Package 'Moonlight2R'

March 23, 2024

Type Package

Title Identify oncogenes and tumor suppressor genes from omics data

Version 1.0.0

Date

Depends R (>= 4.3), doParallel, foreach

Imports parmigene, randomForest, gplots, circlize, RColorBrewer, HiveR, clusterProfiler, DOSE, Biobase, grDevices, graphics, GEOquery, stats, purrr, RISmed, grid, utils, ComplexHeatmap, GenomicRanges, dplyr, fuzzyjoin, rtracklayer, magrittr, qpdf, readr, seqminer, stringr, tibble, tidyHeatmap, tidyr, AnnotationHub, easyPubMed, org.Hs.eg.db

Description The understanding of cancer mechanism requires the identification of genes playing a role in the development of the pathology and the characterization of their role (notably oncogenes and tumor suppressors). We present an updated version of the R/bioconductor package called MoonlightR, namely Moonlight2R, which returns a list of candidate driver genes for specific cancer types on the basis of omics data integration. The Moonlight framework contains a primary layer where gene expression data and information about biological processes are integrated to predict genes called oncogenic mediators, divided into putative tumor suppressors and putative oncogenes. This is done through functional enrichment analyses, gene regulatory networks and upstream regulator analyses to score the importance of well-known biological processes with respect to the studied cancer type. By evaluating the effect of the oncogenic mediators on biological processes or through random forests, the primary layer predicts two putative roles for the oncogenic mediators: i) tumor suppressor genes (TSGs) and ii) oncogenes (OCGs). As gene expression data alone is not enough to explain the deregulation of the genes, a second layer of evidence is needed. We have automated the integration of a secondary mutational layer through new functionalities in Moonlight2R. These functionalities analyze mutations in the cancer cohort and classifies these into driver and passenger mutations using the driver mutation prediction tool, CScape-somatic. Those oncogenic mediators with at least one driver mutation are retained as the driver genes. As a consequence, this methodology does not only identify genes playing a dual role (e.g. TSG in one cancer type and OCG in another) but also helps in elucidating the biological processes underlying their specific roles. In particular, Moonlight2R can be used to discover OCGs and TSGs in the same cancer type. This may for instance help in answering the question whether some genes change role between early stages (I, II) and late stages (III, IV). In the future, this analysis could be useful to determine the causes of different resistances to chemotherapeutic treatments.

License GPL-3

biocViews DNAMethylation, DifferentialMethylation, GeneRegulation, GeneExpression, MethylationArray, DifferentialExpression, Pathways, Network, Survival, GeneSetEnrichment, NetworkEnrichment

Suggests BiocStyle, knitr, rmarkdown, testthat (>= 3.0.0), devtools, roxygen2, png

SystemRequirements CScapeSomatic

VignetteBuilder knitr

URL `https://github.com/ELELAB/Moonlight2R`

BugReports `https://github.com/ELELAB/Moonlight2R/issues`

RoxygenNote 7.2.3

LazyData false

Encoding UTF-8

Config/testthat/edition 3

git_url https://git.bioconductor.org/packages/Moonlight2R

git_branch RELEASE_3_18

git_last_commit f5bfcb9

git_last_commit_date 2023-10-24

Repository Bioconductor 3.18

Date/Publication 2024-03-22

Author Mona Nourbakhsh [aut],
Astrid Saksager [aut],
Nikola Tom [aut],
Xi Steven Chen [aut],
Antonio Colaprico [aut],
Catharina Olsen [aut],
Matteo Tiberti [cre, aut],
Elena Papaleo [aut]

Maintainer Matteo Tiberti <`tiberti@cancer.dk`>

R topics documented:

`confidence` *confidence*

Description

This function annotated a confidence level to the score

Usage

```
confidence(s, type)
```

Arguments

`s`	the score
`type`	coding or noncoding

Value

returns a confidence level or remark/error message

Examples

```
remark <- confidence(0.8, type='Coding')
```

`cscape_somatic_output` *Cscape-somatic annotations of TCGA-LUAD*

Description

Output from DMA. This contains the cscape-somatic annotations for all differentially expressed genes

Usage

```
data(cscape_somatic_output)
```

Format

A 645x7 matrix.

Value

A 645x7 matrix.

`dataDMA` *Output example from the function Driver Mutation Analysis*

Description

The predicted driver genes, which have at least one driver mutation.

