

Code Notebooks

Module 2: Code Notebooks

In order to make analysis **practically reproducible**, one should strive to make analysis

1. easy to interact with
2. easy to understand

code notebooks help achieve these goals via a **literate programming** format that interweaves

1. text
2. code
3. output

all together.

Popular Notebooks and Software

Often, code notebooks and their editing software are discussed as a single object. However one may separate

1. the notebook file format, from
2. the software used to interact with that format

Two most popular notebook formats/software:

1. “jupyter”: **jupyter lab** software and .ipynb format
2. “quarto”: **Rstudio** software and .qmd format

We will give a brief overview of these, and then turn to comparing them and their advantages for reproducibility.

Jupyter Lab Example

An example of **jupyter lab**:

The screenshot shows a Jupyter Lab interface with the following elements:

- Rich Text:** A section labeled "rich text" with a red box and arrow.
- Code:** A section labeled "code" with a red box and arrow pointing to the code cell.
- Suppressed Output:** A section labeled "suppressed output" with a red box and arrow pointing to the ellipsis (...).
- Output:** A section labeled "output" with a red box and arrow pointing to the resulting tibble.
- Code Cell Content:** [1]:
library("palmerpenguins")
library('tidyverse')
...
[2]: penguins %>% sample_n(3)
- Output Content:** A tibble: 3 × 8
species island bill_length_mm bill_depth_mm flipper_length_mm body_mass_g sex year
<fct> <fct> <dbl> <dbl> <int> <int> <fct> <int>
Adelie Torgersen 42.5 20.7 197 4500 male 2007
Adelie Biscoe 41.1 18.2 192 4050 male 2008

Text in markdown

jupyter allows text to be written in `markdown` which is a light-weight markup language.

More-or-less: if you can display it on a webpage, you can write it in `markdown`. (Additionally, one can directly embed `html`)

One can also use extended markdown languages like `myst` which enables features like references, figures, bibliographies, . . .

In any case, jupyter's `markdown` enables rich-text commentary on **both the code and the output**

Basic Markdown

The screenshot shows a Jupyter Notebook interface with the following details:

- Title Bar:** "intro_to_jupyter.ipynb" (2) - Jupyter
- URL:** localhost:8889/fab/tree/intro_to_jupyter.ipynb
- Toolbar:** File, Edit, View, Run, Kernel, Tabs, Settings, Help
- Code Cell:** "intro_to_jupyter.ipynb" contains the following text:

```
# heading level 1
## heading level 2
### heading level 3
```
- Output Cells:** Three heading cells are displayed:
 - heading level 1
 - heading level 2
 - heading level 3
- Status Bar:** Simple, 0, \$ 2, R | Idle, Mode: Command, Ln 1, Col 1, intro_to_jupyter.ipynb

Basic Markdown

The screenshot shows a Jupyter Notebook interface with the following content:

```
intro_to_jupyter.ipynb
File Edit View Run Kernel Tabs Settings Help
intro_to_jupyter.ipynb Raw git
**bold text** displays bold text
*italic text* displays italic text
> block quote
    block quote
```

The notebook title is "intro_to_jupyter.ipynb". The toolbar includes File, Edit, View, Run, Kernel, Tabs, Settings, and Help. The top bar shows the file name and a raw mode button. The main area contains three code blocks demonstrating bold, italic, and block quotes.

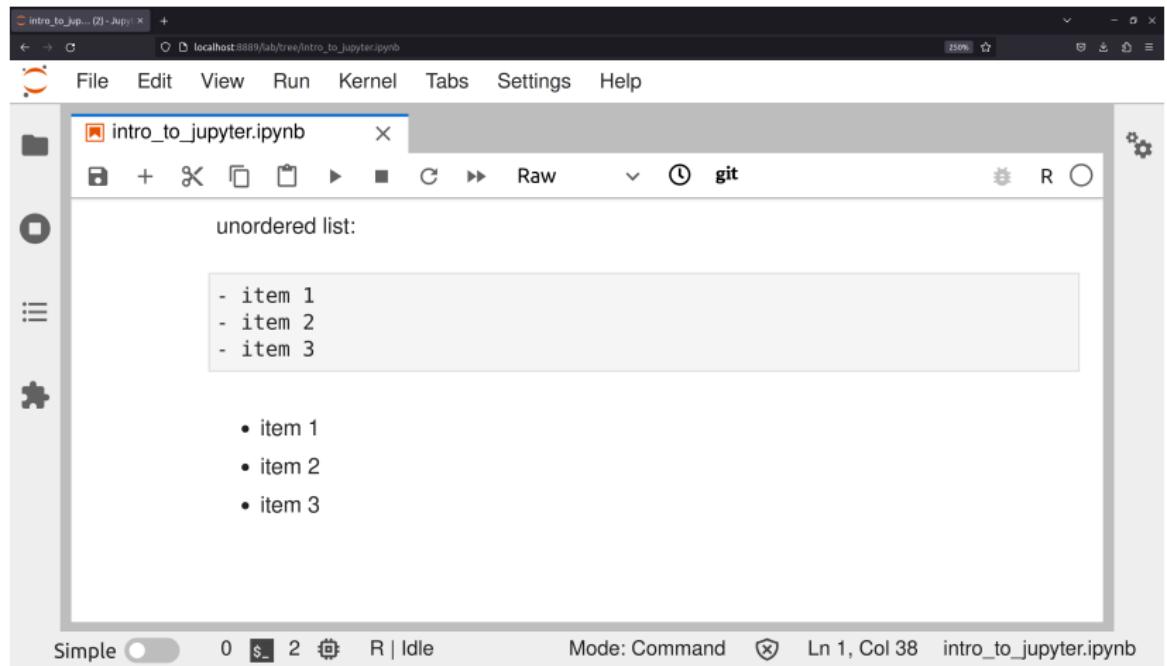
Basic Markdown

The screenshot shows a Jupyter Notebook interface with the following details:

- Title Bar:** "intro_to_jupyter.ipynb" (2) - Jupyter
- URL:** localhost:8889/fab/tree/intro_to_jupyter.ipynb
- Toolbar:** File, Edit, View, Run, Kernel, Tabs, Settings, Help
- Section Header:**

ordered lists:
- Code Cell 1:** A code cell containing the following ordered list:
 1. item 1
 1. item 2
 1. item 3
- Code Cell 2:** A code cell containing the following ordered list:
 1. item 1
 2. item 2
 3. item 3
- Status Bar:** Simple, 0 \$ 2 R | Idle, Mode: Command, Ln 1, Col 38, intro_to_jupyter.ipynb

Basic Markdown



Basic Markdown

The screenshot shows a Jupyter Notebook interface with the following details:

- Title Bar:** intro_to_jupyter.ipynb (2) - Jupyter
- URL:** localhost:8889/lab/tree/intro_to_jupyter.ipynb
- Toolbar:** File, Edit, View, Run, Kernel, Tabs, Settings, Help
- Code Cell:** `code`
displays code
- Text Cell:** you can link to webpages like this:
[link title](http://www.example.com/)
link title
- Status Bar:** Simple 0 \$ 2 R | Idle Mode: Command Ln 1, Col 1 intro_to_jupyter.ipynb

Basic Markdown

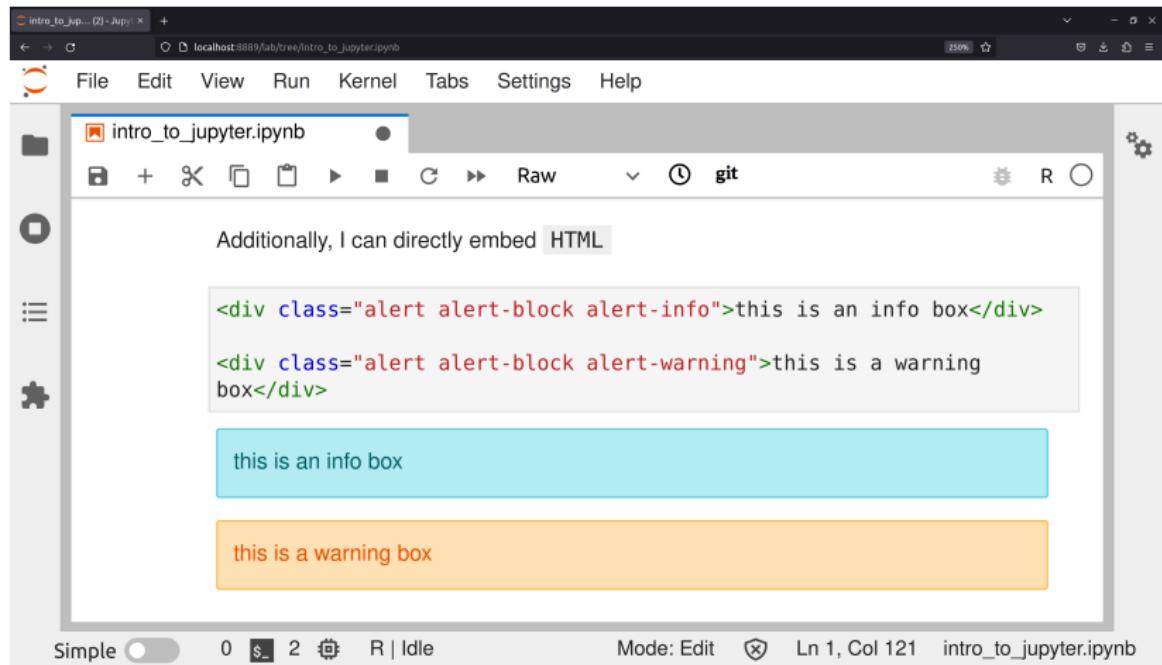
The screenshot shows a Jupyter Notebook interface with the following details:

- Title Bar:** "intro_to_jupyter.ipynb" (2) - Jupyter
- URL:** localhost:8889/fab/tree/intro_to_jupyter.ipynb
- Toolbar:** File, Edit, View, Run, Kernel, Tabs, Settings, Help
- Section Header:**

intro_to_jupyter.ipynb
- Text Editor:** "and embed images like this:" followed by a code block:

```
![alt](https://upload.wikimedia.org/wikipedia/commons/thumb/7/73/A_doing_aisatsu_kitten_%28Flickr%29.jpg/320px-A_doing_aisatsu_kitten_%28Flickr%29.jpg)
```
- Image Preview:** A small thumbnail of an orange and white kitten looking up.
- Bottom Status Bar:** Simple, 0, \$, 2, R | Idle, Mode: Command, Ln 1, Col 1, intro_to_jupyter.ipynb

Basic Markdown



Basic Markdown

The screenshot shows a Jupyter Notebook interface with the following details:

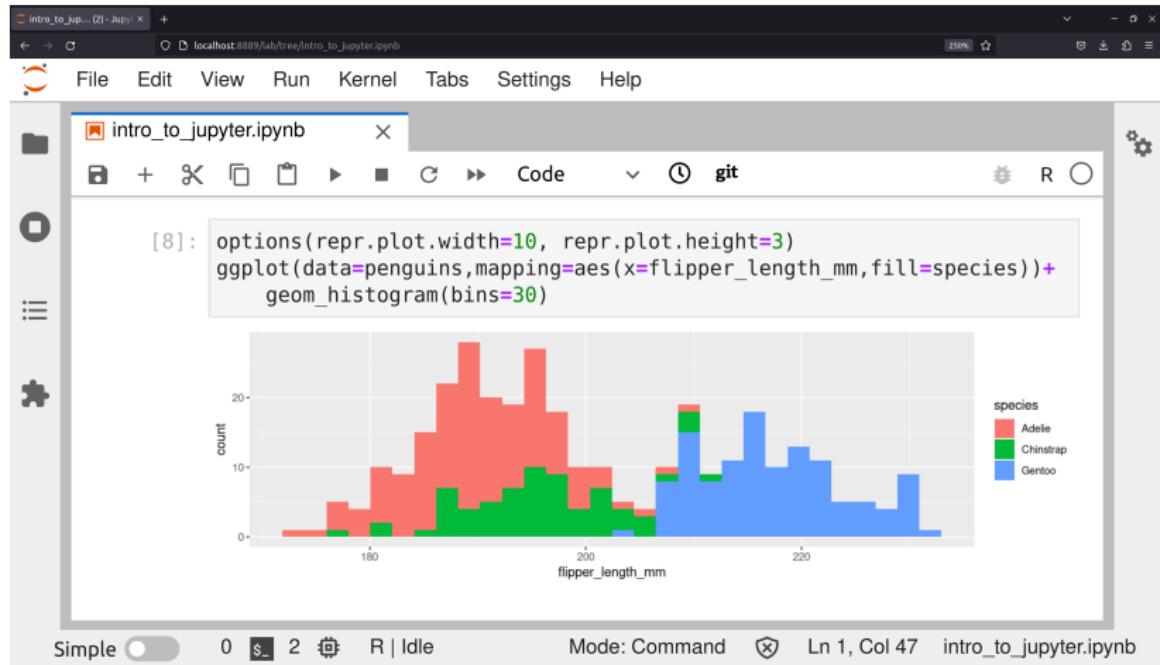
- Title Bar:** The title bar displays "intro_to_jupyter.ipynb" and "localhost:8889/lab/tree/intro_to_jupyter.ipynb".
- Toolbar:** The toolbar includes File, Edit, View, Run, Kernel, Tabs, Settings, Help, and a Git icon.
- Header Bar:** The header bar shows the notebook name "intro_to_jupyter.ipynb" and a cell type indicator (circle).
- Content Area:** The main area contains the text "I can also embed *ETEX* mathematics type" followed by a code block:

```
$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$
```

which is rendered as the quadratic formula:
$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$
- Sidebar:** On the left, there are icons for file operations (New, Open, Save, etc.) and a vertical list of notebook files.
- Bottom Bar:** The bottom bar shows "Simple" mode, cell count (0), a refresh button, and the status "R | Idle". It also displays the mode as "Command", line 1, column 1, and the notebook name "intro_to_jupyter.ipynb".

