Package: shinyCox (via r-universe)

October 27, 2024

Title Create 'shiny' Applications for Cox Proportional Hazards Models

Version 1.1.0

Description Takes one or more fitted Cox proportional hazards models and writes a 'shiny' application to a directory specified by the user. The 'shiny' application displays predicted survival curves based on user input, and contains none of the original data used to create the Cox model or models. The goal is towards visualization and presentation of predicted survival curves.

License LGPL (>= 3)

URL https://github.com/harryc598/shinyCox

BugReports https://github.com/harryc598/shinyCox/issues

Imports shiny, survival (>= 3.3)

Suggests DT, knitr, rmarkdown, shinydashboard

Config/testthat/edition 3

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.1

Repository https://harryc598.r-universe.dev

RemoteUrl https://github.com/harryc598/shinycox

RemoteRef HEAD

RemoteSha 348d6a96004a8a05e5aec01608d34e9ac1be68d0

Contents

cox_KM_plots 2
cox_times_table 3
get_confint 5
make_coxph 6
predict_one_coxfit 7

predict_se 8
prep_coxfit 9
shine_coxph 9
surv_pred_info 11

Index **13**

`cox_KM_plots` *Generate Cox-model predicted Kaplan-Meier plots*

Description

The main purpose of this function is to be used to create plots within the shiny app created by `shine_coxph()`. For this reason the argument it takes, `KM.hat`, is created through a process delineated in the example. This can make the function more complicated if you want to use it outside of the shiny app, although it is fully possible to do so.

Usage

```
cox_KM_plots(KM.hat, clrs = NULL, confint, ylab = "Prob")
```

Arguments

`KM.hat`	Time and survival probability created by `predict_one_coxfit()`
`clrs`	color of lines
`confint`	logical value to determine if confidence intervals should be plotted
`ylab`	text label for y-axis

Value

Plot of predicted survival curve(s)

Examples

```
library(survival)
# First colon is split into three treatment arms to compare predicted
# survival across arms
split_colon <- split(colon, colon$rx)

colon_arm1 <- split_colon$Obs
colon_arm2 <- split_colon$Lev
colon_arm3 <- split_colon$`Lev+5FU`

# One coxph model is fit for each treatment

colon1ph <- coxph(Surv(time, status) ~sex +  age + obstruct + nodes,
                  colon_arm1, x = TRUE, model = TRUE)

colon2ph <- coxph(Surv(time, status) ~ sex + age + obstruct + nodes,
```

```
                  colon_arm2, x = TRUE, model = TRUE)

colon3ph <- coxph(Surv(time, status) ~ sex + age + obstruct + nodes,
                  colon_arm3, x = TRUE, model = TRUE)

# Creating list of models
cox.fit.list <- vector("list", 3)
cox.fit.list[[1]] <- prep_coxfit(colon1ph)
cox.fit.list[[2]] <- prep_coxfit(colon2ph)
cox.fit.list[[3]] <- prep_coxfit(colon3ph)

# Creating new data row for predictions
new.data <- colon[1, ]
# Creating KM.hat object
n.models=length(cox.fit.list)
KM.hat=vector('list',n.models)
lp=rep(NA,n.models)
names(KM.hat)=names(cox.fit.list)
for (i in 1:n.models)
{
 km.hat=predict_one_coxfit(cox.fit.list[[i]],new.data)
 lp[i]=attr(km.hat,'lp')
 sfit=list(time=km.hat$time,surv=km.hat$surv)
 class(sfit)='survfit'
 KM.hat[[i]]=sfit
}
# Plot
cox_KM_plots(KM.hat)
```

`cox_times_table` *Create table of Cox-model predicted probabilities*

Description

Generates tables of predicted probabilities at specified time or vector of times. The `KM.hat` object contains time and predicted survival probability information as a list of `survfit` objects.

