Package ‘binr’

January 9, 2015

Title Cut Numeric Values Into Evenly Distributed Groups

Version 1.1

Author Sergei Izrailev

Maintainer Sergei Izrailev <sizrailev@jabiruventures.com>

Description This package provides algorithms for cutting numerical values exhibiting a potentially highly skewed distribution into evenly distributed groups (bins). This functionality can be applied for binning discrete values, such as counts, as well as for discretization of continuous values, for example, during generation of features used in machine learning algorithms.

URL http://github.com/jabiru/binr

Depends R (>= 2.15),

License Apache License (== 2.0)

Copyright Copyright (C) Collective, Inc. | file inst/COPYRIGHTS

LazyData true

R topics documented:

---

binr — *Cut Numeric Values Into Evenly Distributed Groups (bins).*

Description

Package binr (pronounced as "binner") provides algorithms for cutting numerical values exhibiting a potentially highly skewed distribution into evenly distributed groups (bins). This functionality can be applied for binning discrete values, such as counts, as well as for discretization of continuous values, for example, during generation of features used in machine learning algorithms.

Maintainer

Sergei Izrailev

Copyright

License

Apache License, Version 2.0, available at http://www.apache.org/licenses/LICENSE-2.0

URL

http://github.com/jabiru/binr

Installation from github

```
devtools::install_github("jabiru/binr")
```

Author(s)

Sergei Izrailev

See Also

`bins`, `bins.quantiles`, `bins.optimize`, `bins.greedy`

`bins` *Cut Numeric Values Into Evenly Distributed Groups (Bins)*

Description

`bins` - Cuts points in vector x into evenly distributed groups (bins). `bins` takes 3 separate approaches to generating the cuts, picks the one resulting in the least mean square deviation from the ideal cut - `length(x) / target.bins` points in each bin - and then merges small bins unless excat.groups is `TRUE` The 3 approaches are:

1. Use quantiles, and increase the number of even cuts up to max.breaks until the number of groups reaches the desired number. See `bins.quantiles`.
2. Start with a single bin with all the data in it and perform bin splits until either the desired number of bins is reached or there's no reduction in error (the latter is ignored if `exact.groups` is `TRUE`). See `bins.split`.
3. Start with `length(table(x))` bins, each containing exactly one distinct value and merge bins until the desired number of bins is reached. If `exact.groups` is `FALSE`, continue merging until there's no further reduction in error. See `bins.merge`.

For each of these approaches, apply redistribution of points among existing bins until there's no further decrease in error. See `bins.move`.

`bins.getvals` - Extracts cut points from the object retured by `bins`. The cut points are always between the values in `x` and weighed such that the cut point splits the area under the line from (lo, n1) to (hi, n2) in half.

bins.merr - Partitioning the data into bins using splitting, merging and moving optimizes this error function, which is the mean squared error of point counts in the bins relative to the optimal number of points per bin.

Usage

```
bins(x, target.bins, max.breaks = NA, exact.groups = F, verbose = F,
  errthresh = 0.1, minpts = NA)

bins.getvals(lst, minpt = -Inf, maxpt = Inf)

bins.merr(binct, target.bins)
```

Arguments

x Vector of numbers

target.bins Number of groups desired; this is also the max number of groups.

max.breaks Used for initial cut. If exact.groups is FALSE, bins are merged until there's no bins with fewer than length(x) / max.breaks points. In bins, one of max.breaks and minpts must be supplied.

exact.groups if TRUE, the result will have exactly the number of target.bins; if FALSE, the result may contain fewer than target.bins bins

verbose Indicates verbose output.

errthresh If the error is below the provided value, stops after the first rough estimate of the bins.

minpts Minimum number of points in a bin. In bins, one of max.breaks and minpts must be supplied.

lst The list returned by the bins function.

minpt The value replacing the lower bound of the cut points.

maxpt The value replacing the upper bound of the cut points.

binct The number of points falling into the bins.

Details

The gains are computed using incremental analytical expresions derived for moving a value from one bin to the next, splitting a bin into two or merging two bins.

Value

A list containing the following items (not all of them may be present):

- binlo - The "low" value falling into the bin.
- binhi - The "high" value falling into the bin.
- binct - The number of points falling into the bin.
- xtbl - The result of a call to table(x).
- xval - The sorted unique values of the data points x. Essentially, a numeric version of names(xtbl).
- changed - Flag indicating whether the bins have been modified by the function.
- err - Mean square root error between the resulting counts and ideal bins.

- imax - For the move, merge and split operations, the index of the bin with the maximum gain.
- iside - For the move operation, the side of the move: 0 = left, 1 = right.
- gain - Error gain obtained as the result of the function call.

`bins.getvals` returns a vector of cut points extracted from the `lst` object.

See Also

`binr`, `bins.greedy`, `bins.quantiles` `bins.optimize`

Examples

```
## Not run:
   # Seriously skewed x:
   x <- floor(exp(rnorm(200000 * 1.3)))
   cuts <- bins(x, target.bins = 10, minpts = 2000)
   cuts$breaks <- bins.getvals(cuts)
   cuts$binct
   #   [0, 0]    [1, 1]    [2, 2]    [3, 3]    [4, 4]    [5, 5]    [6, 7]   [8, 10]
   # 129868     66611     28039     13757      7595      4550      4623      2791
   #   [11, 199]
   # 2166

   # Centered x:
   x <- rep(c(1:10,20,31:40), c(rep(1, 10), 100, rep(1,10)))
   cuts <- bins(x, target.bins = 3, minpts = 10)
   cuts$binct
   # [1, 10] [20, 20] [31, 40]
   #      10      100       10

## End(Not run)
```

`bins.greedy` *Greedy binning algorithm.*

Description

`bins.greedy` - Wrapper around `bins.greedy.impl`. Goes over the sorted values of `x` left to right and fills the bins with the values until they are about the right size.

