STAT:5400, 22S:166
Computing in Statistics

More on R

Lecture 7
Sept 14, 2016

Kate Cowles
374 SH, 335-0727
kate-cowles@uiowa.edu

Factors in R

- vector object used to specify a discrete classification (grouping) of the components of other vectors of the same length

- default way of storing character data in data frames

- used in formulas in R

- used in `tapply` function

Example

```
> help(state,package="datasets")
state                   package:datasets              R Documentation
US State Facts and Figures

Description:

     Data sets related to the 50 states of the United States of America.

Usage:

     state.abb
     state.area
     state.center
     state.division
     state.name
     state.region
     state.x77

Details:

     R currently contains the following "state" data sets.  Note that
     all data are arranged according to alphabetical order of the state
     names.

     'state.abb': character vector of 2-letter abbreviations for the
          state names.

     'state.area': numeric vector of state areas (in square miles).

     'state.center': list with components named 'x' and 'y' giving the
          approximate geographic center of each state in negative
          longitude and latitude.  Alaska and Hawaii are placed just
```

```
          off the West Coast.

     'state.division': factor giving state divisions (New England,
          Middle Atlantic, South Atlantic, East South Central, West
          South Central, East North Central, West North Central,
          Mountain, and Pacific).

     'state.name': character vector giving the full state names.

     'state.region': factor giving the region (Northeast, South, North
          Central, West) that each state belongs to.

     'state.x77': matrix with 50 rows and 8 columns giving the
          following statistics in the respective columns.

          'Population': population estimate as of July 1, 1975

          'Income': per capita income (1974)

          .
          .
          .

          'Area': land area in square miles

Source:
     U.S. Department of Commerce, Bureau of the Census (1977)
     _Statistical Abstract of the United States_.
     U.S. Department of Commerce, Bureau of the Census (1977) _County
     and City Data Book_.

References:
     Becker, R. A., Chambers, J. M. and Wilks, A. R. (1988) _The New S
     Language_. Wadsworth & Brooks/Cole.
```

```
> data(state)

> statedf <- data.frame( abb = state.abb, div = state.division,
+ reg = state.region, state.x77[,c("Population","Area")] )

> statedf[1:15,]
            abb              div         reg Population   Area
Alabama      AL East South Central       South      3615  50708
Alaska       AK          Pacific          West       365 566432
Arizona      AZ         Mountain          West      2212 113417
Arkansas     AR West South Central       South      2110  51945
California   CA          Pacific          West     21198 156361
Colorado     CO         Mountain          West      2541 103766
Connecticut  CT      New England     Northeast      3100   4862
Delaware     DE    South Atlantic        South       579   1982
Florida      FL    South Atlantic        South      8277  54090
Georgia      GA    South Atlantic        South      4931  58073
Hawaii       HI          Pacific          West       868   6425
Idaho        ID         Mountain          West       813  82677
Illinois     IL East North Central North Central   11197  55748
Indiana      IN East North Central North Central    5313  36097
Iowa         IA West North Central North Central    2861  55941
```

Functions operating on factors

```
> is.factor(statedf[,"div"])
[1] TRUE

> levels(statedf[,"div"])
[1] "New England"        "Middle Atlantic"    "South Atlantic"
[4] "East South Central" "West South Central" "East North Central"
[7] "West North Central" "Mountain"           "Pacific"
```

