Package ‘ipptoolbox’

January 7, 2019

Type Package
Title IPP Toolbox
Version 1.2
Date 2019-01-02
Author Philipp Limbourg <p.limbourg@uni-due.de>
Depends R (>= 3.5), AlgDesign, copula, evd, triangle, kolmim
Maintainer Philipp Limbourg <p.limbourg@uni-due.de>
Description Uncertainty quantification and propagation in the framework of
Dempster-Shafer Theory and imprecise probabilities. This toolbox offers
easy-to-use methods for using imprecise probabilities for applied uncertainty
modelling and simulation. The package comprises the basic functionality
needed, with usability similar to standard probabilistic analysis:
- Fit imprecise probability distributions from data,
- Define imprecise probabilities based on distribution functions,
- Combine with various aggregation rules (e. g. Dempster's rule),
- Plotting tools,
- Propagate through arbitrary functions / simulations via Monte Carlo,
- Perform sensitivity analyses with imprecise distributions,
- Example models for a quick start.
License GPL (>= 2)
LazyLoad yes
RoxygenNote 6.1.1
NeedsCompilation no
Repository CRAN
Date/Publication 2019-01-07 17:10:09 UTC

R topics documented:

ipptoolbox-package ............................................. 2
aggregation ...................................................... 3
basicfunctions ................................................. 4
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

<table>
<thead>
<tr>
<th>Package</th>
<th>ipptoolbox</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Package</td>
</tr>
<tr>
<td>Version</td>
<td>1.2</td>
</tr>
<tr>
<td>Date</td>
<td>2019-01-02</td>
</tr>
<tr>
<td>License</td>
<td>GPL version 2 or newer</td>
</tr>
<tr>
<td>LazyLoad</td>
<td>yes</td>
</tr>
</tbody>
</table>

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


aggregation

Examples

```r
print("Example Fcrues")
print("See code")
print(dsfcrusexexample)
print("Execute example")
dsfcrusexexample()
```

---

<table>
<thead>
<tr>
<th>aggregation</th>
<th>Aggregation</th>
</tr>
</thead>
</table>

Description

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

Usage

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

Arguments

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


Examples

\begin{verbatim}
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);
\end{verbatim}

Description

Helpful functions for creating and handling BPAs

Usage

\begin{verbatim}
dssstruct(x)
dsnorm(y)
dssred(y, thres = 0.001)
\end{verbatim}

Arguments

\begin{verbatim}
x      Matrix/BPA
y      BPA
thres  Minimal mass value of a focal element desired
\end{verbatim}
bpafromdata

Details

dssstruct  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).
dssred    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 >= 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


Examples

a=dssstruct(matrix(c(2,3,0.1,1.5,5,0.9),ncol=3,byrow=TRUE))
b=dssstruct(matrix(c(2,3,0.5,10,5,0.9),ncol=3,byrow=TRUE))
c=dsodf('qexp',10000,dssstruct(c(10,20,1)));
d=dssred(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

dsecdffit(x)
dslapconf(x, lims = c(-Inf, Inf))
dskscconf(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

dsecdffit  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
dskconff  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). dsecdffit creates a BPA of the data set analogous to probabilistic empirical CDF. It is not conservative respective to distribution tails. dslapconf creates a BPA according to the Laplace method, dsksconf 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, dsecdffit, dslapconf and dsksconf 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


Examples

```r
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=dsecdffit(setofpoints)
dcdf(ecdfp)

ecdf=dsksconf(setofintervals,lim=c(-5,5))
dcdf(ecdf)

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

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

print("Plot Kolmogorov-Smirnov 75 percent bound BPA for both points and interval data")
ksp=dsksconf(setofpoints,conf=0.75,lim=c(-5,5))
```
Construct pboxes

Description

Methods to construct pboxes from distributions.

Usage

dsadf(fhandle, intervalnumber, ...)
dsfdf(fhandle, intervalnumber, ...)

