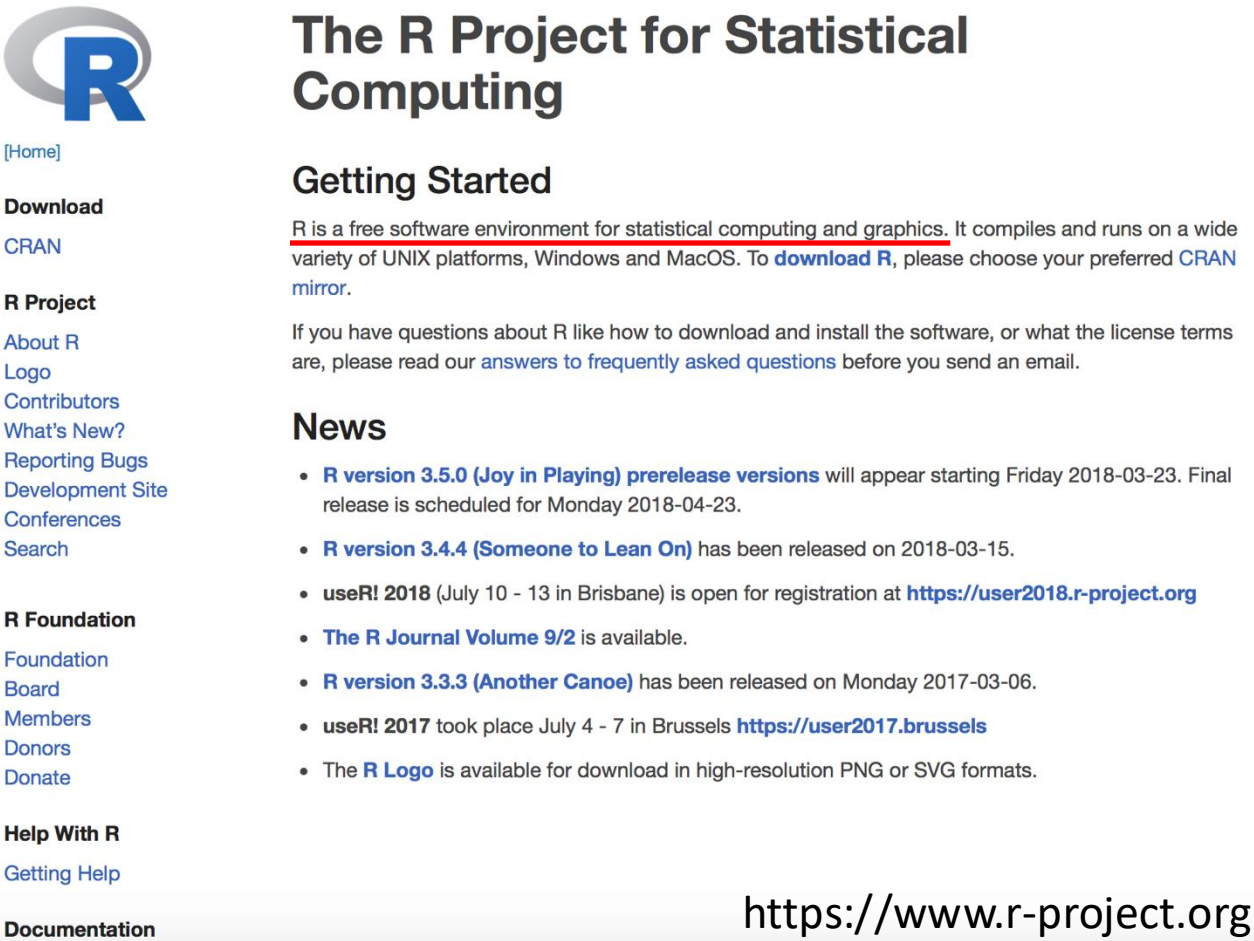
生物資訊分析
Bioinformatic analysis

Introduction to R

蔡佩倩
Pei-Chien Tsai

R/RStudio

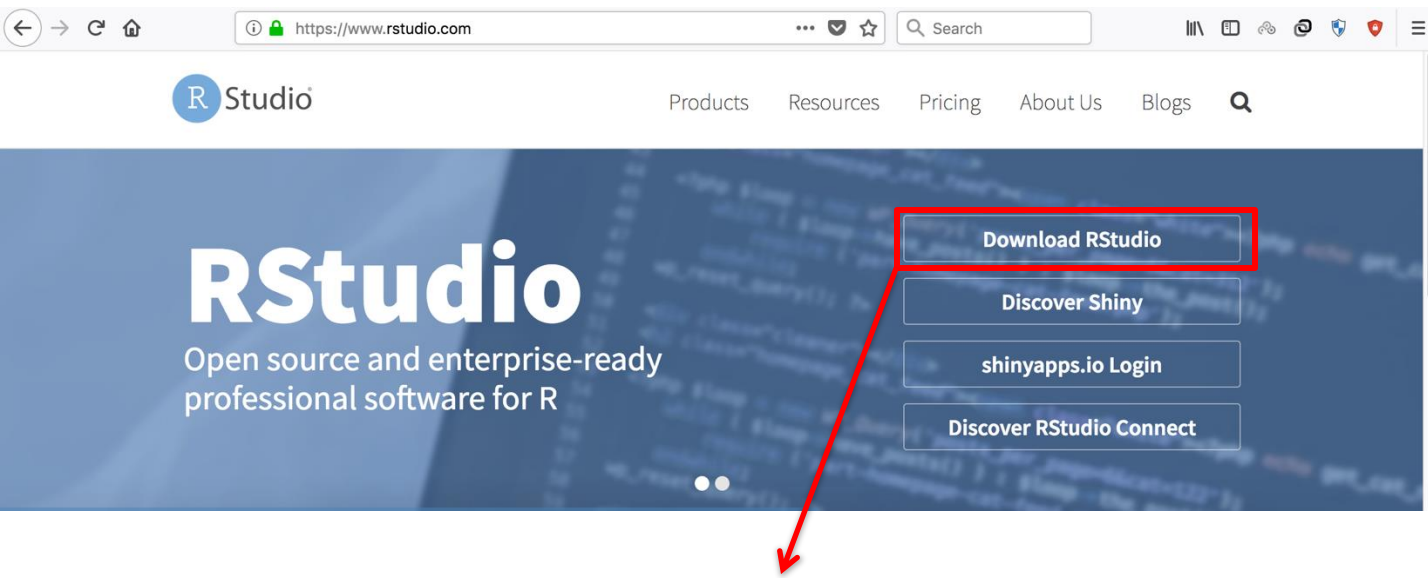
- R is one of the most commonly used software for statistical computing and graphic.
- There are other software packages for statistics, such as SAS, SPSS, JUMP
- R is specifically good at handling big data
- R is the most commonly used software in the bioinformatics area, where SAS is used in the CRO company.
- R is **free**!



