Communication

Presenting Results and Conclusions

Data 102 - Fernando Pérez
Slides credit: Lindsey Heagy

Communication

Presenting Results and Conclusions

Data 102 - Fernando Pérez
Slides credit: Lindsey Heagy

*presentation matters

Outline

- Course so far:
  - Techniques for decision making
  - Understanding of their foundations and assumptions
  - (little) impact of those
- You have RESULTS! CONCLUSIONS!!
- Now What?
- What are results, and conclusions in this context?
- Models? Data?
  - Hydro example: a geophysicist draws a model of the ground and draws a line, and hands this model to a hydrologist.
  - The hydrologist then makes decisions, runs simulations, MCMC, generates confidence intervals.
  - Whose ‘model’, whose ‘data’, whose ‘truth’?
  - Moritz’s point: model -> data -> truth -> action. The model enforces reality
- How do we open this?

Course themes

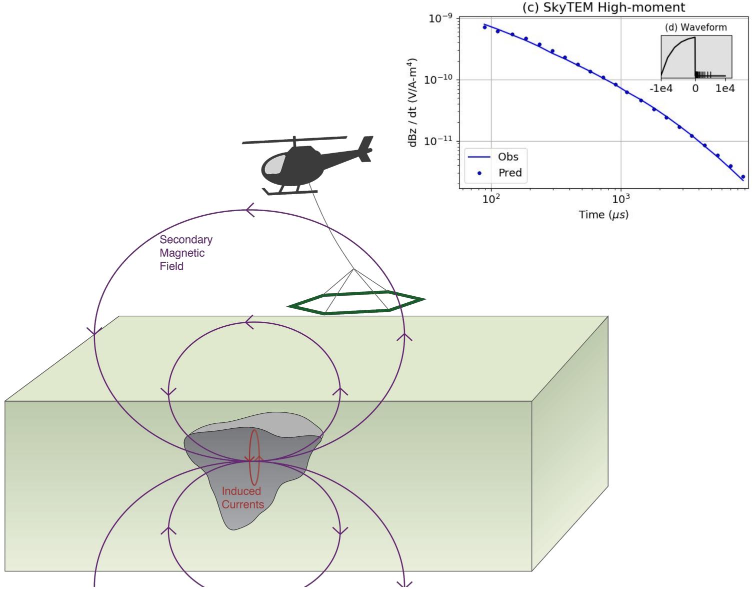
- Techniques for decision making
- Understanding of their foundations and assumptions
- Impact of these

You have RESULTS! CONCLUSIONS!!

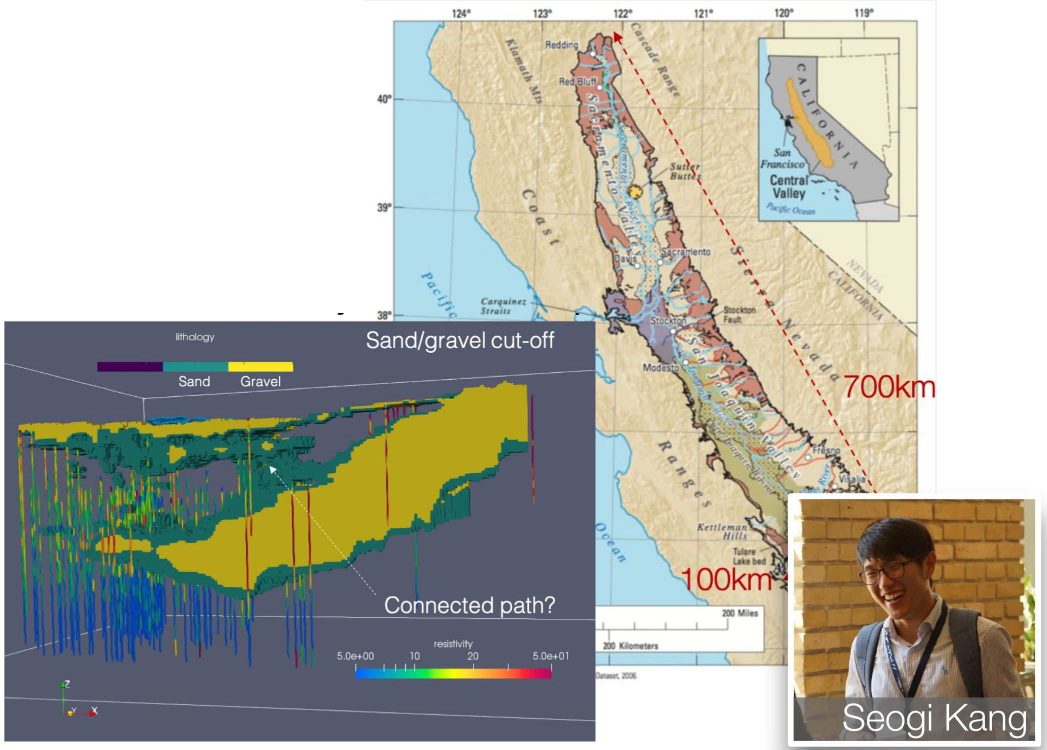
Now What?

What are results, and conclusions in this context?

Data?



Models?



what are “models” made of?

- Algorithmic ideas
- Mathematical structure (choices of features, etc.)
- Data to feed them!

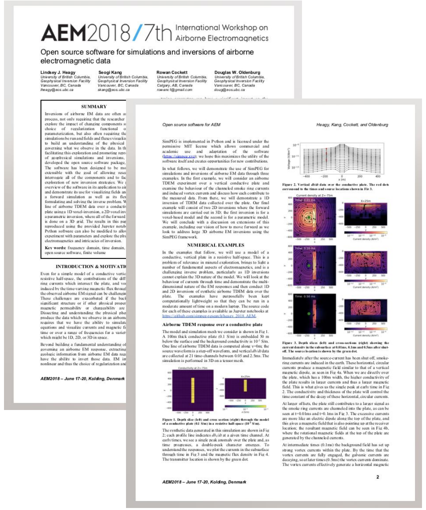
Today, “model” often refers to an “embodied model” that has been “fed data”.

models → data → truth?

communication?

publication, teaching, ...

results



AEM2018/7th International Workshop on Airborne Electromagnetics

Open source software for simulations and inversions of airborne electromagnetic data

Lindsey J. Heagy
University of British Columbia, Geophysical Inversion Facility
Vancouver, BC, Canada
lheagy@eos.ubc.ca

Seogi Kang
University of British Columbia, Geophysical Inversion Facility
Vancouver, BC, Canada
skang@eos.ubc.ca

Rowan Cockett
University of British Columbia, Geophysical Inversion Facility
Calgary, AB, Canada
rowanc1@gmail.com

Douglas W. Oldenburg
University of British Columbia, Geophysical Inversion Facility
Vancouver, BC, Canada
doug@eos.ubc.ca

SUMMARY

Inversions of airborne EM data are often a process, not only requiring that the researcher explore the impact of changing components s choice of regularization functional o parameterization, but also often requiring tha simulations be run and fields and fluxes visualiz to build an understanding of the physical governing what we observe in the data. In th facilitating this exploration and promoting repr of geophysical simulations and inversions, developed the open source software package, The software has been designed to be mo extensible with the goal of allowing rese interrogate all of the components and to fac exploration of new inversion strategies. We overview of the software in its application to ai and demonstrate its use for visualizing fields an a forward simulation as well as its flex formulating and solving the inverse problem. W line of airborne TDEM data over a conducti plate using a 1D voxel-inversion, a 2D voxel inv a parametric inversion, where all of the forward is done on a 3D grid. The results in this pa reproduced using the provided Jupyter noteb Python software can also be modified to allo experiment with parameters and explore the ph electromagnetics and intricacies of inversion.

Key words: frequency domain, time domain, open source software, finite volume

INTRODUCTION & MOTIVATI

Even for a simple model of a conductive vertic resistive half-space, the contributions of the dif ring currents which intersect the plate, and vo induced by the time-varying magnetic flux throug the observed airborne EM signal can be challengin These challenges are exacerbated if the bac significant structure or if other physical prope magnetic permeability or chargeability are Dissecting and understanding the physical phe produce the data which we observe in an airborn requires that we have the ability to simulat equations and visualize currents and magnetic fi time or over a range of frequencies for a variet which might be 1D, 2D, or 3D in space.

Beyond building a fundamental understanding o governing an airborne EM response, extractin geologic information from airborne EM data requ have the ability to invert those data. EM in nonlinear and thus the choice of regularization an

SimPEG is implemented in Python and is licensed under the permissive MIT license which allows commercial and academic use and adaptation of the software (https://simpeg.xyz); we hope this maximizes the utility of the software itself and creates opportunities for new contributions.

In what follows, we will demonstrate the use of SimPEG for simulations and inversions of airborne EM data through three examples. In the first example, we will consider an airborne TDEM experiment over a vertical conductive plate and examine the behaviour of the channeled smoke ring currents and induced vortex currents and discuss how each contribute to the measured data. From there, we will demonstrate a 1D inversion of TDEM data collected over the plate. Our final example will consist of two 2D inversions where the forward simulations are carried out in 3D; the first inversion is for a voxel-based model and the second is for a parametric model. We will conclude with a discussion on extensions of this example, including our vision of how to move forward as we look to address large 3D airborne EM inversions using the SimPEG framework.

NUMERICAL EXAMPLES

In the examples that follow, we will use a model of a conductive, vertical plate in a resistive half-space. This is a problem of relevance in mineral exploration, brings to light a number of fundamental aspects of electromagnetics, and is a challenging inverse problem, particularly as 1D inversions cannot explain the 3D nature of the model. We will look at the behaviour of currents through time and demonstrate the multi-dimensional nature of the EM responses and then conduct 1D and 2D inversions of synthetic airborne TDEM data over the plate. The examples have purposefully been kept computationally lightweight so that they can be run in a moderate amount of time on a modern laptop. The source code for each of these examples is available as Jupyter notebooks at https://github.com/simpeg-research/heagy_2018_AEM.

Airborne TDEM response over a conductive plate

The model and simulation mesh we consider is shown in Fig 1. A 100m thick conductive plate (0.1 S/m) is embedded 50 m below the surface and the background conductivity is 10^{-5} S/m. One line of airborne TDEM data is computed along y=0m; the source waveform is a step-off waveform, and vertical db/dt data are collected at 21 time-channels between 0.05 and 2.5ms. The simulation is performed in 3D on a tensor mesh.

Figure 1. Depth slice (left) and cross section (right) through the model of a conductive plate (0.1 S/m) in a resistive half-space (10^{-5} S/m).

The synthetic data generated in this simulation are shown in Fig 2; each profile line indicates db_z/dt at a given time channel. At early times, we see a single peak anomaly over the plate and, as time progresses, a double-peak character emerges. To understand the responses, we plot the currents in the subsurface through time in Fig 3 and the magnetic flux density in Fig 4. The transmitter location is shown by the green dot.

Figure 2. Vertical db/dt data over the conductive plate. The red dots correspond to the times and source locations shown in Fig 3.

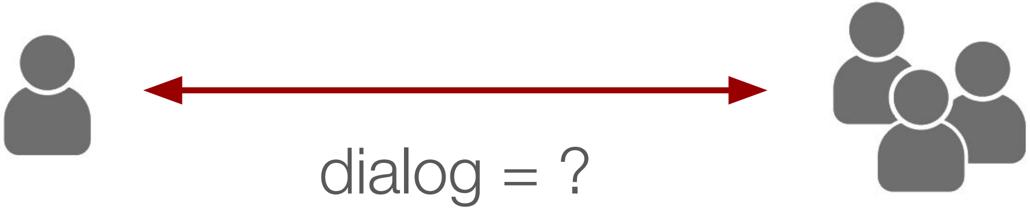
Figure 3. Depth slices (left) and cross-sections (right) showing the current density in the subsurface at 0.01ms, 0.1ms and 0.5ms after shut-off. The source location is shown by the green dot.

Immediately after the source-current has been shut off, smoke-ring currents are induced in the earth. These horizontal, circular currents produce a magnetic field similar to that of a vertical magnetic dipole, as seen in Fig 4a. When we are directly over the plate, which has a 100m width, the higher conductivity of the plate results in larger currents and thus a larger magnetic field. This is what gives us the single peak at early time in Fig 2. The conductivity and thickness of the plate will control the time constant of the decay of these horizontal, circular currents.

At larger offsets, the plate still contributes to a larger signal as the smoke ring currents are channeled into the plate, as can be seen at t=0.01ms and t=0.1ms in Fig 3. The excessive currents are more like an electric dipole along the top of the plate, and this gives a magnetic field that is also pointing up at the receiver location; the resultant magnetic field can be seen in Fig 4b, where the rotational magnetic fields at the top of the plate are generated by the channeled currents.

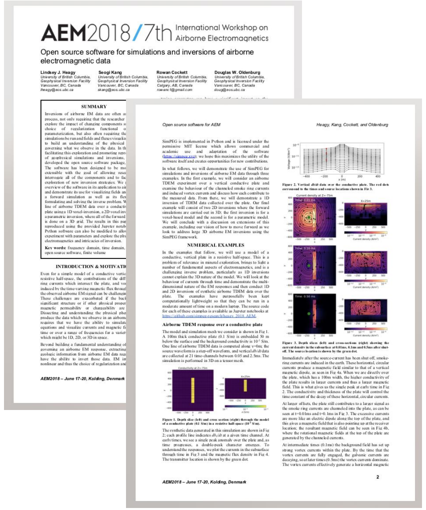
At intermediate times (0.1ms) the background field has set up strong vortex currents within the plate. By the time that the vortex currents are fully engaged, the galvanic currents are decaying, so at later times (0.5ms) the vortex currents dominate. The vortex currents effectively generate a horizontal magnetic

models → data → truth?



dialog = ?

results



AEM2018/7th International Workshop on Airborne Electromagnetics

Open source software for simulations and inversions of airborne electromagnetic data

Lindsey J. Heagy
University of British Columbia, Geophysical Inversion Facility
Vancouver, BC, Canada
lheagy@eos.ubc.ca

Seogi Kang
University of British Columbia, Geophysical Inversion Facility
Vancouver, BC, Canada
skang@eos.ubc.ca

Rowan Cockett
University of British Columbia, Geophysical Inversion Facility
Calgary, AB, Canada
rowanc1@gmail.com

Douglas W. Oldenburg
University of British Columbia, Geophysical Inversion Facility
Vancouver, BC, Canada
doug@eos.ubc.ca

SUMMARY

Inversions of airborne EM data are often a
process, not only requiring that the researcher
explore the impact of changing components s
choice of regularization functional o
parameterization, but also often requiring tha
simulations be run and fields and fluxes visualiz
to build an understanding of the physical
governing what we observe in the data. In th
facilitating this exploration and promoting repro
of geophysical simulations and inversions,
developed the open source software package,
The software has been designed to be mo
extensible with the goal of allowing resea
interrogate all of the components and to fac
exploration of new inversion strategies. We
overview of the software in its application to ai
and demonstrate its use for visualizing fields an
a forward simulation as well as its flex
formulating and solving the inverse problem. W
line of airborne TDEM data over a conductiv
plate using a 1D voxel-inversion, a 2D voxel inv
a parametric inversion, where all of the forward
is done on a 3D grid. The results in this pap
reproduced using the provided Jupyter noteb
Python software can also be modified to allo
experiment with parameters and explore the phy
electromagnetics and intricacies of inversion.

