Package 'probhat'

July 14, 2020

Title Multivariate Generalized Kernel Smoothing and Related Statistical Methods

Version 0.3.1

Date 2020-07-14

License GPL (>= 2)

Maintainer Abby Spurdle <spurdle.a@gmail.com>

Author Abby Spurdle

URL https://sites.google.com/site/spurdlea/r

Description Mass functions, density functions, distribution functions and quantile functions via continuous kernel smoothing, and to a lesser extent, discrete kernel smoothing. Also, supports categorical distributions and smooth empirical-like distributions. There are univariate, multivariate and conditional distributions, including multivariate-conditional distributions and conditional distributions with mixed input types, along with functions for plotting univariate, bivariate and trivariate distributions. Conditional categorical distributions with mixed input types can be used for statistical classification purposes. And there are extensions for computing multivariate probabilities, multivariate random numbers, moment-based statistics, robust-based statistics and mode estimates.

Imports intoo, barsurf, kubik

Suggests vectools, bivariate, fclust, scatterplot3d

NeedsCompilation no

Repository CRAN

Date/Publication 2020-07-14 07:30:03 UTC

R topics documented:

`11_discrete_kernels` *Discrete Kernels*

Description

Discrete kernels (without standardized intervals), for discrete kernel smoothing.

NOTE THAT THESE MAY BE REMOVED.

Usage

```
binomial.dkernel (bw)
```

Arguments

`bw`	Integer, the bandwidth. This should be a positive odd number. (Even numbers are rounded up).

Details

NOTE THAT THESE MAY BE REMOVED.

This is a constructor for a binomial kernel object.

Currently, this object is a named list (with pmf and cdf components).

Here, the PMF is symmetric about zero (in contrast to how a binomial distribution is normally defined), and has zero-probability outside the interval [-hbw, hbw], where:
hbw = (bw - 1) / 2

This object is used inside discrete kernel smoothing.
(In which case, you provide the constructor, not the object).

Value

A dkernel object.

References

Refer to the vignette for an overview, references and better examples.

See Also

Continuous Kernels

plot.dkernel

Discrete Kernel Smoothing

Examples

```
k <- binomial.dkernel (5)
plot (k)

k$pmf (-2:2)
```

12_continuous_kernels *Continuous Kernels*

Description

Continuous kernels (over the interval, negative one to positive one), for continuous kernel smoothing.

NOTE THAT THESE MAY BE CHANGED.

Usage

```
uniform.ckernel ()
triangular.ckernel ()
epanechnikov.ckernel ()
truncnorm.ckernel ()
biweight.ckernel ()
triweight.ckernel ()
tricube.ckernel ()

bell.spline ()
```

Details

NOTE THAT THESE MAY BE CHANGED.
AND DISCRETE KERNELS MAY BE REMOVED.

These are constructors for continuous kernel objects.

Currently, these objects are named lists (with pdf and cdf components), however, it's possible that they may be changed, in the future.

Here, PDFs are symmetric about zero, and have positive density over the interval (-1, 1).

The truncnorm constructor creates a truncated normal (or truncated Gaussian) kernel, which is symmetrically truncated, such that the area from the untruncated distribution, within the truncation points, is 0.995 of that distribution. It's possible that I may allow relatively arbitrary truncation points, in the future.

The bell spline is a novel kernel, constructed from a three-piece quadratic spline, with knots at -0.5 and 0.5.

These objects are used inside continuous kernel smoothing.
(In which case, you provide the constructor, not the object).

Note that the kernel.array function can be used to plot and compare multiple kernels.

Value

A ckernel object.

References

Refer to the vignette for an overview, references and better examples.

See Also

Discrete Kernels

plot.ckernel
kernel.array

Continuous Kernel Smoothing

Examples

```
k <- biweight.ckernel ()
plot (k)

k$pdf (0)
```

21_succinct_constructors
Succinct Constructors

Description

Currently, these functions call the corresponding univariate constructors.

Usage

```
pmf.dks (...)
cdf.dks (...)
qf.dks (...)

pdf.cks (...)
cdf.cks (...)
qf.cks (...)

pmf.cat (...)
cdf.cat (...)
qf.cat (...)

cdf.el (...)
qf.el (...)

dfreq (...)
gfreq (...)
```

Arguments

...	Argument list for the corresponding constructor. Refer to the details section.

Details

Currently, these functions call the corresponding constructor with a uv suffix.

i.e.
pmf.dks calls pmfuv.dks
pdf.cks calls pdfuv.cks

The last two, dfreq and gfreq construct pmfuv.dks and pmfuv.cat models, but with freq=TRUE.

Value

Refer to univariate constructors.
i.e. pmfuv.dks.

References

Refer to the vignette for an overview, references and better examples.

See Also

Discrete Kernel Smoothing, Continuous Kernel Smoothing
Categorical Distributions, Empirical-Like Distributions

Conditional Distributions with Mixed Input Types
These can be used for statistical classification purposes.

Examples

```
ph.data.prep ()

cFht <- qf.cks (height)
cFht (0.5)
```

22_discrete_kernel_smoothing

Discrete Kernel Smoothing Models

Description

Fit probability distributions, via discrete kernel smoothing over integer-indexed frequency data.

Usage

```
pmfuv.dks (x, h, ...,
    bw.method="ph.default",
    kernel=binomial.dkernel,
    bounded = c (TRUE, FALSE), freq=FALSE,
    lower, upper,
    bw, smoothness=1)

cdfuv.dks (x, h, ...,
    bw.method="ph.default",
    kernel=binomial.dkernel,
    bounded = c (TRUE, FALSE),
    lower, upper,
    bw, smoothness=1)

qfuv.dks (x, h, ...,
```

```
bw.method="ph.default",
kernel=binomial.dkernel,
bounded = c (TRUE, FALSE),
lower, upper,
bw, smoothness=1)
```

Arguments

x	Integer vector of integer-indexed bins. Also, can be a single-column integer matrix, preferably with a column (variable) name, and optionally with row (bin) names. In principle, this is required. (It can be omitted if h supplied, but generates a warning).
h	Optional numeric vector of frequencies (or weights), which can be fractional. Defaults to a vector of ones.
bw.method	String, the bandwidth selection method. Refer to Bandwidth Selection.
kernel	Constructor for a dkernel (discrete kernel) object.
bounded	Logical vector of length one or two, determining whether the probability distribution is lower or upper bounded.
freq	Logical, if true, the resulting function object returns frequencies, by default.
bw, smoothness	Integer and numeric values, the bandwidth and smoothness parameters, respectively. The bandwidth parameter should be a positive odd number. (Even numbers are incremented). If the bandwidth is missing, it's computed using bw.method (see above) and the smoothness.
lower, upper	Optional integers, the lower and upper truncation values. They default to the lowest and highest bin values. Note that x (the bin values) must not be outside the truncation values. If so, and error is produced. Ignored, unless the corresponding bounded values are true.
...	Ignored.

