

Introduction to R

Boby Mathew

April 25, 2018

'What is data?"

Data definition

"Information, especially facts or numbers, collected to be examined and considered and used to help decision-making" : Cambridge dictionary.

Different form of data



Different form of data



Different form of data



Different form of data



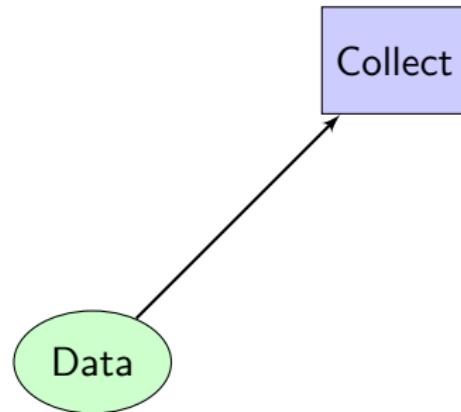
Data analysis

"Analysis of data is a process of inspecting, cleaning, transforming, and modeling data with the goal of discovering useful information, suggesting conclusions, and supporting decision-making". Wikipedia

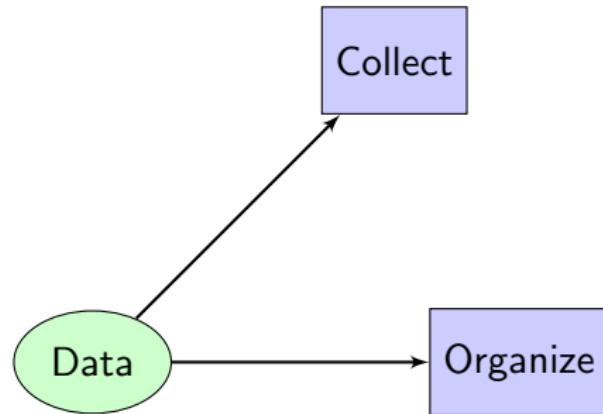
Different steps in analysis



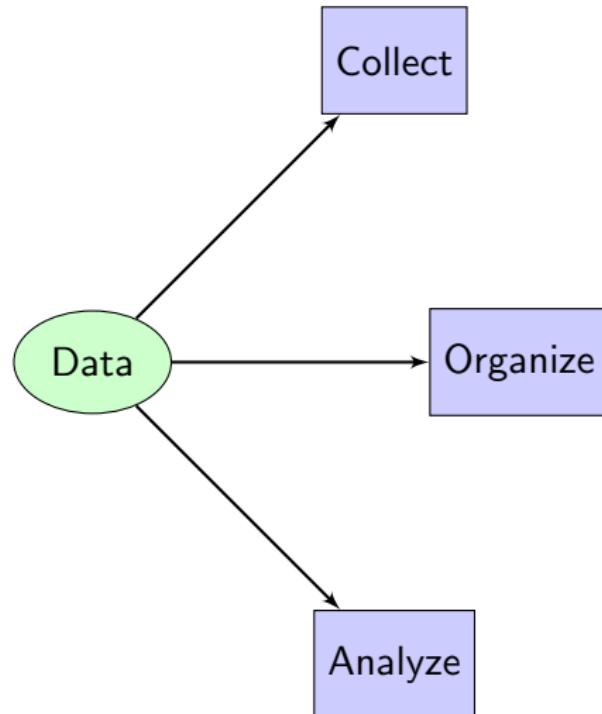
Different steps in analysis



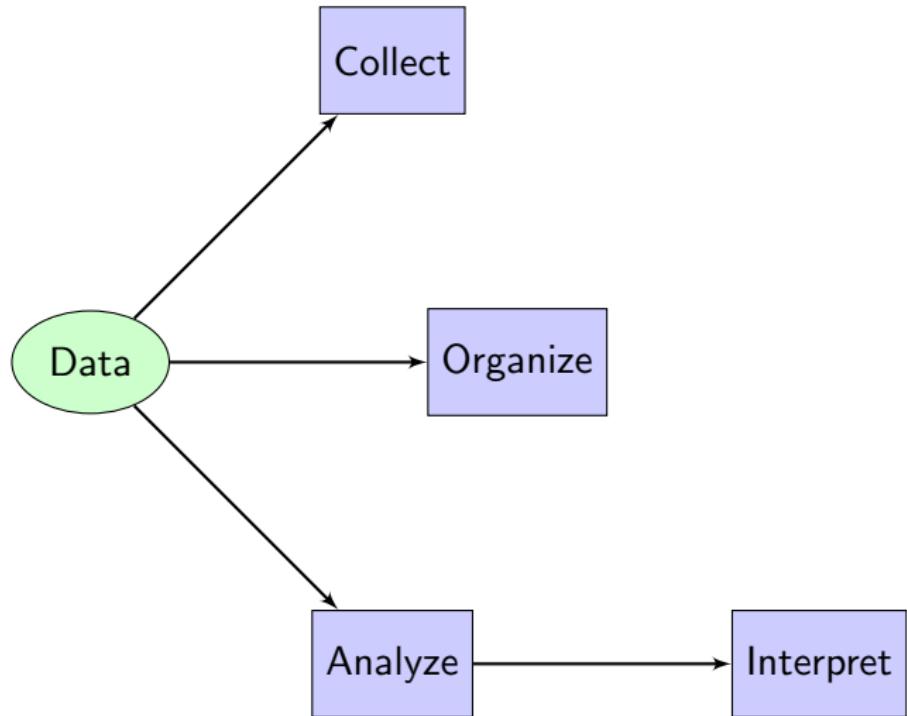
Different steps in analysis



Different steps in analysis



Different steps in analysis



Right tool???



Common tools

- Excel

Common tools

- Excel
 - Easy to use and plot.

Common tools

- Excel
 - Easy to use and plot.
 - Slow and cannot handle complex analysis.

Common tools

- Excel
 - Easy to use and plot.
 - Slow and cannot handle complex analysis.
- Matlab=Matrix Laboratory

Common tools

- Excel
 - Easy to use and plot.
 - Slow and cannot handle complex analysis.
- Matlab=Matrix Laboratory
 - Powerful for matrix calculation, have good graphics functions.

Common tools

- Excel
 - Easy to use and plot.
 - Slow and cannot handle complex analysis.
- Matlab=Matrix Laboratory
 - Powerful for matrix calculation, have good graphics functions.
 - Not free and not good for statistical analysis.

Common tools

- Excel
 - Easy to use and plot.
 - Slow and cannot handle complex analysis.
- Matlab=Matrix Laboratory
 - Powerful for matrix calculation, have good graphics functions.
 - Not free and not good for statistical analysis.
- SAS

Common tools

- Excel
 - Easy to use and plot.
 - Slow and cannot handle complex analysis.
- Matlab=Matrix Laboratory
 - Powerful for matrix calculation, have good graphics functions.
 - Not free and not good for statistical analysis.
- SAS
 - Good GUI and have many functions for the statistical analysis.

Common tools

- Excel
 - Easy to use and plot.
 - Slow and cannot handle complex analysis.
- Matlab=Matrix Laboratory
 - Powerful for matrix calculation, have good graphics functions.
 - Not free and not good for statistical analysis.
- SAS
 - Good GUI and have many functions for the statistical analysis.
 - Expensive and poor graphical capabilities

Common tools

- Excel
 - Easy to use and plot.
 - Slow and cannot handle complex analysis.
- Matlab=Matrix Laboratory
 - Powerful for matrix calculation, have good graphics functions.
 - Not free and not good for statistical analysis.
- SAS
 - Good GUI and have many functions for the statistical analysis.
 - Expensive and poor graphical capabilities
- Python

Common tools

- Excel
 - Easy to use and plot.
 - Slow and cannot handle complex analysis.
- Matlab=Matrix Laboratory
 - Powerful for matrix calculation, have good graphics functions.
 - Not free and not good for statistical analysis.
- SAS
 - Good GUI and have many functions for the statistical analysis.
 - Expensive and poor graphical capabilities
- Python
 - Open source and powerful

Common tools

- Excel
 - Easy to use and plot.
 - Slow and cannot handle complex analysis.
- Matlab=Matrix Laboratory
 - Powerful for matrix calculation, have good graphics functions.
 - Not free and not good for statistical analysis.
- SAS
 - Good GUI and have many functions for the statistical analysis.
 - Expensive and poor graphical capabilities
- Python
 - Open source and powerful
 - Not easy to learn and support is bad

War of tools. <https://media.licdn.com>



Why R???

- R is a free open source statistical programming package.

Why R???

- R is a free open source statistical programming package.
- There are versions available for Windows, Linux and Mac.

Why R???

- R is a free open source statistical programming package.
- There are versions available for Windows, Linux and Mac.
- R offers many statistical functions they are already built in.

Why R???

- R is a free open source statistical programming package.
- There are versions available for Windows, Linux and Mac.
- R offers many statistical functions they are already built in.
- R offers many powerful graphical tools

Advantages of R!!

- Easy to write your own R scripts and share with other people.

Advantages of R!!

- Easy to write your own R scripts and share with other people.
- R user community is very large and the support is really good.

Advantages of R!!

- Easy to write your own R scripts and share with other people.
- R user community is very large and the support is really good.
- It provided functions for most of the standard statistical analysis.

Advantages of R!!

- Easy to write your own R scripts and share with other people.
- R user community is very large and the support is really good.
- It provided functions for most of the standard statistical analysis.
- R is cross-platform: means can be run on different operating systems.

R work space

- Need time and effort to learn R.

R work space

- Need time and effort to learn R.
- Sometimes the error messages are not easy to understand.

R work space

- Need time and effort to learn R.
- Sometimes the error messages are not easy to understand.
- There is no one to complain if some programs does not work.

R work space

- Need time and effort to learn R.
- Sometimes the error messages are not easy to understand.
- There is no one to complain if some programs does not work.
- R is an interpreted language and can be slow when you have complicated scripts.

R work space

- Need time and effort to learn R.
- Sometimes the error messages are not easy to understand.
- There is no one to complain if some programs does not work.
- R is an interpreted language and can be slow when you have complicated scripts.
- Minimal GUI.

Got to the link <https://www.r-project.org/>



[Home]

Download

CRAN

R Project

About R

Logo

Contributors

What's New?

Reporting Bugs

Development Site

Conferences

Search

R Foundation

Foundation

Board

Members

Donors

Donate

Help With R

Getting Help

Documentation

The R Project for Statistical Computing

Getting Started

R is a free software environment for statistical computing and graphics. It compiles and runs on a wide variety of UNIX platforms, Windows and MacOS. To download R, please choose your preferred CRAN mirror.

If you have questions about R like how to download and install the software, or what the license terms are, please read our answers to frequently asked questions before you send an email.

News

- The [useR! 2017](#) conference will take place in Brussels, July 4 - 7, 2017, and details will be appear here in due course.
- R version 3.3.1 ([Bug in Your Hair](#)) has been released on Tuesday 2016-06-21.
- R version 3.2.5 ([Very, Very Secure Dishes](#)) has been released on 2016-04-14. This is a rebadging of the quick-fix release 3.2.4-revised.
- [Notice XQuartz users \(Mac OS X\)](#) A security issue has been detected with the Sparkle update mechanism used by XQuartz. Avoid updating over insecure channels.
- The [R Logo](#) is available for download in high-resolution PNG or SVG formats.
- [useR! 2016](#), has taken place at Stanford University, CA, USA, June 27 - June 30, 2016.
- The [R Journal Volume 7/2](#) is available.
- R version 3.2.3 ([Wooden Christmas-Tree](#)) has been released on 2015-12-10.
- R version 3.1.3 ([Smooth Sidewalk](#)) has been released on 2015-03-09.



Click Download R

CRAN Mirrors

The Comprehensive R Archive Network is available at the following URLs, please choose a location close to you. Some statistics on the status of the mirrors can be found here: [main page](#), [windows release](#).

0-Cloud

<https://cloud.r-project.org/>
<http://cloud.r-project.org/>

Automatic redirection to servers worldwide, currently sponsored by Rstudio
Automatic redirection to servers worldwide, currently sponsored by Rstudio

Algeria

<https://cran.usthb.dz/>
<http://cran.usthb.dz/>

University of Science and Technology Houari Boumediene
University of Science and Technology Houari Boumediene

Argentina

<http://mirror.fcaglp.unlp.edu.ar/CRAN/>

Universidad Nacional de La Plata

Australia

<http://cran.csiro.au/>
<https://cran.ms.unimelb.edu.au/>
<http://cran.ms.unimelb.edu.au/>
<https://cran.curtin.edu.au/>

CSIRO
University of Melbourne
University of Melbourne
Curtin University of Technology

Austria

<https://cran.wu.ac.at/>
<http://cran.wu.ac.at/>

Wirtschaftsuniversität Wien
Wirtschaftsuniversität Wien

Belgium

<http://www.freestatistics.org/cran/>
<https://lib.ugent.be/CRAN/>
<http://lib.ugent.be/CRAN/>

K.U.Leuven Association
Ghent University Library
Ghent University Library

Brazil

<http://nbcqib.uesc.br/mirrors/cran/>
<http://cran.r.csl.ufpr.br/>
<http://cran.fluxcruz.br/>
<https://vps.fmvz.usp.br/CRAN/>
<http://vps.fmvz.usp.br/CRAN/>

Center for Comp. Biol. at Universidade Estadual de Santa Cruz
Universidade Federal do Parana
Oswaldo Cruz Foundation, Rio de Janeiro
University of Sao Paulo, Sao Paulo
University of Sao Paulo, Sao Paulo

Bulgaria

<https://ftp.uni-sofia.bg/CRAN/>
<http://ftp.uni-sofia.bg/CRAN/>

Sofia University
Sofia University

Canada

<http://cran.stat.sfu.ca/>
<https://mung.ca/mirror/cran/>
<http://mung.ca/mirror/cran/>
<http://mirror.its.dal.ca/cran/>
<http://cran.utstat.utoronto.ca/>

Simon Fraser University, Burnaby
Manitoba Unix User Group
Manitoba Unix User Group
Dalhousie University, Halifax
University of Toronto

Chile



Choose Download R for windows

The Comprehensive R Archive Network

Download and Install R

Precompiled binary distributions of the base system and contributed packages. **Windows and Mac** users most likely want one of these versions of R:

- [Download R for Linux](#)
- [Download R for \(Mac\) OS X](#)
- [Download R for Windows](#)

R is part of many Linux distributions, you should check with your Linux package management system in addition to the link above.

Source Code for all Platforms

Windows and Mac users most likely want to download the precompiled binaries listed in the upper box, not the source code. The sources have to be compiled before you can use them. If you do not know what this means, you probably do not want to do it!

- The latest release (Tuesday 2016-06-21, Bug in Your Hair) [R-3.3.1.tar.gz](#), read [what's new](#) in the latest version.
- Sources of [R alpha and beta releases](#) (daily snapshots, created only in time periods before a planned release).
- Daily snapshots of current patched and development versions are [available here](#). Please read about [new features and bug fixes](#) before filing corresponding feature requests or bug reports.
- Source code of older versions of R is [available here](#).
- Contributed extension [packages](#)

Questions About R

- If you have questions about R like how to download and install the software, or what the license terms are, please read our [answers to frequently asked questions](#) before you send an email.

What are R and CRAN?

R is 'GNU S', a freely available language and environment for statistical computing and graphics which provides a wide variety of statistical and graphical techniques: linear and nonlinear modelling, statistical tests, time series analysis, classification, clustering, etc. Please consult the [R project homepage](#) for further information.

CRAN is a network of ftp and web servers around the world that store identical, up-to-date, versions of code and documentation for R. Please use the CRAN [mirror](#) nearest to you to minimize network load.

Submitting to CRAN

To "submit" a package to CRAN, check that your submission meets the [CRAN Repository Policy](#) and then use the [web form](#).

If this fails, upload to <http://CRAN.R-project.org/incoming> and send an email to CRAN@R-project.org following the policy. Please do not attach submissions to emails, because



Click [Install R for the first time](#)

R for Windows

Subdirectories:

[base](#)

Binaries for base distribution (managed by Duncan Murdoch). This is what you want to [install R for the first time](#).

[contrib](#)

Binaries of contributed CRAN packages (for R >= 2.11.x; managed by Uwe Ligges). There is also information on [third party software](#) available for CRAN Windows services and corresponding environment and make variables.

[old contrib](#)

Binaries of contributed CRAN packages for outdated versions of R (for R < 2.11.x; managed by Uwe Ligges).

[Rtools](#)

Tools to build R and R packages (managed by Duncan Murdoch). This is what you want to build your own packages on Windows, or to build R itself.

Please do not submit binaries to CRAN. Package developers might want to contact Duncan Murdoch or Uwe Ligges directly in case of questions / suggestions related to Windows binaries.

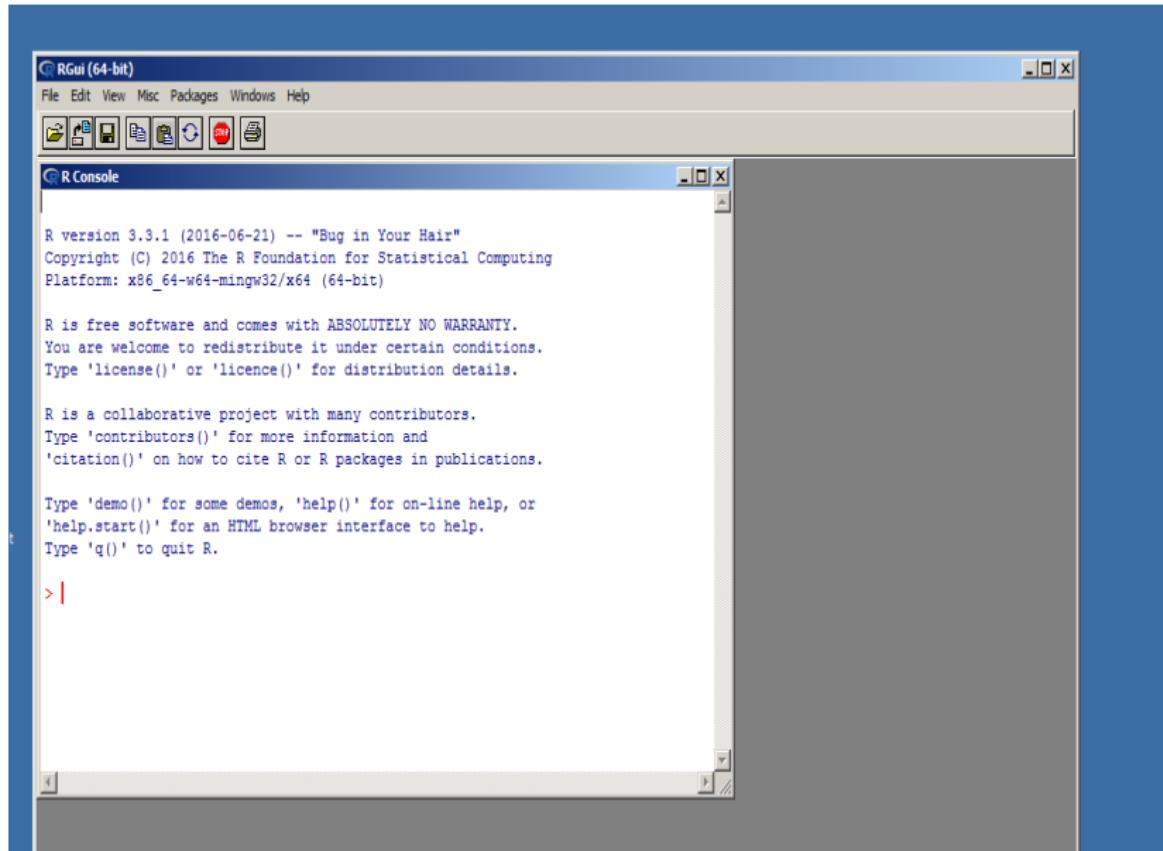
You may also want to read the [R FAQ](#) and [R for Windows FAQ](#).

Note: CRAN does some checks on these binaries for viruses, but cannot give guarantees. Use the normal precautions with downloaded executables.

