Reproducible science through versioning of data and analysis

Tomas Knapen
VU Cognitive Psychology & Spinoza Centre, KNAW

VU VRIJE UNIVERSITEIT AMSTERDAM

SPINOZA
Spinoza Centre for Neuroimaging

Reproducible Science

- How to make science reproducible?
  - Work on practices:
    - no HARKing, don't keep testing until significant, preregister(?), etc.
    - Teach people Statistical/Reasoning literacy
  - Make everything check-able, meaning: ***Open***:
    - Don't keep data/methods/papers to yourselves
    - Teach people ***Open Science*** skills and ways of thinking

Trust me, it checks out!

The end result: Open publishing

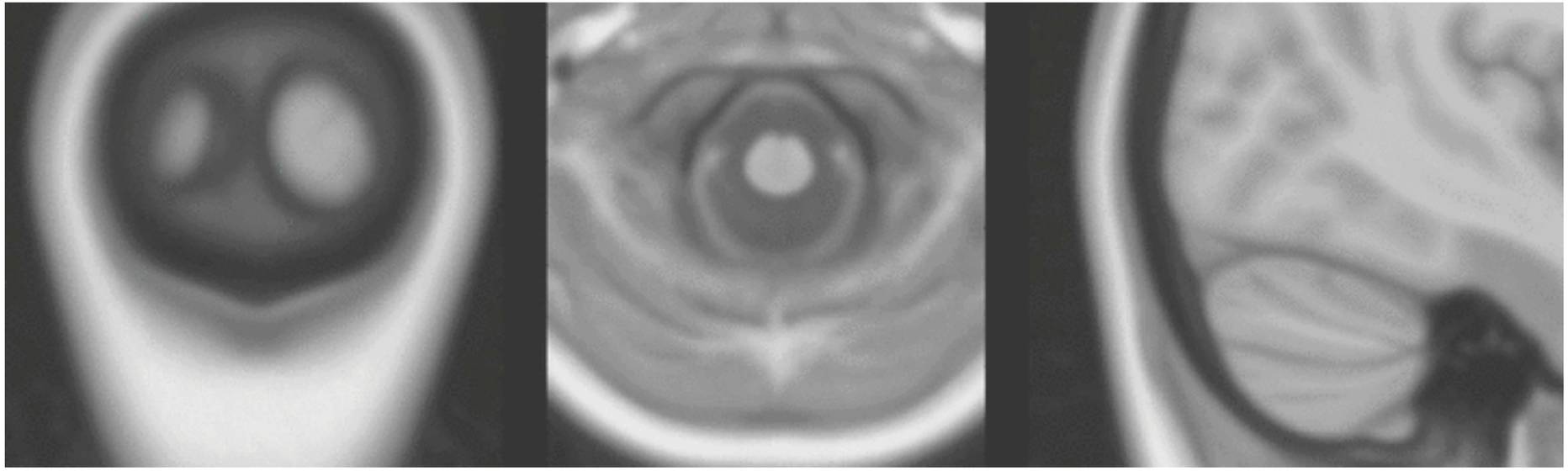
- Publish “Open”:
  - ***Open access:*** you pay so that anyone can access
- Publish 2.0:
  - ***Creative Commons license:*** You maintain copyright
  - Wrest control over publication from the publishing corporations
  - ***Preprint*** servers: speed up dissemination, and make publication more fluid
  - Perhaps start experimenting with ***new ways of doing peer review***, etc: eLife, PLoS, Frontiers, *Twitter* (?), etc.

The start: Open data

- Data are the bedrock of scientific results, of course.
- Save your data in a common format (not your own!), extra work!
  Many people see it as a chore
- Many people feel ownership of their data:
  I'm not giving that away!
- But it's not all bad:
  - It's becoming more common to share data.
  - With soo much data at your fingertips, you become science Superman/Woman!

An example use of open data

- Human Connectome Project:
  - 1200 participants scanned multiple times:
    - Tasks, Resting State, Movie Watching, Retinotopy, Anatomy, DTI
  - All data freely available

