

Package ‘ipptoolbox’

April 17, 2009

Type Package

Title IPP Toolbox

Version 1.0

Date 2008-11-04

Author Philipp Limbourg <p.limbourg@uni-due.de>

Depends R (>= 2.7.0), AlgDesign, copula, evd, triangle

Maintainer Philipp Limbourg <p.limbourg@uni-due.de>

Description An R package for uncertainty quantification and propagation in the framework of Dempster-Shafer Theory and imprecise probabilities. This package was developed in the context of a collaborative research project between EDF R&D (<http://rd.edf.com>) and the University of Duisburg-Essen, Information Logistics (<http://www.uni-due.de/il>).

License GPL (>= 2)

LazyLoad yes

Repository CRAN

Date/Publication 2008-11-07 14:34:38

R topics documented:

| | |
|------------------------------|----|
| ipptoolbox-package | 2 |
| aggregation | 3 |
| basicfunctions | 4 |
| bpafromdata | 5 |
| bpafromdist | 7 |
| bpafromdist | 8 |
| bpafromprobs | 9 |
| examples | 11 |
| plotting | 12 |
| propsens | 13 |
| statprops | 16 |

ipptoolbox-package *IPP Toolbox*

Description

An R package for uncertainty quantification and propagation in the framework of Dempster-Shafer Theory and imprecise probabilities.

This package was developed in the context of a collaborative research project between EDF R&D (<http://rd.edf.com>) and the University of Duisburg-Essen, Information Logistics (<http://www.uni-due.de/il>).

Details

| | |
|-----------|------------------------|
| Package: | ipptoolbox |
| Type: | Package |
| Version: | 1.0 |
| Date: | 2008-11-04 |
| License: | GPL version 2 or newer |
| LazyLoad: | yes |

The package contains a set of methods for conducting uncertainty studies using imprecise probabilities. Please type `help(dsfcruesexample)` for an example.

Author(s)

Philipp Limbourg <p.limbourg@uni-due.de>

Maintainer: Philipp Limbourg <p.limbourg@uni-due.de>

References

Ferson, S., V. Kreinovich, et al. (2003). Constructing Probability Boxes and Dempster-Shafer Structures. Sandia Report. Albuquerque, USA, Sandia National Laboratories.

Limbourg, P. (2008). Dependability Modelling under Uncertainty. Berlin, Heidelberg, Springer.

Examples

```
print("Example Fcrues")
print("See code")
print(dsfcruesexample)
print("Execute example")
dsfcruesexample()
```

aggregation *Aggregation*

Description

Various functions to aggregate a set of estimated BPAs to a single aggregation.

Usage

```
dsdempstersrule(..., maxfocals = 1e+07)
dsenvelope(...)
dsintersect(...)
dssaveintersect(..., method = "averaging")
dsxaveraging(..., w = NULL, maxfocals = 1e+07)
dsxenvelope(..., maxfocals = 1e+07)
dswavg(..., w = NULL)
dswmix(..., w = NULL)
```

Arguments

| | |
|------------------------|---|
| <code>...</code> | 1...n BPAs to be aggregated. |
| <code>w</code> | Vector of n weights for weighted aggregations |
| <code>maxfocals</code> | Maximal number of focal elements of the result. Warning: If not constrained, some rules use time and memory $O(m ^{ x })$, <code>lml</code> number of focals by structure, <code>lxl</code> number of structures. Meaning: 5 structures with 1000 focals: Maximal 10^{15} bytes |
| <code>method</code> | <code>method="averaging"</code> : In case of empty intersection, focal elements will be averaged. <code>method="averaging"</code> : In case of empty intersection, focal elements will be enveloped. |

Details

A set of functions to aggregate a set of estimated BPAs to a single aggregation. Some functions have an optional `maxfocals` parameter that limits the size of the resulting BPA (which theoretically is the product of all focal elements). The vector `w` contains weights in case of a weighted aggregation type. The `dssaveintersect` method has an optional parameter that specifies the alternative rule in case of empty intersections.

In the example, two p-boxes defined from exponential distributions are aggregated using `dsdempstersrule` and `dswmix`.

Value

Aggregation of the BPAs according to the selected rule.

