

# Package ‘SurfR’

September 18, 2024

**Type** Package

**Title** Surface Protein Prediction and Identification

**Version** 1.1.0

**Description** Identify Surface Protein coding genes from a list of candidates.  
Systematically download data from GEO and TCGA or use your own data.  
Perform DGE on bulk RNAseq data.  
Perform Meta-analysis. Descriptive enrichment analysis and plots.

**License** GPL-3 + file LICENSE

**Encoding** UTF-8

**LazyData** false

**BugReports** <https://github.com/auroramaurizio/SurfR/issues>

**URL** <https://github.com/auroramaurizio/SurfR>

**biocViews** Software, Sequencing, RNASeq, GeneExpression, Transcription,  
DifferentialExpression, PrincipalComponent, GeneSetEnrichment,  
Pathways, BatchEffect, FunctionalGenomics, Visualization,  
DataImport, FunctionalPrediction, GenePrediction, GO

**VignetteBuilder** knitr

**RoxygenNote** 7.2.3

**Imports** httr, BiocFileCache, BiocStyle, SPsimSeq, DESeq2, edgeR,  
openxlsx, stringr, rhdf5, ggplot2, ggrepel, stats, magrittr,  
assertr, tidyr, dplyr, TCGAAbiolinks, biomaRt, metaRNASeq,  
enrichR, scales, venn, gridExtra, SummarizedExperiment, knitr,  
grDevices, graphics, utils

**Depends** R (>= 4.3.0)

**Suggests** testthat (>= 3.0.0)

**Config/testthat/edition** 3

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

**git\_branch** devel

**git\_last\_commit** b43b6d4

**git\_last\_commit\_date** 2024-04-30

**Repository** Bioconductor 3.20

**Date/Publication** 2024-09-17

**Author** Aurora Maurizio [aut, cre] (<<https://orcid.org/0000-0002-7194-4637>>),  
 Anna Sofia Tascini [aut, ctb] (<<https://orcid.org/0000-0001-5731-5490>>)

**Maintainer** Aurora Maurizio <auroraurizio1@gmail.com>

## Contents

.format_str . . . . .	2
Annotate_SPID . . . . .	3
combine_fisher_invnorm . . . . .	4
countData . . . . .	5
DGE . . . . .	6
DownloadArchS4 . . . . .	7
enrichedList . . . . .	8
Enrichment . . . . .	8
Enrichment_barplot . . . . .	9
enrichr_connect . . . . .	10
enrichr_download . . . . .	11
enrichr_urls . . . . .	11
Gene2SProtein . . . . .	12
GEOmetadata . . . . .	13
ind_deg . . . . .	14
metadata . . . . .	14
metaRNAseq . . . . .	15
plotPCA . . . . .	16
Splot . . . . .	17
SVenn . . . . .	18
TCGA_download . . . . .	19
<b>Index</b>	<b>21</b>

---

.format_str	<i>Format a string using placeholders - function from hypeR</i>
-------------	---

---

### Description

Format a string using placeholders - function from hypeR

### Usage

```
.format_str(string, ...)
```

### Arguments

string	A an unformatted string with placeholders
...	Variables to format placeholders with

### Value

A formatted string

## Examples

```
## Not run:  
format_str("Format with {1} and {2}", "x", "y")  
  
## End(Not run)
```

---

Annotate\_SPID

*Annotate\_SPID*

---

## Description

Annotate Surface Protein Coding genes according to EnrichR libraries

## Usage

```
Annotate_SPID(  
  DGE,  
  enrich.database = "WikiPathway_2021_Human",  
  output_tsv = FALSE  
)
```

## Arguments

DGE	Data.frame containing annotated DEG list, as the output of DGE or Gene2SProtein functions.
enrich.database	String containing the EnrichR databases you would like to consult. Default: WikiPathway_2021_Human.
output_tsv	Logical. If TRUE, outputs a tsv file with the results. By default, FALSE.

## Value

A dataframe with surface protein coding DEGs annotation.

## Warning

Be sure that enrich.database exists.

