Bioinformatics softwares (R/PLINK)

GBIO0002

Archana Bhardwaj

Outline

- **R**

- **Features of R**

- **Simple Data Visualization**

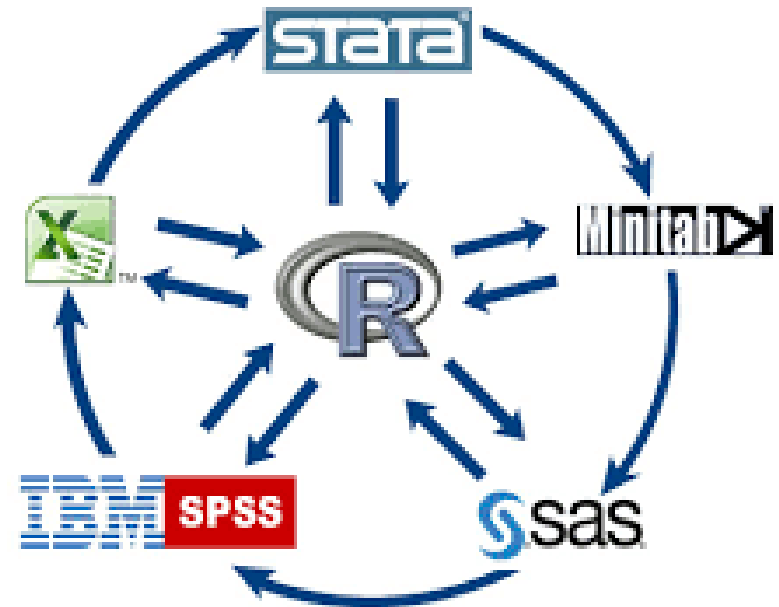- **Advance Data Visualization**

- **Data Transformation**

- **PLINK**

Introduction to R

A basic tutorial

Definition of R

- **R is a free software environment for statistical computing and grpahics**

- **R is considered to be one of the most widely used languages amongst statisticians , data miners, bioinfomaticians and others.**

- **Other commercial statistical packages are SPASS , SAS, Matlab**

Statistical language GUI

SAS

SPSS

Differences between Matlab and R

1. **Open source:**

   - **R is an open source while Matlab is not open source**

2. **Speed:**

   - **R is slower than Matlab**

3. **Functionality:**

   - **R is mainly used for statistical analysis.**

   - **Matlab is used for performing various engineering applications like image processing.**

4. **Ease of use :**

   - **R follows programming lagugage syntax.**

   - **Matlab is easy to program as multiple toolbox are available**

R GUI
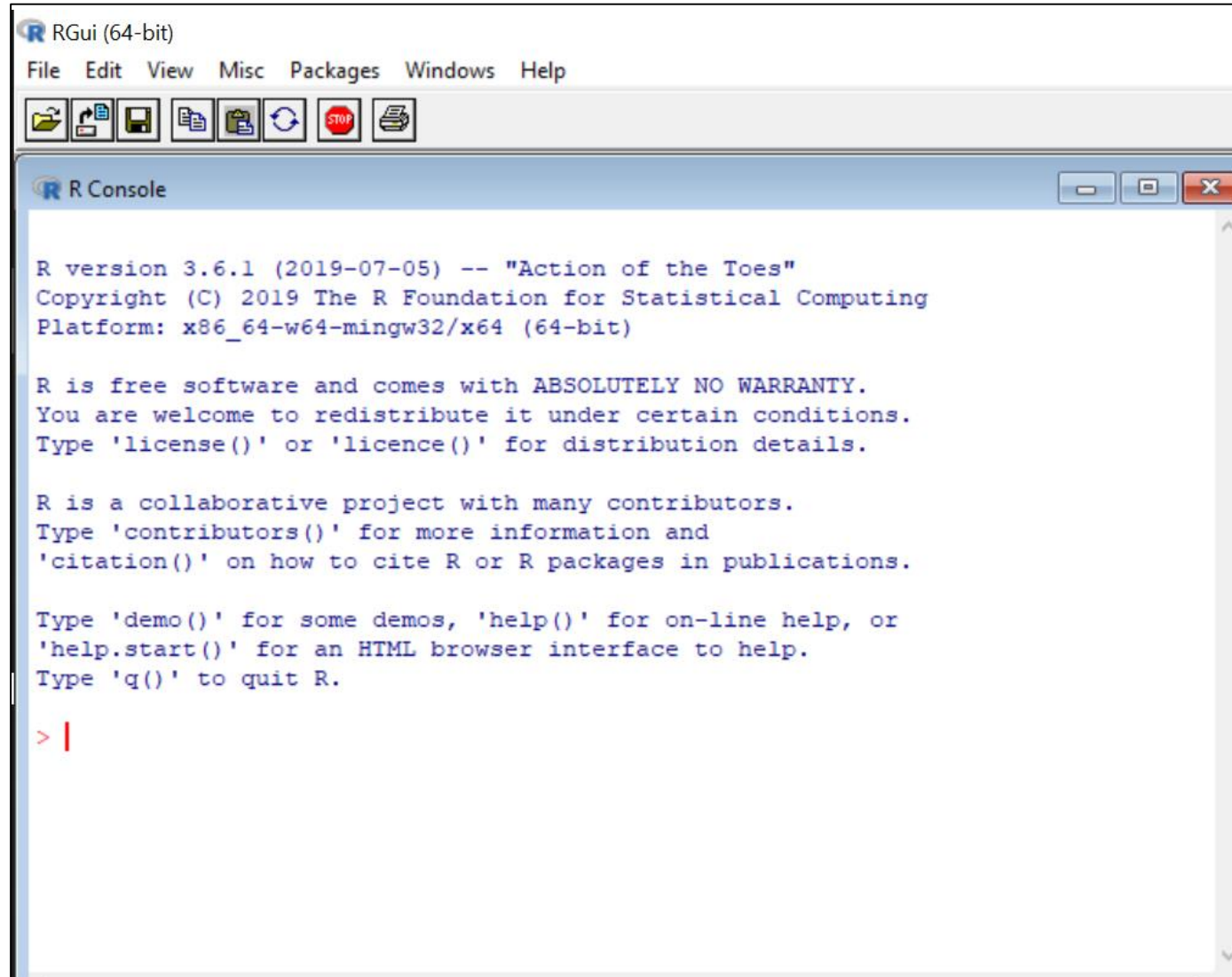
Less fancy and no frills, but free !

Why to learn R ?

- **Since it is free and open-source.**

- **R consist of wide selection of additional libraries.**

- **Main library repositories CRAN and Bioconductor.**

- **A massive set of packages for statistical modelling, machine learning, visualization, and importing and manipulating data.**

- **Powerful tools for communicating your results.**

- **Deep-seated language support for data analysis. This includes features like missing values, data frames, and vectorization.**

Why to learn R ?

- **Cutting edge tools. Researchers in statistics and machine learning will often publish an R package to accompany their articles. This means immediate access to the very latest statistical techniques and implementations.**

- **RMarkdown makes it easy to turn your results into HTML files, PDFs, Word documents, PowerPoint presentations, dashboards and more.**

- **Shiny allows you to make beautiful interactive apps without any knowledge of HTML or javascript.**

- **RStudio, the IDE, provides an integrated development environment, tailored to the needs of data science, interactive data analysis, and statistical programming.**

- **R can connect easily to high-performance programming languages like C, Fortran, and C++.**
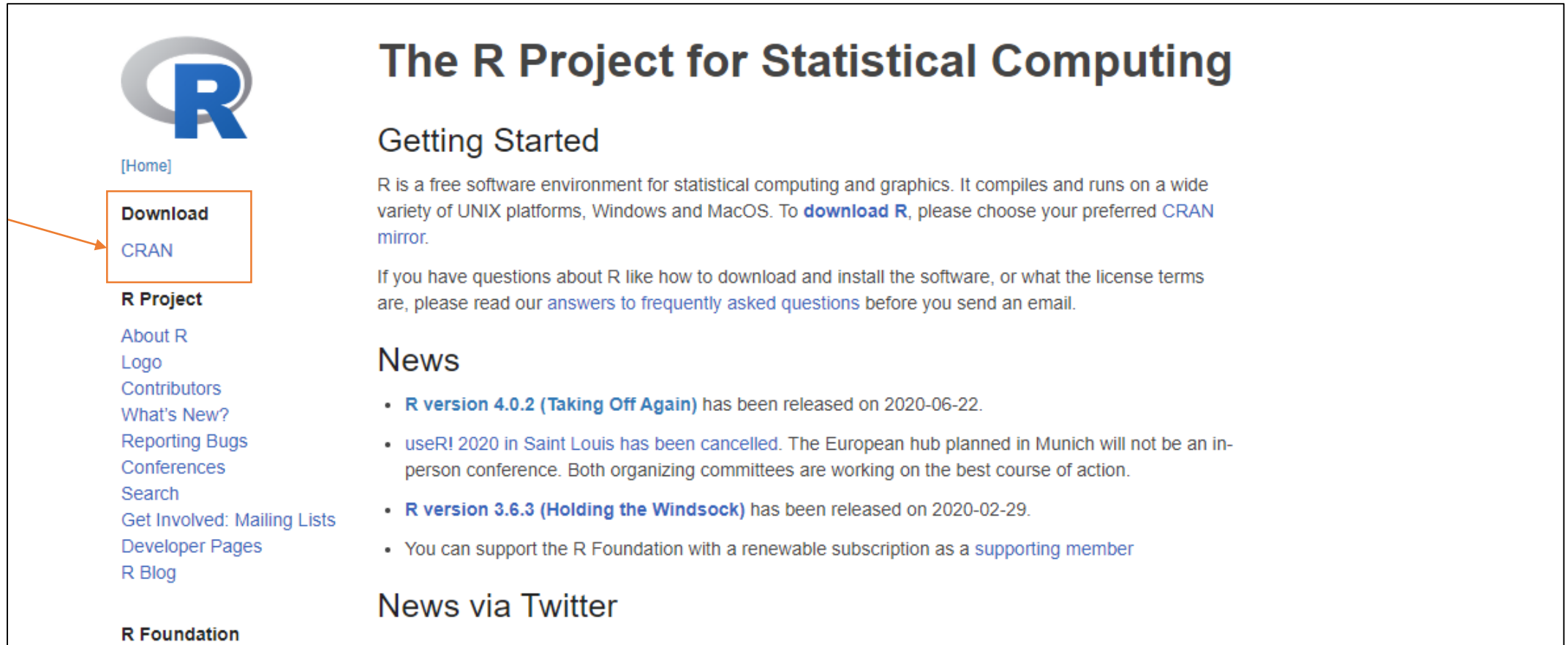
Why to learn R ?

- R's metaprogramming capabilities allow you to write magically succinct and concise functions.

- Excellent environment to handle ggplot2, dplyr, data.table, and more.

- Compared to other programming languages, the R community is more focused on results than processes.

Install R

https://www.r-project.org/

and do the following (assuming you work on a windows computer)

Choose download site

Choose Windows as target operation system

Click base

Click Download R 4.0.2

Download the .exe file and run it (choose default answers for all questions)

Install Rstudio

https://rstudio.com/products/rstudio/download/

RStudio Desktop	RStudio Desktop	RStudio Server	RStudio Server Pro
Open Source License	Commercial License	Open Source License	Commercial License
Free	$995	Free	$4,975
	/year		/year
			(5 Named Users)
DOWNLOAD	BUY	DOWNLOAD	BUY

RStudio Desktop 1.3.1073 - Release Notes

1. Install R.   RStudio requires R 3.0.1+.

2. Download RStudio Desktop.   Recommended for your system:

> DOWNLOAD RSTUDIO FOR WINDOWS
> 1.3.1073 | 171.62MB

Requires Windows 10/8/7 (64-bit)

Genetics and Bioinformatics

Rstudio layout

RStudio

- **Bottom left : console window . Here you can type simple commands after the > prompt and R will execute your command. This is the most important window, because this is where R actually does stuff.**

- **Top left : editor window (also called script window). Collections of commands (scripts) can be edited and saved.**

- **Top right : workspace/history window. In the workspace window, you can see which data and values R has in its memory. You can view and edit the values by clicking on them. This history window shows what has been typed before.**

- **Bottom right : files/plots/packages/help window. Here you can open files, view plots (also previous plots), install and load packages and use the help function.**

Use R or RStudio

Lets use it!

