Package 'probhat'

May 12, 2021

Title Multivariate Generalized Kernel Smoothing and Related Statistical Methods

Version 0.4.1

Date 2021-05-12

License GPL (>= 2)

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

Author Abby Spurdle

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

Description Probability mass functions (PMFs), probability density functions (PDFs), cumulative distribution functions (CDFs) and quantile functions, mainly via (optionally bounded/truncated) kernel smoothing. In the continuous case, there's support for univariate, multivariate and conditional distributions, including distributions that are both multivariate and conditional. Refer to the book ``Kernel Smoothing'' by Wand and Jones (1995), whose methods are generalized by the methods here. Also, supports categorical distributions, mixed conditional distributions (with mixed input types) and smooth empirical-like distributions, some of which, can be used for statistical classification. There are extensions for computing distance matrices (between distributions), multivariate probabilities, multivariate random numbers, moment-based statistics and mode estimates.

Depends methods

Imports barsurf, kubik

Suggests bivariate, fclust, scatterplot3d

Contact Primary <spurdle.a@gmail.com>, Secondary <spurdle.a@aol.com>

NeedsCompilation no

Repository CRAN

Date/Publication 2021-05-12 09:40:02 UTC

R topics documented:

`00_kernels` *Kernel Objects*

Description

Discrete and continuous kernel objects.

NOTE THAT THEIR INTERNAL STRUCTURE (THAT IS, THEIR ATTRIBUTES/SLOTS), IS SUBJECT TO CHANGE.

Usage

```
discretized.kernel (n, ck=BIWEIGHT.CKERNEL, ..., xlim)

UNIFORM.CKERNEL
TRIANGULAR.CKERNEL
EPANECHNIKOV.CKERNEL
TRGAUSSIAN.CKERNEL
BIWEIGHT.CKERNEL
TRIWEIGHT.CKERNEL
TRICUBE.CKERNEL

BELL.SPLINE
```

Arguments

n	Integer, number of bins. Needs to be positive and odd. Ignored, if xlim provided.
ck	A continuous kernel object.
xlim	A length two ascending integer vector.
...	Ignored.

Details

Kernel objects are S4 objects with two slots representing the corresponding PMF/PDF and CDF.

Continuous kernels are predefined constants.
Discrete kernels are constructed by using the discretized.kernel() function to discretize a predefined continuous kernel.

Currently, constructors for both DKS and CKS objects take a continuous kernel object.
(Where the DKS constructors discretize it, internally).

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

Currently, the truncated Gaussian kernel is symmetrically truncated (then transformed), such that the area from the untruncated distribution is 0.995. The bell spline is a novel kernel, constructed from a three-piece quadratic spline, with knots at -0.5 and 0.5.

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

Value

A Kernel object.

References

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

See Also

ph.plotf.DKernel, ph.plotf.CKernel
plot_kernel_array

DKS and CKS Models
Conditional Distributions with Mixed Input Types

Examples

```
dk <- discretized.kernel (7)
plot (dk)

plot (BIWEIGHT.CKERNEL)
```

01_bridging_functions *Bridging Functions*

Description

Generic functions for calling S3 generics with function objects, or objects that contain function objects.
In this package, such objects are often labelled, sf, for suitable function.
(They're also often used to represent probability distributions).

In general, it's easier to call the standard generic.
i.e. It's easier to call print() rather than ph.printf().

However, if using the standard generic, it's best not to name the first argument.

Usage

```
ph.namesf (...)
ph.printf (...)
ph.plotf (...)
ph.linesf (...)

## S3 method for class 'phob'
names(x, ...)
## S3 method for class 'phob'
print(x, ...)
## S3 method for class 'phob'
plot(x, ...)
## S3 method for class 'phob'
lines(x, ...)
```

Arguments

x	The first argument.
...	Further arguments, for the S3 method.

References

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

Examples

```
print (BIWEIGHT.CKERNEL)
ph.printf (BIWEIGHT.CKERNEL)
```

`10_is_functions` *Is Functions*

Description

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

Usage

```
is.phob (object)
is.phpd (object)
is.phmodel (object)

is.dpd (object)
is.cpd (object)

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

is.pduv (object, include.conditional=TRUE)

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

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

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

is.cchqf (object)

is.dks (object)
is.cks (object, include.xmix=TRUE)
is.cat (object, include.gmix=TRUE)
is.el (object)

is.gmix (object)
is.xmix (object)

is.phspline (object)
```

Arguments

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

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.

Also, note that these relate to the objects, and not the number of variables. (i.e. Object of class pdfmv.cks, are designed for multivariate models, but can be constructed with a single variable).

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

```
prep.ph.data ()

dfh <- pmfuv.dks (traffic.bins, traffic.freq)
is.dpd (dfh)
is.cpd (dfh)
```

`11_names_methods` *Names Methods*

Description

Print variable names, taken from the input data.

Usage

```
## S3 method for class 'phmodel'
ph.namesf(sf, ..., all=FALSE)
## S3 method for class 'ph4.gset'
ph.namesf(sf, ...)
```

Arguments

`sf`	An object extending phmodel (or a gset object, for the gset method), which is most of the objects in this package.
`all`	Logical, if false (the default), only include conditional/random variables.
`...`	Ignored.

