# test-demo Documentation

***Release 0.0.1***

**julievdh**

**Dec 16, 2018**

# Contents:

CHAPTER 1

# What I learned

## 1.1 For next/new projects

1. Directory structure idea to work towards
2. Do automated testing for small things (units, etc) or big things. Matlab (maybe outdated) or documentation
3. Related, Travis for auto-test and to display test to community
4. snakemake and makefiles in general

## 1.2 For existing projects

- branch from master to test/add new things, then merge
- git graph
- README all the time

## 1.3 When interacting on GitHub

- fork/config - branch - pull request
- “fixes #N” to address issue in pull request

## 1.4 In general

- so much more familiarity with terminal
- used nano for the first time (and got into the swing of things)

- wrote first shell commands
- wrote first python scripts
- realized that jupyter notebooks aren't all I thought they would be [I think this is a good thing – version control > notebook]
- fixed my website!
- **gh-pages or sphinx to generate html, make various pages**
    - pair with readthedocs

## 1.5 Useful commands

- touch

CHAPTER 2

# Feedback

## 2.1 Day 1

Intro to version control, Social Coding, Open Data

- Really useful and important background, foundation of the course
- Good background even to difference between git and github, git vs "open data" as many think these are synonymous
- Time management can be challenging but I appreciate the time given to specific questions, follow-up, etc.
- Tutorials were good in that they reminded us of code/lessons we'd previously learned, re-emphasized tricks, etc.
- Good progression of lessons/code-alongs (time for us to implement, then listen, then implement, then listen)

## 2.2 Day 2

Automated testing

- Something I'd never implemented that I see being extremely useful
- Excellent practice and aspect to introduce my research group to
- Tutorial allowed us to implement things we'd learned the previous day
- Great introduction to Travis, and implementation. Interesting social platform to show that we use testing, etc.

Archaeology with git

- Also extremely useful, as this is something many of us face.
- Tutorial could have used more show, then let us do it, then show more, then let us do it, to keep everyone following along.
- grep-ing grep.c was confusing - consider using a different file to grep as an example.

## 2.3 Day 3

Reproducible research

- Great practices to implement (e.g. file tree example, makefiles/snakemake)
- Principles were relevant and tool to implement them was clear
- Good check with audience for whether or not certain things needed to be discussed (e.g. 'is reproducibility important' or the degree to which we used containers)

Documentation

- I liked the different methods shown to do this, the exposure to the different platforms etc.
- Some of the lessons were too demo and then implement after (more code-along with many short breaks might be good in the future)
- Really important skills to be able to produce documentation on in these different formats, with different levels of distribution, etc.

Jupyter Lab

- Good to get acquainted with it – I'd heard a lot and had tried to implement it, but unsuccessfully.
- If anything, learned that it's not all I expected it to be and that other aspects are more important to me (version control, documentation, distribution, workflows)

Modular coding

- Was shorter than planned. Good discussion and important messages.

Overall - Budget a bit more time in for wrap-up, discussion, questions - Keep the model of short pauses in code-along for attendees to implement - Day on/day off model, or morning on/afternoon off model - I hope this can continue to grow around the Nordics - really valuable, and something I hope others attend in the future. I'll spread the word! - Trying to implement the things I learned right away, and remember which ones are important as I open new projects, etc.

Thanks!!!