Collaborating reproducibly

Karl Broman

Biostatistics & Medical Informatics
Univ. Wisconsin–Madison

kbroman.org
github.com/kbroman
@kwbroman
Slides: bit.ly/rrcollab

These are slides for a talk I gave at the AAAS meeting in Washington, DC, on 17 Feb 2019.

Source: https://github.com/kbroman/Talk_AAAS2019
These slides, with notes: https://bit.ly/rrcollab
Full slides without notes: https://bit.ly/rrcollab_nonotes

By "reproducibly," I'm referring to "computational reproducibility," by which I mean that the data and code for a project are packaged together in a way that they can be handed to someone else, who can rerun the code and get the same results—the same figures and tables. This is surprisingly hard to do, and it's even more difficult in the context of a collaboration between two or more data analysts.

```
Karl -- this is very interesting,
however you used an old version of
the data (n=143 rather than n=226).

I'm really sorry you did all that
work on the incomplete dataset.

Bruce
```


I'm an applied statistician; my goal is to help people make sense of their data. I have a lot of collaborators, and there's nothing I enjoy more than puzzling over their data. So I write a lot of reports, describing what I've done and what I've learned.

This is an email I got from a collaborator, in response to an analysis report that I had sent him. It's always a bit of a shock to get an email like this: what have I done? Why am I working with the wrong data, and where is the right data?

But what he didn't know is that by this point in my life, I'd adopted a reproducible workflow. Because I'd set things up carefully, I could just substitute in the newer dataset, type a single command ("`make`") to rerun the analyses, and get the revised report.

This is a reproducibility success story. We all make mistakes, but if our projects are reproducible, we can nimbly recover from those mistakes.

There is a second important lesson here: At the start of such reports, I always include a paragraph about our shared goals, along with some brief data summaries. By doing so, he immediately saw that I had an old version of the data. If I hadn't done so, we might never have discovered my error.

The results in Table 1 don't seem to correspond to those in Figure 2.

My computational life is not entirely rosy. This is the sort of email that will freak me out.

In what order do I run these scripts?

Sometimes the process of data file manipulation and data cleaning gets spread across a bunch of scripts that need to be executed in a particular order. Will I record this information? Is it obvious what script does what?

Where did we get this data file?

Record the provenance of all data or metadata files.

Why did I omit those samples?

I may decide to omit a few samples. Will I record why I omitted those particular samples?

How did I make that figure?

Sometimes, in the midst of a bout of exploratory data analysis, I'll create some exciting graph and have a heck of a time reproducing it afterwards.

Which image goes with which experiment?

For experimental biologists, it can be tricky to keep track of the vast set of images and experiments they perform.

“Your script is now giving an error.”

It was working last week. Well, last month, at least.

How easy is it to go back through that script’s history to see when and why it stopped working?

“The attached is similar to the code we used.”

From an email in response to my request for code used for a paper.

Reproducible

vs.

Replicable

Computational work is reproducible if the data and code are organized in a way that they can be handed to someone else, who can rerun the code and get the same results—the same figures and tables. Replicable is more stringent: can someone repeat the experiment and get the same results?

Reproducibility is a minimal standard. That something is reproducible doesn't imply that it is correct. The code may have bugs. The methods may be poorly behaved. There could be experimental artifacts.

(But reproducibility is probably associated with correctness.)

Note that some scientists say replicable for what I call reproducible, and vice versa.

kbroman.org/steps2rr

1. Organize your data & code
2. Everything with a script
3. Automate the process (GNU Make)
4. Turn scripts into reproducible reports
5. Turn repeated code into functions
6. Create a package/module
7. Use version control (git/GitHub)
8. License your software

The above are my thoughts on the basic steps to take towards full reproducibility. But don't try to change every aspect of your workflow all at once.

Organize your data and code, segregated into a single directory with clearly named files and subdirectories. Everything you do with the data is done with a script. Automate the full analysis process. (I use GNU Make.) Even better than scripts are reproducible reports, where you can capture your motivation. Then turn to improving the quality of your code, first by writing functions and then by packaging those functions with documentation. Version control is not strictly necessary for reproducibility, but it will change your life and is especially useful for collaborative projects. Finally, license your software so that others may use it.

I'll say just a bit about some of these steps.

Organize your project

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

(paraphrasing Mark Holder)

The first thing to do is to make your project understandable to others (or yourself, later, when you try to figure out what it was that you did).

Organize your project

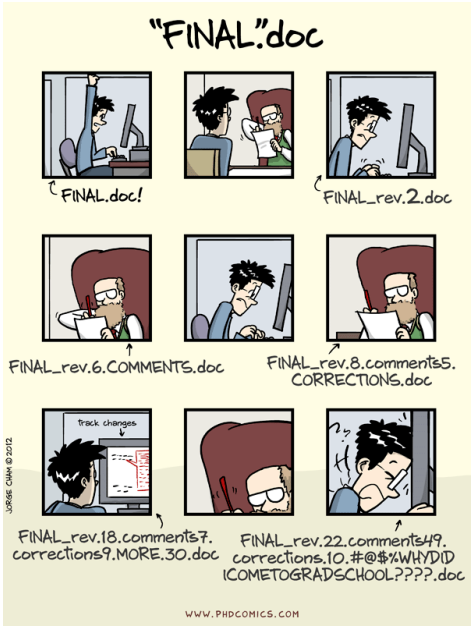
```
RawData/                    Notes/
DerivedData/                Refs/

Python/                     ReadMe.txt
R/                          ToDo.txt
Ruby/                       Makefile
```


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

This is the way I organize a project directory. The key principles are to put everything related to a project in a common directory, but then to separate data from code and separate raw data from processed data.

