Package ‘kdevine’

June 13, 2024

Type Package

Title Multivariate Kernel Density Estimation with Vine Copulas

Version 0.4.5

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

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

Description Implements the vine copula based kernel density estimator of Nagler and Czado (2016) <doi:10.1016/j.jmva.2016.07.003>. The estimator does not suffer from the curse of dimensionality and is therefore well suited for high-dimensional applications.

License GPL-3

Imports graphics, stats, utils, MASS, Rcpp, qrng, KernSmooth, cctools, kdecopula (>= 0.8.1), VineCopula, doParallel, parallel, foreach

LazyData yes

LinkingTo Rcpp

RoxygenNote 7.3.1

Suggests testthat

Encoding UTF-8

NeedsCompilation yes

Author Thomas Nagler [aut, cre]

Maintainer Thomas Nagler <mail@tnagler.com>

Repository CRAN

Date/Publication 2024-06-13 21:40:02 UTC

Contents

`kdevine-package` *Kernel Smoothing for Bivariate Copula Densities*

Description

This package implements a vine copula based kernel density estimator. The estimator does not suffer from the curse of dimensionality and is therefore well suited for high-dimensional applications (see, Nagler and Czado, 2016).

Details

The multivariate kernel density estimators is implemented by the `kdevine` function. It combines a kernel density estimator for the margins (`kde1d`) and a kernel estimator of the vine copula density (`kdevinecop`). The package is built on top of the copula density estimators in the kdecopula::kdecopula-package and let's you choose from all its implemented methods. Optionally, the vine copula can be estimated parameterically (only the margins are nonparametric).

Author(s)

Thomas Nagler

References

Nagler, T., Czado, C. (2016)
Evading the curse of dimensionality in nonparametric density estimation with simplified vine copulas.
Journal of Multivariate Analysis 151, 69-89 (doi:10.1016/j.jmva.2016.07.003)

Nagler, T., Schellhase, C. and Czado, C. (2017)
Nonparametric estimation of simplified vine copula models: comparison of methods arXiv:1701.00845

Nagler, T. (2017)
A generic approach to nonparametric function estimation with mixed data.
arXiv:1704.07457

See Also

Useful links:

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

---

contour.kdevinecop *Contour plots of pair copula kernel estimates*

Description

Contour plots of pair copula kernel estimates

Usage

```
## S3 method for class 'kdevinecop'
contour(x, tree = "ALL", xylim = NULL, cex.nums = 1, ...)
```

Arguments

`x`	a `kdevinecop` object.
`tree`	`"ALL"` or integer vector; specifies which trees are plotted.
`xylim`	numeric vector of length 2; sets `xlim` and `ylim` for the contours.
`cex.nums`	numeric; expansion factor for font of the numbers.
`...`	arguments passed to `contour.kdecopula`.

Examples

```
data(wdbc, package = "kdecopula")                      # load data
u <- VineCopula::pobs(wdbc[, 5:7], ties = "average")  # rank-transform

# estimate density
fit <- kdevinecop(u)

# contour matrix
contour(fit)
```

dkde1d *Working with a kde1d object*

Description

The density, cdf, or quantile function of a kernel density estimate are evaluated at arbitrary points with dkde1d, pkde1d, and qkde1d respectively.

Usage

```
dkde1d(x, obj)

pkde1d(x, obj)

qkde1d(x, obj)

rkde1d(n, obj, quasi = FALSE)
```

Arguments

x	vector of evaluation points.
obj	a kde1d object.
n	integer; number of observations.
quasi	logical; the default (FALSE) returns pseudo-random numbers, use TRUE for quasi-random numbers (generalized Halton, see ghalton).

Value

The density or cdf estimate evaluated at x.

See Also

kde1d

Examples

```
data(wdbc)  # load data
fit <- kde1d(wdbc[, 5])  # estimate density
dkde1d(1000, fit)        # evaluate density estimate
pkde1d(1000, fit)        # evaluate corresponding cdf
qkde1d(0.5, fit)         # quantile function
hist(rkde1d(100, fit))   # simulate
```

dkdevine — *Evaluate the density of a kdevine object*

Description

Evaluate the density of a kdevine object

Usage

```
dkdevine(x, obj)
```

Arguments

x	(mxd) matrix of evaluation points (or vector of length d).
obj	a `kdevine` object.

Value

The density estimate evaluated at `x`.

See Also

`kdevine`

Examples

```
# load data
data(wdbc)

# estimate density (use xmin to indicate positive support)
fit <- kdevine(wdbc[, 5:7], xmin = rep(0, 3))

# evaluate density estimate
dkdevine(c(1000, 0.1, 0.1), fit)
```

dkdevinecop — *Working with a* `kdevinecop` *object*

Description

A vine copula density estimate (stored in a `kdevinecop` object) can be evaluated on arbitrary points with `dkevinecop`. Furthermore, you can simulate from the estimated density with `rkdevinecop`.

