# Package 'compartmap'

October 23, 2020

**Type** Package

**Title** A/B compartment inference from ATAC-seq and methylation array data

**Description** Compartmap performs shrunken A/B compartment inference from ATAC-seq and methylation arrays.

**Version** 1.7.0

**Date** 2018-10-28

**URL** https://github.com/biobenkj/compartmap

**BugReports** https://github.com/biobenkj/compartmap/issues

**Encoding** UTF-8

**License** GPL-3 + file LICENSE

**biocViews** ImmunoOncology, Genetics, Epigenetics, ATACSeq, MethylSeq, MethylationArray

**Depends** R (>= 3.5.0), minfi, Homo.sapiens, mixOmics

**Imports** SummarizedExperiment, GenomicRanges, gtools, parallel

**Suggests** covr, testthat, knitr

**RoxygenNote** 6.1.0

**Roxygen** list(markdown = TRUE)

**VignetteBuilder** knitr

**git_url** https://git.bioconductor.org/packages/compartmap

**git_branch** master

**git_last_commit** c8c8076

**git_last_commit_date** 2020-04-27

**Date/Publication** 2020-10-22

**Author** Benjamin Johnson [aut, cre],
Tim Triche [aut],
Kasper Hansen [aut],
Jean-Philippe Fortin [aut]

**Maintainer** Benjamin Johnson <ben.johnson@vai.org>

# R topics documented:



---

`array.data.chr14` *Example Illumina 450k methylation array data for compartmap*

---

## Description

This data was generated using the data from the reference via the sesamize function from the SeSAMe package.

## Usage

```
data(meth_array_450k_chr14, package = "compartmap")
```

## Author(s)

Benjamin K Johnson <ben.johnson@vai.org>

## References

https://f1000research.com/articles/5-1281/v3

---

`filtered.data.chr14` *Example ATAC-seq data for compartmap*

---

## Description

This data was generated using the data from the reference via bwa mem and pre-processing the data using the ATACseeker package.

## Usage

```
data(bulkATAC_raw_filtered_chr14, package = "compartmap")
```

## Author(s)

Benjamin K Johnson <ben.johnson@vai.org>

## References

https://trace.ncbi.nlm.nih.gov/Traces/sra/?study=SRP082417

---

`fisherZ` *Fisher's Z transformation*

---

## Description

`fisherZ` returns (squeezed) Fisher's Z transformed Pearson's r

## Usage

```
fisherZ(cormat)
```

## Arguments

| | |
|---|---|
| `cormat` | Pearson correlation matrix |

## Details

This function returns (squeezed) Fisher's Z transformed Pearson's r

## Value

Fisher Z transformed Pearson correlations

## Examples

```
#Generate a random binary (-1, 1) matrix
mat <- matrix(sample(c(1,-1), 10000, replace = TRUE), ncol = 100)

#Correct matrix diag
diag(mat) <- 1

#Transform
mat.transform <- fisherZ(mat)
```

---

`getABSignal` *Calculate Pearson correlations of smoothed eigenvectors*

---

## Description

This function is used to generate a list x to be passed to getABSignal

## Usage

```
getABSignal(x, k = 5, iter = 2, squeeze = FALSE)
```

### Arguments

| | |
|---|---|
| x | A list object from getCorMatrix |
| k | Value of k for smoothing (default = 2) |
| iter | Number of iterations for moving average smoothing (default = 2) |
| squeeze | Whether squeezing was used (implies Fisher's Z transformation) |

### Value

A list x to pass to getABSignal

### Examples

```
library(GenomicRanges)
library(Homo.sapiens)
library(mixOmics)

#Generate random genomic intervals of 1-1000 bp on chr1-22
#Modified from https://www.biostars.org/p/225520/
random_genomic_int <- data.frame(chr = rep("chr14", 100))
random_genomic_int$start <- apply(random_genomic_int, 1, function(x) { round(runif(1, 0, seqlengths(Homo.sapi
random_genomic_int$end <- random_genomic_int$start + runif(1, 1, 1000)
random_genomic_int$strand <- "*"

#Generate random counts
counts <- rnbinom(1000, 1.2, 0.4)

#Build random counts for 10 samples
count.mat <- matrix(sample(counts, nrow(random_genomic_int) * 10, replace = FALSE), ncol = 10)
colnames(count.mat) <- paste0("sample_", seq(1:10))

#Bin counts
bin.counts <- getBinMatrix(count.mat, makeGRangesFromDataFrame(random_genomic_int), chr = "chr14", genome = "h

#Calculate correlations
bin.cor.counts <- getCorMatrix(bin.counts)

#Get A/B signal
absignal <- getABSignal(bin.cor.counts)
```

