National Research University Higher School of Economics, Faculty of Social Sciences

Introduction to Network Analysis, Spring 2019

Seminar 1 and Homework 1 Assignment, 01/18/2019

Dmitry Zaytsev, PhD and Valentina Kuskova, PhD

(with deep appreciation to all used sources; references available in text and upon request)

Welcome to the course! As we have discussed in the first lecture, the purpose of this course is to help you *think* differently about data, assuming relational, rather than independent, data points. But this course will also teach you to work with data differently, learning state-of-the-art tools of data analysis and presentation. We will learn network analysis with R (with a brief introduction to UCINet, Visone and Pajek, for those most adventurous), and learn to create documents using R Markdown - an awesome program for authoring HTML, PDF, MS Word documents and slideshows.

This document is an R Mardown document, created in R studio, and this first seminar and homework assignment will guide you through installation of the programs and introduction to their usage. For today's seminar, the following files are posted on LMS (Week 1 folder):

- This document in PDF;
- R and Latex Installation Guides (prepared for us by our assistant, Stas Pashkov);
- R Markdown Template, in .Rmd format - a document that is already set up in a way that will help you get started on creating your own Markdown documents. It is also posted in a .pdf format, so you can see how R Markdown text translates into a PDF document;
- carprice.csv - data file in .csv format for your first homework;
- Lecture 1 slide deck.

Upon completion of Seminar 1 and Homework 1 assignments, you should have the following accomplished:

- Installed R and R-studio;
- Installed R Mardown;
- Installed the environment to convert R Mardown documents to PDF (MikTex or other, see below);
- Have gone through the basics of R programming in either Russian or English, become familiar with R and R-studio environments and understood basic commands enough to complete the actual assignments.

Seminar 1 assignment. Please reproduce the R code in this document and answer the questions that are marked as *Assignment questions*. Please include your answers right after each question in your RMarkdown file. You should submit both the .Rmd and the .pdf files with results.

Homework 1 assignment is provided at the end of this document.

Both assignments are due on **Monday, January 25, by midnight**. All submissions must be done via the LMS system (via Projects link, there is a project folder for every assignment). Late assignments are **not** accepted for **any** reason; at midnight on the due date the carriage turns into a pumpkin. If you do not submit your assignment by then, you get a grade of 0.

Getting hands-on with R

There are many ways to start learning R. We suggest that you use Swirl as an interactive option to learn R right in the R-studio environment. Alternatively, we have provided brief instructions in this document (to

also demontrate capabilities of RMarkdown). No matter the way you go, R is a fun and very powerful tool, and we hope you become familiar with it enough to use it long after this class ends, for all of your computing needs.

Important: This first assignment is quite involved - it requires installation of several programs onto your computers. There could be some issues with version compatibility, time delays with slow internet, etc. If you had R and RStudio installed prior to first lab, you should do just fine; if not, please do your best to follow along with someone else.

Installing necessary software

R is a very powerful statistical instrument that (through open source and user support) has by far exceeded capabilities of other programs such as SPSS, SAS and Stata. It used to require a lot of programming, but now there are many user-written packages that do the nitty gritty work for us. There is still a bit of a learning curve, but we will take it one step at a time, starting with the basics. Those of you who are familiar with R already and have all programs installed on your computer, can take the first week easy. The rest - do not worry; you need not be an experienced programmer in order to start navigating in R very quickly.

Installation order

Because we had all software installed on our machines long ago, we do not have a good understanding of the best order to install packages onto a machine where none were installed previously. However, upon examining our workspace, If you do not have any of the required software on your machines, then install in the following order:

1. R first
2. LaTex files second
3. R-studio last

We do not have an order preference, and for that reason, installation of LaTex files is covered first. Again, this is optional for those who want to use RMarkdown to create documents. What is mandatory, however, is that you install R-studio AFTER you install R.

Generating PDFs from R Studio

There are several ways to generate PDFs in R Studio, but since RMarkdown has that functionality already, you will only need to install the necessary libraries for LaTex language support. We use MikTex, but apparently, the file is very heavy and could take quite some time to download. For one of our prior classes, Stas Pashkov created a document with several alternative ways to load LaTex on your machine (available from your Lab 1 folder), so please take a look and choose the one that works the best for you.

Installing R and R Studio

Go to http://swirlstats.com/students.html. Install (in that order!) R, then R-studio. Make sure you install a free version of R Studio (not server; we won't need it for this class).

R and R studio help guides

There are many, many references that will help you learn R. The list below is just to get you started; you should be able to find ample sources on Google if you need more help. For now, these are just basics; we'll start working with network packages next week.

