

R for Medical Students

(Session 1)

Introduction

R is a free software environment for statistical computing and graphics. R provides a wide variety of statistical and graphical techniques and is highly extensible. This command-based package runs on Windows, Mac, and Linux platforms. You can download it free.

Once you have installed R, the next step is to install the RStudio (an integrated development environment (IDE) for R) to avoid the difficulty of using the R console alone.

The main objective of this course is to introduce the R statistical package and provide hands-on experience related to Medical Science using the RStudio. If interested in R, you can have comprehensive knowledge utilizing the website: <https://mran.microsoft.com/documents/getting-started>.

How to install the RStudio

To run the RStudio on your PC, you need to do the following two steps in the same order.

1. Install R (<https://cran.r-project.org/bin/windows/base/>)
2. Install R-Studio Desktop (Free version) (<https://www.rstudio.com/products/rstudio>)

Interface

RStudio interface consists of 4 sections as shown below. Each section provides separate facilities to make the R easy for you. That includes a **Console**, syntax-highlighting editor (**Script Area**) with the direct code execution facility, tools for plotting, history and debugging etc. Four sections in the RStudio altogether make an easy environment and increase the productivity of the beginners in R.

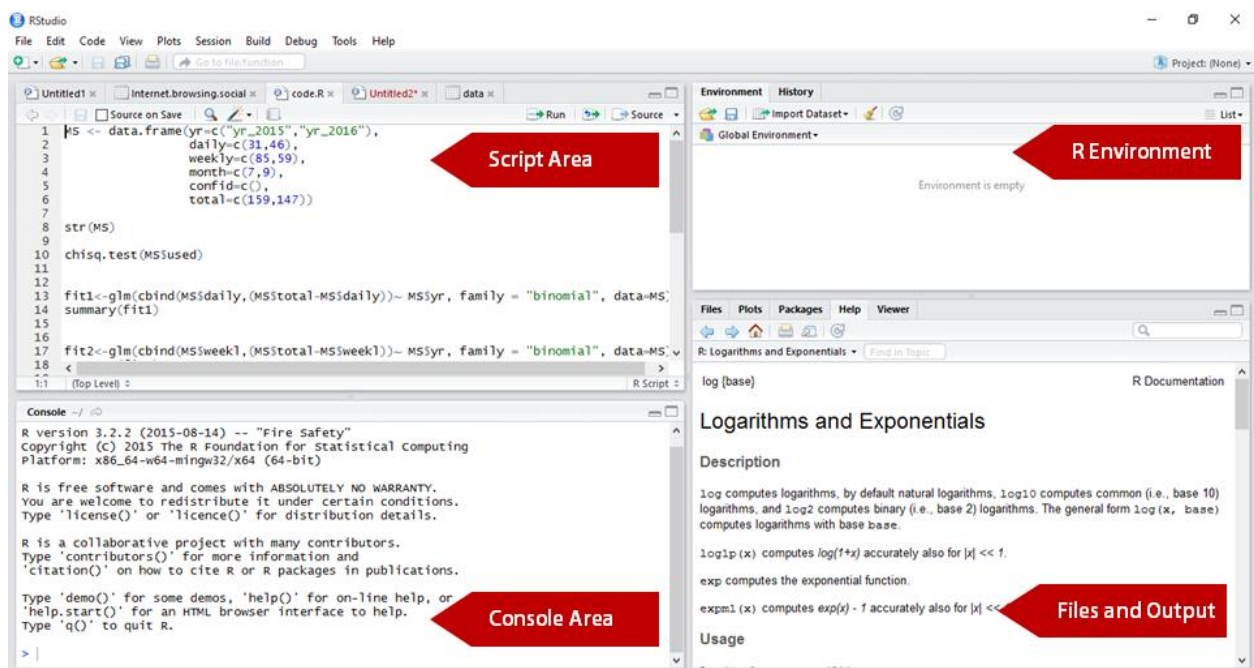


Figure: Graphical User Interface of RStudio

Organize your Project in RStudio

RStudio lets us create a separate folder to include current project files and, most importantly, provide us with direct access to files in the folder. So we can conveniently import data sets.

