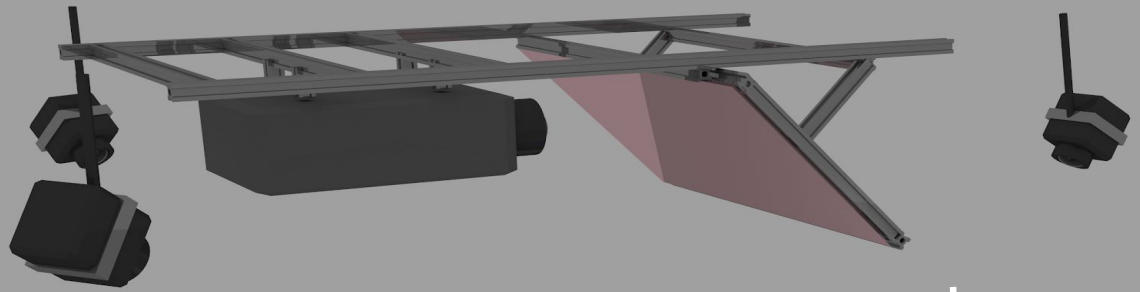
Scientific DevOps:
Designing Reproducible Data Analysis Pipelines with Containerized Workflow Managers

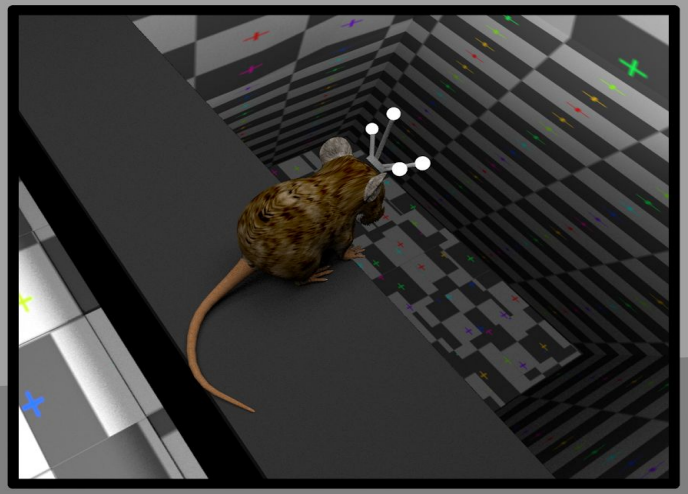
Nicholas Del Grosso

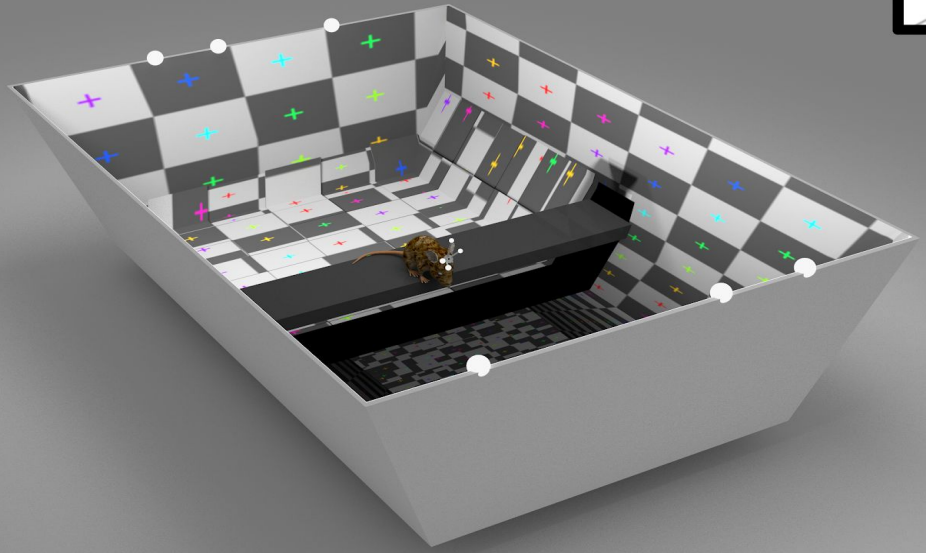
EuroScipy 2019

Bilbao, Spain



Closed-Loop Research and Software Development





Before Starting, Let’s Get Into the Shoes of a Researcher...

