Standalone Web Applications

with Jupyter Notebooks

Nicole Brewer
Software Engineer
Scientific Solution Group
Purdue University

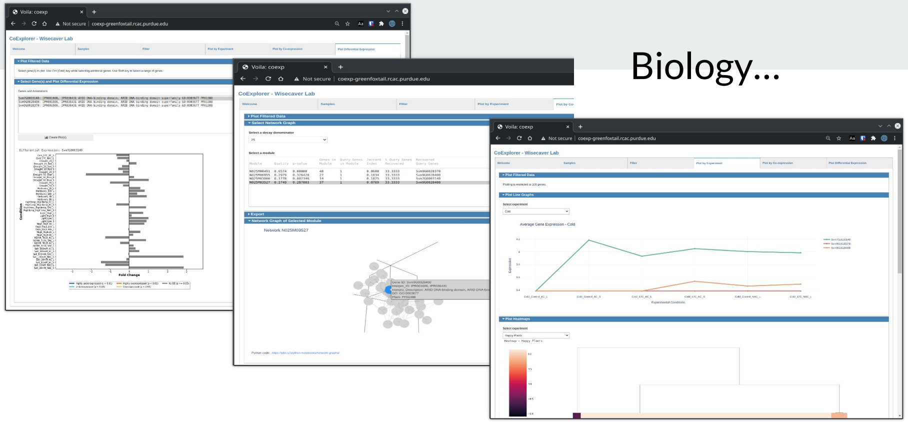
@catch_me_coding

Standalone Web Applications

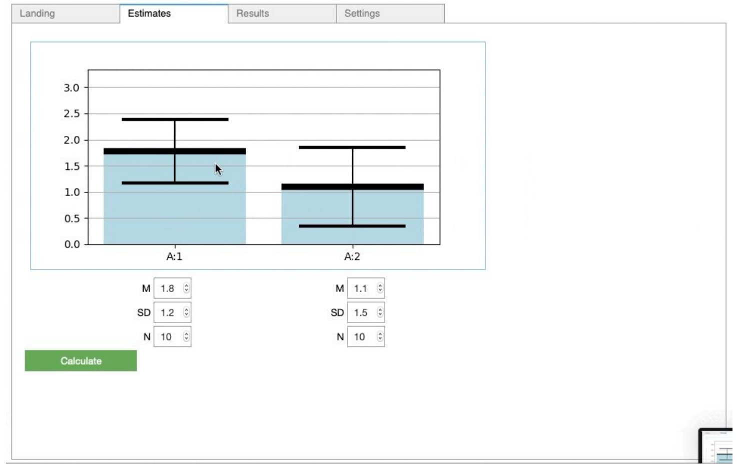
1. Using Jupyter Notebooks is a good idea

2. How to best go about doing it

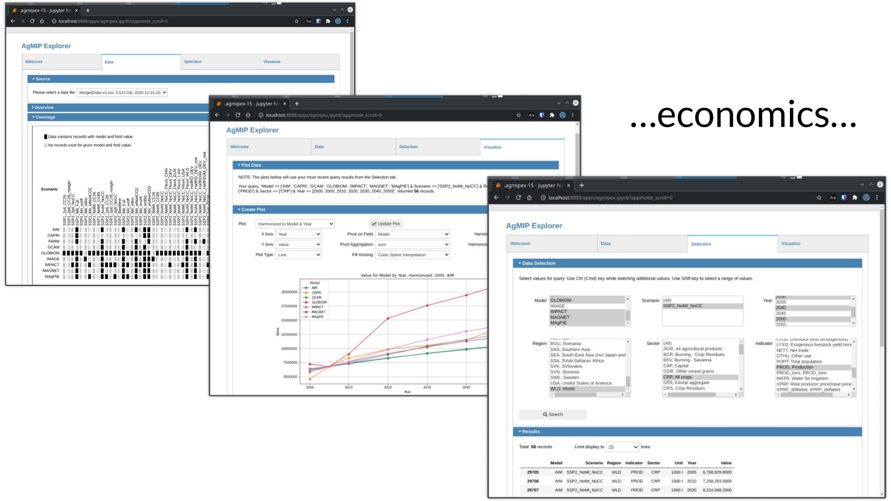
Using Notebooks for web app development is a good idea



Biology…



M	1.8	M	1.1
SD	1.2	SD	1.5
N	10	N	10



Select values for query. Use Ctrl (Cmd) key while selecting additional values. Use Shift key to select a range of values.

