Package 'postNet'

November 12, 2026

Type Package

Title Post-transcriptional network modeling

Description
A tool that enables in silico identification, integration, and modeling of mRNA features that influence post-transcriptional regulation of gene expression at a transcriptome-wide scale.

Version 1.0.0

License MIT + file LICENSE

Encoding UTF-8

LazyLoad true

Depends R (>= 4.5.0),

Imports dplyr, plyr, Biostrings, data.table, gridExtra, seqinr, R.utils, reshape2, vioplot, stringr, plotrix, gplots, ggplot2, ggrepel, anota2seq, memes, GenomicRanges, IRanges, WriteXLS, randomForest, igraph, Boruta, ROCR, caret, msigdb, ExperimentHub, AnnotationHub, GSEABase, fgsea, org.Hs.eg.db, org.Mm.eg.db, RColorBrewer, httr2, rvest, umap, clusterProfiler (>= 4.18.4), gage, withr, grDevices, graphics, methods, stats, utils, tools, BiocFileCache, curl

LinkingTo Rcpp, BH

Suggests knitr, rmarkdown, BiocStyle, pdftools, magick, testthat (>= 3.0.0)

biocViews GeneExpression, GeneRegulation, Transcriptomics, RiboSeq, RNASeq, Sequencing, Annotation, Network, FeatureExtraction

VignetteBuilder knitr

URL https://github.com/kszkop/postNet

BugReports https://github.com/kszkop/postNet/issues

Config/testthat/edition 3

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

git_branch RELEASE_3_23

git_last_commit 7f82693

git_last_commit_date 2026-04-28

Repository Bioconductor 3.23

Date/Publication 2026-11-11

Author Krzysztof Szkop [aut, cre],
Kathleen Watt [aut],
Ola Larsson [aut]

Maintainer Krzysztof Szkop <krzysztof.szkop@gmail.com>

Contents

postNet-package . . . 3
codonCalc . . . 4
codonUsage . . . 6
contentAnalysis . . . 10
contentMotifs . . . 12
featureIntegration . . . 16
foldingEnergyAnalysis . . . 22
gageAnalysis . . . 25
get_signatures . . . 28
goAnalysis . . . 31
goDotplot . . . 34
gseaAnalysis . . . 36
gseaPlot . . . 40
humanSignatures . . . 42
lengthAnalysis . . . 45
miRNAanalysis . . . 47
motifAnalysis . . . 51
mouseSignatures . . . 53
plotFeaturesMap . . . 56
plotSignatures . . . 58
plotSignatures_ads . . . 60
postNetData-class . . . 63
postNetExample . . . 65
postNetStart . . . 68
postNetVignette . . . 77
ptn_background . . . 79
ptn_check_models . . . 80
ptn_codonAnalysis . . . 81
ptn_codonSelection . . . 82
ptn_colours . . . 83
ptn_dataIn . . . 84
ptn_effect . . . 85
ptn_features . . . 86
ptn_GAGE . . . 87
ptn_geneID . . . 89
ptn_geneList . . . 90

ptn_GO 91
ptn_GSEA 92
ptn_id 94
ptn_miRNA_analysis 95
ptn_miRNA_to_gene 97
ptn_model 98
ptn_motifGeneList 100
ptn_motifSelection 101
ptn_networkGraph 103
ptn_selectedFeatures 104
ptn_selection 106
ptn_sequences 107
ptn_species 108
ptn_version 108
rfPred 109
signaturesHeatmap 111
signCalc 113
slopeFilt 114
uorfAnalysis 116

Index **119**

`postNet-package` *Post-transcriptional network modelling with postNet*

Description

Tools for the identification, integration, and modelling of mRNA features that influence post-transcriptional regulation of gene expression at a transcriptome-wide scale.

Details

For an introduction to the package and example workflows, see the package vignette:

```
vignette("postNet")
```

Author(s)

See `packageDescription("postNet")`.

See Also

Useful starting points:

```
vignette("postNet")
```

Package overview and workflow examples are provided in the vignette.

codonCalc *Quantify and compare codon or amino acid usage across gene sets*

Description

After running codonUsage, the codonCalc function can be used to calculate the number or frequency of selected codons or amino acids (AA) for each gene. These values can be used in the downstream featureIntegration analysis. In addition, codon or amino acid content for gene sets of interest can be compared against background, or other gene sets. Several options are available for plotting the comparisons.

Usage

```
codonCalc(ptn,
          featsel,
          analysis = "codon",
          unit = "count",
          comparisons = NULL,
          plotOut = TRUE,
          plotType = "ecdf",
          pdfName = NULL)
```

Arguments

ptn	A postNetData object.
featsel	A named list of vectors specifying one or more codons or amino acids to quantify and/or visualize. For use in the downstream featureIntegration, it is usually desirable to select those with high frequency, and the highest and lowest odds ratios identified using the codonUsage function. This list can be easily generated using the S4 method ptn_codonSelection.
analysis	A string specifying whether to assess "codon", or "AA" (amino acid) usage. The default is "codon".
unit	A string specifying the unit to quantify. The options are "count" or "freq" for frequency. Here, frequency corresponds to the number of codons relative to all codons in the gene. The default is "count".
comparisons	A list of numeric vectors specifying pairwise comparisons between gene sets defined in the regulation or geneList slots of the postNetData object. Use 0 to denote the background set. For example, list(c(0,2), c(1,2)) compares gene set 2 to the background and gene set 1 to gene set 2. The default is NULL.
plotOut	Logical indicating whether PDF files of plots are generated. If FALSE, only the values for the selected analysis and unit will be returned, and statistical comparisons between gene sets will not be performed. The default is TRUE.
plotType	If plotOut is TRUE, indicate the type of output plot to generate. Options are "boxplot", "violin", or "ecdf". The default is "ecdf".
pdfName	Name to be appended to output PDF files. The default is NULL.

Details

The number or frequency of selected codons or amino acids is calculated for each gene, and a two-sided Wilcoxon Rank Sum test is performed to identify significant differences between gene sets of interest, or against the background. If `plotOut` is `TRUE`, all indicated comparisons between different sets of genes will be performed and plotted.

Value

A named list of numeric vectors for selected codons or amino acids with the calculated counts or frequencies for each gene. The elements in each vector are named by gene ID. This list can be used as input for the downstream featureIntegration analysis. The `codonCalc` function can also write PDF files of output plots with statistical comparisons between gene sets of interest.

See Also

codonUsage
ptn_codonSelection
ptn_codonAnalysis

Examples

```
 tmp <- tempfile(fileext = ".pdf")

# load example data:
 data("postNetExample", package = "postNet")
 ptn <- postNetExample$ptn

# Run codon usage analysis


 ptn <- codonUsage(ptn = ptn,
          annotType = "ptnCDS",
          sourceSeq = "load",
          analysis = "codon",
          codonN = 1,
          pAdj = 0.01,
          rem5 = TRUE,
          plotHeatmap = FALSE,
          thresOddsUp = 0.4,
          thresFreqUp = 0.4,
          thresOddsDown = 0.4,
          thresFreqDown = 0.4,
          subregion = NULL,
          subregionSel = NULL,
          comparisons = list(c(1,2)),
          plotType_index = "violin",
          pdfName = tmp)

# Select codons of interest with high frequency, and the highest and lowest odds ratios
 codons <- ptn_codonSelection(ptn, comparison = 1)
```

```
# Calculate and plot ecdfs of codons counts, and compare between gene sets of interest
 codonCounts <- codonCalc(ptn = ptn,
                          analysis = "codon",
                          featsel = codons,
                          unit = "count",
                          comparisons = list(c(1,2)),
                          pdfName = tmp,
                          plotType = "ecdf")

 str(codonCounts)
```

---

codonUsage *Perform differential codon or amino acid usage analysis between gene sets*

Description

The codonUsage function performs differential codon or amino acid (AA) usage analysis. The analysis enumerates single (or multiple, e.g. dicodons) codons or amino acids, and identifies those that are enriched or depleted between different gene sets of interest. Codon usage indexes can also be compared between gene sets of interest, and differences in average counts and frequencies are visualized as scatterplots and heatmaps.

Usage

```
codonUsage(ptn,
           annotType = "ptnCDS",
           sourceSeq = "load",
           analysis,
           codonN = 1,
           pAdj = 0.01,
           rem5 = TRUE,
           plotHeatmap = TRUE,
           thresOddsUp = 0.25,
           thresFreqUp = 0.25,
           thresOddsDown = 0.25,
           thresFreqDown = 0.25,
           subregion = NULL,
           subregionSel = NULL,
           comparisons,
           plotType_index = "boxplot",
           setSeed = NULL,
           pdfName = NULL)
```

Arguments

ptn	A postNetData object.

annotType — A string specifying the reference coding sequences to be used for the analysis. The options are: "ccds" (the NCBI Consensus CDS database), or "ptnCDS" (the CDS reference sequence annotations already stored in the postNetData object). The default is "ptnCDS".

sourceSeq — A string specifying the source of annotType. This parameter is only required if annotType = "ccds". The options are: "create" to automatically prepare reference sequences from CCDS database, or "load" to load pre-prepared sequences for selected species. The default is "load".

analysis — A string specifying the type of the analysis to perform. The options are "codon", or "AA" to assess amino acid usage. The default is "codon".

codonN — An integer specifying the number of codon combinations to count. For example, codonN = 1 will count each individual codon, while codonN = 2 will count dicodons, and so on. Note that this option is only available for analysis = "codon" and that when codonN > 1, odds ratios will not be calculated and additional codon usage indexes and plots will not be generated. The default is 1.

pAdj — A numeric value specifying the adjusted p-value threshold for selecting significant Chi-square test results. The default is 0.01.

rem5 — Logical specifying whether codons with fewer than five counts should be filtered out from the contingency table (counts of codons by gene set) before performing the Chi-square test. The default is TRUE.

plotHeatmap — Logical specifying whether to plot a heatmap of the standardized residuals for each codon from the Chi-square test. The default is TRUE.

thresOddsUp — A numeric value between 0 and 1 specifying the percentage threshold for the odds ratio in selecting codons or AAs in the enriched set. For example, thresOddsUp = 0.25 indicates that the top 25% of enriched codons with the highest odds ratios will be selected. The default is 0.25.

thresFreqUp — A numeric value between 0 and 1 specifying the percentage threshold for the frequency in selecting codons or AAs in the enriched set. For example, thresFreqUp = 0.25 indicates that the top 25% of the most frequently occurring enriched codons are selected. This metric is combined with thresOddsUp to select the most frequent codons with the highest odds ratio. The default is 0.25.

thresOddsDown — A numeric value between 0 and 1 specifying the percentage threshold for the odds ratio in selecting codons or AAs in the depleted set. For example, thresOddsDown = 0.25 indicates that the top 25% of depleted codons with the lowest odds ratios will be selected. The default is 0.25.

thresFreqDown — A numeric value between 0 and 1 specifying the percentage threshold for the frequency in selecting codons or AAs in the depleted set. For example, thresFreqDown = 0.25 indicates that the top 25% of the most frequently occurring depleted codons are selected. This metric is combined with thresOddsDown to select the most frequent codons with the lowest odds ratio. The default is 0.25.

subregion — Optionally, it is possible to specify a more specific subregion of the sequences to either select or exclude from the analysis. This is done by providing a numeric value indicating the number of nucleotides from either the start of the sequence region if positive, or the end if negative. For example, subregion =

50, would denote the first 50 nucleotides of the CDS, while subregion = -50, would denote the last 50 nucleotides of the CDS. The default is NULL.

subregionSel — If a value is provided for subregion, indicate whether the subregion should be selected (limiting the analysis to this region) or excluded by specifying either "select" or "exclude". The default is NULL.

comparisons — A list of numeric vectors specifying pairwise comparisons between gene sets defined in the regulation or geneList slots of the postNetData object. Use 0 to denote the background set. For example, list(c(0,2), c(1,2)) compares gene set 2 to the background and gene set 1 to gene set 2. The default is NULL.

plotType_index — A string to indicate the type of output plot to generate for codon indexes. Options are "boxplot", "violin", or "ecdf". The default is "boxplot".

pdfName — Name to be appended to output PDF files. The default is NULL.

setSeed — If selection is "random", provide a single integer value to be used with set.seed to ensure that the random isoform selection is reproducible between different runs of the postNetStart function. Any integer can be provided (for example 123), however, the same value must be provided between different analyses for isoform selection to be reproducible.

Details

In the first step of the analysis, for each gene set comparison, a Chi-square test is applied to assess significant differences in codon or AA usage. Then, for codons or AAs passing significance thresholds, the odds ratio for each desired comparison pair is calculated and plotted against the frequency. The codons or AAs of interest with differential usage between gene sets are usually identified as those with a high/low odds ratio and high frequency. Codons or AAs of interest can then be further assessed using the codonCalc function (see example below).

Several codon indexes are also compared between gene sets of interest, including the CAI (codon adaptation index), CBI (Codon Bias Index), FOP (frequency of optimal codons), L_aa (Number of amino acids in the protein), tAI (tRNA adaptation index).

Value

The codonUsage function returns an updated postNetData object with results compiled and stored in the codons slot as an S4 object of class postNetCodons.

The codonAnalysis slot provides a summary of calculations for all codons for each gene, and stores an S4 object of class postNetCodonsAll with the following slots: geneID, codon, AA, count, frequency, AACountPerGene, and relative_frequency. Here, frequency corresponds to the number of codons relative to all codons in the gene, and relative_frequency corresponds to the number of codons relative to all synonymous codons in the gene. The contents of the "codonAnalysis" slot can be accessed using the ptn_codonAnalysis S4 method.

The codonSelection slot stores a named list of the selected codons or AAs that were significantly enriched or depleted (according to the selected thresholds) in the specified comparisons. This list can be used as input for the codonCalc function (see example below). The contents of the "codonSelection" slot can be accessed using the ptn_codonSelection S4 method.

The codonUsage function also generates plots comparing the codon indexes described above between gene sets, a clustered heatmap of the Chi-square residuals for each codon between the gene

sets of interest, and plots comparing the average codon count and frequency between the gene sets of interest (codons encoding the same amino acid are coloured and connected by lines).

References

The NCBI Consensus CDS database: https://www.ncbi.nlm.nih.gov/projects/CCDS/CcdsBrowse.cgi

CAI:
Sharp, P. M., and W. H. Li, (1987). The codon adaptation index a measure of directional synonymous codon usage bias, and its potential applications. Nucleic Acids Research 15: 1281-1295.

CBI:
Bennetzen, J. L., and B. D. Hall, (1982). Codon selection in yeast. Journal of Biological Chemistry 257: 3026-3031.

FOP:
Ikemura, T., (1981). Correlation between the abundance of Escherichia coli transfer RNAs and the occurrence of the respective codons in its protein genes: a proposal for a synonymous codon choice that is optimal for the E. coli system. Journal of Molecular Biology 151: 389-409.

tAI:
dos Reis M, Savva R, Wernisch L. Solving the riddle of codon usage preferences: a test for translational selection. Nucleic Acids Res. 2004 Sep 24;32(17):5036-44.

See Also

codonCalc
ptn_codonAnalysis
ptn_codonSelection

Examples

```
 tmp <- tempfile(fileext = ".pdf")

# load example data:
 data("postNetExample", package = "postNet")
 ptn <- postNetExample$ptn

# Run codon usage analysis of single codons:
 ptn <- codonUsage(ptn = ptn,
          annotType = "ptnCDS",
          sourceSeq = "load",
          analysis = "codon",
          codonN = 1,
          pAdj = 0.01,
          rem5 = TRUE,
          plotHeatmap = FALSE,
```

```
            thresOddsUp = 0.25,
            thresFreqUp = 0.25,
            thresOddsDown = 0.25,
            thresFreqDown = 0.25,
            subregion = NULL,
            subregionSel = NULL,
            comparisons = list(c(1,2)),
            plotType_index = "violin",
            pdfName = tmp)

# Access the full results of the analysis for each gene:

 codonResults <- ptn_codonAnalysis(ptn)
 str(codonResults)

# Select codons of interest that were significantly enriched
# or depleted with high frequency, and the highest and lowest odds ratios:

 codons <- ptn_codonSelection(ptn, comparison = 1)
 str(codons)

# Count and plot eCDFs for sets of enriched/depleted codons,
# and compare between gene sets of interest:

 codonCounts  <- codonCalc(ptn = ptn,
                           analysis = "codon",
                           featsel = codons,
                           unit = "count",
                           comparisons = list(c(1,2)),
                           pdfName = tmp,
                           plotType = "ecdf")

 str(codonCounts)
```

`contentAnalysis` *Calculate nucleotide content of mRNA sequence regions and compare between gene sets*

Description

The `contentAnalysis` function calculates the nucleotide content (as a percentage) of mRNA sequences (5'UTR, 3'UTR, and CDS) for each gene. Nucleotide content of different sequence regions can be compared between gene sets of interest, or against background. Several options are available for plotting the comparisons.

Usage

```
contentAnalysis(ptn,
                contentIn,
                region,
```

```
subregion = NULL,
subregionSel = NULL,
comparisons = NULL,
plotOut = TRUE,
plotType = "boxplot",
pdfName = NULL)
```

Arguments

ptn	A postNetData object.
contentIn	A character vector specifying one or more nucleotide(s) or nucleotide combinations to quantify. These can be any selection or combination of A, T, G, or C. For example `c("GC")`, or `c("G", "A")`. When `region` includes `"CDS"`, nucleotide content at specific codon positions can also be quantified. For example, `c("GC3")` calculates GC content in the third codon position, and `c("A12")` calculates the percentage of A content in the first and second codon positions.
region	A character vector specifying the sequence region(s) to be analyzed. This can be `"UTR5"`, `"UTR3"`, `"CDS"`, or any combination of the regions.
subregion	Optionally, specify a more specific subregion of the sequences to either select or exclude from the analysis. Provide an integer indicating the number of nucleotides to include or exclude from either the start of the sequence region if positive, or the end if negative. For example, `subregion = 50` denotes the first 50 nucleotides of the sequence region(s) specified by `region`, while `subregion = -50` denotes the last 50 nucleotides of the sequence region(s). The default is `NULL`.
subregionSel	If a value is provided for `subregion`, indicate whether the subregion should be selected (limiting the analysis to this region) or excluded by specifying either `"select"` or `"exclude"`. The default is `NULL`.
comparisons	A list of numeric vectors specifying pairwise comparisons between gene sets defined in the `regulation` or `geneList` slots of the `postNetData` object. Use 0 to denote the background set. For example, `list(c(0,2), c(1,2))` compares gene set 2 to the background and gene set 1 to gene set 2. The default is `NULL`.
plotOut	Logical indicating whether PDF files of plots are generated. If `FALSE`, only the nucleotide content for the sequence regions of interest will be returned, and statistical comparisons between gene sets will not be performed. The default is `TRUE`.
plotType	If `plotOut` is `TRUE`, indicate the type of output plot to generate. Options are `"boxplot"`, `"violin"`, or `"ecdf"`. The default is `"boxplot"`.
pdfName	Name to be appended to output PDF files. The default is `NULL`.

Details

When `plotOut = TRUE`, a two-sided Wilcoxon Rank Sum test is performed to identify significant differences in mRNA sequence region nucleotide content between gene sets of interest, and/or against the background gene set.

Value

A named list of vectors for selected mRNA sequence regions with the calculated nucleotide content for each gene. This list can be used as input for the downstream featureIntegration analysis. The elements in each vector are named with the gene ID. The `contentAnalysis` function can also return PDF files of output plots with statistical comparisons between gene sets of interest.

Examples

```
 tmp <- tempfile(fileext = ".pdf")

# load example data:
 data("postNetExample", package = "postNet")
 ptn <- postNetExample$ptn

# Calculate the GC of the first 50 nucleotides of the 5'UTR for each gene,
# and compare between translationUp genes vs. background,
# translationDown genes vs. background,
# and translationUp vs. translationDown genes:

 content_UTR5_first50 <- contentAnalysis(ptn = ptn,
                           region = c("UTR5"),
                           subregion = 50,
                           subregionSel = "select",
                           comparisons = list(c(0,1),c(0,2),c(1,2)),
                           contentIn = c("GC"),
                           plotOut = TRUE,
                           plotType = "ecdf",
                           pdfName = tmp)

 str(content_UTR5_first50)
```

`contentMotifs` *Detect, quantify, and compare sequence motifs between gene sets*

Description

The `contentMotifs` function quantifies the number or position of motifs in sequence regions of interest and performs statistical comparisons between gene sets of interest, or against background. Motifs can be specified directly, or the output of the motifAnalysis function, which detects de-novo motifs using STREME (part of the MEME-Suite), can be supplied. Finally, the function can also implement pqsfinder to identify G-quadruplexes.

Usage

```
contentMotifs(ptn,
              motifsIn,
              seqType = "dna",
              dist = 1,
```

```
                min_score = 47,
                unitOut = "number",
                resid = FALSE,
                region,
                subregion = NULL,
                subregionSel = NULL,
                comparisons = NULL,
                pdfName = NULL,
                plotOut = TRUE)
```

Arguments

ptn	A postNetData object.
motifsIn	A character vector specifying the motif sequences to be detected and quantified. Ambiguities can be specified using IUPAC codes or [ ] (bracket) annotations. Motifs discovered with motifAnalysis can be retrieved using ptn_motifSelection and used. Optionally, G-quadruplexes can be specified as "G4".
seqType	A string specifying the type of sequence being provided to search for motifs. The options are "dna", "rna", or "protein". The default is "dna".
dist	A numeric value specifying the minimal distance between motifs (in nucleotides for DNA/RNA or amino acids for protein). The default is 1.
min_score	A numeric value specifying the threshold for the quality of the G-quadruplexes prediction. This is only required when motifsIn includes the "G4" option. The default is 47. See pqsfinder for details.
unitOut	A string to specify whether the output should be the "number" of motifs detected per gene (for a given sequence region), or the "position" of the detected motifs in the mRNA sequence (start and end). Note that only the output with "number" can be used in the downstream analysis with the featureIntegration function. The default is "number".
resid	Logical indicating if the quantification of motifs should be corrected for the length of the sequences. If TRUE, the values returned are the residuals from the linear model: number of motifs ~ log2(sequence region length). Note that this option can only be set to TRUE when unitOut = "number". See details below for usage recommendations. The default is FALSE.
region	A character vector specifying the sequence region(s) to be analyzed. This can be "UTR5", "UTR3", "CDS", or any combination of the regions.
subregion	Optionally, it is possible to specify a more specific subregion of the sequences to either select or exclude from the analysis. This is done by providing a numeric value indicating the number of nucleotides from either the start of the sequence region if positive, or the end if negative. For example, subregion = 50 denotes the first 50 nucleotides of the sequence region(s) specified by region, while subregion = -50 denotes the last 50 nucleotides of the sequence region(s). The default is NULL.
subregionSel	If a value is provided for subregion, indicate whether the subregion should be selected (limiting the analysis to this region) or excluded by specifying either "select" or "exclude". The default is NULL.

comparisons	A list of numeric vectors specifying pairwise comparisons between gene sets defined in the regulation or geneList slots of the postNetData object. Use 0 to denote the background set. For example, list(c(0,2), c(1,2)) compares gene set 2 to the background and gene set 1 to gene set 2. The default is NULL.
pdfName	Name to be appended to output PDF files. The default is NULL.
plotOut	Logical indicating whether PDF files of plots are generated. If FALSE, only the values specified with unitOut will be returned, and statistical comparisons between gene sets will not be performed. Note that plots will only be generated if unitOut = "number". The default is TRUE.

Details

A two-sided Wilcoxon Rank Sum test is performed to identify significant differences in motif content between gene sets of interest, or against the background gene set.

Note that the contentMotifs function uses a more strict method of sequence matching to count motifs than the MEME-Suite, and for this reason some motifs identified with motifAnalysis, which implements STREME, may have divergent quantifications between methods. Using a more stringent threshold with the stremeThreshold parameter in motifAnalysis may remedy this. Alternatively, motifs identified with STREME can be counted using the runFimo tool provided with the MEME-Suite.

If the output motifs will be used in downstream analysis with featureIntegration, consider carefully whether to apply the resid correction. If the length of the sequence regions will be included in the featureIntegration model, it is not recommended to correct the motif content for the sequence length. However, if length will not be included in featureIntegration, the correction can be performed.

Value

If unitOut = "number", the output will be a named list of vectors with the number of motifs detected for each gene in each sequence region specified. This list can be used as input for the downstream featureIntegration analysis. If unitOut = "position", the output will be a named list of lists with the start and end positions of each motif for each gene in each sequence region specified. These positions cannot be used in featureIntegration. The contentMotifs function can also return PDF files of output plots with statistical comparisons between gene sets of interest when unitOut = "number".

References

If you use the in-built "G4" option in your analysis, please cite pqsfinder:

Hon J, Martínek T, Zendulka J, Lexa M. pqsfinder: an exhaustive and imperfection-tolerant search tool for potential quadruplex-forming sequences in R. Bioinformatics. 2017 Nov 1;33(21):3373-3379. doi: 10.1093/bioinformatics/btx413. PMID: 29077807.

See Also

pqsfinder
runFimo
motifAnalysis

Examples

```
 tmp <- tempfile(fileext = ".pdf")

# load example data:
 data("postNetExample", package = "postNet")
 ptn <- postNetExample$ptn

# ----------------------------------------------------------------------
# Example 1: Detect and quantify a given motif within 5'UTRs
# ----------------------------------------------------------------------

# Detect and quantify a given motif within 5'UTRs,
# and compare between translationUp vs. translationDown genes

 UTR5_SGCSGCS_num <- contentMotifs(ptn = ptn,
                                   motifsIn = "SGCSGCS",
                                   region = c("UTR5"),
                                   comparisons = list(c(1,2)),
                                   dist = 1,
                                   unitOut = "number",
                                   pdfName = tmp,
                                   plotOut = TRUE)

 str(UTR5_SGCSGCS_num)

 # Now find the positions of the motif in the 5'UTR

 UTR5_SGCSGCS_pos <- contentMotifs(ptn = ptn,
                                   motifsIn = "SGCSGCS",
                                   region = c("UTR5"),
                                   comparisons = list(c(1,2)),
                                   dist = 1,
                                   unitOut = "position")

 str(UTR5_SGCSGCS_pos)

 ## Not run:
# ----------------------------------------------------------------------
# Example 2: Detect and quantify motifs identified using motifAnalysis
# ----------------------------------------------------------------------

# First run motifAnalysis to identify significantly enriched motifs in the 3'UTR
 ptn <- motifAnalysis(ptn = ptn,
                      streMeThreshold = 0.05,
                      minwidth = 6,
                      memePath = "/meme/bin",
                      region = c("UTR3"))

# Quantify the presence of these motifs in transcripts,
# and compare between  translationUp vs. translationDown genes,
# adjusting for the length of the sequence.
```

```
UTR3_denovo_motifs <- contentMotifs(ptn = ptn,
                                    motifsIn = ptn_motifSelection(ptn, region = "UTR3"),
                                     region = c("UTR3"),
                                     comparisons = list(c(1,2)),
                                     dist = 1,
                                     resid = TRUE,
                                     unitOut = "number",
                                     pdfName = "example",
                                     plotOut = TRUE)

str(UTR3_denovo_motifs)

# ------------------------------------------------------------------
# Example 3: Detect and quantify potential G-quadruplex-forming sequences
# ------------------------------------------------------------------

# Quantify the presence of G4 motifs in 5'UTR and CDS,
# and compare between translationUp vs. translationDown genes.

G4_UT5_CDS <- contentMotifs(ptn = ptn,
                            motifsIn = "G4",
                            region = c("UTR5","CDS"),
                            comparisons = list(c(1,2)),
                            dist = 1,
                            resid = FALSE,
                            unitOut = "number",
                            pdfName = "example",
                            plotOut = TRUE)

str(G4_UT5_CDS)

## End(Not run)
```

`featureIntegration` *Feature integration with post-transcriptional network modelling*

Description

The `featureIntegration` function is used to model changes in post-transcriptional regulation and identify cis and/or trans factors (features of mRNA molecules, signatures of upstream regulators, etc.) that can explain the observed regulatory effects. The method integrates mRNA features and signatures obtained in previous steps of the postNet workflow (or from custom sources), and has several options for implementation, including:

- Stepwise linear regression modelling and network analysis. This analysis uses hierarchical multiple regression to identify features explaining changes in the regulatory effect, rank them according to their importance, and reveals independent and combinatorial effects (for example, co-occurrence of regulatory features in the same mRNA molecules). The results of this analysis can be visualized using network plots showing the association of features to regulatory effects, and the relationships between features.

