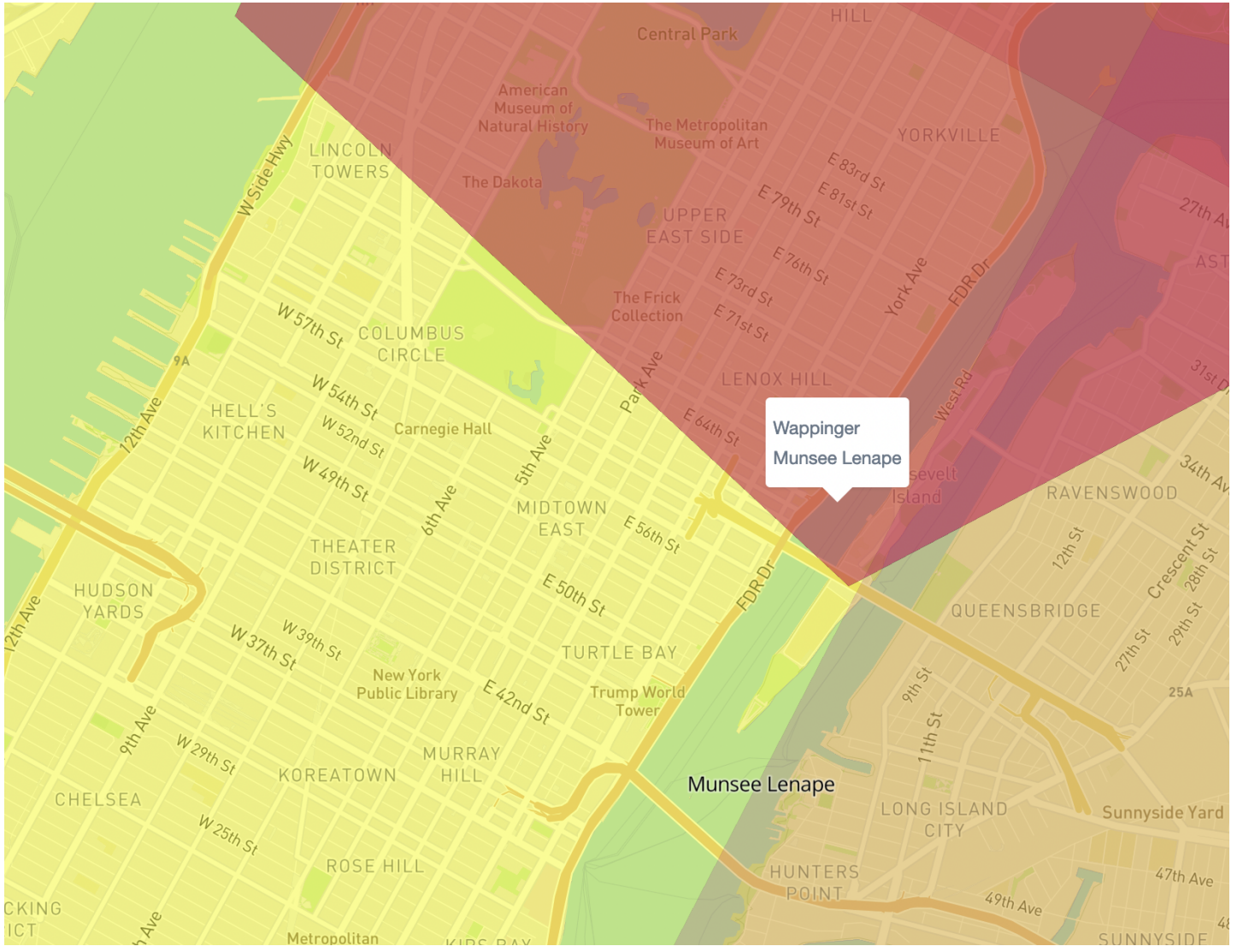
Combining Analysis Work with Reports and Presentations

