KIT306/606 Tutorial 2

Student ID	Name

The following tutorial work should be completed by tutorial 3 (week4).

- **Data Handling**
- Data Frame with vector (using c – combine values into a vector or list)

```
> id<-c(1,2,3,4,5)
> gender<-c("F","F","M","M","M")
> height<-c(160,165,170,175,180)
> weight<-c(50,65,60,55,70)
> MYDATA<-data.frame(ID=id,GENDER=gender,HEIGHT=height,WEIGHT=weight)
> MYDATA
  ID GENDER HEIGHT WEIGHT
1  1      F    160     50
2  2      F    165     65
3  3      M    170     60
4  4      M    175     55
5  5      M    180     70
```

You can show the above script as the following figure:

ID	GENDER	HEIGHT	WEIGHT
1	F	160	50
2	F	165	65
3	M	170	60
4	M	175	55
5	M	180	70

ID	GENDER	HEIGHT	WEIGHT
1	F	160	50
2	F	165	65
3	M	170	60
4	M	175	55
5	M	180	70

- Data Frame with txt and csv file

Download grade.txt and grade.csv files from MyLo, and save it to the folder 'R' in your desktop

```
> MYdatatxt<-read.table("~/Desktop/R/grade.txt",header=T)
>
> MYdatatxt
  studentid    Name KIT101 KIT102 KIT103
1         1   James      8      7      2
2         2 Richard      8      5      7
3         3    John      9      4      9
4         4    Nick      4      5      7
5         5  Andrew      5      8      9
6         6    Alex      5      9      6
7         7     Leo      2      7     10
```

```
> Mydatacsv<-read.csv("~/Desktop/R/grade.csv",head=T)
> Mydatacsv
  studentid    Name KIT101 KIT102 KIT103
1         1   James      8      7      2
2         2 Richard      8      5      7
3         3    John      9      4      9
4         4    Nick      4      5      7
5         5  Andrew      5      8      9
6         6    Alex      5      9      6
7         7     Leo      2      7     10
```

- Let's have a look at the summary of the dataset

(summary is a generic function used to produce result summaries of the results of various model fitting functions.)

```
> summary(Mydatacsv)
   studentid        Name       KIT101          KIT102          KIT103
 Min.   :1.0   Alex   :1   Min.   :2.000   Min.   :4.000   Min.   : 2.000
 1st Qu.:2.5   Andrew :1   1st Qu.:4.500   1st Qu.:5.000   1st Qu.: 6.500
 Median :4.0   James  :1   Median :5.000   Median :7.000   Median : 7.000
 Mean   :4.0   John   :1   Mean   :5.857   Mean   :6.429   Mean   : 7.143
 3rd Qu.:5.5   Leo    :1   3rd Qu.:8.000   3rd Qu.:7.500   3rd Qu.: 9.000
 Max.   :7.0   Nick   :1   Max.   :9.000   Max.   :9.000   Max.   :10.000
               Richard:1
```

You can find the summary of the specific column.

```
> summary(Mydatacsv$KIT101)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  2.000   4.500   5.000   5.857   8.000   9.000
```

The database is attached to the **R** search path. This means that R when evaluating a variable searches the database; so simply giving their names can access objects in the database.

```
> attach(Mydatacsv)
> names(Mydatacsv)
[1] "studentid" "Name"      "KIT101"    "KIT102"    "KIT103"
```

The attach() and names() function in R can be used to make objects within data frames accessible in R with fewer keystrokes

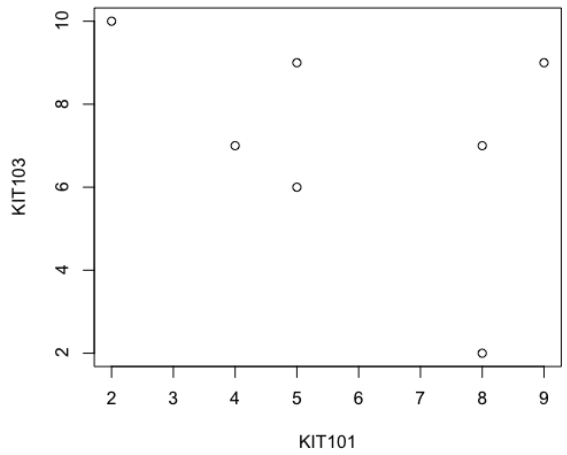
```
> summary(KIT101)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  2.000   4.500   5.000   5.857   8.000   9.000
```

- **Scatter plot**

A scatter plot is a type of mathematical diagram using Cartesian coordinates to display values for **two variables for a set of data**.