1. https://cs.hse.ru/data/2015/04/03/1096436989/Torfs+Brauer-Short-R-Intro.pdf

2. https://cs.hse.ru/data/2015/05/13/1098775155/R%20Tutorial.pdf
3. https://cs.hse.ru/data/2015/04/03/1096437593/R-refcard.pdf
4. https://support.rstudio.com/hc/en-us
5. http://www.statmethods.net/
6. http://www.burns-stat.com/pages/Tutor/hints_R_begin.html
7. http://data.princeton.edu/R/gettingStarted.html
8. http://www.ats.ucla.edu/stat/R/sk/
9. Of course, just as for any data analytics nowdays, there is a Coursera course for R as well: https://www.coursera.org/learn/r-programming
10. There are two books in Russian on R; they are available from your Readings folder.

RMarkdown

RMarkdown is simply awesome. Once you learn to use it, you will never go back to other editing software, we promise! Despite its simplicity, it's quite powerful and allows you to take advantage of best features of R (you can embed fully functioning code chunks) and LaTex (you can create beautiful documents); you can create HTML, PDF, and Word outputs with the same script. No other software, as far as we know, has this capacity.

To install R Markdown, go here: http://rmarkdown.rstudio.com/. The site has full instructions, though just as Swirl, Markdown is installed as a package - just follow the instructions. The very first Markdown template that gets generated upon installation will have a set of basic instructions, and the button you will learn to love, the **Knit** button, will generate a document with both the content and the embedded R chunks.

There are many useful hints right on that website. In addition, you will find other useful references and links, such as the ones below, to help you learn to format and render your documents just the way you like them. We will greatly appreciate the time you take to learn new tips and tricks, and will give extra-credit points to assignments that go above and beyond the very basic requirements that we specify for each assignment.

Here are some useful references for R Markdown:

1. http://www.stat.cmu.edu/~cshalizi/rmarkdown/#mark-up-markdown
2. https://cs.hse.ru/data/2015/04/03/1096437543/1501.01613v1.pdf
3. https://www.rstudio.com/wp-content/uploads/2016/03/rmarkdown-cheatsheet-2.0.pdf
4. https://www.rstudio.com/wp-content/uploads/2015/03/rmarkdown-reference.pdf
5. http://web.stanford.edu/class/psych252/section/Rmarkdown_info.html
6. RMarkdownhelpinRussian:http://dkhramov.dp.ua/Comp/FromMarkdownToLaTeX

Have fun exploring this wonderful new tool!

Learning R

There are many, many ways to get started. We recommend that you use Swirl, a package that will work right from inside the R. There is also some literature in Russian to help you, and

to demonstrate capabilities of R Markdown, we also included several basic commands with explanations at the end of this file. No matter the way, do not wait until the last possible moment to start working on learning R! Otherwise, you will quickly fall behind.

Swirl

Swirl is just a package, similar to all other packages we will be running, and it is step 3 on the web site above. During the first seminar, we will show you how to install package in R and run it, though there is nothing more to it than simply to run the code at the prompt, as shown on the web site:

install.packages("swirl")

Then, after the package is installed, type (after the ">" prompt, which shows up automatically)

library("swirl")

hit Enter, then

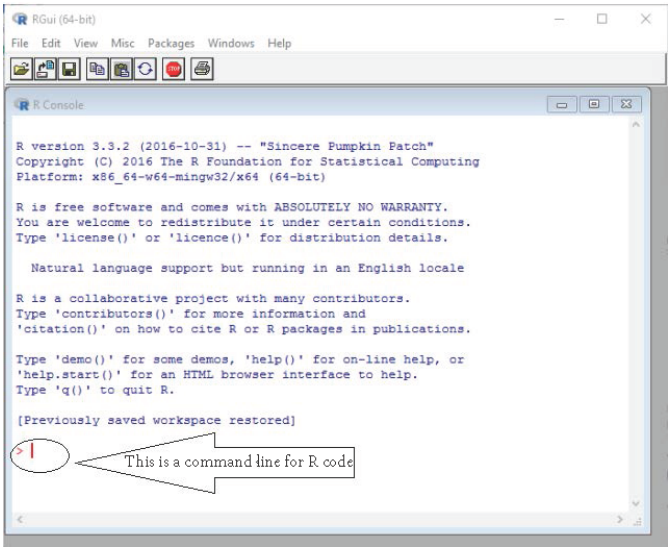
swirl()

and hit Enter again to run Swirl. During the first seminar, we will show you how to work with it. To start, complete basic Swirl lessons in R programming: 1, 3-8 (sufficient for now), the rest - optional.