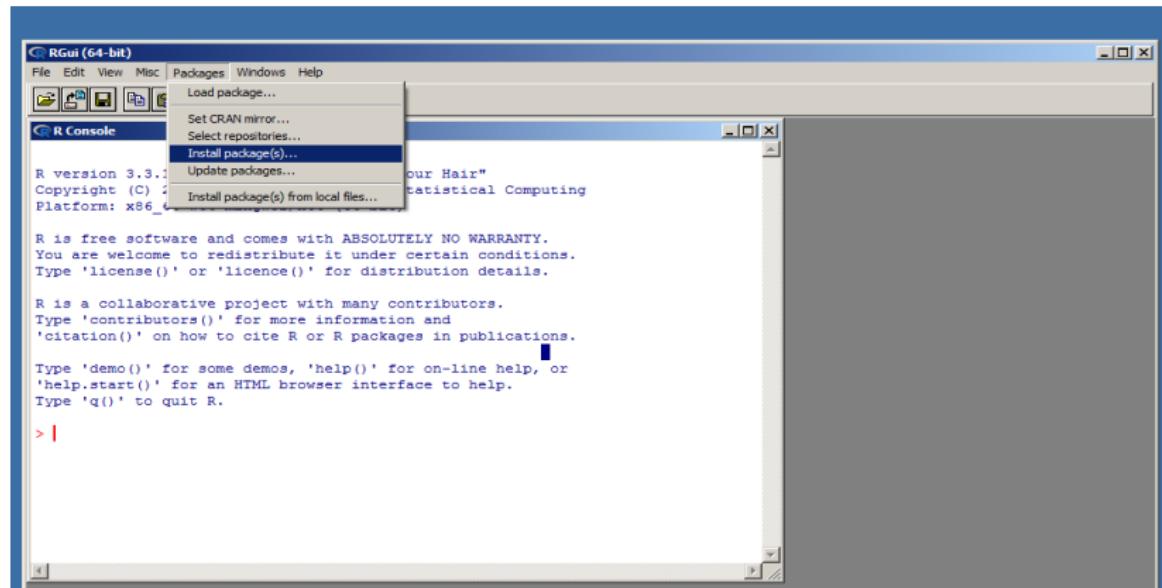
Starting R



GUI for R

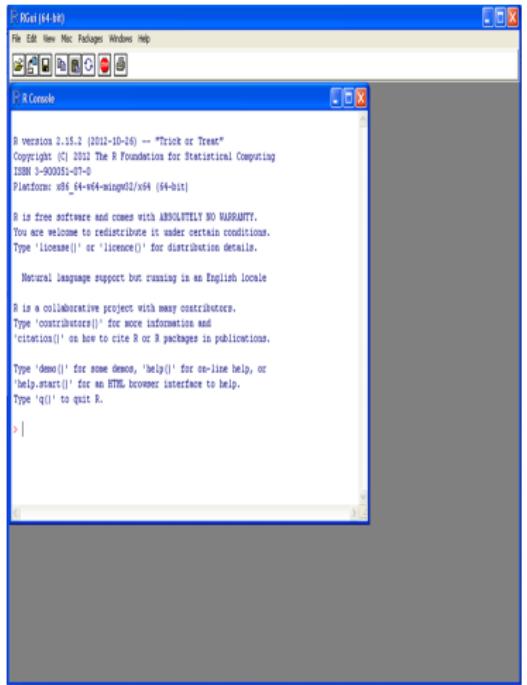


Install new package



R work space

- R calculations not stored in the permanent storage.
- If you want to use the tables or results later you should store them before closing the R GUI.
- When you close the GUI, the system will ask whether you want to save the workspace
- You can save the workspace in .Rdata file(as a binary file)



R basics

R command prompt start with > sign and there you can type your commands

```
>1+1
```

R basics

R command prompt start with > sign and there you can type your commands

```
>1+1
```

'Enter' key will print the result

```
> 1+1
```

```
[1] 2
```

R basics

R command prompt start with `>` sign and there you can type your commands

```
>1+1
```

'Enter' key will print the result

```
> 1+1
```

```
[1] 2
```

Here the result is not stored. We can store the result into a variable called '`x`' using '`=`' or '`<-`' sign

```
> x=1+1
```

```
> x<-1+1
```

R basics

R command prompt start with `>` sign and there you can type your commands

```
>1+1
```

'Enter' key will print the result

```
> 1+1
```

```
[1] 2
```

Here the result is not stored. We can store the result into a variable called '`x`' using '`=`' or '`<-`' sign

```
> x=1+1
```

```
> x<-1+1
```

Later you can see the result by typing `x` in the prompt.

```
> x
```

```
[1] 2
```

R Variables

"In programming languages, a variable can store a value and that can be changed any time". In the above example x is a variable. In every programing language there are some rules to define a variable.

R Variables

"In programming languages, a variable can store a value and that can be changed any time". In the above example x is a variable. In every programming language there are some rules to define a variable.

In R the variable name should start with A-Z or a-z

```
_x = 20
```

Error: unexpected input in "_"

R Variables

"In programming languages, a variable can store a value and that can be changed any time". In the above example x is a variable. In every programming language there are some rules to define a variable.

In R the variable name should start with A-Z or a-z

```
_x = 20
```

Error: unexpected input in "_"

In R variables are case sensitive

```
x = 1 + 1
```

```
X
```

Error: object 'X' not found

R Variables

"In programming languages, a variable can store a value and that can be changed any time". In the above example x is a variable. In every programming language there are some rules to define a variable.

In R the variable name should start with A-Z or a-z

```
_x = 20
```

Error: unexpected input in "_"

In R variables are case sensitive

```
x = 1 + 1
```

```
X
```

Error: object 'X' not found

You cannot start variable name with a number

```
1x = 2
```

Error: unexpected symbol in "1x"

R basics

But from the second position onwards you can use numbers or special characters to name them.

```
> x1=2
```

```
> x1
```

```
[1] 2
```

```
> x_=3
```

```
> x_
```

```
[1] 3
```

R basics

But from the second position onwards you can use numbers or special characters to name them.

```
> x1=2  
> x1  
[1] 2  
  
> x_=3  
> x_  
[1] 3
```

There are some variables already assigned in R

```
> pi  
[1] 3.141593  
  
> month.name  
[1] "January"    "February"   "March"      "April"       "May"
```

R functions

- A function is a piece of code designed to do specific task.

The screenshot shows a Microsoft Excel spreadsheet. The table has two columns: 'Students' (Column A) and 'Marks' (Column B). The data includes 12 rows of student names and their corresponding marks. Row 13 is labeled 'Average' and contains the formula '=AVERAGE(B2:B12)' in its formula bar, which is highlighted with a red box. The cell containing the result, 61.90909, is also highlighted with a red box.

	B13	f _x	=AVERAGE(B2:B12)
1	Students	Marks	
2	Alex	85	
3	Laura	55	
4	Tim	65	
5	Mitchelle	73	
6	Christian	42	
7	Gracy	55	
8	Kate	40	
9	William	77	
10	Dora	30	
11	Ana	95	
12	Mili	64	
13	Average	61.90909	

- Here AVERAGE is the function name and B2:B12 is the argument
- R also provide many functions like Excel and the general format is **name of the function (argument)**: In R, () is always associated with function call.

R data types

Main data types in R are scalar, vector, matrix and data frames.

Scalar is a variable that contains an individual value.

```
> x=2  
> x  
[1] 2
```

R data types

Main data types in R are scalar, vector, matrix and data frames.

Scalar is a variable that contains an individual value.

```
> x=2  
> x  
[1] 2
```

Boolean data type having two values TRUE or FALSE

```
> B=c(TRUE,TRUE,TRUE, FALSE, TRUE, FALSE)  
> B  
[1] TRUE TRUE TRUE FALSE TRUE FALSE
```

R data types

1	
2	
3	
4	
5	
6	
7	
8	
-	

R data types

1	
2	
3	
4	
5	
6	
7	
8	
-	

A vector is a one dimensional array

```
> A=c(12,13,14,15,17,18)
```

```
> A
```

```
[1] 12 13 14 15 17 18
```

R data types

[] is used to access the elements of a vector

```
> A
```

```
[1] 12 13 14 15 17 18
```

```
> A[1]
```

```
[1] 12
```

```
> A[4]
```

```
[1] 15
```

```
> A[10]
```

```
[1] NA
```

R data types

`[-]` is used to exclude the elements of a vector

```
> A
```

```
[1] 12 13 14 15 17 18
```

```
> A[1:3]
```

```
[1] 12 13 14
```

```
> A[-1]
```

```
[1] 13 14 15 17 18
```

```
> A[-(1:2)]
```

```
[1] 14 15 17 18
```

R data types

Working with vectors

```
> A  
[1] 12 13 14 15 17 18  
  
> sum(A)  
[1] 89  
  
> sort(A)  
[1] 12 13 14 15 17 18  
  
> sort(A,decreasing=T)  
[1] 18 17 15 14 13 12  
  
> mean(A)  
[1] 14.83333
```

Matrices

Create a vector

```
> A=c(12,13,14,15,17,18,20,21,22)  
> A  
[1] 12 13 14 15 17 18 20 21 22
```

Matrices

Create a vector

```
> A=c(12,13,14,15,17,18,20,21,22)  
> A  
[1] 12 13 14 15 17 18 20 21 22
```

Convert to a matrix

```
> H=matrix(A,nrow=3)  
> H  
[,1] [,2] [,3]  
[1,] 12 15 20  
[2,] 13 17 21  
[3,] 14 18 22
```

Matrices

Matrices have two dimension (row and column), so [] require two values.

```
> H
```

```
 [,1] [,2] [,3]  
[1,] 12   15   20  
[2,] 13   17   21  
[3,] 14   18   22
```

```
> H[1,3]
```

```
[1] 20
```

```
> H[3,1]
```

```
[1] 14
```

Matrices

To access the row

```
> H
```

```
      [,1] [,2] [,3]  
[1,]    12    15    20  
[2,]    13    17    21  
[3,]    14    18    22
```

```
> H[1,]
```

```
[1] 12 15 20
```

Matrices

To access the row

```
> H
```

```
      [,1] [,2] [,3]  
[1,]    12    15    20  
[2,]    13    17    21  
[3,]    14    18    22
```

```
> H[1,]
```

```
[1] 12 15 20
```

To access the column

```
> H[,3]
```

```
[1] 20 21 22
```

List

List can hold different data structures like vector matrix string..

```
> lst=list(mat=H,year=2012,name="INRES")
```

List

List can hold different data structures like vector matrix string..

```
> lst=list(mat=H,year=2012,name="INRES")
```

To access the column

```
> lst  
  
$mat  
     [,1] [,2] [,3]  
[1,]    12    15    20  
[2,]    13    17    21  
[3,]    14    18    22
```

```
$year  
[1] 2012
```

```
$name  
[1] "INRES"
```

List

You can access the individual items of a list using \$ sign

```
> lst$mat
```

```
      [,1] [,2] [,3]  
[1,]    12    15    20  
[2,]    13    17    21  
[3,]    14    18    22
```

```
> lst$name
```

```
[1] "INRES"
```

```
> lst$year
```

```
[1] 2012
```

Data frame

- Data frame is used to handle Excel like tables in R.

Data frame

- Data frame is used to handle Excel like tables in R.
- Data frame is the one of the most commonly used data type in R

Data frame

- Data frame is used to handle Excel like tables in R.
- Data frame is the one of the most commonly used data type in R
- Data frames are matrix like but can have different values in different columns
- Each vector in data frame should have equal no of elements

Creating a data frame

First create the vectors

```
> line = c(1, 2, 3,4)  
> year = c(2001, 2002, 2003,2004)  
> obs = c(45, 55, 36,37)  
> dat = data.frame(Line=line, year=year, obs=obs)  
> dat
```

	Line	year	obs
1	1	2001	45
2	2	2002	55
3	3	2003	36
4	4	2004	37

Creating a data frame

```
> dat
```

	Line	year	obs
1	1	2001	45
2	2	2002	55
3	3	2003	36
4	4	2004	37

Line	year	obs
1	2001	45
2	2002	55
3	2003	36
4	2004	37

Creating a data frame

First create the vectors

```
line = c(1, 2, 3)
```

```
year = c(2001, 2002, 2003, 2004)
```

```
obs = c(45, 55, 36, 37)
```

```
dat = data.frame(Line=line, year=year, obs=obs)
```

```
Error in data.frame(Line = line, year = year, obs = obs) :  
arguments imply differing number of rows: 3, 4
```

"Reading tables into R work space!!!"

Example table: Excel table and the fields are tab separated

Number	Genotype	Location	Height		Yield
1	GE_1	Bonn	52.27573	18.911348	149.6148
2	GE_1	Cologne	53.35362	20.923011	151.0124
3	GE_2	Bonn	45.08251	14.31079	145.3708
4	GE_2	Cologne	52.47572	19.164253	152.5497
5	GE_3	Bonn	51.08301	21.76464	150.1298
6	GE_3	Cologne	47.48507	15.341368	147.4639
7	GE_4	Bonn	46.34643	15.990194	147.0186
8	GE_4	Cologne	56.21228	25.884787	153.6735
9	GE_5	Bonn	42.24373	10.963465	142.6255
10	GE_5	Cologne	45.69879	13.224158	143.3145

Example table: the fields are comma separated

Number	Genotype	Location	Height	Root_length	Yield
1	GE_1	Bonn	52.27573	18.911348	149.6148
2	GE_1	Cologne	53.35362	20.923011	151.0124
3	GE_2	Bonn	45.08251	14.31079	145.3708
4	GE_2	Cologne	52.47572	19.164253	152.5497
5	GE_3	Bonn	51.08301	21.76464	150.1298
6	GE_3	Cologne	47.48507	15.341368	147.4639
7	GE_4	Bonn	46.34643	15.990194	147.0186
8	GE_4	Cologne	56.21228	25.884787	153.6735
9	GE_5	Bonn	42.24373	10.963465	142.6255
10	GE_5	Cologne	45.69879	13.224158	143.3145

Reading data into R work space

	Genotype	Location	Height	Root_length	Yield
1	GE_1	Bonn	52.27573	18.911348	149.6148
2	GE_1	Cologne	53.35362	20.923011	151.0124
3	GE_2	Bonn	45.08251	14.31079	145.3708
4	GE_2	Cologne	52.47572	19.164253	152.5497
5	GE_3	Bonn	51.08301	21.76464	150.1298
6	GE_3	Cologne	47.48507	15.341368	147.4639
7	GE_4	Bonn	46.34643	15.990194	147.0186
8	GE_4	Cologne	56.21228	25.884787	153.6735
9	GE_5	Bonn	42.24373	10.963465	142.6255
10	GE_5	Cologne	45.69879	13.224158	NA

read.table() function

```
read.table(file, # Name of the file  
          header = FALSE, # whether the table has column names  
          Sep = "", # indicate how the fields are seperated  
          row.names, # indicate id the table has row names  
          na.strings = "NA") # how the missing values are coded.
```



Reading data into R work space

Rules to prepare the table

- The table should have equal number of rows/columns
- Missing values should be coded as "NA"
- No empty space is allowed for the missing values.
- The elements in the table should be separated by a unique separator (comma, tab or semicolon)

Reading data into R work space

What type of field separator is used?

- Space " "
- Tab "\t"
- Comma " , "
- Semicolon " ; "

Reading data into R work space

Check for the row and column names

- row names?
- column names/header ?
- missing values?

Reading data into R work space

Use `read.table()` function to import data into R

```
> dat=read.table("test.txt")
```

```
> dat
```

	V1	V2	V3	V4	V5	V6
1	Number	Genotype	Location	Treatment	Height	Root_length
2	1	GE_1	Bonn	T	22.27573	18.911348
3	2	GE_1	Bonn	N	53.35362	20.923011
4	3	GE_1	Cologne	T	45.08251	14.31079
5	4	GE_1	Cologne	N	52.47572	19.164253
6	5	GE_2	Bonn	T	21.08301	21.76464
7	6	GE_2	Bonn	N	47.48507	15.341368
8	7	GE_2	Cologne	T	46.34643	15.990194
9	8	GE_2	Cologne	N	56.21228	25.884787
10	9	GE_3	Bonn	T	22.24373	10.963465
11	10	GE_3	Bonn	N	45.69879	13.224158

Reading data into R work space

Use option *header = T*

```
> dat=read.table("test.txt",header=T)  
> dat
```

	Number	Genotype	Location	Treatment	Height	Root_length	
1	1	GE_1	Bonn	T	22.27573	18.91135	149
2	2	GE_1	Bonn	N	53.35362	20.92301	151
3	3	GE_1	Cologne	T	45.08251	14.31079	145
4	4	GE_1	Cologne	N	52.47572	19.16425	152
5	5	GE_2	Bonn	T	21.08301	21.76464	150
6	6	GE_2	Bonn	N	47.48507	15.34137	147
7	7	GE_2	Cologne	T	46.34643	15.99019	147
8	8	GE_2	Cologne	N	56.21228	25.88479	153
9	9	GE_3	Bonn	T	22.24373	10.96346	142
10	10	GE_3	Bonn	N	45.69879	13.22416	143

Functions to inspect the data table

Row names given by R

```
> row.names(dat)
```

```
[1] "1"  "2"  "3"  "4"  "5"  "6"  "7"  "8"  "9"  "10"
```

Functions to inspect the data table

Row names given by R

```
> row.names(dat)
```

```
[1] "1"  "2"  "3"  "4"  "5"  "6"  "7"  "8"  "9"  "10"
```

Use `row.names = 1` for user defined row names.

```
> dat=read.table("course_data.txt",header=T,row.names=1)
```

```
> row.names(dat)
```

```
[1] "Line1"    "Line2"    "Line3"    "Line4"    "Line5"  
[6] "Line6"    "Line7"    "Line8"    "Line9"    "Line10"  
[11] "Line11"   "Line12"   "Line13"   "Line14"   "Line15"  
[16] "Line16"   "Line17"   "Line18"   "Line19"   "Line20"  
[21] "Line21"   "Line22"   "Line23"   "Line24"   "Line25"  
[26] "Line26"   "Line27"   "Line28"   "Line29"   "Line30"  
[31] "Line31"   "Line32"   "Line33"   "Line34"   "Line35"  
[36] "Line36"   "Line37"   "Line38"   "Line39"   "Line40"  
[41] "Line41"   "Line42"   "Line43"   "Line44"   "Line45"
```

Functions to inspect the data table

colnames() function returns the column names.

```
> colnames(dat)  
[1] "Genotype"      "Location"       "Height"        "Root_length"  
[5] "Yield"
```

Functions to inspect the data table

colnames() function returns the column names.

```
> colnames(dat)  
[1] "Genotype"      "Location"       "Height"        "Root_length"  
[5] "Yield"
```

names() function print the column names and same as *colnames()*

```
> names(dat)  
[1] "Genotype"      "Location"       "Height"        "Root_length"  
[5] "Yield"
```

Functions to inspect the data table

`str()` function provide information about the data types of each column

```
> str(dat)

'data.frame':      100 obs. of  5 variables:
 $ Genotype    : Factor w/ 50 levels "GE_1","GE_10",...
 $ Location     : Factor w/ 2 levels "Bonn","Cologne": 1 2 1 2 ...
 $ Height       : num  52.3 53.4 45.1 52.5 51.1 ...
 $ Root_length: num  18.9 20.9 14.3 19.2 21.8 ...
 $ Yield        : num  150 151 145 153 150 ...
```

Functions to inspect the data table

summary() function returns a short summary of the table

```
> summary(dat)
```

Genotype	Location	Height	Root_length
GE_1 : 2	Bonn : 50	Min. :34.06	Min. : 2.34
GE_10 : 2	Cologne:50	1st Qu.:47.12	1st Qu.:16.11
GE_11 : 2		Median :50.37	Median :19.22
GE_12 : 2		Mean :50.24	Mean :19.36
GE_13 : 2		3rd Qu.:53.45	3rd Qu.:22.83
GE_14 : 2		Max. :65.84	Max. :34.31
(Other):88			
Yield			
Min. :135.6			
1st Qu.:147.3			
Median :150.6			
Mean :150.4			
3rd Qu.:153.7			

Functions to inspect the data table

`head()` function will print the first few lines of the table. Here $n = 3$ option set the number of lines to print

```
> head(dat, n=3)
```

	Genotype	Location	Height	Root_length	Yield
Line1	GE_1	Bonn	52.27573	18.91135	149.6148
Line2	GE_1	Cologne	53.35362	20.92301	151.0124
Line3	GE_2	Bonn	45.08251	14.31079	145.3708

Functions to inspect the data table

`head()` function will print the first few lines of the table. Here $n = 3$ option set the number of lines to print

```
> head(dat,n=3)
```

	Genotype	Location	Height	Root_length	Yield
Line1	GE_1	Bonn	52.27573	18.91135	149.6148
Line2	GE_1	Cologne	53.35362	20.92301	151.0124
Line3	GE_2	Bonn	45.08251	14.31079	145.3708

`tail()` function will print the last few lines of the table

```
> tail(dat,n=3)
```

