Executable Notebooks
(whats, whys, & hows)

Shawn Rhoads

https://shawnrhoads.github.io

https://twitter.com/ShawnRhoads56

https://github.com/shawnrhoads

What

Jupyter Notebooks

What

Jupyter Notebooks are open source web applications

What

Jupyter Notebooks are open source web applications that you can use to create and share documents

What

Jupyter Notebooks are open source web applications that you can use to create and share documents containing live code, equations, visualizations, and text

What

Jupyter Notebooks are open source web applications that you can use to create and share documents containing live code, equations, visualizations, and text

What

Ju**pyt**er Notebooks are open source web applications that you can use to create and share documents containing live code, equations, visualizations, and text

What

Jupyte**r** Notebooks are open source web applications that you can use to create and share documents containing live code, equations, visualizations, and text

Why

Boost your research productivity

Why

Boost your research productivity while simultaneously helping make science more <u>open</u>, <u>accessible</u>, and <u>reproducible</u>!

Why

More **than 2.5 million public Jupyter notebooks in September 2018 on GitHub**, up from 200,000 or so in 2015.

nature

Explore content ⌄   About the journal ⌄   Publish with us ⌄   Subscribe

nature > toolbox > article

TOOLBOX | 30 October 2018

Why Jupyter is data scientists' computational notebook of choice

An improved architecture and enthusiastic user base are driving uptake of the open-source web tool.

Jeffrey M. Perkel

How

Running notebooks

- [Anaconda](#) + Web Browser
- Base [Python](#) + Web Browser
- [Anaconda](#) + VS Code
- Base [Python](#) + VS Code
- [Google Colaboratory](#)
- [Binder](#)
- [Jupyter Lab](#)

How

Running notebooks

- Anaconda + Web Browser
- Base Python + Web Browser
- Anaconda + VS Code
- Base Python + VS Code
- Google Colaboratory
- Binder
- Jupyter Lab

Using notebooks

- Data processing, modeling/analysis, visualization

How

Running notebooks

- Anaconda + Web Browser
- Base Python + Web Browser
- Anaconda + VS Code
- Base Python + VS Code
- Google Colaboratory
- Binder
- Jupyter Lab

Using notebooks

- Data processing, modeling/analysis, visualization
- Sharing code + collaborating with others

How

Running notebooks

- Anaconda + Web Browser
- Base Python + Web Browser
- Anaconda + VS Code
- Base Python + VS Code
- Google Colaboratory
- Binder
- Jupyter Lab

Using notebooks

- Data processing, modeling/analysis, visualization
- Sharing code + collaborating with others
- **Learning new skills**

How

Running notebooks

- Anaconda + Web Browser
- Base Python + Web Browser
- Anaconda + VS Code
- Base Python + VS Code
- Google Colaboratory
- Binder
- Jupyter Lab

Using notebooks

- Data processing, modeling/analysis, visualization
- Sharing code + collaborating with others
- Learning new skills
- **Teaching others**

How

Running notebooks

- Anaconda + Web Browser
- Base Python + Web Browser
- Anaconda + VS Code
- Base Python + VS Code
- Google Colaboratory
- Binder
- Jupyter Lab

Using notebooks

- Data processing, modeling/analysis, visualization
- Sharing code + collaborating with others
- Learning new skills
- Teaching others
- **Publishing work and/or supplemental materials**

Use cases

Data processing, analysis, visualization

Markdown

fMRIPrep

All preprocessing was implemented using **fMRIPrep** (Esteban et al., 2018), which includes anatomical T1-weighted brain extraction, anatomical surface extraction, head-motion estimation and correction, susceptibility-derived distortion estimation and unwarping, intrasubject registration, and spatial normalization (intersubject registration). All volumes were registered to the MNI (Montreal Neurological Institute) standard template. Automatic removal of motion artifacts using independent component analysis (ICA-AROMA) was performed on the preprocessed BOLD time series in MNI space and the resulting data was smoothed using a 6.0 mm^3 FWHM Gaussian kernel and the time series was scaled within voxels to represent percent signal change.

