

NOTEBOOKS

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

JUPYTER LAB EXAMPLE

An example of jupyter lab:

The screenshot shows the Jupyter Lab interface with several UI elements annotated with red boxes and arrows:

- A red box labeled "rich text" surrounds the toolbar icons.
- A red arrow points from the "rich text" box to the toolbar icon for rich text.
- A red box labeled "code" surrounds the code input area.
- A red arrow points from the "code" box to the code input area.
- A red box labeled "suppressed output" surrounds the ellipsis (...).
- A red arrow points from the "suppressed output" box to the ellipsis (...).
- A red box labeled "output" surrounds the data table.
- A red arrow points from the "output" box to the data table.

The Jupyter Lab interface includes a sidebar with file, folder, and settings icons, a toolbar with file operations and git integration, and a main area for code and data visualization. The code cell [1] contains:

```
library("palmerpenguins")
library('tidyverse')
...
```

The code cell [2] contains:

```
penguins %>% sample_n(3)
```

The resulting output is a tibble:

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, a terminal prompt (0 \$_), and the kernel state (R | Idle). The bottom right shows the mode is "Command", the line and column are "Ln 1, Col 35", and the file 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**

(In fact, this presentation is written in markdown)

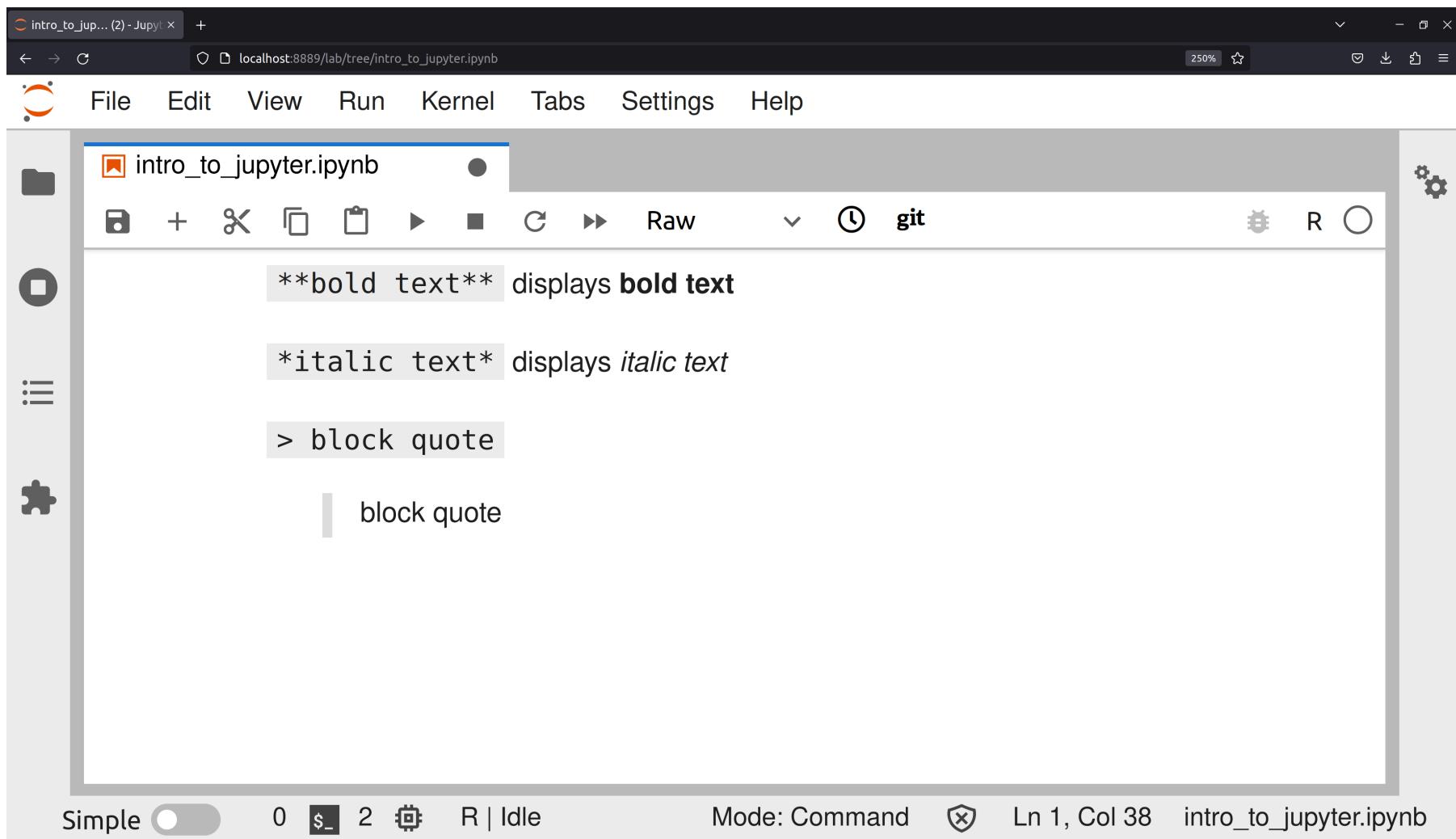
BASIC MARKDOWN

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

- Title Bar:** "intro_to_jupyter.ipynb" (2 tabs open)
- Address Bar:** "localhost:8889/lab/tree/intro_to_jupyter.ipynb"
- Toolbar:** File, Edit, View, Run, Kernel, Tabs, Settings, Help
- File List:** Shows a folder icon and the file "intro_to_jupyter.ipynb".
- Tool Buttons:** Includes icons for file operations like Open, Save, New, Cut, Copy, Paste, and Run.
- Code Cell:** Contains the following text:

```
# heading level 1
## heading level 2
### heading level 3
```
- Output Cells:** Three large text blocks representing the output of the code:
 - "heading level 1"
 - "heading level 2"
 - "heading level 3"
- Bottom Status Bar:** "Simple" mode switch, cell count (0), R | Idle, Mode: Command, Line 1, Col 1, and the file name "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 tabs open)
- Toolbar:** File, Edit, View, Run, Kernel, Tabs, Settings, Help
- File List:** A sidebar on the left contains icons for file operations: folder, new file, cut, copy, paste, delete, and refresh.