Value

A character vector of variable 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

Examples

```
prep.ph.data ()

cfh <- pdfmv.cks (trees2)
names (cfh)
```

12_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 (...)
```

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

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

```
prep.ph.data ()

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

`20_discrete_kernel_smoothing`
Discrete Kernel Smoothing Models

Description

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

NOTE THAT THESE OBJECTS ARE LIKELY TO BE CONVERTED TO S4 OBJECTS, IN THE NEAR FUTURE.
ALSO, NOTE THAT THEIR INTERNAL STRUCTURE (THAT IS, THEIR ATTRIBUTES/SLOTS), IS SUBJECT TO CHANGE.

IN PRINCIPLE, YOU SHOULD NOT ACCESS ATTRIBUTES/SLOTS, DIRECTLY.

Usage

```
pmfuv.dks (x = 1:length (h), h=1, ...,
    bw, smoothness=1,
    kernel=BIWEIGHT.CKERNEL,
    bw.method="ph.default",
    Xlim = c (a, b),
    a = min1 (x), b=Inf)

cdfuv.dks (x = 1:length (h), h=1, ...,
    bw, smoothness=1,
    kernel=BIWEIGHT.CKERNEL,
    bw.method="ph.default", tail="lower",
    Xlim = c (a, b),
    a = min1 (x), b=Inf)

qfuv.dks (x = 1:length (h), h=1, ...,
```

```
bw, smoothness=1,
kernel=BIWEIGHT.CKERNEL,
bw.method="ph.default",
Xlim = c (a, b),
a = min1 (x), b=Inf)
```

Arguments

x	Integer vector of integer-indexed discrete observations, or bins of such observations. (If duplicates, they, along with their frequencies, are aggregated). Also, can be a single-column integer matrix, preferably with a column (variable) name, and optionally with row (bin) names. Defaults to a sequence from one to the length of h.
h	Positive numeric vector of frequencies (or weights), which can be fractional. (If scalar, it's recycled to match the length of pre-aggregate x). Defaults to one, such that each x value represents a single discrete value.
bw	Odd positive integer value, giving the bandwidth parameter. If missing, an initial bandwidth is computed via the the bandwidth method (see bw.method below), which is subject to the smoothness parameter.
smoothness	Positive numeric value, giving the relative bandwidth. Ignored, if bw is provided.
kernel	A (continuous) kernel object.
bw.method	String, the bandwidth selection method. Refer to Bandwidth Selection.
tail	String, either "lower" or "upper". If lower (the default), lower tail probabilities, P (X <= x), are used. If upper, upper tail probabilities, P (X >= x), are used.
Xlim	In principle, a length-two integer vector, giving the limits of X. But a numeric vector is allowed, to support -Inf/Inf. The corresponding random variable is regarded as bounded, if either limit is finite. In which case, a truncated smoothing algorithm is applied.
a, b	In principle, integer values. This is an alternative way of specifying Xlim, above. The min1 function will return one, if one is the minimum x value, otherwise, it will return zero.
...	Additional arguments not allowed.

Details

PLEASE SET NOTES IN DESCRIPTION FIELD.

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.

Also note that the truncation method may change in future updates.

Value

Self-referencing function objects.

Refer to Runtime Function Objects

References

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

See Also

Kernels

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

is.dks, ph.printf.phmodel, ph.plotf.dksuv

Bandwidth Selection

Examples

```
prep.ph.data ()

dfh <- pmfuv.dks (traffic.bins, traffic.freq)
dFh <- cdfuv.dks (traffic.bins, traffic.freq)
dFht <- qfuv.dks (traffic.bins, traffic.freq)

plot (dfh)
plot (dfh, TRUE)
plot (dfh, freq=TRUE)
plot (dFh, freq=TRUE)

dFht (0.5)
```

21_continuous_kernel_smoothing

Continuous Kernel Smoothing Models

Description

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

NOTE THAT THESE OBJECTS ARE LIKELY TO BE CONVERTED TO S4 OBJECTS, IN THE NEAR FUTURE.
ALSO, NOTE THAT THEIR INTERNAL STRUCTURE (THAT IS, THEIR ATTRIBUTES/SLOTS), IS SUBJECT TO CHANGE.

IN PRINCIPLE, YOU SHOULD NOT ACCESS ATTRIBUTES/SLOTS, DIRECTLY.

