
Simulation of Atmosphere, Oceans and Climate

Keeping track of your code:

using **git** and **defensive programming**
for version control and code reproducibility

Dr Erik van Sebille

Institute for Marine and Atmospheric Research, Utrecht University

Why version control?

2

"FINAL".doc



FINAL.doc!



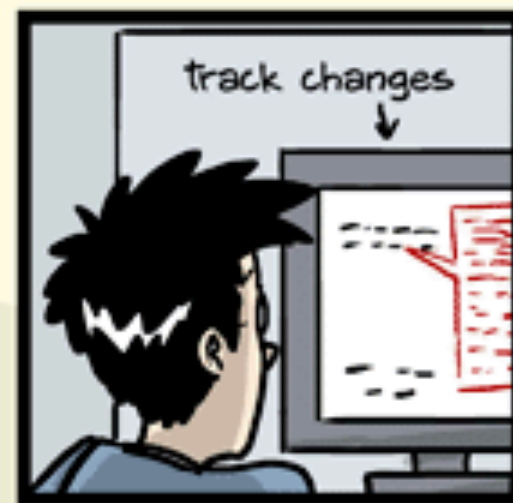
FINAL_rev.2.doc



FINAL_rev.6.COMMENTS.doc



FINAL_rev.8.comments5.
CORRECTIONS.doc



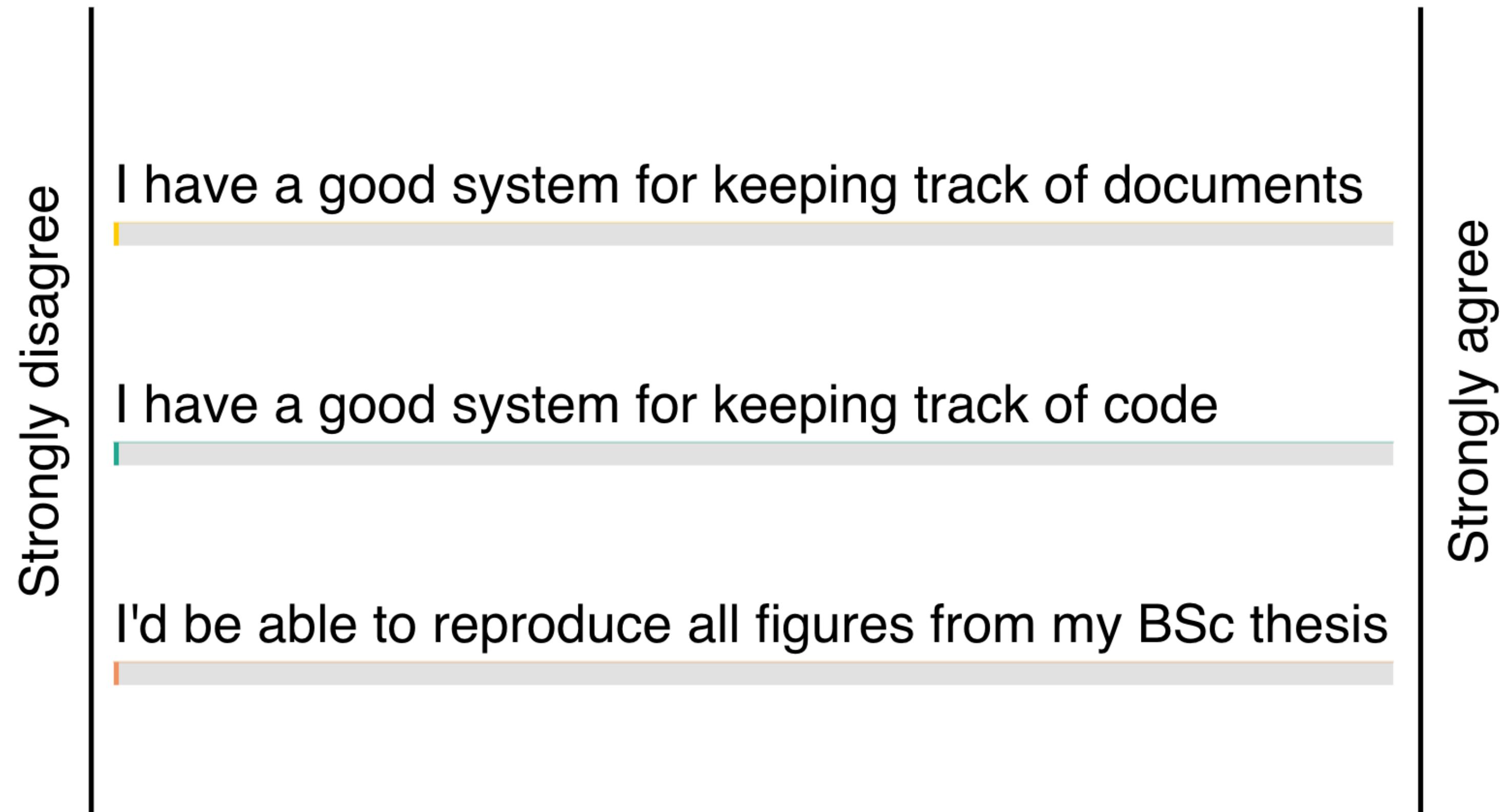
FINAL_rev.18.comments7.
corrections9.MORE.30.doc



