

# Package ‘CRISPRball’

November 8, 2024

**Title** Shiny Application for Interactive CRISPR Screen Visualization,  
Exploration, Comparison, and Filtering

**Version** 1.2.0

**Description** A Shiny application for visualization, exploration, comparison,  
and filtering of CRISPR screens analyzed with MAGeCK RRA or MLE. Features  
include interactive plots with on-click labeling, full customization of plot aesthetics,  
data upload and/or download, and much more. Quickly and easily explore your CRISPR screen  
results and generate publication-quality figures in seconds.

**License** MIT + file LICENSE

**URL** <https://github.com/j-andrews7/CRISPRball>

**BugReports** <https://support.bioconductor.org/>

**Encoding** UTF-8

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 7.2.3

**biocViews** Software, ShinyApps, CRISPR, QualityControl, Visualization,  
GUI

**Imports** DT, shiny, grid, ComplexHeatmap, InteractiveComplexHeatmap,  
graphics, stats, ggplot2, plotly, shinyWidgets,  
shinycssloaders, shinyjqui, dittoSeq, matrixStats,  
colourpicker, shinyjs, MAGeCKFlute, circlize, PCAtools, utils,  
grDevices, htmlwidgets, methods

**Suggests** BiocStyle, msigdb, depmap, pool, RSQLite, mygene, testthat  
(>= 3.0.0), knitr, rmarkdown

**Depends** R (>= 4.4.0), shinyBS

**LazyData** false

**Config/testthat/edition** 3

**VignetteBuilder** knitr

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

**git\_branch** RELEASE\_3\_20

**git\_last\_commit** 83a6b84

**git\_last\_commit\_date** 2024-10-29

**Repository** Bioconductor 3.20

**Date/Publication** 2024-11-08

**Author** Jared Andrews [aut, cre] (<<https://orcid.org/0000-0002-0780-6248>>),  
Jacob Steele [ctb] (<<https://orcid.org/0000-0001-9924-2226>>)

**Maintainer** Jared Andrews <jared.andrews07@gmail.com>

## Contents

build_depmap_db . . . . .	2
CRISPRball . . . . .	3
depmap_22q1_cn . . . . .	6
depmap_22q1_crispr . . . . .	6
depmap_22q1_crispr_rnai . . . . .	7
depmap_22q1_rnai . . . . .	8
depmap_22q1_TPM . . . . .	9
gene_ingress . . . . .	9
get_depmap_essentiality . . . . .	10
get_depmap_plot_data . . . . .	11
plot_bar . . . . .	12
plot_correlation_heatmap . . . . .	13
plot_depmap_cn . . . . .	14
plot_depmap_dependency . . . . .	15
plot_depmap_expression . . . . .	16
plot_depmap_lineages . . . . .	17
plot_hist . . . . .	18
plot_lawn . . . . .	19
plot_pca_biplot . . . . .	22
plot_rank . . . . .	23
plot_volcano . . . . .	26
read_mle_gene_summary . . . . .	29
<b>Index</b>	<b>30</b>

---

build_depmap_db	<i>Build SQLite database of DepMap data</i>
-----------------	---

---

## Description

Build SQLite database of DepMap data

## Usage

```
build_depmap_db(
  retrieve = c("rnai", "crispr", "dependency", "cn", "ccle_tpm", "meta", "drug",
    "gene.summary", "release"),
  file = "depmap_db.sqlite"
)
```

**Arguments**

retrieve	Character vector of data to retrieve from DepMap. Options include: <ul style="list-style-type: none"> <li>• rnai: RNAi screen data from <a href="#">depmap_rnai</a>.</li> <li>• crispr: CRISPR screen data from <a href="#">depmap_crispr</a>.</li> <li>• cn: Copy number data from <a href="#">depmap_copyNumber</a>.</li> <li>• ccle_tpm: TPM data from <a href="#">depmap_TPM</a>.</li> <li>• meta: Cell line metadata from <a href="#">depmap_metadata</a>.</li> <li>• drug: Drug sensitivity data from <a href="#">depmap_drug_sensitivity</a>.</li> <li>• gene.summary: Gene summary data from <a href="#">depmap_gene_summary</a>.</li> <li>• release: DepMap release information from from <a href="#">depmap_release</a>.</li> </ul>
file	Name of SQLite database file to create.

**Value**

Name of SQLite database containing DepMap data.

**Author(s)**

Jared Andrews

**See Also**

[depmap\\_rnai](#), [depmap\\_crispr](#), [depmap\\_copyNumber](#), [depmap\\_TPM](#), [depmap\\_metadata](#), [depmap\\_gene\\_summary](#), [depmap\\_drug\\_sensitivity](#), [depmap\\_release](#) [dbPool](#), [dbWriteTable](#)

**Examples**

```
library(CRISPRball)
build_depmap_db(retrieve = "release")
```

---

CRISPRball	<i>Create an interactive Shiny app for visualization &amp; exploration of CRISPR analyses</i>
------------	---

---

**Description**

This package was designed to make the visualization, interpretation, and exploration of CRISPR screen analyses accessible for both analysts and bench scientists. It makes high-quality, publication-ready figures simple to create and customize. It also provides a simple interface to DepMap data for comparison and filtering.

