Package: hidecan (via r-universe)

September 18, 2024

Title Create HIDECAN Plots for Visualising Genome-Wide Association Studies and Differential Expression Results

Version 1.1.0.9000

Description Generates HIDECAN plots that summarise and combine the results of genome-wide association studies (GWAS) and transcriptomics differential expression analyses (DE), along with manually curated candidate genes of interest. The HIDECAN plot is presented in: Angelin-Bonnet, O., Vignes, M., Biggs, P. J., Baldwin, S., & Thomson, S. (2023). Visual integration of GWAS and differential expression results with the hidecan R package. bioRxiv, 2023-03.

License MIT + file LICENSE

URL https://plantandfoodresearch.github.io/hidecan/, https://github.com/PlantandFoodResearch/hidecan

BugReports https://github.com/PlantandFoodResearch/hidecan/issues

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Depends R (>= 2.10)

Imports dplyr, ggnewscale, ggplot2, ggrepel, purrr, RColorBrewer, shiny, tibble, tidyr, viridis, vroom

Suggests knitr, rmarkdown, stringr, testthat (>= 3.0.0)

Config/testthat/edition 3

VignetteBuilder knitr

Language en-GB

Repository https://plantandfoodresearch.r-universe.dev

RemoteUrl https://github.com/plantandfoodresearch/hidecan

RemoteRef HEAD

RemoteSha ee0d6a100bf0e60288d63bb86b6ba38ae5e5f7c1

Contents

`.add_data_type` *Add data type to HIDECAN plot*

Description

Add data type to HIDECAN plot

Usage

```
.add_data_type(
  p,
  toplot,
  i,
  paes,
  add_new_legend,
  point_size,
  label_size,
  label_padding
)
```

Arguments

p	Current ggplot.
toplot	Tibble of data to plot.
i	Character, data type to add in the plot.
paes	Named list, aesthetics to use for this data type (see hidecan_aes() for information about the required values).
add_new_legend	Logical, whether the current data type requires a new legend for fill.
point_size	Numeric, size of the points in the plot.
label_size	Numeric, size of the gene labels in the plot.
label_padding	Numeric, amount of padding around gene labels in the plot, as unit or number.

Value

Ggplot p to which the new data type has been added.

.check_chroms *Check chromosomes to plot*

Description

Check chromosomes to plot

Usage

```
.check_chroms(toplot, chrom_length, chroms, remove_empty_chrom)
```

Arguments

toplot	Tibble, main data-frame for plot.
chrom_length	Tibble of chromosome length.
chroms	Character vector, name of chromosomes to include in the plot. If NULL (default value), all chromosomes will be included.
remove_empty_chrom	Logical, should chromosomes with no significant markers/genes nor candidate genes be removed from the plot? Default value if FALSE.

Value

Character vector of chromosomes to plot.

.check_chrom_limits *Check chromosome limits*

Description

Check chromosome limits

Usage

```
.check_chrom_limits(chrom_limits, chrom_length, chroms)
```

Arguments

chrom_limits	Integer vector of length 2, or named list where the elements are integer vectors of length 2. If vector, gives the lower and upper limit of the chromosomes (in bp) to use in the plot. If a named list, names should correspond to chromosome names. Gives for each chromosome the lower and upper limits (in bp) to use in the plot. Doesn't have to be specified for all chromosomes. Default value is NULL, i.e. no limits are applied to the chromosomes (they will be plotted in their entirety).
chrom_length	Tibble of chromosome length.
chroms	Character vector, name of chromosomes to include in the plot.

Value

Tibble of chromosome limits to use in the plot.

.check_cols *Checks whether some columns are present in a tibble*

Description

Checks whether some columns are present in a tibble

Usage

```
.check_cols(x, col_names, param_name = "Input data-frame")
```

Arguments

x	Tibble
col_names	character vector of column names
param_name	Character, name of the dataframe to use in the error message.

Value

invisible NULL

.compute_chrom_length_genes *Computes chromosomes' length for a tibble of genes*

Description

Computes the length (in bp) of each chromosome as the maximum position of genes on the chromosome.

