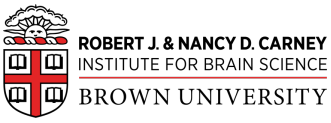
Computational Fluency Short Course

Creating, Checking, and Correcting Code

Jason Ritt
jason_ritt@brown.edu
Scientific Director of Quantitative Neuroscience



https://github.com/brownritt/cfsc25

What is the purpose of programming for research?

For most scientific applications, code is not "the product", it is a means to an end.

In data analysis, code is an exact, concrete manifestation of the quantitative logic of the study. The code matters only to the extent that doing the study is informative.

Scientific code should be (at minimum):

- *Useful*: it accomplishes some scientific goal
- *Honest*: it actually does the thing it claims to do
- *Reproducible*: others can run it and get exactly the same results
- *Auditable*: others can inspect and understand what it does

It is your responsibility to produce code that meets these criteria, even when using AI, online sources, or any other process.

*The **precise** specification of the problem and your method of solution is the real work (and the hard part);* the implementation is just details.

Implementing in code is precise except when it is not

Using Numpy or Pandas to implement "We found the variance across subjects…" gives two different outcomes:

```
Python 3.9.5 (default, May 18 2021, 12:31:01)
[Clang 10.0.0 ] :: Anaconda, Inc. on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy as np
>>> import pandas as pd
>>> df = pd.DataFrame([-2, 1, 3, -1, 0, 2])
>>> print( df.var() )
0    3.5
dtype: float64
>>> print( np.var(df) )
0    2.916667
dtype: float64
>>> print( np.var(df) * 6/5 )
0    3.5
dtype: float64
```

The variance according to Pandas

The variance according to Numpy

Both definitions are correct in different settings

Implementing in code is precise except when it is not

Using Numpy or Pandas to implement "We found the variance across subjects…" gives two different outcomes:

```
Python 3.9.5 (default, May 18 2021, 12:31:01)
Python 3.11.9 | packaged by conda-forge | (main, Apr 19 2024, 18:34:54) [Cla
ng 16.0.6 ] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy as np
>>> import pandas as pd
>>> df = pd.DataFrame([-2, 1, 3, -1, 0, 2])
>>> print( df.var() )
0    3.5
dtype: float64
>>> print( np.var(df) )
/opt/anaconda3/lib/python3.11/site-packages/numpy/core/fromnumeric.py:3785:
FutureWarning: The behavior of DataFrame.var with axis=None is deprecated, i
n a future version this will reduce over both axes and return a scalar. To r
etain the old behavior, pass axis=0 (or do not pass axis)
  return var(axis=axis, dtype=dtype, out=out, ddof=ddof, **kwargs)
0    2.916667
dtype: float64
```

Making good code is as much checking as creating

Coding is less about knowing things and more about iteratively finding solutions.

Following any systematic process almost always yields better results for less effort, though some processes are better than others.

Trouble shooting is a learnable skill.

https://xkcd.com/627/