Package 'YaleToolkit'

May 7, 2026

Version 4.2.3

Date 2022-05-09

Title Data Exploration Tools from Yale University

Author John W. Emerson and Walton A. Green

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

Depends grid, utils

Imports foreach, iterators

Description This collection of data exploration tools was developed at Yale University for the graphical exploration of complex multivariate data; barcode and gpairs now have their own packages. The big.read.table() function provided here may be useful for large files when only a subset is needed (but please see the note in the help page for this function).

Encoding UTF-8

RoxygenNote 7.1.2

NeedsCompilation no

Repository CRAN

Date/Publication 2022-05-09 17:20:06 UTC

Contents

`big.read.table` *Read in chunks from a large file with row/column filtering to obtain a reasonable-sized data.frame.*

Description

Read in chunks from a large file with row/column filtering to obtain a reasonable-sized data.frame.

Usage

```
big.read.table(
  file,
  nrows = 1e+05,
  sep = ",",
  header = TRUE,
  row.names = NULL,
  cols = NULL,
  rowfilter = NULL,
  as.is = TRUE,
  estimate = FALSE
)
```

Arguments

`file`	the name of the file, obviously
`nrows`	the chunk size; consider reducing this if there are lots of columns
`sep`	by default we expect a CSV file
`header`	is `TRUE` by default
`row.names`	I really dislike row names
`cols`	for filtering column by name or number (supporting negative indexing)
`rowfilter`	a function that is assumed to take a chunk as a data frame and return a smaller data frame (with fewer rows), separately from the column filtering.
`as.is`	`TRUE` by default
`estimate`	do a preliminary estimation of the work to be done, and then have a chance to bail out if it looks like a bad idea

Note

This is very much 'in development' and could be buggy. I put it here as I used some example in one of my courses, but then I needed to update the package to keep CRAN happy. So here it is. Buyer Beware. - Jay

Examples

```
data(CO2)
write.csv(CO2, "CO2.csv", row.names=FALSE)
x <- big.read.table("CO2.csv", nrows=10)
unlink("CO2.csv")
head(x)
```

---

getnrows *Get the number of rows of the file*

Description

Use iterators to avoid the memory overhead of obtaining the number of rows of a file.

Usage

```
getnrows(file, n = 10000)
```

Arguments

file	the name of a file (possible with a path)
n	the size of the chunks used by the iterator

Value

an integer

Examples

```
data(CO2)
write.csv(CO2, "CO2.csv", row.names=FALSE)
getnrows("CO2.csv")
unlink("CO2.csv")
```

