Lecture 3.8 – Documenting Your Work

Specific Learning Objectives:

2.2.2 – Include effective documentation in scripts and projects.

2.2.4 – Create and use Notebooks and documents using RMarkdown.

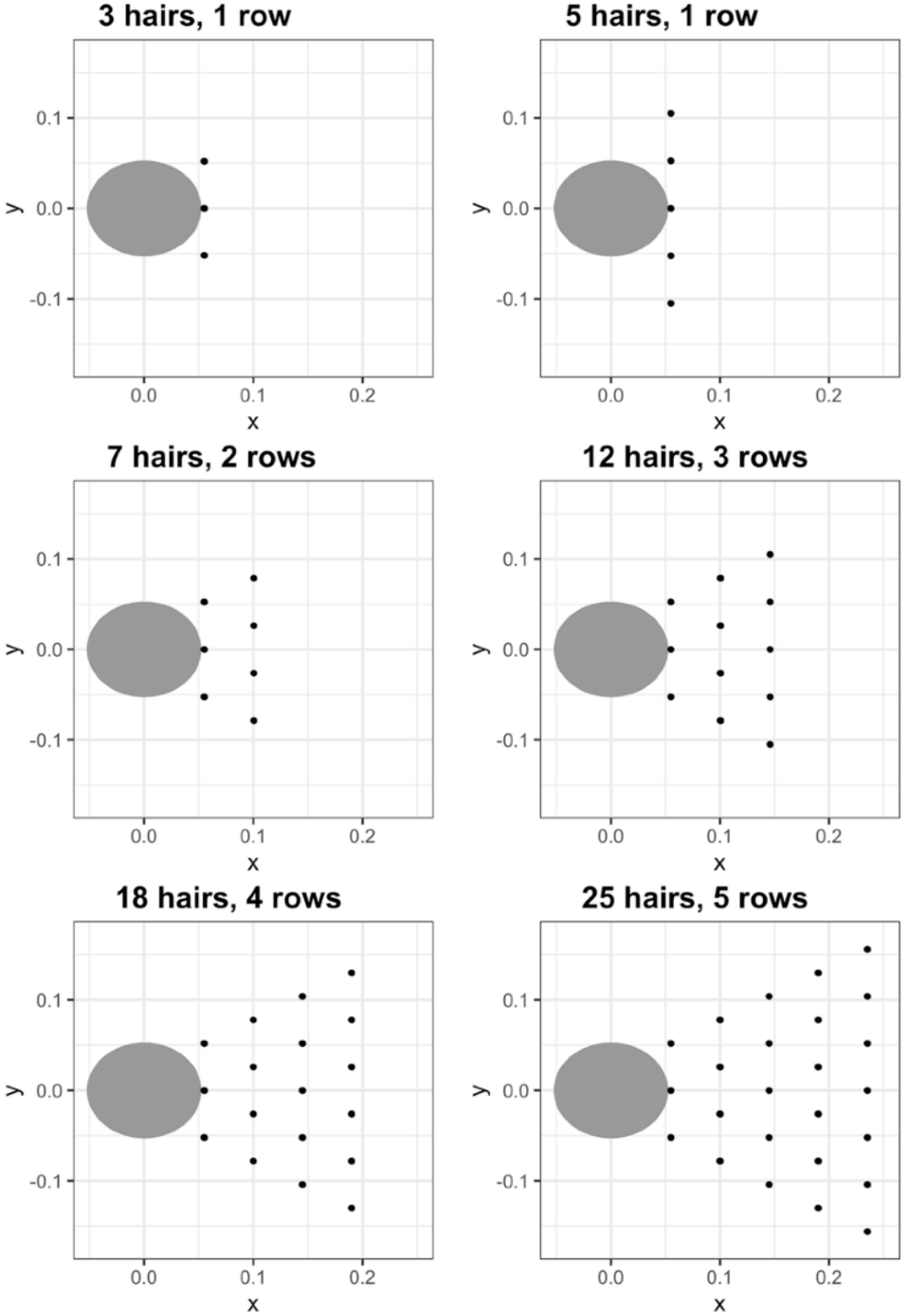
Where the rubber hits the road

- We've already talked about the importance of documentation for projects:
    - Improves reproducibility and replicability
    - Improves understanding of analysis and conclusions
    - Improves transparency, easier to catch mistakes and errors
    - Improves ability of others (including your future self) to use your work
- We've already discussed some strategies:
    - literate programming
    - R Markdown
    - R Projects

What are additional things we can do to document our projects?