Author(s)

Philipp Limbourg <p.limbourg@uni-due.de>

References

Sentz, K. and S. Ferson (2002). Combination of Evidence in Dempster-Shafer Theory. Sandia Report. Albuquerque, USA, Sandia National Laboratories.

Examples

```
lambda1=dsstruct(c(2,3,1))
dss1=dsodf('qexp',100,lambda1);
lambda2=dsstruct(c(5,6,1))
dss2=dsodf('qexp',100,lambda2);
y=dsdempstersrule(dss1,dss2);
y2=dswmix(dss1,dss2,w=c(2,0.5));
dscdf(y);

dscdf(y2);
```

basicfunctions *Basic functions*

Description

Helpful functions for creating and handling BPAs

Usage

```
dsstruct(x)
dsnorm(y)
dsred(y, thres = 0.001)
```

Arguments

| | |
|-------|---|
| x | Matrix/BPA |
| y | BPA |
| thres | Minimal mass value of a focal element desired |

Details

dsstruct Creates a new BPA from a matrix and normalizes it.

dsnorm Normalizes a BPA (i. e. masses sum up to 1, no lower bound larger than higher bound).

dsred Reduces the number of focal elements in a BPA by merging adjoint elements.

The IPP Toolbox represents BPAs in a discretized form. A BPA is stored as a matrix (3 columns) that lists the focal elements. The first and second columns contain lower and upper bounds, the third column the masses of the focal elements. The function `dsstruct` is a simple function to create a valid BPA. It takes a three-column matrix of the above type as an input. It performs some checks (upper bound \geq lower bound, masses sum up to 1) and returns a valid BPA (possibly switching lower & upper bounds of focal elements, normalizing masses to 1) with the according warning messages.

Author(s)

Philipp Limbourg <p.limbourg@uni-due.de>

References

Ferson, S., V. Kreinovich, et al. (2003). Constructing Probability Boxes and Dempster-Shafer Structures. Sandia Report. Albuquerque, USA, Sandia National Laboratories.

Examples

```
a=dsstruct(matrix(c(2,3,0.1,1.5,5,0.9),ncol=3,byrow=TRUE))
b=dsstruct(matrix(c(2,3,0.5,10,5,0.9),ncol=3,byrow=TRUE))
c=dsodf('qexp',10000,dsstruct(c(10,20,1)));
d=dsred(c,0.02)
dscdf(d);
```

bpafromdata

Generating BPAs from point / interval data

Description

A set of functions to create empirical BPAs from data sets (either points or intervals).

Usage

```
dsecdfit(x)
dslapconf(x, lims = c(-Inf, Inf))
dskskonf(x, conf = 0.95, lims = c(-Inf, Inf))
```

Arguments

| | |
|------|--|
| x | Array of points / matrix with interval data |
| lims | Optional: Limits of the BPA for cutting first and last focal elements. |
| conf | Confidence level of Kolmogorov-Smirnov fit in]0,1[. |

Details

dsecdfit Creates a BPA of the data set analogous to probabilistic empirical CDF

dslapconf Creates a BPA of the data set according to the Laplace method

dskskonf Creates a BPA of the data set from the Kolmogorov-Smirnov bounds for a given confidence.

The toolbox contains a set of functions to create empirical BPAs from data sets. These data sets may be either standard data values or intervals (e.g. caused by measurement imprecision). dsecdfit

creates a BPA of the data set analogous to probabilistic empirical CDF. It is not conservative re-
spective to distribution tails. `dslapconf` creates a BPA according to the Laplace method, `dskskonf`
from the Kolmogorov-Smirnov bounds for a given confidence.

The example illustrates the three different methods on interval and point data. First, 20 random
values and intervals are generated. Then, `dsecdfit`, `dslapconf` and `dskskonf` are used to generate fits
for both data sets.

Value

BPA according to the chosen method.

Author(s)

Philipp Limbourg <p.limbourg@uni-due.de>

References

Ferson, S., V. Kreinovich, et al. (2003). Constructing Probability Boxes and Dempster-Shafer
Structures. Sandia Report. Albuquerque, USA, Sandia National Laboratories.