FINAL_rev.22.comments49.
corrections.10.#@\$%WHYDID
ICOMETOGRADSCHOOL?????.doc

JORGE CHAM © 2012

Are you organised



- ▶ Why version control?
- ▶ Open science and reproducibility
- ▶ Git and GitHub (and gitlab)
- ▶ Defensive Programming, unit testing, and Continuous Integration
- ▶ Some of the content based on
 - ▶ Software Carpentry (lesson: Version control with Git; <http://swcarpentry.github.io/git-novice/>)
 - ▶ Github guides
- ▶ More info and training at Python 4 Lunch:
 - ▶ Biweekly meetings (in BBG 607) organised by my IMAU PhD students and postdocs
 - ▶ Open to all. See also <https://github.com/UU-IMAU/Python-for-lunch-Notebooks>

- ▶ Keep track of versions/changes of your code
 - ▶ E.g. easily return to the last working version of your code
- ▶ Ensure reproducibility of simulations
- ▶ Facilitate collaboration on code
- ▶ Get acknowledged for your code
- ▶ Contribute to 'open science'

Why 'open science'?

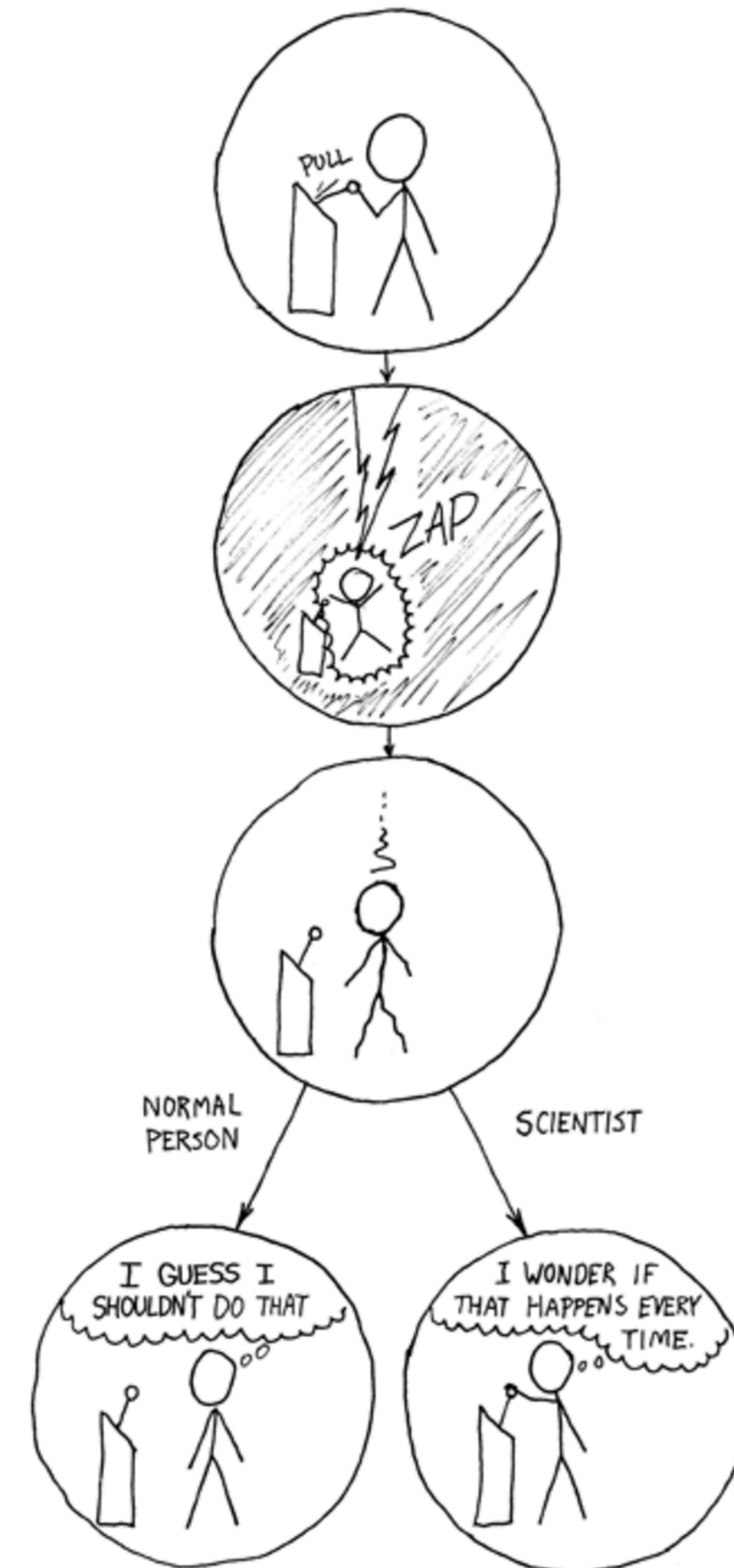
6

- ▶ Open science is about collaborative science

The opposite of "open" isn't "closed". The opposite of "open" is "broken"

- John Wilbanks, Creative Commons

- ▶ Science needs to be reproducible
- ▶ UU and funders require Data Management Plan (DMP) for each project
 - ▶ Storing data and metadata (how the data is generated) for 10 year period
 - ▶ Someone might come back in 7 years and demand figure regeneration
 - ▶ Crucial that the experiment is reproducible
 - ▶ In climate simulations, documenting code is key aspect of DMP