Steps to create new R Project

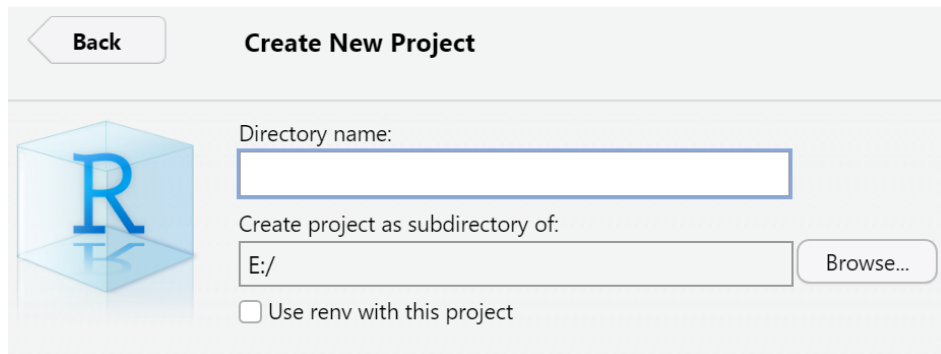
1. Open RStudio
2. File → New Project → New Directory → **Project Type** (New Project)
3. Provide Directory Name (*your group name*) and Location (Changing the default location is optional)
4. Press “Create Project”

Now RStudio switches to the new project you created, and it appears in the Files and Output section of the RStudio. You can copy the data files you want to analyze into this folder to keep the project's contents in this folder.

Activity 1- Create a new Project R project in RStudio using following information.

Directory name: My Project

Subdirectory: Change to Desktop



Important: Don't use “/” symbol in the project name (it separates folders)

R Online Tools

R online tools allow you to run R command directly from your browser without any software installation or dedicated hardware requirements. The user only needs a browser to study, teach analysis and share your work with others. Underneath are some examples of online tools.

JDoodle

<https://www.jdoodle.com/execute-r-online/>

RDRR.io

<https://rdrr.io/snippets/>

Rextester

https://rextester.com/l/r_online_compiler

RStudio Cloud

<https://rstudio.cloud/>

R Syntax

As in any language, R also has rules/protocols to follow. Since computers are machines, it is necessary to enter coding without error to obtain the correct output. The following basic syntax helps you write codes in R.

- To output text in R, use quotation (e.g., *"Hello World!"*).
- Numbers are not required quotes (e.g., 7).
- To calculate, use the numbers with Arithmetic Operator (e.g., $6 + 7$).
- To add comments, use # at the beginning (e.g., *# This is the r code for*).
- To assign values to variables, "=" operator or "<-" can be used (e.g., $a = 5$ OR $a <- 5$).

Mathematical calculations in R

We start with mathematical calculations in R to know how to write basic codings in R to have expected outputs. Firstly, a new **R Script** must be created by pressing **Ctrl+Shift+N** or clicking


icon  below the **File** tab and then selecting **R Script** from the dropdown list.

Activity 2- Type sample codes in a new R Script and press **Ctrl + Enter** or click **Run** button on the Script window to get the desired outputs of this arithmetic operations.

Arithmetic Operators	Code Samples	Outputs
+ Addition	$5 + 2$	7
- Subtraction	$5 - 2$	3
* Multiplication	$5 * 2$	10
/ Division	$5 / 2$	2.5
^ Exponentiation	$5 ^ 2$	25
%% Modulo	$5 \% 2$	1
< Less than	$5 < 8$	TRUE
> Greater than	$5 > 8$	FALSE
<= Less than or equal to	$5 <= 8$	TRUE
>= Greater than or equal to	$5 >= 8$	FALSE
== Equal to	$5 == 8$ $12 == 12$	FALSE TRUE
!= Not equal to	$5 != 8$	TRUE

There are three methods to save your R Script file:

1. Press **Ctrl + S** and give a file name

2. Go to **File** → **Save** and give a file name and
3. Use save button  in the toolbar

Activity 3 - Save the script in the project folder as **MyFirstRScript**.

