Introduction to R

CMED6020 – Session 1

Eric Lau (ehylau@hku.hk)

School of Public Health
The University of Hong Kong

11 Jan 2021

Overview of the course

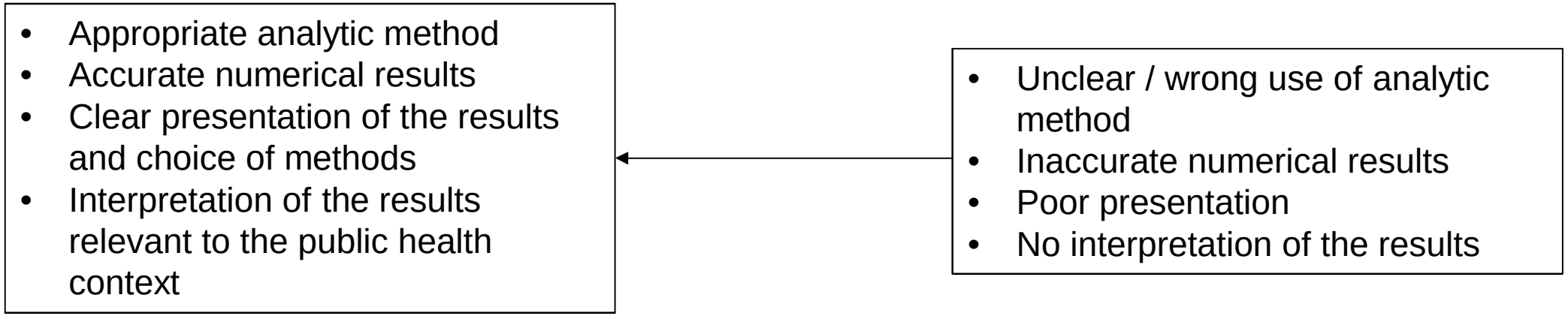
Course outline and schedule of lecture

Date	**Time**	**Topic**
Jan 11	18.30	Introduction to R
Jan 18	18.30	Regression model in R
Jan 25	18.30	Applied regression I
Feb 1	18.30	Applied regression II
Feb 22	18.30	Applied regression III
Mar 1	18.30	Conditional logistic regression and propensity score method
Mar 8	18.30	Inverse probability weighting and meta analysis
Mar 15	18.30	Instrumental variable analysis

- format: online lecture + practical
- online tutorials after sessions 2, 5 and 8
- 1st tutorial: Jan 21 (Thu) 2-5pm / Jan 23 (Sat) 2-5pm

Course assessment

- Coursework: 30% - 3 assignments
- Final exam: 70%
  - Mar 29, 2021 (Monday)
  - 18:30-21:30 online
  - Open book exam
- Grading: high



Outline

- 6:30 to 7:45 – Introduction to R
- 7:45 to 8:30 – R Graphics
- 8:30 to 9:00 – Practical

Session 1 learning objectives

After this session, students should be able to

- Use R to perform basic algebraic operations
- Work with variables, vectors and matrices in R
- Produce clear and well-formatted graphs in R
- Install and load R packages for specific needs

Introduction to R

Introduction to R

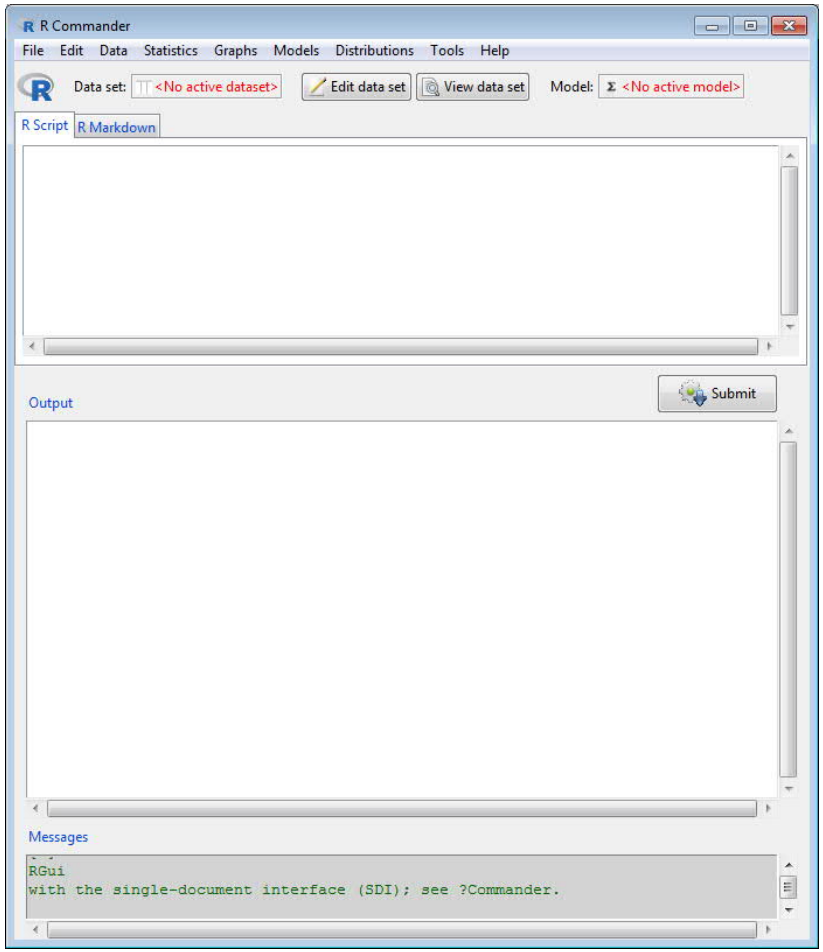
- Use of R as a calculator
- Vectors, matrices, and their operations
- Logic in R
- Data frames
- Summaries, tables
- Reading in data

