Package 'gofedf'

April 12, 2026

Title Goodness of Fit Tests Based on Empirical Distribution Functions

Version 1.1.0

Description Routines that allow the user to run goodness of fit tests based on empirical distribution functions for formal model evaluation in a general likelihood model. In addition, functions are provided to test if a sample follows Normal or Gamma distributions, validate the normality assumptions in a linear model, and examine the appropriateness of a Gamma distribution in generalized linear models with various link functions. Michael Arthur Stephens (1976) <http://www.jstor.org/stable/2958206>.

License GPL (>= 3)

URL https://github.com/pnickchi/gofedf

BugReports https://github.com/pnickchi/gofedf/issues

Encoding UTF-8

RoxygenNote 7.3.3

Depends R (>= 2.10)

Imports stats, CompQuadForm, MASS, glm2, statmod

NeedsCompilation no

Author Richard Lockhart [aut],
Payman Nickchi [aut, cre]

Maintainer Payman Nickchi <payman.nickchi@gmail.com>

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

VignetteBuilder knitr

Repository CRAN

Date/Publication 2026-04-12 02:20:08 UTC

Contents

`IGMLE` *Compute the maximum likelihood estimate of parameters in Inverse Gaussian distribution with weighted observations.*

Description

This function is used in `testYourModel` function for example purposes.

Usage

```
IGMLE(obs, ...)
```

Arguments

`obs`	a numeric vector of sample observations.
`...`	a list of additional parameters to define the likelihood.

Value

The function compute the MLE of parameters in Inverse Gaussian distribution and returns a vector of estimates. The first and second elements of the vector are MLE of the mean and shape, respectively.

`IGPIT` *Compute the probability integral transformed values for a sample from Inverse Gaussian distribution.*

Description

This function is used in `testYourModel` function for example purposes.

Usage

```
IGPIT(obs, ...)
```

Arguments

obs	A numeric vector of sample observations.
...	A list of additional parameters to define the likelihood.

Value

A numeric vector of probability integral transformed values of sample observations.

IGScore — *Compute the score function of the Inverse Gaussian distribution based on a sample.*

Description

This function is used in testYourModel function for example purposes.

Usage

```
IGScore(obs, ...)
```

Arguments

obs	a numeric vector of sample observations.
...	a list of additional parameters to define the likelihood.

Value

The score matrix with n rows (number of sample observations) and 2 columns (mean and shape).

---

testExponential — *Apply Goodness of Fit Test for Exponential Distribution*

Description

Performs the goodness-of-fit test based on empirical distribution function to check if an i.i.d sample follows an Exponential distribution.

Usage

```
testExponential(
  x,
  discretize = FALSE,
  ngrid = length(x),
  gridpit = FALSE,
  hessian = FALSE,
  method = "cvm",
  weight_function = NULL
)
```

Arguments

x	a non-empty numeric vector of sample data.
discretize	If TRUE, the covariance function of $W_n(u)$ process is evaluated at some data points (see ngrid and gridpit), and the integral equation is replaced by a matrix equation. If FALSE (the default value), the covariance function is first estimated, and then the integral equation is solved to find the eigenvalues. The results of our simulations recommend using the estimated covariance for solving the integral equation. The parameters ngrid, gridpit, and hessian are only relevant when discretize = TRUE.
ngrid	the number of equally spaced points to discretize the (0,1) interval for computing the covariance function.
gridpit	logical. If TRUE (the default value), the parameter ngrid is ignored and (0,1) interval is divided based on probability integral transforms or PITs obtained from the sample. If FALSE, the interval is divided into ngrid equally spaced points for computing the covariance function.
hessian	logical. If TRUE the Fisher information matrix is estimated by the observed Hessian Matrix based on the sample. If FALSE (the default value) the Fisher information matrix is estimated by the variance of the observed score matrix.
method	a character string indicating which goodness-of-fit statistic is to be computed. The default value is 'cvm' for the Cramer-von-Mises statistic. Other options include 'ad' for the Anderson-Darling statistic, 'both' to compute both cvm and ad statistics, and 'user' for custom weight function. See weight_function for details about custom weight function.
weight_function	a function representing the weight function $w(u)$ used to compute the weighted Cramér-von Mises statistic when method = 'user'. The function must take a numeric vector $u \in (0, 1)$ as input and return a numeric vector of the same length. The statistic is computed as $$T_n = n \int_0^1 w^2(u) \left(F_n(u) - u\right)^2 du$$ where $w^2(u)$ is computed internally by squaring the supplied function. The default value is NULL and when method is 'cvm', 'ad', or 'both' the weight_function is ignored. When method = 'user', this argument must be provided, otherwise an error is returned.