Credit: Randall Munroe/XKCD, CC BY-NC

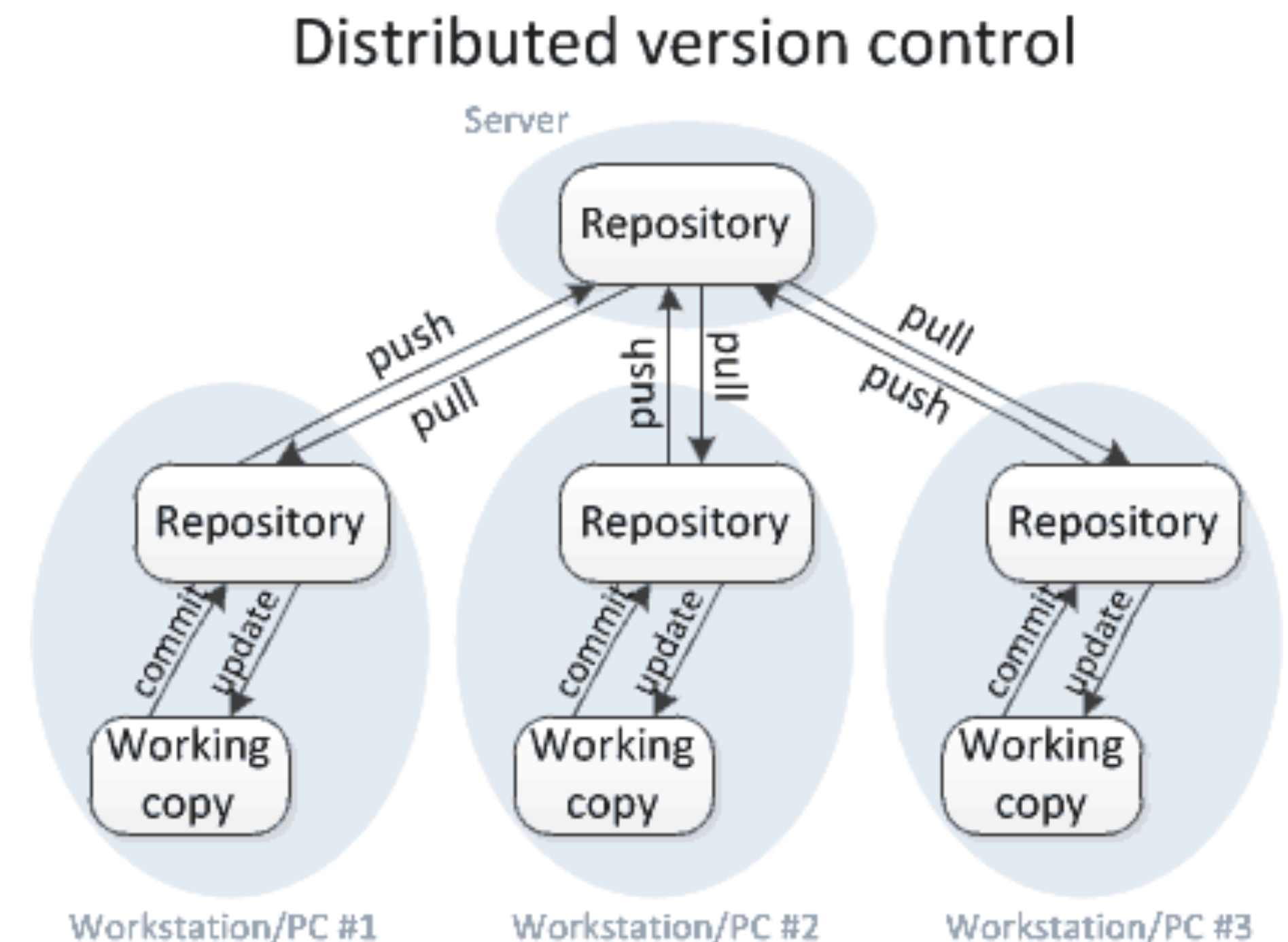


- adding alternative evaluation e.g. with altmetrics
- communicating through social media e.g. Twitter
- sharing posters & presentations e.g. at FigShare
- using open licenses e.g. CC0 or CC-BY
- publishing open access, 'green' or 'gold'
- using open peer review e.g. at Peerage of Science
- sharing preprints e.g. at arXiv, bioRxiv or OSF
- using actionable formats e.g. with Jupyter
- open XML-drafting e.g. at Overleaf or Authorea
- sharing protocols & workflow e.g. at MyExperiment
- sharing notebooks e.g. at OpenNotebookScience
- sharing code e.g. at GitHub with GNU license
- sharing data e.g. at Zenodo, Dryad, Dataverse
- pre-registering e.g. at OSF or AsPredicted
- commenting openly e.g. with Hypothes.is
- using shared reference libraries e.g. with Zotero
- sharing (grant) proposals e.g. at RIO



- ▶ Changes to code can be tracked in a **Version Control system**
- ▶ All Version Control systems work with tracking changes
 - ▶ Apple TimeMachine, Google Drive history, Dropbox history etc are all Version Control systems
 - ▶ However, these don't leave the user much control of which changes to track, or to add comments
- ▶ In code development, **git** is the most widely used system

