What Data Management Does?

1) Creating New Variables
2) Sorting Data
3) Merging Data
4) Aggregating Data
5) Reshaping Data
6) Subsetting Data
7) Data Type Conversion

(1-1) Adding new variables

Function: <-

```
    input$x1<-input$ELEV
    input$x2<-(input$ELEV+100)/2
# Three examples for doing the same computations
# 1
    input$sum <- input$x1 + input$x2
    input$mean <- (input$x1 + input$x2)/2
# 2
    attach(input)
       input$sum <- x1 + x2
       input$mean <- (x1 + x2)/2
    detach(input)
# 3
    input <- transform( input, sum = x1 + x2,mean = (x1 + x2)/2 )
```

x1	x2	sum	mean
4	3	7	3.5
6	5	11	5.5
7	8	15	7.5

(1-2) Recoding variables

Functions: ifelse,……;

```
# create 2 elevation categories
```

input$elevcat <- ifelse(input$ELEV > 500,~~c(~~"low"~~),c(~~"high"~~)~~)

```
# another example: create 3 elevation categories
   attach(input)
         input$elevcat[ELEV > 600] <- "high"
         input$elevcat[ELEV> 400 & ELEV <= 600] <- "medium"
         input$elevcat[ELEV<= 400] <- "low"
   detach(input)
# restore the input file (for further use):
   input<-input[,1:5]
```

(1-3) Renaming variables

Functions: none.

You can rename variables interactively or programmatically.

```
# interactively
   fix(input) # results are saved on close

# programmatically
   library(reshape)                              # you have to install it
   input <- rename(input, c(ELEV="elev"))   # do we miss anything?

# rename all the variable names
   input<-input[,1:5]  # restore the data
   names(input) <- c("ID1", "Province","LAT", "LONG","ELEV ")
```

(2) Sorting Data

Function: order( )

Default: ASCENDING

Minus sign: DESCENDING

Examples:

```
# using built-in "mtcars" dataset, sort by mpg
    dat <- mtcars[order(mpg),]
# sort by mpg and cyl
    dat <- mtcars[order(mpg, cyl),]
# sort by mpg (ascending) and cyl (descending)
    dat <- mtcars[order(mpg, -cyl),]
```

(3) Merging Data

Adding Columns

Function: merge(), cbind();

```
# merge two dataframes by ID….. Note: next line is not executable
    total <- merge(dataframeA,dataframeB,by="ID")
# merge two dataframes by ID and Country
    total <-merge(dataframeA,dataframeB,by=c("ID","Country"))
```

```
# cbind()
a<-1:10; b<-10:1; d<-cbind(a,b); d;
```

Adding Rows

Function: rbind()

```
total <- rbind(dataframeA, dataframeB)
```

(4) Aggregating Data

Function: aggregate()

Examples:

```
# aggregate dataframe "mtcars" by cyl and vs
# returning means for numeric variables
    attach(mtcars)
           aggdata <-aggregate(mtcars,
           by=list(cyl,vs),FUN=mean,na.rm=TRUE)
    print(aggdata)
    detach(mtcars)
```

Notes: When using the **aggregate()** function, the by variables must be in a list (even if there is only one). The function can be built-in or user provided.

(5) Reshaping Data

Transpose

function: t()

```
# example using built-in dataset
   t(mtcars)
```

Reshape Package

functions: melt(), cast()

```
# example of melt function
   library(reshape)
     input1<-read.csv("input1.csv")
     mdata <- melt(input1, id= c("id","time"))
    print(mdata)


# cast the melted data
# cast(data, formula, function)
   subjmeans <- cast(mdata, id~variable, mean)
   timemeans <- cast(mdata, time~varable, mean)
```

(6) Subsetting Data: Selecting (keeping) variables

```
# select variables ID1 and ELEV
    myvars <- c("ID1", "ELEV")
    newdata<- input[myvars]

# another method
newdata <- input[,c("ID1", "ELEV")]

# select 1st and 3th thru 5th variables
    newdata <- input[c(1,3:5)]
  # of course it is equivalent to:
    newdata <- input[2]
```

(6) Subsetting Data: Excluding (DROPPING) Variables

```
# exclude variables ID1 and Province:
    myvars <- names(input) %in% c("ID1", "Province")
    newdata <- input[!myvars]


# exclude 3rd and 5th variable :
    newdata <- mydata[c(-3,-5)]


# delete variables v3 and v5
    input$LAT <- input$LONG <- NULL
```

(6) Subsetting Data: Selecting Observations

```
# first 5 observerations
    newdata <- input[1:5,]


# based on variable values
    newdata <- input[ which(input$ELEV<500
    & input$Province=="BC"), ]


# or
    attach(input)
    newdata <- input[ which(ELEV<500 &
    Province=="BC"),]
    detach(newdata)
```

(6) Subsetting Data: Selection using the Subset Function

```
# using subset function
    newdata <- subset(input, ELEV<300 | ELEV >
    1000, select=c(ID1, ELEV))


# more options
    newdata <- subset(input, Province=="BC" &
    ELEV > 500, select=LAT:ELEV)
```

(6) Subsetting Data: Random Samples

```
# take a random sample of size 20 from a datase
# mydata sample without replacement
    mysample <- input[sample(1:nrow(mydata),
    20,replace=FALSE),]
```

# Going Further

- **R** has extensive facilities for sampling, including drawing and calibrating survey samples (see the sample package), analyzing complex survey data (see the survey package and it's homepage) and bootstrapping.

