Package: Rgof (via r-universe)

November 21, 2026

Title 1d Goodness of Fit Tests

Version 4.1.1

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

LinkingTo Rcpp

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

Suggests rmarkdown, knitr, testthat (>= 3.0.0), dgof, R2sample (>= 5.1.0)

VignetteBuilder knitr

Depends R (>= 3.5)

LazyData true

Config/roxygen2/version 8.1.1

Config/testthat/edition 3

NeedsCompilation yes

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

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

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

Date/Publication 2026-11-21 16:00:08 UTC

RemoteUrl https://github.com/cran/Rgof

RemoteRef HEAD

RemoteSha 1320809359c44118a0f674e7ecc8c3e1b1d3a544

Contents

as.data.frame.Rgof_test . . . 2
case.studies . . . 3
case_studies_sample_sizes . . . 4
funs_list . . . 4
gof_power . . . 5
gof_test . . . 7
gof_test_adjusted_pvalue . . . 9
hybrid_power . . . 12
hybrid_test . . . 13
myTS_cont . . . 15
myTS_disc . . . 16
newTSdisc . . . 17
plot_power . . . 17
power.uniform.linear . . . 18
power_study . . . 18
power_study_hybrid . . . 19
power_study_hybrid_nMC5 . . . 19
print.Rgof_test . . . 20
pvaluecdf . . . 20
run.studies . . . 21
summary.Rgof_power_values . . . 22
summary.Rgof_test . . . 23
timecheck . . . 23

Index **25**

`as.data.frame.Rgof_test` *Convert an Rgof test result to a data frame*

Description

Convert an Rgof test result to a data frame

Usage

```
## S3 method for class 'Rgof_test'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)
```

Arguments

`x`	An object of class "`Rgof_test`".
`row.names`	Ignored.
`optional`	Ignored.
`...`	Currently unused.

Value

A data frame with columns `method`, `statistic`, and `p.value`.

`case.studies` *Create case studies*

Description

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

Usage

```
case.studies(
  which,
  which_diff = "mean",
  data_type = "cont",
  n,
  nbins = 50,
  funs_list,
  ReturnCaseNames = FALSE
)
```

Arguments

`which`	name of the case study, or its number in the list.
`which_diff`	="mean", name of parameter in which data sets differ, or its number in the list.
`data_type`	="cont", is data continuous or discrete?
`n`	sample sizes.
`nbins`	=50, number of bins for discrete data
`funs_list`	list of alternative functions
`ReturnCaseNames`	=FALSE, should list of case studies be returned?

Value

a list of functions to create case studies

`case_studies_sample_sizes` *case_studies_sample_sizes*

Description

sample sizes for the various case studies

Usage

```
case_studies_sample_sizes
```

Format

'case_studies_sample_sizes':
A list

`funs_list` *funs_list*

Description

function definitions for case studies

Usage

```
funs_list
```

Format

'funs_list':
A list of functions

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 = FALSE,
  SuppressMessages = FALSE,
  CI = FALSE,
  conf.level = 0.95,
  seed = NULL,
  list.with.everything,
  qnull
)
```

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	= FALSE, if TRUE param is estimated parameter, otherwise minimum chi square method is used.
SuppressMessages	=FALSE
CI	=FALSE if TRUE, return Monte Carlo standard errors and Wilson confidence intervals for the estimated power.
conf.level	=0.95 confidence level used when CI=TRUE.
seed	=NULL, optional seed for reproducibility
list.with.everything	A list with all needed info, such as pnull, rnull etc
qnull	(optional) quantile function under the null hypothesis, with the same arguments as pnull. Needed for the Wasserstein test (Wassp1) with continuous data; if it is missing that test is not run. It can also be given as TSextra=list(qnull=...), as in earlier versions.

Details

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

Value

By default, a numeric matrix (or vector for one alternative) of power values, with class "Rgof_power_values" so that summary gives a summary of the study (it prints and is used like the plain matrix). If CI=TRUE, an object of class Rgof_power containing power estimates, Monte Carlo standard errors, and Wilson confidence limits.

Examples