**Usage**

```
CRISPRball(
  gene.data = NULL,
  sgrna.data = NULL,
  count.summary = NULL,
  norm.counts = NULL,
  h.id = "mag1",
  positive.ctrl.genes = NULL,
```

```

essential.genes = NULL,
depmap.db = NULL,
genesets = NULL,
return.app = TRUE
)

```

### Arguments

<code>gene.data</code>	A named list containing <code>gene_summary.txt</code> tables as data.frames. Multiple data.frames may be provided, one per element of the list. Users will be able to swap between them within the app. List element names should match names of <code>sgrna.data</code> list elements.
<code>sgrna.data</code>	A named list containing <code>sgrna_summary.txt</code> tables as data.frames. Multiple data.frames may be provided, one per element of the list. Users will be able to swap between them within the app. List element names should match names of <code>gene.data</code> list elements.
<code>count.summary</code>	Matrix or dataframe containing count summary ( <code>countsummary.txt</code> ) as generated by mageck count.
<code>norm.counts</code>	Matrix or dataframe containing normalized counts ( <code>count_normalized.txt</code> ) as generated by mageck count.
<code>h.id</code>	String indicating unique ID for interactive plots. Required if multiple apps are run within the same Rmd file.
<code>positive.ctrl.genes</code>	Optional character vector of gene identifiers for positive control genes from the screen so that they can be easily filtered.
<code>essential.genes</code>	Optional character vector of gene identifiers of common essential genes (i.e. pan-lethal) so that they can be easily filtered. If provided, overrides the <code>depmap.essential.genes</code> .
<code>depmap.db</code>	Optional character scalar for name of SQLite database returned by <code>build_depmap_db</code> .
<code>genesets</code>	Optional named list containing genesets that can be interactively highlighted on the plots. The elements of the list should each be a geneset with gene identifiers matching those used in the results.
<code>return.app</code>	Optional boolean indicating whether a Shiny app should be returned. TRUE by default. If FALSE, a named list of app elements (ui and server) will be returned instead. Useful for deploying as a standalone shiny app.

### Details

Features with no variation will be removed prior to `pca` being run for the PCA visualization. Gene labels can be added to the MAplot and volcano plot by clicking a point. The labels can also be dragged around, though adding labels will reset the positions, so it's recommended to add all labels prior to re-positioning them.

Most users will be interested in the main function to create a Shiny application (`CRISPRball`), though there are certain plotting functions that may be of interest to developers (`plot_volcano`, `plot_rank`, `plot_lawn`, etc). Most included plotting functions produce a ggplot object by default and have few required arguments. Many additional arguments are available for customization to generate complex, publication-ready figures.

This package supplements the `MAGeCKFlute` package, adding additional functionality, visualizations, and a Shiny interface.

To report bugs, suggest new features, or ask for help, the best method is to create an issue on the github, [here](#), or the bioconductor support site (be sure to tag 'CRISPRball' so that I get a notification!), [here](#)

## Value

A Shiny app containing interactive visualizations of CRISPR analysis results.

## Author(s)

Jared Andrews, Jacob Steele

## Examples

```
library(CRISPRball)
# Create app with no data loaded.
app <- CRISPRball()
if (interactive()) {
  shiny::runApp(app)
}

# Create app with data loaded.
# Create lists of results summaries for each dataset.
d1.genes <- read.delim(system.file("extdata", "esc1.gene_summary.txt",
  package = "CRISPRball"
), check.names = FALSE)
d2.genes <- read.delim(system.file("extdata", "plasmid.gene_summary.txt",
  package = "CRISPRball"
), check.names = FALSE)

d1.sgrnas <- read.delim(system.file("extdata", "esc1.sgrna_summary.txt",
  package = "CRISPRball"
), check.names = FALSE)
d2.sgrnas <- read.delim(system.file("extdata", "plasmid.sgrna_summary.txt",
  package = "CRISPRball"
), check.names = FALSE)

count.summ <- read.delim(system.file("extdata", "escneg.countsummary.txt",
  package = "CRISPRball"
), check.names = FALSE)
norm.counts <- read.delim(system.file("extdata", "escneg.count_normalized.txt",
  package = "CRISPRball"
), check.names = FALSE)

genes <- list(ESC = d1.genes, plasmid = d2.genes)
sgrnas <- list(ESC = d1.sgrnas, plasmid = d2.sgrnas)

app <- CRISPRball(
  gene.data = genes, sgrna.data = sgrnas,
  count.summary = count.summ, norm.counts = norm.counts
)
if (interactive()) {
  shiny::runApp(app)
}
```

---

depmap\_22q1\_cn      *DepMap copy number data*

---

**Description**

The DepMap copy number data for CDK2.

**Usage**

```
data(depmap_22q1_cn)
```

**Format**

depmap\_22q1\_TPM:

A dataframe with 1754 rows and 9 columns:

**depmap\_id** Depmap cell line ID

**log\_copy\_number** log2 copy number

**entrez\_id** Numeric entrez gene ID

**gene\_name** Gene symbol

**dataset** Screen type, either CRISPR or RNAi

**cell\_line\_name** Cell line name

**primary\_disease** Cell line disease origin

**lineage** Cell line lineage

**lineage\_subtype** Cell line lineage subtype, usually a malignancy

**hover.string** String for hover text when plotting

**Value**

A dataframe containing DepMap 22Q1 copy number data.

**Source**

<https://depmap.org/portal/download/all/>

<https://bioconductor.org/packages/release/data/experiment/html/depmap.html>

---

depmap\_22q1\_crispr      *DepMap CRISPR screen data*

---

**Description**

The DepMap CRISPR screen data for CDK2.

**Usage**

```
data(depmap_22q1_crispr)
```

**Format**

depmap\_22q1\_crispr:

A dataframe with 1070 rows and 9 columns:

**depmap\_id** Depmap cell line ID  
**dependency** Depmap dependency score  
**entrez\_id** Numeric entrez gene ID  
**gene\_name** Gene symbol  
**cell\_line\_name** Cell line name  
**primary\_disease** Cell line disease origin  
**lineage** Cell line lineage  
**lineage\_subtype** Cell line lineage subtype, usually a malignancy  
**hover.string** String for hover text when plotting

**Value**

A dataframe containing DepMap 22Q1 CRISPR screen data.

**Source**

<https://depmap.org/portal/download/all/>

<https://bioconductor.org/packages/release/data/experiment/html/depmap.html>

---

depmap\_22q1\_crispr\_rnai

*DepMap CRISPR & RNAi screen data*

---

**Description**

The DepMap CRISPR & RNAi screen data for CDK2.

