UNIVERSITY of WASHINGTON

DATA 515

Command line and intro to Jupyter notebooks

Joseph Hellerstein[1,3], **Bernease Herman[1]**, David Beck[1,2,3], Sam Gao[3]

[1]eScience Institute
[2]Chemical Engineering
[3]Computer Science

April 2, 2019

Survey the room

Raise your hand if…

Your primary computer's operating system is:

Linux or Mac

Windows 10

Earlier version of Windows than 10

You're inexperienced / never used a Unix command line

You've used Unix command line, but not made Bash script

You're experienced in Unix command line, Bash scripts

Agenda

1. Hands-on practice in using Unix command line
2. Hands-on practice creating Bash scripts
3. Brief introduction to Jupyter notebooks
4. **On separate video captures:** Version Control 1, Turning in homework

Confirm access to Unix command line

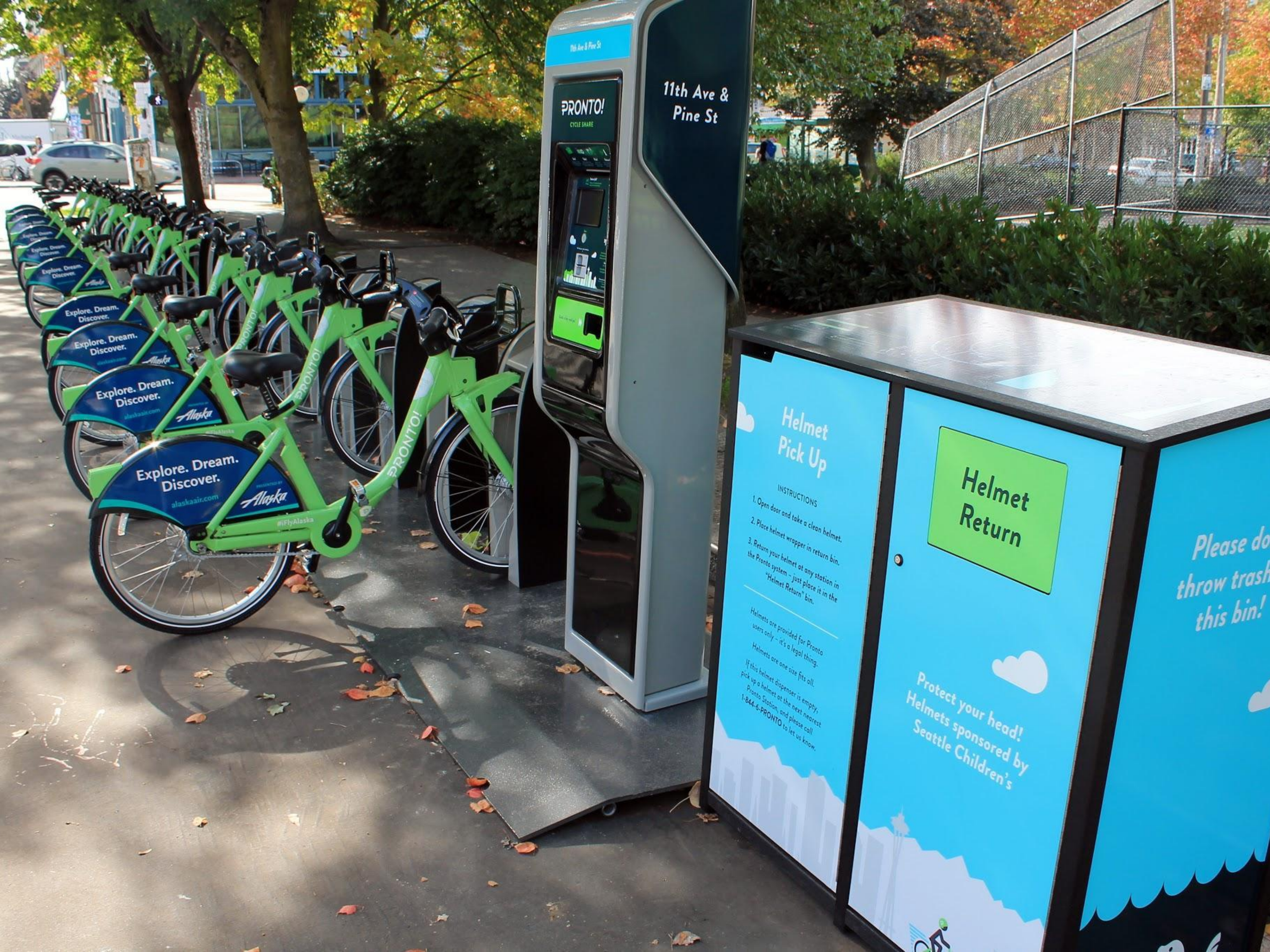
Linux/Mac: “Terminal”