## See Also

[DGE](#) function for DGE, and [Gene2SProtein](#) function for Gene2SProtein analysis

Other functional-annotation functions: [Enrichment\\_barplot\(\)](#), [Enrichment\(\)](#)

**Examples**

```
# Deseq2 output sample
DGE = data.frame(GeneID = c("DLK1", "TOP2A"),
                 Mean_CPM_T = c(5.92, 9.91),
                 Mean_CPM_C = c(0.04, 0.03),
                 log2FoldChange = c(10.22, 8.42),
                 lfcSE = c(0.80, 0.48),
                 stat = c(12.68, 17.69),
                 pvalue = c(7.30135e-37, 4.37011e-70),
                 padj = c(1.49936e-35, 1.12976e-67))

library(enrichR)
annotated_DGE = Annotate_SPID(DGE, "WikiPathway_2021_Human")

# Output of Gene2SProtein function
GeneNames = c("CIITA", "EPCAM", "DLK1", "CD24")
SurfaceProteins_df = Gene2SProtein(GeneNames, input_type = "gene_name")
annotated_SP = Annotate_SPID(SurfaceProteins_df, "GO_Biological_Process_2021")
```

---

```
combine_fisher_invnorm
```

```
combine_fisher_invnorm
```

---

**Description**

Combine Meta-Analysis results with individual DE tables

**Usage**

```
combine_fisher_invnorm(
  ind_deg,
  invnorm,
  fishercomb,
  adjpval = 0.05,
  output_tsv = TRUE,
  output_filename = "combine_fisher_invnorm.tsv"
)
```

**Arguments**

<code>ind_deg</code>	List of independent DEG dataframes with p-values to be combined.
<code>invnorm</code>	inverse normal p-value combination technique dataframe (output of metaRNAseq)
<code>fishercomb</code>	Fisher p-value combination technique dataframe (output of metaRNAseq)
<code>adjpval</code>	threshold to represent as binary the Meta-Analysis output adjpval.
<code>output_tsv</code>	logical. If TRUE, it outputs table with results. Default: TRUE
<code>output_filename</code>	File name for the results file.

**Value**

A dataframe with DEindices and DName of DEG at the chosen Benjamini Hochberg threshold, and TestStatistic, rawpval, adjpval, binaryadjpval vectors for differential expression in the meta-analysis.

**See Also**

DGE function for DGE analysis, and <https://cran.r-project.org/web/packages/metaRNASeq/vignettes/metaRNASeq.pdf> for metaRNASeq package info

Other meta-analysis functions: [metaRNAseq\(\)](#)

**Examples**

```
# Deseq2 output samples
DGE1 <- data.frame(GeneID = c("DLK1", "EPCAM"),
                  Mean_CPM_T = c(5.92, 9.91),
                  Mean_CPM_C = c(0.04, 0.03),
                  log2FoldChange = c(10.22, 8.42),
                  lfcSE = c(0.80, 0.48),
                  stat = c(12.68, 17.69),
                  pvalue = c(7.30135e-37, 4.37011e-70),
                  padj = c(1.49936e-35, 1.12976e-67),
                  row.names = c("DLK1", "EPCAM"))
DGE2 <- data.frame(GeneID = c("DLK1", "EPCAM"),
                  Mean_CPM_T = c(3.92, 8.91),
                  Mean_CPM_C = c(0.04, 0.03),
                  log2FoldChange = c(7.22, 5.81),
                  lfcSE = c(0.80, 0.48),
                  stat = c(12.68, 17.69),
                  pvalue = c(7.30135e-37, 4.37011e-70),
                  padj = c(1.49936e-35, 1.12976e-67),
                  row.names = c("DLK1", "EPCAM"))

# input list
ind_deg <- list(DEG1_df = DGE1, DEG2_df = DGE2)
# perform invnorm meta-analysis
invnorm <- metaRNAseq(ind_deg, test_statistic = "invnorm", BHth = 0.05, nrep = c(2,2))
# perform fishercomb meta-analysis
fishercomb <- metaRNAseq(ind_deg, test_statistic = "fishercomb", BHth = 0.05)
# combine results
comb_pval_df <- combine_fisher_invnorm(ind_deg,
                                     invnorm, fishercomb,
                                     adjpval = 0.05,
                                     output_tsv = FALSE)
```

---

countData

*countData*

---

**Description**

Simulated raw counts to use as input for DGE and plotPCA functions. metadata is available.

