Chapter 2: Exploring Data

Wang Shujia

Contents

1 Data

Types of data

- Qualitative: categorical (sometime called ” Factors”)
    - nominal: categories have no ordering. e.g. hair color, male/female, head/tail
    - ordinal: categories have a distinct ordering.e.g. grades of study
- Quantitative: numerical variables
    - Interval data: have interpretable distances
    - Ratio data: have a true zero
- Discrete
    - counts (positive integer): number of accidents, typos per page, number of neurons pulses,
    - binary (0,1 or logical): Head/Tail, True/False, Correct/Incorrect
- Continuous
    - measurements (real number): height of trees

2 Displaying Data

2.1 Displaying Qualitative Data

General Method

Describing a qualitative variable:

- Tables: table() and xtabs()
- Plots: Barplots, Dot Charts, Pie Charts

Frequency table

```
> Grades <- c("A", "D", "C", "D", "C", "C", "C", "C", "F", "B")
> Grades

##  [1] "A" "D" "C" "D" "C" "C" "C" "C" "F" "B"

> table(Grades);xtabs(~Grades);prop.table(table(Grades))

## Grades
## A B C D F
## 1 1 5 2 1
## Grades
## A B C D F
## 1 1 5 2 1
## Grades
##   A   B   C   D   F
## 0.1 0.1 0.5 0.2 0.1
```

Example

The quine data frame in the **MASS** package has information on children from Walgett, New South Wales, Australia, who were classified by ***Culture, Age, Sex***, and ***Learner*** status, as well as the number of ***Days*** absent from school in a particular school year.

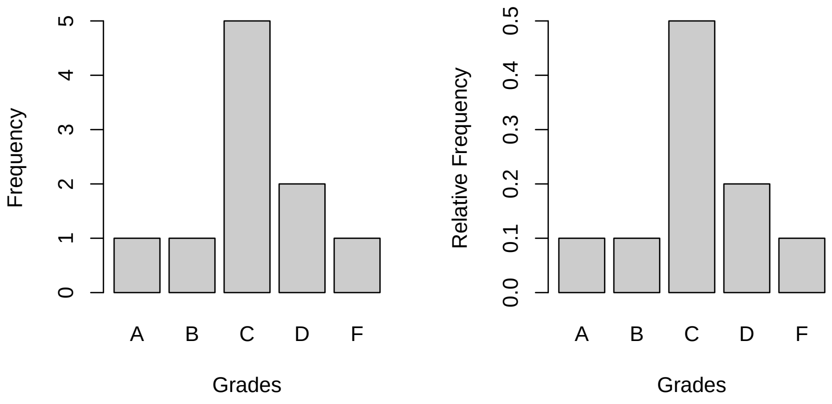
Use the functions **table()** and **xtabs()** to create a frequency table for the variable ***Age***.

Solution

```
> library(MASS)
> table(quine$Age)

##
## F0 F1 F2 F3
## 27 46 40 33

> with(data = quine, table(Age))

## Age
## F0 F1 F2 F3
## 27 46 40 33

> xtabs(~Age, data = quine)

## Age
## F0 F1 F2 F3
## 27 46 40 33
```

Barplots

```
> opar<-par(no.readonly=TRUE)   # read in current parameters
> par(mfrow=c(1,2)) # change parameters
> barplot(xtabs(~Grades),col="gray80",xlab="Grades",ylab="Frequency")
> barplot(prop.table(xtabs(~ Grades)), col = "gray80", xlab = "Grades",
+         ylab = "Relative Frequency")
```



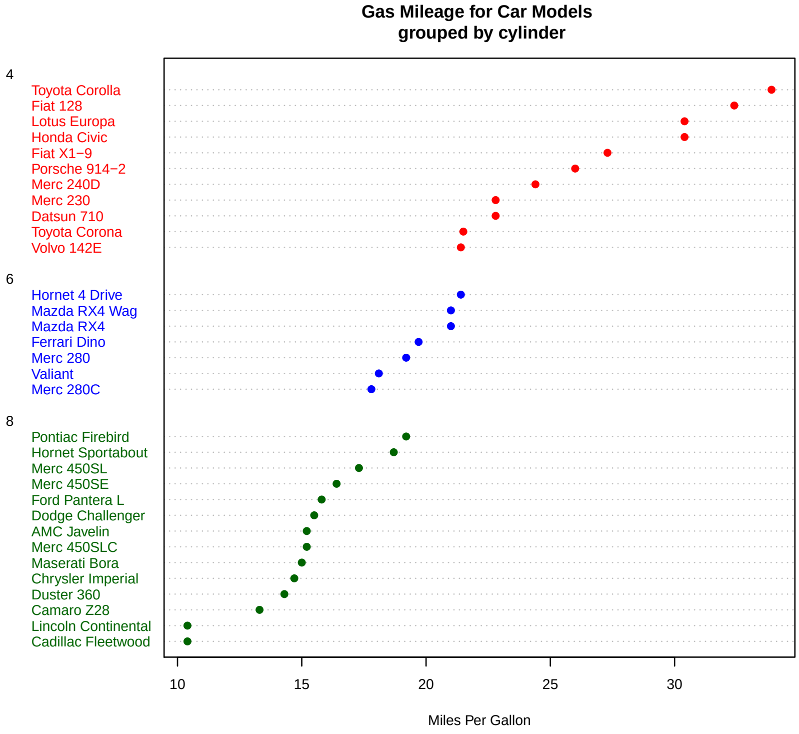
```
> par(opar)  # reset to original parameters
```

Dotchart

R codes for drawing a dotchart of mpg in mtcars data. It's first sorted by *mpg* then plot the dataframe with diferent colours.

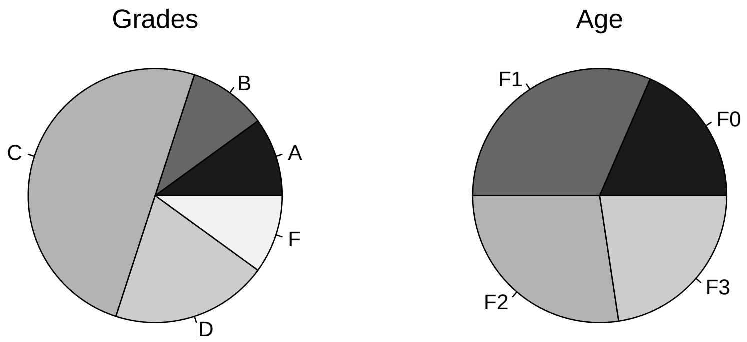
```
x<-mtcars[order(mpg),]
x$cyl<-factor(x$cyl)
x$color[x$cyl==4]<-"red"
x$color[x$cyl==6]<-"blue"
x$color[x$cyl==8]<-"darkgreen"
dotchart(x$mpg,labels=row.names(x),cex=.7,groups=x$cyl,
     gcolor="black",color=x$color,pch=19,
     main="Gas Mileage for Car Models \n grouped by cylinder",
     xlab="Miles Per Gallon")
```

Dotchart



Pie Charts

```
> opar<-par(no.readonly=TRUE) # read in current parameters
> par(mfrow=c(1,2)) # one row two columns
> GS<-gray(c(0.1,0.4,0.7,0.8,0.95))  #different grays
> pie(xtabs(~Grades),radius=1,col=GS)
> mtext("Grades",side=3,cex=1.25,line=1)
> pie(xtabs(~Age,data=quine),radius=1,col=GS)
> mtext("Age",side=3,cex=1.25,line=1)
```



```
> par(opar)    # reset to original parameters
```

2.2 Displaying Quantitative Data

General Method

Describing a quantitative variable:

- ▷ Numerical: descriptive statistics, mean, variance, sd, skewness, kurtosis, etc.
- ▷ graphical: Boxplot,histogram,density curve

