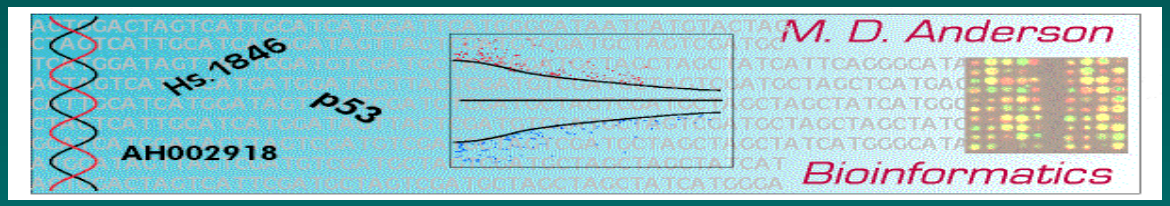
Reproducible Research with R and RStudio C02-C11: Infrastructure

Keith A. Baggerly
Bioinformatics and Computational Biology
UT M. D. Anderson Cancer Center
kabagg@mdanderson.org

ENAR, Mar 25, 2018



C02: What We Did First

What We Did First

Adopt literate programming

Keep text describing what we intend to do next to code showing how we did it

Use Sweave

Adopt a common report structure

Assemble template reports for common tasks

Impose second person review

Things I'd Add/Change Today

Use Markdown/RMarkdown

Work publicly in spirit

Keep everything together in a folder (project)

Use a consistent folder structure and workflow

Write a README for the project

Put raw data in/pull raw data from repositories

Use nice filenames

Use relative paths (use here)

Bundle scripts and templates in packages

C03: Markdown/RMarkdown

Markdown

“Markdown is a text-to-HTML conversion tool for web writers. Markdown allows you to write using easy-to-read, easy-to-write plain text format, then convert it to structurally valid XHTML (or HTML).”

“The idea is that a Markdown-formatted document should be publishable as-is, as plain text, without looking like it’s been marked up with tags or formatting instructions.”

– *John Gruber*, creator of Markdown

Markdown is Pretty Simple

Text based, like (and somewhat derived from) email

Ideally lets you focus on writing, not formatting

Aside: working with HTML really brings home what a large fraction of printing is associated with proper placement of items on a fixed page.

Markdown can be filtered to produce output in several formats (e.g., html, pdf, docx)

RMarkdown

Lets us include code chunks which will be evaluated in-place (mostly R, but also allows for some calls to other languages, e.g. Python)

Lets us include figures

Is built into RStudio along with pandoc

Is *much* easier than LaTeX-based *Sweave* for a novice

This last was the “killer app” aspect that made us push it strongly in our department

RMarkdown Demo Concepts

It's much easier to show most of this live.
I'm listing topics to make sure I hit them

RStudio's Markdown help

The YAML header

Code in YAML (e.g., dates)

Tables of contents

toc: true, number_sections: true, code_folding: hide

Section headings

Demo Concepts Contd

Links and URLs

Including figures

Emphasis and italics

Inline code

Code font

Block quotes

Bullet Lists

Numbered Lists

Code Chunk Concepts

Name your code chunks!

eval=FALSE, warning=FALSE, message=FALSE

echo=FALSE

fig.width, fig.height

cache=TRUE

auto-completion!

Different Output Formats

html (default) - html_document

pdf (requires LaTeX) - pdf_document

docx - word_document

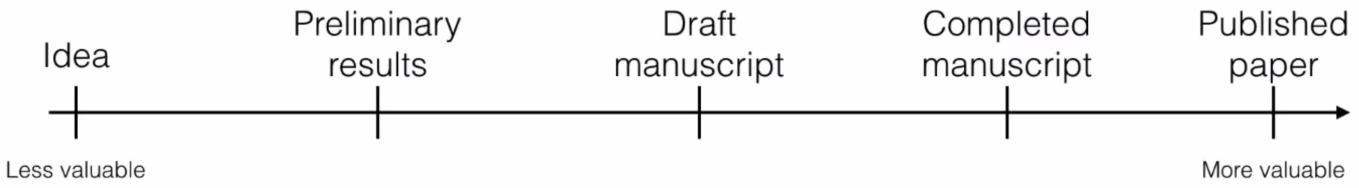
markdown - github_document

Slides (beamer)

C04: Work Publicly In Spirit

Work Publicly (In Spirit)

How I thought of my goals in grad school:



How I should have been thinking of them:

From David Robinson's keynote at eUSR 2017 (27m01s)

Get Used to Putting Things on the Web

Sanity self-check: If I start from just what I've posted, could I get what I want?

Would someone else find this clear?

Ease of sharing with collaborators

I do this by defining a GitHub repository, and then creating a new RStudio project from GitHub

Jenny Bryan: Excuse me...

Jenny Bryan: Happy Git and GitHub for the useR

The project name should be clear, even years later

Why “In Spirit”?

If you’re working with certain types of data (e.g., patient data with HIPAA constraints),
there may be limits on what can be placed out in the wild.

It’s often possible to set up instantiations of similar repositories (e.g., gitlab) within a department or institution which can allow for greater access control and authentication.

The mindset: There is a place, outside my computer and potentially accessible by others, where the main part of my analysis will be consistently findable. Once I’ve shared this with others, I won’t modify it in an unrecoverable way.

Using Git and GitHub

If you have an account on GitHub (free), and you create a git version controlled repository (a project), RStudio can set up a new project using a link to the repository.