Usage

```
data(dataDMA)
```

Format

A list of two.

Value

A list of two, containing 0 tumor-suppressor and 1 oncogene.

`dataFEA` *Functional enrichment analysis*

Description

The output of the FEA function which does enrichment analysis

Usage

```
data(dataFEA)
```

Format

A dataframe of dimension 101x7

Details

The input to the FEA is the differentially expressed genes.

Value

A dataframe of dimension 101x7

`dataFilt` *Gene expression data from TCGA-LUAD*

Description

A matrix that provides processed gene expression data (obtained from RNA seq) from the TCGA-LUAD project

Usage

```
data(dataFilt)
```

Format

A 3000x20 matrix

Details

The matrix contains the genes in rows and samples in columns. The data has been downloaded and processed using TCGAbiolinks.

Value

A 3000x20 matrix

`dataGLS` *Literature search of driver genes*

Description

A tibble containing results of literature search where predicted driver genes stored in dataDMA were queried for their role as drivers in PubMed

Usage

```
data(dataGLS)
```

Format

A 13x8 tibble.

Details

The tibble contains PubMed IDs, doi, title, abstract, year of publication, keywords, and total number of publications for the genes.

Value

A 13x8 tibble.

`dataGRN` *Gene regulatory network*

Description

The output of the GRN function which finds connections between genes.

Usage

```
data(dataGRN)
```

Format

A list of 2 elements where the first element is a 23x613 matrix and the second element is a vector of length 23

Details

The input to the GRN is the differentially expressed genes and the gene expression data.

Value

A list of 2 elements where the first element is a 23x613 matrix and the second element is a vector of length 23

`dataGRN_no_noise` *Gene regulatory network*

Description

The output of the GRN function which finds connections between genes where the noise is set to 0 for testing reproducibility purposes.

Usage

```
data(dataGRN_no_noise)
```

Format

A list of 2 elements where the first element is a 23x613 matrix and the second element is a vector of length 23

Details

The input to the GRN is the differentially expressed genes and the gene expression data.

Value

A list of 2 elements where the first element is a 23x613 matrix and the second element is a vector of length 23

`dataMAF` *Mutation data from TCGA LUAD*

Description

An examplary MAF file from TCGA on lung cancer LUAD. It contains 500 randomly selected mutations.

Usage

```
data(dataMAF)
```

Format

A 500x141 matrix.

Value

A 500x141 matrix.

`dataPRA` *Output example from function Pattern Recognition Analysis*

Description

The predicted TSGs and OCGs and their moonlight gene z-score based on the small sample TCGA-LUAD data. The PRA() were run with expert-based approach with apoptosis and proliferation of cells.

Usage

```
data(dataPRA)
```

Format

A list of two.

Value

A list of two.

`dataURA` *Upstream regulator analysis*

Description

The output of the URA function which carries out the upstream regulator analysis

Usage

```
data(dataURA)
```

Format

A 23x2 matrix

Details

The input to URA is the output of GRN and a list of biological processes and the differentially expressed genes

Value

A 23x2 matrix

`dataURA_plot` *Upstream regulator analysis*

Description

The output of the URA function which carries out the upstream regulator analysis

Usage

```
data(dataURA_plot)
```

Format

A 12x2 matrix

Details

This URA data is used to showcase some of the visualization functions

Value

A 12x2 matrix

DEGsmatrix *Differentially expressed genes*

Description

A matrix containing differentially expressed genes between lung cancer and normal samples found using TCGA-LUAD data and TCGAbiolinks.

Usage

```
data(DEGsmatrix)
```

Format

A 3390x5 matrix

Details

The matrix contains the differentially expressed genes in rows and log2 fold change and FDR values in columns.

Value

A 3390x5 matrix

DEG_Mutations_Annotations *Differentially expressed genes's Mutations*

Description

Output from DMA. This contains the differentially expressed genes's mutations and all annotations generated in DMA() on the TCGA-LUAD project.

Usage

```
data(DEG_Mutations_Annotations)
```

Format

A 3561x173 matrix.

Value

A 3561x173 matrix.

DiseaseList *Cancer-related biological processes*

Description

A dataset containing information about 101 cancer-related biological processes.

Usage

```
data(DiseaseList)
```

Format

A list of 101 elements

Details

The dataset contains a list of the 101 biological processes which includes genes playing a role in each biological processes including literature findings of the genes' function in the biological processes.