**Usage**

```
data(depmap_22q1_crispr_rnai)
```

**Format**

depmap\_22q1\_crispr\_rnai:

A dataframe with 1782 rows and 9 columns:

**depmap\_id** Depmap cell line ID  
**dependency** Depmap dependency score  
**entrez\_id** Numeric entrez gene ID  
**gene\_name** Gene symbol  
**dataset** Screen type, either CRISPR or RNAi  
**cell\_line\_name** Cell line name  
**primary\_disease** Cell line disease origin  
**lineage** Cell line lineage  
**lineage\_subtype** Cell line lineage subtype, usually a malignancy  
**hover.string** String for hover text when plotting

**Value**

A dataframe containing DepMap 22Q1 CRISPR and RNAi data.

**Source**

<https://depmap.org/portal/download/all/>

<https://bioconductor.org/packages/release/data/experiment/html/depmap.html>

---

depmap_22q1_rnai	<i>DepMap RNAi screen data</i>
------------------	--------------------------------

---

**Description**

The DepMap RNAi screen data for CDK2.

**Usage**

```
data(depmap_22q1_rnai)
```

**Format**

depmap\_22q1\_rnai:

A dataframe with 712 rows and 9 columns:

**dependency** Depmap dependency score

**entrez\_id** Numeric entrez gene ID

**gene\_name** Gene symbol

**depmap\_id** Depmap cell line ID

**cell\_line\_name** Cell line name

**primary\_disease** Cell line disease origin

**lineage** Cell line lineage

**lineage\_subtype** Cell line lineage subtype, usually a malignancy

**hover.string** String for hover text when plotting

**Value**

A dataframe containing DepMap 22Q1 RNAi screen data.

**Source**

<https://depmap.org/portal/download/all/>

<https://bioconductor.org/packages/release/data/experiment/html/depmap.html>



---

depmap_22q1_TPM	<i>DepMap expression data</i>
-----------------	-------------------------------

---

### Description

The DepMap expression data for CDK2.

### Usage

```
data(depmap_22q1_TPM)
```

### Format

depmap\_22q1\_TPM:

A dataframe with 1393 rows and 9 columns:

**depmap\_id** Depmap cell line ID  
**rna\_expression** log2(TPM+1) expression  
**entrez\_id** Numeric entrez gene ID  
**gene\_name** Gene symbol  
**dataset** Screen type, either CRISPR or RNAi  
**cell\_line\_name** Cell line name  
**primary\_disease** Cell line disease origin  
**lineage** Cell line lineage  
**lineage\_subtype** Cell line lineage subtype, usually a malignancy  
**hover.string** String for hover text when plotting

### Value

A dataframe containing DepMap 22Q1 expression data.

### Source

<https://depmap.org/portal/download/all/>

<https://bioconductor.org/packages/release/data/experiment/html/depmap.html>

---

gene_ingress	<i>Parse gene summary data for easier plotting and display</i>
--------------	--

---

### Description

Parse gene summary data for easier plotting and display

**Usage**

```
gene_ingress(  
  df,  
  sig.thresh,  
  es.thresh,  
  es.col,  
  sig.col,  
  positive.ctrl.genes = NULL,  
  essential.genes = NULL,  
  depmap.genes = NULL  
)
```

**Arguments**

df	data.frame of gene summary data. Gene IDs should be in the first column.
sig.thresh	Numeric scalar for significance threshold to consider a gene a hit.
es.thresh	Numeric scalar for absolute log fold change threshold to consider a gene a hit.
es.col	Character scalar for the column name of the effect size value.
sig.col	Character scalar for the column name of the significance value.
positive.ctrl.genes	Character vector of gene identifiers to label as positive controls.
essential.genes	Character vector of gene identifiers to label as essential genes.
depmap.genes	data.frame of DepMap gene summary data.

**Value**

A data.frame of gene summary with additional, easier to plot, columns added.

**Author(s)**

Jared Andrews

**Examples**

```
library(CRISPRball)  
d1.genes <- read.delim(system.file("extdata", "esc1.gene_summary.txt",  
  package = "CRISPRball"  
) , check.names = FALSE)  
out.df <- gene_ingress(d1.genes, 0.05, 0.5, es.col = "LFC", sig.col = "fdr")
```

---

get\_depmap\_essentiality

*Get essential/selective gene information from DepMap summary table.*

---

**Description**

Get essential/selective gene information from DepMap summary table.

**Usage**

```
get_depmap_essentiality(gene, depmap.summary)
```

**Arguments**

gene                    Character scalar for gene symbol.  
 depmap.summary   data.frame containing DepMap gene summary information.

**Value**

Named list containing RNAi and CRISPR named lists containing dataset information for the provided gene, if available. If the gene is not found in the summary data.frame, the avail element for the RNAi and CRISPR lists will be set to FALSE.

**Author(s)**

Jared Andrews

**Examples**

```
library(CRISPRball)
build_depmap_db(retrieve = "gene.summary")
pool <- pool::dbPool(RSQLite::SQLite(), dbname = "depmap_db.sqlite")
depmap.gene <- pool::dbGetQuery(pool, "SELECT * FROM 'gene.summary'")

essentials <- get_depmap_essentiality(gene = "CDK2", depmap.summary = depmap.gene)
```

---

get\_depmap\_plot\_data    *Create DepMap dataframe for plotting*

---

**Description**

Based on the requested data type, this function will create a dataframe from the DepMap database that can be used for plotting.

**Usage**

```
get_depmap_plot_data(gene, data.type, depmap.meta, depmap.pool)
```

**Arguments**

gene                    Character scalar of gene name.  
 data.type              Character scalar of data type to retrieve. One of "dependency", "crispr", "rna",  
 "cn", or "ccl\_tpm".  
 depmap.meta            data.frame of DepMap cell line metadata, as stored in the 'meta' table of the  
 SQLite database built by [build\\_depmap\\_db](#).  
 depmap.pool            pool connection to DepMap SQLite database built with [build\\_depmap\\_db](#).

**Value**

data.frame containing appropriate DepMap data for plotting.

