Package 'survutils'

July 22, 2018

Title Utility Functions for Survival Analysis

Version 1.0.2

Description Functional programming principles to iteratively run Cox regression and plot its results. The results are reported in tidy data frames. Additional utility functions are available for working with other aspects of survival analysis such as survival curves, C-statistics, etc.

License GPL-3

LazyData true

Depends R (>= 3.1.2)

Imports stats, dplyr (>= 0.4.3), survC1 (>= 1.0.2), survival (>= 2.38.3), ggplot2 (>= 1.0.0), purrr (>= 0.1.0), magrittr (>= 1.5), broom (>= 0.3.7), lazyeval (>= 0.2.0), glue

Suggests tidyr (>= 0.4.1), testthat

URL https://github.com/tinyheero/survutils

BugReports https://github.com/tinyheero/survutils/issues

RoxygenNote 6.0.1

NeedsCompilation no

Author Fong Chun Chan [aut, cre]

Maintainer Fong Chun Chan <fongchunchan@gmail.com>

Repository CRAN

Date/Publication 2018-07-22 17:50:02 UTC

R topics documented:

add_sig_line_to_plot 2
get_cox_res 2
get_cox_summary 3
get_c_stat 4
get_logrank_res 5

get_nrisk_tbl 5
get_surv_prob 6
iter_get_cox_res 7
plot_cox_res 8
survutils 9

Index **11**

`add_sig_line_to_plot` *Add a Cox regression statistical significance line*

Description

This function will add a Cox regression statistical significance line to the input ggplot2.

Usage

```
add_sig_line_to_plot(cur_plot, coord_flip, add_sig_line)
```

Arguments

`cur_plot`	ggplot2 object to add on to.
`coord_flip`	Boolean to indicate if the ggplot2 has been flipped or not.
`add_sig_line`	Boolean to indicate whether the significance line should be added or not.

`get_cox_res` *Run Cox Regression on a Single or Multiple Groups of Data*

Description

`get_cox_res` is a wrapper around coxph. It can run univariate or multivariate cox regression. If the group parameter is used, then cox regression is run for each group separately.

Usage

```
get_cox_res(in.df, endpoint, endpoint.code, features, group = NULL,
  broom.fun = c("tidy", "glance"))
```

Arguments

`in.df`	Input data.frame.
`endpoint`	Column name of the endpoint.
`endpoint.code`	Column name of the endpoint status code.
`features`	Vector containing the features to run cox regression on.
`group`	Column name containing the groups to run cox regression on. If, specified, cox regression is run separately for each group.
`broom.fun`	Which broom function to run on the cox regression results.

Details

The data is returned in a broom::tidy data.frame format.

Value

Cox regression results returned in a tidy data.frame format.

Examples

```
library("survival")
endpoint <- "time"
endpoint.code <- "status"

# Run Univariate Cox Regression on Single Feature
features <- "age"
test.df <- get_cox_res(colon, endpoint, endpoint.code, features)

# Run Univariate Cox Regression on Multiple Features
multi.features <- c("age", "obstruct")
get_cox_res(colon, endpoint, endpoint.code, multi.features)

# Run Univariate Cox Regression on Multiple Features For Each rx group
group <- "rx"
get_cox_res(colon, endpoint, endpoint.code, multi.features, group)

# Run Multivariate Cox Regression
get_cox_res(colon, endpoint, endpoint.code, multi.features)

# Run Multivariate Cox Regression For Each rx Group
get_cox_res(colon, endpoint, endpoint.code, multi.features, group)
```

---

get_cox_summary *Summarizes the Cox Regression Analysis*

Description

This function summarizes the results of a cox regression returning the results in a data.frame. It will calculate the confidence interval.

Usage

```
get_cox_summary(cox.res, ci = 95)
```

Arguments

cox.res	The result of a coxph fit.
ci	The confidence interval to calculate.

Value

A data.frame that summarizes the results.

Examples

```
cox.res <- survival::coxph(survival::Surv(time,status) ~ rx, survival::colon)
get_cox_summary(cox.res)
```

`get_c_stat` *Calculate C-statistics*

Description

Wrapper around the Inf.Cval function from the survC1 R package to calculate C-statistics.

