Package 'Rgof'

June 16, 2025

Title 1d Goodness of Fit Tests

Version 3.3.0

Description Routines that allow the user to run a large number of goodness-of-fit tests. It allows for data to be continuous or discrete. It includes routines to estimate the power of the tests and display them as a power graph. The routine run.studies allows a user to quickly study the power of a new method and how it compares to some of the standard ones.

License GPL (>= 2)

Encoding UTF-8

RoxygenNote 7.3.2

LinkingTo Rcpp

Imports Rcpp, parallel, ggplot2, stats, graphics, microbenchmark, nortest

Suggests rmarkdown, knitr

VignetteBuilder knitr

Depends R (>= 3.5)

LazyData true

NeedsCompilation yes

Author Wolfgang Rolke [aut, cre] (ORCID: <https://orcid.org/0000-0002-3514-726X>)

Maintainer Wolfgang Rolke <wolfgang.rolke@upr.edu>

Repository CRAN

Date/Publication 2025-06-16 18:30:02 UTC

Contents

`case.studies` *This function creates the functions needed to run the various case studies.*

Description

This function creates the functions needed to run the various case studies.

Usage

```
case.studies(which, nsample = 500)
```

Arguments

`which`	name of the case study.
`nsample`	=500, sample size.

Value

a list of functions

check.functions *This function checks whether the inputs have the correct format*

Description

This function checks whether the inputs have the correct format

Usage

```
check.functions(pnull, rnull, phat = function(x) -99, vals, x)
```

Arguments

pnull	cdf under the null hypothesis
rnull	routine to generate data under the null hypothesis
phat	=function(x) -99, function to estimate parameters from the data, or -99
vals	vector of discrete values
x	data

chi_power_cont *This function finds the power of various chi-square tests for continuous data*

Description

This function finds the power of various chi-square tests for continuous data

Usage

```
chi_power_cont(
  pnull,
  ralt,
  param_alt,
  qnull = NA,
  phat = function(x) -99,
  w = function(x) -99,
  alpha = 0.05,
  Range = c(-99999, 99999),
  B = 1000,
  nbins = c(50, 10),
  rate = 0,
  minexpcount = 5,
  ChiUsePhat = TRUE
)
```

Arguments

pnull	function to find cdf under null hypothesis
ralt	function to generate data under alternative hypothesis
param_alt	vector of parameter values for distribution under alternative hypothesis
qnull	=NA function to find quantiles under null hypothesis, if available
phat	=function(x) -99, function to estimate parameters
w	=function(x) -99, optional weight function
alpha	=0.05, the level of the hypothesis test
Range	=c(-99999, 99999) limits of possible observations, if any
B	=1000 number of simulation runs to find power
nbins	=c(50,10), number of bins for chi square tests
rate	=0 rate of Poisson if sample size is random, 0 if sample size is fixed
minexpcount	=5 minimal expected bin count required
ChiUsePhat	=TRUE, if TRUE param is estimated parameters and no minimization is used

Value

A numeric matrix of power values.

---

chi_power_disc *This function finds the power of various chi-square tests for continuous data*

Description

This function finds the power of various chi-square tests for continuous data

Usage

```
chi_power_disc(
  pnull,
  ralt,
  param_alt,
  phat = function(x) -99,
  alpha = 0.05,
  B = 1000,
  nbins = c(50, 10),
  rate = 0,
  minexpcount = 5,
  ChiUsePhat = TRUE
)
```

Arguments

pnull	function to find cdf under null hypothesis
ralt	function to generate data under alternative hypothesis
param_alt	vector of parameter values for distribution under alternative hypothesis
phat	=function(x) -99, routine to estimate parameters
alpha	=0.05, the level of the hypothesis test
B	=1000 number of simulation runs to find power
nbins	=c(50,10), number of bins for chi square tests
rate	=0 rate of Poisson if sample size is random, 0 if sample size is fixed
minexpcount	=5 minimal expected bin count required
ChiUsePhat	= TRUE, should chi square use minimum chi square method?

Value

A numeric matrix of power values.

