Session 2

A Workflow for the Social Sciences

Joan-Josep Vallbé

Friday, May 29, 2015

Quote of the day

"The problem is that doing scholarly work is intrinsically a mess."

K. Healy

How do you organize your data work?

- Where did you store the results of the exercise?
- How would you reproduce yesterday's exercise?
  - Create a script!
- Data analysis projects are fraught with danger
  - *errare humanum est*
  - keeping track of past decisions
- Now might be the right moment to think about "how you're going to organize and manage your work" (Healy 2013):
  - post-graduate stage is good to make changes
  - most of these skills not taught to students in college
- There's NOT ONE single right way to do things

A workflow for the social sciences

- **Objectives:**
  - minimize error
  - do reproducible work
  - aesthetics: turn all the material involved in a scholarly paper, report or thesis (written draft, figures, tables, references) into something beautiful—i.e. not likely a Word file
- **Background literature**:
  - Kieran Healy: Choosing Your Workflow Applications.
  - Kieran Healy: Plain Text, Papers, Pandoc

Basic ideas

- In empirical social sciences, writing a paper is **not just getting to think about something and writing it down**
- A lot of stuff is **done** before, during, and after the paper is ready
  - Drafting preliminary, sparse ideas quickly
  - Get the right data, and get the data right
  - References and citations
  - Data analysis proper
  - Keeping track of what you've done with the data... several months later
  - Documentation
  - Writing a final draft

The "Office model"

- K. Healy's distinction
- "Office model"
  - The center of your work is a Word file
  - "What you see is what you get" typewriting philosophy
  - Changes take place in **that** file
  - Data analysis is done with some other software, which produces tables and figures
  - You have to **insert** or **drag** tables and figures in that file
  - Changes in data analysis are not documented
  - The master file (`.docx`) must be circulated to other people who will edit it until a final version is reached

The "Engineer model"

- The center of your work are various **plain text** files
- "What you get is what you mean" philosophy
- Data analysis takes place in a reproducible manner
- Graphics are **referenced** in plain text files, not **dragged**, and therefore can be updated continually
- Final output files are assembled from various plain text files (`.bib`, `.Rmd`, `.R`) and compiled into `.pdf` or `.html`
- They can even be converted to `.docx`

Plain text is great

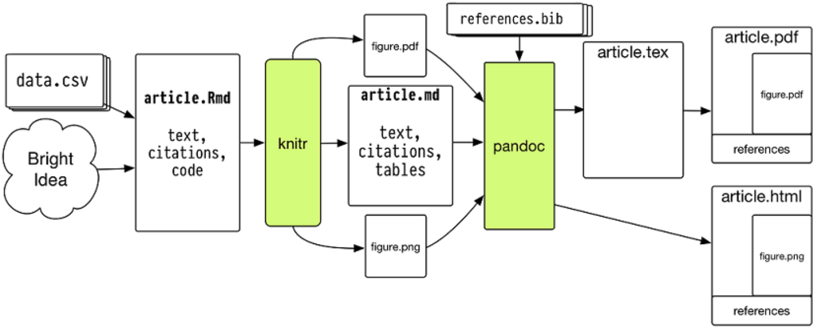
- For quantitative data analysis
  - code highlight
  - keep track of past decisions
  - can't be easily done with SPSS (click and point)
- For writing structured documents that must be revised several times
  - focus on the structure and content of what you're writing
  - let other programs take care of aesthetics
  - final output superior to Word

Adopt a model

- It's important to adopt a model, but not to be a dogmatist
  - I tend to use the Engineer model... but
  - My last co-authored paper was a `.docx` file all the time... but
  - I kept doing the data analysis "the Engineer way"
- The "Office model" dominates in Humanities and Social sciences... but this is changing
  - Need to document all your actions (write code)
  - Need to decide on some way to organize all your work
  - Try to not repeat yourself (let machines do repetitive work)
  - The "Engineer model" ensures high-quality output in pdf