Usage

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

rkdevinecop(n, obj, U = NULL, quasi = FALSE)
```

Arguments

u	$mx2$ matrix of evaluation points.
obj	kdevinecop object.
stable	logical; option for stabilizing the estimator: the estimated pair copula density is cut off at 50.
n	integer; number of observations.
U	(optional) nxd matrix of independent uniform random variables.
quasi	logical; the default (FALSE) returns pseudo-random numbers, use TRUE for quasi-random numbers (generalized Halton, see ghalton).

Value

A numeric vector of the density/cdf or a $nx2$ matrix of simulated data.

Author(s)

Thomas Nagler

References

Nagler, T., Czado, C. (2016)
Evading the curse of dimensionality in nonparametric density estimation.
Journal of Multivariate Analysis 151, 69-89 (doi:10.1016/j.jmva.2016.07.003)

Dissmann, J., Brechmann, E. C., Czado, C., and Kurowicka, D. (2013).
Selecting and estimating regular vine copulae and application to financial returns.
Computational Statistics & Data Analysis, 59(0):52–69.

See Also

kdevinecop, dkdecop, rkdecop, ghalton

Examples

```
data(wdbc, package = "kdecopula")                    # load data
u <- VineCopula::pobs(wdbc[, 5:7], ties = "average") # rank-transform

fit <- kdevinecop(u)                 # estimate density
dkdevinecop(c(0.1, 0.1, 0.1), fit)  # evaluate density estimate
```

kde1d — *Univariate kernel density estimation for bounded and unbounded support*

Description

Discrete variables are convoluted with the uniform distribution (see, Nagler, 2017). If a variable should be treated as discrete, declare it as `ordered()`.

Usage

```
kde1d(x, mult = 1, xmin = -Inf, xmax = Inf, bw = NULL, bw_min = 0, ...)
```

Arguments

`x`	vector of length n.
`mult`	numeric; the actual bandwidth used is $bw * mult$.
`xmin`	lower bound for the support of the density.
`xmax`	upper bound for the support of the density.
`bw`	bandwidth parameter; has to be a positive number or `NULL`; the latter calls `KernSmooth::dpik()`.
`bw_min`	minimum value for the bandwidth.
`...`	unused.

Details

If `xmin` or `xmax` are finite, the density estimate will be 0 outside of $[xmin, xmax]$. Mirror-reflection is used to correct for boundary bias. Discrete variables are convoluted with the uniform distribution (see, Nagler, 2017).

Value

An object of class `kde1d`.

References

Nagler, T. (2017). *A generic approach to nonparametric function estimation with mixed data.* arXiv:1704.07457

See Also

`dkde1d`, `pkde1d`, `qkde1d`, `rkde1d` `plot.kde1d` , `lines.kde1d`

Examples

```
data(wdbc, package = "kdecopula")  # load data
fit <- kde1d(wdbc[, 5])            # estimate density
dkde1d(1000, fit)                  # evaluate density estimate
```

`kdevine` *Kernel density estimator based on simplified vine copulas*

Description

Implements the vine-copula based estimator of Nagler and Czado (2016). The marginal densities are estimated by `kde1d`, the vine copula density by `kdevinecop`. Discrete variables are convoluted with the uniform distribution (see, Nagler, 2017). If a variable should be treated as discrete, declare it as `ordered()`. Factors are expanded into binary dummy codes.

Usage

```
kdevine(x, mult_1d = NULL, xmin = NULL, xmax = NULL, copula.type = "kde", ...)
```

Arguments

`x`	(nxd) data matrix.
`mult_1d`	numeric; all bandwidhts for marginal kernel density estimation are multiplied with `mult_1d`. Defaults to `log(1 + d)` where `d` is the number of variables after applying `cctools::expand_as_numeric()`.
`xmin`	numeric vector of length d; see `kde1d`.
`xmax`	numeric vector of length d; see `kde1d`.
`copula.type`	either `"kde"` (default) or `"parametric"` for kernel or parametric estimation of the vine copula.
`...`	further arguments passed to `kde1d` or `kdevinecop`.

Value

An object of class `kdevine`.

References

Nagler, T., Czado, C. (2016) *Evading the curse of dimensionality in nonparametric density estimation with simplified vine copulas.* Journal of Multivariate Analysis 151, 69-89 (doi:10.1016/j.jmva.2016.07.003)

Nagler, T. (2017). *A generic approach to nonparametric function estimation with mixed data.* arXiv:1704.07457

See Also

`dkdevine` `kde1d` `kdevinecop`

Examples