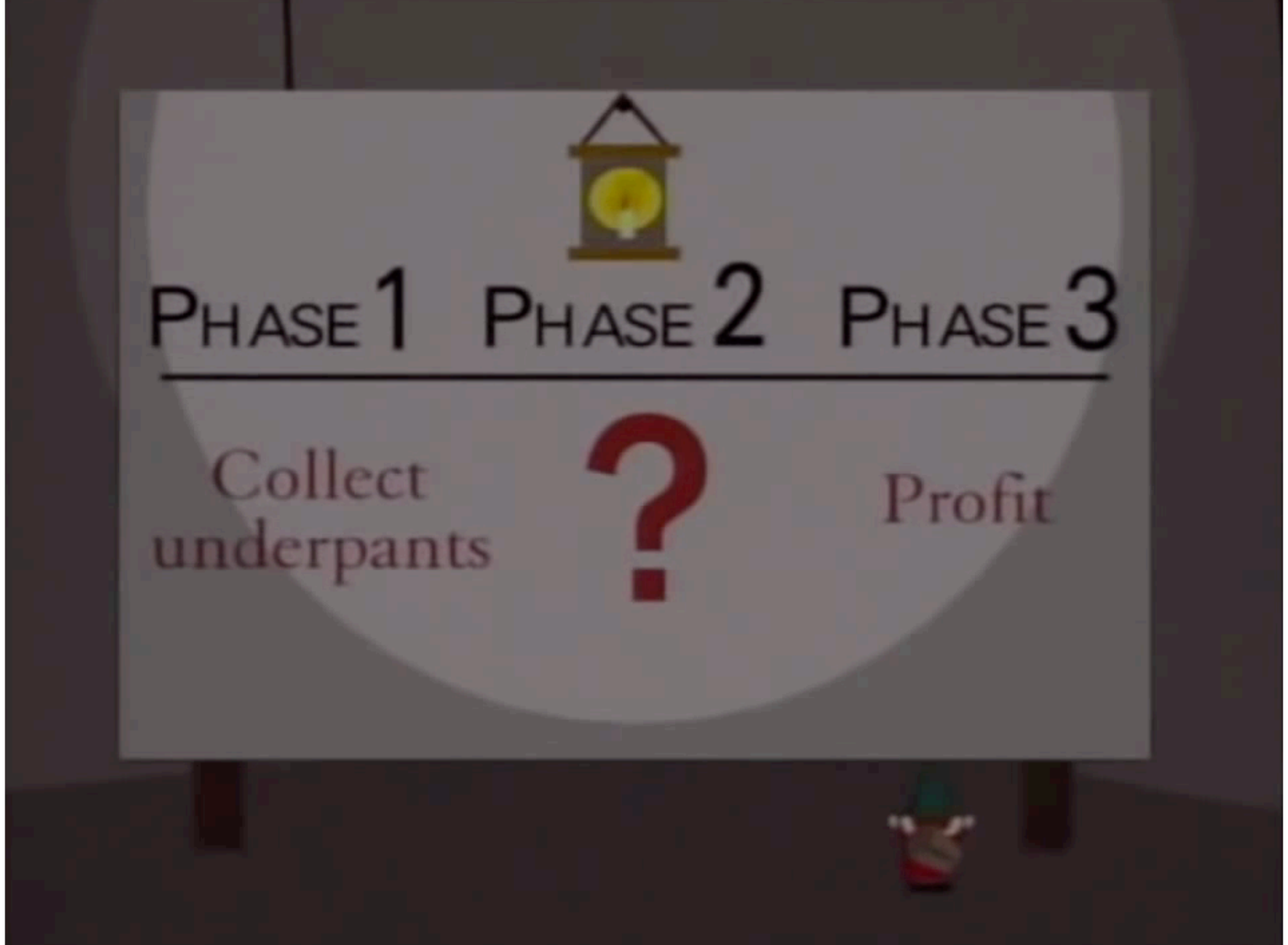


- ***We discovered 5 retinotopic maps of visual space in the cerebellum.***
- Only possible by averaging 181 subjects, otherwise too hard to find: need multiple hours of data for single subject results…

Ho ...

en

ishing



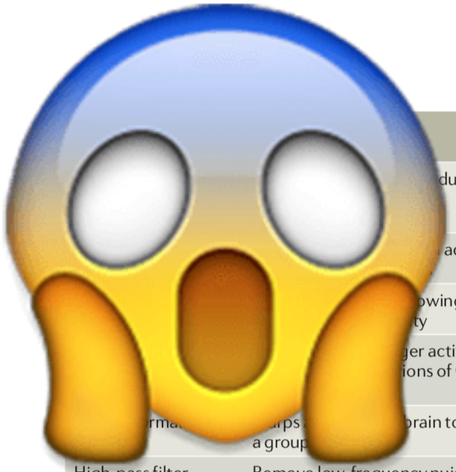
How to get from open data to open papers…

All steps between data and publication present opportunities for non-reproducible outcomes

Open Data

Open Publishing

Computer Hardware



Operating System

		Options [suboptions]	Number of plausible options
		during	• 'Interpolation' [linear or sinc] • 'Reference volume' [single or mean]	4
		acquisition	'No', 'before motion correction' or 'after motion correction'	3
		owing to	'Yes' or 'no'	2
		ger activations ions of GRF	'FV	
		rain to match	'M	
High-pass filter	Remove low-frequency nuisance signals from data		'Fr 12	
Head motion regressors	Remove remaining signals owing to head motion via statistical model		'Ye pa 'sc	
Haemodynamic response	Account for delayed nature of haemodynamic response to neuronal activity		• 'E o • 'Derivatives' ['none', 'shift' or 'dispersion']	
Temporal autocorrelation model	Model for the temporal autocorrelation inherent in fMRI signals	'Yes' or 'no'		2
Multiple-comparison correction	Correct for large number of comparisons across the brain	'Voxel-based GRF', 'cluster-based GRF', 'FDR' or 'non-parametric'		4
Total possible workflows				69,120

Preprocessing

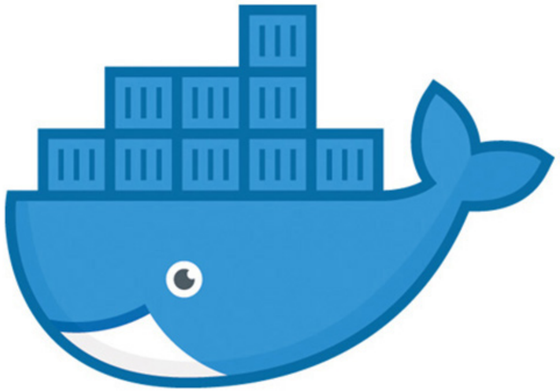
Your own manual data analysis

```
#!/usr/bin/env python
import sys
import os
import simpleknn
from bigfile import BigFile

if __name__ == "__main__":
    trainCollection = 'toydata'
    nimages = 2
    feature = 'f1'
    dim = 3

    testCollection = trainCollection
    testset = testCollection

    featureDir = os.path.join(rootpath, trainCollect
```

Your own automated data analysis

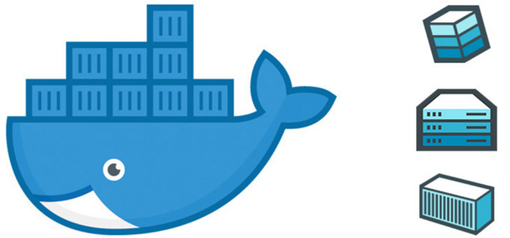
Tools for open science

- Make computing environments reproducible:
Docker



- Make our own workflow reproducible:
Version Control, Git and derivatives

Docker in science

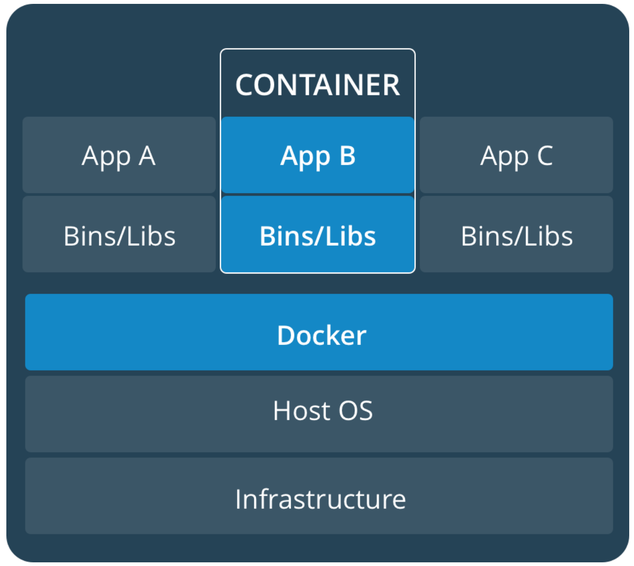


For reproducibility:

It matters which operating system you use

It matters which versions of installed software you use (Matlab2015A or 2018B, FSL5.09 or 6.0, etc, etc.)