Details

Refer to the vignette for more information.

Note that if x has non-unique values, then duplicated x (and their h) values are aggregated.
And currently, any row names will be ignored.

Value

Self-referencing function objects.

Refer to Mockup Function Objects

References

Refer to the vignette for an overview, references and better examples.

See Also

Discrete Kernels

Succinct Constructors
Continuous Kernel Smoothing, Categorical Distributions, Empirical-Like Distributions

is.dks, print.phmodel, plot.dksuv

Bandwidth Selection

Examples

```
ph.data.prep ()

dFht <- qfuv.dks (traffic.bins, traffic.freq, lower=0)
dFht (0.5)
```

23_continuous_kernel_smoothing

Continuous Kernel Smoothing Models

Description

Fit probability distributions, via continuous kernel smoothing, from data.

Usage

```
#univariate
pdfuv.cks (x, ..., spline=TRUE,
    bw.method="ph.default",
    kernel=biweight.ckernel,
    nc=30, bw, smoothness=1, w=NA)
cdfuv.cks (x, ..., spline=TRUE,
    bw.method="ph.default",
    kernel=biweight.ckernel,
    nc=30, bw, smoothness=1, w=NA)
qfuv.cks (x, ...,
    bw.method="ph.default",
    kernel=biweight.ckernel,
    nc=30, bw, smoothness=1, w=NA)

#multivariate
pdfmv.cks (x, ...,
    bw.method="ph.default",
    kernel=biweight.ckernel,
```

```
    bw, smoothness=1, w=NA)
cdfmv.cks (x, ...,
    bw.method="ph.default",
    kernel=biweight.ckernel,
    bw, smoothness=1, w=NA)

#conditional
pdfc.cks (x, ..., spline=TRUE,
    bw.method="ph.default",
    kernel=biweight.ckernel,
    nc=30, bw, smoothness=1, w=NA,
    conditions, preserve.range=FALSE,
    warning=TRUE)
cdfc.cks (x, ..., spline=TRUE,
    bw.method="ph.default",
    kernel=biweight.ckernel,
    nc=30, bw, smoothness=1, w=NA,
    conditions, preserve.range=FALSE,
    warning=TRUE)
qfc.cks (x, ...,
    bw.method="ph.default",
    kernel=biweight.ckernel,
    nc=30, bw, smoothness=1, w=NA,
    conditions, preserve.range=FALSE,
    warning=TRUE)

#multivariate-conditional
pdfmvc.cks (x, ...,
    bw.method="ph.default",
    kernel=biweight.ckernel,
    bw, smoothness=1, w=NA,
    conditions, preserve.range=FALSE,
    warning=TRUE)
cdfmvc.cks (x, ...,
    bw.method="ph.default",
    kernel=biweight.ckernel,
    bw, smoothness=1, w=NA,
    conditions, preserve.range=FALSE,
    warning=TRUE)

#other
chqf.cks (x, ...,
    bw.method="ph.default",
    kernel=biweight.ckernel,
    nc=16, bw, smoothness=1, w=NA)
```

Arguments

x	IN UNIVARIATE CONSTRUCTORS: Numeric vector of data. Also, can be a single-column numeric matrix, preferably with a column name. IN OTHER CONSTRUCTORS: Numeric matrix, preferably, with column names.
spline	Logical, if true, use cubic Hermite splines as intermediate models. In general, this should be true.
bw.method	String, the bandwidth selection method. Refer to Bandwidth Selection.
kernel	Constructor for a ckernel (continuous kernel) object.
nc	Integer, number of control points, in the spline. Ignored, if spline is false.
bw, smoothness	Numeric vectors of length one, or length m (the number of variables), the bandwidth and smoothness parameters. If bw is missing, the bandwidth is computed for each variable using bw.method (see above) and the smoothness.
w	Optional numeric vector of weights.
conditions	A numeric vector of conditioning values, preferably named. If named, then the names are matched against the variable names. If unnamed, then the first condition applies to the first variable, and the second condition applies to the second variable, and so on. Note that in univariate-conditional distributions, the number of conditions needs to equal the number of variables minus one.
preserve.range	Logical. If true, the default range used for range/sequence methods and plotting functions, will be the same as the original data. If false (the default), the range is based on the data within a conditioning window, which is often smaller.
warning	Logical, if true, generate warning if there's no observations within the conditional window.
...	Ignored.

Details

Refer to the vignette for more information.

Value

Self-referencing function objects.

Refer to Mockup Function Objects

Except:
The constructors for conditional distributions, return NULL, if there's no observations within the

conditional window.
(And by default, generate a warning).

References

Refer to the vignette for an overview, references and better examples.

See Also

Continuous Kernels

Succinct Constructors
Discrete Kernel Smoothing, Categorical Distributions, Empirical-Like Distributions

Conditional Distributions with Mixed Input Types
These can be used for statistical classification purposes.

is.cks, print.phmodel, plot.cksuv, plot.cksmv

Bandwidth Selection

Examples

```
ph.data.prep ()

cFht <- qfuv.cks (height)
cFht (0.5)
```

24_categorical_distributions

Categorical Models

Description

Fit categorical distributions, from data.

Usage

```
#univariate
pmfuv.cat (g, h, ..., freq=FALSE)
cdfuv.cat (g, h, ...)
qfuv.cat (g, h, ...)

#conditional
pmfc.cat (g, h, ..., conditions, warning=TRUE, freq=FALSE)
cdfc.cat (g, h, ..., conditions, warning=TRUE)
qfc.cat (g, h, ..., conditions, warning=TRUE)
```

Arguments

g	Integer/factor/character vector of groups. Also, can be a named list of such vectors. For univariate distributions, the list should only have one vector. For conditional distributions, the list needs two or more equal-length vectors.
h	Optional numeric vector of frequencies (or weights). The length of n should equal the length of the g vectors.
conditions	An integer vector of category indices, a character vector of category names, or a list which can contain either integers (indices) or strings (names). The vector or list can be named (which is preferable) or unnamed. If named, then the names are matched against the variable names. If unnamed, then the first condition applies to the first variable, and the second condition applies to the second variable, and so on. Note that the number of conditions needs to equal the number of variables minus one.
freq	Logical, if true, the resulting function object returns frequencies, by default.
warning	Logical, if true, generate warning if there's no observations within the conditional window.
...	Ignored.