Key words: frequency domain, time domain,
open source software, finite volume

INTRODUCTION & MOTIVATI

Even for a simple model of a conductive vertic
resistive half-space, the contributions of the diff
ring currents which intersect the plate, and vo
induced by the time-varying magnetic flux throug
the observed airborne EM signal can be challengin
These challenges are exacerbated if the back
significant structure or if other physical proper
magnetic permeability or chargeability are
Dissecting and understanding the physical phe
produce the data which we observe in an airborn
requires that we have the ability to simulat
equations and visualize currents and magnetic fi
time or over a range of frequencies for a variet
which might be 1D, 2D, or 3D in space.

Beyond building a fundamental understanding o
governing an airborne EM response, extracting
geologic information from airborne EM data requ
have the ability to invert those data. EM in
nonlinear and thus the choice of regularization and

tuning parameters can have a significant impact on the

SimPEG is implemented in Python and is licensed under the permissive MIT license which allows commercial and academic use and adaptation of the software (https://simpeg.xyz); we hope this maximizes the utility of the software itself and creates opportunities for new contributions.

In what follows, we will demonstrate the use of SimPEG for simulations and inversions of airborne EM data through three examples. In the first example, we will consider an airborne TDEM experiment over a vertical conductive plate and examine the behaviour of the channeled smoke ring currents and induced vortex currents and discuss how each contribute to the measured data. From there, we will demonstrate a 1D inversion of TDEM data collected over the plate. Our final example will consist of two 2D inversions where the forward simulations are carried out in 3D; the first inversion is for a voxel-based model and the second is for a parametric model. We will conclude with a discussion on extensions of this example, including our vision of how to move forward as we look to address large 3D airborne EM inversions using the SimPEG framework.

NUMERICAL EXAMPLES

In the examples that follow, we will use a model of a conductive, vertical plate in a resistive half-space. This is a problem of relevance in mineral exploration, brings to light a number of fundamental aspects of electromagnetics, and is a challenging inverse problem, particularly as 1D inversions cannot explain the 3D nature of the model. We will look at the behaviour of currents through time and demonstrate the multi-dimensional nature of the EM responses and then conduct 1D and 2D inversions of synthetic airborne TDEM data over the plate. The examples have purposefully been kept computationally lightweight so that they can be run in a moderate amount of time on a modern laptop. The source code for each of these examples is available as Jupyter notebooks at https://github.com/simpeg-research/heagy_2018_AEM.

Airborne TDEM response over a conductive plate

The model and simulation mesh we consider is shown in Fig 1. A 100m thick conductive plate (0.1 S/m) is embedded 50 m below the surface and the background conductivity is 10^{-5} S/m. One line of airborne TDEM data is computed along y=0m; the source waveform is a step-off waveform, and vertical *db/dt* data are collected at 21 time-channels between 0.05 and 2.5ms. The simulation is performed in 3D on a tensor mesh.

Figure 1. Depth slice (left) and cross section (right) through the model of a conductive plate (0.1 S/m) in a resistive half-space (10^{-5} S/m).

The synthetic data generated in this simulation are shown in Fig 2; each profile line indicates *db/dt* at a given time channel. At early times, we see a single peak anomaly over the plate and, as time progresses, a double-peak character emerges. To understand the responses, we plot the currents in the subsurface through time in Fig 3 and the magnetic flux density in Fig 4. The transmitter location is shown by the green dot.

Figure 2. Vertical *db/dt* data over the conductive plate. The red dots correspond to the times and source locations shown in Fig 3.

Figure 3. Depth slices (left) and cross-sections (right) showing the current density in the subsurface at 0.01ms, 0.1ms and 0.5ms after shut-off. The source location is shown by the green dot.

Immediately after the source-current has been shut off, smoke-ring currents are induced in the earth. These horizontal, circular currents produce a magnetic field similar to that of a vertical magnetic dipole, as seen in Fig 4a. When we are directly over the plate, which has a 100m width, the higher conductivity of the plate results in larger currents and thus a larger magnetic field. This is what gives us the single peak at early time in Fig 2. The conductivity and thickness of the plate will control the time constant of the decay of these horizontal, circular currents.

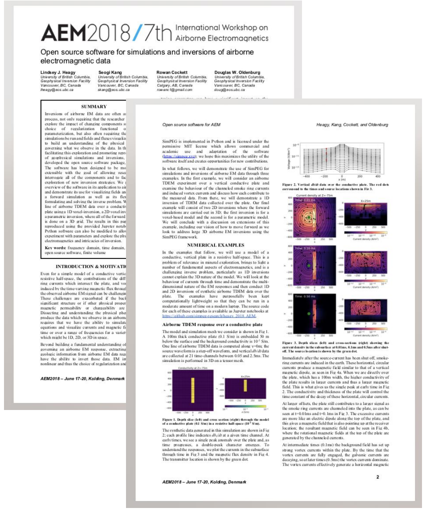
At larger offsets, the plate still contributes to a larger signal as the smoke ring currents are channeled into the plate, as can be seen at t=0.01ms and t=0.1ms in Fig 3. The excessive currents are more like an electric dipole along the top of the plate, and this gives a magnetic field that is also pointing up at the receiver location; the resultant magnetic field can be seen in Fig 4b, where the rotational magnetic fields at the top of the plate are generated by the channeled currents.

At intermediate times (0.1ms) the background field has set up strong vortex currents within the plate. By the time that the vortex currents are fully engaged, the galvanic currents are decaying, so at later times (0.5ms) the vortex currents dominate. The vortex currents effectively generate a horizontal magnetic

models → data → truth?

dialog = ?

results



AEM2018/7th International Workshop on Airborne Electromagnetics

Open source software for simulations and inversions of airborne electromagnetic data

Lindsey J. Heagy, University of British Columbia, Geophysical Inversion Facility, Vancouver, BC, Canada, lheagy@eos.ubc.ca
Seogi Kang, University of British Columbia, Geophysical Inversion Facility, Vancouver, BC, Canada, skang@eos.ubc.ca
Rowan Cockett, University of British Columbia, Geophysical Inversion Facility, Calgary, AB, Canada, rowanc1@gmail.com
Douglas W. Oldenburg, University of British Columbia, Geophysical Inversion Facility, Vancouver, BC, Canada, doug@eos.ubc.ca

SUMMARY

Inversions of airborne EM data are often a[illegible] process, not only requiring that the researcher [illegible] explore the impact of changing components s[illegible] choice of regularization functional o[illegible] parameterization, but also often requiring tha[illegible] simulations be run and fields and fluxes visualiz[illegible] to build an understanding of the physical [illegible] governing what we observe in the data. In th[illegible]

Open source software for AEM — Heagy, Kang, Cockett, and Oldenburg

SimPEG is implemented in Python and is licensed under the permissive MIT license which allows commercial and academic use and adaptation of the software [illegible]

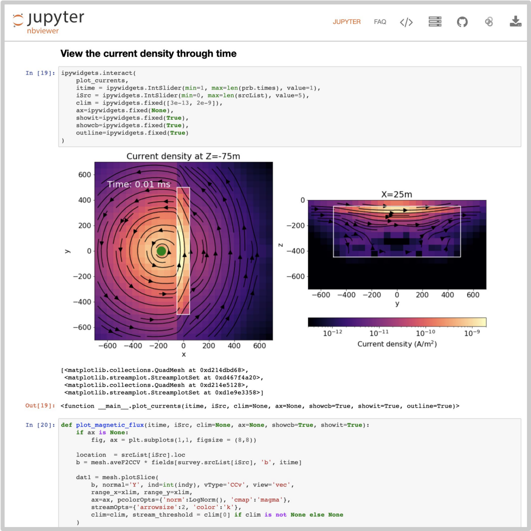
NUMERICAL EXAMPLES

Airborne TDEM response over a conductive plate

AEM2018 – June 17-20, Kolding, Denmark

AEM2018 – June 17-20, Kolding, Denmark

2



View the current density through time

```
In [19]: ipywidgets.interact(
             plot_currents,
             itime = ipywidgets.IntSlider(min=1, max=len(prb.times), value=1),
             iSrc = ipywidgets.IntSlider(min=0, max=len(srcList), value=5),
             clim = ipywidgets.fixed([3e-13, 2e-9]),
             ax=ipywidgets.fixed(None),
             showit=ipywidgets.fixed(True),
             showcb=ipywidgets.fixed(True),
             outline=ipywidgets.fixed(True)
         )
```

```
[<matplotlib.collections.QuadMesh at 0xd214dbd68>,
 <matplotlib.streamplot.StreamplotSet at 0xd467f4a20>,
 <matplotlib.collections.QuadMesh at 0xd214e5128>,
 <matplotlib.streamplot.StreamplotSet at 0xd1e9e3358>]

Out[19]: <function __main__.plot_currents(itime, iSrc, clim=None, ax=None, showcb=True, showit=True, outline=True)>
```

```
In [20]: def plot_magnetic_flux(itime, iSrc, clim=None, ax=None, showcb=True, showit=True):
             if ax is None:
                 fig, ax = plt.subplots(1,1, figsize = (8,8))

             location  = srcList[iSrc].loc
             b = mesh.aveF2CCV * fields[survey.srcList[iSrc], 'b', itime]

             dat1 = mesh.plotSlice(
                 b, normal='Y', ind=int(indy), vType='CCv', view='vec',
                 range_x=xlim, range_y=xlim,
                 ax=ax, pcolorOpts={'norm':LogNorm(), 'cmap':'magma'},
                 streamOpts={'arrowsize':2, 'color':'k'},
                 clim=clim, stream_threshold = clim[0] if clim is not None else None
             )
```

instructions, environment

the science more than the paper

An article about computational science in a scientific publication is not the scholarship itself, it is merely advertising of the scholarship. The actual scholarship is the complete software development environment and the complete set of instructions which generated the figures.

-- Buckheit and Donoho (paraphrasing Claerbout)
WaveLab and Reproducible Research, 1995

the science more than the paper

An article about computational science in a scientific publication is not the scholarship itself, it is merely advertising of the scholarship. The actual scholarship is the complete software development environment and the complete set of instructions which generated the figures.

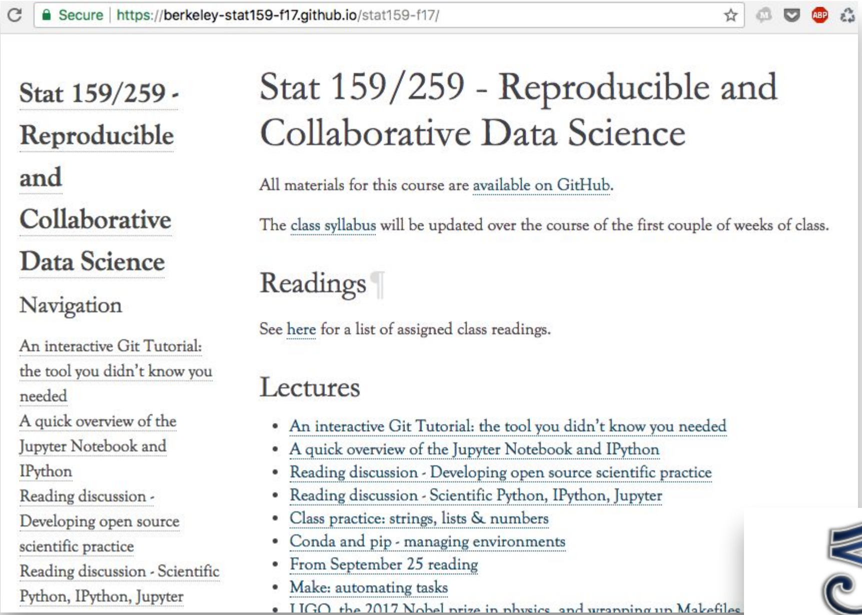
(and a place to run the code?)

-- Buckheit and Donoho (paraphrasing Claerbout)
WaveLab and Reproducible Research, 1995

Core skills

- Version control: Git and GitHub
- Programming: Python
- Process automation: Make
- Data analysis: Numpy, Pandas, Matplotlib, NLTK, Scikit-Learn, …
- Documentation: Sphinx
- Software testing: PyTest
- Continuous Integration: Travis
- Reproducible containers: Binder

Git and Python workflow everywhere



Secure | https://berkeley-stat159-f17.github.io/stat159-f17/

Stat 159/259 - Reproducible and Collaborative Data Science

Navigation

An interactive Git Tutorial: the tool you didn't know you needed
A quick overview of the Jupyter Notebook and IPython
Reading discussion - Developing open source scientific practice
Reading discussion - Scientific Python, IPython, Jupyter

Stat 159/259 - Reproducible and Collaborative Data Science

All materials for this course are available on GitHub.

The class syllabus will be updated over the course of the first couple of weeks of class.

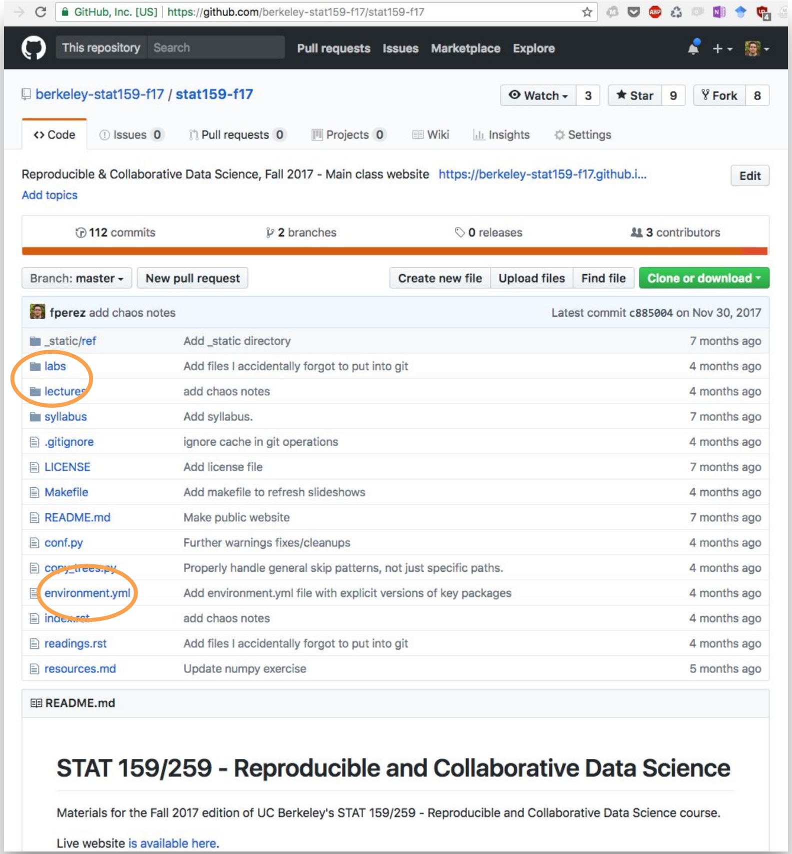
Readings¶

See here for a list of assigned class readings.

