

Publishing Jupyter Notebooks with Quarto

J.J. Allaire — Data Council 2023

“We argue that even though Jupyter helps users perform complex, technical work, Jupyter itself solves problems that are fundamentally human in nature. Namely, **Jupyter helps humans to think and tell stories with code and data.** We illustrate this by describing three dimensions of Jupyter: 1) interactive computing; 2) computational narratives; and 3) the idea that Jupyter is more than software.”

Brian Granger and Fernando Perez
Computing in Science & Engineering
March-April 2021, pp. 7-14, vol. 23
DOI Bookmark: [10.1109/MCSE.2021.3059263](https://doi.org/10.1109/MCSE.2021.3059263)

THEME ARTICLE: JUPYTER IN COMPUTATIONAL SCIENCE

Jupyter: Thinking and Storytelling With Code and Data

Brian E. Granger^a, Amazon Web Services and California Polytechnic State University, San Luis Obispo 93401, USA
Fernando Pérez, UC Berkeley and Lawrence Berkeley National Laboratory, Berkeley, CA, 94720, USA

Project Jupyter is an open-source project for interactive computing widely used in data science, machine learning, and scientific computing. We argue that even though Jupyter helps users perform complex, technical work, Jupyter itself solves problems that are fundamentally human in nature. Namely, Jupyter helps humans to think and tell stories with code and data. We illustrate this by describing three dimensions of Jupyter: 1) interactive computing; 2) computational narratives; and 3) the idea that Jupyter is more than software. We illustrate the impact of these dimensions on a community of practice in earth and climate science.

Project Jupyter^a is an open-source software project and community that builds software, services, and open standards for interactive computing across dozens of programming languages. The core of Jupyter is the Jupyter Notebook,¹ an open document format and web application that enables users to compose and share interactive programs that combine live code with narrative text, equations, interactive visualizations, images, and more. Jupyter was spawned from its parent project, IPython, in 2014 as usage of the Notebook grew outward from its origins

- 1) JupyterLab:^c the project's next-generation extensible notebook user interface.
- 2) nbconvert:^d for converting notebook formats.
- 3) Jupyter Widgets:^e for building interactive interfaces in notebooks
- 4) Voilà:^f for turning notebooks into static and web applications.
- 5) JupyterHub:^g for deploying Jupyter environments

Telling the Whole Story

- Sources, assumptions, and constraints are often as important to understand as our metrics and visualizations
- The insights we glean from data are often contextual and have important qualifications
- Narrative becomes a crucial part of communicating about data
- We need tools for storytelling!