Examples

```
print("Create random data sets")
setofpoints=rnorm(20,0,1)
setofintervals=cbind(setofpoints-runif(20,0,1),setofpoints+runif(20,0,1))
print("Plot ecdf for both points and interval data")
ecdfp=dsecdfit(setofpoints)
dscdf(ecdfp)

ecdfi=dsecdfit(setofintervals)
dscdf(ecdfi)

print("Plot Laplace BPA for both points and interval data")
lapp=dslapconf(setofpoints,lim=c(-5,5))
dscdf(lapp)

lapi=dslapconf(setofintervals,lim=c(-5,5))
dscdf(lapi)

print("Plot Kolmogorov-Smirnov 75 percent bound BPA for both points and interval data")

ksp=dskskonf(setofpoints,conf=0.75,lim=c(-5,5))
dscdf(ksp)

ksi=dskskonf(setofintervals,conf=0.75,lim=c(-5,5))
dscdf(ksi)
```

`bpafromdist`*Construct pboxes*

Description

Methods to construct pboxes from distributions.

Usage

```
dsadf(fhandle, intervalnumber, ...)  
dsodf(fhandle, intervalnumber, ...)
```

Arguments

| | |
|-----------------------------|---|
| <code>fhandle</code> | Inverse CDF function (e. g. <code>qnorm</code>) |
| <code>intervalnumber</code> | Number of intervals of the resulting BPA |
| <code>...</code> | Parameters (either points or BPAs) for <code>fhandle</code> |

Details

`dsadf` Average discretization method

`dsodf` Outer discretization method (conservative regarding the discretization error)

The IPP Toolbox contains two methods to construct pboxes from distributions. The functions `dsadf` (average discretization method) and `dsodf` (outer discretization method) are used to sample a set of focal element from a parametric model. The parametric model must be given as an inverse cdf $F^{-1}(x)$ with precise or imprecise parameters. Both methods sample the provided inverse CDF function $F^{-1}(x)$ to generate a set of focal elements. `dsadf` samples points, e.g. 0.05, 0.15, 0.25, ..., 0.95, while `dsodf` samples intervals, e.g. [0,0.1], [0.1,0.2], ..., [0.9,1]. A BPA constructed by `dsodf` is more conservative and includes a BPA generated by `dsadf`.

The example generates several pboxes from normal distributions. It illustrates the difference of `dsadf` and `dsodf` by plotting BPAs with precise parameters of only 20 samples (`pbox1`, `pbox2`). Then it generates two pboxes with imprecise parameters.

Value

BPA representing a pbox sampled from `fhandle`.

Author(s)

Philipp Limbourg <p.limbourg@uni-due.de>

References

Tonon, F. (2004). "Using random set theory to propagate epistemic uncertainty through a mechanical system." *Reliability Engineering and System Safety* 85(1-3): 169-181.

Examples

```
print("Precise and imprecise parameters for qnorm")
mu=0; sigma=1
mu2=dsstruct(c(-0.5,0.5,1)); sigma2=dsstruct(c(1,2,1))
print("Pbox (distribution) of qnorm with precise parameters:")
pbox1=dsadf(qnorm,20,mu,sigma)
dscdf(pbox1)

print("Pbox of qnorm with precise parameters, outer discretization:")
pbox2=dsodf(qnorm,20,mu,sigma)
dscdf(pbox2)

print("Pbox of qnorm with imprecise mu, precise sigma:")
pbox3=dsodf(qnorm,1000,mu2,sigma)
dscdf(pbox3)

print("Pbox of qnorm with imprecise parameters:")
pbox4=dsodf(qnorm,1000,mu2,sigma2)
dscdf(pbox4)
```

bpafromdist

Construct pboxes

Description

Kolmogorov-Smirnov-Fits for BPAs.

Usage

```
dskstest(val, ds, ...)
```

Arguments

| | |
|-----|---|
| val | Data for K-S / C-M test |
| ds | BPA or handle of a pbox |
| ... | Optional pbox parameters (points, BPAs) |

Details

`dskstest` Evaluates the fit of a pbox using a Kolmogorov-Smirnov test

The method `dskstest` evaluates the fit of a pbox using a Kolmogorov-Smirnov test. The function works both on point values and interval data. The syntax is close to the R syntax for a probabilistic K-S test `ks.test()`. Both a BPA and a parametric pbox (in form of a CDF and a set of precise/imprecise parameters) can be tested.