---

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().
```

`sparkline` *Draws a sparkline*

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

`s`	a vector or time series (class `"ts"` or `"zoo"`) giving the data to be plotted. If `s` is a time series, the `start`, `end`, and `frequency` found in `attributes(s)$tsp` are automatically converted into an argument to `times`.
`times`	the times at which to plot the data; if `NULL` (the default), equal spacing is assumed, equivalent to setting `times = 1:length(s)`.

ylim	the maximum and minimum value on the y-axis; if NULL, defaults to the actual maximum and minimum of the data.
buffer	a buffer above the maximum and below the minimum values attained by the sparkline. Defaults to unit(0, 'lines').
margins	margins around the sparkline-plus-buffer area. NULL (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.
IQR	a list of graphics parameters to shade or otherwise delineate the interquartile range of the sparkline. NULL (the default), does not show the IQR. See Details for more information.
yaxis	draws a vertical axis if TRUE; defaults to FALSE in which case no axis is drawn.
xaxis	'interior' draws a horizontal axis inside the plotting frame; 'exterior' outside the plotting frame (in the margins); defaults to FALSE, in which case no axis is drawn.
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 ptopts$labels is 'first.last' or 'min.max'. In addition to labels, other relevant parameters from gpar should be valid. See Details for more information.
margin.pars	a list of graphics parameters describing the margin area. See Details for more information.
buffer.pars	a list of graphics parameters describing the buffer area. See Details for more information.
frame.pars	a list of graphics parameters describing the exact area taken up by the plotted sparkline. See Details for more information.
line.pars	a list of graphics parameters describing the sparkline. See Details for more information.
main	a main title, above the sparkline.
sub	a subtitle, to the right of the sparkline.
xlab	a string to label the x-axis.
ylab	a string to label the y-axis.
new	defaults to TRUE, 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

`ss`	a data frame whose columns give the time series to be plotted
`overlap`	`FALSE` for stacked sparklines; `TRUE` for all plotted on the same y-axis.
`times`	the times at which to plot the data; if `NULL` (the default), equal spacing is assumed. All the sparklines must share the same `times` argument. If unaligned time series must be plotted, multiple calls to `sparklines()` are required.

yscale	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 ss (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.
buffer	a buffer above the maximum and below the minimum values attained by the sparkline. Defaults to unit(0, 'lines').
buffer.pars	a list of graphics parameters describing the buffer area. See Details for more information.
IQR	a list of graphics parameters to shade or otherwise delineate the interquartile range of the sparkline. Defaults to NULL, in which case the IQR is not shown. See Details 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 ptopts\$labels is 'first.last' or 'min.max'.
yaxis	draws a vertical axis if TRUE; defaults to FALSE, in which case no axis is drawn.
xaxis	'interior' draws horizontal axes inside the plotting frame (for each sparkline); 'exterior' draws the common axis for all the sparklines outside the plotting frame; defaults to FALSE (no axis).
labeled.points	not implemented. See ptopts.
point.labels	not implemented. See ptopts.
label.just	not implemented. See ptopts.
frame.pars	a list of graphics parameters describing the exact area taken up by the plotted sparkline. See Details for more information.
line.pars	a list of graphics parameters describing the sparkline. See Details 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 unit(c(0,0,0,0), 'lines').
outer.margin.pars	a list of graphics parameters describing the outer margin. See Details for more information.
main	a main title, above the stack of sparklines.
sub	a character vector the length of length(ss) 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 length(ss) providing x-axis labels for the individual sparklines.
ylab	a character vector the length of length(ss) providing y-axis labels for the individual sparklines.
lcol	a vector of colors the same length as the number of columns in ss 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 pallette, or the output of functions like hsv or rgb

new — defaults to TRUE, 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,
         yscales = NULL, tile.shading = NULL,
         tile.margin = unit(c(0, 0, 0, 0), "points"),
         tile.pars = NULL, just = c("right", "top"),
         new = TRUE, ...)
```

Arguments

x	a list of data frames, all with the same dimensions, one for each panel of vertically aligned sparklines.
locs	a data frame with x-coordinates in the first variable and y-coordinates in the second variable, giving locations of each of the `length(x)` sparkline panels.
w	vector of unit widths (or native widths if not specified as units).
h	vector of unit heights (or native heights if not specified as units).
lcol	vector of `ncol(x[[1]])` line colors, one for each sparkline in each panel.
yscales	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 `ss` (each component of which is a 2-vector giving scales for the individual sparklines). Defaults to `NULL`, in which case the scales for each sparkline are set to its minimum and maximum value within the panel.
tile.shading	vector of background shadings for the panels.
tile.margin	an outer margin around each tile (panel of sparklines). A 4-vector of units giving the bottom, left, top and right margins; defaults to `unit(c(0,0,0,0), 'points')`.

tile.pars	a list of graphics parameters describing the buffer area. See Details for more information.
just	default is c("right", "top"); controls the justification of the sparklines relative to the provided location coordinates.
new	defaults to TRUE, which creates a new, empty page; otherwise adds the sparkline to the existing plot.
...	for arguments to be passed through to sparklines().

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

`x`	a data frame
`var.name.truncate`	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.
`type.truncate`	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`.

variable.name from the `names(x)` attribute, possibly truncated to `var.name.truncate` characters in length.

type the possibilities include `"pure factor"`, `"mixed factor"`, `"ordered factor"`, `"character"`, and `"numeric"`; `whatis()` 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 `NA`s in the variable.

distinct.values the number of distinct values in the variable, equal to `length(table(variable))`.

precision the number of decimal places of precision.

min the minumum 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)
```

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))
```

`YaleToolkit` *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()`, although `barcode()` and `gpairs()` are now provided by packages of the same names, respectively.

Details

The package also includes several data sets. For more information, please see the help files for `nasa` and `YaleEnergy`. Please get in touch with us if you note any problems.

Author(s)

John W. Emerson, Walton Green

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 https://www.edwardtufte.com for this and other references.

Index