Best Practices in Data Science for Social Scientists

Marco Morales
mam2519@columbia.edu

GR5069
Topics in Applied Data Science
for Social Scientists

Spring 2017
Columbia University

Housekeeping

- Have you signed up on the **Slack** team?

  https://columbia-gr5069.slack.com/signup

- The course **GitHub** repo, clone it!

  https://github.com/marco-morales/QMSS-GR5069

- You've been assigned to teams. Next week, we'll:
  - communicate your project
  - create a backlog
  - have planning session

RECAP: What is Data Science?

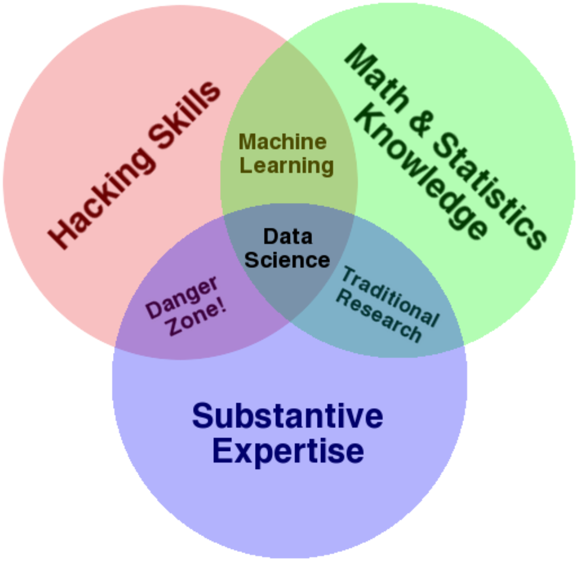

Figure: Drew Conway (2013)

RECAP: What is Data Science?

a continuum of tools...

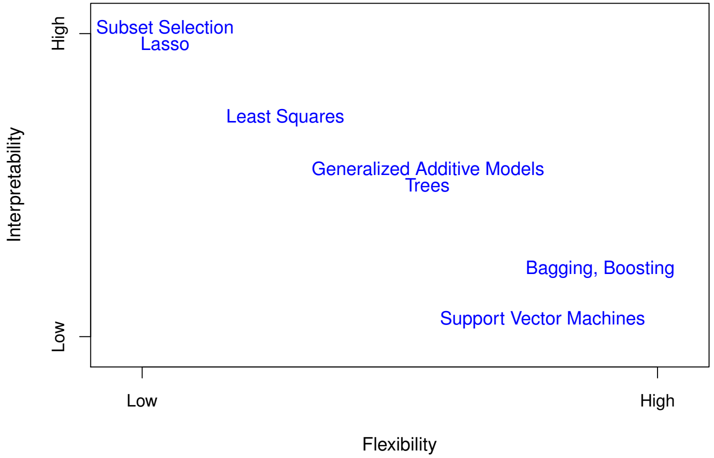

Figure: James et al. (2016)

Structuring projects

- two necessary characteristics of DS projects:
  - **reproducible**
    - a tenet of science (and of hacking too!)
  - **structured**
    - anyone can “understand” the project
- save time for you (and future you), as well as others collaborating in the project

Structuring projects

a thin layer...

```
project\
 |
 | -- src                      <- Code
 |
 | -- data                     <- Inputs
 |
 | -- reports                  <- Outputs
 |
 | -- references               <- Data dictionaries, explanatory materials.
 |
 | -- TODO.txt
 | -- README.md
 | -- LabNotebook.txt
```

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
|
| -- reports
|
| -- references
|
| -- TODO.txt
| -- README.md
| -- LabNotebook.txt
```

- **principle:** separate function definition and application

Structuring projects
a thin layer...

```
#  ########################################################################
#       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
#  ########################################################################

rm(list=ls(all=TRUE))

library(ggplot2)
library(dplyr)
```

- **principle:** include all relevant information for each script

Structuring projects

a thin layer...

```
project\
 |
 | -- src
 |
 | -- data
 |      |-- raw              <- The original, immutable data dump.
 |      |-- external         <- Data from third party sources.
 |      |-- interim          <- Intermediate transformed data.
 |      |-- processed        <- Final processed data set.
 |
 | -- reports
 |
 | -- references
 |
 | -- TODO.txt
 | -- README.md
 | -- LabNotebook.txt
```

- **principle:** input raw data and its format is always immutable

Structuring projects

a thin layer...

