Reproducible Computational Research

Mark Stenglein, MIP 280A4

Components of reproducible computational research

- Data sharing
- Method documentation and transparency
- Code sharing and documentation
- Code portability and reproducibility
- Code version control
- Software dependency management

Some good resources

Pat Schloss

https://riffomonas.org/reproducible_research/

OPEN ACCESS Freely available online

PLOS | COMPUTATIONAL BIOLOGY

Editorial

Ten Simple Rules for Reproducible Computational Research

Geir Kjetil Sandve[1,2*], Anton Nekrutenko[3], James Taylor[4], Eivind Hovig[1,5,6]

OPEN ACCESS Freely available online

PLOS | COMPUTATIONAL BIOLOGY

Editorial

Ten Simple Rules for Reproducible Computational Research

Geir Kjetil Sandve[1,2]*, Anton Nekrutenko[3], James Taylor[4], Eivind Hovig[1,5,6]

Rule 1: For Every Result, Keep Track of How It Was Produced

Rule 2: Avoid Manual Data Manipulation Steps

Rule 3: Archive the Exact Versions of All External Programs Used

Rule 4: Version Control All Custom Scripts

Rule 5: Record All Intermediate Results, When Possible in Standardized Formats

Rule 6: For Analyses That Include Randomness, Note Underlying Random Seeds

Rule 7: Always Store Raw Data behind Plots

Rule 8: Generate Hierarchical Analysis Output, Allowing Layers of Increasing Detail to Be Inspected

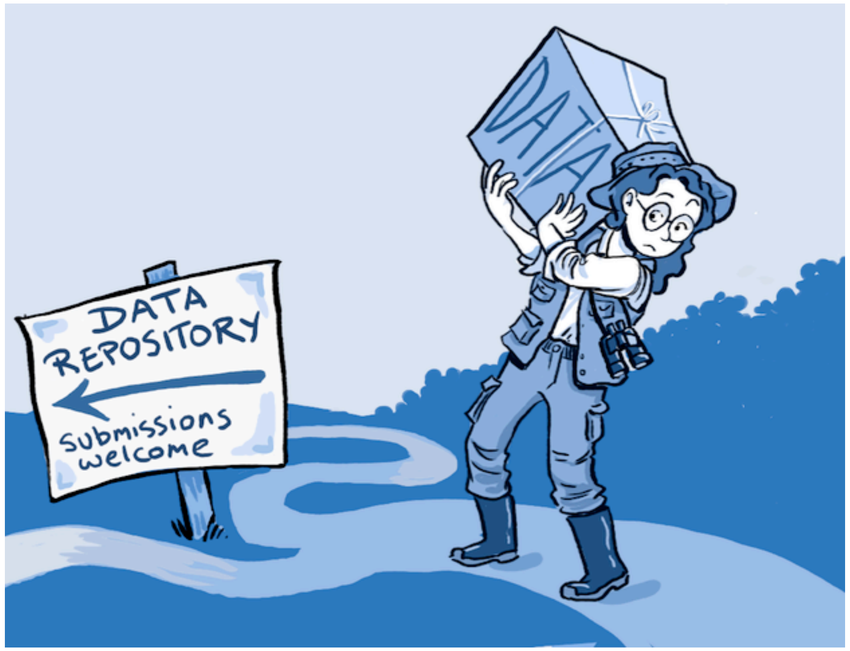
Rule 9: Connect Textual Statements to Underlying Results

Rule 10: Provide Public Access to Scripts, Runs, and Results

Making your data available to others is critical

Best practices for data sharing:

- Deposit NGS data in the SRA database
- Deposit assembled sequences in Genbank or Assembly databases
- Put other important data (multiple alignments, tree files, etc.) in supplemental material and/or a Github repository



Roche et al (2014) PLoS Biology

Like in typical lab research, it is critical that you document what you are doing when performing computational analyses

In other words, keep a “computational lab notebook”