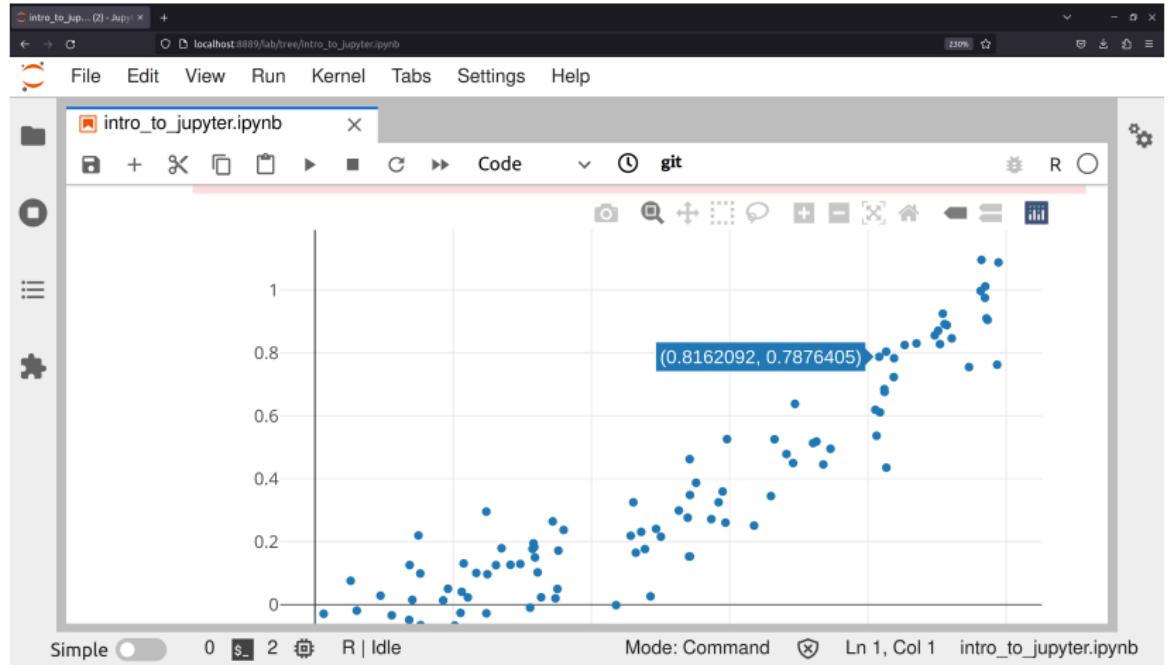
Code: Basic Output

Interweaved with the rich-text markdown commentary, one intersperses **code and in-line output**, e.g.,



Code: Interactive Widgets

One can also embed **interactive widgets**, though this is somewhat notebook/language-specific. For example, in R one can use `htmlwidgets` or `plotly`, e.g.,



Code: Languages

There are **many** language backends that jupyter can use. (These are called **kernels** in jupyter-speak). Jupyter lists well over 100 available kernels here including kernels for:

- python,

Code: Languages

There are **many** language backends that jupyter can use. (These are called **kernels** in jupyter-speak). Jupyter lists well over 100 available kernels here including kernels for:

- python,
- r,

Code: Languages

There are **many** language backends that jupyter can use. (These are called **kernels** in jupyter-speak). Jupyter lists well over 100 available kernels here including kernels for:

- python,
- r,
- julia,

Code: Languages

There are **many** language backends that jupyter can use. (These are called **kernels** in jupyter-speak). Jupyter lists well over 100 available kernels here including kernels for:

- python,
- r,
- julia,
- stata,

Code: Languages

There are **many** language backends that jupyter can use. (These are called **kernels** in jupyter-speak). Jupyter lists well over 100 available kernels here including kernels for:

- python,
- r,
- julia,
- stata,
- sql,

Code: Languages

There are **many** language backends that jupyter can use. (These are called **kernels** in jupyter-speak). Jupyter lists well over 100 available kernels here including kernels for:

- python,
- r,
- julia,
- stata,
- sql,
- octave,

Code: Languages

There are **many** language backends that jupyter can use. (These are called **kernels** in jupyter-speak). Jupyter lists well over 100 available kernels here including kernels for:

- python,
- r,
- julia,
- stata,
- sql,
- octave,
- matlab,

Code: Languages

There are **many** language backends that jupyter can use. (These are called **kernels** in jupyter-speak). Jupyter lists well over 100 available kernels here including kernels for:

- python,
- r,
- julia,
- stata,
- sql,
- octave,
- matlab,
- java,

Code: Languages

There are **many** language backends that jupyter can use. (These are called **kernels** in jupyter-speak). Jupyter lists well over 100 available kernels here including kernels for:

- python,
- r,
- julia,
- stata,
- sql,
- octave,
- matlab,
- java,
- go,

Code: Languages

There are **many** language backends that jupyter can use. (These are called **kernels** in jupyter-speak). Jupyter lists well over 100 available kernels here including kernels for:

- python,
- r,
- julia,
- stata,
- sql,
- octave,
- matlab,
- java,
- go,
- C,

Code: Languages

There are **many** language backends that jupyter can use. (These are called **kernels** in jupyter-speak). Jupyter lists well over 100 available kernels here including kernels for:

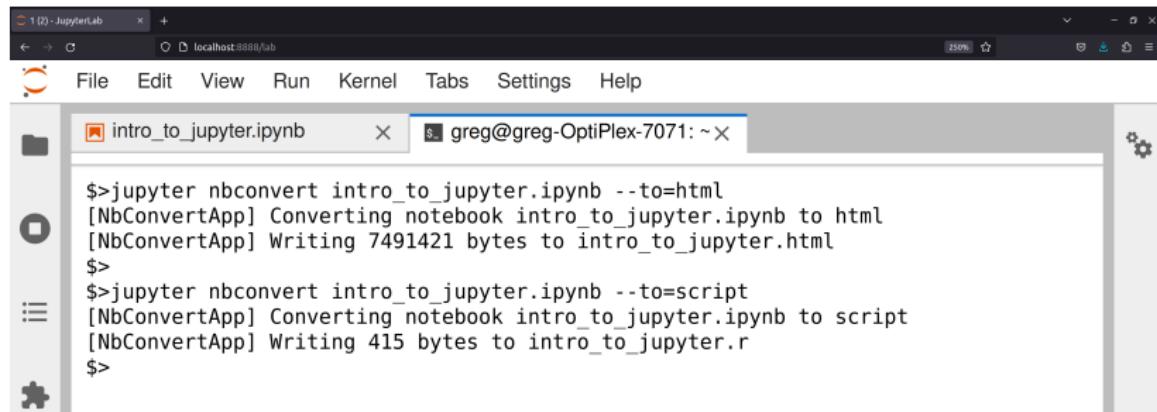
- python,
- r,
- julia,
- stata,
- sql,
- octave,
- matlab,
- java,
- go,
- C,
- ...