Dataset: mtcars

We will consider a data frame in package dataset with 32 observations on 11 variables:

mpg Miles/(US) gallon，油耗

cyl Number of cylinders 汽缸数

disp Displacement (cu.in.)，排量

hp Gross horsepower，总马力

drat Rear axle ratio，后轴比

wt Weight (lb/1000)

qsec 1/4 mile time

vs V/S

am Transmission (0 = automatic, 1 = manual)，变速箱

gear Number of forward gears，前齿轮数

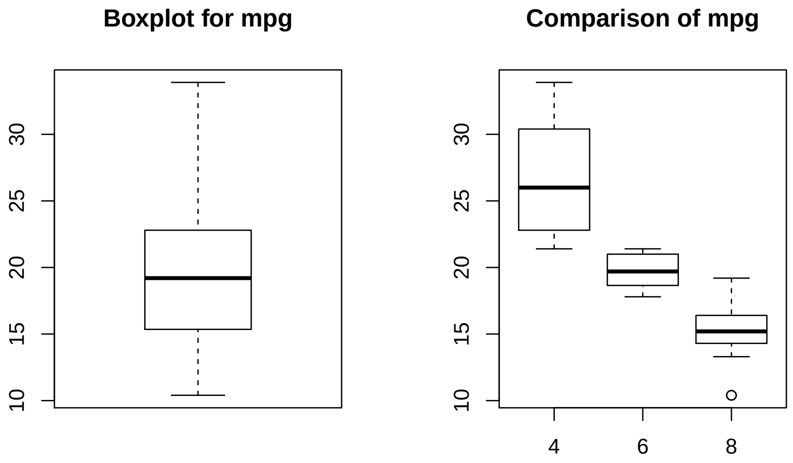
carb Number of carburetors，化油器数

mtcars

```
> head(mtcars)

##                    mpg cyl disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
## Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1

> str(mtcars)

## 'data.frame': 32 obs. of  11 variables:
##  $ mpg : num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
##  $ cyl : num  6 6 4 6 8 6 8 4 4 6 ...
##  $ disp: num  160 160 108 258 360 ...
##  $ hp  : num  110 110 93 110 175 105 245 62 95 123 ...
##  $ drat: num  3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
##  $ wt  : num  2.62 2.88 2.32 3.21 3.44 ...
##  $ qsec: num  16.5 17 18.6 19.4 17 ...
##  $ vs  : num  0 0 1 1 0 1 0 1 1 1 ...
##  $ am  : num  1 1 1 0 0 0 0 0 0 0 ...
##  $ gear: num  4 4 4 3 3 3 3 4 4 4 ...
##  $ carb: num  4 4 1 1 2 1 4 2 2 4 ...
```

Boxplot

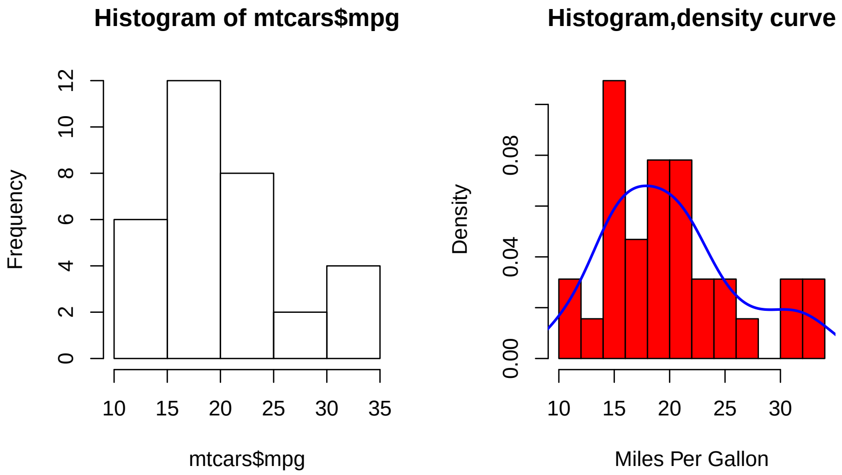
```
> opar<-par(no.readonly=TRUE) # read in current parameters
> par(mfrow=c(1,2))
> boxplot(mtcars$mpg,main="Boxplot for mpg")
> boxplot(mtcars$mpg~mtcars$cyl,main="Comparison of mpg")
```



```
> par(opar)    # reset to original parameters
```

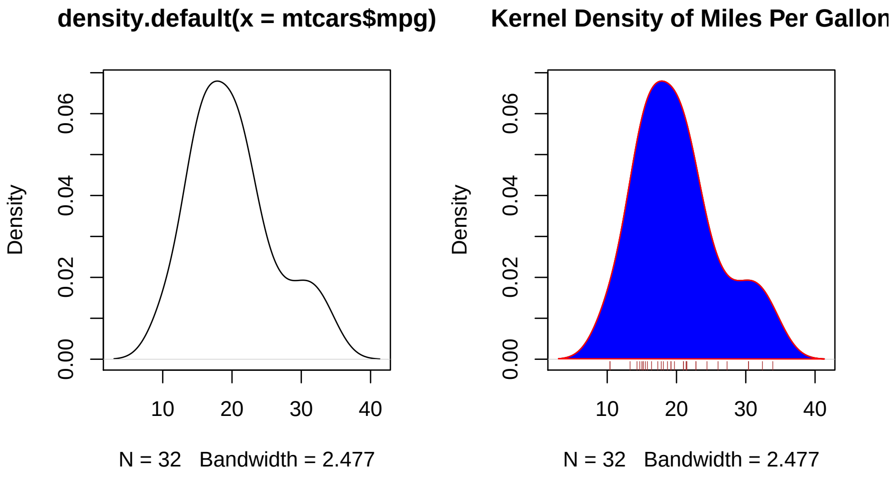
Histogram

```
> opar<-par(no.readonly=TRUE);par(mfrow=c(1,2))
> hist(mtcars$mpg)
> hist(mtcars$mpg,freq=FALSE,breaks=12,col="red",xlab="Miles Per Gallon",
+ main="Histogram,density curve")
> lines(density(mtcars$mpg),col="blue",lwd=2);par(opar)
```



Kernel density plots

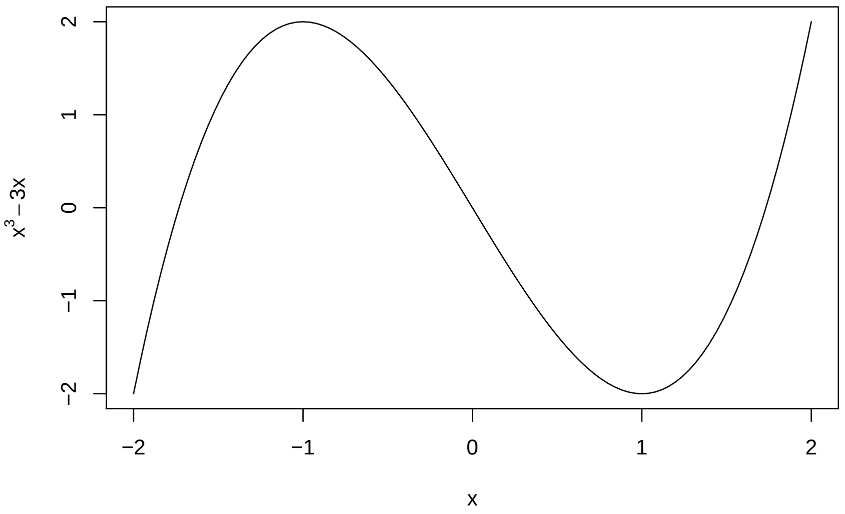
```
> opar<-par(no.readonly=TRUE) # read in current parameters
> par(mfrow=c(1,2))
> d<-density(mtcars$mpg);plot(d)
> plot(d,main="Kernel Density of Miles Per Gallon")
> polygon(d,col="blue",border="red");rug(mtcars$mpg,col="brown")
```



```
> par(opar)    # reset to original parameters
```

Curve

```
> curve(x^3-3*x,from=-2,to=2,ylab=expression(x^3-3*x))
```



3 Descriptive Statistics

Descriptive Statistics

Numerical summaries of the *population* are called **parameters**, while numerical summaries of the *sample* are called **statistics**.

