Oxford Q-Step Centre Data Day: Introduction to R

Claire Peacock

April 1, 2017

In this lab we will be getting to know R using the EU referendum data and text data on referendum-related speeches by the party leaders. You will get the chance to:

- see how R works (using a program called RStudio),
- make your own plots,
- and more!

For the lab today, **we will work in partners (or small groups of 3)**. You can choose to all run the code on your own computers or all gather around one person's computer.

0.1 Some basic information

- A **script** is a text file in which you write your R commands (code) and comments.
- Remember to **save your script** or send it to yourself at the end of the lab session so you can use it at home. To do this click *File –> Save as*. This makes a .R file which can be opened in Rstudio at home or in a text editor.
- If you put the **# character** in your command, anything in the same line following the # will not be executed; this is useful to **add comments** to your script!
- R is **case sensitive**, so be careful when typing.
- To **run your commands**, highlight the relevant line of code in your script and click on **Run**, or select the line and hit *ctrl + enter* (*cmd + enter* on Mac).
- By pressing the **up key**, you can go back to the commands you have used before.
- Press the **tab key** to auto-complete variable names and commands.

0.2 Getting Started in the Q-Step Lab

Begin by opening RStudio (located on the desktop).

Your first task is to create a new script (this is where we will write our commands). To do so, click *File –> New File –> R Script*.

Your screen should now have four panes:

- the **Source Editor** (top left)
- the **Console** (bottom left)
- the **Environment/History** (top right),
- and **Files/Plots/Packages/Help/Viewer** (bottom right).

Now you are now ready to get started!

0.3 A simple script

The *Source Editor* (top left) is where we write our commands for R. Let's try writing your first command. Type the following command into your script (editing the message as you like)!

```
x <- "My message goes here"
```

To tell R to run the command, highlight the relevant row in your script and click the **Run** button (top right of the *Source Editor*) or hold down *ctrl* + *enter* (PC) or *cmd* + *enter* (Mac).

Running the command above creates an object named 'x', now visible in the *Environment* (top right) that contains the words of your message. **It is good practice to name your objects** so you can later refer back to them!

You can now see 'x' in the *Environment* (top right). To view what is contained in x, in the *Console* (bottom left) type:

```
x
```

- After pressing *enter*, what happens?
- What happens when you run the previous command with an uppercase 'X'?

Now you are ready to load the data we will need in the lab.

1 Part 1: The Demographics of Brexit

1.1 Loading Data into R

To load the data we will be using, run the following command. To do so, either type the command into your script and run it from there or paste it directly into the *Console* and press *enter*. From this point onwards, you can choose how to run your commands (i.e. directly in the *Console* or in your script).

```
#TO DO: ADJUST AS NEEDED FOR THE LAB!
Brexit_data <- read.csv("/Users/clairepeacock/Documents/Q_step/Access_labs/data/brexit_lab_vars.csv", s
```

Now we have loaded our data and named it "Brexit_data". To view the data, click on it in the *Environment* or run the following command:

```
View(Brexit_data)
```

In the data viewer, you can click on the row names of the columns to sort the data. Test it out!

1.2 Working with the EU referendum data

Now it's time to get to know our data. The EU referendum data comes directly from the Electoral Commission. All remaining variables are from the 2011 UK Census (except for the 'earnings' variables which are from the 2016 Annual Survey of Hours and Earnings).

While we are going to look at correlations in this lab between demographic features and voting patterns, it is important to recognize the limitations of (and possibilities associated with!) the data we rely on. Before continuing, take a moment to think about the following questions with your partner.

- If you had voted in the EU referendum, what factors would have influenced your decision?
- Are the factors that would have influenced you possible to measure? Why or why not? Are they included in the data we have provided?

1.3 Plotting

Let's begin our analysis by making some plots!

Age was a key explanation discussed in the media of the referendum results. In order to analyze for ourselves the relationship between age and EU referendum voting, we can make a plot of the percent of voters who voted "remain" against the mean age in each area. To do so run the following commands:

Hint: the command is set up as follows:

name_of_plot <- ggplot(name_of_data, aes(x=name_of_x, y=name_of_y)) + add some specifications ...

