

Package ‘YaleToolkit’

April 17, 2009

Type Package

Title Data exploration tools from Yale University.

Version 3.1

Date 2007-10-24

Author John W. Emerson and Walton Green

Maintainer John W. Emerson <john.emerson@yale.edu>

Depends grid, lattice, vcd, MASS, colorspace

Description This collection of data exploration tools was developed at Yale University for the graphical exploration of complex multivariate data. Functions provided include `barcode()`, `gpairs()`, `corrgram()`, `whatis()`, `sparkline()`, `sparklines()`, and `sparkmat()`.

License GPL (>= 2)

URL <http://www.stat.yale.edu/~jay/R/YaleToolkit>

Repository CRAN

Date/Publication 2007-11-03 18:20:00

R topics documented:

YaleToolkit-package	2
barcode	3
gpairs	5
Leaves	9
nasa	11
NewHavenResidential	12
sparkline	13
sparklines	16
sparkmat	19
whatis	22
YaleEnergy	24

YaleToolkit-package

Data exploration tools from the Department of Statistics at Yale University

Description

This collection of data exploration tools was developed at Yale University for the graphical exploration of complex multivariate data. The main functions provided are `barcode()`, `gpairs()`, `whatis()`, and `sparkmat()`.

Details

Package: YaleToolkit
Type: Package
Version: 3.1
Date: 2007-10-24
License: GPL version 2 or newer

See individual help files for more information about the functions `barcode()`, `gpairs()`, `corrgram()`, `whatis()`, `sparkmat()`, `sparklines()`, `sparkline()`.

The package also includes three data sets. For more information, please see the help files for `NewHavenResidential`, `Leaves`, and `YaleEnergy`.

Non-default options to these functions may need debugging and further development, and we maintain our “todo” list here:

`gpairs()`: `xylim` can cause cropping (or clipping), which fails to work with `barcode`. This difficulty with rotation and clipping.
`gpairs()`: more Friendly (2002) options to `corrgram`; without support for ellipses in grid, we’re tabling this for the moment.
`passim`: make use of single and double quotation marks consistent.

Version 3.1 is the second (beta, if you will) version released to CRAN. Please get in touch with us if you note any problems.

Note that some terminology from Paul Murrell’s `grid` package is used throughout the documentation. Users unfamiliar with `grid` concepts should probably look at `unit` before reading our help files.

Author(s)

John W. Emerson, Walton Green

Maintainer: John W. Emerson <john.emerson@yale.edu>

References

Chambers, J.M., Cleveland, W.S., Kleiner, B., and Tukey, P.A. (1983), *Graphical Methods for Data Analysis*, Belmont, CA: Wadsworth.

Friendly, M. (2002) 'Corrgrams: Exploratory displays for correlation matrices' *American Statistician* 56(4), 316–324.

Tufte, Edward R. (2006) *Beautiful Evidence* The Graphics Press, Cheshire, Connecticut. See <http://www.edwardtufte.com> for this and other references.

See Also

[barcode](#), [gpairs](#), [sparkmat](#), [sparklines](#), [sparkline](#), [whatis](#), [corrgram](#).

Examples

```
# See individual function documentation for examples.
```

barcode

Barcode plots

Description

Produce barcode plot(s) of the given (grouped) values.

Usage

```
barcode(x, outer.margins = list(bottom = unit(2, "lines"),
                                left = unit(2, "lines"),
                                top = unit(2, "lines"),
                                right = unit(2, "lines")),
        horizontal = TRUE, xlim = NULL, nint = 0, main = "", xlab = "",
        labelloc = TRUE, axisloc = TRUE, labelouter = FALSE,
        newpage = TRUE, fontsize = 9, ptsize = unit(0.25, "char"),
        ptpch = 1, bcspace = NULL, use.points = FALSE, buffer = 0.02,
        log = FALSE)
```

```
barcode.panel(x, horizontal = TRUE, xlim = NULL, labelloc = TRUE, axisloc = TRUE,
             labelouter = FALSE, nint = 0, fontsize = 9,
             ptsize = unit(0.25, "char"), ptpch = 1, bcspace = NULL,
             xlab = "", xlaboffset = unit(2.5, "lines"),
             use.points = FALSE, buffer = 0.02, log = FALSE)
```

Arguments

<code>x</code>	a vector of values for which the barcode is desired, or a list of such vectors for “side-by-side” barcodes. Matrices are coerced to data frames and treated as lists NA’s are allowed in the data.
<code>outer.margins</code>	a list of length 4 with units as components named bottom, left, top, and right, giving the outer margins. Defaults to two lines of text.
<code>horizontal</code>	logical indicating the barcode orientation; the default, TRUE, produces horizontal barcodes.
<code>xlim</code>	the x limits (<code>xmin</code> , <code>xmax</code>) of the plot; the default, NULL, uses the range of the full data, <code>range(unlist(x))</code> , plus the multiplicative <code>buffer</code> .
<code>nint</code>	default, 0, uses no “binning”— i.e., the barcode presents the exact measurements, to the precision of the data set; <code>nint=100</code> uses roughly 100 “bins” in constructing the barcode; fewer bins give a more histogram-like plot.
<code>main</code>	the plot title.
<code>xlab</code>	the axis label for the quantitative measurements.
<code>labelloc</code>	for the location of the factor labels of the barcodes; default TRUE may also be specified as ‘left’ or ‘top’ (having similar results but relating to the horizontal alignment); values ‘right’ or ‘bottom’ are available as alternatives to FALSE.
<code>axisloc</code>	for the location of the quantitative axis labels; default, TRUE, may also be specified as ‘left’ or ‘top’ (having similar results but relating to the horizontal alignment); values ‘right’ or ‘bottom’ are available as alternatives to FALSE.
<code>labelouter</code>	default, FALSE, positions all labels within the viewport; TRUE forces the barcodes to the edge of the viewport, with the labels outside the viewport. May be of use to advanced users.
<code>newpage</code>	default, TRUE, creates the barcodes in a new graphics device instead of adding the plot to the current viewport.
<code>fontsize</code>	for the size of the axis and factor labels.
<code>ptsize</code>	for the size of the plotted points.
<code>ptpch</code>	for the type of plotted points.
<code>bcspace</code>	indicates the proportion of total available space occupied by the barcode part of the displays. Can range from 0 to 1; reasonable values seem to be between 0.1 and 0.5.
<code>use.points</code>	default FALSE uses segments instead of points in the histogram-style display.
<code>xlaboffset</code>	used for tuning the position of the label of the quantitative variable; needs to be a unit.
<code>buffer</code>	an additional proportion of empty space added to the right and left of the barcode, to avoid having the maximum and minimum on the frame of the plot.
<code>log</code>	if TRUE, use the log scale for the y-axis of the histogram-like part of the barcodes.