Docker uses a “container” architecture, where a container ***encapsulates an entire operating system and its installed software.***



Docker in science

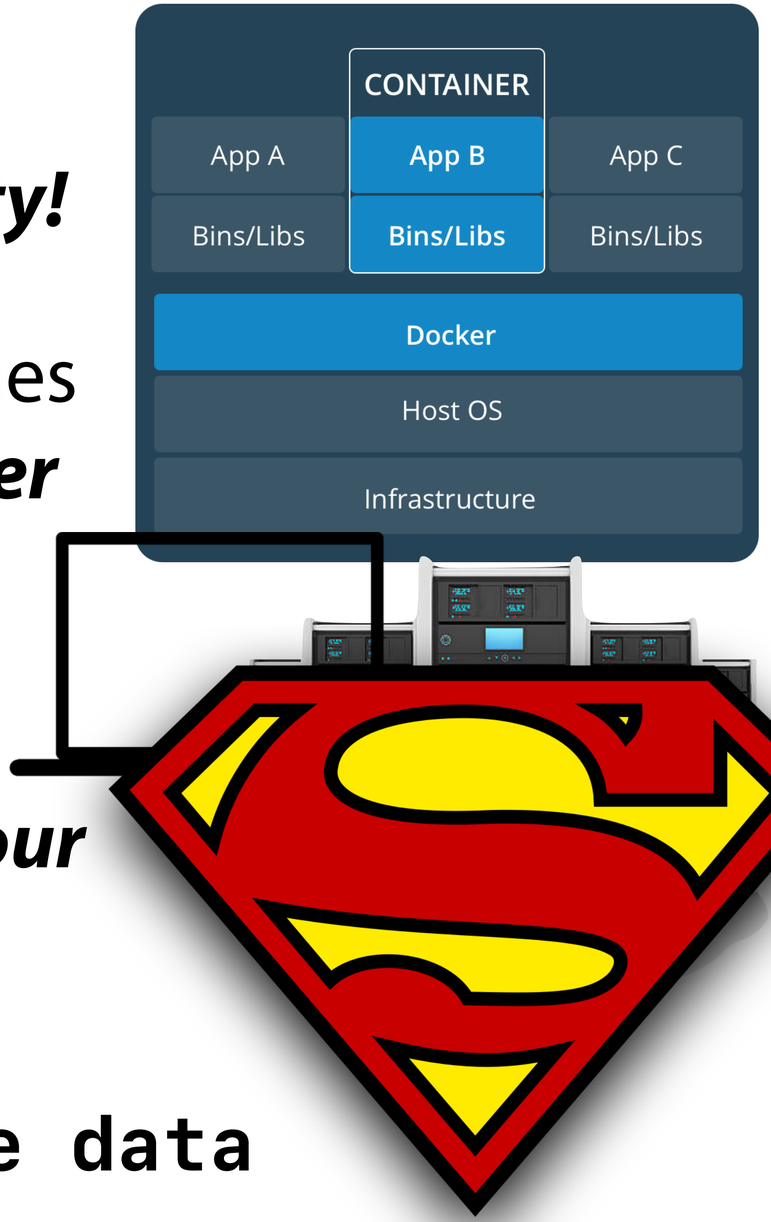
This makes it easy to ***share the entire software environment*** that ran your analysis: ***Reproducibility!***

You don't have to install all of your different packages on different computers: just ***run the same container everywhere***: on your laptop, desktop, or cluster!

Different docker images for different analyses:
Run standardised (pre-)processing pipelines on your data

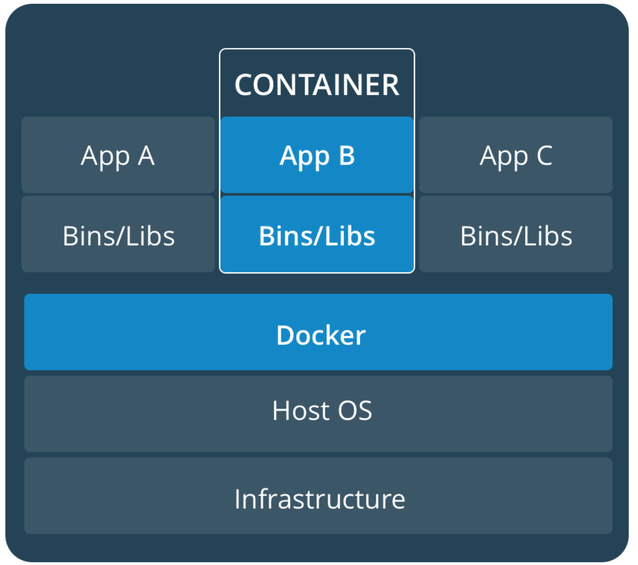
As easy as:

```
$ docker run container-image data
```



Docker in science

Docker containers are based on an image, which you can create using a recipe, called a *Dockerfile*



It prescribes exactly which operating system and software should be installed in the image:

```
FROM ubuntu:18.04
RUN apt-get install –y R
CMD [“echo”,”Image created, R installed”]
```

This creates a new ubuntu image, and installs R in there, ready to use.