Learning the Old Way

The R-Studio

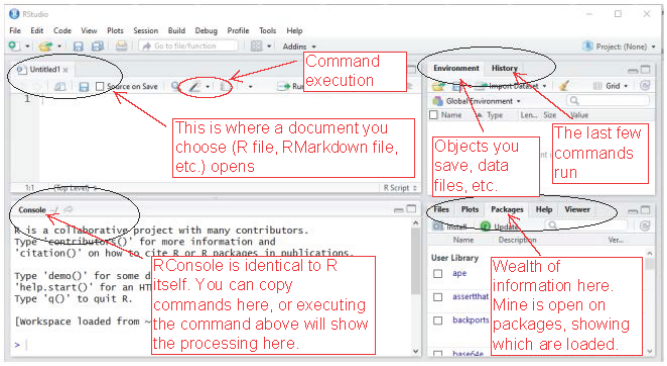
Before R-studio, the user interface for R, was created, all we had was the RConsole - simply a window for writing the code. Some of us still use RConsole when we do heavy-duty programming, because RStudio code sometimes interferes with some user-created functions. But the RConsole simply looked like this:



The RConsole is where you type all your commands, and simply hitting "Enter" will execute the command. Please note that every new line in R starts with the ">" (greater than) sign. It's a signal for R to start a new line, and a signal for you that the code has been executed

correctly. If you do not see the ">" sign after an executed command, but see a plus sign ("+") instead, it means that the code line above was incomplete.

The RConsole alone was not very user-friendly; a better version of it, called RStudio, was released several years ago.



The R-Studio has a lot of features, which make our lives much easier. Some of the most common we will discuss in class, but please feel free to explore on your own.

Before we begin: The R Help Files

R has built-in help, which simply starts with a question mark. In the help-page for each function (called by ?functionName) you can find how to use a function including the required input types and the default specifications. The good thing about r is that it is very "google-able". If you want to calculate the e.g. standard deviation, just search in google: "calculate standard deviation in R." You can also do a keyword-search the following way, but we had better experience with google.

The code below is commented out to keep RMarkdown document clean. Remove the ## from the code and copy it into your RConsole in order to see the results that the execution produces. Note where in RStudio the answers appear.

For exampme, the mean() command calculates the mean of an object (often a vector/ series of numbers. check also the standard deviation: note that we did not know what the command was, so we simply typed the name of the function we needed.

```
## ?mean
## ??standarddeviation

# If you know or remember the command, but not how to use it, type:

## ?sd
```

For a list and details about available functions see help(Arithmetic), help(Trig), help(complex), help(log), help(Round), help(sqrt), etc.

The R Basics

Below are some instructions and commands in R to help you get started right from this file.

```
print("Hello World")
```

```
## [1] "Hello World"
```

"Print" is a command, or a function. functions need one or more arguments that you write in parantheses behind the functionName(). Note that R is case-sensitive, i.e. Print("Hello World") does not work (Capital P).

R can work as a calculator. Try the following commands:

```
1+5
```

```
## [1] 6
```

```
105*99+6
```

```
## [1] 10401
```

```
exp(0.09855)
```

```
## [1] 1.10357
```

```
mean(c(1, 5, 8, 7, 6, 4, 22, 1, 0.9))
```

```
## [1] 6.1
```

In the normal case, however, you will want to use a script in order to retrace what you have done, be able to repeat what you have done, and to be able to share what you have done.

Objects in R

We generally work with so called objects in R. An object can be a lot different things. You can think of an object as a container, in which you can put whatever you want (numbers, matrices, words, functions...). Objects have a name and content. To store a computed value we use an assignment statement. The "<-" assigns a value to an object. Here are some very basic objects with values assigned to them:

```
NumberA <- 5

NumberB <- exp(2)

VectorC <- c(1,5,8,7,6,4,22,1,0.9,500)
```

c(1,5,8,7,6,4,22,1,0.9) is a vector that is used for these commands. Vectors with a series of integers can be obtained by the colon:

```
VectorD <- 5:15
```

or by the sequence command

```
quarters <- seq(0, 5, by=0.25)
```

we can also create matrices:

```
MatrixE <- matrix(0, 5, 5)
MatrixF <- matrix(1:4, 6, 6)
```

Each of these objects can be called by their name:

```
NumberA
```

```
## [1] 5
```

```
NumberB
```

```
## [1] 7.389056
```

```
MatrixF
```

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

Objects are the core of R. When using R to analyze networks, everything we do are operations on objects. Our data is stored in an object, the algorithm to estimate, the effects we include...

Operations on Objects

We can do the same type of operations we do we do in the console also with these objects:

```
NumberA*NumberB
```

```
## [1] 36.94528
```

The content of one object is multiplied with the content of another object.

```
NumberA*VectorC
```

```
##  [1]    5.0   25.0   40.0   35.0   30.0   20.0  110.0    5.0    4.5 2500.0
```

```
mean(VectorC)
```

```
## [1] 55.49
```

Most of the time, operations can have multiple arguments if the operation is executed on multiple objects or where you can specfify some more parameters. Multiple arguments are separated by a comma ","

```
intersect(VectorC, VectorD)
```

```
## [1] 5 8 7 6
```

```
mean(VectorC)
```

```
## [1] 55.49
```

```
mean(VectorC, trim = 0.1)
```

```
## [1] 6.75
```

