Week 1: Overview of R

CKMT 105

Data Science Certificate Program

Ryerson University

Bora Caglayan

15 Jan, 2015

========

Instructor

Bora Caglayan, PhD - bora.caglayan@ryerson.ca - http://bora.tarla.org

GA

Parisa Lak, PhD Student - parisa.lak@ryerson.ca

Data Science Laboratory

http://www.ryerson.ca/~abener/dsl.html

Course Logistics

Required Text: - Course notes - Handouts - All announcements through Blackboard CMS.

Recommended Reading: - Weekly reading material

Grading: - Lab Attendance - 15% - 3 homeworks - 30 (10%x3) - Homework deadlines (week 6, week 9, week 12) - Midterm - 20% - Final - 35%

Weekly Schedule Overview

- Each week a lecture followed by a lab session.
- Lab sessions are scored based on attendance. Collaboration during the labs is encouraged.
- Individual work on homeworks.
- Expected course work by the students is one hour per week.
- Homework sets will be given in the end of week 3, week 6 and week 8.
- Please check Blackboard CMS regularly and follow the announcements.
- Course notes will be provided before the lectures in Blackboard.

Weekly Schedule

- Week 1: Review of R
- Week 2: Review of R
- Week 3: Design of data analytics experiments
- Week 4: Univariate linear regression
- Week 5: Multivariate linear regression
- Week 6: Midterm
- Week 7: Introduction to classification, k nearest neighbour (KNN) model

Weekly Schedule

- Week 8: Logistic regression model
- Week 9: Bayesian theorem, naive bayes classifier
- Week 10: Data pre-processing
- Week 11: Complex networks, network measures
- Week 12: Centrality estimation in complex networks
- Week 13: Final

Motivation

Discussion of applications - Prediction of housing prices - Hand written digit recognition - Risk assessment models - Movie recommender models - Identification of influencers in social networks - ...

We need knowledge about both statistics and software tools to address these problems.

Motivation

title: false If there is a problem you can't solve, then there is an easier problem you can solve: find it.

George Pólya

Outline

- Setting up R environment
- Basic data structures
- Vector
- Matrix
- Data Frame
- Loading external files
- Custom Functions
- Importing Packages

Introduction

- R is one of the most popular languages for statistical programming.
- Used commonly by *data scientists* for prototyping in data science projects.
- Huge amount of libraries and resources are available on the web.
- These slides were prepared with R itself!
- In this lecture we will briefly go through the basic data structures in the R language and introduce statistical programming with R.

Some R history

- R is based on S programming language.
- S is a statistical programming language developed by John Chambers from Bell Labs.
- R was created by Ross Ihaka and Robert Gentleman at the University of Auckland, New Zealand as an open-source alternative for S.
- Over the years the popularity of R grew dramatically in the academy and the industry.

Setting up R Environment

- to practice basic commands online: http://r-fiddle.org (you can even use it from your smart phone!)
- R environment is set up previously in the labs for you.
- Setup R on your computer:
- Download and install R: http://www.r-project.org/
- Download and install Rstudio (optional): http://www.rstudio.com/

Interactive Mode vs Batch Mode

- R is an interpreted programming language.
- **Interactive mode**: Commands are evaluated line by line.
- After installing R, you may enter interactive mode by running it from the command line.
- Previously executed commands can be seen on R history.
- **Batch mode**: A script file is run as batch.

A rule of thumb: Interactive mode is good for prototyping with a few lines of code, batch mode is better for complex scripts.

R Workspace

- R workspace stores the command history and the objects in a session.
- You can save the state of your work in an .Rdata file.

Basic Data Structures

- In computer science, a data structure is a particular way of organizing data in a computer so that it can be used efficiently.
- In this lecture we will introduce three basic data structures in R.
- *Vector*: An array of numbers, characters or boolean values.
- *Matrix*: matrix in linear algebra
- *Data frame*: Two dimensional representation of a dataset with column and row labels.

