Package: motifbreakR (via r-universe)

October 11, 2026

Title A Package for Predicting the Disruptiveness of Single Nucleotide Polymorphisms on Transcription Factor Binding Sites

Version 2.27.4

Description We introduce motifbreakR, which allows the biologist to judge in the first place whether the sequence surrounding the polymorphism is a good match, and in the second place how much information is gained or lost in one allele of the polymorphism relative to another. MotifbreakR is both flexible and extensible over previous offerings; giving a choice of algorithms for interrogation of genomes with motifs from public sources that users can choose from; these are 1) a weighted-sum probability matrix, 2) log-probabilities, and 3) weighted by relative entropy. MotifbreakR can predict effects for novel or previously described variants in public databases, making it suitable for tasks beyond the scope of its original design. Both single nucleotide variants and small insertions and deletions (indels) are supported. Lastly, it can be used to interrogate any genome curated within Bioconductor as a BSgenome package.

Depends R (>= 4.4.0), grid, MotifDb

Imports methods, grDevices, stringr, BiocGenerics, S4Vectors (>= 0.9.25), IRanges, GenomeInfoDb, GenomicRanges, Biostrings, BSgenome, rtracklayer, VariantAnnotation, BiocParallel, motifStack, Gviz, matrixStats, TFMPvalue, SummarizedExperiment, pwalign, DT, bsicons, BiocFileCache, biomaRt, bslib, shiny, vroom

Suggests BSgenome.Hsapiens.UCSC.hg19, SNPlocs.Hsapiens.dbSNP155.GRCh37, knitr, rmarkdown, BSgenome.Drerio.UCSC.danRer7, BiocStyle, BSgenome.Hsapiens.1000genomes.hs37d5, BSgenome.Hsapiens.UCSC.hg19.masked, BSgenome.Hsapiens.NCBI.GRCh38, BSgenome.Hsapiens.UCSC.hg38.masked, BSgenome.Hsapiens.UCSC.hg38, testthat (>= 3.0.0), withr

VignetteBuilder knitr

Encoding UTF-8

License GPL-2

BugReports https://github.com/Simon-Coetzee/motifbreakR/issues

biocViews ChIPSeq, Visualization, MotifAnnotation, Transcription

NeedsCompilation no

Config/testthat/edition 3

Config/roxygen2/version 8.1.0

Config/pak/sysreqs cmake make libgsl0-dev libbz2-dev libicu-dev libjpeg-dev liblzma-dev libpng-dev libuv1-dev libxml2-dev libssl-dev xz-utils zlib1g-dev

Repository https://bioc.r-universe.dev

Date/Publication 2026-10-07 04:30:20 UTC

RemoteUrl https://github.com/bioc/motifbreakR

RemoteRef HEAD

RemoteSha 79191d66d267418c562dc812ba622ad3c1e6ac85

Contents

calculatePvalue . . . 3
encodemotif . . . 4
example.results . . . 5
exportMBbed . . . 6
exportMBtable . . . 7
factorbook . . . 8
findSupportingRemapPeaks . . . 9
hocomoco . . . 11
homer . . . 13
motifbreakR . . . 15
motifbreakR_motif . . . 19
plotMB . . . 20
shiny_motifbreakR . . . 21
snps.from.file . . . 22
snps.from.rsid . . . 24

Index **26**

`calculatePvalue` *Calculate the significance of the matches for the reference and alternate alleles for their PWM*

Description

Calculate the significance of the matches for the reference and alternate alleles for their PWM

Usage

```
calculatePvalue(
  results,
  granularity = NULL,
  BPPARAM = BiocParallel::SerialParam()
)
```

Arguments

`results`	The output of `motifbreakR` that was run with `filterp=TRUE`
`granularity`	Numeric; the granularity to which to round the PWM, larger values compromise full accuracy for speed of calculation. A value of `NULL` does no rounding.
`BPPARAM`	a `BiocParallelParam-class` object controlling parallel evaluation. Try `BiocParallel::registered()` to see what is available; the default `BiocParallel::SerialParam()` gives serial evaluation.

Details

This function is intended to be used on a selection of results produced by `motifbreakR`, and this can be (although not always) a very memory and time intensive process if the algorithm doesn't converge rapidly. The nucleotide background used is the one stored by `motifbreakR` in `attributes(results)$bkg`.

Value

a GRanges object. The same GRanges object that was input as `results`, but with `pValueRef` and `pValueAlt` columns in the output modified from `NA` to the p-value calculated by `TFMsc2pv`. Additionally a `pValueEffect` column that indicates "strong" when the two p-values differ by more than an order of magnitude, otherwise "weak".

Source

Hélène Touzet and Jean-Stéphane Varré (2007) Efficient and accurate P-value computation for Position Weight Matrices. Algorithms for Molecular Biology, **2: 15**.

See Also

See `TFMsc2pv` from the **TFMPvalue** package for information about how the p-values are calculated.

