

# Package ‘ScreenR’

March 28, 2025

**Type** Package

**Title** Package to Perform High Throughput Biological Screening

**Version** 1.8.0

**Date** 2022-04-01

**Description** ScreenR is a package suitable to perform hit identification in loss of function High Throughput Biological Screenings performed using barcoded shRNA-based libraries. ScreenR combines the computing power of software such as edgeR with the simplicity of use of the Tidyverse metapackage. ScreenR executes a pipeline able to find candidate hits from barcode counts, and integrates a wide range of visualization modes for each step of the analysis.

**License** MIT + file LICENSE

**Encoding** UTF-8

**RoxygenNote** 7.2.0

**Imports** methods (>= 4.0), rlang (>= 0.4), stringr (>= 1.4), limma (>= 3.46), patchwork (>= 1.1), tibble (>= 3.1.6), scales (>= 1.1.1), ggvenn (>= 0.1.9), purrr (>= 0.3.4), ggplot2 (>= 3.3), stats, tidyr (>= 1.2), magrittr (>= 1.0), dplyr (>= 1.0), edgeR (>= 3.32), tidyselect (>= 1.1.2)

**Suggests** rmarkdown (>= 2.11), knitr (>= 1.37), testthat (>= 3.0.0), BiocStyle (>= 2.22.0), covr (>= 3.5)

**Config/testthat/edition** 3

**Depends** R (>= 4.2)

**VignetteBuilder** knitr

**biocViews** Software, AssayDomain, GeneExpression

**BugReports** <https://github.com/EmanuelSoda/ScreenR/issues>

**URL** <https://emanuelsoda.github.io/ScreenR/>

**Collate** 'ScreenR-package.R' 'annotation\_table.R' 'barcode\_lost.R' 'camera\_method.R' 'compute\_data\_table.R' 'compute\_metrics.R' 'count\_mapped\_reads.R' 'count\_table.R' 'create\_object.R' 'distribution\_mapped\_reads.R' 'filter\_by.R' 'find\_common\_hit.R' 'find\_roast\_hit.R' 'find\_robust\_zscore\_hit.R' 'find\_zscore\_hit.R' 'generics.R' 'mapped\_reads.R' 'normalize\_data.R' 'plot\_barcode\_hit.R' 'plot\_barcode\_trend.R'

'plot\_boxplot.R' 'plot\_mapped\_reads.R' 'plot\_mds.R'  
 'plot\_trend.R' 'plot\_zscore\_distribution.R' 'screenr-class.R'  
 'zzz.R'

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

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

**git\_last\_commit** 5c07e14

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

**Repository** Bioconductor 3.20

**Date/Publication** 2025-03-27

**Author** Emanuel Michele Soda [aut, cre] (0000-0002-2301-6465),  
 Elena Ceccacci [aut] (0000-0002-2285-8994),  
 Saverio Minucci [fnd, ths] (0000-0001-5678-536X)

**Maintainer** Emanuel Michele Soda <emanuelsoda@gmail.com>

## Contents

ScreenR-package . . . . .	3
annotation_table . . . . .	4
barcode_lost . . . . .	4
compute_camera . . . . .	5
compute_data_table . . . . .	6
compute_explained_variance . . . . .	6
compute_metrics . . . . .	7
compute_slope . . . . .	7
compute_trend . . . . .	8
count_mapped_reads . . . . .	9
count_table . . . . .	9
create_edger_obj . . . . .	10
create_screenr_object . . . . .	11
filter_by_slope . . . . .	12
filter_by_variance . . . . .	13
find_camera_hit . . . . .	14
find_common_hit . . . . .	15
find_roast_hit . . . . .	16
find_robust_zscore_hit . . . . .	17
find_zscore_hit . . . . .	17
get_annotation_table . . . . .	18
get_count_table . . . . .	19
get_data_table . . . . .	20
get_groups . . . . .	20
get_normalized_count_table . . . . .	21
get_replicates . . . . .	21
mapped_reads . . . . .	22
normalize_data . . . . .	22
plot_barcode_hit . . . . .	23
plot_barcode_lost . . . . .	24
plot_barcode_lost_for_gene . . . . .	25
plot_barcode_trend . . . . .	25
plot_boxplot . . . . .	26

plot_common_hit . . . . .	27
plot_explained_variance . . . . .	29
plot_mapped_reads . . . . .	29
plot_mapped_reads_distribution . . . . .	30
plot_mds . . . . .	31
plot_trend . . . . .	32
plot_zscore_distribution . . . . .	33
remove_all_zero_row . . . . .	33
select_number_barcode . . . . .	34
set_annotation_table . . . . .	35
set_count_table . . . . .	35
set_data_table . . . . .	36
set_groups . . . . .	36
set_normalized_count_table . . . . .	37
set_replicates . . . . .	38
unique_gene_symbols . . . . .	38

<b>Index</b>	<b>39</b>
--------------	-----------

---

ScreenR-package

*Tools for analyzing shRNAs screening data*

---

## Description

**ScreenR** is an easy and effective package to perform hits identification in loss of function High Throughput Biological Screening performed with shRNAs library. ScreenR combines the power of software like edgeR with the simplicity of the Tidyverse metapackage. ScreenR executes a pipeline able to find candidate hits from barcode counts data and integrates a wide range of visualization for each step of the analysis.

## Details

**ScreenR** takes the a count table as input and create the screenr\_object to perform the analysis. Throught the pipeline **ScreenR** enable the user to perform quality control, visual inspection, dimensionality reduction of the data. Using three statistical methods:

- **ROAST**
- **CAMERA**
- **Z-score**

it is able to find new candidate hits. Moreover in order to improve the quality of the hit found it is also possible to further filter the list of hit using other filter like the variance and the slope filters.

## Author(s)

Emanuel Michele Soda <emanuelsoda@gmail.com>

---

annotation_table	<i>Table for the annotation of Barcode</i>
------------------	--

---

**Description**

Table for the annotation of Barcode

**Usage**

```
data(annotation_table)
```

**Format**

A data frame with 5320 rows and 2 columns obtained from a loss-of-function genetic screening. This table is used to store information about the shRNAs:

**Gene** It Contains the gene name

**Barcode** It contains an ID that identify each barcode (it is an unique identifier for an shRNA). It can be use to merge the annotation table with t he count table

**Gene\_ID** It Contains a unique Gene ID

**Sequence** It contains the cDNA sequence of the shRNA associated to the barcode

**Library** It contains the library from which the shRNA come from. In this case is a pooled from <https://collecta.com/collecta>

---

barcode_lost	<i>Count number of barcode lost</i>
--------------	-------------------------------------

---

**Description**

This function counts the number of barcodes lost during the sequencing. A barcode is lost if its associated shRNA has zero mapped read in a sample.

**Usage**

```
barcode_lost(screenR_Object)
```

**Arguments**

screenR\_Object The ScreenR object obtained using the [create\\_screenr\\_object](#)

**Value**

Return a tibble containing the number of barcode lost for each sample

**Examples**

```
object <- get0("object", envir = asNamespace("ScreenR"))

# In order to count the number of barcodes lost just the ScreenR object is
# needed
head(barcode_lost(object))
```

---

compute_camera	<i>Compute Camera</i>
----------------	-----------------------

---

## Description

This internal function computes the actual hits using the camera method.