Details

The barcode plot aids in comparing distributions. It shares some of the characteristics of side-by-side histograms or boxplots, and of rugs or stripplots. We have found it particularly useful with clumped data, when other methods obscure detail.

Note

John Hartigan designed and implemented an early version of the barcode plot. This implementation using `grid` graphics adds some useful options and is better suited for general distribution.

Author(s)

John W. Emerson and Walton Green

References

Chambers, J. M. and Hastie, T. J. (1992) *Statistical Models in S*. Wadsworth & Brooks/Cole.

See Also

[gpairs](#), [rug](#), [stripplot](#)

Examples

```
# Simulate some data:
x <- list(Rounded.2=round(rnorm(500, 2, 1),2),
         SmallerLevel=c(rnorm(100), rnorm(100,4,1)),
         LargerBivariateRounded.4=round(c(rnorm(500), rnorm(500,3,1)),4))

barcode(x)
barcode(x, main="Different orientatation", horizontal=FALSE)

data(NewHavenResidential)
barcode(split(NewHavenResidential$dep, NewHavenResidential$zone),
        xlab="Percent Depreciation",
        main=paste("New Haven Residential Depreciation by Residential Zone",
                  "RS = Single Family, RM = Mixed Residential", sep = "\n"))
```

gpairs

Generalized Pairs Plots

Description

Produces a matrix of plots showing pairwise relationships between quantitative and categorical variables in a complex data set.

Usage

```
gpairs(x,
       upper.pars = list(scatter = "points",
                        conditional = "barcode",
                        mosaic = "mosaic"),
       lower.pars = list(scatter = "points",
                        conditional = "boxplot",
                        mosaic = "mosaic"),
       diagonal = "default",
       outer.margins = list(bottom = unit(2, "lines"),
                            left = unit(2, "lines"),
                            top = unit(2, "lines"),
                            right = unit(2, "lines")),
       xlim = NULL,
       outer.labels = NULL, outer.rot = c(90, 0), gap = 0.05,
       buffer = 0.02, reorder = NULL, cluster.pars = NULL,
       stat.pars = NULL, scatter.pars = NULL,
       bwplot.pars = NULL, stripplot.pars = NULL, barcode.pars=NULL,
       mosaic.pars = NULL, axis.pars = NULL, diag.pars = NULL,
       whatis = FALSE)

corrgram(x)
```

Arguments

<code>x</code>	a data frame (or matrix the relationships between whose columns are to be examined). Any combination of quantitative and categorical variables is acceptable.
<code>upper.pars</code>	see Details
<code>lower.pars</code>	see Details
<code>diagonal</code>	by default, the diagonal from the top left to the bottom right is used for displaying the variable names (and, in our version, the marginal distributions of the variables); <code>diagonal="other"</code> will place the diagonal running from the top right down to the bottom left.
<code>outer.margins</code>	a list of length 4 with units as components named bottom, left, top, and right, giving the outer margins; the default uses two lines of text. A vector of length 4 with units (ordered properly) will work, as will a vector of length 4 with numeric values (interpreted as lines).
<code>xylim</code>	optionally specify a single range to be used as <code>xlim</code> and <code>ylim</code> where appropriate. Note that if this option causes cropping, it will fail to work in barcode panels.
<code>outer.labels</code>	the default is <code>NULL</code> , for alternating axis labels around the perimeter. If <code>"all"</code> , all labels are printed, and if <code>"none"</code> no labels are printed.
<code>outer.rot</code>	a 2-vector (x, y) rotating the top/bottom outer labels <code>x</code> degrees and the left/right outer labels <code>y</code> degrees. Only works for categorical labels of boxplot and mosaic panels.

gap	the gap between the tiles; defaulting to 0.05 of the width of a tile.
buffer	the fraction by which to expand the range of quantitative variables to provide plots that will not truncate plotting symbols. Defaults to 0 percent of range currently.
reorder	currently only support for the string "cluster", which reorders the columns according to the output of <code>hclust</code> . Note that factors are coerced to numbers by replacing them with integers, which implicitly assumes what is probably an arbitrary ordering.
cluster.pars	a list with two elements named <code>dist.method</code> and <code>hclust.method</code> . These are passed respectively to <code>dist</code> and <code>hclust</code> . NULL is equivalent to <code>list(dist.method = "euclidean", hclust.method = "complete")</code> .
stat.pars	NULL is equivalent to <code>list(fontsize = 7, signif = 0.05, verbose = FALSE, use.color = TRUE, missing = 'missing', just = 'centre')</code> ; <code>stat.pars\$verbose</code> can be TRUE (providing 5 statistics), FALSE (providing 2 statistics), or NA (nothing). The string <code>missing</code> is used in summaries where there are missing values; <code>fontsize</code> and <code>just</code> control the size and justification of the text summaries (see <code>grid.text</code> and <code>gpar</code> . The <code>use.color=FALSE</code> option provides an alternative summary of the strength of the correlation (see Green and Hickey (2006)). This is only used with <code>scatter="stats"</code> in <code>upper.pars</code> and <code>lower.pars</code> .
scatter.pars	NULL is equivalent to <code>list(pch = 1, size = unit(0.25, "char"), col = "black", frame.fill = NULL, border.col = "black")</code> .
bwplot.pars	NULL, passed to <code>bwplot</code> for producing boxplots.
stripplot.pars	NULL is equivalent to <code>list(pch = 1, size = unit(0.5, 'char'), col = 'black', jitter = FALSE)</code> .
barcode.pars	NULL is equivalent to <code>list(nint = 0, ptsize = unit(0.25, "char"), ptpch = 1, bcspace = NULL, use.points = FALSE)</code> .
mosaic.pars	NULL. Currently, only <code>shade</code> and <code>gp_labels</code> are passed through to <code>strucplot</code> for producing mosaic tiles.
axis.pars	NULL is equivalent to <code>list(n.ticks = 5, fontsize = 9)</code> .
diag.pars	NULL is equivalent to <code>list(fontsize = 9, show.hist = TRUE, hist.color = 'black')</code> .
whatis	default is FALSE; TRUE returns <code>whatis(x)</code> .