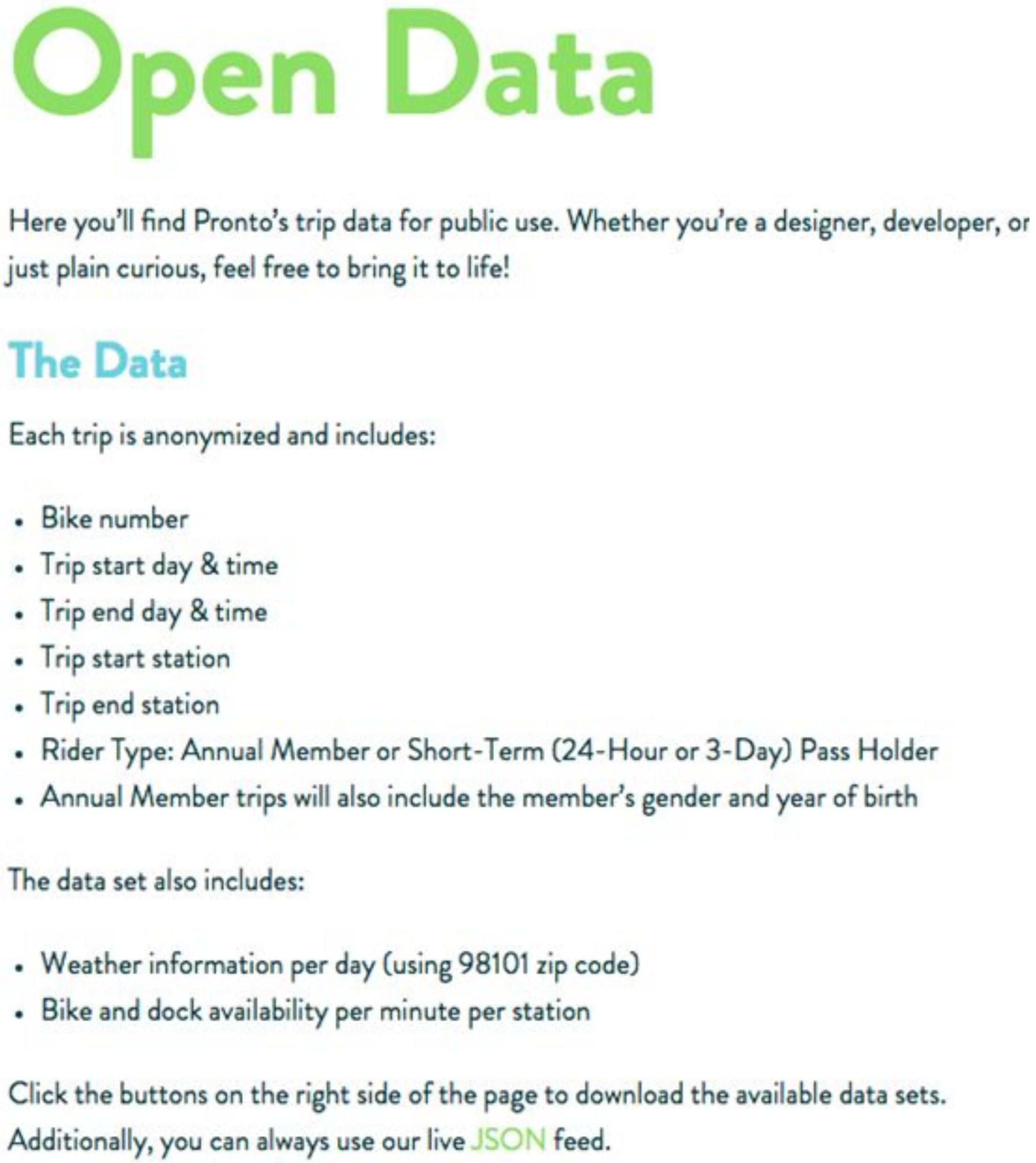
Windows 10: Linux subsystem, e.g. “Ubuntu”

Windows Pre-10: “Git Bash” or “Cygwin”

Introduction to Unix command line

Playing with Pronto data


11th Ave & Pine St
PRONTO! CYCLE SHARE
11th Ave & Pine St
Explore. Dream. Discover.
alaskaair.com
Alaska
#iFlyAlaska
PRONTO!
Helmet Pick Up
INSTRUCTIONS
1. Open door and take a clean helmet.
2. Place helmet wrapper in return bin.
3. Return your helmet at any station in the Pronto system – just place it in the "Helmet Return" bin.
Helmets are provided for Pronto users only – it's a legal thing.
Helmets are one size fits all.
If this helmet dispenser is empty, pick up a helmet at the next nearest Pronto Station, and please call 1-844-6-PRONTO to let us know.
Helmet Return
Protect your head! Helmets sponsored by Seattle Children's
Please do throw trash this bin!



Open Data

Here you'll find Pronto's trip data for public use. Whether you're a designer, developer, or just plain curious, feel free to bring it to life!