	Genotype	Location	Height	Root_length	Yield
Line98	GE_49	Cologne	49.04374	18.31187	149.5071
Line99	GE_50	Bonn	43.16764	13.84799	143.1218
Line100	GE_50	Cologne	51.47421	20.57106	150.9898

Functions to inspect the data table

dim() function returns the no of rows and columns of the table

```
> dim(dat)
```

```
[1] 100    5
```

Functions to inspect the data table

dim() function returns the no of rows and columns of the table

```
> dim(dat)
```

```
[1] 100    5
```

nrow() function provides the number of rows whereas the *ncol()* returns the number of columns

```
> nrow(dat)
```

```
[1] 100
```

```
> ncol(dat)
```

```
[1] 5
```

Functions to inspect the data table

dim() function returns the no of rows and columns of the table

```
> dim(dat)
```

```
[1] 100    5
```

nrow() function provides the number of rows whereas the *ncol()* returns the number of columns

```
> nrow(dat)
```

```
[1] 100
```

```
> ncol(dat)
```

```
[1] 5
```

object.size() will return the memory used by a table in R

```
> object.size(dat)
```

```
13552 bytes
```

Access columns of a table

\$ symbol can be used to access a column of a table.

```
> dat$Location
```

[1]	Bonn	Cologne	Bonn	Cologne	Bonn	Cologne	Bonn	Cologne
[7]	Bonn	Cologne	Bonn	Cologne	Bonn	Cologne	Bonn	Cologne
[13]	Bonn	Cologne	Bonn	Cologne	Bonn	Cologne	Bonn	Cologne
[19]	Bonn	Cologne	Bonn	Cologne	Bonn	Cologne	Bonn	Cologne
[25]	Bonn	Cologne	Bonn	Cologne	Bonn	Cologne	Bonn	Cologne
[31]	Bonn	Cologne	Bonn	Cologne	Bonn	Cologne	Bonn	Cologne
[37]	Bonn	Cologne	Bonn	Cologne	Bonn	Cologne	Bonn	Cologne
[43]	Bonn	Cologne	Bonn	Cologne	Bonn	Cologne	Bonn	Cologne
[49]	Bonn	Cologne	Bonn	Cologne	Bonn	Cologne	Bonn	Cologne
[55]	Bonn	Cologne	Bonn	Cologne	Bonn	Cologne	Bonn	Cologne
[61]	Bonn	Cologne	Bonn	Cologne	Bonn	Cologne	Bonn	Cologne
[67]	Bonn	Cologne	Bonn	Cologne	Bonn	Cologne	Bonn	Cologne
[73]	Bonn	Cologne	Bonn	Cologne	Bonn	Cologne	Bonn	Cologne
[79]	Bonn	Cologne	Bonn	Cologne	Bonn	Cologne	Bonn	Cologne

Access columns of a table

\$ symbol can be used to access a column of a table.

```
> dat$Location
```

[1]	Bonn	Cologne	Bonn	Cologne	Bonn	Cologne	Bonn	Cologne
[7]	Bonn	Cologne	Bonn	Cologne	Bonn	Cologne	Bonn	Cologne
[13]	Bonn	Cologne	Bonn	Cologne	Bonn	Cologne	Bonn	Cologne
[19]	Bonn	Cologne	Bonn	Cologne	Bonn	Cologne	Bonn	Cologne
[25]	Bonn	Cologne	Bonn	Cologne	Bonn	Cologne	Bonn	Cologne
[31]	Bonn	Cologne	Bonn	Cologne	Bonn	Cologne	Bonn	Cologne
[37]	Bonn	Cologne	Bonn	Cologne	Bonn	Cologne	Bonn	Cologne
[43]	Bonn	Cologne	Bonn	Cologne	Bonn	Cologne	Bonn	Cologne
[49]	Bonn	Cologne	Bonn	Cologne	Bonn	Cologne	Bonn	Cologne
[55]	Bonn	Cologne	Bonn	Cologne	Bonn	Cologne	Bonn	Cologne
[61]	Bonn	Cologne	Bonn	Cologne	Bonn	Cologne	Bonn	Cologne
[67]	Bonn	Cologne	Bonn	Cologne	Bonn	Cologne	Bonn	Cologne
[73]	Bonn	Cologne	Bonn	Cologne	Bonn	Cologne	Bonn	Cologne
[79]	Bonn	Cologne	Bonn	Cologne	Bonn	Cologne	Bonn	Cologne

Access columns of a table

\$ symbol can be used to access a column of a table.

```
> dat$Location
```

[1]	Bonn	Cologne	Bonn	Cologne	Bonn	Cologne	Bonn	Cologne
[7]	Bonn	Cologne	Bonn	Cologne	Bonn	Cologne	Bonn	Cologne
[13]	Bonn	Cologne	Bonn	Cologne	Bonn	Cologne	Bonn	Cologne
[19]	Bonn	Cologne	Bonn	Cologne	Bonn	Cologne	Bonn	Cologne
[25]	Bonn	Cologne	Bonn	Cologne	Bonn	Cologne	Bonn	Cologne
[31]	Bonn	Cologne	Bonn	Cologne	Bonn	Cologne	Bonn	Cologne
[37]	Bonn	Cologne	Bonn	Cologne	Bonn	Cologne	Bonn	Cologne
[43]	Bonn	Cologne	Bonn	Cologne	Bonn	Cologne	Bonn	Cologne
[49]	Bonn	Cologne	Bonn	Cologne	Bonn	Cologne	Bonn	Cologne
[55]	Bonn	Cologne	Bonn	Cologne	Bonn	Cologne	Bonn	Cologne
[61]	Bonn	Cologne	Bonn	Cologne	Bonn	Cologne	Bonn	Cologne
[67]	Bonn	Cologne	Bonn	Cologne	Bonn	Cologne	Bonn	Cologne
[73]	Bonn	Cologne	Bonn	Cologne	Bonn	Cologne	Bonn	Cologne
[79]	Bonn	Cologne	Bonn	Cologne	Bonn	Cologne	Bonn	Cologne

Sub setting a table

`subset()` symbol can be used to sub set a table based on some criteria

```
> dat_150=subset(dat, dat$Yield<150)
```

```
> head(dat_150,3)
```

	Genotype	Location	Height	Root_length	Yield
Line1	GE_1	Bonn	52.27573	18.91135	149.6148
Line3	GE_2	Bonn	45.08251	14.31079	145.3708
Line6	GE_3	Cologne	47.48507	15.34137	147.4639

Sub setting a table

`subset()` symbol can be used to sub set a table based on some criteria

```
> dat_150=subset(dat, dat$Yield<150)
```

```
> head(dat_150,3)
```

	Genotype	Location	Height	Root_length	Yield
Line1	GE_1	Bonn	52.27573	18.91135	149.6148
Line3	GE_2	Bonn	45.08251	14.31079	145.3708
Line6	GE_3	Cologne	47.48507	15.34137	147.4639

`%in%` operator can be used to filter the data

```
> bonn=dat[dat$Location %in% "Bonn",]# A new table for Bonn
```

```
> head(bonn,3)
```

	Genotype	Location	Height	Root_length	Yield
Line1	GE_1	Bonn	52.27573	18.91135	149.6148
Line3	GE_2	Bonn	45.08251	14.31079	145.3708
Line5	GE_3	Bonn	51.08301	21.76464	150.1298

Adding a new column

\$ symbol can be used add a new column to the table

```
> dat$Diff=dat$Height-dat$Root_length  
> head(dat,3)
```

	Genotype	Location	Height	Root_length	Yield
Line1	GE_1	Bonn	52.27573	18.91135	149.6148
Line2	GE_1	Cologne	53.35362	20.92301	151.0124
Line3	GE_2	Bonn	45.08251	14.31079	145.3708
		Diff			
Line1		33.36438			
Line2		32.43061			
Line3		30.77172			

Deleting a column

\$ operator with *NULL* can be used to delete a column

```
> dat$Diff=NULL  
> head(bonn,3)
```

	Genotype	Location	Height	Root_length	Yield
Line1	GE_1	Bonn	52.27573	18.91135	149.6148
Line3	GE_2	Bonn	45.08251	14.31079	145.3708
Line5	GE_3	Bonn	51.08301	21.76464	150.1298

Deleting a row

[] operator can be used to delete rows

```
> dat_new=dat[-c(1:2),]  
> head(dat_new,3)
```

	Genotype	Location	Height	Root_length	Yield
Line3	GE_2	Bonn	45.08251	14.31079	145.3708
Line4	GE_2	Cologne	52.47572	19.16425	152.5497
Line5	GE_3	Bonn	51.08301	21.76464	150.1298

Saving the table

`write.table()` function can be used to save a table into hard disk

```
> write.table(dat, "test_one.txt")
```

R will a create new file called `test.txt` and save the table `dat`.

Saving the table

`write.table()` function can be used to save a table into hard disk

```
> write.table(dat, "test_one.txt")
```

R will a create new file called `test.txt` and save the table `dat`.

In a more ordered way

```
> write.table(dat, "test_one.txt", quote=F, row.name=F)
```

Quotes and the row names will be omitted from the out put file.

"Introduction to R graphics!!!"

Plotting system in R

- R provides functions for both simple and high-level plotting functions.
- *plot()* function is the one of the most commonly used function for plotting simple graphics
- R supports different file formats like pdf, jpeg, png, tif, postscript.

Read an example dataset from Excel table

`read.xls()` function can be used to read a Excel table

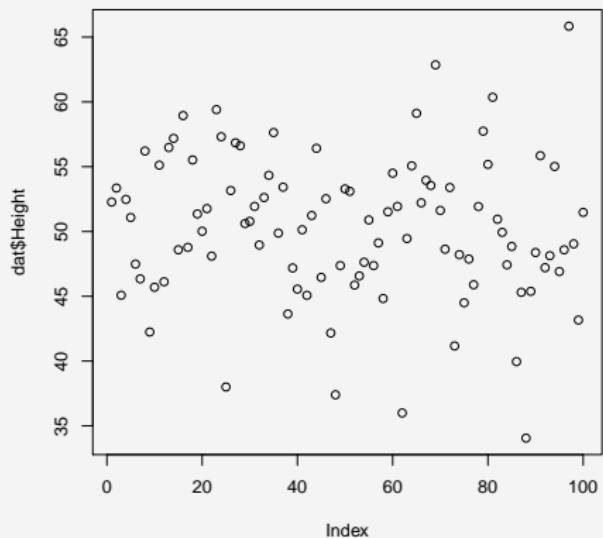
```
> dat=read.xls("course_data.xlsx",sheet = 1, header = TRUE)  
> head(dat,3)
```

	Number	Genotype	Location	Height	Root_length	Yield
1	1	GE_1	Bonn	52.27573	18.91135	149.6148
2	2	GE_1	Cologne	53.35362	20.92301	151.0124
3	3	GE_2	Bonn	45.08251	14.31079	145.3708

Scatter plot

Scatter plot with points.

```
> plot(dat$Height)
```



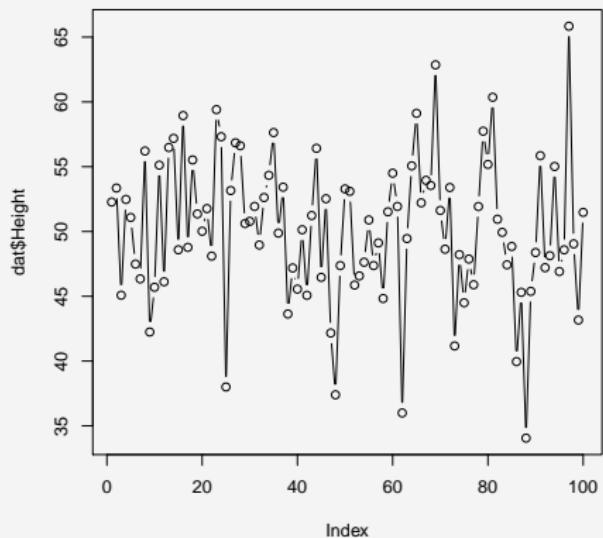
Different options to plot the graph

- "l" for lines
- "p" for points
- "b" for both points and lines
- "o" for both over plotted image
- "h" for histogram like vertical lines

Scatter plot

Scatter plot with points and line.

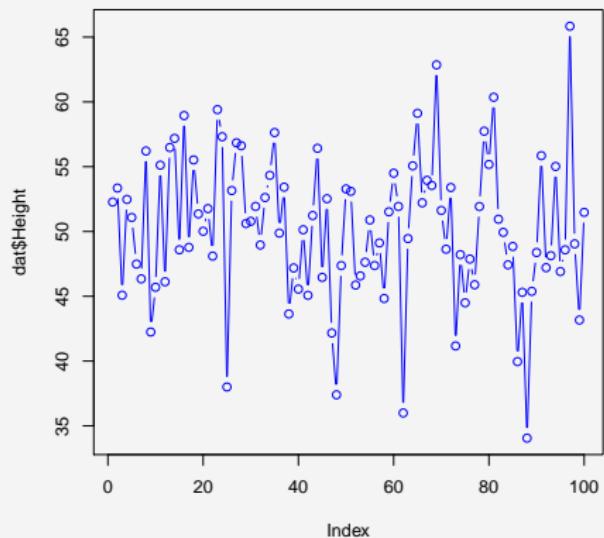
```
> plot(dat$Height, type="b")
```



Changing color

`col="option"` can be used to change the color of the graph

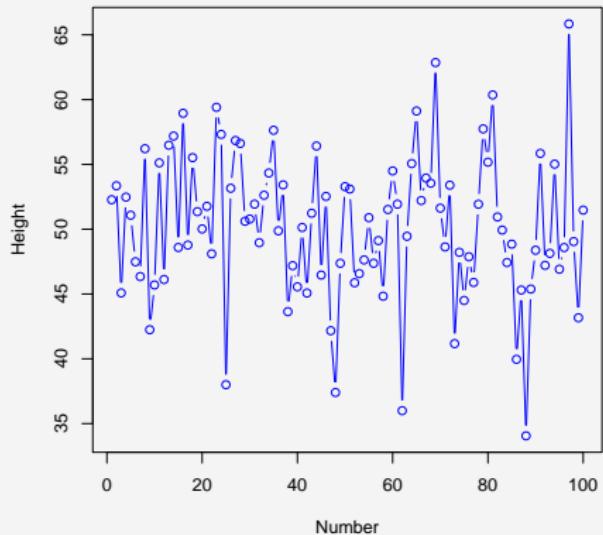
```
> plot(dat$Height, type="b", col="blue")
```



Naming the X and Y axis.

'*xlab*' and '*ylab*' option can be used to label the X and Y axis.

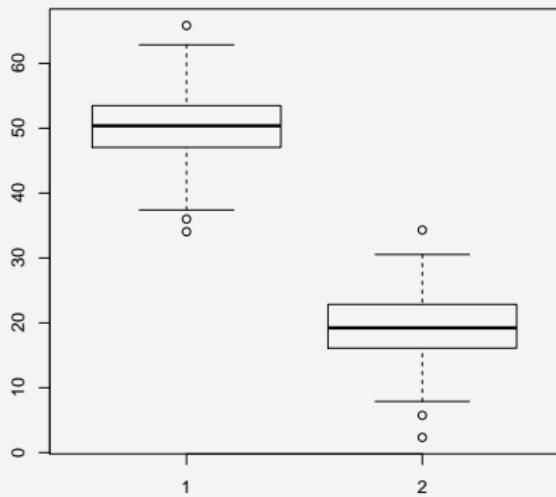
```
> plot(dat$Height, type="b", col="blue",
+       xlab="Number", ylab="Height")
```



Box plots in R

`boxplot()` function is used to draw box plots in R.

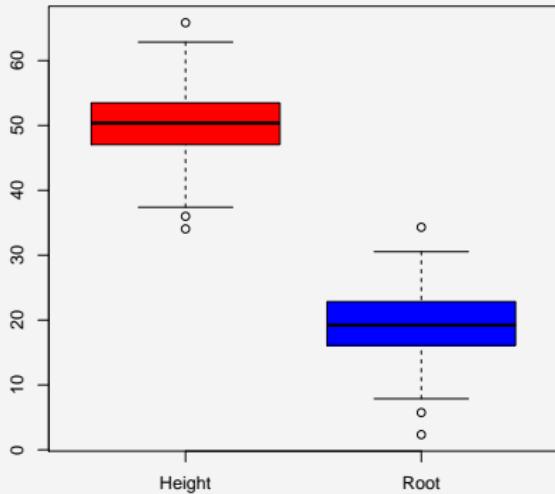
```
> boxplot(dat$Height,dat$Root_length)
```



Box plots in R

col option is used to define the color for the plot.

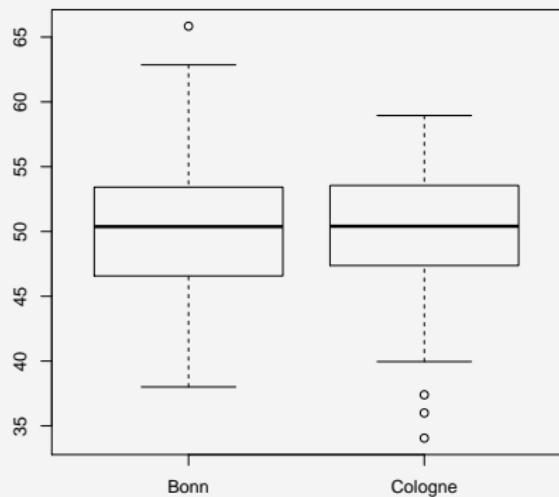
```
> boxplot(dat$Height,dat$Root_length,col=c("red","blue"),  
+ names=c("Height","Root"))
```



Box plots in R

Plot location specific box plot

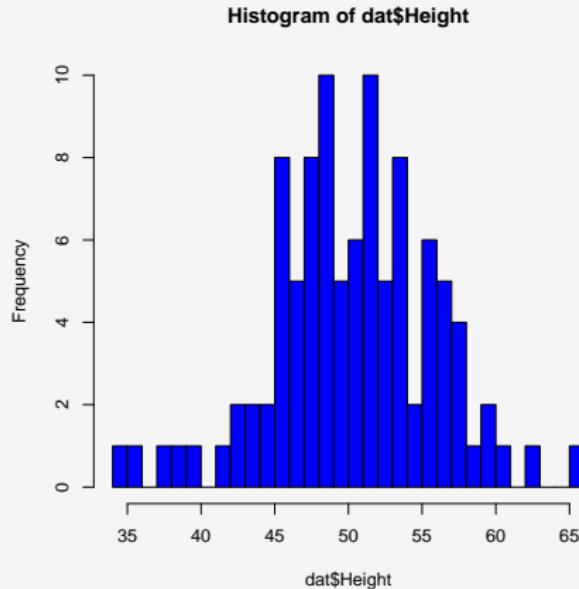
```
> boxplot(dat$Height ~ dat$Location)
```



Histogram in R

Histogram is an easy way to show the distribution of the data

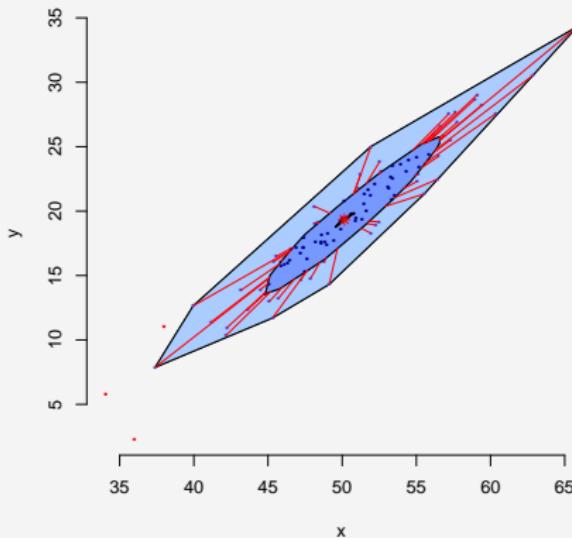
```
> hist(dat$Height, breaks=30, col="blue")
```



Bagplot

Bag plot is useful to show the spread and outliers in the data.

```
> library(aplpack)  
> bagplot(dat$Height,dat$Root_length)
```



Saving plots in R

R graphics can be saved in different formats like 'pdf', 'tif', 'jpg', 'png'

- First create the file to save the plot: `pdf('example.pdf')`

Saving plots in R

R graphics can be saved in different formats like 'pdf', 'tif', 'jpg', 'png'

- First create the file to save the plot: `pdf('example.pdf')`
- Then plot the figure: `plot(dat$Height)`