We used the following `bash` command with `singularity` to run `fmriprep`:

```
SUBID=1234
singularity run --bind /mnt:/mnt --cleanenv /mnt/data/singularity/fmriprep-20.2.3.simg /mnt/
data/study/subjects /mnt/data/study/subjects/derivatives participant \
--work-dir /mnt/data/study/fmriprep-20.2.3_work \
--participant_label $SUBID \
--bold2t1w-dof 9 \
--output-spaces T1w:res-native MNI152NLin2009cAsym:res-native MNI152NLin6Asym:res-2 fsaverag
e:res-native \
--skull-strip-t1w force \
--ignore slicetiming \
--use-aroma \
--dummy-scans 0 \
--fs-license-file /mnt/data/freesurfer/license.txt \
--no-submm-recon \
--write-graph \
--fd-spike-threshold 0.5 \
--random-seed 2021
```

For group-level statistical inference, we ran a sign permutation test using the following `bash` command with `AFNI` to run `3dttest++` with `-Clustsim` :

```
dirA=mnt/data/study/firstlevel_glm/
3dttest++ -prefix /mnt/data/study/results/Condition_A.nii.gz \
          -mask /mnt/data/study/code/modeling/MNI152-graymatter-thr50-2mm.nii.gz \
          -Clustsim \
          -setA Condition_A \
              101 "$dirA/sub-101_stats.nii.gz[0]" \
              102 "$dirA/sub-102_stats.nii.gz[0]" \
              103 "$dirA/sub-103_stats.nii.gz[0]" \
              104 "$dirA/sub-104_stats.nii.gz[0]" \
              105 "$dirA/sub-105_stats.nii.gz[0]" \
              106 "$dirA/sub-106_stats.nii.gz[0]" \
              107 "$dirA/sub-107_stats.nii.gz[0]" \
```