Exporting

One can **export** a **.ipynb** notebook using **jupyter** to many different formats like: **html**, **markdown** **pdf**, **reveal.js** **html slides**, . . . , and even an executable script.

This can all be done using the command

```
jupyter nbconvert notebook.ipynb --to=format
```

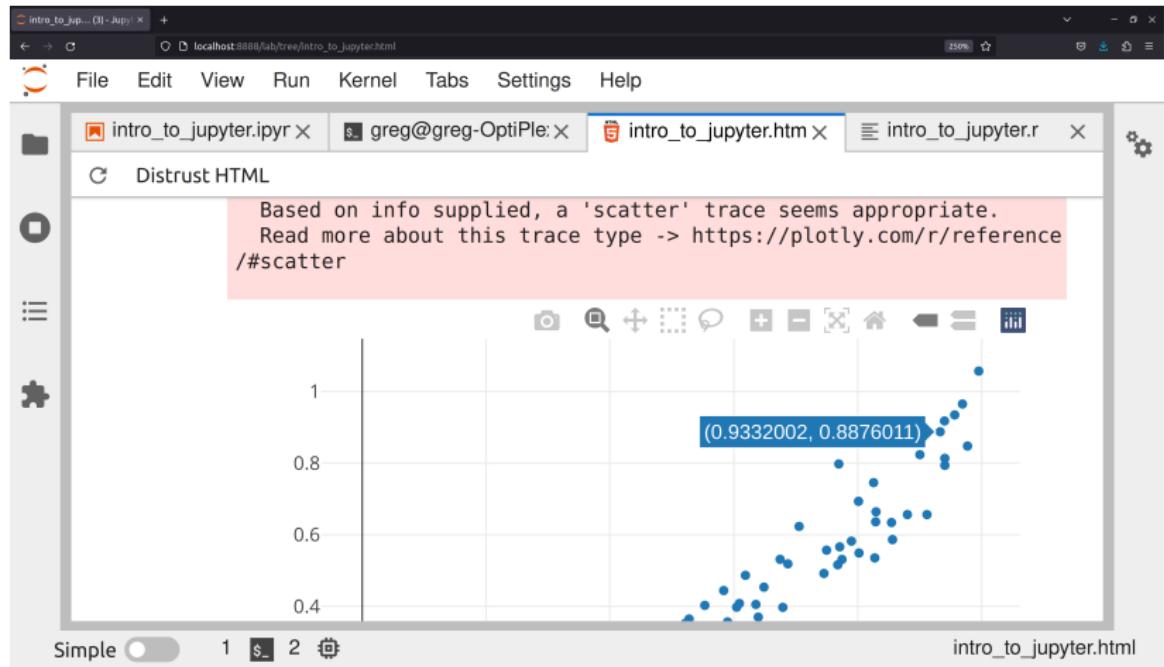


The screenshot shows the JupyterLab interface with a terminal window open. The terminal title is "intro_to_jupyter.ipynb". The user has run two commands: "jupyter nbconvert intro_to_jupyter.ipynb --to=html" and "jupyter nbconvert intro_to_jupyter.ipynb --to=script". The terminal output indicates that the notebook was converted to an HTML file ("Writing 7491421 bytes to intro_to_jupyter.html") and a script file ("Writing 415 bytes to intro_to_jupyter.r").

```
$>jupyter nbconvert intro_to_jupyter.ipynb --to=html
[NbConvertApp] Converting notebook intro_to_jupyter.ipynb to html
[NbConvertApp] Writing 7491421 bytes to intro_to_jupyter.html
$>
$>jupyter nbconvert intro_to_jupyter.ipynb --to=script
[NbConvertApp] Converting notebook intro_to_jupyter.ipynb to script
[NbConvertApp] Writing 415 bytes to intro_to_jupyter.r
$>
```

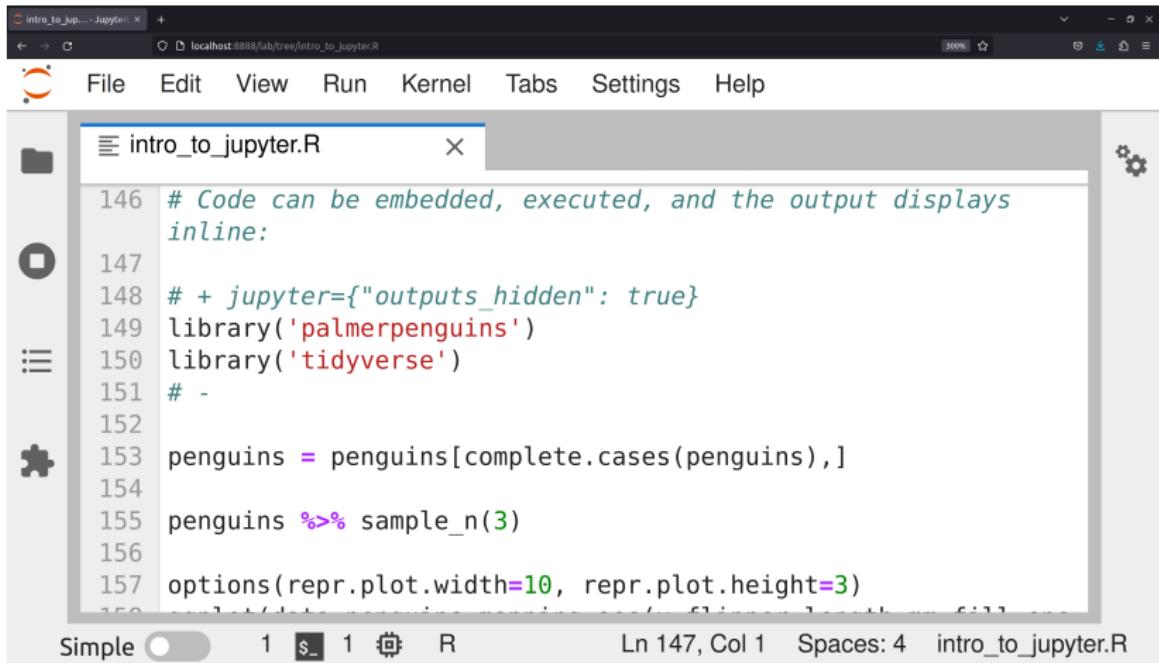
Exporting: HTML

Nicely, for a HTML export many of the interactive widgets still work!



