Introduction to Statistical Modelling in R

BCAM - Basque Center for Applied Mathematics, Applied Statistics

Dae-Jin Lee < dlee@bcamath.org >

CHAPTER 2. Basic data analysis and plotting in R

Contents

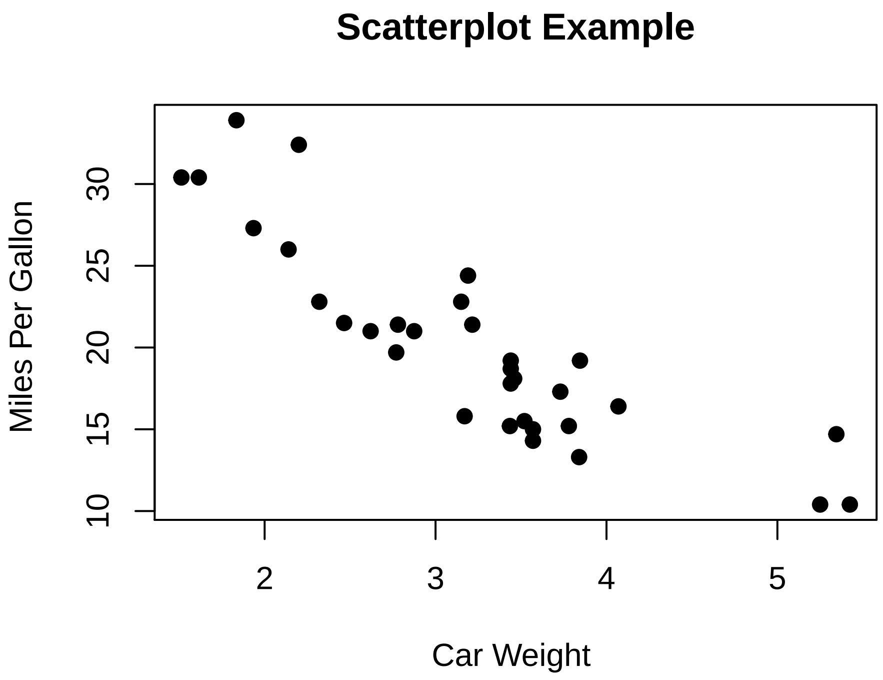
---

1 Basic data analysis in R

1.1 Basic plotting

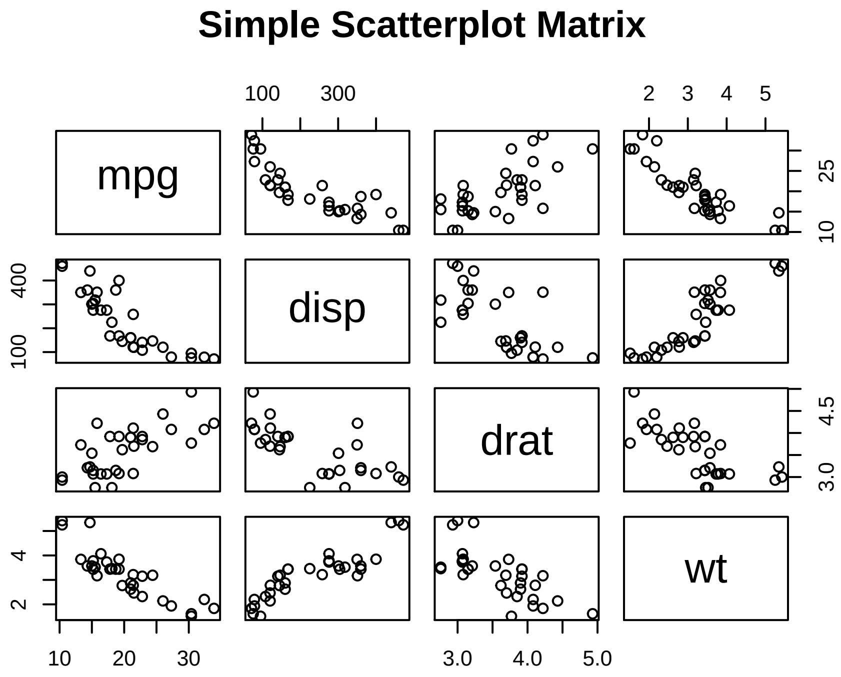
- Scatterplot

```r
attach(mtcars)
plot(wt, mpg, main="Scatterplot Example",
   xlab="Car Weight ", ylab="Miles Per Gallon ", pch=19)
```



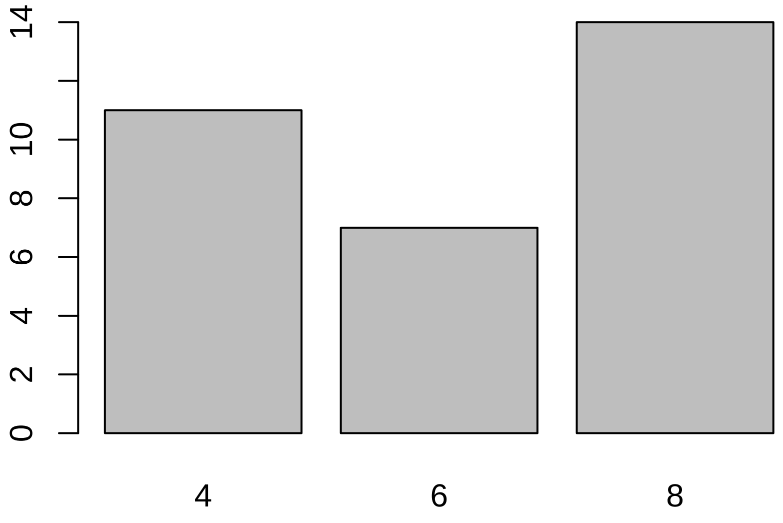
- Basic Scatterplot Matrix

```r
pairs(~mpg+disp+drat+wt,data=mtcars,
   main="Simple Scatterplot Matrix")
```



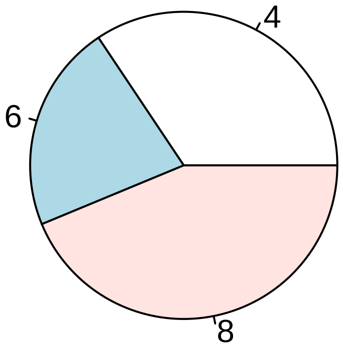
- Barplot

```r
tab <- table(mtcars[,c("cyl")])
barplot(tab)
```



- Piechart

```r
pie(tab)
```



Exercises:

1. The data.frame `VADeaths` contains the death rates per 1000 in Virginia (US) in 1940

- The death rates are measured per 1000 population per year. They are cross-classified by age group (rows) and population group (columns). The age groups are: 50-54, 55-59, 60-64, 65-69, 70-74 and the population groups are Rural/Male, Rural/Female, Urban/Male and Urban/Female.

```r
data(VADeaths)
VADeaths
```

```
##       Rural Male Rural Female Urban Male Urban Female
## 50-54       11.7          8.7       15.4          8.4
## 55-59       18.1         11.7       24.3         13.6
## 60-64       26.9         20.3       37.0         19.3
## 65-69       41.0         30.9       54.6         35.1
## 70-74       66.0         54.3       71.1         50.0
```

- Compute the mean for each age group.
  - **Result:**

```
##  50-54  55-59  60-64  65-69  70-74
## 11.050 16.925 25.875 40.400 60.350
```

- Compute the mean for each population group.

  - **Result:**

```
##   Rural Male Rural Female   Urban Male Urban Female
##        32.74        25.18        40.48        25.28
```

2. The `data.frame` `rainforest` contains several variables from different `species`

```
library(DAAG)
```

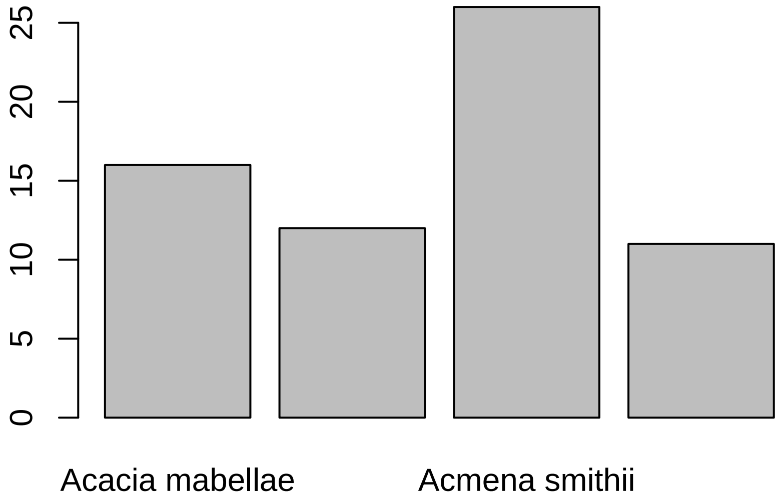
```
## Loading required package: lattice
```

```
rainforest
?rainforest
names(rainforest)
```

- Create a table of counts for each `species` and make a graphic with the results.

