Package ‘EvCombR’

February 19, 2015

Type Package
Title Evidence Combination in R
Version 0.1-2
Date 2014-04-22
Author Alexander Karlsson
Maintainer Alexander Karlsson <alexander.karlsson@his.se>
Copyright Alexander Karlsson
Description Package for combining pieces of evidence
License MIT + file LICENSE
Depends methods
NeedsCompilation no
Repository CRAN
Date/Publication 2014-04-24 18:24:54

R topics documented:

EvCombR-package ................................................. 2
  cComb .......................................................... 4
  cComb-methods ............................................... 5
  credal .......................................................... 5
  credal-class ................................................. 6
  credal-methods .............................................. 7
  dComb .......................................................... 8
  dComb-methods ............................................... 9
  disc ............................................................ 9
  disc-methods ............................................... 10
  extPoints ..................................................... 10
  extPoints-methods .......................................... 11
  focal ......................................................... 11
  focal-methods ............................................... 12
  focal<- ....................................................... 13
  focal<-methods ........................................... 14
EvCombR-package

Description

Package for combining pieces of evidence.

Details

Package: EvCombR
Type: Package
Version: 0.1-2
Date: 2014-04-22
License: MIT
Depends: methods

Implements Dempster’s, Yager’s, modified Dempster’s, Bayesian, and credal combination (based on intervals).
Author(s)

Alexander Karlsson

Maintainer: Alexander Karlsson <alexander.karlsson@his.se>

References


Arnborg, S. (2006), Robust Bayesianism: Relation to Evidence Theory, *Journal of Advances in Information Fusion*, 1, 63-74


Examples

```r
# construct a state space
stateSpace <- c("a", "b", "c")

# construct credal sets with the given state space
c1 <- credal(c(0.1, 0.1, 0.1), c(0.8, 0.8, 0.8), stateSpace)
c2 <- credal(c(0.2, 0.2, 0.2), c(0.9, 0.9, 0.9), stateSpace)

# combine the credal sets
cComb(c1, c2)

# construct mass functions
m1 <- mass(list("a"=0.1, "b"=0.1, "c"=0.4, "a/b/c"=0.4), stateSpace)
m2 <- mass(list("a"=0.2, "b"=0.2, "c"=0.2, "a/b/c"=0.4), stateSpace)

# combine the mass functin by using Dempster's combination
dComb(m1, m2)

# Yager's combination operator
yComb(m1, m2)

# modified Dempster's combination using uniform prior
mComb(m1, m2)
```
cComb

**Credal Combination Operator (restricted to intervals)**

**Description**
Combine evidence in the form of credal sets (based on intervals) using the credal combination operator (also known as the robust Bayesian combination operator). The resulting credal set is approximated by using probability intervals.

**Usage**
\[cComb(x, y)\]

**Arguments**
- \(x\) credal set or a list of credal sets
- \(y\) credal set if \(x\) is a credal set, otherwise missing

**Value**
credal set

**Author(s)**
Alexander Karlsson

**References**
Arnborg, S. (2006), Robust Bayesianism: Relation to Evidence Theory, *Journal of Advances in Information Fusion*, 1, 63-74

**See Also**
dComb, yComb, mComb

**Examples**
```r
# construct a state space
stateSpace <- c("a", "b", "c")

# construct credal sets with the given state space
credal(c(0.1, 0.1, 0.1), c(0.8, 0.8, 0.8), stateSpace)
credal(c(0.2, 0.2, 0.2), c(0.9, 0.9, 0.9), stateSpace)
```
# combine the credal sets
cComb(c1, c2)
# or by

cComb(list(c1, c2))

---

## Description

Combine credal sets (based on intervals) using the credal combination operator (also known as the robust Bayesian combination operator). For more detail see `cComb`.

## Methods

- **signature(x = "credal", y = "credal")** Combine two credal sets using the credal combination operator
- **signature(x = "list", y = "missing")** Combine a list of credal sets using the credal combination operator

---

## credal

### Constructor Function for Credal Sets (based on intervals)

Construct a credal set based on probability intervals or a single probability function. The algorithm used for finding the extreme points corresponding to lower and upper bounds is described in De Campos et al. (1994).

