

# Package ‘MDgof’

February 12, 2026

**Title** Various Methods for the Goodness-of-Fit Problem in  $D > 1$   
Dimensions

**Version** 1.0.0

**Description** The routine `gof_test()` in this package runs the goodness-of-fit test using various test statistic for multivariate data. Models under the null hypothesis can either be simple or allow for parameter estimation. p values are found via the parametric bootstrap (simulation). The routine `gof_test_adjusted_pvalues()` runs several tests and then finds a p value adjusted for simultaneous inference. The routine `gof_power()` allows the estimation of the power of the tests. `hybrid_test()` and `hybrid_power()` do the same by first generating a Monte Carlo data set under the null hypothesis and then running a number of two-sample methods. The routine `run.studies()` allows a user to quickly study the power of a new method and how it compares to those included in the package via a large number of case studies. For details of the methods and references see the included vignettes.

**License** GPL ( $\geq 2$ )

**Encoding** UTF-8

**RoxygenNote** 7.3.3

**LinkingTo** Rcpp

**Imports** Rcpp, parallel, stats, microbenchmark, spatstat.geom,  
spatstat.explore, FNN, copula, mvtnorm, ggplot2,  
microbenchmark, MD2sample

**Suggests** rmarkdown, knitr

**VignetteBuilder** knitr

**Depends** R ( $\geq 3.5$ )

**LazyData** true

**NeedsCompilation** yes

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

**Maintainer** Wolfgang Rolke <[wolfgang.rolke@upr.edu](mailto:wolfgang.rolke@upr.edu)>

**Repository** CRAN

**Date/Publication** 2026-02-12 20:40:03 UTC

## Contents

bakshaev_rudzkis	3
case.studies	3
case.studies.cont	4
case.studies.cont.D5	5
case.studies.disc	5
case.studies.est	6
check.functions	6
chi_cont_test	7
chi_disc_test	7
chi_power	8
compute_M_for_dtaset	9
discretize	10
draw_case	10
estimateEV	11
examples.mdgof.vignette	12
gauss_kernel_matrix	12
gen.cop	13
gen_eval	13
gof_power	14
gof_test	16
gof_test_adjusted_pvalue	18
grad_mat	20
hybrid.mdgof.vignette	21
hybrid_power	21
hybrid_test	23
ksd	24
makeTSextra	25
newTS	26
p2dC	26
power_studies_cont_D5_results	27
power_studies_cont_est_results	27
power_studies_cont_hybrid_results	28
power_studies_cont_nMC5_hybrid_results	28
power_studies_cont_results	29
power_studies_disc_est_results	29
power_studies_disc_hybrid_results	30
power_studies_disc_nMC5_hybrid_results	30
power_studies_disc_results	31
RipleyK	31
run.studies	32
signif.digits	33
sq2rec	34
timecheck	34
TS_cont	35
TS_disc	35

---

bakshaev_rudzkis	<i>Run Bakshaev and Rudzkis Test</i>
------------------	--------------------------------------

---

**Description**

Run Bakshaev and Rudzkis Test

**Usage**

```
bakshaev_rudzkis(dta, rnull, p, m_eval = 100L, nsim = 200L, nsim_mc = 1000L)
```

**Arguments**

dta	data matrix.
rnull	generate new data.
p	, parameters for parametric bootstrap.
m_eval	=100, number of evaluation points of kde.
nsim	=200, number of simulation runs.
nsim_mc	=1000, number of simulation runs.

**Value**

a list

---

case.studies	<i>Create various case studies</i>
--------------	------------------------------------

---

**Description**

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

**Usage**

```
case.studies(  
  which,  
  Continuous = TRUE,  
  WithEstimation = FALSE,  
  Dim = 2,  
  nsample = 250,  
  nbins = c(5, 5),  
  ReturnCaseNames = FALSE  
)
```

**Arguments**

which                name or number of the case study.  
 Continuous        = TRUE for continuous data  
 WithEstimation   =FALSE, with parameter estimation  
 Dim                =2 dimension of data  
 nsample            =250, sample size.  
 nbins              =c(5,5) number of bins in x and y direction  
 ReturnCaseNames   =FALSE, just return names of case studies?

**Value**

a list of functions

---

case.studies.cont	<i>Create various case studies for continuous data without parameter estimation</i>
-------------------	---

---

**Description**

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

**Usage**

```
case.studies.cont(which, nsample = 250, ReturnCaseNames = FALSE)
```

**Arguments**

which                name of the case study.  
 nsample            =250, sample size.  
 ReturnCaseNames   =FALSE, just return names of case studies?

**Value**

a list of functions

---

case.studies.cont.D5 *Create various case studies for continuous data in 5 dimensions without parameter estimation*

---

### Description

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

### Usage

```
case.studies.cont.D5(which, nsample = 250, ReturnCaseNames = FALSE)
```

### Arguments

which                    name of the case study.  
 nsample                 =250, sample size.  
 ReturnCaseNames       =FALSE, just return names of case studies?

