Package: ggforestplot (via r-universe)

September 30, 2024

Type Package

Title Forestplots of Measures of Effects and Their Confidence Intervals

Version 0.1.0

Description A collection of functions, based on ggplot2, to plot forestplots of measures of effects, e.g. linear associations, with their confidence intervals.

Depends R (>= 3.5.0)

Imports broom, dplyr (>= 0.8.0), generics, ggforce, ggplot2, ggplotify, ggstance, grid, gridExtra, lazyeval, magrittr, patchwork, purrr, rlang, scales, stringr, survival, tibble, tidyr (>= 1.0.0), tidyselect

Suggests knitr, readr, rmarkdown, tidyverse, testthat, vdiffr

License MIT + file LICENSE

BugReports https://github.com/nightingalehealth/ggforestplot/issues

URL https://nightingalehealth.github.io/ggforestplot/index.html, https://github.com/nightingalehealth/ggforestplot

Encoding UTF-8

LazyData true

RoxygenNote 7.0.2

VignetteBuilder knitr

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

RemoteUrl https://github.com/NightingaleHealth/ggforestplot

RemoteRef HEAD

RemoteSha 547617e63fa481a5f28ffc56c07d46be4af688b2

Contents

df_demo_metabolic_data 2
df_grouping_all_NG_biomarkers 3
df_linear_associations 4
df_logodds_associations 5
df_NG_biomarker_metadata 6
discovery_regression 6
forestplot 10
geom_effect 13
geom_stripes 16
ng_colour 18
ng_palette_d 19
plot_all_NG_biomarkers 20
scale_colour_ng_d 24
theme_forest 25

Index **27**

`df_demo_metabolic_data` *Simulated Metabolic Data*

Description

A simulated, demo data frame that contains metabolic profiles, basic information and diabetes outcomes for 1887 fictional individuals. The data frame contains values for 250 serum biomarkers quantified by the NMR platform of Nightingale Health Ltd.

Usage

```
df_demo_metabolic_data
```

Format

A data frame (tibble) with 1887 rows and 256 columns. `id` is a character variable with the ID number of fictional individual; `gender` is a character variable with the gender information on each individual; `baseline_age` is a numeric variable with the age at the time of blood draw; `BMI` is a numeric variable with the BMI of each individual at the time of blood draw; `incident_diabetes` is a numeric variable with values 1 or 0 for whether the diabetes event occured or not during the observed time, respectively; `age_at_diabetes` is a numeric variable with the age at the end of the study and the rest of the variables are numeric containing the machine readable names of Nightingale NMR blood biomarkers.

Source

Simulated NMR data; Nightingale Health Ltd, https://nightingalehealth.com/.

See Also

df_NG_biomarker_metadata

`df_grouping_all_NG_biomarkers`
Example of Biomarker Grouping Data Frames for Nightingale Biomarkers

---

Description

A data frame containing example custom groupings for the Nightingale Health biomarkers in a machine readable name format. There are two custom groupings available for 2016 and 2020 biomarker platforms. These groupings are used with `plot_all_NG_biomarkers`.

Usage

```
df_grouping_all_NG_biomarkers
```

Format

A data frame (tibble) with 2 rows and 2 columns:

version Biomarker platform version.

layout A data frame (tibble) defining layout:

- `machine_readable_name` Biomarker machine readable name, i.e. the one delivered with the csv data format.
- `group_custom` A character indication group titles.
- `column` An integer indicating the column number in a layout, see `plot_all_NG_biomarkers`.
- `page` An integer indicating the page number in a layout, see `plot_all_NG_biomarkers`.

See Also

Data frame `df_NG_biomarker_metadata` with information on the Nightingale Health Ltd. NMR-quantified blood biomarkers

`df_linear_associations` *Linear Associations of Blood Biomarkers to BMI, HOMA-IR and Fasting Glucose*

Description

A data frame containing cross-sectional associations of the Nightingale blood biomarkers to Body Mass Index (BMI), insulin resistance (log(HOMA-IR)) and fasting glucose. For these values a linear regression model was used, adjusted for age and sex.

Usage

```
df_linear_associations
```

Format

A data frame (tibble) with 687 rows and 5 columns:

name Blood biomarker name. Note: glucose is missing as the results are adjusted for this biomarker.

trait The response variable of the regression model, either BMI, log(HOMA-IR) or fasting glucose.

beta Linear regression coefficient β.

se Standard error.

pvalue P-value.

