

Package ‘Brundle’

October 12, 2022

Type Package

Title Normalisation Tools for Inter-Condition Variability of ChIP-Seq Data

Version 1.0.9

Author Andrew N Holding

Maintainer Andrew N Holding <andrew.holding@cruk.cam.ac.uk>

Description Inter-sample condition variability is a key challenge of normalising ChIP-seq data. This implementation uses either spike-in or a second factor as a control for normalisation. Input can either be from 'DiffBind' or a matrix formatted for 'DESeq2'. The output is either a 'DiffBind' object or the default 'DESeq2' output. Either can then be processed as normal. Supporting manuscript Guertin, Markowetz and Holding (2017) <[doi:10.1101/182261](https://doi.org/10.1101/182261)>.

License CC BY 4.0

Encoding UTF-8

LazyData true

Depends R (>= 2.10), DiffBind, Rsamtools, DESeq2, lattice, stats, utils, graphics

RoxygenNote 6.1.1

biocViews Software, Technology, Sequencing, ChIPSeq

NeedsCompilation no

Repository CRAN

Date/Publication 2019-04-23 22:00:23 UTC

R topics documented:

Brundle	2
dbaControl	3
dbaExperiment	4
jg.applyNormalisation	4
jg.conditions	5
jg.controlCountsTreated	5
jg.controlCountsUntreated	6

jg.controlPeaks	6
jg.controlPeakset	7
jg.controlPeaksetDeSeq	7
jg.controlResultsDeseq	8
jg.convertPeakset	8
jg.correctDBASizeFactors	9
jg.countAlignedMReads	9
jg.dbaGetPeakset	10
jg.experimentPeakset	10
jg.experimentResultsDeseq	11
jg.getControlCounts	11
jg.getCorrectionFactor	12
jg.getDbA	13
jg.getNormalizationCoefficient	13
jg.getSampleIds	14
jg.plotDeSeq	14
jg.plotDeSeqCombined	15
jg.plotMA	16
jg.plotNormalization	16
jg.runDeSeq	17

Index	18
--------------	-----------

 Brundle

Brundle

Description

Normalise one DiffBind object to a second control set of peaks.

Usage

```
Brundle(dbaExperiment, dbaControl, jg.treatedCondition,
        jg.untreatedCondition, jg.experimentSampleSheet, jg.controlSampleSheet,
        jg.correctionFactor = FALSE, jg.noBAMs = FALSE)
```

Arguments

dbaExperiment	DiffBind object to be normalised
dbaControl	DiffBind object of control peaks
jg.treatedCondition	Identical to treated condition in the sample sheet.
jg.untreatedCondition	Identical to the control condition in the sample sheet.
jg.experimentSampleSheet	Filename of samplesheet for experimental peaks

`jg.controlSampleSheet` Filename of samplesheet for control peaks

`jg.correctionFactor` Manually specify correction factor.

`jg.noBAMs` If set to true, correction factor is estimated from the DiffBind object.

Examples

```
data(dbaExperiment,package="Brundle")
data(dbaControl,package="Brundle")
fpath <- system.file("extdata", "samplesheet_SLX14438_hs_ER_DBA.csv",package="Brundle")
jg.ExperimentSampleSheet<-fpath
fpath <- system.file("extdata", "samplesheet_SLX14438_hs_CTCF_DBA.csv",package="Brundle")
jg.ControlSampleSheet<-fpath
Brundle(dbaExperiment,dbaControl,"Fulvestrant","none",
        jg.ExperimentSampleSheet,jg.ControlSampleSheet,jg.noBAMs=TRUE)
```

dbaControl

Example DiffBind Object of Control Peaks

Description

This data set is a DiffBind object of example ChIP-seq experiments

Usage

`jdbaExperiment`

Format

Diffbind Object

Author(s)

Andrew Holding

Source

Original data from experiment

`dbaExperiment`*Example DiffBind Object of Experimental Peaks*

Description

This data set is a DiffBind object of example ChIP-seq experiments

Usage

```
dbaExperiment
```

Format

Diffbind Object

Author(s)

Andrew Holding

Source

Original data from experiment

`jg.applyNormalisation` *jg.applyNormalisation*

Description

Takes the experimental peakset and applies the calculated coefficient and correction factor.