### Value

a list of functions

---

case.studies.disc        *Discretize 2D data from case studies*

---

### Description

This function provides the info necessary to run the case studies for discrete data.

### Usage

```
case.studies.disc(
  which,
  WithEstimation = FALSE,
  nbins = c(5, 5),
  nsample = 250
)
```

### Arguments

which                    name or number of desired case study.  
 WithEstimation        = FALSE, case study with or without parameter estimation.  
 nbins                   =c(5, 5) number of bins to use in x and y direction  
 nsample                 = 250, required sample size

**Value**

a list with needed stuff

---

case.studies.est      *Create various case studies with parameter estimation*

---

**Description**

This function creates the functions needed to run the various case studies that include parameter estimation.

**Usage**

```
case.studies.est(which, nsample = 250, ReturnCaseNames = FALSE)
```

**Arguments**

which                    name of the case study.  
 nsample                 =250, sample size.  
 ReturnCaseNames       =FALSE, just return names of case studies?

**Value**

a list of functions

---

check.functions      *Sanity Checks*

---

**Description**

This function checks whether the inputs have the correct format

**Usage**

```
check.functions(pnull, rnull, phat = function(x) -99, 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  
 x                        matrix with data

---

chi_cont_test	<i>Chi-square test for 2D data</i>
---------------	------------------------------------

---

**Description**

This function does the chi square goodness-of-fit test for continuous data in two dimensions.

**Usage**

```
chi_cont_test(
  dta,
  pnull,
  phat = function(x) -99,
  Ranges = matrix(c(-Inf, Inf, -Inf, Inf), 2, 2),
  nbins = c(5, 5),
  minexpcount = 5,
  SuppressMessages = TRUE
)
```

**Arguments**

dta	a matrix of numbers.
pnull	function to calculate expected counts.
phat	=function(x) -99, function to estimate parameters of pnull.
Ranges	=matrix(c(-Inf, Inf, -Inf, Inf),2,2), a 2x2 matrix with lower and upper bounds
nbins	=c(5,5) number of bins in x and y direction
minexpcount	=5 minimum counts required per bin
SuppressMessages	=FALSE, should info be shown?

**Value**

a matrix with statistics, p values and degree of freedoms

---

chi_disc_test	<i>Chi-square test for discrete 2D data</i>
---------------	---

---

**Description**

This function does the chi square goodness-of-fit test for discrete data in two dimensions.

**Usage**

```
chi_disc_test(
  dta,
  pnull,
  dnull,
  phat = function(x) -99,
  minexpcount = 5,
  SuppressMessages = FALSE
)
```

**Arguments**

dta                    a matrix of numbers.  
 pnull                distribution function to calculate expected counts.  
 dnull                density to calculate expected counts.  
 phat                 =function(x) -99, function to estimate parameters of pnull.  
 minexpcount        =5 minimum counts required per bin  
 SuppressMessages   =TRUE, should info be shown?

**Value**

a vector with statistic, p value and degree of freedom

---

 chi\_power

*Power Estimation of Chi Square Tests*


---

**Description**

This function finds the power of various chi-square tests.

**Usage**

```
chi_power(
  pnull,
  ralt,
  param_alt,
  phat = function(x) -99,
  alpha = 0.05,
  Ranges = matrix(c(-Inf, Inf, -Inf, Inf), 2, 2),
  nbins = c(5, 5),
  rate = 0,
  minexpcount = 5,
  dnull = function(x) -99,
  Retry = TRUE,
  SuppressMessages = TRUE,
  B = 1000
)
```



**Arguments**

pnull	distribution 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, function to estimate parameters
alpha	=0.05, the level of the hypothesis test
Ranges	=matrix(c(-Inf, Inf, -Inf, Inf),2,2), a 2x2 matrix with lower and upper bounds, if any
nbins	=c(5, 5), 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
dnull	=function(x) -99, density function to find probabilities under null hypothesis, mostly used for discrete data, or -99 if missing.
Retry	=TRUE, retry if test fails?
SuppressMessages	=TRUE, should info be shown?
B	=1000 number of simulation runs to find power

**Value**

A numeric matrix of power values.

---

compute\_M\_for\_dtaset    *Compute M statistic for one dtaset*

---

**Description**

Compute M statistic for one dtaset

**Usage**

```
compute_M_for_dtaset(dta, Eval, hs, rnull, p, nsim_mc)
```

**Arguments**

dta	data matrix.
Eval	matrix of evaluations
hs	bandwidths
rnull	generate new data
p	values for parametric bootstrap
nsim_mc	number of simulation runs

**Value**

a double

---

discretize	<i>Bins continuous data</i>
------------	-----------------------------

---

**Description**

Bins continuous data

**Usage**

```
discretize(x, Range, nbins, ChangeVals = FALSE)
```

**Arguments**

x	A numeric matrix with two columns.
Range	range of variables
nbins	number of bins.
ChangeVals	=FALSE, should values of discrete rv's be adjusted to midpoints?

**Value**

A numeric matrix

---

draw_case	<i>Create plot for any case study</i>
-----------	---------------------------------------

---

**Description**

This function illustrates any of the case studies.

**Usage**