`bins.greedy.impl` - Implementation of a single-pass binning algorithm that examines sorted data left to right and builds bins of the target size. The `bins.greedy` wrapper around this function provides a less involved interface. This is not symmetric wrt direction: symmetric distributions may not have symmetric bins if there are multiple points with the same values. If a single value accounts for more than thresh * binsz points, it will be placed in a new bin.

Usage

```
bins.greedy(x, nbins, minpts = floor(0.5 * length(x)/nbins), thresh = 0.8,
  naive = FALSE)

bins.greedy.impl(xval, xtbl, xstp, binsz, nbins, thresh, verbose = F)
```

Arguments

x — Vector of numbers.

nbins — Target number of bins.

minpts — Minimum number of points in a bin. Only used if naive = FALSE.

thresh — Threshold fraction of bin size for the greedy algorithm. Suppose there's n < binsz points in the current bin already. Also suppose that the next value V is represented by m points, and m + n > binsz. Then the algorithm will check if m > thresh * binsz, and if so, will place the value V into a new bin. If m is below the threshold, the points having value V are added to the current bin.

naive — When TRUE, simply calls bins.greedy.impl with data derived from x. Otherwise, makes an extra step of marking the values that by themselves take a whole bin to force the algorithm to place these values in a bin separately.

xval — Sorted unique values of the data set x. This should be the numeric version of names(xtbl).

xtbl — Result of a call to table(x).

xstp — Stopping points; if xstp[i] == TRUE, the i-th value can't be merged to the (i-1)-th one. xstp[1] value is ignored.

binsz — Target bin size, i.e., the number of points falling into each bin; for example, floor(length(x) / nbins)

verbose — When TRUE, prints the number of points falling into the bins.

Value

A list with the following items:

- binlo - The "low" value falling into the bin.
- binhi - The "high" value falling into the bin.
- binct - The number of points falling into the bin.
- xtbl - The result of a call to table(x).
- xval - The sorted unique values of the data points x. Essentially, a numeric version of names(xtbl).

See Also

binr, bins, bins.quantiles bins.optimize

bins.optimize *Algorithms minimizing the binning error function* bins.merr.

Description

bins.move - Compute the best move of a value from one bin to its neighbor

bins.split - Split a bin into two bins optimally.

bins.merge - Merges the two bins yielding the largest gain in error reduction.

bins.move.iter - Apply bins.move until there's no change. Can only reduce the error.

bins.split.iter Iterate to repeatedly apply bins.split.

bins.merge.iter Iterate to repeatedly apply bins.merge.

Usage

```
bins.move(xval, xtbl, binlo, binhi, binct, target.bins, verbose = F)

bins.split(xval, xtbl, binlo, binhi, binct, target.bins, force = F,
  verbose = F)

bins.merge(xval, xtbl, binlo, binhi, binct, target.bins, force = F,
  verbose = F)

bins.move.iter(lst, target.bins, verbose = F)

bins.split.iter(lst, target.bins, exact.groups = F, verbose = F)

bins.merge.iter(lst, target.bins, exact.groups = F, verbose = F)
```

Arguments

`xval` Sorted unique values of the data set x. This should be the numeric version of `names(xtbl)`.

`xtbl` Result of a call to `table(x)`.

`binlo` The "low" value falling into the bin.

`binhi` The "high" value falling into the bin.

`binct` The number of points falling into the bin.

`target.bins` Number of bins desired; this is also the max number of bins.

`verbose` When `TRUE`, prints resulting `binct`.

`force` When `TRUE`, splits or merges bins regardless of whether the best gain is positive.

`lst` List containing `xval, xtbl, binlo, binhi, binct`.

`exact.groups` If `FALSE`, run until either the target.bins is reached or there's no more splits or merges that reduce the error. Otherwise (`TRUE`), run until the target.bins is reached, even if that increases the error.

Value

A list containing the following items (not all of them may be present):

- binlo - The "low" value falling into the bin.
- binhi - The "high" value falling into the bin.
- binct - The number of points falling into the bin.
- xtbl - The result of a call to `table(x)`.
- xval - The sorted unique values of the data points x. Essentially, a numeric version of `names(xtbl)`.
- changed - Flag indicating whether the bins have been modified by the function.
- err - Mean square root error between the resulting counts and ideal bins.
- imax - For the move, merge and split operations, the index of the bin with the maximum gain.
- iside - For the move operation, the side of the move: 0 = left, 1 = right.
- gain - Error gain obtained as the result of the function call.

See Also

`bins`, `binr`, `bins.greedy`, `bins.quantiles`

`bins.quantiles` *Quantile-based binning*

Description

Cuts the data set x into roughly equal groups using quantiles.

Usage

```
bins.quantiles(x, target.bins, max.breaks, verbose = FALSE)
```

Arguments

`x`	A numeric vector to be cut in bins.
`target.bins`	Target number of bins, which may not be reached if the number of unique values is smaller than the specified value.
`max.breaks`	Maximum number of quantiles; must be at least as large as `target.bins`.
`verbose`	Indicates verbose output.

Details

Because the number of unique values may be smaller than target.bins, the function gradually increases the number of quantiles up to max.breaks or until the target.bins number of bins is reached.

See Also

`binr`, `bins`, `bins.greedy`, `bins.optimize`

Index