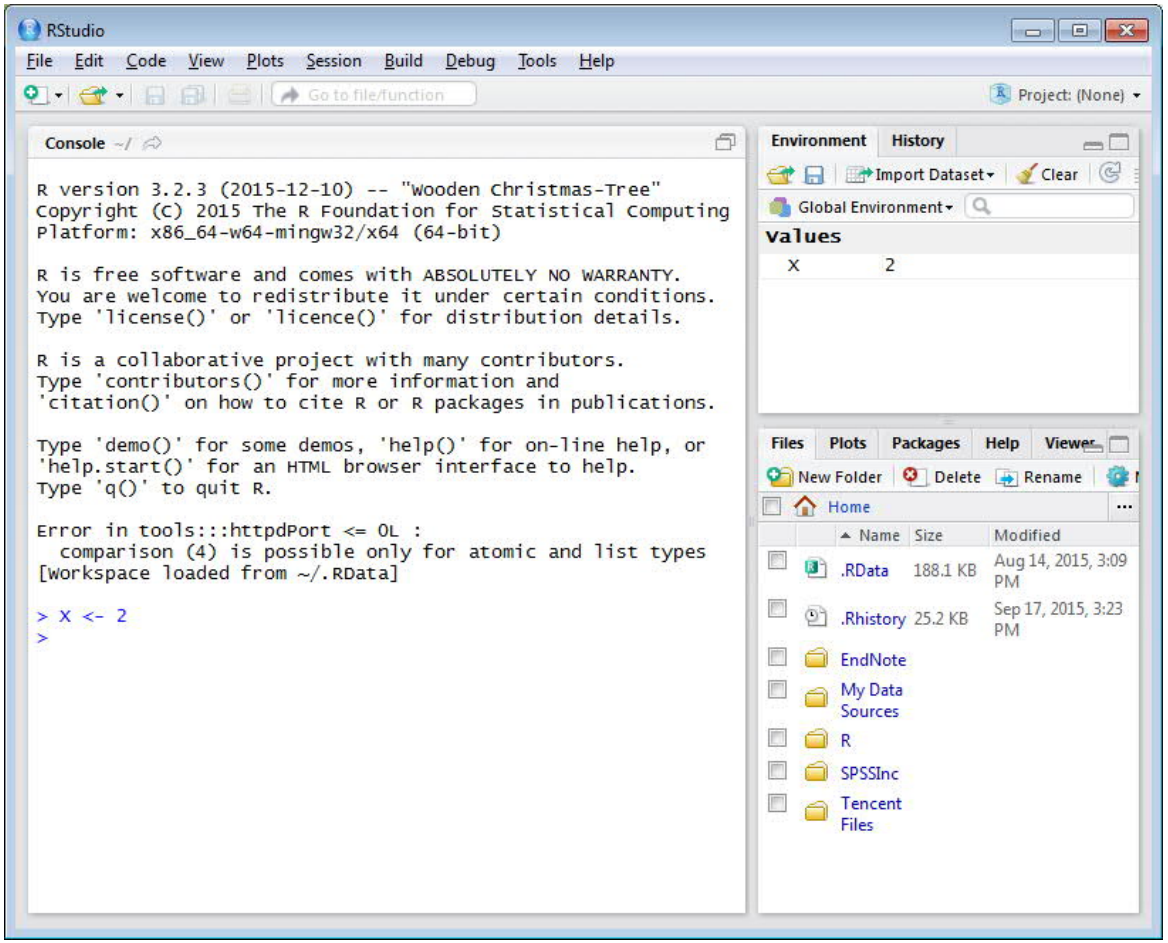
Introduction to R

- Open source, free statistical software
- Why do we use R in this course?
  - powerful computing
  - flexible for advanced methods
  - frequently updated packages
  - publication-quality graphics
- Latest version: 4.0.3 (as of Jan 2021)
- R resources: www.r-project.org (or just google “R”)

Other GUI for R

- RStudio (highly recommended)
- R-commander





Use of R as a calculator

- Basic algebraic operations

```
5 + 2
[1] 7
5 * 2
[1] 10
5 * 2 + 1
[1] 11
5 * (2 + 1)
[1] 15
5^2
[1] 25
5 / 2
[1] 2.5
```

Basic algebraic operations

```
sqrt(16)
[1] 4
exp(1)
[1] 2.718282
log(10)
[1] 2.302585
log(10, base=10)
[1] 1
```

- Other algebraic function
  - sin, cos, tan, abs, round, ceiling, floor, etc.
- Help
  - help(**func**), ?**func**, ??**keyword**
- Example
  - example(**func**)

Working with variables

- Variable names
  - acceptable characters: letters, underscores, dots
  - case sensitive
  - cannot start with a number
  - e.g. age, age.male, max_height, etc…
- Variable assignment

```
x <- 5
x + 2
[1] 7
x = 4
x^2
[1] 16
x = x*3 + 1
x
[1] 13
```

Vector assignment

```
lat <- c(2, 3, 5, 1, 3, 4, 2, 2)

lat
[1] 2 3 5 1 3 4 2 2
out <- c("recovery", "death")

out
[1] "recovery" "death"
rep(2,5)
[1] 2 2 2 2 2
1:6
[1] 1 2 3 4 5 6
seq(1,10,by=3)
[1] 1 4 7 10
```

Vector operations

```
length(lat)
[1] 8
sum(lat)
[1] 22
mean(lat)
[1] 2.75
median(lat)
[1] 2.5
```

```
lat
[1] 2 3 5 1 3 4 2 2
```

Vector operations

```
lat
[1] 2 3 5 1 3 4 2 2
```

```
min(lat)
[1] 1
max(lat)
[1] 5
var(lat)
[1] 1.642857
sort(lat)
[1] 1 2 2 2 3 3 4 5
which(lat==3)
[1] 2 5
summary(lat)
Min.  1st Qu.  Median   Mean  3rd Qu.   Max.
1.00     2.00    2.50   2.75    3.25    5.00
```

Vector operations

```
x <- 1:4
x
[1] 1 2 3 4
y <- c(2,5,3,1)
x + y
[1] 3 7 6 5

x * y
[1] 2 10 9 4
2 * x
[1] 2 4 6 8
x^y
[1] 1 32 27 4
x^2
[1] 1 4 9 16
```

Question:

How to calculate 1+2+…+1000 using R?

Accessing vectors

```
lat[2]
[1] 3
lat[c(2,5:7)]
[1] 3 3 4 2
lat[-4]
[1] 2 3 5 3 4 2 2
lat[-c(2:6)]
[1] 2 2 2
```

```
lat
[1] 2 3 5 1 3 4 2 2
```

Working with matrices

- Matrix assignment (default is by column)

```
matrix(1:9, nrow=3, ncol=3)
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9

matrix(1:9, nrow=3, ncol=3, byrow=T)
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9
```

Matrix operations

- Element-wise operation

```
X <- matrix(1:9, nrow=3, ncol=3)
Y <- matrix(c(-2, 0, 0, 0, 0, 1, 0, 1, 0), nrow=3,
   ncol=3)
X + Y
     [,1] [,2] [,3]
[1,]   -1    4    7
[2,]    2    5    9
[3,]    3    7    9
X * Y
     [,1] [,2] [,3]
[1,]   -2    0    0
[2,]    0    0    8
[3,]    0    6    0
```

```
X
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9

Y
     [,1] [,2] [,3]
[1,]   -2    0    0
[2,]    0    0    1
[3,]    0    1    0
```

Matrix operations

```
X
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9

Y
     [,1] [,2] [,3]
[1,]   -2    0    0
[2,]    0    0    1
[3,]    0    1    0
```