What I Would Change About My Dissertation

Daniel Chen

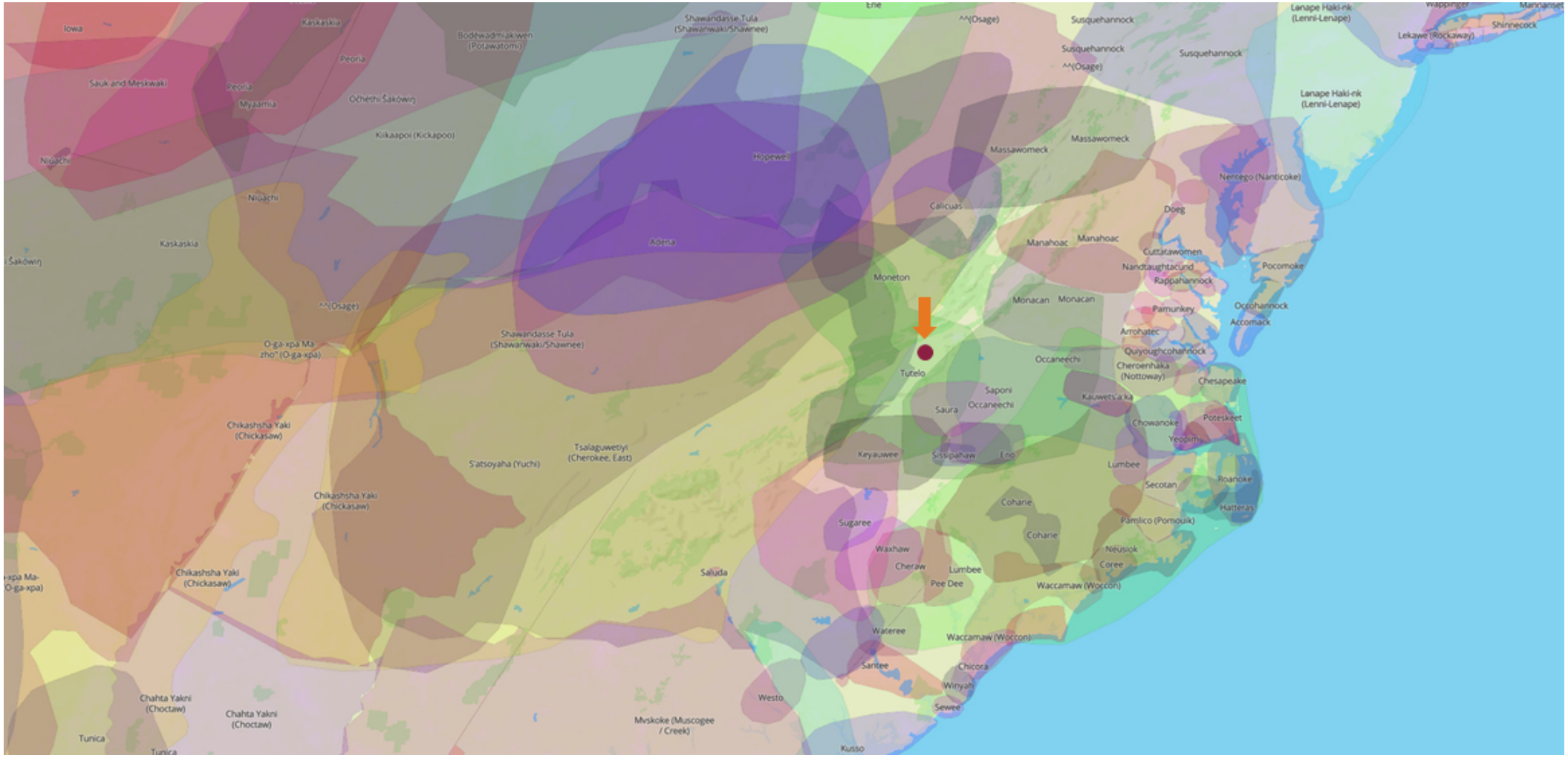
Thursday, June 9, 2022

Munsee Lenape



- https://native-land.ca/

Tutelo

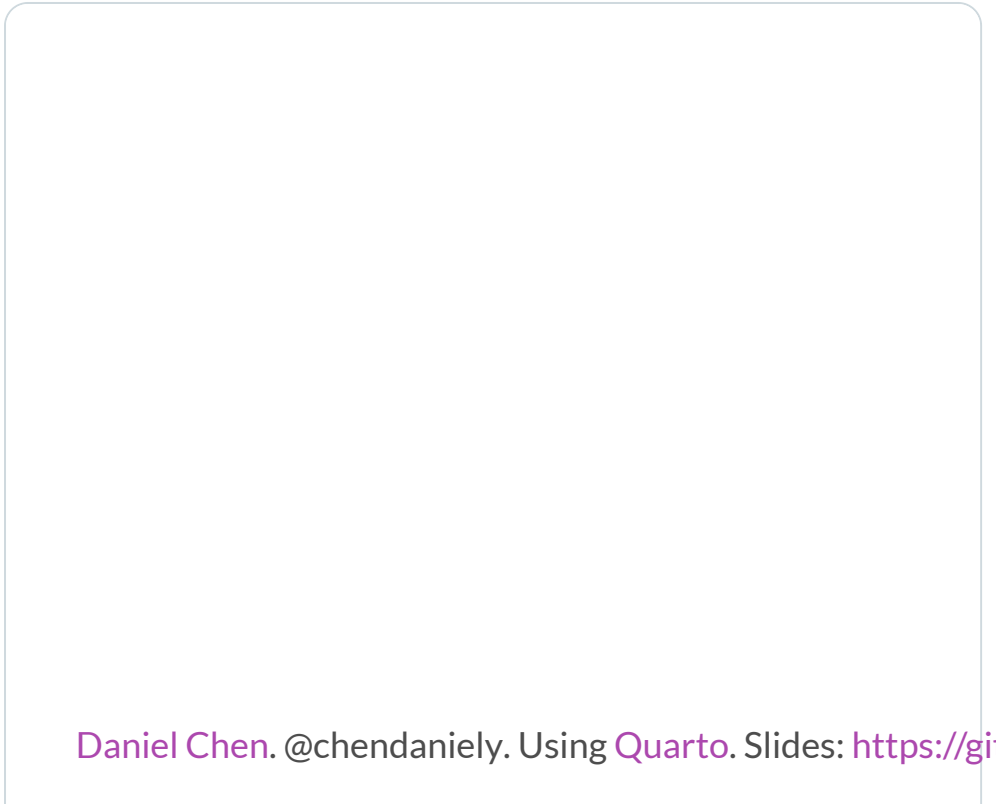


- https://native-land.ca/

Thank You

Beatriz Milz
@BeaMilz

I'm developing a presentation for @seruff_ using @quarto_pub presentations.
I started to implement a similar theme as the xaringan @RLadiesGlobal theme made by @apreshill !
If anyone wants to help to improve it, It would be awesome 💜 #rladies #RStats
github.com/quarto-dev/qua...



Daniel Chen. @chendaniely. Using Quarto. Slides: https://github.com/chendaniely/rstatsnyc-2022-analysis_presentations

I Finally Graduated...

Start Times:

- RStatsNYC: 2015

- Grad School: 2015

Daniel Chen, PhD, MPH

@chendaniely

- Postdoctoral Research and Teaching Fellow, University of British Columbia
- Data Science Educator, RStudio, PBC (RStudio Academy)
- Data Scientist, Lander Analytics
- Author, Pandas for Everyone
- Alumni
  - The Carpentries

Combining Analysis Work with Reports and Presentations

I talk about this a lot...

- Building Reproducible and Replicable Projects
- Structuring Your Data Science Projects
- Doing Data Science

... and teach it.

Constrants + Incremental Improvements

- Working with real-world constraints
- Incremental improvement
  - JD Long, Empathy in action: Building community of practice for analytics
- Time-box your learning

Back in 2019...

Building Reproducible and Replicable Projects

- **reproducibility** - the extent to which consistent results are obtained when an experiment is repeated
- **replicability** - the ability of a scientific experiment or trial to be repeated to obtain a consistent result

Important for scale

- Build more features
- Move to larger/cloud compute
- Collaborate with other people

Analysis Project Structure

Folder Setup

OPEN ACCESS Freely available online

PLOS COMPUTATIONAL BIOLOGY

Education

A Quick Guide to Organizing Computational Biology Projects

William Stafford Noble[1,2]*

1 Department of Genome Sciences, School of Medicine, University of Washington, Seattle, Washington, United States of America, **2** Department of Computer Science and Engineering, University of Washington, Seattle, Washington, United States of America