Image: Katherine Stember, CC-BY-SA-4.0, Link

```
---
title: "R Notebook"
output:
  html_document:
    df_print: paged
editor_options:
  chunk_output_type: inline
---

This notebook is for the in vitro transcribed minigenome cocktail sequencing analysis.

### Backgroung/Purpose
BSR T7/5 cells were transfected with Kairi L + N helper plasmids (T7 backbone) and 24 hours later transfected with IVT M segment r
at 24, 48, and 72 hours post-minigenome transfection. Harvested cells were split 80/20 for sequencing analysis and dual-luciferas
sequencing analysis.

The IVT M segment minigenome cocktail contains:
- ivt.mg. 1: Kairi M
- ivt.mg.3: Kairi M negative
- ivt.mg.4: Cache valley M
- ivt.mg.6: Cache valley M negative
- ivt.mg.11: Boaraceia M
- ivt.mg.13: Brazoran M
- ivt.mg.14: Gamboa M
- ivt.mg.15: LaCrosse M
- ivt.mg.17: Oropouche M

### Analysis
#### Sequencing
Pooled libraries were sequenced on the MiSeq with a v2 300 cycle nano kit with 2x151 reads and 2x8 bp indexes. The run directory
/home/ngsdata/210901_M03520_0432_000000000-DCV9B
##### FASTQC
Fastq files and analysis are on the cctsi-103 server here: /home/mllayton/analyses/2021_09_01_minigenome_ivt_cocktail

Reads were quality filtered and collapsed for PCR duplicate using the run_preprocessing_pipeline. Specifically, FASTQC and cutadal
```{r}
library(tidyverse)
library(readxl)
fastq_data <- read_excel("./fastqc_read_counts.xlsx")
fastq_data
fastq_data_tidy <- fastq_data %>%
```
```

Computational lab notebooks can take many forms

- Handwritten, like a traditional lab notebook
- A word or google doc document
- A plain text file
- A “markdown” format file
- Shell scripts

It is more important that you keep track of what you did than how exactly you do so

The target audience of these documents are not necessarily other people.
You are often your own target audience. (How did I do that one thing 6 months ago?)

Markdown format is a easy way to make nice-looking documents

Markdown format is just a plain text format

```
## Variant calling exercise

MIP 280A4
---

### In this exercise, we will go through a variant calling pipeline.

The data we'll use was generated from a pool of _Anopheles gambiae_ mosquitoes collected in Liberia.  The dataset contains re
ads mapping to a virus (Anopheles flavivirus).  We'll map the reads to the viral genome and use a variant caller to identify
variants and determine their frequencies.

We will:

- download the files we need
- make a bowtie index from the viral genome (reference) sequence
- map reads to the reference sequence using bowtie2
- use lofreq to call variants
- visualize and inspect variants in vcf output file and in Geneious

### Download files needed for exercise from github

First, we need to download some files.

Open the terminal and change directory to a directory of your choosing (make a new directory if you'd like).

The files for this exercise are in the directory `/home/data_for_classes/2021_MIP_280A4/`.  You need to copy this file into y
our current directory.

```
```
```

Plain-text markdown can be rendered into a nice looking version

Variant calling exercise

MIP 280A4

In this exercise, we will go through a variant calling pipeline.

The data we'll use was generated from a pool of *Anopheles gambiae* mosquitoes collected in Liberia. The dataset contains reads mapping to a virus (Anopheles flavivirus). We'll map the reads to the viral genome and use a variant caller to identify variants and determine their frequencies.

We will:

- download the files we need
- make a bowtie index from the viral genome (reference) sequence
- map reads to the reference sequence using bowtie2
- use lofreq to call variants
- visualize and inspect variants in vcf output file and in Geneious

Download files needed for exercise from github

First, we need to download some files.

