


# Data Analytics for Social Science
# Transforming and cleaning data

Johan A. Elkink

School of Politics & International Relations
University College Dublin

7 February 2017



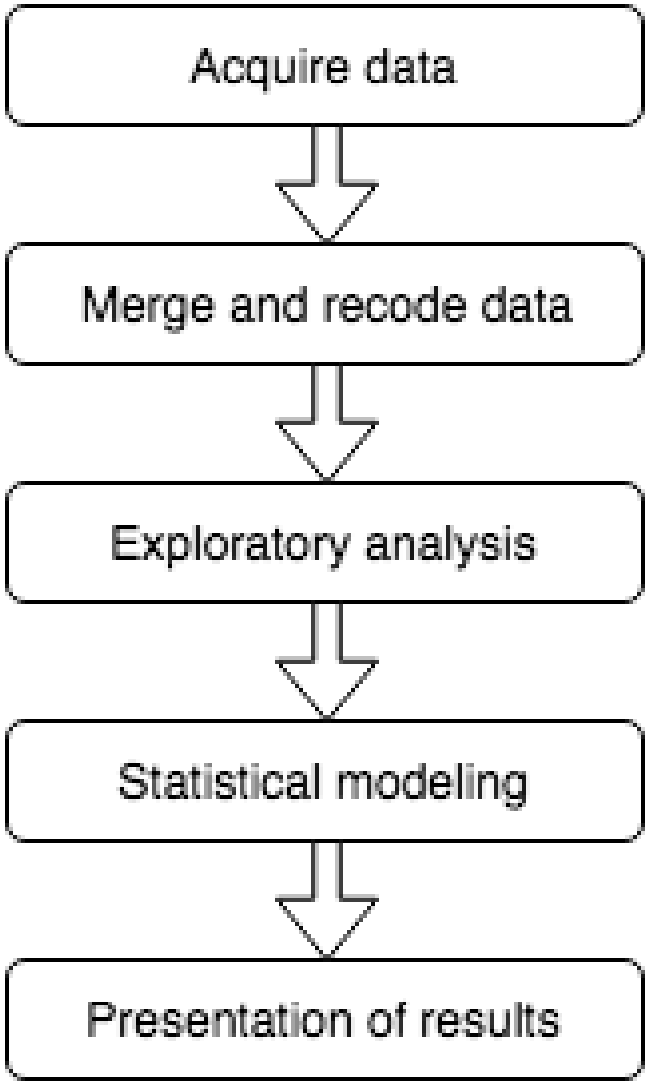

# Recap

R at the cross-section of social science analysis and data science

R can be extended using packages (see tutorial video)

Data to be found in many formats

Typically most attention to statistical modeling and interpretation, but good **data preparation** is essential ... and hard.






# Outline




# Basic computations

R can be used as a basic calculator, or variables can be created using computations.

```
5 * 3 / (3 + 1)
age <- 2017 - birthyear
age2 <- age ^ 2
```

Here, we calculate $\frac{5 \cdot 3}{3+1}$ and create a variable age by subtracting the year of birth from the current year (i.e. year of the survey) and a variable age2 by taking the square of age.




# Transformations

Similar computations can also involve other functions, such as a logarithmic transformation:

```
logGDP <- log(GDP, base = 10)
```

We can also create our own functions, for example, a stddize() function that calculates the standardized value of a variable:

```
stdize <- function(x) { (x - mean(x)) / sd(x) }
zGDP <- stdize(GDP)
zAge <- stdize(age)
```

(That function really should deal with missing values better, though.)




