GEOG 432/832: Programming, Scripting, and Automation for GIS

Unit 09.01: Geographic data science and tools

Dr. Bitterman

Today's schedule

- Open discussion
- Project update presentation(s)
- Slides, discussion and exercises
- For next class

Open discussion

The second half of this course

- Bye, bye ArcPy
- Readings:
  - posted to Canvas
  - from: Geographic Data Science with PySAL and the PyData Stack (https://geographicdata.science/book/intro.html)
- In class work: in Jupyter notebooks via Anaconda (but not in ArcGIS)
- Your "at home" work: notebooks preferred, but .py scripts ok too
- 4 labs remaining:
  - Shorter in length
  - But each is due in just 6 days (but you don't need to come to the lab)
  - **100% open source**

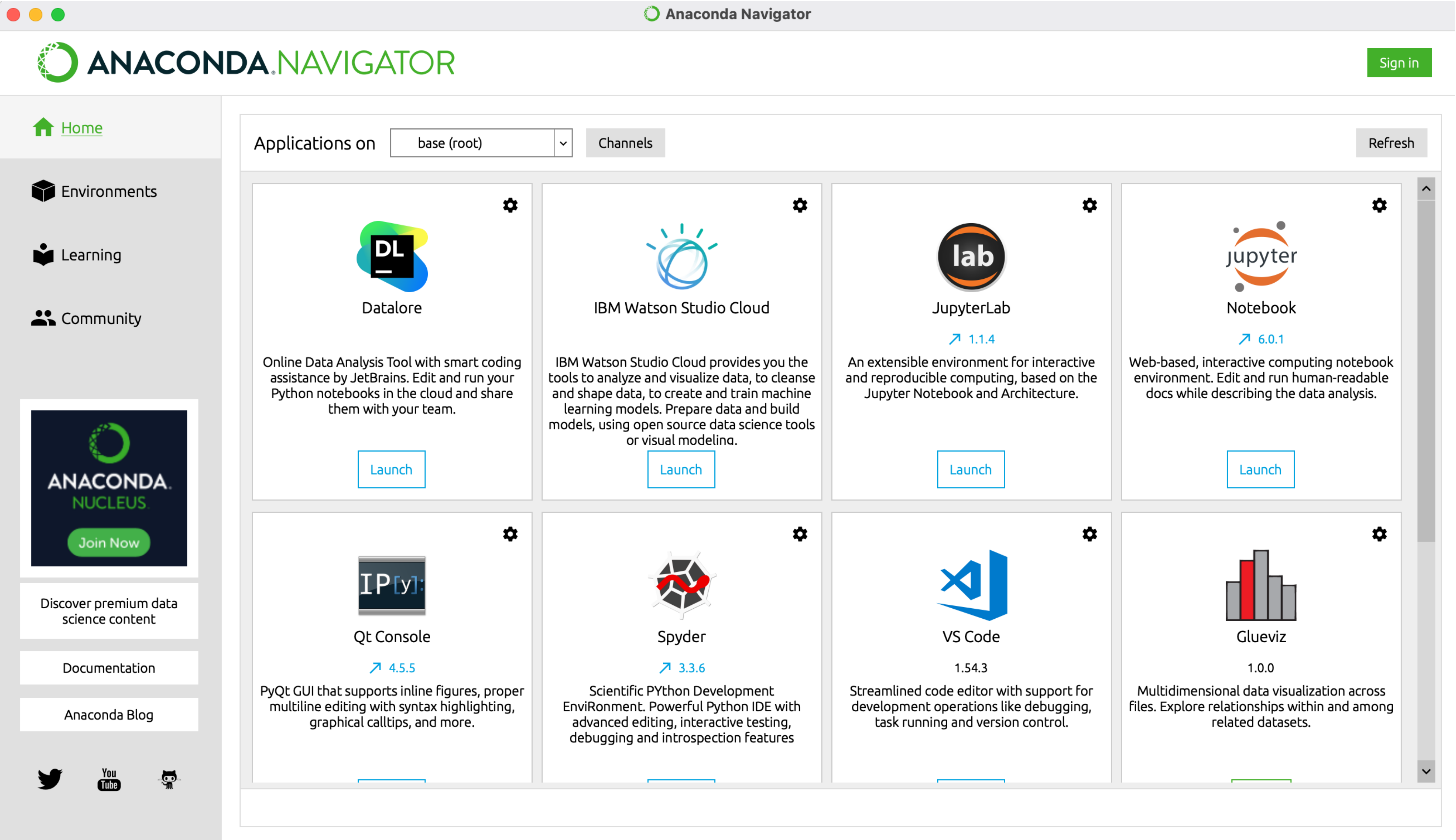
Today's prep:

- We'll use *unit09inclass.zip* from Canvas
- Open Anaconda
- Wait

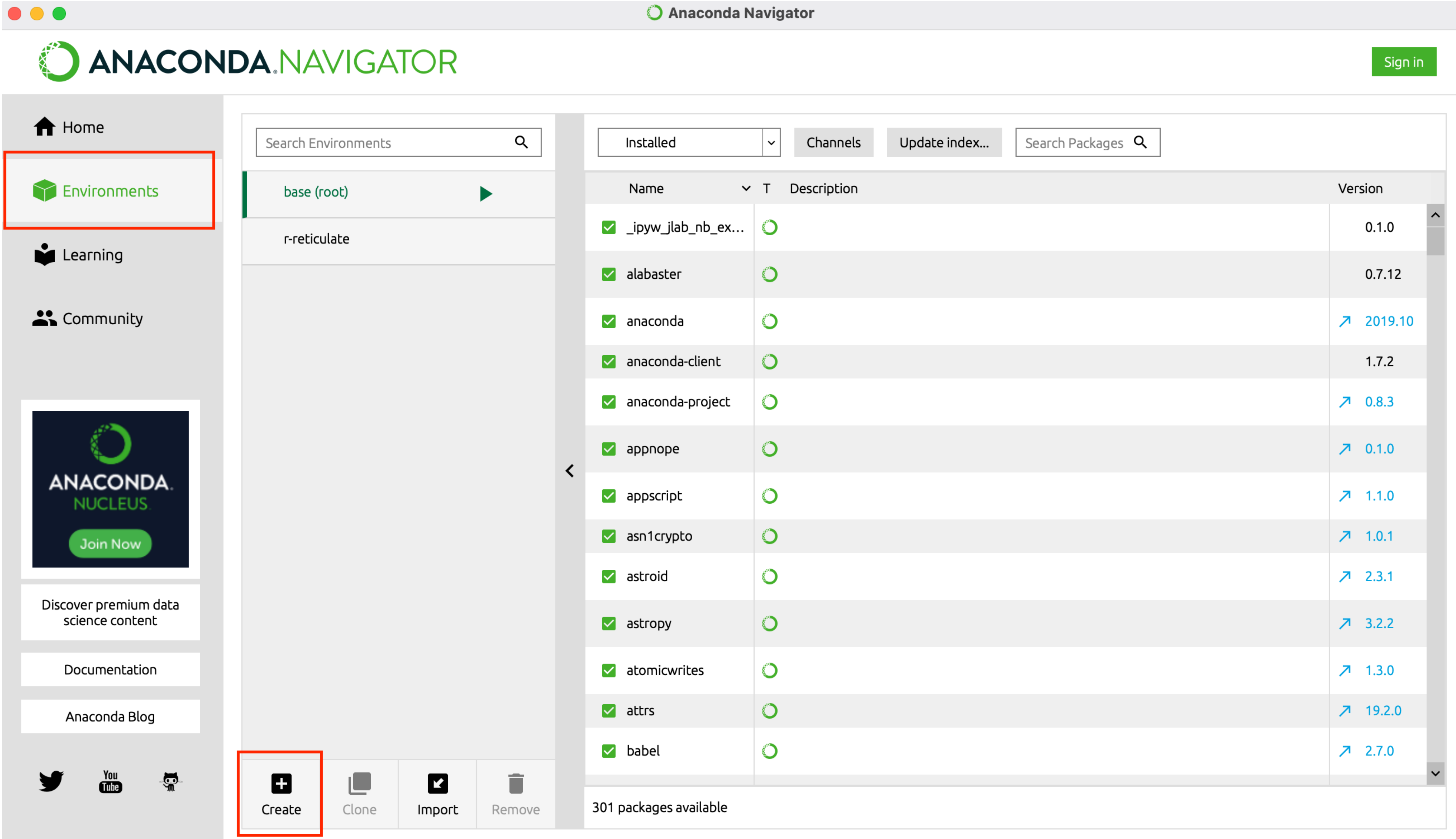
Challenges in a group computing environment

