BTEP class

Alexandra L Emmons Ph.D. & Joe Wu Ph.D.
BTEP/GAU/CCR/NCI/NIH - email ncibtep@mail.nih.gov
Bioinformatics Training and Education Program

Table of Contents

Class Overview

Class Slides

Documenting Data Analysis with Jupyter Lab

Class Data

Class Overview

Keeping track of data analysis steps is as important as laboratory experiment procedures. These processes ensure that researchers can retrace steps during trouble shooting, enable scientists to share procedures with their peers and promote reproducibility of results in research. Open source bioinformatics software require the use of code or command line. For instance, RNA sequencing analysis packages such as DESeq2 and edgeR require knowledge of R. Biopython, enables tasks such as sequence manipulation and querying of NCBI databases requires knowledge of Python. This class will introduce participants to Jupyter Lab, an open source tool that allows investigators to keep analysis steps, code, and output in one place. It is compatible with languages used in bioinformatics such as R, Python, and Bash but can also be used for other languages such as Julia and C/C++.

Disclaimer

This class will not make participants experts in using Jupyter Lab or scripting. Also, this class is a demo, experience using or installation onto personal computer of Jupyter Lab is not required to participate. BTEP staff will be happy to meet with participants to discuss installing or accessing Jupyter Lab after the class, as they are many options. Just email us at ncibtep@nih.gov if you need help! Further, do not worry about the coding part of this class but focus more on what Jupyter Lab can do in terms of organizing analysis steps.

Learning Objectives

After this class, participants will be able to

- Have obtained an appreciation for Jupyter Lab as an one-stop shop for organizing analysis steps, code, and output.
- Be aware of or able to describe the steps involved in using Jupyter Lab for documenting data analysis including:
    - Initiating a new Jupyter Lab session.
    - Navigating through and transferring data into the file explorer.
    - Starting a language specific notebook.
    - Writing formatted text and code.

Class Slides

Documenting Data Analysis with Jupyter Lab

Joe Wu, Phd

NCI/CCR Bioinformatics Training and Education Program

ncibtep@nih.gov

September 24, 2024

Start Jupyter Lab

To start Jupyter Lab, type the following into the command prompt. The `--no-browser` option prevents a web browser from opening.

```
jupyter lab --no-browser
```

Copy and paste any of the following URLs into a web browser to start using Jupyter. These URLs will be different for every Jupyter Lab session. Note users cannot copy one of the URLs below into a web browser on their computer and hope that it pulls up Jupyter Lab. These URLs are machine specific and Jupyter Lab also needs to be installed.

```
To access the server, open this file in a browser:
        file:///Users/wuz8/Library/Jupyter/runtime/jpserver-30985-ope
    Or copy and paste one of these URLs:
        http://localhost:8890/lab?token=1952fcce201164f8368f2666f2f2(
        http://127.0.0.1:8890/lab?token=1952fcce201164f8368f2666f2f2(
```

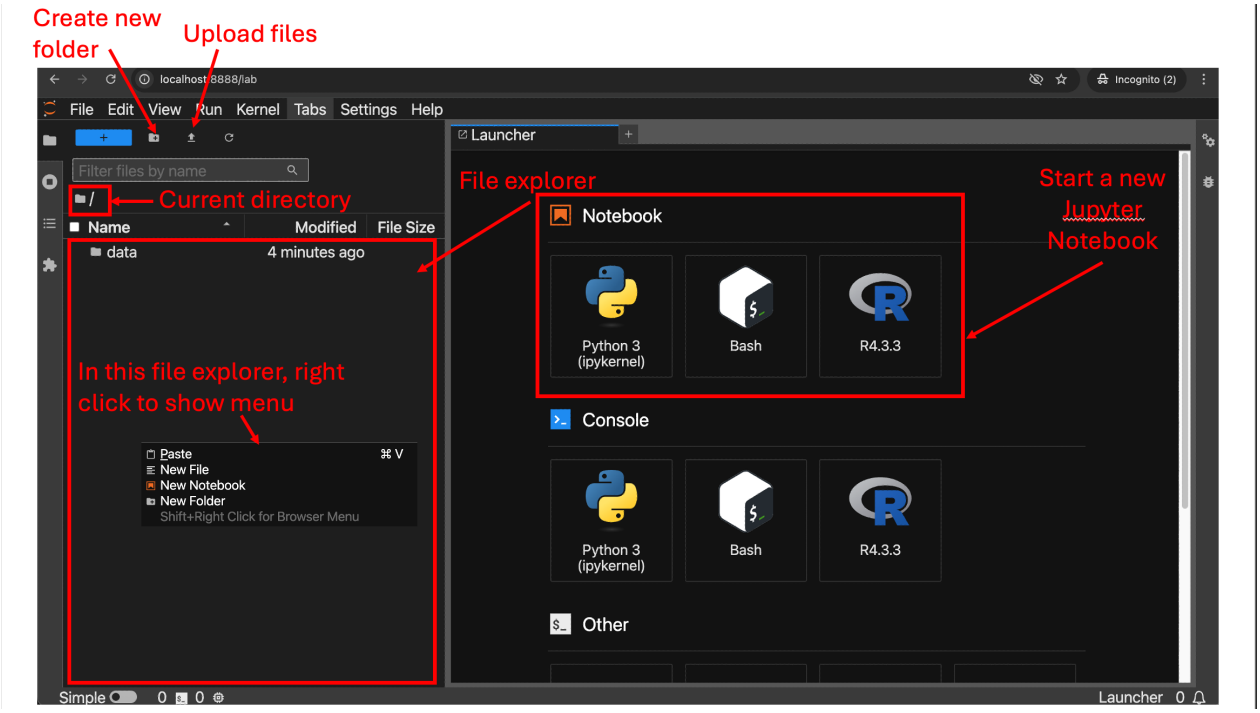
Jupyter Lab Interface

Upon starting Jupyter Lab, users will see an interface with the following.

