Package 'normr'

December 17, 2025

Type Package

Title Normalization and difference calling in ChIP-seq data

Version 1.37.0

Date 2021-09-21

Author Johannes Helmuth [aut, cre], Ho-Ryun Chung [aut]

Maintainer Johannes Helmuth <johannes.helmuth@laborberlin.com>

Description Robust normalization and difference calling procedures for ChIP-seq and alike data. Read counts are modeled jointly as a binomial mixture model with a user-specified number of components. A fitted background estimate accounts for the effect of enrichment in certain regions and, therefore, represents an appropriate null hypothesis. This robust background is used to identify significantly enriched or depleted regions.

License GPL-2

Depends R (>= 3.3.0)

LinkingTo Rcpp

SystemRequirements C++11

Imports methods, stats, utils, grDevices, parallel, GenomeInfoDb, GenomicRanges, IRanges, Rcpp (>= 0.11), qvalue (>= 2.2), bamsignals (>= 1.4), rtracklayer (>= 1.32)

Suggests BiocStyle, testthat (>= 1.0), knitr, rmarkdown

Enhances BiocParallel

biocViews Bayesian, DifferentialPeakCalling, Classification, DataImport, ChIPSeq, RIPSeq, FunctionalGenomics, Genetics, MultipleComparison, Normalization, PeakDetection, Preprocessing, Alignment

VignetteBuilder knitr

LazyData true

NeedsCompilation yes

Collate 'NormRFit.R' 'methods.R' 'NormRCountConfig.R' 'RcppExports.R'

RoxygenNote 6.0.1

URL https://github.com/your-highness/normR

BugReports https://github.com/your-highness/normR/issues

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

git_branch devel

git_last_commit 76e2183

git_last_commit_date 2025-10-29

Repository Bioconductor 3.23

Date/Publication 2025-12-17

Contents

`diffR` *Difference Calling between conditional ChIP-seq data in normR with diffR*

Description

Difference calling between `treatment` (ChIP-seq 1) and `control` (ChIP-seq 2) in normR is done by fitting background and two conditional enrichments simultaenously. Therefore, a mixture of three binomials is fit to the data with Expectation Maximization (EM). After convergence of the EM, the fitted background component is used to calculate significance for treatment and control count pair. Based on this statistic, user can extract significantly enriched/depleted regions in a condition with a desired significance level. These regions can be further analyzed within R or exported (see `NormRFit-class`). Furthermore, diffR calculates a standardized conditional-specific enrichment given the fitted background component. See also Details

Usage

```
diffR(treatment, control, genome, ...)

## S4 method for signature 'integer,integer,GenomicRanges'
diffR(treatment, control, genome,
  procs = 1L, verbose = TRUE, eps = 1e-05, iterations = 10,
  minP = 0.05)

## S4 method for signature 'character,character,GenomicRanges'
diffR(treatment, control, genome,
  countConfig = countConfigSingleEnd(), procs = 1L, verbose = TRUE,
  eps = 1e-05, iterations = 10, minP = 0.05)

## S4 method for signature 'character,character,data.frame'
diffR(treatment, control, genome,
  countConfig = countConfigSingleEnd(), procs = 1L, verbose = TRUE,
```

```
  eps = 1e-05, iterations = 10, minP = 0.05)

## S4 method for signature 'character,character,character'
diffR(treatment, control,
  genome = "", countConfig = countConfigSingleEnd(), procs = 1L,
  verbose = TRUE, eps = 1e-05, iterations = 10, minP = 0.05)
```

Arguments

treatment	An integer vector of treatment counts or a character pointing to the treatment bam file. In the latter case an "treatment.bai" index file should exist in the same folder.
control	An integer vector of control counts or a character pointing to the control bam file. In the latter case an "control.bai" index file should exist in the same folder.
genome	Either NULL (default), a character specifying a USCS genome identifier, a data.frame consisting of two columns or a GenomicRanges specifying the genomic regions (see Details).
...	Optional arguments for the respective implementations of diffR.
procs	An integer giving the number of parallel threads to use.
verbose	A logical indicating whether verbose output is desired.
eps	A numeric specifying the T Filter threshold and the threshold for EM convergence, *i.e.* the minimal difference in log-likelihood in two consecutive steps.
iterations	An integer specifying how many times the EM is initialized with random model parameters.
minP	An integer controlling the threshold for the T method when filtering low power regions, i.e. regions with low counts.
countConfig	A NormRCountConfig object specifying bam counting parameters for read count retrieval. See Details.

Details

Supplied count vectors for treatment and control should be of same length and of type integer.

