

# 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

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 "rich text" button in the toolbar.
- code**: A red box highlights the code input area.
- suppressed output**: A red box highlights the ellipsis (...).
- output**: A red box highlights the output table.

The code input area contains:

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

The output section displays the following table:

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 0 \$ 1 R | Idle, Mode: Command, and Ln 1, Col 35 example.ipynb.

# 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/>

## 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:** The title bar displays "intro\_to\_jupyter.ipynb" and the URL "localhost:8889/lab/tree/intro\_to\_jupyter.ipynb".
- Toolbar:** The toolbar includes icons for File, Edit, View, Run, Kernel, Tabs, Settings, Help, and a gear icon.
- Code Cell:** A code cell contains the following Python code:

```
# heading level 1
## heading level 2
### heading level 3
```
- Output:** Below the code cell, three headings are displayed:
  - heading level 1**
  - heading level 2**
  - heading level 3**
- Status Bar:** The status bar at the bottom shows "Simple" mode, 0 cells run, 2 cells total, R | Idle kernel, Mode: Command, and Ln 1, Col 1.
- Git Status:** A "git" icon is visible in the toolbar and the status bar.

# Basic Markdown

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

- Title Bar:** intro\_to\_jupyter.ipynb (2) - Jupyter
- Toolbar:** File, Edit, View, Run, Kernel, Tabs, Settings, Help
- File List:** intro\_to\_jupyter.ipynb
- Toolbar Buttons:** File, New, Cut, Copy, Paste, Run, Stop, Raw, Cell Type, Git, R, Cell Selection
- Text Cells:**
  - \*\*bold text\*\* displays **bold text**
  - \*italic text\* displays *italic text*
  - > block quote
    - block quote
- Bottom Status Bar:** Simple (button), 0 \$ 2 (button), R | Idle, Mode: Command, X, Ln 1, Col 38, 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
- Toolbar:** File, Edit, View, Run, Kernel, Tabs, Settings, Help
- File List:** intro\_to\_jupyter.ipynb
- Toolbar Buttons:** File, New, Cut, Copy, Paste, Run, Stop, Cell, Cell Kernel, Raw, Git, Cell Kernel Selection, R, Cell Selection
- Content Area:** ordered lists:
  - 1. item 1
  - 1. item 2
  - 1. item 3  
  - 1. item 1
  - 2. item 2
  - 3. item 3
- Bottom Status Bar:** Simple (button), 0 \$ 2 (button), R | Idle, Mode: Command, X, Ln 1, Col 38, 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
- Toolbar:** File, Edit, View, Run, Kernel, Tabs, Settings, Help
- File List:** intro\_to\_jupyter.ipynb
- Toolbar Buttons:** File, New, Cut, Copy, Paste, Run, Stop, Cell, Cell Kernel, Raw, Git, Cell Kernel Selection, R, Cell Selection
- Content Area:**
  - Text: unordered list:
  - Code block:
    - item 1
    - item 2
    - item 3
  - Text: • item 1
  - Text: • item 2
  - Text: • item 3
- Bottom Status Bar:** Simple (button), 0 \$ 2 (button), R | Idle, Mode: Command, X, Ln 1, Col 38, 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
- Address Bar:** localhost:8889/lab/tree/intro\_to\_jupyter.ipynb
- Toolbar:** File, Edit, View, Run, Kernel, Tabs, Settings, Help
- File List:** intro\_to\_jupyter.ipynb
- Tool Buttons:** File, New, Cut, Copy, Paste, Run, Stop, Cell, Next, Previous, Markdown, Git, Kernel, R, Help
- Code Block:** ``code``
- Text:** displays code
- Text:** you can link to webpages like this:
- Link Example:** [link title](http://www.example.com/)
- Link Text:** link title
- Bottom Status Bar:** Simple, 0 \$ 2, R | Idle, Mode: Command, X, 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
- Toolbar:** File, Edit, View, Run, Kernel, Tabs, Settings, Help
- File List:** intro\_to\_jupyter.ipynb
- Toolbar Buttons:** File, New, Delete, Copy, Paste, Run, Stop, Cell, Cell, Raw, Git, R, Fullscreen, Settings
- Text Content:** "and embed images like this:" followed by a code block containing the alt text for an image.
- Image Preview:** A small thumbnail of an orange and white kitten looking up.
- Bottom Status Bar:** Simple, 0 \$ 2, R | Idle, Mode: Command, Error icon, Ln 1, Col 1, intro\_to\_jupyter.ipynb

