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

Introduction to Network Analysis, Spring 2019

Seminar 1 and Homework 1 Answers

Dmitry Zaytsev, PhD, Valentina Kuskova, PhD

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

Seminar 1 Answers

Your assignment questions are below; I provide answers as comments in the document or the R code, as appropriate.

---

1. Create a new R Markdown document or by opening the template I 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) # This is a column vector
```

As you know, we have column and row vectors. Row vectors have to be created from column vectors. Compare the two:

```
x <- c(3,2,5)
x
```

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

```
x <- t(x)
x
```

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

```
# Or it is the same as:
x <- t(c(3,2,5))
x
```

```
##      [,1] [,2] [,3]
```

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

As you can see, when you get a row vector, you also get indicators for rows (in this case, 1 row) and columns (in this case, three columns).

The three structures below are matrices. You probably noticed that there are two parentheses for each matrix. First parentheses contains the matrix data, that is, values from 1-6. However, notice how the matrices differ, depending on the order in which the numbers are given.

```
A1 <- matrix(c(1,3,5,2,4,6),2,3)
A1
```

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

```
A <- matrix(c(1,4,2,5,3,6),2,3)
A
```

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

```
B <- matrix(c(4,1,5,2,6,3),2,3)
B
```

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

If you paid attention, then you noticed that just as with the vector, the program "binds" each column vector first: [1,3] for first column of vector A1; [5,2] for second column, etc.

But how does the program know where the vector stops? The length of the vector is given by the dimension in the second parentheses, after the data. In our case, each matrix has 2 rows and 3 columns, and these are the numbers 2 and 3 in the structure matrix(c(),2,3).

Therefore, we conclude that matrix is given by the following formula:

```
## matrix(c(data),Nrows,Ncolumns)
```

Of course, this is why we take [x,] when we need a row of the matrix and [,x] - when we need a column.

If you were to get adventurous, you could always try # reversing the order of rows and columns to see what happens:

```
A1 <- matrix(c(1,3,5,2,4,6),3,2)
A1 # See how this matrix is different from the first A1?
```

```
##      [,1] [,2]
## [1,]    1    2
```

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

Of course, command %*% for matrices X1-X4 below is matrix multiplication, and t(A) is the transpose of a matrix (meaning, columns become rows and rows become columns). Transpose of a matrix is also indicated by what is called a *prime* - single quotation mark, t(B) = $\mathbf{B}'$.

```
X1<-A%*%t(B)
X1
```

```
##      [,1] [,2]
## [1,]   32   14
## [2,]   77   32
```

```
X2<-B%*%t(A)
X2
```

```
##      [,1] [,2]
## [1,]   32   77
## [2,]   14   32
```

```
X3<-t(A)%*%B
X3
```

```
##      [,1] [,2] [,3]
## [1,]    8   13   18
## [2,]   13   20   27
## [3,]   18   27   36
```

