Package 'subscreen'

March 18, 2025

Type Package

Title Systematic Screening of Study Data for Subgroup Effects

Version 4.0.1

Description Identifying outcome relevant subgroups has now become as simple as possible!
The formerly lengthy and tedious search for the needle in a haystack will be replaced by a single, comprehensive and coherent presentation.
The central result of a subgroup screening is a diagram in which each single dot stands for a subgroup. The diagram may show thousands of them. The position of the dot in the diagram is determined by the
sample size of the subgroup and the statistical measure of the treatment effect in that subgroup. The sample size is shown on the horizontal axis while the treatment effect is displayed on the vertical axis. Furthermore, the diagram shows the line of no effect and the overall study results. For small subgroups, which are found on the left side of the plot, larger random deviations from the mean study effect are expected, while for larger subgroups only small deviations from the study mean can be expected to be chance findings.
So for a study with no conspicuous subgroup effects, the dots in the figure are expected to form a kind of funnel. Any deviations from this funnel shape hint to conspicuous subgroups.

License GPL-3

Depends R (>= 3.5.0)

Encoding UTF-8

LazyData TRUE

Imports utils, plyr, data.table, ggplot2, ggrepel, rlang, stringr, grDevices, graphics, shiny, DT, stats, shinyjs, methods, bsplus, colourpicker, dplyr, ranger, shinyWidgets

Suggests parallel, survival, knitr, rmarkdown, testthat

NeedsCompilation no

RoxygenNote 7.2.3

VignetteBuilder knitr

Config/testthat/edition 3

Author Bodo Kirsch [aut, cre],
Steffen Jeske [aut],
Julia Eichhorn [aut],
Susanne Lippert [aut],
Thomas Schmelter [aut],
Christoph Muysers [aut],
Hermann Kulmann [aut]

Maintainer Bodo Kirsch <kirschbodo@gmail.com>

Repository CRAN

Date/Publication 2025-03-18 21:20:02 UTC

Contents

subscreencalc	*(i) Calculation of the results for the subgroups*

Description

This function systematically calculates the defined outcome for every combination of subgroups up to the given level (max_comb), i.e. the number of maximum combinations of subgroup defining factors. If, e.g., in a study sex, age group (<=60, >60), BMI group (<=25, >25) are of interest, subgroups of level 2 would be, e.g, male subjects with BMI>25 or young females, while subgroups of level 3 would be any combination of all three variables.

Usage

```
subscreencalc(
  data,
  eval_function,
  subjectid = "subjid",
  factors = NULL,
  max_comb = 3,
  nkernel = 1,
  par_functions = "",
  verbose = TRUE,
  factorial = FALSE,
  use_complement = FALSE,
  ...
)
```

Arguments

`data`	dataframe with study data
`eval_function`	name of the function for data analysis
`subjectid`	name of variable in data that contains the subject identifier, defaults to subjid
`factors`	character vector containing the names of variables that define the subgroups (required)
`max_comb`	maximum number of factor combination levels to define subgruops, defaults to 3
`nkernel`	number of kernels for parallelization (defaults to 1)
`par_functions`	vector of names of functions used in eval_function to be exported to cluster (needed only if nkernel > 1)
`verbose`	logical value to switch on/off output of computational information (defaults to TRUE)
`factorial`	logical value to switch on/off calculation of factorial contexts (defaults to FALSE)
`use_complement`	logical value to switch on/off calculation of complement subgroups (defaults to FALSE)
`...`	further parameters which where outdated used for notes and errors.

Details

The evaluation function (eval_function) has to defined by the user. The result needs to be a vector of numerical values, e.g., outcome variable(s) and number of observations/subjects. The input of eval_function is a data frame with the same structure as the input data frame (data) used in the subsreencalc call. See example below. Potential errors occurring due to small subgroups should be caught and handled within eval_function. As the eval_function will be called with every subgroup it may happen that there is only one observation or only one treatment arm or only observations with missing data going into the eval_function. There should always be valid result vector be returned (NAs allowed) and no error causing program abort. For a better display the results may be cut-off to a reasonable range. For example: If my endpoint is a hazard ratio that is expected to be between 0.5 and 2 I would set all values smaller than 0.01 to 0.01 and values above 100 to 100.