- Summary Measures of Location
  - mean, median, mode, quantiles,
- Summary Measures of Spread
  - range, interquartile-range(IQR), variance, standard-deviation(sd),The Median Absolute Deviation (MAD)
- Summary Measures of Shape
  - skewness, kurtosis

3.1 Summary Measures of Location

R functions for location

- Population mean: μ
- Sample mean: $\bar{x}$
- R functions: mean(x), median(x), mode(x)
- Quantiles: the x_p is called a p**-quantile** of a distribution, if $\mathrm{P}(X \leq x_p) \geq p$ and $\mathrm{P}(X \geq x_p) \leq 1 - p$
  - for continuous r.v., $\mathrm{P}(X \leq x_p) = p$
- quantile(x, probs=c(0.25, 0.5, 0.75)): Q_1, Q_2, Q_3

Mtcar data

```
> attach(mtcars)
> mean(mpg)

## [1] 20.09062

> median(mpg)

## [1] 19.2

> quantile(mpg,probs=c(0.25,0.5,0.75))

##    25%    50%    75%
## 15.425 19.200 22.800

> detach(mtcars)
```

3.2 Summary Measures of Spread

functions

- ▷ range(x): returns the smallest and largest values in x
- ▷ IQR(x): Interquartile Range, IQR= Q3 - Q1
- ▷ var(x): $s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2$
- ▷ sd(x): $s = \sqrt{s^2}$
- ▷ Sample Coeffcient of Variation: $CV = S/\bar{X}$
- ▷ Relative Standard Deviation: $RSD = |S/\bar{X}| \times 100$
- ▷ The Median Absolute Deviation (MAD): is a robust measure of spread, often used when the median is reported to describe the center of a *skewed data set*.

  $$MAD = \text{median}\{|x_i - m|\}$$

  where m is the median of x.

3.3 Summary Measures of Shape

Skewness and Kurtosis

The base installation of R doesn't provide functions for skew and kurtosis

- ▷ **Skewness** is a measure of the asymmetry of the probability distribution of a real-valued random variable about its mean.
- ▷ **Kurtosis** is a measure of the ”tailedness” of the probability distribution of a real-valued random variable.

$$\text{Skew} = \frac{1}{n}\sum_{i=1}^{n}(\frac{x_i - \bar{x}}{s})^3$$

$$\text{Kurt} = \frac{1}{n}\sum_{i=1}^{n}(\frac{x_i - \bar{x}}{s})^4$$

- ▷ Negative skew: The left tail is longer; the mass of the distribution is concentrated on the right of the figure.The distribution is said to be *left-skewed, left-tailed*, or *skewed to the left*
- ▷ Positive skew: *right-skewed, right-tailed*, or *skewed to the right*
- ▷ The ***excess kurtosis*** = *Kurt - 3* (Kurt=3 for Normal)

Own-written function for descriptive statistics

```
> mystats<-function(x,na.omit=FALSE){
+   if(na.omit)
+      x<-x[!is.na(x)]
+   m<-mean(x)
+   n<-length(x)
+   s<-sd(x)
+   skew<-sum((x-m)^3/s^3)/n
+   kurt<-sum((x-m)^4/s^4)/n - 3
+ return(c(n=n,mean=m,stdev=s,skew=skew,kurtosis=kurt))
+ }
> round(mystats(mtcars$mpg),3)

##        n     mean    stdev     skew kurtosis
##   32.000   20.091    6.027    0.611   -0.373
```

4 General graphical methods

4.1 High-level plotting functions

The four graphics systems in R

graphics The base graphics system, written by Ross Ihaka, is included in every R installation. It is easy to produce plots but it is difficult to produce good plots

grid Written by Paul Murrell (2011), Offer a lower-level alternative to the standard graphics system. Doesn’ t provide functions for producing statistical graphics or complete plots

lattice Written by Deepayan Sarkar (2008) during his Ph.D at U. of Wisconsin. Displays plots separately for each level of one or more other variables. Good but is a bit advanced for most people

ggplot2 Written by Hadley Wickham (2009) during his Ph.D at Iowa State. Provide a comprehensive, grammar-based system for generating graphs in a unified and coherent manner, is becoming more important for visualizing data

Three types of plotting commands

High-level plotting functions create a new plot (usually with axes, labels, titles and so on)

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

Interactive graphics functions allow you to interactively add information to an existing plot or to extract information from an existing plot using the mouse

The plot() function

The standard high-level plotting function is plot().

plot(x, y) If x and y are numerical vectors, then plot(x, y) produces a scatterplot of y against x.

plot(y) If y is a numerical vector, then this is (almost) the same as plot(1:length(y), y).

plot(f) If f is a factor, then plot(f) is a barplot of f.

plot$(f,\ y)$ Iff is a factor and y is a numeric vector, then plot(f, y) produces boxplots of y for each level of f.

plot(fun) If fun is a function, then plot(fun, from=a, to=b) plots fun in the range $[a, b]$.

plot(df) Distributional plots of variables in data frame df

plot$(\sim x + y)$ Distributional plots of the variables x and y.

plot$(y \sim x_1 + x_2)$ Plots y against x_1 and x_2 repectively.

Other high-level graphics functions

barplot displays the distribution (frequency) of a categorical variable

dotchart plots a large number of labeled values on a simple horizontal scale

mosaicplot shows a set of boxes corresponding to different factor values

spineplot shows different boxes corresponding to the number of observations associated with two factors

hist displays the distribution of a continuous variable by dividing the range of scores into a specified number of bins on the x-axis

density kernel density plots

curve draws a curve on the interval specified by the bounds *from* and *to*

smoothScatter smooth scatter plots

pairs pairwise scatterplot matrix

contour draws contour lines

persp draws a 3D surface

Plots for different types of data

▷ Visualizaton of categorical data

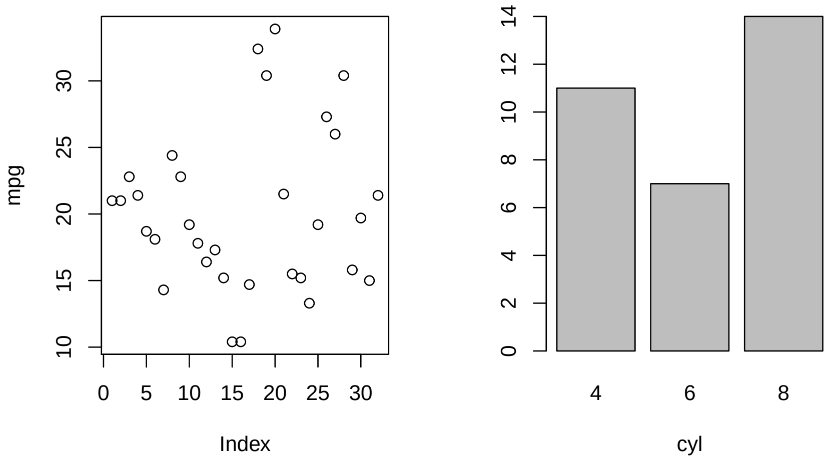
- univariate: bar chart, dot chart, and the pie chart and its variants
- multivariate(contingency table): spine and mosaic plots

▷ Visualizaton of contnuous data

- univariate: boxplot, histogram
- multivariate(correlations of variables): scater plot and its variants

Plot a single variable

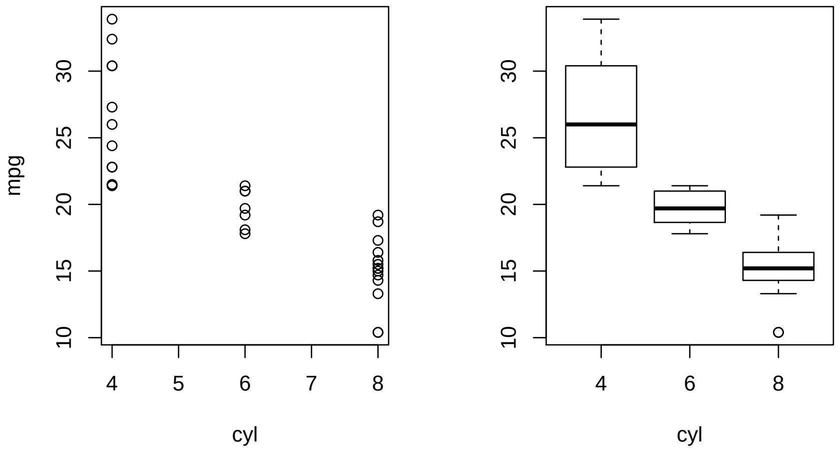
```
> attach(mtcars)
> par(mfrow=c(1,2))
> plot(mpg) #plot a numerical vector
> plot(factor(cyl),xlab="cyl") #plot a factor
```



```
> detach(mtcars)
```