An example - entcode!

- A project that my undergrads helped me complete in 2020 created hair arrays for studying odor capture by crabs and insects. It is based on code published in 2018.

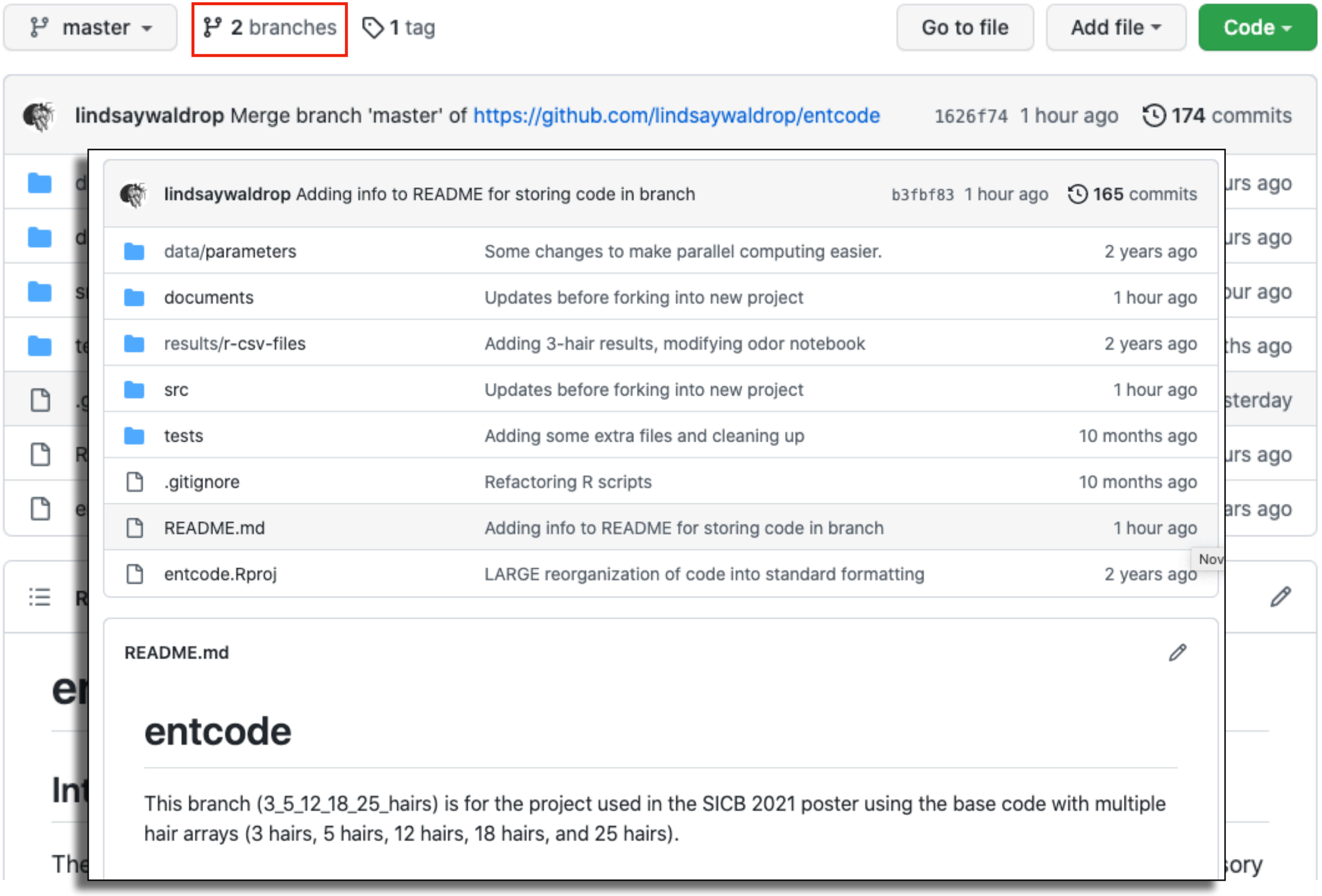


- Each student had a hair array which required them to run the same code with a bit of variation to accommodate their specific array.
- This resulted in more than **88,000 files** which needed to be organized and documented so that the data could be analyzed and stored!
- The project had internal documentation in the form of comments and descriptions, but also a pipeline document outlining flow.

https://github.com/lindsaywaldrop/entcode

How can we keep it all straight?

