UT Southwestern Medical Center
Lyda Hill Department of Bioinformatics

BioHPC

Jupyter Notebook

[web] portal.biohpc.swmed.edu
[email] biohpc-help@utsouthwestern.edu

2022-03-09

Outline

- Overview
- Getting started on BioHPC
  - Demo
- Basic functionality
  - Markdown cells
  - Code cells
  - Extras
- Create your own Jupyter Notebook environment
  - 10 recipes of kernels
- Suggestion
- Reference

Jupyter Notebook Example

jupyter Untitled Last Checkpoint: Last Thursday at 4:21 PM (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Run Code

The Experiment

Ultrices eros in cursus turpis massa tincidunt dui ut. Sed libero enim sed faucibus. Arcu dictum varius duis at consectetur. Ultricies mi eget mauris pharetra. Id faucibus nisl tincidunt eget nullam. In eu mi bibendum neque egestas. Ultrices dui sapien eget mi proin sed libero enim. Pellentesque elit ullamcorper dignissim cras. Mattis molestie a iaculis at erat pellentesque adipiscing commodo. Nulla facilisi cras fermentum odio. Malesuada bibendum arcu vitae elementum curabitur vitae nunc. Eget egestas purus viverra accumsan in nisl. Faucibus purus in massa tempor nec feugiat. Turpis nunc eget lorem dolor sed. Malesuada fames ac turpis egestas maecenas pharetra convallis. Sed augue lacus viverra vitae congue eu consequat. Laoreet non curabitur gravida arcu.

In [15]:
```
import pandas as pd

experiment = pd.read_csv('experiment_data.csv')
experiment.head()
```

Out[15]:

	x	y
0	-6.283185	0.352810
1	-6.156252	0.206624
2	-6.029319	0.446896
3	-5.902386	0.819841
4	-5.775453	0.859708

Fermentum dui faucibus in ornare quam viverra orci sagittis. Interdum velit euismod in pellentesque massa placerat. Dignissim sodales ut eu sem integer vitae justo eget magna. Purus gravida quis blandit turpis cursus. Ornare massa eget egestas purus viverra. Congue eu consequat ac felis. Interdum varius sit amet mattis vulputate enim nulla aliquet. Elementum facilisis leo vel fringilla est ullamcorper eget nulla facilisi. Sodales ut etiam sit amet nisl purus. Adipiscing elit ut aliquam purus. Nulla porttitor massa id neque aliquam vestibulum morbi. Consectetur adipiscing elit pellentesque habitant morbi tristique senectus et netus.

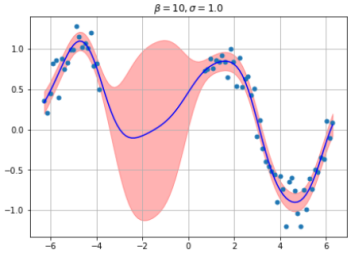
$$E[y] = k(x, x)^T \mathbf{C}_N^{-1} y$$

In [18]:
```
import gkernel as gk

beta = 10
sigma = 1.0

x_train = experiment['x'].to_numpy()
y_train = experiment['y'].to_numpy()

x_r = np.linspace(x_train[0], x_train[-1], 100)
model = gk.MyModel(beta=beta, sigma=sigma)
model.train(x_train, y_train)
y_pred, y_std = model.predict(x_r)
```



Load and reference data

Formatted text along with equations

Executable code snippets

Comparison

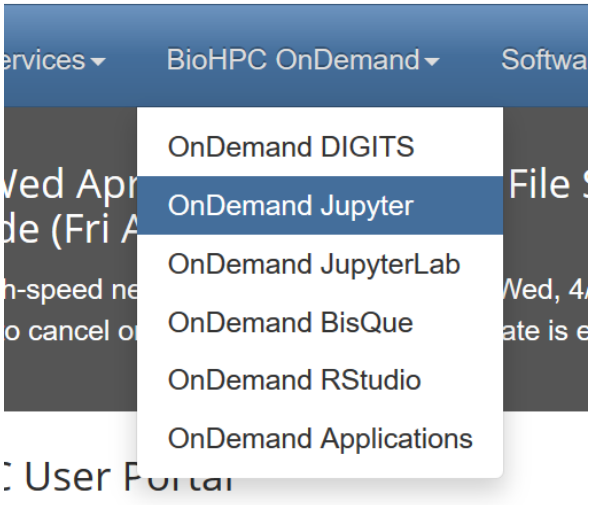
MS Word Document	Jupyter Notebook
• Single binary file (.docx, .doc)	• Single main file, text formatted (.ipynb) and additional optional assets
• MS Word application to open and edit	• Browser displays text file via a web server (needs python runtime env)
• WYSIWYG editor	• Simple markup (md) text
• Editable (text), uninterpreted, static content (generally)	• Editable, interactive, executable content for in-place (re-)creation.