A possible workflow

K. Healy



One paper, one project

- Social scientist as a programmer: give order to your **data analysis**
- Elements of a project
  1. Objectives & hypotheses (very brief)
  2. Data importing
  3. Data munging (clean, transform, etc.)
  4. Analysis
  5. Graphics
  6. Drafting paper, report, book, etc.
  7. Documentation
- R provides tools to make all this possible in one single environment

Data Analysis Project

- **What? Project on the social structure of vote and ideology**
  - Class $\rightarrow$ ideology
  - Class $\rightarrow$ party preference
- **How?** First, let's have a common project structure
  - ProjectTemplate is a very nice R package created by John Miles White that creates the structure of a project
  - **Create your project!**

```
> install.packages("ProjectTemplate") #First install the pa
> library(ProjectTemplate)            #Load the package
> getwd()                             #Know where you are
> setwd("path_to_your_project_location") #Set a location
> create.project("name_your_project") #Create the project f
```

Project structure

- Inspect the project structure
- The really relevant sub-directories are:
  - `data`: where we store our dataset(s)
  - `doc`: where we store documentation
  - `graphs`: where we send our plots
  - `munge`: where we store and execute data munging scripts
  - `reports`: where we draft our paper, report, book, etc.
  - `tests`: where we store and execute data analysis scripts
  - The rest are for more advanced users (but in 5 years I actually never used them)
- Once the structure is created, you do everything **within** the project

Brief report on objectives

- Another really nice R package that will help you is `knitr`
- It integrates the plain text formatting syntax `R Markdown` with the conversion suite `Pandoc`
- The process of writing reports, papers, or presentations becomes really easy
- Documentation on R Markdown
- Let's create your first document with `R Markdown` and `knitr`

Data import

- Once we know what we want to do, we need DATA
- Yesterday I explained that R treats data differently than other statistical packages
- One of the main differences is that R doesn't display data as others do
  - data are stored as an object or series of objects
- There are 2 ways through which we can access data in R
  - Some datasets can be accessed directly with certain R packages
  - Importing our own data

```
data()
```

Datasets from R packages

```
data(USArrests)   #This will call the object "USArrests"
                  #Note that we don't have to use <- here
ls()
?USArrests
head(USArrests)   #Head of the dataset
tail(USArrests)   #Tail fo the dataset
USArrests       #We can't do this always, be careful
nrow(USArrests)
length(USArrests)
dim(USArrests)
```

Exercise: explore the USArrests data

1. Create a `arrests.R` file in `/tests` and execute all the commands from there
2. Name all the variables with `names()`
3. List the different USA states using `rownames()` and `unique()`
4. What's the mean, median, standard deviation, maximum and minimum rate of Rape?
5. Order the data from highest murder rate to the lowest (Hint: explore the `order()` function, try `?order`)
6. What's the state with the highest murder rate? And the lowest?
7. Get the number of assaults of the state with the 20th highest murder rate

Write/save data in plain text

- We can write datasets and save tem in our `/data` directory for later use
- Imagine we want to expand the `USArrests` dataset to add further information to it (state population, ethnic diversity, etc.)
- All the locations now are relative **within** the project

```
write.table(USArrests, file="data/USArrests_data.csv")
#Go to the /data directory
```

- We must specify the criterion to separate values (comma separated values) in the plain text data file

```
write.table(USArrests, file="data/USArrests_data.csv",
            sep=",",
            row.names=TRUE)
```

Importing data in plain text

- **Good news:** `R` does not have its particular data extension (unlike SPSS with `.sav`)
- Anything in plain text can be loaded into `R` (e.g., `.csv`)

```
arrests <- read.table("data/USArrests_data.csv")
head(arrests)     #What happened?
arrests <- read.table("data/USArrests_data.csv",
                      sep=",",
                      header=TRUE)
#read.csv does this directly
arrests <- read.csv("data/USArrests_data.csv")
```

- Working with `.csv` files is the easiest, lightest way to work with data, for it's just plain text files
- But sometimes we are given data already formatted in other forms: SPSS, STATA, Excel, etc.