Operators and DATA TYPES

Variables/Operators

- **Variables store one element**

x <- 25

Here x variable is assigned value 25

- **Check value assigned to the variable x**

> x
[1] 25

- **Basic mathematical operators that could be applied to variables : (+), (-), (-),(*)**

- **Use parenthesis to obtain desired sequence of mathematical operations**

Calculator

R can be used as a calculator. You can just type your equation in the command windows after the > :

> 10 + 20

Workspace

You can also give numbers a name. By doing so, they become so-called variables which can be used later. For example, you can type in the command window:

> a = 4

- You can also ask R what a is (just type a ENTER in the command window):

> a

- Or do calculations with a :

> a * 5

- To remove all variables from R's memory, type

> rm(list=ls())

Concatenation function

c() ,

>x <- c(1,2,3,4,5)
>x
>y <- c("a", "b", "c", "d")
>y

Vectors

- **Vectors have only 1 dimension and represent enumerated sequence of data. They can also store variables**

> v1 <- c(1,2,3,4)
> mean (v1)

- **The elements of a vector are specified/modified with braces (e.g. [number])**

> v1[1] <- 48
v1
[1] 48 2 3 4

Logical operators

- **These operators mostly work on vectors, matrices and other data types.**

- **Type of data is not important, the same operators are used for numeric and character data type**

Operator	Description
<	less than
<=	less than or equal to
>	greater than
>=	greater than or equal to
==	exactly equal to
!=	not equal to

R workspace

- **Display all workplace objects (variables, vectors etc.) via ls():**

  > a= 10
  > b =20
  > ls()
  [1] "a" "b"
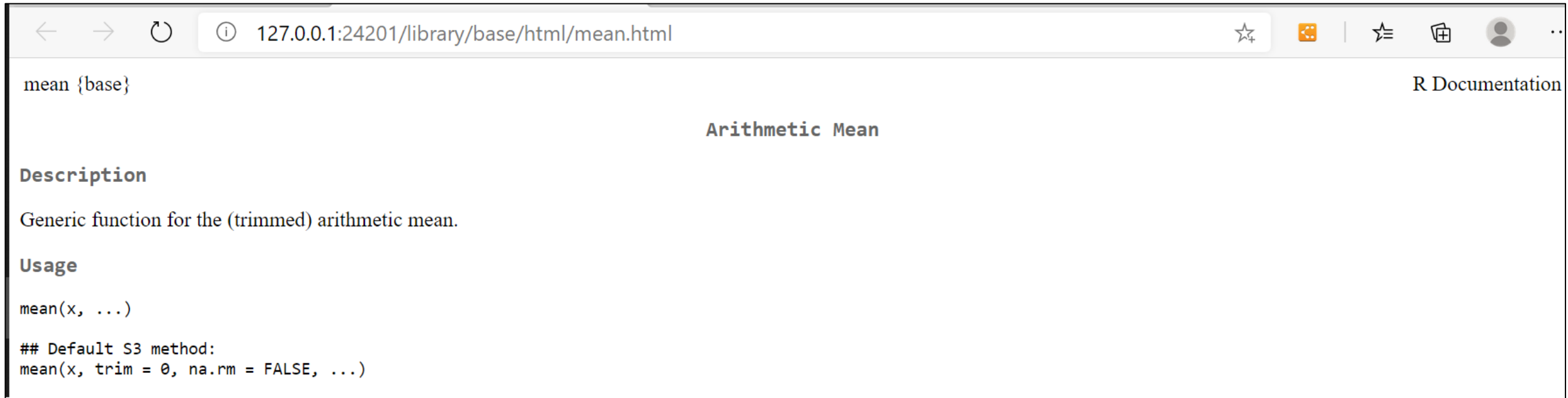
- **Useful tip : to save workplace and restore from a file use:**

  >save.image(file="workspace.rda")
  >load(file="workspace.rda")

Information of function

- **Any function in R has help information**

- **To invoke help use ? Sign and help():**

> **? mean**
> **help(mean)**

127.0.0.1:24201/library/base/html/mean.html

mean {base}   R Documentation

Arithmetic Mean

Description

Generic function for the (trimmed) arithmetic mean.

Usage

```
mean(x, ...)

## Default S3 method:
mean(x, trim = 0, na.rm = FALSE, ...)
```

Data types

- **Data could be of 3 basic data types:**

- **numeric**

- **Character**

- **logical**

Numeric variable type :

> > x <- 1
> > mode(x)
> [1] "numeric"

logical variable type :

```
> y <- 3 < 4
> y
[1] TRUE
> mode(y)
[1] "logical"
```

Character variable type :

```
> a = "bioinfo"
> mode(a)
[1] "character"
```

Data objects

- **The main data objects in R :**

  - **Matrices (single data type)**

  - **Data frames (supports various data types)**

  - **Lists (contains set of vectors)**

- **Matrices are 2D objects (rows/columns) :**

```
>  m <- matrix(0,2,3)
> m
        [,1] [,2] [,3]
[1,]   0    0    0
[2,]   0    0    0
```

List

- **List contain various vectors. Each vectors in the list can be acessed by double braces [[number]]**

```
> x <- c(1,2,3,4)
> y <- c(2,3,4)
> L1 <- list(x,y)
> L1
        [[1]]
        [1] 1 2 3 4
        [[2]]
        [1] 2 3 4
```

Data frames

- **Data frames are similar to matrices but can contain various data types**

```
>  x <- c(1,5,10)
>  y <- c("A","B","C")
> z <- data.frame(x,y)
> z
   x y
1  1 A
2  5 B
3 10 C
```

RStudio : Advance Feature

R markdown

- **Markdown is a simple formatting language designed to make authoring content easy for everyone.**

- **Rather than write in complex markup code (e.g. HTML or LaTex), you write in plain text with formatting cues.**
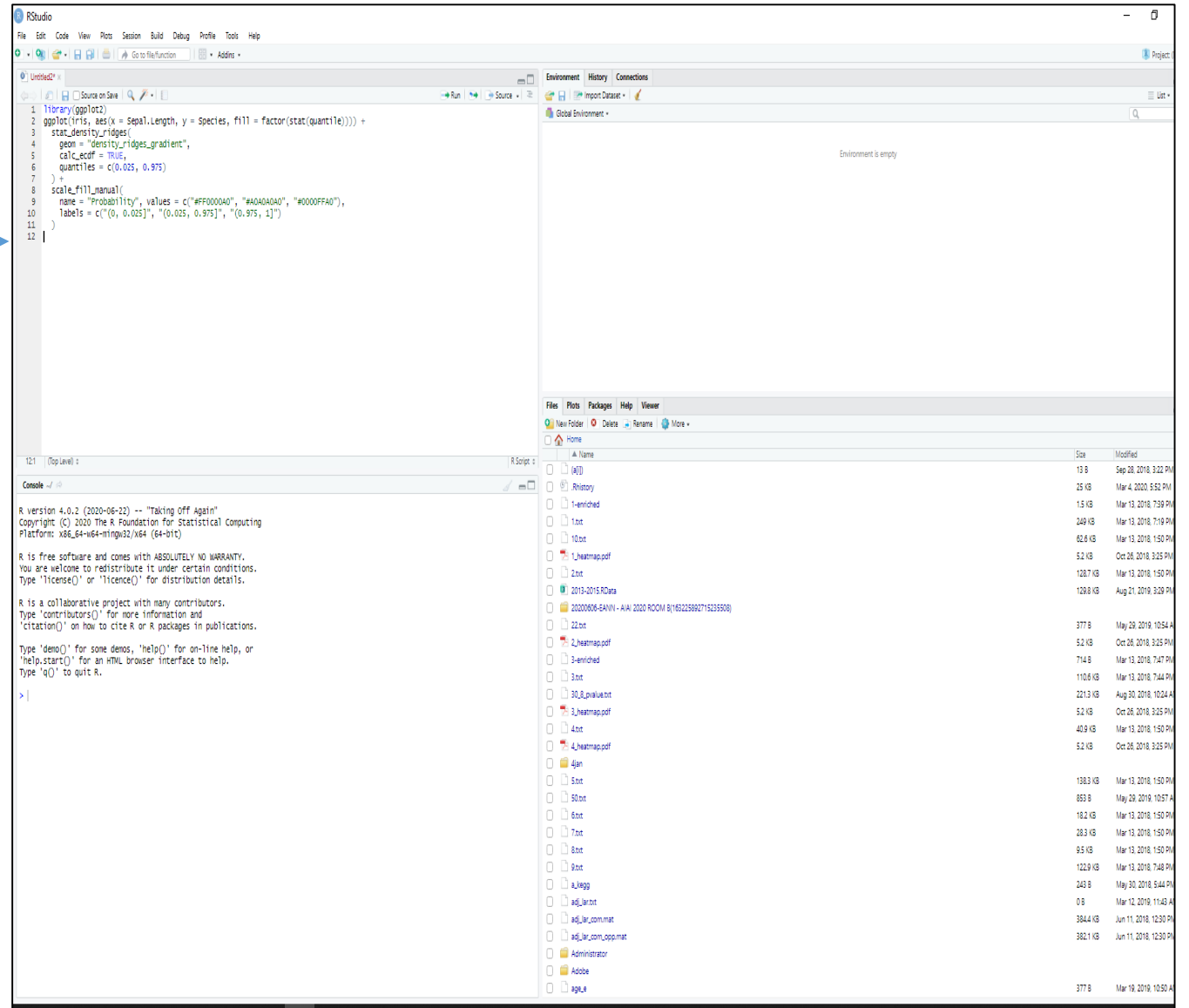
install.packages("rmarkdown")

install.packages("ggplot2")
install.packages("ggridges")

```
library(ggplot2)
library(ggridges)

ggplot(iris, aes(x = Sepal.Length, y =
Species, fill = factor(stat(quantile)))) +
  stat_density_ridges(
    geom = "density_ridges_gradient",
    calc_ecdf = TRUE,
    quantiles = c(0.025, 0.975)
  ) +
  scale_fill_manual(
    name = "Probability", values =
c("#FF0000A0", "#A0A0A0A0",
"#0000FFA0"),
    labels = c("(0, 0.025]", "(0.025, 0.975]",
"(0.975, 1]")
  )
```

- **Run code**

- **Click option File -> knit document**

Report done

Inbuilt functions in R

duplicated()function in R

- **Duplicate data can be removed during analysis.**

- **It returns a logical vector that tells you whether the specified value is a duplicate of a previous value.**

> duplicated(c(1,2,1,3,1,4))

For all those values which are duplicate in the sample, true is returned.

Read data in R

- **file.choose() to walk through your directory to select a file and load.**

- **In case you don't want header then set header=FALSE.**

> data <- file.choose()
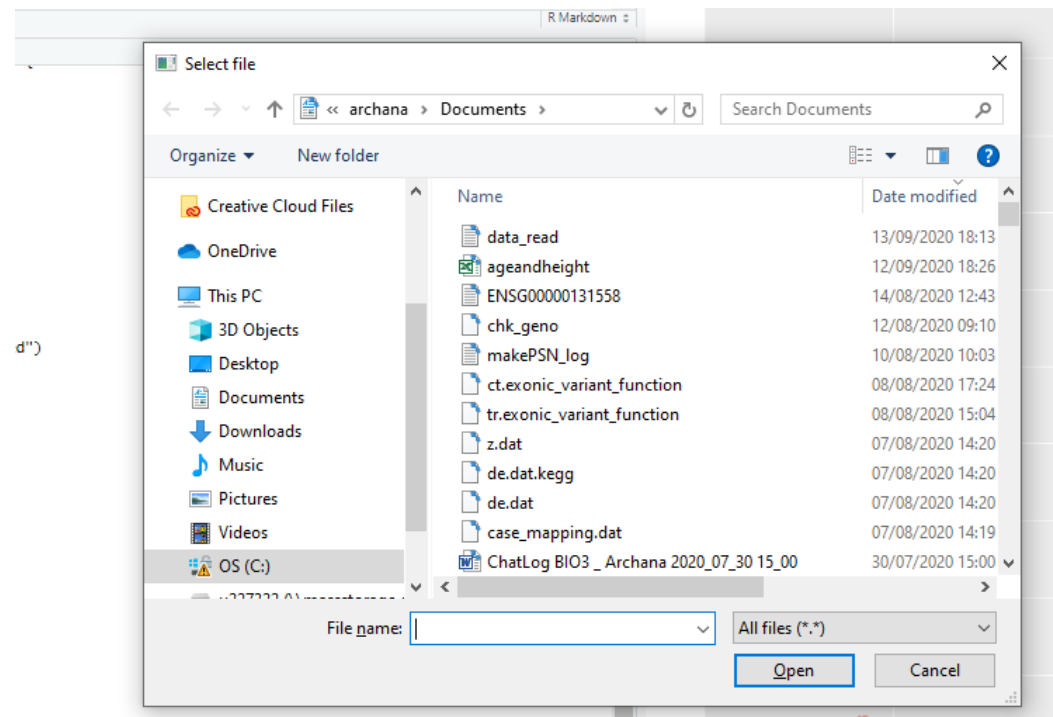
> data_R <- read.table(data)
> data_R

```
> data_R
      sample1 sample2 sample3 sample4 sample5 sample6
gene1    3       1       3       1       3       1
gene2    2       2       2       2       2       2
gene3    1       7       1       7       1       7
gene4    2       6       2       6       2       6
gene5    3       3       3       3       3       3
gene6    3       2       3       2       3       2
gene7    2       2       2       2       2       2
gene8    1       1       1       1       1       1
```

Write data in R

- In R, we can write data frames easily to a file, using the write. table() command.

- The first argument refers to the data frame to be written to the output file, the second is the name of the output file.