```
project\
 |
 | -- src
 |
 | -- data
 |
 | -- reports
 |      |-- documents       <- Documents synthesizing the analysis.
 |      |-- figures         <- Images generated by the code.
 |
 | -- references
 |
 | -- TODO.txt
 | -- README.md
 | -- LabNotebook.txt
```

- **principle:** outputs are disposable

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.
|
| -- TODO.txt               <- Future improvements, bug fixes
| -- README.md              <- High-level project description.
| -- LabNotebook.txt        <- Chronological records of project.
```

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

Structuring projects

yet another layer for naming conventions...

`FinalProject_final_ThisOneForReal_LastOne.R`

- may not be easy to remember, or scalable for reproducibility

- A few pointers:
  - create a specific structure for your filenames

    `[FUNCTION]_[PROJECT]_[VERSION]`
  - use same function names consistently across projects

    i.e. `GatherData` for ETL, `MakeGraphs` for visualizations...
  - no special characters, replace spaces with underscores

Carrying out projects

the AGILE way...

- **AGILE** is one common method in DS environments
- main entities:
  i) Dev team
  ii) Product Owner
  iii) Scrum Master
- main principle: break project down into tasks and iterate

Carrying out projects

the AGILE way: product development

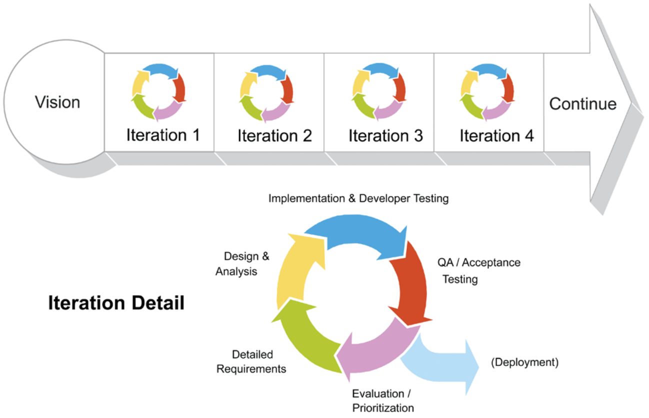

Figure: SCRUM Reference Card

Carrying out projects

the AGILE way: Backlog

ETL	**Exploration**	**Analysis**	**Output**
- input data	- descriptives	-modeling	- graphs
- clean data	- visualization		- report
- reshape data			- presentation

- each element to be broken down into **tasks**
- define taks to complete on each **sprint**
- **important concept:** definition of **done**

Carrying out projects

the AGILE way: Sprints

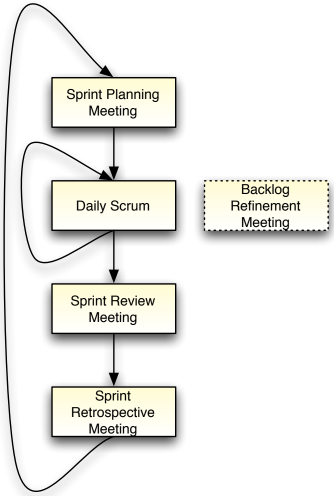

Figure: SCRUM Reference Card

Carrying out projects

the Kanban alternative...

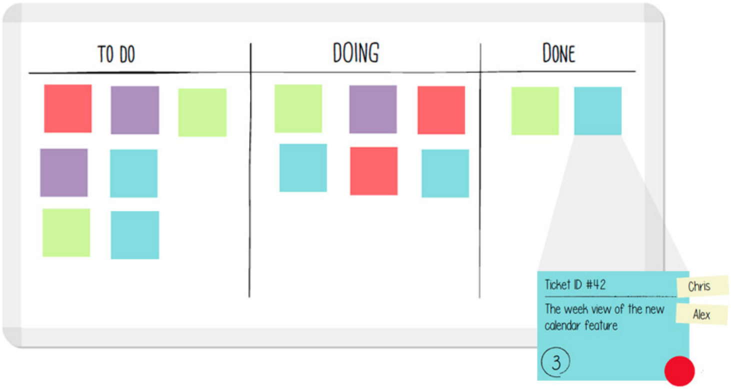

Figure: LeanKit.com

Slack

getting started...