Saving plots in R

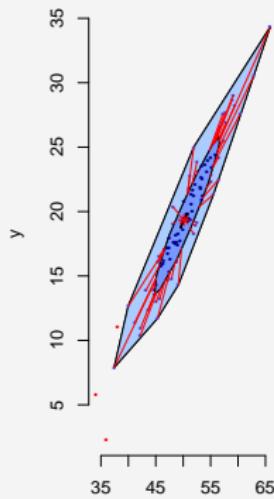
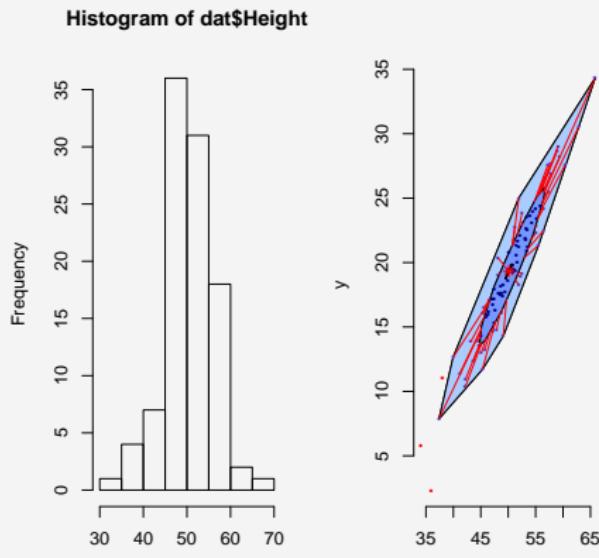
R graphics can be saved in different formats like 'pdf', 'tif', 'jpg', 'png'

- First create the file to save the plot: `pdf('example.pdf')`
- Then plot the figure: `plot(dat$Height)`
- Finally call the `dev.off()` function to close graphic device

Combining plots in R

`par()` function can be used to combine different plots

```
> gp=par(mfrow = c(1,2)) #graph with 1 row and 2 columns  
> hist(dat$Height)  
> bagplot(dat$Height,dat$Root_length)
```



Loops in R

Loop are used execute codes repeatedly

```
> for(i in 1:5){  
+ print("hai")  
+ }
```

```
[1] "hai"  
[1] "hai"  
[1] "hai"  
[1] "hai"  
[1] "hai"
```

Loops in R

Loop to access the elements of an array

```
> num=c("One", "Two", "Three", "Four", "five")
> for(i in 1:5){
+ print(num[i])
+ }
[1] "One"
[1] "Two"
[1] "Three"
[1] "Four"
[1] "five"
```

Functions in R

Functions are blocks of code designed to do specific task

```
> user_square=function(x)
+ {
+ return (x*x)
+ }
```

Here *user_square* is the function name and *x* is the argument

How to call the function

```
> user_square(3)
[1] 9
```

"Introduction to R graphics with ggplot2"

Grammar of graphics plot (ggplot)

- Grammar of graphics plot (ggplot) is based on concepts from The Grammar of Graphics by Leland Wilkinson, 2005

Grammar of graphics plot (ggplot)

- Grammar of graphics plot (ggplot) is based on concepts from The Grammar of Graphics by Leland Wilkinson, 2005
- It is one of the most widely used multivariate data visualization tools in R

Grammar of graphics plot (ggplot)

- Grammar of graphics plot (ggplot) is based on concepts from The Grammar of Graphics by Leland Wilkinson, 2005
- It is one of the most widely used multivariate data visualization tools in R
- It provides two main functions *qplot()* and *ggplot()*.

Grammar of graphics plot (ggplot)

- Grammar of graphics plot (ggplot) is based on concepts from The Grammar of Graphics by Leland Wilkinson, 2005
- It is one of the most widely used multivariate data visualization tools in R
- It provides two main functions *qplot()* and *ggplot()*.
- *qplot()* is similar to *plot()*.

Components of *ggplot*

- data: What you want to visualize like plant height, Yield...

Components of *ggplot*

- **data:** What you want to visualize like plant height, Yield...
- **geoms:** Defines which type of plot. eg, Scatter plot, histogram, boxplot..

Components of *ggplot*

- **data:** What you want to visualize like plant height, Yield...
- **geoms:** Defines which type of plot. eg, Scatter plot, histogram, boxplot..
- **aesthetics:** Defines the color, shape or size of the points or bins on the plot.

Components of *ggplot*

- **data:** What you want to visualize like plant height, Yield...
- **geoms:** Defines which type of plot. eg, Scatter plot, histogram, boxplot..
- **aesthetics:** Defines the color, shape or size of the points or bins on the plot.
- **scale:** Defines if you wan to scale the data. e.e, log scale.

Components of *ggplot*

- **data:** What you want to visualize like plant height, Yield...
- **geoms:** Defines which type of plot. eg, Scatter plot, histogram, boxplot..
- **aesthetics:** Defines the color, shape or size of the points or bins on the plot.
- **scale:** Defines if you wan to scale the data. e.e, log scale.
- **facet:** Used to break the dataset and plot them separately. Eg. plots based on year or Location..

Components of *ggplot*

- **data:** What you want to visualize like plant height, Yield...
- **geoms:** Defines which type of plot. eg, Scatter plot, histogram, boxplot..
- **aesthetics:** Defines the color, shape or size of the points or bins on the plot.
- **scale:** Defines if you wan to scale the data. e.e, log scale.
- **facet:** Used to break the dataset and plot them separately. Eg. plots based on year or Location..
- **stat:** Used to summarize the data.

Import data into R

Read the data into R using `read.table()` function

```
> dat=read.table("real_data.txt",header=T)  
> head(dat,8)
```

	Line	Treat	Place	Year	Hea	Height	Yield
1	Sca001	T	Diko	2003	72	100	41.63330
2	Sca001	N	Diko	2003	72	95	37.50000
3	Sca001	T	Gudow	2003	71	84	43.00000
4	Sca001	N	Gudow	2003	71	90	34.00000
5	Sca001	T	Irlbach	2003	62	55	28.20513
6	Sca001	N	Irlbach	2003	63	50	33.05861
7	Sca001	T	Morgenrot	2003	64	100	33.78400
8	Sca001	N	Morgenrot	2003	65	108	27.35000

Check the data for factors

```
> str(dat)

'data.frame':      1006 obs. of  7 variables:
 $ Line   : Factor w/ 63 levels "Sca001","Sca002",...: 1 1 1 1 ...
 $ Treat  : Factor w/ 2 levels "N","T": 2 1 2 1 2 1 2 1 2 1 ...
 $ Place  : Factor w/ 4 levels "Diko","Gudow",...: 1 1 2 2 3 3 4 ...
 $ Year   : int  2003 2003 2003 2003 2003 2003 2003 2003 2004 ...
 $ Hea    : int  72 72 71 71 62 63 64 65 73 74 ...
 $ Height: num  100 95 84 90 55 ...
 $ Yield  : num  41.6 37.5 43 34 28.2 ...
```

Continuous and categorical variables

- Continuous variables can take many possible values. eg the plant height, population in a town..

Continuous and categorical variables

- Continuous variables can take many possible values. eg the plant height, population in a town..
- Continuous variables shows a continuous variation in the measurements.

Continuous and categorical variables

- Continuous variables can take many possible values. eg the plant height, population in a town..
- Continuous variables shows a continuous variation in the measurements.
- Categorical variable defines a group to which the measurements belongs

Continuous and categorical variables

- Continuous variables can take many possible values. eg the plant height, population in a town..
- Continuous variables shows a continuous variation in the measurements.
- Categorical variable defines a group to which the measurements belongs
- Example: Gender, City, infected with a bacteria or virus (0,1)

Defining factor variables in R

- Categorical variables are treated as factors in R.

Defining factor variables in R

- Categorical variables are treated as factors in R.
- By default R convert all fields with string as factors.

Defining factor variables in R

- Categorical variables are treated as factors in R.
- By default R convert all fields with string as factors.
- But numerical categorical variables need to be converted to factors.

Defining factor variables in R

- Categorical variables are treated as factors in R.
- By default R convert all fields with string as factors.
- But numerical categorical variables need to be converted to factors.
- Use *as.factor()* function to define factor variables in the data frame

Defining factor variables in R

Use `as.factor()` function to define the factors

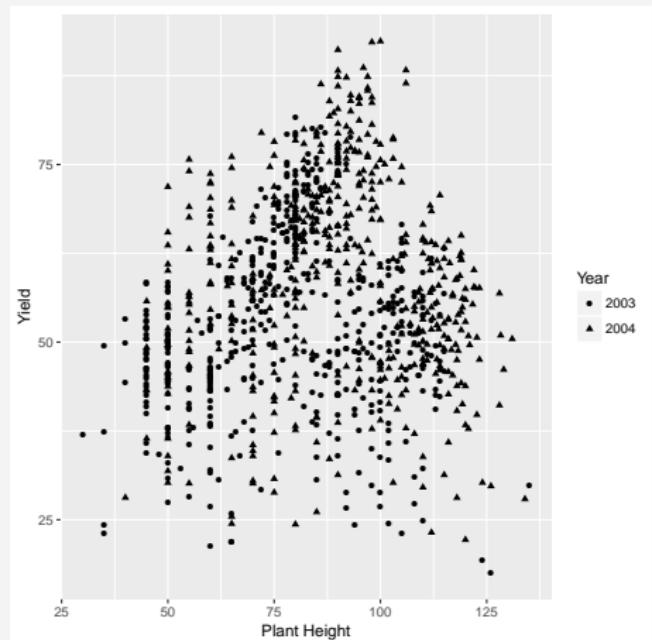
```
> dat$Treat=as.factor(dat$Treat)
> dat$Year=as.factor(dat$Year)
> dat$Place=as.factor(dat$Place)
> str(dat)
```

```
'data.frame': 1006 obs. of 7 variables:
 $ Line   : Factor w/ 63 levels "Sca001","Sca002",...
 $ Treat   : Factor w/ 2 levels "N","T": 2 1 2 1 2 1 2 1 ...
 $ Place   : Factor w/ 4 levels "Diko","Gudow",...
 $ Year    : Factor w/ 2 levels "2003","2004": 1 1 1 1 1 1 1 ...
 $ Hea     : int  72 72 71 71 62 63 64 65 73 74 ...
 $ Height: num  100 95 84 90 55 ...
 $ Yield   : num  41.6 37.5 43 34 28.2 ...
```

Scatter plot with different shapes for the year

qplot example

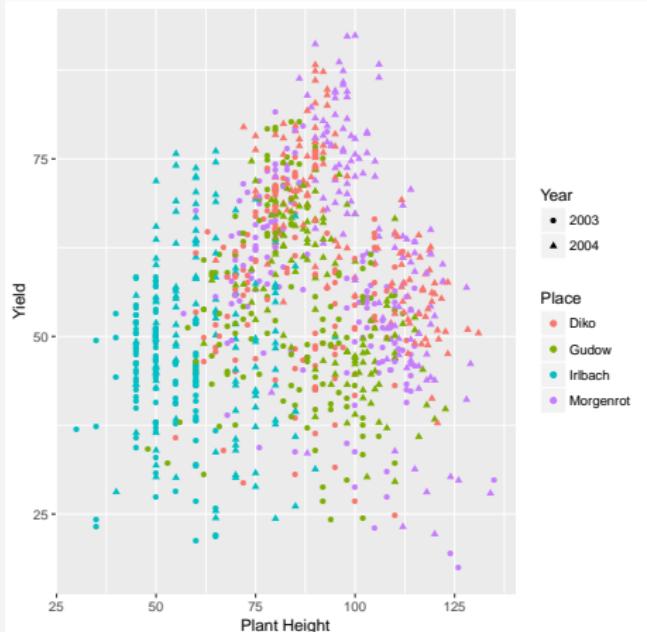
```
> qplot(Height, Yield, data=dat, shape=Year,  
+ xlab="Plant Height", ylab="Yield")
```



Scatter plot with different color for different locations

qplot() example

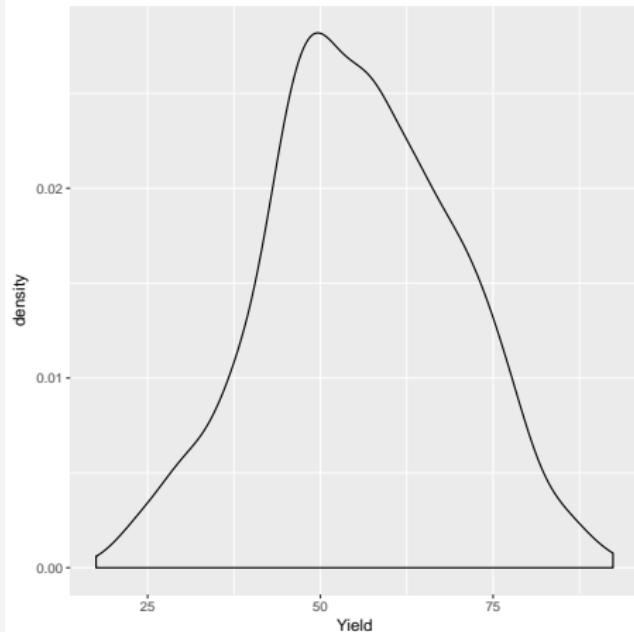
```
> qplot(Height, Yield, data=dat, shape=Year, color=Place,  
+ xlab="Plant Height", ylab="Yield")
```



Density plot

geom example

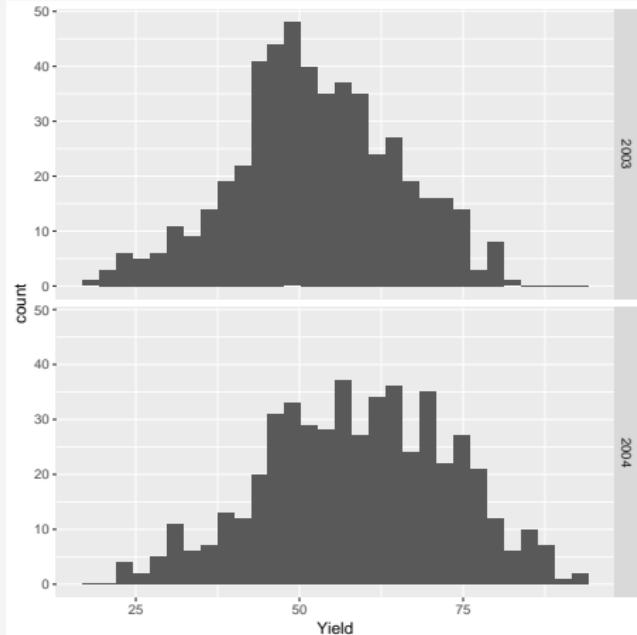
```
> qplot(Yield,data=dat,geom="density")
```



Histogram for each year

facets example

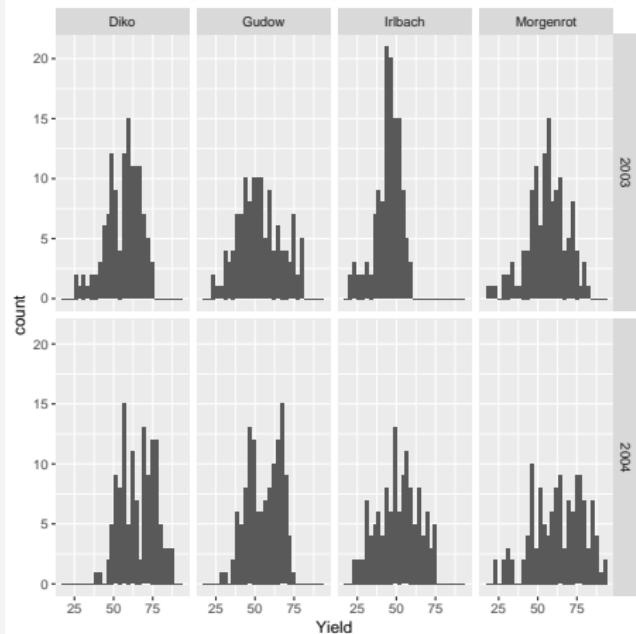
```
> qplot(Yield,data=dat,geom="histogram",facets=Year~.)
```



Histogram for each year and each location

facets example

```
> qplot(Yield, data=dat, geom="histogram", facets=Year~Place)
```



saving the image

`ggsave` to save the image

First create the image

```
> p = qplot(Yield, Hea, data=dat, geom="jitter")
```

Then save to the disk using function `ggsave`.

```
> ggsave("example.png", width=5, height=5)
```

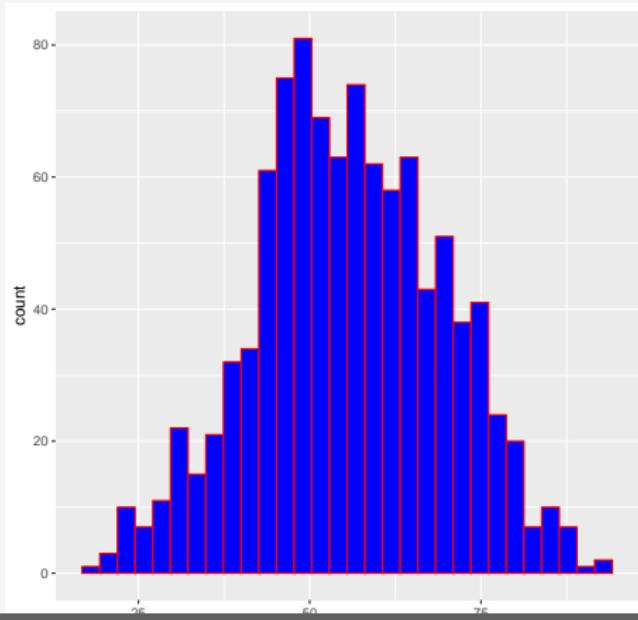
You can save in different format like `jpeg`, `png`, `tif`, `pdf`.

```
> ggsave("example.jpeg", width=5, height=5)
```

Examples of *ggplot*

ggplot gives even more control over your plots

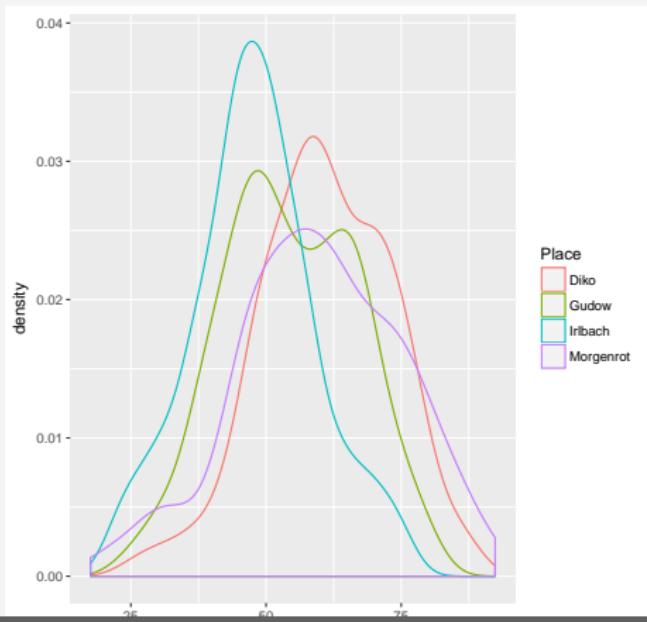
```
> c=ggplot(dat, aes(x=Yield)) # will create a plot object\\
> c=c+geom_histogram(color="red",fill="blue")
> plot(c)
```



Examples of *ggplot*

Density plot with different colors for different regions

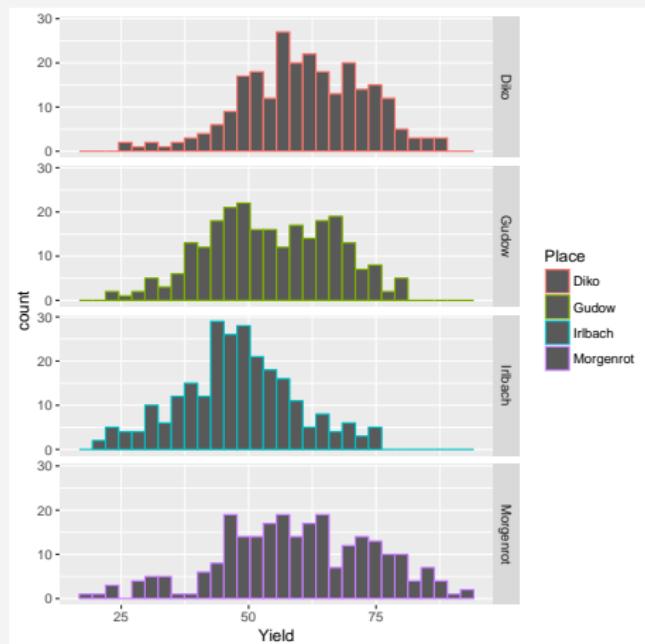
```
> c=ggplot(dat, aes(x=Yield, group=Place,color=Place))  
> c+ geom_density()  
> plot(c)
```



Examples of *ggplot*

Histogram for different location in a single panel

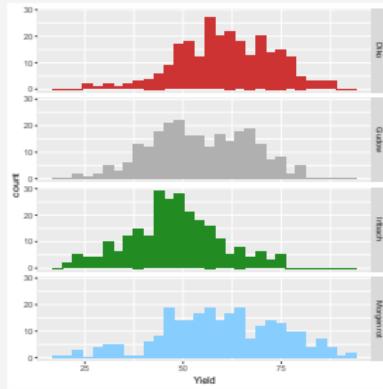
```
> p=ggplot(dat,aes(Yield,color=Place))+geom_histogram()  
> p+facet_grid(Place~.)
```



Examples of *ggplot*

Histogram with user defined colors

```
> p=ggplot(dat,aes(Yield, fill = factor(Place,  
+ levels=unique(Place))))+  
+ scale_fill_manual(name="Place",  
+ values = c("brown3", "grey69", "forestgreen", "skyblue1"))  
> p+theme(legend.position="none")+  
+ geom_histogram()+facet_grid(Place~.)
```



"Some terminologies"

Some terminologies

Data in statistics

In Statistics data is a set of **objects** for which we measure one or more properties (or characteristics)

Object

Object is an individual, like a person, a car or one plant

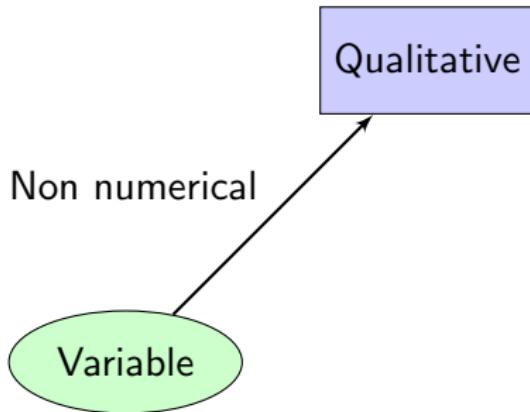
Characteristics

Characteristics are **variables** we measure about one object and they show variation. Eg. person height, weight, plant height or Yield...