<https://www.edwardtufte.com/tufte/powerpoint>

Edward R. Tufte

*The Cognitive Style of PowerPoint:
Pitching Out Corrupts Within*



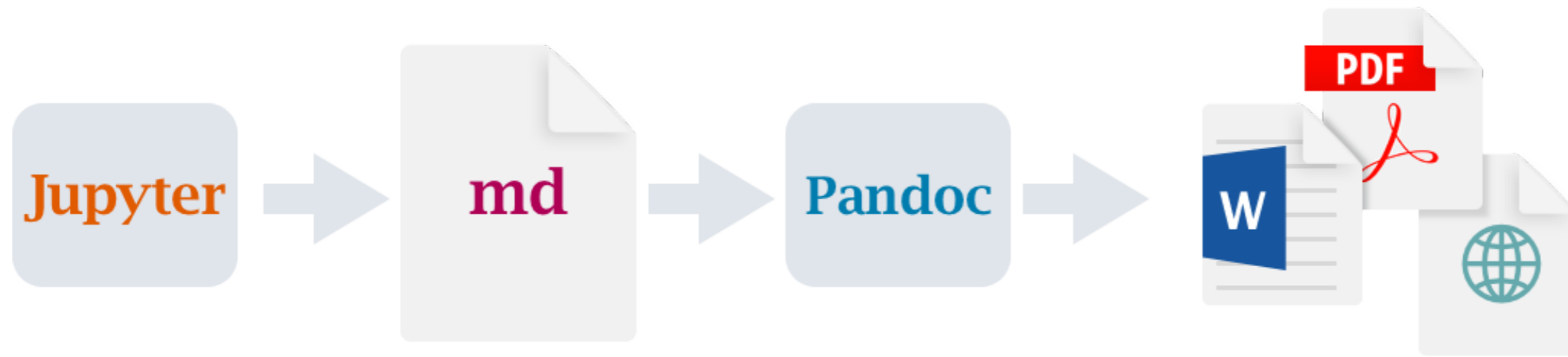
Some History

1978	TeX	Donald Knuth
1984	Literate Programming	Donald Knuth
1988	Mathematica Notebooks	Stephen Wolfram
2001	IPython	Fernando Perez
2003	Emacs org-mode	Carsten Dominik
2004	Markdown	John Gruber
2005	Sage Notebook	William Stein
2006	Pandoc	John MacFarlane
2009	GitHub Flavored Markdown	Tom Preston-Werner
2011	iPython Notebook	Fernando Perez
2012	knitr	Yihui Xie
2014	Project Jupyter	Fernando Perez

Next Up: Quarto

- J.J. Allaire—Founder/CEO of RStudio (now [Posit](#)). Worked on RStudio IDE, R Markdown, the R interface to Python (reticulate), and the R interfaces to Spark, Keras, and TensorFlow.
- Spent the last few years focused on a new project ([Quarto](#)), an open-source scientific and technical publishing system for creating computational narratives.
- 10 years of experience with [Pandoc](#) and [R Markdown](#) (a similar system that was R-specific) convinced us that the core ideas were sound
- Quarto is a ground-up re-write that is multi-language and multi-engine

What is Quarto?



- Computations: [Jupyter](#)¹
- Markdown: [Pandoc](#) w/ many enhancements
- Output: Documents, presentations, websites, books, blogs

1. [Knitr](#) and [ObservableJS](#) also supported.

Core Ideas

Core Ideas

- **Computational Narratives** Telling stories w/ code and data

Core Ideas

- **Computational Narratives** Telling stories w/ code and data
- **Technical Communications** Code, math, diagrams, etc.

Core Ideas

- **Computational Narratives** Telling stories w/ code and data
- **Technical Communications** Code, math, diagrams, etc.
- **Semantic Authoring** Write once, publish everywhere

Core Ideas

- **Computational Narratives** Telling stories w/ code and data
- **Technical Communications** Code, math, diagrams, etc.
- **Semantic Authoring** Write once, publish everywhere
- **Many Uses of Notebooks** The coin of the realm

Core Ideas

- **Computational Narratives** Telling stories w/ code and data
- **Technical Communications** Code, math, diagrams, etc.
- **Semantic Authoring** Write once, publish everywhere
- **Many Uses of Notebooks** The coin of the realm
- **Under the Hood** How to hack and extend the system



Computational Narratives

Tools for Computational Narratives

Core requirements:

- Render executable content from Jupyter
- Include code, math, diagrams, citations, crossrefs, etc.
- Semantic authoring with markdown
- Extensible output formats

Tools for Computational Narratives

Core requirements:

- Render executable content from Jupyter
- Include code, math, diagrams, citations, crossrefs, etc.
- Semantic authoring with markdown
- Extensible output formats

Variety of tools available: [nbconvert](#), [Jupyter Book](#), [Myst-JS](#), [Quarto](#)

Tools for Computational Narratives

Core requirements:

- Render executable content from Jupyter
- Include code, math, diagrams, citations, crossrefs, etc.
- Semantic authoring with markdown
- Extensible output formats

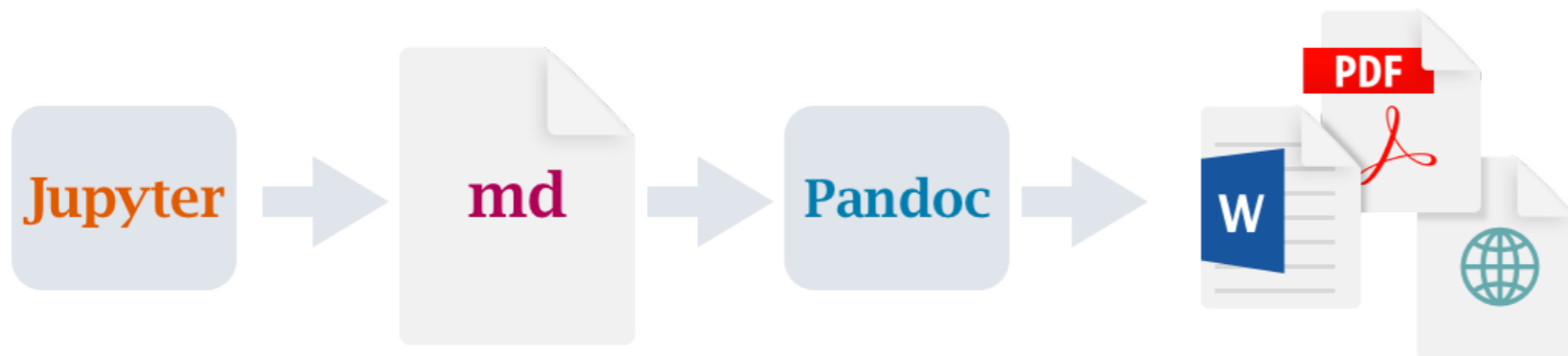
Variety of tools available: [nbconvert](#), [Jupyter Book](#), [Myst-JS](#), [Quarto](#)