# Basic Markdown

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

- Header Bar:** Shows the title "intro\_to\_jupyter.ipynb" (2 tabs open), a back/forward button, a refresh button, a search bar with "localhost:8889/lab/tree/intro\_to\_jupyter.ipynb", a zoom level of "250%", a star icon, and a gear icon.
- Toolbar:** Includes File, Edit, View, Run, Kernel, Tabs, Settings, Help, and a Git icon.
- File List:** On the left, there's a sidebar with icons for file operations and a list of files: "intro\_to\_jupyter.ipynb" (selected).
- Content Area:** Displays the notebook content.
  - A text cell containing: "Additionally, I can directly embed `HTML`".
  - A code cell containing:

```
<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>
```
  - The rendered output of the code cell:
    - A light blue box with the text "this is an info box".
    - An orange box with the text "this is a warning box".
- Bottom Bar:** Shows "Simple" mode, cell counts (0, \$, 2), kernel status ("R | Idle"), mode ("Edit"), a shield icon, line/character count ("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:** intro\_to\_jupyter.ipynb (2) - Jupyter
- Toolbar:** File, Edit, View, Run, Kernel, Tabs, Settings, Help
- File List:** intro\_to\_jupyter.ipynb
- Tool Buttons:** File, New, Cut, Copy, Paste, Run, Stop, Kernel, Markdown, Cell Type, Git
- Content Area:**
  - I can also embed *LATEX* mathematics type
  - Mathematical code cell:

```
$$
x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}
$$
```
  - Output:
$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$
- Bottom Status Bar:** Simple, 0, \$, 2, R | Idle, Mode: Command, X, Ln 1, Col 1, 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:

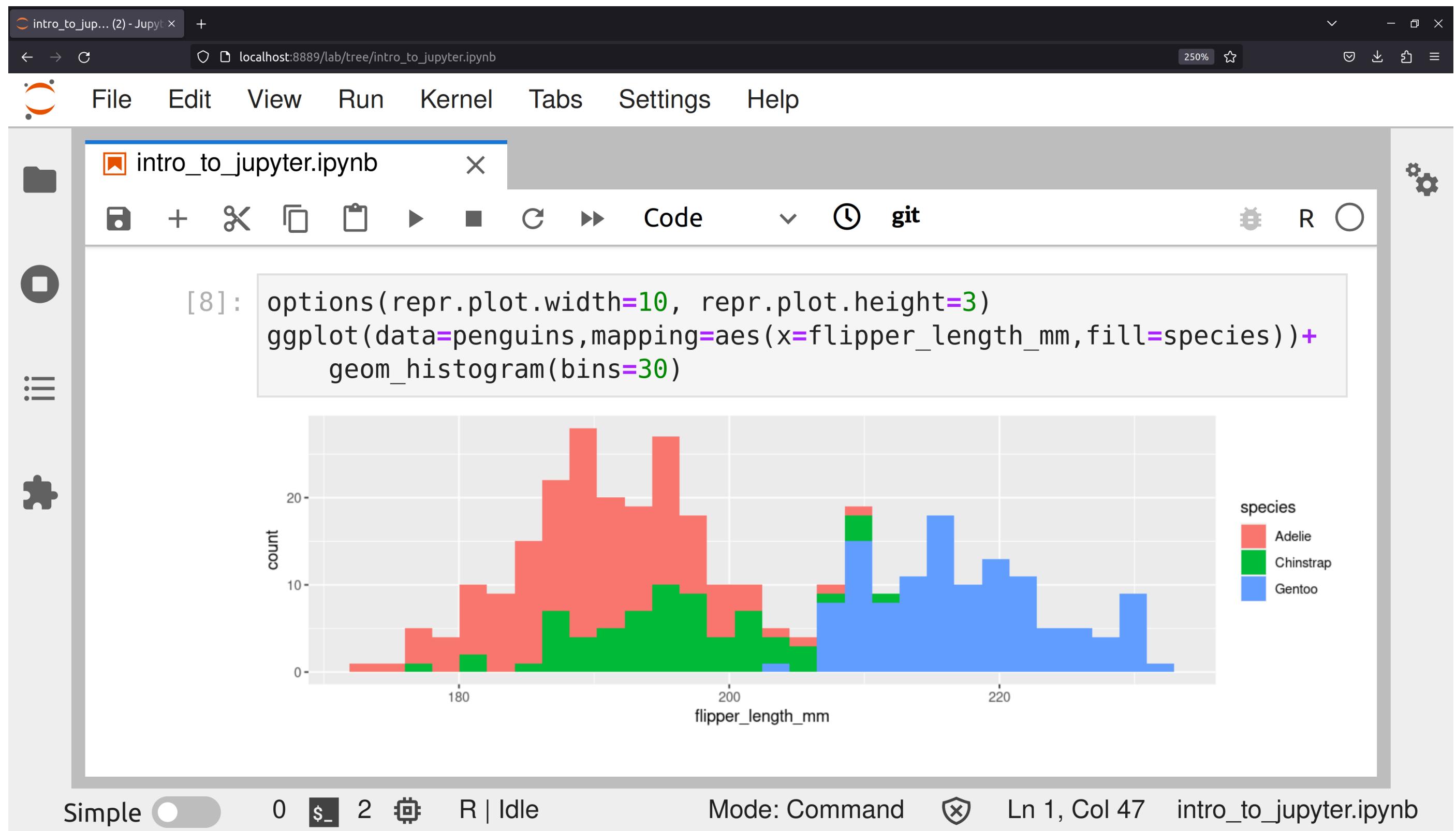
- a heading
- bold and italic text
- a numbered and un-numbered list of items
- displayed pseudo code
- a link to an external webpage
- an image
- embedded math in LaTeX