```
X4<-t(B)%*%A
X4
```

```
##      [,1] [,2] [,3]
## [1,]    8   13   18
## [2,]   13   20   27
## [3,]   18   27   36
```

You were supposed to notice that matrices X1 and X2 are NOT indentical, but matrices X3 and X4 are. Why, if each contains identical numbers?

Matrix multiplication is a complex operation. The order and dimensions of the matrices matter. If you recall, when multiplying matrices, matrix dimensions have to be (a,b)*(b,a), so the number of rows in the first matrix has to be equal to the number of columns in the second. This is because matrix multiplication is not a simple multiplication command, and moreover, almost always $\mathbf{A} * \mathbf{B} \neq \mathbf{B} * \mathbf{A}$.

So let's look again at our matrices and how it is done. If you have never done matrix multiplication by hand, look carefully at what I do to figure out the process.

$$\mathbf{A} = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}, \qquad \mathbf{B} = \begin{pmatrix} 4 & 5 & 6 \\ 1 & 2 & 3 \end{pmatrix}$$

First, transpose of the **A** and **B** matrices are defined as follows:

$$\mathbf{A}' = \begin{pmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{pmatrix}, \qquad \mathbf{B}' = \begin{pmatrix} 4 & 1 \\ 5 & 2 \\ 6 & 3 \end{pmatrix}$$

Then, $\mathbf{AB}'$ is calculated as follows:

$$\mathbf{AB}' = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix} \begin{pmatrix} 4 & 1 \\ 5 & 2 \\ 6 & 3 \end{pmatrix} = \begin{pmatrix} (4+10+18) & (1+4+9) \\ (16+25+36) & (4+10+18) \end{pmatrix} = \begin{pmatrix} 32 & 14 \\ 77 & 32 \end{pmatrix}$$

$\mathbf{BA}'$ is calculated as follows:

$$\mathbf{BA}' = \begin{pmatrix} 4 & 5 & 6 \\ 1 & 2 & 3 \end{pmatrix} \begin{pmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{pmatrix} = \begin{pmatrix} (4+10+18) & (16+25+36) \\ (1+4+9) & (4+10+18) \end{pmatrix} = \begin{pmatrix} 32 & 77 \\ 14 & 32 \end{pmatrix}$$

$\mathbf{A}'\mathbf{B}$ is calculated as follows:

$$\mathbf{A}'\mathbf{B} = \begin{pmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{pmatrix} \begin{pmatrix} 4 & 5 & 6 \\ 1 & 2 & 3 \end{pmatrix} = \begin{pmatrix} (4+4) & (5+8) & (6+12) \\ (8+5) & (10+10) & (12+15) \\ (12+6) & (15+12) & (18+18) \end{pmatrix} = \begin{pmatrix} 8 & 13 & 18 \\ 13 & 20 & 27 \\ 18 & 27 & 36 \end{pmatrix}$$

$\mathbf{B}'\mathbf{A}$ is calculated as follows:

$$\mathbf{B}'\mathbf{A} = \begin{pmatrix} 4 & 1 \\ 5 & 2 \\ 6 & 3 \end{pmatrix} \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix} = \begin{pmatrix} (4+4) & (8+5) & (12+6) \\ (5+8) & (10+10) & (15+12) \\ (6+12) & (12+15) & (18+18) \end{pmatrix} = \begin{pmatrix} 8 & 13 & 18 \\ 13 & 20 & 27 \\ 18 & 27 & 36 \end{pmatrix}$$

The next two tasks are straightforward. We have the "diag" command, which takes the diagonal elements of a matrix (and returns d as a vector of these diagnonal elements), then - the command that sums them up.

```
d<-diag(A%*%t(B))
d
```

```
## [1] 32 32
```

```
s<-sum(diag(A%*%t(B)))
s
```

```
## [1] 64
```

But what did we get as a result? We got what we call a *trace* of a matrix. Let's look at the math:

A trace of a matrix is the sum of its diagonal elements. Therefore,

$\text{tr}(\mathbf{AB}') = 32 + 32 = 64$

$\text{tr}(\mathbf{BA}') = 32 + 32 = 64$

$\text{tr}(\mathbf{A}'\mathbf{B}) = 8 + 20 + 36 = 64$

$\text{tr}(\mathbf{B}'\mathbf{A}) = 8 + 20 + 36 = 64$

So, though matrix multiplication results were different, traces of matrices are identical.

The final task was not straightforward at all. You had to look up the "solve" command to realize what it means. Those of you who did got full credit.

Solve command performs a mathematical calculation that finds such matrix (or vector) $\mathbf{Y}(\mathbf{y})$ that makes the following equation true: $\mathbf{A} * \mathbf{Y} = \mathbf{B}$, or in our case, $\mathbf{A} * \mathbf{Y} = \mathbf{B}'$. So the matrix Y we find is such a matrix, which, when $\mathbf{A}$ is multiplied by it, gives us the transpose of matrix B, $\mathbf{B}'$.

```
Y<-solve(A%*%t(B))
Y
```

```
##             [,1]       [,2]
## [1,] -0.5925926  0.2592593
## [2,]  1.4259259 -0.5925926
```

You get full credit for your answers if you have taken the time to figure out what each of the commands truly means.

Homework 1 Solutions

Again, just as with Seminar 1, I duplicate the assignment and provide the answers in teh R code.

1. Create a new R Markdown document or by opening the template I 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 I 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.

```
# Here, we just execute the read.csv command:
datain<-read.csv('carprice.csv') # I call the file "datain"
datain # check what it looks like
```

```
##    Car Age Miles Price
## 1    1   5    57    85
## 2    2   4    40   103
## 3    3   6    77    70
## 4    4   5    60    82
## 5    5   5    49    89
## 6    6   5    47    98
## 7    7   6    58    66
## 8    8   6    39    95
## 9    9   2     8   169
## 10  10   7    69    70
## 11  11   7    89    48
```

3. Create a vector x, which contains second column of the data matrix.

