

HPRC Data Workshop

Texas A&M Conference on Energy

Zhenhua He

12/03/2021



Data Labs

Lab I. JupyterLab

We will load a module with required libraries and run JupyterLab on the HPRC Terra Portal.

Lab II. Data Exploration

We will go through some examples with two popular Python libraries: Pandas and Matplotlib for data exploration.



Lab I. JupyterLab

Screenshot of the JupyterLab interface showing a notebook titled "Lorenz.ipynb".

The interface includes:

- File Bar:** File, Edit, View, Run, Kernel, Tabs, Settings, Help.
- File Explorer:** Shows files like "notebooks", "Data.ipynb", "Fasta.ipynb", "Julia.ipynb", "Lorenz.ipynb", "R.ipynb", "iris.csv", "lightning.json", and "lorenz.py".
- Code Editor:** Displays the code for the Lorenz system:

```
In [4]: from lorenz import solve_lorenz
t, x_t = solve_lorenz(N=10)
```
- Output View:** Shows sliders for parameters sigma (10.00), beta (2.67), and rho (28.00), and a 3D plot of the Lorenz attractor.
- Code Cell:** Displays the Python code for generating the attractor:

```
def solve_lorenz(N=10, max_time=4.0, sigma=10.0, beta=8./3, rho=28.0):
    """Plot a solution to the Lorenz differential equations."""
    fig = plt.figure()
    ax = fig.add_axes([0, 0, 1, 1], projection='3d')
    ax.axis('off')

    # prepare the axes limits
    ax.set_xlim((-25, 25))
    ax.set_ylim((-35, 35))
    ax.set_zlim((5, 55))

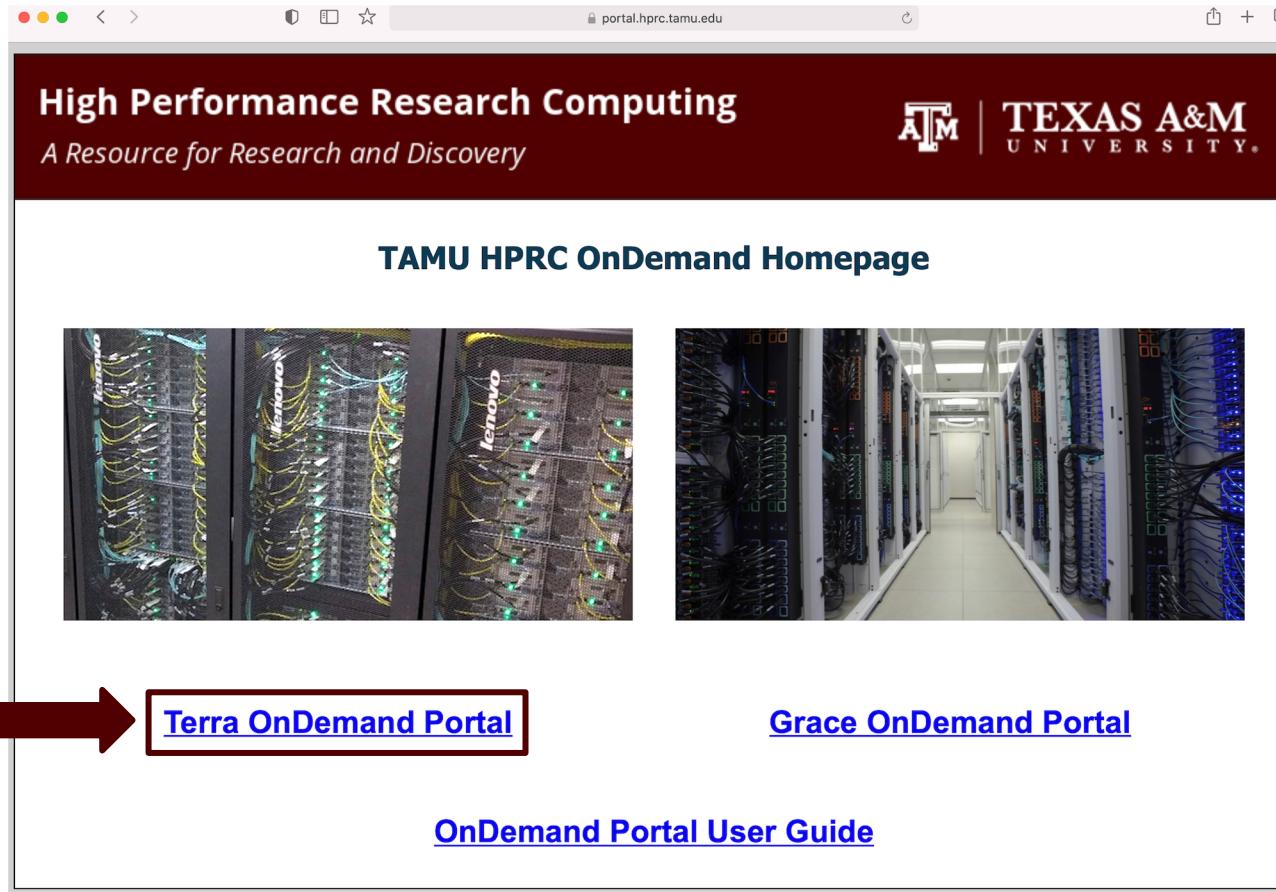
    def lorenz_deriv(x_y_z, t0, sigma=sigma, beta=beta, rho=rho):
        """Compute the time-derivative of a Lorenz system."""
        x, y, z = x_y_z
        return [sigma * (y - x), x * (rho - z) - y, x * y - beta * z]

    # Choose random starting points, uniformly distributed from -15 to 15
    np.random.seed(1)
    x0 = -15 + 30 * np.random.random((N, 3))
```

