Package: osprey (via r-universe)

October 16, 2024

Type Package

Title R Package to Create TLGs

Version 0.1.16.9015

Date 2024-09-16

Description Community effort to collect TLG code and create a catalogue.

License Apache License 2.0 | file LICENSE

URL https://insightsengineering.github.io/osprey/, https://github.com/insightsengineering/osprey/

BugReports https://github.com/insightsengineering/osprey/issues

Depends dplyr (>= 0.8.0), ggplot2 (>= 3.5.0), R (>= 3.6)

Imports checkmate (>= 2.1.0), cowplot, DescTools (>= 0.99.53), grDevices, grid, gridExtra, gtable (>= 0.3.4), methods, rlang (>= 1.1.0), stats, stringr (>= 1.4.1), tibble (>= 2.0.0), tidyr (>= 1.0.0)

Suggests knitr (>= 1.42), nestcolor (>= 0.1.0), rmarkdown (>= 2.23), tern (>= 0.7.10), testthat (>= 2.0)

Config/Needs/verdepcheck tidyverse/dplyr, tidyverse/ggplot2, mllg/checkmate, wilkelab/cowplot, AndriSignorell/DescTools, baptiste/gridExtra, r-lib/gtable, r-lib/rlang, tidyverse/stringr, tidyverse/tibble, tidyverse/tidyr, yihui/knitr, insightsengineering/nestcolor, rstudio/rmarkdown, insightsengineering/tern, r-lib/testthat

Config/Needs/website insightsengineering/nesttemplate

Encoding UTF-8

Language en-US

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

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

RemoteUrl https://github.com/insightsengineering/osprey

RemoteRef HEAD

RemoteSha 13264769f50465eee8d9ee08468cf8afe806f919

Contents

`as_pdf` *Output decorated grob (`gTree`) objects as PDF*

Description

This is an utility function to output a decorated grob (`gTree`) object

Usage

```
as_pdf(grobs, outpath, pagesize = "letter.landscape")
```

Arguments

`grobs`	a grid grob (`gTree`) object, optionally `NULL` if only a grob with the decoration should be shown.
`outpath`	specify full path to output pdf to `BCE` or `BEE`
`pagesize`	name of `pagesize` (print size) and orientation, accepted values include `"a4.landscape"`, `"a4.portrait"`, `"letter.portrait"` and `"letter.landscape"` (default)

Value

a pdf file

Author(s)

Chendi Liao (liaoc10) <chendi.liao@roche.com>

See Also

grobs2pdf()

Examples

```
## Not run:
library(ggplot2)

g <- list(
  ggplotGrob(
    ggplot(iris, aes(x = Sepal.Length, y = Sepal.Width, color = Species)) +
      geom_point()
  ),
  ggplotGrob(
    ggplot(iris, aes(x = Sepal.Length, y = Petal.Length, color = Species)) +
      geom_point()
  ),
  ggplotGrob(
    ggplot(iris, aes(x = Sepal.Length, y = Petal.Width, color = Species)) +
      geom_point()
  )
)

# output to pdf
as_pdf(g, "~/example_aspdf1.pdf")

## End(Not run)
```

---

create_flag_vars *create* AE *overview flags*

Description

create AE overview flags

Usage

```
create_flag_vars(
  df,
  fatal = AESDTH == "Y",
  serious = AESER == "Y",
```

```
  serious_withdrawl = AESER == "Y" & grepl("DRUG WITHDRAWN", AEACN),
 serious_modified = AESER == "Y" & grepl("DRUG (INTERRUPTED|INCREASED|REDUCED)", AEACN),
  serious_related = AESER == "Y" & AEREL == "Y",
  withdrawl = grepl("DRUG WITHDRAWN", AEACN),
  modified = grepl("DRUG (INTERRUPTED|INCREASED|REDUCED)", AEACN),
  related = AEREL == "Y",
  related_withdrawl = AEREL == "Y" & grepl("DRUG WITHDRAWN", AEACN),
 related_modified = AEREL == "Y" & grepl("DRUG (INTERRUPTED|INCREASED|REDUCED)", AEACN),
  ctc35 = AETOXGR %in% c("3", "4", "5"),
  ...
)
```

Arguments

df	data frame of AE
fatal	AE with fatal outcome derivation
serious	Serious AE derivation.
serious_withdrawl	Serious AE leading to withdrawal derivation
serious_modified	Serious AE leading to dose modification derivation
serious_related	Related Serious AE derivation
withdrawl	AE leading to withdrawal derivation
modified	AE leading to dose modification derivation
related	Related AE derivation
related_withdrawl	Related AE leading to withdrawal derivation
related_modified	Related AE leading to dose modification derivation
ctc35	Grade 3-5 AE derivation
...	named expressions used to generate categories

Details

in this function, all flags are expressions calls, for simpler usage.

Examples

```
library(dplyr)

ADAE <- osprey::rADAE

# add additional dummy causality flags
ADAE <- ADAE %>%
  mutate(AEREL1 = (AEREL == "Y" & ACTARM == "A: Drug X")) %>%
  mutate(AEREL2 = (AEREL == "Y" & ACTARM == "B: Placebo"))
```

```
attr(ADAE[["AEREL1"]], "label") <- "AE related to A: Drug X"
attr(ADAE[["AEREL2"]], "label") <- "AE related to B: Placebo"

create_flag_vars(ADAE)
# create other flags
create_flag_vars(ADAE, `AENSER` = AESER != "Y")
# remove flags that are not needed
create_flag_vars(ADAE, fatal = NULL)
```

---

grobs2pdf *Decorate grob (*gTree*) objects then outputs as* IDM *compatible PDF*

Description

This is an utility function to decorated grob (gTree) object with titles and footnotes in accordance with IDM specification and export as PDF file with full path to program and the output for easy tracking and archiving.

Usage

```
grobs2pdf(
  grobs,
  titles,
  footnotes,
  progpath,
  outpath,
  fontsize = 9,
  pagesize = "letter.landscape"
)
```

Arguments

grobs	A grid grob (gTree) object, optionally NULL if only a grob with the decoration should be shown
titles	Vector of character strings. Vector elements are separated by a newline and strings are wrapped according to the page with
footnotes	Vector of character string. Same rules as for titles
progpath	Specify the full path to the R program that generate the grobs and the PDF
outpath	Specify full path to output pdf to BCE or BEE
fontsize	Base font size used in pdf, default set to 9. Font size for title is set to fontsize + 1 (default = 10) and for footnotes set to fontsize - 1 (default = 8)
pagesize	name of paper size and orientation, accepted values include "a4.landscape", "a4.portrait", "letter.portrait" and "letter.landscape" (default)

Value

a pdf file

Author(s)

Chendi Liao (liaoc10) <chendi.liao@roche.com>

Examples

```
## Not run:
library(ggplot2)

g <- list(
  ggplotGrob(
    ggplot(iris, aes(x = Sepal.Length, y = Sepal.Width, color = Species)) +
      geom_point()
  ),
  ggplotGrob(
    ggplot(iris, aes(x = Sepal.Length, y = Petal.Length, color = Species)) +
      geom_point()
  ),
  ggplotGrob(
    ggplot(iris, aes(x = Sepal.Length, y = Petal.Width, color = Species)) +
      geom_point()
  )
)

grobs2pdf(
  grobs = g,
  titles = "Visualization of Iris Data",
  footnotes = "This is a footnote",
  progpath = "~/example_prog.R",
  outpath = "~/example_grobs2pdf.pdf"
)

## End(Not run)
```

---

g_ae_sub *Adverse Event Category Plot*

Description

Draw adverse event category plot.