Value

A list of two containing the following components:

- Statistic: the value of goodness-of-fit statistic.
- p-value: the approximate p-value for the goodness-of-fit test. if method = 'cvm' or method = 'ad', it returns a numeric value for the statistic and p-value. If method = 'both', it returns a numeric vector with two elements and one for each statistic. If method = 'user' it returns the weighted statistic.

Examples

```
set.seed(123)
n <- 50
sim_data <- rexp(n, rate = 2)
testExponential(x = sim_data)

# Example to show custom weight function
w_cvm <- function(u) rep(1, length(u))
testExponential(x = sim_data, method = 'user', weight_function = w_cvm)
```

`testGamma` *Apply Goodness of Fit Test for Gamma Distribution*

Description

Performs the goodness-of-fit test based on empirical distribution function to check if an i.i.d sample follows a Gamma distribution.

Usage

```
testGamma(
  x,
  discretize = FALSE,
  ngrid = length(x),
  gridpit = FALSE,
  hessian = FALSE,
  rate = TRUE,
  method = "cvm",
  weight_function = NULL
)
```

Arguments

`x`	a non-empty numeric vector of sample data.
`discretize`	If `TRUE`, the covariance function of $W_n(u)$ process is evaluated at some data points (see `ngrid` and `gridpit`), and the integral equation is replaced by a matrix equation. If `FALSE` (the default value), the covariance function is first estimated, and then the integral equation is solved to find the eigenvalues. The results of our simulations recommend using the estimated covariance for solving the integral equation. The parameters `ngrid`, `gridpit`, and `hessian` are only relevant when `discretize = TRUE`.
`ngrid`	the number of equally spaced points to discretize the (0,1) interval for computing the covariance function.
`gridpit`	logical. If `TRUE` (the default value), the parameter ngrid is ignored and (0,1) interval is divided based on probability integral transforms or PITs obtained from the sample. If `FALSE`, the interval is divided into ngrid equally spaced points for computing the covariance function.

`hessian` logical. If `TRUE` the Fisher information matrix is estimated by the observed Hessian Matrix based on the sample. If `FALSE` (the default value) the Fisher information matrix is estimated by the variance of the observed score matrix.

`rate` logical. If `TRUE` (the default value), the rate is estimated in Gamma distribution. If `FALSE`, scale is estimated. See `GammaDist` for more details.

`method` a character string indicating which goodness-of-fit statistic is to be computed. The default value is 'cvm' for the Cramer-von-Mises statistic. Other options include 'ad' for the Anderson-Darling statistic, 'both' to compute both cvm and ad statistics, and 'user' for custom weight function. See weight_function for details about custom weight function.

`weight_function`

a function representing the weight function $w(u)$ used to compute the weighted Cramér-von Mises statistic when `method = 'user'`. The function must take a numeric vector $u \in (0, 1)$ as input and return a numeric vector of the same length. The statistic is computed as

$$T_n = n \int_0^1 w^2(u) \left(F_n(u) - u\right)^2 du$$

where $w^2(u)$ is computed internally by squaring the supplied function. The default value is `NULL` and when `method` is `'cvm'`, `'ad'`, or `'both'` the weight_function is ignored. When `method = 'user'`, this argument must be provided, otherwise an error is returned.

Value

A list of two containing the following components:

- Statistic: the value of goodness-of-fit statistic.
- p-value: the approximate p-value for the goodness-of-fit test. if method = 'cvm' or method = 'ad', it returns a numeric value for the statistic and p-value. If method = 'both', it returns a numeric vector with two elements and one for each statistic. If method = 'user' it returns the weighted statistic.