Details

"Values are beta-correlations from cross-sectional metabolite associations with BMI, log(HOMA-IR) and fasting glucose. For comparison of the patterns of associations, magnitudes are scaled to 1-SD in each of the outcomes (corresponding to 4.2 kg/m2 for BMI, 0.57 for log(HOMA-IR) and 0.56 mmol/l for glucose) per 1-SD log-transformed metabolite concentration. Results were adjusted for sex and age, and meta-analyzed for 11,896 individuals from the four cohorts. Error bars denote 95% confidence intervals; the large sample size and consistency across cohorts make confidence intervals narrow for the cross-sectional linear regression analyses." The values are shown in Figure S5 of A. V. Ahola-Olli et al. (2019): https://www.biorxiv.org/content/early/2019/01/08/513648

Source

These data are taken from the Supplementary material of A. V. Ahola-Olli et al. (2019). https://www.biorxiv.org/content/early/2019/01/08/513648

`df_logodds_associations`
Odds Ratios of Blood Biomarkers with Incident Type 2 Diabetes

Description

A data frame containing log odds ratios of the Nightingale blood biomarkers with risk for future type 2 diabetes.

Usage

```
df_logodds_associations
```

Format

A data frame (tibble) with 228 rows and 5 columns:

name Biomarker abbreviation.

study Short description of the association type, here the cohort name.

beta Log odds for incident type 2 diabetes.

se Standard error.

pvalue P-value.

n Sample size.

Details

"Values are odds ratios (95% confidence intervals) per 1-SD log-transformed metabolite concentration. Odds ratios were adjusted for sex, baseline age, BMI, and fasting glucose. YFS, Cardiovascular risk in Young Finns Study; NFBC, Northern Finland Birth Cohort." The values are shown in Figure S3 of A. V. Ahola-Olli et al. (2019): https://www.biorxiv.org/content/early/2019/01/08/513648

Source

These data are taken from the Supplementary material of A. V. Ahola-Olli et al. (2019). https://www.biorxiv.org/content/early/2019/01/08/513648

`df_NG_biomarker_metadata` *Information and Grouping for Nightingale's Blood Biomarkers*

Description

A data frame with information on the Nightingale Health Ltd. NMR-quantified blood biomarkers.

Usage

```
df_NG_biomarker_metadata
```

Format

A data frame (tibble) with 252 rows and 8 columns:

abbreviation Biomarker abbreviation, i.e. the one delivered with the xlsx data format.

machine_readable_name Biomarker machine readable name, i.e. the one delivered with the csv data format.

name Biomarker name.

alternative_names Alternative biomarker names.

description Biomarker description.

group The group the biomarker belongs to.

subgroup The subgroup the biomarker belongs to.

unit Unit is deprecated. Use the units delivered with biomarkers.

Source

Nightingale Health Ltd. https://nightingalehealth.com/

`discovery_regression` *Exploratory Analysis*

Description

Fit multiple regression models in one go.

Usage

```
discovery_regression(
  df_long,
  model = c("lm", "glm", "coxph"),
  formula,
  key = key,
  predictor = predictor,
  verbose = FALSE
)
```

Arguments

`df_long`	a data frame in a long format that contains a `key` column with the names of the variables for which to estimate the measures of effect, e.g. Nightingale NMR biomarker names, a `value` column with the numeric values of the respective keys and other columns with the response and other variables to adjust for. Note: You may use `tidyr::gather` to collapse your data frame to key-value pairs; it is recommended to `dplyr::select` only the variables that you want to go into your model, in order to avoid memory overheads in case of large datasets.
`model`	a character, setting the type of model to fit on the input data frame. Must be one of `'lm'`, `'glm'` or `'coxph'`, for linear, logistic and cox proportional hazards regression, respectively.
`formula`	a formula object. For the case of models `'lm'` and `'glm'` it must be an object of class `stats::formula` (or one that can be coerced to that class): a symbolic description of the model to be fitted. The details of model specification are given under 'Details' in the documentation of `stats::formula`. For the case of model `'coxph'`, the formula object must have the response on the left of a ~ operator, and the rest of the terms on the right. The response must be a survival object, as returned by the `Surv` function.
`key`	the name of the `key` variable.
`predictor`	the name of the variable for which (adjusted) univariate associations are estimated. This is stated here in order to individuate the predictor of interest over many, possible cofactors that are also present in `formula`.
`verbose`	logical (default FALSE). If TRUE it prints a message with the names of the predictor and outcome. This may come in handy when, for example, fitting multiple outcomes.