- By default R will surround each entry in the output file by quotes, so we use quote=F

> write.table(x, file, sep = " ", row.names = TRUE, col.names = TRUE)

na.omit() function in R

- **Rows which have NA values can be removed using the na.omit() function as below:**
- **Download data from website**

```
> read <- read.table(file="data_read2.txt")
> read
      sample1 sample2 sample3 sample4 sample5 sample6
gene1     3       1       3       1       3       1
gene2     2       2       2       2       2       2
gene3    NA       7       1       7       1       7
gene4     2       6       2       6       2       6
gene5     3       3       3       3       3       3
gene6    NA       2       3       2       3       2
gene7     2       2       2       2       2       2
gene8    NA       1       1       1       1       1
> na.omit(read)
      sample1 sample2 sample3 sample4 sample5 sample6
gene1     3       1       3       1       3       1
gene2     2       2       2       2       2       2
gene4     2       6       2       6       2       6
gene5     3       3       3       3       3       3
gene7     2       2       2       2       2       2
```

table() function in R

 To count the number of observations in each level of factor, we can use the R table() command as below:

> data(iris)

> head(iris)

> table(iris$Species)

```
> data(iris)
> head(iris)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
> table(iris$Species)

    setosa versicolor  virginica
        50         50         50
```

merge() function in R

If you want to combine data from different sources in <u>R</u>, you can combine different sets of data in three ways:

```
> data(cars)
> data(iris)
> str(cars)
'data.frame':    50 obs. of  2 variables:
 $ speed: num  4 4 7 7 8 9 10 10 10 11 ...
 $ dist : num  2 10 4 22 16 10 18 26 34 17 ...
> str(iris)
'data.frame':    150 obs. of  5 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
> merged <- merge(iris,cars)
> str(merged)
'data.frame':    7500 obs. of  7 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ speed       : num  4 4 4 4 4 4 4 4 4 4 ...
 $ dist        : num  2 2 2 2 2 2 2 2 2 2 ...
> |
```

The *str()* command is designed to help you examine the structure of a data object rather than providing a statistical summary.

Input/output

- **Input from keyboard**

> **> z <- scan()**

- **You can use readline() for inputing a line from the keyboard in the form of a string:**

> **> w <- readline()**
> **bioinfo**
> **> w**
> **[1] "bioinfo"**

- **In interactive mode, one can print the value of that variable by just typing the variable name or expression. In batch mode, one can use the print() function.**

Input/output

- The function read.table() is used usually.

- The default value of a header is 'FALSE' and hence when you do not have a header, you need not say such.

- Basically, the character strings are considered as R factors. For turning this "feature" off, you can include the argument as.is=T in your call to read.table().

- When you have a spreadsheet export file, i.e. having a type .csv where the fields are divided by commas in place of spaces, use read.csv() in place of read.table().

- You can also use read.xls for reading core spreadsheet files.

Exercise

- **a <- c(1,2,3,4,5,6,7,8)**

- **b <- c(3,4,5,6,7,8,10)**

- **Calculate mean of a and b**

- **Check if mean(a) > mean(b) or not ?**

- **Download data from website**

- **Write a program to read data (data_read3.txt) and display content.**

summary() function in R

The *summary()* command will provide you with a statistical summary of your data.

> summary(iris)

```
> summary(iris)
  Sepal.Length    Sepal.Width     Petal.Length    Petal.Width          Species
 Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100   setosa    :50
 1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300   versicolor:50
 Median :5.800   Median :3.000   Median :4.350   Median :1.300   virginica :50
 Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
 3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
 Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
>
```

The summary command is, therefore, more useful as we can see minimum, maximum, mean, etc values. The summary() command works for both matrix and data frame objects by summarizing the columns rather than the rows.

Name Commands in R

- **Name command and its variants are used to find or add names to rows and columns of data structures.**

- **Below specified are few of the commands and their explanation:**

- **names() – It works on matrix or data frame objects.**

- **rownames() – It works on matrix or data frame objects and is used to give names to rows.**

- **colnames() – It works on matrix or data frame objects and is used to give names to columns.**

- **dimnames() – Gets row and column names for matrix or data frame objects, that is, it is used to see dimensions of the data frame.**

Exercise
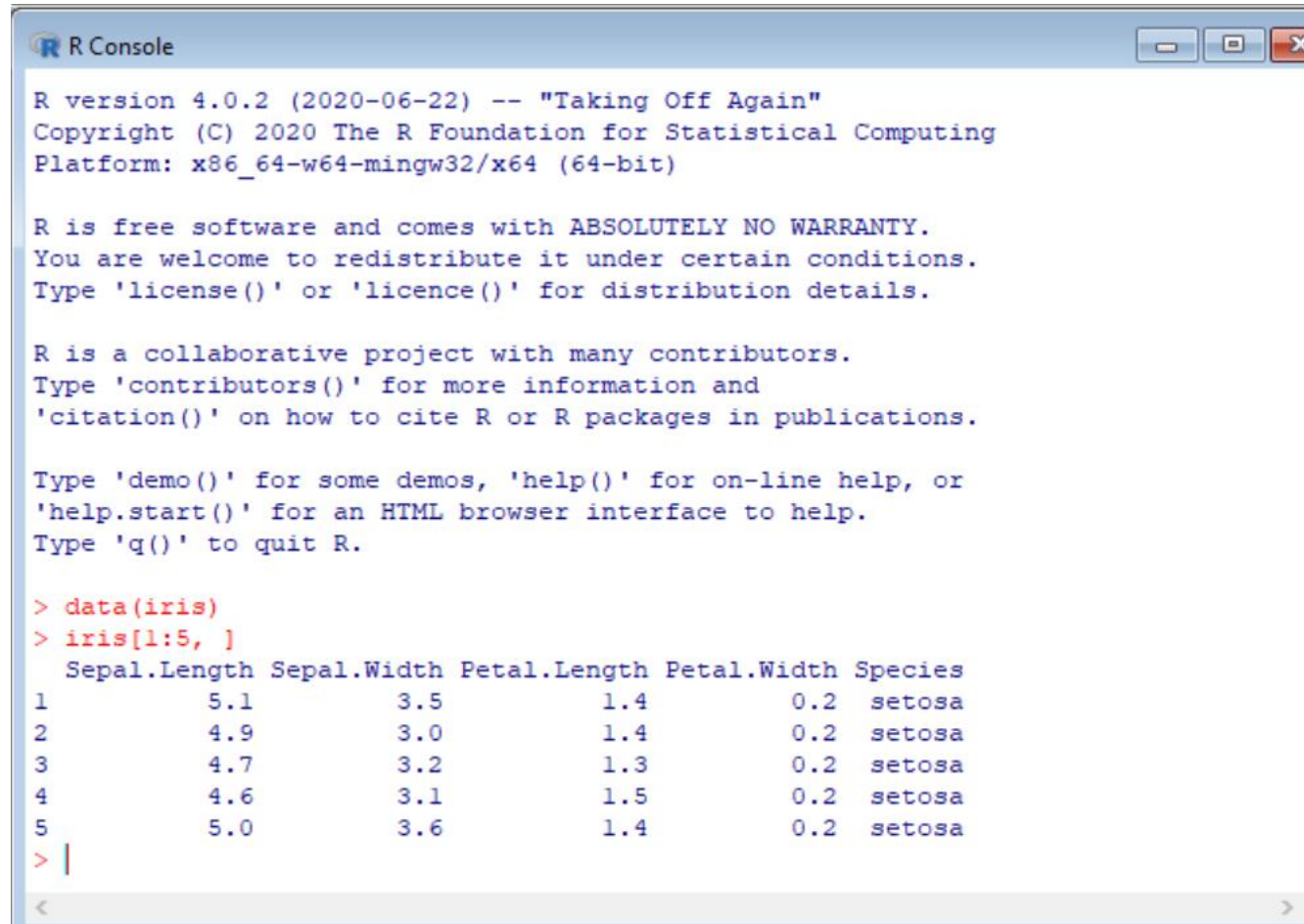
- **Write a program to read data (data_read3.txt) and display content.**

- **Display rownames() and colnames()**

Creating Subsets of Data in R

- The process of creating samples is called subsetting.

- As we know, data size is increasing exponentially.

- So, the data is divided into small-sized samples and analysis of samples is done.

- Different methods of subsetting in R are:

- $

- The dollar sign operator selects a single element of data. The result of this operator is always a vector when we use it with a data-frame.

- [[

- **To retrieve 5 rows and all columns of already built-in dataset iris, the below command, is used:**

> data(iris)
> iris[1:5,]

```
R R Console                                                    ─  ▢  ✕

R version 4.0.2 (2020-06-22) -- "Taking Off Again"
Copyright (C) 2020 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> data(iris)
> iris[1:5, ]
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
>
```

Data Visualization in R

Data Visualization in R

- **R Programming helps us to learn this art by offering a set of inbuilt functions and also libraries to build visualizations and present data.**

- **Before we move forward for the technical implementation of the visualization, let's see first how to select the right chart type.**

Data Visualization in R - Simple

Plotting in R

- **R provides very rich set of plotting possibilities**

- **The basic command is plot()**

- **Each library has its own version of plot() function**

- **When R plots graphics , it opens 'graphical device' that could be either a window or file**

Genetics and Bioinformatics
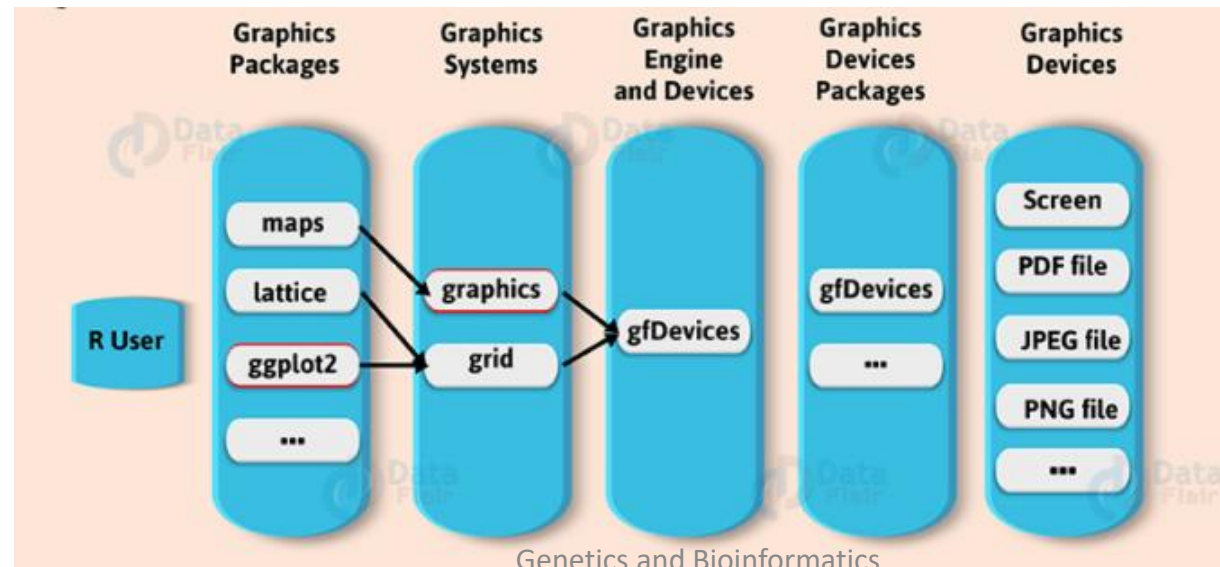
Plotting functions

- R standard graphics available through package graphics, include several functions that provide statistical plots, like:

Function	Description
plot(x)	plot of the values of x variable on the y axis
plot(x,y)	bi-variable plot of x and y values (both axis scaled based on values of x and y variables)
pie(y)	circular pie-char
boxplot(x)	Plots a box plot showing variables via their quantiles
hist(x)	Plots a histogram(bar plot)
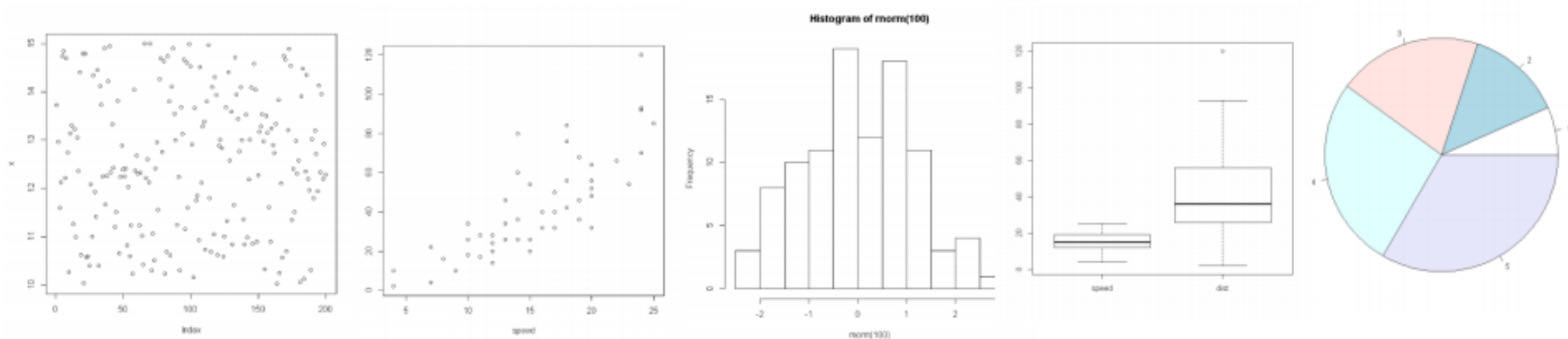
Plot function

```
> data(iris)
> head(iris)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1      5.1         3.5         1.4          0.2      setosa
2      4.9         3.0         1.4          0.2      setosa
3      4.7         3.2         1.3          0.2      setosa
4      4.6         3.1         1.5          0.2      setosa
5      5.0         3.6         1.4          0.2      setosa
6      5.4         3.9         1.7          0.4      setosa

> plot(iris$Sepal.Length,iris$Petal.Length)
```
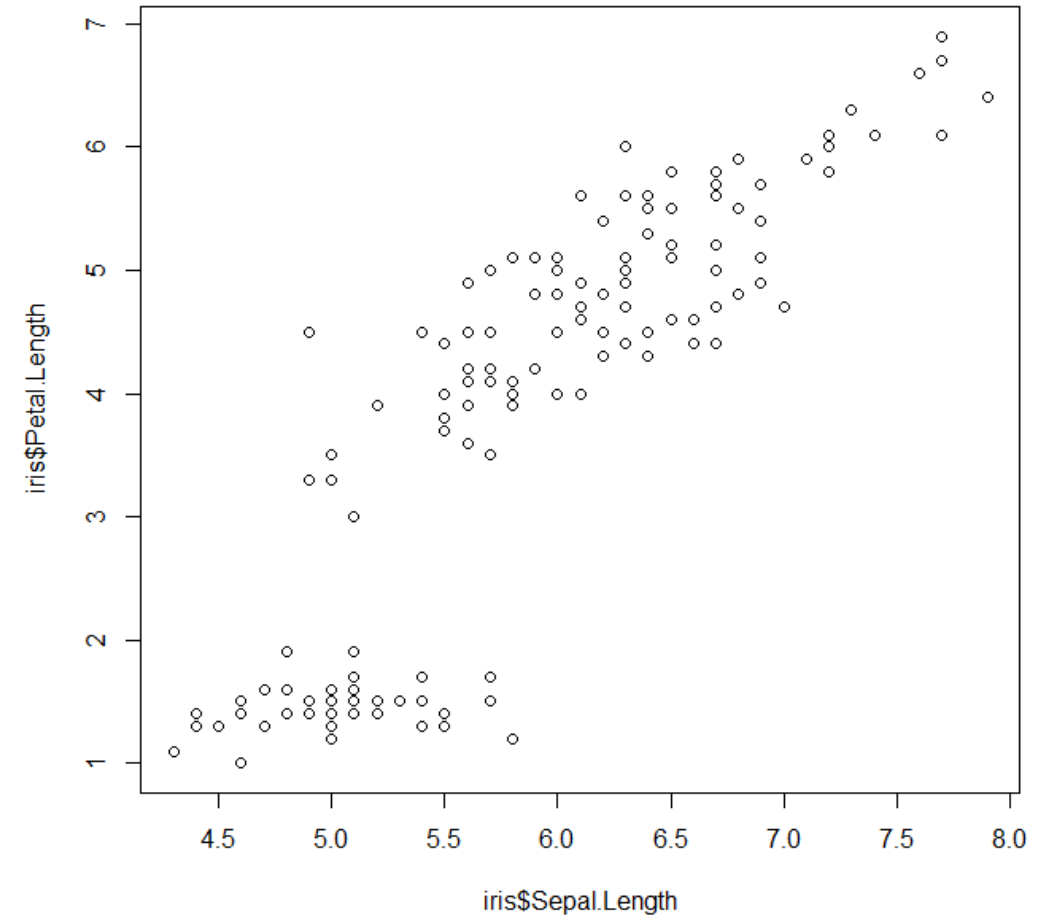
Hist function

- **Histogram**
- **A histogram is used to plot a continuous variable. Also, It helps to break the data into bins and shows the frequency distribution of these bins. Thus, we can always change the bin size and see the effect it has on visualization.**

> hist(iris[,1])
> hist(iris[,2])
> hist(iris[,3])
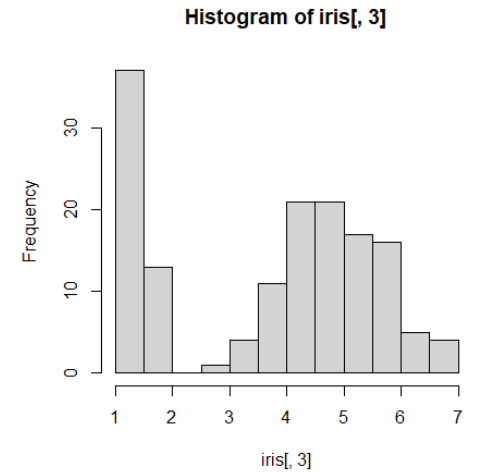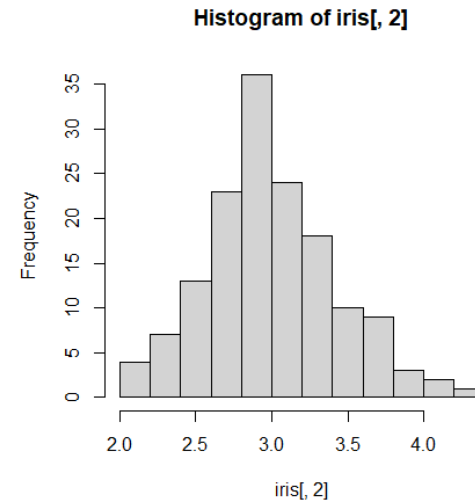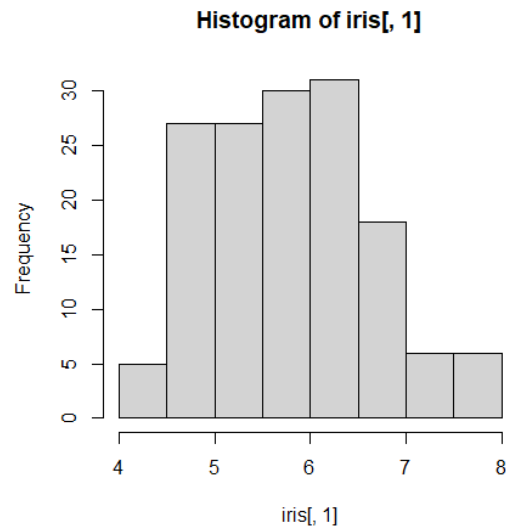
Boxplot function

```
> x <- c(1,5,10)
> y <- c(20,30,40)
> boxplot(x,y)
```
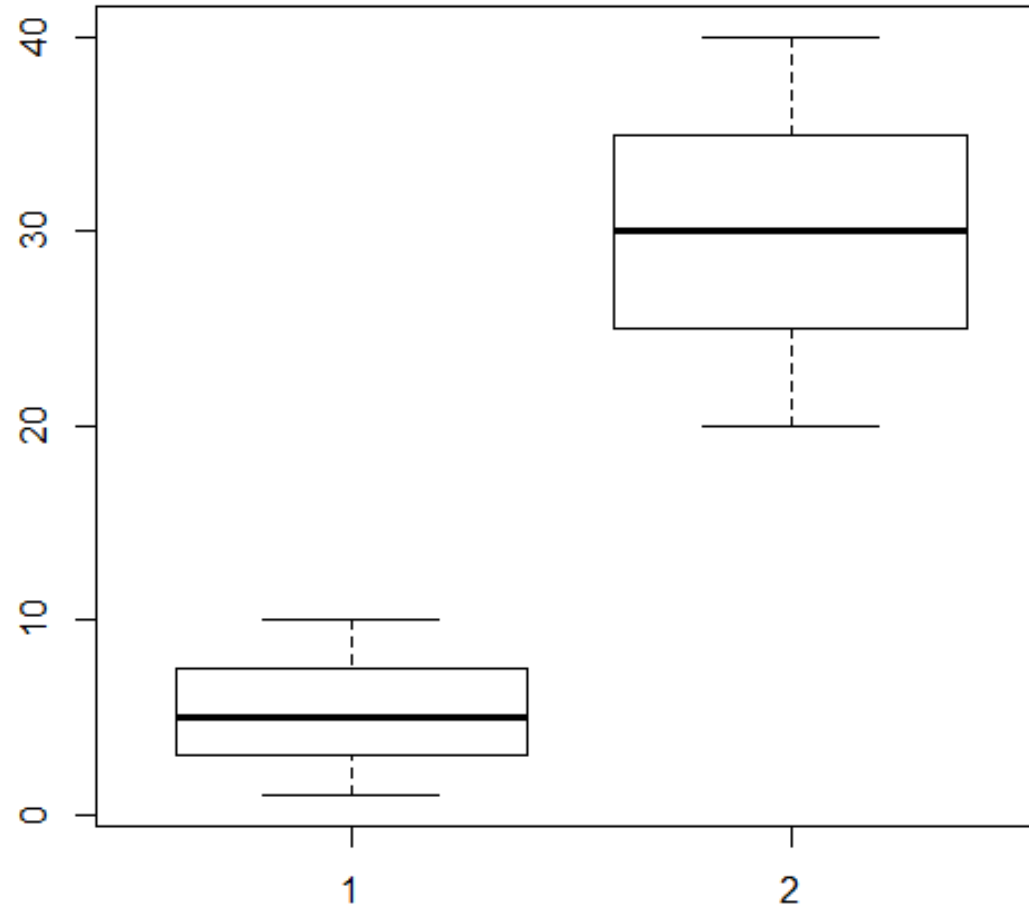
Heatmap in R

- **We use it for the intensity of colours. It is also used to display a relationship between two or three or many variables in a two-dimensional image. Thus, it allows us to explore two dimensions of the axis and the third dimension by an intensity of colour.**

> iris_filtered <- iris[,1:4]

> heatmap(as.matrix(iris_filtered))
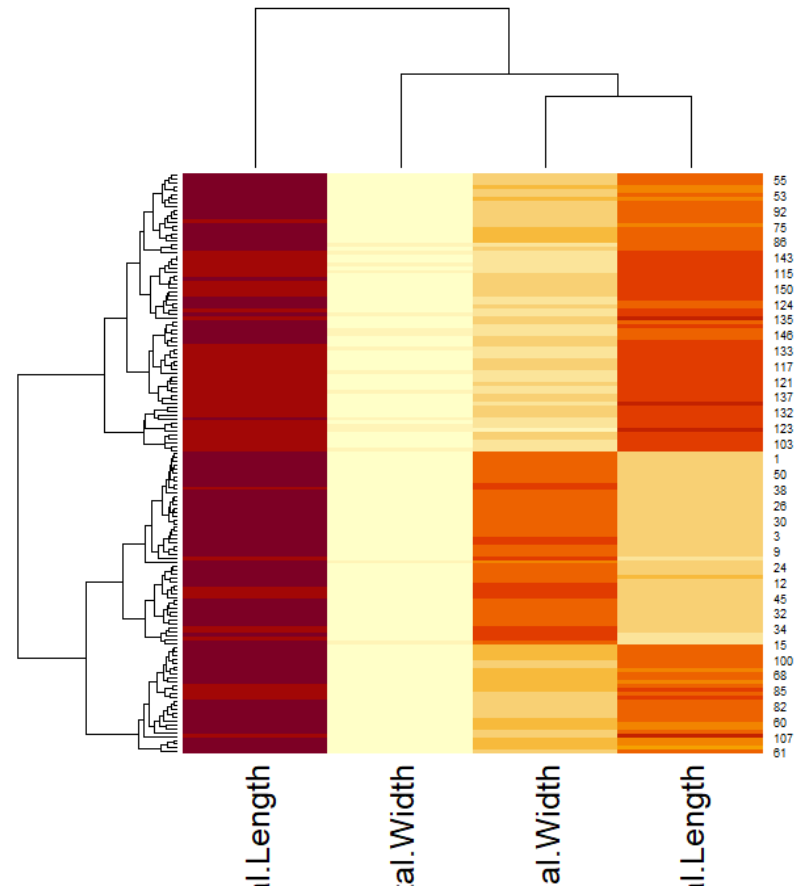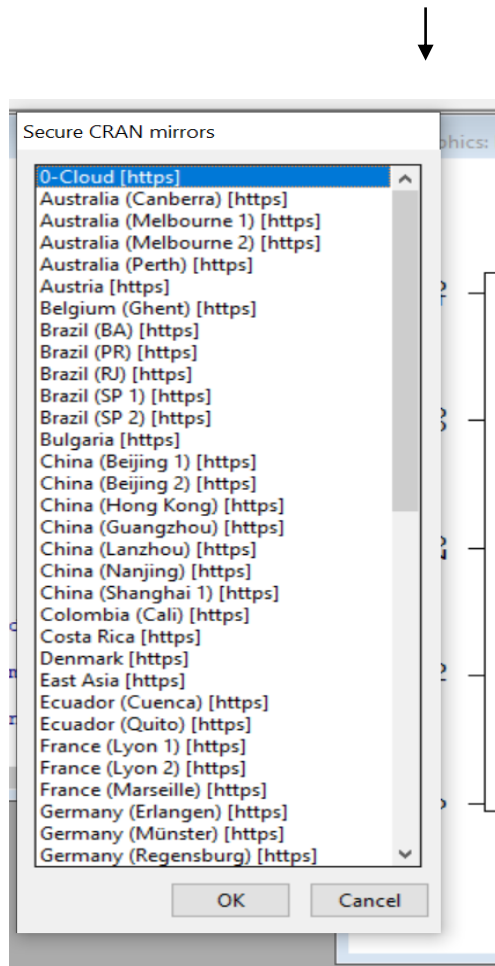
Data Visualization in R - Advanced

Library installation

install.packages

> install.packages("ggplot2")

↓


```
> install.packages("ggplot2")
Installing package into 'C:/Users/archana/Documents/Administrator/R/win-library/3.6'
(as 'lib' is unspecified)
trying URL 'https://lib.ugent.be/CRAN/bin/windows/contrib/3.6/ggplot2_3.3.2.zip'
Content type 'application/zip' length 4068914 bytes (3.9 MB)
downloaded 3.9 MB

package 'ggplot2' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\archana\AppData\Local\Temp\Rtmpw5XZH3\downloaded_packages
```

ggplot2 installed successfully!