Details

In some cases, the graphics device can not be resized after production of the plot because of the way rotation of barcodes is performed.

`upper.pars` and `lower.pars` are lists possibly containing named elements `'scatter'`, `'conditional'` and `'mosaic'`. Each element of the list is a string implementing the following options: `scatter = exactly one of ('points', 'lm', 'ci', 'symlm', 'loess', 'corrgram', 'stats', 'qqplot')`; `conditional = exactly one of ('boxplot', 'stripplot', 'barcode')`; `mosaic='mosaic'` (only option currently implemented).

`corrgram()` is just a wrapper to `gpairs()` producing a `'corrgram'` in the style of Michael Friendly.

Value

If `what.is=TRUE`, the value is a data frame containing variable names, types, numbers of missing values, numbers of distinct values, precisions, maxima and minima.

Author(s)

John W. Emerson, Walton Green

References

Emerson, John W. (1998) "Mosaic Displays in S-PLUS: A General Implementation and a Case Study." *Statistical Computing and Graphics Newsletter* Vol. 9, No. 1, 1998.

Basford, K. E. and J. W. Tukey (1999) *Graphical Analysis of Multiresponse Data: Illustrated with a Plant Breeding Trial*.

Friendly, M. (2000). *Visualizing Categorical Data*. SAS Press.

Friendly, M., 2002, 'Corrgrams: Exploratory displays for correlation matrices.' *American Statistician* 56(4), 316–324.

Green, W. A. (2006) Loosening the CLAMP: An exploratory graphical approach to the Climate Leaf Analysis Multivariate Program *Palaeontologia Electronica* 9(2):9A.

See Also

[pairs](#), [splom](#), [mosaicplot](#), [strucplot](#), [bwplot](#), [barcode](#), [stripplot](#).

Examples

```
allexamples <- FALSE

y <- data.frame(A=c(rep("red", 100), rep("blue", 100)),
               B=c(rnorm(100), round(rnorm(100, 5, 1), 1)), C=runif(200),
               D=c(rep("big", 150), rep("small", 50)),
               E=rnorm(200))

gpairs(y)

data(iris)
gpairs(iris)
if (allexamples) {
  gpairs(iris, upper.pars = list(scatter = 'stats'),
        scatter.pars = list(pch = substr(as.character(iris$Species), 1, 1),
                           col = as.numeric(iris$Species)),
        stat.pars = list(verbose = FALSE))
  gpairs(iris, lower.pars = list(scatter = 'corrgram'),
        upper.pars = list(conditional = 'boxplot', scatter = 'loess'),
        scatter.pars = list(pch = 20))
}

data(Leaves)
gpairs(Leaves[1:10], lower.pars = list(scatter = 'loess'))
```

```

if (allexamples) {
  gpairs(Leaves[1:10], upper.pars = list(scatter = 'stats'),
        lower.pars = list(scatter = 'corrgram'),
        stat.pars = list(verbose = FALSE), gap = 0)
  corrgram(Leaves[, -33])
}

runexample <- FALSE
if (runexample) {
  data(NewHavenResidential)
  gpairs(NewHavenResidential)
}

```

Leaves

Morphological descriptions of leaf floras

Description

Measurements of the percentages of leaves in 31 morphological (or architectural) categories found in 245 leaf floras from 4 studies.

Usage

```
data(Leaves)
```

Format

A data frame with 245 observations on the following 33 variables.

Lobd a numeric vector giving percentage Lobed leaves
Entr a numeric vector giving percentage Entire leaves
TReg a numeric vector giving percentage leaves with Regular Teeth
TCls a numeric vector giving percentage leaves with Close Teeth
TRnd a numeric vector giving percentage leaves with Round Teeth
TAcu a numeric vector giving percentage leaves Acute Teeth
TCmp a numeric vector giving percentage leaves with Compound Teeth
ZNan a numeric vector giving percentage Nanophyll leaves
ZLe1 a numeric vector giving percentage Leptophyll1 leaves
ZLe2 a numeric vector giving percentage Leptophyll2 leaves
ZMi1 a numeric vector giving percentage Microphyll1 leaves
ZMi2 a numeric vector giving percentage Microphyll2 leaves
ZMi3 a numeric vector giving percentage Microphyll3 leaves
ZMe1 a numeric vector giving percentage Megaphyll1 leaves

- ZMe2** a numeric vector giving percentage Megaphyll2 leaves
- ZMe3** a numeric vector giving percentage Megaphyll3 leaves
- AEmg** a numeric vector giving percentage leaves with Emarginate Apexes
- ARnd** a numeric vector giving percentage leaves with Round Apexes
- AAcu** a numeric vector giving percentage leaves with Acute Apexes
- AAtn** a numeric vector giving percentage leaves with Attenuate Apexes
- BCor** a numeric vector giving percentage leaves with Cordate Bases
- BRnd** a numeric vector giving percentage leaves with Round Bases
- BAcu** a numeric vector giving percentage leaves with Acute Bases
- Rlt1** a numeric vector giving percentage leaves with aspect ratio less than 1:1 (i.e. wider than long)
- Rb12** a numeric vector giving percentage leaves with aspect ratio between 1:1 and 1:2
- Rb23** a numeric vector giving percentage leaves with aspect ratio between 1:2 and 1:3
- Rb34** a numeric vector giving percentage leaves with aspect ratio between 1:3 and 1:4
- Rgt4** a numeric vector giving percentage leaves with aspect ratio between greater than 1:4
- SObo** a numeric vector giving percentage Obovate leaves
- SElp** a numeric vector giving percentage Elliptical leaves
- SOvt** a numeric vector giving percentage Ovate leaves
- MAT** a numeric vector giving mean annual temperature in degrees Centigrade
- Study** a factor with levels Wolfe173 Jacobs Gregory Kowalski

Details

Data consists of a data frame with 245 rows and 33 columns (variables). The rows represent floras (collections of plants from a defined locality); the first 31 variables are percentages of leaves in each flora in each of 31 morphological categories; the 32nd variable is mean annual temperature of the area from which the floras was collected in degrees C, and the 32nd is a factor indicating which of 4 published studies the floras come from. See cited publications for more details.