### Usage

```
credal(x, y, z)
```

### Arguments

- **x** lower bounds of probability intervals (in the form of a numeric vector)
- **y** upper bounds for probability intervals or missing (i.e., upper bound of 1)
- **z** character vector representing the state space

### Value

A credal set represented by a set of extreme points.
Author(s)

Alexander Karlsson

References


Arnborg, S. (2006), Robust Bayesianism: Relation to Evidence Theory, Journal of Advances in Information Fusion, 1, 63-74


See Also

cComb

Examples

# state space
stateSpace <- c("a", "b", "c")

# lower and upper bounds for probability intervals
c1 <- credal(c(0.1, 0.1, 0.1), c(0.8, 0.8, 0.8), stateSpace)

# single probability function (lower and upper bounds of probability intervals are equal)
c2 <- credal(c(0.1, 0.2, 0.7), c(0.1, 0.2, 0.7), stateSpace)

credal-class

Class "credal"

Description

Represents a credal set by a set of extreme points. For more detail see credal.

Objects from the Class

Objects can be created by credal.

Slots

extPoints: Object of class "matrix". Each row is an extreme point of the credal set.
Methods

[ signature(x="credal", i="ANY", j="ANY"): extract an extreme point
lique signature(x="credal", i="ANY", j="ANY", value="ANY"): replace an extreme point

cComb signature(x = "credal", y = "credal"): combine two credal sets

clower signature(x = "credal", set = "character"): calculate the lower bound for a specific set of states

clower signature(x = "credal", set = "missing"): calculate the lower bounds for all singleton states

cupper signature(x = "credal", set = "character"): calculate the upper bound for a specific set of states

cupper signature(x = "credal", set = "missing"): calculate the upper bounds for all singleton states

cextPoints signature(x = "credal"): access method for the slot points

cspace signature(x = "credal"): access method for names of singleton states

cspace<- signature(x = "credal"): replace method for names of singleton states

Author(s)

Alexander Karlsson

Description

Methods for constructing a credal set. For more detail see credal.

Methods

signature(x = "numeric", y = "missing", z = "character") Construct a credal set based on the lower bounds of probability intervals for states (1 will be the upper bound for all probability intervals)

signature(x = "numeric", y = "numeric", z = "character") Construct a credal based on probability intervals for states

Author(s)

Alexander Karlsson
Dempster’s Combination Operator

Description

Combine evidence in the form of mass functions using Dempster’s combination operator.

Usage

dComb(x, y)

Arguments

x
single mass function or a list of mass functions

y
single mass function if x is a single mass function, otherwise missing

Value

mass function

Author(s)

Alexander Karlsson

References


See Also

yComb, mComb, cComb

Examples

# state space
stateSpace <- c("a", "b", "c")

# mass functions
m1 <- mass(list("a"=0.1, "a/b/c"=0.9), stateSpace)
m2 <- mass(list("a"=0.2, "a/b/c"=0.8), stateSpace)

# Dempster's combination
dComb(m1, m2)
# or
dComb(list(m1, m2))
Methods for Function `dComb`

**Description**

Combine mass functions using Dempster's combination operator. For more detail see `dComb`.

**Methods**

- `signature(x = "mass", y = "mass")` Combine two mass functions using Dempster's combination operator.
- `signature(x = "list", y = "missing")` Combine a list of mass functions using Dempster's combination operator.

**Author(s)**

Alexander Karlsson

---

**disc**

**Discounting Operator**

**Description**

Discounts a mass function.

**Usage**

`disc(x, y)`

**Arguments**

- `x` a mass function
- `y` degree of reliability

**Value**

mass function

**Author(s)**

Alexander Karlsson

**References**

Examples

```r
# state space
stateSpace <- c("a", "b", "c")

# mass function
m <- mass(list("a"=0.1, "a/b/c"=0.9), stateSpace)

# source is only 80% reliable
mdisc <- disc(m, 0.8)
```

**disc**-methods  
*Methods for Function disc*

**Description**

Discount an evidence structure. For more detail see `disc`

**Methods**

`signature(x = "mass", y = "numeric")` Discount a mass function.