Details

Refer to the vignette for more information.

Value

Self-referencing function objects.

Refer to Mockup Function Objects

Except:
The constructors for conditional distributions, return NULL, if there's no observations within the conditional window.
(And by default, generate a warning).

Note

WARNING:

If a categorical distribution is constructed from integers, the category indices won't necessarily equal the category names.

References

Refer to the vignette for an overview, references and better examples.

See Also

Succinct Constructors
Discrete Kernel Smoothing, Continuous Kernel Smoothing, Empirical-Like Distributions
Conditional Distributions with Mixed Input Types
These can be used for statistical classification purposes.
is.cat, print.phmodel, plot.catuv

Examples

```
ph.data.prep ()

gfh <- pmfuv.cat (crime.type, n.arrests)
gFht <- qfuv.cat (crime.type, n.arrests)

ph.mode (gfh)
gmode (gfh)

gFht (0.5)
gFht (0.5, category=TRUE)
```

25_empirical-like_distributions

Empirical-Like Models

Description

Fit empirical-like probability distributions, from data.

Usage

```
cdfuv.el (x, ..., w=NA)
qfuv.el (x, ..., w=NA)
```

Arguments

x	Numeric vector of data. Also can be a single-column numeric matrix, preferably with a column name.
w	Optional numeric vector of weights.
...	Ignored.

Details

Refer to the vignette for more information.

Value

Self-referencing function objects.

Refer to Mockup Function Objects

References

Refer to the vignette for an overview, references and better examples.

See Also

Succinct Constructors
Discrete Kernel Smoothing, Continuous Kernel Smoothing, Categorical Distributions

is.el, print.phmodel, plot.eluv

Examples

```
ph.data.prep ()

eFht <- qfuv.el (height)
eFht (0.5)
```

`26_mixed_conditional` *Conditional Distributions with Mixed Input Types*

Description

Fit conditional categorical or continuous distributions with mixed input data.
Such categorical distributions can be used for statistical classification purposes.

Usage

```
#conditional categorical
pmfc.gmixp (g, x, ..., conditions, warning=TRUE, w=NA, freq=FALSE)
cdfc.gmixp (g, x, ..., conditions, warning=TRUE, w=NA)
qfc.gmixp (g, x, ..., conditions, warning=TRUE, w=NA)

#conditional continuous
pdfc.xmixp (g, x, ..., conditions, warning=TRUE, w=NA)
cdfc.xmixp (g, x, ..., conditions, warning=TRUE, w=NA)
qfc.xmixp (g, x, ..., conditions, warning=TRUE, w=NA)
```

Arguments

g	Integer/factor/character vector of groups. Also, can be a named list of one or more such vectors.
x	A numeric vector or a numeric matrix, preferably with column names. The length of x (if standard vector) or the number or rows (if a matrix) should equal the length of the g vectors.

conditions	Refer to the conditions arg in categorical and continuous conditional models. This is the same, except that the vector or list, needs to be named (unnamed conditions are not allowed), and can include both categorical and continuous variables. Note that the number of conditions needs to equal the number of variables minus one. (For categorical distributions, there should be one categorical variable left out, and for continuous distributions there should be one continuous variable left out). The resulting probability distribution should be the conditional distribution of the variable not included in the conditions.
freq	Logical, if true, the resulting function object returns frequencies, by default.
w	Optional numeric vector of weights.
warning	Logical, if true, generate warning if there's no observations within the conditional window.
...	In categorical distributions, further arguments for pdfmv.cks, which is called on the continuous conditioning variables. In continuous distributions, further arguments for pdfuv or pdfc, which is called on the continuous conditional variable.

Details

Refer to the vignette for more information.

Note that categorical and continuous variables need different names.

Value

Self-referencing function objects.

Refer to Mockup Function Objects

Except:
The constructors for conditional distributions, return NULL, if there's no observations within the conditional window.
(And by default, generate a warning).

Note

WARNING:

If a conditional categorical distribution is constructed with integer g values, the category indices won't necessarily equal the category names.

References

Refer to the vignette for an overview, references and better examples.

See Also

Succinct Constructors
Discrete Kernel Smoothing, Continuous Kernel Smoothing
Categorical Distributions, Empirical-Like Distributions

is.cat, print.phmodel, plot.catuv

Examples

```
ph.data.prep ()

fh.gmix <- pmfc.gmixp (species, cbind (sepal.length, sepal.width),
    conditions = c (sepal.length=6, sepal.width=3) )
Fht.gmix <- qfc.gmixp (species, cbind (sepal.length, sepal.width),
    conditions = c (sepal.length=6, sepal.width=3) )

ph.mode (fh.gmix)
gmode (fh.gmix)

Fht.gmix (0.5)
Fht.gmix (0.5, category=TRUE)
```

27_distribution_sets *Distribution Sets*

Description

Sets of distributions.

THESE FUNCTIONS HAVE HAD LIMITED TESTING.

Usage

```
#categorical sets
#(grouped by a categorical variable)
pdfuv.gset.cks (x, ..., group.by)
cdfuv.gset.cks (x, ..., group.by)
qfuv.gset.cks (x, ..., group.by)
cdfuv.gset.el (x, ..., group.by)
qfuv.gset.el (x, ..., group.by)

#marginal sets
pdfuv.mset.cks (x, ..., bw, smoothness=1)
cdfuv.mset.cks (x, ..., bw, smoothness=1)
qfuv.mset.cks (x, ..., bw, smoothness=1)
cdfuv.mset.el (x, ...)
qfuv.mset.el (x, ...)
```

Arguments

x	Vector (for categorical sets) or matrix (for marginal sets).
group.by	A character vector, with the same length as x. Or an object which can be coerced to such a vector. If a list, its first element is used.
bw, smoothness	Bandwidth and smoothness parameters, same as pdfuv.cks, except that they can be an m-length vector, where m is the number of variables, equal to number of columns in x.
...	Other arguments for the corresponding univariate constructor.

Details

THESE FUNCTIONS HAVE HAD LIMITED TESTED.

These functions construct distribution set objects, which are lists of probability distributions.

Value

A ph3.gset or ph3.mset object.

References

Refer to the vignette for an overview, references and better examples.