  - **Result:**

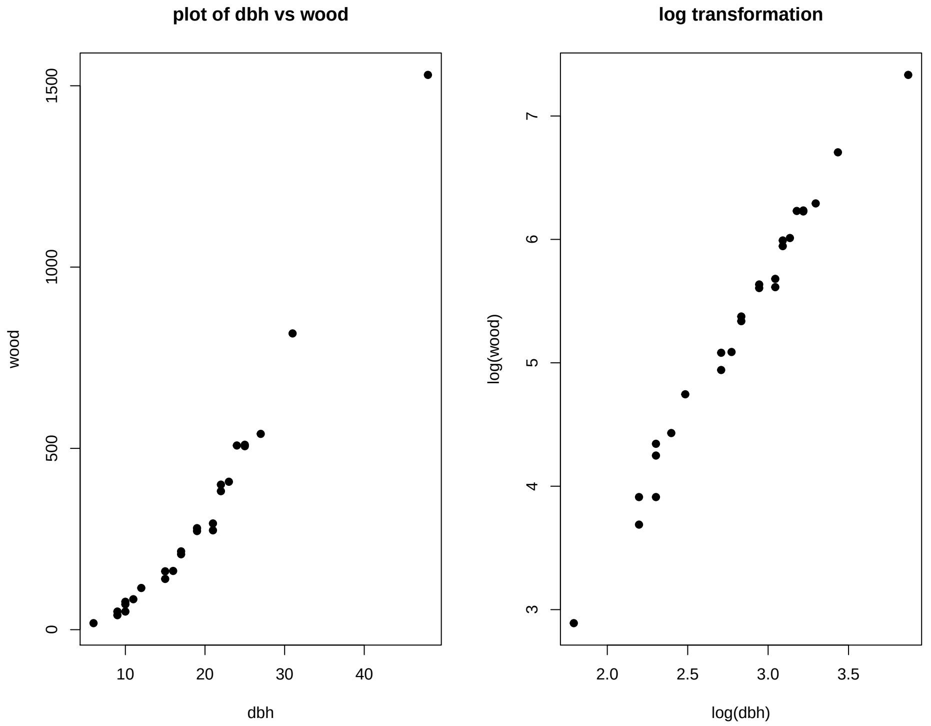
```
##
## Acacia mabellae      C. fraseri   Acmena smithii    B. myrtifolia
##              16              12               26               11
```



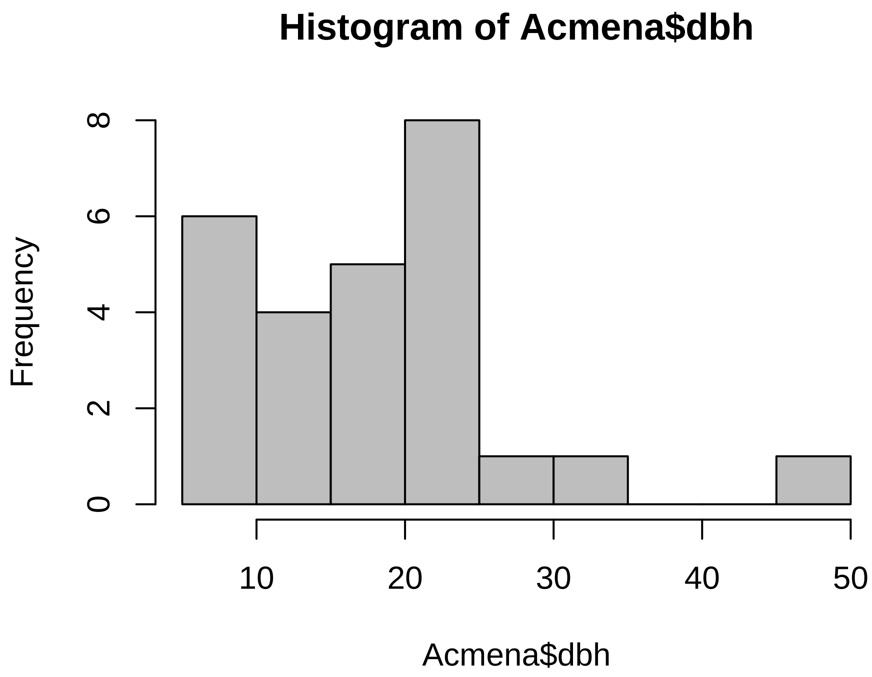
3. The `Acmena` `data.frame` is created from `rainforest` using the function `subset`.

- Plot the relationship between the wood biomass (`wood`) and the diameter of the breast height (`dbh`). Use also a logarithm scale.

```r
Acmena <- subset(rainforest, species == "Acmena smithii")
```



- Compute a histogram of variable `dbh` using function `hist`



4. Create a vector of the positive odd integers less than 100 and remove the values greater than 60 and less than 80.

    - **Result:**

```
##  [1] 61 63 65 67 69 71 73 75 77 79
```

- Solutions here

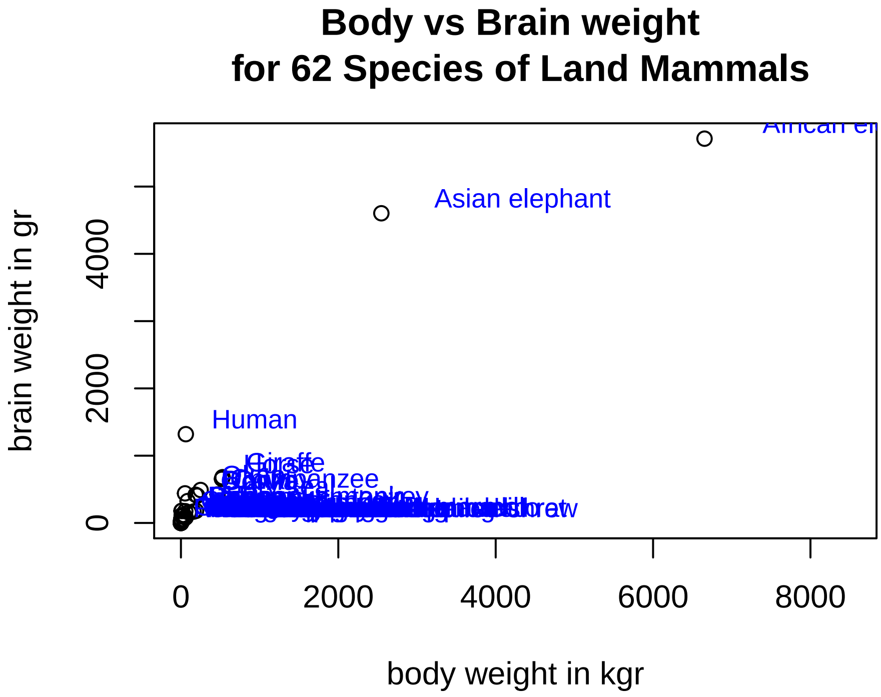
1.2 Scatterplots

```
library(MASS)
data("mammals")
?mammals
head(mammals)
```

```
##                      body brain
## Arctic fox          3.385  44.5
## Owl monkey         0.480  15.5
## Mountain beaver    1.350   8.1
## Cow              465.000 423.0
## Grey wolf         36.330 119.5
## Goat              27.660 115.0
```

```r
attach(mammals)
species <- row.names(mammals)
x <- body
y <- brain
```

```r
library(calibrate)
# scatterplot
plot(x,y, xlab = "body weight in kgr", ylab = "brain weight in gr",
     main="Body vs Brain weight \n for 62 Species of Land Mammals",xlim=c(0,8500))
textxy(x,y,labs=species,col = "blue",cex=0.85)
```

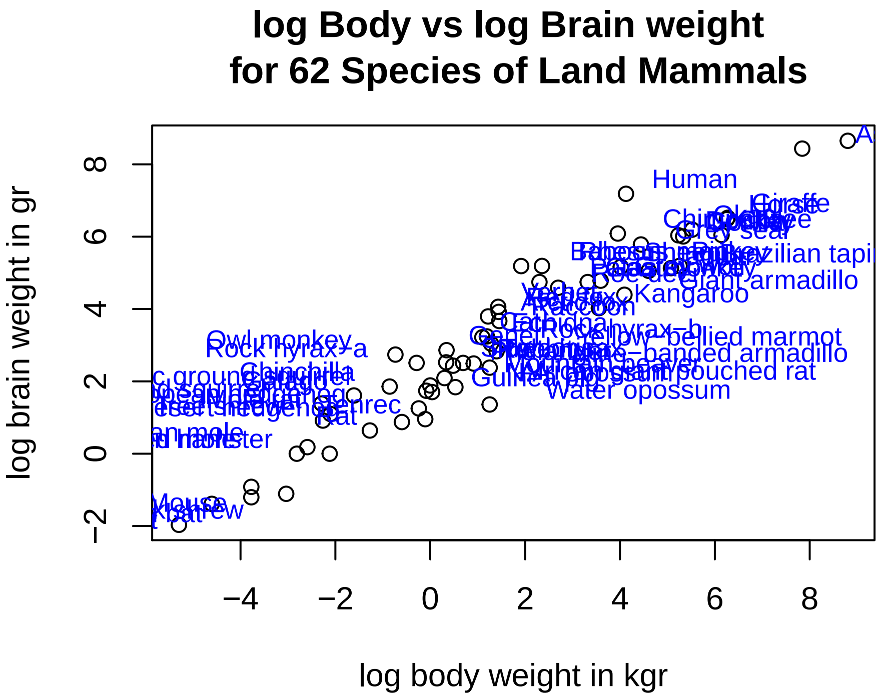


Identify a point in the scatterplot

```r
identify(x,y,species)
```

Plot in the log scale

```r
plot(log(x),log(y), xlab = "log body weight in kgr", ylab = "log brain weight in gr",
     main="log Body vs log Brain weight \n for 62 Species of Land Mammals")
textxy(log(x),log(y),labs=species,col = "blue",cex=0.85)
```



Identify a point in the log scale scatterplot

```r
identify(log(x),log(y),species)
```

