

Building a Reproducible Data Analysis Pipeline

MATTHEW DOUGLAS, COMPUTATIONAL BIOLOGIST

Canada's Michael Smith Genome Sciences Centre at BC Cancer

mdouglas@bcgsc.ca



WestGrid Summer School - June 18, 2020

Outline

Background

Key Concepts (theory and code examples)

- Sharing code online
- Writing good documentation
- Code dependencies (Python)
- Reproducing runtime environments (Docker)

Conclusion



My Background

Computational Biologist (Genome Sciences Centre)

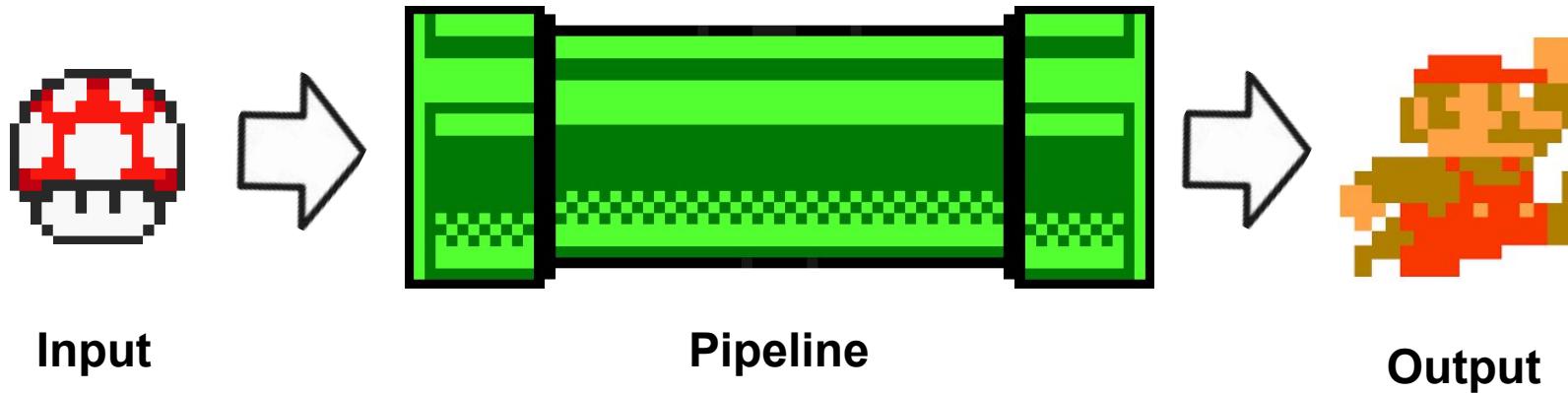
- Developing pipelines for tumour characterization
- Work mostly Python, R, CWL, Docker



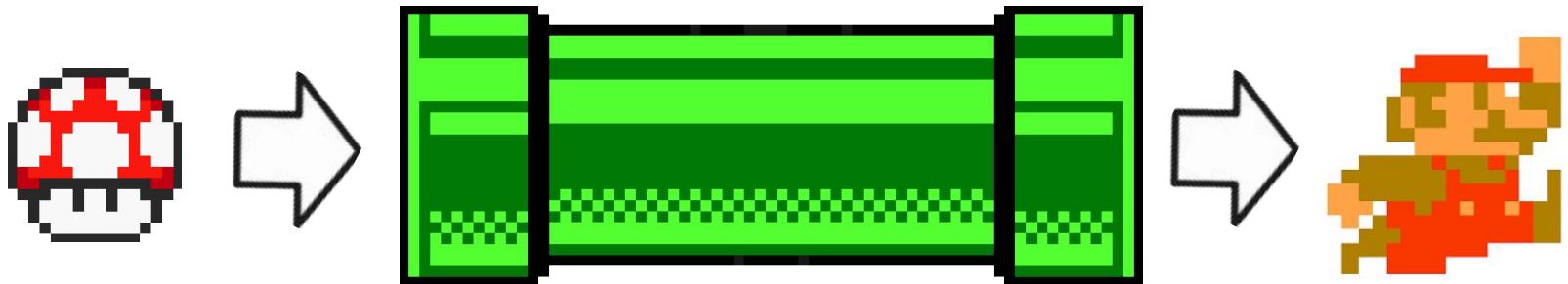
Background in Bio

- B.Sc. Microbiology (University of Victoria)
- M.Sc. Bioinformatics (Simon Fraser University)

A Data Analysis Pipeline



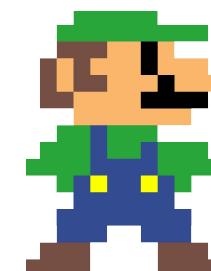
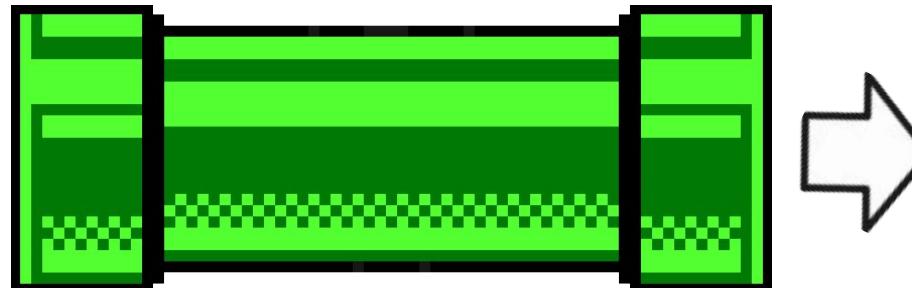
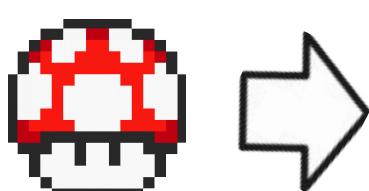
A Data Analysis Pipeline



Input

Pipeline

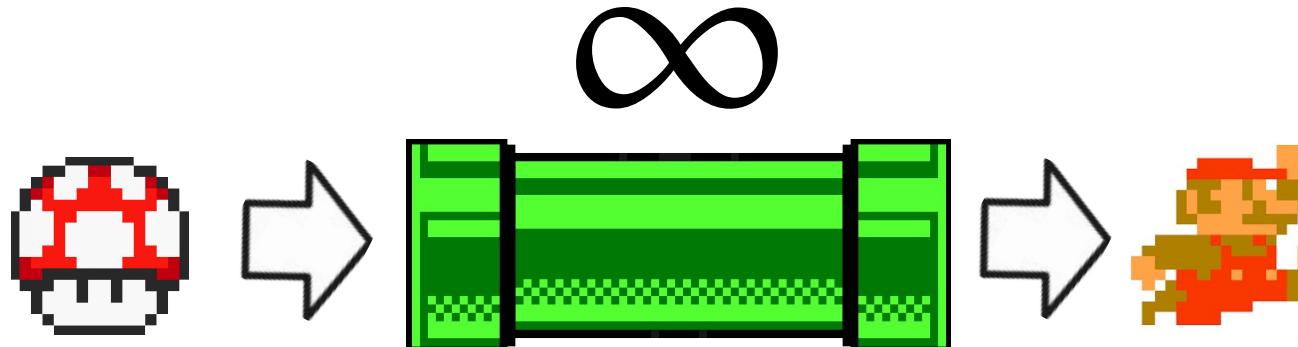
Output



Reproducibility vs. Replicability

Reproducibility

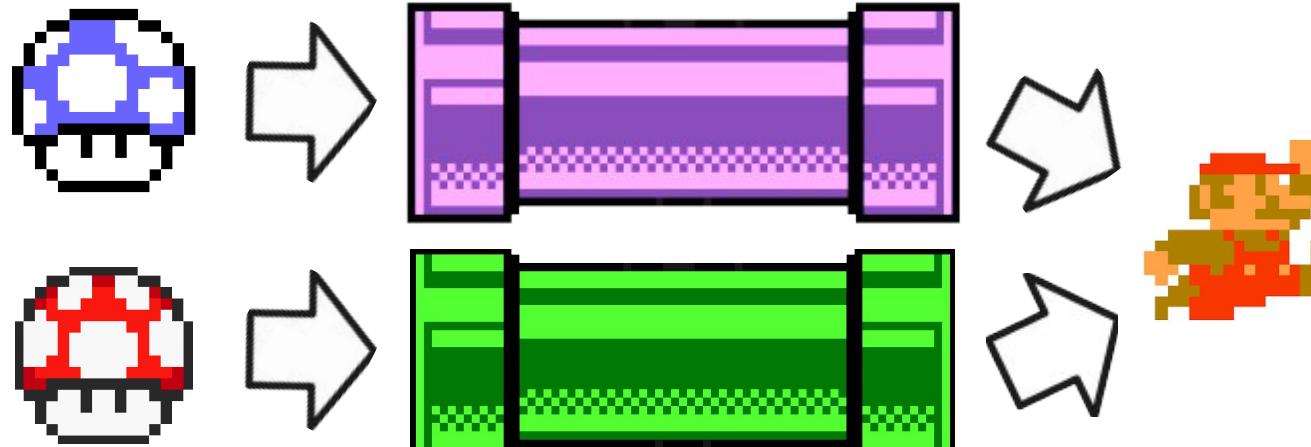
obtaining consistent results using the same input data; computational steps/methods/code; and conditions of analysis.



