Package 'classInt'

September 29, 2015

Version 0.1-23

Date 2015-09-27

Title Choose Univariate Class Intervals

Depends R (>= 2.2)

Imports grDevices, stats, graphics, e1071, class

Description Selected commonly used methods for choosing univariate class intervals for mapping or other graphics purposes.

License GPL (>= 2)

NeedsCompilation yes

Author Roger Bivand [aut, cre],
Hisaji Ono [ctb],
Richard Dunlap [ctb],
Matthieu Stigler [ctb]

Maintainer Roger Bivand <Roger.Bivand@nhh.no>

Repository CRAN

Date/Publication 2015-09-29 01:16:28

R topics documented:

`classIntervals` *Choose univariate class intervals*

Description

The function provides a uniform interface to finding class intervals for continuous numerical variables, for example for choosing colours or symbols for plotting. Class intervals are non-overlapping, and the classes are left-closed — see `findInterval`. Argument values to the style chosen are passed through the dot arguments. `classIntervals2shingle` converts a `classIntervals` object into a shingle. Labels generated in methods are like those found in `cut` unless cutlabels=FALSE.

Usage

```
classIntervals(var, n, style = "quantile", rtimes = 3, ...,
 intervalClosure = c("left", "right"), dataPrecision = NULL)
## S3 method for class 'classIntervals'
plot(x, pal, ...)
## S3 method for class 'classIntervals'
print(x, digits = getOption("digits"), ...,
 under="under", over="over", between="-", cutlabels=TRUE, unique=FALSE)
nPartitions(x)
classIntervals2shingle(x)
```

Arguments

`var`	a continuous numerical variable
`n`	number of classes required, if missing, `nclass.Sturges` is used
`style`	chosen style: one of "fixed", "sd", "equal", "pretty", "quantile", "kmeans", "hclust", "bclust", "fisher", or "jenks"
`rtimes`	number of replications of var to catenate and jitter; may be used with styles "kmeans" or "bclust" in case they have difficulties reaching a classification
`intervalClosure`	default “left”, allows specification of whether partition intervals are closed on the left or the right (added by Richard Dunlap). Note that the sense of interval closure is hard-coded as “right”-closed when`style="jenks"` (see Details below).
`dataPrecision`	default NULL, permits rounding of the interval endpoints (added by Richard Dunlap)
`...`	arguments to be passed to the functions called in each style
`x`	"classIntervals" object for printing, conversion to shingle, or plotting
`under`	character string value for "under" in printed table labels if cutlabels=FALSE
`over`	character string value for "over" in printed table labels if cutlabels=FALSE
`between`	character string value for "between" in printed table labels if cutlabels=FALSE
`digits`	minimal number of significant digits in printed table labels

cutlabels	default TRUE, use cut-style labels in printed table labels
unique	default FALSE; if TRUE, collapse labels of single-value classes
pal	a character vector of at least two colour names for colour coding the class intervals in an ECDF plot; colorRampPalette is used internally to create the correct number of colours

Details

The "fixed" style permits a "classIntervals" object to be specified with given breaks, set in the fixedBreaks argument; the length of fixedBreaks should be n+1; this style can be used to insert rounded break values.

The "sd" style chooses breaks based on pretty of the centred and scaled variables, and may have a number of classes different from n; the returned par= includes the centre and scale values.

The "equal" style divides the range of the variable into n parts.

The "pretty" style chooses a number of breaks not necessarily equal to n using pretty, but likely to be legible; arguments to pretty may be passed through ....

The "quantile" style provides quantile breaks; arguments to quantile may be passed through ....

The "kmeans" style uses kmeans to generate the breaks; it may be anchored using set.seed; the pars attribute returns the kmeans object generated; if kmeans fails, a jittered input vector containing rtimes replications of var is tried — with few unique values in var, this can prove necessary; arguments to kmeans may be passed through ....

The "hclust" style uses hclust to generate the breaks using hierarchical clustering; the pars attribute returns the hclust object generated, and can be used to find other breaks using getHclustClassIntervals; arguments to hclust may be passed through ....

The "bclust" style uses bclust to generate the breaks using bagged clustering; it may be anchored using set.seed; the pars attribute returns the bclust object generated, and can be used to find other breaks using getBclustClassIntervals; if bclust fails, a jittered input vector containing rtimes replications of var is tried — with few unique values in var, this can prove necessary; arguments to bclust may be passed through ....

The "fisher" style uses the algorithm proposed by W. D. Fisher (1958) and discussed by Slocum et al. (2005) as the Fisher-Jenks algorithm; added here thanks to Hisaji Ono.