- Content Area:** The main area displays the following Markdown content:

```
ordered lists:  
  
1. item 1  
1. item 2  
1. item 3  
  
1. item 1  
2. item 2  
3. item 3
```
- Bottom Status Bar:** Shows "Simple" mode, cell count (0), kernel status (R | Idle), mode (Command), line/col (Ln 1, Col 38), and the notebook name ("intro_to_jupyter.ipynb").

BASIC MARKDOWN

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

- Title Bar:** "intro_to_jupyter.ipynb" (2 tabs)
- Address Bar:** "localhost:8889/lab/tree/intro_to_jupyter.ipynb"
- Toolbar:** File, Edit, View, Run, Kernel, Tabs, Settings, Help
- File List:** Shows "intro_to_jupyter.ipynb" and other icons for files and notebooks.
- Toolbar Buttons:** Includes file operations (+, -, X), cell types (Code, Markdown, Rich, Raw), and git integration.
- Text Area:** Displays the following Markdown content:

```
unordered list:
  - item 1
  - item 2
  - item 3
  - item 1
  - item 2
  - item 3
```
- Bottom Status Bar:** "Simple" mode switch, cell count (0), R | Idle, Mode: Command, Line 1, Col 38, and the notebook name "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

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

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

- Title Bar:** "intro_to_jupyter.ipynb" (2 tabs)
- Toolbar:** File, Edit, View, Run, Kernel, Tabs, Settings, Help
- File List:** Shows "intro_to_jupyter.ipynb" selected.
- Toolbar Buttons:** File, New, Cut, Copy, Paste, Run, Stop, Cell, Cell Kernel, Cell Kernel, Raw, Git, Cell Kernel, Cell Kernel, Cell Kernel.
- Text Area:** "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 (button), 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:** The title bar displays "intro_to_jupyter.ipynb (2) - Jupyter" and the URL "localhost:8889/lab/tree/intro_to_jupyter.ipynb".
- Toolbar:** The toolbar includes File, Edit, View, Run, Kernel, Tabs, Settings, Help, and a Git icon.
- File List:** A sidebar on the left lists files: "intro_to_jupyter.ipynb" (selected), a folder icon, and a file icon.
- Code Cell:** The main area contains the following code cell:

```
<div class="alert alert-block alert-info">this is an info box</div>

<div class="alert alert-block alert-warning">this is a warning
box</div>
```
- Output Cells:** Two output cells are shown:
 - A light blue box containing the text "this is an info box".
 - An orange box containing the text "this is a warning box".
- Bottom Status Bar:** The status bar shows "Simple" mode, 0 cells run, 2 cells total, R | Idle, Mode: Edit, a shield icon, Ln 1, Col 121, and the file name "intro_to_jupyter.ipynb".