For convenience, read count vectors can be obtained directly from bam files. In this case, please specify a bam file for treatment and control each and a genome. Bam files should be indexed using samtools (*i.e.* samtools index file file.bai). Furthermore, bam files should contain a valid header with given chromosome names. If genome == NULL(default), chromosome names will be read from treatment bamheader. Please be aware that bamheader might contain irregular contigs and chrM which influence the fit. Also be sure that treatment and control contain the same chromosomes. Otherwise an error will be thrown. If genome is a character, fetchExtendedChromInfoFromUCSC is used to resolve this to a valid UCSC genome identifier (see https://genome.ucsc.edu/cgi-bin/hgGateway for available genomes). In this case, only assembled molecules will be considered (no circular). Please check if your bam files obey this annotation. If genome is a data.frame, it represents the chromosome specification. The first column will be used as chromosome ID and the second column will be used as the chromosome lengths. If genome is a GenomicRanges, it should contain the equally sized genomic loci to count in, e.g. promoters. The binsize in the supplied NormRCountConfig is ignore in this case.

bamCountConfig is an instance of class NormRCountConfig specifying settings for read counting on bam files. You can specify the binsize, minimum mapping quality, shifting of read ends etc.. Please refer to NormRFit-class for details.

Value

A NormRFit container holding results of the fit with type diffR.

Author(s)

Johannes Helmuth <helmuth@molgen.mpg.de>

See Also

NormRFit-class for functions on accessing and exporting the diffR fit. NormRCountConfig-class for configuration of the read counting procedure (binsize, mapping quality,...).

Examples

```
require(GenomicRanges)

### enrichR(): Calling Enrichment over Input
#load some example bamfiles
input <- system.file("extdata", "K562_Input.bam", package="normr")
chipK4 <- system.file("extdata", "K562_H3K4me3.bam", package="normr")
#region to count in (example files contain information only in this region)
gr <- GRanges("chr1", IRanges(seq(22500001, 25000000, 1000), width = 1000))
#configure your counting strategy (see BamCountConfig-class)
countConfiguration <- countConfigSingleEnd(binsize = 1000,
                                           mapq = 30, shift = 100)
#invoke enrichR to call enrichment
enrich <- enrichR(treatment = chipK4, control = input,
                  genome = gr,  countConfig = countConfiguration,
                  iterations = 10, procs = 1, verbose = TRUE)
#inspect the fit
enrich
summary(enrich)

## Not run:
#write significant regions to bed
#exportR(enrich, filename = "enrich.bed", fdr = 0.01)
#write normalized enrichment to bigWig
#exportR(enrich, filename = "enrich.bw")
## End(**Not run**)

### diffR(): Calling differences between two conditions
chipK36 <- system.file("extdata", "K562_H3K36me3.bam", package="normr")
diff <- diffR(treatment = chipK36, control = chipK4,
              genome = gr,  countConfig = countConfiguration,
              iterations = 10, procs = 1, verbose = TRUE)
summary(diff)

### regimeR(): Identification of broad and peak enrichment
regime <- regimeR(treatment = chipK36, control = input, models = 3,
                  genome = gr,  countConfig = countConfiguration,
                  iterations = 10, procs = 1, verbose = TRUE)
summary(regime)
```

enrichR *Enrichment Calling on ChIP-seq data in normR with enrichR*

Description

Enrichment calling between `treatment` (ChIP-seq) and `control` (Input) in normR is done by fitting background and enrichment simultaenously. Therefore, a mixture of two binomials is fit to the data with Expectation Maximization (EM). After convergence of the EM, the fitted background component is used to calculate significance for treatment and control count pair. Based on this statistic, user can extract significantly enriched regions with a desired significance level. These regions can be further analyzed within R or exported (see `NormRFit-class`). Furthermore, enrichR calculates a standardized enrichment given the fitted background component. See also Details

Usage

```
enrichR(treatment, control, genome, ...)

## S4 method for signature 'integer,integer,GenomicRanges'
enrichR(treatment, control, genome,
  procs = 1L, verbose = TRUE, eps = 1e-05, iterations = 10,
  minP = 0.05)

## S4 method for signature 'character,character,GenomicRanges'
enrichR(treatment, control,
  genome, countConfig = countConfigSingleEnd(), procs = 1L,
  verbose = TRUE, eps = 1e-05, iterations = 10, minP = 0.05)

## S4 method for signature 'character,character,data.frame'
enrichR(treatment, control, genome,
  countConfig = countConfigSingleEnd(), procs = 1L, verbose = TRUE,
  eps = 1e-05, iterations = 10, minP = 0.05)

## S4 method for signature 'character,character,character'
enrichR(treatment, control, genome,
  countConfig = countConfigSingleEnd(), procs = 1L, verbose = TRUE,
  eps = 1e-05, iterations = 10, minP = 0.05)
```