Usage

```
get_c_stat(in.df, endpoint, endpoint.code, prog.factor, tau.val)
```

Arguments

`in.df`	data.frame containing all the input data.
`endpoint`	Column name of endpoint.
`endpoint.code`	Column name of endpoint code.
`prog.factor`	Column name of the prognostic factor to test.
`tau.val`	Vector of tau values to be used for C-statistics inference.

Value

data.frame containing the c-statistic, 95

Examples

```
# Example taken from survC1
## Not run:
library("survival")
in.df <- survC1::CompCase(pbc[1:200, c(2:4,10:14)])
in.df[, 2] <- as.numeric(in.df[,2]==2)
tau <- 365.25*8
prog.factor <- c("trt", "edema", "bili", "chol", "albumin", "copper")
get_c_stat(in.df, "time", "status", prog.factor, tau)

## End(Not run)
```

get_logrank_res *Run Log-Rank Test*

Description

get_logrank_res is a wrapper over the survival::survdiff() function return the direct results or the log rank p-value only if specified.

Usage

```
get_logrank_res(in.formula, in.df, return.p = FALSE)
```

Arguments

in.formula	Survival model formula. Can extract from an existing survfit object with formula(survfit).
in.df	data.frame Corresponding data for the survival model.
return.p	If set to TRUE, return only the log rank p-value.

Value

Results of survdiff or a log rank p-value if return.p is set to TRUE.

Examples

```
library("survival")

# Get survdiff results
fit <- survfit(Surv(time, status) ~ rx, data = colon)
get_logrank_res(formula(fit), colon)

# Get only log-rank p-value
get_logrank_res(formula(fit), colon, return.p = TRUE)
```

get_nrisk_tbl *Returns a Number At Risk Table from a survfit Object*

Description

This function generates a number at risk table that typically seen in publications.

Usage

```
get_nrisk_tbl(sfit, timeby)
```

Arguments

sfit	A survival::survfit object.
timeby	The "step" in which to calculate the risk.

Value

A data.frame with the number of risks at each timeby step.

Author(s)

Abhijit Dasgupta https://gist.github.com/araastat/9927677.

Examples

```
fit <- survival::survfit(survival::Surv(time,status) ~ rx, data = survival::colon)
get_nrisk_tbl(fit, timeby = 500)
```

---

get_surv_prob *Get Survival Probability at Specified Times*

Description

get_surv_prob retrieves the survival probability at a specific time from a survival curve object from the survival::survfit function. The survival curve object can only have one group.

Usage

```
get_surv_prob(fit, times)
```

Arguments

fit	survival::survfit object.
times	Vector of times to lookup survival probabilities.

Value

Vector of survival probabilities based on the input times.

Examples

```
library("survival")

# Get Survival Probabilities Based on Whole Cohort
fit <- survfit(Surv(time, status) ~ 1, data = colon)
times <- c(100, 200, 300)
get_surv_prob(fit, times)

# Get Survival Probabilities for Each rx Group
```

```
library("purrr")
library("dplyr")
library("tidyr")

surv.prob.res <-
  colon %>%
  split(.$rx) %>%
  map(~ survfit(Surv(time, status) ~ 1, data = .)) %>%
  map(get_surv_prob, times)

surv.prob.res.df <- as_data_frame(surv.prob.res)
colnames(surv.prob.res.df) <- names(surv.prob.res)
surv.prob.res.df <-
  surv.prob.res.df %>%
  mutate(surv_prob_time = times)

gather(surv.prob.res.df, "group", "surv_prob", Obs:`Lev+5FU`)
```

`iter_get_cox_res` *Runs get_cox_res Over a Range of Features*

Description

This is a modified version of `get_cox_res` allowing for multiple runs of get_cox_res.