**extPoints**  
*Extreme Points of a Credal Set*

**Description**

Returns the extreme points of a credal set

**Usage**

`extPoints(x)`

**Arguments**

- `x`  
a credal set

**Value**

a matrix where the extreme points are stored by row

**Author(s)**

Alexander Karlsson

**See Also**

`lower, upper`
Examples

# state space
stateSpace <- c("a", "b", "c")

# construct credal set
c <- credal(c(0.1, 0.1, 0.1), c(0.8, 0.8, 0.8), stateSpace)

# obtain extrem points
eMat <- extPoints(c)

Description

Returns the set of extreme points of a credal set. For more detail see extPoints.

Methods

signature(x = "credal") Returns the set of extreme points

Author(s)

Alexander Karlsson

Focal Elements of a Mass Function

Description

Returns the set of focal elements of a mass function.

Usage

focal(x)

Arguments

x a mass function

Value

focal elements of x
Author(s)

Alexander Karlsson

References


See Also

points

Examples

```r
# state space
stateSpace <- c("a", "b", "c")

# mass functions
m <- mass(list("a"=0.1, "b"=0.1, "c"=0.4, "a/b/c"=0.4), stateSpace)

# obtain focal elements
focal(m)
```

Description

Methods for function focal

Methods

```r
signature(x = "mass") Access function for slot focal
```

Note

See further focal
Description

Replaces focal elements of a mass function.

Usage

focal(x) <- value

Arguments

x a mass function
value new focal elements for the mass function

Value

mass function with focal elements replaced.

Author(s)

Alexander Karlsson

References


Examples

# state space
stateSpace <- c("a", "b", "c")

# mass functions
m <- mass(list("a"=0.1, "b"=0.1 , "c"=0.4, "a/b/c"=0.4), stateSpace)

# replace focal elements
focal(m) <- list("a/b"=1)
Description

Replacement function for focal elements. For more detail see `focal<-`

Methods

`signature(x = "mass")` Replace focal elements

lower

Lower Bounds Based on Evidence Structure

Description

Calculate the lower bounds for a vector of sets

Usage

`lower(x, sets)`

Arguments

`x` credal set or mass function

`sets` vector of sets where each set is represented by state names separated by "/". If sets are missing, lower bounds on singletons are calculated.

Value

lower bound of mass or probability for each set in the vector sets or if sets is missing lower bounds on singletons

Note

This is equivalent to belief in Dempster-Shafer theory

Author(s)

Alexander Karlsson

References


See Also

upper

Examples

# state space
stateSpace <- c("a", "b", "c")

# mass function
m <- mass(list("a"=0.1, "b"=0.1,
               "c"=0.4, "a/b/c"=0.4), stateSpace)

# credal set
c <- credal(c(0.1, 0.1, 0.1),
             c(0.8, 0.8, 0.8), stateSpace)

# calculate lower bounds
lower(m, c("a", "a/b"))
lower(c, c("a", "a/b"))

# lower bounds on singletons
lower(m)

lower-methods

Methods for Function lower

Description

Calculate lower bounds for a vector of sets with respect to the evidence structure. For more detail see lower

Methods

signature(x = "credal", sets = "character") obtain lower bounds for a vector of sets
signature(x = "credal", sets = "missing") obtain lower bounds for all singleton states
signature(x = "mass", sets = "character") obtain the belief, or lower bounds, for a vector of sets
signature(x = "mass", sets = "missing") obtain the belief, or lower bounds, for all singleton states
mass

Constructor Function for Mass Functions

Description

Construct a mass function based on a named list of focal elements or a \texttt{massQ-class} object. For more information, see the details section.

Usage

\texttt{mass(x, y)}

Arguments

\begin{itemize}
\item \texttt{x} \quad \text{a named list of focal elements or a \texttt{massQ-class} object}
\item \texttt{y} \quad \text{a character vector representing the state space or missing if \texttt{x} is an \texttt{massQ} object.}
\end{itemize}

Details

Focal elements are represented by the notation "\textless s1\textgreater /.../\textless sn\textgreater " where \textless s1\textgreater ...\textless sn\textgreater are any states within the state space (see the examples below). Note that the word "ES" and the symbol "/" are reserved.