L1 - Resources

- [Texas A&M High Performance Research Computing \(HPRC\)](#)
- [Terra Quick Start Guide](#)
- [HPRC Portal](#)
- [HPRC YouTube Channel](#)
- [Jupyter Project](#)
- help@hprc.tamu.edu

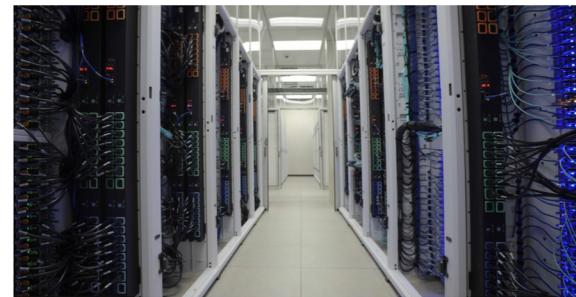
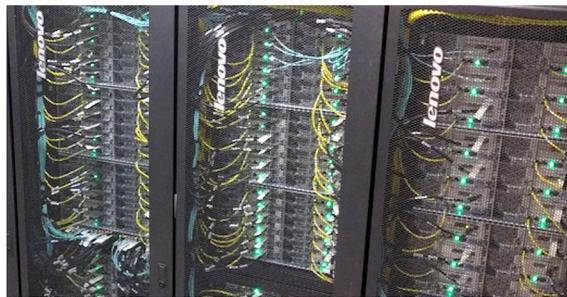
Login HPRC Portal



A screenshot of a web browser displaying the TAMU HPRC OnDemand Homepage. The page has a dark header with the text "High Performance Research Computing" and "A Resource for Research and Discovery". To the right of the header is the Texas A&M University logo. Below the header, the text "TAMU HPRC OnDemand Homepage" is centered. Two photographs of server racks are displayed: one showing three server units with yellow cables and another showing a long corridor between server racks. At the bottom of the page, there are three blue links: "Terra OnDemand Portal" (with a red arrow pointing to it), "Grace OnDemand Portal", and "OnDemand Portal User Guide".

High Performance Research Computing
A Resource for Research and Discovery

TAMU HPRC OnDemand Homepage

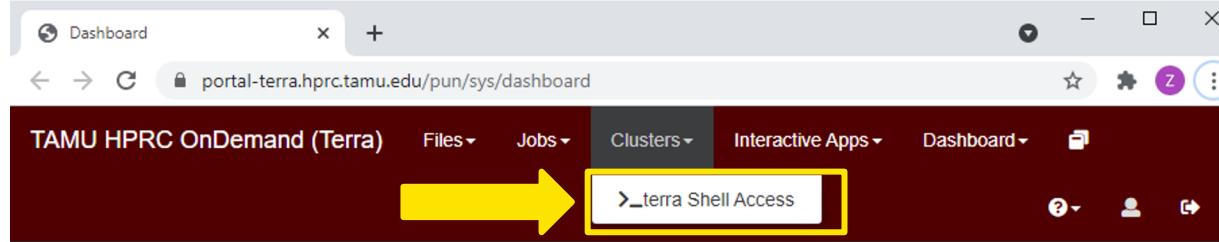


[Terra OnDemand Portal](#)

[Grace OnDemand Portal](#)

[OnDemand Portal User Guide](#)

Shell Access - I



The screenshot shows a browser window for the TAMU HPRC OnDemand (Terra) portal. The URL in the address bar is `portal-terra.hprc.tamu.edu/pun/sys/dashboard`. The navigation bar includes links for Dashboard, Clusters, Interactive Apps, and a highlighted 'Files' section. A yellow arrow points to the '>_terra Shell Access' link under the 'Interactive Apps' dropdown. Below the navigation bar is a photograph of server racks in a data center.



OnDemand provides an integrated, single access point for all of your HPC resources.

Message of the Day

IMPORTANT POLICY INFORMATION

- Unauthorized use of HPRC resources is prohibited and subject to criminal prosecution.
- Use of HPRC resources in violation of United States export control laws and regulations is prohibited. Current HPRC staff members are US citizens and legal residents.
- Sharing HPRC account and password information is in violation of State Law. Any shared accounts will be DISABLED.
- Authorized users must also adhere to ALL policies at: <https://hprc.tamu.edu/policies>

Shell Access - II

A screenshot of a terminal window titled "happidence1@login-0502:~". The window shows a terminal session with the following content:

```
*****  
This computer system and the data herein are available only for authorized  
purposes by authorized users: use for any other purpose is prohibited and may  
result in administrative/disciplinary actions or criminal prosecution against  
the user. Usage may be subject to security testing and monitoring to ensure  
compliance with the policies of Texas A&M University, Texas A&M University  
System, and the State of Texas. There is no expectation of privacy on this  
system except as otherwise provided by applicable privacy laws. Users should  
refer to Texas A&M University Standard Administrative Procedure 29.01.03.M0.02,  
Rules for Responsible Computing, for guidance on the appropriate use of Texas  
A&M University information resources.  
*****  
  
Password:  
Duo two-factor login for happidence1  
  
Enter a passcode or select one of the following options:  
  
1. Duo Push to XXX-XXX-9472  
2. Phone call to XXX-XXX-9472  
3. SMS passcodes to XXX-XXX-9472 (next code starts with: 1)  
  
Passcode or option (1-3): 1  
Success. Logging you in...  
Last login: Sun Jan 31 20:33:32 2021 from 165.91.16.42  
=====| Texas A&M University High Performance Research Computing |  
| Website: https://hprc.tamu.edu |  
| Consulting: help@hprc.tamu.edu (preferred) or (979) 845-0219 |  
| Ada Documentation: https://hprc.tamu.edu/wiki/Ada |  
| Terra Documentation: https://hprc.tamu.edu/wiki/Terra |  
| Grace Documentation: https://hprc.tamu.edu/wiki/Grace |
```