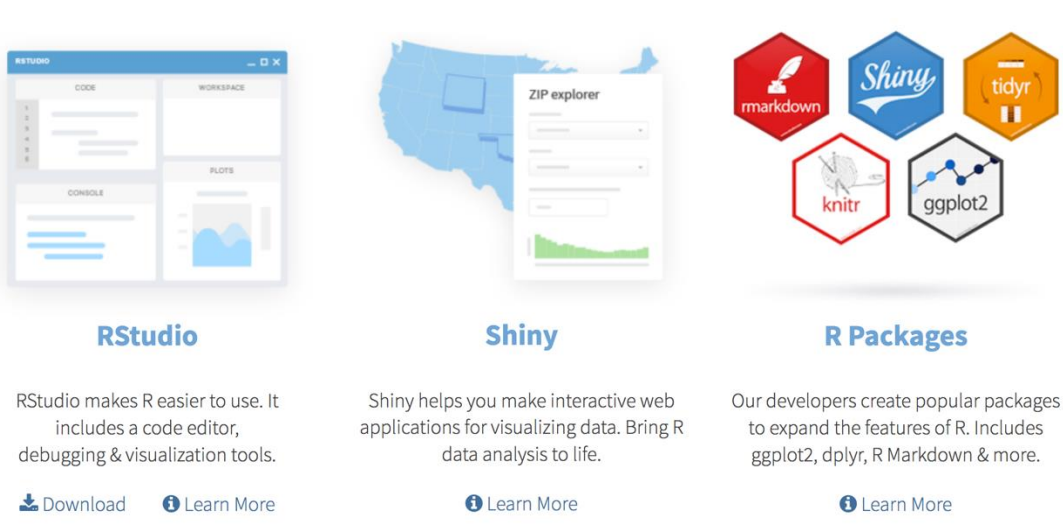
https://www.r-project.org

R/RStudio

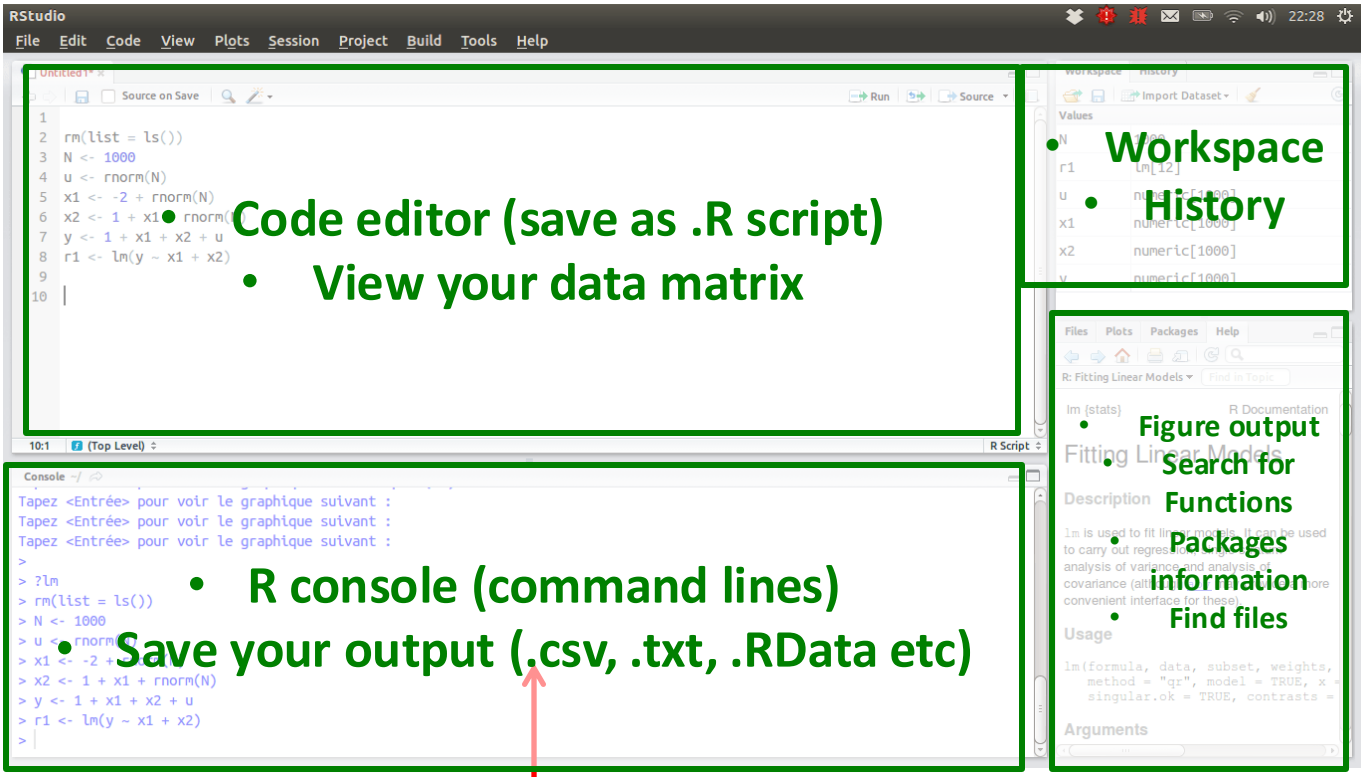
- RStudio is an easier interface for the starters because it shows ‘everything you need’ on the same page



Download from the website (make sure you download R first)

