SAKARYA ÜNİVERSİTESİ

BSM 461

INTRODUCTION TO BIG DATA

Lecture 3 – R Programming

Kevser Ovaz Akpınar, PhD

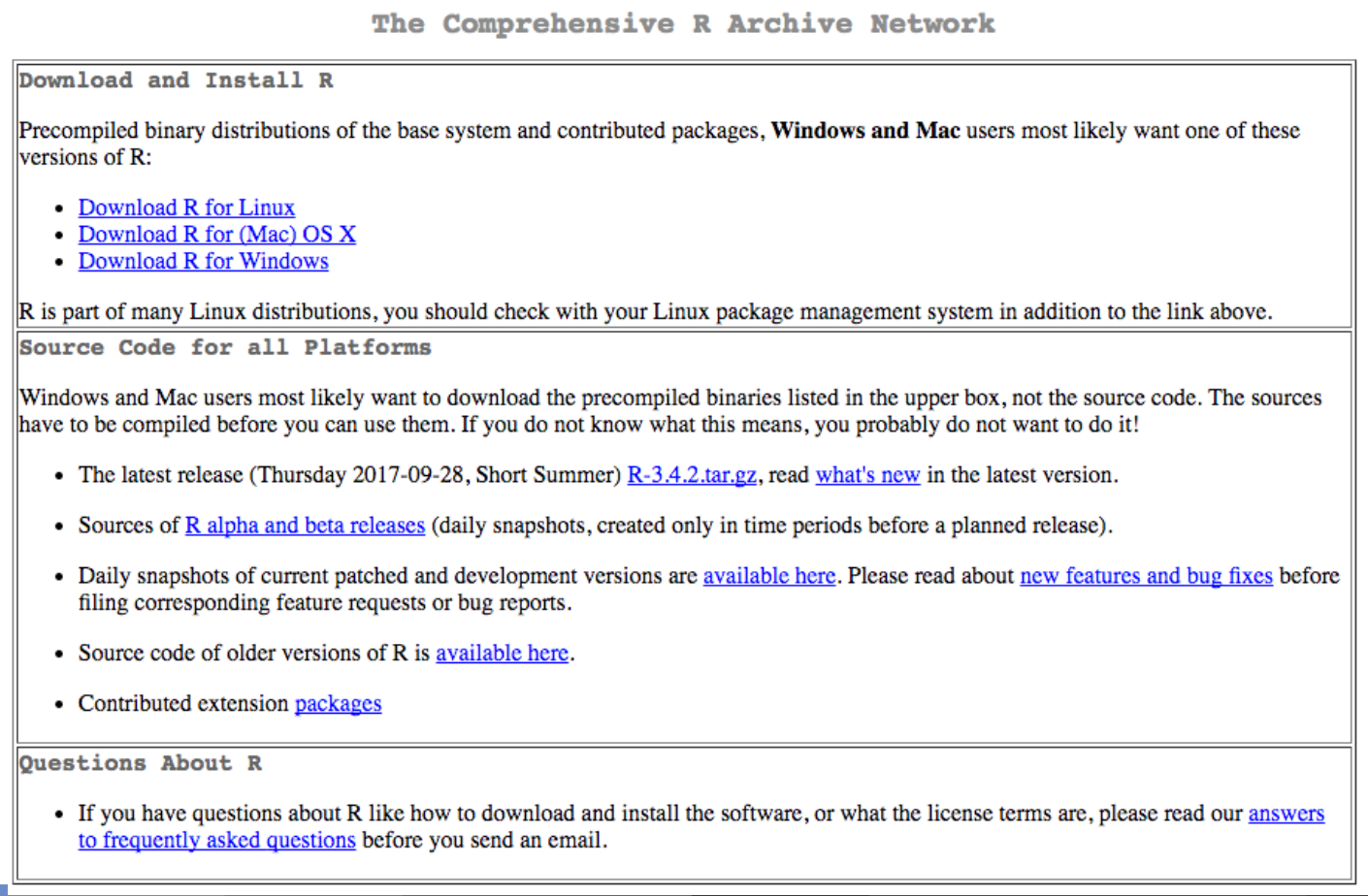
kovaz.sakarya.edu.tr
kovaz@sakarya.edu

Agenda

- R on OS
- What is R?
- Why R?
- Data Types in R
- Subsetting
- Libraries
- Exercises

R on OS

- R can be downloaded from the CRAN repository (https://cran.r-project.org)



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!