Total: 56 records

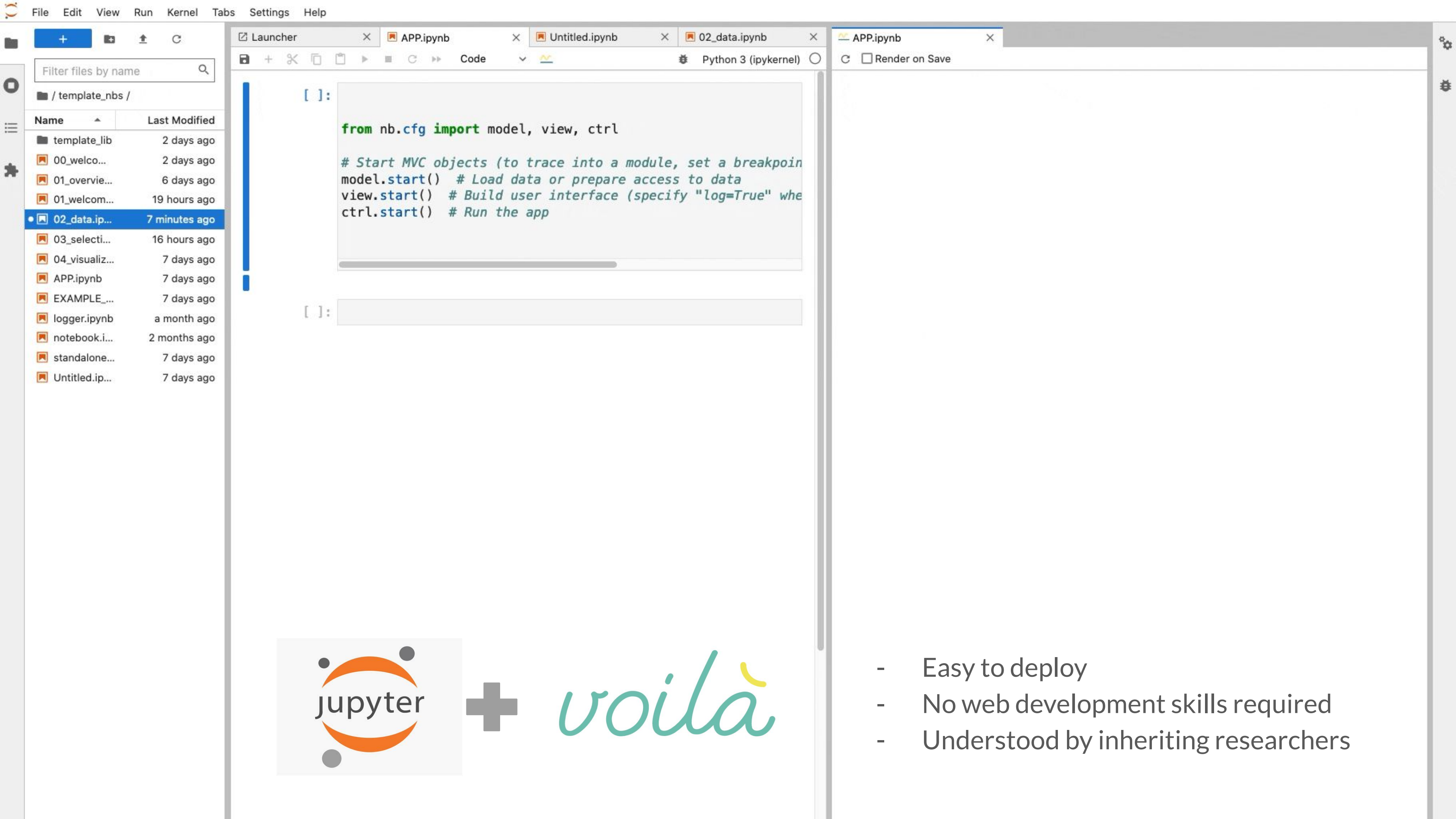
	Model	Scenario	Region	Indicator	Sector	Unit	Year	Value
29705	AIM	SSP2_NoMt_NoCC	WLD	PROD	CRP	1000 t	2005	6,768,829.8000
29706	AIM	SSP2_NoMt_NoCC	WLD	PROD	CRP	1000 t	2010	7,258,283.5000
29707	AIM	SSP2_NoMt_NoCC	WLD	PROD	CRP	1000 t	2020	8,224,068.2000