Importing data from other formats

- `foreign`, `memisc`, `xlsx`, `Hmisc` packages enable loading data with many formats
  - SPSS (`.sav`, `.por`)
  - STATA (`.dta`)
  - Excel
  - Minitab
  - etc.
- Start your `.R` script file and store it in `/tests`

Importing data

- Store the SPSS data for our project in your project tree
- Use the /doc and /data subdirectories
- Tip: never use original data!!!
- Search for the foreign package in Google

```
library(foreign)
?read.spss
#With labels
data <- read.spss("data/barauto_12.sav",
                  to.data.frame=TRUE)

#Labels?
```

Exercise: Know your data

1. First impression from the data
   1.2 Preliminarily explore the data with `head()` and `tail()`
   1.3 Read the documentation of the data (conveniently stored in `/doc`)
2. What's the variable indicating an individual's social class?
3. What kind of data (type of measurement) is this variable? (Hint: try `?class`)
4. How many individuals of each class are there in the data? (Hint: try `?table`)

Basic data types

- Numerical data = `a <- c(2,5,4,7,8)`
- Logical data (TRUE/FALSE data)

```
a <- c(2,5,4,7,8) #Create a numerical vector
a > 5             #We interrogate R about it
```

- Character data = `c("Theory", "of", "the", "Crows")`
- Factors
  - unordered: nominal variables = `c("PSOE", "PP", "Podemos")`
  - ordered: ordinal variables = `c("Strongly agree", "Somewhat agree", "Agree", "Somewhat disagree", "Strongly disagree")`

Levels and labels in factors

```
class(data$ESTATUS)  #Type of data
levels(data$ESTATUS) #Levels of the factor
```

```
# variable XX is coded 1, 2 or 3
# we want to attach value labels 1=psoe, 2=pp, 3=podemos (
data$XX <- factor(data$XX,
levels = c(1,2,3),
labels = c("psoe", "pp", "podemos"))
```

```
# variable YY is coded 1, 3 or 5
# we want to attach value labels 1=Low, 3=Medium, 5=High (
data$YY <- ordered(data$YY,
levels = c(1,3, 5),
labels = c("Low", "Medium", "High"))
```

Practice on factors

1. Create a numeric variable `x` that contains 5 repetitions of only 3 numeric values 1, 2, 3 (Hint: explore ?rep)
2. Convert variable `x` into a nominal factor `f` with labels 1=`A`, 2=`B`, 3=`C`
3. Convert variable `x` into an ordinal (ordered) factor `g` where 3=`Rich`, 2=`Middle`, 1=`Poor`

Managing factors

More info about factors

```
x <- rep(1:3,5)
f <- factor(x, levels=c(1,2,3),
            labels=c("A", "B", "C"))
f
```

```
##  [1] A B C A B C A B C A B C A B C
## Levels: A B C
```

```
g <- ordered(x,levels=c(3,2,1),
             labels=c("Rich", "Middle", "Poor"))
g
```

```
##  [1] Poor   Middle Rich   Poor   Middle Rich   Poor   Mi
## [11] Middle Rich   Poor   Middle Rich
## Levels: Rich < Middle < Poor
```

Tables

```
load("../data/titanic.RData")
head(titanic)
```

```
##   Class  Sex   Age Survived
## 1   1st Male Child      Yes
## 2   1st Male Child      Yes
## 3   1st Male Child      Yes
## 4   1st Male Child      Yes
## 5   1st Male Child      Yes
## 6   1st Male Adult       No
```

Contingency tables

```
attach(titanic)
table(Class,Sex)
```

```
##       Sex
## Class  Male Female
##   1st   180    145
##   2nd   179    106
##   3rd   510    196
##   Crew  862     23
```

Contingency tables

```r
table(Sex, Class, Survived)
```

```
## , , Survived = No
##
##         Class
## Sex      1st 2nd 3rd Crew
##   Male   118 154 422  670
##   Female   4  13 106    3
##
## , , Survived = Yes
##
##         Class
## Sex      1st 2nd 3rd Crew
##   Male    62  25  88  192
##   Female 141  93  90   20
```