Data Analysis Requested...

You request data from a study you're excited about, hoping to replicate it and build your new study on top of that one's findings, and write to the study authors.

A month later, they send you a zip file with code and data, but after a week of trying to get it to run, it becomes clear that what you have is incomplete.

Should you still begin your study?

Should you request more time to search for a new direction?

Do you know, for sure, that the files you received don't work as-is?

A little help...

Data collection takes all day, but the department has hired a student assistant to help you out. How long will training take?

If the new data has collection-related problems, will you catch them in time to retrain your assistant?

Is it worth getting help, or should you work extra hours on the weekend to keep the status quo and still make your deadline?

Collaborating on the analysis...

Instead, you request help on the data analysis, describing a new analysis approaches you'd like done and giving them your analysis scripts so they can write more.

A month later, they come back with a two figures--one you requested that shows the pattern you had hoped for, and another that is an "improvement" of an another analysis, but doesn't show any clear differences. They give you their code, which now has hundreds of lines of changes in many files.

Who's code do you work on from that point on: your original files, or theirs? Do you accept their code?

Death of the Magic Machine...

You go back to the experiment machine, only to find that your data acquisition software won’t start! It’s the only computer your department has with that program, and the lab member that set it up and modified it left long ago.

After a couple days of searching online, you finally find the key program online.

Should you use it to collect your data?

How will this new change it affect what you do with the existing data?

Have you tried...

After six months of data collection and analysis, you give a presentation on your results to your lab/department, and someone suggests a change in methods or a new parameter.

What do you do with that feedback?

Do you remember what parameter you used in the first place?

How much effort will it be to change your code and re-run the analysis in the first place?

Accepted, with Revisions...

The manuscript you've been writing for 2 years is under review, but the editors want a specific new figure before accepting your paper.

How much work will it be to make the requested changes?

Is it worth submitting to a more-lenient journal?

Do you write a rebuttal letter, explaining that it is too much work or claim that the suggestion is out of scope for your study?

Data Analysis Requested...

Your paper is finally accepted! You move on to a new laboratory and continue doing amazing work, but one day you receive an email requesting the code and data for your prior work.

Do you know where it all is?

How much effort will it be to find it?

How much of it needs to be modified before sending it out?

How long does it take to reply to the email?

DevOps in Research Software

Research Software is...

- ...intended to produce validated and verifiable knowledge about our world.
- ...often specialized for a specific experiment or field of knowledge
- ...often built and used by only a few individuals.
- ...intentionally archived.
- ...often used **without ever being run**.

Some Takeways

DevOps is...

- An observation that all steps in a pipeline contribute toward the achievement of some goal.
- the observation of short-sightedness within individual steps in a pipeline.
- An observation of how much wasted effort this short-sightedness produces.
- The combination of both top-down and bottom-up methods for solving a problem and finding solutions.
- An application of the scientific method and queuing theory to productivity.
- A set of management principles adapted from the Lean community to software.
- A long-view mindset that recognizes change and ever-increasing knowledge as essential and never-ending steps in achieving a goal.
- A set of software tools to help a team work with that mindset and build and maintain a productive environment.

The Three “Ways” of DevOps

1. Build Smooth Forward Flow
2. Increase Feedback
3. Foster an Environment of Continuous Improvement

Version Control: Essential Tool 1

- Take snapshots of a folder, creating a timeline of changes to that folder.
- Has tools for copying that timeline and making alternative timelines.
- Asks users to describe, in their own words, the point of each snapshot.

Version Control System

Code Hosting Website & Social Network

Code Hosting Tool

Building Smooth, Forward Flow

The First Way

- MyProject
  - README.txt

README.txt

My Project
This project makes histograms of my awesome data!

My Project
This project makes histograms of my awesome data!

- MyProject
  - README.md

README.md

```
# My Project

This project makes histograms of my awesome data!

## Run the Analysis

```python
import pandas as pd
df = pd.read_csv("mydata1.csv")
fig = df.plot.hist()
fig.savefig("myresults.png")
```
```

My Project

This project makes histograms of my awesome data!