Usage

```
.compute_chrom_length_genes(x)
```

Arguments

x	Either a DE_data or CAN_data object.

Value

A tibble with two columns: chromosome (chromosome name) and length (chromosome length in base pair).

.compute_chrom_length_markers *Computes chromosomes' length for a tibble of markers*

Description

Computes the length (in bp) of each chromosome as the maximum position of chromosomes on the chromosome.

Usage

```
.compute_chrom_length_markers(x)
```

Arguments

x	Either a GWAS_data or CUSTOM_data object.

Value

A tibble with two columns: chromosome (chromosome name) and length (chromosome length in base pair).

.get_aes_type *Returns either "aes_type" attribute or object class*

Description

Returns either "aes_type" attribute or object class

Usage

```
.get_aes_type(x)
```

Arguments

x	A GWAS_data_thr, DE_data_thr, CAN_data_thr or CUSTOM_data_thr object.

Value

Either "aes_type" attribute or object class.

.get_plot_aes *Computes plot aesthetics*

Description

Get the list of aesthetics for each data types in the plot.

Usage

```
.get_plot_aes(aes_types, colour_genes_by_score, custom_aes)
```

Arguments

aes_types	Character vector of data types (one per dataset to plot), should be computed with .get_aes_type().
colour_genes_by_score	Logical, whether to colour the genes by score (TRUE) or by log2(fold-change) (FALSE).
custom_aes	Named list of plot aesthetics for custom data types. See hidecan_aes() for information about the content of each element.

Value

Named list of plot aesthetics.

apply_threshold *Filters GWAS or DE results based on a threshold*

Description

Filters markers or genes/transcripts based on a threshold applied to their GWAS or DE score, and log2(fold-change) (if applicable). For a set of candidate genes, simply returns the list. Note that markers or genes with a missing score or log2(fold-change) will be removed from the dataset.

Usage

```
apply_threshold(x, score_thr = 0, log2fc_thr = 0)

## S3 method for class 'GWAS_data'
apply_threshold(x, score_thr = 0, log2fc_thr = 0)

## S3 method for class 'DE_data'
apply_threshold(x, score_thr = 0, log2fc_thr = 0)

## S3 method for class 'CAN_data'
```

```
apply_threshold(x, score_thr = 0, log2fc_thr = 0)

## S3 method for class 'CUSTOM_data'
apply_threshold(x, score_thr = 0, log2fc_thr = 0)

## Default S3 method:
apply_threshold(x, score_thr = 0, log2fc_thr = 0)
```

Arguments

x	Either a GWAS_data, DE_data, CAN_data or CUSTOM_data object.
score_thr	Numeric, threshold to use on markers' or genes/transcripts' score. Only markers or genes with a score equal to or higher than this threshold will be retained. Default value is 0. Ignored for CAN_data.
log2fc_thr	Numeric, threshold to use on the absolute value of genes/ transcripts' log2(fold-change). Only genes/transcripts with an absolute log2(fold-change) equal to or higher than this threshold will be retained. Ignored for GWAS_data, CAN_data and CUSTOM_data.

Value

A filtered tibble (of class GWAS_data_thr, DE_data_thr, CAN_data_thr or CUSTOM_data_thr).

Examples

```
x <- get_example_data()

## For GWAS results
apply_threshold(GWAS_data(x[["GWAS"]]), score_thr = 4)

## For DE results - in second line, no threshold is applied
## on the log2(fold-change)
apply_threshold(DE_data(x[["DE"]]), score_thr = -log10(0.05), log2fc_thr = 1)
apply_threshold(DE_data(x[["DE"]]), score_thr = -log10(0.05), log2fc_thr = 0)

## No effect on the Candidate genes
apply_threshold(CAN_data(x[["CAN"]]))
```

CAN_data *Creates a* CAN_data *object*

Description

Creates a CAN_data object from a tibble or data-frame of candidate genes.

Usage

```
CAN_data(dat, keep_rownames_as = NULL)
```

Arguments

dat	Tibble, set of candidate genes of interest. See Details.
keep_rownames_as	Character, the name of the column in which to save the rownames of the input data-frame. Default value is NULL, i.e. rownames will be discarded.