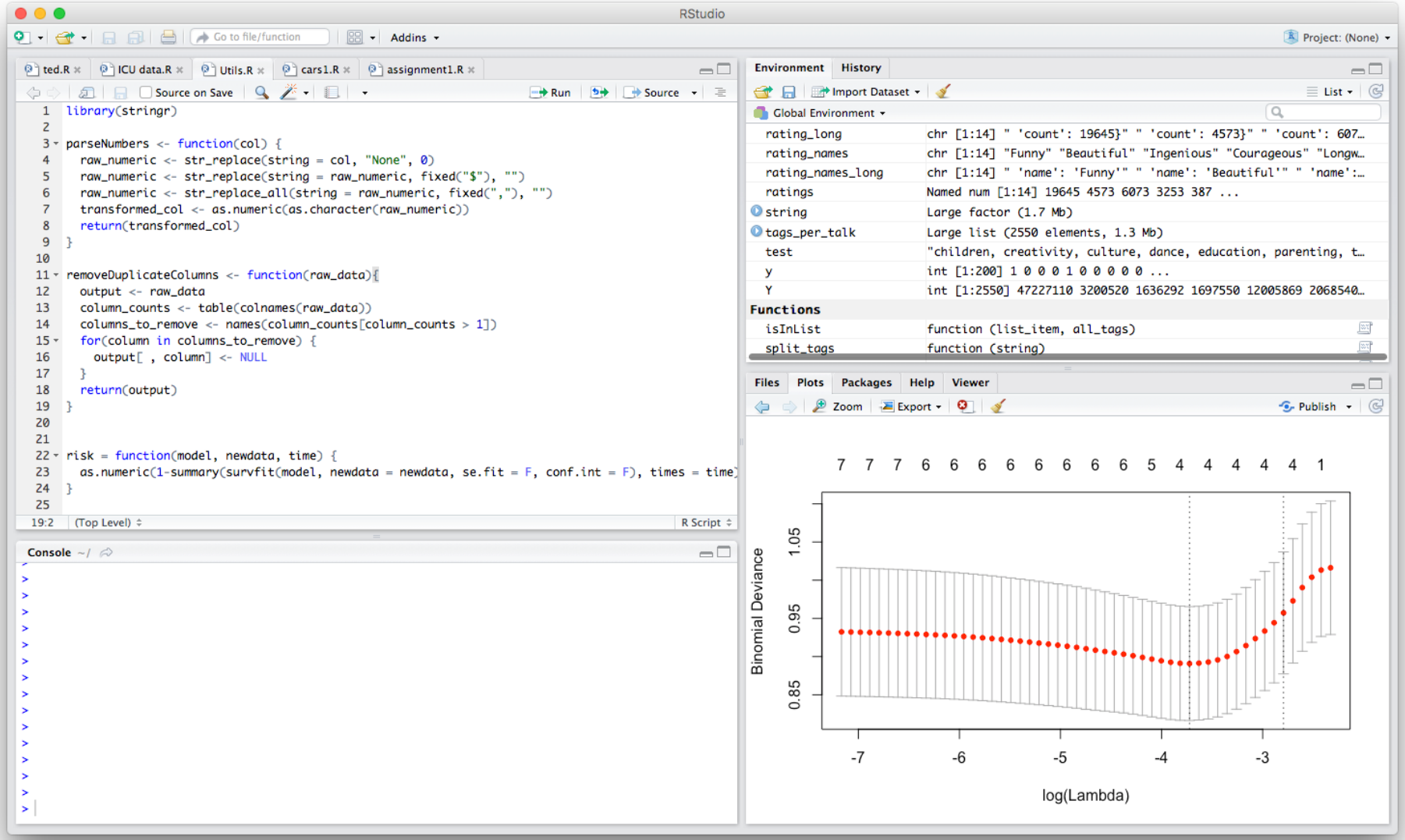
- The latest release (Thursday 2017-09-28, Short Summer) R-3.4.2.tar.gz, read what's new in the latest version.
- Sources of R alpha and beta releases (daily snapshots, created only in time periods before a planned release).
- Daily snapshots of current patched and development versions are available 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.