(7) Data Type Conversion

Functions: **is.***foo;* **as.***foo;*

is.numeric(), is.character(), is.vector(), is.matrix(), is.data.frame()
as.numeric(), as.character(), as.vector(), as.matrix(), as.data.frame)

Example:

```
a<-input$ELEV
is.numeric(a)
b<-as.character(a)
is.numeric(b)
```

Basic Graphic in R

1) Create a graph
2) Density plots
3) Dot plots
4) Bar plots
5) Line charts
6) Pie charts
7) Boxplots
8) Scatter plots

Creating a Graph

in **R**, graphs are created interactively:

```
# Creating a Graph
    attach(mtcars)
    plot(wt, mpg)
    abline(lm(mpg~wt))
    title("Regression of MPG on
    Weight")
# type help(plot) to learn more;
```

Saving graphs

- From the menu
  File -> Save As.
- via functions:
  - **pdf("mygraph.pdf")** #pdf file
  - **win.metafile("mygraph.wmf")** #windows metafile
  - **png("mygraph.png")** #png file
  - **jpeg("mygraph.jpg")** #jpeg file
  - **bmp("mygraph.bmp")** #bmp file
  - **postscript("mygraph.ps")** #postscript file

```
#Example:
    jpeg("c:/mygraphs/myplot.jpg")
        plot(x)    # this can be a block of codes
    dev.off()
```

Viewing several graphs

- Creating a new graph by issuing a high level plotting command (plot, hist, boxplot, etc.) will typically **overwrite** a previous graph.
- **Open a new graph windows:**

```
windows()
X11()
```

Graphical parameters (1)

# Set a graphical parameter using par()

```
par()                # view current settings
opar <- par()        # make a copy of current settings
par(col.lab="red")   # red x and y labels
hist(mtcars$mpg)     # create a plot with these new settings
par(opar)            # restore original settings
```

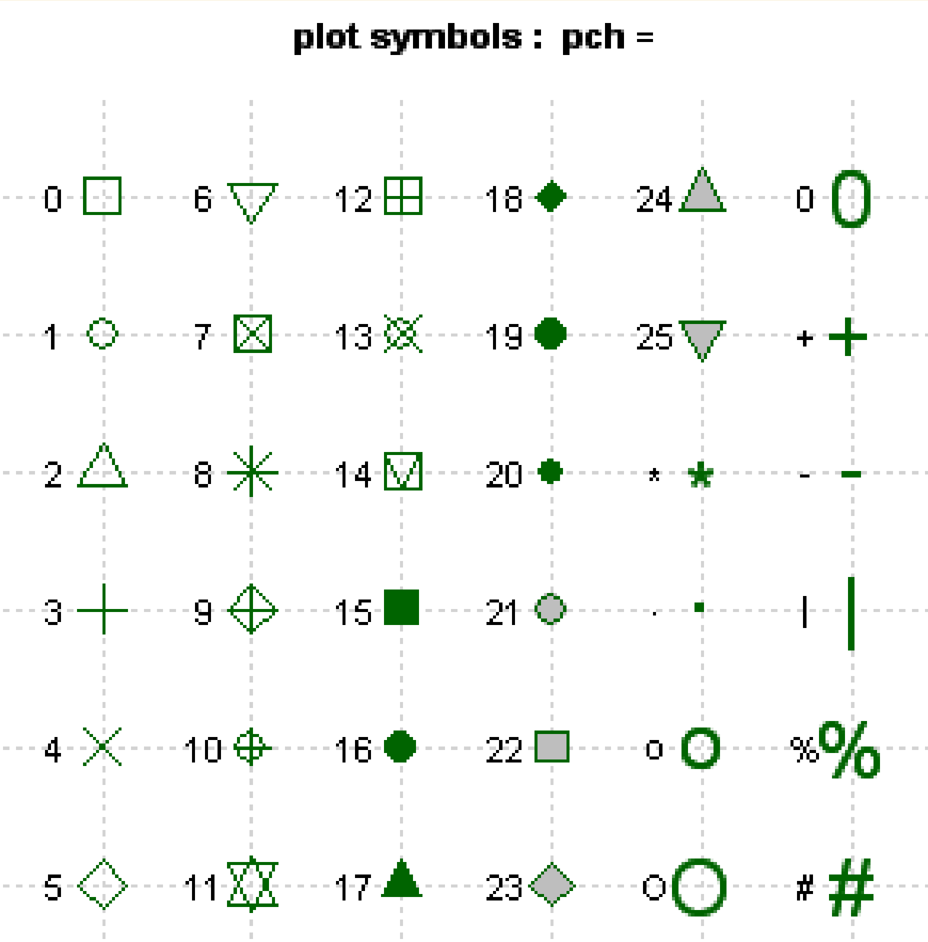
# Set a graphical parameter within the plotting function

```
hist(mtcars$mpg, col.lab="red")
```

# Plotting Symbols

Use the pch= option to specify symbols to use when plotting points. For symbols 21 through 25, specify border color (col=) and fill color (bg=).