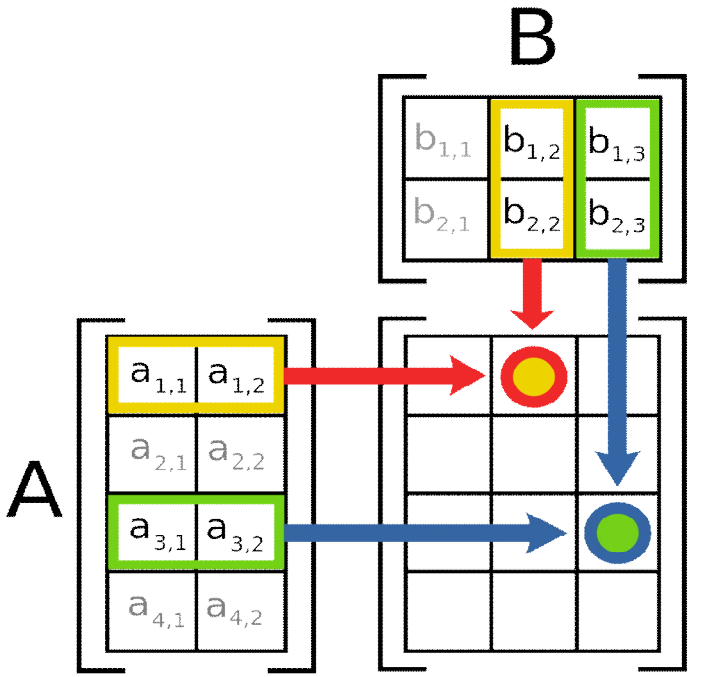
- Matrix multiplication

```
X %*% Y
     [,1] [,2] [,3]
[1,]   -2    7    4
[2,]   -4    8    5
[3,]   -6    9    6
```

- Inverse

```
solve(Y)
     [,1] [,2] [,3]
[1,] -0.5    0    0
[2,] -0.0    0    1
[3,] -0.0    1    0
```

Matrix multiplication (from Wikipedia)



Matrix operations

- Transpose

```
t(X)
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9
```

```
X
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
```

Matrix operations

- Column sum

```
colSums(X)
[1] 6 15 24
```

- Row sum

```
rowSums(X)
[1] 12 15 18
```

- Column mean

```
colMeans(X)
[1] 2 5 8
```

- Row mean

```
rowMeans(X)
[1] 4 5 6
```

- Matrix dimension

```
dim(X)
[1] 3 3
```

```
X
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
```

Matrix operations

- Combining matrices

```
> cbind(X, Y)
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    1    4    7   -2    0    0
[2,]    2    5    8    0    0    1
[3,]    3    6    9    0    1    0

> rbind(X, Y)
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
[4,]   -2    0    0
[5,]    0    0    1
[6,]    0    1    0
```

```
X
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9

Y
     [,1] [,2] [,3]
[1,]   -2    0    0
[2,]    0    0    1
[3,]    0    1    0
```

Accessing matrices

```
X[1,2]
[1] 4
X[ ,2]
[1] 4 5 6
X[1, ]
[1] 1 4 7
X[1,2:3]
[1] 4 7
```

```
X
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
```

Logic in R

- Logical operators, <, >, <=, >=, == != (inequality)
- ! (negation)
- & (and)
- | (or)

```
lat
[1] 2 3 5 1 3 4 2 2
```

Question:

How to add up those numbers in the variable *lat* which is not equal to 2?

```
lat >= 4
[1] FALSE FALSE  TRUE FALSE FALSE  TRUE FALSE FALSE
lat == 3
[1] FALSE  TRUE FALSE FALSE  TRUE FALSE FALSE FALSE
(lat > 2) & (lat != 4)
[1] FALSE  TRUE  TRUE FALSE  TRUE FALSE FALSE FALSE
```

Data frame

- A general purpose table to store data

```
sex <- c(rep("M",4), rep("F",4))
age <- c(18, 25, 23, 21, 25, 32, 22, 28)
height <- c(170, 182, 169, 170, 165, 161, 169, 163)
data1 <- data.frame(Sex=sex, Age=age, Height=height)
```

```
data1
  Sex Age Height
1   M  18    170
2   M  25    182
3   M  23    169
4   F  21    170
5   F  25    165
6   F  32    161
7   F  22    169
8   F  28    163
```

Accessing data frame

```
data1$Age
[1] 18 25 23 21 25 32 22 28
data1[,2]
[1] 18 25 23 21 25 32 22 28
data1[3,2]
[1] 23
data1[1:2,]
  Sex Age Height
1   M  18    170
2   M  25    182
colnames(data1)
[1] "Sex"  "Age"  "Height"
```

```
> data1
  Sex Age Height
1   M  18    170
2   M  25    182
3   M  23    169
4   M  21    170
5   F  25    165
6   F  32    161
7   F  22    169
8   F  28    163
```

- head(*data.frame*) to show the upper part of the data frame

Accessing data frame

```
data1$Sex
[1] M M M M F F F F
Levels: F M
```

```
> data1
  Sex Age Height
1   M  18    170
2   M  25    182
3   M  23    169
4   M  21    170
5   F  25    165
6   F  32    161
7   F  22    169
8   F  28    163
```

- Characters are automatically stored as factor in data frames
- The first level (F) is the reference group

```
is.factor(data1$Sex)
[1] TRUE
is.factor(data1$Age)
[1] FALSE
```

- as.factor(*vector*) to coerce characters to factors
- attach(*data.frame*) to access the variables directly using their names (without "*data.frame*$")
- detach(*data.frame*) to remove these variables

Accessing data frame

- cbind and rbind can also be used to combine data frames

```
data1$Smoking <- c("Y", "N", "Y", "N", "N", "N", "N", "N")
data1
  Sex Age Height Smoking
1   M  18    170       Y
2   M  25    182       N
3   M  23    169       Y
4   F  21    170       N
5   F  25    165       N
6   F  32    161       N
7   F  22    169       N
8   F  28    163       N
```

Summaries and tables

- Summarize data frame in table form
- aggregate(*x*, *by*, *FUN*)
  - **x** is an R object, e.g. vector, matrix, data frame
  - **by** is a list of grouping elements
  - **FUN** is a function to be evaluated

```
> data1
  Sex Age Height Smoking
1   M  18    170       Y
2   M  25    182       N
3   M  23    169       Y
4   M  21    170       N
5   F  25    165       N
6   F  32    161       N
7   F  22    169       N
8   F  28    163       N
```

```
aggregate(data1$Height, by=list(data1$Sex), FUN="mean")
  Group.1      x
1       F 164.50
2       M 172.75
```

```
aggregate(data1[,2:3], by=list(data1$Sex), FUN="mean")
  Group.1    Age Height
1       F  26.75 164.50
2       M  21.75 172.75
```

Summaries and tables

```
> data1
  Sex Age Height Smoking
1   M  18    170       Y
2   M  25    182       N
3   M  23    169       Y
4   M  21    170       N
5   F  25    165       N
6   F  32    161       N
7   F  22    169       N
8   F  28    163       N
```

- Produce contingency table in frequencies

```
table1 <- table(data1$Sex, data1$Smoking)
table1
   N Y
F  4 0
M  2 2
```

- Produce contingency table in proportions
- prop.table(*table*, *margin*)
  - **margin** specifies whether row proportions (**margin** = 1) or column proportions (**margin** = 2) are calculated

```
prop.table(table1, margin=2)
            N    Y
F 0.6666667 0.00
M 0.3333333 1.00
```

