

Organizing collaborative projects; capturing exploratory data analysis

Karl Broman

Biostatistics & Medical Informatics, UW–Madison

`kbroman.org`

`github.com/kbroman`

`@kwbroman`

Course web: kbroman.org/AdvData

I'm trying to cover two things here: how to organize collaborative data analysis projects so that their results will be reproducible, and how to capture the results of exploratory data analysis.

The hardest part, regarding organizing projects, concerns how to coordinate with collaborators: to keep data, code, and results synchronized among collaborators. The key thing is communication, to establish shared goals and procedures.

Regarding exploratory data analysis, we want to capture the whole process: what you're trying to do, what you're thinking about, what you're seeing, and what you're concluding and why. And we want to do so without getting in the way of the creative process.

I'll sketch what I try to do, and the difficulties I've had. But I don't have all of the answers.

File organization and naming
are powerful weapons against chaos.

– Jenny Bryan

2

You don't **need** to be organized, but it sure will help others (or yourself, later), when you try to figure out what it was that you did.

Segregate all the materials for a project in one directory/folder on your harddrive.

I prefer to separate raw data from processed data, and I put code in a separate directory.

Write **ReadMe** files to explain what's what.

Organizing your stuff

```
Code/d3examples/  
  /Others/  
  /PyBroman/  
  /Rbroman/  
  /Rqtl/  
  /Rqtlcharts/  
Docs/Talks/  
  /Meetings/  
  /Others/  
  /Papers/  
  /Resume/  
  /Reviews/  
  /Travel/  
Play/  
Projects/AlanAttie/  
  /BruceTempel/  
  /Hassold_QTL/  
  /Hassold_Age/  
  /Payseur_Gough/  
  /PhyloQTL/  
  /Tar/
```

3

This is basically how I organize my hard drive. You want it to be clear where things are. You shouldn't be searching for stuff.

In my `Projects/` directory, I have a `Tar/` directory with `tar.gz` files of older projects; the same is true for other directories, like `Docs/Papers/` and `Docs/Talks/`.

Organizing your projects

```
Projects/Hassold_QTL/  
  
  Data/  
  Notes/  
  R/  
  R/Figs/  
  R/Cache/  
  Rawdata/  
  Refs/  
  
  Makefile  
  Readme.txt  
  
  Python/convertGeno.py  
  Python/convertPheno.py  
  Python/combineData.py  
  
  R/prepData.R  
  R/analysis.R  
  R/diagnostics.Rmd  
  R/ql_analysis.Rmd
```

4

This is how I'd organize a simple project.

Separate the raw data from processed data.

Separate code from data.

Include a Readme file and a Makefile.

I tend to reuse file names. Almost every project will have an `R/prepData.R` script.

Of course, each project is under version control (with git)!

`R/analysis.R` usually has exploratory analyses, and then there'll be separate `.Rmd` files with more finalized work.

Organizing a paper

```
Docs/Papers/PhyloQTL/
```

```
  Analysis/
```

```
  Data/
```

```
  Figs/
```

```
  Notes/
```

```
  R/
```

```
  SuppFigs/
```

```
  ReadMe.txt
```

```
  Makefile
```

```
  phyloqtl.tex
```

```
  phyloqtl.bib
```

```
  Submitted/
```

```
  Reviews/
```

```
  Revised/
```

```
  Final/
```

```
  Proofs/
```

5

This is how I organize the material for a paper.

R/ contains code for figures; **Analysis/** contains other analysis code; **Data/** contains data; **Figs/** contains the figures; **Notes/** contains notes or references.

Of course, a **Makefile** for compiling the PDF, and perhaps a **ReadMe** file to explain where things are.

And I'll save the submitted version (and text files with bits for web forms at submission), plus reviews, the revised version plus response to reviews, and then the final submitted version and the proofs.

Organizing a talk

```
Docs/Talks/SampleMixups/
```

```
  Figs/  
  R/
```

```
  ReadMe.txt  
  Makefile  
  bmi2013.tex
```

```
  Old/
```

6

This is how I organize the material for a talk: much like a paper, but generally a bit simpler.

Again, `R/` contains code for figures and `Figs/` contains the actual figures.

And again, a `Makefile` for compiling the PDF, and perhaps a `ReadMe` file to explain where things are.

