

# Package ‘DChIPRep’

April 11, 2018

**Title** DChIPRep - Analysis of chromatin modification ChIP-Seq data with replication

**Version** 1.8.0

**Description** The DChIPRep package implements a methodology to assess differences between chromatin modification profiles in replicated ChIP-Seq studies as described in Chabbert et. al - <http://www.dx.doi.org/10.15252/msb.20145776>. A detailed description of the method is given in the software paper at <https://doi.org/10.7717/peerj.1981>

**Depends** R (>= 3.4), DESeq2

**Imports** methods, stats, utils, ggplot2, fdrtool, reshape2, GenomicRanges, SummarizedExperiment, smoothmest, plyr, tidyr, assertthat, S4Vectors, purrr, soGGi, ChIPpeakAnno

**License** MIT + file LICENCE

**LazyData** true

**Suggests** mgcv, testthat, BiocStyle, knitr, rmarkdown

**Collate** 'AllClasses.R' 'AllGenerics.R' 'DChipRep.R' 'dataImport.R' 'dataImportsoGGi.R' 'documentData.R' 'methods.R' 'plottingFunctions.R' 'runTesting.R'

**VignetteBuilder** knitr

**biocViews** Sequencing, ChIPSeq, WholeGenome

**SystemRequirements** Python 2.7, HTSeq (>= 0.6.1), numpy, argparse, sys

**NeedsCompilation** no

**Author** Bernd Klaus [aut, cre], Christophe Chabbert [aut], Sebastian Gibb [ctb]

**Maintainer** Bernd Klaus <bernd.klaus@embl.de>

**RoxygenNote** 6.0.1

## R topics documented:

chip_galonska . . . . .	2
DChIPRep . . . . .	3
DChIPRepResults-class . . . . .	3
DESeq2Data . . . . .	4
exampleChipData . . . . .	4
exampleInputData . . . . .	5
exampleSampleTable . . . . .	5

FDRresults	6
getMATfromDataFrame	6
importData	7
importDataFromMatrices	8
importData_soGGi	9
input_galonska	10
plotProfiles	10
plotSignificance	11
resultsDChIPRep	12
robust_mean	12
runTesting	13
sample_table_galonska	14
show	14
summarizeCountsPerPosition	15
testData	16
TSS_galonska	16

## Index 17

---

chip_galonska	<i>Another example ChIP data set that can be used with <a href="#">importDataFromMatrices</a> from rom Galonska et. al., 2015.</i>
---------------	--

---

## Description

The contains the H3Kme3 data in the serum and 24h\_2i conditions from Galonska et. al., 2015 as well as the whole cell extract data, which is treated as input for all four samples. The data were downloaded from the SRA at the european nucleotide archive (ENA, accession PRJNA242892). The reads were aligned ot the mm9 reference genome using bowtie2 (Langmead and Salzberg, 2012) with default options. Then, filtering of unmapped, low mapping quality (< 10), duplicated and multi-mapping reads was performed with Picard tools. The fragment length was inferred using cross correlation plots from SPP (Kharchenko, et. al., 2008).

## Usage

```
data(chip_galonska)
```

## Format

a matrix

## Value

a matrix

## References

Galonska, Christina, Michael J. Ziller, Rahul Karnik, and Alexander Meissner. 2015. "Ground State Conditions Induce Rapid Reorganization of Core Pluripotency Factor Binding Before Global Epigenetic Reprogramming." *Cell Stem Cell* 17 (4). Elsevier BV: 462-70. <http://dx.doi.org/10.1016/j.stem.2015.07.005>. Kharchenko, Peter V, Michael Y Tolstorukov, and Peter J Park. 2008. "Design and Analysis of ChIP-Seq Experiments for DNA-Binding Proteins." *Nat Biotechnol*

26 (12). Nature Publishing Group: 1351-9. <http://dx.doi.org/10.1038/nbt.1508>. Langmead, Ben, and Steven L Salzberg. 2012. "Fast Gapped-Read Alignment with Bowtie 2." Nature Methods 9 (4). Nature Publishing Group: 357-59. <http://dx.doi.org/10.1038/nmeth.1923>. Picard Tools - by Broad Institute. 2016. <http://broadinstitute.github.io/picard/>.