Reading in data

- read.table(*file*, *header*, *sep*)
  - **file** is the path and filename of the dataset
  - **header=T** indicates column headings are included in the first line of the file
  - **sep** specifies how data are separated (e.g. commas **","**; blank spaces **" "**; tabs **"\t"**)

```
mvc <- read.table("http://web.hku.hk/~ehylau/mvc.csv",
   header=T, sep=",")
mvc[1:2,]
  age height MVC
1  24    166 466
2  27    175 304
```

- read.csv can also be used for .csv (comma separated values) files

```
mvc <- read.csv("http://web.hku.hk/~ehylau/mvc.csv",
   header=T)
```

(or “d:/<directory/folder>/mvc.csv” if the file is in the local computer)

Working directory

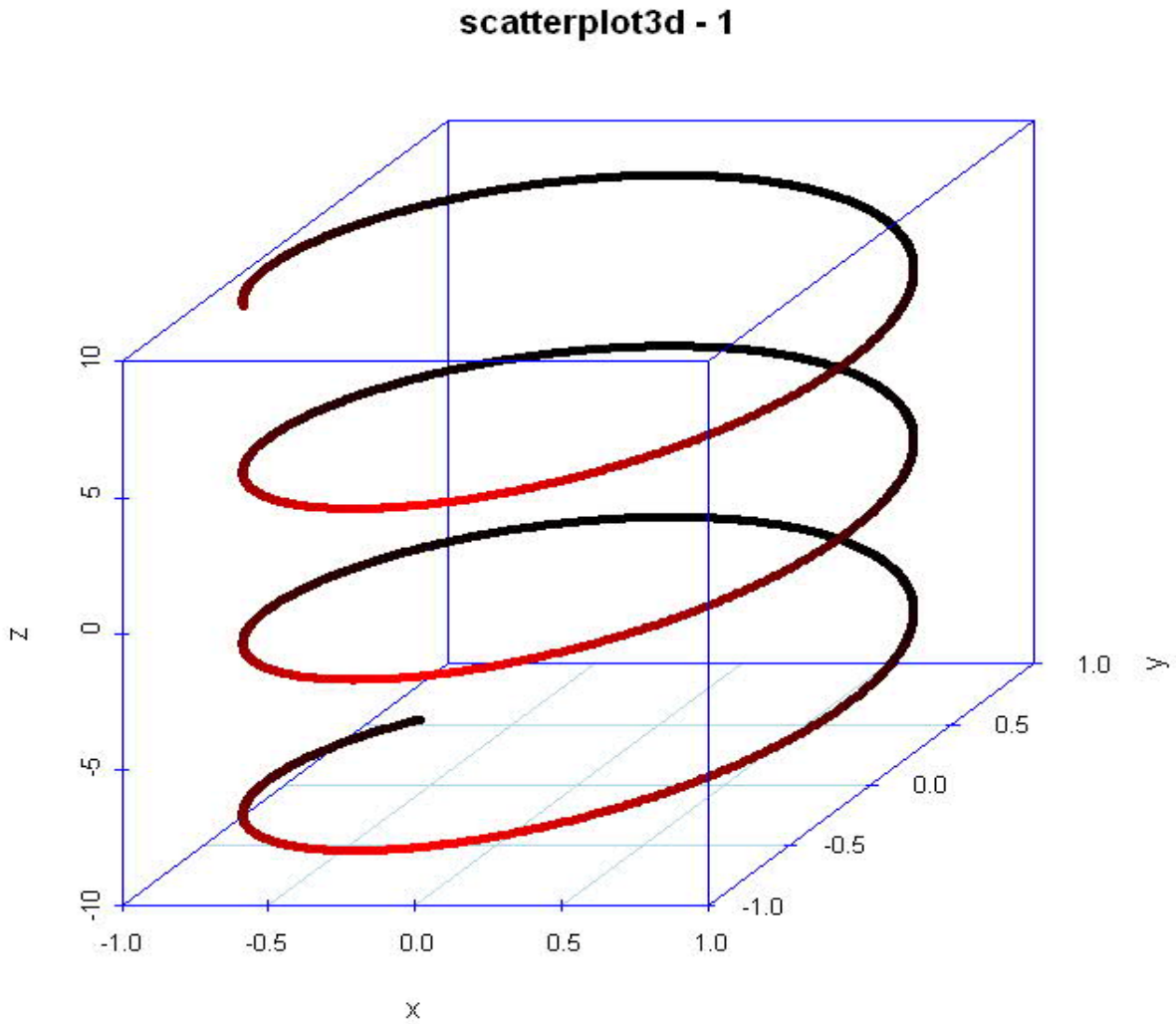
- Setting the default directory/folder to retrieve or save files
- getwd()
  - get working directory
- setwd()
  - set working directory

Packages in R

- Written by different authors for specific purpose
- A lot of packages can be downloaded to the “library” folder from the CRAN website (http://cran.r-project.org)
- Packages can install from remote website in “Packages” → “Install package(s)”
- library(*package*) or require(*package*) to load packages

```
library(scatterplot3d)
z <- seq(-10, 10, 0.01)
x <- cos(z)
y <- sin(z)
scatterplot3d(x, y, z, highlight.3d=TRUE, col.axis="blue",
  col.grid="lightblue", main="scatterplot3d - 1", pch=20)
```

Packages in R



R Graphics

R Graphics

- **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.

High-level plotting functions

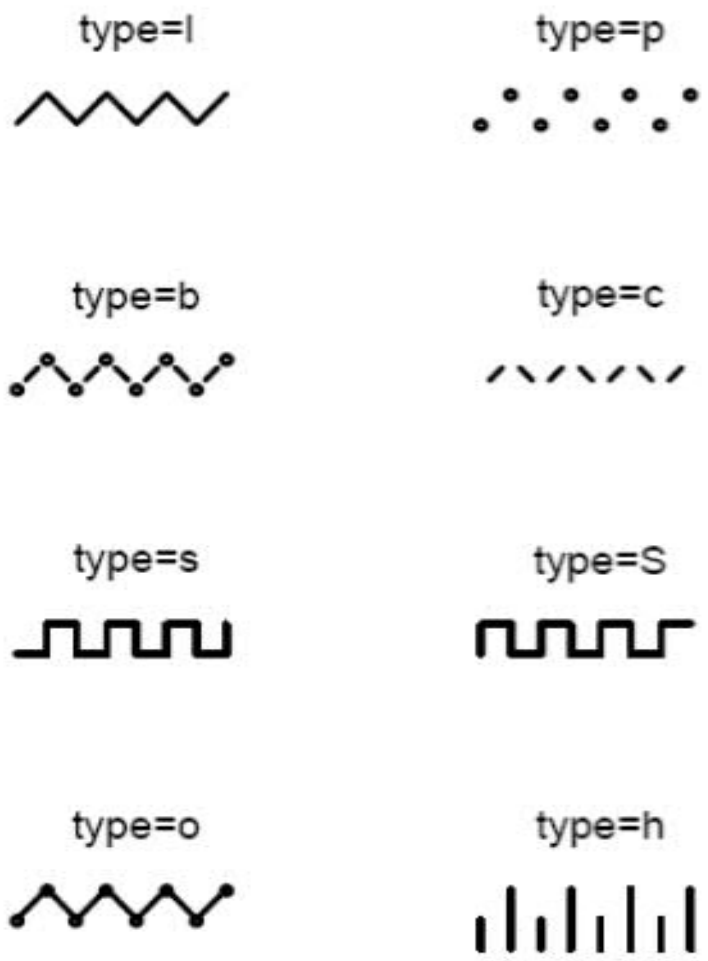
- plot(x, y, …)
- pairs(data)
- hist(x)
- boxplot(x) and boxplot(y~x)

The plot() function

- plot(x, y)
- plot(x, y, type="p", axes=TRUE, xlim=c(0,1), ylim=c(0,1), main="Main title", xlab="x-axis title", ylab="y-axis title")
  - **type** indicates type of plot (points, line, both, …).
  - **axes** indicates whether axes should be plotted.
  - **xlim, ylim** specify the limits of the x- and y-axes.
  - **main, xlab, ylab** specify the main and axes titles.