- Version control with git/Github:



How can we keep it all straight?

- Documentation:
  - **Comments in code** – These are notes that describe the purpose or action of specific lines or a small chunk of code, usually in place.

```
13 #### Loads required packages ####
14 library(pracma)
15 library(useful)
16 plotit <- 1
17 # plot the hairs? yes = 1, no = 0
18 startrun <- 1
19 endrun <- 1
20 nohairs <- 25  # 2 row: 7; 3 row: 12; 4 row: 18; 5 row: 25
21
```

  - **File/Function Descriptions** – These are notes that describe the purpose of specific files or functions and how to use them. Function descriptions will usually include information about arguments and outputs (think help doc!)

compile_sniff.sh | generate_grid2d_multirow.R

Source on Save | Run

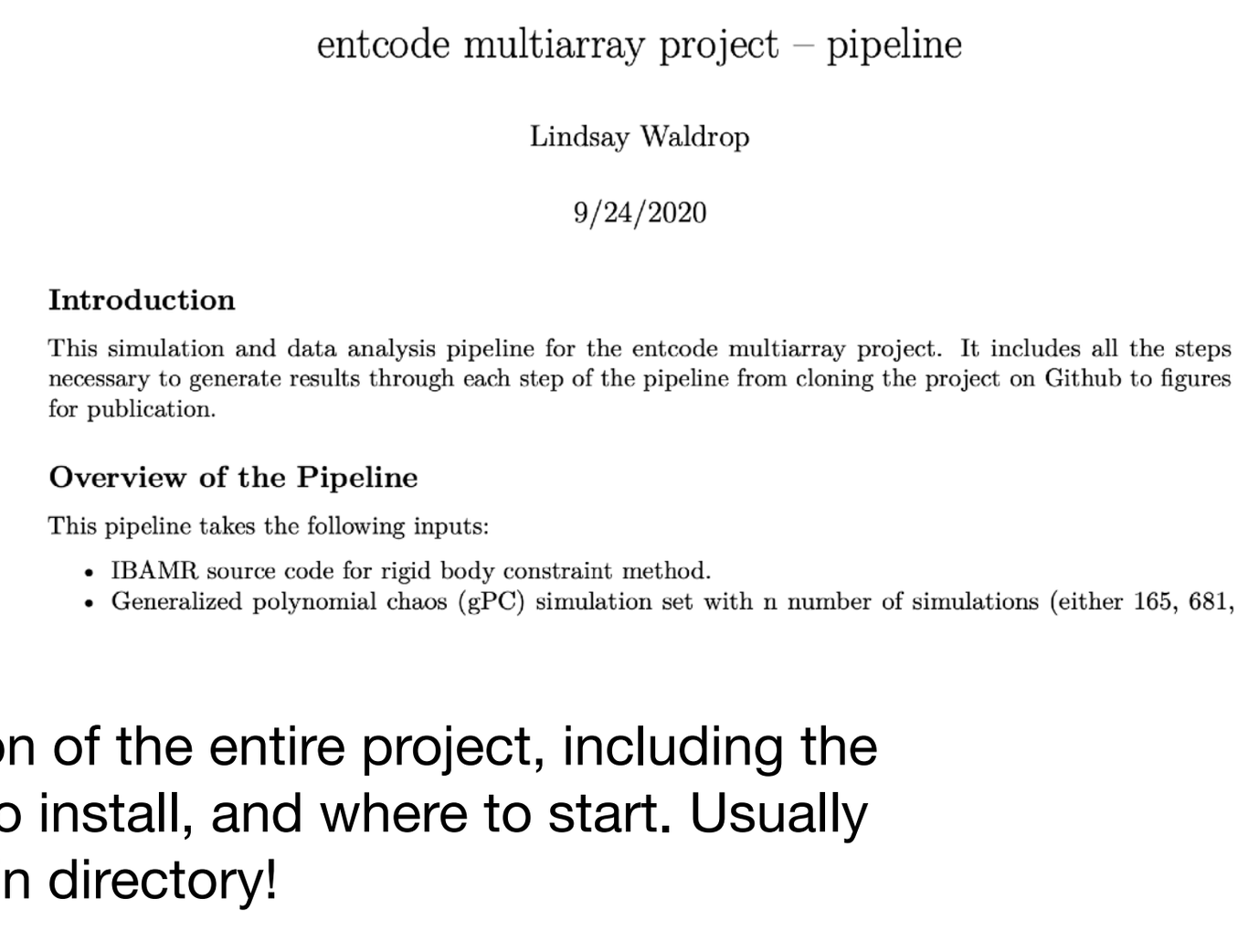
```
1  # Generates grid for use with Constraint IB Method IBAMR
2  # GtD = gap to diameter ratio between hairs
3  # dist = distance from antennule
4  # theta = angle of center hair with positive x-axis
5  #
6  # To run this R script on Bridges, enter the following commands:
7  # - module add R
8  # - R   ## This will start R!##
9  # - source('generate_grid2d.R')  ## Follow prompts if installing packages
10 # - quit()
11 # - n
12
```