Value

A list of 101 elements

DMA *DMA*

Description

This function carries out the driver mutation analysis.

Usage

```
DMA(
  dataMAF,
  dataDEGs,
  dataPRA,
  runCscape = TRUE,
  coding_file,
  noncoding_file,
  results_folder = "./DMAresults"
)
```

Arguments

`dataMAF`	A MAF file rda object. The MAF file must at least contain the following columns: • Hugo_Symbol eg. BRCA1 • Chromosome eg. chr1 • Start_Position eg. 54402 • End_Position e.g. 54443 • Strand eg. + • Variant_Classification • Variant_Type • Reference_Allele • Tumor_Seq_Allele1 • Tumor_Seq_Allele2
`dataDEGs`	Output DEA function.
`dataPRA`	Output PRA function.
`runCscape`	Bolean. If `FALSE` will load CScape output file from results-folder Default = `TRUE`.
`coding_file`	A character string. Path to and name of CScape-somatic coding file. Can be downloaded at `http://cscape-somatic.biocompute.org.uk/#download`. The .tbi file must be placed in the same folder.
`noncoding_file`	A charcter string. Path to and name of CScape-somatic noncoding file. Can be downloaded at `http://cscape-somatic.biocompute.org.uk/#download`. The .tbi file must be placed in the same folder.
`results_folder`	A character string. Path to the results generated by this function.

Details

For more information about the different annotations added to the mutations please see the documentation as followes: `data(NCG)`, `data(EncodePromoters)`, `data(LOC_protein)` `data(LOC_transcription)` and `data(LOC_translation)`.

Value

List of two, containing TSGs and OCGs with at least one driver mutation. Additionally files are saved in `results_folder`. All output files are in compressed .rda format.

DEG_mutations_annotations.rda All differentially expressed genes' mutations and their annotations. These annotations include e.g. Cscape-somatic assessment, Level of Consequence, overlab with promoter sites and information from Network of Cancer Genes (NCG 7.0). All information from MAF and DEA is contained.

Oncogenic_mediators_annotation_summary.rda All oncogenic mediators and an summarisation of their mutation based on CScape-somatic assessment, Level of Consequences and total number of mutations. If a gene as previously been assessed as a driver in Network of Cancer Genes (7.0), it is annotated in a separate column.

Cscape_somatic_output.rda The file contain the cscape-somatic assessment for every mutation found in the differentially expressed genes. It is formatted exactly as the output of cscape-somatic, as if it was run in the terminal, except it is saved as .rda instead of csv.

Examples

```
DMA(dataMAF = dataMAF,
    dataDEGs = DEGsmatrix,
    dataPRA = dataPRA,
    coding_file = "path/css_coding.vcf.gz",
    noncoding_file = "path/css_noncoding.vcf.gz",
    results_folder = "path/results")

#If the cscape-somatic file have already been created
cscape_somatic_output <- read.csv("./results/Cscape_somatic_output.csv")
save(cscape_somatic_output, file = "./results/Cscape_somatic_output.rda")

DMA(dataMAF = dataMAF,
    dataDEGs = DEGsmatrix,
    dataPRA = dataPRA,
    runCscape = FALSE,
    results_folder = "./results")
```

`EAGenes` *Information about genes*

Description

A matrix containing information about 20038 genes including their gene description, location and family

Usage

`data(EAGenes)`

Format

A 20038x5 matrix

Details

The matrix contains the genes in rows and description, location and family in columns.

Value

A 20038x5 matrix

`EncodePromoters` *Promoters*

Description

Experimentially verified promoter sites by J. Michael Cherry, Stanford. Downloaded from the ENCODE identifier ENCSR294YNI. It contains chromosome, start and end sites of promoters.

Usage

```
data(EncodePromoters)
```

Format

A tibble with no columnnames or rownames.

1. The first column is chromosome eg. chr1
2. The second column is start position eg. 10451
3. The third column is end position eg. 10563

Value

A 84738x6 table

Source

https://www.encodeproject.org/

References

ENCODE identifier: ENCSR294YNI

Luo Y, Hitz BC, Gabdank I, Hilton JA, Kagda MS, Lam B, Myers Z, Sud P, Jou J, Lin K, Baymuradov UK, Graham K, Litton C, Miyasato SR, Strattan JS, Jolanki O, Lee JW, Tanaka FY, Adenekan P, O'Neill E, Cherry JM. New developments on the Encyclopedia of DNA Elements (ENCODE) data portal. Nucleic Acids Res. 2020 Jan 8;48(D1):D882-D889. doi: 10.1093/nar/gkz1062. PMID: 31713622; PMCID: PMC7061942.