For group-level statistical inference, we ran a sign permutation test using the following `bash` command with `AFNI` to run `3dttest++` with `-Clustsim`:

```bash
dirA=mnt/data/study/firstlevel_glm/
3dttest++ -prefix /mnt/data/study/results/Condition_A.nii.gz \
        -mask /mnt/data/study/code/modeling/MNI152-graymatter-thr50-2mm.nii.gz \
        -Clustsim \
        -setA Condition_A \
            101 "$dirA/sub-101_stats.nii.gz[0]" \
            102 "$dirA/sub-102_stats.nii.gz[0]" \
            103 "$dirA/sub-103_stats.nii.gz[0]" \
            104 "$dirA/sub-104_stats.nii.gz[0]" \
            105 "$dirA/sub-105_stats.nii.gz[0]" \
            106 "$dirA/sub-106_stats.nii.gz[0]" \
            107 "$dirA/sub-107_stats.nii.gz[0]" \
```

```python
In [14]: print("Begin Searchlight\n")
         sl_result = sl.run_searchlight(calc_svm, pool_size=pool_size)
         print("End Searchlight\n")

         end_time = time.time()

         # Print outputs
         print("Summarize searchlight results")
         print("Number of searchlights run: " + str(len(sl_result[mask==1])))
         print("Accuracy for each kernel function: " +str(sl_result[mask==1].astype('double')))
         print('Total searchlight duration (including start up time): %.2f' % (end_time - begin_time))

         # Save the results to a .nii file
         output_name = os.path.join(output_dir, ('subj%s_SL_result.nii.gz' % (sub_id)))
         sl_result = sl_result.astype('double')  # Convert the output into a precision
         format that can be used by other applications
         sl_result[np.isnan(sl_result)] = 0  # Exchange nans with zero to ensure compatibility with other applications
         sl_nii = nib.Nifti1Image(sl_result, affine_mat)  # create the volume image
         hdr = sl_nii.header  # get a handle of the .nii file's header
         hdr.set_zooms((dimsize[0], dimsize[1], dimsize[2]))
         nib.save(sl_nii, output_name)  # Save the volume
```
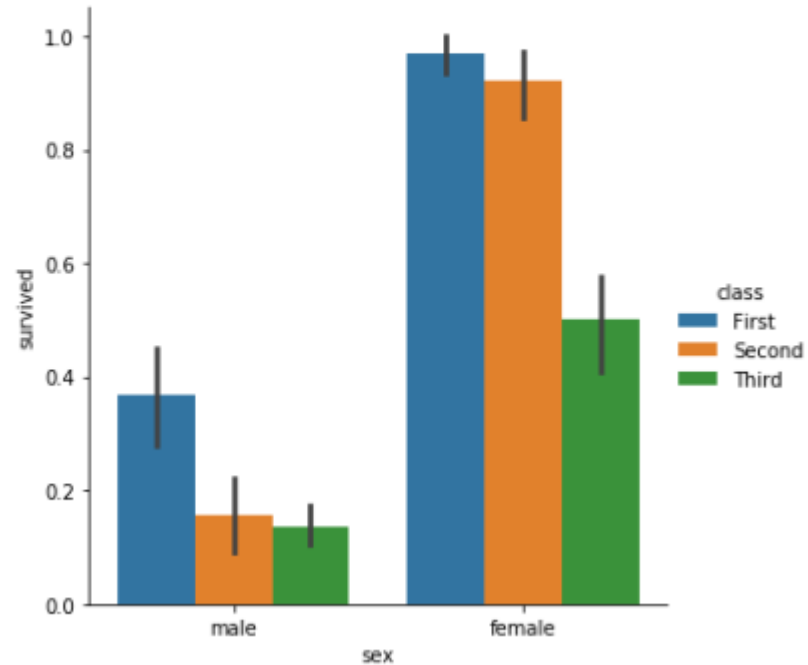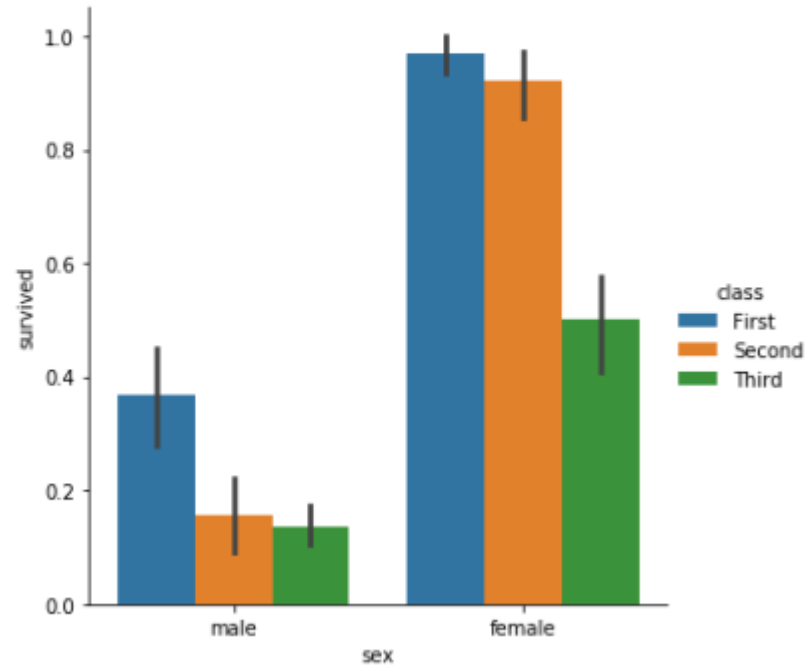
Data processing, analysis, visualization