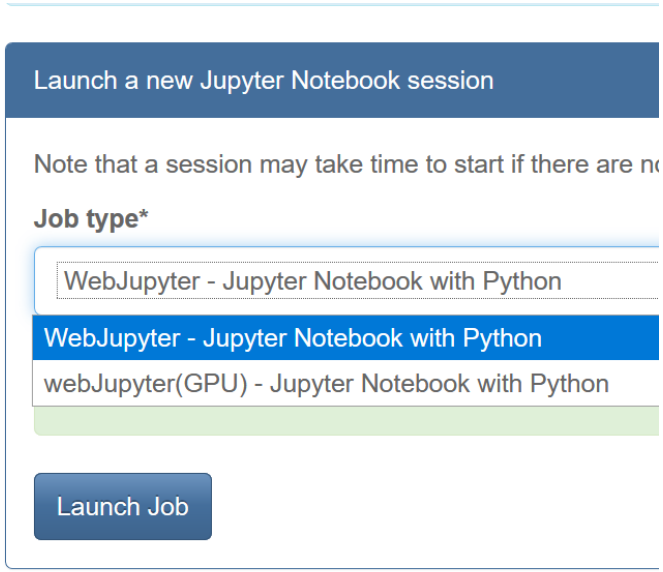
Why Jupyter Notebook

- Interactive session that can include
  - Formatted text
  - Executable code snippets
  - Equations (LaTeX syntax)
  - Updatable graphs and images
- Can improve reproducibility and portability of documents and results
- Example use cases
  - Documented workflows and pipelines: paper, open-source code
  - Course material

Getting Started – Jupyter Notebook on BioHPC

- Portal https://portal.biohpc.swmed.edu/terminal/ondemand_jupyter/



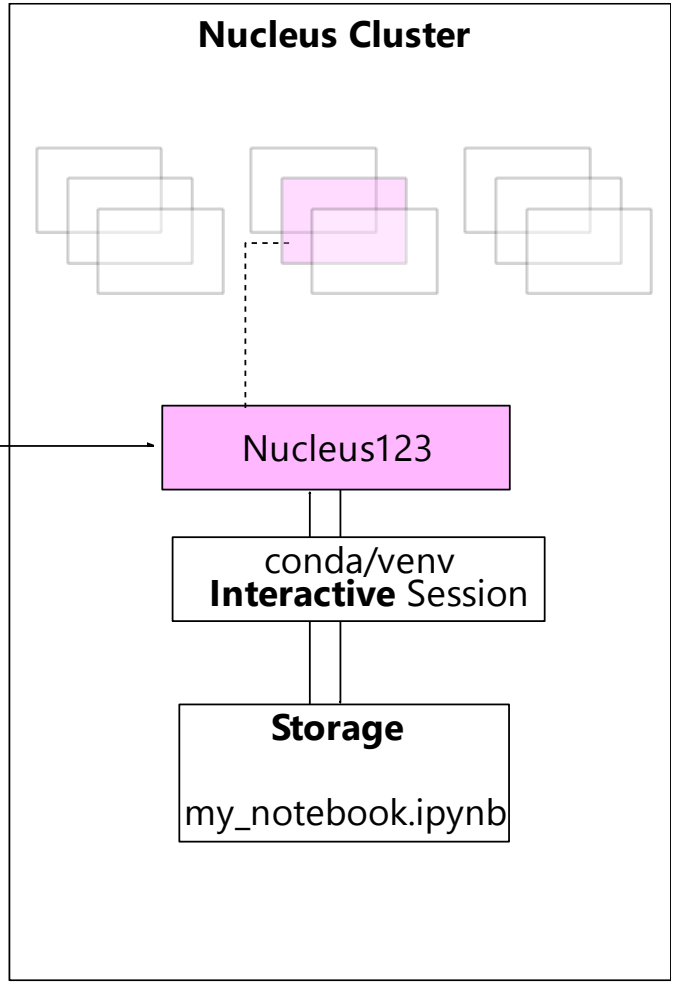


- Demo

OnDemand Jupyter Notebook on BioHPC

Run Jupyter Notebook in two ways:

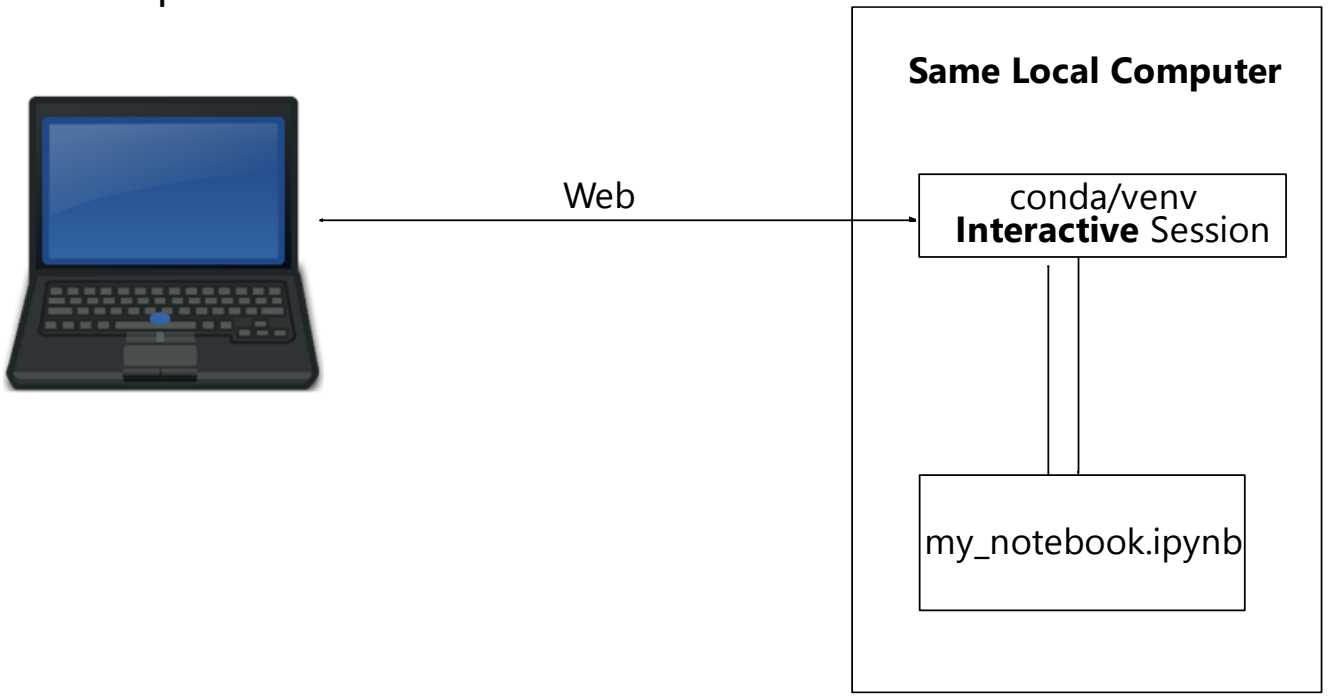
- ✓ BioHPC
- Local computer



Jupyter Notebook on local machine

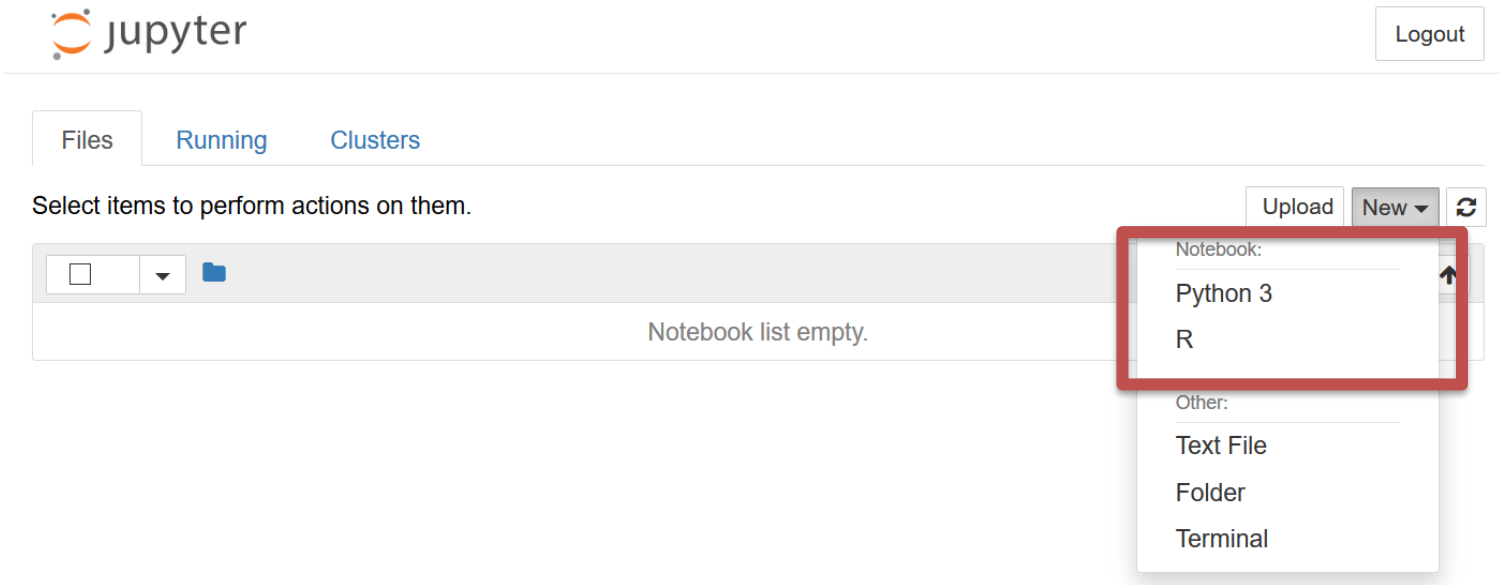
Run Jupyter Notebook in two ways:

- BioHPC
- ✓ Local computer



Getting Started - BioHPC

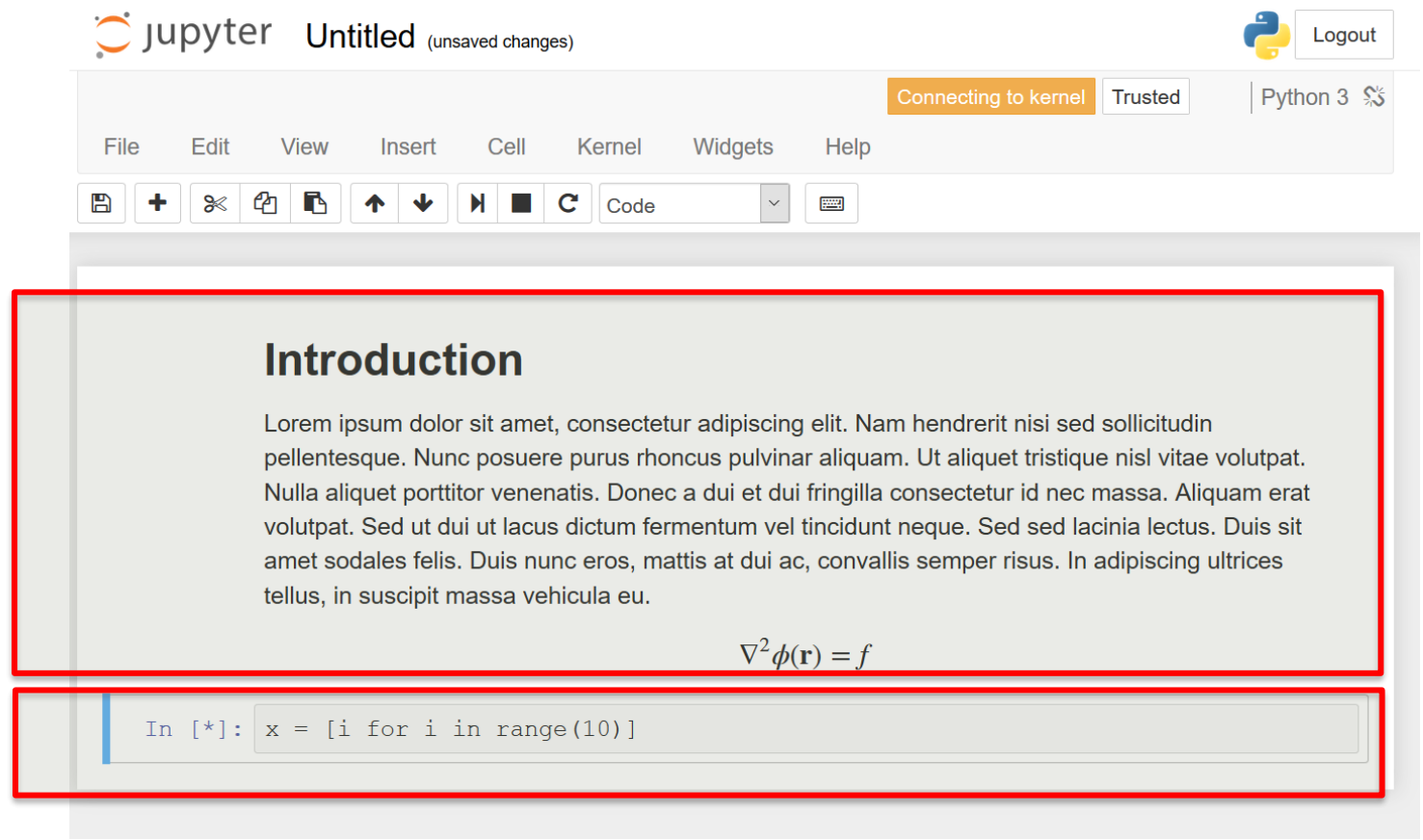
- Login to Notebook



Kernels (will come back to this)

- Location: `/home2/username/jupyter_notebooks/`

Usage



Cell 1

(markdown)

Cell 2

(code)

Usage

Markdown Cell → Evaluate → Formatted Text, Equation Image, etc.

Code Cell → Evaluate → Evaluate expression, display output, graphs, etc.

Usage - Markdown

Markup → Evaluate → Rendered

```
# Markdown Basics

Jupyter Notebook markdown supports extended markdown.
You can insert **bold text**, have *text in italics*,
or insert inline monospace code `let x = 42;`.

Insert lists by prefixing them with a dash:

- item 1
- item 2
- item 3
```

Markdown Basics

Jupyter Notebook markdown supports extended markdown. You can insert **bold text**, have *text in italics*, or insert inline monospace code `let x = 42;`.

Insert lists by prefixing them with a dash:

- item 1
- item 2
- item 3

Usage - Markdown

Markup → Evaluate → Rendered

```
Insert tables as text:

| Fruit    | Color  |
|----------|--------|
| Banana   | Yellow |
| Apple    | Red    |
| Cucumber | Green  |

Insert code blocks
```python
square = lambda x : x**2
```

See Github-flavor basic markdown [cheat sheet]
(https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet) here.
```

Insert tables as text:

Fruit	Color
Banana	Yellow
Apple	Red
Cucumber	Green

Insert code blocks

```python
square = lambda x : x**2
```

See Github-flavor basic markdown cheat sheet here.

Usage - Markdown

Markup → Evaluate → Rendered

```
## Equations Subsection

But wait, there's more! Jupyter notebook supports an
extended markdown. You can insert basic insert equation
blocks directly in your document:

$$
\nabla^2 \phi(\mathbf{r}) = f
$$

You can also inline your equation. For example $ x \in
\mathbb{C}^{N} $ can be inlined.
```

Equations Subsection

But wait, there's more! Jupyter notebook supports an extended markdown. You can insert basic insert equation blocks directly in your document:

$$\nabla^2 \phi(\mathbf{r}) = f$$

You can also inline your equation. For example $x \in \mathbb{C}^N$ can be inlined.

