

# Package: distfreereg (via r-universe)

June 8, 2026

**Type** Package

**Title** Distribution-Free Goodness-of-Fit Testing for Regression

**Version** 1.2

**Depends** R (>= 4.4)

**Imports** calculus, clue, lme4, methods

**Suggests** knitr, rmarkdown

**VignetteBuilder** knitr, rmarkdown

**Date** 2026-05-05

**Maintainer** Jesse Miller <smallepsilon@proton.me>

**Description** Implements the distribution-free goodness-of-fit regression testing procedure, introduced by Estate Khmaladze (2021, <[doi:10.1007/s10463-021-00786-3](https://doi.org/10.1007/s10463-021-00786-3)>) to test whether or not the mean structure of a parametric model belongs to a specified model family. The test is implemented for general mean functions with minimal distributional assumptions as well as common models (e.g., lm, glm) with the usual model assumptions.

**License** GPL-3

**NeedsCompilation** no

**Author** Jesse Miller [aut, cre] (ORCID:  
<<https://orcid.org/0009-0005-9465-7461>>)

**Config/pak/sysreqs** cmake make

**Repository** <https://small-epsilon.r-universe.dev>

**Date/Publication** 2026-05-08 09:58:47 UTC

**RemoteUrl** <https://github.com/cran/distfreereg>

**RemoteRef** HEAD

**RemoteSha** 11014a53d340821ea17ec58be0c75c5b3d0508f4

## Contents

distfreereg-package . . . . .	2
asymptotics . . . . .	4
coef.distfreereg . . . . .	5
compare . . . . .	5
confint.distfreereg . . . . .	10
distfreereg . . . . .	11
fitted.distfreereg . . . . .	17
formula.distfreereg . . . . .	18
ks.test.compare . . . . .	18
plot.compare . . . . .	19
plot.distfreereg . . . . .	22
predict.distfreereg . . . . .	25
print.compare . . . . .	26
print.distfreereg . . . . .	26
rejection . . . . .	27
residuals.distfreereg . . . . .	29
update.distfreereg . . . . .	29
vcov.distfreereg . . . . .	31

<b>Index</b>	<b>33</b>
--------------	-----------

---

distfreereg-package	<i>Distribution-Free Goodness-of-Fit Testing for Regression</i>
---------------------	---

---

## Description

Implements the distribution-free goodness-of-fit regression testing procedure, introduced by Estate Khmaladze (2021, <doi:10.1007/s10463-021-00786-3>) to test whether or not the mean structure of a parametric model belongs to a specified model family. The test is implemented for general mean functions with minimal distributional assumptions as well as common models (e.g., lm, glm) with the usual model assumptions.

## Details

The DESCRIPTION file:

```

Package:      distfreereg
Type:         Package
Title:        Distribution-Free Goodness-of-Fit Testing for Regression
Version:      1.2
Depends:      R (>= 4.4)
Imports:      calculus, clue, lme4, methods
Suggests:    knitr, rmarkdown
VignetteBuilder: knitr, rmarkdown
Date:         2026-05-05
Authors@R:    person(given = "Jesse", family = "Miller", role = c("aut", "cre"), email = "smallepsilon@proton.me", comm

```

Maintainer: Jesse Miller <smallepsilon@proton.me>  
 Description: Implements the distribution-free goodness-of-fit regression testing procedure, introduced by Estate Khmaladze  
 License: GPL-3  
 Author: Jesse Miller [aut, cre] (ORCID: <<https://orcid.org/0009-0005-9465-7461>>)

## Index of help topics:

asymptotics	Convenience Function for Exploring Asymptotic Behavior and Sample Size Adequacy
coef.distfreereg	Extract Estimated Parameters from 'distfreereg' Objects
compare	Compare the Distributions of Empirical and Theoretical Statistics Used in Distribution-Free Parametric Regression Testing
confint.distfreereg	Calculate Confidence Intervals with a 'distfreereg' Object
distfreereg	Distribution-Free Parametric Regression Testing
distfreereg-package	Distribution-Free Goodness-of-Fit Testing for Regression
fitted.distfreereg	Extract Fitted Values from 'distfreereg' Objects
formula.distfreereg	Extract Formulas from 'distfreereg' Objects
ks.test.compare	Formally Compare Empirical and Theoretical Statistics from a 'compare' Object
plot.compare	Summary and Diagnostic Plots for 'compare' Objects
plot.distfreereg	Summary and Diagnostic Plots for 'distfreereg' Objects
predict.distfreereg	Generate Predicted Values from 'distfreereg' Objects
print.compare	Printing 'compare' Objects
print.distfreereg	Printing 'distfreereg' Objects
rejection	Compute Rejection Rates of a Distribution-Free Test from a 'compare' Object
residuals.distfreereg	Extract Residuals from 'distfreereg' Objects
update.distfreereg	Update 'distfreereg' Objects
vcov.distfreereg	Estimate Parameter Covariance Matrices from 'distfreereg' Objects

Further information is available in the following vignettes:

v1_introduction	An Introduction to the distfreereg Package (source)
v2_compare	Comparing Distributions with the distfreereg Package (source)
v3_plotting	Plotting with the distfreereg Package (source)
v4_parameter-estimation	Parameter Estimation with distfreereg (source)

v5\_advanced-options      Advanced Options for the distfreereg Package (source)

### Author(s)

Jesse Miller [aut, cre] (ORCID: <<https://orcid.org/0009-0005-9465-7461>>)

Maintainer: Jesse Miller <[smallepsilon@proton.me](mailto:smallepsilon@proton.me)>

---

asymptotics      *Convenience Function for Exploring Asymptotic Behavior and Sample Size Adequacy*

---

### Description

This is a convenience function that calls [compare](#) using object to create the true and test model specifications.

### Usage

```
asymptotics(object, ...)
```

### Arguments

object      Object of class distfreereg.  
...      Additional arguments to pass to [compare](#).

### Details

An important step in implementing [distfreereg](#) is determining the plausibility that the sample size of a data set is adequate to produce the desired asymptotic behavior of the simulated statistics. This can be done by calling [compare](#) with the appropriate argument values. Because of the importance of this step, this convenience function automates that call using a single argument, namely the object of class distfreereg in question.

### Value

Object of class compare.

### Warning

The fitted model in object is assumed to be the true model, not an estimation of the true model. Further, the simulation of values requires assuming a particular error distribution. See the "Warnings" section in [compare](#)'s documentation for details.

### Author(s)

Jesse Miller

**See Also**

[distfreereg](#), [compare](#)

---

coef.distfreereg	<i>Extract Estimated Parameters from distfreereg Objects</i>
------------------	--

---

**Description**

This is a [coef](#) method for objects of class `distfreereg`. It extracts the estimated parameters from a model in a `distfreereg` object.

**Usage**

```
## S3 method for class 'distfreereg'
coef(object, ...)
```

**Arguments**

object	Object of class <code>distfreereg</code> .
...	Additional parameters passed to or from other methods. Currently ignored.

**Value**

Numeric vector of estimated model parameters.

**Author(s)**

Jesse Miller

**See Also**

[distfreereg](#), [vcov.distfreereg](#), [confint.distfreereg](#)

---

compare	<i>Compare the Distributions of Empirical and Theoretical Statistics Used in Distribution-Free Parametric Regression Testing</i>
---------	--

---

**Description**

Simulate response data repeatedly with `true_mean` as the mean and `true_covariance` as the covariance structure, each time running [distfreereg](#) on the simulated data. The observed statistics and p-values are saved, as are the simulated statistics from the first replication.

This function is intended to facilitate exploring sample size adequacy and the tests' powers. For a convenient wrapper to expedite investigating sample size adequacy using a `distfreereg` object, see [asymptotics](#).

See the [Comparing Distributions with the distfreereg Package](#) vignette for an introduction.

