

# Package ‘Moonlight2R’

January 20, 2026

**Type** Package

**Title** Identify oncogenes and tumor suppressor genes from omics data

**Version** 1.8.1

**Depends** R (>= 4.5), 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, EpiMix, BiocGenerics, ggplot2, ExperimentHub, rlang, withr, data.table

**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. An additional mechanistic layer evaluates if there are mutations affecting the protein stability of the transcription factors (TFs) of the TSGs and OCGs, as that may have an effect on the expression of the genes.

**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** 7.3.3

**LazyData** false

**Encoding** UTF-8

**Config/testthat/edition** 3

**git\_url** <https://git.bioconductor.org/packages/Moonlight2R>

**git\_branch** RELEASE\_3\_22

**git\_last\_commit** 7b6c9f0

**git\_last\_commit\_date** 2025-12-09

**Repository** Bioconductor 3.22

**Date/Publication** 2026-01-19

**Author** Mona Nourbakhsh [aut],  
Astrid Saksager [aut],  
Nikola Tom [aut],  
Katrine Meldgård [aut],  
Anna Melidi [aut],  
Xi Steven Chen [aut],  
Antonio Colaprico [aut],  
Catharina Olsen [aut],  
Alessia Campo [aut],  
Matteo Tiberti [cre, aut],  
Elena Papaleo [aut]

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

## Contents

confidence . . . . .	4
cscap_somatic_output . . . . .	5
dataDMA . . . . .	5
dataFEA . . . . .	6
dataFilt . . . . .	6
dataGLS . . . . .	7
dataGMA . . . . .	7
dataGRN . . . . .	8
dataGRN_no_noise . . . . .	8
dataMAF . . . . .	9
dataMAVISp . . . . .	9
dataMethyl . . . . .	10
dataPRA . . . . .	10
dataURA . . . . .	11
dataURA_plot . . . . .	11
DEGsmatrix . . . . .	12
DEG_Methylation_Annotations . . . . .	12
DEG_Mutations_Annotations . . . . .	13
DiseaseList . . . . .	13
DMA . . . . .	14
EAGenes . . . . .	15
EncodePromoters . . . . .	16
EpiMix_getInfiniumAnnotation . . . . .	17
EpiMix_Results_Regular . . . . .	17
FEA . . . . .	18
GEO_TCGAtab . . . . .	18
getDataGEO . . . . .	19
GLS . . . . .	19
GMA . . . . .	20
GRN . . . . .	22
GSEA . . . . .	23
knownDriverGenes . . . . .	23
LiftMAF . . . . .	24
listMoonlight . . . . .	25
loadMAVISp . . . . .	25
LOC_protein . . . . .	26
LOC_transcription . . . . .	27
LOC_translation . . . . .	27
LPA . . . . .	28
LUAD_sample_anno . . . . .	29
MAFtoCscape . . . . .	29
MetEvidenceDriver . . . . .	30
moonlight . . . . .	30
NCG . . . . .	31
Oncogenic_mediators_methylation_summary . . . . .	32
Oncogenic_mediators_mutation_summary . . . . .	33
plotCircos . . . . .	33
plotDMA . . . . .	34
plotFEA . . . . .	35
plotGMA . . . . .	36

plotHeatmap . . . . .	37
plotMetExp . . . . .	38
plotMoonlight . . . . .	39
plotMoonlightMet . . . . .	40
plotNetworkHive . . . . .	41
plotURA . . . . .	42
PRA . . . . .	42
PRAtoTibble . . . . .	43
RunCscape_somatic . . . . .	43
tabGrowBlock . . . . .	44
tabix_func . . . . .	45
TFinfluence . . . . .	45
URA . . . . .	47
<b>Index</b>	<b>48</b>

---

confidence	<i>confidence</i>
------------	-------------------

---

**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')
```

---

<code>cscape_somatic_output</code>	<i>Cscape-somatic annotations of TCGA-LUAD</i>
------------------------------------	--

---

### **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.

---

<code>dataDMA</code>	<i>Output example from the function Driver Mutation Analysis</i>
----------------------	--

---

### **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	<i>Functional enrichment analysis</i>
---------	---------------------------------------

---

**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	<i>Gene expression data from TCGA-LUAD</i>
----------	--

---

**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.

---

dataGMA

*Output example from GMA function*

---

**Description**

The predicted driver genes based on methylation evidence

**Usage**

