Resources: python & jupyter

George McNinch

2024-01-01

Overview

Our course will use the `python` programming language to perform some machine computations needed to carry out mathematical modeling tasks.

And parts of the notes of the course lectures will be made available as `jupyter notebooks`.

There are a few ways that you should learn to interact with `python` and `jupyter` for our class:

1. via Google's colab
2. by executing `python` code in a `python` interpreter on your computer
3. (optionally) using jupyter notebooks on your computer.

Here is a bit of what you need to know to enable these interactions:

- Overview of `jupyter` notebooks
- Interacting with `jupyter` notebooks via colab
- installing the tools on your computer
- using the `jupyter notebook` viewer on your computer

Before diving into the details, let me try to answer a few question that you may have:

Q - do I need `juptyer` and `colab` on my computer, or I just use `colab`? A - It is probably possible to just use `colab`. `colab` provides a reasonable environment for reading the course material, though for code execution it may (?) be a tad slow.

But: using `python` exclusively through `colab` is a somewhat imperfect environment for producing code, and at least in the long run you'll be better served by learning to work with these tools on your own computer.

An overview of `jupyter` notebooks

`Jupyter` notebooks provide a way to package written material – with possible mathematical markup – together with an interactive environment for executing computer code.

The material for this course will mostly be presented in the form of these `notebooks`. You can recognize `notebook` files by their filename extension – their file name has the form `*.ipynb`.

On the course site, notebooks will be posted with *two links*.

- The simplest way to interact with `jupyter notebooks` is to use Google's colab. Clicking one of the links will open the notebook in `colab`.
- The other one link permits you to download the `notebook` file to your computer. In order to use this file, you will need to have a working installation of `python` and `jupyter`. See below for some discussion of how to set up the required tools and how to use `jupyter`.

Interacting with `jupyter` notebooks via Google's CoLab

As mentioned above, the course material will be presented in the form of `jupyter` notebooks, and for each notebook there will be a link to a copy of this notebook available via `colab`.

`Colab` is a web-based free-to-use Jupyter notebook environment that runs in *the cloud* and stores its notebooks on Google Drive. When you view a notebook in `colab` containing code, that code can be executed. The code execution occurs *in the cloud*, which basically means *on some google computer*. Thus, google's computer has `python` and the relevant libraries installed and available.

You can choose to interact with the course notebooks on `colab`, abd create your own notebooks there.

A good starting point to learn how to use colab (and jupyter notebooks in general) is the document Overview of Colaboratory Features.

Installing the tools on your computer

Here I hope to answer the questions: "What do I need to do to be able to create and execute `python` code for the course on my computer?" and "What do I need to do to be able to view and execute `jupyter` notebooks on my computer?"

I suggest that you use a tool known as `conda` (short for *anaconda*; together with `python`, this amounts to a bunch of snakes …).

`conda` describes itself as follows:

> Initially started as a multi-platform package management tool, the term "conda" has since evolved to encompass an entire open-source packaging ecosystem and philosophy. This ecosystem is supported by many organizations who all share the common goal of providing easier access to programming tools and libraries.

Specifically, I recommend installing `conda` using something called the `miniforge` – here is the `miniforge` github repository.

Here is the list of *installers*. The installer must be chose to match your computer's architecture and operating system.

- If you have a recent Apple MacBook ("M1" or "M2") your architecture is likely `arm64` and you need to use the `OS X arm64 (Apple Silicon)` installer.
- somewhat older Apple MacBooks need to use the `OS X x86_64` installer.
- Windows users need to use the `Windows x86_64` installer.
- There are installers for `linux` users as well.

You should now read and follow the installation instructions for your operating system. (I've provided some annotation below, which you might read first).

installing in `Mac OS` or `Linux`

The `Mac OS & Linux` instructions tell you to execute the following commands in a terminal window (i.e. a "shell"):

```
~$ curl -L -O "https://github.com/conda-forge/miniforge/releases/latest/download/Miniforge3-$(uname)-$

~$ bash Miniforge3-$(uname)-$(uname -m).sh
```

Note that these should be executed in a `shell`– in particular, you don't type the prompts ("`~$`"). The first command uses the utility `curl` to download the relevant installer (the expressions `$(uname)` and `$(uname- m)` should select the correct installer for your computers architecture + operating system, without you having to make the choice).

