# Package: consensusRegions (via r-universe)

November 25, 2026

**Type** Package

**Title** Consensus Regions from Replicated ChIP-seq and ATAC-seq Experiments

**Version** 1.0.0

**Description** Merges the evidence carried by replicated ChIP-seq, ATAC-seq or CUT&RUN experiments into a single set of consensus regions. When several replicates agree on a position their p-values are combined, so that a peak too weak to survive on its own can still be retained. Fisher, Stouffer, Lancaster and rank-product combinations are available; the weighted ones accept per-replicate weights taken from FRiP scores, library sizes, or the peak sets themselves when no alignment file is at hand. Input can be anything from a full narrowPeak to a bare BED3: scores are rank-transformed when p-values are missing, and a weighted k-of-n presence rule takes over when even the score is absent. The threshold on the combined significance can be calibrated against a permutation null rather than fixed a priori. Quality control plots and export to SummarizedExperiment are included. The approach follows MSPC (Jalili et al., 2015). It differs from consensus sets built on occupancy, such as the minOverlap rule of DiffBind, in that a region is kept on the combined significance of the peaks that overlap it rather than on how many replicates called it, and from IDR, which ranks a pair of replicates by reproducibility, in that any number of replicates is reduced to one set of regions.

**License** GPL-3

**Encoding** UTF-8

**Depends** R (>= 4.5.0), GenomicRanges (>= 1.56.0)

**Imports** BiocGenerics, BiocParallel, dplyr (>= 1.1.0), GenomeInfoDb, ggplot2, grDevices, IRanges, methods, rlang, Rsamtools, rtracklayer, S4Vectors, stats, SummarizedExperiment, tibble, tidyr, tools

**Suggests** BiocStyle, ComplexUpset, knitr, rmarkdown, testthat (>= 3.0.0)

**biocViews** ChIPSeq, ATACSeq, PeakDetection, Epigenetics, Sequencing, QualityControl, MultipleComparison, Software

**URL** https://github.com/sebastian-gregoricchio/consensusRegions

**BugReports** https://github.com/sebastian-gregoricchio/consensusRegions/issues

**VignetteBuilder** knitr

**Config/testthat/edition** 3

**Collate** 'consensusRegions-package.R' 'AllClasses.R' 'AllGenerics.R' 'utils.R' 'combineEvidence.R' 'readPeakSets.R' 'computeReplicateWeights.R' 'buildConsensus.R' 'calibrateThreshold.R' 'runConsensus.R' 'recentrePeaks.R' 'exportConsensus.R' 'asSummarizedExperiment.R' 'methods-ConsensusRegions.R' 'plots.R'

**Config/roxygen2/version** 8.1.0

**Config/pak/sysreqs**
make libbz2-dev libicu-dev liblzma-dev libxml2-dev libssl-dev xz-utils zlib1g-dev

**Repository** Bioconductor 3.24

**Date/Publication** 2026-11-23 23:28:10 UTC

**RemoteUrl** https://github.com/bioc/consensusRegions

**RemoteRef** RELEASE_3_24

**RemoteSha** 5fd17144f4fa8cf7828d7f980d331991624fc547

# Contents

consensusRegions-package . . . 3
asSummarizedExperiment . . . 4
buildConsensus . . . 5
calibrateThreshold . . . 8
combineEvidence . . . 11
computeReplicateWeights . . . 13
ConsensusRegions-accessors . . . 15
ConsensusRegions-class . . . 16
exportConsensus . . . 17
length,ConsensusRegions-method . . . 18
plotCalibration . . . 19
plotJaccard . . . 20
plotRescue . . . 21
plotUpset . . . 22
readPeakSets . . . 23
recentrePeaks . . . 24
runConsensus . . . 26
show,ConsensusRegions-method . . . 28

show,ThresholdCalibration-method . . . . . 29
ThresholdCalibration-accessors . . . . . 29
ThresholdCalibration-class . . . . . 31

**Index** **33**

---

`consensusRegions-package`

*consensusRegions: consensus peaks across replicated experiments*

---

## Description

Replicated ChIP-seq and ATAC-seq experiments rarely agree peak for peak. A region can be convincing in two replicates and just below threshold in the third, and discarding it wastes real signal. ‘consensusRegions‘ looks at every peak together with the peaks that overlap it in the other replicates and combines their p-values, so that repeated weak evidence can compensate for the lack of a single strong call.

The approach follows the one introduced by MSPC (Jalili and colleagues, 2015) and adds three things: replicate weights, so a shallow or noisy sample counts for less; a permutation null, so the threshold on the combined significance does not have to be guessed; and tolerance for incomplete input, since not everyone keeps the BAM files or the p-value column.

## Entry points

**[readPeakSets()** ] reads narrowPeak, broadPeak, BED6 or BED3 files and decides which statistics are usable.

**[computeReplicateWeights()** ] derives weights from BAM files, from pre-computed FRiP values, or from the peak sets alone.

**[buildConsensus()** ] runs the analysis and returns a [ConsensusRegions-class] object.

**[calibrateThreshold()** ] estimates the combined threshold from shuffled peak sets.

**[recentrePeaks()** ] recovers summits and returns fixed-width regions.

## Provenance

A substantial part of the code in this package was written with AI assistance (Assisted-by: Claude, Anthropic). The methods, design and validation are the author’s, who maintains the package and is responsible for its correctness.

## Author(s)

Sebastian Gregoricchio

## References

Jalili V., Matteucci M., Masseroli M., Morelli M.J. (2015). Using combined evidence from replicates to evaluate ChIP-seq peaks. *Bioinformatics* 31(17):2761-2769.

Corces M.R. *et al.* (2018). The chromatin accessibility landscape of primary human cancers. *Science* 362(6413):eaav1898.

## See Also

‘vignette("consensusRegions.vignette", package = "consensusRegions")‘ walks through a full analysis, from reading the peak files to writing the consensus out.

---

`asSummarizedExperiment` *Hand the consensus over to a differential analysis*

---

## Description

Builds a ‘RangedSummarizedExperiment‘ whose rows are the consensus regions and whose columns are the replicates. The assay holds the per-replicate significance at each region, zero where the replicate had no peak, together with a presence matrix.

The point of this is the row ranges. Counting reads over a fixed, shared set of regions is what csaw, DiffBind and edgeR all want, and this puts the consensus in the shape they expect without asking anyone to write out a BED file and read it back.

## Usage

```
asSummarizedExperiment(object, assayName = "negLog10P")
```

## Arguments

| | |
|---|---|
| `object` | A [ConsensusRegions-class] object. |
| `assayName` | Name given to the significance assay. Default: `"negLog10P"`. |

## Details

The assay is not a count matrix and must not be used as one. It records how convincing each replicate found each region, which is useful for clustering and for spotting a replicate that disagrees with the rest, but a differential test needs actual reads counted over ‘rowRanges(se)‘.