```
# We simply take the second column of the data matrix.
# It means in the square brackets, we want values after the comma (meaning, in a colum
# and we want values that are in column 2
x<-datain[,2]
x # look at our vector, it's the Age variable
```

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

4. Create a vector y, which contains the fourth column of the data matrix.

```
# Same code as above, but column 4
y<-datain[,4]
y # look at our vector, it's the Price variable
```

```
##  [1]  85 103  70  82  89  98  66  95 169  70  48
```

5. Estimate a linear model by regressing y on x (hint: command is lm(formula = y ~ x))

```r
# Since I provided the command, it should be easy.
# However, I also assign the model to an object called lmout:
lmout<-lm(formula=y~x)
lmout
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Coefficients:
## (Intercept)            x
##      195.47       -20.26
```

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).

```r
# Command is simply summary:
summary(lmout)
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -12.162  -8.531  -5.162   8.946  21.099
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   195.47      15.24  12.826 4.36e-07 ***
## x             -20.26       2.80  -7.237 4.88e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 12.58 on 9 degrees of freedom
## Multiple R-squared:  0.8534, Adjusted R-squared:  0.8371
## F-statistic: 52.38 on 1 and 9 DF,  p-value: 4.882e-05
```

8. For the same model, create ANOVA output.

```r
# Command is simply anova:
anova(lmout)
```

```
## Analysis of Variance Table
##
## Response: y
```

```
##           Df Sum Sq Mean Sq F value    Pr(>F)
## x          1 8285.0  8285.0   52.38 4.882e-05 ***
## Residuals  9 1423.5   158.2
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

9. Run the following code:

```
# We create a variable n=11 - if you noticed, this is the number of observations
n<-11
X<-cbind(1,x)
X
```

```
##         x
##  [1,] 1 5
##  [2,] 1 4
##  [3,] 1 6
##  [4,] 1 5
##  [5,] 1 5
##  [6,] 1 5
##  [7,] 1 6
##  [8,] 1 6
##  [9,] 1 2
## [10,] 1 7
## [11,] 1 7
```

$\mathbf{X}$ is a 2-column matrix, where the first column is simply a column of 1s. Why? This is where your intercept of the regression equation will come from.

Next, we calculate what is known to be a "hat" matrix, or the $\mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'$ matrix. We are not going to get into much math here, but when this $\mathbf{H}$ matrix is multiplied by the vector of dependent variable, $\mathbf{y}$, it will give us a vector of regression coefficients.

```
H<-X%*%solve(t(X)%*%X)%*%t(X) # "Hat" matrix
J<-matrix(1,n,n) # a nxn matrix of 1s
J
```

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

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

```
In<-diag(n) # this is a matrix that contains 1s in the diagonal elements
# 0s everywhere else
In
```

```
##       [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11]
##  [1,]    1    0    0    0    0    0    0    0    0     0     0
##  [2,]    0    1    0    0    0    0    0    0    0     0     0
##  [3,]    0    0    1    0    0    0    0    0    0     0     0
##  [4,]    0    0    0    1    0    0    0    0    0     0     0
##  [5,]    0    0    0    0    1    0    0    0    0     0     0
##  [6,]    0    0    0    0    0    1    0    0    0     0     0
##  [7,]    0    0    0    0    0    0    1    0    0     0     0
##  [8,]    0    0    0    0    0    0    0    1    0     0     0
##  [9,]    0    0    0    0    0    0    0    0    1     0     0
## [10,]    0    0    0    0    0    0    0    0    0     1     0
## [11,]    0    0    0    0    0    0    0    0    0     0     1
```

Well, we are ready to calculate the Sums of Squares for our regression. If you are adventurous enough, you can look at the formulas to see how we arrived at this calculation from the formula for the SStotal:

```
SStotal<-t(y)%*%(In-1/n*J)%*%y # Total sum of squares
SSreg<-t(y)%*%(H-1/n*J)%*%y # Regression sum of squares
SSres<-t(y)%*%(In-H)%*%y # Residual sum of squares
SSreg; SSres
```

```
##          [,1]
## [1,] 8285.014
```

```
##          [,1]
## [1,] 1423.532
```

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

Well, we observe that we can generate enough code to calculate regression by hand! The Regression and Residual Sum of Squares are identical (in fact, more accurate) to the numbers that are generated by the R-driven regression model code.

Grading

I will give you full credit if you have taken time to figure out the meaning of assignment commands, matrix operation commands, and the "solve" command. However, I did not expect you to get too involved in math here, so my grading on the assignment is rather generous.