Method to Save Graphs to Files in R

- In order to save graphics to an image file, there are three steps in R:

- You can create a graphics device of PNG format using png(), JPG format using jpg() and PDF format using pdf().

- Plot your data.

- Closing the graphics device and saving the image using dev.off.

```
> jpeg("c:/mygraphs/myplot.jpg")
> x <- c(1,2,2)
> y <- c(3,4,4)
> plot(x,y)
> dev.off()
```

Corrgram Library installation

install.packages

> **install.packages("corrgram")**

↓

Secure CRAN mirrors	
0-Cloud [https]	
Australia (Canberra) [https]	
Australia (Melbourne 1) [https]	
Australia (Melbourne 2) [https]	
Australia (Perth) [https]	
Austria [https]	
Belgium (Ghent) [https]	
Brazil (BA) [https]	
Brazil (PR) [https]	
Brazil (RJ) [https]	
Brazil (SP 1) [https]	
Brazil (SP 2) [https]	
Bulgaria [https]	
China (Beijing 1) [https]	
China (Beijing 2) [https]	
China (Hong Kong) [https]	
China (Guangzhou) [https]	
China (Lanzhou) [https]	
China (Nanjing) [https]	
China (Shanghai 1) [https]	
Colombia (Cali) [https]	
Costa Rica [https]	
Denmark [https]	
East Asia [https]	
Ecuador (Cuenca) [https]	
Ecuador (Quito) [https]	
France (Lyon 1) [https]	
France (Lyon 2) [https]	
France (Marseille) [https]	
Germany (Erlangen) [https]	
Germany (Münster) [https]	
Germany (Regensburg) [https]	

OK   Cancel

→ **Correlograms installed successfully!**

Corrgram in R

- We use it to test the level of correlation and also among the variable available in the dataset.

- Thus, the cells of the matrix can be shaded or coloured to show the co-relation value.

```
> library(corrgram)
> data(mtcars)
> corrgram(mtcars, order=TRUE, lower.panel=panel.shade, upper.panel=panel.pie,
text.panel=panel.txt, main="Car Milage Data in PC2/PC1 Order")
```

Car Milage Data in PC2/PC1 Order
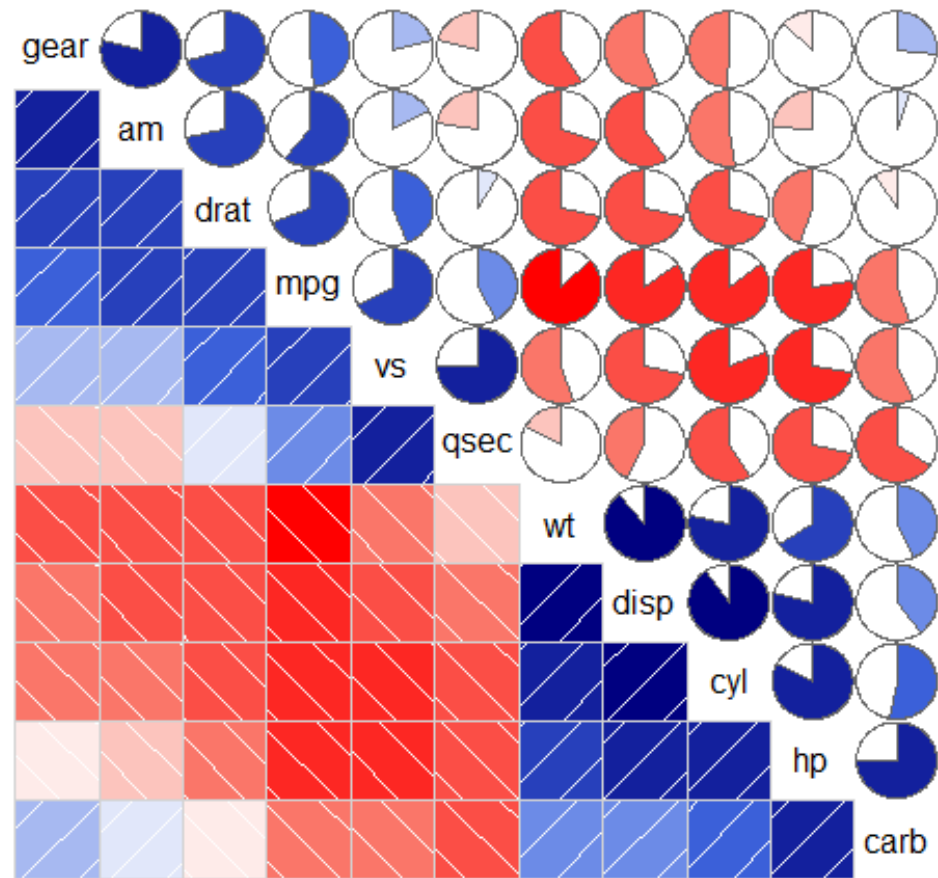
circlize Library installation

install.packages

> install.packages("circlize")

↓
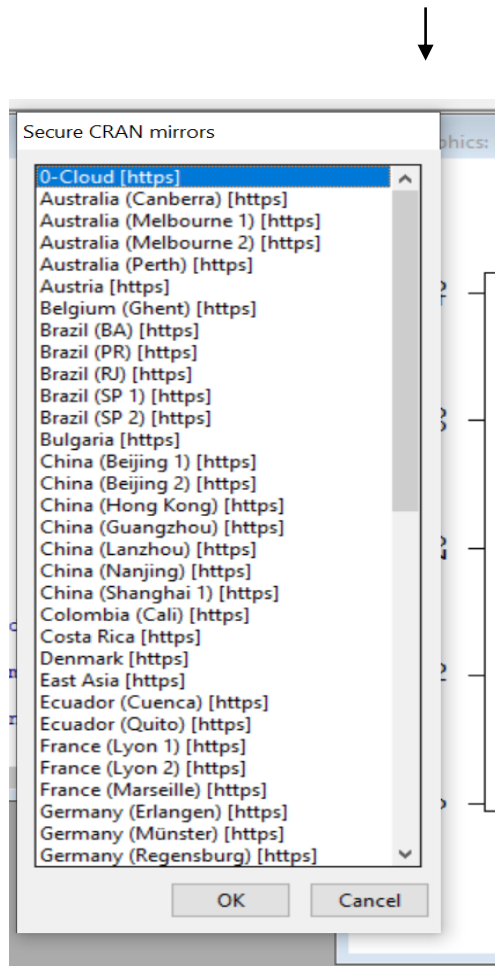
circlize installed successfully!

> library("circlize")
> mat = matrix(rnorm(36), 6, 6)
> rownames(mat) = paste0("R", 1:6)
> colnames(mat) = paste0("C", 1:6)

```
> mat = matrix(rnorm(36), 6, 6)
> rownames(mat) = paste0("R", 1:6)
> colnames(mat) = paste0("C", 1:6)
> mat
           C1          C2          C3          C4          C5          C6
R1 -0.3726024   0.6870889   0.3497039   0.24123038 -2.3779166   1.4447286
R2  0.2924820   0.1061324  -0.6347521   0.34156695  0.8794742  -0.5098149
R3  0.2520366   0.6003891   0.5031263  -0.25550267  0.4714030   0.3368118
R4 -1.1833159  -0.2451636   1.0311421  -1.55260575  1.5363097   0.3335470
R5  0.1313151   0.3054917   0.4118880  -0.02681157  0.8029791   2.2468392
R6  2.1757592   0.8985047   1.4857436  -0.30758183 -2.1677454   0.3984287
```

> chordDiagram(mat)
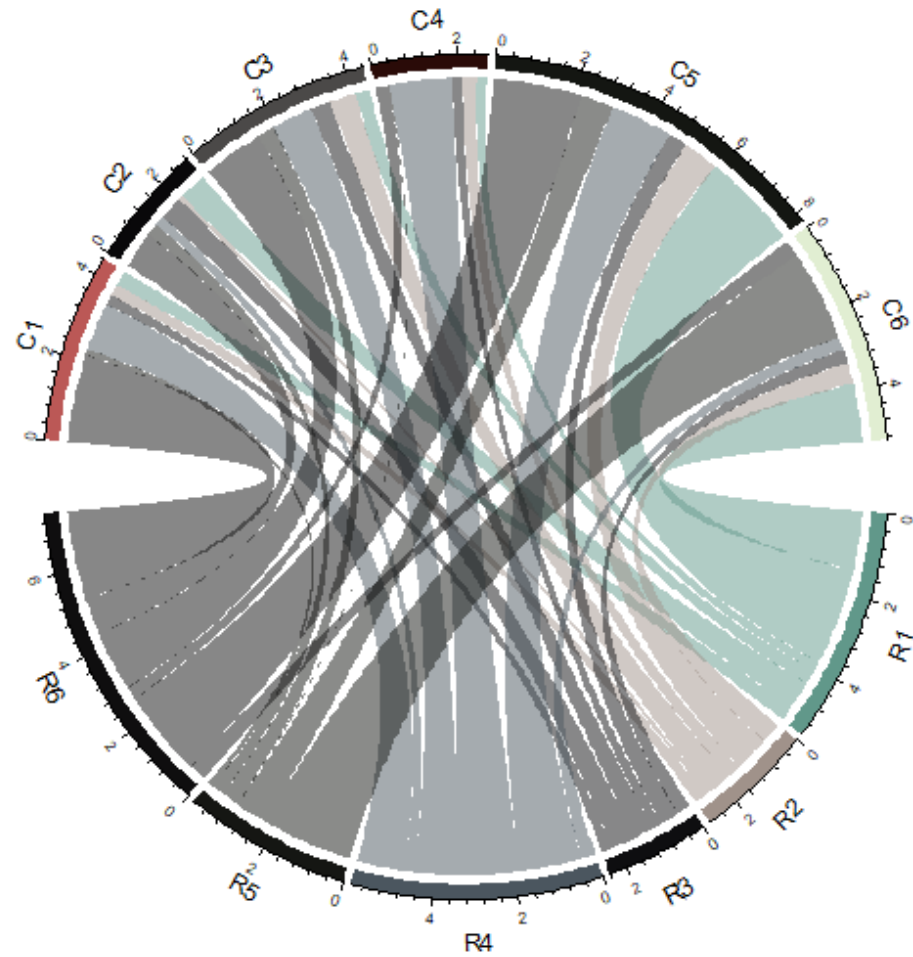
matrix data in R

igraph Library installation

install.packages

> install.packages("igraph")

↓

igraph installed successfully!

library(igraph)

A graph in made up of vertices (also called nodes or points) which are connected by edges (also called links or lines).