This means the simplest interactions with GitHub (pushing to upload to, and pulling to download from) are button clicks.

GitHub "automatically" renders Markdown as html

GitHub will automatically render a README.md as (part of) the default home page for the repository.

GitHub and Findability

Include the GitHub repo URL in the intro of reports you write.
Your collaborators will be able to find it later.

At least as important, if they come back asking for modifications a few years(!) later, you can find it.

Every journal article does essentially this
by including the name of the journal, page numbers, and other information at the top or bottom of every page.

Looking at a New R Project

Note the presence of ".Rproj" and ".gitignore" files

The .Rproj file is placed in the top (root) folder of the project's hierarchy, and locations of all other files and folders can be specified relative to .Rproj.

This type of "anchoring" is what git uses (targeting .git) to position files, and can be used in R with the "rprojroot" and "here" packages (the latter has a simpler syntax)

Anchoring lets us make the entire project more "modular". We never have to refer to where things are relative to our own (unique) filesystem.

We want to AVOID using getwd and setwd.

Using here

Inside an R project, invoke

```
library(here)
here()
```

If you're working in an R script, and you want to save my_stuff.RData to results/, then

```
save(my_stuff,
   file = here("results", "my_stuff.RData"))
```

should position things correctly, as long as your working directory is in any subfolder of the project.

Loading should work similarly

Jenny Bryan: here here!

C05: Use a Consistent File Structure

Use a Consistent Folder Structure

Good Enough Guidelines

Who among us has never had a folder filled with a plethora of apparently unrelated files?

I still have plenty...

Something that'll help you and others understand the workflow (and potentially automate it) is using a consistent set of rules for putting files in common places.

This is not news.

But keep the rules simple and minimal if possible.

What I Try to Use

R/ (for .R and .Rmd files)
data/ (for raw data and metadata)
results/ (for processed data, md files)
reports/ (for shared html, docx, and pdf)
README.Rmd, .md

This is minimal, but I have a pretty clear idea where to go.

I may also include:
figures/,
LICENSE.Rmd,
TODO.Rmd,
CITATION.Rmd,
etc.

What Would You Use?

My list is not sacred.

Pick a structure you're comfortable with, but use it consistently.

For each folder name in your starting list, create that folder, in the given position relative to the project root, if it doesn't already exist.

Repeat this process for files you want to create *a priori*.

Avoid Complete Uniqueness

Particularly when collaborating,
it's easier to share if we can build on existing conventions.

MS Windows

Driving on the right

Value added from interoperability

We're not the only ones to notice this, and certainly not first.

Project Template
is a neat R package that implements a few different layouts

C06: Write a README First

Write a README First

GitHub's treatment enforces the idea that
starting a project with a README is good practice.

At least in outline,
what do I think the goal of the project is?
how do I intend to pursue it?
what data do I have?
what would constitute "success" or "failure"?
what don't I understand yet?

As the project proceeds, I can add questions, links, and workflow

Sharing your README with collaborators before beginning analyses can save time!

It's Worth Spending Time On This

For me, writing out my own understanding of the project
in a way I think my collaborators will understand
generally clarifies where the gaps in my understanding are.

If I don't understand it well enough to describe it,
I don't understand it well enough to analyze it.

This can also help your collaborators clarify their thinking, and prevent you spending time on analyses they don't care about.

Let's Write a README Now

Using RMarkdown, of course!

Given the Rmd, we want to produce md for GitHub

Edit the YAML to indicate output will be a "github_document".

In addition to the md, this will produce a "preview version" html so you can see what your file will look like when posted.

As it happens, md is an intermediate step in producing many other types of output. Thus, we can also set "keep_md: true" for other document types in the YAML.

C07: Gathering and Describing Data

Gather the Raw Data

Ideally, this involves writing an R script which will acquire the raw data from a formal repository (which is unlikely to move).

This can make your project and analysis more "lightweight", in that you don't need to include the raw data in what gets posted to GitHub or your other shared site.

This also encourages not keeping many different copies of the raw data in different places.

This latter practice can be frustrating, especially if the file contents are slightly different and you're not sure which one is most current.

If the Data's Not in a Repo, Change That

There should be one canonical copy.
With trackable provenance.

If I worry about the stability of the place I got it from, or I'm only getting the data from back and forth emails, one goal is to establish an agreed upon place for everyone to look for it.

I did this for one of our datasets by posting the data (and a description) to figshare. The first 20G is free...

Don't go overboard here.

If your entire dataset is fairly small (say under 1Mb), go ahead and post it, possibly compressed.

Let's Gather the Raw Data Now

Here, we'll work with the script

01_gather_raw_data.R

We won't run it now, since we're not sure about internet connectivity, just look at it.

And, before looking inside it, let's look at the name.

Nomenclature en passant

Numbers at left (with zero padding!) for sequencing

Dates use YYYY-MM-DD format

On Naming Files

From Jenny Bryan's Speakerdeck slides on Naming Things

Three principles:

machine readable (parseable with regular expressions)

human readable (we can guess what it does)

plays well with default ordering
(if we run all files in a folder in sequence, will we be ok?)

I use “downloader”

As with other steps, I really try to avoid pointing and clicking.