See Also

Succinct Constructors
Discrete Kernel Smoothing, Continuous Kernel Smoothing
Categorical Distributions, Empirical-Like Distributions

plot.ph3.gset, plot.ph3.mset

Examples

```
ph.data.prep ()

plot (pdfuv.gset.cks (sepal.length, group.by=species) )
plot (qfuv.mset.el (trees), nr=2, nc=2)
```

31_is_functions *Is Functions*

Description

Functions to test if an object is of a particular class.

Usage

```
is.phob (xf)
is.phmodel (xf)

is.dpd (xf)
is.cpd (xf)

is.pmf (xf)
is.dcdf (xf)
is.dqf (xf)
is.pdf (xf)
is.ccdf (xf)
is.cqf (xf)

is.dpduv (xf, include.conditional=TRUE)
is.dpdc (xf, include.multivariate=TRUE)
is.cpduv (xf, include.conditional=TRUE)
is.cpdmv (xf, include.conditional=TRUE)
is.cpdc (xf, include.multivariate=TRUE)
is.cpdmvc (xf)

is.pmfuv (xf, include.conditional=TRUE)
is.pmfc (xf, include.multivariate=TRUE)
is.dcdfuv (xf, include.conditional=TRUE)
is.dcdfc (xf, include.multivariate=TRUE)
is.dqfuv (xf, include.conditional=TRUE)
is.dqfc (xf)

is.pdfuv (xf, include.conditional=TRUE)
is.pdfmv (xf, include.conditional=TRUE)
is.pdfc (xf, include.multivariate=TRUE)
is.pdfmvc (xf)
is.ccdfuv (xf, include.conditional=TRUE)
is.ccdfmv (xf, include.conditional=TRUE)
is.ccdfc (xf, include.multivariate=TRUE)
is.ccdfmvc (xf)
is.cqfuv (xf, include.conditional=TRUE)
is.cqfc (xf)

is.cchqf (xf)

is.dks (xf)
is.cks (xf)
is.cat (xf)
is.el (xf)

is.phspline (xf)
```

Arguments

`xf`	An object to test.
`include.conditional`	Logical, if true (the default), include conditional versions.
`include.multivariate`	Logical, if true (the default), include multivariate versions.

Details

Note that DPD and CPD stand for discrete and continuous probability distributions, respectively.

A leading "d" is discrete and a leading "c" is continuous.

Value

A single logical value.

References

Refer to the vignette for an overview, references and better examples.

See Also

Succinct Constructors
Discrete Kernel Smoothing, Continuous Kernel Smoothing
Categorical Distributions, Empirical-Like Distributions

Examples

```
ph.data.prep ()

dfh <- pmfuv.dks (traffic.bins, traffic.freq, lower=0)

is.dpd (dfh)
is.cpd (dfh)
is.pmf (dfh)
is.dpduv (dfh)
```

`32_range_and_sequence_methods`

Range and Sequence Functions

Description

Range and sequence methods for probability distributions.

Usage

```
## S3 method for class 'dpduv'
min(sf, infv=FALSE, ...)
## S3 method for class 'dpduv'
max(sf, infv=FALSE, ...)
## S3 method for class 'cpduv'
min(sf, infv=FALSE, ...)
## S3 method for class 'cpduv'
max(sf, infv=FALSE, ...)

## S3 method for class 'dpduv'
range(sf, infv=FALSE, ..., freq)
## S3 method for class 'cpduv'
range(sf, infv=FALSE, ...)

## S3 method for class 'dpduv'
seq(sf, infv=FALSE, ..., midpoints=TRUE, freq)
## S3 method for class 'cpduv'
seq(sf, infv=FALSE, ..., n=200)
```

Arguments

`sf`	A suitable function object. Here, this refers to a univariate probability distribution. Refer to the references and see also sections.
`infv`	Logical, in function value. Except for quantile functions, where this refers to the probabilities. Refer to the details section.
`midpoints`	Logical, if true, return midpoints. Ignored, except for discrete quantile functions with infv=TRUE. Refer to the details section.
`freq`	Logical, if true, return frequencies. Ignored, except for PMFs with infv=TRUE.
`n`	Integer, the number of points in the resulting sequence.
`...`	Ignored.

Details

By default, the min/max, range and sequence methods apply to range of the random variable.
Often this the range of the observations plus/minus half the bandwidth at each end.

Calling the sequence method on a discrete quantile function, with infv=TRUE:

If midpoints is true, then midpoints of the intervals are returned.
If midpoints is false, then breakpoints, including the outermost values, are returned.

Each interval is defined by one consecutive pair of breakpoints.
Where the breakpoints are (unique) values from the CDF, including zero (at the start) and one (at the end).

In general, these sequences are not equally-spaced.

Calling the sequence method on a continuous quantile function, with infv=TRUE:

Simply returns a returns an equally-spaced sequence between zero and one.

Value

Integer types are returned for discrete probability distributions with infv=FALSE.
Otherwise, numeric types are returned.

The min and max methods return a single integer/numeric value.
The range methods return an length-two integer/numeric vector.

And the seq methods return an integer/numeric vector.
This will be equally-spaced, if infv=FALSE.

References

Refer to the vignette for an overview, references and better examples.

See Also

Succinct Constructors
Discrete Kernel Smoothing, Continuous Kernel Smoothing
Categorical Distributions, Empirical-Like Distributions

Examples

```
ph.data.prep ()

dfh <- pmfuv.dks (traffic.bins, traffic.freq, lower=0)

seq (dfh)
seq (dfh, TRUE)
```

`33_print_methods` *Print Methods*

Description

Print methods for objects in this package.

Usage

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

Arguments

`x`	An object extending phmodel, which is most of the objects in this package.
`...`	Other arguments for the object.summary function.

Details

The print method calls intoo::object.summary.

References

Refer to the vignette for an overview, references and better examples.

See Also

Succinct Constructors
Discrete Kernel Smoothing, Continuous Kernel Smoothing
Categorical Distributions, Empirical-Like Distributions

Examples

```
ph.data.prep ()

dfh <- pmfuv.dks (traffic.bins, traffic.freq, lower=0)
dfh
```

34_kernel_plot_methods

Kernel Plot Methods

Description

Plot methods for kernel objects.

Usage

```
####################################
#discrete kernels
#(call plot_dpd)
####################################
## S3 method for class 'dkernel'
plot(x, ..., cdf=FALSE)

####################################
#continuous kernels
#(call plot_cpd)
####################################
## S3 method for class 'ckernel'
plot(x, ..., cdf=FALSE)
```

Arguments

x	A kernel object. Refer to the references and see also sections.
cdf	Logical, if true, plot the CDF.
...	Other arguments, for plot_dpd or plot_cpd.