Usage

```
jg.applyNormalisation(jg.experimentPeakset, jg.coefficient,  
jg.correctionFactor, jg.treatedNames)
```

Arguments

`jg.experimentPeakset`

is the peakset extracted from the Diffbind object

`jg.coefficient` is the coefficient calculated by `jg.getNormalizationCoefficient`

`jg.correctionFactor`

is the correction factor calculated by `jg.getCorrectionFactor`

`jg.treatedNames`

is the names of the treated samples

Examples

```
data(jg.experimentPeakset, package="Brundle")
jg.experimentPeaksetNormalised<-jg.applyNormalisation(jg.experimentPeakset,
  1.267618,
  0.6616886,
  c("1b", "2b", "3b"))
```

jg.conditions

Example Sample Condition Matrix

Description

This data set is a list of example conditions

Usage

jg.conditions

Format

list

Author(s)

Andrew Holding

Source

Original data from experiment

jg.controlCountsTreated

Example ChIP-seq Count Matrix

Description

This data set is a list counts from an Example ChIP-seq experiment

Usage

jg.controlCountsTreated

Format

matrix

Author(s)

Andrew Holding

Source

Original data from experiment

jg.controlCountsUntreated

Example ChIP-seq Count Matrix

Description

This data set is a list counts from an Example ChIP-seq experiment

Usage

jg.controlCountsUntreated

Format

matrix

Author(s)

Andrew Holding

Source

Original data from experiment

jg.controlPeaks

Example ChIP-seq Control Peakset

Description

This data set is a matrix of example peaks for normalisation of ChIP-seq data

Usage

jg.controlPeaks

Format

a matrix of locations and counts per experimental sample

Author(s)

Andrew Holding

Source

Original data from experiment

jg.controlPeakset *Example ChIP-seq Control Peakset*

Description

This data set is a matrix of example control peaks for normalisation of ChIP-seq data

Usage

jg.controlPeakset

Format

a matrix of locations and counts per control sample

Author(s)

Andrew Holding

Source

Original data from experiment

jg.controlPeaksetDeSeq
Example ChIP-seq Control Peakset

Description

This data set is a matrix of example control peaks for normalisation of ChIP-seq data formatted for DeSEQ2

Usage

jg.controlPeaksetDeSeq

Format

a matrix of locations and counts sample

Author(s)

Andrew Holding

Source

Original data from experiment

`jg.controlResultsDeseq`

Example ChIP-seq DeSEQ2 Control results

Description

This data set is a GRanges object of example control results from DeSEQ2

Usage

`jg.controlResultsDeseq`

Format

DeSEQ2 results object

Author(s)

Andrew Holding

Source

Original data from experiment

`jg.convertPeakset`

jg.convertPeakset

Description

Converts a DiffBind object into a DESeq2 compatible form for the workflow.

Usage

`jg.convertPeakset(jg.controlPeakset)`

Arguments

`jg.controlPeakset`

is the name of the DiffBind object to convert

Examples

```
jg.convertPeakset(jg.controlPeakset)
```

```
jg.correctDBASizeFactors  
      jg.correctDBASizeFactors
```

Description

Correct the size factors in a DiffBind object using our DESeq2 pipeline for normalisation.

Usage

```
jg.correctDBASizeFactors(dba, jg.controlSizeFactors)
```

Arguments

dba Diffbind object to have size factors corrected
jg.controlSizeFactors Vector of replacement size factors

Examples

```
data(jg.controlPeaksetDeSeq, package="Brundle")  
data(dbaExperiment, package="Brundle")  
jg.controlSizeFactors = estimateSizeFactorsForMatrix(jg.controlPeaksetDeSeq)  
jg.correctDBASizeFactors(dbaExperiment, jg.controlSizeFactors)
```

```
jg.countAlignedMReads    jg.countAlignedMReads
```

Description

This function counts the number of aligned reads in millions from a list of bam files. It returns these in as a list of numbers in the same order.

Usage

```
jg.countAlignedMReads(jg.bamFiles)
```

Arguments

jg.bamFiles is a list of bam files to count.

jg.dbaGetPeakset *dbaGetPeakset*

Description

Extracts a peakset from a dba object.