- Random Forest feature selection and classification. This analysis implements Random Forest classification using randomForest. Feature selection is performed using Boruta to identify the set of features that best classify genes according to their regulation. The Random Forest model can then also be used to predict regulation for other gene lists or datasets using the rfPred function.

Both implementations of featureIntegration produce statistics and plots visualizing the relationship between the regulatory effect and the selected features. The relationships between selected features identified by featureIntegration and regulatory effects can be further explored using UMAP visualizations with the plotFeaturesMap function.

Usage

```
featureIntegration(ptn,
                   features,
                   lmfeatGroup = NULL,
                   lmfeatGroupColour = NULL,
                   analysis_type,
                   regOnly = TRUE,
                   allFeat = FALSE,
                   useCorel = TRUE,
                   covarFilt = 20,
                   NetModelSel = "omnibus",
                   comparisons = NULL,
                   fdrUni = 0.05,
                   stepP = 0.05,
                   pdfName = NULL)
```

Arguments

ptn	A postNetData object.
features	A named list of numeric vectors corresponding to features that will be included in modelling of regulatory effects. Each vector element in the list must have names corresponding to gene IDs. These vectors quantifying features can be the outputs of previous steps of the analysis, including from contentAnalysis, lengthAnalysis, etc. Custom inputs can also be included if they are named numeric vectors with geneIDs matching those in the postNetData object. Finally, lists of geneIDs without numeric values, for example genes regulated downstream of a pathway, can be converted to binary feature vectors using the signCalc function.
lmfeatGroup	If analysis_type = "lm", optionally provide a character vector specifying how features will be grouped in the network plot. Groups of features will be circled, with circle colours corresponding to the groups specified. For example, if the input list for features includes 5'UTR length, uORFs, enriched codons, 3'UTR length, and a 3'UTR motif, a grouping vector could be: c("UTR5", "UTR5", "CDS", "UTR3", "UTR3"), to organize the layout of features in the network plot according to regions of the mRNA sequence. Grouping vectors must be the same length as the input features list. Default is NULL.

lmfeatGroupColour
: If analysis_type = "lm", optionally provide a named character vector specifying colours for the different groups defined with lmfeatGroup that will be used in the network plot. Colour vectors must be the same length as the number of groups in lmfeatGroup, and be named according to group. For example, c("red", "blue", "green") with names c("UTR5", "CDS", "UTR3"). Default is NULL.

analysis_type
: A string specifying the method for the analysis. The options are "lm" to perform stepwise linear regression modelling, or "rf" to perform a random forest-based analysis.

regOnly
: If analysis_type = "lm", provide a logical specifying whether modelling should be performed using only regulated genes (i.e., those included in geneList), or with all genes in the dataset. If regOnly = TRUE, only regulated genes will be used, if regOnly = FALSE, all genes will be considered. Default is TRUE.

allFeat
: If analysis_type = "lm", provide a logical specifying which input features should be included in the stepwise linear regression modelling. If allFeat = TRUE, all input features will be included, if FALSE, only features passing the FDR threshold specified with fdrUni in univariate models will be included. Default is FALSE.

useCorel
: If analysis_type = "lm", provide a logical specifying if substantial correlations between features should be displayed as edges in network plots and coefficients be returned along with the final model. If TRUE, Pearson's correlation coefficients will be calculated between associated features, allowing visualization of how features co-vary in network plots. A threshold for the strength of association between features in order for correlation coefficients to be calculated must be provided using the covarFilt parameter. If FALSE, edges in network plots will instead represent the magnitude of the change in F-value for a given feature between sequential steps of stepwise regression to display the strength of associations between features, and no correlation coefficients will be calculated. Default is TRUE.

covarFilt
: If analysis_type = "lm", provide an integer specifying the minimum threshold for the change in F-value between sequential steps of stepwise regression to display the strength of associations between features in network graphs. Larger changes in F-values upon addition of a new feature to the model are indicative of a stronger association between features. Default is 20, indicating that edges between feature nodes in network plots will be displayed between features where there was a change of 20 or more in the F-value for one feature upon addition of the other to the model. Note that this parameter is only applied to network analysis and not F-table visualizations.

NetModelSel
: If analysis_type = "lm", provide a string specifying how the variance explained should be displayed in the network plot. If NetModelSel = "omnibus", the total variance explained by the omnibus model (the smallest model to explain the greatest proportion of variance) will be displayed. The size of each node (representing features) is proportional to the variance explained, where covariance is assigned to the most influential feature (a rank-based greedy model). If NetModelSel = "adjusted", the covariance will be eliminated and the values displayed will reflect the adjusted total variance explained, and the independent contribution of each feature. Default is "omnibus".

comparisons — If regOnly = TRUE, provide a list of numeric vectors specifying pairwise comparisons between gene sets defined in the regulation or geneList slots of the postNetData object. Use 0 to denote the background set. For example, list(c(0,2), c(1,2)) compares gene set 2 to the background and gene set 1 to gene set 2. The default is NULL. If both regulation and geneList are provided, geneList is appended to the end so numbering will continue sequentially.

fdrUni — If analysis_type = "lm", provide a numeric value between 0 and 1 specifying the FDR threshold to select significant features in univariate analyses. When allFeat = FALSE, only the features passing this threshold will be included in the stepwise regression modelling. Default is 0.05.

stepP — If analysis_type = "lm", provide a numeric value between 0 and 1 specifying the p-value threshold to determine if features should be retained in the omnibus model after each step of stepwise regression. Default is 0.05.

pdfName — Name to be appended to output PDF files.

Details

The featureIntegration analysis assesses whether a catalog of cis and/or trans regulatory features appear to modulate the observed post-transcriptional regulation. Details of the implementations and considerations for feature inputs are described below.

Stepwise linear regression modelling and network analysis: featureIntegration uses stepwise linear regressions to model changes in post-transcriptional regulation across subsets of mRNAs and assesses the contribution of each feature included in the modelling in a hierarchical manner. This allows features to be ranked according to their ability to explain changes in the observed regulatory effect. Due to the hierarchical approach, identification of both distinct (independent), and overlapping (covarying) associations between features and regulatory effects is possible. featureIntegration with stepwise linear regression is performed in three phases:

- **Phase 1**: First, each candidate feature is evaluated separately in univariate linear models to identify significant associations between the regulatory effect and individual features.
- **Phase 2**: Second, starting with the feature that best explained changes in the regulatory effect from univariate models, forward stepwise regression is performed by adding features to the model in an iterative fashion, keeping covariance assigned to the most influential feature (a rank-based greedy model). In each step, the best performing model (the feature with the strongest association with the regulatory effect) is retained and features that fail to explain additional variance are discarded. The resulting omnibus model represents the smallest set of features that can explain the greatest proportion of variance in the regulatory effect. This step ranks features according to their importance, and reveals dependence between them. For example, regulation may depend partially on the length of the 5'UTR, but also on the GC content, or the presence of a motif, etc.
- **Phase 3**: Finally, the independent contribution of each feature identified in phase 2 is determined. This is done by removing covariance from the omnibus model (that was previously assigned to the most influential features in phase 2) to provide the adjusted contribution of each feature. This phase provides the percentage of variance in the regulatory effect explained by a given feature, independent of all other features.

The results of all three phases of the stepwise regression modelling are summarized in table outputs and visualized as network plots (described below).

Random Forest feature selection and classification: `featureIntegration` can also apply Breiman's random forest algorithm (implemented in the `randomForest` package) to classify genes according to their regulation, and identify features associated with regulatory classes (for example, translationally activated or suppressed genes). This implementation is carried out in two steps:

- **Pre-modelling and feature selection**: Genes are first divided into training (70%) and validation (30%) sets. Modelling is performed and importance is calculated for all features included in the input. Feature selection is then performed using `Boruta` to identify all input features relevant to model performance.
- **Final modelling**: Next, modelling using `randomForest` is repeated using the set of selected features. The performance of the model is assessed using Receiver Operating Characteristic (ROC) curves (implemented with `performance`), and feature importance from the final model is reported as Mean Decrease Accuracy (see `importance` for details), where higher values indicate more important features.

Feature inputs for modelling: Many different types of features can be included in modelling with `featureIntegration`. From the postNet workflow, the outputs of the `lengthAnalysis`, `contentAnalysis`, `uorfAnalysis`, `foldingEnergyAnalysis`, `contentMotifs`, and `codonCalc` functions can be supplied directly as features. Gene signatures, such as those included with the package (see `humanSignatures` or `mouseSignatures`) can also be converted to feature inputs using the `signCalc` function. In addition, custom feature inputs can be supplied depending on the application and research question.

In general, features that can be included in modelling can be either numeric, or categorical. Numeric features can be both continuous (for example, sequence region nucleotide content or length), or discrete (for example, the number of uORFs in 5'UTRs). Categorical features can also be included by converting to binary variables. This is useful for evaluating gene signatures in relation to regulatory outcomes, but can also be applied in many scenarios where mRNA can be divided into two classes (for example those with, or without a certain property, etc.). It is also possible to supply categorical features with a directionality. For example, a feature that can be increased, decreased, or unchanged could be coded as 1, -1, or 0 for each gene to be included in modelling.

Careful consideration should be given when selecting the catalog of features to include in modelling. For example, highly, or perfectly correlated variables cannot be supplied together in the same model. This can be relevant when examining the percentage of nucleotide content for each base, since the content of G and C may often be highly correlated, and the sum of A, T, G, and C will add to 100%. In instances where features are too highly correlated, a warning message will be displayed and one of the correlated features must be removed from the input. In addition, when enumerating some features such as codon composition or folding energy, these values can be corrected for the length of the sequence region. However, it would then not be advisable to also include the sequence region length as a separate feature in the modelling. Please see the postNet vignette for more in-depth examples and discussion on the selection of features and interpretation of results.

Value

The `featureIntegration` function returns an updated `postNetData` object with results compiled and stored in the `features` and `featureIntegration` slots.

The `features` slot stores a `data.frame` with the features used as input for feature integration modelling, where rows correspond to genes, and columns correspond to features. The input feature `data.frame` can be retrieved using the ptn_features S4 method.

The stepwise regression implementation of `featureIntegration` will store results for each comparison in the `lm` slot. This includes the results of univariate and stepwise regression modelling, and the final omnibus and adjusted models. Modelling results can be retrieved using the ptn_model S4 method. The features selected as significant in the omnibus model are listed in the `selectedFeatures` slot, along with the proportion of variance in the regulatory effect explained by each feature (either in omnibus, or adjusted models depending on the selection in the `NetModelSel` parameter). Selected features can be retrieved using the ptn_selectedFeatures S4 method. Finally, the network analysis results are stored as an igraph object, and can be retrieved using the ptn_networkGraph S4 method.

The resulting network plot is output as a PDF. Modelling is summarized in a PDF file with a table displaying results for features significant in univariate models, a table of F-values from linear models in stepwise regression, and a table summarizing features in the final omnibus and adjusted models. Optionally, if `useCorel = TRUE`, a table of Pearson's correlation coefficients between features will also be displayed. For F-value tables, the feature explaining the greatest proportion of variance in the regulatory effect at each step is highlighted in green, while those removed from the model due to inability to significantly explain variance are marked in red. Features highlighted in orange indicate those where there was a substantial change in F-value (an increase or decrease of at least 50%) following the addition of another feature to the model. Larger changes in F-value are indicative of a stronger relationship between features. Note that this highlighting is independent of the threshold applied with the `covarFilt` parameter, which controls edge selection for network analyses. For more details please refer to the `postNet` vignette.

The random forest implementation of `featureIntegration` will store results for each comparison in the `rf` slot. The pre-model, feature selection, and final model are stored as objects of class `randomForest` and `Boruta`. The selected features are listed in the `selectedFeatures` slot, along with their importance (Mean Decrease Accuracy) from the final model. Feature importance calculated by `Boruta` is plotted and output as a PDF file. PDF files are also output with ROC curves for the final model, and the importance of selected features.

Both implementations of `featureIntegration` output PDF files with scatterplots for each individual feature identified in final stepwise regression or random forest models, with Pearson's correlation coefficient between the feature and regulatory effect (two-sided test). The linear trend line, and cubic smoothing spline (allowing assessment of linearity) are displayed.

References

If you use random forest analysis, please cite:

Sing T, Sander O, Beerenwinkel N, Lengauer T (2005). "ROCR: visualizing classifier performance in R." Bioinformatics, 21(20), 7881. http://rocr.bioinf.mpi-sb.mpg.de.

Liaw A, Wiener M (2002). "Classification and Regression by randomForest." R News, 2(3), 18-22. https://CRAN.R-project.org/doc/Rnews/.

Kursa MB., Witold R. Rudnicki. (2010). "Feature Selection with the Boruta Package." Journal of Statistical Software, 36(11), 1–13. https://doi.org/10.18637/jss.v036.i11.

See Also

Boruta
randomForest
performance
rfPred
plotFeaturesMap
ptn_model
ptn_check_models
ptn_features
ptn_selectedFeatures
ptn_networkGraph

Examples

```
 tmp <- tempfile(fileext = ".pdf")

# load and create example data:
 data("postNetExample", package = "postNet")
 ptn <- postNetExample$ptn

# Prepare the list of pre-calculated features to be used in feature integration modelling:
 myFeatures <- postNetExample$features
 str(myFeatures)

# Run feature integration modelling using stepwise regression:
 ptn <- featureIntegration(ptn = ptn,
                           features = myFeatures,
                           pdfName = tmp,
                           regOnly = TRUE,
                           allFeat = FALSE,
                           analysis_type = "lm",
                           covarFilt = 20,
                           comparisons = list(c(1,2)),
                           NetModelSel = "omnibus")

# Run feature integration modelling using random forest classification:
 ptn <- featureIntegration(ptn = ptn,
                           features = myFeatures,
                           pdfName = tmp,
                           analysis_type ="rf",
                           comparisons = list(c(1,2)))
```

`foldingEnergyAnalysis` *Compare mRNA folding energy between gene sets*

Description

The `foldingEnergyAnalysis` function compares folding energies for each mRNA sequence region. Pre-calculated folding energies for reference sequences are available for certain RefSeq releases (for human and mouse). These values were calculated using the mfold algorithm. Alternatively, user-supplied values can be used in analyses. In addition, folding energies for gene sets of interest can be compared against background, or other gene sets. Several options are available for plotting the comparisons.

Usage

```
foldingEnergyAnalysis(ptn,
                      sourceFE = "load",
                      customFileFE = NULL,
                      residFE = FALSE,
                      region,
                      comparisons = NULL,
                      plotOut = TRUE,
                      plotType = "ecdf",
                      pdfName = NULL)
```

Arguments

`ptn`	A postNetData object.
`sourceFE`	A string specifying the source of the folding energy values. The options are `"load"`, or `"custom"`. If `"load"`, pre-calculated folding energies for the selected species ("human" or "mouse") will be used. If `"custom"`, a custom file with pre-calculated folding energies must be provided using the `customFileFE` argument. The default is `"load"`.
`customFileFE`	If `sourceFE = "custom"`, provide a string specifying the name and file path to the custom folding energy file. The custom file with calculated folding energies must be provided in a tab-delimited format with three columns: transcript ID, folding energy, and length of the sequence. Note that folding energies for different sequence regions (e.g. 5'UTR and 3'UTR) must be provided separately.
`residFE`	Logical indicating if the folding energies should be corrected for the length of the sequences. If `TRUE`, the values returned are the residuals from the linear model: FE ~ log2(sequence region length). See details below for usage recommendations. The default is `FALSE`.
`region`	A character vector specifying the sequence region(s) to be analyzed. This can be `"UTR5"`, `"UTR3"`, `"CDS"`, or any combination of the regions.
`comparisons`	A list of numeric vectors specifying pairwise comparisons between gene sets defined in the `regulation` or `geneList` slots of the `postNetData` object. Use 0 to denote the background set. For example, `list(c(0,2), c(1,2))` compares gene set 2 to the background and gene set 1 to gene set 2. The default is `NULL`.
`plotOut`	Logical indicating whether PDF files of plots are generated. If `FALSE`, only the folding energies for the specified sequence region(s) will be returned, and statistical comparisons between gene sets will not be performed. The default is `TRUE`.

plotType	If plotOut is TRUE, provide a string to indicate the type of output plot to generate. Options are "boxplot", "violin", or "ecdf". The default is "ecdf".
pdfName	Name to be appended to output PDF files. The default is NULL.

Details

Pre-calculated folding energies supplied with the package are currently available for human and mouse RefSeq releases "rel_109.20201120", and "rel_109.20200923". These values were calculated for all sequence regions using the mfold algorithm, available here: https://www.unafold.org/.

If the output folding energies will be used in downstream analysis with featureIntegration, consideration should be given to the residFE argument. If the length of the sequence regions will be included in the featureIntegration model, it is not recommended to correct the folding energies for the sequence length. However, if length will not be included in featureIntegration, the correction can be performed.

Value

The output will be a named list of vectors corresponding to each sequence region, with the folding energy for each gene. This list can be used as input for the downstream featureIntegration analysis. The foldingEnergyAnalysis function can also return PDF files of output plots with statistical comparisons between gene sets of interest, or against background.

References

If you use the foldingEnergyAnalysis function in your analysis with the sourceFE = "load", please cite:

M. Zuker. Mfold web server for nucleic acid folding and hybridization prediction. Nucleic Acids Res. 31 (13), 3406-3415, 2003. https://doi.org/10.1093/nar/gkg595

See Also

https://www.unafold.org/

Examples

```
 tmp <- tempfile(fileext = ".pdf")

# load example data:
 data("postNetExample", package = "postNet")
 ptn <- postNetExample$ptn

# Compare the folding energy of the 5'UTR for each gene between translationUp genes vs. background,
# translationDown genes vs. background, and translationUp vs. translationDown genes.
 FE <- foldingEnergyAnalysis(ptn = ptn,
                             region=c("UTR5"),
                             comparisons = list(c(0,1),c(0,2),c(1,2)),
                             residFE = FALSE,
                             plotType = "ecdf",
```

```
                                sourceFE = "load",
                                plotOut = TRUE,
                                pdfName = tmp)

  str(FE)
```

---

gageAnalysis *Perform GAGE analysis with a* postNetData *object*

Description

The gageAnalysis function implements the gage package to perform Generally Applicable Gene-set Enrichment (GAGE) analysis using the regulatory effect measurement contained in a postNetData object. These values are determined by the regulationGen or effectMeasure parameters in the postNetStart function.

Usage

```
gageAnalysis(ptn,
             category,
             genesSlopeFiltOut = NULL,
             maxSize = 500,
             minSize = 10)
```

Arguments

ptn	A postNetData object.
category	A character vector specifying one or more gene ontology categories to be included in the analysis. The options are "BP", "CC", "MF", and "KEGG". For example, category = c("BP", "MF").
genesSlopeFiltOut	If using an Anota2seqDataSet with analysis results for the "translation" or "buffering" regulatory modes, optionally provide a character vector with gene IDs to be excluded from GAGE analysis. This can be generated using the slopeFilt function. Although optional, this step is strongly recommended. This filtering is not necessary if using "mRNAAbundance", "totalmRNA", or "translatedmRNA". The default is NULL.
maxSize	Integer specifying the maximal size of a gene set to test. Terms/pathways with more genes will be excluded. The default is 500.
minSize	Integer specifying the minimal size of a gene set to test. Terms/pathways with fewer genes will be excluded. The default is 10.

Details

After running GAGE analysis, the results can be retrieved from the postNetData object using the S4 method ptn_GAGE.

Note that if using results from an **anota2seq** analysis, it is strongly recommended to first apply filtering using the slopeFilt function to exclude genes with unrealistic regression slopes.

Value

The results returned by the gageAnalysis function are compiled into an S4 object of class postNetGAGE and stored in the "GAGE" slot of the postNetData object. As described in the gage package, each gene ontology category has a slot in the results object that stores a named list with three elements ("greater", "less" and "stats"). The "stats" element contains the test statistics. The "greater" and "less" elements are data.frames with the results of a two-directional test, each having the following columns:

- **p.geomean**: The geometric mean of the individual p-values from multiple single array-based gene set tests.
- **stat.mean**: The mean of the individual statistics from multiple single array-based gene set tests. The value denotes the magnitude of the gene-set level changes, and the sign denotes the direction of the changes.
- **p.val**: The global p-value or summary of the individual p-values from multiple single array-based gene set tests.
- **q.val**: The Benjamini–Hochberg adjusted p-value, as implemented by the **multtest** package.
- **set.size**: The number of genes included in the gene set.
- **Genes**: The gene IDs in the input data belonging to the gene ontology gene set.

References

If you use the gageAnalysis function in your analysis, please cite:

Luo, W., Friedman, M., Shedden K., Hankenson K., and Woolf P. GAGE: Generally Applicable Gene Set Enrichment for Pathways Analysis. BMC Bioinformatics 2009, 10:161.

See Also

gage
ptn_GAGE
slopeFilt

Examples

```
# ------------------------------------------------------------------
# Example 1: Running GAGE with gene lists
# ------------------------------------------------------------------

# load example data:
 data("postNetExample", package = "postNet")
 ptn <- postNetExample$ptn

# Run GAGE:
 ptn <- gageAnalysis(ptn,
                     category = "CC")

# Extract the significant enrichment results from the postNetData object:
 gageOut <- ptn_GAGE(ptn = ptn,
```

```
                    category = "CC",
                    direction = "greater",
                    threshold = 1) # This will return the complete results.
                                   # For only significant results
                                   # the threshold should be lowered.
 str(gageOut)


# ------------------------------------------------------------------
# Example 2: Running GAGE requiring anota2seq analysis slope filtering
# ------------------------------------------------------------------

# Initialize Anota2seqDataSet (see anota2seq vignette for details)
 ads <- anota2seq::anota2seqDataSetFromMatrix(
    dataP = postNetExample$ads_data$dataP,
    dataT = postNetExample$ads_data$dataT,
    phenoVec = postNetExample$ads_data$phenoVec,
    batchVec = c(1, 2, 3, 4, 1, 2, 3, 4),
    dataType = "RNAseq",
    normalize = FALSE)

# Run an anota2seq analysis:
# Note that the quality control and residual outlier testing are not
# performed to limit the running time of this example. For full details
# on running an analysis please see the anota2seq vignette and help manual.
 ads <- anota2seq::anota2seqRun(ads,
    performQC = FALSE,
    performROT = FALSE,
    useProgBar = FALSE)

# Initialize the postNetData object, where "ads" is an Anota2seqDataSet object
# resulting from an anota2seqRun call:
  ptn <- postNetStart(
      ads = ads,
      regulation = c("translationUp","translationDown"),
      contrast = c(1,1),
      regulationGen = "translation",
      contrastSel = 1,
      selection = "random",
      setSeed = 123,
      source = "load",
      species = "human"
    )

# Run slope filtering to remove the genes with unrealistic slopes:
 filtOutGenes <- slopeFilt(ads = ads,
                           regulationGen = "translation",
                           contrastSel = 1)

# Run GAGE:
 ptn <- gageAnalysis(ptn,
                    genesSlopeFiltOut = filtOutGenes,
                    category = "CC")
```

`get_signatures` *Retrieve gene signatures provided in the postNet*

Description

The `get_signatures` function retrieves gene signatures of interest provided with the package for the indicated species.

Usage

```
get_signatures(species)
```

Arguments

`species`	A string to select the species of interest. Currently, `"human"` and `"mouse"` are supported.

Details

The data retrieved contains a list of gene signatures relevant to understanding possible pathways and mRNA features involved in regulation of mRNA translation, with a specific focus on mTOR and integrated stress response (ISR)-sensitive translation. These signatures have been compiled either directly from published studies, or from analyses of published datasets (sources and references described in detail below). Note that if the original source of the gene signature was from human, these gene IDs have been converted to mouse, and vice versa.

These gene signatures can be used with the plotSignatures, plotSignatures_ads, and signaturesHeatmap functions to perform statistical analyses and visualizations of how each gene signature is regulated in a given dataset. Furthermore, gene signatures can also be provided as features in the featureIntegration function which performs forward stepwise regression modelling or Random Forest classification to identify features that are most important in explaining post-transcriptional regulation in a given dataset. The signCalc function can be used to convert gene signatures into a format compatible with featureIntegration.

Value

The format is:
List of 5
$ Gandin_etal_2016_mTOR_transUp : chr [1:1487] "Zfy1" "Zfy2" "Isca2" "Rps14" ... $ Gandin_etal_2016_mTOR_transDo
: chr [1:1765] "Lpin1" "Galnt1" "Atrx" "Tbc1d31" ... $ Cockman_etal_2020_classicTOP : chr
[1:70] "Rpl37" "Rps14" "Hnrnpa1" "Rpl39" ... $ Guan_etal_2017_Tg1_transUp : chr [1:1336]
"Mcm7" "Bcl2" "Usp33" "Gm13212" ... $ Guan_etal_2017_Tg1_transDown : chr [1:774] "Mapre3"
"Mettl24" "Isg15" "Idh3g" ...

Source

Genes translationally regulated downstream of mTOR:

- **Signature Information**:
  - Signature Name(s): Gandin_etal_2016_mTOR_transUp, Gandin_etal_2016_mTOR_transDown
  - Source: MCF7 cells
  - Species: Human
  - Conditions: Insulin (4.2 nM) + torin1 (250 nM), Insulin (4.2 nM) + DMSO; 4 hrs
  - Replicates: 4 Reps/Condition
  - Comparison: Insulin + torin1 vs. Insulin
- **Publication Information**:
  - Title: mTORC1 and CK2 coordinate ternary and eIF4F complex assembly
  - Method: Polysome fractionation followed by microarray
  - Data Source: Anota2seq analysis of microarray data deposited in GSE76766.
  - Analysis Method: Anota2seq algorithm (version 1.14.0). The following thresholds were applied within the anota2seqRun function: maxPAdj = 0.15; deltaP = log2(1.2); deltaT = log2(1.2); deltaPT = log2(1.2); deltaTP = log2(1.2); maxSlopeTranslation = 2; minSlopeTranslation = -1; minSlopeBuffering = -2; maxSlopeBuffering = 1. Replicate was included in the anota2seq model using the 'batchVec' parameter to account for batch effects.
  - First Author, Last Author, Year: Gandin, Topisirovic, 2016
  - PMID: 27040916
  - GEO Accession: GSE76766
  - DOI: 10.1038/ncomms11127
- **Dataset Summary**: Genes classified as translationally activated (transUp) or suppressed (transDown) upon inhibition of mTOR via torin1 treatment in insulin-stimulated MCF7 cells.

mRNAs containing 5'terminal oligopyrimidine (5'TOP) motifs:

- **Signature Information**:
  - Signature Name: Cockman_etal_2020_classicTOP
  - Source: N/A
  - Species: Human
  - Conditions: N/A
  - Replicates: N/A
  - Comparison: N/A
- **Publication Information**:
  - Title: TOP mRNPs: Molecular Mechanisms and Principles of Regulation
  - Method: N/A
  - Data Source: Data obtained from supplementary materials (see: Appendix A Table 1A of publication)
  - Analysis Method:

  - First Author, Last Author, Year: Cockman, Ivanov, 2020
  - PMID: 32605040
  - GEO Accession: N/A
  - DOI: 10.3390/biom10070969
- **Dataset Summary**: A curated list of well-validated 5'TOP motif-containing mRNAs.