Usage

```
#univariate
pdfuv.cks (x, ..., w,
    bw, smoothness=1,
    kernel=BIWEIGHT.CKERNEL,
    spline=TRUE, bw.method="ph.default", nc=30,
    trtype="local",
    Xlim = cbind (a, b), a=-Inf, b=Inf)

cdfuv.cks (x, ..., w,
    bw, smoothness=1,
    kernel=BIWEIGHT.CKERNEL,
    spline=TRUE, bw.method="ph.default", nc=30, tail="lower",
    trtype="local",
    Xlim = cbind (a, b), a=-Inf, b=Inf)

qfuv.cks (x, ..., w,
    bw, smoothness=1,
    kernel=BIWEIGHT.CKERNEL,
    bw.method="ph.default", nc=30,
    trtype="local",
    Xlim = cbind (a, b), a=-Inf, b=Inf)

#multivariate
pdfmv.cks (x, ..., w,
    bw, smoothness=1,
    kernel=BIWEIGHT.CKERNEL,
    bw.method="ph.default",
    Xlim = cbind (a, b), a=-Inf, b=Inf)

cdfmv.cks (x, ..., w,
    bw, smoothness=1,
    kernel=BIWEIGHT.CKERNEL,
    bw.method="ph.default", tail="lower",
    Xlim = cbind (a, b), a=-Inf, b=Inf)

#conditional
pdfc.cks (x, ..., conditions, w,
    bw, smoothness=1,
```

```
    kernel=BIWEIGHT.CKERNEL,
    spline=TRUE, bw.method="ph.default", nc=30,
    Xlim = cbind (a, b), a=-Inf, b=Inf,
    preserve.range=FALSE, as.cset=FALSE, as.list.cset=FALSE,
    warning=TRUE)

cdfc.cks (x, ..., conditions, w,
    bw, smoothness=1,
    kernel=BIWEIGHT.CKERNEL,
    spline=TRUE, bw.method="ph.default", nc=30, tail="lower",
    Xlim = cbind (a, b), a=-Inf, b=Inf,
    preserve.range=FALSE, as.cset=FALSE, as.list.cset=FALSE,
    warning=TRUE)

qfc.cks (x, ..., conditions, w,
    bw, smoothness=1,
    kernel=BIWEIGHT.CKERNEL,
    bw.method="ph.default", nc=30,
    Xlim = cbind (a, b), a=-Inf, b=Inf,
    preserve.range=FALSE, as.cset=FALSE, as.list.cset=FALSE,
    warning=TRUE)

pdfmvc.cks (x, ..., conditions, w,
    bw, smoothness=1,
    kernel=BIWEIGHT.CKERNEL,
    bw.method="ph.default",
    Xlim = cbind (a, b), a=-Inf, b=Inf,
    preserve.range=FALSE, as.cset=FALSE, as.list.cset=FALSE,
    warning=TRUE)

#multivariate-conditional
cdfmvc.cks (x, ..., conditions, w,
    bw, smoothness=1,
    kernel=BIWEIGHT.CKERNEL,
    bw.method="ph.default", tail="lower",
    Xlim = cbind (a, b), a=-Inf, b=Inf,
    preserve.range=FALSE, as.cset=FALSE, as.list.cset=FALSE,
    warning=TRUE)

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

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.

`conditions` SIMPLE USE:
(SINGLE SET OF CONDITIONS)
A numeric vector of conditioning values.

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. MULTIPLE SETS OF CONDITIONS: A numeric matrix of conditioning values. Names are matched, same as in the simple use, except using column names.
Each row defines one set of conditions.

Note that in univariate-conditional distributions, the number of conditioning variables needs to equal the total number of variables minus one.
By default (depending on the as.cset argument), a cks object is returned if there's one set of conditions, and a cset object is returned, if there's two or more sets conditions).

`w` Optional numeric vector of weights.

`bw` Postive numeric vector of length [1 or m], the bandwidth parameter(s).
If bw is missing, the bandwidth is computed for each variable using the bandwidth method (see bw.method below), which is subject to the smoothness parameter(s).

`smoothness` Positive numeric vector of length [1 or m], the relative smoothness parameter(s).
Ignored, if bw provided

`kernel` A (continuous) kernel object.

`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.

`nc` Integer, number of control points, in the spline.
Ignored, if spline is false.

`tail` Character vector of length [1 or M], either "lower" or "upper".
If lower (the default), lower tail probabilities, P (..., Xj <= xj, ...), are used.
If upper, upper tail probabilities, P (..., X >= xj, ...), are used.

`trtype` String, either "simple", "local" or "reflect", refer to vignette.
Note that the local method is used, where there's no trtype argument.

`Xlim` An ([1 or m] by 2) numeric matrix, giving the limits of each X variable.
The corresponding random variable is regarded as bounded, if either limit is finite.
In which case, a truncated smoothing algorithm is applied.

`a, b` Numeric vectors of length [1 or m].
This is an alternative way of specifying Xlim, above.

`preserve.range`	Logical vector of length [1 or M]. 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.
`as.cset`	Logical, if true, a cset object is returned regardless of the number of conditions. Note that this is ignored, if there is not one set of conditions. Also note that as.list.cset needs to be true, if as.cset is true, or the number of sets of conditions is not one. (This requirement is temporary).
`as.list.cset`	Logical, refer to as.cset, above.
`warning`	Logical, if true, generate warning if there's no observations within the conditional window.
`...`	Additional arguments not allowed

Details

PLEASE SET NOTES IN DESCRIPTION FIELD.

Here, M refers to the number of random variables (in the model), and m refers to the total number of variables.
In nonconditional models, these are the same.
In univariate-conditional models, M is one and m is equals to the number of conditions plus one.

Variable names use default values, if the x matrix is unnamed.
(In conditional models, a warning is generated).

In conditional models, variables are reordered (internally), if there's named conditions.
Note you can check names (and their order), with names method.
(By default, only the conditional variables are returned).

Parameters that need to match m (the total number of variables), such as bw, smoothness and Xlim, will match the original data, regardless of the order of the conditions.
i.e. The first bw value will match the first column in x, regardless of the order of the conditions.

However, parameters that need to match M (the number of random variables), such as tail, need to match the order of the resulting conditional variables.
The same principle, applies to the x argument in the resulting function objects.

Refer to the vignette for more information.

Value

Self-referencing function objects.

Refer to Runtime 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).

They may also return cset objects, if as.cset is true, or there's more than one set of conditions.

References

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

See Also

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, ph.printf.phmodel, ph.plotf.cksuv, ph.plotf.cksmv

Bandwidth Selection

Examples

```
prep.ph.data ()

