Package: kdecopula (via r-universe)

October 28, 2026

Type Package

Title Kernel Smoothing for Bivariate Copula Densities

Version 0.9.3

Description Provides fast implementations of kernel smoothing techniques for bivariate copula densities, in particular density estimation and resampling, see Nagler (2018) <doi:10.18637/jss.v084.i07>.

URL https://github.com/tnagler/kdecopula

BugReports https://github.com/tnagler/kdecopula/issues

Depends R (>= 3.0.0)

Imports lattice, locfit, qrng, Rcpp (>= 0.11.2), graphics, grDevices, stats, utils, quadprog

LinkingTo Rcpp, RcppArmadillo

Suggests R.rsp, VineCopula, testthat

VignetteBuilder R.rsp

License GPL-3

RoxygenNote 7.3.2

Encoding UTF-8

NeedsCompilation yes

Author Thomas Nagler [aut, cre], Kuangyu Wen [ctb]

Maintainer Thomas Nagler <mail@tnagler.com>

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

Date/Publication 2025-03-24 18:10:01 UTC

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

RemoteRef HEAD

RemoteSha 57bb1da0d004880f45693026cb2626a40161b6a8

Contents

bw_bern . . . 2
bw_beta . . . 3
bw_mr . . . 3
bw_t . . . 4
bw_tll . . . 5
bw_tll_nn . . . 5
bw_tt_pi . . . 6
dep_measures . . . 7
dkdecop . . . 8
fitted.kdecopula . . . 9
hkdecop . . . 10
kdecop . . . 11
kdecopula . . . 14
logLik.kdecopula . . . 15
plot.kdecopula . . . 16
predict.kdecopula . . . 17
simulate.kdecopula . . . 18
wdbc . . . 19

Index **21**

`bw_bern` *Bandwidth selection for the Bernstein copula estimator*

Description

The optimal size of knots is chosen by a rule of thumb adapted from Rose (2015).

Usage

```
bw_bern(udata)
```

Arguments

`udata`	data.

Details

The formula is

$$max(1, round(n^(1/3) * exp(abs(rho)^(1/n)) * (abs(rho) + 0.1))),$$

where ρ is the empirical Spearman's rho of the data.

Value

optimal order of the Bernstein polynomials.

bw_beta *Bandwidth selection for the beta kernel estimator*

Description

The bandwidth is selected by minimizing the MISE using the Frank copula as the reference family. The copula parameter is set by inversion of Kendall's tau. See Nagler (2014) for details.

Usage

```
bw_beta(udata)
```

Arguments

udata	data.

Details

To speed things up, optimal bandwidths have been pre-calculated on a grid of tau values.

Value

A scalar bandwidth parameter.

References

Nagler, T. (2014). Kernel Methods for Vine Copula Estimation. Master's Thesis, Technische Universitaet Muenchen, https://mediatum.ub.tum.de/node?id=1231221

bw_mr *Bandwidth selection for the mirror-reflection estimator*

Description

The bandwidth is selected by minimizing the MISE using the Frank copula as the reference family. The copula parameter is set by inversion of Kendall's tau. See Nagler (2014) for details.

Usage

```
bw_mr(udata)
```

Arguments

udata	data.

Details

To speed things up, optimal bandwidths have been pre-calculated on a grid of tau values.

Value

A scalar bandwidth parameter.

References

Nagler, T. (2014). Kernel Methods for Vine Copula Estimation. Master's Thesis, Technische Universitaet Muenchen, https://mediatum.ub.tum.de/node?id=1231221

`bw_t` *Bandwidth selection for the transformation kernel estimator*

Description

The bandwidth is selected by a rule of thumb. It approximately minimizes the MISE of the Gaussian copula on the transformed domain. The usual normal reference matrix is multiplied by 1.25 to account for the higher variance on the copula level.

Usage

```
bw_t(udata)
```

Arguments

`udata`	data.

Details

The formula is

$$1.25 n^{-1/6} \hat{\Sigma}^{1/2},$$

where $\hat{Sigma}$ is empirical covariance matrix of the transformed random vector.

Value

A `2 x 2` bandwidth matrix.