The Data

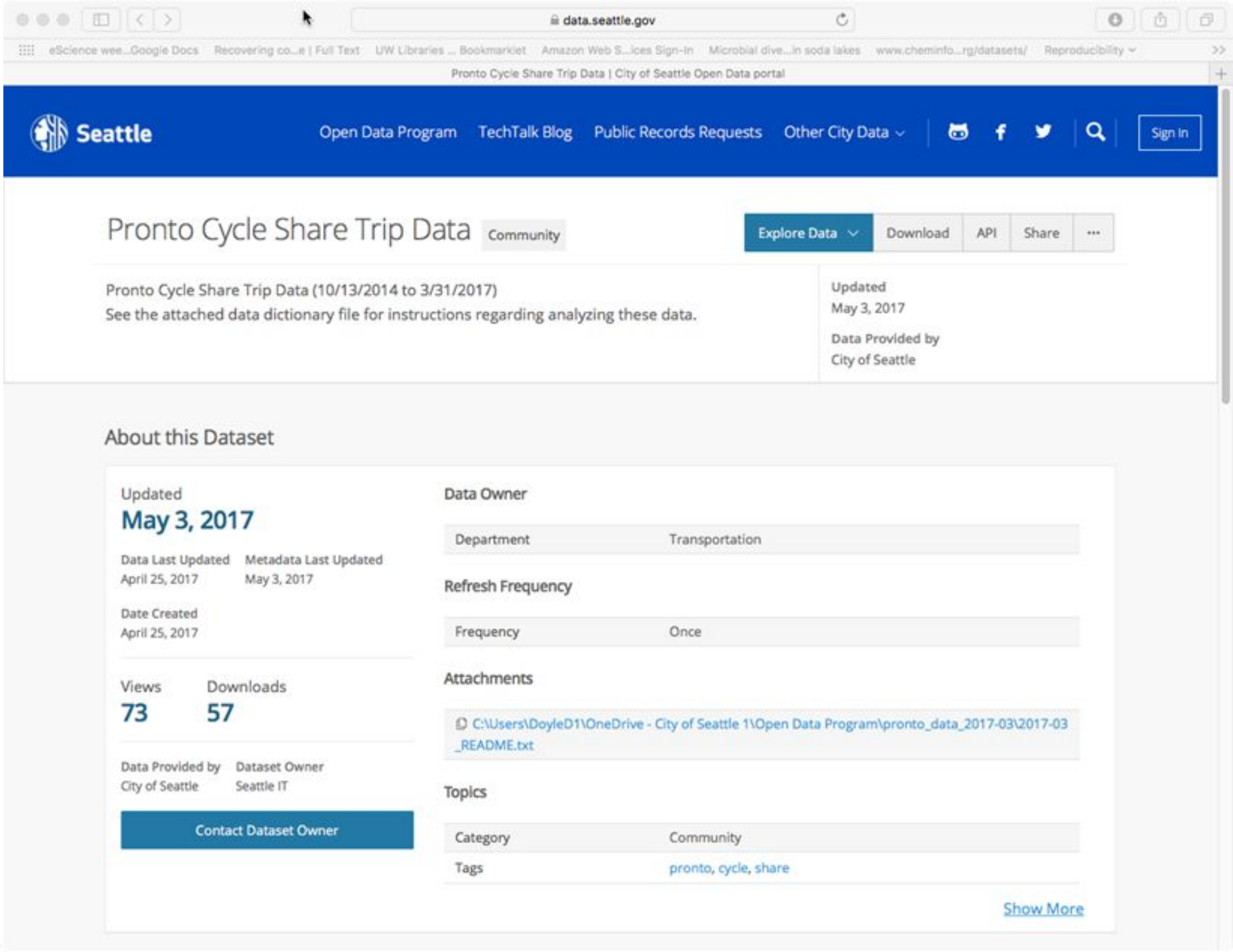
Each trip is anonymized and includes:

- Bike number
- Trip start day & time
- Trip end day & time
- Trip start station
- Trip end station
- Rider Type: Annual Member or Short-Term (24-Hour or 3-Day) Pass Holder
- Annual Member trips will also include the member's gender and year of birth

The data set also includes:

- Weather information per day (using 98101 zip code)
- Bike and dock availability per minute per station

Click the buttons on the right side of the page to download the available data sets. Additionally, you can always use our live JSON feed.


data.seattle.gov
eScience wee...Google Docs
Recovering co...e | Full Text
UW Libraries ... Bookmarklet
Amazon Web S...ices Sign-In
Microbial dive...in soda lakes
www.cheminfo...rg/datasets/
Reproducibility
Pronto Cycle Share Trip Data | City of Seattle Open Data portal
Seattle
Open Data Program
TechTalk Blog
Public Records Requests
Other City Data
Sign In
Explore Data
Download
API
Share
···

Pronto Cycle Share Trip Data

Community

Pronto Cycle Share Trip Data (10/13/2014 to 3/31/2017)
See the attached data dictionary file for instructions regarding analyzing these data.