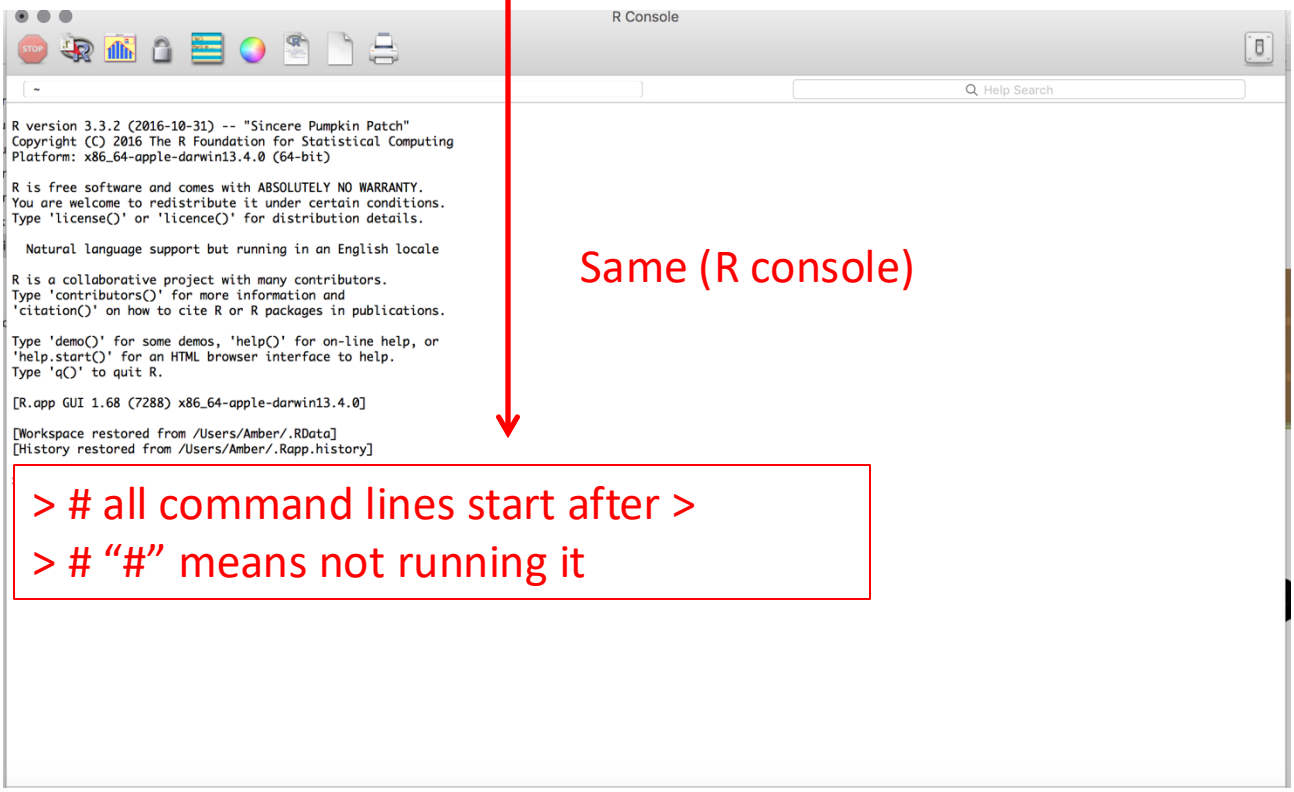


RStudio



- Code editor (save as .R script)
- View your data matrix
- Workspace
- History
- R console (command lines)
- Save your output (.csv, .txt, .RData etc)
- Figure output
- Search for Functions
- Packages information
- Find files

R



Same (R console)

> # all command lines start after >
> # "#" means not running it

I. Variable Types

- **Variable types**
  - Data frame
  - Numbers
  - Strings
  - Factors

- **Built-in data in R: mtcars**

there are many built-in data in R, mtcars is one of the dataset
Let's type mtcars in R (# notice that here I only shows the first 6 rows of the data)

> mtcars

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21	6	160	110	3.9	2.62	16.46	0	1	4	4
Mazda RX4 Wag	21	6	160	110	3.9	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108	93	3.85	2.32	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	3.44	17.02	0	0	3	2
Valiant	18.1	6	225	105	2.76	3.46	20.22	1	0	3	1

The first thing to do, is to check the meanings of your variables.

The data was extracted from the 1974 *Motor Trend* US magazine, and comprises fuel consumption and 10 aspects of automobile design and performance for 32 automobiles (1973–74 models).

In total there are 11 variables, including:

Columns	Variable name	Explaination
[, 1]	mpg	Miles/(US) gallon
[, 2]	cyl	Number of cylinders
[, 3]	disp	Displacement (cu.in.)
[, 4]	hp	Gross horsepower
[, 5]	drat	Rear axle ratio
[, 6]	wt	Weight (1000 lbs)
[, 7]	qsec	1/4 mile time
[, 8]	vs	Engine (0 = V-shaped, 1 = straight)
[, 9]	am	Transmission (0 = automatic, 1 = manual)
[,10]	gear	Number of forward gears
[,11]	carb	Number of carburetors

I. Variable Types

Let's get the first row of data:
> mtcars[1,]

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21	6	160	110	3.9	2.62	16.46	0	1	4	4

Let's get the first column of data:
> mtcars[,1]
[1] 21.0 21.0 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 17.8 16.4 17.3 15.2 10.4
[16] 10.4 14.7 32.4 30.4 33.9 21.5 15.5 15.2 13.3 19.2 27.3 26.0 30.4 15.8 19.7
[31] 15.0 21.4

Why the names of cars are not the first column of this data? Because these are the 'rownames'!

In R, the [,] actually shows the [row,column] of your data
For example, the 2nd row and 3rd column in the mtcars is:
> mtcars[2,3]
[1] 160

We can also check a 'range' of data, say row 5 to 7, column 3 to 7
> mtcars[5:7,3:7]

	disp	hp	drat	wt	qsec
Hornet Sportabout	360	175	3.15	3.44	17.02
Valiant	225	105	2.76	3.46	20.22
Duster 360	360	245	3.21	3.57	15.84

- **Data frame**

in this section, let's use a built-in data frame "mtcars"
a data frame may contain many lists, and each list might be a list of factors, strings, or numbers.
a data frame usually looks like a table or data matrix

> dim(mtcars) # check dimension of the data frame
[1] 32 11 # first number: rows; 2nd number: columns
> nrow(mtcars) # check how many 'rows' of the data frame
[1] 32
> ncol(mtcars) # check how many 'columns' of the data frame
[1] 11

> str(mtcars) # check the structure of the data frame
'data.frame': 32 obs. of 11 variables: # obs = rows; variables= cols
$ 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 …