### See Also

[sample\\_table\\_galonska](#) [input\\_galonska](#) [TSS\\_galonska](#)

---

DChIPRep	<i>DChIPRep: A package for differential analysis of histone modification ChIP-Seq profiles</i>
----------	--

---

### Description

The DChIPRep package provides functions to perform a differential analysis of histone modification profiles at base-pair resolution

### DChIPRep functions

The DChIPRep packages provides functions for data import `importData` and performing position wise tests. After data import, a `DChIPRepResults` object on which the function `runTesting` is run to perform the tests and add the result to the object. Then, plots can be created from this object. See the vignette for additional details: `vignette("DChIPRepVignette")`

---

DChIPRepResults-class	<i>DChIPRepResults object and constructor</i>
-----------------------	---

---

### Description

The DChIPRepResults contains a DESeqDataSet as obtained after the initial import.

### Usage

```
DChIPRepResults(object)
```

### Arguments

object	A DESeqDataSet
--------	----------------

### Value

A DChIPRepResult object.

### Examples

```
data(testData)
dcr <- DChIPRepResults(testData)
```

---

DESeq2Data	<i>Accessors for the 'DESeq2Data' slot of a DChIPRepResults object.</i>
------------	---

---

**Description**

The slot contains the DESeqDataSet as it is obtained after the initial data import. The DESeqDataSet contains the counts per position and the normalization factors as computed using the input counts.

**Usage**

```
## S4 method for signature 'DChIPRepResults'  
DESeq2Data(object)  
  
## S4 replacement method for signature 'DChIPRepResults,DESeqDataSet'  
DESeq2Data(object) <- value
```

**Arguments**

object	a DChIPRepResults object
value	A DESeqDataSet object

**Value**

the DESeq2Data object contained in the DChIPRepResults object

**Examples**

```
data(testData)  
dcr <- DChIPRepResults(testData)  
DESeq2Data(dcr)
```

---

exampleChipData	<i>An example ChIP data.</i>
-----------------	------------------------------

---

**Description**

An example ChIP data set that can be used with [importDataFromMatrices](#). Its associated sample table can be accessed via `data(data(exampleSampleTable))`.

**Usage**

```
data(exampleChipData)
```

**Format**

a matrix

**Value**

a matrix

**See Also**

[exampleSampleTable](#) [exampleInputData](#)

---

exampleInputData      *An example input data.*

---

**Description**

An example input data set that can be used with [importDataFromMatrices](#). Its associated sample table can be accessed via `data(data(exampleSampleTable))`.

**Usage**

```
data(exampleInputData)
```

**Format**

a matrix

**Value**

a matrix

**See Also**

[exampleSampleTable](#) [exampleChipData](#)

---

exampleSampleTable      *An example sample table data.frame*

---

**Description**

An example sample table

**Usage**

```
data(exampleSampleTable)
```

**Format**

a data.frame

**Value**

a data.frame

**See Also**

[exampleChipData](#) [exampleInputData](#)

---

FDRresults	<i>Accessor and setter for the 'FDRresults' slot of a DChIPRepResults object.</i>
------------	---

---

### Description

The slot contains the results of the FDR estimation as performed within the function `runTesting`. It is the complete output of the `fdrtool` function.

### Usage

```
## S4 method for signature 'DChIPRepResults'
FDRresults(object)

## S4 replacement method for signature 'DChIPRepResults,list'
FDRresults(object) <- value
```

### Arguments

object	a DChIPRepResults object
value	A DESeqDataSet object

### Value

a list containing the estimated false discovery rates

### Examples

```
data(testData)
dcr <- DChIPRepResults(testData)
dcr <- runTesting(dcr)
str(FDRresults(dcr))
```

---

getMATfromDataFrame	<i>Helper function to turn a data.frame into a matrix and remove the ID column.</i>
---------------------	---

---

### Description

This function takes a data.frame, with the genomic features (e.g. transcripts or genes) in the rows and the positions upstream and downstream of the TSS in the columns as well as a column ID containing a genomic feature ID and returns the data.frame with the ID column removed. The input for this function are tables obtained after running the Python import script.