############
#univariate
############
uvfh <- pdfuv.cks (height)
uvFh <- cdfuv.cks (height)
uvFht <- qfuv.cks (height)

plot (uvfh)
plot (uvFh)
plot (uvfh, TRUE)

uvFht (c (0.25, 0.5, 0.75) )
ph.mode (uvfh)
ph.mode (uvfh, TRUE)

#############
#multivariate
#############
mvfh <- pdfmv.cks (trees2 [,-2])
mvFh <- cdfmv.cks (trees2 [,-2])

plot (mvfh, TRUE)
plot (mvfh,, TRUE)
plot (mvFh, TRUE)
plot (mvFh, fb = c (0.2, 0.8), clabs = c ("A", "B") )

##############
#conditional
#(single model)
###############
cons.c1 <- c (height=25, girth=35)
cfh <- pdfc.cks (trees2, conditions=cons.c1)
cFh <- cdfc.cks (trees2, conditions=cons.c1)
```

```
cFht <- qfc.cks (trees2, conditions=cons.c1)

plot (cfh)
plot (cFh)
plot (cfh, TRUE)

cFht (c (0.25, 0.5, 0.75) )
ph.mode (cfh)
ph.mode (cfh, TRUE)

#############
#conditional
#(two models)
#############
cons.c2 <- cbind (height=25:26, girth=35:36)

cfhs <- pdfc.cks (trees2, conditions=cons.c2, as.list.cset=TRUE)

plot (cfhs [[1]])

########################
#multivariate-conditional
########################
cons.mvc <- c (depth=311)

mvcFh <- cdfmvc.cks (quakes2, conditions=cons.mvc)
```

22_categorical_distributions
Categorical Models

Description

Fit categorical distributions, from data.

NOTE THAT THESE OBJECTS ARE LIKELY TO BE CONVERTED TO S4 OBJECTS, IN THE NEAR FUTURE.
ALSO, NOTE THAT THEIR INTERNAL STRUCTURE (THAT IS, THEIR ATTRIBUTES/SLOTS), IS SUBJECT TO CHANGE.

IN PRINCIPLE, YOU SHOULD NOT ACCESS ATTRIBUTES/SLOTS, DIRECTLY.

Usage

```
#univariate
pmfuv.cat (g, h=1)
cdfuv.cat (g, h=1)
qfuv.cat (g, h=1)

#conditional
```

```
pmfc.cat (g, h=1, ..., conditions, warning=TRUE)
cdfc.cat (g, h=1, ..., conditions, warning=TRUE)
qfc.cat (g, h=1, ..., 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). It's length should be one or n, 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.
warning	Logical, if true, generate warning if there's no observations within the conditional window.
...	Additional arguments not allowed.

Details

PLEASE SET NOTES IN DESCRIPTION FIELD.

Refer to the vignette for more information.

Default variable names are generated, if the g list is unnamed.
(In conditional models, a warning is generated).

Value

Self-referencing function objects.

Refer to Runtime 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. e.g. If unique g values were 10, 11 and 12, in contrast to 1, 2 and 3.

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, ph.printf.phmodel, ph.plotf.catuv

Examples

```
prep.ph.data ()

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

plot (gfh, freq=TRUE)
ph.mode (gfh)
ph.mode (gfh, level.names=TRUE)

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

23_mixed_conditional *Conditional Distributions with Mixed Categorical-Continuous Input*

Description

Fit conditional categorical or continuous distributions with mixed categorical-continuous input.
(These resemble conditional CAT and CKS models).
And the categorical distributions can be used for statistical classification purposes.

NOTE THAT THESE OBJECTS ARE LIKELY TO BE CONVERTED TO S4 OBJECTS, IN THE NEAR FUTURE.
ALSO, NOTE THAT THEIR INTERNAL STRUCTURE (THAT IS, THEIR ATTRIBUTES/SLOTS), IS SUBJECT TO CHANGE.

IN PRINCIPLE, YOU SHOULD NOT ACCESS ATTRIBUTES/SLOTS, DIRECTLY.

Usage

```
#conditional categorical
ph4.pmfc.gmix (g, x, ..., conditions, warning=TRUE, w)
ph4.cdfc.gmix (g, x, ..., conditions, warning=TRUE, w)
ph4.qfc.gmix (g, x, ..., conditions, warning=TRUE, w)
```

```
#conditional continuous
ph4.pdfc.xmix (g, x, ..., conditions, warning=TRUE, w)
ph4.cdfc.xmix (g, x, ..., conditions, warning=TRUE, w)
ph4.qfc.xmix (g, x, ..., conditions, warning=TRUE, w)
```

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.
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

PLEASE SET NOTES IN DESCRIPTION FIELD.

Refer to the vignette for more information.

Default variable names are generated, if the g/x list/matrix are unnamed.
(And a warning is generated).

Note that categorical and continuous variables need different names.

Value

Self-referencing function objects.

Refer to Runtime 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 you can check names (and their order), with names method.
This may be useful for conditional distributions.
(By default, only the conditional variables are returned).

Note

WARNING:

If a conditional categorical distribution is constructed with integer g values, the category indices won't necessarily equal the category names. e.g. If unique g values were 10, 11 and 12, in contrast to 1, 2 and 3.

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, ph.printf.phmodel, ph.plotf.catuv

Examples

```
prep.ph.data ()

fh.gmix <- ph4.pmfc.gmix (species, cbind (sepal.length, sepal.width),
    conditions = c (sepal.length=6, sepal.width=3) )
Fht.gmix <- ph4.qfc.gmix (species, cbind (sepal.length, sepal.width),
    conditions = c (sepal.length=6, sepal.width=3) )

plot (fh.gmix)
ph.mode (fh.gmix)
ph.mode (fh.gmix, level.names=TRUE)

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

`24_empirical-like_distributions` *Empirical-Like Models*

Description

Fit empirical-like probability distributions, from data.