The second command (`bash...`) *executes* the installer that you just downloaded. When it is running, you'll need to interact a bit with it (pressing "ENTER" when requested, accepting the licence terms, confirming the install location, etc.)

Once the installer is done downloading packages from the `conda-forge`, you will be asked:

```
Do you wish the installer to initialize Miniforge3
by running conda init? [yes|no]
```

In order to use `conda`, you should answer `yes`. It will add some code to your config file `~/.bashrc`.

After closing and reopening your shell, you should have a prompt that looks something like the following:

```
(base) george@valhalla:~$
```

The `(base)` prefix indicates that the `base` environment of `conda` has been `activated`.

Typing something like "`which python`", you should get a response like the following:

```
(base) george@valhalla:~$ which python
/home/george/miniforge3/bin/python
```

This indicates that if you execute `python` from the command shell, it will run the just-installed version from the `miniforge3/bin` directory.

You can start a python `repl`:

```
(base) mcninch-store@valhalla:~$ python
Python 3.10.12 | packaged by conda-forge | (main, Jun 23 2023, 22:40:32) [GCC 12.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

You can exit the `python repl` by typing `C-d` ("control d").

You can deactivate `conda` with the command

```
(base) george@valhalla:~$ conda deactivate
```

And you can reactivate it with the command

```
george@valhalla:~$ conda activate base
```

installing in `Windows`

As the installation instructions say, you should download and execute the windows installer.

Note that there will be a `Miniforge Prompt` installed to the start menu; opening that prompt should enable you to run software installed in the `miniforge`.

For example, from that prompt you should be able to start a `python repl` by executing `python`.

Remaining installation steps.

In this class, we are going to use some `python` libraries, as well as `jupyterlab`.

Not everything we need was installed from `miniforge`, but it is now easy to install what we need.

Execute the following commands:

```
(base) george@valhalla~$ mamba install jupyterlab numpy sympy pandas pydot
```

(You'll have to respond to `Confirm changes: [Y/n]` by typing `Y`. And then the `mamba` will install probably a large number of new packages…)

An editor

You need to use a text editor to create and edit `python` code. What editor you choose to use is a matter of personal preference, but I'm going to recommend that you use the Atom editor. This doesn't mean you are required to use `Atom`, but it is the editor that I'll use in demonstrations. The above-linked web-site has installation instructions for Windows, Mac, and Linux computers.

Interacting with `Jupyter` on your computer

You can always view and interact with `jupyter notebooks` on `colab`. But if you carried out the above instructions for installing `conda`, you can now use `jupyter` notebooks on your own computer.

Start `jupyter` from the command-line. I actually find that I prefer the `jupyter lab` interface to the the plain `jupyter notebook` interface (and honestly I'm not sure I understand why there are two different interfaces) but you should experiment for yourself.

So, type:

```
george@valhalla:~$ jupyter-lab
```

There will be some output to the terminal that you can mostly ignore. The significant effect of running this program is that *a page should open in your web browser*.

The page that opens will have a `URL` that looks something like:

```
http://localhost:8888/lab
```

This `URL` tells you that there is now a `server` running on your local machine, and that your web browser is communicating with that server.

In the web page at this location, you should see in particular an interface where you can interact with `jupyter notebooks`. You can create a new notebook from the `File` menu: `New -> Notebook`. If you are asked to `Select Kernel`, you should select `Python 3`.

You are now editing a file name `Untitled.ipynb`.

If you save a `jupyter notebook` (i.e. a `ipynb` file) in your directory tree, you can open it using the `File` menu: `Open from path`

You can end the `jupyter` notebook session by typing `C-c` (twice) in the terminal where you started the command, or by menu-driven commands in the web-interface.

Some learning resources for python and scientific computing

(Honestly, I just borrowed this list from a previous instructor, but the links look useful).

- DataCamp's interactive Intro to Python tutorial
- Python for Everybody (University of Michigan MOOC)
- A short Python and NumPy tutorial (Stanford CS231)
- Official Python tutorial
- Numpy
  - Introduction to NumPy
  - NumPy reference guide (chapter 2 of VanderPlas' *Python Data Science Handbook*)
- Matplotlib
  - Official Pyplot tutorial (short)
  - Visualization with Matplotlib (chapter 4 of VanderPlas' *Python Data Science Handbook*)
  - Gallery of Matplotlib examples
