Package ‘GRaNIE’

March 25, 2024

Title GRaNIE: Reconstruction cell type specific gene regulatory networks including enhancers using chromatin accessibility and RNA-seq data

Version 1.6.1

Encoding UTF-8

Description Genetic variants associated with diseases often affect non-coding regions, thus likely having a regulatory role. To understand the effects of genetic variants in these regulatory regions, identifying genes that are modulated by specific regulatory elements (REs) is crucial. The effect of gene regulatory elements, such as enhancers, is often cell-type specific, likely because the combinations of transcription factors (TFs) that are regulating a given enhancer have celltype specific activity. This TF activity can be quantified with existing tools such as diffTF and captures differences in binding of a TF in open chromatin regions. Collectively, this forms a gene regulatory network (GRN) with cell-type and data-specific TF-RE and RE-gene links. Here, we reconstruct such a GRN using bulk RNAseq and open chromatin (e.g., using ATACseq or ChIPseq for open chromatin marks) and optionally TF activity data. Our network contains different types of links, connecting TFs to regulatory elements, the latter of which is connected to genes in the vicinity or within the same chromatin domain (TAD). We use a statistical framework to assign empirical FDRs and weights to all links using a permutation-based approach.

Imports futile.logger, checkmate, patchwork, reshape2, data.table, matrixStats, Matrix, GenomicRanges, RColorBrewer, ComplexHeatmap, DESeq2, circlize, progress, utils, methods, stringr, scales, igraph, S4Vectors, ggplot2, rlang, Biostrings, GenomeInfoDb (>= 1.34.8), SummarizedExperiment, forcats, gridExtra, limma, tidyselect, readr, grid, tidyr, dplyr, stats, grDevices, graphics, magrittr, tibble, viridis, colorspace, biomaRt, topGO, AnnotationHub, ensembldb

Depends R (>= 4.2.0)

Suggests knitr, BSgenome.Hsapiens.UCSC.hg19, BSgenome.Hsapiens.UCSC.hg38, BSgenome.MusMusculus.UCSC.mm10, BSgenome.MusMusculus.UCSC.mm9, BSgenome.Rnorvegicus.UCSC.rn6, BSgenome.Rnorvegicus.UCSC.rn7, BSgenome.Dmela disgust.UCSC.dm6, BSgenome.Mmulatta.UCSC.rheMac10, TxDb.Hsapiens.UCSC.hg19.knownGene,
R topics documented:

- addConnections_peak_gene
- addConnections_TF_peak
R topics documented:

addData ........................................... 8
addData_TFActivity .............................. 10
addSNPData ...................................... 11
addTFBS .......................................... 13
add_featureVariation ............................ 15
add_TF_gene_correlation ...................... 17
AR_classification_wrapper .................... 18
build_eGRN_graph .............................. 19
calculateCommunitiesEnrichment .............. 20
calculateCommunitiesStats ..................... 23
calculateGeneralEnrichment .................... 24
calculateTFEnrichment ......................... 26
changeOutputDirectory ......................... 28
deleteIntermediateData ......................... 29
filterConnectionsForPlotting .................. 29
filterData ....................................... 30
filterGRNAndConnectGenes ..................... 32
generateStatsSummary ......................... 36
getCounts ....................................... 38
getGRNConnections ............................... 39
getGRNSummary .................................. 42
getParameters .................................... 43
getTopNodes ..................................... 43
GRaNIE ......................................... 44
GRN-class ....................................... 45
importTFData .................................... 46
initializeGRN .................................... 47
loadExampleObject .............................. 48
nGenes ......................................... 48
nPeaks .......................................... 49
nTFs ............................................. 50
overlapPeaksAndTFBS ............................ 51
performAllNetworkAnalyses ..................... 51
plotCommunitiesEnrichment .................... 54
plotCommunitiesStats ........................... 56
plotCorrelations ................................ 58
plotDiagnosticPlots_peakGene .................. 60
plotDiagnosticPlots_TFPeaks .................... 62
plotDiagnosticPlots_TFPeaks_GC .............. 63
plotGeneralEnrichment ......................... 65
plotGeneralGraphStats ......................... 66
plotPCA_all .................................... 68
plotTFEnrichment ............................... 69
plot_stats_connectionSummary .................. 72
visualizeGRN .................................... 73

Index ............................................. 76
**addConnections_peak_gene**

Add peak-gene connections to a GRN object

**Description**

After the execution of this function, QC plots can be plotted with the function `plotDiagnosticPlots_peakGene` unless this has already been done by default due to `plotDiagnosticPlots = TRUE`

**Usage**

```r
addConnections_peak_gene(
  GRN, 
  overlapTypeGene = "TSS", 
  corMethod = "pearson", 
  promoterRange = 250000, 
  TADs = NULL, 
  TADs_mergeOverlapping = FALSE, 
  shuffleRNACounts = TRUE, 
  nCores = 4, 
  plotDiagnosticPlots = TRUE, 
  plotGeneTypes = list(c("all"), c("protein_coding")), 
  outputFolder = NULL, 
  forceRerun = FALSE 
)
```

**Arguments**

- **GRN**: Object of class `GRN`
- **overlapTypeGene**: Character. "TSS" or "full". Default "TSS". If set to "TSS", only the TSS of the gene is used as reference for finding genes in the neighborhood of a peak. If set to "full", the whole annotated gene (including all exons and introns) is used instead.
- **corMethod**: Character. One of `pearson`, `spearman` or `bicor`. Default `pearson`. Method for calculating the correlation coefficient. For `pearson` and `spearman`, see `cor` for details. `bicor` denotes the *biweight midcorrelation*, a correlation measure based on medians as calculated by `WGCNA::bicorAndPvalue`. Both `spearman` and `bicor` are considered more robust measures that are less prone to be affected by outliers.
- **promoterRange**: Integer >=0. Default 250000. The size of the neighborhood in bp to correlate peaks and genes in vicinity. Only peak-gene pairs will be correlated if they are within the specified range. Increasing this value leads to higher running times and more peak-gene pairs to be associated, while decreasing results in the opposite.
addConnections_peak_gene

TADs Data frame with TAD domains. Default NULL. If provided, the neighborhood of a peak is defined by the TAD domain the peak is in rather than a fixed-sized neighborhood. The expected format is a BED-like data frame with at least 3 columns in this particular order: chromosome, start, end, the 4th column is optional and will be taken as ID column. All additional columns as well as column names are ignored. For the first 3 columns, the type is checked as part of a data integrity check.

TADs_mergeOverlapping TRUE or FALSE. Default FALSE. Should overlapping TADs be merged? Only relevant if TADs are provided.

shuffleRNACounts TRUE or FALSE. Default TRUE. Should the RNA sample labels be shuffled in addition to testing random peak-gene pairs for the background? When set to FALSE, only peak-gene pairs are shuffled, but for each pair, the counts from peak and RNA that are correlated are matched (i.e., sample 1 counts from peak data are compared to sample 1 counts from RNA). If set to TRUE, however, the RNA sample labels are in addition shuffled so that sample 1 counts from peak data are compared to sample 4 data from RNA, for example. Shuffling truly randomizes the resulting background eGRN. Note that this parameter and its influence is still being investigated. Until version 1.0.7, this parameter (although not existent explicitly) was implicitly set to TRUE.

nCores Integer >0. Default 1. Number of cores to use. A value >1 requires the BiocParallel package (as it is listed under Suggests, it may not be installed yet).

plotDiagnosticPlots TRUE or FALSE. Default TRUE. Run and plot various diagnostic plots? If set to TRUE, PDF files will be produced and saved in the output directory (in a sub-folder called plots).

plotGeneTypes List of character vectors. Default list(c("all"), c("protein_coding")). Each list element may consist of one or multiple gene types that are plotted collectively in one PDF. The special keyword "all" denotes all gene types that are found (be aware: this typically contains 20+ gene types, see https://www.gencodegenes.org/pages/biotypes.html for details).

outputFolder Character or NULL. Default NULL. If set to NULL, the default output folder as specified when initiating the object in initializeGRN will be used. Otherwise, all output from this function will be put into the specified folder. If a folder is provided, while we recommend specifying an absolute path, a relative one also works.

forceRerun TRUE or FALSE. Default FALSE. Force execution, even if the GRN object already contains the result. Overwrites the old results.

Value An updated GRN object, with additional information added from this function.

See Also

plotDiagnosticPlots_peakGene
Examples

```r
# See the Workflow vignette on the GRaNIE website for examples
GRN = loadExampleObject()
GRN = addConnections_peak_gene(GRN, promoterRange=10000, plotDiagnosticPlots = FALSE)
```

---

**addConnections_TF_peak**

*Add TF-peak connections to a GRN object*

---

**Description**

After the execution of this function, QC plots can be plotted with the function `plotDiagnosticPlots_TFPeaks` unless this has already been done by default due to `plotDiagnosticPlots = TRUE`.

**Usage**

```r
addConnections_TF_peak(
  GRN,
  plotDiagnosticPlots = TRUE,
  plotDetails = FALSE,
  outputFolder = NULL,
  corMethod = "pearson",
  connectionTypes = c("expression"),
  removeNegativeCorrelation = c(FALSE),
  maxFDRToStore = 0.3,
  addForBackground = TRUE,
  useGCCorrection = FALSE,
  percBackground_size = 75,
  percBackground_resample = TRUE,
  forceRerun = FALSE
)
```

**Arguments**

- **GRN**: Object of class `GRN`
- **plotDiagnosticPlots**: `TRUE` or `FALSE`. Default `TRUE`. Run and plot various diagnostic plots? If set to `TRUE`, PDF files will be produced and saved in the output directory (in a sub-folder called plots).
- **plotDetails**: `TRUE` or `FALSE`. Default `FALSE`. Print additional plots that may help for debugging and QC purposes? Note that these plots are currently less documented or not at all.
- **outputFolder**: Character or `NULL`. Default `NULL`. If set to `NULL`, the default output folder as specified when initiating the object in `initializeGRN` will be used. Otherwise, all output from this function will be put into the specified folder. If a folder is provided, while we recommend specifying an absolute path, a relative one also works.
addConnections_TF_peak

**corMethod**  
Character. One of pearson, spearman or bicor. Default pearson. Method for calculating the correlation coefficient. For pearson and spearman, see cor for details. bicor denotes the *biweight midcorrelation*, a correlation measure based on medians as calculated by WGCNA::bicorAndPvalue. Both spearman and bicor are considered more robust measures that are less prone to be affected by outliers.

**connectionTypes**  
Character vector. Default expression. Vector of connection types to include for the TF-peak connections. If an additional connection type is specified here, it has to be available already within the object (EXPERIMENTAL). See the function addData_TFActivity for details.

**removeNegativeCorrelation**  
Vector of TRUE or FALSE. Default FALSE. EXPERIMENTAL. Must be a logical vector of the same length as the parameter connectionType. Should negatively correlated TF-peak connections be removed for the specific connection type? For connection type expression, the default is FALSE, while for any TF Activity related connection type, we recommend setting this to TRUE.

**maxFDRToStore**  
Numeric[0,1]. Default 0.3. Maximum TF-peak FDR value to permanently store a particular TF-peak connection in the object? This parameter has a large influence on the overall memory size of the object, and we recommend not storing connections with a high FDR due to their sheer number.

**addForBackground**  
TRUE or FALSE. Default TRUE. Add connections also for background data. Leave at TRUE unless you know what you are doing.

**useGCCorrection**  
TRUE or FALSE. Default FALSE. EXPERIMENTAL. Should a GC-matched background be used when calculating FDRs? For more details, see the Package Details vignette.

**percBackground_size**  
Numeric[0,100]. Default 75. EXPERIMENTAL. Percentage of the background to use as basis for sampling. If set to 0, an automatic iterative procedure will identify the maximum percentage so that all relevant GC bins with a rel. frequency above 5% from the foreground can be matched. For more details, see the Package Details vignette. Only relevant if useGCCorrection is set to TRUE, ignored otherwise.

**percBackground_resample**  
TRUE or FALSE. Default TRUE. EXPERIMENTAL. Should resampling be enabled for those GC bins for which not enough background peaks are available?. For more details, see the Package Details vignette. Only relevant if useGCCorrection is set to TRUE, ignored otherwise.

**forceRerun**  
TRUE or FALSE. Default FALSE. Force execution, even if the GRN object already contains the result. Overwrites the old results.

**Value**

An updated GRN object, with additional information added from this function.
addData

### Description

This function adds both RNA and peak data to a GRN object, along with data normalization. In addition, and highly recommended, sample metadata can be optionally provided.

### Usage

```r
addData(
  GRN,  
  counts_peaks,  
  normalization_peaks = "DESeq2_sizeFactors",  
  idColumn_peaks = "peakID",  
  counts_rna,  
  normalization_rna = "limma_quantile",  
  idColumn_RNA = "ENSEMBL",  
  sampleMetadata = NULL,  
  additionalParams.l = list(),  
  allowOverlappingPeaks = FALSE,  
  keepOriginalReadCounts = FALSE,  
  EnsemblVersion = NULL,  
  genomeAnnotationSource = "AnnotationHub",  
  forceRerun = FALSE
)
```

### Arguments

- **GRN**: Object of class `GRN`
- **counts_peaks**: Data frame. No default. Counts for the peaks, with raw or normalized counts for each peak (rows) across all samples (columns). In addition to the count data, it must also contain one ID column with a particular format, see the argument `idColumn_peaks` below. Row names are ignored, column names must be set to the sample names and must match those from the RNA counts and the sample metadata table.

### See Also

`plotDiagnosticPlots_TFPeaks`

### Examples

```r
# See the Workflow vignette on the GRaNIE website for examples
GRN = loadExampleObject()
GRN = addConnections_TF_peak(GRN, plotDiagnosticPlots = FALSE, forceRerun = FALSE)
```
### addData

**normalization_peaks**
Character. Default DESeq2_sizeFactors. Normalization procedure for peak data. Must be one of limma_cyclicloess, limma_quantile, limma_scale, csaw_cyclicLoess_orig, csaw_TMM, EDASeq_GC_peaks, gcqn_peaks, DESeq2_sizeFactors, none.

**idColumn_peaks**
Character. Default peakID. Name of the column in the counts_peaks data frame that contains peak IDs. The required format must be chr:start-end", with chr denoting the abbreviated chromosome name, and start and end the begin and end of the peak coordinates, respectively. End must be bigger than start. Examples for valid peak IDs are chr1:400-800 or chrX:20-25.

**counts_rna**
Data frame. No default. Counts for the RNA-seq data, with raw or normalized counts for each gene (rows) across all samples (columns). In addition to the count data, it must also contain one ID column with a particular format, see the argument idColumn_rna below. Row names are ignored, column names must be set to the sample names and must match those from the RNA counts and the sample metadata table.

**normalization_rna**
Character. Default limma_quantile. Normalization procedure for peak data. Must be one of limma_cyclicloess, limma_quantile, limma_scale, csaw_cyclicLoess_orig, csaw_TMM, DESeq2_sizeFactors, none.

**idColumn_RNA**
Character. Default ENSEMBL. Name of the column in the counts_rna data frame that contains Ensembl IDs.

**sampleMetadata**
Data frame. Default NULL. Optional, additional metadata for the samples, such as age, sex, gender etc. If provided, the @seealso [plotPCA_all()] function can then incorporate and plot it. Sample names must match with those from both peak and RNA-Seq data. The first column is expected to contain the sample IDs, the actual column name is irrelevant.

**additionalParams.l**
Named list. Default list(). Additional parameters for the chosen normalization method. Currently, only the GC-aware normalization methods EDASeq_GC_peaks and gcqn_peaks are supported here. Both support the parameters roundResults (logical flag, TRUE or FALSE) and nBins (Integer > 0), and EDASeq_GC_peaks supports three additional parameters: withinLane_method (one of: "loess","median","upper","full") and betweenLane_method (one of: "median","upper","full"). For more information, see the EDASeq vignette.

**allowOverlappingPeaks**
TRUE or FALSE. Default FALSE. Should overlapping peaks be allowed (then only a warning is issued when overlapping peaks are found) or (the default) should an error be raised?

**keepOriginalReadCounts**
TRUE or FALSE. Default FALSE. Should the original read counts as provided to the function be kept in addition to storing the rad counts after a (if any) normalization? This increases the memory footprint of the object because 2 additional count matrices have to be stored.

**EnsemblVersion**
NULL or Character(1). Default NULL. The Ensembl version to use for genome annotation retrieval via biomart, which is only used to populate the gene annotation metadata that is stored in GRN@annotation$genes. By default (NULL),
the newest version is selected for the most recent genome assembly versions is used (see `biomaRt::listEnsemblArchives()` for supported versions). This parameter can override this to use a custom (older) version instead.

**genomeAnnotationSource**
AnnotationHub or biomaRt. Default AnnotationHub. Annotation source to retrieve genome annotation data from.

**forceRerun**
TRUE or FALSE. Default FALSE. Force execution, even if the GRN object already contains the result. Overwrites the old results.

### Details

If the ChIPseeker package is installed, additional peak annotation is provided in the annotation slot and a peak annotation QC plot is produced as part of peak-gene QC. This is fully optional, however, and has no consequences for downstream functions. Normalizing the data sensibly is very important. When quantile is chose, `limma::normalizeQuantiles` is used, which in essence does the following: Each quantile of each column is set to the mean of that quantile across arrays. The intention is to make all the normalized columns have the same empirical distribution. This will be exactly true if there are no missing values and no ties within the columns: the normalized columns are then simply permutations of one another.

