May 2025
Irvine, CA

Going public with your CPS code and data

Jonathan Sprinkle (Vanderbilt University)
Cathy Wu (MIT)
Riley Wagner (Vanderbilt University)
Katie Dey (Vanderbilt University)
Stephen Rees (Vanderbilt University)
Matt Bunting (Vanderbilt University)
Junyi Ji (Vanderbilt University)

Tutorial, CPS-IoT Week 2025

Going public with your CPS code and data

Tutorial agenda

Session 1: Making your code do the talking

0900 - 0930	Introductions, the state of reproducibility
0930 - 1030	Code tutorial: make your code pretty and reusable
1030 - 1100	Coffee Break

Session 2: Creating an accessible and active resource

1100 - 1130	Demonstration of posting various types of resources with the CPS-VO
1130 - 1200	CPS open data sharing practice: I-24 MOTION testbed demonstration
1200 - 1230	CPS open data sharing practice: LADDMS demonstration

Quick survey

Make your code pretty and reusable

Going public with your CPS code and data

Junyi Ji - Vanderbilt University

CPS-IoT Week 2025

Tutorial

Prerequisites: https://qtext.io/6sf0

This tutorial is designed to be accessible. We welcome all skill levels.

You will need:

- A Github account (see tutorial)
- Basic command line knowledge (see tutorial)
  - If you can change directories, create & delete directories, and run a python script, you'll be fine.
  - Otherwise, you can take a few minutes to learn how. Here are some tutorials for Unix (Mac & Linux) and Windows.
- Software requirements
  - A command line program (E.g., Terminal or iTerm2 for MacOS, Command Prompt for Windows)
  - git (see tutorial)
  - Anaconda Python
    - 👉 For Mac users with M1/M2 chips, please still install the Intel version
    - Miniconda should work fine, if you prefer.
  - A Python IDE
    - 👉 We will be using Visual Studio Code (VSCode). Install VSCode to best follow along.
- Knowledge of git and python are helpful but not required. Part 3 of the tutorial ("Make it pretty") will benefit from proficiency in Python.

Reproducibility

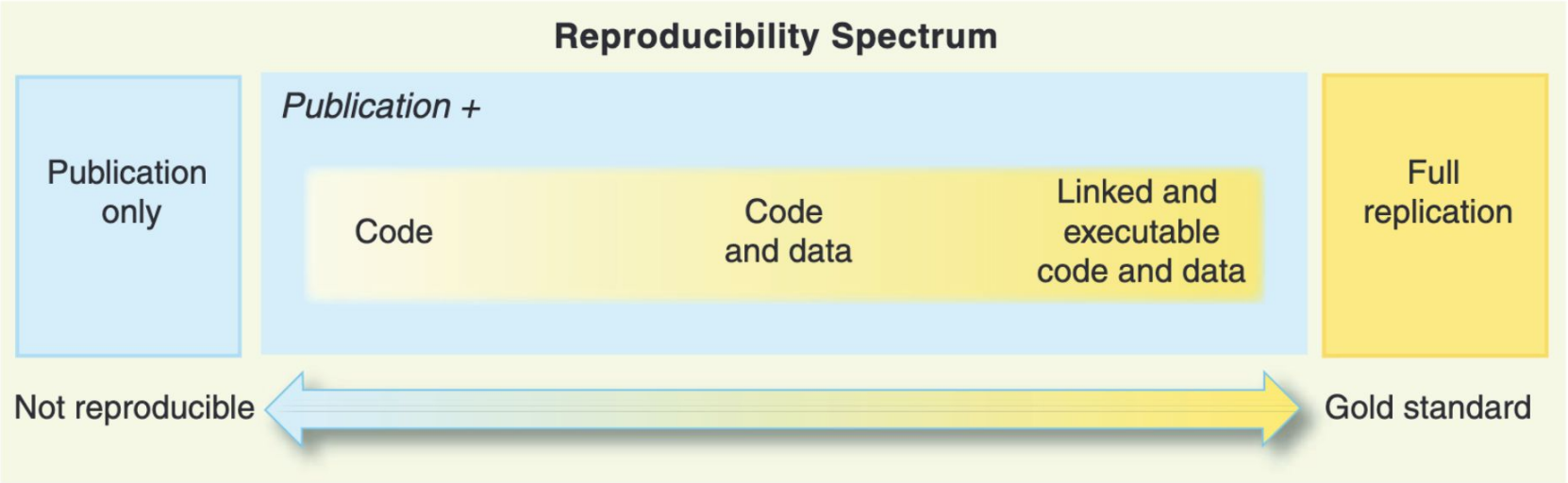
Quick survey

- Reproducibility is obtaining **consistent results** using the same input data, computational steps, methods, and code, and conditions of analysis. This definition is synonymous with “computational reproducibility.”

- Replicability is obtaining consistent results across studies aimed at **answering the same scientific question**, each of which has obtained its own data. Two studies may be considered to have replicated if they obtain consistent results given the level of uncertainty inherent in the system under study.

Stodden, Victoria, et al. "Enhancing reproducibility for computational methods." *Science* 354.6317 (2016): 1240-1241.
National Academies of Sciences, et al. *Reproducibility and replicability in science*. National Academies Press, 2019.

Reproducibility levels



Where we are?

Peng, Roger D. "Reproducible research in computational science." *Science* 334.6060 (2011): 1226-1227.

Transformation in the publication pipeline

1660's:

1. Enough detail on equipment, materials, and procedures for reproducibility
2. "Communal witnessing"
3. Exhaustive details on experimental settings, false starts, failures, etc.

1900's

Standards for journal publication: e.g. Introduction, Methods, Results, Discussion.

2025 and future

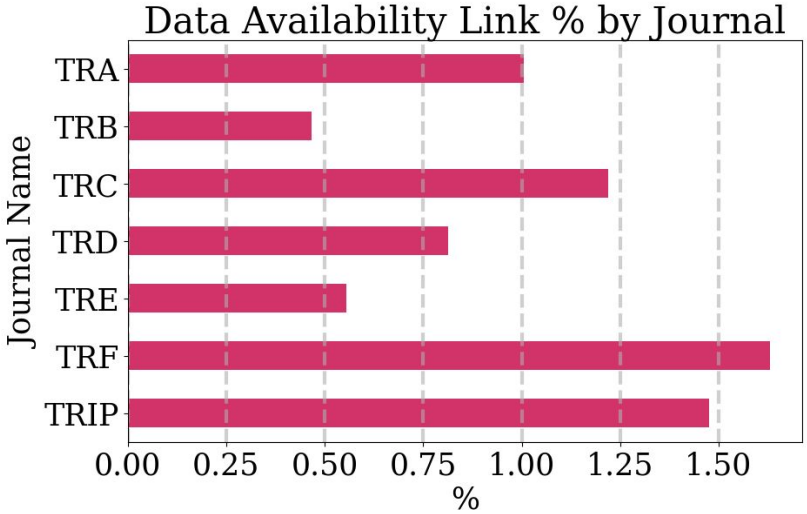
Executable workflows with links to corresponding data and results. Open accessibility.

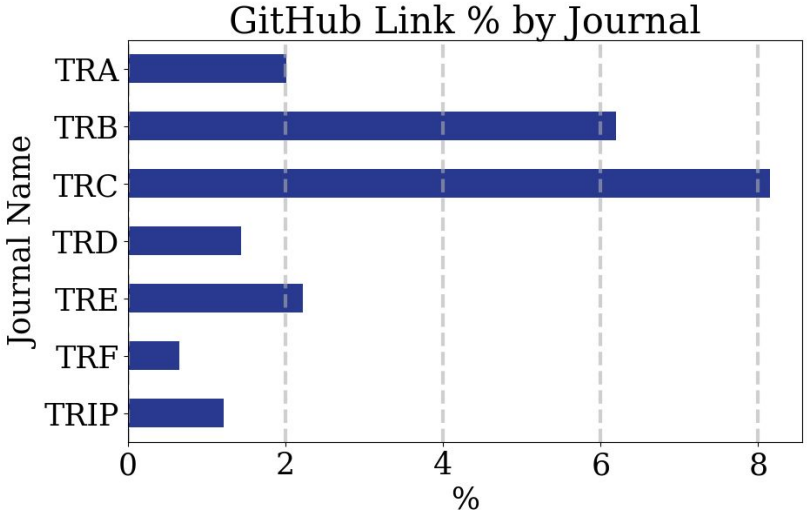
The question is: Who is the reader?
How to measure the credibility of the results?

[Victoria Stodden on AI challenges in scientific discovery, https://youtu.be/oMUPoCWh_Q4, 2025]

Reproducibility - A closer look at empirical data

- Retrieved full-text data for 10,000+ papers (2019–2024) in TR journals





Data availability*: **1.0%**

Code availability*: **3.1%**

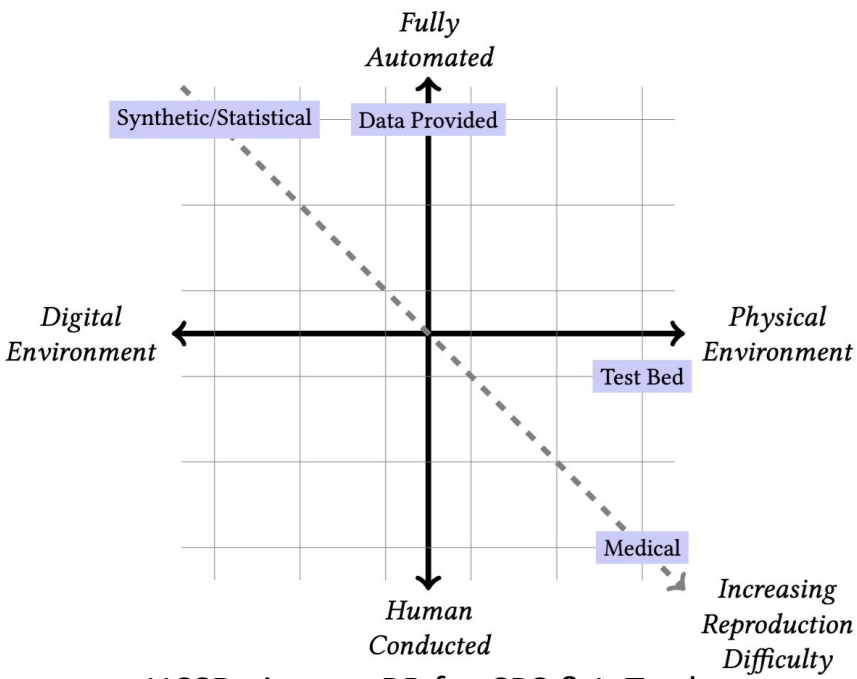
* Preliminary estimate. Could include use of open data or code from other projects, and exclude data or code mentioned elsewhere.

Is that our fault?

No!