Usage

```
g_ae_sub(
  id,
  arm,
  arm_sl,
  subgroups,
  subgroups_sl,
```

```
  trt = levels(arm)[1],
  ref = levels(arm)[2],
  indent = 4,
  subgroups_levels = NULL,
  xmax = 0,
  conf_level = 0.95,
 diff_ci_method = c("wald", "waldcc", "ac", "score", "scorecc", "mn", "mee", "blj",
   "ha", "beal"),
  fontsize = 4,
  arm_n = FALSE,
  draw = TRUE
)
```

Arguments

id	(vector) contains subject identifier. Usually it is ADAE$USUBJID.
arm	(factor) vector that contains arm information in analysis data. For example, ADAE$ACTARMCD.
arm_sl	(vector) contains the subject level treatment variable. For example, ADSL$ACTARM.
subgroups	(data.frame) Variables to conduct analysis.
subgroups_sl	(data.frame) Subject level variables to conduct analysis. Usually from ADSL.
trt	(character) indicates the name of the treatment arm. Default is the second level of arm.
ref	(character) indicates the name of the reference arm. Default is the first level of arm.
indent	(numeric) non-negative integer where 0 means that the subgroup levels should not be indented
subgroups_levels	(list) A nested named list of variables to conduct analysis. The names of the nested lists are used to show as the label. The children lists should start with "Total" = variable label, followed by labels for each level of said variable. See example for reference.
xmax	(numeric) maximum range for the x-axis. x-axis range will be automatically assigned based on risk output when xmax is less than or equal to 0. xmax is 0 by default
conf_level	(numeric) the confidence interval level, default is 0.95.
diff_ci_method	(character) the method used to calculate confidence interval. Default is "wald". Possible choices are methods supported in BinomDiffCI.

fontsize	(numeric) font size for the plot. It is the size used in ggplot2 with default unit "mm", if you want "points" you will need to divide the point number by ggplot2:::.pt.
arm_n	(logical) whether to display the N in each arm.
draw	(logical) whether to draw the plot.

Details

there is no equivalent STREAM output

Value

grob object

Author(s)

Liming Li (Lil128) <liming.li@roche.com>

Examples

```
library(grid)
ADAE <- osprey::rADAE
ADSL <- osprey::rADSL

id <- ADAE$USUBJID
arm <- ADAE$ACTARMCD
arm_sl <- as.character(ADSL$ACTARMCD)
subgroups_sl <- ADSL[, c("SEX", "RACE", "STRATA1")]
subgroups <- ADAE[, c("SEX", "RACE", "STRATA1")]
subgroups_levels <- list(
  RACE = list(
    "Total" = "Race",
    "AMERICAN INDIAN OR ALASKA NATIVE" = "American",
    "WHITE" = "White",
    "ASIAN" = "Asian",
    "BLACK OR AFRICAN AMERICAN" = "African"
  ),
  STRATA1 = list(
    "Total" = "Strata",
    "A" = "TypeA",
    "B" = "TypeB",
    "C" = "Typec"
  ),
  SEX = list(
    "Total" = "Sex",
    "M" = "Male",
    "F" = "Female"
  )
)
```

```
# Example 1
p1 <- g_ae_sub(id,
  arm,
  arm_sl,
  subgroups,
  subgroups_sl,
  trt = "ARM A",
  ref = "ARM C",
  subgroups_levels = subgroups_levels,
  arm_n = FALSE
)
grid::grid.newpage()

# Example 2: display number of patients in each arm
p2 <- g_ae_sub(id,
  arm,
  arm_sl,
  subgroups,
  subgroups_sl,
  trt = "ARM A",
  ref = "ARM C",
  subgroups_levels = subgroups_levels,
  arm_n = TRUE
)
grid::grid.newpage()

# Example 3: preprocess data to only include treatment and control arm patients
trt <- "ARM A"
ref <- "ARM C"
ADAE <- osprey::rADAE
ADSL <- osprey::rADSL %>% filter(ACTARMCD %in% c(trt, ref))
id <- ADAE$USUBJID
arm <- ADAE$ACTARMCD
arm_sl <- as.character(ADSL$ACTARMCD)
subgroups_sl <- ADSL[, c("SEX", "RACE", "STRATA1")]
subgroups <- ADAE[, c("SEX", "RACE", "STRATA1")]
subgroups_levels <- list(
  RACE = list(
    "Total" = "Race",
    "AMERICAN INDIAN OR ALASKA NATIVE" = "American",
    "WHITE" = "White",
    "ASIAN" = "Asian",
    "BLACK OR AFRICAN AMERICAN" = "African"
  ),
  STRATA1 = list(
    "Total" = "Strata",
    "A" = "TypeA",
    "B" = "TypeB",
    "C" = "Typec"
  ),
  SEX = list(
    "Total" = "Sex",
    "M" = "Male",
```

```
    "F" = "Female"
  )
)
p3 <- g_ae_sub(id,
  arm,
  arm_sl,
  subgroups,
  subgroups_sl,
  trt,
  ref,
  subgroups_levels = subgroups_levels,
  arm_n = FALSE
)
```

`g_butterfly` *Butterfly Plot*

Description

The butterfly plot is often used in Early Development (ED) and is an opposed barplot that shows instances of AEs or # of patients by category separated by a dichotomization variable. Each bar can be color coded according to a variable of choice and sorted according to either alphabetical order or the maximum count.

Usage

```
g_butterfly(
  category,
  right_flag,
  left_flag,
  id = NULL,
  group_names = NULL,
  block_count = c("# of patients", "# of AEs"),
  block_color = NULL,
  facet_rows = NULL,
  x_label = block_count,
  y_label = "AE Derived Terms",
  legend_label = "AETOXGR",
  sort_by = c("count", "alphabetical", "right", "left"),
  show_legend = TRUE
)
```

Arguments

`category`	vector of y values
`right_flag`	vector of `logical` of the same length as `category`. used to filter `category` for the right side of the barplot. to maintain backward compatibility, a vector of 1s and 0s would also work.

left_flag	vector of logical of the same length as category. used to filter category for the left side of the barplot. to maintain backward compatibility, a vector of 1s and 0s would also work.
id	unique subject identifier variable.
group_names	string vector of length 2 with desired names of dichotomization variables required format : first name corresponds to the name of the right side second name corresponds to name of the left side default: will extract column names from group
block_count	string - what to count by (ex: # of AEs or # of patients)
block_color	vector - color coding of bar segments
facet_rows	vector defines what variable is used to split the plot into rows, default here is NULL
x_label	string of text for x axis label, default is block_count
y_label	string of text for y axis label, default is AE Derived Terms
legend_label	character for legend label, default is "AETOXGR"
sort_by	character string that defines the ordering of the class and term variables in the output table, options: "alphabetical", "count", "left", "right", default here is set to "count"
show_legend	logical(1) of whether color coding legend is included, default here is FALSE

Details

there is no equivalent STREAM output

Value

ggplot object

Author(s)

Carolyn Zhang (zhanc107) <carolyn.zhang@duke.edu>

Ting Qi (qit3) <qit3@gene.com>

Examples

```
library(dplyr)
library(nestcolor)

ADSL <- osprey::rADSL %>%
  select(USUBJID, STUDYID, SEX, ARM, RACE) %>%
  dplyr::filter(SEX %in% c("F", "M"))
ADAE <- osprey::rADAE %>% select(USUBJID, STUDYID, AEBODSYS, AETOXGR)

ANL <- left_join(ADAE, ADSL, by = c("STUDYID", "USUBJID"))
ANL <- ANL %>%
  dplyr::mutate(flag1 = ifelse(RACE == "ASIAN", 1, 0)) %>%
  dplyr::mutate(flag2 = ifelse(SEX == "M", 1, 0))
```

```
ANL <- na.omit(ANL)
# Example 1, # of AEs
g_butterfly(
  category = ANL$AEBODSYS,
  right_flag = ANL$flag1,
  left_flag = ANL$flag2,
  group_names = c("flag1 Asian", "flag2 M"),
  block_count = "# of AEs",
  block_color = ANL$AETOXGR,
  id = ANL$USUBJID,
  x_label = "# of AEs",
  y_label = "AE Body System",
  legend_label = "AETOXGR",
  sort_by = "count",
  show_legend = TRUE
)

# Example 2, # of patients with facet
g_butterfly(
  category = ANL$AEBODSYS,
  right_flag = ANL$flag1,
  left_flag = ANL$flag2,
  group_names = c("flag1 Asian", "flag2 M"),
  block_count = "# of patients",
  block_color = ANL$AETOXGR,
  facet_rows = ANL$ARM,
  id = ANL$USUBJID,
  x_label = "# of patients",
  y_label = "AE Derived Terms",
  legend_label = "AETOXGR",
  sort_by = "count",
  show_legend = TRUE
)
```

`g_events_term_id` *Events by Term Plot*

Description

This function plots commonly occurred events by number of unique subjects with events. It creates basic summary of events and compares event occurrences between comparison and reference arms, and can be used for events data such as Adverse Events.

