Package 'afttest'

February 15, 2026

Type Package

Title Model Diagnostics for Accelerated Failure Time Models

Version 4.5.2.1

Date 2026-2-15 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. See the following references; A general model-checking procedure for semiparametric accelerated failure time models, Statistics and Computing, 34 (3), 117 <doi:10.1007/s11222-024-10431-7>; Diagnostics for semiparametric accelerated failure time models with R package 'afttest', arXiv, <doi:10.48550/arXiv.2511.09823>.

Imports survival, aftgee, ggplot2, gridExtra

LinkingTo Rcpp, RcppArmadillo

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

Depends R (>= 3.4.0)

Config/testthat/edition 3

License GPL (>= 3)

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

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

Encoding UTF-8

Language en-US

RoxygenNote 7.3.2

NeedsCompilation yes

Author Woojung Bae [aut, cre] (ORCID: <https://orcid.org/0000-0001-6760-9900>),
Dongrak Choi [aut] (ORCID: <https://orcid.org/0000-0003-3280-3329>),
Jun Yan [aut] (ORCID: <https://orcid.org/0000-0003-4401-7296>),
Sangwook Kang [aut] (ORCID: <https://orcid.org/0000-0003-2658-481X>)

Repository CRAN

Date/Publication 2026-02-15 22:50:02 UTC

Contents

`afttest` *Model Diagnostics for Semiparametric AFT Models*

Description

Performs model-checking procedures for a semiparametric AFT model. This is a generic function with methods for formulas and fitted objects from the ‘aftgee‘ package.

Usage

```
afttest(object, ...)

## S3 method for class 'formula'
afttest(
  object,
  data,
  npath = 200,
  testType = "omnibus",
  estMethod = "rr",
  eqType = "ns",
  covTested = 1,
  npathsave = 50,
  linApprox = TRUE,
  seed = NULL,
  ...
)

## S3 method for class 'aftsrr'
```

```
afttest(
  object,
  data,
  npath = 200,
  testType = "omnibus",
  eqType = "ns",
  covTested = 1,
  npathsave = 50,
  linApprox = TRUE,
  seed = NULL,
  ...
)

## S3 method for class 'aftgee'
afttest(
  object,
  data,
  npath = 200,
  testType = "omnibus",
  eqType = "ls",
  covTested = 1,
  npathsave = 50,
  linApprox = TRUE,
  seed = NULL,
  ...
)
```

Arguments

`object`	A formula expression, of the form `response ~ predictors`. The `response` is a `Surv` object with right censoring. See the documentation of `lm`, `coxph` and `formula` for details.
`...`	Other arguments passed to methods.
`data`	An optional data frame in which to interpret the variables occurring in the formula.
`npath`	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: `omnibus` an omnibus test `link` a link function test `covForm` a functional form of a covariate
`estMethod`	A character string specifying the type of the estimator used. The readers are refered to the **aftgee** package for details. The following are permitted: `ls` Least-Squares Approach for Accelerated Failure Time with Generalized Estimating Equation `rr` Accelerated Failure Time with Smooth Rank Regression

`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:

- `ns` Regression parameters are estimated by directly solving the nonsmooth estimating equations.
- `is` Regression parameters are estimated by directly solving the induced-smoothing estimating equations.

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

`npathsave` 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 npath andso npath*N*N elements)

`linApprox` A logical value. If `TRUE`, the multiplier bootstrap is computed using the asymptotic linear approximation, which is significantly faster. If `FALSE`, the estimating equations are solved numerically for each bootstrap replication. Defaults to `TRUE`.

`seed` An optional integer specifying the random seed for reproducibility.