#Define Nodes
```
 nodes=cbind('id'=c('Fermenters','Methanogens','carbs','CO2','H2','other','CH4','H2O'),
       'type'=c(rep('Microbe',2),rep('nonBio',6)))
nodes
```

#Define Links
```
links=cbind('from'=c('carbs',rep('Fermenters',3),rep('Methanogens',2),'CO2','H2'),
       'to'=c('Fermenters','other','CO2','H2','CH4','H2O',rep('Methanogens',2)),
       'type'=c('uptake',rep('output',5),rep('uptake',2)),
       'weight'=rep(1,8))
```

links

#Make the network
library(igraph)
net = graph_from_data_frame(links,vertices = nodes,directed = T)
plot(net)
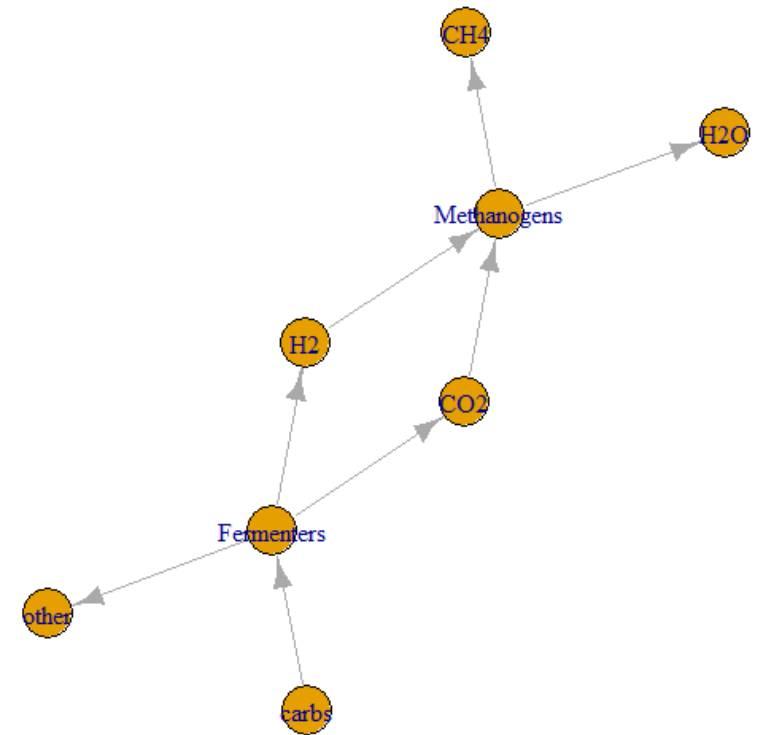
Change Appearance

> colrs.v = c(nonBio = "lightblue",Microbe = "gold") #node colours
> V(net)$color = colrs.v[V(net)$type]

> colrs.e = c(output = "grey", uptake = "magenta") #edge colours
> E(net)$color = colrs.e[E(net)$type]

> plot(net, edge.curved=0.2,vertex.size=30) #make nodes bigger, curve arrows
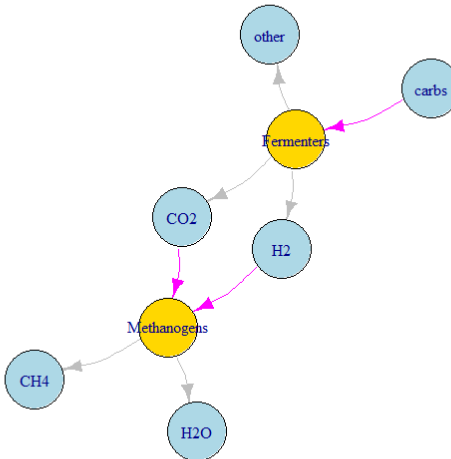
Comparison of both plots
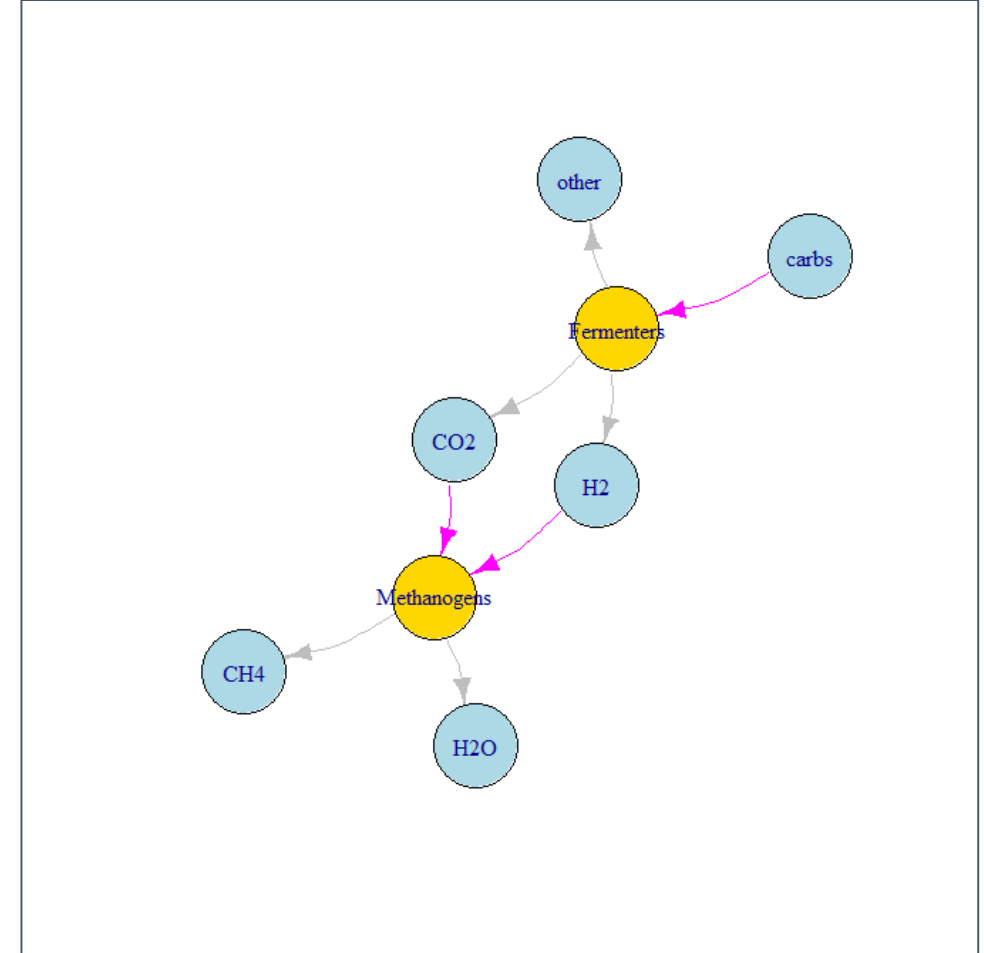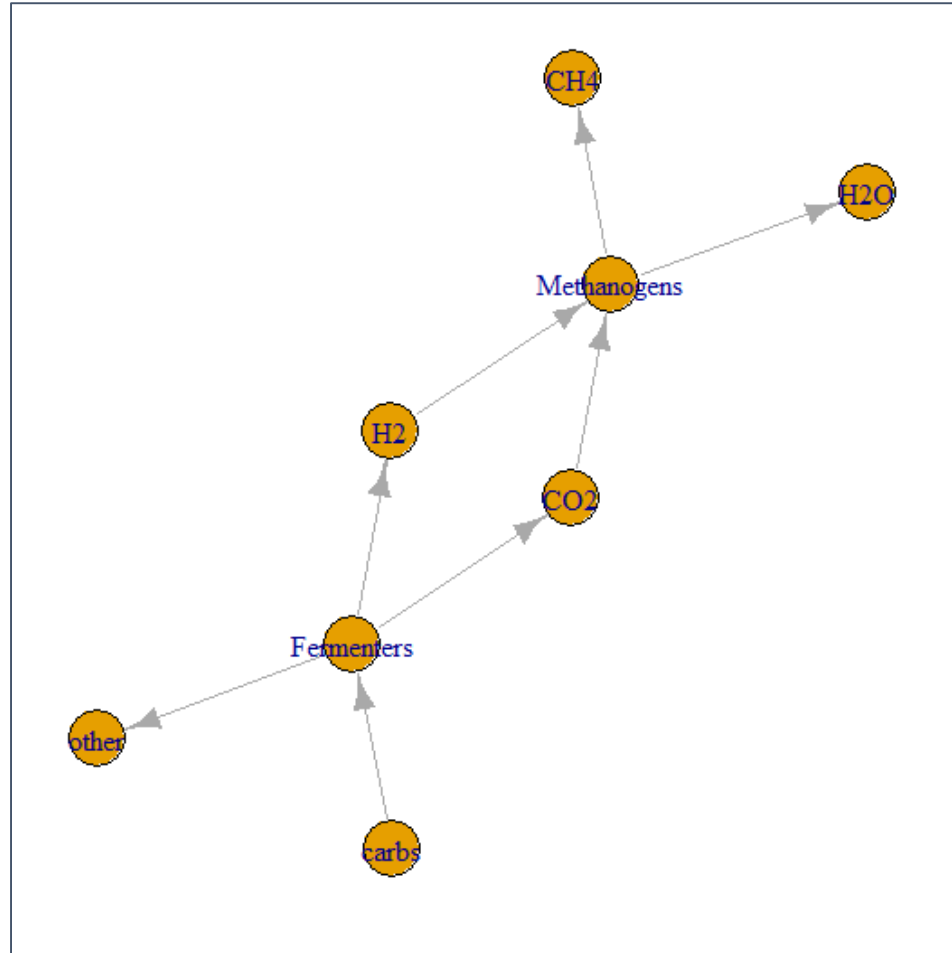
Density ridgeline plots

- **The geom geom_density_ridges calculates density estimates from the provided data and then plots those, using the ridgeline visualization.**

```
>ggplot(iris, aes(x = Sepal.Length, y = Species)) + geom_density_ridges()
```

- **The extent to which the different densities overlap can be controlled with the scale parameter.**
- **A setting of scale=1 means the tallest density curve just touches the baseline of the next higher one.**
- **Smaller values create a separation between the curves, and larger values create more overlap.**

```
>ggplot(iris, aes(x = Sepal.Length, y = Species)) + geom_density_ridges(scale = 0.9)
```
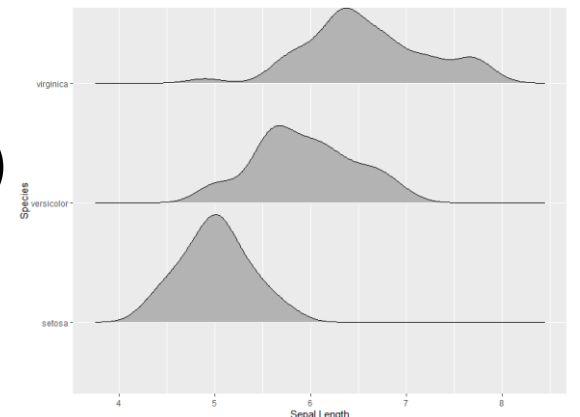
Density ridgeline plots

- **We can also specify quantiles by cut points rather than number. E.g., we can indicate the 2.5% and 97.5% tails.**

```
> ggplot(iris, aes(x = Sepal.Length, y = Species)) + stat_density_ridges(quantile_lines = TRUE, quantiles = c(0.025, 0.975), alpha = 0.7)
```
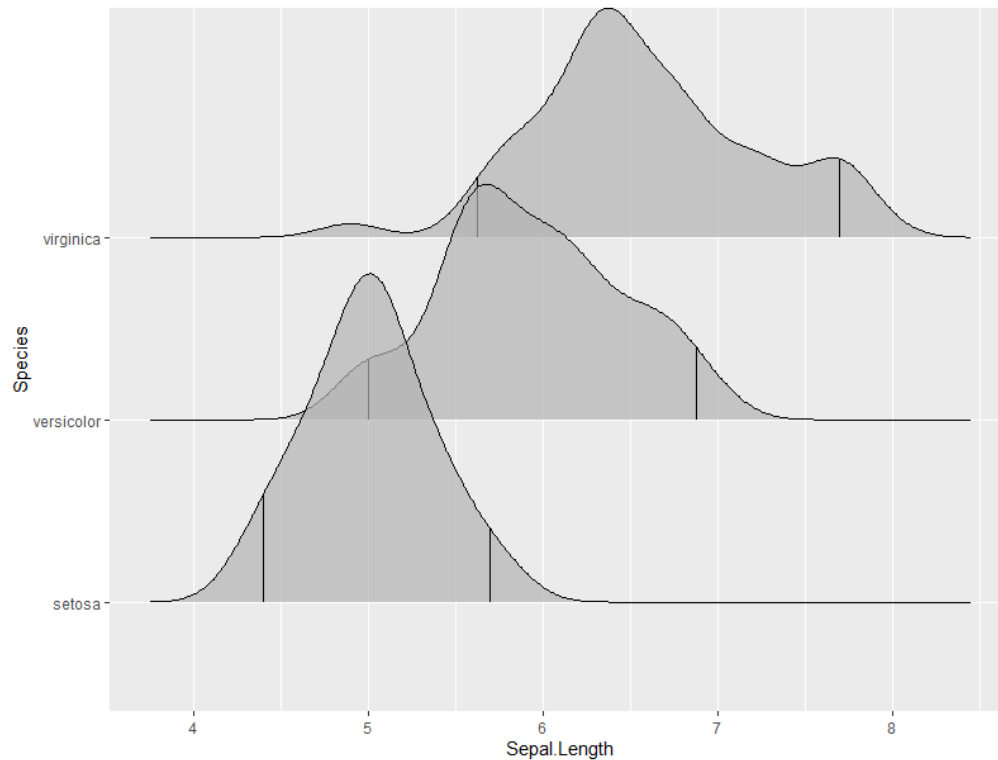
Quantile plots

▪ Using the geom geom_density_ridges_gradient we can also color by quantile, via the calculated stat(quantile) aesthetic. Note that this aesthetic is only calculated if calc_ecdf = TRUE.