Value

mass function

Author(s)

Alexander Karlsson

References


See Also

\texttt{dComb, mComb, yComb}
Examples

```r
# state space
stateSpace <- c("a", "b", "c")

# construct mass functions
m1 <- mass(list("a"=0.1, "b"=0.1, "c"=0.4, "a/b/c"=0.4), stateSpace)

m2 <- mass(list("a"=0.1, "b"=0.1, "c"=0.1, "a/b"=0.1, "a/c"=0.1, "b/c"=0.1, "a/b/c"=0.4), stateSpace)

# apply Yager's combination operator, m12 will be a massQ-object
m12Q <- yComb(m1, m2)

# construct a mass function from an massQ-object
m12 <- mass(m12Q)
```

mass-class

Class "mass"

Description

Represents a mass function by a list of focal elements and corresponding mass. For more detail see `mass`.

Objects from the Class

Objects can be created by `credal`.

Slots

- `focal`: a list of focal elements represented by statenames separated by "/"
- `space`: the state space represented by a character vector

Methods

- `signature(x = "mass", i = "character", j = "missing")`: extract focal elements
- `signature(x = "mass", i = "character", j = "missing")`: extract a single focal element
- `signature(x="mass", i="character", j="missing", value="ANY")`: replace focal elements
- `signature(x="mass", i="character", j="missing", value="ANY")`: replace a single focal element
- `dComb signature(x = "mass", y = "mass")`: combine two mass functions by Dempster's combination
- `focal signature(x = "mass")`: access focal elements
- `focal<- signature(x = "mass")`: replace focal elements
lower signature(x = "mass", set = "character"): calculate the lower bounds for some focal element
lower signature(x = "mass", set = "missing"): calculate the lower bounds for singletons
mComb signature(x = "mass", y = "mass", z = "function"): combine two mass functions by modified Dempster's combination using a prior distribution z
mComb signature(x = "mass", y = "mass", z = "missing"): combine two mass functions by modified Dempster's combination using a uniform prior distribution z
pign signature(x = "mass"): calculate the pignistic transformation for single states
relPl signature(x = "mass"): calculate the relative plausibility for single states
space signature(x = "mass"): access the state space (frame of discernment)
space<- signature(x = "mass"): replace the state space (frame of discernment)
upper signature(x = "mass", set = "character"): calculate the upper bound for some focal element
upper signature(x = "mass", set = "character"): calculate the upper bounds for singletons
yComb signature(x = "mass", y = "mass"): combine two mass functions using Yager's rule
disc signature(x = "mass", y = "numeric"): discount mass function

Author(s)
Alexander Karlsson

References

mass-methods

Methods for Function mass

Description
Methods for constructing a mass function. For more detail see mass

Methods
signature(x = "list", y = "character") Construct a mass functions by a named list of focal elements and a given state space
signature(x = "massQ", y = "missing") Construct a mass function from a massQ-class object
massQ-class

Author(s)
Alexander Karlsson

massQ-class  Class "massQ"

Description
Class that maintains information about the mass on the empty set. The class is used for Yager’s combination operator

Objects from the Class
A massQ-object is obtained as a result of Yager’s combination operator \( y_{\text{Comb}} \).

Slots
- qEmpty: mass on the empty set with respect to the previous combination
- focal: a list of focal elements represented by statenames seperated by "/"
- space: the state space represented by a character vector

Extends
Class "mass", directly.

Methods
All methods inherited from mass-class and in addition:

mass signature(x = "massQ", y = "missing"): convert the massQ-object to a mass-object

Author(s)
Alexander Karlsson

References
Description

Combine evidence in the form of mass functions using modified Dempster’s combination operator.

Usage

\[ m\text{Comb}(x, y, z) \]

Arguments

- \(x\): single mass function or a list of mass functions
- \(y\): single mass function if \(x\) is a single mass function, a prior distribution or missing if \(x\) is a list
- \(z\): prior distribution if \(x\) and \(y\) are mass functions, otherwise missing

Details

The prior distribution is provided in the form of a list where the names are equivalent to the state space. See the examples.

Value

mass function

Author(s)

Alexander Karlsson

References


See Also

dComb, yComb, cComb

Examples

```r
# state space
stateSpace <- c("a", "b", "c")

# mass functions
m1 <- mass(list("a"=0.1, "a/b/c"=0.9), stateSpace)
m2 <- mass(list("a"=0.2, "a/b/c"=0.8), stateSpace)
```
mComb-methods

Description

Combine mass functions using modified Dempster's combination operator. For more detail see mComb.