Markdown is simple plain text syntax that maps to nice-looking output formats

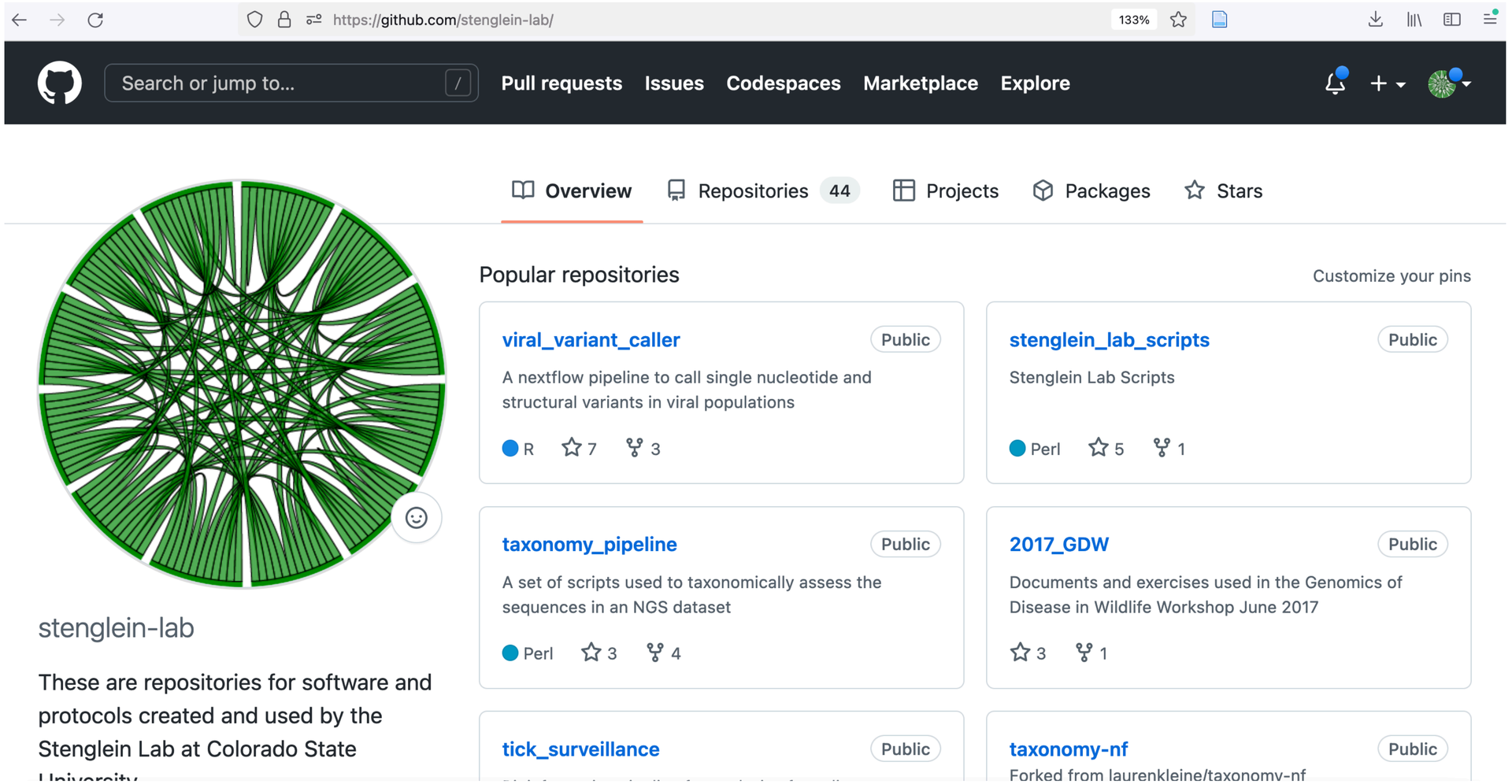
https://www.markdownguide.org/basic-syntax/