Examples

```
set.seed(123)
sim_data <- rgamma(n = 50, shape = 3)
# Cramer-von-Mises
testGamma(x = sim_data)
# Anderson-Darling
testGamma(x = sim_data, method = 'ad')
# Example to show custom weight function
w_cvm <- function(u) rep(1, length(u))
testGamma(x = sim_data, method = 'user', weight_function = w_cvm)
w_ad <- function(u) 1 / sqrt(u * (1 - u))
testGamma(x = sim_data, method = 'user', weight_function = w_ad)


sim_data <- runif(n = 50)
```

```
testGamma(x = sim_data)
```

`testGLMGamma` *Apply Goodness of Fit Test to the Residuals of a Generalized Linear Model with Gamma Link Function*

Description

`testGLMGamma` is used to check the validity of Gamma assumption for the response variable when fitting generalized linear model. Common link functions in `glm` can be used here.

Usage

```
testGLMGamma(
  x,
  y,
  fit = NULL,
  l = "log",
  discretize = FALSE,
  ngrid = length(y),
  gridpit = TRUE,
  hessian = FALSE,
  start.value = NULL,
  control = NULL,
  method = "cvm",
  weight_function = NULL
)
```

Arguments

`x`	is either a numeric vector or a design matrix. In the design matrix, rows indicate observations and columns presents covariats.
`y`	is a vector of numeric values with the same number of observations or number of rows as x.
`fit`	is an object of class `glm` and its default value is NULL. If a fit of class `glm` is provided, the arguments `x`, `y`, and `l` will be ignored. We recommend using `glm2` function from `glm2` package since it provides better convergence while optimizing the likelihood to estimate coefficients of the model by IWLS method. It is required to return design matrix by `x = TRUE` in `glm` or `glm2` function. For more information on how to do this, refer to the help documentation for the `glm` or `glm2` function.
`l`	a character vector indicating the link function that should be used for Gamma family. Acceptable link functions for Gamma family are inverse, identity and log. For more details see `Gamma` from stats package.

`discretize` If `TRUE`, the covariance function of $W_n(u)$ process is evaluated at some data points (see `ngrid` and `gridpit`), and the integral equation is replaced by a matrix equation. If `FALSE` (the default value), the covariance function is first estimated, and then the integral equation is solved to find the eigenvalues. The results of our simulations recommend using the estimated covariance for solving the integral equation. The parameters `ngrid`, `gridpit`, and `hessian` are only relevant when `discretize = TRUE`.

`ngrid` the number of equally spaced points to discretize the (0,1) interval for computing the covariance function.

`gridpit` logical. If `TRUE` (the default value), the parameter ngrid is ignored and (0,1) interval is divided based on probability integral transforms or PITs obtained from the sample. If `FALSE`, the interval is divided into ngrid equally spaced points for computing the covariance function.

`hessian` logical. If `TRUE` the Fisher information matrix is estimated by the observed Hessian Matrix based on the sample. If `FALSE` (the default value) the Fisher information matrix is estimated by the variance of the observed score matrix.

`start.value` a numeric value or vector. This is the same as `start` argument in `glm` or `glm2`. The value is a starting point in iteratively reweighted least squares (IRLS) algorithm for estimating the MLE of coefficients in the model.

`control` a list of parameters to control the fitting process in `glm` or `glm2` function. For more details, see `glm.control`.

`method` a character string indicating which goodness-of-fit statistic is to be computed. The default value is 'cvm' for the Cramer-von-Mises statistic. Other options include 'ad' for the Anderson-Darling statistic, 'both' to compute both cvm and ad statistics, and 'user' for custom weight function. See weight_function for details about custom weight function.

`weight_function`

a function representing the weight function $w(u)$ used to compute the weighted Cramér-von Mises statistic when `method = 'user'`. The function must take a numeric vector $u \in (0, 1)$ as input and return a numeric vector of the same length. The statistic is computed as

$$T_n = n \int_0^1 w^2(u) \left(F_n(u) - u\right)^2 du$$

where $w^2(u)$ is computed internally by squaring the supplied function. The default value is `NULL` and when `method` is `'cvm'`, `'ad'`, or `'both'` the weight_function is ignored. When `method = 'user'`, this argument must be provided, otherwise an error is returned.

Value

A list of two containing the following components:

- Statistic: the value of goodness-of-fit statistic.
- p-value: the approximate p-value for the goodness-of-fit test. if method = 'cvm' or method = 'ad', it returns a numeric value for the statistic and p-value. If method = 'both', it returns a numeric vector with two elements and one for each statistic. If method = 'user' it returns the weighted statistic.