Will talk about tools for computational narratives through the lens of Quarto, but these tools share many features and are all worth evaluating.

How Does Quarto Work?

Quarto *renders* notebooks into various output formats:

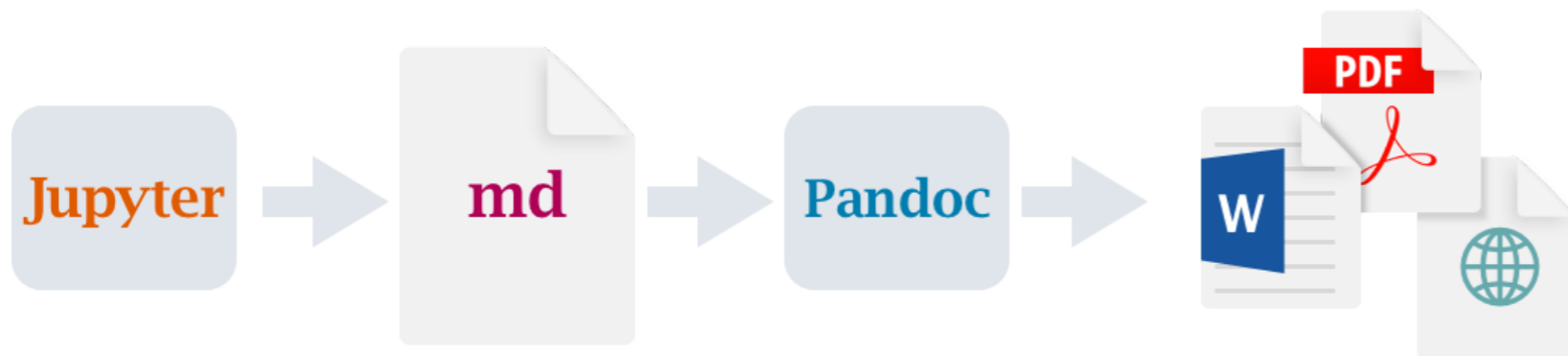
```
$ quarto render notebook.ipynb
```



How Does Quarto Work?

Quarto *renders* notebooks into various output formats:

```
$ quarto render notebook.ipynb
```

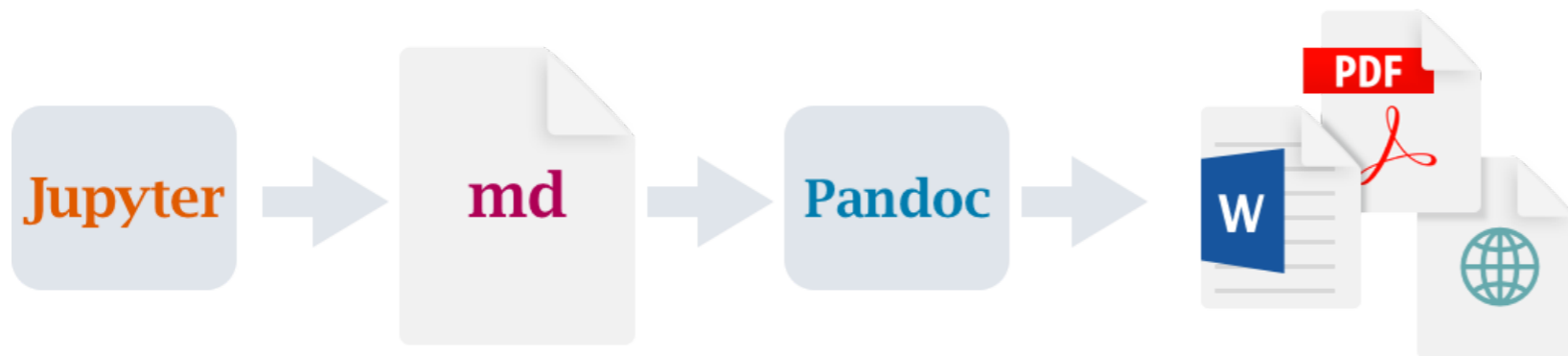


- Start with a Jupyter notebook (executed or not)

How Does Quarto Work?