Lectures

- An interactive Git Tutorial: the tool you didn't know you needed
- A quick overview of the Jupyter Notebook and IPython
- Reading discussion - Developing open source scientific practice
- Reading discussion - Scientific Python, IPython, Jupyter
- Class practice: strings, lists & numbers
- Conda and pip - managing environments
- From September 25 reading
- Make: automating tasks
- LIGO, the 2017 Nobel prize in physics, and wrapping up Makefiles

SPHINX
PYTHON DOCUMENTATION GENERATOR



GitHub, Inc. [US] | https://github.com/berkeley-stat159-f17/stat159-f17

berkeley-stat159-f17 / stat159-f17

Reproducible & Collaborative Data Science, Fall 2017 - Main class website https://berkeley-stat159-f17.github.i...

Add topics

112 commits | 2 branches | 0 releases | 3 contributors

fperez add chaos notes — Latest commit c885004 on Nov 30, 2017

_static/ref	Add _static directory	7 months ago
labs	Add files I accidentally forgot to put into git	4 months ago
lectures	add chaos notes	4 months ago
syllabus	Add syllabus.	7 months ago
.gitignore	ignore cache in git operations	4 months ago
LICENSE	Add license file	7 months ago
Makefile	Add makefile to refresh slideshows	4 months ago
README.md	Make public website	7 months ago
conf.py	Further warnings fixes/cleanups	4 months ago
copy_trees.py	Properly handle general skip patterns, not just specific paths.	4 months ago
environment.yml	Add environment.yml file with explicit versions of key packages	4 months ago
index.rst	add chaos notes	4 months ago
readings.rst	Add files I accidentally forgot to put into git	4 months ago
resources.md	Update numpy exercise	5 months ago

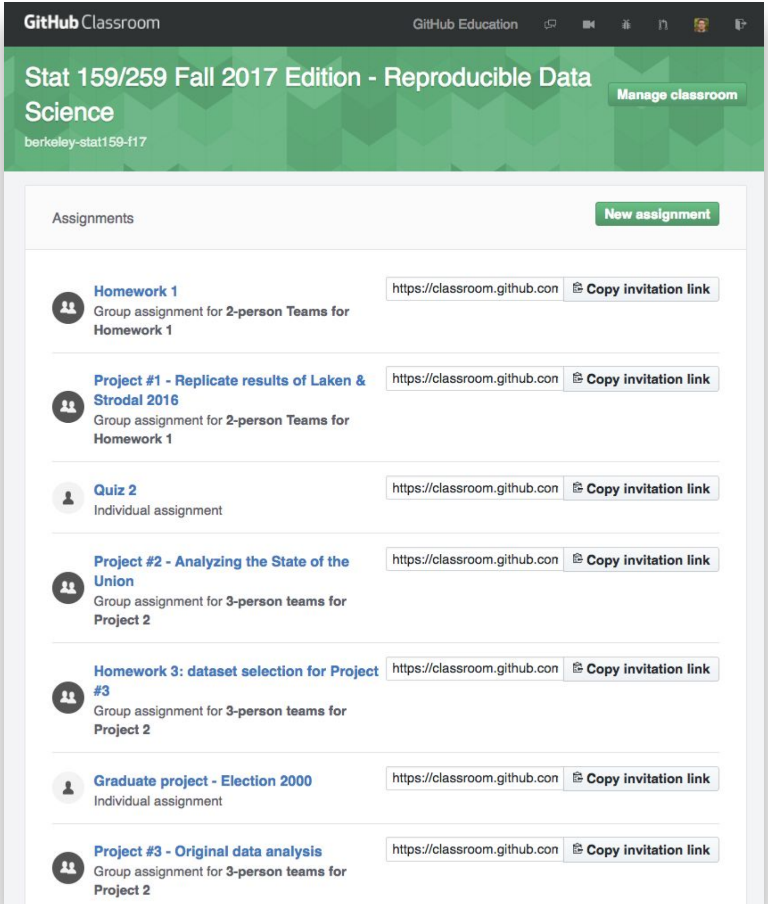
README.md

STAT 159/259 - Reproducible and Collaborative Data Science

Materials for the Fall 2017 edition of UC Berkeley's STAT 159/259 - Reproducible and Collaborative Data Science course.

Live website is available here.

Computational hygiene: a daily habit



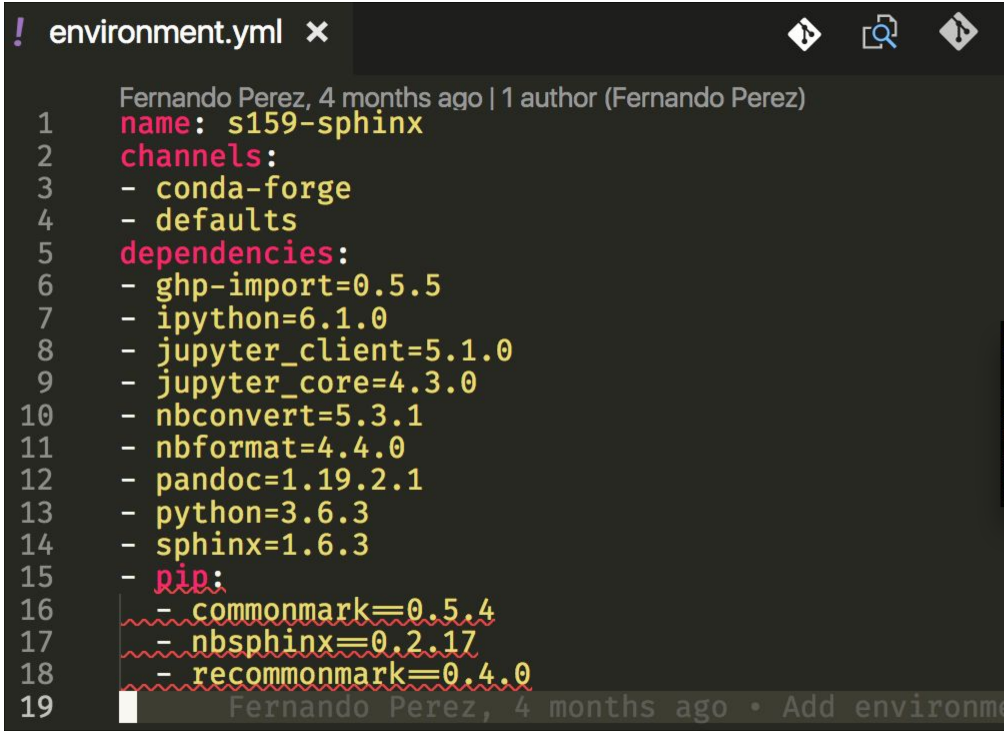
Stat 159/259 Fall 2017 Edition - Reproducible Data Science

berkeley-stat159-f17

Assignments

Homework 1 Group assignment for **2-person Teams for Homework 1**	https://classroom.github.con	Copy invitation link
Project #1 - Replicate results of Laken & Strodal 2016 Group assignment for **2-person Teams for Homework 1**	https://classroom.github.con	Copy invitation link
Quiz 2 Individual assignment	https://classroom.github.con	Copy invitation link
Project #2 - Analyzing the State of the Union Group assignment for **3-person teams for Project 2**	https://classroom.github.con	Copy invitation link
Homework 3: dataset selection for Project #3 Group assignment for **3-person teams for Project 2**	https://classroom.github.con	Copy invitation link
Graduate project - Election 2000 Individual assignment	https://classroom.github.con	Copy invitation link
Project #3 - Original data analysis Group assignment for **3-person teams for Project 2**	https://classroom.github.con	Copy invitation link

Explicit dependency management

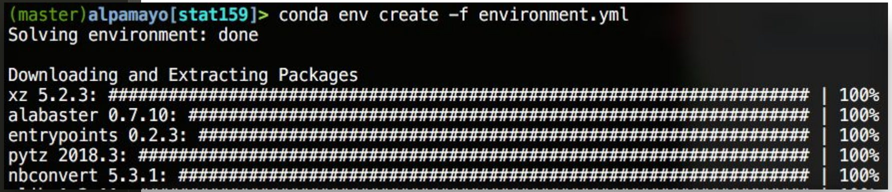


environment.yml

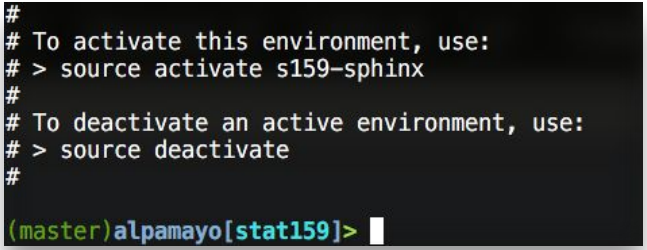
Fernando Perez, 4 months ago | 1 author (Fernando Perez)

```yaml
name: s159-sphinx
channels:
- conda-forge
- defaults
dependencies:
- ghp-import=0.5.5
- ipython=6.1.0
- jupyter_client=5.1.0
- jupyter_core=4.3.0
- nbconvert=5.3.1
- nbformat=4.4.0
- pandoc=1.19.2.1
- python=3.6.3
- sphinx=1.6.3
- pip:
  - commonmark==0.5.4
  - nbsphinx==0.2.17
  - recommonmark==0.4.0
```

Fernando Perez, 4 months ago • Add environment.yml

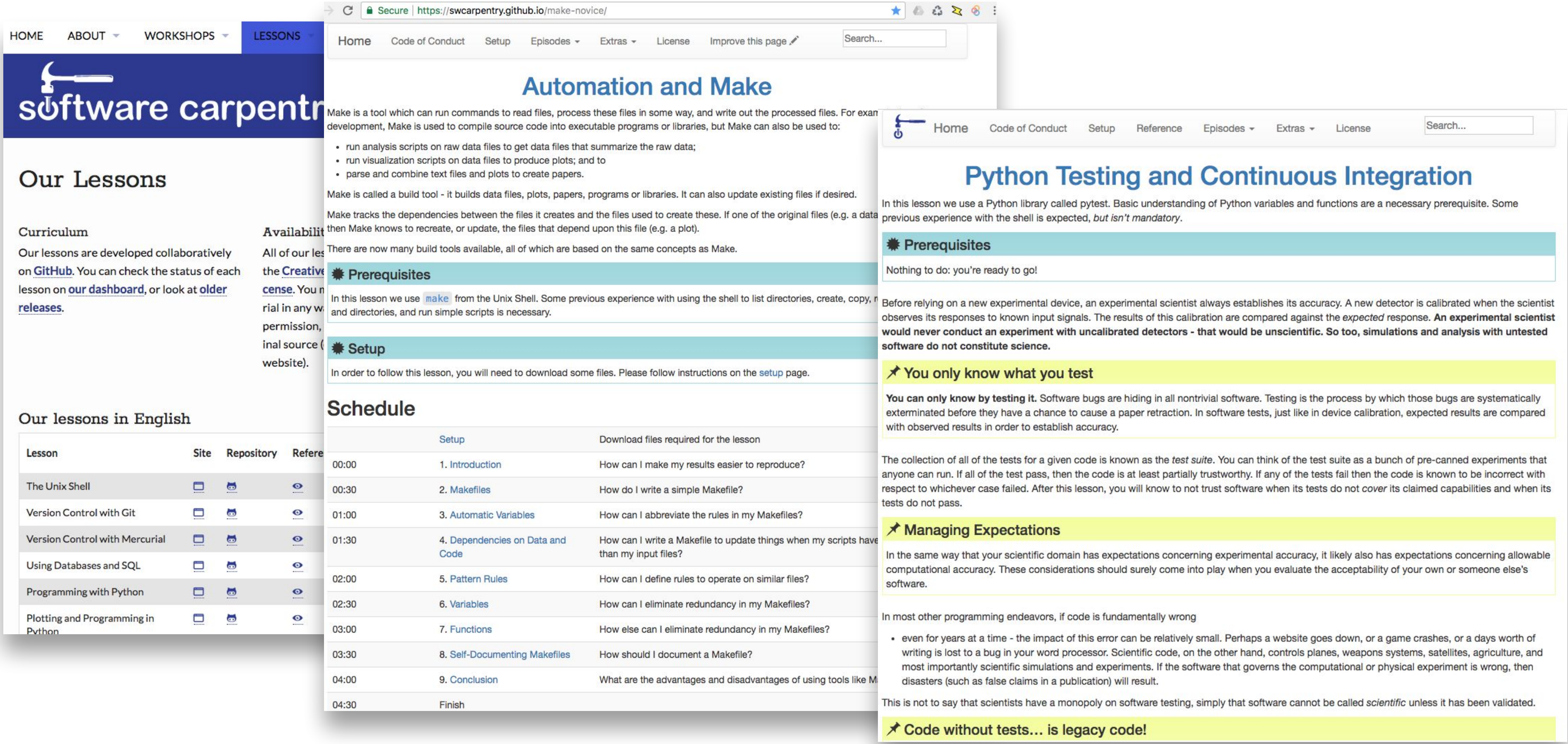


```
(master)alpamayo[stat159]> conda env create -f environment.yml
Solving environment: done

Downloading and Extracting Packages
xz 5.2.3: ##################################################################### | 100%
alabaster 0.7.10: ############################################################## | 100%
entrypoints 0.2.3: ############################################################# | 100%
pytz 2018.3: ################################################################### | 100%
nbconvert 5.3.1: ############################################################### | 100%
```



```
#
# To activate this environment, use:
# > source activate s159-sphinx
#
# To deactivate an active environment, use:
# > source deactivate
#

(master)alpamayo[stat159]>
```

Automation and Testing: SW Carpentry



Our Lessons

Curriculum

Our lessons are developed collaboratively on GitHub. You can check the status of each lesson on our dashboard, or look at older releases.

Availabilit

All of our les
the Creative
cense. You n
rial in any w
permission,
inal source (
website).