There are many ways to create a scatterplot in R. The basic function is **plot(x,y)**, where x and y are numeric vectors denoting the (x,y) points to plot.

```
> plot(KIT101,KIT103)
```

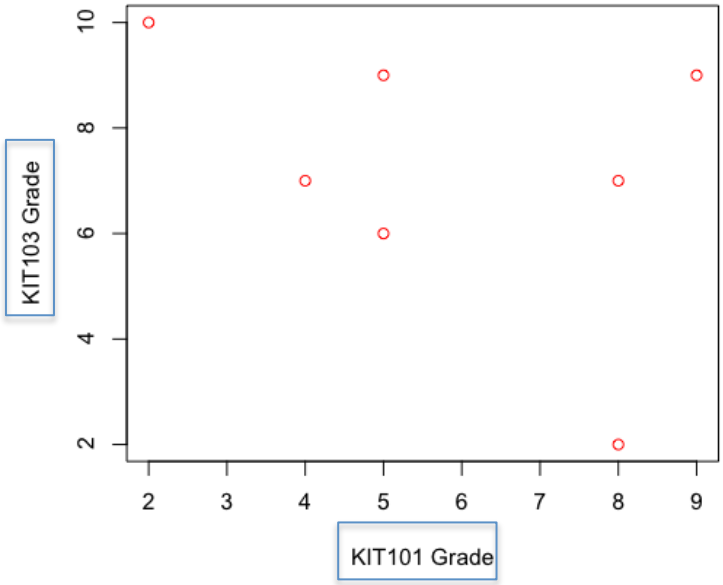


You can also choose the colour for scatters. Try it and check the changes.

```
> plot(KIT101,KIT103,col="red")
> plot(KIT101,KIT103,col="green")
> plot(KIT101,KIT103,col="blue")
```

Previously, the labels are based on the column. You can set up the label for x-axis and y-axis manually.

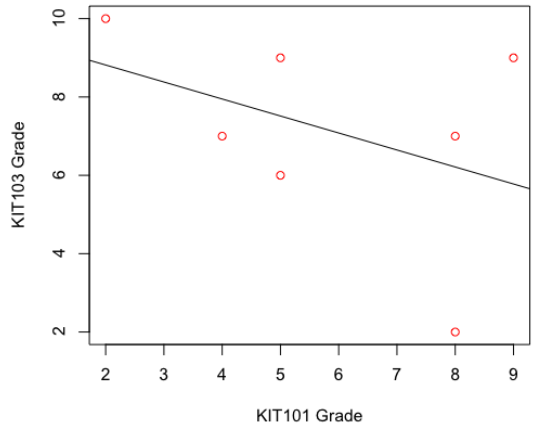
```
> plot(KIT101,KIT103,col="red",xlab="KIT101 Grade", ylab="KIT103 Grade")
```



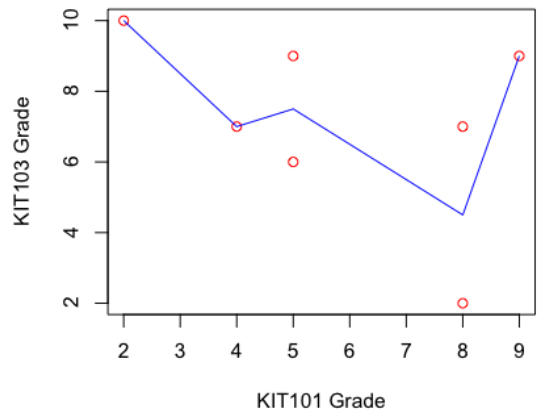
You can add the fit lines

- abline with lm function allows you to draw regression line (y~x)
- lines with lowess allows you to draw lowess line (x,y)

```
> abline(lm(KIT103~KIT101))
```




```
> lines(lowess(KIT101,KIT103),col="blue")
```