- We don't have individual admin-level access
- Warning: some workarounds required (they are NOT required on your personal machines)
- Process:
  i. Create a new environment
  ii. Install packages
  iii. Re-add Jupyter Notebook to the environment
  iv. Finally do some work

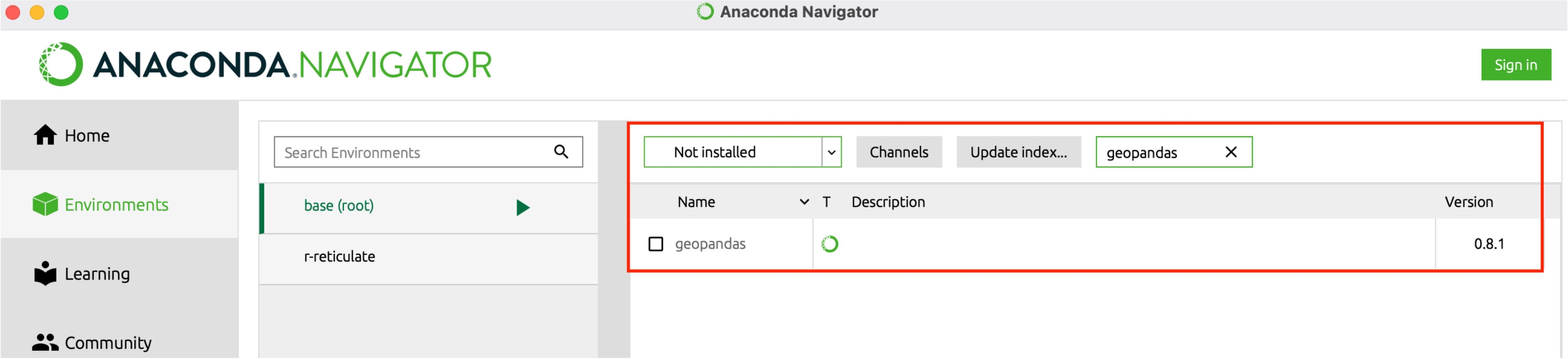
Anaconda


Anaconda Navigator
ANACONDA.NAVIGATOR
Sign in
Home
Environments
Learning
Community
ANACONDA. NUCLEUS
Join Now
Discover premium data science content
Documentation
Anaconda Blog
Applications on
base (root)
Channels
Refresh
Datalore
Online Data Analysis Tool with smart coding assistance by JetBrains. Edit and run your Python notebooks in the cloud and share them with your team.
Launch
IBM Watson Studio Cloud
IBM Watson Studio Cloud provides you the tools to analyze and visualize data, to cleanse and shape data, to create and train machine learning models. Prepare data and build models, using open source data science tools or visual modeling.
Launch
JupyterLab
1.1.4
An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.
Launch
Notebook
6.0.1
Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.
Launch
Qt Console
4.5.5
PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.
Spyder
3.3.6
Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features
VS Code
1.54.3
Streamlined code editor with support for development operations like debugging, task running and version control.
Glueviz
1.0.0
Multidimensional data visualization across files. Explore relationships within and among related datasets.