**Author(s)**

Jared Andrews

**See Also**[plot\\_depmap\\_lineages](#)**Examples**

```
## Not run:
library(CRISPRball)
build_depmap_db(retrieve = c("meta", "crispr"))
pool <- pool::dbPool(RSQLite::SQLite(), dbname = "depmap_db.sqlite")
depmap.meta <- pool::dbGetQuery(pool, "SELECT * FROM 'meta'")

df <- get_depmap_plot_data(
  gene = "CDK2", data.type = "crispr",
  depmap.meta = depmap.meta, depmap.pool = pool
)

## End(Not run)
```

---

`plot_bar`*Create interactive bar plot*

---

**Description**

Create an interactive bar plot for specific summary information for each sample in the dataset.

**Usage**

```
plot_bar(
  count.summary,
  x = "Label",
  y = "GiniIndex",
  title = "sgRNA Read Distribution",
  xlab = NULL,
  ylab = "Gini Index",
  fill = "#E69F00",
  yaxis.addition = 0.05
)
```

**Arguments**

<code>count.summary</code>	data.frame containing the count summary information for each sample.
<code>x</code>	Character scalar for column of count.summary to plot along the x-axis.
<code>y</code>	Character scalar for column of count.summary to plot along the y-axis.
<code>title</code>	Character scalar for title of the plot.
<code>xlab</code>	Character scalar for label of the x-axis.
<code>ylab</code>	Character scalar for label of the y-axis.
<code>fill</code>	Character scalar for bar fill color in hex.
<code>yaxis.addition</code>	Numeric scalar for additional space to add to the y-axis.

**Value**

An interactive bar chart showing the specified summary information for each sample. The axis and plot title are editable.

**Author(s)**

Jared Andrews

**See Also**

[BarView](#), for a static bar plot from the same count summary data.

**Examples**

```
library(CRISPRball)
count.summ <- read.delim(system.file("extdata", "escneg.countsummary.txt",
  package = "CRISPRball"
), check.names = FALSE)
# Gini Index plot
plot_bar(count.summ)

# Zero count sgRNAs plot
plot_bar(count.summ,
  x = "Label", y = "Zerocounts", title = "Fully Depleted sgRNAs",
  fill = "#394E80", ylab = "Zero Count sgRNAs", yaxis.addition = 10
)
```

---

plot\_correlation\_heatmap

*Plot a Correlation Heatmap*

---

**Description**

This function creates a heatmap using ComplexHeatmap to display the correlation values in a matrix. The color of each cell in the heatmap is determined by the corresponding correlation value, using a color ramp that ranges from the minimum value color to a maximum value color.

**Usage**

```
plot_correlation_heatmap(
  mat,
  min.color = "#FF0000",
  max.color = "#0000FF",
  legend.title = "Pearson Corr.",
  plot.title = "Correlation Matrix"
)
```

**Arguments**

mat	A matrix containing the correlation values.
min.color	Character scalar for the hexadecimal color code for the minimum values in the heatmap.
max.color	Character scalar for the hexadecimal color code for the maximum values in the heatmap.
legend.title	Character scalar for title of the legend.
plot.title	Character scalar for title of the plot.

**Value**

A Heatmap object.

**Author(s)**

Jared Andrews

**Examples**

```
library(CRISPRball)
norm.counts <- read.delim(system.file("extdata", "escneg.count_normalized.txt",
  package = "CRISPRball"
), check.names = FALSE)
norm.counts <- as.matrix(norm.counts[, c(-1, -2)])
norm.counts.log <- log2(norm.counts + 1)
cor.mat <- cor(norm.counts.log)
plot_correlation_heatmap(cor.mat)
```

---

plot\_depmap\_cn

*Plot gene copy number information from DepMap*

---

**Description**

Plot gene copy number information from DepMap

**Usage**

```
plot_depmap_cn(df, color = "#CEA3CB", plot.grid = FALSE)
```

**Arguments**

df	data.frame containing information for a single gene as returned by <a href="#">get_depmap_plot_data</a> .
color	Character scalar for trace color.
plot.grid	Boolean indicating whether to plot gridlines.

**Value**

plotly object

**Author(s)**

Jared Andrews

**See Also**[get\\_depmap\\_plot\\_data](#)**Examples**

```
library(CRISPRball)
data(depmap_22q1_cn)
plot_depmap_cn(depmap_22q1_cn)
```

---

`plot_depmap_dependency`

*Plot gene dependency information from DepMap CRISPR and RNAi tables*

---

**Description**

Plot gene dependency information from DepMap CRISPR and RNAi tables

**Usage**

```
plot_depmap_dependency(  
  df,  
  crispr.color = "#3584B5",  
  rnai.color = "#52288E",  
  depline = TRUE,  
  plot.grid = FALSE  
)
```

**Arguments**

<code>df</code>	data.frame containing information for a single gene as returned by <a href="#">get_depmap_plot_data</a> .
<code>crispr.color</code>	Character scalar for CRISPR trace color as hexcode.
<code>rnai.color</code>	Character scalar for RNAi trace color as hexcode.
<code>depline</code>	Boolean indicating whether to show the dependency threshold line.
<code>plot.grid</code>	Boolean indicating whether to plot gridlines.

**Value**

plotly object

**Author(s)**

Jared Andrews

**See Also**[get\\_depmap\\_plot\\_data](#)

**Examples**

```
library(CRISPRball)
data(depmap_22q1_crispr_rnai)
plot_depmap_dependency(depmap_22q1_crispr_rnai)
```

---

plot\_depmap\_expression

*Plot gene expression information from DepMap, mostly from CCLE*

---

**Description**

Plot gene expression information from DepMap, mostly from CCLE

**Usage**

```
plot_depmap_expression(df, color = "#7B8CB2", plot.grid = FALSE)
```

**Arguments**

df	data.frame containing information for a single gene as returned by <a href="#">get_depmap_plot_data</a> .
color	Character scalar for trace color.
plot.grid	Boolean indicating whether to plot gridlines.