The "jenks" style has been ported from Jenks' Basic code, and has been checked for consistency with ArcView, ArcGIS, and MapInfo (with some remaining differences); added here thanks to Hisaji Ono. Note that the sense of interval closure is reversed from the other styles, and in this implementation has to be right-closed - use cutlabels=TRUE in findColours on the object returned to show the closure clearly, and use findCols to extract the classes for each value.

Value

an object of class "classIntervals":

var	the input variable
brks	a vector of breaks

and attributes:

style	the style used
parameters	parameter values used in finding breaks
nobs	number of different finite values in the input variable
call	this function's call
intervalClosure	string, whether closure is "left" or "right"
dataPrecision	the data precision used for printing interval values in the legend returned by findColours, and in the print method for classIntervals objects. If intervalClosure is "left", the value returned is ceiling of the data value multiplied by 10 to the dataPrecision power, divided by 10 to the dataPrecision power.

Note

From version 0.1-11, the default representation has been changed to use cutlabels=TRUE, and representation within intervals has been corrected, thanks to Richard Dunlap. From version 0.1-15, the print method drops the calculation of the possible number of combinations of observations into classes, which generated warnings for n > 170.

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

References

Armstrong, M. P., Xiao, N., Bennett, D. A., 2003. "Using genetic algorithms to create multicriteria class intervals for choropleth maps". Annals, Association of American Geographers, 93 (3), 595–623;

Jenks, G. F., Caspall, F. C., 1971. "Error on choroplethic maps: definition, measurement, reduction". Annals, Association of American Geographers, 61 (2), 217–244;

Dent, B. D., 1999, Cartography: thematic map design. McGraw-Hill, Boston, 417 pp.;

Slocum TA, McMaster RB, Kessler FC, Howard HH 2005 Thematic Cartography and Geographic Visualization, Prentice Hall, Upper Saddle River NJ.;

Fisher, W. D. 1958 "On grouping for maximum homogeneity", Journal of the American Statistical Association, 53, pp. 789–798 (http://lib.stat.cmu.edu/cmlib/src/cluster/fish.f)

See Also

findColours, findCols, pretty, quantile, kmeans, hclust, bclust, findInterval, colorRamp, nclass, shingle

Examples

```
data(jenks71)
pal1 <- c("wheat1", "red3")
opar <- par(mfrow=c(2,3))
plot(classIntervals(jenks71$jenks71, n=5, style="fixed",
 fixedBreaks=c(15.57, 25, 50, 75, 100, 155.30)), pal=pal1, main="Fixed")
```

```
plot(classIntervals(jenks71$jenks71, n=5, style="sd"), pal=pal1, main="Pretty standard deviations")
plot(classIntervals(jenks71$jenks71, n=5, style="equal"), pal=pal1, main="Equal intervals")
plot(classIntervals(jenks71$jenks71, n=5, style="quantile"), pal=pal1, main="Quantile")
set.seed(1)
plot(classIntervals(jenks71$jenks71, n=5, style="kmeans"), pal=pal1, main="K-means")
plot(classIntervals(jenks71$jenks71, n=5, style="hclust", method="complete"),
 pal=pal1, main="Complete cluster")
plot(classIntervals(jenks71$jenks71, n=5, style="hclust", method="single"),
 pal=pal1, main="Single cluster")
set.seed(1)
plot(classIntervals(jenks71$jenks71, n=5, style="bclust", verbose=FALSE),
 pal=pal1, main="Bagged cluster")
plot(classIntervals(jenks71$jenks71, n=5, style="fisher"), pal=pal1, main="Fisher's method")
plot(classIntervals(jenks71$jenks71, n=5, style="jenks"), pal=pal1, main="Jenks' method")
par(opar)
classIntervals(jenks71$jenks71, n=5, style="fixed", fixedBreaks=c(15.57, 25, 50, 75, 100, 155.30))
classIntervals(jenks71$jenks71, n=5, style="sd")
classIntervals(jenks71$jenks71, n=5, style="equal")
classIntervals(jenks71$jenks71, n=5, style="quantile")
set.seed(1)
classIntervals(jenks71$jenks71, n=5, style="kmeans")
set.seed(1)
classIntervals(jenks71$jenks71, n=5, style="kmeans", intervalClosure="right")
set.seed(1)
classIntervals(jenks71$jenks71, n=5, style="kmeans", dataPrecision=0)
set.seed(1)
print(classIntervals(jenks71$jenks71, n=5, style="kmeans"), cutlabels=FALSE)
classIntervals(jenks71$jenks71, n=5, style="hclust", method="complete")
classIntervals(jenks71$jenks71, n=5, style="hclust", method="single")
set.seed(1)
classIntervals(jenks71$jenks71, n=5, style="bclust", verbose=FALSE)
classIntervals(jenks71$jenks71, n=5, style="bclust", hclust.method="complete", verbose=FALSE)
classIntervals(jenks71$jenks71, n=5, style="fisher")
classIntervals(jenks71$jenks71, n=5, style="jenks")
x <- c(0, 0, 0, 1, 2, 50)
classIntervals(x, n=3, style="fisher")
classIntervals(x, n=3, style="jenks")

# Argument 'unique' will collapse the label of classes containing a
# single value. This is particularly useful for 'censored' variables
# that contain for example many zeros.

data_censored<-c(rep(0,10), rnorm(100, mean=20,sd=1),rep(26,10))
plot(density(data_censored))

cl2<-classIntervals(data_censored, n=5, style="jenks", dataPrecision=2)

print(cl2, unique=FALSE)
print(cl2, unique=TRUE)
```

