1 Jupyter Notebooks (with STATA?!)

1.1 What are Jupyter Notebooks?

- A way to do literate programming
- Provide code and writing/analysis, on a language agnostic platform
  - Meaning that it is not restricted to just one language
  - Currently there are so-called kernels for many languages
  - Including Stata, Python, R, C, Golang, C++, Fortran and more coming!
- Uses the power of Markdown/Latex Math and Code to tell a story and provide an efficient workflow
- Convert into several different formats including Latex, HTML, Presentations etc...
- The Jupyter engine is also available in other text editors such as Atom and VS Code.
- And now available in STATA!

1.2 Under the Hood

- Jupyter Notebooks are written in python and are themselves a JSON document
- Which makes them suited for working on in a browser

1.3 Extensions

- Jupyter can be made to be a full featured IDE (Integrated Development Environment)
- Which really means you can get all kinds of nifty things
  - Autocompletion
  - Multi-cursor support
  - Scratchpad
  - Highlighting a selected word
  - Translation
  - Spellcheck

1.4 Installing Extensions

- In order to do this, we need to go to our conda console and type:

```
conda install -c conda-forge jupyter_contrib_nbextensions
```

- And restart Jupyter

1.5 Markdown

- Using the same idea as in markstat that Oscar showed you before.

1.6 Showing Math

- It is possible to show math
  - $y_{it} = \alpha + \beta \cdot X$

1.7 The Stata Kernel

- This is a relatively new kernel that is implemented by Kyle Barron, Mauricio Cáceres, and other contributors
  - It provides the ability to run code and show graphics, which was previously unavailable for Stata in Jupyter.
- Ironically, even though we are using Stata in these presentations, there are other, free, open-source languages that are just as good (if not more powerful) for which dynamic documents have existed for over a decade.
- As a small nudge towards getting you to try something like R or Python, here's an addendum that Kyle Barron wrote on this State_kernel page:

  As an ardent open-source advocate and someone who actively dislikes using Stata, it somewhat pains me that my work creates value for a proprietary, closed-source program. I hope that this program improves research in a utilitarian way, and shows to new users the scope of the open-source tools that have existed for upwards of a decade.

1.8 Running Code

- In this case we will be using the Stata kernel, so we will have Stata running in the background.

1.9 Stata Kernel Magics

- Many Jupyter kernels have something called magics
  - A way to make certain actions easy without having to write too much code
  - Stata has some magics that make things a little easier

1.10 %browse, %head, %tail

- This has the ability to choose varlist, the number of observations and with `if` statements as well

1.11 %html and %latex

- This allows the rendering of table during export into html or latex, as well as rendering in the notebook (with HTML only)