Check out Exercises

The screenshot shows a GitHub repository page for 'happidence1/DataLabs'. The repository is public and contains 1 branch and 0 tags. The 'Code' tab is selected. The repository's purpose is described as 'for 2021 Texas A&M Conference on Energy'. It includes files like 'data', 'images', and Jupyter notebooks ('01_jupyterlab.ipynb', '02_data_exploration_pand...'). The last commit was made 29 minutes ago by user 'f055687'.

```
# git clone (check out) the Jupyter notebooks for the labs  
git clone https://github.com/happidence1/DataLabs.git
```

DataLabs

for 2021 Texas A&M Conference on Energy

Languages

Jupyter Notebook 100.0%

Go to JupyterLab Page

The screenshot shows a web browser window for the TAMU HPRC OnDemand (Terra) dashboard at portal-terra.hprc.tamu.edu/pun/sys/dashboard. The 'Interactive Apps' menu is open, displaying various software options under categories like BIO, GUI, and Imaging. A red arrow points to the 'JupyterLab' option in the 'Servers' section of the main content area.

Dashboard

portal-terra.hprc.tamu.edu/pun/sys/dashboard

TAMU HPRC OnDemand (Terra)

Files ▾ Jobs ▾ Clusters ▾

Interactive Apps ▾ Dashboard ▾

BIO

- Beauti
- DIYABC
- FigTree
- IGV
- JBrowse
- Krait
- Mauve
- Structure
- Tracer

GUI

- ANSYS Workbench
- Abaqus/CAE
- LS-PREPOST
- LS-PREPOST (workshop)
- MATLAB
- Paraview
- VNC

Imaging

- Chimera
- Coot

OnDemand provides an integrated, single access to all your resources.

Message of the Day

IMPORTANT POLICY INFORMATION

- Unauthorized use of HPRC resources is prohibited and subject to disciplinary action.
- Use of HPRC resources in violation of United States export control laws is illegal. Current HPRC staff members are US citizens and legal residents.
- Sharing HPRC account and password information is in violation of University policy and will be DISABLED.
- Authorized users must also adhere to ALL policies at: <https://hprc.tamu.edu/policy/>

Servers

- Jupyter Notebook
- JupyterLab**
- RStudio
- Spark-Jupyter Notebook

https://portal-terra.hprc.tamu.edu/pun/sys/dashboard/batch_connect/sys/chimera/session_contexts/new

Load module

The screenshot shows a web browser window for JupyterLab on the TAMU HPRC OnDemand (Terra) platform. The URL is https://portal-terra.hprc.tamu.edu/pun/sys/dashboard/batch_connect/sys/jupyterlab/session_contexts/new. The page title is "JupyterLab - TAMU HPRC OnD". The main content area is titled "JupyterLab" and describes launching a JupyterLab server on the Terra cluster. It includes a dropdown menu for "Module" set to "Anaconda3/2020.07" (highlighted with a red box and arrow), a section for "Optional Environment to be activated" (empty), and a note about optional conda environments. At the bottom, there's a "Number of hours" input field containing "2" (highlighted with a red box and arrow) and a "Number of cores:" label.