Introduction

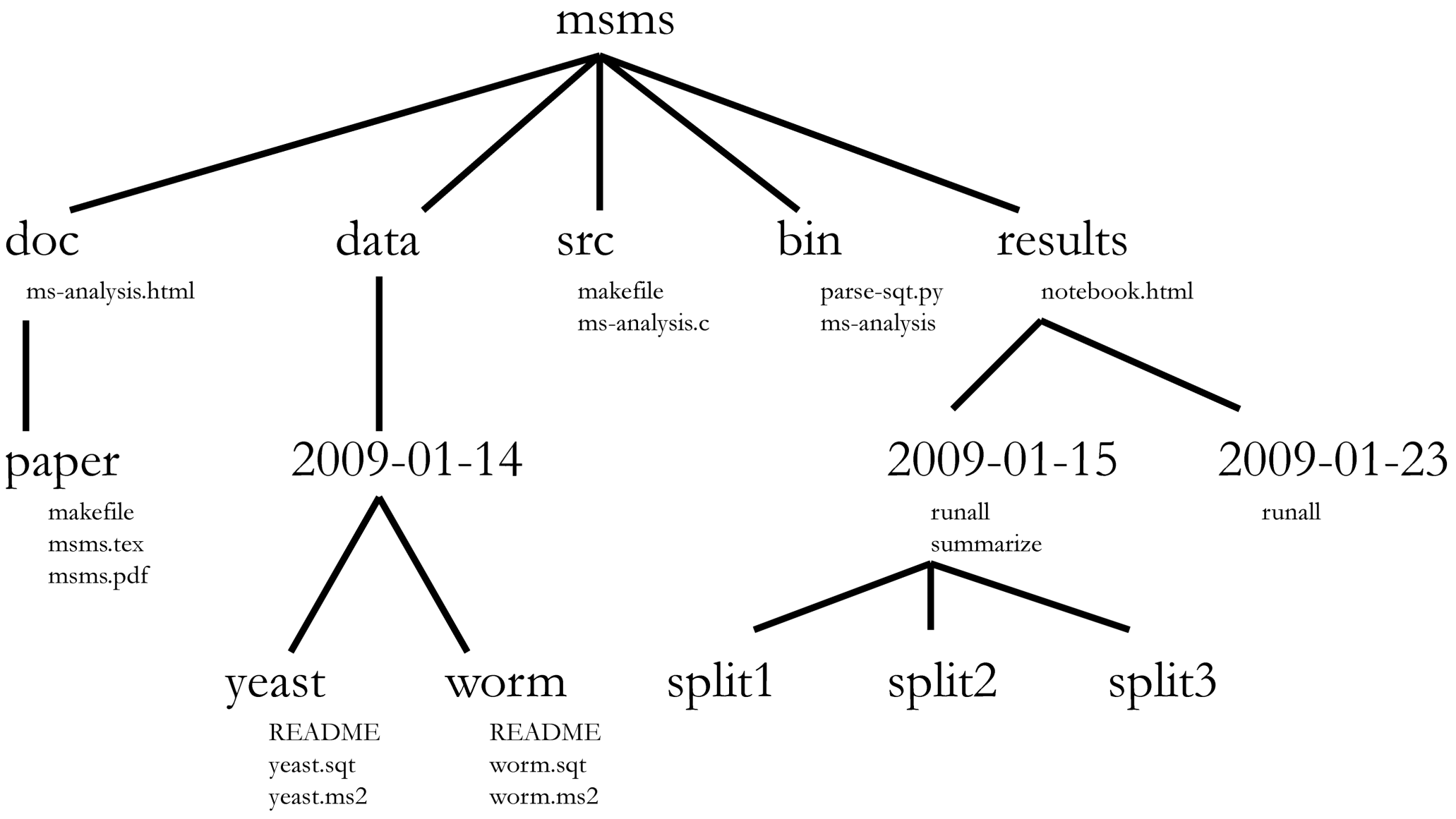
Most bioinformatics coursework focuses on algorithms, with perhaps some components devoted to learning programming skills and learning how to use existing bioinformatics software. Unfortunately, for students who are preparing for a research career, this type of curriculum fails to address many of the day-to-day organizational challenges as-

understanding your work or who may be evaluating your research skills. Most commonly, however, that "someone" is you. A few months from now, you may not remember what you were up to when you created a particular set of files, or you may not remember what conclusions you drew. You will either have to then spend time reconstructing your previous experiments or lose whatever insights you gained from those experiments.

under a common root directory. The exception to this rule is source code or scripts that are used in multiple projects. Each such program might have a project directory of its own.

Within a given project, I use a top-level organization that is logical, with chronological organization at the next level, and logical organization below that. A sample project, called `msms`, is shown in Figure 1. At the root of most of my projects, I have a

Folder Setup



Dissertation Analysis

Repo: https://github.com/chendaniely/dissertation-analysis

Project/Packaging

```
├── LICENSE
├── R
│   ├── get_survey.R
│   ├── gg_plot_dendro.R
│   ├── likert.R
│   ├── offset_number.R
│   ├── plot_question_bar.R
│   ├── question_str_to_int.R
│   ├── recode_occupation.R
│   ├── remove_duplicate_ids.R
│   ├── remove_identifiers.R
│   ├── remove_invalid_rows.R
│   ├── save_analysis_edt.R
│   ├── strip_html.R
│   └──
survey_q_multi_choice_multi_answer.R
├── README.md
├── dissertation-analysis.Rproj
```

Analysis

```
├── analysis
│   ├── 010-qualtrics
│   ├── 020-validation
│   ├── 030-persona
│   ├── 040-workshop
│   ├── 050-exercises
│   └── emails
├── build
│   ├── 00-clean.R
│   ├── 01-download_process_qualtrics.R
│   └── 10-persona.R
├── data
│   └── original
├── output
│   ├── exercises
│   ├── persona
│   ├── survey
│   └── validation
```