Data processing, analysis, visualization

Supplemental Figure 1
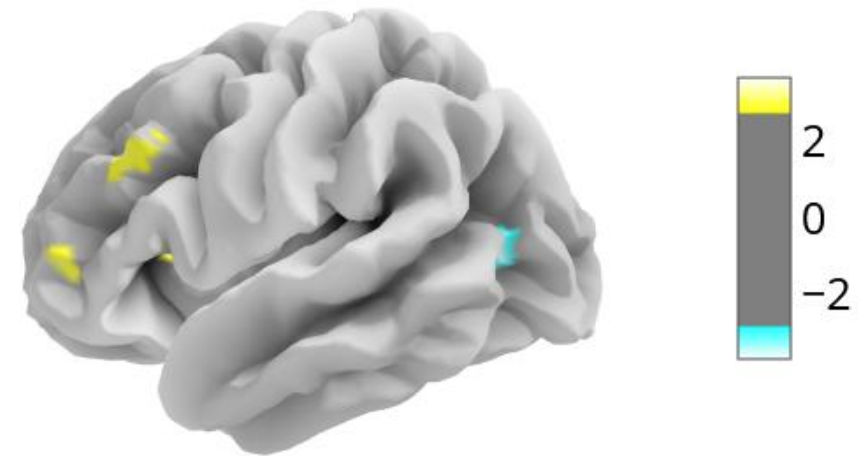
```
In [2]:  1  sns.catplot(x="sex", y="survived", hue="class",
            kind="bar", data=titanic)
         2  plt.show()
```


Supplemental Figure 3

```
In [2]:  1  from nilearn.plotting import view_img_on_surf
         2
         3  view_img_on_surf(zmap,threshold=2.8)

Out[2]:
```

Left hemisphere   Pial   view: Left

Sharing your work!

- Lab mates

Sharing your work!

- Lab mates
- Collaborators

Sharing your work!

- Lab mates
- Collaborators
- Reviewers

Sharing your work!

- Lab mates
- Collaborators
- Reviewers
- Yourself!

Learning

03 - Classification

https://brainiak.org/tutorials/

Running Classifiers

Contributions

The spam folder did not exist on email systems in the recent past. Emails that were relevant to the reader had to be manually (and painfully) sorted from an array of emails soliciting money or selling hoax products, among other things. The classification of our emails, by machines, into relevant emails and spam, has advanced to such a degree that we now take for granted that all the spam email we receive is automatically routed to the spam folder, needing little oversight from us. These machine classifiers use algorithms that are applicable to a wide variety of fields: written language; spoken language (e.g. "Hello Google", "Alexa", "Siri"); navigating driverless cars; and we'll also use them to understand brain activity.

This notebook, will walk you through the steps of extracting fMRI signal, and then training and testing classifiers on the brain.

Learning

NHA2020: Nilearn

🔍 Search this book...

An introduction to nilearn

An example classification problem

Powered by Jupyter Book

←

An introduction to nilearn

In this tutorial, we'll see how the Python library nilearn allows us to easily perform machine learning analyses with neuroimaging data, specifically MRI and fMRI.

You may notice that the name nilearn is reminiscent of scikit-learn, a popular Python library for machine learning. This is no accident! Nilearn and scikit-learn were created by the same team, and nilearn is designed to bring machine **LEARN**ing to the NeuroImaging (**NI**) domain.

With that in mind, let's briefly consider why we might want specialized tools for working with neuroimaging data. When performing a machine learning analysis, our data often look something like this:

Learning

https://dartbrains.org/content/GLM_Single_Subject_Model.html

Learning

https://executablebooks.org/en/latest/gallery.html
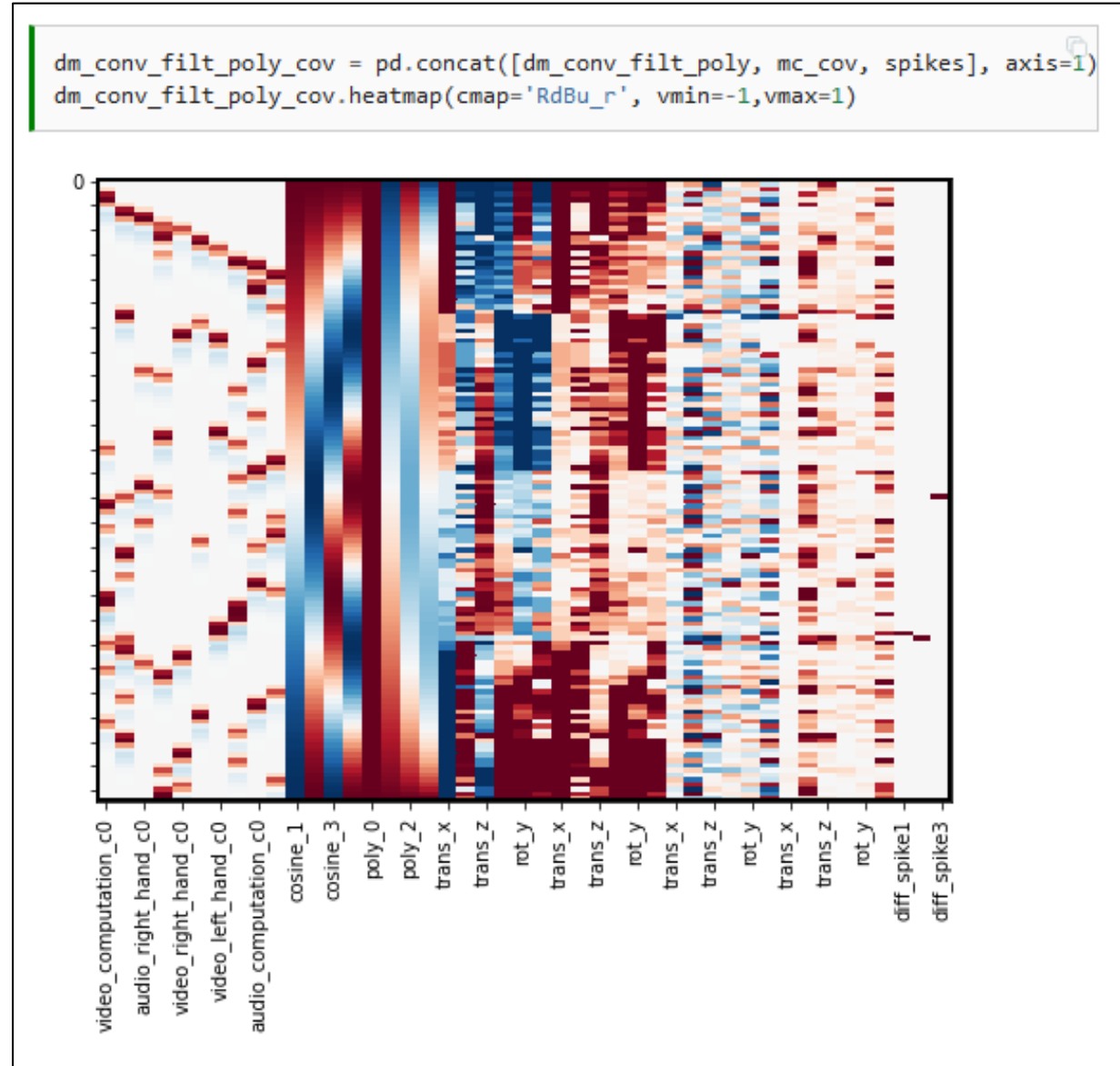
Teaching

psyc {347}
computational models of
human social behavior
and neuroscience

Instructor: Shawn A Rhoads
Georgetown University

Search this book...

MODULE 00

Syllabus

Course Schedule

Course Assignments

Reading List

Getting Started

Final Project Guidelines

Contributing

## Computational Models of Human Social Behavior and Neuroscience

Last updated: September 2021
The content in this Jupyter Book is subject to change.