Symbols:

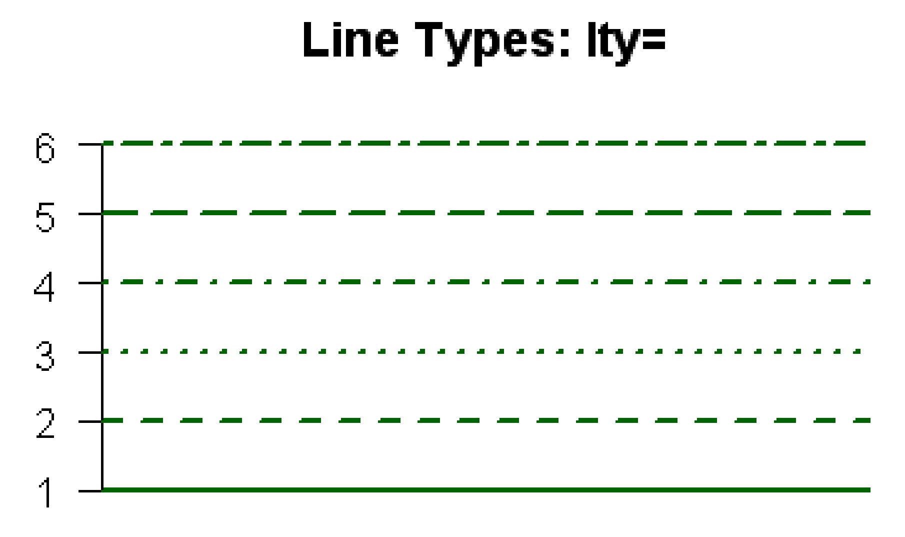

plot symbols : pch =
0
6
12
18
24
0
1
7
13
19
25
+
2
8
14
20
*
-
3
9
15
21
.
|
4
10
16
22
o
%
5
11
17
23
O
#

Graphical parameters (2): text and symbol size

option	description
cex	number indicating the amount by which plotting text and symbols should be scaled relative to the default. 1=default, 1.5 is 50% larger, 0.5 is 50% smaller, etc.
cex.axis	magnification of axis annotation relative to cex
cex.lab	magnification of x and y labels relative to cex
cex.main	magnification of titles relative to cex
cex.sub	magnification of subtitles relative to cex

Graphical parameters (3): lines

Option	Description
lty	Line types, see the graph below.
lwd	Line width relative to the default (default=1). 2 is twice as wide.



Graphical parameters (4): colors

option	description
col	Default plotting color. Some functions (e.g. lines) accept a vector of values that are recycled.
col.axis	color for axis annotation
col.lab	color for x and y labels
col.main	color for titles
col.sub	color for subtitles
fg	plot foreground color (axes, boxes - also sets col= to same)
bg	plot background color

Graphical parameters (5): fonts

option	description
font	Integer specifying font to use for text. 1=plain, 2=bold, 3=italic, 4=bold italic, 5=symbol
font.axis	font for axis annotation
font.lab	font for x and y labels
font.main	font for titles
font.sub	font for subtitles
ps	font point size (roughly 1/72 inch) text size=ps*cex
family	font family for drawing text. Standard values are "serif", "sans", "mono", "symbol". Mapping is device dependent.

Graphical parameters (6): margins and graph sizes

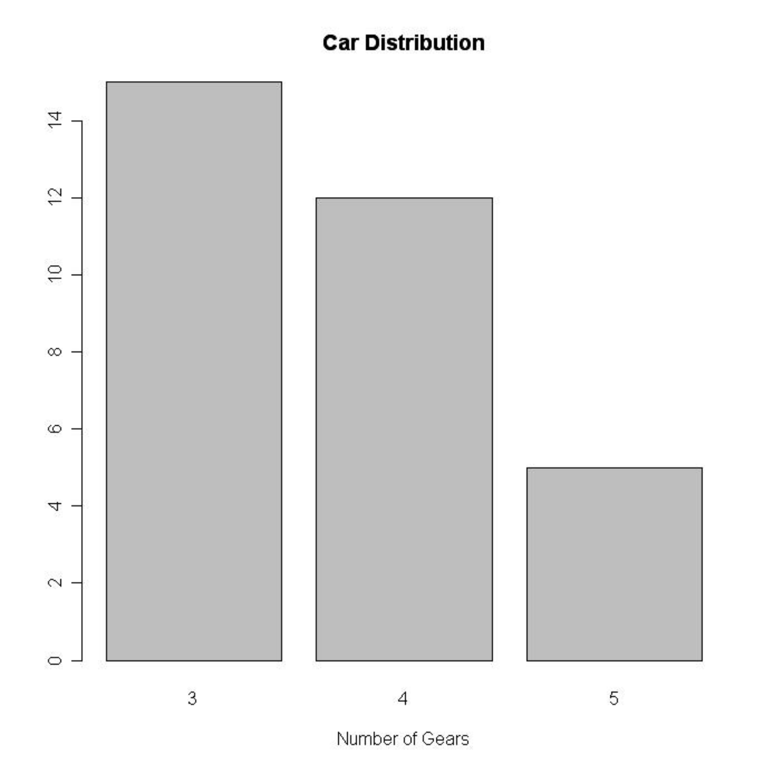
option	description
mar	numerical vector indicating margin size c(bottom, left, top, right) in lines. default = c(5, 4, 4, 2) + 0.1
mai	numerical vector indicating margin size c(bottom, left, top, right) in inches
pin	plot dimensions (width, height) in inches