Contingency tables

```
TitanicTable <- table(Sex,Class,Survived)
margin.table(TitanicTable,2:3)
```

```
##        Survived
## Class    No Yes
##   1st   122 203
##   2nd   167 118
##   3rd   528 178
##   Crew  673 212
```

```
margin.table(TitanicTable,1:2)
```

```
##           Class
## Sex         1st 2nd 3rd Crew
##   Male      180 179 510  862
##   Female    145 106 196   23
```

Contingency tables

```
prop.table(table(Class, Survived),2)*100 #Column percentag
```

```
##       Survived
## Class         No       Yes
##   1st   8.187919 28.551336
##   2nd  11.208054 16.596343
##   3rd  35.436242 25.035162
##   Crew 45.167785 29.817159
```

```
prop.table(table(Class, Survived),1)*100 #Row percentages
```

```
##       Survived
## Class        No      Yes
##   1st  37.53846 62.46154
##   2nd  58.59649 41.40351
##   3rd  74.78754 25.21246
##   Crew 76.04520 23.95480
```

Contingency tables

```r
prop.table(TitanicTable,2:3)*100
```

```
## , , Survived = No
##
##         Class
## Sex             1st        2nd        3rd       Crew
##   Male   96.7213115 92.2155689 79.9242424 99.5542348
##   Female  3.2786885  7.7844311 20.0757576  0.4457652
##
## , , Survived = Yes
##
##         Class
## Sex             1st        2nd        3rd       Crew
##   Male   30.5418719 21.1864407 49.4382022 90.5660377
##   Female 69.4581281 78.8135593 50.5617978  9.4339623
```

Contingency tables: xtabs()

```
detach(titanic)
TitanicTable <- xtabs(~Sex+Class+Survived, data=titanic)
TitanicTable
```

```
## , , Survived = No
##
##         Class
## Sex       1st 2nd 3rd Crew
##   Male    118 154 422  670
##   Female    4  13 106    3
##
## , , Survived = Yes
##
##         Class
## Sex       1st 2nd 3rd Crew
##   Male     62  25  88  192
##   Female  141  93  90   20
```

Flattened tables

```
ftable(TitanicTable, col.vars=c("Sex","Survived"))
```

Flattened tables

```r
ftable(TitanicTable,
       col.vars=c("Sex","Survived"))
```

```
##        Sex       Male      Female    
##        Survived    No Yes      No Yes
## Class                                
## 1st               118  62       4 141
## 2nd               154  25      13  93
## 3rd               422  88     106  90
## Crew              670 192       3  20
```

Flattened tables

```
ftable(100*prop.table(TitanicTable, 1:2),
       col.vars=c("Sex","Survived"))
```

```
##          Sex             Male                Female
##          Survived          No       Yes        No       Yes
## Class
## 1st               65.555556 34.444444  2.758621 97.241379
## 2nd               86.033520 13.966480 12.264151 87.735849
## 3rd               82.745098 17.254902 54.081633 45.918367
## Crew              77.726218 22.273782 13.043478 86.956522
```

Flattened tables

```
round(ftable(100*prop.table(TitanicTable, 1:2),
             col.vars=c("Sex","Survived")),2)
```

```
##        Sex       Male         Female
##        Survived    No   Yes      No   Yes
## Class
## 1st             65.56 34.44    2.76 97.24
## 2nd             86.03 13.97   12.26 87.74
## 3rd             82.75 17.25   54.08 45.92
## Crew            77.73 22.27   13.04 86.96
```

- Check the excellent Martin Elff's `memisc` package for more table functions!

Subsetting data: variables

- As a very flexible language, R has many ways of subsetting data

```
#Select some variables
newdata <- data[,c(1,2,3,4)]
vars <- c(1:4, 6, 9:12)
newdata <- data[vars]
vars <- c("ESTUDIO", "CUES", "CCAA", "PROV")
newdata <- data[vars]

#Exclude some variables
newdata <- data[!vars]
```

Subsetting data: observations

- We can select data on particular values of variables

```
#First 25 values
newdata <- data[1:25,]

#Based on values of 1 variable
newdata <- data[which(data$ESTUDIOS=="Primaria"),]
#Based on values of more than one variable
newdata <- data[which(data$ESTUDIOS=="Primaria" &
                        data$RECUERDO=="PSOE"),]
#More than 2 values of the same variable
newdata <- data[which(data$ESTUDIOS=="Primaria" |
                        data$ESTUDIOS=="Sin estudios"),]
```