### Value

An updated GRN object, with added data from this function (e.g., slots `GRN@data$peaks` and `GRN@data$RNA`)

### See Also

plotPCA_all

### Examples

# See the Workflow vignette on the GRaNIE website for examples
# library(readr)
# rna.df = read_tsv("https://www.embl.de/download/zaugg/GRaNIE/rna.tsv.gz")
# peaks.df = read_tsv("https://www.embl.de/download/zaugg/GRaNIE/peaks.tsv.gz")
# meta.df = read_tsv("https://www.embl.de/download/zaugg/GRaNIE/sampleMetadata.tsv.gz")
# GRN = loadExampleObject()
# We omit sampleMetadata = meta.df in the following line, becomes too long otherwise
# GRN = addData(GRN, counts_peaks = peaks.df, counts_rna = rna.df, forceRerun = FALSE)
Usage

addData_TFActivity(
  GRN,
  normalization = "cyclicLoess",
  name = "TF_activity",
  forceRerun = FALSE
)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GRN</td>
<td>Object of class GRN</td>
</tr>
<tr>
<td>normalization</td>
<td>Character. Default cyclicLoess. One of cyclicLoess, sizeFactors, quantile, or none. Normalization procedure. When set to cyclicLoess, the csaw package is required (as it is listed under Suggests, it may not be installed).</td>
</tr>
<tr>
<td>name</td>
<td>Name in object under which it should be stored. This corresponds to the connectionType afterwards that some functions iterate over.</td>
</tr>
<tr>
<td>forceRerun</td>
<td>TRUE or FALSE. Default FALSE. Force execution, even if the GRN object already contains the result. Overwrites the old results.</td>
</tr>
</tbody>
</table>

Value

An updated GRN object, with added data from this function (GRN$data$TFs[[name]] in particular, with name referring to the value of the name parameter).

Description

This function accepts a vector of SNP IDs (rsID), retrieves their genomic positions and overlaps them with the peaks to extend the peak metadata (‘GRN$data$peaks$count$metadata’) by storing the number, positions and rsids of all overlapping SNPs per peak (new columns starting with ‘SNP_’). Optionally, SNPs in LD with the user-provided SNPs can be identified using the LDlinkR package. Note that only SNPs in LD are associated with a peak for those SNPs directly overlapping a peak. That is, if a user-provided SNP does not overlap with any peak, neither the SNP itself nor any of the SNPs in LD will be associated with any peak, even if an LD SNP overlaps another peak. The results of are stored in GRN@annotation$SNPs (full, unfiltered table) and GRN@annotation$SNPs_filtered (filtered table), and rapid re-filtering is possible without re-querying the database (time-consuming).

Usage

addSNPData(
  GRN,
  SNP_IDs,
  EnsemblVersion = NULL,
)
addSNPData

add_SNPs_LD = FALSE,
requeryLD = FALSE,
population = "CEU",
r2d = "r2",
token = NULL,
filter = "R2 > 0.8",
forceRerun = FALSE
)

Arguments

GRN
Object of class `GRN`

SNP_IDs
Character vector. No default. Vector of SNP IDs (rsID) that should be integrated and overlapped with the peaks.

EnsemblVersion
NULL or Character(1). Default NULL. Only relevant if source is not set to custom, ignored otherwise. The Ensembl version to use for the retrieval of gene IDs from their provided database names (e.g., JASPAR) via biomaRt. By default (NULL), the newest version is selected for the most recent genome assembly versions is used (see biomaRt::listEnsemblArchives() for supported versions). This parameter can override this to use a custom (older) version instead.

add_SNPs_LD
TRUE or FALSE. Default FALSE. Should SNPs in LD with any of the user-provided SNPs that overlap a peak be identified and added to the peak? If set to TRUE, LDlinkR::LDproxy_batch will be used to identify SNPs in LD based on the user-provided SNP_IDs argument, a specific (set of) populations (argument population) and a value for r2d. The full table (stored in GRN@annotation$SNPs) is then subsequently filtered, see also the filter argument.

requeryLD
TRUE or FALSE. Default FALSE. Only applicable if add_SNPs_LD = TRUE and ignored otherwise. Should LDlinkR::LDproxy_batch be re-executed if already present? As this is very time-consuming, querying the database is only performed if this parameter is set to TRUE or if GRN@annotation$SNPs is missing.

population
Character vector. Default CEU. Only applicable if add_SNPs_LD = TRUE and ignored otherwise. Population code(s) from the 1000 Genomes project to be used for LDlinkR::LDproxy_batch. Multiple codes are allowed. For all valid codes, see https://ftp.1000genomes.ebi.ac.uk/vol1/ftp/README_populations.md

r2d
r2 or d. Default r2. Only applicable if add_SNPs_LD = TRUE and ignored otherwise. See the help of the function LDlinkR::LDproxy_batch for more details.

token
Character or NULL. Default NULL. Only applicable if add_SNPs_LD = TRUE and ignored otherwise. LDlink provided user token. Register for token at https://ldlink.nih.gov/?tab=apiaccess. Has to be done only once and is very simple and straightforward but unfortunately necessary. An example, non-functional token is 2c49a3b54g04.

filter
Character. Default R2 > 0.8. Only applicable if add_SNPs_LD = TRUE and ignored otherwise. Filter criteria for the output table as generated by LDlinkR::LDproxy_batch. dplyr::filter style is used to specify filters, and multiple filtering criteria can be used (e.g., R2 > 0.8 & MAF > 0.01). The filtered table is stored in GRN@annotation$SNPs_filtered. Note that re-filtering is quick without the need to re-query the database unless requeryLD = TRUE.
addTFBS

forceRerun TRUE or FALSE. Default FALSE. Force execution, even if the GRN object already contains the result. Overwrites the old results.

Details

'biomaRt' is used to retrieve genomic positions for the user-defined SNPs, which can take a long time depending on the number of SNPs provided. Similarly, querying the LDlink servers may take a long time.

Value

An updated GRN object, with additional information added from this function.

Examples

# See the Workflow vignette on the GRaNIE website for examples
GRN = loadExampleObject()
GRN = addSNPData(GRN, SNP_IDs = c("rs7570219", "rs6445264", "rs12067275"), forceRerun = FALSE)

addTFBS

Add TFBS to a GRN object.

Description

For this, a folder that contains one TFBS file per TF in bed or bed.gz format must be given (see details). The folder must also contain a so-called translation table, see the argument translationTable for details. We provide example files for selected supported genome assemblies (hg19, hg38 and mm10) that are fully compatible with GRaNIE as separate downloads. For more information, check https://difftf.readthedocs.io/en/latest/chapter2.html#dir-tfbs.

Usage

addTFBS(
    GRN, 
    source = "custom", 
    motifFolder = NULL, 
    TFs = "all", 
    translationTable = "translationTable.csv", 
    translationTable_sep = " ", 
    filesTFBSPattern = ".TFBS", 
    fileEnding = ".bed", 
    nTFMax = NULL, 
    EnsemblVersion = NULL, 
    JASPAR_useSpecificTaxGroup = NULL, 
    JASPAR_removeAmbiguousTFs = TRUE, 
    forceRerun = FALSE, 
    ... 
)
Arguments

- **GRN**: Object of class `GRN`.
- **source**: Character. One of `custom`, `JASPAR`. Default `custom`. If a custom source is being used, further details about the motif folder and files will be provided (see the other function arguments). If set to `JASPAR`, the `JASPAR2022` database is used.
- **motifFolder**: Character. No default. Only relevant if `source = "custom"`. Path to the folder that contains the TFBS predictions. The files must be in BED format, 6 columns, one file per TF. See the other parameters for more details. The folder must also contain a so-called translation table, see the argument `translationTable` for details.
- **TFs**: Character vector. Default all. Only relevant if `source = "custom"`. Vector of TF names to include. The special keyword all can be used to include all TF found in the folder as specified by `motifFolder`. If all is specified anywhere, all TFs will be included. TF names must otherwise match the file names that are found in the folder, without the file suffix.
- **translationTable**: Character. Default `translationTable.csv`. Only relevant if `source = "custom"`. Name of the translation table file that is also located in the folder along with the TFBS files. This file must have the following structure: at least 2 columns, called `ENSEMBL` and `ID`. ID denotes the ID for the TF that is used throughout the pipeline (e.g., AHR) and the prefix of how the corresponding file is called (e.g., AHR.0.B if the file for AHR is called AHR.0.B_TFBS.bed.gz), while `ENSEMBL` denotes the `ENSEMBL` ID (dot suffix; e.g., ENSG00000106546, are removed automatically if present).
- **translationTable_sep**: Character. Default `" "` (white space character). Only relevant if `source = "custom"`. The column separator for the `translationTable` file.
- **filesTFBSPattern**: Character. Default `_TFBS`. Only relevant if `source = "custom"`. Suffix for the file names in the TFBS folder that is not part of the TF name. Can be empty. For example, for the TF CTCF, if the file is called `CTCF.all.TFBS.bed`, set this parameter to `".all.TFBS"`.
- **fileEnding**: Character. Default `".bed"`. Only relevant if `source = "custom"`. File ending for the files from the motif folder.
- **nTFMax**: NULL or Integer[1,]. Default NULL. Maximal number of TFs to import. Can be used for testing purposes, e.g., setting to 5 only imports 5 TFs even though the whole `motifFolder` has many more TFs defined.
- **EnsemblVersion**: NULL or Character(). Default NULL. Only relevant if `source` is not set to `custom`, ignored otherwise. The Ensembl version to use for the retrieval of gene IDs from their provided database names (e.g., `JASPAR`) via `biomaRt`. By default (NULL), the newest version is selected for the most recent genome assembly versions is used (see `biomaRt::listEnsemblArchives()` for supported versions). This parameter can override this to use a custom (older) version instead.
- **JASPAR_useSpecificTaxGroup**: NULL or Character(). Default NULL. Should a tax group instead of the specific genome assembly be used for retrieving the TF list? This is useful for genomes
that are not human or mouse for which JASPAR otherwise returns too few TFs otherwise. If set to NULL, the specific genome version as provided in the object is used within TFBSTools::getMatrixSet in the opts list for species, while tax_group will be used instead if this argument is not set to NULL. For example, it can be set to vertebrates to use the vertebrates TF collection. For more details, see ?TFBSTools::getMatrixSet.

**JASPAR_removeAmbiguousTFs**

TRUE or FALSE. Default TRUE. Remove TFs for which the name as provided by JASPAR cannot be mapped uniquely to one and only Ensembl ID? By default (NULL), the newest version is selected (see biomaRt::listEnsemblArchives() for supported versions). This parameter can override this to use a custom (older) version instead.

**forceRerun**

TRUE or FALSE. Default FALSE. Force execution, even if the GRN object already contains the result. Overwrites the old results.

... Additional named elements for the opts function argument from ?TFBSTools::getMatrixSet that is used to query the JASPAR database.

**Value**

An updated GRN object, with additional information added from this function (GRN@annotation$TFs in particular)

**Examples**

# See the Workflow vignette on the GRaNIE website for examples

```r
add_featureVariation
# Quantify and interpret multiple sources of biological and technical variation for features (TFs, peaks, and genes) in a GRN object
```

**Description**

Runs the main function fitExtractVarPartModel of the package variancePartition: Fits a linear (mixed) model to estimate contribution of multiple sources of variation while simultaneously correcting for all other variables for the features in a GRN object (TFs, peaks, and genes) given particular metadata. The function reports the fraction of variance attributable to each metadata variable. **Note: The results are not added to GRN@connections$all.filtered, rerun the function getGRNConnections and set include_variancePartitionResults to TRUE to do so.** The results object is stored in GRN@stats$variancePartition and can be used for the various diagnostic and plotting functions from variancePartition.

**Usage**

```r
add_featureVariation(
  GRN,
  formula = "auto",
  metadata = c("all"),
```
features = "all_filtered",
nCores = 1,
forceRerun = FALSE,
...
)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GRN</td>
<td>Object of class GRN</td>
</tr>
<tr>
<td>formula</td>
<td>Character(1). Either auto or a manually defined formula to be used for the</td>
</tr>
<tr>
<td></td>
<td>model fitting. Default auto. Must include only terms that are part of the</td>
</tr>
<tr>
<td></td>
<td>metadata as specified with the metadata parameter. If set to auto, the</td>
</tr>
<tr>
<td></td>
<td>formula will be build automatically based on all metadata variables as</td>
</tr>
<tr>
<td></td>
<td>specified with the metadata parameter. By default, numerical variables will</td>
</tr>
<tr>
<td></td>
<td>be modeled as fixed effects, while variables that are defined as factors or</td>
</tr>
<tr>
<td></td>
<td>can be converted to factors (characters and logical variables) are modeled</td>
</tr>
<tr>
<td></td>
<td>as random effects as recommended by the variancePartition package.</td>
</tr>
<tr>
<td>metadata</td>
<td>Character vector. Default all. Vector of column names from the metadata</td>
</tr>
<tr>
<td></td>
<td>data frame that was provided when using the function addData. Must either</td>
</tr>
<tr>
<td></td>
<td>contain the special keyword all to denote that all (!) metadata columns from</td>
</tr>
<tr>
<td></td>
<td>GRN$data$metadata are taken or if not, a subset of the column names from</td>
</tr>
<tr>
<td></td>
<td>GRN$data$metadata to include in the model fitting for fitExtractVarPartModel.</td>
</tr>
<tr>
<td>features</td>
<td>Character(1). Either all_filtered or all. Default all_filtered. Should</td>
</tr>
<tr>
<td></td>
<td>variancePartition only be run for the features (TFs, peaks and genes) from</td>
</tr>
<tr>
<td></td>
<td>the filtered set of connections (the result of filterGRNAndConnectGenes) or</td>
</tr>
<tr>
<td></td>
<td>for all genes that are defined in the object? If set to all, the running</td>
</tr>
<tr>
<td></td>
<td>time is greatly increased.</td>
</tr>
<tr>
<td>nCores</td>
<td>Integer &gt;0. Default 1. Number of cores to use. A value &gt;1 requires the</td>
</tr>
<tr>
<td></td>
<td>BiocParallel package (as it is listed under Suggests, it may not be installed</td>
</tr>
<tr>
<td></td>
<td>yet).</td>
</tr>
<tr>
<td>forceRerun</td>
<td>TRUE or FALSE. Default FALSE. Force execution, even if the GRN object</td>
</tr>
<tr>
<td></td>
<td>already contains the result. Overwrites the old results.</td>
</tr>
<tr>
<td>...</td>
<td>Additional parameters passed on to variancePartition::fitExtractVarPartModel</td>
</tr>
<tr>
<td></td>
<td>beyond exprObj, formula and data. See the function help for more information</td>
</tr>
</tbody>
</table>

Details

The normalized count matrices are used as input for fitExtractVarPartModel.

Value

An updated GRN object, with additional information added from this function to GRN@stats$variancePartition as well as the elements genes, consensusPeaks and TFs within GRN@annotation. As noted above, the results are not added to GRN@connections$all.filtered; rerun the function getGRNConnections and set include_variancePartitionResults to TRUE to include the results in the eGRN output table.
add_TF_gene_correlation

See Also

addData
getGRNConnections

Examples

# See the Workflow vignette on the GRaNIE website for examples
# GRN = loadExampleObject()
# GRN = add_featureVariation(GRN, metadata = c("mt_frac"), forceRerun = TRUE)

add_TF_gene_correlation

Add TF-gene correlations to a GRN object.

Description

The information is currently stored in GRN@connections$TF_genes.filtered. Note that raw p-values are not adjusted.

Usage

add_TF_gene_correlation(
  GRN,
  corMethod = "pearson",
  nCores = 1,
  forceRerun = FALSE
)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GRN</td>
<td>Object of class GRN</td>
</tr>
<tr>
<td>corMethod</td>
<td>Character. One of pearson, spearman or bicor. Default pearson. Method for calculating the correlation coefficient. For pearson and spearman, see cor for details. bicor denotes the <em>biweight midcorrelation</em>, a correlation measure based on medians as calculated by WGCNA::bicorAndPvalue. Both spearman and bicor are considered more robust measures that are less prone to be affected by outliers.</td>
</tr>
<tr>
<td>nCores</td>
<td>Integer &gt;0. Default 1. Number of cores to use. A value &gt;1 requires the BiocParallel package (as it is listed under Suggests, it may not be installed yet).</td>
</tr>
<tr>
<td>forceRerun</td>
<td>TRUE or FALSE. Default FALSE. Force execution, even if the GRN object already contains the result. Overwrites the old results.</td>
</tr>
</tbody>
</table>

Value

An updated GRN object, with additional information added from this function.
Examples

# See the Workflow vignette on the GRaNIE website for examples
GRN = loadExampleObject()
GRN = add_TF_gene_correlation(GRN, forceRerun = FALSE)

AR_classification_wrapper

Run the activator-repressor classification for the TFs for a GRN object

Description

Run the activator-repressor classification for the TFs for a GRN object

Usage

AR_classification_wrapper(
  GRN,
  significanceThreshold_Wilcoxon = 0.05,
  plot_minNoTFBS_heatmap = 100,
  deleteIntermediateData = TRUE,
  plotDiagnosticPlots = TRUE,
  outputFolder = NULL,
  corMethod = "pearson",
  forceRerun = FALSE
)

Arguments

GRN Object of class GRN
significanceThreshold_Wilcoxon Numeric[0,1]. Default 0.05. Significance threshold for Wilcoxon test that is run in the end for the final classification. See the Vignette and "diffTF" paper for details.
plot_minNoTFBS_heatmap Integer[1,]. Default 100. Minimum number of TFBS for a TF to be included in the heatmap that is part of the output of this function.
deleteIntermediateData TRUE or FALSE. Default TRUE. Should intermediate data be deleted before returning the object after a successful run? Due to the size of the produced intermediate data, we recommend setting this to TRUE, but if memory or object size are not an issue, the information can also be kept.
plotDiagnosticPlots TRUE or FALSE. Default TRUE. Run and plot various diagnostic plots? If set to TRUE, PDF files will be produced and saved in the output directory (in a subfolder called plots).
**build_eGRN_graph**

Builds a graph out of a set of connections

### Description

This function requires a filtered set of connections in the GRN object as generated by `filterGRNAndConnectGenes`.