---

## getArrayABsignal *Estimate A/B compartments from methylation array data*

---

### Description

getArrayABsignal returns estimated A/B compartments from methylation array data.

### Usage

```
getArrayABsignal(obj, res = 1e+06, parallel = FALSE, allchrs = FALSE,
  chr = NULL, targets = NULL, ...)
```

### Arguments

| | |
|---|---|
| obj | Input GenomicRatioSet object |
| res | Compartment resolution (in bp) |
| parallel | Should the inference be done in parallel? |
| allchrs | Whether all autosomes should be used for A/B inference |
| chr | Specific chromosomes to analyze |
| targets | Specify samples to use as shrinkage targets |
| ... | Additional arguments |

### Details

This function is modified from the minfi::compartments to infer A/B compartments from array data

### Value

A p x n matrix (samples as columns and compartments as rows) of compartments

---

## getATACABsignal *Estimate A/B compartments from ATAC-seq data*

---

### Description

getATACABsignal returns estimated A/B compartments from methylation array data.

### Usage

```
getATACABsignal(obj, res = 1e+06, parallel = FALSE, allchrs = FALSE,
  chr = NULL, targets = NULL, ...)
```

### Arguments

| | |
|---|---|
| obj | Input GenomicRatioSet object |
| res | Compartment resolution (in bp) |
| parallel | Should the inference be done in parallel? |
| allchrs | Whether all autosomes should be used for A/B inference |
| chr | Specify a chromosome to analyze |
| targets | Specify samples as shrinkage targets |
| ... | Additional arguments |

### Details

This function estimates A/B compartments shrinking towards a global mean of targets or across samples

### Value

A p x n matrix (samples as columns and compartments as rows) of compartments

## Examples

```
library(GenomicRanges)
library(SummarizedExperiment)
library(Homo.sapiens)

data(bulkATAC_raw_filtered_chr14, package = "compartmap")
atac_compartments <- getATACABsignal(filtered.data.chr14, chr = "chr14", genome = "hg19")
```

---

`getBinMatrix` *Generate bins for A/B compartment estimation*

---

## Description

Generate bins across a user defined chromosome for A/B compartment estimation. A/B compartment estimation can be used for non-supported genomes if chr.end is set.

## Usage

```
getBinMatrix(x, genloc, chr = "chr1", chr.start = 0, chr.end = NULL,
  res = 1e+05, FUN = sum, genome = "hg19")
```

## Arguments

| | |
|---|---|
| `x` | A p x n matrix where p (rows) = loci and n (columns) = samples/cells |
| `genloc` | GRanges object that contains corresponding genomic locations of the loci |
| `chr` | Chromosome to be analyzed |
| `chr.start` | Starting position (in bp) to be analyzed |
| `chr.end` | End position (in bp) to be analyzed |
| `res` | Binning resolution (in bp) |
| `FUN` | Function to be used to summarize information within a bin |
| `genome` | Genome corresponding to the input data ("hg19" or "mm10") |

## Details

This function is used to generate a list object to be passed to getCorMatrix

## Value

A list object to pass to getCorMatrix

## Examples

```
library(GenomicRanges)
library(Homo.sapiens)

#Generate random genomic intervals of 1-1000 bp on chr1-22
#Modified from https://www.biostars.org/p/225520/
random_genomic_int <- data.frame(chr = rep("chr14", 100))
random_genomic_int$start <- apply(random_genomic_int, 1, function(x) { round(runif(1, 0, seqlengths(Homo.sapi
```

```
random_genomic_int$end <- random_genomic_int$start + runif(1, 1, 1000)
random_genomic_int$strand <- "*"

#Generate random counts
counts <- rnbinom(1000, 1.2, 0.4)

#Build random counts for 10 samples
count.mat <- matrix(sample(counts, nrow(random_genomic_int) * 10, replace = FALSE), ncol = 10)
colnames(count.mat) <- paste0("sample_", seq(1:10))

#Bin counts
bin.counts <- getBinMatrix(count.mat, makeGRangesFromDataFrame(random_genomic_int), chr = "chr14", genome = "h
```