Usage

```
iter_get_cox_res(in.df, endpoint, endpoint.code, features,
  broom.fun = c("tidy", "glance"), group = NULL)
```

Arguments

`in.df`	Input data.frame.
`endpoint`	Column name of the endpoint.
`endpoint.code`	Column name of the endpoint status code.
`features`	This must be a list of features.
`broom.fun`	Which broom function to run on the cox regression results.
`group`	Column name containing the groups to run cox regression on. If, specified, cox regression is run separately for each group.

Value

List of data frames with each data frame being the output of `get_cox_res`.

Examples

```
library("survival")
endpoint <- "time"
endpoint.code <- "status"

# Run Cox Regression on List of Features
features <- list(c("age", "obstruct"),
                 c("nodes"))

iter_get_cox_res(colon, endpoint, endpoint.code, features,
                  group = "rx")
```

`plot_cox_res` *Plot Cox Regression Results*

Description

`plot_cox_res` takes the output from `get_cox_res` and generates a forest plot showing the hazard ratio and confidence interval of the cox cox regression.

Usage

```
plot_cox_res(cox.res.df, x.lab, y.lab, y.col = "term", color.col,
  color.legend.name, coord.flip = FALSE, facet.formula = NULL,
  facet.scales = "fixed", add_sig_line = TRUE)
```

Arguments

`cox.res.df`	data.frame output from `get_cox_res`.
`x.lab`	x-axis label.
`y.lab`	y-axis label.
`y.col`	Column name that contains the values for the y-values.
`color.col`	Column name that contains color groups.
`color.legend.name`	Title for the color legend.
`coord.flip`	By default hazard ratio and its confidence interval is plotted on the y-axis using ggplot2::geom_errorbarh(). If this is set to TRUE, then this information is plotted along the x-axis using ggplot2::geom_errorbar(). This means that the x.lab and y.lab will be flipped to.
`facet.formula`	Facet formula for faceting the plot. This should be used plotting results from `iter_get_cox_res` or when the parameter group is used in `get_cox_res` and `iter_get_cox_res`.
`facet.scales`	Parameter passed to the scales parameter in `ggplot2::facet_grid`.
`add_sig_line`	Boolean to indicate if a red, dotted, vertical line should be added to allow users to see if a Cox regression confidence interval overlaps with 1.

Value

Forest plot of cox regression results in the ggplot framework.

Examples

```
## Not run:
library("survival")
library("magrittr")
library("dplyr")

in.df <- colon
endpoint <- "time"
endpoint.code <- "status"

# Run and Plot Multivariate Cox Regression on Entire data.frame
features <- c("age", "obstruct")
cox.res.df <- get_cox_res(colon, endpoint, endpoint.code, features)
plot_cox_res(cox.res.df)

# Run and Plot Multivariate Cox Regression For Each rx Group
group <- "rx"
cox.res.df <- get_cox_res(colon, endpoint, endpoint.code, features, group)
plot_cox_res(cox.res.df, facet.formula = ". ~ group")

# Change x and y labels
plot_cox_res(cox.res.df, facet.formula = ". ~ group",
             x.lab = "Hazard Ratio", y.lab = "Feature")

# Adding colors
cox.res.df %>%
  mutate(sig_flag = p.value < 0.05) %>%
  plot_cox_res(facet.formula = ". ~ group", x.lab = "Hazard Ratio",
               y.lab = "Feature",
               color.col = "sig_flag",
               color.legend.name = "Significant Flag")

# Flipping Plot
cox.res.df %>%
  mutate(sig_flag = p.value < 0.05) %>%
  plot_cox_res(facet.formula = ". ~ group", x.lab = "Hazard Ratio",
               y.lab = "Feature",
               color.col = "sig_flag",
               color.legend.name = "Significant Flag",
               coord.flip = TRUE)

## End(Not run)
```

---

survutils *survutils: A package for analyzing survival data*

Description

survutils: A package for analyzing survival data

Index

add_sig_line_to_plot, 2

get_c_stat, 4
get_cox_res, 2
get_cox_summary, 3
get_logrank_res, 5
get_nrisk_tbl, 5
get_surv_prob, 6

iter_get_cox_res, 7

plot_cox_res, 8

survutils, 9
survutils-package *(*survutils*)*, 9