Package 'gimap'

February 5, 2026

Type Package

Title Calculate Genetic Interactions for Paired CRISPR Targets

Version 1.1.2

Author Candace Savonen [aut, cre],
Phoebe Parrish [aut],
Kate Isaac [aut],
Howard Baek [aut],
Daniel Grosso [aut],
Siobhan O'Brien [aut],
Alice Berger [aut]

Maintainer Candace Savonen <cansav09@gmail.com>

Description Helps find meaningful patterns in complex genetic experiments. First gimap takes data from paired CRISPR (Clustered regularly interspaced short palindromic repeats) screens that has been pre-processed to counts table of paired gRNA (guide Ribonucleic Acid) reads. The input data will have cell counts for how well cells grow (or don't grow) when different genes or pairs of genes are disabled. The output of the 'gimap' package is genetic interaction scores which are the distance between the observed CRISPR score and the expected CRISPR score. The expected CRISPR scores are what we expect for the CRISPR values to be for two unrelated genes. The further away an observed CRISPR score is from its expected score the more we suspect genetic interaction. The work in this package is based off of original research from the Alice Berger lab at Fred Hutchinson Cancer Center (2021) <doi:10.1016/j.celrep.2021.109597>.

License GPL-3

URL https://github.com/FredHutch/gimap

BugReports https://github.com/FredHutch/gimap/issues

Imports readr, dplyr, tidyr, rmarkdown, vroom, ggplot2, magrittr, pheatmap, purrr, janitor, utils, stats, stringr, httr, jsonlite, openssl

Suggests testthat (>= 3.0.0), roxygen2, kableExtra, knitr

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 7.3.2

VignetteBuilder knitr

NeedsCompilation no

Depends R (>= 3.5.0)

Repository CRAN

Date/Publication 2026-02-05 22:10:02 UTC

Contents

calc_gi *Calculate Genetic Interaction scores*

Description

Create results table that has CRISPR scores, Wilcoxon rank-sum test and t tests. The output of the ‘gimap‘ package is genetic interaction scores which _is the distance between the observed CRISPR score and the expected CRISPR score._ The expected CRISPR scores are what we expect for the CRISPR values should two genes be unrelated to each other. The further away an observed CRISPR scoreis from its expected the more we suspect genetic interaction. This can be true in a positive way (a CRISPR knockout pair caused more cell proliferation than expected) or in a negative way (a CRISPR knockout pair caused more cell lethality than expected).

The genetic interaction scores are based on a linear model calculated for each sample where ‘observed_crispr_single‘ is the outcome variable and ‘expected_crispr_single‘ is the predictor variable. For each sample: lm(observed_crispr_single ~ expected_crispr_single)

Using ‘y = mx+b‘, we can fill in the following values: * ‘y‘ = observed CRISPR score * ‘x‘ = expected CRISPR score * ‘m‘ = slope from linear model for this sample * ‘b‘ = intercept from linear model for this sample

The intercept and slope from this linear model are used to adjust the CRISPR scores for each sample: single target gi score = observed single crispr - (intercept + slope * expected single crispr) double_target_gi_score = double crispr score - (intercept + slope * expected double crispr) These single and double target genetic interaction scores are calculated at the construct level and are then summarized using a t-test to see if the the distribution of the set of double targeting constructs is significantly different than the overall distribution single targeting constructs. After multiple testing correction, FDR values are reported. Low FDR value for a double construct means high suspicion of paralogs.

Usage

```
calc_gi(.data = NULL, gimap_dataset, use_lfc = FALSE, use_ntc = TRUE)
```

Arguments

.data	Data can be piped in with tidyverse pipes from function to function. But the data must still be a gimap_dataset
gimap_dataset	A special dataset structure that is setup using the ‘setup_data()‘ function.
use_lfc	Should Log fold change be used to calculate GI scores instead of CRISPR scores? If you do not have negative controls or CRISPR scores you will need to set this to TRUE.
use_ntc	If you do not have negative controls or do not want them to be used set this to FALSE and then 0 will be used as the comparison value. Default is that use_ntc = TRUE and looks for negative control genes.

Value

A gimap dataset with statistics and genetic interaction scores calculated. Overall results in the returned object can be obtained using gimap_dataset$overall_results Whereas target level genetic interaction scores can be retrieved using ‘gimap_dataset$gi_scores‘.

Examples

```
## Not run:

gimap_dataset <- get_example_data("gimap",
  data_dir = tempdir()
) %>%
  gimap_filter() %>%
  gimap_annotate(
    cell_line = "HELA",
    annot_dir = tempdir()
  ) %>%
  gimap_normalize(
    timepoints = "day",
    missing_ids_file = tempfile()
  ) %>%
  calc_gi()

saveRDS(gimap_dataset, file.path(tempdir(), "gimap_dataset_final.RDS"))

## End(Not run)
```

---

cn_setup *Download and set up DepMap CN*

Description

This function sets up the tpm data from DepMap is called by the ‘gimap_annotate()‘ function if the cn_annotate = TRUE

Usage

```
cn_setup(
  overwrite = TRUE,
  data_dir = gimap_data_dir()
)
```

Arguments

overwrite	Should the files be redownloaded?
data_dir	What directory should this be saved to? Default is a user data directory.

ctrl_genes *Download and set up control genes*

Description

This function sets up the control genes file from DepMap is called by the ‘gimap_annotate()‘

Usage

```
ctrl_genes(
  overwrite = TRUE,
  data_dir = gimap_data_dir()
)
```

Arguments

overwrite	Should the file be redownloaded and reset up?
data_dir	What directory should this be saved to? Default is a user data directory.

delete_annotation *Refresh the annotation files by redownloading them*

Description

This function will set annotation file options to NULL so files will be re-downloaded

Usage

```
delete_annotation()
```

Value

options for tpm_file, cn_file, and ctrl_genes_file are set to NULL.