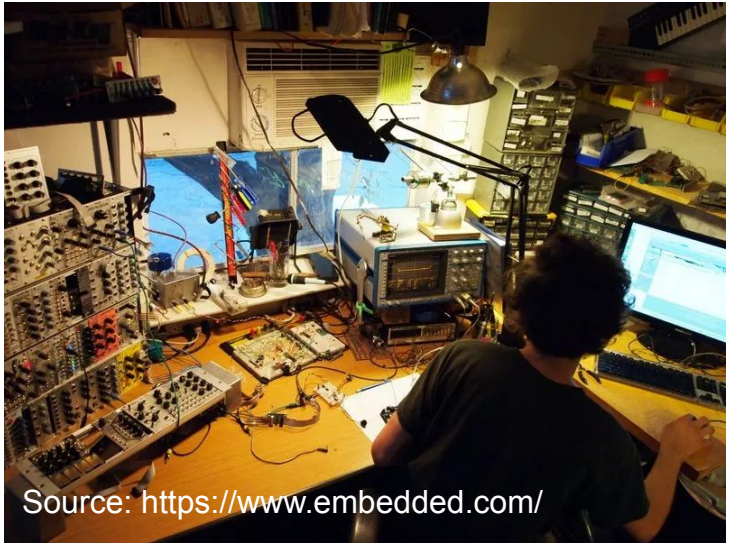
The problem lies in the complexity in the context of CPS.

Reproducibility in CPS



UCSD view on RR for CPS & IoT - the digital and physical components

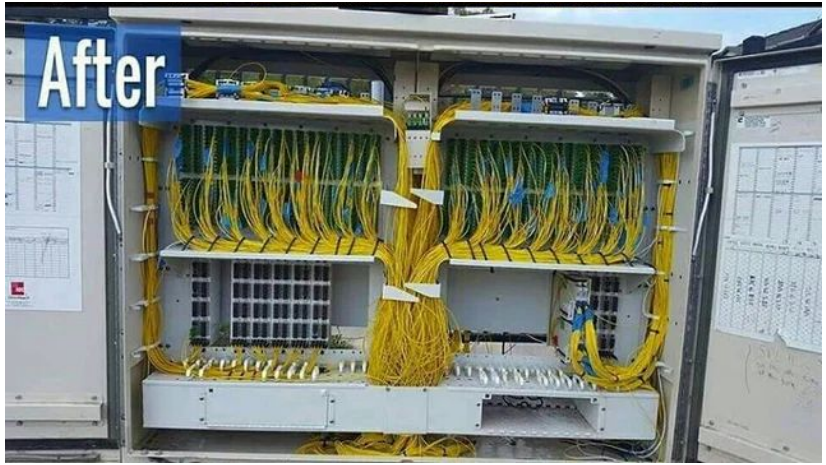
A typical hardware work deck



Instruments with software/hardware in the loop

Yen, Alex, et al. "A UCSD view on replication and reproducibility for CPS & IoT." *Proceedings of the Workshop on Benchmarking Cyber-Physical Systems and Internet of Things*. 2021

My code before after this tutorial is like…



A small step further will bring in huge changes!

Philosophy: ***Get your feet wet with reproducibility***

From start to finish, create a **reproducible project on Github**

To **empower you** to make **your next research paper** reproducible.

Session goals & learning objective

- **Main goal**: Create a project that someone can reproduce in less than 5 minutes.
- **Stretch goal**: Organize the project and make it understandable.

- Learn principles and practical tools for making your research project reproducible. Including:
  - Fundamentals
    - Principles of organizing projects, code, & data
    - Working knowledge of Github
  - Advanced skills
    - Automatically extract the computing environment for reproducibility
    - VS Code and powerful extensions (tools for Jupyter, Markdown, csv, lint)

Prerequisites: https://qtext.io/6sf0

This tutorial is designed to be accessible. We welcome all skill levels.

You will need:

- A Github account (see tutorial)
- Basic command line knowledge (see tutorial)
  - If you can change directories, create & delete directories, and run a python script, you'll be fine.
  - Otherwise, you can take a few minutes to learn how. Here are some tutorials for Unix (Mac & Linux) and Windows.
- Software requirements
  - A command line program (E.g., Terminal or iTerm2 for MacOS, Command Prompt for Windows)
  - git (see tutorial)
  - Anaconda Python
    - 👉 For Mac users with M1/M2 chips, please still install the Intel version
    - Miniconda should work fine, if you prefer.
  - A Python IDE
    - 👉 We will be using Visual Studio Code (VSCode). Install VSCode to best follow along.
- Knowledge of git and python are helpful but not required. Part 3 of the tutorial ("Make it pretty") will benefit from proficiency in Python.

Overview: Making your research project reproducible

- Key lesson: Don’t worry about making things perfect!

- Priorities for research code:
  1. Make it run
  2. Make it available
  3. Make it pretty

- Available is better than perfect!

Outline

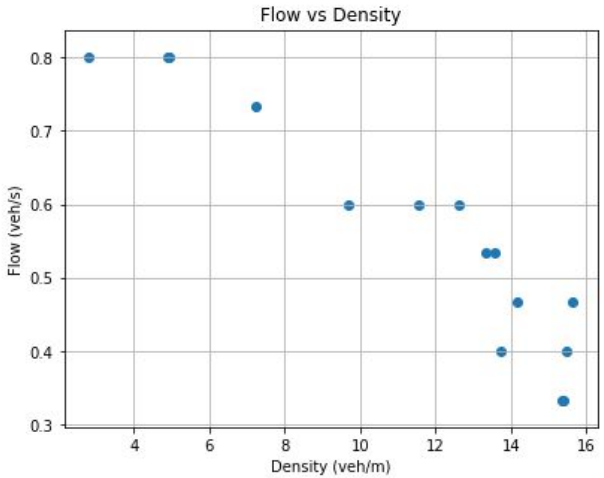
1. Make it run
2. Make it available
3. Make it pretty

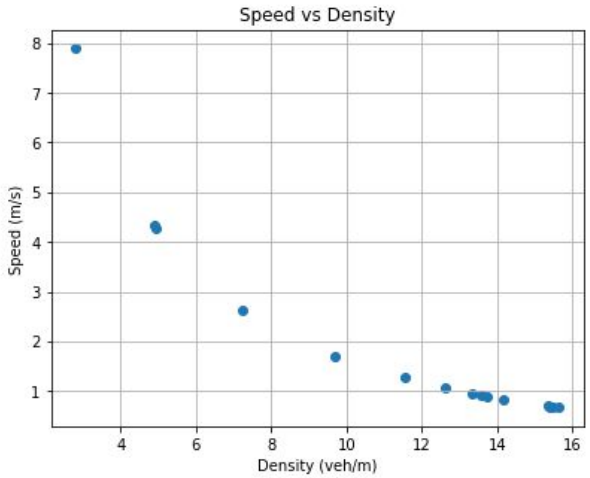
Outline

1. **Make it run**
   a. For you: an ITS example; handling randomness
   b. For others, including future you: README, installation files
2. Make it available
3. Make it pretty

Make it run

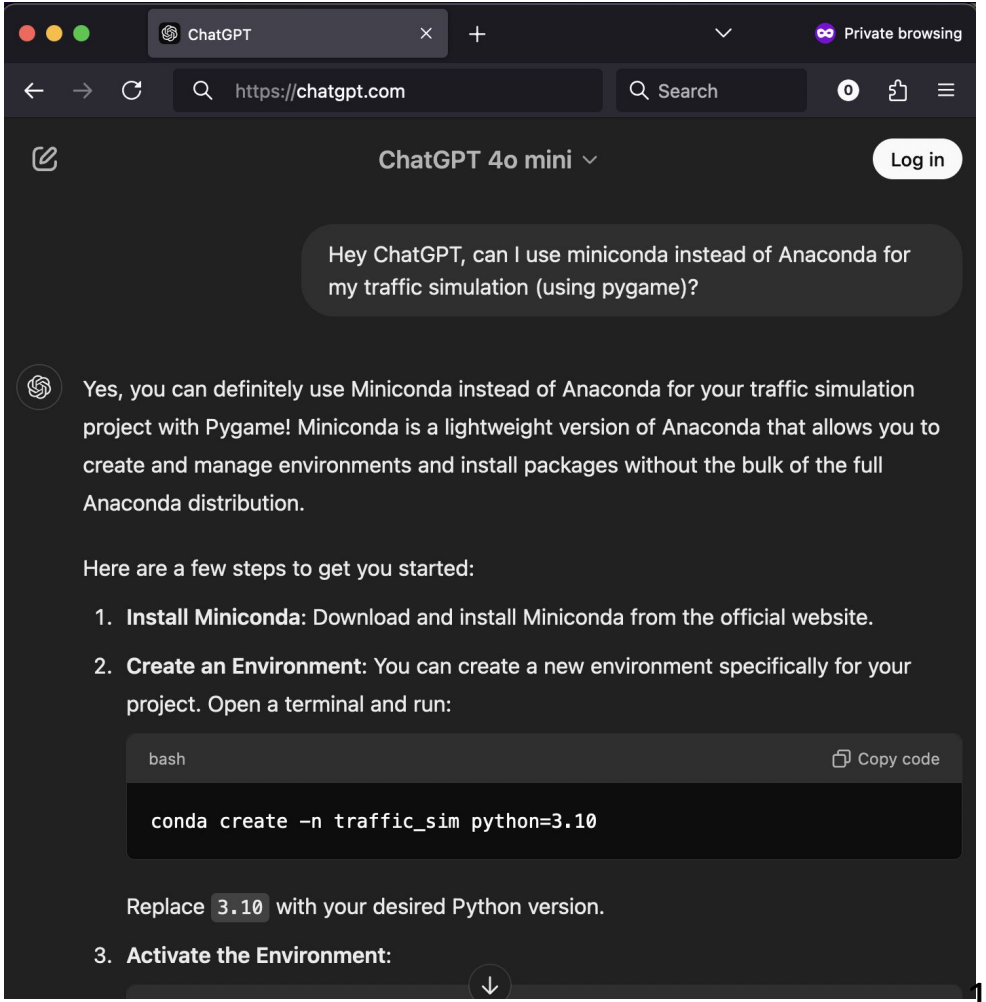
- **Exercise: Reproduce the car following simulation results**
- Download project files: https://qtext.io/6sf0
- **In 2 minutes, try to reproduce my plots. Go go go!**
  - Hint: `simulator.py` is the starting point.
- Fundamental diagrams generated from single lane traffic modeled using the Intelligent Driver Model (IDM)





By the way, if you get stuck…

- Try ChatGPT for resolving issues or clarify instructions.
- This will allow you to unblock yourself in the future in your own research project.



Fine, let me help you reproduce…

- `$ conda create -n cps25-RR-tutorial python=3.8`
- `$ conda activate cps25-RR-tutorial`

- `$ pip install matplotlib==3.1.0`
- `$ pip install pygame==2.6.0`
- `$ pip install pandas==2.0.3`
- `$ pip install numpy==1.24.4`
- `$ pip install jupyterlab==4.2.5`

- `$ python simulator.py --run-idm --no-render`

That was a lot of steps though. Easier…

- `$ conda env create -f environment.yml`
- `$ conda activate RR`

- `$ python simulator.py --run-idm --no-render`

Get the same result each time you run the code

- Handle randomness
- **Exercise: Reproduce the exact car following simulation results**
  - The project code we gave you will produce slightly different data and figures each time you run it.

- Solution: Fix random seed in Python and libraries (add to any simulator.py)
  - `import random`
  - `import numpy as np`
  - `random.seed(my_seed)`
  - `np.random.seed(my_seed)`
- Our reference example uses random seed 175175175

- Check that there are no changes to the data/figures when you re-run `simulator.py`.