Headings

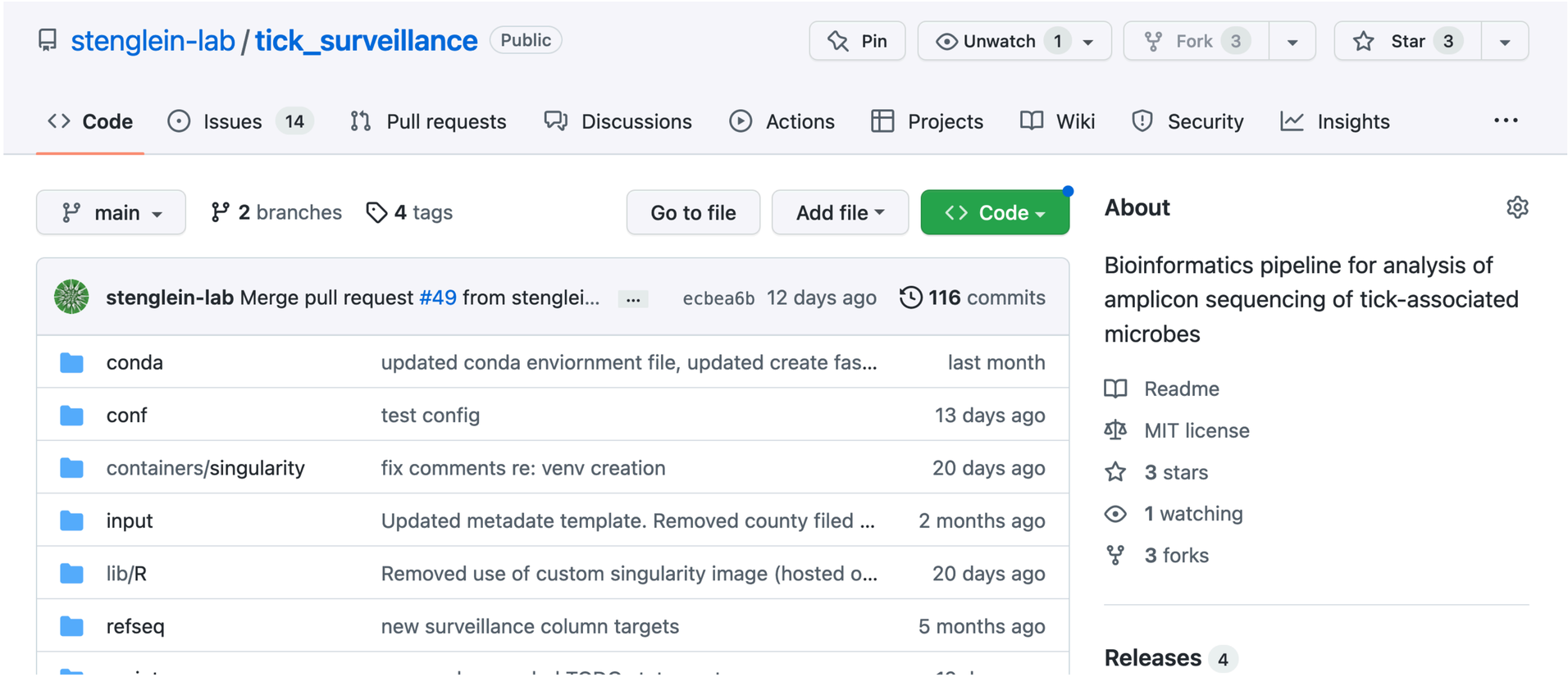
To create a heading, add number signs (#) in front of a word or phrase. The number of number signs you use should correspond to the heading level. For example, to create a heading level three (<h3>), use three number signs (e.g., `### My Header`).

Markdown	HTML	Rendered Output
`# Heading level 1`	`<h1>Heading level 1</h1>`	Heading level 1
`## Heading level 2`	`<h2>Heading level 2</h2>`	Heading level 2
`### Heading level 3`	`<h3>Heading level 3</h3>`	Heading level 3
`#### Heading level 4`	`<h4>Heading level 4</h4>`	Heading level 4
`##### Heading level 5`	`<h5>Heading level 5</h5>`	Heading level 5
`###### Heading level 6`	`<h6>Heading level 6</h6>`	Heading level 6

Github is the de facto standard for sharing code (and data and lots of other things) these days



This repository contains a pipeline to analyze NGS data



This repository (repo) contains code and data needed to make one figure

https://github.com/stenglein-lab/highlight_residues_on_spike_structure

Code | Issues | Pull requests | Actions | Projects | Wiki | Security | Insights | Settings

main | 1 branch | 0 tags | Go to file | Add file | Code

Mark Stenglein and **Mark Stenglein** background color... da2f119 on Feb 28, 2021 | 4 commits

.gitignore	Initial commit	2 years ago
6VXX.afasta	initial commit	2 years ago
6vxx.pdb	initial commit	2 years ago
Figure_X_highlighted_re...	background color white	2 years ago
Figure_X_highlighted_re...	background color white	2 years ago
LICENSE	Initial commit	2 years ago
README.md	fix typo	2 years ago
chimera_commands.cmd	background color white	2 years ago
generate_chimera_comm...	background color white	2 years ago
residues_to_highlight.csv	initial commit	2 years ago