NOTE THAT THESE OBJECTS ARE LIKELY TO BE CONVERTED TO S4 OBJECTS, IN THE NEAR FUTURE.
ALSO, NOTE THAT THEIR INTERNAL STRUCTURE (THAT IS, THEIR ATTRIBUTES/SLOTS), IS SUBJECT TO CHANGE.

IN PRINCIPLE, YOU SHOULD NOT ACCESS ATTRIBUTES/SLOTS, DIRECTLY.

Usage

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

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.
`...`	Additional arguments not allowed.

Details

PLEASE SET NOTES IN DESCRIPTION FIELD.

Refer to the vignette for more information.

Value

Self-referencing function objects.

Refer to Runtime 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, ph.printf.phmodel, ph.plotf.eluv

Examples

```
prep.ph.data ()

eFht <- qfuv.el (height)
eFht (c (0.25, 0.5, 0.75) )
```

`30_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 selection 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

```
prep.ph.data ()
auto.cbw (trees2)
```

31_distribution_sets *Distribution Sets*

Description

Sets of distributions.
Note that distributions sets can also be produced by conditional CKS models.

NOTE THAT THESE OBJECTS ARE SUBJECT TO CHANGE, AND HAVE HAD LIMITED TESTING.

Usage

```
#categorical sets
#(grouped by a categorical variable)
ph4.pdfuv.gset.cks (g, x, ...)
ph4.cdfuv.gset.cks (g, x, ...)
ph4.qfuv.gset.cks (g, x, ...)
ph4.cdfuv.gset.el (g, x, ...)
ph4.qfuv.gset.el (g, x, ...)

#marginal sets
ph4.pdfuv.mset.cks (x, ..., bw, smoothness=1)
ph4.cdfuv.mset.cks (x, ..., bw, smoothness=1)
ph4.qfuv.mset.cks (x, ..., bw, smoothness=1)
ph4.cdfuv.mset.el (x, ...)
ph4.qfuv.mset.el (x, ...)

ph4.pdfmv.gset.cks (g, x, ...)
```

Arguments

g	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.
x	Vector (for categorical sets) or matrix (for marginal sets).
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 constructor.

Details

PLEASE SET NOTES IN DESCRIPTION FIELD.

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

Value

A ph4.gset or ph4.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

ph.plotf.ph4.gset, ph.plotf.ph4.mset

Examples

```
prep.ph.data ()

gs <- ph4.pdfuv.gset.cks (species, sepal.length)
ms <- ph4.qfuv.mset.el (trees2)

names (gs)
plot (gs)
plot (ms, nr=2, nc=2)

#distance matrix
pdist (gs)

plot (gs [[1]])
```

40_as_methods *Convert Distribution Sets to Lists*

Description

Covert dset objects to lists.
(Refer to Distribution Sets).

You should wrap a dset object inside the function.
(An exception is plotting).

Usage

```
## S3 method for class 'dset'
as.list(x, ...)
```

Arguments

x	A dset (distribution set) object.
...	Ignored.

Value

A standard list.

References

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

41_print_methods *Print Methods*

Description

Print methods for objects in this package.

Note that this function is likely to change in the next update.

Usage

```
## S3 method for class 'Kernel'
ph.printf(k, ...)
## S3 method for class 'phmodel'
ph.printf(sf, ...)
## S3 method for class 'dset'
ph.printf(vf, ...)
```

Arguments

k	A kernel object.
sf	An object extending phmodel, which is most of the objects in this package.
vf	A list of probability distributions.
...	Ignored.

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

```
prep.ph.data ()

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

#autoprinting
dfh
```

42_kernel_plot_methods

Kernel Plot Methods

Description

Plot methods for kernel objects.

Usage

```
###################################
#discrete kernels
#(call plot_dpd)
###################################
## S3 method for class 'DKernel'
ph.plotf(dk, ..., cdf=FALSE)

###################################
#continuous kernels
#(call plot_cpd)
###################################
## S3 method for class 'CKernel'
ph.plotf(ck, ..., cdf=FALSE)
```

Arguments

dk, ck	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

Kernels

plot_kernel_array

plot_dpd, plot_cpd

Examples

```
dk <- discretized.kernel (7)
plot (dk)

plot (BIWEIGHT.CKERNEL)
```

43_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'
ph.plotf(sf, data=FALSE, ...)

###################################
#continuous kernel smoothing models
#(call plot_cpd, plot_cpd_bv or plot_cpd_tv)
###################################
## S3 method for class 'cksuv'
ph.plotf(sf, data=FALSE, ...)
## S3 method for class 'cksmv'
```

```
ph.plotf(sf, in3d=FALSE, data=FALSE, ...)
## S3 method for class 'cksc'
ph.plotf(sf, ...)
## S3 method for class 'cksmvc'
ph.plotf(sf, in3d=FALSE, data=FALSE, ...)