Course information 📝

Section: PSYC 347-01
Prerequisites: PSYC 002 - Research Methods and Statistics (or equivalent); No prior programming experience necessary
Required materials: Only a working laptop/computer is needed! Book chapters, journal articles, and software are openly available to all students (no purchase required)
Dates: January 28 - May 6, 2021
Meetings: Tuesdays and Thursdays from 11am-12:15pm
Location: Online

https://shawnrhoads.github.io/gu-psyc-347
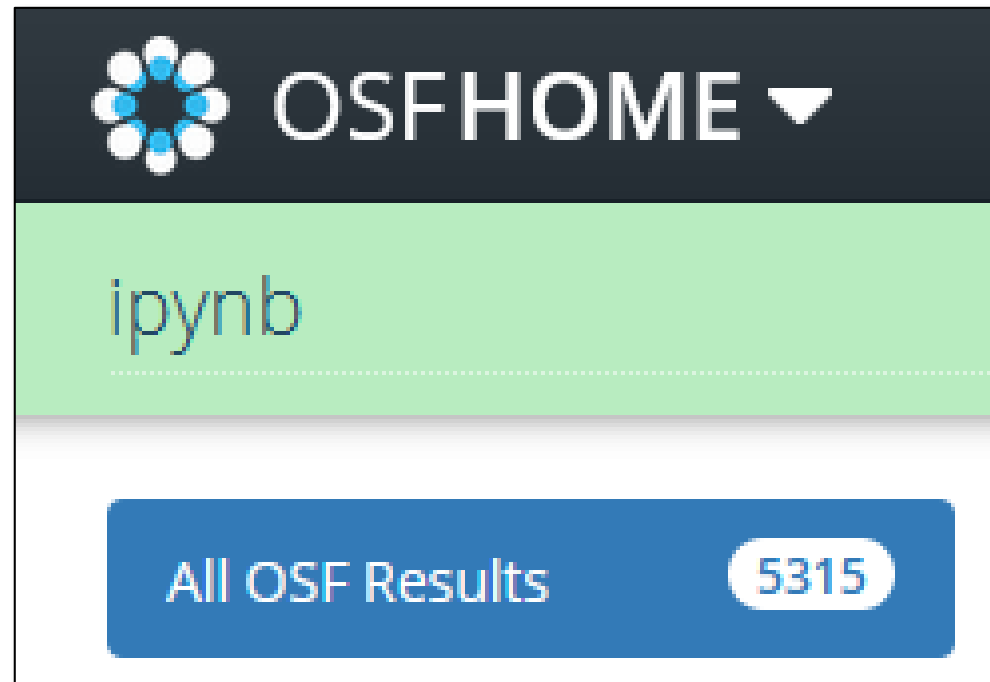
Publishing

eLife launches Executable Research Articles for publishing computationally reproducible results

Authors with a published eLife paper can now enrich their work with embedded code blocks and computed outputs to make their results more transparent, interactive and reproducible.

An interactive meta-analysis of MRI biomarkers of myelin

Matteo Mancini, ✉ Agah Karakuzu, Julien Cohen-Adad, Mara Cercignani, Thomas E Nichols, Nikola Stikov

Department of Neuroscience, Brighton and Sussex Medical School, University of Sussex, Brighton, United Kingdom; NeuroPoly Lab, Polytechnique Montreal, Montreal, Canada; CUBRIC, Cardiff University, Cardiff, United Kingdom; Functional Neuroimaging Unit, CRIUGM, Université de Montréal, Montreal, Canada; Neuroimaging Laboratory, Fondazione Santa Lucia, Rome, Italy; Wellcome Centre for Integrative Neuroimaging (WIN FMRIB), University of Oxford, Oxford, United Kingdom; Big Data Institute, University of Oxford, Oxford, United Kingdom; Montreal Heart Institute, Université de Montréal, Montreal, Canada

https://elifesciences.org/articles/61523/executable

Notebooks of the Future

Notebooks of ~~the Future~~ Now

Outline

- Starting from scratch

- Running Jupyter Notebooks locally
  - Using typical method
  - Using VSCode

- Tutorial with Jupyter (Markdown, R, Python)

- Google Colaboratory

- Using GitHub + JupyterBook to publish your work online

https://github.com/shawnrhoads/executable-book-template