Plot types

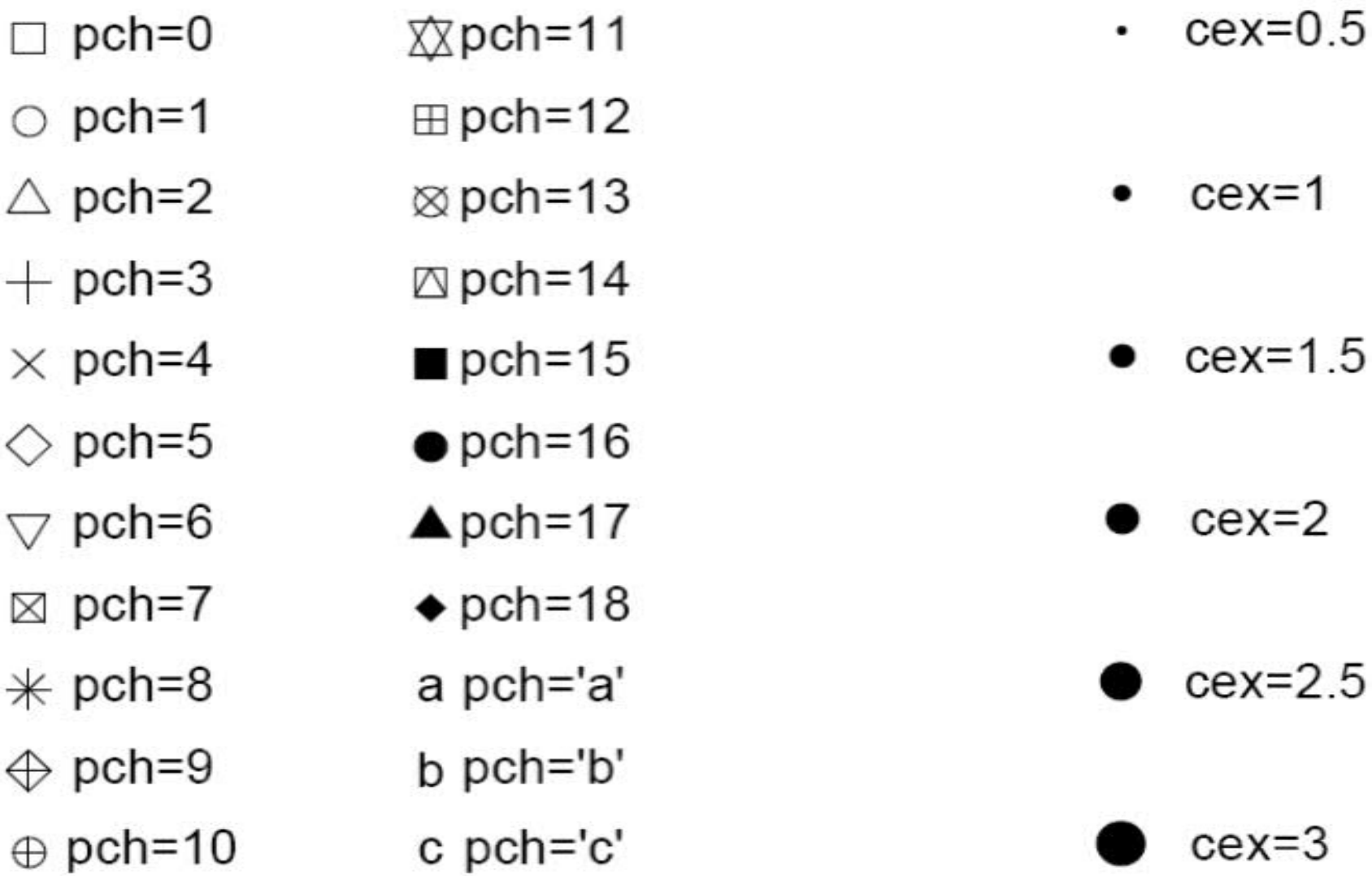
- type="n" plots nothing, but sets up the plot region and coordinate system (this can be very useful!)
- Other plot types shown on the right