---

chi_test_cont *This function performs a number of chi-square gof tests for continuous data*

Description

This function performs a number of chi-square gof tests for continuous data

Usage

```
chi_test_cont(
  x,
  pnull,
  w = function(x) -99,
  phat = function(x) -99,
  qnull = NA,
  nbins = c(50, 10),
  rate = 0,
  Range = c(-99999, 99999),
  minexpcount = 5,
  ChiUsePhat = TRUE,
  allbins
)
```

Arguments

x	data set
pnull	cdf under the null hypothesis
w	function to find weights of observations, returns -99 if data is unweighted
phat	=function(x) -99, estimated parameters, or starting values of multi-D minimum chi square minimization, or -99 if no estimation is done
qnull	=NA quantile function, if available
nbins	=c(50, 10) number of bins for chi-square tests
rate	=0, rate of Poisson if sample size is random
Range	=c(-99999, 99999) limits of possible observations, if any
minexpcount	=5 minimal expected bin count required
ChiUsePhat	=TRUE, if TRUE param is estimated parameters and no minimization is used
allbins	set of bins to use

Value

A numeric matrix of test statistics, degrees of freedom and p.values

---

chi_test_disc *This function performs a number of chi-square gof tests for continuous data*

Description

This function performs a number of chi-square gof tests for continuous data

Usage

```
chi_test_disc(
  x,
  pnull,
  phat = function(x) -99,
  nbins = c(50, 10),
  rate = 0,
  minexpcount = 5,
  ChiUsePhat = TRUE,
  allbins
)
```

Arguments

x	data set
pnull	cdf under the null hypothesis
phat	=function(x) -99, function to estimate parameters, or starting values of multi-D minimum chi square minimization, or -99 if no parameters are estimated
nbins	=c(50, 10) number of bins for chi-square tests
rate	=0, rate of Poisson if sample size is random
minexpcount	=5 minimal expected bin count required
ChiUsePhat	= TRUE, if TRUE param is estimated parameter, otherwise minimum chi square method is used.
allbins	set of bins to use

Value

A numeric matrix of test statistics, degrees of freedom and p.values

---

gof_power *Power estimation of goodness-of-fit tests.*

Description

Find the power of various goodness-of-fit tests.

Usage

```
gof_power(
  pnull,
  vals = NA,
  rnull,
  ralt,
  param_alt,
  w = function(x) -99,
  phat = function(x) -99,
  TS,
  TSextra,
  With.p.value = FALSE,
  alpha = 0.05,
  Range = c(-Inf, Inf),
  B = 1000,
  nbins = c(50, 10),
  rate = 0,
  maxProcessor,
  minexpcount = 5,
  ChiUsePhat = TRUE
)
```

Arguments

pnull	function to find cdf under null hypothesis
vals	=NA values of discrete random variable, or NA
rnull	function to generate data under null hypothesis
ralt	function to generate data under alternative hypothesis
param_alt	vector of parameter values for distribution under alternative hypothesis
w	(Optional) function to calculate weights, returns -99 if no weights
phat	=function(x) -99 function to estimate parameters from the data, or -99
TS	user supplied function to find test statistics
TSextra	list provided to TS (optional)
With.p.value	=FALSE does user supplied routine return p values?
alpha	=0.05, the level of the hypothesis test
Range	=c(-Inf, Inf) limits of possible observations, if any
B	=1000 number of simulation runs
nbins	=c(50,10), number of bins for chi square tests.
rate	=0 rate of Poisson if sample size is random, 0 if sample size is fixed
maxProcessor	maximum of number of processors to use, 1 if no parallel processing is needed or number of cores-1 if missing
minexpcount	=5 minimal expected bin count required
ChiUsePhat	= TRUE, if TRUE param is estimated parameter, otherwise minimum chi square method is used.

Details

For details on the usage of this routine consult the vignette with vignette("Rgof","Rgof")

Value

A numeric matrix of power values.