I. Variable Types

- ## Strings

```
# generate a vector called “a”
# You can assign numbers or strings to a vector

> a <- "good"  # a string always need " "
> a
[1] "good"
> b <- c("good","morning") # you can put many strings
> b
[1] "good"    "morning"
> b[2]
[1] "morning"
```

- ## Numbers

```
> a <- 1 # a is a vector here, you assign number 1 in a
> a
[1] 1

> b <- a*4+3 # you can assign a formula as well
> b
[1] 7

> ls() # check how many vector, you can use ‘objects()’
[1] "a" "b”

> a <- c(1,2,3,4,5)
> b <- c(1:5) # or you can do something like c(1,3:4)
> a
[1] 1 2 3 4 5
> b
[1] 1 2 3 4 5
> a[4] # extract the 4th element from the vector
[1] 4
> a[4]-b[2]
[1] 2
> a[6] # there is no 6th element in ‘a’, returns ‘NA’
[1] NA

> length(a) # find how many data in the vector
[1] 5
```

I. Variable Types

- **Factors**

```
# a factor is usually contains different levels in a variable
# For example, let's take a look at the variable "gear" in "mtcars"
# remember in the previous slide, when we do str(mtcars), it tells you
# $ gear: num  4 4 4 3 3 3 3 4 4 4 ... # says , gear is a numeric vector
# a factor might look like a bunch of numeric numbers, but it means 'category'

> mtcars$gear
 [1] 4 4 4 3 3 3 3 4 4 4 4 3 3 3 3 3 3 4 4 4 3 3 3 3 3 4 5 5 5 5 5 4

> summary(mtcars$gear) # when you summary a numeric variable, you get summary
statistics
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  3.000   3.000   4.000   3.688   4.000   5.000

> factor(mtcars$gear) # when you summary a factor
 [1] 4 4 4 3 3 3 3 4 4 4 4 3 3 3 3 3 3 4 4 4 3 3 3 3 3 4 5 5 5 5 5 4
Levels: 3 4 5

> summary(factor(mtcars$gear)) # it gives you the count in the variable
 3  4  5
15 12  5

# let's try some more useful commands
> colnames(mtcars) # find all column names
 [1] "mpg"  "cyl"  "disp" "hp"   "drat" "wt"   "qsec" "vs"   "am"   "gear"
[11] "carb"
> rownames(mtcars) # find all row names
 [1] "Mazda RX4"          "Mazda RX4 Wag"      "Datsun 710"
 [4] "Hornet 4 Drive"     "Hornet Sportabout"  "Valiant"
 [7] "Duster 360"         "Merc 240D"          "Merc 230"
[10] "Merc 280"           "Merc 280C"          "Merc 450SE"
[13] "Merc 450SL"         "Merc 450SLC"        "Cadillac Fleetwood"
[16] "Lincoln Continental" "Chrysler Imperial"  "Fiat 128"
[19] "Honda Civic"        "Toyota Corolla"     "Toyota Corona"
[22] "Dodge Challenger"   "AMC Javelin"        "Camaro Z28"
[25] "Pontiac Firebird"   "Fiat X1-9"          "Porsche 914-2"
[28] "Lotus Europa"       "Ford Pantera L"     "Ferrari Dino"
[31] "Maserati Bora"      "Volvo 142E"

# Make sure you generate a 'Good/Reasonable' factor/data frame 'name'!!!
# Never use something like 'c', 'sum', 'for' … as your names
```

I. Variable Types

- **Factors**

```
# Let's practice more useful command using mtcars
# You can make x (the first variable you put) by y (the 2nd variable you put) table
# 'table' is useful for categorical variables, where summary is good for numeric variables

> table(mtcars$gear)
 3  4 5
15 12  5

> table(mtcars$gear,mtcars$am)
        0  1
 3     15  0
 4      4  8
 5      0  5


# Here are some advanced commands

# Let's say you want to find all information under the car- Merc 280
# You already know the car style is lists in the rownames

> mtcars[rownames(mtcars)=="Merc 280",]
                mpg cyl disp  hp drat   wt qsec vs am gear carb
Merc 280       19.2   6  168 123 3.92 3.44 18.3  1  0    4    4


# Find all car style if they have gear ==4
> rownames(mtcars[mtcars$gear==4,])
 [1] "Mazda RX4"      "Mazda RX4 Wag"  "Datsun 710"     "Merc 240D"
 [5] "Merc 230"       "Merc 280"       "Merc 280C"      "Fiat 128"
 [9] "Honda Civic"    "Toyota Corolla"     "Fiat X1-9"          "Volvo 142E"

# common calculation: +, - , * , /  & mean(), median(), range(), max(), min(), sum(),
average()…etc

#  Find all information if one car has mpg less than the "median-1SD mpg" of all cars

> mtcars[mtcars$mpg<(median(mtcars$mpg)-sd(mtcars$mpg)),]
                    mpg cyl disp  hp drat    wt  qsec vs am gear carb
Cadillac Fleetwood 10.4   8  472 205 2.93  5.25 17.98  0  0    3    4
Lincoln Continental 10.4  8  460 215    3 5.424 17.82  0  0    3    4
```

I. Variable Types

- **Logical**

There are two predefined variables, *TRUE* and *FALSE*. Let's check this example

```
> a <- c(2,1,5,6,3)
> a[3]
[1] 5

> a[3]==5 # you ask if a[3] equals to 5
[1] TRUE
> a[3]!=5 # you ask if a[3] not equal to 5
[1] FALSE
```

Some logical operators:

<	Less than	|	Entry wise or
>	Greater than	||	or
<=	Less than or equal	&	Entry wise and
>=	Greater than or equal	&&	and
==	Equal to	!	not
!=	Not equal to		

Let's try some of these command

```
> a>5
[1] FALSE FALSE FALSE  TRUE FALSE
> a>=5
[1] FALSE FALSE  TRUE  TRUE FALSE
> a ==2 | a==1
[1]  TRUE  TRUE FALSE FALSE FALSE
> a ==2 & a==1
[1] FALSE FALSE FALSE FALSE FALSE
```

If you want to get the exactly number, you simply add a the 'condition' inside your vector or data frame

```
> a[a>5]
[1] 6
> a[a>=5]
[1] 5 6
> a[a==2 | a==1]
[1] 2 1
```