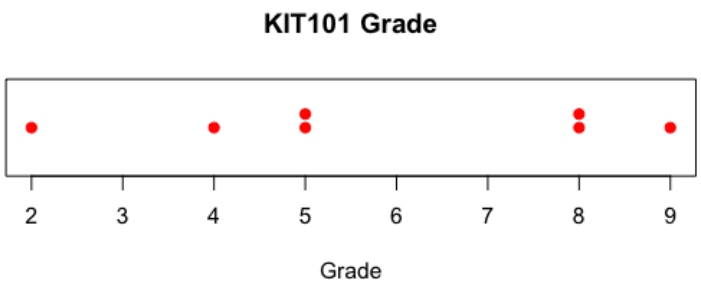
There are several types of plotting symbols: **pch**.
You can download the plotting list from the **plot_char.gif** from MyLo.
Try the following commands and check the changes.

```
> plot(KIT101,KIT103,col="red",xlab="KIT101 Grade", ylab="KIT103 Grade",pch=16)
> plot(KIT101,KIT103,col="red",xlab="KIT101 Grade", ylab="KIT103 Grade",pch=17)
> plot(KIT101,KIT103,col="red",xlab="KIT101 Grade", ylab="KIT103 Grade",pch=22)
```

- **Strip dot Plot**

stripchart() function allows you to create a dotplot.

```
> stripchart(KIT101,
+ method="stack",
+ pch=19,
+ main="KIT101 Grade",
+ col="red",
+ offset=0.5,
+ xlab="Grade Score"
+ )
```



The detailed argument explanation can be found as follows:

- offset: when stacking is used, points are stacked this many line-heights (symbol widths) apart.
- More arguments can be checked using help(stripchart)

• Histogram

A **histogram** is a graphical representation of the distribution of numerical data. It is an estimate of the probability distribution of a **continuous variable** (quantitative variable).

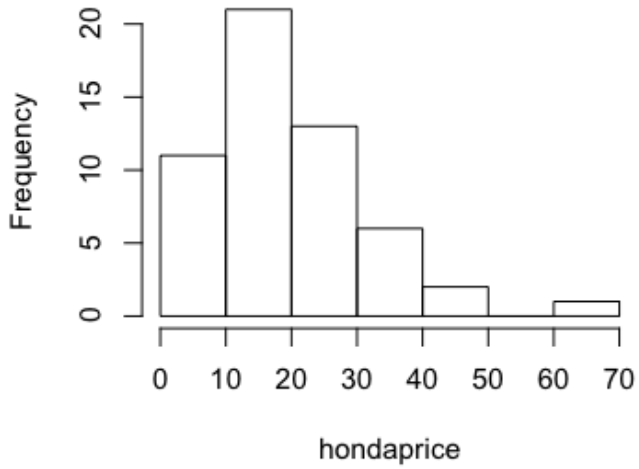
First, let's make a data with continuous variable.

```
> hondaprice<-
c(7.4,8.0,8.3,8.4,8.4,8.6,9.0,9.1,9.2,9.8,10.0,10.1,10.3,10.9,11.1,11.3,11.6,11.8,12.1,12.2,13.9,14.9,15.6,15.7,15.9,15.9,16.3,18.2,18.4,18.5,18.8,19.3,20.2,20.7,20.8,20.9,21.5,23.7,24.4,26.1,26.3,26.7,28.0,29.5,30.0,33.9,34.3,34.7,35.2,36.1,37.7,40.1,47.9,61.9)
```

Now, you can use hist function to draw a histogram.
The generic function hist computes a histogram of the given data values. If plot = TRUE, the resulting object of class "histogram" is plotted by plot.histogram, before it is returned.

