Best Practices for Data Science Projects

Hector Corrada Bravo
Center for Bioinformatics and Computational Biology

Libraries

1. Connect/access databases

2. Data structures for fundamental objects

3. Basic operations/algorithms on these structures

4. Tools for communication

Reproducibility

- Extremely important aspect of data analysis

  - 'Starting from the same raw data, can we reproduce your analysis and obtain the same results?'

- Using libraries helps:

  - Since you don't reimplement everything, reduce programmer error

  - Large user bases serve as 'watchdog' for quality and correctness

- Standard practices help:

  - Version control: git

  - Unit testing: RUnit, testthat

  - Share and publish: github

Practical Tips

- Many tasks can be organized in modular manner:

  - Data acquisition

  - Algorithm/tool development

  - Computational analysis

  - Communication of results

Practical Tips

- Many tasks can be organized in modular manner:

  - Data acquisition: get data, put it in usable format (many 'join' operations), clean it up (wrangling)

  - Algorithm/tool development

  - Computational analysis

  - Communication of results

Practical Tips

- Many tasks can be organized in modular manner:

  - Data acquisition: get data, put it in usable format (many 'join' operations), clean it up

  - Algorithm/tool development: if new analysis tools are required

  - Computational analysis

  - Communication of results

Practical Tips

- Many tasks can be organized in modular manner:

  - Data acquisition: get data, put it in usable format (many 'join' operations), clean it up

  - Algorithm/tool development: if new analysis tools are required

  - Computational analysis: use tools to analyze data

  - Communication of results

Practical Tips

- Many tasks can be organized in modular manner:

  - Data acquisition: get data, put it in usable format (many 'join' operations), clean it up

  - Algorithm/tool development: if new analysis tools are required

  - Computational analysis: use tools to analyze data

  - Communication of results: prepare summaries of experimental results, plots, publication, upload processed data to repositories

Practical Tips

- Many tasks can be organized in modular manner:

  - Data acquisition: get data, put it in usable format (many 'join' operations), clean it up

  - Algorithm/tool development: if new analysis tools are required

  - Computational analysis: use tools to analyze data

  - Communication of results: prepare summaries of experimental results, plots, publication, upload processed data to repositories

Rarely does a single language handle all of these equally well

Practical Tips

- Many tasks can be organized in modular manner:

  - Data acquisition: get data, put it in usable format (many 'join' operations), clean it up

  - Algorithm/tool development: if new analysis tools are required

  - Computational analysis: use tools to analyze data

  - Communication of results: prepare summaries of experimental results, plots, publication, upload processed data to repositories

<p style="text-align:center; color:#ff5a3c;">Choose the best tool
for the job!</p>

Practical Tips

- Many tasks can be organized in modular manner:

  - Data acquisition: get data, put it in usable format (many 'join' operations), clean it up

    ### **R**, python or shell scripting

  - Algorithm/tool development: if new analysis tools are required

  - Computational analysis: use tools to analyze data

  - Communication of results: prepare summaries of experimental results, plots, publication, upload processed data to repositories

Practical Tips

- Many tasks can be organized in modular manner:

  - Data acquisition: get data, put it in usable format (many 'join' operations), clean it up

  - Algorithm/tool development: if new analysis tools are required

C/C++, **R** or python (depending on task)

  - Computational analysis: use tools to analyze data

  - Communication of results: prepare summaries of experimental results, plots, publication, upload processed data to repositories

Practical Tips

- Many tasks can be organized in modular manner:

  - Data acquisition: get data, put it in usable format (many 'join' operations), clean it up

  - Algorithm/tool development: if new analysis tools are required

  - Computational analysis: use tools to analyze data

    ## Best managed as shell or R/python/Ruby scripts

  - Communication of results: prepare summaries of experimental results, plots, publication, upload processed data to repositories

Practical Tips

- Many tasks can be organized in modular manner:

  - Data acquisition: get data, put it in usable format (many 'join' operations), clean it up

  - Algorithm/tool development: if new analysis tools are required

  - Computational analysis: use tools to analyze data

  - Communication of results: prepare summaries of experimental results, plots, publication, upload processed data to repositories

I use R almost exclusively

Practical Tips

- Many tasks can be organized in modular manner:

  - Data acquisition: get data, put it in usable format (many 'join' operations), clean it up

  - Algorithm/tool development: if new analysis tools are required

  - Computational analysis: use tools to analyze data

  - Communication of results: prepare summaries of experimental results, plots, publication, upload processed data to repositories

Usually all of this is managed by a *pipeline* of shell/R/ python/ruby scripts

Practical Tips

- Modularity requires organization and careful thought

- In Data Science we wear two hats

  - Algorithm/tool developer

  - **Experimentalist**: we don't get trained to think this way enough!

- It helps two consciously separate these two jobs

Think like an experimentalist

- Plan your experiment

- Gather your raw data

- Gather your tools

- Execute experiment

- Analyze

- Communicate

Think like an experimentalist

- Let this guide your organization. I find structuring my projects like this to be useful:

```
project/
| data/
| | processing_scripts
| | raw/
| | proc/
| tools/
| | src/
| | bin/
| exps
| | pipeline_scripts
| | results/
| | analysis_scripts
| | figures/
```

Think like an experimentalist

- Keep a lab notebook!

- Literate programming tools are making this easier for computational projects

  - http://en.wikipedia.org/wiki/Literate_programming

  - http://www.rstudio.com/ide/docs/r_markdown

  - http://ipython.org/notebook.html

Think like an experimentalist

- Separate experiment from analysis from communication

  - Store results of computations, write separate scripts to analyze results and make plots/tables

- **Aim for reproducibility**

  - There are serious consequences for not being careful

    - Publication retraction

    - Worse: http://videolectures.net/cancerbioinformatics2010_baggerly_irrh/

  - Lots of tools available to help, use them! Be proactive: learn about them on your own!