Value

an object of type SubScreenResult of the form list(sge=H, max_comb=max_comb, min_comb=min_comb, subjectid=subjectid, treat=treat, factors=factors, results_total=eval_function(cbind(F,T)))

Examples

```
# get the pbc data from the survival package
require(survival)
data(pbc, package="survival")
# generate categorical versions of some of the baseline covariates
pbc$ageg[!is.na(pbc$age)]        <-
  ifelse(pbc$age[!is.na(pbc$age)]        <= median(pbc$age,    na.rm=TRUE), "Low", "High")
pbc$albuming[!is.na(pbc$albumin)]<-
  ifelse(pbc$albumin[!is.na(pbc$albumin)] <= median(pbc$albumin, na.rm=TRUE), "Low", "High")
```

```
pbc$phosg[!is.na(pbc$alk.phos)]  <-
  ifelse(pbc$alk.phos[!is.na(pbc$alk.phos)]<= median(pbc$alk.phos,na.rm=TRUE), "Low", "High")
pbc$astg[!is.na(pbc$ast)]        <-
  ifelse(pbc$ast[!is.na(pbc$ast)]       <= median(pbc$ast,    na.rm=TRUE), "Low", "High")
pbc$bilig[!is.na(pbc$bili)]      <-
  ifelse(pbc$bili[!is.na(pbc$bili)]      <= median(pbc$bili,   na.rm=TRUE), "Low", "High")
pbc$cholg[!is.na(pbc$chol)]      <-
  ifelse(pbc$chol[!is.na(pbc$chol)]      <= median(pbc$chol,   na.rm=TRUE), "Low", "High")
pbc$copperg[!is.na(pbc$copper)]  <-
  ifelse(pbc$copper[!is.na(pbc$copper)]  <= median(pbc$copper, na.rm=TRUE), "Low", "High")
#eliminate treatment NAs
pbcdat <- pbc[!is.na(pbc$trt), ]
# PFS and OS endpoints
set.seed(2006)
pbcdat$'event.pfs' <- sample(c(0,1),dim(pbcdat)[1],replace=TRUE)
pbcdat$'timepfs' <- sample(1:5000,dim(pbcdat)[1],replace=TRUE)
pbcdat$'event.os' <- pbcdat$event
pbcdat$'timeos' <- pbcdat$time
#variable importance for OS for the created categorical variables
#(higher is more important, also works for numeric variables)
varnames <- c('ageg', 'sex', 'bilig', 'cholg', 'astg', 'albuming', 'phosg')
# define function the eval_function()
# Attention: The eval_function ALWAYS needs to return a dataframe with one row.
#            Include exception handling, like if(N1>0 && N2>0) hr <- exp(coxph(...) )
#            to avoid program abort due to errors
hazardratio <- function(D) {

 HRpfs <- tryCatch(exp(coxph(Surv(D$timepfs, D$event.pfs) ~ D$trt )$coefficients[[1]]),
  warning=function(w) {NA})
 HRpfs <- 1/HRpfs
 HR.pfs <- round(HRpfs, 2)
 HR.pfs[HR.pfs > 10]     <- 10
 HR.pfs[HR.pfs < 0.00001] <- 0.00001
 HRos <- tryCatch(exp(coxph(Surv(D$timeos, D$event.os) ~ D$trt )$coefficients[[1]]),
  warning=function(w) {NA})
 HRos <- 1/HRos
 HR.os <- round(HRos, 2)
 HR.os[HR.os > 10]      <- 10
 HR.os[HR.os < 0.00001] <- 0.00001
 data.frame( HR.pfs, HR.os#, N.of.subjects,N1 ,N2
 )
}

 # run subscreen

## Not run:
results <- subscreencalc(
  data=pbcdat,
  eval_function=hazardratio,
  subjectid = "id",
  factors=c("ageg", "sex", "bilig", "cholg", "copperg"),
  use_complement = FALSE,
  factorial = FALSE
```

```
)

# visualize the results of the subgroup screening with a Shiny app
subscreenshow(results)

## End(Not run)
```

---

subscreenshow *(ii) Visualization*

Description

Start the Shiny based interactive visualization tool to show the subgroup results generated by subscreencalc. See and explore all subgroup results at one glance. Pick and chose a specific subgroup, the level of combinations or a certain factor with its combinations. Switch easily between different endpoint/target variables.