1.3 More plotting options

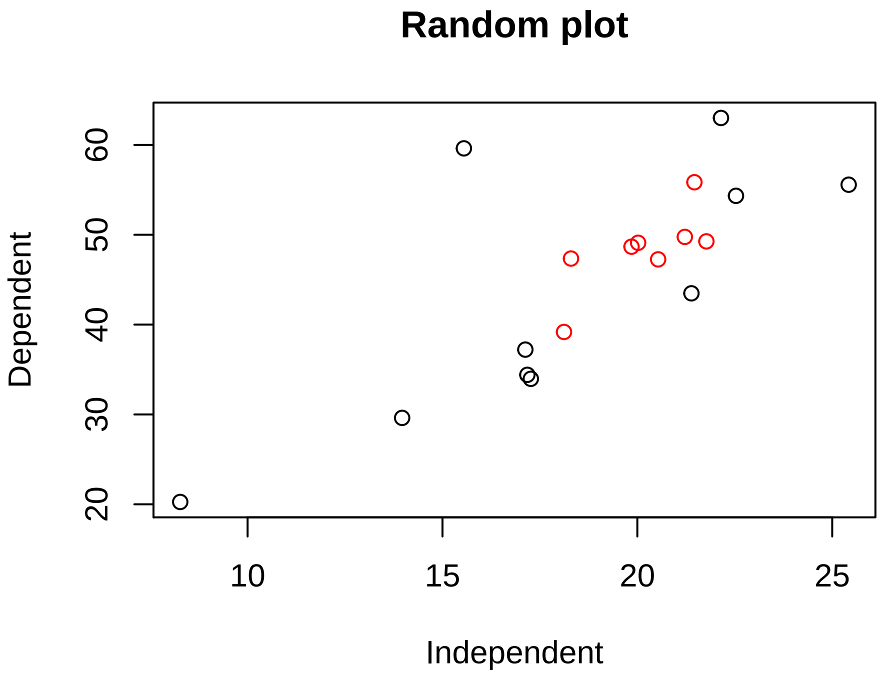
Multiple Data Sets on One Plot

One common task is to plot multiple data sets on the same plot. In many situations, the way to do this is to create the initial plot and then add additional information to the plot. For example, to plot bivariate data the `plot` command is used to initialise and create the plot. The `points` command can then be used to add additional datasets to the plot.

```r
set.seed(1234)
 x <- rnorm(10,sd=5,mean=20)
 y <- 2.5*x - 1.0 + rnorm(10,sd=9,mean=0)
 cor(x,y)
```

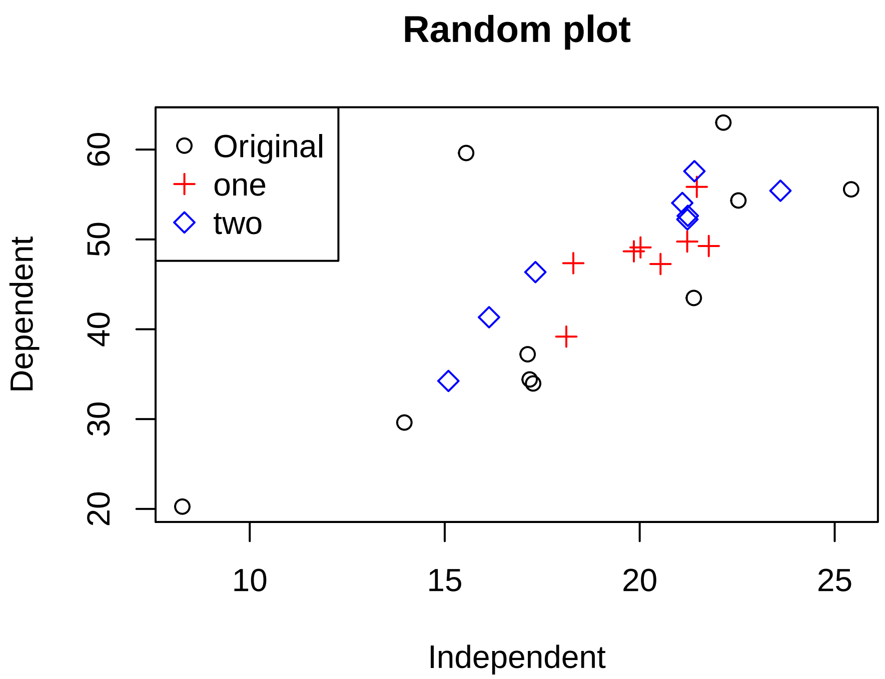
```
## [1] 0.7512194
```

```
plot(x,y,xlab="Independent",ylab="Dependent",main="Random plot")
x1 <- runif(8,15,25)
y1 <- 2.5*x1 - 1.0 + runif(8,-6,6)
points(x1,y1,col=2)
```



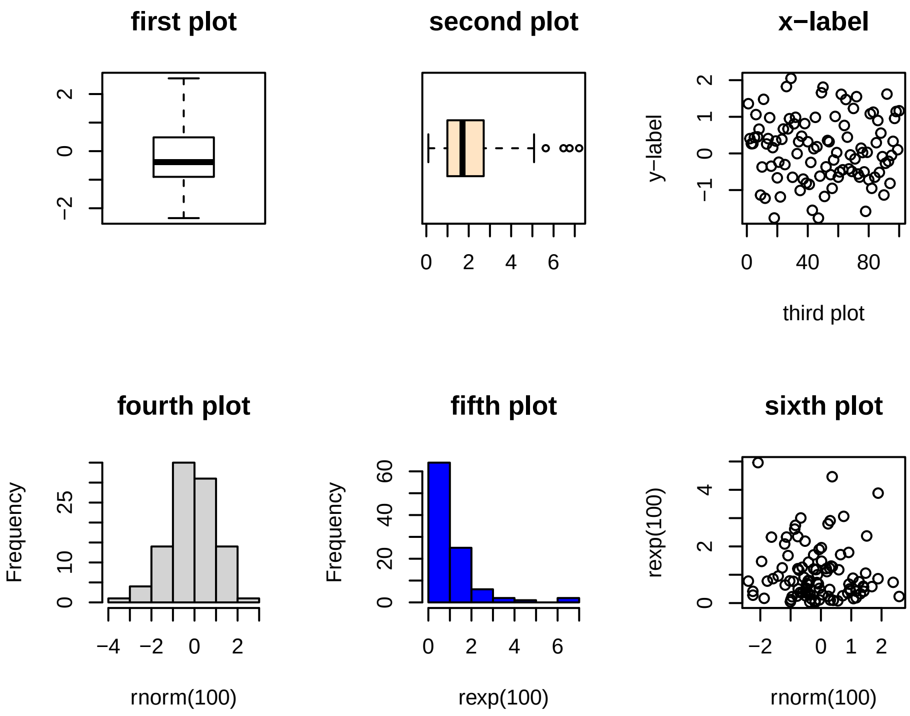
with legend and (x_2, y_2) points:

```
set.seed(1234)
x2 <- runif(8,15,25)
y2 <- 2.5*x2 - 1.0 + runif(8,-6,6)
 plot(x,y,xlab="Independent",ylab="Dependent",main="Random plot")
 points(x1,y1,col=2,pch=3)
 points(x2,y2,col=4,pch=5)
 legend("topleft",c("Original","one","two"),col=c(1,2,4),pch=c(1,3,5))
```



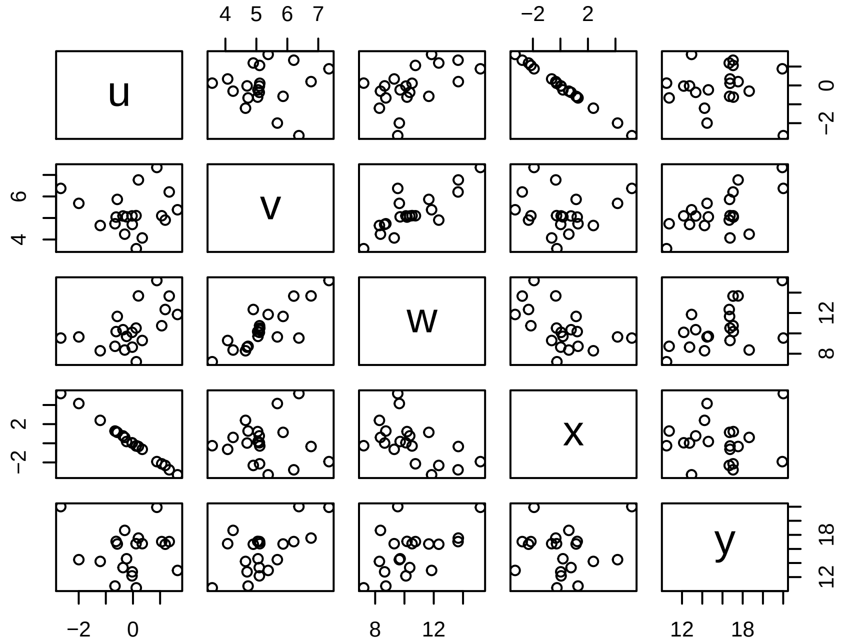
Multiple Graphs on One Image:

```
set.seed(1234)
 par(mfrow=c(2,3))
 boxplot(rnorm(100),main="first plot")
 boxplot(rgamma(100,2),main="second plot", horizontal=TRUE,col="bisque")
 plot(rnorm(100),xlab="third plot",
      ylab="y-label",main="x-label")
 hist(rnorm(100),main="fourth plot",col="lightgrey")
 hist(rexp(100),main="fifth plot",col="blue")
 plot(rnorm(100),rexp(100),main="sixth plot")
```