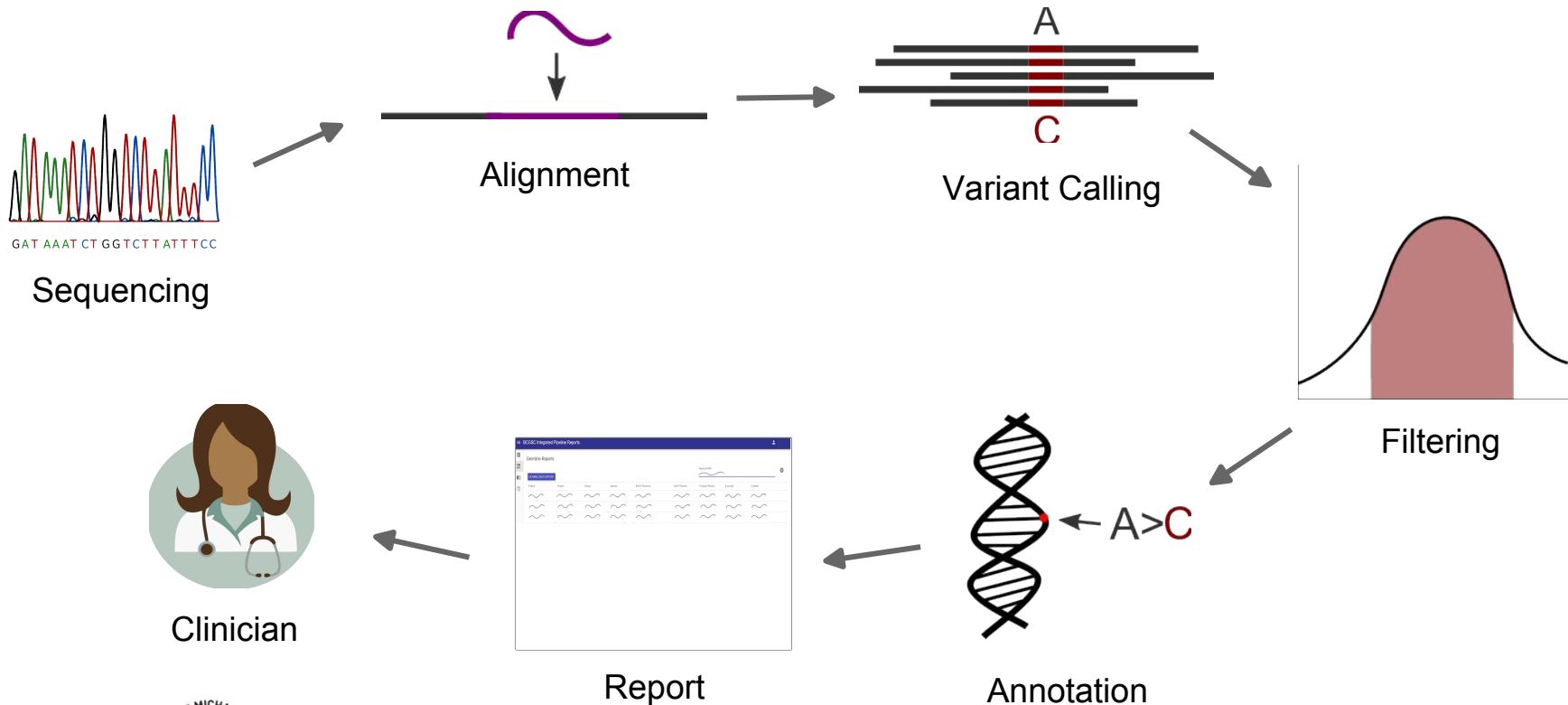
Reproducibility vs. Replicability

Replicability (Repeatability)

obtaining consistent results using different input data; computational steps/methods/code; or conditions of analysis.



Example Pipeline

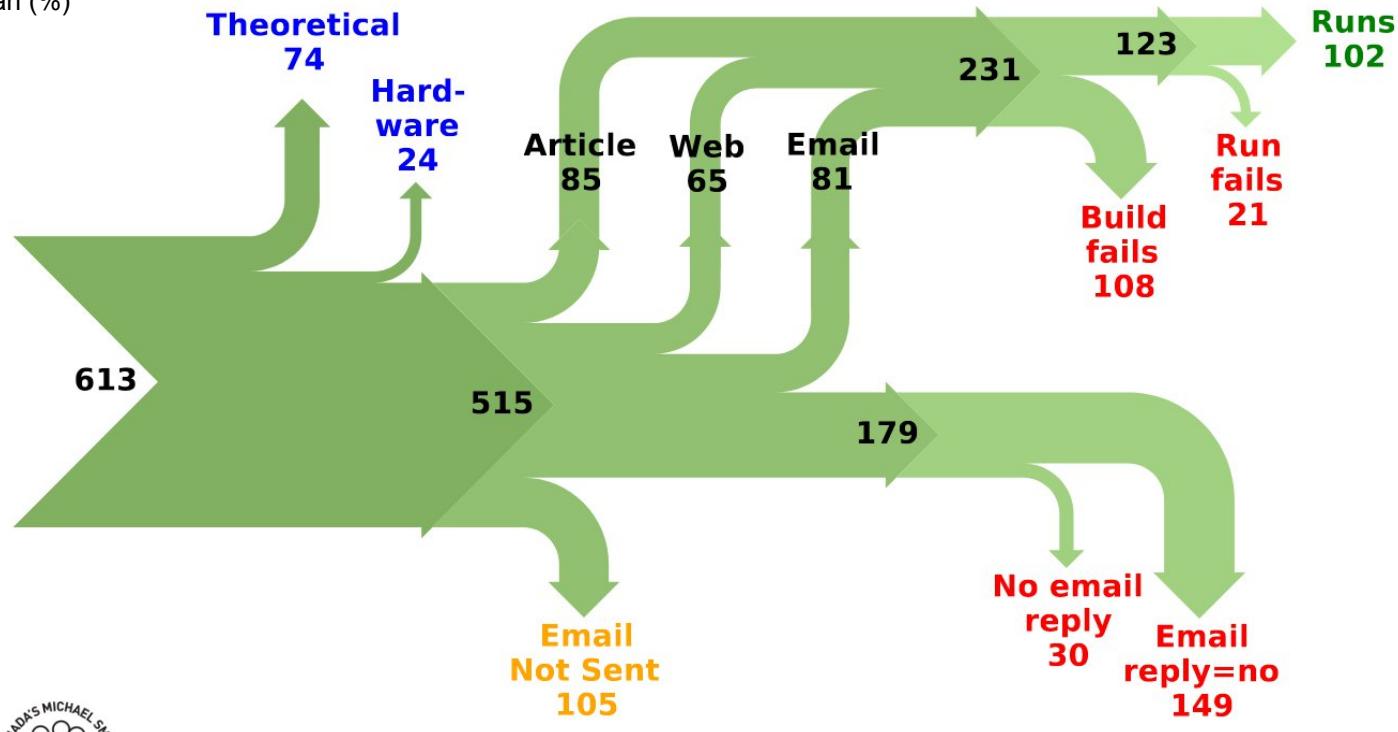


Reproducibility in Data Science

Blue = Excluded (e.g. hardware limitation)

Orange/Red = Could not get or build code

Green = Code ran (%)



C. Collberg, et al. Measuring Reproducibility in Computer Systems Research. (2014)

Reproducibility in Data Science

Most pipelines are built based on good science.

A failure to reproduce a result is often due to
how the pipeline is packaged/shared.

Reproducible Pipeline Checklist

- Code is available
- Good documentation
- Dependencies are listed
- Runtime environment can be reproduced

Code Availability

The first step in reproducing a bioinformatics result is to obtain the code used to produce the result.

Code Comes in Two Forms

Source Code



Binary



Code Comes in Two Forms

Source Code
(Human readable)



Binary
(Machine readable)



Code Comes in Two Forms

Source Code

```
block = TranslationBlock()
last = i[0][0]
count += 1
for pos, kind in sort_by_type(i[1:]):
    if kind == 0: # if start codon
        block.s_sites.append(pos)
    elif kind == 1: # left splice site
        block.r_sites.append(pos)
    elif kind == 2: # right splice site
        block.l_sites.append(pos)
    elif kind == 3: # if stop codons
        block.start = last + 1
        block.end = pos
        if block.has_splice_site():
            blocks_f[chrom][frame].append(block)
        else:
            empty_blocks_f[chrom][frame].append(block)
            empty_blk += 1
```

Human readable

- Verify
- Improve
- Modify

Code Comes in Two Forms

Machine readable

- Not human readable
- Cannot be verified

("Black box")

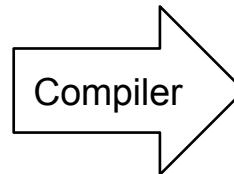
Binary

000000000	01 00 FF FF 00 00 00 00 00 00 00 00 00 00 00 40 00 CC 80@..
000000100	0C 00 00 00 00 00 26 01 8F 00 00 00 00 00 53 00&....S.
000000200	65 00 6C 00 65 00 63 00 74 00 20 00 52 00 75 00	e.l.e.c.t. R.u.
000000300	6C 00 65 00 00 00 08 00 00 00 00 01 4D 00 53 00	l.e.....M.S.
000000400	20 00 53 00 68 00 65 00 6C 00 6C 00 20 00 44 00	S.h.e.l.l.D.
000000500	6C 00 67 00 00 00 00 00 00 00 00 00 02 00 00	l.g.....
000000600	03 01 A1 50 53 00 3A 00 C3 00 36 00 32 25 00 00	...PS:...6.2%..
000000700	FF FF 83 00 00 00 00 00 00 00 00 00 00 00 00 00 00	...P.V.A.J&..
000000800	03 00 01 50 0E 00 56 00 41 00 0A 00 4A 26 00 00	...&A.p.p.l.y.
000000900	FF FF 80 00 26 00 41 00 70 00 70 00 6C 00 79 00	...t.o.a.l.l..P
000000a00	20 00 74 00 6F 00 20 00 61 00 6C 00 6C 00 00 00
000000b00	00 00 00 00 00 00 00 00 00 00 00 01 00 01 50
000000c00	7E 00 7D 00 32 00 0E 00 01 00 00 FF FF 80 00	~}2.....O.K..
000000d00	4F 00 4B 00 00 00 00 00 00 00 00 00 00 00 00 00	...P.}2.....
000000e00	00 00 01 50 B4 00 7D 00 32 00 0E 00 02 00 00	...C.a.n.c.e.l..P
000000f00	FF FF 80 00 43 00 61 00 6E 00 63 00 65 00 6C 00
000001000	00 00 00 00 00 00 00 00 00 00 00 00 00 01 50
000001100	EA 00 7D 00 32 00 0E 00 09 00 00 00 FF FF 80 00	..}2.....P
000001200	26 00 48 00 65 00 6C 00 70 00 00 00 00 00 00 00	&H.e.l.p.....
000001300	00 00 00 00 00 00 00 00 80 08 81 50 0E 00 3A 00P..
000001400	3B 00 0E 00 2F 25 00 00 FF FF 81 00 00 00 00%.....P.0.
000001500	00 00 00 00 00 00 00 00 00 00 02 50 0E 00 30 00
000001600	1E 00 08 00 EE 25 00 00 FF FF 82 00 46 00 69 00%.....F.i.
000001700	6C 00 65 00 20 00 54 00 79 00 70 00 65 00 00	l.e.T.y.p.e..P
000001800	00 00 00 00 00 00 00 00 00 00 00 00 00 00 02 50
000001900	54 00 30 00 2C 00 08 00 EF 25 00 00 FF FF 82 00	T.O.,%.....
000001a00	50 00 61 00 72 00 73 00 69 00 6E 00 67 00 20 00	P.a.r.s.i.n.g..
000001b00	52 00 75 00 6C 00 65 00 73 00 00 00 00 00 00 00	R.u.l.e.s.....
000001c00	00 00 00 00 00 00 00 00 07 00 00 50 06 00 07 00F..
000001d00	1A 01 71 00 ED 25 00 00 FF FF 80 00 00 00 00	..q.%.....P..
000001e00	00 00 00 00 00 00 00 00 00 02 50 0E 00 11 00P..
000001f00	3E 00 08 00 EC 25 00 00 FF FF 82 00 53 00 65 00	>....%.....S.e.
000002000	6C 00 65 00 63 00 74 00 20 00 52 00 75 00 6C 00	l.e.c.t.R.u.l..



