised counts.
`repNormCounts`	Report normalised counts for each replicate (by dividing them by s_k) in the <NcolNormPrefix>.<sampleNo> column (by default, NNorm.1, NNorm.2, etc.). This option is on by default when `mergeMethod == "mean"`. However, it can also be used with `mergeMethod == "weightedMean"` (but the normalised counts will still be produced by dividing the raw counts for each replicate by s_k).

Value

An object of class `chicagoData`, with a `params(cd)$s_k` slot added representing the per-sample scaling factors used in normalisation.

Note

Raw per-sample counts will be stored in the N.<sampleNo> column (N.1, N.2, etc.)

Author(s)

Mikhail Spivakov, Jonathan Cairns, Paula Freire Pritchett

See Also

readAndMerge

Examples

```
filesDir <- file.path(system.file("extdata", package="Chicago"), "unitTestData")
files <- file.path(filesDir, dir(filesDir))
print(files) ##we will read in and merge these files

designDir <- file.path(system.file("extdata", package="Chicago"), "unitTestDesign")

cdA <- setExperiment(designDir=designDir)
cdA <- readSample(files[1], cdA)

cdB <- setExperiment(designDir=designDir)
cdB <- readSample(files[2], cdB)

cdMerged <- mergeSamples(list(cdA, cdB))
```

---

modifySettings *Modify Settings*

Description

Modify the settings in a chicagoData object.

Usage

```
modifySettings(cd, designDir=NULL, settings=list(), settingsFile=NULL)
```

Arguments

cd	A chicagoData object.
designDir	The new location of the design directory, e.g "~/resources/path" or NULL if not modified.
settings	A named list containing settings to modify.
settingsFile	The location of a file containing settings or NULL if not provided. Each row should contain the name of a setting, followed by whitespace, followed by the value of that setting.

Details

cd's settings are updated. For a list of available settings, see defaultSettings.

Value

An object of class chicagoData.

Author(s)

Mikhail Spivakov, Jonathan Cairns, Paula Freire Pritchett

See Also

setExperiment, defaultSettings

Examples

```
designDir <- file.path(system.file("extdata", package="Chicago"), "unitTestDesign")

cd <- setExperiment(designDir)

##Suppose I want to change the number of samples drawn for dispersion estimation
print(settings(cd)$brownianNoise.subset)

cd <- modifySettings(cd, settings=list(brownianNoise.subset = 10))
print(settings(cd)$brownianNoise.subset)
```

---

normaliseBaits *Normalise Baits*

Description

Calculate normalisation factors s_j for each bait.

Usage

```
normaliseBaits(cd, normNcol = "NNb", shrink = FALSE,
      plot = TRUE, outfile = NULL, debug = FALSE)
```

Arguments

cd	A chicagoData object.
normNcol	The name of the column in cd that contains normalised counts.
shrink	Deprecated.
plot	If TRUE, output a diagnostic plot.
outfile	NULL, or a character string. If NULL, the diagnostic plot is outputted to the current plotting device. If a character string, e.g. outfile="tech.pdf", then the plot will be outputted to that file.
debug	Deprecated.

Details

A chicagoData object: intData(cd) gains a new column s_j, and normalised output NNb (unless the normNcol parameter is altered).

Value

An object of class chicagoData.

Warning

The object intData(cd) is updated by reference. Thus, intData(cd) will be altered. See vignette for further information.

Author(s)

Mikhail Spivakov, Jonathan Cairns, Paula Freire Pritchett

Examples

```
data(cdUnitTest)

##modifications to cdUnitTest, ensuring it uses correct design directory
designDir <- file.path(system.file("extdata", package="Chicago"), "unitTestDesign")
cdUnitTest <- modifySettings(cd=cdUnitTest, designDir=designDir)

cdUnitTest <- normaliseBaits(cdUnitTest)
```

---

normaliseOtherEnds *Normalise Other Ends*

Description

Compute s_i normalisation factors for other ends, and normalised counts.