About

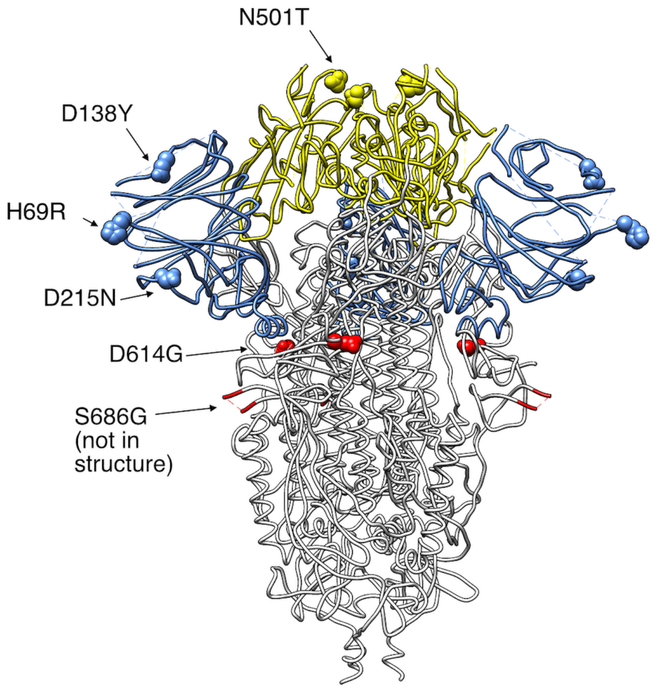
Highlight structures on SARS-CoV-2 spike structure using UCSF chimera

Readme
MIT license
1 star
1 watching
0 forks

Releases

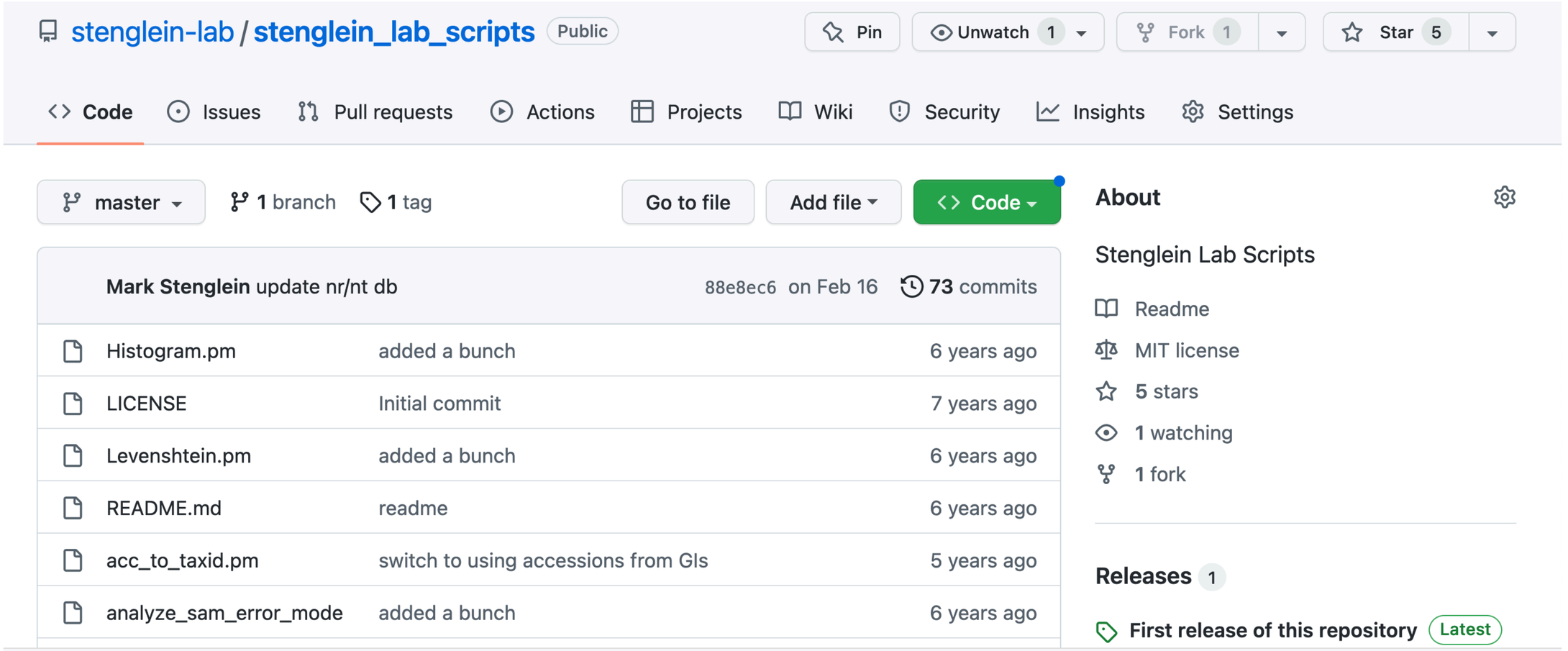
No releases published
Create a new release