Usage

```
cox_times_table(KM.hat, fixTimes = NULL)
```

Arguments

`KM.hat`	List of `survfit` objects
`fixTimes`	character or vector of characters representing times for which predicted survival probability is given

Details

The main purpose of this function is to be used within the shiny app for the purpose of creating predicted probability tables for user-inputted times. For this reason it is not expressly recommended to use this function outside the context of the shiny app, but it is still possible to do so if desired. The time or vector of times are inputted as characters due to the use of this function in the shiny app, where times are inputted as numbers separated by a comma

Value

Table of predicted probabilities, one column for each time, and one row for each curve

Examples

```
library(survival)
library(shinyCox)
# First colon is split into three treatment arms to compare predicted
# survival across arms
split_colon <- split(colon, colon$rx)

colon_arm1 <- split_colon$Obs
colon_arm2 <- split_colon$Lev
colon_arm3 <- split_colon$`Lev+5FU`

# One coxph model is fit for each treatment

colon1ph <- coxph(Surv(time, status) ~sex +  age + obstruct + nodes,
                  colon_arm1, x = TRUE, model = TRUE)

colon2ph <- coxph(Surv(time, status) ~ sex + age + obstruct + nodes,
                  colon_arm2, x = TRUE, model = TRUE)

colon3ph <- coxph(Surv(time, status) ~ sex + age + obstruct + nodes,
                  colon_arm3, x = TRUE, model = TRUE)

# Creating list of models
cox.fit.list <- vector("list", 3)
cox.fit.list[[1]] <- prep_coxfit(colon1ph)
cox.fit.list[[2]] <- prep_coxfit(colon2ph)
cox.fit.list[[3]] <- prep_coxfit(colon3ph)

# Creating new data row for predictions
new.data <- colon[1, ]
# Creating KM.hat object
n.models=length(cox.fit.list)
KM.hat=vector('list',n.models)
lp=rep(NA,n.models)
names(KM.hat)=names(cox.fit.list)
for (i in 1:n.models)
{
 km.hat=predict_one_coxfit(cox.fit.list[[i]],new.data)
 lp[i]=attr(km.hat,'lp')
 sfit=list(time=km.hat$time,surv=km.hat$surv)
```

```
 class(sfit)='survfit'
 KM.hat[[i]]=sfit
}

# Function takes KM.hat object and a time or vector of times
cox_times_table(KM.hat, fixTimes = "100")
```

`get_confint` *Get confidence intervals for predicted survival curves*

Description

Creates confidence levels for plotting predicted survival curves.

Usage

```
get_confint(p, se, conf.type, conf.int, ulimit = TRUE)
```

Arguments

`p`	Vector of survival probabilities
`se`	Vector of standard errors
`conf.type`	Type of confidence interval, includes 'plain', 'log', 'log-log', 'logit', and 'arcsin'.
`conf.int`	The level for two-sided confidence interval on the predicted survival curve, default is 0.95.
`ulimit`	Should upper bound be limited to 1, default is 'TRUE'

Value

list of length two, containing the lower and upper confidence levels

Examples

```
library(survival)
library(shinyCox)
colondeaths <- colon[colon$etype == 2, ]
split_colon <- split(colondeaths, colondeaths$rx)

colon_arm1 <- split_colon$Obs
colon1ph <- coxph(Surv(time, status) ~ factor(extent) + nodes + strata(surg)
                  + factor(differ),
                  colon_arm1,
                  x = TRUE, model = TRUE)

new.data = cbind.data.frame(`factor(extent)` = 3,
                            `surg` = "surg=0",`factor(differ)` = 2,`nodes` = 5)
```

```
coxfit = prep_coxfit(colon1ph)
coxlist = surv_pred_info(colon1ph)

for_ci = predict_se(coxlist, coxfit, new.data)

get_confint(for_ci$surv, for_ci$std.err, conf.int = 0.95,
            conf.type = "log-log")
```

---

make_coxph *Wrapper to create* `survival::coxph()` *object suitable for* `shine_coxph()`

Description

Performs `survival::coxph()` with `model = TRUE` and `x = TRUE` as defaults. Checks that Cox model is appropriate for use with `shine_coxph()`.

Usage

```
make_coxph(formula, data, ...)
```

Arguments

`formula`	a formula object, with the response on the left of a ~ operator, and the terms on the right. The response must be a survival object as returned by the `Surv` function.
`data`	a data.frame in which to interpret the variables named in the `formula`, or in the `subset` and the `weights` argument.
`...`	other arguments which will be passed to `coxph()`. Note that `x = TRUE` and `model = TRUE` are the default arguments (and required by `shine_coxph()`), you do not need to include them here.