**Usage**

```
compare(true_mean, true_method = NULL, true_method_args = NULL, true_covariance,
true_X = NULL, true_data = NULL, theta = NULL, n = NULL, reps = 1e3,
prog = reps/10, simulate_args = NULL, err_dist_fun = NULL,
err_dist_args = NULL, keep = NULL, manual = NULL, update_args = NULL, ...)
```

**Arguments**

<code>true_mean</code>	Object specifying the mean structure of the true model. It is used to generate the true values of $Y$ that are passed internally to <code>distfreereg</code> .
<code>true_method</code>	Character vector of length one; specifies the function (e.g., <code>lm</code> ) to use to create a model when <code>true_mean</code> is a formula; should be <code>NULL</code> otherwise.
<code>true_method_args</code>	Optional list; values are passed to the function specified by <code>true_method</code> . Should be <code>NULL</code> when <code>true_mean</code> is not a formula.
<code>true_covariance</code>	Named list; specifies the covariance structures of the true error distribution in the format described in the documentation for the covariance argument of <code>distfreereg</code> . Required when <code>true_mean</code> is a function or <code>nls</code> object, or <code>true_method</code> is "nls"; should be <code>NULL</code> otherwise.
<code>true_X, true_data</code>	Optional numeric matrix or data frame, respectively; specifies the covariate values for the true model. <code>true_X</code> is used when <code>true_mean</code> is a function that has an <code>X</code> or <code>x</code> argument, and the <code>data</code> argument is used when <code>true_mean</code> is a formula or model object. At most one of these should be non- <code>NULL</code> . Both can be <code>NULL</code> when <code>true_mean</code> is a constant function (that is, it has no covariate argument).
<code>theta</code>	Numeric vector; used as the (true) parameter values for the model when <code>true_mean</code> is a function; should be <code>NULL</code> otherwise.
<code>n</code>	Optional integer; indicates how long each simulated data vector should be. Required only when no covariate values are specified for either the true or test mean; should be <code>NULL</code> otherwise. Silently converted to integer if numeric.
<code>reps</code>	Integer; specifies number of replications. Silently converted to integer if numeric.
<code>prog</code>	Integer or <code>Inf</code> ; if finite, a progress message is given when the current repetition is a multiple of <code>prog</code> . Default value is <code>reps/10</code> , unless <code>reps</code> is less than 10, in which case the default is 1. If <code>Inf</code> , no progress messages are given. Silently converted to integer if finite numeric.
<code>simulate_args</code>	Optional list; specifies additional named arguments to pass to <code>simulate</code> .
<code>err_dist_fun</code>	Character string; specifies the name of the function to be used to simulate errors when <code>true_mean</code> is a function or <code>nls</code> object, or <code>true_method</code> is "nls". See details.
<code>err_dist_args</code>	Optional list; specifies additional named arguments to pass to <code>err_dist_fun</code> .
<code>keep</code>	A vector of integers, or the character string "all". If not <code>NULL</code> , then the output of each replication's call to <code>distfreereg</code> is included in the output if its repetition number is included in <code>keep</code> . Using <code>keep = "all"</code> is equivalent to <code>keep = 1:reps</code> .

manual	Optional function; applied to the <code>distfreereg</code> object created in each iteration, whose output is saved in the list <code>manual</code> in the output.
update_args	Optional named list; specifies arguments to pass to <code>update.distfreereg</code> .
...	Additional arguments passed to <code>distfreereg</code> . See details.

## Details

This function allows the user to explore the asymptotic behavior of the distributions involved in the test conducted by `distfreereg`. If the sample size is large enough and the true covariance matrix of the errors is known or is estimated well enough, then the observed and simulated statistics have nearly the same distribution. How large the sample size must be depends on the details of the situation. This function can be used to determine how large the sample size must be to obtain approximately equal distributions, and to estimate the power of the test against a specific alternative.

The user specifies a particular true model which is used to generate outcome values. There are three cases:

- When `true_mean` is a function, this function determines the mean of the outcome values and `err_dist_fun` is used to generate errors. The error-generating function will usually include an element of `true_covariance` as an argument, and in that case must accept the appropriate class of object. For example, if the true covariance is a list of matrices corresponding to a block-diagonal covariance matrix, then `err_dist_fun` must accept such a list as an argument.
- When `true_mean` is an `nls` object, or when it is a formula and `true_method` is "nls", the function determined by the formula (in the model call or user-specified, respectively) is used to determine the mean function, and `err_dist_fun` generates the errors.
- When `true_mean` is a model object that is not an `nls` object, or a formula and method is not "nls", then `simulate` is used to generate outcome values.

If none of these cases apply to `true_mean`, then `compare()` cannot be used. (E.g., `true_mean` cannot be a `glm` object fitted using a "quasi" family, because `simulate` does not work for that family.)

The user also specifies arguments to pass to `distfreereg`, most notably a model to test comprising a mean function `test_mean` and a covariance structure specified by `covariance`. For each repetition, `compare` sends the simulated data, as `Y` or as part of `data`, to `distfreereg`.

The `true_covariance` argument specifies the covariance structure that is available to `err_dist_fun` for generating errors. The needs of `err_dist_fun` can vary (for example, the default function uses `SqrtSigma` to generate multivariate normal errors), so any one of the elements `Sigma`, `SqrtSigma`, `P`, and `Q` (defined in the documentation of `distfreereg`) can be specified. Any element needed by `err_dist_fun` is calculated automatically if not supplied.

The value of `err_dist_fun` must be a function whose output is a numeric matrix with `n` rows and `reps` columns. Each column is used as the vector of errors in one repetition. The error function's arguments can include the special values `n`, `reps`, `Sigma`, `SqrtSigma`, `P`, and `Q`. These arguments are automatically assigned their corresponding values from the values passed to `compare`. For example, the default value `rmvnorm` uses `SqrtSigma` to generate multivariate normal values with mean 0 and covariance `Sigma`.

The argument `keep` is useful for diagnosing problems, but caution should be used lest a very large object be created. It is often sufficient to save the `distfreereg` objects from only the first few replications.

For more specialized needs, the `manual` argument allows the calculation and saving of objects during each repetition. For example, using `manual = function(x) residuals(x)` will save the (raw) residuals from each repetition.

The first repetition creates a `distfreereg` object. During each subsequent repetition, this object is passed to `update.distfreereg` to create a new object. The `update_args` argument can be used to modify this call.

If necessary, `global_override` can be used to pass an override argument to `distfreereg` in each repetition. For example, using `global_override = list(theta_hat = theta)` forces the estimated parameter vector used in the test in each call to be the true parameter vector `theta`.

## Value

An object of class `compare` with the following components:

<code>call</code>	The matched call.
<code>Y</code>	The matrix whose columns contain the model outcome values used for the corresponding repetitions.
<code>theta</code>	Supplied vector of parameter values.
<code>true_mean</code>	Supplied object specifying the true mean function.
<code>true_covariance</code>	List containing element(s) that specify the true covariance structure.
<code>true_X</code>	Supplied matrix of true covariate values.
<code>true_data</code>	Supplied data frame of true covariate values.
<code>test_mean</code>	Supplied object specifying the mean function being tested.
<code>covariance</code>	List containing element(s) that specify the test covariance structure.
<code>X</code>	Supplied matrix of test covariate values.
<code>data</code>	Supplied data frame of test covariate values.
<code>empirical_stats</code>	The statistics observed in each repetition.
<code>theoretical_stats</code>	The simulated statistics from the first repetition. (They are the same for each repetition, because <code>compare</code> uses <code>update.distfreereg</code> .)
<code>p</code>	The p-values for the observed statistics.
<code>dfers</code>	A list containing the outputs of <code>distfreereg</code> for repetitions specified in <code>keep</code> . Included when <code>keep</code> is not <code>NULL</code> .
<code>manual</code>	A list containing the results of the function specified by the argument <code>manual</code> . Included when <code>manual</code> is not <code>NULL</code> .