…economics…

…psychology,

aeronautics,

and more.

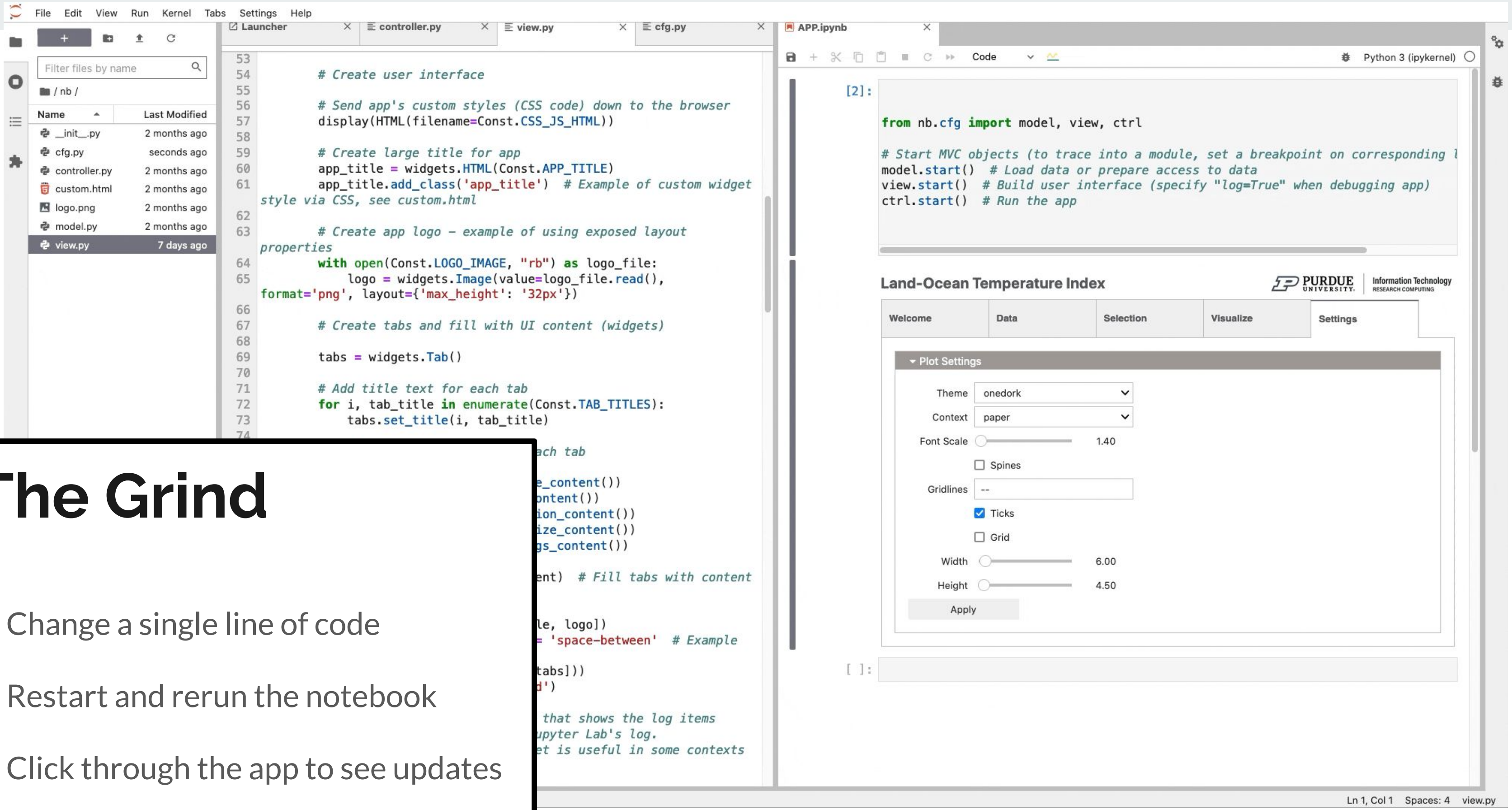


Name	Last Modified
template_lib	2 days ago
00_welco...	2 days ago
01_overvie...	6 days ago
01_welcom...	19 hours ago
02_data.ip...	7 minutes ago
03_selecti...	16 hours ago
04_visualiz...	7 days ago
APP.ipynb	7 days ago
EXAMPLE_...	7 days ago
logger.ipynb	a month ago
notebook.i...	2 months ago
standalone...	7 days ago
Untitled.ip...	7 days ago

```
[ ]:

from nb.cfg import model, view, ctrl

# Start MVC objects (to trace into a module, set a breakpoin
model.start()  # Load data or prepare access to data
view.start()  # Build user interface (specify "log=True" whe
ctrl.start()  # Run the app

[ ]:
```

- Easy to deploy
- No web development skills required
- Understood by inheriting researchers

How to approach development



Name	Last Modified
__init__.py	2 months ago
cfg.py	seconds ago
controller.py	2 months ago
custom.html	2 months ago
logo.png	2 months ago
model.py	2 months ago
view.py	7 days ago

```python
        # Create user interface

        # Send app's custom styles (CSS code) down to the browser
        display(HTML(filename=Const.CSS_JS_HTML))

        # Create large title for app
        app_title = widgets.HTML(Const.APP_TITLE)