Pairwise relationships

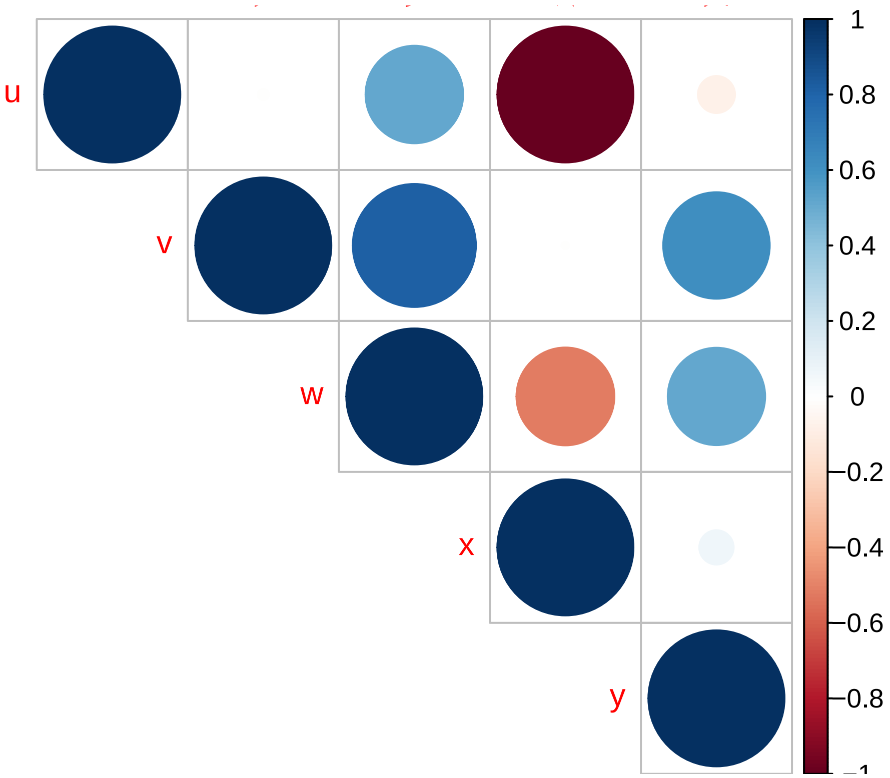
```r
uData <- rnorm(20)
vData <- rnorm(20,mean=5)
wData <- uData + 2*vData + rnorm(20,sd=0.5)
xData <- -2*uData+rnorm(20,sd=0.1)
yData <-  3*vData+rnorm(20,sd=2.5)
d <- data.frame(u=uData,v=vData,w=wData,x=xData,y=yData)
pairs(d)
```



Plotting correlations

The function `corrplot` in the `library(corrplot)` visualizes a correlation matrix calculate with function `cor`

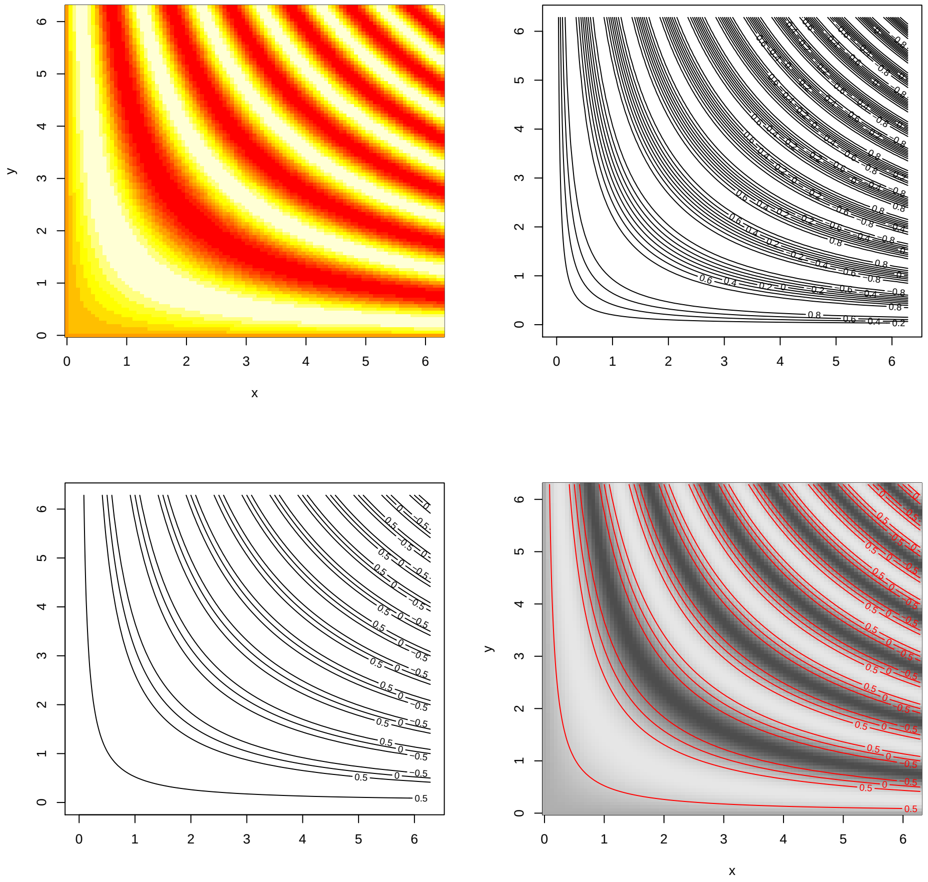
```
library(corrplot)
M <- cor(d)
corrplot(M, method="circle",type="upper")
```



Plotting surfaces: `image`, `contour` and `persp` plots

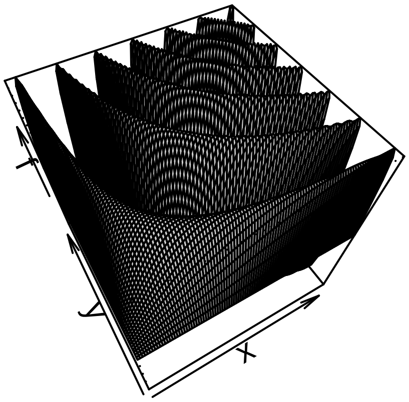
```
x <- seq(0,2*pi,by=pi/50)
y <- x
xg <- (x*0+1) %*% t(y)
yg <- (x) %*% t(y*0+1)
f <- sin(xg*yg)

par(mfrow=c(2,2))
image(x,y,f)
contour(x,y,f)
contour(x,y,f,nlevels=4)
image(x,y,f,col=grey.colors(100))
contour(x,y,f,nlevels=4,add=TRUE,col="red")
```



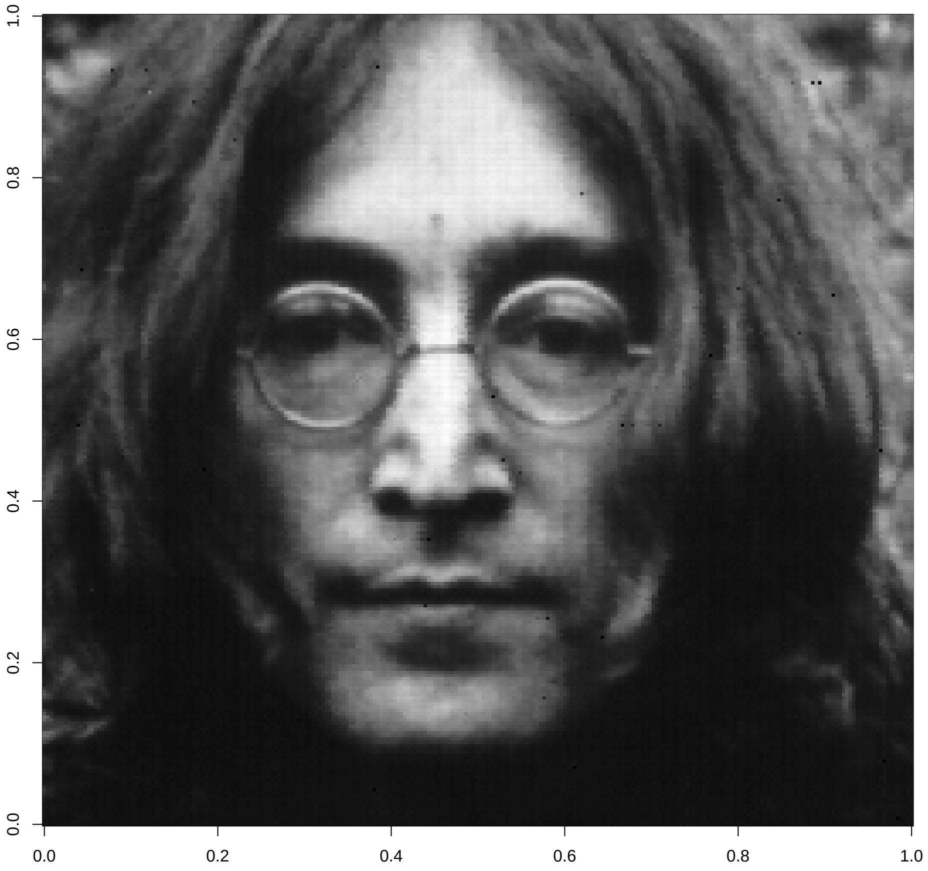
Similarly, one can use `persp` plot

```
persp(x,y,f,theta=-30,phi=55,col="lightgrey",shade=.01)
```



Or plot images

```
library(fields)
data(lennon)
image(lennon,col=grey(seq(0,1,l=256)))
```



1.4 Tables and Cross-classification

```
library(MASS)
data(quine)
?quine
attach(quine)
table(Sex)
```

```
## Sex
##  F  M
## 80 66
```

```r
table(Sex,Age)
```

```
##    Age
## Sex F0 F1 F2 F3
##   F 10 32 19 19
##   M 17 14 21 14
```

```r
# or xtabs
xtabs(~Sex+Age,data=quine)
```

```
##    Age
## Sex F0 F1 F2 F3
##   F 10 32 19 19
##   M 17 14 21 14
```

```r
xtabs(~Sex+Age+Eth,data=quine)
```

```
## , , Eth = A
## 
##    Age
## Sex F0 F1 F2 F3
##   F  5 15  9  9
##   M  8  5 11  7
## 
## , , Eth = N
## 
##    Age
## Sex F0 F1 F2 F3
##   F  5 17 10 10
##   M  9  9 10  7
```

1.5 Calculation of cross-classifications

```r
tapply(Days,Age,mean)
```

```
##       F0       F1       F2       F3 
## 14.85185 11.15217 21.05000 19.60606
```

```r
tapply(Days,list(Sex,Age),mean)
```

```
##          F0       F1       F2       F3
## F 18.70000 12.96875 18.42105 14.00000
## M 12.58824  7.00000 23.42857 27.21429
```

```
tapply(Days,list(Sex,Age),function(x) sqrt(var(x)/length(x)))
```

```
##          F0       F1       F2       F3
## F 4.208589 2.329892 5.299959 2.940939
## M 3.768151 1.418093 3.766122 4.569582
```

1.6 Qualitative data

A data sample is called qualitative, also known as categorical if its values belong to a collection of known defined non-overlapping classes. Common examples include student letter grade (A, B, C, D or F), commercial bond rating (AAA, AAB, . . .) and consumer clothing shoe sizes (1, 2, 3, . . .).

