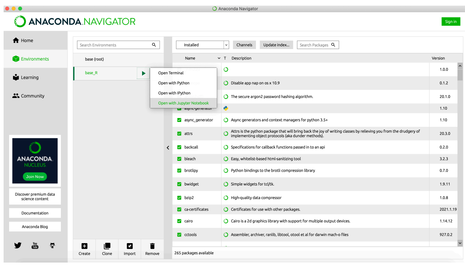
Technical Issues

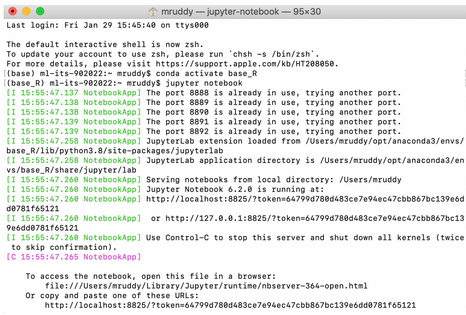
Connecting to R Kernel (Mac)

There are multiple ways to launch Jupyter Notebook after installed Anaconda

1. Environment Tab in Navigator

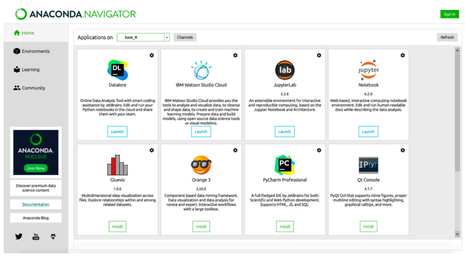


1. Command Line



1. Launch command under the Home Tab in Navigator

!!!Known Issues!!!



With the current version of Anaconda Navigator, you may have trouble connecting the R kernel when using Method 3. If you are using Method 1 or 2 to launch Jupyter notebook and are unable to connect to the R kernel, try removing and then re-creating the environment to use R. If this fails try uninstalling and reinstalling anaconda navigator (See here for details on how to do this). Otherwise please contact me.

Exporting .pdf files using pdf via LaTeX (Mac)

- Currently when running Jupyter notebook using method 1 above, attempting to export your notebook as a pdf via LaTeX returns an error (even if you have installed MiKTeX). However, it seems to work if you launch Jupyter notebook using method 2. I am still investigating why this is the case.
- In any case, I suggest a slightly longer method to create a nice .pdf of your Jupyter notebook:
  1. First download the notebook as an .html file
  2. Open the .html file with your favorite browser
  3. Export as this as an .html

While converting your notebook to a .pdf using LaTeX can give good results, I find that the .html sometimes looks much nicer and cleaner.

Interesting questions about R

1. Why does `seq(1,10)` output a sequence starting at 1? (by Ned Azar)

In [1]:
```
seq(1,10)
```

1. 1
2. 2
3. 3
4. 4
5. 5
6. 6
7. 7
8. 8
9. 9
10. 10

Unlike some other programming languages (Python for instance), R starts indexing lists and sequences at `1` rather than `0`. Which is "better" is subjective and depends on what it is you are trying to do.

- Fun fact: There are similar debates in mathematics for whether the "natural numbers" should start at `0` or `1`.

1. What is the difference between `seq(1,10)` and `seq.int(1,10)`? (by Mia Kobayashi)

In [2]:
```
seq(1,10)
seq.int(1,10)
```

1. 1
2. 2
3. 3
4. 4
5. 5
6. 6
7. 7
8. 8
9. 9
10. 10

1. 1
2. 2
3. 3
4. 4
5. 5
6. 6
7. 7
8. 8
9. 9
10. 10

This is a non-trivial question with an interesting answer and history. It is best described by the most upvoted answer here.

TLDR: They do the same thing, but in slightly different ways. Using either is fine, but to be safe, stick with `seq` for now.

In [ ]:
```

```