Number of cores: 2

Total memory (GB): 4

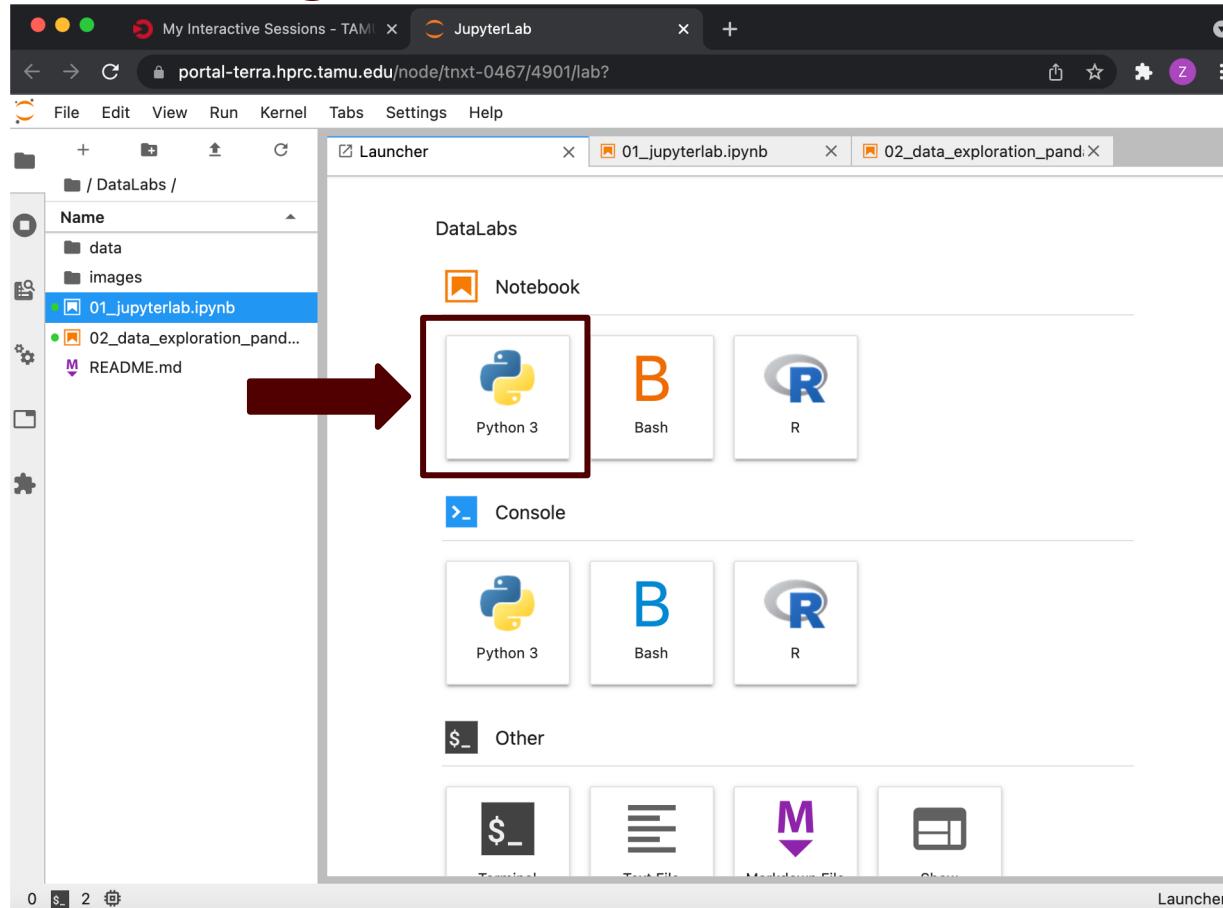
Connect to JupyterLab

The screenshot shows a web browser window for the TAMU HPRC OnDemand (Terra) system. The URL is `portal-terra.hprc.tamu.edu/pun/sys/dashboard/batch_connect/sessions`. The page displays a success message: "Session was successfully deleted." Below this, the "My Interactive Sessions" section is shown. A specific JupyterLab session is listed with the following details:

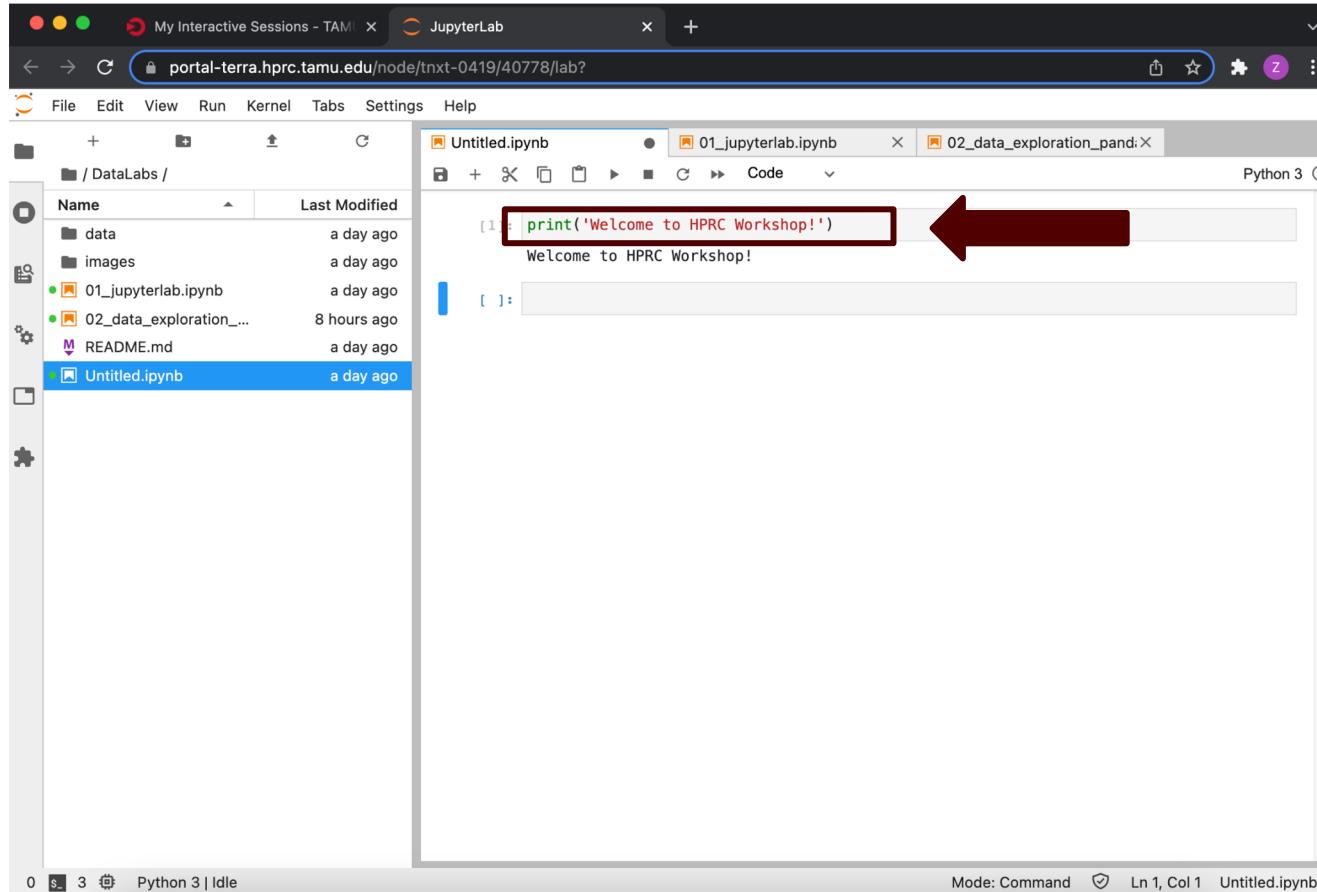
- JupyterLab (9638675)**
- Host:** >_tnxt-0467
- Created at:** 2021-12-01 14:46:42 CST
- Time Remaining:** 1 hour and 57 minutes
- Session ID:** 4150abe8-a7a8-46a1-b228-83f39a373c59
- Actions:** [Delete](#)