## Usage

```
compute_camera(  
  xglm,  
  lrt,  
  DGEList,  
  matrix_model,  
  contrast,  
  number_barcode = 3,  
  thresh = 1e-04,  
  lfc = 1  
)
```

## Arguments

xglm	object created with <a href="#">estimateDisp</a>
lrt	object created with <a href="#">glmFit</a>
DGEList	edgeR object
...	Arguments passed on to <a href="#">find_camera_hit</a>
matrix_model	The matrix that will be used to perform the linear model analysis created using <a href="#">model.matrix</a>
thresh	The threshold for the False Discovery Rate (FDR) that has to be used to select the statistically significant hits.
lfc	The Log2FC threshold.
number_barcode	Number of barcode that as to be differentially expressed (DE) in order to consider the gene associated DE. Example a gene is associated with 10 shRNA we consider a gene DE if it has at least number_barcode = 5 shRNA DE.

## Value

The list of hits found by the camera method

---

compute\_data\_table      *Compute data Table*

---

### Description

This function computes the data table that will be used for the analysis. The data\_table is a tidy and normalized version of the original count\_table and will be used throughout the analysis.

### Usage

```
compute_data_table(screenR_Object)
```

### Arguments

screenR\_Object The ScreenR object obtained using the [create\\_screenr\\_object](#)

### Value

ScreenR\_Object with the data\_table filed containing the table.

### Examples

```
object <- get0("object", envir = asNamespace("ScreenR"))
object <- compute_data_table(object)
head(slot(object, "data_table"))
```

---

compute\_explained\_variance  
                                   *Compute explained variance*

---

### Description

This is an internal function used to compute the explained variance by each of the Principal Components.

### Usage

```
compute_explained_variance(screenR_Object)
```

### Arguments

screenR\_Object The Object of the package

### Value

A data.frame containing all the information of the variance expressed by the components

### Examples

```
object <- get0("object", envir = asNamespace("ScreenR"))

compute_explained_variance(object)
```

---

compute_metrics	<i>Compute Metrics</i>
-----------------	------------------------

---

### Description

This function computes the metrics that will be then used to compute the z-score using the function [find\\_zscore\\_hit](#) starting from the screenr object for a given treatment in a given day. More information about the z-score and other metrics used in genetic screening can be found at this paper [z-score](#)

### Usage

```
compute_metrics(screenR_Object, control, treatment, day)
```

### Arguments

screenR_Object	The ScreenR object obtained using the <a href="#">create_screenr_object</a>
control	A string specifying the sample that as to be used as control in the analysis. This string has to be equal to the interested sample in the Treatment column of the data_table slot
treatment	A string specifying the sample that as to be used as treatment in the analysis. This string has to be equal to the interested sample in the Treatment column of the data_table slot.
day	A string containing the day (time point) to consider in the metrics computation. This string has to be equal to the interested sample in the Day column of the data_table slot.

### Value

Return a tibble with all the measure computed.

### Examples

```
object <- get0("object", envir = asNamespace("ScreenR"))
metrics <- compute_metrics(object,
  control = "TRT",
  treatment = "Time3", day = "Time3"
)
head(metrics)
```

---

compute_slope	<i>Compute Slope of a Gene</i>
---------------	--------------------------------

---

### Description

This function is used to compute the slope of the gene passed as input

### Usage

```
compute_slope(screenR_Object, genes, group_var)
```

**Arguments**

- screenR\_Object The ScreenR object obtained using the [create\\_screenr\\_object](#)
- genes The genes for which the slope as to be computed. Those genes are the result of the three statistical methods selection
- group\_var The variable to use as independent variable (x) for the linear model

**Value**

A tibble containing in each row the gene and the corresponding Slope

**Examples**

```
object <- get0("object", envir = asNamespace("ScreenR"))

compute_slope(object,
  genes = c("Gene_42", "Gene_24"),
  group_var = c("T1", "T2", "TRT")
)
```

---

compute_trend	<i>Compute trend</i>
---------------	----------------------

---

**Description**

This is an internal function used to computes the trend of a gene

**Usage**

```
compute_trend(screenR_Object, genes, group_var)
```

**Arguments**

- screenR\_Object object created with [estimateDisp](#)
- genes a list of genes
- group\_var the variable that as to be used as grouping variable

**Value**

A table with the trend of the genes passed as input



---

count_mapped_reads	<i>Count the number of mapped read</i>
--------------------	--

---

### Description

This function counts the number of reads for each barcode in each sample. It is a quality control function (QC) to see if the biological protocol went as planned. If a sample has very low mapped compared to the other means that it has a lower quality.

### Usage

```
count_mapped_reads(screenR_Object)
```

### Arguments

screenR\_Object The ScreenR object obtained using the [create\\_screenr\\_object](#)

### Value

Return a tibble containing the number of mapped read for sample

### Examples

```
object <- get0("object", envir = asNamespace("ScreenR"))
head(count_mapped_reads(object))
```

---

count_table	<i>Table of the count table</i>
-------------	---------------------------------

---

### Description

Table of the count table

### Usage

```
data(count_table)
```

### Format

A data frame with 5323 rows and 15 variables obtained from barcode alignment to the reference library. It is generated from a [Cellecta](https://cellecta.com/) protocol. The samples generated are then sequenced using an RNA-seq protocol. Due to the fact that different shRNAs are sequenced for a gene each barcode has its associated reads. These reads were aligned to the reference library using [bowtie2](http://bowtie-bio.sourceforge.net/bowtie2/index.shtml) and then sorted with [samtools](https://github.com/samtools/samtools). Since this dataset comes from a Chemical Synthetic Lethality experiment the samples treated and combined with the shRNAs knockdown should present a decreased number of reads compared to the controls.

**Barcode** It contains an ID that identify each barcode. It can be use to marge the annotation table with the count table. A Barcode is a unique identifier of an shRNA. In a genetic screening multiple slightly different shRNAs perform a knockout of a gene each with its efficacy. For this reason it is important to keep track of each shRNA using a unique barcode.

**Time\_1** It contains the counts at time zero. This is the first time point at which cells are not treated and not infected.

**Time\_2** It contains the counts after the cell were washed. At this time point the cells are infected and following the Collecta protocol are washed with the puromycin.

**Time\_3\_TRT\_rep1** It contains the counts for the first replicate of the treated at the first time point. Usually the first time point is 7 day after the puromycin wash.

**Time\_3\_TRT\_rep2** It contains the counts for the second replicate of the treated at the first time point. Usually the first time point is 7 day after the puromycin wash.

**Time\_3\_TRT\_rep3** It contains the counts for the third replicate of the treated at the first time point. Usually the first time point is 7 day after the puromycin wash.

**Time\_3\_rep1** It contains the counts for the first replicate of the control at the first time point. Usually the first time point is 7 day after the puromycin wash.