Basic Data Structures

- Variable: Variable is a storage location paired with an associated symbolic name (an identifier), which contains some known or unknown quantity or information referred to as a value.
- Here is an example:

```
a_variable <- 1
another_variable <- "ckme"
```

Vector

title: True Here is a vector:

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

```
[1] 1 2 3
```

- Vector is an ordered list of values.
- It is similar to array structure found in popular programming languages such as Python and C.

Vector

title: True

Properties: - Vector indices start at 1. - You cannot change the length of a vector after initiation. - In R a vector may have a single *mode*. - In R, there is no scalar variable.

Vector

Basic modes of vectors:

```
1.0   #  double
```

```
[1] 1
```

```
TRUE  #  logical (True or False)
```

```
[1] TRUE
```

```
"a"   #  character
```

```
[1] "a"
```

They are vectors of length one.

Tip

title: False

Tip 1: You can easily check the mode of an R variable easily if you are not sure with the *typeof* function:

```
typeof("what am i?")
```

```
[1] "character"
```

Vector

Declarations:

```
y <- c(1,2,3,4)
z <- vector(length=2)
z[1] <- 1
z[2] <- 2
```

```
y
```

```
[1] 1 2 3 4
```

```
z
```

```
[1] 1 2
```

Vector

You can also define a vector ranging between number a and b as c(a:b).

Here is an example:

```
c(1:3)
```

```
[1] 1 2 3
```

Vector

You can also define a vector ranging between number a to b and with leaps=d as seq(a, b, d). Here is an example:

```
seq(0, 10, 2) #  even numbers between 0 and 10
```

```
[1]  0  2  4  6  8 10
```

You can also create vectors by picking numbers from a probability distribution.

Vector

Slicing vectors:

We can slice vectors to get a certain portion of them.

```
temp <- c(1,2,3,4,5,6,7,8,9)
temp[1]
```

```
[1] 1
```

```
temp[1:6]
```

```
[1] 1 2 3 4 5 6
```

Vector

Vector Arithmetics:

```
a <- c(1,2,3,4,5,6,7,8,9)
b <- c(9,8,7,6,5,4,3,2,1)
a+b
```

```
[1] 10 10 10 10 10 10 10 10 10
```

```
a*b
```

```
[1]  9 16 21 24 25 24 21 16  9
```

Vector

Vector Arithmetics:

```
a*b
```

```
[1]  9 16 21 24 25 24 21 16  9
```

```
a/b
```

```
[1] 0.1111111 0.2500000 0.4285714 0.6666667 1.0000000 1.5000000 2.3333333
[8] 4.0000000 9.0000000
```

Vector

Naming vector indices:

- We can name vector indices for convenience.

```
world_population <- c(1*10^9, 2*10^9, 5*10^9, 7*10^9)
world_population
```

```
[1] 1e+09 2e+09 5e+09 7e+09
```

```
names(world_population) <- c(1804, 1927, 1987, 2012)
world_population
```

```
 1804  1927  1987  2012
1e+09 2e+09 5e+09 7e+09
```

Vector

Vector Evaluation with any() and all(): - any

```
temp <- c(1,2,3)
any(temp > 2)
```

```
[1] TRUE
```

- all

```
all(temp > 2)
```

```
[1] FALSE
```

Vector

title: False *Tip 2:* - You can view the help document of a data structure or function by help(...).

An example:

```
help(seq)
```

Matrix

- Matrices can be used to perform linear algebra equations with R.
- They are similar to Matlab or Python (numpy) matrices.
- We will only cover them briefly since we will not use linear algebra in this course.

Matrix

```
matrix(c(1,2, 3,4),nrow=2,byrow=T)
```

```
     [,1] [,2]
[1,]    1    2
[2,]    3    4
```

```
a <- matrix(c(1,2, 3,4), nrow=2,byrow=T)
b <- matrix(c(1,2, 3,4), nrow=2,byrow=T)
a %*% b
```

```
     [,1] [,2]
[1,]    7   10
[2,]   15   22
```

Data Frame

- Data frame is a powerful data structure in R language.
- It can be used to store two dimensional data efficiently.