---

`getCompartments` *Estimate A/B compartments*

---

## Description

`getCompartments` returns estimated A/B compartments from ATAC-seq and methylation array data

## Usage

```
getCompartments(obj, type = c("atac", "array"), res = 1e+06,
  parallel = FALSE, chrs = "chr1", genome = "hg19", targets = NULL,
  run_examples = FALSE, ...)
```

## Arguments

| | |
|---|---|
| `obj` | The object with which to perform compartment inference |
| `type` | The type of data that obj represents (e.g. atac or array) |
| `res` | Resolution of compartments in base pairs (default is 1e6) |
| `parallel` | Should the estimates be done in parallel (default is FALSE) |
| `chrs` | Chromosomes to operate on (can be individual chromosomes, a list of chromosomes, or all) |
| `genome` | Genome to use (default is hg19) |
| `targets` | Specify samples to use as shrinkage targets |
| `run_examples` | Whether to run ATAC-seq and 450k example analysis |
| `...` | Other parameters to pass to internal functions |

## Details

This is a wrapper function to perform A/B compartment inference. Compartmentalizer implements a Stein estimator to shrink per-sample compartment estimates towards a global mean. The expected input for this function can be generated using packages like SeSAMe and ATACseeker.

## Value

A p x n matrix (samples as columns and compartments as rows) to pass to embed_compartments

## Examples

```
library(GenomicRanges)
library(SummarizedExperiment)
library(Homo.sapiens)

#ATAC-seq data
data(bulkATAC_raw_filtered_chr14, package = "compartmap")
atac_compartments <- getCompartments(filtered.data.chr14, type = "atac", parallel = FALSE, chrs = "chr14")
## Not run:
#450k data
data(meth_array_450k_chr14, package = "compartmap")
array_compartments <- getCompartments(array.data.chr14, type = "array", parallel = FALSE, chrs = "chr14")
## End(Not run)
```

---

`getCorMatrix` *Calculate Pearson correlations of a binned matrix*

---

## Description

This function is used to generate a list object to be passed to getABSignal

## Usage

```
getCorMatrix(binmat, squeeze = FALSE)
```

## Arguments

| | |
|---|---|
| `binmat` | A binned matrix list object from getBinMatrix |
| `squeeze` | Whether to squeeze the matrix for Fisher's Z transformation |

## Value

A list object to pass to getABSignal

## Examples

```
library(GenomicRanges)
library(Homo.sapiens)

#Generate random genomic intervals of 1-1000 bp on chr1-22
#Modified from https://www.biostars.org/p/225520/
random_genomic_int <- data.frame(chr = rep("chr14", 100))
random_genomic_int$start <- apply(random_genomic_int, 1, function(x) { round(runif(1, 0, seqlengths(Homo.sapi
random_genomic_int$end <- random_genomic_int$start + runif(1, 1, 1000)
random_genomic_int$strand <- "*"

#Generate random counts
counts <- rnbinom(1000, 1.2, 0.4)

#Build random counts for 10 samples
count.mat <- matrix(sample(counts, nrow(random_genomic_int) * 10, replace = FALSE), ncol = 10)
```

```
colnames(count.mat) <- paste0("sample_", seq(1:10))

#Bin counts
bin.counts <- getBinMatrix(count.mat, makeGRangesFromDataFrame(random_genomic_int), chr = "chr14", genome = "h

#Calculate correlations
bin.cor.counts <- getCorMatrix(bin.counts)
```

---

`ifisherZ` *Fisher's Z transformation*

---

## Description

`fisherZ` returns the inverse (squeezed) Fisher's Z transformed Pearson's r. This will fail if a matrix is used as input instead of a vector.