Source

Green, W. A. (2006) Loosening the CLAMP: An exploratory graphical approach to the Climate Leaf Analysis Multivariate Program *Palaeontologia Electronica* 9(2):9A.

<http://www.palaeo-electronica.org/2006_2/clamp/index.html>

<<http://www.open.ac.uk/earth-research/spicer/CLAMP/Physg3ar.xls>> (the climatic variables)

<<http://www.open.ac.uk/earth-research/spicer/CLAMP/MET3AR.xls>> (the morphological leaf scores)

References

- Gregory-Wodzicki, K. M. (2000) Relationships between leaf morphology and climate, Bolivia: implications for estimating paleoclimate from fossil floras. *Paleobiology* 26(4):668–688.
- Jacobs, B. F. (1999) Estimation of rainfall variables from leaf characters in tropical Africa. *Palaeogeography, Palaeoclimatology, Palaeoecology* 145:231–250.
- Jacobs, B. F. (2002) Estimation of low-latitude paleoclimates using fossil angiosperm leaves: examples from the Miocene Tugen Hills, Kenya. *Paleobiology* 28(3):399–421.
- Kowalski, E. A. (2002) Mean annual temperature estimation base on leaf morphology: a test from tropical South America. *Palaeogeography, Palaeoclimatology, Palaeoecology* 188:141–165.
- Wolfe, J.A., (1993), A method of obtaining climatic parameters from leaf assemblages. *U.S. Geological Survey Bulletin* 2040, 73 pp.

Examples

```
data(Leaves)
## maybe str(Leaves) ; plot(Leaves) ...
```

nasa

Pressure and High Cloud Cover Spatially Distributed Time Series

Description

Six years of monthly pressure and high cloud cover measurements over a regular grid of the Americas, from NASA's poster competition at the 2006 Joint Statistical Meeting (JSM).

Usage

```
data(nasa)
```

Format

This NASA data set is stored as a list of 3 components: `data` (containing the pressure and high cloud cover measurements), `elev` (the elevation data), and `coast` (the coastline data). To see the structure, type `str(nasa)`, and see `Details` and `Source` for more information, below.

Details

The data are a subset of some geographic and atmospheric measurements on a coarse 24 by 24 grid covering Central America. The variables included are elevation, air pressure, and high cloud cover. With the exception of elevation, the variables are monthly averages, with observations for Jan., 1995 to Dec., 2000. These data were obtained from the NASA Langley Research Center Atmospheric Sciences Data Center.

Source

NASA Langley Research Center Atmospheric Sciences Data Center, with permission. The JSM poster competition was announced at:

<http://www.amstat-online.org/sections/graphics/dataexpo/2006.php>

Examples

```
# See sparkmat().
```

```
NewHavenResidential
```

```
New Haven, CT Residential Property Data
```

Description

Selected characteristics of a set of small residential properties in New Haven, CT (excluding larger multi-family properties and apartment buildings).

Usage

```
data(NewHavenResidential)
```

Format

A data frame with 18221 observations on the following 8 variables.

totalCurrVal the 2006 assessed value of the property

livingArea the living area in square feet

dep the amount of depreciation, as a percent

size the size of the land, in acres

zone the residential zone, a factor with levels `Other` `RM` `RS`

acType whether the property has central air conditioning: a factor with levels `AC` `No` `AC`

bedrms the number of bedrooms

bathrms the number of bathrooms

Details

The data have been cleaned somewhat, with emphasis on somewhat. For example, there is a property (a very nice one), which has an extremely low assessed value, given its characteristics. It happens to straddle the border between New Haven and Hamden, and so it pays only a proportion of its property taxes to the City of New Haven.

Source

John W. Emerson, from the City of New Haven's property database, which contains more than 27,000 property records (including, for example, the New Haven Airport) and many more variables than included here.

Examples

```
data(NewHavenResidential)
gpairs(NewHavenResidential, scatter.pars=list(pch="."))
```

sparkline	<i>Draws a sparkline</i>
-----------	--------------------------

Description

Draws a times series or 'sparkline' in a compact iconic fashion suitable for inclusion in more complex graphics or text.

Usage

```
sparkline(s, times = NULL, ylim = NULL, buffer = unit(0, "lines"),
          margins = NULL, IQR = NULL, yaxis = FALSE, xaxis = FALSE,
          ptopts = list(points = NULL, labels = NULL, labels.ch = NULL,
                        gp = NULL, just = NULL, pch = NULL), margin.pars = NULL,
          buffer.pars = NULL, frame.pars = NULL, line.pars = gpar(lwd = 1),
          main = NULL, sub = NULL, xlab = NULL, ylab = NULL, new = TRUE)
```

Arguments

<code>s</code>	a vector or time series (class "ts" or "zoo") giving the data to be plotted. If <code>s</code> is a time series, the start, end, and frequency found in <code>attributes(s)\$tsp</code> are automatically converted into an argument to <code>times</code> .
<code>times</code>	the times at which to plot the data; if <code>NULL</code> (the default), equal spacing is assumed, equivalent to setting <code>times = 1:length(s)</code> .
<code>ylim</code>	the maximum and minimum value on the y-axis; if <code>NULL</code> , defaults to the actual maximum and minimum of the data.
<code>buffer</code>	a buffer above the maximum and below the minimum values attained by the sparkline. Defaults to <code>unit(0, 'lines')</code> .
<code>margins</code>	margins around the sparkline-plus-buffer area. <code>NULL</code> (the default) provides no margins; the value passed must be a 4-vector of units giving the bottom, left, top and right margins in that order.
<code>IQR</code>	a list of graphics parameters to shade or otherwise delineate the interquartile range of the sparkline. <code>NULL</code> (the default), does not show the IQR. See <i>Details</i> for more information.
<code>yaxis</code>	draws a vertical axis if <code>TRUE</code> ; defaults to <code>FALSE</code> in which case no axis is drawn.