The example generates a random set of points and a random set of intervals from a normal distribution (mean=0.3, sd=1.2). It tests, if a pbox with mean=[-0.5,0.5] and sd=1 fits the point data. In a second example, the user passes a normal CDF with parameters mean=0 and sd=[1,2] to the test. `dskstest` checks, if it fits the interval data.

Value

BPA representing a pbox sampled from `fhandle` / K-S test results.

Author(s)

Philipp Limbourg <p.limbourg@uni-due.de>

Examples

```
data=rnorm(100,0.3,1.2)
intervaldata=cbind(data,data+runiform(100,0,0.2))

mu=0; sigma=1
mu2=dsstruct(c(-0.5,0.5,1)); sigma2=dsstruct(c(1,2,1))
pbox=dsodf(qnorm,1000,mu2,sigma)

print("K-S result, pbox on data:")
ks=dskstest(data,pbox)
print(ks)
print("K-S result, precise mu, imprecise sigma on interval data:")
ks2=dskstest(intervaldata,pnorm,list(mu,sigma2))
print(ks2)
```

bpafromprobs

Obtain BPAs from statistical properties

Description

Various functions to obtain conservative BPA estimates from statistical properties such as mean, variance, ...

Usage

```
dsminmeanmax(intervalnumber, min, mean, max)
dsminmodemax(intervalnumber, min, mode, max)
dsmeanvar(intervalnumber, mean, var)
```

Arguments

| | |
|-----------------------------|--|
| <code>intervalnumber</code> | Number of focal elements of the discretization |
| <code>min</code> | Minimum of the BPA |
| <code>mean</code> | Mean of the BPA |
| <code>mode</code> | Mode of the BPA |
| <code>max</code> | Maximum of the BPA |
| <code>var</code> | Variance of the BPA |

Details

`dsminmeanmax` generates a BPA from min, mean and max.

`dsminmodemax` generates a BPA from min, mode and max.

`dsmeanvar` generates a BPA from mean and variance.

A set of functions to obtain a BPA that bounds all CDFs with given statistical properties (mean, variance...). These functions are especially useful for quantification of expert estimates. `dsminmeanmax` generates a BPA from min, mean and max, `dsminmodemax` from min, mode and max and `dsmeanvar` from mean and variance. It is necessary to provide an amount of focal elements to be generated (discretization accuracy).

The example generates 3 BPAs. The first bounds all distributions with `min=10`, `mode=60` and `max=100`. The second all with `min=10`, `mean=30` and `max=100`. The third all with `mean=30` and `var=5`.

Value

BPA with `intervalnumber` focal elements bounding all possible distributions with the given parameters.

Author(s)

Philipp Limbourg <p.limbourg@uni-due.de>

Examples

```
pmin=10
pmean=30
pmode=60
pmax=100
pvar=5
y1=dsminmodemax(100,pmin,pmode,pmax)
dscdf(y1);

y2=dsminmeanmax(100,pmin,pmean,pmax)
dscdf(y2);
```

```
y3=dsmeanvar(100,pmean,pvar)
dscdf(y3)
```

examples

Examples

Description

Two self-contained examples, see commented code.

Usage

```
dsfcruesexample()
dsexample()
```

Details

Please see the code of the examples.

Author(s)

Philipp Limbourg <p.limbourg@uni-due.de>

References

Limbourg, P., R. Savic, et al. (2007). Fault Tree Analysis in an Early Design Stage using the Dempster-Shafer Theory of Evidence. European Conference on Safety and Reliability - ESREL 2007, Stavanger, Norway, Taylor and Francis.