### Usage

```
getMATfromDataFrame(df, ID = "name")
```

**Arguments**

df	the input data frame with positions in the columns and the genomic features in the rows.
ID	the name of the ID column to be removed.

**Value**

a matrix with the ID column removed

**Examples**

```
data(exampleSampleTable)
directory <- file.path(system.file("extdata", package="DChIPRep"))
df <- lapply(file.path(directory, exampleSampleTable$Input),
  read.delim)[[1]]
mat <- getMATfromDataFrame(df)
```

---

importData

---

*Import the data after running the Python script*


---

**Description**

This function imports the data from the count table files as returned by the accompanying Python script.

**Usage**

```
importData(sampleTable, directory = "", ID = "name", ...)
```

**Arguments**

sampleTable	a data.frame that has to contain the columns ChiP, Input, sampleID, upstream, downstream and condition. Each row of the table describes one experimental sample. Each row of the table describes one experimental sample. See <code>data(exampleSampleTable)</code> for an example table. and the vignette for further information.
directory	the directory relative to which the filenames are specified given as a character.
ID	character giving the name of the feature identifier column in the count tables. Defaults to "name"
...	parameters passed to <a href="#">summarizeCountsPerPosition</a>

**Value**

a DChIPRepResults object containing the imported data as a [DESeqDataSet](#).

**Examples**

```
data(exampleSampleTable)
directory <- file.path(system.file("extdata", package="DChIPRep"))
importedData <- importData(exampleSampleTable, directory)
```

---

`importDataFromMatrices`*Import the data from ChiP and input matrices*

---

### Description

This function imports the data from two matrices that contain counts summarized per position. It computes the normalization factors from the input (one per position) and creates a `DChIPRepResults` object.

### Usage

```
importDataFromMatrices(inputData, chipData, sampleTable)
```

### Arguments

<code>inputData</code>	a matrix containing the counts for the input per position.
<code>chipData</code>	a matrix containing the counts for the ChIP per position.
<code>sampleTable</code>	a data.frame that has to contain the columns <code>sampleID</code> , <code>upstream</code> , <code>downstream</code> and <code>condition</code> . Each row of the table describes one experimental sample. See <code>data(exampleSampleTable)</code> for an example table. and the vignette for further information.

### Details

The normalization factors are computed as  $t(t(inputData) * (covC/covI))$ , Where `covC` and `covI` contain the total sum of the ChIP and the input samples. Zero normalization factors can arise if the input has zero counts for certain positions. That's why input values equal to zero are set to 1 in order to always obtain valid `normalizationFactors`.

### Value

a `DChIPRepResults` object containing the imported data as a `DESeqDataSet`.

### Examples

```
data(exampleSampleTable)
data(exampleInputData)
data(exampleChipData)
imDataFromMatrices <- importDataFromMatrices(inputData = exampleInputData,
chipData = exampleChipData,
sampleTable = exampleSampleTable)
```



---

importData_soGGi	<i>Import the data from bam files directly</i>
------------------	--

---

## Description

This function imports the data from .bam files directly. It will return a matrix with one column per .bam file and the respective counts per position in the rows. It uses the function [regionPlot](#) from the package soGGi.

## Usage

```
importData_soGGi(bam_paths, TSS, fragment_lengths, sample_ids,
  distanceUp = 1000, distanceDown = 1500, ...)
```

## Arguments

bam_paths	a character vector of paths to the bam file(s) to be imported.
TSS	a <a href="#">GRanges (GenomicRanges-class)</a> (or a class that inherits from it) object containing the TSS of interest.
fragment_lengths	an integer vector of fragment lengths,
sample_ids	a character vector of sample ids for the .bam files. This can also be a factor.
distanceUp	Distance upstream from centre of the TSS provided.
distanceDown	Distance downstream from centre of the TSS provided.
...	additional arguments passed to <a href="#">regionPlot</a> .