Usage

```
g_events_term_id(
  term,
  id,
  arm,
  arm_N,
```

```
  ref = levels(arm)[1],
  trt = levels(arm)[2],
  sort_by = c("term", "riskdiff", "meanrisk"),
  rate_range = c(0, 1),
  diff_range = c(-1, 1),
  reversed = FALSE,
  conf_level = 0.95,
 diff_ci_method = c("wald", "waldcc", "ac", "score", "scorecc", "mn", "mee", "blj",
    "ha", "beal"),
  axis_side = c("left", "right"),
  color = c(getOption("ggplot2.discrete.colour"), "blue", "red")[1:2],
  shape = c(16, 17),
  fontsize = 4,
  draw = TRUE
)
```

Arguments

term	character or factor vector, or data.frame Represents events information. term can be a data.frame produced by create_flag_vars, with each column being a logical event indicator
id	(vector) contains subject identifier. Length of id must be the same as the length or number of rows of terms. Usually it is ADAE$USUBJID.
arm	(factor) vector that contains arm information in analysis data. For example, ADAE$ACTARMCD.
arm_N	(numeric vector) Contains information of the number of patients in the levels of arm. This is useful if there are patients that have no adverse events can be accounted for with this argument.
ref	character indicates the name of the reference arm. Default is the first level of arm.
trt	character indicates the name of the treatment arm. Default is the second level of arm.
sort_by	character indicates how each term is sorted in the plot. Choose from "term" for alphabetic terms, "riskdiff" for risk difference, and "meanrisk" for mean risk. Default is "term".
rate_range	Numeric vector of length 2. Range for overall rate display
diff_range	Numeric vector of length 2. Range for rate difference display
reversed	logical whether to reverse the sorting by sort_by. Default is FALSE.
conf_level	(numeric) the confidence interval level, default is 0.95.
diff_ci_method	(character) the method used to calculate confidence interval. Default is "wald". Possible choices are methods supported in BinomDiffCI.

axis_side	character the side of the axis label, "left" or "right". Default is "left".
color	Color for the plot. vector of length 2. Color for reference and treatment arms respectively. Default set to c("blue", "red").
shape	Shape for the plot. vector of length 2. Shape for reference and treatment arms respectively. Default set to c(16, 17) per scale_shape.
fontsize	(numeric) font size for the plot. It is the size used in ggplot2 with default unit "mm", if you want "points" you will need to divide the point number by ggplot2:::.pt.
draw	(logical) whether to draw the plot.

Details

there is no equivalent STREAM output

Value

grob object

Author(s)

Liming Li (Lil128) <liming.li@roche.com>

Molly He (hey59) <hey59@gene.com>

Examples

```
library(dplyr)
library(grid)
library(nestcolor)

ADSL <- osprey::rADSL
ADAE <- osprey::rADAE

# add additional dummy causality flags
ADAE <- ADAE %>%
  mutate(AEREL1 = (AEREL == "Y" & ACTARM == "A: Drug X")) %>%
  mutate(AEREL2 = (AEREL == "Y" & ACTARM == "B: Placebo"))
attr(ADAE[["AEREL1"]], "label") <- "AE related to A: Drug X"
attr(ADAE[["AEREL2"]], "label") <- "AE related to B: Placebo"

term <- ADAE$AEDECOD
id <- ADAE$USUBJID
arm <- ADAE$ACTARMCD
arm_N <- table(ADSL$ACTARMCD)
ref <- "ARM A"
trt <- "ARM C"

# Example 1
p1 <- g_events_term_id(
  term,
```

```
  id,
  arm,
  arm_N
)
grid::grid.newpage()
grid::grid.draw(p1)

# Example 2
p2 <- g_events_term_id(
  term,
  id,
  arm,
  arm_N,
  trt = trt,
  ref = ref,
  sort_by = "riskdiff",
  diff_ci_method = "ac",
  conf_level = 0.9
)
grid::grid.newpage()
grid::grid.draw(p2)

# Example 3
p3 <- g_events_term_id(
  term,
  id,
  arm,
  arm_N,
  sort_by = "meanrisk",
  axis_side = "right",
  fontsize = 5
)
grid::grid.newpage()
grid::grid.draw(p3)

# Example 4
term <- create_flag_vars(ADAE)
g_events_term_id(
  term,
  id,
  arm,
  arm_N,
  fontsize = 3
)
```

---

g_heat_bygrade *Heatmap by Grade*

Description

This function plots heatmap

Usage

```
g_heat_bygrade(
  id_var,
  exp_data,
  visit_var,
  ongo_var,
  anno_data,
  anno_var,
  heat_data,
  heat_color_var,
  heat_color_opt = NULL,
  conmed_data = NULL,
  conmed_var = NULL,
  conmed_color_opt = NULL,
  xlab = "Visit",
  title = NULL
)
```

Arguments

id_var	(character) name of the column that contains the unique subject identifier shared by all data Usually it is "USUBJID".
exp_data	(data.frame) exposure data. Usually it is ADEX.
visit_var	(character) name of the column that contains the analysis visit. Usually it is "AVISIT"
ongo_var	(character) name of the column in exp_data that contains the logical variable indicating whether the treatment is still ongoing. Usually it can be derived from EOSSTT
anno_data	(data.frame) annotation data that contains subject level characteristics. Usually it is ADSL
anno_var	(character) a vector of columns name(s) to include for the annotation
heat_data	(data.frame) data frame that contains the information needed for the text over heatmap Usually it is ADCM.
heat_color_var	(character) name of the column that contains the heat grade
heat_color_opt	optional, (character) a named vector that maps the names to heat colors
conmed_data	optional, (data.frame) concomitant medicine data. Usually it is ADCM default is NULL (no conmed plotted)
conmed_var	optional, (character) concomitant medicine variable name. Must be a column name in conmed_data when conmed_data is provided. default is NULL (no conmed plotted)

conmed_color_opt
: optional, (character)
vector of color name(s) to conmed_data

xlab
: optional, (character)
string to be shown as x-axis label, default is "Visit"

title
: (character)
string to be shown as title of the plot. default is NULL (no plot title is displayed)

Author(s)

Nina Qi (qit3) <qit3@gene.com>

Molly He (hey59) <hey59@gene.com>

Examples

```
library(dplyr)

ADSL <- osprey::rADSL %>% slice(1:30)
ADEX <- osprey::rADEX %>% filter(USUBJID %in% ADSL$USUBJID)
ADAE <- osprey::rADAE %>% filter(USUBJID %in% ADSL$USUBJID)
ADCM <- osprey::rADCM %>% filter(USUBJID %in% ADSL$USUBJID)
# function to derive AVISIT from ADEX
add_visit <- function(data_need_visit) {
  visit_dates <- ADEX %>%
    filter(PARAMCD == "DOSE") %>%
    distinct(USUBJID, AVISIT, ASTDTM) %>%
    group_by(USUBJID) %>%
    arrange(ASTDTM) %>%
    mutate(next_vis = lead(ASTDTM), is_last = ifelse(is.na(next_vis), TRUE, FALSE)) %>%
    rename(this_vis = ASTDTM)
  data_visit <- data_need_visit %>%
    select(USUBJID, ASTDTM) %>%
    left_join(visit_dates, by = "USUBJID", relationship = "many-to-many") %>%
    filter(ASTDTM > this_vis & (ASTDTM < next_vis | is_last == TRUE)) %>%
    left_join(data_need_visit, relationship = "many-to-many")
  return(data_visit)
}
# add AVISIT in ADAE and ADCM
ADAE <- add_visit(ADAE)
ADCM <- add_visit(ADCM)
exp_data <- ADEX %>%
  filter(PARCAT1 == "INDIVIDUAL") %>%
  group_by(USUBJID) %>%
  # create a shorter subject identifier
  mutate(SUBJ = utils::tail(strsplit(USUBJID, "-")[[1]], n = 1)) %>%
  mutate(ongo_var = (EOSSTT == "ONGOING")) %>%
  ungroup()
anno_data <- ADSL %>%
  select(SEX, COUNTRY, USUBJID) %>%
  group_by(USUBJID) %>%
  mutate(SUBJ = utils::tail(strsplit(USUBJID, "-")[[1]], n = 1)) %>%
```

```
  ungroup() %>%
  select(-USUBJID)
heat_data <- ADAE %>%
  select(USUBJID, AVISIT, AETOXGR) %>%
  group_by(USUBJID) %>%
  mutate(SUBJ = utils::tail(strsplit(USUBJID, "-")[[1]], n = 1)) %>%
  ungroup() %>%
  select(-USUBJID)
heat_color_opt <- c(
  "No Event" = "gray90",
  "1" = "lightsteelblue1",
  "2" = "steelblue1",
  "3" = "steelblue4",
  "4" = "maroon",
  "5" = "brown4"
)
cmdecod_label <- attr(ADCM[["CMDECOD"]], "label")
ADCM <- ADCM %>%
  filter(
    CMDECOD == "medname A_1/3" | CMDECOD == "medname A_2/3" | CMDECOD == "medname A_3/3"
  ) %>%
  mutate(CMDECOD = factor(CMDECOD, levels = unique(CMDECOD)))
attr(ADCM[["CMDECOD"]], "label") <- cmdecod_label
conmed_data <- ADCM %>%
  group_by(USUBJID) %>%
  mutate(SUBJ = utils::tail(strsplit(USUBJID, "-")[[1]], n = 1))
# example plotting conmed
g_heat_bygrade(
  id_var = "SUBJ",
  exp_data,
  visit_var = "AVISIT",
  ongo_var = "ongo_var",
  anno_data,
  anno_var = c("SEX", "COUNTRY"),
  heat_data,
  heat_color_var = "AETOXGR",
  heat_color_opt,
  conmed_data,
  conmed_var = "CMDECOD",
  conmed_color_opt = c("green", "green3", "green4")
)
# example not plotting conmed
g_heat_bygrade(
  id_var = "SUBJ",
  exp_data,
  visit_var = "AVISIT",
  ongo_var = "ongo_var",
  anno_data,
  anno_var = c("SEX", "COUNTRY"),
  heat_data,
  heat_color_var = "AETOXGR",
  heat_color_opt
)
```