Further reading: https://www.kaggle.com/code/rtatman/reproducible-research-best-practices-jupytercon#Controlling-randomness

Outline

1. **Make it run**
   a. For you: an ITS example; handling randomness
   b. **For others, including future you: README, installation files**
2. Make it available
3. Make it pretty

Make it run, next time

Your closest collaborator is you six months ago, but you don’t reply to emails.

Credit: Karen Cranston (Twitter)

Overall project README

- **Exercise: Assemble a project README.md**

Paper reference

Any setup commands

paperswithcode / releasing-research-code

Code | Issues 3 | Pull requests 2 | Actions | Projects

Files | master | releasing-research-code / templates / README.md

rstojnic Update README.md a9abf3f · 4 years ago

62 lines (35 loc) · 2.11 KB

Preview | Code | Blame | Raw

> A template README.md for code accompanying a Machine Learning paper

My Paper Title

This repository is the official implementation of My Paper Title.

> Optional: include a graphic explaining your approach/main result, bibtex entry, link to demos, blog posts and tutorials

Requirements

To install requirements:

```
pip install -r requirements.txt
```

> Describe how to set up the environment, e.g. pip/conda/docker commands, download datasets, etc...

Training

To train the model(s) in the paper, run this command:

```
python train.py --input-data <path_to_data> --alpha 10 --beta 20
```

> Describe how to train the models, with example commands on how to train the models in your paper, including the full training procedure and appropriate hyperparameters.

Evaluation

To evaluate my model on ImageNet, run:

```
python eval.py --model-file mymodel.pth --benchmark imagenet
```

> Describe how to evaluate the trained models on benchmarks reported in the paper, give commands that produce the results (section below).

Pre-trained Models

You can download pretrained models here:

- My awesome model trained on ImageNet using parameters x,y,z.

> Give a link to where/how the pretrained models can be downloaded and how they were trained (if applicable). Alternatively you can have an additional column in your results table with a link to the models.

Results

Our model achieves the following performance on :

Image Classification on ImageNet

Model name	Top 1 Accuracy	Top 5 Accuracy
My awesome model	85%	95%

> Include a table of results from your paper, and link back to the leaderboard for clarity and context. If your main result is a figure, include that figure and link to the command or notebook to reproduce it.

Contributing

> Pick a licence and describe how to contribute to your code repository.

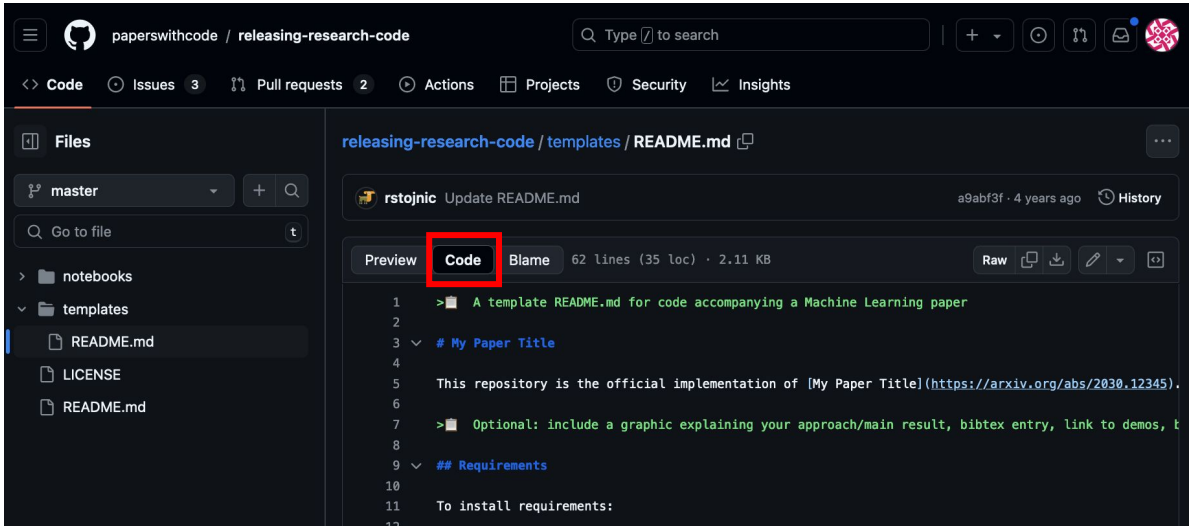
Commands to reproduce data

Commands to reproduce plots

"Tips for Publishing Research Code," GitHub, 9 Sept. 2024, github.com/paperswithcode/releasing-research-code/tree/master.

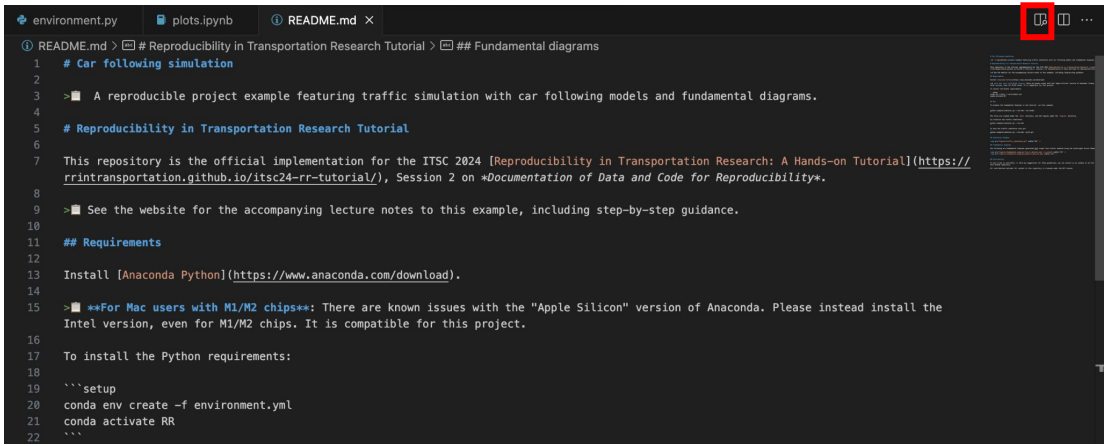
Optional tips: Copy-paste a README template

- Go to https://github.com/paperswithcode/releasing-research-code/tree/master
- Within the `templates` folder, open `README.md`
- Select the `Code` tab
- From here you can copy and paste the template into your project, and adapt it to your liking.
- Templates usually include more than what you need, so delete or adjust the sections based on what others need to reproduce your work.

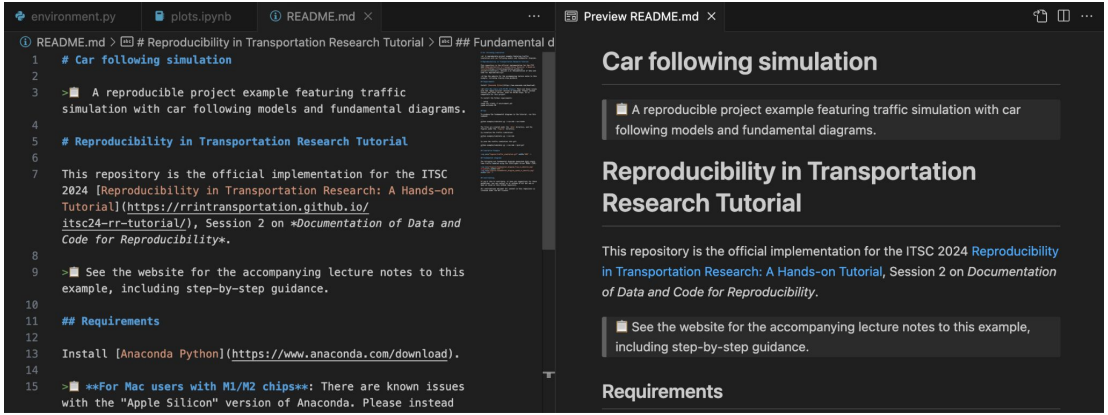


Optional tips: Editing a README.md in VS Code

Open Markdown preview



Side-by-side Markdown + preview



Project setup files

- **Exercise: Create installation files for the project**

README Apache-2.0 license

`pipreqs` - Generate requirements.txt file for any project based on imports

pypi v0.5.0 codecov 90% license Apache-2.0

- Option 1: Extract environment info
  - Often good enough for research
  - User A: Programmatically extract the environment information from the kernel (e.g., anaconda Python)
  - User B: Install environment information in local system, then run code

pypi v24.2 python 3.8 | 3.9 | 3.10 | 3.11 | 3.12 docs passing

pip is the package installer for Python. You can use pip to install packages from the Python Package Index and other indexes.

- Option 2: Docker
  - More advanced, but simpler for the user
  - User A: Create a Docker image of your project
  - User B: Run image in any environment (Windows, Linux, MacOS, etc.)

docker

IP[y]: IPython Interactive Computing

Install · Documentation · Project · Jupyter · News · Cite

IPython provides a rich architecture for interactive computing with:

- A powerful interactive shell.
- A kernel for Jupyter.
- Support for interactive data visualization and use of GUI toolkits.
- Flexible, embeddable interpreters to load into your own projects.
- Easy to use, high performance tools for parallel computing.

Extract installation information

```
l/itsc24-rr-tutorial-example-messy$ python --version
Python 3.8.0
```

Python version

Packages based on imports

setup-info > requirements-imports.txt

```
matplotlib==3.1.0
numpy==1.24.4
pandas==2.0.3
pygame==2.6.0
```

setup-info > requirements-all-dependencies.txt

```
anyio==4.4.0
appnope==0.1.4
argon2-cffi==23.1.0
argon2-cffi-bindings==21.2.0
arrow==1.3.0
asttokens==2.4.1
async-lru==2.0.4
attrs==24.2.0
babel==2.16.0
backcall==0.2.0
beautifulsoup4==4.12.3
bleach==6.1.0
certifi==2024.8.30
cffi==1.17.1
charset-normalizer==3.3.2
comm==0.2.2
cycler==0.12.1
debugpy==1.8.5
decorator==5.1.1
defusedxml==0.7.1
docopt==0.6.2
exceptiongroup==1.2.2
executing==2.1.0
fastjsonschema==2.20.0
fqd
h11
htt
htt
idn
imp
importlib_resources==6.4.5
ipykernel==6.29.5
```

All packages in conda env

Create a conda environment.yml file (1/3)

- Create a blank file `environment.yml`. This file maintains the software dependencies for easy installation by others (& future you).

environment.yml

```yaml
name: RR

dependencies:
    - pip
    - python==3.8
    - pip:
        - matplotlib==3.7.5
        - pygame==2.6.0
        - pandas==2.0.3
        - numpy==1.24.4
        - pillow==10.4.0
```

A short name for the conda environment of your project (no spaces)

Python dependencies
Format: library_name==version_number

A user will install and activate the environment by:

```
$ conda env create -f environment.yml
$ conda activate RR
```


Further reading: https://conda.io/projects/conda/en/latest/user-guide/tasks/creating-projects.html

Create a conda environment.yml file (2/3)

Commands to extract the installation information:

- Get python version
  - `$ python --version`

- Get module information
  - `$ pipreqs . --force --savepath requirements-imports.txt`
    - To install it: `$ pip install pipreqs`
  - `$ pip3 freeze > requirements-all-dependencies.txt`
- pipreqs vs pip3 freeze
  - pipreqs: Based on imports, excludes package dependencies
  - pip3 freeze: Anything installed by pip in the current conda env, including package dependencies

Create a conda environment.yml file (3/3)

- Fill in the `environment.yml` with the extracted installation information.

environment.yml

```yaml
name: RR

dependencies:
    - pip
    - python==3.8
    - pip:
        - matplotlib==3.7.5
        - pygame==2.6.0
        - pandas==2.0.3
        - numpy==1.24.4
        - pillow==10.4.0
```

Fill in from `$ python --version`. Some platforms require excluding the patch version (e.g. use 3.8 instead of 3.8.0).