Run the Analysis

```
import pandas as pd
df = pd.read_csv("mydata1.csv")
fig = df.plot.hist()
fig.savefig("myresults.png")
```

```
cat README.md | codedown python | python
```

- MyProject
  - README.md
  - get_results.py

get_results.py

```
import pandas

read_csv
plot_histogram
savefig
```

My Project

This project makes histograms of my awesome data!

Run the Analysis

```
python get_results.py
```

- MyProject
  - README.md
  - get_results.py

My Project

This project makes histograms of my awesome data!

Install the Dependencies

```
pip install pandas
pip install numpy
pip install matplotlib
```

Run the Analysis

```
python get_results.py
```

- MyProject
  - README.md
  - get_results.py
  - install_requirements.sh

install_requirements.sh

```
pip install pandas
pip install numpy
pip install matplotlib
```

My Project

This project makes histograms of my awesome data!

Install the Dependencies

```
bash install_requirements.sh
```

Run the Analysis

```
python get_results.py
```

- MyProject
  - README.md
  - get_results.py
  - requirements.txt

requirements.txt

```
pandas
matplotlib
numpy
```

My Project

This project makes histograms of my awesome data!

Install the Dependencies

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

Run the Analysis

```
python get_results.py
```

- MyProject
  - README.md
  - get_results.py
  - requirements.txt
  - requirements.lock

requirements.lock

```
cycler==0.10.0
kiwisolver==1.1.0
matplotlib==3.1.1
numpy==1.17.1
pandas==0.25.1
pyparsing==2.4.2
python-dateutil==2.8.0
pytz==2019.2
six==1.12.0
```

My Project

This project makes histograms of my awesome data!

Install the Dependencies

```
pip install -r requirements.lock
```

Run the Analysis

```
python get_results.py
```

Developers

Save the current versions of the packages:

```
pip freeze > requirements.lock
```

- MyProject
  - README.md
  - get_results.py
  - requirements.txt
  - requirements.lock

- (Virtual Environments)
  - myproject
    - bin
      - python
      - activate

My Project

This project makes histograms of my awesome data!

Install the Dependencies

```
conda create --name myproject python=3.7
conda activate myproject
pip install -r requirements.lock
```

Run the Analysis

```
conda activate myproject
python get_results.py
```

Developers

Save the current versions of the packages:

```
pip freeze > requirements.lock
```

- MyProject
  - README.md
  - get_results.py
  - environment.yml
  - requirements.lock

My Project

This project makes histograms of my awesome data!

Install the Dependencies

```
conda create --name myproject python=3.7
conda activate myproject
conda install --file requirements.lock
```

Run the Analysis

```
conda activate myproject
python get_results.py
```

Developers

Save the current versions of the packages:

```
conda list --export  > requirements.lock
```

- MyProject
  - README.md
  - get_results.py
  - Pipfile
  - Pipfile.lock

My Project

This project makes histograms of my awesome data!

Install the Dependencies

```
pip install pipenv
pipenv install
```

Run the Analysis

```
pipenv shell
python get_results
```

```
cd MyProject
pipenv --python 3.7
pipenv install pandas
```

- MyProject
  - README.md
  - get_results.py
  - Pipfile
  - Pipfile.lock

My Project

This project makes histograms of my awesome data!

Install the Dependencies

```
pip install pipenv
pipenv install
```

Run the Analysis

```
pipenv run get_results.py
```

```
cd MyProject
pipenv --python 3.7
pipenv install pandas
pipenv lock
```

- MyProject
  - README.md
  - get_results.py
  - pyproject.toml
  - poetry.lock

pyproject.toml

```
[tool.poetry]
name = "MyProject"
version = "0.1.0"
description = ""
authors = ["Nicholas A. Del Grosso
<delgrosso.nick@gmail.com>"]

[tool.poetry.dependencies]
python = "^3.6"
pandas = "^0.25.1"

[tool.poetry.dev-dependencies]
pytest = "^3.0"
```

My Project

This project makes histograms of my awesome data!

Install the Dependencies

```
pip install poetry
poetry install
```

Run the Analysis

```
poetry shell
python get_results.py
```

```
poetry MyProject
cd MyProject
pipenv add pandas
```

- MyProject
  - README.md
  - get_results.py
  - pyproject.toml
  - poetry.lock

pyproject.toml

```
[tool.poetry]
name = "MyProject"
version = "0.1.0"
description = ""
authors = ["Nicholas A. Del Grosso
<delgrosso.nick@gmail.com>"]

[tool.poetry.dependencies]
python = "^3.7"
pandas = "^0.25.1"

[tool.poetry.dev-dependencies]
pytest = "^3.0"
```

My Project

This project makes histograms of my awesome data!

