# Package 'AssocBin'

November 23, 2023

**Version** 0.1-0

**Encoding** UTF-8

**Title** Measuring Association with Recursive Binning

**Description** An iterative implementation of a recursive binary partitioning algorithm to measure pairwise dependence with a modular design that allows user specification of the splitting logic and stop criteria. Helper functions provide suggested versions of both and support visualization and the computation of summary statistics on final binnings. For a complete description of the functionality and algorithm, see Salahub and Oldford (2023) <arxiv:2311.08561>.

**Maintainer** Chris Salahub <chris.salahub@uwaterloo.ca>

**Depends** R (>= 4.3.0)

**Imports**

**Suggests** knitr, rmarkdown

**Enhances**

**License** GPL (>= 3)

**NeedsCompilation** no

**Repository** CRAN

**VignetteBuilder** knitr

**RoxygenNote** 7.2.3

**Author** Chris Salahub [aut, cre] (<https://orcid.org/0000-0003-3770-6798>)

**Date/Publication** 2023-11-23 12:50:05 UTC

## R topics documented:

---

`binChi` *Statistics for bins*

---

## Description

These functions compute statistics based on observed and expected counts for a list of bins.

## Usage

```
binChi(bins, agg = sum)

binMI(bins, agg = sum)

binAbsDif(bins, agg = sum)
```

## Arguments

| | |
|---|---|
| `bins` | a list of bins, each a list with elements 'x', 'y', 'depth', 'bnds' (list with elements 'x' and 'y'), 'expn', 'n' |
| `agg` | function which is aggregates the individual statistics computed over each bin |

## Details

Binstatistics

Three functions are provided by default, 'binChi' computes the chi-squared statistic by taking the squared difference between observed and expected counts and dividing this by the expected counts. 'binMi' computes the mutual information for each bin using the observed and expected counts. Finally, 'binAbsDif' computes the absolute difference between observed and expected counts. Each function first computes a value on every bin independently and stores all these values in memory before using the function provided in the optional argument 'agg' to aggregate these values.

## Value

A list with elements 'residuals' and 'stat' reporting the individual statistic values (possibly transformed) and the aggegrated statistic value.

## Functions

- `binChi()`: Chi-squared statistic
- `binMI()`: Mutual information
- `binAbsDif()`: Absolute difference between observed and expected

## Author(s)

Chris Salahub

## Examples

```
binList1 <- list(list(x = c(1,2), y = c(3,1), depth = 1, n = 2,
                      expn = 2),
                 list(x = c(3,4), y = c(2,4), depth = 1, n = 2,
                      expn = 2))
binList2 <- list(list(x = c(1,2), y = c(3,1), depth = 6, n = 2,
                      expn = 4),
                 list(x = c(), y = c(), depth = 1, n = 0, expn = 1))
binChi(binList1)
binChi(binList2)
binMI(binList1)
binMI(binList2)
binAbsDif(binList2)
```

---

`binner` *Wrapper for recursive binning*

---

## Description

'binner' is an iterative implementation of a recursive binary partitioning algorithm which accepts the splitting and stopping functions that guide partitioning as arguments.

## Usage

```
binner(x, y, stopper, splitter, init = halfSplit)
```

## Arguments

| | |
|---|---|
| `x` | numeric vector of the first variable to be binned |
| `y` | numeric vector of the second variable to be binned |
| `stopper` | function which accepts a list with elements 'x', 'y', 'bnds', 'expn', and 'n' and returns a logical indicating whether a split should occur for the bin defined by that list |
| `splitter` | function which accepts a list of lists with elements 'x', 'y', 'bnds', 'expn', and 'n' and returns a list where each element is a list of two corresponding to a split of the bin at that position in the original list |
| `init` | function like 'splitter' applied to the sole first bin |

## Details

'binner' creates a two-dimensional histogram of the sample space of 'x' and 'y' by recursively splitting partitions of the data using 'splitter' until 'stopper' indicates that all partitions are not to be split. An optional argument 'init' gives the function applied to the first bin containing all points to initialize the binning algorithm.

## Value

A list of lists each with elements 'x', 'y', 'bnds', 'expn', and 'n'.