## Value

A ‘RangedSummarizedExperiment‘.

## Author(s)

Sebastian Gregoricchio

## Examples

```
peakFiles <- system.file("extdata",
    c(
        "rep1.narrowPeak", "rep2.narrowPeak",
        "rep3.narrowPeak"
    ),
    package = "consensusRegions"
)
peaks <- readPeakSets(peakFiles,
    sampleNames = c("r1", "r2", "r3"),
    verbose = FALSE
)
result <- buildConsensus(peaks, verbose = FALSE)

se <- asSummarizedExperiment(result)
se
```

---

`buildConsensus` *Build consensus regions from replicated peak calls*

---

## Description

Runs the whole analysis. Each peak is classified as stringent, weak or background by two p-value thresholds; the peaks overlapping it in the other replicates are collected; their evidence is combined; and the result is tested against a threshold on the combined significance and then corrected within each replicate. Peaks that survive are merged into consensus regions.

Feed this permissive input. Calling peaks at ‘q < 0.05‘ and running the consensus on the survivors leaves nothing to rescue and turns the whole exercise into an intersection with extra steps. Something around ‘p < 1e-3‘ gives the combination room to work.

## Usage

```
buildConsensus(
  peakList,
  weights = NULL,
  combinationMethod = c("stouffer", "fisher", "lancaster", "rankProduct"),
  replicateType = c("biological", "technical"),
  stringencyThreshold = 1e-08,
  weakThreshold = 1e-04,
  combinedThreshold = NULL,
  alpha = 0.05,
  minSupport = NULL,
  minReplicates = NULL,
  minSupportWeight = NULL,
  adjustmentFamily = c("tested", "confirmed"),
  minOverlap = 1L,
```

```
  minOverlapFraction = NULL,
  multipleIntersections = c("lowest", "highest"),
  recursive = TRUE,
  maxIterations = 10L,
  mergeMethod = c("reduce", "iterative"),
  maxConsensusWidth = NULL,
  calibration = NULL,
  excludeRegions = NULL,
  verbose = TRUE
)
```

## Arguments

| | |
|---|---|
| peakList | A ‘GRangesList‘ from [readPeakSets()]. |
| weights | Named numeric vector of replicate weights. See [computeReplicateWeights()]. Default: NULL, meaning that every replicate counts the same. |
| combinationMethod | Passed to [combineEvidence()]. Default: "stouffer". |
| replicateType | ‘"biological"‘ asks for ‘minSupport‘ supporting replicates; ‘"technical"‘ asks for all of them. Default: "biological". |
| stringencyThreshold | P-value below which a peak is stringent. Default: 1e-08. |
| weakThreshold | P-value above which a peak is background and takes no further part. Default: 1e-04. |
| combinedThreshold | Threshold on the combined p-value. [calibrateThreshold()] can estimate it instead. The scale differs between combination schemes, so a value carried over from one will not mean the same under another. The rank product in particular is bounded by the number of peaks per replicate and usually needs a far more permissive threshold. Default: NULL, meaning that stringencyThreshold is used. |
| alpha | Level for the within-replicate Benjamini-Hochberg step. Default: 0.05. |
| minSupport | Number of *other* replicates that must hold an overlapping peak. This is a count of replicates and weights never substitute for it. Give this or ‘minReplicates‘, not both. Default: NULL, meaning that one supporting replicate is enough. |
| minReplicates | How many replicates in total must hold the peak, counting the one it came from. This is the way the requirement is usually spoken about, and it matches MSPC’s ‘-c‘. Accepts a count (‘5‘), a proportion (‘0.75‘) or a percentage (‘"75 rounded up. ‘minReplicates = 2‘ and ‘minSupport = 1‘ are the same requirement. Default: NULL. |
| minSupportWeight | Optional additional requirement on the summed weight of the supporting replicates. Applied on top of ‘minSupport‘, never instead of it. Only meaningful when weights are not equal. Default: NULL. |
| adjustmentFamily | Which peaks form the family for the Benjamini-Hochberg step. ‘"tested"‘ corrects across every peak that entered the analysis; ‘"confirmed"‘ corrects only |

across the peaks that cleared ‘combinedThreshold‘, which is what MSPC does. Default: `"tested"`.

`minOverlap` Minimum overlap in base pairs. Default: `1L`.

`minOverlapFraction`
Optional minimum overlap as a fraction of the shorter of the two peaks. Applied on top of ‘minOverlap‘. Default: `NULL`.

`multipleIntersections`
How to pick among several overlapping peaks from the same replicate: ‘"lowest"‘ takes the smallest p-value, ‘"highest"‘ the largest. Default: `"lowest"`.

`recursive` Re-run the confirmation after dropping discarded peaks from the pool of eligible supporters. Default: `TRUE`.

`maxIterations` Cap on the recursive rounds. Default: `10L`.

`mergeMethod` ‘"reduce"‘ merges anything that touches; ‘"iterative"‘ seeds on the most significant peak and removes what it overlaps, then repeats, which is the iterative removal procedure of Corces and colleagues. Default: `"reduce"`.

`maxConsensusWidth`
Optional cap in base pairs. Merged regions wider than this are rebuilt with the iterative rule. Default: `NULL`.

`calibration` Optional output of [calibrateThreshold()]. When supplied its threshold is used, overriding ‘combinedThreshold‘, and the whole calibration is kept with the result. Default: `NULL`.

`excludeRegions` Optional ‘GRanges‘ of blacklisted positions, removed from the consensus at the end. Default: `NULL`.

`verbose` Report progress. Default: `TRUE`.

## Details

Two behaviours are worth knowing about.

Transitive merging can run away. Peak A overlaps B, B overlaps C, and A never touches C, yet ‘"reduce"‘ puts all three in one region. On permissive input this occasionally produces regions tens of kilobases long. ‘maxConsensusWidth‘ catches them and rebuilds only the offenders.

Recursive confirmation makes a peak’s fate depend on peaks that were themselves discarded, which is arguably the more defensible reading of the method but is not what the original implementation does. Set ‘recursive = FALSE‘ to reproduce a single-pass analysis.

When the peak sets carry no usable statistic the combination is skipped and a weighted presence rule is applied instead: a peak is kept when the weights of the replicates supporting it sum to at least ‘minSupport‘. With equal weights this is the familiar k-of-n rule.

## Value

A [ConsensusRegions-class] object.