Code Comes in Two Forms

```
block = TranslationBlock()  
last = i[0][0]  
count += 1  
for pos, kind in sort_by_type(i[1]):  
    if kind == 0: # if start codon  
        block.s_sites.append(pos)
```



00000000	01 00 FF FF 00 00 00 00 00 00 00 00 00 00 00 00 00 CC 80
00000010	0C 00 00 00 00 00 26 01 8F 00 00 00 00 00 00 00 00 53 00
00000020	65 00 6C 00 65 00 63 00 74 00 20 00 52 00 75 00
00000030	6C 00 65 00 00 00 08 00 00 00 00 01 4D 00 53 00
00000040	20 00 53 00 68 00 65 00 6C 00 6C 00 20 00 44 00
00000050	6C 00 67 00 00 00 00 00 00 00 00 00 00 00 02 00 00 00
00000060	03 01 A1 50 53 00 3A 00 C3 00 36 00 32 25 00 00
00000070	FF FF 83 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000080	03 00 01 50 0E 00 56 00 41 00 0A 00 4A 26 00 00
00000090	FF FF 80 00 26 00 41 00 70 00 70 00 6C 00 79 00
000000a0	20 00 74 00 6F 00 20 00 61 00 6C 00 6C 00 00 00 00 00
000000b0	00 00 00 00 00 00 00 00 00 00 00 01 00 01 50

Source code

- Need to compile before use*
- May require additional code libraries

*except 'interpreted languages' like Python

Binary

- "Ready to run"
- No need to compile before use
- However, it is limited to the operating system it was compiled for



Code Comes in Two Forms

What form of code should I share with others?



WHY NOT BOTH?

memegenerator.net

Code Comes in Two Forms

Source Code

- ✓ Can be independently verified
- ✓ Can be collaborated on
- ✗ Less user friendly

Binary

- ✓ Convenient
- ✗ "Black box"
*what is the negative of a black box?

Sharing Code

How do I share my code with others?

Obtaining Unpublished Code

From: Christian Collberg <ccollberg@gmail.com>
To: first-or-corresponding-author
Cc: remaining-authors
Subject: Your conference-name paper

Dear Dr. first-or-corresponding-author,

I've been looking at your conference-name paper
paper-title

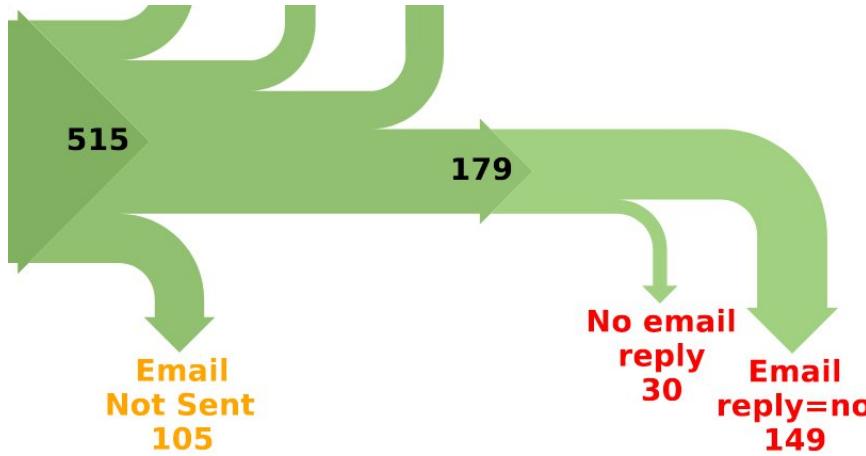
and would like to try out the implementation. However,
I haven't been able to find it online. Would you please
let me know how I can obtain the source code so that I
can try to build and run it?

Thank you very much for your help!

Christian Collberg
ccollberg@gmail.com



Obtaining Unpublished Code



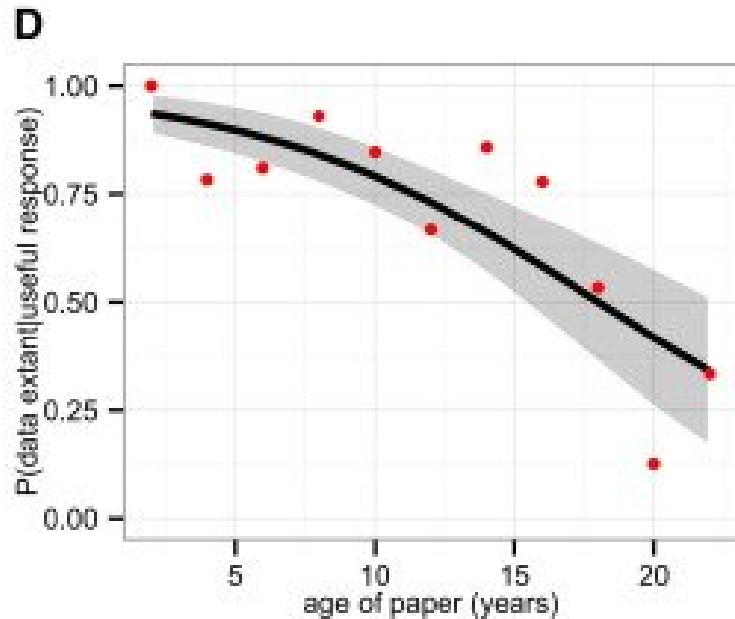
C. Collberg, et al. Measuring Reproducibility in Computer Systems Research. (2014)

- Data gets lost
- Programmer left
- Will be released soon
- No intention to share
- Proprietary
- Will not work outside of <very specific system>
- **Broken email addresses**

Obtaining Unpublished Code

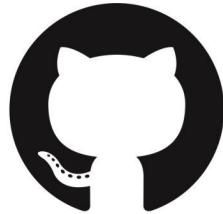
Self-hosted code becomes less available over time.

“Overall, we only received 19.5% [101/512] of the requested data sets...”
(Vines, et al. 2014)



Vines, T.H., et al. The Availability of Research Data Declines Rapidly with Article Age. *Current Biology* 24, 94-97 (2014).

Public Code Repositories - Version Controlled



GitHub



Bitbucket



GitLab

- Git repositories (version control)
- Facilitates collaboration!
- Removes the burden of hosting
- Easy sharing

Public Code Repositories - Non-Version Controlled



SOURCEFORGE

- Hosting and sharing
- Not version controlled
- Not as easy to collaborate

Code

Issues 22

Pull requests 1

Actions

Projects 0

Wiki

Security 0

Insights

Grit is no longer maintained. Check out libgit2/rugged. Grit gives you object oriented read/write access to Git repositories via Ruby.
<http://grit.rubyforge.org/>

516 commits

3 branches

0 packages

15 releases

43 contributors

MIT

Branch: master ▾

New pull request

Create new file

Upload files

Find file

Clone or download ▾



Brandon Keepers Merge pull request #183 from bkeepers/unmaintained ...

Latest commit 5608567 on Feb 3, 2014

Source
Code

examples

Update from GitHub.

10 years ago

lib

Fix Tempfile usage under ruby 1.8.7

7 years ago

test

remove tests for stuff @schacon removed

9 years ago

.gitignore

Added Repo#commit_deltas_from as a (fairly expensive and lazy) way of...

12 years ago

API.txt

added some simple write ops : add, remove, commit

12 years ago

History.txt

Release 2.5.0

8 years ago

LICENSE

convert readme to markdown

11 years ago

PURE_TODO

added some simple write ops : add, remove, commit

12 years ago

README.md

clarify status

7 years ago

Rakefile

Ruby 1.9 compatibility

10 years ago

benchmarks.rb

added some simple write ops : add, remove, commit

12 years ago

Download

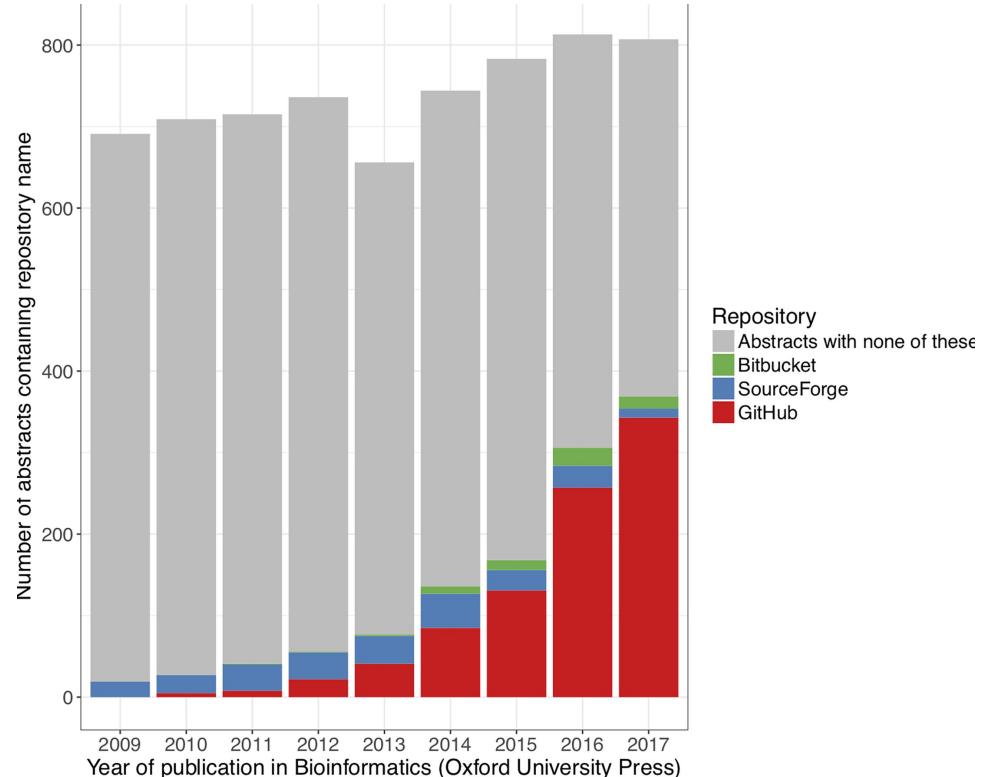


How Common is Publishing Code?

Public repositories are more widespread in bioinformatics.

Exceptions:

- Not allowed to publish (licensing, etc.)
- Published on personal website



Publishing your code online...

- Makes it easy to share your code
- Improves the lifespan of your code
- Facilitates collaboration
- Sometimes required when publishing

Reproducible Pipeline Checklist

- Code is available
- Good documentation
- Dependencies are listed
- Runtime environment can be reproduced

The Value of Good Documentation

You have an awesome pipeline.

How do I use it?



README

A **README** is a text file that explains your pipeline.

Who is it for?

- Others - Explains to others how to install and run
- You - Reflect on how the pipeline is structured

 README.md

This Is a Bad README

© 2020 GitHub, Inc. [Terms](#) [Privacy](#) [Security](#) [Status](#) [Help](#)



[Contact GitHub](#) [Pricing](#) [API](#) [Training](#) [Blog](#) [About](#)

Name  **GATK 4**

Description  This repository contains the next generation of the Genome Analysis Toolkit (GATK). The contents of this repository are 100% open source and released under the BSD 3-Clause license (see [LICENSE.TXT](#)).
GATK4 aims to bring together well-established tools from the [GATK](#) and [Picard](#) codebases under a streamlined framework, and to enable selected tools to be run in a massively parallel way on local clusters or in the cloud using [Apache Spark](#). It also contains many newly developed tools not present in earlier releases of the toolkit.