Fill in from `requirements-imports.txt`. Add whatever else you'd like to. For example, add `jupyterlab`, to run more code snippets.

Further reading: https://conda.io/projects/conda/en/latest/user-guide/tasks/creating-projects.html

Outline

1. Make it run
2. **Make it available**
   a. Open-source licenses
   b. Get it on Github
3. Make it pretty

Code

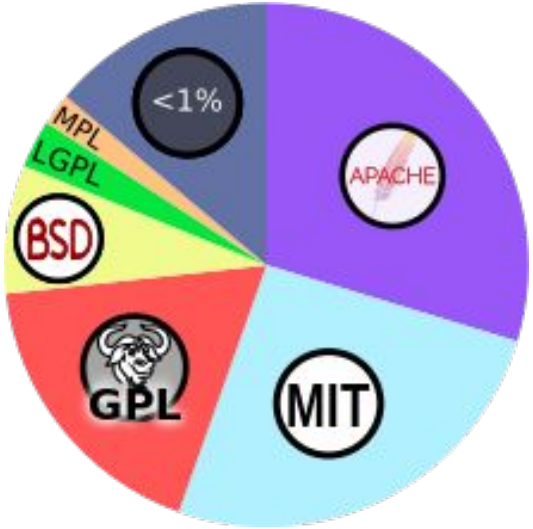
"**An article** about computational science in a scientific publication is not the scholarship itself, it **is merely advertising** of the scholarship. **The actual scholarship is the complete software development environment and the complete set of instructions which generated the figures.**"

– Buckheit and Donoho, 1995

Make it available

tl;dr: Use the MIT license

1. Choose an open-source license.
2. Put your project on Github.



Open-source license, Wikipedia (2021)

Most common for research code

	APACHE	BSD	MIT	GPL v3 Free as in Freedom	LGPL v3 Free as in Freedom	AGPL v3 Free as in Freedom
Type	Permissive	Permissive	Permissive	Copyleft	Copyleft	Copyleft
Provides copyright protection	✓ TRUE	✓ TRUE	✓ TRUE	✓ TRUE	✓ TRUE	✓ TRUE
Can be used in commercial applications	✓ TRUE	✓ TRUE	✓ TRUE	✓ TRUE	✓ TRUE	✓ TRUE
Provides an explicit patent license	✓ TRUE	✗ FALSE	✗ FALSE	✗ FALSE	✗ FALSE	✗ FALSE
Can be used in proprietary (closed source) projects	✓ TRUE	✓ TRUE	✓ TRUE	✗ FALSE	✗ FALSE partially	✗ FALSE for web
Popular open-source and free projects	Kubernetes Swift Firebase	Django React Flutter	Angular.js JQuery, .NET Core Laravel	Joomla Notepad++ MySQL	Qt SharpDevelop	SugarCRM Launchpad

Source: HamRadio.my

Further reading: BSD vs MIT,

Outline

1. Make it run
2. **Make it available**
   a. Open-source licenses
   b. **Get it on Github**
3. Make it pretty

Exercise: Get the code up on Github!

1. Set up git
2. Create a new repo on Github
3. Download the empty repo to your local machine
4. Add files to a new workspace for the repo
5. Commit & sync files to Github

If you're familiar with Github and your IDE, feel free to try without the step-by-step directions!

1. Installing Git (1/2)

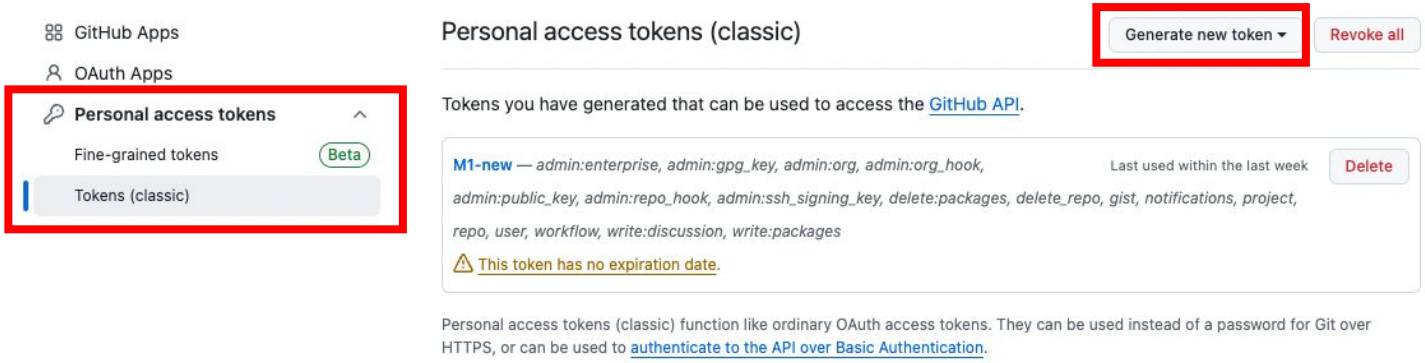
- First check if you already have Git installed
  - In terminal/ command line: `git --version`
    - If you see a git version, you are all set. If not, follow the instructions below.
- On Mac
  - Via Homebrew: `brew install git`
  - Via Xcode: Apple ships a binary package of Git with Xcode. Therefore, just download Xcode from App store and install it.
- On Linux
  - Via Apt: `apt-get install git`
- On Windows
  - Download the standalone installer and install it: https://git-scm.com/download/win

1. Setting up Git

- Open terminal /command line
  - Configure your details
    - `git config --global user.email "you@example.com"`
    - `git config --global user.name "Your Name"`

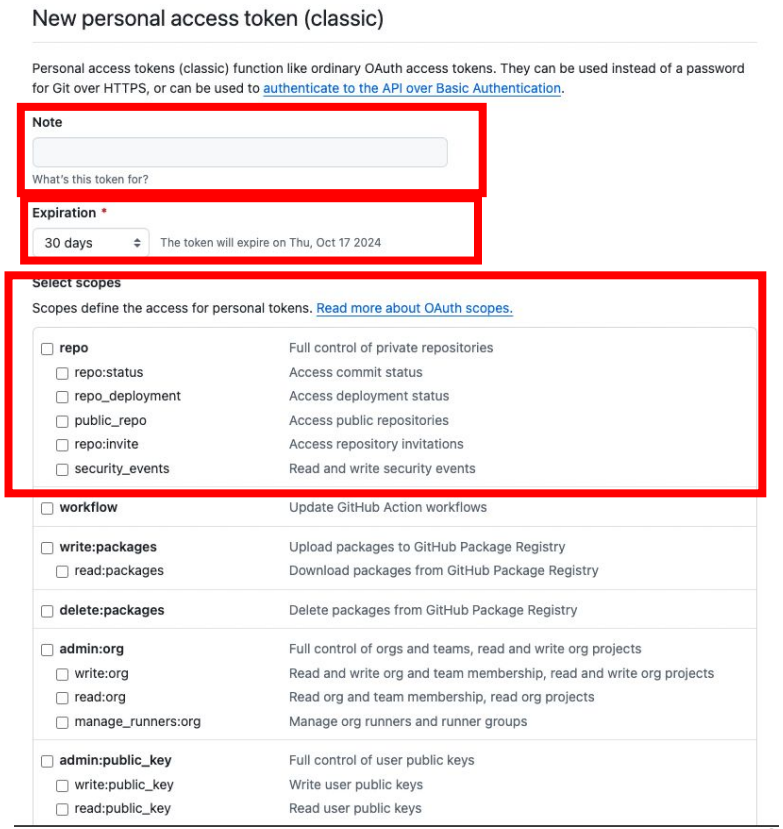
1. Setting up Git (2/6)

- Git authentication
  - To push/pull from a repository in Github, you will need to authenticate with a username and password.
    - Username: your Github username
    - Password: generate an authentication as follows
      - Settings → Developer settings → Personal access tokens → Tokens (classic)
      - Click Generate new token and select generate new token (classic option)



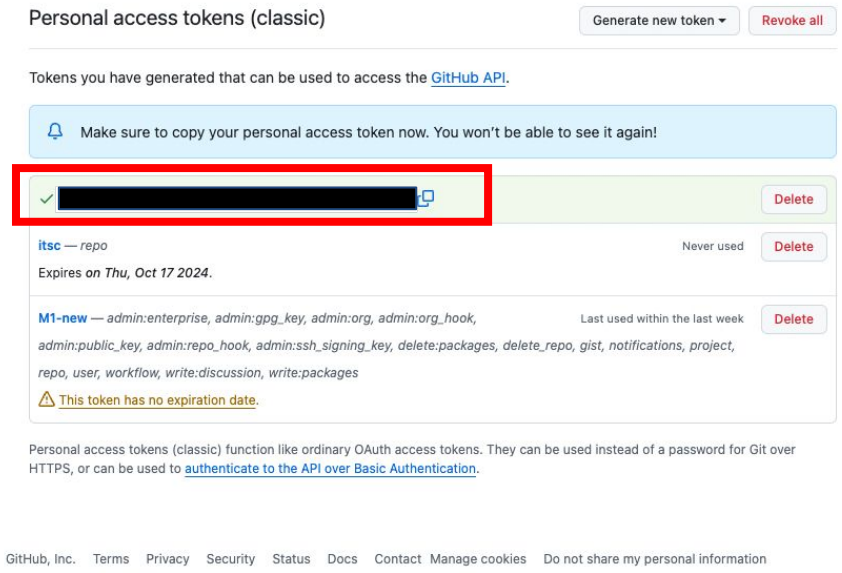
1. Setting up Git (3/6)

- Git authentication (cont…)
  - Provide a name for the token
  - Configure token expiration day
  - Select token scope (ex: what can someone do with the token - generally ‘repo’ scope is sufficient)