**Time\_3\_rep2** It contains the counts for the second replicate of the control at the first time point. Usually the first time point is 7 day after the puromycin wash.

**Time\_3\_rep3** It contains the counts for the third replicate of the control at the first time point. Usually the first time point is 7 day after the puromycin wash.

**Time\_4\_TRT\_rep1** It contains the counts for the first replicate of the treated at the second time point. Usually the first time point is 14 day after the puromycin wash.

**Time\_4\_TRT\_rep2** It contains the counts for the second replicate of the treated at the second time point. Usually the first time point is 14 day after the puromycin wash.

**Time\_4\_TRT\_rep3** It contains the counts for the third replicate of the treated at the second time point. Usually the first time point is 14 day after the puromycin wash.

**Time\_4\_rep1** It contains the counts for the first replicate of the control at the second time point. Usually the first time point is 14 day after the puromycin wash.

**Time\_4\_rep2** It contains the counts for the second replicate of the control at the second time point. Usually the first time point is 14 day after the puromycin wash.

**Time\_4\_rep3** It contains the counts for the third replicate of the control at the second time point. Usually the first time point is 14 day after the puromycin wash.

---

create\_edger\_obj      *Create edgeR Object*

---

## Description

Utility function that using the screenr-class object create the corresponding edgeR object. This function and other utility function enables the user to not worry about the implementation and just focus on the analysis. The ScreenR package will take care of the rest.

## Usage

```
create_edger_obj(screenR_Object)
```

**Arguments**

screenR\_Object The ScreenR object obtained using the [create\\_screenr\\_object](#)

**Value**

The edgeR object will all the needed information for the analysis.

**Examples**

```
object <- get0("object", envir = asNamespace("ScreenR"))
create_edger_obj(object)
```

---

create\_screenr\_object *Create the ScreenR Object*

---

**Description**

Initial function to create the Screen Object.

**Usage**

```
create_screenr_object(
  table = NULL,
  annotation = NULL,
  groups = NULL,
  replicates = c("")
)
```

**Arguments**

table	The count table obtained from the read alignment that contains the Barcodes as rows and samples as columns.
annotation	The annotation table containing the information for each Barcode and the association to the corresponding Gene
groups	A factor containing the experimental design label
replicates	A vector containing the replicates label

**Value**

An object containing all the needed information for the analysis.

**Examples**

```
count_table <-
  data.frame(
    Barcode = c("Code_1", "Code_2", "Code_3", "Code_3"),
    Time_3_rep1 = c("3520", "3020", "1507", "1400"),
    Time_3_rep2 = c("3500", "3000", "1457", "1490"),
    Time_3_TRT_rep1 = c("1200", "1100", "1300", "1350"),
    Time_3_TRT_rep2 = c("1250", "1000", "1400", "1375")
  )
```

```

annotation_table <-
  data.frame(
    Gene = c("Gene_1", "Gene_1", "Code_2", "Code_2"),
    Barcode = c("Code_1", "Code_2", "Code_3", "Code_3"),
    Gene_ID = rep(NA, 4), Sequence = rep(NA, 4),
    Library = rep(NA, 4)
  )

groups <- factor(c("Control", "Control", "Treated", "Treated"))
obj <- create_screenr_object(
  table = count_table,
  annotation = annotation_table,
  groups = groups, replicates = c("")
)
obj

```

---

filter_by_slope	<i>Filter using the slope filter</i>
-----------------	--------------------------------------

---

### Description

This function is used to improve the quality of the hits found. It computes a regression line in the different samples and uses the slope of this line to see the trend

### Usage

```

filter_by_slope(
  screenR_Object,
  genes,
  group_var_treatment,
  group_var_control,
  slope_control,
  slope_treatment
)

```

### Arguments

screenR_Object	The ScreenR object obtained using the <a href="#">create_screenr_object</a>
genes	The genes for which the slope as to be computed. Those genes are the result of the three statistical methods selection
group_var_treatment	The variable to use as independent variable (x) for the linear model of the treatment
group_var_control	The variable to use as independent variable (x) for the linear model of the control
slope_control	A value used as threshold for the control slope
slope_treatment	A value used as threshold for the treatment slope

### Value

A data frame with the slope for the treatment and the control for each gene

**Examples**

```
object <- get0("object", envir = asNamespace("ScreenR"))

filter_by_slope(
  screenR_Object = object, genes = c("Gene_1", "Gene_2"),
  group_var_treatment = c("T1", "T2", "TRT"),
  group_var_control = c("T1", "T2", "Time3", "Time4"),
  slope_control = 0.5, slope_treatment = 1
)
```

---

filter\_by\_variance      *Filter using the variance filter*

---

**Description**

This function is used to improve the quality of the hits. It compute the variance among the hits and filter the one with a value greater than the threshold set

**Usage**

```
filter_by_variance(
  screenR_Object,
  genes,
  matrix_model,
  variance = 0.5,
  contrast
)
```

**Arguments**

screenR_Object	The ScreenR object obtained using the <a href="#">create_screenr_object</a>
genes	The genes for which the variance as to be computed. Those genes are the result of the three statistical methods selection
matrix_model	a matrix created using <a href="#">model.matrix</a>
variance	The maximum value of variance accepted
contrast	The variable to use as X for the linear model for the Treatment

**Value**

A data frame with the variance for the treatment and the control for each gene

**Examples**

```
object <- get0("object", envir = asNamespace("ScreenR"))
matrix_model <- model.matrix(~ slot(object, "groups"))
colnames(matrix_model) <- c("Control", "T1_T2", "Treated")
contrast <- limma::makeContrasts(Treated - Control, levels = matrix_model)

data <- filter_by_variance(
  screenR_Object = object, genes = c("Gene_42"),
```

```

    matrix_model = matrix_model, contrast = contrast
  )
  head(data)

```

---

find_camera_hit	<i>Find Camera Hit</i>
-----------------	------------------------

---

## Description

This function implements the method by proposed by Wu and Smyth (2012). The original [camera](#) method is a gene set test, here is applied in the context of a genetic screening and so it performs a competitive barcode set test. The paper can be found here [CAMERA](#)

## Usage

```

find_camera_hit(
  screenR_Object,
  matrix_model,
  contrast,
  number_barcode = 3,
  thresh = 1e-04,
  lfc = 1,
  direction = "Down"
)

```

## Arguments

screenR_Object	The ScreenR object obtained using the <a href="#">create_screenr_object</a>
matrix_model	The matrix that will be used to perform the linear model analysis created using <a href="#">model.matrix</a>
contrast	A vector or a single value indicating the index or the name of the column the model_matrix with which perform the analysis
number_barcode	Number of barcode that as to be differentially expressed (DE) in order to consider the gene associated DE. Example a gene is associated with 10 shRNA we consider a gene DE if it has at least number_barcode = 5 shRNA DE.
thresh	The threshold for the False Discovery Rate (FDR) that has to be used to select the statistically significant hits.
lfc	The Log2FC threshold.
direction	String containing the direction of the variation, "Down" for the down regulation "Up" for the up regulation.