### Usage

```r
build_eGRN_graph(
  GRN,
  model_TF_gene_nodes_separately = FALSE,
  allowLoops = FALSE,
  removeMultiple = FALSE,
  directed = FALSE,
  forceRerun = FALSE
)
```

### Arguments

- **outputFolder**: Character or NULL. Default NULL. If set to NULL, the default output folder as specified when initiating the object in `initializeGRN` will be used. Otherwise, all output from this function will be put into the specified folder. If a folder is provided, while we recommend specifying an absolute path, a relative one also works.

- **corMethod**: Character. One of `pearson`, `spearman` or `bicor`. Default `pearson`. Method for calculating the correlation coefficient. For `pearson` and `spearman`, see `cor` for details. `bicor` denotes the *biweight midcorrelation*, a correlation measure based on medians as calculated by WGCNA::bicorAndPvalue. Both `spearman` and `bicor` are considered more robust measures that are less prone to be affected by outliers.

- **forceRerun**: TRUE or FALSE. Default FALSE. Force execution, even if the GRN object already contains the result. Overwrites the old results.

### Value

An updated GRN object, with additional information added from this function.

### Examples

```r
# See the Workflow vignette on the GRaNIE website for examples
# GRN = loadExampleObject()
# GRN = AR_classification_wrapper(GRN, outputFolder = ".", forceRerun = FALSE)
```
Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GRN</td>
<td>Object of class <code>GRN</code></td>
</tr>
<tr>
<td>model_TF_gene_nodes_separately</td>
<td>TRUE or FALSE. Should TF and gene nodes be modeled separately? If set to TRUE, this may lead to unwanted effects in case of TF-TF connections (i.e., a TF regulating another TF)</td>
</tr>
<tr>
<td>allowLoops</td>
<td>TRUE or FALSE. Default FALSE. Allow loops in the network (i.e., a TF that regulates itself)</td>
</tr>
<tr>
<td>removeMultiple</td>
<td>TRUE or FALSE. Default FALSE. Remove loops with the same start and end point? This can happen if multiple TF originate from the same gene, for example.</td>
</tr>
<tr>
<td>directed</td>
<td>TRUE or FALSE. Default FALSE. Should the network be directed?</td>
</tr>
<tr>
<td>forceRerun</td>
<td>TRUE or FALSE. Default FALSE. Force execution, even if the GRN object already contains the result. Overwrites the old results.</td>
</tr>
</tbody>
</table>

Value

An updated `GRN` object, with the graph(s) being stored in the slot 'graph' (i.e., 'GRN@graph' for both TF-gene and TF-peak-gene graphs)

See Also

`filterGRNAndConnectGenes`

Examples

```r
# See the Workflow vignette on the GRaNIE website for examples
GRN = loadExampleObject()
GRN = build_eGRN_graph(GRN, forceRerun = FALSE)
```

The enrichment analysis is based on the subset of the network connected to a particular community as identified by `calculateCommunitiesStats`, see `calculateTFEnrichment` and `calculateGeneralEnrichment` for TF-specific and general enrichment, respectively. This function requires the existence of the eGRN graph in the `GRN` object as produced by `build_eGRN_graph` as well as community information as calculated by `calculateCommunitiesStats`. Results can subsequently be visualized with the function `plotCommunitiesEnrichment`. 
calculateCommunitiesEnrichment

**Usage**

```r
calculateCommunitiesEnrichment(
  GRN,
  ontology = c("GO_BP", "GO_MF"),
  algorithm = "weight01",
  statistic = "fisher",
  background = "neighborhood",
  background_geneTypes = "all",
  selection = "byRank",
  communities = NULL,
  pAdjustMethod = "BH",
  forceRerun = FALSE)
```

**Arguments**

- **GRN**: Object of class `GRN`
- **ontology**: Character vector of ontologies. Default `c("GO_BP", "GO_MF")`. Valid values are "GO_BP", "GO_MF", "GO_CC", "KEGG", "DO", and "Reactome", referring to GO Biological Process, GO Molecular Function, GO Cellular Component, KEGG, Disease Ontology, and Reactome Pathways, respectively. GO ontologies require the topGO, "KEGG" the clusterProfiler, "DO" the DOSE, and "Reactome" the ReactomePA packages, respectively. As they are listed under Suggests, they may not yet be installed, and the function will throw an error if they are missing.
- **algorithm**: Character. Default "weight01". One of: "classic", "elim", "weight", "weight01", "lea", "parentchild". Only relevant if ontology is GO related (GO_BP, GO_MF, GO_CC), ignored otherwise. Name of the algorithm that handles the GO graph structures. Valid inputs are those supported by the topGO library. For general information about the algorithms, see [https://academic.oup.com/bioinformatics/article/22/13/1600/193669](https://academic.oup.com/bioinformatics/article/22/13/1600/193669). weight01 is a mixture between the elim and the weight algorithms.
- **statistic**: Character. Default "fisher". One of: "fisher", "ks", "t". Statistical test to be used. Only relevant if ontology is GO related (GO_BP, GO_MF, GO_CC), and valid inputs are a subset of those supported by the topGO library (we had to remove some as they do not seem to work properly in topGO either), ignored otherwise. For the other ontologies the test statistic is always Fisher.
- **background**: Character. Default "neighborhood". One of: "all_annotated", "all_RNA", "all_RNA_filtered", "neighborhood". Set of genes to be used to construct the background for the enrichment analysis. This can either be all annotated genes in the reference genome (all_annotated), all genes from the provided RNA data (all_RNA), all genes from the provided RNA data excluding those marked as filtered after executing filterData (all_RNA_filtered), or all the genes that are within the neighborhood of any peak (before applying any filters except for the user-defined promoterRange value in addConnections_peak_gene) (neighborhood).
- **background_geneTypes**: Character vector of gene types that should be considered for the background.
calculateCommunitiesEnrichment

Default "all". Only gene types as defined in the GRN object, slot GRN@annotation$genes$gene.type are allowed. The special keyword "all" means no filter on gene type.

**selection**  
Character. Default "byRank". One of: "byRank", "byLabel". Specify whether the communities enrichment will be calculated based on their rank, where the largest community (with most vertices) would have a rank of 1, or by their label. Note that the label is independent of the rank.

**communities**  
NULL or numeric vector or character vector. Default NULL. If set to NULL, all community enrichments that have been calculated before are plotted. If a numeric vector is specified (when selection = "byRank"), the rank of the communities is specified. For example, communities = c(1,4) then denotes the first and fourth largest community. If a character vector is specified (when selection = "byLabel"), the name of the communities is specified instead. For example, communities = c("1","4") then denotes the communities with the names "1" and "4", which may or may not be the largest and fourth largest communities among all.

**pAdjustMethod**  
Character. Default "BH". One of: "holm", "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr". This parameter is only relevant for the following ontologies: KEGG, DO, Reactome. For the other ontologies, the algorithm serves as an adjustment.

**forceRerun**  
TRUE or FALSE. Default FALSE. Force execution, even if the GRN object already contains the result. Overwrites the old results.

**Details**

All enrichment functions use the TF-gene graph as defined in the ‘GRN’ object. See the ‘ontology’ argument for currently supported ontologies. Also note that some parameter combinations for ‘algorithm’ and ‘statistic’ are incompatible, an error message will be thrown in such a case.

**Value**

An updated GRN object, with the enrichment results stored in the stats$Enrichment$byCommunity slot.

**See Also**

plotCommunitiesEnrichment  
plotGeneralEnrichment  
calculateGeneralEnrichment  
calculateCommunitiesStats

**Examples**

# See the Workflow vignette on the GRaNIE website for examples  
GRN = loadExampleObject()  
GRN = calculateCommunitiesEnrichment(GRN, ontology = c("GO_BP"), forceRerun = FALSE)
calculateCommunitiesStats

Generate graph communities and their summarizing statistics

Description

The results can subsequently be visualized with the function `plotCommunitiesStats`. This function requires a filtered set of connections in the `GRN` object as generated by `filterGRNAndConnectGenes`. It then generates the TF-gene graph from the filtered connections, and clusters its vertices into communities using established community detection algorithms.

Usage

`calculateCommunitiesStats(GRN, clustering = "louvain", forceRerun = FALSE, ...)`

Arguments

- `GRN` Object of class `GRN`
- `clustering` Character. Default `louvain`. One of: `louvain`, `leiden`, `leading_eigen`, `fast_greedy`, `optimal`, `walktrap`. The community detection algorithm to be used. Please bear in mind the robustness and time consumption of the algorithms when opting for an alternative to the default.
- `forceRerun` `TRUE` or `FALSE`. Default `FALSE`. Force execution, even if the `GRN` object already contains the result. Overwrites the old results.
- `...` Additional parameters for the used clustering method, see the `igraph::cluster_` methods for details on the specific parameters and what they do. For `leiden` clustering, for example, you may add a `resolution_parameter` to control the granularity of the community detection or `n_iterations` to modify the number of iterations.

Value

An updated `GRN` object, with a table that consists of the connections clustered into communities stored in the `GRN@graph$TF_gene$clusterGraph` slot as well as within the `igraph` object in `GRN@graph$TF_gene$graph` (retrievable via `igraph::vertex.attributes(GRN@graph$TF_gene$graph)$community` for example.)

See Also

- `plotCommunitiesStats`
- `calculateCommunitiesEnrichment`

Examples

```r
# See the Workflow vignette on the GRaNIE website for examples
GRN = loadExampleObject()
GRN = calculateCommunitiesStats(GRN, forceRerun = FALSE)
```
calculateGeneralEnrichment

*Run an enrichment analysis for the genes in the whole network in the filtered GRN object*

**Description**

The enrichment analysis is based on the whole network, see `calculateCommunitiesEnrichment` and `calculateTFEnrichment` for community- and TF-specific enrichment, respectively. This function requires the existence of the eGRN graph in the GRN object as produced by `build_eGRN_graph`. Results can subsequently be visualized with the function `plotGeneralEnrichment`.

**Usage**

```r
calculateGeneralEnrichment(
  GRN, 
  ontology = c("GO_BP", "GO_MF"), 
  algorithm = "weight01", 
  statistic = "fisher", 
  background = "neighborhood", 
  background_geneTypes = "all", 
  pAdjustMethod = "BH", 
  forceRerun = FALSE 
)
```

**Arguments**

- **GRN** Object of class GRN
- **ontology** Character vector of ontologies. Default c("GO_BP", "GO_MF"). Valid values are "GO_BP", "GO_MF", "GO_CC", "KEGG", "DO", and "Reactome", referring to GO Biological Process, GO Molecular Function, GO Cellular Component, KEGG, Disease Ontology, and Reactome Pathways, respectively. GO ontologies require the topGO, "KEGG" the clusterProfiler, "DO" the DOSE, and "Reactome" the ReactomePA packages, respectively. As they are listed under Suggests, they may not yet be installed, and the function will throw an error if they are missing.
- **algorithm** Character. Default "weight01". One of: "classic", "elim", "weight", "weight01", "lea", "parentchild". Only relevant if ontology is GO related (GO_BP, GO_MF, GO_CC), ignored otherwise. Name of the algorithm that handles the GO graph structures. Valid inputs are those supported by the topGO library. For general information about the algorithms, see https://academic.oup.com/bioinformatics/article/22/13/1600/193669. weight01 is a mixture between the elim and the weight algorithms.
- **statistic** Character. Default "fisher". One of: "fisher", "ks", "t". Statistical test to be used. Only relevant if ontology is GO related (GO_BP, GO_MF, GO_CC), and valid inputs are a subset of those supported by the topGO library (we had to remove some as they do not seem to work properly in topGO either), ignored otherwise. For the other ontologies the test statistic is always Fisher.
calculateGeneralEnrichment

background Character. Default "neighborhood". One of: "all_annotated", "all_RNA", "all_RNA_filtered", "neighborhood". Set of genes to be used to construct the background for the enrichment analysis. This can either be all annotated genes in the reference genome (all_annotated), all genes from the provided RNA data (all_RNA), all genes from the provided RNA data excluding those marked as filtered after executing filterData (all_RNA_filtered), or all the genes that are within the neighborhood of any peak (before applying any filters except for the user-defined promoterRange value in addConnections_peak_gene) (neighborhood).

background_geneTypes Character vector of gene types that should be considered for the background. Default "all". Only gene types as defined in the GRN object, slot GRN@annotation$genes$gene.type are allowed. The special keyword "all" means no filter on gene type.

pAdjustMethod Character. Default "BH". One of: "holm", "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr". This parameter is only relevant for the following ontologies: KEGG, DO, Reactome. For the other ontologies, the algorithm serves as an adjustment.

forceRerun TRUE or FALSE. Default FALSE. Force execution, even if the GRN object already contains the result. Overwrites the old results.

Details

All enrichment functions use the TF-gene graph as defined in the ‘GRN’ object. See the ‘ontology’ argument for currently supported ontologies. Also note that some parameter combinations for ‘algorithm’ and ‘statistic’ are incompatible, an error message will be thrown in such a case.

Value

An updated GRN object, with the enrichment results stored in the stats$Enrichment$general slot.

See Also

plotGeneralEnrichment
calculateCommunitiesEnrichment
calculateTFEnrichment
plotCommunitiesEnrichment

Examples

# See the Workflow vignette on the GRaNIE website for examples
GRN = loadExampleObject()
GRN = calculateGeneralEnrichment(GRN, ontology = "GO_BP", forceRerun = FALSE)
calculateTFEnrichment  *Run an enrichment analysis for the set of genes connected to a particular TF or sets of TFs in the filtered GRN object*

**Description**

The enrichment analysis is based on the subset of the network connected to particular TFs (TF regulators), see `calculateCommunitiesEnrichment` and `calculateGeneralEnrichment` for community- and general enrichment, respectively. This function requires the existence of the eGRN graph in the GRN object as produced by `build_eGRN_graph`. Results can subsequently be visualized with the function `plotTFEnrichment`.

**Usage**

```r
calculateTFEnrichment(
  GRN,
  rankType = "degree",
  n = 3,
  TF.IDs = NULL,
  ontology = c("GO_BP", "GO_MF"),
  algorithm = "weight01",
  statistic = "fisher",
  background = "neighborhood",
  background_geneTypes = "all",
  pAdjustMethod = "BH",
  forceRerun = FALSE
)
```

**Arguments**

- **GRN**: Object of class `GRN`
- **rankType**: Character. Default "degree". One of: "degree", "EV", "custom". This parameter will determine the criterion to be used to identify the "top" TFs. If set to "degree", the function will select top TFs based on the number of connections to genes they have, i.e. based on their degree-centrality. If set to "EV" it will select the top TFs based on their eigenvector-centrality score in the network. If set to custom, a set of TF IDs will have to be passed to the "TF.IDs" parameter.
- **n**: Numeric. Default 3. If this parameter is passed as a value between 0 and 1, it is treated as a percentage of top nodes. If the value is passed as an integer it will be treated as the number of top nodes. This parameter is not relevant if `rankType` = "custom".
- **TF.IDs**: Character vector. Default NULL. If the rank type is set to "custom", a vector of TF IDs for which the GO enrichment should be calculated should be passed to this parameter.
ontology Character vector of ontologies. Default c("GO_BP", "GO_MF"). Valid values are "GO_BP", "GO_MF", "GO_CC", "KEGG", "DO", and "Reactome", referring to GO Biological Process, GO Molecular Function, GO Cellular Component, KEGG, Disease Ontology, and Reactome Pathways, respectively. GO ontologies require the topGO, "KEGG" the clusterProfiler, "DO" the DOSE, and "Reactome" the ReactomePA packages, respectively. As they are listed under Suggests, they may not yet be installed, and the function will throw an error if they are missing.

algorithm Character. Default "weight01". One of: "classic", "elim", "weight", "weight01", "lea", "parentchild". Only relevant if ontology is GO related (GO_BP, GO_MF, GO_CC), ignored otherwise. Name of the algorithm that handles the GO graph structures. Valid inputs are those supported by the topGO library. For general information about the algorithms, see https://academic.oup.com/bioinformatics/article/22/13/1600/193669. weight01 is a mixture between the elim and the weight algorithms.

statistic Character. Default "fisher". One of: "fisher", "ks", "t". Statistical test to be used. Only relevant if ontology is GO related (GO_BP, GO_MF, GO_CC), and valid inputs are a subset of those supported by the topGO library (we had to remove some as they do not seem to work properly in topGO either), ignored otherwise. For the other ontologies the test statistic is always Fisher.

background Character. Default "neighborhood". One of: "all_annotated", "all_RNA", "all_RNA_filtered", "neighborhood". Set of genes to be used to construct the background for the enrichment analysis. This can either be all annotated genes in the reference genome (all_annotated), all genes from the provided RNA data (all_RNA), all genes from the provided RNA data excluding those marked as filtered after executing filterData (all_RNA_filtered), or all the genes that are within the neighborhood of any peak (before applying any filters except for the user-defined promoterRange value in addConnections_peak_gene) (neighborhood).

background_geneTypes Character vector of gene types that should be considered for the background. Default "all". Only gene types as defined in the GRN object, slot GRN@annotation$genes$gene.type are allowed. The special keyword "all" means no filter on gene type.

pAdjustMethod Character. Default "BH". One of: "holm", "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr". This parameter is only relevant for the following ontologies: KEGG, DO, Reactome. For the other ontologies, the algorithm serves as an adjustment.

forceRerun TRUE or FALSE. Default FALSE. Force execution, even if the GRN object already contains the result. Overwrites the old results.