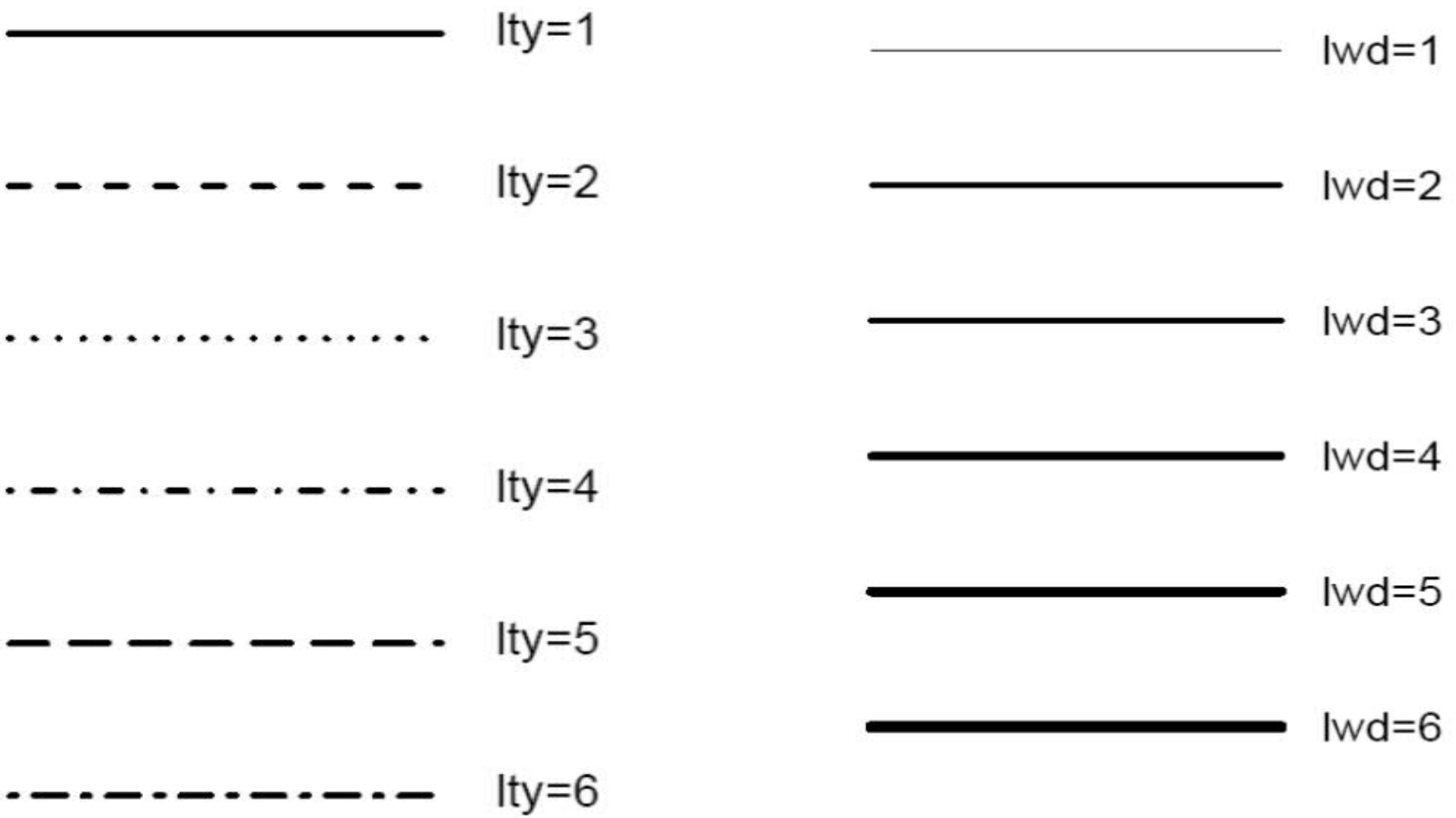
Lower level plotting commands

- points(x, y)
- points(x, y, pch=1, cex=1, col=1)
  - **pch** is the plot character
  - **cex** is the character expansion (relative size)
  - **col** is the colour
- lines(x, y)
- lines(x, y, lty=1, lwd=1)
  - **lty, lwd** are the line type and width respectively

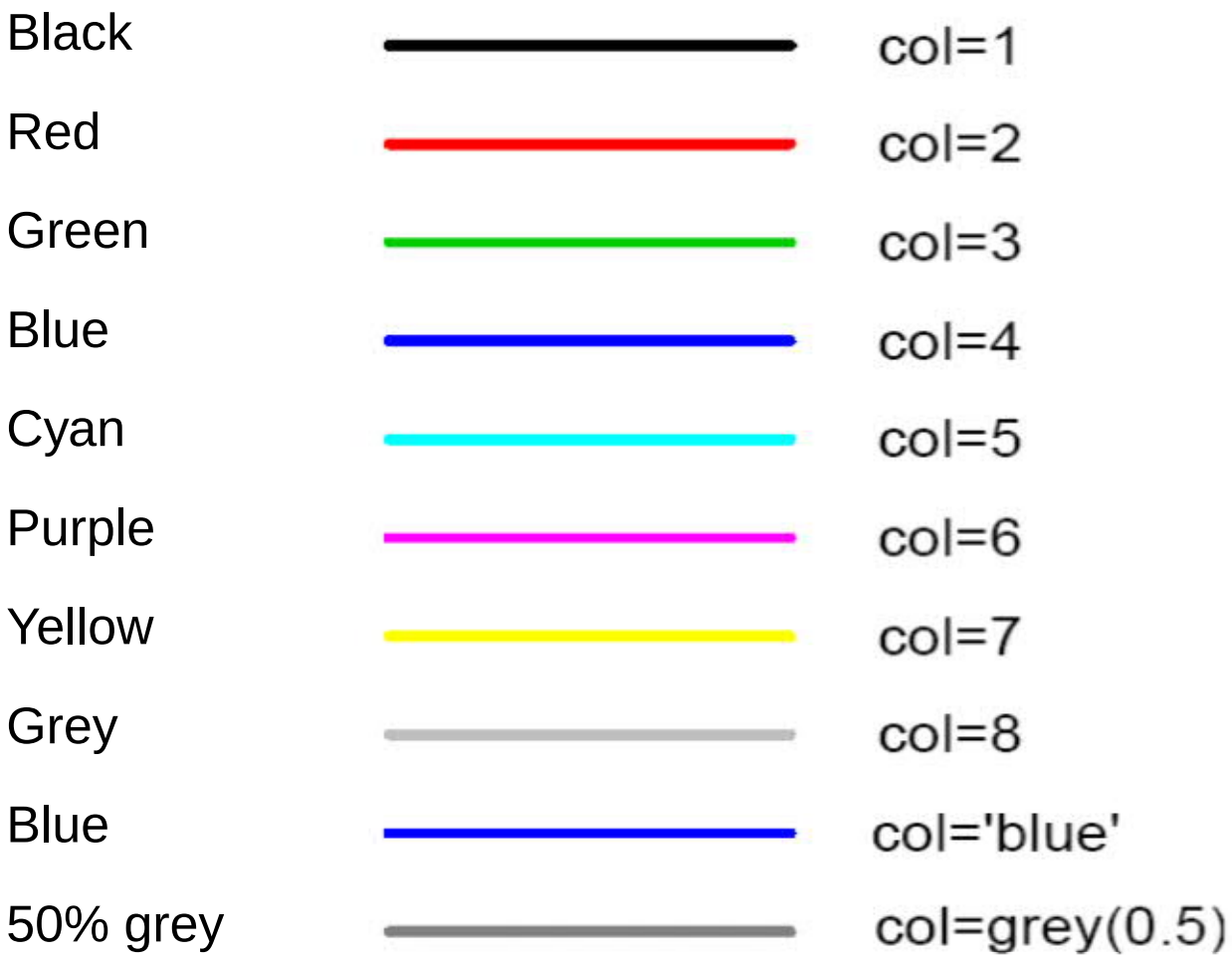
Points – pch and cex



Lines – lty and lwd



Colours


Black
col=1
Red
col=2
Green
col=3
Blue
col=4
Cyan
col=5
Purple
col=6
Yellow
col=7
Grey
col=8
Blue
col='blue'
50% grey
col=grey(0.5)

Other plotting commands

- text(x, y, “text”) … adds “text” at x, y
- abline(a, b) … adds line of slope b, intercept a
- abline(h=y) … adds a horizontal line at y
- abline(v=x) … adds a vertical line at x
- polygon(x, y) … adds a polygon with corners at each x_i, y_i

  (x and y should be vectors)
- legend(x, y, text) … adds a legend at x,y
- title(“title text”) … adds a title

Adding an axis

- axis(side, pos, at, labels)
  - **side** =1 for bottom, 2 for left, 3 for top, 4 for right
  - **pos** = positioning of axis
  - **at** = where tick marks should be drawn
  - **labels** = what to write next to each tick mark
- e.g. axis(1, pos=0, at=0:5, labels=0:5)
  - This adds an x-axis at the horizontal location y=0 with tick marks at x=0,1,…,5 and corresponding labels.
- For a y-axis add the option “las=1” to get horizontal rather than vertical text labels (looks nicer!)