## Warnings

The generation of new outcome values requires specifying an error distribution. The default behavior when `true_mean` is a function, an `nls` object, or a formula with method equal to "nls" is to use a multivariate normal error distribution, but different error-generating functions can be defined by the user. When `true_mean` is a model object that is not an `nls` object, or a formula and method

is not "nls", then the errors are generated using [simulate](#) and are therefore distributed according to that function's specifications. In short, the asymptotic behavior is determined for a specific (true) error distribution, even though the test itself is distribution-free.

Simulated outcome values for model using a subset argument can result in unexpected behavior when using a subsetting condition that involves the outcome variable. Such subsetting is highly discouraged, since the subsetting will occur after outcome values have been replaced and therefore different subsets might be selected for each repetition. Further, because of the way that a data set is retrieved from an nls object, subsetting with [nls](#) using "self-referential" conditions can cause unexpected results. For example, if a condition such as `x > median(x)` is used, then the median will be calculated on each subset in turn, which will shrink the sample size with each repetition and throw an error.

### Note

Some of the processing of the elements of `true_covariance` is analogous to the processing of covariance by [distfreereg](#). Any values of `solve_tol` and `symmetric` specified in [distfreereg](#)'s control argument are used by `compare` to similar effect in processing `true_covariance`.

Support for `glm` objects is limited to those created using a [family](#) that has a `simulate` element.

The presence of `call` in the value allows a `compare` object to be passed to [update](#).

### Author(s)

Jesse Miller

### See Also

[asymptotics](#), [distfreereg](#), [ks.test.compare](#), [plot.compare](#), [rejection](#)

### Examples

```
set.seed(20240201)
n <- 100
func <- function(X, theta) theta[1] + theta[2]*X[,1]
Sig <- rWishart(1, df = n, Sigma = diag(n))[, ,1]
theta <- c(2,5)
X <- matrix(rexp(n, rate = 1))
# In practice, 'reps' should be much larger
cdfr <- compare(true_mean = func, true_X = X, true_covariance = list(Sigma = Sig),
               test_mean = func, X = X, covariance = list(Sigma = Sig),
               reps = 10, prog = Inf, theta = theta, theta_init = rep(1, length(theta)))

cdfr$p
```

---

confint.distfreereg     *Calculate Confidence Intervals with a distfreereg Object*

---

### Description

This is a `confint` method for objects of class `distfreereg`. It calculates confidence intervals for the estimated parameters of a model in a `distfreereg` object.

### Usage

```
## S3 method for class 'distfreereg'  
confint(object, parm, level = 0.95, ...)
```

### Arguments

<code>object</code>	Object of class <code>distfreereg</code> .
<code>parm</code>	Numeric or character vector; specifies which parameters are to be given confidence intervals. If missing, all parameters are considered.
<code>level</code>	Numeric vector of length one; specifies the confidence level.
<code>...</code>	Additional parameters passed to other methods. Currently ignored.

### Details

When `object` contains a model object (either in the `test_mean` or `model` element), then this model is sent to `confint`. Otherwise, `object` is sent to `confint.default`.

### Value

The output from the appropriate `confint` method.

### Note

If `object` was created by calling `distfreereg` with no test mean function (that is, with `test_mean` equal to `NULL`), there is no estimated parameter vector, and therefore this function does not apply.

### Author(s)

Jesse Miller

### See Also

[distfreereg](#), [vcov.distfreereg](#), [confint](#)

## Description

Conduct distribution-free parametric regression testing using the process introduced in *Khmaladze (2021)*. A parametric model for the conditional mean (specified by `test_mean`) is checked against the data by fitting the model, transforming the resulting residuals, and then calculating a statistic on the empirical partial sum process of the transformed residuals. The statistic's null distribution can be simulated in a straight-forward way, thereby producing a p-value.

Using  $f$  to denote the mean function being tested, the specific test has the following null and alternative hypotheses:

$$H_0: \exists \theta \in \Theta \subseteq \mathbb{R}^p \mid E(Y|X) = f(X; \theta) \quad \text{against} \quad H_1: \forall \theta \in \Theta \subseteq \mathbb{R}^p \mid E(Y|X) \neq f(X; \theta).$$

This assumes a known or consistently estimated covariance matrix. See the [An Introduction to the `distfreereg` Package](#) vignette for an introduction.

## Usage

```
distfreereg(test_mean = NULL, covariance = NULL, Y = NULL, X = NULL,
            theta_init = NULL, data = NULL, method = NULL, method_args = NULL,
            stat = c("KS", "CvM"), B = 1e4, ordering = "simplex", group = TRUE,
            control = NULL, override = NULL, verbose = TRUE)
```

## Arguments

<code>test_mean</code>	A specification of the mean function to be tested. Accepted classes are function, formula, glm, lm, lmerMod (from the <b>lme4</b> package), and nls. Can be NULL if <code>override</code> contains <code>fitted_values</code> and <code>J</code> elements. See Details and Warnings.
<code>covariance</code>	Named list; specifies the covariance structure of the model's error distribution. Must be 'NULL' when 'test_mean' is a model object, in which case the covariance structure is extracted automatically from the model object. Valid element names are "Sigma", "SqrtSigma", "P", and "Q", corresponding to the covariance matrix, the square root of the covariance matrix, the precision matrix, and the square root of the precision matrix, respectively. Each element must be one of the following: <ul style="list-style-type: none"> <li>• a numeric matrix.</li> <li>• a list of numeric matrices.</li> <li>• a numeric vector whose length is the sample size.</li> <li>• a numeric vector of length 1.</li> </ul> See Details.
<code>Y</code>	Numeric vector of observations used when <code>test_mean</code> is a function or NULL; should be NULL otherwise. A non-vector value is converted to a vector.

X	Optional numeric matrix of covariates used when <code>test_mean</code> is a function or NULL; should be NULL otherwise. A non-matrix value is converted to a single-column matrix.
<code>theta_init</code>	Numeric vector; specifies the starting parameter values passed to the optimization function used to estimate the parameter vector. Used when <code>test_mean</code> is a function; should be NULL otherwise.
<code>data</code>	Data frame of covariate values used when <code>test_mean</code> is a formula; should be NULL otherwise.
<code>method</code>	Character vector; specifies the function to use for fitting the model when <code>test_mean</code> is a formula; should be NULL otherwise. Possible values are " <code>glm</code> ", " <code>lm</code> ", " <code>lmer</code> ", and " <code>nls</code> ".
<code>method_args</code>	Optional list of argument values to be passed to the function specified by the <code>method</code> argument. Should be NULL when <code>test_mean</code> is not a formula.
<code>stat</code>	Character vector; specifies the names of the functions used to calculate the desired statistics. By default, a Kolmogorov–Smirnov statistic and a Cramer–von Mises-like statistic are calculated:

$$\text{KS} = \max_i |a_i| \quad \text{and} \quad \text{CvM} = \frac{1}{n} \sum_{i=1}^n a_i^2$$

where  $a_i$  is term  $i$  in the empirical partial sum process and  $n$  is the sample size.