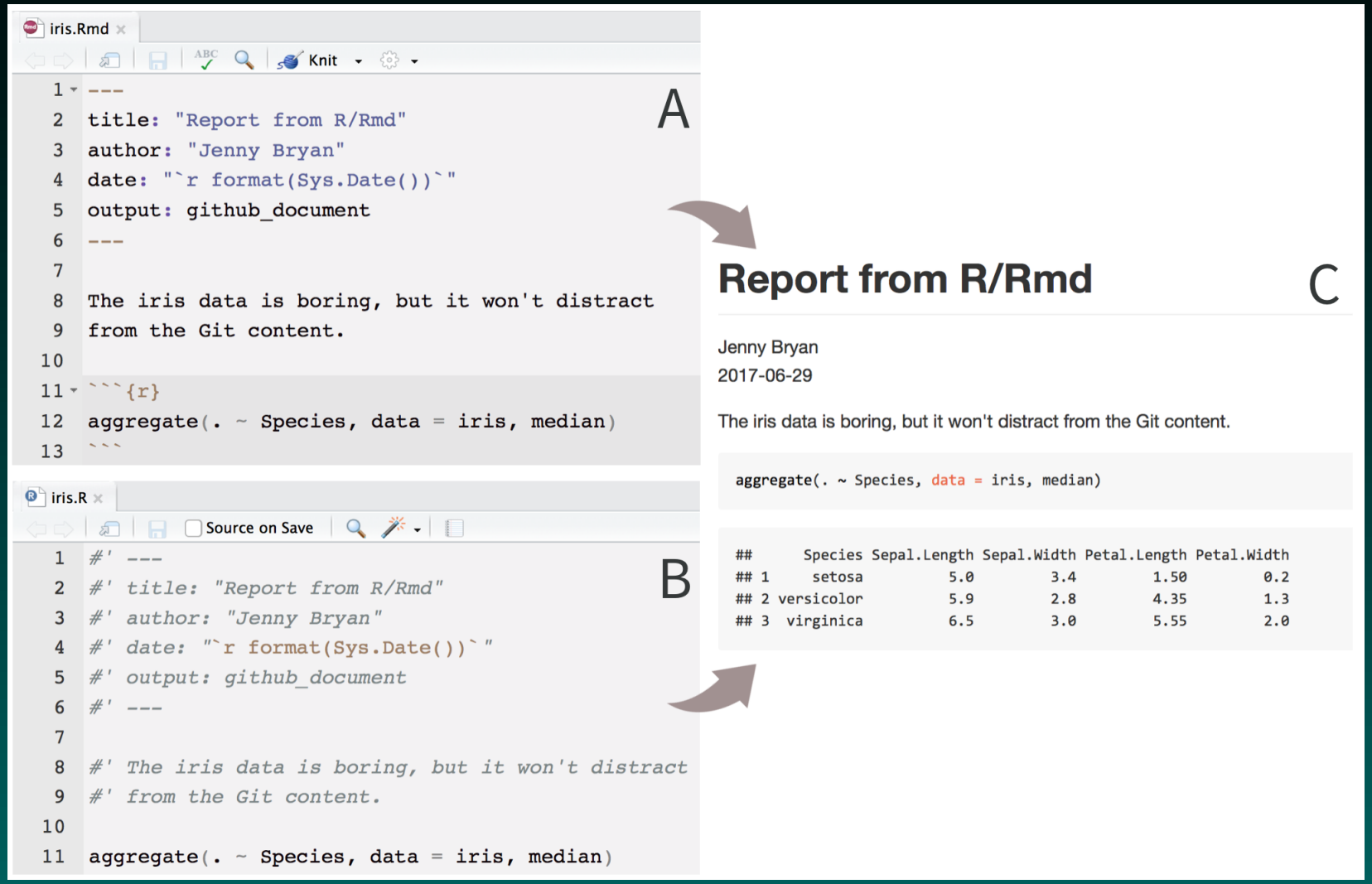
This means that while I’ll often record the base URL where the data can be found, I’ll also spend some time trying to find the right modification which lets me script the file download.

This occasionally involves poking around at the html of the top page with “View Source” in my browser’s tools.

I position the downloaded file with here.

```
download(potti_url,
  destfile = here("data", potti_data_file),
  mode = "wb")
```

R or Rmd? Equivalent!



Jenny Bryan, Excuse Me, Fig 4

Is Code or Text Primary?

We can produce md files from either R or Rmd (given YAML)!

The two types of analysis files can be used interchangeably. This is why I use one folder and one numbering scheme.

You need different comment characters, but the one for R scripts will show up again later today.

Most of the time, I try to describe what I'm doing in words at least as often as I write code, so I default to Rmd files.

If I'm writing more than a few stand-alone functions (un-numbered .R files) I try to put these in an R package.

Looping to Avoid Repeating (DRY)

The current “gather data” script has lots commented out.

Most of these are my initial attempts to download data files one at a time until I got things to work (worth doing!)

But, as I did it, it became apparent
I was using essentially the same code more than twice.

That’s a sign I should be trying to condense things by looping or writing a function.

Condensing makes the final code more consistent and maintainable.

Make All

As we start assembling R and Rmd files, we want to be able to run all of them in our preferred order.

Explicitly specifying the order clarifies the workflow, and (ideally) the dependencies involved.

Pulling files together in a specified order is an old problem. One tool for doing this in great generality is Gnu Make.

But.

Make has a somewhat arcane syntax. If we're just using R, we can just code the most common options.

One of these is make_all.

Make Clean

Aside from make all, the other most common option is make_clean.

make_clean gets rid of intermediate or derived files, both to clean up directories and to ensure that when you next run make_all, there isn't some undocumented "carryover".

Given the directory structure I'm using, I'm emptying out data/ and results/, but leaving R/ alone.