Instruction  **Table of Contents**

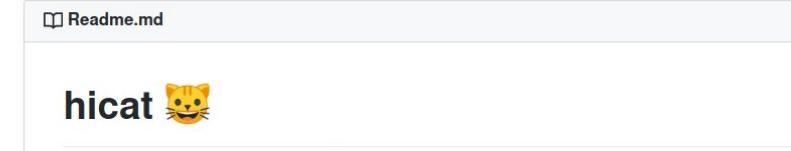
- [Requirements](#)
- [Quick Start Guide](#)
- [Downloading GATK4](#)
- [Building GATK4](#)
- [Running GATK4](#)
 - [Passing JVM options to gatk](#)
 - [Passing a configuration file to gatk](#)
 - [Running GATK4 with inputs on Google Cloud Storage](#)
 - [Running GATK4 Spark tools on a Spark cluster](#)
 - [Running GATK4 Spark tools on Google Cloud Dataproc](#)
 - [Using R to generate plots](#)
 - [GATK Tab Completion for Bash](#)

Key Parts of a README

Name

- Makes your project easier to find
- Lets you be creative
- When in doubt: Acronyms

**Tutorial
On
Reproducible
Pipelines**



Source: <https://github.com/isonic1/hicat>



Key Parts of a README

Introduction

- A brief description of what your project does



hicat 😺

```
o hicat
```

`cat` with syntax highlighting. The language is auto-detected through the file extension.

Source: <https://github.com/isonic1/hicat>

Key Parts of a README

How to Install

- All commands
- All dependencies

Installation

```
$ npm install -g hicat
```

Source: <https://github.com/isonic1/hicat>



Key Parts of a README

How to Use

- Examples
- References explaining commands (if helpful)

```
cat with syntax highlighting. The language is auto-detected through the file extension.
```

```
hicat index.js
```

```
Pipe something to hicat . The language will be inferred from the contents.
```

```
curl http://site.com | hicat
```

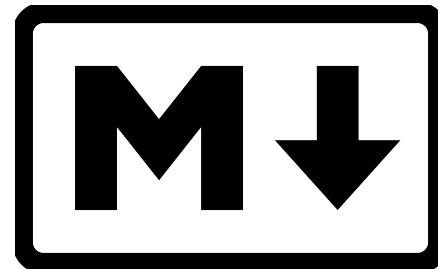
```
If hicat fails to detect a language, specify it using -l LANG .
```

```
curl http://site.com | hicat -l xml
```

MarkDown

READMEs are typically written in **MarkDown**

- Lightweight language for formatting text
- Files have a '.md' (**MarkDown**) extension
- Commonly seen in *Github*



MarkDown logo

i README.md x

1 |

□ □ ...

□ Preview README.md x

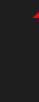
i README.md x

```
1 This is a collection of code used as examples in my tutorial  
"Building a Reproducible Data Analysis Pipeline".  
2
```



i Preview README.md x

This is a collection of code used as examples in my tutorial "Building a Reproducible Data Analysis Pipeline".



README.md ×

...

1 # Reproducible Pipeline Tutorial

2 This is a collection of code used as examples in my tutorial
"Building a Reproducible Data Analysis Pipeline".

4

Preview README.md ×

Reproducible Pipeline Tutorial

This is a collection of code used as examples in my tutorial "Building a Reproducible Data Analysis Pipeline".

1 # Reproducible Pipeline Tutorial
2
3 This is a collection of code used as examples in my tutorial
"Building a Reproducible Data Analysis Pipeline".
4
5 ## Dependencies
6



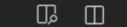
Preview README.md ×

Reproducible Pipeline Tutorial

This is a collection of code used as examples in my tutorial "Building a Reproducible Data Analysis Pipeline".

Dependencies

1 # Reproducible Pipeline Tutorial
2
3 This is a collection of code used as examples in my tutorial
"Building a Reproducible Data Analysis Pipeline".
4
5 ## Dependencies
6
7 Running the example script requires `python3`.
8



Preview README.md ×

Reproducible Pipeline Tutorial

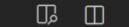
This is a collection of code used as examples in my tutorial "Building a Reproducible Data Analysis Pipeline".

Dependencies

Running the example script requires python3.



1 # Reproducible Pipeline Tutorial
2
3 This is a collection of code used as examples in my tutorial
"Building a Reproducible Data Analysis Pipeline".
4
5 ## Dependencies
6
7 Running the example script requires `python3`.
8
9 All the necessary Python dependencies can be installed by running:
10 ````
11 pip install -r requirements.txt
12 ````
13



Preview README.md ×

Reproducible Pipeline Tutorial

This is a collection of code used as examples in my tutorial "Building a Reproducible Data Analysis Pipeline".

Dependencies

Running the example script requires `python3`.

All the necessary Python dependencies can be installed by running:

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



1 # Reproducible Pipeline Tutorial
2
3 This is a collection of code used as examples in my tutorial
"Building a Reproducible Data Analysis Pipeline".
4
5 ## Dependencies
6
7 Running the example script requires `python3`.
8
9 All the necessary Python dependencies can be installed by running:
10 ``
11 pip install -r requirements.txt
12 ``
13
14 The script can also be run inside a Docker container. Instructions on how to install Docker can be found [here](<https://docs.docker.com/engine/install/>).
15

Reproducible Pipeline Tutorial

This is a collection of code used as examples in my tutorial "Building a Reproducible Data Analysis Pipeline".

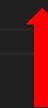
Dependencies

Running the example script requires `python3`.

All the necessary Python dependencies can be installed by running:

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

The script can also be run inside a Docker container. Instructions on how to install Docker can be found [here](#).



```
1 # Reproducible Pipeline Tutorial
2
3 This is a collection of code used as examples in my tutorial
4 "Building a Reproducible Data Analysis Pipeline".
5
6 ## Dependencies
7
8 Running the example script requires `python3`.
9
10 All the necessary Python dependencies can be installed by running
11 ```
12 pip install --r requirements.txt
13 ```
14 The script can also be run inside a Docker container. Instructions
15 on how to install Docker can be found [here](https://docs.docker.com/engine/install/).
16
17 ## Usage
18
19 Run the example script:
20 ```
21 python3 pipeline.py
22 ```
23 Run the example script inside a Docker container:
24 ```
25 docker run mattdoug604/reproducible_tutorial
26 ```
27 |
```

Reproducible Pipeline Tutorial

This is a collection of code used as examples in my tutorial "Building a Reproducible Data Analysis Pipeline".

Dependencias

Running the example script requires `python3`

All the necessary Python dependencies can be installed by running

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

The script can also be run inside a Docker container. Instructions on how to install Docker can be found [here](#).

Usage

Run the example script

python3 pipeline.py

Run the example script inside a Docker container

```
docker run mattdoug604/reproducible_tutorial
```

Summary - Documentation

Documentation is key when writing any sort of software - both for you and others trying to reproduce a result.

A README is a short text file:

- Typically written in MarkDown
- What your project does
- How to install it
- How to run it



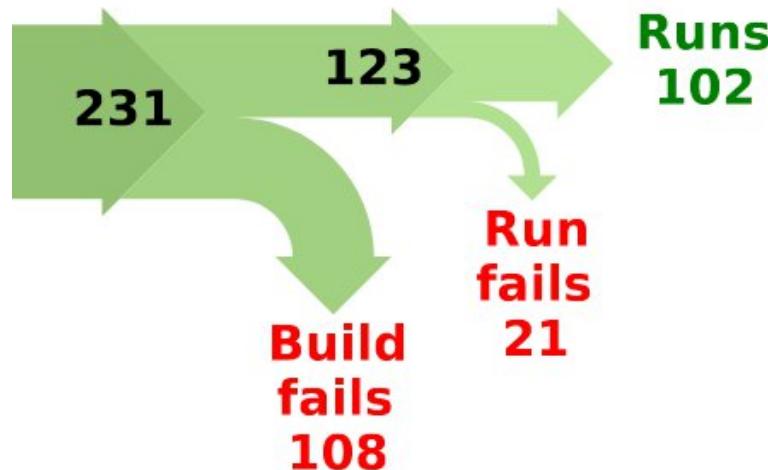
Reproducible Pipeline Checklist

-  Code is available
-  Good documentation
-  Dependencies are listed
-  Runtime environment can be reproduced

Why is it so Hard to Run Someone's Code?

Up to 50% of published code cannot be run (Collberg, *et al.* 2014).

The most common reason is that the build fails.