Plot bivariate variables

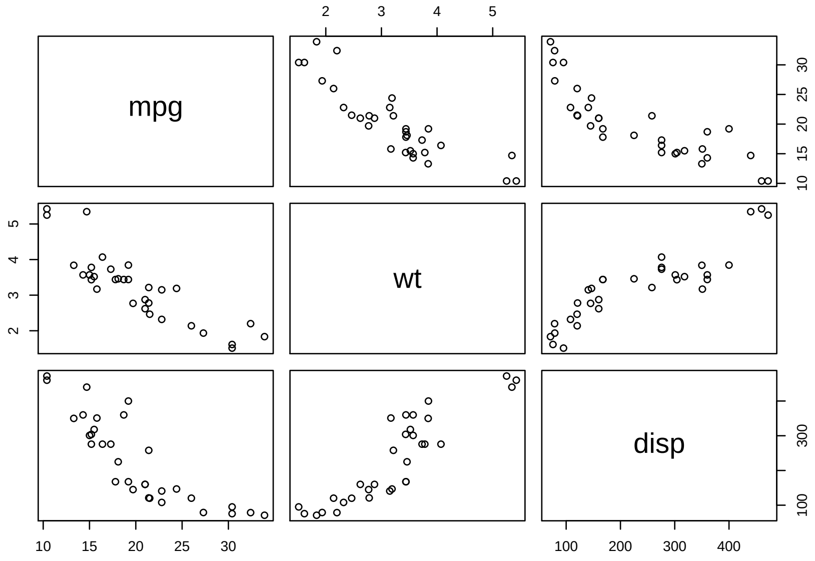
```
> attach(mtcars)
> par(mfrow=c(1,2))
> plot(cyl,mpg) #plot(x,y)
> plot(factor(cyl),mpg,xlab="cyl") #plot(factor,y)
```



```
> detach(mtcars)
```

Plot several variables

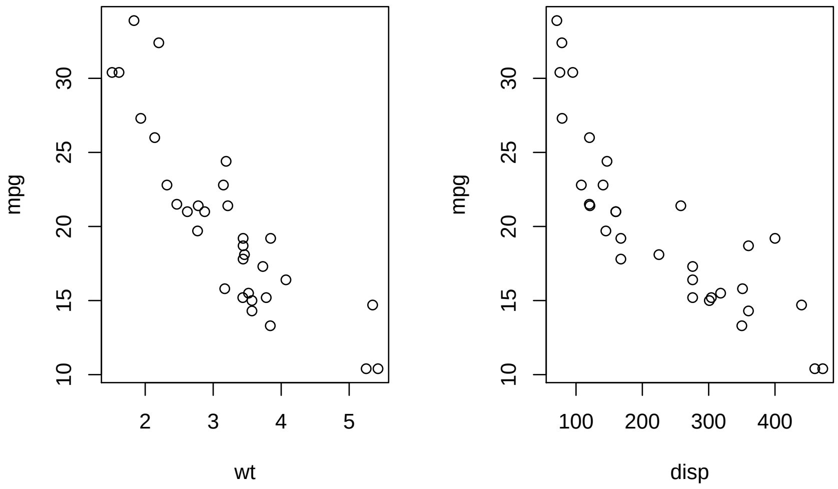
```
> attach(mtcars)
> plot(~mpg+wt+disp)
```



```
> detach(mtcars)
```

Plot several variables

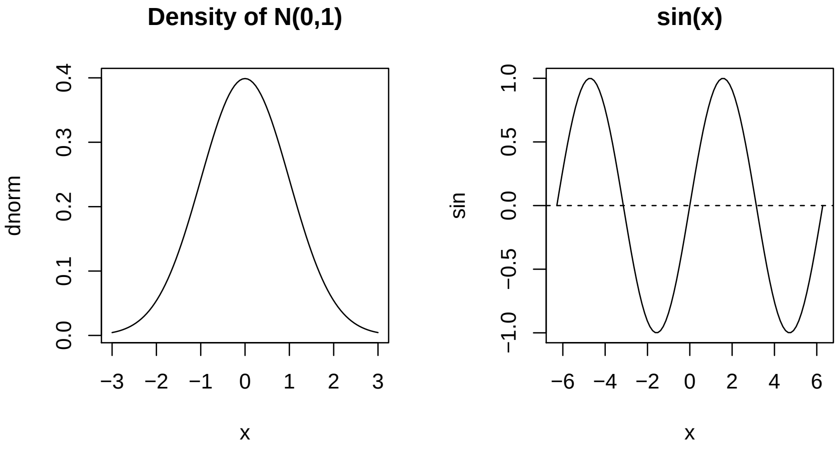
```
> attach(mtcars)
> par(mfrow=c(1,2))
> plot(mpg~wt+disp)
```



```
> detach(mtcars)
```

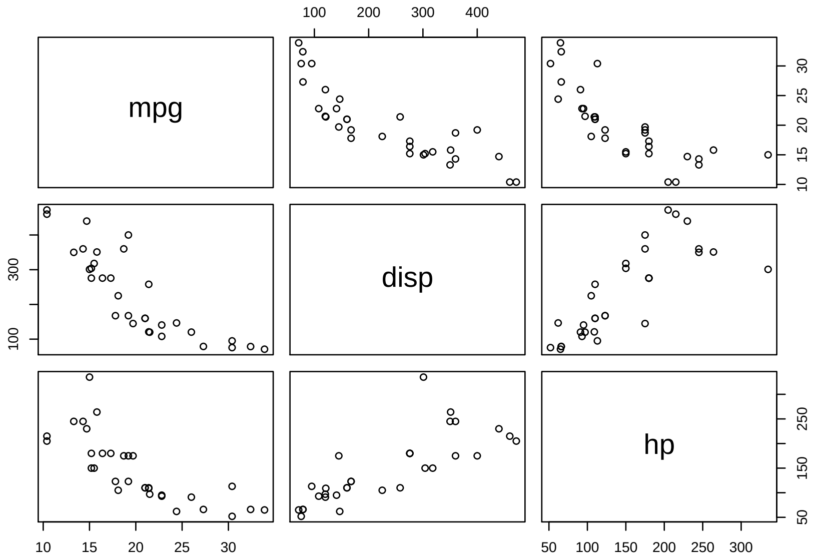
Plot a density curve and a function

```
> par(mfrow=c(1,2))
> plot(dnorm,from=-3,to=3,main="Density of N(0,1)") #plot(fun)
> plot(sin,from=-2*pi,to=2*pi,main="sin(x)") #plot(fun)
> abline(h = 0,lty=2)
```



Plot a data frame

```
> attach(mtcars)
> df<-data.frame(mpg,disp,hp)
> plot(df)
```



```
> detach(mtcars)
```

Arguments to high-level plotting functions

type The type of plot

- **type=”p”** Plot individual points (the default)
- **type=”l”** Plot lines
- **type=”b”** Plot points connected by lines (both)
- **type=”o”** Plot points overlaid by lines
- **type=”h”** Plot vertical lines from points to the zero axis (high-density)
- **type=”s”** Step-function plots, the top of the vertical defines the point
- **type=”S”** Step-function plots, the bottom of the vertical defines the point
- **type=”n”** plot nothing (but to display the window, with axes)

Arguments to high-level plotting functions

main Specify the main title

sub Specify the subtitle

xlab Specify the label of the x axis

ylab Specify the label of the y axis

xlim Vector of length 2. Specify the lower and upper bound for the x axis

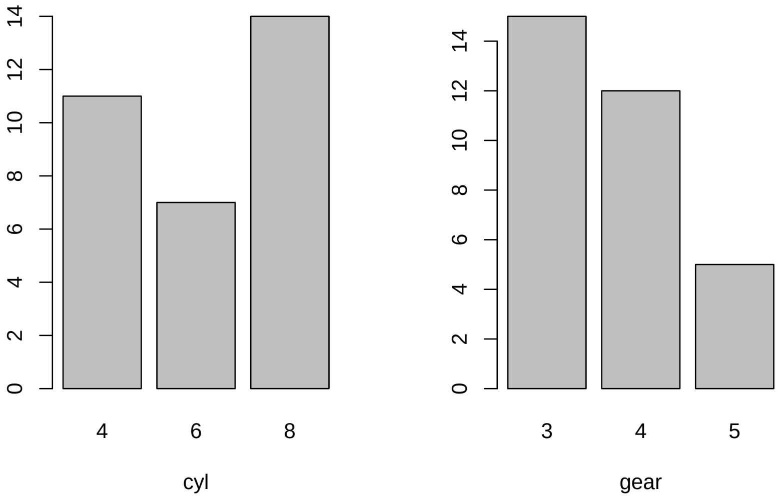
ylim Vector of length 2. Specify the lower and upper bound for the y axis

log log=”x” causes the x axes to be logarithmic. log=”y”, ”xy” or ”yx” respectively the y axis or both is to be logarithmic

cex Amount by which plotting text and symbols should be magnified relative to the default