Details

The input data should have one row per gene, and at least the following columns:

- chromosome: character column, chromosome on which the gene is located.
- start and end: numeric, starting and end position of the gene (in bp). A column position will be constructed as the middle value (mean) between start and end.
- name: character, the name of the candidate genes to be displayed.

Value

A CAN_data object, i.e. a tibble.

Examples

```
x <- get_example_data()

CAN_data(x[["CAN"]])
```

---

combine_chrom_length *Computes chromosomes' length from list*

Description

Computes the length (in bp) of each chromosome from a list of GWAS and DE results as well as candidate gene lists.

Usage

```
combine_chrom_length(x)
```

Arguments

x	A list of GWAS_data, DE_data, CAN_data or CUSTOM_data objects.

Value

A tibble with two columns: chromosome (chromosome name) and length (chromosome length in base pair).

Examples

```
x <- get_example_data()
y <- list("GWAS" = GWAS_data(x[["GWAS"]]),
          "DE" = DE_data(x[["DE"]]),
          "CAN" = CAN_data(x[["CAN"]]))

combine_chrom_length(y)
```

`compute_chrom_length` *Computes chromosomes' length*

Description

Computes the length (in bp) of each chromosome as the maximum position of markers or genes on the chromosome.

Usage

```
compute_chrom_length(x)

## S3 method for class 'GWAS_data'
compute_chrom_length(x)

## S3 method for class 'DE_data'
compute_chrom_length(x)

## S3 method for class 'CAN_data'
compute_chrom_length(x)

## S3 method for class 'CUSTOM_data'
compute_chrom_length(x)
```

Arguments

`x`	Either a `GWAS_data`, `DE_data`, `CAN_data` or `CUSTOM_data` object.

Value

A tibble with two columns: `chromosome` (chromosome name) and `length` (chromosome length in base pair).

Examples

```
x <- get_example_data()

compute_chrom_length(GWAS_data(x[["GWAS"]]))
compute_chrom_length(DE_data(x[["DE"]]))
compute_chrom_length(CAN_data(x[["CAN"]]))
```

`create_hidecan_plot` *Creates a HIDECAN plot*

Description

Creates a HIDECAN plot from a list of filtered GWAS or DE results and/or candidate genes.

Usage

```
create_hidecan_plot(
  x,
  chrom_length,
  colour_genes_by_score = TRUE,
  remove_empty_chrom = FALSE,
  chroms = NULL,
  chrom_limits = NULL,
  title = NULL,
  subtitle = NULL,
  n_rows = NULL,
  n_cols = 2,
  legend_position = "bottom",
  point_size = 3,
  label_size = 3.5,
  label_padding = 0.15,
  custom_aes = NULL
)
```

Arguments

`x`	A list of `GWAS_data_thr`, `DE_data_thr`, `CAN_data_thr` and/or `CUSTOM_data_thr` produced by the `apply_threshold()` function. If named, the names will be appended to the y-axis labels (use `' '` as empty name in the list).
`chrom_length`	Tibble with columns `chromosome` and `length`, giving for each chromosome its length in bp (see `combine_chrom_length()` function).
`colour_genes_by_score`	Logical, whether to colour the genes by score (`TRUE`) or by log2(fold-change) (`FALSE`). Default value is `TRUE`.
`remove_empty_chrom`	Logical, should chromosomes with no significant markers/genes nor candidate genes be removed from the plot? Default value if `FALSE`.
`chroms`	Character vector, name of chromosomes to include in the plot. If `NULL` (default value), all chromosomes will be included.
`chrom_limits`	Integer vector of length 2, or named list where the elements are integer vectors of length 2. If vector, gives the lower and upper limit of the chromosomes (in bp) to use in the plot. If a named list, names should correspond to chromosome names. Gives for each chromosome the lower and upper limits (in bp) to use