Install the Dependencies

```
pip install poetry
poetry install
```

Run the Analysis

```
poetry run python get_results.py
```

- MyProject
  - README.md
  - get_results.py
  - pyproject.toml
  - poetry.lock

pyproject.toml

```
[tool.poetry]
name = "MyProject"
version = "0.1.0"
description = ""
authors = ["Nicholas A. Del Grosso
<delgrosso.nick@gmail.com>"]

[tool.poetry.dependencies]
python = "^3.7"
pandas = "^0.25.1"

[tool.poetry.dev-dependencies]
pytest = "^3.0"
```

My Project

This project makes histograms of my awesome data!

Install the Dependencies

(Runs on Ubuntu 16.4. I haven't tested it on Windows.)

```
pip install poetry
poetry install
```

Run the Analysis

```
poetry run python get_results.py
```

- MyProject
  - README.md
  - get_results.py
  - pyproject.toml
  - poetry.lock
  - Singularity

Dockerfile

```
FROM python:3.7-slim

WORKDIR /MyProject

COPY . /MyProject

RUN pip install poetry
RUN poetry install

CMD ['bash']
```

Singularity Recipe

```
Bootstrap: docker
From: python:3.7-slim

%files
  ./pyproject.toml .
  ./poetry.lock .

%post
  pip install poetry
  poetry install
```

- MyProject
  - README.md
  - get_results.py
  - pyproject.toml
  - poetry.lock
  - Singularity
  - myproject.simg

My Project

This project makes histograms of my awesome data!

Install the Dependencies

1. install singularity

Run the Analysis

```
singularity shell myproject.simg
python scripts/get_results.py
```

Building the Singularity Image

```
sudo singularity build myproject.simg Singularity
```

- MyProject
  - README.md
  - get_results.py
  - pyproject.toml
  - poetry.lock
  - Singularity
  - myproject.simg

My Project

This project makes histograms of my awesome data!

Install the Dependencies

1. install singularity

Run the Analysis

singularity run myproject.simg

```
...

%runscript
  python get_results.py
```

- MyProject
  - README.md
  - get_results.py
  - pyproject.toml
  - poetry.lock
  - Singularity
  - myproject.simg

My Project

This project makes histograms of my awesome data!

Install the Dependencies

1. install singularity
2. Get the data.

Run the Analysis

```
singularity run myproject.simg
```

- MyProject
  - README.md
  - get_results.py
  - pyproject.toml
  - poetry.lock
  - Singularity
  - myproject.simg

- /data
  - raw
    - d1.csv
    - d2.csv

RENKU

Collaborative Data Science

Talk to Chandrasekhar Ramakrishnan and Rok Roškar for more info, here at the conference!

GIN

Modern Research Data Management for Neuroscience

Quilt

Manage data like code

Quilt versions and deploys data

- MyProject
  - README.md
  - get_results.py
  - pyproject.toml
  - poetry.lock
  - Singularity
  - myproject.simg

- /data
  - raw
    - d1.csv
    - d2.csv

My Project

This project makes histograms of my awesome data!

Install the Dependencies

1. install singularity

Run the Analysis

```
singularity run -B DataFolder:/data myproject.simg
```

- MyProject
  - README.md
  - get_results.py
  - pyproject.toml
  - poetry.lock
  - Singularity
  - myproject.simg

- /data
  - raw
    - d1.csv
    - d2.csv

My Project

This project makes histograms of my awesome data!

Install the Dependencies

1. install singularity

Run the Analysis

singularity run myproject.simg

```
...

%files
  DataFolder /data
```

- MyProject
  - **README.md**
  - **pyproject.toml**
  - **poetry.lock**
  - **Singularity**
  - **myproject.simg**
  - scripts
    - get_results.py

- /data
  - raw
    - d1.csv
    - d2.csv

My Project

This project makes histograms of my awesome data!