# Comparison operators

We often need to select cases, or recode variables, based on ranges of data. We can use comparison operators for this.

| | |
|---|---|
| == | Equal |
| != | Not equal |
| > | Greater than |
| < | Less than |
| >= | Greater than or equal |
| <= | Less than or equal |

```
young <- factor(ifelse(age < 41, "Young", "Old"))
```




# Logical operators

In addition to comparison operators, we can use logical operators.

| | |
|---|---|
| or | `TRUE \| TRUE = TRUE`<br>`TRUE \| FALSE = TRUE`<br>`FALSE \| TRUE = TRUE`<br>`FALSE \| FALSE = FALSE` |
| and | `TRUE & TRUE = TRUE`<br>`TRUE & FALSE = FALSE`<br>`FALSE & TRUE = FALSE`<br>`FALSE & FALSE = FALSE` |

```
middleAged <- factor(ifelse(age >= 41 & age < 55,
          "Middle-aged", "Not middle-aged"))
```




# Recoding variables

We often encounter situations where we want to re-order, re-group, or re-label categories of a variable.

```
library(car)
abolishSeanad <- recode(q41,
        "1=1; 2=1; 3=2; 4=3; 5=3; else=NA",
        as.numeric.result = TRUE)
```

| old categories | new categories |
|---|---|
| 1 = Strongly agree | 1 = Agree |
| 2 = Agree | |
| 3 = Neutral | 2 = Neutral |
| 4 = Disagree | 3 = Disagree |
| 5 = Strong disagree | |




# Recoding variables

We often encounter situations where we want to re-order, re-group, or re-label categories of a variable.

```
library(car)
abolishSeanad <- recode(q41,
        "1='Agree'; 2='Agree'; 3='Neutral';
        4='Disagree'; 5='Disagree'; else=NA",
        as.factor.result = TRUE)
```

| old categories | new categories |
|---|---|
| 1 = Strongly agree | 1 = Agree |
| 2 = Agree | |
| 3 = Neutral | 2 = Neutral |
| 4 = Disagree | 3 = Disagree |
| 5 = Strong disagree | |




# Check recoding

When you recode, *always* check how it worked out.

Usually by producing a cross-table, and maybe a plot of the new variable, to see if it makes sense.

```
table(abolishSeanad, q41, exclude = NULL)
```

Note that `exclude = NULL` means that missing values are included, which is often crucial to check the recoding. Normally you do not want this in cross-tables.




# Outline




# Merging data

We often have data from different sources, that we want to put together in one data set for further analysis.

This requires that observations have the same identifier to merge on. For example, we might have data on candidates in electoral districts, and other data on the electoral districts themselves.




| party | name | district | votes |
|---|---|---|---|
| FF | O'Brien | Longford S | 10,432 |
| FG | Fitzgerald | Longford S | 5,429 |
| FF | MacGuinness | Dublin NC | 15,436 |
| Labour | Shenigan | Dublin NC | 2,013 |

| district | seats | total |
|---|---|---|
| Dublin SC | 5 | 100,410 |
| Dublin NC | 4 | 98,991 |
| Longford S | 3 | 70,001 |
| Wexford | 4 | 81,013 |




| party | name | district | votes |
|---|---|---|---|
| FF | O'Brien | Longford S | 10,432 |
| FG | Fitzgerald | Longford S | 5,429 |
| FF | MacGuinness | Dublin NC | 15,436 |
| Labour | Shenigan | Dublin NC | 2,013 |

| district | seats | total |
|---|---|---|
| Dublin SC | 5 | 100,410 |
| Dublin NC | 4 | 98,991 |
| Longford S | 3 | 70,001 |
| Wexford | 4 | 81,013 |

| party | name | district | votes | seats | total |
|---|---|---|---|---|---|
| FF | O'Brien | Longford S | 10,432 | 3 | 70,001 |
| FG | Fitzgerald | Longford S | 5,429 | 3 | 70,001 |
| FF | MacGuinness | Dublin NC | 15,436 | 4 | 98,991 |
| Labour | Shenigan | Dublin NC | 2,013 | 4 | 98,991 |




# Merging data

We often have data from different sources, that we want to put together in one data set for further analysis.

This requires that observations have the same identifier to merge on. For example, we might have data on candidates in electoral districts, and other data on the electoral districts themselves.

```
mergedData <- merge(candidates, districts,
        by = "districtID", all.x = TRUE)
```

This merges two data files, "candidates" and "districts", by a variable that occurs in each called "districtID", and we ensure that all candidates stay in the data set, even if there is no data on that particular district.