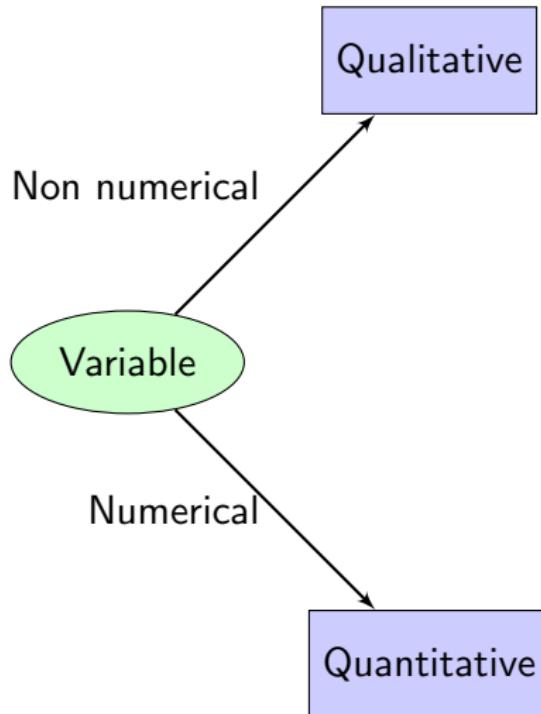
Some terminologies

Variable

Some terminologies



Some terminologies



Some qualitative variables

Gender of a new born baby

Male or Female

Some qualitative variables

Gender of a new born baby

Male or Female

Eye color

Blue, Green, Brown or Black

Some qualitative variables

Gender of a new born baby

Male or Female

Eye color

Blue, Green, Brown or Black

Hair color

Black, Brown, Gray, Red, Yellow)

Some quantitative variables

Temperature in Bonn

21, 21.1, 22.3, 24.2, 25.6...

Some quantitative variables

Temperature in Bonn

21, 21.1, 22.3, 24.2, 25.6...

Rolling a dice

1,2,3,4,5,6

Some quantitative variables

Temperature in Bonn

21, 21.1, 22.3, 24.2, 25.6...

Rolling a dice

1,2,3,4,5,6

Shoe sizes

39, 40, 42..

Some terminologies

Random variable

Random variable is a variable which can take random values (based on chance).

Some terminologies

Random variable

Random variable is a variable which can take random values (based on chance).

Rolling a dice

The outcome could be 1, 2, 3, 4, 5, or 6 with equal chance

Some terminologies

Random variable

Random variable is a variable which can take random values (based on chance).

Rolling a dice

The outcome could be 1, 2, 3, 4, 5, or 6 with equal chance

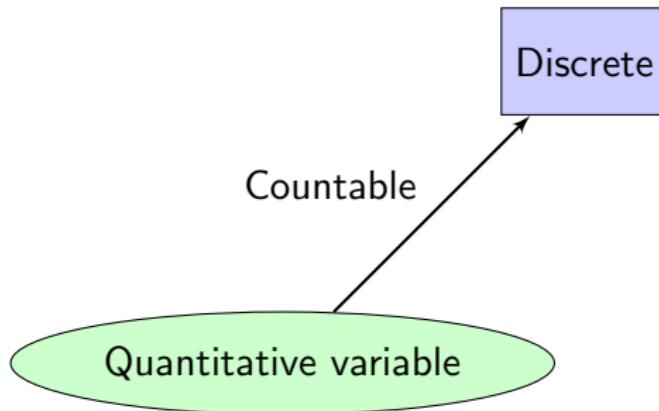
Tossing a coin

Head or Tail with equal chance

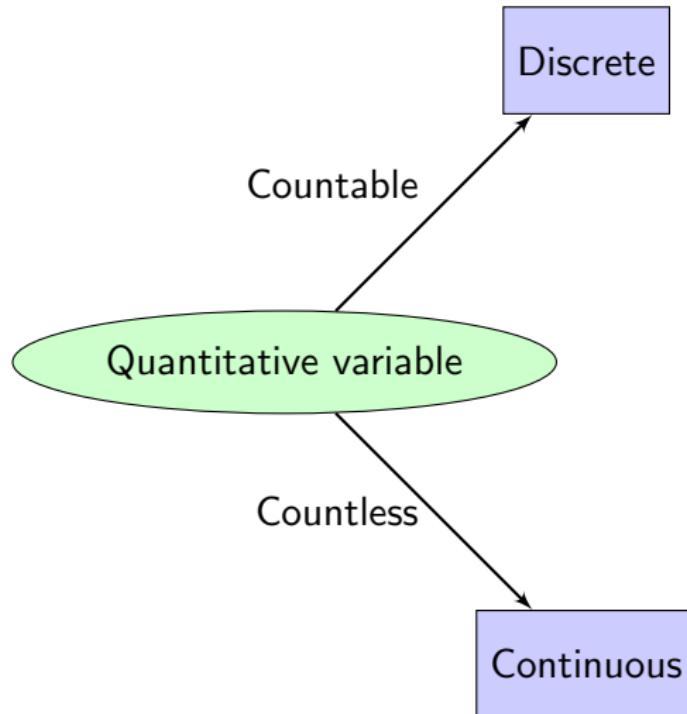
Variables

Quantitative variable

Variables



Variables



Data

The diagram illustrates the concept of data as objects and variables. A table representing a dataset is labeled "Object". Above each column header, there is a label "Variable" with an arrow pointing to the corresponding column.

Name	Age	Height	Weight	Gender
Thomas	27	175.3	70.1	Male
Karin	30	165.4	65.5	Female
John	28	178.1	73.4	Male
Karola	26	168.7	67.7	Female

Some terminologies

What is a population?

Population is the collection of all objects under study. Eg. Height of men and women in Germany or fuel consumption of all Audi cars in Germany

Some terminologies

What is a population?

Population is the collection of all objects under study. Eg. Height of men and women in Germany or fuel consumption of all Audi cars in Germany

What is a sample?

A subset of above population is called sample and we like to make inference about the population based on the sample.

Some terminologies

What is a population?

Population is the collection of all objects under study. Eg. Height of men and women in Germany or fuel consumption of all Audi cars in Germany

What is a sample?

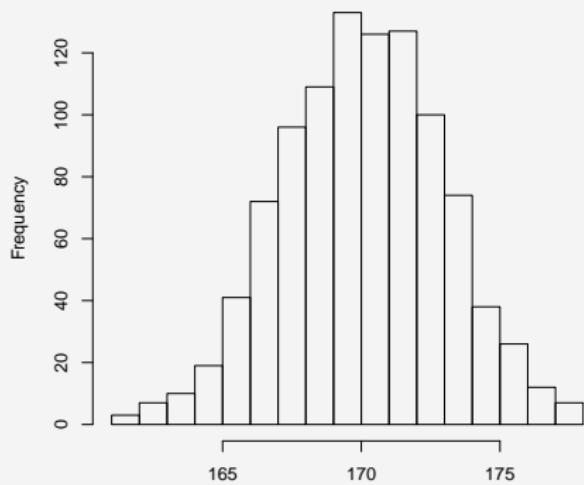
A subset of above population is called sample and we like to make inference about the population based on the sample.

Sampling should be random

Each member of the subset has an equal chance of being selected.

Normal distribution

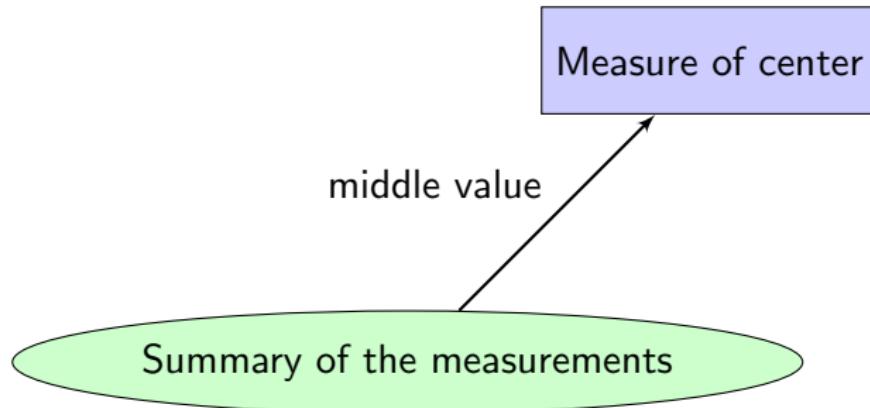
Histogram for normally distributed data with mean(center) of 170 and variance (spread) 9



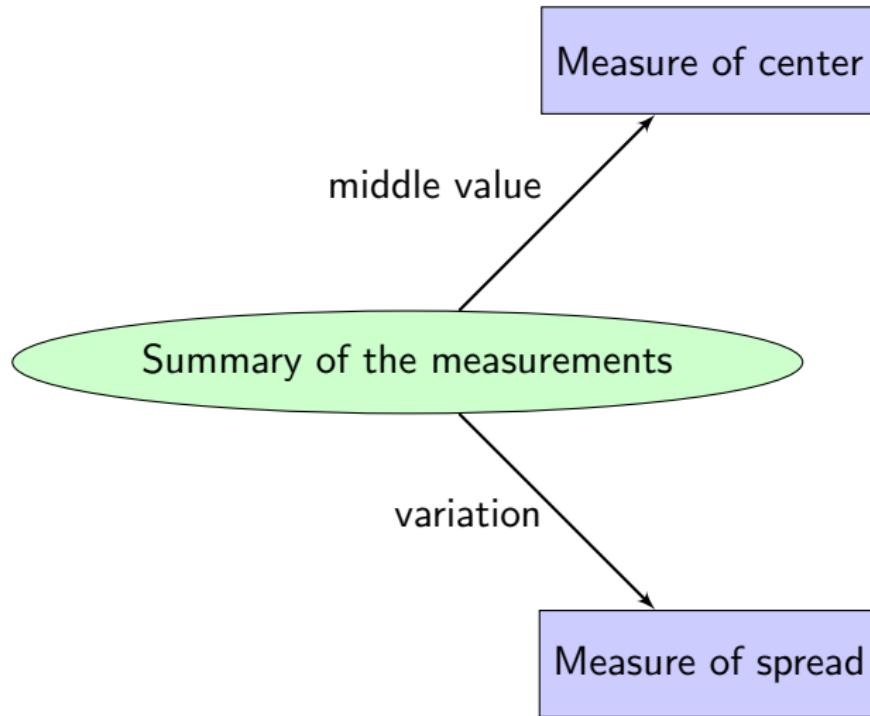
Summary of data

Summary of the measurements

Summary of data



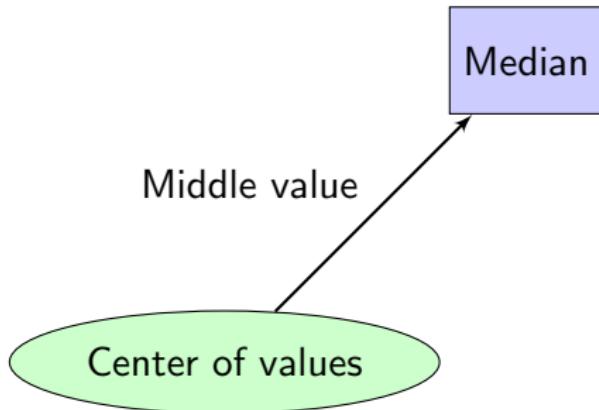
Summary of data



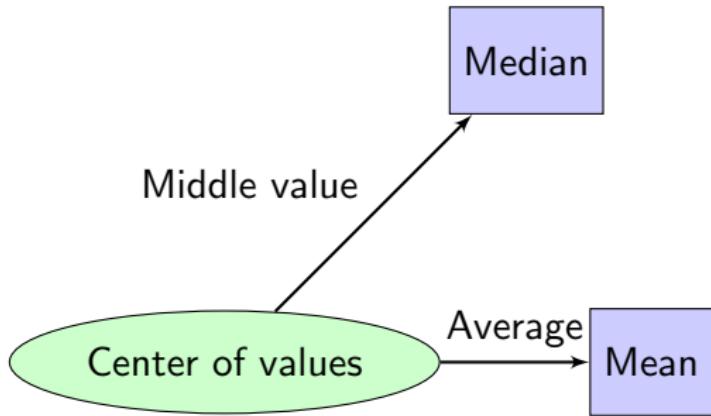
Summary of data

Center of values

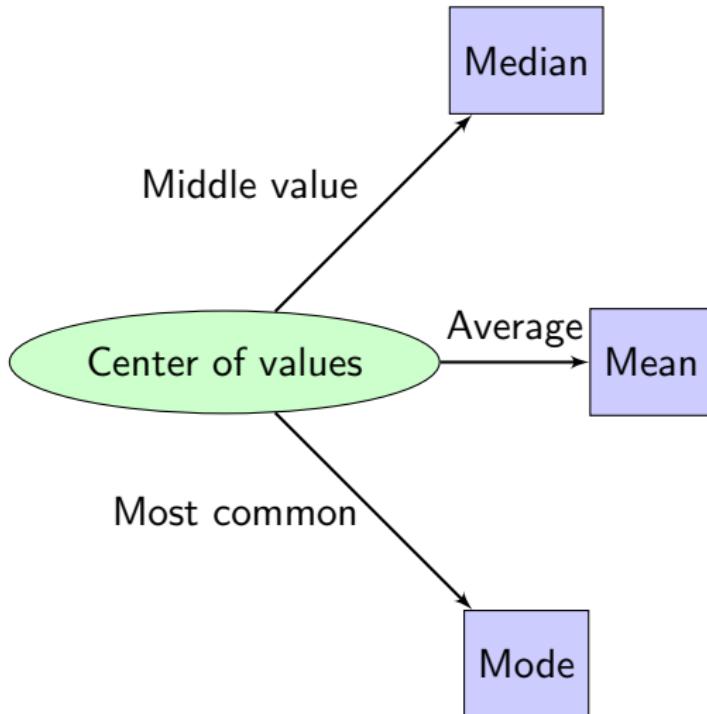
Summary of data



Summary of data



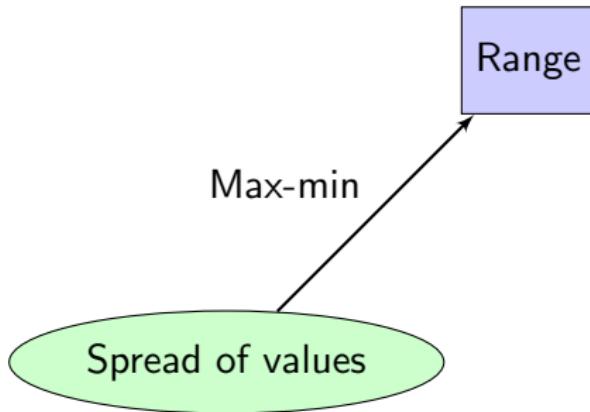
Summary of data



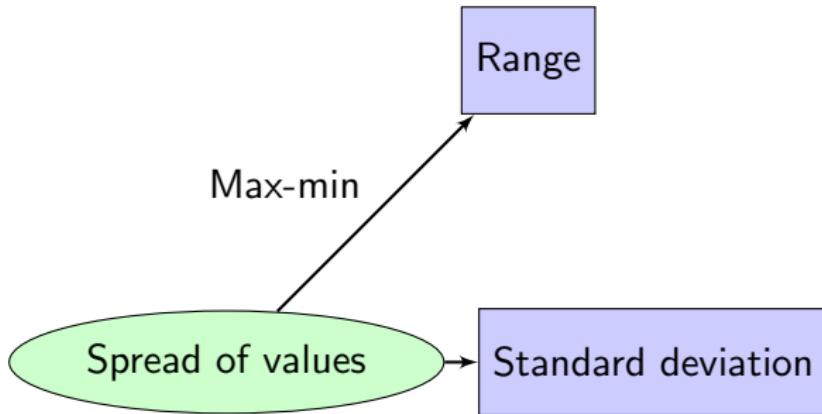
Summary of data

Spread of values

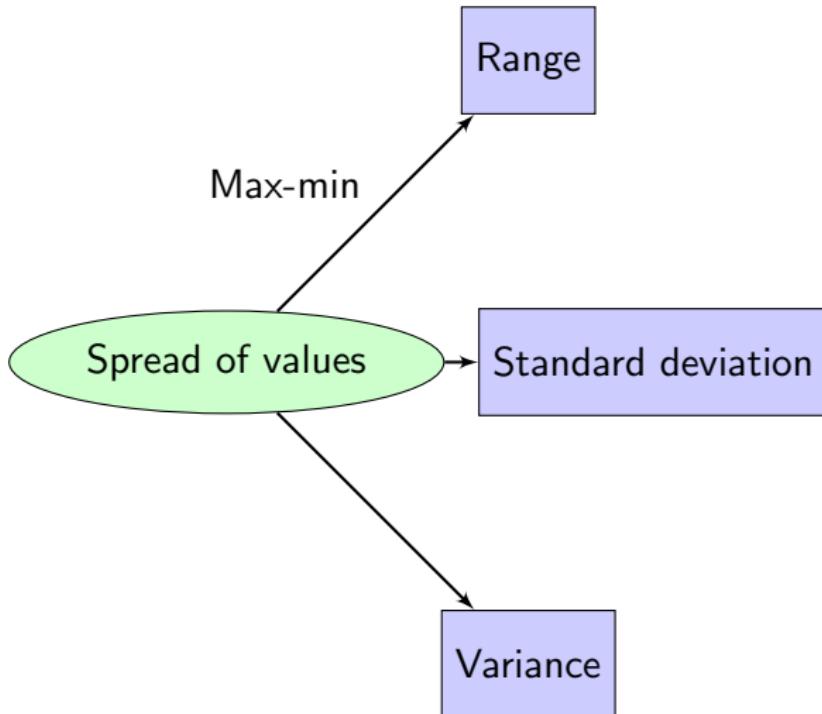
Summary of data



Summary of data



Summary of data



Some terminologies

Statistical test

Statistical test are used to draw conclusion from sample. Commonly used test are:

- T-test: To compare the means of the sample
- ANOVA: Generalization of t-test for multiple comparison
- F-test: Compare the variance of groups

Statistical test

Read the data

```
> dat=read.table("real_data.txt",header=T)
> head(dat,6)
```

	Line	Treat	Place	Year	Hea	Height	Yield
1	Sca001	T	Diko	2003	72	100	41.63330
2	Sca001	N	Diko	2003	72	95	37.50000
3	Sca001	T	Gudow	2003	71	84	43.00000
4	Sca001	N	Gudow	2003	71	90	34.00000
5	Sca001	T	Irlbach	2003	62	55	28.20513
6	Sca001	N	Irlbach	2003	63	50	33.05861

Statistical test

t test

```
> dat_T=dat[dat$Treat %in% 'T',]  
> dat_N=dat[dat$Treat %in% 'N',]  
> t.test(dat_T$Yield,dat_N$Yield,paired=T)
```

Paired t-test

data: dat_T\$Yield and dat_N\$Yield

t = 3.0104, df = 502, p-value = 0.002741

alternative hypothesis: true difference in means is not equal
95 percent confidence interval:

0.4999003 2.3784137

sample estimates:

mean of the differences

1.439157

Statistical test

Read the data

```
> dat_anova=read.table("anova.txt",header=T)
> head(dat,6)
```

	Line	Treat	Place	Year	Hea	Height	Yield
1	Sca001	T	Diko	2003	72	100	41.63330
2	Sca001	N	Diko	2003	72	95	37.50000
3	Sca001	T	Gudow	2003	71	84	43.00000
4	Sca001	N	Gudow	2003	71	90	34.00000
5	Sca001	T	Irlbach	2003	62	55	28.20513
6	Sca001	N	Irlbach	2003	63	50	33.05861

Statistical test

One-way ANOVA

```
> res=aov(Height~Treatment,data=dat_anova)
> summary(res)

      Df Sum Sq Mean Sq F value    Pr(>F)
Treatment     1    2238   2237.9    14.87 0.000206 ***
Residuals   98   14753    150.5
---
Signif. codes:
0
```

Statistical test

Two way ANOVA

```
> res=aov(Height~Treatment+Location,data=dat_anova)
> summary(res)
```

	Df	Sum Sq	Mean Sq	F value	Pr(>F)	
Treatment	1	2238	2237.9	18.16	4.71e-05	***
Location	1	2797	2797.4	22.70	6.65e-06	***
Residuals	97	11955	123.3			

Signif. codes:

0

Some terminologies

ANOVA with interaction

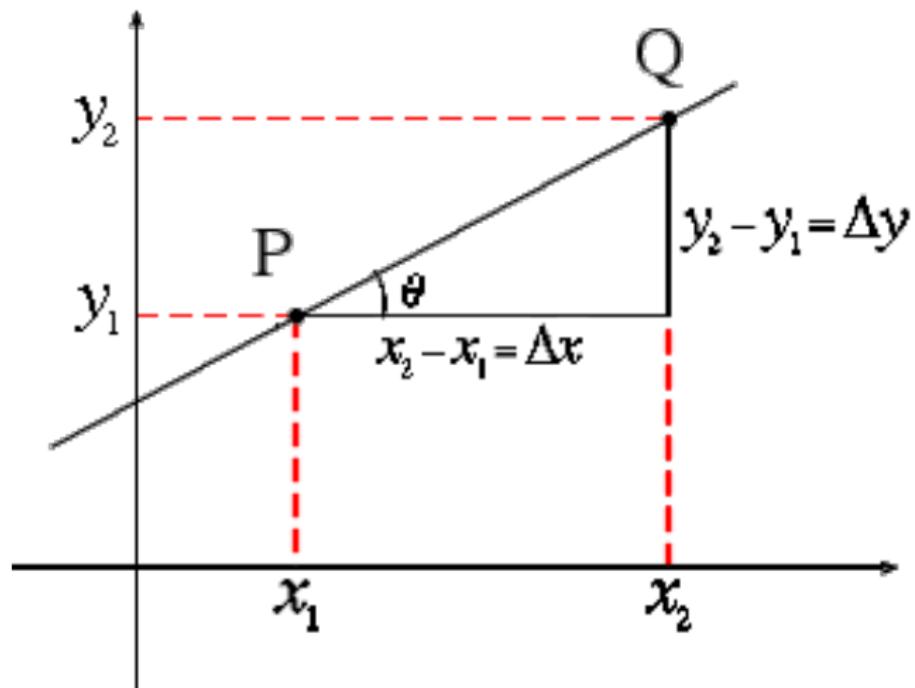
```
> res=aov(Height~Treatment*Location,data=dat_anova)
> summary(res)
```

	Df	Sum Sq	Mean Sq	F value	Pr(>F)	
Treatment	1	2238	2237.9	22.39	7.65e-06	***
Location	1	2797	2797.4	27.99	7.68e-07	***
Treatment:Location	1	2361	2360.6	23.62	4.57e-06	***
Residuals	96	9595	99.9			

Signif. codes:

0

Slope of a line



Modeling variables

Response variable/outcome variable/depended

Response variable are the variables of primary interest. Eg. the total 'Yield' of a plant

Modeling variables

Response variable/outcome variable/depended

Response variable are the variables of primary interest. Eg. the total 'Yield' of a plant

Independent/explanatory variable

One or more independent variables combine to determine the response variable Eg. Root length and Plant Height

Modeling variables

Response variable/outcome variable/depended

Response variable are the variables of primary interest. Eg. the total 'Yield' of a plant

Independent/explanatory variable

One or more independent variables combine to determine the response variable Eg. Root length and Plant Height

Linear relationship

$\text{Yield} \sim \text{Height} + \text{Root length}$

$\text{response variable} \sim \text{explanatory variables(s)}$

Model criticism

Model criticism

- All models are wrong

Model criticism

Model criticism

- All models are wrong
- Some models are better than others

Model criticism

Model criticism

- All models are wrong
- Some models are better than others
- The correct model can never be known with certainty

Model criticism

Model criticism

- All models are wrong
- Some models are better than others
- The correct model can never be known with certainty
- The simpler the model, the better it is

Read the data

Read data

```
> dat=read.table("lm_example.txt",header=T)
> head(dat,5)
```

	Number	Genotype	Location	Treatment	Height	Root_length
1	1	GE_1	Bonn	T	22.27573	18.91135
2	2	GE_1	Bonn	N	53.35362	20.92301
3	3	GE_1	Cologne	T	45.08251	14.31079
4	4	GE_1	Cologne	N	52.47572	19.16425
5	5	GE_2	Bonn	T	21.08301	21.76464

Yield

1	149.6148
2	151.0124
3	145.3708
4	152.5497
5	150.1298

Linear regression

lm() function for the simple linear regression

```
> mod=lm(dat$Yield ~ dat$Height, data=dat)
> summary(mod)
```

Call:

```
lm(formula = dat$Yield ~ dat$Height, data = dat)
```

Residuals:

Min	1Q	Median	3Q	Max
-13.5364	-3.1900	0.1194	2.8269	14.1848

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	144.15166	1.84039	78.33	< 2e-16 ***
dat\$Height	0.13790	0.03917	3.52	0.000656 ***

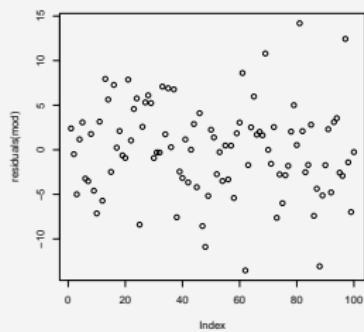
Signif. codes:

Check the residuals

Residuals

Residuals is the difference between the observed values and the predicted values based on the model.

```
> plot(residuals(mod))
```



Coefficients

Coefficient - Estimate

- It contains two rows first is the intercept and the second is the slope
- Intercept corresponds to the adjusted mean.
- Slope measures the change in Y due to the change in X.
- In our example 0.13 increase of Root Length will result in 1 unit increase in Yield

Multiple linear regression

Regression method

$\text{Yield} \sim \text{Height} + \text{Root length} + \text{Height}$

```
> res=lm(Yield ~ Height+Root_length+Location,data=dat)
```

Regression method

'+' sign can be used to add an explanatory variable into the model

'*' is used to include explanatory variables and interactions

Linear regression

Assumptions of Linear regression

- Linearity: The response variable is a linear combination of depended variables.

Linear regression

Assumptions of Linear regression

- Linearity: The response variable is a linear combination of depended variables.
- Independence of errors: The error of the response variables should be uncorrelated

Linear regression

Assumptions of Linear regression

- Linearity: The response variable is a linear combination of depended variables.
- Independence of errors: The error of the response variables should be uncorrelated
- Lack of multicollinearity: The independent variables should be uncorrelated

Linear regression

Assumptions of Linear regression

- Linearity: The response variable is a linear combination of depended variables.
- Independence of errors: The error of the response variables should be uncorrelated
- Lack of multicollinearity: The independent variables should be uncorrelated
- Constant variance: The response variables have the same error variance

Generalized linear regression

GLM

- GLM is appropriate when the response variable is non-normal

Generalized linear regression

GLM

- GLM is appropriate when the response variable is non-normal
- GLM allows different variance structures for the error term

Generalized linear regression

GLM

- GLM is appropriate when the response variable is non-normal
- GLM allows different variance structures for the error term
- The error of the response variables can be uncorrelated

Generalized linear regression (GLM)

Read the data

```
> score=read.table("score.txt",header=T)
> head(score,4)
```

	admit	gre	gpa	rank
1	0	380	3.61	3
2	1	660	3.67	3
3	1	800	4.00	1
4	1	640	3.19	4

Generalized linear regression(GLM)

glm()

```
> res = glm(admit ~ gre + gpa + as.factor(rank), family=binomial, data=score)
> summary(res)
```

Call:

```
glm(formula = admit ~ gre + gpa + as.factor(rank), family = binomial,
     data = score)
```

Deviance Residuals:

Min	1Q	Median	3Q	Max
-1.6268	-0.8662	-0.6388	1.1490	2.0790

Coefficients:

	Estimate	Std. Error	z value	Pr(> z)	
(Intercept)	-3.989979	1.139951	-3.500	0.000465	***
gre	0.002264	0.001094	2.070	0.038465	*
gpa	0.804038	0.331819	2.423	0.015388	*

Mixed effect model

Fixed and random factor

- Fixed effect factor: We have data from all the levels of that factor.
Eg. Male or Female, Treatment effect
- Random effect factor: This factor has many possible levels but we have data from only few levels
- Sometimes we need to consider some factors are random in the model
- Such model we need to used mixed linear model(Mixed means both fixed and random effect)

Mixed effect model

Fixed and random factor

- `lm()` or `glm()` cannot consider both fixed and random effect.
- `lmer()` and `glmer()` are used to model both random and fixed factors in the model.
- `lmer()` is used when the response variable is continuous.
- `glmer()` is for categorical response variable

Mixed effect model

`lmer()`

Syntax for `lmer()`

`lmer(y ~ 1 + (1|first_random_factor) + (1|second_random_factor))`

Interaction term are included with : symbol

`lmer(y ~ 1 + (1|first_random_factor) + (1|second_random_factor : first_random_factor))`

Factors can be nested with '/' symbol

`lmer(y ~ 1 + (1|first_random_factor/second_random_factor))`

Multivariate statistical methods

Commonly used approaches

Multivariate statistical methods are used to find the pattern or structure among the correlated independent variables.

- Principal component analysis
- Cluster analysis
- Factor analysis
- Discriminant analysis

Multivariate statistical methods

Different approaches for PCA

Principal component analysis (PCA) is the one of most commonly used method. In PCA the set of correlated observations are transformed into linearly uncorrelated variables. PCA is done by:

- Eigenvalue decomposition: For covariance matrix
- Singular value decomposition : For data matrix

Multivariate statistical methods

Different approaches for PCA

Principal component analysis (PCA) is the one of most commonly used method. In PCA the set of correlated observations are transformed into linearly uncorrelated variables. PCA is done by:

- Eigenvalue decomposition: For covariance matrix
- Singular value decomposition : For data matrix

Singular value decomposition

$$D = U \Lambda V^*$$

Here the diagonal elements of the Λ contains the singular values

Reading the data

Car data

```
> dat=read.table("car.txt",header=T,row.names=1)  
> head(dat,5)
```

	Cpower	Horsepower	Speed	Weight	Width	Length
Citroen_C2	1024	61	158	922	1659	3666
Smart_Fortwo	698	52	135	730	1515	2500
Mini	1498	160	218	1115	1690	3625
Nissan_Micra	1240	65	154	965	1660	3715
Renault_Clio	2246	205	245	1400	1210	3812

Checking the data

Correlation table

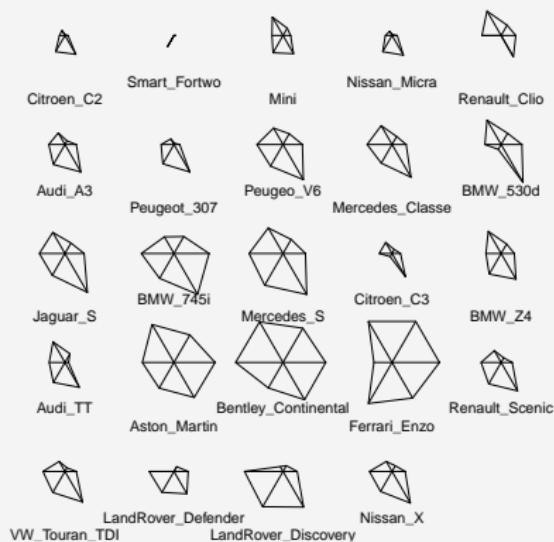
```
> co=cor(dat)  
> round(as.dist(co),3)
```

	Cpower	Horsepower	Speed	Weight	Width
Horsepower	0.943				
Speed	0.810		0.899		
Weight	0.676		0.550	0.474	
Width	0.530		0.518	0.297	0.528
Length	0.679		0.534	0.543	0.777
					0.514

Checking the data

Star plot

```
> stars(dat, nrow=5)
```



Principal component analysis

`prcomp()` function for PCA

```
> pc=prcomp(dat,scale=T)
> ls(pc)
[1] "center"     "rotation"    "scale"        "sdev"         "x"
```

Description

Here `"center"` and `"scale"` are the mean and standard deviations of each column. `"sdev"` is the standard deviation of the eigen values and `"rotation"` is the corresponding eigen vectors. `"x"` is the standardized data matrix multiplied by the `"rotation"`.

Summary of the PCA out put

Variance explained by the component

```
> summary(pc)
```

Importance of components:

	PC1	PC2	PC3	PC4
Standard deviation	2.0334	0.9536	0.75826	0.48510
Proportion of Variance	0.6891	0.1516	0.09583	0.03922
Cumulative Proportion	0.6891	0.8407	0.93648	0.97570
	PC5	PC6		
Standard deviation	0.36506	0.11186		
Proportion of Variance	0.02221	0.00209		
Cumulative Proportion	0.99791	1.00000		

Summary of the PCA out put

The eigen vectors

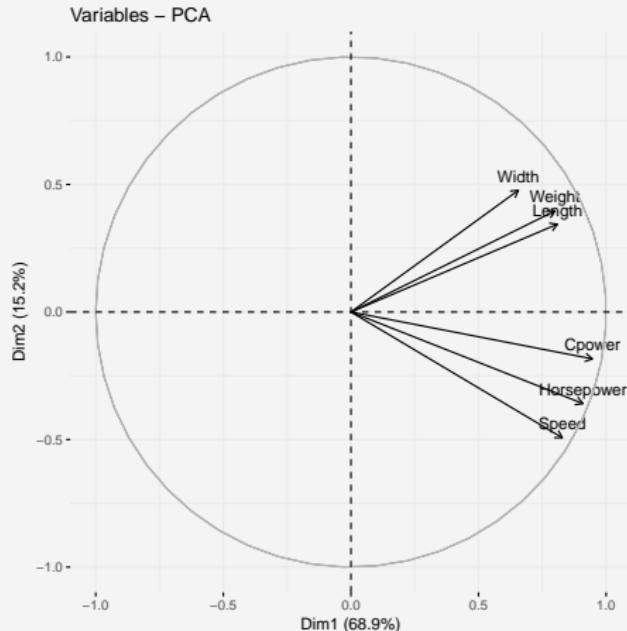
```
> round(pc$rotation,4)
```

	PC1	PC2	PC3	PC4	PC5	PC6
Cpower	0.4654	-0.1928	-0.0416	0.1688	-0.6676	-0.5199
Horsepower	0.4472	-0.3765	-0.2309	0.1416	-0.1178	0.7557
Speed	0.4075	-0.5171	0.0806	-0.2289	0.6299	-0.3330
Weight	0.3934	0.4155	0.3958	0.6596	0.2841	0.0158
Width	0.3228	0.5005	-0.7605	-0.1299	0.1886	-0.1203
Length	0.3980	0.3612	0.4511	-0.6687	-0.1656	0.1818

PCA plots

Plot for the variables

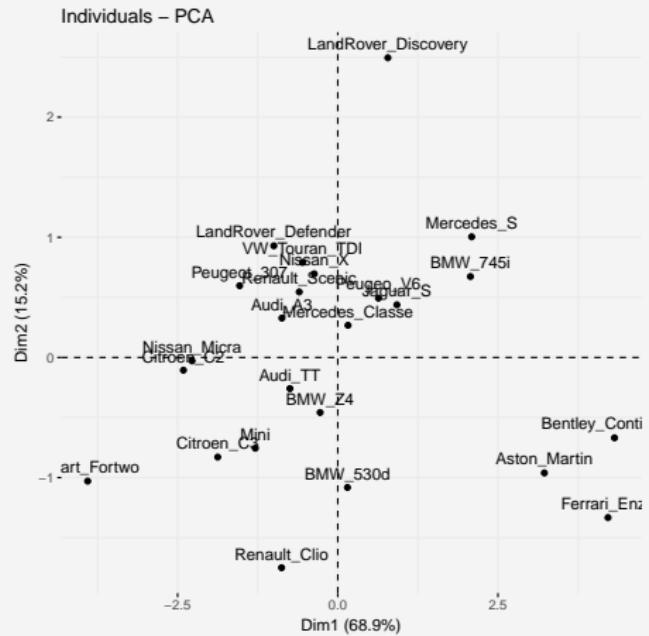
```
> library(factoextra)  
> fviz_pca_var(pc)
```



PCA plots

Plot for the individuals

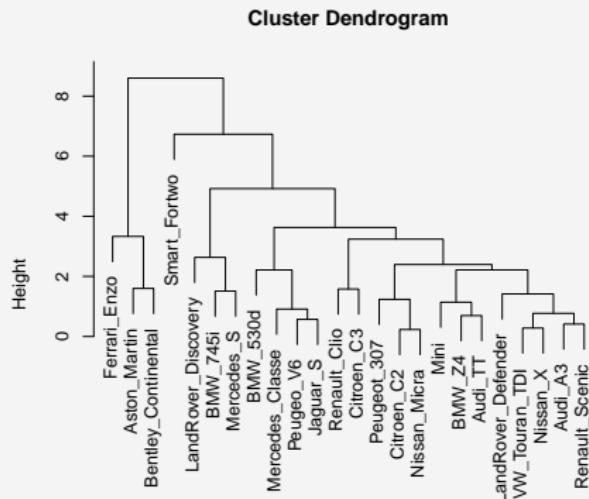
```
> fviz_pca_ind(pc)
```



Hierarchical clustering

Find the clusters using `hclust()`

```
> library("FactoMineR")
> pca=PCA(dat,graph=F)
> clust=hclust(dist(pca$ind$coord))
> plot(clust,xlab="",sub="")
```



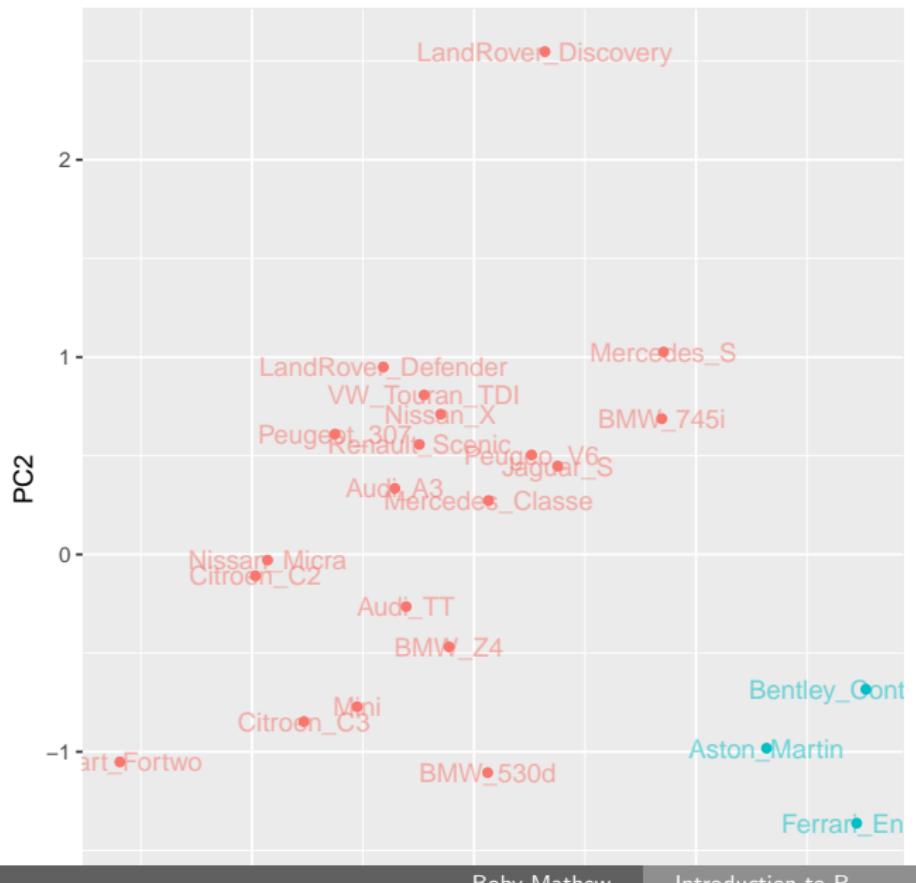
Structuring the data

Create a new data frame with coordinate and group information

```
> obs=data.frame(pca$ind$coord[,1:2])  
> obs$clust=as.factor(cutree(clust,k=2))  
> head(obs,4)
```

	Dim.1	Dim.2	clust
Citroen_C2	-2.461299	-0.10831459	1
Smart_Fortwo	-3.987270	-1.05134100	1
Mini	-1.315167	-0.77151470	1
Nissan_Micra	-2.324196	-0.02872455	1

Plotting the structure



Quantitative Trait Loci (QTL)

QTL????