Methods

signature(x = "mass", y = "mass", z = "list") Combine two mass functions using modified Dempster's combination operator and a prior
signature(x = "mass", y = "mass", z = "missing") Combine two mass functions using modified Dempster's combination operator and the uniform prior
signature(x = "list", y = "list", z = "missing") Combine a list of mass functions using modified Dempster's combination operator and a prior
signature(x = "list", y = "missing", z = "missing") Combine a list of mass functions using modified Dempster's combination operator and the uniform prior

pign

Pignistic Transformation

Description

The pignistic transformation transforms a mass function into a probability function.

Usage

pign(x)

Arguments

x a mass function
Value

a singleton credal set

Author(s)

Alexander Karlsson

References


See Also

relPl

Examples

```r
# state space
stateSpace <- c("a", "b", "c")

# mass function
m <- mass(list("a"=0.1, "a/b/c"=0.9), stateSpace)

# obtaina singleton credal set
c <- pign(m)
```

Description

The pignistic transformation transform a mass function to probability function. For more detail see pign

Methods

`signature(x = "mass")` Apply the pignistic transformation on a mass function
Description

The relative plausibility transform transform a mass function to a probability function.

Usage

relPl(x)

Arguments

x a mass function

Value

a singleton credal set

Author(s)

Alexander Karlsson

References


See Also

pign

Examples

# state space
stateSpace <- c("a", "b", "c")

# mass function
m <- mass(list("a"=0.1, "a/b/c"=0.9), stateSpace)

# obtaina singleton credal set
c <- relPl(m)
**relPl-methods**  
*Methods for Function* relPl

**Description**  
The relative plausability transform transforms a mass function to probability function. For more detail see relPl

**Methods**  
`signature(x = "mass")` Apply the relative plausability transform on a mass function

---

**space**  
*State Space of and Evidence Structure*

**Description**  
This functions returns the state space of an evidence structure.

**Usage**  
`space(x)`

**Arguments**  
`x` mass function or credal set

**Value**  
a character vector with the names within the state space

**Author(s)**  
Alexander Karlsson

**Examples**  
```r
# state space
stateSpace <- c("a", "b", "c")

# construct mass function
m <- mass(list("a"=0.1, "b"=0.1, "c"=0.4, "a/b/c"=0.4), stateSpace)

# obtain state space
space(m)
```
Methods for Function space

Description

Returns the state space for an evidence structure. For more detail see space.

Methods

signature(x = "credal") Returns the state space for a credal set
signature(x = "mass") Returns the state space for a mass function

Replacement Function for State Space

Description

Replace the names of the state space

Usage

space(x) <- value

Arguments

x mass function or credal set
value new state space given as a character vector

Value

new mass function or credal set with the state space replaced

Author(s)

Alexander Karlsson

See Also

focal<-
Examples

```r
# state space
stateSpace <- c("a", "b", "c")

# construct mass function
m <- mass(list("a"=0.1, "b"=0.1, "c"=0.4, "a/b/c"=0.4), stateSpace)

# replace state space
space(m) <- c("d", "e", "f")
```

Methods

**space**<-methods  
*Methods for Function space<-

Description

Replace the state space of an evidence structure. For more details see `space`.

Methods

- `signature(x = "credal")` Replace state space of a credal set
- `signature(x = "mass")` Replace the state space of a mass function

upper

*Upper Bounds Based on Evidence Structure*

Description

Calculate the upper bounds for a vector of sets

Usage

```r
upper(x, sets)
```

Arguments

- `x` credal set or mass function
- `sets` vector of sets where each set is represented by state names separated by "/". If sets are missing, upper bounds on singletons are calculated.