We can also nest operations within one another:

```
mean(intersect(VectorC,VectorD))
```

```
## [1] 6.5
```

Creating objects from operations

We can create new objects from these operations, which we do extensively, because we don't want to create all objects "on foot":

```
product <- NumberA*VectorC
product
```

```
##  [1]    5.0   25.0   40.0   35.0   30.0   20.0  110.0    5.0    4.5 2500.0
```

```
otherNumber <- NumberA*var(VectorC)
otherNumber
```

```
## [1] 122158.6
```

Working with Data

If you attended our first school of this year, TMSA-2017(I), the following chunk of code will look familiar to you. Vladimir Batagelj has done a wonderful job of introducing us to R, so we are incorporating his work into our lab here.

The basic data types in R are: numeric, integer, complex, logical, character, and raw. They can be combined into structured objects using: vector, matrix, array, list, etc.

For testing and converting types we have functions:

1. is.type(x) - is the object x of type type ?
2. as.type(x) - try to convert object x into object of type type.
3. typeof(x) - returns the type of x; similar class(x) and mode(x).

R is a programming language. About the basic control statements see Vlado's existing presentation http://tinyurl.com/Vlado-R-Controls.

Variables and data tables

In a standard setting we have an (ordered) set of **units**. To describe them we select a set of their **properties** (attributes). Data are obtained by **measuring** the properties of units. Here, we have a data table that contains information on **units** (in rows) and their **attributes** (in columns).

	P_1	P_2	P_3	$\cdots$	P_m
U_1	$v_{1,1}$	$v_{1,2}$	$v_{1,3}$	$\cdots$	$v_{1,m}$
U_2	$v_{2,1}$	$v_{2,2}$	$v_{2,3}$	$\cdots$	$v_{2,m}$
U_3	$v_{3,1}$	$v_{3,2}$	$v_{3,3}$	$\cdots$	$v_{3,m}$
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
U_n	$v_{n,1}$	$v_{n,2}$	$v_{n,3}$	$\cdots$	$v_{n,m}$

In statistics, vectors - columns in the data table are called **variables**. Because of this in R the basic data structure is a vector. Most operations and functions are working vector-wise.

Try the following code:

```
a <- c(1,2,3,4,5)
a
```

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

```
b <- c(10,9,8,7,6)
b
```

```
## [1] 10  9  8  7  6
```

```
# To find the fourth element in b:
b[4]
```

```
## [1] 7
```

```
b[c(1,3,5)]
```

```
## [1] 10  8  6
```

```
length(a) # Gives you the number of elements in a
```

```
## [1] 5
```

```
# Some more simple operations
a+b
```

```
## [1] 11 11 11 11 11
```

```
a-b
```

```
## [1] -9 -7 -5 -3 -1
```

```
a*b
```

```
## [1] 10 18 24 28 30
```

```
sqrt(a)
```

```
## [1] 1.000000 1.414214 1.732051 2.000000 2.236068
```

```
2**a # Notice multiplying number by a vector
```

```
## [1]  2  4  8 16 32
```

```
# Now, look at several sum/summary commands:
summary(b)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       6       7       8       8       9      10
```

```
sum(b)
```

```
## [1] 40
```

```
cumsum(b)
```

```
## [1] 10 19 27 34 40
```

```
#And some more manipulations:
integer(10)
```

```
##  [1] 0 0 0 0 0 0 0 0 0 0
```

```
1:9
```

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

```
rep(c(0,1),5)
```

```
##  [1] 0 1 0 1 0 1 0 1 0 1
```

```
seq(1,3,1/3)
```

```
## [1] 1.000000 1.333333 1.666667 2.000000 2.333333 2.666667 3.000000
```

```
c(rep(0,5),rep(1,7))
```

```
##  [1] 0 0 0 0 0 1 1 1 1 1 1 1
```

Measurement

Not all properties of units can be measured, as in physics and geometry, by real numbers. The measurements are made in different **measurement scales** (You can find more information here: https://en.wikipedia.org/wiki/Level_of_measurement and here: https://www.cambridge.org/core/books/measurement-theory/7D75B72C3E5FA676EA7AD6AB4D8DF4A7).

- **Numerical** scales
  - **absolute** (counts)
  - **interval** (temperature F and C, date)
  - **ratio**, or having absolute zero (weight, length, price, temperature K))
- **Ordinal** (grades: excellent, good, average, poor, unsatisfactory; temperature: very cold, cold, cool, warm, hot, very hot)
- **Nominal or Categorical** (nationality, religious preference: Buddhist, Muslim, Christian, Jewish, Other)

The type of the measurement scale determines what we can do with the corresponding variables. For example - central element: geometric mean, (arithmetic) mean, median, mode.