FEA *FEA*

Description

This function carries out the functional enrichment analysis (FEA)

Usage

```
FEA(BPname = NULL, DEGsmatrix)
```

Arguments

BPname	BPname biological process such as "proliferation of cells", "ALL" (default) if FEA should be carried out for all 101 biological processes
DEGsmatrix	DEGsmatrix output from DEA such as dataDEGs

Value

matrix from FEA

Examples

```
data(DEGsmatrix)
data(DiseaseList)
data(EAGenes)
DEGsmatrix <- DEGsmatrix[seq.int(2), ]
dataFEA <- FEA(DEGsmatrix = DEGsmatrix, BPname = "apoptosis")
```

GEO_TCGAtab *Information on GEO and TCGA data*

Description

A matrix that provides the GEO dataset matched to one of 18 TCGA cancer types

Usage

```
data(GEO_TCGAtab)
```

Format

A 18x12 matrix

Details

The matrix contains the cancer types in rows and information about sample type from both TCGA and GEO in columns.

Value

A 18x12 matrix

`getDataGEO` *getDataGEO*

Description

This function retrieves and prepares GEO data

Usage

```
getDataGEO(GEOobject = "GSE39004", platform = "GPL6244", TCGAtumor = NULL)
```

Arguments

`GEOobject`	GEOobject
`platform`	platform
`TCGAtumor`	tumor name

Value

return GEO gset

Examples

```
data(GEO_TCGAtab)
dataGEO <-  getDataGEO(GEOobject = "GSE15641", platform = "GPL96")
```

GLS *GLS This function carries out gene literature search.*

Description

GLS This function carries out gene literature search.

Usage

```
GLS(genes, query_string = "AND cancer AND driver", max_records = 20)
```

Arguments

`genes`	A character string containing the genes to search in PubMed database
`query_string`	A character string containing words in query to follow the gene of interest. Default is "AND cancer AND driver" resulting in a final query of "Gene AND cancer AND driver". Standard PubMed syntax can be used in the query. For example Boolean operators AND, OR, NOT can be applied and tags such as [AU], [TITLE/ABSTRACT], [Affiliation] can be used.
`max_records`	An integer containing the maximum number of records to be fetched from PubMed.

Value

A tibble containing results of literature search where PubMed was queried for information of input genes. Each row in the tibble contains a PubMed ID matching the query, doi, title, abstract, year of publication, keywords, and total number of PubMed publications, resulting in a total of eight columns.

Examples

```
genes_query <- "TP53"
query <-
"AND cancer AND driver AND '1980/01/01'[Date - Publication] : '1980/01/01'[Date - Publication]"
dataGLS <- GLS(genes = genes_query,
               query_string = query)
```

GRN *Generate network*

Description

This function carries out the gene regulatory network inference using parmigene

Usage

```
GRN(
  TFs,
  DEGsmatrix,
  DiffGenes = FALSE,
  normCounts,
  kNearest = 3,
  nGenesPerm = 2000,
  nBoot = 400,
  noise_mi = 1e-12
)
```

Arguments

`TFs`	a vector of genes.
`DEGsmatrix`	DEGsmatrix output from DEA such as dataDEGs
`DiffGenes`	if TRUE consider only diff.expr genes in GRN
`normCounts`	is a matrix of gene expression with genes in rows and samples in columns.
`kNearest`	the number of nearest neighbors to consider to estimate the mutual information. Must be less than the number of columns of normCounts.
`nGenesPerm`	nGenesPerm
`nBoot`	nBoot
`noise_mi`	noise in knnmi.cross function. Default is 1e-12.

Value

an adjacent matrix

Examples

```
data('DEGsmatrix')
data('dataFilt')
dataGRN <- GRN(TFs = sample(rownames(DEGsmatrix), 30),
DEGsmatrix = DEGsmatrix,
DiffGenes = TRUE,
normCounts = dataFilt,
nGenesPerm = 2,
nBoot = 2)
```

`GSEA` *GSEA*

Description

This function carries out the GSEA enrichment analysis.