```
draw_case(  
  which,  
  Continuous = TRUE,  
  WithEstimation = FALSE,  
  Dim = 2,  
  palt,  
  nsample = 1000,  
  Dms = c(1, 2),  
  AltOnly = FALSE  
)
```

**Arguments**

which	name or number of the case study.
Continuous	= TRUE for continuous data
WithEstimation	=FALSE, with parameter estimation
Dim	=2 dimension of data
palt	parameter for alternative. If missing value in study is used.
nsample	=250, sample size.
Dms	=c(1,2, which dimensions are to be shown (for 5D data).
AltOnly	= FALSE show only graph for alternative?

**Value**

a ggplot2 object

---

estimateEV	<i>Estimate E and Var/n at Eval for given h, using MC from rnull</i>
------------	--

---

**Description**

Estimate E and Var/n at Eval for given h, using MC from rnull

**Usage**

```
estimateEV(rnull, p, Eval, h, nsim_mc, n)
```

**Arguments**

rnull	generate data under the null hypothesis.
p	values for rnull
Eval	matrix of evaluations
h	bandwidth, a double
nsim_mc	number of simulation runs
n	sample size

**Value**

a matrix

examples.mdgof.vignette

*examples.mdgof.vignette*

---

**Description**

stuff needed to run vignette fast enough to pass CRAN

**Usage**

examples.mdgof.vignette

**Format**

**'examples.mdgof.vignette':**

A list

---

gauss\_kernel\_matrix *Find gaussian kernel pdf*

---

**Description**

Find gaussian kernel pdf

**Usage**

gauss\_kernel\_matrix(Eval, S, h)

**Arguments**

Eval	a matrix.
S	a matrix
h	bandwidth, a double

**Value**

a matrix

---

gen.cop	<i>Create copula objects</i>
---------	------------------------------

---

**Description**

This function creates copula objects

**Usage**

```
gen.cop(family, p, d = 2)
```

**Arguments**

family	name of copula.
p	parameter of copula.
d	dimension

**Value**

a copula object

---

gen_eval	<i>Find evaluation points</i>
----------	-------------------------------

---

**Description**

Find evaluation points

**Usage**

```
gen_eval(rnull, p, m)
```

**Arguments**

rnull	a function that generate new data.
p	a vector of parameters for rnull.
m	size of matrix.

**Value**

a matrix

gof\_power

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

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

**Usage**

```
gof_power(
  pnull,
  rnull,
  ralt,
  param_alt,
  phat = function(x) -99,
  dnull = function(x) -99,
  TS,
  TSextra,
  With.p.value = FALSE,
  alpha = 0.05,
  Ranges = matrix(c(-Inf, Inf, -Inf, Inf), 2, 2),
  nbins = c(5, 5),
  minexpcount = 5,
  rate = 0,
  SuppressMessages = FALSE,
  maxProcessor,
  B = 1000
)
```

**Arguments**

pnull	function to find cdf under null hypothesis
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
phat	=function(x) -99, function to estimate parameters from the data, or -99
dnull	=function(x) -99, density function under the null hypothesis, if available, or -99 if missing
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
Ranges	=matrix(c(-Inf, Inf, -Inf, Inf),2,2), a 2x2 matrix with lower and upper bounds, if any, for chi-square tests

**nbins** =c(5, 5), number of bins for chi square tests.  
**minexpcount** =5 minimal expected bin count required for chi square tests.  
**rate** =0 rate of Poisson if sample size is random, 0 if sample size is fixed  
**SuppressMessages**  
 =FALSE, should informative messages be shown?  
**maxProcessor** maximum of number of processors to use, 1 if no parallel processing is needed  
 or number of cores-1 if missing  
**B** =1000 number of simulation runs

### Details

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

### Value

A numeric matrix of power values.

### Examples

```

# All examples are run with B=10 and maxProcessor=1 to pass CRAN checks.
# This is obviously MUCH TO SMALL for any real usage.
# Power of tests if null hypothesis specifies a bivariate standard normal
# distribution but data comes from a bivariate normal with different means,
# without parameter estimation.
rnull=function() mvtnorm::rmvnorm(100, c(0, 0))
ralt=function(p) mvtnorm::rmvnorm(100, c(p, p))
pnull=function(x) {
  if(!is.matrix(x)) return(mvtnorm::pmvnorm(rep(-Inf, 2), x))
  apply(x, 1, function(x) mvtnorm::pmvnorm(rep(-Inf, 2), x))
}
gof_power(pnull, rnull, ralt, c(0, 1), B=10, maxProcessor = 1)
# Same as above, but now with density included
dnull=function(x) {
  if(!is.matrix(x)) return(mvtnorm::dmvnorm(x))
  apply(x, 1, function(x) mvtnorm::dmvnorm(x))
}
gof_power(pnull, rnull, ralt, c(0, 1), dnull=dnull, B=10, maxProcessor = 1)
# Power of tests when null hypothesis specifies a bivariate normal distribution,
# with mean parameter estimated, whereas data comes from a t distribution
rnull=function(p) mvtnorm::rmvnorm(100, p)
ralt=function(df) mvtnorm::rmvt(100, sigma=diag(2), df=df)
pnull=function(x,p) {
  if(!is.matrix(x)) return(mvtnorm::pmvnorm(rep(-Inf, 2), x, mean=p))
  apply(x, 1, function(x) mvtnorm::pmvnorm(rep(-Inf, 2), x, mean=p))
}
dnull=function(x, p) {
  if(!is.matrix(x)) return(mvtnorm::dmvnorm(x, mean=p))
  apply(x, 1, function(x) mvtnorm::dmvnorm(x, mean=p))
}
phat=function(x) apply(x, 2, mean)

```