```
data(dataGMA)
```

**Format**

A list of length two

**Details**

The data is a list of two elements where each element represents predicted oncogenes and tumor suppressors

**Value**

A list of length two

---

dataGRN	<i>Gene regulatory network</i>
---------	--------------------------------

---

**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	<i>Gene regulatory network</i>
------------------	--------------------------------

---

**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 exemplary 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.

---

`dataMAVISp`*MAVISp dataset example*

---

**Description**

MAVISp results for proteins TP53 and RUNX1, simple mode. They are used for testing and vignettes. They are stored as a list of tibbles.

**Usage**

```
data(dataMAVISp)
```

**Format**

A list of data frames (183 x 43 and 1180 x 46).

**Details**

The dataframes contain the full available MAVISp data for simple mode at present time.

**Value**

A list of tibbles (183 x 43 and 1180 x 46).

---

`dataMethyl`*Methylation data matrix from TCGA-LUAD project*

---

**Description**

A data matrix containing methylation data from TCGA-LUAD where CpG probes are in rows and samples are in columns.

**Usage**

```
data(dataMethyl)
```

**Format**

A 73x27 matrix.

**Details**

The CpG probes are in rows and samples are in columns.

**Value**

A 73x27 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	<i>Upstream regulator analysis</i>
---------	------------------------------------

---

**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	<i>Upstream regulator analysis</i>
--------------	------------------------------------

---

**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	<i>Differentially expressed genes</i>
------------	---------------------------------------

---

**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_Methylation_Annotations	<i>Output example from GMA function</i>
-----------------------------	---

---

**Description**

Output file from running GMA function which is a summary of DEGs and associated CpG probes

**Usage**

```
data(DEG_Methylation_Annotations)
```

**Format**

A 3435x35 tibble

**Details**

The data is a table where each row is a CpG probe in a DEG. Various annotations such as start/end site of CpG probe, promoter/enhancer annotations, NCG annotations are included in the table.

**Value**

A 3435x35 tibble

---

`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,
  runCscope = 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: <ul style="list-style-type: none"> <li>• Hugo_Symbol eg. BRCA1</li> <li>• Chromosome eg. chr1</li> <li>• Start_Position eg. 54402</li> <li>• End_Position e.g. 54443</li> <li>• Strand eg. +</li> <li>• Variant_Classification</li> <li>• Variant_Type</li> <li>• Reference_Allele</li> <li>• Tumor_Seq_Allele1</li> <li>• Tumor_Seq_Allele2</li> </ul>
dataDEGs	Output DEA function.
dataPRA	Output PRA function.
runCscope	Boolean. If FALSE will load CScope output file from results-folder Default = TRUE.
coding_file	A character string. Path to and name of CScope-somatic coding file. Can be downloaded at <a href="http://cscape-somatic.biocompute.org.uk/#download">http://cscape-somatic.biocompute.org.uk/#download</a> . The .tbi file must be placed in the same folder.
noncoding_file	A character string. Path to and name of CScope-somatic noncoding file. Can be downloaded at <a href="http://cscape-somatic.biocompute.org.uk/#download">http://cscape-somatic.biocompute.org.uk/#download</a> . 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 follows: 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, overlap 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 CScap-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	<i>Promoters</i>
-----------------	------------------

---

**Description**

Experimentally 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.

---

**EpiMix\_getInfiniumAnnotation**

*EpiMix\_getInfiniumAnnotation* This function gets Infinium probe annotation from the sesameData library. This function is from the EpiMix package <https://bioconductor.org/packages/release/bioc/html/EpiMix.html>. Zheng Y, Jun J, Gevaert O (2023). EpiMix: EpiMix: an integrative tool for the population-level analysis of DNA methylation. R package version 1.1.2.

---

**Description**

EpiMix\_getInfiniumAnnotation This function gets Infinium probe annotation from the sesameData library. This function is from the EpiMix package <https://bioconductor.org/packages/release/bioc/html/EpiMix.html>. Zheng Y, Jun J, Gevaert O (2023). EpiMix: EpiMix: an integrative tool for the population-level analysis of DNA methylation. R package version 1.1.2.

**Usage**