```
# Examples are run with B=10 to satisfy CRAN timing constraints.
# 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)
```

```
qnull = function(x) qnorm(x)
gof_power(pnull, NA, rnull, ralt, c(0.25, 0.5), qnull=qnull, B=10)
# 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=10)
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=100)
# 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=100)
```

---

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. The result is an object of class `"Rgof_test"`. The object retains the traditional `$statistics` and `$p.values` components and also contains a data-frame representation and metadata about the test.

Usage

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

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`	List passed to TS, if desired, or missing.
`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.
`Range`	=c(-Inf, Inf), limits of possible observations, if any, for chi-square tests.
`B`	=5000, number of simulation runs. If `B=0`, only test statistics are calculated and p-values are returned as `NA`.

minexpcount	=5 minimal expected bin count required.
ChiUsePhat	=FALSE, if TRUE param is estimated parameter, otherwise minimum chi square method is used.
SuppressMessages	=FALSE
seed	=NULL, optional seed for reproducibility
maxProcessor	Number of processors to use in parallel processing.
doMethods	="all", a vector of codes for the methods to include or all of them.
qnull	(optional) quantile function under the null hypothesis, with the same arguments as pnull. Needed for the Wasserstein test (Wassp1) with continuous data; if it is missing that test is not run. It can also be given as TSextra=list(qnull=...), as in earlier versions.

Details

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

Value

An object of class `"Rgof_test"`. Its main components are `results`, a data frame with method, statistic and p-value; `statistics` and `p.values`, retained for backward compatibility; `parameters`, the parameter estimates when present; and `metadata`, information about the calculation.

Examples

```
pnull <- function(x) pnorm(x)
rnull <- function() rnorm(100)
x <- rnorm(100)
z <- gof_test(x, NA, pnull, rnull, B=100)
z
# with the quantile function the Wasserstein test is included as well
gof_test(x, NA, pnull, rnull, B=100, qnull=function(x) qnorm(x))
z$results
summary(z)
as.data.frame(z)
```

---

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,
  nbins = c(50, 10),
  rate = 0,
  Range = c(-Inf, Inf),
  B = c(5000, 1000),
  minexpcount = 5,
  ChiUsePhat = FALSE,
  maxProcessor,
  SuppressMessages = FALSE,
  seed = NULL,
  doMethods,
  qnull
)
```

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	list passed to TS, if desired, or missing
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	= FALSE, if TRUE param is estimated parameter, otherwise minimum chi square method is used.
maxProcessor	number of cores to use

SuppressMessages
: =FALSE

seed
: =NULL, optional seed for reproducibility

doMethods
: a vector of codes for the methods to include. If missing, a default selection of methods are used.

qnull
: (optional) quantile function under the null hypothesis, with the same arguments as pnull. Needed for the Wasserstein test (Wassp1) with continuous data; if it is missing that test is not run. It can also be given as TSextra=list(qnull=...), as in earlier versions.

Details

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

Value

(invisibly) a list with the p values of the individual tests and the adjusted p value. These are also printed as messages.

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(100, 100),
                            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)
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,
                         qnull=qnull, B=c(100, 100), 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(100, 100), 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(100, 100), maxProcessor=1)
```

hybrid_power *Power Estimation for the goodness-of-fit problem via twosample tests*

Description

This function estimates the power of goodness-of-fit/two-sample hybrid tests (see hybrid_test) by generating data under the alternative, a Monte Carlo data set under the null hypothesis and then running the twosample tests of the package R2sample. The null distributions of the test statistics are found via the parametric bootstrap, as in hybrid_test.

Usage