References

Nagler, T. (2014). Kernel Methods for Vine Copula Estimation. Master's Thesis, Technische Universitaet Muenchen, https://mediatum.ub.tum.de/node?id=1231221

bw_tll *Bandwidth selection for the transformation local likelihood estimator*

Description

The bandwidth is selected by a rule of thumb similar to bw_t.

Usage

```
bw_tll(udata, deg)
```

Arguments

udata	data.
deg	degree of the polynomial.

Details

The formula is

$$5n^{-1/(4q+2)}\hat{\Sigma}^{1/2},$$

where $\hat{Sigma}$ is empirical covariance matrix of the transformed random vector and $q = 1$ for TLL1 and $q = 2$ for TLL2.

Value

A 2 x 2 bandwidth matrix.

bw_tll_nn *Nearest-neighbor bandwidth selection for the transformation local likelihood estimator*

Description

The smoothing parameters is selected by the method of Geenens et al. (2017). It uses principal components for the rotation matrix and selects the nearest neighbor fraction along each principal direction by approximate least-squares cross-validation.

Usage

```
bw_tll_nn(udata, deg)
```

Arguments

udata	data.
deg	degree of the polynomial.

Value

A list with entries:

`B` rotation matrix,

`alpha` nearest neighbor fraction (this one is multiplied with `mult` in `kdecop()`),

`kappa` correction factor for differences in roughness along the axes,

see Geenens et al. (2017).

References

Geenens, G., Charpentier, A., and Paindaveine, D. (2017). Probit transformation for nonparametric kernel estimation of the copula density. Bernoulli, 23(3), 1848-1873.

`bw_tt_pi` *Nearest-neighbor bandwidth selection for the tapered transformation estimator*

Description

The smoothing parameters are selected by the method of Wen and Wu (2015).

Usage

```
bw_tt_pi(udata, rho.add = TRUE)

bw_tt_cv(udata, rho.add = T)
```

Arguments

`udata`	data.
`rho.add`	logical; whether a rotation (correlation) parameter shall be included.

Value

Optimal smoothing parameters as in Wen and Wu (2015): a numeric vector of length 4; entries are $(h, \rho, \theta_1, \theta_2)$.

Author(s)

Kuangyu Wen

References

Wen, K. and Wu, X. (2018). Transformation-Kernel Estimation of Copula Densities Journal of Business & Economic Statistics, 38(1), 148–164.

dep_measures *Dependence measures of a* kdecop() *fit*

Description

Calculates several dependence measures derived from the copula density. All measures except "blomqvist" are computed by quasi Monte Carlo methods (see rkdecop().

Usage

```
dep_measures(object, measures = "all", n_qmc = 10^3, seed = 5)
```

Arguments

object	an object of class kdecopula.
measures	which measures to compute, see *Details*.
n_qmc	the number of quasi Monte Carlo samples.
seed	the seed for quasi Monte Carlo integration.

Value

A named vector of dependence measures.

The following measures are available:

"kendall" Kendall's τ, see Nelsen (2007); computed as the sample version of a quasi Monte Carlo sample.

"spearman" Spearman's ρ, see Nelsen (2007); computed as the sample version of a quasi Monte Carlo sample.

"blomqvist" Blomqvist's β, see Nelsen (2007); computed as $4C(0.5, 0.5) - 1$.

"gini" Gini's γ, see Nelsen (2007); computed by quasi Monte Carlo integration.

"vd_waerden" van der Waerden's coefficient, see Genest and Verret (2005); computed as the sample version of a quasi Monte Carlo sample.

"minfo" mutual information, see Joe (1989); computed by quasi Monte Carlo integration.

"linfoot" Linfoot's correlation coefficient, see Joe (1989); computed by quasi Monte Carlo integration.

References

Nelsen, R. (2007). An introduction to copulas. Springer Science & Business Media, 2007.