**Usage**

```
data(countData)
```

**Format**

dataframe

**Details**

A dataframe with 2500 rows and 4 columns (sample names).

**Value**

A dataframe.

---

DGE

*DGE function*

---

**Description**

Perform Differential Gene Expression Analysis of RNA-Seq Data

**Usage**

```
DGE(
  expression,
  metadata,
  Nreplica,
  design = "~condition",
  condition = "condition",
  TEST,
  CTRL,
  alpha = 0.05,
  FC_filt = 0,
  output_tsv = FALSE,
  output_filename = "DEGs.tsv"
)
```

**Arguments**

expression	Dataframe with counts
metadata	Dataframe with sample metadata
Nreplica	Double. Minimum number of replicates in each group
design	Design formula for DGE
condition	Column of the metadata to use for DGE results
TEST	Character. sample name in metadata
CTRL	Character. sample name in metadata
alpha	Double. the significance cutoff used for optimizing the independent filtering (by default 0.1). If the adjusted p-value cutoff (FDR) will be a value other than 0.1, alpha should be set to that value.
FC_filt	Dataframe with counts
output_tsv	Logical. If TRUE, outputs a tsv file with the results. By default, FALSE.
output_filename	Name of the tsv output file. Default is DEGs.tsv.

**Value**

A dataframe with DEGs

**Examples**

```
# Simulation of bulk RNA data
countData <- matrix(floor(runif(10000, min=0, max=101)),ncol=4)
colnames(countData) <- paste("sample", seq_len(ncol(countData)), sep = "")
rownames(countData) <- paste("gene", seq_along(seq_len(10000/4)), sep = "")
metadata <- data.frame(samplesID = paste("sample", seq_len(ncol(countData)), sep = ""),
                      condition = factor(c("A","A","B","B")))
row.names(metadata) <- metadata$samplesID
# Perform DGE
DGEresults <- DGE(expression = countData, metadata = metadata,
                  Nreplica = 2,
                  design = "~condition",condition = "condition",
                  TEST = "A", CTRL = "B")
```

---

DownloadArchS4

*DownloadArchS4 function*

---

**Description**

Download count matrix from <https://maayanlab.cloud/archs4/>, given a vector of input GEO Sample accessions numbers (GSM).

**Usage**

```
DownloadArchS4(GSM, species, print_tsv = FALSE, filename = NULL)
```

**Arguments**

GSM	Vector with the GSM ids of the samples to consider.
species	Specify the specie of yuor GSM samples. Either human or mouse.
print_tsv	Logical. If TRUE, outputs a tsv file with the count matrix. By default, FALSE.
filename	Name of the tsv output file. Default is matrix.tsv.

**Value**

A count matrix with gene on the row and GSM ID on the column.

**Warning**

If the defined GSM ids do not have any match in ArchS4 database, we suggest to contact ArchS4 curator to add them.

**See Also**

[GEOmetadata](#) function for downloading GEO metadata. <https://www.ncbi.nlm.nih.gov/geo> for info on GSM. <https://maayanlab.cloud/archs4/> for info on ArchS4.

Other public-data functions: [GEOmetadata\(\)](#), [TCGA\\_download\(\)](#)

**Examples**

```
GSM <- c("GSM3447008", "GSM3447009")
GEO_count_matrix <- DownloadArchS4(GSM, species = "human",
                                   print_tsv = FALSE, filename = NULL)
```

---

enrichedList	<i>enrichedList</i>
--------------	---------------------

---

**Description**

Input list for `Enrichment_barplot` function. `enrichedList` is the output of `Enrichment` function applied to `ind_deg` object when `enrich.databases` is equal to `GO_Cellular_Component_2021`, default parameters.

**Usage**

```
data(enrichedList)
```

**Format**

list

**Details**

`enrichedList$fdr_up$GO_Cellular_Component_2021` contains upregulated gene enrichments, `enrichedList$fdr_down$GO_Cellular_Component_2021` contains downregulated gene enrichments.

**Value**

A list of lists.