## Value

The data frame containing the hit found using the camera method

**Examples**

```

object <- get0("object", envir = asNamespace("ScreenR"))

matrix <- model.matrix(~ slot(object, "groups"))
colnames(matrix) <- c("Control", "T1/T2", "Treated")

result <- find_camera_hit(
  screenR_Object = object,
  matrix_model = matrix, contrast = "Treated"
)
head(result)

```

---

find_common_hit	<i>Find common hit</i>
-----------------	------------------------

---

**Description**

This method find the hit in common between the three methods

**Usage**

```
find_common_hit(hit_zscore, hit_camera, hit_roast, common_in = 3)
```

**Arguments**

hit_zscore	The matrix obtained by the <a href="#">find_zscore_hit</a> method
hit_camera	The matrix obtained by the <a href="#">find_camera_hit</a> method
hit_roast	The matrix obtained by the <a href="#">find_roast_hit</a> method
common_in	Number of methods in which the hit has to be in common in order to be considered a candidate hit. The default value is 3, which means that has to be present in the result of all the three methods.

**Value**

A vector containing the common hit

**Examples**

```

hit_zscore <- data.frame(Gene = c("A", "B", "C", "D", "E"))
hit_camera <- data.frame(Gene = c("A", "B", "C", "F", "H", "G"))
hit_roast <- data.frame(Gene = c("A", "L", "N"))

# common among all the three methods
find_common_hit(hit_zscore, hit_camera, hit_roast)

# common among at least two of the three methods
find_common_hit(hit_zscore, hit_camera, hit_roast, common_in = 2)

```

---

find_roast_hit	<i>Find Roast Hit</i>
----------------	-----------------------

---

### Description

Find the hit using the roast method. Roast is a competitive gene set test which uses rotation instead of permutation. Here is applied in a contest of a genetic screening so it perform a barcode competitive test testing for barcode which are differentially expressed within a gene. More information can be found in [Roast](#)

### Usage

```
find_roast_hit(
  screenR_Object,
  matrix_model,
  contrast,
  nrot = 9999,
  number_barcode = 3,
  direction = "Down",
  p_val = 0.05
)
```

### Arguments

screenR_Object	The ScreenR object obtained using the <a href="#">create_screenr_object</a>
matrix_model	The matrix that will be used to perform the linear model analysis. Created using <a href="#">model.matrix</a>
contrast	A vector or a single value indicating the index or the name of the column the model_matrix to which perform the analysis
nrot	Number of rotation to perform the test. Higher number of rotation leads to more statistically significant result.
number_barcode	Number of barcode that as to be differentially expressed (DE)in order to consider the gene associated DE. Example a gene is associated with 10 shRNA we consider a gene DE if it has at least number_barcode = 5 shRNA DE.
direction	Direction of variation
p_val	The value that as to be used as p-value cut off

### Value

The hits found by ROAST method

### Examples

```
set.seed(42)
object <- get0("object", envir = asNamespace("ScreenR"))
matrix_model <- model.matrix(~ slot(object, "groups"))
colnames(matrix_model) <- c("Control", "T1_T2", "Treated")

result <- find_roast_hit(object,
  matrix_model = matrix_model,
```



```

      contrast = "Treated", nrot = 100
    )
  head(result)

```

---

```
find_robust_zscore_hit
```

*Title Find robust Z-score Hit*

---

### Description

Title Find robust Z-score Hit

### Usage

```
find_robust_zscore_hit(table_treat_vs_control, number_barcode)
```

### Arguments

table\_treat\_vs\_control

A table computed with the function compute\_data\_table. It contain for each barcode the associated Gene the counts in the treated and control and the value for the Log2FC, Zscore, ZscoreRobust in each day.

number\_barcode Number of barcode that as to be differentially expressed (DE)in order to consider the gene associated DE. Example a gene is associated with 10 shRNA we consider a gene DE if it has at least number\_barcode = 5 shRNA DE.

### Value

Return a tibble containing the hit for the robust Z-score

### Examples

```

object <- get0("object", envir = asNamespace("ScreenR"))
table <- compute_metrics(object,
  control = "TRT", treatment = "Time3",
  day = "Time3"
)
result <- find_robust_zscore_hit(table, number_barcode = 6)
head(result)

```

---

```
find_zscore_hit
```

*Title Find Z-score Hit*

---

### Description

Title Find Z-score Hit

### Usage

```
find_zscore_hit(table_treat_vs_control, number_barcode = 6, metric = "median")
```

**Arguments**

`table_treat_vs_control` table computed with the function `compute_data_table`

`number_barcode` Number of barcode that as to be differentially expressed (DE) in order to consider the gene associated DE. Example a gene is associated with 10 shRNA we consider a gene DE if it has at least `number_barcode = 5` shRNA DE.

`metric` A string containing the metric to use. The value allowed are "median" or "mean".

**Value**

Return a tibble containing the hit for the Z-score

**Examples**

```
object <- get0("object", envir = asNamespace("ScreenR"))
table <- compute_metrics(object,
  control = "TRT", treatment = "Time3",
  day = "Time3"
)

# For the the median
result <- find_zscore_hit(table, number_barcode = 6)
head(result)

# For the mean
result <- find_zscore_hit(table, number_barcode = 6, metric = "mean")
head(result)
```

---

`get_annotation_table` *Get ScreenR annotation table*

---

**Description**

Get function for the annotation table of the ScreenR object

**Usage**

```
get_annotation_table(object)

## S4 method for signature 'screenr_object'
get_annotation_table(object)
```

**Arguments**

`object` The ScreenR object obtained using the [create\\_screenr\\_object](#)

**Value**

The annotation table of the ScreenR object

**Examples**

```
object <- get0("object", envir = asNamespace("ScreenR"))
annotation_table <- get_annotation_table(object)
head(annotation_table)
```

---

get_count_table	<i>Get ScreenR count table</i>
-----------------	--------------------------------

---

**Description**

Get function for the count table of the ScreenR object

**Usage**

```
get_count_table(object)

## S4 method for signature 'screenr_object'
get_count_table(object)
```

**Arguments**

object            The ScreenR object obtained using the [create\\_screenr\\_object](#)

**Value**

The count table of the ScreenR object

**Slots**

count\_table It is used to store the count table to perform the analysis  
 annotation\_table It is used to store the annotation of the shRNA  
 groups It is used to store the vector of treated and untreated  
 replicates It is used to store information about the replicates  
 normalized\_count\_table It is used to store a normalized version of the count table  
 data\_table It is used to store a tidy format of the count table

**Examples**

```
object <- get0("object", envir = asNamespace("ScreenR"))
count_table <- get_count_table(object)
head(count_table)
data("count_table", package = "ScreenR")
data("annotation_table", package = "ScreenR")

groups <- factor(c(
  "T1/T2", "T1/T2", "Treated", "Treated", "Treated",
  "Control", "Control", "Control", "Treated", "Treated",
  "Treated", "Control", "Control", "Control"
))

obj <- create_screenr_object(
```

```

    table = count_table,
    annotation = annotation_table,
    groups = groups,
    replicates = c("")
  )