Further customisation

- par() … used to access and modify graphics parameters for the current device
- par(col="red", lty=2) … change the default colour to red and the default line type to 2 (dotted)
- par(mar=c(5,4,1,1)) … change the default plot margins to 5,4,1,1 on the bottom, left, top, right (the default is 5,4,2,2).
- cex.axis, cex.lab, cex.main
  - Character expansion (relative sizes) for axis annotation, axis titles and main titles respectively

Plot windows

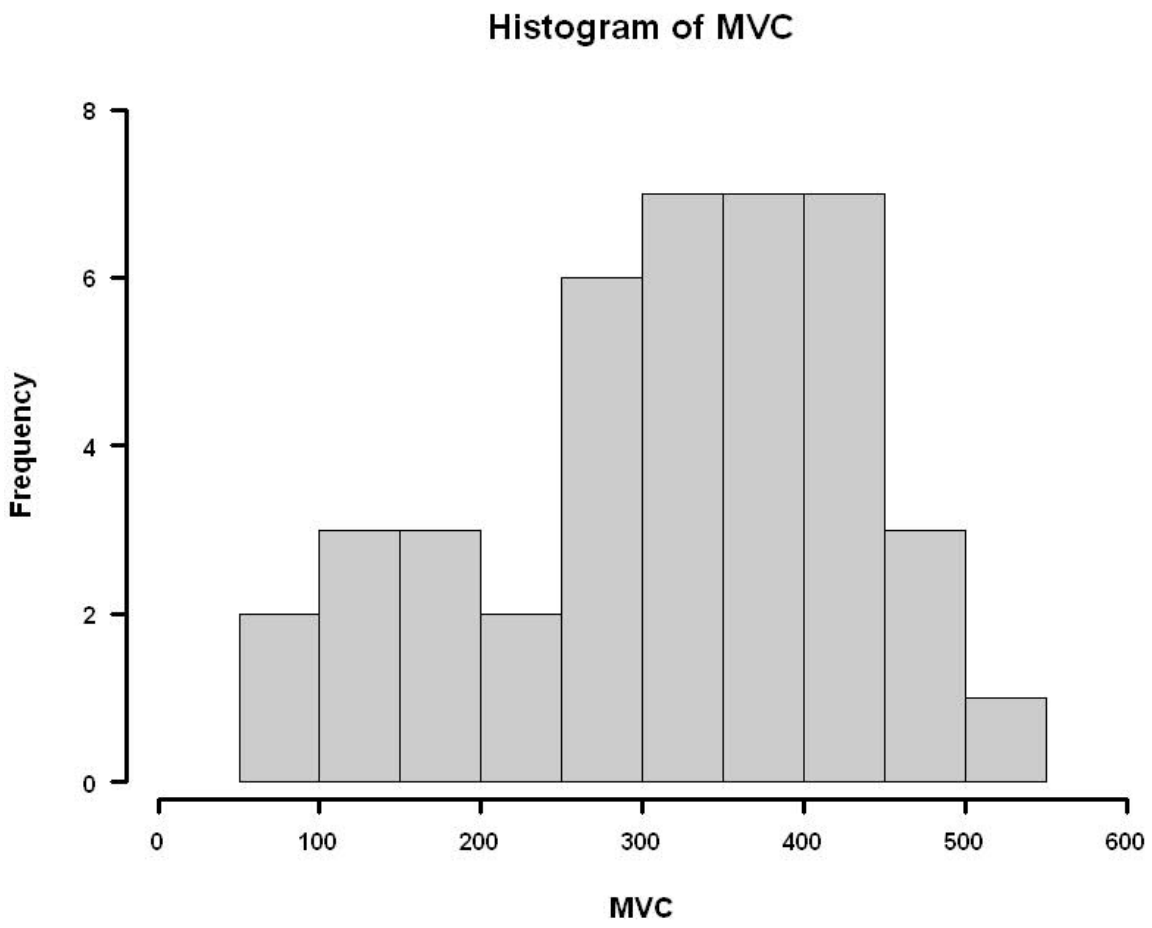
- windows(width=6, height=5)
  - Opens a new graphics window with width=6 inches and height=5 inches
  - quartz("Quartz", width=6 , height=5) [for Mac OS]
- layout(matrix(1:4, nrow=2, byrow=TRUE), widths=c(1,1), heights=c(1,3))
  - Split the graphics window into 4 subplots where the top two occupy 25% and bottom two occupy 75% of the plot region, respectively
- par(mfrow=c(2,2)) / par(mfcol=c(2,2))
  - Split the graphics window evenly into 4 subplots, shown in an order by rows (mfrow) or columns (mfcol)

Example – MVC data

- mvc <- read.csv("http://web.hku.hk/~ehylau/mvc.csv")
- Data includes ages, heights and maximum voluntary contraction of the quadriceps muscle (MVC) in a group of male alcoholics
- Can MVC be predicted from the other variables / what is the association between them?

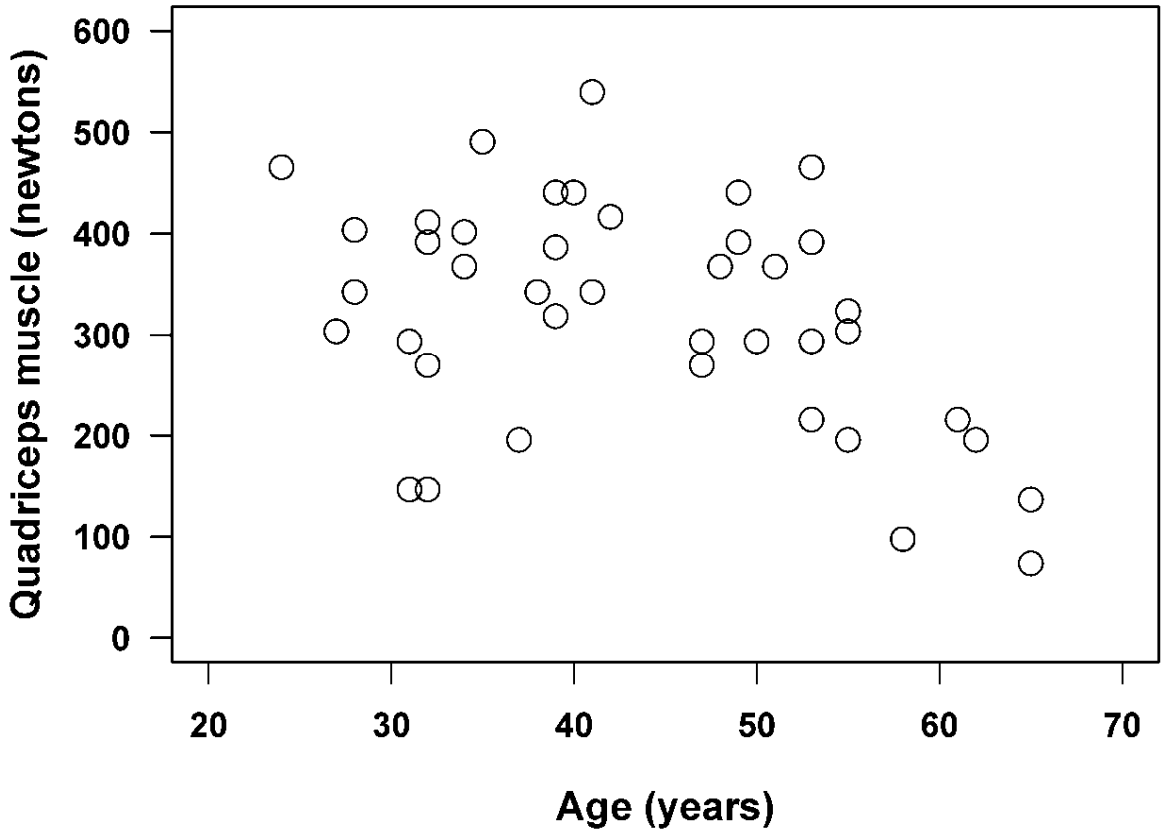
Plot a histogram

```
hist(mvc$MVC, axes=FALSE,
   xlim=c(0, 600), ylim=c(0,
   8), font.lab=2,
   cex.lab=1.2, cex.main=1.5,
   col=grey(0.8),
   xlab="MVC",
   ylab="Frequency",
   main="Histogram of MVC")
axis(1, pos=-0.2, lwd=3.5,
   font=2)
axis(2, pos=-20, lwd=3.5,
   font=2, las=1)
```