Genest, C., and Verret, F. (2005). Locally most powerful rank tests of independence for copula models. Journal of Nonparametric Statistics, 17(5)

Joe, H. (1989). Relative Entropy Measures of Multivariate Dependence. Journal of the American Statistical Association, 84(405)

Examples

```
## load data and transform with empirical cdf
data(wdbc)
udat <- apply(wdbc[, -1], 2, function(x) rank(x) / (length(x) + 1))

## estimate copula density and calculate dependence measures
fit <- kdecop(udat[, 5:6])
dep_measures(fit)
```

---

dkdecop *Working with* kdecopula *objects*

Description

The function kdecop() stores it's result in object of class kdecopula. The density estimate can be evaluated on arbitrary points with dkdecop(); the cdf with pkdecop(). Furthermore, synthetic data can be simulated with rkdecop().

Usage

```
dkdecop(u, obj, stable = FALSE)

pkdecop(u, obj)

rkdecop(n, obj, quasi = FALSE)
```

Arguments

u	mx2 matrix of evaluation points.
obj	kdecopula object.
stable	logical; option for stabilizing the estimator: the estimated density is cut off at 50.
n	integer; number of observations.
quasi	logical; the default (FALSE) returns pseudo-random numbers, use TRUE for quasi-random numbers (generalized Halton, see qrng::ghalton()).

Value

A numeric vector of the density/cdf or a n x 2 matrix of simulated data.

Author(s)

Thomas Nagler

References

#' Nagler, T. (2018) kdecopula: An R Package for the Kernel Estimation of Bivariate Copula Densities. Journal of Statistical Software 84(7), 1-22

#' Geenens, G., Charpentier, A., and Paindaveine, D. (2017). Probit transformation for nonparametric kernel estimation of the copula density. Bernoulli, 23(3), 1848-1873.

Nagler, T. (2014). Kernel Methods for Vine Copula Estimation. Master's Thesis, Technische Universitaet Muenchen, https://mediatum.ub.tum.de/node?id=1231221

Cambou, T., Hofert, M., Lemieux, C. (2015). A primer on quasi-random numbers for copula models, arXiv:1508.03483

See Also

kdecop, plot.kdecopula, ghalton

Examples

```
## load data and transform with empirical cdf
data(wdbc)
udat <- apply(wdbc[, -1], 2, function(x) rank(x) / (length(x) + 1))

## estimation of copula density of variables 5 and 6
fit <- kdecop(udat[, 5:6])
plot(fit)

## evaluate density estimate at (u1,u2)=(0.123,0.321)
dkdecop(c(0.123, 0.321), fit)

## evaluate cdf estimate at (u1,u2)=(0.123,0.321)
pkdecop(c(0.123, 0.321), fit)

## simulate 500 samples from density estimate
plot(rkdecop(500, fit))
```

---

fitted.kdecopula *Extract fitted values from a* kdecop() *fits.*

Description

Simply calls predict(object, object$udata, what).

Usage

```
## S3 method for class 'kdecopula'
fitted(object, what = "pdf", ...)
```

Arguments

object	an object of class kdecopula.
what	what to predict, one of c("pdf", "cdf", "hfunc1", "hfunc2", "hinv1", "hinv2").
...	unused.

See Also

predict.kdecopula()

Examples

```
data(wdbc)
udat <- apply(wdbc[, -1], 2, function(x) rank(x) / (length(x) + 1))
fit <- kdecop(udat[, 5:6])

all.equal(fitted(fit), predict(fit, fit$udata))
```

---

hkdecop *H-function and inverse of a* kdecop() *fit*

Description

Evaluates the h-function (or its inverse) corresponding to a kdecopula object. H-functions are conditional distribution functions obtained by integrating the copula density w.r.t. to one of its arguments (see also VineCopula::BiCopHfunc().

Usage

```
hkdecop(u, obj, cond.var, inverse = FALSE)
```

Arguments

u	$nx2$ matrix of evaluation points.
obj	kdecopula object.
cond.var	integer; cond.var = 1 conditions on the first variable, cond.var = 2 on the second.
inverse	logical; indicates whether the h-function or its inverse shall be calculated.

Value

A length n vector of the (inverse) h-function evaluated at u.