```
EpiMix_getInfiniumAnnotation(plat = "EPIC", genome = "hg38")
```

**Arguments**

plat	A character string representing the methylation platform which can either be HM27, HM450 or EPIC
genome	A character string representing the genome build version which can either be hg19 or hg38

**Value**

Probe annotations

---

**EpiMix\_Results\_Regular**

*Output example from GMA function*

---

**Description**

The object, a list, that was returned from running the EpiMix function and is one of the outputs from the GMA function.

**Usage**

```
data(EpiMix_Results_Regular)
```

**Format**

A list of length nine

**Details**

The data is a list of nine elements which is outputted from the EpiMix function

**Value**

A list of length nine

---

FEA	<i>FEA</i>
-----	------------

---

**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	<i>Information on GEO and TCGA data</i>
-------------	---

---

**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	<i>getDataGEO</i>
------------	-------------------

---

## 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	<i>GLS This function carries out gene literature search.</i>
-----	--

---

## Description

GLS This function carries out gene literature search.

## Usage

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

**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.

**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, mesh\_terms, and total number of PubMed publications, resulting in a total of eight columns.

**Examples**

```
genes_query <- "ABCG2"
dataGLS <- GLS(genes = genes_query,
               query_string = "AND cancer AND driver AND
                             '2020/01/01'[Date - Publication] : '2025/01/01'[Date - Publication]")
```

GMA

*GMA This function carries out Gene Methylation Analysis***Description**

GMA This function carries out Gene Methylation Analysis

**Usage**

```
GMA(
  dataMET,
  dataEXP,
  dataPRA,
  dataDEGs,
  sample_info,
  met_platform = "HM450",
  prevalence_filter = NULL,
  output_dir = "./GMAresults",
  cores = 1,
  roadmap.epigenome.ids = NULL,
  roadmap.epigenome.groups = NULL
)
```

**Arguments**

dataMET	A data matrix containing the methylation data where the CpG probes are in the rows and samples are in the columns
dataEXP	A data matrix containing the gene expression data where the genes are in the rows and the samples are in the columns

<code>dataPRA</code>	A table containing the output of the PRA function
<code>dataDEGs</code>	A table containing the output of a DEA where gene names are rownames
<code>sample_info</code>	A table containing information on the samples. This table needs to contain two columns called <code>primary</code> and <code>sample.type</code> . The <code>primary</code> column contains sample names which should be the same as the column names in <code>dataMET</code> . The <code>sample.type</code> column indicates for each sample if it is a Cancer or Normal sample.
<code>met_platform</code>	A character string representing the microarray type that was used to collect the methylation data. This can either be HM27, HM450 or EPIC. Default is HM450.
<code>prevalence_filter</code>	A float or NULL representing if a prevalence filter should be applied or not. Default is NULL, meaning a prevalence filter will not be applied. If a float is specified, a prevalence filter will be applied where methylation states of probes will be altered depending on the threshold of prevalence supplied as <code>prevalence_filter</code> . For example, if <code>prevalence_filter = 20</code> , it means that if the prevalence of the hyper- or hypomethylated CpG probe exceeds 20, the methylation state will be unchanged but if the prevalence is lower than 20 the methylation state will be changed to NA, meaning no methylation state was detected. In case of dual methylated probes, the methylation state will stay dual if both the prevalence of hyper- and hypomethylations exceed 20, but if only one of the prevalences exceed 20 the dual state will be changed to the state exceeding 20. If none of the prevalences exceed 20, the dual state will be changed to NA.
<code>output_dir</code>	Path to where the results will be stored. If this directory does not exist, it will be created by the function. Default is <code>./GMAresults</code> .
<code>cores</code>	Number of cores to be used. Default is 1.
<code>roadmap.epigenome.ids</code>	A character string representing the epigenome ID that will be used to select enhancers. Since enhancers are tissue-specific, the tissue type needs to be specified in EpiMix. The enhancers are found from the RoadmapEpigenome project and the IDs can be found from Figure 2 in the publication with doi: 10.1038/nature14248. Default is NULL.
<code>roadmap.epigenome.groups</code>	A character string representing the epigenome group that will be used to select enhancers. Details are provided above with the <code>roadmap.epigenome.ids</code> parameter. Default is NULL.

## Value

List of two elements, containing predicted oncogenes and tumor suppressors. Additionally, various output files are saved in the specified output directory: `DEG_Methylation_Annotations.rda`, `Onco-genic_mediators_methylation_summary.rda`, `EpiMix_Results_Enhancer.rds`, `EpiMix_Results_Regular.rds`, `FunctionalPairs_Enhancer.csv`, `FunctionalPairs_Regular.csv`, `FunctionalProbes_Regular.rds`

## Examples