Examples

```
# Power of tests when null hypothesis specifies the standard normal distribution but
# true data comes from a normal distribution with mean different from 0.
pnull = function(x) pnorm(x)
rnull = function()  rnorm(50)
ralt = function(mu)  rnorm(50, mu)
TSextra = list(qnull=function(x) qnorm(x))
gof_power(pnull, NA, rnull, ralt, c(0.25, 0.5), TSextra=TSextra, B=200)
# Power of tests when null hypothesis specifies normal distribution and
# mean and standard deviation are estimated from the data.
# true data comes from a normal distribution with mean different from 0.
pnull = function(x, p=c(0, 1)) pnorm(x, p[1], ifelse(p[2]>0.001, p[2], 0.001))
rnull = function(p=c(0, 1))  rnorm(50, p[1], ifelse(p[2]>0.001, p[2], 0.001))
```

```
ralt = function(mu)  rnorm(50, mu)
phat = function(x) c(mean(x), sd(x))
TSextra = list(qnull = function(x, p=c(0, 1)) qnorm(x, p[1],
               ifelse(p[2]>0.001, p[2], 0.001)))
pwr=gof_power(pnull, NA, rnull, ralt, c(0, 1), phat=phat, TSextra=TSextra, B=200)
pwr
#' Compare power of a new test based on variants of the Cramer-vonMises
#' criterion to the methods included in the package:
newTS = function(x, pnull, param) {
   Fx=sort(pnull(x, param))
   n=length(x)
   out = c(sum(abs( (2*1:n-1)/2/n-Fx )), sum(sqrt(abs( (2*1:n-1)/2/n-Fx ))))
   names(out) = c("CvM alt 1","CvM alt 2")
   out
}
#' Compare power to Lilliefors KS test, which finds its own p value:
LLtest=function(x, pnull, param) {
  out=nortest::lillie.test(x)$p.value
  names(out)="KS - Lilliefors"
  out
}
cbind(gof_power(pnull, NA, rnull, ralt, c(0, 1), TS=LLtest, phat=phat,
       With.p.value=TRUE, TSextra=TSextra, B=200), pwr)
# Power of tests when null hypothesis specifies Poisson rv with rate 100 and
# true rate is 100.5
vals = 0:250
pnull = function() ppois(0:250, 100)
rnull =function () table(c(0:250, rpois(1000, 100)))-1
ralt =function (p) table(c(0:250, rpois(1000, p)))-1
gof_power(pnull, vals, rnull, ralt, param_alt=100.5,  B=200)
# Power of tests when null hypothesis specifies a Binomial n=10 distribution
# with the success probability estimated
vals = 0:10
pnull=function(p) pbinom(0:10, 10, ifelse(0<p&p<1, p, 0.001))
rnull=function(p) table(c(0:10, rbinom(1000, 10, ifelse(0<p&p<1, p, 0.001))))-1
ralt=function(p) table(c(0:10, rbinom(1000, 10, p)))-1
phat=function(x) mean(rep(0:10,x))/10
gof_power(pnull, vals, rnull, ralt, c(0.5, 0.6), phat=phat, B=200)
```

`gof_test` *Tests for the univariate goodness-of-fit problem*

Description

This function runs a number of goodness-of-fit tests using Rcpp and parallel computing.

Usage

```
gof_test(
```

```
  x,
  vals = NA,
  pnull,
  rnull,
  w = function(x) -99,
  phat = function(x) -99,
  TS,
  TSextra = NA,
  nbins = c(50, 10),
  rate = 0,
  Range = c(-Inf, Inf),
  B = 5000,
  minexpcount = 5,
  ChiUsePhat = TRUE,
  maxProcessor,
  doMethods = "all"
)
```

Arguments

x	data set
vals	=NA, values of discrete RV, or NA if data is continuous
pnull	cdf under the null hypothesis
rnull	routine to generate data under the null hypothesis
w	(Optional) function to calculate weights, returns -99 if no weights
phat	=function(x) -99, function to estimate parameters from the data, or -99 if no parameters are estimated
TS	user supplied function to find test statistics, if any
TSextra	=NA, list passed to TS, if desired, or NA
nbins	=c(100, 10) number of bins for chi-square tests
rate	=0 rate of Poisson if sample size is random, 0 if sample size is fixed
Range	=c(-Inf, Inf) limits of possible observations, if any, for chi-square tests
B	=5000 number of simulation runs
minexpcount	=5 minimal expected bin count required
ChiUsePhat	= TRUE, if TRUE param is estimated parameter, otherwise minimum chi square method is used.
maxProcessor	=1, number of processors to use in parallel processing.
doMethods	="all", a vector of codes for the methods to include or all of them.