## Author(s)

Sebastian Gregoricchio

## References

Corces M.R. *et al.* (2018). The chromatin accessibility landscape of primary human cancers. *Science* 362(6413):eaav1898.

## Examples

```
peakFiles <- system.file("extdata",
    c(
        "rep1.narrowPeak", "rep2.narrowPeak",
        "rep3.narrowPeak"
    ),
    package = "consensusRegions"
)
peaks <- readPeakSets(peakFiles,
    sampleNames = c("r1", "r2", "r3"),
    verbose = FALSE
)

result <- buildConsensus(peaks, verbose = FALSE)
consensusRanges(result)

## down-weighting a replicate known to be poor
weighted <- buildConsensus(
    peaks,
    weights = computeReplicateWeights(peaks,
        method = "frip",
        frip = c(0.2, 0.18, 0.05)
    ),
    combinationMethod = "stouffer",
    verbose = FALSE
)
consensusStats(weighted)
```

---

# calibrateThreshold *Calibrate the combined significance threshold*

---

## Description

The threshold on the combined p-value is normally set by hand, and ‘1e-8‘ has been carried around for long enough that it now looks like a property of the method rather than a guess. It is a guess, and the right value depends on how many replicates there are, how they were called, and how densely the peaks sit.

This function estimates it instead. Peak positions are shuffled within each chromosome while their widths, counts and statistics are kept, so any overlap between replicates in the shuffled sets happens by chance. Running the combination on those gives a null distribution, and the threshold is the point where the expected number of null peaks reaching it falls to ‘targetFDR‘ of the observed number.

## Usage

```
calibrateThreshold(
  peakList,
  weights = NULL,
  combinationMethod = c("stouffer", "fisher", "lancaster", "rankProduct"),
  nPermutations = 50L,
  targetFDR = 0.05,
  stringencyThreshold = 1e-08,
  weakThreshold = 1e-04,
  minSupport = NULL,
  minReplicates = NULL,
  minOverlap = 1L,
  minOverlapFraction = NULL,
  recursive = TRUE,
  maxIterations = 10L,
  multipleIntersections = c("lowest", "highest"),
  chromosomeLengths = NULL,
  excludeRegions = NULL,
  BPPARAM = NULL,
  seed = NULL,
  verbose = TRUE
)
```

## Arguments

| | |
|---|---|
| peakList | A 'GRangesList' from [readPeakSets()]. |
| weights | Named numeric vector of replicate weights, or 'NULL'. Default: NULL. |
| combinationMethod | Passed to [combineEvidence()]. Default: "stouffer". |
| nPermutations | Number of shuffled replicate sets to generate. Default: 50L. |
| targetFDR | Empirical false discovery rate to aim for. Default: 0.05. |
| stringencyThreshold | Stringent p-value cut, as in [buildConsensus()]. Default: 1e-08. |
| weakThreshold | Background p-value cut. Default: 1e-04. |
| minSupport | Minimum supporting replicates. Give this or 'minReplicates', not both. Default: NULL. |
| minReplicates | Total replicates that must hold the peak, counting its own. Accepts a count, a proportion or a percentage string, as in [buildConsensus()]. Must match the value used there. Default: NULL. |
| minOverlap | Minimum overlap in base pairs. Default: 1L. |
| minOverlapFraction | Optional fractional overlap requirement. Must match the value used in [buildConsensus()]. Default: NULL. |
| recursive | Whether the confirmation is re-run after pruning unsupported peaks. Must match the value used in [buildConsensus()]. Default: TRUE. |

`maxIterations` Cap on the recursive rounds. Default: `10L`.

`multipleIntersections`
Resolution rule for several overlapping peaks from one replicate. Default: `"lowest"`.

`chromosomeLengths`
Named integer vector. Taken from the input, or inferred from the furthest peak, when left ‘NULL‘. Default: `NULL`.

`excludeRegions` Optional ‘GRanges‘ the shuffled peaks must avoid. Default: `NULL`.

`BPPARAM` A ‘BiocParallelParam‘ object, used as it is. A single number is accepted as a shorthand for that many cores, since asking for more cores and nothing else is the common case. The permutations are independent of one another and are where nearly all the time goes, so raising this is worth it on a large peak set: a single round takes around half a minute on 100,000 peaks per replicate. Default: `NULL`, a `BiocParallel::SerialParam()` seeded from `seed`, so that nothing runs in parallel unasked.

`seed` Number seeding the random positions of the shuffled peaks. It goes to the backend built from ‘BPPARAM‘ as its ‘RNGseed‘, so the same seed returns the same threshold on any number of cores. Ignored, with a warning, when ‘BPPARAM‘ is a ‘BiocParallelParam‘ object: give the seed to it as ‘RNGseed‘ instead. Default: `NULL`, no seed.

`verbose` Report progress. Default: `TRUE`.

## Details

Shuffling within a chromosome preserves peak density but not the relationship between peaks and genes, so the null is a little optimistic wherever peaks cluster around promoters. Passing ‘excludeRegions‘ with a blacklist, and restricting the analysis to mappable chromosomes beforehand, both help.

Fifty permutations are usually enough to place the threshold within a factor of two, which is as much precision as the choice deserves. Push it higher only if the curve looks ragged near ‘targetFDR‘.

The peak positions are drawn at random inside ‘BiocParallel‘, which keeps its random numbers apart from those of the session: [base::set.seed()] before the call does not reach them, on one core or on many. ‘seed‘ is what brings the same threshold back, whatever the number of cores. The function never calls ‘set.seed()‘, so the random numbers the rest of the session draws are left as they were. A ‘BiocParallelParam‘ object passed as ‘BPPARAM‘ is used as it is and carries its own seed, for instance ‘BiocParallel::MulticoreParam(workers = 8, RNGseed = 42)‘.

## Value

A [ThresholdCalibration-class] object, holding the recommended ‘threshold‘ on the p-value scale, the ‘fdrCurve‘ it was read off, the observed and null combined statistics, the ‘seed‘ the permutations were drawn with and its ‘seedSource‘: ‘"seed"‘ for the argument, ‘"BPPARAM"‘ for the ‘RNGseed‘ of a backend passed as ‘BPPARAM‘, ‘"unseeded"‘ when there was none, in which case ‘seed‘ is ‘NULL‘. Its components are reached with the accessors documented in [ThresholdCalibration-accessors], and ‘$‘ still reaches them by name, as it did when this was a plain list.

