# Package 'RBPSpecificity'

August 28, 2026

**Type** Package

**Title** RBP Inherent Specificity and Variation Sensitivity Analysis Tool

**Version** 0.99.6

**Description** Provides tools to analyze RNA Binding Protein (RBP) binding specificities from high-throughput sequencing data. Functions include calculating K-mer enrichment, Inherent Specificity (IS), and Mutational Sensitivity (VS), along with visualization of IS and VS. For detailed methodology and applications, please refer to our manuscript (see URL field or vignette).

**License** GPL-3

**Encoding** UTF-8

**biocViews** Software, MotifAnnotation, Sequencing, Coverage, GenomeAnnotation, Visualization, GeneRegulation, KEGG

**BugReports** https://github.com/S00NYI/RBPSpecificity/issues

**URL** https://www.biorxiv.org/content/10.1101/2025.03.28.646018v2, https://github.com/S00NYI/RBPSpecificity, https://github.com/S00NYI/BITS_Specificity

**Depends** R (>= 4.6.0)

**Imports** Biostrings, BSgenome, GenomeInfoDb, GenomicRanges, ggplot2, methods, reshape2, S4Vectors, stats, utils

**Suggests** BiocStyle, BSgenome.Hsapiens.UCSC.hg38, IRanges, knitr, rmarkdown, testthat

**VignetteBuilder** knitr

**Config/roxygen2/version** 8.0.0

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

**git_branch** devel

**git_last_commit** e1b8712

**git_last_commit_date** 2026-07-18

**Repository** Bioconductor 3.24

**Date/Publication** 2026-08-27

**Author** Soon Yi [aut, cre] (ORCID: <https://orcid.org/0000-0002-4535-6532>), National Institutes of Health [fnd] (T32 GM152319)

**Maintainer** Soon Yi <cu.soonyi@gmail.com>

# Contents



---

`RBPSpecificity-package` *RBPSpecificity: Analyze RBP Binding Specificity and Variation Sensitivity*

---

## Description

The 'RBPSpecificity' package provides a suite of tools for researchers working with RNA Binding Proteins (RBPs) to analyze binding specificity from high-throughput sequencing data such as CLIP (CrossLinking and ImmunoPrecipitation) and RBNS (RNA Bind-n-Seq).

It allows users to quantify K-mer enrichment within RBP binding sites, assess the inherent specificity (IS) and the variation sensitivity (VS) of RBP towards the target motifs. The package aims to streamline bioinformatic analyses and provide clear visualizations for interpreting RBP specificity.

**Note:** This package is under active development.

### Key Functions

The main workflow of the package revolves around the following exported functions:

- `motifEnrichment`: Calculates K-mer enrichment from peak data...
- `returnSpecificity`: Quantifies the specificity of RBP...
- `plotSpecificity`: Visualizes the distribution of motif enrichment scores with annotation...
- `returnSensitivity`: Quantifies the sensitivity of RBP...
- `plotSensitivity`: Generates plots to visualize sensitivity

### Author(s)

Soon Yi <cu.soonyi@gmail.com>

### See Also

Useful links:

- https://github.com/S00NYI/RBPSpecificity (Development repository)
- Report bugs at https://github.com/S00NYI/RBPSpecificity/issues

---

## `calEnrichment` *Calculate Background-Corrected K-mer Enrichment*

---

### Description

Computes enrichment scores comparing peak K-mer counts against background using either ANR (total occurrence) or ZOOPS (per-sequence presence) models.