Our lessons in English

Lesson	Site	Repository	Refere
The Unix Shell	🗔	🐱	👁
Version Control with Git	🗔	🐱	👁
Version Control with Mercurial	🗔	🐱	👁
Using Databases and SQL	🗔	🐱	👁
Programming with Python	🗔	🐱	👁
Plotting and Programming in Python	🗔	🐱	👁

Automation and Make

Make is a tool which can run commands to read files, process these files in some way, and write out the processed files. For exan development, Make is used to compile source code into executable programs or libraries, but Make can also be used to:

- run analysis scripts on raw data files to get data files that summarize the raw data;
- run visualization scripts on data files to produce plots; and to
- parse and combine text files and plots to create papers.

Make is called a build tool - it builds data files, plots, papers, programs or libraries. It can also update existing files if desired.

Make tracks the dependencies between the files it creates and the files used to create these. If one of the original files (e.g. a data then Make knows to recreate, or update, the files that depend upon this file (e.g. a plot).

There are now many build tools available, all of which are based on the same concepts as Make.

Prerequisites

In this lesson we use `make` from the Unix Shell. Some previous experience with using the shell to list directories, create, copy, r and directories, and run simple scripts is necessary.

Setup

In order to follow this lesson, you will need to download some files. Please follow instructions on the setup page.

Schedule

	Setup	Download files required for the lesson
00:00	1. Introduction	How can I make my results easier to reproduce?
00:30	2. Makefiles	How do I write a simple Makefile?
01:00	3. Automatic Variables	How can I abbreviate the rules in my Makefiles?
01:30	4. Dependencies on Data and Code	How can I write a Makefile to update things when my scripts have than my input files?
02:00	5. Pattern Rules	How can I define rules to operate on similar files?
02:30	6. Variables	How can I eliminate redundancy in my Makefiles?
03:00	7. Functions	How else can I eliminate redundancy in my Makefiles?
03:30	8. Self-Documenting Makefiles	How should I document a Makefile?
04:00	9. Conclusion	What are the advantages and disadvantages of using tools like M
04:30	Finish	

Python Testing and Continuous Integration

In this lesson we use a Python library called pytest. Basic understanding of Python variables and functions are a necessary prerequisite. Some previous experience with the shell is expected, *but isn't mandatory*.

Prerequisites

Nothing to do: you're ready to go!

Before relying on a new experimental device, an experimental scientist always establishes its accuracy. A new detector is calibrated when the scientist observes its responses to known input signals. The results of this calibration are compared against the *expected* response. **An experimental scientist would never conduct an experiment with uncalibrated detectors - that would be unscientific. So too, simulations and analysis with untested software do not constitute science.**

You only know what you test

You can only know by testing it. Software bugs are hiding in all nontrivial software. Testing is the process by which those bugs are systematically exterminated before they have a chance to cause a paper retraction. In software tests, just like in device calibration, expected results are compared with observed results in order to establish accuracy.

The collection of all of the tests for a given code is known as the *test suite*. You can think of the test suite as a bunch of pre-canned experiments that anyone can run. If all of the test pass, then the code is at least partially trustworthy. If any of the tests fail then the code is known to be incorrect with respect to whichever case failed. After this lesson, you will know to not trust software when its tests do not *cover* its claimed capabilities and when its tests do not pass.

Managing Expectations

In the same way that your scientific domain has expectations concerning experimental accuracy, it likely also has expectations concerning allowable computational accuracy. These considerations should surely come into play when you evaluate the acceptability of your own or someone else's software.

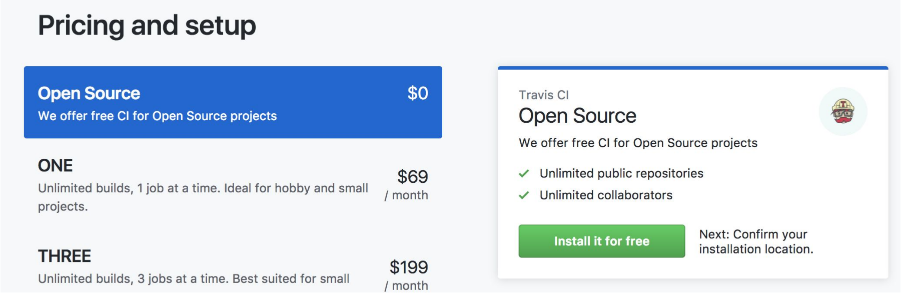
In most other programming endeavors, if code is fundamentally wrong

- even for years at a time - the impact of this error can be relatively small. Perhaps a website goes down, or a game crashes, or a days worth of writing is lost to a bug in your word processor. Scientific code, on the other hand, controls planes, weapons systems, satellites, agriculture, and most importantly scientific simulations and experiments. If the software that governs the computational or physical experiment is wrong, then disasters (such as false claims in a publication) will result.

This is not to say that scientists have a monopoly on software testing, simply that software cannot be called *scientific* unless it has been validated.

Code without tests... is legacy code!

Continuous Integration with Travis



Pricing and setup

Open Source $0
We offer free CI for Open Source projects

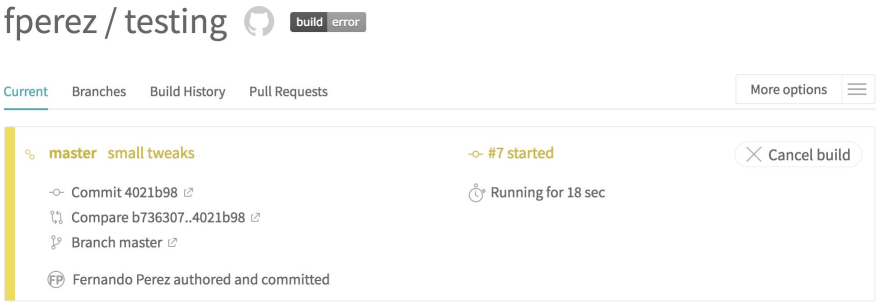
ONE $69 / month
Unlimited builds, 1 job at a time. Ideal for hobby and small projects.

THREE $199 / month
Unlimited builds, 3 jobs at a time. Best suited for small

Travis CI
Open Source
We offer free CI for Open Source projects

- ✓ Unlimited public repositories
- ✓ Unlimited collaborators

Next: Confirm your installation location.

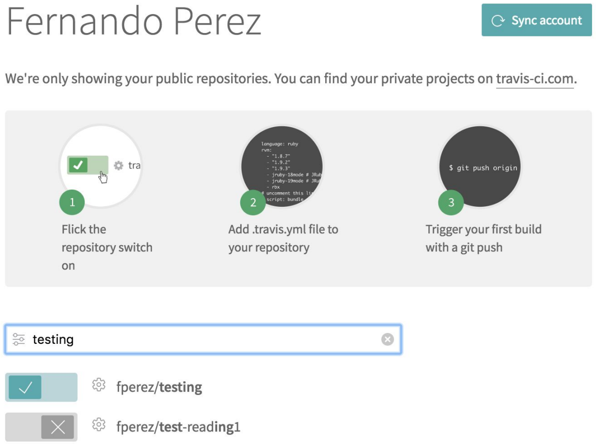


fperez / testing

master small tweaks #7 started

Commit 4021b98
Compare b736307..4021b98
Branch master
Running for 18 sec

FP Fernando Perez authored and committed



Fernando Perez

We're only showing your public repositories. You can find your private projects on travis-ci.com.

1 Flick the repository switch on

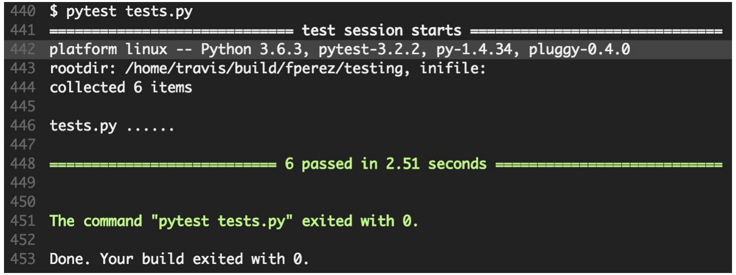
2 Add .travis.yml file to your repository

3 Trigger your first build with a git push

testing

☑ fperez/testing

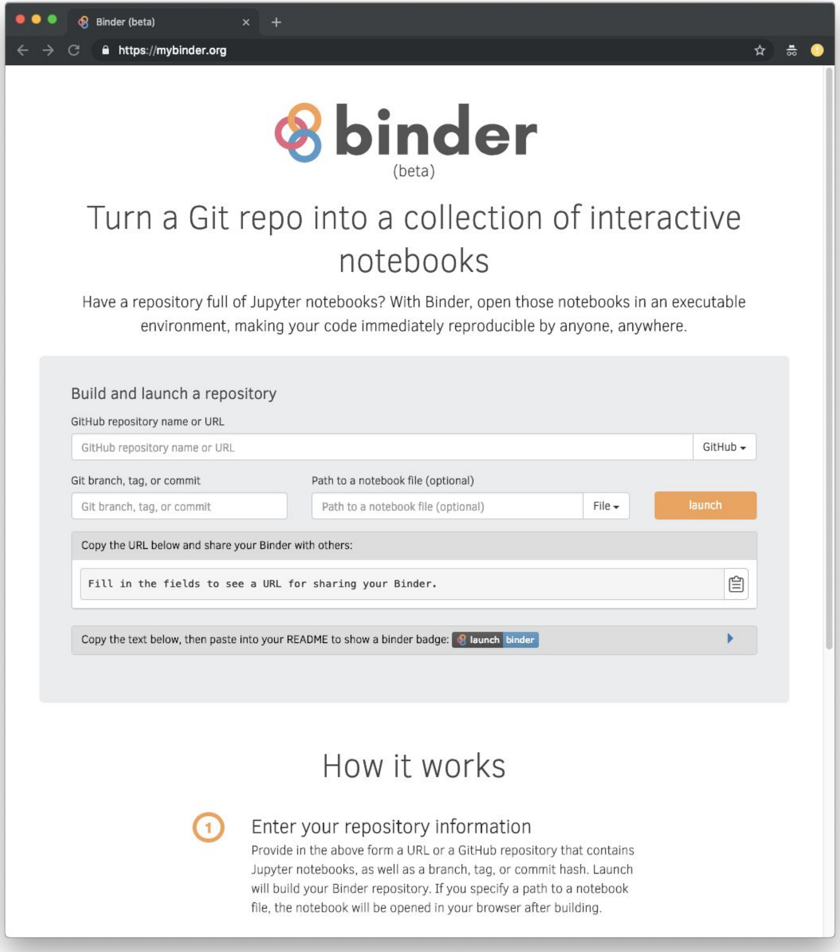
☒ fperez/test-reading1



```
$ pytest tests.py
============================= test session starts ==============================
platform linux -- Python 3.6.3, pytest-3.2.2, py-1.4.34, pluggy-0.4.0
rootdir: /home/travis/build/fperez/testing, inifile:
collected 6 items

tests.py ......

=========================== 6 passed in 2.51 seconds ===========================


The command "pytest tests.py" exited with 0.

Done. Your build exited with 0.
```

binder

shareable, interactive, reproducible
environments from your public git repository



binder
(beta)

Turn a Git repo into a collection of interactive notebooks

Have a repository full of Jupyter notebooks? With Binder, open those notebooks in an executable environment, making your code immediately reproducible by anyone, anywhere.

Build and launch a repository

GitHub repository name or URL

GitHub repository name or URL | GitHub ▾

Git branch, tag, or commit

Git branch, tag, or commit

Path to a notebook file (optional)

Path to a notebook file (optional) | File ▾ | launch

Copy the URL below and share your Binder with others:

```
Fill in the fields to see a URL for sharing your Binder.
```

Copy the text below, then paste into your README to show a binder badge: launch binder

How it works

1 Enter your repository information

Provide in the above form a URL or a GitHub repository that contains Jupyter notebooks, as well as a branch, tag, or commit hash. Launch will build your Binder repository. If you specify a path to a notebook file, the notebook will be opened in your browser after building.

mybinder.org

Black holes! LIGO, Sept 14, 2015

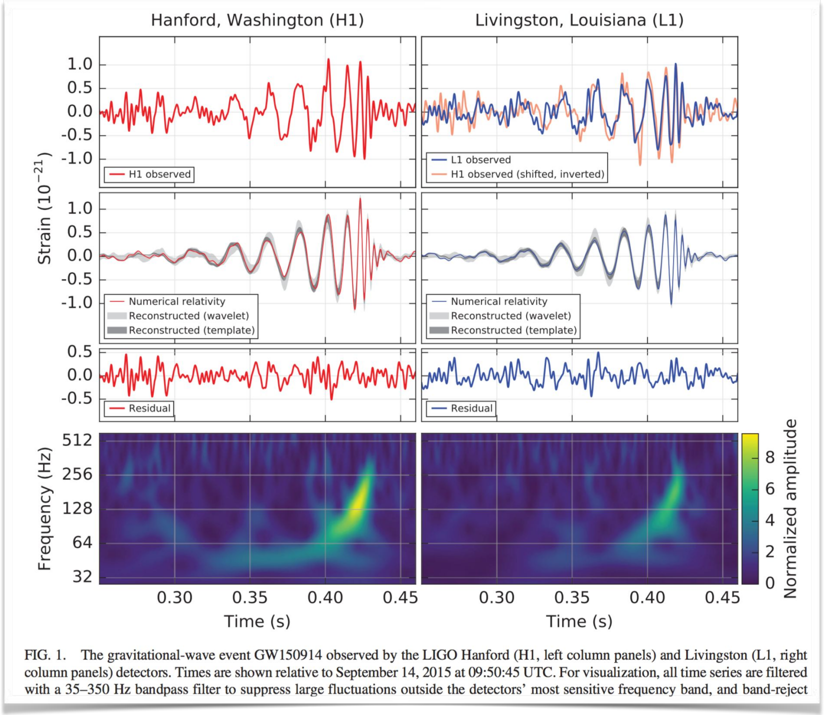
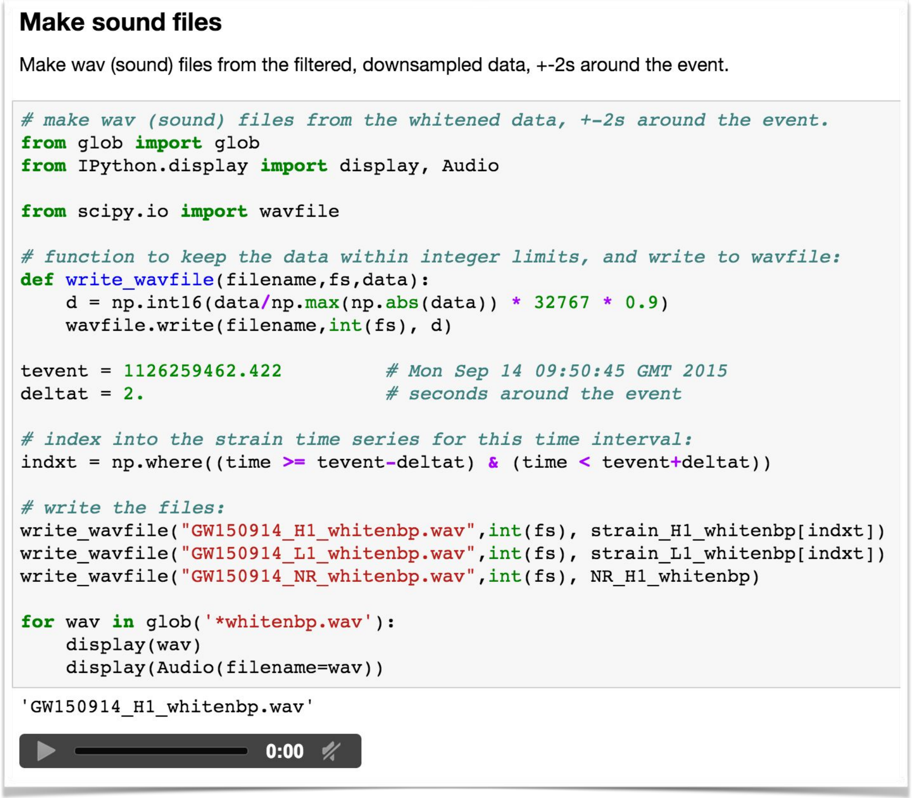


FIG. 1. The gravitational-wave event GW150914 observed by the LIGO Hanford (H1, left column panels) and Livingston (L1, right column panels) detectors. Times are shown relative to September 14, 2015 at 09:50:45 UTC. For visualization, all time series are filtered with a 35–350 Hz bandpass filter to suppress large fluctuations outside the detectors' most sensitive frequency band, and band-reject



Make sound files

Make wav (sound) files from the filtered, downsampled data, +-2s around the event.