Examples

```
delete_annotation()
```

delete_example_data *Refresh the example data files by redownloading them*

Description

This function will set example data file options to NULL so files will be re-downloaded

Usage

```
delete_example_data()
```

Value

options for example data are are set to NULL.

Examples

```
delete_example_data()
```

encrypt_creds_path *Default creds path*

Description

Default creds path

Usage

```
encrypt_creds_path(app_name)
```

Arguments

app_name	What app set up are you looking for? Supported apps are 'google' 'calendly' and 'github'

example_data_folder *Get file path to an default credentials RDS*

Description

Get file path to an default credentials RDS

Usage

```
example_data_folder(data_dir = NULL)
```

Arguments

data_dir	Optional data directory override.

Value

Returns the file path to folder where the example data is stored

get_example_data *Returns example data for package*

Description

This function loads and returns example data for the package. Which dataset is returned must be specified. Data will be downloaded from Figshare the first time it is used.

Usage

```
get_example_data(
  which_data,
  data_dir = NULL,
  refresh_data = FALSE
)
```

Arguments

which_data	options are "count" or "meta"; specifies which example dataset should be returned
data_dir	Where should the data be saved if applicable? If NULL, a user data directory will be used.
refresh_data	should the example data that's been downloaded be deleted and redownloaded?

Value

the respective example data either as a data frame or a specialized gimap_dataset depending on what was requested.

Examples

```
## Not run:

counts_timepoint <- get_example_data("count")
counts_treatment <- get_example_data("count_treatment")
gimap_timepoint_dataset <- get_example_data("gimap")
gimap_treatment_dataset <- get_example_data("gimap_treatment")
metadata <- get_example_data("meta")
annotation <- get_example_data("annotation")

## End(Not run)
```

---

get_figshare *Handler function for GET requests from Figshare*

Description

Handler function for GET requests from Figshare

Usage

```
get_figshare(
  file_name = NA,
  item = "19700056",
  output_dir = tempdir(),
  return_list = FALSE
)
```

Arguments

file_name	Which item are we downloading?
item	What is the item we are retrieving?
output_dir	Where should the file be saved?
return_list	Should the list of files be returned instead of the file

Value

Downloads necessary annotation files from Figshare and reads them in as data frames.

Examples

```
## Not run:

get_figshare(
  return_list = TRUE,
  output_dir = tempdir()
)

get_figshare(
  file_name = "Achilles_common_essentials.csv",
  output_dir = tempdir()
)

## End(Not run)
```

`gimap_annotate` *Annotate gimap data*

Description

In this function, a ‘gimap_dataset‘ is annotated as far as which genes should be used as controls.

Usage

```
gimap_annotate(
  .data = NULL,
  gimap_dataset,
  annotation_file = NULL,
  control_genes = NULL,
  cell_line_annotate = TRUE,
  custom_tpm = NULL,
  cell_line = NULL,
  annot_dir = gimap_data_dir(),
  refresh_annot = FALSE
)
```

Arguments

`.data`	Data can be piped in with tidyverse pipes from function to function. But the data must still be a gimap_dataset
`gimap_dataset`	A special dataset structure that is setup using the ‘setup_data()‘ function.
`annotation_file`	If no file is given, will attempt to use the design file from https://media.addgene.org/cms/filer_public/a9/9a a99a9328-324b-42ff-8ccc-30c544b899e4/pgrna_library.xlsx
`control_genes`	A vector of gene symbols (e.g. AAMP) that should be labeled as control genes. These will be used for log fold change calculations. If no list is given then DepMap Public 23Q4 Achilles_common_essentials.csv is used https://depmap.org/portal/download/all/

`cell_line_annotate`
: (Optional) TRUE or FALSE you'd also like to have cell_line_annotation from DepMap.

`custom_tpm`
: (Optional) You may supply your own data frame of transcript per million expression to be used for this calculation if you can't or don't want to use DepMap data annotation for your cell_line. This data frame needs to have two columns: 'log2_tpm' that has the log2 tpm expression data for this cell line and and 'genes' which needs to be gene symbols that match those in the data. eg. "NDL1". Note that you can use custom_tpm with cell_line_annotate but your custom_tpm will be used instead of the tpm data from DepMap. However other data from DepMap like CN will be added.

`cell_line`
: which cell line are you using? (e.g., HELA, PC9, etc.). Required argument if cell_line_annotate is TRUE.

`annot_dir`
: Where should the annotation files be saved? Default is a user data directory.