> ggplot(iris, aes(x=Sepal.Length, y=Species, fill = factor(stat(quantile)))) + stat_density_ridges(geom = "density_ridges_gradient", calc_ecdf = TRUE, quantiles = 4, quantile_lines = TRUE) + scale_fill_viridis_d(name = "Quartiles")
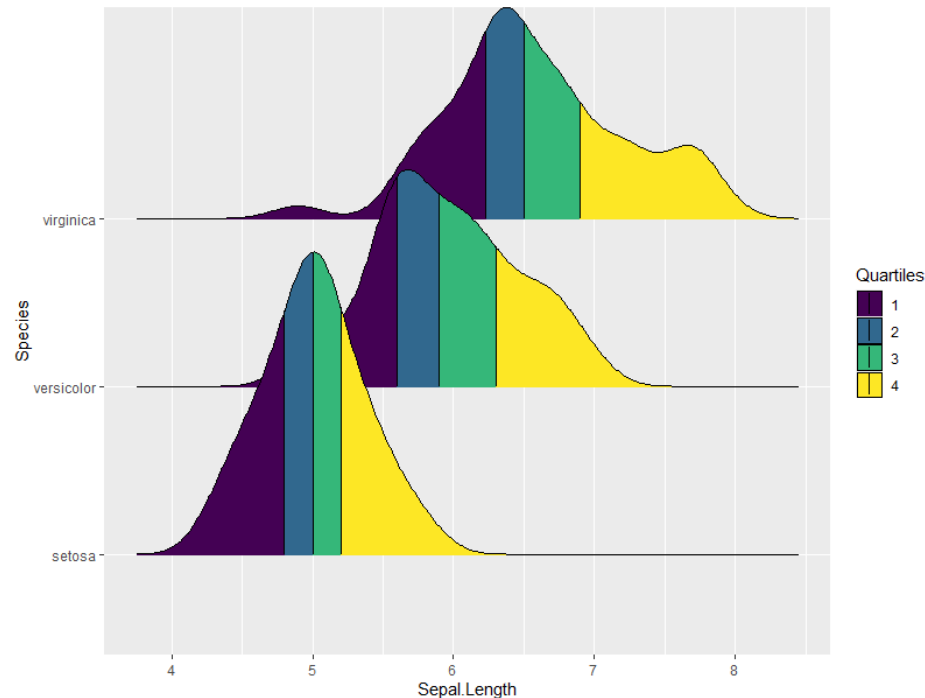
- **We can use the same approach to highlight the tails of the distributions.**

```
> ggplot(iris, aes(x = Sepal.Length, y = Species, fill = factor(stat(quantile)))) +
  stat_density_ridges(
    geom = "density_ridges_gradient",
    calc_ecdf = TRUE,
    quantiles = c(0.025, 0.975)
  ) +
  scale_fill_manual(
    name = "Probability", values = c("#FF0000A0", "#A0A0A0A0", "#0000FFA0"),
    labels = c("(0, 0.025]", "(0.025, 0.975]", "(0.975, 1]")
  )
```
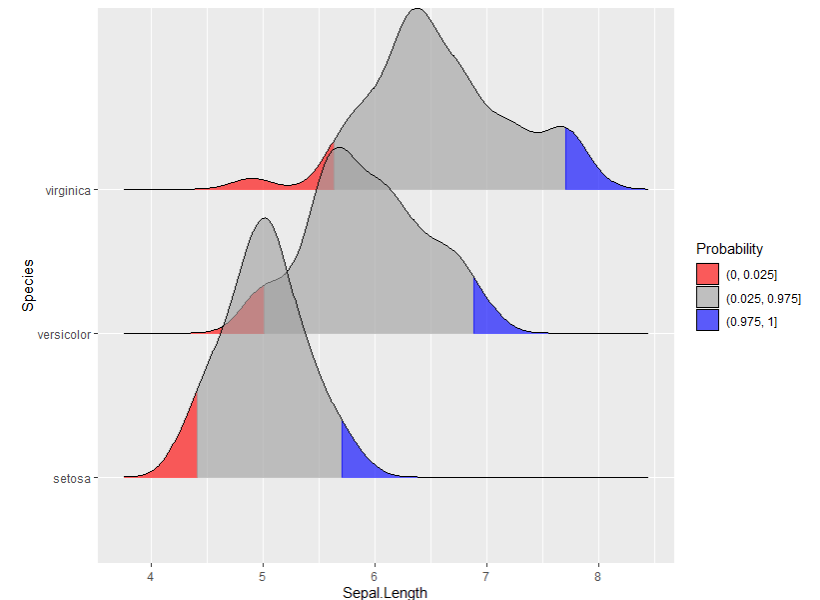
Data Transformation

Dplyr pacakge

- dplyr is a grammar of data manipulation, providing a consistent set of verbs that help you solve the most common data manipulation challenges:

- The easiest way to get dplyr is to install the whole tidyverse:

  >install.packages("tidyverse")

- Alternatively, install just dplyr:

  >install.packages("dplyr")

- mutate() adds new variables that are functions of existing variables
- select() picks variables based on their names.
- filter() picks cases based on their values.
- summarise() reduces multiple values down to a single summary.
- arrange() changes the ordering of the rows.

Dplyr pacakge

```
> head(starwars)
# A tibble: 6 x 14
  name   height   mass hair_color skin_color eye_color birth_year sex    gender
  <chr>   <int> <dbl> <chr>      <chr>      <chr>          <dbl> <chr> <chr>
1 Luke~     172    77 blond      fair       blue              19 male  mascu~
2 C-3PO     167    75 <NA>       gold       yellow           112 none  mascu~
3 R2-D2      96    32 <NA>       white, bl~ red               33 none  mascu~
4 Dart~     202   136 none       white      yellow          41.9 male  mascu~
5 Leia~     150    49 brown      light      brown             19 fema~ femin~
6 Owen~     178   120 brown, gr~ light      blue              52 male  mascu~
# ... with 5 more variables: homeworld <chr>, species <chr>, films <list>,
#   vehicles <list>, starships <list>
```

starwars %>%   filter(species == "Droid")

- **Principal Component Analysis (PCA) is a useful technique for exploratory data analysis, allowing you to better visualize the variation present in a dataset with many variables.**

- **It is particularly helpful in the case of "wide" datasets, where you have many variables for each sample.**

- **<u>Turning your original variables into a smaller number of "Principal Components"</u>**

Principal Component Analysis (PCA)

- **PCA is a type of linear transformation on a given data set that has values for a certain number of variables (coordinates) for a certain amount of spaces.**

- **This linear transformation fits this dataset to a new coordinate system in such a way that the most significant variance is found on the first coordinate.**

- **In this way, you transform a set of x correlated variables over y samples to a set of p uncorrelated principal components over the same samples.**

- **Where many variables correlate with one another, they will all contribute strongly to the same principal component.**

Principal Component Analysis (PCA)

- **Let us use the mtcars dataset, which is built into R.**

- **This dataset consists of data on 32 models of car, taken from an American motoring magazine (1974 Motor Trend magazine). For each car, you have 11 features, expressed in varying units (US units).**

- **Compute the Principal Components**

- **Because PCA works best with numerical data, you'll exclude the two categorical variables (vs and am).**

Principal Component Analysis (PCA)

- **You are left with a matrix of 9 columns and 32 rows, which you pass to the prcomp() function, assigning your output to mtcars.pca.**

    > mtcars.pca <- prcomp(mtcars[,c(1:7,10,11)], center = TRUE,scale. = TRUE)

- **You will also set two arguments, center and scale, to be TRUE. Then you can have a peek at your PCA object with summary().**

    > summary(mtcars.pca)
    >str(mtcars.pca)

- **Let us draw biplot**

>biplot(mtcars.pca)
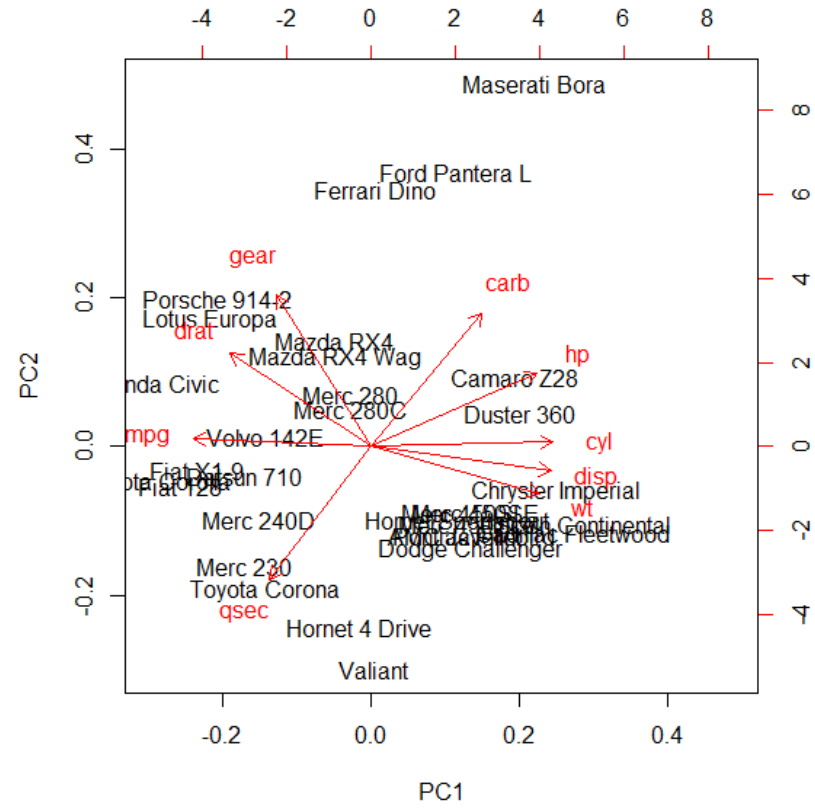
Exercise

- **install library ggfortify**

- **Load data iris**
    - **>data(iris)**

- **Perform the PCA analysis**

- **Develop biplot**

- **List highly correlated variables**

- **Display graph by use of following command :**

autoplot(iris.pca, data = iris, colour = 'Species')

autoplot(iris.pca, data = iris, colour = 'Species', label = TRUE, label.size = 3)

R Linear Regression

- **What is Linear Regression?**

  Regression analysis is a statistical technique for determining the relationship between two or more than two variables.

- **There are two types of variables in regression analysis –**

  independent variable and dependent variable.

- **Linear Regression is of the following two types:**

- **Simple Linear Regression – Based on the value of the single explanatory variable, the value of the response variable changes.**

- **Multiple Linear Regression – The value is dependent upon more than one explanatory variables in case of multiple linear regression.**

R Linear Regression

- Linear regression is one of the most basic statistical models out there.

- A linear regression can be calculated in R with the command lm.

- In the next example, use this command to calculate the height based on the age of the child.

$$Height = a + Age * b$$

- In this case, "a" and "b" are called the intercept and the slope respectively.

- With the same example, "a" or the intercept, is the value from where you start measuring.

- The slope measures the change of height with respect to the age in months. In general, for every month older the child is, his or her height will increase with "b".

Extracting information from a linear regression

- **x <- lm(y~x, data=z)**

- **summary(x) comprehensive summary of results**

- **print(x) precise version of the object**

- **deviance(x) residuals**

- **plot(x) returns plots: residuals, fitted values and some diagnostics**

- **coef(x) extract regression coefficients**

R Linear Regression : example

Let us download the data to an object called ageandheight and then create the linear regression in the third line. The lm command takes the variables in the format:

lm([target variable] ~ [predictor variables], data = [data source])

> library(readxl)

> ageandheight <- read_excel("ageandheight.xls", sheet = "Hoja2")

> lmHeight = lm(height~age, data = ageandheight)

> summary(lmHeight) #Review the results

```
> ageandheight <- read_excel("ageandheight.xls", sheet = "Hoja2")
> head(ageandheight)
# A tibble: 6 x 3
    age height no_siblings
  <dbl>  <dbl>       <dbl>
1    18   76.1           1
2    19   77             2
3    20   78.1           3
4    21   78.2           2
5    22   78.8           0
6    23   79.7           1
> lmHeight = lm(height~age, data = ageandheight)
> summary(lmHeight)

Call:
lm(formula = height ~ age, data = ageandheight)

Residuals:
    Min      1Q  Median      3Q     Max
-66.059   5.193   5.810   6.617   7.533

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  64.8445    43.3815   1.495    0.166
age           0.3835     1.8264   0.210    0.838

Residual standard error: 21.84 on 10 degrees of freedom
Multiple R-squared:  0.00439,    Adjusted R-squared:  -0.09517
F-statistic: 0.0441 on 1 and 10 DF,  p-value: 0.8379
```

```
Call:
lm(formula = height ~ age, data = ageandheight)

Residuals:
    Min       1Q   Median       3Q      Max
 -66.059    5.193    5.810    6.617    7.533

Coefficients:
              Estimate  Std. Error  t value  Pr(>|t|)
(Intercept)   64.8445     43.3815     1.495     0.166
age            0.3835      1.8264     0.210     0.838

Residual standard error: 21.84 on 10 degrees of freedom
Multiple R-squared:  0.00439,   Adjusted R-squared:  -0.09517
F-statistic: 0.0441 on 1 and 10 DF,  p-value: 0.8379
```

Coefficients.