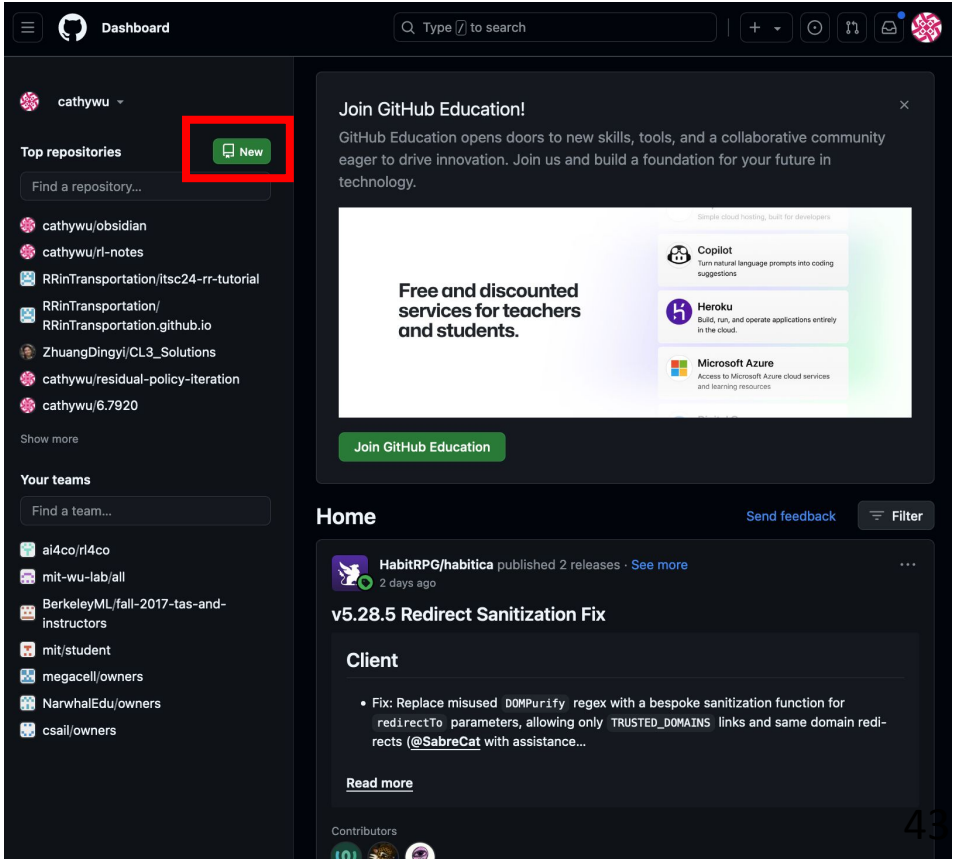
1. Setting up Git (4/6)

- Git authentication (cont…)
  - Copy the token.
    - Note: once you refresh the page, you can not no longer view the token and will have to regenerate.
  - Keep this token saved somewhere as we will use this later



2. Create a new repo on Github (1/2)

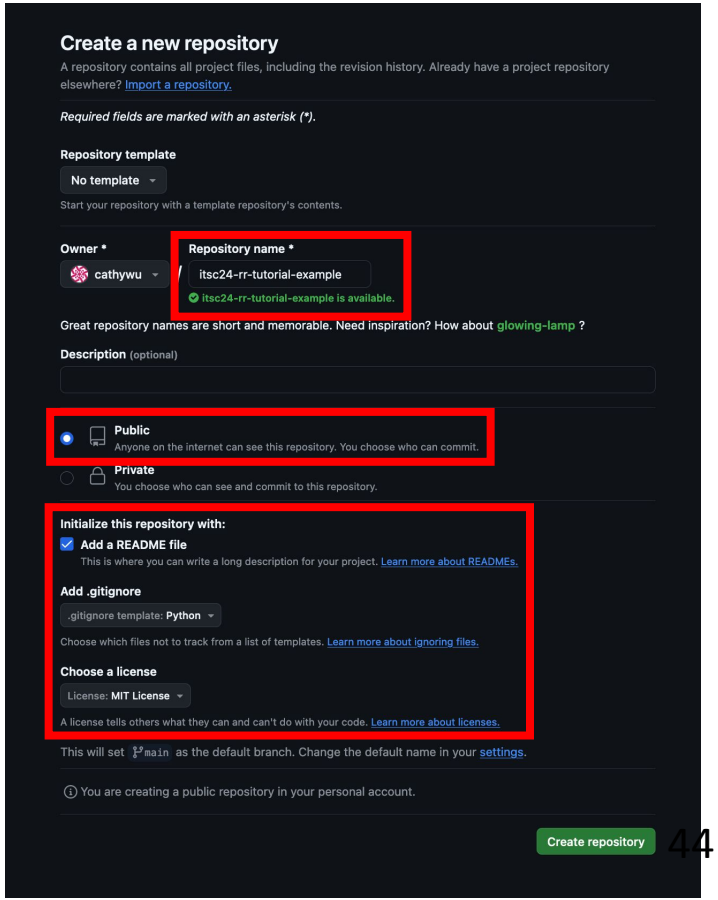
- Sign in to github.com



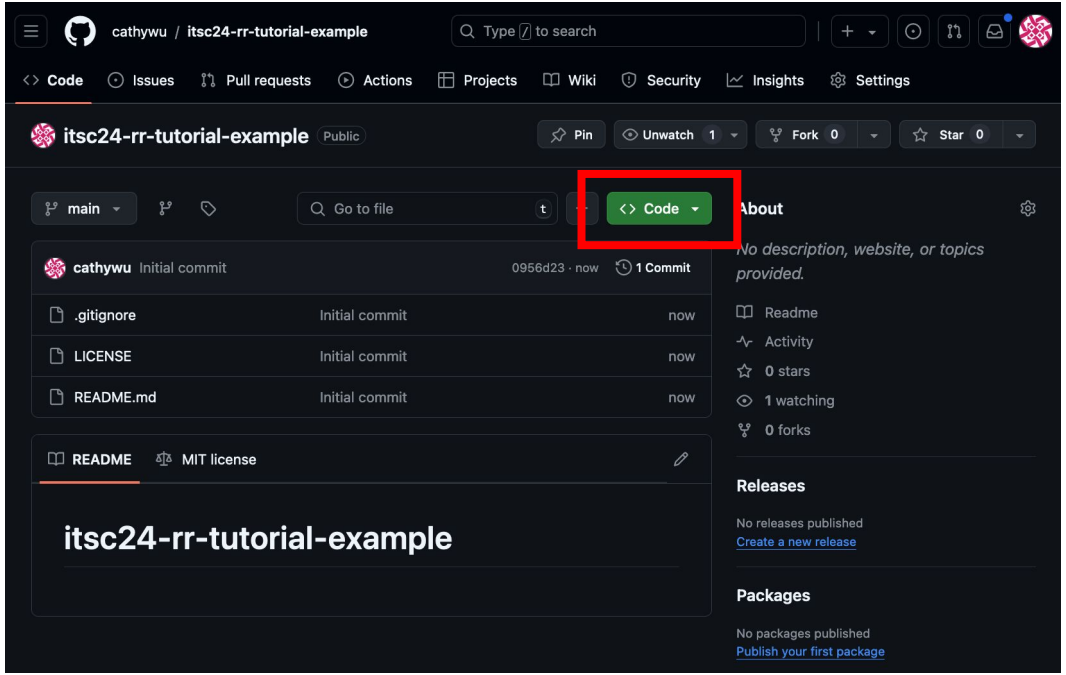
2. Create a new repo on Github (2/2)

- Choose a repo name and select an open-source license

- `.gitignore` files indicate which files NOT to add to the repo, like config or password files.



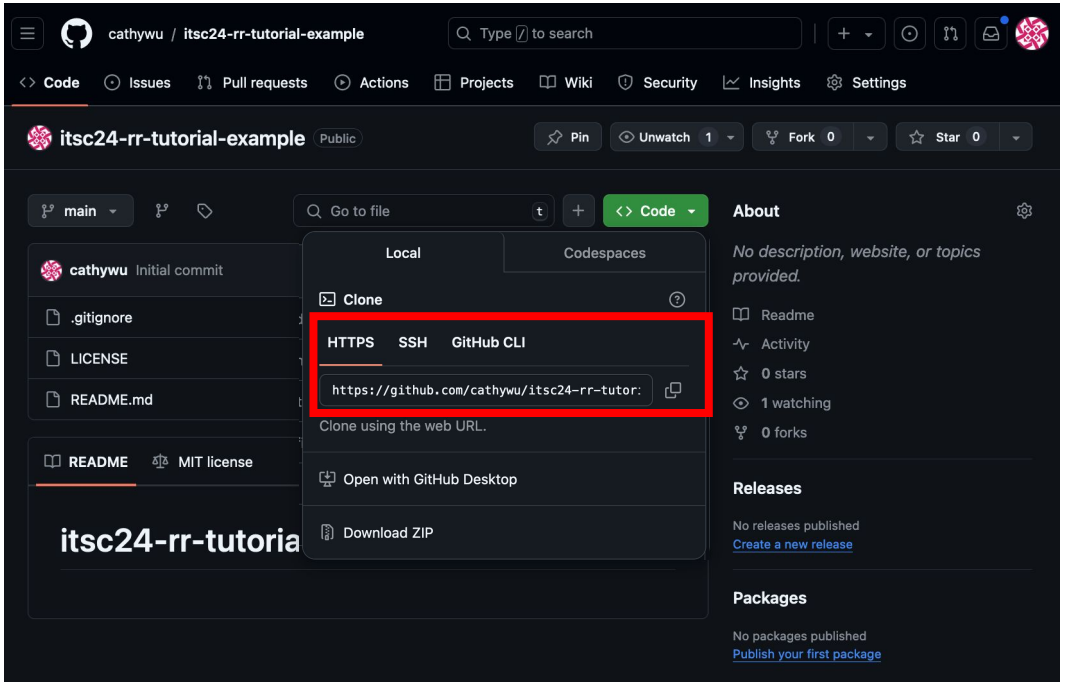
3. Download the empty repo to your local machine (1/2)



3. Download the empty repo to your local machine (1/2)

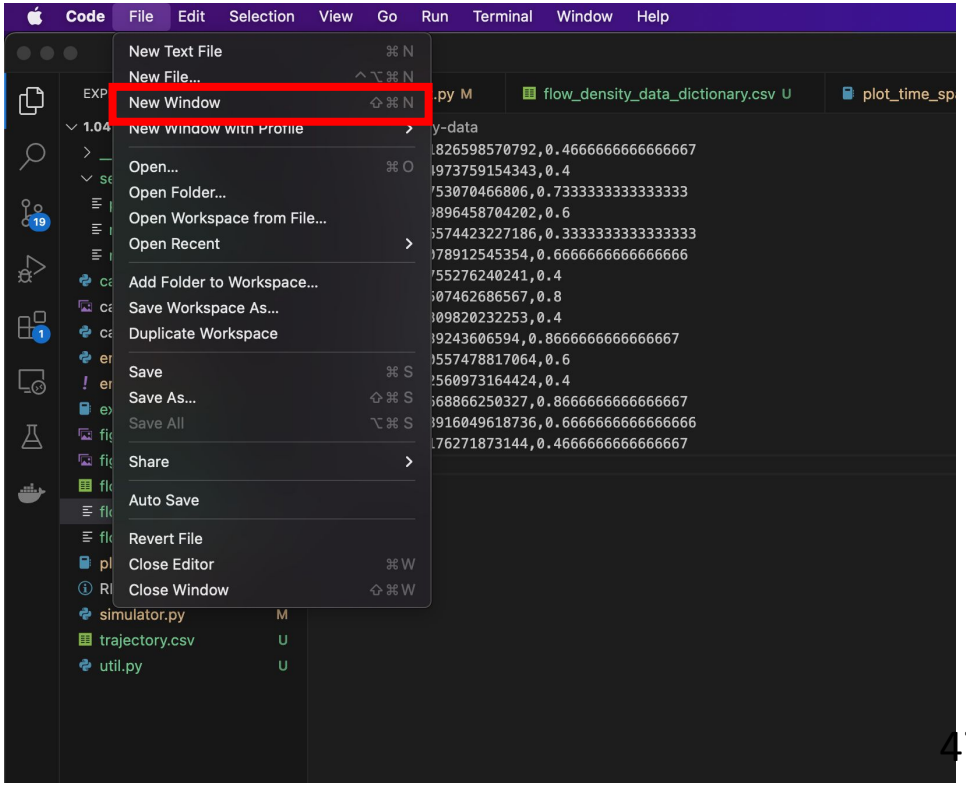
- On the website:
  - Copy the `https://github.com/` …
  - Use the HTTPS option