- QTLs are simply a region within a genome and that affects a trait such as plant height, grain yield..

Quantitative Trait Loci (QTL)

QTL????

- QTLs are simply a region within a genome and that affects a trait such as plant height, grain yield..
- Due to the QTL effect the phenotype show a continuous (like height or weight) variation rather than discrete variation.

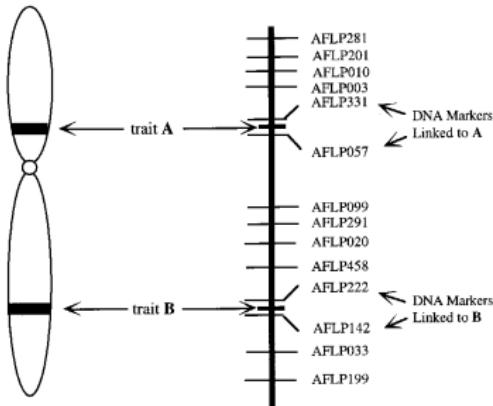
Quantitative Trait Loci (QTL)

QTL????

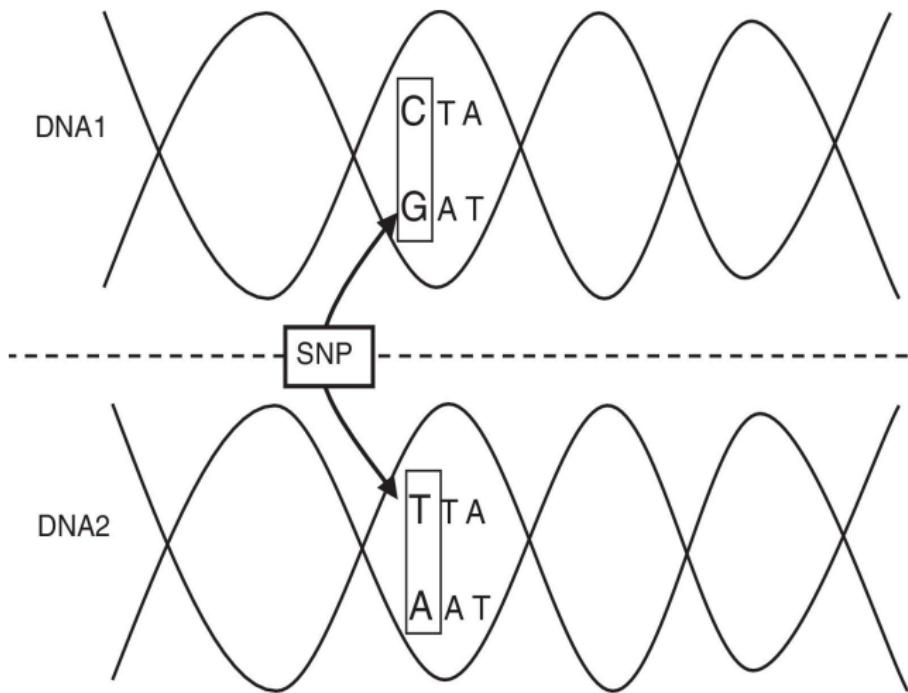
- QTLs are simply a region within a genome and that affects a trait such as plant height, grain yield..
- Due to the QTL effect the phenotype show a continuous (like height or weight) variation rather than discrete variation.
- Quantitative traits are usually polygenes means coded by many genes.

Finding the 'Best' QTL

- QTL identification techniques are simply statistical correlations looking for marker-trait correlations
- But we are looking for many markers that correlate with a single trait
- So QTL detection methods statistically challenging



Single Nucleotide Polymorphism (SNP)



SNP to numerical coding

Marker	Line1	Line2	Line3		Line1	Line2	Line3
AF04309	A/A	A/A	A/T		1	1	-1
BMS2	T/T	T/A	T/A		-1	1	1
BMS64	G/G	G/G	G/C		1	1	-1
Bmac163	C/G	C/C	C/C		-1	1	1
Bmac29	A/T	A/T	A/A		1	1	-1

QTL mapping

Prerequisite of QTL mapping

- Phenotypes, Y_i

QTL mapping

Prerequisite of QTL mapping

- Phenotypes, Y_i
- Molecular marker (SNP) information $X_{ij} = \text{AA/Aa/aa}$.

QTL mapping

Prerequisite of QTL mapping

- Phenotypes, Y_i
- Molecular marker (SNP) information $X_{ij} = \text{AA/Aa/aa}$.
- A good genetic map with the positions of these markers.

QTL mapping

Prerequisite of QTL mapping

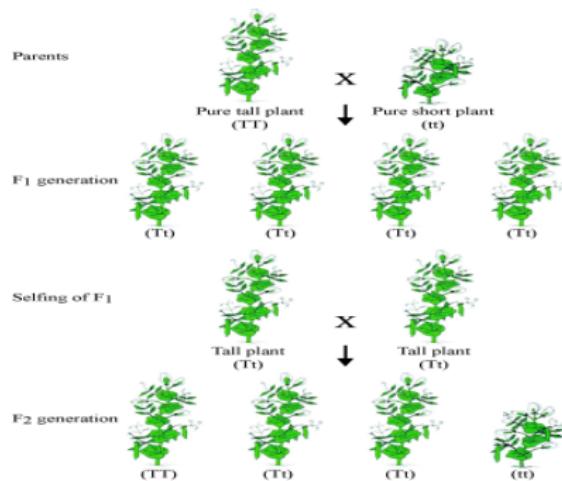
- Phenotypes, Y_i
- Molecular marker (SNP) information $X_{ij} = \text{AA/Aa/aa}$.
- A good genetic map with the positions of these markers.
- Statistical packages for the QTL detection

Different strategies for QTL mapping

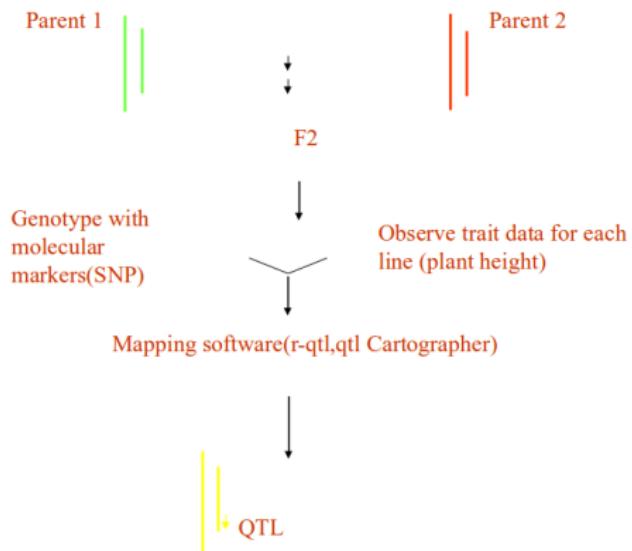
Linkage mapping and Association mapping are mainly used for the QTL mapping.

- Linkage mapping
 - Populations like F2 and the lines are coming from recent crossing.
 - The mapping accuracy is low
 - Need markers in moderate density
- Association mapping
 - Special population and the lines are historically related
 - Can map quantitative traits with high accuracy
 - Need high density markers

Linkage mapping population (F2)



Work flow of QTL detection



Linkage mapping

Single-marker analysis

- Divide the population into subpopulations based on the marker information at a loci

Linkage mapping

Single-marker analysis

- Divide the population into subpopulations based on the marker information at a loci
- Measure the mean of the phenotypic in each of the above subpopulation

Linkage mapping

Single-marker analysis

- Divide the population into subpopulations based on the marker information at a loci
- Measure the mean of the phenotypic in each of the above subpopulation
- Determine the statistical significance of the difference, using $t - test$ or ANOVA

Linkage mapping

Interval mapping

- Look for qtls between the marker positions

Linkage mapping

Interval mapping

- Look for qtls between the marker positions
- Take care about the missing data

Linkage mapping

Interval mapping

- Look for qtls between the marker positions
- Take care about the missing data
- Provide more accurate qtl estimates

QTL mapping tools

Some of the most commonly used QTL mapping packages

- Mapmaker
- QTL Cartographer
- Multimapper
- r/qtl
- MapQTL

Input file for "r/qt"

Height	AF04309	BMS2	BMS64	Bmac163
	5	5	7	5
	156	45	100.3	48.8
72	1	1	1	1
69	1	1	1	1
71	1	1	1	1
77	1	1	1	1
74	1	1	1	1



Importing data

read.cross() function to import data into R

```
> library(qtl)
> dat=read.cross("csv","", "qtl_data.csv",na.string="NA",
+ genotype=c("0","1"))
--Read the following data:
      301 individuals
      364 markers
      1 phenotypes
--Cross type: bc
```

Summary of the imported data

```
> summary(dat)
```

Backcross

No. individuals: 301

No. phenotypes: 1

Percent phenotyped: 100

No. chromosomes: 7

 Autosomes: 1 2 3 4 5 6 7

Total markers: 364

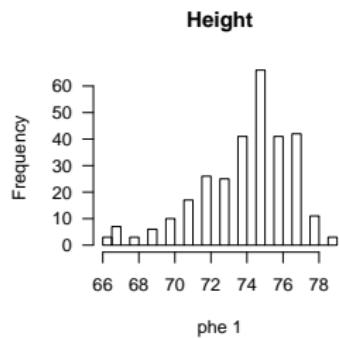
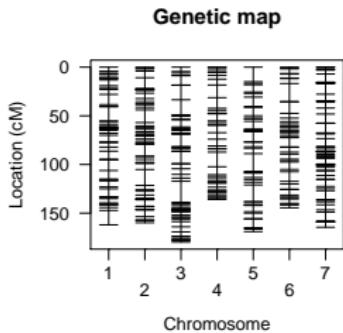
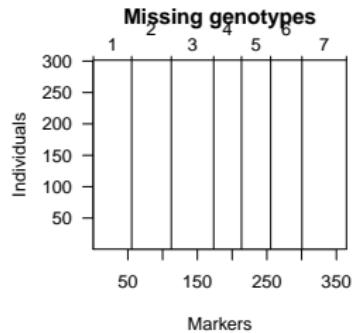
No. markers: 55 57 61 40 42 45 64

Percent genotyped: 100

Genotypes (%): AA:15.5 AB:84.5

Checking the data

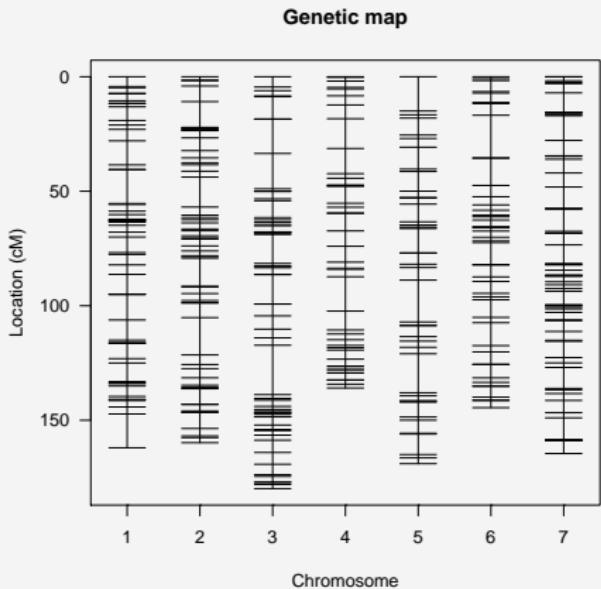
```
> plot(dat)
```



Checking the data

Plot the map

```
> map=pull.map(dat)  
> plot(map)
```



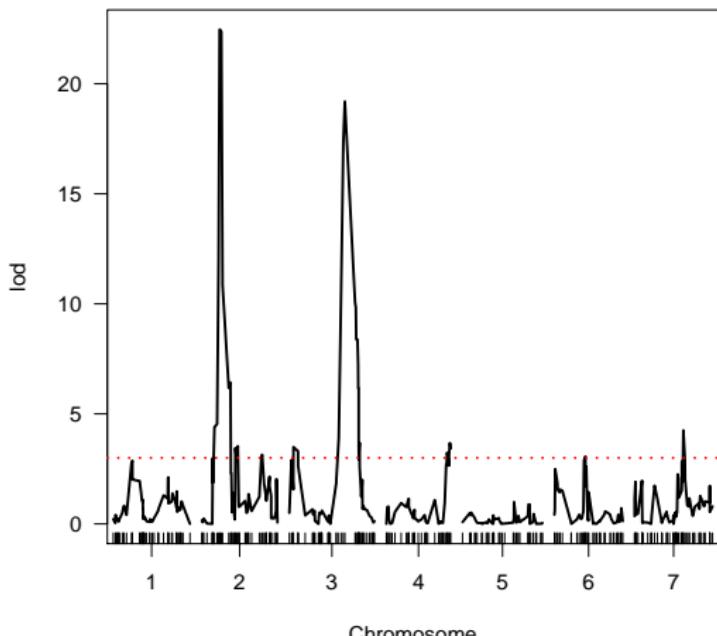
QTL mapping using marker regression

```
> meth_s=scanone(dat,method="mr")
> summary(meth_s)
```

	chr	pos	lod
GBM1042	1	40.5	2.873
PpdH1	2	41.1	22.462
bPb_9110	3	118.7	19.200
HDAMYB	4	146.9	3.678
bPb_0071	5	126.8	0.997
EBmac624	6	68.2	3.064
bPb_9914	7	103.4	4.252

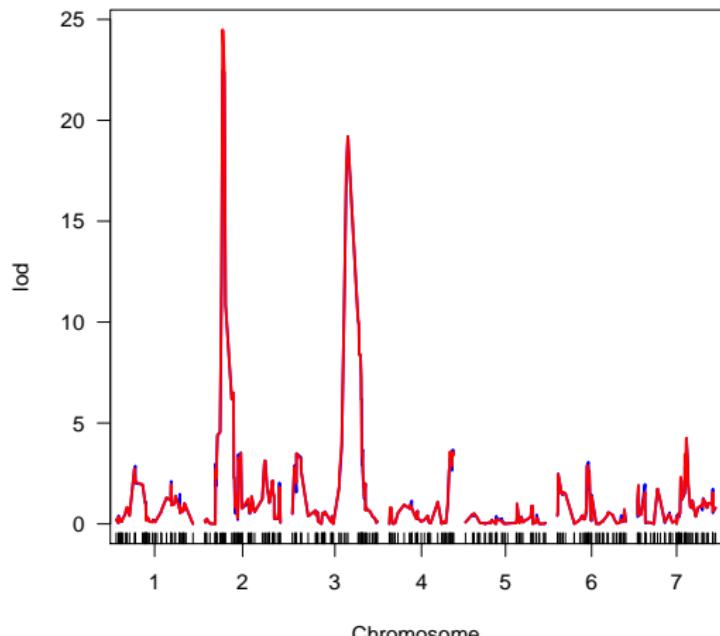
Plot the QTLs

```
> plot(meth_s)  
> abline(3,0,lwd=2,lty = "dotted",col="red")
```



Comparing interval mapping and marker regression

```
> meth_int=scanone(dat,method="em")
> plot(meth_s,meth_int,col=c("blue","red"))
```

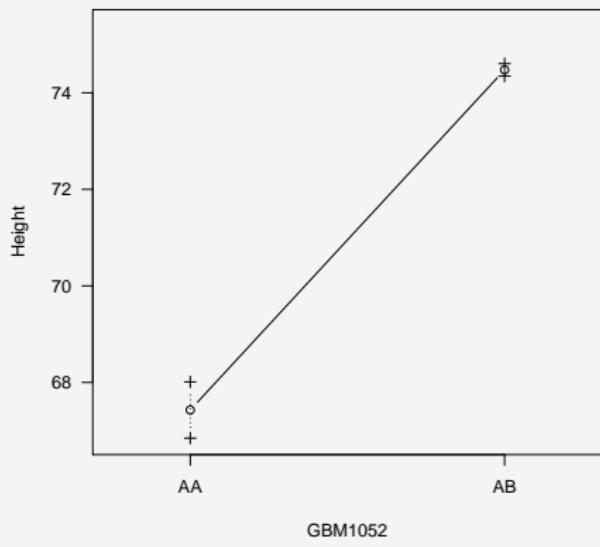


Effect plot

Difference in phenotype at the QTL location.

