Coding Etiquette for (non-coder) Social Scientists

Marco Morales
mam2519@columbia.edu

GR5069
Topics in Applied Data Science
for Social Scientists

Spring 2018
Columbia University

RECAP: A Data Science Project

- Three **aims** of a data science project
  - (a) **reproducibility (and replicability)**
    - anyone should be able to arrive at your **same results**
  - (b) **portability**
    - anyone should be able to **pick up where you left off** on any machine
- (a) and (b) are crucial tenets for **collaborative work**
  - (c) **scalability**
    - your project should also work for **larger data sets** and/or be on the path of **automation**

RECAP: Structuring DS projects

some basic principles...

1. use **scripts for everything** you do
   - **NEVER** do things **manually**
2. organize your scripts in a sequence
   - **separate activities** in sections
   - keep an early section for **definitions**
   - call **other scripts** when necessary
3. write **efficient** (aka lazy) code
   - turn code used multiple times into **functions**
   - **re-use functions:** make them generic enough
4. rely on **version control** (Git)

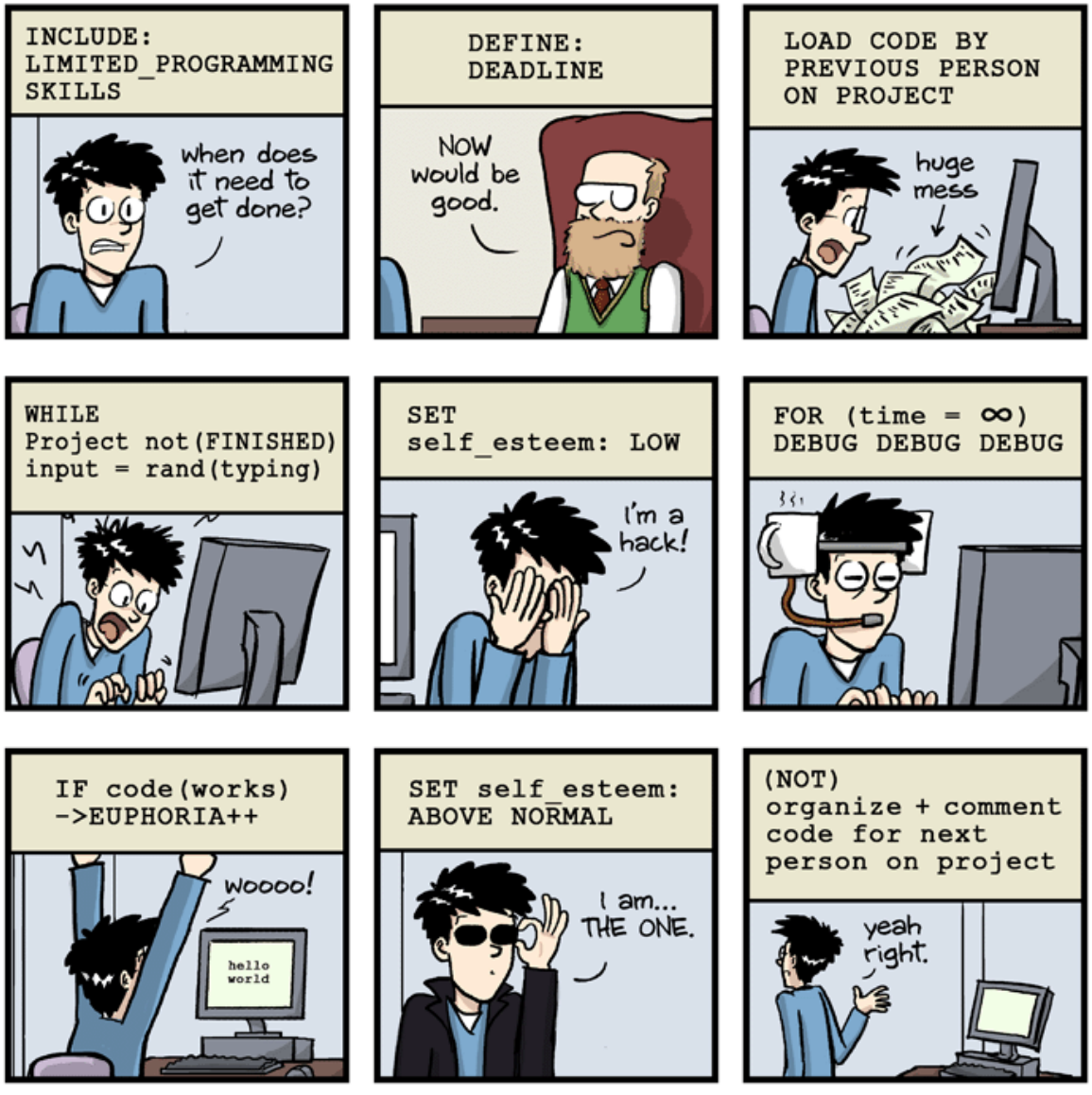
RECAP: Structuring projects

a thin layer...

```
project\
|
| -- src
|      |-- data             <- Code to read/munge raw data.
|      |-- features         <- Code to transform/append data.
|      |-- models           <- Code to analyze the data.
|      |-- visualizations   <- Code to generate visualizations.
|
| -- data
|      |-- raw              <- The original, immutable data dump.
|      |-- external         <- Data from third party sources.
|      |-- interim          <- Intermediate transformed data.
|      |-- processed        <- Final processed data set.
|
| -- reports
|      |-- documents        <- Documents synthesizing the analysis.
|      |-- figures          <- Images generated by the code.
|
| -- references             <- Data dictionaries, explanatory materials.
|
| -- README.md              <- High-level project description.
| -- TODO                   <- Future improvements, bug fixes (opt)
| -- LabNotebook            <- Chronological records of project (opt)
```

Sources: **Cookiecutter for Data Science**, **ProjectTemplate**