```
data("dataMethyl")
data("dataFilt")
data("dataPRA")
data("DEGsmatrix")
data("LUAD_sample_anno")
data("NCG")
data("EncodePromoters")
data("MetEvidenceDriver")
```

```

pattern <- "^(.{4}-{2}-{4}-{2}).*"
dataFilt_subset <- dataFilt[,1:10]
colnames(dataFilt_subset) <- sub(pattern, "\\1", colnames(dataFilt_subset))
dataGMA <- GMA(dataMET = dataMethyl, dataEXP = dataFilt_subset,
dataPRA = dataPRA, dataDEGs = DEGsmatrix,
sample_info = LUAD_sample_anno, met_platform = "HM450",
prevalence_filter = NULL,
output_dir = "../GMAResults", cores = 1, roadmap.epigenome.ids = "E096",
roadmap.epigenome.groups = NULL)

```

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	<i>GSEA</i>
------	-------------

---

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	<i>Information of known cancer driver genes from COSMIC</i>
------------------	---

---

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	<i>LiftMAF</i>
---------	----------------

---

**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 character 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	<i>List of oncogenic mediators of 5 TCGA cancer types</i>
---------------	---

---

**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

---

loadMAVISp	<i>loadMAVISp</i>
------------	-------------------

---

**Description**

This function loads the MAVISp database from the directory specified by the user.

**Usage**

```
loadMAVISp(
  mavisDB = NULL,
  proteins_of_interest = NULL,
  mode = "simple",
  ensemble = "md"
)
```

**Arguments**

mavisDB	path to the MAVISp database as a string. The database can be downloaded from an OSF repository ( <a href="https://osf.io/ufpzm/">https://osf.io/ufpzm/</a> ) and has the following structure.
proteins_of_interest	vector containing specific proteins of interest in HUGO format
mode	string determining whether to use simple or ensemble mode of mavis. Default is simple mode. Takes values: <ul style="list-style-type: none"> <li>• simple</li> <li>• ensemble</li> </ul>
ensemble	string specifying the ensemble to use. The available ensembles can be found in the MAVISp database. This is ignored for simple mode.

**Value**

returns a list of tibbles each containing the MAVISp entry of one protein. Each entry will contain an extra column specifying what data the stability classification is based on.

**Examples**

```
mavis_db_location <- system.file('extdata', 'mavis_db', package='Moonlight2R')

mavis_data <- loadMAVISp(mavisDB = mavis_db_location,
                        proteins_of_interest = c('TP53'),
                        mode = 'ensemble')
```

---

LOC_protein	<i>Level of Consequence: Protein</i>
-------------	--------------------------------------

---

**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	<i>Level of Consequence: Transcription</i>
-------------------	--

---

**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	<i>Level of Consequence: Translation</i>
-----------------	--

---

**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	<i>LPA</i>
-----	------------

---

**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 decrease genes annotated in BP

**Examples**

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

---

LUAD_sample_anno	<i>Sample annotations of TCGA-LUAD project</i>
------------------	--

---

**Description**

A matrix that annotates LUAD samples as either cancer or normal

**Usage**