Examples

```
data(example.results)
rs1006140 <- example.results[example.results$SNP_id %in% "rs1006140"]
# low granularity for speed; 1e-6 or 1e-7 recommended for accuracy
rs1006140 <- calculatePvalue(rs1006140, BPPARAM=BiocParallel::SerialParam(), granularity = 1e-4)
```

`encodemotif` *MotifDb object containing motif information from the known and discovered motifs for the ENCODE TF ChIP-seq datasets*

Description

From the abstract: "Recent advances in technology have led to a dramatic increase in the number of available transcription factor ChIP-seq and ChIP-chip data sets. Understanding the motif content of these data sets is an important step in understanding the underlying mechanisms of regulation. Here we provide a systematic motif analysis for 427 human ChIP-seq data sets using motifs curated from the literature and also discovered de novo using five established motif discovery tools. We use a systematic pipeline for calculating motif enrichment in each data set, providing a principled way for choosing between motif variants found in the literature and for flagging potentially problematic data sets. Our analysis confirms the known specificity of 41 of the 56 analyzed factor groups and reveals motifs of potential cofactors. We also use cell type-specific binding to find factors active in specific conditions. The resource we provide is accessible both for browsing a small number of factors and for performing large-scale systematic analyses. We provide motif matrices, instances and enrichments in each of the ENCODE data sets. The motifs discovered here have been used in parallel studies to validate the specificity of antibodies, understand cooperativity between data sets and measure the variation of motif binding across individuals and species."

Usage

```
data(encodemotif)
```

Format

`MotifDb` object of length 2065; to access metadata use mcols(encodemotif)

providerName Name provided by ENCODE

providerId Same as providerName

dataSource `"ENCODE-motif"`

geneSymbol Gene symbol for the transcription factor

geneId Entrez gene id for the transcription factor

geneIdType `"ENTREZ"`

proteinId UNIPROT id for the transcription factor

proteinIdType `"UNIPROT"`

organism "Hsapiens"

sequenceCount NA not available

bindingSequence Consensus sequence for the motif

bindingDomain NA incomplete

tfFamily NA incomplete

experimentType occurs in two forms:

For motifs that were discovered in this study, the format is cellType_source-LabMetadata:MotifFinder#Location for example H1-hESC_encode-Myers_seq_hsa_v041610.2_r1:MEME#2#Intergenic.

For motifs that were "known" the format tends to be TF_source_sourceId for example AP1_jaspar_MA0099.2.

pubmedID "24335146" see Source for more details

Details

Load with data(encodemotif)

Value

MotifList-class object

Source

Pouya Kheradpour and Manolis Kellis (2013 December 13) Systematic discovery and characterization of regulatory motifs in ENCODE TF binding experiments. Nucleic Acids Research, doi:10.1093/nar/gkt1249

See Also

https://compbio.mit.edu/encode-motifs/

Examples

```
data(encodemotif)
encodemotif
```

---

example.results *Example Results from motifbreakR*

Description

Example output of motifbreakR for the variants rs10034824 and rs1006140 (hg19), scored against the HOCOMOCOv11 core motifs from **MotifDb** with method = "ic", filterp = TRUE and threshold = 1e-4, as in the vignette. Used in the examples of the help pages.

Usage

```
data(example.results)
```

Format

GRanges output from motifbreakR

Value

GRanges object. See motifbreakR for information on its structure.

Examples

```
data(example.results)
example.results
```

---

exportMBbed *Export motifbreakR variants to bed file*

Description

Export motifbreakR variants to bed file

Usage

```
exportMBbed(results, file, name = NULL, color = "effect_size")
```

Arguments

results	The output of motifbreakR
file	Character; the file name of the destination file
name	Character; name for the BED track, defaults to "motifbreakR results"
color	Character; one of ref_sig (pValueRef), alt_sig (pValueAlt), best_sig (lowest between pValueRef and pValueAlt), (each of which require pre-computation of p-values with calculatePvalue), or ref_score (pctRef), alt_score (pctAlt), best_score (highest between pctRef and pctAlt), or the default value of effect_size (alleleDiff).

Value

exportMBbed produces an output BED file, with diverging color scale for effect_size (blue representing stronger binding in REF, red representing stronger binding in ALT), or a sequential color scale otherwise (low values as purple, high values as yellow). The score column is either the effect_size (alleleDiff column), the -log10(p-value) (capped at 10), corresponding to pValueRef, pValueAlt, or the best match of the two, or the score pctRef, pctAlt, or the highest match of the two. The name column is formatted SNP_id:REF/ALT:providerId. Additionally a color key is returned indicating the range of values for each color output.

See Also

See exportMBtable for the function that exports the full motifbreakR results as a tab or comma separated table file.

Examples

```
data(example.results)
example.results

exportMBbed(example.results, file = tempfile(fileext = ".bed"), color = "effect_size")
```

---

exportMBtable *Export motifbreakR results to csv or tsv*

Description

Export motifbreakR results to csv or tsv

Usage

```
exportMBtable(results, file, format = "tsv")
```

Arguments

results	The output of motifbreakR
file	Character; the file name of the destination file
format	Character; one of tsv (tab separated values) or csv (comma separated values)

Value

exportMBtable produces an output file containing the output of the motifbreakR function.

See Also

See exportMBbed for the function that exports the motifbreakR results as a BED file, colored by selected score.