Value

A data frame with the following columns: a character variable with the same name as the `key` parameter and numeric variables `estimate`, `se` and `pvalue` with the values of the respective variables of the linear model. If `predictor` is factor, additional column `term` is returned indicating factor level of `predictor`

Author(s)

Maria Kalimeri, Emmi Tikkanen, Juuso Parkkinen, Vilma Jagerroos

Examples

```
library(magrittr)

# Linear Regression Example

# We will use the simulated demo data that come with the package,
# ggforestplot::df_demo_metabolic_data

# Extract the names of the NMR biomarkers for discovery analysis
nmr_biomarkers <- dplyr::intersect(
  ggforestplot::df_NG_biomarker_metadata$machine_readable_name,
  colnames(df_demo_metabolic_data)
)

# Select only variables to be used for the model and collapse to a long
# format
df_long <-
  df_demo_metabolic_data %>%
  # Select only model variables
  dplyr::select(tidyselect::all_of(nmr_biomarkers), gender, BMI) %>%
  # Log-transform and scale biomarkers
  dplyr::mutate_at(
    .vars = dplyr::vars(tidyselect::all_of(nmr_biomarkers)),
    .funs = ~ .x %>% log1p() %>% scale() %>% as.numeric()
  ) %>%
  # Collapse to a long format
  tidyr::gather(
    key = machine_readable_name,
    value = biomarkervalue,
    tidyselect::all_of(nmr_biomarkers)
  )

df_assoc_per_biomarker <-
  discovery_regression(
    df_long = df_long,
    model = "lm",
    formula =
      formula(
        biomarkervalue ~ BMI + factor(gender)
      ),
    key = machine_readable_name,
    predictor = BMI
  )

# Filter Nightingale metadata data frame for biomarkers of interest
df_grouping <-
  ggforestplot::df_NG_biomarker_metadata %>%
  dplyr::filter(group %in% "Fatty acids")
```

```
# Join the association data frame with the group data above
df <-
  df_assoc_per_biomarker %>%
  # use right_join, with df_grouping on the right, to preserve the order of
  # biomarkers it specifies.
  dplyr::right_join(., df_grouping, by = "machine_readable_name")

# Draw a forestplot of the results
ggforestplot::forestplot(
  df = df,
  name = name,
  estimate = estimate,
  se = se,
  pvalue = pvalue,
  psignif = 0.001,
  xlab = "1-SD increment in biomarker concentration
per 1-SD increment in BMI",
  title = "Associations of fatty acids to BMI",
  logodds = TRUE
)

# Logistic Regression Example

# Extract names of relevant NMR biomarkers
nmr_biomarkers <- dplyr::intersect(
  ggforestplot::df_NG_biomarker_metadata$machine_readable_name,
  colnames(df_demo_metabolic_data)
)

# Select only variables to be used for the model and
# collapse to a long format
df_long <-
  df_demo_metabolic_data %>%
  # Select only model variables (avoid memory overhead)
  dplyr::select(
    tidyselect::all_of(nmr_biomarkers),
    gender,
    incident_diabetes,
    BMI,
    baseline_age
  ) %>%
  dplyr::mutate_at(
    .vars = dplyr::vars(tidyselect::all_of(nmr_biomarkers)),
    .funs = ~ .x %>% log1p() %>% scale() %>% as.numeric()
  ) %>%
  # Collapse to a long format
  tidyr::gather(
    key = machine_readable_name,
    value = biomarkervalue,
    tidyselect::all_of(nmr_biomarkers)
  )
```

```
df_assoc_per_biomarker_gender <-
  discovery_regression(
    df_long = df_long,
    model = "glm",
    formula =
      formula(
        incident_diabetes ~ biomarkervalue + factor(gender) + BMI + baseline_age
      ),
    key = machine_readable_name,
    predictor = biomarkervalue
  )

# Filter Nightingale metadata data frame for biomarkers of interest
df_grouping <-
  ggforestplot::df_NG_biomarker_metadata %>%
  dplyr::filter(
    group %in% "Cholesterol",
    !(machine_readable_name %in% c("HDL2_C", "HDL3_C"))
  )

# Join the association data frame with the group data above
df <-
  df_assoc_per_biomarker_gender %>%
  # use right_join, with df_grouping on the right, to preserve the order of
  # biomarkers it specifies.
  dplyr::right_join(., df_grouping, by = "machine_readable_name")

# Draw a forestplot of the results
ggforestplot::forestplot(
  df = df,
  name = name,
  estimate = estimate,
  se = se,
  pvalue = pvalue,
  psignif = 0.001,
  xlab = "Odds ratio for incident type 2 diabetes (95% CI)
per 1-SD increment in biomarker concentration",
  title = "Cholesterol and risk of future type 2 diabetes",
  logodds = TRUE
)
```

`forestplot` *Draw a Forestplot of Measures of Effects*

Description

Visualize multiple measures of effect with their confidence intervals in a vertical layout.