```

gof_power(pnull, rnull, ralt, c(50, 5), dnull=dnull, phat=phat, B=10, maxProcessor = 1)
# Example of a discrete model, with parameter estimation
# Under null hypothesis:  $X \sim \text{Bin}(10, p)$ ,  $Y|X \sim \text{Bin}(5, 0.5+x/100)$ 
# Under alternative hypothesis:  $X \sim \text{Bin}(10, p)$ ,  $Y|X \sim \text{Bin}(5, K+x/100)$ 
rnull=function(p=0.5) {
  x=stats::rbinom(1000, 10, p)
  y=stats::rbinom(1000, 5, 0.5+x/100)
  MDgof::sq2rec(table(x, y))
}
ralt=function(K=0.5) {
  x=stats::rbinom(1000, 10, 0.5)
  y=stats::rbinom(1000, 5, K+x/100)
  MDgof::sq2rec(table(x, y))
}
pnull=function(x, p) {
  f=function(x) sum(dbinom(0:x[1], 10, p[1])*pbinom(x[2], 5, 0.5+0:x[1]/100))
  if(!is.matrix(x)) x=rbind(x)
  apply(x, 1, f)
}
phat=function(x) {
  tx=tapply(x[,3], x[,1], sum)
  mean(rep(as.numeric(names(tx)), times=tx))/10
}
gof_power(pnull, rnull, ralt, c(0.5, 0.6), phat=phat, B=10, maxProcessor = 1)

```

---

gof\_test

*Tests for the multivariate goodness-of-fit problem*


---

## Description

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

## Usage

```

gof_test(
  x,
  pnull,
  rnull,
  phat = function(x) -99,
  dnull = function(x) -99,
  TS,
  TSextra,
  rate = 0,
  nbins = c(5, 5),
  Ranges = matrix(c(-Inf, Inf, -Inf, Inf), 2, 2),
  minexpcount = 5,
  maxProcessor,
  doMethods,
  B = 5000,

```



```

    ReturnTSextra = FALSE
  )

```

### Arguments

x	a matrix with the data set
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 if no parameters are estimated
dnull	=function(x) -99, density function under the null hypothesis, if available, or -99 if missing
TS	user supplied function to find test statistics, if any.
TSextra	(optional) list passed to TS, if needed.
rate	=0 rate of Poisson if sample size is random, 0 if sample size is fixed
nbins	=c(5, 5) number of bins for chi-square tests
Ranges	=matrix(c(-Inf, Inf, -Inf, Inf),2,2), a 2x2 matrix with lower and upper bounds, if any, for chi-square tests
minexpcount	=5 minimal expected bin count required
maxProcessor	number of processors to use in parallel processing.
doMethods	a vector of codes for the methods to include. If ="all", it does all the included tests. #missing it runs a default selection. I
B	=5000 number of simulation runs. If B=0 the routine returns the test statistics.
ReturnTSextra	=FALSE, should setup info be returned?

### Details

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

### Value

A list with vectors of test statistics and p.values

### Examples

```

# All examples are run with B=10 and maxProcessor=1 to pass CRAN checks.
# This is obviously MUCH TO SMALL for any real usage.
# Tests to see whether data comes from a bivariate standard normal distribution,
# without parameter estimation.
rnull=function() mvtnorm::rmvnorm(100, c(0, 0))
x=rnull()
pnull=function(x) {
  if(!is.matrix(x)) return(mvtnorm::pmvnorm(rep(-Inf, 2), x))
  apply(x, 1, function(x) mvtnorm::pmvnorm(rep(-Inf, 2), x))
}
gof_test(x, pnull, rnull, B=10, maxProcessor = 1)

```