Subsetting data: observations

- We can select data on particular values of variables

```
#First 25 values
newdata <- data[1:25,]

#Based on values of 1 variable
newdata <- data[which(data$ESTUDIOS=="Primaria"),]
#Based on values of more than one variable
newdata <- data[which(data$ESTUDIOS=="Primaria" &
                      data$RECUERDO=="PSOE"),]
#More than 2 values of the same variable
newdata <- data[which(data$ESTUDIOS=="Primaria" |
                      data$ESTUDIOS=="Sin estudios"),]

table(newdata$ESTUDIOS) #What happens?
```

Subsetting data: observations

- We can select data on particular values of variables

```
#First 25 values
newdata <- data[1:25,]

#Based on values of 1 variable
newdata <- data[which(data$ESTUDIOS=="Primaria"),]
#Based on values of more than one variable
newdata <- data[which(data$ESTUDIOS=="Primaria" &
                        data$RECUERDO=="PSOE"),]
#More than 2 values of the same variable
newdata <- data[which(data$ESTUDIOS=="Primaria" |
                        data$ESTUDIOS=="Sin estudios"),]

table(newdata$ESTUDIOS) #What happens?
newdata <- droplevels(newdata)
table(newdata$ESTUDIOS)
```

Variable recoding

- Sometimes raw data present variables with too many values that we don't need
- Usually in political survey data some factor categories present too low frequencies
    - small political parties
    - socially unacceptable opinions
    - etc.
- In practice, we usually have to reorganize variables so that they are workable
- This means recoding previously existing variables to have less or just different categories

Recoding

```
table(data$P42)
```

```
##
## Fue a votar pero no pudo hacerlo                    Fue a vot
##                               22
##                             N.C.    No fue a votar porque
##                               95
##                      No recuerda         No tenía edad p
##                               44
##                Prefirió no votar
##                             1644
```

- We should have only 2 categories here
  - voted/did not vote

Creating a new recoded variable

- **Always** create new variables, **never** recode on a variable
- The easiest way to create 2 categories is with `ifelse()`:

```
data$participation <- ifelse(data$P42=="Fue a votar y votó'
                             c("vote"), c("no vote"))
table(data$participation)
```

```
##
## no vote    vote
##    2249    8932
```

- Notice that the `no vote`category also includes `N.C.`
- Is that correct?

Recode into more than 2 categories

```
data$particip.3[data$P42=="Fue a votar y votó"] <- "vote"
data$particip.3[data$P42=="Fue a votar pero no pudo hacerlo
data$particip.3[data$P42=="No fue a votar porque no pudo"]
data$particip.3[data$P42=="No tenía edad para votar"] <- "r
data$particip.3[data$P42=="Prefirió no votar"] <- "no vote"
data$particip.3[data$P42=="N.C."] <- "N.C."
data$particip.3[data$P42=="No recuerda"] <- "N.C."

table(data$particip.3)
```

```
##
##     N.C. no vote    vote
##      139    1810    8932
```

- What about `N.C.`?

Missing data

- Sometimes we're given data with inconsistent record of missing data
- `R` uses `NA` as universal symbol for missing data
- We must check for missing data before any analysis is done

```
levels(data$P42)
```

```
table(is.na(data$P42))
```

```
##
## FALSE
## 11181
```

- Organizations use different codes for `NA` (even within the same organization: N.C., 98, 99, etc.)

Missing data

- It's usually better first deal with NA and then recode

```
data$P42[data$P42=="N.C." | data$P42=="No recuerda"] <- NA
table(is.na(data$P42))
```

```
##
## FALSE  TRUE
## 11042   139
```

```
data$participation <- ifelse(data$P42=="Fue a votar y votó'
                             c("vote"), c("no vote"))
table(data$participation)
```

```
##
## no vote    vote
##    2110    8932
```

Practice

- Exercise 2