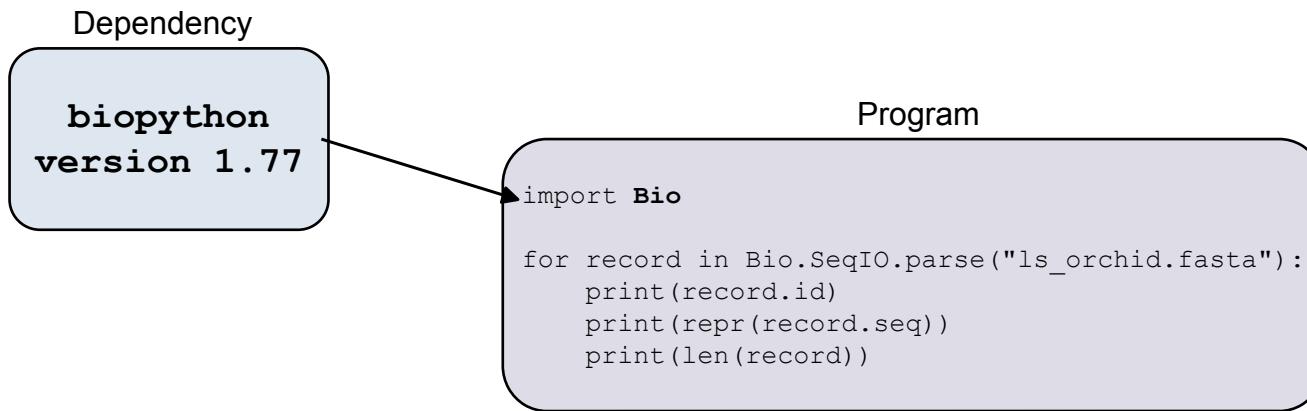


C. Collberg, *et al.* Measuring Reproducibility in Computer Systems Research. (2014)

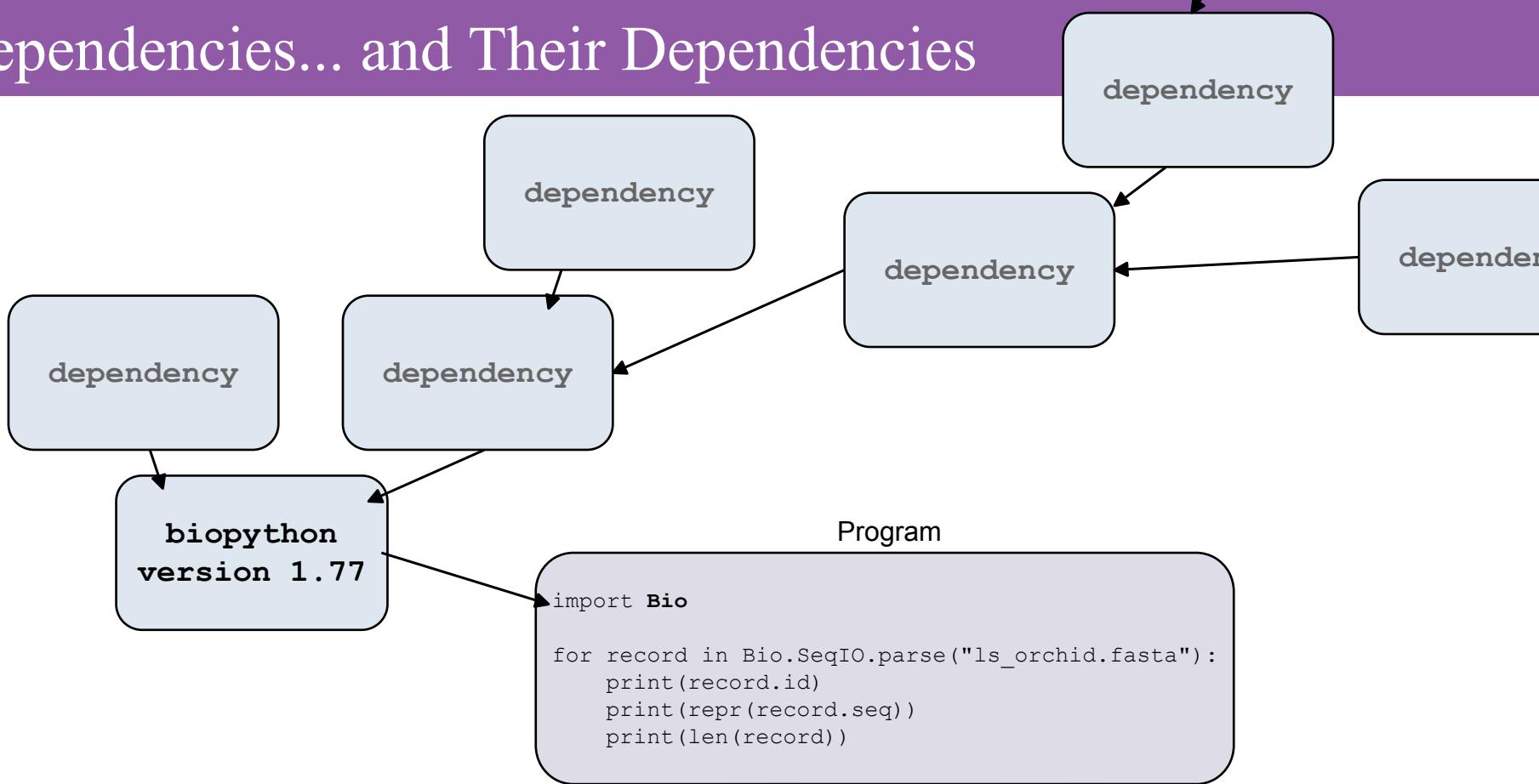
Dependencies...

Dependencies

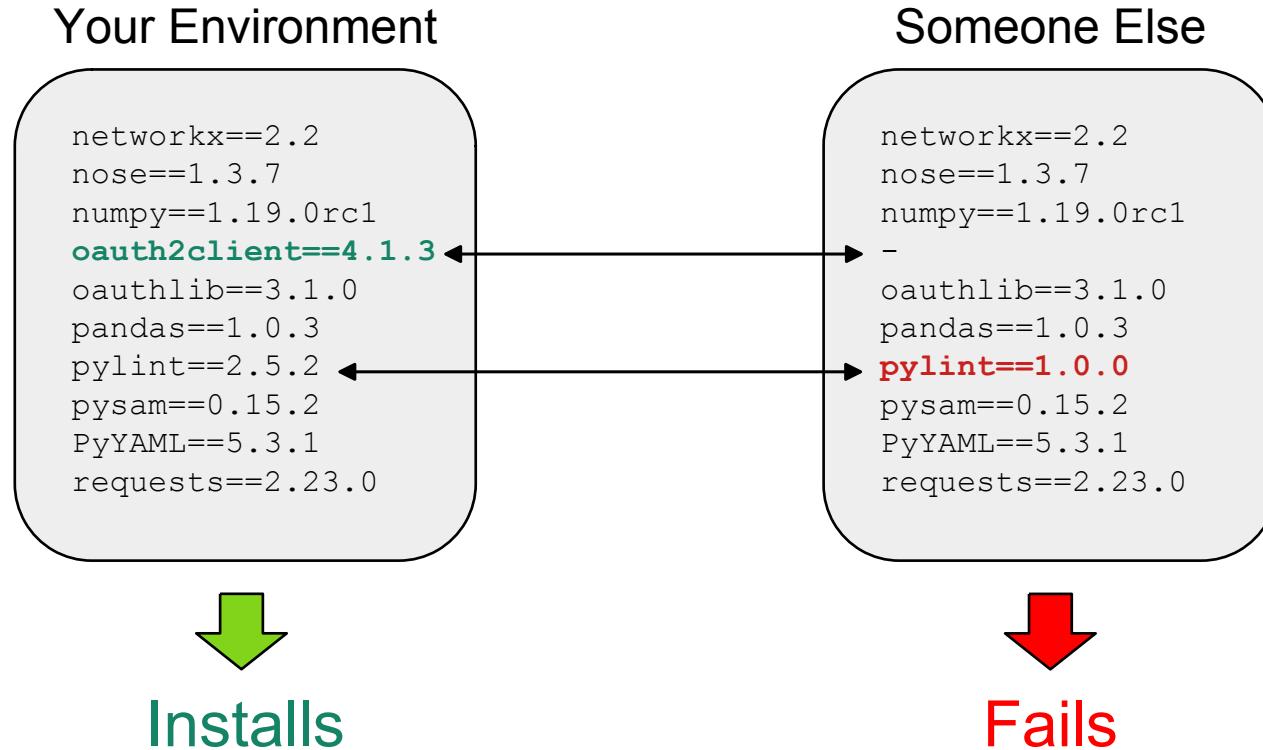
- Code that is required by a different piece of code
- Often, a specific version of a dependency is required



Dependencies... and Their Dependencies



Dependencies are the Main Cause of Failed Builds



Managing Python Dependencies

venv

- Python virtual environment

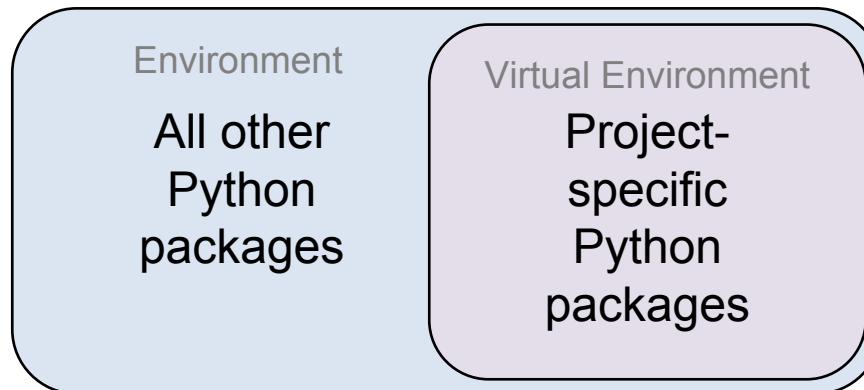
pip

- Python package manager



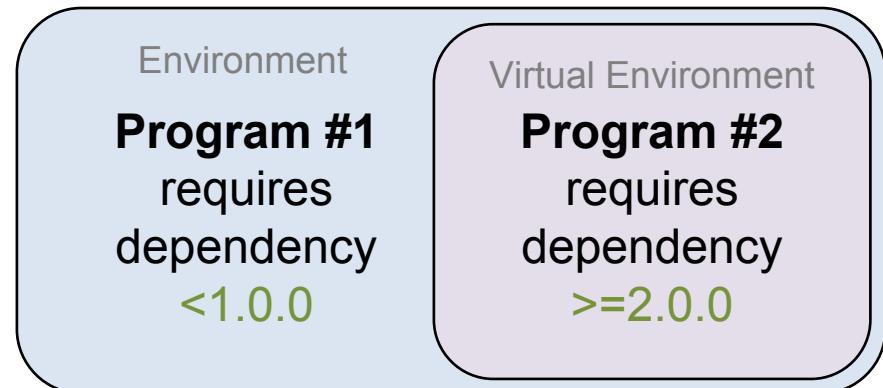
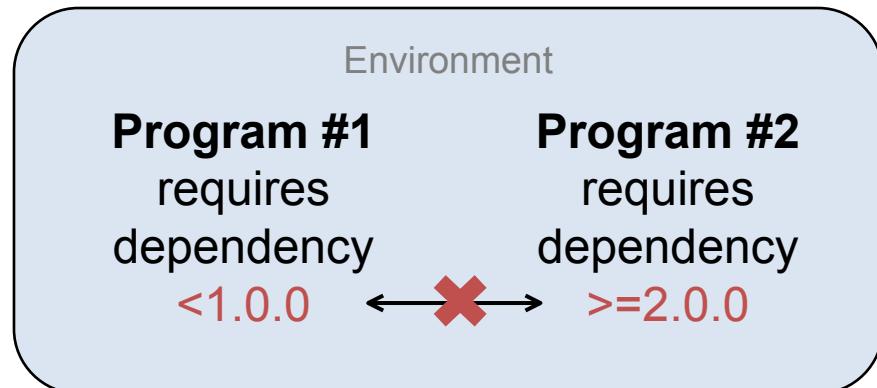
Python Virtual Environments

- Python environment specific to your project
- Libraries and scripts installed into it are isolated from other environments



Python Virtual Environments

Solves the problem of “this program uses v1.x.x of a package but that program uses v2.x.x”