Exporting: Script

Exporting as a script gives us basic executable script



The screenshot shows a Jupyter Notebook interface with the following details:

- Title Bar:** intro_to_jupyter.R - Jupyter
- URL:** localhost:8888/lab/tree/intro_to_jupyter.R
- Toolbar:** File, Edit, View, Run, Kernel, Tabs, Settings, Help
- Code Area:** The notebook cell contains R code:

```
146 # Code can be embedded, executed, and the output displays
147 # inline:
148 # + jupyter={"outputs_hidden": true}
149 library('palmerpenguins')
150 library('tidyverse')
151 #
152 penguins = penguins[complete.cases(penguins),]
153 penguins %>% sample_n(3)
154
155 options(repr.plot.width=10, repr.plot.height=3)
```
- Sidebar:** Includes icons for file operations (New, Open, Save, etc.) and a settings gear icon.
- Status Bar:** Simple (radio button), Line 147, Column 1, Spaces: 4, intro_to_jupyter.R

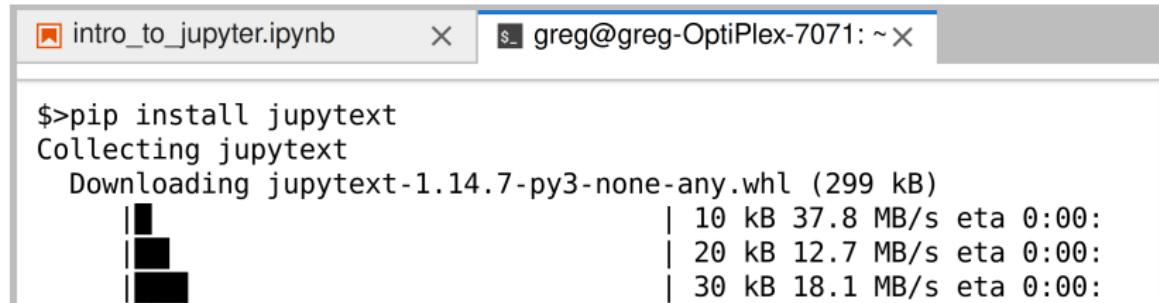
Notebook Interoperability

One can also use third-party tools to convert/maintain versions in **other notebook formats**.

For this, we find the tool `jupytext` to be invaluable.

We can install via

```
pip install jupytext
```



A screenshot of a terminal window titled "intro_to_jupyter.ipynb". The title bar also shows the user "greg@greg-OptiPlex-7071: ~". The terminal content shows the command \$>pip install jupytext being run, followed by the output of the pip command. It includes the message "Collecting jupytext", "Downloading jupytext-1.14.7-py3-none-any.whl (299 kB)", and a progress bar with three vertical bars indicating the download status. The progress bar has three segments: the first segment is at 10 KB with a speed of 37.8 MB/s and an eta of 0:00; the second segment is at 20 KB with a speed of 12.7 MB/s and an eta of 0:00; the third segment is at 30 KB with a speed of 18.1 MB/s and an eta of 0:00.

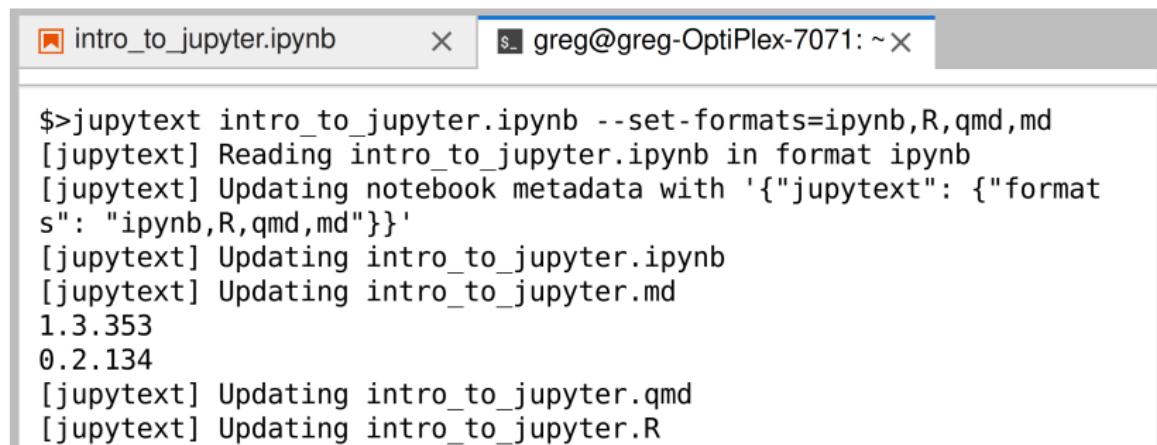
```
$>pip install jupytext
Collecting jupytext
  Downloading jupytext-1.14.7-py3-none-any.whl (299 kB)
    | 10 kB 37.8 MB/s eta 0:00:
    | 20 kB 12.7 MB/s eta 0:00:
    | 30 kB 18.1 MB/s eta 0:00:
```

Notebook Interoperability

jupytext **synchronously** propagates changes in the jupyter notebook (when the .ipynb is saved) to other formats like markdown, annotated scripts, rmarkdown, quarto, ...

via the command

```
jupytext notebook.ipynb --set-formats=format1,format2,...
```



The screenshot shows a terminal window with a light gray background. At the top, there is a header bar with a small orange icon on the left, followed by the text "intro_to_jupyter.ipynb" and a close button "X". To the right of the header is the user information "greg@greg-OptiPlex-7071: ~" and another close button. Below the header, the terminal content starts with a command line prompt "\$>". The command executed is "jupytext intro_to_jupyter.ipynb --set-formats=ipynb,R,qmd,md". The terminal then displays several lines of output from the [jupytext] module, indicating the reading of the notebook, updating of metadata, and the creation of new files in ipynb, R, qmd, and md formats. At the bottom of the terminal window, there are two lines of text: "1.3.353" and "0.2.134", which are likely version numbers for the jupytext package.

```
$>jupytext intro_to_jupyter.ipynb --set-formats=ipynb,R,qmd,md
[jupytext] Reading intro_to_jupyter.ipynb in format ipynb
[jupytext] Updating notebook metadata with '{"jupytext": {"formats": "ipynb,R,qmd,md"}}'
[jupytext] Updating intro_to_jupyter.ipynb
[jupytext] Updating intro_to_jupyter.md
1.3.353
0.2.134
[jupytext] Updating intro_to_jupyter.qmd
[jupytext] Updating intro_to_jupyter.R
```

Jupyter Change

Changes are **synchronously** propagated from .ipynb to the other formats and vice-versa. (Just be wary of editing two files at once!)