Usage

```
GSEA(DEGsmatrix, top, plot = FALSE)
```

Arguments

`DEGsmatrix`	DEGsmatrix output from DEA such as dataDEGs
`top`	is the number of top BP to plot
`plot`	if TRUE return a GSEA's plot

Value

return GSEA result

Examples

```
data("DEGsmatrix")
DEGsmatrix_example <- DEGsmatrix[1:2,]
dataFEA <- GSEA(DEGsmatrix = DEGsmatrix_example)
```

`knownDriverGenes` *Information of known cancer driver genes from COSMIC*

Description

A list of known cancer driver genes from COSMIC

Usage

```
data(knownDriverGenes)
```

Format

A list containing two elements where the first element is a character vector of 55 and the second element is a character vector of #' 84

Details

The list contains two elements: a vector of known tumor #' suppressors and a vector of known oncogenes

Value

A list containing two elements where the first element is a character vector of 55 and the second element is a character vector of #' 84

`LiftMAF` *LiftMAF*

Description

This function lifts a MAF file to a different genomic build.

Usage

```
LiftMAF(Infile, Current_Build)
```

Arguments

`Infile`	A tibble of MAF.
`Current_Build`	A charcter string, either `GRCh38` or `GRCh37`

Value

MAF tibble with positions lifted to another build

Examples

```
data(dataMAF)
dataMAF_example <- dataMAF[1,]
LiftMAF(dataMAF_example, Current_Build = 'GRCh38')
```

`listMoonlight` *List of oncogenic mediators of 5 TCGA cancer types*

Description

A list of oncogenic mediators of 5 TCGA cancer types: BLCA, BRCA, LUAD, READ and STAD

Usage

```
data(listMoonlight)
```

Format

A list containing 5 elements where each element contains differentially expressed genes and output from the URA and PRA functions of 5 TCGA cancer types

Details

Each element in the list contains differentially expressed genes and output from the URA and PRA functions

Value

A list containing 5 elements where each element contains differentially expressed genes and output from the URA and PRA functions of 5 TCGA cancer types

---

`LOC_protein` *Level of Consequence: Protein*

Description

A dataset binary dataset describing if a mutation of a certain class and type possibly have an effect on protein structure or function.

Usage

```
data(LOC_protein)
```

Format

A 18x7 table

Details

The values are binary: 0 no effect is possible, 1 an effect is possible.

See supplementary material for details.

Value

A 18x7 table

References

paper

`LOC_transcription` *Level of Consequence: Transcription*

Description

A dataset describing if a mutation of a certain class and type possibly have an effect on transcript level.

Usage

```
data(LOC_transcription)
```

Format

A 18x7 table

Details

The values are binary: 0 no effect is possible, 1 an effect is possible.

See supplementary material for details.

Value

A 18x7 table

References

paper

`LOC_translation` *Level of Consequence: Translation*

Description

A dataset describing if a mutation of a certain class and type possibly have an effect on peptide level.

Usage

```
data(LOC_translation)
```

Format

A 18x7 table

Details

The values are binary: 0 no effect is possible, 1 an effect is possible.

See supplementary material for details.

Value

A 18x7 table

References

paper

`LPA` *LPA*

Description

This function carries out the literature phenotype analysis (LPA)

Usage

```
LPA(dataDEGs, BP, BPlist)
```

Arguments

`dataDEGs`	is output from DEA
`BP`	is biological process
`BPlist`	is list of genes annotated in BP

Value

table with number of pubmed that affects, increase or decrase genes annotated in BP

Examples

```
data('DEGsmatrix')
data('DiseaseList')
BPselected <- c("apoptosis")
BPannotations <- DiseaseList[[match(BPselected, names(DiseaseList))]]$ID
```

`MAFtoCscape` *MAFtoCscape*

Description

This function extracts columns from a MAF tibble to fit CScape input format

Usage

```
MAFtoCscape(MAF)
```

Arguments

`MAF`	tibble of MAF

Value

tibble of cscape-somatic input

Examples

```
data(dataMAF)
MAFtoCscape(dataMAF[seq.int(2),])
```

`moonlight` *moonlight pipeline*

Description

moonlight is a tool for identification of cancer driver genes. This function wraps the different steps of the complete analysis workflow.