Examples

```
data(example.results)
example.results

exportMBtable(example.results, file = tempfile(fileext = ".tsv"), format = "tsv")
```

`factorbook` *MotifDb object containing motif information from around the genomic regions bound by 119 human transcription factors in Factorbook*

Description

From the abstract: "Chromatin immunoprecipitation coupled with high-throughput sequencing (ChIP-seq) has become the dominant technique for mapping transcription factor (TF) binding regions genome-wide. We performed an integrative analysis centered around 457 ChIP-seq data sets on 119 human TFs generated by the ENCODE Consortium. We identified highly enriched sequence motifs in most data sets, revealing new motifs and validating known ones. The motif sites (TF binding sites) are highly conserved evolutionarily and show distinct footprints upon DNase I digestion. We frequently detected secondary motifs in addition to the canonical motifs of the TFs, indicating tethered binding and cobinding between multiple TFs. We observed significant position and orientation preferences between many cobinding TFs. Genes specifically expressed in a cell line are often associated with a greater occurrence of nearby TF binding in that cell line. We observed cell-line-specific secondary motifs that mediate the binding of the histone deacetylase HDAC2 and the enhancer-binding protein EP300. TF binding sites are located in GC-rich, nucleosome-depleted, and DNase I sensitive regions, flanked by well-positioned nucleosomes, and many of these features show cell type specificity. The GC-richness may be beneficial for regulating TF binding because, when unoccupied by a TF, these regions are occupied by nucleosomes in vivo. We present the results of our analysis in a TF-centric web repository Factorbook (http://factorbook.org) and will continually update this repository as more ENCODE data are generated."

Usage

```
data(factorbook)
```

Format

`MotifDb` object of length 79; to access metadata use mcols(factorbook)

providerName Name listed in meme output of 'Supp TableS2.pdf' for the citation indicated below

providerId Same as providerName

dataSource `"FactorBook"`

geneSymbol `NA` these motifs don't have a direct 1 to 1 relationship with a transcription factor

geneId `NA`

geneIdType `NA`

proteinId `NA`

proteinIdType `NA`

organism `"Hsapiens"`

sequenceCount `NA`

bindingSequence Consensus sequence for the motif

bindingDomain `NA`

tfFamily NA

experimentType NA

pubmedID "22955990" see Source for more details

Details

This is the 2012 Factorbook motif set. Factorbook has since been substantially expanded (Factorbook 2.0; Pratt et al. 2022, doi:10.1093/nar/gkab1039), with motifs from ENCODE phase II and III ChIP-seq and HT-SELEX experiments available from https://www.factorbook.org/; these are not included here.

Load with data(factorbook)

Value

MotifList-class object

Source

J Wang, J Zhuang, S Iyer, XY Lin, et al. (2012) Sequence features and chromatin structure around the genomic regions bound by 119 human transcription factors. Genome Research, **22 (9)**, 1798-1812, doi:10.1101/gr.139105.112

See Also

https://www.factorbook.org/

Examples

```
data(factorbook)
factorbook
```

---

findSupportingRemapPeaks

Find Corresponding TF Binding From The ReMap2022 Project

Description

Find Corresponding TF Binding From The ReMap2022 Project

Usage

```
findSupportingRemapPeaks(results, genome, TFClass = FALSE)
```

Arguments

results	The output of motifbreakR
genome	Character; one of: hg38 or hg19 for Homo sapiens, mm10 or mm39 for Mus musculus, dm6 for Drosophila melanogaster, TAIR10_TF or TAIR10_HISTONE for Arabidopsis thaliana
TFClass	Logical; The user may optionally query an expanded motif/transcription factor relationship encompassing the entire potential transcription factor family as implemented by MotifDb based on TFClass.

Details

TFClass argument works for objects loaded in from the MotifDb package. hg19 and mm39 are data from liftOver.

The ReMap catalogues (2022, 2020, 2018, 2015) are under CC BY-NC 4.0 international license, as described in ReMap.

The CC BY-NC 4.0 license correspond to the following terms: Attribution — You must give appropriate credit, provide a link to the license, and indicate if changes were made. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use. Non-Commercial — You may not use the material for commercial purposes. No additional restrictions — You may not apply legal terms or technological measures that legally restrict others from doing anything the license permits.

Value

the results GenomicRanges object output by motifbreakR with the additional columns:

matchingBindingEvent
: The name of the transcription factor that binds over the motif, or NA if none

matchingCellType
: A list corresponding in length to the number of transcription factors in matchingBindingEvent indicating the biotype/celltype that the transcription factor binding was found in.

See Also

associateTranscriptionFactors for information about TFClass. https://remap.univ-amu.fr/ for details about ReMap2022.

Examples

```
data(example.results)

example.results <- findSupportingRemapPeaks(example.results,
                                            genome = "hg19",
                                            TFClass = TRUE)
```

hocomoco *MotifDb object containing motif information from Homo Sapiens Comprehensive Model Collection (HOCOMOCO) of transcription factor (TF) binding models*

Description

From the abstract: "We present the Homo sapiens comprehensive model collection (HOCOMOCO, http://autosome.ru/HOCOMOCO/, http://cbrc.kaust.edu.sa/hocomoco/) containing carefully hand-curated TFBS models constructed by integration of binding sequences obtained by both low- and high-throughput methods. To construct position weight matrices to represent these TFBS models, we used ChIPMunk software in four computational modes, including newly developed periodic positional prior mode associated with DNA helix pitch. We selected only one TFBS model per TF, unless there was a clear experimental evidence for two rather distinct TFBS models. We assigned a quality rating to each model. HOCOMOCO contains 426 systematically curated TFBS models for 401 human TFs, where 172 models are based on more than one data source."