Examples

```
set.seed(123)
n <- 50
p <- 5
x <- matrix( rnorm(n*p, mean = 10, sd = 0.1), nrow = n, ncol = p)
b <- runif(p)
e <- rgamma(n, shape = 3)
y <- exp(x %*% b) * e
testGLMGamma(x, y, l = 'log')
myfit <- glm(y ~ x, family = Gamma('log'), x = TRUE, y = TRUE)
testGLMGamma(fit = myfit)
# Example for custom weight function
w_cvm <- function(u) rep(1, length(u))
testGLMGamma(fit = myfit, method = 'user', weight_function = w_cvm)
```

`testLMNormal` *Apply Goodness of Fit Test to Residuals of a Linear Model*

Description

`testLMNormal` is used to check the normality assumption of residuals in a linear model. This function can take the response variable and design matrix, fit a linear model, and apply the goodness-of-fit test. Conveniently, it can take an object of class "lm" and directly applies the goodness-of-fit test. The function returns a goodness-of-fit statistic along with an approximate p-value.

Usage

```
testLMNormal(
  x,
  y,
  fit = NULL,
  discretize = FALSE,
  ngrid = length(y),
  gridpit = TRUE,
  hessian = FALSE,
  method = "cvm",
  weight_function = NULL
)
```

Arguments

x	is either a numeric vector or a design matrix. In the design matrix, rows indicate observations and columns presents covariates.
y	is a vector of numeric values with the same number of observations or number of rows as x.

`fit` an object of class "lm" returned by `lm` function in `stats` package. The default value of fit is NULL. If any object is provided, `x` and `y` will be ignored and the class of object is checked. If you pass an object to `fit` make sure to return the design matrix by setting `x = TRUE` and the response variable by setting in `y = TRUE` in `lm` function. To read more about this see the help documentation for `lm` function or see the example below.

`discretize` If `TRUE`, the covariance function of $W_n(u)$ process is evaluated at some data points (see `ngrid` and `gridpit`), and the integral equation is replaced by a matrix equation. If `FALSE` (the default value), the covariance function is first estimated, and then the integral equation is solved to find the eigenvalues. The results of our simulations recommend using the estimated covariance for solving the integral equation. The parameters `ngrid`, `gridpit`, and `hessian` are only relevant when `discretize = TRUE`.

`ngrid` the number of equally spaced points to discretize the (0,1) interval for computing the covariance function.

`gridpit` logical. If `TRUE` (the default value), the parameter ngrid is ignored and (0,1) interval is divided based on probability integral transforms or PITs obtained from the sample. If `FALSE`, the interval is divided into ngrid equally spaced points for computing the covariance function.

`hessian` logical. If `TRUE` the Fisher information matrix is estimated by the observed Hessian Matrix based on the sample. If `FALSE` (the default value) the Fisher information matrix is estimated by the variance of the observed score matrix.

`method` a character string indicating which goodness-of-fit statistic is to be computed. The default value is 'cvm' for the Cramer-von-Mises statistic. Other options include 'ad' for the Anderson-Darling statistic, 'both' to compute both cvm and ad statistics, and 'user' for custom weight function. See weight_function for details about custom weight function.

`weight_function`
a function representing the weight function $w(u)$ used to compute the weighted Cramér-von Mises statistic when `method = 'user'`. The function must take a numeric vector $u \in (0, 1)$ as input and return a numeric vector of the same length. The statistic is computed as

$$T_n = n \int_0^1 w^2(u) \left(F_n(u) - u\right)^2 du$$

where $w^2(u)$ is computed internally by squaring the supplied function. The default value is `NULL` and when `method` is `'cvm'`, `'ad'`, or `'both'` the weight_function is ignored. When `method = 'user'`, this argument must be provided, otherwise an error is returned.

Value

A list of two containing the following components:

- Statistic: the value of goodness-of-fit statistic.

- p-value: the approximate p-value for the goodness-of-fit test. if method = 'cvm' or method = 'ad', it returns a numeric value for the statistic and p-value. If method = 'both', it returns a numeric vector with two elements and one for each statistic. If method = 'user' it returns the weighted statistic.