Write `ReadMe` files to explain what's what. Make sure they stay current.

No “final” in file names



Never include “final” in a file name.

Reproducible reports

Gough project diagnostics

Karl Broman, 3 March 2014

Combine genotypes and phenotypes

I've combined the initial genotypes (using the re-clustered genotypes for plates 14-16) with the well-behaved portion of the re-run genotypes. I'm focusing on 36813 markers that are informative (though, as we'll see, there are still a lot of badly behaved and basically non-informative markers that need to be removed). I've combined data on replicate samples, to give one set of genotype calls for each sample.

There are 1497 genotyped mice and 1464 phenotyped mice. All of the mice in the phenotype data have genotypes, but there are 33 genotyped mice with no phenotypes, including 3 Gough mice and 30 F2 progeny.

I love R Markdown for making reproducible reports that document the full details of my analysis. R Markdown mixes Markdown (for light-weight markup of text) and R code chunks; when processed with knitr, the R code is executed and results inserted into the final document.

With these informal reports, I seek to fully capture the entirety of my data explorations and decisions.

Python people should look at Jupyter notebooks.

kbroman.org/steps2rr

1. Organize your data & code
2. Everything with a script
3. Automate the process (GNU Make)
4. Turn scripts into reproducible reports
5. Turn repeated code into functions
6. Create a package/module
7. Use version control (git/GitHub)
8. License your software

Back to my summary. Again, don't try to change everything at once.

Reproducibility can be surprisingly hard and requires a daily commitment. And here I'm just thinking about a project with a single data analyst. A collaboration with multiple analysts is yet harder.

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

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.

Challenges in collaborations

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

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.

Genetics of metabolic disease in mice

Alan Attie, UW-Madison, Biochemistry

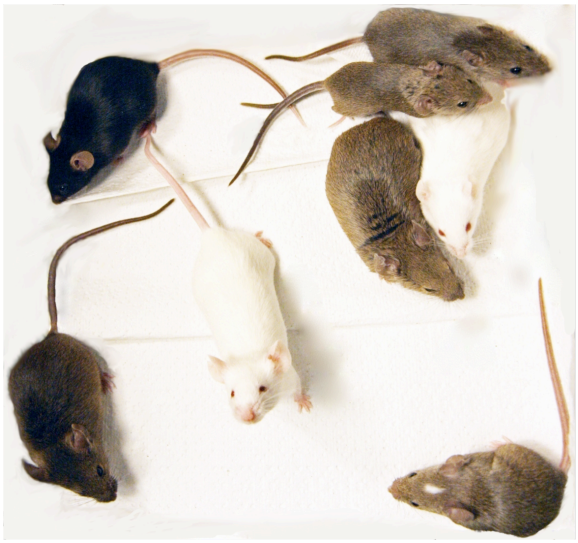
Karl Broman, UW-Madison, Biostat & Med Info