Value

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

beta a vector of beta estimates based on `estMethod`

hypothesis null hypothesis for each `testType`

SE_process estimated standard error of the observed process

obs_process observed process

apprx_process approximated process

obs_std_process standardized observed process

apprx_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

npath the number of sample paths

testType testType

eqType eqType

estMethod estMethod

npathsave npathsave

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

```
datgen <- function(n = 100) {
  z1 <- rbinom(n, 1, 0.5)
  z2 <- rnorm(n)
  e <- rnorm(n)
  tt <- exp(2 + z1 + z2 + 0.5*z2^{2}+ 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(1)
simdata = datgen(300)

# linApprox = TRUE
result = afttest(object = Surv(Time, status) ~ z1 + z2, data = simdata,
                 npath = 100, testType = "covForm", estMethod = "rr",
                 eqType = "ns", covTested = "z2", npathsave = 50,
                 linApprox = TRUE, seed = 1)
result$p_value
result$p_std_value
print(result)
summary(result)
plot(result, std = FALSE)
plot(result, std = TRUE)

# result = afttest(object = Surv(Time, status) ~ z1 + z2, data = simdata,
#                  npath = 100, testType = "covForm", estMethod = "rr",
#                  eqType = "is", covTested = "z2", npathsave = 50,
#                  linApprox = TRUE, seed = 1)
# result$p_value
# result$p_std_value
# print(result)
# summary(result)
# plot(result, std = FALSE)
# plot(result, std = TRUE)
#
# result = afttest(object = Surv(Time, status) ~ z1 + z2, data = simdata,
#                  npath = 100, testType = "covForm", estMethod = "ls",
#                  eqType = "ls", covTested = "z2", npathsave = 50,
#                  linApprox = TRUE, seed = 1)
# result$p_value
# result$p_std_value
# print(result)
# summary(result)
# plot(result, std = FALSE)
# plot(result, std = TRUE)
#
# # linApprox = FALSE
# result = afttest(object = Surv(Time, status) ~ z1 + z2, data = simdata,
#                  npath = 100, testType = "covForm", estMethod = "rr",
#                  eqType = "ns", covTested = "z2", npathsave = 50,
#                  linApprox = FALSE, seed = 1)
```

```
# result$p_value
# result$p_std_value
# print(result)
# summary(result)
# plot(result, std = FALSE)
# plot(result, std = TRUE)
#
# result = afttest(object = Surv(Time, status) ~ z1 + z2, data = simdata,
#                  npath = 100, testType = "covForm", estMethod = "rr",
#                  eqType = "is", covTested = "z2", npathsave = 50,
#                  linApprox = FALSE, seed = 1)
# result$p_value
# result$p_std_value
# print(result)
# summary(result)
# plot(result, std = FALSE)
# plot(result, std = TRUE)
#
# result = afttest(object = Surv(Time, status) ~ z1 + z2, data = simdata,
#                  npath = 100, testType = "covForm", estMethod = "ls",
#                  eqType = "ls", covTested = "z2", npathsave = 50,
#                  linApprox = FALSE, seed = 1)
# result$p_value
# result$p_std_value
# print(result)
# summary(result)
# plot(result, std = FALSE)
# plot(result, std = TRUE)
```

`export_Surv` `Surv` *function imported from* `survival`

Description

This function is imported from the `survival` package. See `Surv`.

`plot.afttest` *plot.afttest*

Description

plot.afttest

Usage

```
## S3 method for class 'afttest'
plot(x, npath = 50, std = TRUE, quantile = NULL, ...)
```

Arguments

x	is a afttest fit
npath	A numeric value specifies the number of approximated processes plotted. The default is set to be 100.
std	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).
...	for future extension

Value

plot.afttest returns a plot based on the testType:

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

covForm an x of the functional form test is the form of n by 1 matrix

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

Examples

```
datgen <- function(n = 100) {
  z1 <- rbinom(n, 1, 0.5)
  z2 <- rnorm(n)
  e <- rnorm(n)
  tt <- exp(2 + z1 + z2 + 0.5*z2^{2}+ 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(1)
simdata = datgen(300)

# linApprox = TRUE
result = afttest(object = Surv(Time, status) ~ z1 + z2, data = simdata,
                 npath = 100, testType = "covForm", estMethod = "rr",
                 eqType = "ns", covTested = "z2", npathsave = 50,
                 linApprox = TRUE, seed = 1)
result$p_value
result$p_std_value
print(result)
summary(result)
plot(result, std = FALSE)
plot(result, std = TRUE)

# result = afttest(object = Surv(Time, status) ~ z1 + z2, data = simdata,
```

```
#                   npath = 100, testType = "covForm", estMethod = "rr",
#                   eqType = "is", covTested = "z2", npathsave = 50,
#                   linApprox = TRUE, seed = 1)
# result$p_value
# result$p_std_value
# print(result)
# summary(result)
# plot(result, std = FALSE)
# plot(result, std = TRUE)
#
# result = afttest(object = Surv(Time, status) ~ z1 + z2, data = simdata,
#                   npath = 100, testType = "covForm", estMethod = "ls",
#                   eqType = "ls", covTested = "z2", npathsave = 50,
#                   linApprox = TRUE, seed = 1)
# result$p_value
# result$p_std_value
# print(result)
# summary(result)
# plot(result, std = FALSE)
# plot(result, std = TRUE)
#
# # linApprox = FALSE
# result = afttest(object = Surv(Time, status) ~ z1 + z2, data = simdata,
#                   npath = 100, testType = "covForm", estMethod = "rr",
#                   eqType = "ns", covTested = "z2", npathsave = 50,
#                   linApprox = FALSE, seed = 1)
# result$p_value
# result$p_std_value
# print(result)
# summary(result)
# plot(result, std = FALSE)
# plot(result, std = TRUE)
#
# result = afttest(object = Surv(Time, status) ~ z1 + z2, data = simdata,
#                   npath = 100, testType = "covForm", estMethod = "rr",
#                   eqType = "is", covTested = "z2", npathsave = 50,
#                   linApprox = FALSE, seed = 1)
# result$p_value
# result$p_std_value
# print(result)
# summary(result)
# plot(result, std = FALSE)
# plot(result, std = TRUE)
#
# result = afttest(object = Surv(Time, status) ~ z1 + z2, data = simdata,
#                   npath = 100, testType = "covForm", estMethod = "ls",
#                   eqType = "ls", covTested = "z2", npathsave = 50,
#                   linApprox = FALSE, seed = 1)
# result$p_value
# result$p_std_value
# print(result)
# summary(result)
# plot(result, std = FALSE)
```

```
# plot(result, std = TRUE)
```

---

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

```
datgen <- function(n = 100) {
  z1 <- rbinom(n, 1, 0.5)
  z2 <- rnorm(n)
  e <- rnorm(n)
  tt <- exp(2 + z1 + z2 + 0.5*z2^{2}+ 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(1)
simdata = datgen(300)

# linApprox = TRUE
result = afttest(object = Surv(Time, status) ~ z1 + z2, data = simdata,
                 npath = 100, testType = "covForm", estMethod = "rr",
                 eqType = "ns", covTested = "z2", npathsave = 50,
                 linApprox = TRUE, seed = 1)
result$p_value
result$p_std_value
print(result)
summary(result)
plot(result, std = FALSE)
plot(result, std = TRUE)
```

```
# result = afttest(object = Surv(Time, status) ~ z1 + z2, data = simdata,
#                  npath = 100, testType = "covForm", estMethod = "rr",
#                  eqType = "is", covTested = "z2", npathsave = 50,
#                  linApprox = TRUE, seed = 1)
# result$p_value
# result$p_std_value
# print(result)
# summary(result)
# plot(result, std = FALSE)
# plot(result, std = TRUE)
#
# result = afttest(object = Surv(Time, status) ~ z1 + z2, data = simdata,
#                  npath = 100, testType = "covForm", estMethod = "ls",
#                  eqType = "ls", covTested = "z2", npathsave = 50,
#                  linApprox = TRUE, seed = 1)
# result$p_value
# result$p_std_value
# print(result)
# summary(result)
# plot(result, std = FALSE)
# plot(result, std = TRUE)
#
# # linApprox = FALSE
# result = afttest(object = Surv(Time, status) ~ z1 + z2, data = simdata,
#                  npath = 100, testType = "covForm", estMethod = "rr",
#                  eqType = "ns", covTested = "z2", npathsave = 50,
#                  linApprox = FALSE, seed = 1)
# result$p_value
# result$p_std_value
# print(result)
# summary(result)
# plot(result, std = FALSE)
# plot(result, std = TRUE)
#
# result = afttest(object = Surv(Time, status) ~ z1 + z2, data = simdata,
#                  npath = 100, testType = "covForm", estMethod = "rr",
#                  eqType = "is", covTested = "z2", npathsave = 50,
#                  linApprox = FALSE, seed = 1)
# result$p_value
# result$p_std_value
# print(result)
# summary(result)
# plot(result, std = FALSE)
# plot(result, std = TRUE)
#
# result = afttest(object = Surv(Time, status) ~ z1 + z2, data = simdata,
#                  npath = 100, testType = "covForm", estMethod = "ls",
#                  eqType = "ls", covTested = "z2", npathsave = 50,
#                  linApprox = FALSE, seed = 1)
# result$p_value
# result$p_std_value
# print(result)
# summary(result)
```

```
# plot(result, std = FALSE)
# plot(result, std = TRUE)
```