Much more to docker, but this is just to make you aware of the possibilities……

How to get from open data to open papers…

Open Data

Docker solves these:

Computer Hardware

Operating System

Preprocessing

Processing step	Reason	Option	Number of plausible options
Motion correction	Correct for head motion during scanning		4
Slice timing correction	Correct for differences in acqui timing of different slices	correction' rrection'	3
Field map correction	Correct for distortion ov magnetic susceptibili		2
Spatial smoothing	Increase SNR for l and ensure ass theory	mm, 6 mm or 8 mm]	3
Spatial normalization	Warps an a group	ethod' [linear or nonlinear]	2
High-pass filter	Rem	'Frequency cut-off' [100 s or 120 s]	2
ead motion ssors	ng to odel	'Yes' or 'no' [if yes: 6/12/24 parameters or single time point 'scrubbing' regressors]	5
nam	ture of onse to	• 'Basis function' ['single-gamma' or 'double-gamma'] • 'Derivatives' ['none', 'shift' or 'dispersion']	6
	temporal tion inherent in fMRI	'Yes' or 'no'	2
	t for large number of parisons across the brain	'Voxel-based GRF', 'cluster-based GRF', 'FDR' or 'non-parametric'	4
			69,120

How about these?

Open Publishing

Your ow ual data

ary

```
#!/usr/bin/env python
import sys
import os
import simpleknn
from bigfile import BigFile

if __name__ == "__main__":
    trainCollection = 'toydata'
    nimages = 2
    feature = 'f1'
    dim = 3

    testCollection = trainCollection
    testset = testCollection

    featureDir = os.path.join(rootpath, trainCollect
```

Your own automated data analysis

How to get from open data to open papers…

```python
#!/usr/bin/env python
import sys
import os
import simpleknn
from bigfile import BigFile

if __name__ == "__main__":
    trainCollection = 'toydata'
    nimages = 2
    feature = 'f1'
    dim = 3

    testCollection = trainCollection
    testset = testCollection

    featureDir = os.path.join(rootpath, trainCollect
```

Your own automated data analysis

How do we keep track of these different steps?

Open Publishing

Final Analysis

Final Data

Analysis Step 3

Intermediate Data 3

Analysis Step 2

Intermediate Data 2

Analysis Step 1

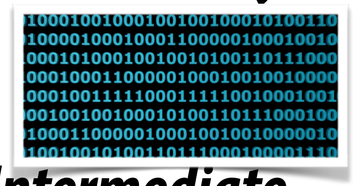
Intermediate Data 1

Keeping track of your work

Analysis Step 1

Intermediate Data 1

Analysis Step 2



Intermediate Data 2

Analysis Step 3

Intermediate Data 3

Final Analysis

Final Data

Your analysis work is a living, breathing work of art!

It changes all the time…..

Name	Date Modified
analysis_version__backup_9.py	Today at 17:26
analysis_version__backup_8.py	Today at 17:26
analysis_version__backup_7.py	Today at 17:26
analysis_version__backup_6.py	Today at 17:26
analysis_version__backup_5.py	Today at 17:26
analysis_version__backup_4.py	Today at 17:26
analysis_version__backup_3.py	Today at 17:26
analysis_version_11.py	Today at 17:26
analysis_version_10.py	Today at 17:26
analysis_version_9.py	Today at 17:26
analysis_version_8.py	Today at 17:26
analysis_version_7.py	Today at 17:26

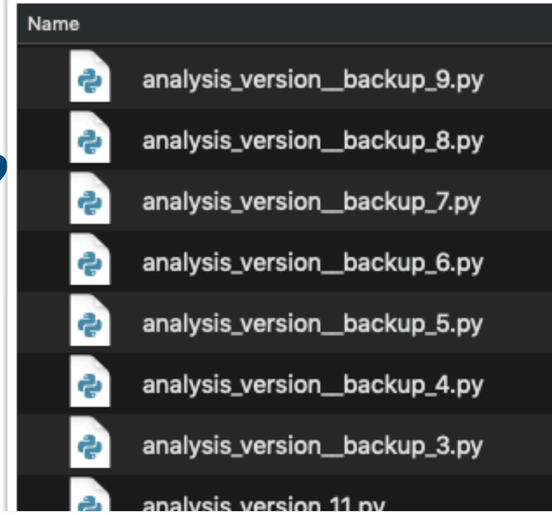
Keeping track of your work

But: ***which version does what?***

And, if I need to go back to an older version, which one?

And, ***what else will have changed*** *when I go back?*

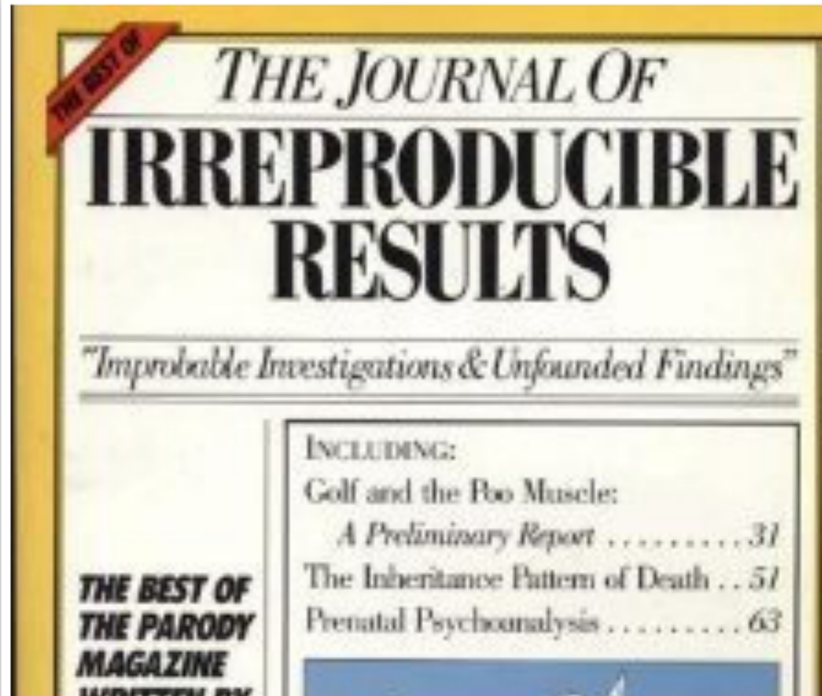
And, which version do I send to my colleagues?