Examples

```
print("Example Fcrues")
print("See code")
print(dsfcruesexample)
print("Execute example")
dsfcruesexample()
```

plotting

*Plotting BPAs***Description**

Various routines to visualize BPAs

Usage

```
dscdf(x, xrange = NULL, col = c(3, 4), ..., newplot = TRUE)
dsqqplot(ds, sample, points = FALSE)
dsbel(x)
dspl(x)
```

Arguments

| | |
|----------------------|---|
| <code>x</code> | BPA to plot |
| <code>xrange</code> | Optional: <code>xrange=c(1,5)</code> limits the x-axis to [1,5] |
| <code>col</code> | Optional: <code>col = c(3,4)</code> defines the colors of the Bel and the PI line. |
| <code>...</code> | Optional: additional parameters passed to the plot routine (except linewidth, linetype and color) |
| <code>newplot</code> | Optional: Create a new plot or draw lines onto an existing plot. |
| <code>sample</code> | set of data values, e. g. <code>c(1.5,2,3.5)</code> |
| <code>ds</code> | BPA for qq plot resembling sample |
| <code>points</code> | Optional, <code>points=TRUE</code> : plot the sample values as points + line, otherwise only a straight line. |

Details

`dscdf` Plots $\text{Bel}([-\text{Inf},x])$ and $\text{Pl}([-\text{Inf},x])$. These functions bound all CDFs.

`dsqqplot` Plots a quantile-quantile plot between data and a BPA.

`dsbel` Returns $\text{Bel}([-\text{Inf},x])$, e. g. for own plotting routines.

`dspl` Returns $\text{Pl}([-\text{Inf},x])$, e. g. for own plotting routines.

The function `dscdf` plots $\text{Bel}([-\text{Inf},x])$ and $\text{Pl}([-\text{Inf},x])$, the bounds on all CDFs. Arbitrary graphics parameters can be passed to `dscdf`. `dsqqplot` plots a quantile-quantile plot between data and a BPA. `dsbel` and `dspl` return the function $\text{Bel}([-\text{Inf},x])/\text{Pl}([-\text{Inf},x])$ as a set of (x,y)-type points, e. g. for own plotting routines.

The example generates a BPA. It plots the BPA first without, then with additional parameters. Finally it uses the functions `dsbel` and `dspl` for plotting points into the diagram. Then it generates a BPA from a normal pbox with imprecise standard deviation. It tests with a QQ plot, if the function fits well to a random sample from $N(0,1)$.

Value

dscdf returns the x and y coordinates of the plotted lines for further use.

Author(s)

Philipp Limbourg <p.limbourg@uni-due.de>

Examples

```
print("Plot new BPA")
a=dsstruct(rbind(c(1,2,0.3),c(1.5,5,0.5),c(4,6,0.2)))
dscdf(a)

dscdf(a,xrange=c(0,7),xlab="Important parameter")
print("Bel([-Inf,x] ")
bel=dsbel(a)
print("Pl([-Inf,x] ")
pl=dspl(a)
points(pl,lwd=5,col='red')
points(bel,lwd=5,col='pink')

print("See if a pbox from N(0,[0.5,1.5]) fits data from N(0,1)")
data=rnorm(100,0,1)
pbox=dsodf(qnorm,1000,0,dsstruct(c(0.5,1.5,1)))
print("Plot pbox")
dscdf(pbox)

print("qq plot")
dsqqplot(pbox,data)
```

propsens

Monte Carlo propagation and sensitivity analysis

Description

Routines for propagating BPAs and sensitivity analysis.

Usage

```
dsevalmc(fhandle, x, mcIT, optimizer = dsbound, corr = NULL, samples = NULL, fnopti
dssensitivity(x, parnums, fhandle, uncfn, mcIT, pinch_samples, pinch_type = NULL, c
```

Arguments

| | |
|---------|--|
| fhandle | Function handle of type $f(x,\dots)$, where x is the vector of uncertain variables. |
| x | List of BPAs representing each variable |
| mcIT | Number of Monte Carlo iterations |

| | |
|----------------------------|---|
| <code>optimizer</code> | Optimizer for solving interval problems. Currently implemented: "dsbound" for monotonously increasing functions, "dsmonotonous" for monotonous functions (unknown direction) and "dsopt" for local optimization using L-BFGS. |
| <code>corr</code> | Optional: In case of dependent inputs: Vector of correlations |
| <code>samples</code> | Currently known samples of the function |
| <code>fnoptions</code> | options to pass to <code>fhandle</code> |
| <code>parnums</code> | Array of parameter numbers for which a sensitivity analysis should be performed. |
| <code>uncfn</code> | Uncertainty function handle (e.g. <code>dsaggwidth</code>). Needs to return a single numeric value for a BPA. |
| <code>pinch_samples</code> | Number of different pinches of an input. |
| <code>pinch_type</code> | Optional: Pinch a 'distribution' (default), 'interval' or 'point'. |