And I'll save all old versions in `Old/`

Basic principles

- ▶ Develop your own system
- ▶ Put everything in a common directory
- ▶ Be consistent
 - directory structure; names
- ▶ Separate raw from processed data
- ▶ Separate code from data
- ▶ It should be obvious what code created what files, and what the dependencies are.
- ▶ No hand-editing of data files
- ▶ Don't use spaces in file names
- ▶ Use relative paths, not absolute paths
 - `../blah` not `~/blah` or `/users/blah`

7

I work on many different projects at the same time, and I'll come back to a project 6 months or a year later.

I don't want to spend much time figuring out where things are and how things were created: have a **Makefile**, and keep notes. But notes are not necessarily correct while a **Makefile** would be.

Plan for the whole deal to ultimately be open to others: will you be proud of the work, or embarrassed by the mess?

Your closest collaborator is you six months ago,
but you don't reply to emails.

8

I heard this from Paul Wilson, UW-Madison.

The original source is a tweet by Karen Cranston, quoting Mark Holder.

<https://twitter.com/kcranstn/status/370914072511791104>

Organization takes time.

9

There's no getting around the fact that doing things properly takes longer, in the short term.

If you have a good system and good habits, it won't seem like it takes so long.

But definitely, it's a large up-front investment in order to potentially save a lot of time and aggravation later.

Painful bits

- ▶ Coming up with good names for things
 - Code as verbs; data as nouns
- ▶ Stages of data cleaning
- ▶ Going back and redoing stuff
- ▶ Clutter of old stuff that you no longer need
- ▶ Keeping track of the order of things
 - dependencies; what gave rise to what
- ▶ Long, messy Makefiles

→ Modularity

10

I don't have many solutions to these problems. Version control helps. And try to break things down into different stages, in case one aspect needs to be revised. Maybe use different subdirectories for the different stages of data cleaning.

A point that was raised in the discussion: Have periodic “versions” for a project, perhaps labeled by date. Move all the good stuff over and retire the stuff that is no longer useful or necessary.


PUBLIC SERVICE ANNOUNCEMENT:

OUR DIFFERENT WAYS OF WRITING DATES AS NUMBERS CAN LEAD TO ONLINE CONFUSION. THAT'S WHY IN 1988 ISO SET A GLOBAL STANDARD NUMERIC DATE FORMAT.

THIS IS *THE* CORRECT WAY TO WRITE NUMERIC DATES:

2013-02-27

THE FOLLOWING FORMATS ARE THEREFORE DISCOURAGED:

02/27/2013 02/27/13 27/02/2013 27/02/13
20130227 2013.02.27 27.02.13 27-02-13
27.2.13 2013. II. 27. $2\frac{1}{2}$ -13 2013.158904109
MMXIII-II-XXVII MMXIII ^{LVII}_{CCLXV} 1330300800
 $((3+3) \times ((111+1) - 1) \times 3 / 3 - 1 / 3^3)$ 2013
10/11011/1101 02/27/20/13 $\overset{2}{0}\overset{3}{1}\overset{4}{2}\overset{7}{3}$  hissss

xkcd.com/1179

11

Go with the xkcd format for writing dates, for ease of sorting.

Problem: Variations across data files

- ▶ Different files (or parts of files!) may have different formats.
- ▶ Variables (or factor levels) may have different names in different files.
- ▶ The names of files may inconsistent.
- ▶ It's tempting to hand-edit the files. **Don't!**
- ▶ Create another meta-data file that explains what's what.

12

Scientists aren't trained in how to organize data.

Multiple people in a lab might have his/her own system, or an individual's system may change over time (or from the top to the bottom of a file!)

Create a separate file with meta-data: "These are the files. In this file, the variable is called **blah** while in that file it's **blather**."

The meta-data file should be structured as data (e.g., as a comma- or tab-delimited file) for easy parsing.

Problem: 80 million side projects

```
$ ls ~/Projects/Attie
```

AimeeNullSims/	Deuterium/	Ping/
AimeeResults/	ExtractData4Gary/	Ping2/
AnnotationFiles/	ForFirstPaper/	Ping3/
Brian/	FromAimee/	Ping4/
Chr10adipose/	GoldStandard/	Play/
Chr6_extrageno/	HumanGWAS/	Proteomics/
Chr6hotspot/	Insulin/	R/
ChrisPlaisier/	Islet_2011-05/	RBM_PlasmaUrine/
Code4Aimee/	Lusis/	R_adipose/
CompAnnot/	MappingProbes/	R_islet/
CondScans/	Microarrays/	Rawdata/
D20_2012-02-14/	MultiProbes/	Scans/
D20_Nrm_2012-02-29/	NewMap/	SimsRePower/
D20_cellcycle/	Notes/	Slco1a6/
D20corr/	NullSims/	StudyLineupMethods/
Data4Aimee/	NullSims_2009-09-10/	eQTLPaper/
Data4Tram/	PepIns_2012-02-09/	transeqTL4Lude/

13

This is a project-gone-wrong.

