

CODE NOTEBOOKS

EXERCISE 0: INSTALL JUPYTER

Install jupyter. First, you need an installation of python/pip:

- <https://www.python.org/downloads/>

Then, install jupyter following <https://jupyter.org/install>:

```
pip install jupyterlab  
jupyter lab
```

To use R, run these commands in an R session

```
install.packages("devtools")  
devtools::install_github("IRkernel/IRkernel")  
IRkernel::installspec()
```

Alternatively, you can use the interactive web-apps

- <https://jupyter.org/try>
- <https://colab.research.google.com/>

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

Primarily, we will give a brief overview of **jupyter** and discuss its advantages for reproducibility.

JUPYTER LAB EXAMPLE

An example of **jupyter lab**:

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

- rich text**: A red box highlights the title "First, let's load the penguins dataset".
- code**: A red box highlights the code cell containing `library("palmerpenguins")` and `library('tidyverse')`.
- suppressed output**: A red box highlights the ellipsis `...` preceding the data output.
- output**: A red box highlights the resulting data frame output.

```
[1]: library("palmerpenguins")
library('tidyverse')
...
[2]: penguins %>% sample_n(3)
```

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

At the bottom, the status bar shows "Simple" mode, 0 cells run, R | Idle, Mode: Command, and Ln 1, Col 35. The file name is example.ipynb.

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:** Shows the title "Intro_to_jupyter.ipynb (2) - Jupyter" and the URL "localhost:8889/lab/tree/intro_to_jupyter.ipynb".
- Toolbar:** Includes File, Edit, View, Run, Kernel, Tabs, Settings, Help, and a Git icon.
- Code Cell:** Displays the following code snippet:

```
# heading level 1
## heading level 2
### heading level 3
```
- Output Cells:** Three cells show the resulting headings:
 - heading level 1
 - heading level 2
 - heading level 3
- Status Bar:** Shows "Simple" mode, cell index "0", cell type "Text", "R | Idle" status, "Mode: Command", "Ln 1, Col 1", and the file name "intro_to_jupyter.ipynb".

BASIC MARKDOWN

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

```
**bold text** displays bold text
*italic text* displays italic text
> block quote
    block quote
```

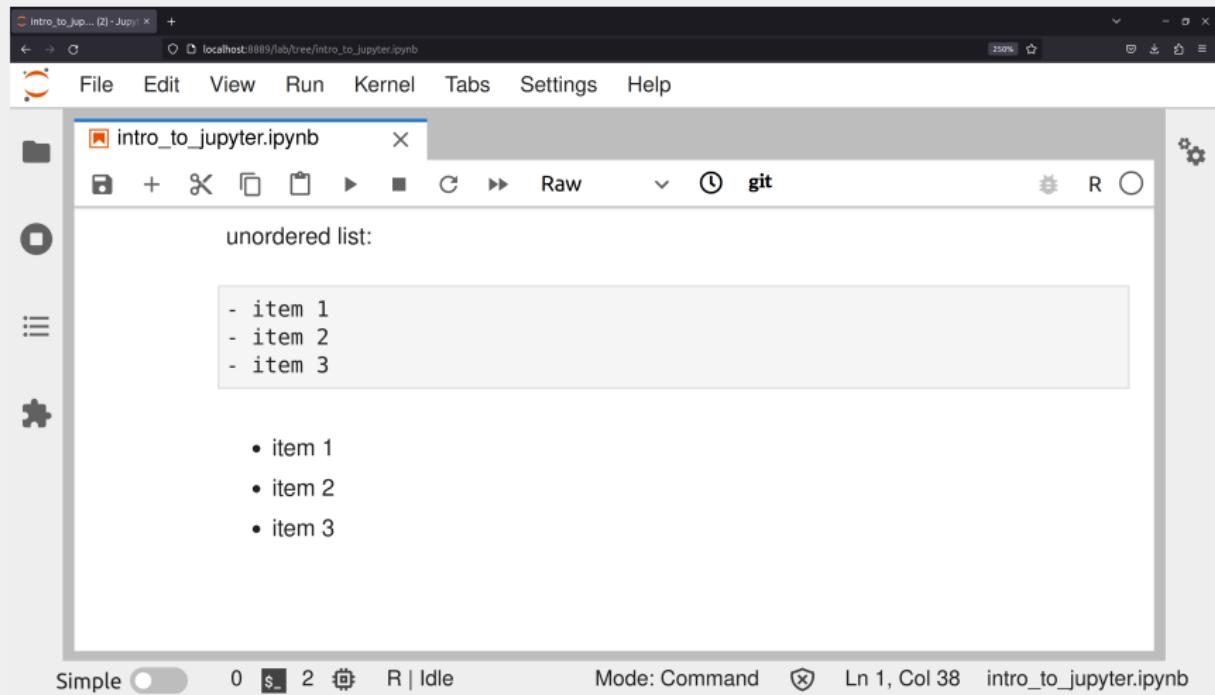
The notebook is titled "intro_to_jupyter.ipynb". The bottom status bar indicates "Mode: Command" and "Ln 1, Col 38".