`refresh_annot`
: Should the annotation file paths saved as options be reset and the files redownloaded? TRUE or FALSE.

Value

A gimap_dataset with annotation data frame that can be retrieve by using gimap_dataset$annotation. This will contain information about your included genes in the set.

Examples

```
## Not run:

# By default DepMap annotation will be used to determine genes which are
# unexpressed. In the `gimap_normalize` this will by default be used to
# normalize to.
gimap_dataset <- get_example_data("gimap",
  data_dir = tempdir()
) %>%
  gimap_filter() %>%
  gimap_annotate(
    cell_line = "HELA",
    missing_ids_file =  tempfile(),
    annot_dir = tempdir()
  )


# You can also say cell_line_annotate = false if you don't want to use DepMap
# annotation BUT if you don't also specify that you say you are
# `normalize_by_unexpressed = FALSE` in the normalize step you will get a
# warning.
gimap_dataset <- get_example_data("gimap",
  data_dir = tempdir()
) %>%
  gimap_filter() %>%
  gimap_annotate(
    cell_line_annotate = FALSE,
```

```
    annot_dir = tempdir()
  ) %>%
  gimap_normalize(
    timepoints = "day",
    normalize_by_unexpressed = FALSE,
    missing_ids_file = tempfile()
  )

### CUSTOM TPM example
# Lastly, this is also an option:
# where custom data is provided to `custom_tpm` is a data frame with
# `genes` and `log2_tpm` as the columns.
gimap_dataset <- get_example_data("gimap",
  data_dir = tempdir()
) %>%
  gimap_filter() %>%
  gimap_annotate(
    cell_line = "HELA",
    custom_tpm = custom_tpm,
    annot_dir = tempdir()
  ) %>%
  gimap_normalize(
    timepoints = "day",
    missing_ids_file = tempfile()
  )

## End(Not run)
```

`gimap_filter` *A function to run filtering*

Description

This function applies filters to the gimap data. By default it runs both the zero count (across all samples) and the low plasmid cpm filters, but users can select a subset of these filters or even adjust the behavior of each filter

Usage

```
gimap_filter(
  .data = NULL,
  gimap_dataset,
  filter_type = "both",
  cutoff = NULL,
  filter_zerocount_target_col = NULL,
  filter_plasmid_target_col = NULL,
  filter_replicates_target_col = NULL,
  min_n_filters = 1
)
```

Arguments

.data	Data can be piped in with tidyverse pipes from function to function. But the data must still be a gimap_dataset
gimap_dataset	A special dataset structure that is setup using the 'setup_data()' function.
filter_type	Can be one of the following: 'zero_count_only', 'low_plasmid_cpm_only' or 'both'. Potentially in the future also 'rep_variation', 'zero_in_last_time_point' or a vector that includes multiple of these filters.
cutoff	default is NULL, relates to the low_plasmid_cpm filter; the cutoff for low log2 CPM values for the plasmid time period; if not specified, The lower outlier (defined by taking the difference of the lower quartile and 1.5 * interquartile range) is used
filter_zerocount_target_col	default is NULL; Which sample column(s) should be used to check for counts of 0? If NULL and not specified, downstream analysis will select all sample columns
filter_plasmid_target_col	default is NULL, and if NULL, will select the first column only; this parameter specifically should be used to specify the plasmid column(s) that will be selected
filter_replicates_target_col	default is NULL, Which sample columns are the final time point replicates; If NULL, the last 3 sample columns are used. This is only used by this function to save a list of which pgRNA IDs have a zero count for all of these samples.
min_n_filters	default is 1; this parameter defines at least how many/the minimum number of independent filters have to flag a pgRNA construct before the construct is filtered when using a combination of filters You should decide on the appropriate filter based on the results of your QC report.

Value

a filtered version of the gimap_dataset returned in the $filtered_data section filter_step_run is a boolean reporting if the filter step was run or not (since it's optional) metadata_pg_ids is a subset the pgRNA IDs such that these are the ones that remain in the dataset following completion of filtering transformed_log2_cpm is a subset the log2_cpm data such that these are the ones that remain in the dataset following completion of filtering removed_pg_ids is a record of which pgRNAs are filtered out once filtering is complete all_reps_zerocount_ids is not actually filtered data necessarily. Instead it's just a record of which pgRNAs have a zero count in all final timepoint replicates

Examples

```
## Not run:

gimap_dataset <- get_example_data("gimap", data_dir = tempdir()) %>%
  gimap_filter()

# To see filtered data
# gimap_dataset$filtered_data
```

```
# If you want to only use a single filter or some subset,
# specify which using the filter_type parameter
gimap_dataset <- get_example_data("gimap") %>%
  gimap_filter(filter_type = "zero_count_only")
# or
gimap_dataset <- get_example_data("gimap") %>%
  gimap_filter(filter_type = "low_plasmid_cpm_only")

# If you want to use multiple filters and more than one to flag a pgRNA
# construct before it's filtered out, use the `min_n_filters` argument
gimap_dataset <- get_example_data("gimap") %>%
  gimap_filter(
    filter_type = "both",
    min_n_filters = 2
  )

# You can also specify which columns the filters will be applied to
gimap_dataset <- get_example_data("gimap") %>%
  gimap_filter(
    filter_type = "zero_count_only",
    filter_zerocount_target_col = c(1, 2)
  )

## End(Not run)
```

`gimap_normalize` *Normalize Log fold changes*

Description

This calculates the log fold change for a gimap dataset based on the annotation and metadata provided. gimap takes in a counts matrix that represents the number of cells that have each type of pgRNA this data needs some normalization before CRISPR scores and Genetic Interaction scores can be calculated.