- if you haven't done so already, sign up for the Slack team
- add your name to your profile:

  @xyz2209 might not make it easy for people to find you

- join all class-related channels and stick to their purpose
  - channels serve to order conversations
  - you will not get notified of messages on channels you are not a member of
- create channels for your teams or other purposes

Slack

class-related channels...

- **#anything-git**: solving Git/GitHub questions collaboratively
- **#anything-r**: solving R questions collaboratively
- **#anything-tidyverse**: solving tidyverse questions collaboratively
- **#anything-viz**: solving visualizations in R questions collaboratively
- **#datachallenge-n**: collaboration on solving each data challenge
- **#general**: all class-related communications, announcements and questions
- **#random**: everything else

Slack

some etiquette...

- mention people (i.e. **@marco-morales**) when speaking to them directly on a channel
  - people will not be notified unless you mention them
- use **@channel** and **@here** with care
  - **@here** notifies all people currently active in the channel
  - **@channel** notifies all members of the channel
  - **@everyone** notifies all members of the team
- be mindful of other people's time and schedules

Slack

some useful gimmicks...

- Slack works on Markdown, so it's simple to format the text of your messages
- easy to share fixed width text, or code, as well as snippets of code
- can edit messages after being sent
- integrations with other apps

Version control (and Git)

though this be madness...

- **version control** allows you to keep track of changes/progress in your code
  - keeps "snapshots" of your code over time
  - helpful to debug, and to enhance reproducibility
  - also great for team collaboration (everyone can see who changed what!)
- **Git** is a version control software
- **GitHub** is an online Git repository (on steroids)
  - widely used by data scientists (and in academia)
  - not (strictly) a "software development" tool

Version control (and Git)

...yet there is method in't!

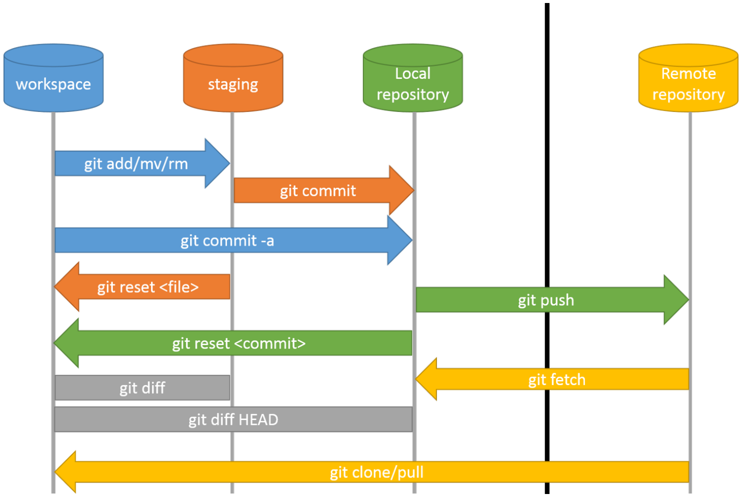

Figure: http://www.moxie.io/images/git-operations.png

Version control (and Git)

...yet there is method in't!

- some Git concepts to keep in mind
  - **clone;** a local copy of a repository that can be updated as changes happen
  - **fork**; a fork is a thread a repository.
  - **pull;** brings changes into master repository
  - **branch**; a local mirror copy of a repository at a given point in time

Version control (and Git)

...yet there is method in't!

- some useful actions in GitHub
  - `git init`: initializes Git, and indicates that the folder should be tracked
  - `git add`: brings new files to the attention of Git to be tracked as well
  - `git commit`: takes a snapshot of alerted files
  - `git push`: sends changes in your local file to the GitHub repository

Best Practices in Data Science for Social Scientists

Marco Morales
mam2519@columbia.edu

GR5069
Topics in Applied Data Science
for Social Scientists

Spring 2017
Columbia University