After writing these blocks of markdown, re-arrange them

- copy and past one
- delete one
- move one

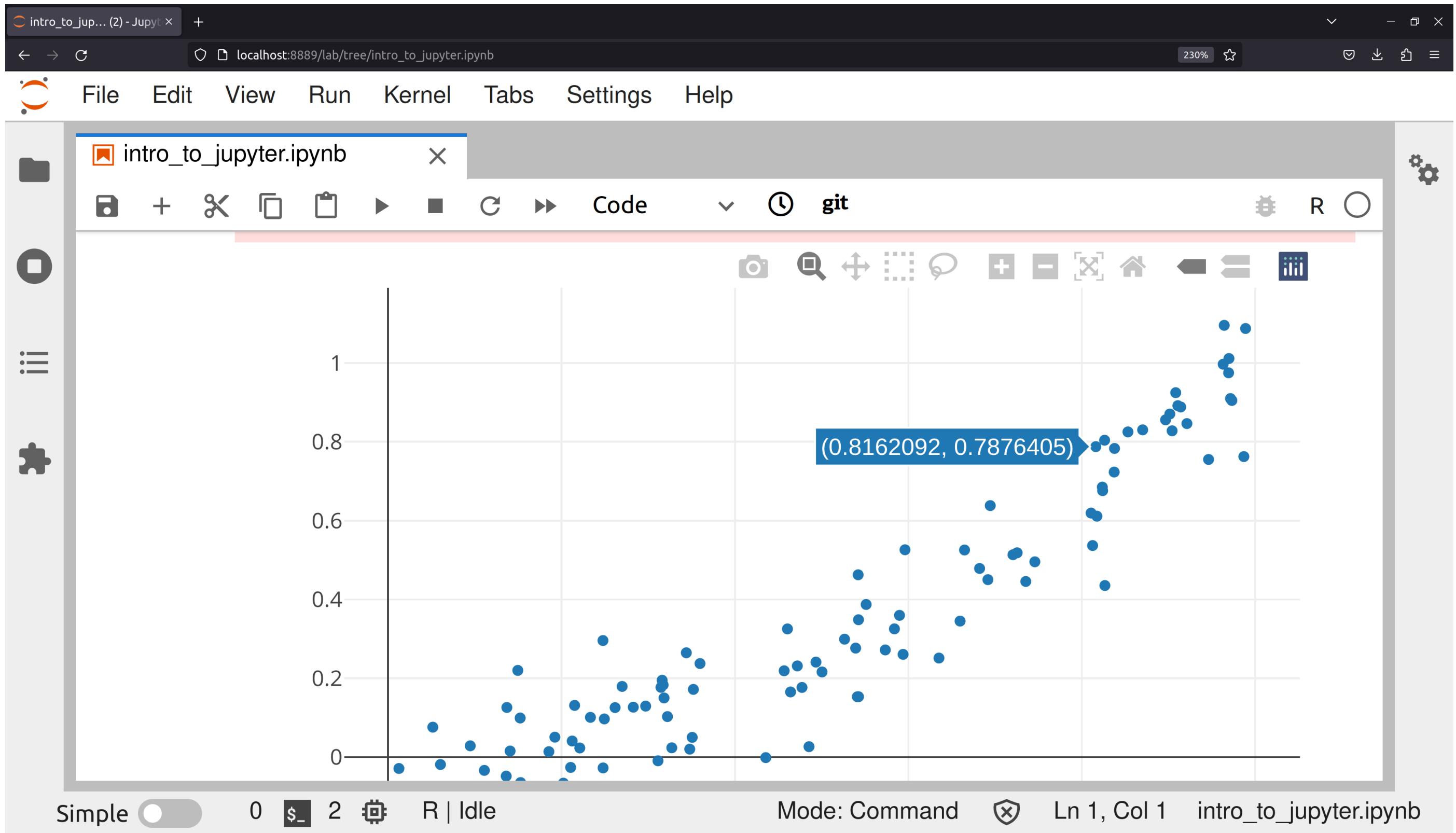
# Code: Basic Output

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