Examples

```
set.seed(123)
n <- 50
p <- 5
x <- matrix( runif(n*p), nrow = n, ncol = p)
e <- rnorm(n)
b <- runif(p)
y <- x %*% b + e
testLMNormal(x, y)
# Or pass lm.fit object directly:
lm.fit <- lm(y ~ x, x = TRUE, y = TRUE)
testLMNormal(fit = lm.fit)
# Example for custom weight function
w_cvm <- function(u) rep(1, length(u))
testLMNormal(fit = lm.fit, method = 'user', weight_function = w_cvm)
```

---

testNormal *Apply Goodness of Fit Test for Normal Distribution*

Description

Performs the goodness-of-fit test based on empirical distribution function to check if an i.i.d sample follows a Normal distribution.

Usage

```
testNormal(
  x,
  discretize = FALSE,
  ngrid = length(x),
  gridpit = TRUE,
  hessian = FALSE,
  method = "cvm",
  weight_function = NULL
)
```

Arguments

x	a non-empty numeric vector of sample data.

`discretize` If `TRUE`, the covariance function of $W_n(u)$ process is evaluated at some data points (see `ngrid` and `gridpit`), and the integral equation is replaced by a matrix equation. If `FALSE` (the default value), the covariance function is first estimated, and then the integral equation is solved to find the eigenvalues. The results of our simulations recommend using the estimated covariance for solving the integral equation. The parameters `ngrid`, `gridpit`, and `hessian` are only relevant when `discretize = TRUE`.

`ngrid` the number of equally spaced points to discretize the (0,1) interval for computing the covariance function.

`gridpit` logical. If `TRUE` (the default value), the parameter ngrid is ignored and (0,1) interval is divided based on probability integral transforms or PITs obtained from the sample. If `FALSE`, the interval is divided into ngrid equally spaced points for computing the covariance function.

`hessian` logical. If `TRUE` the Fisher information matrix is estimated by the observed Hessian Matrix based on the sample. If `FALSE` (the default value) the Fisher information matrix is estimated by the variance of the observed score matrix.

`method` a character string indicating which goodness-of-fit statistic is to be computed. The default value is 'cvm' for the Cramer-von-Mises statistic. Other options include 'ad' for the Anderson-Darling statistic, 'both' to compute both cvm and ad statistics, and 'user' for custom weight function. See weight_function for details about custom weight function.

`weight_function`

a function representing the weight function $w(u)$ used to compute the weighted Cramér-von Mises statistic when `method = 'user'`. The function must take a numeric vector $u \in (0, 1)$ as input and return a numeric vector of the same length. The statistic is computed as

$$T_n = n \int_0^1 w^2(u) \left(F_n(u) - u\right)^2 du$$

where $w^2(u)$ is computed internally by squaring the supplied function. The default value is `NULL` and when `method` is `'cvm'`, `'ad'`, or `'both'` the weight_function is ignored. When `method = 'user'`, this argument must be provided, otherwise an error is returned.

Value

A list of two containing the following components:

- Statistic: the value of goodness-of-fit statistic.
- p-value: the approximate p-value for the goodness-of-fit test. if method = 'cvm' or method = 'ad', it returns a numeric value for the statistic and p-value. If method = 'both', it returns a numeric vector with two elements and one for each statistic. If method = 'user' it returns the weighted statistic.

Examples

```
set.seed(123)
```

```
sim_data <- rnorm(n = 50)
# Cramer-von-Mises
testNormal(x = sim_data)
# Anderson-Darling
testNormal(x = sim_data, method = 'ad')
# Example to show custom weight function
w_cvm <- function(u) rep(1, length(u))
testNormal(x = sim_data, method = 'user', weight_function = w_cvm)
w_ad <- function(u) 1 / sqrt(u * (1 - u))
testNormal(x = sim_data, method = 'user', weight_function = w_ad)


sim_data <- rgamma(n = 50, shape = 3)
testNormal(x = sim_data)
```

`testYourModel` *Apply the Goodness of Fit Test Based on Empirical Distribution Function to Any Likelihood Model.*

Description

This function applies the goodness-of-fit test based on empirical distribution function. It requires certain inputs depending on whether the model involves parameter estimation or not. If the model is known and there is no parameter estimation, the function requires the probability transformed (or pit) values of the sample. This ought to be a numeric vector. If there is parameter estimation in the model, the function additionally requires the score as a matrix with n rows and p columns, where n is the sample size and p is the number of estimated parameters. The function checks if the sum of columns in score is near zero at the estimated parameter (which is assumed to be the maximum likelihood estimate).