<code>xaxis</code>	'interior' draws a horizontal axis inside the plotting frame; 'exterior' outside the plotting frame (in the margins); defaults to <code>FALSE</code> , in which case no axis is drawn.
<code>ptopts</code>	a list of graphics parameters describing the points on the sparkline that are plotted and labelled. In particular the first and last or minimum and maximum points are labeled if <code>ptopts\$labels</code> is 'first.last' or 'min.max'. In addition to labels, other relevant parameters from <code>gpar</code> should be valid. See <code>Details</code> for more information.
<code>margin.pars</code>	a list of graphics parameters describing the margin area. See <code>Details</code> for more information.
<code>buffer.pars</code>	a list of graphics parameters describing the buffer area. See <code>Details</code> for more information.
<code>frame.pars</code>	a list of graphics parameters describing the exact area taken up by the plotted sparkline. See <code>Details</code> for more information.
<code>line.pars</code>	a list of graphics parameters describing the sparkline. See <code>Details</code> for more information.
<code>main</code>	a main title, above the sparkline.
<code>sub</code>	a subtitle, to the right of the sparkline.
<code>xlab</code>	a string to label the x-axis.
<code>ylab</code>	a string to label the y-axis.
<code>new</code>	defaults to <code>TRUE</code> , which creates a new, empty page; otherwise adds the sparkline to an existing plot.

Details

In all the cases where a list of graphics parameters is needed, the valid parameter names are the same as would be valid when passed to `gpar` in the appropriate call. That is, passing `list(fill = 'blue', col = 'red')` to `margin` gives a margin that is blue with a red border; but adding `fontface = 'bold'` will have no effect, just as it would have no effect in a call to `grid.rect()`. In particular, note that `ptopts` takes the following non-standard parameters: `labels`, a vector indexing the points to label or the string 'min.max' or 'first.last'; `labels.ch`, a vector of strings giving the labels; and `points`, a vector indexing the points at which points should be plotted. Passing 'min.max' or 'first.last' to `ptopts$labels` overrides any values of `ptopts$labels.ch`.

Note

This is primarily intended to be called by other functions (`sparklines()` and `sparkmat()`), but it can also be used as an alternative to `ts.plot()`. Thanks to Gabor Grothendieck for suggesting the generalization that provides support of "zoo" objects.

Author(s)

John W. Emerson, Walton Green

References

Tufte, E. R. (2006) *It Beautiful Evidence* Cheshire, Connecticut: Graphics Press.

See Also

[ts.plot](#), [sparklines](#), [sparkmat](#)

Examples

```
### sparkline examples
data(nhtemp)

## The default behaviour of sparkline

sparkline(nhtemp)

## Creating stand-alone plots

sparkline(rnorm(10),
          buffer = unit(1, "lines"),
          ptopts = 'first.last',
          margins = unit(c(1,1,1,1), 'inches'),
          yaxis = TRUE, xaxis=TRUE,
          IQR = gpar(fill = 'grey', col = 'grey'),
          main = "Ten Random Standard Normal Numbers",
          sub = '...plotted here')

data(YaleEnergy)
y <- YaleEnergy[YaleEnergy$name==YaleEnergy$name[2],]
sparkline(y$ELSQFT, times=y$year+y$month/12,
          xaxis=TRUE, yaxis=TRUE, main="Branford College Electrical Consumption",
          buffer=unit(1, "lines"), margins = unit(c(1, 1, 1, 1), 'inches'))

sparkline(Nile,
          buffer = unit(1, "lines"),
          ptopts = list(labels = 'min.max'),
          margin.pars = gpar(fill = 'lightblue'),
          buffer.pars = gpar(fill = 'lightgreen'),
          frame.pars = gpar(fill = 'lightyellow'),
          yaxis = TRUE, xaxis=TRUE,
          IQR = gpar(fill = 'grey', col = 'grey'),
          main="Nile Discharge between 1871 and 1970",
          sub='In what units?')

## Adding a sparkline to an existing plot

grid.newpage()
pushViewport(viewport(w = 0.8, h = 0.8))
sparkline(rnorm(10),
          buffer = unit(1, "lines"),
          margins = unit(c(4,4,4,4), 'points'),
```

```

      ptopts = list(labels = 'min.max'),
      margin.pars = gpar(fill = 'lightblue'),
      buffer.pars = gpar(fill = 'lightgreen'),
      frame.pars = gpar(fill = 'lightyellow'),
      yaxis = TRUE, xaxis=TRUE,
      IQR = gpar(fill = 'grey', col = 'grey'),
      main="Title (plotted OUTSIDE the viewport)", new = FALSE)
popViewport ()

```

 sparklines

Draws a panel of vertically stacked sparklines

Description

Draws a panel of vertically stacked, aligned sparklines, or time series.

Usage

```

sparklines(ss, times = NULL, overlap = FALSE, yscale = NULL,
  buffer = unit(0, "lines"), buffer.pars = NULL, IQR = NULL,
  ptopts = NULL, yaxis = TRUE, xaxis = "exterior",
  labeled.points = NULL, point.labels = NULL,
  label.just = c(1.2, 0.5), frame.pars = NULL,
  line.pars = gpar(lwd = 1),
  outer.margin = unit(c(5, 4, 4, 2), "lines"),
  outer.margin.pars = NULL, main = NULL, sub = NULL,
  xlab = NULL, ylab = NULL, lcol = NULL, new = TRUE)

```

Arguments

<code>ss</code>	a data frame whose columns give the time series to be plotted
<code>overlap</code>	FALSE for stacked sparklines; TRUE for all plotted on the same y-axis.
<code>times</code>	the times at which to plot the data; if NULL (the default), equal spacing is assumed. All the sparklines must share the same <code>times</code> argument. If unaligned time series must be plotted, multiple calls to <code>sparklines()</code> are required.
<code>yscale</code>	either a vector of length 2 giving the y-limits for all sparklines, or a list having the same length as the number of columns in <code>ss</code> (each component of which is a 2-vector giving the associated sparkline scales). Defaults to NULL, in which case the scales for each sparkline are set to the sparkline's minimum and maximum values.
<code>buffer</code>	a buffer above the maximum and below the minimum values attained by the sparkline. Defaults to <code>unit(0, 'lines')</code> .
<code>buffer.pars</code>	a list of graphics parameters describing the buffer area. See <code>Details</code> for more information.