Arguments

`treatment`	An `integer` vector of treatment counts or a `character` pointing to the treatment bam file. In the latter case an "`treatment`.bai" index file should exist in the same folder.
`control`	An `integer` vector of control counts or a character pointing to the control bam file. In the latter case an "`control`.bai" index file should exist in the same folder.
`genome`	Either `NULL` (default), a `character` specifying a USCS genome identifier, a data.frame consisting of two columns or a GenomicRanges specifying the genomic regions (see Details).
`...`	Optional arguments for the respective implementations of `enrichR`.
`procs`	An `integer` giving the number of parallel threads to use.

verbose	A logical indicating whether verbose output is desired.
eps	A numeric specifying the T Filter threshold and the threshold for EM convergence, *i.e.* the minimal difference in log-likelihood in two consecutive steps.
iterations	An integer specifying how many times the EM is initialized with random model parameters.
minP	An integer controlling the threshold for the T method when filtering low power regions, i.e. regions with low counts.
countConfig	A NormRCountConfig object specifying bam counting parameters for read count retrieval. See Details.

Details

Supplied count vectors for treatment and control should be of same length and of type integer.

For convenience, read count vectors can be obtained directly from bam files. In this case, please specify a bam file for treatment and control each and a genome. Bam files should be indexed using samtools (*i.e.* samtools index file file.bai). Furthermore, bam files should contain a valid header with given chromosome names. If genome == NULL(default), chromosome names will be read from treatment bamheader. Please be aware that bamheader might contain irregular contigs and chrM which influence the fit. Also be sure that treatment and control contain the same chromosomes. Otherwise an error will be thrown. If genome is a character, fetchExtendedChromInfoFromUCSC is used to resolve this to a valid UCSC genome identifier (see https://genome.ucsc.edu/cgi-bin/hgGateway for available genomes). In this case, only assembled molecules will be considered (no circular). Please check if your bam files obey this annotation. If genome is a data.frame, it represents the chromosome specification. The first column will be used as chromosome ID and the second column will be used as the chromosome lengths. If genome is a GenomicRanges, it should contain the equally sized genomic loci to count in, e.g. promoters. The binsize in the supplied NormRCountConfig is ignore in this case.

bamCountConfig is an instance of class NormRCountConfig specifying settings for read counting on bam files. You can specify the binsize, minimum mapping quality, shifting of read ends etc.. Please refer to NormRFit-class for details.

Value

A NormRFit container holding results of the fit with type enrichR.

Author(s)

Johannes Helmuth <helmuth@molgen.mpg.de>

See Also

NormRFit-class for functions on accessing and exporting the enrichR fit. NormRCountConfig-class for configuration of the read counting procedure (binsize, mapping quality,...).

Examples

```
require(GenomicRanges)

### enrichR(): Calling Enrichment over Input
#load some example bamfiles
input <- system.file("extdata", "K562_Input.bam", package="normr")
chipK4 <- system.file("extdata", "K562_H3K4me3.bam", package="normr")
```

```
#region to count in (example files contain information only in this region)
gr <- GRanges("chr1", IRanges(seq(22500001, 25000000, 1000), width = 1000))
#configure your counting strategy (see BamCountConfig-class)
countConfiguration <- countConfigSingleEnd(binsize = 1000,
                                           mapq = 30, shift = 100)
#invoke enrichR to call enrichment
enrich <- enrichR(treatment = chipK4, control = input,
                  genome = gr,  countConfig = countConfiguration,
                  iterations = 10, procs = 1, verbose = TRUE)
#inspect the fit
enrich
summary(enrich)

## Not run:
#write significant regions to bed
#exportR(enrich, filename = "enrich.bed", fdr = 0.01)
#write normalized enrichment to bigWig
#exportR(enrich, filename = "enrich.bw")
## End(**Not run**)

### diffR(): Calling differences between two conditions
chipK36 <- system.file("extdata", "K562_H3K36me3.bam", package="normr")
diff <- diffR(treatment = chipK36, control = chipK4,
              genome = gr,  countConfig = countConfiguration,
              iterations = 10, procs = 1, verbose = TRUE)
summary(diff)

### regimeR(): Identification of broad and peak enrichment
regime <- regimeR(treatment = chipK36, control = input, models = 3,
                  genome = gr,  countConfig = countConfiguration,
                  iterations = 10, procs = 1, verbose = TRUE)
summary(regime)
```