## Author(s)

Chris Salahub

## Examples

```
## necessary set up
crits <- makeCriteria(depth >= 4, n < 10, expn <= 5)
stopFn <- function(bns) stopper(bns, crits)
spltFn <- function(bn) maxScoreSplit(bn, chiScores)
## generate data
x <- sample(1:100)
y <- sample(1:100)
## run binner
bins <- binner(x, y, stopper = stopFn, splitter = spltFn)
```

---

# chiScores *Scoring functions to choose splits*

---

## Description

These functions define scores to evaluate candidate splits along a single margin within partition.

## Usage

```
chiScores(vals, expn, minExp = 0)

miScores(vals, expn, minExp = 0)

randScores(vals, expn, minExp = 0)
```

## Arguments

| | |
|---|---|
| vals | numeric vector candidate splits and bounds |
| expn | the expected number of points in the bin |
| minExp | the minimum number of points allowed in a bin |

## Details

Scorings

Each of these functions accepts ‘vals‘, an ordered numeric vector containing the candidate splits within a bin and the bin bounds all in increasing order. To restrict splitting, they also accept ‘expn‘ and ‘minExp‘, which provide the expected count within the split and minimum value of this count, respectively. Any split which produces an expected value less than ‘minExp‘ (assuming a uniform density within the bin) is given a score of zero.

## Value

A vector of scores.

## Functions

- `chiScores()`: A chi-squared statistic score
- `miScores()`: A mutual information score
- `randScores()`: A random score for random splitting

## Author(s)

Chris Salahub

## Examples

```
vals <- c(2, 5, 12, 16, 19)
## restricting the minExp changes output
chiScores(vals, 4, minExp = 0)
chiScores(vals, 4, minExp = 2)
## same for the miScores
miScores(vals, 4, minExp = 0)
miScores(vals, 4, minExp = 2)
## random scoring produces different output every time
randScores(vals, 4, minExp = 0)
randScores(vals, 4, minExp = 0)
```

---

# depthFill *Generate fills encoding bin features*

---

## Description

These functions all accept a list of bins and return a vector of colours of the same length that encode some feature of the bins.

## Usage

```
depthFill(bins, colrng = c("floralwhite", "firebrick"))

residualFill(
  bins,
  resFun = binChi,
  maxRes,
  colrng = c("steelblue", "floralwhite", "firebrick"),
  breaks = NA,
  nbr = 50
)
```

## Arguments

| | |
|---|---|
| bins | list of bins to be visualized |
| colrng | hue range to be passed to ‘colorRampPalette‘ to generate the final hue scale |
| resFun | function which returns a result with a name element ‘residuals‘ that is a numeric vector of the same length as ‘bins‘ |
| maxRes | numeric maximum value of the residuals to maintain the correct origin, taken to be the maximum observed residual if not provided |
| breaks | numeric vector of breakpoints to control hues |
| nbr | number of breakpoints for automatic breakpoint generation if ‘breaks‘ is not provided |

## Details

Shadings

Two functions are provided by default: one which generates a fill based on bin depth and the other based on a residual function applied to each bin.

## Value

A vector of colours the same length as ‘bins‘.

## Functions

- `depthFill()`: Fill by depth
- `residualFill()`: Fill by residual values

## Author(s)

Chris Salahub

## Examples

```
bin <- list(x = 1:10, y = sample(1:10),
            bnds = list(x = c(0, 10), y = c(0, 10)),
            expn = 10, n = 10, depth = 0)
bin2 <- halfSplit(bin, "x")
bin3 <- unlist(lapply(bin2, maxScoreSplit,
                      scorer = chiScores),
               recursive = FALSE)
plotBinning(bin3, fill = depthFill(bin3)) # all the same depth
plotBinning(bin3, fill = residualFill(bin3)) # diff resids
```

---

`halfCutTie` *Halve continuously to break ties*

---

## Description

This function halves a bin based on the midpoint of the bounds along whichever margin produces the larger score.

## Usage

```
halfCutTie(bin, xscore, yscore)
```

## Arguments

| | |
|---|---|
| `bin` | a bin to be split with elements ‘x‘, ‘y‘, ‘depth‘, ‘bnds‘ (list with elements ‘x‘ and ‘y‘), ‘expn‘, ‘n‘ |
| `xscore` | numeric value giving the score for all splits along x |
| `yscore` | numeric value giving the score for all splits along y |

## Details

The goal of this function is to break ties within bin splitting in a way which prevents very small or lopsided bins from forming, a common problem with the ‘halfSplit‘ function

## Value

A list of two bins resulting from the split of ‘bin‘ in half along the margin corresponding to the larger score.