---

Enrichment	<i>Enrichment function</i>
------------	----------------------------

---

**Description**

Perform enrichment Analysis of RNA-Seq Data

**Usage**

```
Enrichment(
  dfList,
  enrich.databases = c("GO_Biological_Process_2021", "GO_Cellular_Component_2021",
                      "GO_Molecular_Function_2021", "KEGG_2021_Human", "MSigDB_Hallmark_2020",
                      "WikiPathways_2016", "BioCarta_2016", "Jensen_TISSUES", "Jensen_COMPARTMENTS",
                      "Jensen_DISEASES"),
  p_adj = 0.05,
  logFC = 1,
  save.results = FALSE
)
```



**Arguments**

dfList            Dataframes list  
 enrich.databases            Vector of EnrichR databases to consult  
 p\_adj            Double. Adjusted pvalue threshold for the enrichment  
 logFC            Double. Fold change threshold for the enrichment  
 save.results    Logical. If TRUE saves input gene lists and enrichment results.

**Value**

A list of enrichment tables for upregulated and downregulated genes in the different enrichr databases

**See Also**

<https://maayanlab.cloud/Enrichr/> for additional information about enrichR.

Other functional-annotation functions: `Annotate_SPID()`, `Enrichment_barplot()`

**Examples**

```
df1 <- data.frame(GeneID = c("MEST", "CDK1", "PCLAF", "BIRC5"),
                  baseMean = c(13490.22, 10490.23, 8888.33, 750.33),
                  log2FoldChange = c(5.78, 6.76, -7.78, -8.78),
                  padj = c(2.28e-143, 2.18e-115, 2.18e-45, 0.006),
                  row.names = c("MEST", "CDK1", "PCLAF", "BIRC5"))
df2 <- data.frame(GeneID = c("MEST", "CDK1", "PCLAF", "BIRC5"),
                  baseMean = c(13490.22, 10490.23, 8888.33, 750.33),
                  log2FoldChange = c(5.78, 6.76, -7.78, -8.78),
                  padj = c(2.28e-143, 2.18e-115, 2.18e-45, 0.006),
                  row.names = c("MEST", "CDK1", "PCLAF", "BIRC5"))
dfList <- list(df1 = df1, df2 = df2)
test <- Enrichment(dfList, enrich.databases = c("GO_Cellular_Component_2021"),
                  save.results = FALSE)
```

---

Enrichment\_barplot    *Enrichment\_barplot*

---

**Description**

Barplot representing the top up-regulated or down-regulated significant pathways

**Usage**

```
Enrichment_barplot(
  Enrich,
  enrich.databases = c("GO_Biological_Process_2021", "GO_Cellular_Component_2021",
                      "GO_Molecular_Function_2021"),
  p_adj = 0.05,
  num_term = 10,
  cond = "UP",
  plot = FALSE
)
```

**Arguments**

Enrich	A list of enrichment tables for up and down-regulated genes in the different enrichR databases. Output of Enrichment.R function for one DGE experiment.
enrich.databases	Vector of EnrichR databases to consider. These databases must be present in the Enrich list.
p_adj	Double. Minimum Adjusted pvalue threshold for the enrichment
num_term	Double. Number of up-regulated and dw-regulated terms to represent
cond	String. Title of the plot.
plot	Logical. If TRUE save plot as pdf.

**Value**

bar plot of significant pathways.

**See Also**

Other functional-annotation functions: [Annotate\\_SPID\(\)](#), [Enrichment\(\)](#)

Other plot functions: [SVenn\(\)](#), [Splot\(\)](#), [plotPCA\(\)](#)

**Examples**

```
dbs <- c("GO_Biological_Process_2021")
dfList <- list()
if (requireNamespace("enrichR", quietly = TRUE)) {
  up_genes <- c("RUNX1", "DLK1", "TOP2A", "EPCAM", "GATA1", "KDR")
  dfList[["fdr_up"]] <- enrichR::enrichr(up_genes, dbs)
  dw_genes <- c("CD275", "COL1A1", "COL1A2", "LUM", "SOX9")
  dfList[["fdr_down"]] <- enrichR::enrichr(dw_genes, dbs)
  # Plot upregulated genes
  Enrichment_barplot(dfList,
                     enrich.databases = dbs,
                     p_adj = 0.01, num_term = 5, cond = "UP")
  # Plot downregulated genes
  #Enrichment_barplot(dfList,
  #                   enrich.databases = dbs,
  #                   p_adj = 0.01, num_term = 5, cond = "DOWN")
} else {
  print("example requires enrichR package")
}
```