####################################
#categorical models
#(call plot_dpd)
####################################
## S3 method for class 'catuv'
ph.plotf(sf, ...)
## S3 method for class 'catc'
ph.plotf(sf, ...)

####################################
#mixed input
####################################
## S3 method for class 'gmix'
ph.plotf(sf, ...)
## S3 method for class 'xmix'
ph.plotf(sf, ...)

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

####################################
#all continuous univariate models
####################################
## S3 method for class 'cpduv'
ph.linesf(sf, ..., xlim, n=200)
```

Arguments

sf	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
xlim	Length two numeric vector, giving plot range. Currently, ignored for quantile functions.
n	Integer, number of points.

... 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

```
prep.ph.data ()

dfh <- pmfuv.dks (traffic.bins, traffic.freq)
cfh <- pdfuv.cks (height)
cfh2 <- pdfmv.cks (trees2 [,-2])

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

44_other_plot_methods *Other Plot Methods*

Description

Plot methods for distribution sets.

NOTE THAT THESE FUNCTIONS WILL BE REPLACED.

Usage

```
## S3 method for class 'ph4.gset'
ph.plotf(sfs, ..., span.win=FALSE, legend=TRUE, colors)
## S3 method for class 'ph4.mset'
ph.plotf(sfs, ..., nr, nc, colors)
```

Arguments

sfs	A ph4.gset or ph4.mset object.
span.win	Logical, if true, plots (most notably CDFs) extend across the entire plotting window.
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

```
prep.ph.data ()

plot (ph4.pdfuv.gset.cks (species, sepal.length) )
plot (ph4.qfuv.mset.el (trees2), nr=2, nc=2)
```

45_range_and_sequence_methods

Range and Sequence Functions

Description

Range and sequence methods for probability distributions.

Usage

```
## S3 method for class 'dpd'
min(sf, infv=FALSE, ..., freq=FALSE, n)
## S3 method for class 'dpd'
max(sf, infv=FALSE, ..., freq=FALSE, n)
## S3 method for class 'dpd'
range(sf, infv=FALSE, ..., freq=FALSE, n)

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

```
range(sf, infv=FALSE, ...)

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

Arguments

sf	A suitable function object. For min/max/range functions, this is any probability distribution (except chqf.cks), from this package. For seq functions, this is any univariate probability distribution, from the package. 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/CDFs with infv=TRUE.
n	An integer. In the discrete case, it represents the sample size, and is ignored unless both infv and freq are TRUE. (If missing, it defaults to the number of observations used, or the sum of their unscaled weights/frequencies). In the continuous case, it represent 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.

In the univariate case:
The min and max methods return a single integer/numeric value.
The range methods return an length-two integer/numeric vector.

And in the multivariate case:
The min and max methods return an integer/numeric vector.
The range methods return a two column integer/numeric matrix.

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

```
prep.ph.data ()

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

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

50_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 = is.dks (sf), freq=FALSE, n, 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. Currently, ignored in quantile functions.
n	An integer, representing the sample size, and is ignored except for PMFs/CDFs with freq set to TRUE. (If missing, it defaults to the number of observations used, or the sum of their frequencies).
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

ph.plotf.dksuv, ph.plotf.catuv

Examples

```
prep.ph.data ()

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

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

51_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,
    n=200)
```

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.
n	Integer, number of (line) points.
...	Ignored.

References

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

See Also

set.ph.options

ph.plotf.cksuv, ph.plotf.eluv

plot_cpd_bv, plot_cpd_tv

Examples

```
prep.ph.data ()

cfh <- pdfuv.cks (height)

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

52_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, ..., n=30,
    main, xlab, ylab, zlab,
    xlim, ylim, zlim,
    add=FALSE, point.color)

#calls barsurf::plot_cfield_3d
plot_cpd_tv (sf, iso=FALSE, ...,
    main, xlab, ylab, zlab,
    xlim, ylim, zlim,
    z.reverse=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.
iso	Logical, if true, produce an isosurface plot.

`data`	Logical, if true, plot data points. Ignored, if in3d or add is true, or sf is a conditional distribution. Defaults to true, if the number of observations is not more than 2000.
`main, xlab, ylab, zlab`	Optional strings, main/axes titles. Note that these depend on the barsurf package.
`xlim, ylim, zlim`	Optional length-2 numeric vectors, giving the plot ranges.
`z.reverse`	Logical, if true, reverse the z axis. Ignored, if zlim supplied.
`n`	Numeric vector of length one or two, giving the number of grid points in each direction.
`add`	Logical, if true, add to a previous plot. Ignored, in 3d case.
`point.color`	String, the (data) point color.
`...`	Other arguments for barsurf functions, refer to details.

Details

These functions call barsurf::plot_cfield, barsurf::plot_surface, barsurf::plotf_cfield_3d and barsurf::plotf_isosurface.

There are some private intermediate functions, that change some of the defaults.
Note that by default, more detail is used in bivariate CDFs than bivariate PDFs.

References

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

See Also

set.ph.options

plot_cpd

ph.plotf.cksmv

Examples

```
prep.ph.data ()

cfh2 <- pdfmv.cks (trees2 [,-2])

plot (cfh2)
plot (cfh2, TRUE)
plot (cfh2,, TRUE)
```

53_pairwise_kernel_arrays *Pairwise Kernel Arrays*

Description

Plots of pairwise kernel arrays.