# Reshaping data

Often data that we download will have the same variable across different columns, for example for different years.

In most analyses this is difficult to work with and we need the same variable to be in one column. This requires reshaping the data.




| country | pop1991 | pop1992 | pop1993 |
|---|---|---|---|
| Brazil | 150337 | 152680 | 154964 |
| Bolivia | 6733 | 6897 | 7065 |
| Paraguay | 4345 | 4470 | 4592 |
| Chile | 13319 | 13544 | 13771 |




| country | pop1991 | pop1992 | pop1993 |
|---|---|---|---|
| Brazil | 150337 | 152680 | 154964 |
| Bolivia | 6733 | 6897 | 7065 |
| Paraguay | 4345 | 4470 | 4592 |
| Chile | 13319 | 13544 | 13771 |

| country | year | pop |
|---|---|---|
| Brazil | 1991 | 150337 |
| Brazil | 1992 | 152680 |
| Brazil | 1993 | 154964 |
| Bolivia | 1991 | 6733 |
| Bolivia | 1992 | 6897 |
| Bolivia | 1993 | 7065 |
| Paraguay | 1991 | 4345 |
| Paraguay | 1992 | 4470 |
| Paraguay | 1993 | 4592 |
| Chile | 1991 | 13319 |
| Chile | 1992 | 13544 |
| Chile | 1993 | 13771 |




# Reshaping data

Often data that we download will have the same variable across different columns, for example for different years.

In most analyses this is difficult to work with and we need the same variable to be in one column. This requires reshaping the data.

In data science (another Wickham package!) we use the terms **melt** and **cast** for the two reshape directions.

```
library(reshape)
stackedData <- melt(downloadData, id = "country")
colnames(stackedData) <- c("country", "year",
        "population")
yearMeans <- cast(stackedData, ~ year, mean,
        value = "population")
```




# Outline




# Using pipes

The below code creates a vector "a" with 4 values, and then uses the function "mean" to calculate the mean.

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

Instead of passing "a" as the first parameter when calling the "mean" function, we can use "%>%" to pass data as the first parameter. This is helpful when you want to "chain together" a series of functions, each time passing the output from one function as the input of the next.

```
library(magrittr)
a <- c(1, 4, 3, 5)
a %>% mean()
```




# Outline




# Assignments

1. **Project I 20%**. The first analysis will concern survey data and make use primarily of graphical and descriptive statistics, based on Quinlan and Okolikj (2016) and the European Election Survey 2014 for Ireland.
2. **Project II 40%**. The second analysis will focus on the use of regression analysis and classification and will be based on data on development and democracy.
3. **Project III 40%**. The third analysis will focus on the statistical analysis of text (details to be announced).




# Project 1

Suggested approach:

1. Check what the main causal relationship is proposed by Quinlan and Okolikj (2016). Visualise this relationship in the data.
2. Check what the key control variables are suggested. Use three-way or four-way visualisation to control.
3. Download the questionnaire from **GESIS**. Think whether there might be other confounding factors missing. Use three-way or four-way visualisation to check.
   Don't forget to recode/clean the variable first.
4. Write up a review of Quinlan and Okolikj (2016)'s thesis on the basis of your analysis.

Make sure to read Quinlan and Okolikj (2016) before the next class.




# Project 1

Each essay should consist of:

- a short introduction, a description and motivation of the data and methods used (25%),
- the analysis including necessary graphs and tables (35%), and
- an interpretation and conclusion (40%).

Everything needs to be properly referenced.

For the theoretical section, you can simply refer to the article the analysis is based on—focus the essay on the analysis and methodology only.

Project 1: 1000–1250 words, deadline: **28 February, 9 am**.

Word count is *excluding* captions, tables, and bibliography.




Quinlan, Stephen and Martin Okolikj. 2016. "This time it's different ... but not really! The 2014 European Parliament elections in Ireland." *Irish Political Studies* 31(2):300–314.