Usage

```
normaliseOtherEnds(cd, Ncol = "NNb", normNcol = "NNboe", plot = TRUE, outfile = NULL)
```

Arguments

cd	A chicagoData object.
Ncol	The name of an input column in intData(cd) that contains counts normalised by bait (i.e. it is output from normaliseBaits.
normNcol	The name of an output column that will contain counts normalised by other ends (in addition to any normalisation already performed on the Ncol column). Useful for plotting.

plot	If TRUE, output a diagnostic plot.
outfile	NULL, or a character string. If NULL, the diagnostic plot is outputted to the current plotting device. If a character string, e.g. outfile="tech.pdf", then the plot will be outputted to that file.

Details

A chicagoData object: intData(cd) gains new columns s_i, and normalised output NNboe (unless the normNcol parameter is altered).

Value

An object of class chicagoData.

Warning

The object intData(cd) is updated by reference. Thus, intData(cd) will be altered. See vignette for further information.

Author(s)

Mikhail Spivakov, Jonathan Cairns, Paula Freire Pritchett

Examples

```
##FIXME: example can be run by loading data package if it is installed, once it exists

if("PCHiCdata" %in% rownames(installed.packages()))
{
  library(PCHiCdata)
  data(smESC)

  ##modifiy smESC to use correct design directory
  designDir <- file.path(system.file("extdata", package="PCHiCdata"), "mm9TestDesign")
  smESC <- modifySettings(cd=smESC, designDir=designDir)

  ##normalise here...
  normaliseOtherEnds(smESC)

} else {
  warning("Please install the PCHiCdata package to run this example.")
}
```

overlapFragWithFeatures *Overlap Other-Ends with Features*

Description

This function checks which other-ends from a chicagoData object overlap with a set of genomic features.

Usage

```
overlapFragWithFeatures(x = NULL, folder = NULL, list_frag, position_otherEnd = NULL,
  sep = "\t")
```

Arguments

x	a chicagoData object or a data table (data.table) containining other end IDs.
folder	the name of the folder where the files containing the features of interest are stored.
list_frag	a list where each element is the name of a file containing a feature of interest (e. g. H3K4me1, CTCF, DHS etc.). These files must have a bed format, with no header. Each element of the list must be named.
position_otherEnd	the name of the file containing the coordinates of the restriction fragments and the corresponding IDs. The coordinates should be "chromosome", "start" and "end", and the ID should be numeric. position_otherEnd only needs to be specified if x is not a chicagoData object.
sep	the field separator character. Values are separated by this character on each line of the file containing the coordinates of the restriction fragments (called by position_otherEnd).

Value

a data table (data.table) built from x, where a column was added for each genomic feature present in list_frag. The new columns contain logical values indicating whether there was an overlap or not.

Author(s)

Mikhail Spivakov, Jonathan Cairns, Paula Freire Pritchett

Examples

```
data(cdUnitTest)

##modifications to cdUnitTest, ensuring it uses correct design directory
designDir <- file.path(system.file("extdata", package="Chicago"), "unitTestDesign")
```

```
cdUnitTest <- modifySettings(cd=cdUnitTest, designDir=designDir)

##get the unit test ChIP tracks
dataPath <- system.file("extdata", package="Chicago")
ChIPdir <- file.path(dataPath, "unitTestChIP")
dir(ChIPdir)

##get a list of the unit test ChIP tracks
featuresFile <- file.path(ChIPdir, "featuresmESC.txt")
featuresTable <- read.delim(featuresFile, header=FALSE, as.is=TRUE)
featuresList <- as.list(featuresTable$V2)
names(featuresList) <- featuresTable$V1

##test for overlap
overlapFragWithFeatures(cdUnitTest, folder=ChIPdir, list_frag = featuresList)
```

---

peakEnrichment4Features

Enrichment for Features

Description

This function computes how many other-ends from a chicagoData object, that engage in significant interactions, overlap with a set of genomic features. In order to determine how those numbers compare to what would be expected if interaction significance had no effect on the overlaps, this function samples different sets of interactions from the non-significant pool and assesses how they overlap with genomic features (it computes the mean and confidence intervals). Results are returned in a table and plotted in a barplot. The difference between the results for the set of significant interactions and the random samples can be used as a measure of the enrichment for genomic features. Samples have the same size as the number of significant interactions called. Moreover, they follow the same distribution of distances between bait and other-end. This is achieved by binning this distribution and drawing interactions per bin, according to the numbers observed in the significant set.