Details

The Monte Carlo propagation engine is the core of the toolbox. It allows propagating BPAs through arbitrary functions. The user can define the amount of Monte Carlo samples and the optimization method. Currently, four methods are implemented (`dsbound`, `dsmonotonous`, `dsmonotonous2`, `dsopt`).

`dsmonotonous` Recommended for monotonous functions (increasing/decreasing/mixed).

`dsmonotonous2` Evaluates all corners of the focal element and is recommended, if the function is not monotonous but quite regular.

`dsopt` Local optimizer for nonmonotonous functions.

`dsbound` Applicable in the special case that the function is monotonously increasing. Dependency between focal elements is modelled by a Gaussian copula as proposed in (Ferson, Hajagos et al. 2004). The copula parameters for n inputs need to be given as a vector of size $n*(n-1)/2$ with the first component $c1$ in $[0,1]$ representing the dependency between input 1&2, $c2$ between 1&3, ..., c_{n-1} between 1& n , c_n between 2&3 and so on. This format is analogous to the format used in the R package "copula". The function returns a data structure with a result BPA and all function evaluations. The method `dssensitivity` allows sensitivity analysis for a custom uncertainty measure. The function receives a list of BPAs x and a vector `parnums` giving all indices for which the sensitivity shall be computed. `fhandle` is the system function, `uncfn` the uncertainty measure. An arbitrary uncertainty measure can be used as long as it returns a single value for a BPA. A useful choice is "dsaggwidth", the aggregated width measure. The input BPAs are "pinch sample" times reduced to a structure with less uncertainty defined by "pinchtype" (either a point, a distribution or an interval). Each time, the BPA is propagated using `mcIT` Monte Carlo samples. The function returns a set of sensitivity indices.

The example illustrates the propagation process. Two functions are defined, `f1` being monotonous and `f2` nonmonotonous. The inputs `dss1` and `dss2` are defined as `pboxes`. The correlation is set to 0.5. Then, `dss1` and `dss2` are propagated through `f1` under independence and dependence. In the next step, `dsevalmc` uses the optimizer `dsopt` to propagate x through `f2`. The method `dssensitivity` is called with the aggregated width measure to calculate AW sensitivity.

Value

dsevalmc Returns list with two elements: [[1]] contains the result, [[2]] all calculated function points.

Author(s)

Philipp Limbourg <p.limboung@uni-due.de>

References

Ferson, S., J. Hajagos, et al. (2004). Dependence in Dempster-Shafer theory and probability bounds analysis. Sandia Report. Albuquerque, USA, Sandia National Laboratories.

Limbourg, P., R. Savic, et al. (2007). Fault Tree Analysis in an Early Design Stage using the Dempster-Shafer Theory of Evidence. European Conference on Safety and Reliability - ESREL 2007, Stavanger, Norway, Taylor and Francis.

Examples

```
print("Define monotonous function f1 and nonmonotonous function f2");
f1=(function(x){ z <- sqrt(abs(x[,1]+ 2*x[,2]));});
f2=(function(x){ z <- sin(x[,1]+ x[,2]);});
print("Create input BPAs x");
mu1=dsstruct(c(2,3,1));
mu2=dsstruct(c(4,4,1));
dss1=dsodf('qnorm',200,mu1);
dss2=dsodf('qnorm',500,mu2);
x=list(dss1,dss2)
correlations=0.5;
print("Propagate through f1 under independence assumption / correlation 0.5");
y_f1_independent=dsevalmc(f1,x,1000)
y_f1_dependent=dsevalmc(f1,x,1000,dsbound,correlations)
print("Propagate through f2 using optimization");

y_f2=dsevalmc(f2,x,1000,dsopt)
print("Plot results:");
dscdf(y_f1_independent[[1]])

dscdf(y_f1_dependent[[1]])

dscdf(y_f2[[1]])

print("Sensitivity on AW measure, inputs 1 & 2");
s=dssensitivity(x,c(1,2),f1,dsaggwidth,20,100,'distribution');
print(s)
```

Description

Functions for various different statistical properties of a BPA.