Usage

```
jg.dbaGetPeakset(dba)
```

Arguments

dba is the name of the DiffBind object

Examples

```
data(dbaExperiment, package="Brundle")
jg.experimentPeakset <- jg.dbaGetPeakset(dbaExperiment)
```

jg.experimentPeakset *Example ChIP-seq Experiment Peakset*

Description

This data set is a matrix of example peaks for normalisation of ChIP-seq data

Usage

```
jg.experimentPeakset
```

Format

a matrix of locations and counts per control sample

Author(s)

Andrew Holding

Source

Original data from experiment

`jg.experimentResultsDeseq`

Example ChIP-seq DeSEQ2 results

Description

This data set is a GRanges object of example results from DeSEQ2

Usage

`jg.experimentResultsDeseq`

Format

DeSEQ2 results object

Author(s)

Andrew Holding

Source

Original data from experiment

`jg.getControlCounts` *jg.getControlCounts*

Description

This function counts the number of aligned reads in millions from a list of bam files. It returns these in as a list of numbers in the same order.

Usage

`jg.getControlCounts(jg.control, jg.controlSampleSheet, jg.Condition)`

Arguments

`jg.control` is a peakset extracted by `jg.dbGetPeakset`.

`jg.controlSampleSheet`
 is the samplesheet supplied to `DiffBind` to generate `jg.control`.

`jg.Condition` is the condition we the counts for as specficed in the samplesheet.

Examples

```
data(jg.controlPeakset, package="Brundle")
fpath <- system.file("extdata", "samplesheet_SLX14438_hs_CTCF_DBA.csv", package="Brundle")
jg.controlSampleSheet<-fpath
jg.controlCountsTreated<-jg.getControlCounts(jg.controlPeakset, jg.controlSampleSheet, "Fulvestrant")
```

```
jg.getCorrectionFactor  
                          jg.getCorrectionFactor
```

Description

Generates a correction factor that is applied before reinserting the data into the DiffBind object for analysis.

Usage

```
jg.getCorrectionFactor(jg.experimentSampleSheet, jg.treatedNames,  
                      jg.untreatedNames)
```

Arguments

`jg.experimentSampleSheet`
is the csv samplesheet used to load the data into DiffBind

`jg.treatedNames`
is a list of the names of samples that are treated

`jg.untreatedNames`
is a list of the names of samples that are untreated

Examples

```
data(jg.controlCountsTreated, package="Brundle")
data(jg.controlCountsUntreated, package="Brundle")
jg.coefficient<-jg.getNormalizationCoefficient(jg.controlCountsTreated, jg.controlCountsUntreated)
```

jg.getDbA

jg.getDbA

Description

Generates a DiffBind object from a valid SampleSheet with the required data for normalisation. No examples are provided as BAM files are not included in this package.

Usage

```
jg.getDbA(jg.experimentSampleSheet, dbaSummits, ...)
```

Arguments

jg.experimentSampleSheet	is the filename of samplesheet to be loaded
dbaSummits	is the peak width in bp from summits (optional)
...	are the parameters to be passed to DiffBinds dba.count function

jg.getNormalizationCoefficient

jg.getNormalizationCoefficient

Description

This function allows the user to carry out the normalisation and returns a coefficient by using a linear fit to the control data.

Usage

```
jg.getNormalizationCoefficient(jg.controlCountsTreated,
jg.controlCountsUntreated)
```

Arguments

jg.controlCountsTreated	Control counts extracted from the Diffbind object for the treated condition using jg.getControlCounts
jg.controlCountsUntreated	Control counts extracted from the Diffbind object for the untreated condition using jg.getControlCounts

Examples

```
data(jg.controlCountsTreated, package="Brundle")
data(jg.controlCountsUntreated, package="Brundle")
jg.coefficient<-jg.getNormalizationCoefficient(jg.controlCountsTreated,
                                              jg.controlCountsUntreated)
```

<code>jg.getSampleIds</code>	<i>jg.getSampleIds</i>
------------------------------	------------------------

Description

Extracts the sample Id from DiffBind formatted SampleSheet in csv format.