### Usage

```
calEnrichment(peak_counts_df, bkg_data, method = "anr", pseudocount = 1e-04)
```

### Arguments

| | |
|---|---|
| `peak_counts_df` | Data frame from `countKmers()`, with columns MOTIF, COUNT, SEQ_WITH_MOTIF, SEQ_TOTAL. |
| `bkg_data` | List from `countKmersBkg()`, with elements MOTIF, bkg_total_count, bkg_presence_count, bkg_total_seqs. |
| `method` | Character, "anr" (default) or "zoops". |
| `pseudocount` | Numeric, added to fractions/rates before log2 to avoid log(0). Default: 1e-4. |

### Value

A data frame with 10 columns: MOTIF, Score, log2FC, pvalue, padj, target_fraction, bkg_fraction, target_occurence, bkg_occurence, multiplicity. Score is min-max normalized enrichment difference (fraction difference for ZOOPS, rate difference for ANR).

## calSNVSens *Calculate Sensitivity Score for SNVs*

### Description

Internal helper to calculate sensitivity based on reference and variant scores using a specified method.

### Usage

```
calSNVSens(reference_score, variant_scores, method = "1_minus_norm_score")
```

### Arguments

| | |
|---|---|
| reference_score | Numeric score of the reference motif. |
| variant_scores | Named numeric vector or list, where names are SNV motifs and values are their scores. |
| method | Character string defining calculation method (e.g., "1_minus_norm_score", "score_diff", "log2_ratio"). |

### Value

A named numeric vector/list similar to ‘variant_scores‘ but containing the calculated sensitivity scores.

## calSpecificity *Calculate Specificity Core Logic*

### Description

Internal helper to calculate the ratio of a target score to the median score.

### Usage

```
calSpecificity(scores, target_score)
```

### Arguments

| | |
|---|---|
| scores | Numeric vector of all scores. |
| target_score | Numeric value, the score of the specific item of interest. |

### Value

Single numeric value (target_score / median(scores)).

countKmers *Count K-mers in Sequences*

## Description

Counts occurrences of all possible K-mers of a given length within a DNAStringSet. This function is case-sensitive for motifs once generated (all K-mers are generated in uppercase).

## Usage

```
countKmers(sequences, K, type = "DNA")
```

## Arguments

| | |
|---|---|
| sequences | A DNAStringSet object, typically the output from getSequence. |
| K | Integer, the K-mer size (length of motifs). Must be a single positive integer. |
| type | Character string specifying the nucleic acid type for K-mer generation. Accepted values are "DNA" (using A,C,G,T) or "RNA" (using A,C,G,U). The input is case-insensitive. Default: "DNA". |

## Value

A data frame with four columns:

**MOTIF** Character, all possible K-mers in uppercase.

**COUNT** Integer, total occurrences of each K-mer across all input sequences (ANR counting).

**SEQ_WITH_MOTIF** Integer, number of sequences containing the K-mer at least once (ZOOPS counting).

**SEQ_TOTAL** Integer, total number of input sequences.

Returns a data frame with all possible K-mers and 0 counts if input sequences are empty or all shorter than K.

## Examples

```
## Not run:
# Assuming 'seqs' is a DNAStringSet object and 'getSequence' is available
# library(Biostrings)
# seqs_dna <- DNAStringSet(c("ATGCGATGC", "GGCCTTAA", "TTT"))
# countKmers(seqs_dna, K = 3, type = "DNA")

# seqs_rna <- DNAStringSet(c("AUGCGAUGC", "GGCCUUAA", "UUU"))
# countKmers(seqs_rna, K = 3, type = "RNA")

# Empty input
# countKmers(DNAStringSet(), K = 3)

# Sequences shorter than K
# countKmers(DNAStringSet(c("A", "CG")), K = 3)

## End(Not run)
```

countKmersBkg *Calculate Average K-mer Counts from Background Sets*

## Description

This function calculates the average K-mer counts over multiple iterations of background sequence generation. For each iteration, it generates a new set of background sequences using ‘generateBkgSet‘, counts K-mers using ‘countKmers‘, and then averages these counts across all iterations.

