Package: gwasforest (via r-universe)

September 12, 2024

Title Make Forest Plot with GWAS Data

Version 1.0.0.9002

Description Extract and reform data from GWAS (genome-wide association study) results, and then make a single integrated forest plot containing multiple windows of which each shows the result of individual SNPs (or other items of interest).

URL `https://yilixu.github.io/gwasforest/`

BugReports `https://github.com/yilixu/gwasforest/issues`

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

Imports colorspace, data.table, dplyr, ggplot2, ggrepel, glue, utils

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

RemoteUrl https://github.com/yilixu/gwasforest

RemoteRef HEAD

RemoteSha d9afc21cacea199f651d7da5ac5b13631be6365f

Contents

`gwasforest` *Make forest plot with GWAS data*

Description

Extract and reform data from GWAS results, and then make a single integrated forest plot containing multiple windows of which each shows the result of individual SNPs (or other items of interest).

Usage

```
gwasforest(
  customFilename,
  customFilename_results = NULL,
  customFilename_studyName = NULL,
  keepStudyOrder = TRUE,
  stdColnames = FALSE,
  customColnames = NULL,
  calculateEXP = FALSE,
  calculateCI = TRUE,
  flipMeta = FALSE,
  valueFormat = "Effect",
  metaStudy = "Study1",
  colorMode = "duo",
  forestLayout = "auto",
  plotTitle = "auto",
  showMetaValue = TRUE,
  outputFolderPath = NULL,
  outputPlot_format = "png",
  outputPlot_resolution = 320
)
```

Arguments

`customFilename`	string, relative or absolute path to the raw GWAS data file. Or dataframe in the same format as provided in the examples (see examples).
`customFilename_results`	string, relative or relative path to the gwasforest-generated results file. Or dataframe in the same format as provided in the examples (see examples). If customFilename_results is provided, certain downstream calculations will be skipped.
`customFilename_studyName`	(optional) string, relative or absolute path to the study name file. Or dataframe in the same format as provided in the examples (see examples). Required if users want to use their own study names which is not standardized (see "stdColnames"); all study names should be in one column with a header; also, study names should be in the order of that they first appear in the input data columns.

`keepStudyOrder`	logical, whether to keep studies (except for meta study) in the original order provided by user (from customFilename_studyName), or sort them alphabetically on the combined forest plot; meta study will always be put at the bottom of the combined forest plot.
`stdColnames`	logical, whether the input data has standardized column names as provided in the instruction example, if TRUE, column order doesn't matter (except that study1 needs to be the Meta study); if FALSE, see "customColnames".
`customColnames`	# character, case-sensitive, can be a vector, choose from c("Value", "StdErr") or c("Value", "Upper", "Lower") based on what columns are contained in the input data; required if stdColnames = FALSE, also the input data need to be grouped by study while in the customColnames order, e.g. Study1__Value, Study1__StdErr, Study2__Value, Study2__StdErr... (watch for the deliberate double underscores "__" used to connect study name and data type); in addition, each study should contain the same number of columns.
`calculateEXP`	logical, whether to calculate exp(Value), if TRUE, downstream calculateCI will also take exp into consideration.
`calculateCI`	logical, whether to calculate Confidence Interval, if TRUE, input data need to contain "StdErr" column; if FALSE, input data need to contain "Upper" and "Lower" columns.
`flipMeta`	logical, whether to flip the input value of meta study for calculating exp(Value) and confidence interval; e.g. when meta beta is < 0, let's say -0.5, set flipMeta = TRUE to let downstream calculation use -(meta beta), which is -(-0.5) = 0.5, or vice versa; calculating exp(Value) and confidence interval will reflect the flip in meta input value.
`valueFormat`	character, format of Value column, e.g. "Effect", "Beta", "OR", "HR", "logRR"...
`metaStudy`	character, which study is the meta study, by default "Study1" (the first study appear in the input data columns).
`colorMode`	character, choose from c("mono", "duo", "diverse"), mono - render all studies including meta study in the same color, duo - highlight meta study, diverse - render all studies in different colors.
`forestLayout`	character, or integer vector, determines the layout of the combined forest plot, by default use "auto" which will automatically arrange the combined forest plot; or user can explicitly set the row/column layout by providing a vector c(rowNum, colNum).
`plotTitle`	character/string, the title of the combined forest plot, can be customized or simply set to "auto".
`showMetaValue`	logical, whether to show value for meta group on the combined forest plot.
`outputFolderPath`	string, relative or absolute path to the output folder (watch for the trailing slash "/"), can be set to NULL (no output file will be written to the file system).
`outputPlot_format`	character, format of the output forest plot, the default and recommended option is "png". Accepted custom formats are (in alphabetical order): “bmp”, “eps”, “jpeg”, “pdf”, “png”, “ps”, “svg”, and “tiff”.

`outputPlot_resolution`
: integer, resolution of the output forest plot (in dpi), the default and recommended option is 320.

Value

list, users can run the function without assigning the return value to a variable. If assigned to a variable, it will be a list containing GWAS results (dataframe) and GWAS forest plot (ggplot2 object).

Examples

```
# customFilename in dataframe format (with standardized column names)
tempValue = runif(n = 18, min = 0.01, max = 2)
tempStdErr = tempValue / rep(3:5, times = 6)
eg_customFilename = data.frame(paste0("Marker", 1:6), tempValue[1:6],
    tempStdErr[1:6], tempValue[7:12], tempStdErr[7:12], tempValue[13:18],
    tempStdErr[13:18], stringsAsFactors = FALSE)
colnames(eg_customFilename) = c("MarkerName", paste0(rep("Study", times = 6),
    rep(1:3, each = 2), sample(LETTERS, 6)))
rm(tempValue, tempStdErr)
eg_customFilename_studyName = data.frame("studyName" = paste0("Study", 1:3),
    stringsAsFactors = FALSE)
eg_returnList = gwasforest(eg_customFilename, customFilename_studyName =
    eg_customFilename_studyName, stdColnames = FALSE, customColnames = c("Value",
    "StdErr"), valueFormat = "Effect", metaStudy = "Study1", colorMode = "duo")

# customFilename in dataframe format (without standardized column names),
# with customFilename_studyName provided in dataframe format
tempValue = runif(n = 18, min = 0.01, max = 2)
tempStdErr = tempValue / rep(3:5, times = 6)
eg_customFilename2 = data.frame(paste0("Marker", 1:6), tempValue[1:6],
    tempStdErr[1:6], tempValue[7:12], tempStdErr[7:12], tempValue[13:18],
    tempStdErr[13:18], stringsAsFactors = FALSE)
colnames(eg_customFilename2) = c("MarkerName", paste0(rep("Study", times = 6),
    rep(1:3, each = 2), sample(LETTERS, 6)))
rm(tempValue, tempStdErr)
eg_customFilename_studyName = data.frame("studyName" = paste0("Study", 1:3),
    stringsAsFactors = FALSE)
eg_returnList2 = gwasforest(eg_customFilename2, customFilename_studyName =
    eg_customFilename_studyName, stdColnames = FALSE, customColnames = c("Value",
    "StdErr"), valueFormat = "Effect", metaStudy = "Study1", colorMode = "duo")

# customFilename_results in dataframe format (run either of the two examples
# above to see the example results)
eg_customFilename_results = eg_returnList[[1]]
```

Index