Value

Object of class `"coxph"` representing the fit

Examples

```
library(survival)
ovarianph <- make_coxph(Surv(futime, fustat) ~ age + strata(rx),
data = ovarian)
```

`predict_one_coxfit` *Compute Cox-model predicted survival function*

Description

Computes Cox-model predicted survival function for one new data row using `coxfit` list object created by `prep_coxfit()`.

Usage

```
predict_one_coxfit(coxfit, newdata)
```

Arguments

`coxfit`	This is an object returned by `prep_coxfit()`
`newdata`	vector of new data

Value

data.frame of predicted survival probabilities over time, one column is time, one is probability

Note

This function's primary use is within the shiny app, where a coxph object is not available. It can be used outside of that context but that is the main purpose of this function, and why it only accepts the return object of `prep_coxfit()`. In the context of the shiny app, the new data is taken from user inputs.

Examples

```
# First, fit model using coxph
library(survival)
bladderph <- coxph(Surv(stop, event) ~ rx + number + size, bladder,
model = TRUE, x = TRUE)
# Use coxph object with function
bladderfit <- prep_coxfit(bladderph)
# Take first row of bladder as 'new data'
newdata <- bladder[1, ]
predictions <- predict_one_coxfit(bladderfit, newdata)
```

---

predict_se *Creates predicted survival and standard errors for confidence intervals*

Description

Adapted from parts of `survfit.coxph()`, computes predictions for standard errors based on `surv_pred_info()` output and `newdata` from the shiny app.

Usage

```
predict_se(listsurv, coxfit, newdata)
```

Arguments

`listsurv`	Output from `surv_pred_info()` function
`coxfit`	`coxfit` object created for predictions. Used to find strata
`newdata`	Data used to make predicted standard errors

Value

a list of number of subjects for each curve, times at which the curve has a step, number at risk for each time, number of events at each time, number censored at each time (no event but exit risk set), estimated survival, cumulative hazard at each transition, and standard error of the cumulative hazard.

Examples

```
library(survival)
library(shinyCox)
colondeaths <- colon[colon$etype == 2, ]
split_colon <- split(colondeaths, colondeaths$rx)

colon_arm1 <- split_colon$Obs
colon1ph <- coxph(Surv(time, status) ~
factor(extent) + nodes + strata(surg) + factor(differ), colon_arm1, x =
TRUE, model = TRUE)

new.data = cbind.data.frame(`factor(extent)` = 3, `surg` =
"surg=0",`factor(differ)` = 2,`nodes` = 5)


coxfit = prep_coxfit(colon1ph)
coxlist = surv_pred_info(colon1ph)

predict_se(coxlist, coxfit, new.data)
```

`prep_coxfit` *Create simplified* `coxph()` *object for shiny app*

Description

Simplifies `coxph()` output and checks that predictions match those of the original object

Usage

```
prep_coxfit(coxph.result, tol = 1e-07)
```

Arguments

`coxph.result`	Result returned by `coxph()`
`tol`	numerical tolerance for prediction differences, default is `1e-7`

Value

list containing baseline survival estimates, linear predictor estimates, predictor types, coefficient estimates, mean and range of numeric predictors, levels of categorical predictors, strata if any, `coxph()` formula, table of hazard ratios, table with proportional hazard assumption results, number of subjects, and number of events

Examples

```
# First, fit model using coxph
library(survival)
bladderph <- coxph(Surv(stop, event) ~ rx + number + size, bladder,
model = TRUE, x = TRUE)
# Use coxph object with function
bladderfit <- prep_coxfit(bladderph)
```

`shine_coxph` *Generates a shiny app for predictions from Cox model(s)*

Description

Writes a shiny app to visualize predicted survival curves from one or multiple Cox models. One feature of this function is that the shiny app, once created, will not contain any identifiable data, containing only information necessary for predictions.