## Usage

```
countKmersBkg(
  original_peak_gr,
  K,
  type = "DNA",
  genome_obj,
  bkg_min_dist = 500,
  bkg_iter = 100,
  bkg_max_dist = 1000,
  internal_min_length_for_bkg_seqs = K,
  scramble_bkg = TRUE,
  chunk_size = NULL,
  extension = c(0L, 0L)
)
```

## Arguments

| | |
|---|---|
| original_peak_gr | A GRanges object representing the initial peaks from which background regions are derived. |
| K | Integer, the length of K-mers to count. |
| type | Character string, "DNA" or "RNA" (case-insensitive), defining the alphabet for K-mer generation. Passed to ‘countKmers‘ and used to generate the K-mer universe. Default: "DNA". |
| genome_obj | A BSgenome object for sequence retrieval by ‘generateBkgSet‘. |
| bkg_min_dist | Integer, the minimum absolute distance (in base pairs) for shifting peaks. Passed to ‘generateBkgSet‘. Default: 500. |
| bkg_iter | Integer, the number of background iterations to perform. Default: 100. |
| bkg_max_dist | Integer, the maximum absolute distance (in base pairs) for shifting peaks. Passed to ‘generateBkgSet‘. Default: 1000. |
| internal_min_length_for_bkg_seqs | Integer, minimum length for sequences generated by ‘generateBkgSet‘ (passed as ‘K‘ to ‘generateBkgSet‘ which passes it as ‘min_length‘ to ‘getSequence‘). Default is ‘K‘ from this function. |
| scramble_bkg | Logical, if TRUE (default), scrambles background sequences. Set to FALSE for faster execution when scrambling is not needed. |

### Value

A list with four elements:

**MOTIF** Character vector of all possible K-mers.

**bkg_total_count** Integer vector, total occurrences of each K-mer pooled across all background iterations (ANR).

**bkg_presence_count** Integer vector, number of background sequences containing each K-mer at least once, pooled across all iterations (ZOOPS).

**bkg_total_seqs** Integer, total number of background sequences generated across all iterations.

---

## findTopMotif *Find Top Motif by Score*

---

### Description

Identifies the motif with the highest score in an enrichment data frame. Handles ties by returning the first one encountered.

### Usage

```
findTopMotif(motif_enrichment)
```

### Arguments

`motif_enrichment`
: Data frame with 'MOTIF' and 'Score' (or 'Enrichment') columns. Assumes higher scores are better.

### Value

Character string, the top-scoring motif.

---

## formatSensitivity *Format Sensitivity Scores into a Matrix*

---

### Description

Internal helper to arrange SNV sensitivity scores into the standard Nucleotide x Position matrix format.

### Usage

```
formatSensitivity(
  snv_sensitivity_scores,
  reference_motif,
  reference_score,
  K,
  nucleotides = c("A", "C", "G", "T"),
  sensitivity_method = "1_minus_norm_score"
)
```

### Arguments

| | |
|---|---|
| snv_sensitivity_scores | Named numeric vector/list of sensitivity scores (output from 'calSNVSens'). |
| reference_motif | Character string, the original motif. |
| reference_score | Numeric score of the reference motif. |
| K | Integer, the length of the motif. |
| nucleotides | Character vector of possible nucleotides. |
| sensitivity_method | Character string specifying the sensitivity method. |

### Value

A matrix (rows=nucleotides, cols=positions 1 to K) with sensitivity scores. The cell corresponding to the original nucleotide at a position might be NA or 0.

---

## generateBkgSet *Generate a Single Set of Background Sequences*

---

### Description

Creates one set of background sequences by randomly shifting original peak locations, removing overlaps with original peaks, and extracting sequences. Optionally scrambles the sequences.