Details

For details on the usage of this routine consult the vignette with vignette("Rgof","Rgof")

Value

A list with vectors of test statistics and p.values

Examples

```
# Tests to see whether data comes from a standard normal distribution.
pnull = function(x) pnorm(x)
rnull = function()  rnorm(100)
x = rnorm(100)
gof_test(x, NA, pnull, rnull, B=500)
# Tests to see whether data comes from a normal distribution with standard deviation 1
# and the mean estimated.
pnull=function(x, m) pnorm(x, m)
rnull=function(m) rnorm(100, m)
TSextra = list(qnull=function(x, m=0) qnorm(x, m),
          pnull=function(x, m=0) pnorm(x, m), phat=function(x) mean(x))
phat=function(x) mean(x)
x = rnorm(100, 1, 2)
gof_test(x, NA, pnull, rnull, phat=phat, TSextra=TSextra, B=500)
# Tests to see whether data comes from a binomial (10, 0.5) distribution.
vals=0:10
pnull = function() pbinom(0:10, 10, 0.5)
rnull = function() table(c(0:10, rbinom(1000, 10, 0.5)))-1
x = rnull()
gof_test(x, vals, pnull, rnull, doMethods="all", B=500)
# Tests to see whether data comes from a binomial distribution with
# the success probability estimated from the data.
pnull = function(p=0.5) pbinom(0:10, 10, ifelse(p>0&&p<1, p, 0.001))
rnull = function(p=0.5) table(c(0:10, rbinom(1000, 10,
                  ifelse(p>0&&p<1, p, 0.001))))-1
phat=function(x) mean(rep(0:10,x))/10
gof_test(x, vals, pnull, rnull, phat=phat, B=500)
```

`gof_test_adjusted_pvalue` *Adjusted p values for simultaneous testing in the goodness-of-fit problem.*

Description

This function performs a number of goodness-of-fit tests and finds the adjusted p value for the combined test.

Usage

```
gof_test_adjusted_pvalue(
  x,
  vals = NA,
  pnull,
  rnull,
  w = function(x) -99,
  phat = function(x) -99,
```

```
  TS,
  TSextra = NA,
  nbins = c(50, 10),
  rate = 0,
  Range = c(-Inf, Inf),
  B = c(5000, 1000),
  minexpcount = 5,
  ChiUsePhat = TRUE,
  maxProcessor,
  doMethods
)
```

Arguments

x	data set
vals	=NA, values of discrete RV, or NA if data is continuous
pnull	cdf under the null hypothesis
rnull	routine to generate data under the null hypothesis
w	(Optional) function to calculate weights, returns -99 if no weights
phat	=function(x) -99, function to estimate parameters from the data, or -99 if no parameters are estimated
TS	user supplied function to find test statistics, if any
TSextra	=NA, list passed to TS, if desired, or NA
nbins	=c(100, 10) number of bins for chi-square tests
rate	=0 rate of Poisson if sample size is random, 0 if sample size is fixed
Range	=c(-Inf, Inf) limits of possible observations, if any, for chi-square tests
B	=c(5000,1000) number of simulation runs for individual and for adjusted p values
minexpcount	=5 minimal expected bin count required
ChiUsePhat	= TRUE, if TRUE param is estimated parameter, otherwise minimum chi square method is used.
maxProcessor	number of cores to use
doMethods	a vector of codes for the methods to include. If missing, a default selection of methods are used.