Usage

```
jg.getSampleIds(jg.controlSampleSheet)
```

Arguments

`jg.controlSampleSheet`
is the filename of the samplesheet

<code>jg.plotDeSeq</code>	<i>jg.plotDeSeq</i>
---------------------------	---------------------

Description

Plots the output from DESeq2 for the Brundle pipeline

Usage

```
jg.plotDeSeq(ma.df, p = 0.01, title.main = "Differential ChIP",
             log2fold = 0.5, flip = FALSE)
```

Arguments

<code>ma.df</code>	is the result Dataframe from DESeq2
<code>p</code>	is the minimum FDR to highlight as significant
<code>title.main</code>	is the plot title
<code>log2fold</code>	is the minimum log2 fold change for highlighted points
<code>flip</code>	when set to TRUE flips the data

Examples

```
data(jg.experimentResultsDeseq, package="Brundle")
jg.experimentResultsDeseq<-suppressWarnings(as.data.frame(jg.experimentResultsDeseq))
jg.plotDeSeq(jg.experimentResultsDeseq,
  p=0.01,
  title.main="Fold-change in ER binding",
  flip=TRUE
)
```

jg.plotDeSeqCombined *jg.plotDeSeqCombined*

Description

Overlays the plots from the output from DESeq2 for the Brundle pipeline

Usage

```
jg.plotDeSeqCombined(jg.controlResultsDeseq, jg.experimentResultsDeseq,
  title.main, padjX, flip = FALSE)
```

Arguments

<code>jg.controlResultsDeseq</code>	is the result Dataframe from DESeq2 for the control conditions
<code>jg.experimentResultsDeseq</code>	is the result Dataframe from DESeq2 for the experimental conditions
<code>title.main</code>	is the plot title
<code>padjX</code>	is the minimum FDR to highlight as significant
<code>flip</code>	when set to TRUE flips the data

Examples

```
data(jg.controlResultsDeseq, package="Brundle")
data(jg.experimentResultsDeseq, package="Brundle")

jg.plotDeSeqCombined(as.data.frame(jg.controlResultsDeseq),
  as.data.frame(jg.experimentResultsDeseq),
  title.main="ER and CTCF Binding Folding changes on ER treatment",
  p=0.01, flip=TRUE)
```

`jg.plotMA` *jg.plotMA*

Description

This function plots both the control and experimental data on an MA plot. It also allows for the user to provide a normalisation coefficient for the data.

Usage

```
jg.plotMA(jg.experimentPeakset, jg.controlPeakset, jg.untreatedNames,
          jg.treatedNames, jg.coefficient)
```

Arguments

`jg.experimentPeakset`
is the peakset of the experimental data extracted from a DiffBind object with `jg.dbaGetPeakset`

`jg.controlPeakset`
is the peakset of the control data extracted from a DiffBind object with `jg.dbaGetPeakset`

`jg.untreatedNames`
is a list of sample names for the control or untreated conditions

`jg.treatedNames`
is a list of sample samples for the treated conditions

`jg.coefficient` is a normalisation coefficient for the data that can be generated via the pipeline. Can be set to 1 to view before normalisation.

`jg.plotNormalization` *jg.plotNormalization*

Description

This function allows the user to visualize the normalisation. It is not needed for the pipeline but provides a helpful illustration of the process.

Usage

```
jg.plotNormalization(jg.controlCountsTreated, jg.controlCountsUntreated)
```

Arguments

`jg.controlCountsTreated`
Control counts extracted from the Diffbind object for the treated condition using `jg.getControlCounts`

`jg.controlCountsUntreated`
Control counts extracted from the Diffbind object for the untreated condition using `jg.getControlCounts`

Examples

```
data(jg.controlCountsTreated, package="Brundle")
data(jg.controlCountsUntreated, package="Brundle")
jg.plotNormalization(jg.controlCountsTreated, jg.controlCountsUntreated)
```

*jg.runDeSeq**jg.runDeSeq*

Description

Runs DESeq2 on our peakset after we have obtained the normalised size factors.