Usage

```
data(hocomoco)
```

Format

MotifDb object of length 426; to access metadata use mcols(hocomoco)

providerName Name provided by HOCOMOCO

providerId ID provided by HOCOMOCO including experiment type

dataSource "HOCOMOCO"

geneSymbol Gene symbol for the transcription factor

geneId Entrez gene id for the transcription factor

geneIdType "ENTREZ"

proteinId UNIPROT id for the transcription factor

proteinIdType "UNIPROT"

organism "Hsapiens"

sequenceCount Number of sequences evaluated for producing the PWM

bindingSequence Consensus sequence for the motif

bindingDomain NA incomplete

tfFamily NA incomplete

experimentType from the HOCOMOCO v9 documentation, quoted here:

"TFBS model identification modes

To construct TFBS models ChIPMunk was run four times: two times (f1) and (f2) with uniform model positional prior and two times (si) and (do) with informative model positional prior.

The min-to-max (f1) model length estimation mode was used with the min length of 7 bp and increasing it by 1 bp until the default max length of 25 bp was reached following the optimal length selection procedure as in Kulakovskiy and Makeev, Biophysics, 2009. For max-to-min (f2) model length estimation mode we started from 25 bp and searched for the best alignment decreasing the length by 1 bp until the minimal length of 7 bp. We also used the single (si) and double box (do) model positional priors in order to simulate DNA helix turn. For a single box, the positional weights are to be distributed as cos2(pi n / T), where T=10.5 is the DNA helix pitch, n is the coordinate within the alignment, and the center of the alignment of the length L is at n=0. During the internal cycle of PWM optimization the PWM column scores are multiplied by prior values so the columns closer to the center of the alignment (n=0) receive no score penalty while the columns around (n = 5,6,-5,-6) contribute much less to the score of the PWM under optimization. The single box model prior was used along with the min-to-max length estimation mode (si). We also used the double box model prior with a shape prior equal to sin2(pin / T), which was used to search for possibly longer double box models in the max-to-min length estimation mode (do).

Model quality assignment

The resulting models were rated (from A to F) according to their quality. Model quality rates from A-to-D were assigned to proteins known to be TFs, including those listed in Schaefer et al., Nucleic Acids Research, 2011 with addition of a number of proteins having relevant models and sufficient evidence to be TFs. The ratings were assigned by human curation according to the following criteria:

Relevant distribution of position-specific information content over alignment columns, which means a model LOGO representation displaying well formed core positions with a high information content surrounded by flanking letters with lower information content; the information content at flanking positions decreasing with the distance from the model core.

"Stability", which means that in more than one of the ChIPMunk modes we obtained models with a similar length, consensus, and comparable number of aligned binding sites, along with a similar shape of model LOGO representation. "Similarity" of the model to the binding sequence consensus for this TF given in the UniProt or other databases, which means similarity of the shape of the model LOGO and TFBS lengths to those of other TFs from the same TF family. "A total number of binding sites" was also considered as a quality measure, as a large set of binding regions (mostly but not limited to ChIP-Seq and parallel SELEX) implies that there are many observations of each letter in any position of the alignment, particularly many observations of non-consensus letters in core positions. In positions with low information content, where there is no strong consensus, all variants have many observations, and thus the observed letter frequencies are less dependent on statistical fluctuations.

Quality A was assigned to high confidence models complying with all four criteria listed in the section above. Quality B was assigned to models built from large sequence sets that failed no more than one out of the three remaining criteria. Quality C was assigned to models built from small sequence sets but (with a number of specifically marked exceptions) complying with the three remaining criteria. Quality D models missed part of the known consensus sequence or had no clearly significant core positions in the TFBS model. Quality E (error) was assigned to models for proteins not convincingly shown to be TFs or to models exhibiting an irrelevant LOGO shape or a wrong consensus sequence. Quality F (failure) was assigned to TFs for which there was no reliable model identified."

pubmedID `"23175603"` see `Source` for more details

Details

This is HOCOMOCO version 9. Newer releases (HOCOMOCO v10, v11 and v13) are available directly from **MotifDb**, for example `subset(MotifDb, dataSource == "HOCOMOCOv13")`.

Load with `data(hocomoco)`

Value

`MotifList-class` object

Source

Kulakovskiy,I.V., Medvedeva,Y.A., Schaefer,U., Kasianov,A.S., Vorontsov,I.E., Bajic,V.B. and Makeev,V.J. (2013) HOCOMOCO: a comprehensive collection of human transcription factor binding sites models. Nucleic Acids Research, **41**, D195–D202.

See Also

https://hocomoco.autosome.org/

Examples

```
data(hocomoco)
hocomoco
```

---

homer *MotifDb object containing motif information from motif databases included in HOMER*

Description

From the website: "Homer includes several motif databases that are used to help annotate results and conduct searches for known motifs. HOMER contains a custom motif database based on independent analysis of mostly ChIP-Seq data sets which is heavily utilized in the software." See http://homer.ucsd.edu/homer/motif/motifDatabase.html for more information on how these files were generated, and Homer's sources.