I. Variable Types

- **Logical**

Example of & and &&

Let's say we have two vector x and y

```
> x <- 1:5
> y <- 5:1

> x
[1] 1 2 3 4 5
> y
[1] 5 4 3 2 1

> x>2
[1] FALSE FALSE  TRUE  TRUE  TRUE
> y<3
[1] FALSE FALSE FALSE  TRUE  TRUE


> x>2 & y<3
[1] FALSE FALSE FALSE  TRUE  TRUE


> x>2 && y<3
[1] FALSE

> x<2 && y>3
[1] TRUE


> x<2 & y>3
[1]  TRUE FALSE FALSE FALSE FALSE
```

Let's do some exercises

#First generate the following a and b:
> a <- c(1,5,2,3,6,4)
> b <- c(4,1,2,3,6,5)

Find the numbers in a and b that are equal
Ans:

what is the maximum product of a and b
Ans:

Let's use the 'mtcars' data frame
What is the number on the 15th row and 4th column?
Ans:

Is the number on the 11th row and 8th column the same as the 3rd row and 11th column?
Ans:

We already know there are 3 types of gears: 3, 4, 5
Find out which gear is less use among all these cars
Find all information of cars that use the gear you found in the previous question

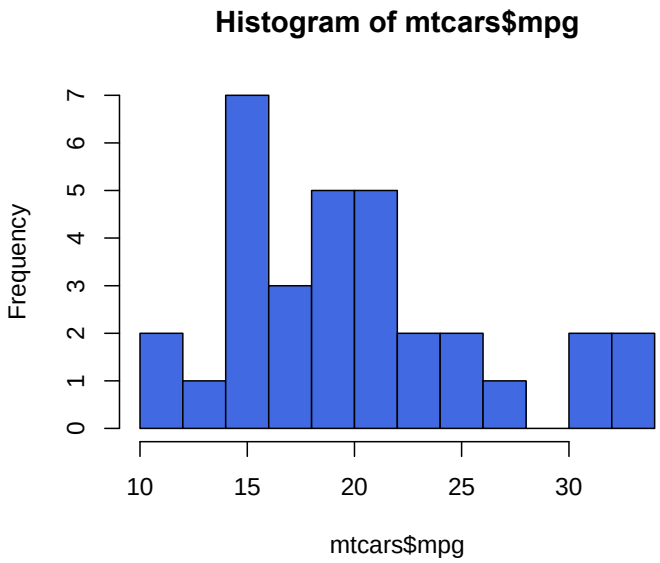
Ans:

III. Basic Plots

Let's use the dataset mtcars

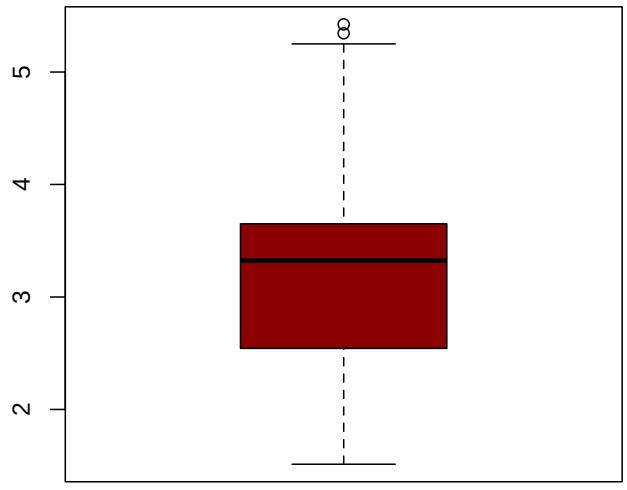
 # Histogram: distribution of a numeric variable

>hist(mtcars$mpg, breaks=12, col="royalblue")



 # Boxplot: A boxplot provides a graphical view of the median, quartiles, maximum, and minimum of a data set.

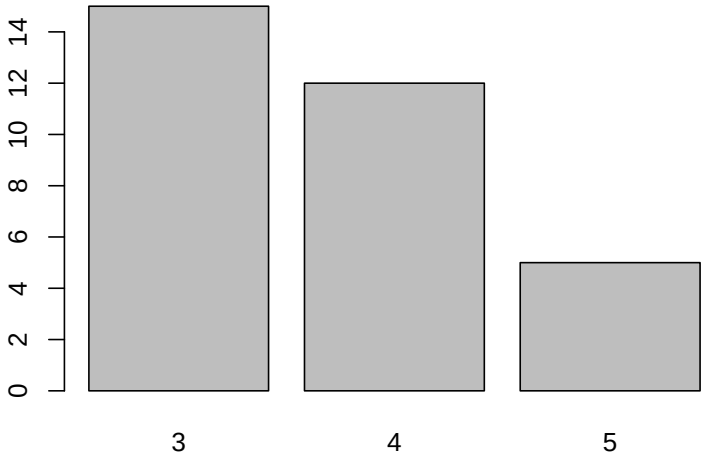
> boxplot(mtcars$wt, col="darkred")



III. Basic Plots

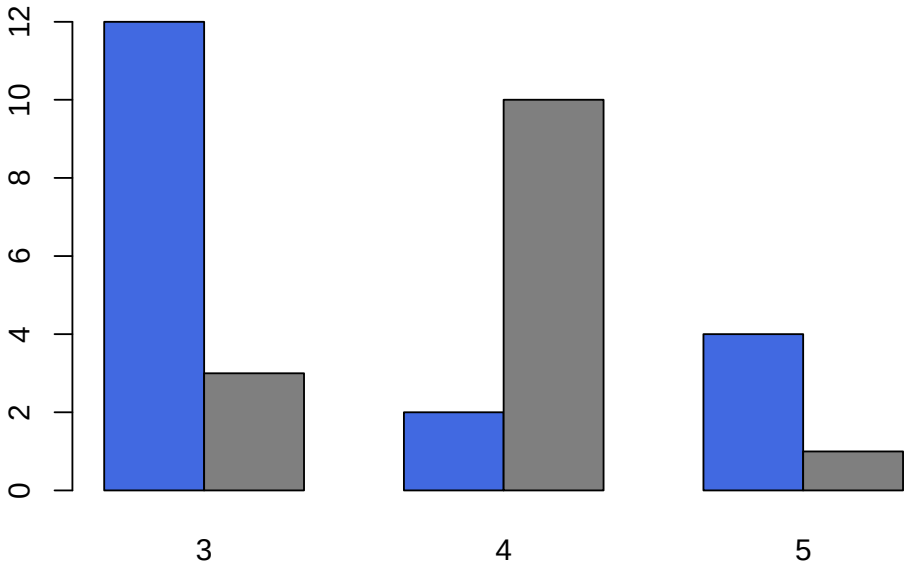
 # Barplot: the counts of a categorical value

```
> counts <- table(mtcars$gear)
> barplot(counts)
```