Nominal and ordinal variables in R

Values of numerical scale measurements are represented with integers or real numbers. Integers are often used (as codes) also for representing ordinal and nominal values - but not all numerical operations on them are meaningful.

```
# Notice what the following code is doing:
t <- c("F","D","F","I","A","F","F","D","B","SLO","I","F","GB","B")
```

```
T <- factor(t)
t
```

```
##  [1] "F"   "D"   "F"   "I"   "A"   "F"   "F"   "D"   "B"   "SLO" "I"
## [12] "F"   "GB"  "B"
```

```
T
```

```
##  [1] F   D   F   I   A   F   F   D   B   SLO I   F   GB  B
## Levels: A B D F GB I SLO
```

```
as.integer(T)
```

```
##  [1] 4 3 4 6 1 4 4 3 2 7 6 4 5 2
```

```
levels(T)
```

```
## [1] "A"   "B"   "D"   "F"   "GB"  "I"   "SLO"
```

```
levels(T)[as.integer(T)]
```

```
##  [1] "F"   "D"   "F"   "I"   "A"   "F"   "F"   "D"   "B"   "SLO" "I"
## [12] "F"   "GB"  "B"
```

```
which(T=="F")
```

```
## [1]  1  3  6  7 12
```

```
table(T)
```

```
## T
##   A   B   D   F  GB   I SLO
##   1   2   2   5   1   2   1
```

```
L <- c("unsatisfactory","poor","average","good","excellent")
s <- c("unsatisfactory","good","good","average")
k <- factor(s,levels=L,ordered=TRUE)
k
```

```
## [1] unsatisfactory good           good           average
## Levels: unsatisfactory < poor < average < good < excellent
```

```
as.integer(k)
```

```
## [1] 1 4 4 3
```

R is also unicode. That means, you can code your variables in Russian if you choose to do so.

Assignment:

1. Compute the empirical probability distribution and the cummulative empirical probability distribution of values in vector t.
2. Create a vector L and s (as above) in Russian.

Data Frames

Data tables are represented in R as data frames (R code is "data.frame"). Data frame is essentially a special list in which all items are vectors (columns, variables) are of the same length. A data frame can be created with an assignment of the form:

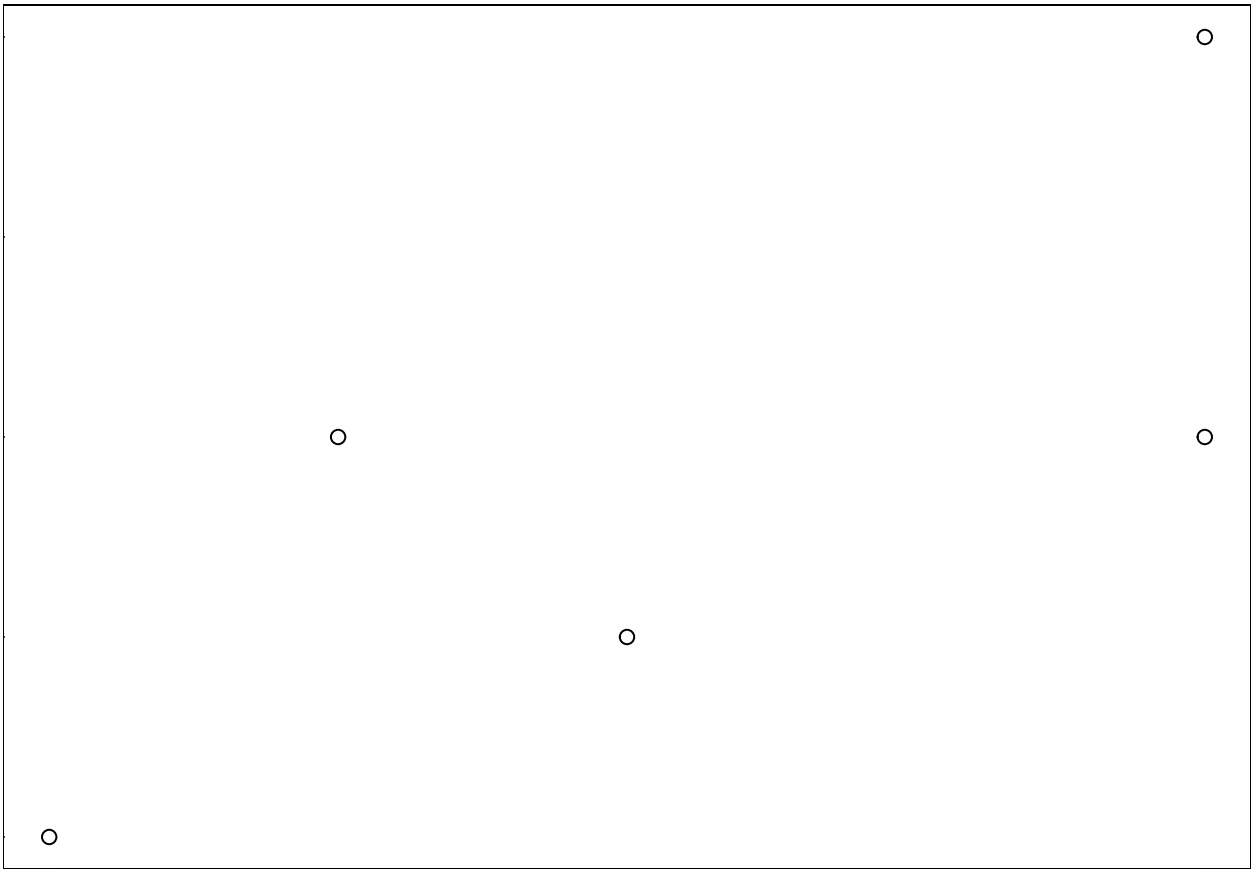
D <- data.frame(e1, e2, e3, ..., ek, row.names=units)