Author(s)

Thomas Nagler

Examples

```
## load data and transform with empirical cdf
data(wdbc)
udat <- apply(wdbc[, -1], 2, function(x) rank(x) / (length(x) + 1))

## estimation of copula density of variables 5 and 6
fit <- kdecop(udat[, 5:6])
plot(fit)

## evaluate h-function estimate and its inverse at (u1|u2) = (0.123 | 0.321)
hkdecop(c(0.123, 0.321), fit, cond.var = 2)
hkdecop(c(0.123, 0.321), fit, cond.var = 2, inverse = TRUE)
```

---

kdecop *Bivariate kernel copula density estimation*

Description

Based on samples from a bivariate copula, the copula density is estimated. The user can choose between different methods. If no bandwidth is provided by the user, it will be set by a method-specific automatic selection procedure. The related (d/p/r)kdecop functions evaluate the density and cdf or simulate synthetic data, respectively.

Usage

```
kdecop(
  udata,
  bw = NA,
  mult = 1,
  method = "TLL2nn",
  knots = 30,
  renorm.iter = 3L,
  info = TRUE
)
```

Arguments

udata	nx2 matrix of copula data.
bw	bandwidth specification; if NA, bw is selected automatically (see Details); Otherwise, please provide "T", "TLL1", "TLL2": a $2x2$ bandwidth matrix, "TLL1nn", "TLL2nn": a list with (named) entries B, alpha, and kappa, "TTCV", "TTPI": a numeric vector of length four containing $(h, \rho, \theta_1, \theta_2)$, c.f. Wen and Wu (2015), "MR", "beta": a positive real number.

`mult`	bandwidth multiplier, has to be positive; useful for making estimates more/less smooth manually.
`method`	`"T"`: transformation estimator based on classical bivariate kernel estimation (e.g., Geenenens et al., 2014), `"TLL1"`: transformation estimator with log-linear local likelihood estimation (Geenenens et al., 2014), `"TLL2"`: transformation estimator with log-quadratic local likelihood estimation (Geenenens et al., 2014), `"TLL1nn"`: transformation estimator with log-linear local likelihood estimation and nearest-neighbor bandwidths (Geenenens et al., 2014), `"TLL2nn"`: transformation estimator with log-quadratic local likelihood estimation and nearest-neighbor bandwidths (Geenenens et al., 2014), `"TTPI"`: tapered transformation estimator with plug-in bandwidths (Wu and Wen, 2015), `"TTCV"`: tapered transformation estimator with profile cross-validation bandwidths (Wu and Wen, 2015), `"MR"`: mirror-reflection estimator (Gijbels and Mielniczuk, 1990), `"beta"`: beta kernel estimator (Charpentier et al., 2006), `"bern"`: Bernstein copula estimator (Sanchetta and Satchell, 2004); the coefficients are adjusted by the method of Weiss and Scheffer (2012).
`knots`	integer; number of knots in each dimension for the spline approximation.
`renorm.iter`	integer; number of iterations for the renormalization procedure (see *Details*).
`info`	logical; if `TRUE`, additional information about the estimate will be gathered (see *Value*).

Details

We use a Gaussian product kernel function for all methods except the beta kernel and Bernstein estimators. For details on bandwidth selection for a specific method, see: `bw_t`, `bw_tll`, `bw_tll_nn`, `bw_tt_pi`, `bw_tt_cv`, `bw_mr`, `bw_beta`, `bw_bern`.

Kernel estimates are usually no proper copula densities. In particular, the estimated marginal densities are not uniform. We mitigate this issue by a renormalization procedure. The number of iterations of the renormalization algorithm can be specified with the `renorm.iter` argument. Typically, a very small number of iterations is sufficient.

Value

The function `kdecop` returns an object of class `kdecopula` that contains all information necessary for evaluation of the estimator. If no bandwidth was provided in the function call, the automatically selected value can be found in the variable `object$bw`. If `info=TRUE`, also the following will be available under `object$info`:

`likvalues`	Estimator evaluated in sample points
`loglik`	Log likelihood
`effp`	Effective number of parameters

AIC	Akaike information criterion
cAIC	Bias-corrected version of Akaike information criterion
BIC	Bayesian information criterion.

The density estimate can be evaluated on arbitrary points with dkdecop; the cdf with pkdecop. Furthermore, synthetic data can be simulated with rkdecop, and several plotting options are available with plot and contour.

Note

The implementation of the tapered transformation estimator ("TTPI"/"TTCV") was kindly provided by Kuangyu Wen.