## Author(s)

Sebastian Gregoricchio

## Examples

```
peakFiles <- system.file("extdata",
    c(
        "rep1.narrowPeak", "rep2.narrowPeak",
        "rep3.narrowPeak"
    ),
    package = "consensusRegions"
)
peaks <- readPeakSets(peakFiles,
    sampleNames = c("r1", "r2", "r3"),
    verbose = FALSE
)

calibration <- calibrateThreshold(peaks,
    nPermutations = 5, seed = 42,
    verbose = FALSE
)
calibration$threshold

result <- buildConsensus(peaks,
    combinedThreshold = calibration$threshold,
    verbose = FALSE
)
```

---

`combineEvidence` *Combine the evidence carried by overlapping peaks*

---

## Description

Pools a set of p-values into a single one. Four schemes are offered and they behave differently enough that the choice matters.

Fisher's method is what MSPC uses. It treats every replicate as equally informative and is the right default when they really are comparable. Its weakness shows up as the number of replicates grows, because the combined statistic keeps accumulating and a region detected faintly in many samples starts to look highly significant.

Stouffer's Z accepts weights, so a shallow replicate can be told to count for less instead of being dropped or trusted blindly. This is the default here.

Lancaster's method also takes weights, folding them into the degrees of freedom of a chi-squared sum. It is closer to Fisher in spirit and tends to react more strongly to a single very small p-value.

The rank product ignores the p-values and works on within-replicate ranks, which makes it the natural option when the input carries only a score. It is also the least sensitive to a replicate whose p-values are poorly calibrated.

## Usage

```
combineEvidence(
  negLog10P,
  weights = NULL,
  method = c("stouffer", "fisher", "lancaster", "rankProduct"),
  rho = NULL
)
```

## Arguments

| | |
|---|---|
| negLog10P | Numeric vector of -log10 transformed p-values. |
| weights | Numeric vector of replicate weights, recycled to the length of ‘negLog10P‘. Ignored by ‘"fisher"‘ and ‘"rankProduct"‘. Defaults to equal weights. Default: NULL. |
| method | One of ‘"stouffer"‘, ‘"fisher"‘, ‘"lancaster"‘ or ‘"rankProduct"‘. Default: "stouffer". |
| rho | Numeric vector of relative within-replicate ranks in ‘(0, 1]‘, required by ‘"rankProduct"‘ and ignored otherwise. Default: NULL. |

## Details

Everything is computed in log space. Peak callers routinely report p-values far below the smallest representable double, and exponentiating them first would collapse the whole set to zero.

## Value

A single numeric value, the combined significance on the -log10 scale.

## Author(s)

Sebastian Gregoricchio

## References

Lancaster H.O. (1961). The combination of probabilities: an application of orthonormal functions. *Australian Journal of Statistics* 3:20-33.

Breitling R., Armengaud P., Amtmann A., Herzyk P. (2004). Rank products: a simple, yet powerful, new method to detect differentially regulated genes. *FEBS Letters* 573(1-3):83-92.

## Examples

```
## three replicates, none convincing alone
combineEvidence(c(3, 3.2, 2.8), method = "fisher")

## the same peaks, but the third replicate is known to be poor
combineEvidence(c(3, 3.2, 2.8),
    weights = c(1.3, 1.3, 0.4),
    method = "stouffer"
)
```

`computeReplicateWeights` *Derive per-replicate weights*

## Description

Fisher's method assumes every replicate is worth the same. In practice one of them is usually shallower, or has a worse signal-to-noise ratio, and the usual responses are to drop it or to let it drag the combined statistic down. Weighting is the middle option: the replicate still contributes, just less.

Weights can come from three places, in decreasing order of how much extra material they require. FRiP and library size need the BAM files or the numbers already computed from them. The intrinsic route needs nothing beyond the peaks themselves and scores each replicate by how well it agrees with the others, which is the only option left when all that survives of an old analysis is a folder of BED files.

## Usage

```
computeReplicateWeights(
  peakList,
  method = c("equal", "frip", "librarySize", "intrinsic", "custom"),
  bamFiles = NULL,
  frip = NULL,
  librarySize = NULL,
  weights = NULL,
  minMapq = 0L,
  verbose = TRUE
)
```

## Arguments

| | |
|---|---|
| `peakList` | A ‘GRangesList‘, typically from [readPeakSets()]. |
| `method` | One of ‘"equal"‘, ‘"frip"‘, ‘"librarySize"‘, ‘"intrinsic"‘ or ‘"custom"‘. Default: `"equal"`. |
| `bamFiles` | Character vector of indexed BAM files, in the same order as ‘peakList‘. Needed by ‘"frip"‘ and ‘"librarySize"‘ unless the values are supplied directly. Default: `NULL`. |
| `frip` | Numeric vector of pre-computed fractions of reads in peaks. Skips the BAM pass. Default: `NULL`. |
| `librarySize` | Numeric vector of pre-computed mapped read counts. Default: `NULL`. |
| `weights` | Numeric vector used as is when ‘method‘ is ‘"custom"‘. Default: `NULL`. |
| `minMapq` | Minimum mapping quality when counting from BAM. Default: `0L`. |
| `verbose` | Report progress. Default: `TRUE`. |

## Details

For FRiP the weight is proportional to the fraction itself. For library size it is proportional to the square root of the depth, which is how the information in a z-score scales, so doubling the depth is worth rather less than twice as much. Bear in mind that peak-caller p-values already respond to sequencing depth, so weighting by depth on top of them counts it twice to some degree; FRiP is the safer choice when both are available.

When ‘minMapq‘ is above zero both the reads in peaks and the library total are counted under that filter, which means the whole file is read rather than just the index.

The intrinsic weight is the mean Jaccard index between one replicate and each of the others. A replicate that shares little with the rest is either the odd one out biologically or the noisy one, and in both cases letting it drive a consensus is unwise. This measure is circular by construction, so prefer FRiP whenever the alignments are available.

## Value

A named numeric vector with mean one, carrying the raw metrics as the ‘metrics‘ attribute.

## Author(s)

Sebastian Gregoricchio

## Examples

```
peakFiles <- system.file("extdata",
    c(
        "rep1.narrowPeak", "rep2.narrowPeak",
        "rep3.narrowPeak"
    ),
    package = "consensusRegions"
)
peaks <- readPeakSets(peakFiles,
    sampleNames = c("r1", "r2", "r3"),
    verbose = FALSE
)

## no BAM files at hand
computeReplicateWeights(peaks, method = "intrinsic")

## FRiP measured elsewhere
computeReplicateWeights(peaks,
    method = "frip",
    frip = c(0.21, 0.19, 0.06)
)
```

ConsensusRegions-accessors
*Accessors for ConsensusRegions objects*

## Description

Retrieve the components of a [ConsensusRegions-class] object.