Naming Things

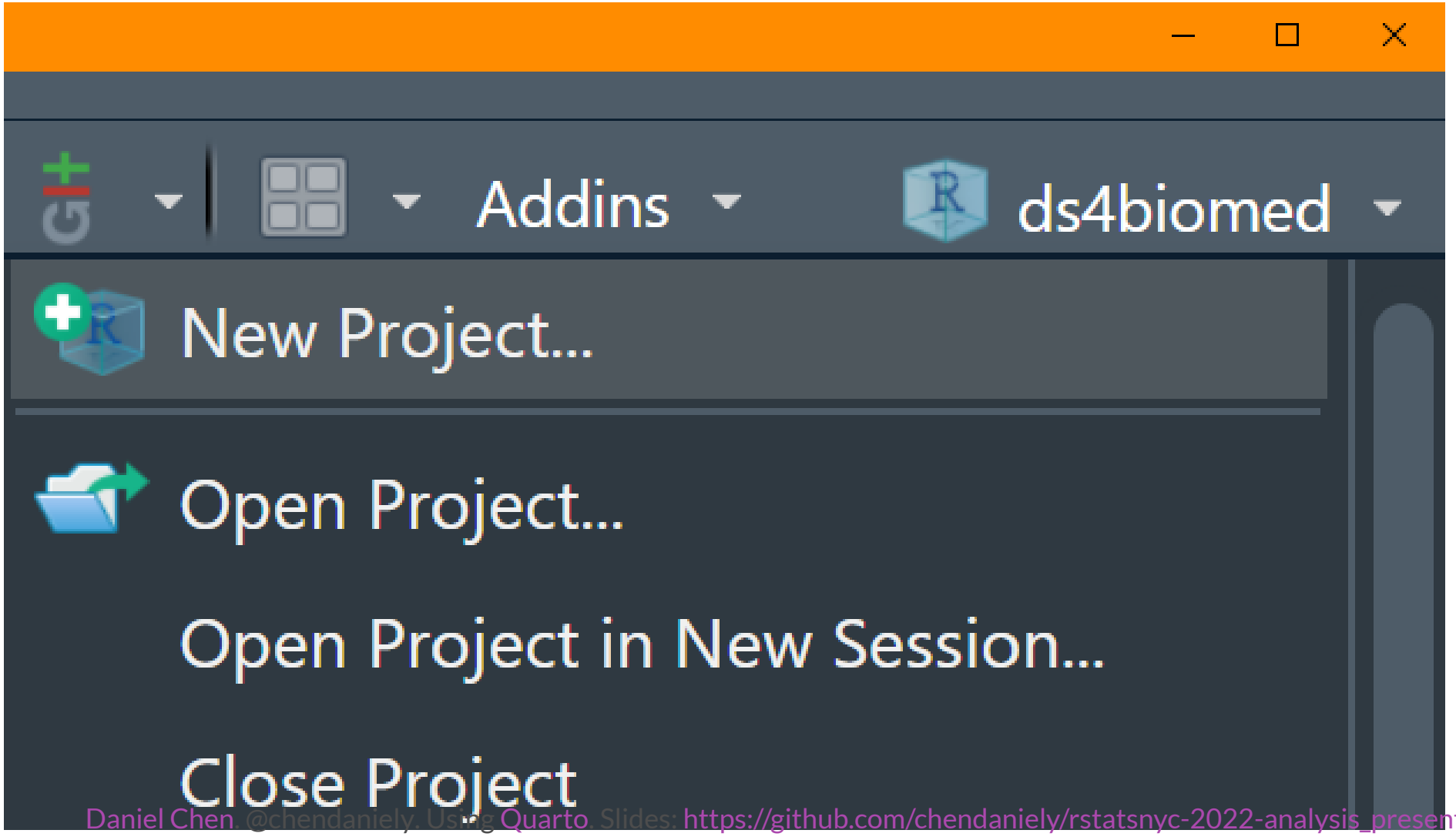
- Karl Broman, "Steps toward reproducible research"
- Jenny Bryan: Naming Things

1. Machine readable
   - Deliberate user of _ and -
2. Human readable
   - contains info on content
3. Plays well with default ordering
   - ISO 8601
   - Left pad with `0`

```
> fs::dir_tree("analysis/020-validation/")
analysis/020-validation/
├── 010-prep_survey_questions.R
├── 020-005-fa.Rmd
├── 020-010-cronbah.Rmd
└── 030-cart.Rmd
```

Project File

- https://rstats.wtf/project-oriented-workflow.html



Working Directories

- https://rstats.wtf/safe-paths.html

Only commiting the "final" artifacts

- So the reports and presentations can always refer to things I need
- This is the stuff that goes into your `output` or `results` folder
  - Other projects / repositories have a consistent place to look for latest artifacts (e.g., images, tables, etc)
  - `data/final`: for final datasets

```
> fs::dir_tree("output/", recurse=1)
output/
├── exercises
│   ├── score_prop-ex-treatment-facet_pre-combine_treatments.png
│   ├── score_prop-ex-treatment-facet_pre.png
│   ├── score_prop-ex-treatment-no_facet-combine_treatments.png
│   ├── score_prop-ex-treatment-no_facet.png
│   ├── score_prop-ex-treatment-pre100.png
│   ├── score_prop-ex-treatment.png
│   └── time_to_complete-ex-treatment-no_facet-combine_treatments.png
├── persona
│   ├── efa_eigen_scree.png
│   ├── efa_eigen_scree_good.png
│   ├── efa_item_correlations.png
```

```
│   ├── likert_only
│   ├── survey_likert
│   └── survey_only
```

Analysis Code Development

Testing...