Usage

```
forestplot(
  df,
  name = name,
  estimate = estimate,
  se = se,
  pvalue = NULL,
  colour = NULL,
  shape = NULL,
  logodds = FALSE,
  psignif = 0.05,
  ci = 0.95,
  ...
)
```

Arguments

`df`	A data frame with the data to plot. It must contain at least three variables, a character column with the names to be displayed on the y-axis (see parameter `name`), a numeric column with the value (or the log of the value) to display (see parameter `estimate`) and a numeric value with the corresponding standard errors (see parameter `se`). It may contain additional columns, e.g. the corresponding p-values (see parameter `pvalue`) in which case, in conjuction with the threshold given in `psignif`, the non-significant results will be displayed as hollow points. Other variables may be used as aesthetics to define the colour and the shape of the points to be plotted.
`name`	the variable in `df` that contains the y-axis names. This argument is automatically quoted and evaluated in the context of the `df` data frame. See Note.
`estimate`	the variable in `df` that contains the values (or log of values) to be displayed. This argument is automatically quoted and evaluated in the context of the `df` data frame. See Note.
`se`	the variable in the `df` data frame that contains the standard error values. This argument is automatically quoted and evaluated in the context of the `df` data frame. See Note.
`pvalue`	the variable in `df` that contains the p-values. Defaults to NULL. When explicitly defined, in conjuction with the p-value threshold provided in the `psignif`, the non-significant entries will be drawn as hollow points. This argument is automatically quoted and evaluated in the context of the `df` data frame. See Note.
`colour`	the variable in `df` by which to colour the different groups of points. This argument is automatically quoted and evaluated in the context of the `df` data frame. See Note.
`shape`	the variable in `df` by which to shape the different groups of points. This argument is automatically quoted and evaluated in the context of the `df` data frame. See Note.

`logodds`	logical (defaults to FALSE) specifying whether the `estimate` parameter should be treated as log odds/hazards ratio (TRUE) or not (FALSE). When `logodds` = TRUE the estimates and corresponding confidence intervals will be exponentiated and a log scale will be used for the x-axis.
`psignif`	numeric, defaults to 0.05. The p-value threshold for statistical significance. Entries with larger than `psignif` will be drawn with a hollow point.
`ci`	A number between 0 and 1 (defaults to 0.95) indicating the type of confidence interval to be drawn.
`...`	`ggplot2` graphical parameters such as `title`, `ylab`, `xlab`, `xtickbreaks` etc. to be passed along.

Value

A `ggplot` object.

Note

See `vignette(programming, package = "dplyr")` for an introduction to non-standard evaluation.

Author(s)

Maria Kalimeri, Ilari Scheinin, Vilma Jagerroos

Examples

```
library(magrittr)

# Linear associations
# Get subset of example data frame
df_linear <-
  df_linear_associations %>%
  dplyr::arrange(name) %>%
  dplyr::filter(dplyr::row_number() <= 30)

# Forestplot
forestplot(
  df = df_linear,
  estimate = beta,
  logodds = FALSE,
  colour = trait,
  xlab = "1-SD increment in cardiometabolic trait
  per 1-SD increment in biomarker concentration"
)

# Log odds ratios
df_logodds <-
  df_logodds_associations %>%
  dplyr::arrange(name) %>%
  dplyr::filter(dplyr::row_number() <= 30) %>%
```

```
  # Set the study variable to a factor to preserve order of appearance
  # Set class to factor to set order of display.
  dplyr::mutate(
    study = factor(
      study,
      levels = c("Meta-analysis", "NFBC-1997", "DILGOM", "FINRISK-1997", "YFS")
    )
  )

# Forestplot
forestplot(
  df = df_logodds,
  estimate = beta,
  logodds = TRUE,
  colour = study,
  xlab = "Odds ratio for incident type 2 diabetes (95% CI)
  per 1-SD increment in biomarker concentration"
)

# For the latter, if you want to restrain the x-axis and crop the large
# errorbar for Acetate you may add the following coord_cartesian layer
forestplot(
  df = df_logodds,
  estimate = beta,
  logodds = TRUE,
  colour = study,
  shape = study,
  xlab = "Odds ratio for incident type 2 diabetes (95% CI)
  per 1-SD increment in biomarker concentration",
  xlim = c(0.5, 2.2),
  # You can explicitly define x-tick breaks
  xtickbreaks = c(0.5, 0.8, 1.0, 1.2, 1.5, 2.0)
) +
  # You may also want to add a manual shape to mark meta-analysis with a
  # diamond shape
  ggplot2::scale_shape_manual(
    values = c(23L, 21L, 21L, 21L, 21L),
    labels = c("Meta-analysis", "NFBC-1997", "DILGOM", "FINRISK-1997", "YFS")
  )
```

`geom_effect` *Horizontal Study Effects with Confidence Intervals*

Description

Builds a custom version of `geom_pointrangeh`.