A large red arrow points to the "Connect to JupyterLab" button, which is highlighted with a red border.

Create a Jupyter Notebook



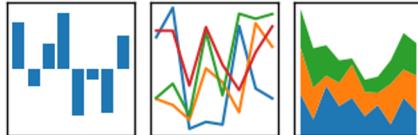
Test JupyterLab



Lab II. Data Exploration

pandas

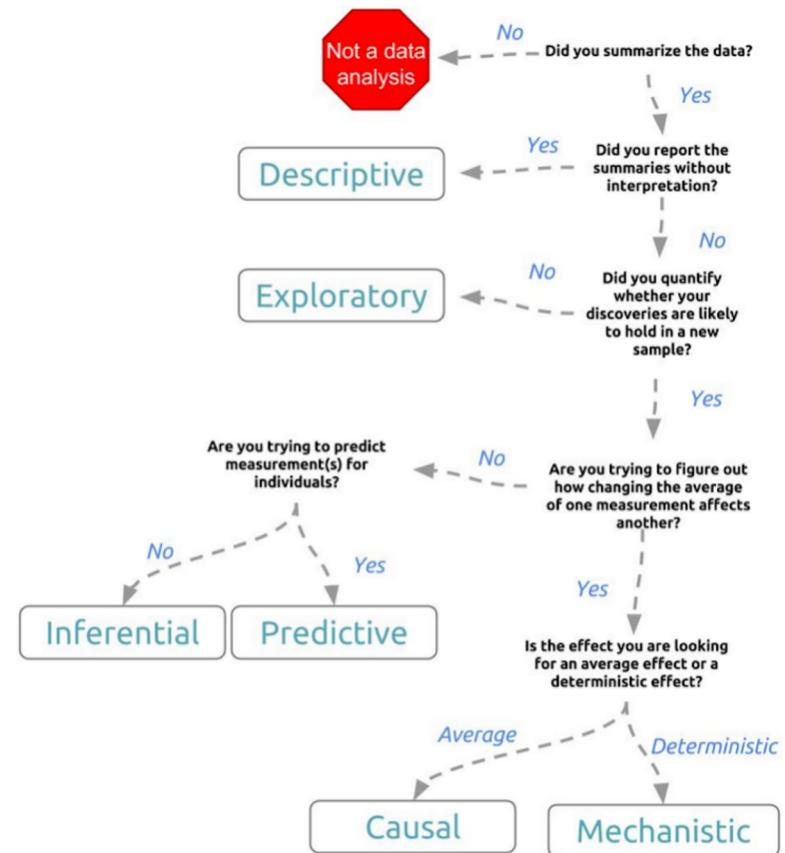
$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$



matplotlib

Types of Data Science Problems

- **Descriptive** (summaries, e.g., census)
 - **Exploratory** (search for unknowns, e.g., four-planet solar system)
 - **Inferential** (find correlations, e.g., many social studies)
 - **Predictive** (make predictions, e.g., Face ID, Echo, Siri)
 - **Causal** (explore causation, e.g., smoking versus lung cancer)
 - **Mechanistic** (determine governing principles, e.g., experimental science)



Credit: Jeff Leek - The Elements of Data Analytic Style

Data Structures

Pandas has two data structures that are descriptive and optimized for data with different dimensions.

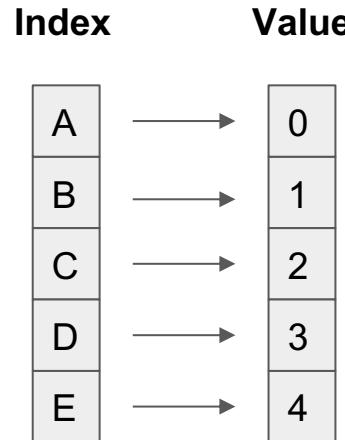
- **Series:** 1D labeled array
- **DataFrame:** General 2D labeled, size-mutable tabular structure with potentially heterogeneously-typed columns

Series in pandas

- One-dimensional labeled array
- Capable of holding any data type (integers, strings, floating point numbers, etc.)
- Example: time-series oil price data



(source: oilprice.com)