References

Refer to the vignette for an overview, references and better examples.

See Also

Discrete Kernels, Continuous Kernels

kernel.array

plot_dpd, plot_cpd

Examples

```
plot (biweight.ckernel () )
```

35_model_plot_methods *Model Plot Methods*

Description

Plots methods for models, excluding distribution sets.

Usage

```
###################################
#discrete kernel smoothing models
#(call plot_dpd)
###################################
## S3 method for class 'dksuv'
plot(x, data=FALSE, ..., freq)

###################################
#continuous kernel smoothing models
#(call plot_cpd, plot_cpd_bv or plot_cpd_tv)
###################################
## S3 method for class 'cksuv'
plot(x, data=FALSE, ...)
## S3 method for class 'cksmv'
plot(x, in3d=FALSE, data=FALSE, ...)
## S3 method for class 'cksc'
plot(x, ...)
```

```
## S3 method for class 'cksmvc'
plot(x, in3d=FALSE, data=FALSE, ...)

###################################
#categorical models
#(call plot_dpd)
###################################
## S3 method for class 'catuv'
plot(x, ..., combine, freq, space)

###################################
#empirical-like models
#(call plot_cpd)
###################################
## S3 method for class 'eluv'
plot(x, data=FALSE, ...)

###################################
#all continuous univariate models
###################################
## S3 method for class 'cpduv'
lines(x, ...)
```

Arguments

x	A probability distribution. Refer to the references and see also sections.
in3d	Logical, if true, create a 3D plot. Ignored, if sf has three or more random variables.
data	If true, include a subpanel with data bars/points. Ignored, if x is a quantile function, a conditional distribution, or has three or more random variables
combine	Logical, if true, combine the bars. By default, categorical distributions are separate and others are combined.
freq	Logical, if true, return frequencies rather than probabilities. Refer to the value section. Default depends on the object. Refer to the corresponding constructors.
space	Numeric, the space (in mm) between bars. Ignored, unless combine is false.
...	Other arguments for plot_dpd, plot_cpd, plot_cpd_bv and plot_cpd_tv.

Details

Refer to the vignette for more information.

Note that these methods call the functions plot_dpd, plot_cpd, plot_cpd_bv and plot_cpd_tv. Please refer to these functions for more information.

References

Refer to the vignette for an overview, references and better examples.

See Also

Succinct Constructors
Discrete Kernel Smoothing, Continuous Kernel Smoothing
Categorical Distributions, Empirical-Like Distributions

plot_dpd, plot_cpd

plot_cpd_bv, plot_cpd_tv

Examples

```
ph.data.prep ()

dfh <- pmfuv.dks (traffic.bins, traffic.freq, lower=0)
cfh <- pdfuv.cks (height)

cfh2 <- pdfmv.cks (trees [,-2])
cfh3 <- pdfmv.cks (trees)

plot (dfh, TRUE)
plot (cfh, TRUE)
plot (cfh2,, TRUE)
plot (cfh3)
```

36_other_plot_methods *Other Plot Methods*

Description

Plot methods for distribution sets.

NOTE THAT THESE FUNCTIONS WILL BE REPLACED.

Usage

```
## S3 method for class 'ph3.gset'
plot(x, ..., legend=TRUE, colors)
## S3 method for class 'ph3.mset'
plot(x, ..., nr, nc, colors)
```

Arguments

x	A ph3.gset or ph3.mset object.
legend	Logical, if true, add a legend to the plot.
colors	Character vector, colors for each distribution.

nr, nc	Optional integers, number of panels.
...	Ignored.

References

Refer to the vignette for an overview, references and better examples.

See Also

Distribution Sets

Examples

```
ph.data.prep ()

plot (pdfuv.gset.cks (sepal.length, group.by=species) )
plot (qfuv.mset.el (trees), nr=2, nc=2)
```

41_global_options *Global Options*

Description

Set the global options, including the color theme.

Usage

```
set.ph.options (...,
    rendering.style="r", theme="blue",
    main.line.width=1, main.line.color="#000000",
    main.fill.color, main.fill.color.2="#B0B0B0",
    semi.fill.color = "#00000030",
    palette="earth")

set.ph.theme (theme="blue")
```

Arguments

rendering.style	Single character. Refer to barsurf::set.bs.options.
theme	Character, either "blue" or "green". Other themes from the barsurf package are partially supported.
main.line.width	Numeric, default main line width.
main.line.color	String (R color string), default main line color.

`main.fill.color`
: String (R color string), default color for the area under univariate probability distributions, excluding quantile functions.

`main.fill.color.2`
: String (R color string), default color for the area under quantile functions, and univariate dks/cks data bars/points.

`semi.fill.color`
: String (R color string), default color for bivariate data points.

`palette`
: String, color palette, for grDevices::hcl.colors, used when plotting distribution sets, and by the kernel.array function.

`...`
: Ignored.

Details

This function calls barsurf::set.bs.options, to set the rendering.style and theme.

References

Refer to the vignette for an overview, references and better examples.

Examples

```
set.ph.theme ("blue")
```

`51_duv_plotting_functions`

Plots of Discrete Univariate Models

Description

Plots of discrete univariate probability distributions.

Usage

```
plot_dpd (sf, data=FALSE, ...,
    main, xlab, ylab,
    xlim, ylim,
    add=FALSE, axes=TRUE,
    combine=FALSE, freq, space=0,
    line.width, line.color, fill.color)
```

Arguments

`sf`	A suitable function object. Here, this is a discrete univariate probability distribution. Refer to the references and see also sections.
`data`	Logical, if true, include a subpanel with the data bars. Ignored, if sf is a quantile function or a conditional distribution.
`main, xlab, ylab`	Optional strings, main/axes titles.
`xlim, ylim`	Optional length-2 numeric vectors, giving the plot ranges.
`add`	Logical, if true, add to an existing plot.
`axes`	Logical vector of length one or two, if true, plot axis ticks with labels.
`combine`	Logical, if true, combine the bars.
`freq`	Logical, if true, plot frequencies. Ignored, except for PMFs. Default depends on the object. Refer to the corresponding constructors.
`space`	Numeric, the space (in mm) between the bars. Ignored, if combine is true.
`line.width`	Optional numeric, giving the main line width. If missing, determined by global options.
`line.color, fill.color`	Optional (R color) strings, giving the main line color and main fill color. If missing, determined by global options.
`...`	Ignored.