Install the Dependencies

1. install singularity

Run the Analysis

```
singularity run myproject.simg
```

```
...

%files
  DataFolder /data
```

Shrinking Your Batch Size

The First Way: Part 2

```
raw_files = glob("data/raw/data_*.csv")
processed_files = []

for raw_file in raw_files:
    session = path.basename(path.splitext(raw_file)[0]).split('_')[-1]

    # Preprocess Data
    df = pd.read_csv(raw_file)
    df2 = do_process(df)
    os.makedirs("data/processed", exist_ok=True)
    process_file = f"data/processed/data_{session}.h5"
    df2.to_hdf(processed_file, '/')
    processed_files.append(processed_file)

    # Make histograms
    fig = df2["Variable"].plot.hist()
    os.makedirs("results/hists", exist_ok=True)
    fig.savefig(f"results/hists/hist_{session}.svg")

# Make Figure1
df_all = pd.concat([pd.read_hdf(f, '/') for f in processed_files])
os.makedirs("results", exist_ok=True)
df_all.plot(x='Time', y='Happiness').savefig("results/figure1.svg")
```

get_results.py

```
raw_files = glob("data/raw/data_*.csv")
processed_files = []

for raw_file in raw_files:
    session = path.basename(path.splitext(raw_file)[0]).split('_')[-1]

    # Preprocess Data
    df = pd.read_csv(raw_file)
    df2 = do_process(df)
    os.makedirs("data/processed", exist_ok=True)
    process_file = f"data/processed/data_{session}.h5"
    df2.to_hdf(processed_file, '/')
    processed_files.append(processed_file)

    # Make histograms
    fig = df2["Variable"].plot.hist()
    os.makedirs("results/hists", exist_ok=True)
    fig.savefig(f"results/hists/hist_{session}.svg")

# Make Figure1
df_all = pd.concat([pd.read_hdf(f, '/') for f in processed_files])
os.makedirs(“results”, exist_ok=True)
df_all.plot(x='Time', y='Happiness').savefig("results/figure1.svg")
```

get_results.py

```
# Preprocess Data
for raw_file in glob("data/raw/data_*.csv"):
    session = path.basename(path.splitext(raw_file)[0]).split('_')[-1]
    df = pd.read_csv(raw_file)
    df2 = do_process(df)
    os.makedirs("data/processed", exist_ok=True)
    processed_file = f"data/processed/data_{session}.h5"
    df2.to_hdf(processed_file, '/')

# Make Histograms
for processed_file in glob("data/processed/data_*.h5"):
    session = path.basename(path.splitext(raw_file)[0]).split('_')[-1]
    df = pd.read_hdf(processed_file, '/')
    fig = df["Variable"].plot.hist()
    os.makedirs("results/hists", exist_ok=True)
    fig_file = f"results/hists/hist_{session}.svg"
    fig.savefig(fig_file)

# Make Figure 1
files = glob("data/processed/data_*.h5")
df_all = pd.concat([pd.read_hdf(f, '/') for f in  files])
fig1 = df_all.plot(x='Time', y='Happiness')
os.makedirs("results", exist_ok=True)
fig1.savefig("results/figure1.svg")
```

get_results.py

```
# Preprocess Data
inputs =  [f"/data/raw/data_{session}.csv" for session in sessions]
outputs = [f"/data/processed/data_{session}.h5" for session in sessions]
os.makedirs("/data/processed", exist_ok=True)
for input, output in zip(inputs, outputs):
    df = pd.read_csv(input)
    df2 = do_process(df)
    df2.to_hdf(output, '/')

# Make Histograms
inputs =  [f"/data/processed/data_{session}.h5" for session in sessions]
outputs = [f"results/hists/hist_{session}.svg" for session in sessions]
os.makedirs("results/hists", exist_ok=True)
for input, output in zip(inputs, outputs):
    df = pd.read_hdf(input, '/')
    fig = df["Variable"].plot.hist()
    fig.savefig(output)

# Make Figure1
inputs =  [f"/data/processed/data_{session}.h5" for session in sessions]
output =  "results/figure1.svg"
os.makedirs("results", exist_ok=True)
df_all = pd.concat([pd.read_hdf(input, '/') for input in inputs])
df_all.plot(x='Time', y='Happiness').savefig(output)
```