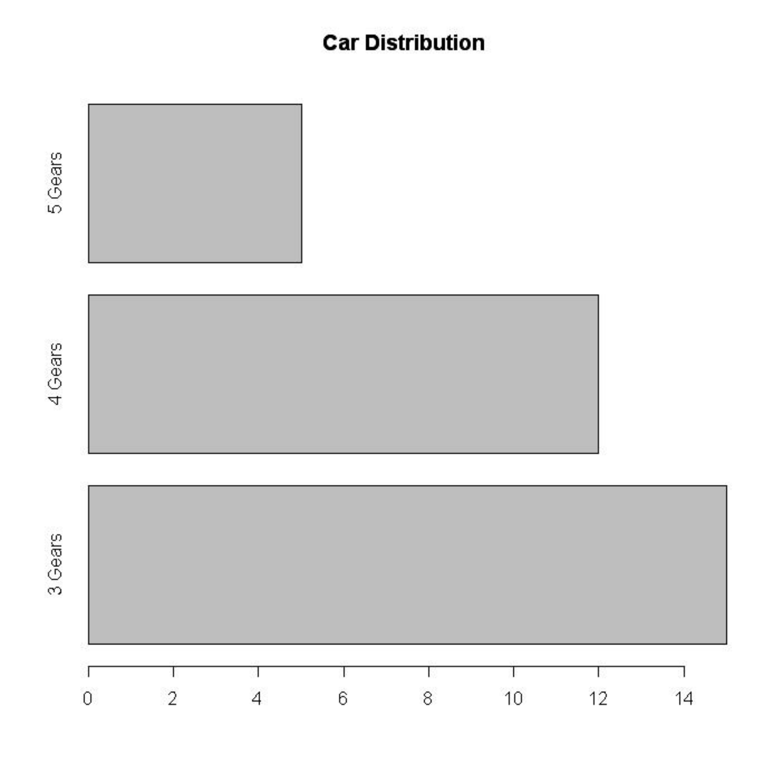
```
# Examples:
x11(height=16,width=16*0.80)
par(mai=c(0.52,0.7,0.2,0.1))
```

Simple Bar Plot

```
#simple bar plot
counts <- table(mtcars$gear)
barplot(counts, main="Car Distribution",
        xlab="Number of Gears")
```

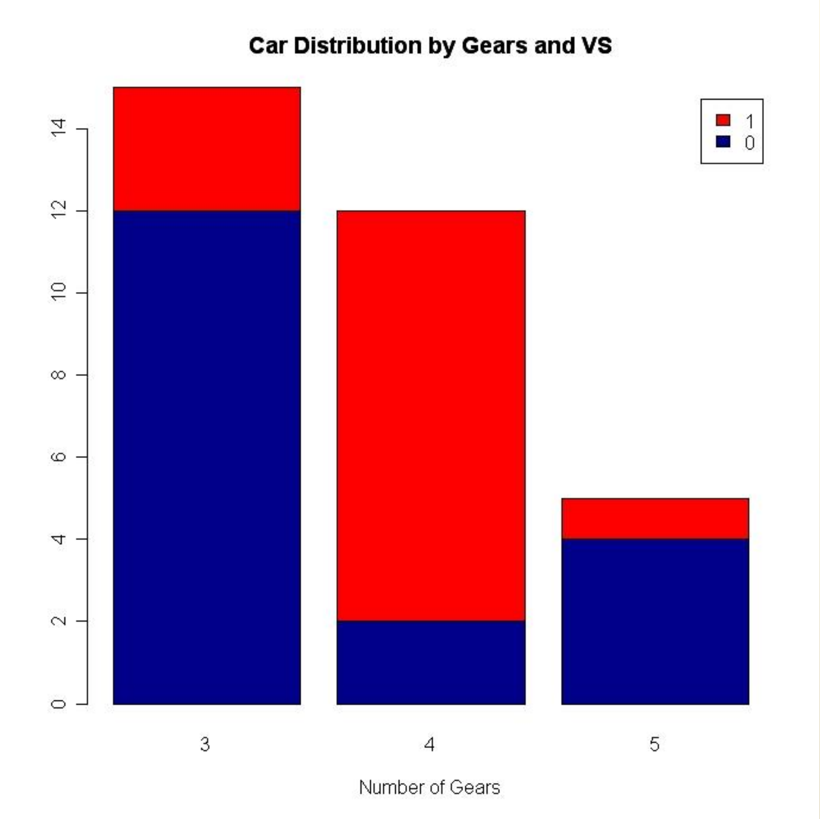
```
# Simple Horizontal Bar Plot with Added Labels
counts <- table(mtcars$gear)
barplot(counts, main="Car Distribution", horiz=TRUE,
  names.arg=c("3 Gears", "4 Gears", "5 Gears"))
```