Usage

```
testYourModel(
  pit,
  score = NULL,
  discretize = FALSE,
  ngrid = length(pit),
  gridpit = TRUE,
  precision = 1e-09,
  method = "cvm",
  weight_function = NULL
)
```

Arguments

`pit`	The probability transformed (or pit) values of the sample which ought to be a numeric vector.

`score` The default value is null and refers to no parameter estimation case. If there is parameter estimation, the score must be a matrix with n rows and p columns, where n is the sample size and p is the number of estimated parameters.

`discretize` If `TRUE`, the covariance function of $W_n(u)$ process is evaluated at some data points (see `ngrid` and `gridpit`), and the integral equation is replaced by a matrix equation. If `FALSE` (the default value), the covariance function is first estimated, and then the integral equation is solved to find the eigenvalues. The results of our simulations recommend using the estimated covariance for solving the integral equation. The parameters `ngrid`, `gridpit`, and `hessian` are only relevant when `discretize = TRUE`.

`ngrid` The number of equally spaced points to discretize the (0,1)interval for computing the covariance function.

`gridpit` logical. If `TRUE` (the default value), the parameter ngrid is ignored and (0,1) interval is divided based on probability integral transforms or PITs obtained from the sample. If `FALSE`, the interval is divided into ngrid equally spaced points for computing the covariance function.

`precision` The theory behind goodness-of-fit test based on empirical distribution function (edf) works well if the MLE is indeed the root of derivative of log likelihood function. A precision of 1e-9 (default value) is used to check this. A warning message is generated if the score evaluated at MLE is not close enough to zero.

`method` a character string indicating which goodness-of-fit statistic is to be computed. The default value is 'cvm' for the Cramer-von-Mises statistic. Other options include 'ad' for the Anderson-Darling statistic, 'both' to compute both cvm and ad statistics, and 'user' for custom weight function. See weight_function for details about custom weight function.

`weight_function`

a function representing the weight function $w(u)$ used to compute the weighted Cramér-von Mises statistic when `method = 'user'`. The function must take a numeric vector $u \in (0, 1)$ as input and return a numeric vector of the same length. The statistic is computed as

$$T_n = n \int_0^1 w^2(u) \left(F_n(u) - u\right)^2 du$$

where $w^2(u)$ is computed internally by squaring the supplied function. The default value is `NULL` and when `method` is `'cvm'`, `'ad'`, or `'both'` the weight_function is ignored. When `method = 'user'`, this argument must be provided, otherwise an error is returned.

Value

A list of two containing the following components:

- Statistic: the value of goodness-of-fit statistic.
- p-value: the approximate p-value for the goodness-of-fit test. if method = 'cvm' or method = 'ad', it returns a numeric value for the statistic and p-value. If method = 'both', it returns a numeric vector with two elements and one for each statistic. If method = 'user' it returns the weighted statistic.

Examples

```
# Example: Inverse Gaussian (IG) distribution with weights

# Set the seed to reproduce example.
set.seed(123)

# Set the sample size
n <- 50

# Assign weights
weights <- rep(1.5,n)

# Set mean and shape parameters for IG distribution.
mio        <- 2
lambda     <- 2

# Generate a random sample from IG distribution with weighted shape.
sim_data <- statmod::rinvgauss(n, mean = mio, shape = lambda * weights)

# Compute MLE of parameters, score matrix, and pit values.
theta_hat    <- IGMLE(obs = sim_data,   w = weights)
scoreMatrix  <- IGScore(obs = sim_data, w = weights, mle = theta_hat)
pitvalues    <- IGPIT(obs = sim_data ,  w = weights, mle = theta_hat)

# Apply the goodness-of-fit test.
testYourModel(pit = pitvalues, score = scoreMatrix)

# Example to show custom weight function
w_cvm <- function(u) rep(1, length(u))
testYourModel(pit = pitvalues, score = scoreMatrix, method = 'user', weight_function = w_cvm)
```

Index