

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 a notebook titled 'example.ipynb'. The interface includes a menu bar (File, Edit, View, Run, Kernel, Tabs, Settings, Help), a toolbar with icons for file operations and execution, and a main workspace. The workspace contains two code cells. The first cell contains the text 'First, let's load the **penguins** dataset' followed by a code cell with the following R code:

```
[1]: library("palmerpenguins")  
library('tidyverse')
```

 Below the code cell is a 'suppressed output' indicator (three dots). The second code cell contains the R code:

```
[2]: penguins %>% sample_n(3)
```

 Below the second code cell is a table of output data. The table is titled 'A tibble: 3 x 8' and has columns: species, island, bill_length_mm, bill_depth_mm, flipper_length_mm, body_mass_g, sex, and year. The data rows are: Adelie, Torgersen, 42.5, 20.7, 197, 4500, male, 2007; and Adelie, Biscoe, 41.1, 18.2, 192, 4050, male, 2008. Red annotations with arrows point to various elements: 'rich text' points to the introductory text; 'code' points to the R code in the first cell; 'suppressed output' points to the three dots; and 'output' points to the table of data. The status bar at the bottom shows 'Simple' mode, kernel 'R | Idle', and file path 'example.ipynb'.

rich text

code

suppressed output

output

```
First, let's load the penguins dataset
```

```
[1]: library("palmerpenguins")  
library('tidyverse')
```

```
...
```

```
[2]: penguins %>% sample_n(3)
```

A tibble: 3 x 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

Simple 0 \$ 1 R | Idle Mode: Command Ln 1, Col 35 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. At the top, there is a browser window with the address bar showing 'localhost:8889/lab/tree/intro_to_jupyter.ipynb'. Below the browser is the Jupyter Notebook menu bar with options: File, Edit, View, Run, Kernel, Tabs, Settings, Help. The notebook interface has a toolbar with icons for file operations and a 'Code' dropdown menu. The main content area shows a code cell with the following text:

```
headings can be written using successive indentation by # , e.g.  
  
# heading level 1  
## heading level 2  
### heading level 3
```

Below the code cell, the rendered output is displayed:

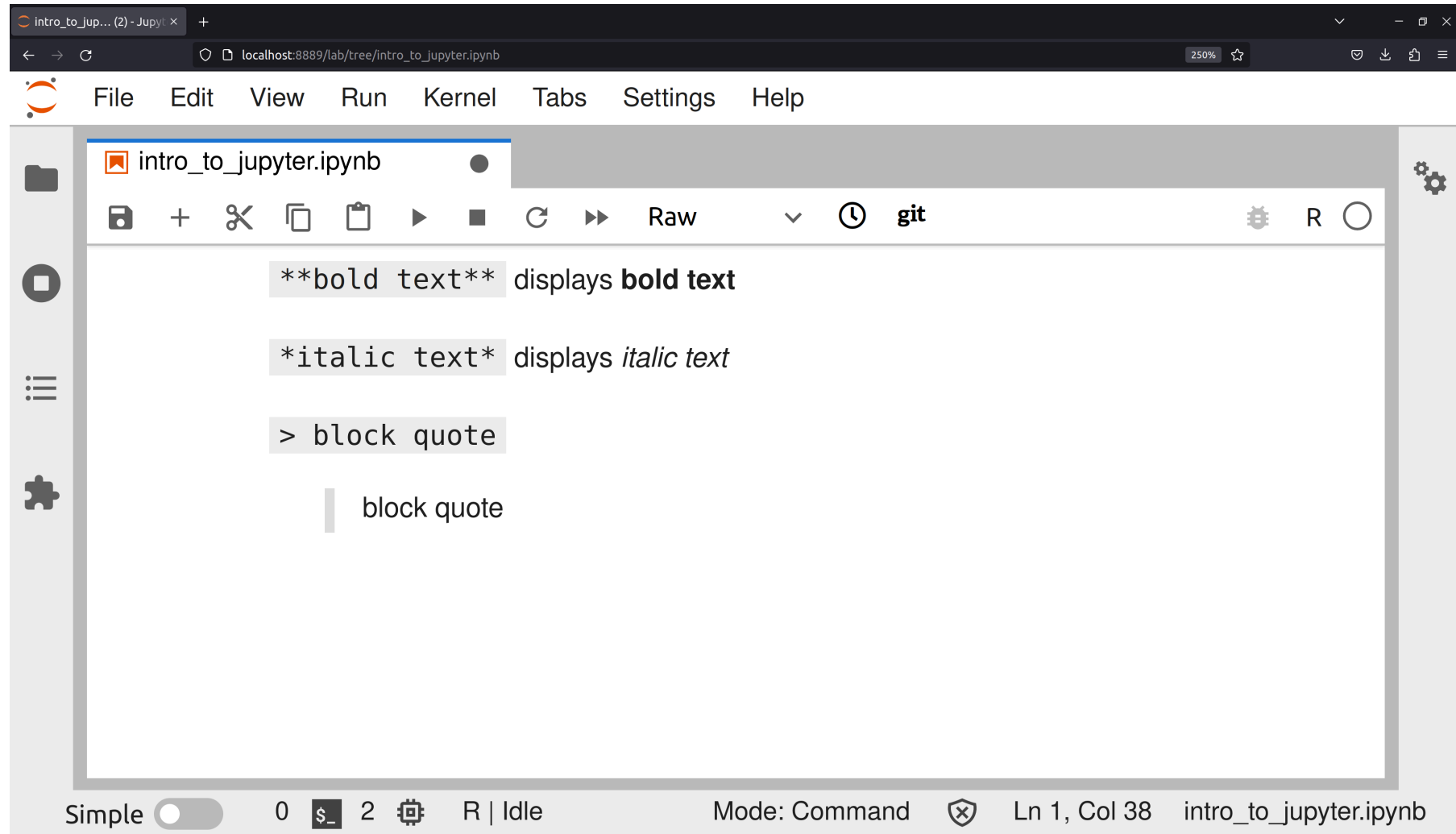
heading level 1

heading level 2

heading level 3

At the bottom of the notebook, there is a status bar showing 'Simple' (with a toggle), '0 \$ 2', 'R | Idle', 'Mode: Command', and 'Ln 1, Col 1 intro_to_jupyter.ipynb'.