B	Numeric vector of length one; specifies the Monte Carlo sample size used when simulating statistics. Silently converted to integer.
<code>ordering</code>	A character string or a list; specifies how to order the residuals to form the empirical partial sum process. Valid character strings are: <ul style="list-style-type: none"> <li>"asis": leaves the order unchanged (that is, in the order in which the observations appear in the supplied data).</li> <li>"natural": orders residuals using column-wise ordering of the covariates.</li> <li>"optimal": orders the residuals by ordering the observations using optimal transport on the covariates. The solution is estimated using the Hungarian method as implemented by <code>solve_LSAP</code>. This option can be very slow for large sets of covariates.</li> <li>"simplex": (the default) orders residuals in order of increasing row sums of the covariates after each column has been scaled to the interval <math>[0, 1]</math>.</li> </ul> <p>If <code>ordering</code> is a list, then its elements specify names columns of <code>X</code> or <code>data</code> to use to determine the order.</p>
<code>group</code>	Logical; if TRUE, then residuals with the same rank (as determined by <code>ordering</code> ) are grouped when calculating the empirical partial sum process.
<code>control</code>	Optional named list of elements that control the details of the algorithm's computations. The following elements are accepted for all methods: <ul style="list-style-type: none"> <li><code>symmetric</code>: Optional named list or FALSE; if a named list, its elements are passed as arguments to <code>isSymmetric</code> when testing elements of covariance for symmetry. If FALSE, then this test is skipped.</li> <li><code>matsqrt_tol</code>: Numeric; specifies the threshold for considering an eigenvalue "too negative" when calculating the square root of a matrix. Must be non-positive. The default value is <code>-.Machine\$double.eps^0.25</code>.</li> </ul>

- `solve_tol`: Numeric; passed as `tol` argument to `solve`, used in particular to invert `Sigma`. The default value is `.Machine$double.eps`.
- `qr_tol`: Numeric; passed as `tol` argument to `qr`. The default value is `sqrt(.Machine$double.eps)`. This value might need to be decreased when the dimensions of the Jacobian are sufficiently large.
- `orth_tol`: Numeric; passed as tolerance argument to `all.equal` when testing whether or not  $r^T r$  and  $\mu^T \mu$  are the identity matrix. The default value is `sqrt(.Machine$double.eps)`.
- `trans_tol`: Numeric; passed as tolerance argument to `all.equal` to internal transformation function when determining whether the normalizing scalar is non-zero. The default value is `sqrt(.Machine$double.eps)`.
- `sym_tol`, `sym_tol1`: Numeric; passed as `tol` and `tol1` arguments, respectively, to `isSymmetric` for checking `Sigma` and `P` elements of covariance. The default values are `sqrt(.Machine$double.eps)` and `8*sqrt(.Machine$double.eps)`, respectively. See Warnings.
- `return_on_error`: Logical; determines whether or not partial output is returned when an error is encountered. Default value is `TRUE`.

The following named elements, all but the first of which control the process of calculating the generalized least squares estimation of the parameter vector, are accepted for the function method:

- `jacobian_args`: Optional list; specifies arguments to pass to `jacobian`.
- `optimization_fun`: Optional function; specifies the function used to estimate the parameters. If not specified, `optim` is used with `method = "BFGS"`.
- `fun_to_optimize_arg`: Optional character string, required when `optimization_fun` is specified; specifies the name of the argument of `optimization_fun` that is assigned the function to optimize. For example, `optim` uses `"fn"`.
- `theta_init_arg`: Optional character string, required when `optimization_fun` is specified; specifies the name of the argument of `optimization_fun` that is assigned the initial parameter values for optimization. For example, `optim` uses `"par"`.
- `theta_hat_name`: Optional character string, required when `optimization_fun` is specified; specifies the name of the element of the output of `optimization_fun` that contains the estimated parameters. For example, `optim` uses `"par"` (the same character string that specifies the argument containing the initial values). See Warnings.
- `optimization_args`: Optional list; specifies additional arguments to pass to `optimization_fun`.

override

Optional named list of arguments that override internally calculated values. Used primarily by `update.distfreereg`, but can be accessed directly. The following named elements are accepted:

- `J`: Numeric matrix. Overrides the calculation of the Jacobian.
- `fitted_values`: Numeric vector. Overrides the calculation of the fitted values.
- `res_order`: Integer vector specifying the ordering used to order the residuals for the computation of the empirical partial sum process. Overrides ordering. See Details.

- `theta_hat`: Numeric vector. Used as the estimated parameter vector when `test_mean` has class `function`, overriding internal computation that would have used `optimization_fun`.
- `r`: Orthogonal (numeric) matrix. Overrides the construction of the transformation anchor matrix.
- `theoretical_stats`: List. Overrides the creation of the list of simulated statistics.

`verbose` Logical; if TRUE, progress messages are printed. Default value is TRUE.

## Details

This function implements distribution-free parametric regression testing. The model is specified by a mean structure and a covariance structure.

The mean structure is specified by the argument `test_mean`. This can be an object of class `function`, `formula`, `glm`, `lm`, `lmerMod`, or `nls`. It can also be `NULL`.

If `test_mean` is a function, then it must have one or two arguments: either `theta` only, or `theta` and either `X` (uppercase) or `x` (lowercase). An uppercase `X` is interpreted in the function definition as a matrix, while a lowercase `x` is interpreted as a vector. (See examples and [this vignette](#).) The primary reason to use a lowercase `x` is to allow for a function definition using an R function that is not vectorized. In general, an uppercase `X` should be preferred for speed.

If `test_mean` is a formula, then it must be a formula that can be passed to `glm`, `lm`, `lmer`, or `nls`, and the `data` argument must be specified. The appropriate model is created and is then sent to `distfreereg()` for method dispatch.

The function method estimates parameter values, and then uses those to evaluate the Jacobian of the mean function and to calculate fitted values.

All of these methods create a Jacobian matrix and a vector of fitted values. These, along with the covariance structure, are sent the default method. The default method also allows the user to implement the algorithm even when the mean structure is not specified in R, as long as the Jacobian, fitted values, and covariance structure can be imported. (This is useful if a particularly complicated function is defined in another language and cannot easily be copied into R.)

The covariance structure for  $Y|X$  must be specified using the `covariance` argument for the function and default methods. For other methods, the covariance is estimated automatically.

Any element of `covariance` can be a numeric matrix, a list of numeric matrices, or a numeric vector. If it is a vector, its length must be either 1 or the sample size. This option is mathematically equivalent to setting a covariance list element to a diagonal matrix with the specified value(s) along the diagonal. Using vectors, when possible, is more efficient than using the corresponding matrix. If an element of `covariance` is a list of numeric matrices, then these matrices are interpreted as the blocks of a block diagonal matrix.

Internally, `distfreereg()` only needs  $Q$ , so some efficiency can be gained by supplying that directly when available. When  $Q$  is not specified, it is calculated using whichever element is specified. When more than one of the other elements are specified, no verification of compatibility of the elements is done.  $Q$  is calculated using the supplied element(s) by preferring one-step operations and operations on the square-root level. (For example, if both `P` and `SqrtSigma` elements are supplied, then  $Q$  is calculated using `SqrtSigma` only. If `Sigma` and `P` are supplied, then  $Q$  is calculated using `P` only.)

The `override` argument is used primarily by `update.distfreereg` to avoid unnecessary and potentially computationally expensive recomputation. This `update` method imports appropriate values

automatically from a previously created object of class `distfreereg`, and therefore validation is not always done. Use manually with caution.

The `res_order` element of `override` must be a vector of integers from 1 to  $n$  (the sample size) that determines the order of the residuals to use when forming the empirical partial sum process. Elements of the vector can be repeated, in which case the residuals corresponding to matching `res_order` values are grouped when `group` is `TRUE`.