Data Frame

```
summary(mtcars)
```

```
      mpg             cyl             disp             hp       
 Min.   :10.40   Min.   :4.000   Min.   : 71.1   Min.   : 52.0  
 1st Qu.:15.43   1st Qu.:4.000   1st Qu.:120.8   1st Qu.: 96.5  
 Median :19.20   Median :6.000   Median :196.3   Median :123.0  
 Mean   :20.09   Mean   :6.188   Mean   :230.7   Mean   :146.7  
 3rd Qu.:22.80   3rd Qu.:8.000   3rd Qu.:326.0   3rd Qu.:180.0  
 Max.   :33.90   Max.   :8.000   Max.   :472.0   Max.   :335.0  
      drat             wt             qsec             vs        
 Min.   :2.760   Min.   :1.513   Min.   :14.50   Min.   :0.0000  
 1st Qu.:3.080   1st Qu.:2.581   1st Qu.:16.89   1st Qu.:0.0000  
 Median :3.695   Median :3.325   Median :17.71   Median :0.0000  
 Mean   :3.597   Mean   :3.217   Mean   :17.85   Mean   :0.4375  
 3rd Qu.:3.920   3rd Qu.:3.610   3rd Qu.:18.90   3rd Qu.:1.0000  
 Max.   :4.930   Max.   :5.424   Max.   :22.90   Max.   :1.0000  
       am              gear            carb      
 Min.   :0.0000   Min.   :3.000   Min.   :1.000  
 1st Qu.:0.0000   1st Qu.:3.000   1st Qu.:2.000  
 Median :0.0000   Median :4.000   Median :2.000  
 Mean   :0.4062   Mean   :3.688   Mean   :2.812  
 3rd Qu.:1.0000   3rd Qu.:4.000   3rd Qu.:4.000  
 Max.   :1.0000   Max.   :5.000   Max.   :8.000  
```

Data Frame

Selecting values from a dataframe

1. Select by index numbers

```
mtcars[1,1]
```

```
[1] 21
```

2. Select by labels

```
mtcars["Mazda RX4", "mpg"]
```

```
[1] 21
```

Data Frame

Listing column names

```
names(mtcars)
```

```
 [1] "mpg"  "cyl"  "disp" "hp"   "drat" "wt"   "qsec" "vs"   "am"   "gear"
[11] "carb"
```

Getting help

```
help(mtcars)
```

Data Frame

Adding a row

```
mtcars["BMW 520",] <- seq(1, 11)
mtcars["BMW 520",]
```

```
        mpg cyl disp hp drat wt qsec vs am gear carb
BMW 520   1   2    3  4    5  6    7  8  9   10   11
```

```
mtcars["BMW 520", "mpg"] <- 40
mtcars["BMW 520",]
```

```
        mpg cyl disp hp drat wt qsec vs am gear carb
BMW 520  40   2    3  4    5  6    7  8  9   10   11
```

Data Frame

Get a column

```
mtcars$mpg
```

```
 [1] 21.0 21.0 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 17.8 16.4 17.3 15.2
[15] 10.4 10.4 14.7 32.4 30.4 33.9 21.5 15.5 15.2 13.3 19.2 27.3 26.0 30.4
[29] 15.8 19.7 15.0 21.4 40.0
```