Usage

```
peakEnrichment4Features(x1, folder=NULL,  list_frag, no_bins=NULL, sample_number,
                        position_otherEnd= NULL,colname_dist=NULL,
                      score=5, colname_score="score",min_dist=0, max_dist=NULL,
                        sep="\t", filterB2B=TRUE, b2bcol="isBait2bait",
                  unique=TRUE,plot_name=NULL, trans=FALSE, plotPeakDensity=FALSE)
```

Arguments

x1	a chicagoData object or a data table (data.table) containining other end IDs.
folder	the name of the folder where the files containing the features of interest are stored.

`list_frag` a list where each element is the name of a file containing a feature of interest (e. g. H3K4me1, CTCF, DHS etc.). These files must have a bed format, with no header. Each element of the list must be named.

`no_bins` Number of bins to divide the range of `colname_dist` (after `colname_dist` has been trimmed according to `min_dist` and `max_dist`). This will be important to determine how many interactions should be sampled according to distance from bait. For more details see `Note` below.

`sample_number` Number of samples to be used in the permutation test. Large numbers of samples (around 100) are recommended. Nevertheless, smaller numbers (around 10) speed up the processing time and have shown to give sensible results when compared to large numbers.

`position_otherEnd`
the name of the file containing the coordinates of the restriction fragments and the corresponding IDs. The coordinates should be "chromosome", "start" and "end", and the ID should be numeric. `position_otherEnd` only needs to be specified if x is not a `chicagoData` object.

`colname_dist` the name of the column which contains the distances between bait and other end. `colname_dist` only needs to be specified if x is not a `chicagoData` object.

`score` the threshold above which interactions start being called as significant.

`colname_score` the name of the column which contains the score values which establish the level of significance of each interaction.

`min_dist` the minimum distance from bait required in the query. If this parameter is set to `NULL` and `trans` is set to `TRUE`, cis interactions are disregarded from the analysis. This parameter is also useful when the user only wants to look at cis distal interactions (very far from bait).

`max_dist` the maximum distance from bait required in the query. This parameter is particularly useful when the user only wants to look at cis proximal interactions (interactions surrounding the bait).

`sep` the field separator character. Values are separated by this character on each line of the file containing the coordinates of the restriction fragments (called by `position_otherEnd`).

`filterB2B` a logical value indicating whether bait-to-bait interactions should be removed from the analysis.

`b2bcol` the name of the column identifying bait-to-bait interactions in the `x1`.

`unique` a logical value indicating whether to removing duplicated other-ends from significant interactions and samples.

`plot_name` the name of the file where to save the resulting plot. This parameter is only required if the user wants to save the plot. Otherwise, the plot will be displayed on the screen, but not saved.

`trans` a logical value indicating whether the enrichment is to be computed for trans interactions. If this parameter is set to `TRUE` and `min_dist` is set to `NULL`, cis interactions are disregarded from the analysis.

`plotPeakDensity`
a logical value indicating whether to plot the density of interactions with distance. Setting this parameter to `TRUE` only applies to cis interactions.

Value

a data frame containing columns for the number of overlaps for each feature in our significant interactions, the average number of overlaps for each feature in our samples, the corresponding standard deviations.

Note

The number of interactions sampled per distance follows the same distribution as the one in the significant set. This is achieved by counting the number of significant interactions per distance bin. In this way, when samples are computed, the number of interactions drawn will depend on each distance bin. Each sample will have the same number of interactions per bin as in the significant set. To improve this computation, we recommend a bin size of around 10-20kb, but this number could be larger when looking at distal interactions only (up to 200kb). This is established using the parameter `no_bins`. For instance, using `min_dist=0` and `max_dist=1e6`, `no_bins` should be set to `100` so to obtain 10kb bins.