Author(s)

Thomas Nagler

References

Nagler, T. (2018) kdecopula: An R Package for the Kernel Estimation of Bivariate Copula Densities. Journal of Statistical Software 84(7), 1-22

Geenens, G., Charpentier, A., and Paindaveine, D. (2017). Probit transformation for nonparametric kernel estimation of the copula density. Bernoulli, 23(3), 1848-1873.

Wen, K. and Wu, X. (2018). Transformation-Kernel Estimation of Copula Densities Journal of Business & Economic Statistics, 38(1), 148–164.

Gijbels, I. and Mielniczuk, J. (1990). Estimating the density of a copula function. Communications in Statistics - Theory and Methods, 19(2):445-464.

Charpentier, A., Fermanian, J.-D., and Scaillet, O. (2006). The estimation of copulas: Theory and practice. In Rank, J., editor, Copulas: From theory to application in finance. Risk Books.

Weiss, G. and Scheffer, M. (2012). Smooth Nonparametric Bernstein Vine Copulas. arXiv:1210.2043

Nagler, T. (2014). Kernel Methods for Vine Copula Estimation. Master's Thesis, Technische Universitaet Muenchen, https://mediatum.ub.tum.de/node?id=1231221

See Also

kdecopula, plot.kdecopula, predict.kdecopula, fitted.kdecopula, simulate.kdecopula, dkdecop, pkdecop, rkdecop

Examples

```
## load data and transform with empirical cdf
data(wdbc)
udat <- apply(wdbc[, -1], 2, function(x) rank(x) / (length(x) + 1))
```

```
## estimation of copula density of variables 5 and 6
fit <- kdecop(udat[, 5:6])
summary(fit)
plot(fit)
contour(fit)

## evaluate density estimate at (u1,u2)=(0.123,0.321)
dkdecop(c(0.123, 0.321), fit)

## evaluate cdf estimate at (u1,u2)=(0.123,0.321)
pkdecop(c(0.123, 0.321), fit)

## simulate 500 samples from density estimate
plot(rkdecop(500, fit))  # pseudo-random
plot(rkdecop(500, fit, quasi = TRUE))  # quasi-random
```

---

kdecopula *Kernel Smoothing for Bivariate Copula Densities*

Description

This package provides fast implementations of kernel estimators for the copula density. Due to its several plotting options it is particularly useful for the exploratory analysis of dependence structures. It can be further used for flexible nonparametric estimation of copula densities and resampling.

Details

The function kdecop can be used to estimate a copula density with a number of popular kernel estimators. The density estimate can be evaluated on arbitrary points with dkdecop; the cdf with pkdecop. Furthermore, synthetic data can be simulated with rkdecop, and several plot options are provided by plot.kdecopula.

Author(s)

Thomas Nagler

References

Nagler, T. (2018) kdecopula: An R Package for the Kernel Estimation of Bivariate Copula Densities. Journal of Statistical Software 84(7), 1-22

Gijbels, I. and Mielniczuk, J. (1990). Estimating the density of a copula function. Communications in Statistics - Theory and Methods, 19(2):445-464.

Charpentier, A., Fermanian, J.-D., and Scaillet, O. (2006). The estimation of copulas: Theory and practice. In Rank, J., editor, Copulas: From theory to application in finance. Risk Books.

Geenens, G., Charpentier, A., and Paindaveine, D. (2017). Probit transformation for nonparametric kernel estimation of the copula density. Bernoulli, 23(3), 1848-1873.

Nagler, T. (2014). Kernel Methods for Vine Copula Estimation. Master's Thesis, Technische Universitaet Muenchen, https://mediatum.ub.tum.de/node?id=1231221

Wen, K. and Wu, X. (2018). Transformation-Kernel Estimation of Copula Densities Journal of Business & Economic Statistics, 38(1), 148–164.