 # Advanced Barplot: by groups

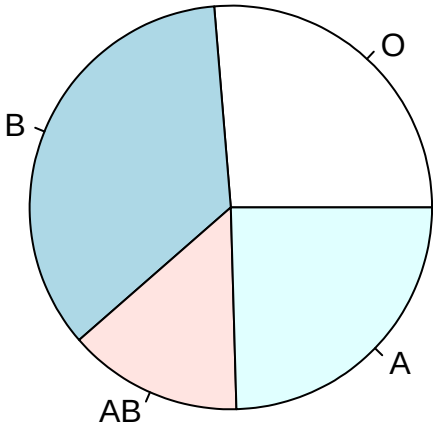
```
> counts <- table(mtcars$vs,mtcars$gear)
> barplot(counts,col=c("royalblue","grey50"),beside=T)
```



III. Basic Plots

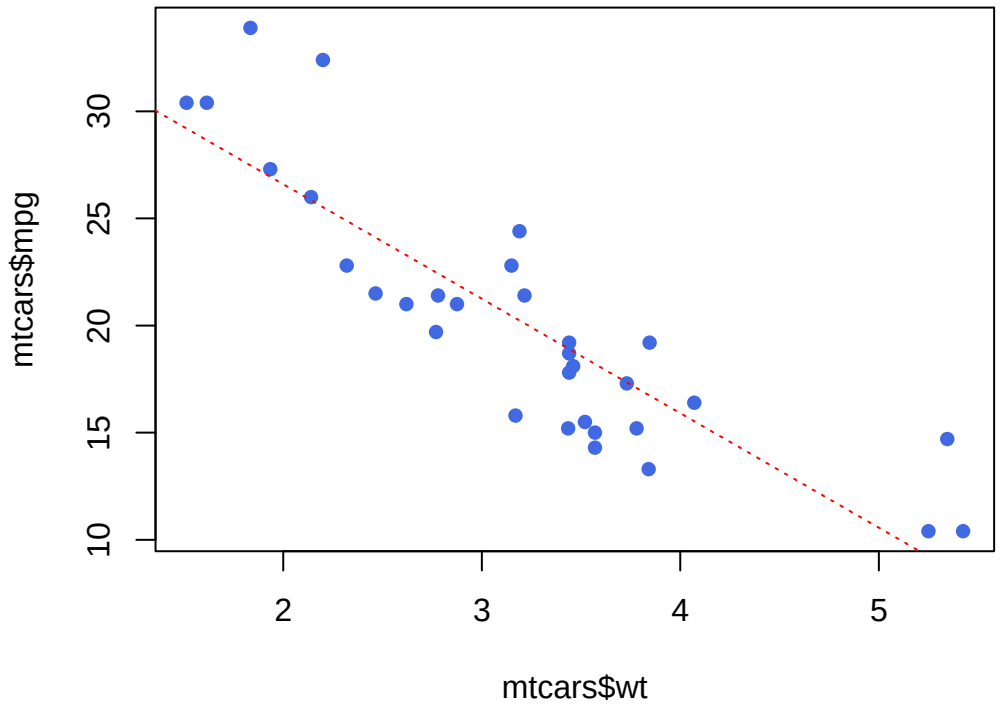
 # Pie Chart: plot proportion

```
> slices <- c(15,20,8,14)
> lbls <- c("O","B","AB","A")
> pie(slices,labels=lbls)
```



 # Scatter plot: check the correlations between to continuous variables

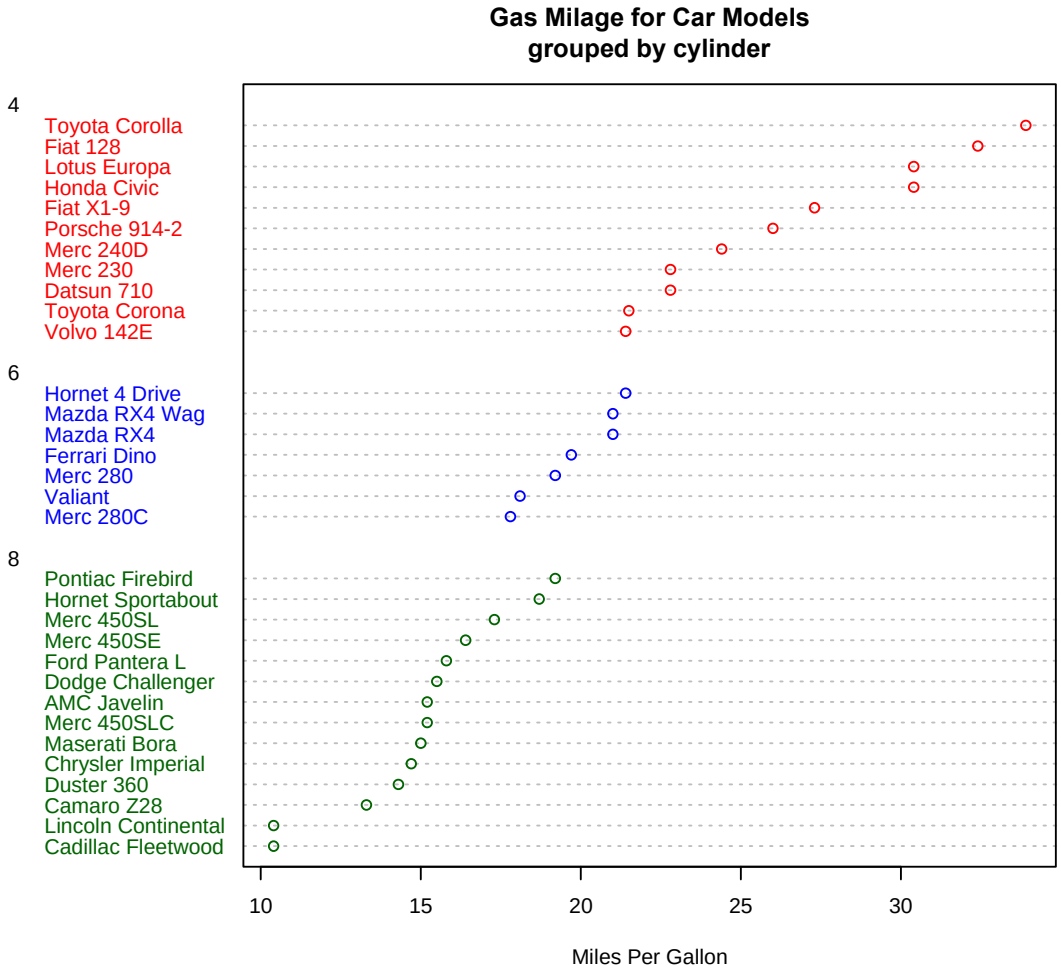
```
> plot(mtcars$mpg~mtcars$wt,pch=16,col="royalblue")
> abline(lm(mtcars$mpg~mtcars$wt),col="red",lty=3)
```