## Details

In the example below, we use a subsampled .bam file (0.1 % of the reads) from the Galonska et. al. WCE (whole cell extract) H3Kme3 data and associated TSS near identified peaks. For additional details on the data, see [input\\_galonska](#) and [TSS\\_galonska](#).

## Value

a matrix that contains the position-wise profiles per .bam file in the columns.

## See Also

[regionPlot](#) [input\\_galonska](#) [TSS\\_galonska](#) [sample\\_table\\_galonska](#)

## Examples

```
## Not run:
data(sample_table_galonska)
data(TSS_galonska)
bam_dir <- file.path(system.file("extdata", package="DChIPRep"))
wce_bam <- "subsampled_0001_pc_SRR2144628_WCE_bowtie2_mapped-only_XS-filt_no-dups.bam"
mat_wce <- importData_soGGi(bam_paths = file.path(bam_dir, wce_bam),
  TSS = TSS_galonska,
  fragment_lengths = sample_table_galonska$input_fragment_length[1],
  sample_ids = sample_table_galonska$input[1],
```

```

        paired = FALSE,
        removeDup=FALSE
    )
    head(mat_wce)

    ## End(Not run)

```

---

input\_galonska      *Another example Input data set that can be used with [importDataFromMatrices](#) from rom Galonska et. al., 2015.*

---

### Description

The matrix contains the whole cell extract (WCE) data for H3Kme3 from the paper in each of the four columns, since this is the only input data provided for all 4 samples. For additional information see the documentation of [chip\\_galonska](#).

### Usage

```
data(input_galonska)
```

### Format

a matrix

### Value

a matrix

### See Also

[chip\\_galonska](#) [sample\\_table\\_galonska](#) [TSS\\_galonska](#)

---

plotProfiles      *Produce a TSS plot of the two conditions in the data*

---

### Description

This function plots the positionwise mean of the log<sub>2</sub> of the normalized counts of the two conditions after [runTesting](#) has been run on a DChIPRepResults object.

### Usage

```

## S4 method for signature 'DChIPRepResults'
plotProfiles(object, meanFunction = robust_mean,
  ...)

```

**Arguments**

object a DChIPRepResults object after [runTesting](#)  
 meanFunction a function to compute the positionwise mean per group, defaults to a Huber estimator of the mean.  
 ... additional parametes for plotting (NOT YET IMPLEMENTED)

**Value**

a ggplot2 object

**Examples**

```
if (requireNamespace("mgcv", quietly=TRUE)) {
  data(testData)
  dcr <- DChIPRepResults(testData)
  dcr <- runTesting(dcr)
  plotProfiles(dcr)
}
```

---

plotSignificance	<i>Produce a plot that colors the positions identified as significant</i>
------------------	---

---

**Description**

This function plots the positionwise mean of the two conditions after [runTesting](#) has been run on a DChIPRepResults object. The points corresponding to significant positions are colored black in both of the conditions. The function returns the plot as a ggplot2 object that can be modified afterwards.

**Usage**

```
## S4 method for signature 'DChIPRepResults'
plotSignificance(object,
  meanFunction = robust_mean, lfdrThresh = 0.2, ...)
```

**Arguments**

object a DChIPRepResults object after [runTesting](#)  
 meanFunction a function to compute the positionwise mean per group, defaults to a Huber estimator of the mean.  
 lfdrThresh Threshold for the local FDR  
 ... additional parameters for plotting (NOT YET IMPLEMENTED)

**Value**

a ggplot2 object

**Examples**

```
data(testData)
dcr <- DChIPRepResults(testData)
dcr <- runTesting(dcr)
plotSignificance(dcr)
```

---

resultsDChIPRep	<i>Accessors and setter for the 'results' slot of a DChIPRepResults object.</i>
-----------------	---

---

### Description

The slot contains the results of the position wise tests in a data.frame after running the function `runTesting`. It is a modified output of the `results` function of the DESeq2 package.