Details

All enrichment functions use the TF-gene graph as defined in the ‘GRN’ object. See the ‘ontology’ argument for currently supported ontologies. Also note that some parameter combinations for ‘algorithm’ and ‘statistic’ are incompatible, an error message will be thrown in such a case.

Value

An updated GRN object, with the enrichment results stored in the stats$Enrichment$byTF slot.
changeOutputDirectory

Description

Change the output directory of a GRN object

Usage

changeOutputDirectory(GRN, outputDirectory = ".")

Arguments

GRN Object of class GRN
outputDirectory Character. Default ".". New output directory for all output files unless overwritten via the parameter outputFolder.

Value

An updated GRN object, with the output directory being adjusted accordingly

Examples

GRN = loadExampleObject()
GRN = changeOutputDirectory(GRN, outputDirectory = ".")

See Also

plotTFEnrichment

Examples

# See the Workflow vignette on the GRaNIE website for examples
GRN = loadExampleObject()
GRN = calculateTFEnrichment(GRN, n = 5, ontology = "GO_BP", forceRerun = FALSE)
deleteIntermediateData

Optional convenience function to delete intermediate data from the function AR_classification_wrapper and summary statistics that may occupy a lot of space

Description

Optional convenience function to delete intermediate data from the function AR_classification_wrapper and summary statistics that may occupy a lot of space

Usage

deleteIntermediateData(GRN)

Arguments

GRN Object of class GRN

Value

An updated GRN object, with some slots being deleted (GRN@data$TFs$classification as well as GRN@stats$connectionDetails.l)

Examples

# See the Workflow vignette on the GRaNIE website for examples
GRN = loadExampleObject()
GRN = deleteIntermediateData(GRN)

filterConnectionsForPlotting

Filter connections for subsequent visualization with `visualizeGRN()` from the filtered eGRN

Description

This helper function provides an easy and flexible way to retain particular connections for plotting and discard all others. Note that this filtering is only relevant and applicable for the function `visualizeGRN()` and ignored anywhere else. This makes it possible to visualize only specific TF regulons or to plot only connections that fulfill particular filter criteria. Due to the flexibility of the implementation by allowing arbitrary filters that are passed directly to dplyr::filter, users can visually investigate the eGRN, which is particularly useful when the eGRNs is large and has many connections.
Usage

filterConnectionsForPlotting(GRN, plotAll = TRUE, ..., forceRerun = FALSE)

Arguments

GRN Object of class GRN
plotAll TRUE or FALSE. Default TRUE. Should all connections be included for plotting? If set to TRUE, all connections are marked for plotting and everything else is ignored. This resets any previous setting. If set FALSE, the filter expressions (if any) are used to determine which connection to plot

... An arbitrary set of arguments that is used directly, without modification, as input for dplyr::filter and therefore has to be valid expression that dplyr::filter understands. The filtering is based on the all.filtered table as stored in GRN@connections$all.filtered$`0`. Thus, the specific filters can be completely arbitrary for ultimate flexibility and must only adhere to the column names and types as defined in GRN@connections$all.filtered$`0`. See the examples also for what you can do.

forceRerun TRUE or FALSE. Default FALSE. Force execution, even if the GRN object already contains the result. Overwrites the old results.

Value

An updated GRN object, with added data from this function.

Examples

# See the Workflow vignette on the GRaNIE website for examples
GRN = loadExampleObject()
GRN = filterConnectionsForPlotting (GRN, plotAll = FALSE, TF.ID == "E2F6.0.A")
GRN = filterConnectionsForPlotting (GRN, plotAll = FALSE, TF_peak.r > 0.7 | TF_peak.fdr < 0.2)
GRN = filterConnectionsForPlotting (GRN, plotAll = FALSE, TF_peak.r > 0.7, TF_peak.fdr < 0.2)

filterData Filter RNA-seq and/or peak data from a GRN object

Description

This function marks genes and/or peaks as filtered depending on the chosen filtering criteria and is based on the count data AFTER potential normalization as chosen when using the addData function. Most of the filters may not be meaningful and useful anymore to apply after using particular normalization schemes that can give rise to, for example, negative values such as cyclic loess normalization. If normalized counts do not represent counts anymore but rather a deviation from a mean or something a like, the filtering criteria usually do not make sense anymore. Filtered genes / peaks will then be disregarded when adding connections in subsequent steps via addConnections_TF_peak and addConnections_peak_gene. This function does NOT (re)filter existing connections when the GRN object already contains connections. Thus, upon re-execution of this function with different filtering criteria, all downstream steps have to be re-run.
Usage

filterData(
  GRN,
  minNormalizedMean_peaks = NULL,
  maxNormalizedMean_peaks = NULL,
  minNormalizedMeanRNA = NULL,
  maxNormalizedMeanRNA = NULL,
  chrToKeep_peaks = NULL,
  minSize_peaks = 20,
  maxSize_peaks = 10000,
  minCV_peaks = NULL,
  maxCV_peaks = NULL,
  minCV_genes = NULL,
  maxCV_genes = NULL,
  forceRerun = FALSE
)

Arguments

GRN Object of class GRN

minNormalizedMean_peaks Numeric[0,] or NULL. Default 5. Minimum mean across all samples for a peak to be retained for the normalized counts table. Set to NULL for not applying the filter. Be aware that depending on the chosen normalization, this filter may not make sense and should NOT be applied. See the notes for this function.

maxNormalizedMean_peaks Numeric[0,] or NULL. Default NULL. Maximum mean across all samples for a peak to be retained for the normalized counts table. Set to NULL for not applying the filter. Be aware that depending on the chosen normalization, this filter may not make sense and should NOT be applied. See the notes for this function.

minNormalizedMeanRNA Numeric[0,] or NULL. Default 5. Minimum mean across all samples for a gene to be retained for the normalized counts table. Set to NULL for not applying the filter. Be aware that depending on the chosen normalization, this filter may not make sense and should NOT be applied. See the notes for this function.

maxNormalizedMeanRNA Numeric[0,] or NULL. Default NULL. Maximum mean across all samples for a gene to be retained for the normalized counts table. Set to NULL for not applying the filter. Be aware that depending on the chosen normalization, this filter may not make sense and should NOT be applied. See the notes for this function.

chrToKeep_peaks Character vector or NULL. Default NULL. Vector of chromosomes that peaks are allowed to come from. This filter can be used to filter sex chromosomes from the peaks, for example (e.g, c(paste0("chr", 1:22), "chrX", "chrY"))

minSize_peaks Integer[1,] or NULL. Default 20. Minimum peak size (width, end - start) for a peak to be retained. Set to NULL for not applying the filter.
filterGRNAAndConnectGenes

**maxSize_peaks**  
Integer[1,] or NULL. Default 10000. Maximum peak size (width, end - start) for a peak to be retained. Set to NULL for not applying the filter.

**minCV_peaks**  
Numeric[0,] or NULL. Default NULL. Minimum CV (coefficient of variation, a unitless measure of variation) for a peak to be retained. Set to NULL for not applying the filter. Be aware that depending on the chosen normalization, this filter may not make sense and should NOT be applied. See the notes for this function.

**maxCV_peaks**  
Numeric[0,] or NULL. Default NULL. Maximum CV (coefficient of variation, a unitless measure of variation) for a peak to be retained. Set to NULL for not applying the filter. Be aware that depending on the chosen normalization, this filter may not make sense and should NOT be applied. See the notes for this function.

**minCV_genes**  
Numeric[0,] or NULL. Default NULL. Minimum CV (coefficient of variation, a unitless measure of variation) for a gene to be retained. Set to NULL for not applying the filter. Be aware that depending on the chosen normalization, this filter may not make sense and should NOT be applied. See the notes for this function.

**maxCV_genes**  
Numeric[0,] or NULL. Default NULL. Maximum CV (coefficient of variation, a unitless measure of variation) for a gene to be retained. Set to NULL for not applying the filter. Be aware that depending on the chosen normalization, this filter may not make sense and should NOT be applied. See the notes for this function.

**forceRerun**  
TRUE or FALSE. Default FALSE. Force execution, even if the GRN object already contains the result. Overwrites the old results.

**Details**

All this function does is setting (or modifying) the filtering flag in `GRN@data$peaks$counts_metadata` and `GRN@data$RNA$counts_metadata`, respectively.

**Value**

An updated `GRN` object, with added data from this function.

**Examples**

```r
# See the Workflow vignette on the GRaNIE website for examples
GRN = loadExampleObject()
GRN = filterData(GRN, forceRerun = FALSE)
```

---

**filterGRNAAndConnectGenes**

Filter TF-peaks and peak-gene connections and combine them to TF-peak-gene connections to construct an eGRN.
Description

This is one of the main integrative functions of the GRaNIE package. It has two main functions: First, filtering both TF-peak and peak-gene connections according to different criteria such as FDR and other properties. Second, joining the three major elements that an eGRN consist of (TFs, peaks, genes) into one data frame, with one row per unique TF-peak-gene connection. After successful execution, the connections (along with additional feature metadata) can be retrieved with the function `getGRNConnections`. Note that a previously stored eGRN graph is reset upon successful execution of this function along with printing a descriptive warning, and re-running the function `build_eGRN_graph` is necessary when any of the network functions of the package shall be executed. If the filtered connections changed, all network related enrichment functions also have to be rerun. Internally, before joining them, both TF-peak links and peak-gene connections are filtered separately for reasons of memory and computational efficacy: First filtering out unwanted links dramatically reduces the memory needed for the full eGRN. Peak-gene p-value adjustment is only done after all filtering steps on the remaining set of connections to lower the statistical burden of multiple-testing adjustment; therefore, this may lead to initially counter-intuitive effects such as a particular connections not being included anymore as compared to a filtering based on different thresholds, or the FDR being different for the same reason.

Usage

```r
filterGRNAndConnectGenes(
  GRN,
  TF_peak.fdr.threshold = 0.2,
  TF_peak.connectionTypes = "all",
  peak.SNP_filter = list(min_nSNPs = 0, filterType = "orthogonal"),
  peak_gene.p_raw.threshold = NULL,
  peak_gene.fdr.threshold = 0.2,
  peak_gene.fdr.method = "BH",
  peak_gene.IHW.covariate = NULL,
  peak_gene.IHW.nbins = "auto",
  outputFolder = NULL,
  gene.types = c("protein_coding"),
  allowMissingTFs = FALSE,
  allowMissingGenes = TRUE,
  peak_gene.r_range = c(0, 1),
  peak_gene.selection = "all",
  peak_gene.maxDistance = NULL,
  filterTFs = NULL,
  filterGenes = NULL,
  filterPeaks = NULL,
  TF_peak_FDR_selectViaCorBins = FALSE,
  filterLoops = TRUE,
  resetGraphAndStoreInternally = TRUE,
  silent = FALSE,
  forceRerun = FALSE
)
```
filterGRNAndConnectGenes

Arguments

GRN Object of class GRN

TF_peak.fdr.threshold Numeric[0,1]. Default 0.2. Maximum FDR for the TF-peak links. Set to 1 or NULL to disable this filter.

TF_peak.connectionTypes Character vector. Default all. TF-peak connection types to consider. The special keyword all denotes all connection types (e.g., expression and TFActivity) that are found in the GRN object. By default, only expression is present in the object, so all and expression are usually equivalent unless calculation of TF-peak links based on TF activity has also been enabled.

peak.SNP_filter Named list. Default list(min_nSNPs = 0, filterType = "orthogonal"). Filters related to SNP data if they have been added with the function addSNPData, ignored otherwise. The named list must contain at least two elements: First, min_nSNPs, an integer >= 0 that denotes how many SNPs a peak has to overlap with at least to pass the filter or be considered for inclusion. Second, filterType, a character that must either be orthogonal or extra and denotes whether the SNP filter is orthogonal to the other filters (i.e., an alternative way of when a peak is considered for being kept) or whether the SNP filter is in addition to all other filters. For more help, see the Vignettes.

peak_gene.p.raw.threshold Numeric[0,1]. Default NULL. Threshold for the peak-gene connections, based on the raw p-value. All peak-gene connections with a larger raw p-value will be filtered out.

peak_gene.fdr.threshold Numeric[0,1]. Default 0.2. Threshold for the peak-gene connections, based on the FDR. All peak-gene connections with a larger FDR will be filtered out.

peak_gene.fdr.method Character. Default "BH". One of: "holm", "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr", "none", "IHW". Method for adjusting p-values for multiple testing. If set to "IHW", the package IHW is required (as it is listed under Suggests, it may not be installed), and independent hypothesis weighting will be performed, and a suitable covariate has to be specified for the parameter peak_gene.IHW.covariate.

peak_gene.IHW.covariate Character. Default NULL. Name of the covariate to use for IHW (column name from the table that is returned with the function getGRNConnections. Only relevant if peak_gene.fdr.method is set to "IHW". You have to make sure the specified covariate is suitable for IHW, see the diagnostic plots that are generated in this function for this. For many datasets, the peak-gene distance (called peak_gene.distance in the object) seems suitable.

peak_gene.IHW.nbins Integer or "auto". Default "auto". Number of bins for IHW. Only relevant if peak_gene.fdr.method is set to "IHW".

outputFolder Character or NULL. Default NULL. If set to NULL, the default output folder as specified when initiating the object in initializeGRN will be used. Otherwise,
all output from this function will be put into the specified folder. If a folder is provided, while we recommend specifying an absolute path, a relative one also works.

gene.types Character vector of supported gene types. Default c("protein_coding", "lincRNA"). Filter for gene types to retain, genes with gene types not listed here are filtered. The special keyword "all" indicates no filter and retains all gene types. The specified names must match the names as stored in the GRN object (see GRN@annotation$genes$gene.type) and correspond 1:1 to the gene type names as provided by biomaRt, with the exception of lincRNAs, which is internally renamed to lincRNAs when first fetching all gene types. This is done due to a recent change in biomaRt and aims at keeping backwards compatibility with GRN objects.

allowMissingTFs TRUE or FALSE. Default FALSE. Should connections be returned for which the TF is NA (i.e., connections consisting only of peak-gene links?). If set to TRUE, this generally greatly increases the number of connections but it may not be what you aim for.

allowMissingGenes TRUE or FALSE. Default TRUE. Should connections be returned for which the gene is NA (i.e., connections consisting only of TF-peak links?). If set to TRUE, this generally increases the number of connections.

peak_gene.r_range Numeric(2). Default c(0,1). Filter for lower and upper limit for the peak-gene links. Only links will be retained if the correlation coefficient is within the specified interval. This filter is usually used to filter out negatively correlated peak-gene links.

peak_gene.selection "all" or "closest". Default "all". Filter for the selection of genes for each peak. If set to "all", all previously identified peak-gene are used, while "closest" only retains the closest gene for each peak that is retained until the point the filter is applied.

peak_gene.maxDistance Integer >0. Default NULL. Maximum peak-gene distance to retain a peak-gene connection.

filterTFs Character vector. Default NULL. Vector of TFs (as named in the GRN object) to retain. All TFs not listed will be filtered out.

filterGenes Character vector. Default NULL. Vector of gene IDs (as named in the GRN object) to retain. All genes not listed will be filtered out.

filterPeaks Character vector. Default NULL. Vector of peak IDs (as named in the GRN object) to retain. All peaks not listed will be filtered out.

TF_peak_FDR_selectViaCorBins TRUE or FALSE. Default FALSE. Use a modified procedure for selecting TF-peak links. Instead of selecting solely based on the user-specified FDR, this procedure first identifies the correlation bin closest to 0 that contains at least one significant TF-peak link according to the chosen TF_peak.fdr.threshold. This is done separately for both FDR directions. It then retains all TF-peak links that have
a correlation bin at least as extreme as the identified pair. For example, if the
correlation bin [0.35,0.40] contains a significant TF-peak link while [0,0.05],
[0.05,0.10], ..., [0.30,0.35] do not, all TF-peak links with a correlation of at least
0.35 or above are selected (i.e., bins [0.35,0.40], [0.40,0.45], ..., [0.95,1.00]).
Thus, for the final selection, also links with a higher FDR but a more extreme
correlation may be selected.

filterLoops TRUE or FALSE. Default TRUE. If a TF regulates itself (i.e., the TF and the gene
are the same entity), should such loops be filtered from the GRN?

resetGraphAndStoreInternally
TRUE or FALSE. Default TRUE. If set to TRUE, the stored eGRN graph (slot graph)
is reset due to the potentially changed connections that would otherwise cause
conflicts in the information stored in the object. Also, a GRN object is returned.
If set to FALSE, only the new filtered connections are returned and the object is
not altered.

silent TRUE or FALSE. Default FALSE. Print progress messages and filter statistics.

forceRerun TRUE or FALSE. Default FALSE. Force execution, even if the GRN object already
contains the result. Overwrites the old results.