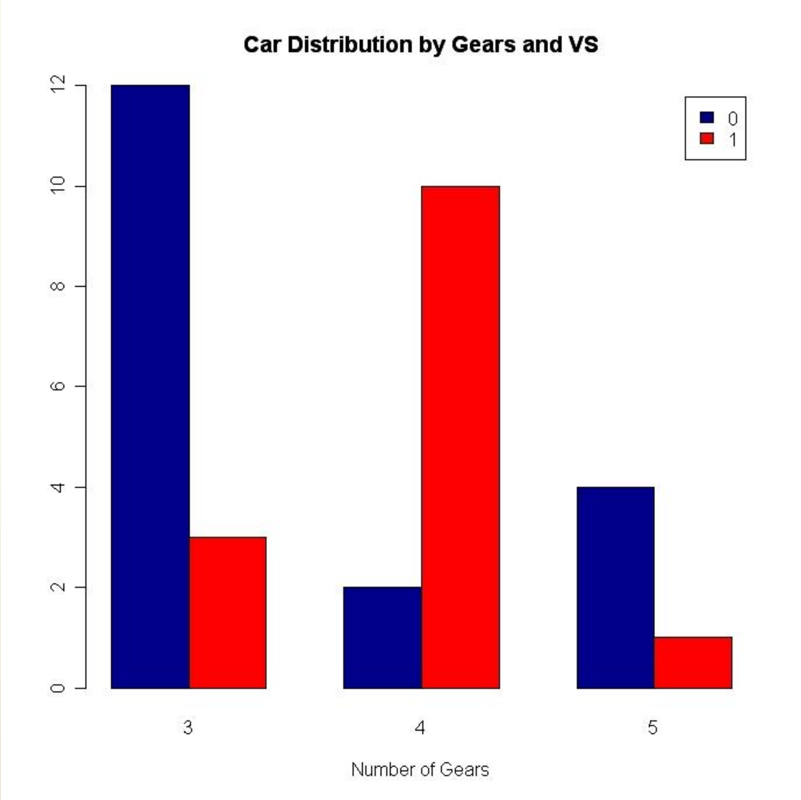
Stacked Bar Plot

```
# Stacked Bar Plot with Colors and Legend
counts <- table(mtcars$vs, mtcars$gear)
barplot(counts, main="Car Distribution by Gears and VS",
        xlab="Number of Gears", col=c("darkblue","red"),
        legend = rownames(counts))
```



Grouped Bar Plot

```
# Grouped Bar Plot
counts <- table(mtcars$vs, mtcars$gear)
barplot(counts, main="Car Distribution by Gears and VS",
        xlab="Number of Gears", col=c("darkblue","red"),
        legend = rownames(counts), beside=TRUE)
```



Advance Graphs: Axes and Texts

In general……

- Many high level plotting functions (plot, hist, boxplot, etc.) allow you to include axis and text options (as well as other graphical parameters).

```
#For example

# Specify axis options within plot()
   plot(x, y, main="title", sub="subtitle",
     xlab="X-axis label", ylab="y-axix label",
     xlim=c(xmin, xmax), ylim=c(ymin, ymax))
```

or use the **title( )** function to add labels to a plot

title(main="*main title*", sub="*sub-title*",
    xlab="*x-axis label*", ylab="*y-axis label*")

# Note: Many other graphical parameters (such as text size, font, rotation, and color) can also be specified in the **title( )** function.

Example:

```
# Add a red title and a blue subtitle. Make x and y
# labels 25% smaller than the default and green.
plot(input$ID1,input$ELEV,xlab="",ylab="")
title(main="My Title", col.main="red",
    sub="My Sub-title", col.sub="blue",
    xlab="My X label", ylab="My Y label",
    col.lab="green", cex.lab=0.75)
```

Text Annotations

- **text( )** and **mtext( )** functions:

# places text within the graph

text(*location,* "*text to place*", pos, ...)

# places text in one of the four margins

mtext("*text to place*", *side*, line=*n*, ...)

- **Labeling points**

```
# Example of labeling points
    attach(mtcars)
    plot(wt, mpg, main="Milage vs. Car Weight",
      xlab="Weight", ylab="Mileage", pch=18, col="blue")
    text(wt, mpg, row.names(mtcars), cex=0.6, pos=4, col="red")
```

- **Math Annotations**

help(plotmath) # for details and examples.

Axes

Format:

axis(*side*, at=, labels=, pos=, lty=, col=, las=, tck=, ...)

```
# A Silly Axis Example

# specify the data
      x <- c(1:10); y <- x; z <- 10/x

# create extra margin room on the right for an axis
      par(mar=c(5, 4, 4, 8) + 0.1)

# plot x vs. y
      plot(x, y,type="b", pch=21, col="red",
        yaxt="n", lty=3, xlab="", ylab="")

# add x vs. 1/x
      lines(x, z, type="b", pch=22, col="blue", lty=2)

# draw an axis on the left
      axis(2, at=x,labels=x, col.axis="red", las=2)

# draw an axis on the right, with smaller text and ticks
      axis(4, at=z,labels=round(z,digits=2),
       col.axis="blue", las=2, cex.axis=0.7, tck=-.01)

# add a title for the right axis
      mtext("y=1/x", side=4, line=3, cex.lab=1,las=2,
      col="blue“)

# add a main title and bottom and left axis labels
      title("An Example of Creative Axes", xlab="X values",
        ylab="Y=X")
```

Reference Lines

Function: abline( )

abline(h=*yvalues*, v=*xvalues*)