Let us consider some artificial data consisting of the `treatment` and `improvement` of patients with rheumatoid arthritis.

```
treatment <- factor(rep(c(1, 2), c(43, 41)), levels = c(1, 2),
                    labels = c("placebo", "treated"))
improved <- factor(rep(c(1, 2, 3, 1, 2, 3), c(29, 7, 7, 13, 7, 21)),
                   levels = c(1, 2, 3),
                   labels = c("none", "some", "marked"))
```

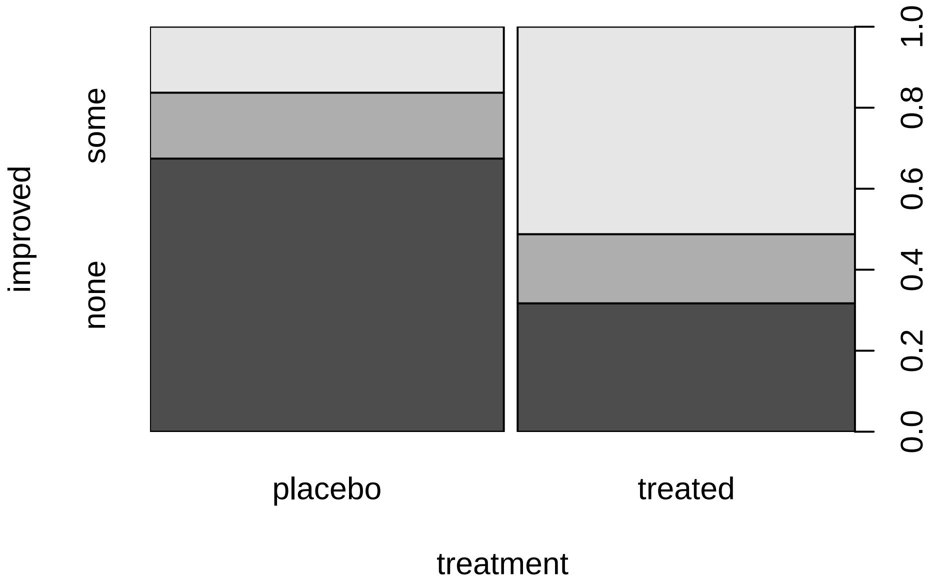
We can compute a cross-classification table

```
xtabs(~treatment+improved)
```

```
##          improved
## treatment none some marked
##   placebo   29    7      7
##   treated   13    7     21
```

Graphically,

```
spineplot(improved ~ treatment)
```



The R dataset UCBAdmissions contains aggregated data on applicants to graduate school at Berkeley for the six largest departments in 1973 classified by admission and sex.

```
data("UCBAdmissions")
?UCBAdmissions
apply(UCBAdmissions, c(2,1), sum)
```

```
##         Admit
## Gender   Admitted Rejected
##   Male       1198     1493
##   Female      557     1278
```

```
prop.table(apply(UCBAdmissions, c(2,1), sum))
```

```
##         Admit
## Gender    Admitted  Rejected
##   Male   0.2646929 0.3298719
##   Female 0.1230667 0.2823685
```

```
ftable(UCBAdmissions)
```

```
##                 Dept   A   B   C   D   E   F
## Admit    Gender
## Admitted Male        512 353 120 138  53  22
##          Female       89  17 202 131  94  24
## Rejected Male        313 207 205 279 138 351
##          Female       19   8 391 244 299 317
```

The same but with a more readable format can be obtained using `ftable`

```
ftable(round(prop.table(UCBAdmissions), 3),
       row.vars="Dept", col.vars = c("Gender", "Admit"))
```

```
##      Gender     Male              Female
##      Admit  Admitted Rejected Admitted Rejected
## Dept
## A              0.113    0.069    0.020    0.004
## B              0.078    0.046    0.004    0.002
## C              0.027    0.045    0.045    0.086
## D              0.030    0.062    0.029    0.054
## E              0.012    0.030    0.021    0.066
## F              0.005    0.078    0.005    0.070
```

More interesting are the proportions admitted for each `Gender` by `Dept` combination (dimensions 2 and 3 of the array). Notice that `male` and `female` admission rates are about the same in all departments, except "A", where female admission rates are higher.

```
# prop.table(UCBAdmissions, c(2,3))
ftable(round(prop.table(UCBAdmissions, c(2,3)), 2),
       row.vars="Dept", col.vars = c("Gender", "Admit"))
```

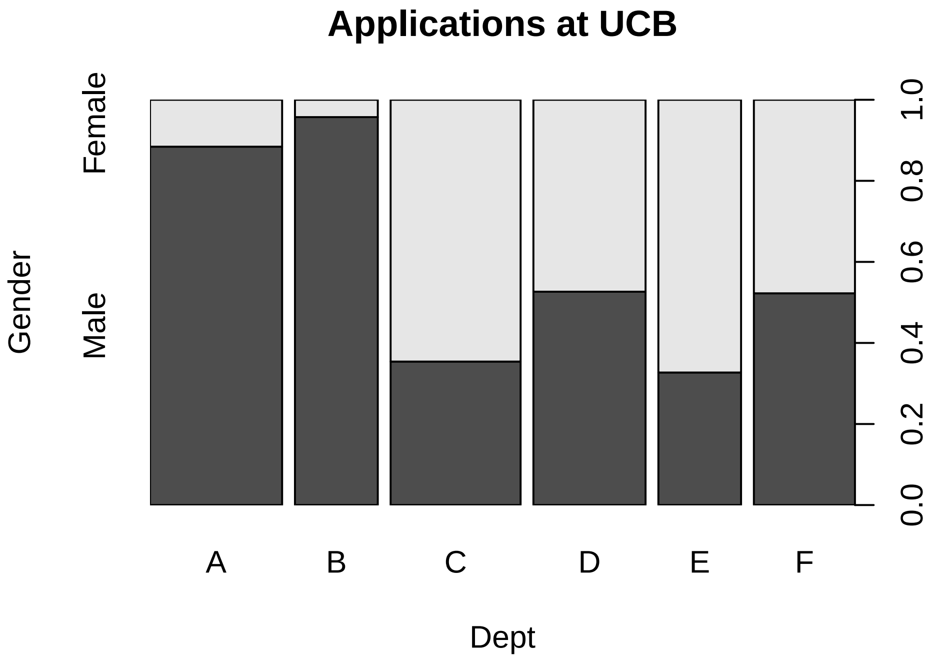
```
##      Gender     Male              Female
##      Admit  Admitted Rejected Admitted Rejected
## Dept
## A               0.62     0.38     0.82     0.18
## B               0.63     0.37     0.68     0.32
## C               0.37     0.63     0.34     0.66
## D               0.33     0.67     0.35     0.65
## E               0.28     0.72     0.24     0.76
## F               0.06     0.94     0.07     0.93
```

```
## Data aggregated over departments
apply(UCBAdmissions, c(1, 2), sum)
```

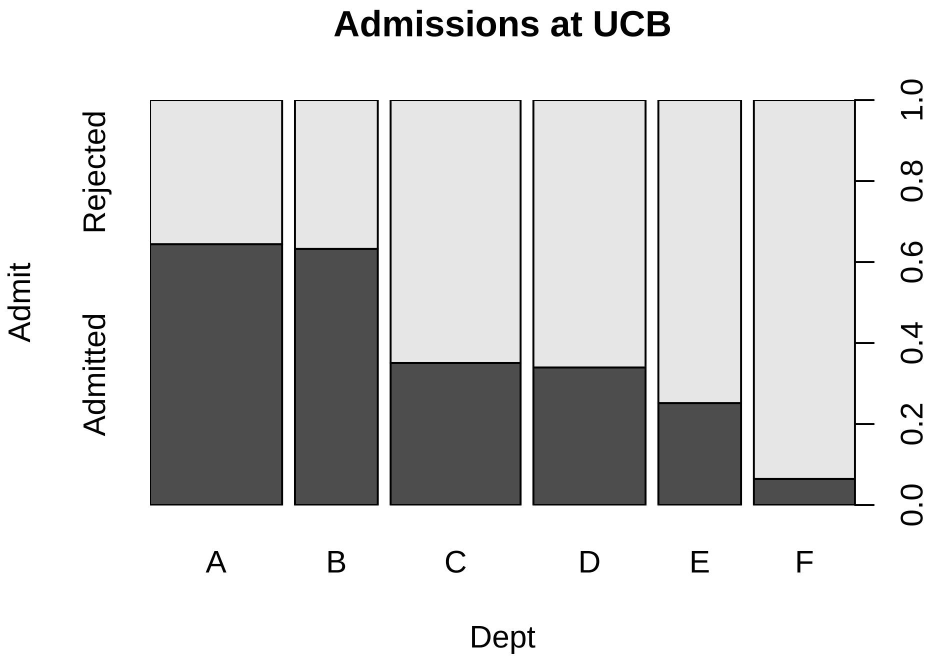
```
##           Gender
## Admit       Male Female
##   Admitted 1198    557
##   Rejected 1493   1278
```

Applications and admissions by department at UC Berkeley can be viewed graphically.

```r
spineplot(margin.table(UCBAdmissions, c(3, 2)),
          main = "Applications at UCB")
```



```r
spineplot(margin.table(UCBAdmissions, c(3, 1)),
          main = "Admissions at UCB")
```



This data set is frequently used for illustrating *Simpson's paradox*. At issue is whether the data show evidence of sex bias in admission practices. There were 2691 male applicants, of whom 1198 (44.5%) were admitted, compared with 1835 female applicants of whom 557 (30.4%) were admitted. Men were much more successful in admissions than women. Wikipedia: Gender Bias UC Berkeley. See animation at link

1.7 Quantitative data

Quantitative data, also known as continuous data, consists of numeric data that support arithmetic operations. This is in contrast with qualitative data, whose values belong to pre-defined classes with no arithmetic operation allowed. We will explain how to apply some of the R tools for quantitative data analysis with examples.