Using factors in formulas for plotting and model fitting

```
> boxplot( Population ~ div, data = statedf )
> boxplot( Population ~ div, data = statedf, pars=list(cex.axis=0.75))
> dev.copy2eps( file="~/166/lects2005/boxplotstatepop.ps", horizontal=T)
```
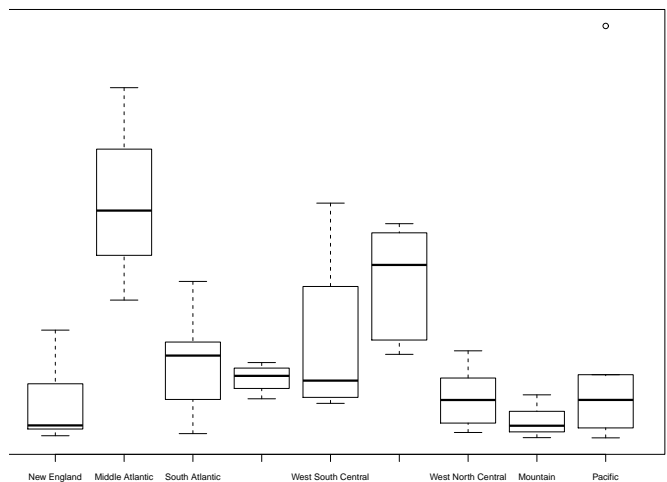
```
> summary(lm(Population ~ div, data = statedf ))

Call:
lm(formula = Population ~ div, data = statedf)

Residuals:
    Min     1Q  Median     3Q     Max
-5289.8 -1667.4  -423.6   987.2 15543.2

Coefficients:
                      Estimate Std. Error t value Pr(>|t|)
(Intercept)             2031.2     1500.3   1.354 0.183207
divMiddle Atlantic     10391.8     2598.6   3.999 0.000259 ***
divSouth Atlantic       2087.1     1984.7   1.052 0.299154
divEast South Central   1347.8     2372.2   0.568 0.573013
divWest South Central   3185.8     2372.2   1.343 0.186664
divEast North Central   6157.8     2225.3   2.767 0.008446 **
divWest North Central    353.3     2044.6   0.173 0.863675
divMountain             -828.0     1984.7  -0.417 0.678704
divPacific              3623.6     2225.3   1.628 0.111109
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

Residual standard error: 3675 on 41 degrees of freedom
Multiple R-squared: 0.433,       Adjusted R-squared: 0.3224
F-statistic: 3.914 on 8 and 41 DF,  p-value: 0.001645
```

```
> statedf[ statedf["div"] == "Middle Atlantic" ,]
             abb            div       reg Population  Area
New Jersey    NJ Middle Atlantic Northeast       7333  7521
New York      NY Middle Atlantic Northeast      18076 47831
Pennsylvania  PA Middle Atlantic Northeast      11860 44966

> statedf[ statedf["div"] == "East North Central" ,]
          abb                div           reg Population  Area
Illinois   IL East North Central North Central      11197 55748
Indiana    IN East North Central North Central       5313 36097
Michigan   MI East North Central North Central       9111 56817
Ohio       OH East North Central North Central      10735 40975
Wisconsin  WI East North Central North Central       4589 54464

> tapply( statedf[,"Population"], statedf[,"div"], mean )
        New England     Middle Atlantic     South Atlantic East South Central
           2031.167           12423.000           4118.250           3379.000
West South Central East North Central West North Central           Mountain
           5217.000            8189.000           2384.429           1203.125
            Pacific
           5654.800
```

Graphics in R

Plotting functions in base R:

- High-level plotting functions create a new plot on the graphics device, possibly with axes, labels, titles and so on.

- Low-level plotting functions add more information to an existing plot, such as extra points, lines and labels.

- Interactive graphics functions allow you interactively add information to, or extract information from, an existing plot, using a pointing device such as a mouse.

Example of high-level function: Plot

`plot` is a generic plotting function whose behavior is determined by the class of the object(s) to which it is applied.

- argument is factor: bar graph of counts of each level

  ```
  > plot( statedf[,"div"], cex.axis=0.75,
  + main = "Number of States per Division" )
  ```

- arguments are two numeric vectors: scatterplot with first vector on x-axis

  ```
  > plot( statedf[,"Area"], statedf[, "Population"],
  + xlab = "Area in Square Miles", ylab = "Population in thousands")
  ```

- argument is a data frame: scatterplot matrix

  ```
  > plot( statedf )
  ```

- plotting one object against each object in an expression

  – object to left of "~" will be on y-axis

  ```
  > par(mfrow=c(1,2) )
  > plot( Population ~ Area + reg, data = statedf )
  ```