Usage

```
geom_effect(
```

```
  mapping = NULL,
  data = NULL,
  stat = "identity",
  position = ggstance::position_dodgev(height = 0.5),
  ...,
  fatten = 2,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

mapping	Set of aesthetic mappings created by aes() or aes_(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).
stat	The statistical transformation to use on the data for this layer, as a string.
position	Position adjustment, either as a string, or the result of a call to a position adjustment function.
...	Other arguments passed on to layer(). These are often aesthetics, used to set an aesthetic to a fixed value, like colour = "red" or size = 3. They may also be parameters to the paired geom/stat.
fatten	A multiplicative factor used to increase the size of the middle bar in geom_crossbar() and the middle point in geom_pointrange().
na.rm	If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.
show.legend	logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display.
inherit.aes	If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. borders().

Author(s)

Ilari Scheinin

Examples

```
library(ggplot2)
library(magrittr)
df <-
  # Use built-in demo dataset
  df_linear_associations %>%
    # Arrange by name in order to filter the first few biomarkers for more
    # than one studies
    dplyr::arrange(name) %>%
    # Estimate confidence intervals
    dplyr::mutate(
      xmin = beta - qnorm(1 - (1 - 0.95) / 2) * se,
      xmax = beta + qnorm(1 - (1 - 0.95) / 2) * se
    ) %>%
    # Select only first 30 rows (10 biomarkers)
    dplyr::filter(dplyr::row_number() <= 30) %>%
    # Add a logical variable for statistical significance
    dplyr::mutate(filled = pvalue < 0.001)

g <-
  ggplot(data = df, aes(x = beta, y = name)) +
  # And point+errorbars
  geom_effect(
    ggplot2::aes(
      xmin = xmin,
      xmax = xmax,
      colour = trait,
      shape = trait,
      filled = filled
    ),
    position = ggstance::position_dodgev(height = 0.5)
  )
print(g)

# Add custom theme, horizontal gray rectangles, vertical line to signify the
# NULL point, custom color palettes.
g <-
  g +
  # Add custom theme
  theme_forest() +
  # Add striped background
  geom_stripes() +
  # Add vertical line at null point
  geom_vline(
    xintercept = 0,
    linetype = "solid",
    size = 0.4,
    colour = "black"
```

```
  )
print(g)
```

`geom_stripes` *Alternating Background Colour*

Description

Add alternating background color along the y-axis. The geom takes default aesthetics `odd` and `even` that receive color codes. The codes would preferably be in the 8-hex ARGB format to allow for transparency if the geom is meant to be used as visual background.

Usage

```
geom_stripes(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  position = "identity",
  ...,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

`mapping`	Set of aesthetic mappings created by `aes()` or `aes_()`. If specified and `inherit.aes = TRUE` (the default), it is combined with the default mapping at the top level of the plot. You must supply `mapping` if there is no plot mapping.
`data`	The data to be displayed in this layer. There are three options: If `NULL`, the default, the data is inherited from the plot data as specified in the call to `ggplot()`. A `data.frame`, or other object, will override the plot data. All objects will be fortified to produce a data frame. See `fortify()` for which variables will be created. A `function` will be called with a single argument, the plot data. The return value must be a `data.frame`, and will be used as the layer data. A `function` can be created from a `formula` (e.g. `~ head(.x, 10)`).
`stat`	The statistical transformation to use on the data for this layer, as a string.
`position`	Position adjustment, either as a string, or the result of a call to a position adjustment function.
`...`	Other arguments passed on to `layer()`. These are often aesthetics, used to set an aesthetic to a fixed value, like `colour = "red"` or `size = 3`. They may also be parameters to the paired geom/stat.

show.legend — logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display.

inherit.aes — If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. borders().

Author(s)