## Usage

```
consensusRanges(object)

peakSets(object)

replicateWeights(object)

analysisParameters(object)

consensusStats(object)

calibrationResult(object)

## S4 method for signature 'ConsensusRegions'
consensusRanges(object)

## S4 method for signature 'ConsensusRegions'
peakSets(object)

## S4 method for signature 'ConsensusRegions'
replicateWeights(object)

## S4 method for signature 'ConsensusRegions'
analysisParameters(object)

## S4 method for signature 'ConsensusRegions'
consensusStats(object)

## S4 method for signature 'ConsensusRegions'
calibrationResult(object)
```

## Arguments

| | |
|---|---|
| object | A [ConsensusRegions-class] object. |

## Value

‘consensusRanges()‘ returns a ‘GRanges‘ with the consensus regions. ‘peakSets()‘ returns the annotated per-replicate peaks as a ‘GRangesList‘. ‘replicateWeights()‘ returns the named numeric

vector of weights. ‘analysisParameters()‘ returns the list of settings used. ‘consensusStats()‘ returns a per-replicate summary ‘data.frame‘. ‘calibrationResult()‘ returns the [ThresholdCalibration-class] object the consensus was built with, or ‘NULL‘ when it was built without one.

## Author(s)

Sebastian Gregoricchio

## Examples

```
peakFiles <- system.file("extdata",
    c(
        "rep1.narrowPeak", "rep2.narrowPeak",
        "rep3.narrowPeak"
    ),
    package = "consensusRegions"
)
peaks <- readPeakSets(peakFiles, sampleNames = c("r1", "r2", "r3"))
result <- buildConsensus(peaks, verbose = FALSE)

consensusRanges(result)
consensusStats(result)
```

---

# ConsensusRegions-class

*ConsensusRegions class*

---

## Description

Container for the result of [buildConsensus()]. It keeps the peaks as they were submitted, the per-peak verdict of the combination step, the consensus regions themselves, and the settings used to get there, so that an analysis can be traced back from its output.

## Slots

`peaks` A ‘GRangesList‘, one element per replicate, holding the peaks after classification. Each element carries the metadata columns ‘negLog10P‘, ‘class‘, ‘nSupport‘, ‘supportWeight‘, ‘combinedNegLog10P‘, ‘combinedNegLog10Padj‘ and ‘status‘.

`consensus` A ‘GRanges‘ of consensus regions.

`weights` A named numeric vector of replicate weights.

`parameters` A list with the call arguments.

`stats` A ‘DataFrame‘ summarising each replicate.

`calibration` A [ThresholdCalibration-class] object when the consensus was built with one, ‘NULL‘ otherwise.

## Author(s)

Sebastian Gregoricchio

`exportConsensus` *Write the results to disk*

## Description

Writes the consensus regions, and optionally the annotated per-replicate peaks, as BED files. The combined significance is carried in the score column, capped at the 1000 that the BED specification allows, so the output loads into a genome browser without complaint.

## Usage

```
exportConsensus(
  object,
  file,
  format = c("bed", "narrowPeak"),
  perReplicate = FALSE,
  verbose = TRUE
)
```

## Arguments

| | |
|---|---|
| `object` | A [ConsensusRegions-class] object. |
| `file` | Path for the consensus file. |
| `format` | ‘"bed"‘ or ‘"narrowPeak"‘. Default: `"bed"`. |
| `perReplicate` | Also write one file per replicate holding every peak with its verdict. The names are derived from ‘file‘. Default: `FALSE`. |
| `verbose` | Report what was written. Default: `TRUE`. |

## Value

Invisibly, a character vector of the paths written.

## Author(s)

Sebastian Gregoricchio

## Examples

```
peakFiles <- system.file("extdata",
    c(
        "rep1.narrowPeak", "rep2.narrowPeak",
        "rep3.narrowPeak"
    ),
    package = "consensusRegions"
)
peaks <- readPeakSets(peakFiles,
    sampleNames = c("r1", "r2", "r3"),
```

```
    verbose = FALSE
)
result <- buildConsensus(peaks, verbose = FALSE)

target <- file.path(tempdir(), "consensus.bed")
exportConsensus(result, target, verbose = FALSE)
```

---

`length,ConsensusRegions-method` *Number of consensus regions*

---

## Description

Number of consensus regions

## Usage

```
## S4 method for signature 'ConsensusRegions'
length(x)
```

## Arguments

| | |
|---|---|
| x | A [ConsensusRegions-class] object. |

## Value

Integer, the number of consensus regions.

## Author(s)

Sebastian Gregoricchio

## Examples

```
peakFiles <- system.file("extdata",
    c(
        "rep1.narrowPeak", "rep2.narrowPeak",
        "rep3.narrowPeak"
    ),
    package = "consensusRegions"
)
peaks <- readPeakSets(peakFiles,
    sampleNames = c("r1", "r2", "r3"),
    verbose = FALSE
)
length(buildConsensus(peaks, verbose = FALSE))
```

`plotCalibration` *The calibration curve*

## Description

Plots the empirical false discovery rate against the cut on the combined statistic, with the chosen threshold marked. A curve that never descends to the target says the replicates do not agree beyond chance often enough to support a consensus at that stringency.

## Usage

```
plotCalibration(calibration, baseSize = 12)
```

## Arguments

| | |
|---|---|
| `calibration` | Output of [calibrateThreshold()]. |
| `baseSize` | Base font size in points. Default: 12. |

## Value

A ‘ggplot‘ object.

## Author(s)

Sebastian Gregoricchio

## Examples

```
peakFiles <- system.file("extdata",
    c(
        "rep1.narrowPeak", "rep2.narrowPeak",
        "rep3.narrowPeak"
    ),
    package = "consensusRegions"
)
peaks <- readPeakSets(peakFiles,
    sampleNames = c("r1", "r2", "r3"),
    verbose = FALSE
)
calibration <- calibrateThreshold(peaks,
    nPermutations = 5, seed = 42,
    verbose = FALSE
)

plotCalibration(calibration)
```

plotJaccard *Pairwise agreement between replicates*

## Description

Jaccard index over the peaks entering the analysis. A replicate that agrees poorly with all the others is a candidate for down-weighting; two blocks of mutually agreeing replicates usually mean a batch effect rather than a quality problem.

## Usage

```
plotJaccard(
  object,
  showValues = TRUE,
  lowColour = "#F5F5F5",
  highColour = "#2C6E91",
  baseSize = 12
)
```

## Arguments

| | |
|---|---|
| object | A [ConsensusRegions-class] object. |
| showValues | Print the index inside each tile. The lettering switches between black and white so that it stays legible whatever the tile is filled with. Default: TRUE. |
| lowColour | Colour for a Jaccard index of zero. Default: "#F5F5F5". |
| highColour | Colour for a Jaccard index of one. Default: "#2C6E91". |
| baseSize | Base font size in points. Default: 12. |

## Value

A ‘ggplot‘ object.