Usage

```
dsaggwidth(x)
dsconf(x, conf, confconf = NULL)
dsexpect(x)
dsvariance(x)
dsbelpl(x, a)
dssummary(x)
```

Arguments

| | |
|----------|--|
| x | BPA |
| conf | Confidence level in]0,1[|
| confconf | Wilk's confidence bounds in]0,1[on the conf (if empty, omitted). |
| a | Interval for Bel(x in a), Pl(x in a) |

Details

Various different statistical properties of a BPA: Aggregated width (dsaggwidth), expected value (dsexpect), quantiles (dsconf), variance (dsvariance) and Belief/Plausibility (dsbelpl). The function dsconf is also able to return Wilks' bounds for a given confidence level. The function dssummary is the analogue to a summary statistics and computes a lot of statistics at once.

The example calculates a set of statistical properties of a pbox x.

Value

Statistical property of a BPA (interval). dssummary returns a list of statistics.

Author(s)

Philipp Limbourg <p.limbourg@uni-due.de>

References

Kreinovich, V., G. Xiang, et al. (2006). "Computing mean and variance under Dempster-Shafer uncertainty: Towards faster algorithms." *International Journal of Approximate Reasoning* 42(3): 212-227.

Examples

```
mu=dsstruct(c(10,12,1))
sigma=dsstruct(c(1,1.5,1))
x=dsadf('qnorm',1000,mu,sigma)
print("Sample of 1000 focal elements from a normal dist")
print("Mean:")
print(dsexpect(x))
print("Variance:")
print(dsvariance(x))
print("Median:")
print(dsconf(x,0.5))
print("Bel and Pl of x in [4,8]:")
print(dsbelpl(x,c(4,8)))
print("Aggregated width:")
print(dsaggwidth(x))
print("95 percent conf. level with 95 percent Wilks bounds")
print(dsconf(x,0.95,0.95))
print("Summary statistics")
print(dssummary(x))
```

Index

*Topic package

ipptoolbox-package, 2

aggregation, 3

basicfunctions, 4

bpafromdata, 5

bpafromdist, 7, 8

bpafromprobs, 9

dsadf (*bpafromdist*), 7

dsaggwidth (*statprops*), 16

dsbel (*plotting*), 12

dsbelpl (*statprops*), 16

dsbelpltests (*bpafromdist*), 8

dsbound (*propsens*), 13

dscdf (*plotting*), 12

dsconf (*statprops*), 16

dsdempstersrule (*aggregation*), 3

dsecdf (*plotting*), 12

dsecdffit (*bpafromdata*), 5

dsenvelope (*aggregation*), 3

dsevalmc (*propsens*), 13

dsexample (*examples*), 11

dsexpect (*statprops*), 16

dsfcruesexample (*examples*), 11

dsintersect (*aggregation*), 3

dskskonf (*bpafromdata*), 5

dskstest (*bpafromdist*), 8

dslapconf (*bpafromdata*), 5

dsmeanvar (*bpafromprobs*), 9

dsminmeanmax (*bpafromprobs*), 9

dsminmodemax (*bpafromprobs*), 9

dsmonotonous (*propsens*), 13

dsmonotonous2 (*propsens*), 13

dsnrm (*basicfunctions*), 4

dsodf (*bpafromdist*), 7

dsopt (*propsens*), 13

dspl (*plotting*), 12

dsqqplot (*plotting*), 12

dsred (*basicfunctions*), 4

dssaveintersect (*aggregation*), 3

dssensitivity (*propsens*), 13

dsstruct (*basicfunctions*), 4

dssummary (*statprops*), 16

dsvariance (*statprops*), 16

dswavg (*aggregation*), 3

dswmix (*aggregation*), 3

dsxaveraging (*aggregation*), 3

dsxenvelope (*aggregation*), 3

examples, 11

ipptoolbox (*ipptoolbox-package*), 2

ipptoolbox-package, 2

plotting, 12

propsens, 13

statprops, 16