Value
An updated GRN object, with additional information added from this function. The filtered and
merged TF-peak and peak-gene connections in the slot GRN@connections$all.filtered and can
be retrieved (along with other feature metadata) using the function getGRNConnections.

See Also
visualizeGRN
addConnections_TF_peak
addConnections_peak_gene
build_eGRN_graph
getGRNConnections

Examples
# See the Workflow vignette on the GRaNIE website for examples
GRN = loadExampleObject()
GRN = filterGRNAndConnectGenes(GRN)
generateStatsSummary

Description

This function calls `filterGRNAndConnectGenes` repeatedly and stores the total number of connections and other statistics each time to summarize them afterwards. All arguments are identical to the ones in `filterGRNAndConnectGenes`, see the help for this function for details. The function `plot_stats_connectionSummary` can be used afterwards for plotting.

Usage

```r
generateStatsSummary(
  GRN,
  TF_peak.fdr = c(0.001, 0.01, 0.05, 0.1, 0.2),
  TF_peak.connectionTypes = "all",
  peak_gene.fdr = c(0.001, 0.01, 0.05, 0.1, 0.2),
  peak_gene.r_range = c(0, 1),
  gene.types = c("protein_coding"),
  allowMissingGenes = c(FALSE, TRUE),
  allowMissingTFs = c(FALSE),
  forceRerun = FALSE
)
```

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GRN</td>
<td>Object of class <code>GRN</code></td>
</tr>
<tr>
<td>TF_peak.fdr</td>
<td>Numeric vector [0,1]. Default c(0.001, 0.01, 0.05, 0.1, 0.2). TF-peak FDR values to iterate over.</td>
</tr>
<tr>
<td>TF_peak.connectionTypes</td>
<td>Character vector. Default all. TF-peak connection types to consider. The special keyword &quot;all&quot; denotes all connection types (e.g., expression and TFActivity) that are found in the <code>GRN</code> object. By default, only expression is present in the object, so all and expression are usually equivalent unless calculation of TF-peak links based on TF activity has also been enabled.</td>
</tr>
<tr>
<td>peak_gene.fdr</td>
<td>Numeric vector [0,1]. Default c(0.001, 0.01, 0.05, 0.1, 0.2). Peak-gene FDR values to iterate over.</td>
</tr>
<tr>
<td>peak_gene.r_range</td>
<td>Numeric vector of length 2[-1,1]. Default c(0, 1). The correlation range of peak-gene connections to keep.</td>
</tr>
<tr>
<td>gene.types</td>
<td>Character vector of supported gene types. Default c(&quot;protein_coding&quot;, &quot;lincRNA&quot;). Filter for gene types to retain, genes with gene types not listed here are filtered. The special keyword &quot;all&quot; indicates no filter and retains all gene types. The specified names must match the names as stored in the <code>GRN</code> object (see <code>GRN@annotation$genes$gene.type</code>) and correspond 1:1 to the gene type names as provided by <code>biomaRt</code>, with the exception of lncRNAs, which is internally renamed to lincRNAs when first fetching all gene types. This is done due to a recent change in <code>biomaRt</code> and aims at keeping backwards compatibility with <code>GRN</code> objects.</td>
</tr>
</tbody>
</table>
getCounts

allowMissingGenes Logical vector of length 1 or 2. Default c(FALSE, TRUE). Allow genes to be missing for peak-gene connections? If both FALSE and TRUE are given, the code loops over both.

allowMissingTFs Logical vector of length 1 or 2. Default c(FALSE). Allow TFs to be missing for TF-peak connections? If both FALSE and TRUE are given, the code loops over both.

forceRerun TRUE or FALSE. Default FALSE. Force execution, even if the GRN object already contains the result. Overwrites the old results.

Value

An updated GRN object, with additional information added from this function.

See Also

plot_stats_connectionSummary
filterGRNAndConnectGenes

Examples

# See the Workflow vignette on the GRaNIE website for examples
GRN = loadExampleObject()
GRN = generateStatsSummary(GRN, TF_peak.fdr = c(0.01, 0.1), peak_gene.fdr = c(0.01, 0.1))

getCounts \( \text{Get counts for the various data defined in a GRN object} \)

Description

Get counts for the various data defined in a GRN object. Note: This function, as all get functions from this package, does NOT return a GRN object.

Usage

getCounts(
  GRN,
  type,
  permuted = FALSE,
  asMatrix = FALSE,
  includeIDColumn = TRUE,
  includeFiltered = FALSE
)

getGRNConnections

Arguments

GRN  Object of class GRN

type  Character. Either peaks or rna. peaks corresponds to the counts for the open
chromatin data, while rna refers to the RNA-seq counts. If set to rna, both real
(_foreground) and background data can be retrieved, while for peaks, only the
real (i.e., the one with index 0) can be retrieved.

permuted  TRUE or FALSE. Default FALSE. Should the permuted data be taken (TRUE) or the
non-permuted, original one (FALSE)?

asMatrix  Logical. TRUE or FALSE. Default FALSE. If set to FALSE, counts are returned as
a data frame with or without an ID column (see includeIDColumn). If set to
TRUE, counts are returned as a matrix with the ID column as row names.

includeIDColumn  Logical. TRUE or FALSE. Default TRUE. Only relevant if asMatrix = FALSE. If
set to TRUE, an explicit ID column is returned (no row names). If set to FALSE,
the IDs are in the row names instead.

includeFiltered  Logical. TRUE or FALSE. Default FALSE. If set to FALSE, genes or peaks marked
as filtered (after running the function filterData) will not be returned. If set to
TRUE, all elements are returned regardless of the currently active filter status.

Value

Data frame of counts, with the type as indicated by the function parameters. This function does
**NOT** return a GRN object.

Examples

# See the Workflow vignette on the GRaNIE website for examples
GRN = loadExampleObject()
counts.df = getCounts(GRN, type = "peaks", permuted = FALSE)

getGRNConnections  Extract connections or links from a GRN object as a data frame.

Description

Returns stored connections/links (either TF-peak, peak-genes, TF-genes or the filtered set of con-
nections as produced by filterGRNAndConnectGenes). Additional meta columns (TF, peak and
gene metadata) can be added optionally. **Note: This function, as all get functions from this
package, does NOT return a GRN object.**
getGRNConnections

Usage

getGRNConnections(
  GRN,
  type = "all.filtered",
  background = FALSE,
  include_TF_gene_correlations = FALSE,
  include_TFMetadata = FALSE,
  include_peakMetadata = FALSE,
  include_geneMetadata = FALSE,
  include_variancePartitionResults = FALSE
)

Arguments

GRN Object of class GRN

type Character. One of TF_peaks, peak_genes, TF_genes or all.filtered. Default all.filtered. The type of connections to retrieve.

background Integer (0 or 1). Default 0. Either 0 or 1). Here, 0 refers to the real (foreground) while 1 to the background.

include_TF_gene_correlations Logical. TRUE or FALSE. Default FALSE. Should TFs and gene correlations be returned as well? If set to TRUE, they must have been computed beforehand with add_TF_gene_correlation.

include_TFMetadata Logical. TRUE or FALSE. Default FALSE. Should TF metadata be returned as well?

include_peakMetadata Logical. TRUE or FALSE. Default FALSE. Should peak metadata be returned as well?

include_geneMetadata Logical. TRUE or FALSE. Default FALSE. Should gene metadata be returned as well?

include_variancePartitionResults Logical. TRUE or FALSE. Default FALSE. Should the results from the function add_featureVariation be included? If set to TRUE, they must have been computed beforehand with add_featureVariation; otherwise, an error is thrown.

Value

A data frame with the requested connections. This function does **NOT** return a GRN object. Depending on the arguments, the data frame that is returned has different columns, which however can be divided into the following classes according to their name:

- TF-related: Starting with TF.:
  - TF.name and TF.ID: Name / ID of the TF
  - TF.ENSEMBL: Ensembl ID (unique)

- peak-related: Starting with peak.:
- peak.ID: ID (coordinates)
- peak.mean, peak.median, peak.CV: peak mean, median and its coefficient of variation (CV) across all samples
- peak.annotation: Peak annotation as determined by ChIPseeker such as Promoter, 5' UTR, 3' UTR, Exon, Intron, Downstream, Intergenic
- peak.nearestGene*: Additional metadata for the nearest gene such as position (chr, start, end, strand), name (name, symbol and ENSEMBL), and distance to the TSS (distanceToTSS)
- peak.GC.perc: GC percentage

- gene-related: Starting with gene.:  
  - gene.name and gene.ENSEMBL: gene name and Ensembl ID  
  - gene.type: gene type (such as protein_coding, lincRNA) as retrieved by biomaRt  
  - gene.mean, gene.median, gene.CV: gene mean, median and its coefficient of variation (CV) across all samples

- TF-peak-related: Starting with TF_peak.:  
  - TF_peak.r and TF_peak.r_bin: Correlation coefficient of the TF-peak pair and its correlation bin (in bins of width 0.05, such as (-0.55,-0.5] for r = -0.53)  
  - TF_peak.fdr and TF_peak.fdr_direction: TF-peak FDR and the directionality from which it was derived (see Methods in the paper, pos or neg)  
  - TF_peak.connectionType: TF-peak connection type. This is by default expression, meaning that expression was used to construct the TF and peak

- peak-gene-related: Starting with peak_gene.:  
  - peak_gene.distance: Peak-gene distance (usually taken the TSS of the gene as reference unless specified otherwise, see the parameter overlapTypeGene for more information from addConnections_peak_gene)  
  - peak_gene.r: Correlation coefficient of the peak-gene pair  
  - peak_gene.p_raw and peak_gene.p_adj: Raw and adjusted p-value of the peak-gene pair

- TF-gene-related: Starting with TF_gene.:  
  - TF_gene.r: Correlation coefficient of the TF-gene pair  
  - TF_gene.p_raw: Raw p-value of the TF-gene pair

See Also

filterGRNAndConnectGenes
add_featureVariation
add_TF_gene_correlation

Examples

# See the Workflow vignette on the GRaNIE website for examples
GRN = loadExampleObject()
GRN_con.all.df = getGRNConnections(GRN)
getGRNSummary

Summarize a GRN object to a named list for comparison with other GRN objects.

Description

Note: This function, as all get functions from this package, does NOT return a GRN object.

Usage

getGRNSummary(GRN, silent = FALSE)

Arguments

GRN Object of class GRN
silent TRUE or FALSE. Default FALSE. Should the function be silent and print nothing?

Value

A named list summarizing the GRN object. This function does **NOT** return a GRN object, but instead a named list with the following elements:

- data:
  - peaks, genes and TFs:
  - sharedSamples:
  - metadata:
- parameters and config: GRN parameters and config information
- connections: Connection summary for different connection types
  - TF_peak: TF-peak (number of connections for different FDR thresholds)
  - peak_genes: Peak-gene
  - TF_peak_gene: TF-peak-gene
- network: Network-related summary, including the number of nodes, edges, communities and enrichment for both the TF-peak-gene and TF-gene network
  - TF_gene
  - TF_peak_gene

Examples

# See the Workflow vignette on the GRaNIE website for examples
GRN = loadExampleObject()
summary.l = getGRNSummary(GRN)
getParameters

Retrieve parameters for previously used function calls and general parameters for a GRN object.

Description

Note: This function, as all get functions from this package, does NOT return a GRN object.

Usage

getParameters(GRN, type = "parameter", name = "all")

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GRN</td>
<td>Object of class GRN</td>
</tr>
<tr>
<td>type</td>
<td>Character. Either function or parameter. Default parameter. When set to function, a valid GRaNIe function name must be given that has been run before. When set to parameter, in combination with name, returns a specific parameter (as specified in GRN@config).</td>
</tr>
<tr>
<td>name</td>
<td>Character. Default all. Name of parameter or function name to retrieve. Set to the special keyword all to retrieve all parameters.</td>
</tr>
</tbody>
</table>

Value

The requested parameters. This function does **NOT** return a GRN object.

Examples

# See the Workflow vignette on the GRaNIe website for examples
GRN = loadExampleObject()
params.l = getParameters(GRN, type = "parameter", name = "all")

gTopNodes

Retrieve the top nodes (TFs or genes) with respect to either degree or Eigenvector centrality in the filtered GRN object.

Description

This function requires a filtered set of connections in the GRN object as generated by filterGRNAndConnectGenes. Note: This function, as all get functions from this package, does NOT return a GRN object.

Usage

gTopNodes(GRN, nodeType, rankType, n = 0.1, use_TF_gene_network = TRUE)
Arguments

GRN Object of class GRN
type Character. One of: "gene" or "TF". Node type.
rankType Character. One of: "degree", "EV". This parameter will determine the criterion to be used to identify the "top" nodes. If set to "degree", the function will select top nodes based on the number of connections they have, i.e. based on their degree-centrality. If set to "EV" it will select the top nodes based on their eigenvector-centrality score in the network.
n Numeric. Default 0.1. If this parameter is passed as a value between [0,1], it is treated as a percentage of top nodes. If the value is passed as an integer >=1 it will be treated as the number of top nodes.

use_TF_gene_network TRUE or FALSE. Default TRUE. Should the TF-gene network be used (TRUE) or the TF-peak-gene network (FALSE)?

Value

A data frame with the node names and the corresponding scores used to rank them

Examples

# See the Workflow vignette on the GRaNIE website for examples
GRN = loadExampleObject()
topGenes = getTopNodes(GRN, nodeType = "gene", rankType = "degree", n = 3)
topTFs = getTopNodes(GRN, nodeType = "TF", rankType = "EV", n = 5)

Description

Genetic variants associated with diseases often affect non-coding regions, thus likely having a regulatory role. To understand the effects of genetic variants in these regulatory regions, identifying genes that are modulated by specific regulatory elements (REs) is crucial. The effect of gene regulatory elements, such as enhancers, is often cell-type specific, likely because the combinations of transcription factors (TFs) that are regulating a given enhancer have celltype specific activity. This TF activity can be quantified with existing tools such as diffTF and captures differences in binding of a TF in open chromatin regions. Collectively, this forms a gene regulatory network (eGRN) with cell-type and data-specific TF-RE and RE-gene links. Here, we reconstruct such a eGRN using bulk RNAseq and open chromatin (e.g., using ATACseq or ChIPseq for open chromatin marks) and optionally TF activity data. Our network contains different types of links, connecting TFs to regulatory elements, the latter of which is connected to genes in the vicinity or within the same chromatin domain (TAD). We use a statistical framework to assign empirical FDRs and weights to all links using a permutation-based approach.
GRN-class

Package functions

See the Vignettes for a workflow example and more generally https://grp-zaugg.embl-community.io/GRaNIE for all project-related information.

GRN object

The GRaNIE package works with GRN objects. See GRN for details.

Contact Information

Please check out https://grp-zaugg.embl-community.io/GRaNIE for how to get in contact with us.

<table>
<thead>
<tr>
<th>GRN-class</th>
<th>Create, represent, investigate, quantify and visualize enhancer-mediated gene regulatory networks (eGRNs)</th>
</tr>
</thead>
</table>

Description

The class GRN stores data and information related to our eGRN approach to construct enhancer-mediated gene regulatory networks out of open chromatin and RNA-Seq data. See the description below for more details, and visit our project website at https://grp-zaugg.embl-community.io/GRaNIE and have a look at the various Vignettes.

Slots

data  Currently stores 4 different types of data:

- peaks:
  - counts:
  - counts_metadata:
- RNA:
  - counts:
  - counts_metadata:
  - counts_permuted_index:
- TFs:
  - TF_activity:
  - TF_peak_overlap:
  - classification:

config  Contains general configuration data and parameters such as parameters, files, directories, flags, and recorded function parameters.

connections  Stores various types of connections

annotation  Stores annotation data for peaks and genes

stats  Stores statistical and summary information for a GRN network. Currently, connection details are stored here.

graph  Stores the eGRN graph related information and data structures
Constructors

Currently, a GRN object is created by executing the function `initializeGRN`.

Accessors

In the following code snippets, GRN is a GRN object.

```r
# Get general annotation of a GRN object from the GRaNIE package
nPeaks(GRN), nTFs(GRN) and nGenes(GRN)): Retrieve the number of peaks, TFs and genes, respectively, that have been added to the object (both before and after filtering)
```

**importTFData**

*Import externally derived TF Activity data. EXPERIMENTAL.*

**Description**

We do not yet provide full support for this function. It is currently being tested. Use at our own risk.