## Value

An object of class `distfreereg` with the following components:

<code>call</code>	The matched call.
<code>data</code>	A list containing the data argument value, if present, and $Y$ and $X$ .
<code>test_mean</code>	The value supplied to the argument <code>test_mean</code> .
<code>model</code>	The model built when using the <code>formula</code> method; present if and only if using that method.
<code>covariance</code>	The list of covariance matrices, containing at least $Q$ .
<code>theta_hat</code>	The estimated parameter vector. When the model being tested has class <code>lmerMod</code> , this is the vector of fixed effect parameters as returned by <code>fixef</code> .
<code>optimization_output</code>	The output of <code>optimization_fun</code> or <code>nls</code> from calculating <code>theta_hat</code> .
<code>fitted_values</code>	The vector of fitted values, $f(X; \hat{\theta})$ . When testing a generalized linear model, these are the fitted values on the response scale.
<code>J</code>	The Jacobian matrix, $\frac{\partial f}{\partial \theta}(X; \hat{\theta})$ .
<code>mu</code>	The mu matrix.
<code>r</code>	The matrix of transformation anchor vectors.
<code>r_tilde</code>	The matrix of modified transformation anchor vectors.
<code>residuals</code>	A named list of three vectors containing raw, sphered, and transformed residuals.
<code>res_order</code>	A numeric vector indicating the ranking of the residuals used to form the empirical partial sum process, in a format to be used as the input of <code>order</code> .
<code>grouping_matrix</code>	The matrix used to group residuals; present if <code>group</code> is <code>TRUE</code> and some covariate combinations are repeated.
<code>epsp</code>	The empirical partial sum process formed by calculating the scaled partial sums of the transformed residuals ordered according to <code>res_order</code> .
<code>observed_stats</code>	A named list of the observed statistic(s) corresponding to the transformed residuals.
<code>theoretical_stats</code>	A named list, each element of which contains the values of a simulated statistic.
<code>p</code>	A named list with two elements: <code>value</code> , which contains the p-values for each observed statistic, and <code>mcse</code> , which contains the Monte Carlo standard errors for the p-values.

## Warnings

Methods for model objects (e.g., `lm` objects) are intended to be used with objects created using a `data` argument that contains all variables used by the model. This `data` argument is assumed to be defined in the same environment as the model object, and this is assumed to be the environment in which `distfreereg()` is called.

Consistency between `test_mean` and `theta_init` is verified only indirectly. Uninformative errors can occur when, for example, `theta_init` does not have the correct length. The most common error message that arises in this case is "f\_out cannot have NA values", which occurs when `theta_init` is too short. To be safe, always define `test_mean` to use every element of `theta`.

No verification of consistency is done when multiple elements of covariance are specified. For example, if `P` and `Sigma` are both specified, then the code will use only one of these, and will not verify that `P` is the inverse of `Sigma`.

When using the `control` argument element `optimization_fun` to specify an optimization function other than `optim`, the verification that `theta_hat_name` actually matches the name of an element of the optimization function's output is done only after the optimization has been done. If this optimization will likely take a long time, it is important to verify the value of `theta_hat_name` before running `distfreereg()`.

The default values of `sym_tol` and `sym_tol1` are intended to check for substantial asymmetry such as would be caused by the user inputting the wrong matrix. Therefore, they do not check for symmetry with high precision. In particular, these default values are orders of magnitude larger than the default values in `isSymmetric`.

The theory described by *Khmaladze (2021)* does not apply directly to generalized linear models nor linear mixed-effects models, but extensive simulations indicate that the method applies nonetheless to these models. Further investigation can be done using `asymptotics`. Non-null values for the `weights` argument in `glm` are not supported.

## Author(s)

Jesse Miller

## References

Khmaladze, Estate V. *Distribution-free testing in linear and parametric regression*, 2021-03, Annals of the Institute of Statistical Mathematics, Vol. 73, No. 6, p. 1063–1087. doi:10.1007/s10463021-007863

## See Also

`coef.distfreereg`, `confint.distfreereg`, `fitted.distfreereg`, `formula.distfreereg`, `plot.distfreereg`, `predict.distfreereg`, `print.distfreereg`, `residuals.distfreereg`, `update.distfreereg`, `vcov.distfreereg`

## Examples

```
set.seed(20240218)
n <- 1e2
func <- function(X, theta) X[,1]^theta[1] + theta[2]*X[,2]
Sig <- runif(n, min = 1, max = 3)
```

```

theta <- c(2,5)
X <- matrix(runif(2*n, min = 1, max = 5), nrow = n)
Y <- X[,1]^theta[1] + theta[2]*X[,2] + rnorm(n, sd = sqrt(Sig))
(dfr <- distfreereg(Y = Y, X = X, test_mean = func,
covariance = list(Sigma = Sig),
theta_init = c(1,1)))

# Same test with lowercase "x" for reference;
# use uppercase whenever possible.
func_lower <- function(x, theta) x[1]^theta[1] + theta[2]*x[2]
(dfr_lower <- distfreereg(Y = Y, X = X, test_mean = func_lower,
covariance = list(Sigma = Sig),
theta_init = c(1,1)))

```

---

fitted.distfreereg      *Extract Fitted Values from distfreereg Objects*

---

## Description

This is a [fitted](#) method for objects of class `distfreereg`.

## Usage

```

## S3 method for class 'distfreereg'
fitted(object, ...)

```

## Arguments

<code>object</code>	Object of class <code>distfreereg</code> .
<code>...</code>	Additional parameters passed to or from other methods. Currently ignored.

## Value

Numeric vector of fitted values.

## Author(s)

Jesse Miller

## See Also

[distfreereg](#)

---

`formula.distfreereg`     *Extract Formulas from distfreereg Objects*

---

### Description

This is a [formula](#) method for objects of class `distfreereg`. It extracts the formula from a model in a `distfreereg` object.

### Usage

```
## S3 method for class 'distfreereg'
formula(x, ...)
```

### Arguments

`x`                     Object of class `distfreereg`.  
`...`                    Additional parameters passed to or from other methods. Currently ignored.

### Value

Formula extracted from `x$test_mean`, or `NULL` if such a formula cannot be extracted.

### Author(s)

Jesse Miller

### See Also

[distfreereg](#)

---

`ks.test.compare`             *Formally Compare Empirical and Theoretical Statistics from a compare Object*

---

### Description

This is a [ks.test](#) method for objects of class `compare`. It performs a two-sample Kolmogorov–Smirnov test to compare the observed and simulated statistics in an object of class `compare`.

### Usage

```
## S3 method for class 'compare'
ks.test(x, ..., stat = NULL)
```

**Arguments**

x	Object of class compare.
...	Additional parameters passed to <code>ks.test</code> .
stat	Character string specifying the statistic on which to run the test.

**Details**

When `stat` is `NULL`, the default value is the first statistic appearing in the `observed_stats` element of object.

**Value**

A list of the form specified in [ks.test](#).

**Author(s)**

Jesse Miller

**See Also**

[compare](#), [distfreereg](#), [ks.test](#)

**Examples**

```
# In practice, set "reps" larger than 200.
set.seed(20240201)
n <- 100
func <- function(X, theta) theta[1] + theta[2]*X[,1]
Sig <- rWishart(1, df = n, Sigma = diag(n))[, , 1]
theta <- c(2,5)
X <- matrix(rexp(n, rate = 1))
cdfr <- compare(true_mean = func, true_X = X, true_covariance = list(Sigma = Sig),
               test_mean = func, X = X, covariance = list(Sigma = Sig), reps = 200,
               prog = Inf, theta = theta, theta_init = rep(1, length(theta)))

ks.test(cdfr)
ks.test(cdfr, stat = "CvM")
```

**Description**

This is a [plot](#) method for objects of class `compare`. It automates the creation of four summary and diagnostic plots for `compare` objects. See the [Plotting with the distfreereg Package](#) vignette for examples.

