The data scientist's toolbox

David Puelz

The four pillars of data science

1. Data collection
2. Data cleaning and wrangling
3. Data analysis
4. Summary (figures, prose, etc.)

This class focuses mainly on 3-4, a little on 2, and not at all on 1.

Our core principles

You will analyze a lot of data in this course. Our watchwords are *transparency* and *reproducibility*.

- The end product: you will write a report with beautiful figures, and someone else will marvel at it.
- Data science is hard enough already: there is zero room for ambiguity or confusion about data or methods.
- Any competent person should be able to read your description and reproduce exactly what you did.

Our core principles

You will analyze a lot of data in this course. Our watchwords are *transparency* and *reproducibility*.

- The end product: you will write a report with beautiful figures, and someone else will marvel at it.
- Data science is hard enough already: there is zero room for ambiguity or confusion about data or methods.
- Any competent person should be able to read your description and reproduce exactly what you did.

The ideal: “hit-enter” reproducibility.

- Someone hits enter; your analyses and figures are reproduced from scratch and merged with prose, before their eyes.
- We rely on few easily mastered tools to put this ideal into practice: RMarkdown, Python/Jupyter, and git (NB: this is a “bilingual” course)

What is RMarkdown?

RMarkdown is an authoring format that allows you to create dynamic documents with R code embedded in them.

- Integrate R code, output, and visualizations seamlessly.
- Generate reports in various formats (HTML, PDF, Word, etc.).
- Reproducible data science: Share your code and results in one document.
- Easy to learn and use for data scientists and researchers.
- Supports interactive elements like Shiny apps and HTML widgets.

These slides were written in RMarkdown. You can find the raw .Rmd file on the website.

Basic RMarkdown workflow

1. Install RStudio (if you haven't already).
2. Create a new RMarkdown document in RStudio.
3. Write your narrative text using Markdown syntax.
4. Embed R code chunks to execute code.
5. "Knit" (i.e. compile) the document to generate the output in the desired format.

Follow the tutorials linked on the class website. Note: you're welcome to use Quarto instead!

Jupyter notebooks

Jupyter Notebooks are interactive computing environments that enable you to create and share documents containing live code, equations, visualizations, and narrative text.

- Code execution in real-time, providing instant feedback.
- Mix code, Markdown, LaTeX, and HTML in a single document.
- Rich media support: Display images, videos, and interactive plots.
- Facilitates data cleaning, analysis, visualization, and machine learning.
- Great for teaching, research, data exploration, and data communication.

Summary

- RMarkdown and Jupyter Notebooks are powerful tools for interactive data analysis and reporting.
- RMarkdown is (obviously) centered around R and is ideal for generating dynamic reports with R code.
- Jupyter Notebooks support multiple programming languages, with Python being the most popular.
- Both tools encourage reproducible research and facilitate collaboration among data scientists and researchers.

You can use either in this course. You'll see me use both!

You can also use Quarto, the new kid on the block in this space.

Git and GitHub

git:

- software for version control.
- ideal for collaborative work.
- the basic unit in the git universe is a *repository*, aka “repo”: a collection of files/directories all related to a single task, project, or piece of software.
- Example: the class website is a git repo.

Git and GitHub

GitHub:

- a git repository hosting service.
- a location to store your code in the cloud and easily sync it across multiple machines and multiple collaborators
- the coolest place on the Internet :-)

Git and GitHub

The git repo for our class website is stored both on GitHub (the `remote` copy) and my own computer (the `local` copy).

Basic workflow:

- Make changes to files in the `local` copy of the repo.
- `commit` those changes, thereby creating a snapshot of the repo at a single moment in time that can always be restored.
- `push` those changes to `remote`

Git and GitHub

You can use git either through:

- the command line in a Unix/Linux shell (the hard-core coder's approach)
- a graphical front-end (e.g. GitHub Desktop, SourceTree). I strongly recommend this for git first-timers!