Genes translationally regulated downstream of ISR activation:

- **Signature Information**:
  - Signature Name(s): Guan_etal_2017_Tg1_transUp, Guan_etal_2017_Tg1_transDown
  - Source: Mouse embryonic fibroblasts (MEFs)
  - Species: Mouse
  - Conditions: Thapsigargin (400 nM), Vehicle control (DMSO); 1 hr
  - Replicates: 4 Reps/Condition
  - Comparison: Thapsigargin vs. DMSO
- **Publication Information**:
  - Title: A Unique ISR Program Determines Cellular Responses to Chronic Stress
  - Method: Polysome fractionation followed by RNA sequencing
  - Data Source:
  - Analysis Method:
  - First Author, Last Author, Year: Guan, Hatzoglou, 2017
  - PMID: 29220654
  - GEO Accession: GSE90070
  - DOI: 10.1016/j.molcel.2017.11.007
- **Dataset Summary**: Genes translationally activated (transUp) or suppressed (transDown) following activation of the ISR via thapsigargin treatment in MEFs.

References

If you use these gene signatures in your analysis, please cite:

Gandin V, Masvidal L, Cargnello M, Gyenis L, McLaughlan S, Cai Y, Tenkerian C, Morita M, Balanathan P, Jean-Jean O, Stambolic V, Trost M, Furic L, Larose L, Koromilas AE, Asano K, Litchfield D, Larsson O, Topisirovic I. mTORC1 and CK2 coordinate ternary and eIF4F complex assembly. Nat Commun. 2016 Apr 4;7:11127. doi: 10.1038/ncomms11127. PMID: 27040916; PMCID: PMC4822005.

Cockman E, Anderson P, Ivanov P. TOP mRNPs: Molecular Mechanisms and Principles of Regulation. Biomolecules. 2020 Jun 27;10(7):969. doi: 10.3390/biom10070969. PMID: 32605040; PMCID: PMC7407576.

Guan BJ, van Hoef V, Jobava R, Elroy-Stein O, Valasek LS, Cargnello M, Gao XH, Krokowski D, Merrick WC, Kimball SR, Komar AA, Koromilas AE, Wynshaw-Boris A, Topisirovic I, Larsson O, Hatzoglou M. A Unique ISR Program Determines Cellular Responses to Chronic Stress. Mol Cell. 2017 Dec 7;68(5):885-900.e6. doi: 10.1016/j.molcel.2017.11.007. PMID: 29220654; PMCID: PMC5730339.

See Also

plotSignatures
plotSignatures_ads
signaturesHeatmap
signCalc

Examples

```
# Load the human gene signatures:
 humanSignatures <- get_signatures("human")
 str(humanSignatures)
```

---

goAnalysis *Perform Gene Ontology Analysis using a* postNetData *object*

Description

The goAnalysis function implements clusterProfiler to perform Gene Ontology enrichment analysis using the gene sets of interest associated with the regulatory effect measurement defined within the postNetData object.

Usage

```
goAnalysis(ptn,
           genesSlopeFiltOut = NULL,
           category,
           maxSize = 500,
           minSize = 10,
           counts = 10,
           FDR = 0.15,
           name = NULL)
```

Arguments

ptn	A postNetData object.
genesSlopeFiltOut	If using an Anota2seqDataSet with analysis results for the "translation" or "buffering" regulatory modes, optionally provide a character vector with gene IDs to be excluded from enrichment analysis. This can be generated using the slopeFilt function. Although optional, this step is strongly recommended. This filtering is not necessary if using "mRNAAbundance", "totalmRNA", or "translatedmRNA". The default is NULL.
category	A character vector specifying one or more Gene Ontology categories to be included in the analysis. The options are 'BP', "CC", "MF", and "KEGG". For example, category = c("BP", "MF").

maxSize	Integer specifying the maximal size of a gene set to test. Terms/pathways with more genes will be excluded. The default is 500.
minSize	Integer specifying the minimal size of a gene set to test. Terms/pathways with fewer genes will be excluded. The default is 10.
counts	Integer specifying the minimal number of genes in the gene set of interest that must be included in the term. Terms with fewer genes overlapping with the gene set of interest than the threshold specified will be discarded, regardless of significance. The default is 10.
FDR	A numeric value providing the FDR threshold to be used to select significant enrichments. The default is 0.15.
name	Optionally, provide a name to be appended to the Excel output file storing the GO analysis results. Note that the "GO" slot of the postNetData object will reflect the results of the most recent analysis. However, multiple analyses can be run and saved by providing unique names to the output file. The default is NULL.

Details

After running GO term analysis, the results can be retrieved from the postNetData object using the S4 method ptn_GO function. Results can also be visualized using the goDotplot function.

Note that if using results from an **anota2seq** analysis, it is strongly recommended to first apply filtering using the slopeFilt function to exclude genes with unrealistic regression slopes.

Value

An S4 enrichResult-class object (DOSE package) is stored in the analysis slot of the postNetData object for each category of GO terms assessed. In addition, the "results" slot of each enrichResult object is written to an Excel file. Separate Excel files are created for each category.

References

See clusterProfiler for full details of enrichment analysis implementation in R.

If you use the goAnalysis function in your analysis, please cite:

S Xu, E Hu, Y Cai, Z Xie, X Luo, L Zhan, W Tang, Q Wang, B Liu, R Wang, W Xie, T Wu, L Xie, G Yu. Using clusterProfiler to characterize multiomics data. Nature Protocols. 2024, 19(11):3292-3320

See Also

clusterProfiler
postNetStart
slopeFilt
ptn_GO
goDotplot

Examples

```
 tmp <- tempfile(fileext = ".xlxs")
 tmp2 <- tempfile(fileext = ".pdf")

# ------------------------------------------------------------------
# Example 1: Running GO term analysis
# ------------------------------------------------------------------

# load example data:
 data("postNetExample", package = "postNet")
 ptn <- postNetExample$ptn

# Run GO term analysis:
 ptn <- goAnalysis(ptn = ptn,
                   category=c("BP"),
                   name = tmp)

# Extract the significant enrichment results from the postNetData object:
 goOut <- ptn_GO(ptn,
                 category = "BP",
                 geneList = "translationUp",
                 threshold = 0.05)
 str(goOut)

# Plot GO term analysis results:
 goDotplot(ptn = ptn,
           category = "BP",
           nCategories = 50,
           pool = TRUE,
           size = "geneRatio",
           pdfName = tmp2)



# ------------------------------------------------------------------
# Example 2: Running GO term analysis with anota2seq results
# ------------------------------------------------------------------

# Initialize Anota2seqDataSet (see anota2seq vignette for details):
 ads <- anota2seq::anota2seqDataSetFromMatrix(
    dataP = postNetExample$ads_data$dataP,
    dataT = postNetExample$ads_data$dataT,
    phenoVec = postNetExample$ads_data$phenoVec,
    batchVec = c(1, 2, 3, 4, 1, 2, 3, 4),
    dataType = "RNAseq",
    normalize = FALSE)

# Run an anota2seq analysis:
# Note that the quality control and residual outlier testing are not
# performed to limit the running time of this example. For full details
# on running an analysis please see the anota2seq vignette and help manual.
 ads <- anota2seq::anota2seqRun(ads,
```

```
    performQC = FALSE,
    performROT = FALSE,
    useProgBar = FALSE)

# Initialize the postNetData object, where "ads" is an Anota2seqDataSet object
# resulting from an anota2seqRun call:
  ptn <- postNetStart(
      ads = ads,
      regulation = c("translationUp","translationDown"),
      contrast = c(1,1),
      regulationGen = "translation",
      contrastSel = 1,
      selection = "random",
      setSeed = 123,
      source = "load",
      species = "human"
    )

# Run slope filtering to remove the genes with unrealistic slopes:
 filtOutGenes <- slopeFilt(ads = ads,
                           regulationGen = "translation",
                           contrastSel = 1)


# Run GO term analysis:
 ptn <- goAnalysis(ptn = ptn,
                   genesSlopeFiltOut = filtOutGenes,
                   category = c("BP", "KEGG"),
                   name = "example")
```

---

goDotplot *Produce dot plots with results of GO term enrichment analysis*

Description

The goDotplot function can be used to produce dot plots visualizing the results of GO term enrichment analyses performed using goAnalysis. The input is a postNetData object, with several options for plotting.

Usage

```
goDotplot(ptn,
          category,
          pool = TRUE,
          termSel = NULL,
          nCategories = 10,
          size = "Count",
          pdfName = NULL)
```

Arguments

ptn	A postNetData object.
category	A character vector specifying one or more Gene Ontology categories to plot results for. The options are "BP", "CC", "MF", and "KEGG". For example, category = c("BP", "MF")
pool	Logical indicating if results for all gene sets of interest should be pooled into a single plot. For example, if TRUE, results for "translationUp" and "translationDown" will be displayed together, automatically using the colours stored in the postNetData object. If FALSE, a separate plot will be generated for each gene set of interest. Default is TRUE.
termSel	A character vector can be supplied to specify specific terms to be plotted. For example, c("GO:0008380", "GO:0090069"), or c("hsa03013") if using KEGG terms. Default is NULL.
nCategories	An integer specifying the maximum number of terms to include on each plot. Default is 10.
size	A string specifying the metric used to determine the size of the dots. The options are "Count", which is the number of genes from the gene set of interest included in each specific term, or "geneRatio" which is the "Count" divided by the total number of genes in the term. Default is "Count".
pdfName	Name to be appended to output PDF files. Default is NULL.

Details

To produce dot plots, the goAnalysis function must first be run on the postNetData object.

Value

No value is returned. Graphical outputs are generated as PDF files.

See Also

goAnalysis
ptn_GO

Examples

```
# load example data:
 data("postNetExample", package = "postNet")
 ptn <- postNetExample$ptn

# Run GO term analysis:
 ptn <- goAnalysis(ptn = ptn,
                   category = c("BP"),
                   name = "example")

# Plot GO term analysis results for both gene sets of interest on one plot:
```

```
goDotplot(ptn=ptn,
          category = "BP",
          nCategories = 20,
          pool = TRUE,
          size = "Count",
          pdfName = "example")
```

`gseaAnalysis` *Perform Gene Set Enrichment Analysis (GSEA) with a* `postNetData` *object*

Description

The `gseaAnalysis` function implements the fgsea package to perform Gene Set Enrichment Analysis using the gene sets associated with the regulatory effect measurement contained in a `postNetData` object. These gene sets are determined by the `regulationGen` or `effectMeasure` parameters in the postNetStart function.

Usage

```
gseaAnalysis(ptn,
             genesSlopeFiltOut = NULL,
             collection = NULL,
             subcollection = NULL,
             subsetNames = NULL,
             geneSet = NULL,
             maxSize = 500,
             minSize = 10,
             name = NULL)
```

Arguments

`ptn` A postNetData object.

`genesSlopeFiltOut`
If using an `Anota2seqDataSet` with analysis results for the "translation" or "buffering" regulatory modes, optionally provide a character vector with gene IDs to be excluded from GSEA. This can be generated using the slopeFilt function. Although optional, this step is strongly recommended. This filtering is not necessary if using "mRNAAbundance", "totalmRNA", or "translatedmRNA". Default is `NULL`.

`collection` A character vector selecting the MSigDB collection of interest, for example: `'c8'`, or `'h'`. Although possible, it is not recommended to run all collections in the same analysis. Available options for collections can be found at https://www.gsea-msigdb.org/gsea/msigdb/index.jsp, or by using the **msigdb** package (see example below). Currently, collections are only available for human and mouse.

`subcollection` Optionally, provide a character vector selecting the MSigDB subcollection of interest. For example: `"GO:BP"`, or `"CP:REACTOME"`. Available options for subcollections can be found at https://www.gsea-msigdb.org/gsea/msigdb/index.jsp, or by using the **msigdb** package (see example below). Default is `NULL`.

`subsetNames` Optionally, provide a character vector specifying specific terms to use for GSEA by providing the names of the pathways. For example: `c("GOBP_RIBOSOME_BIOGENESIS","GOBP_CAP_DE` Default is `NULL`.

`geneSet` Optionally, instead of using a GSEA collection, provide named list of character vectors with custom gene signatures to be used in GSEA. Default is `NULL`.

`maxSize` Integer specifying the maximal size of a gene set to test. Terms/pathways with more genes will be excluded. Default is `500`.

`minSize` Integer specifying the minimal size of a gene set to test. Terms/pathways with fewer genes will be excluded. Default is `10`.

`name` Optionally, provide a name for the tab-delimited output file storing the results of the GSEA. Note that the "GSEA" slot of the `postNetData` object will reflect the results of the most recent analysis. However, multiple analyses can be run and saved by providing unique names to the output file. Default is `NULL`.

Details

After running GSEA, the results of the analysis can be retrieved from the `postNetData` object using the S4 method ptn_GSEA function. These results can then be provided as input to the gseaPlot function, which can be used to generate GSEA plots for all results, or for specific terms of interest.

Note that if the results of an **anota2seq** analysis are being used for GSEA, it is strongly recommended to first perform filtering using the slopeFilt function.

Value

A data.frame of GSEA results that is stored in the GSEA slot of the `postNetData` object, and written to a tab-delimited text file. Each row of the data.frame corresponds to a term, with the following columns (also see fgseaMultilevel from the **fgsea** package):

- **Term**: The name of the term or pathway tested.
- **ES**: The enrichment score.
- **NES**: The normalized enrichment score, adjusted to the mean enrichment of randomly sampled size-matched sets.
- **log2err**: The expected error for the standard deviation of the P-value logarithm.
- **count**: The number of genes in the set belonging to the term or pathway.
- **size**: The size of the pathway after removing genes not present in the input.
- **pvalue**: The enrichment p-value.
- **adjusted_pvalue**: The BH-adjusted p-value.
- **Genes**: The gene IDs in the input data belonging to the term/pathway.

References

See fgsea for full details of GSEA implementation in R.

See https://www.gsea-msigdb.org/gsea/index.jsp for GSEA documentation.

If you use the gseaAnalysis function in your analysis, please cite:

A. Subramanian, P. Tamayo, V.K. Mootha, S. Mukherjee, B.L. Ebert, M.A. Gillette, A. Paulovich, S.L. Pomeroy, T.R. Golub, E.S. Lander, & J.P. Mesirov. Gene set enrichment analysis: A knowledge-based approach for interpreting genome-wide expression profiles, Proc. Natl. Acad. Sci. U.S.A. 102 (43) 15545-15550, (2005).

Mootha, V., Lindgren, C., Eriksson, KF. et al. PGC-1alpha-responsive genes involved in oxidative phosphorylation are coordinately downregulated in human diabetes. Nat Genet 34, 267–273 (2003).

Korotkevich G, Sukhov V, Sergushichev A (2019). "Fast gene set enrichment analysis." bioRxiv. doi:10.1101/060012, http://biorxiv.org/content/early/2016/06/20/060012.

See Also

GSEABase
msigdb
fgsea
postNetStart
slopeFilt
ptn_GSEA
gseaPlot

Examples

```
 tmp <- tempfile(fileext = ".txt")
 tmp2 <- tempfile(fileext = ".pdf")

# ------------------------------------------------------------------
# Example 1: Running GSEA with a custom gene set
# ------------------------------------------------------------------

# load example data:
 data("postNetExample", package = "postNet")
 ptn <- postNetExample$ptn

# Create example custom gene sets for GSEA:
 inSet <- list(set=sample(unlist(postNetExample$geneList[[1]]), 10))

# Run GSEA on custom gene sets:
 ptn <- gseaAnalysis(ptn = ptn,
                     geneSet = inSet,
                     name = tmp)
```

```
# Extract the significant enrichment results from the postNetData object:
 gseaOut <- ptn_GSEA(ptn,
                     threshold = 0.05)
 str(gseaOut)

# Plot GSEA results:
 gseaPlot(ptn = ptn,
          termNames = gseaOut$Term[1],
          pdfName = tmp2)



# ------------------------------------------------------------------
# Example 2: Running GSEA with MSigDB collections
# ------------------------------------------------------------------

# If using a GSEA collection from MSigDB, check the available versions:
 version <- msigdb::getMsigdbVersions()
 str(version)

# Retrieve the MSigDB data for "human":
 msigdbOut <- msigdb::getMsigdb(org = "hs", id = "SYM", version = version)

# Check the available collections or subcollections:
 msigdb::listCollections(msigdbOut)
 msigdb::listSubCollections(msigdbOut)

# Initialize a postNetData object:
 data("postNetExample", package = "postNet")
 ptn <- postNetExample$ptn

# Run GSEA on the C5 collection with GO:BP and specific terms:
 ptn <- gseaAnalysis(ptn = ptn,
                     collection = "c5",
                     subcollection = "GO:BP",
                     subsetNames = c("GOBP_CELL-CELL_SIGNALING_BY_WNT",
                     "GOBP_ENDOCYTOSIS"),
                     name = "c5_Example")

# ------------------------------------------------------------------
# Example 3: Running GSEA requiring anota2seq analysis slope filtering
# ------------------------------------------------------------------

# Initialize Anota2seqDataSet (see anota2seq vignette for details)
 ads <- anota2seq::anota2seqDataSetFromMatrix(
    dataP = postNetExample$ads_data$dataP,
    dataT = postNetExample$ads_data$dataT,
    phenoVec = postNetExample$ads_data$phenoVec,
    batchVec = c(1, 2, 3, 4, 1, 2, 3, 4),
    dataType = "RNAseq",
    normalize = FALSE)
```

```
# Run an anota2seq analysis:
# Note that the quality control and residual outlier testing are not
# performed to limit the running time of this example. For full details
# on running an analysis please see the anota2seq vignette and help manual.
 ads <- anota2seq::anota2seqRun(ads,
    performQC = FALSE,
    performROT = FALSE,
    useProgBar = FALSE)

# Initialize the postNetData object, where "ads" is an Anota2seqDataSet object
# resulting from an anota2seqRun call:
  ptn <- postNetStart(
      ads = ads,
      regulation = c("translationUp","translationDown"),
      contrast = c(1,1),
      regulationGen = "translation",
      contrastSel = 1,
      selection = "random",
      setSeed = 123,
      source = "load",
      species = "human"
    )

# Run slope filtering to remove the genes with unrealistic slopes:
 filtOutGenes <- slopeFilt(ads = ads,
                           regulationGen = "translation",
                           contrastSel = 1)

# Create example custom gene sets for GSEA:
 set1 <- sample(row.names(postNetExample$ads_data$dataP), 10)
 inSet <- list(Set1 = set1)

# Run GSEA on the C5 collection:
 ptn <- gseaAnalysis(ptn = ptn,
                     genesSlopeFiltOut = filtOutGenes,
                     geneSet = inSet,
                     name = "ads_example")
```

---

gseaPlot *Plot enriched terms from GSEA*

Description

The gseaPlot function generates visualizations of the results of GSEA. The rank is plotted on the x-axis against the enrichment score on the y-axis. The positions of genes belonging to the term or pathway of interest are indicated in the ranking by red bars.

Usage

```
gseaPlot(ptn,
        termNames,
        genesSlopeFiltOut = NULL,
        gseaParam = 1,
        ticksSize = 0.3,
        pdfName = NULL)
```

Arguments

ptn	A postNetData object.
termNames	Character vector specifying the names of the terms to be plotted.
genesSlopeFiltOut	If using an Anota2seqDataSet with analysis results for the "translation" or "buffering" regulatory modes, optionally provide a character vector with gene IDs to be excluded from GSEA that is the output of the slopeFilt function. Although optional, this step is strongly recommended. This filtering is not necessary if using "mRNAAbundance", "totalmRNA", or "translatedmRNA". Default is NULL.
gseaParam	The GSEA parameter used to adjust the displayed statistic values. Can be either 0 or 1. Default is 1. See fgseaMultilevel from the **fgsea** package for details.
ticksSize	Number to adjust the thickness of the lines indicating the position of genes in the ranking. Default is 0.3.
pdfName	Optionally, provide a name for the output PDF file.

Value

No value is returned. A graphical output is generated as a PDF file.

References

See fgsea for full details of GSEA implementation in R.

See https://www.gsea-msigdb.org/gsea/index.jsp for GSEA documentation.

A. Subramanian, P. Tamayo, V.K. Mootha, S. Mukherjee, B.L. Ebert, M.A. Gillette, A. Paulovich, S.L. Pomeroy, T.R. Golub, E.S. Lander, & J.P. Mesirov. Gene set enrichment analysis: A knowledge-based approach for interpreting genome-wide expression profiles, Proc. Natl. Acad. Sci. U.S.A. 102 (43) 15545-15550, (2005).

Mootha, V., Lindgren, C., Eriksson, KF. et al. PGC-1alpha-responsive genes involved in oxidative phosphorylation are coordinately downregulated in human diabetes. Nat Genet 34, 267–273 (2003).

See Also

fgsea
gseaAnalysis
ptn_GSEA

Examples

```
 tmp <- tempfile(fileext = ".txt")
 tmp2 <- tempfile(fileext = ".pdf")

# load example data:
 data("postNetExample", package = "postNet")
 ptn <- postNetExample$ptn

# Create example custom gene sets for GSEA:
 inSet <- list(set = sample(unlist(postNetExample$geneList[[1]]), 10))

# Run GSEA on custom gene sets:
 ptn <- gseaAnalysis(ptn = ptn,
                     geneSet = inSet,
                     name = tmp)

# Extract the significant enrichment results from the postNetData object:
 gseaOut <- ptn_GSEA(ptn, threshold = 0.05)

# Plot GSEA results:
 gseaPlot(ptn = ptn,
          termNames = gseaOut$Term[1],
          pdfName = tmp)
```

---

humanSignatures *Human gene signatures of translational control*

Description

The humanSignatures dataset is a list of example gene signatures derived from published studies profiling mRNA translation downstream of mTOR signalling and activation of the integrated stress response (ISR). In addition, an example of a signature comprised of mRNAs containing 5'TOP motifs is also included. The list is not intended to be exhaustive, but provides examples of how gene signatures can be used in a **postNet** analysis to gain insights into post-transcriptional regulation of gene expression.

Usage

```
data("humanSignatures")
```

Format

The format is: List of 5 $ Gandin_etal_2016_mTOR_transUp : chr [1:1569] "ZNF467" "PHLDA2" "PYCARD" "FHOD1" ... $ Gandin_etal_2016_mTOR_transDown : chr [1:1721] "DST" "OTULINL" "MME" "BRD8" ... $ Cockman_etal_2020_classicTOP : chr [1:92] "RPSA" "RPS2" "RPS3" "RPS3A" ... $ Guan_etal_2017_Tg1_transUp : chr [1:1192] "AR" "ATP7A" "BRCA2" "LYST" ... $ Guan_etal_2017_Tg1_transDown : chr [1:731] "AGA" "BCKDHB" "CST3" "CYBA" ...

Details

The `humanSignatures` dataset contains a list of gene signatures relevent to understanding possible pathways and mRNA features involved in regulation of mRNA translation, with a specific focus on mTOR and integrated stress response (ISR)-sensitive translation. These signatures have been compiled either directly from published studies, or from analyses of published datasets (sources and references described in detail below). Note that if the orignal source of the gene signature was from mouse, these gene IDs have been converted to human.

These gene signatures can be used with the plotSignatures, plotSignatures_ads, and signaturesHeatmap functions to perform statistical analyses and visualizations of how each gene signature is regulated in a given dataset. Furthermore, gene signatures can also be provided as features in the featureIntegration function which performs step-wise regression modelling or random forest classification to identify features that are most important in explaining post-transcriptional regulation in a given dataset.

Source

Genes translationally regulated downstream of mTOR:

- **Signature Information**:
  - Signature Name(s): Gandin_etal_2016_mTOR_transUp, Gandin_etal_2016_mTOR_transDown
  - Source: MCF7 cells
  - Species: Human
  - Conditions: Insulin (4.2 nM) + torin1 (250 nM), Insulin (4.2 nM) + DMSO; 4 hrs
  - Replicates: 4 Reps/Condition
  - Comparison: Insulin + torin1 vs. Insulin
- **Publication Information**:
  - Title: mTORC1 and CK2 coordinate ternary and eIF4F complex assembly
  - Method: Polysome fractionation followed by microarray
  - Data Source: Anota2seq analysis of microarray data deposited in GSE76766.
  - Analysis Method: Anota2seq algorithm (version 1.14.0). The following thresholds were applied within the anota2seqRun function: maxPAdj = 0.15; deltaP = log2(1.2); deltaT = log2(1.2); deltaPT = log2(1.2); deltaTP = log2(1.2); maxSlopeTranslation = 2; minSlopeTranslation = -1; minSlopeBuffering = -2; maxSlopeBuffering = 1. Replicate was included in the anota2seq model using the 'batchVec' parameter to account for batch effects.
  - First Author, Last Author, Year: Gandin, Topisirovic, 2016
  - PMID: 27040916

  - GEO Accession: GSE76766
  - DOI: 10.1038/ncomms11127
- **Dataset Summary**: Genes classified as translationally activated (transUp) or suppressed (transDown) upon inhibition of mTOR via torin1 treatment in insulin-stimulated MCF7 cells.

mRNAs containing 5'terminal oligopyrimidine (5'TOP) motifs:

- **Signature Information**:
  - Signature Name: Cockman_etal_2020_classicTOP
  - Source: N/A
  - Species: Human
  - Conditions: N/A
  - Replicates: N/A
  - Comparison: N/A
- **Publication Information**:
  - Title: TOP mRNPs: Molecular Mechanisms and Principles of Regulation
  - Method: N/A
  - Data Source: Data obtained from supplementary materials (see: Appendix A Table 1A of publication)
  - Analysis Method:
  - First Author, Last Author, Year: Cockman, Ivanov, 2020
  - PMID: 32605040
  - GEO Accession: N/A
  - DOI: 10.3390/biom10070969
- **Dataset Summary**: A curated list of well-validated 5'TOP motif-containing mRNAs.

Genes translationally regulated downstream of ISR activation:

- **Signature Information**:
  - Signature Name(s): Guan_etal_2017_Tg1_transUp, Guan_etal_2017_Tg1_transDown
  - Source: Mouse embryonic fibroblasts (MEFs)
  - Species: Mouse
  - Conditions: Thapsigargin (400 nM), Vehicle control (DMSO); 1 hr
  - Replicates: 4 Reps/Condition
  - Comparison: Thapsigargin vs. DMSO
- **Publication Information**:
  - Title: A Unique ISR Program Determines Cellular Responses to Chronic Stress
  - Method: Polysome fractionation followed by RNA seqeuncing
  - Data Source:
  - Analysis Method:
  - First Author, Last Author, Year: Guan, Hatzoglou, 2017
  - PMID: 29220654

  - GEO Accession: GSE90070
  - DOI: 10.1016/j.molcel.2017.11.007
- **Dataset Summary**: Genes translationally activated (transUp) or suppressed (transDown) following activation of the ISR via thapsigargin treatment in MEFs.

References

If you use these gene signatures in your analysis, please cite:

Gandin V, Masvidal L, Cargnello M, Gyenis L, McLaughlan S, Cai Y, Tenkerian C, Morita M, Balanathan P, Jean-Jean O, Stambolic V, Trost M, Furic L, Larose L, Koromilas AE, Asano K, Litchfield D, Larsson O, Topisirovic I. mTORC1 and CK2 coordinate ternary and eIF4F complex assembly. Nat Commun. 2016 Apr 4;7:11127. doi: 10.1038/ncomms11127. PMID: 27040916; PMCID: PMC4822005.

Cockman E, Anderson P, Ivanov P. TOP mRNPs: Molecular Mechanisms and Principles of Regulation. Biomolecules. 2020 Jun 27;10(7):969. doi: 10.3390/biom10070969. PMID: 32605040; PMCID: PMC7407576.

Guan BJ, van Hoef V, Jobava R, Elroy-Stein O, Valasek LS, Cargnello M, Gao XH, Krokowski D, Merrick WC, Kimball SR, Komar AA, Koromilas AE, Wynshaw-Boris A, Topisirovic I, Larsson O, Hatzoglou M. A Unique ISR Program Determines Cellular Responses to Chronic Stress. Mol Cell. 2017 Dec 7;68(5):885-900.e6. doi: 10.1016/j.molcel.2017.11.007. PMID: 29220654; PMCID: PMC5730339.