R on OS

- R-studio is an IDE that helps R users be more productive.
- Makes basic analysis, graphics, and code development easier and more efficient.
- Can be found at www.rstudio.com/products/rstudio/download/



What is R?

- R is a scripting language that is very similar to Python
- More focused on the data analysis aspect than Python

- Free language that is used for “statistical computing and graphics which provides a wide variety of statistical and graphical techniques: linear and nonlinear modeling, statistical tests, time series analysis, classification, clustering, etc.”

- Python or R ?

When reading a file, Python reads “sample.txt” line by line and consumes only required memory for a single line. But R reads entire file once and consumes more memory and may be hangs!

Why Use R?

If you currently use another statistical package, why learn R?

- It's free! If you are a teacher or a student, the benefits are obvious.
- It runs on a variety of platforms including Windows, Unix and MacOS.
- It provides an unparalleled platform for programming new statistical methods in an easy and straightforward manner.
- It contains advanced statistical routines not yet available in other packages.
- It has state-of-the-art graphics capabilities.
- R is great for standalone analysis.
- Used by a large number of statisticians. Primarily in academia.
- Historically, has had better packages for newer numerically/statistically based algorithms.

Why Not R?

- Harder to implement in full stack (production) environments.
- Less adoption from developers and programmers .
- Not as useful for tasks such as web-scrapping, lots of string manipulation, etc.
- Python is gaining a ton of ground with non-statisticians who are doing Data Science (lots of new packages).

Data Types in R

Primitive Data types

- Character
- Logical
- Function
- Numeric
  - **Notice, there aren’t separate types for floats, ints, etc.
  - **All numbers are just of type numeric

Variables

```
> a = 49
> sqrt(a)
[1] 7
```

numeric

```
> a = "I like pizza"
> sub("pizza","soup",a)
[1] "I like soup“
```

character string

```
> a = (1+1==3)
> a
[1] FALSE
```

logical

Vectors, Matrices, and Arrays

- vector: an ordered collection of data of the same type

```
> a = c(1,2,3)
> a*2
[1] 2 4 6

> a = c(“alpha”, “omega”, 5)
> a
[1] “alpha” “omega” “5”

> a = 5
> a[1]
[1] 5
```

- In R, a single number is the special case of a vector with 1 element.

Vectors, Matrices, and Arrays

- matrix: a rectangular table of data of the same type

```
> a <- matrix(1:3, 2, 3)
> a
     [,1] [,2] [,3]
[1,]    1    3    2
[2,]    2    1    3
```

**Notice, R “recycles” values. Python does not (e.g. np.arange(25).reshape((5,5)) wouldn’t work)

Assign operator (<-) works both ways, usually pronounced “gets”

1:3 shortcut to build a sequence by 1 from 1 to 3.

Vectors, Matrices, and Arrays

- array: 3-,4-,..dimensional matrix

- list: an ordered collection of data of arbitrary types.

```
> doe = list(name="john",age=28,married=F)
> doe$name
[1] "john“
> doe[[“age”]]
[1] 28
> doe[[3]]
[1] FALSE
```

- Typically, vector elements are accessed by their index (an integer), list elements by their name (a character string). But both types support both access methods.
- Lists can also contain larger items such as matrices, data frames, other lists, etc.

Data Frames

data frame: is supposed to represent the typical data table that researchers come up with – like a spreadsheet.