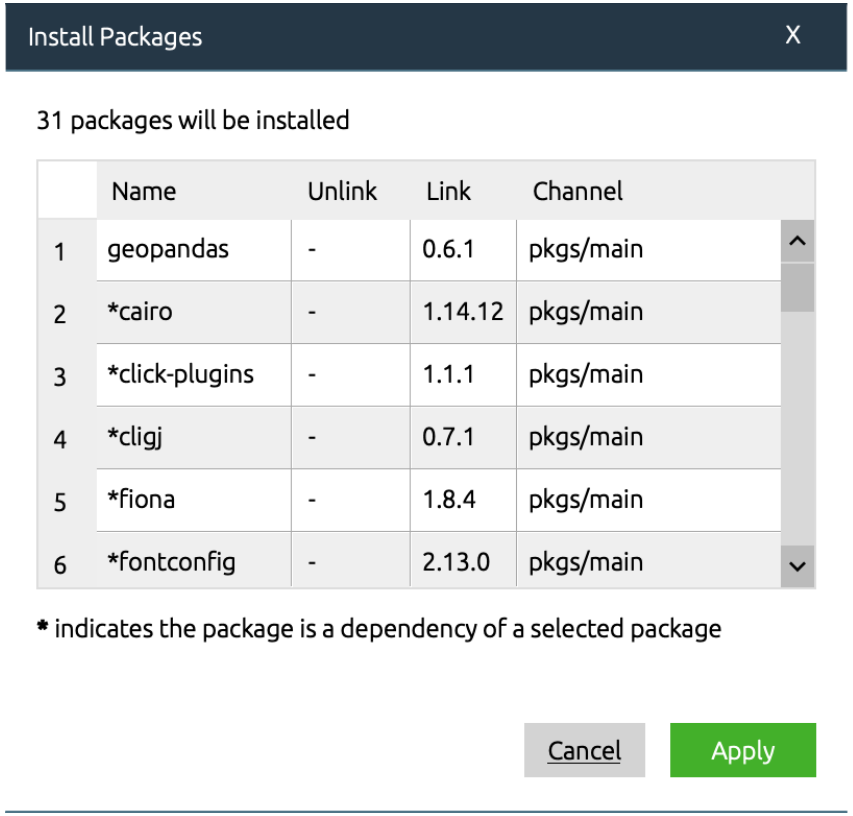
Creating a new environment



Searching for and installing packages



The brilliance of package managers



Install Packages X

31 packages will be installed

	Name	Unlink	Link	Channel
1	geopandas	-	0.6.1	pkgs/main
2	*cairo	-	1.14.12	pkgs/main
3	*click-plugins	-	1.1.1	pkgs/main
4	*cligj	-	0.7.1	pkgs/main
5	*fiona	-	1.8.4	pkgs/main
6	*fontconfig	-	2.13.0	pkgs/main

* indicates the package is a dependency of a selected package

Cancel Apply

Some GIScience

Geographic processes are represented using objects, fields, and networks

- **Objects**: discrete entities that occupy a specific position in space and time
- **Fields**: continuous surfaces that could be measured at any location in space and time
- **Networks**: set of connections between *objects* or between positions in a *field*

Some standard data structurs

- a few key standards:
  - around for a long time
  - proven to be useful
- In this course:
  - geographic tables
  - surfaces
  - spatial graphs

have we seen any of these already?

Geographic tables

- store information about discrete objects
- two dimensional structures: rows & columns
- each row represents an independent object (or feature)
- each column stores an attribute of those objects

"Geographic tables" are (sort of) special

- one column stores geographic information
- combindes geographic and non-geographic information
- examples:
  - PostGIS tables (as a geographic extension of PostgreSQL)
  - R’s sf data frames
  - Python’s GeoDataFrame objects, provided by *geopandas*

Surfaces

- Fields: continuous representation of space (theoretically an infinite set of locations)
- In practice: measured at a discrete set of locations
- In practice: recorded and stored in uniform grids or *arrays*
- Arrays are matrices
  - at least two dimensions
- **Surface arrays:**
  - rows and columns signify location
  - cell values to store information about that location

Similarities to other data structures?

Graphs (networks)

- capture relations relationships between objects that are mediated through space
- Essentially geographic networks
- Store topologies

Examples:

- spatial weights matrices
- adjacency matrices
- spatial networks

Let's do some work

Setup

```
%matplotlib inline      # Ensures visualizations are plotted inside the notebook

import os               # Provides several system utilities
import pandas as pd     # Workhorse of data munging in Python
import geopandas        # The framework for geospatial data tables in Python
import seaborn as sns   # Allows us to efficiently and beautifully plot
```

- Comments are useful
- What does "**as**" do?