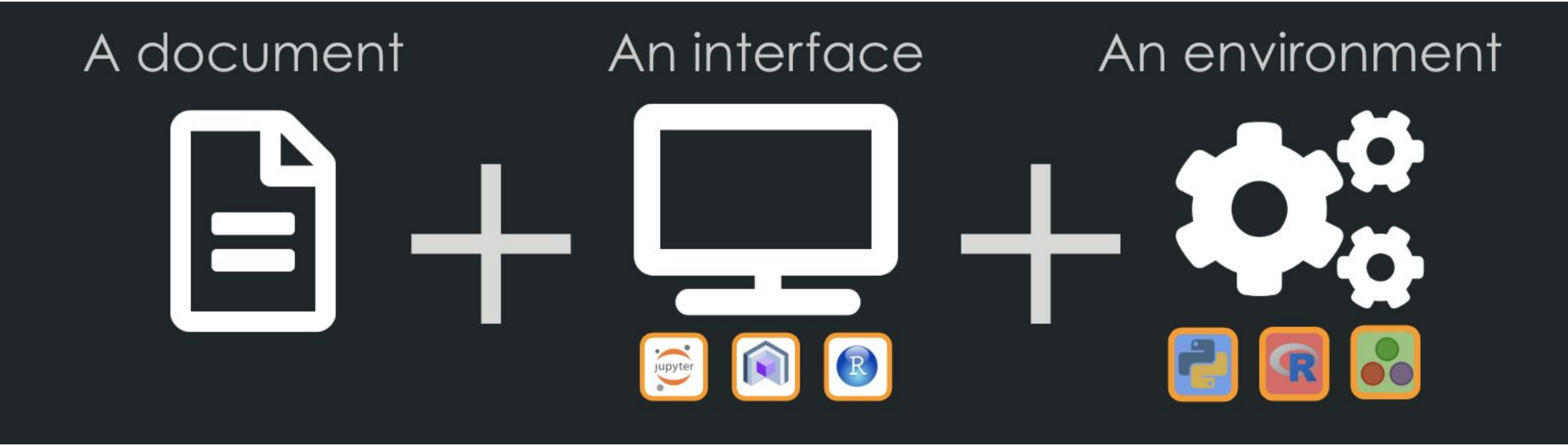
```
# make wav (sound) files from the whitened data, +-2s around the event.
from glob import glob
from IPython.display import display, Audio

from scipy.io import wavfile

# function to keep the data within integer limits, and write to wavfile:
def write_wavfile(filename,fs,data):
    d = np.int16(data/np.max(np.abs(data)) * 32767 * 0.9)
    wavfile.write(filename,int(fs), d)

tevent = 1126259462.422         # Mon Sep 14 09:50:45 GMT 2015
deltat = 2.                     # seconds around the event

# index into the strain time series for this time interval:
indxt = np.where((time >= tevent-deltat) & (time < tevent+deltat))

# write the files:
write_wavfile("GW150914_H1_whitenbp.wav",int(fs), strain_H1_whitenbp[indxt])
write_wavfile("GW150914_L1_whitenbp.wav",int(fs), strain_L1_whitenbp[indxt])
write_wavfile("GW150914_NR_whitenbp.wav",int(fs), NR_H1_whitenbp)

for wav in glob('*whitenbp.wav'):
    display(wav)
    display(Audio(filename=wav))
```

```
'GW150914_H1_whitenbp.wav'
```

http://bit.ly/black-holes-woop

complete set of instructions

capture the steps: what is a notebook?



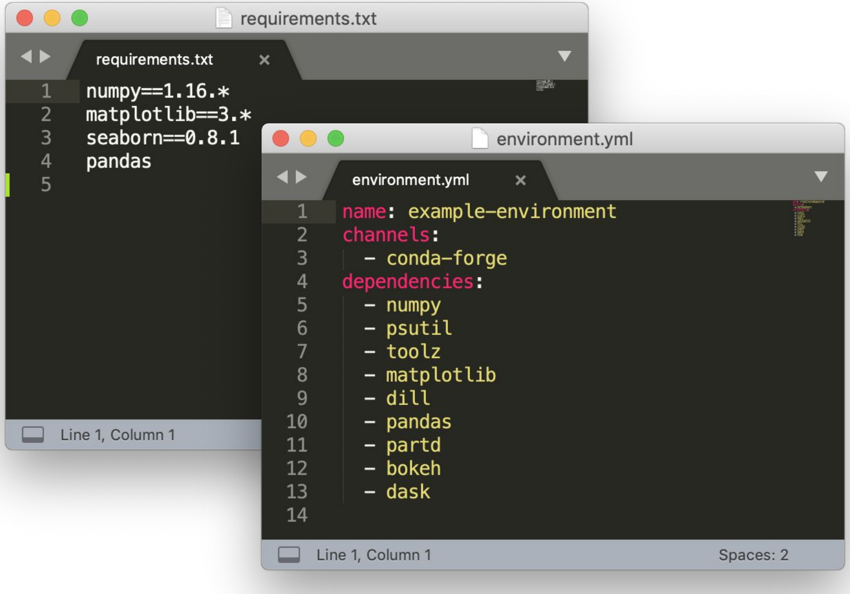
repo2docker

repo2docker deterministically build a docker image from a repository with documented dependencies

complete development environment

define dependencies following community standards of practice

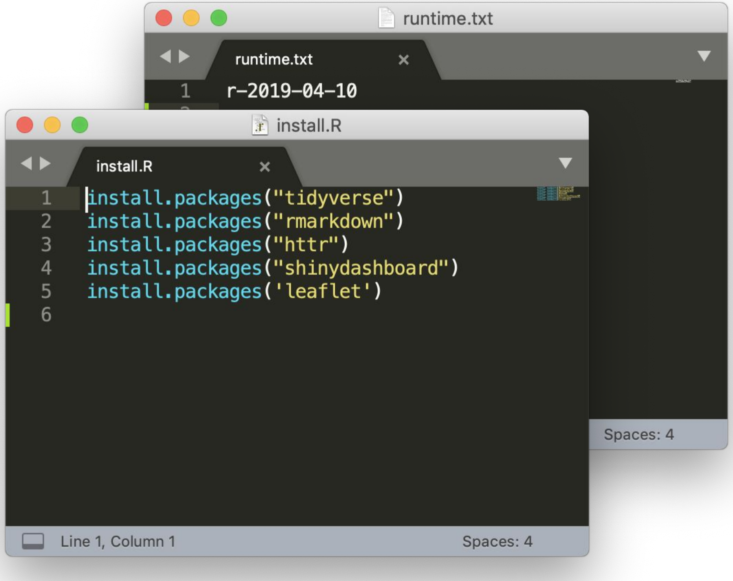
python™



```
numpy==1.16.*
matplotlib==3.*
seaborn==0.8.1
pandas

```

```
name: example-environment
channels:
  - conda-forge
dependencies:
  - numpy
  - psutil
  - toolz
  - matplotlib
  - dill
  - pandas
  - partd
  - bokeh
  - dask

```

R

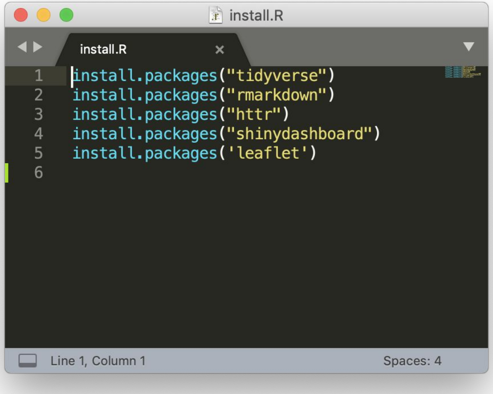


```
r-2019-04-10
```

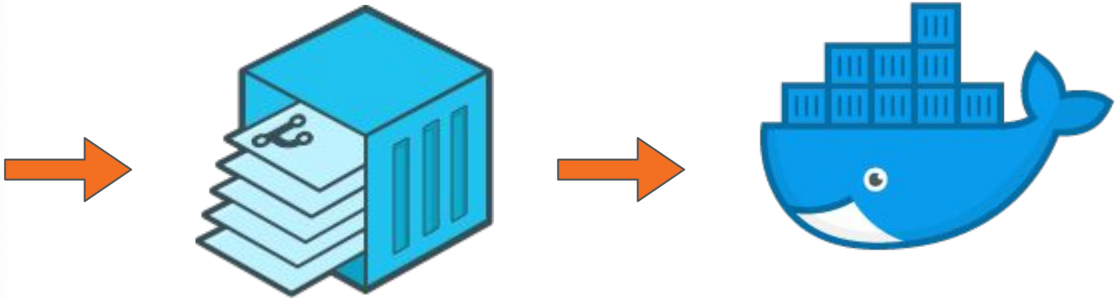
```
install.packages("tidyverse")
install.packages("rmarkdown")
install.packages("httr")
install.packages("shinydashboard")
install.packages('leaflet')

```

complete development environment

repo2docker



```r
install.packages("tidyverse")
install.packages("rmarkdown")
install.packages("httr")
install.packages("shinydashboard")
install.packages('leaflet')
```

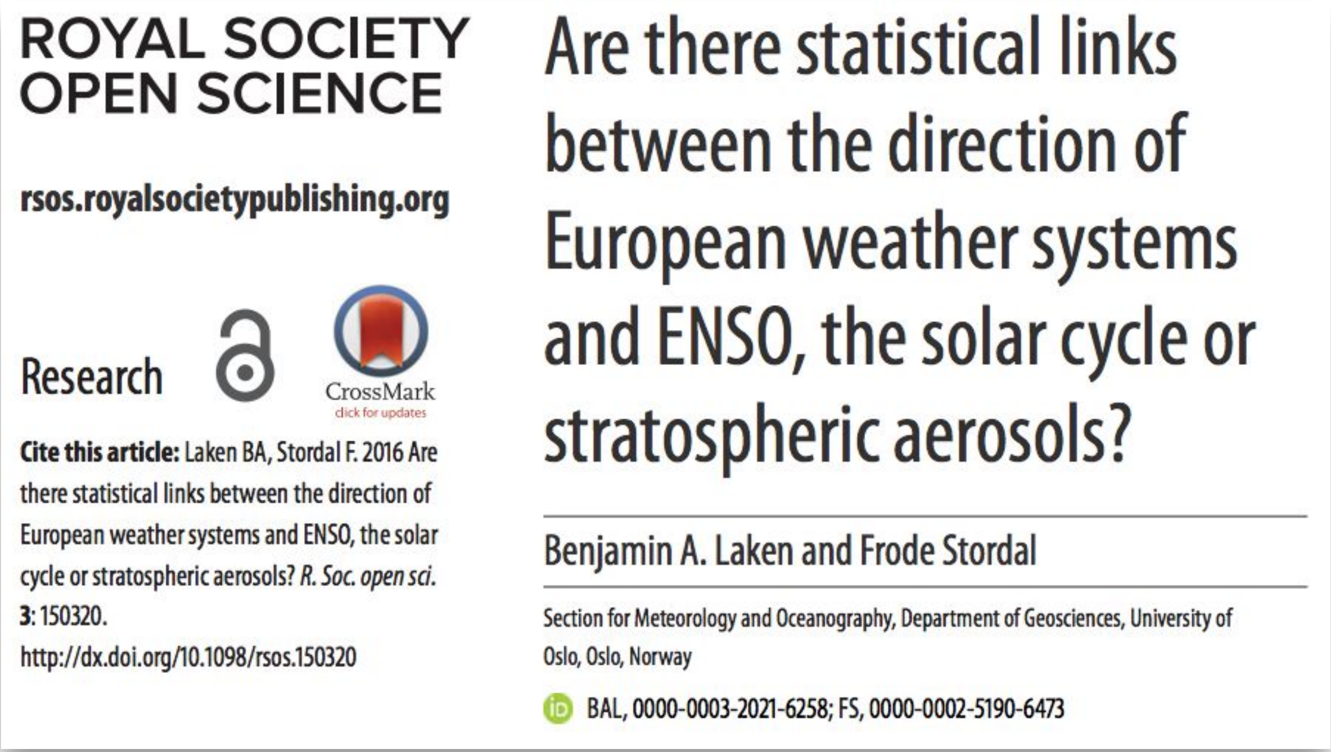


dependencies

repo2docker

container file

Example 1: real-world replication



ROYAL SOCIETY
OPEN SCIENCE

rsos.royalsocietypublishing.org

Research

CrossMark
click for updates

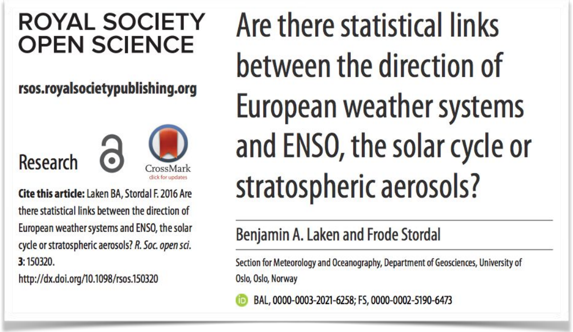
Cite this article: Laken BA, Stordal F. 2016 Are there statistical links between the direction of European weather systems and ENSO, the solar cycle or stratospheric aerosols? *R. Soc. open sci.* **3**: 150320.
http://dx.doi.org/10.1098/rsos.150320

Are there statistical links between the direction of European weather systems and ENSO, the solar cycle or stratospheric aerosols?