III. Basic Plots

 # Dotplot

```
> # Dotplot: Grouped Sorted and Colored
> # Sort by mpg, group and color by cylinder
> x <- mtcars[order(mtcars$mpg),] # sort by mpg
> x$cyl <- factor(x$cyl) # it must be a factor
> x$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$cyl,
+   main="Gas Milage for Car Models\ngrouped by cylinder",
+    xlab="Miles Per Gallon", gcolor="black", color=x$color)
```



Code from: https://www.statmethods.net/graphs/dot.html

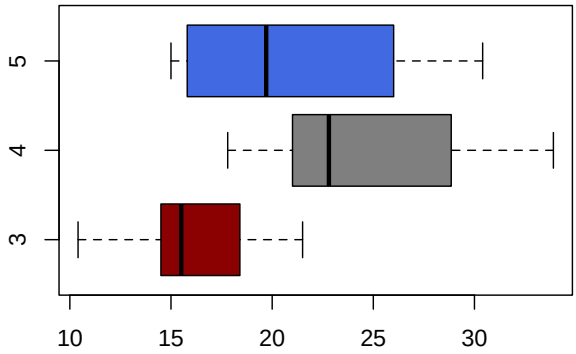
III. Advanced Plots

You can use different R packages for plotting
For example, “ggplot”,”ggplot2”,”plotly”
If you want to install these packages, simply type
> install.packages("plotly")

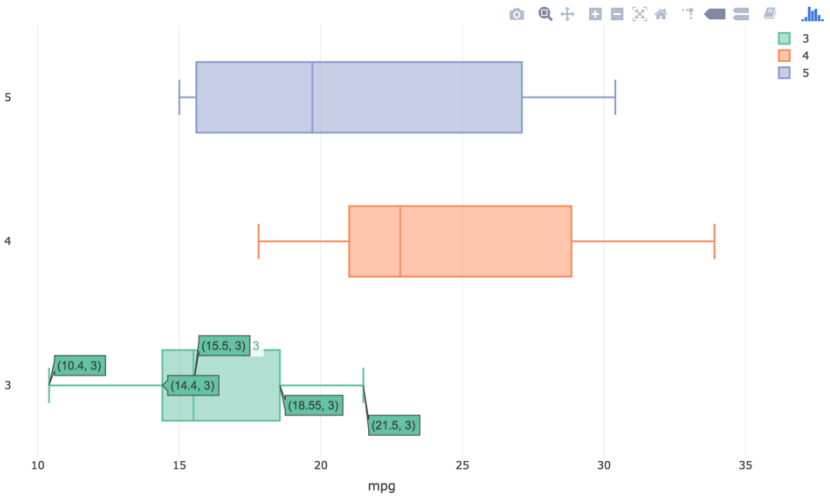
#(R will ask you to select a CRAN mirror for use, just choose a location close by)
57: Taiwan (Chungli) [https]
#17: China (Beijing) [https] 18: China (Hefei) [https]
#19: China (Guangzhou) [https] 20: China (Lanzhou) [https]
#21: China (Shanghai) [https]
#43: Japan (Tokyo) [https] 44: Japan (Yonezawa) [https]

After downloading this package, you need to type
> library("plotly")

Use simple command
>boxplot(mtcars$mpg~factor(mtcars$gear),horizontal=TRUE,col=c("darkred","grey50","royalblue"))



Use plotly
> p <- plot_ly(mtcars, x = ~mpg, color = ~factor(gear), type = "box")
> p



https://plot.ly/feed/

IV. Summary Statistics

```
# Let's use "mtcars" dataset again
# Calculate the average mpg in cars with gear 3, 4, and 5
# First we can use an easier command but slower way to do it
> mean(mtcars[mtcars$gear==3,][,1])
[1] 16.10667
> mean(mtcars[mtcars$gear==4,][,1])
[1] 24.53333
> mean(mtcars[mtcars$gear==5,][,1])
[1] 21.38

# Similarly, you can calculate other summary statistics
# Try to replace "mean" to "sd", "length", "max", "min", "sum"

# There is a smarter way to do the same stuff
# Using "aggregate" function to find summary statistics in different categories

> aggregate(mpg~gear,data=mtcars,mean)
  gear      mpg
1    3 16.10667
2    4 24.53333
3    5 21.38000

# the "mean" here is a function, you can also replace it to to "sd", "length", "max", "min", "sum"

## you can also order your result by mpg
> a <- aggregate(mpg~gear,data=mtcars,mean)
> a[order(a[,2]),]
  gear      mpg
1    3 16.10667
3    5 21.38000
2    4 24.53333

# You can  order your data the other way around, just add
> a[order(-a[,2]),]
  gear      mpg
2    4 24.53333
3    5 21.38000
1    3 16.10667
```

```
# Let’s generate a summary table
# install this package “gtsummary”

> install.packages("gtsummary")

# make sure all the variables are defined by their types

> mtcars$vs<-as.factor(mtcars$vs)
> mtcars$am<-as.factor(mtcars$am)
> mtcars$carb <- as.numeric(mtcars$carb)

> mtcars %>% tbl_summary
# by default, “quantitative variables” will be shown as median (IQR); “categorical variables” will be shown as n (%)

# you can do the summary statistics by groups
> mtcars %>% tbl_summary(by=am)

# you can change the settings
> mtcars %>%
 tbl_summary(by=am,
                    statistic = list(all_continuous() ~ "{mean} ({sd})",
                                     all_categorical() ~ "{n} / {N} ({p}%)"),
                    digits = all_continuous() ~ 2,
                    missing = "no") %>%
  add_n()

# https://www.danieldsjoberg.com/gtsummary/articles/gallery.html
```