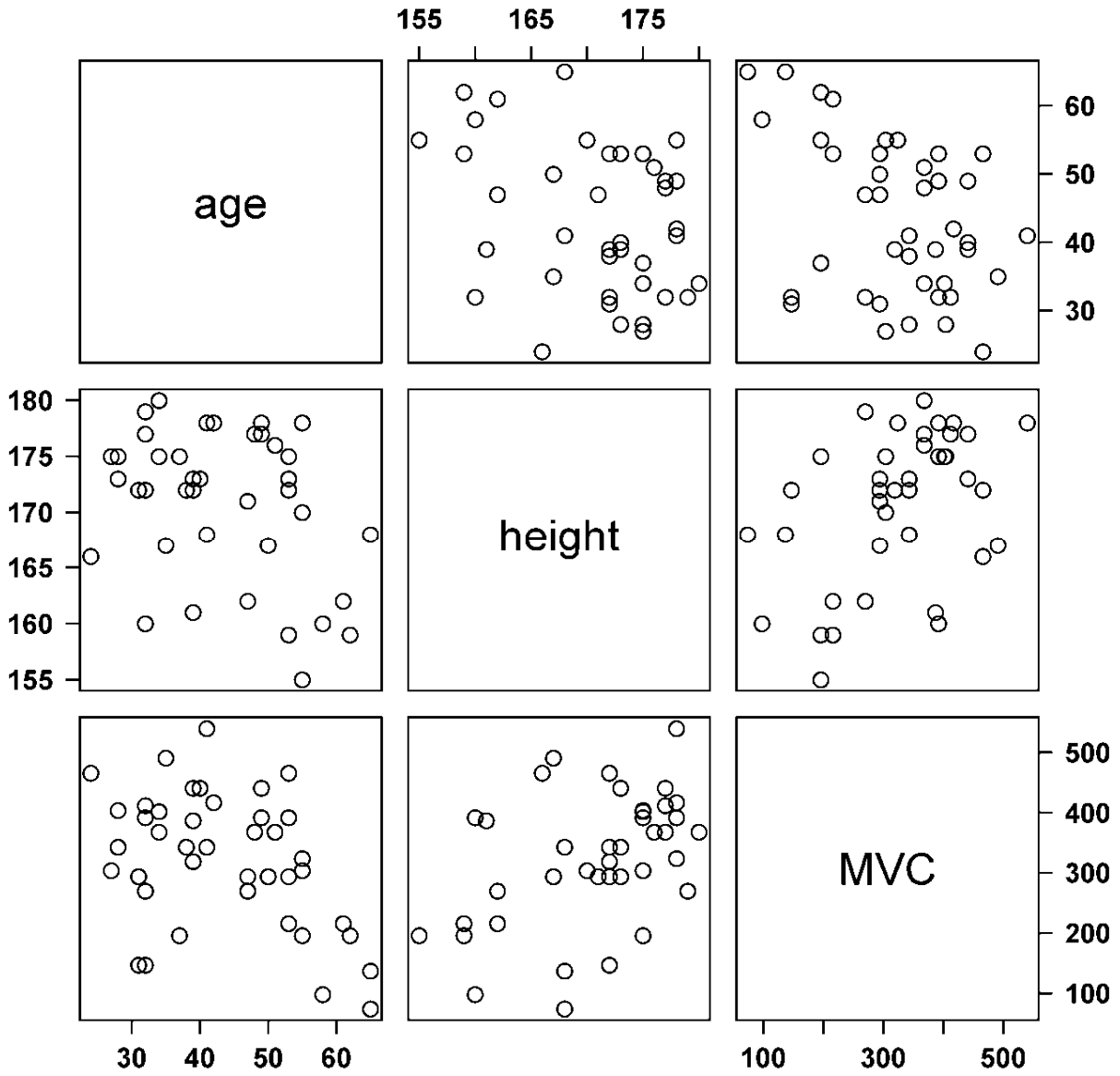
Plot two variables side by side

```
plot(MVC ~ age, data=mvc,
   xlab="Age (years)",
   ylab="Quadriceps muscle
   (newtons)", xlim=c(20,
   70), ylim=c(0, 600),
   cex=1.5, cex.lab=1.2,
   font.lab=2, font.axis=2,
   las=1)
```



Scatter plot matrix

```
pairs(mvc, cex =1.5,
   font=2, las=1,
   cex.axis=1.2)
```



Save graphical output

- pdf(*file*, *width*, *height*)
  - this will start the graphics device driver
  - **file** is the filename and its directory
- can also save in other format: e.g. bmp, jpeg, png, tiff
- dev.off()
  - close the graphics device
- Example:

```
pdf("d:/figure1.pdf", width=6, height=4)
plot(MVC ~ age, data=mvc, xlab="Age (years)", ylab="Quadriceps
   muscle (newtons)", xlim=c(20, 70), ylim=c(0, 600), cex=1.5,
   cex.lab=1.2, font.lab=2, font.axis=2, las=1)
dev.off()
```

Useful R materials

- Paradis E. R for beginners.
  http://cran.r-project.org/doc/contrib/Paradis-rdebuts_en.pdf

- Contributed packages. http://cran.r-project.org/web/packages

- Zheng T. Colors in R.
  http://www.stat.columbia.edu/~tzheng/files/Rcolor.pdf