BASIC MARKDOWN



The screenshot shows a JupyterLab interface with a browser window at the top displaying the URL `localhost:8889/lab/tree/intro_to_jupyter.ipynb`. Below the browser is a menu bar with options: File, Edit, View, Run, Kernel, Tabs, Settings, Help. The main workspace contains a file named `intro_to_jupyter.ipynb` with a toolbar above it including icons for save, add, delete, copy, paste, run, and raw. The code editor shows the following markdown examples:

```
**bold text** displays bold text
```

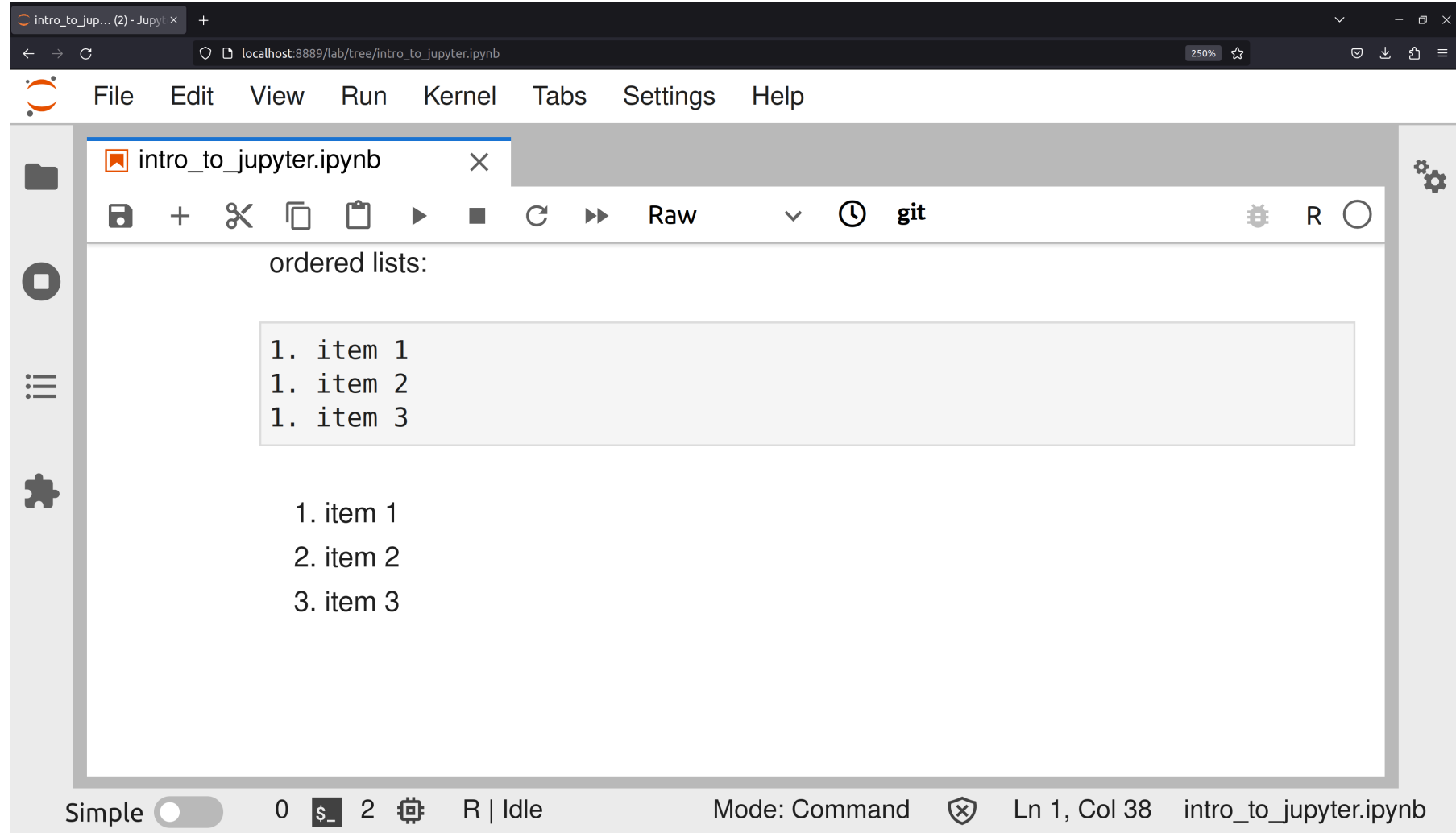
```
*italic text* displays italic text
```

```
> block quote
```

```
    block quote
```

At the bottom of the interface, there is a status bar showing "Simple" (with a toggle), "0 \$ 2", "R | Idle", "Mode: Command", and "Ln 1, Col 38 intro_to_jupyter.ipynb".