Reading some data:

Break it down FIRST

1. What does each line do?
    i. What functions?
    ii. Paramters?
2. What assumptions does the code make about how we've organized our data?

```
# Read table
print(os.getcwd())
d = pd.read_csv("./unit09data/ne_counties_census.csv", index_col='GEOID')
```

Displaying our data:

```
d    #simplest slide EVER!!!
```

What happened?

This is a DataFrame

- two dimensions: rows and columns
- each row and column is assigned an *index* (displayed in bold)
  - column indexes: generated from the .csv file’s column names
  - row indexes: specified when reading the file (**GEOID** in our case)
- VERY usefully, DataFrames (can) contain columns with different *types* of data

Some simple work with DataFrames

```
d.head() # Prints the first n records

d.tail() # Prints the last n records

d.info() # Prints an overview of the DataFarme

d.columns # all the column names (how is this different?)

for x in d.columns:
    print(x)
```

Give it a shot!

Further interrogation of our data

```
d.describe() # summary stats for the attributes

d.describe().T # Transposes the summary stats
```

Even more stats

```
d.max() # the maximum value of each column
maxes = d.max() # assign it to a variable of its own

d.min() # minimums

d.std() # standard deviations
```

Note: not all make sense!

Rows and columns

Get just one column

```
# get one column
d['PerCapInc']

# or just the maximum of one column
d['PerCapInc'].max()
```

Grabbing just one row

```
d.loc[31059]
```

One row, one column:

```
d.loc[31059]['PerCapInc']
```

We can also create new variables

Population of men in their 40s

```
m40s = d['M40to44Y'] + d['M45to49Y']
m40s
```

Add it back to the table

```
d['M40s'] = m40s
```

Or just do it "in place"

```
d['M40s'] = d['M40to44Y'] + d['M45to49Y']
```

Delete the field

```python
del d['M40s']
```

Conditional searches

A quick search

```
lowpops = d.loc[d['Total'] < 25000, :]
lowpops
```

Direct queries

```
d.query("(Total < 2000) & (Vacant / TotalUnits > .2)")
```

Some quick visulizations with histograms

Seaborn

```
sns.distplot(d['Total'], kde=False)
```

pandas

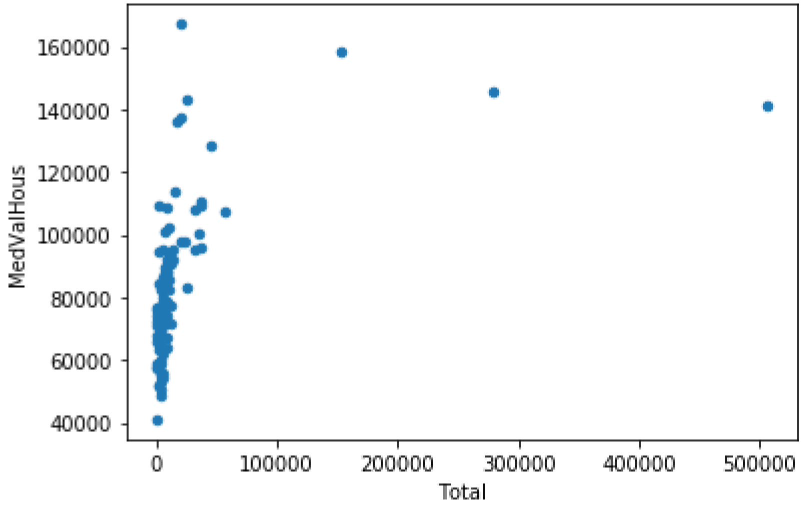
```
d.hist('Total') # calls matplotlib for the histogram
```

matplotlib (explicitly)

```
import matplotlib
matplotlib.pyplot.hist(d['Total'])
```

Or a scatterplot

```
d.plot.scatter('Total', 'MedValHous')
```



For next class

- Lab 4 due March 30th
- Lab 5 starts Friday (so some overlap)
- Readings are linked/posted on Canvas
- For more, look at: https://darribas.org/gds_course/content/bB/lab_B.html (used as a basis of today's examples)