# 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,
- ...

## 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:

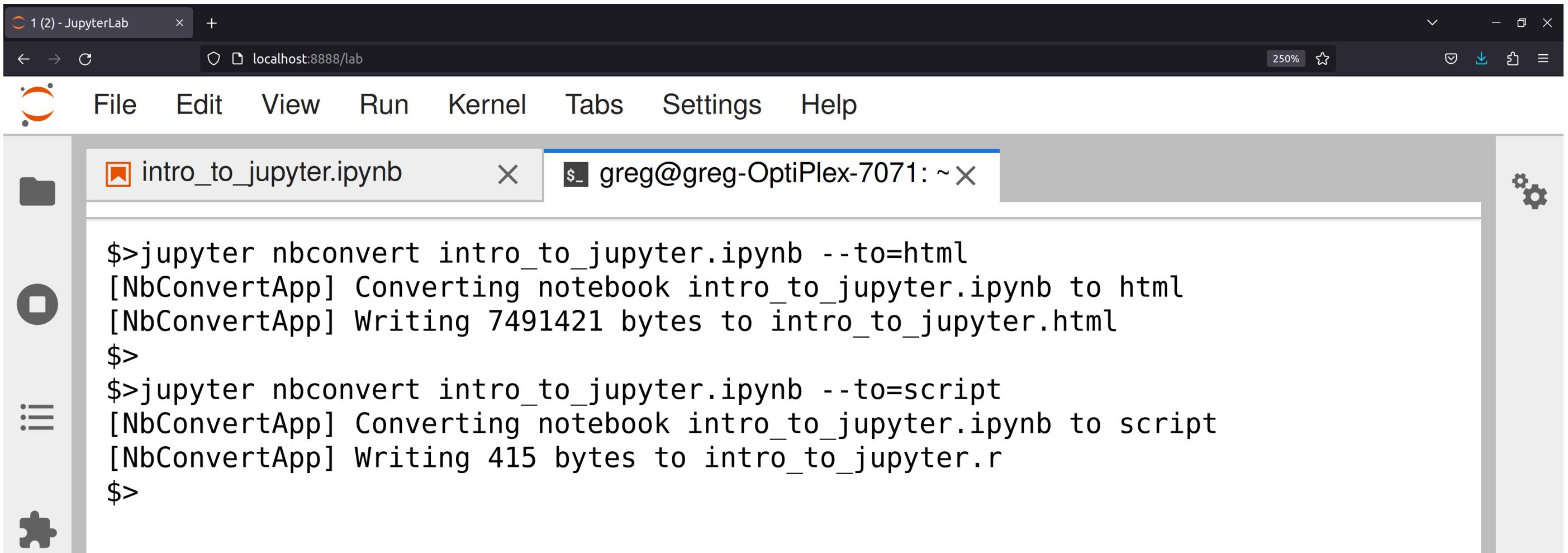
```
p <- plot_ly(x = ___, y = ___,  
mode = "markers", type = "scatter")  
embed_notebook(p)
```