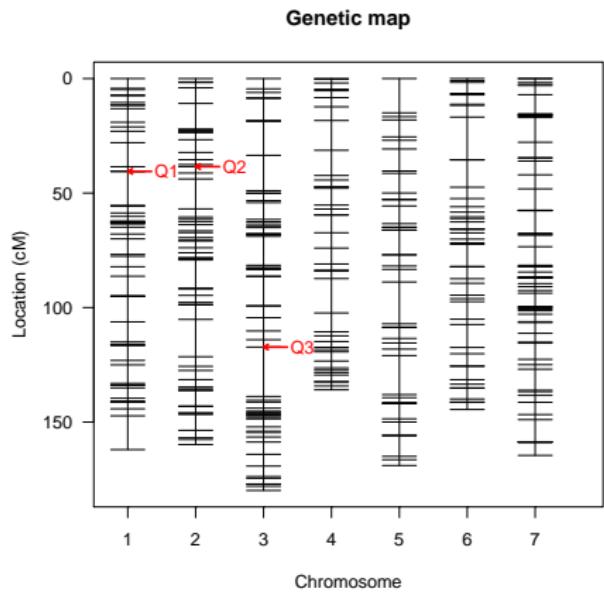
```
> effectplot(dat, mname1="GBM1052")
```

Effect plot for GBM1052



Plotting the QTLs on the chromosomes

```
> dat=sim.genotype(dat)
> qt=makeqtl(dat,chr=c("1", "2", "3"),pos=c(40.5,42.0,118.7))
> plot(qt)
```



Association mapping population



Phenotype file for rrBLUP

Gid	hea	height
Line1	72	102
Line2	69	103
Line3	71	100
Line4	77	102
Line5	74	103
Line6	75	104
Line7	71	101
Line8	70	103
Line9	72	104
Line10	67	105
Line11	73	102
Line12	70	104
Line13	75	105
Line14	72	106
Line15	74	103
Line16	72	105
Line17	76	106
Line18	77	107

Genotype file for rrBLUP

Marker	Chr	Pos	Line1	Line2	Line3	Line4
AF043094A	5	156	1	1	1	1
BMS2	5	45	1	1	1	1
BMS64	7	100.3	1	1	1	1
Bmac163	5	48.8	1	1	1	1
Bmac29	3	176	-1	1	1	-1
Bmac316	6	4.3	1	1	1	1
Bmac32	1	77.6	1	1	1	1
Bmac40	6	120	1	1	1	-1
Bmag105	1	63.5	1	1	1	1
Bmag11	7	87.5	1	1	1	1
Bmag120	7	107	1	1	1	1
Bmag125	2	95.4	1	1	1	1

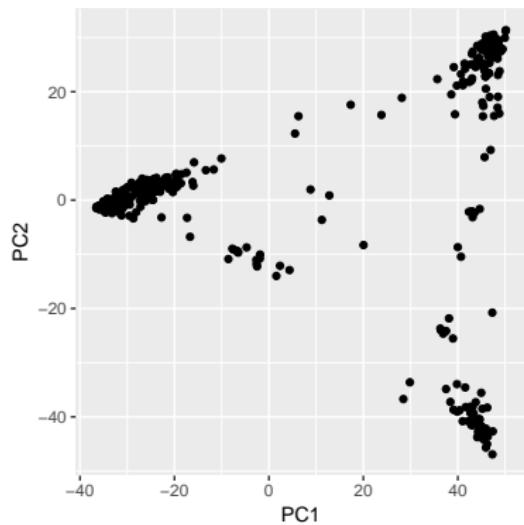
Reading data into R

Import data into R

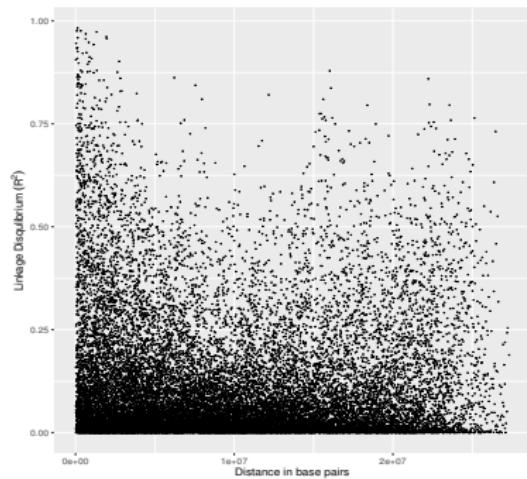
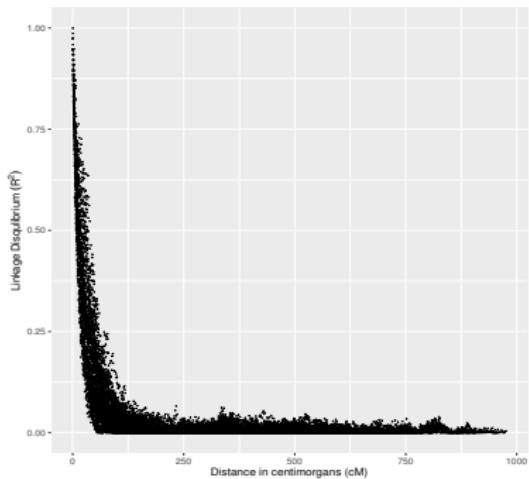
```
> library(rrBLUP)
> pheno=read.table("phenotype.txt",header=T)
> geno=read.table("genotype.txt",header=T,na.strings="NA")
> head(pheno,5)
```

```
    Gid hea
1 Line1  72
2 Line2  69
3 Line3  71
4 Line4  77
5 Line5  74
```

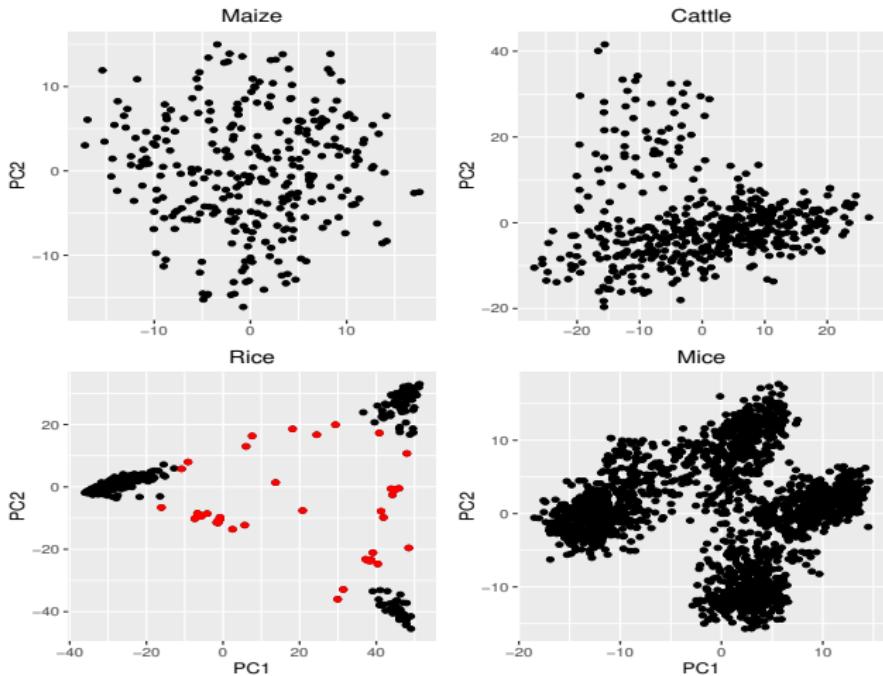
Population Structure and PCA



Patterns of LD



Examples of population structure



Looking for Genome wide association

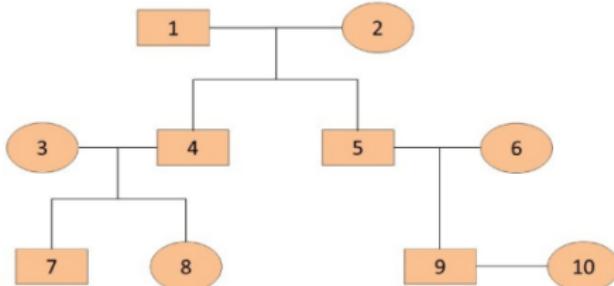
GWAS() function for genome wide association

```
> score=GWAS(pheno,geno,plot=F)  
[1] "GWAS for trait: hea"  
[1] "Variance components estimated. Testing markers."  
> new_score=score[order(-score$hea),]  
> head(new_score,5)
```

	Marker	Chr	Pos	hea
217	bPb_4261	2	44.7851	7.768013
339	bPb_9110	3	118.7241	3.889560
310	bPb_8038	2	38.9744	3.028613
52	GMS3	2	79.6000	2.605129
155	bPb_1579	3	115.5039	2.523110

Relatedness/kinship: what is it?

A Example Pedigree



B Corresponding Additive Genetic Relationship Matrix

	1	2	3	4	5	6	7	8	9	10
1	1	0	0	0.5	0.5	0	0.25	0.25	0.25	0
2	0	1	0	0.5	0.5	0	0.25	0.25	0.25	0
3	0	0	1	0	0	0	0.5	0.05	0	0
4	0.5	0.5	0	1	0.5	0	0.5	0.5	0.25	0
5	0.5	0.5	0	0.5	1	0	0.25	0.25	0.5	0
6	0	0	0	0	0	1	0	0	0.5	0
7	0.25	0.25	0.5	0.5	0.25	0	1	0.5	0.125	0
8	0.25	0.25	0.5	0.5	0.25	0	0.5	1	0.125	0
9	0.25	0.25	0	0.25	0.5	0.5	0.125	0.125	1	0
10	0	0	0	0	0	0	0	0	0	1

Relatedness: how to calculate and use it?

- Sources of relatedness

Relatedness: how to calculate and use it?

- Sources of relatedness
 - Pedigree based relatedness.

Relatedness: how to calculate and use it?

- Sources of relatedness
 - Pedigree based relatedness.
 - Molecular marker based relatedness.

Relatedness: how to calculate and use it?

- Sources of relatedness
 - Pedigree based relatedness.
 - Molecular marker based relatedness.
- How we use the kinship matrix

Relatedness: how to calculate and use it?

- Sources of relatedness
 - Pedigree based relatedness.
 - Molecular marker based relatedness.
- How we use the kinship matrix
 - Estimate heritabilities

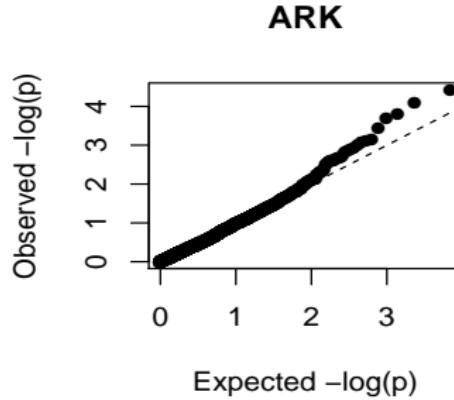
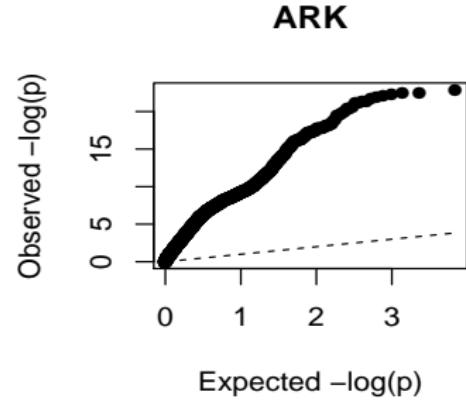
Relatedness: how to calculate and use it?

- Sources of relatedness
 - Pedigree based relatedness.
 - Molecular marker based relatedness.
- How we use the kinship matrix
 - Estimate heritabilities
 - For genomic selection/prediction.

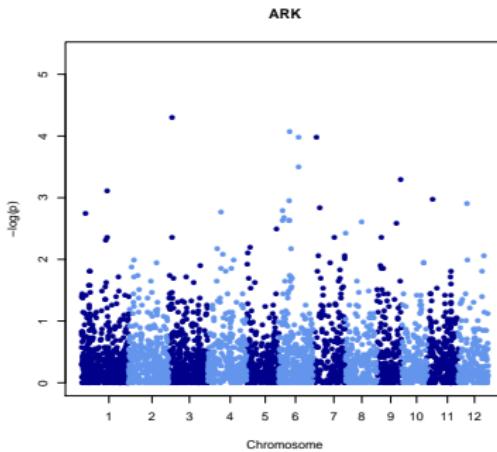
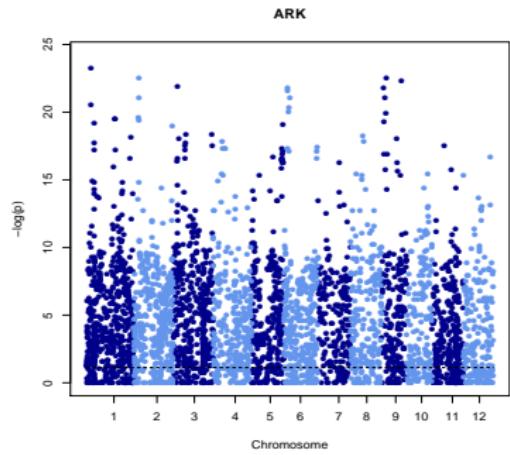
Relatedness: how to calculate and use it?

- Sources of relatedness
 - Pedigree based relatedness.
 - Molecular marker based relatedness.
- How we use the kinship matrix
 - Estimate heritabilities
 - For genomic selection/prediction.
 - To perform association studies

Why kinship is important



Why kinship is important



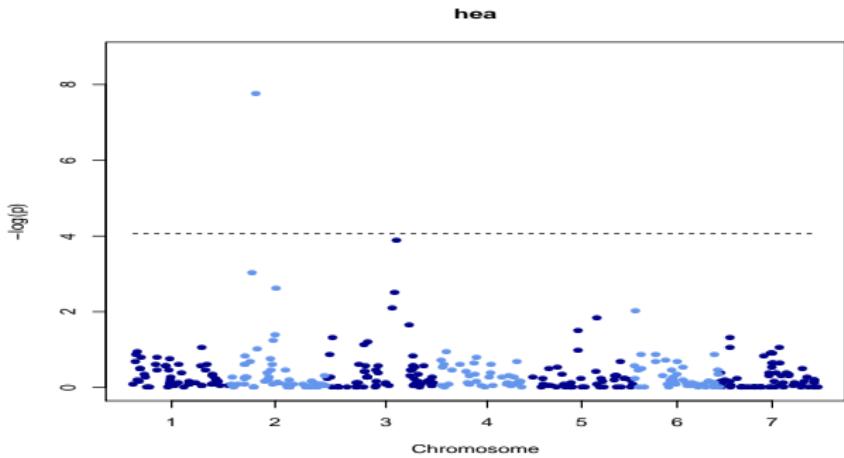
Looking for Genome wide association with PCA

GWAS() function for genome wide association

```
> score=GWAS(pheno,geno,plot=F,n.PC=3)
[1] "GWAS for trait: hea"
[1] "Variance components estimated. Testing markers."
> new_score=score[order(-score$hea),]
> head(new_score,5)
```

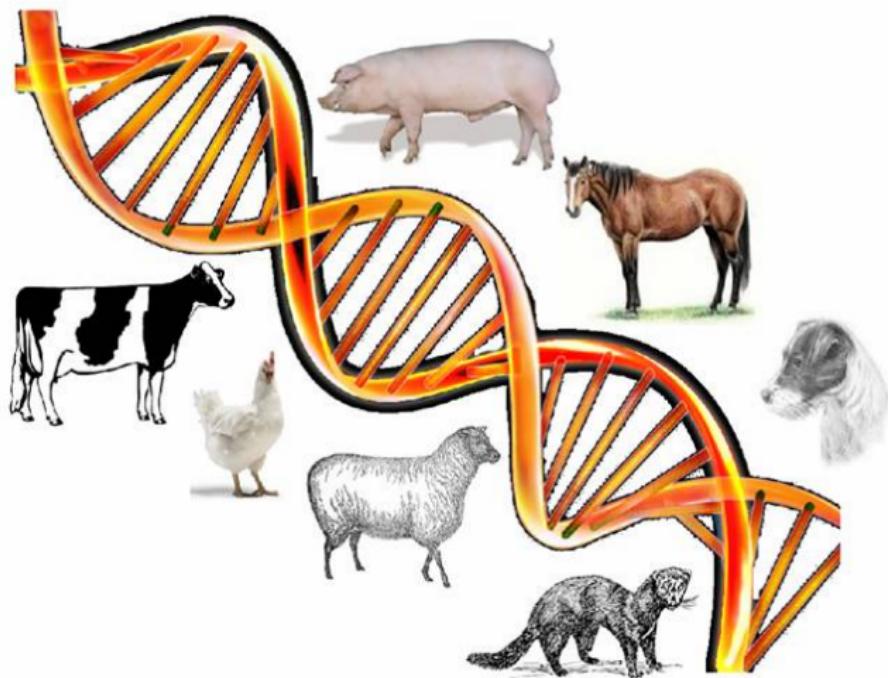
	Marker	Chr	Pos	hea
217	bPb_4261	2	44.7851	7.538538
339	bPb_9110	3	118.7241	3.794749
310	bPb_8038	2	38.9744	2.964361
52	GMS3	2	79.6000	2.688706
155	bPb_1579	3	115.5039	2.433262

Manhattan plot for the p-values



"Genomic Selection (GS)"

Genetic improvement by selection



Quantitative Trait Loci (QTL)

Marker assisted selection (MAS)

Marker assisted selection (MAS) uses molecular markers in linkage disequilibrium (LD) with QTL

- Detect the QTLs responsible
- Find the gene responsible in the QTL region.
- QTL effects on polygenic traits, such as milk yield, are likely to be small.
- Required a large number of QTLs to explain the genetic variation in these traits
- MAS need QTLs with large effect

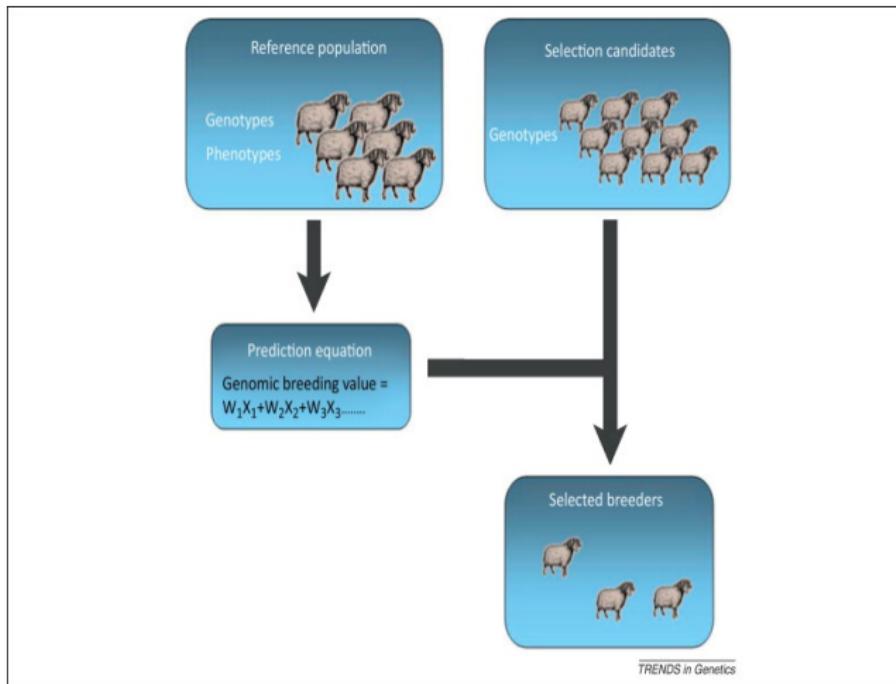
Genomic selection (GS) concepts

Training and validation set

Genomic selection (GS) is a new approach for improving quantitative traits which uses whole genome molecular markers information. High marker density is essential for genomic selection success.

- The training set: to estimate the marker effect
 - Marker information
 - Phenotype
- The validation set
 - Marker information

Work flow of Genomic Selection



Marker effects

Linear regression of marker effects

$$X_1\beta_1 + X_2\beta_2 + X_3\beta_3 + \dots + X_n\beta_n = Y_1$$

$$X_1\beta_1 + X_2\beta_2 + X_3\beta_3 + \dots + X_n\beta_n = Y_2$$

.....

.....

$$X_1\beta_1 + X_2\beta_2 + X_3\beta_3 + \dots + X_n\beta_n = Y_n$$

There are some variables already assigned in R

Here X_i' s can be seen as the marker information coded as -1 and 1. Vector β_i' s having the marker effect and Y_i' s are the phenotypes.

Different approaches for GS

More markers than individuals: Large p , Small n problem

- Shrinkage models: RR-BLUP and G-BLUP

Different approaches for GS

More markers than individuals: Large p , Small n problem

- Shrinkage models: RR-BLUP and G-BLUP
- Variable selection models: BayesB, BayesC, BayesD

Different approaches for GS

More markers than individuals: Large p , Small n problem

- Shrinkage models: RR-BLUP and G-BLUP
- Variable selection models: BayesB, BayesC, BayesD
- Kernel and machine learning methods: Support vector machine regression

Genomic selection using rrBLUP

Import data into R

```
> library(rrBLUP)
> pheno=read.table("phenotype.txt",header=T)
> geno=read.table("genotype.txt",header=T,na.strings="NA")
> head(pheno,5)
```

	Gid	hea
1	Line1	72
2	Line2	69
3	Line3	71
4	Line4	77
5	Line5	74

Genomic Estimated Breeding Value (GEBV)

Breeding value

```
> ans=kin.blup(data=pheno,geno="Gid",pheno="hea",K=A)
> ls(ans)
[1] "g"      "pred"   "resid"  "Ve"      "Vg"
```

Here g vector contains the BLUP values or the breeding values. Vg is the SNP genetic variance and Ve is the error variance.

Narrow sense heritability

```
> ans$Vg/(ans$Vg+ans$Ve)
[1] 0.7992445
```

Estimation of marker effect

mixed.solve() for the estimation of marker effect

```
> marker=read.table("marker.txt",na.strings="NA",header=T, row.names=1)
> colnames(marker)=geno$Marker
> ans <- mixed.solve(pheno$hea,Z=marker)
> head(ans$u,4)
```

AF043094A	BMS2	BMS64	Bmac163
-0.04886119	-0.12888651	0.15193866	-0.01623920