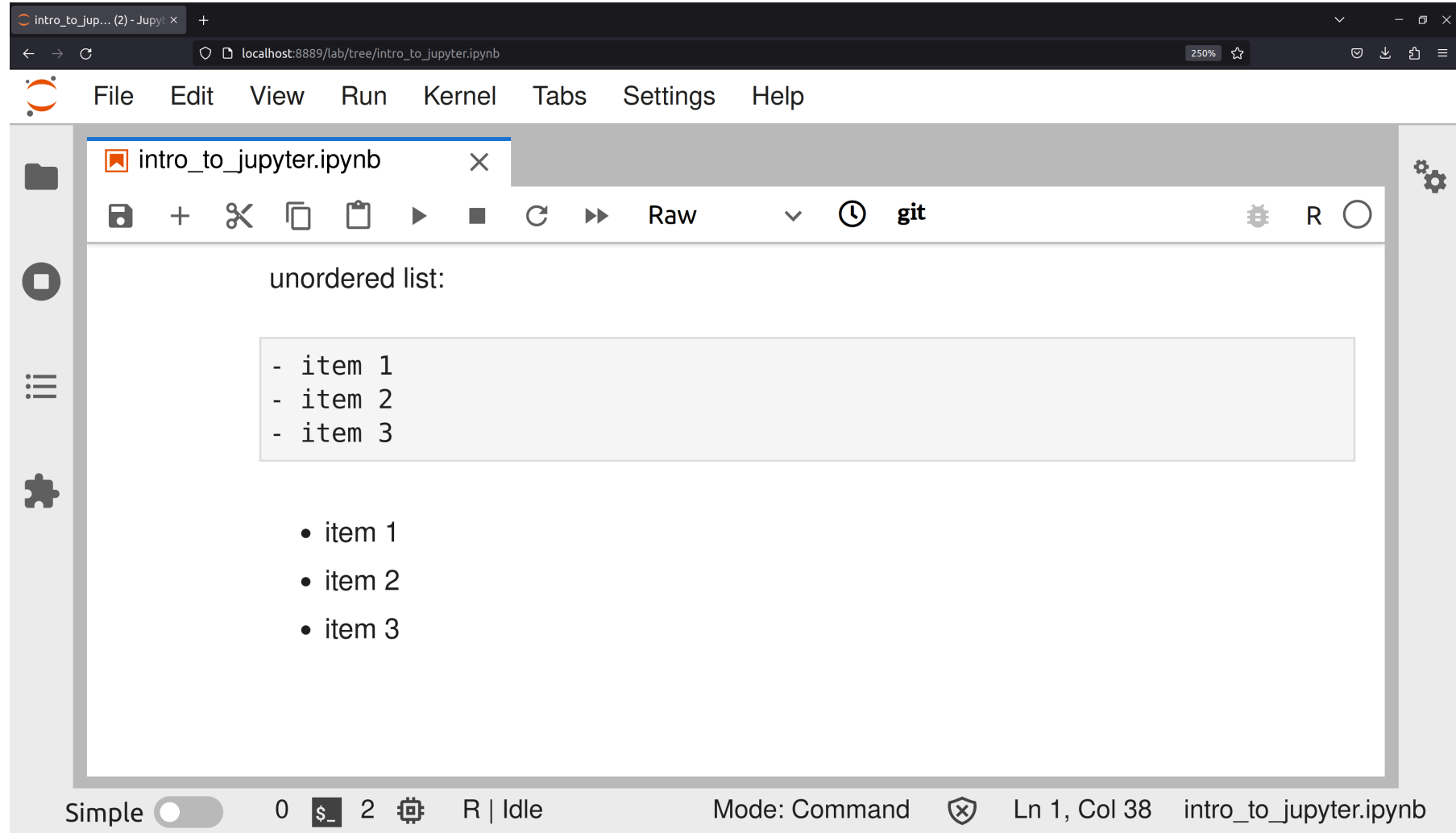
BASIC MARKDOWN



The screenshot shows a Jupyter Notebook interface. The browser address bar indicates the URL is localhost:8889/lab/tree/intro_to_jupyter.ipynb. The notebook's menu bar includes File, Edit, View, Run, Kernel, Tabs, Settings, and Help. The current cell is a code cell containing the text "ordered lists:" followed by a list of three items: "1. item 1", "1. item 2", and "1. item 3". The status bar at the bottom shows "Simple" mode, 0 lines of code, 2 cells, and the kernel is in "Idle" state. The current mode is "Command", and the cursor is at line 1, column 38.

```
ordered lists:  
  
1. item 1  
1. item 2  
1. item 3  
  
1. item 1  
2. item 2  
3. item 3
```


BASIC MARKDOWN



The screenshot shows a Jupyter Notebook interface in a web browser. The browser address bar shows the URL `localhost:8889/lab/tree/intro_to_jupyter.ipynb`. The notebook's menu bar includes `File`, `Edit`, `View`, `Run`, `Kernel`, `Tabs`, `Settings`, and `Help`. The notebook title is `intro_to_jupyter.ipynb`. The toolbar contains icons for saving, adding, deleting, copying, pasting, running, and other actions. The main content area displays a markdown cell with the text `unordered list:` followed by two lists of items. The first list is a bulleted list with three items: `- item 1`, `- item 2`, and `- item 3`. The second list is a bulleted list with three items: `• item 1`, `• item 2`, and `• item 3`. The status bar at the bottom shows `Simple` (with a toggle), `0`, `$_`, `2`, `R | Idle`, `Mode: Command`, `Ln 1, Col 38`, and `intro_to_jupyter.ipynb`.

```
unordered list:
```

- item 1
- item 2
- item 3

- item 1
- item 2
- item 3

BASIC MARKDOWN

The screenshot shows a Jupyter Notebook interface with a dark theme. The browser address bar shows the URL `localhost:8889/lab/tree/intro_to_jupyter.ipynb`. The notebook title is `intro_to_jupyter.ipynb`. The interface includes a menu bar with `File`, `Edit`, `View`, `Run`, `Kernel`, `Tabs`, `Settings`, and `Help`. The notebook content area shows the following text:

```
`code`
```

displays `code`

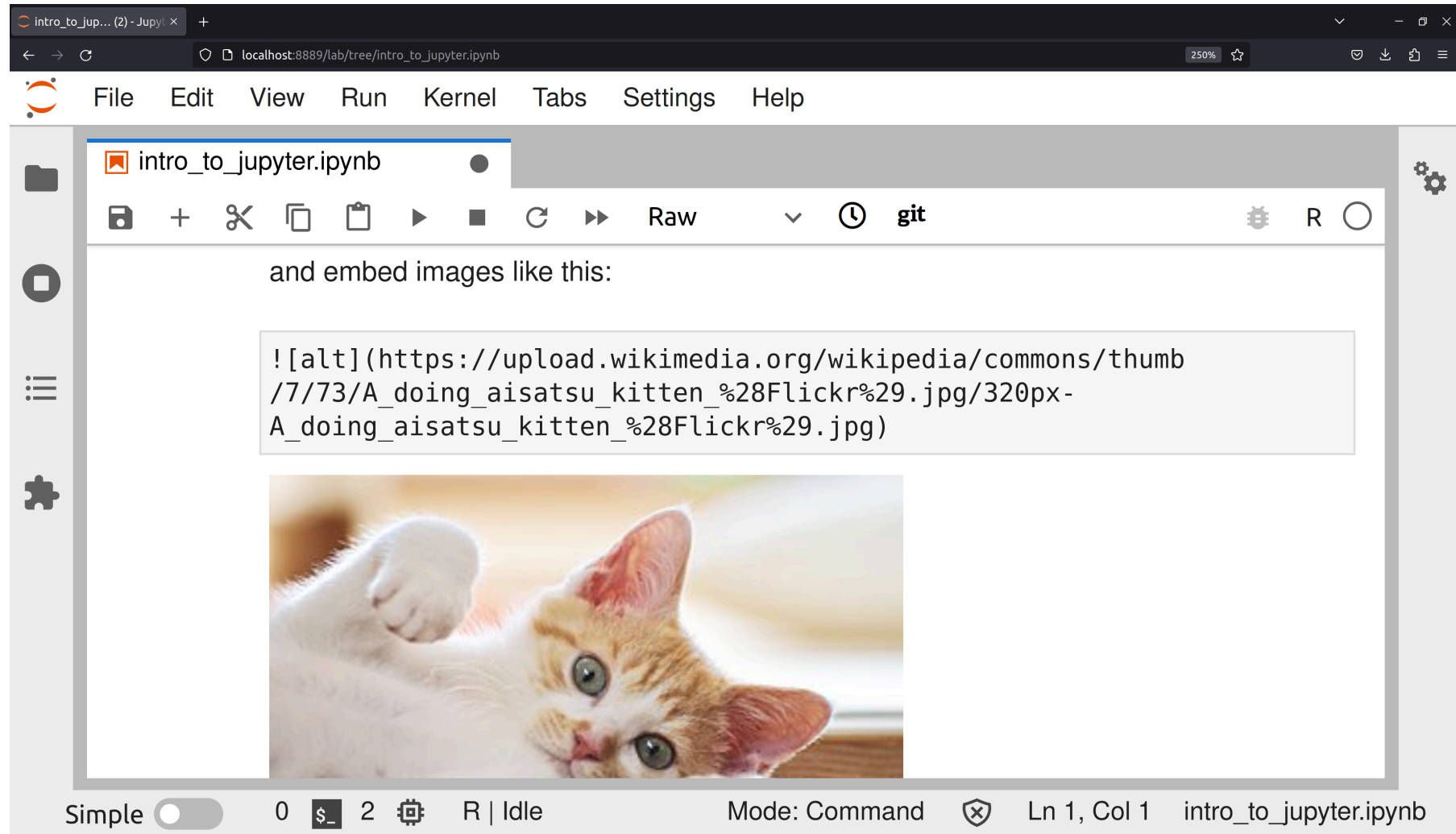
you can link to webpages like this:

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

[link title](#)

The status bar at the bottom shows `Simple` (with a toggle), `0` lines, `2` cells, `R | Idle` kernel, `Mode: Command`, and `Ln 1, Col 1 intro_to_jupyter.ipynb`.

BASIC MARKDOWN



The screenshot shows a Jupyter Notebook interface in a web browser. The browser address bar shows the URL `localhost:8889/lab/tree/intro_to_jupyter.ipynb`. The notebook's menu bar includes `File`, `Edit`, `View`, `Run`, `Kernel`, `Tabs`, `Settings`, and `Help`. The notebook title is `intro_to_jupyter.ipynb`. The toolbar contains icons for saving, adding, deleting, copying, pasting, running, and other actions. The main content area shows a Markdown cell with the text "and embed images like this:" followed by a code block containing the following Markdown image syntax:

```
![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)
```

Below the code block, the image of a white and orange kitten is rendered. The bottom status bar shows the notebook is in `Simple` mode, with `0` lines of code, `2` cells, and the kernel is `R | Idle`. The current cursor position is `Ln 1, Col 1` in the file `intro_to_jupyter.ipynb`.

BASIC MARKDOWN

The screenshot shows a JupyterLab interface with a browser window at the top displaying the URL `localhost:8889/lab/tree/intro_to_jupyter.ipynb`. Below the browser is the JupyterLab menu bar with options: File, Edit, View, Run, Kernel, Tabs, Settings, Help. The main workspace shows a file named `intro_to_jupyter.ipynb` in edit mode. The code cell contains the following HTML code:

```