---

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

```
datgen <- function(n = 100) {
  z1 <- rbinom(n, 1, 0.5)
  z2 <- rnorm(n)
  e <- rnorm(n)
  tt <- exp(2 + z1 + z2 + 0.5*z2^{2}+ 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(1)
simdata = datgen(300)

# linApprox = TRUE
result = afttest(object = Surv(Time, status) ~ z1 + z2, data = simdata,
                 npath = 100, testType = "covForm", estMethod = "rr",
                 eqType = "ns", covTested = "z2", npathsave = 50,
                 linApprox = TRUE, seed = 1)
result$p_value
result$p_std_value
print(result)
summary(result)
plot(result, std = FALSE)
plot(result, std = TRUE)
```

```
# result = afttest(object = Surv(Time, status) ~ z1 + z2, data = simdata,
#                  npath = 100, testType = "covForm", estMethod = "rr",
#                  eqType = "is", covTested = "z2", npathsave = 50,
#                  linApprox = TRUE, seed = 1)
# result$p_value
# result$p_std_value
# print(result)
# summary(result)
# plot(result, std = FALSE)
# plot(result, std = TRUE)
#
# result = afttest(object = Surv(Time, status) ~ z1 + z2, data = simdata,
#                  npath = 100, testType = "covForm", estMethod = "ls",
#                  eqType = "ls", covTested = "z2", npathsave = 50,
#                  linApprox = TRUE, seed = 1)
# result$p_value
# result$p_std_value
# print(result)
# summary(result)
# plot(result, std = FALSE)
# plot(result, std = TRUE)
#
# # linApprox = FALSE
# result = afttest(object = Surv(Time, status) ~ z1 + z2, data = simdata,
#                  npath = 100, testType = "covForm", estMethod = "rr",
#                  eqType = "ns", covTested = "z2", npathsave = 50,
#                  linApprox = FALSE, seed = 1)
# result$p_value
# result$p_std_value
# print(result)
# summary(result)
# plot(result, std = FALSE)
# plot(result, std = TRUE)
#
# result = afttest(object = Surv(Time, status) ~ z1 + z2, data = simdata,
#                  npath = 100, testType = "covForm", estMethod = "rr",
#                  eqType = "is", covTested = "z2", npathsave = 50,
#                  linApprox = FALSE, seed = 1)
# result$p_value
# result$p_std_value
# print(result)
# summary(result)
# plot(result, std = FALSE)
# plot(result, std = TRUE)
#
# result = afttest(object = Surv(Time, status) ~ z1 + z2, data = simdata,
#                  npath = 100, testType = "covForm", estMethod = "ls",
#                  eqType = "ls", covTested = "z2", npathsave = 50,
#                  linApprox = FALSE, seed = 1)
# result$p_value
# result$p_std_value
# print(result)
```

```
# summary(result)
# plot(result, std = FALSE)
# plot(result, std = TRUE)
```

Index