        app_title.add_class('app_title')  # Example of custom widget style via CSS, see custom.html

        # Create app logo - example of using exposed layout properties
        with open(Const.LOGO_IMAGE, "rb") as logo_file:
            logo = widgets.Image(value=logo_file.read(), format='png', layout={'max_height': '32px'})

        # Create tabs and fill with UI content (widgets)

        tabs = widgets.Tab()

        # Add title text for each tab
        for i, tab_title in enumerate(Const.TAB_TITLES):
            tabs.set_title(i, tab_title)
```

```
ach tab
e_content())
ontent())
ion_content())
ize_content())
gs_content())
ent)  # Fill tabs with content

le, logo])
= 'space-between'  # Example

tabs]))
d')

 that shows the log items
upyter Lab's log.
et is useful in some contexts
```

[2]:
```python
from nb.cfg import model, view, ctrl

# Start MVC objects (to trace into a module, set a breakpoint on corresponding l
model.start()  # Load data or prepare access to data
view.start()  # Build user interface (specify "log=True" when debugging app)
ctrl.start()  # Run the app
```

Land-Ocean Temperature Index

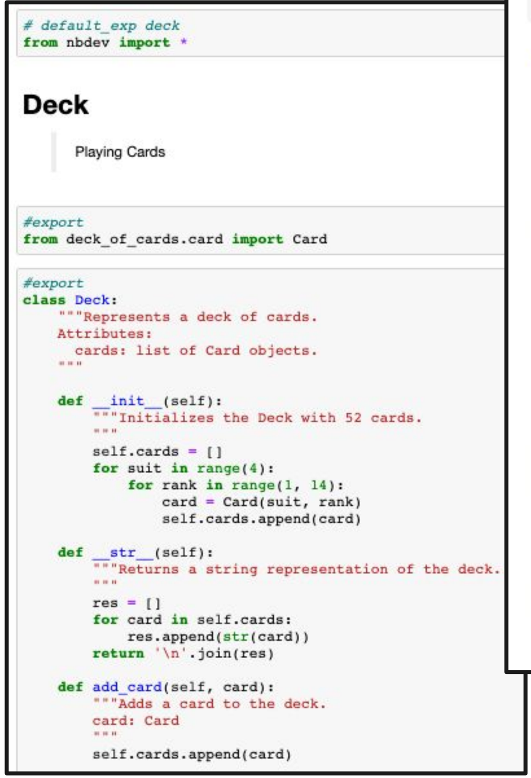
Theme	onedork
Context	paper
Font Scale	1.40
☐ Spines	
Gridlines	--
☑ Ticks	
☐ Grid	
Width	6.00
Height	4.50