# 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
```

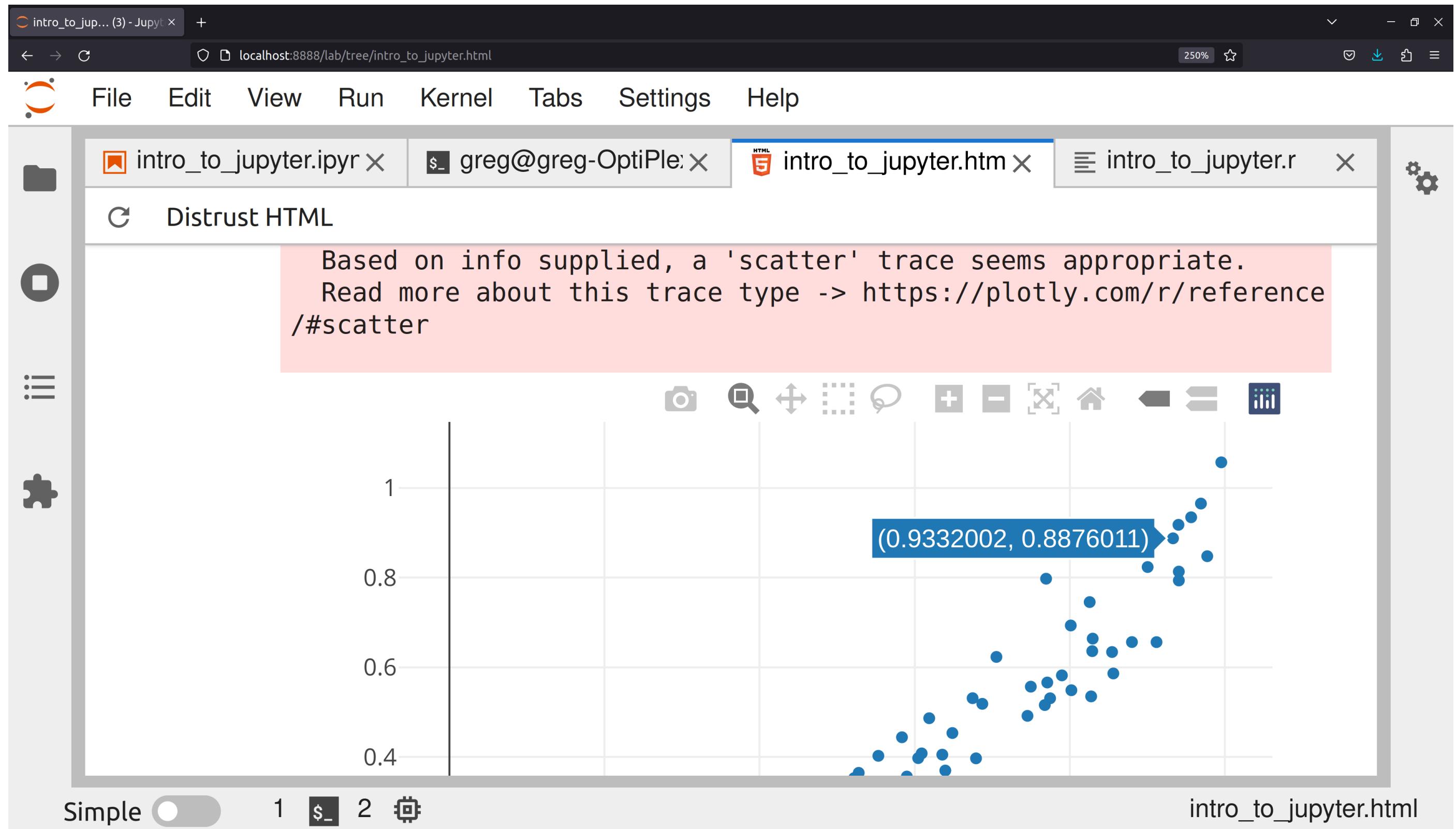


The screenshot shows the JupyterLab interface. At the top, there's a header bar with a tab labeled "1 (2) - JupyterLab". Below the header is a toolbar with icons for back, forward, search, and refresh, followed by the URL "localhost:8888/lab". On the right side of the header are zoom controls (250%), a star icon, and other navigation icons. The main area has a dark background with light-colored text. On the left, there's a sidebar with icons for file, folder, and settings. The central workspace contains two tabs: one titled "intro\_to\_jupyter.ipynb" and another titled "greg@greg-OptiPlex-7071: ~". A terminal window is open, displaying 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  
$>
```