Benjamin A. Laken and Frode Stordal

Section for Meteorology and Oceanography, Department of Geosciences, University of Oslo, Oslo, Norway

iD BAL, 0000-0003-2021-6258; FS, 0000-0002-5190-6473



ROYAL SOCIETY OPEN SCIENCE

rsos.royalsocietypublishing.org

Research

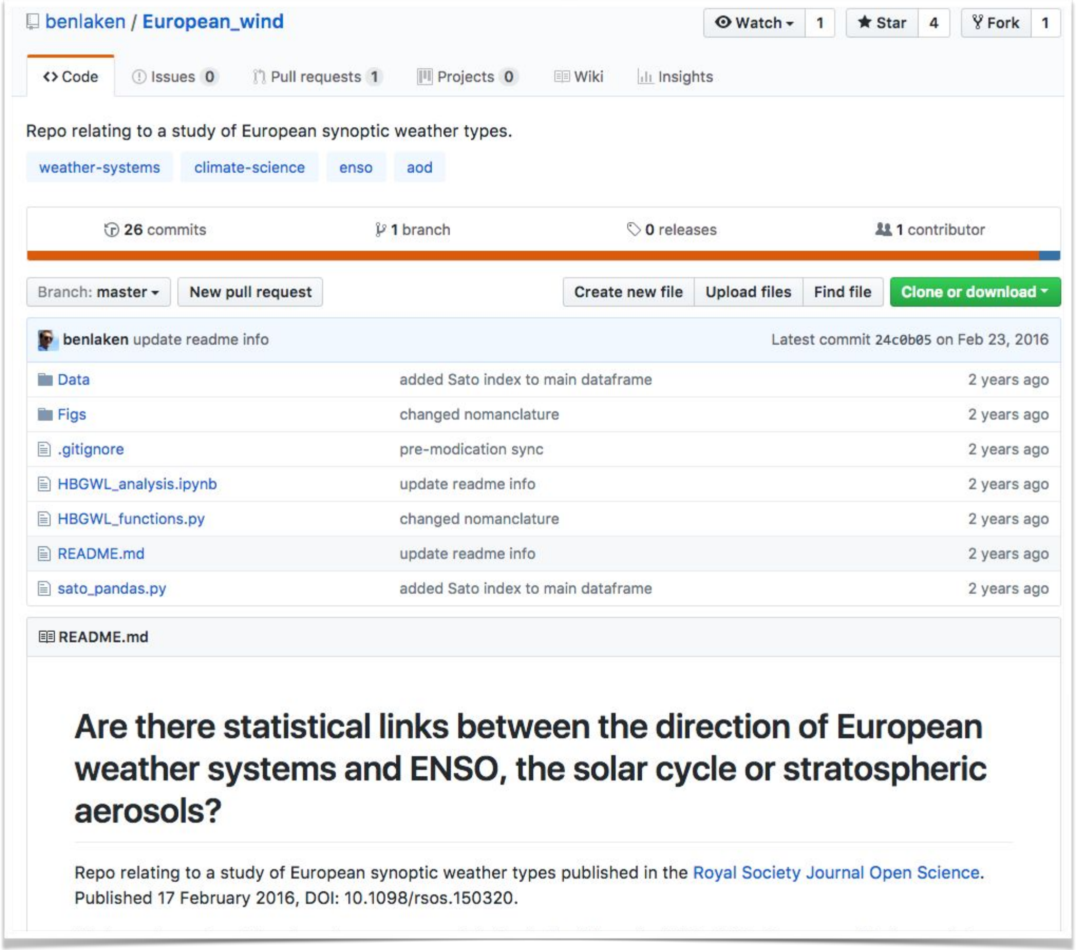
Cite this article: Laken BA, Stordal F. 2016 Are there statistical links between the direction of European weather systems and ENSO, the solar cycle or stratospheric aerosols? *R. Soc. open sci.* **3**: 150320.
http://dx.doi.org/10.1098/rsos.150320

Are there statistical links between the direction of European weather systems and ENSO, the solar cycle or stratospheric aerosols?

Benjamin A. Laken and Frode Stordal

Section for Meteorology and Oceanography, Department of Geosciences, University of Oslo, Oslo, Norway

BAL, 0000-0003-2021-6258; FS, 0000-0002-5190-6473



benlaken / European_wind

Repo relating to a study of European synoptic weather types.

26 commits · 1 branch · 0 releases · 1 contributor

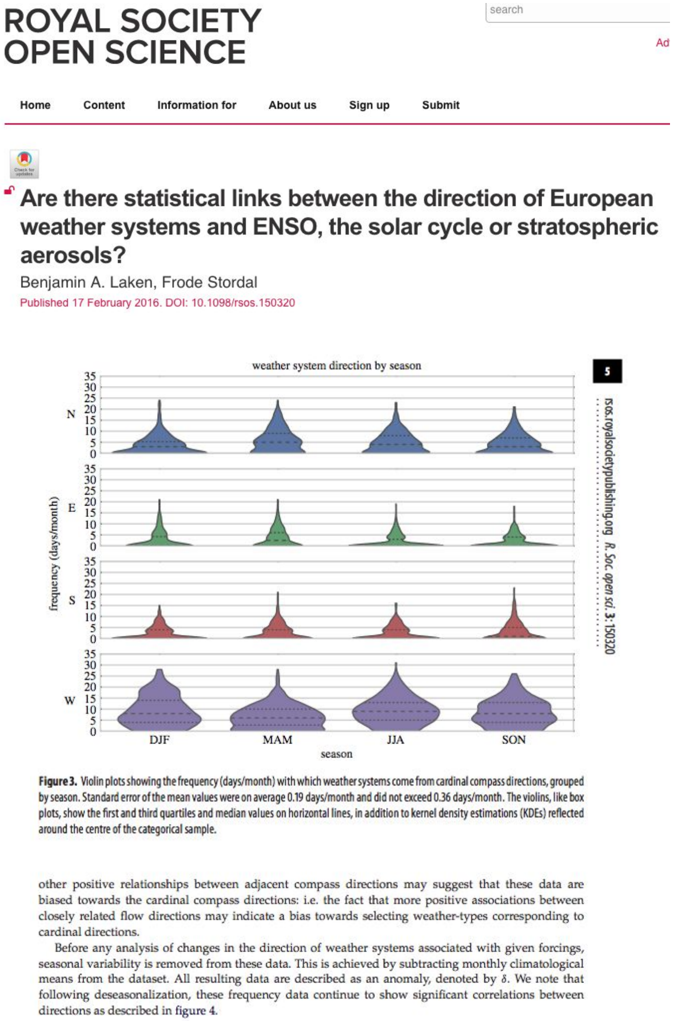
benlaken update readme info — Latest commit 24c0b05 on Feb 23, 2016

Data	added Sato index to main dataframe	2 years ago
Figs	changed nomanclature	2 years ago
.gitignore	pre-modication sync	2 years ago
HBGWL_analysis.ipynb	update readme info	2 years ago
HBGWL_functions.py	changed nomanclature	2 years ago
README.md	update readme info	2 years ago
sato_pandas.py	added Sato index to main dataframe	2 years ago

README.md

Are there statistical links between the direction of European weather systems and ENSO, the solar cycle or stratospheric aerosols?

Repo relating to a study of European synoptic weather types published in the Royal Society Journal Open Science. Published 17 February 2016, DOI: 10.1098/rsos.150320.



ROYAL SOCIETY OPEN SCIENCE

Are there statistical links between the direction of European weather systems and ENSO, the solar cycle or stratospheric aerosols?

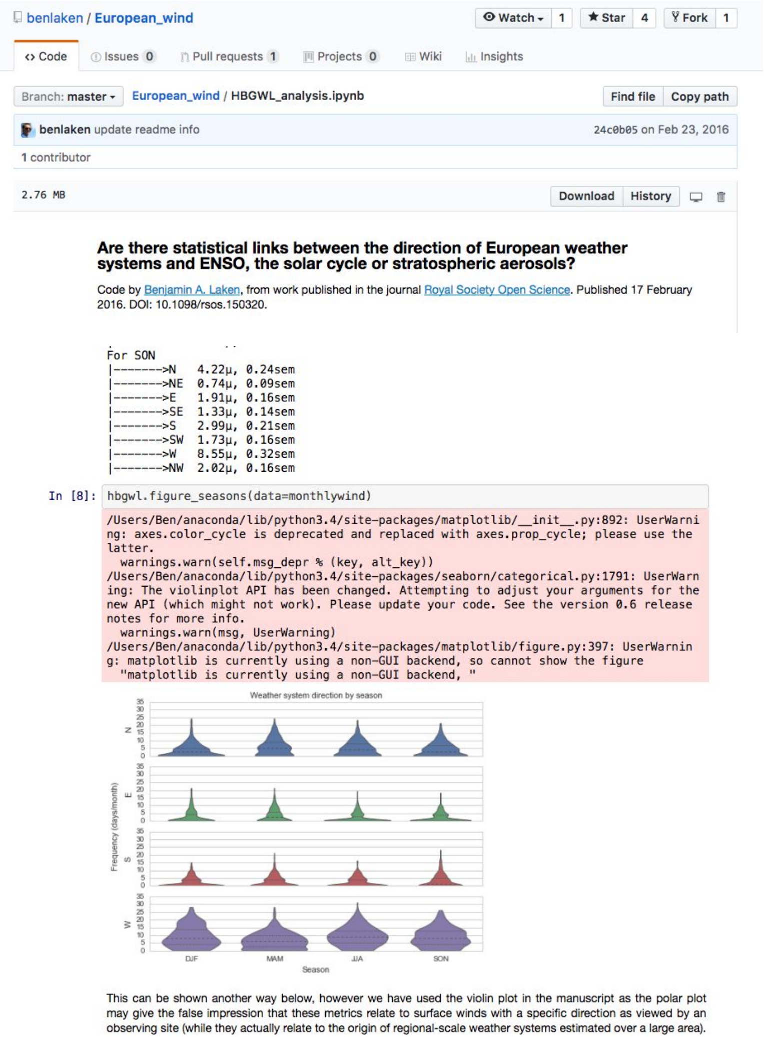
Benjamin A. Laken, Frode Stordal

Published 17 February 2016. DOI: 10.1098/rsos.150320

Figure 3. Violin plots showing the frequency (days/month) with which weather systems come from cardinal compass directions, grouped by season. Standard error of the mean values were on average 0.19 days/month and did not exceed 0.36 days/month. The violins, like box plots, show the first and third quartiles and median values on horizontal lines, in addition to kernel density estimations (KDEs) reflected around the centre of the categorical sample.

other positive relationships between adjacent compass directions may suggest that these data are biased towards the cardinal compass directions: i.e. the fact that more positive associations between closely related flow directions may indicate a bias towards selecting weather-types corresponding to cardinal directions.

Before any analysis of changes in the direction of weather systems associated with given forcings, seasonal variability is removed from these data. This is achieved by subtracting monthly climatological means from the dataset. All resulting data are described as an anomaly, denoted by δ. We note that following deseasonalization, these frequency data continue to show significant correlations between directions as described in figure 4.



Are there statistical links between the direction of European weather systems and ENSO, the solar cycle or stratospheric aerosols?

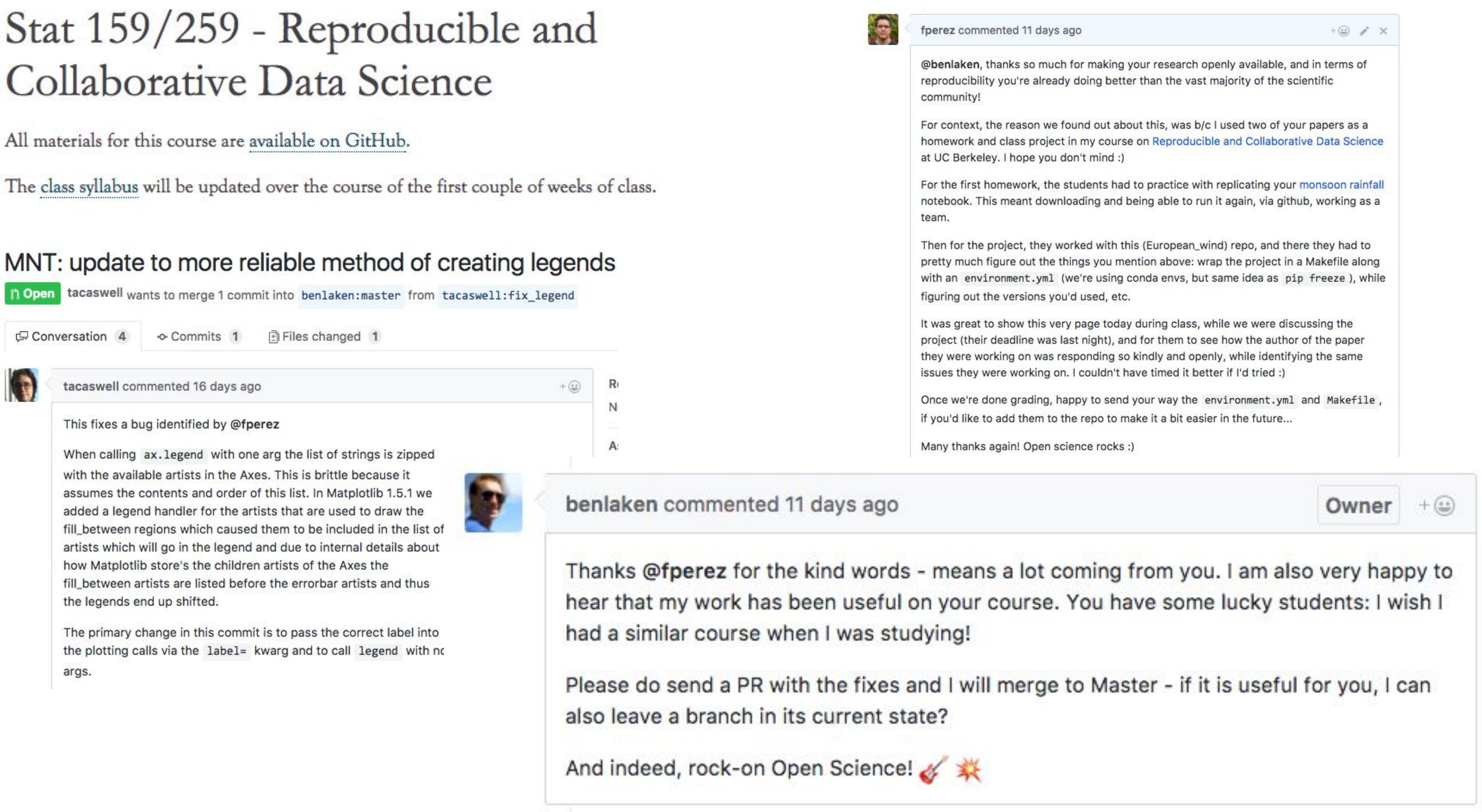
Code by Benjamin A. Laken, from work published in the journal Royal Society Open Science. Published 17 February 2016. DOI: 10.1098/rsos.150320.

```
For SON
|------->N   4.22μ, 0.24sem
|------->NE  0.74μ, 0.09sem
|------->E   1.91μ, 0.16sem
|------->SE  1.33μ, 0.14sem
|------->S   2.99μ, 0.21sem
|------->SW  1.73μ, 0.16sem
|------->W   8.55μ, 0.32sem
|------->NW  2.02μ, 0.16sem
```

```
In [8]: hbgwl.figure_seasons(data=monthlywind)
```

```
/Users/Ben/anaconda/lib/python3.4/site-packages/matplotlib/__init__.py:892: UserWarni
ng: axes.color_cycle is deprecated and replaced with axes.prop_cycle; please use the
latter.
  warnings.warn(self.msg_depr % (key, alt_key))
/Users/Ben/anaconda/lib/python3.4/site-packages/seaborn/categorical.py:1791: UserWarn
ing: The violinplot API has been changed. Attempting to adjust your arguments for the
new API (which might not work). Please update your code. See the version 0.6 release
notes for more info.
  warnings.warn(msg, UserWarning)
/Users/Ben/anaconda/lib/python3.4/site-packages/matplotlib/figure.py:397: UserWarnin
g: matplotlib is currently using a non-GUI backend, so cannot show the figure
  "matplotlib is currently using a non-GUI backend, "
```