g_hy_law *Hy's Law Plot*

Description

A scatter plot typically used to display maximum total bilirubin values (TBL) versus maximum alanine transaminase (ALT) values.

Usage

```
g_hy_law(
  id,
  term,
  aval,
  arm,
  term_selected,
  anrhi,
  folds = c(3, 2),
  text = c("Normal Range", "Hyperbilirubinemia", "Possible Hy's Law Range",
    "Temple's Corollary"),
 caption = paste("Maximum values are those maximum values that occur post-baseline",
    "(no time constraints and not necessarily concurrent events)."),
  title = "Max. Total Bilirubin vs. Max. Alanine Aminotransferase",
  xlab = "Maximum Alanine Aminotransferase (/ULN)",
  ylab = "Maximum Total Bilirubin (/ULN)"
)
```

Arguments

id	unique subject identifier.
term	the term of the observation.
aval	analysis value.
arm	treatment arm. Used as fill color in the plot.
term_selected	string vector of length 2 - the two terms selected to be used in the plot. First value corresponds to parameter plotted on the x-axis. Second value corresponds to that plotted on the y-axis.
anrhi	the high limit of normal range.
folds	numeric vector of length two. Indicates the position of the reference lines to be drawn. Default c(3, 2) corresponds to a line at position 3 on the x-axis and 2 on the y-axis.
text	string vector of length four with the label to be shown on each quadrant. First value corresponds to label shown in the bottom left quadrant. Subsequent values move through the graph clockwise.

caption	string of text for footnote. Details of methodology can be shown here.
title	string of text for plot title.
xlab	string of text for x axis label.
ylab	string of text for y axis label.

Details

This graphic is based upon the eDISH (evaluation of Drug Induced Serious Hepatotoxicity) plot of Watkins et. al. in a 2008 publication from Hepatology. Maximum values are defined as the maximum post-baseline value at any time during the entire length of the observation period. Both axes are in log scale to control for the dispersion of the data. The values are plotted in 'times upper limit of normal' where a value of 1 would mean that the result was normal. Any value above or below 1 would be considered above the upper limit or normal or below the upper limit of normal respectively. For instance, a value of 3 would be read as '3 times the upper limit of normal'. Reference lines are included to determine various states, based upon clinical interpretation of the values and includes the following:

- Hyperbilirubinemia TBL at least 2 xULN and ALT less than 3 xULN
- Normal Range TBL <= 1 xULN and ALT <= 1 xULN
- Temple's Corollary TBL <= 1 xULN and ALT at least 3 xULN
- Possible Hy's Law TBL at least 2 xULN and ALT at least 3 xULN

This plot can easily be adjusted for other lab parameters and reference ranges as needed. Consultation with a clinical expert to determine which associations would be clinically meaningful and how to interpret those associations is recommended.

There is no equivalent STREAM output.

Value

plot object

Author(s)

Katie Withycombe (withycok) <katie.withycombe@roche.com>

Amy Franklin (frankla4) <amy.franklin@roche.com>

William Holmes (holmesw) <william.holmes@roche.com>

Examples

```
library(dplyr)
library(nestcolor)

# Note: CRP is being used in place of Bilirubin here because this is the only available data
adsl <- osprey::rADSL
adlb <- osprey::rADLB %>% mutate(ANRHI = 50)

# Example 1, - Hy's law template (3 and 2 X ULN)
g_hy_law(
```

```
  id = adlb$USUBJID,
  term = adlb$PARAMCD,
  aval = adlb$AVAL,
  arm = adlb$ARM,
  term_selected = c("ALT", "CRP"),
  anrhi = adlb$ANRHI,
  folds = c(3, 2),
 text = c("Normal Range", "Hyperbilirubinemia", "Possible Hy's Law Range", "Temple's Corollary"),
  caption = paste(
    "Maximum values are those maximum values that occur",
    "post-baseline (no time constraints and not necessarily concurrent events)."
  ),
  title = "Max. Total Bilirubin vs. Max. Alanine Aminotransferase",
  xlab = "Maximum Alanine Aminotransferase (/ULN)",
  ylab = "Maximum Total Bilirubin (/ULN)"
)

# Example 2, - change the quadrant lines and labels
g_hy_law(
  id = adlb$USUBJID,
  term = adlb$PARAMCD,
  aval = adlb$AVAL,
  arm = adlb$ARM,
  term_selected = c("ALT", "CRP"),
  anrhi = adlb$ANRHI,
  folds = c(10, 15),
  text = c("Quadrant 1", "Quadrant 2", "Quadrant 3", "Quadrant 4"),
  caption = paste(
    "Maximum values are those maximum values that occur",
    "post-baseline (no time constraints and not necessarily concurrent events)."
  ),
  title = "Max. Total Bilirubin vs. Max. Alanine Aminotransferase",
  xlab = "Maximum Alanine Aminotransferase (/ULN)",
  ylab = "Maximum Total Bilirubin (/ULN)"
)
```

`g_patient_profile` *Patient Profile Plot*

Description

Patient profile plot provides detailed information for a specific subject participating in the study. The plot includes relevant data for one subject that can help correlate adverse events, response, concomitant medications, exposure, and laboratory. The plotting of patient profile is modularized, with each domain plot generated by function `patient_domain_profile`. This `g_patient_profile` function assembles all requested domain plots into one patient profile. `ADSL`, `ADEX`, `ADAE`, `ADRS`, `ADCM` and `ADLB` data must be provided. The plot output will not include domains with data unspecified

Usage

```
g_patient_profile(
  ex = NULL,
  ae = NULL,
  rs = NULL,
  cm = NULL,
  lb = NULL,
  arrow_end_day,
  xlim = c(-28, 365),
  xlab = "Study Day",
  title = "Patient Profile"
)
```

Arguments

ex	list may contain • data dataframe for ADEX domain dataset • var vector to identify each lane of ADEX domain plot
ae	list may contain • data dataframe for ADAE domain dataset • var vector to identify each lane of ADAE plot • line_col factor vector to specify color for segments of ADAE plot • line_col_legend string to be displayed as line color legend title of ADAE plot • line_col_opt aesthetic values to map line color values of ADAE plot (named vector to map color values to each name). If not NULL, please make sure this contains all possible values for line_col values, otherwise color will be assigned by ggplot default, please note that NULL needs to be specified
rs	list may contain • data dataframe for ADRS domain dataset • var vector to identify each lane of ADRS domain plot
cm	list may contain • data dataframe for ADCM domain dataset • var vector to identify each lane of ADCM domain plot
lb	list may contain • data dataframe for ADLB domain dataset • var vector to identify each lane of ADLB domain plot
arrow_end_day	numeric value indicates the end of arrow when arrows are requested
xlim	numeric vector for x-axis limit that will be shared by all domain plots, default is xlim = c(-28, 365)
xlab	string to be shown as x-axis label, default is "Study Day"
title	string to be shown as title of the plot, default is "Patient Profile"

Value

plot object

Author(s)