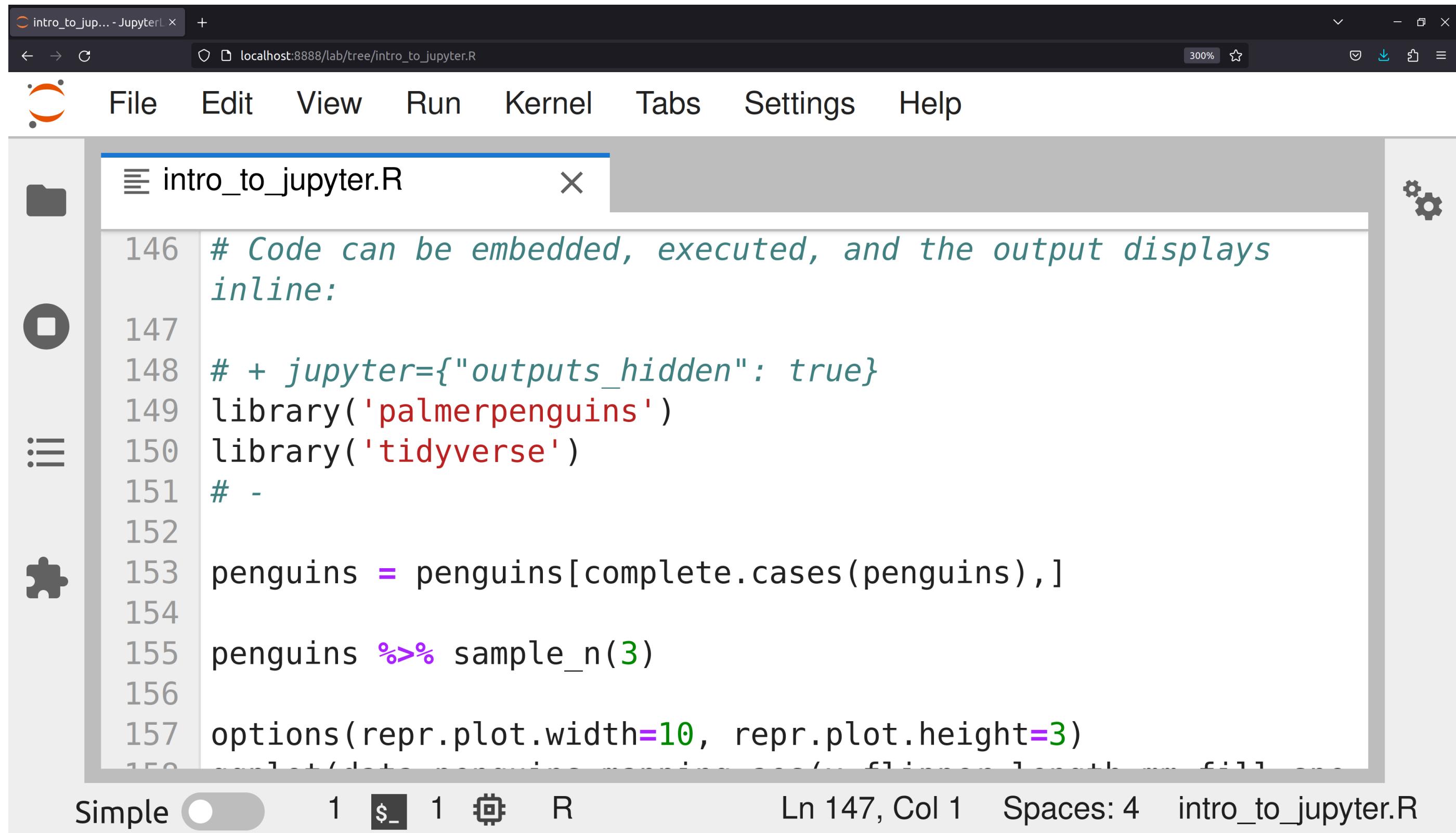
# Exporting: HTML

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 dark theme. The top bar includes tabs for 'intro\_to\_jupyter.R - JupyterLab' and '+', and a URL 'localhost:8888/lab/tree/intro\_to\_jupyter.R'. The main menu bar has options: File, Edit, View, Run, Kernel, Tabs, Settings, and Help. On the left, there's a sidebar with icons for file operations like folder, copy, paste, and settings. The central code editor window is titled 'intro\_to\_jupyter.R' and contains the following 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
153 penguins = penguins[complete.cases(penguins), ]
154
155 penguins %>% sample_n(3)
156
157 options(repr.plot.width=10, repr.plot.height=3)
```

The status bar at the bottom shows 'Simple' mode, line 1, column 1, and the file name 'intro\_to\_jupyter.R'.

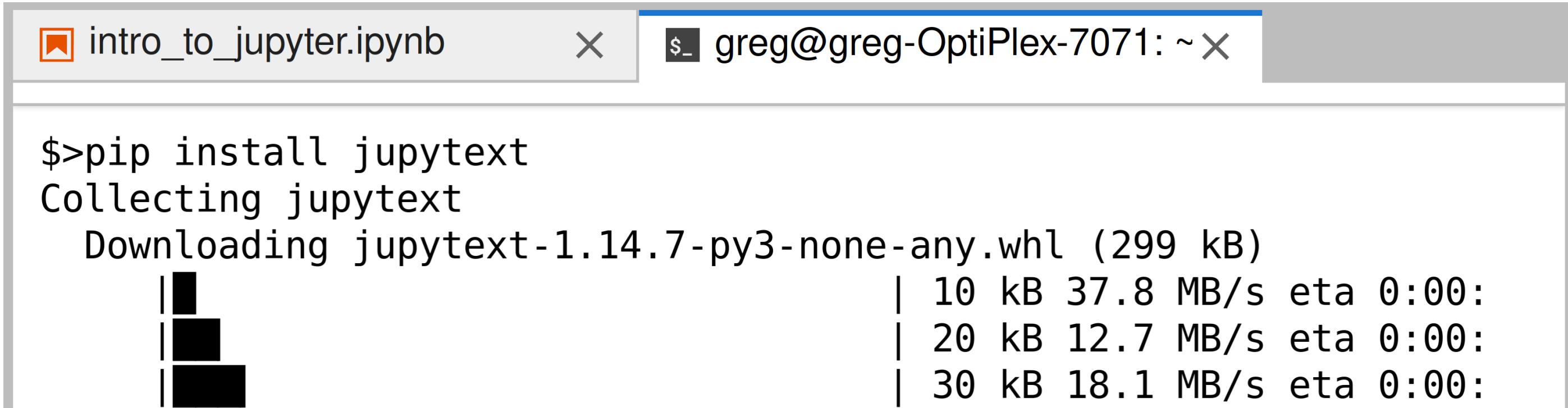
# Notebook Interoperability

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
```