<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 consists of two alert boxes: a light blue box containing the text "this is an info box" and an orange box containing the text "this is a warning box".

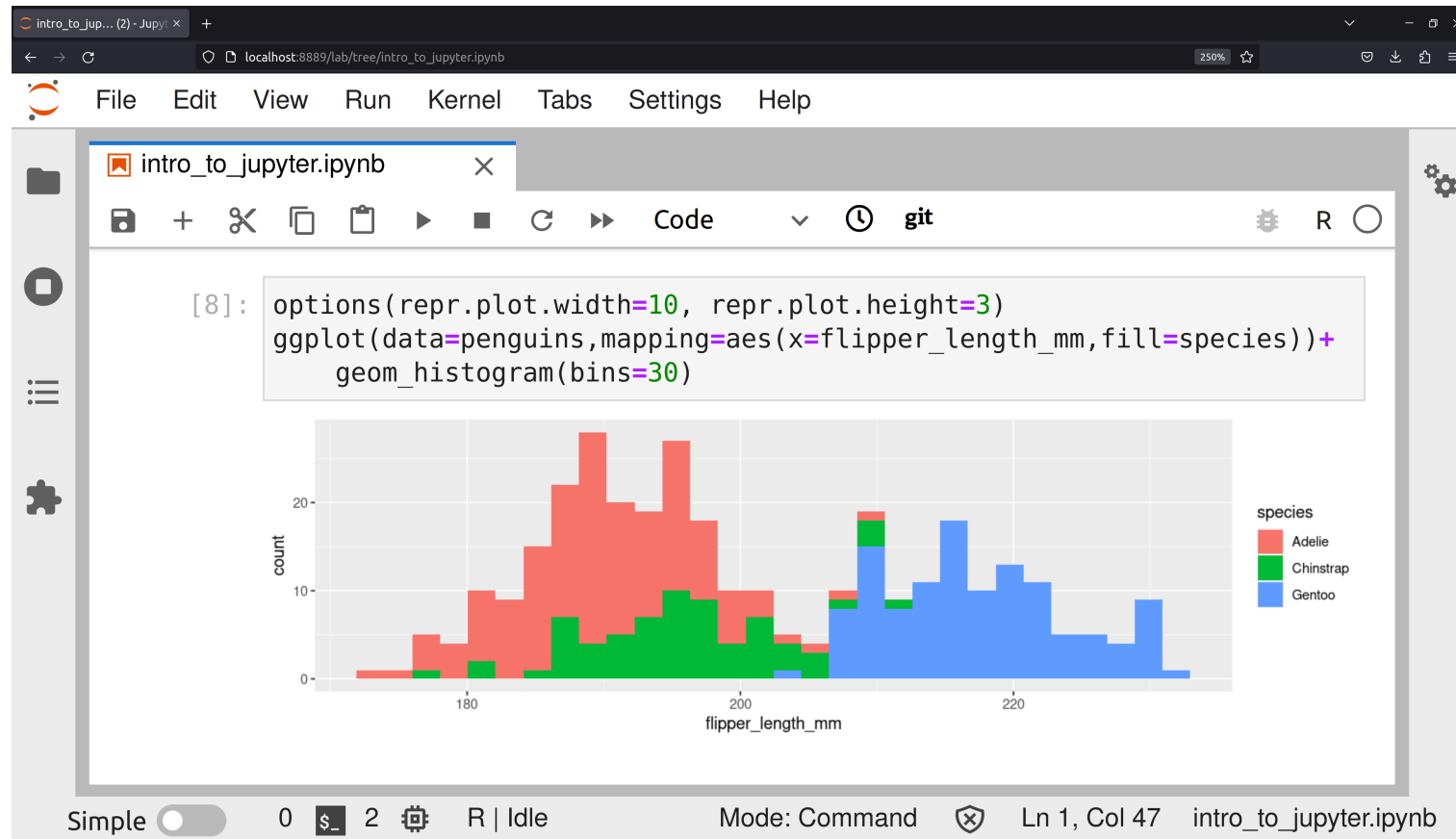
At the bottom of the interface, the status bar shows "Simple" (with a toggle switch), "0 \$ 2" (with a kernel icon), "R | Idle", "Mode: Edit" (with a shield icon), and "Ln 1, Col 121 intro_to_jupyter.ipynb".

BASIC MARKDOWN

The image shows a Jupyter Notebook interface in a web browser. The browser address bar shows the URL `localhost:8889/lab/tree/intro_to_jupyter.ipynb`. The notebook's menu bar includes `File`, `Edit`, `View`, `Run`, `Kernel`, `Tabs`, `Settings`, and `Help`. The notebook title is `intro_to_jupyter.ipynb`. The toolbar contains icons for file operations and a dropdown menu set to `Markdown`. The main content area contains the text `I can also embed LATEX mathematics type`. Below this text is a code editor box containing the LaTeX code `$$ x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} $$`. The rendered output of this code is the quadratic formula
$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$
. The status bar at the bottom shows `Simple` (with a toggle), `0 $ 2`, `R | Idle`, `Mode: Command`, `Ln 1, Col 1`, and `intro_to_jupyter.ipynb`.

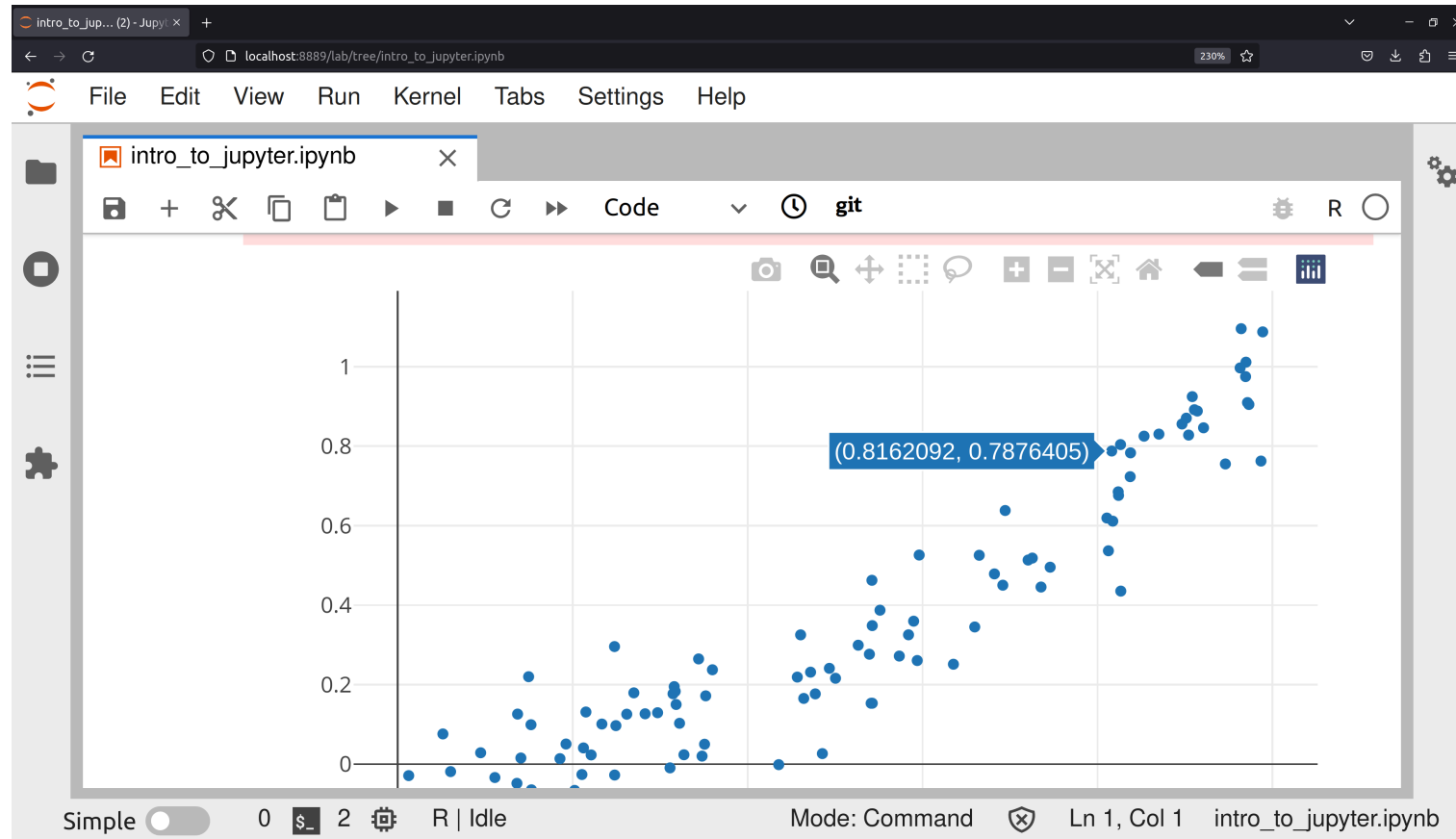
CODE: BASIC OUTPUT

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:

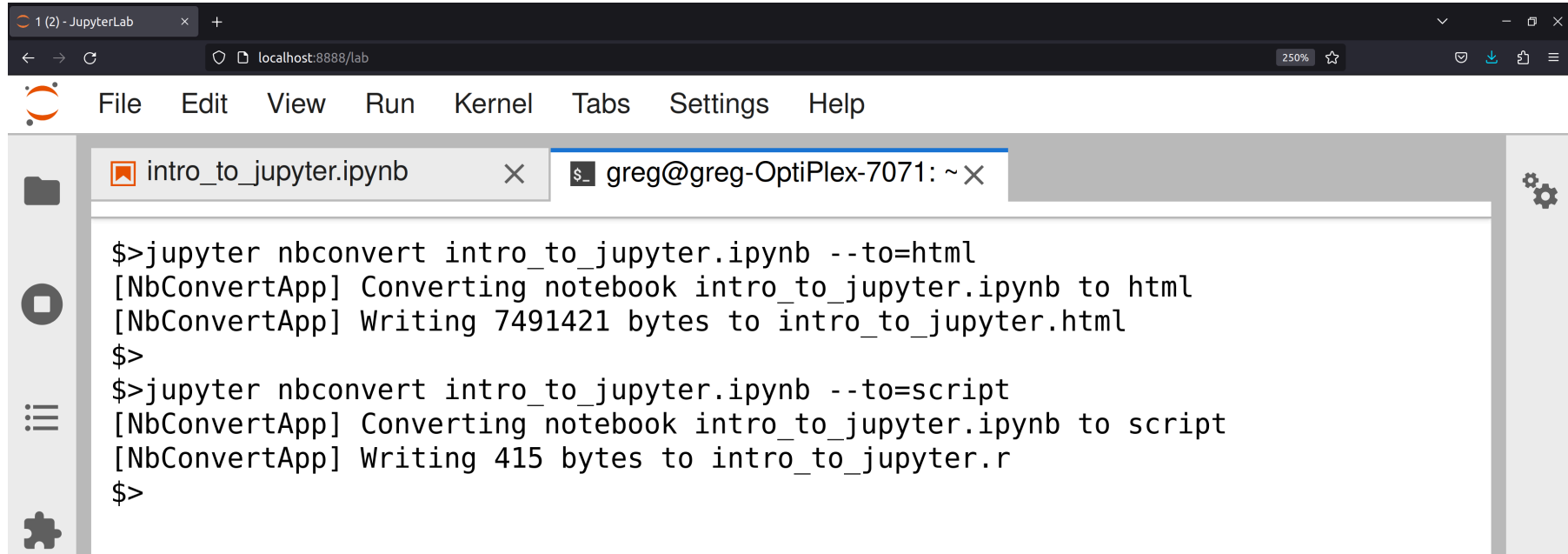
- python,
- r,
- julia,
- stata,
- sql,
- octave,
- matlab,
- java,
- go,
- C,
- ...

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
```