Usage

```
data(homer)
```

Format

`MotifDb` object of length 247; to access metadata use mcols(homer)

providerName Name provided HOMER

providerId Factor Name provided by HOMER

dataSource `"HOMER"`

geneSymbol Symbol provided by HOMER

geneId Entrez gene id for the transcription factor

geneIdType "ENTREZ"

proteinId UNIPROT id for the transcription factor

proteinIdType "UNIPROT"

organism "Hsapiens"

sequenceCount NA

bindingSequence Consensus sequence for the motif

bindingDomain DBD provided by HOMER

tfFamily NA

experimentType The Celltype, IP, Assay, and GEO id if applicable for the motif

pubmedID "20513432" see Source for more details

Details

Load with data(homer)

Value

MotifList-class object

Source

Heinz S, Benner C, Spann N, Bertolino E et al. (2010 May 28) Simple Combinations of Lineage-Determining Transcription Factors Prime cis-Regulatory Elements Required for Macrophage and B Cell Identities. Mol Cell, **38(4):576-589**. PMID: 20513432

See Also

http://homer.ucsd.edu/homer/index.html http://homer.ucsd.edu/homer/motif/motifDatabase.html http://homer.ucsd.edu/homer/motif/HomerMotifDB/homerResults.html

Examples

```
data(homer)
homer
```

`motifbreakR` *Predict The Disruptiveness Of Genetic Variants On Transcription Factor Binding Sites*

Description

Predict The Disruptiveness Of Genetic Variants On Transcription Factor Binding Sites

Usage

```
motifbreakR(
  snpList,
  pwmList,
  threshold = 0.85,
  filterp = FALSE,
  method = "default",
  show.neutral = FALSE,
  verbose = FALSE,
  bkg = c(A = 0.25, C = 0.25, G = 0.25, T = 0.25),
  BPPARAM = bpparam()
)
```

Arguments

`snpList`	The output of `snps.from.rsid` or `snps.from.file`; may contain SNVs and indels
`pwmList`	An object of class `MotifList` containing the motifs that you wish to interrogate
`threshold`	Numeric; with `filterp = TRUE`, the maximum p-value for a match to be reported; otherwise the minimum score, as a fraction (0-1) of the motif's scoring range (see `pctRef` and `pctAlt`).
`filterp`	Logical; filter by p-value instead of by pct score.
`method`	Character; one of `default`, `log`, `ic`, or `notrans`; see Details.
`show.neutral`	Logical; include neutral changes in the output
`verbose`	Logical; show progress messages (from the workers only when running serially)
`bkg`	the background nucleotide frequencies; either a numeric vector of length 4 named `A`, `C`, `G` and `T`, or one of: `"genome"` frequencies across the whole reference genome. This reads the entire genome sequence into memory. `"aggregate"` frequencies across the sequence windows surrounding the input variants. `"pwm"` the average nucleotide composition of the motifs in `pwmList`. The background is used for the motif pseudocounts, by `method = "log"` and `method = "ic"`, and for p-values. It is stored in `attributes(results)$bkg` and reused by `calculatePvalue`.

BPPARAM a BiocParallelParam-class object controlling parallel evaluation; work is split across workers by motif. Try `BiocParallel::registered()` to see what is available; for example `BiocParallel::SerialParam()` gives serial evaluation, and `BiocParallel::SnowParam()` parallel evaluation on Windows.

Details

motifbreakR works with position probability matrices (PPM). PPM are derived as the fractional occurrence of nucleotides A,C,G, and T at each position of a position frequency matrix (PFM). PFM are simply the tally of each nucleotide at each position across a set of aligned sequences. With a PPM, one can generate probabilities based on the genome, or more practically, create any number of position specific scoring matrices (PSSM) based on the principle that the PPM contains information about the likelihood of observing a particular nucleotide at a particular position of a true transcription factor binding site. What follows is a discussion of the different algorithms that may be employed in calls to the **motifbreakR** function via the `method` argument.

Before scoring, a pseudocount is added to each PPM so that logarithms are defined: `ppm <- (ppm * sequenceCount + bkg) / (sequenceCount + 1)`, where `sequenceCount` is taken from the motif metadata, or 20 when it is `NA`.

Suppose we have a frequency matrix M of width n (*i.e.* a PPM as described above). Furthermore, we have a sequence s also of length n, such that $s_i \in \{A, T, C, G\}, i = 1, \ldots n$. Each column of M contains the frequencies of each letter in each position.

Commonly in the literature sequences are scored as the sum of log probabilities:

Equation 1

$$F(s, M) = \sum_{i=1}^{n} \log_2(\frac{M_{s_i,i}}{b_{s_i}})$$

where b_{s_i} is the background frequency of letter s_i in the genome of interest. This method can be specified by the user as `method='log'`.

As an alternative to this method, we introduced a scoring method to directly weight the score by the importance of the position within the match sequence. This method of weighting is accessed by specifying `method='default'` or `method='ic'`. A general representation of this scoring method is given by:

Equation 2

$$F(s, M) = p_s \cdot \omega_M$$

where p_s is the scoring vector derived from sequence s and matrix M, and w_M is a weight vector derived from M. First, we compute the scoring vector of position scores p

Equation 3

$$p_s = (M_{s_i,i}) \quad \text{for} \quad i = 1, \ldots n$$

and second, for each M a constant vector of weights $\omega_M = (\omega_1, \omega_2, \ldots, \omega_n)$.

There are two methods for producing ω_M. The first, which we call weighted sum (`method='default'`), is the difference of the maximum and minimum values for each column of M:

Equation 4.1

$$\omega_i = \max\{M_i\} - \min\{M_i\} \quad \text{where} \quad i = 1, \ldots n$$

The second variation of this theme is to weight by relative entropy. Thus the relative entropy weight for each column i of the matrix is given by:

Equation 4.2

$$\omega_i = \sum_{j \in \{A,C,G,T\}} M_{j,i} \log_2(\frac{M_{j,i}}{b_j}) \quad \text{where} \quad i = 1, \ldots n$$

where b_j is again the background frequency of the letter j.