Usage

```
jg.runDeSeq(jg.PeaksetDeSeq, jg.conditions, jg.SizeFactors = NULL)
```

Arguments

jg.PeaksetDeSeq is the experimental peakset formatted for DESeq2
jg.conditions is the list of conditions to be compared
jg.SizeFactors is the size factors generated from the control samples

Examples

```
data(jg.controlPeaksetDeSeq, package="Brundle")
data(jg.conditions, package="Brundle")
jg.controlSizeFactors = estimateSizeFactorsForMatrix(jg.controlPeaksetDeSeq)
jg.runDeSeq(jg.controlPeaksetDeSeq, jg.conditions, jg.SizeFactors = NULL)
```

Index

- * **Convert**
 - [jg.convertPeakset, 8](#)
 - * **DESeq2**
 - [Brundle, 2](#)
 - [jg.convertPeakset, 8](#)
 - [jg.correctDBASizeFactors, 9](#)
 - [jg.plotDeSeq, 14](#)
 - [jg.plotDeSeqCombined, 15](#)
 - [jg.runDeSeq, 17](#)
 - * **DiffBind**
 - [jg.convertPeakset, 8](#)
 - [jg.dbaGetPeakset, 10](#)
 - [jg.getCorrectionFactor, 12](#)
 - [jg.getDb, 13](#)
 - [jg.getSampleIds, 14](#)
 - * **Diffbind**
 - [Brundle, 2](#)
 - [jg.correctDBASizeFactors, 9](#)
 - * **bamFiles**
 - [jg.countAlignedMReads, 9](#)
 - [jg.getControlCounts, 11](#)
 - * **bam**
 - [jg.countAlignedMReads, 9](#)
 - [jg.getControlCounts, 11](#)
 - * **correction**
 - [jg.applyNormalisation, 4](#)
 - [jg.getCorrectionFactor, 12](#)
 - * **counts**
 - [jg.dbaGetPeakset, 10](#)
 - [jg.getDb, 13](#)
 - * **data**
 - [jg.plotDeSeq, 14](#)
 - [jg.plotDeSeqCombined, 15](#)
 - * **load**
 - [jg.getDb, 13](#)
 - * **normalisation**
 - [jg.applyNormalisation, 4](#)
 - [jg.getCorrectionFactor, 12](#)
 - * **normalization**
 - [jg.getNormalizationCoefficient, 13](#)
 - [jg.plotMA, 16](#)
 - [jg.plotNormalization, 16](#)
 - * **peakset**
 - [jg.applyNormalisation, 4](#)
 - [jg.dbaGetPeakset, 10](#)
 - * **plot**
 - [jg.plotDeSeq, 14](#)
 - [jg.plotDeSeqCombined, 15](#)
 - [jg.plotMA, 16](#)
 - [jg.plotNormalization, 16](#)
 - * **reads**
 - [jg.countAlignedMReads, 9](#)
 - [jg.getControlCounts, 11](#)
 - * **samplesheet**
 - [jg.getDb, 13](#)
 - [jg.getSampleIds, 14](#)
 - * **sample**
 - [jg.getSampleIds, 14](#)
- [Brundle, 2](#)
- [dbaControl, 3](#)
- [dbaExperiment, 4](#)
- [jg.applyNormalisation, 4](#)
- [jg.conditions, 5](#)
- [jg.controlCountsTreated, 5](#)
- [jg.controlCountsUntreated, 6](#)
- [jg.controlPeaks, 6](#)
- [jg.controlPeakset, 7](#)
- [jg.controlPeaksetDeSeq, 7](#)
- [jg.controlResultsDeseq, 8](#)
- [jg.convertPeakset, 8](#)
- [jg.correctDBASizeFactors, 9](#)
- [jg.countAlignedMReads, 9](#)
- [jg.dbaGetPeakset, 10](#)
- [jg.experimentPeakset, 10](#)
- [jg.experimentResultsDeseq, 11](#)
- [jg.getControlCounts, 11](#)

`hg.getCorrectionFactor`, [12](#)
`hg.getDb`, [13](#)
`hg.getNormalizationCoefficient`, [13](#)
`hg.getSampleIds`, [14](#)
`hg.plotDeSeq`, [14](#)
`hg.plotDeSeqCombined`, [15](#)
`hg.plotMA`, [16](#)
`hg.plotNormalization`, [16](#)
`hg.runDeSeq`, [17](#)