Usage

```
subscreenshow(
  scresults = NULL,
  variable_importance = NULL,
  host = NULL,
  port = NULL,
  NiceNumbers = c(1, 1.5, 2, 4, 5, 6, 8, 10),
  windowTitle = "Subgroup Explorer",
  graphSubtitle = NULL,
  favour_label_verum_name = NULL,
  favour_label_comparator_name = NULL
)
```

Arguments

scresults   SubScreenResult object with results from a subscreencalc call

variable_importance

                 variable importance object calculated via subscreenvi to unlock 'variable importance'-tab in the app

host   host name or IP address for Shiny display

port   port number for Shiny display

NiceNumbers   list of numbers used for a 'nice' scale

windowTitle   title which is shown for the browser tab

graphSubtitle   subtitle for explorer plot

favour_label_verum_name

                 verum name for label use in explorer graph

favour_label_comparator_name

                 comparator name for label use in explorer graph

---

| subscreenvi | *(iii) Determine variable importance* |

Description

Determine variable importance for continuous, categorical or right-censored survival endpoints (overall and per treatment group) using random forests

Usage

```
subscreenvi(data, y, cens = NULL, x = NULL, trt = NULL)
```

Arguments

data	The data frame containing the dependent and independent variables.
y	The name of the column in data that contains the dependent variable.
cens	The name of the column in data that contains the censoring variable, if y is an event time (default=NULL).
x	Vector that contains the names of the columns in data with the independent variables (default=NULL, i.e. all remaining variables)
trt	The name of the column in data that contains the treatment variable (default=NULL).

Value

A list containing ordered data frames with the variable importances (one for each treatment level, one with the ranking variability between the treatment levels and one with the total results)

Examples

```
## Not run:
require(survival)
data(pbc, package="survival")
# generate categorical versions of some of the baseline covariates
pbc$ageg[!is.na(pbc$age)]         <-
  ifelse(pbc$age[!is.na(pbc$age)]          <= median(pbc$age,    na.rm=TRUE), "Low", "High")
pbc$albuming[!is.na(pbc$albumin)]<-
  ifelse(pbc$albumin[!is.na(pbc$albumin)] <= median(pbc$albumin, na.rm=TRUE), "Low", "High")
pbc$phosg[!is.na(pbc$alk.phos)]  <-
  ifelse(pbc$alk.phos[!is.na(pbc$alk.phos)]<= median(pbc$alk.phos,na.rm=TRUE), "Low", "High")
pbc$astg[!is.na(pbc$ast)]         <-
  ifelse(pbc$ast[!is.na(pbc$ast)]          <= median(pbc$ast,    na.rm=TRUE), "Low", "High")
pbc$bilig[!is.na(pbc$bili)]       <-
  ifelse(pbc$bili[!is.na(pbc$bili)]        <= median(pbc$bili,   na.rm=TRUE), "Low", "High")
pbc$cholg[!is.na(pbc$chol)]       <-
  ifelse(pbc$chol[!is.na(pbc$chol)]        <= median(pbc$chol,   na.rm=TRUE), "Low", "High")
pbc$copperg[!is.na(pbc$copper)]  <-
  ifelse(pbc$copper[!is.na(pbc$copper)]   <= median(pbc$copper, na.rm=TRUE), "Low", "High")
pbc$ageg[is.na(pbc$age)]          <- "No Data"
```

```
pbc$albuming[is.na(pbc$albumin)] <- "No Data"
pbc$phosg[is.na(pbc$alk.phos)]   <- "No Data"
pbc$astg[is.na(pbc$ast)]         <- "No Data"
pbc$bilig[is.na(pbc$bili)]       <- "No Data"
pbc$cholg[is.na(pbc$chol)]       <- "No Data"
pbc$copperg[is.na(pbc$copper)]   <- "No Data"
#eliminate treatment NAs
pbcdat <- pbc[!is.na(pbc$trt), ]
pbcdat$status <- ifelse(pbcdat$status==0,0,1)
importance <- subscreenvi(data=pbcdat, y='time', cens='status',
 trt='trt', x=c("ageg", "sex", "bilig", "cholg", "copperg"))


## End(Not run)
```

Index