It is a 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.

```
> a
      localization  tumorsize  progress
XX348    proximal        6.3         0
XX234      distal        8.0         1
XX987    proximal       10.0         0
```

Factors

A character string can contain arbitrary text. Sometimes it is useful to use a limited vocabulary, with a small number of allowed words. A factor is a variable that can only take such a limited number of values, which are called levels.

```
> a
[1] Male   Female Male   Female Female Female
Levels: Female Male
> class(a)
[1] "factor"
> as.character(a)
[1] "Male"   "Female" "Male"   "Female" "Female" "Female“
> as.integer(a)
[1] 2 1 2 1 1 1
```

Subsetting in R

Individual elements of a vector, matrix, array or data frame are accessed with “[ ]” by specifying their index, or their name

```
> a
          localization   tumorsize    progress
XX348     proximal         6.3           0
XX234      distal          8.0           1
XX987     proximal        10.0           0
> a[3, 2]
[1] 10
> a["XX987", "tumorsize"]
[1] 10
> a["XX987", ]
          localization    tumorsize     progress
XX987     proximal          10             0
```

** Notice, R does NOT use the 0 index

Subsetting in R

Can also use the $ operator to access lists and data frames

```
> a$tumorsize
[1] 6.3  8.0  10.0
```

Negative index deletes value from vector/list

```
> a$tumorsize[-2]
[1] 6.3 10.0
```

No way to index last item in R. Alternatives:

```
> a$tumorsize[length(a$tumorsize)]
[1] 10.0
> a$tumorsize[nrow(a)]
[1] 10.0
```

Subsetting in R

```
> a
       localization   tumorsize     progress
XX348     proximal         6.3            0
XX234       distal         8.0            1
XX987     proximal        10.0            0
> a[c(1,3),]
       localization     tumorsize   progress
XX348     proximal         6.3            0
XX987     proximal        10.0            0
> a[c(T,F,T), 2:3]
       tumorsize    progress
XX348     6.3             0
XX987     10.0            0

>  a$localization=="proximal"
[1]  TRUE FALSE  TRUE

> a[ a$localization=="proximal", ]
       localisation tumorsize   progress
XX348     proximal       6.3          0
XX987     proximal      10.0          0
```

subset rows by a vector of indices

Subset by logical vector and a slice

comparison resulting in logical vector

subset the selected rows

Loops

- R uses curly braces {} instead of white space.

```
for(i in 1:10) {
  print(i*i)
}

i=1
while(i<=10) {
  print(i*i)
  i=i+sqrt(i)
}

X = 5
If(x==5){return(x)}else{return(y)}
```

**Notice, white space doesn’t matter in R

Functions

```
dummyFunction = function(a,b) {
a * b
}
```

- Notice, functions start and end on {}
- If there is no return statement, R will automatically return the last line (not good practice unless writing very short/one-line functions).
- Functions may also do other things than returning a result. E.g., plot something on the screen: “side effects”

- In R, there are methods but they aren’t as straight forward as in Python
  - In Python model.summary(), in R, you would do summary(model)
- However, what is actually being called is summary.lm (for a linear model).
  - Other functions that have object methods are plot, print, etc.
- You can build methods for different object types using the setMethod.

Common Libraries/Packages

- stringr – for manipulating strings
- plyr, dplyr, data.table – easily manipulate/read data
- lattice, ggplot2 – data visualization
- lubridate – manipulating dates
- glmnet – Lasso/ridge regression