- *aes* stands for aesthetics and is used to specify what we actually want to see in the plot

```
library(ggplot2) # loads the plotting library we need
remain_age <- ggplot(Brexit_data, # make a plot with the ggplot command
                     # using data from Brexit_data & name it 'remain_age'
              aes(x=age_mean, y=Percent_Remain)) + # pick x and y variables
              geom_point(shape=1) # shape=1 --> use hollow circles for dots
remain_age # View the plot
```

- This creates a (very!) basic scatterplot of our two variables.
- What does our plot tell us about the relationship between the two variables? Hint: What do the axes represent?
- How many conclusions can we draw from this plot?
- What should we be aware of when using this data?

When we create scatterplots to assess correlation, it is important that we ask ourselves why the variables we look at might be related to eachother.

- In this case, why might mean age affect voting habits in the EU referendum? What other explanations might do better?

Optional: You can also try running this plot using the "Percent_Leave" data instead. To do so, you can copy the command above but change the relevant variable. Name this new plot **leave_age** (Hint: remember to use the "<-" symbol to name your object).

1.4 Editing a plot

If we want to add more information to our plot, we can do many things. For example, we could:

- rename the axes,
- insert labels,
- change the size of the dots,
- adjust the axes,
- change the background to be black and white,
- add a line of best fit.

```
remain_age <- remain_age + # Tell R to keep our plot but to add some more detail!
  xlab("Age (average)") + # Label x axis
  ylab("Remain votes (%)") +    # Label y axis
  theme_bw() + # Remove gray background (i.e. make black and white)
  geom_point(aes(size = Residents_total, colour=Region)) +
  # change size of dots to population size
```

```
  # change colour to match regions
  geom_smooth(method=lm,     # Add line of best fit
              se=FALSE) +    # Don't add shaded confidence region
scale_x_continuous(limits = c(30, 50)) + # sets x-axis scale
scale_y_continuous(limits = c(20, 80)) # sets y-axis scale

remain_age # View the plot
```

Now we have a lot more information (in fact this figure probably shows too much information)!

- What does the size of the dots represent? What do the colours represent? Are they useful?
- Is the relationship clearer now with the line of best fit? What direction is the line of best fit?

Let's try to make our plot a bit more readable by choosing just a few features to include in preparation for making an interactive plot:

```
remain_age2 <- ggplot(Brexit_data, # make a plot using data from Brexit_data
              # & name it 'remain_age'
              aes(x=age_mean, y=Percent_Remain, label=Area)) + # pick x and y
              # variables & labels
              # note the labels won't show up until the interactive plot
              # in the next command
              theme_bw() + # Remove gray background (i.e. make black and white)
              geom_point(shape=1) + # Use hollow circles for dots with shape=1
              scale_x_continuous(limits = c(30, 50)) + # sets x-axis scale
              scale_y_continuous(limits = c(20, 80)) + # sets y-axis scale
              geom_smooth(method=lm,     # Add line of best fit
              se=FALSE) +    # Don't add shaded confidence region
              xlab("Remain votes (%)") + # Label x axis
              ylab("Age (average)")    # Label y axis

remain_age2 # View the plot
```

1.5 Making our plots interactive

If we want to get more from our plots, we can make them interactive. For our scatterplots, this means that we can hover over parts of our plot to get more details. Try running the following commands to make an interactive scatterplot.

```
library(plotly) # loads the library we need to make our plots interactive
remain_age2 <- ggplotly(remain_age2) # Make it interactive! Now the transparent
# labels from above show up!
remain_age2 # Now view it again and notice the difference.
```

- Hover your mouse over the different points on the plot. What do you notice?
- Try zooming in and out and looking at the other options in the menu at the top of the plot.
- Can you find your home area?
- Are any of the points suprising?