Get multiple columns

```
mtcars[,c("mpg", "cyl")]
```

```
                   mpg cyl
Mazda RX4         21.0   6
Mazda RX4 Wag     21.0   6
Datsun 710        22.8   4
Hornet 4 Drive    21.4   6
Hornet Sportabout 18.7   8
Valiant           18.1   6
Duster 360        14.3   8
Merc 240D         24.4   4
Merc 230          22.8   4
Merc 280          19.2   6
Merc 280C         17.8   6
Merc 450SE        16.4   8
Merc 450SL        17.3   8
Merc 450SLC       15.2   8
Cadillac Fleetwood  10.4   8
Lincoln Continental 10.4   8
Chrysler Imperial   14.7   8
Fiat 128          32.4   4
Honda Civic       30.4   4
Toyota Corolla    33.9   4
Toyota Corona     21.5   4
Dodge Challenger  15.5   8
AMC Javelin       15.2   8
Camaro Z28        13.3   8
Pontiac Firebird  19.2   8
Fiat X1-9         27.3   4
Porsche 914-2     26.0   4
Lotus Europa      30.4   4
Ford Pantera L    15.8   8
```

```
Ferrari Dino        19.7   6
Maserati Bora       15.0   8
Volvo 142E          21.4   4
BMW 520             40.0   2
```

Data Frame

Select a row

```
mtcars["BMW 520",]
```

```
        mpg cyl disp hp drat wt qsec vs am gear carb
BMW 520  40   2    3  4    5  6    7  8  9   10   11
```

Select multiple rows

```
mtcars[c("Mazda RX4","BMW 520"),]
```

```
          mpg cyl disp  hp drat   wt  qsec vs am gear carb
Mazda RX4  21   6  160 110  3.9 2.62 16.46  0  1    4    4
BMW 520    40   2    3   4  5.0 6.00  7.00  8  9   10   11
```

Data Frame

Peeking a data frame - When a data frame is really large we can check the first few lines with *head* function.

```
head(mtcars)
```

```
                   mpg cyl disp  hp drat    wt  qsec vs am gear carb
Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

Data Frame

Creating data frame from vectors

```
cbind(year=seq(2000, 2015, 5), weight=c(65, 80, 92, 90))
```

```
     year weight
[1,] 2000     65
[2,] 2005     80
[3,] 2010     92
[4,] 2015     90
```

Data Frame

Further filter examples

Find the maximum mpg

```
max(mtcars$mpg)
```

```
[1] 40
```

Get the number of 6 cylinder cars

```
nrow(mtcars[mtcars$cyl == 6,])
```

```
[1] 7
```

Data Frame

Export a data frame as text file

```
write.table(mtcars, "mtcars.csv", sep = ",", col.names = NA, qmethod = "double")
```

Data Frame

Import external data

```
mtcars_copy <- read.csv("mtcars.csv", sep=",")
```

Tip 3 : If you have an excel document save it as a csv file before importing to R.
R built-in functions ========== Here are some examples:

```
max(c(1,2,3))
```

```
[1] 3
```

```
min(c(1,2,3))
```

```
[1] 1
```

```
help(max) # gives some documentation about a built-in function
```

A custom function

```
say_hi <- function(txt){
   # paste function concatanates two strings.
   return_text <- paste("Hello ", txt)
   return(return_text)
}
say_hi("R")
```

```
[1] "Hello  R"
```

In this class you will not write any custom R functions.

Package import

- Packages are collections of R functions, data and compiled code in a well-defined format.
- We can import non-standard libraries by using the *require* function.
- Here is a package import example:

```
require(Matrix)
```

- New Packages can be installed to R instance from Rstudio gui.
- There is a package for a lot of applications. This is one of the main strengths of the R ecosystem.
- The full list: http://cran.r-project.org/web/packages/available_packages_by_name.html

Further reading

- [book] The Art of R Programming (26$): http://www.amazon.ca/The-Art-Programming-Statistical-Software/dp/1593273843
- Free Online interactive tutorial: http://tryr.codeschool.com/
- Free comprehensive introduction: http://www.r-tutor.com/

Lab Section

Part 1

Hands-on review of the code in the slides. - In this part, we will introduce R studio and check the basic data structures on our lab desktops.

Lab Section

Part 2

Lab exercises 1. Form a vector with the leap years in the 20th century (1902 is a leap year). 2. Find the maximum sepal length in iris dataset. 3. Export iris dataset to a csv file. Import it to a different variable.