- In the local terminal
  - Navigate to where you want to download the repo (`$ cd` …)
  - `$ git clone https://github.com/` …



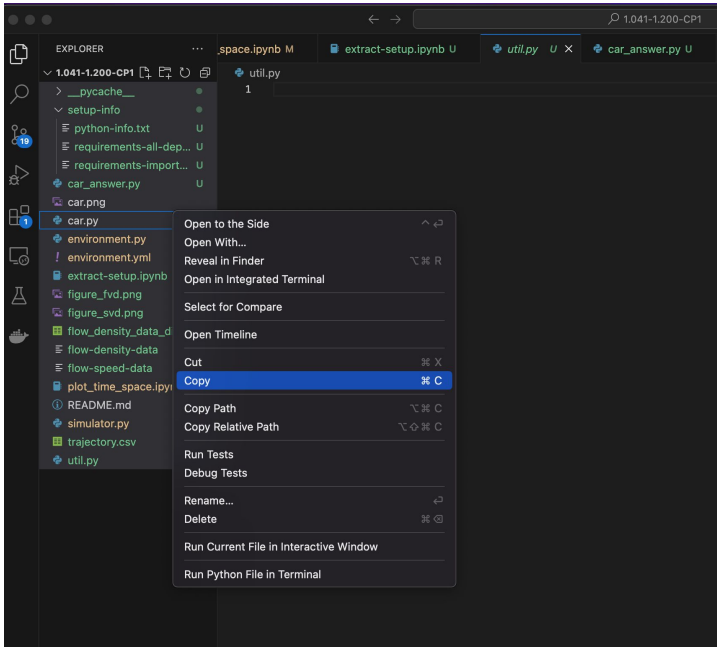
4. Add files to a new workspace for the repo (1/2)

- Open up the repo in Visual Studio Code
  - New Window
  - Open Folder…
  - Navigate to where you put the repo

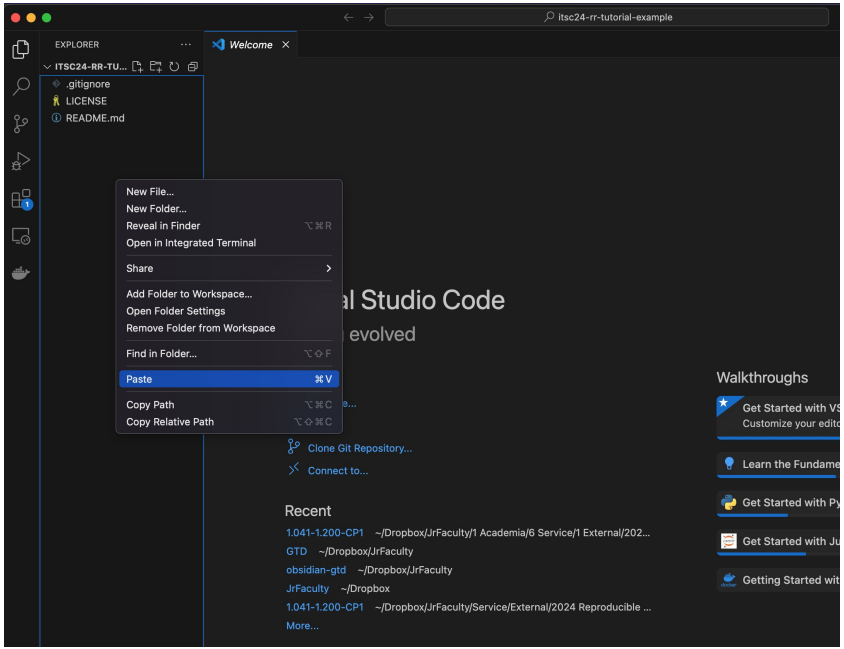


4. Add files to a new workspace for the repo (2/2)

- Copy all the files from the workspace you've been working in
- Paste them into the new workspace you just opened



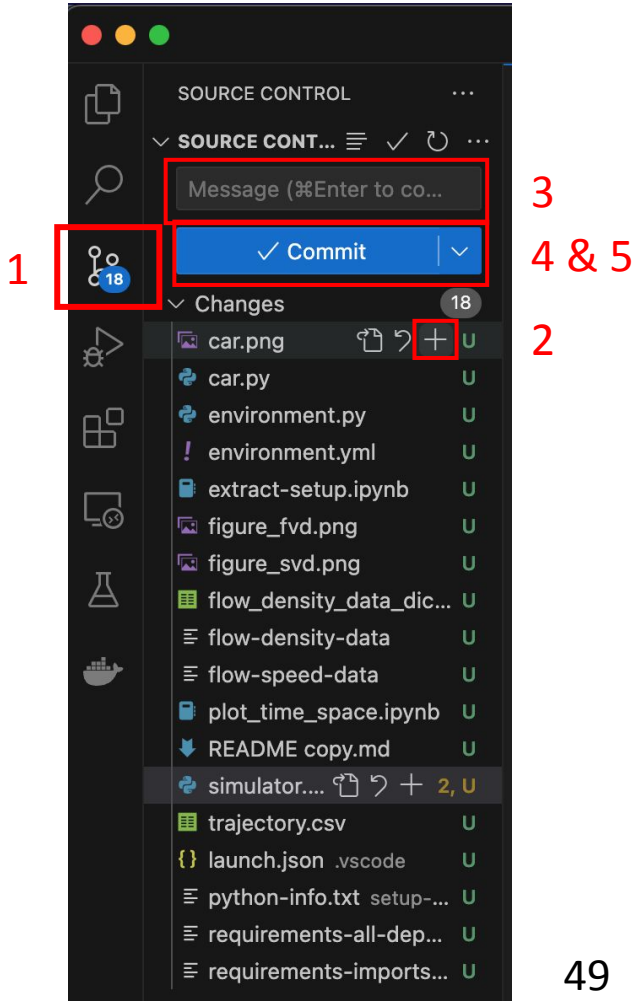
Old workspace



New workspace

5. Commit & sync files to Github

- Commit = designate the files you want to add/change to the repo (i.e., upload to Github)
- First, make sure the code in the new folder runs as expected
- Then, add the files you want to commit to your repo by:
  1. Navigate to the source control tab (left side panel)
  2. Click the "+" on the corresponding files
- Commit and sync!
  3. Add a commit message describing what you are uploading, like "A fully working car following simulation example."
  4. Click Commit
  5. Click Sync Changes
  6. VSCode will prompt a window and ask you to sign into your Github account. Once signed, the commit will be synced.
- Check that your changes now appear in your repo on Github!
  - Refresh the page



5. Commit & sync files to Github (terminal)

- If you prefer, you can do all this in the terminal
- `$ git status`
  - See which files have changed
- `$ git add <file>`
  - Add the files you wish to commit
- `$ git status`
  - See which files you have staged for committing, remaining files which have changed
- `$ git commit -m "<commit message>"`
  - Commit the changes with a commit message describing what you are uploading
- `$ git pull`
  - In this case, this won't do anything, but it's always good practice to pull in changes to the repo (that your collaborators may have made) before pushing your new changes
- `$ git push`
  - This pushes your commit to Github!

Further reading (git): https://github.com/kbroman/Tools4RR/blob/main/04_Git/GitCommands/git_notes.md

Show time!

- Double check that the project works and reproduces the plots
  - Exit conda environment: `$ conda deactivate`
  - Clone a fresh copy of the project from Github into a new directory
    - Navigate to where you want to put the new directory. You want to explicitly give the new directory name so that it does not conflict with the repo you original cloned
    - `$ git clone git@github.com:... <new directory name>`
  - Follow the README and confirm that it works
    - The data and plots should look **exactly** the same as in the Github repo!

- Show us that it works

- **Bonus hard mode**: Ask someone else to confirm that it works.

Optional tips

- Your project probably won't work the first time you try to run your own instructions. Here are some suggestions on what to do when it doesn't work.

- If you conda environment creation fails, you'll need to delete the partially created conda environment before trying again
  - Command to delete conda env: `$ rm -rf /opt/anaconda3/envs/RR`

Outline

1. Make it run
2. Make it available
3. **Make it pretty**
   a. Organizing data and code
   b. Making your code readable

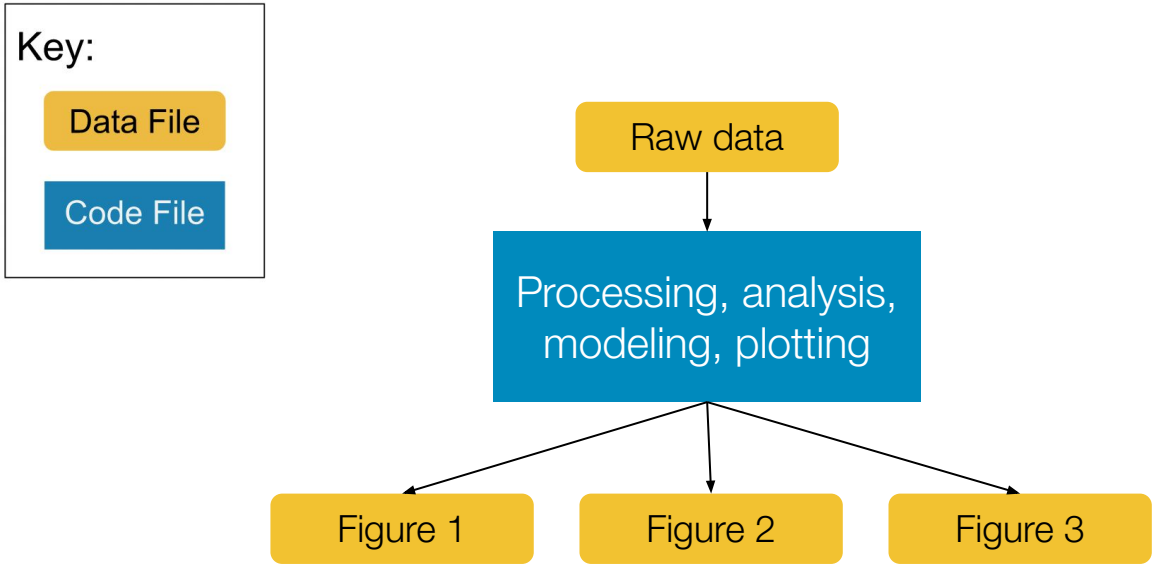
Outline

1. Make it run
2. Make it available
3. **Make it pretty**
   a. **Organizing data and code**
   b. Making your code readable

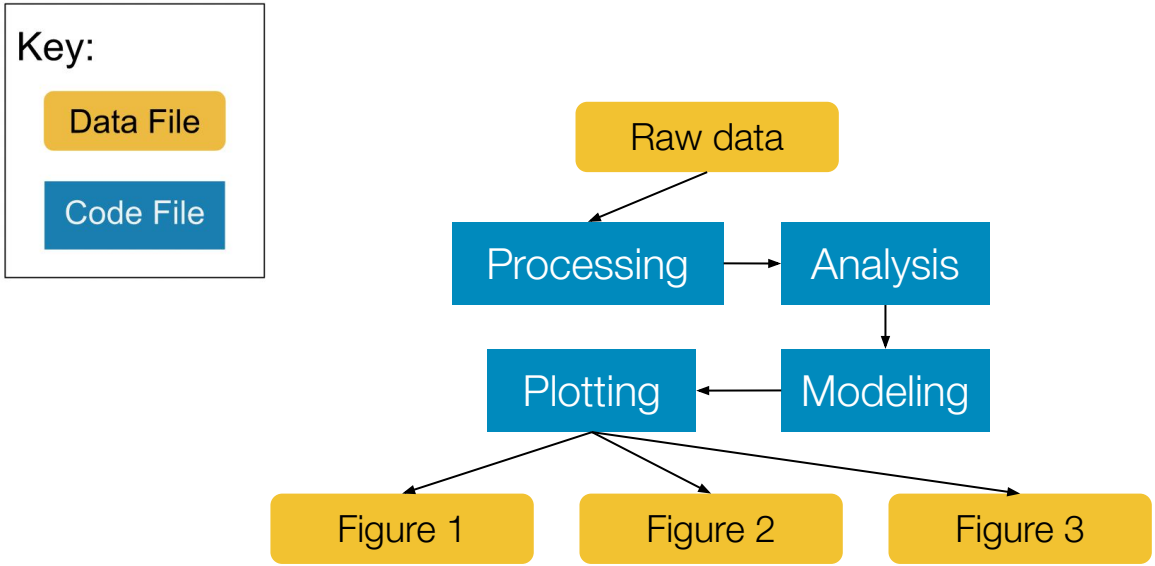
Organizing data and code

The best time to figure out how you want your project to be organized is before you begin. The second best time is now!