where "units" is a vector containing names of units, and e_i has either a form $n_i = v_i$ or v_i (where n_i is the name of the $_i$-th column and v_i is the vector of values).

```
units <- c('a','b','c','d','e')
u <- c(1,2,3,5,5)
v <- c(1,3,2,3,5)
D <- data.frame(U=u,V=v,row.names=units)
D
```

```
##   U V
## a 1 1
## b 2 3
## c 3 2
## d 5 3
## e 5 5
```

```
par(mar=c(0,0,0,0))
plot(D) # this will give you a picture
```



We can get the $_i$th variable by expression D[[i]].

```
# Extract the second item in the data frame:
D[[2]]
```

```
## [1] 1 3 2 3 5
```

A data frame D can be created also interactively using

D <- edit(data.frame())

We can also prepare it as an Excel spreadsheet, save it as a CSV file, and read it into R (we'll talk about it in one of the following labs).

Assignment. Represent the following data as a data frame:

	Income	**Age**	**Number of kids**	**Education**
Alex	2,000	50	2	12
Vlado	1,000	25	1	16

Other useful information about R

Working directory

R needs to know where to find files on your computer. Usually there is a default working directory that you can find using the getwd() command. We can change it using the function setwd(*[path]*) command or going to Session->Set Working Directory menu option in RStudio.

Saving files

In addition to familiar *Save* and *Save as* commands in the File menu, you can also use the *save*.Rdata using the command

dump(c("v1","v2",. . . ,"vk"),"save.Rdata")

When we need these files again, we can call them using

source("save.Rdata")

R Packages

The Basic R has only the limited number of functions. There are many, many more that have been added by users. Because not all the code is required by everyone, expanding the software itself is unreasonable. Therefore, people created a variety of packages that contain only the functions that they would need for a certain task. We will be using several main SNA packages, and at the beginning of every lab, we will list the necessary packages (which you add using the 'install.packages' command) for that particular lab.

R packages also contain data. Explore a few, as is shown in the code below:

```
data(package=.packages(all.available=TRUE))
```

```
## Warning in data(package = .packages(all.available = TRUE)): datasets have
## been moved from package 'base' to package 'datasets'
```

```
## Warning in data(package = .packages(all.available = TRUE)): datasets have
## been moved from package 'stats' to package 'datasets'
```

```
library(cluster)
data(ruspini)
head(ruspini)
```