Arguments available in default scatter plot function

```
> help(plot.default)
plot.default              package:graphics              R Documentation

The Default Scatterplot Function

Description:

    Draw a scatter plot with decorations such as axes and titles in
    the active graphics window.

Usage:

    ## Default S3 method:
    plot(x, y = NULL, type = "p",  xlim = NULL, ylim = NULL,
         log = "", main = NULL, sub = NULL, xlab = NULL, ylab = NULL,
         ann = par("ann"), axes = TRUE, frame.plot = axes,
         panel.first = NULL, panel.last = NULL, asp = NA, ...)
.
.
.
    log: a character string which contains "x" if the x axis is to
         be logarithmic, "y" if the y axis is to be logarithmic and
         "xy" or "yx" if both axes are to be logarithmic.
```
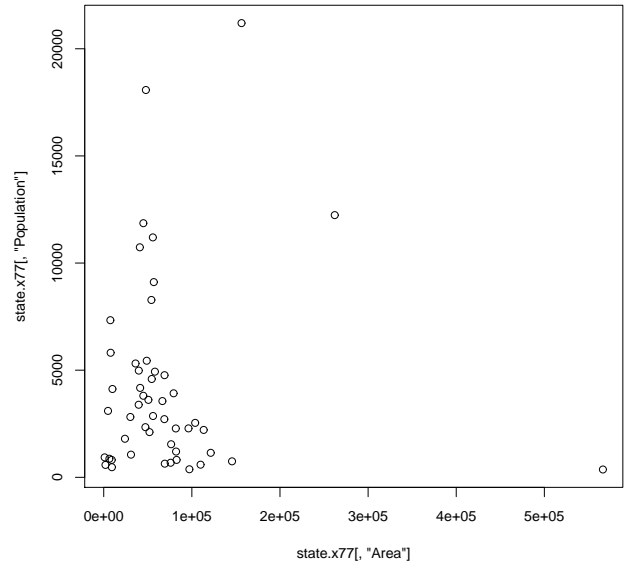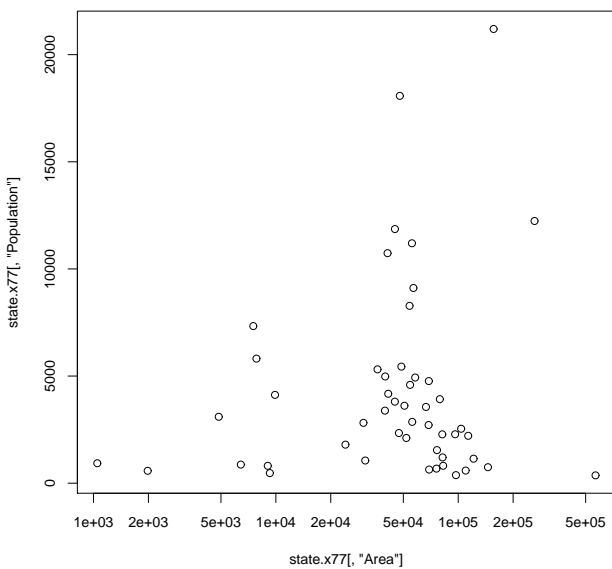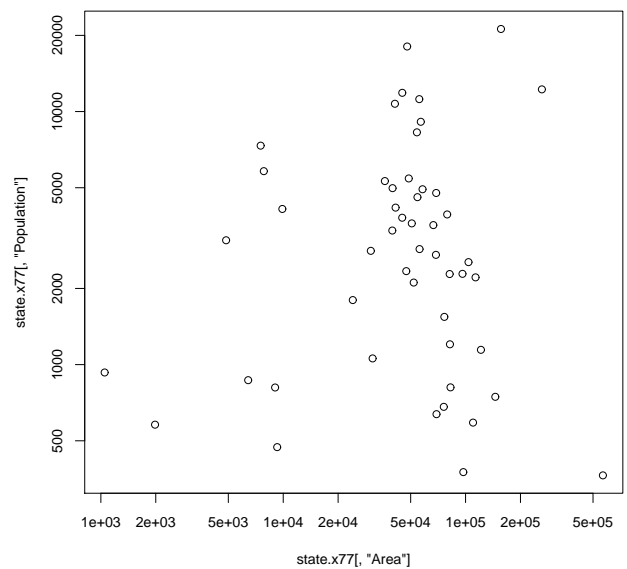
Example

```
> plot( state.x77[,"Area"], state.x77[,"Population"])
> dev.copy2eps(file="~/166/lects2005/rawareapop.ps",horizontal=T)
```

```
> plot( state.x77[,"Area"], state.x77[,"Population"],log="x")
> dev.copy2eps(file="~/166/lects2005/logxareapop.ps",horizontal=T)
```

```
> plot( state.x77[,"Area"], state.x77[,"Population"],log="xy")
> dev.copy2eps(file="~/166/lects2005/logxyareapop.ps",horizontal=T)
```

Low-level plotting functions

- add extra information (such as points, lines or text) to the current plot.

- `points(x,y)`

- `lines(x,y)`

- `text(x,y,labels,...`

  ```
  > attach(statedf)
  > plot( Area, Population, type="n" )
  > text( Area, Population, abb)
  ```

- `legend(x, y, legend, ...)`

```
> statedf <- statedf[ order( statedf[,"Area"] ) , ]
> detach(statedf)
> statedf <- statedf[ order( statedf[,"Area"] ) , ]
> attach(statedf)
> Area
 [1]    1049    1982    4862    6425    7521    7826    9027    9267    9891   24070
[11]   30225   30920   36097   39650   39780   40975   41328   44930   44966   47296
[21]   47831   48798   50708   51945   54090   54464   55748   55941   56817   58073
[31]   66570   68782   68995   69273   75955   76483   79289   81787   82096   82677
[41]   96184   97203  103766  109889  113417  121412  145587  156361  262134  566432
> plot( Area, Population )
> lines( lowess( Population ~ Area) )
> lines( lowess( Population ~ Area, f= 0.25), lty = 2)
> legend( 400000, 15000, legend = c("f=2/3","f = 1/14"), lty=1:2)
```

Interactive graphic functions

- `locator( n, type)`
  - Waits for the user to select n locations on the current plot using the left mouse button.
  - returns the locations of the points selected as a list with two components x and y.

  ```
  > text( locator(2), "Outlier")
  ```

The `apply` family of functions in R

- very much faster than `for` loops that would accomplish the same purpose

- `apply` lets you apply the same function to each row or column of a matrix, and returns results in a vector

- `lapply` applies same function to each member of a list

- `tapply`

- `mapply`