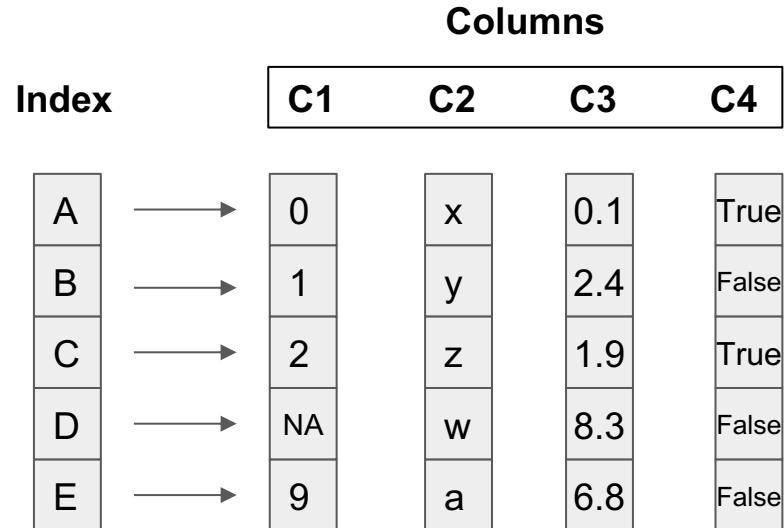


DataFrame in pandas

- Primary Pandas data structure
- A dict-like container for Series objects
- Two-dimensional size-mutable
- Heterogeneous tabular data structure

Top producing oil fields in the United States, 2019* being updated				
Rank	Field	State	Discovery Year	Million Bbl/Day
1	Permian	Texas/New Mexico	1920	4.2
2	Eagle Ford Shale	Texas	2008	1.34
3	Bakken	North Dakota/Montana	1951	1.33
4	Prudhoe Bay Oil Field	Alaska	1967	.791
5	Wattenberg Gas Field	Colorado	1970	.473
6	Shenzi	Federal Gulf of Mexico	2002	.353
7	Kuparuk River oil field	Alaska	1969	.295
8	Midway-Sunset Oil Field	California	1901	.288
9	Atlantis Oil Field	Federal Gulf of Mexico	1998	.273
10	Sugarkane	Texas	2009	.258

(source: [wikipedia](#))



Pandas Learning Objectives

After this lesson, you will know how to:

- Create a DataFrame
- Drop Entries
- Index, Select, and Filter data
- Sort data
- Input and Output

 **JupyterLab Exercises**

Pandas Cheat Sheet

Data Wrangling with pandas Cheat Sheet

<http://pandas.pydata.org>

Syntax – Creating DataFrames

```
df = pd.DataFrame({  
    "a": [4, 5, 6],  
    "b": [7, 8, 9],  
    "c": [10, 11, 12]],  
    index=[1, 2, 3])  
Specify values for each column.  
  
df = pd.DataFrame(  
    [[4, 7, 10],  
     [5, 8, 11],  
     [6, 9, 12]],  
    index=[1, 2, 3],  
    columns=['a', 'b', 'c'])  
Specify values for each row.
```

```
df = pd.DataFrame(  
    {"a": [4, 5, 6],  
     "b": [7, 8, 9],  
     "c": [10, 11, 12]},  
    index=pd.MultiIndex.from_tuples(  
        [('d',1),('d',2),('e',2)],  
        names=['n','v']))  
Create DataFrame with a MultiIndex.
```

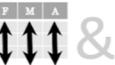
Method Chaining

Most pandas methods return a DataFrame so that another pandas method can be applied to the result. This improves readability of code.

```
df = (pd.melt(df)  
    .rename(columns={  
        'variable': 'var',  
        'value': 'val'})  
    .query('val > 200'))
```

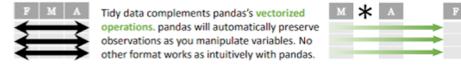
Tidy Data – A foundation for wrangling in pandas

In a tidy data set:



&

Tidy data complements pandas's **vectorized operations**. pandas will automatically preserve observations as you manipulate variables. No other format works as intuitively with pandas.



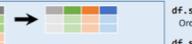
M * A

Reshaping Data – Change the layout of a data set

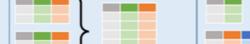
pd.melt(df) Gather columns into rows.



df.pivot(columns='var', values='val') Spread rows into columns.



pd.concat([df1, df2]) Append rows of DataFrames



pd.concat([df1, df2], axis=1) Append columns of DataFrames



Subset Observations (Rows)

df[df.Length > 7] Extract rows that meet logical criteria.

df.drop_duplicates() Remove duplicate rows (only considers columns).

df.head(n) Select first n rows.

df.tail(n) Select last n rows.

Subset Variables (Columns)

df[['width', 'length', 'species']] Select multiple columns with specific names.

df['width'] or df.width Select single column with specific name.

df.filter(regex='regex') Select columns whose name matches regular expression regex.

regex (Regular Expressions) Examples	Description
'.'	Matches strings containing a period.'
'Length\$'	Matches strings ending with word 'Length'
'Sepal'	Matches strings beginning with the word 'Sepal'
'x{1-3}S'	Matches strings beginning with 'x' and ending with 1,2,3,4,5
'^(Species)\$'	Matches strings except the string 'Species'

df.loc[:, 'x2': 'x4'] Select all columns between x2 and x4 (inclusive).

df.iloc[:, 1, 2, 5]] Select columns in positions 1, 2 and 5 (first column is 0).

df.loc[(df['a'] > 10, ['a', 'c'])] Select rows meeting logical condition, and only the specific columns.

Summarize Data

```
df['w'].value_counts()  
Count number of rows with each unique value of variable  
len(df)
```

Handling Missing Data

```
df.dropna()  
Drop rows with any column having NA/null data.  
df.fillna(value)
```

Combine Data Sets



https://pandas.pydata.org/Pandas_Cheat_Sheet.pdf

Key Plotting Concepts in Matplotlib

• Matplotlib: Figure

Figure is the object that keeps the whole image output.