**Usage**

```
## S3 method for class 'compare'
plot(x, y, ..., which = "cdf", stat = NULL, hlines = NULL, curve_args = NULL,
      confband_args = FALSE, density_args = NULL, poly = NULL, legend = NULL,
      qqline = NULL)
```

**Arguments**

x	Object of class compare.
y	Optional object of class compare.
...	Additional parameters passed to a plotting function depending on the value of which: to <a href="#">plot</a> for "cdf" and "dens"; to <a href="#">qqplot</a> for "qq" and "qqp".
which	Character string. Acceptable values are "cdf", "dens", "qq", and "qqp": <ul style="list-style-type: none"> <li>"cdf" produces a plot of the estimated cumulative distribution functions of the two vectors of statistics being compared.</li> <li>"dens" produces a plot of the estimated density functions of the two vectors of statistics being compared.</li> <li>"qq" produces a quantile–quantile plot comparing the two vectors of statistics.</li> <li>"qqp" produces a quantile–quantile plot comparing the p-values with uniform quantiles. (This is not available when y is present.)</li> </ul>
stat	Character string, specifies the statistic to plot.
hlines	An optional list of arguments to pass to <a href="#">abline</a> , used to create the horizontal dashed lines when which is "cdf". Setting equal to FALSE prevents the call, and no lines are drawn.
curve_args	An optional list used to pass arguments to <a href="#">lines</a> (not <a href="#">curve</a> !), used to create the curves when which is "cdf" or "dens". It can have two special named arguments, obs and mcsim, whose values must be lists. Those lists contain arguments passed to the calls to <a href="#">lines</a> for plotting the curves for the empirical and theoretical statistics, respectively. Any other elements are passed to both calls.
confband_args	An optional list of values that control the calculation and plotting of confidence bands when which is "cdf" or "dens". Any of the following named elements are allowed. <ul style="list-style-type: none"> <li>w: Numeric; the sequence of points on which to evaluate the confidence band. By default, it is a sequence of m evenly-spaced numbers whose endpoints are determined by the quantiles of x calculated using the probabilities specified by q_probs, described below.</li> <li>m: Integer; the length of w, used only when w is NULL. The default value is 100.</li> <li>batch_len: Integer; the batch length for the algorithm. The default value is 50.</li> <li>N: Integer; the number of multivariate t samples to use in the simulation.</li> <li>conf.level: Numeric; the desired confidence level.</li> <li>q_probs: Numeric vector of length 2; the probabilities sent to <a href="#">quantile</a> to calculate the bounds used to define w, used only when w is NULL.</li> </ul>

- `curve_args`: An optional list of arguments passed to `lines` (again, not `curve!`), used to create the boundaries of the confidence band. It can have two special named arguments, `obs` and `mcsim`, which function in the same way as the corresponding elements of the `curve_args` argument described above.
- `polygon_args`: An optional list of arguments passed to `polygon`, used to shade the confidence region. Setting equal to `FALSE` prevents the call, and no shading is done.
- `shade_col`: This provides a shortcut to the `col` argument of `polygon` to change the color of the shaded region.

Setting equal to `FALSE` prevents calculation and plotting of the band.

<code>density_args</code>	An optional list of arguments passed to <code>density</code> when which is "dens", which calculates the points used to plot the density curves. The list can have two special named elements, <code>obs</code> and <code>mcsim</code> , which function in the same way as the corresponding elements of the <code>curve_args</code> argument described above.
<code>poly</code>	An optional list of arguments passed to <code>polygon</code> when which is "dens", which shades the area under the density curves. The list can have two special named elements, <code>obs</code> and <code>mcsim</code> , which modify the shadings for their respective curves, analogous to their behavior in the <code>curve_args</code> argument. When <code>poly</code> is equal to <code>FALSE</code> , no call is made, and therefore no shading is done.
<code>legend</code>	An optional list of arguments passed to <code>legend</code> when which is "cdf" or "dens". When equal to <code>FALSE</code> , no call is made, and therefore no legend is created.
<code>qqline</code>	An optional list of arguments passed to <code>abline</code> when which is "qq" or "qqp". By default, this plots the line $y = x$ . When equal to <code>FALSE</code> , no call is made, and therefore no line is plotted.

## Details

This function produces a plot of a type specified by `which`. The values plotted depend on whether or not `y` is present and the value of `which`. When `y` is present, the plots compare the empirical statistics in `x` and the empirical statistics in `y`. When `y` is missing, the plots compare the empirical and theoretical statistics in `x`. (The exception is when `which` is "qqp", which is only available when `y` is missing.)

When `which` is "cdf" or "dens", the plotting region and associated labels, tick marks, etc., are created by an initial call to `plot`. The curves themselves are drawn with `lines`. The arguments specified in `...` are passed to the initial call to `plot`.

## Value

The values used to create the curves (or points, in the case of a Q–Q plot) are returned invisibly. The details depend on the value of `which`:

- `cdf`: A list with two or four elements, all lists. The first two sub-lists contain the  $x$ - and  $y$ -values cdf curves. If confidence bands are plotted, then two additional elements are included with output from the confidence band calculations, including elements `w`, `cb_lower`, and `cb_upper`, which contain, respectively, the  $x$ -coordinates for both the upper and lower bounds of the band, the  $y$ -coordinates for the lower band, and the  $y$ -coordinates for the upper band.

- dens: A list with two or four elements, all lists. The first two sub-lists contain  $x$ - and  $y$ -values for the density curves. If confidence bands are plotted, then two additional sub-lists are supplied, with contents identical to what is described for "cdf".
- qq, qqp: The output of `qqplot`.

For "cdf" and "dens", the names of the elements of the returned list depend on whether or not a value for the argument `y` was supplied.

### Author(s)

Jesse Miller

### References

Flegal, James M. et al. **Simultaneous confidence bands for (Markov chain) Monte Carlo simulations**, forthcoming.

### See Also

[distfreereg](#), [compare](#)

---

plot.distfreereg

*Summary and Diagnostic Plots for distfreereg Objects*

---

### Description

This is a `plot` method for objects of class `distfreereg`. It automates the creation of summary and diagnostic plots for `distfreereg` objects. See the [Plotting with the distfreereg Package](#) vignette for examples.

### Usage

```
## S3 method for class 'distfreereg'
plot(x, which = "dens", stat = NULL, density_args = NULL,
     polygon_args = NULL, confband_args = NULL, abline_args = NULL,
     shade_col = rgb(1,0,0,0.5), text_args = NULL, ...)
```

### Arguments

- |                    |  |
|--------------------|--|
| <code>x</code>     | Object of class <code>distfreereg</code> .   |
| <code>which</code> | Character string. Acceptable values are "dens", "ecdf", "residuals", and "epsp": <ul style="list-style-type: none"> <li>• "dens" produces a plot of the estimated density curve of the specified statistic.</li> <li>• "ecdf" produces a plot of the empirical cumulative distribution function of the specified statistic.</li> </ul> |

	<ul style="list-style-type: none"> <li>• "residuals" produces a plot of the transformed residuals in the order specified by <code>x\$res_order</code>.</li> <li>• "epsp" produces a plot of the empirical partial sum process of the (ordered) transformed residuals.</li> </ul>
stat	Character vector of length one specifying the name of the statistic to plot when which is "dens" or "ecdf". By default, the first statistic in <code>x\$observed_stats</code> is used.
density_args	An optional list of arguments to pass to <code>density</code> when which is "dens".
polygon_args	An optional list of arguments to pass to <code>polygon</code> , used when which is "dens" to shade under the density curve to the right of the value of the observed statistic. Setting equal to FALSE prevents the call, and no shading is done.
confband_args	<p>An optional list of values that control the calculation and plotting of confidence bands. Any of the following named elements are allowed.</p> <ul style="list-style-type: none"> <li>• w: Numeric; the sequence of points on which to evaluate the confidence band. By default, it is a sequence of <code>m</code> evenly-spaced numbers whose endpoints are determined by the quantiles of <code>x</code> calculated using the probabilities specified by <code>q_probs</code>, described below.</li> <li>• m: Integer; the length of <code>w</code>, used only when <code>w</code> is NULL. The default value is 100.</li> <li>• batch_len: Integer; the batch length for the algorithm. The default value is 50.</li> <li>• N: Integer; the number of multivariate <math>t</math> samples to use in the simulation.</li> <li>• conf.level: Numeric; the desired confidence level.</li> <li>• q_probs: Numeric vector of length 2; the probabilities sent to <code>quantile</code> to calculate the bounds used to define <code>w</code>, used only when <code>w</code> is NULL.</li> <li>• curve_args: An optional list of arguments passed to <code>lines</code> (not <code>curve!</code>), used to create the boundaries of the confidence band.</li> <li>• polygon_args: An optional list of arguments passed to <code>polygon</code>, used to shade the confidence region. Setting equal to FALSE prevents the call, and no shading is done.</li> <li>• shade_col: This provides a shortcut to the <code>col</code> argument of <code>polygon</code> to change the color of the shaded region.</li> </ul> <p>Setting equal to FALSE prevents calculation and plotting of the confidence band.</p>
abline_args	An optional list of arguments to pass to <code>abline</code> , used to draw a vertical line at the value of the observed statistic. Setting equal to FALSE prevents the call, and no line is drawn.
shade_col	Character string or other value specifying the color to use to shade the upper tail of the distribution when which is "dens". Default value is red with 50% transparency. This is a convenience argument, and the same functionality is available by defining a <code>col</code> element in the <code>polygon_args</code> argument.
text_args	An optional list of arguments to pass to <code>text</code> , used to label the vertical line with the p-value of the observed statistic. Setting equal to FALSE prevents the call, and no text is printed.
...	Additional arguments to pass to <code>plot</code> .