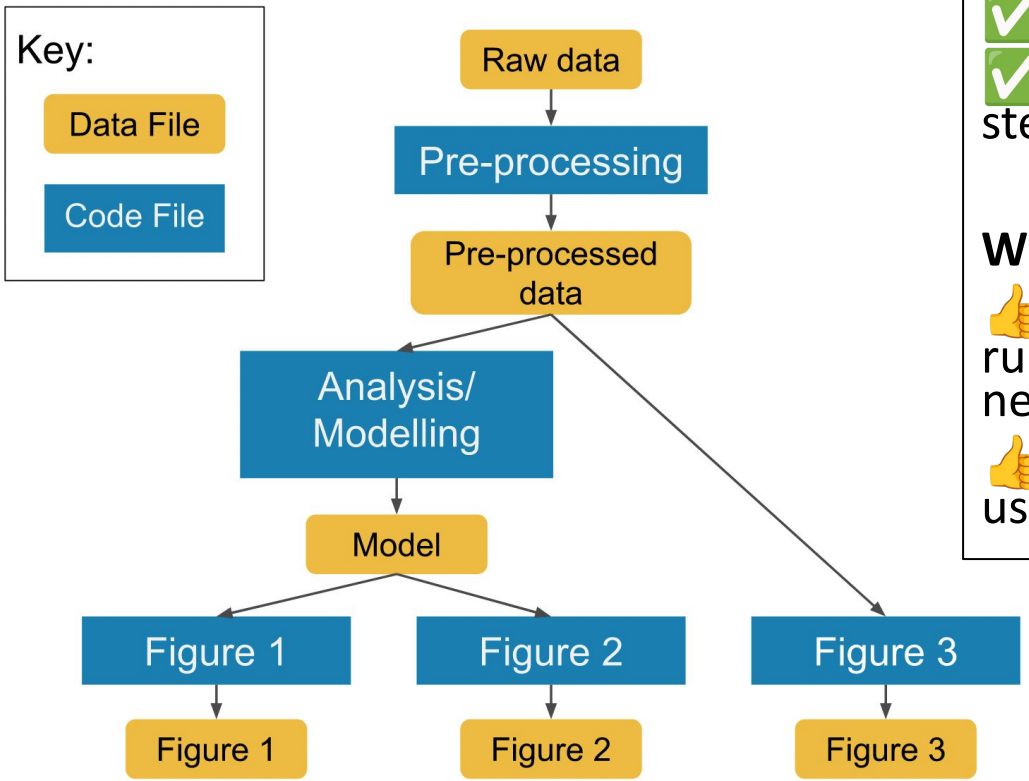
Organizing code and data



Organizing code and data



Organizing code and data



Organization principles

✅ As modular as possible

✅ Save the output of every step

Why?

👍 Research productivity: avoid running the full pipeline unless necessary

👍 Ease reproducibility: can use intermediate outputs

Exercise: Organize the ITS code example into directories

"Reproducible research best practices @JupyterCon." 8 Sept. 2024, www.kaggle.com/code/rtatman/reproducible-research-best-practices-jupytercon.

Organization takes time.

Be prepared to refactor as the project evolves.

The sooner the better, to minimize “tech debt.”

Car following simulation

- **Discussion:**
  - (a) Is it easy to understand what everything is?
  - (b) Did you document the process of data preparation and analysis?
  - (c) How might you re-organize things, to make it more clear to someone else, or to you a year from now?
  - (d) Do you need more documentation (e.g., README files) or are the file names and organization self-explanatory?

Suggestions for organizing code and data

- Modularize the codebase by splitting the simulation code from the plotting code.
- Provide an example of how to plot the figures in a Jupyter Notebook, separate from the python scripts.
- Create separate directories for assets, data, figures.
- Create a config.py for shared parameters.

Outline

1. Make it run
2. Make it available
3. **Make it pretty**
   a. Organizing data and code
   b. **Making your code readable**

What does this code do?

```python
import numpy as np
M, s1 = 0, 0.1
 S2 = np.random.normal(M, s1, 1000)
import matplotlib.pyplot as plt; x, Y, zZ = plt.hist(S2,
30, density=True); plt.plot(Y, 1/(s1 * np.sqrt(2 *
np.pi)) * np.exp( - (Y - M)**2 / (2 * s1**2) ),
linewidth=2, color='r'); plt.show()
```

What does this code do?

```python
import numpy as np
import matplotlib.pyplot as pltsigma

# generate random normal distribution
mu, sigma = 0, 0.1 # mean and standard deviation
sample = np.random.normal(mu, sigma, 1000)

# plot histogram with density curve
count, bins, ignored = plt.hist(sample, 30, density=True)
plt.plot(bins, 1/(sigma * np.sqrt(2 * np.pi)) *
         np.exp( - (bins - mu)**2 / (2 * sigma**2) ),
         linewidth=2, color='r')
plt.show()
```

Make it pretty: making your code readable

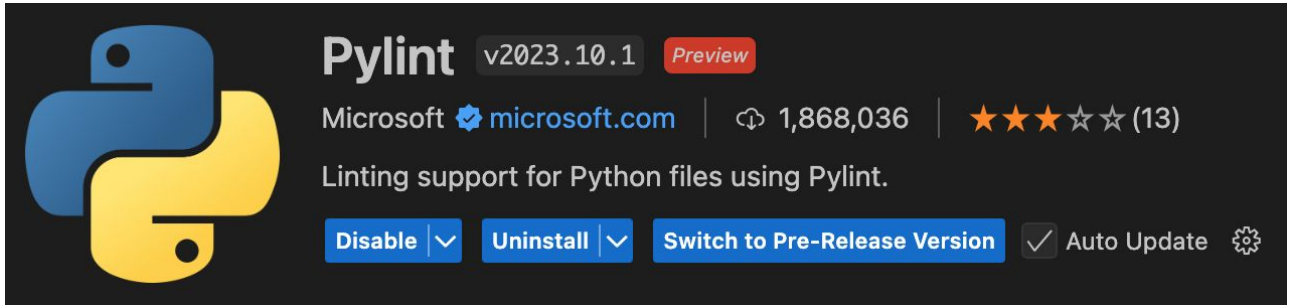
Principles for making your code readable

- Help readers follow the logic
  - Logical and human readable variable names
  - Modular classes and functions
  - Comment your code! Explain your thought process & design choices
- Ease readability by following stylistic conventions
  - Put all package imports at the top of the file
  - Use white space for readability
    - Break up long lines (keep lines <120 characters)
  - Consistent patterns of capitalization (e.g., ALL CAPS for constants, `lower_case_with_underscores` for functions and variables, `lower-case-with-dashes` for directories)

Style guides

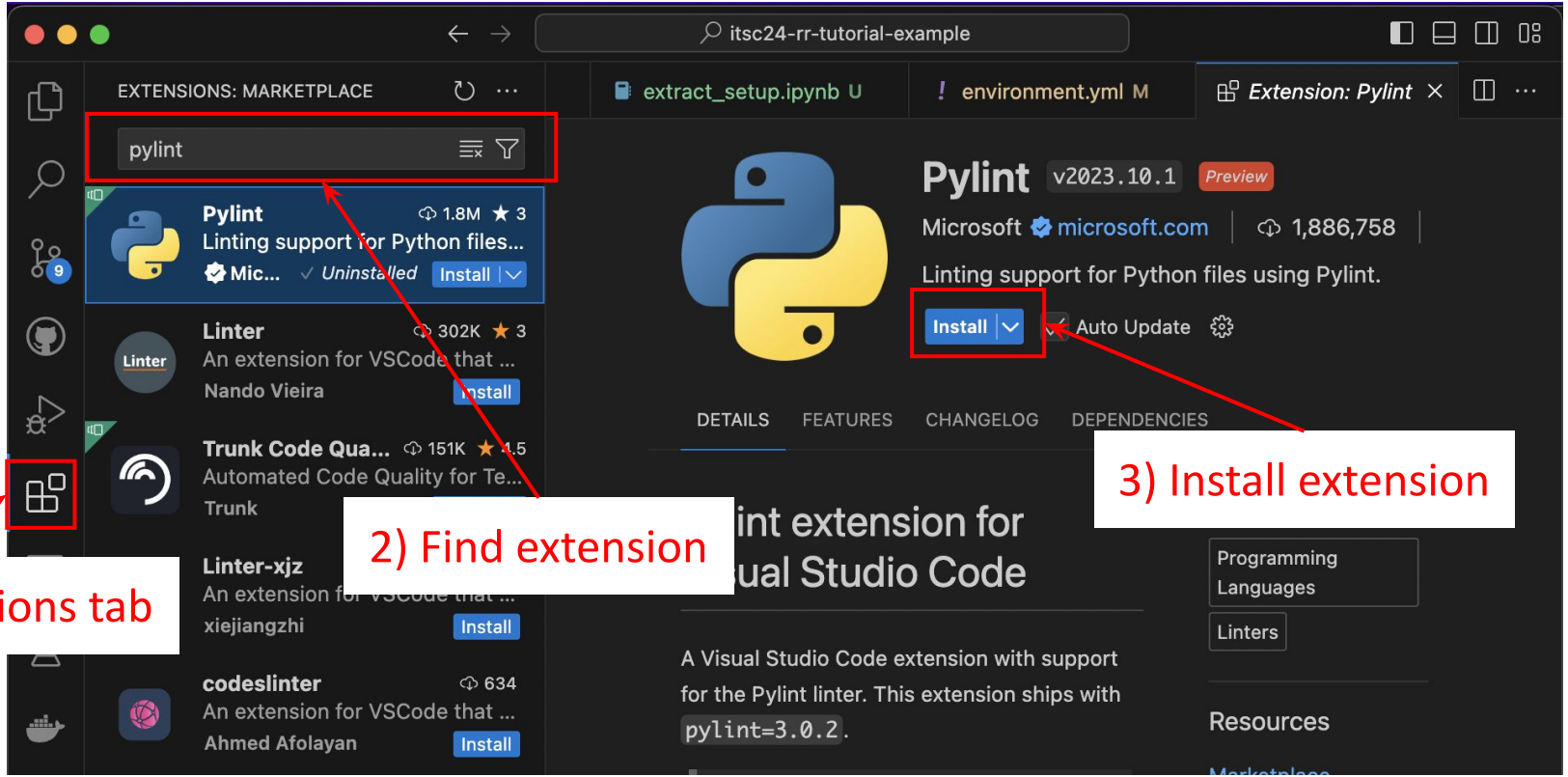
- Style guides for academic writing:
  - MLA, APA or Chicago/Turabian
- Style guides for code (Python)
  - PEP 8, Google style guide
- **Linter**: Tools that analyze source code to flag programming errors, bugs, stylistic errors, and suspicious constructs.
  - Like a grammar checker but for code
  - Ex. Pylint for Python