At intermediate stages, I may not clean everything every time I add one file. For example, I often don't clean out data/ if I'm sure the download step works.

rmarkdown::render

As with downloading files, we want to get away from button clicking as we produce reports (even if the knitr icon is cute).

We do this by invoking "render".

```
rmarkdown::render(
  here("R", files_in_r_to_run[i1]),
  output_format =
    github_document(html_preview = TRUE,
      toc = TRUE),
  output_dir = here("results"))
```

Again, we control locations with here. The github_document function is also in rmarkdown.

Describing the Data

I'm now shifting to 02_describing_raw_data.Rmd.

Some repositories contain metadata about the raw data (e.g., data dictionaries, sample run dates, array types, etc).

If these exist as separate files (ideal), these should also be downloaded into data/.

If the metadata do not exist in such form, they should be briefly summarized in a separate note in this Rmd file.

Loading and Viewing the Data

I also use this step to make comments about the data I see in the files supplied, before doing any processing.

If the data are tables, I often use View.

Do I understand what the measurements mean?
Do I see clearly what samples are to be contrasted?
Do the dimensions make sense?

Simply staring at the numbers for a few minutes often highlights additional questions I want to ask.

Are there ties? Missing values? Thresholded values?

C08: Data Cleaning

Cleaning the Raw Data

There's too much to this to cover it in complete generality (sorry).

That said, we want to define what constitutes clean data for our project.

Then we want to load and process the raw data to put it into this clean form.

As we do this, there are basically 3 rules.

The 3 Rules

1. treat “data/” as read-only (Jenny Bryan)
2. don’t copy and paste
3. script everything (Karl Broman)

These rules are definitely somewhat redundant.

They’re there because there should be one canonical set of raw data, and preprocessing often has as much of an effect as later analysis choices.

Save the Cleaned Data in results/

Putting these files in results/ as opposed to data/ emphasizes that these are derived quantities.

There are two things I waffle about.

1. I've saved the processed data in RData form. I might want to save the data in some type of plain text format in case someone else wants to analyze it differently, but I haven't.

2. I haven't numbered the saved files to indicate which report generated them. This is something I may change later; I'm still mulling this one. Thoughts?

Making Data Tidy?

Hadley Wickham has made a good case for assembling the data to be analyzed in a consistent, tidy, form.

In a tidy dataset, each row is an observation, each column is a variable, and each cell is a value. In the tidyverse, such data frames have been reformatted (slightly) as tibbles with cleaner display syntax and no rownames.

Being able to assume a common data format lets code be written far more generalizably, and allows for “piping” using the magrittr package, which can make code more readable.

I definitely recommend reading R for Data Science, which desribes this in more detail.

I Haven't Made These Data Tidy

I have no objection to doing so.

I'd welcome any recasting of the preprocessing and analyses that did so!

In this case, however, I found it useful to envision the array data in gene-by-sample matrix form.

This arrangement of the data let me spot something stupid which I think I might've missed had I tidied everything.

I expect I'll shift over time, but I want to better understand how to preserve my ability to check for stupidity.

C09: Analyzing Data

Analyzing Data

Adopt specific coding conventions.

Style is important. It will affect how readily people will be able to read and understand your code, and their willingness to try.

As with directory structures, picking one set of conventions and using it consistently is more important than which one you choose.

That said, in the R space, there's now one set that has a good deal of momentum.

Tidyverse Conventions

There is a tidyverse style guide.

The tidyverse is popular. Following these rules expands the set of people who'll find it easier to read your code.

There are also two R packages, styler and lintr, which will automatically check selected blocks of code for conformity with the style guide.

Style covers things like indentation, spacing, line wrapping, variable naming, etc.

Explain What You're Trying to Do and Why

I find it far easier to read code if I'm regularly reminded of what the goal is.

Avoid long blocks (say more than half a page) of uninterrupted code.

Each block of code is performing some set of operations.

Why are you performing them?
What do you expect to see as a result?
Looking at the resulting data/figure, do you see it?
What interpretation do you give to what you see?

Who is This For?