Usage

```
list.ckernels ()

plot_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 plot_kernel_array function plot pairs of kernels, for comparison purposes.

References

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

See Also

Kernels

ph.plotf.DKernel, ph.plotf.CKernel

Examples

```
plot_kernel_array ()
```

`60_distance_matrices` *Distance Matrices*

Description

Compute the distance (or dissimilarity) between pairs of density functions.

The main function, pdist maps a list (of density functions) to a matrix.
The rdist function, converts the above matrix to a two-column sorted data.frame.

Note that these methods are relatively new.
I can not give any guarantee of optimality.
(This applies to the whole package, but in particular to these functions).

Also, the distance values should be not be interpreted, other than for ranking purposes.

Usage

```
psv (sf)

pdist (sf, ..., sqrt.mse=TRUE)

ph4.rdist (d, n)

ph4.pcomp2 (fh, gh, sqrt.mse=TRUE, aggregate=TRUE,
    dfh = psv (fh), dgh = psv (gh) )
```

Arguments

`fh, gh`	A density function. Refer to pdfuv.cks and pdfmv.cks.
`sf`	In psv, a density function. In pdist, a list of density functions. Optionally, they can be produced by ph4.pdfuv.gset.cks and ph4.pdfmv.gset.cks.
`aggregate`	If true (the default), return the average of the two one-sided distances.
`sqrt.mse`	If true (the default), the square root of the MSE is used, otherwise, the MSE is used.
`dfh, dgh`	Numeric vectors, of self-evaluated densities, see details. Note that no validation is done on these arguments.
`d`	A distance matrix, as returned by pdist.
`n`	Integer, the closest n pairs. If missing, all are returned.
`...`	Ignored.

Details

Here, self-evaluated density values are computed by evaluating a density function at it's own data. (In contrast to arbitrary evaluation points).

And cross-evaluated densities are values are computed by evaluating a density function at another density function's data.

The psv function computes self-evaluated densities.
The ph.pcomp2 function computes one (or two) distances.
With a single distance between the average of two one-sided distances.

If dfh-xf is the self-evaluated density of fh, and dgh-xf is the density of gh evaluated at fh's data: (i.e. Two density functions are evaluated at the same points, which are the data points of the first density function).

Then a one-sided distance can be computed as the mean squared error (MSE), or it's square root. Where the MSE is the sum of the squared differences between dfh-xf and dgh-xf, over n.

The second one-sided distance is the same, except that f and g are reversed.

Value

psv returns a numeric vector.

pdist returns a numeric square matrix.

By default, ph4.pcomp2 returns a single value.
(If aggregate is false, then it returns a pair of values).

References

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

See Also

pwith, probmv

pdfuv.cks, pdfmv.cks
ph4.pdfuv.gset.cks, ph4.pdfmv.gset.cks

Examples

```
prep.ph.data ()

gs1a <- ph4.pdfuv.gset.cks (species, cbind (sepal.length) )
gs1b <- ph4.pdfuv.gset.cks (species, cbind (sepal.width) )
gs2 <- ph4.pdfmv.gset.cks (species, cbind (sepal.length, sepal.width) )

d1a <- pdist (gs1a)
d1b <- pdist (gs1b)
d2 <- pdist (gs2)

#print out distance matrix
#(for bivariate models)
d2
```

```
#print out distances, ranked/sorted
ph4.rdist (d1a)
ph4.rdist (d1b)
ph4.rdist (d2)
```

61_main_multivariate_probabilities
Multivariate Probabilities

Description

Compute the probability of observing a point within a rectangular/cuboidal/etc region.

Currently, these functions are limited to nonconditional CKS models.
The function probmv can be used for almost-arbitrary continuous CDFs, but is deprecated.

Usage

```
pwith (...)

## S3 method for class 'cksuv'
pwith(sf, xlim = c (a, b), ..., a=-Inf, b=Inf)
## S3 method for class 'cksmv'
pwith(sf, xlim = cbind (a, b), ..., a=-Inf, b=Inf)
```

Arguments

sf	A suitable function object. (Here, this is a cksuv or cksmv object, which are nonconditional CKS objects.) PDFs, CDFs and quantile functions, are all allowed.
xlim	A length-two numeric vector or a two-column numeric matrix. A standard vector is the same as a one-row matrix. And the number of rows needs to match the number of random variables. This represents the lower and upper limits, corresponding to the limits of integration, if one was integrating the corresponding PDF. By default, this is just the combination a and b, meaning that limits can be specified in two ways.
a, b	Numeric vectors, giving the lower and upper limits, respectively.
...	Ignored.

Details

Refer to the vignette for more information.

Note that evaluating a spline-based CDF is likely to faster, ignoring the construction of the object.

Value

A single numeric value.

References

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

See Also

pdist, probmv

cdfmv.cks, cdfmvc.cks

ph.mean
moment

rng

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

Examples

```
prep.ph.data ()

cfh3 <- pdfmv.cks (trees2)

xlim <- 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") ) )

pwith (cfh3, xlim=xlim)
```

62_other_multivariate_probabilities
Multivariate Probabilities

Description

Compute probabilities, from multivariate CDFs.

THIS FUNCTION IS DEPRECATED, BUT YOU STILL MAY USE.
THE pwith FUNCTION IS PREFERABLE FOR NONCONDITIONAL CKS MODELS

Usage

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

Arguments

sf	A cdfmv.cks or cdfmvc.cks object. (Lower tail only).
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