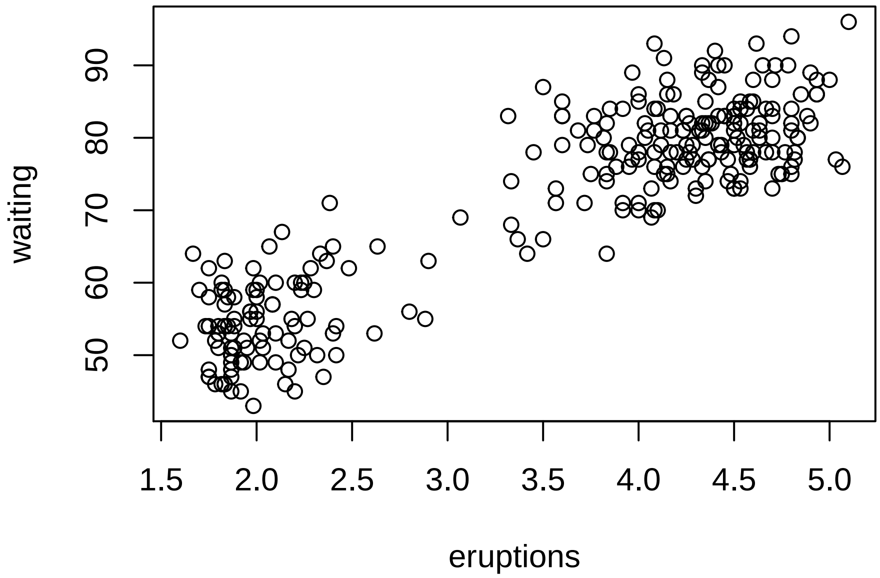
```
head(faithful)
```

```
##   eruptions waiting
## 1     3.600      79
## 2     1.800      54
## 3     3.333      74
## 4     2.283      62
## 5     4.533      85
## 6     2.883      55
```

It consists of a collection of observations of the Old Faithful geyser in the USA Yellowstone National Park.

There are two observation variables in the dataset. The first one, called `eruptions`, is the duration of the geyser eruptions. The second one, called `waiting`, is the length of waiting period until the next eruption. It turns out there is a correlation between the two variables.

```
plot(faithful)
```



1.7.1 Frequency distribution of quantitative data

The frequency distribution of a data variable is a summary of the data occurrence in a collection of non-overlapping categories.

Let us find the frequency distribution of the eruption duration in `faithful` data set.

```
duration <- faithful$eruptions
range(duration)
```

```
## [1] 1.6 5.1
```

Now we create the range of non-overlapping sub-intervals by defining a sequence of equal distance break points. If we round the endpoints of the interval [1.6, 5.1] to the closest half-integers, we come up with the interval [1.5, 5.5]. Hence we set the breakpoints to be the half-integer sequence { 1.5, 2.0, 2.5, . . . }.

```
breaks <- seq(1.5,5.5,by=0.5)
breaks
```

```
## [1] 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0 5.5
```

Classify the eruption durations according to the half-unit-length sub-intervals with `cut`. As the intervals are to be closed on the left, and open on the right, we set the right argument to `FALSE`.

```
duration.cut = cut(duration, breaks, right=FALSE)
```

Compute the frequency of eruptions in each sub-interval with the table function.

```
duration.freq = table(duration.cut)
duration.freq
```

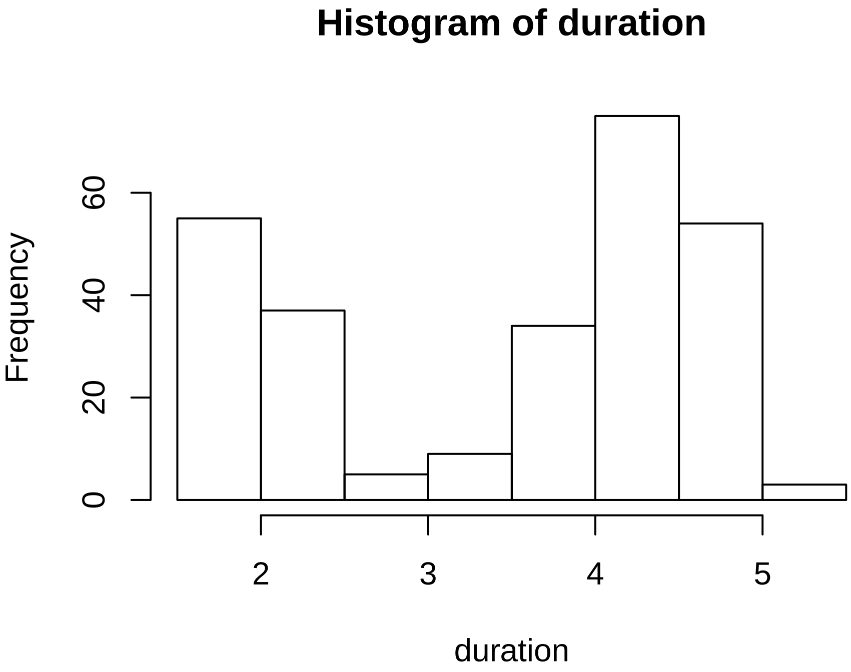
```
## duration.cut
## [1.5,2)  [2,2.5)  [2.5,3)  [3,3.5)  [3.5,4)  [4,4.5)  [4.5,5)  [5,5.5)
##      51       41        5        7       30       73       61        4
```

`hist` function does all the computaions to find the frequency distribution:

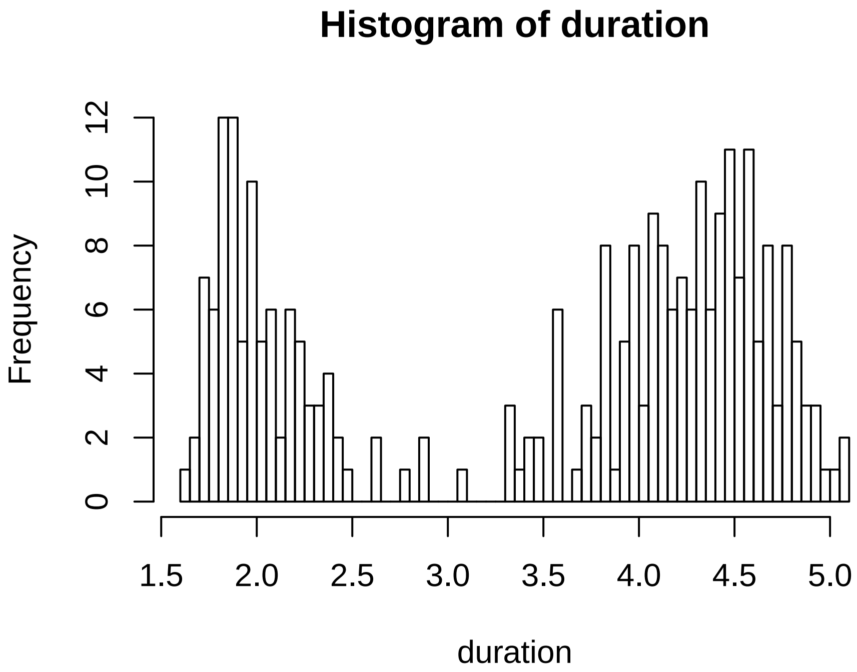
```
freq <- hist(duration)
freq
```

```
## $breaks
## [1] 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0 5.5
##
## $counts
## [1] 55 37  5  9 34 75 54  3
##
## $density
## [1] 0.40441176 0.27205882 0.03676471 0.06617647 0.25000000 0.55147059
## [7] 0.39705882 0.02205882
##
## $mids
## [1] 1.75 2.25 2.75 3.25 3.75 4.25 4.75 5.25
##
## $xname
## [1] "duration"
##
## $equidist
## [1] TRUE
##
## attr(,"class")
## [1] "histogram"
```

```
freq <- hist(duration,breaks = breaks)
```



```
hist(duration,50)
```