Thus, there are 3 scoring algorithms to apply via the `method` argument. The first is the standard summation of log probabilities (`method='log'`). The second and third are the weighted sum and information content methods (`method='default'` and `method='ic'`) specified by equations 4.1 and 4.2, respectively. Additionally `method='notrans'` scores the sum of the untransformed probabilities, $\sum p_s$. **motifbreakR** assumes a uniform background nucleotide distribution (b) in equations 1 and 4.2 unless otherwise specified by the user. Since we are primarily interested in the difference between alleles, background frequency is not a major factor, although it can change the results. Additionally, inclusion of background frequency introduces potential bias when collections of motifs are employed, since motifs are themselves unbalanced with respect to nucleotide composition. With these cautions in mind, users may override the uniform distribution if so desired. For all three methods, **motifbreakR** scores and reports the reference and alternate alleles of the sequence ($F(s_{\text{REF}}, M)$ and $F(s_{\text{ALT}}, M)$), and provides the matrix scores $p_{s_{\text{REF}}}$ and $p_{s_{\text{ALT}}}$ of the SNP (or variant). The scores are scaled as a fraction of scoring range 0-1 of the motif matrix, M. If either of $F(s_{\text{REF}}, M)$ and $F(s_{\text{ALT}}, M)$ is greater than a user-specified threshold (default value of 0.85) the SNP is reported. The effect of a variant is classified from the absolute difference in score between the alleles (`alleleDiff`): `"strong"` above 0.7, `"weak"` from 0.4 to 0.7 and `"neut"` below 0.4. By default **motifbreakR** does not report neutral effects; set `show.neutral = TRUE` to include them.

Additionally, now, with the use of `TFMPvalue-package`, we may filter by p-value of the match. This is unfortunately a two step process. First, by invoking `filterp=TRUE` and setting a threshold at a desired p-value e.g 1e-4, we perform a rough filter on the results by rounding all values in the PWM to two decimal places, and calculating a scoring threshold based upon that. The second step is to use the function `calculatePvalue()` on a selection of results which will change the `pValueRef` and `pValueAlt` columns in the output from `NA` to the p-value calculated by `TFMsc2pv`. This can be (although not always) a very memory and time intensive process if the algorithm doesn't converge rapidly.

Value

a GRanges object containing:

`SNP_id`	the identifier of the variant
`REF`	the reference allele for the variant
`ALT`	the alternate allele for the variant
`varType`	one of `SNV`, `Insertion`, `Deletion`, or `Other`

`windowIdxRef`, `windowIdxAlt`
: the start of the best motif match on the reference and alternate allele, relative to the first base of the variant

`motifStrandRef`, `motifStrandAlt`
: the strand (`+` or `-`) of the best motif match on the reference and alternate allele

`geneSymbol`
: the geneSymbol corresponding to the TF of the TF binding motif

`dataSource`
: the source of the TF binding motif

`motifID`, `providerName`, `providerId`
: the name and id provided by the source

`seqMatch`
: the sequence matched by the TF binding motif on the allele with the stronger match (see `strongerIn`)

`pwmConsensus`
: the consensus sequence of the TF binding motif, on the strand of the stronger match

`pctRef`
: The score as determined by the scoring method, when the sequence contains the reference variant allele, normalized to a scale from 0 - 1. If `filterp` = `FALSE`, this is the value that is thresholded.

`pctAlt`
: The score as determined by the scoring method, when the sequence contains the alternate variant allele, normalized to a scale from 0 - 1. If `filterp` = `FALSE`, this is the value that is thresholded.

`scoreRef`
: The score as determined by the scoring method, when the sequence contains the reference variant allele

`scoreAlt`
: The score as determined by the scoring method, when the sequence contains the alternate variant allele

`pValueRef`
: p-value for the match for the pctRef score, initially set to `NA`; only present when `filterp` = `TRUE`. see `calculatePvalue` for more information

`pValueAlt`
: p-value for the match for the pctAlt score, initially set to `NA`; only present when `filterp` = `TRUE`. see `calculatePvalue` for more information

`strongerIn`
: `Ref` or `Alt`, the allele with the stronger motif match

`alleleDiff`
: The difference between the score on the reference allele and the score on the alternate allele

`alleleEffectSize`
: The ratio of the `alleleDiff` and the maximal score of a sequence under the PWM

`effect`
: one of `"strong"`, `"weak"`, or `"neut"` (neutral, only with `show.neutral` = `TRUE`) indicating the strength of the effect; see Details.

each SNP in this object may be plotted with `plotMB`

See Also

See `snps.from.rsid` and `snps.from.file` for information about how to generate the input to this function and `plotMB` for information on how to visualize its output

Examples

```
library(BSgenome.Hsapiens.UCSC.hg19)
# prepare variants
load(system.file("extdata",
                 "pca.enhancer.snps.rda",
                 package = "motifbreakR")) # loads snps.mb
pca.enhancer.snps <- sample(snps.mb, 20)
# Get motifs to interrogate
data(hocomoco)
motifs <- sample(hocomoco, 50)
# run motifbreakR
results <- motifbreakR(pca.enhancer.snps,
                       motifs, threshold = 0.85,
                       method = "ic",
                       BPPARAM=BiocParallel::SerialParam())
```

`motifbreakR_motif` *MotifDb object containing motif information from the motif databases of HOCOMOCO, Homer, FactorBook and ENCODE*

Description

This object contains all the `MotifList-class` objects that were generated for this package. See the individual help sections for `hocomoco`, `homer`, `factorbook`, and `encodemotif`, for how the data is formatted.