`findColours` *assign colours to classes from classInterval object*

Description

This helper function is a wrapper for `findCols` to extract classes from a "classInterval" object and assign colours from a palette created by `colorRampPalette` from the two or more colours given in the `pal` argument. It also returns two attributes for use in constructing a legend.

Usage

```
findColours(clI, pal, under="under", over="over", between="-",
 digits = getOption("digits"), cutlabels=TRUE)
```

Arguments

`clI`	a "classIntervals" object
`pal`	a character vector of at least two colour names; `colorRampPalette` is used internally to create the required number of colours
`under`	character string value for "under" in legend if cutlabels=FALSE
`over`	character string value for "over" in legend if cutlabels=FALSE
`between`	character string value for "between" in legend if cutlabels=FALSE
`digits`	minimal number of significant digits in legend
`cutlabels`	use cut-style labels in legend

Value

a character vector of colours with attributes: "table", a named frequency table; "palette", a character vector of colours corresponding to the specified breaks.

Author(s)

Roger Bivand <Roger.Bivand@nhh.no

See Also

`classIntervals`, `findInterval`, `findCols`, `colorRamp`

Examples

```
data(jenks71)
mypal <- c("wheat1", "red3")
h5 <- classIntervals(jenks71$jenks71, n=5, style="hclust", method="complete")
findColours(h5, mypal)
findColours(getHclustClassIntervals(h5, k=7), mypal)
h5Colours <- findColours(h5, mypal)
plot(h5, mypal, main="Complete hierarchical clustering")
legend(c(95, 155), c(0.12, 0.4), fill=attr(h5Colours, "palette"),
```

```
 legend=names(attr(h5Colours, "table")), bg="white")
h5tab <- attr(h5Colours, "table")
legtext <- paste(names(h5tab), " (", h5tab, ")", sep="")
plot(h5, mypal, main="Complete hierarchical clustering (with counts)")
legend(c(95, 165), c(0.12, 0.4), fill=attr(h5Colours, "palette"),
 legend=legtext, bg="white")
```

---

findCols *extract classes from classInterval object*

Description

This helper function is a wrapper for `findInterval` to extract classes from a "classInterval" object

Usage

```
findCols(clI)
```

Arguments

`clI`	a "classIntervals" object

Value

an integer vector of class indices

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

See Also

`classIntervals`, `findInterval`

Examples

```
data(jenks71)
fix5 <- classIntervals(jenks71$jenks71, n=5, style="fixed",
 fixedBreaks=c(15.57, 25, 50, 75, 100, 155.30))
fix5
findCols(fix5)
```

getBclustClassIntervals *Change breaks in a "classIntervals" object*

Description

Because "classIntervals" objects of style "hclust" or "bclust" contain hierarchical classification trees in their "par" attribute, different numbers of classes can be chosen without repeating the initial classification. This function accesses the "par" attribute and modifies the "brks" member of the returned "classIntervals" object.

Usage

```
getBclustClassIntervals(clI, k)
getHclustClassIntervals(clI, k)
```

Arguments

clI	a "classIntervals" object
k	number of classes required

Value

a "classIntervals" object with a "modified" attribute set

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

See Also

classIntervals

Examples

```
data(jenks71)
pal1 <- c("wheat1", "red3")
opar <- par(mfrow=c(2,2))
hCI5 <- classIntervals(jenks71$jenks71, n=5, style="hclust", method="complete")
plot(attr(hCI5, "par"))
plot(hCI5, pal=pal1, main="hclust k=5")
plot(getHclustClassIntervals(hCI5, k=7), pal=pal1, main="hclust k=7")
plot(getHclustClassIntervals(hCI5, k=9), pal=pal1, main="hclust k=9")
par(opar)
set.seed(1)
bCI5 <- classIntervals(jenks71$jenks71, n=5, style="bclust")
plot(attr(bCI5, "par"))
opar <- par(mfrow=c(2,2))
plot(getBclustClassIntervals(bCI5, k=3), pal=pal1, main="bclust k=3")
```

```
plot(bCI5, pal=pal1, main="bclust k=5")
plot(getBclustClassIntervals(bCI5, k=7), pal=pal1, main="bclust k=7")
plot(getBclustClassIntervals(bCI5, k=9), pal=pal1, main="bclust k=9")
par(opar)
```