Author(s)

Mikhail Spivakov, Jonathan Cairns, Paula Freire Pritchett

Examples

```
data(cdUnitTest)

##modifications to cdUnitTest, ensuring it uses correct design directory
designDir <- file.path(system.file("extdata", package="Chicago"), "unitTestDesign")
cdUnitTest <- modifySettings(cd=cdUnitTest, designDir=designDir)

##get the unit test ChIP tracks
dataPath <- system.file("extdata", package="Chicago")
ChIPdir <- file.path(dataPath, "unitTestChIP")
dir(ChIPdir)

##get a list of the unit test ChIP tracks
featuresFile <- file.path(ChIPdir, "featuresmESC.txt")
featuresTable <- read.delim(featuresFile, header=FALSE, as.is=TRUE)
featuresList <- as.list(featuresTable$V2)
names(featuresList) <- featuresTable$V1

##test for overlap
peakEnrichment4Features(cdUnitTest, folder=ChIPdir, list_frag = featuresList, no_bins = 500, sample_number = 100)
```

---

plotBaits *Plot Baits*

Description

Plot the read counts around baits.

Usage

```
plotBaits(cd, pcol = "score", Ncol = "N", n = 16, baits = NULL,
      plotBaitNames = TRUE, plotBprof = FALSE, plevel1 = 5, plevel2 = 3,
      outfile = NULL, removeBait2bait = TRUE, width = 20, height = 20,
      maxD = 1e6, bgCol = "black", lev2Col = "blue", lev1Col = "red",
      bgPch = 1, lev1Pch = 20, lev2Pch = 20, ...)
```

Arguments

cd	A chicagoData object.
pcol	The name of the column that contains the score.
Ncol	The name of the column that contains counts.
n	The number of baits to plot (ignored if baits is specified).
baits	The IDs of the baits to plot.
plotBaitNames	If TRUE, the names of the baits, rather than their IDs, will appear in the plot.
plotBprof	If TRUE, display a line representing the expected Brownian noise at each distance.
plevel1, plevel2	Thresholds used on the pcol column. plevel1 should be the more stringent threshold.
outfile	If NULL, output to current plotting device. Otherwise, this specifies a pdf file to write to.
removeBait2bait	If TRUE, bait-to-bait interactions are not plotted.
width, height	Passed through to pdf
maxD	The maximum (linear) distance each side of the bait to plot (NULL to include the whole chromosome).
bgCol, lev1Col, lev2Col	Colours to be used for background points, and for the two stringency levels defined by plevel1 and plevel2, respectively.
bgPch, lev1Pch, lev2Pch	Plotting character for background points, and for points exceeding the two stringency levels defined by plevel1 and plevel2, respectively. Specified as per pch in points.
...	Additional arguments passed to plot

Value

Vector of the baitIDs plotted (useful if baitIDs were sampled randomly).

Author(s)

Mikhail Spivakov, Jonathan Cairns, Paula Freire Pritchett

Examples

```
data(cdUnitTest)

##modifications to cdUnitTest, ensuring it uses correct design directory
designDir <- file.path(system.file("extdata", package="Chicago"), "unitTestDesign")
cdUnitTest <- modifySettings(cd=cdUnitTest, designDir=designDir)

plotBaits(cdUnitTest)
```

`plotDistFun` *Plot the Distance Function*

Description

Estimates the function that models how the expected number of counts decreases with increasing distance.

Usage

```
plotDistFun(cd, ...)
```

Arguments

`cd`	A `chicagoData` object.
`...`	Further arguments passed to `plot`.

Value

A plot.

Author(s)

Jonathan Cairns

See Also

`estimateDistFun`

Examples

```
data(cdUnitTest)
plotDistFun(cdUnitTest)
```

readAndMerge *Read And Merge*

Description

A wrapper that calls readSample() on a number of files, then mergeSamples().

Usage

```
readAndMerge(files, cd, ...)
```

Arguments

files	Character vector containing the locations of the files to read in.
cd	A chicagoData object, usually created with setExperiment.
...	Further arguments passed to mergeSamples.

Value

An object of class chicagoData.