There are three steps of normalization. 1. 'Calculate log2CPM' - First we account for different read depths across samples and transforms data to log2 counts per million reads. 'log2((counts / total counts for sample)) * 1 million) + 1)' 2. 'Calculate log2 fold change' - This is done by subtracting the log2CPM for the pre-treatment from each sample. control is what is highlighted. The pretreatment is the day 0 of CRISPR treatment, before CRISPR pgRNAs have taken effect. 'log2FC = log2CPM for each sample - pretreament log2CPM'

3. 'Normalize by negative and positive controls' - Calculate a negative control median for each sample and a positive control median for each sample and divide each log2FC by this value. log2FC adjusted = log2FC / (median negative control for a sample - median positive control for a sample)

Usage

```
gimap_normalize(
  .data = NULL,
```

```
  gimap_dataset,
  normalize_by_unexpressed = TRUE,
  timepoints = NULL,
  treatments = NULL,
  control_name = NULL,
  adj_method = "negative_control_adj",
  num_ids_wo_annot = 20,
  rm_ids_wo_annot = TRUE,
  missing_ids_file = "missing_ids_file.csv",
  overwrite = TRUE
)
```

Arguments

.data	Data can be piped in with a tidyverse pipe from function to function. But the data must still be a gimap_dataset
gimap_dataset	A special dataset structure that is setup using the ‘setup_data()‘ function.
normalize_by_unexpressed	TRUE/FALSE crispr data should be normalized so that the median of unexpressed controls is 0. For this to happen set this to TRUE but you need to have added TPM data in the gimap_annotate step using cell_line_annotation or custom_tpm.
timepoints	Specifies the column name of the metadata set up in ‘$metadata$sample_metadata‘ that has a factor that represents the timepoints. Timepoints will be made into three categories: plasmid for the earliest time point, early for all middle timepoints and late for the latest timepoints. The late timepoints will be the focus for the calculations. The column used for timepoints must be numeric or at least ordinal.
treatments	Specifies the column name of the metadata set up in ‘$metadata$sample_metadata‘ that has a factor that represents column that specifies the treatment applied to each. The replicates will be kept collapsed to an average.
control_name	A name that specifies the data either in the treatments column that should be used as the control. This could be the Day 0 of treatment or an untreated sample. For timepoints testing it will be assumed that the mininmum timepoint is the control.
adj_method	Must be one of three methods as stated by a character string "negative_control_adj" or "no_adjustment". Default is "negative_control_adj" "negative_control_adj" where CRISPR scores will be used for the GI scores "no_adjustment" is where LFC adjusted will be used for the GI scores
num_ids_wo_annot	default is 20; the number of pgRNA IDs to display to console if they don’t have corresponding annotation data; ff there are more IDs without annotation data than this number, the output will be sent to a file rather than the console.
rm_ids_wo_annot	default is TRUE; whether or not to filter out pgRNA IDs from the input dataset that don’t have corresponding annotation data available

`missing_ids_file`
: If there are missing IDs and a file is saved, where do you want this file to be saved? Provide a file path.

`overwrite`
: Should existing normalized_log_fc data in the gimap_dataset be overwritten?

Value

A gimap_dataset with normalized log FC as a data frame that can be retrieve by using gimap_dataset$normalized_log_fc. This will contain log2FC adjusted stored in a column named ‘log_adj‘ and the CRISPR scores stored in a column named ‘crispr_score‘. genes in the set.

Examples

```
## Not run:

gimap_dataset_org <- get_example_data("gimap") %>%
  gimap_filter() %>%
  gimap_annotate(cell_line = "HELA") %>%
  gimap_normalize(
    timepoints = "day",
    missing_ids_file = tempfile()
  )

## End(Not run)
```

---

gimap_object *Make an empty gimap dataset object*

Description

This function makes an empty gimap data object

Usage

```
gimap_object()
```

Value

an empty ‘gimap_dataset‘ which is a named list which will be filled by various ‘gimap‘ functions.

gimap_stats *Do tests –an internal function used by calc_gi() function*

Description

Create results table that has t test p values

Usage

```
gimap_stats(gi_calc_double, gi_calc_single, use_ntc = TRUE, replicate = NULL)
```

Arguments

gi_calc_double	a data.frame with adjusted double gi scores
gi_calc_single	a data.frame with adjusted single gi scores
use_ntc	Logical. If TRUE, uses a two-sample t-test comparing double target GI scores against single target GI scores. If FALSE, uses a one-sample t-test against mu=0 since single target GI scores have zero variance when negative controls are not used.
replicate	a name of a replicate to filter out from gi_calc_adj Optional

key_encrypt_creds_path
Get file path to an key encryption RDS

Description

Get file path to an key encryption RDS

Usage

```
key_encrypt_creds_path()
```

plot_crispr *Plot CRISPR scores after normalization*

Description

This plots normalization after CRISPR scores have been calculated

Usage

```
plot_crispr(.data = NULL, gimap_dataset, output_file = "crispr_norm_plot.png")
```

Arguments

.data	Data can be piped in with tidyverse pipes from function to function. But the data must still be a gimap_dataset
gimap_dataset	A special dataset structure that is setup using the 'setup_data()' function.
output_file	A file for the output

Value

A ggplot2 boxplot of the CRISPR scores separated by the type of target. Can be used to determine the normalization has proceeded properly.