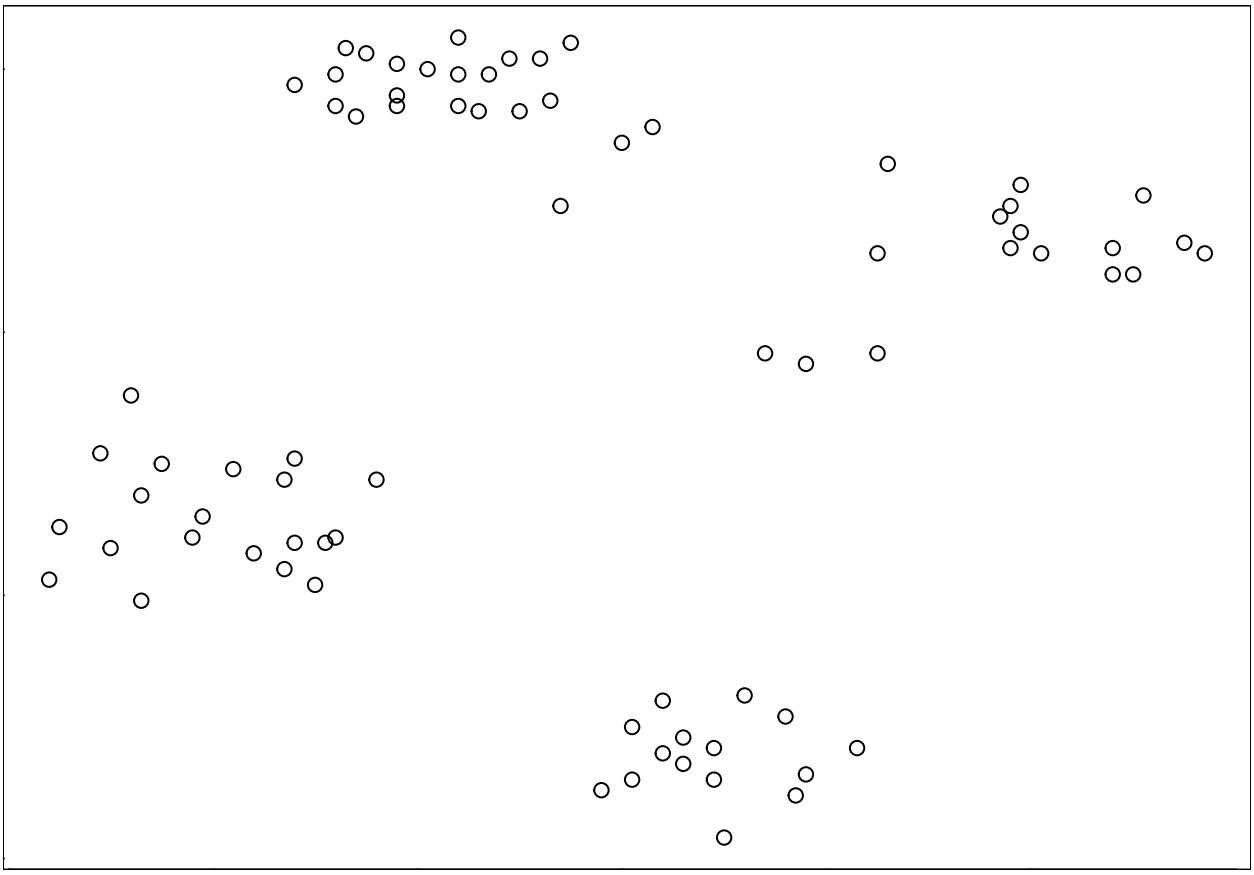
```
##    x  y
## 1  4 53
## 2  5 63
## 3 10 59
## 4  9 77
## 5 13 49
## 6 13 69
```

```
summary(ruspini)
```

```
##        x                y
##  Min.   :  4.00   Min.   :  4.00
##  1st Qu.: 31.50   1st Qu.: 56.50
##  Median : 52.00   Median : 96.00
##  Mean   : 54.88   Mean   : 92.03
##  3rd Qu.: 76.50   3rd Qu.:141.50
```

```
##  Max.   :117.00   Max.   :156.00
```

```
par(mar=c(0,0,0,0))
plot(ruspini)
```



```
# See also data sets: flower, plantTraits, animals, iris, mtcars, etc.
data(iris)
dim(iris)
```

```
## [1] 150   5
```

```
summary(iris)
```

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

```
##
##
```

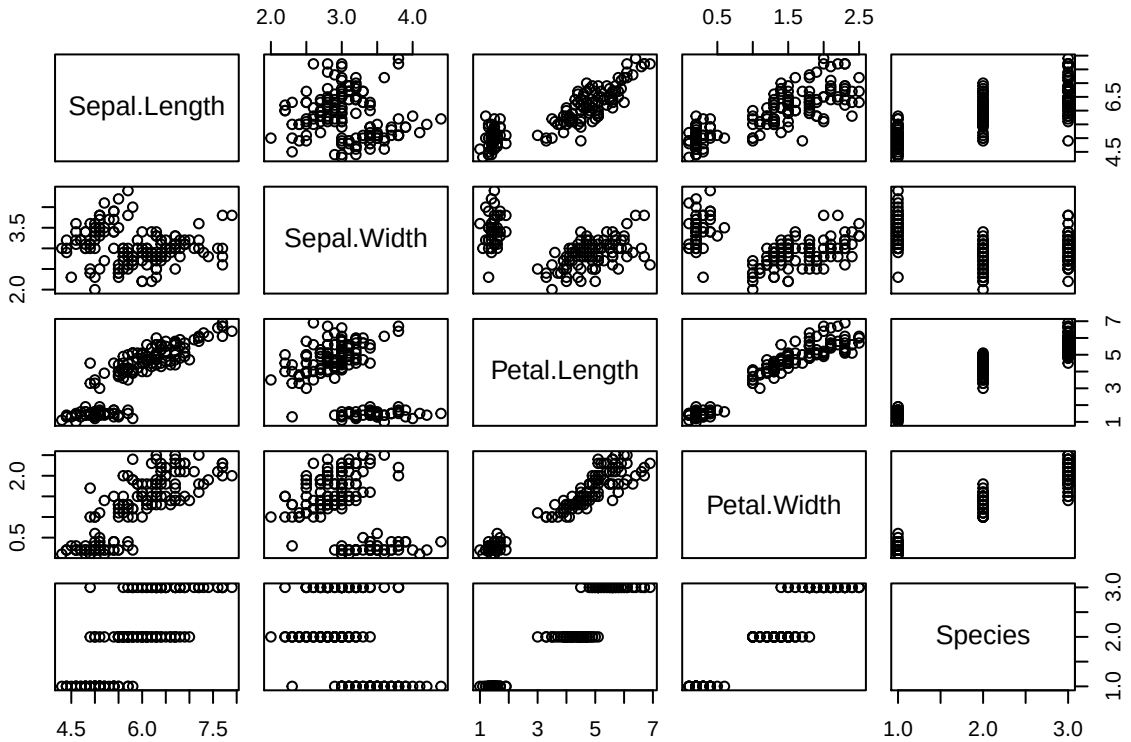
```
head(iris) # First six rows of a data table
```