The screenshot shows a terminal window with two tabs. The active tab is titled '\$ greg@greg-OptiPlex-7071: ~' and contains the command '\$>pip install jupytext'. Below the command, the terminal shows the progress of the download of the 'jupytext-1.14.7-py3-none-any.whl' file, with a progress bar and estimated download speeds of 37.8 MB/s, 12.7 MB/s, and 18.1 MB/s. The non-active tab is titled 'intro\_to\_jupyter.ipynb'.

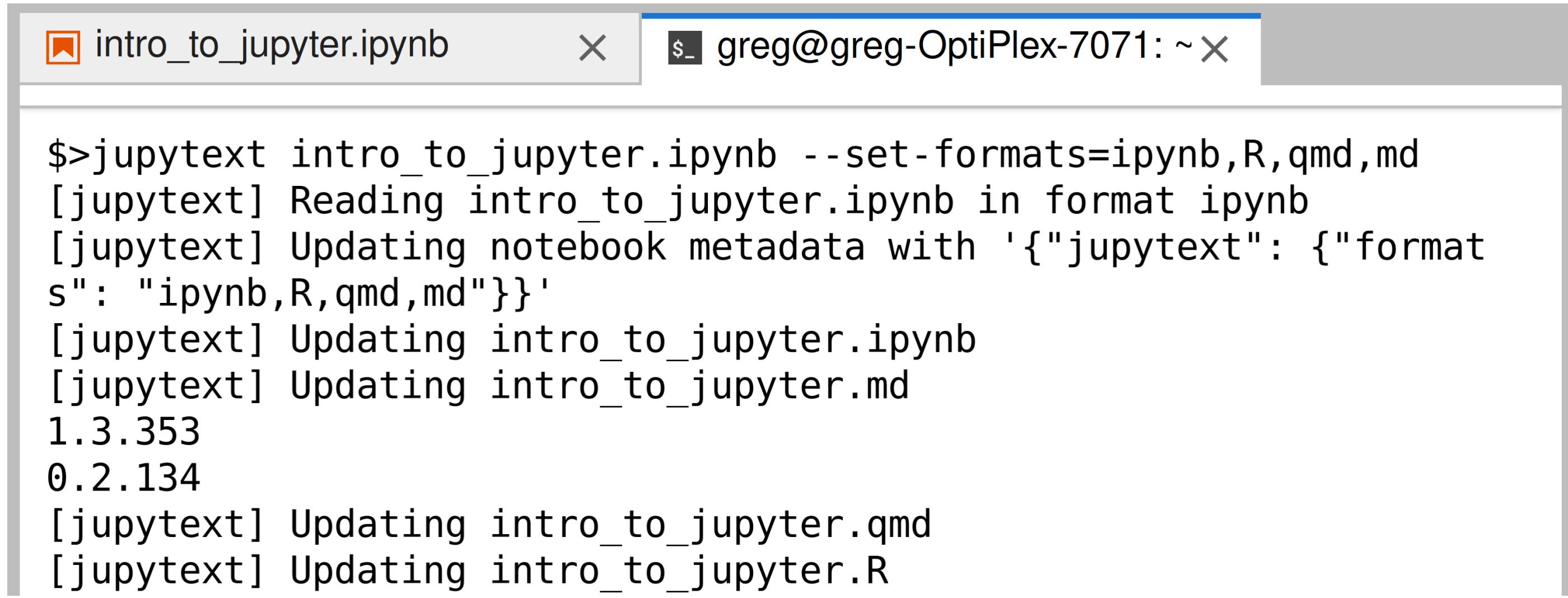
```
$>pip install jupytext
Collecting jupytext
  Downloading jupytext-1.14.7-py3-none-any.whl (299 kB)
|██████████| 10 kB 37.8 MB/s eta 0:00:
|██████████| 20 kB 12.7 MB/s eta 0:00:
|██████████| 30 kB 18.1 MB/s eta 0:00:
```