**Usage**

```r
importTFData( 
  GRN, 
  data, 
  name, 
  idColumn = "ENSEMBL", 
  nameColumn = "TF.name", 
  normalization = "none", 
  forceRerun = FALSE 
)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GRN</td>
<td>Object of class GRN</td>
</tr>
<tr>
<td>data</td>
<td>Data frame. No default. Data with TF data.</td>
</tr>
<tr>
<td>name</td>
<td>Name in object under which it should be stored. This corresponds to the connectionType afterwards that some functions iterate over.</td>
</tr>
<tr>
<td>idColumn</td>
<td>Character. Default ENSEMBL. Name of the ID column. Must not be unique as some TFs may correspond to the same ID.</td>
</tr>
<tr>
<td>nameColumn</td>
<td>Character. Default TF.name. Must be unique for each TF / row.</td>
</tr>
<tr>
<td>normalization</td>
<td>Character. Default cyclicLoess. One of cyclicLoess, sizeFactors, quantile, or none. Normalization procedure. When set to cyclicLoess, the csaw package is required (as it is listed under Suggests, it may not be installed).</td>
</tr>
<tr>
<td>forceRerun</td>
<td>TRUE or FALSE. Default FALSE. Force execution, even if the GRN object already contains the result. Overwrites the old results.</td>
</tr>
</tbody>
</table>
**initializeGRN**

Create and initialize an empty GRN object.

**Description**

Executing this function is the very first step in the *GRaNIE* workflow. After its execution, data can be added to the object.

**Usage**

initializeGRN(objectMetadata = list(), outputFolder = ".", genomeAssembly)

**Arguments**

- **objectMetadata** List. Default list(). Optional (named) list with an arbitrary number of elements, all of which capture metadata for the object. **Only atomic data types are allowed for each list element (see ?is.atomic for more help: logical, integer, numeric, complex, character, raw, and NULL), and this slot is not supposed to store real data.** This is mainly used to distinguish GRN objects from one another by storing object-specific metadata along with the data.

- **outputFolder** Output folder, either absolute or relative to the current working directory. Default ".". Default output folder where all pipeline output will be put unless specified otherwise. We recommend specifying an absolute path. Note that for Windows-based systems, the path must be correctly specified with "/" as path separator.

- **genomeAssembly** Character. No default. The genome assembly of all data that to be used within this object. Currently, supported genomes are: hg19, hg38, mm9, mm10, rn6, rn7, dm6. If you need additional genomes, let us know. See function description for further information and notes.

**Value**

Empty GRN object

**Examples**

```r
meta.l = list(name = "exampleName", date = "01.03.22")
GRN = initializeGRN(objectMetadata = meta.l, outputFolder = "output", genomeAssembly = "hg38")
```
loadExampleObject **Load example GRN dataset**

**Description**

Loads an example GRN object with 6 TFs, ~61,000 peaks, ~19,000 genes, 259 filtered connections and pre-calculated enrichments. This function uses BiocFileCache if installed to cache the example object, which is considerably faster than re-downloading the file anew every time the function is executed. If not, the file is re-downloaded every time anew. Thus, to enable caching, you may install the package BiocFileCache.

**Usage**

```r
loadExampleObject(
  forceDownload = FALSE,
  fileURL = "https://git.embl.de/grp-zaugg/GRaNIE/-/raw/master/data/GRN.rds"
)
```

**Arguments**

- `forceDownload` TRUE or FALSE. Default FALSE. Should the download be enforced even if the local cached file is already present?
- `fileURL` Character. Default https://git.embl.de/grp-zaugg/GRaNIE/-/raw/master/data/GRN.rds. URL to the GRN example object in rds format.

**Value**

An small example GRN object

**Examples**

```r
GRN = loadExampleObject()
```

---

nGenes **Get the number of genes for a GRN object.**

**Description**

Returns the number of genes (all or only non-filtered ones) from the provided RNA-seq data in the GRN object.

**Usage**

```r
nGenes(GRN, filter = TRUE)
```
**nPeaks**

*Get the number of peaks for a GRN object.*

**Arguments**

- **GRN**: Object of class `GRN`  
- **filter**: TRUE or FALSE. Default TRUE. Should peaks marked as filtered be included in the count?

**Value**

Integer. Number of peaks that are defined in the GRN object, either by excluding (filter = TRUE) or including (filter = FALSE) peaks that are currently marked as filtered.

**See Also**

- `nTFs`
- `nPeaks`

**Examples**

```r
# See the Workflow vignette on the GRaNIE website for examples
GRN = loadExampleObject()
nGenes(GRN, filter = TRUE)
nGenes(GRN, filter = FALSE)
```

---

**nPeaks**

Get the number of peaks for a GRN object.

**Description**

Returns the number of peaks (all or only non-filtered ones) from the provided peak data in the GRN object.

**Usage**

```r
nPeaks(GRN, filter = TRUE)
```

**Arguments**

- **GRN**: Object of class `GRN`  
- **filter**: TRUE or FALSE. Default TRUE. Should peaks marked as filtered be included in the count?

**Value**

Integer. Number of peaks that are defined in the GRN object, either by excluding (filter = TRUE) or including (filter = FALSE) peaks that are currently marked as filtered.
nTFs

Get the number of TFs for a GRN object.

Description

Returns the number of TFs from the provided TFBS data in the GRN object.

Usage

nTFs(GRN)

Arguments

GRN Object of class GRN

Value

Integer. Number of TFs that are defined in the GRN object.

See Also

nGenes
nPeaks

Examples

# See the Workflow vignette on the GRaNIE website for examples
GRN = loadExampleObject()
nPeaks(GRN, filter = TRUE)
nPeaks(GRN, filter = FALSE)
overlapPeaksAndTFBS  

Overlap peaks and TFBS for a GRN object

Description

If the source was set to JASPAR in addTFBS, the argument nCores is ignored.

Usage

overlapPeaksAndTFBS(GRN, nCores = 2, forceRerun = FALSE, ...)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GRN</td>
<td>Object of class GRN</td>
</tr>
<tr>
<td>nCores</td>
<td>Integer &gt;0. Default 1. Number of cores to use. A value &gt;1 requires the BiocParallel package (as it is listed under Suggests, it may not be installed yet).</td>
</tr>
<tr>
<td>forceRerun</td>
<td>TRUE or FALSE. Default FALSE. Force execution, even if the GRN object already contains the result. Overwrites the old results.</td>
</tr>
<tr>
<td>...</td>
<td>No default. Only relevant if source = &quot;JASPAR&quot; has been selected in addTFBS, ignored otherwise. Additional arguments for motifmatchr::matchMotifs such as custom background nucleotide frequencies or p-value cutoffs. For more information, type ?motifmatchr::matchMotifs</td>
</tr>
</tbody>
</table>

Value

An updated GRN object, with added data from this function (GRN@data$TFs$TF_peak_overlap in particular)

Examples

# See the Workflow vignette on the GRaNIE website for examples
GRN = loadExampleObject()
GRN = overlapPeaksAndTFBS(GRN, nCores = 2, forceRerun = FALSE)

performAllNetworkAnalyses

Perform all network-related statistical and descriptive analyses, including community and enrichment analyses. See the functions it executes in the @seealso section below.

Description

A convenience function that calls all network-related functions in one-go, using selected default parameters and a set of adjustable ones also. For full adjustment, run the individual functions separately. This function requires a filtered set of connections in the GRN object as generated by filterGRNAndConnectGenes
Usage

performAllNetworkAnalyses(
  GRN,
  ontology = c("GO_BP", "GO_MF"),
  algorithm = "weight01",
  statistic = "fisher",
  background = "neighborhood",
  clustering = "louvain",
  communities = NULL,
  selection = "byRank",
  topnGenes = 20,
  topnTFs = 20,
  maxWidth_nchar_plot = 50,
  display_pAdj = FALSE,
  outputFolder = NULL,
  forceRerun = FALSE
)

Arguments

GRN Object of class \texttt{GRN}

ontology Character vector of ontologies. Default \(c(\text{"GO\_BP"}, \text{"GO\_MF"})\). Valid values are \"GO\_BP\", \"GO\_MF\", \"GO\_CC\", \"KEGG\", \"DO\", and \"Reactome\", referring to \textit{GO Biological Process}, \textit{GO Molecular Function}, \textit{GO Cellular Component}, \textit{KEGG}, \textit{Disease Ontology}, and \textit{Reactome Pathways}, respectively. \textit{GO} ontologies require the \texttt{topGO}, \"KEGG\" the \texttt{clusterProfiler}, \"DO\" the \texttt{DOSE}, and \"Reactome\" the \texttt{ReactomePA} packages, respectively. As they are listed under \textit{Suggests}, they may not yet be installed, and the function will throw an error if they are missing.

algorithm Character. Default \"weight01\". One of: \"classic\", \"elim\", \"weight\", \"weight01\", \"lea\", \"parentchild\". Only relevant if ontology is \textit{GO} related (\texttt{GO\_BP}, \texttt{GO\_MF}, \texttt{GO\_CC}), ignored otherwise. Name of the algorithm that handles the \textit{GO} graph structures. Valid inputs are those supported by the \texttt{topGO} library. For general information about the algorithms, see \url{https://academic.oup.com/bioinformatics/article/22/13/1600/193669}. \texttt{weight01} is a mixture between the \texttt{elim} and \texttt{weight} algorithms.

statistic Character. Default \"fisher\". One of: \"fisher\", \"ks\", \"t\". Statistical test to be used. Only relevant if ontology is \textit{GO} related (\texttt{GO\_BP}, \texttt{GO\_MF}, \texttt{GO\_CC}), and valid inputs are a subset of those supported by the \texttt{topGO} library (we had to remove some as they do not seem to work properly in \texttt{topGO} either), ignored otherwise. For the other ontologies the test statistic is always Fisher.

background Character. Default \"neighborhood\". One of: \"all\_annotated\", \"all\_RNA\", \"all\_RNA\_filtered\", \"neighborhood\". Set of genes to be used to construct the background for the enrichment analysis. This can either be all annotated genes in the reference genome (\texttt{all\_annotated}), all genes from the provided RNA data (\texttt{all\_RNA}), all genes from the provided RNA data excluding those marked as filtered after executing \texttt{filterData} (\texttt{all\_RNA\_filtered}), or all the genes that are within the neighborhood of any peak (before applying any filters
except for the user-defined promoterRange value in addConnections_peak_gene (neighborhood).

**clustering**
Character. Default louvain. One of: louvain, leiden, leading_eigen, fast_greedy, optimal, walktrap. The community detection algorithm to be used. Please bear in mind the robustness and time consumption of the algorithms when opting for an alternative to the default.

**communities**
NULL or numeric vector or character vector. Default NULL. If set to NULL, all community enrichments that have been calculated before are plotted. If a numeric vector is specified (when selection = "byRank"), the rank of the communities is specified. For example, communities = c(1,4) then denotes the first and fourth largest community. If a character vector is specified (when selection = "byLabel"), the name of the communities is specified instead. For example, communities = c("1","4") then denotes the communities with the names "1" and "4", which may or may not be the largest and fourth largest communities among all.

**selection**
Character. Default "byRank". One of: "byRank", "byLabel". Specify whether the communities will be selected based on their rank or explicitly by their label. Note that the label is independent of the rank. When set to "byRank", the largest community (with most vertices) always has a rank of 1.

**topnGenes**
Integer > 0. Default 20. Number of genes to plot, sorted by their rank or label.

**topnTFs**
Integer > 0. Default 20. Number of TFs to plot, sorted by their rank or label.

**maxWidth_nchar_plot**
Integer (>=10). Default 50. Maximum number of characters for a term before it is truncated.

**display_pAdj**
TRUE or FALSE. Default FALSE. Is the p-value being displayed in the plots the adjusted p-value? This parameter is relevant for KEGG, Disease Ontology, and Reactome enrichments, and does not affect GO enrichments.

**outputFolder**
Character or NULL. Default NULL. If set to NULL, the default output folder as specified when initiating the object in initializeGRN will be used. Otherwise, all output from this function will be put into the specified folder. If a folder is provided, while we recommend specifying an absolute path, a relative one also works.

**forceRerun**
TRUE or FALSE. Default FALSE. Force execution, even if the GRN object already contains the result. Overwrites the old results.

**Value**
An updated GRN object, with added data from this function.

**See Also**
build_eGRN_graph
plotGeneralGraphStats
calculateGeneralEnrichment
plotGeneralEnrichment
plotCommunitiesEnrichment

Examples

# See the Workflow vignette on the GRaNIE website for examples
# GRN = loadExampleObject()
# GRN = performAllNetworkAnalyses(GRN, outputFolder = ".", forceRerun = FALSE)

plotCommunitiesEnrichment

Plot community-based enrichment results for a filtered GRN object

Description

Similarly to plotGeneralEnrichment and plotTFEnrichment, the results of the community-based enrichment analysis are plotted. This function produces multiple plots. First, one plot per community to summarize the community-specific enrichment. Second, a summary heatmap of all significantly enriched terms across all communities and for the whole eGRN. The latter allows to compare the results with the general network enrichment. Third, a subset of the aforementioned heatmap, showing only the top most significantly enriched terms per community and for the whole eGRN (as specified by nID) for improved visibility

Usage

plotCommunitiesEnrichment(
  GRN,
  outputFolder = NULL,
  basenameOutput = NULL,
  selection = "byRank",
  communities = NULL,
  topn_pvalue = 30,
  p = 0.05,
  nSignificant = 2,
  nID = 10,
  maxWidth_nchar_plot = 50,
  display_pAdj = FALSE,
  plotAsPDF = TRUE,
  pdf_width = 12,
  pdf_height = 12,
  pages = NULL,
  forceRerun = FALSE
)
Arguments

**GRN**
Object of class `GRN`

**outputFolder**
Character or NULL. Default NULL. If set to NULL, the default output folder as specified when initiating the object in `initializeGRN` will be used. Otherwise, all output from this function will be put into the specified folder. If a folder is provided, while we recommend specifying an absolute path, a relative one also works.

**basenameOutput**
NULL or character. Default NULL. Basename of the output files that are produced. If set to NULL, a default basename is chosen. If a custom basename is specified, all output files will have the chosen basename as file prefix, be careful with not overwriting already existing files (if `forceRerun` is set to TRUE)

**selection**
Character. Default "byRank". One of: "byRank", "byLabel". Specify whether the communities will be selected based on their rank or explicitly by their label. Note that the label is independent of the rank. When set to "byRank", the largest community (with most vertices) always has a rank of 1.

**communities**
NULL or numeric vector or character vector. Default NULL. If set to NULL, all community enrichments that have been calculated before are plotted. If a numeric vector is specified (when `selection = "byRank"`), the rank of the communities is specified. For example, `communities = c(1, 4)` then denotes the first and fourth largest community. If a character vector is specified (when `selection = "byLabel"`), the name of the communities is specified instead. For example, `communities = c("1", "4")` then denotes the communities with the names "1" and "4", which may or may not be the largest and fourth largest communities among all.

**topn_pvalue**
Numeric. Default 30. Maximum number of ontology terms that meet the p-value significance threshold to display in the enrichment dot plot

**p**
Numeric. Default 0.05. p-value threshold to determine significance.

**nSignificant**
Numeric > 0. Default 3. Threshold to filter out an ontology term with less than `nSignificant` overlapping genes.

**nID**
Numeric > 0. Default 10. For the reduced summary heatmap, number of top terms to select per community / for the general enrichment.

**maxWidth_nchar_plot**
Integer (>=10). Default 50. Maximum number of characters for a term before it is truncated.

**display_pAdj**
TRUE or FALSE. Default FALSE. Is the p-value being displayed in the plots the adjusted p-value? This parameter is relevant for KEGG, Disease Ontology, and Reactome enrichments, and does not affect GO enrichments.

**plotAsPDF**
TRUE or FALSE. Default TRUE. Should the plots be printed to a PDF file? If set to TRUE, a PDF file is generated, the name of which depends on the value of `basenameOutput`. If set to FALSE, all plots are printed to the currently active device. Note that most functions print more than one plot, which means you may only see the last plot depending on your active graphics device.

**pdf_width**
Number>0. Default 12. Width of the PDF, in cm.

**pdf_height**
Number >0. Default 12. Height of the PDF, in cm.
plotCommunitiesStats

Description

Similarly to the statistics produced by plotGeneralGraphStats, summaries regarding the vertex degrees and the most important vertices per community are generated. Note that the communities need to first be calculated using the calculateCommunitiesStats function.

Usage

plotCommunitiesStats(  
  GRN,  
  outputFolder = NULL,  
  basenameOutput = NULL,  
  selection = "byRank",  
  communities = seq_len(5),  
  topnGenes = 20,  
  topnTFs = 20,  
  plotAsPDF = TRUE,  
  pdf_width = 12,  
  pdf_height = 12,  
  pages = NULL,  
  forceRerun = FALSE
)

Value

The same GRN object, without modifications.

See Also

plotGeneralEnrichment  
plotTFEnrichment  
calculateCommunitiesEnrichment

Examples

# See the Workflow vignette on the GRaNIE website for examples  
GRN = loadExampleObject()  
GRN = plotCommunitiesEnrichment(GRN, plotAsPDF = FALSE, pages = 1)

---

plotCommunitiesStats  
Plot general structure & connectivity statistics for each community in
a filtered GRN

---

pages  
Integer vector or NULL. Default NULL. Page number(s) to plot. Can be used to plot only specific pages to a PDF or the currently active graphics device.

forceRerun  
TRUE or FALSE. Default FALSE. Force execution, even if the GRN object already contains the result. Overwrites the old results.
Arguments

GRN
Object of class GRN

outputFolder
Character or NULL. Default NULL. If set to NULL, the default output folder as specified when initiating the object in initializeGRN will be used. Otherwise, all output from this function will be put into the specified folder. If a folder is provided, while we recommend specifying an absolute path, a relative one also works.

basenameOutput
NULL or character. Default NULL. Basename of the output files that are produced. If set to NULL, a default basename is chosen. If a custom basename is specified, all output files will have the chosen basename as file prefix, be careful with not overwriting already existing files (if forceRerun is set to TRUE)

selection
Character. Default "byRank". One of: "byRank", "byLabel". Specify whether the communities will be selected based on their rank or explicitly by their label. Note that the label is independent of the rank. When set to "byRank", the largest community (with most vertices) always has a rank of 1.

communities
NULL or numeric vector or character vector. Default NULL. If set to NULL, all community enrichments that have been calculated before are plotted. If a numeric vector is specified (when selection = "byRank"), the rank of the communities is specified. For example, communities = c(1,4) then denotes the first and fourth largest community. If a character vector is specified (when selection = "byLabel"), the name of the communities is specified instead. For example, communities = c("1","4") then denotes the communities with the names "1" and "4", which may or may not be the largest and fourth largest communities among all.

topnGenes
Integer > 0. Default 20. Number of genes to plot, sorted by their rank or label.

topnTFs
Integer > 0. Default 20. Number of TFs to plot, sorted by their rank or label.

plotAsPDF
TRUE or FALSE. Default TRUE. Should the plots be printed to a PDF file? If set to TRUE, a PDF file is generated, the name of which depends on the value of basenameOutput. If set to FALSE, all plots are printed to the currently active device. Note that most functions print more than one plot, which means you may only see the last plot depending on your active graphics device.

pdf_width
Number>0. Default 12. Width of the PDF, in cm.

df_height
Number >0. Default 12. Height of the PDF, in cm.

pages
Integer vector or NULL. Default NULL. Page number(s) to plot. Can be used to plot only specific pages to a PDF or the currently active graphics device.

forceRerun
TRUE or FALSE. Default FALSE. Force execution, even if the GRN object already contains the result. Overwrites the old results.

Value

The same GRN object, without modifications.

See Also

plotGeneralGraphStats
plotCorrelations

The user can select multiple filters to plot only pairs of interest. The data that is shown is the same that has been used to construct the eGRN.

Usage

plotCorrelations(
  GRN,
  type = "all.filtered",
  TF.IDs = NULL,
  peak.IDs = NULL,
  gene.IDs = NULL,
  min_abs_r = 0,
  TF_peak_maxFDR = 0.2,
  peak_gene_max_rawP = 0.2,
  TF_gene_max_rawP = 0.2,
  nMax = 10,
  nSelectionType = "random",
  dataType = c("real"),
  corMethod = NULL,
  outputFolder = NULL,
  basenameOutput = NULL,
  plotAsPDF = TRUE,
  plotsPerPage = c(2, 2),
  pdf_width = 10,
  pdf_height = 8,
  forceRerun = FALSE
)

Arguments

GRN Object of class GRN
type Character(1). Default "all.filtered". Either "TF-peak", "peak-gene", "TF-gene", or "all.filtered". Default "all.filtered". Which connections to plot? "all.filtered" plots all TF-peak and peak-gene pairs from the filtered eGRN as stored in the object.

TF.IDs Character(). Default NULL. Character vector of TF IDs to include. If set to NULL, this filter will be ignored. Only applicable when type = "TF-peak" or type = "TF-gene".

peak.IDs Character(). Default NULL. Character vector of peak IDs to include. If set to NULL, this filter will be ignored. Only applicable when type = "TF-peak" or type = "peak-gene".

gene.IDs Character(). Default NULL. Character vector of gene IDs (Ensembl) to include. If set to NULL, this filter will be ignored. Only applicable when type = "TF-gene" or type = "peak-gene".

min_abs_r Numeric[0,1]. Default 0. Filter for all types of pairs: Minimum correlation coefficient (absolute value) required to include a particular pair.

TF_peak_maxFDR Numeric[0,1]. Default 0.2. Filter for TF-peak pairs: Which maximum FDR should a pair to plot have? Only applicable when type = "TF-peak".

peak_gene_max_rawP Numeric[0,1]. Default 0.2. Filter for peak-gene pairs: Which maximum FDR should a pair to plot have? Only applicable when type = "peak-gene".

TF_gene_max_rawP Numeric[0,1]. Default 0.2. Filter for TF-gene pairs: Which maximum FDR should a pair to plot have? Only applicable when type = "TF-gene".

nMax Numeric(1). Default 10. Filter for all types of pairs: maximum number of selected pairs that fulfill all other filters that should be plotted. If set to 0, this filter will be disabled and all pairs that fulfill the user-defined criteria will be plotted. If set to a value > 0, different pairs may be selected each time the function is run (if the total number of remaining pairs is large enough).

nSelectionType random or top. Default top. Only applicable if nMax is set to a value > 0. If set to top, only the top features will be plotted while random plots randomly selected features.

dataType Character vector. One of, or both of, real or background. Default real. For which type, real or background data, to produce the diagnostic plots?

corMethod Character. One of pearson, spearman or bicor. Default pearson. Method for calculating the correlation coefficient. For pearson and spearman, see cor for details. bicor denotes the *biweight midcorrelation*, a correlation measure based on medians as calculated by WGCNA::bicorAndPvalue. Both spearman and bicor are considered more robust measures that are less prone to be affected by outliers.

outputFolder Character or NULL. Default NULL. If set to NULL, the default output folder as specified when initiating the object in initializeGRN will be used. Otherwise, all output from this function will be put into the specified folder. If a folder is provided, while we recommend specifying an absolute path, a relative one also works.
Plot diagnostic plots for peak-gene connections for a GRN object

Usage

plotDiagnosticPlots_peakGene(
  GRN,
  outputFolder = NULL,
  basenameOutput = NULL,
  gene.types = list(c("all"), c("protein_coding")),
  useFiltered = FALSE,
  plotDetails = FALSE,
  plotPerTF = FALSE,
  plotAsPDF = TRUE,
)
```r
plotDiagnosticPlots_peakGene