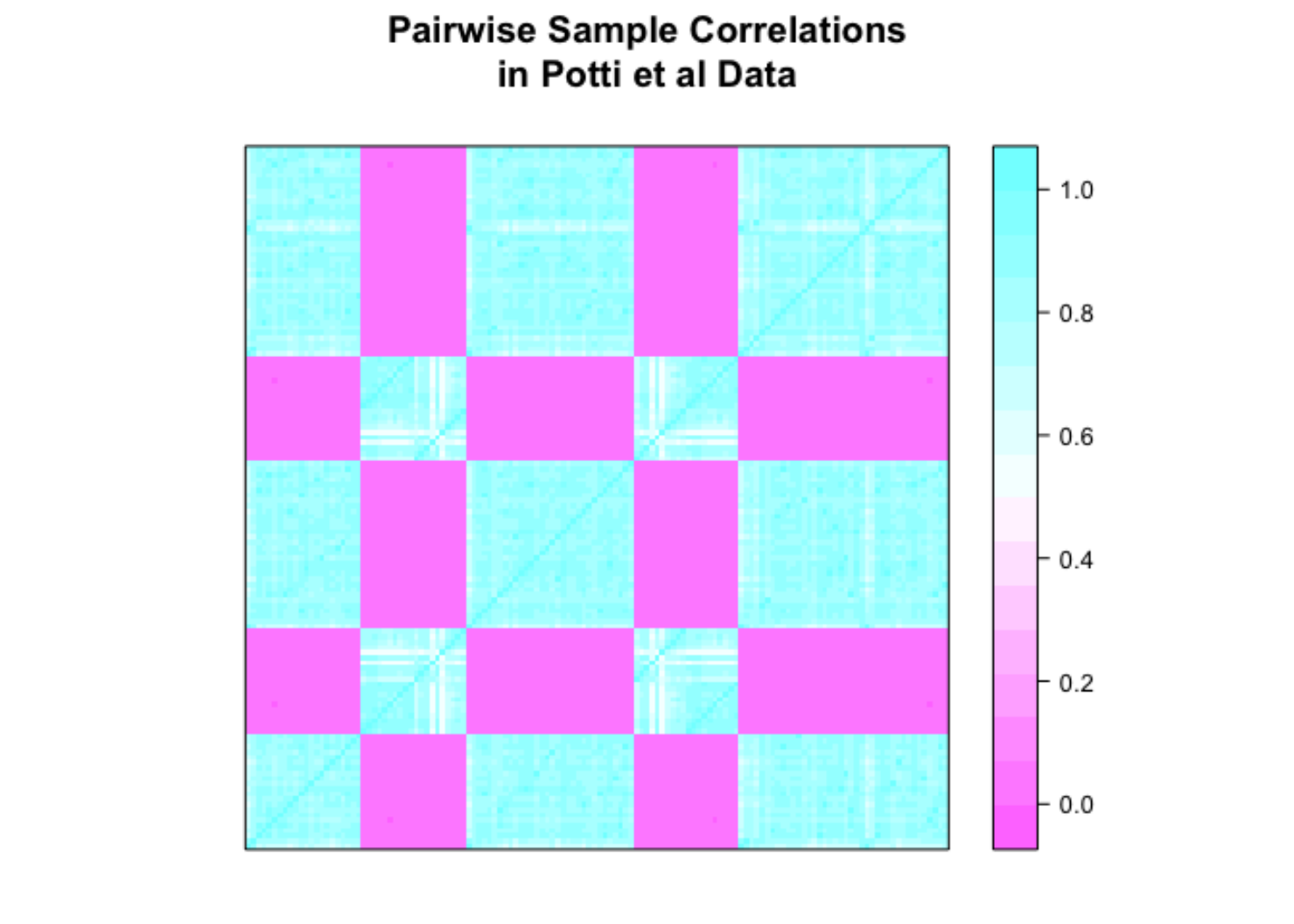
“Your worst collaborator is you from 6 months ago, and you don’t answer emails.”

– various, but I heard it from Karl Broman

Hadley Wickham describes this as writing for “future you”.

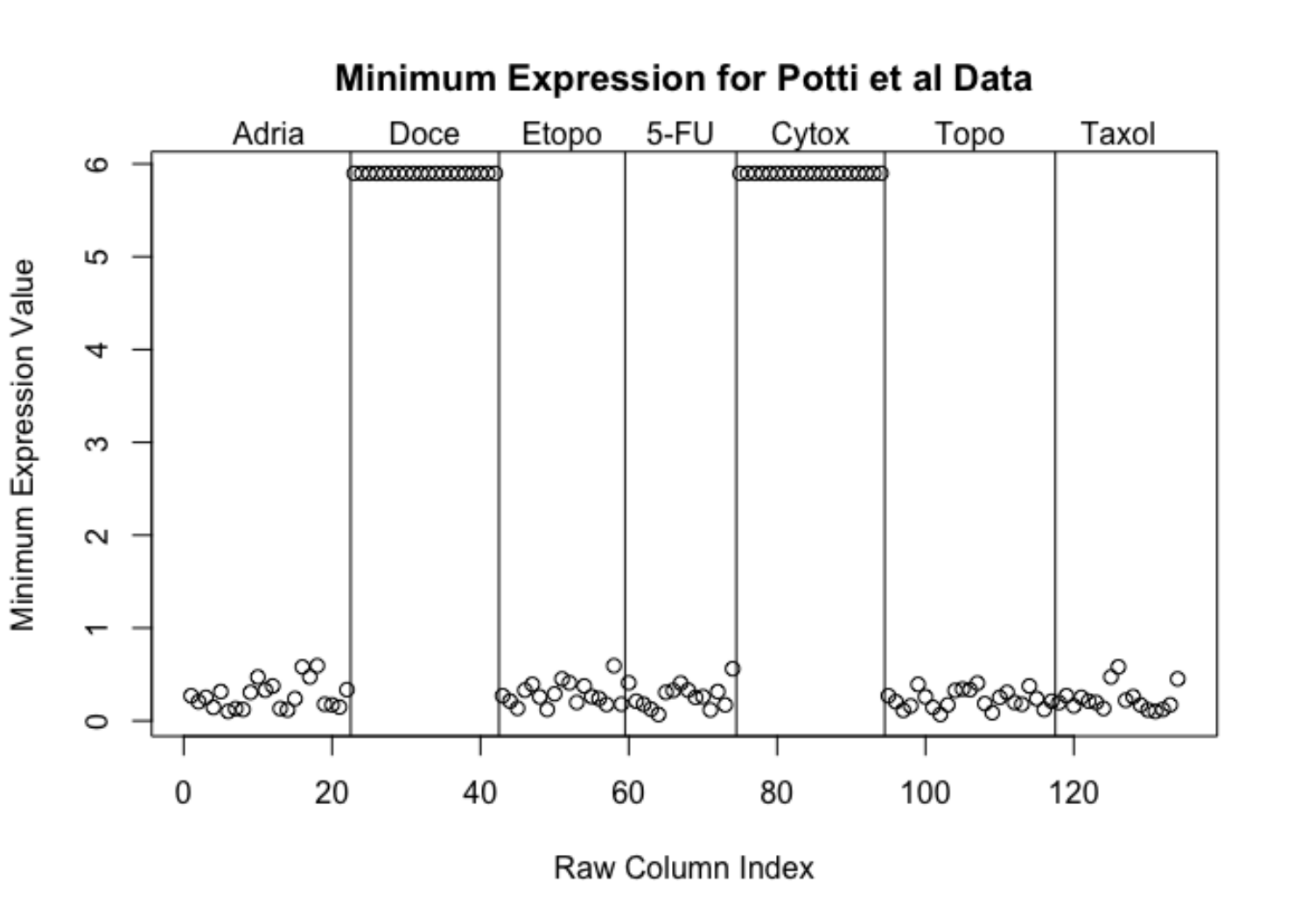
These practices, even if they take a while to become habits, can repay the time spent surprisingly quickly if you ever revisit projects.