---

normR *Enrichment, Difference and Regime Calling in ChIP-seq data.*

Description

A correct background estimation is crucial for calling enrichment and differences in ChIP-seq data. normR provides robust normalization and difference calling in ChIP-seq and alike data. In brief, a binomial mixture model with a given number of components is fit to read count data for a treatment and control experiment. Therein, computational performance is improved by fitting a log-space model via Expectation Maximization in C++. Convergence is achieved by a threshold on the minimum change in model loglikelihood. After the model fit has converged, a robust background estimate is obtained. This estimate accounts for the effect of enrichment in certain regions and, therefore, represents an appropriate null hypothesis. This robust background is used to identify significantly enriched or depleted regions with respect to control. Moreover, a standardized enrichment for each bin is calculated based on the fitted background component. For convenience, read count vectors can be obtained directly from bam files when a compliant chromosome annotation is given. Please refer to the individual documentations of functions for enrichment calling (enrichR), difference calling (diffR) and enrichment regime calling (regimeR).

Details

Available functions are

enrichR: Enrichment calling between `treatment` (*e.g.* ChIP-seq) and `control` (*e.g.* Input).

diffR: Difference calling between `treatment` (*e.g.* ChIP-seq condition 1) and `control` (*e.g.* ChIP-seq condition 2).

regimeR: Enrichment regime calling between `treatment` (*e.g.* ChIP-seq) and `control` (*e.g.* Input) with a given number of model components. For example, 3 regimes recover background, broad and peak enrichment.

The computational performance is improved by fitting a log-space model in C++. Parallization is achieved in C++ via OpenMP (http://openmp.org).

Author(s)

Johannes Helmuth <helmuth@molgen.mpg.de>

See Also

NormRFit-class for functions on accessing and exporting the normR fit. NormRCountConfig-class for configuration of the read counting procedure (binsize, mapping quality,...).

Examples

```
require(GenomicRanges)

### enrichR(): Calling Enrichment over Input
#load some example bamfiles
input <- system.file("extdata", "K562_Input.bam", package="normr")
chipK4 <- system.file("extdata", "K562_H3K4me3.bam", package="normr")
#region to count in (example files contain information only in this region)
gr <- GRanges("chr1", IRanges(seq(22500001, 25000000, 1000), width = 1000))
#configure your counting strategy (see BamCountConfig-class)
countConfiguration <- countConfigSingleEnd(binsize = 1000,
                                           mapq = 30, shift = 100)
#invoke enrichR to call enrichment
enrich <- enrichR(treatment = chipK4, control = input,
                  genome = gr,  countConfig = countConfiguration,
                  iterations = 10, procs = 1, verbose = TRUE)
#inspect the fit
enrich
summary(enrich)

## Not run:
#write significant regions to bed
#exportR(enrich, filename = "enrich.bed", fdr = 0.01)
#write normalized enrichment to bigWig
#exportR(enrich, filename = "enrich.bw")
## End(**Not run**)

### diffR(): Calling differences between two conditions
chipK36 <- system.file("extdata", "K562_H3K36me3.bam", package="normr")
diff <- diffR(treatment = chipK36, control = chipK4,
              genome = gr,  countConfig = countConfiguration,
              iterations = 10, procs = 1, verbose = TRUE)
summary(diff)
```

```
### regimeR(): Identification of broad and peak enrichment
regime <- regimeR(treatment = chipK36, control = input, models = 3,
                  genome = gr,  countConfig = countConfiguration,
                  iterations = 10, procs = 1, verbose = TRUE)
summary(regime)
```

NormRCountConfig-class *Container for configuration of read counting with bamsignals in normR*

Description

This S4 class is a small wrapper for a configuration on obtaining counts from bamfiles with `bamsignals::bamProfile()`. Herein, two functions provide help for creating an instance of this class: `countConfigSingleEnd` creates a configuration for single end reads; and `countConfigPairedEnd` creates a configuration for paired end reads.