IQR	a list of graphics parameters to shade or otherwise delineate the interquartile range of the sparkline. Defaults to <code>NULL</code> , in which case the IQR is not shown. See <code>Details</code> for more information.
ptopts	a list of graphics parameters describing the points on the sparkline that are plotted and labelled. In particular the first and last or minimum and maximum points are labeled if <code>ptopts\$labels</code> is <code>'first.last'</code> or <code>'min.max'</code> .
yaxis	draws a vertical axis if <code>TRUE</code> ; defaults to <code>FALSE</code> , in which case no axis is drawn.
xaxis	<code>'interior'</code> draws horizontal axes inside the plotting frame (for each sparkline); <code>'exterior'</code> draws the common axis for all the sparklines outside the plotting frame; defaults to <code>FALSE</code> (no axis).
labeled.points	not implemented. See <code>ptopts</code> .
point.labels	not implemented. See <code>ptopts</code> .
label.just	not implemented. See <code>ptopts</code> .
frame.pars	a list of graphics parameters describing the exact area taken up by the plotted sparkline. See <code>Details</code> for more information.
line.pars	a list of graphics parameters describing the sparkline. See <code>Details</code> for more information.
outer.margin	a vector of 4 units (bottom, left, top, right) giving the outer margin sizes in order (around the entire panel of sparklines). Defaults to <code>unit(c(0, 0, 0, 0), 'lines')</code> .
outer.margin.pars	a list of graphics parameters describing the outer margin. See <code>Details</code> for more information.
main	a main title, above the stack of sparklines.
sub	a character vector the length of <code>length(ss)</code> providing titles for the individual sparklines, printed to the right of the sparklines.
xlab	a string providing the label for the common x-axis or (probably a useless feature) a character vector the length of <code>length(ss)</code> providing x-axis labels for the individual sparklines.
ylab	a character vector the length of <code>length(ss)</code> providing y-axis labels for the individual sparklines.
lcol	a vector of colors the same length as the number of columns in <code>ss</code> to color the line. As in base graphics, can be either a vector of strings giving the color names, a numeric vector referring to the current palette, or the output of functions like <code>hsv</code> or <code>rgb</code>
new	defaults to <code>TRUE</code> , which creates a new, empty page; otherwise adds the sparkline to the existing plot.

Details

In all the cases where a list of graphics parameters is needed, the valid parameter names are the same as would be valid when passed to `gpar` in the appropriate call. That is, passing `list(fill = 'blue', col = 'red')` to `margin` gives a margin that is blue with a red border; but adding `fontface = 'bold'` will have no effect, just as it would have no effect in a call to `grid.rect`.

Note

We do not support non-aligned time series plots such as `ts.plot(airmiles, Nile, nhtemp)`.

Author(s)

John W. Emerson, Walton Green

References

Tufte, E. R. (2006) *Beautiful Evidence* Cheshire, Connecticut: Graphics Press.

See Also

`ts.plot`, `sparkline`, `sparkmat`

Examples

```
### sparkline examples
data(beaver1)

## The default behaviour of sparklines
sparklines(beaver1)

sparklines(beaver1,
            outer.margin = unit(c(2,4,4,5), 'lines'),
            outer.margin.pars = gpar(fill = 'lightblue'),
            buffer = unit(1, "lines"),
            frame.pars = gpar(fill = 'lightyellow'),
            buffer.pars = gpar(fill = 'lightgreen'),
            yaxis = TRUE, xaxis=FALSE,
            IQR = gpar(fill = 'grey', col = 'grey'),
            main = 'Beaver 1')

data(YaleEnergy)
y <- YaleEnergy[YaleEnergy$name==YaleEnergy$name[2],]
sparklines(y[,c("ELSQFT", "STEAM")], times=y$year+y$month/12,
            main="Branford Electric and Steam Consumption")

## Adding a pair of sparklines to an existing plot

grid.newpage()
pushViewport(viewport(w = 0.8, h = 0.8))
sparklines(data.frame(x = rnorm(10), y = rnorm(10, mean=5)), new = FALSE)
popViewport()

grid.newpage()
pushViewport(viewport(w = 0.8, h = 0.8))
sparklines(data.frame(x = rnorm(10), y = rnorm(10, mean=2)),
            buffer = unit(1, "lines"),
            frame.pars = gpar(fill = 'lightyellow'),
            yaxis = TRUE, xaxis=FALSE,
```

```

IQR = gpar(fill = 'grey', col = 'grey'), new = FALSE)
popViewport()

```

 sparkmat

Draws a sparkmat

Description

Draws multiple time series (or sparklines) at given locations.

Usage

```

sparkmat(x, locs = NULL, w = NULL, h = NULL, lcol = NULL,
         yscalses = NULL, tile.shading = NULL,
         tile.margin = unit(c(0, 0, 0, 0), "points"),
         tile.pars = NULL, just = c("right", "top"),
         new = TRUE, ...)

```

Arguments

<code>x</code>	a list of data frames, all with the same dimensions, one for each panel of vertically aligned sparklines.
<code>locs</code>	a data frame with x-coordinates in the first variable and y-coordinates in the second variable, giving locations of each of the <code>length(x)</code> sparkline panels.
<code>w</code>	vector of unit widths (or native widths if not specified as units).
<code>h</code>	vector of unit heights (or native heights if not specified as units).
<code>lcol</code>	vector of <code>ncol(x[[1]])</code> line colors, one for each sparkline in each panel.
<code>yscales</code>	either a vector of length 2 giving the y-limits for all sparklines, or a list having the same length as the number of columns in <code>ss</code> (each component of which is a 2-vector giving scales for the individual sparklines). Defaults to <code>NULL</code> , in which case the scales for each sparkline are set to its minimum and maximum value within the panel.
<code>tile.shading</code>	vector of background shadings for the panels.
<code>tile.margin</code>	an outer margin around each tile (panel of sparklines). A 4-vector of units giving the bottom, left, top and right margins; defaults to <code>unit(c(0, 0, 0, 0), 'points')</code> .
<code>tile.pars</code>	a list of graphics parameters describing the buffer area. See <code>Details</code> for more information.
<code>just</code>	default is <code>c("right", "top")</code> ; controls the justification of the sparklines relative to the provided location coordinates.
<code>new</code>	defaults to <code>TRUE</code> , which creates a new, empty page; otherwise adds the sparkline to the existing plot.
<code>...</code>	for arguments to be passed through to <code>sparklines()</code> .