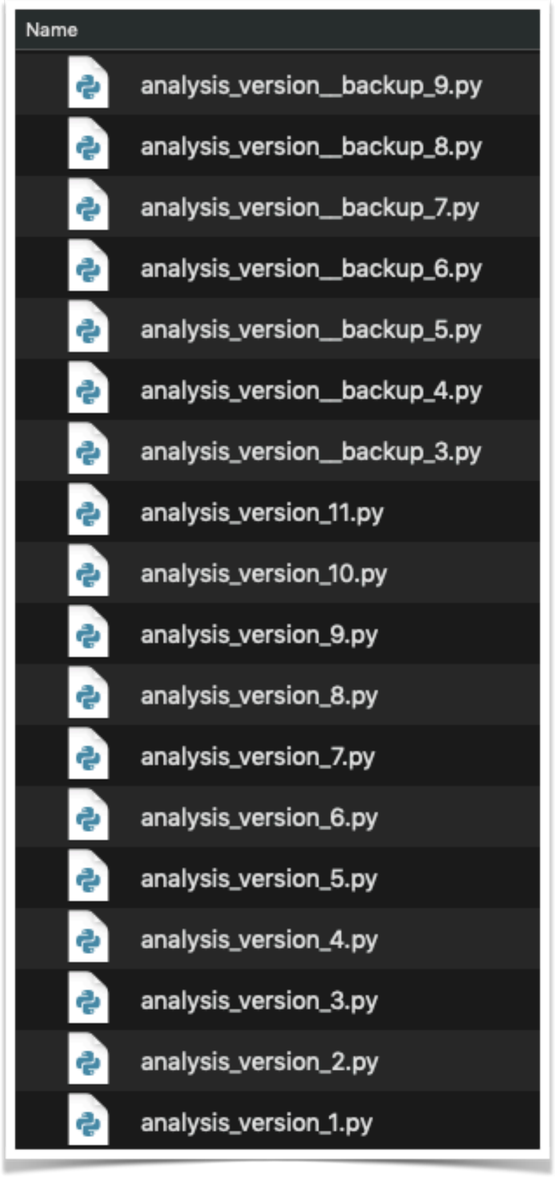
This is NOT the way to go, quick and dirty is DIRTY

You will always have to go back

Have mercy on your future self!



Getting out of the mess

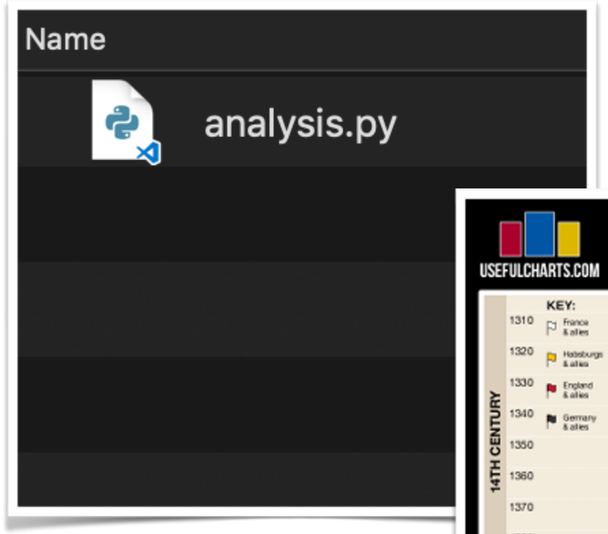


If only we could:

- *Keep our working directory clean*
- *While having a clear history at the same time*
- *Make our analysis easily share-able*

We can!