BASIC MARKDOWN

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

- Title Bar:** Shows the title "intro_to_jupyter.ipynb" and a tab count of "(2 - Jupyter)".
- Header Bar:** Includes File, Edit, View, Run, Kernel, Tabs, Settings, and Help menus.
- Toolbar:** Includes icons for file operations like Open, Save, and New, along with a git icon.
- Code Cell:** Contains the text "ordered lists:" followed by two ordered lists:
 - 1. item 1
 - 1. item 2
 - 1. item 3
- Text Cell:** Contains the following ordered list:
 - 1. item 1
 - 2. item 2
 - 3. item 3
- Bottom Status Bar:** Shows "Simple" mode, cell count (0), language (S), kernel count (2), and status (R | Idle). It also displays "Mode: Command" and "Ln 1, Col 38" along with the file name "intro_to_jupyter.ipynb".

BASIC MARKDOWN



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

code

displays code

you can link to webpages like this:

```
[link title](http://www.example.com/)
```

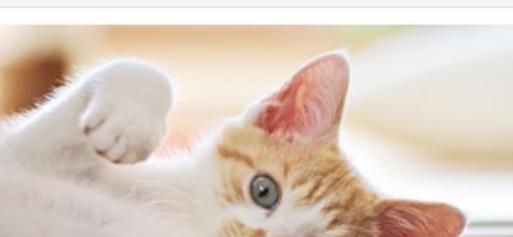
link title

Simple 0 \$ 2 R | Idle Mode: Command Ln 1, Col 1 intro_to_jupyter.ipynb

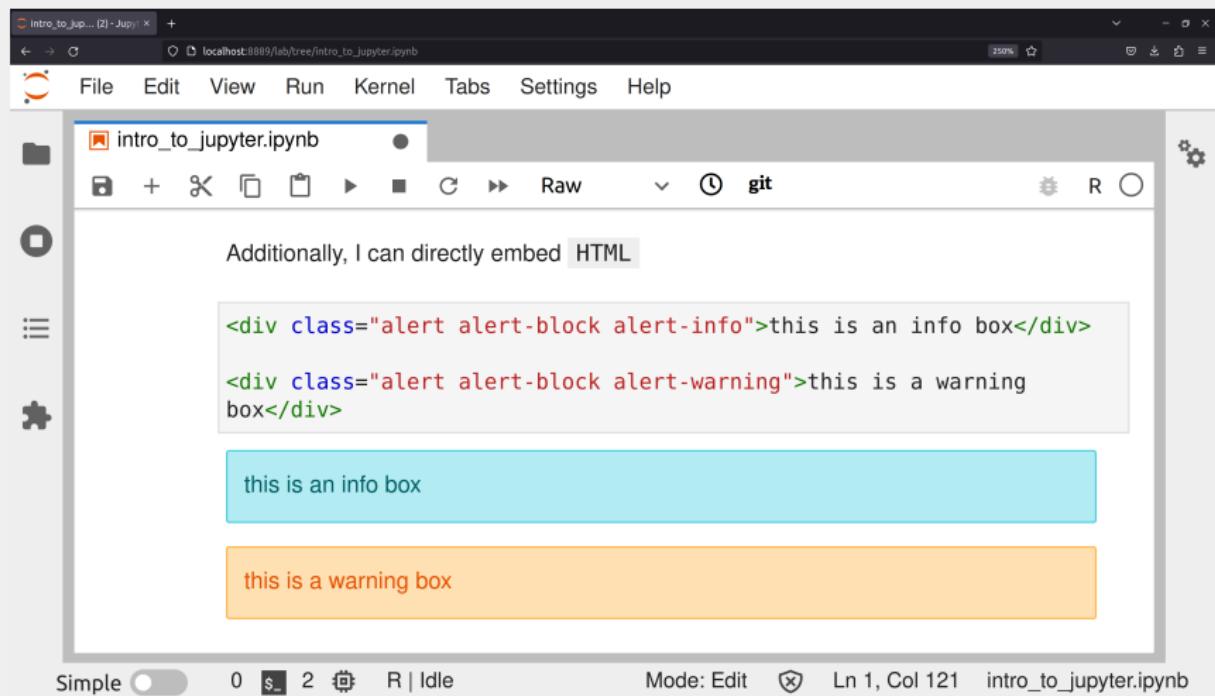
BASIC MARKDOWN

and embed images like this:

```
![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)
```



BASIC MARKDOWN



BASIC MARKDOWN

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

- Title Bar:** Shows the title "intro_to_jupyter.ipynb" and the URL "localhost:8888/lab/tree/intro_to_jupyter.ipynb".
- Toolbar:** Includes File, Edit, View, Run, Kernel, Tabs, Settings, Help, and a Git icon.
- Header Bar:** Shows the notebook name "intro_to_jupyter.ipynb" and a progress bar.
- Content Area:** Displays the text "I can also embed *LATEX* mathematics type" followed by a code block containing the LaTeX code for the quadratic formula:

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

The rendered output below the code block is the quadratic formula:
$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$
- Bottom Status Bar:** Shows "Simple" mode, a toggle switch, cell count (0), a file icon, "R | Idle", "Mode: Command", a search icon, line and column information ("Ln 1, Col 1"), and the file path "intro_to_jupyter.ipynb".

EXERCISE 1: ADD MARKDOWN TO A JUPYTER NOTEBOOK

Load up jupyter and start a new notebook .ipynb file.

In several different cells add markdown and render it. Your markdown should include:

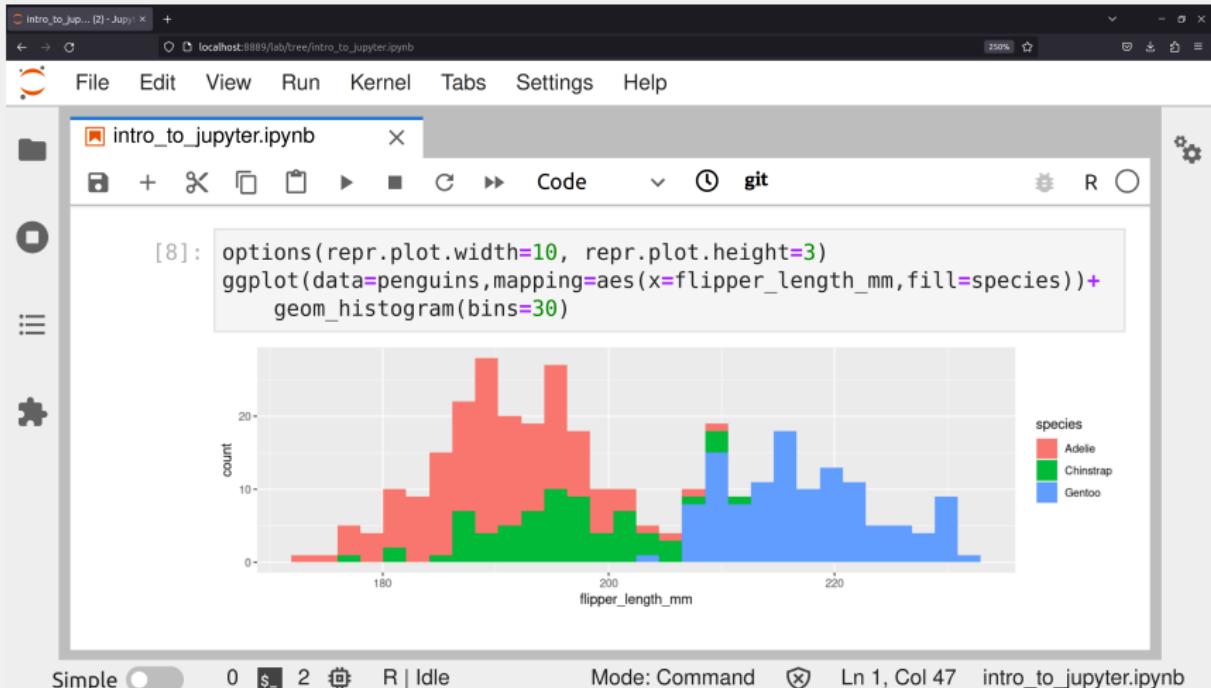
- a heading
- bold and italic text
- a numbered and un-numbered list of items
- displayed pseudocode
- a link to an external webpage
- an image
- embedded math in LaTeX

After writing these blocks of markdown, re-arrange them

- copy and past one
- delete one
- move one

CODE: BASIC OUTPUT

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



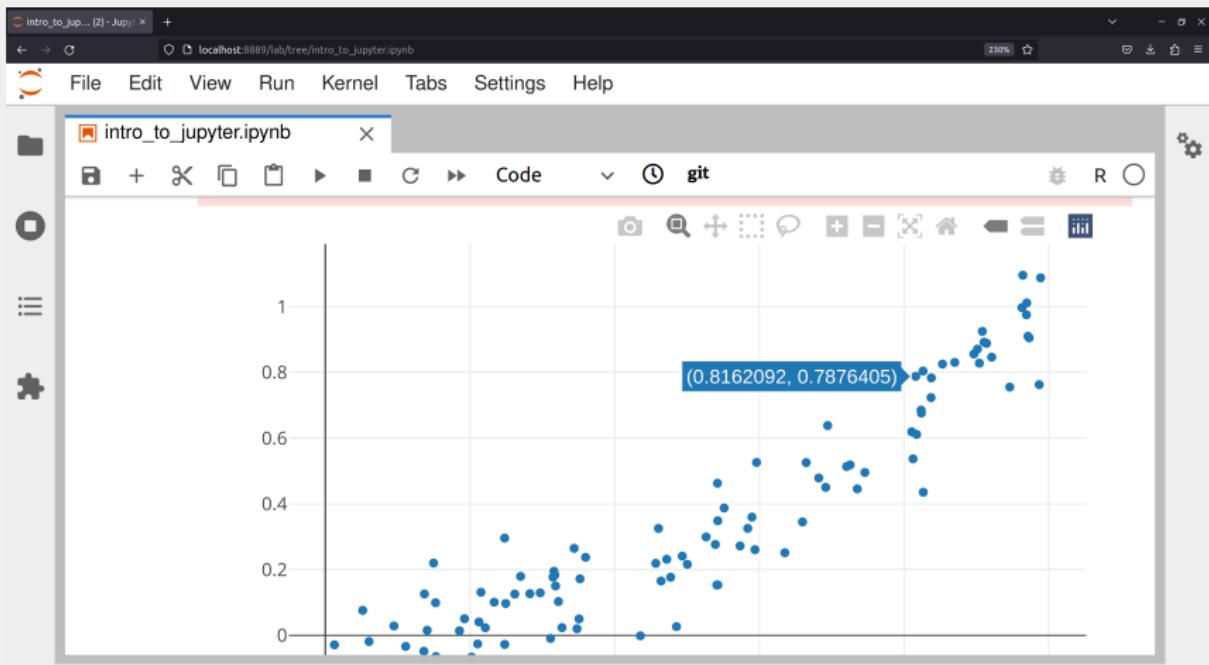
The screenshot shows a Jupyter Notebook interface with the following details:

- Header:** "intro_to_jupyter.ipynb" (2) - Jupyter
- Toolbar:** File, Edit, View, Run, Kernel, Tabs, Settings, Help
- Code Cell [8]:**

```
options(repr.plot.width=10, repr.plot.height=3)
ggplot(data=penguins,mapping=aes(x=flipper_length_mm,fill=species))+
    geom_histogram(bins=30)
```
- Output:** A histogram plot showing the distribution of flipper length (mm) for three penguin species: Adelie (red), Chinstrap (green), and Gentoo (blue). The x-axis ranges from approximately 170 to 230 mm, and the y-axis shows the count of observations (0 to 20+).
- Legend:** species
Adelie (red)
Chinstrap (green)
Gentoo (blue)
- Bottom Status Bar:** Simple, 0 \$ 2, R | Idle, Mode: Command, Ln 1, Col 47, intro_to_jupyter.ipynb

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,
- ...

EXERCISE 2: ADD SOME CODE

Add some code to your notebook file. Here, we'll use the palmer penguins data set. You can download it and load it up with

```
install.packages("palmerpenguins")
library('palmerpenguins')
head('palmerpenguins')
```

Explore this data:

- make a histogram of flipper length for each species
- add some markdown commentary to your plots
- install plotly via `install.packages('plotly')` and then add an interactive scatter plot:

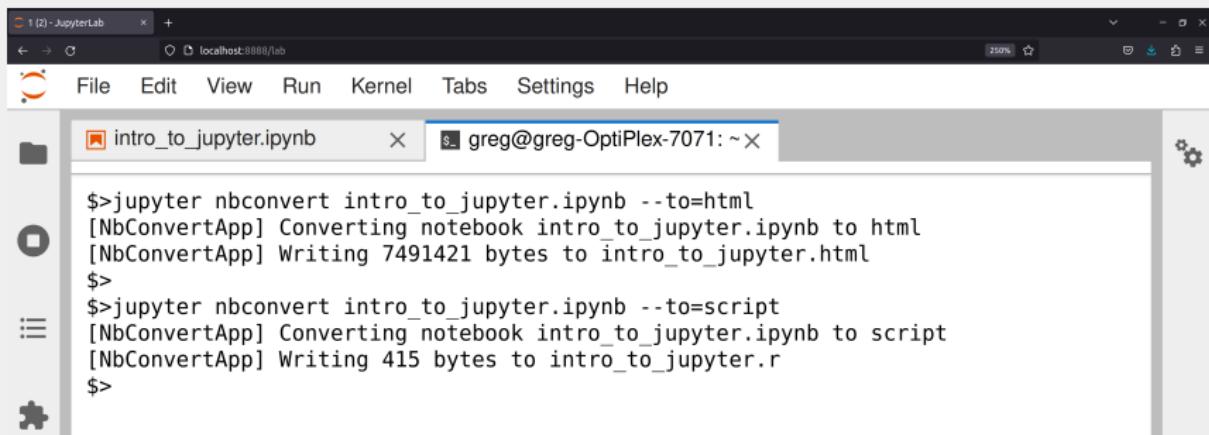
```
p <- plot_ly(x = ____,
               y = ____,
               mode = "markers", type = "scatter")
embed_notebook(p)
```