PROGRAMMING FOR NON-PROGRAMMERS



WWW.PHDCOMICS.COM

Coding Etiquette

Purpose of your (pseudo) code

- (Markdown / Jupyter) **notebooks** are great for **sharing** work and (code) review
  - nice sandbox to **develop** / **test** code
  - nice way to **review** code + output without having to run it
  - (usually) terrible for scaling!
- **scripts** are preferred for **running processes**
  - scripts can be run directly from source
  - you may need to extract your code from a notebook if you developed there
- **define the purpose of your code** early on!
  - avoid doing the same task twice!

Coding Etiquette

Create structured scripts

- each script should perform **only one task**
  - useful to **call additonal scripts** from your script if/when needed
  - create a **global parameters** script if/when needed
  - if too many functions, create a **separate script defining all functions**
  - separate data manipulation from data analysis in different scripts
- your code should be **as simple as possible**
  - being clever can - and will! - come back to haunt you when sharing or revisiting code

Coding Etiquette

Create structured scripts

- start your scripts with a section that provides **all relevant information** that may help you and others make sense of it in the future

```
#  ##############################################################################
#       File-Name:      MakeGraphs_CongressRollCall_160603.R
#       Version:        R 3.3.1
#       Date:           June 03, 2016
#       Author:         MM
#       Purpose:        Exploratory graphs of congressional roll call
#                       data for the 112th US Congress. Simple initial
#                       visualizations to find patterns and outliers.
#       Input Files:    ProcessedRollCall_160225.csv
#       Output Files:   Graph_RollCall_112Congress.gif
#       Data Output:    NONE
#       Previous files: MakeGraphs_CongressRollCall_160524.R
#       Dependencies:   GatherData_CongressRollCall_160222.R
#       Required by:    NONE
#       Status:         IN PROGRESS
#       Machine:        personal laptop
#  ##############################################################################
```