Examples

```
# Load the mouse signatures
 humanSignatures <- get_signatures("human")
 str(humanSignatures)
```

`lengthAnalysis` *Calculate the length of mRNA sequence regions and compare between gene sets*

Description

The `lengthAnalysis` function calculates the log2-transformed length of different regions of mRNA sequences (5'UTR, 3'UTR, and CDS) for each gene. In addition, the length of different sequence regions for gene sets of interest can be compared against background, or other gene sets. Several options are available for plotting the comparisons.

Usage

```
lengthAnalysis(ptn,
               region,
               comparisons = NULL,
               plotOut = TRUE,
               plotType = "boxplot",
               pdfName = NULL)
```

Arguments

ptn	A postNetData object.
region	A character vector specifying the sequence region(s) to be analyzed. This can be "UTR5", "UTR3", "CDS", or any combination of the regions.
comparisons	A list of numeric vectors specifying pairwise comparisons between gene sets defined in the regulation or geneList slots of the postNetData object. Use 0 to denote the background set. For example, list(c(0,2), c(1,2)) compares gene set 2 to the background and gene set 1 to gene set 2. The default is NULL.
plotOut	Logical indicating whether PDF files of plots are generated. If FALSE, only the log2-transformed lengths for the sequence regions of interest will be returned, and statistical comparisons between gene sets will not be performed. The default is TRUE.
plotType	If plotOut is TRUE, indicate the type of output plot to generate. Options are "boxplot", "violin", or "ecdf". The default is "boxplot".
pdfName	Name to be appended to output PDF files.

Details

A two-sided Wilcoxon Rank Sum test is performed to identify significant differences in mRNA sequence region length between gene sets of interest, or against the background gene set.

Value

A named list of vectors for selected mRNA sequence regions with the calculated log2-transformed lengths for each gene. The elements in each vector are named with the gene ID. The lengthAnalysis function can also return PDF files of output plots with statistical comparisons between gene sets of interest.

Examples

```
 tmp <- tempfile(fileext = ".pdf")

# load example data:
 data("postNetExample", package = "postNet")
 ptn <- postNetExample$ptn

# Calculate the length of the 5'UTR, 3'UTR and coding sequence for each gene,
# and compare lengths between translationUp genes vs. background,
# translationDown genes vs. background,
# and translationUp vs. translationDown genes:
 len <- lengthAnalysis(ptn = ptn,
                       region = c("UTR5","CDS","UTR3"),
                       comparisons = list(c(0,1),c(0,2),c(1,2)),
                       plotOut = TRUE,
                       plotType = "boxplot",
                       pdfName = tmp)

 str(len)
```

`miRNAanalysis` *Perform miRNA target enrichment analysis using a* `postNetData` *object*

Description

The `miRNAanalysis` function employs gage in combination with lists of mRNA targets of miRNA from the TargetScan database to identify enrichments in miRNA targets within the gene sets of interest contained in a `postNetData` object. This function does not identify or predict miRNA binding sites, but rather assesses whether gene sets of interest may be regulated by particular miRNAs.

Usage

```
miRNAanalysis(ptn,
              miRNATargetScanFile,
              genesSlopeFiltOut = NULL,
              contextScore = -0.2,
              Pct = 0,
              maxSize = 500,
              minSize = 10)
```

Arguments

`ptn`	A postNetData object.
`miRNATargetScanFile`	The name and path of a TAB delimited text file with miRNA target predictions. The target prediction file must contain the headings 'Cumulative weighted context++ score', 'Aggregate PCT', 'Gene Symbol', and 'Representative miRNA', and can be downloaded from the TargetScan database `https://www.targetscan.org`. Importantly, the file must be subset to contain predictions for only the desired species prior to running the analysis.
`genesSlopeFiltOut`	If using an `Anota2seqDataSet` with analysis results for the "translation" or "buffering" regulatory modes, optionally provide a character vector with gene IDs to be excluded from miRNA target enrichment analysis. This can be generated using the slopeFilt function. Although optional, this step is strongly recommended. This filtering is not necessary if using "mRNAAbundance", "totalmRNA", or "translatedmRNA". Default is `NULL`.
`contextScore`	A negative numeric value specifying the threshold of the cumulative weighted context++ score on which to filter miRNA target predictions. Larger negative values of `contextScore` indicate stronger biochemical response to a given miRNA (more repression). See details below. Default is `-0.2`.
`Pct`	A numeric value between 0 and 1 specifying the threshold of the aggregate Pct score on which to filter miRNA target predictions. Larger values of `Pct` (closer

to 1) indicate a greater probability that sites for a given gene have been evolutionarily conserved due to maintenance of miRNA targeting (biological function). Pct and cumulative weighted context++ score represent independent alternative scoring schemes. Therefore, filtering can be performed on both metrics, or according to one metric at a time. See details below. Default is `0`.

`maxSize` Integer specifying the maximal size of a gene set to test. miRNA with more predicted target genes will be excluded. Default is `500`.

`minSize` Integer specifying the minimal size of a gene set to test. miRNAs with fewer predicted target genes will be excluded. Default is `10`.

Details

For each miRNA-mRNA targeting prediction, two metrics are obtained from the TargetScan database; the cumulative weighted context++ score (CWCS) (see https://www.targetscan.org/vert_71/docs/context_score_totals.html for details) and the Pct score (see https://www.targetscan.org/docs/pct.html for details). In order to limit the analysis to high-confidence targeting predictions, filtering can be performed using one, or both of these metrics. Subsequently, gage is applied to identify enrichments in miRNAs predicted to target gene sets of interest based on ranked genes (ranked by the regulatory effect measurement).

Depending on the biological question, it is important to consider which thresholds to use to filter target predictions. Context scores provide a prediction of targeting efficiency, whereas Pct is an estimate of the probability that the miRNA-mRNA targeting interaction is evolutionarily conserved suggesting important biological function. See https://www.targetscan.org/faqs.html for further details regarding target prediction.

Note that enrichments in miRNA predicted to target genes that are upregulated (e.g., if the regulatory effect measurement is log2 fold change) that appear in the results table labelled "greater" can be interpreted as those miRNA that may be downregulated or otherwise not active in the experimental condition. Likewise, enrichments in miRNA predicted to target genes that are downregulated that appear in the results table labelled "less" can be interpreted as those miRNA that may be upregulated or active in the experimental condition.

After running GAGE miRNA target enrichment analysis, the results can be retrieved from the `postNetData` object using the S4 method ptn_miRNA_analysis function. To include specific miRNA in downstream featureIntegration analysis, lists of genes targeted by particular miRNA can be retrieved using the S4 method ptn_miRNA_to_gene function. These gene lists can then be included as signatures in the models.

Note that if the results of an **anota2seq** analysis are being used for GAGE miRNA target enrichment analysis, it is strongly recommended to first perform filtering using the slopeFilt function.

Value

The results returned by the `miRNAanalysis` function are compiled into an S4 object of class `postNetmiRNA` and stored in the "miRNA" slot of the `postNetData` object. The "miRNA_analysis" slot stores a named list with three elements ("greater", "less" and "stats"). As described in the gage package, the "stats" element contains the test statistics. The "greater" and "less" elements are data.frames with the results of a two-directional test, each having the following columns:

- **p.geomean**: The geometric mean of the individual p-values from multiple single array-based gene set tests.

- **stat.mean**: The mean of the individual statistics from multiple single array-based gene set tests. The value denotes the magnitude of the gene-set level changes, and the sign denotes the direction of the changes.
- **p.val**: The global p-value or summary of the individual p-values from multiple single array-based gene set tests.
- **q.val**: The BH adjusted p-value, as implemented by the **multtest** package.
- **set.size**: The number of genes included in the gene set.

The contents of the "miRNA_analysis" slot can be accessed using the ptn_miRNA_analysis S4 method.
The "miRNA_to_gene" slot stores a list where each element is a vector of genes that are predicted to be targeted by the miRNA that passed the filtering thresholds for `contextScore` and/or `Pct`.
The contents of the "miRNA_to_gene" slot can be accessed using the ptn_miRNA_to_gene S4 method.

References

If you use the `miRNAanalysis` function in your analysis, please cite:

McGeary SE, Lin KS, Shi CY, Pham T, Bisaria N, Kelley GM, Bartel DP. The biochemical basis of microRNA targeting efficacy. Science Dec 5, (2019).

Agarwal V, Bell GW, Nam J, Bartel DP. Predicting effective microRNA target sites in mammalian mRNAs. eLife, 4:e05005, (2015). eLife Lens view.

Luo, W., Friedman, M., Shedden K., Hankenson, K. and Woolf, P GAGE: Generally Applicable Gene Set Enrichment for Pathways Analysis. BMC Bioinformatics 2009, 10:161.

See Also

gage
ptn_miRNA_to_gene
ptn_miRNA_analysis

Examples

```
## Not run:
# ------------------------------------------------------------------
# Example 1: Running GAGE miRNA target enrichment analysis with gene lists
# ------------------------------------------------------------------

# load example data:
 data("postNetExample", package = "postNet")
 ptn <- postNetExample$ptn

# Run GAGE miRNA target enrichment analysis:
 ptn <- miRNAanalysis(ptn = ptn,
                      miRNATargetScanFile = 'miRNA_predictions_hsa.txt',
                      contextScore = -0.2,
                      Pct = 0.4)
```

```
# Extract the significant enrichment results from the postNetData object:
 miRNAout <- ptn_miRNA_analysis(ptn = ptn,
                   direction = "less",
                   threshold = 1)
 str(miRNAout)

# ------------------------------------------------------------------
# Example 2: Running analysis requiring anota2seq analysis slope filtering
# ------------------------------------------------------------------

# Initialize Anota2seqDataSet (see anota2seq vignette for details)
 ads <- anota2seq::anota2seqDataSetFromMatrix(
    dataP = postNetExample$ads_data$dataP,
    dataT = postNetExample$ads_data$dataT,
    phenoVec = postNetExample$ads_data$phenoVec,
    batchVec = c(1, 2, 3, 4, 1, 2, 3, 4),
    dataType = "RNAseq",
    normalize = FALSE)

# Run an anota2seq analysis:
# Note that the quality control and residual outlier testing are not
# performed to limit the running time of this example. For full details
# on running an analysis please see the anota2seq vignette and help manual.
 ads <- anota2seq::anota2seqRun(ads,
    performQC = FALSE,
    performROT = FALSE,
    useProgBar = FALSE)

# Initialize the postNetData object, where "ads" is an Anota2seqDataSet object
# resulting from an anota2seqRun call:
  ptn <- postNetStart(
      ads = ads,
      regulation = c("translationUp","translationDown"),
      contrast = c(1,1),
      regulationGen = "translation",
      contrastSel = 1,
      selection = "random",
      setSeed = 123,
      source = "load",
      species = "human"
    )

# Run slope filtering to remove the genes with unrealistic slopes:
 filtOutGenes <- slopeFilt(ads = ads,
                           regulationGen = "translation",
                           contrastSel = 1)

# Run GAGE miRNA target enrichment analysis:
 ptn <- miRNAanalysis(ptn,
                     genesSlopeFiltOut = filtOutGenes,
                     miRNATargetScanFile = "miRNA_predictions_hsa.txt",
                     Pct = 0.7)
```

```
## End(Not run)
```

`motifAnalysis` *De novo motif discovery using STREME and a* `postNetData` *object*

Description

The `motifAnalysis` function applies the STREME method from the MEME-Suite to discover de novo ungapped motifs (recurring, fixed-length patterns) that are enriched in the selected sequence region(s) of gene sets of interest relative to control sequences (background).

Usage

```
motifAnalysis(ptn,
              stremeThreshold = 0.05,
              minwidth = 6,
              memePath = NULL,
              seqType = "dna",
              region,
              subregion = NULL,
              subregionSel = NULL)
```

Arguments

`ptn`	a postNetData object.
`stremeThreshold`	A numeric value specifying the enrichment p-value threshold for motif selection. The default is 0.05.
`minwidth`	A numeric value specifying the minimal width (in nucleotides) of the motif. The default is 6.
`memePath`	The path to the STREME executables in "meme/bin". Note that the MEME-Suite must be installed.
`seqType`	A string specifying the type of sequence being provided to search for motifs. The options are, `"dna"`, `"rna"`, or `"protein"`. The default is `"dna"`.
`region`	A character vector specifying the sequence region(s) to be analyzed. This can be `"UTR5"`, `"UTR3"`, `"CDS"`, or any combination of the regions.
`subregion`	Optionally, it is possible to specify a more specific subregion of the sequences to either select or exclude from the analysis. This is done by providing a numeric value indicating the number of nucleotides from either the start of the sequence region if positive, or the end if negative. For example, `subregion = 50`, would denote the first 50 nucleotides of the sequence region(s) specified by `region`, while `subregion = -50`, would denote the last 50 nucleotides of the sequence region(s). The default is `NULL`.
`subregionSel`	If a value is provided for `subregion`, indicate whether the subregion should be selected (limiting the analysis to this region) or excluded by specifying either `"select"` or `"exclude"`. The default is `NULL`.

Details

The `motifAnalysis` function applies the runStreme function from the memes package. However, the MEME Suite of software tools must also be installed locally, and the path to the executables must be provided. The MEME Suite can be downloaded here: https://meme-suite.org/meme/doc/download.html. Installation instructions can be found here: https://meme-suite.org/meme/doc/install.html?man_type=web.

Motifs identified using `motifAnalysis` can be quantified and included in downstream featureIntegration using the contentMotifs function. Significantly enriched motifs can be extracted using the S4 method ptn_motifSelection function, and universalmotif results for each gene set included in `geneList` can be retrieved using the ptn_motifgeneList function.

If you use the `motifAnalysis` function in your analysis, please appropriately cite the memes R package, The MEME Suite, and the STREME tool publications provided below. Licensing The MEME Suite is free for non-profit use. However, for-profit users should contact OIC-MEMESuite@ucsd.edu and purchase a license. See http://meme-suite.org/doc/copyright.html for details

Value

The `motifAnalysis` function returns an updated `postNetData` object with results compiled into an S4 object of class `postNetMotifs` and stored in the "motifs" slot. Within the motif slot, results for each sequence region are provided as lists. The "motifSelection" list element corresponds to a character vector of identified motifs in the gene sets of interest that have enrichment p-values below the threshold set with the `stremeThreshold` parameter. Additional list elements are provided for each gene set of interest containing detailed information about the selected motifs in universalmotif format.

References

Timothy L. Bailey, James Johnson, Charles E. Grant, William S. Noble. "The MEME Suite", Nucleic Acids Research, 43(W1):W39-W49, 2015.

Timothy L. Bailey, "STREME: Accurate and versatile sequence motif discovery", Bioinformatics, 2021. https://doi.org/10.1093/bioinformatics/btab203

Nystrom SL, McKay DJ (2021). "Memes: A motif analysis environment in R using tools from the MEME Suite." PLOS Computational Biology, 17, 1-14. doi:10.1371/journal.pcbi.1008991, https://doi.org/10.1371/journal.pcbi.1008991.

See Also

memes
runStreme
universalmotif
contentMotifs
ptn_motifSelection
ptn_motifgeneList

Examples

```
# Note that as users must provide the path to the MEME Suite executables, it is not possible to
# create a fully executable example as this path will vary depending on many factors. An example of
```

```
# how to run this function is provided below, and can be updated with the correct memePath argument.

## Not run:
# load example data:
 data("postNetExample", package = "postNet")
 ptn <- postNetExample$ptn

 ptn <- motifAnalysis(ptn = ptn,
                      streMeThreshold = 0.05,
                      minwidth = 6,
                      memePath = "/meme/bin",
                      region = c('UTR5'),
                      subregion = NULL,
                      subregionSel = NULL)

# Extract the significantly enriched motifs:
 myMotifs <- ptn_motifSelection(ptn = ptn,
                                region = "UTR5")

 str(myMotifs)

# Extract full motif results from one gene set of interest:
  motifsTransUp <- ptn_motifgeneList(ptn = ptn,
                                     region = "UTR5",
                                     geneList = "translationUp")

 str(motifsTransUp)

## End(Not run)
```

`mouseSignatures` *Mouse gene signatures of translational control*

Description

The `mouseSignatures` dataset is a list of example gene signatures derived from published studies profiling mRNA translation downstream of mTOR signalling and activation of the integrated stress response (ISR). In addition, an example of a signature comprised of mRNAs containing 5'TOP motifs is also included. The list is not intended to be exhaustive, but provides examples of how gene signatures can be used in a **postNet** analysis to gain insights into post-transcriptional regulation of gene expression.

Usage

```
data("mouseSignatures")
```

Format

The format is: List of 5 $ Gandin_etal_2016_mTOR_transUp : chr [1:1487] "Zfy1" "Zfy2" "Isca2" "Rps14" ... $ Gandin_etal_2016_mTOR_transDown : chr [1:1765] "Lpin1" "Galnt1" "Atrx" "Tbc1d31"

... $ Cockman_etal_2020_classicTOP : chr [1:70] "Rpl37" "Rps14" "Hnrnpa1" "Rpl39" ... $ Guan_etal_2017_Tg1_transUp : chr [1:1336] "Mcm7" "Bcl2" "Usp33" "Gm13212" ... $ Guan_etal_2017_Tg1_transDown : chr [1:774] "Mapre3" "Mettl24" "Isg15" "Idh3g" ...

Details

The `mouseSignatures` dataset contains a list of gene signatures relevent to understanding possible pathways and mRNA features involved in regulation of mRNA translation, with a specific focus on mTOR and integrated stress response (ISR)-sensitive translation. These signatures have been compiled either directly from published studies, or from analyses of published datasets (sources and references described in detail below). Note that if the orignal source of the gene signature was from human, these gene IDs have been converted to mouse.

These gene signatures can be used with the plotSignatures, plotSignatures_ads, and signaturesHeatmap functions to perform statistical analyses and visualizations of how each gene signature is regulated in a given dataset. Furthermore, gene signatures can also be provided as features in the featureIntegration function which performs step-wise regression modelling or random forest classification to identify features that are most important in explaining post-transcriptional regulation in a given dataset.

Source

Genes translationally regulated downstream of mTOR:

- **Signature Information**:
  - Signature Name(s): Gandin_etal_2016_mTOR_transUp, Gandin_etal_2016_mTOR_transDown
  - Source: MCF7 cells
  - Species: Human
  - Conditions: Insulin (4.2 nM) + torin1 (250 nM), Insulin (4.2 nM) + DMSO; 4 hrs
  - Replicates: 4 Reps/Condition
  - Comparison: Insulin + torin1 vs. Insulin
- **Publication Information**:
  - Title: mTORC1 and CK2 coordinate ternary and eIF4F complex assembly
  - Method: Polysome fractionation followed by microarray
  - Data Source: Anota2seq analysis of microarray data deposited in GSE76766.
  - Analysis Method: Anota2seq algorithm (version 1.14.0). The following thresholds were applied within the anota2seqRun function: maxPAdj = 0.15; deltaP = log2(1.2); deltaT = log2(1.2); deltaPT = log2(1.2); deltaTP = log2(1.2); maxSlopeTranslation = 2; minSlopeTranslation = -1; minSlopeBuffering = -2; maxSlopeBuffering = 1. Replicate was included in the anota2seq model using the 'batchVec' parameter to account for batch effects.
  - First Author, Last Author, Year: Gandin, Topisirovic, 2016
  - PMID: 27040916
  - GEO Accession: GSE76766
  - DOI: 10.1038/ncomms11127

- **Dataset Summary**: Genes classified as translationally activated (transUp) or suppressed (transDown) upon inhibition of mTOR via torin1 treatment in insulin-stimulated MCF7 cells.

mRNAs containing 5'terminal oligopyrimidine (5'TOP) motifs:

- **Signature Information**:
  - Signature Name: Cockman_etal_2020_classicTOP
  - Source: N/A
  - Species: Human
  - Conditions: N/A
  - Replicates: N/A
  - Comparison: N/A
- **Publication Information**:
  - Title: TOP mRNPs: Molecular Mechanisms and Principles of Regulation
  - Method: N/A
  - Data Source: Data obtained from supplementary materials (see: Appendix A Table 1A of publication)
  - Analysis Method:
  - First Author, Last Author, Year: Cockman, Ivanov, 2020
  - PMID: 32605040
  - GEO Accession: N/A
  - DOI: 10.3390/biom10070969
- **Dataset Summary**: A curated list of well-validated 5'TOP motif-containing mRNAs.

Genes translationally regulated downstream of ISR activation:

- **Signature Information**:
  - Signature Name(s): Guan_etal_2017_Tg1_transUp, Guan_etal_2017_Tg1_transDown
  - Source: Mouse embryonic fibroblasts (MEFs)
  - Species: Mouse
  - Conditions: Thapsigargin (400 nM), Vehicle control (DMSO); 1 hr
  - Replicates: 4 Reps/Condition
  - Comparison: Thapsigargin vs. DMSO
- **Publication Information**:
  - Title: A Unique ISR Program Determines Cellular Responses to Chronic Stress
  - Method: Polysome fractionation followed by RNA seqeuncing
  - Data Source:
  - Analysis Method:
  - First Author, Last Author, Year: Guan, Hatzoglou, 2017
  - PMID: 29220654
  - GEO Accession: GSE90070
  - DOI: 10.1016/j.molcel.2017.11.007
- **Dataset Summary**: Genes translationally activated (transUp) or suppressed (transDown) following activation of the ISR via thapsigargin treatment in MEFs.

References

If you use these gene signatures in you analysis, please cite:

Gandin V, Masvidal L, Cargnello M, Gyenis L, McLaughlan S, Cai Y, Tenkerian C, Morita M, Balanathan P, Jean-Jean O, Stambolic V, Trost M, Furic L, Larose L, Koromilas AE, Asano K, Litchfield D, Larsson O, Topisirovic I. mTORC1 and CK2 coordinate ternary and eIF4F complex assembly. Nat Commun. 2016 Apr 4;7:11127. doi: 10.1038/ncomms11127. PMID: 27040916; PMCID: PMC4822005.

Cockman E, Anderson P, Ivanov P. TOP mRNPs: Molecular Mechanisms and Principles of Regulation. Biomolecules. 2020 Jun 27;10(7):969. doi: 10.3390/biom10070969. PMID: 32605040; PMCID: PMC7407576.

Guan BJ, van Hoef V, Jobava R, Elroy-Stein O, Valasek LS, Cargnello M, Gao XH, Krokowski D, Merrick WC, Kimball SR, Komar AA, Koromilas AE, Wynshaw-Boris A, Topisirovic I, Larsson O, Hatzoglou M. A Unique ISR Program Determines Cellular Responses to Chronic Stress. Mol Cell. 2017 Dec 7;68(5):885-900.e6. doi: 10.1016/j.molcel.2017.11.007. PMID: 29220654; PMCID: PMC5730339.

Examples

```
# Load the mouse signatures
 mouseSignatures <- get_signatures("mouse")
 str(mouseSignatures)
```

`plotFeaturesMap` *Explore mRNA features and regulatory effects using UMAP visualizations*

Description

The `plotFeaturesMap` function implements `umap` to visualize regulatory effects and features across genes.

Usage

```
plotFeaturesMap(ptn,
                regOnly = TRUE,
                comparisons = NULL,
                featSel,
                remBinary = TRUE,
                featCol = NULL,
                scaled = FALSE,
                centered = TRUE,
                remExtreme = NULL,
                pdfName = NULL)
```

Arguments

ptn	A postNetData object.
regOnly	Logical specifying whether UMAPs should include only regulated genes (i.e., those included in geneList), or all genes in the dataset. If regOnly = TRUE, only regulated genes will be used, if regOnly = FALSE, all genes will be considered. Default is TRUE.
comparisons	If regOnly = TRUE, provide a list of numeric vectors specifying pairwise comparisons between gene sets defined in the regulation or geneList slots of the postNetData object. Use 0 to denote the background set. For example, list(c(0,2), c(1,2)) compares gene set 2 to the background and gene set 1 to gene set 2. The default is NULL. If both regulation and geneList are provided, geneList is appended to the end so numbering will continue sequentially.
featSel	A character vector specifying the features used in featureIntegration to be used to generate UMAP visualizations. It is often advisable to start by exploring the data using the selected features from stepwise regression or random forest modelling.
remBinary	A logical specifying if binary features should be removed when generating UMAPs. If TRUE, all binary features will be removed before generating UMAPs. If FALSE, binary features will be included. Default is TRUE.
featCol	A character vector specifying which features should be coloured on the UMAPs, visualizing the relationship between the regulatory effect and the specified features. Default is NULL.
scaled	A logical specifying whether the input data should be scaled prior to generating UMAPs. Default is FALSE.
centered	A logical specifying whether the input data should be centered prior to generating UMAPs. Default is TRUE.
remExtreme	A numeric value between 0 and 1 specifying the threshold to define extreme values to be removed from inputs prior to generating colour scales for UMAPs. For example, remExtreme = 0.1 removes the top and bottom 10% of extreme values. This parameter only impacts colour scaling. Default is NULL.
pdfName	Name to be appended to output PDF files.

Details

UMAPs are highly sensitive to input parameters. Therefore, the default settings are a suggested starting point, but may not be appropriate for all inputs. These visualizations are intended to be used as a tool for data exploration and understanding relationships between different features and regulatory effects, particularly in cases where multiple features are co-occurring in the same mRNA.

Value

A PDF file with a UMAP visualization will be output for each feature specified with the featCol parameter. In each plot, the panel on the left is coloured according to the regulatory effect superimposed on the UMAP generated using the features specified in the featSel parameter, while the

panel on the right is coloured according to the individual feature superimposed on the same UMAP. If `featCol = NULL`, a PDF will be output for each feature in `featSel`.

The `plotFeaturesMap` function also returns an updated `postNetData` object with a data.frame of UMAP coordinates stored in the `featuresMap` slot.

See Also

umap
featureIntegration

Examples

```
 tmp <- tempfile(fileext = ".pdf")

# load and create example data:
 data("postNetExample", package = "postNet")
 ptn <- postNetExample$ptn

# Prepare the list of pre-calculated features to be used in feature integration modelling
 myFeatures <- postNetExample$features
 str(myFeatures)

# Run feature integration modelling using stepwise regression
 ptn <- featureIntegration(ptn = ptn,
                           features = myFeatures,
                           pdfName = tmp,
                           regOnly = TRUE,
                           allFeat = FALSE,
                           analysis_type = "lm",
                           covarFilt = 20,
                           comparisons = list(c(1,2)),
                           NetModelSel = "omnibus")

# Plot UMAP visualizations using the features identified in stepwise regression modelling
 ptn <- plotFeaturesMap(ptn = ptn,
                        regOnly = TRUE,
                        comparisons = list(c(1,2)),
                        featSel = c(names(ptn_selectedFeatures(ptn,
                                          analysis_type = "lm",
                                          comparison = 1)),"CDS_C"),
                        remBinary = TRUE,
                        scaled = TRUE,
                        centered = TRUE,
                        remExtreme = 0.1,
                        pdfName = tmp)
```

`plotSignatures` *Analysis of regulation of gene signatures in a* `postNetData` *object*

Description

The plotSignatures function assesses changes in the regulation of known gene signatures in the data set of interest contained in a postNetData object. Outputs include statistical analyses and visualizations of signature regulation.

Usage

```
plotSignatures(ptn,
               signatureList,
               signature_colours = NULL,
               dataName,
               generalName,
               xlim = c(-2, 2),
               tableCex = 0.7,
               pdfName = NULL)
```

Arguments

ptn	A postNetData object.
signatureList	A named list of vectors containing gene IDs for the signatures of interest to be examined. Note that several signatures of translational regulation are provided with the package for both human and mouse. These can be retrieved using the get_signatures function.
signature_colours	An optional character vector specifying colours for plotting each signature. Note that signature_colours must be the same length as signatureList. The default is NULL.
dataName	A character string specifying a name for the data set in which the gene signature(s) will be examined. This name will be appended to the name of the PDF file generated.
generalName	A character string specifying a name for the signatures that will be examined. This name will be appended to the name of the PDF file generated.
xlim	A numeric vector specifying the x-axis limits for the eCDF plots. Default is c(-2, 2).
tableCex	A numeric value specifying the size of the text in the table of statistical results that appears at the top of the eCDF plots. Default is 0.7.
pdfName	Name to be appended to output PDF files.