Details

For details on the usage of this routine consult the vignette with vignette("Rgof","Rgof")

Value

None

Examples

```
# Tests to see whether data comes from a standard normal distribution.
pnull = function(x) pnorm(x)
rnull = function()  rnorm(100)
x = rnorm(100)
gof_test_adjusted_pvalue(x, NA, pnull, rnull, B=c(500, 200),
                         maxProcessor=1)
# Tests to see whether data comes from a normal distribution with standard deviation 1
# and the mean estimated.
pnull=function(x, m) pnorm(x, m)
rnull=function(m) rnorm(100, m)
TSextra = list(qnull=function(x, m=0) qnorm(x, m))
phat=function(x) mean(x)
x = rnorm(100, 1, 2)
gof_test_adjusted_pvalue(x, NA, pnull, rnull, phat=phat,
                       TSextra=TSextra, B=c(500, 200), maxProcessor=1)
# Tests to see whether data comes from a binomial (10, 0.5) distribution.
vals=0:10
pnull = function() pbinom(0:10, 10, 0.5)
rnull = function() table(c(0:10, rbinom(1000, 10, 0.5)))-1
x = rnull()
gof_test_adjusted_pvalue(x, vals, pnull, rnull,
                       B=c(500, 200), maxProcessor=1)
# Tests to see whether data comes from a binomial distribution with
# the success probability estimated from the data.
pnull = function(p=0.5) pbinom(0:10, 10, p)
rnull = function(p=0.5) table(c(0:10, rbinom(1000, 10, p)))-1
phat=function(x) mean(rep(0:10,x))/10
gof_test_adjusted_pvalue(x, vals, pnull, rnull, phat=phat,
                       B=c(500, 200), maxProcessor=1)
```

`make_bins_cont` *This function creates several type of bins for continuous data*

Description

This function creates several type of bins for continuous data

Usage

```
make_bins_cont(
  x,
  pnull,
  qnull = NA,
  phat = function(x) -99,
  DataBased = FALSE,
  nbins = c(50, 10),
  minexpcount = 5,
  Range = c(-99999, 99999)
)
```

Arguments

x	data set
pnull	cdf under the null hypothesis
qnull	=NA quantile function, if available
phat	=function(x) -99 parameters for pnull
DataBased	=FALSE bins based on data, not expected counts
nbins	=c(50, 10) number of bins
minexpcount	=5 smallest expected count per bin
Range	=c(-99999, 99999) limits of possible observations, if any

Value

A list of bins and bin probabilities

---

make_bins_disc *This function creates several types of bins for discrete data*

Description

This function creates several types of bins for discrete data

Usage

```
make_bins_disc(
  x,
  pnull,
  phat = function(x) -99,
  nbins = c(50, 10),
  minexpcount = 5
)
```

Arguments

x	counts
pnull	cumulative distribution function
phat	=function(x) -99, function to estimated parameters, or -99
nbins	=c(50, 10) number of bins
minexpcount	=5 smallest expected count per bin

Value

A list of indices

newTSdisc *a local function needed for the vignette*

Description

a local function needed for the vignette

Usage

```
newTSdisc(x, pnull, param, vals)
```

Arguments

x	An integer vector.
pnull	cdf.
param	parameters for pnull in case of parameter estimation.
vals	A numeric vector with the values of the discrete rv.

Value

A vector with test statistics

plot_power *This function draws the power graph, with curves sorted by the mean power and smoothed for easier reading.*

Description

This function draws the power graph, with curves sorted by the mean power and smoothed for easier reading.

Usage

```
plot_power(pwr, xname = " ", title, Smooth = TRUE, span = 0.25)
```

Arguments

pwr	a matrix of power values, usually from the twosample_power command
xname	Name of variable on x axis
title	(Optional) title of graph
Smooth	=TRUE lines are smoothed for easier reading
span	=0.25bandwidth of smoothing method

Value

plt, an object of class ggplot.

power_newtest *This function estimates the power of test routines that calculate p value(s)*

Description

This function estimates the power of test routines that calculate p value(s)

Usage

```
power_newtest(
  TS,
  vals = NA,
  pnull,
  ralt,
  param_alt,
  phat,
  TSextra,
  alpha = 0.05,
  B = 1000
)
```

Arguments

TS	routine to calculate test statistics.
vals	=NA if data is discrete, a vector of possible values
pnull	routine to calculate the cdf under the null hypothesis
ralt	generate data under alternative hypothesis
param_alt	values of parameter under the alternative hypothesis.
phat	function to estimate parameters, function(x) -99 if no parameter estimation
TSextra	list (possibly) passed to TS
alpha	=0.05 type I error.
B	= 1000 number of simulation runs to estimate the power.