### Usage

```
generateBkgSet(
  peak_gr,
  genome_obj,
  K,
  bkg_min_dist,
  bkg_max_dist,
  scramble = TRUE
)
```

### Arguments

| | |
|---|---|
| peak_gr | A GRanges object representing the original genomic peaks. |
| genome_obj | A BSgenome object from which sequences will be extracted. |
| K | Integer, the K-mer length. This is used to set a minimum length for sequences extracted by the internal call to 'getSequence'. |
| bkg_min_dist | Integer, the minimum absolute distance (in base pairs) by which peaks should be shifted. Passed as 'X' to 'genRNDist()'. |
| bkg_max_dist | Integer, the maximum absolute distance (in base pairs) for shifting peaks. Passed as 'max_dist' to 'genRNDist()'. |
| scramble | Logical, if TRUE (default), scrambles the background sequences after extraction. Set to FALSE for faster execution when scrambling is not needed. |

### Value

A DNAStringSet object of background sequences (scrambled or not). Returns an empty DNAStringSet if no valid background regions/sequences could be generated.

---

## generateBkgSetBatched *Generate Batched Background Sequences Across Multiple Iterations*

---

### Description

Batched version of `generateBkgSet` that processes multiple iterations at once, reducing per-call overhead for GRanges shifting, overlap removal, and sequence extraction.

### Usage

```
generateBkgSetBatched(
  peak_gr,
  genome_obj,
  min_seq_length,
  bkg_min_dist,
  bkg_max_dist,
  scramble,
  n_iter,
  extension = c(0L, 0L)
)
```

### Arguments

| | |
|---|---|
| `peak_gr` | A GRanges object representing the original genomic peaks. |
| `genome_obj` | A BSgenome object. |
| `min_seq_length` | Integer, minimum length for extracted sequences. |
| `bkg_min_dist` | Integer, minimum absolute shift distance. |
| `bkg_max_dist` | Integer, maximum absolute shift distance. |
| `scramble` | Logical, whether to scramble sequences. |
| `n_iter` | Integer, number of iterations to batch. |

### Value

A list with two elements:

**sequences** A DNAStringSet of all background sequences.

**iter_tags** Integer vector of iteration indices (1 to n_iter), same length as sequences, indicating which iteration each sequence belongs to.

## genMotifVar *Generate Single Nucleotide Variants for a Motif*

### Description

Create all possible single nucleotide variants.

### Usage

```
genMotifVar(motif, type = "DNA")
```

### Arguments

| | |
|---|---|
| motif | Character string, the reference motif. Should only contain valid nucleotides. |
| type | Character string specifying the nucleic acid type for the inputmotifand the generated variants. Accepted values are "DNA" (A,C,G,T) or "RNA" (A,C,G,U) (default: 'DNA'). |

### Value

A character vector of all unique SNV motifs (excluding the original motif).

## genRNDist *Generate Random Genomic Distances*

### Description

Generates N random distances, ensuring a minimum separation from zero. Used for shifting background regions. Uses runif for sampling.

### Usage

```
genRNDist(N, X = 500, max_dist = 1000)
```

### Arguments

| | |
|---|---|
| N | Integer, the number of distances to generate. Must be positive. |
| X | Integer, the minimum absolute distance from zero. Must be non-negative (default: 500). |
| max_dist | Integer, the maximum absolute distance for sampling (default: 1000). |

### Value

A numeric vector of N random distances.

`minmaxNorm` *Min-Max Normalization*

## Description

Scale a numeric vector to a specified range. Handles cases where min and max are equal.

## Usage

```
minmaxNorm(x, a = 0, b = 1)
```

## Arguments

| | |
|---|---|
| `x` | Numeric vector. |
| `a` | Numeric, the minimum value of the target range (default: 0). |
| `b` | Numeric, the maximum value of the target range (default: 1). |

## Value

Numeric vector scaled to the range [a, b]. NA values in input will result in NA values in output.

`motifEnrichment` *Calculate K-mer Motif Enrichment from Genomic Coordinates*

## Description

Main workflow function for RBPSpecificity. Processes RBP binding peak data to calculate background-corrected K-mer enrichment scores using either ANR (total occurrence) or ZOOPS (per-sequence presence) statistical models.