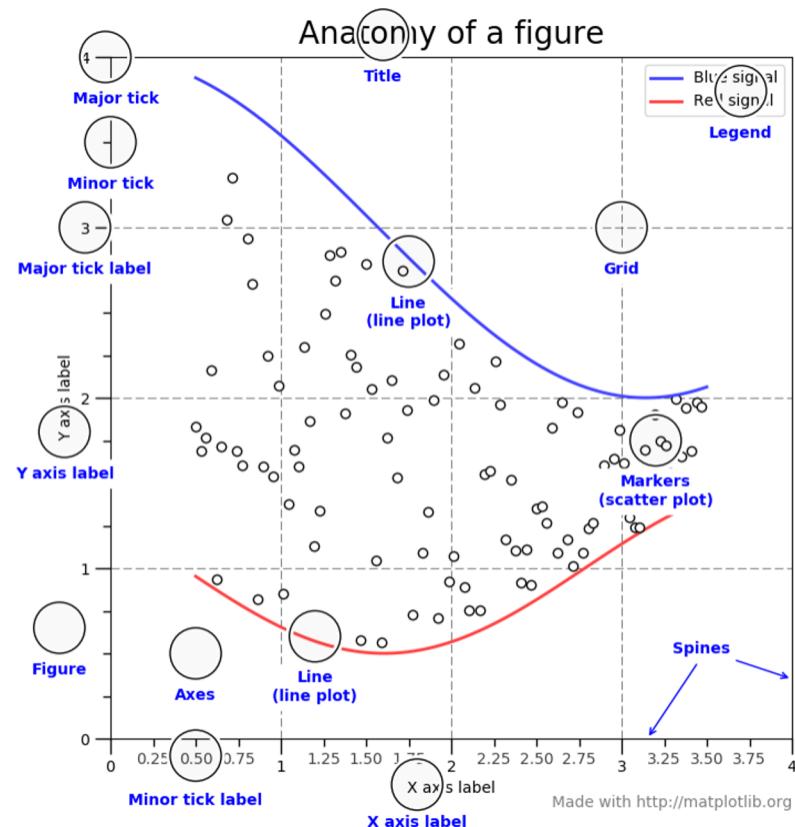
Adjustable parameters include:

1. Image size (`set_size_inches()`)
2. Whether to use `tight_layout` (`set_tight_layout()`)

• Matplotlib: Axes

Axes object represents the pair of axis that contain a single plot (x-axis and y-axis). The Axes object also has more adjustable parameters:

1. The plot frame (`set_frame_on()` or `set_frame_off()`)
2. X-axis and Y-axis limits (`set_xlim()` and `set_ylim()`)
3. X-axis and Y-axis Labels (`set_xlabel()` and `set_ylabel()`)
4. The plot title (`set_title()`)



(Credit: matplotlib.org)

Matplotlib Learning Objectives

After this lesson, you will know how to:

- Scatter plot and Line plot
- Subplots
- Color map
- Contour figures
- 3D figures
 - Surface plots
 - Wire-frame plot
 - Contour plots with projections



JupyterLab Exercises

Matplotlib Cheat Sheet

Python For Data Science Cheat Sheet

Matplotlib

Learn Python Interactively at www.DataCamp.com



Matplotlib
Matplotlib is a Python 2D plotting library which produces publication-quality figures in a variety of hardcopy formats and interactive environments across platforms.

1 Prepare The Data Also see Lists & NumPy

1D Data

```
>>> import numpy as np  
>>> x = np.linspace(0, 10)  
>>> y = np.cos(x)  
>>> z = np.sin(x)
```

2D Data or Images

```
>>> data = 2 * np.random.random((10, 10))  
>>> data[3] = 3 * np.random.random((10, 10))  
>>> data[3][3] = 1000  
>>> U = -1 + x**2 + y**2  
>>> V = 1 + x**2 - y**2  
>>> from matplotlib import pyplot as plt  
>>> img = np.loadtxt('sample_data/asm_grid/bivariate_normal.npy')
```

2 Create Plot

Figure

```
>>> fig = plt.figure()  
>>> fig2 = plt.figure(figsize=plt.rcParams['figure.figsize'])
```

Axes

All plotting is done with respect to an Axes. In most cases, a subplot will fit your needs. A subplot is an axes on a grid system.

```
>>> fig.add_axes()  
>>> ax1 = fig.add_subplot(221) # row-col-num  
>>> ax2 = fig.add_subplot(222)  
>>> fig3, axes = plt.subplots(nrows=2, ncols=2)  
>>> fig4, axes2 = plt.subplots(ncols=3)
```

3 Plotting Routines

1D Data

```
>>> lines = ax.plot(x,y)  
>>> ax.scatter(x,y)  
>>> ax.step([1,2,3],[3,4,5])  
>>> ax.pcolor([[0,1,2,3],[4,5,6,7]])  
>>> ax.pcolor([0,1,2,3],[4,5,6,7],cmap='hot')  
>>> ax.pcolor([0,1,2,3],[4,5,6,7],cmap='hot', interpolation='nearest')  
>>> ax.pcolor([0,1,2,3],[4,5,6,7],cmap='hot', interpolation='nearest', vmin=-2, vmax=21)  
>>> ax.fill_between(x,y,color='yellow')
```