# 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,...
```

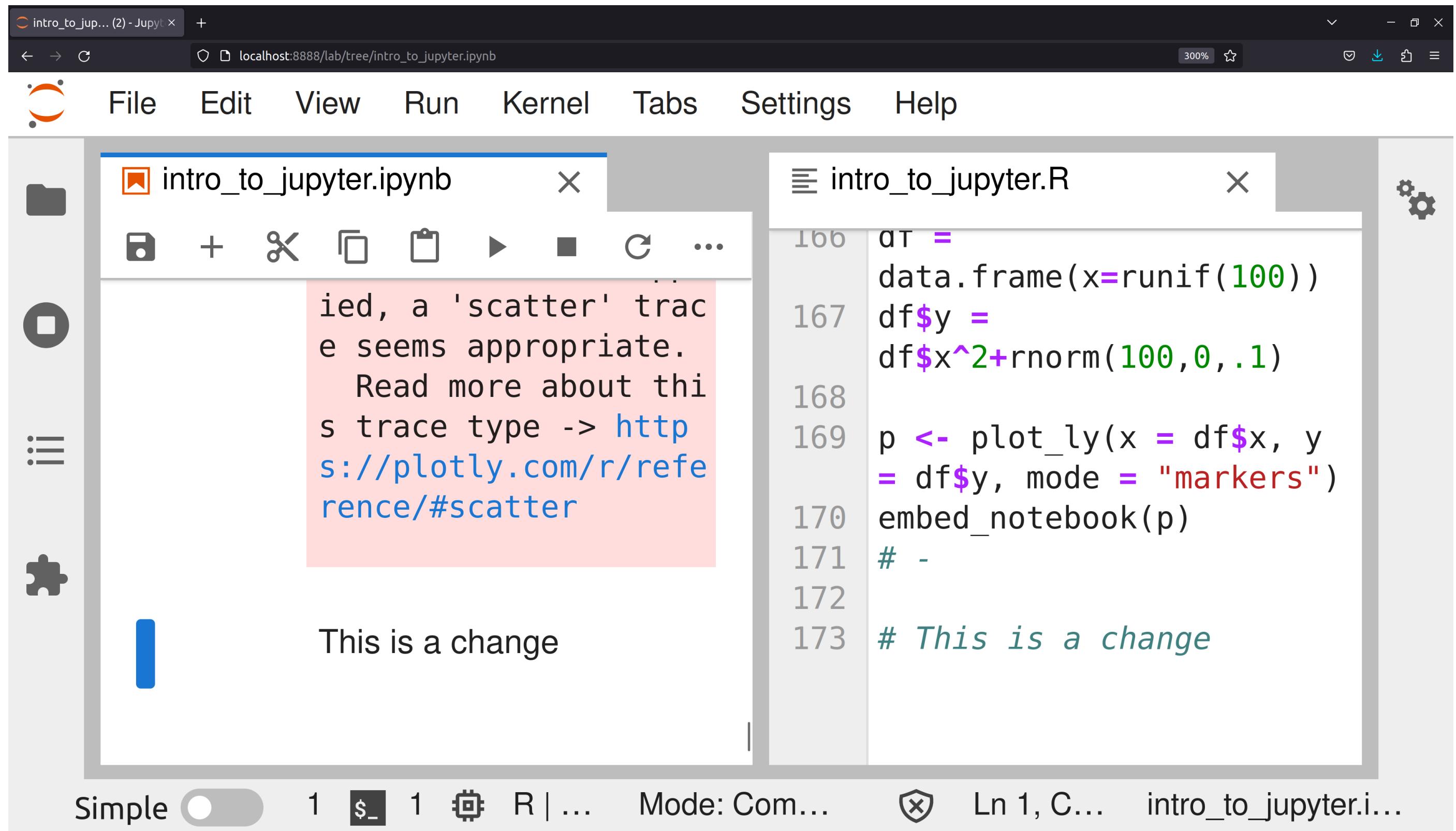


The screenshot shows a terminal window with a light gray background. At the top, there is a header bar with a file icon, the path "intro\_to\_jupyter.ipynb", a close button, and the user information "\$ greg@greg-OptiPlex-7071: ~". The main area of the terminal contains the following text:

```
$>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 Jupyter Notebook interface with two tabs open:

- intro\_to\_jupyter.ipynb**: This tab contains a red box highlighting a note about scatter plots and a message "This is a change".
- intro\_to\_jupyter.R**: This tab shows R code for generating a scatter plot.

```
166 qrt =  
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`
- 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).
- 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
  - 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)
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?