Ilari Scheinin

Examples

```
library(ggplot2)
library(magrittr)
df <-
  # Use built-in demo dataset
  df_linear_associations %>%
    # Arrange by name in order to filter the first few biomarkers for more
    # than one studies
    dplyr::arrange(name) %>%
    # Estimate confidence intervals
    dplyr::mutate(
      xmin = beta - qnorm(1 - (1 - 0.95) / 2) * se,
      xmax = beta + qnorm(1 - (1 - 0.95) / 2) * se
    ) %>%
    # Select only first 30 rows (10 biomarkers)
    dplyr::filter(dplyr::row_number() <= 30) %>%
    # Add a logical variable for statistical significance
    dplyr::mutate(filled = pvalue < 0.001)

g <-
  ggplot(data = df, aes(x = beta, y = name)) +
  # And point+errorbars
  geom_effect(
    ggplot2::aes(
      xmin = xmin,
      xmax = xmax,
      colour = trait,
      shape = trait,
      filled = filled
    ),
    position = ggstance::position_dodgev(height = 0.5)
  )
print(g)

# Add custom theme, horizontal gray rectangles, vertical line to signify the
# NULL point, custom color palettes.
g <-
  g +
```

```
  # Add custom theme
  theme_forest() +
  # Add striped background
  geom_stripes(odd = "#33333333", even = "#00000000") +
  # Add vertical line at null point
  geom_vline(
    xintercept = 0,
    linetype = "solid",
    size = 0.4,
    colour = "black"
  )
print(g)
```

`ng_colour` *Nightingale's colours*

Description

`ng_colour()` returns Nightingale's colours' hex codes. `display_ng_colours()` displays the avaliable colours, with their names and hex codes, in a vertical layout.

Usage

```
ng_colour(...)

display_ng_colours()
```

Arguments

...	Character names of Nightingale's colours.

Author(s)

Ilari Scheinin

Examples

```
# Display Nightingale's colours
display_ng_colours()

# Request for the hex code of colour 'dark pesto'
ng_colour("dark pesto")
```

`ng_palette_d` *Nightingale's colour palettes*

Description

`ng_palette_d()` and `ng_palette_c()` (respectively for discrete and continuous palettes) return functions that take an integer argument (the required number of colours) and return a character vector of colours' hex codes. In addition, the functions also recognize the `viridis` palettes: "magma" (or "A"), "inferno" ("B"), "plasma" ("C"), "viridis" ("D"), or "cividis" ("D").

Usage

```
ng_palette_d(name = "all", reverse = FALSE)

ng_palette_c(name = "magma", reverse = FALSE, ...)

display_ng_palettes()
```

Arguments

`name`	Character name of the Nightingale (or viridis) colour palette.
`reverse`	Boolean indicating whether the palette should be reversed.
`...`	Additional arguments to pass to `colorRampPalette()`

Author(s)

Ilari Scheinin

Examples

```
#' # Display Nightingale's colour palettes
display_ng_palettes()

# Get 4 colours along the spectrum of the nwr palette
ng_palette_d("nwr")(4)

# Notice that the discrete palette "light", cannot return more than 5 colours
ng_palette_d("light")(6)
```

plot_all_NG_biomarkers *Print a Forestplot for All Nightingale Biomarkers*

Description

Save a forestplot of all Nightingale biomarker associations in a 2-page, predefined layout (utilizes `forestplot`).

Usage

```
plot_all_NG_biomarkers(
  df,
  machine_readable_name = machine_readable_name,
  name = NULL,
  estimate = estimate,
  se = se,
  pvalue = NULL,
  colour = NULL,
  shape = NULL,
  logodds = FALSE,
  psignif = 0.05,
  ci = 0.95,
  filename = NULL,
  paperwidth = 15,
  paperheight = sqrt(2) * paperwidth,
  xlims = NULL,
  layout = "2020",
  ...
)
```

Arguments

`df`	A data frame with the data to plot. It must contain at least three variables, a character column with the names to be displayed on the y-axis (see parameter `name`), a numeric column with the value (or the log of the value) to display (see parameter `estimate`) and a numeric value with the corresponding standard errors (see parameter `se`). It may contain additional columns, e.g. the corresponding p-values (see parameter `pvalue`) in which case, in conjuction with the threshold given in `psignif`, the non-significant results will be displayed as hollow points. Other variables may be used as aesthetics to define the colour and the shape of the points to be plotted.
`machine_readable_name`	the variable in df containing the machine readable names of Nightingale blood biomarkers. I.e. the names in this variable must be the same as in the `machine_readable_name` variable of `df_NG_biomarker_metadata`.