Packages

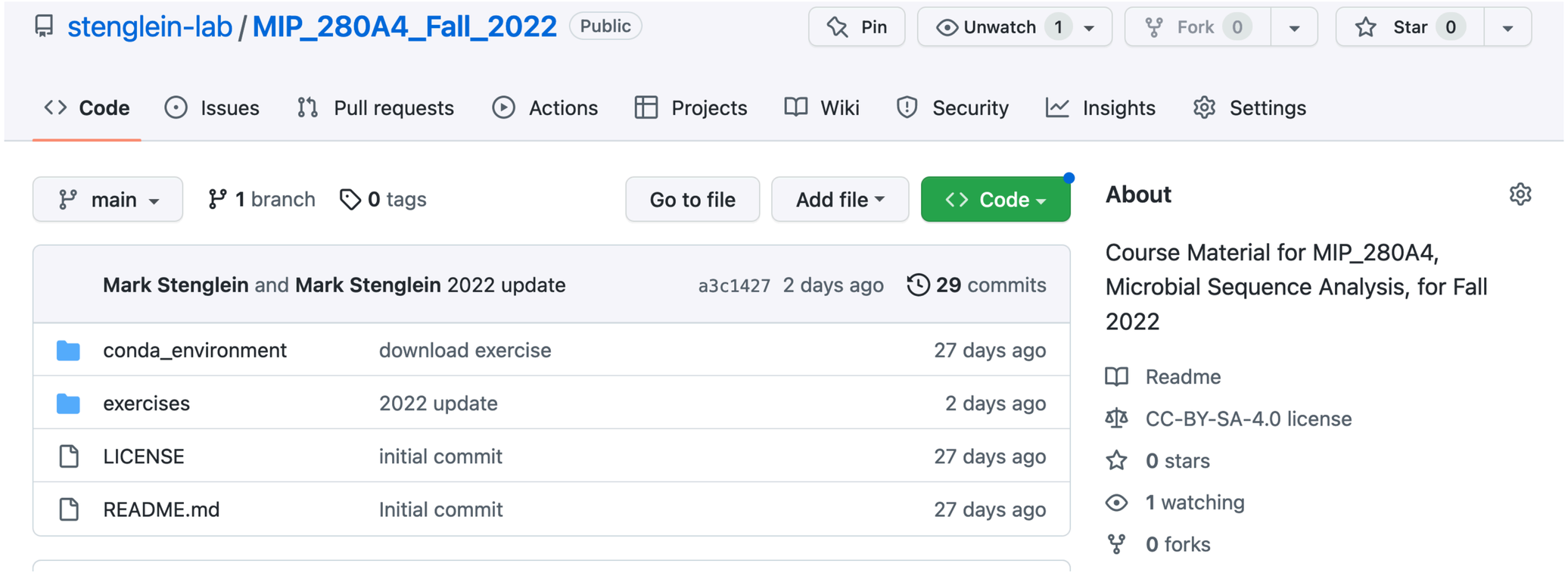


Bashor et al (2021)

This repository contains a bunch of scripts



This repository contains the exercises we've been doing in this course



A feature of git repositories is that they track changes to files (version control)

2022 update

Browse files

main

Mark Stenglein authored and **Mark Stenglein** committed 2 days ago 1 parent cb06113 commit a3c1427c84615ba41cb9a089efeae73979297639

Showing **1 changed file** with **6 additions** and **3 deletions**.