Examples

```
## Not run:

gimap_dataset <- get_example_data("gimap") %>%
  gimap_filter() %>%
  gimap_annotate(cell_line = "HELA") %>%
  gimap_normalize(
    timepoints = "day",
    missing_ids_file = tempfile()
  )

# Plot:
plot_crispr(gimap_dataset)

## End(Not run)
```

plot_exp_v_obs_scatter

Expected vs Observed CRISPR Scatterplot

Description

This plot is meant to be functionally equivalent to Fig S5K (for HeLa, equivalent of Fig 3a for PC9). Scatter plot of target-level observed versus expected CRISPR scores in the screen. The solid line is the linear regression line for the negative control (single KO) pgRNAs, while dashed lines indicate the lower and upper quartile residuals.

Usage

```
plot_exp_v_obs_scatter(gimap_dataset, facet_rep = FALSE)
```

Arguments

gimap_dataset	A special dataset structure that is originally setup using ‘setup_data()‘ and has had gi scores calculated with ‘calc_gi()‘.
facet_rep	Should the replicates be wrapped with facet_wrap()?

Value

A ggplot2 scatterplot of the target level observed vs expected CRISPR scores.

Examples

```
## Not run:

gimap_dataset <- get_example_data("gimap") %>%
  gimap_filter() %>%
  gimap_annotate(cell_line = "HELA") %>%
  gimap_normalize(
    timepoints = "day",
    missing_ids_file = tempfile()
  ) %>%
  calc_gi()

# To plot results
plot_exp_v_obs_scatter(gimap_dataset)
plot_rank_scatter(gimap_dataset)
plot_volcano(gimap_dataset)

## End(Not run)
```

`plot_rank_scatter` *Rank plot for target-level GI scores*

Description

This plot is meant to be functionally equivalent to Fig 5a (for HeLa, equivalent of Fig 3c for PC9). Rank plot of target-level GI scores. Dashed horizontal lines are for GI scores of 0.25 and -0.5

Usage

```
plot_rank_scatter(gimap_dataset, reps_to_drop = "")
```

Arguments

`gimap_dataset`	A special dataset structure that is originally setup using 'setup_data()' and has had gi scores calculated with 'calc_gi()'.
`reps_to_drop`	Names of replicates that should be not plotted (Optional)

Value

A ggplot2 rankplot of the target level genetic interaction scores.

Examples

```
## Not run:

gimap_dataset <- get_example_data("gimap") %>%
  gimap_filter() %>%
  gimap_annotate(cell_line = "HELA") %>%
  gimap_normalize(
    timepoints = "day"
  ) %>%
  calc_gi()

# To plot results
plot_exp_v_obs_scatter(gimap_dataset)
plot_rank_scatter(gimap_dataset)
plot_volcano(gimap_dataset)

## End(Not run)
```

plot_targets *Target bar plot for CRISPR scores*

Description

This plot is for when you'd like to examine a target pair specifically – meant to be functionally equivalent to Fig 3b CRISPR scores for representative synthetic lethal paralog pairs. Data shown are the mean CRISPR score for each single KO or DKO target across three biological replicates with replicate data shown in overlaid points.

Usage

```
plot_targets(gimap_dataset, target1, target2, reps_to_drop = "")
```

Arguments

gimap_dataset	A special dataset structure that is originally setup using 'setup_data()' and has had gi scores calculated with 'calc_gi()'.
target1	Name of the first target to be plotted e.g.
target2	Name of the second target to be plotted e.g.
reps_to_drop	Names of replicates that should be not plotted (Optional)

Value

A ggplot2 bar plot of the specific target's genetic interaction scores.

Examples

```
## Not run:

gimap_dataset <- get_example_data("gimap") %>%
  gimap_filter() %>%
  gimap_annotate(cell_line = "HELA") %>%
  gimap_normalize(
    timepoints = "day"
  ) %>%
  calc_gi()

# To plot results, pick out two targets from the gi_score table
head(dplyr::arrange(gimap_dataset$gi_score, fdr))

# "TIAL1_TIA1" is top result so let's plot that
plot_targets(gimap_dataset, target1 = "TIAL1", target2 = "TIA1")

## End(Not run)
```

plot_theme *Standardized plot theme*

Description

this is a ggplot2 standardized plot theme for this package

Usage

```
plot_theme()
```

Value

A ggplot2 theme that can be used on the plots.

plot_volcano *Volcano plot for GI scores*

Description

This plot is meant to be functionally equivalent to Fig 5b (for HeLa, equivalent of Fig 3d for PC9). Volcano plot of target-level GI scores Blue points are synthetic lethal paralog GIs with GI < 0.5 and FDR < 0.1; red points are buffering paralog GIs with GI > 0.25 and FDR < 0.1.