	(This argument is automatically quoted and evaluated in the context of the `df` data frame.)
`name`	the variable in `df` that contains the y-axis names. If NULL, names from `df_NG_biomarker_metadata` are used. This argument is automatically quoted and evaluated in the context of the `df` data frame. See Note.
`estimate`	the variable in `df` that contains the values (or log of values) to be displayed. This argument is automatically quoted and evaluated in the context of the `df` data frame. See Note.
`se`	the variable in the `df` data frame that contains the standard error values. This argument is automatically quoted and evaluated in the context of the `df` data frame. See Note.
`pvalue`	the variable in `df` that contains the p-values. Defaults to NULL. When explicitly defined, in conjuction with the p-value threshold provided in the `psignif`, the non-significant entries will be drawn as hollow points. This argument is automatically quoted and evaluated in the context of the `df` data frame. See Note.
`colour`	the variable in `df` by which to colour the different groups of points. This argument is automatically quoted and evaluated in the context of the `df` data frame. See Note.
`shape`	the variable in `df` by which to shape the different groups of points. This argument is automatically quoted and evaluated in the context of the `df` data frame. See Note.
`logodds`	logical (defaults to FALSE) specifying whether the `estimate` parameter should be treated as log odds/hazards ratio (TRUE) or not (FALSE). When `logodds` = TRUE the estimates and corresponding confidence intervals will be exponentiated and a log scale will be used for the x-axis.
`psignif`	numeric, defaults to 0.05. The p-value threshold for statistical significance. Entries with larger than `psignif` will be drawn with a hollow point.
`ci`	A number between 0 and 1 (defaults to 0.95) indicating the type of confidence interval to be drawn.
`filename`	a character string giving the name of the file.
`paperwidth`	page width in inches
`paperheight`	page height in inches
`xlims`	NULL or a numeric vector of length 2 specifying the common x limits across all biomarker subgroups.
`layout`	one of the predefined layouts in `df_grouping_all_NG_biomarkers` or custom layout tibble following the example of predifined layouts
`...`	`ggplot2` graphical parameters such as `title`, `ylab`, `xlab`, `xtickbreaks` etc. to be passed along.

Details

The function uses a custom grouping specified by `df_grouping_all_NG_biomarkers`. The input `df` and `df_grouping_all_NG_biomarkers` are joined by `machine_readable_name`, while another `df` variable may be used for y-axis labels, defined in `name` input parameter.

Value

If filename is NULL, a list of plot objects (one for each page in layout) is returned.

Author(s)

Maria Kalimeri, Ilari Scheinin, Vilma Jagerroos

Examples

```
## Not run:
# Join the built-in association demo dataset with a variable that contains
# the machine readable names of Nightingale biomarkers. (Note: if you
# have built your association data frame using the Nightingale CSV result file,
# then your data frame should already contain machine readable names.)
df <-
  df_linear_associations %>%
  left_join(
    select(
      df_NG_biomarker_metadata,
      name,
      machine_readable_name
    ),
    by = "name"
  )

# Print effect sizes for Nightingale biomarkers in a 2-page pdf
plot_all_NG_biomarkers(
  df = df,
  machine_readable_name = machine_readable_name,
  # Notice that when name is not defined explicitly, names from
  # df_NG_biomarker_metadata are used
  estimate = beta,
  se = se,
  pvalue = pvalue,
  colour = trait,
  filename = "biomarker_linear_associations.pdf",
  xlab = "1-SD increment in BMI
per 1-SD increment in biomarker concentration",
  layout = "2016"
)

# Custom layout can also be provided
layout <- df_NG_biomarker_metadata %>%
  dplyr::filter(
    .data$group == "Fatty acids",
    .data$machine_readable_name %in% df$machine_readable_name
  ) %>%
  dplyr::mutate(
    group_custom = .data$subgroup,
    column = dplyr::case_when(
      .data$group_custom == "Fatty acids" ~ 1,
      .data$group_custom == "Fatty acid ratios" ~ 2
```

```
    ),
    page = 1
  ) %>%
  dplyr::select(
    .data$machine_readable_name,
    .data$group_custom,
    .data$column,
    .data$page
  )

plot_all_NG_biomarkers(
  df = df,
  machine_readable_name = machine_readable_name,
  # Notice that when name is not defined explicitly, names from
  # df_NG_biomarker_metadata are used
  estimate = beta,
  se = se,
  pvalue = pvalue,
  colour = trait,
  xlab = "1-SD increment in BMI
per 1-SD increment in biomarker concentration",
  layout = layout
)

# log odds for type 2 diabetes
df <-
  df_logodds_associations %>%
  left_join(
    select(
      df_NG_biomarker_metadata,
      name,
      machine_readable_name
    ),
    by = "name"
  ) %>%
  # Set the study variable to a factor to preserve order of appearance
  # Set class to factor to set order of display.
  dplyr::mutate(
    study = factor(
      study,
      levels = c("Meta-analysis", "NFBC-1997", "DILGOM", "FINRISK-1997", "YFS")
    )
  )

# Print effect sizes for Nightingale biomarkers in a 2-page pdf
plot_all_NG_biomarkers(
  df = df,
  machine_readable_name = machine_readable_name,
  # Notice that when name is not defined explicitly, names from
  # df_NG_biomarker_metadata are used
  estimate = beta,
  se = se,
  pvalue = pvalue,
```

```
  colour = study,
  logodds = TRUE,
  filename = "biomarker_t2d_associations.pdf",
  xlab = "Odds ratio for incident type 2 diabetes (95% CI)
per 1-SD increment in metabolite concentration",
  layout = "2016",
  # Restrict limits as some studies are very weak and they take over the
  # overall range.
  xlims = c(0.5, 3.2)
)

## End(Not run)
```