A key problem in research is that you don't really know what you're doing when you get started. It seems best to separate out each side-project as a separate directory, but it can be a nightmare to find things later.

If each of these subdirectories was nicely organized and had a **ReadMe** file, you could **grep** your way through them.

I sort of like the idea of separate directories for the different aspects of mucking about. And second versions are always better. Maybe we should plan to muck about separately and then bring a more refined analysis back into a common directory?

A point raised in the discussion: Put defunct side projects into an **Old/** subdirectory, and put active but not yet clearly interesting ones into **New/** or **Play/**. This will help to avoid the clutter.

Saving intermediate results

R Markdown document with details of data cleaning.

- ▶ Within the `.Rmd` file, periodically save the state of things, for further exploratory analysis.
- ▶ Put those intermediate files (which might be large) in a common subdirectory.
- ▶ The subdirectory could be under **separate** version control.
- ▶ But you'll need to **go in there** and commit files.

14

I want a reproducible analysis document, but I want to be able to grab objects from the middle of the process for further exploration. So I'll include code chunks to save the state of things, say in a `Cache` or `RData` subdirectory.

Subdirectories can be their own git repositories: Include that subdirectory in the `.gitignore` file, and then use `git init` within the subdirectory.

A point raised in the discussion: how to synchronize a project between computers? If we don't put the intermediate files in the main repository, we can't rely on GitHub. (For a simple manuscript or talk, it's okay to reconstruct things on another computer, but for big analyses, you wouldn't want to.) I use ChronoSync to synchronize my Mac desktop and laptop. Maybe Dropbox or Google Drive would be useful for this. You'd still want to use git and GitHub, but you could supplement them by having the repository sit in your Dropbox folder.

Problem: Coordinating with collaborators

- ▶ Where to put data that multiple people will work with?
- ▶ Where to put intermediate/processed data?
- ▶ Where to indicate the code that created those processed data files?
- ▶ How to divvy up tasks and know who did what?

- ▶ Need to agree on directory structure and file naming conventions
- ▶ Consider symbolic links for shared data directories

```
ln -s /z/Proj/blah  
ln -s /z/Proj/blah my_blah
```

15

Ideally, everything synchronized with git/GitHub.

The keys: planning and regular communication

Symbolic links are also called “soft links.” It’s just like a file shortcut in Windows.

Problem: Collaborators who don't use git

- ▶ Use git yourself
- ▶ Copy files to/from some shared space
 - Ideally, in an automated way
- ▶ Commit **their** changes.

16

Life would be easier if all of our analysis collaborators adopted git. Teach them how?!

When I'm working with a collaborator on a paper, I may get comments from them as a marked-up PDF. I'll save that in the repository and will incorporate and commit the changes in the source files, on my own.

Collaboration

- ▶ Do more, by working in parallel
- ▶ Do more, through diversity of ideas and skills
- ▶ Reproducible pipelines have immediate advantages
- ▶ Tests of reproducibility
- ▶ Code review

17

Collaboration has a lot of advantages, including for reproducibility efforts.

It can be useful to have a pair of people regularly review each other's code, but it can be hard to get your busy friends to pay attention to your little project. But if you are working together on a project, you can more naturally build in some code review.

Moreover, you can explicitly test the reproducibility of your analyses, by having your collaborator rerun your work, and vice versa.