Updated
May 3, 2017

Data Provided by
City of Seattle

About this Dataset

Updated
May 3, 2017

Data Last Updated
April 25, 2017

Metadata Last Updated
May 3, 2017

Date Created
April 25, 2017

Views
73

Downloads
57

Data Provided by
City of Seattle

Dataset Owner
Seattle IT

Contact Dataset Owner

Data Owner

Department	Transportation

Refresh Frequency

Frequency	Once

Attachments

C:\Users\DoyleD1\OneDrive - City of Seattle 1\Open Data Program\pronto_data_2017-03\2017-03_README.txt

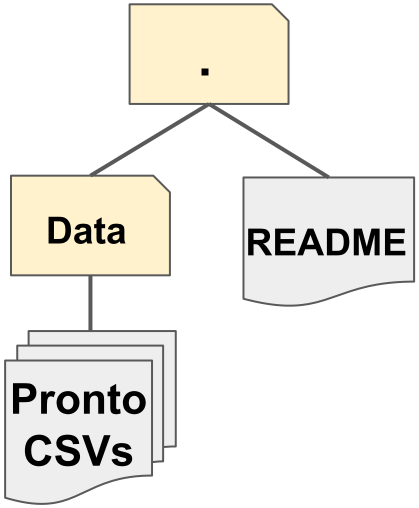
Topics

Category	Community
Tags	pronto, cycle, share

Show More

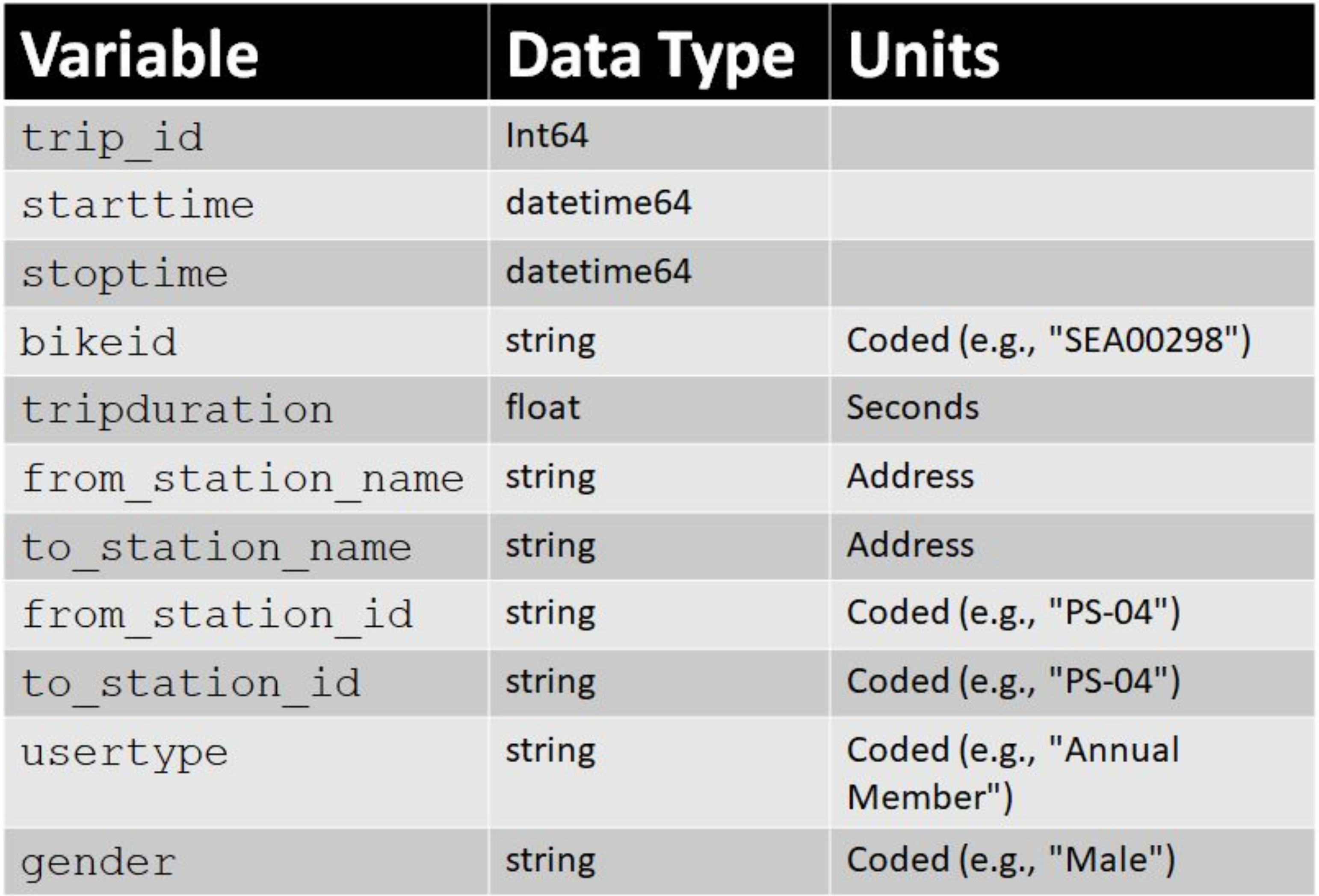
Activity: Storing data in directories

Create directory structure, download Pronto data from internet,
Unpack the data



Link:

https://data.seattle.gov/api/views/tw7j-dfaw/rows.csv?accessType=DOWNLOAD



Variable	Data Type	Units
trip_id	Int64	
starttime	datetime64	
stoptime	datetime64	
bikeid	string	Coded (e.g., "SEA00298")
tripduration	float	Seconds
from_station_name	string	Address
to_station_name	string	Address
from_station_id	string	Coded (e.g., "PS-04")
to_station_id	string	Coded (e.g., "PS-04")
usertype	string	Coded (e.g., "Annual Member")
gender	string	Coded (e.g., "Male")