Coding Etiquette

Create structured scripts

- define **globally** all **important objects** that will be used throughout the project

```
# ::::::: SOME USEFUL DEFINITIONS ::::::::::::::::::::::::::::::::::::::::::

# set the general path for the project at its root, specific files will define
# their own branches individually

path <- "~//Dropbox//GR5069_Spring2017//GR5069//week_06//datachallenge1"

# define additional paths for files you will use. In each case, determine
# appropriate additions to the path

inFileName1  <- "data//raw//A-E.xlsx"           # raw data on confrontations
inFileName2  <- "data//external//ARCH535.csv"  # name equivalence tables
inFileName3  <- "data//raw//tabla9-A-E.xlsx"   # id of federal forces conf
inFileName4  <- "data//raw//A-A.xlsx"           # raw data on agressions
inFileName5  <- "data//raw//tabla9-A-A.xlsx"   # id of federal forces agg
```

  - do not add them manually at different places in the code!
  - place at beginning of the script if using a single short script
  - place on separate script if working on a larger project

Coding Etiquette

Create structured scripts

- each section of your script should perform a **single task**

```
# ::::::::::::::::: APPLY INITIAL DEFINITIONS ::::::::::::::::::::::::::::::::

setwd(path)
getwd()


# ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
# ::::::::::::::::::::: LOADS DATA ::::::::::::::::::::::::::::::::::::::::::::
# ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

Confrontations <- read_excel(inFileName1,
                             sheet = 1,
                             na = "9999"   # converting sentinel value to null
)


# ::::::::::::::::::: SOME DATA PROCESSING ::::::::::::::::::::::::::::::::::::

Forces_Confrontations <- WrangleTable(ForcesTable_Confrontations,
                                      ForcesNameLookup)

Forces_Aggressions <- WrangleTable(ForcesTable_Aggressions,
                                   ForcesNameLookup)
```

Coding Etiquette

Generate readable code

- improve the readability of your code with **spaces**, though never before a comma

```
#Good
inner_join(ForcesTable, by = c("event_id" = "ID")

#Bad
inner_join(ForcesTable,by=c("event_id"="ID")
```

- use extra spaces to **indent and align** your code to enhance readability

```
ForcesTable_Confrontations <- read_excel(
                                  inFileName3,
                                  sheet = 1,
                                  na = "9999"
                                 )
```

- **never mix spaces and tabs** to indent your code

Coding Etiquette

Take good care of your objects!

- name objects consistently - and meaningfully - throughout your scripts
  - objects should always be **lowercase**
  - be consistent if you use **CamelCase**
  - use _ to **separate words**
  - be careful when using . (may cause problems with S3 objects)

```
# Good
navy_deaths
NavyDeaths
navy.deaths # use with care

# Bad
navydeaths
ndths
```

Coding Etiquette

Take good care of your objects!

- use only <- to **assign values** to objects

```
# Good
x <- 8

# Bad
x = 8
8 -> x
```

- in general, **do not use names of existing functions or variables** for your new objects

```
# Bad
mean <- function(x) median(x)
TRUE <- 0
FALSE <- T
```

Coding Etiquette

Take good care of your objects!

- use object names that have **substantive meaning**

```
rename(
       detained = DE,
       total.people.dead = PF,
       military.dead = MIF,
       navy.dead = MAF
      )
```

- transform each object to correspond as closely as possible to a **verbal description of its contents**

```
rename(
       female = ifelse(gender == "female", 1, 0)
      )
```

- use object names that indicate **direction** where possible

```
rename(
       wounded_increase = ifelse(
                            total_wounded_change >0, 1, 0)
      )
```

Coding Etiquette

Comment your code!!

- always start your comments with **# followed by a space**
- separate your code into distinguishable chunks using visually distinct characters like `:`, `-`, or `=`

```
# ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
# ::::::::::::::::::::::: LOADS DATA ::::::::::::::::::::::::::::::::::::::::::::

AllData <- read_csv(inFileName1)


# ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
```

Coding Etiquette

Comment your code!!