Split | Unified

9 exercises/variant_exercise.md

```
@@ -28,7 +28,7 @@ Open the terminal and change directory to a directory of your choosing (make a n
```

28	The files for this exercise are in the directory `/home /data_for_classes/2021_MIP_280A4/`. You need to copy this file into your current directory.	28	The files for this exercise are in the directory `/home /data_for_classes/2021_MIP_280A4/`. You need to copy this file into your current directory.
29		29	
30	```	30	```
31	- cp /home/data_for_classes/2021_MIP_280A4 /variant_exercise_files.tar.gz .	31	+ cp /home/data_for_classes/2022_MIP_280A4 /variant_exercise_files.tar.gz .
32	```	32	```
33		33	

A collection of changes in one or more files is called a “commit”

In this commit, I updated a URL from 2021 to 2022

Workflow managers allow you to make portable, scalable, reproducible pipelines

Portable: run on different types of computers (laptop, local server, cloud)

Scalable: run on one or many datasets or many many datasets

Flexible: change settings or variables (e.g. use a different reference sequence)

Reproducible: handle software dependencies smoothly

There are several popular workflow management systems

Snakemake

The Snakemake workflow management system is a tool to create **reproducible and scalable** data analyses. Workflows are described via a human readable, Python based language. They can be seamlessly scaled to server, cluster, grid and cloud environments, without the need to modify the workflow definition. Finally, Snakemake workflows can entail a description of required software, which will be automatically deployed to any execution environment.

There are several popular workflow management systems

Nextflow

Data-driven computational pipelines

Nextflow enables scalable and reproducible scientific workflows using software containers. It allows the adaptation of pipelines written in the most common scripting languages.

Its fluent DSL simplifies the implementation and the deployment of complex parallel and reactive workflows on clouds and clusters.

There are trade-offs to using workflows: additional work but big improvement in reproducibility

Part of a nextflow pipeline that runs fastqc on input fastq files

```
/*
 Run fastqc on input fastq
*/
process initial_qc {
  label 'many_forks'
  label 'process_low'

  // singularity info for this process
  if (workflow.containerEngine == 'singularity' && !params.singularity_pull_docker_container) {
      container "https://depot.galaxyproject.org/singularity/fastqc:0.11.9--0"
  } else {
      container "quay.io/biocontainers/fastqc:0.11.9--0"
  }

  input:
  tuple val(sample_id), path(initial_fastq) from samples_ch_qc

  output:
  path("*.zip") into post_initial_qc_ch

  script:
  """
  fastqc $initial_fastq
  """
}
```

The actual fastqc command

My advice:

Learn the basics of the tools and the command-line environment first

Then maybe start working with workflows for a pipeline you expect to use repeatedly

Package managers (Conda) or Container systems (Docker, Singularity)
Deal with software dependencies and improve portability

Conda: create environments with one or more pieces of software and all of their dependncies. What we've been using in class (bio_tools). Good reproducibility but there can be some issues (conflicts with already-installed R packages, for instance)

Docker/Singularity: Totally self-contained "container images" that have one or more software tools and their dependencies and an entire operating system. These are "virtual computers". Very portable/reproducible: even better than conda.

There is good integration between workflow managers like nextflow and tools like conda or singularity to manage software dependencies

```
process MAKE_SEGMENT_TABLE {
    tag "$tsv"
    label 'process_low'

    conda (params.enable_conda ? 'conda-forge::r-tidyverse=1.3.1' : null)
    // why aren't singularity biocontainers updated to a newer tidyverse version?
    container "${ workflow.containerEngine == 'singularity' && !task.ext.singularity_pull_docker_container ?
        'https://depot.galaxyproject.org/singularity/r-tidyverse:1.2.1' :
        'quay.io/biocontainers/r-tidyverse:1.2.1' }"
```

Some example nextflow code that includes information about docker packages or singularity containers that will be used.

Let’s make a repository!

Outline

- Make GitHub logins
- Make a repository
- Clone it to Thoth
- Update the README file
- Commit the changes.

You will document your final project for this class in this repository! And it will be the basis for my grading.