```
data(LUAD_sample_anno)
```

**Format**

A 23x2 matrix

**Details**

The matrix contains two columns: "primary" which contains patient barcodes of TCGA-LUAD and "sample.type" which denotes if the sample is either a "Cancer" or "Normal" sample

**Value**

A 23x2 matrix

---

MAFtoCscape	<i>MAFtoCscape</i>
-------------	--------------------

---

**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),])
```

---

MetEvidenceDriver	<i>Methylation evidence table to define driver genes</i>
-------------------	--

---

**Description**

A tibble containing combinations of methylation states of probes used to define driver genes

**Usage**

```
data(MetEvidenceDriver)
```

**Format**

A 30x6 tibble.

**Details**

The tibble contains a value of 1 if a probe is found that is either hypo-, hyper-, dualmethylated or not methylated. This is compared with Moonlight's predictions of role of oncogenic mediators to define driver genes based on methylation evidence.

**Value**

A 30x6 tibble.

---

moonlight	<i>moonlight pipeline</i>
-----------	---------------------------

---

**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	<i>Network of Cancer Genes 7.0</i>
-----	------------------------------------

---

Description

A dataset retrived from Network of Cancer Genes 7.0

Usage

```
data(NCG)
```

Format

The format have been rearranged from the original. <symbol>|<NCG\_driver>|<NCG\_cgc\_annotation>|<NCG\_vogelstein\_annotation>|<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\_methylation\_summary

*Output example from GMA function*

---

**Description**

Output file from running the GMA function which is a summary of the oncogenic mediators and their sum of methylated CpG probes together with the evidence level of their role as driver gene.

**Usage**

```
data(Oncogenic_mediators_methylation_summary)
```

**Format**

A 25x7 tibble

**Details**

The data is a table where each row is an oncogenic mediator and the columns represent the predicted driver role and the sum of hypo-, hyper-, and dualmethylated CpG sites.

**Value**

A 25x7 tibble

---

Oncogenic_mediators_mutation_summary
<i>Oncogenic Mediators Mutation Summary</i>

---

**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	<i>plotCircos</i>
------------	-------------------

---

**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
intensityColDual
fontSize
```

Value

no return value, plot is saved

Examples

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

---

plotDMA	<i>plotDMA</i>
---------	----------------

---

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". <ul style="list-style-type: none"><li>"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.</li><li>"complete" will create one plot, though it will not be possible to see the individual gene names. The heatmap will be clustered hierarchically.</li></ul>
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	<i>plotFEA</i>
---------	----------------

---

**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", "#FFFB3", "#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)
```

---

plotGMA	<i>plotGMA This function plots results of the Gene Methylation Analysis. It visualizes the number of hypo/hyper/dual methylated CpG sites in oncogenic mediators or in a user-supplied gene list. The results are visualized either in a single heatmap or split into different ones which is specified in the function's three modes: split, complete and genelist.</i>
---------	--

---

**Description**

plotGMA This function plots results of the Gene Methylation Analysis. It visualizes the number of hypo/hyper/dual methylated CpG sites in oncogenic mediators or in a user-supplied gene list. The results are visualized either in a single heatmap or split into different ones which is specified in the function's three modes: split, complete and genelist.

**Usage**

```
plotGMA(
  DEG_Methylation_Annotations,
  Oncogenic_mediators_methylation_summary,
  type = "split",
  genelist = NULL,
  additionalFilename = ""
)
```

**Arguments**

DEG_Methylation_Annotations	A tibble which is outputted from the GMA function.
Oncogenic_mediators_methylation_summary	A tibble which is outputted from the GMA function.
type	A character string which can either be split, complete or genelist. If type is set to split, the entire dataset is split into groups of 40 genes and individual heatmaps of groups each containing 40 genes will be created and subsequently merged into one pdf where each page in the pdf is an individual heatmap. The genes will be sorted alphabetically. If type is set to complete, a single heatmap

is created where the number of differentially methylated CpG sites are shown for all oncogenic mediators. If type is set to genelist, a single heatmap will be created for genes supplied by the user in the genelist parameter. Default is split.