Arguments

fhandle Inverse CDF function (e.g. qnorm)
intervalnumber Number of intervals of the resulting BPA
... Parameters (either points or BPAs) for fhandle

Details

dsadf Average discretization method
dsfdf 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


Examples

```r
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=dsadfqnorm(20,mu,sigma)
dscdf(pbox1)

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

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

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

---

**bpafromprobs**

*Obtain BPAs from statistical properties*

**Description**

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

**Usage**

```r
dsmninmeanmax(intervalnumber, min, mean, max)
dsmninmodemax(intervalnumber, min, mode, max)
dsmmeanvar(intervalnumber, mean, var)
```

**Arguments**

- **intervalnumber**: Number of focal elements of the discretization
- **min**: Minimum of the BPA
- **mean**: Mean of the BPA
- **mode**: Mode of the BPA
- **max**: Maximum of the BPA
- **var**: Variance of the BPA
Details

dsmimmeanmax generates a BPA from min, mean and max.
dsmimmodemax 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

```r
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)
```

Description

Two self-contained examples, see commented code.
Usage

dsfcruesexample()
dexample()

Details

Please see the code of the examples.

Author(s)

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

References

Philipp Limbourg, Etienne de Rocquigny (2010). Uncertainty analysis using evidence theory -
confronting level-1 and level-2 approaches with data availability and computational constraints.

Philipp Limbourg, Robert 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

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

Details

- **dscdf**: Plots Bel([-Inf,x]) and Pl([-Inf,x]). These functions bound all CDFs.
- **dsqqplot**: Plots a quantile-quantile plot between data and a BPA.
- **dsbel**: Returns Bel([-Inf,x]), e.g. for own plotting routines.
- **dspl**: Returns Pl([-Inf,x]), e.g. for own plotting routines.

The function dscdf plots Bel([-Inf,x]) and Pl([-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 Bel([-Inf,x])/ Pl([-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

```r
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)```
Description
Routines for propagating BPAs and sensitivity analysis.

Usage

dsevalmc(fhandle, x, mcIT, optimizer = dsbound, corr = NULL, 
samples = NULL, fnoptions = NULL)

dssensitivity(x, parnums, fhandle, uncfn, mcIT, pinch_samples, 
pinch_type = NULL, optimizer = dsmonotonous, fnoptions = NULL)

Arguments

fhandle Function handle of type f(x,...), where x is the vector of uncertain variables.
x List of BPAs representing each variable
mcIT Number of Monte Carlo iterations
optimizer 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.
corr Optional: In case of dependent inputs: Vector of correlations
samples Currently known samples of the function
fnoptions options to pass to fhandle
parnums Array of parameter numbers for which a sensitivity analysis should be per-
formed.
uncfn Uncertainty function handle (e.g. dsaggwidth). Needs to return a single numeric 
value for a BPA.
pinch_samples Number of different pinches of an input.
pinch_type 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, ..., cn-1 between 1&n, cn 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 pnums 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 "dsagwidth", the aggregated width measure. The input BPAs are "pinch sample" times reduced to a structure with less uncertainty defined by "pinchtyle" (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.limbourg@uni-due.de>

References

Examples

```r
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 BPA 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)
```

---

**statprops**  
_Statistical properties of a BPA_

**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[
**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**


**Examples**

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

*Topic package
  ipptoolbox-package, 2

  aggregation, 3

  basicfunctions, 4
  bpafromdata, 5
  bpafromdist, 7
  bpafromprobs, 8

  dsadf (bpafromdist), 7
  dsaggwidth (statprops), 14
  dsbel (plotting), 10
  dsbelpl (statprops), 14
  dsbelptests (tests), 16
  dsbounds (propsens), 12
  dscdf (plotting), 10
  dsconf (statprops), 14
  dsdempstersrule (aggregation), 3
  dsedcdf (plotting), 10
  dsedcddf (bpafromdata), 5
  dsenvelope (aggregation), 3
  dsevalue (propsens), 12
  dsexample (examples), 9
  dsexpect (statprops), 14
  dsfcruesexample (examples), 9
  dsintersect (aggregation), 3
  dsksconf (bpafromdata), 5
  dskskstest (tests), 16
  dslapconf (bpafromdata), 5
  dsmeanvar (bpafromprobs), 8
  dsminmeanmax (bpafromprobs), 8
  dsminmodmax (bpafromprobs), 8
  dsmonotonous (propsens), 12
  dsmonotonous (propsens), 12
  dsnorm (basicfunctions), 4
  dsodf (bpafromdist), 7
  dsospt (propsens), 12
  dspl (plotting), 10
  dsqqplot (plotting), 10

  dssaveintersect (aggregation), 3
  dssensitivity (propsens), 12
  dsdstruct (basicfunctions), 4
  dssummary (statprops), 14
  dsvariance (statprops), 14
  dswavg (aggregation), 3
  dswnmix (aggregation), 3
  dsxaveraging (aggregation), 3
  dsxaveraging (aggregation), 3

  examples, 9

  ipptoolbox (ipptoolbox-package), 2
  ipptoolbox-package, 2

  plotting, 10
  propsens, 12
  statprops, 14

  tests, 16