Brackets in Mathematical operations

We use the parentheses () in the mathematical calculation in R. The equation $10 + (\frac{3-5}{2})^2$ is written with parentheses as `10+((3-5)/2)^2`. The square brackets [] are not used in mathematical expression. The use of square brackets in R is explained later.

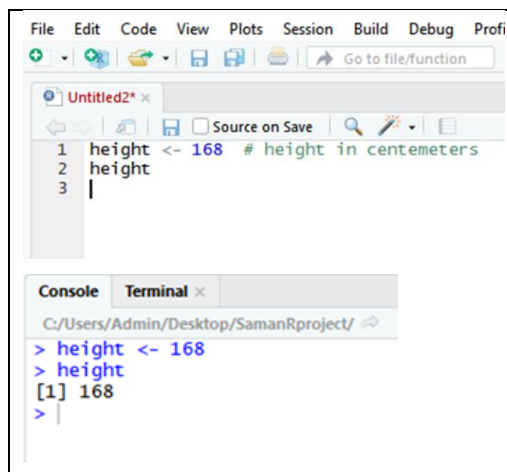
Variables in R

You can store values into a variable to access it later. To assign values to variables, “=” operator or “<-” can be used (e.g., `a = 5` OR “`a <- 5`”). Type `x <- 42` to store the number +42 in variable x. All mathematical operations can be performed on variable x (e.g., `x/2`)

R is case sensitive and there are basic rules to follow when naming variables.

Basic rules of naming variables

- case sensitive (eg. a vs A)
- cannot start with a number
- dot (.) or underscore (_) are the only characters allowed to use for naming but can not start with (.) (e.g., `.total_average` is permitted, and `_total_average` is not allowed).
- cannot be a name of a built in keyword (You can find full list of reserved words by typing `help(reserved)` or `?reserved` in the R)



1. Type `height <- 160` and press **Run** button or Press **Ctrl+Enter** to assign the number 160 to variable **height**. Anything after # mark is ignored by R so you can use # as explained to include additional text to your script.
2. Just type `height` and press **Run** button or Press **Ctrl+Enter** to see the value assigned the variable in the console.

Activity 4- Simple mathematical calculations.

Calculate the circumference of a circle having 5 cm radius (r).

$$Circumference = \pi d$$

Hint: d, diameter equals to 2r and (pi) equals to 3.14 or you can use pi directly as a variable.

Calculate the area of a circle having 5 cm radius (r).

$$Area = \pi r^2$$

Hint: (pi) equals to 3.14 or you can use pi directly as a variable.

Find the volume of a cylinder having a radius of 5 cm and a height of 12 cm.

Hint: $Volume = \pi r^2 h$
Take (pi) equals to 3.14

A cylindrical swimming pool is 4.5 feet high and has a diameter of 24 feet. If the pool is filled right up to the top edge how many cubic feet of water can the pool hold?

Hint: $Volume = \pi r^2 h$
Use the word “pi” in your r coding instead 3.14 for π . It gives the exact value for the “ π ” in R.

Activity 5: Calculation examples in Medicine

A bottle contains 156 ml of liquid. How much is that in dm^3 . (1 ml equals to $0.001 dm^3$)

$$Solution: 156 ml = 156 \times 0.001 dm^3 = 0.156 dm^3$$

Fahrenheit and Celsius are two units use to measure temperature. A patient's body temperature was measured as 102.2°F by a nurse. Express it in Celsius. Following is the formula to convert Fahrenheit to Celsius

$$C = \frac{5}{9} \times (F - 32)$$

$$Solution: C = \frac{5}{9} \times (102.2 - 32) = 39^\circ C$$

A microscopic image shows a cross section of a lump with diameter of 1.36 cm. Assuming it is in spherical shape, calculate volume and surface area of the lump. Use following equations for the calculation.