**Value**

plotly object

**Author(s)**

Jared Andrews

**See Also**

[get\\_depmap\\_plot\\_data](#)

**Examples**

```
library(CRISPRball)
data(depmap_22q1_TPM)
plot_depmap_expression(depmap_22q1_TPM)
```



---

plot\_depmap\_lineages *Plot selected information across lineages from DepMap.*

---

### Description

Plot selected information across lineages from DepMap.

### Usage

```
plot_depmap_lineages(  
  df,  
  plot.val,  
  group.by,  
  lineage = NULL,  
  depline = TRUE,  
  label.size = 12,  
  pt.size = 5,  
  pt.color = "#56B4E9",  
  boxplot.fill = "#E2E2E2",  
  boxplot.line.color = "#000000"  
)
```

### Arguments

df	data.frame containing information for a single gene as returned by <a href="#">get_depmap_plot_data</a> .
plot.val	Character scalar of column name to plot values from.
group.by	Character scalar of column name to group by.
lineage	Character scalar of lineage for which to plot sub-lineage data.
depline	Boolean indicating whether to show the dependency threshold line.
label.size	Numeric scalar for axis label size.
pt.size	Numeric scalar for point size.
pt.color	Character scalar for point color.
boxplot.fill	Character scalar for boxplot fill color.
boxplot.line.color	Character scalar for boxplot line color.

### Value

plotly object

### Author(s)

Jared Andrews

### See Also

[get\\_depmap\\_plot\\_data](#)

**Examples**

```
library(CRISPRball)
data("depmap_22q1_rnai")
plot_depmap_lineages(df = depmap_22q1_rnai, plot.val = "dependency", group.by = "lineage")
```

---

plot\_hist

---

*Create a plotly plot from a distribution of values*


---

**Description**

This function creates a plotly plot with the distribution of values for each column in the mat matrix, using different colors for each column. The legend will show the column names from mat, and the plot will have the title "Distribution of read counts".

**Usage**

```
plot_hist(
  mat,
  title = NULL,
  xlab = "Values",
  ylab = "Frequency",
  show.grid = FALSE
)
```

**Arguments**

mat	A matrix with the data to plot.
title	A character scalar for the title of the plot.
xlab	Character scalar for label of the x-axis.
ylab	Character scalar for label of the y-axis.
show.grid	A boolean for whether to show the grid lines.

**Value**

A plotly plot with the distribution of read counts.

**Author(s)**

Jared Andrews

**Examples**

```
library(CRISPRball)
cts <- read.delim(system.file("extdata", "escneg.count_normalized.txt",
  package = "CRISPRball"
), check.names = FALSE)
cts.log <- as.matrix(log2(cts[, c(-1, -2)] + 1))
colnames(cts.log) <- colnames(cts)[c(-1, -2)]

plot_hist(cts.log,
  title = "Distribution of read counts",
  xlab = "log2(counts + 1)", ylab = "Frequency"
)
```

---

`plot_lawn`*Create an interactive lawn plot*

---

## Description

Create an interactive lawn plot for data with significance values. Typically, this plot is randomly ordered along the x-axis, but the user is free to order it by any term in `res` that they'd like.

## Usage

```
plot_lawn(  
  res,  
  ylim = 5,  
  fc.thresh = 0.5,  
  hover.info = NULL,  
  sig.line = TRUE,  
  h.id = "crispr",  
  feat.term = "rows",  
  x.term = "RandomIndex",  
  sig.term = "fdr",  
  lfc.term = "LFC",  
  down.color = "#0026ff",  
  up.color = "#ff0000",  
  insig.color = "#A6A6A6",  
  sig.thresh = 0.05,  
  fs = NULL,  
  sig.size = 6,  
  insig.size = 5,  
  sig.opacity = 1,  
  insig.opacity = 0.5,  
  label.size = 10,  
  webgl = TRUE,  
  webgl.ratio = 7,  
  show.counts = TRUE,  
  show.hl.counts = TRUE,  
  counts.size = TRUE,  
  highlight.featsets = NULL,  
  highlight.feats = NULL,  
  featsets = NULL,  
  highlight.feats.color = "#E69F00",  
  highlight.feats.size = 7,  
  highlight.feats.opac = 1,  
  highlight.feats.linecolor = "#000000",  
  highlight.feats.linewidth = 1,  
  highlight.feats.label = TRUE,  
  highlight.featsets.color = "#009E73",  
  highlight.featsets.size = 7,  
  highlight.featsets.opac = 1,  
  highlight.featsets.linecolor = "#000000",  
  highlight.featsets.linewidth = 1,  
  highlight.featsets.label = FALSE,
```

```

  h.id.suffix = "_lawn"
)

```

### Arguments

res	Dataframe containing, at minimum, significance values and a term by which the x-axis can be ordered.
ylim	Positive numeric scalar indicating y-axis limits. The negative value will be used for the lower limit.
fc.thresh	Numeric scalar indicating the fold change threshold for coloring significant features.
hover.info	Character vector indicating which additional columns from res to include in the hover info.
sig.line	Logical indicating whether to add a significance threshold line to the plot.
h.id	Character scalar indicating the unique ID of the plotly object. Can usually be ignored, but should be used if multiple plots are being created in the same R session (e.g. Shiny app).
feat.term	Character scalar indicating the column name of the feature IDs in res.
x.term	Character scalar for the x-axis term from res to be plotted.
sig.term	Character scalar indicating the column name of the significance values in res.
lfc.term	Character scalar indicating the column name of the log fold change values in res.
down.color	Character scalar indicating the color of down-regulated features.
up.color	Character scalar indicating the color of up-regulated features.
insig.color	Character scalar indicating the color of insignificant features.
sig.thresh	Numeric scalar indicating the significance threshold for coloring significant features.
fs	Dataframe containing coordinates and label information for points that should be labeled. Columns should be: <ul style="list-style-type: none"> <li>• x - x coordinate of the point</li> <li>• y - y coordinate of the point</li> <li>• customdata - label to be displayed</li> </ul>
sig.size	Numeric scalar indicating the size of significant feature points.
insig.size	Numeric scalar indicating the size of insignificant feature points.
sig.opacity	Numeric scalar indicating the opacity of significant feature points.
insig.opacity	Numeric scalar indicating the opacity of insignificant feature points.
label.size	Numeric scalar indicating the size of feature labels.
webgl	Logical indicating whether to use WebGL for rendering the plot.
webgl.ratio	Numeric scalar indicating the ratio of WebGL to HTML5 canvas rendering, increases resolution of saved plot when WebGL plotting is not used.
show.counts	Logical indicating whether to show annotations for the number of features in the plot.
show.hl.counts	Logical indicating whether to show annotations for the number of highlighted features in the plot.
counts.size	Numeric scalar indicating the size of the feature counts labels.