## Author(s)

Chris Salahub

## Examples

```
bin <- list(x = 1:10, y = sample(1:10),
            bnds = list(x = c(0, 10), y = c(0, 10)),
            expn = 10, n = 10, depth = 0)
halfCutTie(bin, 1, 2) # splits on y
halfCutTie(bin, 2, 1) # splits on x
halfCutTie(bin, 1, 1) # ties are random
```

---

`halfSplit` *Halve at an observed point*

---

## Description

This function halves a bin under the restriction that splits can only occur at observation coordinates.

## Usage

```
halfSplit(bin, margin = "x")
```

## Arguments

| | |
|---|---|
| `bin` | a bin to be split with elements ‘x‘, ‘y‘, ‘depth‘, ‘bnds‘ (list with elements ‘x‘ and ‘y‘), ‘expn‘, ‘n‘ |
| `margin` | string, one of ‘x‘ or ‘y‘ |

## Details

Given a bin and a margin, this function splits the bin so half the points are above the new split point and half are below.

## Value

A list of two bins resulting from the split of ‘bin‘ in half along the specified margin

## Author(s)

Chris Salahub

## Examples

```
bin <- list(x = 1:10, y = sample(1:10),
            bnds = list(x = c(0, 10), y = c(0, 10)),
            expn = 10, n = 10, depth = 0)
halfSplit(bin)
halfSplit(bin, margin = "y")
```

`makeCriteria` *Make stop crteria*

## Description

Capture a sequence of logical statements and append them into a single expression.

## Usage

```
makeCriteria(...)
```

## Arguments

| | |
|---|---|
| `...` | an arbitrary number of expressions which evaluate to logicals |

## Details

This function, along with ‘stopper‘ dictates the stop behaviour of recursive binning. It accepts an arbitrary number of arguments, each a logical statement, and appends them all into a string separated by the pipe character.

## Value

A string which appends all expressions together.

## Author(s)

Chris Salahub

## Examples

```
makeCriteria(depth >= 5, n < 1)
```

`maxScoreSplit` *Bivariate score maximizing splitting*

## Description

A function which splits a bin based on the location maximizing a score function.

## Usage

```
maxScoreSplit(bin, scorer, ties = halfCutTie, pickMax = which.max, ...)
```

### Arguments

| | |
|---|---|
| bin | a bin to be split with elements ‘x‘, ‘y‘, ‘depth‘, ‘bnds‘ (list with elements ‘x‘ and ‘y‘), ‘expn‘, ‘n‘ |
| scorer | function which accepts a numeric vector of potential split coordinates and the bounds of ‘bin‘ and returns a numeric vector of scores for each |
| ties | function which is called to break ties when all splits generate the same score |
| pickMax | function which accepts a list of scores and returns the element of the largest score according to some rule |
| ... | optional additional arguments to ‘scorer‘ |

### Details

This function serves as a wrapper which manages the interaction of a score function, marginal splitting functions, tie breaking function, and a maximum selection function to split a bin at the observation coordinate which maximizes the score function.

### Value

A list of two bins resulting from the split of ‘bin‘ along the corresponding margin at the maximum location

### Author(s)

Chris Salahub

### Examples

```
bin <- list(x = 1:10, y = sample(1:10),
            bnds = list(x = c(0, 10), y = c(0, 10)),
            expn = 10, n = 10, depth = 0)
maxScoreSplit(bin, chiScores)
maxScoreSplit(bin, miScores) # pretty similar for both
maxScoreSplit(bin, randScores)
maxScoreSplit(bin, randScores) # different every time
```

---

## plotBinning *Plot a binning using shaded rectangles*

---

### Description

Use a binning and vector of fill colours to visualize the sample space of pairwise data.

### Usage

```
plotBinning(bins, fill, add = FALSE, xlab = "x", ylab = "y", ...)
```

### Arguments

| | |
|---|---|
| bins | list of lists each with a named elements ‘x‘, ‘y‘, and ‘bnds‘, the last of which is a list having named elements ‘x‘ and ‘y‘ |
| fill | vector of values which can be interpreted as colours of the same length as ‘bins‘ |
| add | logical, should the plot of bins be added to the current plot area? |
| xlab | string, the label to be placed on the x axis |
| ylab | string, the label to be placed on the y axis |
| ... | optional additional arguments to be passed to ‘plot‘, ‘points‘ |

### Details

‘plotBinning‘ plots each bin within a list of bins with custom shading to communicate large residuals, the depth of bins, or highlight particular bins

### Value

A list of lists each with elements ‘x‘, ‘y‘, ‘bnds‘, ‘expn‘, and ‘n‘.