Details

Gene signatures of interest are visualized using empirical cumulative distribution functions (eCDFs) of the regulatory effect measurement in a postNetData object (often log2 fold changes). ECDFs for genes belonging to the signatures of interest are compared to those for all other genes (background). Differences in the regulatory effect measurement distributions are calculated at the quantiles. Significant directional shifts in the eCDFs for gene signature versus background are identified using a Wilcoxon rank-sum test.

Value

No value is returned. Graphical outputs are generated as PDF files.

See Also

signaturesHeatmap
plotSignatures_ads
get_signatures

Examples

```
 tmp <- tempfile(fileext = ".pdf")

# load example data:
 data("postNetExample", package = "postNet")
 ptn <- postNetExample$ptn

# load signature data:
 signatures <- get_signatures("human")

# Examine the regulation of mTOR-sensitive transcripts
# in the regulatory effects stored in a postNetData object:
 plotSignatures(ptn = ptn,
                signatureList = signatures[c("Gandin_etal_2016_mTOR_transUp",
                                             "Gandin_etal_2016_mTOR_transDown")],
                signature_colours = c("red", "blue"),
                dataName = "Example_data",
                generalName = "mTOR_sensitive_translation",
                xlim = c(-3, 3),
                tableCex = 1,
                pdfName = tmp)
```

`plotSignatures_ads` *Analysis of regulation of gene signatures in an* `Anota2seqDataSet` *object*

Description

The `plotSignatures_ads` function assesses changes in the regulation of known gene signatures of interest in an `Anota2seqDataSet` object from an **anota2seq** analysis. Outputs include statistical analyses and visualizations of signature regulation.

Usage

```
plotSignatures_ads(ads,
                   contrast,
                   dataName,
                   effects_names = c("Total mRNA Log2FC",
```

```
                                "Polysome-associated mRNA Log2FC",
                                "Buffering Log2FC",
                                "Translation Log2FC"),
    signatureList,
    generalName,
    signature_colours,
    xlim = c(-2, 2),
    scatterXY = NULL,
    tableCex = 1,
    pdfName = NULL)
```

Arguments

ads	An S4 object of class Anota2seqDataSet (resulting from the anota2seqRun function in the **anota2seq** package).
contrast	An integer selecting the **anota2seq** contrast to retrieve regulatory effect measurements from. For more details on contrasts, please see the **anota2seq** vignette.
dataName	A string specifying the name that will be given to the selected contrast from the Anota2seqDataSet object. For example: "Test vs. DMSO".
effects_names	A character vector specifying axis names for the fold change scatterplot and ecdf plots. The default is, c("Total mRNA Log2FC", "Polysome-associated mRNA Log2FC", "Buffering Log2FC", "Translation Log2FC").
signatureList	A named list of vectors containing gene IDs for the signatures of interest to be examined. Note that several signatures of translational regulation are provided with the package for both human and mouse. These can be retrieved using the get_signatures function.
generalName	A character vector specifying the names of the signatures to be plotted.
signature_colours	A character vector specifying colours for plotting each signature. Note that signature_colours must be the same length as signatureList.
xlim	A numeric vector specifying the x-axis limits for the ecdf plots. Default is c(-2, 2).
scatterXY	A numeric value specifying the scale of the x and y axes for the fold change scatterplot. For example, scatterXY = 3, would indicate that both the x and y axes will have a Log2FC range from -3 to 3. The default is NULL.
tableCex	A numeric value specifying the size of the text in the table of statistical results that appears at the top of the ecdf plots. Default is 1.
pdfName	Name to be appended to output PDF files.

Details

Gene signatures of interest are visualized in a scatterplot of log2 fold changes in polysome-associated and total mRNA. The empirical cumulative distribution functions (ECDFs) of log2 fold changes in polysome-associated mRNA, total mRNA, translation, and buffering (offsetting) are also plotted independently. Fold change ECDFs for genes belonging to the signatures of interest are compared

to those for all other genes (background). Differences in the fold change distributions are calculated at the quantiles. Significant directional shifts in the ECDFs for gene signature versus background are identified using a Wilcoxon Rank Sum test.

Value

No value is returned. Graphical outputs are generated as PDF files.

See Also

anota2seqRun
signaturesHeatmap
plotSignatures

Examples

```
local({
  oldwd <- getwd()
  on.exit(setwd(oldwd), add = TRUE)
  setwd(tempdir())

  # load example data:
  data("postNetExample", package = "postNet")

 # Initialize Anota2seqDataSet (see anota2seq vignette for details)
 ads <- anota2seq::anota2seqDataSetFromMatrix(
    dataP = postNetExample$ads_data$dataP,
    dataT = postNetExample$ads_data$dataT,
    phenoVec = postNetExample$ads_data$phenoVec,
    batchVec = c(1, 2, 3, 4, 1, 2, 3, 4),
    dataType = "RNAseq",
    normalize = FALSE)

  # Run an anota2seq analysis:
  # Note that the quality control and residual outlier testing are not
  # performed to limit the running time of this example. For full details
  # on running an analysis please see the anota2seq vignette and help manual.
  ads <- anota2seq::anota2seqRun(ads,
    performQC = FALSE,
    performROT = FALSE,
    useProgBar = FALSE)

  # load signature data:
  humanSignatures <- get_signatures("human")

 # Examine the regulation of mTOR-sensitive transcripts in the results of the anota2seq analysis
  plotSignatures_ads(ads = ads,
                                contrast = 1,
                                dataName = "Osmosis_1h",
                      signatureList = humanSignatures[c("Gandin_etal_2016_mTOR_transUp",
                                                        "Gandin_etal_2016_mTOR_transDown")],
```

```
                                generalName = "mTOR-sensitive translation",
                                signature_colours = c("red", "blue"),
                                xlim = c(-3, 3),
                                scatterXY = 4,
                                tableCex = 0.65,
                                pdfName = "mTORsignatures")
})
```

---

postNetData-class *S4 Class* postNetData*: Core data container for postNet analyses*

Description

The postNetData class stores the core data structure used throughout the postNet analysis. It contains input gene lists, sequence annotations and source information, and results from codon, motif, GSEA, GAGE, GO term, and feature integration analyses. This object is created by the postNetStart function and serves as the input/output container for most downstream functions.

Objects from the Class

Objects of this class are typically created using the constructor function postNetStart. Alternatively, they can be created manually using new("postNetData", ...).

Slots

species: Object of class "characterOrNULL" specifying the species associated with the sequence annotations and input gene lists, e.g. "human", "mouse". Used to load species-specific reference sequences.

version: Object of class "characterOrNULL" storing the version identifier for the RefSeq sequence annotations loaded.

selection: Object of class "character" indicating the method used to select representative transcript isoforms, e.g. "random", "shortest", or "longest".

seed: Object of class "ANY" storing the integer used to set the seed for random sampling if selection = "random"

annot: Object of class "postNetAnnot" storing the annotated sequences for each mRNA transcript region, "UTR5", "CDS", "CCDS", and "UTR3".

dataIn: Object of class "postNetDataIn" containing the user-defined input gene lists of interest, background gene set, regulatory effect measurement, and selected colours for plotting.

features: Object of class "dataframeOrNULL" with a data.frame containing the features for each gene (e.g., UTR lengths, GC content, codon or motif counts) that were used as input for stepwise regression and/or random forest models during featureIntegration analysis.

analysis: Object of class "postNetAnalysis" storing the results from analysis functions including codonUsage, motifAnalysis, gseaAnalysis, gageAnalysis, goAnalysis, miRNAanalysis and featureIntegration. Structured into nested S4 objects.

Methods

The following S4 accessor and utility methods are available for `postNetData` objects:

ptn_background `signature(x = "postNetData")`: Retrieve the background gene set.

ptn_codonAnalysis `signature(x = "postNetData")`: Access results from the codon or amino acid usage analysis.

ptn_codonSelection `signature(x = "postNetData")`: Retrieve the selected significantly enriched/depleted codons or amino acids.

ptn_colours `signature(x = "postNetData")`: Return the colours that will be used for visualizations.

ptn_dataIn `signature(x = "postNetData")`: Access the user-defined background gene set, gene lists of interest, regulatory effect measurements, and colours for visualizations.

ptn_effect `signature(x = "postNetData")`: Retrieve the user-supplied regulatory effect measurement (e.g., Log2 Fold Changes).

ptn_features `signature(x = "postNetData")`: Extract the `features` `data.frame` used as input for stepwise regression and random forest analyses.

ptn_geneID `signature(x = "postNetData")`: Return a character vector of gene IDs for the genes included in the gene sets of interest and background gene sets (all genes used in the analysis).

ptn_geneList `signature(x = "postNetData")`: Access the list of user-supplied gene sets of interest used in analyses.

ptn_id `signature(x = "postNetData")`: Return a character vector of transcript IDs for the transcripts included in the gene sets of interest and background gene sets (all transcripts used in the analysis).

ptn_motifgeneList `signature(x = "postNetData")`: Access the results and additional information on the de novo identified motifs for the specified gene list in universalmotif format.

ptn_motifSelection `signature(x = "postNetData")`: Return a character vector of de novo identified motifs in the gene sets of interest that have enrichment p-values below a user-specified threshold.

ptn_networkGraph `signature(x = "postNetData")`: Retrieve the network structure in igraph format generated during linear stepwise regression modelling with the featureIntegration function.

ptn_selection `signature(x = "postNetData")`: Return the user-defined transcript isoform selection strategy (e.g., `random`, `shortest`, `longest`).

ptn_sequences `signature(x = "postNetData")`: Retrieve a character vector containing the reference annotation sequences corresponding to the specified transcript region.

ptn_species `signature(x = "postNetData")`: Return the species associated with the analysis.

ptn_version `signature(x = "postNetData")`: Return the release version of the RefSeq annotations used to generate the reference sequences.

ptn_GAGE `signature(x = "postNetData")`: Access the results of GAGE analysis.

ptn_GSEA `signature(x = "postNetData")`: Access the results of GSEA analysis.

ptn_GO `signature(x = "postNetData")`: Access the results of GO term analysis.

ptn_miRNA_analysis `signature(x = "postNetData")`: Access the results of GAGE-based miRNA enrichment analysis.

ptn_miRNA_to_gene `signature(x = "postNetData")`: Return a list of genes that are predicted to be targeted by the miRNA that passed the filtering thresholds in miRNA analysis.

See Also

postNetStart
motifAnalysis
codonUsage
miRNAanalysis
gseaAnalysis
gageAnalysis
goAnalysis
featureIntegration
plotFeaturesMap
ptn_background
ptn_check_models
ptn_codonAnalysis
ptn_codonSelection
ptn_colours
ptn_dataIn
ptn_effect
ptn_features
ptn_GAGE
ptn_geneID
ptn_geneList
ptn_GO
ptn_GSEA
ptn_id
ptn_miRNA_analysis
ptn_miRNA_to_gene
ptn_model
ptn_motifgeneList
ptn_motifSelection
ptn_networkGraph
ptn_selectedFeatures
ptn_selection
ptn_sequences
ptn_species
ptn_version

Examples

```
showClass("postNetData")
```

`postNetExample` *An example dataset for demonstrating the usage of postNet functions*

Description

This object contains example data for testing postNet functions, including lists of post-transcriptionally regulated genes, background gene sets, and log_2 fold changes in translation efficiency between two treatment conditions.

Usage

```
data(postNetExample, package='postNet')
```

Format

The format is: List of 6 $ geneList :List of 2 ..$ translationUp : chr [1:10] "OCIAD1" "MSH2" "SCFD1" "INO80C" ... ..$ translationDown: chr [1:10] "XKR8" "ANPEP" "AGPAT2" "UBN2" ... $ background: chr [1:100] "SMARCD2" "ZNF239" "MTIF3" "ADGRE2" ... $ effect : Named num [1:100] -0.849 -2.506 -2.615 1.676 0.315 ... ..- attr(*, "names")= chr [1:100] "ADGRE2" "AGFG1" "AGPAT2" "AGPAT4" ... $ ptn :Formal class 'postNetData' [package "postNet"] with 8 slots .. ..@ species : chr "human" .. ..@ version : chr "ver_40.202408" .. ..@ selection: chr "random" .. ..@ seed : num 123 .. ..@ annot :Formal class 'postNetAnnot' [package "postNet"] with 4 slots .. ..@ UTR5:Formal class 'postNetRegion' [package "postNet"] with 3 slots .. ..@ id : chr [1:100] "NM_152916" "NM_001135189" "NM_001012727" "NM_020133" ... .@ geneID : chr [1:100] "ADGRE2" "AGFG1" "AGPAT2" "AGPAT4" ... .@ sequences: chr [1:100] "ACAGCTGCCCAGCCTGCGGAGACGGGACAGCCCTGTCC-CACTCACTCTTTCCCCTGCTGCTCCTGCCGGCAGCTCAGCTGGAACC" "CCCCTCAGGA-GAAGTCGGGAAGGTGGCGGCGGCGGCGGCGGTTGTCCCGGCTGTGCCGGTTGGTGTGGCCCGTCAGCCCGCG__truncated__ "AGCCCCGCCGCCCTCGCAATAAGGGGCCTGAGCGCGCGGGGGAGAAGCGGGAGCGGGAGCGGG"GCTTTTTCTTTCCAGTGTTGGCTGACTTACAGCTCTTATAAACTAGTGGCAATTTCTGAACCCAGCCGGCTCCAT__truncated__ ... .@ CDS :Formal class 'postNetRegion' [package "postNet"] with 3 slots .. ..@ id : chr [1:100] "NM_152916" "NM_001135189" "NM_001012727" "NM_020133" ... .@ geneID : chr [1:100] "ADGRE2" "AGFG1" "AGPAT2" "AGPAT4" ... .@ sequences: chr [1:100] "ATGGGAGGCCGCGTCTTTCTCGTCTTTCTCGCATTCTGTGTCTG-GCTGACTCTGCCGGGAGCTGAAACCCAGGACTCCAGGGGCTGTGCCCGGTGGTGCCCTCAGGACTCC"| __truncated__ "ATGGCGGCCAGCGCGAAGCGGAAGCAGGAGGAGAAGCACCTGAAGATGCT-GCGGGACATGACCGGCCTCCCGCACAACCGAAAGTGCTTCGACTGCGACCAGCGCGGCCCC"| __truncated__ "ATGGAGCTGTGGCCGTGTCTGGCCGCGGCGCTGCTGTTGCTGCTGCTGCTG-GTGCAGCTGAGCCGCGCGGCCGAGTTCTACGCCAAGGTCGCCCTGTACTGCGCGCTGTGC"| __truncated__ "ATGGACCTCGCGGGACTGCTGAAGTCTCAGTTCCTGTGCCACCTGGTCTTCT-GCTACGTCTTTATTGCCTCAGGGCTAATCATCAACACCATTCAGCTCTTCACTCTCCTC"| __truncated__ ... .@ UTR3:Formal class 'postNetRegion' [package "postNet"] with 3 slots .. ..@ id : chr [1:100] "NM_152916" "NM_001135189" "NM_001012727" "NM_020133" ... .@ geneID : chr [1:100] "ADGRE2" "AGFG1" "AGPAT2" "AGPAT4" ... .@ sequences: chr [1:100] "AAAAATCTTCTGAATAAGATCTTCCCTCTTTGCCCGTGGAAAATCT-GAACAATCTTTGAGCCATCTAGAGGGGAAAGAAAAGACTTTGTTCTGTGTGTTTCAAGAAATTCA"| __truncated__ "CCTTATATAGACAATTTACTGGAACGAACTTTTATGTGGTCACATTACATCTCTC-CACCTCTTGCACTGTTGTCTTGTTTCACTGATCTTAGCTTTAAACACAAGAGAAGT"| __truncated__ "CCCAGACCACGGCAGGGCATGACCTGGGGAGGGCAGGTGGAAGCCGATGGCTG-GAGGATGGGCAGAGGGGACTCCTCCCGGCTTCCAAATACCACTCTGTCCGGCTCCCCC"| __truncated__ "CTCAGGGAGGTGTCACCATCCGAAGGGAACCTTGGGGAACTGGTGGCCTCT-GCATATCCTCCTTAGTGGGACACGGTGACAAAGGCTGGGTGAGCCCCTGCTGGGCACGGC"| __truncated__ ... .@ CCDS: NULL .. ..@ dataIn :Formal class 'postNetDataIn' [package "postNet"] with 4 slots .. ..@ background: chr [1:100] "SMARCD2" "ZNF239" "MTIF3" "ADGRE2" ... .@ geneList :List of 2 .. ..$ translationUp : chr [1:10] "OCIAD1" "MSH2" "SCFD1" "INO80C" ... .$ translationDown: chr [1:10] "XKR8" "ANPEP" "AGPAT2" "UBN2" ... .@ effect : Named num [1:100] -0.849 -2.506 -2.615 1.676 0.315 ... .- attr(*, "names")= chr [1:100] "ADGRE2" "AGFG1" "AGPAT2" "AGPAT4" ... .

.. ..@ colours : chr [1:2] "#7FB7BE" "#DB7F67" .. ..@ features : NULL .. ..@ analysis :Formal class 'postNetAnalysis' [package "postNet"] with 7 slots .. ..@ featureIntegration: NULL .. ..@ motifs : NULL .. ..@ codons : NULL .. ..@ GO : NULL .. ..@ GSEA : NULL .. ..@ GAGE : NULL .. ..@ miRNA : NULL $ ads_data :List of 3 ..$ phenoVec: chr [1:8] "Control" "Control" "Control" "Control" ... ..$ dataP : num [1:100, 1:8] 3.3 6.54 4.69 3.62 5.78 ... .- attr(*, "dimnames")=List of 2 .. .. ..$: chr [1:100] "ADGRE2" "AGFG1" "AGPAT2" "AGPAT4" ... .$: chr [1:8] "riboSeq_Control_rep1" "riboSeq_Control_rep2" "riboSeq_Control_rep3" "riboSeq_Control_rep4" ... ..$ dataT : num [1:100, 1:8] 3.53 6.78 5.64 3.05 4.85 ... .- attr(*, "dimnames")=List of 2 .. .. ..$: chr [1:100] "ADGRE2" "AGFG1" "AGPAT2" "AGPAT4" ... .$: chr [1:8] "rnaSeq_Control_rep1" "rnaSeq_Control_rep2" "rnaSeq_Control_rep3" "rnaSeq_Control_rep4" ... $ features :List of 17 ..$ UTR5_length : Named num [1:100] 6.41 8.06 6.64 7.73 5.25 ... .- attr(*, "names")= chr [1:100] "ADGRE2" "AGFG1" "AGPAT2" "AGPAT4" ... ..$ CDS_length : Named num [1:100] 11.18 10.62 9.53 10.15 10.61 ... .- attr(*, "names")= chr [1:100] "ADGRE2" "AGFG1" "AGPAT2" "AGPAT4" ... ..$ UTR3_length : Named num [1:100] 11.91 12.71 9.25 12.68 4.17 ... .- attr(*, "names")= chr [1:100] "ADGRE2" "AGFG1" "AGPAT2" "AGPAT4" ... ..$ UTR5_G : Named num [1:100] 24.7 42.1 49 21.6 44.7 ... .- attr(*, "names")= chr [1:100] "ADGRE2" "AGFG1" "AGPAT2" "AGPAT4" ... ..$ UTR5_C : Named num [1:100] 42.4 39.8 33 24.9 18.4 ... .- attr(*, "names")= chr [1:100] "ADGRE2" "AGFG1" "AGPAT2" "AGPAT4" ... ..$ UTR5_A : Named num [1:100] 15.29 8.27 13 20.19 18.42 ... .- attr(*, "names")= chr [1:100] "ADGRE2" "AGFG1" "AGPAT2" "AGPAT4" ... ..$ UTR5_T : Named num [1:100] 17.65 9.77 5 33.33 18.42 ... .- attr(*, "names")= chr [1:100] "ADGRE2" "AGFG1" "AGPAT2" "AGPAT4" ... ..$ CDS_G : Named num [1:100] 25.6 21.7 31.2 27.1 19.2 ... .- attr(*, "names")= chr [1:100] "ADGRE2" "AGFG1" "AGPAT2" "AGPAT4" ... ..$ CDS_C : Named num [1:100] 30.5 26.1 33.6 27.2 23.1 ... .- attr(*, "names")= chr [1:100] "ADGRE2" "AGFG1" "AGPAT2" "AGPAT4" ... ..$ CDS_A : Named num [1:100] 21.8 27.3 15.5 22.2 23.6 ... .- attr(*, "names")= chr [1:100] "ADGRE2" "AGFG1" "AGPAT2" "AGPAT4" ... ..$ CDS_T : Named num [1:100] 22.1 24.9 19.7 23.6 34.1 ... .- attr(*, "names")= chr [1:100] "ADGRE2" "AGFG1" "AGPAT2" "AGPAT4" ... ..$ UTR3_G : Named num [1:100] 20.7 17.5 34 23.9 22.2 ... .- attr(*, "names")= chr [1:100] "ADGRE2" "AGFG1" "AGPAT2" "AGPAT4" ... ..$ UTR3_C : Named num [1:100] 22.3 15.7 32.5 22.7 11.1 ... .- attr(*, "names")= chr [1:100] "ADGRE2" "AGFG1" "AGPAT2" "AGPAT4" ... ..$ UTR3_A : Named num [1:100] 28.7 30.8 17.7 27.1 44.4 ... .- attr(*, "names")= chr [1:100] "ADGRE2" "AGFG1" "AGPAT2" "AGPAT4" ... ..$ UTR3_T : Named num [1:100] 28.3 36 15.8 26.2 22.2 ... .- attr(*, "names")= chr [1:100] "ADGRE2" "AGFG1" "AGPAT2" "AGPAT4" ... ..$ uORFs_ATG_strong: Named num [1:100] 0 0 0 0 0 0 0 0 1 ... .- attr(*, "names")= chr [1:100] "ADGRE2" "AGFG1" "AGPAT2" "AGPAT4" ... ..$ UTR5_SGCSGCS : Named num [1:100] 0 6 2 0 0 2 0 0 0 1 ... .- attr(*, "names")= chr [1:100] "ADGRE2" "AGFG1" "AGPAT2" "AGPAT4" ...

Details

The example data provided is a small subset of the data taken from a study of the effect of osmotic stress on mRNA translation. Immortalized human corneal epithelial cells (10.014O pRSV-T) were cultured under osmotic stress (500 mOsm, NaCl) for 1h, along with controls.The provided gene lists are detrived from an anota2seq analysis of ribosome profiling and RNA-seq data to identify genes that were translationally activated or suppressed under osmotic stress.

A set of representative mRNA features was also enumerated in these translationally regulated genes, along with signatures of genes for which translation is know to be regulated downstream of mTOR and the integrated stress response (ISR). This set of features is used to illustrate the feature integra-

tion modelling and UMAP visualizations.

Source

Krokowski D, Jobava R, Szkop KJ, Chen CW, Fu X, Venus S, Guan BJ, Wu J, Gao Z, Banaszuk W, Tchorzewski M, Mu T, Ropelewski P, Merrick WC, Mao Y, Sevval AI, Miranda H, Qian SB, Manifava M, Ktistakis NT, Vourekas A, Jankowsky E, Topisirovic I, Larsson O, Hatzoglou M. Stress-induced perturbations in intracellular amino acids reprogram mRNA translation in osmoadaptation independently of the ISR. Cell Rep. 2022 Jul 19;40(3):111092. doi: 10.1016/j.celrep.2022.111092. PMID: 35858571; PMCID: PMC9491157.

Data available at GEO: GSE200097.

Examples

```
postNetExample <- data(postNetExample, package='postNet')
str(postNetExample)
```

`postNetStart` *Create the data container and inputs for postNet analyses*

Description

The `postNetStart` function is the first step in a **postNet** analysis workflow. It retrieves and compiles all necessary inputs and annotations into a `postNetData` S4 object, including:

- Reference sequence annotations (5'UTR, CDS, 3'UTR), including in-built, custom, or retrieved directly from the NCBI RefSeq database. By default, 'source = "load"' meaning 'postNetStart()' will load one of the in-built reference sequence annotations provided with the package when it is first needed. The data are downloaded from the postNet GitHub releases and stored in a user-specific cache directory managed by BiocFileCache. Once cached, the reference data are reused in subsequent sessions and do not require re-downloading. This option is available when 'species = "human"' or '"mouse"'. In-built annotations are based on NCBI RefSeq GRCh38 (human) and GRCm39 (mouse) genome assemblies and corresponding transcript annotations.The following reference annotation versions are currently available:

  Human (RefSeq GRCh38): ver_40.202408 Mouse (RefSeq GRCm39): ver_27.202402

  Optionally, UTR sequences can also be adjusted if more precise sequences are available (e.g. from CAGE, QuantSeq, etc.)
- Gene sets of interest (either from an **anota2seq** analysis by providing the `Anota2seqDataSet` object, or using custom gene lists of interest).
- A numeric regulation effect measurement (e.g., fold changes in gene expression or translation efficiency, etc.) either from an **anota2seq** analysis, or from a custom source. These values will be used in the featureIntegration analysis, where mRNA features, gene signatures, and/or other variables can be used to explain changes in the regulation effect measurement using forward stepwise regression and/or random forest approaches.

The resulting `postNetData` object is then passed to downstream functions in the **postNet** workflow, and stores analysis results.

Usage

```
postNetStart(
  ## ----------------------------
  ## 1. Input Data (Anota2seq or Custom)
  ## ----------------------------
  ads = NULL,
  regulation = NULL,
  contrast = NULL,
  regulationGen = NULL,
  contrastSel = NULL,
  geneList = NULL,
  geneListcolours = NULL,
  customBg = NULL,
  effectMeasure = NULL,

  ## ----------------------------
  ## 2. Reference Sequence Annotations
  ## ----------------------------
  source,
  species = NULL,
  customFile = NULL,
  fastaFile = NULL,
  posFile = NULL,
  rna_gbff_file = NULL,
  rna_fa_file = NULL,
  genomic_gff_file = NULL,
  selection = "random",
  setSeed = NULL,

  ## ----------------------------
  ## 3. Adjusting Sequences (Custom UTRs)
  ## ----------------------------
  adjObj = NULL,
  region_adj = NULL,
  excl = FALSE,
  keepAll = FALSE
)
```

Arguments

ads	An optional S4 object of class Anota2seqDataSet-class (resulting from the anota2seqRun function in the **anota2seq** package). Lists of regulated genes (regulatory modes) and regulatory effect measures will be retrieved from this object for the analysis. **Do not** provide both ads and geneList at the same time.
regulation	If using ads, provide a character vector of regulatory modes to retrieve from the Anota2seqDataSet object. Valid options include any individual selection, or combination of: "translationUp", "translationDown", "translatedmRNAUp", "translatedmRNADown", "bufferingmRNAUp", "bufferingmRNADown", "mRNAAbundanceUp",

"mRNAAbundanceDown", "totalmRNAUp", "totalmRNADown". Please see the **anota2seq** vignette for additional details on regulatory modes.

contrast — If using ads, provide a numeric vector specifying which **anota2seq** contrast each element in regulation should be selected from. For example, c(1,1,2) selects the first two regulatory modes provided in regulation from contrast 1, and the third mode from contrast 2. This vector must always be the same length as the input for regulation, even if the Anota2seqDataSet has only one contrast. For more details on contrasts, please see the **anota2seq** vignette.

regulationGen — If using ads, select the "general" regulation effect measurement that will be taken from the ads object. These values will be used in the downstream feature integration where the mRNA features (and/or other input signatures) quantified in the specified gene sets (regulation) will be used to explain changes in the regulatory effect measurement (i.e. translation efficiency, etc.) with the featureIntegration function. This parameter can be **one** of: "translation", "buffering", "mRNAAbundance", "translatedmRNA", or "totalmRNA".

contrastSel — If using ads, select the **anota2seq** contrast where the "general" regulation effect measurement corresponding to regulationGen should be extracted from. This must be a single numeric value.

geneList — If **anota2seq** was not used, instead of providing ads, gene sets of interest can be provided as a named list of one or more character vectors. Each list element should be a set of genes (for example, list(set1=c("GeneA","GeneB"), set2=c("GeneC","GeneD"))). Ensure that the gene IDs provided match those in the reference sequence annotations.

geneListcolours — If using geneList, provide a character vector of colours to be used for plotting. This must have the same length as geneList. If using ads, the colours for each regulatory mode from **anota2seq** will be used automatically.

customBg — If using geneList, optionally provide a character vector of gene/transcript IDs corresponding to the custom background. This background will be used for statistical comparisons against the gene sets of interest provided in geneList. Note that the background should correspond to the set of genes or transcripts that was quantified (i.e., all genes used as input in the analysis that generated the regulatory effect measurement [effectMeasure]). For example, all genes in a given data set passing expression/reproducibility thresholds. Note that all genes in geneList must be found in customBg. If customBg is NULL, the background will be set as all gene IDs present in the provided geneList. Although optional, it is strongly recommended to carefully consider that the appropriate background is used. If using ads, the appropriate background is automatically retrieved from the Anota2seqDataSet object, and customBg should be NULL.

effectMeasure — If using geneList, provide a numeric vector corresponding to a custom regulation effect measurement. For example, fold changes in mRNA expression or translation efficiency (although any continuous numeric variable could theoretically be used). The elements of the vector should be named with the gene/transcript IDs in a format corresponding to those provided in the geneList. This parameter should not be provided when using ads, as the regulation effect measurement will be retrieved from the Anota2seqDataSet using the regulationGen parameter.