## Details

This function produces one of three specified plots, depending on the value of `which`.

When `which` is "dens" or "ecdf", a plot of the estimated density or empirical cumulative distribution function, respectively, of the simulated statistics is produced, including a vertical line at the value of the observed test statistic with the p-value displayed.

The default placement of the p-value text is on the left side of the line indicating the statistic value. Specifically, the default values of `x` and `y` passed to `text` are the statistic value itself and the midpoint between zero and the maximum value of the density curve. The default value passed to `adj` is `c(1, 0.5)`, meaning that the text is aligned to the left of the value  $(x, y)$  and centered vertically on it. (The default value for the text itself, which can be modified via the `label` argument of `text`, includes a space on the left and the right for padding so the text does not overlap the vertical line itself.) To align the text so it appears on the right side (for example, to avoid overlapping the density curve), use `text_args = list(adj = c(0, 0.5))`. See documentation for `text` for details on this and other arguments.

When `which` is "residuals", a time-series-like plot is produced showing transformed residuals in the order given by `x$res_order`. In the case that the null hypothesis is rejected, this plot can help determine where (in terms of the linearly ordered covariates) a discrepancy between the model and the data occurs.

When `which` is "epsp", a plot of the empirical partial sum process is produced; that is, the  $y$ -values are

$$y_j = \frac{1}{\sqrt{n}} \sum_{i=1}^j \hat{\epsilon}_i$$

where  $\hat{\epsilon}_i$  is the  $i$ th transformed residual in the order given by `x$res_order`. Similar to the case when `which` is "residuals", this plot can help determine where (in terms of the linearly ordered covariates) a discrepancy between the model and the data occurs.

## Value

When `which` is "dens" or "ecdf", the values used to create the plot are returned invisibly in a list with two named elements, `x` and `y`. If the confidence band is plotted, then it is included as an element named `confband`.

For other values of `which`, nothing is returned.

## Author(s)

Jesse Miller

## References

Flegal, James M. et al. **Simultaneous confidence bands for (Markov chain) Monte Carlo simulations**, forthcoming.

## See Also

[distfreereg](#)

---

predict.distfreereg    *Generate Predicted Values from distfreereg Objects*

---

## Description

This is a [predict](#) method for objects of class distfreereg.

## Usage

```
## S3 method for class 'distfreereg'  
predict(object, ...)
```

## Arguments

object	Object of class distfreereg.
...	Additional arguments passed to other <a href="#">predict</a> methods. In particular, can include a newdata value.

## Details

When object\$test\_mean is a model object ("lm", "glm", or "nls"), object\$test\_mean is sent to [predict](#) for method dispatch. When object\$test\_mean is of class "formula", object\$model is sent to [predict](#).

## Value

Numeric vector of predicted values.

## Author(s)

Jesse Miller

## See Also

[distfreereg](#)

---

`print.compare`      *Printing compare Objects*

---

**Description**

This is a [print](#) method for objects of class `compare`.

**Usage**

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

**Arguments**

`x`                    Object of class `compare`.  
`...`                 Additional parameters, currently ignored.

**Details**

This function prints a useful summary of the `compare` object `x`.

**Value**

No return value (NULL).

**Author(s)**

Jesse Miller

**See Also**

[compare](#)

---

`print.distfreereg`      *Printing distfreereg Objects*

---

**Description**

This is a [print](#) method for objects of class `distfreereg`.

**Usage**

```
## S3 method for class 'distfreereg'  
print(x, ..., digits = 3, col_sep = 2, show_params = TRUE)
```

**Arguments**

<code>x</code>	Object of class <code>distfreereg</code> .
<code>...</code>	Additional parameters, currently ignored.
<code>digits</code>	Integer; passed to <code>signif</code> to determine the number of significant digits to display.
<code>col_sep</code>	Integer; specifies the padding (in units of spaces) between columns in the printed table of statistics.
<code>show_params</code>	Logical; determines whether or not the parameter estimates, if present in <code>x</code> , are displayed.

**Details**

This function prints a useful summary of the `distfreereg` object `x`.

**Value**

No return value (NULL).

**Author(s)**

Jesse Miller

**See Also**

[distfreereg](#)

---

<code>rejection</code>	<i>Compute Rejection Rates of a Distribution-Free Test from a compare Object</i>
------------------------	--

---

**Description**

Compute the rejection rates of the tests simulated in a `compare` object. Specifically, this function estimates the rejection rates of the tests conducted with specified statistics of the hypothesis that the mean function is `test_mean` when the true mean function is `true_mean`.

**Usage**

```
rejection(object, alpha = 0.05, stat = names(object[["empirical_stats"]]), ...)
```

**Arguments**

object	Object of class <code>compare</code> .
alpha	Numeric vector; specifies the $\alpha$ -levels to use. Passed as <code>probs</code> argument to <a href="#">quantile</a> .
stat	Character vector; specifies the names of the statistics to use. The default value computes the rejection rate associated with every statistic in <code>object</code> .
...	Additional arguments to pass to <a href="#">quantile</a> to estimate the $1 - \alpha$ quantiles of the distribution of simulated statistics.

**Value**

Data frame containing estimated rejection rates and associated Monte Carlo standard errors, with one row for each combination of `stat` and `alpha` elements.

**Warning**

The reported Monte Carlo standard error does not account for the uncertainty of the estimation of the  $1 - \alpha$  quantiles of the distribution of simulated statistics. The number of Monte Carlo simulations should be large enough to make this estimate sufficiently accurate that it can be considered known for practical purposes. The standard errors of estimated quantiles can be calculated using the `mcmcse` package.

**Author(s)**

Jesse Miller

**See Also**

[distfreereg](#), [compare](#)

**Examples**

```
# In practice, set "reps" much larger than 20.
set.seed(20240201)
n <- 100
func <- function(X, theta) theta[1] + theta[2]*X[,1]
Sig <- rWishart(1, df = n, Sigma = diag(n))[, , 1]
theta <- c(2,5)
X <- matrix(rexp(n, rate = 1))
cdfr <- compare(true_mean = func, true_X = X, true_covariance = list(Sigma = Sig),
               test_mean = func, X = X, covariance = list(Sigma = Sig), reps = 20,
               prog = Inf, theta = theta, theta_init = rep(1, length(theta)))

rejection(cdfr)
rejection(cdfr, stat = "CvM")
rejection(cdfr, alpha = c(0.1, 0.2))
```

---

residuals.distfreereg *Extract Residuals from distfreereg Objects*

---

### Description

This is a [residuals](#) method for objects of class `distfreereg`. It can extract any of the three available types of residuals.