## Usage

```
motifEnrichment(
  coordinates,
  species_or_build,
  K,
  extension = c(0, 0),
  method = "anr",
  bkg_iter = 100,
  bkg_min_dist = 500,
  bkg_max_dist = 1000,
  scramble_bkg = FALSE,
  nucleic_acid_type = "DNA"
)
```

## Arguments

| | |
|---|---|
| coordinates | A data frame or GRanges object with genomic coordinates. Data frames need chr, start, end columns. |
| species_or_build | Character, genome build (e.g. "hg38"). |
| K | Integer, K-mer length (e.g. 5). |
| extension | Numeric vector of length 2: c(five_prime, three_prime). Positive extends, negative trims. Default: c(0, 0). |
| method | Character, enrichment model: "anr" (default) for total occurrences or "zoops" for per-sequence presence. |
| bkg_iter | Integer, background iterations. Default: 100. |
| bkg_min_dist | Integer, min shift distance. Default: 500. |
| bkg_max_dist | Integer, max shift distance. Default: 1000. |
| scramble_bkg | Logical, scramble background sequences to control for nucleotide composition. Default: FALSE. |
| nucleic_acid_type | Character, "DNA" or "RNA". Default: "DNA". |

## Details

The workflow proceeds as follows: 1. Parses input coordinates into a GRanges object. 2. Loads the specified BSgenome for sequence retrieval. 3. Extracts peak sequences with optional extension/trimming. 4. Counts K-mers (both total and per-sequence presence). 5. Generates background K-mer profiles from shifted regions. 6. Computes enrichment with statistical testing.

**ANR vs. ZOOPS Motif Occurrence Models:** Two statistical models are available via the `method` parameter:

- **ANR (Any Number of Repetitions)**: Counts every occurrence of a motif within each sequence (retains multiplicity). Enrichment is evaluated using a conditional binomial test on aggregate occurrence counts, under the assumption that occurrences are Poisson-distributed.
- **ZOOPS (Zero-or-One Occurrence Per Sequence)**: Treats each sequence as a binary outcome (motif present or absent). Enrichment is evaluated using the cumulative hypergeometric distribution.

ANR is sensitive to motif multiplicity and cooperative binding hotspots, whereas ZOOPS is robust to repetitive elements and compositional noise.

## Value

A data frame with 10 columns:

**MOTIF** K-mer sequence.

**Score** Min-max normalized log2FC, range [0, 1].

**log2FC** Log2 fold-change (method-specific).

**pvalue** Statistical significance (method-specific).

**padj** Benjamini-Hochberg adjusted p-value.

**target_fraction** Fraction of peak sequences containing the motif (ZOOPS).

**bkg_fraction** Fraction of background sequences containing the motif (ZOOPS).

**target_occurence** Total motif occurrences in peaks.

**bkg_occurence** Background occurrence rate (per sequence).

**multiplicity** Average occurrences per containing sequence.

## Examples

```
if (requireNamespace("BSgenome.Hsapiens.UCSC.hg38", quietly = TRUE)) {
    peaks_file <- system.file("extdata",
        "ENCODE_eCLIP_HNRNPC_K562_narrowPeak.bed",
        package = "RBPSpecificity"
    )
    peaks <- read.table(peaks_file,
        header = FALSE, sep = "\t"
    )
    colnames(peaks) <- c(
        "chr", "start", "end",
        "name", "score", "strand"
    )
    result <- motifEnrichment(peaks, "hg38",
        K = 5,
        method = "anr", bkg_iter = 5
    )
    head(result)
}
```

---

`normalizeScores` *Normalize a Vector of Scores*

---

## Description

Applies a specified normalization method to a numeric vector of scores.