Genetics of metabolic disease in mice

Alan Attie, UW-Madison, Biochemistry

Karl Broman, UW-Madison, Biostat & Med Info

Gary Churchill, Jackson Lab

Josh Coon, UW-Madison, Chemistry

Federico Rey, UW-Madison, Microbiology

Brian Yandell, UW-Madison, Statistics

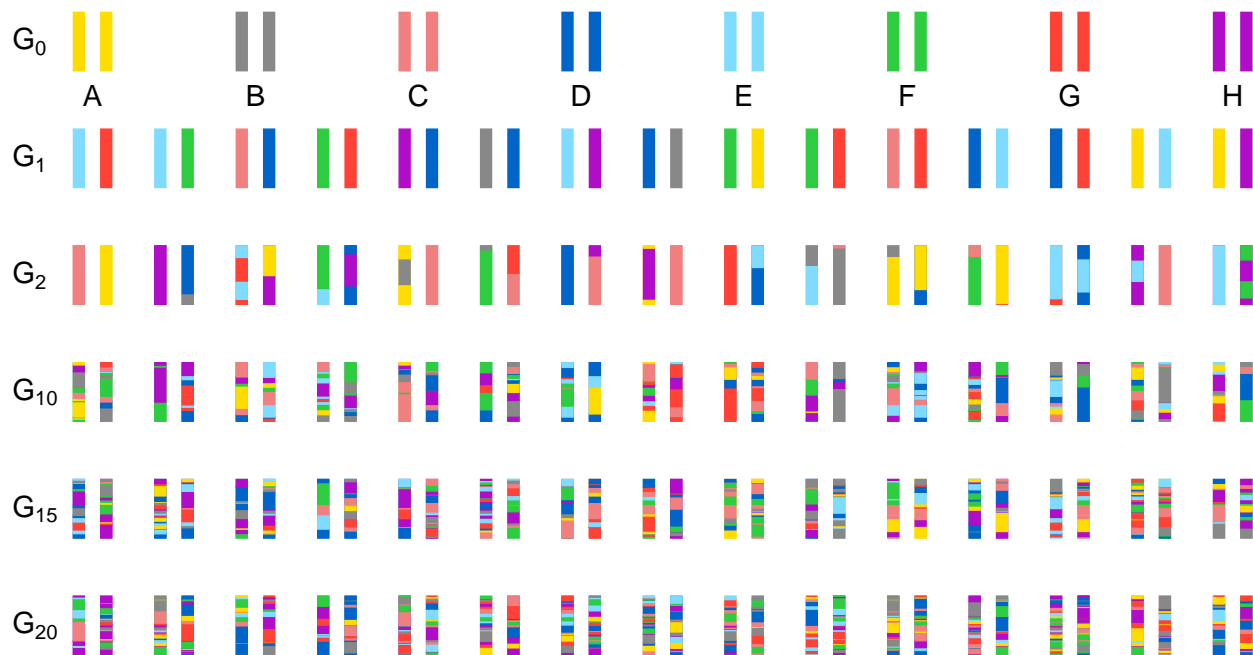


18

I want to motivate subsequent discussion with a particular example of a collaboration. There are lots of people involved. A particular challenge is that there are folks from two institutions. But even within UW-Madison, we are in five different departments, with five separate computing systems.

Our project concerns the genetics of diabetes and obesity, using an advanced intercross among eight strains of mice.

Diversity outbred mice



19

We're using an experimental mouse population called the Diversity Outbreds, which are derived by repeated outcrossing among eight inbred lines. We have data on 500 mice from generations 17–23.