- To the left, there is a file explorer where users can access files and subfolders available in the directory in which the Jupyter Lab session was started (also referred to as the project directory in these documents). In the example below, there is a subfolder called "data".
- A folder icon with a "+" enclosed can be used to create new folders within the project directory. These will be available in the file explorer.

- In case there are files or folders that are not in the project directory, users can use the upload button to transfer these so that they will appear in the file explorer. An alternative is to click and drag content into the file explorer.
- The Launcher contains options for starting a Jupyter Notebook in various languages. Users will have to install the specific language and add it into their Jupyter Lab environment to be able to use it. The link addressing Jupyter Lab language compatibility is included in the section titled "Jupyter Lab is Compatible with Many Languages".
- Right clicking on the file explorer will bring up a menu where users can create new files, folders, and Jupyter Notebooks.

Caution: The file explorer will only display files and subfolders in the directory in which the Jupyter Lab session was created. To transfer files that reside in a different directory, either click and drag into the file explorer or use the upload button. Alternatively, users may create a symbolic link, which will appear in the file explorer to access content outside of the Jupyter Lab session directory.



Jupyter Lab is Compatible with Many Languages

- Bash, Python, and R
- See https://docs.jupyter.org/en/latest/projects/kernels.html *(https://docs.jupyter.org/en/latest/projects/kernels.html)* for a list of Jupyter compatible languages.

Start a New Jupyter Notebook

To start a new Jupyter Notebook, click on any of the available panels under the "Notebook" section in the Launcher. Each of the panels represent a notebook that will work with a specific language that was added to the Jupyter environment by the user.

This class will use a Python notebook as an example. Click on the "Python 3 (ipykernel)" panel to start a Python Jupyter Noteook. The instructor has Python 3.12.2 installed and this will be differ depeneding on what each user has installed. Remember to keep track of language and package versions to ensure analysis reproducibility.

The Jupyter Notebook is a part of Jupyter Lab.

The note book is where users

- Write code
- View and maintain output
- Document analysis steps using formatted text written in markdown

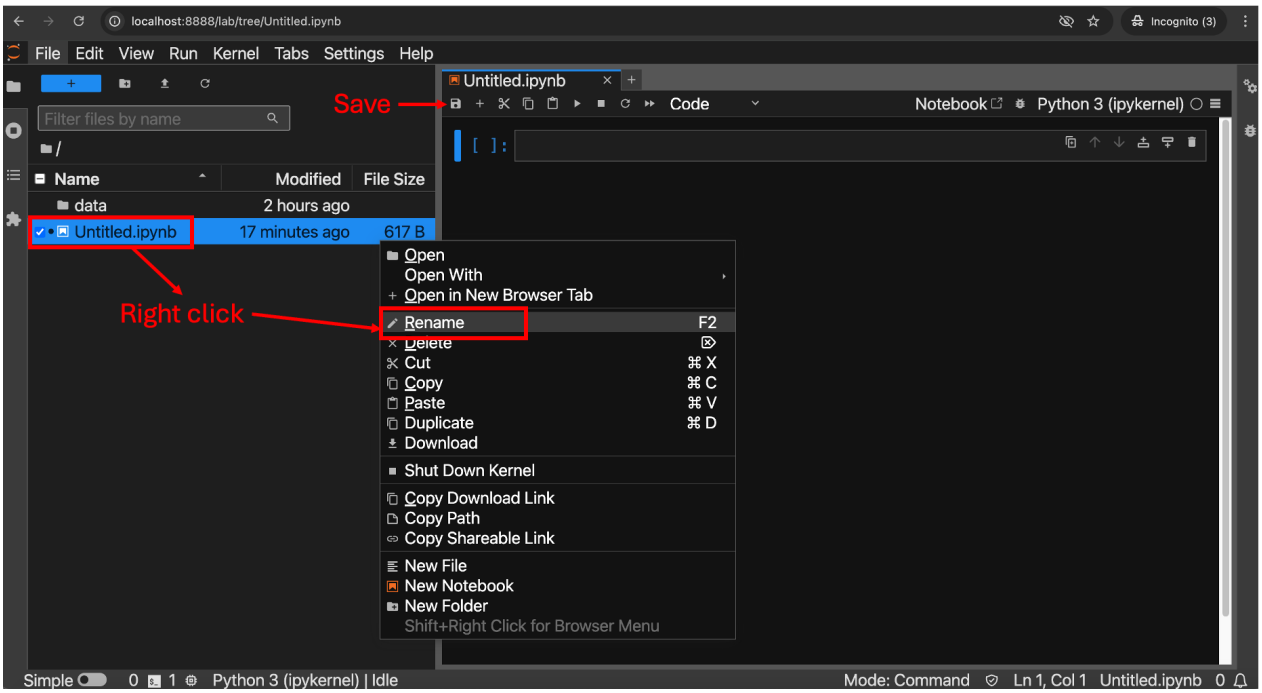
Note:

"Markdown is a lightweight markup language for creating formatted text using a plain-text editor." -- https://en.wikipedia.org/wiki/Markdown *(https://en.wikipedia.org/wiki/Markdown)*