Usage

```
## S4 method for signature 'ANY'
countConfigSingleEnd(binsize = 250L, mapq = 20L,
  filteredFlag = 1024L, shift = 0L)

## S4 method for signature 'ANY'
countConfigPairedEnd(binsize = 250L, mapq = 20L,
  filteredFlag = 1024L, shift = 0L, midpoint = TRUE, tlenFilter = c(70L,
  200L))

## S4 method for signature 'NormRCountConfig'
getFilter(x)

## S4 method for signature 'NormRCountConfig'
print(x, ...)

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

Arguments

`binsize`	An `integer()` specifying the binsize in bp.
`mapq`	An `integer()` specifying the minimal mapping quality for a read to be counted.
`filteredFlag`	An `integer()` to filter for in the SAMFLAG field. For example, 1024 filters out marked duplicates (default). Refer to `https://broadinstitute.github.io/picard/explain-flags.html` for details.
`shift`	An `integer()` specifing a shift of the read counting position in 3'-direction. This can be handy in the analysis of chip-seq data.
`midpoint`	Paired End data only: A `logical()` indicating whether fragment midpoints instead of 5'-ends should be counted.

tlenFilter	An integer() of length two specifying the lower and upper length bound for a fragment to be considered. The fragment length as estimated from alignment in paired end experiments and written into the TLEN column.
x	A NormRCountConfig object.
...	optional arguments to be passed directly to the inherited function without alteration and with the original names preserved.
object	A NormRCountConfig object.

Value

A NormRCountConfig with specified counting parameters for normr methods (enrichR, diffR, regimeR

Methods (by generic)

- countConfigSingleEnd: Setup single end count configuration
- countConfigPairedEnd: Setup paired end count configuration
- getFilter: Get the filter compliant to bamsignals::bamProfile()
- print: Prints a given BamCounConfig
- show: Shows a given BamCounConfig

Slots

type A character of value paired.end or single.end.

binsize An integer specifying the binsize in bp.

mapq An integer specifying the minimal mapping quality for a read to be counted.

filteredFlag An integer to filter for in the SAMFLAG field. For example, 1024 filters out marked duplicates (default). Refer to https://broadinstitute.github.io/picard/explain-flags.html for details.

shift An integer specifing a shift of the read counting position in 3'-direction. This can be handy in the analysis of chip-seq data.

midpoint Paired End data only: A logical indicating whether fragment midpoints instead of 5'-ends should be counted.

tlenFilter Paired End data only: An integer of length two specifying the lower and upper length bound for a fragment to be considered. The fragment length as estimated from alignment in paired end experiments and written into the TLEN column.

Author(s)

Johannes Helmuth <helmuth@molgen.mpg.de>

See Also

normr for functions that use this object.

Examples

```
### Create an instance of this class (see below for helper functions)
# 250bp bins; single end reads; MAPQ>=10; no duplicates
countConfig <- new("NormRCountConfig",
                   type = "single.end", binsize = 250L, mapq = 10L,
                   filteredFlag = 1024L, shift = 0L, midpoint = FALSE,
                   tlenFilter = NULL)

### Counting configuration for Single End alignment files
# 250bp bins (default); only reads with MAPQ>=20; move counting position 100bp
countConfigurationSE <- countConfigSingleEnd(mapq = 20, shift = 100)
countConfigurationSE

### Counting configuration for Paired End alignment files
# 250bp bins; count center of fragments; only fragments with 70bp<=length<=200
countConfigurationPE <- countConfigPairedEnd(midpoint = TRUE,
                                             tlenFilter = c(70, 200))
countConfigurationPE
```

NormRFit-class *Container for a fit done with normR*

Description

This S4 class wraps a normR fit containing counts, fit configuration and results of the fit. Herein, functions for printing, summarization and accessing are provided. The functions enrichR, diffR and regimeR generate a container of this class to save results of a normR binomial mixture fitting. Please refer to their documentation for conventional usage of the normR package.

Usage

```
## S4 method for signature 'NormRFit,character'
exportR(x, filename, fdr = 0.01, color = NA,
  type = c(NA, "bed", "bedGraph", "bigWig"))

## S4 method for signature 'NormRFit,missing'
plot(x, y, ...)

## S4 method for signature 'NormRFit'
getCounts(x)

## S4 method for signature 'NormRFit'
getRanges(x, fdr = NA, k = NULL)

## S4 method for signature 'NormRFit'
getPosteriors(x)

## S4 method for signature 'NormRFit'
getEnrichment(x, B = NA, F = NA, standardized = TRUE,
  procs = 1L)

## S4 method for signature 'NormRFit'
```

```
getPvalues(x, filtered = FALSE)