Fields in Pronto data

Introduction to pipes, filters, and Bash scripts

Introduction to Jupyter notebooks

DATA 515

Version Control I

Joseph Hellerstein[1,3], **Bernease Herman[1]**, David Beck[1,2,3], Sam Gao[3]

[1]eScience Institute
[2]Chemical Engineering
[3]Computer Science

April 2, 2019

Survey the room

Raise your hand if…

You've never used source / version control

You've used source control, but not Git

Beginner in Git as an individual (`git add`, `git commit`)

Moderate/Advanced in Git as an individual (`git rebase`)

Beginner in Git as collaborative team (`git push, git pull`)

Moderate/Advanced in Git as collaborative team

(pull requests, hooks)

Agenda

1. Confirm sign-in to GitHub, send with UW Net ID and student number to instructors
2. Version Control, Git, and GitHub
3. Hands on practice with Git & GitHub for individuals
4. **On separate video capture:** Turning in homework

Questions from last lecture on Procedural Python?

Why use version control?

Compare writing software with writing a manuscript.

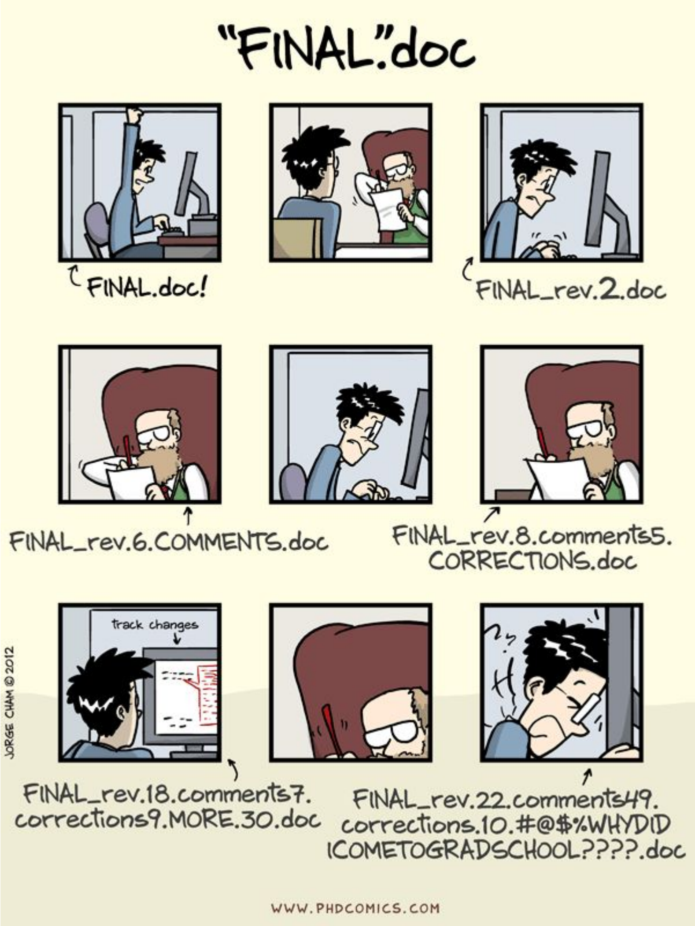
- Use undo to revert to a previous state
- Use track changes when sharing document with advisor

Still, word processors can still be frustrating.

- Intelligently combine changes made concurrently
- Record reasoning (why? for code, what?) for changes
- Efficient use of file storage