```

---

get_data_table	<i>Get ScreenR data_table</i>
----------------	-------------------------------

---

### Description

Get function for the data\_table of the ScreenR object

### Usage

```

get_data_table(object)

## S4 method for signature 'screenr_object'
get_data_table(object)

```

### Arguments

object            The ScreenR object obtained using the [create\\_screenr\\_object](#)

### Value

The data\_table of the ScreenR object

### Examples

```

object <- get0("object", envir = asNamespace("ScreenR"))
data_table <- get_data_table(object)
data_table

```

---

get_groups	<i>Get ScreenR groups</i>
------------	---------------------------

---

### Description

Get function for the groups of the ScreenR object

### Usage

```

get_groups(object)

## S4 method for signature 'screenr_object'
get_groups(object)

```

### Arguments

object            The ScreenR object obtained using the [create\\_screenr\\_object](#)

**Value**

The groups of the ScreenR object

**Examples**

```
object <- get0("object", envir = asNamespace("ScreenR"))
groups <- get_groups(object)
groups
```

---

```
get_normalized_count_table
```

*Get ScreenR normalized\_count\_table*

---

**Description**

Get function for the normalized\_count\_table of the ScreenR object

**Usage**

```
get_normalized_count_table(object)

## S4 method for signature 'screenr_object'
get_normalized_count_table(object)
```

**Arguments**

object            The ScreenR object obtained using the [create\\_screenr\\_object](#)

**Value**

The normalized\_count\_table of the ScreenR object

**Examples**

```
object <- get0("object", envir = asNamespace("ScreenR"))
normalized_count_table <- get_normalized_count_table(object)
normalized_count_table
```

---

```
get_replicates
```

*Get ScreenR replicates*

---

**Description**

Get function for the replicates of the ScreenR object

**Usage**

```
get_replicates(object)

## S4 method for signature 'screenr_object'
get_replicates(object)
```

**Arguments**

object            The ScreenR object obtained using the [create\\_screenr\\_object](#)

**Value**

The replicates of the ScreenR object

**Examples**

```
object <- get0("object", envir = asNamespace("ScreenR"))
replicates <- get_replicates(object)
replicates
```

---

mapped_reads	<i>Mapped Reads</i>
--------------	---------------------

---

**Description**

This function returns the number of mapped reads inside the ScreenR object

**Usage**

```
mapped_reads(screenR_Object)
```

**Arguments**

screenR\_Object   The ScreenR object obtained using the [create\\_screenr\\_object](#)

**Value**

Return a tibble containing the number of mapped read for sample

**Examples**

```
object <- get0("object", envir = asNamespace("ScreenR"))
mapped_reads(object)
```

---

normalize_data	<i>Normalize data</i>
----------------	-----------------------

---

**Description**

This function perform a normalization on the data considering the fact that each shRNA has a defined length so this will not influence the data. Basically is computed the sum for each row and then multiply by 1e6. At the end the data obtained will be Count Per Million.

**Usage**

```
normalize_data(screenR_Object)
```

**Arguments**

screenR\_Object The ScreenR object obtained using the [create\\_screenr\\_object](#)

**Value**

Return the ScreenR object with the normalize data

**Examples**

```
object <- get0("object", envir = asNamespace("ScreenR"))
object <- normalize_data(object)

slot(object, "normalized_count_table")
```

---

plot_barcode_hit	<i>Plot barcode hit</i>
------------------	-------------------------

---

**Description**

Create a barcode plot for a hit. A barcode plot displays if the hit is differentially up or down regulated. If most of the vertical line are on the left side the gene associated to the barcodes is down regulated otherwise is up regulated.

**Usage**

```
plot_barcode_hit(
  screenR_Object,
  matrix_model,
  contrast,
  number_barcode = 3,
  gene,
  quantile = c(-0.5, 0.5),
  labels = c("Negative logFC", "Positive logFC")
)
```

**Arguments**

screenR\_Object The ScreenR object obtained using the [create\\_screenr\\_object](#)

matrix\_model The matrix that will be used to perform the linear model analysis. It is created using `model.matrix`.

contrast An object created with [makeContrasts](#) function.

number\_barcode Number of barcode that as to be differentially expressed (DE) in order to consider the associated gene DE. Example a gene is associated with 10 shRNA we consider a gene DE if it has at least `number_barcode = 5` shRNA DE.

gene The name of the gene that has to be plot

quantile Quantile to display on the plot

labels The label to be displayed on the quantile side

**Value**

The barcode plot

**Examples**

```

object <- get0("object", envir = asNamespace("ScreenR"))
matrix_model <- model.matrix(~ slot(object, "groups"))
colnames(matrix_model) <- c("Control", "T1_T2", "Treated")
contrast <- limma::makeContrasts(Treated - Control, levels = matrix_model)

plot_barcode_hit(object, matrix_model,
  contrast = contrast,
  gene = "Gene_300"
)

```

---

plot_barcode_lost	<i>Plot number of barcode lost</i>
-------------------	------------------------------------

---

**Description**

This function plots the number of barcode lost in each sample. Usually lots of barcodes lost mean that the sample has low quality.

**Usage**

```

plot_barcode_lost(
  screenR_Object,
  palette = NULL,
  alpha = 1,
  legende_position = "none"
)

```

**Arguments**

**screenR\_Object** The ScreenR object obtained using the [create\\_screenr\\_object](#)

**palette** A vector of colors to be used to fill the barplot.

**alpha** A value for the opacity of the plot. Allowed values are in the range 0 to 1

**legende\_position** Where to positioning the legend of the plot. Allowed values are in the "top", "bottom", "right", "left", "none".

**Value**

Returns the plot displaying the number of barcode lost in each sample

**Examples**

```

object <- get0("object", envir = asNamespace("ScreenR"))

plot_barcode_lost(object)

```



---

`plot_barcode_lost_for_gene`*Plot number of barcode lost for gene*

---

### Description

This function plots the number of barcodes lost in each sample for each gene. Usually in a genetic screening each gene is associated with multiple shRNAs and so barcodes. For this reason a reasonable number of barcodes associated with the gene has to be retrieved in order to have a robust result. Visualizing the number of genes that have lost lots of barcode is a Quality Check procedure in order to be aware of the number of barcode for the hit identified.

### Usage

```
plot_barcode_lost_for_gene(screenR_Object, facet = TRUE, samples)
```

### Arguments

`screenR_Object` The ScreenR object obtained using the [create\\_screenr\\_object](#)

`facet` A boolean to use the facet.

`samples` A vector of samples that as to be visualize

### Value

Return the plot displaying the number of barcode lost for each gene in each sample.

### Examples

```
object <- get0("object", envir = asNamespace("ScreenR"))

plot_barcode_lost_for_gene(object,
  samples = c("Time3_A", "Time3_B")
)
plot_barcode_lost_for_gene(object,
  samples = c("Time3_A", "Time3_B"),
  facet = FALSE
)
```

---

`plot_barcode_trend` *Plot the trend over time of the barcodes*

---

### Description

Plot the log2FC over time of the barcodes in the different time point. This plot is useful to check we efficacy of each shRNA. Good shRNAs should have consistent trend trend over time.