Developing in a Python Virtual Environment

```
matt@matt:~$
```

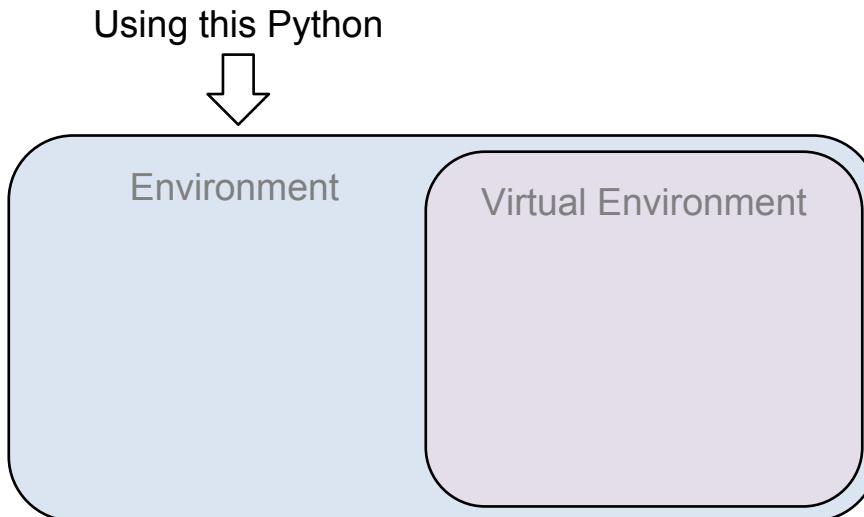
Using this Python



Environment

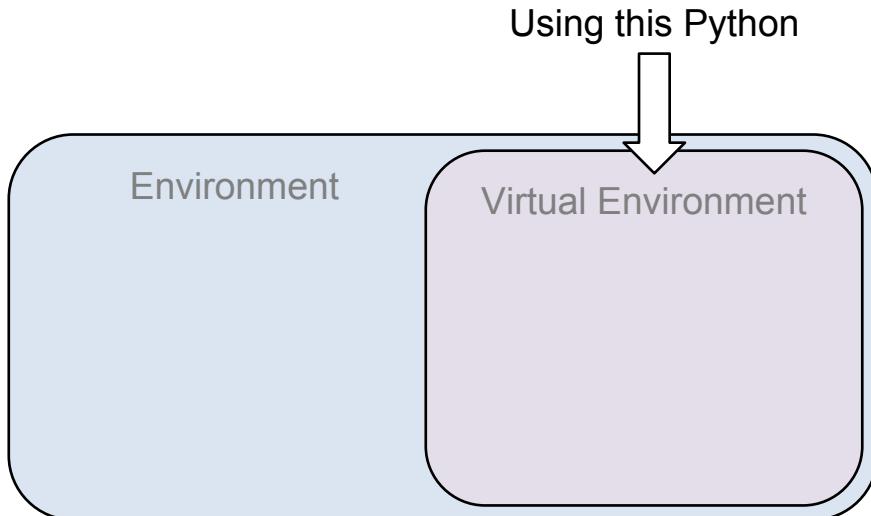
Developing in a Python Virtual Environment

```
matt@matt:~$ python3 -m venv my_venv
```



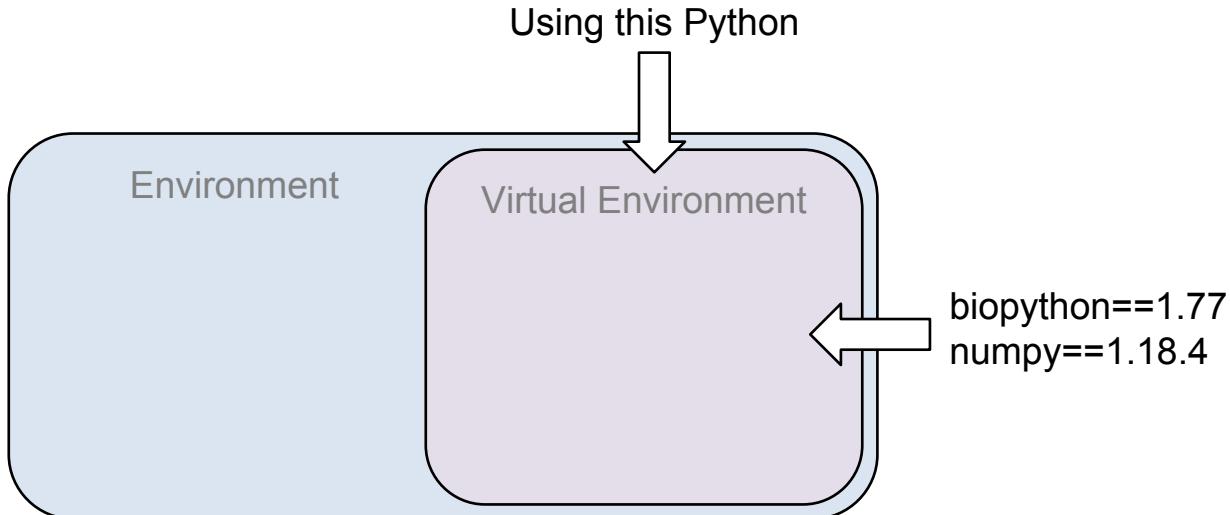
Developing in a Python Virtual Environment

```
matt@matt:~$ source my_venv/bin/activate
```



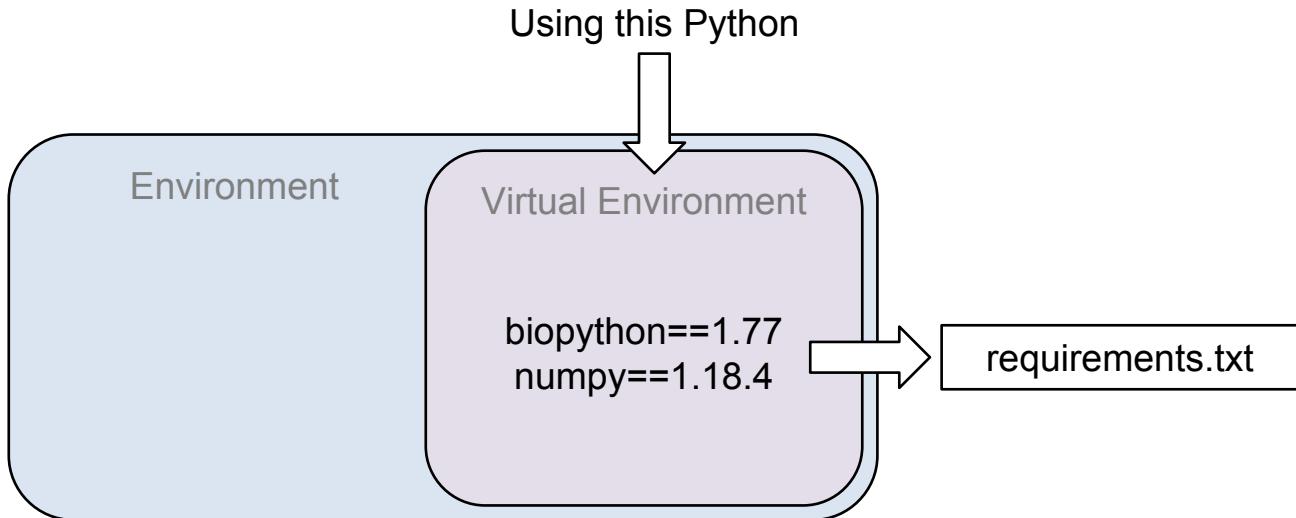
Developing in a Python Virtual Environment

```
(my_venv) matt@matt:~$ pip install biopython
```



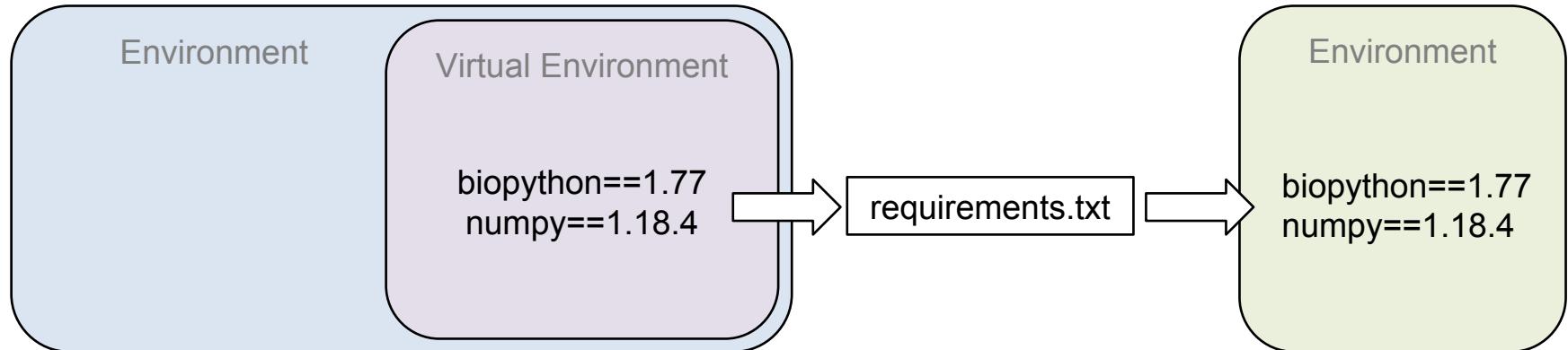
Developing in a Python Virtual Environment

```
(my_venv) matt@matt:~$ pip freeze > requirements.txt
```



Developing in a Python Virtual Environment

```
(my_venv) matt@matt:~$ pip install -r requirements.txt
```



Developing in a Python Virtual Environment

```
matt@matt:~$
```



Developing in a Python Virtual Environment

Step 1) Create the virtual environment with the 'venv' python package.

```
matt@matt:~$ python3 -m venv my_venv
```

Developing in a Python Virtual Environment

Step 2) Activate the virtual environment.

```
matt@matt:~$ python3 -m venv my_venv  
matt@matt:~$ source my_venv/bin/activate  
(my_venv) matt@matt:~$ pip freeze # nothing prints
```

Developing in a Python Virtual Environment

Step 3) Install Python dependencies.

```
matt@matt:~$ python3 -m venv my_venv
matt@matt:~$ source my_venv/bin/activate
(my_venv) matt@matt:~$ pip freeze  # nothing prints
(my_venv) matt@matt:~$ pip install biopython
```

Developing in a Python Virtual Environment

Step 4) List Python dependencies.

```
matt@matt:~$ python3 -m venv my_venv
matt@matt:~$ source my_venv/bin/activate
(my_venv) matt@matt:~$ pip freeze # nothing prints
(my_venv) matt@matt:~$ pip install biopython
(my_venv) matt@matt:~$ pip freeze > requirements.txt
(my_venv) matt@matt:~$ cat requirements.txt
biopython==1.77
numpy==1.18.4
```

Developing in a Python Virtual Environment

Python dependencies can be installed with 'pip install'.

```
matt@matt:~$ python3 -m venv my_venv
matt@matt:~$ source my_venv/bin/activate
(my_venv) matt@matt:~$ pip freeze  # nothing prints
(my_venv) matt@matt:~$ pip install biopython
(my_venv) matt@matt:~$ pip freeze > requirements.txt
(my_venv) matt@matt:~$ cat requirements.txt
biopython==1.77
numpy==1.18.4
```

```
matt@matt:~$ pip install -r requirements.txt
```

Summary - Dependencies

Everyone should be running the same version of your code.

Dependencies are other pieces of code your software depends on.

Missing or incompatible dependencies prevent others from running your software.