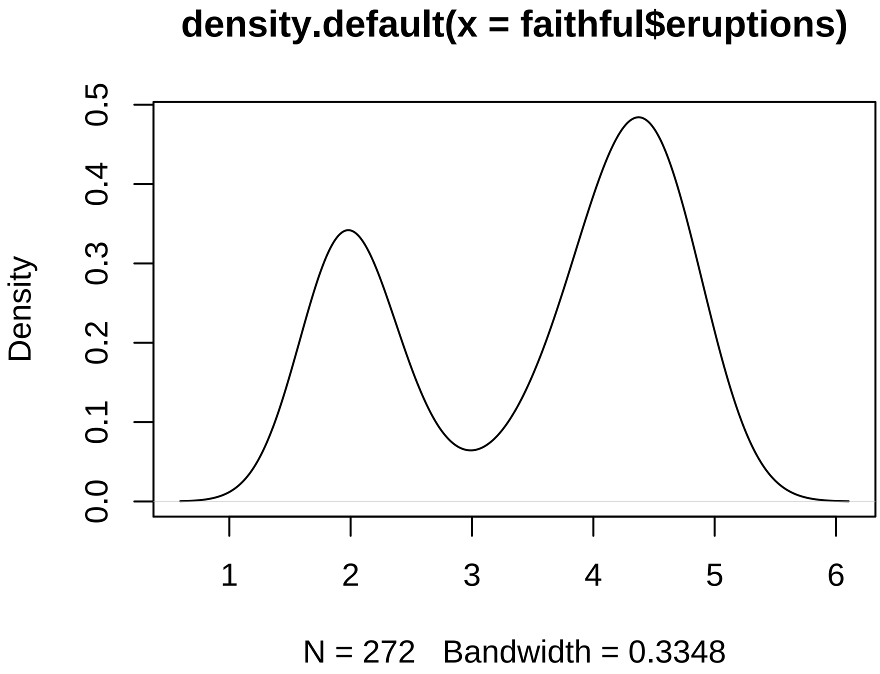


Density estimation builds an estimate of some underlying probability density function using an observed data sample.

```r
require(graphics)
d <- density(faithful$eruptions)
d
```

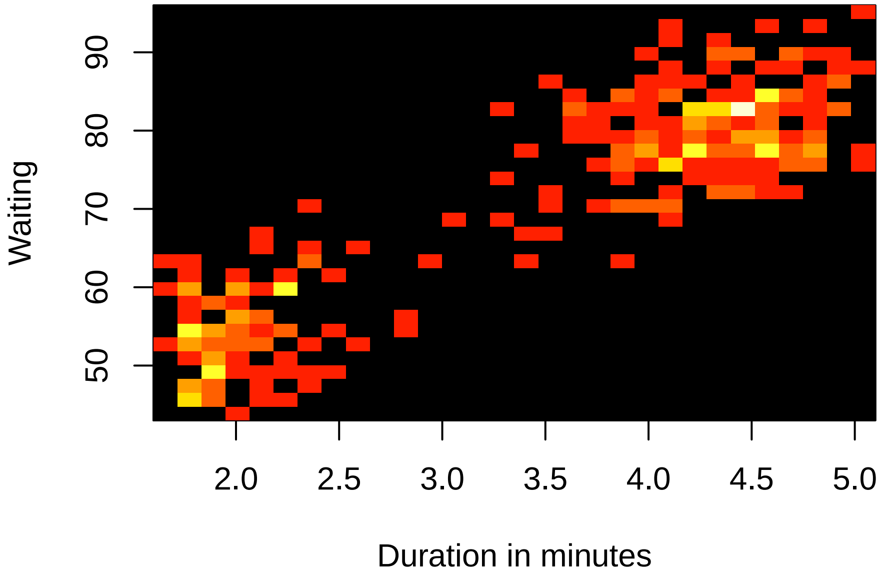
```
##
## Call:
##  density.default(x = faithful$eruptions)
##
## Data: faithful$eruptions (272 obs.); Bandwidth 'bw' = 0.3348
##
##        x                y
##  Min.   :0.5957   Min.   :0.0002262
##  1st Qu.:1.9728   1st Qu.:0.0514171
##  Median :3.3500   Median :0.1447010
##  Mean   :3.3500   Mean   :0.1813462
##  3rd Qu.:4.7272   3rd Qu.:0.3086071
##  Max.   :6.1043   Max.   :0.4842095
```

```r
plot(d)
```



Two dimension histogram:

```
library(gplots)
h2 <- hist2d(faithful, nbins=30,xlab="Duration in minutes",ylab="Waiting")
```



```
h2
```

```
##
## ----------------------------
## 2-D Histogram Object
## ----------------------------
##
## Call: hist2d(x = faithful, nbins = 30, xlab = "Duration in minutes",
##     ylab = "Waiting")
##
## Number of data points:  272
## Number of grid bins:  30 x 30
## X range: ( 1.6 , 5.1 )
## Y range: ( 43 , 96 )
```

```
names(h2)
```

```
## [1] "counts"   "x.breaks" "y.breaks" "x"        "y"        "nobs"
## [7] "call"
```

Relative frequencies

```
duration.relfreq <- duration.freq / nrow(faithful)
tab <- cbind(duration.freq, duration.relfreq)
apply(tab,2,sum)
```

```
##    duration.freq duration.relfreq
##              272                1
```

Cumulative frequency distribution

```
cumsum(duration.freq)
```

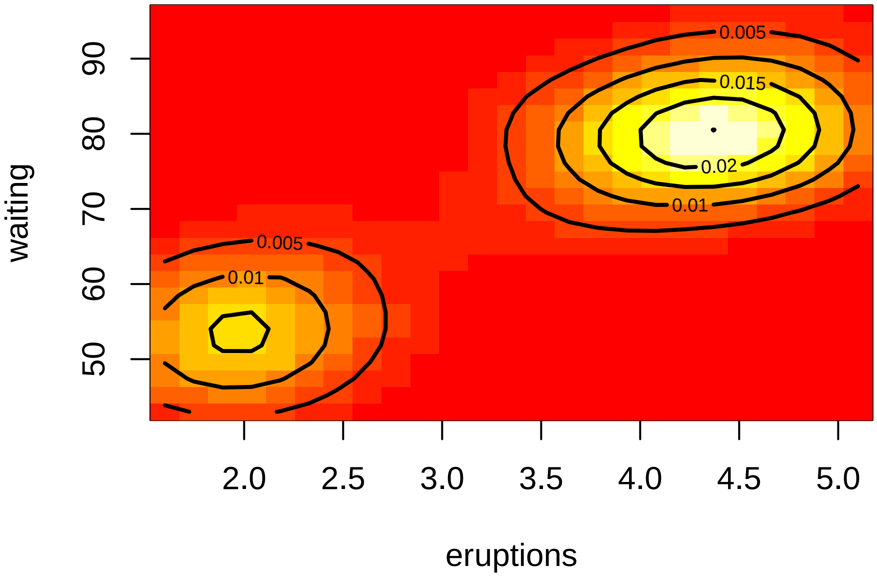
```
## [1.5,2) [2,2.5) [2.5,3) [3,3.5) [3.5,4) [4,4.5) [4.5,5) [5,5.5)
##      51      92      97     104     134     207     268     272
```

```
cumsum(duration.relfreq)
```

```
##   [1.5,2)   [2,2.5)   [2.5,3)   [3,3.5)   [3.5,4)   [4,4.5)   [4.5,5)
## 0.1875000 0.3382353 0.3566176 0.3823529 0.4926471 0.7610294 0.9852941
##   [5,5.5)
## 1.0000000
```

Bivariante Density estimation:

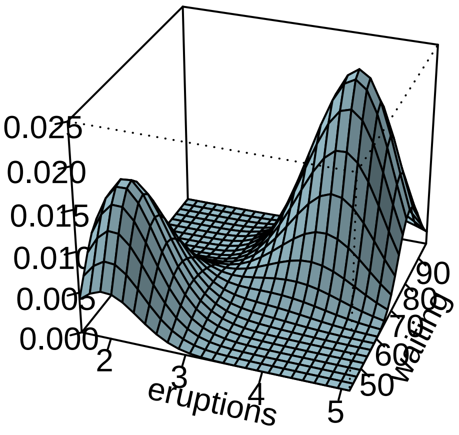
```
data("faithful")
attach(faithful)
Dens2d<-kde2d(eruptions,waiting)
image(Dens2d,xlab="eruptions",ylab="waiting")
contour(Dens2d,add=TRUE,col="black",lwd=2,nlevels=5)
```



```
detach("faithful")
```

Perspective plot:

```
persp(Dens2d,phi=30,theta=20,d=5,xlab="eruptions",ylab="waiting",zlab="",shade=.2,col="light
```



1.8 Advanced plotting

1.8.1 ggplot2

```
library(ggplot2)
```

1.9 Why ggplot2?

Advantages of ggplot2

- consistent underlying 'grammar of graphics' (Wilkinson, 2005)
- plot specification at a high level of abstraction
- very flexible
- theme system for polishing plot appearance
- mature and complete graphics system
- many users, active mailing list

Example: `Housing data`

```
housing <- read.csv("dataSets/landdata-states.csv")
head(housing[1:5])
```

```
##   State region    Date Home.Value Structure.Cost
## 1    AK   West 2010.25     224952         160599
## 2    AK   West 2010.50     225511         160252
## 3    AK   West 2009.75     225820         163791
## 4    AK   West 2010.00     224994         161787
## 5    AK   West 2008.00     234590         155400
## 6    AK   West 2008.25     233714         157458
```