### Usage

```
## S4 method for signature 'DChIPRepResults'
resultsDChIPRep(object)

## S4 replacement method for signature 'DChIPRepResults,list'
resultsDChIPRep(object) <- value
```

### Arguments

object	a DChIPRepResults object
value	A DESeqDataSet object

### Value

a data.frame containing the results of the position wise tests

### Examples

```
data(testData)
dcr <- DChIPRepResults(testData)
dcr <- runTesting(dcr)
head(resultsDChIPRep(dcr))
```

---

robust_mean	<i>Use a huber type estimator to produce a robust mean</i>
-------------	--

---

### Description

This function uses a Huber type estimator as implemented in the function `smhuber` from the package <https://cran.r-project.org/web/packages/smoothmest/smoothmest>. This is used to summarize the profiles across replicates. We provide a wrapper around the original function that catches the case that we want to produce a mean of a single value. It is used in the functions `plotProfiles` and `plotSignificance`.

### Usage

```
robust_mean(x)
```

**Arguments**

x                    a numerical vector

**Value**

a robust mean of a numerical vector

**Examples**

```
data(testData)
robust_mean(counts(testData[, 1]))

x <- rcauchy(10)
robust_mean(x)
```

---

runTesting

*Run the tests on a DChIPRepResults object.*


---

**Description**

This function runs the testing on a DChIPRepResults object. It adds the FDR calculations and the result table to the DChIPRepResults object.

**Usage**

```
## S4 method for signature 'DChIPRepResults'
runTesting(object, lfcThreshold = 0.05,
           plotFDR = FALSE, ...)
```

**Arguments**

object            A [DChIPRepResults](#) object.

lfcThreshold    A non-negative threshold value, which determines the null hypothesis. The null hypothesis is  $H_0 : |\log_2(FC)| > \text{lfcThreshold}$

plotFDR         If set to TRUE a plot showing the estimated FDRs will be displayed

...              not used currently

**Value**

a modified [DChIPRepResults](#) object containing the testing results

**See Also**

[resultsDChIPRep](#)

**Examples**

```
data(testData)
dcr <- DChIPRepResults(testData)
dcr <- runTesting(dcr)
```

---

sample\_table\_galonska *Another example sample table based on data from rom Galonska et. al., 2015.*

---

**Description**

This table contains the sample annotation for the H3Kme3 data from Galonska et. al., 2015. For additional information see the documentation of [chip\\_galonska](#).

**Usage**

```
data(sample_table_galonska)
```

**Format**

a data.frame

**Value**

a data.frame

**See Also**

[chip\\_galonska](#) [input\\_galonska](#) [TSS\\_galonska](#)

---

show *prints the DESeq2Data slot of the DChIPRepResults object*

---

**Description**

prints the data

**Usage**

```
## S4 method for signature 'DChIPRepResults'  
show(object)
```

**Arguments**

object            A DChIPRepResults object

**Value**

A compact representation of the DChIPRepResults object

**Examples**

```
data(testData)  
dcr <- DChIPRepResults(testData)  
dcr  
dcr <- runTesting(dcr)  
dcr
```

---

`summarizeCountsPerPosition`*Helper function to summarize the counts per position*

---

## Description

This function takes a matrix of counts, with the genomic features (e.g. transcripts or genes) in the rows and the positions upstream and downstream of the TSS in the columns and returns a vector with the summarized counts per position.

## Usage

```
summarizeCountsPerPosition(mat, ct = 0, trim = 0.15)
```

## Arguments

<code>mat</code>	the input matrix with positions in the columns and the genomic features in the rows.
<code>ct</code>	the count threshold to use.
<code>trim</code>	the trimming percentage for the trimmed mean.