## Author(s)

Sebastian Gregoricchio

## Examples

```
peakFiles <- system.file("extdata",
    c(
        "rep1.narrowPeak", "rep2.narrowPeak",
        "rep3.narrowPeak"
    ),
    package = "consensusRegions"
)
peaks <- readPeakSets(peakFiles,
    sampleNames = c("r1", "r2", "r3"),
```

```
    verbose = FALSE
)
plotJaccard(buildConsensus(peaks, verbose = FALSE))
```

---

plotRescue *Rescue and discard rates per replicate*

---

## Description

Shows what happened to the peaks of each replicate. The bar to watch is the rescued one: peaks that were only weak on their own and were kept because the other replicates agreed. That is the whole point of the method, but a rescued fraction far above what the replicates otherwise share is a sign that ‘weakThreshold‘ was set too permissively and the combination is confirming noise.

## Usage

```
plotRescue(object, proportion = FALSE, baseSize = 12)
```

## Arguments

| | |
|---|---|
| object | A [ConsensusRegions-class] object. |
| proportion | Scale the bars to one instead of showing counts. Default: FALSE. |
| baseSize | Base font size in points. Default: 12. |

## Value

A ‘ggplot‘ object.

## Author(s)

Sebastian Gregoricchio

## Examples

```
peakFiles <- system.file("extdata",
    c(
        "rep1.narrowPeak", "rep2.narrowPeak",
        "rep3.narrowPeak"
    ),
    package = "consensusRegions"
)
peaks <- readPeakSets(peakFiles,
    sampleNames = c("r1", "r2", "r3"),
    verbose = FALSE
)
result <- buildConsensus(peaks, verbose = FALSE)

plotRescue(result)
```

`plotUpset` *Which replicates contribute to each consensus region*

## Description

An UpSet plot of the replicate combinations found across the consensus regions. Regions supported by every replicate should dominate; a large partial set points at a replicate that is pulling in a different direction.

## Usage

```
plotUpset(object, minSize = 0)
```

## Arguments

| | |
|---|---|
| `object` | A [ConsensusRegions-class] object. |
| `minSize` | Drop intersections smaller than this. Default: `0`. |

## Value

A 'ggplot' object built by 'ComplexUpset'.

## Author(s)

Sebastian Gregoricchio

## Examples

```
peakFiles <- system.file("extdata",
    c(
        "rep1.narrowPeak", "rep2.narrowPeak",
        "rep3.narrowPeak"
    ),
    package = "consensusRegions"
)
peaks <- readPeakSets(peakFiles,
    sampleNames = c("r1", "r2", "r3"),
    verbose = FALSE
)
result <- buildConsensus(peaks, verbose = FALSE)

if (requireNamespace("ComplexUpset", quietly = TRUE)) {
    plotUpset(result)
}
```

`readPeakSets` *Read peak files into a GRangesList*

## Description

Imports peak calls and works out which statistic can actually be used. A narrowPeak carries -log10 p-values in its eighth column and gets the full treatment. A BED with a score column is rank-transformed. A bare BED3 has nothing to combine, so the analysis later falls back to a presence rule. The detection is reported so the choice is never silent.

## Usage

```
readPeakSets(
  peaks,
  sampleNames = NULL,
  scoreType = c("auto", "log10pvalue", "pvalue", "score", "none"),
  scoreColumn = NULL,
  keepStandardChromosomes = TRUE,
  seqlevelsStyle = "UCSC",
  genome = NA,
  verbose = TRUE
)
```

## Arguments

| | |
|---|---|
| `peaks` | Character vector of file paths, or a ‘GRanges‘, ‘GRangesList‘, or list of ‘GRanges‘. |
| `sampleNames` | Character vector naming the replicates. Taken from the file names or the list names when left ‘NULL‘. Default: `NULL`. |
| `scoreType` | One of ‘"auto"‘, ‘"log10pvalue"‘, ‘"pvalue"‘, ‘"score"‘ or ‘"none"‘. ‘"auto"‘ inspects the input. Default: `"auto"`. |
| `scoreColumn` | Name of the metadata column holding the statistic. Only needed when the input is not a standard peak format. Default: `NULL`. |
| `keepStandardChromosomes` | Drop scaffolds and patches. Default: `TRUE`. |
| `seqlevelsStyle` | String indicating the chromosome naming style to apply to every replicate, one among ‘"UCSC"‘ (chr1), ‘"Ensembl"‘ (1) or ‘"NCBI"‘. When set to ‘NULL‘ the names are left as they are and the loading is interrupted if the replicates use different styles. Default: ‘"UCSC"‘. |
| `genome` | Genome identifier passed to the importer, for instance ‘"hg38"‘. Fills in the chromosome lengths used by [calibrateThreshold()]. |
| `verbose` | Report what was detected. Default: `TRUE`. |

## Details

A narrowPeak whose p-value column is filled with ‘-1‘, which is what several callers write when they have nothing to report, is treated as score-only rather than as a set of p-values equal to ‘10‘.

## Value

A ‘GRangesList‘, one element per replicate, with a ‘negLog10P‘ metadata column when a statistic was available. The resolved score type is stored in the object metadata.

## Author(s)

Sebastian Gregoricchio

## Examples

```
peakFiles <- system.file("extdata",
    c(
        "rep1.narrowPeak", "rep2.narrowPeak",
        "rep3.narrowPeak"
    ),
    package = "consensusRegions"
)
peaks <- readPeakSets(peakFiles, sampleNames = c("r1", "r2", "r3"))
peaks
```

---

# recentrePeaks *Recover summits and return fixed-width regions*

---

## Description

Consensus regions inherit their edges from whichever peaks happened to be merged, so their widths vary a great deal. That variation biases every count-based and enrichment-based analysis downstream, since a wider region collects more reads and more motif matches for reasons that have nothing to do with the biology.

Recentring fixes it. Each region is reduced to a single position and re-expanded to a common width, which is what motif analysis wants and what the published accessibility atlases do: Corces and colleagues build their pan-cancer ATAC-seq atlas out of fixed-width windows taken around summits for the same reason.