- Controversial: I didn't write unit tests for my actual functions
  - this wasn't that practical
- Instead, I wrote code to verify my data
  - `stopifnot`, `{testthat}`, `{checkmate}`
- Research Software Engineers
  - This is an analysis project, not software

```
1  x <- 42
2  stopifnot(x == 42)
```

```
1  stopifnot(x == 525600)
```

```
Error: x == 525600 is not TRUE
```

Pinning your package versions

- `renv`
- `renv::init()` to create the `renv.lock` file
- `renv::snapshot()`: to update the lockfile
- `renv::restore()`: to restore the packages

```
├── renv
│   ├── activate.R
│   └── settings.dcf
└── renv.lock
```

```
    "tidyverse": {
      "Package": "tidyverse",
      "Version": "1.3.1",
      "Source": "Repository",
      "Repository": "CRAN",
      "Hash":
"fc4c72b6ae9bb283416bd59a3303bbab"
    },
```

What I would Improve

- Actually use the R package structure
- I know I was asking to get my computer set on fire!
- But the structure was there

```
source(here("./R/remove_identifiers.R"))
source(here("./R/remove_invalid_rows.R"))
source(here("./R/remove_duplicate_ids.R"))
source(here("./analysis/010-qualtrics/survey_search_names.R"))
```

DESCRIPTION to fake a package

```
Type: Project
Package: dissertation-analysis
Title: Dissertation Analysis
Version: 0.0.0.9000
Authors@R:
    person("Daniel", "Chen",
           "chendaniely@gmail.com",
           role = c("aut", "cre"))
Description: Dissertation Analysis
Imports:
    tidyverse,
    qualtRics
Suggests:
    testthat (>= 3.0.0)
Config/testthat/edition: 3
Encoding: UTF-8
LazyData: true
```

Load Your Functions

```
1 pkgload::load_all()
```

Rig to manage Multiple R versions

https://github.com/r-lib/rig

- Already built-in to windows
- RSwitch isn't maintained
- Linux now has *something*

My Build Enviornemnt

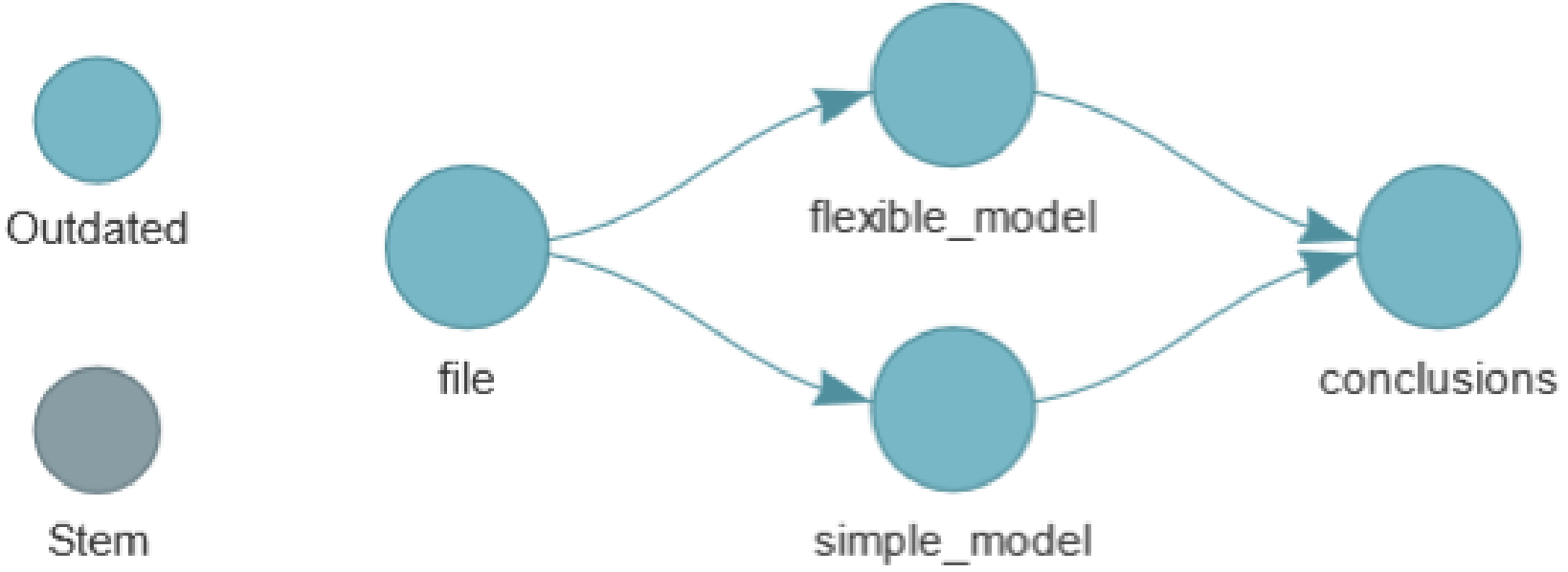
What I did:

- Write an R script that runs all my parameterized reports with different `params`
- All I had were R scripts
- Not the worst thing in the world, I didn't mind re-running the entire pipeline

Improving the Build Enviornment

What I would change + look into

- `{targets}`: https://docs.ropensci.org/targets/



Other project repositories

Don't keep everything in a mono repo

1. ds4biomed book: https://github.com/chendaniely/ds4biomed/
2. IRB: https://github.com/chendaniely/dissertation-irb/
3. Initial Plan: https://github.com/chendaniely/dissertation-plan
4. Presentations: https://github.com/chendaniely/dissertation-presentations
5. Prelims: https://github.com/chendaniely/dissertation-prelim
6. Submitted Paper: https://github.com/chendaniely/dissertation-paper-03-assessment
7. Actual Dissertation (EDT): https://github.com/chendaniely/dissertation-edt
8. etc…

Bookdown

- ds4biomed book: https://github.com/chendaniely/ds4biomed/
- Initial Plan: https://github.com/chendaniely/dissertation-plan
- Prelims: https://github.com/chendaniely/dissertation-prelim
  - Also the submitted word document

Mixed + Non-reproducible formats

- IRB: https://github.com/chendaniely/dissertation-irb/
- Presentations: https://github.com/chendaniely/dissertation-presentations

Collaborating with + without Git

Dissertation EDT: https://github.com/chendaniely/dissertation-edt

Workflow:

- Git repo + Local writing/Compiling
- Collaboration / Feedback via Overleaf + git integration
- Pull / Push to sync changes
- Still had to manually move over artifacts (overleaf limitation)

Use Child Documents

- LaTeX
- RMarkdown
- Quarto

`main.tex`:

```
\usepackage{subfiles}

\begin{document}
    \chapter{Introduction}
        \label{ch:introduction
        \subfile{010-intro/010
\end{document}
```

`010-intro/010-intro.tex`:

```
\documentclass[../main.tex]{subfiles}
\begin{document}

    \subfile{010-010-intro}

    \section{History of Data Science}
        \label{se:intro-ds-history}

        Statistics, data science, and computatio
\end{document}
```

`010-intro/010-010-intro.tex`:

```
\documentclass[010-intro.tex]{subfiles}
\begin{document}

This dissertation describes the current state of
and explores ways to improve data science educat
...
\end{document}
```

https://github.com/chendaniely/dissertation-edt/blob/main/main.tex

Using a Build file

- Do not forget what commands I need to run
- Especially compiling latex bibliography

```
LATEX=xelatex
BIBTEX=bibtex
STEM=main

all : commands

## commands   : show all commands.
commands :
    @grep -E '^##' Makefile | sed -e 's/## //g'

## counts      : get tex word counts
counts :
    find . -type f -name "*.tex" | xargs texcount 2>/dev/null | grep -w "Words in text:"
| cut -d : -f 2 | awk '{Total=Total+$$1} END {print "Total is: " Total}'

## pdf        : re-generate PDF
pdf :
    ${LATEX} -synctex=1 -interaction=nonstopmode ${STEM}
```

https://github.com/chendaniely/dissertation-edt/blob/main/Makefile

Make commands

```
% make
commands    : show all commands.
counts      : get tex word counts
pdf         : re-generate PDF
clean       : clean up junk files.
sync        : sync overleaf -> local -> GitHub
push        : push local to GitHub and Overleaf
fetch       : fetch remotes origin + leaf
```

What I would change

- Quarto
- knitr + LaTex Rnw files
- Challenge: Still need to keep the original LaTeX Source
  - even though Overleaf can do basic R computation

Keeping Everything Together

All these seprate repositories

- Keep them together
- Sometimes I want to reference things with relative paths
- Maybe an overall project to tie things together