## Usage

```
ifisherZ(cormat)
```

## Arguments

| | |
|---|---|
| `cormat` | vector of Fisher's Z transformed Pearson correlations or an eignevector |

## Details

This function returns the inverse (squeezed) Fisher's Z transformed Pearson's r

## Value

Back transformed Fisher's Z

## Examples

```
#Generate a random binary (-1, 1) matrix
mat <- matrix(sample(c(1,-1), 10000, replace = TRUE), ncol = 100)

#Correct matrix diag
diag(mat) <- 1

#Transform
mat.transform <- fisherZ(mat)

#Back transform
mat.transform.inverse <- apply(mat.transform, 1, ifisherZ)
```

plotAB *Plots A/B compartment estimates on a per chromosome basis*

## Description

Plot A/B compartments bins

## Usage

```
plotAB(x, main = "", ylim = c(-1, 1), unitarize = FALSE,
  reverse = FALSE, top.col = "deeppink4", bot.col = "grey50")
```

## Arguments

| | |
|---|---|
| x | The matrix obejct returned from getCompartments |
| main | Title for the plot |
| ylim | Y-axis limits (default is -1 to 1) |
| unitarize | Should the data be unitarized? (not explicitly necessary for arrays) |
| reverse | Reverse the sign of the PC values? |
| top.col | Top (pos. PC values) chromatin color to be plotted |
| bot.col | Bottom (neg. PC values) chromatin color to be plotted |

## Value

invisibly, the compartment estimates from the plot

## Examples

```
library(GenomicRanges)
library(Homo.sapiens)

#Generate random genomic intervals of 1-1000 bp on chr1-22
#Modified from https://www.biostars.org/p/225520/
random_genomic_int <- data.frame(chr = rep("chr14", 100))
random_genomic_int$start <- apply(random_genomic_int, 1, function(x) { round(runif(1, 0, seqlengths(Homo.sapi
random_genomic_int$end <- random_genomic_int$start + runif(1, 1, 1000)
random_genomic_int$strand <- "*"

#Generate random counts
counts <- rnbinom(1000, 1.2, 0.4)

#Build random counts for 10 samples
count.mat <- matrix(sample(counts, nrow(random_genomic_int) * 10, replace = FALSE), ncol = 10)
colnames(count.mat) <- paste0("sample_", seq(1:10))

#Bin counts
bin.counts <- getBinMatrix(count.mat, makeGRangesFromDataFrame(random_genomic_int), chr = "chr14", genome = "h

#Calculate correlations
bin.cor.counts <- getCorMatrix(bin.counts)
```

```
#Get A/B signal
absignal <- getABSignal(bin.cor.counts)

#Plot the A/B signal
par(mar=c(1,1,1,1))
par(mfrow=c(1,1))
plotAB(absignal$pc, ylim = c(-0.2, 0.2), unitarize = TRUE)

## Not run:
#If plotting individual A/B signals using output from getCompartments
#Note: this function currently only supports plotting individual chromosomes from single samples
bin.chr1.ab <- getCompartments(data, "array", chrs = "chr1", genome = "hg19")

#For 7 samples
#Adjust ylim as necessary
par(mar=c(1,1,1,1))
par(mfrow=c(7,1))
plotAB(bin.chr1.ab[,1], ylim = c(-0.2, 0.2), unitarize = TRUE)
plotAB(bin.chr1.ab[,2], ylim = c(-0.2, 0.2), unitarize = TRUE, top.col = "goldenrod")
plotAB(bin.chr1.ab[,3], ylim = c(-0.2, 0.2), unitarize = TRUE, top.col = "darkblue")
plotAB(bin.chr1.ab[,4], ylim = c(-0.2, 0.2), unitarize = TRUE, top.col = "red")
plotAB(bin.chr1.ab[,5], ylim = c(-0.2, 0.2), unitarize = TRUE, top.col = "black")
plotAB(bin.chr1.ab[,6], ylim = c(-0.2, 0.2), unitarize = TRUE, top.col = "cyan")
plotAB(bin.chr1.ab[,7], ylim = c(-0.2, 0.2), unitarize = TRUE, top.col = "seagreen")

## End(Not run)
```

# Index