	in the plot. Doesn't have to be specified for all chromosomes. Default value is NULL, i.e. no limits are applied to the chromosomes (they will be plotted in their entirety).
title	Character, title of the plot. Default value is NULL (i.e. no title will be added to the plot).
subtitle	Character, subtitle of the plot. Default value is NULL (i.e. no subtitle will be added to the plot).
n_rows	Integer, number of rows of chromosomes to create in the plot. Default value is NULL.
n_cols	Integer, number of columns of chromosomes to create in the plot. Default value is 2. Will be set to NULL if n_rows is not NULL.
legend_position	Character, position of the legend in the plot. Can be bottom (default value), top, right, left or none.
point_size	Numeric, size of the points in the plot. Default value is 3.
label_size	Numeric, size of the gene labels in the plot. Default value is 3.5 (for geom_label_repel).
label_padding	Numeric, amount of padding around gene labels in the plot, as unit or number. Default value is 0.15 (for geom_label_repel).
custom_aes	Named list of plot aesthetics for custom data types. See hidecan_aes() for information about the content of each element. Default is NULL (only needed if there are CUSTOM_data_thr objects in x or to customise the aesthetics for the default data tracks).

Value

A ggplot.

Examples

```
if (interactive()) {
x <- get_example_data()
y <- list("GWAS" = GWAS_data(x[["GWAS"]]),
          "DE" = DE_data(x[["DE"]]),
          "CAN" = CAN_data(x[["CAN"]]))

chrom_length <- combine_chrom_length(y)

z <- list(
  apply_threshold(y[["GWAS"]], score_thr = 4),
  apply_threshold(y[["DE"]], score_thr = 1.3, log2fc_thr = 0.5),
  apply_threshold(y[["CAN"]])
)

create_hidecan_plot(z,
                    chrom_length,
                    label_size = 2)

## Colour genes according to their fold-change
```

```
create_hidecan_plot(z,
                    chrom_length,
                    colour_genes_by_score = FALSE,
                    label_size = 2)

## Add names to the datasets
create_hidecan_plot(setNames(z, c("Genomics", "RNAseq", "My list")),
                    chrom_length,
                    colour_genes_by_score = FALSE,
                    label_size = 2)

## Add names to some of the datasets only (e.g. not for GWAS results)
create_hidecan_plot(setNames(z, c(" ", "RNAseq", "My list")),
                    chrom_length,
                    colour_genes_by_score = FALSE,
                    label_size = 2)

## Set limits on all chromosomes (to "zoom in" to the 10-20Mb region)
create_hidecan_plot(z,
                    chrom_length,
                    label_size = 2,
                    chrom_limits = c(10e6, 20e6))

## Set limits on some chromosomes only
create_hidecan_plot(z,
                    chrom_length,
                    label_size = 2,
                    chrom_limits = list("ST4.03ch00" = c(10e6, 20e6),
                                        "ST4.03ch02" = c(15e6, 25e6)))
}
```

CUSTOM_data *Creates a* CUSTOM_data *object*

Description

Creates a CUSTOM_data object from a tibble or data-frame of custom genomic features.

Usage

```
CUSTOM_data(dat, keep_rownames_as = NULL)
```

Arguments

dat	Tibble of custom genomic features. See Details.
keep_rownames_as	Character, the name of the column in which to save the rownames of the input data-frame. Default value is NULL, i.e. rownames will be discarded.

Details

The input data should have one row per marker, and at least the following columns:

- chromosome: character column, chromosome on which the feature is located.
- position: numeric, the physical position of the feature along the chromosome (in bp).
- score: numeric, score to be used for the genomic feature.

Value

A CUSTOM_data object, i.e. a tibble.

Examples

```
x <- get_example_data()

CUSTOM_data(x[["GWAS"]])
```

DE_data *Creates a* DE_data *object*

Description

Creates a DE_data object from a tibble or data-frame of differential expression results.

Usage

```
DE_data(dat, keep_rownames_as = NULL)
```

Arguments

dat	Tibble, results from a differential expression analysis. See Details.
keep_rownames_as	Character, the name of the column in which to save the rownames of the input data-frame. Default value is NULL, i.e. rownames will be discarded.