Xuefeng Hou (houx14) <houx14@gene.com>

Molly He (hey59) <hey59@gene.com>

Ting Qi (qit3) <qit3@gene.com>

See Also

patient_domain_profile

Examples

```
library(dplyr)
library(nestcolor)

# ADSL
ADSL <- osprey::rADSL %>%
  filter(USUBJID == rADSL$USUBJID[1]) %>%
  mutate(
    TRTSDT = as.Date(TRTSDTM),
    max_date = max(as.Date(LSTALVDT), as.Date(DTHDT), na.rm = TRUE),
    max_day = as.numeric(as.Date(max_date) - as.Date(TRTSDT)) + 1
  ) %>%
  select(USUBJID, STUDYID, TRTSDT, max_day)


# ADEX
ADEX <- osprey::rADEX %>%
  select(USUBJID, STUDYID, ASTDTM, PARCAT2, AVAL, AVALU, PARAMCD)
ADEX <- left_join(ADSL, ADEX, by = c("USUBJID", "STUDYID"))

ADEX <- ADEX %>%
  filter(PARAMCD == "DOSE") %>%
  arrange(PARCAT2, PARAMCD) %>%
  mutate(diff = c(0, diff(AVAL, lag = 1))) %>%
  mutate(Modification = case_when(
    diff < 0 ~ "Decrease",
    diff > 0 ~ "Increase",
    diff == 0 ~ "None"
  )) %>%
  mutate(ASTDT_dur = as.numeric(
    as.Date(substr(as.character(ASTDTM), 1, 10)) -
      as.Date(TRTSDT) + 1
  ))

# ADAE
ADAE <- osprey::rADAE %>%
  select(USUBJID, STUDYID, AESOC, AEDECOD, AESER, AETOXGR, AEREL, ASTDY, AENDY)
```

```
ADAE <- left_join(ADSL, ADAE, by = c("USUBJID", "STUDYID"))

# ADRS
ADRS <- osprey::rADRS %>%
  select(USUBJID, STUDYID, PARAMCD, PARAM, AVALC, AVAL, ADY, ADTM)
ADRS <- left_join(ADSL, ADRS, by = c("USUBJID", "STUDYID"))

# ADCM
ADCM <- osprey::rADCM %>%
  select(USUBJID, STUDYID, ASTDTM, AENDTM, CMDECOD, ASTDY, AENDY)
ADCM <- left_join(ADSL, ADCM, by = c("USUBJID", "STUDYID"))

# ADLB
ADLB <- osprey::rADLB %>%
  select(
    USUBJID, STUDYID, LBSEQ, PARAMCD, BASETYPE, ADTM,
    ADY, ATPTN, AVISITN, LBTESTCD, ANRIND
  )
ADLB <- left_join(ADSL, ADLB, by = c("USUBJID", "STUDYID"))

ADLB <- ADLB %>%
  group_by(USUBJID) %>%
  mutate(ANRIND = factor(ANRIND, levels = c("LOW", "NORMAL", "HIGH")))

# Example Patient Profile plot 5 domains
g_patient_profile(
  ex = list(
    data = ADEX,
    var = ADEX$PARCAT2
  ),
  ae = list(
    data = ADAE,
    var = ADAE$AEDECOD,
    line_col = factor(ADAE$AESER),
    line_col_legend = "Serious",
    line_col_opt = c("Y" = "red", "N" = "blue")
  ),
  rs = list(
    data = ADRS,
    var = ADRS$PARAMCD
  ),
  cm = list(
    data = ADCM,
    var = ADCM$CMDECOD
  ),
  lb = list(
    data = ADLB,
    var = ADLB$LBTESTCD
  ),
  arrow_end_day = ADSL$max_day,
  xlim = c(-28, ADSL$max_day),
  xlab = "Study Day",
  title = paste("Patient Profile: ", ADSL$USUBJID)
```

```
)

# Example Patient Profile plot without ADCM and ADLB
g_patient_profile(
  ex = list(
    data = ADEX,
    var = ADEX$PARCAT2
  ),
  ae = list(
    data = ADAE,
    var = ADAE$AEDECOD,
    line_col = factor(ADAE$AESER),
    line_col_legend = "Serious",
    line_col_opt = c("Y" = "red", "N" = "blue")
  ),
  rs = list(
    data = ADRS,
    var = ADRS$PARAMCD
  ),
  arrow_end_day = ADSL$max_day,
  xlim = c(-28, ADSL$max_day),
  xlab = "Study Day",
  title = paste("Patient Profile: ", ADSL$USUBJID)
)
```

---

g_spiderplot *Spider Plot*

Description

Spider plot is often used in Early Development (ED) and displays individual patient plot of an endpoint over time by group.

Usage

```
g_spiderplot(
  marker_x,
  marker_id,
  marker_y,
  line_colby = NULL,
  line_color_opt = NULL,
  marker_shape = NULL,
  marker_shape_opt = NULL,
  marker_size = 3,
  datalabel_txt = NULL,
  facet_rows = NULL,
  facet_columns = NULL,
  vref_line = NULL,
  href_line = NULL,
```

```
  x_label = "Time (Days)",
  y_label = "Change (%) from Baseline",
  show_legend = FALSE
)
```

Arguments

marker_x	vector of x values (must be in sorted order)
marker_id	vector to group the points together (default should be USUBJID)
marker_y	vector of y values
line_colby	vector defines by what variable plot is color coded, default here is NULL
line_color_opt	vector defines line color, default here is NULL
marker_shape	vector defines by what variable points are shape coded, , default here is NULL
marker_shape_opt	vector defines marker shape code, default here is NULL
marker_size	size of markers in plot, default here is NULL
datalabel_txt	list defines text (at last time point) and flag for an arrow annotation: • (per defined variable) elements must be labeled txt_ann/mrkr_all/mrkr_ann. • txt_ann text annotation next to final data point (for text annotation) • mrkr_all vector of ID's (for annotation marker) • mrkr_ann vector of ID's (subset of mrkr_all) where arrow is desired to indicate any study interim points. Default here is NULL.
facet_rows	dataframe defines what variable is used to split the plot into rows, default here is NULL.
facet_columns	dataframe defines what variable is used to split the plot into columns, default here is NULL.
vref_line	value defines vertical line overlay (can be a vector), default here is NULL.
href_line	value defines horizontal line overlay (can be a vector), default here is NULL.
x_label	string of text for x axis label, default is time.
y_label	string of text for y axis label, default is % change.
show_legend	boolean of whether marker legend is included, default here is FALSE.

Details

there is no equivalent STREAM output

Value

ggplot object

Author(s)

Carolyn Zhang (zhanc107) <carolyn.zhang@duke.edu>

Examples

```
# simple example
library(dplyr)
library(nestcolor)

ADTR <- osprey::rADTR %>% select(STUDYID, USUBJID, ADY, AVISIT, CHG, PCHG, PARAMCD)
ADSL <- osprey::rADSL %>% select(STUDYID, USUBJID, RACE, SEX, ARM)
ANL <- left_join(ADTR, ADSL, by = c("STUDYID", "USUBJID"))
ANL <- ANL %>%
  dplyr::filter(PARAMCD == "SLDINV" & AVISIT != "POST-BASELINE MINIMUM") %>%
  dplyr::filter(RACE %in% c("WHITE", "ASIAN")) %>%
  group_by(USUBJID) %>%
  dplyr::arrange(ADY) %>%
  dplyr::mutate(
    CHG = ifelse(AVISIT == "Screening", 0, CHG),
    PCHG = ifelse(AVISIT == "Screening", 0, PCHG)
  )
ANL$USUBJID <- substr(ANL$USUBJID, 14, 18)

# Plot 1 - default color and shape mapping
g_spiderplot(
  marker_x = ANL$ADY,
  marker_id = ANL$USUBJID,
  marker_y = ANL$PCHG,
  line_colby = ANL$USUBJID,
  marker_shape = ANL$USUBJID,
  # marker_size = 5,
  datalabel_txt = list(txt_ann = ANL$USUBJID),
  # facet_rows = data.frame(sex = ANL$SEX),
  # facet_columns = data.frame(arm = ANL$ARM),
  vref_line = c(42, 86),
  href_line = c(-20, 20),
  x_label = "Time (Days)",
  y_label = "Change (%) from Baseline",
  show_legend = TRUE
)

# Plot 2 - with line color mapping
g_spiderplot(
  marker_x = ANL$AVISIT,
  marker_id = ANL$USUBJID,
  marker_y = ANL$CHG,
  line_colby = ANL$RACE,
  line_color_opt = c("WHITE" = "red", "ASIAN" = "blue"),
  marker_shape = ANL$USUBJID,
  x_label = "Visit",
  y_label = "Change from Baseline",
  show_legend = TRUE
)
```

g_swimlane Swimlane *Plot*

Description

Swimlane plot is often used in Early Development (ED) and displays individual patient bar plot with markers of events and patient level annotation