`selection` Specifies which mRNA isoform will be selected from the reference sequence annotation for use in analyses in cases when multiple isoforms are present for a given gene. This parameter can be one of:

`"longest"` Selects the longest sequence isoform.

`"shortest"` Selects the shortest sequence isoform.

`"random"` (Default) Selects an isoform at random. Note that selecting the longest or shortest isoforms for all genes may skew results for some analyses. For this reason, the default selection is `"random"`. However, this may lead to slight differences in results if the `postNetStart` function is run multiple times. For cases where multiple analyses may be run using the same data, the `setSeed` parameter can be used to ensure that random isoform selection is stable each time `postNetStart` is run.

`setSeed` If `selection` is `"random"`, provide a single integer value to be used with `set.seed` to ensure that the random isoform selection is reproducible between different runs of the `postNetStart` function. Any integer can be provided (for example 123), however, the same value must be provided between different analyses for isoform selection to be reproducible.

`source` Select the source of the reference sequence annotations. Note that the gene/transcript annotations used in the `ads` or `geneList` should be the same as those used in the reference sequence annotations. This parameter can be set to:

`"load"` (Default) Load a pre-made RefSeq reference sequence annotation of the selected `species` included in **postNet**. Currently, this option is supported with the `species` `"human"` and `"mouse"`.

`"create"` Automatically download the most recent version of the RefSeq annotation files from the NCBI database for the selected `species` and build a new reference sequence annotation locally. Currently, this option is supported with the `species` `"human"` and `"mouse"`.

`"createFromSourceFiles"` Use local RefSeq reference files already downloaded from the NCBI RefSeq database to create annotation sequences. Three files are required, the RNA GBFF ("rna.gbff"), RNA FASTA ("rna.fna"), and genomic GFF ("genomic.gff"). These files must be provided using the `rna_gbff_file`, `rna_fa_file`, and `genomic_gff_file` parameters. This option can be used to assemble reference sequence annotations for RefSeq versions not currently supported with `"load"`, and unlike the `"create"` option does not require an internet connection if the reference files have already been downloaded. Currently, this option is supported with the `species` `"human"` and `"mouse"`.

`"createFromFasta"` Alternatively, reference sequence annotations can be assembled from a FASTA file (provided with the `fastaFile` parameter) with reference sequences, and a file specifying the positions of the sequence regions (provided with the `posFile` parameter). Optionally provide a genomic GFF file (with the `genomic_gff_file` parameter). If no GFF is provided the latest version for `species` will be automatically downloaded from the NCBI RefSeq database as for `"create"`. Currently, this option is supported with the `species` `"human"` and `"mouse"`.

`"custom"` Use a pre-prepared reference sequence annotation file provided with the `customFile` parameter. This option can be used when working with

sequences from species not currently supported by the options above, or with annotations other than those in the NCBI RefSeq database. The customFile provided must be tab-delimited and contain the columns: transcriptID, geneID, UTR5_seq, CDS_seq, and UTR3_seq.

species Select the species of interest. Currently, "human" and "mouse" are supported. This parameter is required unless source is "custom".

customFile If source is "custom", provide the name and file path to the custom reference sequence annotation file. The customFile must be tab-delimited, with the headings: transcriptID, geneID, UTR5_seq, CDS_seq, UTR3_seq.

fastaFile If source is "createFromFasta", provide the name and file path to a FASTA file to be used to assemble reference sequence annotations.

posFile If source is "createFromFasta", provide the name and file path to a file specifying positions of transcript sequence regions in the fastaFile. The posFile must be tab-delimited, with the headings: transcriptID, utr5_stop, cds_stop, total_length. Please see the vignette for additional details.

rna_gbff_file If source is "createFromSourceFiles", provide the name and file path to an RNA GBFF reference file for either human or mouse. This can be downloaded from the NCBI RefSeq database. Note that the file should be compressed with gzip (.gz extension), which is the default format for NCBI RefSeq downloads.

rna_fa_file If source is "createFromSourceFiles", provide the name and file path to an RNA FASTA reference file for either human or mouse. This can be downloaded from the NCBI RefSeq database. Note that the file should be compressed with gzip (.gz extension), which is the default format for NCBI RefSeq downloads.

genomic_gff_file

If source is "createFromSourceFiles", provide the name and file path to a genomic GFF reference file for either human or mouse. This can be downloaded from the NCBI RefSeq database. Note that the file should be compressed with gzip (.gz extension). If source is "createFromFasta", an unzipped genomic GFF can be specified manually using this parameter. Alternatively, leaving NULL will automatically download the most recent version for species from the NCBI RefSeq database.

adjObj A named list of custom UTR sequences that will replace the existing UTR sequences in the annotation file. List elements must be named "UTR5" and/or "UTR3", and elements should be a named character vector of UTR sequences where names are transcript IDs (for example, NM_...), and values are the new UTR sequences. This parameter is useful when more precise measurements of UTR sequences are available (for example, from CAGE or QuantSeq data).

region_adj Character vector specifying which UTR regions in your annotation should be adjusted by the provided adjObj. This parameter can be used to adjust both UTR regions (e.g., c("UTR5","UTR3")), or one at a time.

excl When providing custom UTR sequences with adjObj, sequences may not be available for all genes/isoforms included in the existing annotation file. This parameter determines whether to keep all genes/isoforms, or exclude those that are not included in adjObj. If TRUE, any transcript IDs present in the existing annotation file that are not present in adjObj will be removed. If FALSE,

then the UTR sequences in the annotation file will be replaced with the custom sequences in adjObj when they are available, but for transcript IDs where a custom sequence is not available, the original sequence present in the annotation file will be retained and used in the analysis. The default is FALSE.

keepAll When using custom UTR sequences with adjObj and excl is FALSE, the keepAll parameter determines how the custom UTR sequences are integrated into the existing annotation file for genes with multiple isoforms. If TRUE, the UTR sequences in the annotation file will be replaced with the custom sequences corresponding to the same transcript IDs. However, if there are multiple transcript IDs (isoforms) for the same gene ID, all isoforms will be retained, including transcript IDs where custom sequences were not available in adjObj. Note that unless the custom sequences are always the longest or shortest isoforms that can be selected using the selection parameter, when keepAll is TRUE, the custom sequences are not guaranteed to be the ones considered if a gene-level analysis is performed. If FALSE, transcript IDs (isoforms) corresponding to the same gene that are not found in the custom sequences provided in adjObj will be removed, such that only the custom UTR sequences will be available for genes with multiple isoforms. The default is FALSE.

Details

By default, postNetStart will load or create reference sequence annotations from RefSeq for species, assemble or load gene lists of interest, define a suitable background gene set, and store a numeric regulation effect measurement (e.g., fold change) for downstream modeling.

Reproducibility Note: If selection = "random", repeated calls without setSeed can produce different isoform selections. To ensure reproducible results between different analyses of the same dataset, set setSeed to a fixed integer.

When adjusting annotations via adjObj, ensure that transcript IDs match those in your reference annotation. You can choose to exclude or keep genes that lack updated sequences via excl and keepAll.

The outputs of each step of postNet analysis will also be stored in the postNetData object, and can be retrieved through the use of a series of helper functions (see Note and examples below).

See the **postNet** vignette for a more extensive tutorial and examples.

Value

An S4 object of class postNetData containing:

- **Reference Sequence Annotations:** 5'UTR, CDS, and 3'UTR sequences for each gene and/or transcript isoform, with the possibility to customize UTRs via adjObj.
- **Gene Lists:** One or more sets of gene IDs of interest, as well as a set of gene IDs to be used as background for statistical comparisons (from ads, or geneList and customBg).
- **Regulation Effect Measurement:** A numeric vector of an effect of interest, (e.g., translation efficiency fold changes) that will be modelled using mRNA features and/or other signatures of interest.
- **Metadata:** The chosen species, sequence annotation version, and isoform selection method.

- **Analysis Results:** The results of subsequent steps of the **postNet** workflow, such as `featureIntegration`, are stored in the `postNetData` object.

Note

The helper functions available to retrieve inputs and results from the postNetData object are:

ptn_background

ptn_codonAnalysis

ptn_codonSelection

ptn_colours

ptn_dataIn

ptn_effect

ptn_features

ptn_GAGE

ptn_geneID

ptn_geneList

ptn_GO

ptn_GSEA

ptn_id

ptn_miRNA_analysis

ptn_miRNA_to_gene

ptn_model

ptn_motifgeneList

ptn_motifSelection

ptn_selectedFeatures

ptn_selection

ptn_sequences

ptn_species

ptn_version

References

https://www.ncbi.nlm.nih.gov/refseq/ for RefSeq documentation. https://bioconductor.org/packages/release/bioc/vignettes/anota2seq/inst/doc/anota2seq.pdf for **anota2seq** details.

See Also

`anota2seqRun` for obtaining the anota2seq `ads` object.
`featureIntegration` for requirements of postNet feature integration analysis.

Examples

```
 tmp <- tempfile(fileext = ".pdf")

# ------------------------------------------------------------------------
# Example 1: Using custom gene lists and background
# ------------------------------------------------------------------------

# load example data:
 data("postNetExample", package = "postNet")

# Genes of interest should be provided in a named list.
 myGenes <- postNetExample$geneList
 str(myGenes)

# All gene IDs in the list should be present in the background.
 myBg <- postNetExample$background
 str(myBg)

# The regulation effect measurement must be named with the same gene IDs
# present in the background (or the gene list if no custom background is provided).
 myEffect <- postNetExample$effect
 str(myEffect)

# Initialize the postNetData object with custom gene lists:
 ptn <- postNetStart(
   geneList = myGenes,
   geneListcolours = c("#7FB7BE","#DB7F67"),
   customBg = myBg,
   effectMeasure = myEffect,
   selection = "random",
   setSeed = 123,
   source = "load",
   species = "human"
 )

## Not run:

# ------------------------------------------------------------------------
# Example 2: Using an anota2seq object with built-in RefSeq annotations
# ------------------------------------------------------------------------

# Instead of using gene sets of interest, the results of an anota2seq analysis
# can also be provided.

# Initialize Anota2seqDataSet (see anota2seq vignette for details)
 ads <- anota2seq::anota2seqDataSetFromMatrix(
    dataP = postNetExample$ads_data$dataP,
    dataT = postNetExample$ads_data$dataT,
    phenoVec = postNetExample$ads_data$phenoVec,
    batchVec = c(1, 2, 3, 4, 1, 2, 3, 4),
    dataType = "RNAseq",
    normalize = FALSE)
```

```
# Run an anota2seq analysis:
# Note that the quality control and residual outlier testing are not
# performed to limit the running time of this example. For full details
# on running an analysis please see the anota2seq vignette and help manual.
 ads <- anota2seq::anota2seqRun(ads,
    performQC = FALSE,
    performROT = FALSE,
    useProgBar = FALSE)

# This postNet analysis will allow identification of mRNA features,
# and comparison between translationally activated and suppressed genes.

# Feature integration analysis will model changes in translation
# efficiency.

# Initialize the postNetData object, where "ads" is an Anota2seqDataSet object
# resulting from an anota2seqRun call:
  ptn <- postNetStart(
      ads = ads,
      regulation = c("translationUp","translationDown"),
      contrast = c(1,1),
      regulationGen = "translation",
      contrastSel = 1,
      selection = "random",
      setSeed = 123,   # ensures reproducibility of random isoform selection
      source = "load",
      species = "human"
    )

# ---------------------------------------------------------------------------
# Example 3: Creating annotation from local RefSeq files
# ---------------------------------------------------------------------------

#  Example RefSeq annotation files for mouse downloaded from the NCBI database:
#  - "myMouse_rna.gbff.gz"
#  - "myMouse_rna.fa.gz"
#  - "myMouse_genomic.gff.gz"

# Initialize the postNetData object with custom gene lists and RefSeq annotations,
# and considering the longest mRNA isoform for genes with multiple isoforms:
 ptn <- postNetStart(
   geneList = myGenes,
   geneListcolours = c("#F2A104","#00743F"),
   customBg = myBg,
   effectMeasure = myEffect,
   selection = "longest",
   source = "createFromSourceFiles",
   species = "mouse",
   rna_gbff_file = "myMouse_rna.gbff.gz",
   rna_fa_file = "myMouse_rna.fa.gz",
   genomic_gff_file = "myMouse_genomic.gff.gz"
 )
```

```
## End(Not run)

# ------------------------------------------------------------------------
# Example 4: Retrieving data from the postNetData object
# ------------------------------------------------------------------------

# Inputs and results stored in the slots of the postNetData object can be
# easily retrieved with the use of ptn helper functions.

# To extract reference annotation sequences for 5'UTRs from a postNetData object ptn:
 myUTR5seqs <- ptn_sequences(ptn,region = "UTR5")

 str(myUTR5seqs)

# To extract the RefSeq annotation release version
# from postNetData object ptn (if using source = "load"):
 myVersion <- ptn_version(ptn)

 myVersion
```

`postNetVignette` *An example dataset for demonstrating the workflow of a postNet ananlysis*

Description

This object contains example data to run a postNet analysis, including lists of post-transcriptionally regulated and background genes, and log2 fold changes in translation efficiency between two treatment conditions.

Usage

```
data(postNetVignette, package='postNet')
```

Format

The format is: List of 5 $ geneList :List of 2 ..$ translationUp : chr [1:1445] "DDX6" "CLK1" "DUSP1" "TXNIP" ... ..$ translationDown: chr [1:1378] "PPDPF" "NCKAP1" "AP2M1" "ABHD12" ... $ background: chr [1:9555] "A4GALT" "AAAS" "AACS" "AADAT" ... $ effect : Named num [1:9555] -2.095 0.275 -0.57 -0.608 0.576 ... ..- attr(*, "names")= chr [1:9555] "A4GALT" "AAAS" "AACS" "AADAT" ... $ features :List of 18 ..$ UTR5_length : Named num [1:9555] 8.34 7.18 7.23 7.71 4.91 ... .- attr(*, "names")= chr [1:9555] "A4GALT" "AAAS" "AACS" "AADAT" ... ..$ UTR3_length : Named num [1:9555] 9.59 4.86 11.96 9.46 11.07 ... .- attr(*, "names")= chr [1:9555] "A4GALT" "AAAS" "AACS" "AADAT" ... ..$ UTR5_C : Named num [1:9555] 28.1 32.4 54.7 30.1 36.7 ... .- attr(*, "names")= chr [1:9555] "A4GALT" "AAAS" "AACS" "AADAT" ... ..$ UTR5_A : Named num [1:9555] 18.5 14.5 6 21.1 16.7 ... .- attr(*, "names")= chr [1:9555] "A4GALT" "AAAS" "AACS" "AADAT" ... ..$ UTR5_T : Named num [1:9555] 19.4

22.1 14 14.8 10 ... .- attr(*, "names")= chr [1:9555] "A4GALT" "AAAS" "AACS" "AADAT" ... ..$ UTR3_G : Named num [1:9555] 36.8 10.3 26.6 16.6 23.4 ... .- attr(*, "names")= chr [1:9555] "A4GALT" "AAAS" "AACS" "AADAT" ... ..$ UTR3_C : Named num [1:9555] 26.3 13.8 25.9 17 17.9 ... .- attr(*, "names")= chr [1:9555] "A4GALT" "AAAS" "AACS" "AADAT" ... ..$ UTR3_A : Named num [1:9555] 20.4 27.6 20 34.1 28.1 ... .- attr(*, "names")= chr [1:9555] "A4GALT" "AAAS" "AACS" "AADAT" ... ..$ UTR3_T : Named num [1:9555] 16.5 48.3 27.4 32.3 30.5 ... .- attr(*, "names")= chr [1:9555] "A4GALT" "AAAS" "AACS" "AADAT" ... ..$ Up : Named num [1:9555] 0.00567 0.05068 0.04141 0.11059 0.13968 ... .- attr(*, "names")= chr [1:9555] "A4GALT" "AAAS" "AACS" "AADAT" ... ..$ Down : Named num [1:9555] 0.2125 0.1326 0.1871 0.0541 0.0762 ... .- attr(*, "names")= chr [1:9555] "A4GALT" "AAAS" "AACS" "AADAT" ... ..$ uORFs_ATG_strong : Named num [1:9555] 2 0 0 0 0 0 0 0 0 0 ... .- attr(*, "names")= chr [1:9555] "A4GALT" "AAAS" "AACS" "AADAT" ... ..$ UTR5_SCSCGS : Named num [1:9555] 0 1 5 1 0 3 0 0 0 1 ... .- attr(*, "names")= chr [1:9555] "A4GALT" "AAAS" "AACS" "AADAT" ... ..$ Gandin_etal_2016_mTOR_transUp : Named num [1:9555] 1 0 0 0 0 0 0 0 0 0 ... .- attr(*, "names")= chr [1:9555] "A4GALT" "AAAS" "AACS" "AADAT" ... ..$ Gandin_etal_2016_mTOR_transDown: Named num [1:9555] 0 0 0 0 0 0 0 0 0 0 ... .- attr(*, "names")= chr [1:9555] "A4GALT" "AAAS" "AACS" "AADAT" ... ..$ Cockman_etal_2020_classicTOP : Named num [1:9555] 0 0 0 0 0 0 0 0 0 0 ... .- attr(*, "names")= chr [1:9555] "A4GALT" "AAAS" "AACS" "AADAT" ... ..$ Guan_etal_2017_Tg1_transUp : Named num [1:9555] 0 0 0 0 0 0 0 0 0 0 ... .- attr(*, "names")= chr [1:9555] "A4GALT" "AAAS" "AACS" "AADAT" ... ..$ Guan_etal_2017_Tg1_transDown : Named num [1:9555] 0 0 0 0 0 0 0 0 0 0 ... .- attr(*, "names")= chr [1:9555] "A4GALT" "AAAS" "AACS" "AADAT" ... $ ads_data :List of 3 ..$ phenoVec: chr [1:8] "Control" "Control" "Control" "Control" ... ..$ dataP : num [1:100, 1:8] 6.15 7.85 8.62 6.15 5.8 ... .- attr(*, "dimnames")=List of 2 .. .. ..$: chr [1:100] "ACOX1" "ACTR2" "AGRN" "APOBEC3C" ... .$: chr [1:8] "riboSeq_Control_rep1" "riboSeq_Control_rep2" "riboSeq_Control_rep3" "riboSeq_Control_rep4" ... ..$ dataT : num [1:100, 1:8] 5.77 8.23 9.39 5.34 6.4 ... .- attr(*, "dimnames")=List of 2 .. .. ..$: chr [1:100] "ACOX1" "ACTR2" "AGRN" "APOBEC3C" ... .$: chr [1:8] "rnaSeq_Control_rep1" "rnaSeq_Control_rep2" "rnaSeq_Control_rep3" "rnaSeq_Control_rep4" ...

Details

The example data provided with the postNet package is taken from a study of the effect of osmotic stress on mRNA translation. Immortalized human corneal epithelial cells (10.014O pRSV-T) were cultured under osmotic stress (500 mOsm, NaCl) for 1h, along with controls.The provided gene lists are detrived from an anota2seq analysis of ribosome profiling and RNA-seq data to identify genes that were translationally activated or suppressed under osmotic stress.

A set of representative mRNA features was also enumerated in these translationally regulated genes, along with signatures of genes for which translation is know to be regulated downstream of mTOR and the integrated stress response (ISR). This set of features is used to illustrate the feature integration modelling and UMAP visualizations.

Source

Krokowski D, Jobava R, Szkop KJ, Chen CW, Fu X, Venus S, Guan BJ, Wu J, Gao Z, Banaszuk W, Tchorzewski M, Mu T, Ropelewski P, Merrick WC, Mao Y, Sevval AI, Miranda H, Qian SB, Manifava M, Ktistakis NT, Vourekas A, Jankowsky E, Topisirovic I, Larsson O, Hatzoglou M. Stress-induced perturbations in intracellular amino acids reprogram mRNA translation in osmoadaptation

independently of the ISR. Cell Rep. 2022 Jul 19;40(3):111092. doi: 10.1016/j.celrep.2022.111092. PMID: 35858571; PMCID: PMC9491157.

Data available at GEO: GSE200097.

Examples

```
postNetVignette <- data(postNetVignette, package='postNet')
str(postNetVignette)
```

`ptn_background` *Accessor for the gene set used as background in a* `postNetData` *object*

Description

The `background` slot holds a character vector of gene IDs used as the background in statistical comparisons performed in **postNet** analyses.

Usage

```
## S4 method for signature 'postNetData'
ptn_background(ptn)
```

Arguments

`ptn`	A postNetData object.

Value

A character vector containing the gene IDs for the set of genes used as background.

See Also

`postNetStart` for details on background gene sets.

Examples

```
# load and create example data:
  data("postNetExample", package = "postNet")
  ptn <- postNetExample$ptn

# Access the set of background genes used in postNet analyses
  myBg <- ptn_background(ptn)
  str(myBg)
```

ptn_check_models *Accessor for the feature integration model comparisons in a* postNetData *object*

Description

Retrieve the comparisons used for the models resulting from both the stepwise linear regression and random forest implementations of featureIntegration from a postNetData object.

Usage

```
ptn_check_models(ptn, analysis_type)
```

Arguments

ptn	A postNetData object.
analysis_type	A string specifying the method of analysis used for featureIntegration to retrieve the selected features from. The options are "lm" to retrieve features selected by stepwise linear regression, or "rf" for features selected using random forest-based analyses.

Value

Returns a character string specifying the comparisons that were used in modelling. For example "translationUp_c1_translationDown_c1" indicates the gene sets that were compared, and which anota2seq contrast they were selected from (e.g. c1 = contrast 1).

See Also

featureIntegration
ptn_model

Examples

```
 tmp <- tempfile(fileext = ".pdf")

 # load and create example data:
 data("postNetExample", package = "postNet")
 ptn <- postNetExample$ptn

# Prepare the list of pre-calculated features to be used in feature integration modelling
 myFeatures <- postNetExample$features
 str(myFeatures)

# Run feature integration modelling using stepwise regression
 ptn <- featureIntegration(ptn = ptn,
                           features = myFeatures,
                           pdfName = tmp,
```

```
                            regOnly = TRUE,
                            allFeat = FALSE,
                            analysis_type = "lm",
                            covarFilt = 20,
                            comparisons = list(c(1,2)),
                            NetModelSel = "omnibus")

# Retrieve the contrasts that were used in modelling
  ptn_check_models(ptn, analysis_type = "lm")
```

`ptn_codonAnalysis` *Accessor for the "codonAnalysis" slot of a* `postNetData` *object*

Description

Retrieve the "codonAnalysis" slot from a `postNetData` object, which holds the results of the codon usage analysis produced using the codonUsage function.

Usage

```
## S4 method for signature 'postNetData'
ptn_codonAnalysis(ptn)
```

Arguments

`ptn`	A postNetData object.

Value

The value returned by the `ptn_codonAnalysis` function is a `data.frame` with the following columns: geneID, codon, AA, count, frequency, AACountPerGene, and relative_frequency. Here, frequency corresponds to the number of codons relative to all codons in the gene, and relative_frequency corresponds to the number of codons relative to all synonymous codons in the gene.

See Also

codonUsage
ptn_codonSelection

Examples

```
tmp <- tempfile(fileext = ".pdf")

# load example data:
data("postNetExample", package = "postNet")
ptn <- postNetExample$ptn

 # Run codon usage analysis of single codons
 ptn <- codonUsage(ptn = ptn,
```

```
                    annotType = 'ptnCDS',
                    sourceSeq = "load",
                    analysis = 'codon',
                    codonN = 1,
                    pAdj = 0.01,
                    rem5 = TRUE,
                    plotHeatmap = FALSE,
                    thresOddsUp = 0.4,
                    thresFreqUp = 0.4,
                    thresOddsDown = 0.4,
                    thresFreqDown = 0.4,
                    subregion = NULL,
                    subregionSel = NULL,
                    comparisons = list(c(1,2)),
                    plotType_index = "violin",
                    pdfName = tmp)

# Get the results of the codon usage analysis

codonResults <- ptn_codonAnalysis(ptn)
str(codonResults)
```

`ptn_codonSelection` *Accessor for the "codonSelection" slot of a* `postNetData` *object*

Description

Retrieve the codons or amino acids that were selected as significantly enriched or depleted in the specified comparison from a `postNetData` object.

Usage

```
## S4 method for signature 'postNetData'
ptn_codonSelection(ptn, comparison)
```

Arguments

`ptn`	A postNetData object.
`comparison`	An integer specifying which comparison to select results from. For example, `1` would select the first comparison specified in the `comparisons` parameter of the codonUsage function.

Value

The value returned is a named list of the selected codons or AAs that were significantly enriched or depleted (according to the thresholds selected in the codonUsage function) in the specified comparisons. This list can be used as input for the codonCalc function.

See Also

codonUsage
codonCalc

Examples

```
 tmp <- tempfile(fileext = ".pdf")

# load example data:
 data("postNetExample", package = "postNet")
 ptn <- postNetExample$ptn

# Run codon usage analysis of single codons
 ptn <- codonUsage(ptn = ptn,
          annotType = 'ptnCDS',
          sourceSeq = "load",
          analysis = 'codon',
          codonN = 1,
          pAdj = 0.01,
          rem5 = TRUE,
          plotHeatmap = FALSE,
          thresOddsUp = 0.4,
          thresFreqUp = 0.4,
          thresOddsDown = 0.4,
          thresFreqDown = 0.4,
          subregion = NULL,
          subregionSel = NULL,
          comparisons = list(c(1,2)),
          plotType_index = "violin",
          pdfName = tmp)

# Select codons of interest that were significantly enriched or depleted
# with high frequency, and the highest and lowest odds ratios

 codons <- ptn_codonSelection(ptn, comparison = 1)
```

---

ptn_colours *Accessor for the "colours" slot of a* postNetData *object*

Description

Retrieve the "colours" slot of a postNetData object, which holds a character vector with the user-provided colours used for plotting. The vector must be the same length as the number of elements in geneList (e.g., one colour for each gene set of interest).

Usage

```
## S4 method for signature 'postNetData'
ptn_colours(ptn)
```

Arguments

ptn	A postNetData object.

Value

A character vector specifying the colours to be used for plotting.