## S4 method for signature 'NormRFit'
getQvalues(x)

## S4 method for signature 'NormRFit'
getClasses(x, fdr = NA)

## S4 method for signature 'NormRFit'
length(x)

## S4 method for signature 'NormRFit'
print(x, digits = max(3L, getOption("digits") - 3L), ...)

## S4 method for signature 'NormRFit'
show(object)

## S4 method for signature 'NormRFit'
summary(object, print = TRUE, digits = 3, ...)
```

Arguments

x	A NormRFit object.
filename	A character specifying the file to write to.
fdr	NA or a numeric between 0 and 1 specifying a FDR-level. Only regions with a q-value smaller than fdr will be returned. If set to NA, all regions analyzed will be returned (getRanges), classes are assigned by Maximum A Posteriori (exportR).
color	Specified color(s) when printing a bed file. If x@type == "enrichR", color should of length 1 and color shading will be done on this color. If x@type == "diffR", color should be of length 2 giving, firstly, the color for control and, secondly, the color for treatment. If x@type == "regimeR", color should be of length x@k-1, specifying a color for each enrichment component. Per default an appropriate color palette is used.
type	A character specifying the filetype for exporting results. If NA, format is guessed from filename's extension.
y	not used.
...	optional arguments to be passed directly to the inherited function without alteration and with the original names preserved.
k	NULL or a integer specifying a model component for which regions have to be returned. If set to NULL, regions are not filtered on component assignments. If fdr is set and k == x@B, the function stops.
B	An integer specifying the index of a mixture component. The enrichment is calculated relative to this component used as a background component. If <NA> (default), the background is determined by normR.
F	An integer specifying the index of a mixture component. The enrichment is calculated for this component over background B. If <NA> (default), the component with theta closest to B is used (enrichR, regimeR). For diffR, F is not effective.

standardized	A logical indicating if the enrichment should be standardized betwen 0 and 1. A non-standardized enrichment is particular useful when comparing intensities for ChIP-seq against the same antigen in different conditions (default = TRUE).
procs	An integer specifying the number of threads to use.
filtered	A logical specifying if T-filtered or all (default) P-values should be returned.
digits	Number of digits to show in number formatting.
object	A NormRCountConfig object.
print	logical() indicating if summary should be print to screen

Details

When working with instances of this S4 class, it is recommended to only use functions to access contents of this object. Internally, the class holds a map structure of unique elements to reduce memory requirements. #'

Value

getCounts: A list of length 2 with integer for control and treatment each.

getRanges: A GenomicRanges object.

getPosteriors: A matrix of posteriors for x@k mixture components

getEnrichment: A numeric of length length(x@n) giving the normR computed enrichment.

getPvalues: A numeric of length length(x@n) giving the normR computed Pvalues.

getQvalues: A numeric of length length(x@filteredT) giving the FDR-corrected q-values using Storey's method.

getClasses: A integer specifying assignments of regions to the mixture model. If x@type == "enrichR", it contains 1 for enriched regions and NA for non-enriched regions. If x@type == "diffR", it contains 1 for control-enriched regions, 2 for treatment-enriched regions and NA for non-enriched regions. If x@type == "regimeR", it contains >= 1 for regime-enriched regions and NA for non-enriched regions.

Methods (by generic)

- exportR: Export results of a normR fit to common file formats.
- plot: Plot a NormRFit.
- getCounts: Get count data for control and treatment.
- getRanges: Get the genomic coordinates of regions analyzed with information about component assignment.
- getPosteriors: Get computed posteriors for each mixture component.
- getEnrichment: Get normalized enrichment.
- getPvalues: Get normR-computed P-values.
- getQvalues: Get FDR-corrected q-values.
- getClasses: Get component assignments for each region analyzed.
- length: Returns the number of regions analyzed.
- print: Prints a small summary on a NormRFit.
- show: Shows a small summary on a NormRFit.
- summary: Prints a concise summary of a NormRFit.

Slots

`type` A `character` representing the type of fit. One of `c("enrichR","diffR", "regimeR")`.

`n` An `integer` specifying the number of regions.

`ranges` A `GenomicRanges` specifying the genomic coordinates of the regions.

`k` An `integer` giving the number of binomial mixture components.

`B` An `integer` specifying the index of the background component.

`map` A vector of `integer` holding a map to map back `counts`, `lnposteriors`, `lnenrichment`, `lnpvals`, `lnqvals` and `classes`. See low level function `normr:::map2uniquePairs` for how the map is generated.

`counts` A `list` of length two containing a vector of `integer` holding unique counts for control and treatment each. Use `getCounts` to retrieve original count matrix.

`amount` A vector of `integer` specifying the number of occurences of each unique control / treatment count pair.

`names` A `character` of length two specifying the names for control and treatment.

`thetastar` A `numeric` giving the calculated naive background estimation, *i.e.* `sum(getCounts(obj)[2,])/sum(getCo`

`theta` A `numeric` of length `k` giving the normR fitted parametrization of `k` binomial mixture components.

`mixtures` A `numeric` of length `k` giving the normR fitted mixture proportions of `k` binomial mixture components. Should add up to one.

`lnL` A vector of `numeric` holding the log-likelihood-trace of a normR model fit.

`eps` A `numeric` used as threshold for normR fit EM convergence.

`lnposteriors` A `matrix` with `length(amount)` rows and `k` columns. It contains the ln posterior probabilities for each unique control / treatment count pair. Use `getPosteriors` to get the posterior matrix for the original data.

`lnenrichment` A `numeric` of length `length(amount)` holding calculared normalized enrichment for each unique control / treatment count pair. The enrichment is calculated with respect to the fitted component `B`. Use `getEnrichment` to retrieve enrichment for the original data.

`lnpvals` A `numeric` of length `length(amount)` holding ln P-values for each unique control / treatment count pair. Given `theta` of `B` the signifcane of enrichment is assigned. Use `getPvalues` to retrieve Pvalues for original data.

`thresholdT` An `integer` giving the threshold used to filter P-values for FDR correction. The T-Filter threshold is a calculated population size for which the null hypothesis (`theta` of `B`) can be rejected. `eps` specifies the significance level.

`filteredT` A vector of `integer` giving indices of P-values passing `thresholdT`. Only these P-values will be considered for FDR correction.

`lnqvals` A `numeric` of length `length(filteredT)` holding ln q-values (FDR correction). P-values are corrected for multiple testing using Storey's method.

`classes` A `integer` of length `length(amount)` specifying the class assignments for each unique control / treatment count pair. These class assignments are based on the normR model fit. For `type == "enrichR"`, this vector contains either `NA` (not enriched) or `1` (enriched). For `type == "diffR"`, this vector contains `NA` (unchanged), `1` (differential in ChIP-seq 1) and `2` (differential in ChIP-seq 2). For `type == "regimeR"`, this vector contains `NA` (not enriched) and an arbitary number of enrichment class `>= 1`.

Author(s)

Johannes Helmuth <helmuth@molgen.mpg.de>

See Also

normr for function creating this container

Examples