```

# Same as above, but now with density included
dnull=function(x) {
  if(!is.matrix(x)) return(mvtnorm::dmvnorm(x))
  apply(x, 1, function(x) mvtnorm::dmvnorm(x))
}
gof_test(x, pnull, rnull, dnull=dnull, B=20, maxProcessor = 1)
# Tests to see whether data comes from a standard normal distribution,
# with mean parameter estimated.
rnull=function(p) mvtnorm::rmvnorm(100, p)
x=rnull(c(0,1))
pnull=function(x,p) {
  if(!is.matrix(x)) return(mvtnorm::pmvnorm(rep(-Inf, 2), x, mean=p))
  apply(x, 1, function(x) mvtnorm::pmvnorm(rep(-Inf, 2), x, mean=p))
}
dnull=function(x, p) {
  if(!is.matrix(x)) return(mvtnorm::dmvnorm(x, mean=p))
  apply(x, 1, function(x) mvtnorm::dmvnorm(x, mean=p))
}
phat=function(x) apply(x, 2, mean)
gof_test(x, pnull, rnull, dnull=dnull, phat=phat,B=20, maxProcessor = 1)
# Example of a discrete model, with parameter estimation
#  $X \sim \text{Bin}(10, p1)$ ,  $Y|X=x \sim \text{Bin}(5, p2+x/100)$ 
rnull=function(p) {
  x=rbinom(1000, 10, p[1])
  y=rbinom(1000, 5, p[2]+x/100)
  MDgof::sq2rec(table(x, y))
}
pnull=function(x, p) {
  f=function(x) sum(dbinom(0:x[1], 10, p[1])*pbinom(x[2], 5, p[2]+0:x[1]/100))
  if(!is.matrix(x)) x=rbind(x)
  apply(x, 1, f)
}
phat=function(x) {
  tx=tapply(x[,3], x[,1], sum)
  p1=mean(rep(as.numeric(names(tx)), times=tx))/10
  ty=tapply(x[,3], x[,2], sum)
  p2=mean(rep(as.numeric(names(ty)), times=ty))/5-p1/10
  c(p1, p2)
}
x=rnull(c(0.5, 0.5))
gof_test(x, pnull, rnull, phat=phat,B=10, maxProcessor = 1)

```

---

gof\_test\_adjusted\_pvalue

*Adjusted p values*

---

### Description

This function runs a number of goodness-f-fit tests using Rcpp and parallel computing and then finds the correct p value for the combined tests.

**Usage**

```

gof_test_adjusted_pvalue(
  x,
  pnull,
  rnull,
  phat = function(x) -99,
  dnull = function(x) -99,
  B = c(5000, 1000),
  nbins = c(5, 5),
  minexpcount = 5,
  Ranges = matrix(c(-Inf, Inf, -Inf, Inf), 2, 2),
  SuppressMessages = FALSE,
  maxProcessor,
  doMethods
)

```

**Arguments**

x	matrix with data
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 if no parameters are estimated
dnull	=function(x) -99, density function under the null hypothesis, if available, or -99 if missing
B	=c(5000, 1000), number of simulation runs for permutation test and for estimation of the empirical distribution function.
nbins	=c(5, 5), number of bins for chi square tests (2D only).
minexpcount	= 5, minimum required expected counts for chi-square tests.
Ranges	=matrix(c(-Inf, Inf, -Inf, Inf),2,2) a 2x2 matrix with lower and upper bounds.
SuppressMessages	= FALSE, show informative messages?
maxProcessor	number of cores for parallel processing.
doMethods	Which methods should be included? If missing a small number of methods that generally have good power are used.

**Details**

For details consult the vignette("MDgof", "MDgof")

**Value**

a vector of p values.

**Examples**

```
# All examples are run with B=10 and maxProcessor=1 to pass CRAN checks.
# This is obviously MUCH TO SMALL for any real usage.
# Tests to see whether data comes from a bivariate standard normal distribution,
# without parameter estimation.
rnull=function() mvtnorm::rmvnorm(100, c(0, 0))
x=rnull()
pnull=function(x) {
  if(!is.matrix(x)) return(mvtnorm::pmvnorm(rep(-Inf, 2), x))
  apply(x, 1, function(x) mvtnorm::pmvnorm(rep(-Inf, 2), x))
}
dnull=function(x) {
  if(!is.matrix(x)) return(mvtnorm::dmvnorm(x))
  apply(x, 1, function(x) mvtnorm::dmvnorm(x))
}
gof_test_adjusted_pvalue(x, pnull, rnull, dnull=dnull, B=10, maxProcessor = 1)
```

---

grad\_mat

*Find gradient of log(f) for a matrix of points*

---

**Description**

Find gradient of log(f) for a matrix of points

**Usage**

```
grad_mat(x, f)
```

**Arguments**

x	point of evaluation
f	function

**Value**

a matrix of gradient vectors

---

hybrid.mdgof.vignette *hybrid.mdgof.vignette*

---

### Description

stuff needed to run vignette MDgof::hybrid fast enough to pass CRAN

### Usage