highlight.featsets	Character vector indicating which feature sets should be highlighted.
highlight.feats	Character vector indicating which features should be highlighted.
featsets	Named list of feature sets to be used for highlighting.
highlight.feats.color	Character scalar indicating the color of highlighted features.
highlight.feats.size	Numeric scalar indicating the size of highlighted features.
highlight.feats.opac	Numeric scalar indicating the opacity of highlighted features.
highlight.feats.linecolor	Character scalar indicating the line color of highlighted features.
highlight.feats.linewidth	Numeric scalar indicating the line width of highlighted features.
highlight.feats.label	Logical indicating whether to label highlighted features.
highlight.featsets.color	Character scalar indicating the color of highlighted feature sets.
highlight.featsets.size	Numeric scalar indicating the point size of highlighted feature sets.
highlight.featsets.opac	Numeric scalar indicating the opacity of highlighted feature sets.
highlight.featsets.linecolor	Character scalar indicating the line color of highlighted feature sets.
highlight.featsets.linewidth	Numeric scalar indicating the line width of highlighted feature sets.
highlight.featsets.label	Logical indicating whether to label highlighted feature sets.
h.id.suffix	Character scalar indicating the suffix to be added to the plotly object ID.

**Value**

An interactive plotly rank plot.

**Author(s)**

Jared Andrews

**Examples**

```
library(CRISPRball)
d1.genes <- read.delim(system.file("extdata", "esc1.gene_summary.txt",
  package = "CRISPRball"
), check.names = FALSE)
plot.df <- gene_ingress(d1.genes,
  sig.thresh = 0.05, es.thresh = 0.5,
  es.col = "LFC", sig.col = "fdr"
)
plot_lawn(plot.df, feat.term = "id")
```

---

plot_pca_biplot	<i>Plot a biplot from a PCAtools pca object</i>
-----------------	---

---

### Description

This function plots a biplot from a PCAtools [pca](#) object.

### Usage

```
plot_pca_biplot(  
  pca.res,  
  dim.x = "PC1",  
  dim.y = "PC2",  
  dim.z = NULL,  
  plot.title = "PCA Biplot",  
  color.by = NULL,  
  shape.by = NULL,  
  hover.info = NULL,  
  show.loadings = FALSE,  
  n.loadings = 3,  
  pt.size = 12  
)
```

### Arguments

<code>pca.res</code>	A <a href="#">pca</a> generated by the PCAtools package.
<code>dim.x</code>	Character scalar for the principal component to plot on the x-axis.
<code>dim.y</code>	Character scalar for the principal component to plot on the y-axis.
<code>dim.z</code>	Character scalar for the principal component to plot on the z-axis.
<code>plot.title</code>	Character scalar for the title of the plot.
<code>color.by</code>	Character scalar for the column name to use for coloring points.
<code>shape.by</code>	Character scalar for the column name to use for shaping points.
<code>hover.info</code>	Character vector of column name(s) to include in the hover info for each point.
<code>show.loadings</code>	Boolean indicating whether to show component loadings on the plot.
<code>n.loadings</code>	Integer for number of loadings to show.
<code>pt.size</code>	Numeric size of the plotted points.

### Value

A plotly plot of the PCA biplot, or a text grob indicating no PCA was provided.

### Author(s)

Jared Andrews

### See Also

[pca](#)

## Examples

```
library("PCAtools")
library("CRISPRball")
col <- 10
row <- 2000
mat <- matrix(
  rexp(col * row, rate = 0.1),
  ncol = col
)
rownames(mat) <- paste0("gene", seq_len(nrow(mat)))
colnames(mat) <- paste0("sample", seq_len(ncol(mat)))

metadata <- data.frame(row.names = colnames(mat))
metadata$Group <- rep(NA, ncol(mat))
metadata$Group[seq(1, 10, 2)] <- "A"
metadata$Group[seq(2, 10, 2)] <- "B"

p <- pca(mat, metadata = metadata, removeVar = 0.1)
plot_pca_biplot(p, color.by = "Group")
```

---

plot\_rank

*Create an interactive rank plot*

---

## Description

Create an interactive rank plot for data with fold change, significance terms, and rank.

## Usage

```
plot_rank(
  res,
  ylim = c(-10, 10),
  y.thresh = 0.5,
  y.lines = TRUE,
  hover.info = NULL,
  h.id = "crispr",
  feat.term = "rows",
  sig.term = "fdr",
  rank.term = "LFC",
  rank.ascending = TRUE,
  down.color = "#0026ff",
  up.color = "#ff0000",
  insig.color = "#A6A6A6",
  sig.thresh = 0.05,
  fs = NULL,
  sig.size = 6,
  insig.size = 5,
  sig.opacity = 1,
  insig.opacity = 0.5,
  label.size = 10,
  webgl = TRUE,
  webgl.ratio = 7,
```

```

show.counts = TRUE,
show.hl.counts = TRUE,
counts.size = TRUE,
highlight.featsets = NULL,
highlight.feats = NULL,
featsets = NULL,
highlight.feats.color = "#E69F00",
highlight.feats.size = 7,
highlight.feats.opac = 1,
highlight.feats.linecolor = "#000000",
highlight.feats.linewidth = 1,
highlight.feats.label = TRUE,
highlight.featsets.color = "#009E73",
highlight.featsets.size = 7,
highlight.featsets.opac = 1,
highlight.featsets.linecolor = "#000000",
highlight.featsets.linewidth = 1,
highlight.featsets.label = FALSE,
h.id.suffix = "_volc"
)