Make a histogram with hondaprice data

```
> hist(hondaprice)
```



Breaks is
one of:

- a vector giving the breakpoints between histogram cells,
- a function to compute the vector of breakpoints,
- a single number giving the number of cells for the histogram,
- a character string naming an algorithm to compute the number of cells
- a function to compute the number of cells.

```
> hist(hondaprice,breaks=10)
```

You can try the sequence function that generates regular sequence. You can define the starting and (maximal) end values of the sequences. The last value is bin width for the histogram.

```
> hist(hondaprice,breaks=seq(0,70,10))
> hist(hondaprice,breaks=seq(0,70,5))
> hist(hondaprice,breaks=seq(0,70,2))
> hist(hondaprice,breaks=seq(0,70,10), main="Breaks=10")
```

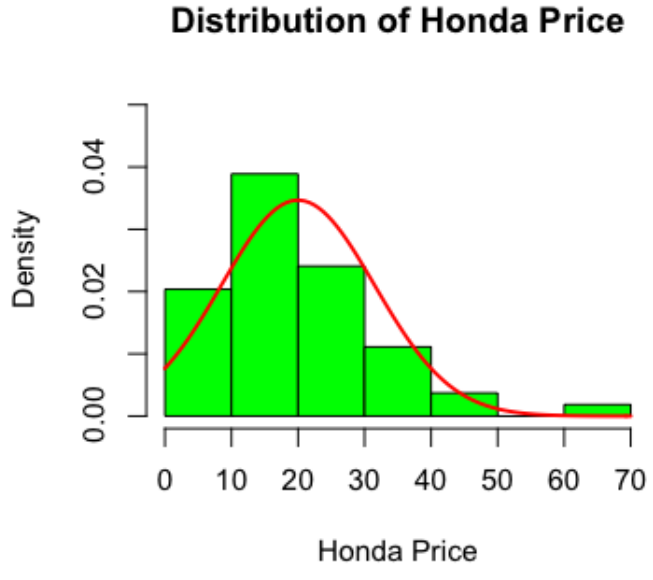
The above commands are for drawing frequency but you can also make the histogram with density.

```
> hist(hondaprice, main="Frequency")
> hist(hondaprice, main="Density",freq=FALSE)
```

The distribution curve can be provided with density.

```
> hist(hondaprice,
+ freq=FALSE,
+ xlab="Honda Price",
+ main="Distribution of Honda Price",
+ col="green",
+ xlim=c(0,70),
+ ylim=c(0,0.05)
+ )
> curve(dnorm(x,
+ mean=mean(hondaprice),
+ sd=sd(hondaprice)),
+ add=TRUE,
+ col="red",
+ lwd=2)
```

- dnorm: Density, distribution function, quantile function and random generation for the normal distribution with mean equal to mean and standard deviation equal to sd.
- mean: This function is for the (trimmed) arithmetic mean.
- sd: This function computes the standard deviation of the values in x
- lwd: line width



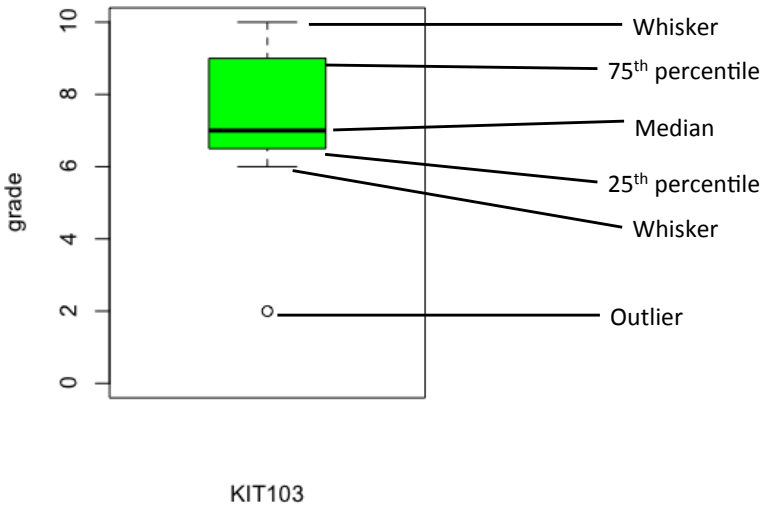
- **Box Plot**

Boxplot is a convenient way of graphically depicting groups of numerical data through their quartiles. Box plots may also have lines extending vertically from the boxes (whiskers) indicating variability outside the upper and lower quartiles, hence the terms box-and-whisker plot and box-and-whisker diagram. Outliers may be plotted as individual points.

```
> Mydatatxt<-read.table("~/Desktop/R/grade.txt",header=T)
> grade<-Mydatatxt
```