Try to Use Figures



Label them. Explain what you see.

Try to Use Figures (2)



Remember your audience as you explain.

Figures Will Get Reused

Your pictorial summaries are the bits of your analyses that most people will focus on.

You may be asked (especially for publications) to modify your figures “just a little bit”.

If you come back to a figure more than once, consider splitting off the code generating that figure into its own standalone R script.

C10: Reporting Analyses

What A Report Means to Me

This is a summary that will go out to my collaborators directly.

While I'm happy to make my underlying scripts and component pieces available, this is where my audience changes a bit.

As a practical matter, this also means I often produce reports in several different output formats (html, pdf, docx)

I include a link to where things are (the GitHub URL), and to subsidiary reports as needed.

Writing Good Reports Enhances Reproducibility

I treat report writing as a separate topic in part based on negative feedback we got.

When we first implemented *Sweave* reporting in our group, our collaborators hated it.

They had no objections to reproducibility, but there was now too much emphasis on code.
They couldn't quickly find the results they cared about.

So, we restructured to meet both goals.

What Should A Report Contain?

At the outset, a report should clearly state

The underlying question we hope to address.

What experiments were performed, and how these experiments are expected to provide some type of answer to the question.

What analyses are being performed in the report at hand.

The results of these analyses.

The conclusions we draw from these results, and indicate where we should go next.

It should then present the analysis itself.

The Executive Summary: Structure

Introduction
– Background and Rationale
– Objectives
Data & Methods
– Data Description
– Statistical/Analytical Methods
Results
Conclusions

This what our clinicians are used to, and find easy to digest.

This summary should be in prose, ideally ≤ 2 pages.

Strive for Parallelism

If there are multiple objectives listed in the introduction,
address them in the same order
when discussing the conclusions.

Similarly, if tests A, B, and C are described in the methods,
present the results in the same order.

Parallelism improves clarity and makes it easier to “check off”
whether everything has been done.

Clarity, clarity, clarity

Questions I find myself asking report writers regularly:

Do all team members share an understanding of the goals?

Common changes I request (beyond spelling and grammar):

What do + and - mean?
Is a high value good or bad for the patient?

What do you infer from the plot you're showing?
Describe what lets lets you do this.
What would the plot look like if there's no structure?

What is this chunk of code meant to do?

What Sanity Checks Have You Employed?

What changes did you expect to see? Did you?

What changes shouldn't be there if this is working?

Have you plotted the p-values from all contrasts?

Have you plotted low dimensional summaries of the data? (e.g., PCA?)

How have you plotted the data?

Do the results make sense?