Value

A matrix of power values

power_studies_results *power_studies_results*

Description

the results of the included power studies

Usage

```
power_studies_results
```

Format

'power_studies_results':
A list of matrices with powers

pvaluecdf *pvaluecdf*

Description

the info needed to draw a graph

Usage

```
pvaluecdf
```

Format

'pvaluecdf':
A matrix

`run.studies` *Power Comparisons*

Description

This function runs the case studies included in the package and compares the power of a new test to those included.

Usage

```
run.studies(
  TS,
  study,
  TSextra = list(aaa = 1),
  With.p.value = FALSE,
  BasicComparison = TRUE,
  nsample = 500,
  alpha = 0.05,
  param_alt,
  maxProcessor,
  B = 1000
)
```

Arguments

`TS`	routine to calculate test statistic(s) or p value(s).
`study`	either the name of the study, or its number. If missing all the studies are run.
`TSextra`	=list(aaa=1), list passed to TS.
`With.p.value`	=FALSE does user supplied routine return p values?
`BasicComparison`	=TRUE if true compares tests on one default value of parameter of the alternative distribution.
`nsample`	= 500, desired sample size.
`alpha`	=0.05 type I error
`param_alt`	(list of) values of parameter under the alternative hypothesis. If missing included values are used.
`maxProcessor`	number of cores to use for parallel programming
`B`	= 1000 number of simulation runs

Details

For details on the usage of this routine consult the vignette with vignette("Rgof","Rgof")

Value

A (list of) matrices of power values

Examples

```
# New test is a simple chi-square test:
chitest=function(x, pnull, param, TSextra) {
    nbins=TSextra$nbins
    bins=quantile(x, (0:nbins)/nbins)
    O=hist(x, bins, plot=FALSE)$counts
    if(param[1]!=-99) { #with parameter estimation
        E=length(x)*diff(pnull(bins, param))
        chi=sum((O-E)^2/E)
        pval=1-pchisq(chi, nbins-1-length(param))
    }
    else {
      E=length(x)*diff(pnull(bins))
      chi=sum((O-E)^2/E)
      pval=1-pchisq(chi,nbins-1)
    }
    out=ifelse(TSextra$statistic, chi, pval)
    names(out)="ChiSquare"
    out
}
TSextra=list(nbins=10, statistic=FALSE) # Use 10 bins, test routine returns p-value
run.studies(chitest, TSextra=TSextra, With.p.value=TRUE, maxProcessor=1, B=200)
```

`signif.digits` *This function does some rounding to nice numbers*

Description

This function does some rounding to nice numbers

Usage

```
## S3 method for class 'digits'
signif(x, d = 3)
```

Arguments

x	a list of two vectors
d	=4 number of digits to round to

Value

A list with rounded vectors

timecheck *estimate run time function*

Description

estimate run time function

Usage

```
timecheck(dta, TS, typeTS, TSextra)
```

Arguments

dta	data set
TS	test statistic
typeTS	format of TS
TSextra	additional info TS

Value

Mean computation time

TS_cont *Find test statistics for continuous data*

Description

Find test statistics for continuous data

Usage

```
TS_cont(x, pnull, param, qnull)
```

Arguments

x	A numeric vector.
pnull	cdf.
param	parameters for pnull in case of parameter estimation.
qnull	An R function, the quantile function under the null hypothesis.

Value

A numeric vector with test statistics

TS_disc *Find test statistics for discrete data*

Description

Find test statistics for discrete data

Usage

```
TS_disc(x, pnull, param, vals)
```

Arguments

x	An integer vector.
pnull	cdf.
param	parameters for pnull in case of parameter estimation.
vals	A numeric vector with the values of the discrete rv.

Value

A vector with test statistics

Index