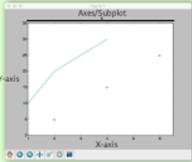
2D Data or Images

```
>>> fig, ax = plt.subplots()  
>>> im = ax.imshow(lsqg,  
>>> cmap='gist_earth',  
>>> interpolation='nearest',  
>>> vmin=-2,  
>>> vmax=21)
```

Colormapped or RGB arrays

Plot Anatomy & Workflow

Plot Anatomy



Figure

Workflow

The basic steps to creating plots with matplotlib are:

- 1 Prepare data
- 2 Create plot
- 3 Plot
- 4 Customize plot
- 5 Save plot
- 6 Show plot

```
>>> import matplotlib.pyplot as plt  
>>> x = [1,2,3,4]  
>>> y = [1,4,9,16]  
>>> fig = plt.figure() Step 1  
>>> ax = fig.add_subplot(111) Step 2  
>>> ax.plot(x, y, color='lightblue', linewidth=3) Step 3  
>>> ax.scatter([4,6,8],  
>>> [5,10,15],  
>>> color='darkgreen',  
>>> marker='*')  
>>> ax.set_xlim(1, 6.5)  
>>> plt.savefig('foo.png') Step 4  
>>> plt.show() Step 6
```

4 Customize Plot

Colors, Color Bars & Color Maps

```
>>> plt.plot(x, x, x**2, x**3)  
>>> ax.plot(x, y, alpha=0.4)  
>>> ax.set_color_cycle(['red','blue'])  
>>> fig.colorbar(im, orientation='horizontal')  
>>> im = ax.imshow(img,  
>>> cmap='seismic')
```

Markers

```
>>> fig, ax = plt.subplots()  
>>> ax.plot(x,y,marker='o')  
>>> ax.plot(x,y,marker='*')
```

Line Styles

```
>>> plt.plot(x,y,linewidth=4.0)  
>>> plt.plot(x,y,ls='solid')  
>>> plt.plot(x,y,ls='dashed')  
>>> plt.plot(x,y,'--',x**2,y**2,'-')  
>>> plt.setp(lines,color='r',linewidth=4.0)
```

Text & Annotations

```
>>> ax.text(2,1,  
>>> "Example Graph",  
>>> style='italic')  
>>> ax.annotate("Sine",  
>>> xy=(0, 1),  
>>> xycoords='data',  
>>> xytext=(10.5, 0),  
>>> textcoords='offset pixels',  
>>> arrowprops=dict(arrowstyle=">-",  
>>> connectionstyle="arc3",
```

Vector Fields

```
>>> axes[0,1].arrow(0,0,0.5,0.5)  
>>> axes[1,1].quiver(x,y)  
>>> axes[0,1].streamplot(X,Y,U,V)
```

Data Distributions

```
>>> ax1.hist(y)  
>>> ax1.violinplot(y)  
>>> ax3.violinplot(z)
```

5 Save Plot

Save Figures

```
>>> plt.savefig('foo.png')  
>>> plt.savefig('foo.png', transparent=True)
```

6 Show Plot

```
>>> plt.show()
```

Close & Clear

Close

```
>>> plt.close()  
>>> plt.clf()  
>>> plt.cla()
```

Clear

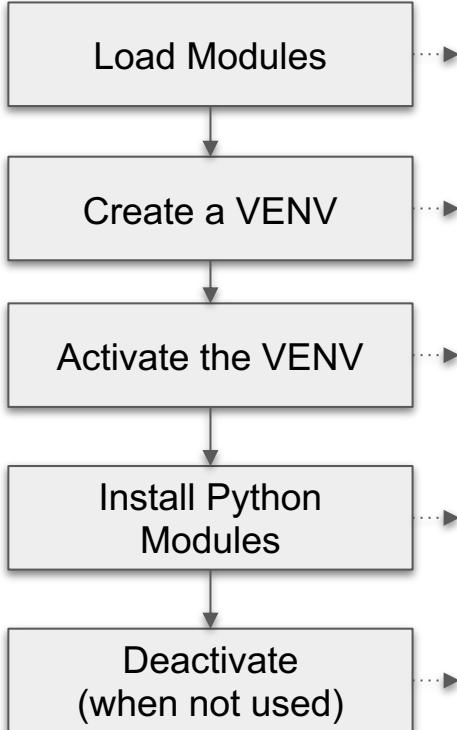
```
>>> clear axis  
>>> clear figure  
>>> clear window
```

DataCamp
Learn Python For Data Science Interactively

https://s3.amazonaws.com/assets.datacamp.com/blog_assets/Python_Matplotlib_Cheat_Sheet.pdf

Additional Materials

Python Virtual Environment (VENV)



```
# Change to /scratch/user/netid directory
cd $SCRATCH

# clean up and load Anaconda
module purge
module load Anaconda/3-5.0.0.1

# create a Python virtual environment
conda create -n mylab

# activate the virtual environment
source activate mylab

# install required package to be used in the portal
conda install -c conda-forge jupyterlab=1.2.2
conda install pandas matplotlib
conda install scikit-learn
conda install tensorflow

# deactivate the virtual environment
# source deactivate
```

Common Anaconda Commands

```
# Conda virtual environment
conda info                                # show Conda installation
conda create -n VENV                      # create a virtual environment
conda create -n VENV python=3.4             # create a venv with a py version
conda env list                            # list installed venv

# Conda package management
conda list                                 # list all installed packages
conda search PACKAGENAME                  # search a Conda package
conda install PACKAGENAME                 # install a Conda package
conda update PACKAGENAME                  # update a Conda package
conda remove PACKAGENAME                  # remove a Conda package
```