References

Refer to the vignette for an overview, references and better examples.

See Also

set.ph.options

plot.dksuv, plot.catuv

Examples

```
ph.data.prep ()

dfh <- pmfuv.dks (traffic.bins, traffic.freq, lower=0)

plot (dfh)
plot (dfh, TRUE)
```

52_cuv_plotting_functions

Plots of Continuous Univariate Models

Description

Plots of continuous univariate probability distributions.

Usage

```
plot_cpd (sf, data=FALSE, ...,
    main, xlab, ylab,
    xlim, ylim,
    add=FALSE, axes=TRUE,
    line.width, line.color, fill.color)
```

Arguments

`sf`	A suitable function object. Here, this is a continuous univariate probability distribution. Refer to the references and see also sections.
`data`	Logical, if true, include a subpanel with data points. Ignored, if sf is a quantile function, or a conditional distribution.
`main, xlab, ylab`	Optional strings, main/axes titles.
`xlim, ylim`	Optional length-2 numeric vectors, giving the plot ranges.
`add`	Logical, if true, add to an existing plot.
`axes`	Logical vector of length one or two, if true, plot axis ticks with labels.
`line.width`	Optional numeric, giving the main line width. If missing, determined by global options.
`line.color, fill.color`	Optional (R color) strings, giving the main line color and main fill color. If missing, determined by global options.
`...`	Ignored.

References

Refer to the vignette for an overview, references and better examples.

See Also

set.ph.options

plot.cksuv, plot.eluv

plot_cpd_bv, plot_cpd_tv

Examples

```
ph.data.prep ()

cfh <- pdfuv.cks (height)

plot (cfh)
plot (cfh, TRUE)
```

53_cmv_plotting_functions

Plots of Continuous Multivariate Models

Description

Plots of bivariate and trivariate continuous probability distributions.

Usage

```
#calls barsurf::plot_cfield or barsurf::plot_surface
plot_cpd_bv (sf, in3d=FALSE, data=FALSE, ..., all=FALSE, n=30,
    main, xlab, ylab,
    xlim, ylim, zlim)

#calls barsurf::plot_cfield_3d
plot_cpd_tv (sf, ...,
    main, xlab, ylab, zlab,
    xlim, ylim, zlim,
    reverse.z=FALSE)
```

Arguments

sf	A suitable function object. Here, this is a continuous multivariate probability distribution, with two or three random variables. Refer to the references and see also sections.
in3d	Logical, if true, produce a 3D plot.
data	Logical, if true, plot data points. Ignored, if in3d is true, or sf is a conditional distribution.
main, xlab, ylab, zlab	Optional strings, main/axes titles. Note that these depend on the barsurf package. And at the time of writing (with barsurf version 0.5.0), zlab is not supported.
xlim, ylim, zlim	Optional length-2 numeric vectors, giving the plot ranges.

reverse.z	Logical, if true, reverse the z axis. Ignored, if zlim supplied.
all	Logical, if true, plot a 2x2 plot array, with different subplots. Ignored, if sf is a conditional distribution.
n	Numeric vector of length one or two, giving the number of grid points in each direction.
...	Other arguments for barsurf::plot_cfield, barsurf::plot_surface, plot::plot_cfield_3d.

Details

These functions call barsurf::plot_cfield, barsurf::plot_surface, plot::plot_cfield_3d.

References

Refer to the vignette for an overview, references and better examples.

See Also

set.ph.options

plot_cpd

plot.cksmv

Examples

```
ph.data.prep ()

cfh2 <- pdfmv.cks (trees [,-2])
cfh3 <- pdfmv.cks (trees)

plot (cfh2)
plot (cfh2, TRUE)
plot (cfh2,, TRUE)
plot (cfh2, all=TRUE)

plot (cfh3)
```

54_pairwise_kernel_arrays

Pairwise Kernel Arrays

Description

Plots of pairwise kernel arrays.

Usage

```
list.ckernels ()

kernel.array (ks = list.ckernels (), ..., ref.line=TRUE, colors)
```

Arguments

ks	List of kernel objects.
ref.line	If true, add a reference line.
colors	Optional character vector of colors for each plot.
...	Ignored.

Details

The function list.ckernels, simply returns a list of ckernel (continuous kernel) objects.

The kernel.array function plot pairs of kernels, for comparison purposes.

References

Refer to the vignette for an overview, references and better examples.

See Also

Discrete Kernels, Continuous Kernels

plot.dkernel, plot.ckernel

Examples

```
kernel.array ()
```

61_bandwidth_selection

Bandwidth Selection

Description

Functions for bandwidth selection.

THESE FUNCTIONS SHOULD BE REGARDED AS SUB-OPTIMAL.

Usage

```
auto.dbw (x, ..., bw.method="ph.default", smoothness=1)
auto.cbw (x, ..., bw.method="ph.default", smoothness=1)
```

Arguments

x	A numeric vector, of data. In the continuous case, x may also be a matrix.
bw.method	String, the initial bandwidth selction method. Currently, "ph.default", "Scott" or "Silverman". Refer to details.
smoothness	Numeric, smoothness (scaling) parameter. Refer to details.
...	Ignored.

Details

THESE FUNCTIONS SHOULD BE REGARDED AS SUB-OPTIMAL.

These functions computes an initial bandwidth.
Then the initial bandwidth parameter is multiplied by the smoothness parameter.

In the discrete case (auto.dbw), this bandwidth is rounded up to the nearest odd integer.

Currently, there are three options:

(1) bw.method="ph.default".

For a single variable/column, the bandwidth is equal to the difference between the quantiles, marking the middle 0.66 of observations.
i.e. diff (quantile (x, c (0.17, 0.83)))

For m variables/columns, 0.66 is replaced with 0.66^(1 / m).

(2) bw.method="Scott", which calls stats::bw.nrd, for each variable/column.

(3) bw.method="Silverman", which call stats::bw.nrd0, for each variable/column.

Value

In the discrete case, a single integer.
In the continuous case, a numeric vector.

References

Refer to the vignette for an overview, references and better examples.

Also please refer to stats::bw.nrd and stats::bw.nrd0 for references, and more information.

See Also

bw.nrd, bw.nrd0

Discrete Kernel Smoothing, Continuous Kernel Smoothing

Examples

```
ph.data.prep ()

auto.cbw (trees)
```

71_multivariate_probabilities
Multivariate Probabilities

Description

Compute probabilities, from multivariate CDFs.