V. Vector/List/Matrix/Array/Data frame

- A **vector** is a collection of cells with a fixed size and all cells hold the same data type (numeric, strings, etc).
- A **list** can contain cells of different types.
- A **matrix** is a two-dimensional vector. In a matrix, all columns must have the **same** data type (numeric, strings, etc) and the same length. There is no difference between a matrix and a 2D array (matrices).
- An **array** is a vector with one or more dimensions.
- In a **data frame** (or just think it's an Excel table), different columns can have **different** data input (numeric, strings, factor, etc).

```
# Let's generate a 4 by 5 array that contains number 1 to 20.
> x <- array(1:20,dim=c(4,5))
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    5    9   13   17
[2,]    2    6   10   14   18
[3,]    3    7   11   15   19
[4,]    4    8   12   16   20

# We can try another method to generate it.
> y  <- matrix(1:20,nrow=4,byrow=FALSE)
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    5    9   13   17
[2,]    2    6   10   14   18
[3,]    3    7   11   15   19
[4,]    4    8   12   16   20


> y  <- matrix(1:20,nrow=4,byrow=TRUE)
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    2    3    4    5
[2,]    6    7    8    9   10
[3,]   11   12   13   14   15
[4,]   16   17   18   19   20

## notice the "fill order" of numbers has been changed
```

V. Vector/List/Matrix/Array/Data frame

Let‘s generate a 4 by 5 array that contains number 1 to 20.

```
> x <- array(1:20,dim=c(4,5))
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    5    9   13   17
[2,]    2    6   10   14   18
[3,]    3    7   11   15   19
[4,]    4    8   12   16   20
```

If we have a smaller 3 by 2 array

```
> i <- array(c(1:3,3:1),dim=c(3,2))
     [,1] [,2]
[1,]    1    3
[2,]    2    2
[3,]    3    1
```

Let‘s find the numbers in “x“ based on the “i“ array

```
> x[i]
[1] 9 6 3
```

You can replace these numbers (x[i]) into any numbers you want, let‘s try to assign “0“

```
> x[i] <- 0
> x
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    5    0   13   17
[2,]    2    0   10   14   18
[3,]    0    7   11   15   19
[4,]    4    8   12   16   20
```

Let‘s try to assign all the numbers > 15 as 25

```
> x[x>15] <- 25
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    5    0   13   25
[2,]    2    0   10   14   25
[3,]    0    7   11   15   25
[4,]    4    8   12   25   25
```

V. Vector/List/Matrix/Array/Data frame

```
# In a matrix, find all columns that listed in a vector
# Let‘s use “mtcars“, say you only want to get columns: “wt“,“am“,“gear“,“carb“
# Here is the straightforward command:

> mtcars[c("wt","am","gear","carb")]

# Or you can make a list contains these names first then extract these columns

> a1 <- c("wt","am","gear","carb")
> mtcars[,colnames(mtcars)%in%a1]

# This “%in%“ is very useful if you have a long list of columns you want to extract

# Similarly, you can also select specific rows
# Let‘s say you are interested in “Maserati Bora“, “Ferrari Dino“, “Porsche 914-2“

>mtcars[rownames(mtcars)%in%c("Maserati Bora","Ferrari Dino","Porsche 914-2"),]


###
# Combine two matrix to increase rows- make sure they have same colnames
> x <- matrix(1:9,nrow=3)
> x1 <- matrix(9:1,nrow=3)
> rbind(x,x1)

# Combine two matrix to increase columns- make sure they have same number or rows
> cbind(x,x1)

###
# Merge two files together based on a specific column
> x <- matrix(1:9,nrow=3)
> x1 <- matrix(1:9,nrow=3)
> colnames(x) <- c("A","B","C")
> colnames(x1) <- c("A","D","E")

> merge(x,x1,by.x="A",by.y="A")
  A B C D E
1 1 4 7 4 7
2 2 5 8 5 8
3 3 6 9 6 9
```

Save/read in files

```
# Read .txt files
> Data <- read.delim("xxx.txt",header=T)
> Data <- read.table("xxx.txt",sep="\t",head=T)

# Read .csv files
> Data <- read.csv("xxx.csv",head=T)

# Load a .Rdata
>load("xxx.RData")

### write a .txt file
> write.table("xxx.txt",row.names=F,col.names=T,sep="\t",quote=F)

### write a .csv file
> write.csv("xxx.csv",row.names=F)

### save a .RData
> save(list=" yy", file="xxx.RData")


## let's read in this file: "practice0315.csv"
data <- read.csv("practice0315.csv",head=T)

> str(data)
'data.frame':    90 obs. of  9 variables:
 $ Gender       : chr  "Female" "Female" "Female" "Male" ...  # gender
 $ Height       : num  168 158 167 165 175 ...                # body height
 $ Weight       : num  52 75 60 60 85 53 53 54 54 41 ...      # body weight
 $ BT           : chr  "B" "A" "A" "AB" ...                   # blood pressure
 $ pet          : chr  "both" "cat" "both" "both" ...         # prefer cat or dog or both or neither
 $ Game         : int  0 1 1 1 1 0 1 1 1 0 ...                # play video game or not (1: yes)
 $ Short.sighted: int  1 0 1 1 1 1 1 1 1 1 ...                # short-sighted or not (1: yes)
 $ Computer_game: num  4 5 5 8 4 8 8 6 8 8 ...                # time spent on computer games (hrs)
 $ drink        : int  2 4 2 2 2 2 2 2 2 1 ...                # cups of sugar drinks per week
```

Practice time!

1. What is the dimension (how many rows, columns) in this data matrice?

2. What is the female proportion in this dataset?

3. Generate a new variable called BMI (BMI = kg/m^2), make sure you add this extra new column back to the dataset.

4. Which blood type has the highest frequency in the dataset?

5. How many subjects loves cat in this dataset?

6. Which gender likes pets more?

7. Make a boxplot of height by blood type in female subjects.

8. Please summarize the hours students spent on computer games per day. Do you find anything strange about this data?

9. Please assign ”NA” to the strange value. What is the mean “hours students spent on computer games per day” before and after you assign the new value?