Quarto *renders* notebooks into various output formats:

```
$ quarto render notebook.ipynb
```

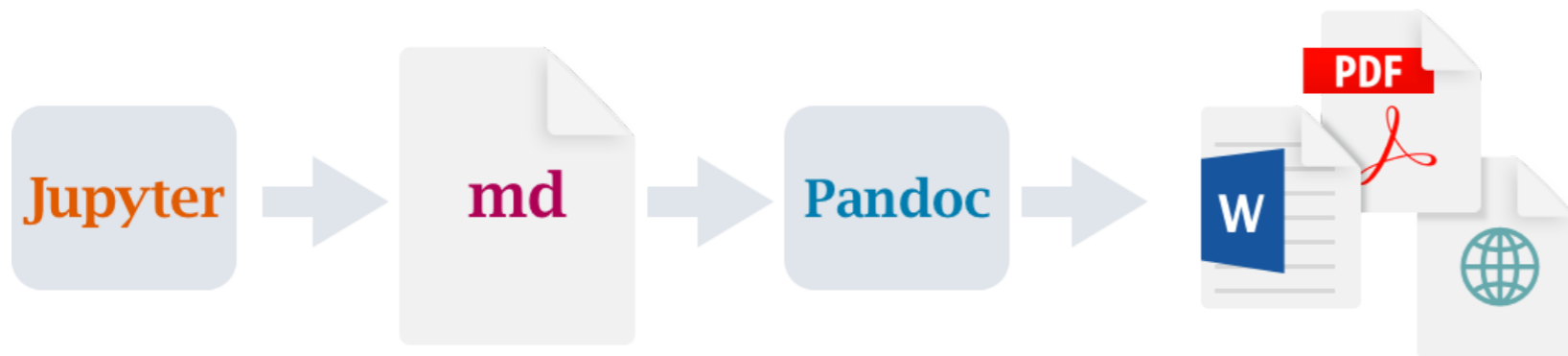


- Start with a Jupyter notebook (executed or not)
- Add document and cell level output options using YAML

How Does Quarto Work?

Quarto *renders* notebooks into various output formats:

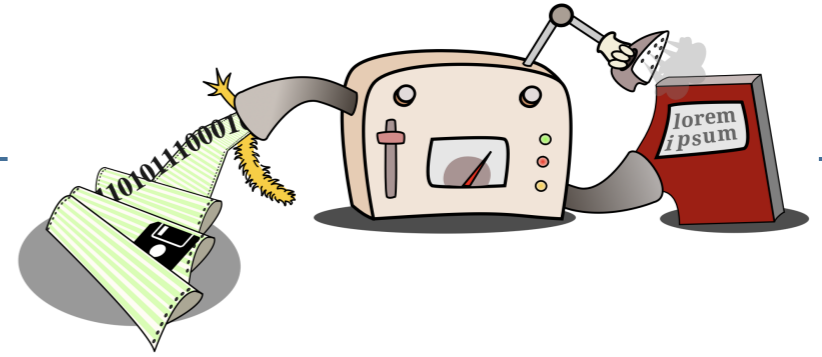
```
$ quarto render notebook.ipynb
```



- Start with a Jupyter notebook (executed or not)
- Add document and cell level output options using YAML
- `quarto render` to the desired output format