Usage

```
probmv (sf, a, b)
```

Arguments

sf	A cdfmv.cks or cdfmvc.cks object.
a, b	Numeric vectors (or matrices) of lower and upper limits, corresponding to each variable. If they're matrices, then each row defines the limits for one region and each column defines the limits for one variable.

Details

Refer to the vignette for more information.

Value

A single numeric value (if a and b are both standard vectors) and a numeric vector (if either a or b are matrices).

References

Refer to the vignette for an overview, references and better examples.

See Also

cdfmv.cks, cdfmvc.cks

ph.mean
moment

rng

quartiles, ntiles, ph.median, ph.quantile
ph.mode, ph.modes\

Examples

```
ph.data.prep ()

cFh3 <- cdfmv.cks (trees)

q <- matrix (c (
    22, 24,    #height in 22 to 24
    28, 38,    #girth  in 28 to 38
    0.55, 1.05 #volume in 0.55 to 1.05
    ),, 2, byrow=TRUE, dimnames = list (colnames (trees), c ("a", "b") ) )

probmv (cFh3, q [,1], q [,2])
```

72_random_number_generation

Random Numbers

Description

Generate random numbers (or synthetic data), univariate or multivariate.

Usage

```
rng (xf, n=1, ...)
```

Arguments

xf	A numeric vector, suitable function object, or an object that can be coerced to a numeric vector. Here, a suitable function object is quantile function, or a chained quantile function. Refer to the references and see also sections.
n	Integer, number of random numbers.
...	Other arguments. Refer to the details section.

Details

If xf is a numeric vector, a qfuv.el object is created using xf as the main argument.
Any arguments contained within ..., are passed to the qfuv.el constructor.

If xf is not a quantile function, these functions try to coerce it to a numeric vector, and apply the above.

Note that the method used for multivariate random number generation is not efficient.

Value

A numeric vector, or numeric matrix.

References

Refer to the vignette for an overview, references and better examples.

See Also

Succinct Constructors
Discrete Kernel Smoothing, Continuous Kernel Smoothing
Categorical Distributions, Empirical-Like Distributions

Examples

```
ph.data.prep ()

cFht <- qfuv.cks (height)
rng (cFht, 30)

chFht <- chqf.cks (trees)
rng (chFht, 30)

rng (height, 30)
```

73_moment-based_statistics

Moment-Based Statistics

Description

Compute moment-based statistics from probability distributions.

Usage

```
ph.mean (sf, ..., n.intervals=200)
ph.sd (sf, ..., n.intervals=200)
ph.var (sf, ..., n.intervals=200)
ph.skewness (sf, ..., n.intervals=200)
ph.kurtosis (sf, ..., n.intervals=200)

moment (sf, nth, ..., n.intervals=200)

central.moment (sf, nth, ..., n.intervals=200)
standardized.moment (sf, nth, ..., n.intervals=200)

raw.moment (sf, nth, about=0, ..., n.intervals=200)
```

Arguments

sf	A suitable function object. Here, this is a univariate PMF or spline-based CDF. Refer to the references and see also sections.
nth	Integer, the nth moment.
about	Numeric, the about constant for raw moments.
n.intervals	Integer. In the discrete case, ignored. In the continuous case, the number of intervals, used in the numerical approximation.
...	Ignored.

Details

The mean/sd/var/skewness/kurtosis functions all call the moment function.

If the moment function is called with nth equal zero, it returns one.
If called with nth=1 (the mean), it computes the the first raw moment.
If called with nth=2 (the variance), it computes the second central moment.
If called with nth=3 (the skewness), it computes the third standardized moment.
If called with nth=4 (the kurtosis), it computes the fourth standardized moment.
And if called with nth>4, it also computes the nth standardized moment.

Note that currently, the standard deviation, variance and higher moments, should should not be regarded as accurate.

Value

A single numeric value.

References

Refer to the vignette for an overview, references and better examples.

See Also

Succinct Constructors
Discrete Kernel Smoothing, Continuous Kernel Smoothing, Empirical-Like Distributions

probmv, rng

quartiles, ntiles, ph.median, ph.quantile
ph.mode, ph.modes

Examples

```
ph.data.prep ()

cFh <- cdfuv.cks (height)
```

```
ph.mean (cFh)
ph.sd (cFh)
ph.skewness (cFh)
ph.kurtosis (cFh)
```

`74_order-based_statistics` *Order-Based Statistics*

Description

Compute named sequential quantiles, primarily for producing summary-style output.

Usage

```
ntile.names (symbol="q", n, ..., emph = n / 2)

quartiles (xf, col=FALSE, ...,
    rank=TRUE, names = ntile.names ("Q", 4, emph=emph), emph=2)
deciles (xf, col=FALSE, ...,
    rank=TRUE, names = ntile.names ("D", 10, emph=emph), emph=5)

ntiles (xf, n, col=FALSE, ..., rank=TRUE, names)
```

Arguments

`symbol`	String, letter/symbol for the quantile names.
`n`	Integer, the number of sequential quantiles.
`xf`	A numeric vector, suitable function object, or an object that can be coerced to a numeric vector. Here, a suitable function object is a quantile function. Refer to the references and see also sections.
`col`	Logical, if true, return a single-column matrix.
`rank`	Logical, if true, name quantiles by index/names, if false, name quantiles by probability.
`names`	Character vector, giving the names. Ignored, if rank is false.
`emph`	In principle, an integer vector in 1:(n-1), which quantiles to emphasize. Can also be a numeric vector, but the floor/ceiling values are used.
`...`	Other arguments. Refer to details.

Details

If xf is a numeric vector, a qfuv.el object is created using xf as the main argument.
Any arguments contained within ..., are passed to the qfuv.el constructor.

If xf is not a quantile function, these functions try to coerce it to a numeric vector, and apply the above.

Value

ntiles.names returns a character vector.

The other functions return a named numeric vector (if col=FALSE), or a named single-column matrix (if col=TRUE).

References

Refer to the vignette for an overview, references and better examples.

See Also

Succinct Constructors
Discrete Kernel Smoothing, Continuous Kernel Smoothing, Empirical-Like Distributions

probmv, rng

ph.mean, moment
ph.median, ph.quantile
ph.mode, ph.modes

Examples

```
ph.data.prep ()

cFht <- qfuv.cks (height)

quartiles (cFht)
quartiles (cFht, rank=FALSE)

quartiles (height)
```

75_robust-based_statistics

Robust Statistics

Description

Compute robust statistics from probability distributions.