Usage

```
plot_volcano(gimap_dataset, facet_rep = FALSE)
```

Arguments

gimap_dataset	A special dataset structure that is originally setup using ‘setup_data()‘ and has had gi scores calculated with ‘calc_gi()‘.
facet_rep	Should the replicates be wrapped with facet_wrap()?

Value

A ggplot2 volcano plot of the target level genetic interaction scores.

Examples

```
## Not run:

gimap_dataset <- get_example_data("gimap") %>%
  gimap_filter() %>%
  gimap_annotate(cell_line = "HELA") %>%
  gimap_normalize(
    timepoints = "day"
  ) %>%
  calc_gi()

# To plot results
plot_exp_v_obs_scatter(gimap_dataset)
plot_rank_scatter(gimap_dataset)
plot_volcano(gimap_dataset)

## End(Not run)
```

---

qc_cdf *Create a CDF for the pgRNA normalized counts*

Description

This function uses pivot_longer to rearrange the data for plotting and then plots a CDF of the normalized counts

Usage

```
qc_cdf(gimap_dataset, wide_ar = 0.75)
```

Arguments

gimap_dataset	The special gimap_dataset from the ‘setup_data‘ function which contains the transformed data
wide_ar	aspect ratio, default is 0.75

Value

counts_cdf a ggplot

Examples

```
## Not run:

gimap_dataset <- get_example_data("gimap")
qc_cdf(gimap_dataset)

## End(Not run)
```

qc_constructs_countzero_bar *Create a bar graph that shows the number of replicates with a zero count for pgRNA constructs flagged by the zero count filter*

Description

This bar graph first uses the specified 'filter_zerocount_target_col' columns to flag pgRNA constructs that have a raw count of 0 in any one of those columns/samples of interest. Then, it looks at the specified columns for the final day/sample replicates ('filter_replicates_target_col') to see for pgRNAs that were flagged by the filter, how many of those replicate samples had raw counts of zeros. And it produces a bar plot reporting on this. Note, if you select samples/columns to check with the filter that don't have the replicate samples, this graph won't be informative. So you want there to be overlap between the columns for the two target_col parameters to have an informative graph

Usage

```
qc_constructs_countzero_bar(
  gimap_dataset,
  filter_zerocount_target_col = NULL,
  filter_replicates_target_col = NULL,
  wide_ar = 0.75
)
```

Arguments

gimap_dataset	The special gimap_dataset from the 'setup_data' function which contains the transformed data
filter_zerocount_target_col	default is NULL; Which sample column(s) should be used to check for counts of 0? If NULL and not specified, downstream analysis will select all sample columns
filter_replicates_target_col	default is NULL; Which sample columns are replicates whose variation you'd like to analyze; If NULL, the last 3 sample columns are used
wide_ar	aspect ratio, default is 0.75

Value

a ggplot barplot

Examples

```
## Not run:
gimap_dataset <- get_example_data("gimap")
qc_constructs_countzero_bar(gimap_dataset)
```

```
# or if you want to select a specific column(s) for
# looking at where/which samples zero counts are present for
qc_constructs_countzero_bar(gimap_dataset, filter_zerocount_target_col = 3:5)

# or if you want to select a specific column(s) for the final day/sample replicates
qc_constructs_countzero_bar(gimap_dataset, filter_replicates_target_col = 3:5)

# or some combination of those
qc_constructs_countzero_bar(gimap_dataset,
  filter_zerocount_target_col = 3:5,
  filter_replicates_target_col = 3:5
)

## End(Not run)
```

---

qc_cor_heatmap *Create a correlation heatmap for the pgRNA CPMs*

Description

This function uses the ‘cor‘ function to find correlations between the sample CPM’s and then plots a heatmap of these

Usage

```
qc_cor_heatmap(gimap_dataset)
```

Arguments

gimap_dataset	The special gimap_dataset from the ‘setup_data‘ function which contains the transformed data

Value

‘sample_cor_heatmap‘ a pheatmap

Examples

```
## Not run:
gimap_dataset <- get_example_data("gimap")
qc_cor_heatmap(gimap_dataset)

## End(Not run)
```

qc_filter_plasmid *Create a filter for pgRNAs which have a low log2 CPM value for the plasmid/Day 0 sample/time point*

Description

This function flags and reports which and how many pgRNAs have low log2 CPM values for the plasmid/Day 0 sample/time point. If more than one column is specified as the plasmid sample, we pool all the replicate samples to find the lower outlier and flag constructs for which any plasmid replicate has a log2 CPM value below the cutoff

Usage

```
qc_filter_plasmid(
  gimap_dataset,
  cutoff = NULL,
  filter_plasmid_target_col = NULL
)
```