```
require(GenomicRanges)

#Create a toy instance of type 'enrichR'
fit <- new("NormRFit",
           type="enrichR", n=10L,
           ranges=GRanges("chr1", IRanges(seq(1,100,10), width=10)),
           k=2L, B=1L, map=rep(1:5,2), counts=list(1:5, 1:5),
           amount=rep(2L,5), names=c("chip", "input"), thetastar=.35,
           theta=c(.15,.55), mixtures=c(.9,.1), lnL=seq(-50,-1,10), eps=.001,
           lnposteriors=log(matrix(runif(10), ncol=2)),
           lnenrichment=log(runif(5,0,.2)), lnpvals=log(runif(5)),
           filteredT=2:5, thresholdT=1L, lnqvals=log(runif(5,0,.2)),
           classes=sample(1:2,5,TRUE))

#print some statistics on fits
fit
summary(fit)

## Not run:
#write significant regions to bed
#exportR(fit, filename = "enrich.bed", fdr = 0.1)
#write normalized enrichment to bigWig
#exportR(fit, filename = "enrich.bw")
## End(**Not run**)

###AccessorMethods
#get original counts
getCounts(fit)
#get genomic coordinates for significant ranges as a GenomicRanges instance
getRanges(fit, fdr = .1)
getPosteriors(fit)
getEnrichment(fit)
getPvalues(fit)
getQvalues(fit)
getClasses(fit)
```

`regimeR` *Regime Enrichment Calling for ChIP-seq data in normR with regimeR*

Description

Regime enrichment calling between `treatment` (ChIP-seq) and `control` (Input) in normR is done by fitting background and multiple enrichment regimes simultaenously. Therefore, a mixture of `models` binomials is fit to the data with Expectation Maximization (EM). After convergence of the EM, the fitted background component is used to calculate significance for treatment and control count pair. Based on this statistic, user can extract significantly enriched regions with a desired significance level. Regime assignments are done by Maximum A Posteriori. Regions can be further analyzed within R or exported (see `NormRFit-class`). Furthermore, regimeR calculates a standardized enrichment given the fitted background component. For example, 3 regimes discriminate background, broad and peak enrichment. See also Details.

Usage

```
regimeR(treatment, control, genome, models, ...)

## S4 method for signature 'integer,integer,GenomicRanges,numeric'
regimeR(treatment, control,
  genome, models = 3, procs = 1L, verbose = TRUE, eps = 1e-05,
  iterations = 10, minP = 0.05)

## S4 method for signature 'character,character,GenomicRanges,numeric'
regimeR(treatment,
  control, genome, models = 3, countConfig = countConfigSingleEnd(),
  procs = 1L, verbose = TRUE, eps = 1e-05, iterations = 10,
  minP = 0.05)

## S4 method for signature 'character,character,data.frame,numeric'
regimeR(treatment, control,
  genome, models = 3, countConfig = countConfigSingleEnd(), procs = 1L,
  verbose = TRUE, eps = 1e-05, iterations = 10, minP = 0.05)