Details

In all the cases where a list of graphics parameters is needed, the valid parameter names are the same as would be valid when passed to `gpar` in the appropriate call. That is, passing `list(fill = 'blue', col = 'red')` to `margin` gives a margin that is blue with a red border; but adding `fontface = 'bold'` will have no effect, just as it would have no effect in a call to `grid.rect()`.

Author(s)

John W. Emerson, Walton Green

References

Tufte, E. R. (2006) *Beautiful Evidence* Cheshire, Connecticut: Graphics Press.

See Also

[ts.plot](#), [sparkline](#), [sparklines](#)

Examples

```
# An example with a time series of energy consumption at Yale colleges.
data(YaleEnergy)
y <- YaleEnergy

# Need list of 12 data frames, each with one time series.

z <- list(data.frame(y[y$name==y$name[1], "ELSQFT"]),
          data.frame(y[y$name==y$name[2], "ELSQFT"]),
          data.frame(y[y$name==y$name[3], "ELSQFT"]),
          data.frame(y[y$name==y$name[4], "ELSQFT"]),
          data.frame(y[y$name==y$name[5], "ELSQFT"]),
          data.frame(y[y$name==y$name[6], "ELSQFT"]),
          data.frame(y[y$name==y$name[7], "ELSQFT"]),
          data.frame(y[y$name==y$name[8], "ELSQFT"]),
          data.frame(y[y$name==y$name[9], "ELSQFT"]),
          data.frame(y[y$name==y$name[10], "ELSQFT"]),
          data.frame(y[y$name==y$name[11], "ELSQFT"]),
          data.frame(y[y$name==y$name[12], "ELSQFT"]))

sparkmat(z, locs=data.frame(y$lon, y$lat), new=TRUE,
          w=0.002, h=0.0002, just=c("left", "top"))
grid.text(y[1:12,1], unit(y$lon[1:12]+0.001, "native"),
          unit(y$lat[1:12]+0.00003, "native"),
          just=c("center", "bottom"), gp=gpar(cex=0.7))
grid.text("Degrees Longitude", 0.5, unit(-2.5, "lines"))
grid.text("Degrees Latitude", unit(-4.5, "lines"), 0.5, rot=90)
grid.text("Monthly Electrical Consumption (KwH/SqFt)",
          0.5, 0.82, gp=gpar(cex=1, font=2))
grid.text("of Yale Residential Colleges",
```

```

        0.5, 0.77, gp=gpar(cex=1, font=2))
grid.text("July 1999 - July 2006",
        0.5, 0.72, gp=gpar(cex=1, font=2))

# An example with pressure and high cloud cover over a regular grid of the
# Americas, provided by NASA ().

runexample <- FALSE
if (runexample) {

data(nasa)

grid.newpage()
pushViewport(viewport(w = unit(1, "npc")-unit(2, "inches"),
        h = unit(1, "npc")-unit(2, "inches")))
v <- viewport(xscale = c(-115, -55),
        yscale = c(-22.5, 37.5))
pushViewport(v)

y <- vector(mode="list", length=24*24)
locs <- as.data.frame(matrix(0, 24*24, 2))
tile.shading <- rep(0, 24*24)
for(i in 1:24) { # Latitudes
  for(j in 1:24) { # Longitudes
    y[[i-1]*24+j]] <- as.data.frame(t(nasa$data[,i,j]))
    locs[(i-1)*24+j,] <- c(as.numeric(dimnames(nasa$data)$lon[j]),
        as.numeric(dimnames(nasa$data)$lat[i]))
    tile.shading[(i-1)*24+j] <- gray( 1-.5*(nasa$elev[i,j]/max(nasa$elev)) )
  }
}

yscales <- list(quantile(nasa$data["pressure",,,], c(0.01, 0.99), na.rm=TRUE),
        quantile(nasa$data["cloudhigh",,,], c(0.01, 0.99), na.rm=TRUE))

sparkmat(y, locs=locs, just='center', w=2.5, h=2.5,
        tile.shading=tile.shading, lcol=c(6,3), yscales=yscales,
        tile.margin = unit(c(2,2,2,2), 'points'), new=FALSE)

grid.xaxis(gp=gpar(fontface=2, fontsize=14))
grid.yaxis(gp=gpar(fontface=2, fontsize=14))
grid.rect()

grid.text("Degrees Latitude", x=unit(-0.75, "inches"), y=0.5, rot=90,
        gp=gpar(fontface=2, fontsize=14))
grid.text("Degrees Longitude", x=0.5, y=unit(-0.75, "inches"), rot=0,
        gp=gpar(fontface=2, fontsize=14))
grid.text("Grayscale shading reflects",
        x=unit(1, "npc")+unit(0.6, "inches"), y=0.5, rot=270,
        gp=gpar(fontface=2, fontsize=14))
grid.text("average elevation above sea level",
        x=unit(1, "npc")+unit(0.3, "inches"), y=0.5, rot=270,
        gp=gpar(fontface=2, fontsize=14))

```

```

grid.lines(nasa$coast[,1], nasa$coast[,2], default.units = 'native',
          gp = gpar(col = 'black', lwd = 1))

grid.text("Pressure",
          x=0.25, y=unit(1, "npc")+unit(1.25, "lines"),
          gp=gpar(fontface=2, fontsize=14))
grid.rect(x=0.25, y=unit(1, "npc") + unit(0.5, "lines"),
          width=0.4, height=unit(0.05, "inches"), gp=gpar(col=6, fill=6))
grid.text("High Cloud",
          x=0.75, y=unit(1, "npc")+unit(1.25, "lines"),
          gp=gpar(fontface=2, fontsize=14))
grid.rect(x=0.75, y=unit(1, "npc") + unit(0.5, "lines"),
          width=0.4, height=unit(0.05, "inches"), gp=gpar(col=3, fill=3))
}

```

whatis

Data frame summary

Description

Summarize the characteristics of variables (columns) in a data frame.

Usage

```
whatis(x, var.name.truncate = 20, type.truncate = 14)
```

Arguments

<code>x</code>	a data frame
<code>var.name.truncate</code>	maximum length (in characters) for truncation of variable names. The default is 20; anything less than 12 is less than the column label in the resulting data frame and is a waste of information.
<code>type.truncate</code>	maximum length (in characters) for truncation of variable type; 14 is the full width, but 4 works well if space is at a premium.

Details

The function `whatis()` provides a basic examination of some characteristics of each variable (column) in a data frame.