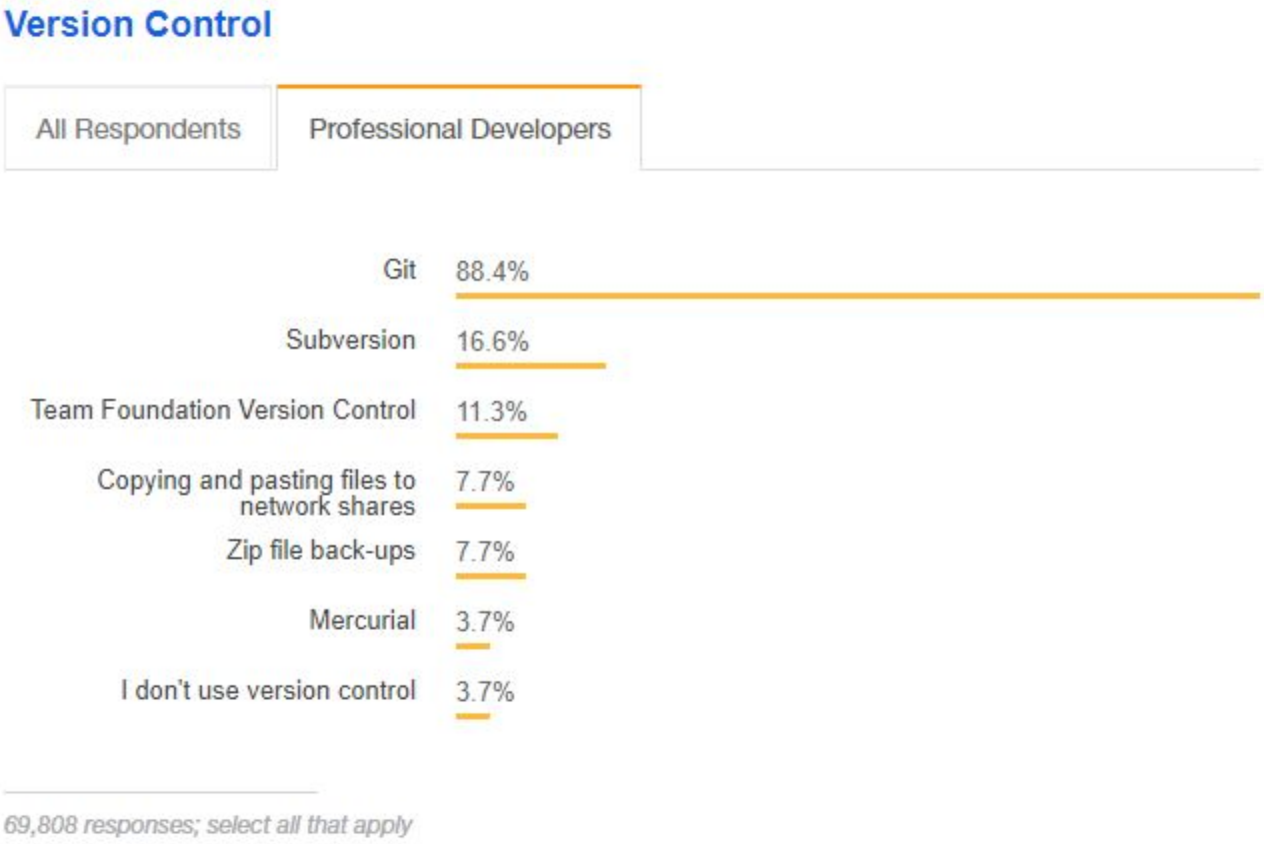
Modern version control systems can address all of these.

Tracking versions and efficient storage



http://phdcomics.com/comics/archive_print.php?comicid=1531

Stack Overflow Developer Survey 2018



Version Control

Git	88.4%
Subversion	16.6%
Team Foundation Version Control	11.3%
Copying and pasting files to network shares	7.7%
Zip file back-ups	7.7%
Mercurial	3.7%
I don't use version control	3.7%

69,808 responses; select all that apply

Git is a de facto standard for VCS

Benefits (**+++++**)

- **Performance:** operations in Git are optimized and fast
- **Flexibility:** doesn’t require use of a particular workflow
- **Security:** protection against untraceable changes
- **Popularity:** employment, available on many platforms
- **Distributed:** can be used offline, no need for server

Downsides (**-**)

- **Distributed:** not ideal for large files, merging changes

Version control in the cloud, GitHub

A working copy of your repository stored on GitHub’s servers connected and accessible via the internet

Others can download and use your code

Online repository is suitable as central repository

Social features, i.e. issue tracker, comments, notifications

Alternatives: Atlassian’s BitBucket, GitLab, SourceForge

Hands on Git / GitHub I

INTRODUCTION TO GIT

*(and some GitHub)

0. Set up

>git config [options]
>git init
>.gitignore

INTRODUCTION TO GIT

*(and some GitHub)

0. Set up

```
> git config [options]
> git init
>.gitignore
```

1. Make changes

(use your preferred editor and tools.)

INTRODUCTION TO GIT

*(and some GitHub)

0. Set up

```
>git config [options]
>git init
>.gitignore
```

1. Make changes

(use your preferred editor and tools.)

2. Stage changed files

```
>git add
>git add -A
>git rm [path]
```

INTRODUCTION TO GIT

*(and some GitHub)

0. Set up

```
>git config [options]
>git init
>.gitignore
```

1. Make changes

(use your preferred editor and tools.)

2. Stage changed files

```
>git add
>git add -A
>git rm [path]
```

3. Create snapshot

```
>git commit
>git commit
   -m "[msg]
```

INTRODUCTION TO GIT

*(and some GitHub)

0. Set up

```
>git config [options]
>git init
>.gitignore
```

1. Make Changes

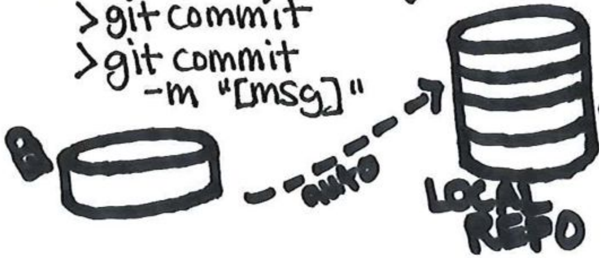
(use your preferred editor and tools.)