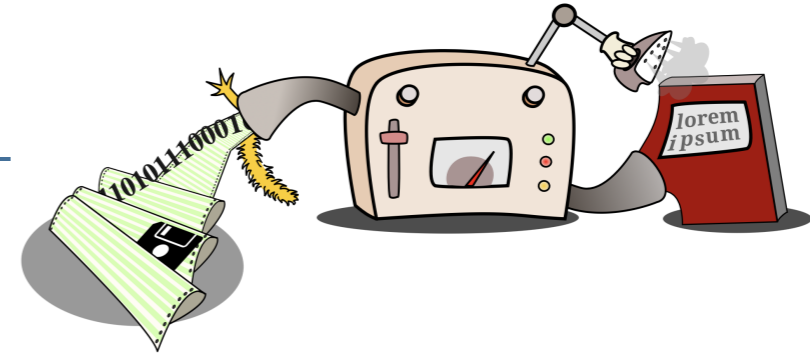
Rendering with Pandoc

Universal document converter



Rendering with Pandoc

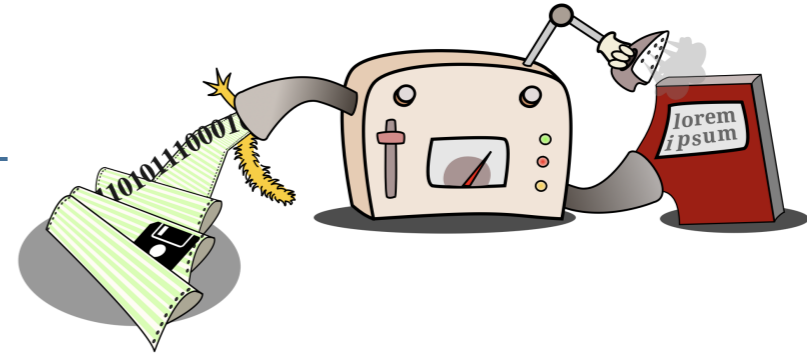
Universal document converter



- Created in 2006 by John MacFarlane (who is also the author of the CommonMark spec and CommonMark reference implementations in JavaScript, C, and Haskell)

Rendering with Pandoc

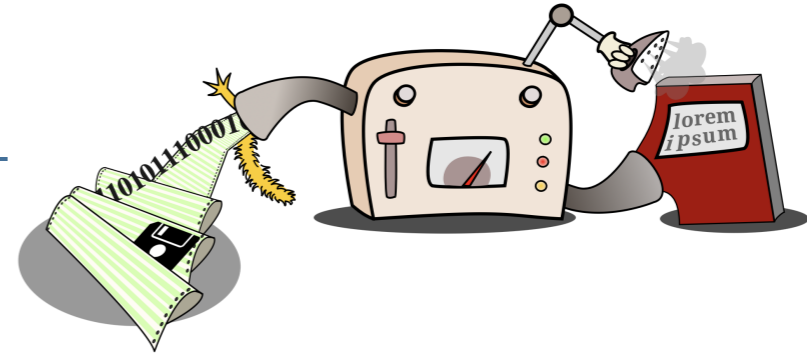
Universal document converter



- Created in 2006 by John MacFarlane (who is also the author of the CommonMark spec and CommonMark reference implementations in JavaScript, C, and Haskell)
- CommonMark + many extensions for technical writing

Rendering with Pandoc

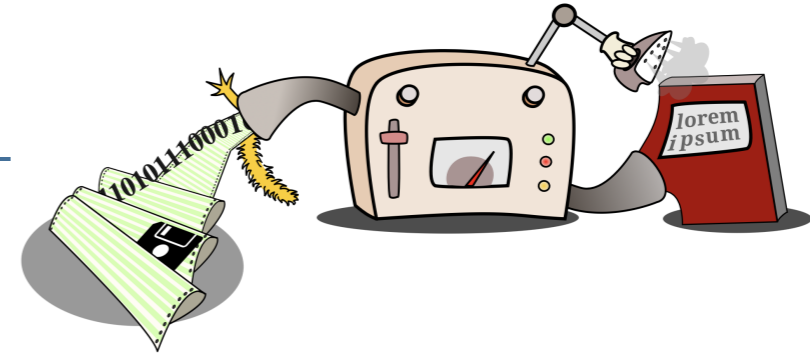
Universal document converter



- Created in 2006 by John MacFarlane (who is also the author of the CommonMark spec and CommonMark reference implementations in JavaScript, C, and Haskell)
- CommonMark + many extensions for technical writing
- Supports dozens of output formats (just about any format you can name)