Now that we have our basic interactive plot, we can also go a bit further. Let's check to see if there is any regional variation of interest. This time we can run all our commands together (i.e. make a plot with a couple of specifications and then make it interactive in one go)!

```
remain_age_region <- ggplot(Brexit_data,
                            aes(x=age_mean, y=Percent_Remain, colour=Region)) +
  theme_bw() + # Remove gray background (i.e. make black & white)
  geom_point() +    # Use filled in circles by not selecting a shape
  geom_smooth(method=lm,   # Add line of best fit
              se=FALSE) +   # Don't add shaded confidence region
scale_y_continuous(limits = c(20, 80)) # sets x axis
remain_age_region <- ggplotly(remain_age_region) # Make the plot interactive
remain_age_region # View the plot
```

- Can you go back and label the axes on this plot using the commands you learned above?
- Does this plot give you any more useful information than the previous ones?
- Can this plot tell us anything about regional variation in the relationship between age and voting 'remain' in the referendum? (Hint: look at the slope of the lines, also try clicking on the coloured dots in the legend to include and remove certain data)

1.6 On your own

Now it's your turn to use the data to inspect some correlations between referendum voting and the demographics of an area!

- With your partner, try out some of the code given to you above but with different variables alongside the voting data to see what you can find. At the end of the lab, you will have the opportunity to share some of your results.

2 Part 2: Brexit in words

One aspect of understanding the results of the EU referendum is looking at the correlations between voting for Brexit and demographics in each area. However, a key battle of the Brexit campaign waged with words through social media, speeches, and in the newspapers. This type of text data can also be analysed!

One way to assess this text data is to use some of the basic tools of text analysis.

For this activity, we will load and analyze three texts from June 21, 2016 - two days before the final vote.

- The first is a speech made by then Prime Minister David Cameron of the Conservative Party.
- The second is an article written by UKIP's Nigel Farage.
- The third is a speech made by Labour leader Jeremy Corbyn.

Together, these three texts form what is referred to as a *corpus*. A corpus is simply a grouping of texts.

2.1 Loading the text data

Run the following command to load the text data:

```
# TO DO: ADJUST FILEPATHS AS NECESSARY!
library(quanteda) # The library with the tools we need!
# Load the files
speech_files <- textfile("/Users/clairepeacock/Documents/Q_step/Access_labs/Data/*.txt",
                         cache = FALSE)
```

Next we can have a look at what we have loaded. Let's start with the first file.

```
summary(corpus(speech_files), 1) # shows overview of first speech
texts(speech_files)[1] # shows text of 1st speech
```

In the output, note that *Types* tells you how many unique words there are and *Tokens* tells you how many words there are in the given text. The n's you see in the print out are simply placemarkers indicating a new paragraph in the original text formatting.

Now check the next two files.

- What do you need to change in the commands above to look at the next files?

2.2 Formatting the text data

Now we need to do a bit of final formatting before we analyze the data. First we need to tell R how to format our files.

```
speech_corpus <- corpus(speech_files) # Tell R to make the files into the corpus format
# (alltogether as one object)
```

Now, we can make a table counting the occurence of words in each speech.

```
speech_table <- dfm(speech_corpus, ignoredFeatures=stopwords("SMART"),
                    stem = FALSE, removeTwitter=TRUE)
# SMART means change everything --> lowercase, remove punctuation & numbers & whitespace
# removeTwitter means @ & # are taken out.
```

To view the top 10 most occuring words in the table, run the following command:

```
topfeatures(speech_table, 10) # top 10 most-occuring words/features
# of all texts taken together
```

What if we want to view each text individually? Re-run the previous 3 command chunks for each speech using the following commands:

```
cameron_corpus <- corpus(speech_files)[1] # Make a corpus for each individual text
corbyn_corpus <- corpus(speech_files)[2]
farage_corpus <- corpus(speech_files)[3]
# Now make a table for each text:
cameron_table <- dfm(cameron_corpus, ignoredFeatures=stopwords("SMART"),
                    stem = FALSE, removeTwitter=TRUE)
corbyn_table <- dfm(corbyn_corpus, ignoredFeatures=stopwords("SMART"),
                    stem = FALSE, removeTwitter=TRUE)
farage_table <- dfm(farage_corpus, ignoredFeatures=stopwords("SMART"),
                    stem = FALSE, removeTwitter=TRUE)
topfeatures(cameron_table, 10) # top 10 most-occuring words/features
topfeatures(farage_table, 10) # top 10 most-occuring words/features
topfeatures(corbyn_table, 10) # top 10 most-occuring words/features
```