get_results.py

```
sessions = [1, 2, 3]

rule process_data:
    input:  "/data/raw/data_{session}.csv"
    output: "/data/processed/data_{session}.h5"
    run:
        df = pd.read_csv(input)
        df2 = do_process(df)
        df2.to_hdf(output, '/')

rule build_histograms:
    input:  "/data/processed/data_{session}.h5"
    output: "results/hists/hist_{session}.svg"
    run:
        df = pd.read_hdf(input, '/')
        fig = df["Variable"].plot.hist()
        fig.savefig(output)

rule figure1:
    input: expand("/data/processed/data_{session}.h5", session=sessions)
    output: "results/figure1.svg"
    run:
        df_all = pd.concat([pd.read_hdf(input, '/') for input in inputs])
        df_all.plot(x='Time', y='Happiness').savefig(output)
```

Snakefile

snakemake

```
pip install snakemake
snakemake figure1
```

```
sessions = [1, 2, 3]

rule process_data:
    input:  "/data/raw/data_{session}.csv"
    output: "data/processed/data_{session}.h5"
    script:  “scripts/process_data.py”

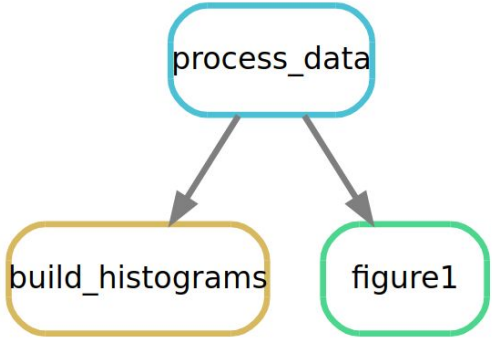
rule build_histograms:
    input:  "/data/processed/data_{session}.h5"
    output: "results/hists/hist_{session}.svg"
    script:  “scripts/plot_histogram.py”

rule figure1:
    input: expand("/data/processed/data_{session}.h5", session=sessions)
    output: "results/figure1.svg"
    script:  “scripts/plot_figure1.py”
```

scripts/process_data.py

```
import pandas as pd

filename = snakemake.input[0]
df = pd.read_csv(filename)
df2 = do_process(df)
df2.write_csv(snakemake.output[0])
```



- MyProject
  - README.md
  - requirements.txt
  - Singularity
  - myproject.simg
  - .gitignore
  - **Snakefile**
  - scripts
    - process_data.py
    - plot_histogram.py
    - plot_figure1.py
  - results
- /data
  - raw
    - data_1.csv
    - data_2.csv

My Project

This project makes histograms from my data!

Before You Begin

1. install singularity

Running the Analysis Script

Run the Singularity App, connecting its /data folder to the folder you downloaded the data to.

```
singularity run myproject.simg
```

```
%runscript
  snakemake figure1
```

Of Courses, It Can Get Much More Complex!