The screenshot shows a Jupyter interface with two tabs open:

- intro_to_jupyter.ipynb**: This tab is active. It contains Python code:

```
166 at =
167 data.frame(x=runif(100))
168 df$y =
169 df$x^2+rnorm(100,0,.1)
170 p <- plot_ly(x = df$x, y
171 = df$y, mode = "markers")
172 embed_notebook(p)
173 # This is a change
```
- intro_to_jupyter.R**: This tab is visible in the background. It contains R code:

```
166 at =
167 data.frame(x=runif(100))
168 df$y =
169 df$x^2+rnorm(100,0,.1)
170 p <- plot_ly(x = df$x, y
171 = df$y, mode = "markers")
172 embed_notebook(p)
173 # This is a change
```

A red box highlights a note in the IPython code: "ied, a 'scatter' trace seems appropriate. Read more about this trace type -> <http://plotly.com/r/reference/#scatter>". A blue box highlights the text "This is a change" in the IPython code.

Exporting and Interoperability

- **Exporting** via nbconvert mostly immortalizes analysis for display e.g. as HTML or a PDF

Exporting and Interoperability

- **Exporting** via nbconvert mostly immortalizes analysis for display e.g. as HTML or a PDF
- We can also export executable scripts and (via tools like jupytext) other notebook formats for

Exporting and Interoperability

- **Exporting** via nbconvert mostly immortalizes analysis for display e.g. as HTML or a PDF
- We can also export executable scripts and (via tools like jupytext) other notebook formats for
 - deployment in **production**,

Exporting and Interoperability

- **Exporting** via nbconvert mostly immortalizes analysis for display e.g. as HTML or a PDF
- We can also export executable scripts and (via tools like jupytext) other notebook formats for
 - deployment in **production**,
 - deployment in a **non-interactive cluster**,

Exporting and Interoperability

- **Exporting** via nbconvert mostly immortalizes analysis for display e.g. as HTML or a PDF
- We can also export executable scripts and (via tools like jupytext) other notebook formats for
 - deployment in **production**,
 - deployment in a **non-interactive cluster**,
 - or making analysis more amenable for **version control**

Exporting and Interoperability

- **Exporting** via nbconvert mostly immortalizes analysis for display e.g. as HTML or a PDF
- We can also export executable scripts and (via tools like jupytext) other notebook formats for
 - deployment in **production**,
 - deployment in a **non-interactive cluster**,
 - or making analysis more amenable for **version control**
- Exporting to display formats and other notebook formats also makes them more **sharable** and thus analysis more easily **reproducible**

Exporting and Interoperability

- **Exporting** via nbconvert mostly immortalizes analysis for display e.g. as HTML or a PDF
- We can also export executable scripts and (via tools like jupytext) other notebook formats for
 - deployment in **production**,
 - deployment in a **non-interactive cluster**,
 - or making analysis more amenable for **version control**
- Exporting to display formats and other notebook formats also makes them more **sharable** and thus analysis more easily **reproducible**
 - we provide same analysis in many notebook formats so users have choice

Exporting and Interoperability

- **Exporting** via nbconvert mostly immortalizes analysis for display e.g. as HTML or a PDF
- We can also export executable scripts and (via tools like jupytext) other notebook formats for
 - deployment in **production**,
 - deployment in a **non-interactive cluster**,
 - or making analysis more amenable for **version control**
- Exporting to display formats and other notebook formats also makes them more **sharable** and thus analysis more easily **reproducible**
 - we provide same analysis in many notebook formats so users have choice
 - this is probably a good final step before sharing analysis

quarto: latest R studio notebook format

- quarto is the latest notebook format from Posit (RStudio).

quarto: latest R studio notebook format

- quarto is the latest notebook format from Posit (RStudio).
- Largely, it supplants R notebooks and R markdown formats (and is back-compatable).

quarto: latest R studio notebook format

- quarto is the latest notebook format from Posit (RStudio).
- Largely, it supplants R notebooks and R markdown formats (and is back-compatable).
- Technically, quarto is a command line publishing tool. This means:

quarto: latest R studio notebook format

- quarto is the latest notebook format from Posit (RStudio).
- Largely, it supplants R notebooks and R markdown formats (and is back-compatable).
- Technically, quarto is a command line publishing tool. This means:
 - can be run via the command line

quarto: latest R studio notebook format

- quarto is the latest notebook format from Posit (RStudio).
- Largely, it supplants R notebooks and R markdown formats (and is back-compatable).
- Technically, quarto is a command line publishing tool. This means:
 - can be run via the command line
 - don't need R or Rstudio **at all** to run quarto

quarto: latest R studio notebook format

- quarto is the latest notebook format from Posit (RStudio).
- Largely, it supplants R notebooks and R markdown formats (and is back-compatable).
- Technically, quarto is a command line publishing tool. This means:
 - can be run via the command line
 - don't need R or Rstudio **at all** to run quarto
 - can be used to weave documents in python or julia (can even use a jupyter back-end)

quarto: latest R studio notebook format

- quarto is the latest notebook format from Posit (RStudio).
- Largely, it supplants R notebooks and R markdown formats (and is back-compatable).
- Technically, quarto is a command line publishing tool. This means:
 - can be run via the command line
 - don't need R or Rstudio **at all** to run quarto
 - can be used to weave documents in python or julia (can even use a jupyter back-end)
- Nonetheless, still has great integration in Rstudio with a nice WYSIWYG editor.