In the orange square, you can see the values of the intercept ("a" value) and the slope ("b" value) for the age. These "a" and "b" values plot a line between all the points of the data.

Exercise

1) **Open Rstudio**

2) **Run small code (given in exercise)**

3) **Select option : html, pdf or doc**

4) **Save file**

5) **Show your reports**

PLINK : GWAS DATA

PLINK : Introduction

- **PLINK is a free, open-source designed to perform a range of basic, large-scale analyses in a computationally efficient manner.**

- **PLINK is whole genome association analysis tool.**

- **PLINK has a well documented manual.**

- **Available for linux, MAC ansd MAC-DOS.**

- **Command line version is faster than graphical PLINK.**

PLINK : Multi-feature tool

- **Merge two or more files**

- **Extracts subsets (SNPs or individuals)**

- **Compress data in a binary file format**

- **PLINK has numerous useful features for managing and analyzing genetic data**

- **Read data in a variety of formats**

- **Recode and reorder files**

Input Files

- **Genotype data is a text file**

- **Pedigree file (.ped)**

- **Map file (.map)**

- **Genotype data is a compressed binary file**

- **Fam File (.fam)**

- **Bim file (.bim)**

- **Bed file (.bed)**

PED Input File

Pedigree File - the first six columns are mandatory:

- **Family ID**

- **Individual ID**

- **Paternal ID**

- **Maternal ID**

- **Sex (1=male; 2=female; other=unknown)**

- **Phenotype**

Column1	Column2	Column3	Column4	Column5	Column6				
1	1	0	0	1	1	A	A	G	T
2	1	0	0	1	1	A	C	T	G
3	1	0	0	1	1	C	C	G	G
4	1	0	0	1	2	A	C	T	T
5	1	0	0	1	2	C	C	G	T

MAP Input File

MAP File has 4 columns:

- chromosome (1-22, X, Y or 0 if unplaced)

- rs# or snp identifier

- Genetic distance (morgans)

- Base-pair position (bp units)

Column1 Column2 Column3 Column4

```
1 snp1 0 1
1 snp2 0 2
```

Others Input File

*.ped

FID	IID	PID	MID	Sex	P	rs1	rs2	rs3
1	1	0	0	2	1	CT	AG	AA
2	2	0	0	1	0	CC	AA	AC
3	3	0	0	1	1	CC	AA	AC

*.map

Chr	SNP	GD	BPP
1	rs1	0	870000
1	rs2	0	880000
1	rs3	0	890000

*.fam

FID	IID	PID	MID	Sex	P
1	1	0	0	2	1
2	2	0	0	1	0
3	3	0	0	1	1

*.bed

Contains binary version of the SNP info of the *.ped file. (not in a format readable for humans)

*.bim

Chr	SNP	GD	BPP	Allele 1	Allele 2
1	rs1	0	870000	C	T
1	rs2	0	880000	A	G
1	rs3	0	890000	A	C

Covariate file

FID	IID	C1	C2	C3
1	1	0.00812835	0.00606235	-0.000871105
2	2	-0.0600943	0.0318994	-0.0827743
3	3	-0.0431903	0.00133068	-0.000276131

Legend			
FID	Family ID	rs{x}	Alleles per subject per SNP
IID	Individual ID	Chr	Chromosome
PID	Paternal ID	SNP	SNP name
MID	Maternal ID	GD	Genetic distance (morgans)
Sex	Sex of subject	BPP	Base-pair position (bp units)
P	Phenotype	C{x}	Covariates (e.g., Multidimensional Scaling (MDS) components)

- **Type command :**
  plink --file test –noweb

Provide you detail information of SNPs count, individuals count and check for the missingness and frequency test

```
Options in effect:
        --file test
        --noweb

2 (of 2) markers to be included from [ test.map ]
6 individuals read from [ test.ped ]
6 individuals with nonmissing phenotypes
Assuming a disease phenotype (1=unaff, 2=aff, 0=miss)
Missing phenotype value is also -9
3 cases, 3 controls and 0 missing
6 males, 0 females, and 0 of unspecified sex
Before frequency and genotyping pruning, there are 2 SNPs
6 founders and 0 non-founders found
Total genotyping rate in remaining individuals is 1
0 SNPs failed missingness test ( GENO > 1 )
0 SNPs failed frequency test ( MAF < 0 )
After frequency and genotyping pruning, there are 2 SNPs
After filtering, 3 cases, 3 controls and 0 missing
After filtering, 6 males, 0 females, and 0 of unspecified sex
```

Data Transformation

- **Convert map and ped into binary format**

  ➤ *plink.exe --file test --make-bed --out test --noweb*

- **Read binary file in plink**

  ➤ *plink.exe –bfile test*

```
Reading map (extended format) from [ test.bim ]
2 markers to be included from [ test.bim ]
Reading pedigree information from [ test.fam ]
6 individuals read from [ test.fam ]
6 individuals with nonmissing phenotypes
Assuming a disease phenotype (1=unaff, 2=aff, 0=miss)
Missing phenotype value is also -9
3 cases, 3 controls and 0 missing
6 males, 0 females, and 0 of unspecified sex
Reading genotype bitfile from [ test.bed ]
Detected that binary PED file is v1.00 SNP-major mode
Before frequency and genotyping pruning, there are 2 SNPs
6 founders and 0 non-founders found
Total genotyping rate in remaining individuals is 1
0 SNPs failed missingness test ( GENO > 1 )
0 SNPs failed frequency test ( MAF < 0 )
After frequency and genotyping pruning, there are 2 SNPs
After filtering, 3 cases, 3 controls and 0 missing
After filtering, 6 males, 0 females, and 0 of unspecified sex

Analysis finished: Sun Oct 20 18:49:10 2019
```

Comparision

```
2 (of 2) markers to be included from [ test.map ]
6 individuals read from [ test.ped ]
6 individuals with nonmissing phenotypes
Assuming a disease phenotype (1=unaff, 2=aff, 0=miss)
Missing phenotype value is also -9
3 cases, 3 controls and 0 missing
6 males, 0 females, and 0 of unspecified sex
Before frequency and genotyping pruning, there are 2 SNPs
6 founders and 0 non-founders found
Total genotyping rate in remaining individuals is 1
0 SNPs failed missingness test ( GENO > 1 )
0 SNPs failed frequency test ( MAF < 0 )
After frequency and genotyping pruning, there are 2 SNPs
After filtering, 3 cases, 3 controls and 0 missing
After filtering, 6 males, 0 females, and 0 of unspecified sex
```

TEXT output

```
Reading map (extended format) from [ test.bim ]
2 markers to be included from [ test.bim ]
Reading pedigree information from [ test.fam ]
6 individuals read from [ test.fam ]
6 individuals with nonmissing phenotypes
Assuming a disease phenotype (1=unaff, 2=aff, 0=miss)
Missing phenotype value is also -9
3 cases, 3 controls and 0 missing
6 males, 0 females, and 0 of unspecified sex
Reading genotype bitfile from [ test.bed ]
Detected that binary PED file is v1.00 SNP-major mode
Before frequency and genotyping pruning, there are 2 SNPs
6 founders and 0 non-founders found
Total genotyping rate in remaining individuals is 1
0 SNPs failed missingness test ( GENO > 1 )
0 SNPs failed frequency test ( MAF < 0 )
After frequency and genotyping pruning, there are 2 SNPs
After filtering, 3 cases, 3 controls and 0 missing
After filtering, 6 males, 0 females, and 0 of unspecified sex

Analysis finished: Sun Oct 20 18:49:10 2019
```

BED OUTPUT

QC of genetic DATA

- **A vital step that should be part of any GWAS is the use of appropriate QC.**

- **Without extensive QC, GWAS will not generate reliable results because raw genotype data are inherently imperfect.**

- **Errors in the data can arise for numerous reasons, for example, due to poor quality of DNA samples, poor DNA hybridization to the array, poorly performing genotype probes, and sample mix-ups or contamination.**

QC of genetic DATA

The seven QC steps consist of filtering out of SNPs and individuals based on the following:

(1) individual and SNP missingness,

(2) inconsistencies in assigned and genetic sex of subjects (see sex discrepancy),

(3) minor allele frequency (MAF),

(4) deviations from Hardy–Weinberg equilibrium (HWE),

(5) heterozygosity rate,

(6) relatedness, and

(7) ethnic outliers (see population stratification).

Important Commands

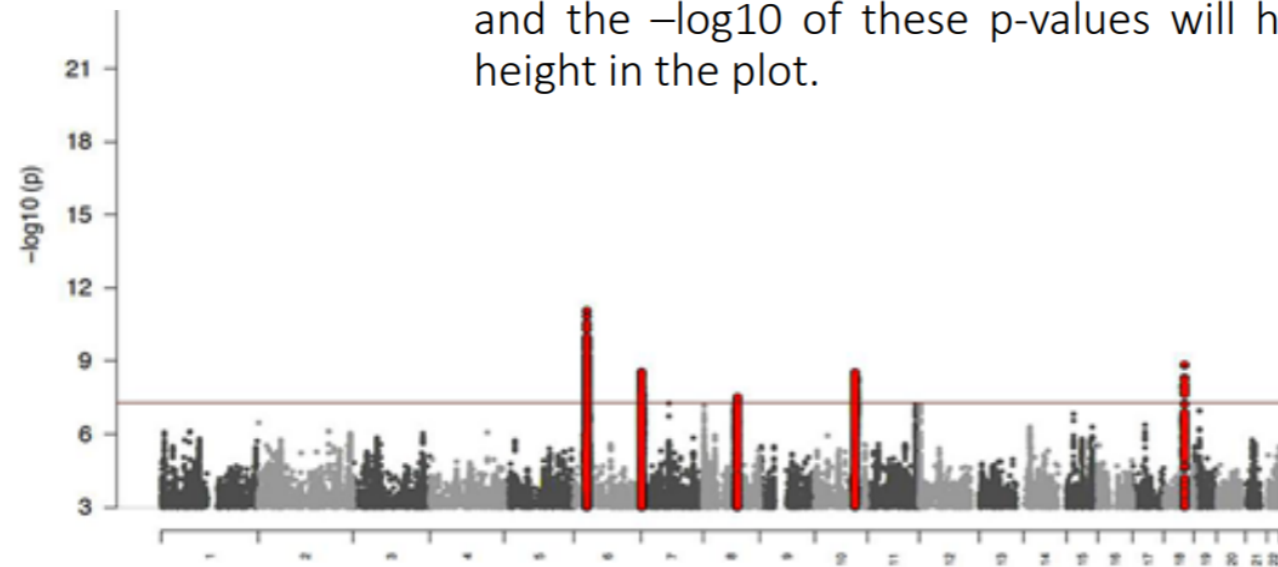| Step	Command	Function
1: Missingness of SNPs and individuals	--geno	Excludes SNPs that are missing in a large proportion of the subjects. In this step, SNPs with low genotype calls are removed.
	--mind	**Excludes individuals who have high rates of genotype missingness. In this step, individual with low genotype calls are removed.**
2: Sex discrepancy	--check-sex	**Checks for discrepancies between sex of the individuals recorded in the dataset and their sex based on X chromosome heterozygosity/homozygosity rates.**
3: Minor allele frequency (MAF)	--maf	**Includes only SNPs above the set MAF threshold.**
4: Hardy–Weinberg equilibrium (HWE)	--hwe	**Excludes markers which deviate from Hardy–Weinberg equilibrium.**
5: Heterozygosity		**Excludes individuals with high or low heterozygosity rates**
6: Relatedness	--genome	**Calculates identity by descent (IBD) of all sample pairs.**
	--min	**Sets threshold and creates a list of individuals with relatedness above the chosen threshold. Meaning that subjects who are related at, for example, pi-hat >0.2 (i.e., second degree relatives) can be detected.**
7: Population stratification	--genome	**Calculates identity by descent (IBD) of all sample pairs.**
	--cluster --mds-plot k	**Produces a k-dimensional representation of any substructure in the data, based on IBS.**

-- assoc

- **It works with case/control and continuous phenotypes.**

- **Case-control (1df chi-square test, outputs assoc)**

- **PLINK will recognise this is a case/control analysis because the phenotype just has:**

- **1 (for controls),**

- **2 (for cases), and**

- **0/-9/non-numeric (for missing).**

- **Given a quantitative phenotype, --assoc writes regression statistics and Wald test results to plink.qassoc.**

Manhattan plot

- Plots the -log10 of the association p-value for each SNP against the genomic coordinates.
- The strongest associations will have the smallest p-values and the −log10 of these p-values will have the highest height in the plot.

Useful Link

https://jokergoo.github.io/circlize_book/book/

https://igraph.org/r/doc/aaa-igraph-package.html

https://www.statmethods.net/advgraphs/correlograms.html