Package: afttest (via r-universe)

September 3, 2024

Type Package

Title Model diagnostics for accelerated failure time models

Version 4.3.3

Date 2024-05-09 EDT

Maintainer Woojung Bae <matt.woojung@gmail.com>

Description A collection of model checking methods for semiparametric accelerated failure time (AFT) models under the rank-based approach. For the (computational) efficiency, Gehan's weight is used. It provides functions to verify whether the observed data fit the specific model assumptions such as a functional form of each covariate, a link function, and an omnibus test. The p-value offered in this package is based on the Kolmogorov-type supremum test and the variance of the proposed test statistics is estimated through the re-sampling method. Furthermore, a graphical technique to compare the shape of the observed residual to a number of the approximated realizations is provided.

Imports survival, aftgee, ggplot2, gridExtra

LinkingTo Rcpp, RcppArmadillo

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

Depends R (>= 3.4.0)

Config/testthat/edition 3

LazyData true

License GPL (>= 3)

URL https://github.com/WooJungBae/afttest

BugReports https://github.com/WooJungBae/afttest/issues

Encoding UTF-8

Language en-US

RoxygenNote 7.3.1

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

RemoteUrl https://github.com/woojungbae/afttest

RemoteRef HEAD

RemoteSha 510ce77a458ddc0e70771c33aede19589730e495

Contents

`afttest` *afttest*

Description

afttest

Usage

```
afttest(
  formula,
  data,
  path = 200,
  testType = "omni",
  eqType = "mns",
  optimType = "DFSANE",
  form = 1,
  pathsave = 50
)
```

Arguments

`formula`	A formula expression, of the form `response ~ predictors`. The `response` is a `Surv` object object with right censoring. See the documentation of `lm`, `coxph` and `formula` for details.
`data`	An optional data frame in which to interpret the variables occurring in the formula.
`path`	An integer value specifies the number of approximated processes. The default is given by 200.
`testType`	A character string specifying the type of the test. The following are permitted: `omni` an omnibus test `link` a link function test

`form` a functional form

`eqType` A character string specifying the type of the estimating equation used to obtain the regression parameters. The readers are refered to the **aftgee** package for details. The following are permitted:

`mns` Regression parameters are estimated by iterating the monotonic non-smoothed Gehan-based estimating equations.

`mis` Regression parameters are estimated by iterating the monotonic smoothed Gehan-based estimating equations.

`optimType` A character string specifying the type of the optimization method. The following are permitted:

`DFSANE` See the documentation of **BB** packages for details.

`Nelder-Mead` See the documentation of `optim` for details.

`BFGS` See the documentation of `optim` for details.

`CG` See the documentation of `optim` for details.

`L-BFGS-B` See the documentation of `optim` for details.

`SANN` See the documentation of `optim` for details.

`Brent` See the documentation of `optim` for details.

`form` A character string specifying the covariate which will be tested. The argument form is necessary only if `testType` is `form`. The default option for `form` is given by "1", which represents the first covariate in the formula argument.

`pathsave` An integer value specifies he number of paths saved among all the paths. The default is given by 50. Note that it requires a lot of memory if save all sampled paths (N by N matrix for each path andso path*N*N elements)

Value

`afttest` returns an object of class `afttest`. An object of class `afttest` is a list containing at least the following components:

beta a vector of beta estimates based on `aftsrr`

SE_process estimated standard error of the observed process

obs_process observed process

app_process approximated process

obs_std_process standardized observed process

app_std_process standardized approximated processes

p_value obtained by the unstandardized test

p_std_value obtained by the standardized test

DF a data frame of observed failure time, right censoring indicator, covariates (scaled), time-transformed residual based on beta estimates

path the number of sample paths

eqType eqType

testType testType

optimType optimType

For an omnibus test, the observed process and the realizations are composed of the n by n matrix that rows represent the t and columns represent the x in the time-transformed residual order.The observed process and the simulated processes for checking a functional form and a link function are given by the n by 1 vector which is a function of x in the time-transformed residual order.