Usage

```
data(motifbreakR_motif)
```

Format

`MotifDb` object of length 2817; to access metadata use mcols(motifbreakR_motif)

Details

Load with `data(motifbreakR_motif)`

Value

`MotifList-class` object

Source

Kulakovskiy,I.V., Medvedeva,Y.A., Schaefer,U., Kasianov,A.S., Vorontsov,I.E., Bajic,V.B. and Makeev,V.J. (2013) HOCOMOCO: a comprehensive collection of human transcription factor binding sites models. Nucleic Acids Research, **41**, D195–D202.

Heinz S, Benner C, Spann N, Bertolino E et al. (2010 May 28) Simple Combinations of Lineage-Determining Transcription Factors Prime cis-Regulatory Elements Required for Macrophage and B Cell Identities. Mol Cell, **38(4):576-589**. PMID: 20513432

J Wang, J Zhuang, S Iyer, XY Lin, et al. (2012) Sequence features and chromatin structure around the genomic regions bound by 119 human transcription factors. Genome Research, **22 (9)**, 1798-1812, doi:10.1101/gr.139105.112

Pouya Kheradpour and Manolis Kellis (2013 December 13) Systematic discovery and characterization of regulatory motifs in ENCODE TF binding experiments. Nucleic Acids Research, doi:10.1093/nar/gkt1249

See Also

hocomoco, homer, factorbook, and encodemotif

Examples

```
data(motifbreakR_motif)
motifbreakR_motif
```

`plotMB` *Plot a genomic region surrounding a genomic variant, and potentially disrupted motifs*

Description

Plot a genomic region surrounding a genomic variant, and potentially disrupted motifs

Usage

```
plotMB(
  results,
  rsid,
  reverseMotif = TRUE,
  effect = c("strong", "weak"),
  altAllele = NULL
)
```

Arguments

`results`	The output of `motifbreakR`
`rsid`	Character; the identifier (`SNP_id`) of the variant to be visualized
`reverseMotif`	Logical; for motifs matched on the "-" strand, show the reverse complement of the motif (`TRUE`) or the motif reversed only (`FALSE`)
`effect`	Character; show motifs that are strongly affected `c("strong")`, weakly affected `c("weak")`, or both `c("strong", "weak")`
`altAllele`	Character; The default value of `NULL` uses the first (or only) alternative allele for the SNP to be plotted.

Details

plotMB produces output showing the location of the variant on the chromosome, the reference and alternate sequence of the + strand, the footprint of any motif that is disrupted by the variant, and the DNA sequence motif(s), with the position of the variant marked on each motif. The altAllele argument is included for variants like rs1006140 where multiple alternate alleles exist, the reference allele is A, and the alternate can be G,T, or C. plotMB only plots one alternate allele at a time.

Value

plots a figure representing the results of motifbreakR at the location of a single SNP, returns invisible NULL.

See Also

See motifbreakR for the function that produces output to be visualized here, also snps.from.rsid and snps.from.file for information about how to generate the input to motifbreakR function.

Examples

```
data(example.results)
example.results

library(BSgenome.Hsapiens.UCSC.hg19)
plotMB(results = example.results, rsid = "rs1006140", effect = "strong", altAllele = "C")
```

---

shiny_motifbreakR *Run Shiny version of the motifbreakR package*

Description

Run Shiny version of the motifbreakR package

Usage

```
shiny_motifbreakR()
```

Value

returns a shinyAppDir that launches the shiny app when printed.

Examples

```
library(motifbreakR)

app <- shiny_motifbreakR()

if (interactive()) {
  shiny::runApp(app)
}
```

`snps.from.file` *Import variants from a BED file or VCF file for use in motifbreakR*

Description

Import variants from a BED file or VCF file for use in motifbreakR

Usage

```
snps.from.file(
  file = NULL,
  dbSNP = NULL,
  search.genome = NULL,
  format = "bed",
  biomart.dataset = NULL,
  check.unnamed.for.rsid = FALSE
)

variants.from.file(
  file = NULL,
  dbSNP = NULL,
  search.genome = NULL,
  biomart.dataset = NULL,
  format = "bed"
)
```

Arguments

`file`	Character; the path to a BED or VCF file; see Details for the required format
`dbSNP`	OPTIONAL; an object of class SNPlocs to look up rsids; run `available.SNPs()` to list the available SNPlocs packages
`search.genome`	an object of class BSgenome for the species you are interrogating; see `available.genomes` for a list of species
`format`	Character; one of `bed` or `vcf`
`biomart.dataset`	a Mart object from `useEnsembl` specifying the `snps` biomart, which dataset i.e., `hsapiens_snp`, and which version i.e., `111` or `GRCm39`. This will override `SNPlocs` and must be compatible with search.genome selection, and will query from `biomaRt`, which may be considerably faster than a lookup from a `SNPlocs` object.
`check.unnamed.for.rsid`	Logical; for BED input, look up variants named in the form chr:pos:ref:alt in dbSNP and use the rsid if one is found. Lookup may be slow and requires either `dbSNP` or `biomart.dataset`.