## S4 method for signature 'character,character,character,numeric'
regimeR(treatment, control,
  genome = "", models = 3, countConfig = countConfigSingleEnd(),
  procs = 1L, verbose = TRUE, eps = 1e-05, iterations = 10,
  minP = 0.05)
```

Arguments

treatment	An integer vector of treatment counts or a character pointing to the treatment bam file. In the latter case an "treatment.bai" index file should exist in the same folder.
control	An integer vector of control counts or a character pointing to the control bam file. In the latter case an "control.bai" index file should exist in the same folder.
genome	Either NULL (default), a character specifying a USCS genome identifier, a data.frame consisting of two columns or a GenomicRanges specifying the genomic regions (see Details).
models	An integer specifying the number of mixture components to fit [regimeR only]. Default is 3.
...	Optional arguments for the respective implementations of regimeR.
procs	An integer giving the number of parallel threads to use.
verbose	A logical indicating whether verbose output is desired.
eps	A numeric specifying the T Filter threshold and the threshold for EM convergence, *i.e.* the minimal difference in log-likelihood in two consecutive steps.
iterations	An integer specifying how many times the EM is initialized with random model parameters.
minP	An integer controlling the threshold for the T method when filtering low power regions, i.e. regions with low counts.
countConfig	A NormRCountConfig object specifying bam counting parameters for read count retrieval. See Details.

Details

Supplied count vectors for treatment and control should be of same length and of type `integer`.

For convenience, read count vectors can be obtained directly from bam files. In this case, please specify a bam file for treatment and control each and a `genome`. Bam files should be indexed using samtools (*i.e.* samtools index file file.bai). Furthermore, bam files should contain a valid header with given chromosome names. If `genome == NULL`(default), chromosome names will be read from treatment bamheader. Please be aware that bamheader might contain irregular contigs and chrM which influence the fit. Also be sure that treatment and control contain the same chromosomes. Otherwise an error will be thrown. If `genome` is a `character`, `fetchExtendedChromInfoFromUCSC` is used to resolve this to a valid UCSC genome identifier (see `https://genome.ucsc.edu/cgi-bin/hgGateway` for available genomes). In this case, only assembled molecules will be considered (no circular). Please check if your bam files obey this annotation. If `genome` is a `data.frame`, it represents the chromosome specification. The first column will be used as chromosome ID and the second column will be used as the chromosome lengths. If `genome` is a `GenomicRanges`, it should contain the equally sized genomic loci to count in, e.g. promoters. The binsize in the supplied NormRCountConfig is ignore in this case.

`bamCountConfig` is an instance of class `NormRCountConfig` specifying settings for read counting on bam files. You can specify the binsize, minimum mapping quality, shifting of read ends etc.. Please refer to `NormRFit-class` for details.

Value

A `NormRFit` container holding results of the fit with type `regimeR`.

Author(s)

Johannes Helmuth <helmuth@molgen.mpg.de>

See Also

`NormRFit-class` for functions on accessing and exporting the regimeR fit. `NormRCountConfig-class` for configuration of the read counting procedure (binsize, mapping quality,...).

Examples

```
require(GenomicRanges)

### enrichR(): Calling Enrichment over Input
#load some example bamfiles
input <- system.file("extdata", "K562_Input.bam", package="normr")
chipK4 <- system.file("extdata", "K562_H3K4me3.bam", package="normr")
#region to count in (example files contain information only in this region)
gr <- GRanges("chr1", IRanges(seq(22500001, 25000000, 1000), width = 1000))
#configure your counting strategy (see BamCountConfig-class)
countConfiguration <- countConfigSingleEnd(binsize = 1000,
                                           mapq = 30, shift = 100)
#invoke enrichR to call enrichment
enrich <- enrichR(treatment = chipK4, control = input,
                  genome = gr,  countConfig = countConfiguration,
                  iterations = 10, procs = 1, verbose = TRUE)
#inspect the fit
enrich
summary(enrich)
```

```
## Not run:
#write significant regions to bed
#exportR(enrich, filename = "enrich.bed", fdr = 0.01)
#write normalized enrichment to bigWig
#exportR(enrich, filename = "enrich.bw")
## End(**Not run**)

### diffR(): Calling differences between two conditions
chipK36 <- system.file("extdata", "K562_H3K36me3.bam", package="normr")
diff <- diffR(treatment = chipK36, control = chipK4,
              genome = gr,  countConfig = countConfiguration,
              iterations = 10, procs = 1, verbose = TRUE)
summary(diff)

### regimeR(): Identification of broad and peak enrichment
regime <- regimeR(treatment = chipK36, control = input, models = 3,
                  genome = gr,  countConfig = countConfiguration,
                  iterations = 10, procs = 1, verbose = TRUE)
summary(regime)
```

Index