$$\text{radius } (r) = \frac{\text{diameter } (d)}{2}$$

$$\text{Volume } (V) = \frac{4\pi r^3}{3}$$

$$\text{Surface Area } (A) = 4\pi r^2$$

$$\text{Solution: } V = \frac{4 \times \pi \times (1.36/2)^3}{3} = 1.317 \text{ cm}^3$$

$$A = 4 \times \pi \times (1.36/2)^2 = 2.905 \text{ cm}^2$$

Note: you can directly use `pi` as π in the code. R have already assigned 3.14... to the variable `pi`

Calculate the pH of a 0.0048 M HCl solution

$$pH = -\log [H_3O^+] \text{ and } [H_3O^+] = 0.0048 \text{ M}$$

Hint: You can find correct syntax for **log** calculation in R using `?log` command. Correct syntax for this particular case is `-log10(0.0048)`. Here 10 represents the base value for the logarithm. Check whether you receive a value approximately to 2.319.

Calculate the Hydronium concentration of 8.34 pH solution

$$[H_3O^+] = 10^{-pH}$$

$$\text{Solution: } [H_3O^+] = 10^{-8.34} = 4.57 \times 10^{-9} \text{ (this equals to } 4.57\text{e-9)}$$

Note: When you do math in R (or anywhere), it is important to use parenthesis () in correct order. Otherwise correct mathematical expression will not execute. As an example,

$$y = 10 + \left(\frac{3-5}{2}\right)^2 \text{ this equal to } y <- 10 + ((3-5)/2)^2$$

BMI Calculation

Calculate the BMI of a person having 75 kg of weight and 160 cm height. BMI formula is given below

$$BMI = \frac{\text{Weight } (kg)}{\text{Height}^2 \text{ (m)}}$$

Code sample	What the sample codes mean
<pre># Assign variables weight <- 75 height <- 160 # in cm # Convert centimeters to meters heightInMeters = 160/100 # Calculate BMI bmi <- weight/(heightInMeters^2) # Print BMI bmi</pre>	<p>Assign values for variables</p> <p>Convert height in to meters and assign the new value to another variable.</p> <p>R expression for BMI calculation</p> <p>Print the BMI value in console</p>

Activity 6- Try to calculate the above BMI without defining intermediate variable *heightInMeters*.

Working with data

Previously we calculated the BMI of a single person. However, when it comes to practical scenario, there may be many of samples to calculate BMI. In that case we need to store multiple values in a single variable to calculate all BMIs at once. Vectors in R can be used for this purpose.

Vectors

Vector is a basic data structure in R and it can store list of data in the same type. We use the `c` function in R to define vectors. Below we define a two vectors (*groupWeight* and *groupHeight*) to store weight and height of five persons in order to calculate BMI of all at once.

Calculate BMI of a group	Code explanation
<pre># create a vector to store multiple values groupWeight <- c(75, 63, 70, 58, 60) groupHeight <- c(160, 152, 161, 154, 156) # Display the all values of the vector groupWeight groupHeight # Display the 1st value of the vector groupWeight groupWeight[1] # Convert centimeters to meters and groupHeightInMeters = groupHeight/100 # show groupHeightInMeters groupHeightInMeters</pre>	<p>R function <code>c()</code> combines multiple variables into a vector.</p> <p>Same mathematical operators used for variables can be used for vectors too. <code>groupHeight/100</code> divide all values assigned to vector by 100 (performed individually), thus converts centimeters to meters</p>
Calculate BMI of a group	Code explanation
<pre># Calculate BMI groupBMI <- groupWeight/(groupHeightInMeters^2) # Print groupBMI groupBMI</pre>	<p>R expression for the group calculation</p>

Activity 7- Applications in Vectors.

Deaths due to snake bites in Kurunegala district for three years are shown below.

Deaths from snake bites in Kurunegala District						
	January	March	May	July	September	November
2010	10	8	3	0	8	6
2011	8	8	2	2	5	9
2012	7	10	0	1	6	7

1. Create vectors for three years and name as Y2010, Y2011 and Y2012. Then assign 6 values to newly created vectors.
2. Find total death for all three years (monthwise) by adding previously created vectors. Vector name should be “totalDeaths”

	January	March	May	July	September	November
totalDeaths	25	26	5	3	19	22

3. Lately found that 2 deaths in January, 1 death in May and 3 deaths in September caused by another source. Create new vector named “correction” and assign values. Add value 0 if there’s nothing to reduce.