Value

A list of characteristics describing the variables in the data frame, `x`. Each component of the list has `length(x)` values, one for each variable in the data frame `x`.

<code>variable.name</code>	from the <code>names(x)</code> attribute, possibly truncated to <code>var.name.truncate</code> characters in length.
----------------------------	--

type	the possibilities include "pure factor", "mixed factor", "ordered factor", "character", and "numeric"; <code>whatIs()</code> considers the possibility that a factor or a vector could contain character and/or numeric values. If both character and numeric values are present, and if the variable is a factor, then it is called a mixed factor. If the levels of a factor are purely character or numeric (but not both), it is a pure factor. Non-factors must then be either character or numeric.
missing	the number of NAs in the variable.
<code>distinct.values</code>	the number of distinct values in the variable, equal to <code>length(table(variable))</code> .
precision	the number of decimal places of precision.
min	the minimum value (if numeric) or first value (alphabetically) as appropriate.
max	the maximum value (if numeric) or the last value (alphabetically) as appropriate.

Author(s)

John W. Emerson, Walton Green

References

Special thanks to John Hartigan and the students of 'Statistical Case Studies' of 2004 for their help troubleshooting and developing the function `whatIs()`.

See Also

See also [str](#).

Examples

```
mydf <- data.frame(a=rnorm(100),
                  b=sample(c("Cat", "Dog"), 100, replace=TRUE),
                  c=sample(c("Apple", "Orange", "8"), 100, replace=TRUE),
                  d=sample(c("Blue", "Red"), 100, replace=TRUE))
mydf$d <- as.character(mydf$d)
whatIs(mydf)

data(iris)
whatIs(iris)

data(NewHavenResidential)
whatIs(NewHavenResidential)
```

 YaleEnergy

Monthly energy consumption of Yale residential colleges.

Description

The data set contains monthly energy time series for Yale residential college, from July 1999 through July 2006

Usage

```
data(YaleEnergy)
```

Format

A data frame with 1020 observations on the following 18 variables.

name a factor with levels BERKELEY BRANFORD CALHOUN DAVENPORT EZRA STILES JONATHAN EDWARDS MORSE PIERSON SAYBROOK SILLIMAN TIMOTHY DWIGHT TRUMBULL

address a factor with levels 189 ELM ST. 205 ELM ST. 241 ELM ST. 242 ELM ST. 248 YORK ST. 261 PARK ST. 302 YORK ST. 345 TEMPLE ST. 505 COLLEGE ST. 70 HIGH ST. 74 HIGH ST.

gsf gross square footage of the college

EL electrical consumption in kilowatt hours

ELSQFT electrical consumption per square foot

CHW chilled water consumption in tons

SQFTCHW square feet per ton of chilled water

STEAM steam consumption in pounds

STEAMSQFT steam per square foot

MBTU million British Thermal Units (BTU) from chilled water and steam

MBTUSQFT million BTUs per square foot

year year of the record

month month of the record

lon degrees longitude of the college

lat degrees latitude

Source

John W. Emerson, Yale University

Examples

```

data(YaleEnergy)
whatis(YaleEnergy)

y <- YaleEnergy          # This is just for convenience.
esqft <- list(data.frame(y[y$name==y$name[1], "ELSQFT"]),
              data.frame(y[y$name==y$name[2], "ELSQFT"]),
              data.frame(y[y$name==y$name[3], "ELSQFT"]),
              data.frame(y[y$name==y$name[4], "ELSQFT"]),
              data.frame(y[y$name==y$name[5], "ELSQFT"]),
              data.frame(y[y$name==y$name[6], "ELSQFT"]),
              data.frame(y[y$name==y$name[7], "ELSQFT"]),
              data.frame(y[y$name==y$name[8], "ELSQFT"]),
              data.frame(y[y$name==y$name[9], "ELSQFT"]),
              data.frame(y[y$name==y$name[10], "ELSQFT"]),
              data.frame(y[y$name==y$name[11], "ELSQFT"]),
              data.frame(y[y$name==y$name[12], "ELSQFT"]))

# The sparkmat() command does most of the work:
sparkmat(esqft, locs=data.frame(y$lon, y$lat), new=TRUE,
         w=0.002, h=0.0002, just=c("left", "top"))

# We'll add some text for a nice finished product:
grid.text(y[1:12,1], unit(y$lon[1:12]+0.001, "native"),
         unit(y$lat[1:12]+0.00003, "native"),
         just=c("center", "bottom"), gp=gpar(cex=0.7))
grid.text("Degrees Longitude", 0.5, unit(-2.5, "lines"))
grid.text("Degrees Latitude", unit(-4.5, "lines"), 0.5, rot=90)
grid.text("Monthly Electrical Consumption (KwH/SqFt) of Yale Colleges",
         0.5, 0.8, gp=gpar(cex=1, font=2))
grid.text("July 1999 - July 2006",
         0.5, 0.74, gp=gpar(cex=1, font=2))

```

Index

*Topic **datasets**

Leaves, 9
nasa, 11
NewHavenResidential, 12
whatis, 22
YaleEnergy, 24

*Topic **hplot**

YaleToolkit-package, 2

*Topic **multivariate**

barcode, 3
gpairs, 5
YaleToolkit-package, 2

*Topic **package**

YaleToolkit-package, 2

*Topic **ts**

gpairs, 5
sparkline, 13
sparklines, 16
sparkmat, 19
YaleToolkit-package, 2

barcode, 3, 3, 8

bwplot, 8

corrgram, 3

corrgram(*gpairs*), 5

dist, 7

gpairs, 3, 5, 5

gpar, 7, 13, 14, 17, 20

grid.rect, 17

grid.text, 7

hclust, 6, 7

hsv, 17

Leaves, 9

mosaicplot, 8

nasa, 11

NewHavenResidential, 12

pairs, 8

rgb, 17

rug, 5

sparkline, 3, 13, 18, 20

sparklines, 3, 14, 16, 20

sparkmat, 3, 14, 18, 19

splom, 8

str, 23

stripplot, 5, 8

strucplot, 8

ts.plot, 14, 18, 20

unit, 2

viewport, 2

whatis, 3, 22

YaleEnergy, 24

YaleToolkit

(*YaleToolkit-package*), 2

YaleToolkit-package, 2