BASIC MARKDOWN

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

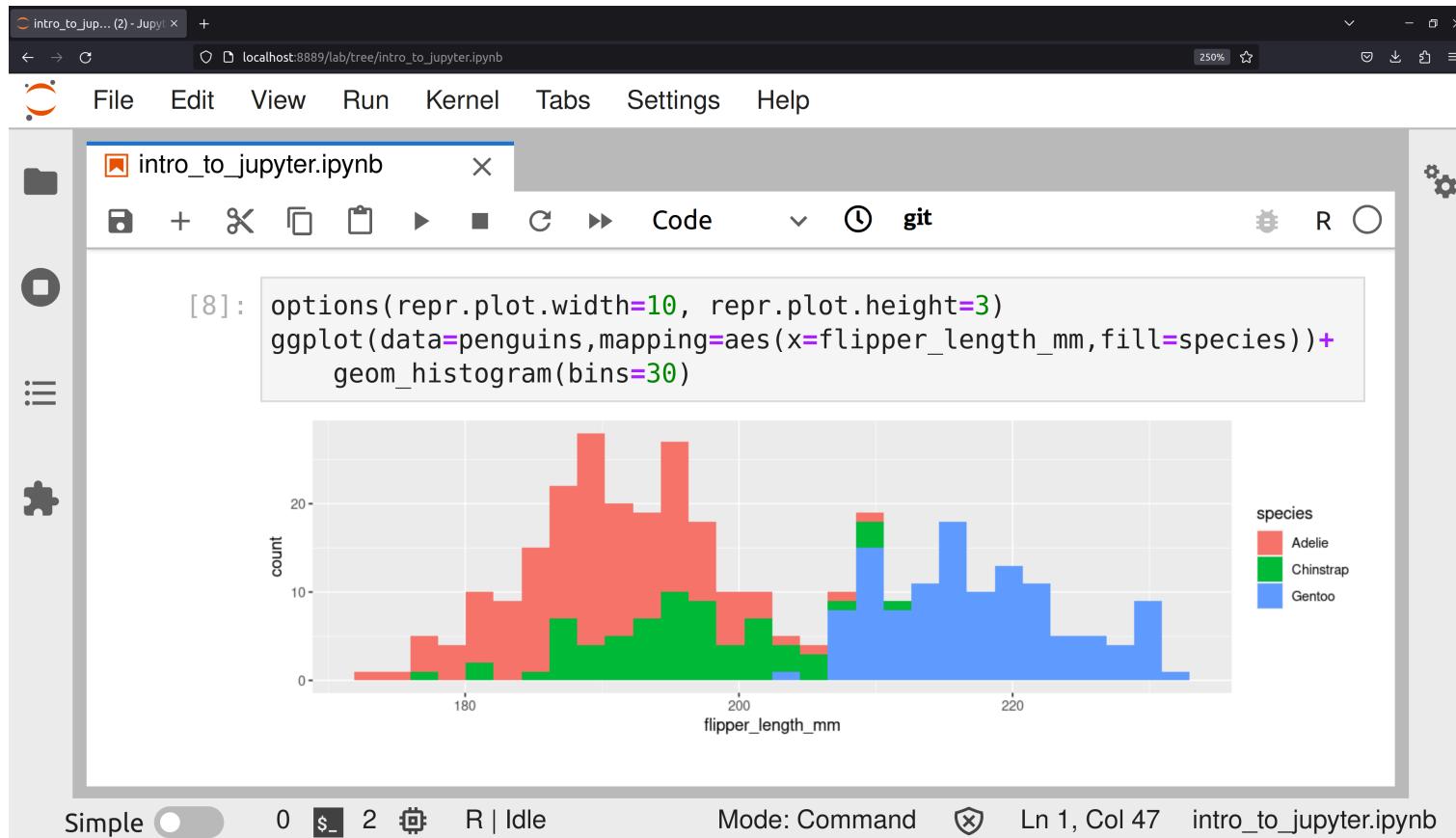
- Title Bar:** The title bar displays "intro_to_jupyter.ipynb (2) - Jupyter" and the URL "localhost:8889/lab/tree/intro_to_jupyter.ipynb".
- Toolbar:** The toolbar includes File, Edit, View, Run, Kernel, Tabs, Settings, Help, and a search bar.
- Header Bar:** The header bar shows the current file "intro_to_jupyter.ipynb", a dark mode indicator, and icons for file operations like save, new, cut, copy, paste, and run.
- Content Area:** The main content area contains the text "I can also embed *LATEX* 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:** A vertical sidebar on the left features icons for file operations (New, Open, Save, Delete), a search bar, and a gear icon for settings.
- Bottom Bar:** The bottom bar shows the mode switch ("Simple"), cell count (0), and R | Idle status, along with the current mode ("Command"), line and column information ("Ln 1, Col 1"), and the file name ("intro_to_jupyter.ipynb").