Author(s)

Mikhail Spivakov, Jonathan Cairns, Paula Freire Pritchett

See Also

readSample, mergeSamples

Examples

```
filesDir <- file.path(system.file("extdata", package="Chicago"), "unitTestData")
files <- file.path(filesDir, dir(filesDir))
print(files) ##we will read in and merge these files

designDir <- file.path(system.file("extdata", package="Chicago"), "unitTestDesign")

cd <- setExperiment(designDir=designDir)
cd <- readAndMerge(files, cd)
```

readSample *Read Sample*

Description

This function reads input data from a file, into a chicagoData object.

Usage

```
readSample(file, cd)
```

Arguments

file	The location of an input file FIXME more details!
cd	A chicagoData object.

Value

An object of class chicagoData.

Warning

The object intData(x) is updated by reference. Thus, intData(cd) will be altered. See vignette for further information.

Author(s)

Mikhail Spivakov, Jonathan Cairns, Paula Freire Pritchett

Examples

```
filesDir <- file.path(system.file("extdata", package="Chicago"), "unitTestData")
file <- file.path(filesDir, dir(filesDir))[1]
print(file) ##we will read in this file

designDir <- file.path(system.file("extdata", package="Chicago"), "unitTestDesign")

cd <- setExperiment(designDir=designDir)
cd <- readAndMerge(file, cd)
```

setExperiment *Set Experiment*

Description

Creates a template CHiCAGO experiment object. This should be the first function called.

Usage

```
setExperiment(designDir = "", settings = list(),
      settingsFile = NULL, def.settings=defaultSettings())
```

Arguments

designDir	The location of the design directory, e.g "~/resources/path". (Should not end with a slash.)
settings	A named list containing settings to apply. Setting names(settings)[1] is set to (settings)[[1]], and so on. This overrides anything specified in settingsFile, or in def.settings.
settingsFile	The location of a file containing settings. Each row should contain the name of a setting, followed by whitespace, followed by the value of that setting. Overrides anything specified in def.settings.
def.settings	These are the default settings.

Details

For a list of settings, see defaultSettings.

Value

An object of class chicagoData.

Author(s)

Mikhail Spivakov, Jonathan Cairns, Paula Freire Pritchett

See Also

defaultSettings

Examples

```
designDir <- file.path(system.file("extdata", package="Chicago"), "unitTestDesign")

cd <- setExperiment(designDir)
```

Index

∗ **datasets**
    cdUnitTest, 3
∗ **package**
    Chicago-package, 2

cdUnitTest, 3
Chicago *(*Chicago-package*)*, 2
Chicago-package, 2
chicagoData, *3*, 4, *6*, *23–25*
chicagoPipeline, 5, *10*, *12*, *14, 15*, *17*
copyCD, 6

data.table, *23, 24*
defaultSettings, *4*, 7, *20*, *31*

estimateBrownianComponent, 10
estimateBrownianNoise, *5*
estimateBrownianNoise
        *(*estimateBrownianComponent*)*, 10
estimateDistFun, *5*, 11, *28*
estimateTechnicalNoise, *5*, 12
exportResults, *5, 6*, 13
exportToGI *(*exportResults*)*, 13

GenomicInteractions, *13, 14*
getPvals, *5*, 15
getScores, *5*, 16
getSkOnly, 17

intData *(*chicagoData*)*, 4
intData,chicagoData-method
        *(*chicagoData*)*, 4
intData<- *(*chicagoData*)*, 4
intData<-,chicagoData-method
        *(*chicagoData*)*, 4

mergeSamples, 18, *29*
modifySettings, *4*, *10*, 19

normaliseBaits, *5*, 20, *21*
normaliseOtherEnds, *5*, 21

overlapFragWithFeatures, 23

params *(*chicagoData*)*, 4
params,chicagoData-method
        *(*chicagoData*)*, 4
params<- *(*chicagoData*)*, 4
params<-,chicagoData-method
        *(*chicagoData*)*, 4
pdf, *27*
peakEnrichment4Features, 24
plot, *27, 28*
plotBaits, 26
plotDistFun, *12*, 28
points, *27*

readAndMerge, *19*, 29
readSample, *29*, 30

setExperiment, *4*, *10*, *17*, *20*, *29*, 31
settings *(*chicagoData*)*, 4
settings,chicagoData-method
        *(*chicagoData*)*, 4
smESC, *3*