---

enrichr\_connect

*Connect to the enrichr web application - function from hypeR*

---

**Description**

Connect to the enrichr web application - function from hypeR

**Usage**

```
enrichr_connect(endpoint, db = c("Enrichr"))
```

**Arguments**

endpoint	The url endpoint to connect to
db	A species

**Value**

A web response

---

enrichr_download	<i>Download data from enrichr in the form of a named list - function from hypeR</i>
------------------	---

---

**Description**

Download data from enrichr in the form of a named list - function from hypeR

**Usage**

```
enrichr_download(genesets, db = c("Enrichr"))
```

**Arguments**

genesets	A name corresponding to available genesets
db	A species

**Value**

A list of genesets

**Examples**

```
ATLAS <- enrichr_download("Human_Gene_Atlas")
```

---

enrichr_urls	<i>Get url base for species-specific enrichr libraries - function from hypeR</i>
--------------	--

---

**Description**

Get url base for species-specific enrichr libraries - function from hypeR

**Usage**

```
enrichr_urls(db = c("Enrichr"))
```

**Arguments**

db	A species
----	-----------

**Value**

A url

---

Gene2SProtein	<i>Gene2SProtein function</i>
---------------	-------------------------------

---

### Description

Detect Surface Proteins from a vector of genes. The surface proteins are identified according to the in silico human surfaceome database, available at <https://wlab.ethz.ch/surfaceome>.

### Usage

```
Gene2SProtein(  
  genes,  
  input_type = "gene_name",  
  output_tsv = FALSE,  
  output_filename = "surfaceProteins.tsv",  
  Surfpy_version = "log"  
)
```

### Arguments

genes	A vector of genes.
input_type	The gene identification type: gene_name, ensembl, entrez or uniProt_name. By default: gene_name.
output_tsv	Logical. If TRUE, outputs a tsv file with the results. By default, FALSE.
output_filename	Name of the tsv output file. Default is surfaceProteins.tsv.
Surfpy_version	The version of surfpy dataframe you wish to use. Choose between log or newest. By default use the most recent log version. If a log dataframe does not exist the newest is downloaded from <a href="https://wlab.ethz.ch/surfaceome">https://wlab.ethz.ch/surfaceome</a> .

### Value

A data frame with filtered surface proteins from the genes array. The dataframe contains also addition information obtained from surfpy.

### Warning

The surfpy database is interrogated using the gene identification type of your preference between gene\_name, ensembl, entrez or uniProt\_name. Note that you might loose some matches due to different gene version IDs.

### See Also

[DGE](#) for DGE analysis, <https://wlab.ethz.ch/surfaceome> for info on Surfpy

**Examples**

```
# from gene name IDs to Surface proteins
GeneNames <- c("CIITA", "EPCAM", "DLK1", "CD24", "CDCP1", "LYVE1", "ABCD1", "VAMP1")
SurfaceProteins_df <- Gene2SProtein(GeneNames, input_type = "gene_name")

# from ensembl IDs to Surface proteins
Ensembl <- c("ENSG00000178343", "ENSG00000176895", "ENSG00000162419", "ENSG00000170776",
            "ENSG00000092529", "ENSG00000135926", "ENSG00000152595", "ENSG00000121577",
            "ENSG00000186094", "ENSG00000126773", "ENSG00000198918", "ENSG00000167378",
            "ENSG00000095574", "ENSG00000140678", "ENSG00000262484", "ENSG00000133739",
            "ENSG00000172469", "ENSG00000112992", "ENSG00000148343", "ENSG00000138593")
SurfaceProteins_df <- Gene2SProtein(Ensembl, input_type = "ensembl",
                                   output_tsv = FALSE, Surfy_version = "new")
```

GEOmetadata

*GEOmetadata function***Description**

Download metadata from <https://www.ncbi.nlm.nih.gov/geo>, given an input GEO accession series.