Usage

```
moonlight(
  dataDEGs,
  dataFilt,
  BPname = NULL,
  Genelist = NULL,
  kNearest = 3,
  nGenesPerm = 2000,
  DiffGenes = FALSE,
  nBoot = 400,
  nTF = NULL,
  thres.role = 0,
  dataMAF,
  path_cscape_coding,
  path_cscape_noncoding
)
```

Arguments

`dataDEGs`	table of differentially expressed genes
`dataFilt`	matrix of gene expression data with genes in rows and samples in columns
`BPname`	biological processes to use, if NULL: all processes will be used in analysis, RF for candidate; if not NULL the candidates for these processes will be determined (no learning)
`Genelist`	Genelist
`kNearest`	kNearest
`nGenesPerm`	nGenesPerm
`DiffGenes`	DiffGenes
`nBoot`	nBoot
`nTF`	nTF
`thres.role`	thres.role
`dataMAF`	A MAF file rda object for DMA
`path_cscape_coding`	A character string to path of CScape-somatic coding file
`path_cscape_noncoding`	A character string to path of CScape-somatic non-coding file

Value

table with cancer driver genes TSG and OCG.

Examples

```
drivers <- moonlight(dataDEGs = DEGsmatrix,
dataFilt = dataFilt,
BPname = c("apoptosis", "proliferation of cells"),
dataMAF = dataMAF,
path_cscape_coding = "css_coding.vcf.gz",
path_cscape_noncoding = "css_noncoding.vcf.gz")
```

NCG *Network of Cancer Genes 7.0*

Description

A dataset retrived from Network of Cancer Genes 7.0

Usage

```
data(NCG)
```

Format

The format have been rearranged from the original. <symbold>|<NCG_driver>|<NCG_cgc_annotation>|<NCG_vogelstein_a
<NCG_saito_annotation>|<NCG_pubmed_id>

Details

The NCG_driver is reported as a OCG or TSG when at least one of three three databases have documented it. These are cosmic gene census (cgc), vogelstein et al. 2013 or saito et al. 2020. The NCG_driver is reported as a candidate, when literature support the gene as a cancer driver.

Value

A 3347x7 table

Source

http://ncg.kcl.ac.uk/

References

Comparative assessment of genes driving cancer and somatic evolution in non-cancer tissues: an update of the Network of Cancer Genes (NCG) resource. Dressler L., Bortolomeazzi M., Keddar M.R., Misetic H., Sartini G., Acha-Sagredo A., Montorsi L., Wijewardhane N., Repana D., Nulsen J., Goldman J., Pollit M., Davis P., Strange A., Ambrose K. and Ciccarelli F.D.

`Oncogenic_mediators_mutation_summary` *Oncogenic Mediators Mutation Summary*

Description

Output from DMA. This contains the oncogenic mediator from the TCGA-LUAD project, and their mutation assesments summarised based on CSCape-somatic and Level of Consequence.

Usage

```
data(Oncogenic_mediators_mutation_summary)
```

Format

A 12x15 matrix.

Value

A 12x15 matrix.

`plotCircos` *plotCircos*

Description

This function visualize the plotCircos

Usage

```
plotCircos(
  listMoonlight,
  listMutation = NULL,
  additionalFilename = NULL,
  intensityColOCG = 0.5,
  intensityColTSG = 0.5,
  intensityColDual = 0.5,
  fontSize = 1
)
```

Arguments

`listMoonlight`	output Moonlight function
`listMutation`	listMutation
`additionalFilename`	additionalFilename
`intensityColOCG`	intensityColOCG
`intensityColTSG`	intensityColTSG
`intensityColDual`	intensityColDual
`fontSize`	fontSize

Value

no return value, plot is saved

Examples

```
data('listMoonlight')
plotCircos(listMoonlight = listMoonlight, additionalFilename = "_ncancer5")
```

`plotDMA` *plotDMA*

Description

This function creates one or more heatmaps on the output from DMA. It visualises the CScape-Somatic annotations per oncogenic mediator either in a single heatmap or split into several different ones. It is also possible to provide a personalised genelist to visualise.

Usage

```
plotDMA(
  DEG_Mutations_Annotations,
  Oncogenic_mediators_mutation_summary,
  type = "split",
  genelist = c(),
  additionalFilename = ""
)
```

Arguments

`DEG_Mutations_Annotations`
: A tibble, output file from DMA.

`Oncogenic_mediators_mutation_summary`
: A tibble, output file from DMA.