Code Cell

In [10]:
```
from functools import reduce
import operator

def factorial(n):
    return reduce(operator.mul, range(1, n+1))

factorial(4)
```

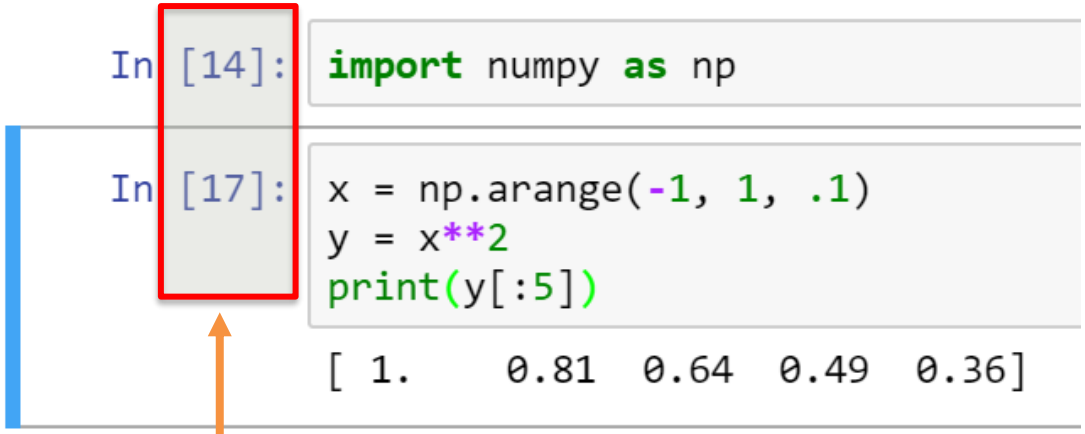
Out[10]: 24

Code

Cell 1

Cell 2

'np' is also available in this cell



Execution Order

(Recall: **interactive** session)

Interactive session implications

Out of place execution order

Dataframe default table rendering

In [75]:
```
import pandas as pd

rstate = np.random.RandomState(123)
df = pd.DataFrame(rstate.randn(100, 6))
df.head()
```

Out[75]:

	0	1	2	3	4	5
0	-1.085631	0.997345	0.282978	-1.506295	-0.578600	1.651437
1	-2.426679	-0.428913	1.265936	-0.866740	-0.678886	-0.094709
2	1.491390	-0.638902	-0.443982	-0.434351	2.205930	2.186786
3	1.004054	0.386186	0.737369	1.490732	-0.935834	1.175829
4	-1.253881	-0.637752	0.907105	-1.428681	-0.140069	-0.861755

Extras

Cell Command	Evaluation Result
object**?**	List documentation for 'object'
%lsmagic	List available extra functionality built into the kernel
%quickref	Quick reference on the kernel

Extras

```
In [8]: np.linspace?
```

```
Signature: np.linspace(start, stop, num=50, endpoint=True, retstep=False, dtype=None)
Docstring:
Return evenly spaced numbers over a specified interval.

Returns `num` evenly spaced samples, calculated over the
interval [`start`, `stop`].

The endpoint of the interval can optionally be excluded.

Parameters
----------
start : scalar
    The starting value of the sequence.
stop : scalar
```

Code - Magics

Built-in to Kernel. Not platform dependent

```
In [38]: %lsmagic

Out[38]: Available line magics:
%alias  %alias_magic  %autocall  %automagic  %autosave  %bookmark
%cat  %cd  %clear  %colors  %config  %connect_info  %cp  %debug
%dhist  %dirs  %doctest_mode  %ed  %edit  %env  %gui  %hist  %his
tory  %killbgscripts  %ldir  %less  %lf  %lk  %ll  %load  %load_e
xt  %loadpy  %logoff  %logon  %logstart  %logstate  %logstop  %ls
%lsmagic  %lx  %macro  %magic  %man  %matplotlib  %mkdir  %more
%mv  %notebook  %page  %pastebin  %pdb  %pdef  %pdoc  %pfile  %pi
nfo  %pinfo2  %popd  %pprint  %precision  %profile  %prun  %psear
ch  %psource  %pushd  %pwd  %pycat  %pylab  %qtconsole  %quickref
%recall  %rehashx  %reload_ext  %rep  %rerun  %reset  %reset_sele
ctive  %rm  %rmdir  %run  %save  %sc  %set_env  %store  %sx  %sys
tem  %tb  %time  %timeit  %unalias  %unload_ext  %who  %who_ls  %
whos  %xdel  %xmode

Available cell magics:
%%!  %%HTML  %%SVG  %%bash  %%capture  %%debug  %%file  %%html  %
%javascript  %%js  %%latex  %%perl  %%prun  %%pypy  %%python  %%p
ython2  %%python3  %%ruby  %%script  %%sh  %%svg  %%sx  %%system
%%time  %%timeit  %%writefile

Automagic is ON, % prefix IS NOT needed for line magics.
```