**Usage**

```
GEOmetadata(GSE, GPL = "")
```

**Arguments**

GSE	The GSE series ID.
GPL	The GPL series numbers. Required only if the chosen GSE series ID include data from multiple sequencing platforms.

**Value**

A dataframe with all the available characteristics in GEO metadata genes array.

**Warning**

If the GEO accession series has more than 1 sequencing platforms you need to specify the GPL series numbers.

**See Also**

<https://www.ncbi.nlm.nih.gov/geo> for info on GEO repository

Other public-data functions: [DownloadArchS4\(\)](#), [TCGA\\_download\(\)](#)

**Examples**

```
# only one sequencing platform
mGSE133671 <- GEOmetadata(GSE = "GSE133671")
# multiple sequencing platforms
mGSE59483 <- GEOmetadata("GSE59483", GPL = c("GPL11154", "GPL15520"))
```

*ind\_deg**ind\_deg*

---

**Description**

Input list for metaRNAseq function made of 2 different small Deseq2 output samples dataframes for testing purposes: DEG1\_df and DEG2\_df.

**Usage**

```
data(ind_deg)
```

**Format**

dataframe list

**Details**

Each dataframe has 2 rows and 9 columns.

**Value**

A list of dataframes.

---

*metadata**metadata*

---

**Description**

Metadata associated with countData for testing purposes (functions DGE, plotPCA).

**Usage**

```
data(metadata)
```

**Format**

dataframe.

**Details**

A dataframe with 4 rows (sample names) and 3 columns (samplesID, condition A and B, therapy T1 and T2).

**Value**

A dataframe.

---

metaRNAseq	<i>metaRNAseq function</i>
------------	----------------------------

---

## Description

Perform Meta-Analysis of RNA-Seq Data

## Usage

```
metaRNAseq(  
  ind_deg,  
  test_statistic = "fishercomb",  
  BHth = 0.05,  
  adjpval.t = 0.05,  
  nrep = NULL,  
  plot = FALSE  
)
```

## Arguments

<code>ind_deg</code>	List of independent named DEG dataframes with p-values to be combined.
<code>test_statistic</code>	p-value combination technique (inverse normal or Fisher): <code>fishercomb</code> , <code>invnorm</code> . By default: <code>fishercomb</code> .
<code>BHth</code>	Benjamini Hochberg threshold.
<code>adjpval.t</code>	threshold to represent as binary the Meta-Analysis output <code>adjpval</code> .
<code>nrep</code>	Vector of numbers of replicates used in each study to calculate the previous one-sided p-values.
<code>plot</code>	Logical. If TRUE plot histogram of pvalues. By default, the False Discovery Rate is controlled at 0.05.

## Value

A list with DEindices of DEG at the chosen Benjamini Hochberg threshold, and `TestStatistic`, `rawpval`, `adjpval`, `binaryadjpval` vectors for differential expression in the meta-analysis.

## See Also

[DGE](#) for DGE analysis, and <https://cran.r-project.org/web/packages/metaRNASeq/vignettes/metaRNASeq.pdf> for metaRNASeq package info.

Other meta-analysis functions: `combine_fisher_invnorm()`

## Examples

```
# Deseq2 output samples  
DGE1 <- data.frame(GeneID = c("DLK1", "EPCAM"),  
  Mean_CPM_T = c(5.92, 9.91),  
  Mean_CPM_C = c(0.04, 0.03),  
  log2FoldChange = c(10.22, 8.42),  
  lfcSE = c(0.80, 0.48),  
  stat = c(12.68, 17.69),
```

```

        pvalue = c(7.30135e-37, 4.37011e-70),
        padj = c(1.49936e-35, 1.12976e-67),
        row.names = c("DLK1", "EPCAM"))
DGE2 <- data.frame(GeneID = c("DLK1", "EPCAM"),
                  Mean_CPM_T = c(3.92, 8.91),
                  Mean_CPM_C = c(0.04, 0.03),
                  log2FoldChange = c(7.22, 5.81),
                  lfcSE = c(0.80, 0.48),
                  stat = c(12.68, 17.69),
                  pvalue = c(7.30135e-37, 4.37011e-70),
                  padj = c(1.49936e-35, 1.12976e-67),
                  row.names = c("DLK1", "EPCAM"))

# input list
ind_deg <- list(DEG1_df = DGE1, DEG2_df = DGE2)
# perform meta-analysis
comb_pval_df <- metaRNAseq(ind_deg, test_statistic = "invnorm", BHth = 0.05, nrep = c(2,2))