`type`
: A character string. It can take the values `"split"` or `"complete"`. If both type and genelist are `NULL`, the function will default to `"split"`.
  - `"split"` will split the entire dataset into sections of 40 genes and create individual plots. These plots will be merged into one pdf. The genes will be sorted alphabeatically.
  - `"complete"` will create one plot, though it will not be possible to see the individual gene names. The heatmap will be clustered hierarchically.

`genelist`
: A character vector containing HUGO symbols. A single heatmap will be created with only these genes. The heatmap will be hierarchically clustered. This will overwrite `type`.

`additionalFilename`
: A character string. Adds prefix or filepath to the filename of the pdf.

Value

No return value. DMA results are plotted.

Examples

```
data('DEG_Mutations_Annotations')
data('Oncogenic_mediators_mutation_summary')
plotDMA(DEG_Mutations_Annotations,
        Oncogenic_mediators_mutation_summary,
        genelist = c("ACSS2", "AFAP1L1"), additionalFilename = "myplots_")
```

`plotFEA` *plotFEA*

Description

This function visualize the functional enrichment analysis (FEA)’s barplot

Usage

```
plotFEA(
  dataFEA,
  topBP = 10,
  additionalFilename = NULL,
  height,
  width,
  offsetValue = 5,
  angle = 90,
  xleg = 35,
  yleg = 5,
  titleMain = "",
  minY = -5,
  maxY = 10,
  mycols = c("#8DD3C7", "#FFFFB3", "#BEBADA")
)
```

Arguments

`dataFEA`	dataFEA
`topBP`	topBP
`additionalFilename`	additionalFilename
`height`	Figure height
`width`	Figure width
`offsetValue`	offsetValue
`angle`	angle
`xleg`	xleg
`yleg`	yleg
`titleMain`	title of the plot
`minY`	minY
`maxY`	maxY
`mycols`	colors to use for the plot

Value

no return value, FEA result is plotted

Examples

```
data(DEGsmatrix)
data(DiseaseList)
data(EAGenes)
data(dataFEA)
plotFEA(dataFEA = dataFEA[1:10,], additionalFilename = "_example",height = 20,width = 10)
```

`plotHeatmap` *plotHeatmap*

Description

This function creates a unclustered heatmap from the inputted data tibble and saves it

Usage

```
plotHeatmap(df)
```

Arguments

`df`	a tibble

Value

The name of the alphabeatically first gene in the tibble

`plotMoonlight` *plotMoonlight*

Description

This function creates a heatmap of Moonlight gene z-scores for selected genes.

Usage

```
plotMoonlight(
  DEG_Mutations_Annotations,
  Oncogenic_mediators_mutation_summary,
  dataURA,
  gene_type = "drivers",
  n = 50,
  genelist = c(),
  BPlist = c(),
  additionalFilename = ""
)
```

Arguments

`DEG_Mutations_Annotations`	A tibble, output file from DMA.
`Oncogenic_mediators_mutation_summary`	A tibble, output file from DMA.
`dataURA`	Output URA function.
`gene_type`	A character string either `"mediators"` or `"drivers"`. • If `NULL` defaults to `"drivers"`. • `"mediators"` will show the oncogenic mediators with the highst number of mutations regardless of driver/passenger classification. • `"drivers"` will show the driver genes with the highest number of driver mutations.
`n`	Numeric. The top number of genes to be plotted. If `NULL` defaults to 50.
`genelist`	A vector of strings containing Hugo Symbols of genes. Overwrites `gene_type` argument.
`BPlist`	A vector of strings. Selection of the biological processes to visualise. If left empty defaults to every BP provided in the URA file.
`additionalFilename`	A character string. Adds prefix or filepath to the filename of the pdf.

Value

No return value. Moonlight scores are plotted for selected genes.