pdf_width = 12,
pdf_height = 12,
pages = NULL,
forceRerun = FALSE
)
```

**Arguments**

- **GRN** Object of class `GRN`
- **outputFolder** Character or `NULL`. Default `NULL`. If set to `NULL`, the default output folder as specified when initiating the object in `initializeGRN` will be used. Otherwise, all output from this function will be put into the specified folder. If a folder is provided, while we recommend specifying an absolute path, a relative one also works.
- **basenameOutput** `NULL` or character. Default `NULL`. Basename of the output files that are produced. If set to `NULL`, a default basename is chosen. If a custom basename is specified, all output files will have the chosen basename as file prefix, be careful with not overwriting already existing files (if `forceRerun` is set to `TRUE`)
- **gene.types** List of character vectors. Default list(c("protein_coding", "lincRNA")). Vectors of gene types to consider for the diagnostic plots. Multiple distinct combinations of gene types can be specified. For example, if set to `list(c("protein_coding", "lincRNA"), c("protein_coding"), c("all"))`, 3 distinct PDFs will be produced, one for each element of the list. The first file would only consider protein-coding and lincRNA genes, while the second plot only considers protein-coding ones. The special keyword "all" denotes all gene types found (usually, there are many gene types present, also more exotic and rare ones).
- **useFiltered** Logical. `TRUE` or `FALSE`. Default `FALSE`. If set to `FALSE`, the diagnostic plots will be produced based on all peak-gene connections. This is the default and will usually be best to judge whether the background behaves as expected. If set to `TRUE`, the diagnostic plots will be produced based on the filtered set of connections. For this, the function `filterGRNAndConnectGenes` must have been run before.
- **plotDetails** `TRUE` or `FALSE`. Default `FALSE`. Print additional plots that may help for debugging and QC purposes? Note that these plots are currently less documented or not at all.
- **plotPerTF** Logical. `TRUE` or `FALSE`. Default `FALSE`. If set to `FALSE`, the diagnostic plots will be done across all TF (the default), while setting it to `TRUE` will generate the QC plots TF-specifically, including "all" TF, sorted by the number of connections.
- **plotAsPDF** `TRUE` or `FALSE`. Default `TRUE`. Should the plots be printed to a PDF file? If set to `TRUE`, a PDF file is generated, the name of which depends on the value of `basenameOutput`. If set to `FALSE`, all plots are printed to the currently active device. Note that most functions print more than one plot, which means you may only see the last plot depending on your active graphics device.
- **pdf_width** Number > 0. Default 12. Width of the PDF, in cm.
- **pdf_height** Number > 0. Default 12. Height of the PDF, in cm.
plotDiagnosticPlots_TFPeaks

Description
Plot diagnostic plots for TF-peak connections for a GRN object

Usage
plotDiagnosticPlots_TFPeaks(
    GRN,
    outputFolder = NULL,
    basenameOutput = NULL,
    plotDetails = FALSE,
    dataType = c("real", "background"),
    nTFMax = NULL,
    plotAsPDF = TRUE,
    pdf_width = 12,
    pdf_height_base = 8,
    pages = NULL,
    forceRerun = FALSE
)

Arguments
GRN Object of class GRN
outputFolder Character or NULL. Default NULL. If set to NULL, the default output folder as specified when initiating the object in initializeGRN will be used. Otherwise, all output from this function will be put into the specified folder. If a folder is provided, while we recommend specifying an absolute path, a relative one also works.
basenameOutput: NULL or character. Default NULL. Basename of the output files that are produced. If set to NULL, a default basename is chosen. If a custom basename is specified, all output files will have the chosen basename as file prefix, be careful with not overwriting already existing files (if forceRerun is set to TRUE).

plotDetails: TRUE or FALSE. Default FALSE. Print additional plots that may help for debugging and QC purposes? Note that these plots are currently less documented or not at all.

dataType: Character vector. One of, or both of, "real" or "background". For which type, real or background data, to produce the diagnostic plots?

nTFMax: NULL or Integer > 0. Default NULL. Maximum number of TFs to process. Can be used for testing purposes by setting this to a small number (e.g., 10).

plotAsPDF: TRUE or FALSE. Default TRUE. Should the plots be printed to a PDF file? If set to TRUE, a PDF file is generated, the name of which depends on the value of basenameOutput. If set to FALSE, all plots are printed to the currently active device. Note that most functions print more than one plot, which means you may only see the last plot depending on your active graphics device.

pdf_width: Number>0. Default 12. Width of the PDF, in cm.

df_height_base: Number. Default 8. Base height of the PDF, in cm, per connection type. The total height is automatically determined based on the number of connection types that are found in the object (e.g., expression or TF activity). For example, when two connection types are found, the base height is multiplied by 2.

pages: Integer vector or NULL. Default NULL. Page number(s) to plot. Can be used to plot only specific pages to a PDF or the currently active graphics device.

forceRerun: TRUE or FALSE. Default FALSE. Force execution, even if the GRN object already contains the result. Overwrites the old results.

Value:
An updated GRN object.

Examples:

# See the Workflow vignette on the GRaNIE website for examples
GRN = loadExampleObject()
GRN = plotDiagnosticPlots_TFPeaks(GRN, outputFolder = ".", dataType = "real", nTFMax = 2, pages = 1)

Description:

Note: The arguments nTFMax and pages are not implemented yet.
Usage

plotDiagnosticPlots_TFPeaks_GC(
  GRN,
  outputFolder = NULL,
  basenameOutput = NULL,
  dataType = c("real", "background"),
  nTFMax = NULL,
  plotAsPDF = TRUE,
  pdf_width = 12,
  pdf_height_base = 15,
  pages = NULL,
  forceRerun = FALSE
)

Arguments

GRN Object of class GRN

outputFolder Character or NULL. Default NULL. If set to NULL, the default output folder as specified when initiating the object in initializeGRN will be used. Otherwise, all output from this function will be put into the specified folder. If a folder is provided, while we recommend specifying an absolute path, a relative one also works.

basenameOutput NULL or character. Default NULL. Basename of the output files that are produced. If set to NULL, a default basename is chosen. If a custom basename is specified, all output files will have the chosen basename as file prefix, be careful with not overwriting already existing files (if forceRerun is set to TRUE)

dataType Character vector. One of, or both of, "real" or "background". For which type, real or background data, to produce the diagnostic plots?

nTFMax NULL or Integer > 0. Default NULL. Maximum number of TFs to process. Can be used for testing purposes by setting this to a small number (e.g., 10)

plotAsPDF TRUE or FALSE. Default TRUE. Should the plots be printed to a PDF file? If set to TRUE, a PDF file is generated, the name of which depends on the value of basenameOutput. If set to FALSE, all plots are printed to the currently active device. Note that most functions print more than one plot, which means you may only see the last plot depending on your active graphics device.

pdf_width Number>0. Default 12. Width of the PDF, in cm.

df_height_base Number>0. Default 8. Base height of the PDF, in cm, per connection type. The total height is automatically determined based on the number of connection types that are found in the object (e.g., expression or TF activity). For example, when two connection types are found, the base height is multiplied by 2.

pages Integer vector or NULL. Default NULL. Page number(s) to plot. Can be used to plot only specific pages to a PDF or the currently active graphics device.

forceRerun TRUE or FALSE. Default FALSE. Force execution, even if the GRN object already contains the result. Overwrites the old results.
plotGeneralEnrichment

Plot the general enrichment results

Description

This function plots the results of the general enrichment analysis for every specified ontology.

Usage

plotGeneralEnrichment(
  GRN,
  outputFolder = NULL,
  basenameOutput = NULL,
  ontology = NULL,
  topn_pvalue = 30,
  p = 0.05,
  display_pAdj = FALSE,
  maxWidth_nchar_plot = 50,
  plotAsPDF = TRUE,
  pdf_width = 12,
  pdf_height = 12,
  pages = NULL,
  forceRerun = FALSE
)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GRN</td>
<td>Object of class GRN</td>
</tr>
<tr>
<td>outputFolder</td>
<td>Character or NULL. Default NULL. If set to NULL, the default output folder as specified when initiating the object in initializeGRN will be used. Otherwise, all output from this function will be put into the specified folder. If a folder is provided, while we recommend specifying an absolute path, a relative one also works.</td>
</tr>
<tr>
<td>basenameOutput</td>
<td>NULL or character. Default NULL. Basename of the output files that are produced. If set to NULL, a default basename is chosen. If a custom basename is specified, all output files will have the chosen basename as file prefix, be careful with not overwriting already existing files (if forceRerun is set to TRUE)</td>
</tr>
<tr>
<td>ontology</td>
<td>Character. NULL or vector of ontology names. Default NULL. Vector of ontologies to plot. The results must have been previously calculated otherwise an error is thrown.</td>
</tr>
<tr>
<td>topn_pvalue</td>
<td>Numeric. Default 30. Maximum number of ontology terms that meet the p-value significance threshold to display in the enrichment dot plot</td>
</tr>
<tr>
<td>p</td>
<td>Numeric. Default 0.05. p-value threshold to determine significance.</td>
</tr>
<tr>
<td>display_pAdj</td>
<td>TRUE or FALSE. Default FALSE. Is the p-value being displayed in the plots the adjusted p-value? This parameter is relevant for KEGG, Disease Ontology, and Reactome enrichments, and does not affect GO enrichments.</td>
</tr>
</tbody>
</table>
plotGeneralGraphStats

maxWidth_nchar_plot
   Integer (>=10). Default 50. Maximum number of characters for a term before it is truncated.

plotAsPDF
   TRUE or FALSE. Default TRUE. Should the plots be printed to a PDF file? If set to TRUE, a PDF file is generated, the name of which depends on the value of basenameOutput. If set to FALSE, all plots are printed to the currently active device. Note that most functions print more than one plot, which means you may only see the last plot depending on your active graphics device.

pdf_width
   Number >0. Default 12. Width of the PDF, in cm.

pdf_height
   Number >0. Default 12. Height of the PDF, in cm.

classes
   Integer vector or NULL. Default NULL. Page number(s) to plot. Can be used to plot only specific pages to a PDF or the currently active graphics device.

forceRerun
   TRUE or FALSE. Default FALSE. Force execution, even if the GRN object already contains the result. Overwrites the old results.

Value

The same GRN object, without modifications.

See Also

plotCommunitiesEnrichment
plotTFEnrichment
calculateGeneralEnrichment

Examples

# See the Workflow vignette on the GRaNIE website for examples
GRN = loadExampleObject()
GRN = plotGeneralEnrichment(GRN, plotAsPDF = FALSE, pages = 1)

Description

This function generates graphical summaries about the structure and connectivity of the TF-peak-gene and TF-gene graphs. These include, distribution of vertex types (TF, peak, gene) and edge types (tf-peak, peak-gene), the distribution of vertex degrees, and the most "important" vertices according to degree centrality and eigenvector centrality scores.
Usage

plotGeneralGraphStats(
  GRN,
  outputFolder = NULL,
  basenameOutput = NULL,
  plotAsPDF = TRUE,
  pdf_width = 12,
  pdf_height = 12,
  pages = NULL,
  forceRerun = FALSE
)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GRN</td>
<td>Object of class <code>GRN</code></td>
</tr>
<tr>
<td>outputFolder</td>
<td>Character or NULL. Default NULL. If set to NULL, the default output folder as specified when initiating the object in <code>initializeGRN</code> will be used. Otherwise, all output from this function will be put into the specified folder. If a folder is provided, while we recommend specifying an absolute path, a relative one also works.</td>
</tr>
<tr>
<td>basenameOutput</td>
<td>NULL or character. Default NULL. Basename of the output files that are produced. If set to NULL, a default basename is chosen. If a custom basename is specified, all output files will have the chosen basename as file prefix, be careful with not overwriting already existing files (if <code>forceRerun</code> is set to <code>TRUE</code>)</td>
</tr>
<tr>
<td>plotAsPDF</td>
<td>TRUE or FALSE. Default TRUE. Should the plots be printed to a PDF file? If set to <code>TRUE</code>, a PDF file is generated, the name of which depends on the value of <code>basenameOutput</code>. If set to <code>FALSE</code>, all plots are printed to the currently active device. Note that most functions print more than one plot, which means you may only see the last plot depending on your active graphics device.</td>
</tr>
<tr>
<td>pdf_width</td>
<td>Number&gt;0. Default 12. Width of the PDF, in cm.</td>
</tr>
<tr>
<td>pdf_height</td>
<td>Number&gt;0. Default 12. Height of the PDF, in cm.</td>
</tr>
<tr>
<td>pages</td>
<td>Integer vector or NULL. Default NULL. Page number(s) to plot. Can be used to plot only specific pages to a PDF or the currently active graphics device.</td>
</tr>
<tr>
<td>forceRerun</td>
<td>TRUE or FALSE. Default FALSE. Force execution, even if the GRN object already contains the result. Overwrites the old results.</td>
</tr>
</tbody>
</table>

Value

The same `GRN` object, without modifications.

See Also

- `plotGeneralEnrichment`
- `plotCommunitiesStats`
- `plotCommunitiesEnrichment`
Examples

# See the Workflow vignette on the GRaNIE website for examples
GRN = loadExampleObject()
GRN = plotGeneralGraphStats(GRN, plotAsPDF = FALSE, pages = 1)

plotPCA_all

Produce a PCA plot of the data from a GRN object

Description

Produce a PCA plot of the data from a GRN object

Usage

plotPCA_all(
  GRN,
  outputFolder = NULL,
  basenameOutput = NULL,
  data = c("rna", "peaks"),
  topn = c(500, 1000, 5000),
  type = "normalized",
  removeFiltered = TRUE,
  plotAsPDF = TRUE,
  pdf_width = 12,
  pdf_height = 12,
  pages = NULL,
  forceRerun = FALSE
)

Arguments

GRN
  Object of class GRN

outputFolder
  Character or NULL. Default NULL. If set to NULL, the default output folder as specified when initiating the object in initializeGRN will be used. Otherwise, all output from this function will be put into the specified folder. If a folder is provided, while we recommend specifying an absolute path, a relative one also works.

basenameOutput
  NULL or character. Default NULL. Basename of the output files that are produced. If set to NULL, a default basename is chosen. If a custom basename is specified, all output files will have the chosen basename as file prefix, be careful with not overwriting already existing files (if forceRerun is set to TRUE)

data
  Character. Either "peaks" or "rna" or "all". Default c("rna", "peaks"). Type of data to plot a PCA for. "peaks" corresponds to the the open chromatin data, while "rna" refers to the RNA-seq counts. If set to "all", PCA will be done for both data modalities. In any case, PCA will be based on the original provided data before any additional normalization has been run (i.e., usually the raw data).
Description

Similarly to `plotGeneralEnrichment` and `plotCommunitiesEnrichment`, the results of the TF-based enrichment analysis are plotted. This function produces multiple plots. First, one plot per community to summarize the TF-specific enrichment. Second, a summary heatmap of all significantly enriched terms across all TFs and for the whole eGRN. The latter allows to compare the results with the general network enrichment. Third, a subset of the aforementioned heatmap, showing only the top most significantly enriched terms per TF and for the whole eGRN (as specified by nID) for improved visibility.
plotTFEnrichment

Usage

plotTFEnrichment(
  GRN,
  rankType = "degree",
  n = NULL,
  TF.IDs = NULL,
  topn_pvalue = 30,
  p = 0.05,
  nSignificant = 2,
  nID = 10,
  display_pAdj = FALSE,
  maxWidth_nchar_plot = 50,
  outputFolder = NULL,
  basenameOutput = NULL,
  plotAsPDF = TRUE,
  pdf_width = 12,
  pdf_height = 12,
  pages = NULL,
  forceRerun = FALSE
)

Arguments

GRN Object of class GRN

rankType Character. One of: "degree", "EV", "custom". This parameter will determine the criterion to be used to identify the “top” nodes. If set to "degree", the function will select top nodes based on the number of connections they have, i.e. based on their degree-centrality. If set to "EV" it will select the top nodes based on their eigenvector-centrality score in the network.

n NULL or numeric. Default NULL. If set to NULL, all previously calculated TF enrichments will be plotted. If set to a value between (0,1), it is treated as a percentage of top nodes. If the value is passed as an integer it will be treated as the number of top nodes. This parameter is not relevant if rankType = "custom".

TF.IDs NULL or character vector. Default NULL. For rankType="custom" the IDs of the TFs to plot. Ignored otherwise.

topn_pvalue Numeric. Default 30. Maximum number of ontology terms that meet the p-value significance threshold to display in the enrichment dot plot

p Numeric. Default 0.05. p-value threshold to determine significance.

nSignificant Numeric > 0. Default 3. Threshold to filter out an ontology term with less than nSignificant overlapping genes.

nID Numeric > 0. Default 10. For the reduced summary heatmap, number of top terms to select per community / for the general enrichment.

display_pAdj TRUE or FALSE. Default FALSE. Is the p-value being displayed in the plots the adjusted p-value? This parameter is relevant for KEGG, Disease Ontology, and Reactome enrichments, and does not affect GO enrichments.
maxWidth_nchar_plot
   Integer (>=10). Default 50. Maximum number of characters for a term before it is truncated.

outputFolder
   Character or NULL. Default NULL. If set to NULL, the default output folder as specified when initiating the object in initializeGRN will be used. Otherwise, all output from this function will be put into the specified folder. If a folder is provided, while we recommend specifying an absolute path, a relative one also works.

basenameOutput
   NULL or character. Default NULL. Basename of the output files that are produced. If set to NULL, a default basename is chosen. If a custom basename is specified, all output files will have the chosen basename as file prefix, be careful with not overwriting already existing files (if forceRerun is set to TRUE)

plotAsPDF
   TRUE or FALSE. Default TRUE. Should the plots be printed to a PDF file? If set to TRUE, a PDF file is generated, the name of which depends on the value of basenameOutput. If set to FALSE, all plots are printed to the currently active device. Note that most functions print more than one plot, which means you may only see the last plot depending on your active graphics device.

pdf_width
   Number>0. Default 12. Width of the PDF, in cm.

pdf_height
   Number >0. Default 12. Height of the PDF, in cm.

pages
   Integer vector or NULL. Default NULL. Page number(s) to plot. Can be used to plot only specific pages to a PDF or the currently active graphics device.

forceRerun
   TRUE or FALSE. Default FALSE. Force execution, even if the GRN object already contains the result. Overwrites the old results.

Value
   The same GRN object, without modifications.

See Also
   plotGeneralEnrichment
   plotCommunitiesEnrichment
   calculateTFEnrichment

Examples
   # See the Workflow vignette on the GReANIE website for examples
   GRN = loadExampleObject()
   GRN = plotTFEnrichment(GRN, n = 5, plotAsPDF = FALSE, pages = 1)
plot_stats_connectionSummary

*Plot various network connectivity summaries for a *GRN* object*

**Description**

Plot various network connectivity summaries for a *GRN* object

**Usage**