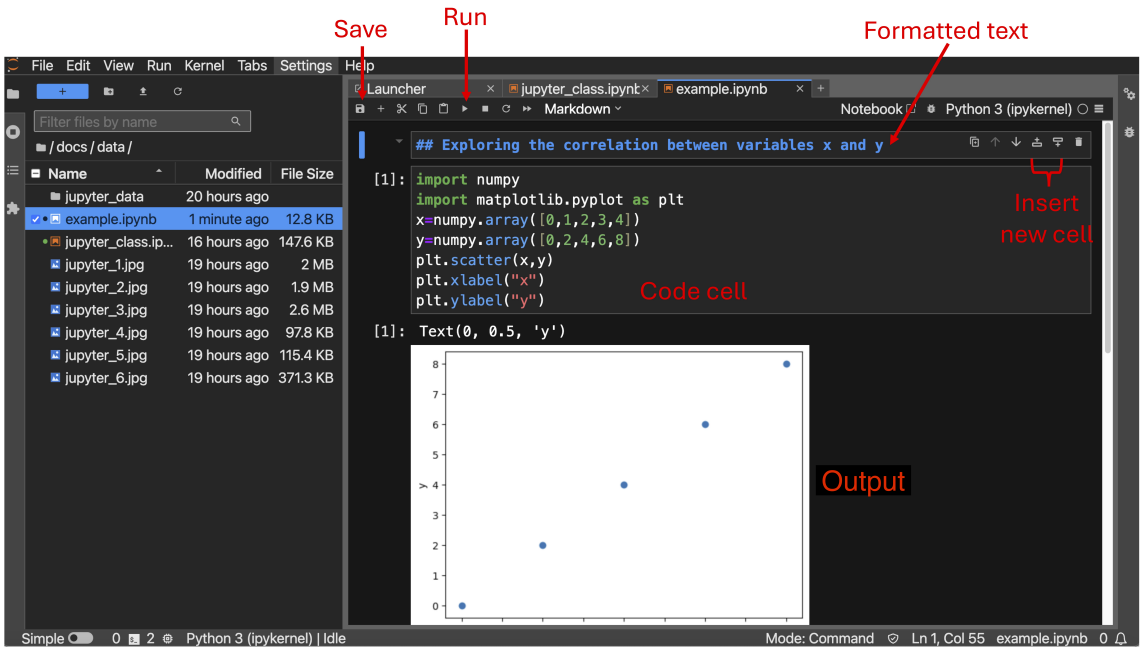
Rename Untitled Jupyter Notebook

A new Jupyter Notebook is given the name "Untitled". Change this to something meaningful either using the save icon on the notebook menu bar or right-clicking on the "Untitled.ipynb" notebook in the file explorer and choose "Rename".

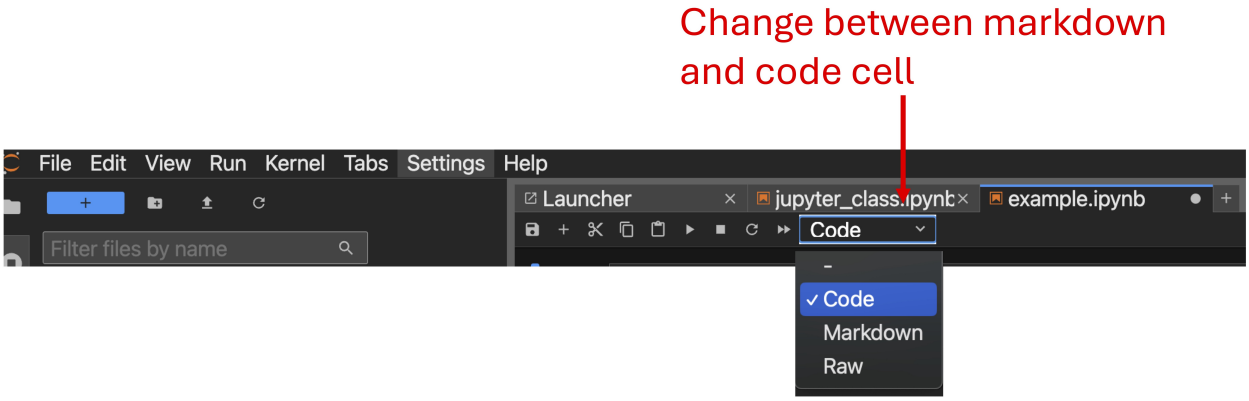
Jupyter Notebooks have extension ipynb, which stands for interactive Python notebook.



Keeping Code, Output, and Analysis Steps in One Place



Changing Between Markdown and Code



Ways to Access and Use Jupyter Lab

- Biowulf (https://hpc.nih.gov/apps/jupyter.html *(https://hpc.nih.gov/apps/jupyter.html)*). Biowulf is the NIH Unix-based high performance computing system.
  - Requires getting a Biowulf account *(https://hpc.nih.gov/docs/accounts.html)*
  - Jupyter is available through Biowulf's Open On Demand *(https://hpcondemand.nih.gov/)*. Open On Demand enables users to run interactive applications such as Jupyter Lab through a web browser.
- Google Colab *(https://colab.research.google.com/)*

Installing Jupyter Lab on Personal Computer

To run Jupyter Lab on a personal computer, users will need to have

- Python installed
  - Jupyter Lab is installed using either package managers `anaconda`, `mini conda` or `mamba` (these install Python by default) or `pip`, which is the package installer for Python.
- Jupyter Lab installed, either through
  - Anaconda *(https://www.anaconda.com/)*: Jupyter Lab is packaged with this.
  - Install package managers miniconda *(https://docs.anaconda.com/miniconda/)* or mamba *(https://mamba.readthedocs.io/en/latest/index.html)*. Both package manager come with Python.
    - Install Jupyter Lab following the instructions at https://jupyterlab.readthedocs.io/en/stable/getting_started/installation.html *(https://jupyterlab.readthedocs.io/en/stable/getting_started/installation.html)*.
  - As a stand alone without using package managers:
    - https://www.python.org/downloads/ *(https://www.python.org/downloads/)* for installing Python.
    - https://jupyterlab.readthedocs.io/en/stable/getting_started/installation.html *(https://jupyterlab.readthedocs.io/en/stable/getting_started/installation.html)* for installing Jupyter Lab.
- Additional languages and language specific packages installed
- Add languages to Jupyter Lab environment (Python is added automatically)

> **Tip**
>
> Jupyter Lab is compatible with many languages. For more information about Jupyter compatible languages see https://docs.jupyter.org/en/latest/projects/kernels.html *(https://docs.jupyter.org/en/latest/projects/kernels.html)*.

Code and Visualization

Jupyter Notebook enables organization of code, analysis steps, and output all in one place.

The example below will use Python code to demonstrate the utility of Jupyter Notebook for keeping code, output, and analysis steps all in one place.