	January	March	May	July	September	November
correction	2	0	1	0	3	0

4. Now subtract values that need to remove from “totalDeaths” and assign new values to vector named “finalDeaths”) by adding previously created vectors. Vector name should be “totalDeaths”

	January	March	May	July	September	November
finalDeaths	23	26	4	3	16	22

Functions in R

Functions are created using the command **function()**. It is a block of code to perform a specific task. In general, function has an argument (or multiple arguments) as the input and pass it to the body of the function and finally gets the output.

For example, the following function (**my_func**) in R adds 1 to the number it receives as the argument. The last value in the function, in this case that stored in *y*, gets returned

```
my_func <- function(x){
  y <- x + 1
  y
}
```

To check this function in R, you can run the code below and it will return 6 as the output.

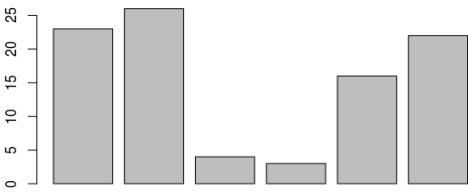
```
> my_func(5)
```

R consists of **in-built** functions which can be directly called in coding. Simple examples of in-built functions are `seq()`, `mean()`, `log()` etc. Sometimes, we have to write functions to perform specific tasks unavailable with **in-built** functions.

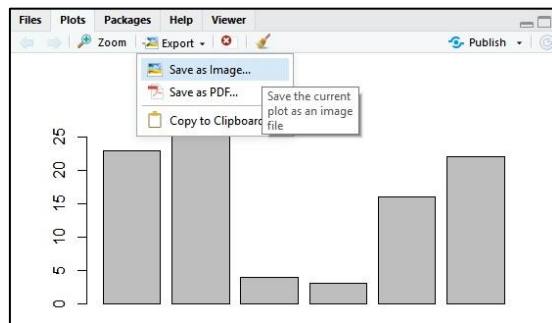
Activity 8- Write a function, named `bmi_func()` to calculate the BMI, give weight (kg) and height (cm) as the arguments to this function.

There are functions in R to draw graphs for data illustration. *barplot()* and *plot()* are commonly used functions for drawing bar plots and scatter/line graphs. You can access help files for these functions by entering `?barplot` and `?plot` commands, respectively. And also, R provides various plotting options with a vast range of packages (e.g., *ggplot2*, *Lattice*, *plotrix* ect.), that are ready to install to produce high-end-publication ready graphics.

Plotting in R

Bar plot	Output														
<pre># Barplot for finalDeaths barplot(finalDeaths)</pre> <p>Add names → mention names</p> <ol style="list-style-type: none"> 1. Weight → names?? 2. height 3. Bmi 	 <table border="1"> <caption>Data for Bar Plot</caption> <thead> <tr> <th>Category</th> <th>Frequency</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>23</td> </tr> <tr> <td>2</td> <td>25</td> </tr> <tr> <td>3</td> <td>4</td> </tr> <tr> <td>4</td> <td>3</td> </tr> <tr> <td>5</td> <td>16</td> </tr> <tr> <td>6</td> <td>22</td> </tr> </tbody> </table>	Category	Frequency	1	23	2	25	3	4	4	3	5	16	6	22
Category	Frequency														
1	23														
2	25														
3	4														
4	3														
5	16														
6	22														

To save the bar plot as a picture, Go to Graph window and choose **Export > Save as Image** as shown below.



To add the additional information (e.g., Add a title,color the bars) to the graph, Arguments need to be used in the code as shown below.