```r
plot_stats_connectionSummary(
  GRN,
  type = "heatmap",
  outputFolder = NULL,
  basenameOutput = NULL,
  plotAsPDF = TRUE,
  pdf_width = 12,
  pdf_height = 12,
  pages = NULL,
  forceRerun = FALSE
)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>GRN</strong></td>
<td>Object of class <em>GRN</em></td>
</tr>
<tr>
<td><strong>type</strong></td>
<td>Character. Either &quot;heatmap&quot; or &quot;boxplot&quot;. Default &quot;heatmap&quot;. Which plot type to produce?</td>
</tr>
<tr>
<td><strong>outputFolder</strong></td>
<td>Character or NULL. Default NULL. If set to NULL, the default output folder as specified when initiating the object in <em>initializeGRN</em> will be used. Otherwise, all output from this function will be put into the specified folder. If a folder is provided, while we recommend specifying an absolute path, a relative one also works.</td>
</tr>
<tr>
<td><strong>basenameOutput</strong></td>
<td>NULL or character. Default NULL. Basename of the output files that are produced. If set to NULL, a default basename is chosen. If a custom basename is specified, all output files will have the chosen basename as file prefix, be careful with not overwriting already existing files (if <em>forceRerun</em> is set to TRUE)</td>
</tr>
<tr>
<td><strong>plotAsPDF</strong></td>
<td>TRUE or FALSE. Default TRUE. Should the plots be printed to a PDF file? If set to TRUE, a PDF file is generated, the name of which depends on the value of basenameOutput. If set to FALSE, all plots are printed to the currently active device. Note that most functions print more than one plot, which means you may only see the last plot depending on your active graphics device.</td>
</tr>
<tr>
<td><strong>pdf_width</strong></td>
<td>Number&gt;0. Default 12. Width of the PDF, in cm.</td>
</tr>
<tr>
<td><strong>pdf_height</strong></td>
<td>Number &gt;0. Default 12. Height of the PDF, in cm.</td>
</tr>
<tr>
<td><strong>pages</strong></td>
<td>Integer vector or NULL. Default NULL. Page number(s) to plot. Can be used to plot only specific pages to a PDF or the currently active graphics device.</td>
</tr>
</tbody>
</table>
forceRerun: TRUE or FALSE. Default FALSE. Force execution, even if the GRN object already contains the result. Overwrites the old results.

Value
The same GRN object, without modifications.

Examples
# See the Workflow vignette on the GRaNIE website for examples
GRN = loadExampleObject()
GRN = plot_stats_connectionSummary(GRN, forceRerun = FALSE, plotAsPDF = FALSE, pages = 1)

Description
This function can visualize a filtered eGRN in a very flexible manner and requires a GRN object as generated by build_eGRN_graph.

Usage
visualizeGRN(
  GRN,
  outputFolder = NULL,
  basenameOutput = NULL,
  plotAsPDF = TRUE,
  pdf_width = 12,
  pdf_height = 12,
  title = NULL,
  maxEdgesToPlot = 500,
  nCommunitiesMax = 8,
  graph = "TF-gene",
  colorby = "type",
  layout = "fr",
  vertice_color_TFs = list(h = 10, c = 85, l = c(25, 95)),
  vertice_color_peaks = list(h = 135, c = 45, l = c(35, 95)),
  vertice_color_genes = list(h = 260, c = 80, l = c(30, 90)),
  vertexLabel_cex = 0.4,
  vertexLabel_dist = 0,
  forceRerun = FALSE
)
### Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GRN</td>
<td>Object of class GRN</td>
</tr>
<tr>
<td>outputFolder</td>
<td>Character or NULL. Default NULL. If set to NULL, the default output folder as specified when initiating the object in <code>initializeGRN</code> will be used. Otherwise, all output from this function will be put into the specified folder. If a folder is provided, while we recommend specifying an absolute path, a relative one also works.</td>
</tr>
<tr>
<td>basenameOutput</td>
<td>NULL or character. Default NULL. Basename of the output files that are produced. If set to NULL, a default basename is chosen. If a custom basename is specified, all output files will have the chosen basename as file prefix, be careful with not overwriting already existing files (if <code>forceRerun</code> is set to TRUE)</td>
</tr>
<tr>
<td>plotAsPDF</td>
<td>TRUE or FALSE. Default TRUE. Should the plots be printed to a PDF file? If set to TRUE, a PDF file is generated, the name of which depends on the value of <code>basenameOutput</code>. If set to FALSE, all plots are printed to the currently active device. Note that most functions print more than one plot, which means you may only see the last plot depending on your active graphics device.</td>
</tr>
<tr>
<td>pdf_width</td>
<td>Number &gt;0. Default 12. Width of the PDF, in cm.</td>
</tr>
<tr>
<td>pdf_height</td>
<td>Number &gt;0. Default 12. Height of the PDF, in cm.</td>
</tr>
<tr>
<td>title</td>
<td>NULL or Character. Default NULL. Title to be assigned to the plot.</td>
</tr>
<tr>
<td>maxEdgesToPlot</td>
<td>Integer &gt;0. Default 500. Refers to the maximum number of connections to be plotted. If the network size is above this limit, nothing will be drawn. In such a case, it may help to either increase the value of this parameter or set the filtering criteria for the network to be more stringent, so that the network becomes smaller.</td>
</tr>
<tr>
<td>nCommunitiesMax</td>
<td>Integer &gt;0. Default 8. Maximum number of communities that get a distinct coloring. All additional communities will be colored with the same (gray) color.</td>
</tr>
<tr>
<td>graph</td>
<td>Character. Default TF-gene. One of: TF-gene, TF-peak-gene. Whether to plot a graph with links from TFs to peaks to gene, or the graph with the inferred TF to gene connections.</td>
</tr>
<tr>
<td>colorby</td>
<td>Character. Default type. Either type or community. Color the vertices by either type (TF/peak/gene) or community. See <code>calculateCommunitiesStats</code></td>
</tr>
<tr>
<td>layout</td>
<td>Character. Default fr. One of star, fr, sugiyama, kk, lgl, graphopt, mds, sphere.</td>
</tr>
<tr>
<td>vertice_color_TFs</td>
<td>Named list. Default list(h = 10, c = 85, l = c(25, 95)). The list must specify the color in hcl format (hue, chroma, luminence). See the <code>colorspace</code> package for more details and examples</td>
</tr>
<tr>
<td>vertice_color_peaks</td>
<td>Named list. Default list(h = 135, c = 45, l = c(35, 95)).</td>
</tr>
<tr>
<td>vertice_color_genes</td>
<td>Named list. Default list(h = 260, c = 80, l = c(30, 90)).</td>
</tr>
<tr>
<td>vertexLabel_cex</td>
<td>Numeric. Default 0.4. Font size (multiplication factor, device-dependent)</td>
</tr>
</tbody>
</table>
vertexLabel_dist
   Numeric. Default 0 vertex. Distance between the label and the vertex.

forceRerun
   TRUE or FALSE. Default FALSE. Force execution, even if the GRN object already
   contains the result. Overwrites the old results.

Value
   The same GRN object, without modifications.

See Also
   build_eGRN_graph

Examples
   GRN = loadExampleObject()
   GRN = visualizeGRN(GRN, maxEdgesToPlot = 700, graph = "TF-gene", colorby = "type")
Index

* GRN-class, GRN-class, 45
* GRN
  GRN-class, 45
* GRaNIE, GRaNIE, 44
* GRaNIE-package
  GRaNIE, 44

add_featureVariation, 15, 40, 41
add_TF_gene_correlation, 17, 40, 41
addConnections_peak_gene, 4, 30, 36, 41
addConnections_TF_peak, 6, 30, 36
addData, 8, 16, 17, 30
addData_TFActivity, 7, 10
addSNPData, 11, 34
addTFBS, 13, 51
AR_classification_wrapper, 18, 29
build_eGRN_graph, 19, 20, 24, 26, 33, 36, 53, 73, 75

calculateCommunitiesEnrichment, 20, 23–26, 54, 56, 58
calculateCommunitiesStats, 20, 22, 23, 54, 56, 58, 74
calculateGeneralEnrichment, 20, 22, 24, 26, 53, 66
calculateTFEnrichment, 20, 24, 25, 26, 54, 71
changeOutputDirectory, 28
cor, 4, 7, 17, 19, 59
deleteIntermediateData, 29

filterConnectionsForPlotting, 29
filterData, 30
filterGRNAndConnectGenes, 16, 19, 20, 23, 32, 37–39, 41, 43, 51, 61
generateStatsSummary, 36
genres (nGenes), 48
getCounts, 38
getGRNConnections, 15–17, 33, 36, 39
getGRNSummary, 42
getParameters, 43
getTopNodes, 43
GRaNIE, 44
GRN, 4–8, 10–32, 34–40, 42–58, 60–75
GRN-class, 45

importTFData, 46
initializeGRN, 5, 6, 19, 34, 46, 47, 53, 55, 57, 59, 61, 62, 64, 65, 67, 68, 71, 72, 74
loadExampleObject, 48
nGenes, 48, 50
nPeaks, 49, 49, 50
nTFs, 49, 50, 50
overlapPeaksAndTFBS, 51

peaks (nPeaks), 49
performAllNetworkAnalyses, 51
plot_stats_connectionSummary, 37, 38, 72
plotCommunitiesEnrichment, 20, 22, 25, 54, 56, 66, 69, 71
plotCommunitiesStats, 23, 54, 56, 67
plotCorrelations, 58
plotDiagnosticPlots_peakGene, 4, 5, 60
plotDiagnosticPlots_TFPeaks, 6, 8, 62
plotDiagnosticPlots_TFPeaks_GC, 63
plotGeneralEnrichment, 22, 24, 25, 53, 54, 56, 65, 67, 69, 71
plotGeneralGraphStats, 53, 56, 57, 66
plotPCA_all, 10, 68
plotTFEnrichment, 26, 28, 54, 56, 66, 69
TFs (nTFs), 50
visualizeGRN, 36, 73

76