```
prep.ph.data ()

cFh3 <- cdfmv.cks (trees2)

xlim <- 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, xlim [,1], xlim [,2])
```

63_probability_matrices *Bivariate Density/etc Matrices*

Description

Convenience function to construct an S4 object representing a bivariate density/etc matrix from a bivariate PDF/CDF.

Unlike other objects in the package, you're welcome to access the slots.

Usage

```
ph4.BVMatrix (sf, xlim, ylim, ..., n=10)
```

Arguments

sf	A suitable function object. (Here, this refers to a bivariate PDF/CDF).
xlim, ylim	Numeric length two vectors, giving the evaluation ranges.
n	Integer vector of length one or two, giving the number of points, in each x and y direction.
...	Ignored.

Value

An S4 object with three slots, fv (a matrix), x (x evaluation points) and y (y evaluation points).

Each row in fv, corresponds to one x value, and each column to one y value.

References

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

Examples

```
prep.ph.data ()

cfh2 <- pdfmv.cks (trees2 [,-2])
ph4.BVMatrix (cfh2)
```

64_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

```
prep.ph.data ()

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

chFht <- chqf.cks (trees2)
rng (chFht, 30)

rng (height, 30)
```

`70_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

```
prep.ph.data ()

cFh <- cdfuv.cks (height)

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

71_quantile-based_summaries

Order-Based Summary Statistics

Description

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

Usage

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

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

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

Arguments

n	Integer, the number of sequential quantiles.
symbol	String, letter/symbol for the quantile names.
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.
prob	Logical, if false (the default), name quantiles by index/names, if true, 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

```
prep.ph.data ()

cFht <- qfuv.cks (height)
quartiles (cFht)
quartiles (cFht, prob=TRUE)

quartiles (height)
```

72_quantile-based_statistics

Robust Statistics

Description

Compute robust (quantile based) 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

```
prep.ph.data ()

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

ph.median (cFht)
#iqr (cFht)
```

73_mode_estimation *Mode Estimation*

Description

Compute the mode or modes, from probability distributions.

Usage

```
ph.mode (sf, infv=FALSE, ..., level.names=FALSE, freq, n)
ph.modes (sf, infv=FALSE)
```

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).
level.names	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.
n	Integer, the sample size. Ignored, unless both infv and freq are true.
...	Ignored.

Value

ph.mode returns a single integer or numeric value.
(Except for categorical PMFs with infv=FALSE and level.names=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

```
prep.ph.data ()

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

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

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

80_runtime_function_objects
Runtime Function Objects

Description

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

DO NOT CALL THESE FUNCTIONS.
CALL A CONSTRUCTOR, WHICH SHOULD RETURN A FUNCTION OBJECT.
THE RESULTING FUNCTION OBJECTS SHOULD HAVE THE SAME ARGUMENTS AS BELOW.

Usage

```
#categorical models
#(CAT/gMIX)
ph.gfh.rtf (g, ..., freq=FALSE, n)      #PMF
ph.gFh.rtf (g, ..., freq=FALSE, n)      #CDF
ph.gFht.rtf (p, ..., level.names=FALSE) #QF

#other discrete models
#(DKS)
ph.dfh.rtf (x, ..., freq=FALSE, n)      #PMF
ph.dFh.rtf (x, ..., freq=FALSE, n)      #CDF
ph.dFht.rtf (p)                         #QF

#univariate continuous models
#(CKS/EL/xMIX, UV/C)
ph.cfh.rtf (x)                          #PDF
ph.cFh.rtf (x)                          #CDF
ph.cFht.rtf (p)                         #QF

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

#chained quantile functions
ph.chFht.rtf (p)
```

Arguments

g, x	IN (DKS) MODELS: An integer vector, of quantiles. IN (CAT/gMIX) MODELS:

	An integer/factor/character vector, of quantiles. Integers represent category indices. Characters and formatted factors represent category names. IN (CKS/EL/xMIX, 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.
level.names	Logical, if true, return category names rather than category indices. Refer to the value section.
n	Sample size. Ignored, unless freq is true.
...	Ignored.

Details

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

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

Also, if a bounded interval was used, then the quantiles need to be within the limits.

Value

By default:
PMFs return a numeric vector, giving probability mass.
PDFs return a numeric vector, giving probability density.
CDFs return a numeric vector, giving cumulative probability (from zero to one).
Discrete quantile functions, return an integer vector of quantiles.
Continuous quantile functions, return a numeric vector of quantiles.

In the discrete case, setting freq=TRUE, scales the values to match the n, the sample size.
By default, n is the number of observation used, or the sum of the unscaled weights/frequencies.

In categorical quantile functions, setting level.names=TRUE, returns a character vector of levels names.

Chained quantile functions, 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

`90_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")
```

91_other_functions *Other Functions*

Description

Other functions.
The prep.ph.data function is mainly for examples.

Usage

```
prep.ph.data (..., eval=TRUE, echo=FALSE)
min1 (x)
```

Arguments

eval	Logical, if true, run the script.
echo	Logical, if true, echo the script.
x	An integer vector.
...	Ignored.

Details

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

The min1 function will return one, if one is the minimum x value, otherwise, it will return zero.

References

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

Examples

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

92_temp_and_deprecated *Temporary/Deprecated Functions*

Description

PLEASE DO NOT USE, THESE ARE LIKELY TO BE REMOVED.

Usage

```
object %$% name
object %$% name <- value
```

Arguments

object, name, value
: .

Index