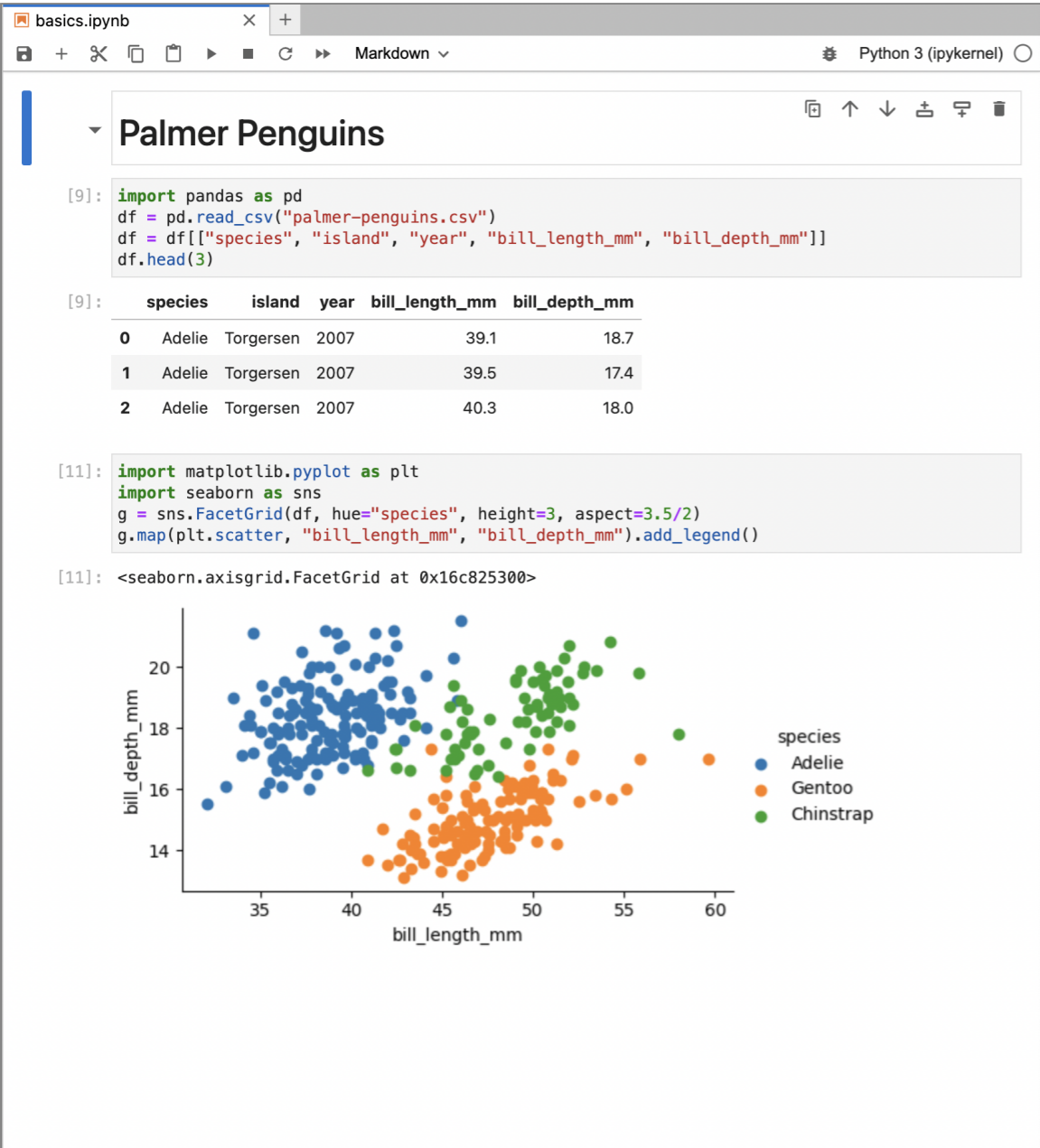
Rendering with Pandoc

Universal document converter

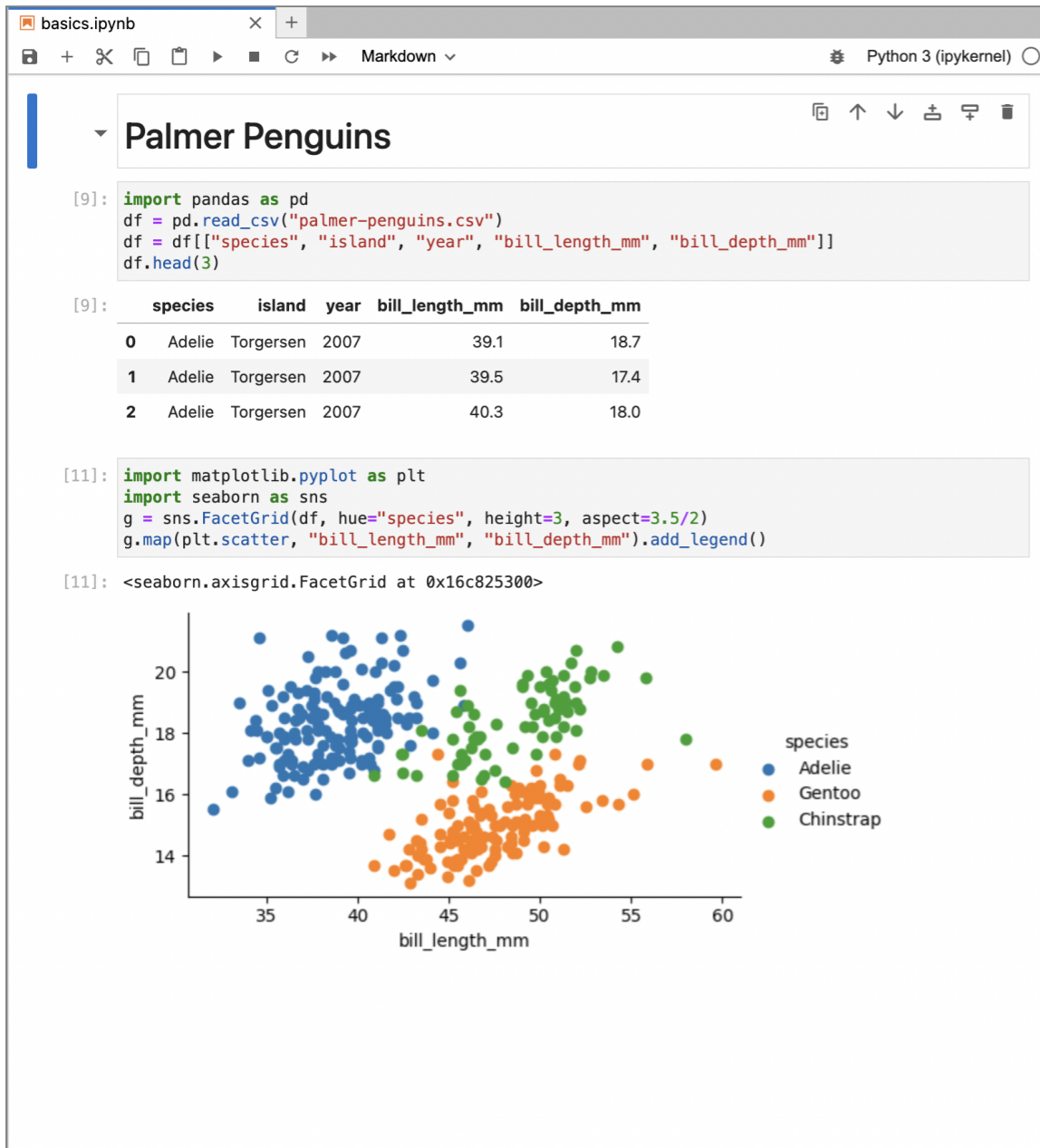


- Created in 2006 by John MacFarlane (who is also the author of the CommonMark spec and CommonMark reference implementations in JavaScript, C, and Haskell)