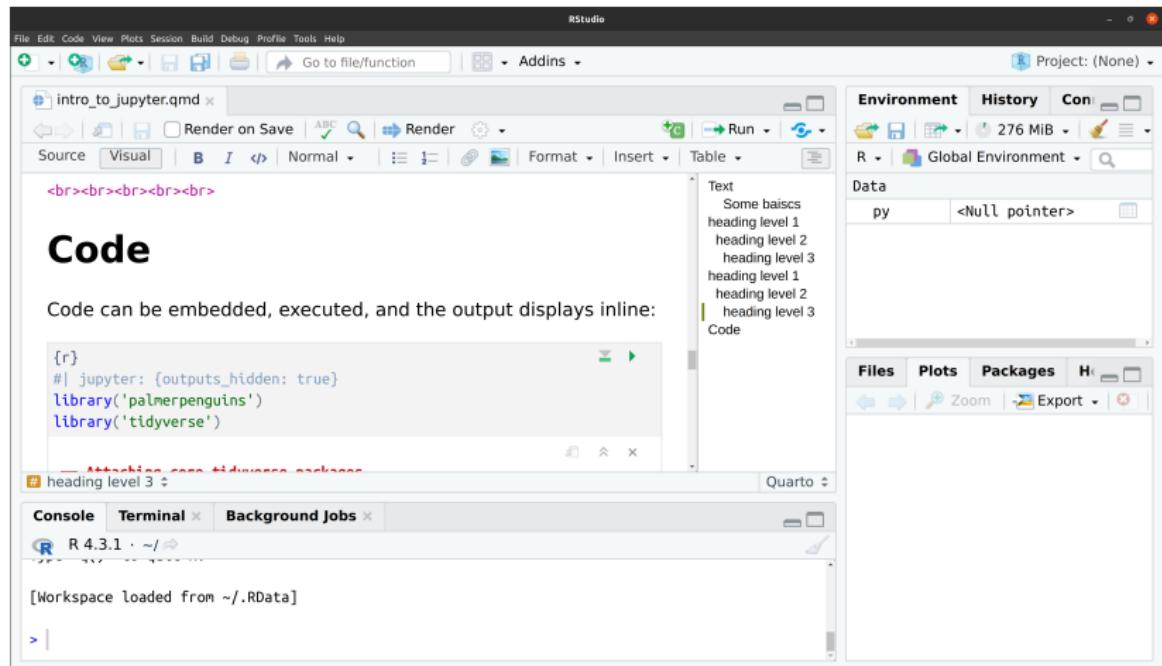
quarto: latest R studio notebook format

- quarto is the latest notebook format from Posit (RStudio).
- Largely, it supplants R notebooks and R markdown formats (and is back-compatable).
- Technically, quarto is a command line publishing tool. This means:
 - can be run via the command line
 - don't need R or Rstudio **at all** to run quarto
 - can be used to weave documents in python or julia (can even use a jupyter back-end)
- Nonetheless, still has great integration in Rstudio with a nice WYSIWYG editor.
- Rstudio/quarto are a similar, nice alternative to jupyter lab/jupyter notebooks (in my opinion)

quarto examples

quarto uses the markdown-like .qmd format (an evolution of .rmd, .md).

Rstudio visual editor:



quarto examples

quarto uses the markdown-like .qmd format (an evolution of .rmd, .md).

Rstudio visual markdown-like source:

The screenshot shows the RStudio interface with a Quarto document open. The left pane displays the Quarto code:

```
140
141 # Code
142
143 Code can be embedded, executed, and the output displays inline:
144
145 ```{r}
146 #| jupyter: {outputs_hidden: true}
147 library('palmerpenguins')
148 library('tidyverse')
149 ```
150
151 ```{r}
152 penguins = penguins[complete.cases(penguins),]
153 ```

173:13 C Chunk 5 :
```

The right pane shows the rendered output of the code, including:

- Text: Some basics
- heading level 1
- heading level 2
- heading level 3
- heading level 1
- heading level 2
- heading level 3

The bottom pane shows the R console output:

```
R 4.3.1 · ~/ ↘
[Workspace loaded from ~/.RData]

> |
```

quarto rendering

- quarto can render documents to a static display format (e.g. HTML, PDF, ...) either via command line or GUI (e.g. Rstudio calls quarto CLI)

```
#> quarto render intro_to_jupyter.qmd
Starting ir kernel...Done
Executing 'intro_to_jupyter.ipynb'
  Cell 1/5...Done
...
Output created: intro_to_jupyter.html
```

- quarto can also render jupyter .ipynb notebooks

```
#> quarto render intro_to_jupyter.ipynb --execute
...
Output created: intro_to_jupyter.html
```

- default: render does not **re-run** code when rendering .ipynb, so you'll need to --execute flag if you want to re-run code before rendering.

quarto preview and convert

- quarto can also preview documents
- this maintains a running background server and displays a real-time updated preview of the document in the web browser

```
#> quarto preview intro_to_jupyter.qmd
Executing 'intro_to_jupyter.ipynb'
Cell 1/5...Done
...
Watching files for changes
Browse at http://localhost:6971/
```

quarto has a command to convert from .qmd to .ipynb

```
#>quarto convert intro_to_jupyter.ipynb --output quarto_nb.qmd
Converted to quarto_nb.qmd
```

and from .ipynb to .qmd

```
#>quarto convert quarto_nb.qmd --output jupyter_nb.ipynb
Converted to jupyter_nb.ipynb
```

quarto and jupyter

quarto/Rstudio and jupyter/jupyter lab are both great notebook formats/tools. One major difference:

- quarto uses markdown-like .qmd while jupyter uses JSON .ipynb
 - .qmd are probably easier for version control (e.g. git) and other editors (e.g. VSCode)
- quarto is more geared towards publishing polished formats (e.g. figures, tables, references, . . .)
 - myst extension to markdown can enable this for jupyter
- .qmd doesn't store output, .ipynb encodes/stores output
- **A useful idiom:** do your analysis in jupyter and mirror into several formats e.g. ipynb, md, R, qmd, . . .

Notebooks and Reproducibility

Why do notebooks help **reproducibility**:

1. literate programming: interweaving code/commentary/output

Notebooks and Reproducibility

Why do notebooks help **reproducibility**:

1. literate programming: interweaving code/commentary/output
 - allows rich commentary on code, output

Notebooks and Reproducibility

Why do notebooks help **reproducibility**:

1. literate programming: interweaving code/commentary/output
 - allows rich commentary on code, output
 - develops a narrative that is easy to read

Notebooks and Reproducibility

Why do notebooks help **reproducibility**:

1. literate programming: interweaving code/commentary/output
 - allows rich commentary on code, output
 - develops a narrative that is easy to read
 - document diagnostic/exploratory/micro-decision analysis

Notebooks and Reproducibility

Why do notebooks help **reproducibility**:

1. literate programming: interweaving code/commentary/output
 - allows rich commentary on code, output
 - develops a narrative that is easy to read
 - document diagnostic/exploratory/micro-decision analysis
2. keeps commentary/output close to code

Notebooks and Reproducibility

Why do notebooks help **reproducibility**:

1. literate programming: interweaving code/commentary/output
 - allows rich commentary on code, output
 - develops a narrative that is easy to read
 - document diagnostic/exploratory/micro-decision analysis
2. keeps commentary/output close to code
 - good tool for playing with code, immediately observing output

Notebooks and Reproducibility

Why do notebooks help **reproducibility**:

1. literate programming: interweaving code/commentary/output
 - allows rich commentary on code, output
 - develops a narrative that is easy to read
 - document diagnostic/exploratory/micro-decision analysis
2. keeps commentary/output close to code
 - good tool for playing with code, immediately observing output
3. good for showcasing results and interoperability

Notebooks and Reproducibility

Why do notebooks help **reproducibility**:

1. literate programming: interweaving code/commentary/output
 - allows rich commentary on code, output
 - develops a narrative that is easy to read
 - document diagnostic/exploratory/micro-decision analysis
2. keeps commentary/output close to code
 - good tool for playing with code, immediately observing output
3. good for showcasing results and interoperability
 - can be converted to many sharable formats (html, pdf, ...)