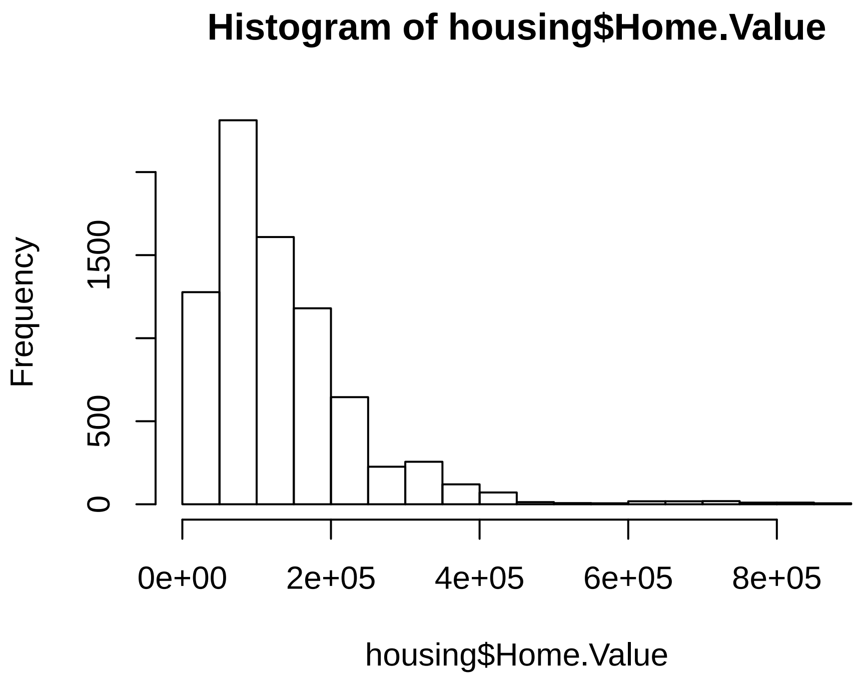
```
# change column names
names(housing)[names(housing) == "Land.Share..Pct."] <- "Land.Share.Pct"
head(housing, 10)
```

```
##    State region    Date Home.Value Structure.Cost Land.Value
## 1     AK   West 2010.25     224952         160599      64352
## 2     AK   West 2010.50     225511         160252      65259
## 3     AK   West 2009.75     225820         163791      62029
## 4     AK   West 2010.00     224994         161787      63207
## 5     AK   West 2008.00     234590         155400      79190
## 6     AK   West 2008.25     233714         157458      76256
## 7     AK   West 2008.50     232999         160092      72906
## 8     AK   West 2008.75     232164         162704      69460
## 9     AK   West 2009.00     231039         164739      66299
## 10    AK   West 2009.25     229395         165424      63971
##    Land.Share.Pct Home.Price.Index Land.Price.Index Year Qrtr
## 1            28.6            1.481            1.552 2010    1
## 2            28.9            1.484            1.576 2010    2
## 3            27.5            1.486            1.494 2009    3
## 4            28.1            1.481            1.524 2009    4
## 5            33.8            1.544            1.885 2007    4
## 6            32.6            1.538            1.817 2008    1
## 7            31.3            1.534            1.740 2008    2
## 8            29.9            1.528            1.660 2008    3
## 9            28.7            1.521            1.587 2008    4
## 10           27.9            1.510            1.536 2009    1
```

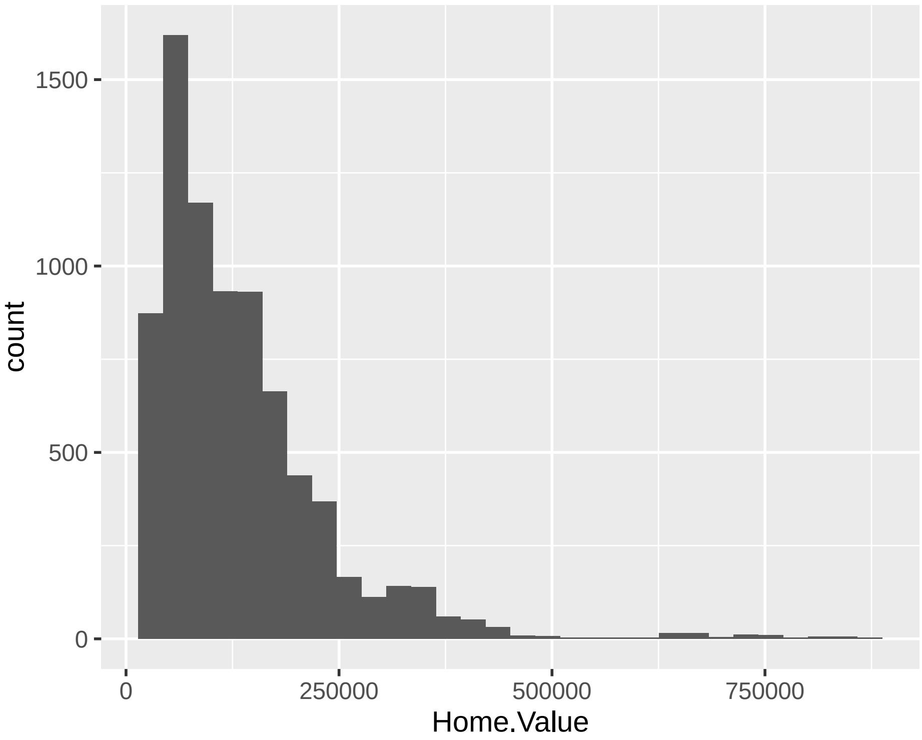
1.10 ggplot2 VS Base for simple graphs

Base graphics histogram are:

```
hist(housing$Home.Value)
```

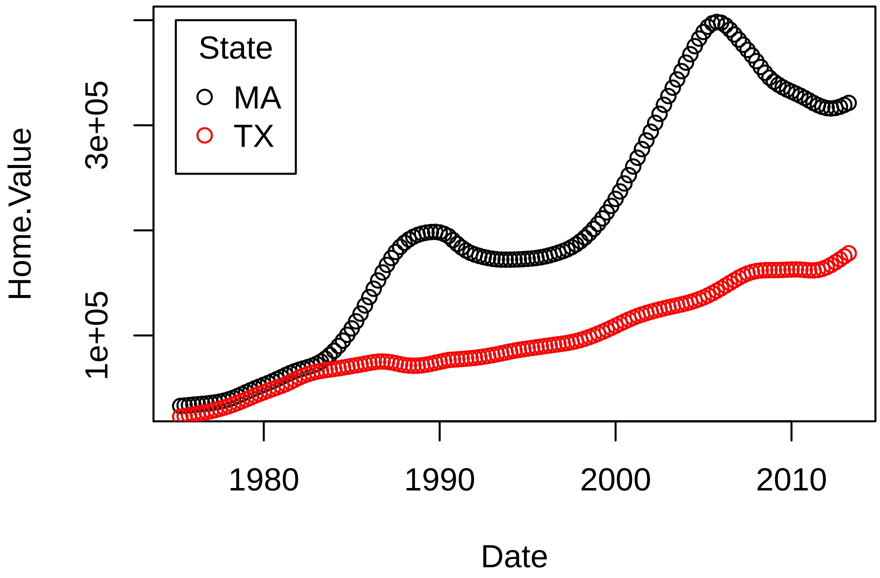


```r
library(ggplot2)
ggplot(housing, aes(x = Home.Value)) +
  geom_histogram()
```



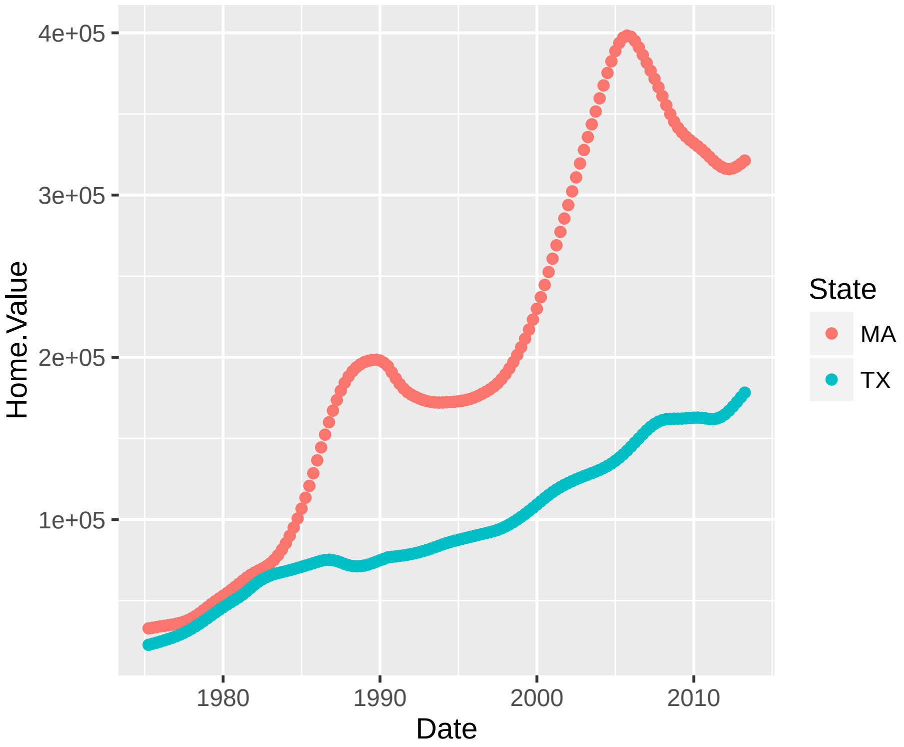
Another simple graph

```
plot(Home.Value ~ Date,
     data=subset(housing, State == "MA"))
points(Home.Value ~ Date, col="red",
       data=subset(housing, State == "TX"))
legend(1975, 400000,
       c("MA", "TX"), title="State",
       col=c("black", "red"),
       pch=c(1, 1))
```



ggplot version, colored scatter plot example:

```
ggplot(subset(housing, State %in% c("MA", "TX")),
       aes(x=Date,
           y=Home.Value,
           color=State))+
  geom_point()
```



1.10.1 Geometric Objects And Aesthetics

Aesthetic Mapping:

In `ggplot` land /aesthetic/ means “something you can see”. Examples include:

- position (i.e., on the x and y axes)
- color (“outside” color)
- fill (“inside” color)
- shape (of points)
- linetype
  - size

Each type of geom accepts only a subset of all aesthetics–refer to the geom help pages to see what mappings each geom accepts. Aesthetic mappings are set with the `aes()` function.

1.10.2 Geometic Objects (‘geom’)

Geometric objects are the actual marks we put on a plot. Examples include:

```
 - points (`geom_point`, for scatter plots, dot plots, etc)

 - lines (`geom_line`, for time series, trend lines, etc)

 - boxplot (`geom_boxplot`, for, well, boxplots!)

 - A plot must have at least one geom; there is no upper limit. You can add a `geom` to a pl
```