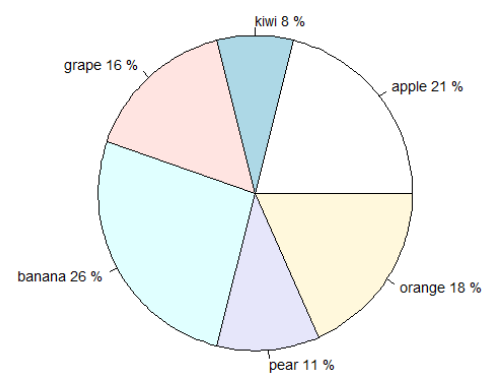
Bar plot	Output
<pre># Barplot for finalDeaths barplot(finalDeaths, xlab = "Months", ylab = "Deaths", main = "Snake bites deaths in Kurunegala District")</pre>	

You can draw a pie chart easily with pie() function in R. Let's consider the below dataset of 190 people who answered the question "What's your favorite fruit?"

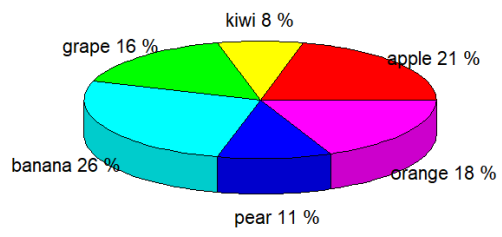
Fruit:	Apple	Kiwi	Grapes	Banana	Pears	Orange
People:	40	15	30	50	20	35

Pie Chart	Output
<pre># Define a vector for dataset FruitChoice <- c(apple=40, kiwi=15, grape=30, banana=50, pear=20, orange=35) # Create a pie chart pie(FruitChoice)</pre>	

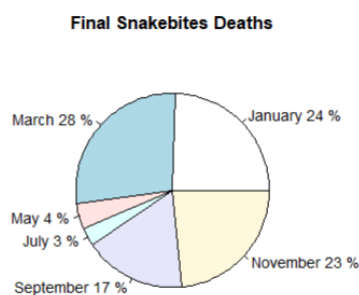
With the *lables* argument, we can label each pie slice of the chart with the percentage.

Pie Chart	Output														
<pre># Define a vector for dataset FruitChoice <- c(apple=40, kiwi=15, grape=30, banana=50, pear=20, orange=35) # calculate percentages pct <- round(FruitChoice /sum(FruitChoice)*100) # add percentages to labels lbls <- paste(names(FruitChoice), pct, "%") # Draw the pie chart pie(FruitChoice, labels=lbls)</pre>	 <table border="1"> <caption>Fruit Choice Data</caption> <thead> <tr> <th>Fruit</th> <th>Percentage</th> </tr> </thead> <tbody> <tr> <td>apple</td> <td>21 %</td> </tr> <tr> <td>kiwi</td> <td>8 %</td> </tr> <tr> <td>orange</td> <td>18 %</td> </tr> <tr> <td>pear</td> <td>11 %</td> </tr> <tr> <td>banana</td> <td>26 %</td> </tr> <tr> <td>grape</td> <td>16 %</td> </tr> </tbody> </table>	Fruit	Percentage	apple	21 %	kiwi	8 %	orange	18 %	pear	11 %	banana	26 %	grape	16 %
Fruit	Percentage														
apple	21 %														
kiwi	8 %														
orange	18 %														
pear	11 %														
banana	26 %														
grape	16 %														

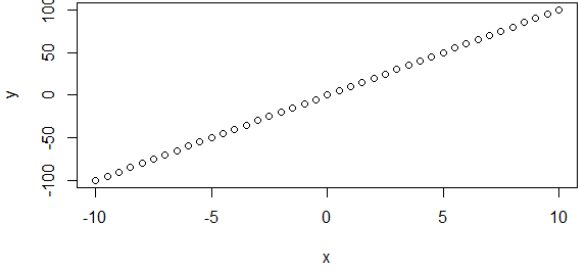
If you want to have a 3D pie chart, the additional package, *plotrix* needs to be installed.