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
```

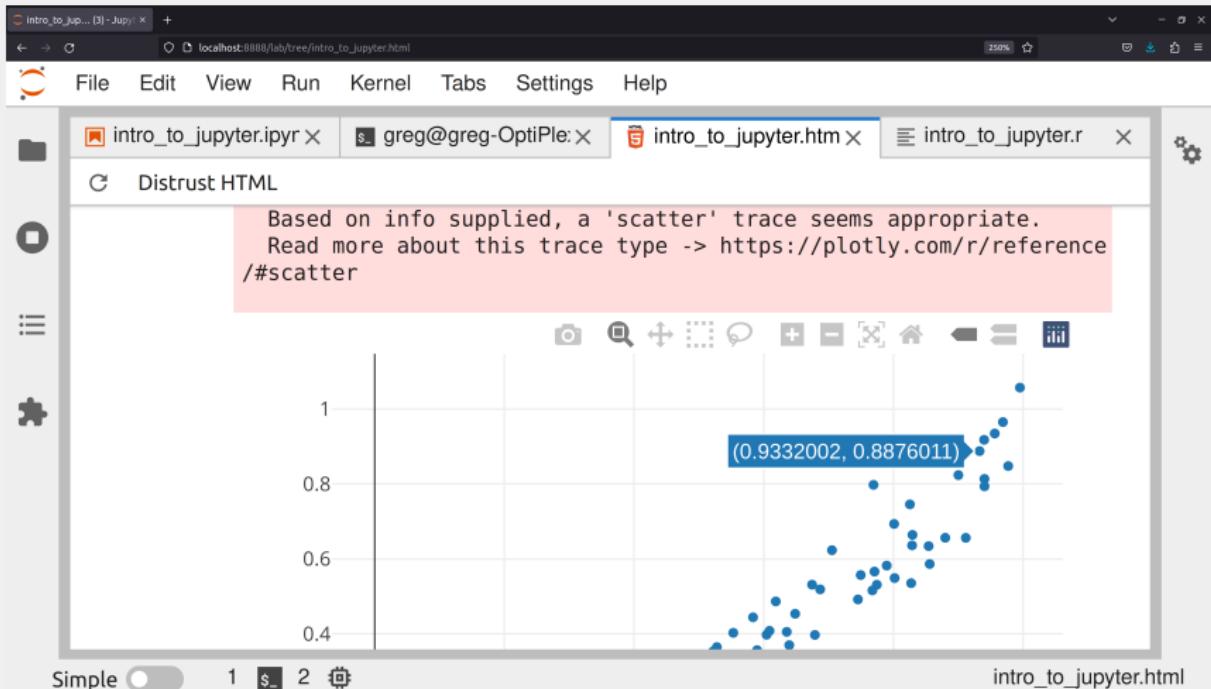


The screenshot shows the JupyterLab interface with a terminal window open. The terminal window title is "intro_to_jupyter.ipynb". The terminal output shows two commands being run:

```
$>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
$>
```

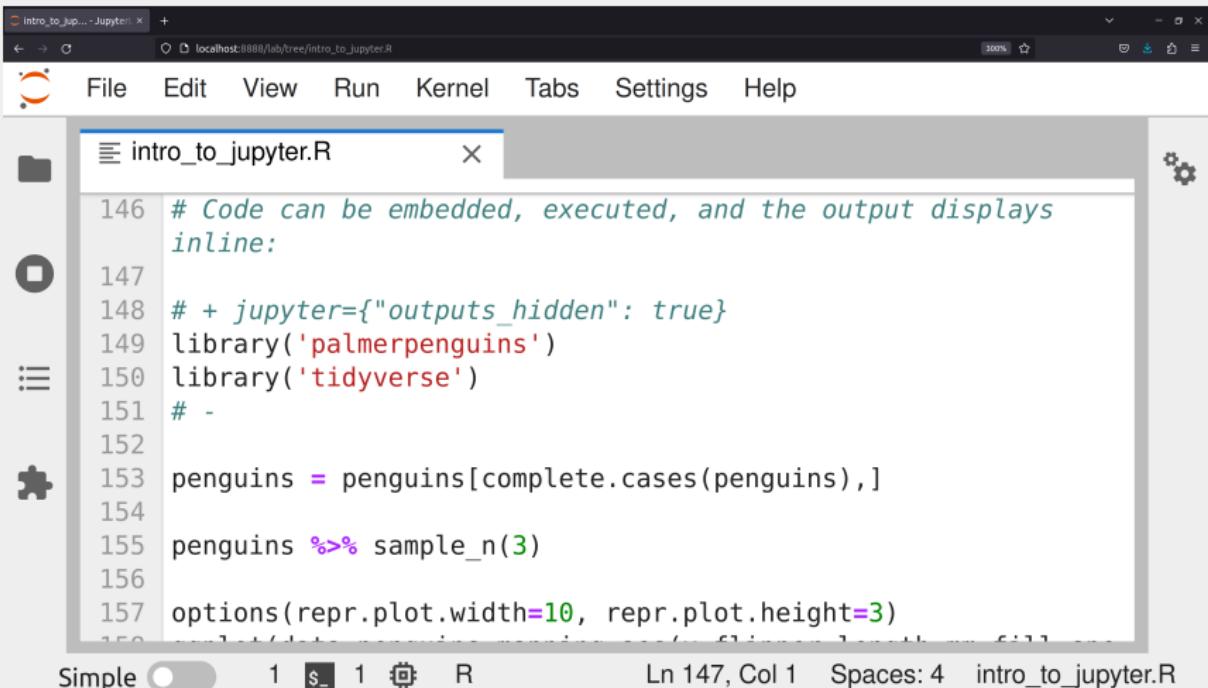
EXPORTING: HTML

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 - Jupyter
- URL:** localhost:8888/lab/tree/intro_to_jupyter.R
- Menu Bar:** File, Edit, View, Run, Kernel, Tabs, Settings, Help
- Code Cell:** intro_to_jupyter.R
- Code Content:**