Usage

```
g_swimlane(
  bar_id,
  bar_length,
  sort_by = NULL,
  col_by = NULL,
  marker_id = NULL,
  marker_pos = NULL,
  marker_shape = NULL,
  marker_shape_opt = NULL,
  marker_color = NULL,
  marker_color_opt = NULL,
  anno_txt = NULL,
  xref_line = NULL,
  xtick_at = waiver(),
  xlab,
  title
)
```

Arguments

bar_id	vector of IDs to identify each bar
bar_length	numeric vector to be plotted as length for each bar
sort_by	vector to sort bars
col_by	vector to color bars
marker_id	vector of IDs to identify markers within each bar. Default is the same as bar_id.
marker_pos	numeric vector to specify position for each marker point
marker_shape	vector to specify shape for markers
marker_shape_opt	aesthetic values to map shape values (named vector to map shape values to each name)
marker_color	vector to specify color for markers
marker_color_opt	aesthetic values to map shape values (named vector to map shape values to each name)

anno_txt	dataframe of subject-level variables to be displayed as annotation on the left
xref_line	numeric vector to plot reference lines
xtick_at	optional break interval of bar length axis
xlab	label for bar length
title	string to be displayed as plot title

Value

plot object

Author(s)

Ting Qi (qit3) <qit3@gene.com>

Examples

```
# Example 1
library(dplyr)
library(nestcolor)

ADSL <- osprey::rADSL[1:20, ]
ADRS <- filter(rADRS, PARAMCD == "OVRINV")
ANL <- left_join(ADSL, ADRS, by = c("STUDYID", "USUBJID"), multiple = "all")
anno_txt <- ADSL[, c("ARMCD", "SEX")]

g_swimlane(
  bar_id = ADSL$USUBJID,
  bar_length = as.integer(ADSL$TRTEDTM - ADSL$TRTSDTM),
  sort_by = ADSL$ARM,
  col_by = ADSL$ARM,
  marker_id = ANL$USUBJID,
  marker_pos = ANL$ADY,
  marker_shape = ANL$AVALC,
  marker_shape_opt = c("CR" = 16, "PR" = 17, "SD" = 18, "PD" = 15, "NE" = 4),
  marker_color = NULL,
  marker_color_opt = NULL,
  anno_txt = anno_txt,
  xref_line = c(50, 100),
  xtick_at = waiver(),
  xlab = "Time from First Treatment (Day)",
  title = "Swimlane Plot"
)

# Example 2
library(dplyr)
library(nestcolor)

ADSL <- osprey::rADSL[1:20, ]
ADRS <- osprey::rADRS

anno_txt_vars <- c("ARMCD", "SEX", "COUNTRY")
```

```
anno_txt <- ADSL[, anno_txt_vars]

# markers from ADRS
ADRS <- dplyr::filter(ADRS, PARAMCD == "OVRINV") %>% select(USUBJID, ADY, AVALC)

# markers from ADSL - discontinuation
ADS <- ADSL %>%
  dplyr::filter(EOSSTT == "Discontinued" | DCSREAS != "") %>%
  select(USUBJID, EOSDY, DCSREAS) %>%
  dplyr::rename(ADY = EOSDY, AVALC = DCSREAS)

# combine ADRS with ADS records as one data for markers and join with ADSL
ANL <- inner_join(ADSL, rbind(ADRS, ADS), by = "USUBJID", multiple = "all")

g_swimlane(
  bar_id = sub(".*-", "", ADSL$USUBJID),
  bar_length = as.integer(ADSL$TRTEDTM - ADSL$TRTSDTM),
  sort_by = NULL,
  col_by = ADSL$ARMCD,
  marker_id = sub(".*-", "", ANL$USUBJID),
  marker_pos = ANL$ADY,
  marker_shape = ANL$AVALC,
  marker_shape_opt = c(
    "CR" = 16, "PR" = 17, "SD" = 18, "PD" = 15, "NE" = 0,
   "Adverse Event" = 7, "Death" = 8, "Physician Decision" = 9, "Progressive Disease" = 10,
    "Symptomatic Deterioation" = 11, "Withdrawal by Subject" = 12
  ),
  marker_color = ANL$AVALC,
  marker_color_opt = c(
    "CR" = "green", "PR" = "blue", "SD" = "yellow", "PD" = "red",
   "NE" = "grey", "Adverse Event" = "orange", "Death" = "black", "Physician Decision" = "navy",
    "Progressive Disease" = "purple", "Symptomatic Deterioation" = "cyan",
    "Withdrawal by Subject" = "darkred"
  ),
  anno_txt = anno_txt,
  xref_line = c(50, 100),
  xtick_at = waiver(),
  xlab = "Time from First Treatment (Day)",
  title = "Swimlane Plot"
)
```

`g_waterfall` *Waterfall Plot*

Description

Waterfall plot is often used in Early Development (ED) to present each individual patient's best response to a particular drug based on a parameter.

Usage

```
g_waterfall(
  bar_id,
  bar_height,
  sort_by = NULL,
  col_by = NULL,
  bar_color_opt = NULL,
  anno_txt = NULL,
  href_line = NULL,
  facet_by = NULL,
  show_datavalue = TRUE,
  add_label = NULL,
  gap_point = NULL,
  ytick_at = 20,
  y_label = "Best % Change from Baseline",
  title = "Waterfall Plot"
)
```

Arguments

bar_id	(vector) contains IDs to identify each bar
bar_height	numeric vector to be plotted as height of each bar
sort_by	(vector) used to sort bars, default is NULL in which case bars are ordered by decreasing height
col_by	(vector) used to color bars, default is NULL in which case bar_id is taken if the argument bar_color_opt is provided
bar_color_opt	(vector) aesthetic values to map color values (named vector to map color values to each name). If not NULL, please make sure this contains all possible values for col_by values, otherwise default ggplot color will be assigned, please note that NULL needs to be specified in this case
anno_txt	(dataframe) contains subject-level variables to be displayed as annotation below the waterfall plot, default is NULL
href_line	(numeric vector) to plot horizontal reference lines, default is NULL
facet_by	(vector) to facet plot and annotation table, default is NULL
show_datavalue	(boolean) controls whether value of bar height is shown, default is TRUE
add_label	(vector) of one subject-level variable to be added to each bar except for bar_height, default is NULL

gap_point (numeric)
value for adding bar break when some bars are significantly higher than others, default is NULL

ytick_at (numeric)
optional bar height axis interval, default is 20

y_label (string)
label for bar height axis, default is "Best % Change from Baseline"

title (string)
displayed as plot title, default is "Waterfall Plot"

Value

plot object

Author(s)

Xuefeng Hou (houx14) <houx14@gene.com>

Ting Qi (qit3) <qit3@gene.com>

Examples

```
library(tidyr)
library(dplyr)
library(nestcolor)

g_waterfall(
  bar_id = letters[1:3], bar_height = c(3, 5, -1),
  bar_color_opt = c("red", "green", "blue")
)

# Example 1
ADSL <- osprey::rADSL[1:15, ]
ADRS <- osprey::rADRS %>%
  filter(USUBJID %in% ADSL$USUBJID)
ADTR <- osprey::rADTR %>%
  filter(USUBJID %in% ADSL$USUBJID) %>%
  select(USUBJID, PCHG) %>%
  group_by(USUBJID) %>%
  slice(which.min(PCHG))

TR_SL <- inner_join(ADSL, ADTR, by = "USUBJID", multiple = "all")

SUB_ADRS <- ADRS %>%
  filter(PARAMCD == "BESRSPI" | PARAMCD == "INVET") %>%
  select(USUBJID, PARAMCD, AVALC, AVISIT, ADY) %>%
  spread(PARAMCD, AVALC)

ANL <- TR_SL %>%
  left_join(SUB_ADRS, by = "USUBJID", multiple = "all") %>%
  mutate(TRTDURD = as.integer(TRTEDTM - TRTSDTM) + 1)
```

```
anno_txt_vars <- c("TRTDURD", "BESRSPI", "INVET", "SEX", "BMRKR2")

g_waterfall(
  bar_height = ANL$PCHG,
  bar_id = sub(".*-", "", ANL$USUBJID),
  col_by = ANL$SEX,
  sort_by = ANL$ARM,
  # bar_color_opt = c("F" = "red", "M" = "green", "U" = "blue"),
  anno_txt = ANL[, anno_txt_vars],
  facet_by = NULL,
  href_line = c(-30, 20),
  add_label = ANL$BESRSPI,
  ytick_at = 20,
  gap_point = NULL,
  show_datavalue = TRUE,
  y_label = "Best % Change from Baseline",
  title = "Waterfall Plot"
)

# Example 2 facetting
anno_txt_vars <- c("BESRSPI", "INVET")

g_waterfall(
  bar_id = sub(".*-", "", ANL$USUBJID),
  bar_height = ANL$PCHG,
  sort_by = ANL$COUNTRY,
  col_by = ANL$SEX,
  bar_color_opt = c("F" = "tomato", "M" = "skyblue3", "U" = "darkgreen"),
  anno_txt = ANL[, anno_txt_vars],
  facet_by = ANL$STRATA2,
  href_line = c(-30, 20),
  add_label = ANL$BESRSPI,
  ytick_at = 20,
  gap_point = 260,
  y_label = "Best % Change from Baseline",
  title = "Waterfall Plot"
)

# Example 3 extreme value
ANL$PCHG[3] <- 99
ANL$PCHG[5] <- 199
ANL$PCHG[7] <- 599
ANL$BESRSPI[3] <- "PD"
ANL$BESRSPI[5] <- "PD"
ANL$BESRSPI[7] <- "PD"

g_waterfall(
  bar_id = sub(".*-", "", ANL$USUBJID),
  bar_height = ANL$PCHG,
  sort_by = ANL$ARM,
  col_by = ANL$SEX,
  bar_color_opt = c("F" = "tomato", "M" = "skyblue3", "U" = "darkgreen"),
```

```
  anno_txt = ANL[, anno_txt_vars],
  facet_by = NULL,
  href_line = c(-30, 20),
  add_label = ANL$BESRSPI,
  ytick_at = 20,
  gap_point = 260,
  y_label = "Best % Change from Baseline",
  title = "Waterfall Plot"
)
```