**Usage**

```
plot_barcode_trend(
  list_data_measure,
  genes,
  n_col = 1,
  size_line = 1,
  color = NULL
)
```

**Arguments**

list_data_measure	A list containing the measure table of the different time point. Generated using the compute_metrics function.
genes	The vector of genes name.
n_col	The number of column to use in the facet wrap.
size_line	The thickness of the line.
color	The vector of colors. One color for each barcode.

**Value**

The trend plot for the genes in input.

**Examples**

```
object <- get0("object", envir = asNamespace("ScreenR"))

metrics <- dplyr::bind_rows(
  compute_metrics(object,
    control = "TRT", treatment = "Time3",
    day = "Time3"
  ),
  compute_metrics(object,
    control = "TRT", treatment = "Time4",
    day = "Time4"
  )
)
# Multiple Genes
plot_barcode_trend(metrics,
  genes = c("Gene_1", "Gene_50"),
  n_col = 2
)
# Single Gene
plot_barcode_trend(metrics, genes = "Gene_300")
```

---

plot\_boxplot

*Plot Barcodes Hit*

---

**Description**

This function plots a boxplot for each sample for the genes passed as input. It can be used to see the overall trend of a gene and so to visualize if the gene is up or down regulated.

## Usage

```
plot_boxplot(  
  screenR_Object,  
  genes,  
  group_var,  
  alpha = 0.5,  
  nrow = 1,  
  ncol = 1,  
  fill_var = "Sample",  
  type = "boxplot",  
  scales = "free"  
)
```

## Arguments

screenR_Object	The ScreenR object obtained using the <a href="#">create_screenr_object</a>
genes	The vector of genes that will be displayed
group_var	The variable that as to be used to filter the data, for example the different treatment
alpha	A value for the opacity of the plot. Allowed values are in the range 0 to 1
nrow	The number of rows in case multiple genes are plotted
ncol	The number of columns in case multiple genes are plotted
fill_var	The variable used to fill the boxplot
type	The type of plot to use "boxplot" or "violinplot"
scales	The scales used for the facet. Possible values can be "free", "fixed" and "free_y"

## Value

A boxplot

## Examples

```
object <- get0("object", envir = asNamespace("ScreenR"))  
  
plot_boxplot(object,  
  genes = c("Gene_34"),  
  group_var = c("T1", "T2", "TRT"), nrow = 1, ncol = 2,  
  fill_var = "Day", type = "violinplot"  
)
```

---

plot\_common\_hit

*Plot common hit*

---

## Description

This method plot the hits in common among the three methods is a wrapper for the [ggvenn](#) function.

**Usage**

```
plot_common_hit(
  hit_zscore,
  hit_camera,
  hit_roast,
  alpha = 0.5,
  stroke_size = 0.5,
  set_name_size = 4,
  text_color = "black",
  text_size = 4,
  show_percentage = TRUE,
  title = "",
  color = c("#1B9E77", "#D95F02", "#7570B3"),
  show_elements = TRUE
)
```

**Arguments**

hit_zscore	The list of hits of the <a href="#">find_zscore_hit</a>
hit_camera	The list of hits of the <a href="#">find_camera_hit</a>
hit_roast	The list of hits of the <a href="#">find_roast_hit</a>
alpha	A value for the opacity of the plot. Allowed values are in the range 0 to 1
stroke_size	Stroke size for drawing circles
set_name_size	Text size for set names
text_color	Text color for intersect contents
text_size	Text size for intersect contents
show_percentage	Show percentage for each set
title	The title to display above the plot
color	The three vector color for the venn
show_elements	Show set elements instead of count/percentage.

**Value**

A vector containing the common hit

**Examples**

```
hit_zscore <- data.frame(Gene = c("A", "B", "C", "D", "E"))
hit_camera <- data.frame(Gene = c("A", "B", "C", "F", "H", "G"))
hit_roast <- data.frame(Gene = c("A", "L", "N"))
plot_common_hit(hit_zscore, hit_camera, hit_roast)
```

---

`plot_explained_variance`*Plot the explained variance by the PC*

---

**Description**

This function plot the explained variance by the Principal Component analysis.

**Usage**

```
plot_explained_variance(  
  screenR_Object,  
  cumulative = FALSE,  
  color = "steelblue"  
)
```

**Arguments**

`screenR_Object` The ScreenR object obtained using the [create\\_screenr\\_object](#)  
`cumulative` A boolean value which indicates whether or not to plot the cumulative variance.  
The default value is FALSE.  
`color` The color to fill the barplot the default value is steelblue

**Value**

The explained variance plot

**Examples**

```
object <- get0("object", envir = asNamespace("ScreenR"))  
  
plot_explained_variance(object)  
  
# For the cumulative plot  
plot_explained_variance(object, cumulative = TRUE)
```

---

`plot_mapped_reads`*Plot mapped reads*

---

**Description**

This function plots the number of reads mapped for each sample. It internally call the [count\\_mapped\\_reads](#) function, to compute the number of mapped reads.

**Usage**

```
plot_mapped_reads(  
  screenR_Object,  
  palette = NULL,  
  alpha = 1,  
  legende_position = "none"  
)
```

**Arguments**

screenR\_Object The ScreenR object obtained using the [create\\_screenr\\_object](#)  
 palette A vector of color that as to be used to fill the barplot.  
 alpha A value for the opacity of the plot. Allowed values are in the range 0 to 1  
 legende\_position Where to positioning the legend of the plot ("none", "left", "right", "bottom", "top")

**Value**

return a ggplot object

**Examples**

```
object <- get0("object", envir = asNamespace("ScreenR"))
plot_mapped_reads(object)
```

---

plot\_mapped\_reads\_distribution

*Plot the distribution of the mapped reads*

---

**Description**

This function creates a boxplot or a densityplot to show the distribution of the mapped reads in different samples. This function can be used to assess the quality of the samples. Samples which show roughly the same distribution have good quality.

**Usage**

```
plot_mapped_reads_distribution(
  screenR_Object,
  palette = NULL,
  alpha = 1,
  type = "boxplot"
)
```

**Arguments**

screenR\_Object The ScreenR object obtained using the [create\\_screenr\\_object](#).  
 palette The color vector that as to be used for the plot.  
 alpha A value for the opacity of the plot. Allowed values are in the range 0 to 1  
 type The type of plot. The default is "boxplot" the other option is "density."

**Value**

Return a tibble containing the number of mapped read for each sample

**Examples**

```
object <- get0("object", envir = asNamespace("ScreenR"))

# Boxplot
plot_mapped_reads_distribution(object)

# Density
plot_mapped_reads_distribution(object, type = "density")

plot_mapped_reads_distribution(object, type = "density", alpha = 0.2)
```

---

plot\_mds

*Multidimensional Scaling Plot*

---

**Description**

Plot samples on a two-dimensional scatterplot so that distances on the plot approximate the typical log<sub>2</sub> fold changes between the samples.