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

```
tail(iris) # Last six rows of a data table
```

```
##     Sepal.Length Sepal.Width Petal.Length Petal.Width   Species
## 145          6.7         3.3          5.7         2.5 virginica
## 146          6.7         3.0          5.2         2.3 virginica
## 147          6.3         2.5          5.0         1.9 virginica
## 148          6.5         3.0          5.2         2.0 virginica
## 149          6.2         3.4          5.4         2.3 virginica
## 150          5.9         3.0          5.1         1.8 virginica
```

```
pairs(iris)
```



Seminar 1 Assignment

As you know, we will have a seminar every week. We will give you an assignment to complete in class. Ideally, once you are comfortable with software, you should be able to complete the assignment in class and turn it in before you leave the seminar. However, we realize that we all work at different pace, and not everyone will work fast enough to complete the assignment in the time allowed. Therefore, you can take incomplete work home and submit it by midnight the day before the next class. We will create a folder in LMS for each submission; as stated in the syllabus, late work is **not accepted for any reason.**

Here is the assignment:

1. Create a new R Markdown document or by opening the template we prepared for you. Go to File -> New File -> R Markdown.... or File -> Open -> M Markdown Template. In "Title" please put "Seminar 1," in "Author" - your name and group number. Select PDF as a default output format. When Markdown file opens, you will see the **Knit** button - use it often to make sure your document looks just the way you want it to look.
2. Embed and execute the code below; after each line, briefly describe what each line does (you will need to complete your Swirl lessons for that). In other words, tell me what is x, A1, A, B, all Xs, d, s and Y.
3. Please submit both the .Rmd and the PDF files of your work. Full credit will not be given if one of the files is missing.

```
x <- c(3,2,5)
A1 <- matrix(c(1,3,5,2,4,6),2,3)
A <- matrix(c(1,4,2,5,3,6),2,3)
B <- matrix(c(4,1,5,2,6,3),2,3)
X1<-A%*%t(B)
X2<-B%*%t(A)
X3<-t(A)%*%B
X4<-t(B)%*%A
d<-diag(A%*%t(B))
s<-sum(diag(A%*%t(B)))
Y<-solve(A%*%t(B))
```

If you have any questions, please remember to utilize the LMS Forum. We will give extra-credit points to students who actively answer questions on the Forum.

Homework 1 Assignment

Homeworks are designed to take more time than seminars; they are given for you to think about during the week and spend a little more time than is allotted for the seminar. Because all of you should be familiar with regression analysis, this assignment allows you to practice your R skills with a basic univariate regression.

Here is the assignment:

1. Create a new R Markdown document or by opening the template we prepared for you. Go to File -> New File -> R Markdown.... or File -> Open -> M Markdown Template. In "Title" please put "Homework 1," in "Author" - your name and group number. Select PDF as a default output format.
2. Data is in the file carprise.csv that we uploaded for you on LMS. Read the data into R by using the read.csv command. Hint: to simplify the task, save the data file in your working directory, or change your working directory to where you store the file.
3. Create a vector x, which contains second column of the data matrix.
4. Create a vector y, which contains the fourth column of the data matrix.
5. Estimate a linear model by regressing y on x (hint: command is lm(formula = y ~ x))
6. Generate a summary of your linear model (command summary(model_name), where model_name is any name you've given your model (you have to use the name you have assigned your command to).
7. For the same model, create ANOVA output.
8. Run the following code:

```
n <- 11

X<- cbind(1,x)

H <- X%*%solve(t(X)%*%X)%*%t(X)

J <- matrix(1,n,n)

In <- diag(n)

SStotal <- t(y)%*%(In-1/n*J)%*%y

SSreg <- t(y)%*%(H-1/n*J)%*%y

SSres <- t(y)%*%(In-H)%*%y

SSreg;SSres
```

What do you observe? Explain the numbers you have just generated by comparing them to the ANOVA output.

Please submit both the .Rmd and the PDF files of your work. Full credit will not be given if one of the files is missing.