Git Submodules

https://github.com/chendaniely/dissertation

- "Nest" git repository

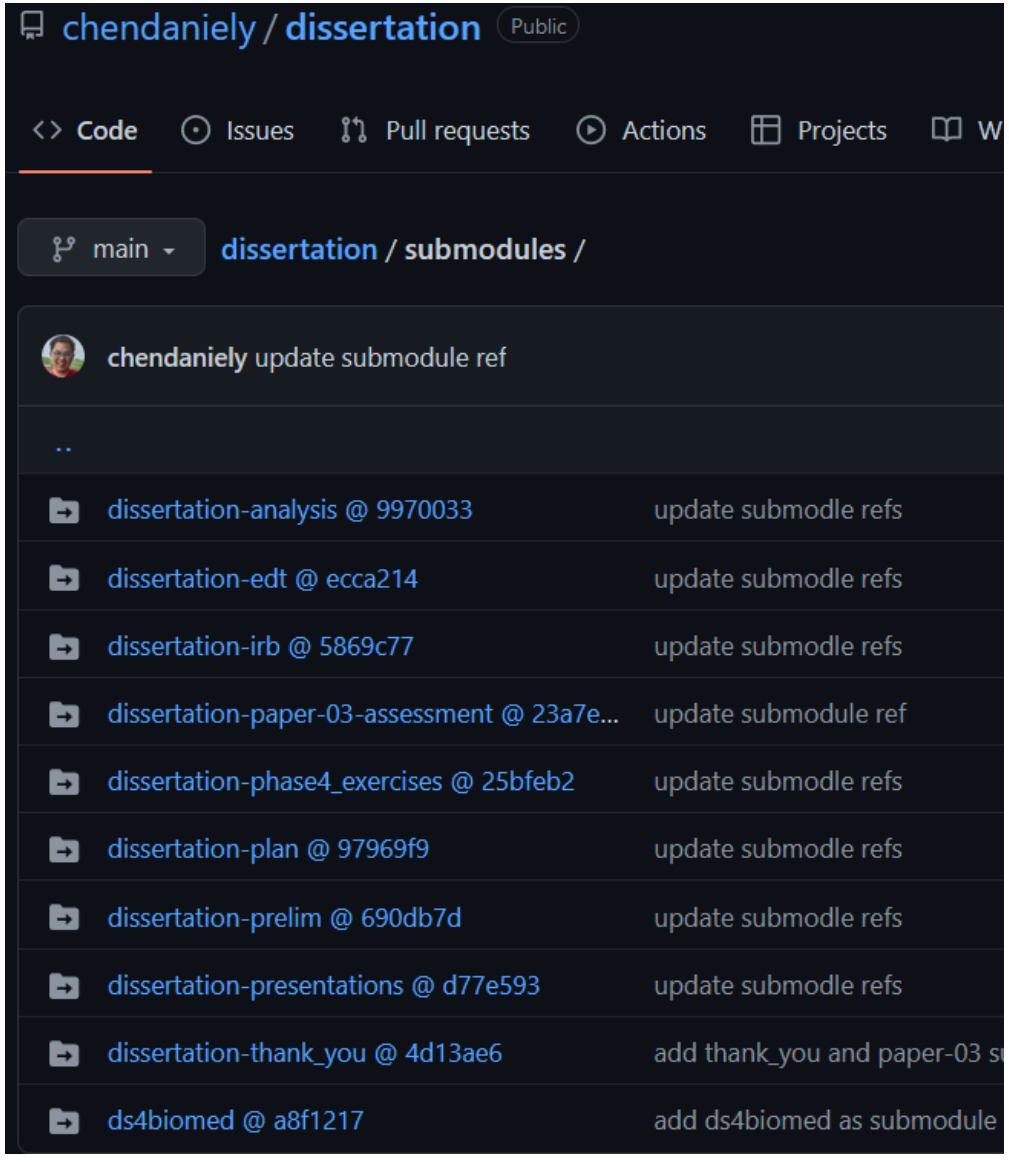
```
> fs::dir_tree(".", recurse=1)
.
├── 010-intro.md
├── README.md
├── dissertation.Rproj
└── submodules
    ├── dissertation-analysis
    ├── dissertation-edt
    ├── dissertation-irb
    ├── dissertation-paper-03-assessment
    ├── dissertation-phase4_exercises
    ├── dissertation-plan
    ├── dissertation-prelim
    ├── dissertation-presentations
    ├── dissertation-thank_you
    └── ds4biomed
```

```
> fs::dir_tree(".", recurse=2)
.
├── 010-intro.md
├── README.md
├── dissertation.Rproj
└── submodules
    ├── dissertation-analysis
    │   ├── LICENSE
    │   ├── R
    │   ├── README.md
    │   ├── analysis
    │   ├── build
    │   ├── data
    │   ├── dissertation-analysis.Rproj
    │   ├── output
    │   ├── renv
    │   └── renv.lock
    ├── dissertation-edt
```

Git Submodules: Mental Model

- Each sub repo is a regular git repository
  - `add` / `commmit` / `push` / `pull`
  - is exactly the same
- Every time you `push`/`pull`/`commit` from the submodule
  - go back to the main parent repo
  - update the commit references with `add`, `commit`, and `push`
- In the main repo, it's really just tracking the latest tracked commit
  - it doesn't automatically keep up with `main`

Parent repo only tracking the commit



Experiments

Running Shiny experiments

https://github.com/chendaniely/dissertation-phase4_exercises

What I would change

- `{shinysurveys}`: https://cran.r-project.org/web/packages/shinysurveys/index.html
- https://www.rstudio.com/resources/rstudioglobal-2021/designing-randomized-studies-using-shiny/
- A better way to capture learnr submissions
- Learnr + gradethis isn't really made for this kind of work?

In Sum: What I would change

1. DESCRIPTION to fake a package
2. Rig to manage Multiple R versions
3. `{targets}` + `Makefile`: Improving the Build Environment
4. Quarto for my websites, books, and presentations
    - maybe even the final EDT?
5. `{shinysurveys}`: to do better block randomization

Thanks!