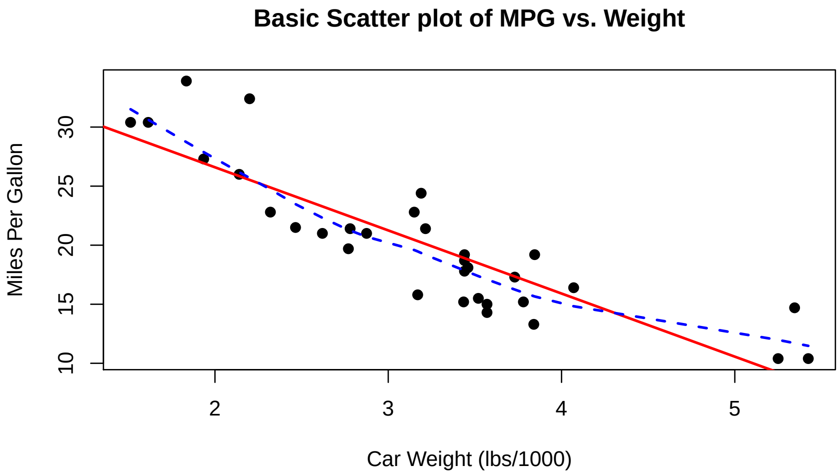
Barplot

```
> par(mfrow=c(1,2))
> barplot(table(mtcars$cyl),xlab="cyl")
> barplot(table(mtcars$gear),xlab="gear")
```



Scatter plots

```
> attach(mtcars)
> plot(wt,mpg,main="Basic Scatter plot of MPG vs. Weight",
+      xlab="Car Weight (lbs/1000)",ylab="Miles Per Gallon",pch=19)
> abline(lm(mpg~wt),col="red", lwd=2, lty=1)
> lines(lowess(wt,mpg),col="blue",lwd=2,lty=2)
```



```
> detach(mtcars)
```

4.2 Low-level plotting functions

Useful low-level plotting functions

Low-level plotting commands can be used to add extra information (such as points, lines or text) to the current plot.

points(x, y) Adds points to the current plot

lines(x, y) Adds lines to the current plot

text$(x, y, labels, ...)$ Add text to a plot at points given by x, y. To display a *mathematical expression*, use the functions `expression()`and `bquote()`.

abline(a, b) Adds a line of slope b and intercept a to the current plot

abline$(h = y)$ Specify horizontal lines to go across a plot

abline$(v = x)$ Specify vertical lines to go across a plot

abline$(lm.obj)$ Specify lines of model-fitting functions, e. g. abline$(lm(y \sim x))$

Useful low-level plotting functions

title(main=main,sub=sub) Adds a title main to the top of the current plot in a large font and (optionally) a sub-title sub at the bottom in a smaller font

axis(side=side,...) Adds an axis to the current plot on the side given by the first argument (1 to 4, counting clockwise from the bottom) Useful for adding custom axes after calling `plot()` with the `axes=FALSE` argument

polygon$(x, y, ...)$ Draws a polygon defined by the ordered vertices in (x, y) and (optionally) shade it in with hatch lines, or fill it if the graphics device allows the filling of figures

arrows($x_0, y_0, x_1, y_1, ...$) Draw arrows from (x_0, y_0) to (x_1, y_1)

pretty() Calculate a 'pretty' scaling of the axis

plot.new() Empty the current plotting window (open a new window if none is open)

mtext() Write text in the margins

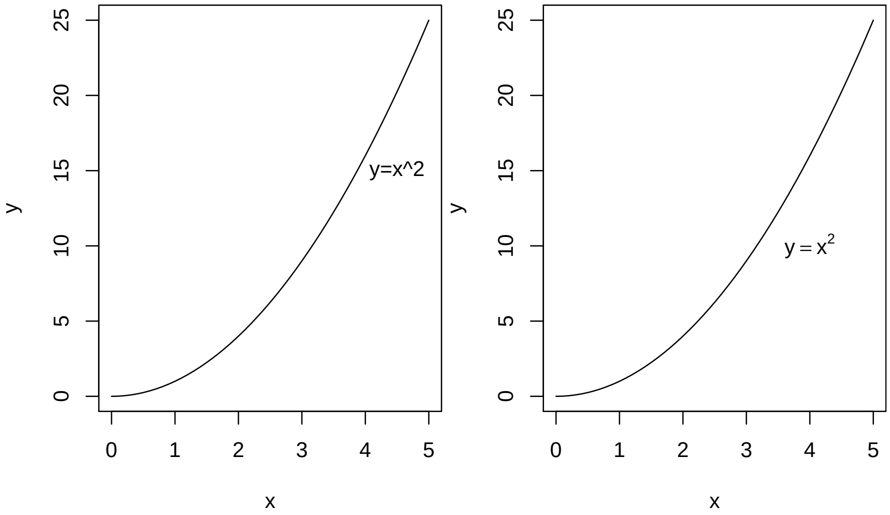
Useful low-level plotting functions

legend(x,y,legend,...) Adds a legend to the current plot at the specified position. Plotting characters, line styles, colors etc., are identified with the labels in the character vector legend. At least one other argument v (a vector the same length as legend) with the corresponding values of the plotting unit must also be given, as follows:

- ▷ `legend(,fill=v)`: Colors for filled boxes
- ▷ `legend(,col=v)`: Colors in which points or lines will be drawn
- ▷ `legend(,lty=v)`: Line styles
- ▷ `legend(,lwd=v)`: Line widths
- ▷ `legend(,pch=v)`: Plotting characters (character vector)

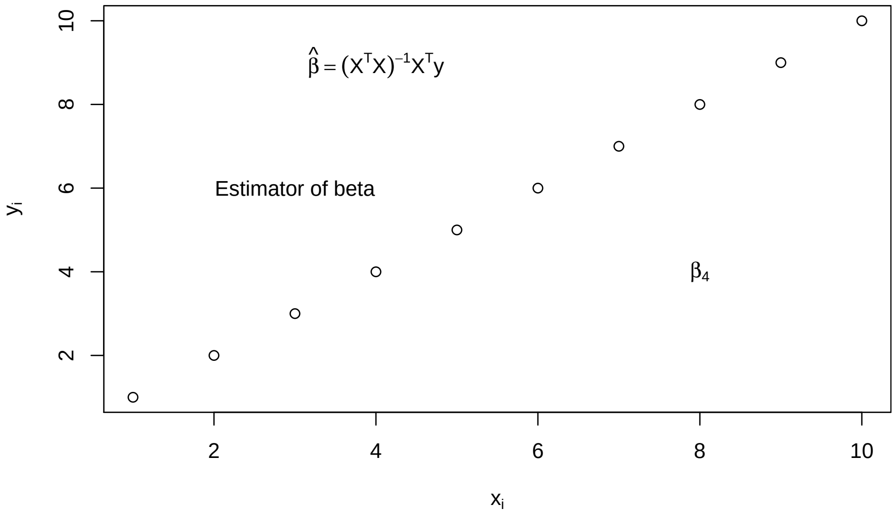
text()

```
> par(mfrow=c(1,2))
> x<-seq(from=0,to=5,by=0.1);y<-x^2
> plot(x,y,type="l");text(4.5,15,"y=x^2")
> plot(x,y,type="l");text(4,10,expression(y==x^2))
```