- First, the `import` command of Python will be used to load packages `numpy` which is used for numeric computation and `matplotlib` which is a popular Python data visualization tool.
- Next, two numeric arrays (x and y) will be generated using `numpy` and its `array` sub-command. To call a Python sub-command just type the package name (ie. numpy),

followed by a ".", and then the sub-command. So to create an array, use `numpy.array`. The elements in the array are enclosed in "([])".

- Finally, a scatter plot will be generated using `matplotlib's pyplot` module, which will be loaded under the alias `plt` for easy referencing in the code. The `scatter` sub-command will be used to generate the scatter plot showing the relationship between the values in array x versus those in array y. The arguments enclosed in "()" are name of the two arrays separated by a comma (ie. (x,y)).
- After scatter plot has been generated, the following sub-commands from `matplotlib's pyplot` or `plt` will be used to customize the graph:
    - `xlabel`: assign label to the x axis, the option `size=` allows font size adjustment.
    - `ylabel`: assign label to the y axis, the option `size=` allows font size adjustment.
    - `plt.tick_params`: adjust axis tick mark label font size by setting the `labelsize` option.
    - `plt.title`: assigns a plot title and `size` is used to adjust the fontsize of the title.

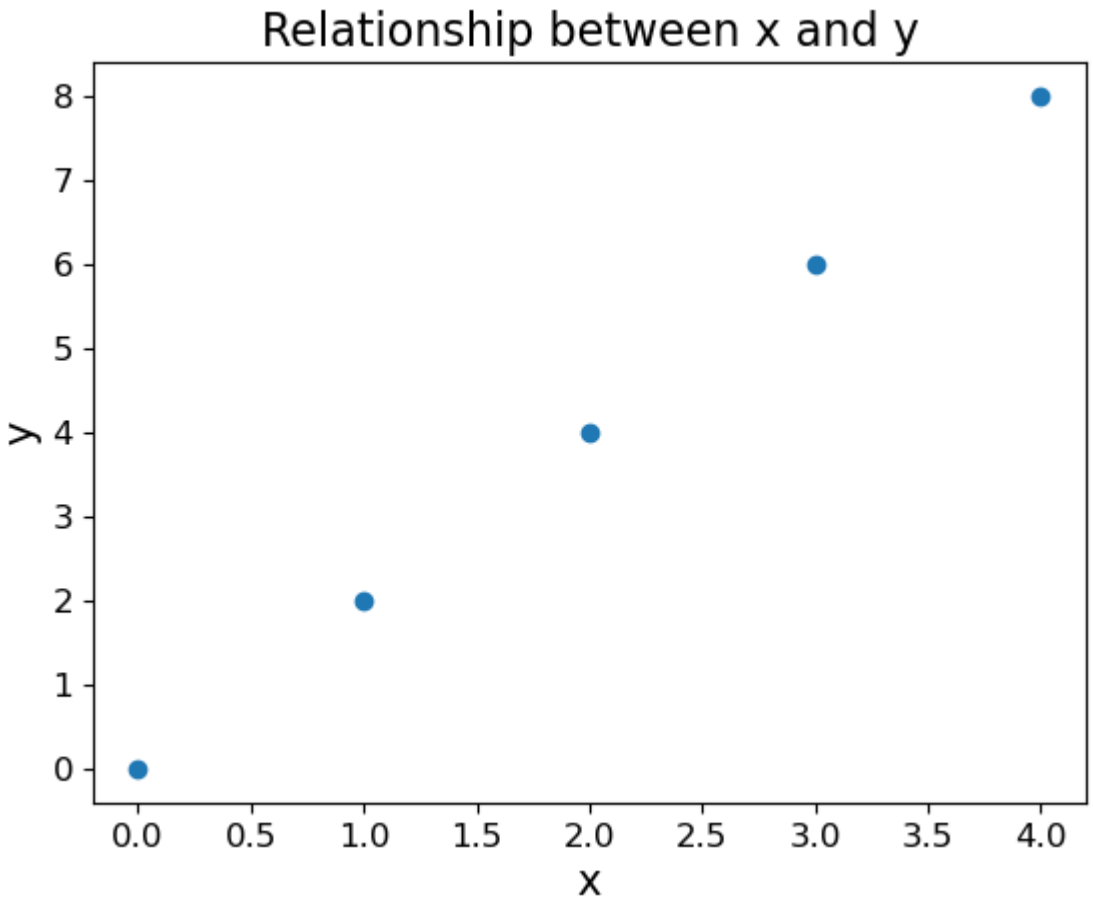
```
# Import some Python packages
import numpy
import matplotlib.pyplot as plt
```

```
# Generate numeric arrays x and y using numpy
x=numpy.array([0,1,2,3,4])
y=numpy.array([0,2,4,6,8])
print("The array x contains: ", x)
print("The array y contains: ", y)
```

```
The array x contains:  [0 1 2 3 4]
The array y contains:  [0 2 4 6 8]
```

```
# Generate scatter plot using matplotlib
plt.scatter(x,y)
plt.xlabel("x", size=15)
plt.ylabel("y", size=15)
plt.tick_params(labelsize=12)
plt.title("Relationship between x and y", size=16)
```

```
Text(0.5, 1.0, 'Relationship between x and y')
```



Importing data tables into Jupyter Notebook

Users will be working with data tables during analyses. These can be imported into the a Jupyter Notebook although the approach for doing so will be language dependent.

The example below will use a Python package called Pandas *(https://pandas.pydata.org)* to import a data table. The data table contains the differential gene expression analysis results from the HBR and UHR study *(https://rnabio.org/module-01-inputs/0001/05/01/RNAseq_Data/)* and is in the comma separated value or CSV format where each column in the table is separated by a comma.

The file used is hbr_uhr_deg_chr22_with_significance.csv.

Step 1 in the process is to load `pandas` into the work environment using the `import` command of Python (ie. `import pandas`). The data will be stored in a data table called hbr_uhr_deg_chr22. Use `pandas.read_csv` to call the `read_csv` sub-command. Enclosed in parentheses is the path to the file hbr_uhr_deg_chr22_with_significance.csv. Once the data has been loaded, append the `.head()` attribute to the hbr_uhr_deg_chr22 to preview the first five lines.