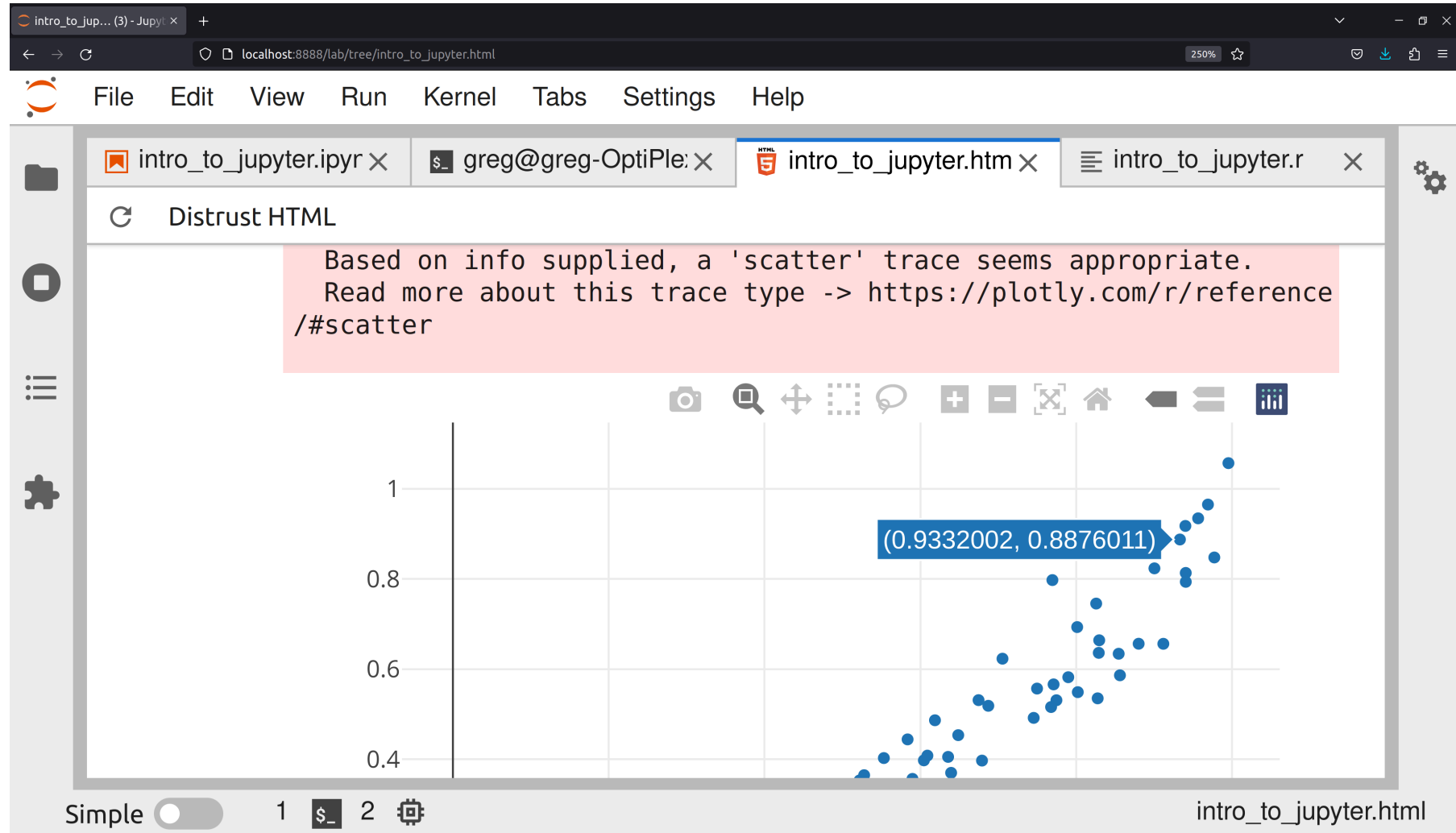


The screenshot shows a JupyterLab interface with a terminal window open. The terminal displays the following commands and output:

```
$>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
$>
```