**Usage**

```
plot_mds(
  screenR_Object,
  groups = NULL,
  alpha = 0.8,
  size = 2.5,
  color = "black"
)
```

**Arguments**

screenR_Object	The Object of the package <a href="#">create_screenr_object</a>
groups	The vector that has to be used to fill the plot if NULL the function will use the default groups slot in the object passed as input.
alpha	The opacity of the labels. Possible value are in a range from 0 to 1.
size	The dimension of the labels. The default value is 2.5
color	The color of the labels. The default value is black

**Value**

The MDS Plot

**Examples**

```
object <- get0("object", envir = asNamespace("ScreenR"))

plot_mds(object)
```

---

 plot\_trend

*Plot the trend hit gene*


---

### Description

This function plot the trend of a gene resulted as hit

### Usage

```
plot_trend(
  screenR_Object,
  genes,
  group_var,
  alpha = 0.5,
  se = FALSE,
  point_size = 1,
  line_size = 1,
  nrow = 1,
  ncol = 1,
  scales = "free"
)
```

### Arguments

screenR_Object	The ScreenR object obtained using the <a href="#">create_screenr_object</a>
genes	The vector of genes to use
group_var	The variable that as to be used to filter the data, for example the different treatment
alpha	A value for the opacity of the plot. Allowed values are in the range 0 to 1
se	A boolean to indicate where or not to plot the standard error
point_size	The dimension of each dot
line_size	The dimension of the line
nrow	The number of rows in case multiple genes are plotted
ncol	The number of columns in case multiple genes are plotted
scales	The scales to be used in the facette

### Value

The plot of the trend over time for a specific treatment.

### Examples

```
object <- get0("object", envir = asNamespace("ScreenR"))

plot_trend(object, genes = "Gene_42", group_var = c("T1", "T2", "TRT"))

plot_trend(object,
  genes = c("Gene_42", "Gene_100"),
  group_var = c("T1", "T2", "TRT"),
  nrow = 2
)
```



---

`plot_zscore_distribution`*Plot distribution Z-score*

---

**Description**

This function plots the Log2FC Z-score distribution of the treated vs control in the different time points.

**Usage**

```
plot_zscore_distribution(time_point_measure, alpha = 1)
```

**Arguments**

`time_point_measure`

A list containing the table for each time point. Each table contains for each barcode the counts for the treated and control the Log2FC, Zscore, ZscoreRobust, Day.

`alpha`

A value for the opacity of the plot. Allowed values are in the range 0 to 1

**Value**

return the density plot of the distribution of the Z-score

**Examples**

```
object <- get0("object", envir = asNamespace("ScreenR"))

table1 <- compute_metrics(object,
  control = "TRT", treatment = "Time3",
  day = "Time3"
)

table2 <- compute_metrics(object,
  control = "TRT", treatment = "Time4",
  day = "Time4"
)

plot_zscore_distribution(list(table1, table2), alpha = 0.5)
```

---

`remove_all_zero_row`*Remove rows that have zero count in all samples*

---

**Description**

This function removes the rows that have zero count in all samples. It takes care of updating both `count_table` and `annotation_table`. **Very Important:** It has to be performed before the data normalization.

**Usage**

```
remove_all_zero_row(screenR_Object)
```

**Arguments**

screenR\_Object The ScreenR object obtained using the [create\\_screenr\\_object](#)

**Value**

The ScreenR object with the count\_table and the annotation\_table filtered.

**Examples**

```
object <- get0("object", envir = asNamespace("ScreenR"))
counts <- get_count_table(object)
nrow(counts)
object <- remove_all_zero_row(object)
counts <- get_count_table(object)
nrow(counts)
```

---

select\_number\_barcode *Select number of Barcode*

---

**Description**

Compute a unique gene symbol for gene

**Usage**

```
select_number_barcode(gene, gene_symbols, number_barcode)
```

**Arguments**

gene The gene name  
... Arguments passed on to [unique\\_gene\\_symbols](#)  
gene\_symbols The gene symbols list

**Value**

The barcode of the gene passed as input

---

set\_annotation\_table    *Set ScreenR annotation table*

---

**Description**

Set function for the annotation table of the ScreenR object

**Usage**

```
set_annotation_table(object, annotation_table)
```

```
## S4 method for signature 'screenr_object'  
set_annotation_table(object, annotation_table)
```

**Arguments**

object            The ScreenR object obtained using the [create\\_screenr\\_object](#)  
annotation\_table            a table containing the annotation for each shRNA

**Value**

The ScreenR object with the annotation table

**Examples**

```
object <- get0("object", envir = asNamespace("ScreenR"))  
annotation <- get_annotation_table(object)  
set_annotation_table(object, annotation)
```

---

set\_count\_table            *Set ScreenR count table*

---

**Description**

Set function for the count table of the ScreenR object

**Usage**

```
set_count_table(object, count_table)
```

```
## S4 method for signature 'screenr_object'  
set_count_table(object, count_table)
```

**Arguments**

object            The ScreenR object obtained using the [create\\_screenr\\_object](#)  
count\_table        A count table containing in each row an shRNA and in each column a sample

**Value**

The ScreenR object with the count table

**Examples**

```
object <- get0("object", envir = asNamespace("ScreenR"))
counts <- get_count_table(object)
set_count_table(object, counts)
```

---

set_data_table	<i>Set ScreenR data_table</i>
----------------	-------------------------------

---

**Description**

Set function for the data\_table of the ScreenR object

**Usage**

```
set_data_table(object, data_table)

## S4 method for signature 'screenr_object'
set_data_table(object, data_table)
```

**Arguments**

object	The ScreenR object obtained using the <a href="#">create_screenr_object</a>
data_table	A count table in a tidy format

**Value**

The ScreenR object with the set data\_table

**Examples**

```
object <- get0("object", envir = asNamespace("ScreenR"))
data_table <- get_data_table(object)
set_data_table(object, data_table)
```

---

set_groups	<i>Set ScreenR groups</i>
------------	---------------------------

---

**Description**

Set function for the groups of the ScreenR object

**Usage**

```
set_groups(object, groups)

## S4 method for signature 'screenr_object'
set_groups(object, groups)
```

**Arguments**

object            The ScreenR object obtained using the [create\\_screenr\\_object](#)  
groups            The treatment and control groups

**Value**

The ScreenR object containing the group field

**Examples**

```
object <- get0("object", envir = asNamespace("ScreenR"))  
groups <- get_groups(object)  
set_groups(object, groups)
```

---

set\_normalized\_count\_table  
*Set ScreenR normalized\_count\_table*

---

**Description**

Set function for the normalized\_count\_table of the ScreenR object

**Usage**

```
set_normalized_count_table(object, normalized_count_table)  
  
## S4 method for signature 'screenr_object'  
set_normalized_count_table(object, normalized_count_table)
```

**Arguments**

object            The ScreenR object obtained using the [create\\_screenr\\_object](#)  
normalized\_count\_table  
                  A table of the normalized count table

**Value**

The ScreenR object with the set normalized\_count\_table

**Examples**

```
object <- get0("object", envir = asNamespace("ScreenR"))  
normalized_count_table <- get_normalized_count_table(object)  
normalized_count_table  
set_normalized_count_table(object, normalized_count_table)
```