How can we keep it all straight?

- Documentation:

  - **Workflow Descriptions –** Describes the necessary steps to execute the entire workflow.

    https://github.com/lindsaywaldrop/entcode/blob/master/documents/entcode-Pipeline.pdf



entcode multiarray project – pipeline

Lindsay Waldrop

9/24/2020

Introduction

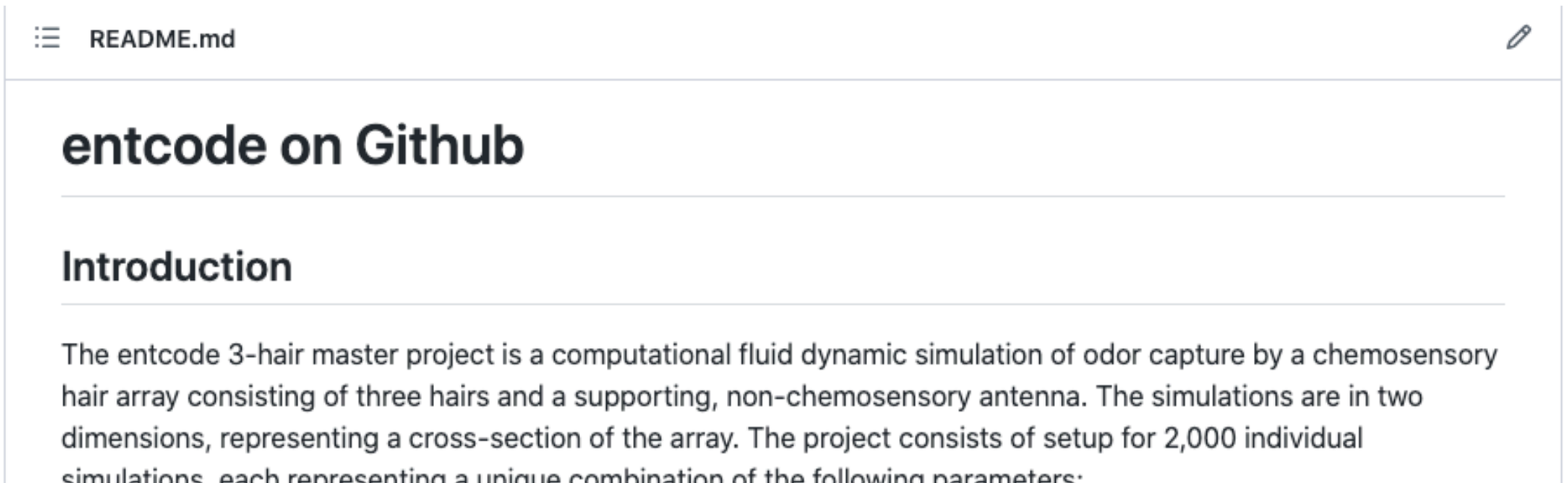
This simulation and data analysis pipeline for the entcode multiarray project. It includes all the steps necessary to generate results through each step of the pipeline from cloning the project on Github to figures for publication.