`patient_domain_profile` *Patient Domain Profile*

Description

Patient domain profile provides information for a specific subject that participated in the study. The plot includes relevant data for one subject in a user specified domain, including adverse events (`ADAE`), response (`ADRS`), concomitant medications (`ADCM`), exposure (`ADEX`), and laboratory (`ADLB`).

Usage

```
patient_domain_profile(
  domain = NULL,
  var_names,
  marker_pos,
  arrow_end,
  xtick_at = waiver(),
  line_col_list = NULL,
  line_width = 1,
  arrow_size = 0.1,
  no_enddate_extention = 0,
  marker_col_list = NULL,
  marker_shape_list = NULL,
  show_days_label = TRUE,
  xlim = c(-28, max(marker_pos) + 5),
  xlab = NULL,
  show_title = TRUE,
  title = NULL
)
```

Arguments

`domain`	string of domain name to be shown as y-axis label, default is `NULL` (no y-axis label shown)
`var_names`	character vector to identify each lane

marker_pos — Depending on the domain, this can be

- marker position numeric vector for domains ADEX, ADLB, and ADRS
- numeric data frame with two columns, start and end time marker position, for domains ADAE and ADCM

arrow_end — numeric value indicates the end of arrow when arrows are requested

xtick_at — numeric vector with the locations of the x-axis tick marks

line_col_list — a list may contain

- line_col: factor vector to specify color for segments , default is NULL (no line color is specified)
- line_col_opt aesthetic values to map color values (named vector to map color values to each name). If not NULL, please make sure this contains all possible values for line_col values, otherwise color will be assigned by hcl.colors
- line_col_legend: a string to be displayed as line color legend title when line_col is specified, default is NULL (no legend title is displayed)

line_width — numeric value for segment width, default is line_width = 1

arrow_size — numeric value for arrow size, default is arrow_size = 0.1

no_enddate_extention — numeric value for extending the arrow when end date is missing for ADAE or ADCM domain. Default is no_enddate_extention = 0.

marker_col_list — a list may contain

- marker_col a factor vector to specify color for markers, default is NULL (no color markers is specified)
- marker_col_opt aesthetic values to map color values (named vector to map color values to each name) If not NULL, please make sure this contains all possible values for marker_col values, otherwise color will be assigned by hcl.colors
- marker_col_legend a string to be displayed as marker color legend title, default is NULL (no legend title is displayed)

marker_shape_list — a list may contain

- marker_shape factor vector to specify shape for markers, default is NULL (no shape marker is specified)
- marker_shape_opt aesthetic values to map shape values (named vector to map shape values to each name). If not NULL, please make sure this contains all possible values for marker_shape values, otherwise shape will be assigned by ggplot default
- marker_shape_legend string to be displayed as marker shape legend title, default is NULL (no legend title is displayed)

show_days_label
: boolean value for showing y-axis label, default is TRUE

xlim
: numeric vector for x-axis limit, default is xlim = c(-28, max(marker_pos) + 5)

xlab
: string to be shown as x-axis label, default is "Study Day"

show_title
: boolean value for showing title of the plot, default is TRUE

title
: string to be shown as title of the plot, default is NULL (no plot title is displayed)

Value

plot object

Author(s)