```
hybrid_power(
  rnull,
  ralt,
  param_alt,
  vals = NA,
  phat = function(x) -99,
  nMC = 1,
  TS,
  TSextra,
  With.p.value = FALSE,
  alpha = 0.05,
  B = 1000,
  seed = NULL,
  maxProcessor,
  doMethods = "all",
  SuppressMessages = FALSE
)
```

Arguments

rnull	routine to generate data under the null hypothesis, with no arguments or one argument, the parameters.
ralt	routine to generate data under the alternative hypothesis.
param_alt	values passed to ralt.
vals	=NA, values of discrete random variable, or NA if data is continuous.
phat	=function(x) -99, function to estimate parameters from the data, if needed.
nMC	=1, sample size of Monte Carlo data set. If nMC<=10 the sample size used will be nMC times the sample size of the data.
TS	user supplied function to find test statistics, if any. It has to have the format required by R2sample::twosample_power.
TSextra	(optional) list passed to TS, if needed.
With.p.value	=FALSE, does user supplied method find its own p-values?

alpha	=0.05 type I error rate used in tests.
B	=1000 number of simulation runs.
seed	=NULL (optional) seed for simulations, to aid reproducibility.
maxProcessor	number of processors to use in parallel processing.
doMethods	="all", a vector of codes for the methods to include or all of them.
SuppressMessages	=FALSE, print informative messages?

Details

Requires the package R2sample.

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

Value

A matrix (or vector) of power values, with class "Rgof_power_values" so that summary gives a summary of the study (it prints and is used like the plain matrix).

Examples

```
# All examples are run with B=10 and maxProcessor=1 to pass CRAN checks.
# (they need R2sample, in a version newer than 5.0.0)
if(requireNamespace("R2sample", quietly=TRUE) &&
   utils::packageVersion("R2sample") > "5.0.0") {
  # Power of tests whether data comes from a standard normal distribution,
  # if the true distribution is normal with mean mu.
  rnull=function() rnorm(100)
  ralt=function(mu) rnorm(100, mu)
  hybrid_power(rnull, ralt, c(0, 0.3), B=10, maxProcessor=1)
  # Power of tests whether data comes from a normal distribution,
  # with mean and standard deviation estimated. True data comes from a t distribution.
  rnull=function(p) rnorm(100, p[1], p[2])
  ralt=function(df) rt(100, df)
  phat=function(x) c(mean(x), sd(x))
  hybrid_power(rnull, ralt, 5, phat=phat, B=10, maxProcessor=1)
}
```

---

hybrid_test *Tests for the goodness-of-fit problem via twosample tests*

Description

This function runs a number of goodness-of-fit tests in the case where data can be generated under the null hypothesis but its cdf can not be calculated. A large Monte Carlo data set is generated under the null hypothesis, and then the twosample tests of the package R2sample are run on the data and the Monte Carlo data set. p values are found via the parametric bootstrap.

Usage

```
hybrid_test(
  x,
  vals = NA,
  rnull,
  phat = function(x) -99,
  nMC = 1,
  TS,
  TSextra,
  B = 5000,
  seed = NULL,
  maxProcessor,
  doMethods = "all",
  SuppressMessages = FALSE
)
```

Arguments

x	data set: a numeric vector for continuous data, a vector of counts for discrete data.
vals	=NA, values of discrete random variable, or NA if data is continuous.
rnull	routine to generate data under the null hypothesis, with no arguments or one argument, the parameters.
phat	=function(x) -99, function to estimate parameters from the data, if needed.
nMC	=1, sample size of Monte Carlo data set. If nMC<=10 the sample size used will be nMC times the sample size of x.
TS	user supplied function to find test statistics, if any. It has to have the format required by R2sample::twosample_test.
TSextra	(optional) list passed to TS, if needed.
B	=5000, number of simulation runs. If B=0 the routine returns the test statistics.
seed	=NULL (optional) seed for simulations, to aid reproducibility.
maxProcessor	number of processors to use in parallel processing.
doMethods	="all", a vector of codes for the methods to include or all of them.
SuppressMessages	=FALSE, print informative messages?

Details

If parameters are estimated, the Monte Carlo data set is generated with the parameters estimated from the data. In each bootstrap run a new data set is generated with these parameters, the parameters are estimated again from this new data set, and the Monte Carlo data set is generated with those estimates.

If the cdf under the null hypothesis can be calculated, gof_test should be used instead, as it generally has higher power.

The chi square tests use the usual chi square approximation and treat the Monte Carlo data as an independent sample, so their p values are only approximate if parameters are estimated.

Requires the package R2sample.

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

Value

An object of class "Rgof_test", as returned by gof_test, with components results, statistics, p.values, parameters and metadata (which also has hybrid=TRUE and nMC, the size of the Monte Carlo data set).

Examples