```

### Arguments

res	Dataframe containing, at minimum, significance values and something to rank by (LFC, RRA score, betas, etc).
y.lim	Numeric vector of length two for the y-axis limits.
y.thresh	Numeric scalar used as the y-axis threshold for point coloring. The negative of this value is also used as the threshold.
y.lines	Logical as for whether or not to show horizontal lines at y.thresh.
hover.info	Character vector indicating which additional columns from res to include in the hover info.
h.id	Character scalar indicating the unique ID of the plotly object. Can usually be ignored, but should be used if multiple plots are being created in the same R session (e.g. Shiny app).
feat.term	Character scalar indicating the column name of the feature IDs in res.
sig.term	Character scalar indicating the column name of the significance values in res.
rank.term	Character scalar for the term to rank by from res. This will be used as the y-axis.
rank.ascending	Boolean indicating whether or not the rank should be ascending.
down.color	Character scalar indicating the color of down-regulated features.
up.color	Character scalar indicating the color of up-regulated features.
insig.color	Character scalar indicating the color of insignificant features.
sig.thresh	Numeric scalar indicating the significance threshold for coloring significant features.
fs	Dataframe containing coordinates and label information for points that should be labeled. Columns should be: <ul style="list-style-type: none"> <li>• x - x coordinate of the point</li> <li>• y - y coordinate of the point</li> </ul>



	<ul style="list-style-type: none"> <li>• customdata - label to be displayed</li> </ul>
sig.size	Numeric scalar indicating the size of significant feature points.
insig.size	Numeric scalar indicating the size of insignificant feature points.
sig.opacity	Numeric scalar indicating the opacity of significant feature points.
insig.opacity	Numeric scalar indicating the opacity of insignificant feature points.
label.size	Numeric scalar indicating the size of feature labels.
webgl	Logical indicating whether to use WebGL for rendering the plot.
webgl.ratio	Numeric scalar indicating the ratio of WebGL to HTML5 canvas rendering, increases resolution of saved plot when WebGL plotting is not used.
show.counts	Logical indicating whether to show annotations for the number of features in the plot.
show.hl.counts	Logical indicating whether to show annotations for the number of highlighted features in the plot.
counts.size	Numeric scalar indicating the size of the feature counts labels.
highlight.featsets	Character vector indicating which feature sets should be highlighted.
highlight.feats	Character vector indicating which features should be highlighted.
featsets	Named list of feature sets to be used for highlighting.
highlight.feats.color	Character scalar indicating the color of highlighted features.
highlight.feats.size	Numeric scalar indicating the size of highlighted features.
highlight.feats.opac	Numeric scalar indicating the opacity of highlighted features.
highlight.feats.linecolor	Character scalar indicating the line color of highlighted features.
highlight.feats.linewidth	Numeric scalar indicating the line width of highlighted features.
highlight.feats.label	Logical indicating whether to label highlighted features.
highlight.featsets.color	Character scalar indicating the color of highlighted feature sets.
highlight.featsets.size	Numeric scalar indicating the point size of highlighted feature sets.
highlight.featsets.opac	Numeric scalar indicating the opacity of highlighted feature sets.
highlight.featsets.linecolor	Character scalar indicating the line color of highlighted feature sets.
highlight.featsets.linewidth	Numeric scalar indicating the line width of highlighted feature sets.
highlight.featsets.label	Logical indicating whether to label highlighted feature sets.
h.id.suffix	Character scalar indicating the suffix to be added to the plotly object ID.

**Value**

An interactive plotly rank plot.

**Author(s)**

Jared Andrews

**Examples**

```
library(CRISPRball)
d1.genes <- read.delim(system.file("extdata", "esc1.gene_summary.txt",
  package = "CRISPRball"
), check.names = FALSE)
plot.df <- gene_ingress(d1.genes,
  sig.thresh = 0.05, es.thresh = 0.5,
  es.col = "LFC", sig.col = "fdr"
)
plot_rank(plot.df, feat.term = "id")
```

---

plot\_volcano

*Create an interactive volcano plot*

---

**Description**

Create an interactive volcano plot for data with fold change and significance terms.

**Usage**

```
plot_volcano(
  res,
  xlim = 5,
  ylim = 5,
  fc.thresh = 0.5,
  fc.lines = TRUE,
  hover.info = NULL,
  sig.line = TRUE,
  h.id = "crispr",
  feat.term = "rows",
  sig.term = "fdr",
  lfc.term = "LFC",
  down.color = "#0026ff",
  up.color = "#ff0000",
  insig.color = "#A6A6A6",
  sig.thresh = 0.05,
  fs = NULL,
  sig.size = 6,
  insig.size = 5,
  sig.opacity = 1,
  insig.opacity = 0.5,
  label.size = 10,
  webgl = TRUE,
  webgl.ratio = 7,
```

```

show.counts = TRUE,
show.hl.counts = TRUE,
counts.size = 8,
highlight.featsets = NULL,
highlight.feats = NULL,
featsets = NULL,
highlight.feats.color = "#E69F00",
highlight.feats.size = 7,
highlight.feats.opac = 1,
highlight.feats.linecolor = "#000000",
highlight.feats.linewidth = 1,
highlight.feats.label = TRUE,
highlight.featsets.color = "#009E73",
highlight.featsets.size = 7,
highlight.featsets.opac = 1,
highlight.featsets.linecolor = "#000000",
highlight.featsets.linewidth = 1,
highlight.featsets.label = FALSE,
h.id.suffix = "_volc"
)