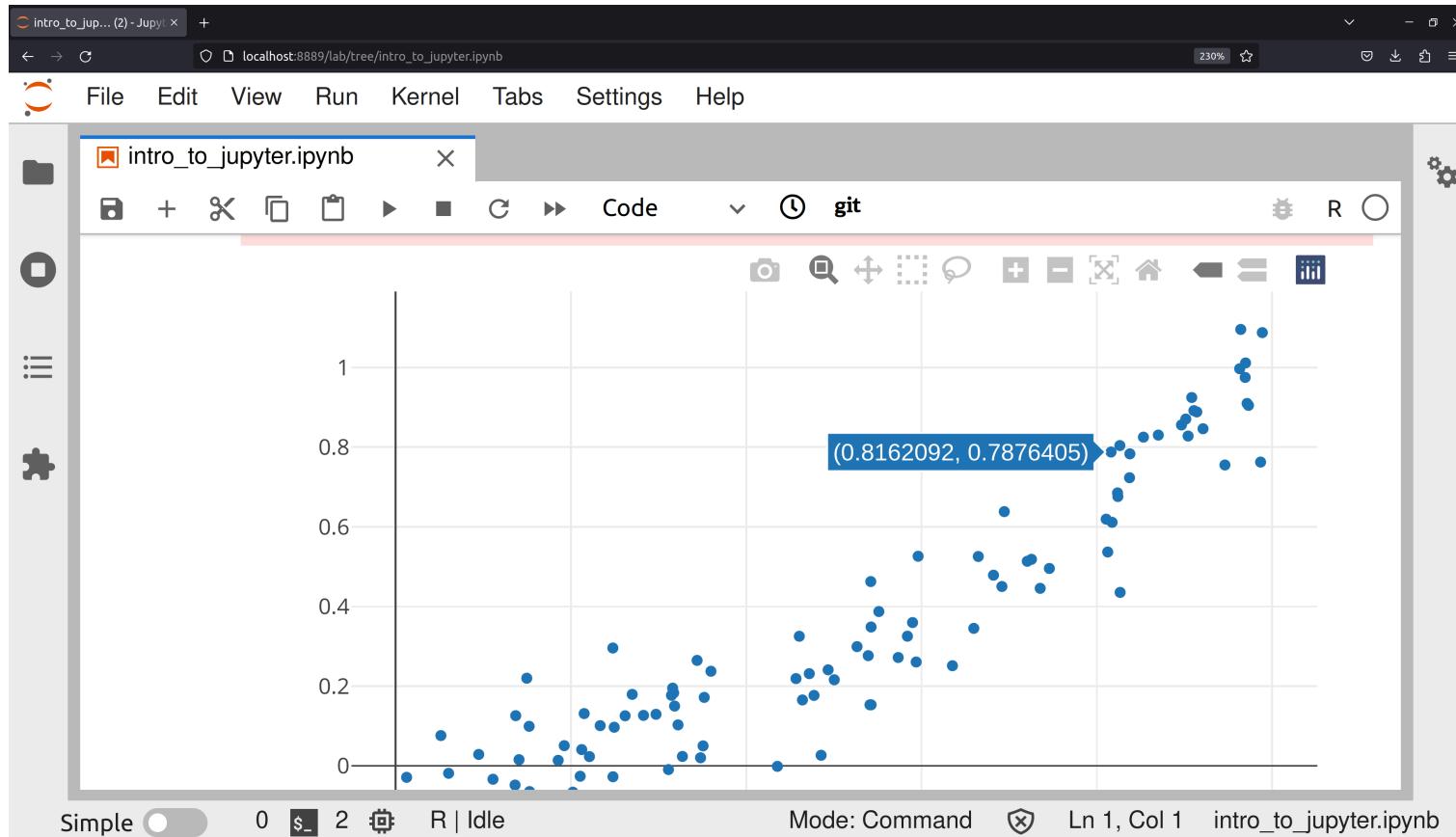
CODE: BASIC OUTPUT

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:

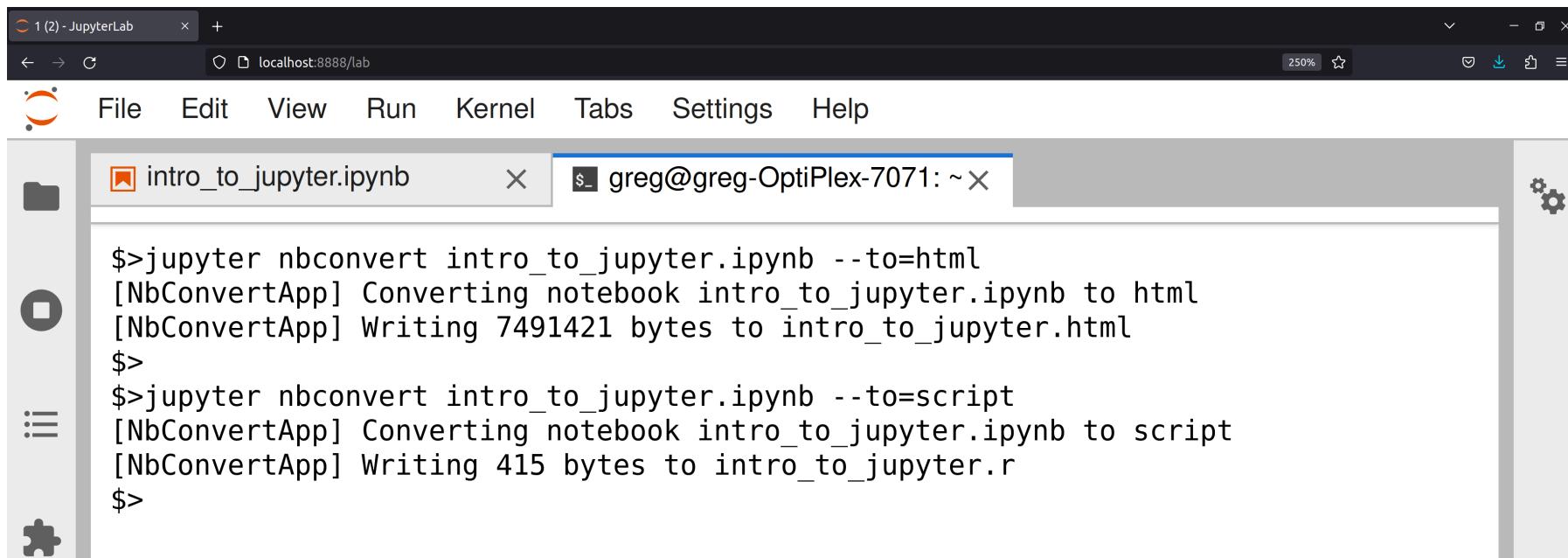
- python,
- r,
- julia,
- stata,
- sql,
- octave,
- matlab,
- java,
- go,
- C,
- ...