**genelist** A character string containing HUGO symbols of genes to be visualized in a single heatmap. Default is NULL.

**additionalFilename** A character string that can be used to add a prefix or filepath to the filename of the pdf visualizing the heatmap. Default is an empty string.

### Value

No value is returned. Visualizations in form of heatmaps are saved.

### Examples

```
data("DEG_Methylation_Annotations")
data("Oncogenic_mediators_methylation_summary")
genes <- c("ACAN", "ACE2", "ADAM19", "AFAP1L1")
plotGMA(DEG_Methylation_Annotations = DEG_Methylation_Annotations,
        Oncogenic_mediators_methylation_summary = Oncogenic_mediators_methylation_summary,
        type = "genelist", genelist = genes,
        additionalFilename = "./GMAresults/")
```

---

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

---

plotMetExp

*plotMetExp This function visualizes results of EpiMix.*


---

## Description

plotMetExp This function visualizes results of EpiMix.

## Usage

```
plotMetExp(
  EpiMix_results,
  probe_name,
  dataMET,
  dataEXP,
  gene_of_interest,
  additionalFilename = ""
)
```

## Arguments

**EpiMix\_results** The object, a list, that was returned from running the EpiMix function and is one of the outputs from the GMA function.

**probe\_name** A character string containing the name of the CpG probe that will be plotted.

**dataMET** A data matrix containing the methylation data where the CpG probes are in the rows and samples are in the columns

**dataEXP** A data matrix containing the gene expression data where the genes are in the rows and the samples are in the columns

**gene\_of\_interest** A character string containing the name of the gene that will be plotted.

**additionalFilename** A character string that can be used to add a prefix or filepath to the filename of the pdf visualizing the heatmap. Default is an empty string.

## Value

No value is returned. Visualizations are saved.

## Examples

```
data("EpiMix_Results_Regular")
data("dataMethyl")
data("dataFilt")
pattern <- "^(.{4}-.{2}-.{4}-.{2}).*"
colnames(dataFilt) <- sub(pattern, "\\1", colnames(dataFilt))
plotMetExp(EpiMix_results = EpiMix_Results_Regular,
  probe_name = "cg03035704",
  dataMET = dataMethyl,
  dataEXP = dataFilt,
  gene_of_interest = "ACVRL1",
  additionalFilename = "./GMAresults/")
```

---

plotMoonlight	<i>plotMoonlight</i>
---------------	----------------------

---

## 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". <ul style="list-style-type: none"> <li>• If NULL defaults to "drivers".</li> <li>• "mediators" will show the oncogenic mediators with the highest number of mutations regardless of driver/passenger classification.</li> <li>• "drivers" will show the driver genes with the highest number of driver mutations.</li> </ul>
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_")
```

---

plotMoonlightMet	<i>plotMoonlightMet</i> This function visualizes the effect of genes on biological processes and total number of hypo/hyper/dual methylated CpG sites in genes.
------------------	---

---

## Description

plotMoonlightMet This function visualizes the effect of genes on biological processes and total number of hypo/hyper/dual methylated CpG sites in genes.

## Usage

```
plotMoonlightMet(
  DEG_Methylation_Annotations,
  Oncogenic_mediators_methylation_summary,
  dataURA,
  genes,
  additionalFilename = ""
)
```

## Arguments

DEG_Methylation_Annotations	A tibble which is outputted from the GMA function.
Oncogenic_mediators_methylation_summary	A tibble which is outputted from the GMA function.
dataURA	Output of the URA function: a table containing the effect of oncogenic mediators on biological processes. This effect is quantified through Moonlight Gene Z-scores.
genes	A character string containing the genes to be visualized.
additionalFilename	A character string that can be used to add a prefix or filepath to the filename of the pdf visualizing the heatmap. Default is an empty string.

## Value

No value is returned. Visualization in form of a heatmap is saved.

**Examples**