The hbr_uhr_deg_chr22 data table contains five columns and these are:

- name: contains gene names

- log2FoldChange: gene expression change between two experimental conditions on the log2 scale
- PAdj: adjusted p-value indicating statistical confidence of the calculated gene expression change
- -log10PAj: negative of the values in the PAdj column on a log10 scale
- significance: whether the gene expression is up regulated, down regulated, or has no change

```
# Load the Pandas package
import pandas
```

```
# Import the data

hbr_uhr_deg_chr22=pandas.read_csv("./hbr_uhr_deg_chr22_with_significa

# View the first several lines of hbr_uhr_deg_chr22 using the head a
## of the imported data table (hbr_uhr_deg_chr22)
hbr_uhr_deg_chr22.head()
```

	name	log2FoldChange	PAdj	-log10PAdj	significance
0	SYNGR1	-4.6	5.200000e-217	216.283997	down
1	SEPT3	-4.6	4.500000e-204	203.346787	down
2	YWHAH	-2.5	4.700000e-191	190.327902	down
3	RPL3	1.7	5.400000e-134	133.267606	ns
4	PI4KA	-2.0	2.900000e-118	117.537602	down

Data visualization in Jupyter Notebook: scatter plot

The exercise below will use the Python package Seaborn *(https://seaborn.pydata.org)* to create a volcano plot for the HBR and UHR differential analysis results table that was previously imported. A volcano plot is essentially a scatter plot that shows magnitude of change on one axis and statistical confidence of the change on another axis. In the case of RNA sequencing differential expression, this will be log2 of fold change and -log10 of adjusted p-values.

The `scatterplot` function in seaborn will be used to generate volcano plots. This function takes on arguments below:

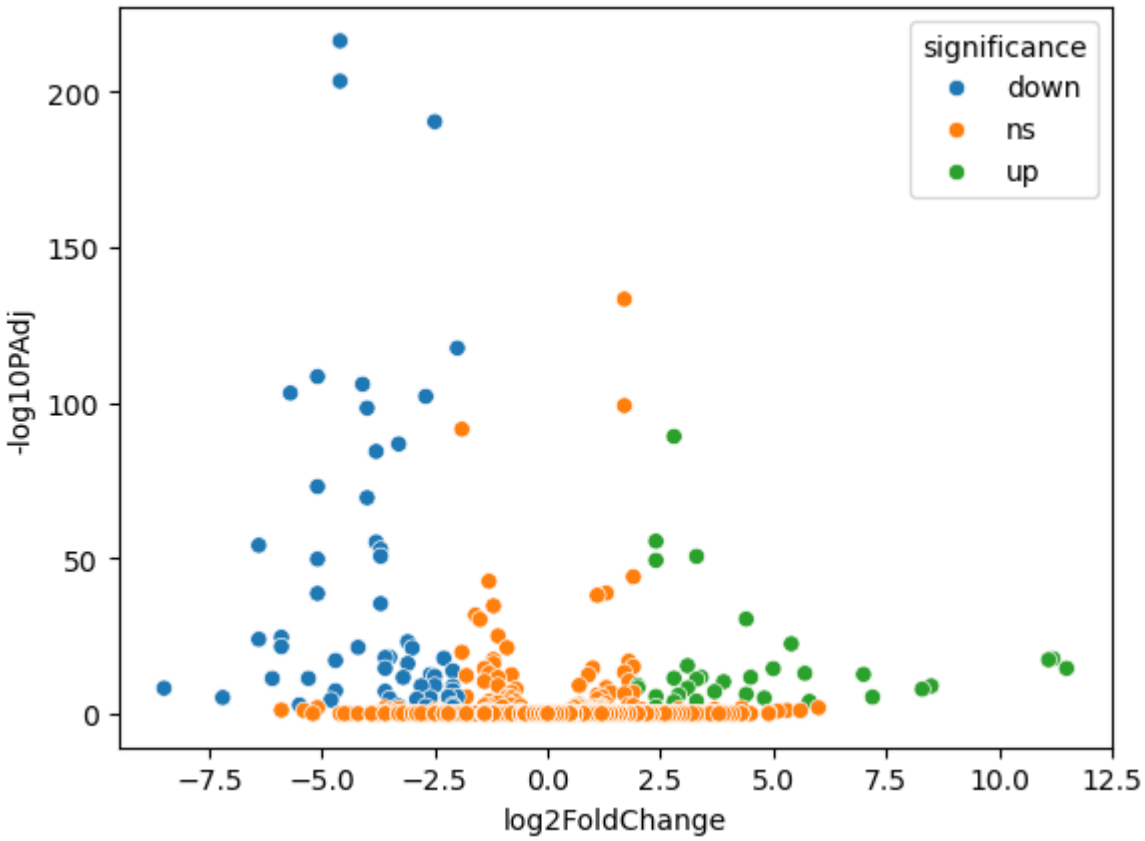
- Data: hbr_uhr_deg_chr22 (differential gene expression analysis results table imported using `pandas.read_csv`)
- `x`: x-axis values (ie. gene expression log2FoldChange)
- `y`: y-axis values (ie. -log10 of adjusted p-value)

- `hue`: color dots by whether gene expression change is up, down, or has no change (see signifcance column of the data)

```
## Load the seaborn plotting package
import seaborn

seaborn.scatterplot(hbr_uhr_deg_chr22,x="log2FoldChange", y="-log10PA
```

```
<Axes: xlabel='log2FoldChange', ylabel='-log10PAdj'>
```



Data visualization in Jupyter Notebook: heatmap

This exercise will use Seaborn's `clustermap` function to construct a gene expression heatmap of top differentially expressed genes in the HBR and UHR study. Heatmaps are another common visualization in RNA sequencing and allow scientists to identify clusters of samples with similar gene expression patterns.

First, import the dataset (hbr_uhr_top_deg_normalized_counts.csv) using `pandas.read_csv` and assign it to the variable hbr_uhr_top_deg_normalized_counts. The `clustermap` function of seaborn takes the following arguments and options.