```
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
153 penguins = penguins[complete.cases(penguins),]
154
155 penguins %>% sample_n(3)
156
157 options(repr.plot.width=10, repr.plot.height=3)
```
- Toolbar Icons:** (File, Cell, Kernel, Help) and a gear icon.
- Status Bar:** Simple, 1, \$, 1, R, Ln 147, Col 1, Spaces: 4, intro_to_jupyter.R

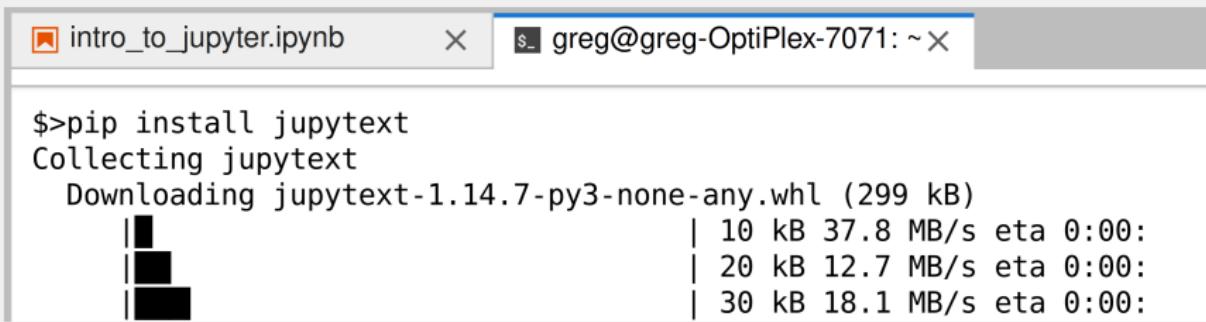
NOTEBOOK INTEROPERABILITY

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. The title bar says "intro_to_jupyter.ipynb". The command \$>pip install jupytext is entered. The output shows the package being collected and downloaded, with a progress bar and estimated download times for each segment.

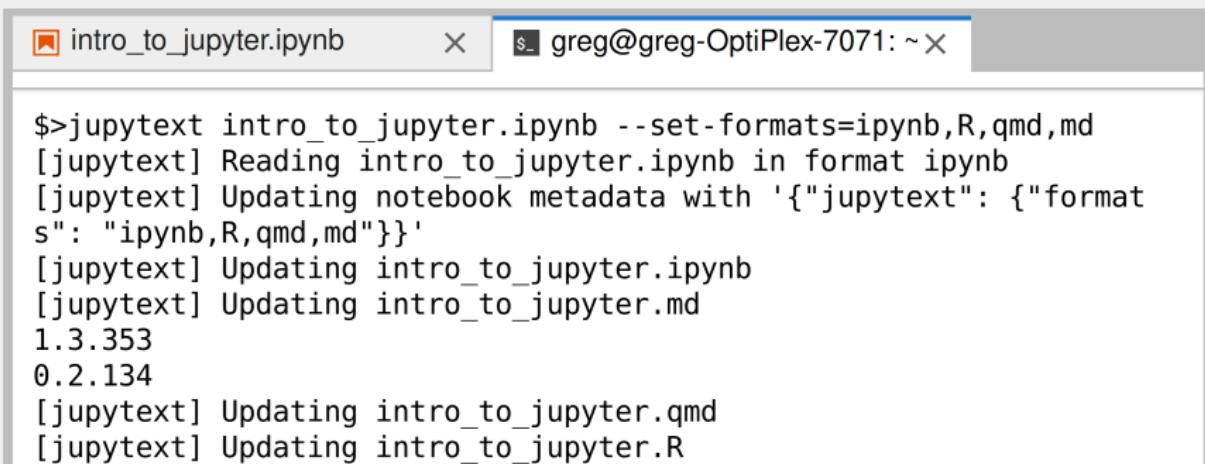
```
$>pip install jupytext
Collecting jupytext
  Downloading jupytext-1.14.7-py3-none-any.whl (299 kB)
    | 10 kB 37.8 MB/s eta 0:00:
    | 20 kB 12.7 MB/s eta 0:00:
    | 30 kB 18.1 MB/s eta 0:00:
```

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 file icon, the path "intro_to_jupyter.ipynb", a close button ("X"), and a user prompt "\$ greg@greg-OptiPlex-7071: ~". Below the header, the terminal output is displayed in a white area with black text. The command "\$>jupytext intro_to_jupyter.ipynb --set-formats=ipynb,R,qmd,md" is entered, followed by several lines of "[jupytext]" log messages indicating the conversion of the notebook to various formats.

```
$>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
```

JUPYTEXT 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 Notebook interface with two tabs open: 'intro_to_jupyter.ipynb' and 'intro_to_jupyter.R'. The left pane displays the Python code and a warning message about a scatter plot trace. The right pane displays the R code, which includes a comment indicating a change.

