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
In what order do I run these scripts?
Where did we get this data file?
Why did I omit those samples?
How did I make that figure?
"Your script is now giving an error."
"The attached is similar to the code we used."
Reproducible
Reproducible vs. Replicable
Reproducible

vs.

Correct
Levels of quality

► Are the tables and figures reproducible from the code and data?

► Does the code actually do what you think it does?

► In addition to what was done, is it clear why it was done?
  (e.g., how were parameter settings chosen?)

► Can the code be used for other data?

► Can you extend the code to do other things?
Steps toward reproducible research

kbroman.org/steps2rr
1. Everything with a script

If you do something once, you'll do it 1000 times.
2. Organize your data & code

File organization and naming are powerful weapons against chaos.

– Jenny Bryan
2. Organize your data & code

You mostly collaborate with yourself, and me-from-2-months-ago never responds to email.

– Mark Holder
2. Organize your data & code

<table>
<thead>
<tr>
<th>Directory</th>
<th>File</th>
</tr>
</thead>
<tbody>
<tr>
<td>RawData/</td>
<td>ReadMe.txt</td>
</tr>
<tr>
<td>DataSummary/</td>
<td>ToDo.txt</td>
</tr>
<tr>
<td>DerivedData/</td>
<td>Makefile</td>
</tr>
<tr>
<td>Python/</td>
<td>Notes/</td>
</tr>
<tr>
<td>R/</td>
<td>Refs/</td>
</tr>
<tr>
<td>Ruby/</td>
<td></td>
</tr>
</tbody>
</table>
3. Automate the process (GNU Make)

```
rqtlexper.pdf: rqtlexper.tex rqtlexper.bib fig1.pdf fig2.pdf
    pdflatex rqtlexper
    bibtex rqtlexper
    pdflatex rqtlexper
    pdflatex rqtlexper

rqtlexper.tex: rqtlexper.Rnw Data/lines_code_detail.txt
    R -e 'library(knitr);knit("rqtlexper.Rnw")'

fig1.pdf: R/create_fig1.R
    cd R;R CMD BATCH create_fig1.R

fig2.pdf: R/create_fig2.R Data/nlines_by_ver.csv
    cd R;R CMD BATCH create_fig2.R

Data/nlines_by_ver.csv: Python/grab_lines.py Data/versions.txt
    Python/grab_lines.py
```
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.
4. Turn scripts into reproducible reports

Gough project diagnostics

Karl Broman, 3 March 2014

<table>
<thead>
<tr>
<th>Line</th>
<th>Code Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>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 <code>r totmar(g)</code> 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).</td>
</tr>
<tr>
<td>30</td>
<td>I've combined data on replicate samples, to give one set of genotype calls for each sample.</td>
</tr>
<tr>
<td>33</td>
<td>There are <code>r nind(g)</code> genotyped mice and <code>r nrow(phe)</code> phenotyped mice. All of the mice in the phenotype data have genotypes, but there are <code>r sum(is.na(match(gid, pid)))</code> genotyped mice with no phenotypes, including <code>r sum(g$pheno$gen[which(is.na(match(gid, pid)))]==0)</code> Gough mice and <code>r sum(g$pheno$gen[which(is.na(match(gid, pid)))]==2)</code> F2 progeny.</td>
</tr>
</tbody>
</table>
5. Turn repeated code into functions

# Python
def read_genotypes (filename):
    "Read matrix of genotype data"

# R
plot_genotypes <- function(genotypes, ...)
{
}
6. Create a package/module

Don't repeat yourself
7. Use version control (git/GitHub)
7. Use version control (git/GitHub)

Talk for MAGIC Workshop in Cambridge, UK

These are slides for a talk I will give at the Workshop on MAGIC-type populations in Cambridge, UK, on 12 June 2013.

The PDF is here.

To the extent possible under law, Karl Broman has waived all copyright and related or neighboring rights to "MAGIC design and other topics". This work is published from: United States.
7. Use version control (git/GitHub)

Greatly simplify the public domain stuff in the ReadMe

- **Figs**: Add crazy table from preCC paper
- **Perl**: Add lines_of_code_by_version.csv to repository
- **R**: Another fix regarding map expansion in 8-way RIL by selfing at k=0
- **.gitignore**: Add lines_of_code_by_version.csv to repository
- **Makefile**: Revise Readme to link to version for web
- **ReadMe.md**: Greatly simplify the public domain stuff in the ReadMe
- **magic.tex**: Fix two slight bugs in slides:
7. Use version control (git/GitHub)
7. Use version control (git/GitHub)
8. License your software

Pick a license, any license

– Jeff Atwood
Summary

1. Everything with a script
2. Organize your data & code
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. Pick a license, any license
Slides: bit.ly/Broman2015-05

kbroman.org

github.com/kbroman

@kwkbroman