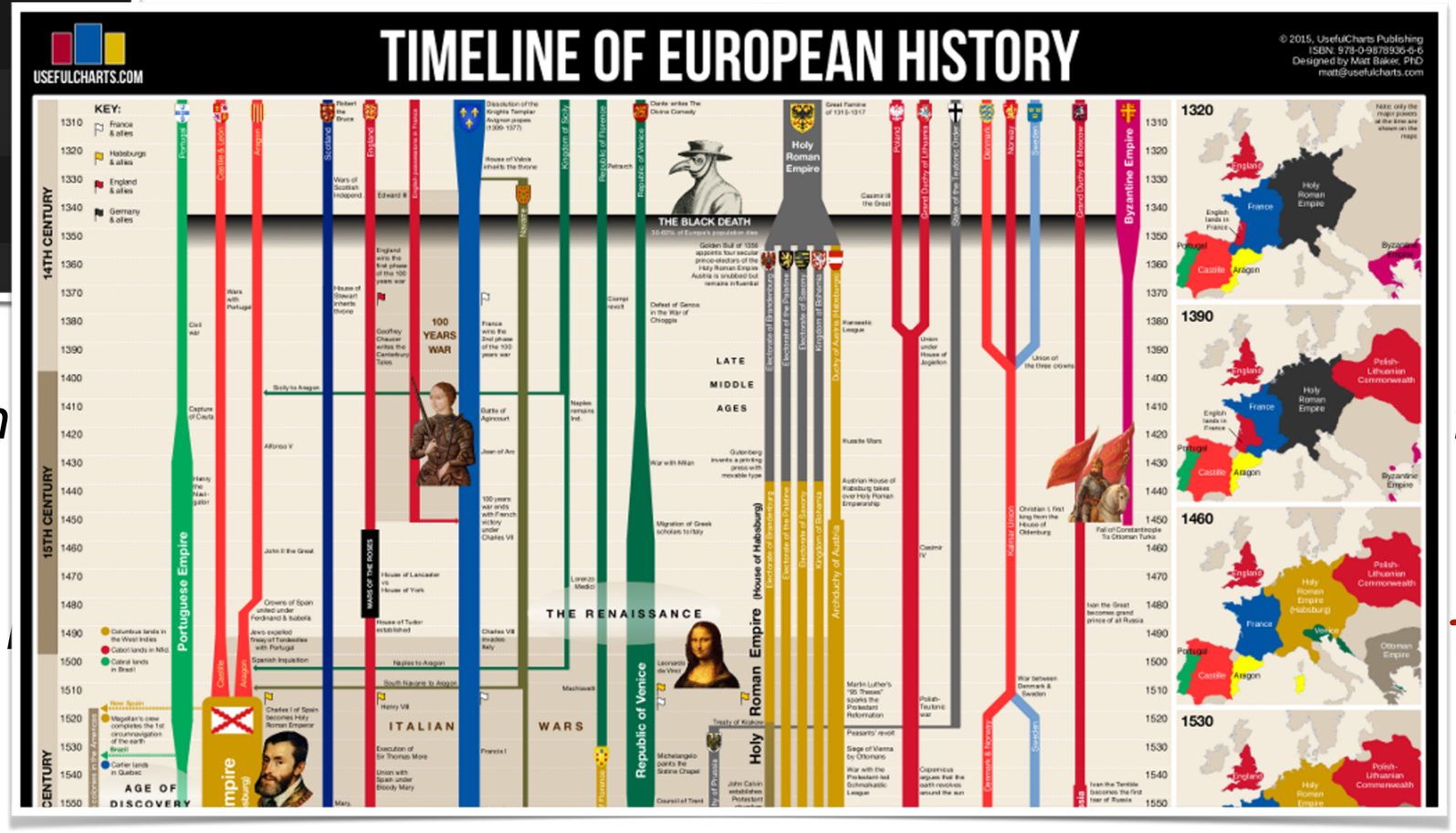
Version control



Use a version control system!

In *for*

f

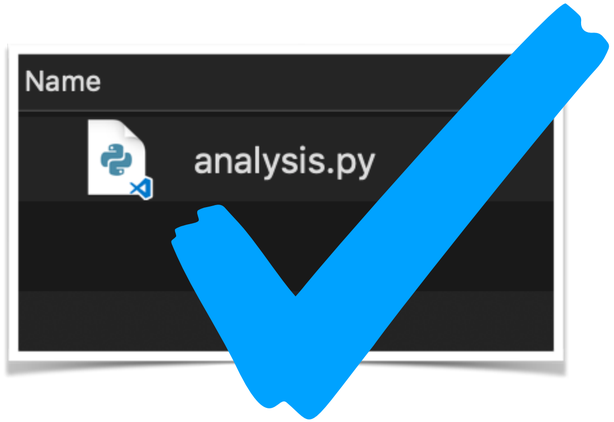


Versions

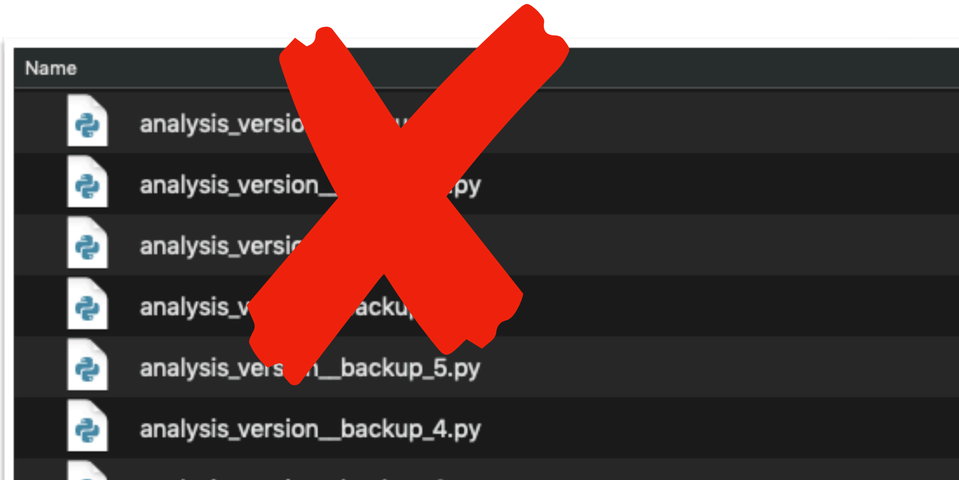
Version control works in the background, making it easy for you to keep track of changes in your work.

You only see one version of a file/folder, which can make you nervous, but:

Git keeps track of the changes



Implicit timeline



Basic Git

Start a repository in a folder:

`git init`

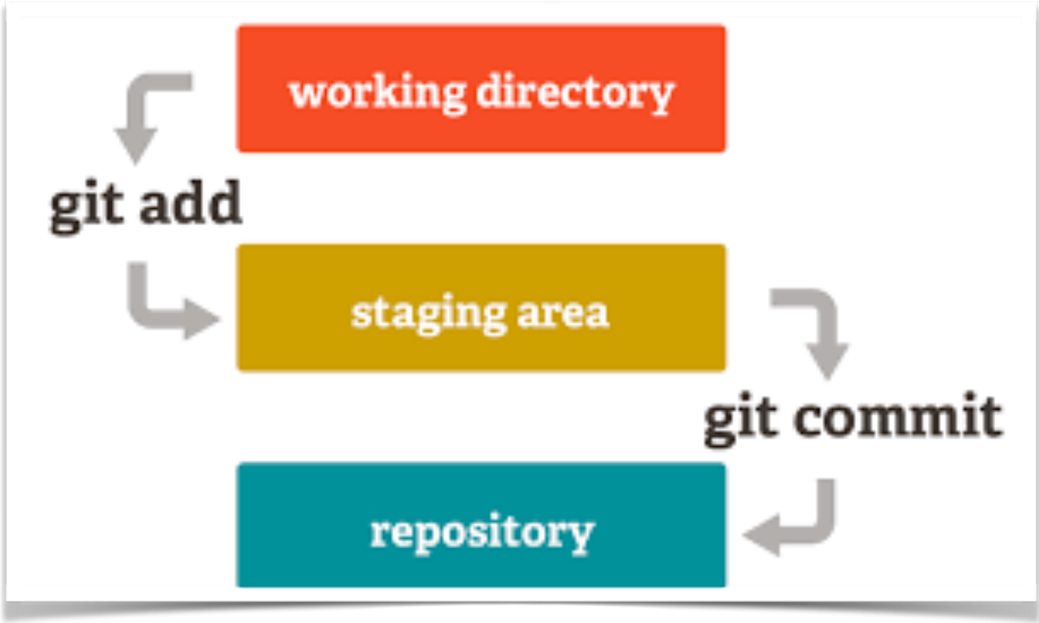
When you add a file to your work, you have to add it to the repository