2. Stage changed files

```
>git add
>git add -A
>git rm [path]
```

3. Create snapshot

```
>git commit
>git commit
   -m "[msg]"
```



INTRODUCTION TO GIT

*(and some GitHub)

0. Set up

```
>git config [options]
>git init
>.gitignore
```

1. Make Changes

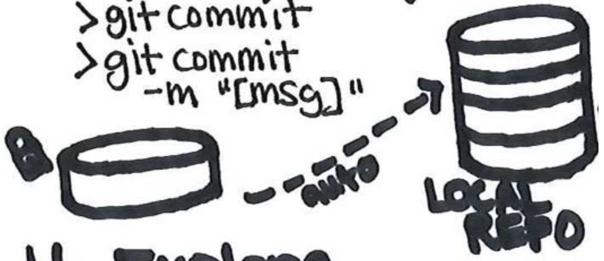
(Use your preferred editor and tools.)

2. Stage changed files

```
>git add
>git add -A
>git rm [path]
```

3. Create snapshot

```
>git commit
>git commit
   -m "[msg]"
```



4. Explore

```
>git status
>git log [options]
>git show [sha1]
```

(Repeat 1-4 as desired.)

INTRODUCTION TO GIT

*(and some GitHub)

0. Set up

```
>git config [options]
>git init
>.gitignore
```

1. Make changes

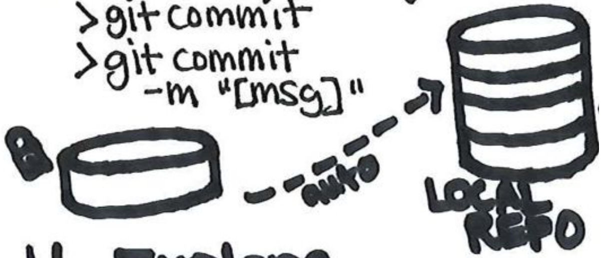
(use your preferred editor and tools.)

2. Stage changed files

```
>git add
>git add -A
>git rm [path]
```

3. Create snapshot

```
>git commit
>git commit
   -m "[msg]"
```



REMOTE REPO

4. Explore

```
>git status
>git log [options]
>git show [sha1]
```

(Repeat 1-4 as desired.)

INTRODUCTION TO GIT

*(and some GitHub)

0. Set up

```
>git config [options]
>git init
>.gitignore
```

1. Make changes

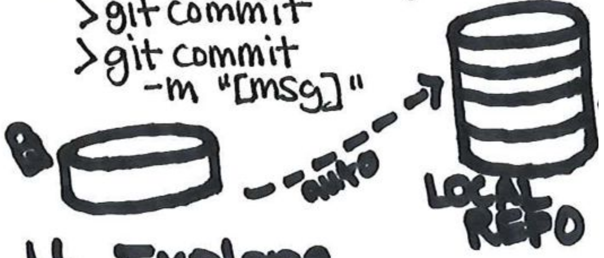
(use your preferred editor and tools.)

2. Stage changed files

```
>git add
>git add -A
>git rm [path]
```

3. Create snapshot

```
>git commit
>git commit
   -m "[msg]"
```



4. Explore

```
>git status
>git log [options]
>git show [sha1]
```

(Repeat 1-4 as desired.)

5. Add remote

```
>git remote add [name][url]
>git remote -v
```

INTRODUCTION TO GIT

*(and some GitHub)

0. Set up

```
>git config [options]
>git init
>.gitignore
```

1. Make changes

(use your preferred editor and tools.)

2. Stage changed files

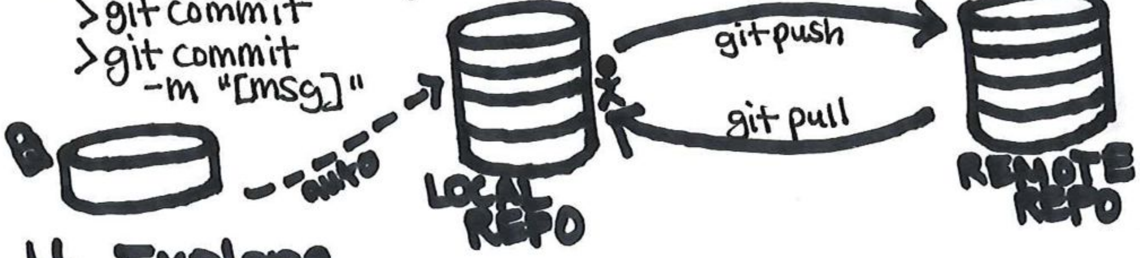
```
>git add
>git add -A
>git rm [path]
```

3. Create snapshot

```
>git commit
>git commit
   -m "[msg]"
```

5. Add remote

```
>git remote add [name][url]
>git remote -v
```



4. Explore

```
>git status
>git log [options]
>git show [sha1]
```

(Repeat 1-4 as desired.)

INTRODUCTION TO GIT

*(and some GitHub)

0. Set up

```
>git config [options]
>git init
>.gitignore
```

1. Make Changes

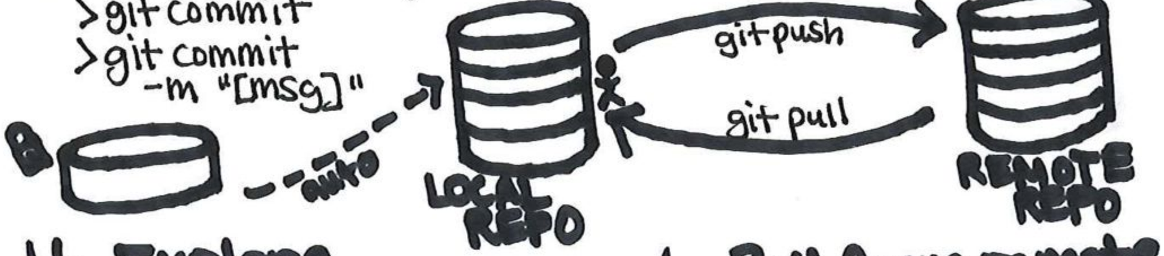
(use your preferred editor and tools.)