Details

The input data should have one row per gene or transcript, and at least the following columns:

- chromosome: character column, chromosome on which the gene/transcript is located.
- start and end: numeric, starting and end position of the gene/transcript (in bp). A column position will be constructed as the middle value (mean) between start and end.
- score or padj: numeric, the DE score or adjusted p-value of the gene/transcript. If column score column is missing, will be constructed as -log10(padj).
- foldChange or log2FoldChange: numeric, the fold-change or log2(fold-change) of the gene/transcript. If column log2FoldChange is missing, will be constructed as log2(foldChange).

Value

A DE_data object, i.e. a tibble.

Examples

```
x <- get_example_data()

DE_data(x[["DE"]])
```

---

get_example_data *Example dataset*

Description

Returns a list of example datasets.

Usage

```
get_example_data()
```

Details

The dataset used in this example is presented in: Angelin-Bonnet et al., BMC Plant Biology (2023). In this study, tetraploid potato plants from a half-sibling breeding population were used to assess the genetic components of tuber bruising. Capture sequencing was used to obtain genomic information about the individuals, and a genome-wide association study (GWAS) was performed on 72,847 genomic biallelic variants obtained from 158 plants for which a bruising score was measured. The GWAS analysis was carried with the GWASpoly package. In addition, expression data was obtained for 25,163 transcribed genes, and a differential expression (DE) analysis was carried out between 41 low- and 33 high-bruising samples. Finally, a literature search yielded a list of 42 candidate genes identified in previous studies as involved in potato tuber bruising mechanisms. A subset of the GWAS and DE results, as well as the list of candidate genes from the literature, are made available in this function. From the complete GWAS results table, half of the genomic variants with a GWAS score < 3.5 were randomly selected and consequently discarded, yielding a dataset with GWAS scores for 35,481 variants. Similarly, half of the transcribed genes in the DE results table with an adjusted p-value > 0.05 were randomly selected and discarded, yielding a dataset with DE results for 10,671 transcribed genes. This filtering was performed to reduce the size of the datasets (in accordance with CRAN policies), but ensures that all significant markers and genes are retained in the datasets. Finally, some of the candidate genes located on chromosome 3 were removed from the example dataset for better clarity in the resulting HIDECAN plot, leaving 32 candidate genes.

Value

A list with the following elements:

- GWAS: a tibble of GWAS results, with columns id, chromosome, position and score.
- DE: a tibble of differential expression results, with columns gene, chromosome, padj, log2FoldChange, start, end and label.

- CAN: a tibble of candidate genes, with columns id, chromosome, start, end, name and gene_name.

GWAS_data *Creates a* GWAS_data *object*

Description

Creates a GWAS_data object from a tibble or data-frame of GWAS results.

Usage

```
GWAS_data(dat, keep_rownames_as = NULL)
```

Arguments

dat	Tibble, results from a GWAS analysis. See Details.
keep_rownames_as	Character, the name of the column in which to save the rownames of the input data-frame. Default value is NULL, i.e. rownames will be discarded.

Details

The input data should have one row per marker, and at least the following columns:

- chromosome: character column, chromosome on which the marker is located.
- position: numeric, the physical position of the marker along the chromosome (in bp).
- score or padj: numeric, the GWAS score or adjusted p-value of the marker. If column score column is missing, will be constructed as -log10(padj).

Value

A GWAS_data object, i.e. a tibble.

Examples

```
x <- get_example_data()

GWAS_data(x[["GWAS"]])
```

GWAS_data_from_gwaspoly *Extracts information from GWASpoly output*

Description

Extracts GWAS results and chromosome length from GWASpoly output.

Usage

```
GWAS_data_from_gwaspoly(gwaspoly_output, traits = NULL, models = NULL)
```

Arguments

gwaspoly_output	A GWASpoly.fitted or GWASpoly.thresh object (returned by GWASpoly::GWASpoly() or GWASpoly::set.threshold() functions).
traits	Character vector, traits for which GWAS results should be extracted. If NULL (default value), all traits present are considered.
models	Character vector, genetic models for which GWAS results should be extracted. If NULL (default value), all genetic models present are considered.