Arguments

gimap_dataset	The special gimap_dataset from the ‘setup_data‘ function which contains the log2 CPM transformed data
cutoff	default is NULL, the cutoff for low log2 CPM values for the plasmid time period; if not specified, The lower outlier (defined by taking the difference of the lower quartile and 1.5 * interquartile range) is used
filter_plasmid_target_col	default is NULL, and if NULL, will select the first column only; this parameter specifically should be used to specify the plasmid column(s) that will be selected

Value

a named list with the filter ‘filter‘ specifying which pgRNAs have low plasmid log2 CPM (column of interest is ‘plasmid_cpm_filter‘) and a report df ‘reportdf‘ for the number and percent of pgRNA which have a low plasmid log2 CPM

Examples

```
## Not run:
gimap_dataset <- get_example_data("gimap", data_dir = tempdir())

qc_filter_plasmid(gimap_dataset)

# or to specify a cutoff value to be used in the filter rather than the lower
# outlier default
qc_filter_plasmid(gimap_dataset, cutoff = 2)

# or to specify a different column (or set of columns to select)
```

```
qc_filter_plasmid(gimap_dataset, filter_plasmid_target_col = 1:2)

# or to specify a cutoff value that will be used in the filter rather than
# the lower outlier default as well as to specify a different column (or set
# of columns) to select
qc_filter_plasmid(gimap_dataset,
  cutoff = 1.75,
  filter_plasmid_target_col = 1:2
)

## End(Not run)
```

`qc_filter_zerocounts` *Filter out samples of zero counts Create a filter for pgRNAs which have a raw count of 0 for any sample/time # point*

Description

This function flags and reports which and how many pgRNAs have a raw count of 0 for any sample/time point

Usage

```
qc_filter_zerocounts(gimap_dataset, filter_zerocount_target_col = NULL)
```

Arguments

`gimap_dataset`	The special gimap_dataset from the ‘setup_data‘ function which contains the raw count data
`filter_zerocount_target_col`	default is NULL; Which sample column(s) should be used to check for counts of 0? If NULL and not specified, downstream analysis will select all sample columns

Value

a named list with the filter ‘filter‘ specifying which pgRNA have a count zero for at least one sample/time point and a report df ‘reportdf‘ for the number and percent of pgRNA which have a count zero for at least one sample/time point

Examples

```
## Not run:
gimap_dataset <- get_example_data("gimap", data_dir = tempdir())
qc_filter_zerocounts(gimap_dataset)

# or to specify a different column (or set of columns to select)
qc_filter_zerocounts(gimap_dataset, filter_zerocount_target_col = 1:2)
```

```
## End(Not run)
```

`qc_plasmid_histogram` *Create a histogram with plasmid log2 CPM values and ascertain a cutoff for low values*

Description

Find the distribution of plasmid (day0 data) pgRNA log2 CPM values, and ascertain a cutoff or filter for low log2 CPM values. Assumes the first column of the dataset is the day0 data; do I need a better method to tell, especially if there are reps?

Usage

```
qc_plasmid_histogram(
  gimap_dataset,
  cutoff = NULL,
  filter_plasmid_target_col = NULL,
  wide_ar = 0.75
)
```

Arguments

`gimap_dataset`	The special gimap_dataset from the ‘setup_data‘ function which contains the transformed data
`cutoff`	default is NULL, the cutoff for low log2 CPM values for the plasmid time period; if not specified, The lower outlier (defined by taking the difference of the lower quartile and 1.5 * interquartile range) is used
`filter_plasmid_target_col`	default is NULL, and if NULL, will select the first column only; this parameter specifically should be used to specify the plasmid column(s) that will be selected
`wide_ar`	aspect ratio, default is 0.75

Value

a ggplot histogram

Examples

```
## Not run:

gimap_dataset <- get_example_data("gimap")

qc_plasmid_histogram(gimap_dataset)

# or to specify a "cutoff" value that will be displayed as a dashed vertical line
```

```
qc_plasmid_histogram(gimap_dataset, cutoff = 1.75)

# or to specify a different column (or set of columns) to select
qc_plasmid_histogram(gimap_dataset, filter_plasmid_target_col = 1:2)

# or to specify a "cutoff" value that will be displayed as a dashed vertical
# line as well as to specify a different column (or set of columns) to select
qc_plasmid_histogram(gimap_dataset, cutoff = 2, filter_plasmid_target_col = 1:2)

## End(Not run)
```

`qc_sample_hist` *Create a histogram for the pgRNA log2 CPMs, faceted by sample*

Description

This function uses pivot_longer to rearrange the data for plotting and then plots sample specific histograms of the pgRNA cpm's

Usage

```
qc_sample_hist(gimap_dataset, wide_ar = 0.75)
```

Arguments

`gimap_dataset`	The special gimap_dataset from the 'setup_data' function which contains the transformed data
`wide_ar`	aspect ratio, default is 0.75

Value

sample_cpm_histogram a ggplot

Examples

```
## Not run:
gimap_dataset <- get_example_data("gimap")
qc_sample_hist(gimap_dataset)

## End(Not run)
```

qc_variance_hist *Create a histogram for the variance within replicates for each pgRNA*

Description

This function uses pivot_longer to rearrange the data for plotting, finds the variance for each pgRNA construct (using row number as a proxy) and then plots a histogram of these variances