This can be shown another way below, however we have used the violin plot in the manuscript as the polar plot may give the false impression that these metrics relate to surface winds with a specific direction as viewed by an observing site (while they actually relate to the origin of regional-scale weather systems estimated over a large area).



Stat 159/259 - Reproducible and Collaborative Data Science

All materials for this course are available on GitHub.

The class syllabus will be updated over the course of the first couple of weeks of class.

MNT: update to more reliable method of creating legends

tacaswell wants to merge 1 commit into `benlaken:master` from `tacaswell:fix_legend`

tacaswell commented 16 days ago

This fixes a bug identified by **@fperez**

When calling `ax.legend` with one arg the list of strings is zipped with the available artists in the Axes. This is brittle because it assumes the contents and order of this list. In Matplotlib 1.5.1 we added a legend handler for the artists that are used to draw the fill_between regions which caused them to be included in the list of artists which will go in the legend and due to internal details about how Matplotlib store's the children artists of the Axes the fill_between artists are listed before the errorbar artists and thus the legends end up shifted.

The primary change in this commit is to pass the correct label into the plotting calls via the `label=` kwarg and to call `legend` with nc args.

fperez commented 11 days ago

@benlaken, thanks so much for making your research openly available, and in terms of reproducibility you're already doing better than the vast majority of the scientific community!

For context, the reason we found out about this, was b/c I used two of your papers as a homework and class project in my course on Reproducible and Collaborative Data Science at UC Berkeley. I hope you don't mind :)

For the first homework, the students had to practice with replicating your monsoon rainfall notebook. This meant downloading and being able to run it again, via github, working as a team.

Then for the project, they worked with this (European_wind) repo, and there they had to pretty much figure out the things you mention above: wrap the project in a Makefile along with an `environment.yml` (we're using conda envs, but same idea as `pip freeze`), while figuring out the versions you'd used, etc.

It was great to show this very page today during class, while we were discussing the project (their deadline was last night), and for them to see how the author of the paper they were working on was responding so kindly and openly, while identifying the same issues they were working on. I couldn't have timed it better if I'd tried :)

Once we're done grading, happy to send your way the `environment.yml` and `Makefile` , if you'd like to add them to the repo to make it a bit easier in the future...

Many thanks again! Open science rocks :)

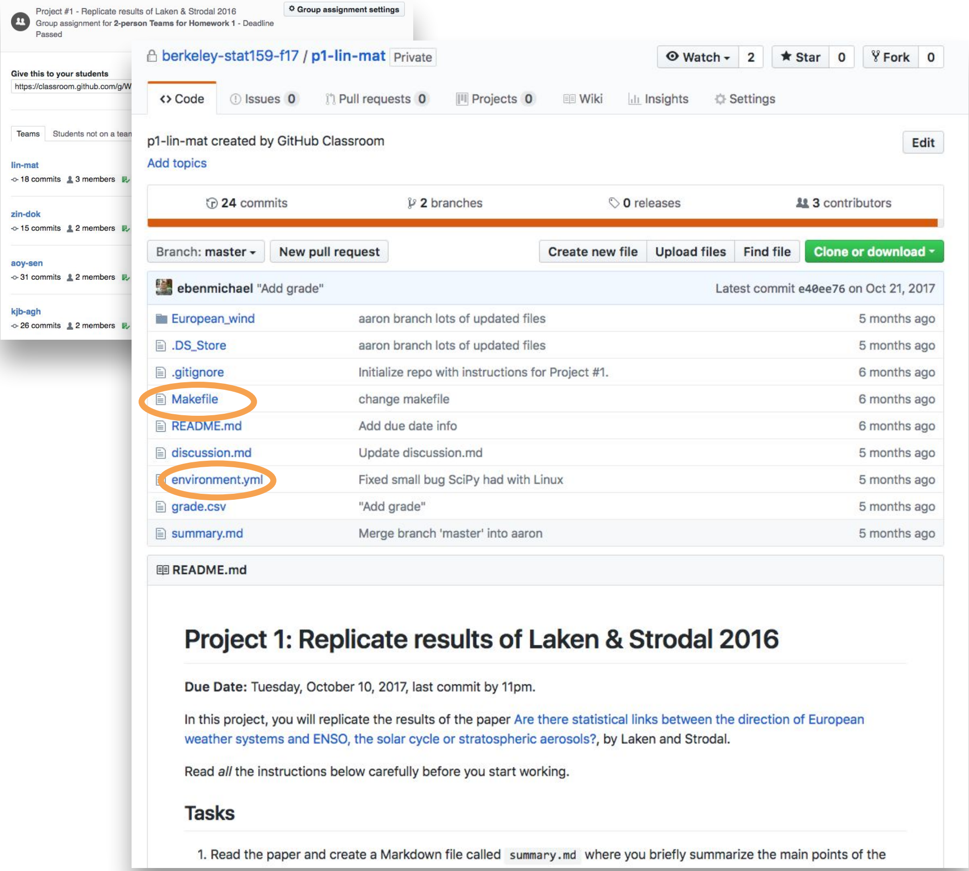
benlaken commented 11 days ago

Thanks **@fperez** for the kind words - means a lot coming from you. I am also very happy to hear that my work has been useful on your course. You have some lucky students: I wish I had a similar course when I was studying!

Please do send a PR with the fixes and I will merge to Master - if it is useful for you, I can also leave a branch in its current state?

And indeed, rock-on Open Science! 🎸 💥

Standard workflow: Makefile and environment.yml



Project #1 - Replicate results of Laken & Strodal 2016
Group assignment for **2-person Teams for Homework 1** - Deadline Passed

https://classroom.github.com/g/W

lin-mat
18 commits 3 members

zin-dok
15 commits 2 members

aoy-sen
31 commits 2 members

kjb-agh
26 commits 2 members

p1-lin-mat created by GitHub Classroom

Add topics

24 commits · 2 branches · 0 releases · 3 contributors

ebenmichael "Add grade" — Latest commit e40ee76 on Oct 21, 2017

European_wind	aaron branch lots of updated files	5 months ago
.DS_Store	aaron branch lots of updated files	5 months ago
.gitignore	Initialize repo with instructions for Project #1.	6 months ago
Makefile	change makefile	6 months ago
README.md	Add due date info	6 months ago
discussion.md	Update discussion.md	5 months ago
environment.yml	Fixed small bug SciPy had with Linux	5 months ago
grade.csv	"Add grade"	5 months ago
summary.md	Merge branch 'master' into aaron	5 months ago

Project 1: Replicate results of Laken & Strodal 2016

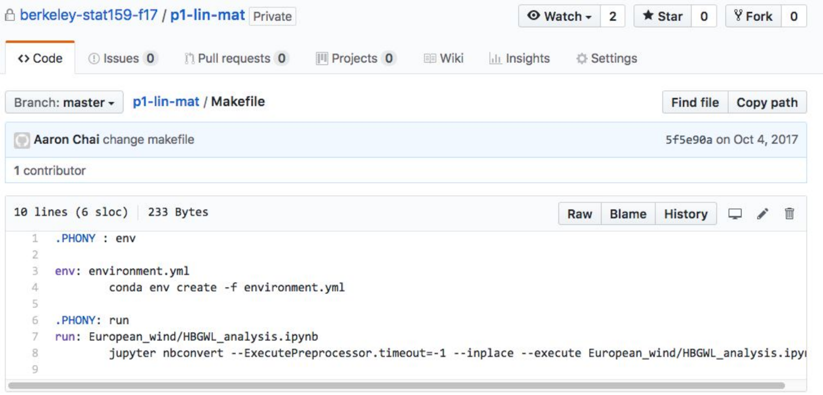
Due Date: Tuesday, October 10, 2017, last commit by 11pm.

In this project, you will replicate the results of the paper Are there statistical links between the direction of European weather systems and ENSO, the solar cycle or stratospheric aerosols?, by Laken and Strodal.

Read *all* the instructions below carefully before you start working.

Tasks

1. Read the paper and create a Markdown file called `summary.md` where you briefly summarize the main points of the



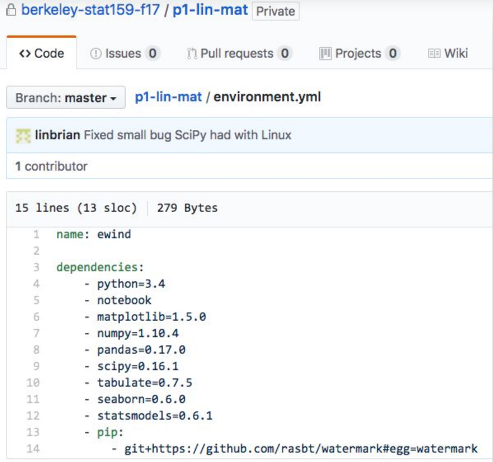
p1-lin-mat / Makefile

Aaron Chai change makefile — 5f5e90a on Oct 4, 2017

1 contributor

10 lines (6 sloc) 233 Bytes

```
.PHONY : env

env: environment.yml
	conda env create -f environment.yml

.PHONY: run
run: European_wind/HBGWL_analysis.ipynb
	jupyter nbconvert --ExecutePreprocessor.timeout=-1 --inplace --execute European_wind/HBGWL_analysis.ipy
```



p1-lin-mat / environment.yml

linbrian Fixed small bug SciPy had with Linux

1 contributor

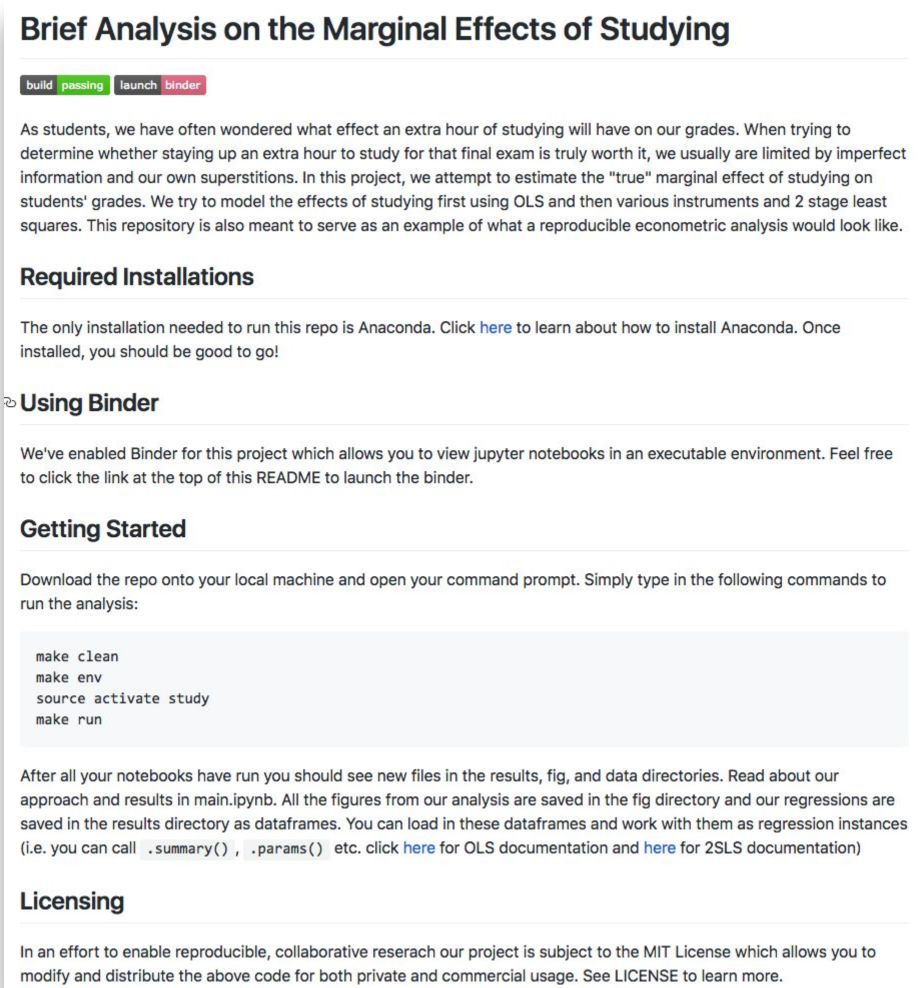
15 lines (13 sloc) 279 Bytes

```
name: ewind

dependencies:
    - python=3.4
    - notebook
    - matplotlib=1.5.0
    - numpy=1.10.4
    - pandas=0.17.0
    - scipy=0.16.1
    - tabulate=0.7.5
    - seaborn=0.6.0
    - statsmodels=0.6.1
    - pip:
      - git+https://github.com/rasbt/watermark#egg=watermark
```

"Atomic unit"of communicable results

- **Data:** included in repo or linked if too large.
- **Clean, tested code.**
- **Analysis notebooks and supporting code**
  - Break down your analysis into as many notebooks as is reasonable for convenient reading and execution.
- **Main narrative notebook:** summarizes and discusses results.
- **Reproducibility support: Makefile** and **environment.yml**
- **Good repository practices: README.md, LICENSE, .gitignore.**
  - Use Victoria Stodden's ENABLING REPRODUCIBLE RESEARCH: LICENSING SCIENTIFIC INNOVATION.