Code - Magics

```
In [9]: %matplotlib?
```

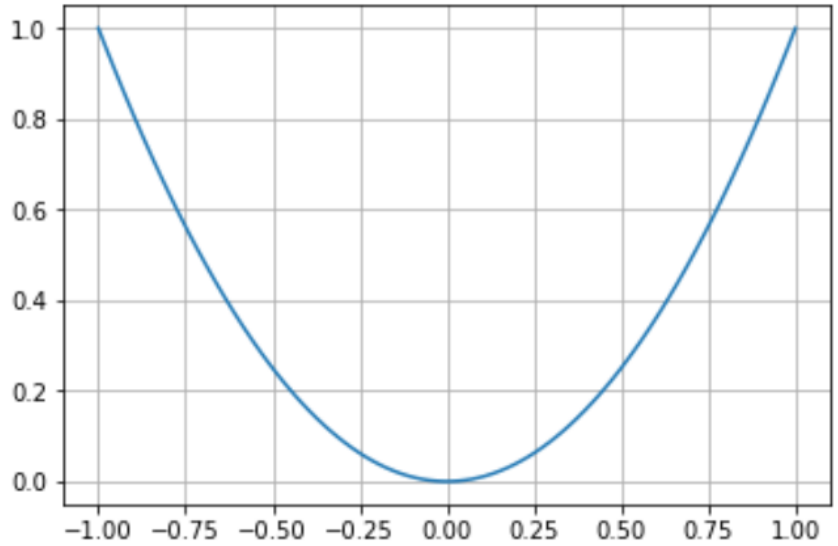
```
Docstring:
::

  %matplotlib [-l] [gui]

Set up matplotlib to work interactively.

This function lets you activate matplotlib interactive support
at any point during an IPython session. It does not import anything
into the interactive namespace.

If you are using the inline matplotlib backend in the IPython Notebook
you can set which figure formats are enabled using the following::

    In [1]: from IPython.display import set_matplotlib_formats
```

Code - Magics

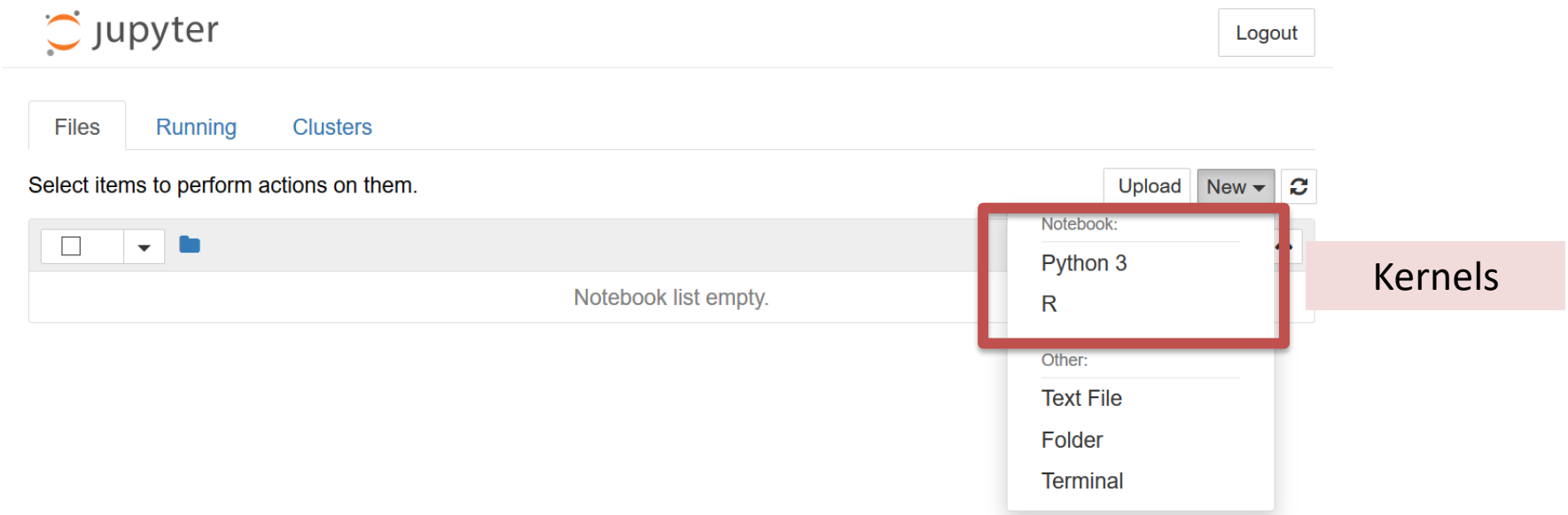
```
In [41]: %matplotlib inline
```

```
In [63]: x = np.linspace(-1.0, 1.0, 100)
         y = x**2
         fig, ax = plt.subplots()
         ax.grid(True)
         ax.plot(x, y)
```

```
Out[63]: [<matplotlib.lines.Line2D at 0x7fffbc30ffd0>]
```



Getting Started - BioHPC



Run Jupyter Notebook in two ways:

- ✓ BioHPC
- Local computer
- Note: can modify environment directly from login node

Recipe 1 – Create a new isolated python3 kernel using Conda on BioHPC

```
$ ssh yourusername@nucleus.biohpc.swmed.edu

$ module load python/3.7.x-anaconda

$ conda create --name foo python=3          # Look under /home2/<yourusername>/.conda/envs

$ conda activate foo

(foo) $ pip install ipykernel

(foo) $ pip install <additional modules as needed>

(foo) $ python -m ipykernel install --user --name foo --display-name "my foo env“

(foo) $ conda deactivate
```

1) create conda env
2) install packages
3) create jupyter kernel
4) exit conda env