Notebooks and Reproducibility

Why do notebooks help **reproducibility**:

1. literate programming: interweaving code/commentary/output
 - allows rich commentary on code, output
 - develops a narrative that is easy to read
 - document diagnostic/exploratory/micro-decision analysis
2. keeps commentary/output close to code
 - good tool for playing with code, immediately observing output
3. good for showcasing results and interoperability
 - can be converted to many sharable formats (html, pdf, ...)
 - can convert **among** the notebook formats

Notebooks and Reproducibility

Why do notebooks help **reproducibility**:

1. literate programming: interweaving code/commentary/output
 - allows rich commentary on code, output
 - develops a narrative that is easy to read
 - document diagnostic/exploratory/micro-decision analysis
2. keeps commentary/output close to code
 - good tool for playing with code, immediately observing output
3. good for showcasing results and interoperability
 - can be converted to many sharable formats (html, pdf, ...)
 - can convert **among** the notebook formats
4. creates a reproducible record

Notebooks and Reproducibility

Why do notebooks help **reproducibility**:

1. literate programming: interweaving code/commentary/output
 - allows rich commentary on code, output
 - develops a narrative that is easy to read
 - document diagnostic/exploratory/micro-decision analysis
2. keeps commentary/output close to code
 - good tool for playing with code, immediately observing output
3. good for showcasing results and interoperability
 - can be converted to many sharable formats (html, pdf, ...)
 - can convert **among** the notebook formats
4. creates a reproducible record
 - code automatically generates results from data

Notebooks and Reproducibility

Why do notebooks help **reproducibility**:

1. literate programming: interweaving code/commentary/output
 - allows rich commentary on code, output
 - develops a narrative that is easy to read
 - document diagnostic/exploratory/micro-decision analysis
2. keeps commentary/output close to code
 - good tool for playing with code, immediately observing output
3. good for showcasing results and interoperability
 - can be converted to many sharable formats (html, pdf, ...)
 - can convert **among** the notebook formats
4. creates a reproducible record
 - code automatically generates results from data
 - this forces documentation on how the results were produced

Notebooks and Reproducibility

Why do notebooks help **reproducibility**:

1. literate programming: interweaving code/commentary/output
 - allows rich commentary on code, output
 - develops a narrative that is easy to read
 - document diagnostic/exploratory/micro-decision analysis
2. keeps commentary/output close to code
 - good tool for playing with code, immediately observing output
3. good for showcasing results and interoperability
 - can be converted to many sharable formats (html, pdf, ...)
 - can convert **among** the notebook formats
4. creates a reproducible record
 - code automatically generates results from data
 - this forces documentation on how the results were produced
5. promotes good code organization via chunking

Notebooks and Reproducibility

Why do notebooks help **reproducibility**:

1. literate programming: interweaving code/commentary/output
 - allows rich commentary on code, output
 - develops a narrative that is easy to read
 - document diagnostic/exploratory/micro-decision analysis
2. keeps commentary/output close to code
 - good tool for playing with code, immediately observing output
3. good for showcasing results and interoperability
 - can be converted to many sharable formats (html, pdf, ...)
 - can convert **among** the notebook formats
4. creates a reproducible record
 - code automatically generates results from data
 - this forces documentation on how the results were produced
5. promotes good code organization via chunking
6. software/formats not proprietary, easy to distribute

Some Potential Downsides

While code notebooks can be great, there are some **potential issues**, including:

1. chunks can be run in non-sequential order, making them not reproducible (soln: re-run all analysis at the end sequentially)

Some Potential Downsides

While code notebooks can be great, there are some **potential issues**, including:

1. chunks can be run in non-sequential order, making them not reproducible (soln: re-run all analysis at the end sequentially)
2. saved format of notebook may make version control difficult (soln: jupytext)

Some Potential Downsides

While code notebooks can be great, there are some **potential issues**, including:

1. chunks can be run in non-sequential order, making them not reproducible (soln: re-run all analysis at the end sequentially)
2. saved format of notebook may make version control difficult (soln: jupytext)
3. not great for non-interactive environments (soln: jupytext)

Some Potential Downsides

While code notebooks can be great, there are some **potential issues**, including:

1. chunks can be run in non-sequential order, making them not reproducible (soln: re-run all analysis at the end sequentially)
2. saved format of notebook may make version control difficult (soln: jupytext)
3. not great for non-interactive environments (soln: jupytext)
4. conversion among various formats isn't 100% fool-proof

Discussion

- Do you use notebooks regularly?
- Where do you find notebooks to be helpful?
- Where do you find notebook **not** to be helpful?

Example: Writing and converting notebooks.

1. Download and read in the titanic dataset:

```
url = 'https://tinyurl.com/35vnmbzm'  
titanic = read.csv(url)
```

1. Explore the data, and make a plot or two.

2.
 - If using jupyter, use jupytext to mirror the analysis to a .qmd and html
 - If using Rstudio, use quarto to mirror the analysis to a .ipynb and html

We'll reconvene and discuss thoughts.

Notebooks and Reproducibility

A notebook example:

We are going to generate some simulated data with the following model: First, we sample $N = 200$ samples

$$X_i \stackrel{iid}{\sim} U(-4\pi, 4\pi)$$

and then we calculate

$$Y_i = \sin(X_i) + \varepsilon_i$$

where $\varepsilon_i \stackrel{iid}{\sim} N(0, \sigma^2)$. Here $\sigma = .25$.

```
[1]: N = 200  
      sig = .25
```

```
[2]: x = sort(runif(N,-4*pi,4*pi))  
      eps = rnorm(N,0,sig)  
      y = sin(x)+eps
```

Notebooks and Reproducibility

A notebook example:

Now we're going to fit using a kernel regression from the `locpol` package

```
[3]: library('locpol')
```

a particularly important tuning parameter is the kernel bandwidth `bandw`, in our analysis we set it to 0.5

```
[4]: bandw = 0.5
```

```
[5]: smoothed = locPolSmootherC(x=x, y=y, xeval=x,
                                 deg=0, kernel=gaussK,
                                 bw=bandw)
preds = smoothed$beta0
```

Notebooks and Reproducibility

A notebook example:

```
[6]: plot(x,y)
      lines(x,preds,col='red',lwd=2)
```