```

---

plotPCA

*plotPCA function*


---

## Description

Plot PCA highlighting one or two data features

## Usage

```

plotPCA(
  matrix,
  metadata,
  nTOP = 500,
  dims = c(1, 2),
  centering = TRUE,
  scaling = TRUE,
  color.by = NULL,
  shape.by = NULL,
  pt.size = 6,
  cols.use = NULL,
  shape.use = NULL,
  main = "PCA",
  label = FALSE,
  new.label = NULL
)

```

## Arguments

matrix	Filtered count matrix in CPM or RPKM with gene on the row and sample ID on the column.
metadata	Sample metadata, row.names must be samples names.
nTOP	number of top genes to use for principal components, selected by highest row variance
dims	Dimensions to plot, must be a two-length numeric vector specifying x- and y-dimensions



centering	Logical. If TRUE center PCs
scaling	Logical. If TRUE scales PCs
color.by	Name of one or more metadata columns to color point by.
shape.by	Name of one or more metadata columns to shape point by. If NULL, all points are circles \(\default\).
pt.size	Size of the points in the plot.
cols.use	Vector of colors, each color corresponds to an identity class. By default, ggplot assigns colors.
shape.use	Vector of shape, each shape corresponds to an identity class.
main	Plot title. Default = PCA.
label	Logical. If TRUE adds samples label. Default = FALSE.
new.label	If NULL, use the sample names as in metadata row.names. Otherwise you can specify new labels.

### Value

PCA plot object created by ggplot2, which can be assigned and further customized.

### See Also

Other plot functions: [Enrichment\\_barplot\(\)](#), [SVenn\(\)](#), [Splot\(\)](#)

### Examples

```
# Simulation of bulk RNA data
countData <- matrix(floor(runif(10000, min=0, max=101)), ncol=4)
colnames(countData) <- paste("sample", seq_len(ncol(countData)), sep = "")
rownames(countData) <- paste("gene", seq_along(seq_len(10000/4)), sep = "")
metadata <- data.frame(samplesID = paste("sample", seq_len(ncol(countData)), sep = ""),
                      condition = factor(c("A", "A", "B", "B")),
                      therapy = factor(c("T1", "T2", "T1", "T2")))
row.names(metadata) <- metadata$samplesID
library(edgeR)
SurfR::plotPCA(matrix = cpm(countData),
               metadata = metadata,
               nTOP = 100,
               dims = c(1,2),
               color.by = "condition", shape.by = "therapy",
               label = FALSE, main = "PCA")
```

---

Splot

*Splot function*

---

### Description

Plot a barplot with features of Surface Protein

**Usage**

```
Splot(
  SurfaceProteins_df,
  group.by = "Membranome.Almen.main-class",
  cols.use = NULL,
  main = "Almen main class"
)
```

**Arguments**

SurfaceProteins_df	Output dataframe of Gene2SProtein function.
group.by	Name of columns to plot. Default = Membranome.Almen.main-class.
cols.use	Vector of colors, each color corresponds to an identity class. By default, ggplot assigns colors.
main	Plot title. Default = Almen main class.

**Value**

plot objec created by ggplot2, which can be assigned and further customized.

**See Also**

Other plot functions: [Enrichment\\_barplot\(\)](#), [SVenn\(\)](#), [plotPCA\(\)](#)

**Examples**

```
GeneNames <- c("CIITA", "EPCAM", "DLK1", "CD24", "CDCP1", "LYVE1", "ABCD1", "VAMP1")
SurfaceProteins_df <- Gene2SProtein(GeneNames, input_type = "gene_name")
Splot(SurfaceProteins_df)
```

---

SVenn

*SVenn*


---

**Description**

Venn diagram of common surface proteins overexpressed among up to 7 different studies

**Usage**

```
SVenn(
  S_list,
  cols.use = NULL,
  opacity = 0.5,
  output_intersectionFile = TRUE,
  filename = "intersection.xlsx"
)
```

**Arguments**

<code>S_list</code>	A list of a maximum of 7 surface protein sets detected in different studies.
<code>cols.use</code>	Vector of colors, each color corresponds to a study. By default, ggplot assigns colors.
<code>opacity</code>	Degree of opacity for the colors specified with <code>cols.use</code> (less opacity, more transparency).
<code>output_intersectionFile</code>	logical. If TRUE (default) write an xlsx output of protein in the intersections.
<code>filename</code>	Name of the output file with the intersections.