### Author(s)

Chris Salahub

### Examples

```
bin <- list(x = 1:10, y = sample(1:10),
            bnds = list(x = c(0, 10), y = c(0, 10)),
            expn = 10, n = 10, depth = 0)
bin2 <- halfSplit(bin, "x")
bin3 <- unlist(lapply(bin2, maxScoreSplit, scorer = chiScores),
               recursive = FALSE)
plotBinning(bin3)
```

---

## sp500pseudo *De-Garched S&P 500 returns*

---

### Description

This data uses code from the 'zenplots' package to process S&P 500 consituent stock returns into uniform pseudo-observations for measuring association.

### Usage

```
data(sp500pseudo)
```

### Format

A matrix with 755 rows and 461 columns, the rows correspond to dates between 2007 and 2009 and the columns correspond to the different S&P 500 constituent stocks.

splitX *Helper functions for marginal splitting*

## Description

These functions are helpers to safely split bins along X or Y.

## Usage

```
splitX(bin, bd, above, below)

splitY(bin, bd, above, below)
```

## Arguments

| | |
|---|---|
| bin | a bin to be split with elements ‘x‘, ‘y‘, ‘depth‘, ‘bnds‘ (list with elements ‘x‘ and ‘y‘), ‘expn‘, ‘n‘ |
| bd | numeric split point within the bin bounds |
| above | indices of ‘x‘ and ‘y‘ points in the bin above ‘bd‘ |
| below | indices of ‘x‘ and ‘y‘ points in the bin below ‘bd‘ |

## Details

These unexported functions have been defined primarily to clean up other code, but could be changed to obtain different core functionality.

## Value

A list of two bins resulting from the split of ‘bin‘ at ‘bds‘.

## Functions

- `splitX()`: Splitting on x
- `splitY()`: Splitting on y

## Author(s)

Chris Salahub

stopper *Check bins against stop criteria*

## Description

Evaluate the stop ‘criteria‘ for each bin in ‘binList‘

## Usage

```
stopper(binList, criteria)
```

## Arguments

| | |
|---|---|
| binList | a list of bins, each a list which can be cast as an environment for evaluation |
| criteria | string of logical expressions separated by pipes to be evaluated within each bin of ‘binList‘ |

## Details

This function makes use of R’s lexical scoping to evaluate ‘criteria‘ (a string), within each bin of ‘binList‘.

## Value

A logical vector of the same length as ‘binList‘.

## Author(s)

Chris Salahub

## Examples

```
crits <- makeCriteria(depth >= 5, n < 1)
binList1 <- list(list(x = c(1,2), y = c(3,1), depth = 1, n = 2),
                 list(x = c(3,4), y = c(2,4), depth = 1, n = 2))
binList2 <- list(list(x = c(1,2), y = c(3,1), depth = 6, n = 2),
                 list(x = c(), y = c(), depth = 1, n = 0))
stopper(binList1, crits)
stopper(binList2, crits)
```

`uniMaxScoreSplit` *Univariate score maximizing splitting*

## Description

A function which splits a bin based on the location maximizing a score function.

## Usage

```
uniMaxScoreSplit(bin, scorer = diff, pickMax = which.max, ...)
```

## Arguments

| | |
|---|---|
| `bin` | a bin to be split with elements ‘x‘, ‘y‘, ‘depth‘, ‘bnds‘ (list with elements ‘x‘ and ‘y‘), ‘expn‘, ‘n‘ |
| `scorer` | function which accepts a numeric vector of potential split coordinates and the bounds of ‘bin‘ and returns a numeric vector of scores for each |
| `pickMax` | function which accepts a list of scores and returns the element of the largest score according to some rule |
| `...` | optional additional arguments to ‘scorer‘ |

## Details

This function is the univariate version of ‘maxScoreSplit‘ and so is considerably simpler. It assumes the variable to be split is named ‘x‘ in the bin, and the other variable is to remain unsplit.

## Value

A list of two bins resulting from the split of ‘bin‘ at the maximum split location along x

## Author(s)

Chris Salahub

# Index