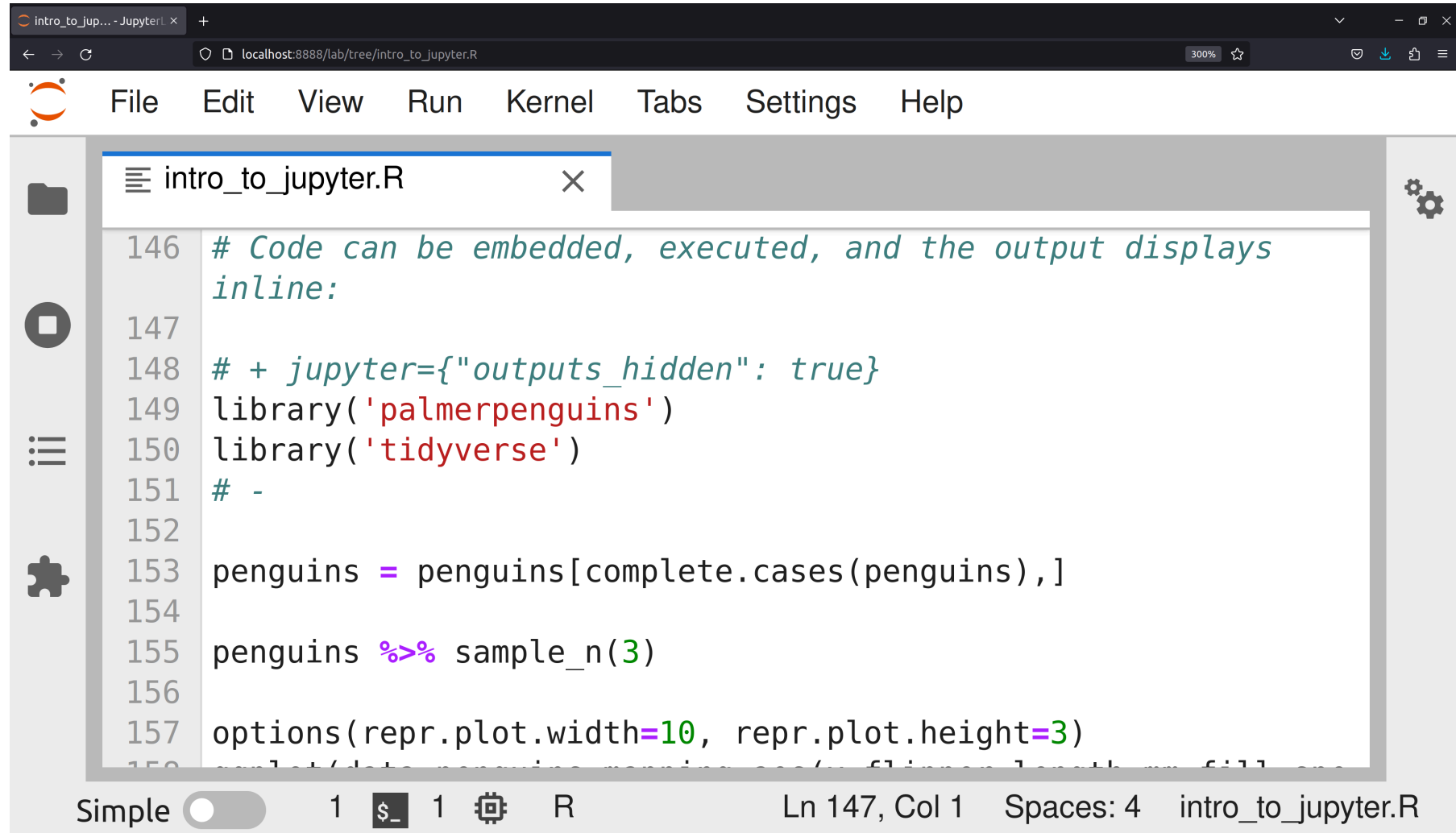
EXPORTING: HTML

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



EXPORTING: SCRIPT

Exporting as a script gives us basic executable script



```
intro_to_jup... - JupyterL x +
localhost:8888/lab/tree/intro_to_jupyter.R
300% ☆
File Edit View Run Kernel Tabs Settings Help
intro_to_jupyter.R x
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
Simple 1 $ 1 R Ln 147, Col 1 Spaces: 4 intro_to_jupyter.R
```

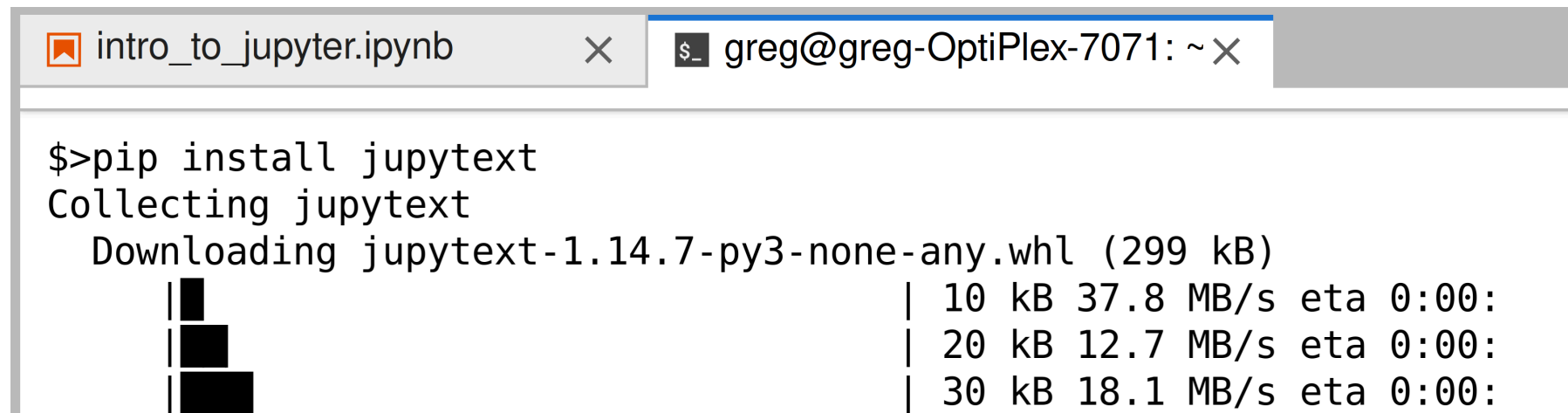
NOTEBOOK INTEROPERABILITY

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

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

We can install via

```
pip install jupyter
```



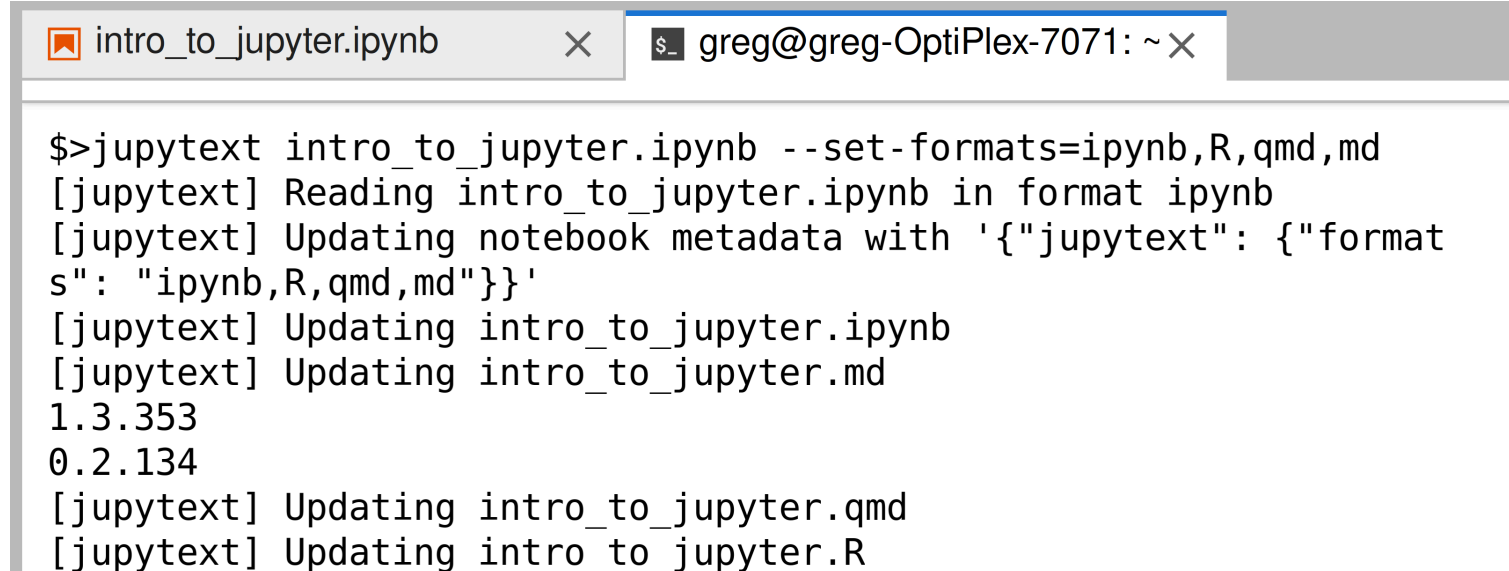
```
intro_to_jupyter.ipynb x greg@greg-OptiPlex-7071: ~ x
$>pip install jupyter
Collecting jupyter
  Downloading jupyter-1.14.7-py3-none-any.whl (299 kB)
  | 10 kB 37.8 MB/s eta 0:00:
  | 20 kB 12.7 MB/s eta 0:00:
  | 30 kB 18.1 MB/s eta 0:00:
```