Overview of the Pipeline

This pipeline takes the following inputs:

- IBAMR source code for rigid body constraint method.
- Generalized polynomial chaos (gPC) simulation set with n number of simulations (either 165, 681,

- **Project Descriptions –** Description of the entire project, including the purpose, required software, how to install, and where to start. Usually the README document in the main directory!



README.md

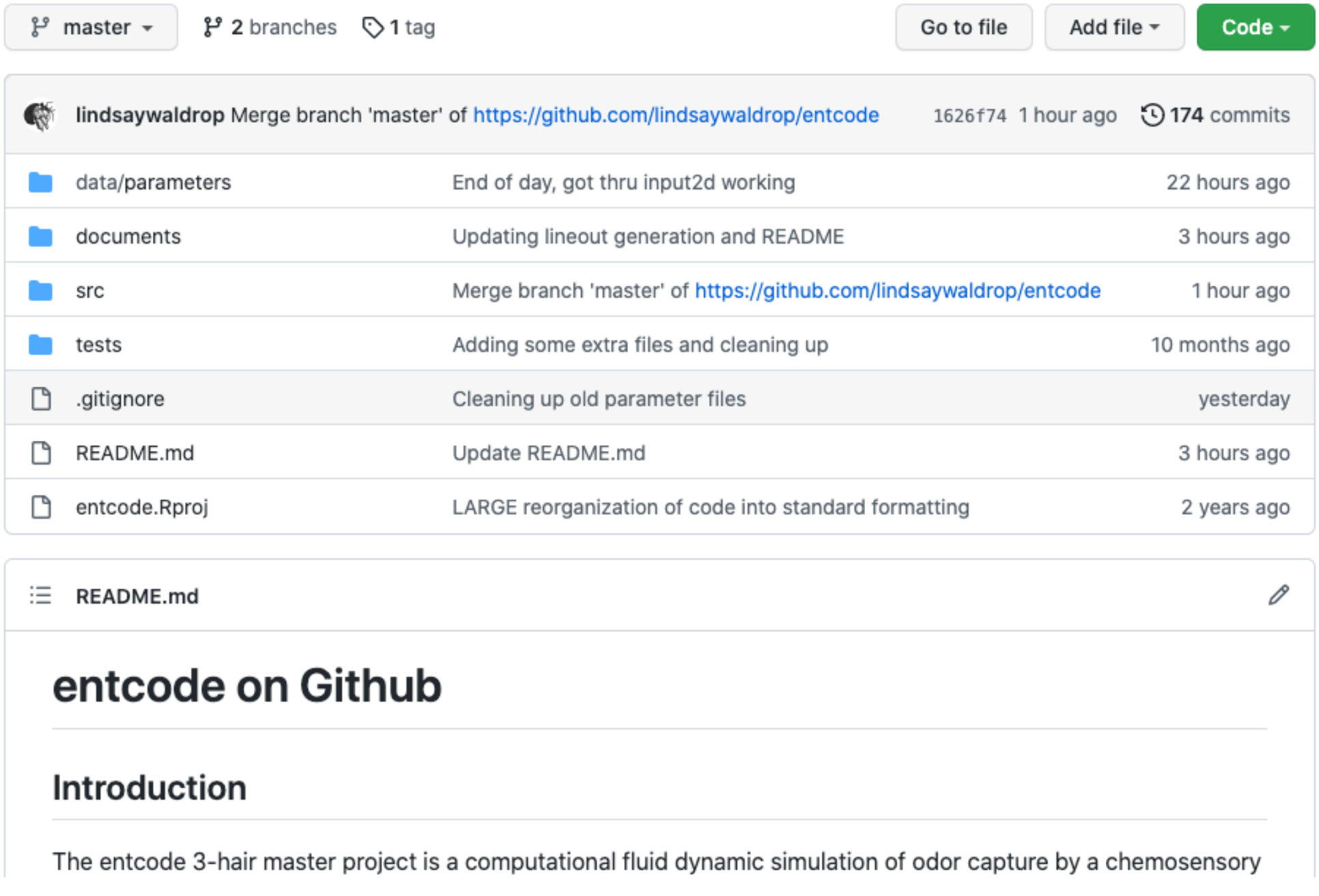
entcode on Github

Introduction

The entcode 3-hair master project is a computational fluid dynamic simulation of odor capture by a chemosensory hair array consisting of three hairs and a supporting, non-chemosensory antenna. The simulations are in two dimensions, representing a cross-section of the array. The project consists of setup for 2,000 individual simulations, each representing a unique combination of the following parameters:

How can we keep it all straight?

- Organization!
  - **Standard organization** exists to simplify both the creation of projects and the navigation of projects.



Standard Organization for R Projects

- **Standard organization** refers to the accepted way to organize parts of a project that helps orient users to your code.

MyProject

- ALL PARTS of the project belong in the main directory!

RProject file

- Established automatically when opening a new project

README file

- README files give instructions on how to run code to user and info about the project

data/

- directory containing all raw data (csv files)

src/

- directory containing all scripts (you might not have this)

results/

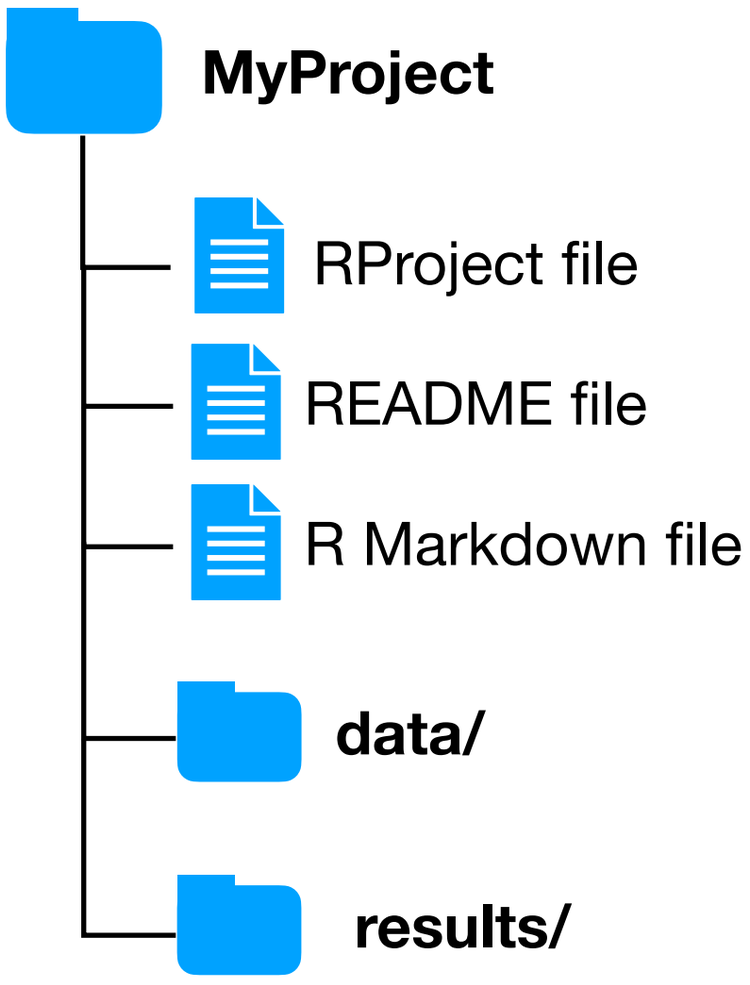
- directory containing all results produced by your code

doc/

- directory containing all documentation (including RMD files)

Standard Organization for R Projects

- **Modified standard organization** is acceptable for projects in this course!



- If you only have one R Markdown file, you can put it in the main directory. But data and results must still be separated!

In Class Exercises

1. **Take this time to reorganize your Project 2. Make it conform to standard organization (or modified standard organization).**

2. **Add some documentation to your project 2 code. It can be in-code description, function/file level descriptions, adding to a workflow and/or README.**

Action Items

1. **Complete previous assignments.**

2. **Read Davies Ch. 11 for next time.**