Introduction to Statistics using R/Rstudio
R and RStudio
Getting Started

• Assume that R for Windows and Macs already installed on your laptop. (Instructions for installations sent)
R version 3.0.0 (2013-04-03) -- "Masked Marvel"
Copyright (C) 2013 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.

Natural language support but running in an English locale

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.

[Previously saved workspace restored]

>
R on MACs

R version 3.0.0 (2013-04-03) -- "Masked Marvel"
Copyright (C) 2013 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin10.8.0 (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.

[R.app GUI 1.60 (6476) x86_64-apple-darwin10.8.0]
[Workspace restored from /Users/stephenopiyo/.RData]
[History restored from /Users/stephenopiyo/.Rapp.history]

>
RStudio
RStudio

• RStudio provides a nice work environment because it presents several windows on the screen that

• Make it easy to view code, results, and plots at once.
Using RStudio

- **Script editor**: View help, plots & files; manage packages.
- **R console**: View variables in workspace and history file.
Open RStudio and set up a working directory
Open RStudio and set up a working directory

New Folder

Please enter the new folder name

PP8300

OK  Cancel
Open RStudio and set up a working directory
R: Session management
R: session management

- You can enter a command at the command prompt in a RStudio console `>.

- To quit R, use `typr q()`.

- Result is stored in an **object** using the assignment operator: `<-` or the equal character `=`.
  Examples: `Test <- 2` or `Test = 2`

- Test is an object with a value of 2

- To print (show) the object just enter the name of the object

  - Test
Naming object in R

• Object names cannot contain `strange' symbols like !, +, -, #.

• A dot (.) and an underscore (_) are allowed, also a name starting with a dot (.)

• Object names can contain a number but cannot start with a number. (E.g., Example_1, not 1Example_1)

• R is case sensitive, X and x are two different objects, as well as temp.1 and tempP.1
R: session management

• Your R objects are stored in a workspace.

• Workspace is not saved on disk unless you tell R to do so.

• Objects are lost when you close R and not save the objects.

• R session are saved in a file with extension .RData.
R: session management

• Just checking what the current working directory type
  `getwd()`

• To list the objects in your workspace:  `ls()`

• To remove objects you no longer need:
  `rm()`

• To remove ALL objects in your workspace:
  `rm(list=ls())` or use Remove all objects

• To save your workspace to a file, you may type
  `save.image()` or use Save Workspace… in the File menu

• The default workspace file is called `.RData`
R: session management examples

• Example1 <- c(1,2,3,4,5)

• Example2 <- c("Second", "Example")

• Group_4<- c("Group",4)

• Save the three objects in the workspace and name is PP8300_E1

• Remove object Example1

• Save the remaining two objects in the workspace and name is PP8300_E2
Basic data types
Basic data types

- In this class we will use mainly three data types:
  - Logical (True or False)
  - Numerical (relating to numbers, e.g., 1, 0.1, -3, etc.)
  - Characters (letters, numerical digits, common punctuation marks such as "." or ",", etc.)
Vectors, Matrices, Data frame

• A vector
  – *Ordered collection* of data of the same type
  – Example: last names of all students in this class

• A matrix
  – *Rectangular table* of data of the same type
  – Example: Mean intensities of all genes measured during a microarray experiment

• A data frame
  – *Rectangular table with rows and columns*
  – May have the same or different type of data
Vectors

- Vector: Ordered collection of data of the same data type
  
  \[ X \leftarrow c(1, 2, 3, 4, 5) \]  
  \( c = \text{concatenate.} \)
  
  \[ Y \leftarrow 1:5 \]

  \[ \log(X) \]  
  do math on object X
  \[ [1] 1.6486586 0.5306283 1.8405496 \]

  \[ Z \leftarrow \text{seq}(1, 1.4, \text{by} = 0.1) \]
  \[ [1] 1.0 1.1 1.2 1.3 1.4 \]

  \[ YZ \leftarrow Y + Z \]
  \[ [1] 2.0 3.1 4.2 5.3 6.4 \]

  \[ \text{length}(Y) \]
  \[ [1] 5 \]

  \[ M \leftarrow \text{mean}(Y + Z) \]
  \[ [1] 4.2 \]
Operation on vector elements

- Mydata <- c(2, 3.5, -0.2)  Vector (c="concatenate")

Mydata
[1] 2 3.5 -0.2

x4 <- Mydata[1]
[1] 2

x5 <- Mydata[Mydata>0]
[1] 2 3.5

x6 <- Mydata[-c(1,3)]
[1] 3.5

- Extract element in the first position
- Extract the positive elements
- Remove elements 1 and 3
Operation on vector elements

- Colors <- c("Red","Green","Red")  Character vector

  Colors[2]
  [1] "Green"

  x1 <- 25:30 : Number sequences

  x1
  [1] 25 26 27 28 29 30

  x2 <- x1[3:5]
  [1] 27 28 29 Various elements 3 to 5

  x3 <- x1[c(2,6)] Elements 2 and 6
  [1] 26 30
Matrices

• Matrix: Rectangular table of data of the same type.
• Create a matrix with a function called `matrix`
  ```
m <- matrix(1:12) Create matrix using the “matrix function”
```
  ```
m [,1]
[1,]  1
[2,]  2
[3,]  3
[4,]  4
[5,]  5
[6,]  6
[7,]  7
[8,]  8
[9,]  9
[10,] 10
[11,] 11
[12,] 12
```

Create a vector of 1 to 12

```
V1 <- c(1,2,3,4,5,6,7,8,9,10,11,12) vector
```
**Matrices**