Value

upper bound of mass or probability for each set in the vector sets or if sets is missing upper bounds on singletons
Note
This is equivalent to Belief in Dempster-Shafer theory

Author(s)
Alexander Karlsson

References

See Also
upper

Examples
# state space
stateSpace <- c("a", "b", "c")

# mass function
m <- mass(list("a"=0.1, "b"=0.1,
               "c"=0.4, "a/b/c"=0.4), stateSpace)

# credal set
c <- credal(c(0.1, 0.1, 0.1),
             c(0.8, 0.8, 0.8), stateSpace)

# calculate upper bounds
upper(m, c("a", "a/b"))
upper(c, c("a", "a/b"))

# upper bounds on singletons
upper(m)

Methods for Function upper

Description
Calculate lower bounds for a vector of sets with respect to the evidence structure. For more detail see upper
Methods
signature(x = "credal", sets = "character") obtain upper bounds for a vector of sets
signature(x = "credal", sets = "missing") obtain upper bounds for all singletons
signature(x = "mass", sets = "character") obtain the plausability, or upper bounds, for a vector of sets
signature(x = "mass", sets = "missing") obtain the plausability, or upper bounds, for all singletons

---

yComb  Yager's Combination Operator

Description
Combine evidence in the form of mass functions using Yager’s combination operator.

Usage
yComb(x, y)

Arguments
x single mass function or a list of mass functions
y single mass function if x is a single mass function, otherwise missing

Value
mass function (massQ-class)

Note
Yager’s combination operator is quasi-associative and therefore we need to keep track of the mass on the empty set by using the class massQ.

Author(s)
Alexander Karlsson

References

See Also
dComb, mComb, cComb
Examples

```r
# state space
stateSpace <- c("a", "b", "c")

# mass functions
m1 <- mass(list("a"=0.1, "a/b/c"=0.9), stateSpace)
m2 <- mass(list("b"=0.2, "a/b/c"=0.8), stateSpace)

# Yager's combination
yComb(m1, m2)
# or
yComb(list(m1, m2))
```

yComb-methods

Methods for Function `yComb`

Description

Combine mass functions using Yager's combination operator. For more detail see `yComb`.

Methods

- `signature(x = "mass", y = "mass")` Combine two mass functions using Yager's combination operator
- `signature(x = "list", y = "missing")` Combine a list of mass functions using Yager's combination operator

[ ]-methods

Methods for Function `[`

Description

Extract part of evidence structure `[`

Methods

- `signature(x = "credal", i = "ANY", j="ANY", value="ANY")` Extract probabilities
- `signature(x = "mass", i = "character", j="missing", value="ANY")` Extract focal element(s)

Author(s)

Alexander Karlsson
Examples

# construct a state space
stateSpace <- c("a", "b", "c")

# construct credal sets with the given state space
c <- credal(c(0.1, 0.1, 0.1), c(0.8, 0.8, 0.8), stateSpace)

# extract first and second extreme point
c[1:2,]

# mass functions
m <- mass(list("a"=0.1, "b"=0.1, "c"=0.4, "a/b/c"=0.4), stateSpace)

# extract focal elements
m[c("a","a/b/c")]

Description

Replace part of an evidence structure

Methods

signature(x="credal", i="ANY", j="ANY", value="ANY") Replace probabilities
signature(x="mass", i="character", j="missing", value="ANY") Replace focal element(s)

Author(s)

Alexander Karlsson

Examples

# construct a state space
stateSpace <- c("a", "b", "c")

# construct credal sets with the given state space
c <- credal(c(0.1, 0.1, 0.1), c(0.8, 0.8, 0.8), stateSpace)

# replace first and second extreme point
c[1:2,] <- rbind(c(0.1, 0.1, 0.8), c(0.2, 0.2, 0.6))

# mass function
m <- mass(list("a"=0.1, "b"=0.1, "c"=0.4, "a/b/c"=0.4), stateSpace)

# switch mass on focal elements "b" and "a/b/c"
temp <- m["b"]
m["b"] <- m["a/b/c"]
Methods for Function \[<\]

Description
Methods for function \[<\]

Methods

\[
\text{signature(x="mass", i="character", j="missing") Extract a single focal element from the list of focal elements}
\]

Author(s)
Alexander Karlsson

Examples

\[
\text{C construct a state space}
\text{statespace <- c("a", "b", "c")}
\text{Cmass functions}
\text{m <- mass(list("a"=0.1, "b"=0.1, "c"=0.4, "a/b/c"=0.4), statespace)}
\text{C extract focal element}
\text{m[["a"]]}\]

Description
Replace part of an evidence structure

Methods

\[
\text{signature(x="mass", i="character", j="missing", value="ANY") Replace focal element(s)}
\]