Installation

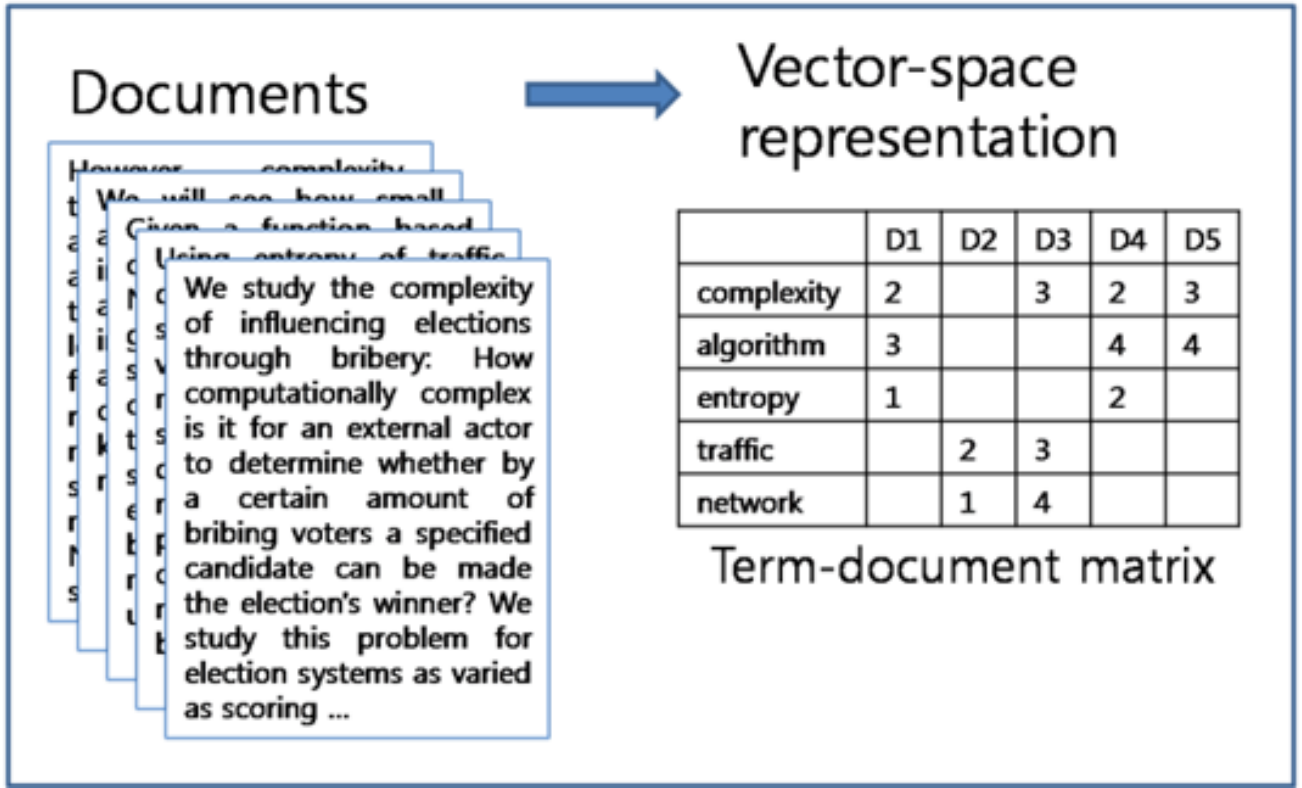
- install.packages() or the GUI in R-Studio
- To load an individual library – library(ggplot2)
- Can call individual functions without loading a library by using the double :
  - scales::dollar(20000)
- Can access internal functions by using the triple :
  - package:::fun(x)

3 Main Structures of R in Text Mining

1. **String**
2. **Corpus** : A collection of text documents that we can include in the analytics.

Mainly additional metadata and details are also stored with the strings. (Vcorpus and Pcorpus)

3. **Document-term matrix** : For each document a column, each term a row is used.



	D1	D2	D3	D4	D5
complexity	2		3	2	3
algorithm	3			4	4
entropy	1			2	
traffic		2	3		
network		1	4		

ADDITIONAL REFERENCES

BYU, Big Data Science & Capstone Lecture Notes - Python

Stanford University Lecture Notes,
http://web.stanford.edu/class/cs102

Chapter 7, Hands-On Big Data Modeling, James Lee, Tao Wei, Suresh Kumar Mukhiya