## Details

The summary per condition is computed as a trimmed mean per position. First, counts greater than `ct` are removed and then a trimmed mean with a trimming percentage of `trim` is computed on the log scale. This mean is then exponentiated again and multiplied by the total number of features passing the threshold `ct` per position. If a position contains only zero counts, its mean is returned as zero.

## Value

a vector containing the summarized counts per condition

## Examples

```
data(exampleSampleTable)
directory <- file.path(system.file("extdata", package="DChIPRep"))
df <- lapply(file.path(directory, exampleSampleTable$Input),
  read.delim)[[1]]
mat <- getMATfromDataFrame(df)
summaryPerPos <- summarizeCountsPerPosition(mat)
```

---

testData	<i>A test DESeqDataSet</i>
----------	----------------------------

---

**Description**

test data to check the functions

**Usage**

```
data(testData)
```

**Format**

a DESeqDataSet

**Value**

a DESeqDataSet

---

TSS_galonska	<i>TSS around called peak regions from Galonska et. al.</i>
--------------	---

---

**Description**

This data contains mouse mm9 TSS close to called peak regions for H3Kme3 from Galonska et al. The original peak lists are from GEO (GSE56312) and have been merged into a common peaklist and then annotated to the closest mm9 TSS using [annotatePeakInBatch](#).

**Usage**

```
data(exampleChipData)
```

**Format**

an annoGR object from the package CHIPpeakAnno.

**Value**

an annoGR object from the package CHIPpeakAnno.

**See Also**

[chip\\_galonska](#) [input\\_galonska](#) [sample\\_table\\_galonska](#)



# Index

## \*Topic **datasets**

- chip\_galonska, 2
  - exampleChipData, 4
  - exampleInputData, 5
  - exampleSampleTable, 5
  - input\_galonska, 10
  - sample\_table\_galonska, 14
  - testData, 16
  - TSS\_galonska, 16
- annotatePeakInBatch, 16
- chip\_galonska, 2, 10, 14, 16
- DChIPRep, 3
- DChIPRep-package (DChIPRep), 3
- DChIPRepResults, 3, 13
- DChIPRepResults
- (DChIPRepResults-class), 3
- DChIPRepResults-class, 3
- DESeq2Data, 4
- DESeq2Data, DChIPRepResults-method
- (DESeq2Data), 4
- DESeq2Data<- (DESeq2Data), 4
- DESeq2Data<- , DChIPRepResults, DESeqDataSet-method
- (DESeq2Data), 4
- DESeqDataSet, 7, 8
- exampleChipData, 4, 5
- exampleInputData, 5, 5
- exampleSampleTable, 5, 5
- FDRresults, 6
- FDRresults, DChIPRepResults-method
- (FDRresults), 6
- FDRresults<- (FDRresults), 6
- FDRresults<- , DChIPRepResults, list-method
- (FDRresults), 6
- fdrtool, 6
- GenomicRanges-class, 9
- getMATfromDataFrame, 6
- importData, 3, 7
- importData\_soGGi, 9
- importDataFromMatrices, 2, 4, 5, 8, 10
- input\_galonska, 3, 9, 10, 14, 16
- plotProfiles, 10, 12
- plotProfiles, DChIPRepResults-method
- (plotProfiles), 10
- plotSignificance, 11, 12
- plotSignificance, DChIPRepResults-method
- (plotSignificance), 11
- regionPlot, 9
- results, 12
- resultsDChIPRep, 12, 13
- resultsDChIPRep, DChIPRepResults-method
- (resultsDChIPRep), 12
- resultsDChIPRep<- (resultsDChIPRep), 12
- resultsDChIPRep<- , DChIPRepResults, list-method
- (resultsDChIPRep), 12
- robust\_mean, 12
- runTesting, 3, 6, 10–12, 13
- runTesting, DChIPRepResults-method
- (runTesting), 13
- sample\_table\_galonska, 3, 9, 10, 14, 16
- show, 14
- show, DChIPRepResults-method (show), 14
- smhuber, 12
- summarizeCountsPerPosition, 7, 15
- testData, 16
- TSS\_galonska, 3, 9, 10, 14, 16