---

set\_replicates      *Set ScreenR replicates*

---

**Description**

Set function for the replicates of the ScreenR object

**Usage**

```
set_replicates(object, replicates)

## S4 method for signature 'screenr_object'
set_replicates(object, replicates)
```

**Arguments**

object              The ScreenR object obtained using the [create\\_screenr\\_object](#)  
 replicates         The vector containing the replicates name

**Value**

The ScreenR object with the specific replicates

**Examples**

```
object <- get0("object", envir = asNamespace("ScreenR"))
replicates <- get_replicates(object)
set_replicates(object, replicates)
```

---

unique\_gene\_symbols      *Unique gene Symbols*

---

**Description**

Compute a unique gene symbol for gene

**Usage**

```
unique_gene_symbols(gene_symbols, number_barcode = 3)
```

**Arguments**

gene\_symbols        The gene symbols list  
 ...                  Arguments passed on to [find\\_camera\\_hit](#)  
 number\_barcode     Number of barcode that as to be differentially expressed (DE)in  
                       order to consider the gene associated DE. Example a gene is associated  
                       with 10 shRNA we consider a gene DE if it has at least number\_barcode =  
                       5 shRNA DE.

**Value**

A list of unique gene symbols

# Index

- \* **compute**
  - barcode\_lost, 4
  - compute\_data\_table, 6
  - compute\_metrics, 7
  - compute\_slope, 7
  - count\_mapped\_reads, 9
  - mapped\_reads, 22
  - normalize\_data, 22
  - remove\_all\_zero\_row, 33
- \* **datasets**
  - annotation\_table, 4
  - count\_table, 9
- \* **data**
  - annotation\_table, 4
  - count\_table, 9
- \* **filter**
  - filter\_by\_slope, 12
  - filter\_by\_variance, 13
- \* **find**
  - find\_camera\_hit, 14
  - find\_common\_hit, 15
  - find\_roast\_hit, 16
  - find\_robust\_zscore\_hit, 17
  - find\_zscore\_hit, 17
- \* **internal**
  - compute\_camera, 5
  - compute\_explained\_variance, 6
  - compute\_trend, 8
  - ScreenR-package, 3
  - select\_number\_barcode, 34
  - unique\_gene\_symbols, 38
- \* **objects**
  - create\_edger\_obj, 10
  - create\_screenr\_object, 11
  - get\_annotation\_table, 18
  - get\_count\_table, 19
  - get\_data\_table, 20
  - get\_groups, 20
  - get\_normalized\_count\_table, 21
  - get\_replicates, 21
  - set\_annotation\_table, 35
  - set\_count\_table, 35
  - set\_data\_table, 36
  - set\_groups, 36
  - set\_normalized\_count\_table, 37
  - set\_replicates, 38
- \* **plot**
  - plot\_barcode\_hit, 23
  - plot\_barcode\_lost, 24
  - plot\_barcode\_lost\_for\_gene, 25
  - plot\_boxplot, 26
  - plot\_mapped\_reads\_distribution, 30
- annotation\_table, 4
- barcode\_lost, 4
- camera, 14
- compute\_camera, 5
- compute\_data\_table, 6
- compute\_explained\_variance, 6
- compute\_metrics, 7
- compute\_slope, 7
- compute\_trend, 8
- count\_mapped\_reads, 9, 29
- count\_table, 9
- create\_edger\_obj, 10
- create\_screenr\_object, 4, 6–9, 11, 11, 12–14, 16, 18–25, 27, 29–32, 34–38
- estimateDisp, 5, 8
- filter\_by\_slope, 12
- filter\_by\_variance, 13
- find\_camera\_hit, 5, 14, 15, 28, 38
- find\_common\_hit, 15
- find\_roast\_hit, 15, 16, 28
- find\_robust\_zscore\_hit, 17
- find\_zscore\_hit, 7, 15, 17, 28
- get\_annotation\_table, 18
- get\_annotation\_table, screenr\_object (get\_annotation\_table), 18
- get\_annotation\_table, screenr\_object-method (get\_annotation\_table), 18
- get\_count\_table, 19
- get\_count\_table, screenr\_object (get\_count\_table), 19

- get\_count\_table, screenr\_object-method  
(get\_count\_table), 19
- get\_data\_table, 20
- get\_data\_table, screenr\_object  
(get\_data\_table), 20
- get\_data\_table, screenr\_object-method  
(get\_data\_table), 20
- get\_groups, 20
- get\_groups, screenr\_object (get\_groups),  
20
- get\_groups, screenr\_object-method  
(get\_groups), 20
- get\_normalized\_count\_table, 21
- get\_normalized\_count\_table, screenr\_object  
(get\_normalized\_count\_table),  
21
- get\_normalized\_count\_table, screenr\_object-method  
(get\_normalized\_count\_table),  
21
- get\_replicates, 21
- get\_replicates, screenr\_object  
(get\_replicates), 21
- get\_replicates, screenr\_object-method  
(get\_replicates), 21
- ggvenn, 27
- glmFit, 5
- makeContrasts, 23
- mapped\_reads, 22
- model.matrix, 5, 13, 14, 16
- normalize\_data, 22
- plot\_barcode\_hit, 23
- plot\_barcode\_lost, 24
- plot\_barcode\_lost\_for\_gene, 25
- plot\_barcode\_trend, 25
- plot\_boxplot, 26
- plot\_common\_hit, 27
- plot\_explained\_variance, 29
- plot\_mapped\_reads, 29
- plot\_mapped\_reads\_distribution, 30
- plot\_mds, 31
- plot\_trend, 32
- plot\_zscore\_distribution, 33
- remove\_all\_zero\_row, 33
- ScreenR (ScreenR-package), 3
- ScreenR-package, 3
- screenr\_object (get\_count\_table), 19
- screenr\_object-class (get\_count\_table),  
19
- select\_number\_barcode, 34
- set\_annotation\_table, 35
- set\_annotation\_table, screenr\_object  
(set\_annotation\_table), 35
- set\_annotation\_table, screenr\_object-method  
(set\_annotation\_table), 35
- set\_count\_table, 35
- set\_count\_table, screenr\_object  
(set\_count\_table), 35
- set\_count\_table, screenr\_object-method  
(set\_count\_table), 35
- set\_data\_table, 36
- set\_data\_table, screenr\_object  
(set\_data\_table), 36
- set\_data\_table, screenr\_object-method  
(set\_data\_table), 36
- set\_groups, 36
- set\_groups, screenr\_object (set\_groups),  
36
- set\_groups, screenr\_object-method  
(set\_groups), 36
- set\_normalized\_count\_table, 37
- set\_normalized\_count\_table, screenr\_object  
(set\_normalized\_count\_table),  
37
- set\_normalized\_count\_table, screenr\_object-method  
(set\_normalized\_count\_table),  
37
- set\_replicates, 38
- set\_replicates, screenr\_object  
(set\_replicates), 38
- set\_replicates, screenr\_object-method  
(set\_replicates), 38
- unique\_gene\_symbols, 34, 38