## Usage

```
normalizeScores(scores, method = "min_max", pseudocount = 1, ...)
```

## Arguments

| | |
|---|---|
| `scores` | A numeric vector. |
| `method` | Character string (case-insensitive), the normalization method to apply. Supported methods: "min_max" (default), "z_score", "log2", "none". |
| `pseudocount` | Numeric, pseudocount added before log2 transformation (relevant only if ‘method = "log2"‘). Default: 1. |
| `...` | Additional arguments passed to specific normalization methods. For "min_max", these are ‘a‘ and ‘b‘ for the target range (defaults to 0 and 1 respectively if not provided, via ‘minmaxNorm‘ defaults). |

## Value

A numeric vector of normalized scores.

## Examples

```
## Not run:
test_scores <- c(10, 20, 30, 40, 50, NA)
normalizeScores(test_scores, method = "min_max")
normalizeScores(test_scores, method = "min_max", a = 0, b = 10)
normalizeScores(test_scores, method = "z_score")
normalizeScores(c(0, 1, 3, 7, 15, NA), method = "log2") # Uses pseudocount = 1
normalizeScores(c(0, 1, 3, 7, 15), method = "log2", pseudocount = 0.1)
normalizeScores(test_scores, method = "none") # Returns original scores

## End(Not run)
```

---

# peakParse *Parse Peak Input Data*

---

## Description

Validates and converts peak input (currently data frame or GRanges) into a standardized GRanges object. Basic column checking included.

## Usage

```
peakParse(input, standard_chroms_only = TRUE, keep_extra = TRUE)
```

## Arguments

| | |
|---|---|
| input | Can be a data frame (with chr, start, end columns), or a GRanges object. File path input is not yet supported by this helper. |
| keep_extra | Logical, if TRUE, keep additional metadata columns when converting data frame to GRanges (default: TRUE). |

## Value

A GRanges object representing the peaks.

---

# plotSensitivity *Plot Sensitivity Matrix*

---

## Description

Visualizes the sensitivity scores for a motif, typically showing sensitivity to SNVs at each position.

## Usage

```
plotSensitivity(motif_enrichment, motif = NULL, ...)
```

### Arguments

| | |
|---|---|
| motif_enrichment | A data frame containing motif enrichment scores (output from 'motifEnrichment()'). Requires 'MOTIF' and 'Score' columns. |
| motif | Character string, the reference motif for which sensitivity was calculated. If NULL or "top", the top-scoring motif is used (default: NULL). |
| ... | Additional arguments passed to ggplot theme layers or geoms. |

### Value

A ggplot object visualizing the sensitivity matrix (e.g., using points sized or colored by sensitivity).

### Examples

```
# Mock data with variants
motifs <- c("AAAA", "CAAA", "GAAA", "TAAA")
scores <- c(10, 5, 4, 3)
df <- data.frame(MOTIF = motifs, Score = scores)

# Plot sensitivity for "AAAA"
plotSensitivity(df, motif = "AAAA")
```

---

## plotSpecificity *Plot Specificity Distribution*

---

### Description

Generates a histogram of motif enrichment scores, annotated with the median score, the score of a specific motif, and its specificity value.

### Usage

```
plotSpecificity(motif_enrichment, motif = NULL, bins = 50, ...)
```

### Arguments

| | |
|---|---|
| motif_enrichment | A data frame containing motif enrichment scores (output from 'motifEnrichment()'). Requires 'MOTIF' and 'Score' columns. |
| motif | Character string, the specific motif to highlight and calculate specificity for. If NULL or "top", the top-scoring motif is used (default: NULL). |
| bins | Integer, number of bins for the histogram (default: 50). |
| ... | Additional arguments passed to ggplot theme layers or geoms. |

### Value

A ggplot object representing the annotated histogram.

## Examples

```
# Dummy data
df <- data.frame(MOTIF = c("AAAA", "CCCC", "GGGG", "UUUU"), Score = c(10, 2, 5, 1))

# Plot specific motif
plotSpecificity(df, motif = "AAAA")

# Plot top motif (AAAA)
plotSpecificity(df)

# Change number of bins
plotSpecificity(df, bins = 20)
```

---

`returnSensitivity` *Calculate Sensitivity*

---

## Description

Calculates the sensitivity for a target motif or for all motifs. Sensitivity reflects how much the enrichment score changes upon single nucleotide variations (SNVs).