Value

A list with the following elements:

- gwas_data_list: A named list of GWAS_data objects, giving the markers score for each possible trait/genetic model combination. The names of the list are in the form trait (genetic model).
- gwas_data_thr_list: if the input data is a GWASpoly.thresh object (from the GWASpoly::set.threshold() function), a named list of GwAS_data_thr, with the significant markers score for each possible trait/genetic model combination. The names of the list are in the form trait (genetic model).
- chrom_length: A tibble with one row per chromosome, giving the length (in bp) of each chromosome.

hidecan_aes *Default aesthetics for HIDECAN plot*

Description

Generates a list of the default aesthetics used for HIDECAN plots.

Usage

```
hidecan_aes(colour_genes_by_score = TRUE)
```

Arguments

colour_genes_by_score
: Logical, whether to colour the genes by score (TRUE) or by log2(fold-change) (FALSE). Default value is TRUE.

Value

A named list, with one element per type of data (e.g. GWAS, DE, etc). Each element is itself a list with the following elements:

- y_label: prefix added to the name of a track on the y-axis of the plot.
- line_colour: colour of the vertical line used to show the position of elements of this type.
- point_shape: shape used for the points of this type.
- show_name: whether a label with name value should be added to points of this type.
- fill_scale: fill scale to use for the points of this type.

---

hidecan_plot *Wrapper to create a HIDECAN plot*

Description

Wrapper function to create a HIDECAN plot from GWAS results, DE results or candidate genes.

Usage

```
hidecan_plot(
  gwas_list = NULL,
  de_list = NULL,
  can_list = NULL,
  score_thr_gwas = 4,
  score_thr_de = 2,
  log2fc_thr = 1,
  chrom_length = NULL,
  colour_genes_by_score = TRUE,
  remove_empty_chrom = FALSE,
  chroms = NULL,
  chrom_limits = NULL,
  title = NULL,
  subtitle = NULL,
  n_rows = NULL,
  n_cols = 2,
  legend_position = "bottom",
  point_size = 3,
  label_size = 3.5,
  label_padding = 0.15,
  custom_aes = NULL,
```

```
  custom_list = NULL,
  score_thr_custom = 0
)
```

Arguments

gwas_list	Data-frame or list of data-frames containing GWAS results, each with at least a chromosome, position and either padj or score columns. If a named list, the names will be used in the plot.
de_list	Data-frame or list of data-frames containing DE results, each with at least a chromosome, start, end, log2FoldChange and either padj or score columns. If a named list, the names will be used in the plot.
can_list	Data-frame or list of data-frames containing candidate genes, each with at least a chromosome, start, end and name columns. If a named list, the names will be used in the plot.
score_thr_gwas	Numeric, the score threshold for GWAS results that will be used to select which markers will be plotted. Default value is 4.
score_thr_de	Numeric, the score threshold for DE results that will be used to select which markers will be plotted. Default value is 2.
log2fc_thr	Numeric, the log2(fold-change) threshold that will be used to select which genes will be plotted. Default value is 1.
chrom_length	Optional, tibble with columns chromosome and length, giving for each chromosome its length in bp. If NULL (the default), will be inferred from the GWAS, DE and candidate gene data.
colour_genes_by_score	Logical, whether to colour the genes by score (TRUE) or by log2(fold-change) (FALSE). Default value is TRUE.
remove_empty_chrom	Logical, should chromosomes with no significant markers/genes nor candidate genes be removed from the plot? Default value if FALSE.
chroms	Character vector, name of chromosomes to include in the plot. If NULL (default value), all chromosomes will be included.
chrom_limits	Integer vector of length 2, or named list where the elements are integer vectors of length 2. If vector, gives the lower and upper limit of the chromosomes (in bp) to use in the plot. If a named list, names should correspond to chromosome names. Gives for each chromosome the lower and upper limits (in bp) to use in the plot. Doesn't have to be specified for all chromosomes. Default value is NULL, i.e. no limits are applied to the chromosomes (they will be plotted in their entirety).
title	Character, title of the plot. Default value is NULL (i.e. no title will be added to the plot).
subtitle	Character, subtitle of the plot. Default value is NULL (i.e. no subtitle will be added to the plot).
n_rows	Integer, number of rows of chromosomes to create in the plot. Default value is NULL.