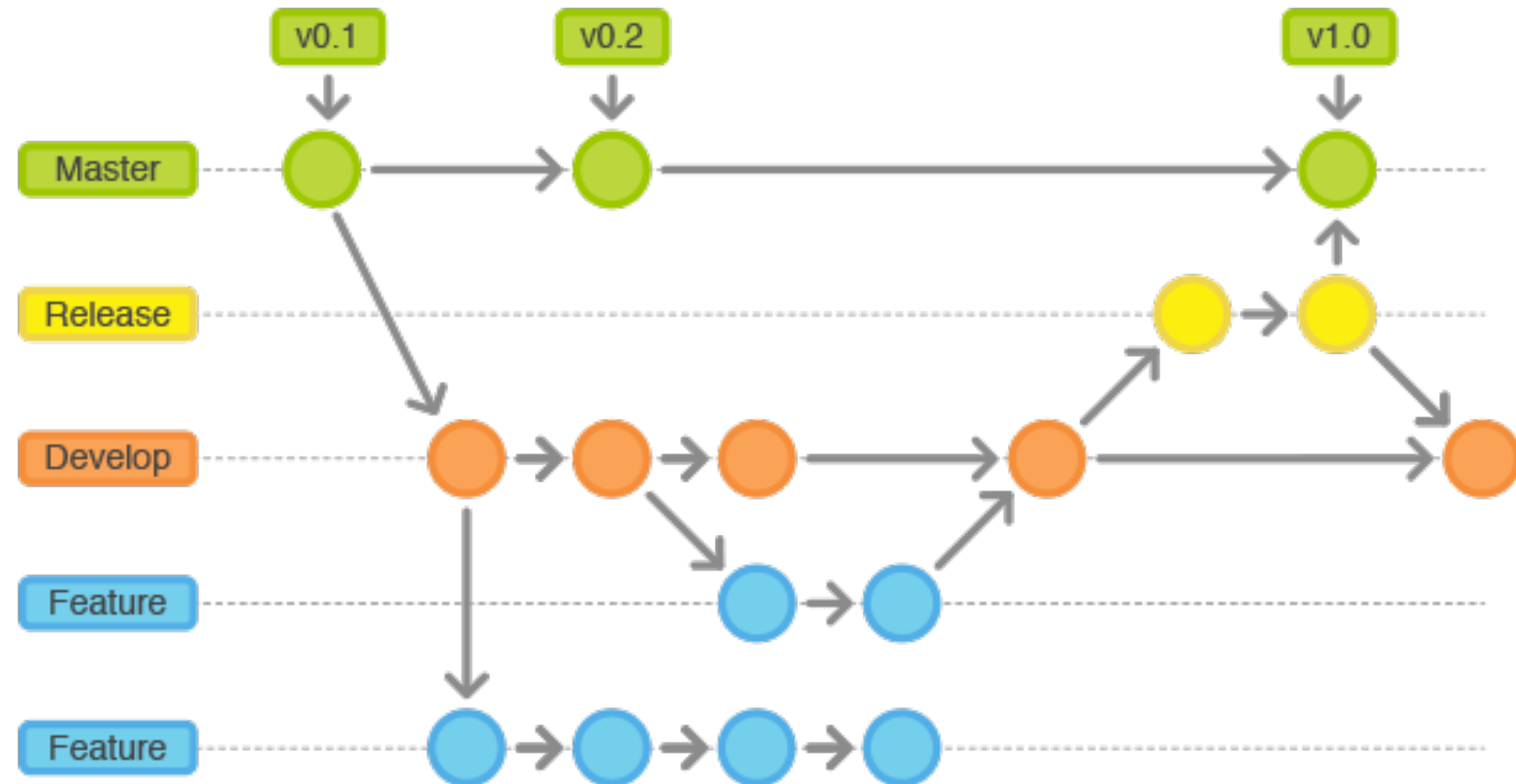
- ▶ git works with **repositories** and **local working copies**
- ▶ Repository holds the 'clean' version of the code, while you make changes in your working copy
- ▶ Once you're happy with changes, you commit from working copy to *local* repository
- ▶ Then, you push from local repository to *central* repository
- ▶ Other users can then pull from the central repository



- ▶ Git by itself is open-source software run from the command line
- ▶ Many websites provide server capacity, and also easy-to-use tools for managing git
 - ▶ [GitHub.com](https://github.com), bitbucket.com, UU gitlab at git.science.uu.nl
- ▶ [GitHub.com](https://github.com) has the biggest user base of these, and is a fantastic platform.
 - ▶ But was recently purchased by Microsoft...