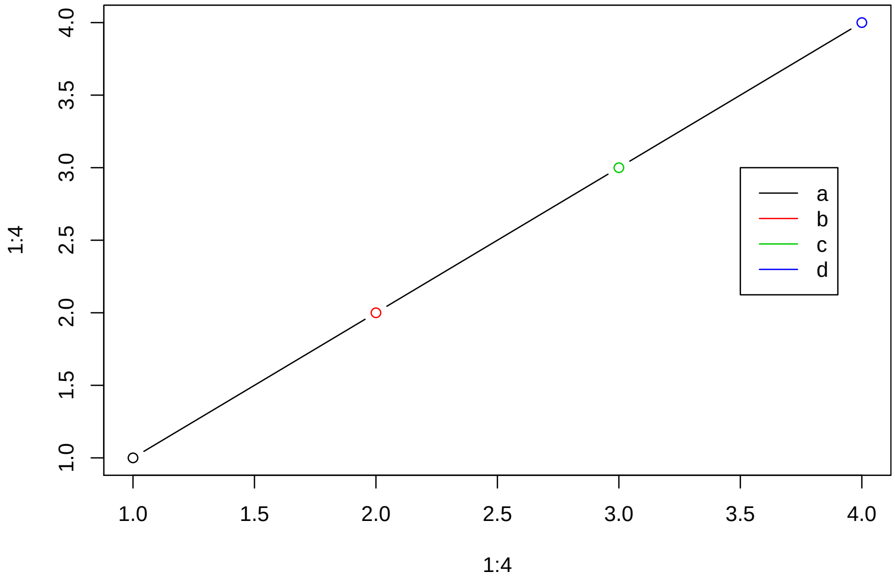
text()

```
> plot(1:10,1:10,xlab=bquote(x[i]),ylab=bquote(y[i]))
> text(3,6,"Estimator of beta")
> text(4,9,expression(hat(beta)==(X^T*X)^{-1}*X^T*y))
> p<-4;text(8,4,bquote(beta[.(p)])) # Combining "math" and numerical variables
```



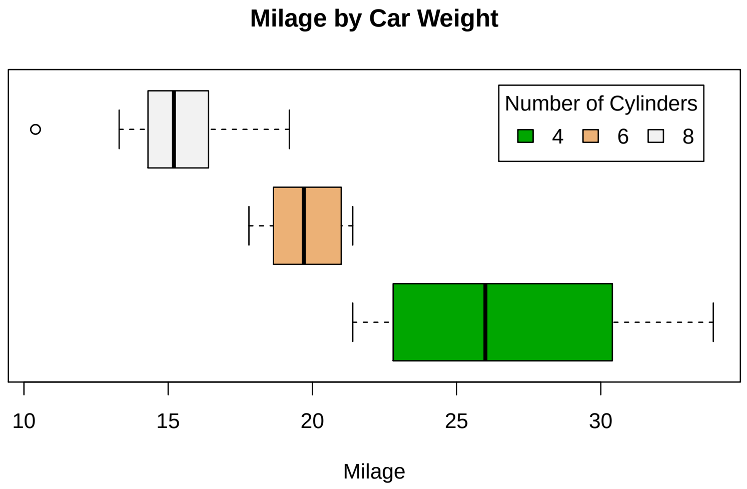
legend()

```
> plot(1:4,1:4,col=1:4,type="b")
> legend(x=3.5,y=3,legend=c("a","b","c","d"),col=1:4,lty=1)
```



legend()

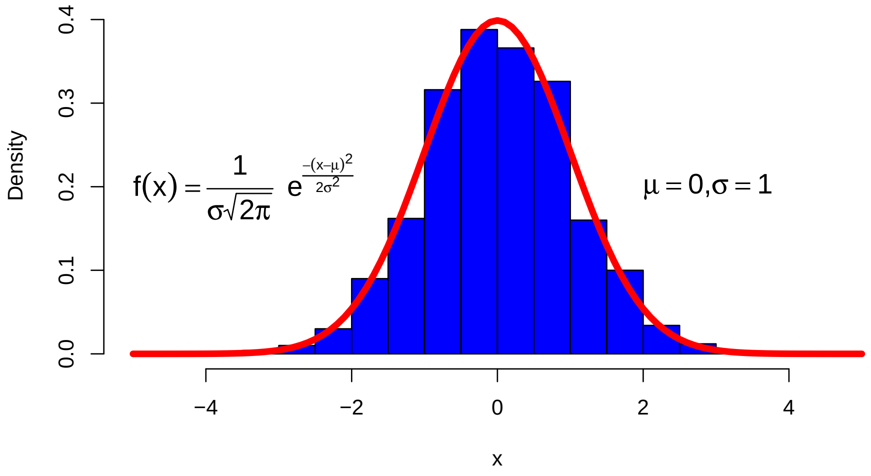
```
> attach(mtcars)
> boxplot(mpg~cyl,main="Milage by Car Weight",yaxt="n",
+ xlab="Milage",horizontal=TRUE,col=terrain.colors(3))
> legend("topright",inset=.05,title="Number of Cylinders",
+ c("4","6","8"),fill=terrain.colors(3),horiz=TRUE)
```



```
> attach(mtcars)
```

Normal density

```
> x<-rnorm(1000)
> hist(x,main="",probability=TRUE,col="blue",ylab="Density",
+ xlim=c(-5,5),ylim=c(0,0.45))
> plot(dnorm,from=-5,to=5,add=TRUE,lwd=5,lty=1,col="red")
> text(-5,0.2,adj=0,cex=1.3,expression(f(x)==frac(1,sigma*sqrt(2*pi))
+ ~~e^{frac(-(x-mu)^2, 2*sigma^2)}))
> text(2,0.2,adj=0,cex=1.3,expression({mu==0}*","*{sigma==1}))
```



4.3 Graphical parameters

The par() function

You can customize many features of your graphs (fonts, colors, axes, titles) through graphic options.

- One way is to specify these options in through the par( ) function. If you set parameter values here, the changes will be in effect for the rest of the session or until you change them again.

```
par(optionname=value, optionname=value, ...)
```

- A second way to specify graphical parameters is by providing the `optionname=value` pairs directly to a high level plotting function. In this case, the options are only in effect for that specific graph.

The par() function

```
# 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")
```

Text and Symbol Size

The following options can be used to control **text** and **symbol** size in graphs:

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

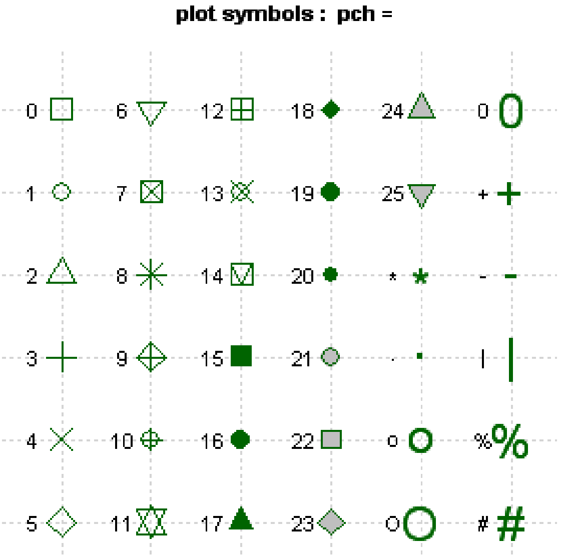
cex.lab magnification of x and y labels relative to cex

cex.main magnification of titles relative to cex

cex.sub magnification of subtitles relative to cex

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=).

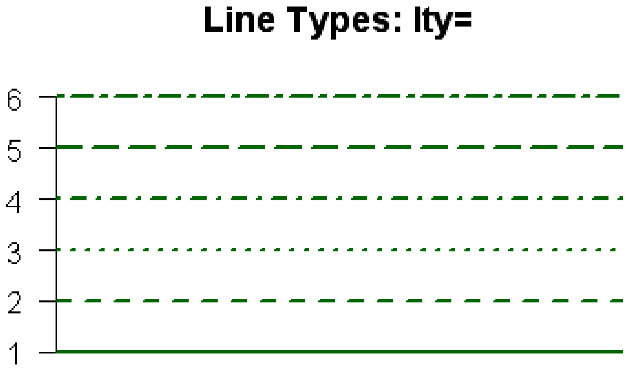


Lines

You can change lines using the following options. This is particularly useful for reference lines, axes, and fit lines.

lty line type. see the chart below.

lwd line width relative to the default (default=1). 2 is twice as wide.



Colors

Options that specify colors include the following:

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

Fonts

You can easily set font size and style, but font family is a bit more complicated.

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.