`jenks.tests` *Indices for assessing class intervals*

Description

The function returns values of two indices for assessing class intervals: the goodness of variance fit measure, and the tabular accuracy index; optionally the overview accuracy index is also returned if the `area` argument is not missing.

Usage

```
jenks.tests(clI, area)
```

Arguments

`clI`	a "classIntervals" object
`area`	an optional vector of object areas if the overview accuracy index is also required

Details

The goodness of variance fit measure is given by Armstrong et al. (2003, p. 600) as:

$$GVF = 1 - \frac{\sum_{j=1}^{k}\sum_{i=1}^{N_j}\left(z_{ij} - \bar{z}_j\right)^2}{\sum_{i=1}^{N}\left(z_i - \bar{z}\right)^2}$$

where the $z_i, i = 1, \ldots, N$ are the observed values, k is the number of classes, $\bar{z}_j$ the class mean for class j, and N_j the number of counties in class j.

The tabular accuracy index is given by Armstrong et al. (2003, p. 600) as:

$$TAI = 1 - \frac{\sum_{j=1}^{k}\sum_{i=1}^{N_j}\left|z_{ij} - \bar{z}_j\right|}{\sum_{i=1}^{N}\left|z_i - \bar{z}\right|}$$

The overview accuracy index for polygon observations with known areas is given by Armstrong et al. (2003, p. 600) as:

$$OAI = 1 - \frac{\sum_{j=1}^{k}\sum_{i=1}^{N_j}\left|z_{ij} - \bar{z}_j\right|a_{ij}}{\sum_{i=1}^{N}\left|z_i - \bar{z}\right|a_i}$$

where $a_i, i = 1, \ldots, N$ are the polygon areas, and as above the a_{ij} term is indexed over $j = 1, \ldots, k$ classes, and $i = 1, \ldots, N_j$ polygons in class j.

Value

a named vector of index values

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

References

Armstrong, M. P., Xiao, N., Bennett, D. A., 2003. "Using genetic algorithms to create multicriteria class intervals for choropleth maps". Annals, Association of American Geographers, 93 (3), 595–623; Jenks, G. F., Caspall, F. C., 1971. "Error on choroplethic maps: definition, measurement, reduction". Annals, Association of American Geographers, 61 (2), 217–244

See Also

classIntervals

Examples

```
data(jenks71)
fix5 <- classIntervals(jenks71$jenks71, n=5, style="fixed",
 fixedBreaks=c(15.57, 25, 50, 75, 100, 155.30))
jenks.tests(fix5, jenks71$area)
q5 <- classIntervals(jenks71$jenks71, n=5, style="quantile")
jenks.tests(q5, jenks71$area)
set.seed(1)
k5 <- classIntervals(jenks71$jenks71, n=5, style="kmeans")
jenks.tests(k5, jenks71$area)
h5 <- classIntervals(jenks71$jenks71, n=5, style="hclust", method="complete")
jenks.tests(h5, jenks71$area)
jenks.tests(getHclustClassIntervals(h5, k=7), jenks71$area)
jenks.tests(getHclustClassIntervals(h5, k=9), jenks71$area)
set.seed(1)
b5 <- classIntervals(jenks71$jenks71, n=5, style="bclust")
jenks.tests(b5, jenks71$area)
jenks.tests(getBclustClassIntervals(b5, k=7), jenks71$area)
jenks.tests(getBclustClassIntervals(b5, k=9), jenks71$area)
```

---

jenks71 *Illinois 1959 county gross farm product value per acre*

Description

Classic data set for the choice of class intervals for choropleth maps.

Usage

```
data(jenks71)
```

Format

A data frame with 102 observations on the following 2 variables.

`jenks71` a numeric vector: Per acre value of gross farm products in dollars by county for Illinois in 1959

`area` a numeric vector: county area in square miles

Source

Jenks, G. F., Caspall, F. C., 1971. "Error on choroplethic maps: definition, measurement, reduction". Annals, Association of American Geographers, 61 (2), 217–244

Examples

```
data(jenks71)
(jenks71)
```

Index