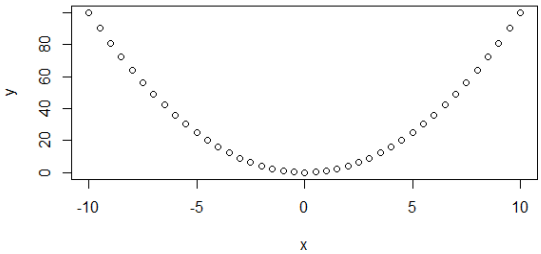
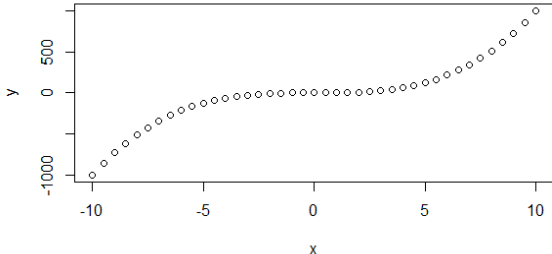
Pie Chart	Output														
<pre># Install new packages install.packages(plotrix) # Load the package plotrix library(plotrix) # Draw 3D pie chart Pie3D(FruitChoice, labels=lbls)</pre>	 <table border="1"> <caption>Fruit Choice Data</caption> <thead> <tr> <th>Fruit</th> <th>Percentage</th> </tr> </thead> <tbody> <tr> <td>apple</td> <td>21 %</td> </tr> <tr> <td>kiwi</td> <td>8 %</td> </tr> <tr> <td>orange</td> <td>18 %</td> </tr> <tr> <td>pear</td> <td>11 %</td> </tr> <tr> <td>banana</td> <td>26 %</td> </tr> <tr> <td>grape</td> <td>16 %</td> </tr> </tbody> </table>	Fruit	Percentage	apple	21 %	kiwi	8 %	orange	18 %	pear	11 %	banana	26 %	grape	16 %
Fruit	Percentage														
apple	21 %														
kiwi	8 %														
orange	18 %														
pear	11 %														
banana	26 %														
grape	16 %														

Activity 9- Draw a pie chart for snakebites finalDeaths as shown, also draw the same chart with pie3D function.



Hint : Modify the code shown in the example and use the argument **main** to add the title

Scatter plot	Output
<p>You can plot functions like $y = mx$ in R. First you need to generate sequence of numbers for x. The R function <code>seq()</code> can be used to generate a sequence of numbers.</p> <pre># Generate a sequence from -7 to 7: u <- seq(-7,7) u</pre> <pre># From -7 till 7, step=2: v <- seq(-7,7,2) v</pre> <p>Now you have a series of x values, which we can use for X axis and by same values, we can generate a function like $y = mx$</p> <pre># Assume m as 10 and let's make y=mx function x <- seq(-7,7) y <- 10*x</pre> <pre># Lets plot the graph plot(x,y)</pre>	<pre>-7 -6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6 7</pre> <pre>-7 -5 -3 -1 1 3 5 7</pre> 

Activity 10: Plotting	
Plot following functions in RStudio, assign values from -10 to 10 for x.	
 <p style="text-align: center;">$y = x^2$</p>	 <p style="text-align: center;">$y = x^3$</p>

Activity 11: Plotting

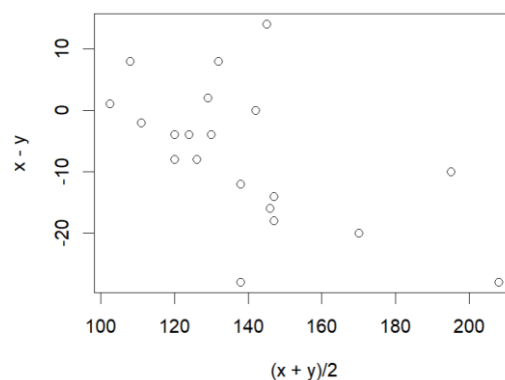
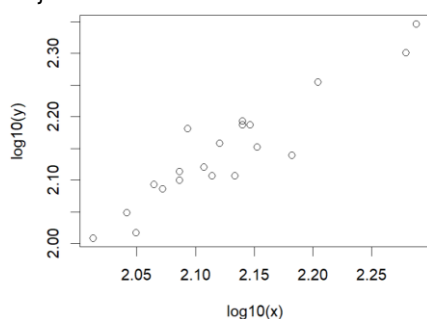
The following two rows of data (a sample of 20 people) represent the measurements of **systolic blood pressure (sbp)** made using a standard arm cuff and a finger monitor.