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
```

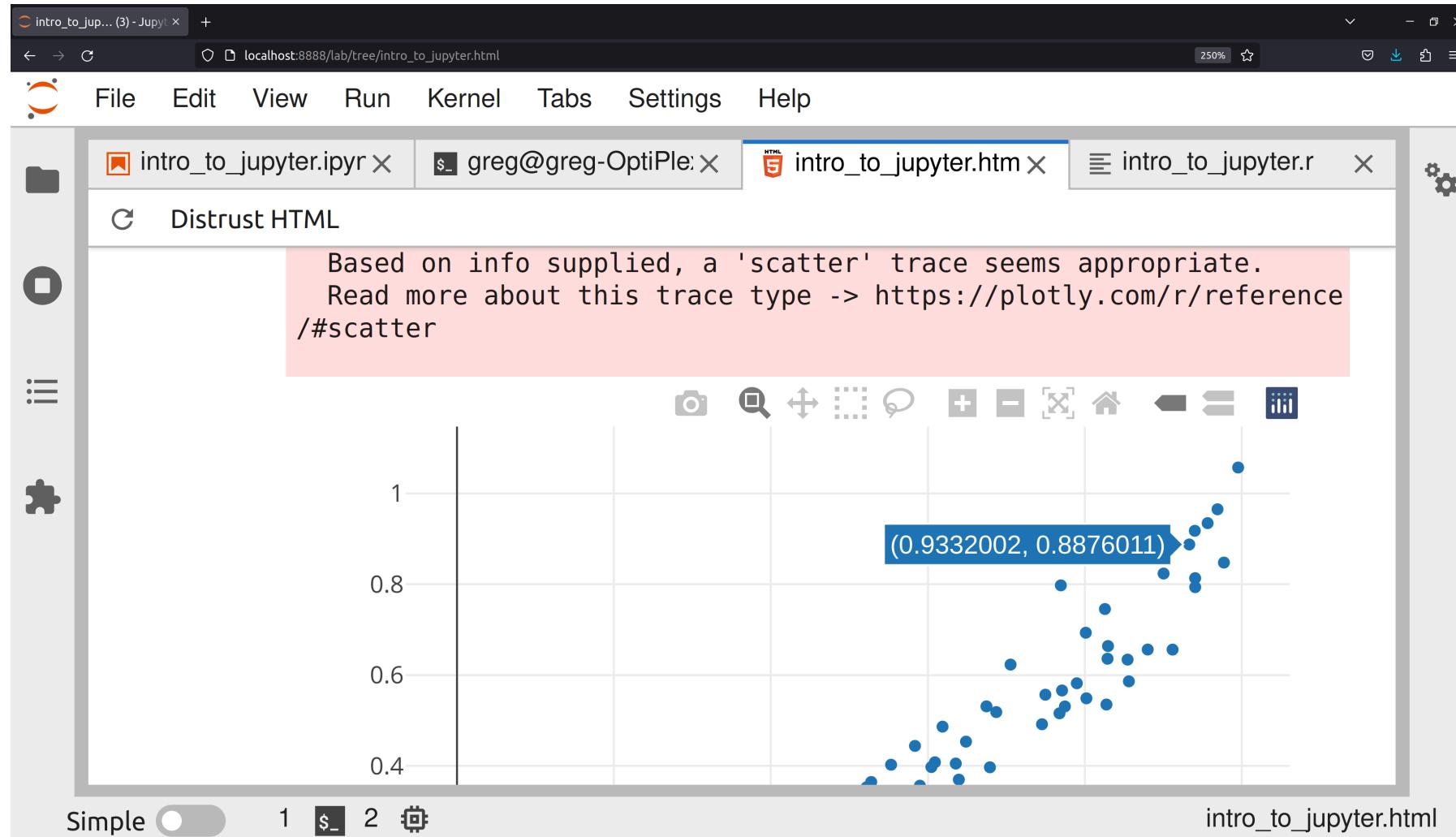


The screenshot shows the JupyterLab interface with a terminal tab open. The terminal window displays the following command and its execution:

```