Usage

```
qc_variance_hist(
  gimap_dataset,
  filter_replicates_target_col = NULL,
  wide_ar = 0.75
)
```

Arguments

gimap_dataset	The special gimap_dataset from the ‘setup_data‘ function which contains the transformed data
filter_replicates_target_col	default is NULL; Which sample columns are replicates whose variation you’d like to analyze; If NULL, the last 3 sample columns are used
wide_ar	aspect ratio, default is 0.75

Value

a ggplot histogram

Examples

```
## Not run:
gimap_dataset <- get_example_data("gimap")
qc_variance_hist(gimap_dataset)

## End(Not run)
```

run_qc *Run Quality Control Checks*

Description

This function takes a ‘gimap_dataset‘ and creates a QC report

Usage

```
run_qc(
  gimap_dataset,
  output_file,
  plots_dir,
  overwrite = FALSE,
  filter_zerocount_target_col = NULL,
  filter_plasmid_target_col = NULL,
  filter_replicates_target_col = NULL,
  open_results = TRUE,
  ...
)
```

Arguments

gimap_dataset	A special dataset structure that is setup using the ‘setup_data()‘ function.
output_file	Needs to be a string that ends with ".Rmd" What the name of the output QC report file should be.
plots_dir	directory to save plots created with this function, if it doesn't exist already it will be created
overwrite	default is FALSE; whether to overwrite the QC Report file
filter_zerocount_target_col	default is NULL; Which sample column(s) should be used to check for counts of 0? If NULL and not specified, downstream analysis will select all sample columns
filter_plasmid_target_col	default is NULL; Which sample columns(s) should be used to look at log2 CPM expression for plasmid pgRNA constructs? If NULL and not specified, downstream analysis will select the first sample column only
filter_replicates_target_col	default is NULL; Which sample columns are replicates whose variation you'd like to analyze; If NULL, the last 3 sample columns are used
open_results	default is TRUE but if you don't want the report automatically opened, choose FALSE.
...	additional parameters are sent to ‘rmarkdown::render()‘

Value

a QC report saved locally

Examples

```
## Not run:

gimap_dataset <- get_example_data("gimap")

run_qc(
```

```
  gimap_dataset,
  plots_dir = tempdir(),
  output_file = paste0(tempfile(), "_QC_Report.Rmd")
)

## End(Not run)
```

`save_example_timepoint_data` *Set up example count data*

Description

Set up example count data

Usage

```
save_example_timepoint_data(data_dir = NULL)
```

Arguments

`data_dir`	Optional data directory override.

Value

Returns the file path to folder where the example data is stored

---

`save_example_treatment_data` *Set up example count data*

Description

Set up example count data

Usage

```
save_example_treatment_data(data_dir = NULL)
```

Arguments

`data_dir`	Optional data directory override.

Value

Returns the file path to folder where the example data is stored

setup_data *Making a new gimap dataset*

Description

This function allows people to have their data ready to be processed by gimap

Usage

```
setup_data(counts = NULL, pg_ids = NULL, sample_metadata = NULL)
```

Arguments

counts	a matrix of data that contains the counts where rows are each paired_guide target and columns are each sample
pg_ids	the pgRNA IDs: metadata associated with the pgRNA constructs that correspond to the rows of the counts data
sample_metadata	metadata associated with the samples of the dataset that correspond to the columns of the counts data. Should include a column that has replicate information as well as a column that contains timepoint information respectively (this will be used for log fold calculations). These columns should be factors.

Value

A special gimap_dataset to be used with the other functions in this package.

Examples

```
## Not run:

counts <- get_example_data("count", data_dir = tempdir()) %>%
  dplyr::select(c(
    "Day00_RepA", "Day05_RepA", "Day22_RepA", "Day22_RepB",
    "Day22_RepC"
  )) %>%
  as.matrix()

pg_ids <- get_example_data("count", data_dir = tempdir()) %>%
  dplyr::select("id")

sample_metadata <- data.frame(
  col_names = c("Day00_RepA", "Day05_RepA", "Day22_RepA", "Day22_RepB", "Day22_RepC"),
  day = as.numeric(c("0", "5", "22", "22", "22")),
  rep = as.factor(c("RepA", "RepA", "RepA", "RepB", "RepC"))
)

gimap_dataset <- setup_data(
```

```
  counts = counts,
  pg_ids = pg_ids,
  sample_metadata = sample_metadata
)

## End(Not run)
```

`supported_cell_lines` *List the supported cell lines*

Description

This function downloads the metadata for DepMap and lists which cell lines are supported.

Usage

```
supported_cell_lines()
```

Value

A list of the cell line names that are available in DepMap for use for annotation in this package.

Examples

```
## Not run:

cell_lines <- supported_cell_lines()

## End(Not run)
```

`tpm_setup` *Download and set up DepMap TPM data*

Description

This function sets up the tpm data from DepMap is called by the ‘gimap_annotate()‘ function

Usage

```
tpm_setup(
  overwrite = TRUE,
  data_dir = gimap_data_dir()
)
```

Arguments

`overwrite`	should the files be re downloaded
`data_dir`	What directory should this be saved to? Default is a user data directory.

Index