Details

snps.from.file reads a BED or VCF file and generates the input for motifbreakR. Both SNVs and indels are imported.

For BED files see https://genome.ucsc.edu/FAQ/FAQformat.html#format1 for a complete description of the format. Our convention deviates in that there is a required format for the name field: name is either chromosome:start:REF:ALT or an rsid from dbSNP (in which case dbSNP or biomart.dataset must be supplied). For example rs123 would be entered in the first format as chr7:24966446:C:A. Several alternate alleles may be separated by commas, e.g. chr7:24966446:C:A,G.

For VCF files, multi-allelic records are split into one variant per alternate allele. Variants whose alleles contain characters outside the IUPAC DNA alphabet (for example symbolic alleles such as <DEL>) are removed with a warning.

Value

a GRanges object with one range per variant and alternate allele (multi-allelic variants are split into one range per alternate allele), containing:

SNP_id	The identifier of the variant
REF	The reference allele, as a DNAStringSet
ALT	The alternate allele, as a DNAStringSet

The name of the BSgenome package is stored in attributes(x)$genome.package for use by motifbreakR.

Functions

- variants.from.file(): Identical to snps.from.file; retained for backwards compatibility from when snps.from.file excluded indels by default.

See Also

See motifbreakR for analysis; See snps.from.rsid for an alternate method for generating a list of variants.

Examples

```
library(BSgenome.Drerio.UCSC.danRer7)
snps.bed.file <- system.file("extdata", "danRer.bed", package = "motifbreakR")
# see the contents
read.table(snps.bed.file, header = FALSE)
#import the BED file
snps.mb <- snps.from.file(snps.bed.file,
                          search.genome = BSgenome.Drerio.UCSC.danRer7,
                          format = "bed")
```

snps.from.rsid *Import SNPs from rsid for use in motifbreakR*

Description

Import SNPs from rsid for use in motifbreakR

Usage

```
snps.from.rsid(
  rsid = NULL,
  dbSNP = NULL,
  search.genome = NULL,
  biomart.dataset = NULL
)
```

Arguments

rsid	Character; a character vector of rsid values from dbSNP
dbSNP	an object of class SNPlocs to look up rsids; run available.SNPs() to list the available SNPlocs packages
search.genome	an object of class BSgenome for the species you are interrogating; see available.genomes for a list of species.
biomart.dataset	a Mart object from useEnsembl specifying the snps biomart, which dataset i.e., hsapiens_snp, and which version i.e., 111 or GRCm39. This will override SNPlocs and must be compatible with search.genome selection, and will query from biomaRt, which may be considerably faster than a lookup from a SNPlocs object.

Details

snps.from.rsid takes an rsid, or a character vector of rsids, and generates the input required by motifbreakR. rsids that are not found are dropped with a warning.

Value

a GRanges object with one range per variant and alternate allele (multi-allelic variants are split into one range per alternate allele), containing:

SNP_id	The identifier of the variant
REF	The reference allele, as a DNAStringSet
ALT	The alternate allele, as a DNAStringSet

The name of the BSgenome package is stored in attributes(x)$genome.package for use by motifbreakR.

See Also

See motifbreakR for analysis; See snps.from.file for an alternate method for generating a list of variants.

Examples

```
library(BSgenome.Hsapiens.UCSC.hg19)
library(SNPlocs.Hsapiens.dbSNP155.GRCh37)
snps.file <- system.file("extdata", "pca.enhancer.snps", package = "motifbreakR")
snps <- as.character(read.table(snps.file)[,1])
snps.mb <- snps.from.rsid(snps[1],
                          dbSNP = SNPlocs.Hsapiens.dbSNP155.GRCh37,
                          search.genome = BSgenome.Hsapiens.UCSC.hg19)

## alternatively using biomaRt (requires access to the Ensembl servers)

library(biomaRt)
library(BSgenome.Hsapiens.UCSC.hg38)
ensembl_snp <- useEnsembl(biomart = "snps",
                          dataset = "hsapiens_snp",
                          version = "112")
snps.mb <- snps.from.rsid(snps,
                          biomart.dataset = ensembl_snp,
                          search.genome = BSgenome.Hsapiens.UCSC.hg38)
```

Index

∗ **datasets**
encodemotif, 4
example.results, 5
factorbook, 8
hocomoco, 11
homer, 13
motifbreakR_motif, 19

associateTranscriptionFactors, *10*
available.genomes, *22*, *24*
available.SNPs, *22*, *24*

calculatePvalue, 3, *6*, *15*, *17, 18*

encodemotif, 4, *19, 20*
example.results, 5
exportMBbed, 6, *7*
exportMBtable, 7, *7*

factorbook, 8, *19, 20*
findSupportingRemapPeaks, 9

GRanges, *6*

hocomoco, 11, *19, 20*
homer, 13, *19, 20*

motifbreakR, *3*, *5, 6*, *10*, 15, *21*, *23–25*
motifbreakR_motif, 19
MotifDb, *4*, *8*, *10, 11*, *13*, *19*

plotMB, *18*, 20

shiny_motifbreakR, 21
shinyAppDir, *21*
snps.from.file, *15*, *18*, *21*, 22, *25*
snps.from.rsid, *15*, *18*, *21*, *23*, 24

TFMsc2pv, *3*, *17*

useEnsembl, *22*, *24*

variants.from.file *(*snps.from.file*)*, 22