```

data("DEG_Methylation_Annotations")
data("Oncogenic_mediators_methylation_summary")
data("dataURA_plot")
genes <- c("ACAN", "ACE2", "ADAM19", "AFAP1L1")
plotMoonlightMet(DEG_Methylation_Annotations = DEG_Methylation_Annotations,
                  Oncogenic_mediators_methylation_summary = Oncogenic_mediators_methylation_summary,
                  dataURA = dataURA_plot,
                  genes = genes,
                  additionalFilename = "./GMAresults/")

```

---

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	<i>plotURA: Upstream regulatory analysis heatmap plot</i>
---------	---

---

### Description

This function visualizes the URA in a heatmap

### Usage

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

### 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	<i>Pattern Recognition Analysis (PRA)</i>
-----	---

---

### 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	<i>PRAtoTibble</i>
-------------	--------------------

---

**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	<i>RunCscape_somatic</i>
-------------------	--------------------------

---

**Description**

This function retrieve 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	<i>Information of growing/blocking characteristics of 101 biological processes</i>
--------------	--

---

**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	<i>tabix_func</i>
------------	-------------------

---

**Description**

This function retrieves the individual 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	cscap_table with coding scores
file_noncoding	cscap_table with noncoding scores

**Value**

returns the score

**Examples**

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

---

TFinfluence	<i>TFinfluence</i>
-------------	--------------------

---

**Description**

This function finds mutations in the transcription factors (TF) of the DEGs that have a TF in the database.

**Usage**

```
TFinfluence(
  dataPRA,
  dataDEGs,
  dataTRRUST,
  dataMAF,
  dataMAVISp,
  stabClassMAVISp = "rasp"
)
```

**Arguments**

dataPRA	Output PRA function. List of TSG and OCG Must contain the following elements <ul style="list-style-type: none"> <li>• TSG (The TSGs identified by moonlight along with the moonlight score)</li> <li>• OCG (The OCGs identified by moonlight along with the moonlight score)</li> </ul>
dataDEGs	Output DEA function. Tibble containing differentially expressed genes. Must contain the following columns <ul style="list-style-type: none"> <li>• GENE (HUGO symbol of DEG)</li> <li>• logFC (The log fold change of DEG)</li> </ul>
dataTRRUST	Tibble containing gene-TF pairs and type of interaction Must contain the following columns: <ul style="list-style-type: none"> <li>• TF (HUGO symbol of TF)</li> <li>• Target (HUGO symbol of target gene of TF)</li> <li>• InteractionType (The effect the TF has on the target gene (Activation, Repression))</li> </ul>
dataMAF	Tibble containing mutation info from MAF Must at least contain the following columns: <ul style="list-style-type: none"> <li>• Hugo_Symbol eg. BRCA1</li> <li>• HGVS_Short eg. p.V83F</li> </ul>
dataMAVISp	Output loadMAVISp function. List of tibbles, one for each protein. The tibbles must contain at least <ul style="list-style-type: none"> <li>• (First column must contain the mutation (e.g. A54W). It is assumed that the column name is empty)</li> </ul> <p>Then the tibbles must contain columns matching either of the following names(or they will be excluded)</p> <ul style="list-style-type: none"> <li>• (Stability classification, [A-Za-z0-9]+, \\\(Rosetta, FoldX\\)) (Values: Stabilizing, Neutral, Destabilizing, Uncertain)</li> <li>• (Stability classification, [A-Za-z0-9]+, \\\(RaSP, FoldX\\)) (Values: Stabilizing, Neutral, Destabilizing, Uncertain)</li> </ul>
stabClassMAVISp	The protocol to use for mutation stability classification. Default is rasp. Accepts one of the following strings <ul style="list-style-type: none"> <li>• rosetta (uses the FoldX/Rosetta protocol)</li> <li>• rasp (uses the FoldX/RaSP protocol)</li> </ul>

**Value**

returns a tibble containing:

- GENE
- TF
- InteractionType
- PMID
- tf\_mutation
- stab\_class (the effect on stability as classified by MAVISp)

## Examples

```
data('dataPRA')
data('DEGsmatrix')
data('dataTRRUST')
data('dataMAF')
data('dataMAVISp')