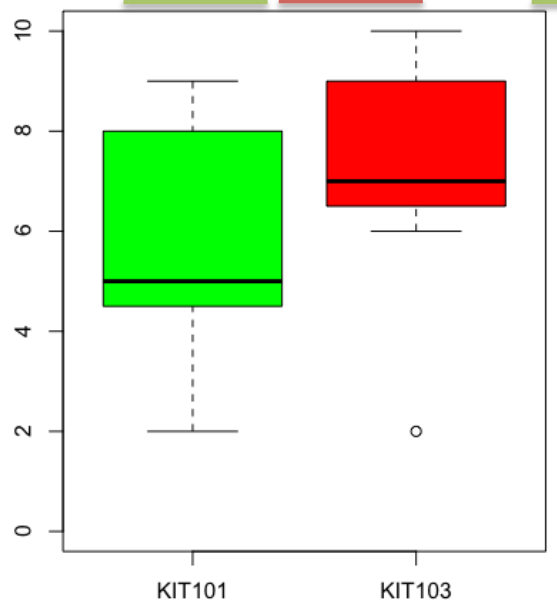
You can draw the box-and-whisker plot of given values using boxplot function.

```
> boxplot(grade$KIT103,col="green",ylim=c(0,10),xlab="KIT103",ylab="grade")
```



You can draw multiple boxplots as below:

```
> boxplot(grade$KIT101,grade$KIT103,names=c("KIT101","KIT103"),col=c("green","red"),ylim=c(0,10))
```



Tutorial Questions

1. Convert the following data (in the table) into two csv files: diet_pills.csv and fake_pills.csv.

Diet Pills					Fake Pills				
ID	**Name**	**Previous Weight (kg)**	**Current Weight (kg)**	**Changes**	**ID**	**Name**	**Previous Weight (kg)**	**Current Weight (kg)**	**Changes**
1	Andrew	96	99	3	1	Alex	74	70	-4
2	Brian	65	63	-2	2	Ben	61	62	1
3	Christian	75	74	-1	3	Daniel	70	67	-3
4	David	58	58	0	4	Elyse	80	75	-5
5	Elizabeth	59	60	1	5	George	71	69	-2
6	Fred	66	64	-2	6	Michael	68	60	-8

And, define diet_pill.csv **file as a variable** Dietcsv **and** fake_pill.csv **as a variable** Fakecsv**.**

2. Make a function drawhist that draw a histogram. The histogram should have the following functions:

1. There are two types of histogram you should draw.
   1) diethist and 2)fakehist
2. Check the type is diethist or fakehist
3. If the type is diethist, it should draw a green coloured histogram of the specified column (from **dietcsv file**) with the specified starting and end value, bin width, and label.
4. If the type is fakehist, it should draw a red coloured histogram of the specified column (from **fakecsv file**) with the specified starting and end value, bin width, and label.

```
drawhist<-function(type,data,from,to,bin,label){

}
```

Example command: **drawhist("diethist",dietcsv$Changes,-3,3,1,"diet_changes")**

3. Draw the following multiple scatterplots in one graph. The following shows the detail information of the scatterplots.

- X-axis represents previous weight, and y-axis represents current weight.
- Diet_pill group should be coloured in red (you should find the exact symbol from the list of plot characters)
- Fake_pill group should be coloured in blue (you should find the exact symbol from the list of plot characters)

- The range for x-axis and y-axis is 50-100.
- Legend should be located on top-left of the box (Tip: search legend function to use: help(legend))