See Also

Useful links:

- https://github.com/tnagler/kdecopula
- Report bugs at https://github.com/tnagler/kdecopula/issues

Examples

```
## load data and transform with empirical cdf
data(wdbc)
udat <- apply(wdbc[, -1], 2, function(x) rank(x)/(length(x)+1))

## estimation of copula density of variables 5 and 6
dens.est <- kdecop(udat[, 5:6])
plot(dens.est)

## evaluate density estimate at (u1,u2)=(0.123,0.321)
dkdecop(c(0.123, 0.321), dens.est)

## evaluate cdf estimate at (u1,u2)=(0.123,0.321)
pkdecop(c(0.123, 0.321), dens.est)

## simulate 500 samples from density estimate
rkdecop(500, dens.est)
```

---

logLik.kdecopula *Log-Likelihood of a* kdecopula *object*

Description

Log-Likelihood of a kdecopula object

Usage

```
## S3 method for class 'kdecopula'
logLik(object, ...)
```

Arguments

object	an object of class kdecopula.
...	not used.

Value

Returns an object of class logLik containing the log- likelihood, number of observations and effective number of parameters ("df").

Author(s)

Thomas Nagler

See Also

logLik, AIC, BIC

Examples

```
## load data and transform with empirical cdf
data(wdbc)
udat <- apply(wdbc[, -1], 2, function(x) rank(x) / (length(x) + 1))

## estimation of copula density of variables 5 and 6
fit <- kdecop(udat[, 5:6])

## compute fit statistics
logLik(fit)
AIC(fit)
BIC(fit)
```

---

plot.kdecopula *Plotting* kdecopula *objects*

Description

Produces perspective or contour plots for a kdecopula object.

Usage

```
## S3 method for class 'kdecopula'
plot(x, type = "surface", margins, size, ...)

## S3 method for class 'kdecopula'
contour(x, margins = "norm", size = 100L, ...)
```

Arguments

x	kdecopula object.
type	plot type; either "contour" or "surface" (partial matching is activated) for a contour or perspective/surface plot respectively.
margins	"unif" for the original copula density, "norm" for the transformed density with standard normal margins, "exp" with standard exponential margins, and "flexp" with flipped exponential margins. Default is "norm" for type = "contour", and "unif" for type = "surface".
size	integer; the plot is based on values on a $sizexsize$ grid; default is 100 for type = "contour", and 25 for type = "surface".
...	optional arguments passed to contour or wireframe.

Author(s)

Thomas Nagler

References

Nagler, T. (2018) kdecopula: An R Package for the Kernel Estimation of Bivariate Copula Densities. Journal of Statistical Software 84(7), 1-22

See Also

kdecop, contour, wireframe

Examples

```
## load data and transform with empirical cdf
data(wdbc)
udat <- apply(wdbc[, -1], 2, function(x) rank(x)/(length(x)+1))

## estimation of copula density of variables 5 and 6
obj <- kdecop(udat[, 5:6])

## plots
plot(obj)  # surface plot of copula density
contour(obj)  # contour plot with standard normal margins
contour(obj, margins = "unif")  # contour plot of copula density
```

---

predict.kdecopula *Prediction method for* kdecop() *fits*

Description

Predicts the pdf, cdf, or (inverse) h-functions by calling dkdecop(), pkdecop(), or hkdecop().

Usage

```
## S3 method for class 'kdecopula'
predict(object, newdata, what = "pdf", stable = FALSE, ...)
```

Arguments

object	an object of class kdecopula.
newdata	evaluation point(s), a length two vector or a matrix with two columns.
what	what to predict, one of c("pdf", "cdf", "hfunc1", "hfunc2", "hinv1", "hinv2").
stable	only used for what = "pdf", see dkdecop().
...	unused.

Value

A numeric vector of predictions.

Examples

```
data(wdbc)
udat <- apply(wdbc[, -1], 2, function(x) rank(x) / (length(x) + 1))
fit <- kdecop(udat[, 5:6])

all.equal(predict(fit, c(0.1, 0.2)), dkdecop(c(0.1, 0.2), fit))
all.equal(predict(fit, udat, "hfunc1"), hkdecop(udat, fit, cond.var = 1))
```

---

simulate.kdecopula *Simulate data from a* kdecop() *fit.*

Description

See rkdecop().