Gary Churchill, Jackson Lab

Josh Coon, UW-Madison, Chemistry

Federico Rey, UW-Madison, Microbiology

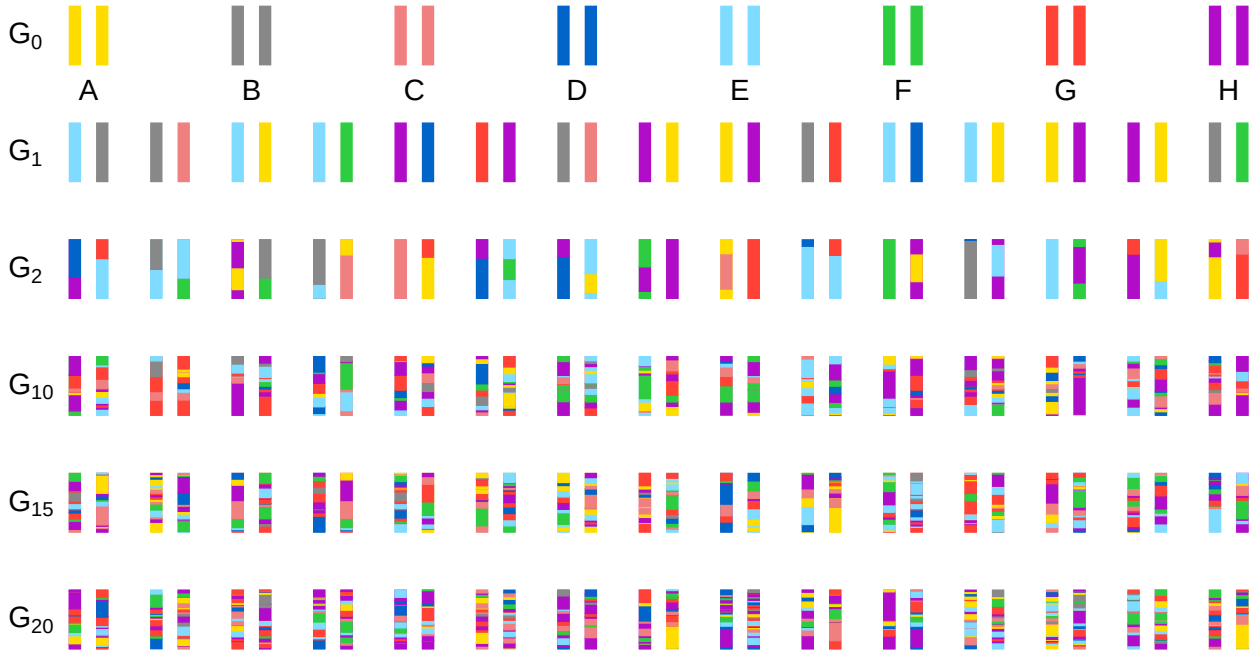
Brian Yandell, UW-Madison, Statistics



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



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.

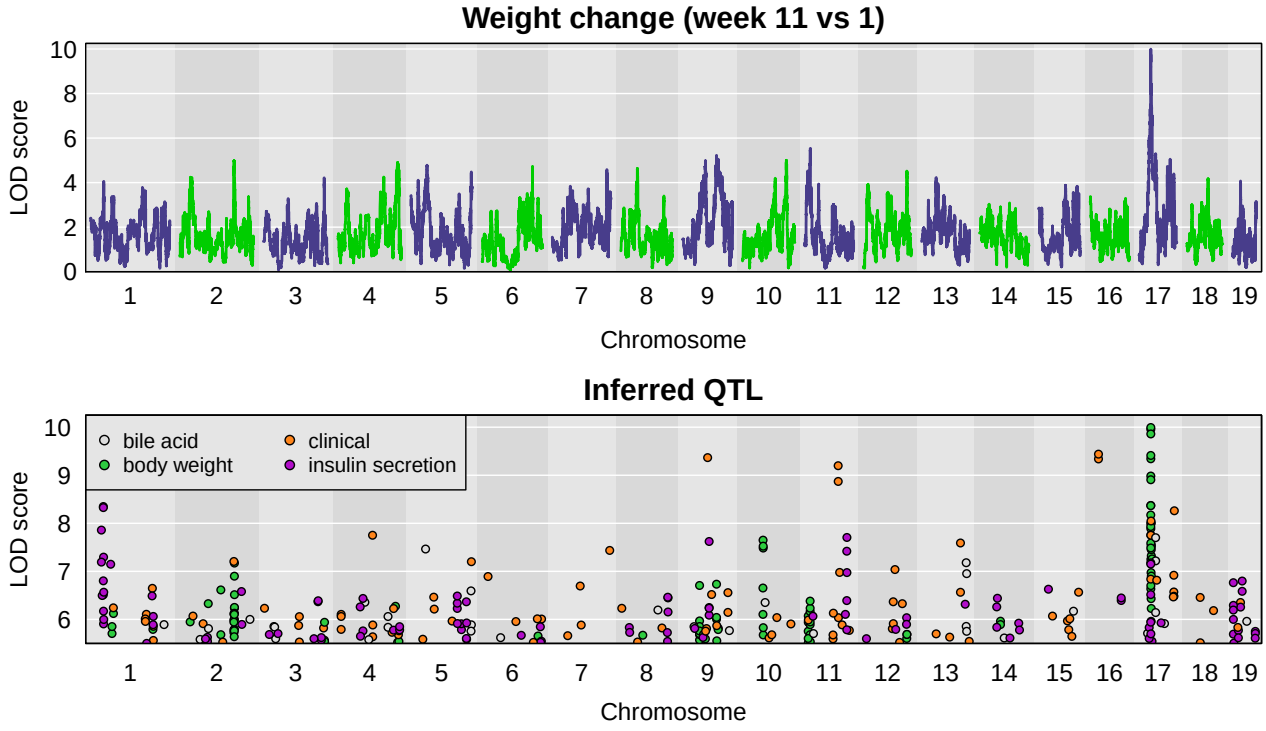
Data

- 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

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



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

(totally hypothetical)

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.”

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?

“What the heck is ‘`FAD_NAD SI 8.3_3.3G`’?”

Is there metadata, such as a data dictionary? Are those metadata in some standard form?

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

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.”

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."

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

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)

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`

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.

The second-most important tool is training.

– me

I've long felt that the key need, in getting computational scientists to adopt a reproducible workflow, is training. For the most part, all of the software tools are available, but many people haven't incorporated them into their daily work.

Slides: bit.ly/rrcollab 

kbroman.org

github.com/kbroman

@kwbroman

Here's where you can find me, as well as the slides for this talk.