- How do the tables of top 10 words compare in each text (the top row is Cameron, the middle is Farage, and the final row is Corbyn)?
- Are you suprised by anything?

2.3 Making word clouds

In our last step, we can visualize the texts using a **word cloud**. Hint: In order for the plot to work properly make sure you make your **Plot** pane large by dragging the left side across your screen.

```
plot(cameron_table, comparison=FALSE, max.words=30)
```

```
plot(farage_table, comparison=FALSE, max.words=30)
```

```
plot(corbyn_table, comparison=FALSE, max.words=30)
```

- What do the plots tell us about the word choice of each politician?
- What differences are there between the plots?

2.4 Add-on activity at home (or in the lab if you have time!)

The same principles that we applied above (i.e. loading a .txt file into R and analyze it using basic text descriptives) can be used on other types of text data.

For example, you can easily analyze Twitter feeds using R by copying the ones you are interested in into a .txt file. If you wanted to go further, you could even load your own entire Twitter archive or excerpts from others' and analyze them using R (try advanced this tutorial).

- For this activity, choose another source of text that interests you and load it into R to analyze using the commands you learned above (and you can share it with the group if you would like)!

3 Extra information

3.1 Getting Started with R at Home

If you are on a computer at home, you can load R to continue working on some of the examples from the lab and try out some new ideas.

- **PC**: you will need to load R and the desktop version of RStudio.
- **Mac**: you will need R and RStudio but also XQuartz.
- All three programs are free. Make sure to load everything listed above for your operating system or R will not work properly!

Once you have loaded the programs above, you can open and use RStudio (you do not need to separately open XQuartz or R).

3.2 Knowing where R saves your documents

If you are at home, when you open a new script make sure to check and set your working directory (i.e. the folder where the files you create will be saved).

- To check your working directory use the getwd() command (type it into the Console or write it in your script in the Source Editor):

```
getwd()
```

- To set your working directory, run the following command, substituting the file directory of your choice. Remember that anything following the '#' symbol is simply a clarifying comment and R will not process it.

```
## Example for Mac (remove the 1st '#' symbol to run this command)
# setwd("/Users/Documents/mydir/")
```

```
## Example for PC (remove the 1st '#' symbol to run this command)
# setwd("c:/docs/mydir")
```

4 Variable Cheatsheet

GENERAL:

- [1] **Area** *Voting Area*
- [2] **Region_Code**
- [3] **Region**
- [4] **Area_Code**
- [5] **Electorate** *Number of Voters in the Area*

EU Referendum Data:

- [6] **ExpectedBallots** *Expected ballots for referendum*
- [7] **VerifiedBallotPapers** *Verified ballot papers for referendum*
- [8] **Percent_Turnout** *Percent turnout in referendum*
- [9] **Votes_Cast** *Total votes cast in referendum*
- [10] **Valid_Votes** *These are the votes that actually counted (i.e. they were not rejected) in the referendum*

- [11] **Remain** *Number of votes for remain in referendum*
- [12] **Leave** *Number of votes for leave in referendum*
- [13] **Percent_Remain** *Percent of total votes for remain in referendum*
- [14] **Percent_Leave** *Percent of total votes for leave in referendum*
- [15] **Rejected_Ballots**
- [16] **No_official_mark** *Reason for rejecting referendum ballot*
- [17] **Voting_for_both_answers** *Reason for rejecting referendum ballot*
- [18] **Writing_or_mark** *Reason for rejecting referendum ballot*
- [19] **Unmarked_or_void** *Reason for rejecting referendum ballot*
- [20] **Percent_Rejected** *Percent of referendum ballots rejected*