Usage

```
## S3 method for class 'kdecopula'
simulate(object, nsim = 1, seed = NULL, quasi = FALSE, ...)
```

Arguments

object	an object of class kdecopula.
nsim	integer; number of observations.
seed	integer; set.seed(seed) will be called prior to rkdecop().
quasi	logical; the default (FALSE) returns pseudo-random numbers, use TRUE for quasi-random numbers (generalized Halton, see
...	unused.

Value

Simulated data from the fitted `kdecopula` model.

Examples

```
data(wdbc)
udat <- apply(wdbc[, -1], 2, function(x) rank(x) / (length(x) + 1))
fit <- kdecop(udat[, 5:6])
plot(simulate(fit, 500))
```

---

wdbc *Wisconsin Diagnostic Breast Cancer (WDBC)*

Description

The data contain measurements on cells in suspicious lumps in a woman's breast. Features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. They describe characteristics of the cell nuclei present in the image. All samples are classified as either *benign* or *malignant*.

Format

`wdbc` is a `data.frame` with 31 columns. The first column indicates whether the sample is classified as benign (`B`) or malignant (`M`). The remaining columns contain measurements for 30 features.

Details

Ten real-valued features are computed for each cell nucleus:

a) radius (mean of distances from center to points on the perimeter)
b) texture (standard deviation of gray-scale values)
c) perimeter
d) area
e) smoothness (local variation in radius lengths)
f) compactness (perimeter^2 / area - 1.0)
g) concavity (severity of concave portions of the contour)
h) concave points (number of concave portions of the contour)
i) symmetry
j) fractal dimension ("coastline approximation" - 1)

The references listed below contain detailed descriptions of how these features are computed.

The mean, standard error, and "worst" or largest (mean of the three largest values) of these features were computed for each image, resulting in 30 features.

Note

This breast cancer database was obtained from the University of Wisconsin Hospitals, Madison from Dr. William H. Wolberg.

Source

Bache, K. & Lichman, M. (2013). UCI Machine Learning Repository. Irvine, CA: University of California, School of Information and Computer Science.

References

O. L. Mangasarian and W. H. Wolberg: "Cancer diagnosis via linear programming",
SIAM News, Volume 23, Number 5, September 1990, pp 1 & 18.

William H. Wolberg and O.L. Mangasarian: "Multisurface method of pattern separation for medical diagnosis applied to breast cytology",
Proceedings of the National Academy of Sciences, U.S.A., Volume 87, December 1990, pp 9193-9196.

K. P. Bennett & O. L. Mangasarian: "Robust linear programming discrimination of two linearly inseparable sets",
Optimization Methods and Software 1, 1992, 23-34 (Gordon & Breach Science Publishers).

Examples

```
data(wdbc)
str(wdbc)
```

Index

∗ **datasets**
wdbc, 19
∗ **package**
kdecopula, 14
∗ **plot**
plot.kdecopula, 16

AIC, *16*

BIC, *16*
bw_bern, 2, *12*
bw_beta, 3, *12*
bw_mr, 3, *12*
bw_t, 4, *5*, *12*
bw_tll, 5, *12*
bw_tll_nn, 5, *12*
bw_tt_cv, *12*
bw_tt_cv *(*bw_tt_pi*)*, 6
bw_tt_pi, 6, *12*

contour, *13*, *17*
contour.kdecopula *(*plot.kdecopula*)*, 16

dep_measures, 7
dkdecop, 8, *13, 14*
dkdecop(), *8*

fitted.kdecopula, 9, *13*

ghalton, *9*

hkdecop, 10

kdecop, *9*, 11, *12*, *14*, *17*
kdecop(), *6*, *8*
kdecopula, *13*, 14
kdecopula-package *(*kdecopula*)*, 14

logLik, *16*
logLik.kdecopula, 15

pkdecop, *13, 14*
pkdecop *(*dkdecop*)*, 8
pkdecop(), *8*
plot, *13*
plot.kdecopula, *9*, *13, 14*, 16
predict.kdecopula, *13*, 17

qrng::ghalton(), *8*

rkdecop, *13, 14*
rkdecop *(*dkdecop*)*, 8
rkdecop(), *7, 8*

simulate.kdecopula, *13*, 18

VineCopula::BiCopHfunc(), *10*

wdbc, 19
wireframe, *17*