Xuefeng Hou (houx14) <houx14@gene.com>

Tina Cho (chot) <tina.cho@roche.com>

Molly He (hey59) <hey59@gene.com>

Ting Qi (qit3) <qit3@gene.com>

Examples

```
library(dplyr)

# ADSL
ADSL <- osprey::rADSL %>%
  filter(USUBJID == rADSL$USUBJID[1]) %>%
  mutate(
    TRTSDT = as.Date(TRTSDTM),
    max_date = max(as.Date(LSTALVDT), as.Date(DTHDT), na.rm = TRUE),
    max_day = as.numeric(as.Date(max_date) - as.Date(TRTSDT)) + 1
  ) %>%
  select(USUBJID, STUDYID, TRTSDT, max_day)



# Example 1 Exposure "ADEX"
ADEX <- osprey::rADEX %>%
  select(USUBJID, STUDYID, ASTDTM, PARCAT2, AVAL, AVALU, PARAMCD)
ADEX <- left_join(ADSL, ADEX, by = c("USUBJID", "STUDYID"))
ADEX <- ADEX %>%
  filter(PARAMCD == "DOSE") %>%
  arrange(PARCAT2, PARAMCD) %>%
  mutate(diff = c(0, diff(AVAL, lag = 1))) %>%
  mutate(
    Modification = case_when(
      diff < 0 ~ "Decrease",
      diff > 0 ~ "Increase",
      diff == 0 ~ "None"
    )
  ) %>%
```

```
  mutate(
    ASTDT_dur = as.numeric(
      as.Date(
        substr(as.character(ASTDTM), 1, 10)
      ) - as.Date(TRTSDT) + 1
    )
  )

p1 <- patient_domain_profile(
  domain = "Exposure (ADEX)",
  var_names = ADEX$PARCAT2,
  marker_pos = ADEX$ASTDT_dur,
  arrow_end = ADSL$max_day,
  xtick_at = waiver(),
  line_col_list = NULL,
  line_width = 1,
  arrow_size = 0.1,
  no_enddate_extention = 0,
  marker_col_list = list(
    marker_col = factor(ADEX$Modification),
    marker_col_opt = c("Increase" = "red", "Decrease" = "green", "None" = "blue"),
    marker_col_legend = NULL
  ),
  marker_shape_list = list(
    marker_shape = factor(ADEX$Modification),
    marker_shape_opt = c("Increase" = 24, "Decrease" = 25, "None" = 23),
    marker_shape_legend = "Dose Modification"
  ),
  show_days_label = TRUE,
  xlim = c(-28, ADSL$max_day),
  xlab = "Study Day",
  title = paste("Patient Profile: ", ADSL$USUBJID)
)
p1

# Example 2 Adverse Event "ADAE"
# Note that ASTDY is represented by a circle and AENDY is represented by a square.
# If AENDY and ASTDY occur on the same day only AENDY will be shown.

# Adverse Event ADAE
ADAE <- osprey::rADAE %>%
  select(USUBJID, STUDYID, AESOC, AEDECOD, AESER, AETOXGR, AEREL, ASTDY, AENDY)
ADAE <- left_join(ADSL, ADAE, by = c("USUBJID", "STUDYID"))

p2 <- patient_domain_profile(
  domain = "Adverse Event (ADAE)",
  var_names = ADAE$AEDECOD,
  marker_pos = ADAE[, c("ASTDY", "AENDY")],
  arrow_end = ADSL$max_day,
  xtick_at = waiver(),
  line_col_list = list(
    line_col = ADAE$AESER,
    line_col_legend = "Serious",
```

```
    line_col_opt = c("blue", "green")
  ),
  line_width = 1,
  arrow_size = 0.1,
  no_enddate_extention = 0,
  marker_col_list = list(
    marker_col = factor(ADAE$AETOXGR),
    marker_col_opt = c("3" = "yellow", "4" = "red"),
    marker_col_legend = NULL
  ),
  marker_shape_list = list(
    marker_shape = NULL,
    marker_shape_opt = NULL,
    marker_shape_legend = "Grade"
  ),
  show_days_label = TRUE,
  xlim = c(-28, ADSL$max_day),
  xlab = "Study Day",
  title = paste("Patient Profile: ", ADSL$USUBJID)
)
p2

# Example 3 Tumor Response "ADRS"
ADRS <- osprey::rADRS %>%
  select(USUBJID, STUDYID, PARAMCD, PARAM, AVALC, AVAL, ADY, ADTM)
ADRS <- left_join(ADSL, ADRS, by = c("USUBJID", "STUDYID"))
p3 <- patient_domain_profile(
  domain = "Tumor Response (ADRS)",
  var_names = ADRS$PARAMCD,
  marker_pos = ADRS$ADY,
  arrow_end = ADSL$max_day,
  xtick_at = waiver(),
  line_col_list = NULL,
  line_width = 1,
  arrow_size = 0.1,
  no_enddate_extention = 0,
  marker_col_list = list(
    marker_col = factor(ADRS$AVALC),
    marker_col_opt = c(
      "CR" = "green", "PR" = "blue",
      "SD" = "yellow", "PD" = "red", "NE" = "pink",
      "Y" = "lightblue", "N" = "darkred"
    ),
    marker_col_legend = NULL
  ),
  marker_shape_list = list(
    marker_shape = factor(ADRS$AVALC),
    marker_shape_opt = c(
      "CR" = 21, "PR" = 24,
      "SD" = 23, "PD" = 22, "NE" = 14,
      "Y" = 11, "N" = 8
    ),
    marker_shape_legend = "Response"
```

```
  ),
  show_days_label = TRUE,
  xlim = c(-28, ADSL$max_day),
  xlab = "Study Day",
  title = paste("Patient Profile: ", ADSL$USUBJID)
)
p3

# Example 4 Concomitant Med "ADCM"
ADCM <- osprey::rADCM %>%
  select(USUBJID, STUDYID, ASTDTM, AENDTM, CMDECOD, ASTDY, AENDY)
ADCM <- left_join(ADSL, ADCM, by = c("USUBJID", "STUDYID"))
p4 <- patient_domain_profile(
  domain = "Concomitant Med (ADCM)",
  var_names = ADCM$CMDECOD,
  marker_pos = ADCM[, c("ASTDY", "AENDY")],
  arrow_end = ADSL$max_day,
  xtick_at = waiver(),
  line_col_list = list(line_col_opt = "orange"),
  line_width = 1,
  arrow_size = 0.1,
  no_enddate_extention = 50,
  marker_col_list = list(marker_col_opt = "orange"),
  marker_shape_list = NULL,
  show_days_label = TRUE,
  xlim = c(-28, ADSL$max_day),
  xlab = "Study Day",
  title = paste("Patient Profile: ", ADSL$USUBJID)
)
p4

# Example 5 Laboratory "ADLB"
ADLB <- osprey::rADLB %>%
  select(
    USUBJID, STUDYID, LBSEQ, PARAMCD, BASETYPE,
    ADTM, ADY, ATPTN, AVISITN, LBTESTCD, ANRIND
  )
ADLB <- left_join(ADSL, ADLB, by = c("USUBJID", "STUDYID"))

ADLB <- ADLB %>%
  group_by(USUBJID) %>%
  mutate(ANRIND = factor(ANRIND, levels = c("LOW", "NORMAL", "HIGH")))

p5 <- patient_domain_profile(
  domain = "Laboratory (ADLB)",
  var_names = ADLB$LBTESTCD,
  marker_pos = ADLB$ADY,
  arrow_end = ADSL$max_day,
  xtick_at = waiver(),
  line_col_list = NULL,
  line_width = 1,
  arrow_size = 0.1,
  no_enddate_extention = 0,
```

```
  marker_col_list = list(
    marker_col = factor(ADLB$ANRIND),
    marker_col_opt = c(
      "HIGH" = "red", "LOW" = "blue",
      "NORMAL" = "green", "NA" = "green"
    )
  ),
  marker_shape_list = list(
    marker_shape = factor(ADLB$ANRIND),
    marker_shape_opt = c(
      "HIGH" = 24, "LOW" = 25,
      "NORMAL" = 23, "NA" = 23
    ),
    marker_shape_legend = "Labs Abnormality"
  ),
  show_days_label = TRUE,
  xlim = c(-30, ADSL$max_day),
  xlab = "Study Day",
  title = paste("Patient Profile: ", ADSL$USUBJID)
)
p5
```

`spiderplot_simple` *Simple spider plot*

Description

Description of this plot

Usage

```
spiderplot_simple(
  anl,
  byvar = "USUBJID",
  days = "TRTDURD",
  mes_value = "PARAM",
  group_col = "USUBJID",
  baseday = 0
)
```

Arguments

`anl`	The analysis data frame
`byvar`	Analysis dataset
`days`	Variable with time in days
`mes_value`	Variable with measurement
`group_col`	Variable to color the individual lines and id in plot
`baseday`	Numeric Value, points with only smaller values will be cut out

Value

ggplot object

Author(s)

Mika Maekinen

Examples

```
library(dplyr)
library(nestcolor)

ADSL <- osprey::rADSL[1:15, ]
ADTR <- osprey::rADTR
ANL <- left_join(ADSL, ADTR)

ANL %>%
  dplyr::filter(ANL01FL == "Y" & PARAMCD == "SLDINV") %>%
  spiderplot_simple(group_col = "SEX", days = "ADY", mes_value = "AVAL")
```

---

stream_filter *Applies STREAM style filtering to datasets*

Description

One of slref or anl need to be specified. The conversion from SAS code in filters dataset may not work in all cases. In case of failure a sensible error message should be returned.

Usage

```
stream_filter(
  slref = NULL,
  anl = NULL,
  filters,
  suffix,
  slref_keep = NULL,
  usubjid = "USUBJID"
)
```

Arguments

slref	The subject level data frame (typically ADSL)
anl	The analysis data frame
filters	The name of the filters dataset
suffix	The suffix to apply in quotes (e.g. "ITT_PFSINV")
slref_keep	Variables to keep from slref (e.g. c("REGION", "SEX"))
usubjid	The unique subject identifier variable in quotes (e.g. "USUBJID")

Value

dataframe object

Author(s)

Iain Bennett

Examples

```
ADSL <- osprey::rADSL
ADTTE <- osprey::rADTTE
filters <- as.data.frame(rbind(
  c(ID = "IT", FLTTARGET = "SLREF", FLTWHERE = "where 1 eq 1"),
  c(ID = "BIO", FLTTARGET = "SLREF", FLTWHERE = "where BMRKR1 ge 4.3"),
  c(ID = "M", FLTTARGET = "SLREF", FLTWHERE = "where SEX eq 'M'"),
  c(ID = "PFS", FLTTARGET = "ANL", FLTWHERE = "where PARAMCD eq 'PFS'"),
  c(ID = "OS", FLTTARGET = "ANL", FLTWHERE = "where PARAMCD eq 'OS'")
))

ANL <- stream_filter(
  slref = ADSL,
  anl = ADTTE,
  suffix = "IT_PFS_BIO",
  filters = filters
)
```

---

stream_filter_convwhere

Convert SAS *code to R code*

Description

Will convert following SAS operators: eq, =, le, lt, ge, gt, index Will convert following logic: and, or, () Will convert all unquoted values to upper case (assumed to be variable names) All quoted values will be returned with single quotes - may fail if have quotes within quotes

Usage

```
stream_filter_convwhere(x)
```

Arguments

x	a character string of SAS code

Value

a character string of R code

Author(s)

Iain Bennett

Examples

```
stream_filter_convwhere(x = "where X in (1 2 3 4) and Y gt 4 ")
stream_filter_convwhere(x = "where X = \"fred\" and Y gt 4 ")
```

---

stream_filter_index *Replicates the use of index function in* SAS *for logic options*

Description

Assumption is that use in filters is to only resolve true vs false Primarily for use with stream_filter and related `stream_filter_convwhere` functions

Usage

```
stream_filter_index(string1, string2)
```

Arguments

`string1`	The string to search within - can be a vector
`string2`	The string to search for - must have length 1

Value

`boolean` indicator

Author(s)

Iain Bennett

Examples

```
AEACN <- c("DRUG MODIFIED", "DRUG STOPPED", "DOSE/DRUG MODIFIED")
stream_filter_index(AEACN, "DRUG MODIFIED")
```

Index