`git add myfile`

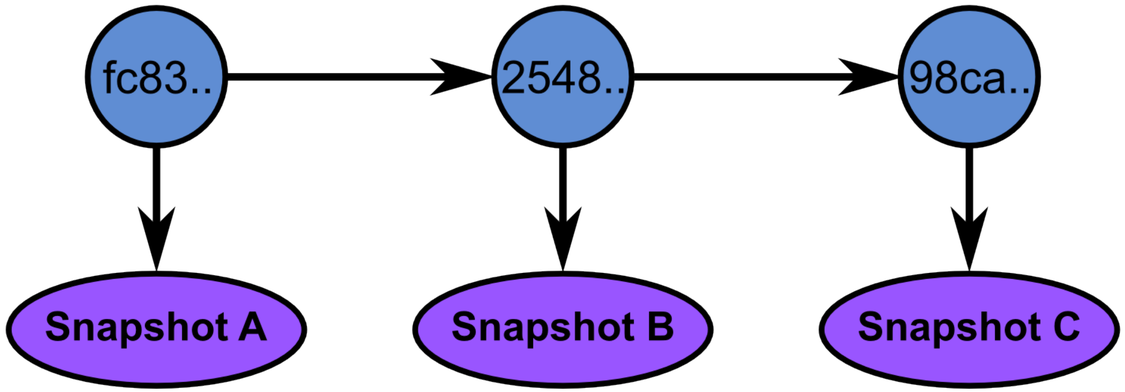
When you've changed files, and want to be able to go back to this point, you commit your changes, to create a *snapshot*

`git commit`

The part you see



The whole history



Basic Git

History for **hcp_movie** / CF_hcp_movie / **utils.py**

Commits on Jun 3, 2019

updated imports 2

tknapen committed 3 days ago

7e347e9 <>

for upload to cartesius

tknapen committed 3 days ago

925eacf <>

synced functions and notebook. Added run.py for cluster running.

tknapen committed 3 days ago

5c3d6b3 <>

Commits on May 31, 2019

added polar angle volume plotting

tknapen committed 6 days ago

fd1f6d0 <>

first pca glm with history components

tknapen committed 6 days ago

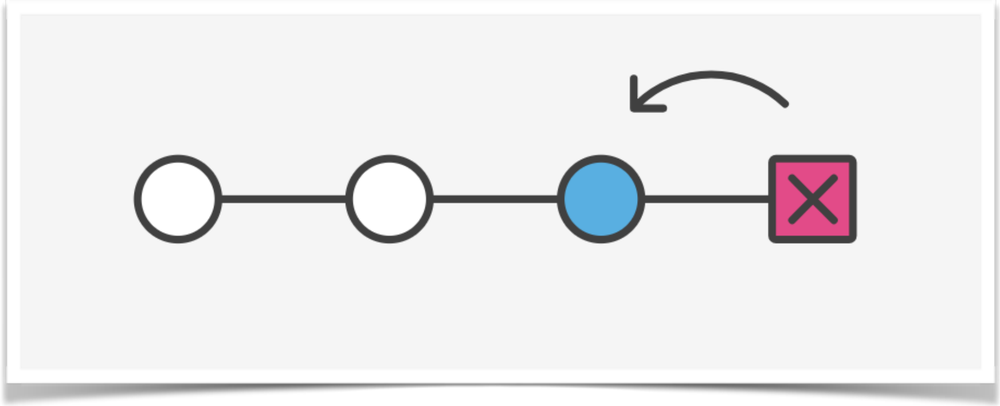
d860889 <>

Commits on May 29, 2019

Implicit, automatic backup

Why do this extra work?

You can always go back to a previous version of your code!



But, this is all still local, on your own computer

GitHub

Github is a huge online database of repositories. It contains 57 million repositories of code, in all manner of programming languages.

You can make an account, and start uploading your repositories to GitHub, either open to the world - or private.

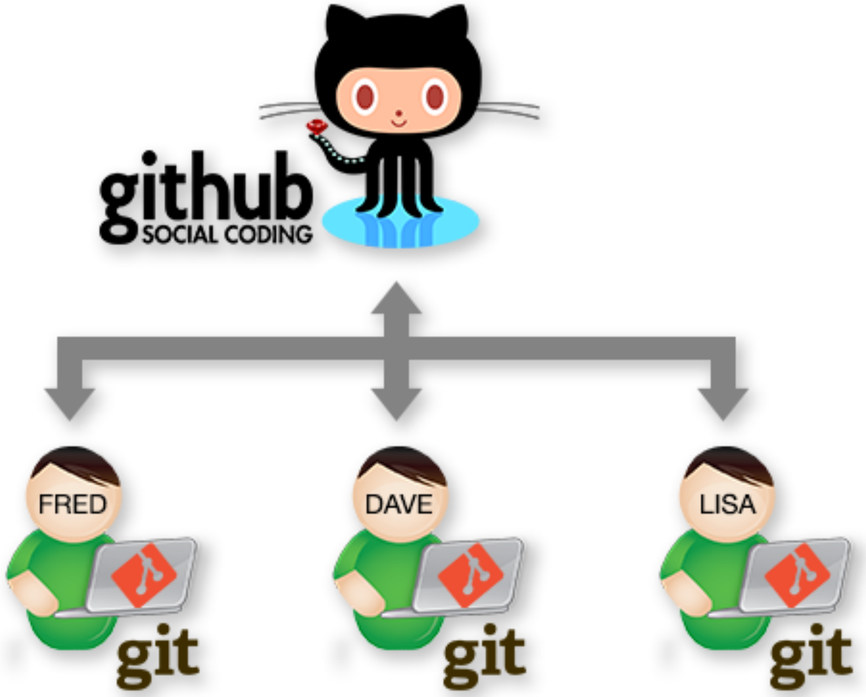
Easy way for visualising your repository's history in the browser or a GUI program - no need for command line terror!