Examples

```
data(DEG_Mutations_Annotations)
data(Oncogenic_mediators_mutation_summary)
data(dataURA_plot)
plotMoonlight(DEG_Mutations_Annotations,
              Oncogenic_mediators_mutation_summary,
              dataURA_plot, genelist = c("AFAP1L1", "ABCG2"),
              additionalFilename = "myplot_")
```

`plotNetworkHive` *plotNetworkHive: Hive network plot*

Description

This function visualizes the GRN as a hive plot

Usage

```
plotNetworkHive(dataGRN, namesGenes, thres, additionalFilename = NULL)
```

Arguments

`dataGRN`	output GRN function
`namesGenes`	list TSG and OCG to define axes
`thres`	threshold of edges to be included
`additionalFilename`	additionalFilename

Value

no results Hive plot is executed

Examples

```
data(knownDriverGenes)
data(dataGRN)
plotNetworkHive(dataGRN = dataGRN, namesGenes = knownDriverGenes, thres = 0.55)
```

`plotURA` *plotURA: Upstream regulatory analysis heatmap plot*

Description

This function visualizes the URA in a heatmap

Usage

```
plotURA(dataURA, additionalFilename = "URAplot")
```

Arguments

`dataURA`	output URA function
`additionalFilename`	figure name

Value

heatmap

Examples

```
data(dataURA)
data(DiseaseList)
data(tabGrowBlock)
data(knownDriverGenes)
dataDual <- PRA(dataURA = dataURA,
BPname = c("apoptosis","proliferation of cells"),
thres.role = 0)
```

```
TSGs_genes <- names(dataDual$TSG)
OCGs_genes <- names(dataDual$OCG)
plotURA(dataURA = dataURA[c(TSGs_genes, OCGs_genes),],additionalFilename = "_example")
```

`PRA` *Pattern Recognition Analysis (PRA)*

Description

This function carries out the pattern recognition analysis

Usage

```
PRA(dataURA, BPname, thres.role = 0)
```

Arguments

`dataURA`	output URA function
`BPname`	BPname
`thres.role`	thres.role

Value

returns list of TSGs and OCGs when biological processes are provided, otherwise a randomForest based classifier that can be used on new data

Examples

```
data(dataURA)
data(DiseaseList)
data(tabGrowBlock)
data(knownDriverGenes)
dataPRA <- PRA(dataURA = dataURA[seq.int(2),],
BPname = c("apoptosis","proliferation of cells"),
thres.role = 0)
```

`PRAtoTibble` *PRAtoTibble*

Description

This function changes the PRA output to tibble format

Usage

```
PRAtoTibble(dataPRA)
```

Arguments

`dataPRA`	RDA object (list of two) from PRA

Value

tibble with drivers

Examples

```
data('dataPRA')
PRAtoTibble(dataPRA)
```

`RunCscape_somatic` *RunCscape_somatic*

Description

This function retrive cscape-scores to SNPs

Usage

```
RunCscape_somatic(input, coding_file, noncoding_file)
```

Arguments

`input`	Input matching cscape input
`coding_file`	cscape_table with coding scores
`noncoding_file`	cscape_table with noncoding scores

Value

returns a tibble with a score and remark for each SNP

Examples

```
cscape_out <- RunCscape_somatic(input, coding_file, noncoding_file)
```

`tabGrowBlock` *Information of growing/blocking characteristics of 101 biological processes*

Description

A matrix with biological processes in rows and the cancer #' growing or blocking effect of the process in columns

Usage

```
data(tabGrowBlock)
```

Format

A 101x3 matrix

Details

For each biological processes the cancer growing/blocking effect is indicated

Value

A 101x3 matrix

`tabix_func` *tabix_func*

Description

This function retrives the individial score for a SNP

Usage

```
tabix_func(Ranges, Reference_Allele, Mutant, file_coding, file_noncoding)
```

Arguments

`Ranges`	The position
`Reference_Allele`	The reference nucleotide
`Mutant`	The mutant nucleotide
`file_coding`	cscape_table with coding scores
`file_noncoding`	cscape_table with noncoding scores

Value

returns the score

Examples

```
data <- tabix_func(Ranges, Reference_Allele, Mutant, file_coding, file_noncoding)
```

`URA` *URA Upstream Regulator Analysis*

Description

This function carries out the upstream regulator analysis

Usage

```
URA(dataGRN, DEGsmatrix, BPname, nCores = 1)
```

Arguments

`dataGRN`	output GNR function
`DEGsmatrix`	output DPA function
`BPname`	biological processes
`nCores`	number of cores to use

Value

an adjacent matrix

Examples

```
data(DEGsmatrix)
dataDEGs <- DEGsmatrix
data(dataGRN)
data(DiseaseList)
data(EAGenes)
dataURA <- URA(dataGRN = dataGRN,
DEGsmatrix = dataDEGs,
BPname = c("apoptosis",
"proliferation of cells"))
```

Index