TFinfluence(dataTRRUST = dataTRRUST,
            dataMAF = dataMAF,
            dataDEGs = DEGsmatrix,
            dataPRA = dataPRA,
            dataMAVISp = dataMAVISp,
            stabClassMAVISp = 'rasp')
```

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

## \* datasets

cscope\_somatic\_output, [5](#)  
dataDMA, [5](#)  
dataFEA, [6](#)  
dataFilt, [6](#)  
dataGLS, [7](#)  
dataGMA, [7](#)  
dataGRN, [8](#)  
dataGRN\_no\_noise, [8](#)  
dataMAF, [9](#)  
dataMAVISp, [9](#)  
dataMethyl, [10](#)  
dataPRA, [10](#)  
dataURA, [11](#)  
dataURA\_plot, [11](#)  
DEG\_Methylation\_Annotations, [12](#)  
DEG\_Mutations\_Annotations, [13](#)  
DEGsmatrix, [12](#)  
DiseaseList, [13](#)  
EAGenes, [15](#)  
EncodePromoters, [16](#)  
EpiMix\_Results\_Regular, [17](#)  
GEO\_TCGatab, [18](#)  
knownDriverGenes, [23](#)  
listMoonlight, [25](#)  
LOC\_protein, [26](#)  
LOC\_transcription, [27](#)  
LOC\_translation, [27](#)  
LUAD\_sample\_anno, [29](#)  
MetEvidenceDriver, [30](#)  
NCG, [31](#)  
Oncogenic\_mediators\_methylation\_summary, [32](#)  
Oncogenic\_mediators\_mutation\_summary, [33](#)  
tabGrowBlock, [44](#)

## \* internal

EpiMix\_getInfiniumAnnotation, [17](#)  
tabix\_func, [45](#)

confidence, [4](#)

cscope\_somatic\_output, [5](#)

dataDMA, [5](#)

dataFEA, [6](#)

dataFilt, [6](#)

dataGLS, [7](#)

dataGMA, [7](#)

dataGRN, [8](#)

dataGRN\_no\_noise, [8](#)

dataMAF, [9](#)

dataMAVISp, [9](#)

dataMethyl, [10](#)

dataPRA, [10](#)

dataURA, [11](#)

dataURA\_plot, [11](#)

DEG\_Methylation\_Annotations, [12](#)

DEG\_Mutations\_Annotations, [13](#)

DEGsmatrix, [12](#)

DiseaseList, [13](#)

DMA, [14](#)

EAGenes, [15](#)

EncodePromoters, [16](#)

EpiMix\_getInfiniumAnnotation, [17](#)

EpiMix\_Results\_Regular, [17](#)

FEA, [18](#)

GEO\_TCGatab, [18](#)

getDataGEO, [19](#)

GLS, [19](#)

GMA, [20](#)

GRN, [22](#)

GSEA, [23](#)

knownDriverGenes, [23](#)

LiftMAF, [24](#)

listMoonlight, [25](#)

loadMAVISp, [25](#)

LOC\_protein, [26](#)

LOC\_transcription, [27](#)

LOC\_translation, [27](#)

LPA, [28](#)

LUAD\_sample\_anno, [29](#)

MAFtoCscape, [29](#)

MetEvidenceDriver, [30](#)

moonlight, [30](#)

NCG, [31](#)

Oncogenic\_mediators\_methylation\_summary,  
[32](#)

Oncogenic\_mediators\_mutation\_summary,  
[33](#)

plotCircos, [33](#)

plotDMA, [34](#)

plotFEA, [35](#)

plotGMA, [36](#)

plotHeatmap, [37](#)

plotMetExp, [38](#)

plotMoonlight, [39](#)

plotMoonlightMet, [40](#)

plotNetworkHive, [41](#)

plotURA, [42](#)

PRA, [42](#)

PRAtiTibble, [43](#)

RunCscape\_somatic, [43](#)

tabGrowBlock, [44](#)

tabix\_func, [45](#)

TFinfluence, [45](#)

URA, [47](#)