```
hybrid.mdgof.vignette
```

### Format

**'hybrid.mdgof.vignette':**

A list

---

hybrid_power	<i>Power Estimation for the multivariate goodness-of-fit problem via twosample tests</i>
--------------	--

---

### Description

This function estimates the power of goodness-of-fit/two-sample hybrid tests using Rcpp and parallel computing by generating a Monte Carlo data set and then running a twosample test.

### Usage

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

**Arguments**

<code>rnull</code>	routine to generate data under the null hypothesis.
<code>ralt</code>	routine to generate data under the alternative hypothesis.
<code>param_alt</code>	values passed to <code>ralt</code> .
<code>phat</code>	<code>=function(x) -99</code> parameter estimation, if needed.
<code>nMC</code>	<code>=1</code> sample size of Monte Carlo data set, if it is a number <code>nMC &lt;= 10</code> sample size used will be <code>nMC * sample size of x</code> .
<code>TS</code>	user supplied function to find test statistics, if any.
<code>TSextra</code>	(optional) list passed to <code>TS</code> , if needed.
<code>With.p.value</code>	<code>=FALSE</code> , does user supplied method find its own p-values?
<code>alpha</code>	<code>=0.05</code> type I error rate used in tests.
<code>B</code>	<code>=5000</code> number of simulation runs. If <code>B=0</code> the routine returns the test statistics.
<code>maxProcessor</code>	number of processors to use in parallel processing.
<code>doMethods</code>	<code>"all"</code> , 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("MDgof-hybrid", "MDgof-hybrid")`

**Value**

A list with vectors of test statistics and p.values

**Examples**

```
# All examples are run with B=20 and maxProcessor=1 to pass CRAN checks.
# Power of tests see whether data comes from a bivariate standard normal distribution,
# without parameter estimation. True Distribution is bivariate normal with
# correlation r.
rnull=function() mvtnorm::rmvnorm(100, c(0, 0))
ralt=function(r) mvtnorm::rmvnorm(100, sigma=matrix(c(1,r,r,1),2,2))
hybrid_power(rnull, ralt, 0.3, B=20, maxProcessor = 1)
# Power of tests to see whether data comes from a standard normal distribution,
# with mean parameter estimated. True data comes from t distribution.
rnull=function(p) mvtnorm::rmvnorm(100, p)
ralt=function(df) mvtnorm::rmvt(100, df=df)
phat=function(x) apply(x, 2, mean)
hybrid_power(rnull, ralt, 5, phat, B=20, maxProcessor = 1)
```

---

 hybrid\_test

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


---

**Description**

This function runs a number of goodness-of-fit tests using Rcpp and parallel computing by generating a Monte Carlo data set and then running a twosample test.

**Usage**

```

hybrid_test(
  x,
  rnull,
  phat = function(x) -99,
  nMC = 1,
  TS,
  TSextra,
  B = 1000,
  maxProcessor,
  doMethods = "all"
)

```

**Arguments**

x	a matrix with the data set
rnull	routine to generate data under the null hypothesis.
phat	=function(x) -99 parameter estimation, if needed.
nMC	=1 sample size of Monte Carlo data set, if it is a number nMC<=10 sample size used will be nMC*sample size of x.
TS	user supplied function to find test statistics, if any.
TSextra	(optional) list passed to TS, if needed.
B	=5000 number of simulation runs. If B=0 the routine returns the test statistics.
maxProcessor	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("MDgof-hybrid", "MDgof-hybrid")

**Value**

A list with vectors of test statistics and p.values

**Examples**

```

# All examples are run with B=20 and maxProcessor=1 to pass CRAN checks.
# Tests to see whether data comes from a bivariate standard normal distribution,
# without parameter estimation.
rnull=function() mvtnorm::rmvnorm(100, c(0, 0))
x=rnull()
hybrid_test(x, rnull, B=20, maxProcessor = 1)
# Tests to see whether data comes from a standard normal distribution,
# with mean parameter estimated.
rnull=function(p) mvtnorm::rmvnorm(100, p)
phat=function(x) apply(x, 2, mean)
x=rnull(c(0,1))
hybrid_test(x, rnull, phat, B=20, maxProcessor = 1)
# Example of a discrete model, without parameter estimation
#  $X \sim \text{Bin}(5, 0.5)$ ,  $Y|X=x \sim \text{Bin}(4, 0.5+x/100)$ 
rnull=function() {
  x=rbinom(1000, 5, 0.5)
  y=rbinom(1000, 4, 0.5)
  MDgof::sq2rec(table(x, y))
}
x=rnull()
hybrid_test(x, rnull, B=50, maxProcessor = 1)
# Example of a discrete model, with parameter estimation
#  $X \sim \text{Bin}(5, p)$ ,  $Y|X=x \sim \text{Bin}(4, 0.5+x/100)$ 
rnull=function(p) {
  x=rbinom(1000, 5, p)
  y=rbinom(1000, 4, 0.5+x/100)
  MDgof::sq2rec(table(x, y))
}
phat=function(x) {
  tx=tapply(x[,3], x[,1], sum)
  p1=mean(rep(as.numeric(names(tx)), times=tx))/5
  p1
}
x=rnull(0.5)
hybrid_test(x, rnull, phat, B=20, maxProcessor = 1)

```

ksd

*Find test statistic for Kernel Stein Discrepancy test***Description**

Find test statistic for Kernel Stein Discrepancy test

**Usage**

```
ksd(X, scf, p)
```



**Arguments**

X	data set.
scf	function to find scores
p	(possible) parameters

**Value**

a double (test statistic)

---

makeTSextra	<i>Create list with needed info</i>
-------------	-------------------------------------

---

**Description**

This function creates a list with info needed in various parts of the package

**Usage**