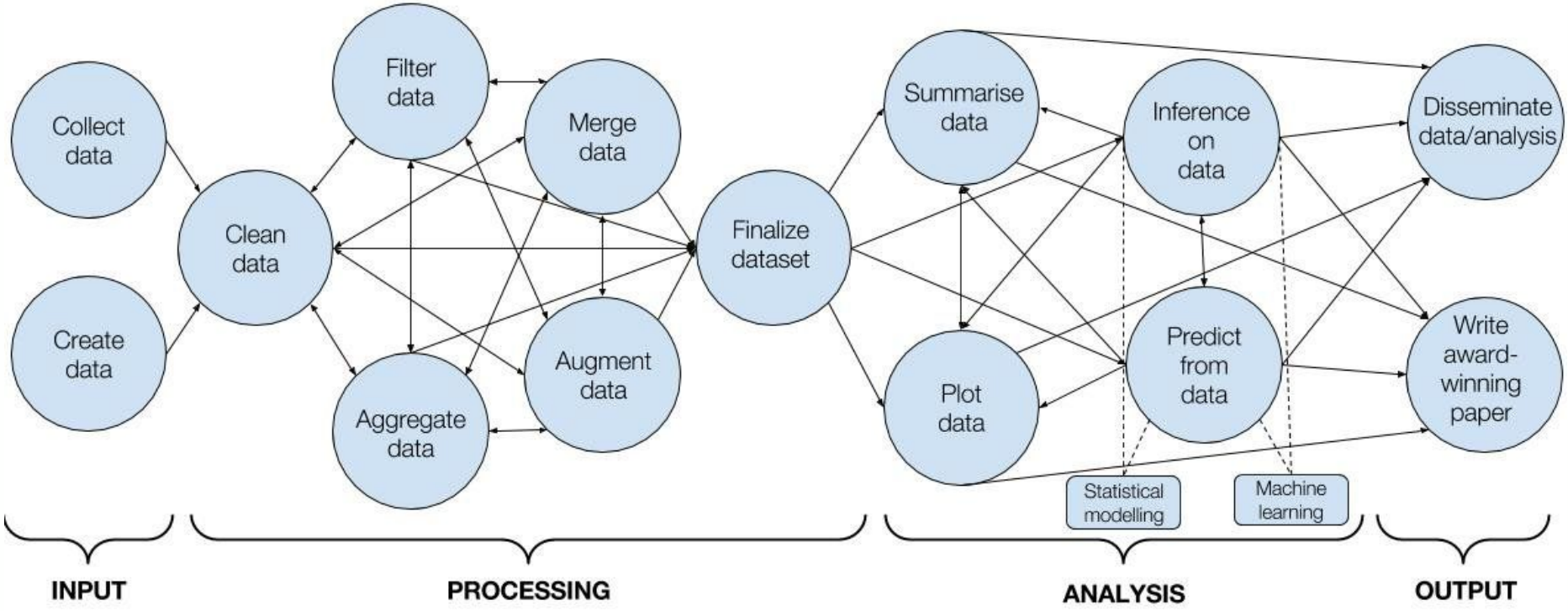


Image: "Data Science as a DAG" by Soubhik Barari

```
def task_dot():
    return {
        'file_dep': ['requests.models.deps'],
        'targets': ['requests.models.dot'],
        'actions': [module_to_dot],
    }
```

```
rule process_data:
    input:  "data/raw/data_{session}.csv"
    output: "data/processed/data_{session}.h5"
    script:  process_data.py
```

```
dag = DAG('tutorial')

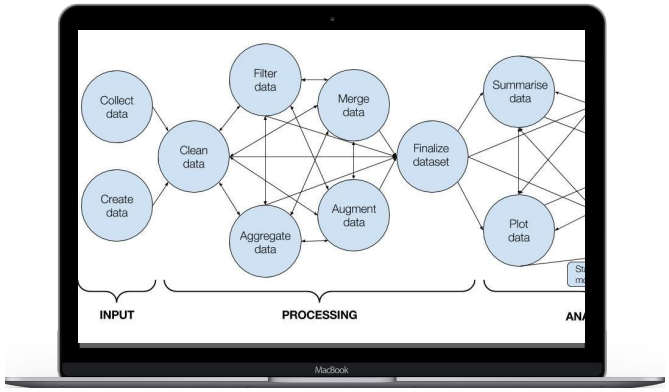
t1 = BashOperator(ttask_id='print_date',
    bash_command='date',  dag=dag)

t2 = BashOperator(task_id='sleep',
    bash_command='sleep 5', dag=dag)

t2.set_upstream(t1)
```

Because Workflow Managers Build Task Graphs, They Can Do:

- Partial Processing
- Parallel Processing
- Multi-Language Gluing
- Auto-Retries
- Remote Data Connections
- Workflow Monitoring
- Workflow Scheduling
- ...and much more!



SCIENTIFIC FILESYSTEM

```
%apphelp figure1
  Figure 1 from my paper,
showing how time spent with
Python increases your
happiness!

%apprun figure1
  snakemake figure1

%appinstall notebook
  pip install jupyterlab

%appinstall notebook
  cd notebooks
  jupyter lab

%apprun overview
  snakemake all -n --rulegraph
```

My Project

This project makes histograms from my data!

Before You Begin

1. install singularity

Exploring the Container

```
singularity myproject.simg help
singularity myproject.simg apps
```

Get Figure 1

```
singularity help myproject.simg  --app figure1
singularity run myproject.simg  --app figure1
```

Get a Graphical Overview of the Pipeline

```
singularity run myproject.simg  --app overview
```

Interact with the Analysis Notebooks

```
singularity run myproject.simg  --app notebook
```

Building Smooth, Forward Flow

The First Way

Readme Files as the human entry and first goal statement.