n_cols — Integer, number of columns of chromosomes to create in the plot. Default value is 2. Will be set to NULL if n_rows is not NULL.

legend_position — Character, position of the legend in the plot. Can be bottom (default value), top, right, left or none.

point_size — Numeric, size of the points in the plot. Default value is 3.

label_size — Numeric, size of the gene labels in the plot. Default value is 3.5 (for geom_label_repel).

label_padding — Numeric, amount of padding around gene labels in the plot, as unit or number. Default value is 0.15 (for geom_label_repel).

custom_aes — Named list of plot aesthetics for custom data types. See hidecan_aes() for information about the content of each element. Default is NULL (only needed if there are CUSTOM_data_thr objects in x or to customise the aesthetics for the default data tracks).

custom_list — Data-frame or list of data-frames containing custom genomic features, each with at least a chromosome, position and score columns. If a named list, the names will be used in the plot.

score_thr_custom — Numeric, the score threshold for custom genomic features that will be used to select which markers will be plotted. Default value is 0.

Value

a ggplot.

Examples

```
if (interactive()) {
x <- get_example_data()

## Typical example with one GWAs result table, one DE result table and
## one table of candidate genes
hidecan_plot(gwas_list = x[["GWAS"]],
             de_list = x[["DE"]],
             can_list = x[["CAN"]],
             score_thr_gwas = -log10(0.0001),
             score_thr_de = -log10(0.005),
             log2fc_thr = 0,
             label_size = 2)

## Example with two sets of GWAS results
hidecan_plot(gwas_list = list(x[["GWAS"]], x[["GWAS"]]),
             score_thr_gwas = 4)

## Example with two sets of DE results, with names
hidecan_plot(de_list = list("X vs Y" = x[["DE"]],
                            "X vs Z" = x[["DE"]]),
             score_thr_de = -log10(0.05),
             log2fc_thr = 0)
```

```
## Set limits on all chromosomes (to "zoom in" to the 10-20Mb region)
hidecan_plot(gwas_list = x[["GWAS"]],
             de_list = x[["DE"]],
             can_list = x[["CAN"]],
             score_thr_gwas = -log10(0.0001),
             score_thr_de = -log10(0.005),
             log2fc_thr = 0,
             label_size = 2,
             chrom_limits = c(10e6, 20e6))

## Set limits on some chromosomes only
hidecan_plot(gwas_list = x[["GWAS"]],
             de_list = x[["DE"]],
             can_list = x[["CAN"]],
             score_thr_gwas = -log10(0.0001),
             score_thr_de = -log10(0.005),
             log2fc_thr = 0,
             label_size = 2,
             chrom_limits = list("ST4.03ch00" = c(10e6, 20e6),
                                 "ST4.03ch02" = c(15e6, 25e6)))
}
```

`hidecan_plot_from_gwaspoly` *Creates a HIDECAN plot from GWASpoly output*

Description

Creates a HIDECAN plot from the GWAS results from GWASpoly.

Usage

```
hidecan_plot_from_gwaspoly(gwaspoly_output, traits = NULL, models = NULL, ...)
```

Arguments

`gwaspoly_output`	A `GWASpoly.thresh` object (returned by the `GWASpoly::set.threshold()` function).
`traits`	Character vector, traits for which GWAS results should be extracted. If `NULL` (default value), all traits present are considered.
`models`	Character vector, genetic models for which GWAS results should be extracted. If `NULL` (default value), all genetic models present are considered.
`...`	Further arguments passed to the `create_hidecan_plot()` function.

Value

A `ggplot`.

manhattan_plot *Creates a Manhattan plot*

Description

Creates a Manhattan plot from a data-frame of GWAS results.