Data

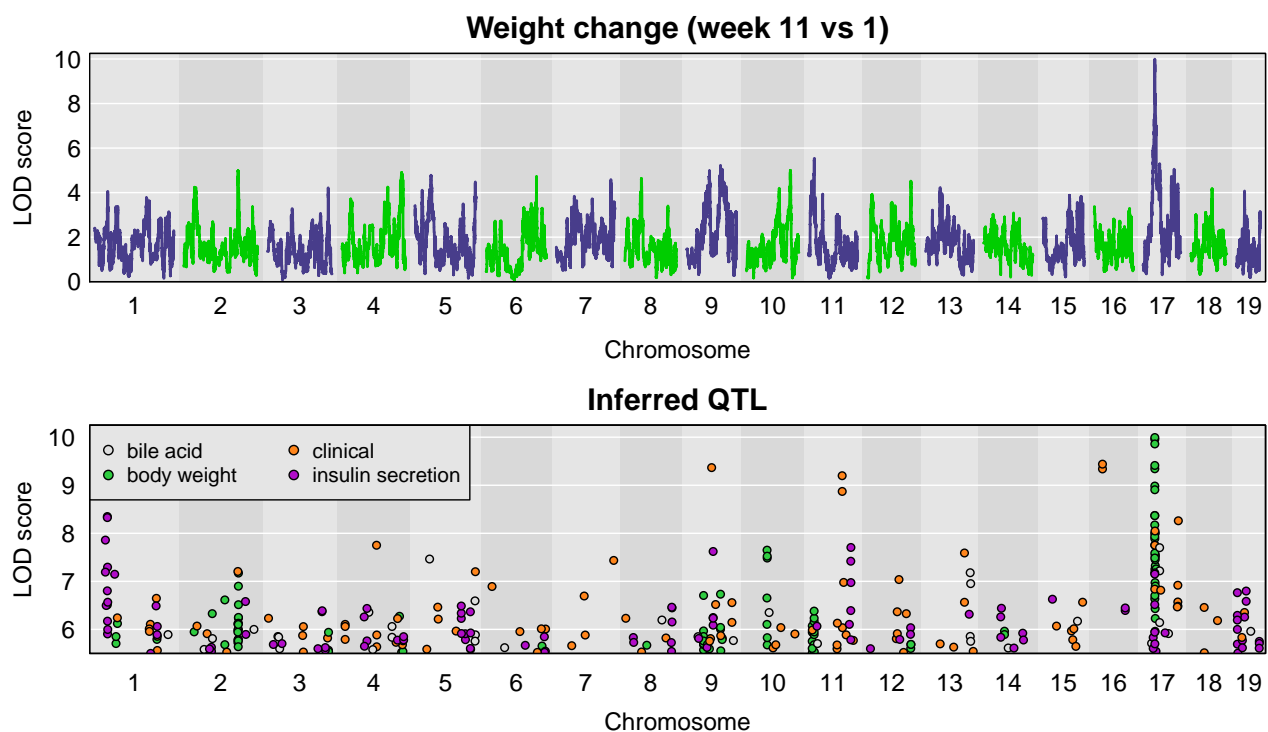
- ▶ 500 DO mice
 - generations 17–23
 - high fat, high sugar diet
- ▶ GigaMUGA SNP arrays
 - 140k SNPs
- ▶ Clinical traits
 - Weekly body weight
 - Glucose tolerance test
 - Longitudinal serum samples
 - ex vivo islet insulin secretion
- ▶ Islet gene expression by RNA-seq
- ▶ Proteins by mass spec
- ▶ Lipids by mass spec
- ▶ Gut microbiome
 - 16S RNA
 - metagenomic data

20

We have a large and diverse set of data on 500 DO mice who were fed a high fat, high sugar diet. We have high-density SNP data, a variety of clinical traits, gene expression, protein, and lipid measurements, and gut microbiome data.

Different data sets were generated at different times in different labs, and need separate preprocessing and data cleaning procedures.

Genome scans



21

Our basic analysis is to scan the genome for each trait, one at a time, assessing the association between genotype at each position and the trait data. We look for peaks in the test statistic, like that on chromosome 17 in the top panel.

In the lower panel, we plot all of these inferred QTL (quantitative trait loci, genetic loci that affect quantitative traits) for about 100 or so traits. There are a lot of downstream analyses to look at, but we're particularly interested in diverse traits that are affected by genotype at a common locus.

Challenges in collaborations

- ▶ Shared vision?
- ▶ Compromise
- ▶ Coordination
- ▶ Communication
- ▶ Sharing code and data
- ▶ Synchronization
- ▶ Weakest link?

22

Collaboration also has challenges.

Do you have a shared vision for the reproducibility of the project? You'll no doubt need to make some compromises about how things are done: you can't both just do things the way you've always done them. Careful coordination and regular communication are key.

And then there are the technical challenges of how to share the code and data and make sure your two working projects remain in sync.

In a sense, the reproducibility of a collaborative project is dependent on the weakest link. If one collaborator refuses to fully participate and share their work, the chain is broken.

Challenges

(totally hypothetical)

23

A collaboration like this will pose many challenges. The following are **totally hypothetical**. Really.

“Could we meet to talk about the data file structure?”

“No.”

24

Say the first of many sets of data are set up in a way that is complicated to handle, both in data entry and for analysis. Will your collaborator work with you to refine things?

Or will every new data file require a day of work, so that it can be combined with prior data?

**“Wait, these results seem to be based
on the older SNP map.”**

25

It can be hard to keep in sync across groups in a multi-site project. If a problem is discovered and some aspect of data preprocessing needs to be redone, will this get communicated to all analysis teams, so that relevant analyses get rerun as needed?

“Could you write the methods section?”

“But I didn’t do the work,
and we don’t have the code that was used.”

Are all teams sharing their work with each other?

“My data analyst has taken a job at Google.”

27

What happens if a key data analyst leaves the project?

“Could you do these analyses? X said they would, but they’re not responding to my emails.”