```

### Arguments

res	Dataframe containing, at minimum, fold change and significance values.
xlim	Positive numeric scalar indicating x-axis limits. The negative value will be used for the lower limit.
ylim	Positive numeric scalar indicating y-axis limits. The negative value will be used for the lower limit.
fc.thresh	Numeric scalar indicating the fold change threshold for coloring significant features.
fc.lines	Logical indicating whether to add fold change threshold lines to the plot.
hover.info	Character vector indicating which additional columns from res to include in the hover info.
sig.line	Logical indicating whether to add a significance threshold line to the plot.
h.id	Character scalar indicating the unique ID of the plotly object. Can usually be ignored, but should be used if multiple plots are being created in the same R session (e.g. Shiny app).
feat.term	Character scalar indicating the column name of the feature IDs in res.
sig.term	Character scalar indicating the column name of the significance values in res.
lfc.term	Character scalar indicating the column name of the log fold change values in res.
down.color	Character scalar indicating the color of down-regulated features.
up.color	Character scalar indicating the color of up-regulated features.
insig.color	Character scalar indicating the color of insignificant features.
sig.thresh	Numeric scalar indicating the significance threshold for coloring significant features.
fs	Dataframe containing coordinates and label information for points that should be labeled. Columns should be:

- x - x coordinate of the point
- y - y coordinate of the point
- customdata - label to be displayed

sig.size	Numeric scalar indicating the size of significant feature points.
insig.size	Numeric scalar indicating the size of insignificant feature points.
sig.opacity	Numeric scalar indicating the opacity of significant feature points.
insig.opacity	Numeric scalar indicating the opacity of insignificant feature points.
label.size	Numeric scalar indicating the size of feature labels.
webgl	Logical indicating whether to use WebGL for rendering the plot.
webgl.ratio	Numeric scalar indicating the ratio of WebGL to HTML5 canvas rendering, increases resolution of saved plot when WebGL plotting is not used.
show.counts	Logical indicating whether to show annotations for the number of features in the plot.
show.hl.counts	Logical indicating whether to show annotations for the number of highlighted features in the plot.
counts.size	Numeric scalar indicating the size of the feature counts labels.
highlight.featsets	Character vector indicating which feature sets should be highlighted.
highlight.feats	Character vector indicating which features should be highlighted.
featsets	Named list of feature sets to be used for highlighting.
highlight.feats.color	Character scalar indicating the color of highlighted features.
highlight.feats.size	Numeric scalar indicating the size of highlighted features.
highlight.feats.opac	Numeric scalar indicating the opacity of highlighted features.
highlight.feats.linecolor	Character scalar indicating the line color of highlighted features.
highlight.feats.linewidth	Numeric scalar indicating the line width of highlighted features.
highlight.feats.label	Logical indicating whether to label highlighted features.
highlight.featsets.color	Character scalar indicating the color of highlighted feature sets.
highlight.featsets.size	Numeric scalar indicating the point size of highlighted feature sets.
highlight.featsets.opac	Numeric scalar indicating the opacity of highlighted feature sets.
highlight.featsets.linecolor	Character scalar indicating the line color of highlighted feature sets.
highlight.featsets.linewidth	Numeric scalar indicating the line width of highlighted feature sets.
highlight.featsets.label	Logical indicating whether to label highlighted feature sets.
h.id.suffix	Character scalar indicating the suffix to be added to the plotly object ID.

**Value**

An interactive plotly volcano plot.

**Author(s)**

Jared Andrews

**Examples**

```
library(CRISPRball)
d1.genes <- read.delim(system.file("extdata", "esc1.gene_summary.txt",
  package = "CRISPRball"
), check.names = FALSE)
plot.df <- gene_ingress(d1.genes,
  sig.thresh = 0.05, es.thresh = 0.5,
  es.col = "LFC", sig.col = "fdr"
)
plot_volcano(plot.df, feat.term = "id")
```

---

read\_mle\_gene\_summary *Read and parse MAGECK MLE output gene summary file*

---

**Description**

This function reads the gene summary file output by mageck mle and parses it into a list of data.frames, one for each sample. The sample names are extracted from the column names of the input file and used as the names of the list elements.

**Usage**

```
read_mle_gene_summary(filepath)
```

**Arguments**

filepath            Path to the gene summary file output by mageck mle.

**Value**

A named list of data.frames containing MAGECK MLE output, one for each sample contained in the file.

**Author(s)**

Jared Andrews

**Examples**

```
library(CRISPRball)
mle_gene_summary <- file.path(system.file("extdata", "beta_leukemia.gene_summary.txt",
  package = "CRISPRball"
))
gene_data <- read_mle_gene_summary(mle_gene_summary)
```

# Index

## \* datasets

- depmap\_22q1\_cn, 6
- depmap\_22q1\_crispr, 6
- depmap\_22q1\_crispr\_rnai, 7
- depmap\_22q1\_rnai, 8
- depmap\_22q1\_TPM, 9

BarView, 13

build\_depmap\_db, 2, 4, 11

CRISPRball, 3, 4

dbPool, 3

dbWriteTable, 3

depmap\_22q1\_cn, 6

depmap\_22q1\_crispr, 6

depmap\_22q1\_crispr\_rnai, 7

depmap\_22q1\_rnai, 8

depmap\_22q1\_TPM, 9

depmap\_copyNumber, 3

depmap\_crispr, 3

depmap\_drug\_sensitivity, 3

depmap\_gene\_summary, 3

depmap\_metadata, 3

depmap\_release, 3

depmap\_rnai, 3

depmap\_TPM, 3

gene\_ingress, 9

get\_depmap\_essentiality, 10

get\_depmap\_plot\_data, 11, 14–17

pca, 4, 22

plot\_bar, 12

plot\_correlation\_heatmap, 13

plot\_depmap\_cn, 14

plot\_depmap\_dependency, 15

plot\_depmap\_expression, 16

plot\_depmap\_lineages, 12, 17

plot\_hist, 18

plot\_lawn, 4, 19

plot\_pca\_biplot, 22

plot\_rank, 4, 23

plot\_volcano, 4, 26

read\_mle\_gene\_summary, 29