Usage

```
shine_coxph(..., app.dir = NULL, theme = c("default", "dashboard"))
```

Arguments

`...`	Arbitrary number of Cox proportional hazard models, created by `survival::coxph()` or `make_coxph()`, which automatically ensures the models are appropriate for `shine_coxph()`
`app.dir`	Directory where shiny app is created. Specifically, a sub-folder will be made containing the `app.R` file as well as the `.Rdata` file within `app.dir`. If no directory is provided, execution will pause and the user will be asked to confirm whether this sub-folder may be created in the working directory or to stop the function and provide an input `app.dir`.
`theme`	Theme of shiny app. • `"default`: default theme, requires only shiny • `"dashboard"`: requires `"shinydashboard"` and `"DT"` packages

Value

A list containing Cox model information along with the shiny app code. The app is written to the directory while the function is operating.

Notes

There are some requirements in order for this function to run without error: in your original `survival::coxph()` function or functions, `model = TRUE` and `x = TRUE` are required arguments (used to create the simplified `"coxph"` object). Currently, this function does not support penalized models (e.g., as created by `ridge()` and `pspline()`). Multiple strata terms and strata by covariate interaction terms in the formula are also not currently supported, but workarounds are available by respectively using a new strata factor variable encompassing all combinations of desired stratum variable levels. Use of time-varying covariates (e.g. with `tt()`) and multi-state models is not supported in our function. The package is not intended to support Fine-gray models by `survival::finegray()` creating Cox models, but doing so will not result in an error.

Guidelines

This package is intended to visualize and present predicted survival functions for fitted Cox models. In regards to formula notation, the variable names used are ultimately what will be displayed in the application. Using functions in the formula will work, but with multiple nested functions it will fail. Using "." notation is not currently supported. The `na.action` is inherited from the Cox models, with `omit` being the only option with support at this time. For these reasons, we recommend creating all final variables (including suitable transformations) with meaningful names prior to using `survival::coxph()`.

Examples

```
library(survival)

# Data used is from survival package, renamed for legibility
names(leukemia)[names(leukemia) == "x"] <- "treatment"
# Make Cox model, with x = TRUE and model = TRUE
```

```
model1 <- coxph(Surv(time, status) ~ treatment,
leukemia, x = TRUE, model = TRUE)

# Use shine_coxph() to create shiny app in temporary directory
shine_coxph("Model 1" = model1)

# Get directory for shiny app (should be first, check file list if not)
filedir <- list.files(tempdir())[1]

# Run shiny app from temporary directory
shiny::runApp(paste0(tempdir(), "/", filedir))
# Remove app from directory once finished
unlink(paste0(tempdir(),"/",filedir), recursive = TRUE)
```

`surv_pred_info` *Obtains information for standard errors of predictions*

Description

Computes necessary information to calculate standard errors and confidence intervals in shiny app. This is adapted from parts of `survfit.coxph()`. This function is meant to be used in conjunction with `predict_se()`.

Usage

```
surv_pred_info(model, ctype, individual = FALSE, id, se.fit = TRUE, stype = 2)
```

Arguments

`model`	a coxph object
`ctype`	whether the cumulative hazard computation should have a correction for ties, 1=no, 2=yes.
`individual`	deprecated argument, replaced by `id`
`id`	optional variable name of subject identifiers. Not supported in app
`se.fit`	a logical value indicating whether standard errors should be computed. Default is TRUE for standard models, FALSE for multi-state (code not yet present for that case.)
`stype`	computation of the survival curve, 1=direct, 2=exponential of the cumulative hazard. Default is 2.

Value

A list of information needed for computing predicted standard errors.

Examples

```
library(survival)

colondeaths <- colon[colon$etype == 2, ]
split_colon <- split(colondeaths, colondeaths$rx)
colon_arm1 <- split_colon$Obs

colon1ph <- coxph(Surv(time, status) ~ factor(extent) + nodes + strata(surg)
                   + factor(differ),
                   colon_arm1,
                   x = TRUE, model = TRUE)
surv_pred_info(colon1ph)
```

Index

cox_KM_plots, 2
cox_times_table, 3

get_confint, 5

make_coxph, 6
make_coxph(), *10*

predict_one_coxfit, 7
predict_one_coxfit(), *2*
predict_se, 8
predict_se(), *11*
prep_coxfit, 9
prep_coxfit(), *7*

shine_coxph, 9
shine_coxph(), *2*, *6*
surv_pred_info, 11
surv_pred_info(), *8*
survfit.coxph(), *8*, *11*
survival::coxph(), *6*, *10*
survival::finegray(), *10*