```
# All examples are run with B=10 and maxProcessor=1 to pass CRAN checks.
# (they need R2sample, in a version newer than 5.0.0)
if(requireNamespace("R2sample", quietly=TRUE) &&
   utils::packageVersion("R2sample") > "5.0.0") {
  # Test whether data comes from a standard normal distribution.
  rnull=function() rnorm(100)
  x=rnull()
  hybrid_test(x, rnull=rnull, B=20, maxProcessor=1)
  # Test whether data comes from a normal distribution,
  # with mean and standard deviation estimated.
  rnull=function(p) rnorm(100, p[1], p[2])
  phat=function(x) c(mean(x), sd(x))
  x=rnull(c(0, 1))
  hybrid_test(x, rnull=rnull, phat=phat, B=10, maxProcessor=1)
  # Discrete data: test whether data comes from a binomial(10, p) distribution,
  # with p estimated
  vals=0:10
  rnull=function(p) table(c(0:10, rbinom(1000, 10, p)))-1
  phat=function(x) mean(rep(0:10, x))/10
  x=rnull(0.5)
  hybrid_test(x, vals, rnull, phat, B=10, maxProcessor=1)
}
```

---

myTS_cont *Testing routine for continuous data*

Description

find test statistic or p value for continuous data

Usage

```
myTS_cont(x, pnull, p = 0, TSextra)
```

Arguments

x	data set
pnull	function to find cdf under null hypothesis
p	=0, parameter estimates or 0
TSextra	list with additional info

Value

a number, value fo statistic or p value

---

myTS_disc *Testing routine for discrete data*

Description

find test statistic or p value for discrete data

Usage

```
myTS_disc(x, pnull, p = 0, vals, TSextra)
```

Arguments

x	data set
pnull	function to find cdf under null hypothesis
p	=0, parameter estimates or 0
vals	=NA values of discrete random variable, or NA
TSextra	list with additional info

Value

a number, value fo statistic or p value

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 = 0.75)
```

Arguments

pwr	a matrix of power values, usually from gof_power or hybrid_power
xname	Name of variable on x axis
title	(Optional) title of graph
Smooth	=TRUE lines are smoothed for easier reading. Smoothing needs enough alternatives; with fewer than 10 values of the parameter the powers are joined by straight lines instead.
span	=0.75 span of the loess smoother (as a fraction of the points)

Value

plt, an object of class ggplot.

`power.uniform.linear` *power.uniform.linear*

Description

matrix with power values, for vignette

Usage

```
power.uniform.linear
```

Format

'power.uniform.linear':

matrix with power values, for vignette

`power_study` *power_study*

Description

the results of the included power studies

Usage

```
power_study
```

Format

'power_study':

A list with parts cont and disc, each a list of matrices with powers (case studies in the rows, methods in the columns). attr(power_study, "Null") has the type I errors, a list with matrices cont and disc.

power_study_hybrid *power_study_hybrid*

Description

the results of the included power studies for the hybrid tests (hybrid_power), with a Monte Carlo data set of the same size as the data (nMC=1)

Usage

```
power_study_hybrid
```

Format

'power_study_hybrid':

As power_study, including attr(power_study_hybrid, "Null").

power_study_hybrid_nMC5 *power_study_hybrid_nMC5*

Description

the results of the included power studies for the hybrid tests (hybrid_power), with a Monte Carlo data set five times the size of the data (nMC=5)

Usage

```
power_study_hybrid_nMC5
```

Format

'power_study_hybrid_nMC5':

As power_study, including attr(power_study_hybrid_nMC5, "Null").

print.Rgof_test *Print an Rgof goodness-of-fit test result*

Description

Print an Rgof goodness-of-fit test result

Usage

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

Arguments

x	An object of class "Rgof_test".
...	Further arguments passed to print.data.frame.