# add solid horizontal lines at y=1,5,7
abline(h=c(1,5,7))

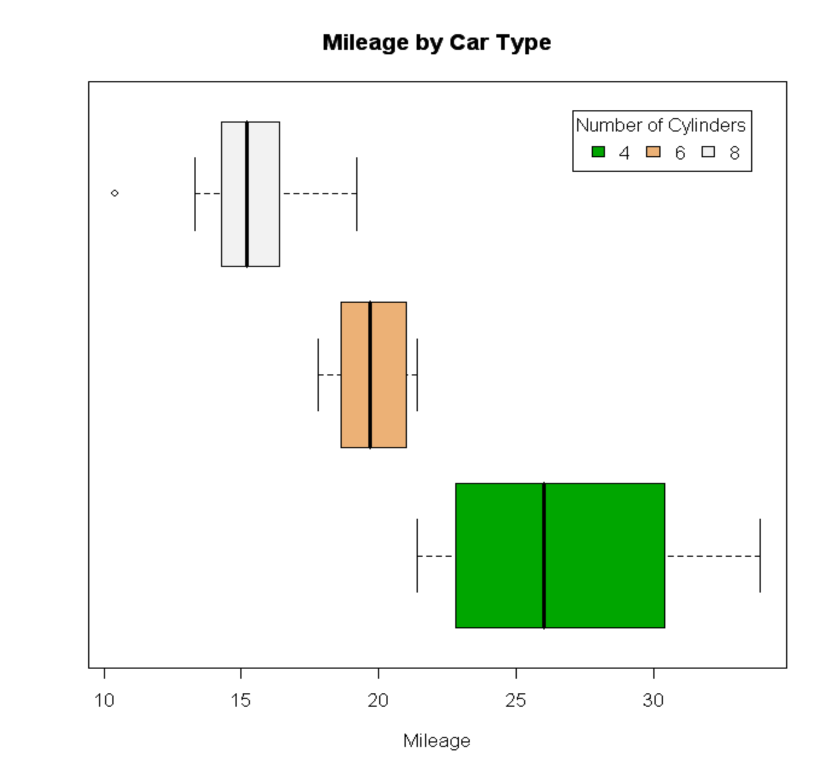
# add dashed blue verical lines at x = 1,3,5,7,9
abline(v=seq(1,10,2),lty=2,col="blue")

# Note: You can also use the **grid( )** function to add reference lines.

Legend

Function: legend()

```
   legend(location, title, legend, ...)
# Legend Example
attach(mtcars)
      boxplot(mpg~cyl, main="Milage by Car Weight",
         yaxt="n", xlab="Milage", horizontal=TRUE,
         col=terrain.colors(3))
      legend("topright", inset=.05, title="Number of
      Cylinders", c("4","6","8"), fill=terrain.colors(3),
      horiz=TRUE)
detach(mtcars)
```



Advance Graphs: Combining plots

Stephan knows all about it!

Combining Plots (1)

Functions: par( ) or **layout( )** function.

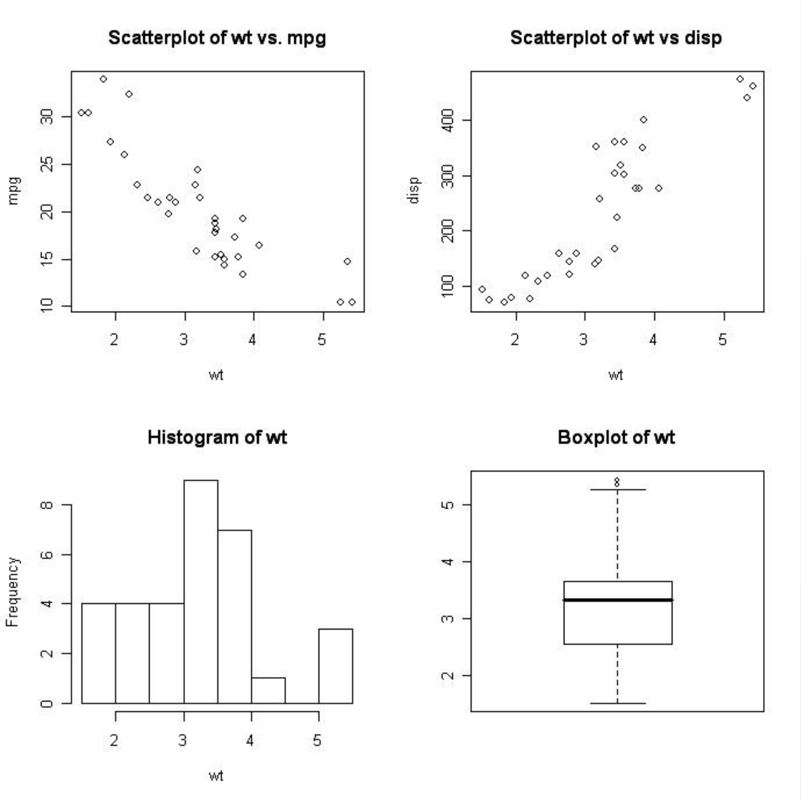
(1) par( )

```
mfrow=c(nrows, ncols)  # plots filled in by row
mfcol=c(nrows, ncols)   # plots fill in the matrix by
      columns
```

#Example:

# 4 figures arranged in 2 rows and 2 columns

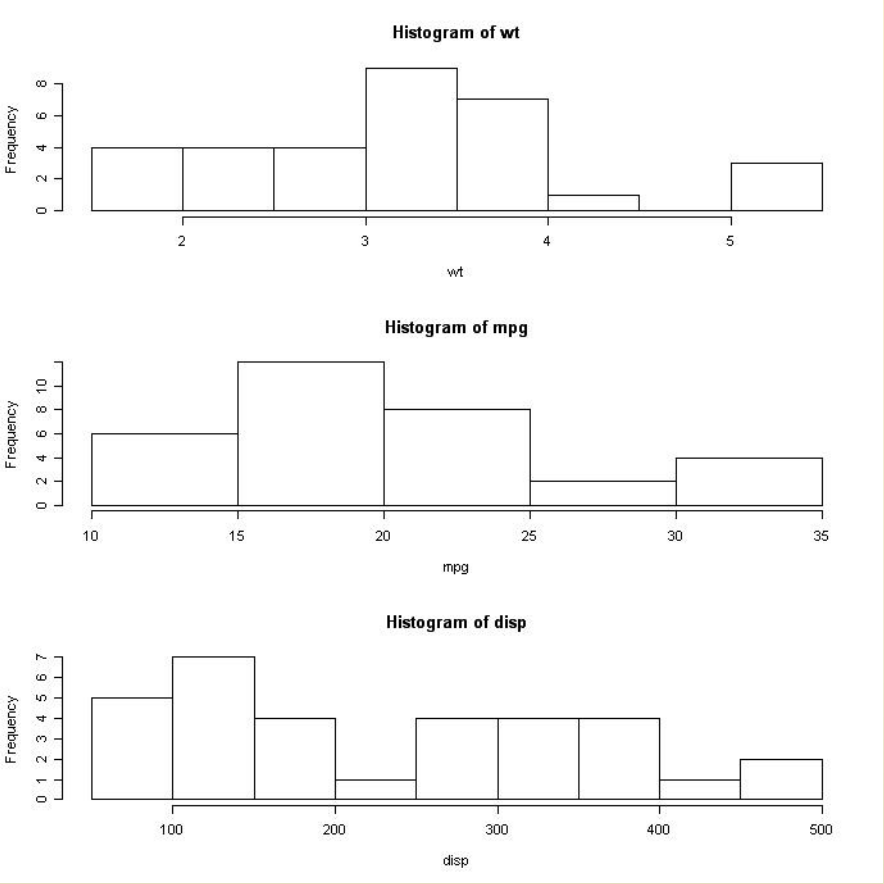
```
attach(mtcars)
  par(mfrow=c(2,2))
  plot(wt,mpg, main="Scatterplot of wt vs. mpg")
  plot(wt,disp, main="Scatterplot of wt vs disp")
  hist(wt, main="Histogram of wt")
  boxplot(wt, main="Boxplot of wt")
detach(mtcars)
```



Combining Plots (2)

```
# 3 figures arranged in 3 rows
    and 1 column


attach(mtcars)
    par(mfrow=c(3,1))
    hist(wt)
    hist(mpg)
    hist(disp)
detach(mtcars)
```



Combining Plots (3)

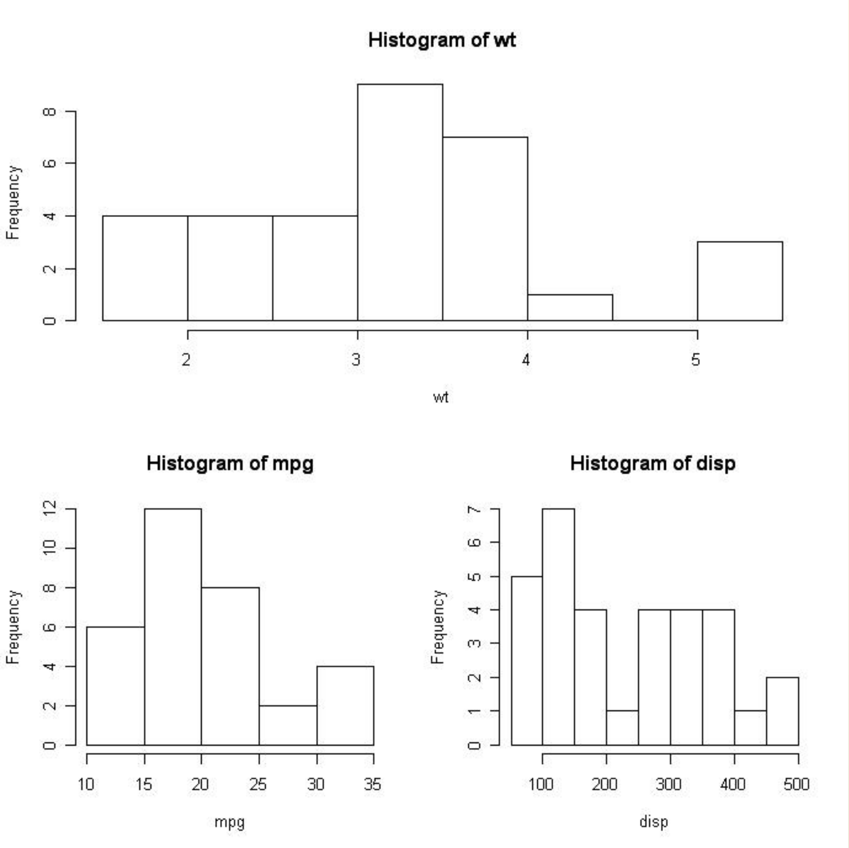
(2) layout( )

layout(*mat*)

mat is a matrix specifying the location of figures to plot.