- Data: hbr_uhr_top_deg_normalized_counts (the expression counts table imported using `pandas.read_csv`).

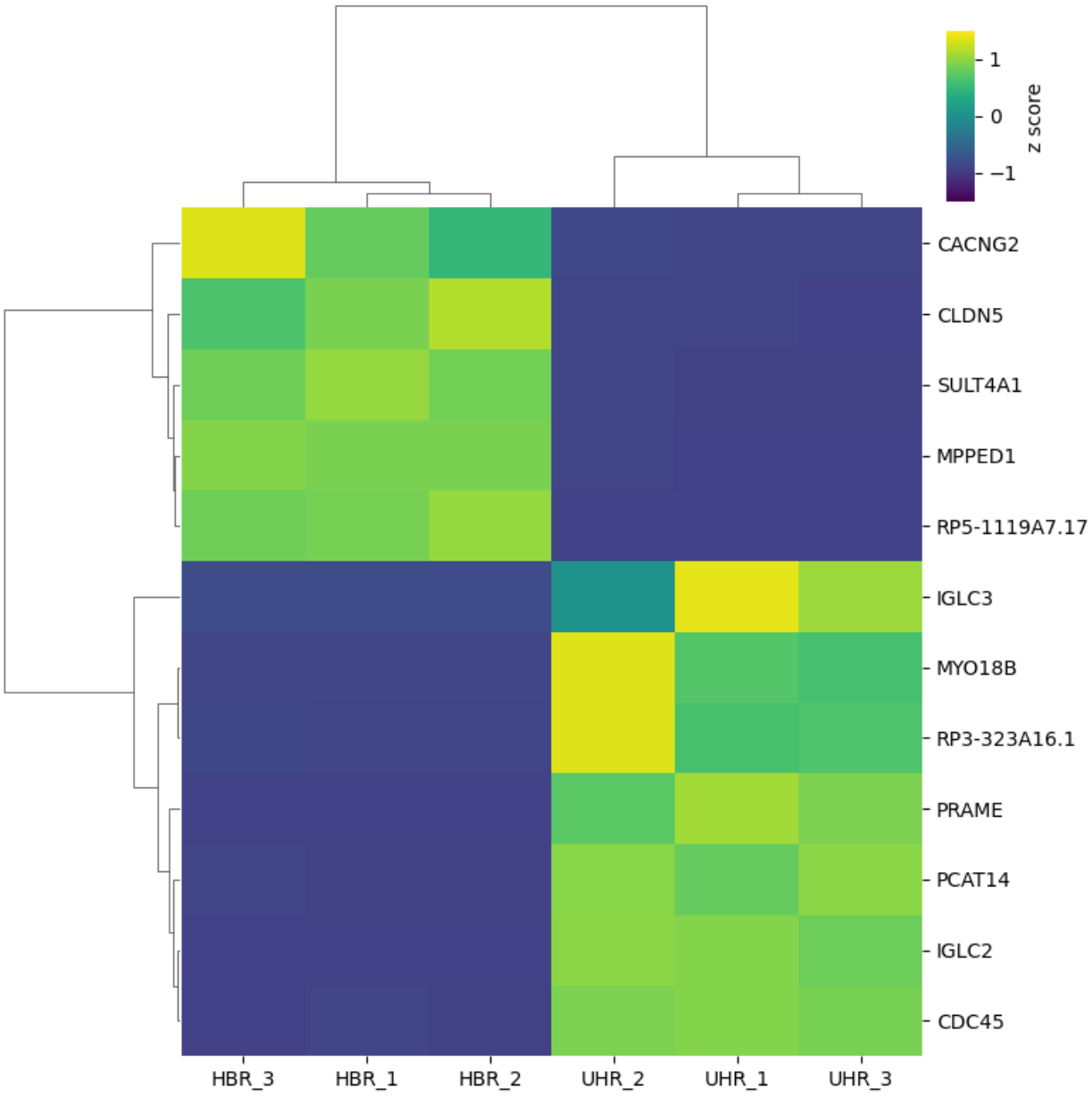
- `z_score`: set this to 0 to scale the expression counts for each gene by the z_score.
- `cmap`: specify a coloring scheme (ie. viridis)
- `figsize`: specify figure size
- `cbar_kws`: specify title for the heatmap color bar using a key-value pair
- `cbar_pos`: specify coordinate to place the heatmap color bar

```
# Import the data

hbr_uhr_top_deg_normalized_counts=pandas.read_csv("./hbr_uhr_top_deg_
```

```
seaborn.clustermap(hbr_uhr_top_deg_normalized_counts,z_score=0,cmap='
                   figsize=(8,8),vmin=-1.5, vmax=1.5,cbar_kws={"labe
                   cbar_pos=(0.855,0.8,0.025,0.15))
```

```
<seaborn.matrix.ClusterGrid at 0x137331340>
```



R Code in a Python Jupyter Notebook

Using the `[rpy2.ipython](https://rpy2.github.io)` package, users can run R code inside a Python Jupyter Notebook. This package requires user installation (see https://pypi.org/project/rpy2/ *(https://pypi.org/project/rpy2/)*). To load `rpy2.ipython`, use `%load_ext`

```
# Load rpy2.ipython
%load_ext rpy2.ipython
```

Using R to generate a volcano plot

Here, R will be used to generate a volcano plot for the HBR and UHR study differential expression analysis results as was done using Python. To run R code in a Python Jupyter Notebook, place at the top of the code cell `%%R` then proceed to write the R code.