CENSUS DATA 2011/ANNUAL SURVEY OF HOURS AND EARNINGS 2016

- [21] **Residents_total** *Number of residents*
- [22] **Population_Density** *Population density*
- [23] **Economically_active_percent** *Percent of residents who are economically active*
- [24] **Employed_of_economically_active_percent** *Percent of economically active residents who are employed*
- [25] **Unemployed_Age50_74_percent** *Percent of residents unemployed between ages 50 and 74*
- [26] **Health_very_good** *Self-reported general health - number*

- [27] **Health_very_good_percent** *Self-reported general health - percent*
- [28] **Health_good** *Self-reported general health - number*
- [29] **Health_good_percent** *Self-reported general health - percent*
- [30] **Health_fair** *Self-reported general health - number*
- [31] **Health_fair_percent** *Self-reported general health - percent*
- [32] **Health_bad** *Self-reported general health - number*
- [33] **Health_bad_percent** *Self-reported general health - percent*
- [34] **Health_very_bad** *Self-reported general health - number*
- [35] **Health_very_bad_percent** *Self-reported general health - percent*
- [36] **Single_never_married_percent** *Percent of all usual residents aged 16 and over who are single and have never married*
- [37] **Married_percent** *Percent of all usual residents aged 16 and over who are married*
- [38] **Occup_high_manage_admin_profess_percent** *Occupation of usual residents aged 16-74: percent in high managerial, administrative, or professional positions*
- [39] **Occup_low_manage_admin_profess_percent** *Occupation of usual residents aged 16-74: percent in low managerial, administrative, or professional positions*
- [40] **Occup_intermediate_percent** *Occupation of usual residents aged 16-74: percent in intermediate jobs*
- [41] **Occup_small_employer_percent** *Occupation of usual residents aged 16-74: percent employed by a small employer*
- [42] **Occup_low_supervis_technical_percent** *Occupation of usual residents aged 16-74: percent in low supervisory or technical jobs*
- [43] **Occup_semi_routine_percent** *Occupation of usual residents aged 16-74: percent in semi routine occupations*
- [44] **Occup_routine_percent** *Occupation of usual residents aged 16-74: percent in routine occupations*
- [45] **Earnings_Median** *Median value of gross pay for full time workers, before tax, National Insurance or other deductions*
- [46] **Earnings_Mean** *Mean value of gross pay for full time workers, before tax, National Insurance or other deductions*
- [47] **Bachelors_deg_percent** *Percent of all usual residents aged 16 and over with a bachelors degree*
- [48] **age_mean** *Mean age of usual residents*
- [49] **age_median** *Median age of usual residents*
- [50] **Birth_UK** *Country of birth of usual residents - number born in UK*
- [51] **Birth_UK_percent** *Country of birth of usual residents - percent born in UK*
- [52] **Birth_other_EU** *Country of birth of usual residents - number born in other EU country*
- [53] **Birth_other_EU_percent** *Country of birth of usual residents - percent born in other EU country*
- [54] **Birth_Africa** *Country of birth of usual residents - number born in other Africa*

- [55] **Birth_Africa_percent** *Country of birth of usual residents - percent born in other Africa*
- [56] **Birth_MidEast_Asia** *Country of birth of usual residents - number born in other Middle East or Asia*
- [57] **Birth_MidEast_Asia_percent** *Country of birth of usual residents - percent born in other Middle East or Asia*
- [58] **Birth_Americas_Carrib** *Country of birth of usual residents - number born in other Americas or Carribean*
- [59] **Birth_Americas_Carrib_percent** *Country of birth of usual residents - percent born in other Americas or Carribean*
- [60] **Birth_Antarctica_Oceania_Other** *Country of birth of usual residents - number born in other Antarctica, Oceania, or other*
- [61] **Birth_Antarctica_Oceania_Other_percent** *Country of birth of usual residents - percent born in other Antarctica, Oceania, or other*