Examples

```
## Simulate data from an AFT model
library(afttest)
library(survival)
datgen <- function(n = 100) {
  z1 <- rbinom(n, 1, 0.5)
  z2 <- rnorm(n)
  e <- rnorm(n)
  tt <- exp(2 + z1 + z2 + e)
  cen <- runif(n, 0, 100)
  data.frame(Time = pmin(tt, cen), status = 1 * (tt < cen),
             z1 = z1, z2 = z2, id = 1:n)
}
set.seed(0)
simdata <- datgen(n = 20)
result <- afttest(Surv(Time, status) ~ z1 + z2, optimType = "DFSANE",
                  data = simdata, testType="link", eqType="mns")
summary(result)
# afttestplot(result)
```

`afttestplot` *afttestplot*

Description

afttestplot

Usage

```
afttestplot(object, path = 50, stdType = "std", quantile = NULL)
```

Arguments

`object`	is a `afttest` fit
`path`	A numeric value specifies the number of approximated processes plotted The default is set to be 100.
`stdType`	A character string specifying if the graph is based on the unstandardized test statistics or standardized test statistics The default is set to be "std".
`quantile`	A numeric vector specifies 5 of five quantiles within the range [0,1]. The default is set to be c(0.1,0.25,0.5,0.75,0.9).

Value

afttestplot returns a plot based on the testType:

omni an object of the omnibus test is the form of n by n matrix, some quantiles of x, which are used in weight, are plotted for graphs, i.e. 0%, 10%, 25%, 40%, 50%, 60%, 75%, 90%, and 100% are used.

link an object of the link function test is the form of n by 1 matrix

form an object of the functional form test is the form of n by 1 matrix

See the documentation of **ggplot2** and **gridExtra** for details.\

Examples

```
## Simulate data from an AFT model
library(afttest)
library(survival)
datgen <- function(n = 100) {
  z1 <- rbinom(n, 1, 0.5)
  z2 <- rnorm(n)
  e <- rnorm(n)
  tt <- exp(2 + z1 + z2 + e)
  cen <- runif(n, 0, 100)
  data.frame(Time = pmin(tt, cen), status = 1 * (tt < cen),
             z1 = z1, z2 = z2, id = 1:n)
}
set.seed(0)
simdata <- datgen(n = 20)
result <- afttest(Surv(Time, status) ~ z1 + z2, optimType = "DFSANE",
                  data = simdata, testType="link", eqType="mns")
# summary(result)
afttestplot(result)
```

---

print.afttest *print.afttest*

Description

print.afttest

Usage

```
## S3 method for class 'afttest'
print(x, ...)
```

Arguments

x	is a afttest fit.
...	other options.

Value

print.afttest returns a summary of a afttest fit:

Examples

```
## Simulate data from an AFT model
library(afttest)
library(survival)
datgen <- function(n = 100) {
  z1 <- rbinom(n, 1, 0.5)
  z2 <- rnorm(n)
  e <- rnorm(n)
  tt <- exp(2 + z1 + z2 + e)
  cen <- runif(n, 0, 100)
  data.frame(Time = pmin(tt, cen), status = 1 * (tt < cen),
             z1 = z1, z2 = z2, id = 1:n)
}
set.seed(0)
simdata <- datgen(n = 20)
result <- afttest(Surv(Time, status) ~ z1 + z2, optimType = "DFSANE",
                  data = simdata, testType="link", eqType="mns")
summary(result)
# afttestplot(result)
```

---

summary.afttest *summary.afttest*

Description

summary.afttest

Usage

```
## S3 method for class 'afttest'
summary(object, ...)
```

Arguments

object	is a afttest fit.
...	other options.

Value

summary.afttest returns a summary of a afttest fit:

Examples

```
## Simulate data from an AFT model
library(afttest)
library(survival)
datgen <- function(n = 100) {
  z1 <- rbinom(n, 1, 0.5)
  z2 <- rnorm(n)
  e <- rnorm(n)
  tt <- exp(2 + z1 + z2 + e)
  cen <- runif(n, 0, 100)
  data.frame(Time = pmin(tt, cen), status = 1 * (tt < cen),
             z1 = z1, z2 = z2, id = 1:n)
}
set.seed(0)
simdata <- datgen(n = 20)
result <- afttest(Surv(Time, status) ~ z1 + z2, optimType = "DFSANE",
                  data = simdata, testType="link", eqType="mns")
summary(result)
# afttestplot(result)
```

Index