Margins and Graph Size

You can control the margin size using the following parameters:

mar numerical vector indicating margin size `c(bottom, left, top, right)` in lines. default = c(5, 4, 4, 2) + 0.1

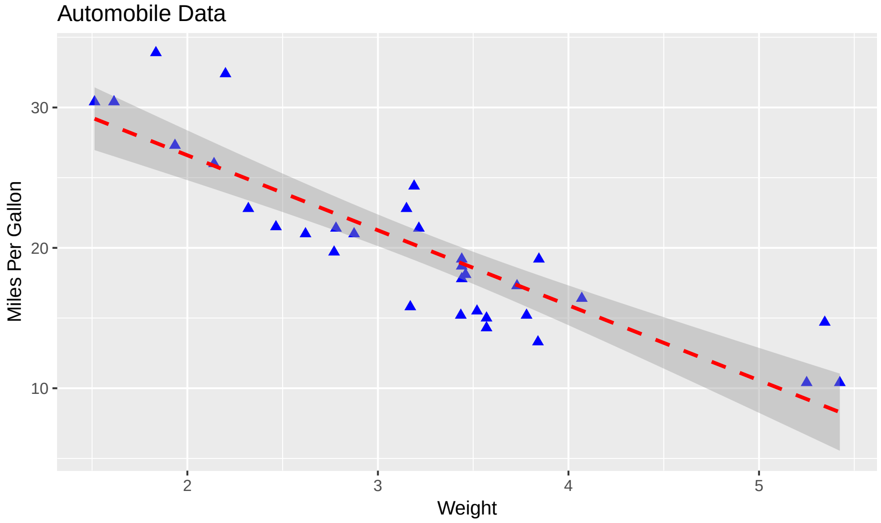
mai numerical vector indicating margin size `c(bottom, left, top, right)` in inches

pin plot dimensions (width, height) in inches

See help(par) for more information on graphical parameters.

Example 1: Scatter by ggplot2

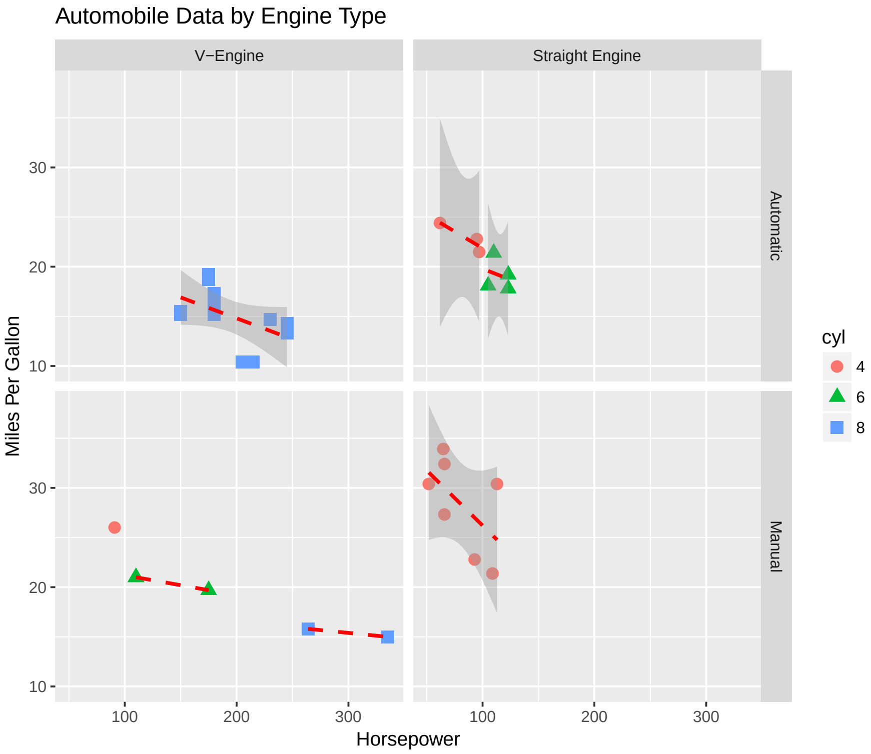
```
> library(ggplot2)
> ggplot(data=mtcars,aes(x=wt, y=mpg))+
+  geom_point(pch=17,color="blue",size=2)+
+  geom_smooth(method="lm",color="red",linetype=2)+
+  labs(title="Automobile Data",x="Weight",y="Miles Per Gallon")
```



Example 2: Scatter plot with faceting and grouping

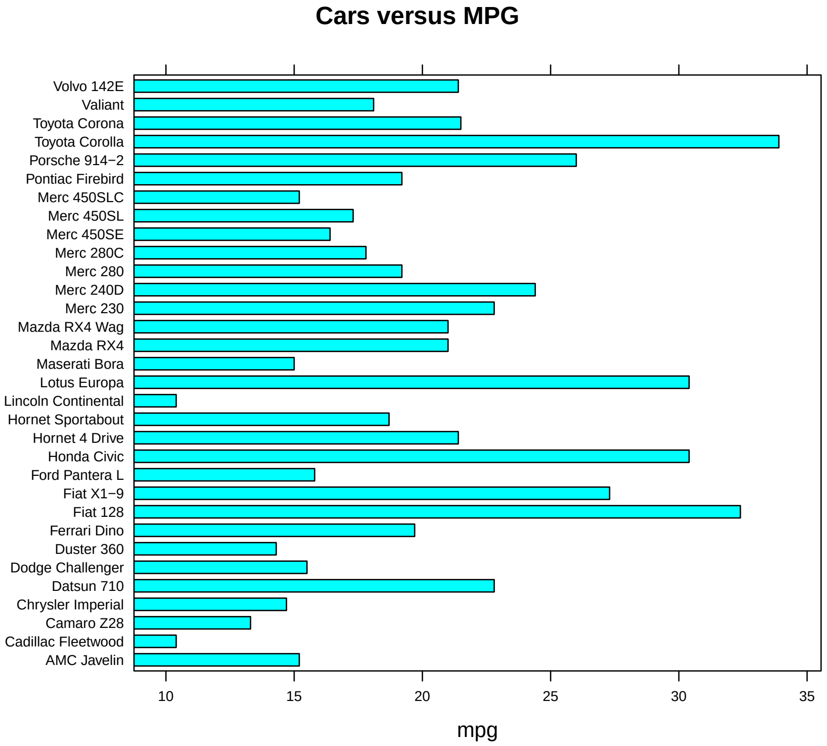
```
data(mtcars)
mtcars$am<-factor(mtcars$am, levels=c(0,1),
                  labels=c("Automatic", "Manual"))
mtcars$vs<-factor(mtcars$vs, levels=c(0,1),
                  labels=c("V-Engine", "Straight Engine"))
mtcars$cyl<-factor(mtcars$cyl)
library(ggplot2)
ggplot(data=mtcars, aes(x=hp, y=mpg,
                       shape=cyl, color=cyl)) +
       geom_point(size=3) +
       facet_grid(am~vs) +
       geom_smooth(method="lm", color="red", linetype=2) +
       labs(title="Automobile Data by Engine Type",
       x="Horsepower", y="Miles Per Gallon")
```