```
makeTSextra(
  x,
  Continuous,
  pnull,
  rnull,
  phat = function(x) -99,
  dnull = function(x) -99,
  Ranges,
  TSextra
)
```

**Arguments**

x	data set
Continuous	=TRUE, is data continuous?
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 if no parameters are estimated
dnull	=function(x) -99, density function under the null hypothesis, if available, or -99 if missing
Ranges	Range of variables
TSextra	(optional) list passed to TS, if needed.

**Value**

A list with vectors of test statistics and p.values

---

 newTS

*Example for a new test*


---

**Description**

This shows how a new test routine can be used with Mgof, based on chi square tests

**Usage**

```
newTS(x, pnull, p, TSextra)
```

**Arguments**

x	a data set.
pnull	function to calculate expected counts.
p	parameter for pnull, if needed
TSextra	a list with setup info

**Value**

a vector with either values of the test statistic, or p values

---

p2dC

*Find probabilities from cdf for discrete data*


---

**Description**

Find probabilities from cdf for discrete data

**Usage**

```
p2dC(x, cdf, p, Fx = as.numeric(c(-1)))
```

**Arguments**

x	matrix with data
cdf	function to find distribution function
p	(possible) arguments for cdf
Fx	(if available) already calculated values of cdf

**Value**

a matrix with probabilities added

---

`power_studies_cont_D5_results`  
*power\_studies\_cont\_D5\_results*

---

**Description**

the results of the included power studies for continuous data without estimation in 5 dimensions

**Usage**

`power_studies_cont_D5_results`

**Format**

**'power\_studies\_cont\_D5\_results':**  
A list of matrices with powers

---

`power_studies_cont_est_results`  
*power\_studies\_cont\_est\_results*

---

**Description**

the results of the included power studies for continuous data with estimation

**Usage**

`power_studies_cont_est_results`

**Format**

**'power\_studies\_cont\_est\_results':**  
A list of matrices with powers

---

`power_studies_cont_hybrid_results`

*power\_studies\_cont\_hybrid\_results*

---

**Description**

the results of the included power studies for continuous data without estimation using two-sample methods

**Usage**

`power_studies_cont_hybrid_results`

**Format**

**'power\_studies\_cont\_hybrid\_results':**

A list of matrices with powers

---

`power_studies_cont_nMC5_hybrid_results`

*power\_studies\_cont\_nMC5\_hybrid\_results*

---

**Description**

the results of the included power studies for continuous data without estimation using two-sample methods and nMC=5

**Usage**

`power_studies_cont_nMC5_hybrid_results`

**Format**

**'power\_studies\_cont\_nMC5\_hybrid\_results':**

A list of matrices with powers

---

`power_studies_cont_results`  
*power\_studies\_cont\_results*

---

**Description**

the results of the included power studies for continuous data without estimation

**Usage**

`power_studies_cont_results`

**Format**

**'power\_studies\_cont\_results':**  
A list of matrices with powers

---

`power_studies_disc_est_results`  
*power\_studies\_disc\_est\_results*

---

**Description**

the results of the included power studies for discrete data with estimation

**Usage**

`power_studies_disc_est_results`

**Format**

**'power\_studies\_disc\_est\_results':**  
A list of matrices with powers

---

`power_studies_disc_hybrid_results`  
*power\_studies\_disc\_hybrid\_results*

---

**Description**

the results of the included power studies for discrete data without estimation using two-sample methods

**Usage**

`power_studies_disc_hybrid_results`

**Format**

**'power\_studies\_disc\_hybrid\_results':**

A list of matrices with powers

---

`power_studies_disc_nMC5_hybrid_results`  
*power\_studies\_disc\_nMC5\_hybrid\_results*

---

**Description**

the results of the included power studies for discrete data without estimation using two-sample methods and nMC=5

**Usage**

`power_studies_disc_nMC5_hybrid_results`

**Format**

**'power\_studies\_disc\_nMC5\_hybrid\_results':**

A list of matrices with powers

---

power\_studies\_disc\_results  
*power\_studies\_disc\_results*

---

### **Description**

the results of the included power studies for discrete data without estimation

### **Usage**

power\_studies\_disc\_results

### **Format**

'power\_studies\_disc\_results':  
A list of matrices with powers

---

RipleyK *Ripley's K function test*

---

### **Description**

this function calculates the test statistic of Ripley's K function test

### **Usage**

RipleyK(x)

### **Arguments**

x                    matrix with data

### **Value**

a number (test statistic)

run.studies

*Benchmarking for Multivariate Goodness-of-fit Tests***Description**

This function runs the case studies included in the package.

**Usage**