$>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  
$>
```

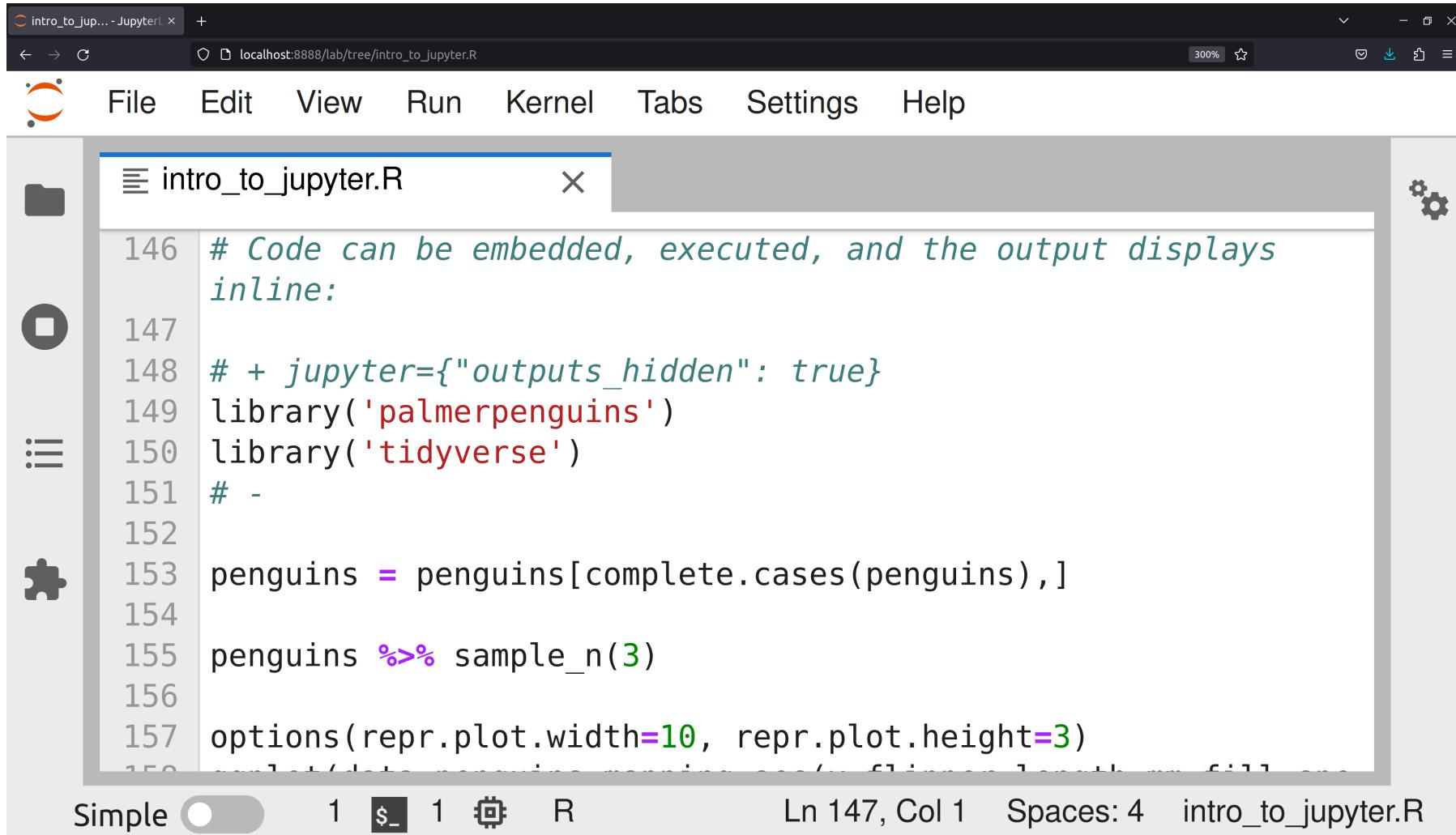
EXPORTING: HTML

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 a single R script file open. The file is titled "intro_to_jupyter.R". The code in the file is as follows:

```
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)
158
#<code>#</code>
#<code>#</code>
```

The interface includes a toolbar at the top with File, Edit, View, Run, Kernel, Tabs, Settings, and Help. On the left, there's a sidebar with icons for file operations like New, Open, Save, and a gear icon for settings. The bottom of the window shows status information: Simple (button), 1 \$ 1 R, Ln 147, Col 1, Spaces: 4, and the file name intro_to_jupyter.R.

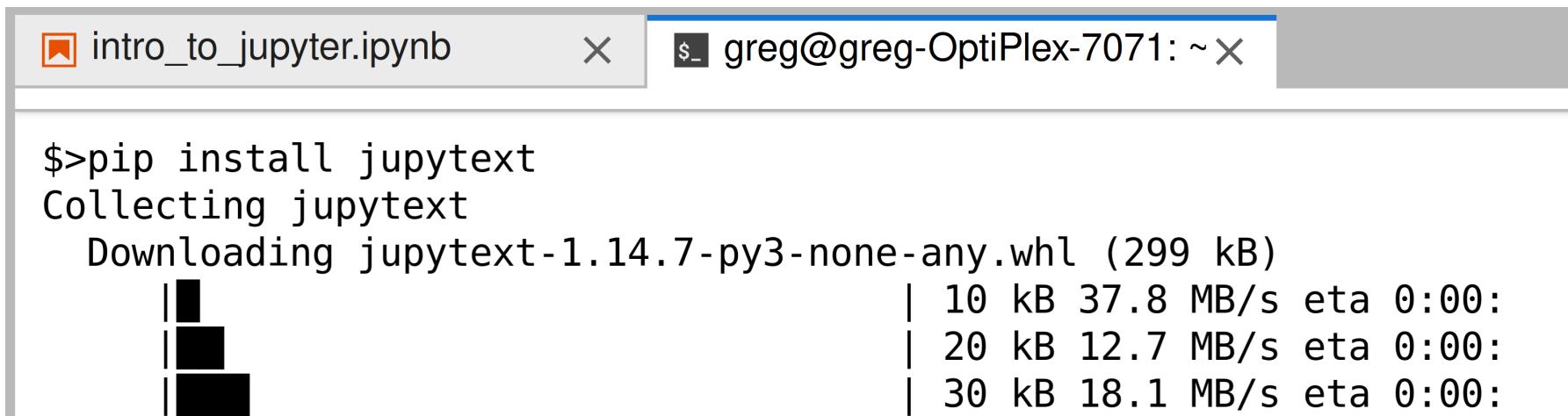
NOTEBOOK INTEROPERABILITY

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 window shows the command \$>pip install jupytext being run, followed by the output of the package download. The output includes a progress bar for the download of "jupytext-1.14.7-py3-none-any.whl" (299 kB), with speeds of 37.8 MB/s, 12.7 MB/s, and 18.1 MB/s indicated for different segments of the file.

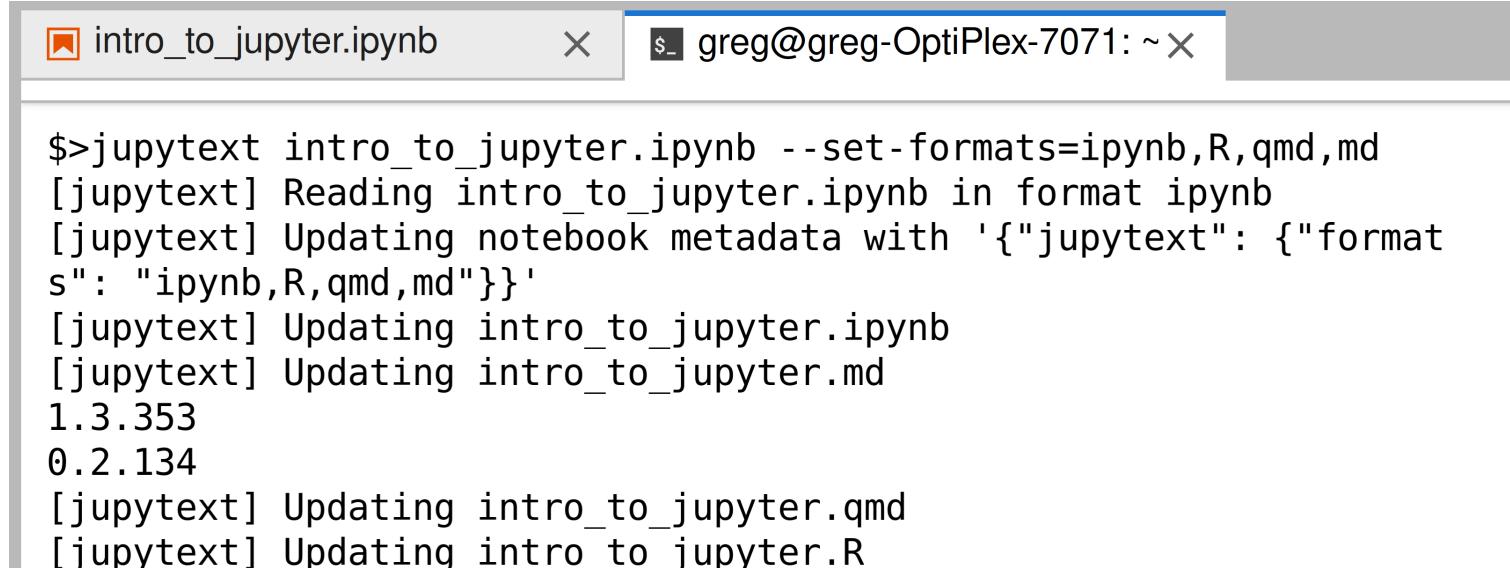
```
$>pip install jupytext
Collecting jupytext
  Downloading jupytext-1.14.7-py3-none-any.whl (299 kB)
    |██████████| 10 kB 37.8 MB/s eta 0:00:
    |██████████| 20 kB 12.7 MB/s eta 0:00:
    |██████████| 30 kB 18.1 MB/s eta 0:00:
```

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,...
```



A screenshot of a terminal window titled "intro_to_jupyter.ipynb". The window shows the command \$>jupytext intro_to_jupyter.ipynb --set-formats=ipynb,R,qmd,md being run. The terminal output indicates that jupytext is reading the notebook, updating its metadata to include a "jupytext" key with a "formats" value of "ipynb,R,qmd,md", and then updating the notebook file itself along with its md, qmd, and R counterparts.

```
$>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 the JupyterLab interface with two tabs open:

- `intro_to_jupyter.ipynb`: An IPython notebook file. It contains a red-highlighted text block with the following content:

```
ied, a 'scatter' trac
e seems appropriate.
Read more about thi
s trace type -> http
s://plotly.com/r/refer
ence/#scatter
```
- `intro_to_jupyter.R`: An R script file. It contains the following 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
```

The status bar at the bottom indicates "Mode: Com...".

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).
- 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)
- can edit `quarto` documents in jupyter

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
 - 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/format 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)
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? Might you, now?
- Where do you find notebooks to be helpful?
- Where do you find notebooks **not** to be helpful?