## Usage

```
returnSensitivity(
  motif_enrichment,
  motif = NULL,
  return_type = "specific",
  output_type = "matrix",
  sensitivity_method = "1_minus_norm_score"
)
```

## Arguments

| | |
|---|---|
| `motif_enrichment` | A data frame with 'MOTIF' and 'Score' columns. |
| `motif` | Character string, the reference motif. Only used if 'return_type = "specific"'. If NULL or "top", the top-scoring motif is used. |
| `return_type` | Character string, either "specific" (default) or "all". If "specific", calculates sensitivity for one motif. If "all", calculates an average sensitivity score for every motif. |
| `output_type` | Character string, specifying the output format if 'return_type = "specific"': "matrix" (default) or "number". Ignored if 'return_type = "all"'. |
| `sensitivity_method` | Character string defining how sensitivity is calculated. Default: "1_minus_norm_score". |

## Value

Depends on inputs: - If 'return_type = "specific"' and 'output_type = "matrix"': A sensitivity matrix. - If 'return_type = "specific"' and 'output_type = "number"': A single numeric sensitivity score. - If 'return_type = "all"': A data frame with 'MOTIF' and 'Sensitivity' columns.

## Examples

```
# Mock data with a motif (AAAAA) and some variants
motifs <- c("AAAAA", "CAAAA", "GAAAA", "TAAAA", "ACAAA")
scores <- c(100, 50, 40, 30, 60)
df <- data.frame(MOTIF = motifs, Score = scores)

# Calculate sensitivity matrix
sens_mat <- returnSensitivity(df, motif = "AAAAA", output_type = "matrix")

# Calculate average sensitivity score
sens_score <- returnSensitivity(df, motif = "AAAAA", output_type = "number")
```

---

# returnSensitivityAll *Calculate Sensitivity for All Motifs*

---

## Description

Calculate Sensitivity for All Motifs

## Usage

```
returnSensitivityAll(motif_enrichment)
```

## Arguments

| | |
|---|---|
| motif_enrichment | A data frame with 'MOTIF' and 'Score' columns. |

---

# returnSensitivitySpecific *Calculate Sensitivity for a Specific Motif*

---

## Description

Calculate Sensitivity for a Specific Motif

## Usage

```
returnSensitivitySpecific(
  motif_enrichment,
  motif,
  output_type,
  sensitivity_method
)
```

### Arguments

| | |
|---|---|
| motif_enrichment | A data frame with 'MOTIF' and 'Score' columns. |
| motif | Character string, the reference motif. Only used if 'return_type = "specific"'. If NULL or "top", the top-scoring motif is used. |
| output_type | Character string, specifying the output format if 'return_type = "specific"': "matrix" (default) or "number". Ignored if 'return_type = "all"'. |
| sensitivity_method | Character string defining how sensitivity is calculated. Default: "1_minus_norm_score". |

---

## returnSpecificity *Calculate Specificity*

---

### Description

Calculates the specificity for a given motif based on its enrichment score relative to the median enrichment score of all motifs.

### Usage

```
returnSpecificity(motif_enrichment, motif = NULL, return_type = "specific")
```

### Arguments

| | |
|---|---|
| motif_enrichment | A data frame containing motif enrichment scores, typically output from 'motifEnrichment()'. Must have columns 'MOTIF' and 'Score'. |
| motif | Character string, the specific motif for which to calculate specificity. Only used if 'return_type = "specific"'. If NULL or "top", the top-scoring motif is used. Default: NULL. |
| return_type | Character string, either "specific" (default) or "all". If "specific", returns a single specificity value for the target motif. If "all", returns a data frame with specificity values for every motif. |

### Value

A single numeric specificity value if 'return_type = "specific"', or a data frame with 'MOTIF' and 'Specificity' columns if 'return_type = "all"'.