```
# load data
data(wdbc, package = "kdecopula")

# estimate density (use xmin to indicate positive support)
fit <- kdevine(wdbc[, 5:7], xmin = rep(0, 3))

# evaluate density estimate
dkdevine(c(1000, 0.1, 0.1), fit)

# plot simulated data
pairs(rkdevine(nrow(wdbc), fit))
```

---

kdevinecop — *Kernel estimation of vine copula densities*

Description

The function estimates a vine copula density using kernel estimators for the pair copulas (based on the kdecopula package).

Usage

```
kdevinecop(
  data,
  matrix = NA,
  method = "TLL2",
  renorm.iter = 3L,
  mult = 1,
  test.level = NA,
  trunc.level = NA,
  treecrit = "tau",
  cores = 1,
  info = FALSE
)
```

Arguments

data	(nxd) matrix of copula data (have to lie in $[0, 1^d]$).
matrix	R-Vine matrix (nxd) specifying the structure of the vine; if NA (default) the structure selection heuristic of Dissman et al. (2013) is applied.
method	see kdecop.
renorm.iter	see kdecop.
mult	see kdecop.

test.level	significance level for independence test. If you provide a number in $[0, 1]$, an independence test (BiCopIndTest) will be performed for each pair; if the null hypothesis of independence cannot be rejected, the independence copula will be set for this pair. If test.level = NA (default), no independence test will be performed.
trunc.level	integer; the truncation level. All pair copulas in trees above the truncation level will be set to independence.
treecrit	criterion for structure selection; defaults to "tau".
cores	integer; if cores > 1, estimation will be parallized within each tree (using foreach).
info	logical; if TRUE, additional information about the estimate will be gathered (see kdecop).

Value

An object of class kdevinecop. That is, a list containing

T1, T2, ...	lists of the estimted pair copulas in each tree,
matrix	the structure matrix of the vine,
info	additional information about the fit (if info = TRUE).

References

Nagler, T., Czado, C. (2016)
Evading the curse of dimensionality in nonparametric density estimation with simplified vine copulas.
Journal of Multivariate Analysis 151, 69-89 (doi:10.1016/j.jmva.2016.07.003)

Nagler, T., Schellhase, C. and Czado, C. (2017)
Nonparametric estimation of simplified vine copula models: comparison of methods arXiv:1701.00845

Dissmann, J., Brechmann, E. C., Czado, C., and Kurowicka, D. (2013).
Selecting and estimating regular vine copulae and application to financial returns.
Computational Statistics & Data Analysis, 59(0):52–69.

See Also

dkdevinecop, kdecop, BiCopIndTest, foreach

Examples

```
data(wdbc, package = "kdecopula")
# rank-transform to copula data (margins are uniform)
u <- VineCopula::pobs(wdbc[, 5:7], ties = "average")

fit <- kdevinecop(u)                    # estimate density
dkdevinecop(c(0.1, 0.1, 0.1), fit)      # evaluate density estimate
contour(fit)                            # contour matrix (Gaussian scale)
pairs(rkdevinecop(500, fit))            # plot simulated data
```

plot.kde1d — *Plotting kde1d objects*

Description

Plotting kde1d objects

Usage

```
## S3 method for class 'kde1d'
plot(x, ...)

## S3 method for class 'kde1d'
lines(x, ...)
```

Arguments

`x`	`kde1d` object.
`...`	further arguments passed to `plot.default`.

See Also

`kde1d` `lines.kde1d`

Examples

```
data(wdbc)  # load data
fit <- kde1d(wdbc[, 7])  # estimate density
plot(fit)  # plot density estimate

fit2 <- kde1d(as.ordered(wdbc[, 1])) # discrete variable
plot(fit2, col = 2)
```

rkdevine — *Simulate from a kdevine object*

Description

Simulate from a kdevine object

Usage

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

Arguments

n	number of observations.
obj	a kdevine object.
quasi	logical; the default (FALSE) returns pseudo-random numbers, use TRUE for quasi-random numbers (generalized Halton, only works for fully nonparametric fits).

Value

An nxd matrix of simulated data from the kdevine object.

See Also

kdevine, rkdevinecop, rkde1d

Examples

```
# load and plot data
data(wdbc)

# estimate density
fit <- kdevine(wdbc[, 5:7], xmin = rep(0, 3))

# plot simulated data
pairs(rkdevine(nrow(wdbc), fit))
```

---

wdbc *Wisconsin Diagnostic Breast Cancer (WDBC)*

Description

The data contain measurements on cells in suspicious lumps in a women'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 classsified as either *benign* or *malignant*.

Usage

```
data(wdbc)
```

Format

wdbc is a data.frame with 31 columns. The first column indicates wether 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

https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic)

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