`scale_colour_ng_d` *Colour scale constructor for Nightingale colours*

Description

Colour scale constructor for Nightingale colours

Usage

```
scale_colour_ng_d(..., palette = "all", reverse = FALSE, aesthetics = "colour")

scale_fill_ng_d(..., palette = "all", reverse = FALSE, aesthetics = "fill")

scale_colour_ng_c(
  ...,
  palette = "magma",
  reverse = FALSE,
  values = NULL,
  space = "Lab",
  na.value = "grey50",
  guide = "colourbar",
  aesthetics = "colour"
)

scale_fill_ng_c(
  ...,
  palette = "magma",
  reverse = FALSE,
  values = NULL,
  space = "Lab",
  na.value = "grey50",
  guide = "colourbar",
  aesthetics = "fill"
)
```

Arguments

...	Additional arguments passed to discrete_scale() or continuous_scale() to control name, limits, breaks, labels and so forth.
palette	Character name of the Nightingale (or viridis) colour palette.
reverse	Boolean indicating whether the palette should be reversed.
aesthetics	Character string or vector of character strings listing the name(s) of the aesthetic(s) that this scale works with. This can be useful, for example, to apply colour settings to the 'colour' and 'fill' aesthetics at the same time, via 'aesthetics = c("colour", "fill")'.
values	if colours should not be evenly positioned along the gradient this vector gives the position (between 0 and 1) for each colour in the colours vector. See rescale() for a convenience function to map an arbitrary range to between 0 and 1.
space	colour space in which to calculate gradient. Must be "Lab" - other values are deprecated.
na.value	Missing values will be replaced with this value.
guide	A function used to create a guide or its name. See guides() for more info.

Author(s)

Ilari Scheinin

Examples

```
# Example taken from ggplot2::scale_colour_discrete()
dsamp <- ggplot2::diamonds[sample(nrow(ggplot2::diamonds), 1000), ]
d <- ggplot2::ggplot(dsamp, ggplot2::aes(carat, price)) +
  ggplot2::geom_point(ggplot2::aes(colour = clarity)) +
  ggforestplot::scale_colour_ng_d()
print(d)
```

---

theme_forest *Forestplot Theme*

Description

A custom theme used in forestplot that builts upon theme_minimal.

Usage

```
theme_forest(
  base_size = 13,
  base_line_size = base_size/22,
  base_rect_size = base_size/22
)
```

Arguments

`base_size`	base font size
`base_line_size`	base size for line elements
`base_rect_size`	base size for rect elements

Author(s)

Maria Kalimeri

See Also

`forestplot`, `geom_effect`, `geom_stripes`

Index

∗ **datasets**
    df_demo_metabolic_data, 2
    df_grouping_all_NG_biomarkers, 3
    df_linear_associations, 4
    df_logodds_associations, 5
    df_NG_biomarker_metadata, 6

aes(), *14*, *16*
aes_(), *14*, *16*

borders(), *14*, *17*

colorRampPalette(), *19*
continuous_scale(), *25*

df_demo_metabolic_data, 2
df_grouping_all_NG_biomarkers, 3, *21*
df_linear_associations, 4
df_logodds_associations, 5
df_NG_biomarker_metadata, *3*, 6, *20*, *21*
discovery_regression, 6
discrete_scale(), *25*
display_ng_colours *(ng_colour)*, 18
display_ng_palettes *(ng_palette_d)*, 19

evaluated, *11*, *21*

forestplot, 10, *20*, *25*, *26*
formula, *7*
fortify(), *14*, *16*

gather, *7*
geom_effect, 13, *26*
geom_pointrangeh, *13*
geom_stripes, 16, *26*
ggplot(), *14*, *16*
guides(), *25*

layer(), *14*, *16*

ng_colour, 18
ng_palette_c *(ng_palette_d)*, 19
ng_palette_d, 19

plot_all_NG_biomarkers, *3*, 20

quoted, *11*, *21*

rescale(), *25*

scale_colour_ng_c *(scale_colour_ng_d)*, 24
scale_colour_ng_d, 24
scale_fill_ng_c *(scale_colour_ng_d)*, 24
scale_fill_ng_d *(scale_colour_ng_d)*, 24
select, *7*
Surv, *7*

theme_forest, 25
theme_minimal, *25*

viridis, *19*