### Examples

```
# Create a dummy enrichment dataframe
df <- data.frame(MOTIF = c("AAAAA", "CCCCC", "GGGGG"), Score = c(10, 2, 5))

# Calculate specificity for a specific motif
returnSpecificity(df, motif = "AAAAA")

# Calculate specificity for all motifs
returnSpecificity(df, return_type = "all")
```

## scrambleDNA *Scramble DNA Sequence*

### Description

Randomly shuffles nucleotides within a sequence. Input can be character string or DNAString. Output is always DNAString.

### Usage

```
scrambleDNA(seq)
```

### Arguments

| | |
|---|---|
| seq | A DNAString or character string representing a sequence. |

### Value

A DNAString object with shuffled nucleotides. Returns empty DNAString if input is empty or NA.

## scrambleDNASet *Scramble a Set of DNA Sequences*

### Description

Vectorized version of scrambleDNA that operates on an entire DNAStringSet. More efficient than applying scrambleDNA element-wise because it avoids N individual DNAString constructor calls, constructing one DNAStringSet from a character vector at the end.

### Usage

```
scrambleDNASet(dna_set)
```

### Arguments

| | |
|---|---|
| dna_set | A DNAStringSet object. |

### Value

A DNAStringSet object with each sequence independently shuffled. Returns an empty DNAStringSet if input is empty.

## selectGenome *Select and Load BSgenome Object*

### Description

Loads the appropriate BSgenome package based on a species or build identifier. Checks if the package is installed.

### Usage

```
selectGenome(species_or_build)
```

### Arguments

| | |
|---|---|
| species_or_build | Character string (e.g., "hg38", "mm10", "mm39"). Currently supports hg19, hg38, mm9, mm10, mm39. |

### Value

A BSgenome object.

## validateGetSequenceInputs *Get Sequences from GRanges*

### Description

Extracts DNA sequences for a set of genomic ranges from a BSgenome object. Optionally extends or trims ranges from their 5'-end and/or 3'-end in a strand-aware manner, filters by minimum length, and ensures ranges are valid within chromosome boundaries. Original metadata columns (‘mcols‘) and names from the input ‘granges_obj‘ are preserved for the ranges that successfully yield sequences.

### Usage

```
validateGetSequenceInputs(granges_obj, genome_obj, extension, min_length)
```

### Arguments

| | |
|---|---|
| granges_obj | A GRanges object. |
| genome_obj | A BSgenome object. |
| extension | A numeric vector of length 2: ‘c(five_prime, three_prime)‘. Positive values extend the range, negative values trim the range. Extension/trimming is strand-aware: - For + strand (and * strand): 5'-end is start, 3'-end is end - For - strand: 5'-end is end, 3'-end is start Default: ‘c(0, 0)‘ (no change). |
| min_length | Integer, the minimum acceptable length of a range after any extension/trimming. Ranges shorter than this will be removed. Default: 1. |

## Value

A DNAStringSet object containing the extracted sequences. Names of the sequences in the set will correspond to the names of the processed GRanges object (if names were present on the input and preserved). Original metadata columns are preserved on the intermediate GRanges object used for sequence extraction. Returns an empty DNAStringSet if input ‘granges_obj‘ is empty or if all ranges are filtered out.

---

`valInputMotif` *Validate Motif Input*

---

## Description

Checks if a user-provided motif is valid (correct length, exists within a provided set of available motifs). Stops execution if invalid.

## Usage

```
valInputMotif(motif, kmer_size, available_motifs)
```

## Arguments

| | |
|---|---|
| `motif` | Character string, the motif to validate. |
| `kmer_size` | Integer, the expected K-mer length. |
| `available_motifs` | Character vector of all motifs present in the data. |

## Value

Returns ‘TRUE‘ invisibly if valid, otherwise stops with an error.

# Index