140, 110, 138, 124, 142, 112, 122, 128, 132, 160, 103, 136, 130, 152, 138, 190, 122, 118, 116, 194
 154, 112, 156, 152, 142, 104, 126, 132, 144, 180, 102, 128, 128, 138, 154, 200, 130, 122, 124, 222

Assign these values in rows to vectors *x* and *y* (you may copy and paste data to R script). It is possible to write functions in R that produce graphs. Use the below given function (`log_plot`) to draw the chart shown below (remember to call the function as `log_plot(x,y)` to get the output).

Then write function `altman_plot()` (difference of the sbp against the average) to have the output on the right (modify the `log_plot()` appropriately).

```
log_plot <- function(x, y){
  plot(log10(x), log10(y))
}
```



IF statement in R

if is a mostly used conditional statement in R. You should read the **if** statement as “If X is TRUE, do a thing”, adding an **else** statement, you can simply extend the logic as “If X is TRUE, do a thing, or else do something different”.

Combination of **if** and **else** statements let you to run sections of code, depending on a condition is either TRUE or FALSE as examples below.

Use of if and if else	Output
<pre># Example - use of if statement x <-100 y <- 30 if (x > y) { print("x is greater than y") } # Example 1 - use of else and if statements together x <-10 y <- 30 if (x > y) { print("x is greater than y") } else { print("x is not greater than y") }</pre>	<p>The code left-side produces the output in R console as shown below.</p> <p>[1] "x is greater than y"</p> <p>If "x is not greater than y", nothing will print in the console. You can avoid it adding else with if .</p> <p>Output for the codes left that combine if and else.</p> <p>[1] "x is not greater than y"</p>
<pre># Example 2 - use of else and if statements together x <-10 y <- 30 if (x > y) { print("x is greater than y") } else if (x == y) { print("x and y are equal") } else { print("x is not greater than y") }</pre>	<p>If we have more than two conditions to check, use the if else as coded in left. This code gives the below output.</p> <p>[1] "x is not greater than y"</p>

Loops in R

If we want to perform repetitive tasks in programming, we use what's known as loops for this purpose. There are two kinds of loops, namely, **for** and **while**.

Use of for and while loops	Output
<pre># Example - use of for loop. province <- c("Central", "Eastern", "Northern", "Southern", "Western", "North Western", "North Central", "Uva", "Sabbaragamuwa") for (i in province) { print(i) }</pre>	<p>Using the for loop, you can print names of the provinces stored in the vector, province.</p> <p>Output:</p> <pre>[1] "Central" [1] "Eastern" [1] "Northern" [1] "Southern" [1] "Western" [1] "North Western" [1] "North Central" [1] "Uva" [1] "Sabbaragamuwa"</pre>
<pre># Example - use of while loop i = 0 while (i < 20) { i = i + 1 if (i %% 5 == 0){ print(i) } }</pre>	<p>This coding shows you the use of while loop for printing only numbers up to 20 that can be divided by 5. Within the code if statement and %% operator have been used appropriately.</p> <p>Output:</p> <pre>[1] 5 [1] 10 [1] 15 [1] 20</pre>

Activity 12: if statement, for and while loop

In this activities, x and y refer to the vectors defined in the activity 11. The following coding prints values in the vector x and count negative value in vector x. Run this coding in R script and check the result. Modify this coding to print the values of the vector x-y and count the zero values, if any.

Then, write a coding to count zero of x-y vector using while loop (refer while loop coding example in his note and you can get the total number of elements in the vector using length() function, if required).

```
count = 0
for (i in x) {
  print(i)
  if (i < 0) {
    count =count + 1
  }
}
print(count)
```