Pylint + Visual Studio Code demo

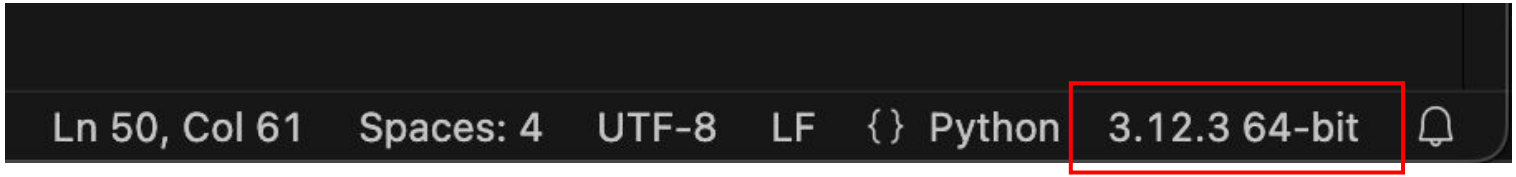


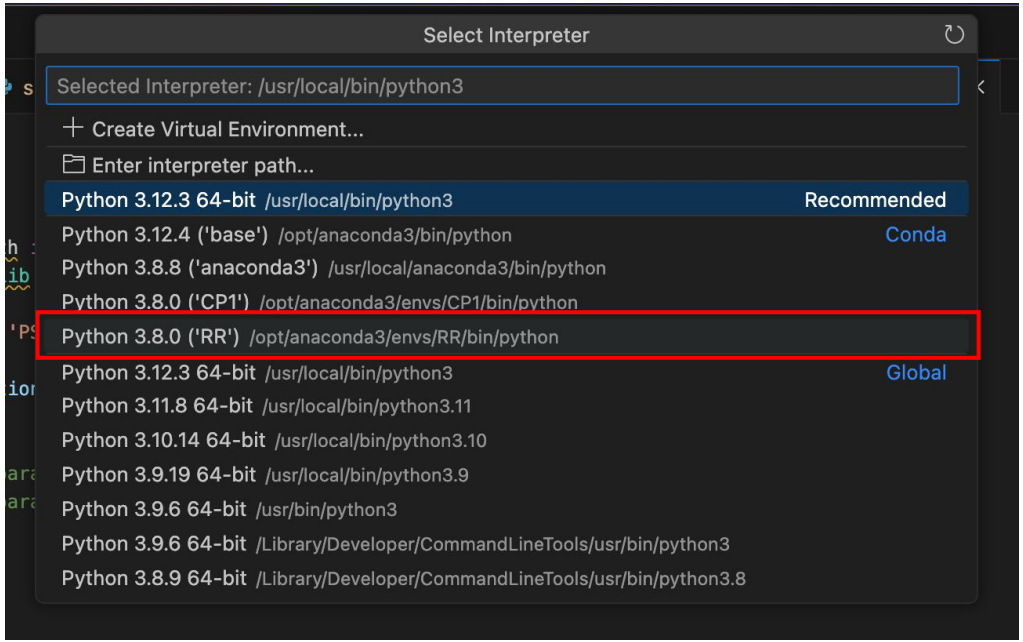
Further reading: https://code.visualstudio.com/docs/python/linting

Adding your first VS Code extension



Setting up pylint: select the python interpreter





Select python interpreter

Using pylint: Problems pane

PROBLEMS 52 OUTPUT DEBUG CONSOLE TERMINAL PORTS JUPYTER

car.py 52

- No name 'Vector2' in module 'pygame.math' Pylint(E0611:no-name-in-module) [Ln 2, Col 1]
- TODO Define parameter specifications of your IDM model Pylint(W0511:fixme) [Ln 12, Col 2]
- TODO : In this method, make the position and velocity update of the vehicle according Pylint(W0511:fixme) [Ln 92, Col 10]
- TODO: Calculate the amount of change in position here Pylint(W0511:fixme) [Ln 111, Col 10]
- TODO Define your IDM variant. Pylint(W0511:fixme) [Ln 120, Col 10]
- Unused argument 'dt' Pylint(W0613:unused-argument) [Ln 177, Col 26]
- Unused argument 'vehicle_lead' Pylint(W0613:unused-argument) [Ln 177, Col 30]
- Missing module docstring Pylint(C0114:missing-module-docstring) [Ln 1, Col 1]
- standard import "import math" should be placed before "import pygame" Pylint(C0411:wrong-imp Missing module docstring
- Constant name "ppu" doesn't conform to UPPER_CASE naming style Pylint(C0103:invalid-name) [Ln 9, Col 1]
- Constant name "a_max" doesn't conform to UPPER_CASE naming style Pylint(C0103:invalid-name) [Ln 14, Col 1]
- Constant name "b" doesn't conform to UPPER_CASE naming style Pylint(C0103:invalid-name) [Ln 15, Col 1]
- Constant name "delta" doesn't conform to UPPER_CASE naming style Pylint(C0103:invalid-name) [Ln 16, Col 1]
- Constant name "s_min" doesn't conform to UPPER_CASE naming style Pylint(C0103:invalid-name) [Ln 17, Col 1]
- Constant name "max_velocity" doesn't conform to UPPER_CASE naming style Pylint(C0103:invalid-name) [Ln 18, Col 1]
- Missing class docstring Pylint(C0115:missing-class-docstring) [Ln 20, Col 1]
- Missing function or method docstring Pylint(C0116:missing-function-docstring) [Ln 30, Col 5]
- Missing function or method docstring Pylint(C0116:missing-function-docstring) [Ln 41, Col 5]

Show time!

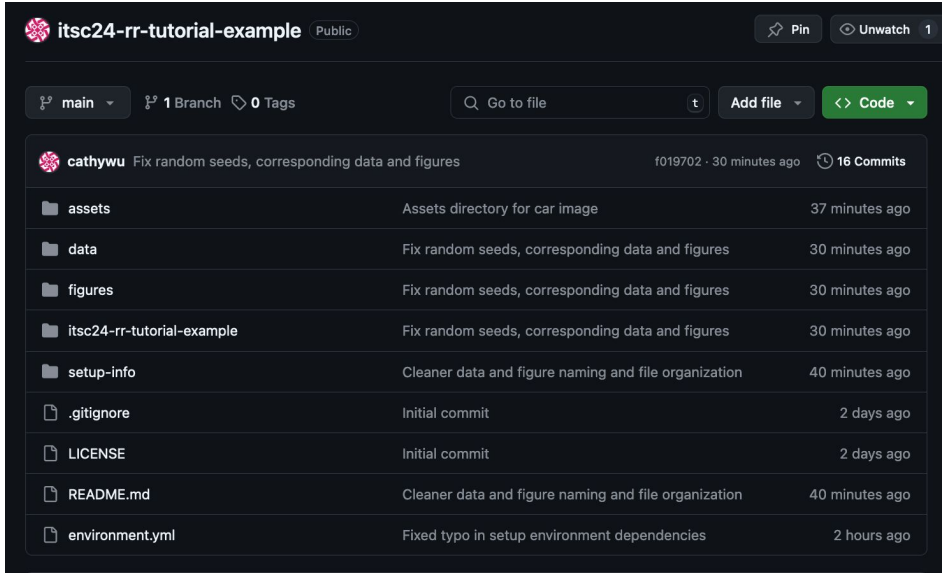
- Commit your changes (same as before)
- Show us that it works (same as before)

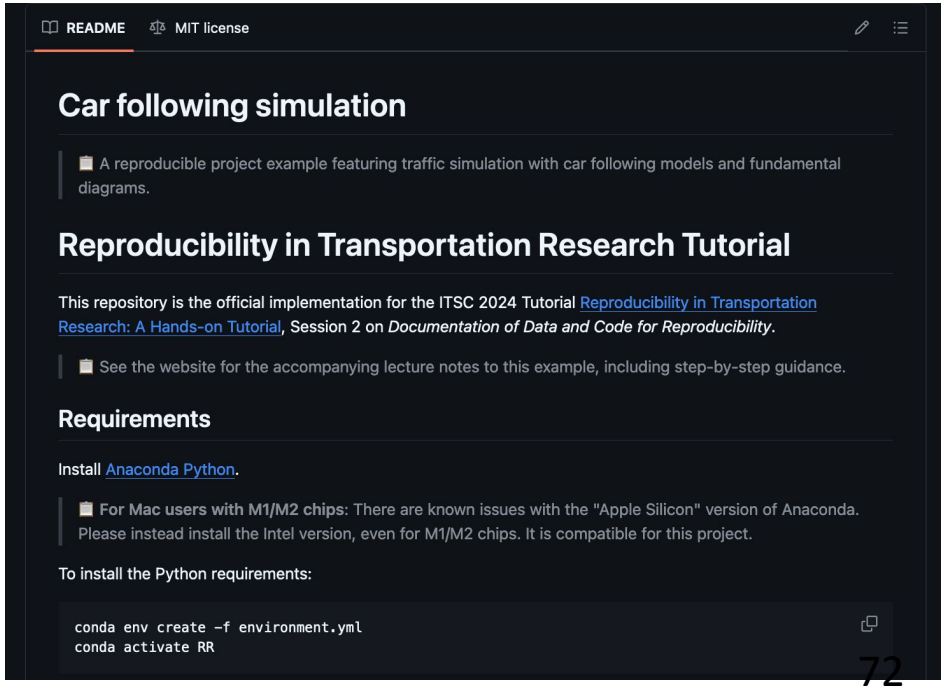
- Exchange projects with someone and ask them to critique yours. What organization could help someone more easily build on the project?

- Remember, that someone could be you in 6 months!

Reference implementation (try it yourself first!)

- See https://github.com/cathywu/itsc24-rr-tutorial-example for a reference implementation of this session's hands on activity.





Acknowledgements

- Main contributor:
  - Cathy Wu
- Contributing content
  - Vindula Jayawardana, Jung-Hoon Cho
- Testers
  - Cathy Wu's research group, Xiaoyi Wu, Kate Sanborn
- Brainstorming
  - Bidisha Ghosh, Irene Martínez, Nicholas Saunier
- Support
  - RERITE working group
- CPS-IoT Week
  - For giving us an opportunity to broadcast the efforts