Example 2: Scatter plot with faceting and grouping

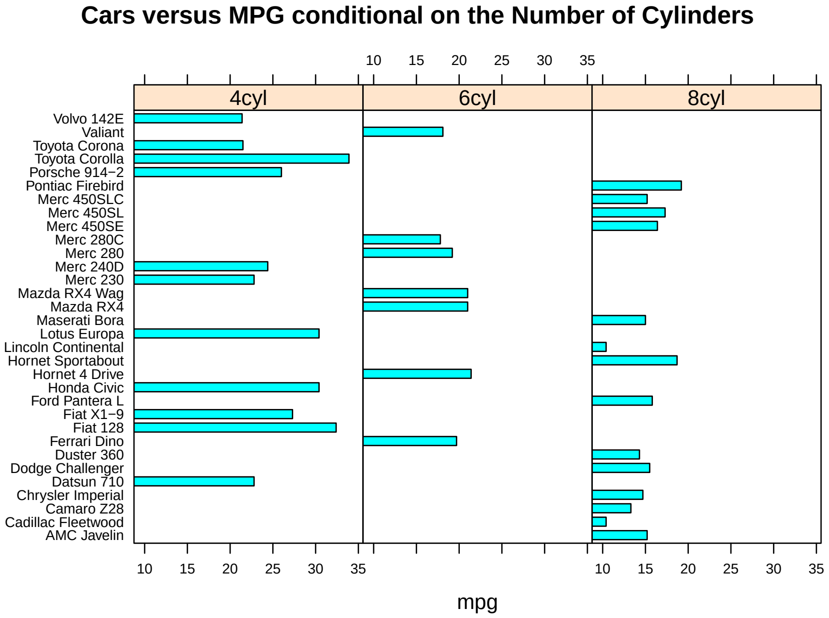


Example 3: Using lattice to plot mtcars data

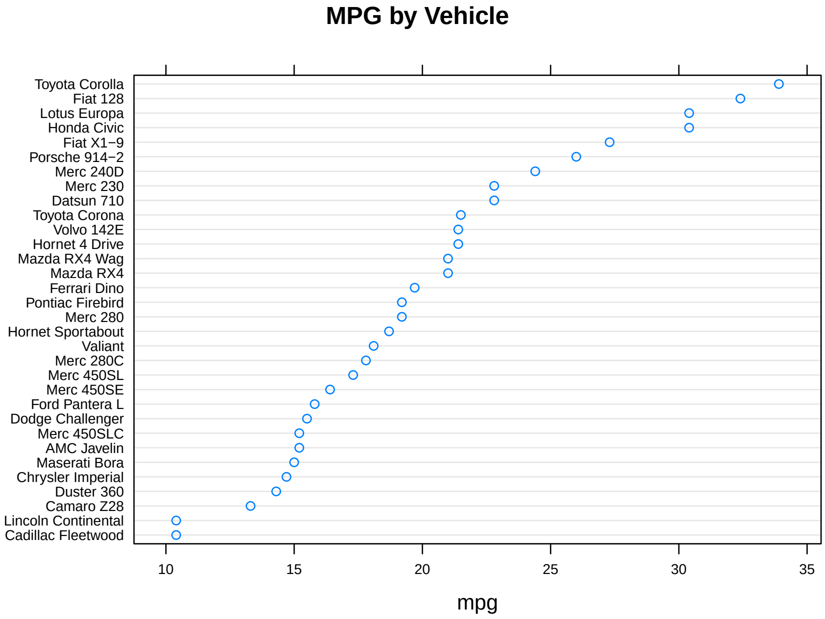
```
# Prepare variables
mtcars$cars<-rownames(mtcars)
mtcars$gear.f<-factor(mtcars$gear,levels=c(3,4,5),
  labels=c("3gears","4gears","5gears"))
mtcars$cyl.f<-factor(mtcars$cyl,levels=c(4,6,8),
     labels=c("4cyl","6cyl","8cyl"))
library("lattice")
# barchart
barchart(cars~mpg,
     data=mtcars,
     scales=list(cex=0.7),#shrink the axis text a little bit
     main="Cars versus MPG")
# barchart conditional on cyl
barchart(cars~mpg|cyl.f,data=mtcars,scales=list(cex=0.7),
main="Cars versus MPG conditional on the Cyl")
```



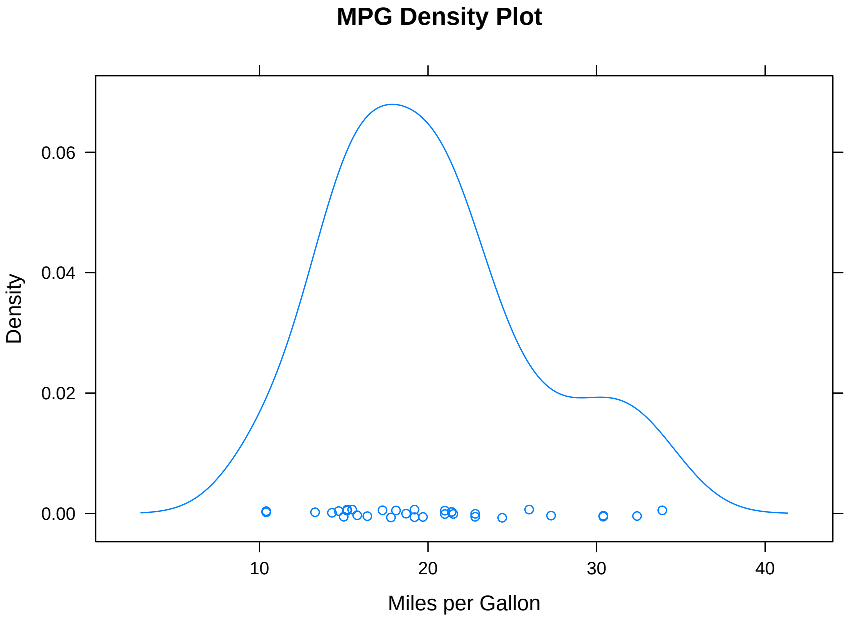
```
> barchart(cars~mpg | cyl.f,data=mtcars,scales=list(cex=0.7),
+   main="Cars versus MPG conditional on the Number of Cylinders")
```



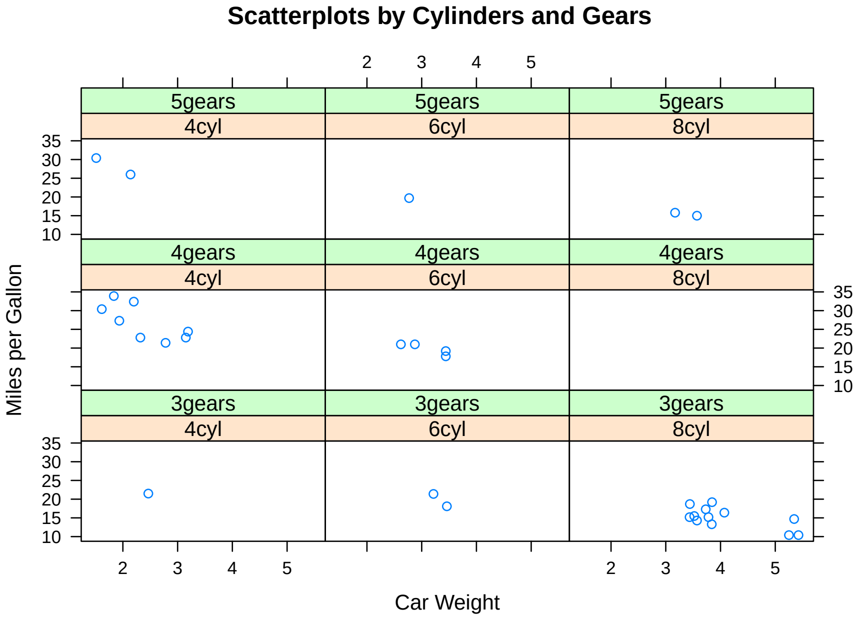
```
> dotplot(reorder(cars,mpg)~mpg,data=mtcars,pch=1,
+   scales=list(cex=0.7),main="MPG by Vehicle")
```



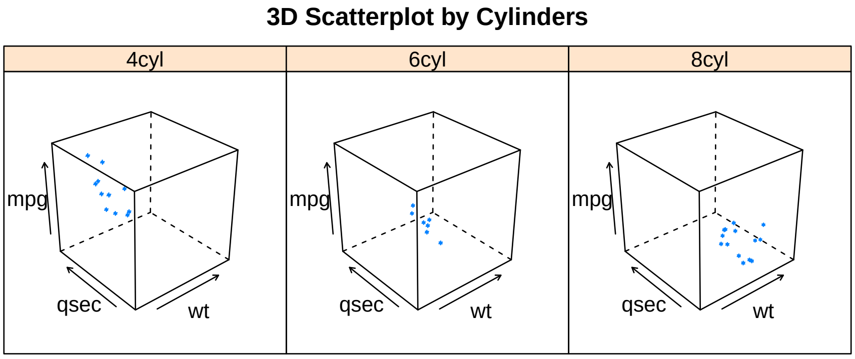
```
> densityplot(~mpg,data=mtcars,main="MPG Density Plot",xlab="Miles per Gallon")
```



```
> xyplot(mpg~wt|cyl.f*gear.f,data=mtcars,ylab="Miles per Gallon",
+ xlab="Car Weight", main="Scatterplots by Cylinders and Gears")
```



```
> cloud(mpg~wt*qsec|cyl.f,data=mtcars,main="3D Scatterplot by Cylinders")
```


3D Scatterplot by Cylinders
4cyl
6cyl
8cyl
mpg
qsec
wt