NOTEBOOK INTEROPERABILITY

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

via the command

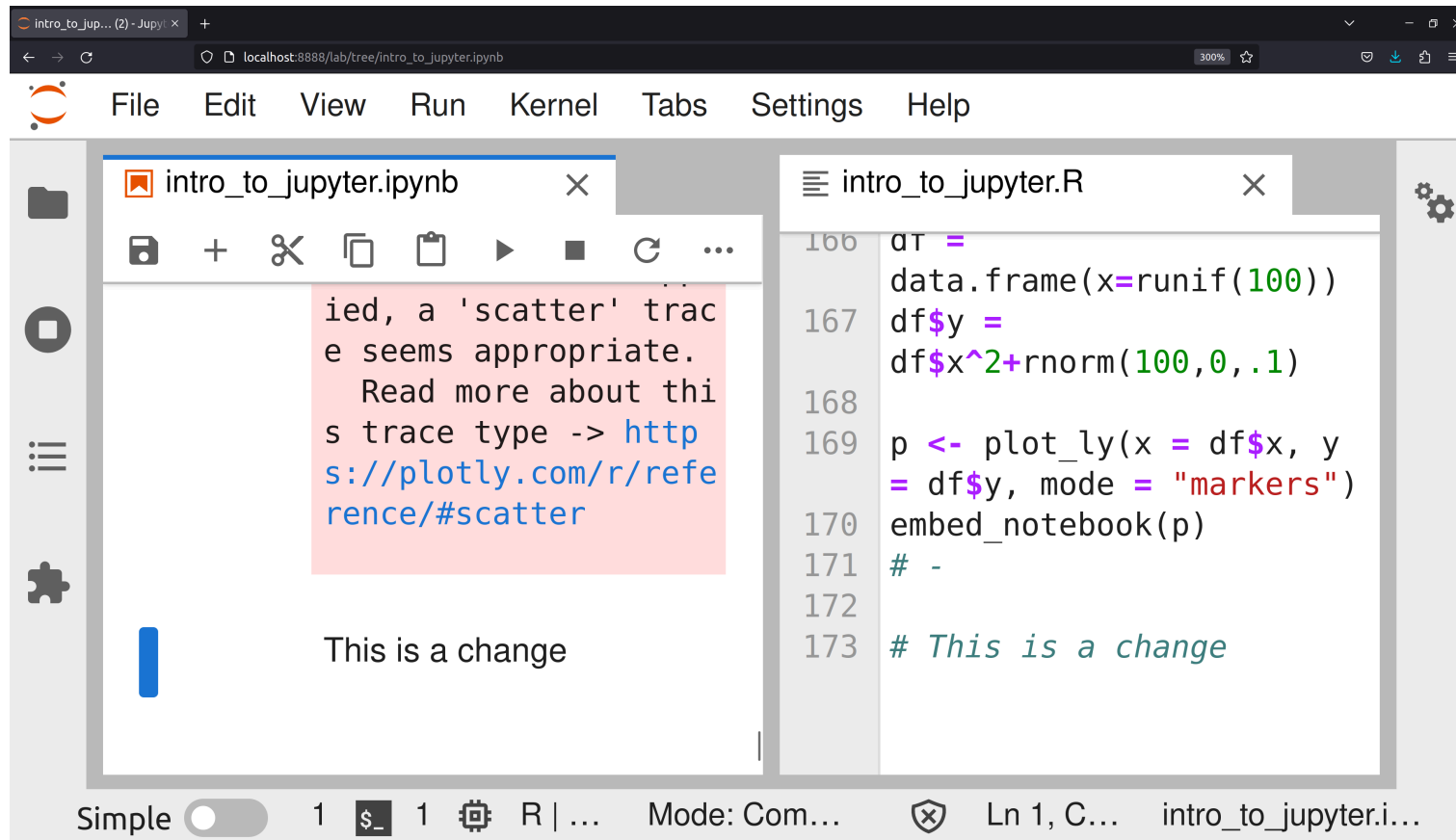
```
jupyter notebook.ipynb --set-formats=format1,format2,...
```



```
intro_to_jupyter.ipynb x greg@greg-OptiPlex-7071: ~ x
$>jupyter notebook.ipynb --set-formats=ipynb,R,qmd,md
[jupyter] Reading intro_to_jupyter.ipynb in format ipynb
[jupyter] Updating notebook metadata with '{"jupyter": {"formats": "ipynb,R,qmd,md"}}'
[jupyter] Updating intro_to_jupyter.ipynb
[jupyter] Updating intro_to_jupyter.md
1.3.353
0.2.134
[jupyter] Updating intro_to_jupyter.qmd
[jupyter] 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 notebooks open side-by-side. The left notebook, 'intro_to_jupyter.ipynb', is in 'Simple' mode and displays a text cell with a pink highlight. The right notebook, 'intro_to_jupyter.R', is in 'Code' mode and shows R code with a green highlight. The status bar at the bottom indicates the current mode is 'Code' and the cursor is at line 1, column 1 of the R notebook.

```
intro_to_jupyter.ipynb
```

ied, a 'scatter' trace seems appropriate.
Read more about this trace type -> <http://plotly.com/r/reference/#scatter>

This is a change

```
intro_to_jupyter.R
```

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

Simple 1 \$ 1 R | ... Mode: Com... Ln 1, C... intro_to_jupyter.i...

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 `jupyter`) 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-compatible).
- 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/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: jupyter text)
3. not great for non-interactive environments (soln: jupyter text)
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?