1. Set up a [GitHub.com](https://github.com) account (or login to git.science.uu.nl)
2. Create a new repository on the site (e.g. 'SOACproject'?)
3. Download the GitHub Desktop tool
4. Clone your repository within the GitHub Desktop tool (also works for git.science.uu.nl)
5. Create a text file with "Hello world" and add it to the folder of your local repository
6. In the GitHub Desktop tool, commit the text file to the local repository
7. Push the changes to the repository on the GitHub server.
8. Go to the GitHub website to check if file is there
9. Make a change to the text file on your local computer and repeat steps 6-8

- ▶ One great feature of git is the support for branches
- ▶ Allows developers to code away at features independently
- ▶ After completion of the feature, it can be **merged** back into the **master branch**
- ▶ At GitHub, this is called a **Pull Request**



- ▶ How to make sure that a merge fixing one part of your code doesn't break another part?
- ▶ Use **unit tests**: small snippets of code that test individual features
 - ▶ In Python you can use *assert* statements
 - ▶ More advanced is the `py.test` module and environment
 - ▶ This is also called Defensive Programming
- ▶ Then hook a Continuous Integration service (e.g. [Travis-CI.org](https://travis-ci.org/) or Github Actions) to your repository
 - ▶ Every push to the server repository will trigger a check of all unit tests
 - ▶ Commit is flagged if some unit tests fail
 - ▶ Never need to worry anymore on breaking your code!



- See interactive tutorial on Defensive Programming at <https://tinyurl.com/y3llfmxi>

- ▶ Depending on the needs of your project, GitHub has many more features
- ▶ Issue tracker to keep track of bugs, requests of new features etc
- ▶ README.md to provide introduction and background for interested visitors
- ▶ LICENSE.md to provide information on which license the repository is released under
 - ▶ Examples are Creative Commons, MITlicense, GPL
- ▶ Releases to tag certain versions as official release
 - ▶ Web-hook to zenodo.org can automatically create Digital Object Identifier (doi) for release

- ▶ There are many open-source projects on GitHub, where you can contribute to if you want
- ▶ See <https://pangeo-data.github.io/> for a great community whose aim is to:
- ▶ “cultivate an ecosystem in which the next generation of open-source analysis tools for ocean, atmosphere and climate science can be developed, distributed, and sustained.”



A community platform for Big Data geoscience

- ▶ Overleaf is a collaborative writing environment for LaTeX
 - ▶ ‘Like Google Docs for LaTeX’
- ▶ Cool feature is that it runs git ‘under the hood’
- ▶ So you can work on your local machine and use GitHub Desktop to sync with overleaf.com server
 - ▶ Take advantage of commit comments, branches etc, while also provide interactive writing environment
- ▶ If you use your UU email address to log in: ‘professional’ features like track-change and comments

- ▶ Open science and reproducibility of results are very important in academia these days
- ▶ For modelling and simulations, version control is an important aspect of reproducibility
 - ▶ Also facilitates collaboration and reuse of code
- ▶ Git is the most widely used Version Control system, and [GitHub.com](https://github.com) the dominant server solution
 - ▶ overleaf.com is a nice Git solution specific for LaTeX
- ▶ Defensive programming is one way to reduce the number of bugs in your code
- ▶ Check out github.com/UU-IMAU/Python-for-lunch-Notebooks for tips and tricks on python!
- ▶ I hope you take this opportunity to try out a git workflow for SOAC project (and thesis?)
- ▶ Note, we haven't answered the question of what to do with the *data* from simulations
 - ▶ Difficult to give generic answer: consult with your supervisor. YODA is an option

THE #1 PROGRAMMER EXCUSE
FOR LEGITIMATELY SLACKING OFF:

"GITHUB IS DOWN."

HEY! GET BACK
TO WORK!

GITHUB IS DOWN!

OH. CARRY ON.