R's `read.csv` function takes on a file path enlosed in "()" as an argument. There are many options for this command that could alter the default ways in which it imports data. Here, the file hbr_uhr_deg_chr22_with_significance.csv will be imported using `read.csv` with default settings and stored as hbr_uhr_chr22_deg. The `head` command will be used to preview the first six lines of the data table. Because the "-" sign in the column name `-log10PAdj` violates R's column heading naming convention, `read.csv` by default changes the "-" with "X.". This column will be renamed `neg.log10PAdj`.

```
%%R
# Import gene expression data using read.csv and store is as variable
hbr_uhr_chr22_deg <- read.csv("./hbr_uhr_deg_chr22_with_significance
```

```
%%R
# Look at the first few lines of counts using the head command
head(hbr_uhr_chr22_deg)
```

```
    name log2FoldChange     PAdj X.log10PAdj significance
1 SYNGR1           -4.6 5.2e-217    216.2840         down
2  SEPT3           -4.6 4.5e-204    203.3468         down
3  YWHAH           -2.5 4.7e-191    190.3279         down
4   RPL3            1.7 5.4e-134    133.2676           ns
5  PI4KA           -2.0 2.9e-118    117.5376         down
6  SEZ6L           -5.1 4.2e-109    108.3768         down
```

```
%%R

# Rename the X.log10PAdj column header to neg.log10PAdj

colnames(hbr_uhr_chr22_deg)[4] <- "neg.log10PAdj"
```

```
%%R
# Load the ggplot2 package using R's library command
library(ggplot2)
```

Using ggplot2 to construct a volcano plot

To create the volcano plot initiate it with the `ggplot` command, which creates a blank plotting canvas. Enclosed in the `ggplot` command are the arguments and these are as follows.

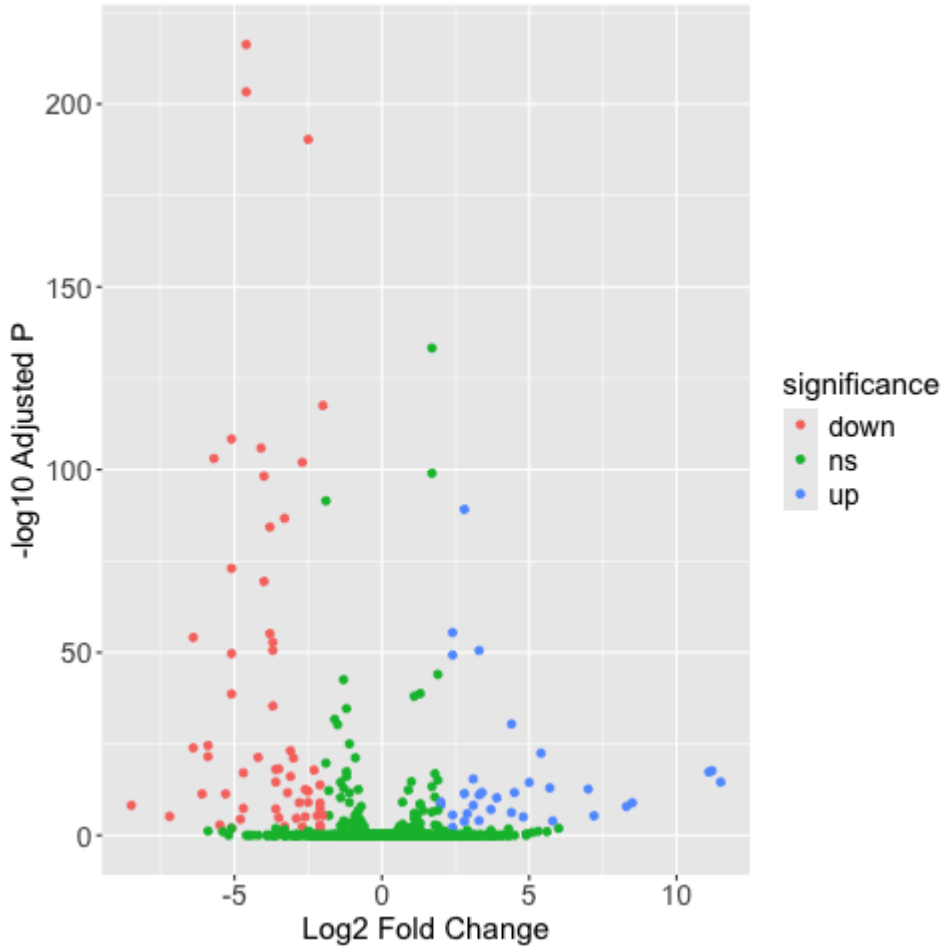
- Data table to generate a plot from: hbr_uhr_chr22_deg
- aes: specifies aesthetics of the plot include:
- Values to plot on the x axis (ie. log2FoldChange)
- Values to plot on the y axis (ie. neg.log10PAdj)
- What to color the points by (set by `color` and in this example values in the significance column in the data table showing whether gene expression change was statistically significant)

Next, specify the plot type. For a volcano plot (or scatter plot), `geom_point` will be used. The labels for the x and y axes are set by `xlab` and `ylab`, respectively. Finally, inside axis label, tick mark, and legend text fontsizes are specified in `theme`.

```
%%R

# Construct volcano plot.

ggplot(hbr_uhr_chr22_deg, aes(x=log2FoldChange, y=neg.log10PAdj, colo
geom_point()+xlab("Log2 Fold Change")+ylab("-log10 Adjusted P")+
theme(axis.title.x=element_text(size=15),axis.title.y=element_text(s
axis.text.x=element_text(size=14),axis.text.y=element_text(size=14),
legend.title=element_text(size=15),legend.text=element_text(size=14))
```



Running Unix commands

Users can run Unix commands within a Python Jupyter Notebook. To do this start a code block with "!" followed by the unix command. For instance, using the `pwd` command in the code block below to list the present working directory.

```
!pwd
```

```
/Users/wuz8/Library/CloudStorage/OneDrive-NationalInstitutesofHealth/tutorials/do
```

Writing Formatted Text

See https://www.markdownguide.org/basic-syntax/ *(https://www.markdownguide.org/basic-syntax/)* for a markdown guide.