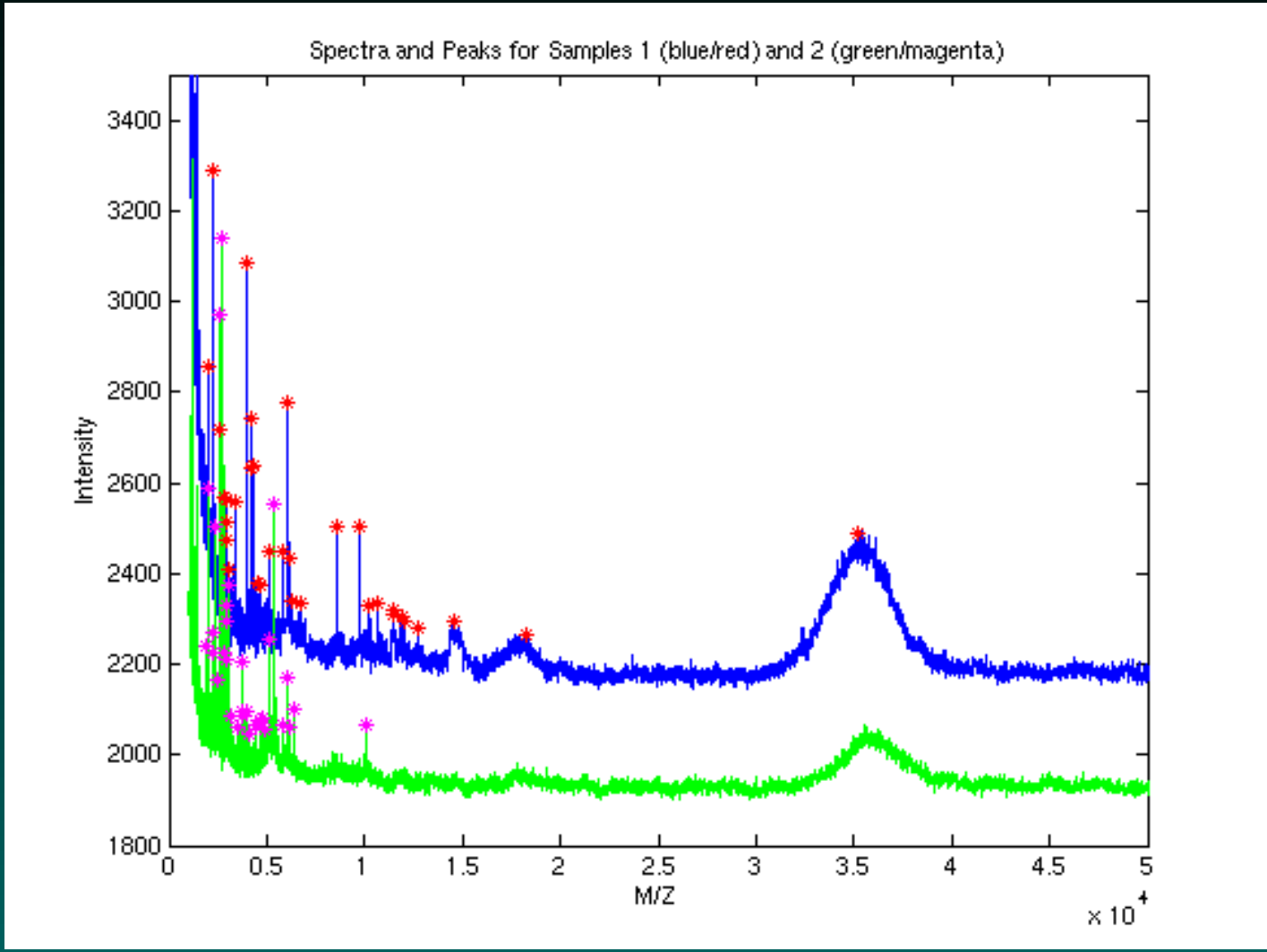
The Proteomics Data Mining Competition

41 samples, 24 with disease*, 17 controls.

20 fractions per sample.

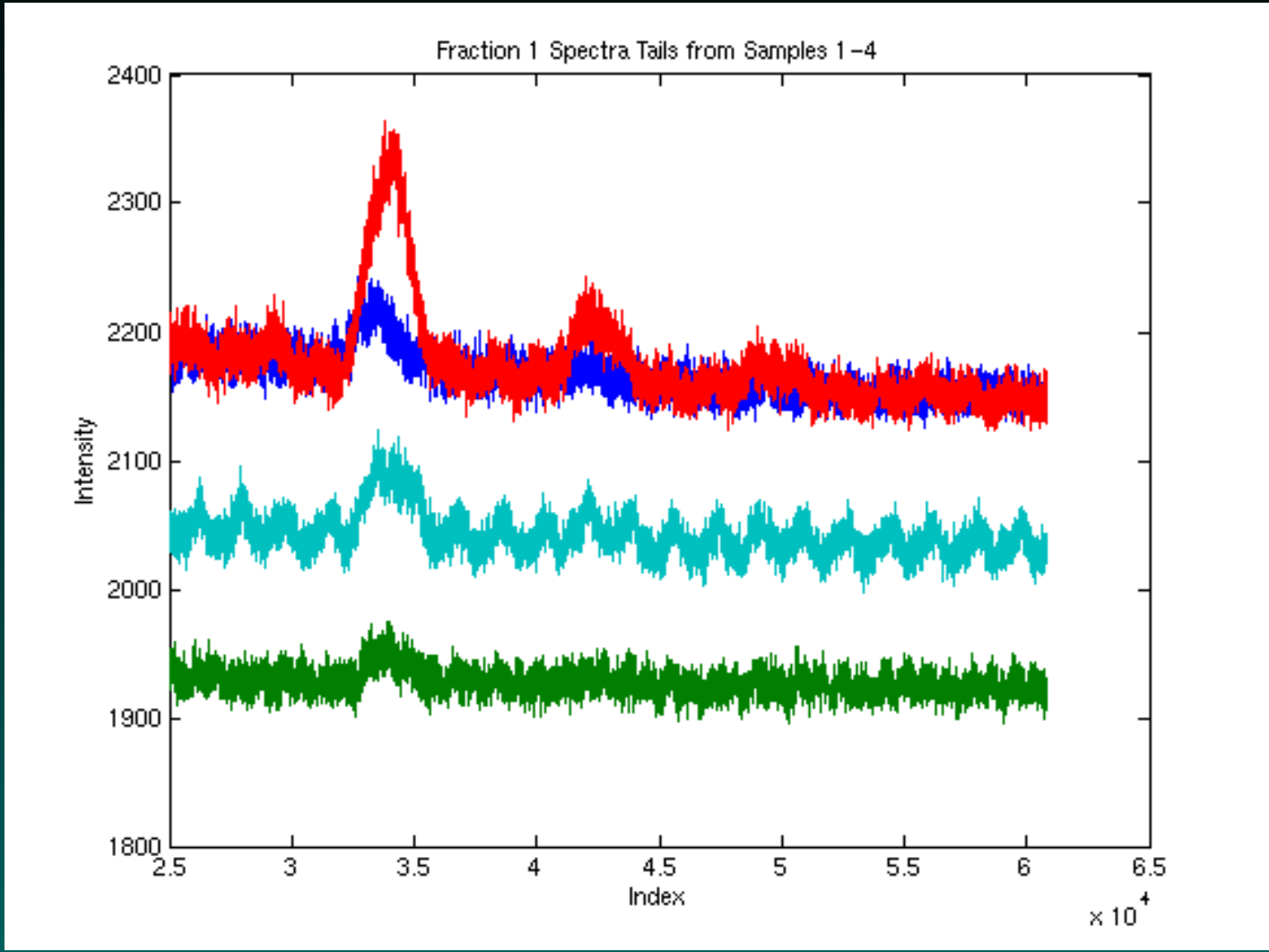
Goal: distinguish the two groups.
We know this can be done due to the “zip effect”.