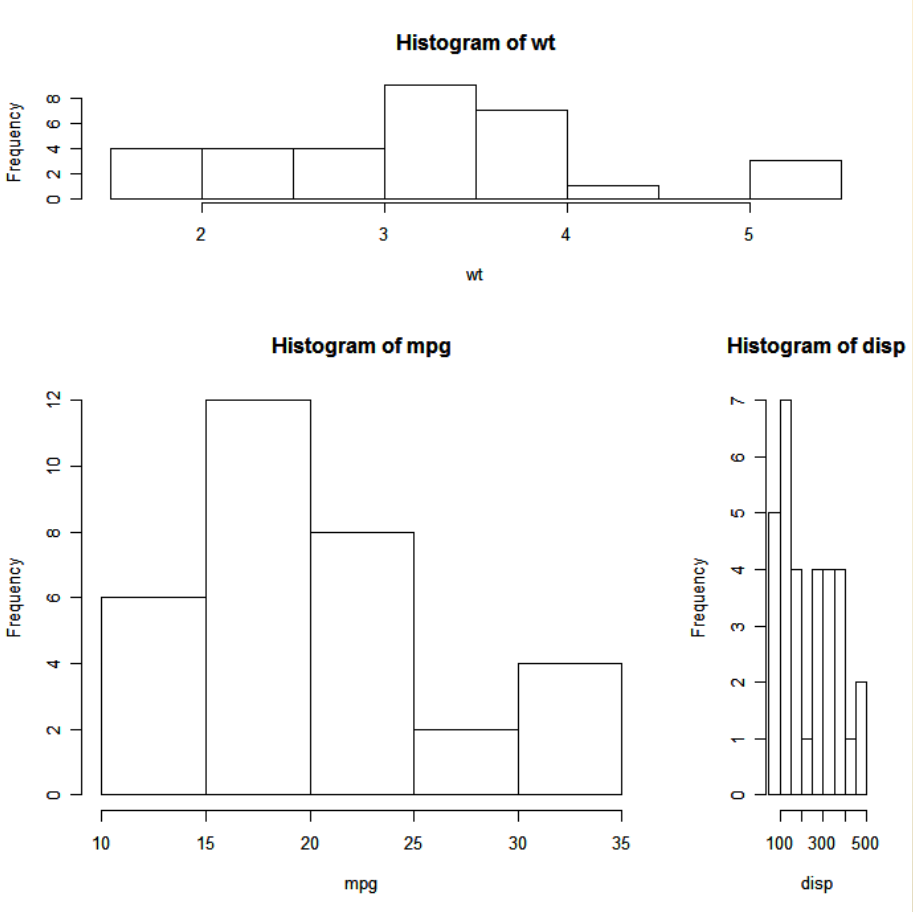
Example:

```
# One figure in row 1 and two figures in row 2
attach(mtcars)
    layout(matrix(c(1,1,2,3), 2, 2, byrow = TRUE))
    hist(wt)
    hist(mpg)
    hist(disp)
detach(mtcars)
```



Becoming a control freak (1)

```
# One figure in row 1 and two figures in row 2
# row 1 is 1/3 the height of row 2
# column 2 is 1/4 the width of the column 1
attach(mtcars)
    layout(matrix(c(1,1,2,3), 2, 2, byrow =
    TRUE), widths=c(3,1), heights=c(1,2))
    hist(wt)
    hist(mpg)
    hist(disp)
detach(mtcars)
```

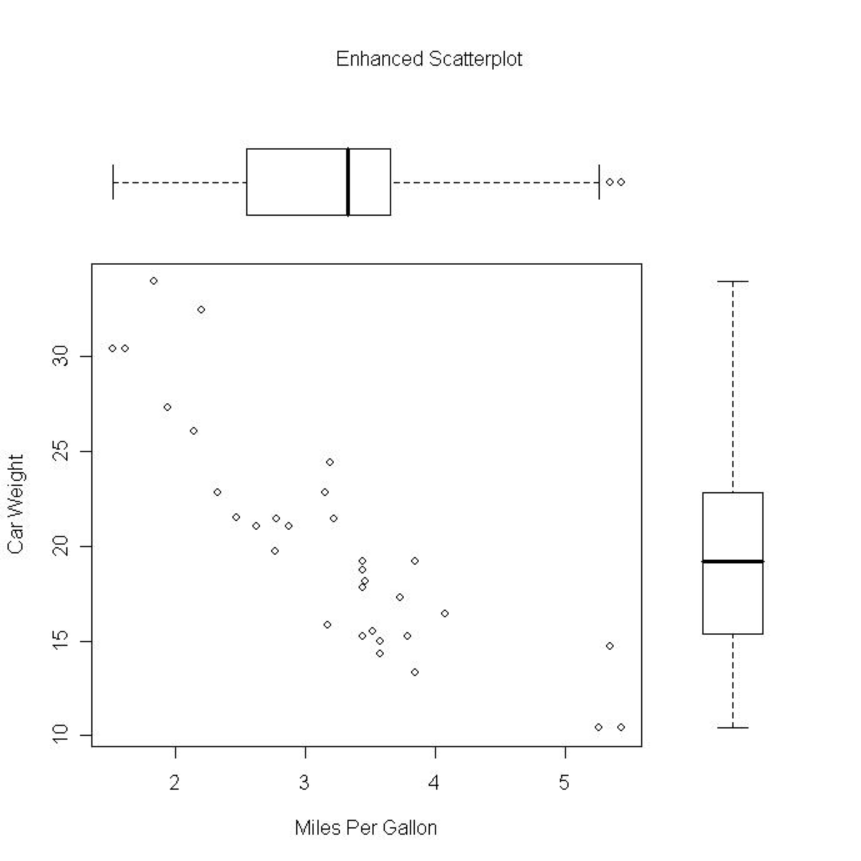


Becoming a control freak (2)

# In the following example, two box plots are added to scatterplot to create an enhanced graph.

# Add boxplots to a scatterplot

```
par(fig=c(0,0.8,0,0.8), new=TRUE)
plot(mtcars$wt, mtcars$mpg,
xlab="Miles Per Gallon",
  ylab="Car Weight")
par(fig=c(0,0.8,0.55,1), new=TRUE)
boxplot(mtcars$wt, horizontal=TRUE,
axes=FALSE)
par(fig=c(0.65,1,0,0.8),new=TRUE)
boxplot(mtcars$mpg, axes=FALSE)
mtext("Enhanced Scatterplot", side=3,
outer=TRUE, line=-3)
```


Enhanced Scatterplot
Car Weight
Miles Per Gallon