## Usage

```
recentrePeaks(
  object,
  width = 400L,
  summitSource = c("best", "weighted", "centre"),
  chromosomeLengths = NULL
)
```

## Arguments

| | |
|---|---|
| `object` | A [ConsensusRegions-class] object. |
| `width` | Width of the returned regions in base pairs. Default: `400L`. |
| `summitSource` | How to place the anchor. ‘"best"‘ takes the summit of the most significant member peak, ‘"weighted"‘ averages the member summits weighted by their significance, and ‘"centre"‘ uses the midpoint of the merged region. Default: `"best"`. |
| `chromosomeLengths` | Named integer vector used to trim regions that would run off a chromosome end. Taken from the object when ‘NULL‘. Default: `NULL`. |

## Details

‘"best"‘ needs a summit offset, which narrowPeak files carry in their tenth column and other formats do not. When it is missing the midpoint of the member peak is used instead, and a warning says so.

Widths between 200 and 500 bp suit transcription factors and ATAC peaks. Broad histone domains should not be recentred at all, since collapsing a domain to a point throws away the thing being measured.

## Value

A ‘GRanges‘ of fixed-width regions carrying the summit position and the metadata of the consensus it came from.

## Author(s)

Sebastian Gregoricchio

## References

Corces M.R. *et al.* (2018). The chromatin accessibility landscape of primary human cancers. *Science* 362(6413):eaav1898.

## Examples

```
peakFiles <- system.file("extdata",
    c(
        "rep1.narrowPeak", "rep2.narrowPeak",
        "rep3.narrowPeak"
    ),
    package = "consensusRegions"
)
peaks <- readPeakSets(peakFiles,
    sampleNames = c("r1", "r2", "r3"),
    verbose = FALSE
)
result <- buildConsensus(peaks, verbose = FALSE)

recentrePeaks(result, width = 400)
```

`runConsensus` *Run the whole analysis in one call*

## Description

Chains the steps that a consensus analysis almost always performs in the same order: read the peaks, work out the replicate weights, optionally calibrate the threshold, build the consensus, optionally recentre it on summits, and optionally write it out.

The individual functions remain the way to go when a step needs inspecting before the next one runs, and calibration in particular is worth looking at rather than trusting blind. This is for the case where the settings are already decided and the analysis is being repeated across experiments.

## Usage

```
runConsensus(
  peaks,
  sampleNames = NULL,
  weightMethod = c("equal", "frip", "librarySize", "intrinsic"),
  bamFiles = NULL,
  frip = NULL,
  librarySize = NULL,
  calibrate = FALSE,
  nPermutations = 50L,
  targetFDR = 0.05,
  recentre = FALSE,
  width = 400L,
  outputFile = NULL,
  seqlevelsStyle = "UCSC",
  excludeRegions = NULL,
  BPPARAM = NULL,
  seed = NULL,
  verbose = TRUE,
  ...
)
```

## Arguments

| | |
|---|---|
| `peaks` | Passed to [readPeakSets()]: file paths, a ‘GRangesList‘, or a list of ‘GRanges‘. |
| `sampleNames` | Character vector naming the replicates. Default: `NULL`. |
| `weightMethod` | Passed to [computeReplicateWeights()]. ‘"equal"‘ leaves every replicate counting the same, which is the default and the reproducible choice. Default: `"equal"`. |
| `bamFiles`, `frip`, `librarySize` | Passed to [computeReplicateWeights()] when ‘weightMethod‘ needs them. Default: `NULL`. |
| `calibrate` | Estimate the combined threshold with [calibrateThreshold()] instead of using ‘combinedThreshold‘. Default: `FALSE`. |

| | |
|---|---|
| nPermutations | Number of permutations, passed to [calibrateThreshold()]. Default: 50L. |
| targetFDR | Empirical false discovery rate to aim for, passed to [calibrateThreshold()]. Default: 0.05. |
| recentre | Reduce the consensus to fixed-width regions around summits with [recentrePeaks()]. Sensible for transcription factors and ATAC, wrong for broad domains. Default: FALSE. |
| width | Width of the recentred regions. Default: 400L. |
| outputFile | Optional path. When given the consensus is written there with [exportConsensus()]. Default: NULL. |
| seqlevelsStyle | Chromosome naming style, passed to [readPeakSets()]. Default: "UCSC". |
| excludeRegions | Optional 'GRanges' of blacklisted positions. Default: NULL. |
| BPPARAM | A 'BiocParallelParam' object, or a number of cores as a shorthand for one. Only the calibration uses it. Default: NULL, a BiocParallel::SerialParam() seeded from seed. |
| seed | Number seeding the random positions of the calibration, passed to [calibrateThreshold()], which explains why 'set.seed()' cannot do the same. It is kept with the calibration in the result. Ignored without 'calibrate'. Default: NULL, no seed. |
| verbose | Report progress through the steps. Default: TRUE. |
| ... | Further arguments passed to [buildConsensus()], for example 'combinationMethod', 'minReplicates' or 'mergeMethod'. |

## Value

A [ConsensusRegions-class] object. When 'recentre = TRUE' the consensus it carries is the fixed-width version.

## Author(s)

Sebastian Gregoricchio

## See Also

[readPeakSets()], [computeReplicateWeights()], [calibrateThreshold()], [buildConsensus()], [recentrePeaks()]

## Examples

```
peakFiles <- system.file("extdata",
    c(
        "rep1.narrowPeak", "rep2.narrowPeak",
        "rep3.narrowPeak"
    ),
    package = "consensusRegions"
)

result <- runConsensus(peakFiles,
    sampleNames = c("r1", "r2", "r3"),
```

```
    minReplicates = 2,
    verbose = FALSE
)
result

## the same with measured weights and a calibrated threshold. Five
## permutations are enough to show the shape of it; fifty is the
## working number, and the positions are drawn at random, so the seed
## is what brings the same threshold back.
calibrated <- runConsensus(peakFiles,
    sampleNames = c("r1", "r2", "r3"),
    weightMethod = "frip",
    frip = c(0.21, 0.19, 0.06),
    calibrate = TRUE,
    nPermutations = 5,
    seed = 42,
    verbose = FALSE
)
calibrated
```

---

`show,ConsensusRegions-method`
*Display a ConsensusRegions object*

---

## Description

Display a ConsensusRegions object

## Usage

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

## Arguments

| | |
|---|---|
| `object` | A [ConsensusRegions-class] object. |

## Value

Invisibly 'NULL', called for the printed output.

## Author(s)

Sebastian Gregoricchio

## show,ThresholdCalibration-method *Summary of a threshold calibration*

### Description

Summary of a threshold calibration

### Usage

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

### Arguments

| | |
|---|---|
| object | A [ThresholdCalibration-class] object. |

### Value

'NULL', invisibly. Called for the summary it prints.

### Author(s)

Sebastian Gregoricchio

## ThresholdCalibration-accessors *Accessors for ThresholdCalibration objects*

### Description

Retrieve the components of a [ThresholdCalibration-class] object, the result of [calibrateThreshold()].