- include **comments before each block of code** describing its **purpose**

```
# :::::: LOADING NAME CONVERSION TABLE
# the original file treats numeric codes as strings, must convert to integers
# upon loading. Also, names of municipalities are in Spanish, so must specify
# the encoding as the file is read

NameTable <- read_csv(inFileName2,
                      col_types = cols(
                          CVE_ENT = col_integer(),
                          NOM_ENT = col_character(),
                          NOM_ABR = col_character(),
                          CVE_MUN = col_integer(),
                          NOM_MUN = col_character()
                          ),
                      locale = locale(encoding = "ISO-8859-1")
                      )
```

Coding Etiquette

Comment your code!!

- **comment your functions** thoroughly, including **inputs** and **outputs**

```
MungeData <- function(baseEventData, StateNames, ForcesTable, SourceString){

    # ::::::::: DESCRIPTION
    #
    # The function performs the following transformations in the data to
    # produce the desired output data:
    #
    # 1. add actual names of states and municipalities from a Census table;
    #    currently the database only has their numeric codes
    # 2. rename columns from Spanish to English (not everyone speaks both languages)
    # 3. adding a new variable that indicates the armed force involved in the
    #    confrontation event
    # 4. replace all missing values with 0; this will come in handy as we start to
    #    explore the data futher
    #
    # ::::: INPUTS
    #
    # i)   BaseEventData - the raw database to be munged
    # ii)  StatesName - a table with State/Municipality names
    # iii) ForcesTable - a table that identifies armed forces involved in the event
    # iv)  SourceString  - a string that will identify origin of the table
    #
    #:::::: OUTPUT
    #
    # the function returns a dataframe
```

Coding Etiquette

Comment your code!!

- include comments for any line of code **if meaning would be ambiguous** to someone other than yourself
- sometimes not only the **why** but the **what** may be needed for others to understand the code

```
# filling in NAs with zeros, to facilitate graphing and basic computations
# replace_na() requires a list of columns and rules to apply. Code below
# provides that
replace_na(
    # creates an object with numeric column names
    setNames(
        lapply(          # applies a function that links numeric column names
                         # with the asignment of 0
           vector("list", length(select_if(., is.numeric))), # creates list len= 25
           function(x) x <- 0),  # defines assignment of 0 to numeric col names
        names(select_if(., is.numeric)))  # provides numeric column names
)
```

Coding Etiquette

Always validate that your code does what you think it does

- verify that **transformed variables** resemble what you intended

```
# create a new global unique ID
AllData %<>%
+     mutate(
+          global_id = 1:nrow(.)
+     )

# verify there are no duplicates
length(AllData$global_id) == length(unique(AllData$global_id))
[1] TRUE

# a quick look to see the distribution of the variable
summary(AllData$global_id)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
      1    1350    2698    2698    4047    5396
```

Coding Etiquette

Always validate that your code does what you think it does

- verify that **missing data is handled correctly** on any recode or creation of a new variable

```
# computes lethality indices
AllData %<>%
+   mutate(organized_crime_lethality =
+              organized_crime_dead /
+              organized_crime_wounded
+   )

# exploration to identify undefined values
summary(AllData$organized_crime_lethality)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
      0       1     Inf     Inf     Inf     Inf    3090
```

Coding Etiquette

Some general principles

- **80 characters** should be the maximum length of any line in your code
- if you find an **error** in your code, **correct it exactly where it happened**
  - do not try to fix it from a later chunk of code
- when you are done with your project, go back and:
  - **clean up** your code
  - **add comments** where appropriate (for the *you* of the future)
  - perform **stress tests** with as many **edge cases** as you can imagine
  - make sure to **document future enhancements** (especially to scale up)

Coding Etiquette for (non-coder) Social Scientists

Marco Morales
mam2519@columbia.edu

GR5069
Topics in Applied Data Science
for Social Scientists

Spring 2018
Columbia University