- **Matrix**: Rectangular table of data of the same type
  
  ```r
  mr <- matrix(1:12, 4) four rows
  mr
  [,1] [,2] [,3]
  [1,]  1  5  9
  [2,]  2  6 10
  [3,]  3  7 11
  [4,]  4  8 12
  ```

- **Matrix by row**:
  
  ```r
  mm <- matrix(1:12, 4, byrow = T); mm By row creation
  [,1] [,2] [,3]
  [1,]  1  2  3
  [2,]  4  5  6
  [3,]  7  8  9
  [4,] 10 11 12
  ```
Matrices

• Matrix: by row

```r
mm <- matrix(1:12, 4, byrow = T); mm

[,1] [,2] [,3]
[1,]  1  2  3
[2,]  4  5  6
[3,]  7  8  9
[4,] 10 11 12

tmm <- t(mm) t is transpose

[1,]  1  4  7 10
[2,]  2  5  8 11
[3,]  3  6  9 12
```
Operation on matrices

\[
x.\text{matr}[ ,2 ]
\]
\[
[1] \quad -1 \quad 0 \quad 6
\]

\[
x.\text{matr}[ c(1, 3), ]
\]
\[
[,1] [,2] \\
[1,] \quad 3 \quad -1 \\
[2,] \quad -3 \quad 6
\]

\[
x.\text{mat}[-2, ]
\]
\[
[,1] [,2] \\
[1,] \quad 3 \quad -1 \\
[2,] \quad -3 \quad 6
\]
Data frame

Data frame: Rectangular table with rows and columns; data within each column has the same type (e.g. number, text, logical), but different columns may have different types.

Example of a data frame with 10 rows and 3 columns

<table>
<thead>
<tr>
<th>Name</th>
<th>Phone</th>
<th>Country</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stephen</td>
<td>25677643240</td>
<td>Uganda</td>
</tr>
<tr>
<td>Richard</td>
<td>19545551234</td>
<td>USA</td>
</tr>
<tr>
<td>Monica</td>
<td>2547876655</td>
<td>Kenya</td>
</tr>
<tr>
<td>Fred</td>
<td>54113876</td>
<td>Argentina</td>
</tr>
<tr>
<td>Jessica</td>
<td>448756509</td>
<td>United Kingdom</td>
</tr>
<tr>
<td>Milly</td>
<td>3389756585</td>
<td>France</td>
</tr>
<tr>
<td>Norbert</td>
<td>5876454534</td>
<td>Venezuela</td>
</tr>
<tr>
<td>Jenifer</td>
<td>23480312345</td>
<td>Nigeria</td>
</tr>
<tr>
<td>Jimmy</td>
<td>866586968405</td>
<td>Taiwan</td>
</tr>
<tr>
<td>Rose</td>
<td>861069445464</td>
<td>China</td>
</tr>
</tbody>
</table>
Creating a data frame

- # create a data frame from scratch
- age <- c(25, 30, 56, 49, 12, 16, 60, 34, 45, 22)
- gender <- c("male", "female", "male","male", "female", "male","male", "female", "male","male")
- weight <- c(160, 110, 220, 100, 65, 120, 179, 134, 165, 153)
- mydata <- data.frame(age,gender,weight)

data.frame function
Importing and exporting data in R
Importing Data

• The easiest way to enter data in R is to work with a text file, in which the columns are separated by tabs, comma-separated values (csv), or excel files.

Example of importing data are provided below (console).
• mydata <- read.table("D1_Data_1.csv", sep=",", header=TRUE)
• mydata <- read.csv("D1_Data_1.csv", header=TRUE)
• mydatab<- read.table("D1_Data_1.txt", sep=",\t", header=TRUE)
• mydatab<- read.delim("D1_Data_1.txt", header=TRUE)

• Importing data in Rstudio using (Import Dataset) on the Workspace
Viewing Data

There are a number of functions for listing the contents of an object or dataset.

# list the variables in mydata
names(mydata)

# list the structure of mydata
str(mydata)

# dimensions of an object
dim(mydata)
Viewing Data

# class of an object (numeric, matrix, dataframe, etc)
class(mydata)

# print mydata
mydata

# print first 6 rows of mydata
head(mydata)

# print first 2 rows of mydata
head(mydata, n=2)

print last 6 rows of mydata
tail(mydata)

# print last 2 rows of mydata
tail(mydata, n=2)
Operation on Data Frame

Data_Frame[row, column]

Data_Frame[1,]  Data_Frame row 1

Data_Frame[,1]  Data_Frame column 1

Data_Frame[-1,]  Remove row 1 from Data_Frame

Data_Frame[,][-1]  Remove column 1 from Data_Frame

Data_Frame[c(1,3),]  Remove rows 1 and 3 from Data_Frame

Data_Frame[,][-1:3]  Remove columns 1 to 3 from Data_Frame
Missing Data

In R, missing values are represented by the symbol **NA** (not available). Impossible values (e.g., dividing by zero) are represented by the symbol **NaN** (not a number).

Testing for Missing Values
is.na(x) # returns TRUE of x is missing
y <- c(1,2,3,NA)
is.na(y) # returns a vector (F F F T)

Excluding missing values from analyses
Arithmetic functions on missing values yield missing values.
x <- c(1,2,NA,3)
mean(x) # returns NA
mean(x, na.rm=TRUE) # returns 2
Exporting Data

- **To A csv File**
  
  ```r
  write.table(mydata, "mydata.csv", sep="", ")
  write.csv(mydata, "mydata.csv")
  ```

- **To A Tab Delimited Text File**
  
  ```r
  write.table(mydata, "mydata.txt", , sep="\t ")
  ```

- **Exporting R objects into other formats**. For SPSS, SAS and Stata. you will need to load the **foreign** packages.