Author(s)
Alexander Karlsson
Examples

# construct a state space
stateSpace <- c("a", "b", "c")

# mass function
m <- mass(list("a"=0.1, "b"=0.1, "c"=0.4, "a/b/c"=0.4), stateSpace)

# obtain value only
m[["a"]]

Index

*Topic **classes**
  credal-class, 6
  mass-class, 17

*Topic **methods**
  [methods, 29
  [methods, 30
  [methods, 31
  [methods, 31
  cComb-methods, 5
  dComb-methods, 9
  disc-methods, 10
  focal-methods, 12
  focal<-methods, 14
  lower-methods, 15
  mass-methods, 18
  mComb-methods, 21
  pign-methods, 22
  relP1-methods, 24
  space-methods, 25
  space<-methods, 26
  upper-methods, 27
  yComb-methods, 29

*Topic **package**
  EvCombr-package, 2
  [,credal,ANY,ANY-method ([methods], 29
  [,mass,character,missing-method ([methods], 29
  [methods, 29
  [,credal,ANY,ANY-method ([methods], 30
  [,mass,character,missing-method ([methods], 30
  [methods, 30
  [mass,character,missing-method ([methods], 31
  [methods, 31
  [,mass,character,missing-method ([methods], 31
  cComb, 4, 5, 6, 8, 20, 28
  cComb, credal, credal-method (cComb-methods), 5
  cComb, list, missing-method (cComb-methods), 5
  cComb-methods, 5
  credal, 5, 6, 7, 17
  credal, matrix, character, missing-method (credal-methods), 7
  credal, numeric, missing, character-method (credal-methods), 7
  credal, numeric, numeric, character-method (credal-methods), 7
  credal-class, 6
  credal-methods, 7
  dComb, 4, 8, 9, 16, 20, 28
  dComb, list, missing-method (dComb-methods), 9
  dComb, mass, mass-method (dComb-methods), 9
  dComb-methods, 9
  disc, 9, 10
  disc, mass, numeric-method (disc-methods), 10
  disc-methods, 10
  EvCombr (EvCombr-package), 2
  EvCombr-package, 2
  extPoints, 10, 11
  extPoints, credal-method (extPoints-methods), 11
  extPoints-methods, 11
  focal, 11, 12
  focal, mass-method (focal-methods), 12
  focal-methods, 12
  focal<-, 13
focal<-, mass, list-method
  (focal<---methods), 14
focal<---methods, 14

lower, 10, 14, 15
lower, credal, character-method
  (lower-methods), 15
lower, credal, missing-method
  (lower-methods), 15
lower, mass, character-method
  (lower-methods), 15
lower, mass, missing-method
  (lower-methods), 15
lower-methods, 15

mass, 16, 17–19
mass, list, character-method
  (mass-methods), 18
mass, massQ, missing-method
  (mass-methods), 18
mass-class, 17
mass-methods, 18
massQ-class, 19
mComb, 4, 8, 16, 20, 21, 28
mComb, list, list, missing-method
  (mComb-methods), 21
mComb, list, missing, missing-method
  (mComb-methods), 21
mComb, mass, mass, list-method
  (mComb-methods), 21
mComb, mass, mass, missing-method
  (mComb-methods), 21
mComb-methods, 21

pign, 21, 22, 23
pign, mass-method (pign-methods), 22
pign-methods, 22
points, 12

relPL, 22, 23, 24
relPL, mass-method (relPL-methods), 24
relPL-methods, 24

space, 24, 25, 26
space, credal-method (space-methods), 25
space, mass-method (space-methods), 25
space-methods, 25
space<-, 25
space<-, credal, character-method
  (space<--methods), 26
space<--methods, 26

upper, 10, 15, 26, 27
upper, credal, character-method
  (upper-methods), 27
upper, credal, missing-method
  (upper-methods), 27
upper, mass, character-method
  (upper-methods), 27
upper, mass, missing-method
  (upper-methods), 27
upper-methods, 27

yComb, 4, 8, 16, 19, 20, 28, 29
yComb, list, missing-method
  (yComb-methods), 29
yComb, mass, mass-method (yComb-methods), 29
yComb-methods, 29