```
run.studies(
  study,
  Continuous = TRUE,
  WithEstimation = FALSE,
  Dim = 2,
  TS,
  TSextra,
  With.p.value = FALSE,
  nsample = 250,
  nbins = c(5, 5),
  alpha = 0.05,
  param_alt,
  SuppressMessages = TRUE,
  B = 1000,
  maxProcessor
)
```

**Arguments**

study	either the name of the study, or its number in the list. If missing all the studies are run.
Continuous	=TRUE, run cases for continuous data.
WithEstimation	=FALSE, run case studies with or without parameter estimation?
Dim	=2 two or five-dimensional continuous data sets?
TS	routine to calculate new test statistics.
TSextra	list passed to TS (optional).
With.p.value	=FALSE, does user supplied routine return p values?
nsample	= 250, desired sample size. 250 is used in included case studies.
nbins	=c(5,5) number of bins for discretized data.
alpha	=0.05, type I error probability of tests. 0.05 is used in included case studies.
param_alt	(list of) values of parameter under the alternative hypothesis. If missing included values are used.
SuppressMessages	=TRUE, should informative messages be shown?
B	= 1000, number of simulation runs.
maxProcessor	number of cores to use. If missing the number of physical cores-1 is used. If set to 1 no parallel processing is done.



**Details**

For details consult vignette(package="MDgof")

**Value**

A (list of ) matrices of p.values.

**Examples**

```
#Examples are run with a super small B=25 simulation runs to satisfy CRAN submission rules.
#Run a new test for studies 1-3 for continuous data and without estimation.
#The new test is an (included) chi square test that finds it's own p value.
TSextra=list(Continuous=TRUE, WithEstimation=FALSE, Withpvalue=TRUE)
MDgof::run.studies(Continuous=TRUE, WithEstimation=FALSE,
                   study=1:3, TS=MDgof::newTS, TSextra=TSextra,
                   With.p.value = TRUE, B=25, maxProcessor = 1)
#Run included tests for studies 1-3 for discrete data and without estimation,
#but with type I error alpha=0.1
p=MDgof::power_studies_disc_results[[3]][1:3,,drop=FALSE]
MDgof::run.studies(Continuous=FALSE, WithEstimation=FALSE,
                   study=1:3, param_alt=p,alpha=0.1, B=25, maxProcessor = 1)
```

---

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

---

sq2rec	<i>Rearrange 2D discrete data</i>
--------	-----------------------------------

---

**Description**

This function changes a discrete data set given as a  $n \times m$  counting matrix to a  $nm \times 3$  matrix

**Usage**

```
sq2rec(x)
```

**Arguments**

x                    a matrix of discrete data.

**Value**

a rearranged matrix

---

timecheck	<i>estimate run time function</i>
-----------	-----------------------------------

---

**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	<i>Find test statistics for continuous data</i>
---------	---

---

**Description**

Find test statistics for continuous data

**Usage**

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

**Arguments**

x	A numeric matrix.
pnull	cdf.
param	parameters for pnull in case of parameter estimation.
TSextra	list with additional info

**Value**

A numeric vector with test statistics

---

TS_disc	<i>Find test statistics for discrete data</i>
---------	---

---

**Description**

Find test statistics for discrete data

**Usage**

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

**Arguments**

x	A numeric matrix.
pnull	cdf.
param	parameters for pnull in case of parameter estimation.
TSextra	list with additional info

**Value**

A numeric vector with test statistics

# Index

## \* datasets

- examples.mdgof.vignette, 12
- hybrid.mdgof.vignette, 21
- power\_studies\_cont\_D5\_results, 27
- power\_studies\_cont\_est\_results, 27
- power\_studies\_cont\_hybrid\_results, 28
- power\_studies\_cont\_nMC5\_hybrid\_results, 28
- power\_studies\_cont\_results, 29
- power\_studies\_disc\_est\_results, 29
- power\_studies\_disc\_hybrid\_results, 30
- power\_studies\_disc\_nMC5\_hybrid\_results, 30
- power\_studies\_disc\_results, 31

bakshaev\_rudzkis, 3

case.studies, 3

- case.studies.cont, 4
- case.studies.cont.D5, 5
- case.studies.disc, 5
- case.studies.est, 6

check.functions, 6

- chi\_cont\_test, 7
- chi\_disc\_test, 7
- chi\_power, 8

compute\_M\_for\_dtaset, 9

discretize, 10

draw\_case, 10

estimateEV, 11

examples.mdgof.vignette, 12

gauss\_kernel\_matrix, 12

- gen.cop, 13
- gen\_eval, 13
- gof\_power, 14
- gof\_test, 16
- gof\_test\_adjusted\_pvalue, 18
- grad\_mat, 20
- hybrid.mdgof.vignette, 21
- hybrid\_power, 21
- hybrid\_test, 23
- ksd, 24
- makeTSextra, 25
- newTS, 26
- p2dC, 26
- power\_studies\_cont\_D5\_results, 27
- power\_studies\_cont\_est\_results, 27
- power\_studies\_cont\_hybrid\_results, 28
- power\_studies\_cont\_nMC5\_hybrid\_results, 28
- power\_studies\_cont\_results, 29
- power\_studies\_disc\_est\_results, 29
- power\_studies\_disc\_hybrid\_results, 30
- power\_studies\_disc\_nMC5\_hybrid\_results, 30
- power\_studies\_disc\_results, 31
- RipleyK, 31
- run.studies, 32
- signif.digits, 33
- sq2rec, 34
- timecheck, 34
- TS\_cont, 35
- TS\_disc, 35