Recipe 2 – Create a new isolated R kernel using Conda on BioHPC

```
$ ssh yourusername@nucleus.biohpc.swmed.edu
$ module load python/3.7.x-anaconda
$ conda create --name my_renv r r-essentials r-base        # Look under /home2/<yourusername>/.conda/envs
$ conda activate my_renv
(my_renv) $ R
> IRkernel::installspec(name="my_renv", displayname="my R demo")        # Displayname shows in jupyter notebook
> q()
(my_renv) $ conda deactivate
```

1) create conda env
2) install packages
3) create jupyter kernel
4) exit conda env

Recipe 3 – Create a new MATLAB kernel using Conda on BioHPC

```
$ ssh yourusername@nucleus.biohpc.swmed.edu

$ module load python/3.7.x-anaconda

$ conda create --name my_matlab_env python=3.6        # currently MATLAB only supports 2.7, 3.5, 3.6

$ conda activate my_matlab_env

(my_matlab_env) $ cd /home1/apps/MATLAB/R2020a/extern/engines/python

(my_matlab_env) $  python3 setup.py build --build-base=/home2/yourusername/tmp install \
    --prefix=/home2/yourusername/.conda/envs/my_matlab_env/

(my_matlab_env) $ pip install matlab_kernel

(my_matlab_env) $ python -m matlab_kernel install --user --name "my_matlab_demo"

(my_matlab_env) $ conda deactivate
```

1) create conda env
2) install packages
3) create jupyter kernel
4) exit conda env

Recipe 4 – Create a new conda environment on different partition

(One conda env can easily use several GB. /home2 Have only 50 GB storage under ~/)

```
$ ssh yourusername@nucleus.biohpc.swmed.edu
$ module load python/3.7.x-anaconda
$ mkdir /project/yourdepartment/yourusername/conda_envs          # 'conda_env' is arbitrary name
$ conda create --prefix /project/yourdepartment/youruser/conda_envs/your_env_name
$ conda activate /project/yourdepartment/youruser/conda_envs/your_env_name          # must reference by path
(long/path/your_env_name) $ <install additional modules as needed>
(long/path/your_env_name) $ conda deactivate
```

Recipe 5 – View and Modify Jupyter kernels

```
$ module load python/3.7.x-anaconda

$ jupyter kernelspec list

my_env
foo

my_matlab_demo

$ jupyter kernelspec uninstall my_env                    # Removes kernel, not environment
```

Recipe 6 – Remove Conda environment

```
$ module load python/3.7.x-anaconda

$ conda env list

my_env                                    # Look under ~/.conda/envs
foo

my_matlab_demo

$ conda env remove --name my_env
```

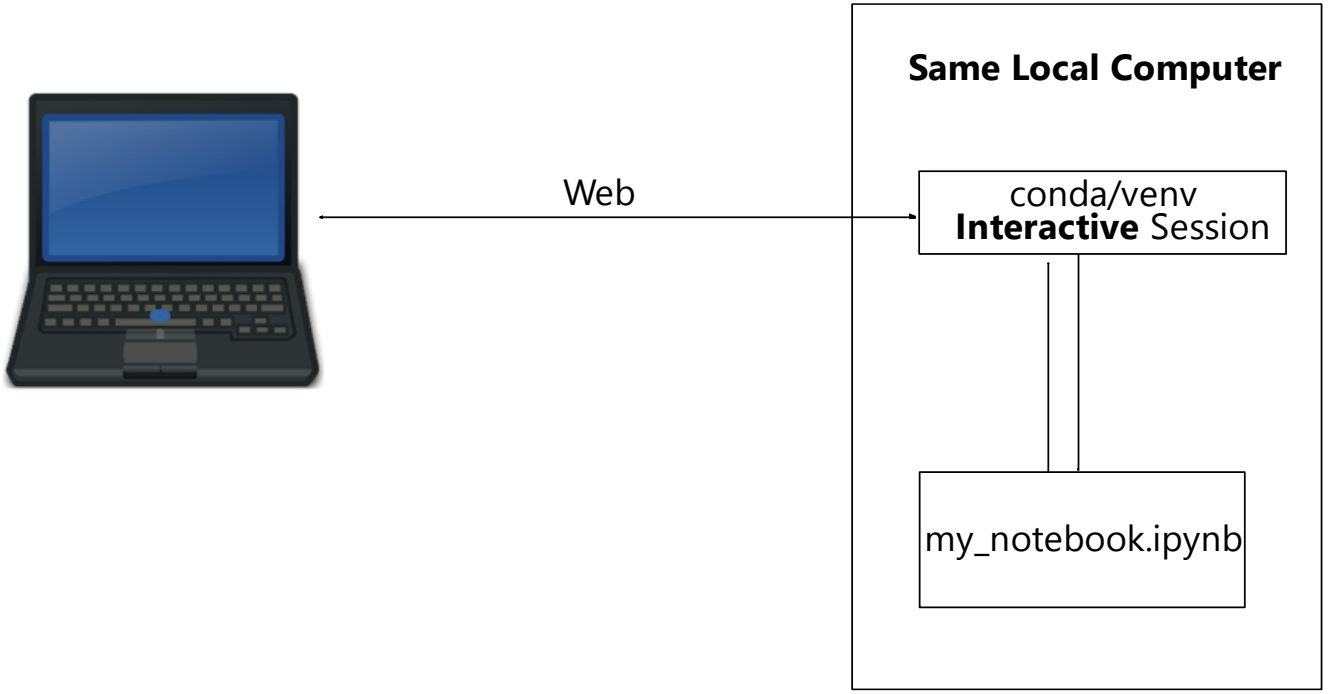
Recipe 6.1 – Clean Conda unused packages and caches

```
$ module load python/3.7.x-anaconda
$ conda clean --all --dry-run                                   # Dry run will NOT delete anything
        Will remove the following tarballs:
        /home2/<your username>/.conda/pkgs
            pandocfilters-1.4.2-py_1.tar.bz2            9 KB
            r-broom-0.7.6-r40hc72bb7e_0.tar.bz2         1.7 MB
            fonts-conda-forge-1-0.tar.bz2               4 KB
# if satisfied with the operation to be performed
$ conda clean –all                                              # This command will DELETE those packages
```