Raw vs Processed – Use Raw



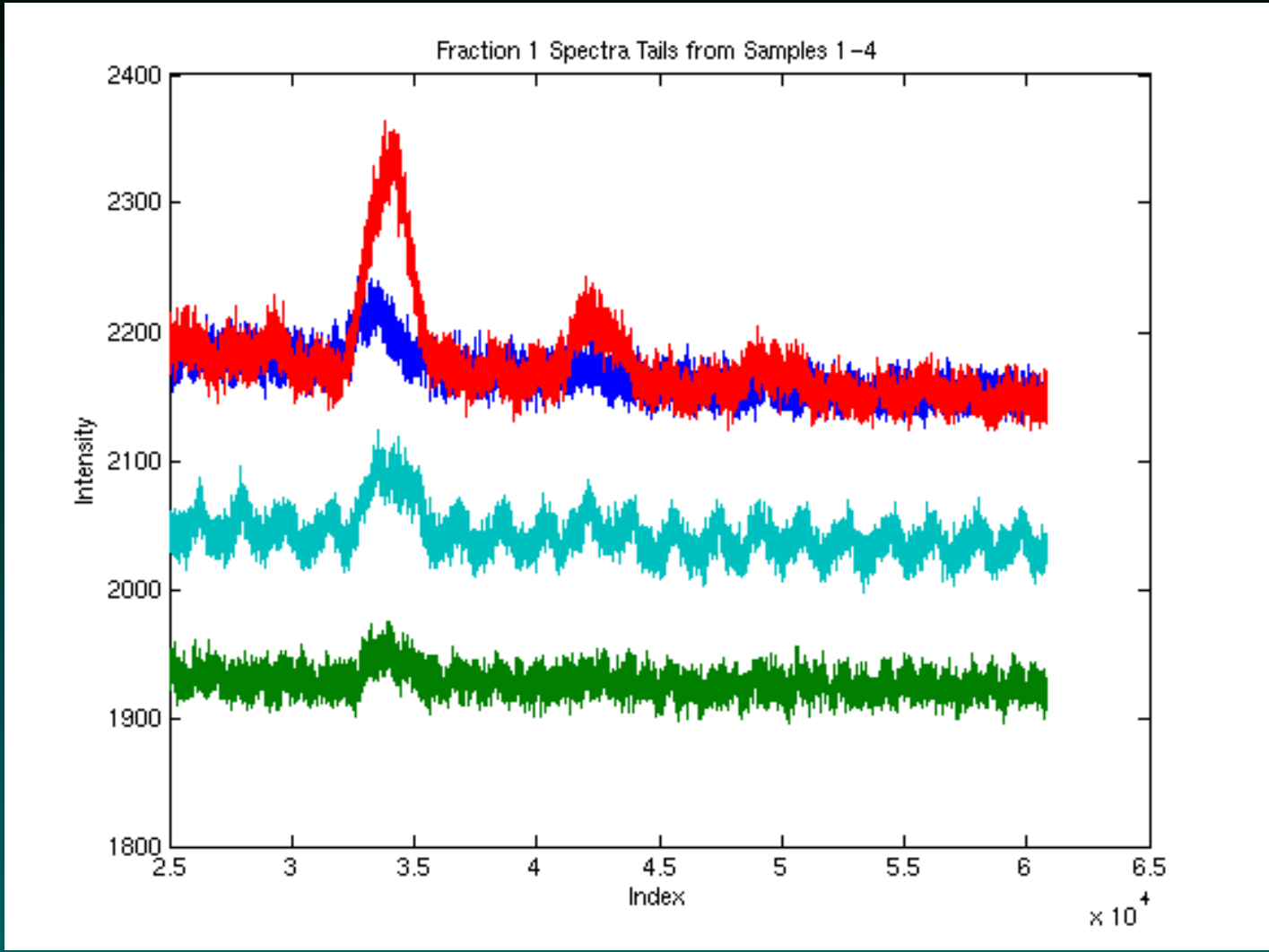
Note the need for baseline correction!

Oscillatory Behavior...



Half the spectra are “wiggly”!

Oscillatory Behavior...



Half the spectra are “wiggly”! It’s the A/C power cord.

C11: Posting and Sharing

What Should You Post?

What did I put in .gitignore?

.Rproj.user
.Rhistory
.RData
.Ruserdata
.html
data/*
results/*.html
results/*.RData

I post all .R, .Rmd, and .md files by default

What's Enough?

I've posted enough if I'd be comfortable removing the corresponding folder on my own computer because I could regenerate everything fairly quickly from what I posted.

For the project on GitHub, rerunning the entire analysis takes about 1.5 minutes on my laptop.

A big chunk of that is downloading the raw data from the repositories.