Value

x, invisibly.

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(
  studies = 1:60,
  types = 1:5,
  data_type = "cont",
  TS,
  TSextra,
  With.p.value = FALSE,
  n,
  alpha = 0.05,
  B = 1000,
  SuppressMessages = TRUE,
  RerunIncludedCases = FALSE,
  seed = NULL,
  maxProcessor,
  Null = FALSE,
  Hybrid = FALSE,
  nMC = 1
)
```

Arguments

`studies`	=1:60, either the names of the studies, or their numbers. If missing all the studies are run.
`types`	=1:5, either the types of studies, or their numbers. If missing all types are run.
`data_type`	="cont", continuous or discrete data
`TS`	routine to calculate test statistics. If Hybrid=TRUE it has to have the format required by R2sample (a two-sample test).
`TSextra`	list passed to TS.
`With.p.value`	=FALSE does user supplied routine return p values?
`n`	desired sample size.
`alpha`	=0.05 type I error
`B`	= 1000
`SuppressMessages`	=TRUE

`RerunIncludedCases`
: =FALSE, set to true to verify powers in Rgof::power_study (or power_study_hybrid, power_study_hybrid_nMC5)

`seed`
: =NULL, optional seed for reproducibility. Case study i in the list of studies/types that are run uses the seed seed+i, both for the new test and (if they are recomputed) for the included methods.

`maxProcessor`
: number of processors used in gof_power or hybrid_power. If missing it is chosen automatically. Results with a given seed are reproducible only with the same number of processors.

`Null`
: =FALSE, if TRUE the data is generated under the null hypothesis (by rnull), so the result is the type I error.

`Hybrid`
: =FALSE, if TRUE the goodness-of-fit/two-sample hybrid tests (hybrid_power, requires package R2sample) are run instead of the goodness-of-fit tests.

`nMC`
: =1, size of the Monte Carlo data set for the hybrid tests, as a multiple of the sample size. Stored results exist for nMC=1 (power_study_hybrid) and nMC=5 (power_study_hybrid_nMC5).

Details

For details consult vignette("Rgof","Rgof")

Value

A matrix of power values.

Examples

```
TSextra=list(nbins=5, statistic=FALSE) # Use 5 bins and calculate p values
run.studies(1:2, 1:2, TS=myTS_cont, TSextra=TSextra, With.p.value=TRUE, B=10)
```

---

summary.Rgof_power_values

Summarize an Rgof power study

Description

Summarize an Rgof power study

Usage

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

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

Arguments

object	An object of class "Rgof_power_values" (the default output of gof_power) or "Rgof_power" (the output with CI=TRUE).
...	Currently unused.

Value

An object of class "summary.Rgof_power".

---

summary.Rgof_test *Summarize an Rgof goodness-of-fit test result*

Description

Summarize an Rgof goodness-of-fit test result

Usage

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

Arguments

object	An object of class "Rgof_test".
...	Currently unused.

Value

An object of class "summary.Rgof_test".

---

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

Index

∗ **datasets**
case_studies_sample_sizes, 4
funs_list, 4
power.uniform.linear, 18
power_study, 18
power_study_hybrid, 19
power_study_hybrid_nMC5, 19
pvaluecdf, 20

as.data.frame.Rgof_test, 2

case.studies, 3
case_studies_sample_sizes, 4

funs_list, 4

gof_power, 5
gof_test, 7
gof_test_adjusted_pvalue, 9

hybrid_power, 12
hybrid_test, 13

myTS_cont, 15
myTS_disc, 16

newTSdisc, 17

plot_power, 17
power.uniform.linear, 18
power_study, 18
power_study_hybrid, 19
power_study_hybrid_nMC5, 19
print.Rgof_test, 20
pvaluecdf, 20

run.studies, 21

summary.Rgof_power
*(*summary.Rgof_power_values*)*, 22
summary.Rgof_power_values, 22
summary.Rgof_test, 23

timecheck, 23