Python dependencies can be managed using:

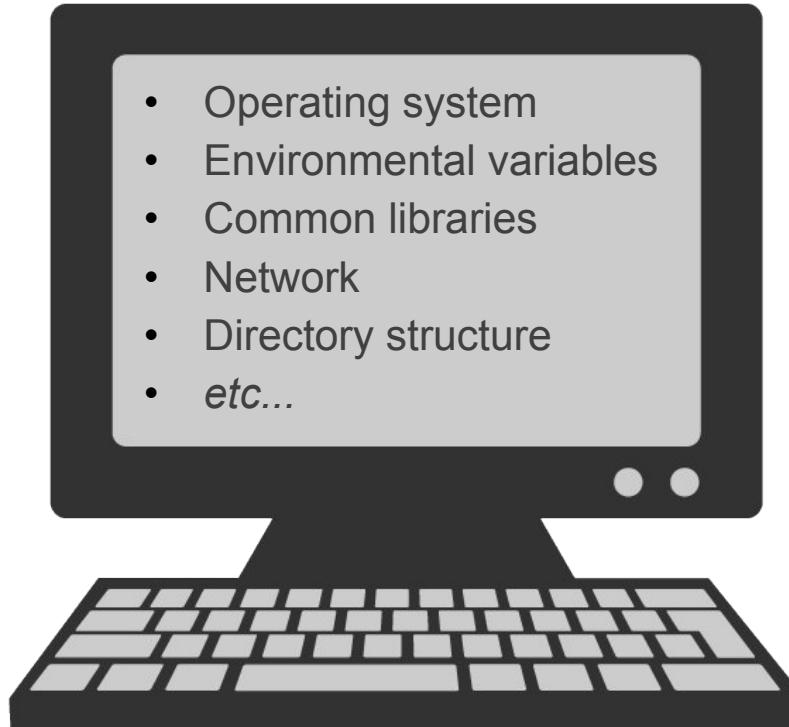
- 'venv' - gives an isolated python environment
- 'pip freeze' - lists all the python dependencies of your software



Reproducible Pipeline Checklist

- Code is available
- Good documentation
- Dependencies are listed
- Runtime environment can be reproduced

What is a Runtime Environment?



**Everything you need
to run a program.**

Runtime Environment Can Affect Results

BARD1 : P24S
ERCC2 : D312N
PDGFRA : S478P
BRCA1 : S24P
PMS2 : P470S
TP53 : P72R

OS Upgrade

BARD1 : P24S
ERCC2 : D312N
PDGFRA : S478P
KRAS : P102A
PMS2 : P470S
TP53 : P72R



Runtime Environment Can Affect Results

CentOS 6			CentOS 7			
Rank	Mutation	Score	Rank	Mutation	Score	
98	TP53:P72R	23	98	TP53:P72R	23	
99	ERCC2:D312N	22	99	ERCC2:D312N	22	
Threshold	100	BRCA1:S24P	20	100	KRAS:P102A	20
	101	KRAS:P102A	20	101	BRCA1:S24P	20
	102	PMS2:P470S	18	102	PMS2:P470S	18
	103	BARD1:P24S	18	103	BARD1:P24S	18



Managing Dependencies in Your Runtime Environment

- Able to install and run the pipeline in different runtime environments (different operating systems)
- However, we got inconsistent results between environments
- We cannot expect everyone to have the exact same runtime environment

How do we ensure our pipeline runs the same in any environment?

Containers

Container

- A package of one or more applications and all dependencies
- The environment is isolated from the host OS/infrastructure
- Extremely portable

Docker or Singularity

- An application for running containers

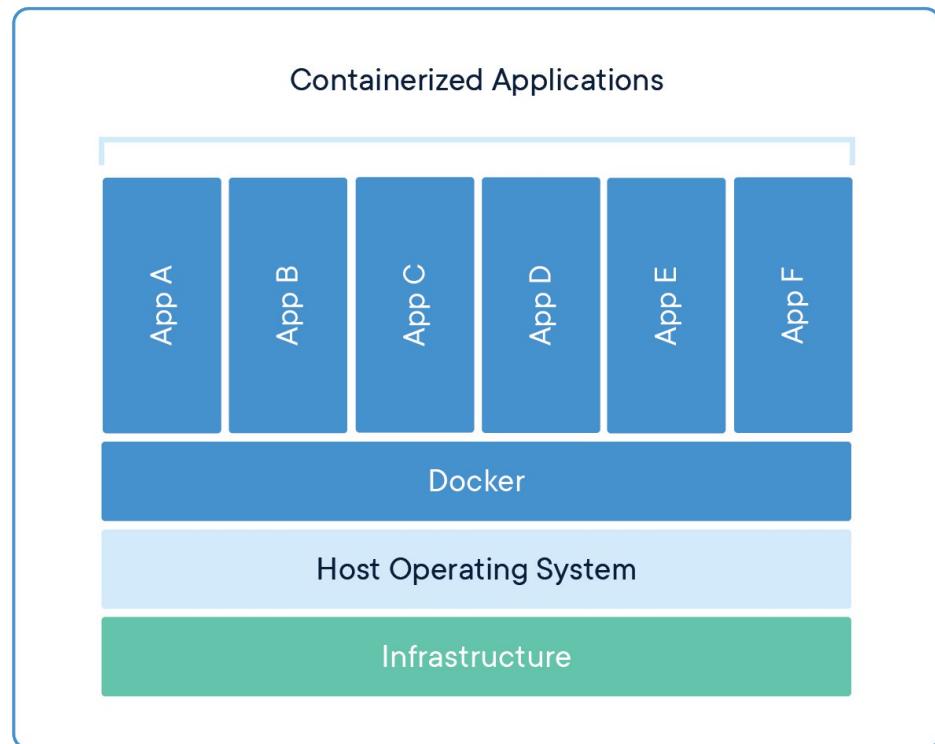


Image: <https://www.docker.com/resources/what-container>

Dockerhub



Docker EE Docker CE

Containers

Plugins

Explore Repositories Organizations Get Help ▾

gscmdouglas ▾



Filters

Docker Certified ⓘ

Docker Certified

Images

Verified Publisher ⓘ
Docker Certified And Verified Publisher Content

Official Images ⓘ
Official Images Published By Docker

Categories ⓘ

- Analytics
- Application Frameworks
- Application Infrastructure
- Application Services
- Base Images
- Databases
- DevOps Tools
- Featured Images
- Messaging Services
- Monitoring
- Operating Systems
- Programming Languages
- Security
- Storage

1 - 25 of 81,615 results for **ubuntu**. [Clear search](#)

Most Popular ▾



ubuntu

Updated 17 minutes ago

OFFICIAL IMAGE

10M+ 10K+
Downloads Stars

Ubuntu is a Debian-based Linux operating system based on free software.

Container Linux ARM 386 IBM Z PowerPC 64 LE ARM 64 x86-64 Base Images Operating Systems



ubuntu-debootstrap

Updated 15 minutes ago

OFFICIAL IMAGE

5M+ 44
Downloads Stars

debootstrap --variant=minbase --components=main,universe --include=inetutils-ping,iproute2 <suite> /

Container



ubuntu-upstart

Updated 15 minutes ago

OFFICIAL IMAGE

1M+ 109
Download Stars

Upstart is an event-based replacement for the /sbin/init daemon which starts processes at boot

Container



neurodebian

Updated 15 minutes ago

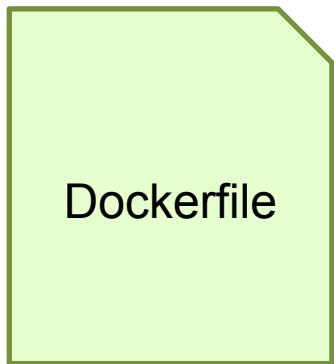
OFFICIAL IMAGE

5M+ 68
Downloads Stars

NeuroDebian provides neuroscience research software for Debian, Ubuntu, and other derivatives.

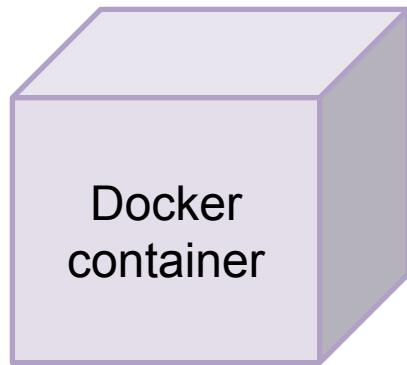
Building a Docker Container

Step 1



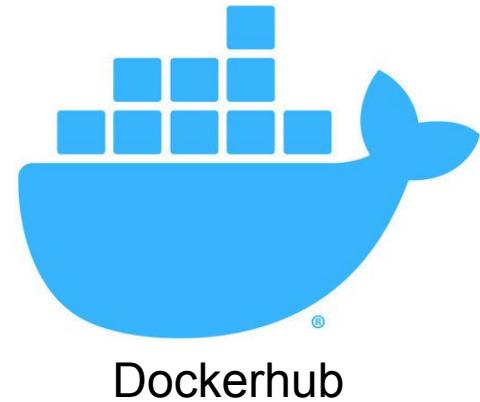
Build

Step 2



Upload

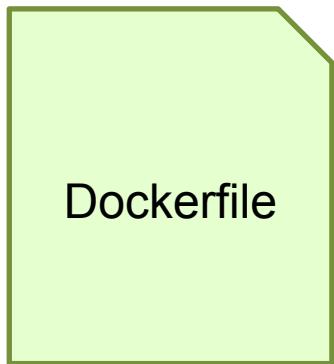
Step 3



Dependencies

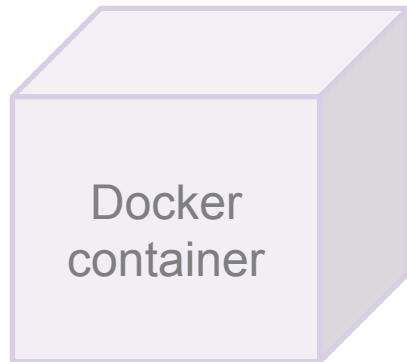
Building a Docker Container

Step 1



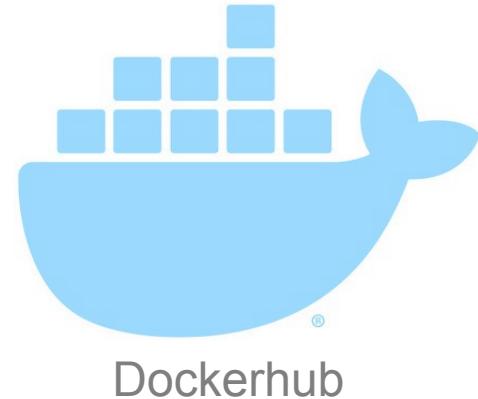
Build

Step 2



Upload

Step 3



Dockerhub

Dependencies

Building a Docker Container - Dockerfile

A '**Dockerfile**' is a text file.

It is a set of instructions on how to build and run the Docker container.

Contains all the commands the user would run to install and run the pipeline.

```
matt@matt:~$ vi Dockerfile
```



Building a Docker Container - Dockerfile

1) Pick a parent image.

Parent image = An existing docker image to base your image on.

Declare a parent image using the **FROM** keyword.

```
FROM ubuntu:20.04
```



Building a Docker Container - Dockerfile

2) Install your pipeline.

RUN executes the following command when the image is being built.