intro_to_jupyter.ipynb:

```
ied, a 'scatter' trac  
e seems appropriate.  
Read more about thi  
s trace type -> http://plotly.com/r/referenc...#scatter
```

This is a change

intro_to_jupyter.R:

```
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
```

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

EXERCISE 3: EXPORTING

Export your notebook to:

- html, then inspect the interactive html
- a .R script, then try running the script separately

Install jupytext via

```
pip install jupytext
```

Mirror your notebook into

- an .R script,
- a .md markdown file

Open up the .R script and edit it. Go back and re-load the .ipynb jupyter notebook and observe the changes.

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 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

EXERCISE 4: PUTTING IT ALL TOGETHER

1. Load up the plates.csv data from Bray et al.
2. Using jupyter, conduct some exploratory analysis.
 - A good example of an exploratory analysis is to conduct PCA on the data and visualize the first several principal components, coloring the data using the metadata. Some data-cleaning might be in order.
 - Make sure to document your code and use the markdown text to write comments on the analysis.
1. Export your analysis as a HTML document using jupyter.
2. Using jupytext, mirror the analysis to a .R script.
 - Run the script independently after exporting to it.

DISCUSSION

- Do you use notebooks regularly? Might you, now?
- Where do you find notebooks to be helpful?
- Where do you find notebooks **not** to be helpful?