28

Everyone has multiple things going on, and sometimes there is need for rush analyses, say for a grant submission or conference presentation. Is there a shared understanding of who will do what when, and how emergencies can be handled?

The organization of a project often depends on the worst day you spent on it. If you need to do a bunch of stuff last-minute, will you leave the project directory in a mess, or will you clean up after yourself?

Shared vision

- ▶ Publication
- ▶ Code & data sharing
- ▶ Who will do what
- ▶ Timeline
- ▶ Ongoing sharing of methods, results

29

Critical for a successful collaboration is that the collaborators have a shared vision for the project. We often maybe think about being in agreement on the approach to publication and co-authorship. But perhaps more difficult is coming to an agreement on data and code sharing (what, where, and when?), on who will do what, on how soon it will be done, and on the ongoing sharing, among collaborators, of detailed methods and results.

Shared workspace

- ▶ Project structure
- ▶ Data and metadata formats
- ▶ Software environment
- ▶ Automated sync (or it won't happen)

30

Also important is the technology or engineering of sharing. Can the collaborators agree on the project structure, data and metadata formats, and the software environment?

Some groups may use R and some python. This should not pose a problem.

A key issue is how to keep the multiple groups' work in sync. It is best that this can be done automatically. Experience demonstrates that if synchronization approach requires some manual steps, they will not be done consistently.

Technology for sharing

- ▶ Data
 - figshare
 - dropbox / box / google drive
- ▶ Code
 - github / bitbucket
- ▶ Pipeline / workflow
 - make / drake / snakemake / rake
- ▶ Full environment
 - docker containers
 - mybinder.org / wholetale.org

31

I must admit to not being totally confident about what advice to give, regarding the tools to use for sharing data and code among collaborators.

For sharing data, simple options include posting large files on a data repository like figshare, or using cloud drive like dropbox, box, or google drive.

For sharing code, I prefer to use a version control system like git, with github, bitbucket, or a locally-managed equivalent.

For sharing the analysis pipeline or workflow, one can incorporate a system like make (or drake, snakemake, or rake) with the code.

The full software environment could be replicated across teams using docker containers. Binder and Whole Tale are two systems for making this easier.

The most important tool is the **mindset**,
when starting, that the end product
will be reproducible.

– Keith Baggerly

So true. Desire for reproducibility is step one.

Exploratory data analysis

- ▶ what were you trying to do?
- ▶ what you're thinking about?
- ▶ what did you observe?
- ▶ what did you conclude, and why?

33

We want to be able to capture the full outcome of exploratory data analysis.

But we don't want to inhibit the creative flow. How to capture this stuff?

Avoid

- ▶ "How did I create this plot?"
- ▶ "Why did I decide to omit those six samples?"
- ▶ "Where (on the web) did I find these data?"
- ▶ "What was that interesting gene?"

I've said all of these things to myself.

Basic principles

Step 1: slow down and document.

Step 2: have sympathy for your future self.

Step 3: have a system.

35

I can't emphasize these things enough.

If you're not **thinking** about keeping track of things, you won't keep track of things.

One thing I like to do: write a set of comments describing my basic plan, and then fill in the code afterwards. It forces you to think things through, and then you'll have at least a rough sense of what you were doing, even if you don't take the time to write further comments.

Capturing EDA

- ▶ copy-and-paste from an R file
- ▶ grab code from the `.Rhistory` file
- ▶ Write an informal R Markdown file
- ▶ Write code for use with the KnitR function `spin()`

Comments like `#' This will become text`

Chunk options like so: `#+ chunk_label, echo=FALSE`

36

There are a number of techniques you can use to capture the EDA process.

You don't need to save all of the figures, but you do need to save the code and write down your motivation, observations, and conclusions.

I usually start out with a plain R file and then move to more formal R Markdown or AsciiDoc reports.

A file to `spin()`

```
#' This is a simple example of an R file for use with spin().  
  
#' We'll start by setting the seed for the RNG.  
set.seed(53079239)  
  
#' We'll first simulate some data with  $x \sim N(\mu=10, \sigma=5)$  and  
#'  $y = 2x + e$ , where  $e \sim N(\mu=0, \sigma=2)$   
x <- rnorm(100, 10, 5)  
y <- 2*x + rnorm(100, 0, 2)  
  
#' Here's a scatterplot of the data.  
plot(x, y, pch=21, bg="slateblue", las=1)
```

Here's an example R file for use with `spin()`.