Git / GitHub Demo

Collaboration & GitHub

```
git push origin master
```

Code is backed up online



Awesome way of sharing code!

You become a better coder by reading other people's code

Access to other people's code makes you

GitHub & Further

With a bit of extra work you get a lot for free!

Getting the hang of this will make it easier for you to make your methods ***open***

Similar paradigms are available for data, too

Datalad is built on top of git to track versions of data.

Analysis Step 2

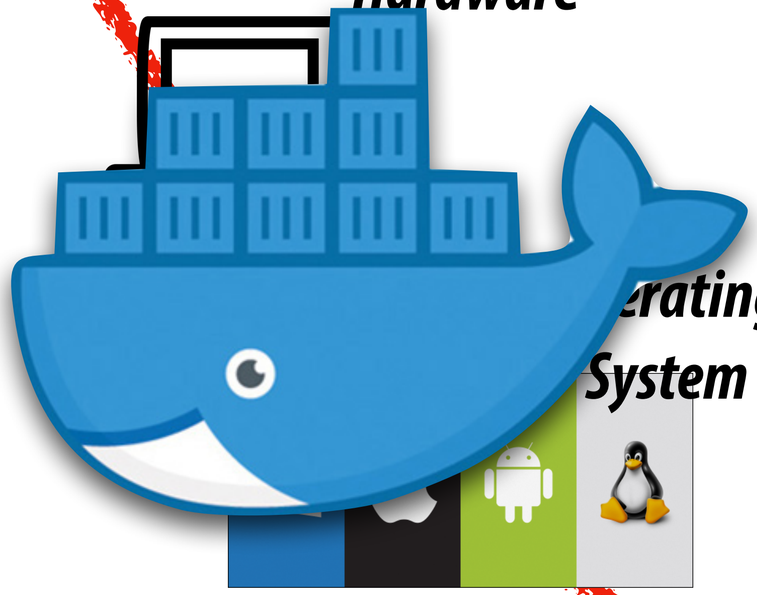
Intermediate Data 2

Take away

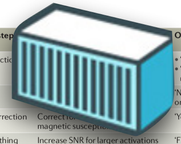
Open Data

Open Publishing

Computer Hardware



erating System

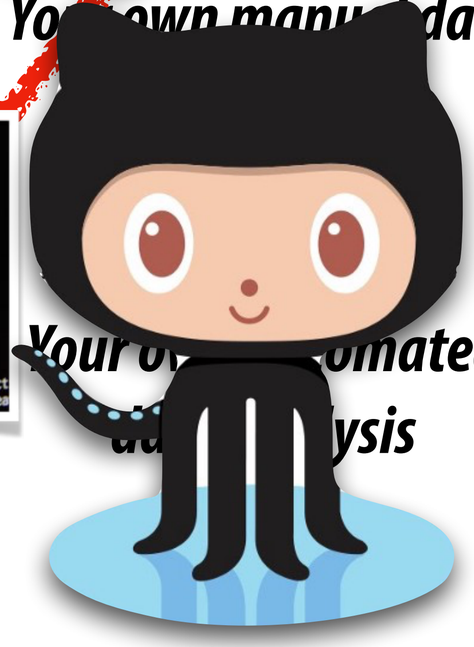


Processing step		Options [suboptions]	Number of plausible options
Motion correctio		• 'Interpolation' [linear or sinc] • 'Reference volume' [single or mean]	4
Slice timing correction		'No', 'before motion correction' or 'after motion correction'	3
Field map correction	Correct for magnetic susceptibi	'Yes' or 'no'	2
Spatial smoothing	Increase SNR for larger activations and ensure assumptions of GRF theory	'FWHM' [4 mm, 6 mm or 8 mm]	3
Spatial normalization	Warps an individual brain to match a group template	'Method' [linear or nonlinear]	2
High-pass filter	Remove low-frequency nuisance signals from data	'Frequency cut-off' [100 s or 120 s]	2
Head motion regressors	Remove remaining signals owing to head motion via statistical model	'Yes' or 'no' [if yes; 6/12/24 parameters or single time point 'scrubbing' regressors]	5
Haemodynamic response	Account for delayed nature of haemodynamic response to neuronal activity	• 'Basis function' ['single-gamma' or 'double-gamma'] • 'Derivatives' ['none', 'shift' or 'dispersion']	6
Temporal autocorrelation model	Model for the temporal autocorrelation inherent in fMRI signals	'Yes' or 'no'	2
Multiple-comparison correction	Correct for large number of comparisons across the brain	'Voxel-based GRF', 'cluster-based GRF', 'FDR' or 'non-parametric'	4
Total possible workflows			69,120

Preprocessing

```
1  #
2  impo
3  impor
4  import s
5  from bigfi    port BigFile
6
7  if __name__ == "__main__":
8      trainCollection = 'toydata'
9      nimages = 2
10     feature = 'f1'
11     dim = 3
12
13     testCollection = trainCollection
14     testset = testCollection
15
16     featureDir = os.path.join(rootpath, trainCollect
```

Yo own manu data



Your o omated
da ysis

No description, website, or topics provided.

7 commits | 1 branch | 0 releases | 1 contributor

Branch: master | Create new file | Find File | Clone or download

daanvanes refined repo after analysis — Latest commit 86a3426 on Feb 20

ipynbs	refined repo after analysis	4 months ago
resources	refined repo after analysis	4 months ago
scripts	refined repo after analysis	4 months ago
submissions	refined repo after analysis	4 months ago
.gitignore	removed .out files	4 months ago
README.md	refined repo after analysis	4 months ago

README.md

cerebellum_prf repo

This repo includes all code for preprocessing and fitting the pRFs to individual subject data.

Files necessary for pRF fitting (such as design matrix) can be found in the resources folder.

The data have been preprocessed through fmriprep. This includes motion correction and spatial alignment to MNI space.