Jupyter Notebook on local machine

Run Jupyter Notebook in two ways:

- BioHPC
- ✓ Local computer



Recipe 7 – Install Jupyter Notebook locally using venv

(virtual environment of python)

```
$ mkdir myproject && cd myproject

$ python -m venv foo

$ ./foo/bin/activate

(foo) $ pip install jupyter

(foo) $ <install additional modules>

(foo) $ jupyter notebook                    # Start Jupyter Notebook webserver
```

Example Workflow

Recipe 8: Distribute Conda-based project

(Assumes that you have already followed **Recipe 1** or **Recipe 3**)

Let's say your research is contained in the betsy_research environment

a) You can export conda env to environment.yml

```
$ ssh yourusername@nucleus.biohpc.swmed.edu

$ module load python/3.7.x-anaconda

$ conda activate betsy_research

(betsy_research) $ conda env export --file environment.yml
```

b) If applicable, you can also create venv export of environment

```
(betsy_research) $ pip freeze > requirements.txt
```

Share your project files along with environment.yml and/or requirements.txt

Example Workflow

Recipe 9: Work on BioHPC, on a project shared with you.
Restore environment based on yml

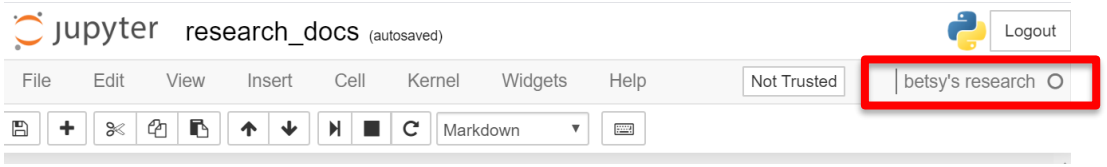
Let's say **`betsy_research`** was shared with you:

- research_docs.ipynb
- environment.yml
- other assets

$ ssh yourusername@nucleus.biohpc.swmed.edu

$ module load python/3.7.x-anaconda

$ cd path_to_environment.yml



change env name (line1 of environment.yml) to *created_env_name* if needed

$ conda env create -f environment.yml

$ conda activate *created_env_name*

(*created_env_name*) $ python -m ipykernel install --user --name *created_env_name* --display-name "*some display name*"

Example Workflow

Recipe 10: Load venv-compatible project locally using virtual-environment
Restore environment based on requirements.txt

Let's say `betsy_research` was shared with you that has requirements.txt:
- research_docs.ipynb
- requirements.txt
- other assets

```
$ cd path_to_betsy_research_directory

$ python3 -m venv betsy_env

$ ./betsy_venv/bin/activate

(betsy_venv) $ pip install -r requirements.txt

(betsy_venv) $ jupyter notebook
```

Suggestions

- Where applicable:
  - **Consider** not linking to contents on the internet
  - **Consider** using a fixed seed for 'random' number generation for reproducibility
  - **Seriously consider** using different environments for different projects
  - **Always** store and distribute your Notebooks (and projects) with dependency specifications

Check training materials on portal – Training – Training slides and handouts

Usage: (Open on BioHPC)

$ ssh yourusername@nucleus.biohpc.swmed.edu

$ cd path/to/demo/file

$ module load python/3.7.x-anaconda

$ conda create --name biohpc_demo

$ conda activate biohpc_demo

(biohpc_demo) $ pip install --requirement requirements.txt

(biohoc_demo) $ *python -m ipykernel install --user --name biohpc_demo --display-name="BioHPC_Demo"*

(biohpc_demo) $ conda deactivate

Check training materials on portal – Training – Training slides and handouts

Usage (Your own machine):

```
$ cd path/to/demo/files
$ python3 -m venv biohpc_demo                  # This will create biohpc_demo folder in the current directory
$ chmod +x biohpc_demo_env/bin/activate        # add execution permission to file activate
$ source biohpc_demo_env/bin/activate
(biohpc_demo) $ pip install -r requirements.txt
(biohpc_demo) $ jupyter notebook
```

References

- Jupyter Notebook Documentation
  - https://jupyter.org/documentation

- Conda getting started

https://docs.conda.io/projects/conda/en/latest/user-guide/getting-started.html

- Markdown
  - Github Markdown cheat sheet

    https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet
  - Interactive Markdown

    https://www.markdowntutorial.com/
  - Markdown Pad

    http://www.markdownpad.com/