1.12 %help

- You can use this to get a help file

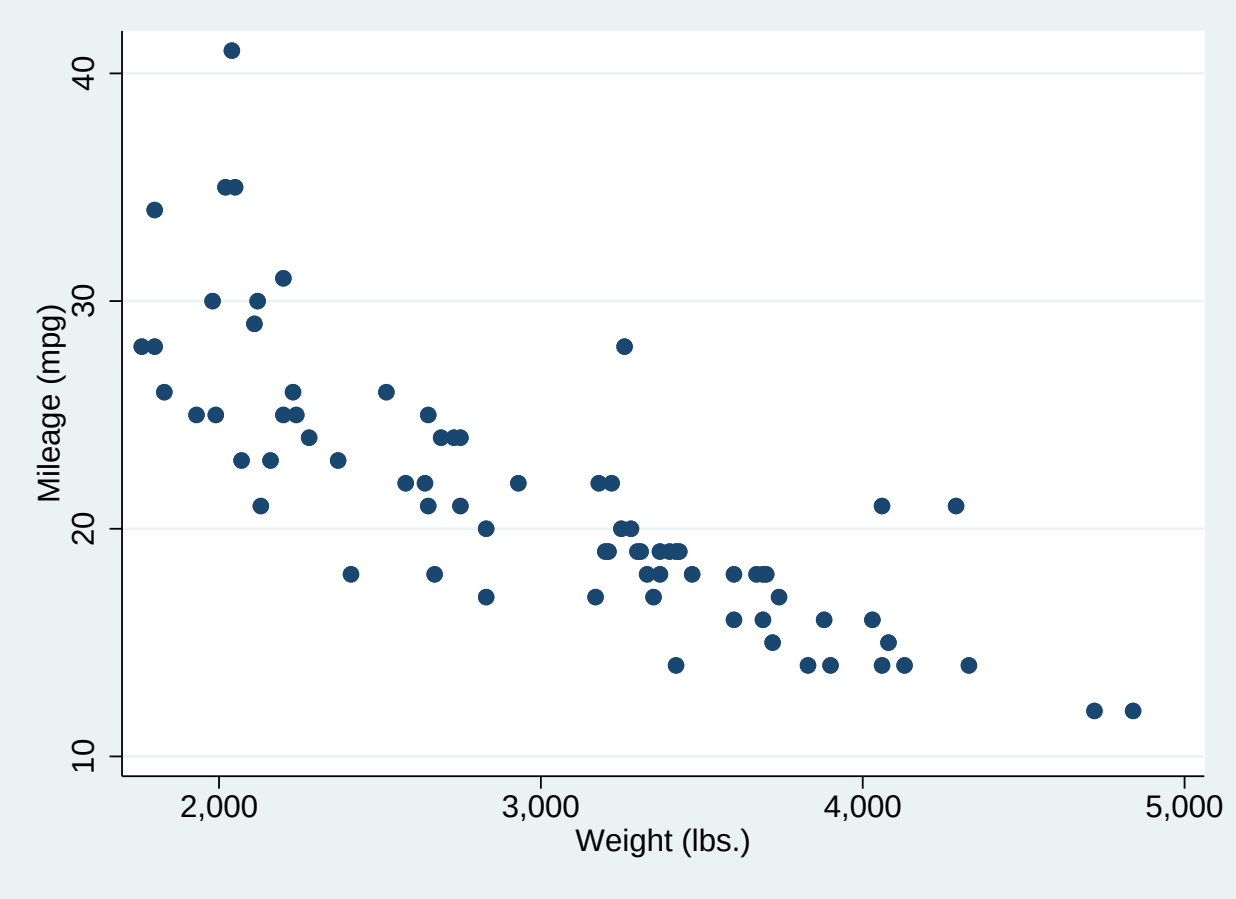

Figure 1.1: A scatter plot

1.13 Exporting

1.14 Using `ipypublish` to Get Publication Ready PDFs

- `ipypublish` is a utility developed for Jupyter Notebooks to make nice looking documents
- To get this working, we need to use `pip`
  - In the conda console, type `pip install ipypublish`
  - Hopefully it'll work
- Doing this requires playing with the JSON code of a cell itself (called the metadata).
- This allows a subsequent PDF output to be processed through latex, without any code cells and with figure and table environments.

1.15 Port-forwarding and setting up Jupyter to work on a server

- Many people might have servers in their universities/organizations that are more powerful than a laptop.
- Jupyter allows the ability to run a notebook locally (on your laptop screen), but using the power of the server.
  - This requires jupyter being installed on the server
  - This isn't a difficult thing to do for a sysadmin, so it's worth finding out whether that's possible

1.16 Setting up jupyter on a server

- The first thing you need to do is log on to the server and start a jupyter instance:

```
jupyter notebook --no-browser --port=8888
```

- This tells the server to start an instance of jupyter, without a browser (we won't need it, nor can a server open up a browser window), in port 8888 (this will be important later)
- For Mac users, you can use `ssh` to finish the process. Just type: `ssh username@host -L 8888:localhost:8888`
- Which will forward your computer 8888 port, to the server's 8888 port.
- For Windows, ssh also exists, but you will need to enable it.
  - head to Settings > Apps and click "Manage optional features" under Apps & features.
  - Click Add a Feature, and find OpenSSH
- Then use the same command as for Macs: `ssh username@host -L 8888:localhost:8888`
- Then go to your browser:
  - `localhost:8888` and you should be taken to a Jupyter page and prompted for a token.
  - You can find this token in the window where you started Jupyter on the server
    - Copy and paste this token into the prompt, and VOILA!
- Now you have Jupyter running on your computer's browser window, but with the power of the server!