Usage

```
manhattan_plot(
  gwas_list,
  score_thr = NULL,
  chroms = NULL,
  title = NULL,
  subtitle = NULL,
  size_range = c(1, 3),
  chrom_col = NULL,
  ncol = NULL
)
```

Arguments

gwas_list	Data-frame or list of data-frames containing GWAS results, each with at least a chromosome, position and either padj or score columns. If a named list, the names will be used in the plot.
score_thr	Numeric, the significance threshold on GWAS score. If a value is provided, will be represented in the Manhattan plot as a red dashed line. If NULL (default value), no significance threshold line will be drawn.
chroms	Character vector, name of chromosomes to include in the plot. If NULL (default value), all chromosomes will be included.
title	Character, title of the plot. Default value is NULL (i.e. no title will be added to the plot).
subtitle	Character, subtitle of the plot. Default value is NULL (i.e. no subtitle will be added to the plot).
size_range	Numeric vector of length 2, the minimum and maximum point size in the plot. Points size is proportional to their GWAS score. Default value is c(1, 3).
chrom_col	Character vector of colour names or code, colours to use to draw the points for each chromosome. Names will be ignored. If vector provided contains less colours than the number of chromosomes to plot, the values will be recycled. If NULL, default, colours will be automatically chosen from a predefined palette.
ncol	Integer, number of Manhattan plots per row when several GWAS results are provided.

Value

A ggplot.

Examples

```
if (interactive()){
x <- get_example_data()[["GWAS"]]

manhattan_plot(x)

## Adding a significance threshold line in the plot
manhattan_plot(x, score_thr = 4)


## Use only two colours for the chromosomes
manhattan_plot(x,
               score_thr = 4,
               chrom_col = c("dodgerblue1", "dodgerblue4"))
}
```

---

new_CAN_data CAN_data *constructor*

Description

CAN_data constructor

Usage

```
new_CAN_data(dat)
```

Arguments

dat	Tibble, containing information about genes of interest, with at least columns chromosome, start, end, position and name.

Value

A CAN_data object, i.e. a tibble.

---

new_CUSTOM_data CUSTOM_data *constructor*

Description

CUSTOM_data constructor

Usage

```
new_CUSTOM_data(dat)
```

Arguments

dat	Tibble, custom genomic features, with at least columns chromosome, position and score.

Value

A GWAS_data object, i.e. a tibble.

---

new_DE_data DE_data *constructor*

Description

DE_data constructor

Usage

```
new_DE_data(dat)
```

Arguments

dat	Tibble, results from a differential expression analysis, with at least columns chromosome, score, log2FoldChange, start, end and position.

Value

A DE_data object, i.e. a tibble.

---

new_GWAS_data GWAS_data *constructor*

Description

GWAS_data constructor

Usage

```
new_GWAS_data(dat)
```

Arguments

dat	Tibble, results from a GWAS analysis, with at least columns chromosome, position and score.

Value

A GWAS_data object, i.e. a tibble.

run_hidecan_shiny *Launches the HIDECAN shiny app*

Description

Starts the HIDECAN shiny app. The app reads in csv data to produce a HIDECAN plot.

Usage

```
run_hidecan_shiny()
```

Value

No return value, called for side effects (launching the shiny app).

validate_CAN_data *Checks validity of input for* CAN_data *constructor*

Description

Checks validity of input for CAN_data constructor

Usage

```
validate_CAN_data(x)
```

Arguments

x	a CAN_data object constructed via new_CAN_data.

Value

A CAN_data object, i.e. a tibble.

validate_CUSTOM_data *Checks validity of input for* CUSTOM_data *constructor*

Description

Checks validity of input for CUSTOM_data constructor

Usage

```
validate_CUSTOM_data(x)
```

Arguments

x	a CUSTOM_data object constructed via new_CUSTOM_data.

Value

A CUSTOM_data object, i.e. a tibble.

validate_DE_data *Checks validity of input for* DE_data *constructor*

Description

Checks validity of input for DE_data constructor

Usage

```
validate_DE_data(x)
```

Arguments

x	a DE_data object constructed via new_DE_data.

Value

A DE_data object, i.e. a tibble.

validate_GWAS_data *Checks validity of input for* GWAS_data *constructor*

Description

Checks validity of input for GWAS_data constructor

Usage

```
validate_GWAS_data(x)
```

Arguments

x	a GWAS_data object constructed via new_GWAS_data.

Value

A GWAS_data object, i.e. a tibble.

Index