See Also

postNetStart for details on selecting colours.
postNetData-class

Examples

```
# load and create example data:
 data("postNetExample", package = "postNet")
 ptn <- postNetExample$ptn

# Access version of the RefSeq annotation used to create the reference sequences
  myCols <- ptn_colours(ptn)
  str(myCols)
```

---

ptn_dataIn *Accessor for the "background", "geneList", "effect", and "colours" slots of a* postNetData *object*

Description

Retrieve the "background", "geneList", "effect", and "colours" slots from a postNetData object. If ads has been provided, these are retrieved from the Anota2seqDataSet based on the regulation, contrast, regulationGen, and contrastSel arguments of the postNetStart function. If not using ads, these inputs are compiled from the geneList, geneListcolours, effectMeasure, and optionally the customBg arguments.

Usage

```
## S4 method for signature 'postNetData'
ptn_dataIn(ptn)
```

Arguments

ptn	A postNetData object.

Value

Returns an S4 object of class postNetDataIn with the following slots:

- **background**: a character vector of gene/transcript IDs corresponding to the background used for statistical comparisons against the gene sets of interest in geneList. This background is either taken automatically from the Anota2seqDataSet if using ads, or if not using ads it is created from a user-provided geneList or provided directly with customBg.
- **geneList**: a list of one or more character vectors corresponding to the gene sets of interest to be used in the **postNet** analysis. If using ads, gene sets corresponding to **anota2seq** regulatory modes are selected using the regulation argument. If not using ads, the list of gene sets is provided using geneList.
- **effect**: a numeric vector corresponding to the regulation effect measurement. The elements of the vector are named with the gene/transcript IDs present in either ads, geneList, or customBg. If using ads, the regulatory effect measurement is selected by regulationGen. If using geneList, it is provided with effectMeasure.
- **colours**: a character vector of the same length as geneList specifying the colours to be used for plotting. If using geneList, colours are user-provided. If using ads, the default colours for each regulatory mode from **anota2seq** will be used automatically.

See Also

postNetStart for more information on required inputs for postNet analysis.
postNetData-class

Examples

```
# load and create example data:
 data("postNetExample", package = "postNet")
 ptn <- postNetExample$ptn

# Access the input data (background, geneList, effect, colours) stored in the postNetData object:
  myDataIn <- ptn_dataIn(ptn)
  str(myDataIn)
```

---

ptn_effect *Accessor for the regulatory effect measurement from the "effect" slot of a* postNetData *object*

Description

Retrieve the regulatory effect measurement from a postNetData object. If using geneList, the "effect" slot holds a user-provided named numeric vector corresponding to a custom regulation effect measurement. If using ads, the values in the "effect" slot will correspond to the selected regulationGen from ads.

Usage

```
## S4 method for signature 'postNetData'
ptn_effect(ptn)
```

Arguments

ptn	A postNetData object.

Value

A numeric vector with names corresponding to the gene/transcript IDs in geneList or ads.

See Also

postNetStart for details on selecting or providing the regulatory effect measurement.
postNetData-class

Examples

```
# load and create example data:
 data("postNetExample", package = "postNet")
 ptn <- postNetExample$ptn

# Access the regulatory effect measurement stored in the postNetData object:
  myEffect_out <- ptn_effect(ptn)
  str(myEffect_out)
```

---

ptn_features *Accessor for the* features *slot of a* postNetData *object*

Description

Retrieve the input features used in stepwise regression and random forest modelling from the postNetData object.

Usage

```
## S4 method for signature 'postNetData'
ptn_features(ptn)
```

Arguments

ptn	A postNetData object.

Value

Returns a data.frame of features (or gene signatures, see signCalc) quantified during a postNet analysis and used as input for featureIntegration. Rows correspond to genes, and columns correspond to features.

See Also

featureIntegration
signCalc

Examples

```
 tmp <- tempfile(fileext = ".pdf")

# load and create example data:
 data("postNetExample", package = "postNet")
 ptn <- postNetExample$ptn

# Prepare the list of pre-calculated features to be used in feature integration modelling:
 myFeatures <- postNetExample$features
 str(myFeatures)

# Run feature integration modelling using forward stepwise regression:
 ptn <- featureIntegration(ptn = ptn,
                           features = myFeatures,
                           pdfName = tmp,
                           regOnly = TRUE,
                           allFeat = FALSE,
                           analysis_type = "lm",
                           covarFilt = 20,
                           comparisons = list(c(1,2)),
                           NetModelSel = "omnibus")

# Retrieve a data.frame summarizing the features for each gene used in modelling:
 featureInput <- ptn_features(ptn)
 str(featureInput)
```

---

ptn_GAGE *Accessor for the "GAGE" slot of a* postNetData *object*

Description

Retrieve the "GAGE" slot from a postNetData object, which holds the results of GAGE analysis created by the gageAnalysis function.

Usage

```
ptn_GAGE(ptn,
        category,
        direction,
        threshold)
```

Arguments

ptn	A postNetData object.
category	A character vector specifying one or more gene ontology categories to be included in the analysis. The options are "BP", "CC", "MF", and "KEGG". For example, category = c("BP", "MF").
direction	A character indicating the directionality of regulation of the gene sets. Can be either "greater", or "less".
threshold	A numeric value providing the FDR threshold to be used to select results from the GAGE analysis.

Value

The value returned by the ptn_GAGE function is a data.frame with the following columns (see the gage package for full details):

- **p.geomean**: The geometric mean of the individual p-values from multiple single array-based gene set tests.
- **stat.mean**: The mean of the individual statistics from multiple single array-based gene set tests. The value denotes the magnitude of the gene-set level changes, and the sign denotes the direction of the changes.
- **p.val**: The global p-value or summary of the individual p-values from multiple single array-based gene set tests.
- **q.val**: The BH adjusted p-value, as implemented by the **multtest** package.
- **set.size**: The number of genes included in the gene set.
- **Genes**: The gene IDs in the input data belonging to the gene ontology gene set.

References

See the gage package for details.

Luo, W., Friedman, M., Shedden K., Hankenson, K. and Woolf, P GAGE: Generally Applicable Gene Set Enrichment for Pathways Analysis. BMC Bioinformatics 2009, 10:161.

See Also

gage
gageAnalysis
postNetStart

Examples

```
# load example data:
 data("postNetExample", package = "postNet")
 ptn <- postNetExample$ptn

# Run GAGE:
```

```
ptn <- gageAnalysis(ptn,
                    category = "CC")

# Extract the significant enrichment results from the postNetData object:
gageOut <- ptn_GAGE(ptn = ptn,
                    category = "CC",
                    direction = "greater",
                    threshold = 1)
str(gageOut)
```

`ptn_geneID` *Accessor for the gene IDs in the "geneID" slot of a* `postNetData` *object*

Description

Retrieve the gene IDs for the reference annotation sequences from a `postNetData` object. The annotations for the different sequence regions (5'UTR, CDS, 3'UTR, and optionally CCDS) are lists of three character vectors ("id", "geneID", and "sequence"). The `ptn_geneID` function retrieves the gene IDs for sequences from the specified mRNA region.

Usage

```
## S4 method for signature 'postNetData'
ptn_geneID(ptn, region)
```

Arguments

`ptn`	A postNetData object.
`region`	The sequence region from which to access the transcript IDs. Can be either: `"UTR5"`, `"CDS"`, or `"UTR3"`.

Value

A character vector containing the gene IDs corresponding to the reference sequence annotations for the specified mRNA region.

See Also

`postNetStart` for more information on reference sequence annotations.
`postNetData-class`

Examples

```
# load and create example data:
 data("postNetExample", package = "postNet")
 ptn <- postNetExample$ptn

# Access the transcript IDs corresponding to the
# reference coding sequences (CDS) from the postNetData object:
  myGeneIDs <- ptn_geneID(ptn, "CDS")
  str(myGeneIDs)
```

---

ptn_geneList *Accessor for the "geneList" slot of a* postNetData *object*

Description

Retrieve the geneList slot from a postNetData object, which holds a user-provided named list of one or more character vectors corresponding to gene sets of interest to be examined in the postNet analysis.

Usage

```
## S4 method for signature 'postNetData'
ptn_geneList(ptn)
```

Arguments

ptn	a postNetData object.

Value

A named list of one or more character vectors corresponding to gene sets of interest.

See Also

postNetStart for details on providing gene sets of interest.
postNetData-class

Examples

```
# load and create example data:
 data("postNetExample", package = "postNet")
 ptn <- postNetExample$ptn

# Access the gene sets of interest stored in the postNetData object:
  myGeneList <- ptn_geneList(ptn)
  str(myGeneList)
```

ptn_GO *Accessor for the "GO" slot of a* postNetData *object*

Description

Retrieve the "GO" slot from a postNetData object, which holds the results of GO term enrichment analysis created by the goAnalysis function.

Usage

```
ptn_GO(ptn,
       category,
       geneList,
       threshold)
```

Arguments

ptn	A postNetData object.
category	A character vector specifying one or more Gene Ontology categories to extract results for. The options are 'BP', 'CC', 'MF', and 'KEGG'. For example, category = c('BP','MF').
geneList	A string specifying the name of the gene set of interest to extract results for. This must match one of the names of the elements of geneList in the postNetData object.
threshold	A numeric value providing the FDR threshold to be used to select results from the GO term analysis.

Value

A data.frame of GO term enrichment analysis results that is stored in the GO slot of the postNetData object, and written to an Excel file. Each row of the data.frame corresponds to a term, with the following columns:

- **ID**: The ID of the term or pathway tested.
- **Description**: The name of the term. Note that if the category used in the analysis was "KEGG", there are two additional columns with descriptive information called "category" and "subcategory".
- **Count**: The number of genes in the set belonging to the term or pathway
- **Size**: The size of the pathway after removing genes not present in the input.
- **pvalue**: The enrichment p-value.
- **adjusted_pvalue**: The BH-adjusted p-value.
- **geneID**: The gene IDs in the input data belonging to the term/pathway.

References

See clusterProfiler for full details of enrichment analysis implementation in R.

If you use the `goAnalysis` function in your analysis, please cite:

S Xu, E Hu, Y Cai, Z Xie, X Luo, L Zhan, W Tang, Q Wang, B Liu, R Wang, W Xie, T Wu, L Xie, G Yu. Using clusterProfiler to characterize multiomics data. Nature Protocols. 2024, 19(11):3292-3320

See Also

clusterProfiler
goAnalysis
postNetStart
slopeFilt
goDotplot

Examples

```
 tmp <- tempfile(fileext = ".xlxs")

# load example data:
 data("postNetExample", package = "postNet")
 ptn <- postNetExample$ptn

# Run GO term analysis:
 ptn <- goAnalysis(ptn = ptn,
                   category=c("BP"),
                   name = tmp)

# Extract the significant enrichment results from the postNetData object:
 goOut <- ptn_GO(ptn,
                 category = "BP",
                 geneList = "translationUp",
                 threshold = 0.05)
 str(goOut)
```

`ptn_GSEA` *Accessor for the "GSEA" slot of a* `postNetData` *object*

Description

Retrieve the "GSEA" slot from a `postNetData` object, which holds the results of GSEA created by the gseaAnalysis function.

Usage

```
ptn_GSEA(ptn, threshold)
```

Arguments

ptn	A postNetData object.
threshold	A numeric value providing the FDR threshold to be used to select results from the GSEA.

Value

A data.frame of GSEA results that is stored in the GSEA slot of the postNetData object, and written to a tab-delimited text file. Each row of the data.frame corresponds to a term, with the following columns (also see fgseaMultilevel from the **fgsea** package):

- **Term**: The name of the term or pathway tested.
- **ES**: The enrichment score.
- **NES**: The normalized enrichment score, adjusted to the mean enrichment of randomly sampled size-matched sets.
- **log2err**: The expected error for the standard deviation of the P-value logarithm.
- **count**: The number of genes in the set belonging to the term or pathway.
- **size**: The size of the pathway after removing genes not present in the input.
- **pvalue**: The enrichment p-value.
- **adjusted_pvalue**: The BH-adjusted p-value.
- **Genes**: The gene IDs in the input data belonging to the term/pathway.

References

See fgsea for full details of GSEA implementation in R.

See https://www.gsea-msigdb.org/gsea/index.jsp for GSEA documentation.

A. Subramanian, P. Tamayo, V.K. Mootha, S. Mukherjee, B.L. Ebert, M.A. Gillette, A. Paulovich, S.L. Pomeroy, T.R. Golub, E.S. Lander, & J.P. Mesirov. Gene set enrichment analysis: A knowledge-based approach for interpreting genome-wide expression profiles, Proc. Natl. Acad. Sci. U.S.A. 102 (43) 15545-15550, (2005).

Mootha, V., Lindgren, C., Eriksson, KF. et al. PGC-1alpha-responsive genes involved in oxidative phosphorylation are coordinately downregulated in human diabetes. Nat Genet 34, 267–273 (2003).

See Also

fgsea
gseaAnalysis
gseaPlot

Examples

```
 tmp <- tempfile(fileext = ".txt")

# load example data:
 data("postNetExample", package = "postNet")
 ptn <- postNetExample$ptn

# Create example custom gene sets for GSEA:
 inSet <- list(set=sample(unlist(postNetExample$geneList[[1]]), 10))

# Run GSEA on custom gene sets:
 ptn <- gseaAnalysis(ptn = ptn,
                     geneSet = inSet,
                     name = tmp)

# Extract enrichment results from the postNetData object with an FDR < 0.05:
 gseaOut <- ptn_GSEA(ptn, threshold=0.05)
```

---

ptn_id *Accessor for the transcript IDs in the "id" slot of a* postNetData *object*

Description

Retrieve the transcript IDs for the reference annotation sequences from a postNetData object. The annotations for the different sequence regions (5'UTR, CDS, 3'UTR, and optionally CCDS) are lists of three character vectors ("id", "geneID", and "sequence"). The ptn_id function retrieves the transcript IDs for sequences from the specified mRNA region.

Usage

```
## S4 method for signature 'postNetData'
ptn_id(ptn, region)
```

Arguments

ptn	A postNetData object.
region	The sequence region from which to access the transcript IDs. Can be either: "UTR5", "CDS", or "UTR3".

Value

A character vector containing the transcript IDs corresponding to the reference sequence annotations for the specified mRNA region.

See Also

postNetStart for more information on reference sequence annotations.
postNetData-class

Examples

```
# load and create example data:
 data("postNetExample", package = "postNet")
 ptn <- postNetExample$ptn

# Access the transcript IDs corresponding
# to the reference 5'UTR sequences from the postNetData object
  myTransIDs <- ptn_id(ptn, "UTR5")
  str(myTransIDs)
```

`ptn_miRNA_analysis` *Accessor for the "miRNA_analysis" slot of a* `postNetData` *object*

Description

Retrieve the "miRNA_analysis" slot from a `postNetData` object, which holds the results of GAGE analysis identifying miRNA target enrichments in gene sets of interest created by the miRNAanalysis function.

Usage

```
ptn_miRNA_analysis(ptn,
                   direction,
                   threshold)
```

Arguments

`ptn`	A postNetData object.
`direction`	A character indicating the directionality of regulation of the gene sets. Can be either `"greater"`, or `"less"`.
`threshold`	A numeric value providing the FDR threshold to be used to select results from the GAGE miRNA enrichment analysis.

Details

Note that enrichments in miRNA predicted to target genes that are upregulated (e.g., if the regulatory effect measurement is log2 fold change) that appear in the results table labelled "greater" can be interpreted as those miRNA that may be downregulated or otherwise not active in the experimental condition. Likewise, enrichments in miRNA predicted to target genes that are downregulated that appear in the results table labelled "less" can be interpreted as those miRNA that may be upregulated or active in the experimental condition.

Value

The value returned by the `ptn_miRNA_analysis` function is a data.frame with the following columns (see the gage package for full details):

- **p.geomean**: The geometric mean of the individual p-values from multiple single array-based gene set tests.
- **stat.mean**: The mean of the individual statistics from multiple single array-based gene set tests. The value denotes the magnitude of the gene-set level changes, and the sign denotes the direction of the changes.
- **p.val**: The global p-value or summary of the individual p-values from multiple single array-based gene set tests.
- **q.val**: The BH adjusted p-value, as implemented by the **multtest** package.
- **set.size**: The number of genes included in the gene set.

References

If you use the `miRNAanalysis` function in your analysis, please cite:

McGeary SE, Lin KS, Shi CY, Pham T, Bisaria N, Kelley GM, Bartel DP. The biochemical basis of microRNA targeting efficacy. Science Dec 5, (2019).

Agarwal V, Bell GW, Nam J, Bartel DP. Predicting effective microRNA target sites in mammalian mRNAs. eLife, 4:e05005, (2015). eLife Lens view.

Luo, W., Friedman, M., Shedden K., Hankenson, K. and Woolf, P GAGE: Generally Applicable Gene Set Enrichment for Pathways Analysis. BMC Bioinformatics 2009, 10:161.

See Also

gage
miRNAanalysis
ptn_miRNA_to_gene
postNetStart

Examples

```
## Not run:
# load example data:
 data("postNetExample", package = "postNet")
 ptn <- postNetExample$ptn

# Run GAGE miRNA target enrichment analysis:
 ptn <- miRNAanalysis(ptn = ptn,
                      miRNATargetScanFile = "miRNA_predictions_hsa.txt",
                      contextScore = -0.2,
                      Pct = 0.4)

# Extract the significant enrichment results from the postNetData object:
 miRNAout <- ptn_miRNA_analysis(ptn = ptn,
                     direction = "less",
```

```
                    threshold = 1)
 str(miRNAout)

## End(Not run)
```

`ptn_miRNA_to_gene` *Accessor for the "miRNA_to_gene" slot of a* `postNetData` *object*

Description

Retrieve the "miRNA_to_gene" slot from a `postNetData` object, which stores a list of genes that are predicted to be targeted by the miRNA that passed the filtering thresholds for `contextScore` and/or `Pct`.

Usage

```
ptn_miRNA_to_gene(ptn, miRNAs)
```

Arguments

`ptn`	A postNetData object.
`miRNAs`	A character vector specifying one or more miRNAs for which to extract the list of target genes.

Value

A list where each element is a vector of genes that are predicted to be targeted by the miRNA that passed the filtering thresholds for `contextScore` and/or `Pct`.

References

If you use the `miRNAanalysis` function in your analysis, please cite:

McGeary SE, Lin KS, Shi CY, Pham T, Bisaria N, Kelley GM, Bartel DP. The biochemical basis of microRNA targeting efficacy. Science Dec 5, (2019).

Agarwal V, Bell GW, Nam J, Bartel DP. Predicting effective microRNA target sites in mammalian mRNAs. eLife, 4:e05005, (2015). eLife Lens view.

Luo W., Friedman M., Shedden K., Hankenson K., and Woolf P. GAGE: Generally Applicable Gene Set Enrichment for Pathways Analysis. BMC Bioinformatics 2009, 10:161.

See Also

gage
miRNAanalysis
ptn_miRNA_analysis
postNetStart

Examples

```
## Not run:
# load example data:
 data("postNetExample", package = "postNet")
 ptn <- postNetExample$ptn

# Run GAGE miRNA target enrichment analysis:
 ptn <- miRNAanalysis(ptn = ptn,
                      miRNATargetScanFile = "miRNA_predictions_hsa.txt",
                      contextScore = -0.2,
                      Pct = 0.4)

# Extract the significant enrichment results from the postNetData object:
 miRNAtargets <- ptn_miRNA_to_gene(ptn = ptn,
                     miRNAs = c("hsa-miR-138-5p", "hsa-miR-182-5p"))
 str(miRNAtargets)

## End(Not run)
```

`ptn_model` *Accessor for the feature integration model slots of a* `postNetData` *object*

Description

Retrieve the models resulting from both the stepwise linear regression and random forest implementations of featureIntegration from a `postNetData` object.

Usage

```
ptn_model(ptn,
          analysis_type,
          model,
          comparison)
```

Arguments

`ptn`	A postNetData object.
`analysis_type`	A string specifying the method of analysis used for featureIntegration to retrieve the selected features from. The options are `"lm"` to retrieve features selected by forward stepwise linear regression, or `"rf"` for features selected using random forest-based analyses.
`model`	A string specifying the model to retrieve. If `analysis_type = "lm"`, the options are: `"univariateModel"`, `"stepwiseModel"`, or `"finalModel"`. If `analysis_type = "rf"`, the options are: `"preModel"`, `"borutaModel"`, or `"finalModel"`.
`comparison`	An integer specifying which comparison to select the model from. For example, `1` would select the first comparison specified in the `comparisons` parameter of the featureIntegration function.

Value

If `analysis_type = "lm"` and `model = "univariateModel"`, an object of class `postNetUnivariate` is returned with three slots containing the p-values, FDRs, and proportion of variance explained for each feature passing the significance threshold in univariate models.

If `analysis_type = "lm"` and `model = "stepwiseModel"`, an object of class `postNetStepWise` is returned with two slots containing the linear models from each step of forward stepwise regression, and the F-value table storing the F-values for each feature at each step of forward stepwise regression.

If `analysis_type = "lm"` and `model = "finalModel"`, an object of class `postNetFinalModel` is returned with three slots containing the total variance explained by the final omnibus model, the final omnibus linear model, and a table summarizing the p-values and variance explained by each feature in the omnibus and adjusted models.

If `analysis_type = "rf"` and `model = "preModel"`, an object of class `randomForest` is returned for the pre-model using all features. See randomForest for full description of the output object.

If `analysis_type = "rf"` and `model = "borutaModel"`, an object of class `Boruta` is returned with the results of feature selection. See Boruta for full description of the output object.

If `analysis_type = "rf"` and `model = "finalModel"`, an object of class `randomForest` is returned for the final model using the selected features. See randomForest for full description of the output object.

References

Liaw A, Wiener M (2002). "Classification and Regression by randomForest." R News, 2(3), 18-22. https://CRAN.R-project.org/doc/Rnews/.

Kursa MB., Witold R. Rudnicki (2010). "Feature Selection with the Boruta Package." Journal of Statistical Software, 36(11), 1–13. https://doi.org/10.18637/jss.v036.i11.

See Also

featureIntegration
Boruta
randomForest

Examples

```
 tmp <- tempfile(fileext = ".pdf")

# load and create example data:
 data("postNetExample", package = "postNet")
 ptn <- postNetExample$ptn

# Prepare the list of pre-calculated features to be used in feature integration modelling
 myFeatures <- postNetExample$features
 str(myFeatures)

# Run feature integration modelling using forward stepwise regression:
 ptn <- featureIntegration(ptn = ptn,
                           features = myFeatures,
```

```
                        pdfName = tmp,
                        regOnly = TRUE,
                        allFeat = FALSE,
                        analysis_type = "lm",
                        covarFilt = 20,
                        comparisons = list(c(1,2)),
                        NetModelSel = "omnibus")

# Retrieve the omnibus model from forward stepwise regression analysis:
 finalModel <- ptn_model(ptn, analysis_type = "lm", model = "finalModel", comparison = 1)
 str(finalModel)
```

`ptn_motifGeneList` *Accessor for the de novo motif analysis results for each gene set of interest in a* `postNetData` *object*

Description

Retrieve the results of de novo motif identification for each gene set of interest in a `postNetData` object. Data for all motifs identified for each gene set of interest in the input `geneList` with enrichment p-values below the given threshold are stored in universalmotif format.

Usage

```
## S4 method for signature 'postNetData'
ptn_motifGeneList(ptn, region, geneList)
```

Arguments

`ptn`	A postNetData object.
`region`	A string specifying the sequence region of interest. This can be one of `"UTR5"`, `"UTR3"`, or `"CDS"`.
`geneList`	A string providing the name of the gene set of interest from `geneList` to extract de novo motif results for.

Value

The `ptn_motifGeneList` function returns additional information on the selected de novo identified motifs for the specified gene list in universalmotif format.

References

Timothy L. Bailey, James Johnson, Charles E. Grant, William S. Noble, "The MEME Suite", Nucleic Acids Research, 43(W1):W39-W49, 2015.

Timothy L. Bailey, "STREME: Accurate and versatile sequence motif discovery", Bioinformatics, 2021. https://doi.org/10.1093/bioinformatics/btab203

See Also

memes
runStreme
universalmotif
motifAnalysis

Examples

```
# Note that as users must provide the path to the MEME Suite executables, it is not possible to
# create a fully executable example as this path will vary depending on many factors. An example of
# how to run this function is provided below, and can be updated with the correct memePath argument.

## Not run:
# load example data:
 data("postNetExample", package = "postNet")
 ptn <- postNetExample$ptn

 ptn <- motifAnalysis(ptn = ptn,
                      stremeThreshold = 0.05,
                      minwidth = 6,
                      memePath = "/meme/bin",
                      region = c('UTR5'),
                      subregion = NULL,
                      subregionSel = NULL)


# Extract full motif results from one gene set of interest:
  motifsTransUp <- ptn_motifGeneList(ptn = ptn,
                                     region = 'UTR5',
                                     geneList = 'translationUp')

 str(motifsTransUp)

## End(Not run)
```

`ptn_motifSelection` *Accessor for the "motifSelection" slot of a* `postNetData` *object*

Description

Retrieve the "motifSelection" slot from a `postNetData` object, which holds de novo identified motifs in the gene sets of interest that have enrichment p-values below the given threshold.

Usage

```
## S4 method for signature 'postNetData'
ptn_motifSelection(ptn, region)
```

Arguments

ptn	A postNetData object.
region	A string specifying the sequence region of interest. This can be one of "UTR5", "UTR3", or "CDS".

Value

The ptn_motifSelection function returns a character vector of de novo identified motifs in the gene sets of interest that have enrichment p-values below the threshold set with the stremeThreshold parameter of the motifAnalysis function.

References

Timothy L. Bailey, James Johnson, Charles E. Grant, William S. Noble, "The MEME Suite", Nucleic Acids Research, 43(W1):W39-W49, 2015.

Timothy L. Bailey, "STREME: Accurate and versatile sequence motif discovery", Bioinformatics, 2021. https://doi.org/10.1093/bioinformatics/btab203

See Also

memes
runStreme
motifAnalysis

Examples

```
# Note that as users must provide the path to the MEME Suite executables, it is not possible to
# create a fully executable example as this path will vary depending on many factors. An example of
# how to run this function is provided below, and can be updated with the correct memePath argument.

## Not run:
# load example data:
 data("postNetExample", package = "postNet")
 ptn <- postNetExample$ptn

 ptn <- motifAnalysis(ptn = ptn,
                      streme Threshold = 0.05,
                      minwidth = 6,
                      memePath = "/meme/bin",
                      region = c('UTR5'),
                      subregion = NULL,
                      subregionSel = NULL)

# Extract the significantly enriched motifs:
 myMotifs <- ptn_motifSelection(ptn = ptn,
                                region = 'UTR5')

 str(myMotifs)

## End(Not run)
```

ptn_networkGraph *Accessor for the* networkGraph *slot of a* postNetData *object*

Description

Retrieve the network (an S3 igraph object) resulting from stepwise regression modelling from the postNetData object.

Usage

```
## S4 method for signature 'postNetData'
ptn_networkGraph(ptn, comparison)
```

Arguments

ptn	A postNetData object.
comparison	An integer specifying which comparison to select results from. For example, 1 would select the first comparison specified in the comparisons parameter of the featureIntegration function.

Value

An S3 igraph object based on the results of stepwise regression modelling with featureIntegration.

See Also

featureIntegration
igraph

Examples

```
 tmp <- tempfile(fileext = ".pdf")

# load and create example data:
 data("postNetExample", package = "postNet")
 ptn <- postNetExample$ptn

# Prepare the list of pre-calculated features to be used in feature integration modelling:
 myFeatures <- postNetExample$features
 str(myFeatures)

# Run feature integration modelling using forward stepwise regression:
 ptn <- featureIntegration(ptn = ptn,
                           features = myFeatures,
                           pdfName = tmp,
                           regOnly = TRUE,
                           allFeat = FALSE,
                           analysis_type = "lm",
                           covarFilt = 20,
```

```
                        comparisons = list(c(1,2)),
                        NetModelSel = "omnibus")

# Retrieve the network graph of the results of the forward stepwise regression omnibus model:
 networkGraph <- ptn_networkGraph(ptn, comparison = 1)
 str(networkGraph)
```

`ptn_selectedFeatures` *Accessor for the* `selectedFeatures` *slot of a* `postNetData` *object*

Description

Retrieve the features explaining changes in the regulatory effect identified by stepwise regression or random forest models from the `postNetData` object.