WORKDIR sets the directory commands will be run in ('cd' in bash).

```
FROM ubuntu:20.04

RUN git clone https://github.com/mattdoug604/reproducible_tutorial.git
WORKDIR reproducible_tutorial
RUN pip install -r requirements.txt
```

Building a Docker Container - Dockerfile

3) OPTIONAL: Run a command when the Docker container is started.

Alternative is to run the Docker container "interactively" - behaves essentially the same as the normal command line environment (run commands, navigate directories, etc.).

```
FROM ubuntu:20.04

RUN git clone https://github.com/mattdoug604/reproducible_tutorial.git
WORKDIR reproducible_tutorial
RUN pip install -r requirements.txt

CMD bash run_pipeline.sh
```

Building a Docker Container - Version Control

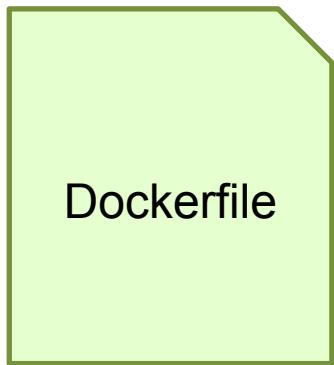
What if we rebuild the docker image and the git repository has changed?

```
RUN git clone https://github.com/mattdoug604/reproducible_tutorial.git  
WORKDIR reproducible_tutorial  
RUN git checkout v1.0.0  
RUN pip install -r requirements.txt
```

Best practice is to control the version of each program included in the image.

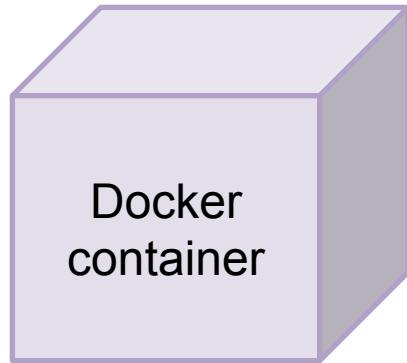
Building a Docker Container

Step 1



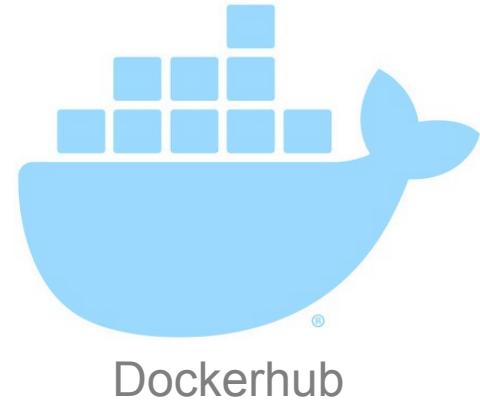
Build

Step 2



Upload

Step 3



Dependencies

Building a Docker Container - Build Command

```
matt@matt:~$ docker build -t reproducible_tutorial:1.0.0 .
```



'docker'
command

Building a Docker Container - Build Command

```
matt@matt:~$ docker build -t reproducible_tutorial:1.0.0 .
```



'build' a
docker
image

Building a Docker Container - Build Command

```
matt@matt:~$ docker build -t reproducible_tutorial:1.0.0 .
```

<name>:<tag>

name = name of the Docker container

- displayed when running the container

tag = version of the Docker container

- multiple versions of the same container



Building a Docker Container - Build Command

```
matt@matt:~$ docker build -t reproducible_tutorial:1.0.0 .
```



(Optional)
full path
to the
Dockerfile

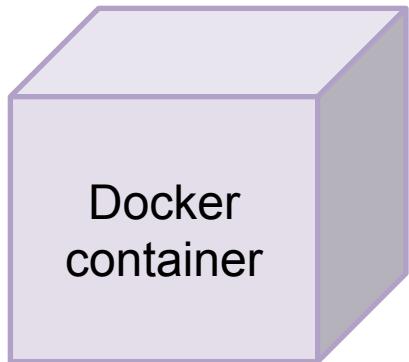
Building a Docker Container

Step 1



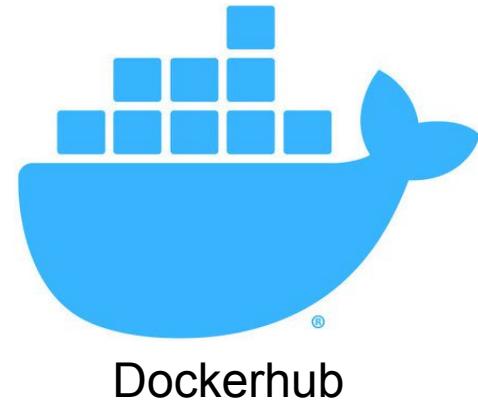
Build

Step 2



Upload

Step 3



Dependencies

Building a Docker Container - Dockerfile

```
matt@matt:~$ docker push mattdoug604/reproducible_tutorial:1.0.0
```



'docker'
command

Building a Docker Container - Dockerfile

```
matt@matt:~$ docker push mattdoug604/reproducible_tutorial:1.0.0
```



'push'
image to
Dockerhub

Building a Docker Container - Dockerfile

```
matt@matt:~$ docker push mattdoug604/reproducible_tutorial:1.0.0
```



your
Dockerhub
username

Building a Docker Container - Dockerfile

```
matt@matt:~$ docker push mattdoug604/reproducible_tutorial:1.0.0
```



<name>:<version>



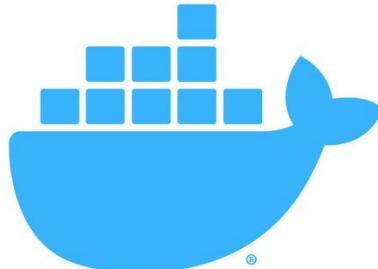
Running a Docker Container

Run the **CMD** we defined in the Dockerfile:

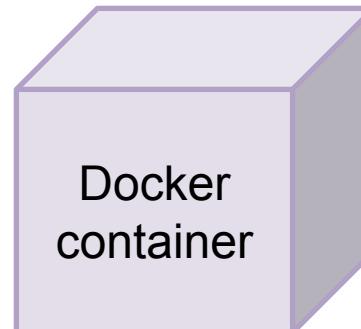
```
matt@matt:~$ docker run reproducible_tutorial:1.0.0
```

OR, run the Dockerfile interactively (-it):

```
matt@matt:~$ docker run -it reproducible_tutorial:1.0.0
```



Dockerhub

A screenshot of a terminal window titled "less". It displays the man page for the "less" command, which is a program similar to "more" but more feature-rich. The terminal shows various options and descriptions for the "less" command.

Summary - Runtime Environment

Installed software should run the same on any machine.

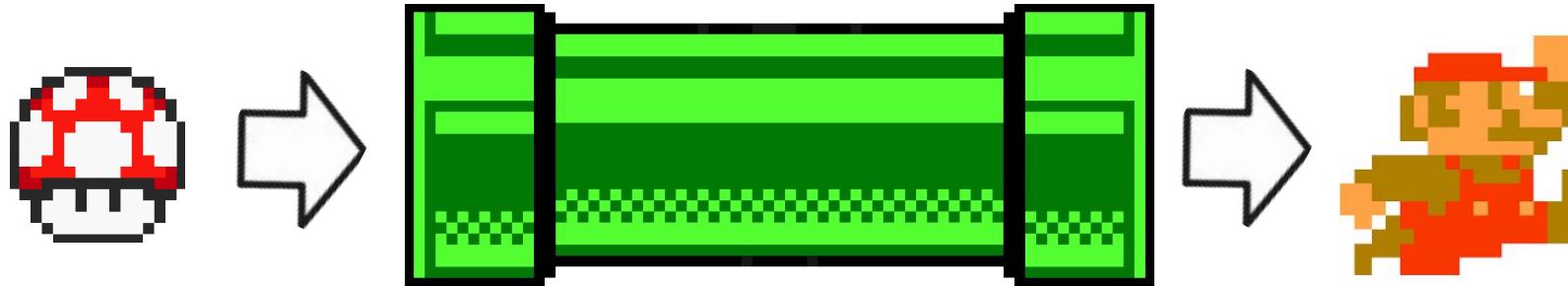
Containers

- Docker or Singularity
- Very portable
- Ensures everyone is running the same code

Reproducible Pipeline Checklist

-  Code is available
-  Good documentation
-  Dependencies are listed
-  Runtime environment can be reproduced

Summary



A failure to reproduce a result is often due to how the pipeline is packaged/shared.

Summary



- Code is easily accessible online
 - Github, Sourceforge



- Documentation explains how to obtain, install, and run code
 - READMEs

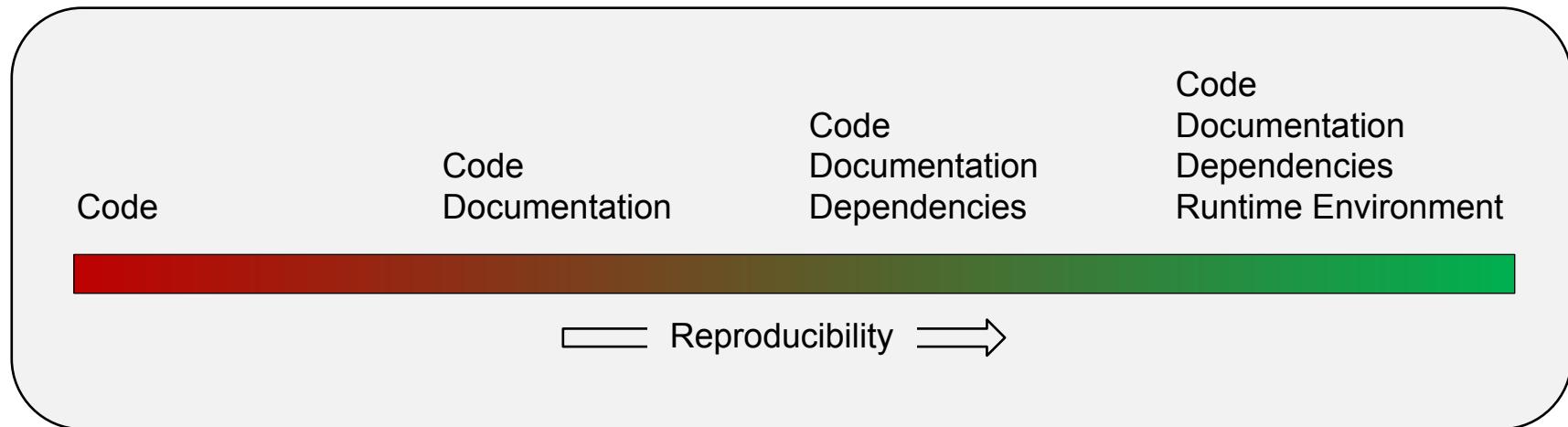


- All dependencies are listed
 - 'venv' and 'pip freeze' (Python)



- Runtime environment can be reproduced
 - Containers (Docker, Singularity) contain all dependencies

Summary



Summary

So much amazing data science software!

Pipelines are not living up to their full potential if they are not accessible.

Making sure a pipeline produces the same output anywhere is not trivial - but is achievable!



Thank You for Listening!