**Value**

venn plot of common genes.

**See Also**

[Gene2SProtein](#) for detection of Surface proteins from a list of genes.

Other plot functions: [Enrichment\\_barplot\(\)](#), [Splot\(\)](#), [plotPCA\(\)](#)

**Examples**

```
S_list <- list(SP1 <- c("EPCAM", "CD24", "DLK1", "CDCP1", "LYVE1"),
              SP2 <- c("DLK1", "EPCAM", "EGFR", "UPK1A", "UPK2"))
SP <- SVenn(S_list, cols.use = c("pink", "yellow"), output_intersectionFile = FALSE)
```

---

TCGA\_download

*TCGA\_download function*

---

**Description**

Downloads count matrix data from TCGA

**Usage**

```
TCGA_download(
  project,
  whichcounts = "unstranded",
  save.matrix = FALSE,
  save.metadata = FALSE,
  barcodes = NULL
)
```

**Arguments**

<code>project</code>	Character. A valid project from <code>TCGAbiolinks::getGDCprojects()</code> \$project_id
<code>whichcounts</code>	Character. Counts data to use. Choose from: <code>unstranded</code> , <code>stranded_first</code> , <code>stranded_second</code> . By default, <code>unstranded</code> .
<code>save.matrix</code>	Logical. If TRUE, outputs a tsv file with the Matrix. By default, FALSE.
<code>save.metadata</code>	Logical. If TRUE, outputs a tsv file with the metadata. By default, FALSE.
<code>barcodes</code>	Character. A vector with names of the barcodes you want to download. If NULL (default) it downloads all the available barcodes in the project.

**Value**

A list containing the Matrix and the metadata.

**See Also**

Other public-data functions: [DownloadArchS4\(\)](#), [GEOmetadata\(\)](#)

**Examples**

```
GBM_list_s1 <- TCGA_download(project="TCGA-GBM",
                             whichcounts = "unstranded",
                             save.matrix = FALSE, save.metadata = FALSE,
                             barcodes = c("TCGA-06-0878-01A-01R-1849-01"))
# remove downloaded data from TCGA
unlink('GDCdata', recursive = TRUE, force = TRUE)
file.remove("MANIFEST.txt")
```

# Index

## \* functional-annotation functions

Annotate\_SPID, 3  
Enrichment, 8  
Enrichment\_barplot, 9

## \* internal

.format\_str, 2  
enrichr\_connect, 10  
enrichr\_urls, 11

## \* meta-analysis functions

combine\_fisher\_invsnorm, 4  
metaRNAseq, 15

## \* plot functions

Enrichment\_barplot, 9  
plotPCA, 16  
Splot, 17  
SVenn, 18

## \* public-data functions

DownloadArchS4, 7  
GEOmetadata, 13  
TCGA\_download, 19

.format\_str, 2

Annotate\_SPID, 3, 9, 10

combine\_fisher\_invsnorm, 4, 15  
countData, 5

DGE, 3, 5, 6, 12, 15  
DownloadArchS4, 7, 13, 20

enrichedList, 8  
Enrichment, 3, 8, 10  
Enrichment\_barplot, 3, 9, 9, 17–19  
enrichr\_connect, 10  
enrichr\_download, 11  
enrichr\_urls, 11

Gene2SProtein, 3, 12, 19  
GEOmetadata, 7, 13, 20

ind\_deg, 14

metadata, 14  
metaRNAseq, 5, 15

plotPCA, 10, 16, 18, 19

Splot, 10, 17, 17, 19

SVenn, 10, 17, 18, 18

TCGA\_download, 7, 13, 19