Usage

```
ph.median (xf, ...)
ph.quantile (xf, p, ...)

iqr (xf, P=0.5, ...)
```

Arguments

`xf`	A numeric vector, suitable function object, or an object that can be coerced to a numeric vector. Here, a suitable function object is a quantile function. Refer to the references and see also sections.
`P, p`	Numeric vectors, the probabilities. P is the area (probability) between the lower and upper limits.
`...`	Other arguments. Refer to the details section.

Details

If xf is a numeric vector, a qfuv.el object is created using xf as the main argument.
Any arguments contained within ..., are passed to the qfuv.el constructor.

If xf is not a quantile function, these functions try to coerce it to a numeric vector, and apply the above.

Value

ph.median returns a single numeric value.

The other functions return a numeric vector.

References

Refer to the vignette for an overview, references and better examples.

See Also

Succinct Constructors
Discrete Kernel Smoothing, Continuous Kernel Smoothing, Empirical-Like Distributions

probmv, rng

ph.mean, moment
quartiles, ntiles
ph.mode, ph.modes

Examples

```
ph.data.prep ()

cFht <- qfuv.cks (height)

cFht (0.5)
ph.median (cFht)

iqr (cFht)
```

`76_mode_estimation` *Mode Estimation*

Description

Compute the mode or modes, from probability distributions.

Usage

```
ph.mode (sf, infv=FALSE, ..., name=FALSE, freq)
ph.modes (sf, infv=FALSE)

gmode (sf)
```

Arguments

`sf`	A suitable function object. For ph.mode, this is a univariate PMF or spline-based PDF. For ph.modes, a spline-based PDF only. Refer to the references and see also sections.
`infv`	Logical, if true, return the value of the PMF/PDF at the mode(s).
`name`	Logical, if false, return the category index, if true, return the category name. Ignored, except for categorical PMFs with infv=FALSE.
`freq`	Logical, if true, return frequencies. Ignored, except for PMFs with infv=TRUE. Default depends on the object. Refer to the corresponding constructors.
`...`	Ignored.

Details

gmode is a wrapper for ph.mode with name=TRUE.

Value

ph.mode returns a single integer or numeric value.
(Except for categorical PMFs with infv=FALSE and name=TRUE, which return a string).

ph.modes returns a numeric vector.

References

Refer to the vignette for an overview, references and better examples.

See Also

Succinct Constructors
Discrete Kernel Smoothing, Continuous Kernel Smoothing, Categorical Distributions, Empirical-Like Distributions

probmv, rng

ph.mean, moment
quartiles, ntiles, ph.median, ph.quantile

Examples

```
ph.data.prep ()

gfh <- pmfuv.cat (crime.type, n.arrests)
cfh.nondefault <- pdfuv.cks (height, smoothness=0.5)

ph.mode (gfh)
ph.mode (gfh, TRUE)
ph.mode (gfh, TRUE, freq=TRUE)
ph.mode (gfh, category=TRUE)

gmode (gfh)

ph.mode (cfh.nondefault)
ph.modes (cfh.nondefault)
```

81_mockup_function_objects

Mockup Function Objects

Description

Hard-coded functions, representing (runtime) function objects.

DO NO CALL THESE FUNCTIONS.
CALL A CONSTRUCTOR, WHICH SHOULD RETURN A FUNCTION OBJECT.

Usage

```
#discrete kernel smoothing models
#(DKS)
mf.dfh (x, ..., freq)         #PMF
mf.dFh (q)                    #CDF
mf.dFht (p)                   #QF

#categorical models
#(CAT/gMIXp)
mf.gfh (g, ..., freq)         #PMF
mf.gFh (q)                    #CDF
mf.gFht (p, ..., name=FALSE)  #QF

#univariate continuous models
#(CKS/EL/xMIXp, UV/C)
mf.cfh (x)                    #PDF
mf.cFh (q)                    #CDF
mf.cFht (p)                   #QF

#multivariate continuous models
#(CKS, MV/MVC)
mf.cfh.mv (x)                 #PDF
mf.cFh.mv (q)                 #CDF

#chained quantile functions
mf.chFht (p)
```

Arguments

g, x, q	IN (DKS) MODELS: An integer vector, of quantiles. IN (CAT/gMIXp) MODELS: An integer/factor/character vector, of quantiles. Integers represent category indices. Characters and formatted factors represent category names. IN (CKS/EL/xMIXp, UV/C) MODELS: A numeric vector, of quantiles. IN (CKS, MV/MVC) MODELS: A numeric vector or matrix, of quantiles. Standard numeric vectors are rbind-ed into single-row matrices. Each row represents one evaluation point, and each column represents one variable.
p	A numeric vector of probabilities, between zero and one. Except in chained quantile functions, where p should be a numeric vector or

	matrix. Standard numeric vectors are rbind-ed into single-row matrices.
`freq`	Logical, if true, return frequencies rather than probabilities. Refer to the value section. Default depends on the object. Refer to the corresponding constructors.
`name`	Logical, if true, return category names rather than category indices. Refer to the value section.
`...`	.

Details

DO NO CALL THESE FUNCTIONS.
CALL A CONSTRUCTOR, WHICH SHOULD RETURN A FUNCTION OBJECT.

If x, q or p are matrices, then the order of the columns should be the same as the order of the random variables in the model.

Value

PMFs return a numeric vector, giving mass (if freq=FALSE) or frequencies (if freq=TRUE).

PDFs return a numeric vector, giving density.

CDFs return a numeric vector, giving cumulative probability (from zero to one).

In (DKS) models, QFs return an integer vector of quantiles.

In (CAT/gMIXp) models, QFs return an integer vector (if name=FALSE) of category indices or a character vector (if name=TRUE) of category names.

In (CKS/EL/xMIXp, UV/C) models, QFs return a numeric vector of quantiles.

Chained quantiles, return a numeric matrix of multivariate quantiles.

References

Refer to the vignette for an overview, references and better examples.

See Also

Discrete Kernel Smoothing, Continuous Kernel Smoothing
Categorical Distributions, Empirical-Like Distributions

Conditional Distributions with Mixed Input Types

82_other_functions *Other Functions*

Description

Emulate an R script, and prepare data for examples.

Usage

```
ph.data.prep (..., eval=TRUE, echo=FALSE)
```

Arguments

eval	Logical, if true, run the script.
echo	Logical, if true, echo the script.
...	Ignored.

Details

The ph.data.prep function, which is only designed for tests and examples, runs an R script stored inside the function, itself.

References

Refer to the vignette for an overview, references and better examples.

Examples

```
ph.data.prep (eval=FALSE, echo=TRUE)
```

Index