### Usage

```
calibrationThreshold(object)

calibrationThresholdNegLog10(object)

calibrationCurve(object)

calibrationTargetFDR(object)

calibrationAchievedFDR(object)
```

```
calibrationSeed(object)

nullStatistics(object)

observedStatistics(object)

## S4 method for signature 'ThresholdCalibration'
calibrationThreshold(object)

## S4 method for signature 'ThresholdCalibration'
calibrationThresholdNegLog10(object)

## S4 method for signature 'ThresholdCalibration'
calibrationCurve(object)

## S4 method for signature 'ThresholdCalibration'
calibrationTargetFDR(object)

## S4 method for signature 'ThresholdCalibration'
calibrationAchievedFDR(object)

## S4 method for signature 'ThresholdCalibration'
calibrationSeed(object)

## S4 method for signature 'ThresholdCalibration'
nullStatistics(object)

## S4 method for signature 'ThresholdCalibration'
observedStatistics(object)

## S4 method for signature 'ThresholdCalibration'
x$name
```

## Arguments

| | |
|---|---|
| object | A [ThresholdCalibration-class] object. |
| x | A [ThresholdCalibration-class] object. |
| name | Slot name, for the '$' shorthand. |

## Value

'calibrationThreshold()' returns the combined p-value threshold, and 'calibrationThresholdNegLog10()' the same value on the -log10 scale. 'calibrationCurve()' returns the FDR curve as a 'data.frame'. 'calibrationTargetFDR()' and 'calibrationAchievedFDR()' return the FDR that was asked for and the one that was reached. 'calibrationSeed()' returns the seed of the permutations, or 'NULL'. 'nullStatistics()' and 'observedStatistics()' return the statistics the curve was built from.

'$' reaches the slots by name, which the result of this function carried as a plain list up to version 0.99.2.

### Author(s)

Sebastian Gregoricchio

### Examples

```
peakFiles <- system.file("extdata",
    c(
        "rep1.narrowPeak", "rep2.narrowPeak",
        "rep3.narrowPeak"
    ),
    package = "consensusRegions"
)
peaks <- readPeakSets(peakFiles,
    sampleNames = c("r1", "r2", "r3"),
    verbose = FALSE
)
calibration <- calibrateThreshold(peaks,
    nPermutations = 5, seed = 42,
    verbose = FALSE
)

calibrationThreshold(calibration)
calibrationSeed(calibration)
```

---

## ThresholdCalibration-class

*ThresholdCalibration class*

---

### Description

The result of [calibrateThreshold()]. It carries the threshold itself, the FDR curve it was read off, the observed and null statistics behind that curve, and the seed the permutations were drawn with, so that a calibration can be inspected and repeated rather than merely trusted.

Until version 0.99.2 this was a plain list, and '$' still reaches the slots so that code written against it keeps working.

### Slots

`threshold` The combined p-value threshold.

`thresholdNegLog10` The same threshold on the -log10 scale.

`achievedFDR` The empirical FDR reached at that threshold.

`targetFDR` The FDR that was asked for.

`fdrCurve` A 'data.frame' with one row per candidate cut, holding 'cut', 'nObserved', 'expectedNull' and 'fdr'.

`observed` The observed combined statistics.

`null` The combined statistics of the shuffled peak sets.

`nPermutations` How many shuffles the null came from.

`seed` The seed the permutations were drawn with, or ‘NULL‘.

`seedSource` ‘"seed"‘ when the seed came from the argument, ‘"BPPARAM"‘ when it came from the backend, ‘"unseeded"‘ when there was none.

## Author(s)

Sebastian Gregoricchio

# Index


$,ThresholdCalibration-method
(ThresholdCalibration-accessors), 29

analysisParameters
(ConsensusRegions-accessors), 15
analysisParameters,ConsensusRegions-method
(ConsensusRegions-accessors), 15
asSummarizedExperiment, 4

buildConsensus, 5

calibrateThreshold, 8
calibrationAchievedFDR
(ThresholdCalibration-accessors), 29
calibrationAchievedFDR,ThresholdCalibration-method
(ThresholdCalibration-accessors), 29
calibrationCurve
(ThresholdCalibration-accessors), 29
calibrationCurve,ThresholdCalibration-method
(ThresholdCalibration-accessors), 29
calibrationResult
(ConsensusRegions-accessors), 15
calibrationResult,ConsensusRegions-method
(ConsensusRegions-accessors), 15
calibrationSeed
(ThresholdCalibration-accessors), 29
calibrationSeed,ThresholdCalibration-method
(ThresholdCalibration-accessors), 29
calibrationTargetFDR
(ThresholdCalibration-accessors), 29
calibrationTargetFDR,ThresholdCalibration-method
(ThresholdCalibration-accessors), 29
calibrationThreshold
(ThresholdCalibration-accessors), 29
calibrationThreshold,ThresholdCalibration-method
(ThresholdCalibration-accessors), 29
calibrationThresholdNegLog10
(ThresholdCalibration-accessors), 29
calibrationThresholdNegLog10,ThresholdCalibration-method
(ThresholdCalibration-accessors), 29
combineEvidence, 11
computeReplicateWeights, 13
consensusRanges
(ConsensusRegions-accessors), 15
consensusRanges,ConsensusRegions-method
(ConsensusRegions-accessors), 15
consensusRegions
(consensusRegions-package), 3
ConsensusRegions-accessors, 15
ConsensusRegions-class, 16
consensusRegions-package, 3
consensusStats
(ConsensusRegions-accessors), 15
consensusStats,ConsensusRegions-method
(ConsensusRegions-accessors), 15

exportConsensus, 17

length,ConsensusRegions-method, 18

nullStatistics
        (ThresholdCalibration-accessors), 29
nullStatistics,ThresholdCalibration-method
        (ThresholdCalibration-accessors), 29

observedStatistics
        (ThresholdCalibration-accessors), 29
observedStatistics,ThresholdCalibration-method
        (ThresholdCalibration-accessors), 29

peakSets (ConsensusRegions-accessors), 15
peakSets,ConsensusRegions-method
        (ConsensusRegions-accessors), 15
plotCalibration, 19
plotJaccard, 20
plotRescue, 21
plotUpset, 22

readPeakSets, 23
recentrePeaks, 24
replicateWeights
        (ConsensusRegions-accessors), 15
replicateWeights,ConsensusRegions-method
        (ConsensusRegions-accessors), 15
runConsensus, 26

show,ConsensusRegions-method, 28
show,ThresholdCalibration-method, 29

ThresholdCalibration-accessors, 29
ThresholdCalibration-class, 31