### Usage

```
## S3 method for class 'distfreereg'  
residuals(object, ..., type = "raw")
```

### Arguments

<code>object</code>	Object of class <code>distfreereg</code> .
<code>...</code>	Additional parameters passed to or from other methods. Currently ignored.
<code>type</code>	Character string specifying the type of residuals to return. Must be one of "raw", "sphered", and "transformed".

### Value

Numeric vector of residuals.

### Author(s)

Jesse Miller

### See Also

[distfreereg](#)

---

update.distfreereg *Update distfreereg Objects*

---

### Description

This is a `distfreereg` method for [update](#). The method takes advantage of the `override` argument of [distfreereg](#) to prevent unnecessary recalculation of potentially computationally expensive objects.

### Usage

```
## S3 method for class 'distfreereg'  
update(object, ..., smart = TRUE)
```

**Arguments**

object	Object of class <code>distfreereg</code> .
...	Additional named parameters to pass to <code>distfreereg</code> .
smart	Logical. If TRUE, then saved values from object are passed to <code>distfreereg</code> using the <code>override</code> argument, when they need not themselves be updated. See details.

**Details**

This function updates an object of class `distfreereg`. By default, it does so "intelligently" in the sense that it does not unnecessarily recompute elements that are already saved in object. For example, if a new value for covariance is not included in ..., then the value of covariance saved in object is automatically passed to the new call, preventing recalculating  $Q$ . If a new value of covariance is specified, then all objects dependent on that (e.g.,  $\hat{\theta}$ ) are recomputed by default.

In particular, the simulated samples depend on the data and function only through the number of observations, the covariates (if any), and the dimension of the parameter space of the function. If none of these change, then the updated object reuses the simulated samples from the supplied object.

This function uses the `'override'` argument of `distfreereg`, and therefore it must be handled carefully when it itself is being updated. To create the value of `'override'` for the updated call, the following three named lists are combined. When a name appears in more than one of these lists, priority is given in the order in which the lists are shown:

- The value of `'override'` supplied to `update`.
- Values of `'fitted_values'` and `'J'` supplied to the `'override'` argument in the `'call'` element of `'object'`, if the value of `'test_mean'` in the updated call is `'NULL'`.
- The list of "intelligently chosen" override values determined by `update.distfreereg`.

**Value**

An updated object of class `distfreereg`.

**Note**

The usual behavior of `update` is to create an updated call and then evaluate that call. This is what `update.distfreereg` does, as well, but some of the updated elements are drawn from object itself for use as override values. In general, an object created by `update.distfreereg` is not **identical** to the object created by `distfreereg` using corresponding arguments, because the call values will differ.

**Author(s)**

Jesse Miller

**See Also**

[distfreereg](#)

**Examples**

```

set.seed(20240218)
n <- 1e2
func <- function(X, theta) X[,1]^theta[1] + theta[2]*X[,2]
Sig <- runif(n, min = 1, max = 3)
theta <- c(2,5)
X <- matrix(runif(2*n, min = 1, max = 5), nrow = n)
Y <- X[,1]^theta[1] + theta[2]*X[,2] + rnorm(n, sd = sqrt(Sig))
dfr_1 <- distfreereg(Y = Y, X = X, test_mean = func,
                    covariance = list(Sigma = Sig),
                    theta_init = c(1,1))

func_updated <- function(X, theta) X[,1]^theta[1] + theta[2]*X[,2]^2
dfr_2 <- update(dfr_1, test_mean = func_updated)

```

vcov.distfreereg

*Estimate Parameter Covariance Matrices from distfreereg Objects***Description**

This is a [vcov](#) method for objects of class `distfreereg`. It estimates the covariance matrix of the estimated parameters in a model from a `distfreereg` object.

**Usage**

```

## S3 method for class 'distfreereg'
vcov(object, ..., jacobian_args, hessian_args)

```

**Arguments**

<code>object</code>	Object of class <code>distfreereg</code> .
<code>...</code>	Additional parameters passed to other methods when <code>test_mean</code> element of <code>object</code> is not of class function.
<code>jacobian_args, hessian_args</code>	Lists of additional arguments to pass to <a href="#">jacobian</a> and <a href="#">hessian</a> .

**Details**

When the `test_mean` element of `object` is of class function, the covariance matrix is estimated using the method described in section 5.3 of *Van der Vaart (2007)*. Otherwise, `test_mean` is of a class that has its own method for [vcov](#), which is used to calculate the output.

**Value**

Named numeric matrix equal to the estimated covariance matrix of the parameter estimates from `object`.

**Warning**

This calculation can be computationally intensive when the sample size is large and `object$test_mean` is a function.

**Note**

If object was created by calling `distfreereg` with no test mean function (that is, with `test_mean` equal to `NULL`), there is no estimated parameter vector, and therefore this function does not apply.

**Author(s)**

Jesse Miller

**References**

Vaart, A. W. **Asymptotic statistics**, 2007, *Cambridge series on statistical and probabilistic mathematics*, Cambridge University Press.

**See Also**

[distfreereg](#), [confint.distfreereg](#)

# Index

abline, [20](#), [21](#), [23](#)  
all.equal, [13](#)  
asymptotics, [4](#), [5](#), [9](#), [16](#)  
  
coef, [5](#)  
coef.distfreereg, [5](#), [16](#)  
compare, [4](#), [5](#), [5](#), [19](#), [22](#), [26](#), [28](#)  
confint, [10](#)  
confint.default, [10](#)  
confint.distfreereg, [5](#), [10](#), [16](#), [32](#)  
curve, [20](#), [21](#), [23](#)  
  
density, [21](#), [23](#)  
distfreereg, [4–10](#), [11](#), [17–19](#), [22](#), [24](#), [25](#),  
[27–30](#), [32](#)  
distfreereg-package, [2](#)  
  
family, [7](#), [9](#)  
fitted, [17](#)  
fitted.distfreereg, [16](#), [17](#)  
fixef, [15](#)  
formula, [18](#)  
formula.distfreereg, [16](#), [18](#)  
  
glm, [12](#), [14](#), [16](#)  
  
hessian, [31](#)  
  
identical, [30](#)  
isSymmetric, [12](#), [13](#), [16](#)  
  
jacobian, [13](#), [31](#)  
  
ks.test, [18](#), [19](#)  
ks.test.compare, [9](#), [18](#)  
  
legend, [21](#)  
lines, [20](#), [21](#), [23](#)  
lm, [6](#), [12](#), [14](#)  
lmer, [12](#), [14](#)  
  
nls, [9](#), [12](#), [14](#), [15](#)  
  
optim, [13](#), [16](#)  
order, [15](#)  
  
plot, [19–23](#)  
plot.compare, [9](#), [19](#)  
plot.distfreereg, [16](#), [22](#)  
polygon, [21](#), [23](#)  
predict, [25](#)  
predict.distfreereg, [16](#), [25](#)  
print, [26](#)  
print.compare, [26](#)  
print.distfreereg, [16](#), [26](#)  
  
qqplot, [20](#), [22](#)  
qr, [13](#)  
quantile, [20](#), [23](#), [28](#)  
  
rejection, [9](#), [27](#)  
residuals, [29](#)  
residuals.distfreereg, [16](#), [29](#)  
  
signif, [27](#)  
simulate, [6](#), [7](#), [9](#)  
solve, [13](#)  
solve\_LSAP, [12](#)  
  
text, [23](#), [24](#)  
  
update, [9](#), [14](#), [29](#), [30](#)  
update.distfreereg, [7](#), [8](#), [13](#), [14](#), [16](#), [29](#)  
  
vcov, [31](#)  
vcov.distfreereg, [5](#), [10](#), [16](#), [31](#)