Custom heading sizes

Use # to specify heading levels

```

```

```
# Heading level 1 (largest)
## Heading level 2 (second largest)
### Heading level 3 (third largest)
...
```

Lists

Un-ordered lists: use * or -

```
- DNA
- RNA
- protein
- metabolite
```

Ordered list: use numbers

```
1. Obtain sequencing data
2. Perform pre-alignment QC
3. Adapter and/or quality trim
3. Align sequencing data to reference genome
4. Obtain gene expression count matrix
5. Run differential expression analysis
6. Pathway analysis
```

Insert images

- Via HTML: `<img src="image_path" />`
- Via markdown: `![](image_path)`

Insert links

```
[Description of website](insert url)
```

Build a Table of Contents

To build a table of contents that links from one section of a Jupyter notebook to another do the following, where

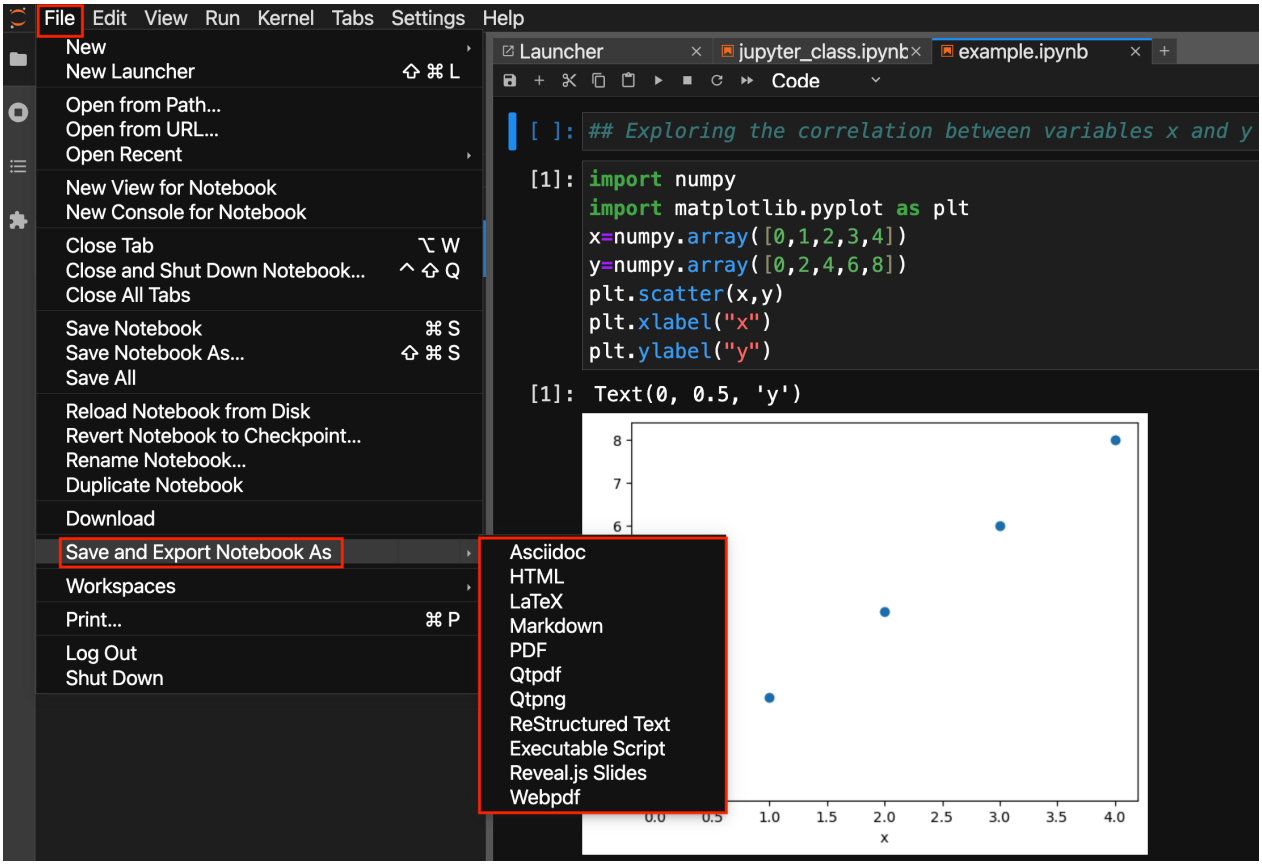
```

```

```
[Display name](#Section-header)
```

- Enclosed in `[]` is the display name for the link. The display name cannot have punctuations such as commas.
- Enclosed in `()` is the Markdown header for that section (ie. section header)
    - This is prepended by #
    - Each part the header is connected by `-`.

Exporting Jupyter Notebook using GUI



Exporting Jupyter Notebook using Command Line

Use the `jupyter nbconvert` command at the command prompt to convert Jupyter Notebook to various available format, including html, pdf, and slides. The format is specified after the `--to` option.

```
jupyter nbconvert --to format
```

Sharing Jupyter Notebook

- Github *(https://github.com)*
  - Static notebook (ie. users will not be able to run)
- Binder *(https://mybinder.org)*
  - Provide data
  - Provide list of packages
  - Users can run the notebook
  - Example *(https://mybinder.org/v2/gh/ncbi/workshop-ncbi-data-with-python/main?filepath=notebooks%2Fworkshop.py)*

Class Data

Option 1 for obtaining the class data and example Jupyter Notebook is to download the following to personal computer. Again, installation of Jupyter Lab and other computation tools is required.

- Example Jupyter notebook
- hbr_uhr_deg_chr22_with_significance.csv
- hbr_uhr_normalized_counts_pca.csv
- hbr_uhr_top_deg_normalized_counts.csv
- image1
- image2
- image3
- image4
- image5

> **Tip**
>
> To download the example Jupyter notebook on Google Chrome right click on the link and choose "Save Link As". On Safari, right click on the link and select "Download Linked File".

Option 2 is to visit https://github.com/JWrows2014/document_analysis_with_jupyter *(https://github.com/JWrows2014/document_analysis_with_jupyter)*, click on the green button labeled "<> Code" and select "Download as ZIP". The example Jupyter Notebook, data, and images will download as a zip file and unzipping, change into the project folder and start Jupyter Lab to try out the example notebook.