The Grind

1. Change a single line of code
2. Restart and rerun the notebook
3. Click through the app to see updates

Literate Programming with nbdev

- Interactive development
- Notebook-friendly merge conflict resolution
- Documentation generation
- Inline testing



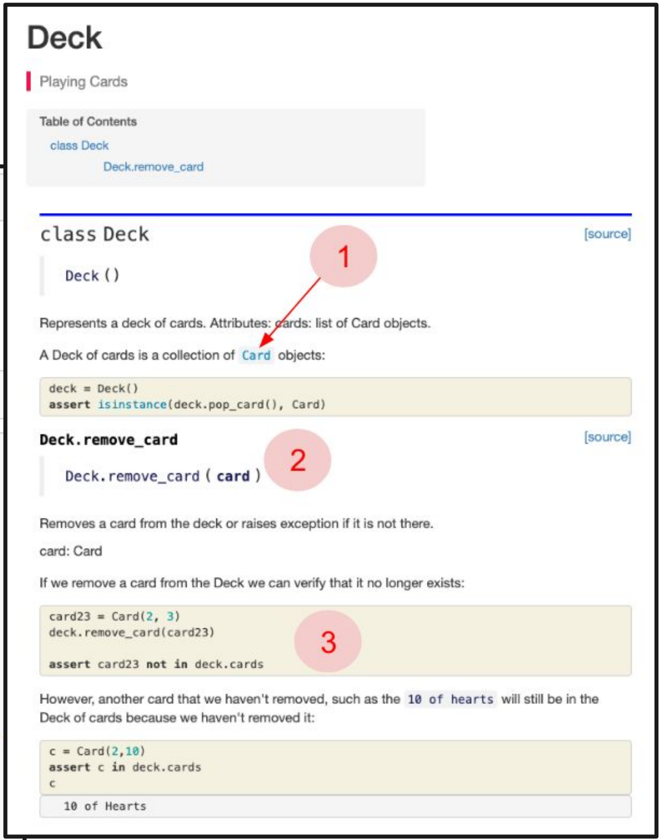
```
# default_exp deck
from nbdev import *
```

Deck

> Playing Cards

```
#export
from deck_of_cards.card import Card
```

```
#export
class Deck:
    """Represents a deck of cards.
    Attributes:
      cards: list of Card objects.
    """

    def __init__(self):
        """Initializes the Deck with 52 cards.
        """
        self.cards = []
        for suit in range(4):
            for rank in range(1, 14):
                card = Card(suit, rank)
                self.cards.append(card)

    def __str__(self):
        """Returns a string representation of the deck.
        """
        res = []
        for card in self.cards:
            res.append(str(card))
        return '\n'.join(res)

    def add_card(self, card):
        """Adds a card to the deck.
        card: Card
        """
        self.cards.append(card)
```



Deck

> Playing Cards

class Deck [source]

> Deck ()

Represents a deck of cards. Attributes: cards: list of Card objects.

A Deck of cards is a collection of `Card` objects:

```
deck = Deck()
assert isinstance(deck.pop_card(), Card)
```

Deck.remove_card [source]

> Deck.remove_card (card)

Removes a card from the deck or raises exception if it is not there.

card: Card

If we remove a card from the Deck we can verify that it no longer exists:

```
card23 = Card(2, 3)
deck.remove_card(card23)

assert card23 not in deck.cards
```

However, another card that we haven't removed, such as the `10 of hearts` will still be in the Deck of cards because we haven't removed it:

```
c = Card(2,10)
assert c in deck.cards
c
```

```
10 of Hearts
```

IDE-like environment with JupyterLab

- Tab-completion
- Notebook debugger
- Linters
- Live update Voila preview
- Variable explorer

nicole-brewer/nbdev_app_template

- Dockerfile for deploying on a composable platform
- Docker-compose for development environment
- Extended version of the nbdev template with instructional notebooks and templates

Give the repo a ★ and watch for updates

UNDER CONSTRUCTION