Usage

```
ptn_selectedFeatures(ptn,
                     analysis_type,
                     comparison)
```

Arguments

`ptn`	A postNetData object.
`analysis_type`	A string specifying the method of analysis used for featureIntegration to retrieve the selected features from. The options are `"lm"` to retrieve features selected by stepwise linear regression, or `"rf"` for features selected using random forest-based analyses.
`comparison`	An integer specifying which comparison to select results from. For example, `1` would select the first comparison specified in the `comparisons` parameter of the featureIntegration function.

Details

The relationships between selected features and changes in the regulatory effect can be further explored using the plotFeaturesMap function, which uses UMAPs to visualize regulatory effects and features across genes.

The selected features can also be quantified in other gene lists or datasets and used to predict regulation using random forest classification, implemented with the rfPred function.

Value

If `analysis_type = "lm"` a named numeric vector of the features selected by stepwise regression modelling will be returned. Selected features are those passing the significance defined using the `stepP` parameter of the featureIntegration function. Numeric values correspond to the proportion of variance in the regulatory effect explained by each feature (from either omnibus, or adjusted models).

If `analysis_type = "rf"`, the selected features identified by Boruta and used in random forest classification will be returned, with values corresponding to feature importance calculated by randomForest.

References

Liaw A, Wiener M (2002). "Classification and Regression by randomForest." R News, 2(3), 18-22. https://CRAN.R-project.org/doc/Rnews/.

Witold R. Rudnicki M (2010). "Feature Selection with the Boruta Package." Journal of Statistical Software, 36(11), 1–13. https://doi.org/10.18637/jss.v036.i11.

See Also

featureIntegration
plotFeaturesMap
rfPred
Boruta
randomForest

Examples

```
 tmp <- tempfile(fileext = ".pdf")

# load and create example data:
 data("postNetExample", package = "postNet")
 ptn <- postNetExample$ptn

# Prepare the list of pre-calculated features to be used in feature integration modelling:
 myFeatures <- postNetExample$features
 str(myFeatures)

# Run feature integration modelling using forward stepwise regression:
 ptn <- featureIntegration(ptn = ptn,
                           features = myFeatures,
                           pdfName = tmp,
                           regOnly = TRUE,
                           allFeat = FALSE,
                           analysis_type = "lm",
                           covarFilt = 20,
                           comparisons = list(c(1, 2)),
                           NetModelSel = "omnibus")

# Retrieve the features selected by the model that explain changes in translation efficiency:
 selectedFeatures <- ptn_selectedFeatures(ptn,
                                          analysis_type = "lm",
                                          comparison = 1)
 str(selectedFeatures)
```

ptn_selection *Accessor for the isoform selection parameter from the "selection" slot of a* postNetData *object*

Description

Retrieve the "selection" slot from a postNetData object, which holds the method chosen to select which mRNA isoforms are used in analyses. These can be one of: "longest", "shortest", or "random".

Usage

```
## S4 method for signature 'postNetData'
ptn_selection(ptn)
```

Arguments

ptn	A postNetData object.

Value

A character specifying the method used to select mRNA isoforms. Can be either "longest", "shortest", or "random".

See Also

postNetStart for details on the different mRNA isoform selection options.
postNetData-class

Examples

```
# load and create example data:
 data("postNetExample", package = "postNet")
 ptn <- postNetExample$ptn

# Access mRNA isoform selection method:
  ptn_selection(ptn)
```

ptn_sequences *Accessor for the reference annotation sequences in the sequence annotation slots of a* postNetData *object*

Description

The "postNetAnnot" slot holds the reference sequence annotations as postNetRegion lists. The annotations for the different sequence regions (5'UTR, CDS, 3'UTR, and optionally CCDS) are lists of three character vectors ("id", "geneID", and "sequence"). The ptn_sequences function retrieves sequences from the specified mRNA region.

Usage

```
## S4 method for signature 'postNetData'
ptn_sequences(ptn, region)
```

Arguments

ptn	A postNetData object.
region	The sequence region to be accessed. Can be either: "UTR5", "CDS", or "UTR3".

Value

A character vector containing the reference annotation sequences corresponding to the specified region.

See Also

postNetStart for more information on obtaining reference sequence annotations from different sources.
postNetData-class

Examples

```
# load and create example data:
 data("postNetExample", package = "postNet")
 ptn <- postNetExample$ptn

# Access the 5'UTR sequences from the postNetData object
  UTR5seqs <- ptn_sequences(ptn, "UTR5")
  str(UTR5seqs)
```

ptn_species *Accessor for the "species" slot of a* postNetData *object*

Description

Retrieve the "species" slot from a postNetData object, which holds the source species of the RefSeq sequence annotations. Currently, "human" and "mouse" are available if source is "load", "create", "createFromSourceFiles", or "createFromFasta".

Usage

```
## S4 method for signature 'postNetData'
ptn_species(ptn)
```

Arguments

ptn	A postNetData object.

Value

A character specifying the species of origin for the RefSeq annotations used to generate the reference sequences.

See Also

postNetStart for details on available species, and providing custom annotations.
postNetData-class

Examples

```
# load and create example data:
 data("postNetExample", package = "postNet")
 ptn <- postNetExample$ptn

# Access species of origin for the RefSeq annotation used to create the reference sequences
  ptn_species(ptn)
```

ptn_version *Accessor for the RefSeq annotation release version from the "version" slot of a* postNetData *object*

Description

Retrieve the "version" slot from a postNetData object, which holds the release version number of the RefSeq annotations used to generate the reference sequences. Currently, this only applies when source = 'load' in the postNetStart function.

Usage

```
## S4 method for signature 'postNetData'
ptn_version(ptn)
```

Arguments

ptn	A postNetData object.

Value

A character specifying the release version of the RefSeq annotations used to generate the reference sequences.

See Also

postNetStart for details on selecting the appropriate reference sequence annotation versions.
postNetData-class

Examples

```
# load and create example data:
 data("postNetExample", package = "postNet")
 ptn <- postNetExample$ptn

# Access version of the RefSeq annotation used to create the reference sequences
  ptn_version(ptn)
```

---

rfPred *Predict post-transcriptional regulation using trained random forest models*

Description

After identifying features associated with changes in post-transcriptional regulation and training a random forest classification model using the featureIntegration function, the rfPred function can then be used to apply this model to predict post-transcriptional regulation in new gene lists or datasets based on the same set of features. In this way, it is possible to evaluate how well a set of features identified as being associated with post-transcriptional regulation in one context can explain regulation observed in another context.

Usage

```
rfPred(ptn,
      comparison,
      predGeneList,
      predFeatures,
      pdfName = NULL)
```

Arguments

ptn	A postNetData object after running the random forest implementation of featureIntegration, containing the trained final random forest model.
comparison	An integer specifying which comparison to select the model from. For example, 1 would select the first comparison specified in the comparisons parameter of the featureIntegration function.
predGeneList	A named list of character vectors corresponding to the new gene lists for which regulation will be predicted. The list must have two gene lists corresponding to the same classes in the predictive model trained on the original data. For example, if the random forest model from featureIntegration was trained to classify genes into activated and suppressed categories, then the input list of genes to make predictions on should correspond to the same categories in the new dataset. Ensure that the gene IDs provided match those in the reference sequence annotations.
predFeatures	A named list of numeric vectors corresponding to features that will be used with the model trained on the original data to predict regulatory effects in the new data. Each vector element in the list must have names corresponding to gene IDs. These features must be enumerated in the new gene lists or dataset that is being predicted on, and must include the same features used in the final random forest model trained on the original dataset.
pdfName	Name to be appended to output PDF files.

Details

The rfPred function applies a trained Random Forest classification model generated by featureIntegration to an independent set of genes. Prediction is performed using the same feature definitions (enumerated in the new data set) and classes as the original model, enabling assessment of how well features explaining post-transcriptional regulation in one dataset generalize to another context. Model performance is summarized using ROC analysis, providing a quantitative measure of cross-dataset predictive power.

Value

The output of rfPred is a PDF file with the Receiver Operating Characteristic (ROC) curve illustrating the performance of the trained model in predicting regulatory classes in the new dataset.

References

Sing T, Sander O, Beerenwinkel N, Lengauer T (2005). "ROCR: visualizing classifier performance in R." Bioinformatics, 21(20), 7881. http://rocr.bioinf.mpi-sb.mpg.de.

Liaw A, Wiener M (2002). "Classification and Regression by randomForest." R News, 2(3), 18-22. https://CRAN.R-project.org/doc/Rnews/.

See Also

featureIntegration
randomForest
performance

Examples

```
 tmp <- tempfile(fileext = ".pdf")

# load and create example data:
 data("postNetExample", package = "postNet")
 ptn <- postNetExample$ptn

# Prepare the list of pre-calculated features to be used in feature integration modelling
 myFeatures <- postNetExample$features
 str(myFeatures)

# Run feature integration modelling using random forest classification to select features and
# generate the final model that will be used to predict regulation in a new gene list.
 ptn <- featureIntegration(ptn = ptn,
                           features = myFeatures,
                           pdfName = tmp,
                           analysis_type ="rf",
                           comparisons = list(c(1, 2)))

# Simulate a new gene list by randomly sampling genes from the background.
 newGenes <- sample(postNetExample$background, size = 20)
 newGeneList <- list(translationUp = newGenes[1:10], translationDown = newGenes[11:20])

# Select the features that were used to train the final random forest model in the original dataset.
 predFeatureNames <- names(ptn_selectedFeatures(ptn,
                                                analysis_type = "rf",
                                                comparison = 1))

# Prepare the predictive features. In this case the same list of input features
# will be used to predict as the new gene lists are taken from
# the same dataset the model was trained on. However, usually the input
# for the rfPred function would be taken from a postNetData object
# from a separate analysis on a distinct dataset.

 newFeatures <- myFeatures[predFeatureNames]

 ptn <- rfPred(ptn = ptn,
               comparison = 1,
               predGeneList = newGeneList,
               predFeatures = newFeatures,
               pdfName = tmp )
```

`signaturesHeatmap` *Generate heatmaps to evaluate the regulation of gene signatures in a* `postNetData` *object*

Description

The `signaturesHeatmap` function produces a heatmap of the regulation of gene signatures of interest in a `postNetData` object. Regulation of a gene signature of interest in the data set can be

evaluated using the FDR of a Wilcoxon Rank Sum test comparing the regulatory effect measurement for the gene signature against the background. Alternatively, the difference in the regulatory effect measurement for the genes in the signature compared to background can also be visualized at percentiles.

Usage

```
signaturesHeatmap(ptn,
                  signatureList,
                  unit = "FDR",
                  pdfName = NULL)
```

Arguments

ptn	A postNetData object.
signatureList	A named list of vectors containing gene IDs for the signatures of interest to be examined. Note that several signatures of translational regulation are provided with the package for both human and mouse. These can be retrieved using the get_signatures function.
unit	A string specifying the unit to be plotted in the heatmap. The options are "FDR", or any percentile in format p with a number (for example p25, p75, etc.).
pdfName	Name to be appended to output PDF files.

Details

When unit = 'FDR', the values display in the heatmap are obtained from a two-sided Wilcoxon Rank Sum test comparing the regulatory effect measurement values for the gene signature of interest against the background. See postNetStart for details on background gene sets. P-values are then corrected for multiple testing using the Benjamini & Hochberg method, and the -log10 FDR value is multiplied by either 1 or -1 corresponding to up- or down-regulation of the gene signature compared to background, respectively.

When unit = 'p75' (or any other percentile value), the values displayed in the heatmap are obtained by taking the difference between the Empirical Cumulative Distribution Function (ecdf) of the regulatory effect measurements for the gene signature and the background, at the percentile specified.

Value

No value is returned. Graphical outputs are generated as PDF files.

See Also

plotSignatures
plotSignatures_ads

Examples

```
 tmp <- tempfile(fileext = ".pdf")

# load example data:
 data("postNetExample", package = "postNet")
 ptn <- postNetExample$ptn

# load signature data:
 signatures <- get_signatures("human")

# Examine the regulation of various transcripts
# sensitive to translational regulation by various pathways and factors
 signaturesHeatmap(ptn,
                   signatureList = signatures,
                   unit = 'FDR',
                   pdfName = tmp)
```

`signCalc` *Create gene signatures for use in feature integration analysis*

Description

The `signCalc` function converts lists of gene IDs into signatures that are compatible as inputs for downstream featureIntegration models.

Usage

```
signCalc(ptn,
        signatures)
```

Arguments

`ptn`	A postNetData object.
`signatures`	A named list of character vectors containing gene IDs to be converted into signatures that can be used as input with the featureIntegration function.

Value

Named list of vectors indicating inclusion in different signatures for all genes in the `postNetData` object. Genes in the input signature that are included in the `postNetData` object are encoded as 1, and those that are absent are encoded as a 0.

See Also

featureIntegration

Examples

```
# load example data:
 data("postNetExample", package = "postNet")
 ptn <- postNetExample$ptn

# Create signatures of interest to be used in featureIntegration models
 addSign <- list()
 addSign[["TOP_mRNAs"]] <- c("RPL4", "RPL5","EEF1A1","EIF3D")
 addSign[["Hypoxia_Induced"]] <- c("VEGFA","LOX","ENO1","PDK1","PGK1","HIF1")

 mySign <- signCalc(ptn = ptn, signatures = addSign)
 str(mySign)
```

`slopeFilt` *Filtering* **anota2seq** *analysis results according to slopes for use in GSEA, GAGE, and GO term analyses*

Description

The `slopeFilt` function identifies the genes with unrealistic models of changes in translation efficiency according to the slopes fitted in the per-gene **anota2seq** analysis of partial variance (APV) models. These genes are then filtered out of the analysis by providing the output of `slopeFilt` to the `genesSlopeFiltOut` argument of the `gseaAnalysis`, `gseaPlot`, `gageAnalysis`, and `goAnalysis` functions. See the **anota2seq** vignette for additional details on slope filtering.

Usage

```
slopeFilt(ads,
          regulationGen,
          contrastSel,
          minSlope = NULL,
          maxSlope = NULL)
```

Arguments

`ads`	An S4 object of class `Anota2seqDataSet` (resulting from the `anota2seqRun` function in the **anota2seq** package).
`regulationGen`	The regulation effect measurement that will be taken from `ads`. This parameter can be either "translation" or "buffering". Note that slope filtering is not relevant for changes in mRNA abundance.
`contrastSel`	The contrast in `ads` where the regulation effect measurement corresponding to `regulationGen` should be extracted from. This must be a single numeric value.
`minSlope`	The minimum threshold to filter genes whose calculated slopes in anota2seq APV models are too small to be realistic. If `regulationGen = "translation"`, the recommended threshold is -1 (i.e., excludes genes with a slope < (-1)). If `regulationGen = "buffering"`, the recommended threshold is -2 (i.e., excludes genes with a slope < (-2)). Default is `NULL`, which will automatically apply the recommended filtering threshold for the selected `regulationGen`.

`maxSlope` The maximum threshold to filter genes whose calculated slopes in anota2seq APV models are too large to be realistic. If `regulationGen = "translation"`, the recommended threshold is 2 (i.e., excludes genes with a slope > 2). If `regulationGen = "buffering"`, the recommended threshold is 1 (i.e., excludes genes with a slope > 1). Default is `NULL`, which will automatically apply the recommended filtering threshold for the selected `regulationGen`.

Details

When performing GSEA, GAGE, or GO term analysis using the output of an **anota2seq** analysis, it is often necessary to filter the input genes and log2 fold changes for the `"translation"` and `"buffering"` regulatory modes prior to performing enrichment analyses. This is because the slopes fitted by the anota2seq APV models can sometimes have unrealistic values, or suggest unlikely translational regulation, impacting the analysis of changes in translation or translational buffering (or offsetting). Filtering out genes with these unrealistic slopes is especially important for GSEA and GAGE analyses, which rely on rankings. For analyses relying on hypergeometric tests, such as GO term enrichment, the impact of filtering on the analysis is likely to be more negligible. However, slope filtering is still recommended. Note that in high-quality data sets, usually few genes will require slope filtering.

Value

A character vector of the gene identifiers that will be excluded from downstream enrichment analyses based on the selected slope filtering thresholds.

See Also

anota2seqSelSigGenes
anota2seqRun
gseaAnalysis
gseaPlot
gageAnalysis
goAnalysis
miRNAanalysis

Examples

```
local({
  oldwd <- getwd()
  on.exit(setwd(oldwd), add = TRUE)
  setwd(tempdir())
  # load example data:
  data("postNetExample", package = "postNet")

  # Initialize Anota2seqDataSet (see anota2seq vignette for details)
  ads <- anota2seq::anota2seqDataSetFromMatrix(
    dataP = postNetExample$ads_data$dataP,
    dataT = postNetExample$ads_data$dataT,
    phenoVec = postNetExample$ads_data$phenoVec,
    batchVec = c(1, 2, 3, 4, 1, 2, 3, 4),
```

```
  dataType = "RNAseq",
  normalize = FALSE)

# Run an anota2seq analysis:
# Note that the quality control and residual outlier testing are not
# performed to limit the running time of this example. For full details
# on running an analysis please see the anota2seq vignette and help manual.
ads <- anota2seq::anota2seqRun(ads,
  performQC = FALSE,
  performROT = FALSE,
  useProgBar = FALSE)

# Get the genes to be filtered out of downstream enrichment analyses
# using the buffering regulatory mode:
filtOutGenes <- slopeFilt(ads,
                regulationGen = "buffering",
                contrastSel = 1)

str(filtOutGenes)
})
```

`uorfAnalysis` *Detect and quantify upstream open reading frames (uORFs)*

Description

The `uorfAnalysis` function detects the presence and position of uORFs in 5'UTRs. Options are available to examine uORFs with both canonical and non-canonical start codons, as well as different Kozak contexts. In addition, the proportions of mRNA transcripts with uORFs can be plotted, and statistical comparisons can be made between gene sets of interest against background, and/or other gene sets.

Usage

```
uorfAnalysis(ptn,
            startCodon = "ATG",
            KozakContext = "strong",
            onlyUTR5 = FALSE,
            unitOut = "number",
            comparisons = NULL,
            plotOut = TRUE,
            pdfName = NULL)
```

Arguments

`ptn`	A postNetData object.
`startCodon`	A string specifying the start codon. Only uORFs with the specified start codon will be detected. The default is the canonical start codon, `"ATG"`.

KozakContext	A string to select the Kozak context. The options are: "strong" ([AG]..startCodonG), "adequate1" ([AG]..startCodon), "adequate2" (...startCodonG), "weak" (...startCodon.), or "any" (the nucleotide context of the start codon is disregarded). The default is "strong".
onlyUTR5	A logical to select whether the uORFs detected should be completely contained in the 5'UTR, (i.e., the stop codons of the uORFs occur before the start of the CDS). If TRUE, uORFs must be completely 5'UTR-contained to be detected. If FALSE, only the start codon of the uORF must be in the 5'UTR, with the possibility that the stop codon may be in the CDS or 3'UTR. The default is FALSE.
unitOut	A string to specify whether the output should be the "number" of uORFs detected per gene, or the "position" of the detected uORFs in the mRNA sequence (nucleotide position of the start and stop). Note that only the output with "number" can be used in the downstream analysis with the featureIntegration function. The default is "number".
comparisons	A list of numeric vectors specifying pairwise comparisons between gene sets defined in the regulation or geneList slots of the postNetData object. Use 0 to denote the background set. For example, list(c(0,2), c(1,2)) compares gene set 2 to the background and gene set 1 to gene set 2. The default is NULL.
plotOut	Logical indicating whether PDF files of plots are generated. If FALSE, only the values specified with unitOut will be returned, and statistical comparisons between gene sets will not be performed. Note that plots will only be generated if unitOut = "number". The default is TRUE.
pdfName	Name to be appended to output PDF files.

Details

A two-sided Wilcoxon Rank Sum test is performed to identify significant differences in uORFs detected between gene sets of interest, or against the background gene set.

Value

If the unitOut = "number", the output will be a named list of vectors with the number of uORFs detected in a particular Kozak context for each gene. This list can be used with the downstream featureIntegration function. If the unitOut = "position", the output will be a named list of lists with the start and end positions of each uORF for each gene. These positions cannot be used in featureIntegration. The uorfAnalysis function can also return PDF files of output plots with statistical comparisons between gene sets of interest when unitOut = "number".

Examples

```
 tmp <- tempfile(fileext = ".pdf")

# load example data:
 data("postNetExample", package = "postNet")
 ptn <- postNetExample$ptn

# Identify uORFs with canonical start codons in a strong Kozak context,
```

```
# fully contained within 5'UTRs, and compare between
# translationUp vs. translationDown genes:

 uORFs <- uorfAnalysis(ptn = ptn,
                       comparisons = list(c(1, 2)),
                       startCodon = "ATG",
                       KozakContext = c("strong"),
                       onlyUTR5 = TRUE,
                       unitOut = "number",
                       plotOut = TRUE,
                       pdfName = tmp)

 str(uORFs)
```

Index

∗ **Annotation**
postNet-package, 3
∗ **FeatureExtraction**
postNet-package, 3
∗ **GeneExpression**
postNet-package, 3
∗ **GeneRegulation**
postNet-package, 3
∗ **Network**
postNet-package, 3
∗ **RNASeq**
postNet-package, 3
∗ **RiboSeq**
postNet-package, 3
∗ **Sequencing**
postNet-package, 3
∗ **Transcriptomics**
postNet-package, 3
∗ **classes**
postNetData-class, 63
∗ **datasets**
get_signatures, 28
humanSignatures, 42
mouseSignatures, 53
postNetExample, 65
postNetVignette, 77

Anota2seqDataSet-class, *69*
anota2seqRun, *61, 62*, *69*, *74*, *114, 115*
anota2seqSelSigGenes, *115*

Boruta, *17*, *20–22*, *99*, *105*

clusterProfiler, *31, 32*, *92*
codonCalc, 4, *8, 9*, *20*, *82, 83*
codonUsage, *4, 5*, 6, *65*, *81–83*
contentAnalysis, 10, *20*
contentMotifs, 12, *20*, *52*

featureIntegration, *4, 5*, *12–14*, 16, *24*, *28*, *43*, *48*, *52*, *54*, *57, 58*, *63–65*, *68*, *70*, *74*, *80*, *86, 87*, *98, 99*, *103–105*, *109, 110*, *113*, *117*
fgsea, *36*, *38*, *41, 42*, *93*
fgseaMultilevel, *37*, *41*, *93*
foldingEnergyAnalysis, *20*, 22

gage, *25, 26*, *47–49*, *88*, *96, 97*
gageAnalysis, 25, *65*, *87, 88*, *114, 115*
get_signatures, 28, *59–61*, *112*
goAnalysis, 31, *34, 35*, *65*, *91, 92*, *114, 115*
goDotplot, *32*, 34, *92*
gseaAnalysis, 36, *42*, *65*, *92, 93*, *114, 115*
GSEABase, *38*
gseaPlot, *37, 38*, 40, *93*, *114, 115*

humanSignatures, *20*, 42

igraph, *21*, *103*
importance, *20*

lengthAnalysis, *20*, 45

memes, *52*, *101, 102*
miRNAanalysis, 47, *65*, *95–97*, *115*
motifAnalysis, *12–14*, 51, *65*, *101, 102*
mouseSignatures, *20*, 53
msigdb, *38*

performance, *20*, *22*, *110*
plotFeaturesMap, *17*, *22*, 56, *65*, *104, 105*
plotSignatures, *28*, *31*, *43*, *54*, 58, *62*, *112*
plotSignatures_ads, *28*, *31*, *43*, *54*, *60*, 60, *112*
postNet *(*postNet-package*)*, 3
postNet-package, 3
postNetData, *4*, *6*, *11*, *13*, *17*, *23*, *25*, *31*, *35, 36*, *41*, *46, 47*, *51*, *57*, *59*, *73, 74*, *79–82*, *84*, *86*, *88, 89*, *91*, *93–95*, *97, 98*, *100*, *102–104*, *106–110*, *112, 113*, *116*
postNetData-class, 63

postNetExample, 65
postNetStart, *25*, *32*, *36*, *38*, *63*, *65*, 68, *79*, *84–86*, *88–90*, *92*, *94*, *96, 97*, *106–109*, *112*
postNetVignette, 77
pqsfinder, *12–14*
ptn_background, *65*, *74*, 79
ptn_background,postNetData-method *(*ptn_background*)*, 79
ptn_check_models, *22*, *65*, 80
ptn_codonAnalysis, *5*, *8, 9*, *65*, *74*, 81
ptn_codonAnalysis,postNetData-method *(*ptn_codonAnalysis*)*, 81
ptn_codonSelection, *4, 5*, *8, 9*, *65*, *74*, *81*, 82
ptn_codonSelection,postNetData-method *(*ptn_codonSelection*)*, 82
ptn_colours, *65*, *74*, 83
ptn_colours,postNetData-method *(*ptn_colours*)*, 83
ptn_dataIn, *65*, *74*, 84
ptn_dataIn,postNetData-method *(*ptn_dataIn*)*, 84
ptn_effect, *65*, *74*, 85
ptn_effect,postNetData-method *(*ptn_effect*)*, 85
ptn_features, *21, 22*, *65*, *74*, 86
ptn_features,postNetData-method *(*ptn_features*)*, 86
ptn_GAGE, *25, 26*, *65*, *74*, 87
ptn_geneID, *65*, *74*, 89
ptn_geneID,postNetData-method *(*ptn_geneID*)*, 89
ptn_geneList, *65*, *74*, 90
ptn_geneList,postNetData-method *(*ptn_geneList*)*, 90
ptn_GO, *32*, *35*, *65*, *74*, 91
ptn_GSEA, *37, 38*, *42*, *65*, *74*, 92
ptn_id, *65*, *74*, 94
ptn_id,postNetData-method *(*ptn_id*)*, 94
ptn_miRNA_analysis, *48, 49*, *65*, *74*, 95, *97*
ptn_miRNA_to_gene, *48, 49*, *65*, *74*, *96*, 97
ptn_model, *21, 22*, *65*, *74*, *80*, 98
ptn_motifGeneList, 100
ptn_motifgeneList, *52*, *65*, *74*
ptn_motifGeneList,postNetData-method *(*ptn_motifGeneList*)*, 100
ptn_motifSelection, *13*, *52*, *65*, *74*, 101
ptn_motifSelection,postNetData-method *(*ptn_motifSelection*)*, 101
ptn_networkGraph, *21, 22*, *65*, 103
ptn_networkGraph,postNetData-method *(*ptn_networkGraph*)*, 103
ptn_selectedFeatures, *21, 22*, *65*, *74*, 104
ptn_selection, *65*, *74*, 106
ptn_selection,postNetData-method *(*ptn_selection*)*, 106
ptn_sequences, *65*, *74*, 107
ptn_sequences,postNetData-method *(*ptn_sequences*)*, 107
ptn_species, *65*, *74*, 108
ptn_species,postNetData-method *(*ptn_species*)*, 108
ptn_version, *65*, *74*, 108
ptn_version,postNetData-method *(*ptn_version*)*, 108

randomForest, *17*, *20–22*, *99*, *105*, *110*
rfPred, *17*, *22*, *104, 105*, 109
runFimo, *14*
runStreme, *52*, *101, 102*

signaturesHeatmap, *28*, *31*, *43*, *54*, *60*, *62*, 111
signCalc, *17*, *20*, *28*, *31*, *86, 87*, 113
slopeFilt, *25, 26*, *31, 32*, *36–38*, *41*, *47, 48*, *92*, 114

umap, *56*, *58*
universalmotif, *52*, *64*, *100, 101*
uorfAnalysis, *20*, 116