Version Control Systems to continually work towards goals

Package Managers to download versioned software and reduce version regressions

Environment Managers for isolating the Python interpreter

Container Systems for isolating everything.

Workflow Managers to break pipelines into smaller steps and separate out file overhead.

Getting Quick Feedback

The Second Way

Automated Testing to get feedback on the code's functionality
(PyTest, Hypothesis, Pytest-BDD)

Test-Driven Development to set goals, prioritize work, and maintain testable code.

Continuous Integration to ensure constant testing off the developer's machine.
(Travis CI, Jenkins, Circle CI)

Pair Programming to get extremely rapid feedback.

Maintaining an Environment of Continuous Improvement

The Third Way

Opportunistic Refactoring

Pair Programming

Chaos Monkey Testing

Blameless PostMortems

Opportunistic Testing

A Final Tool: The Jupyter Notebook and Binder

jupyter Lorenz Differential Equations (autosaved)

File Edit View Insert Cell Kernel Help Python 3

Code Cell Toolbar: None

Exploring the Lorenz System

In this Notebook we explore the Lorenz system of differential equations:

$$\dot{x} = \sigma(y - x)$$
$$\dot{y} = \rho x - y - xz$$
$$\dot{z} = -\beta z + xy$$

This is one of the classic systems in non-linear differential equations. It exhibits a range of complex behaviors as the parameters (σ, β, ρ) are varied, including what are known as *chaotic solutions*. The system was originally developed as a simplified mathematical model for atmospheric convection in 1963.

In [7]:
```
interact(Lorenz, N=fixed(10), angle=(0.,360.),
         σ=(0.0,50.0),β=(0.,5), ρ=(0.0,50.0))
```

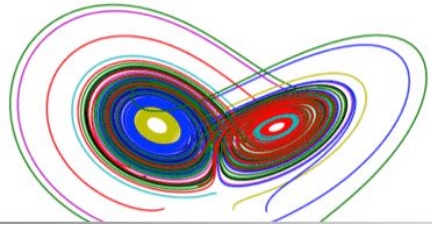
angle 308.2

max_time 12

σ 10

β 2.6

ρ 28



jupyter Welcome to P

File Edit View Insert Cell

jupyter

Welcome to the

This Notebook Server wa

WARNING

Don't rely on this serv

Your server is hosted thar

Run some Python

To run the code below:

1. Click on the cell to se
2. Press SHIFT+ENTER

A full tutorial for using the

In [ ]: %matplotlib inline

```
import pandas as pd
import numpy as np
import matplotlib
```

binder (beta)

Turn a GitHub repo into a collection of interactive notebooks

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

d launch a repository

o or URL

ithub.com/marius311/CMBLensing.jl

tag, or commit

Path to a notebook file (optional)

nb/jointmax_128example/jointmax_128example.ip

File

launch

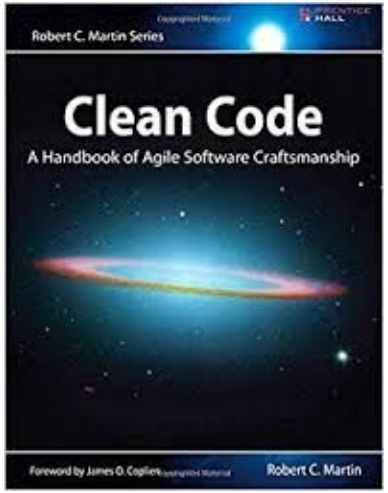
share this URL:

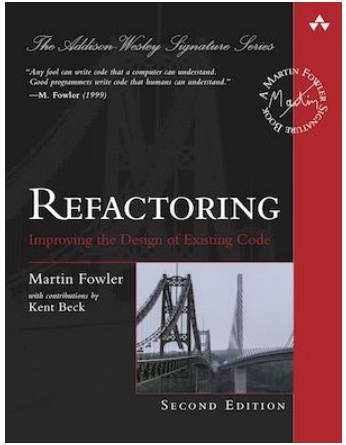
/v2/gh/marius311/CMBLensing.jl/master?filepath=nb%2Fjointmax_128example%2Fjointmax_128example.ipynb

paste into your README to get badge: launch binder

Building

Books Recommendations: DevOps, Coding Practices, and Lean Management





Thank you for your attention!

nickdelgrosso

nick-del-grosso