A "Standard Playbook"



Brief Analysis on the Marginal Effects of Studying

build passing launch binder

As students, we have often wondered what effect an extra hour of studying will have on our grades. When trying to determine whether staying up an extra hour to study for that final exam is truly worth it, we usually are limited by imperfect information and our own superstitions. In this project, we attempt to estimate the "true" marginal effect of studying on students' grades. We try to model the effects of studying first using OLS and then various instruments and 2 stage least squares. This repository is also meant to serve as an example of what a reproducible econometric analysis would look like.

Required Installations

The only installation needed to run this repo is Anaconda. Click here to learn about how to install Anaconda. Once installed, you should be good to go!

Using Binder

We've enabled Binder for this project which allows you to view jupyter notebooks in an executable environment. Feel free to click the link at the top of this README to launch the binder.

Getting Started

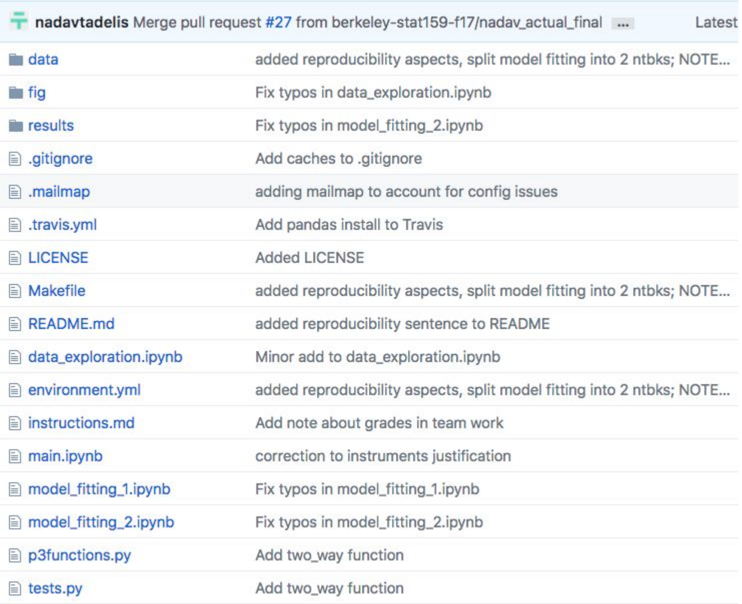
Download the repo onto your local machine and open your command prompt. Simply type in the following commands to run the analysis:

```
make clean
make env
source activate study
make run
```

After all your notebooks have run you should see new files in the results, fig, and data directories. Read about our approach and results in main.ipynb. All the figures from our analysis are saved in the fig directory and our regressions are saved in the results directory as dataframes. You can load in these dataframes and work with them as regression instances (i.e. you can call `.summary()` , `.params()` etc. click here for OLS documentation and here for 2SLS documentation)

Licensing

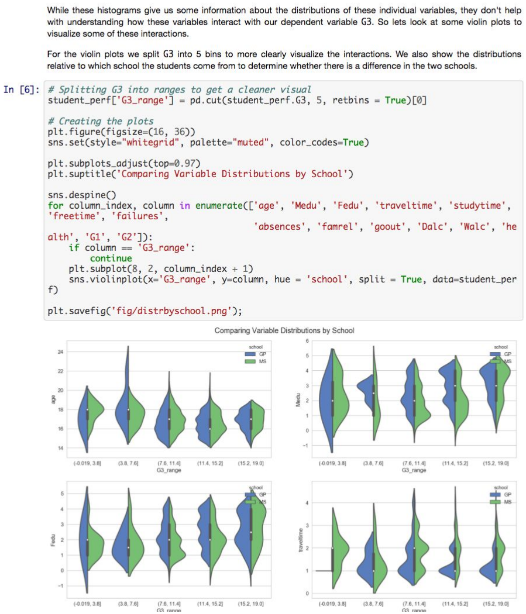
In an effort to enable reproducible, collaborative reserach our project is subject to the MIT License which allows you to modify and distribute the above code for both private and commercial usage. See LICENSE to learn more.



nadavtadelis Merge pull request #27 from berkeley-stat159-f17/nadav_actual_final ... Latest

data	added reproducibility aspects, split model fitting into 2 ntbks; NOTE...
fig	Fix typos in data_exploration.ipynb
results	Fix typos in model_fitting_2.ipynb
.gitignore	Add caches to .gitignore
.mailmap	adding mailmap to account for config issues
.travis.yml	Add pandas install to Travis
LICENSE	Added LICENSE
Makefile	added reproducibility aspects, split model fitting into 2 ntbks; NOTE...
README.md	added reproducibility sentence to README
data_exploration.ipynb	Minor add to data_exploration.ipynb
environment.yml	added reproducibility aspects, split model fitting into 2 ntbks; NOTE...
instructions.md	Add note about grades in team work
main.ipynb	correction to instruments justification
model_fitting_1.ipynb	Fix typos in model_fitting_1.ipynb
model_fitting_2.ipynb	Fix typos in model_fitting_2.ipynb
p3functions.py	Add two_way function
tests.py	Add two_way function

Analysis notebooks



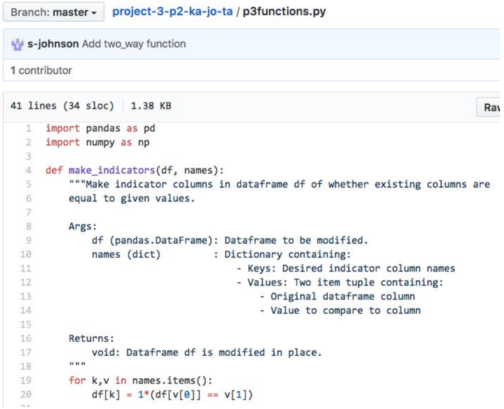
While these histograms give us some information about the distributions of these individual variables, they don't help with understanding how these variables interact with our dependent variable G3. So lets look at some violin plots to visualize some of these interactions.

For the violin plots we split G3 into 5 bins to more clearly visualize the interactions. We also show the distributions relative to which school the students come from to determine whether there is a difference in the two schools.

```
In [6]: # Splitting G3 into ranges to get a cleaner visual
        student_perf['G3_range'] = pd.cut(student_perf.G3, 5, retbins = True)[0]

        # Creating the plots
        plt.figure(figsize=(16, 36))
        sns.set(style="whitegrid", palette="muted", color_codes=True)

        plt.subplots_adjust(top=0.97)
        plt.suptitle('Comparing Variable Distributions by School')

        sns.despine()
        for column_index, column in enumerate(['age', 'Medu', 'Fedu', 'traveltime', 'studytime',
        'freetime', 'failures',
                                               'absences', 'famrel', 'goout', 'Dalc', 'Walc', 'he
        alth', 'G1', 'G2']):
            if column == 'G3_range':
                continue
            plt.subplot(8, 2, column_index + 1)
            sns.violinplot(x='G3_range', y=column, hue = 'school', split = True, data=student_per
        f)

        plt.savefig('fig/distrbyschool.png');
```

Comparing Variable Distributions by School

Code and tests



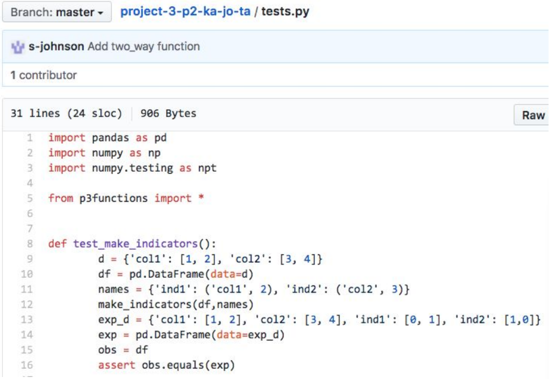
project-3-p2-ka-jo-ta / p3functions.py

s-johnson Add two_way function

1 contributor

41 lines (34 sloc) | 1.38 KB

```python
import pandas as pd
import numpy as np

def make_indicators(df, names):
    """Make indicator columns in dataframe df of whether existing columns are
    equal to given values.

    Args:
        df (pandas.DataFrame): Dataframe to be modified.
        names (dict)        : Dictionary containing:
                                - Keys: Desired indicator column names
                                - Values: Two item tuple containing:
                                    - Original dataframe column
                                    - Value to compare to column

    Returns:
        void: Dataframe df is modified in place.
    """
    for k,v in names.items():
        df[k] = 1*(df[v[0]] == v[1])
```



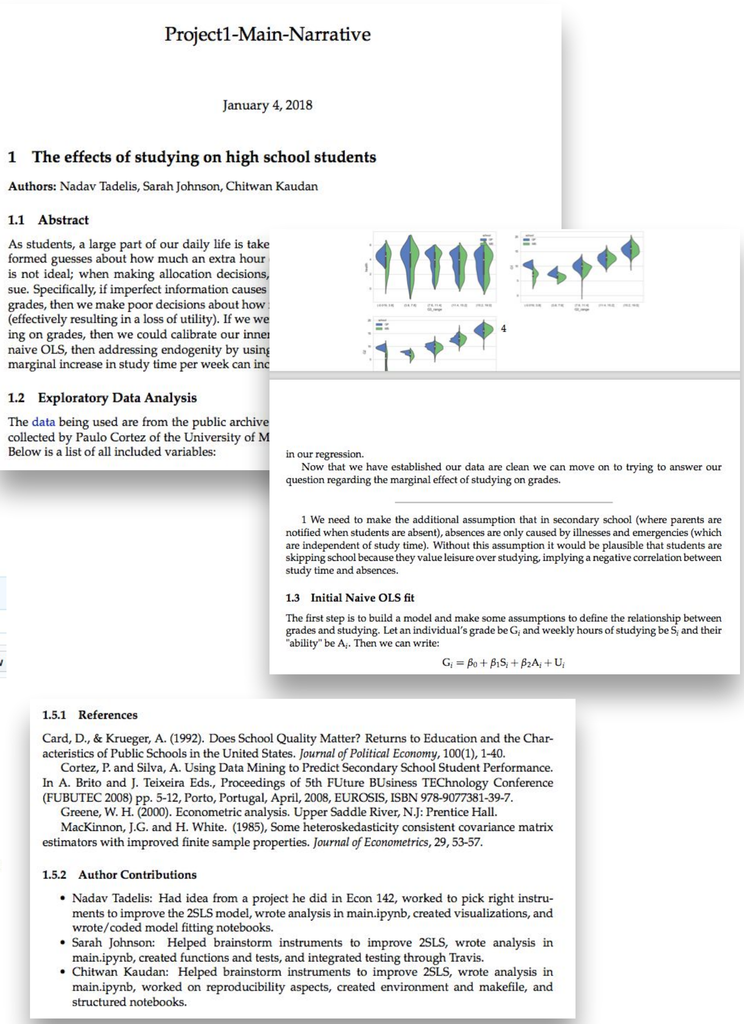
project-3-p2-ka-jo-ta / tests.py

s-johnson Add two_way function

1 contributor

31 lines (24 sloc) | 906 Bytes

```python
import pandas as pd
import numpy as np
import numpy.testing as npt

from p3functions import *


def test_make_indicators():
        d = {'col1': [1, 2], 'col2': [3, 4]}
        df = pd.DataFrame(data=d)
        names = {'ind1': ('col1', 2), 'ind2': ('col2', 3)}
        make_indicators(df,names)
        exp_d = {'col1': [1, 2], 'col2': [3, 4], 'ind1': [0, 1], 'ind2': [1,0]}
        exp = pd.DataFrame(data=exp_d)
        obs = df
        assert obs.equals(exp)
```



Project1-Main-Narrative

January 4, 2018

1 The effects of studying on high school students

Authors: Nadav Tadelis, Sarah Johnson, Chitwan Kaudan

1.1 Abstract

As students, a large part of our daily life is take
formed guesses about how much an extra hour
is not ideal; when making allocation decisions,
sue. Specifically, if imperfect information causes
grades, then we make poor decisions about how
(effectively resulting in a loss of utility). If we we
ing on grades, then we could calibrate our inner
naive OLS, then addressing endogenity by using
marginal increase in study time per week can inc

1.2 Exploratory Data Analysis

The data being used are from the public archive
collected by Paulo Cortez of the University of M
Below is a list of all included variables:

4

in our regression.

Now that we have established our data are clean we can move on to trying to answer our question regarding the marginal effect of studying on grades.

1 We need to make the additional assumption that in secondary school (where parents are notified when students are absent), absences are only caused by illnesses and emergencies (which are independent of study time). Without this assumption it would be plausible that students are skipping school because they value leisure over studying, implying a negative correlation between study time and absences.

1.3 Initial Naive OLS fit

The first step is to build a model and make some assumptions to define the relationship between grades and studying. Let an individual's grade be G_i and weekly hours of studying be S_i and their "ability" be A_i. Then we can write:

$$G_i = \beta_0 + \beta_1 S_i + \beta_2 A_i + U_i$$

1.5.1 References

Card, D., & Krueger, A. (1992). Does School Quality Matter? Returns to Education and the Characteristics of Public Schools in the United States. *Journal of Political Economy*, 100(1), 1-40.

Cortez, P. and Silva, A. Using Data Mining to Predict Secondary School Student Performance. In A. Brito and J. Teixeira Eds., Proceedings of 5th FUture BUsiness TEChnology Conference (FUBUTEC 2008) pp. 5-12, Porto, Portugal, April, 2008, EUROSIS, ISBN 978-9077381-39-7.

Greene, W. H. (2000). Econometric analysis. Upper Saddle River, N.J: Prentice Hall.

MacKinnon, J.G. and H. White. (1985), Some heteroskedasticity consistent covariance matrix estimators with improved finite sample properties. *Journal of Econometrics*, 29, 53-57.

1.5.2 Author Contributions

- Nadav Tadelis: Had idea from a project he did in Econ 142, worked to pick right instruments to improve the 2SLS model, wrote analysis in main.ipynb, created visualizations, and wrote/coded model fitting notebooks.
- Sarah Johnson: Helped brainstorm instruments to improve 2SLS, wrote analysis in main.ipynb, created functions and tests, and integrated testing through Travis.
- Chitwan Kaudan: Helped brainstorm instruments to improve 2SLS, wrote analysis in main.ipynb, worked on reproducibility aspects, created environment and makefile, and structured notebooks.

In closing

The ideas of data analysis ought to survive a look at how data is analyzed.

-- "The future of data analysis", 1961
John Tukey (1915-2000)