2. Stage changed files

```
>git add
>git add -A
>git rm [path]
```

3. Create snapshot

```
>git commit
>git commit
   -m "[msg]"
```



4. Explore

```
>git status
>git log [options]
>git show [sha1]
```

(Repeat 1-4 as desired.)

5. Add remote

```
>git remote add [name][url]
>git remote -v
```

6. Pull from remote

```
>git fetch [remote][branch]
>git pull [remote][branch]
```

INTRODUCTION TO GIT

*(and some GitHub)

0. Set up

```
>git config [options]
>git init
>.gitignore
```

1. Make Changes

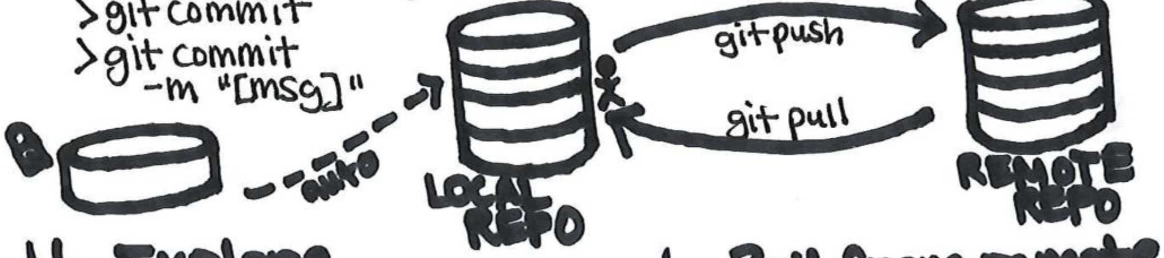
(use your preferred editor and tools.)

2. Stage changed files

```
>git add
>git add -A
>git rm [path]
```

3. Create snapshot

```
>git commit
>git commit
   -m "[msg]"
```



4. Explore

```
>git status
>git log [options]
>git show [sha1]
```

(Repeat 1-4 as desired.)

5. Add remote

```
>git remote add [name][url]
>git remote -v
```

6. Pull from remote

```
>git fetch [remote][branch]
>git pull [remote][branch]
```

7. Push to remote

```
>git push [remote][branch]
```

INTRODUCTION TO GIT

*(and some GitHub)

0. Set up

```
>git config [options]
>git init
>.gitignore
```

1. Make Changes

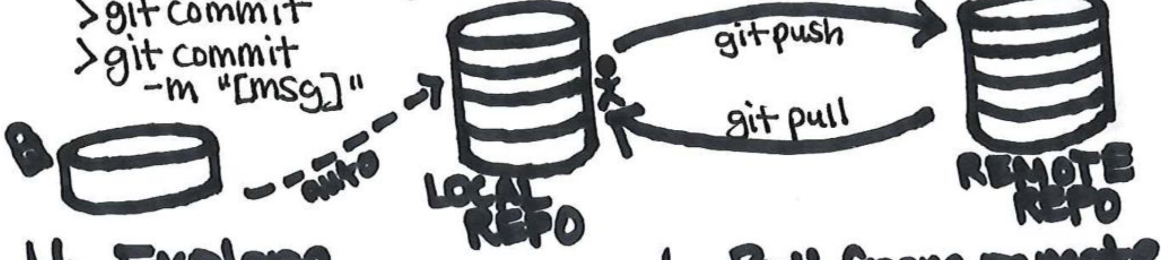
(use your preferred editor and tools.)

2. Stage changed files

```
>git add
>git add -A
>git rm [path]
```

3. Create snapshot

```
>git commit
>git commit
    -m "[msg]"
```



4. Explore

```
>git status
>git log [options]
>git show [sha1]
```

(Repeat 1-4 as desired.)

BONUS: Conflicts

```
>git merge
```

TIP: pull before commit to minimize conflicts!

5. Add remote

```
>git remote add [name][url]
>git remote -v
```

6. Pull from remote

```
>git fetch [remote][branch]
>git pull [remote][branch]
```

7. Push to remote

```
>git push [remote][branch]
```

Questions on version control?

(Note that some topics will be covered in later VCS sessions)

UNIVERSITY of WASHINGTON

DATA 515

Submitting Homework via GitHub Classroom

Joseph Hellerstein[1,3], **Bernease Herman[1]**, David Beck[1,2,3], Sam Gao[3]

[1]eScience Institute
[2]Chemical Engineering
[3]Computer Science

April 2, 2019

Questions on version control?

(Note that some topics will be covered in later VCS sessions)

https://classroom.github.com/a/_tncTr0r