- CommonMark + many extensions for technical writing
- Supports dozens of output formats (just about any format you can name)
- Highly extensible (custom readers, custom writers, AST filters)

Render Notebook to HTML (default options)



Render Notebook to HTML (default options)

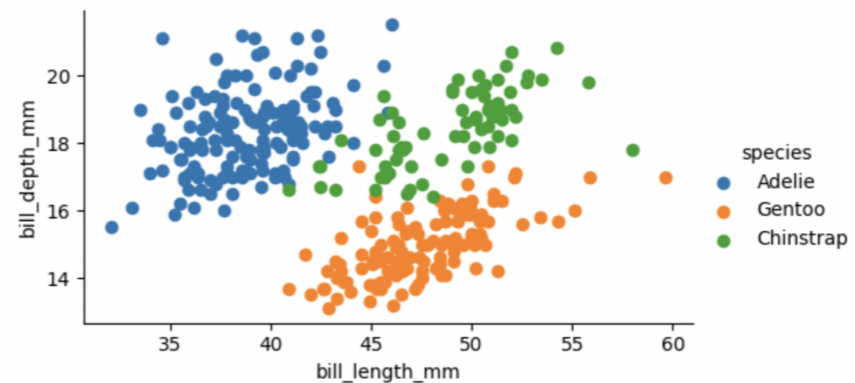


Palmer Penguins

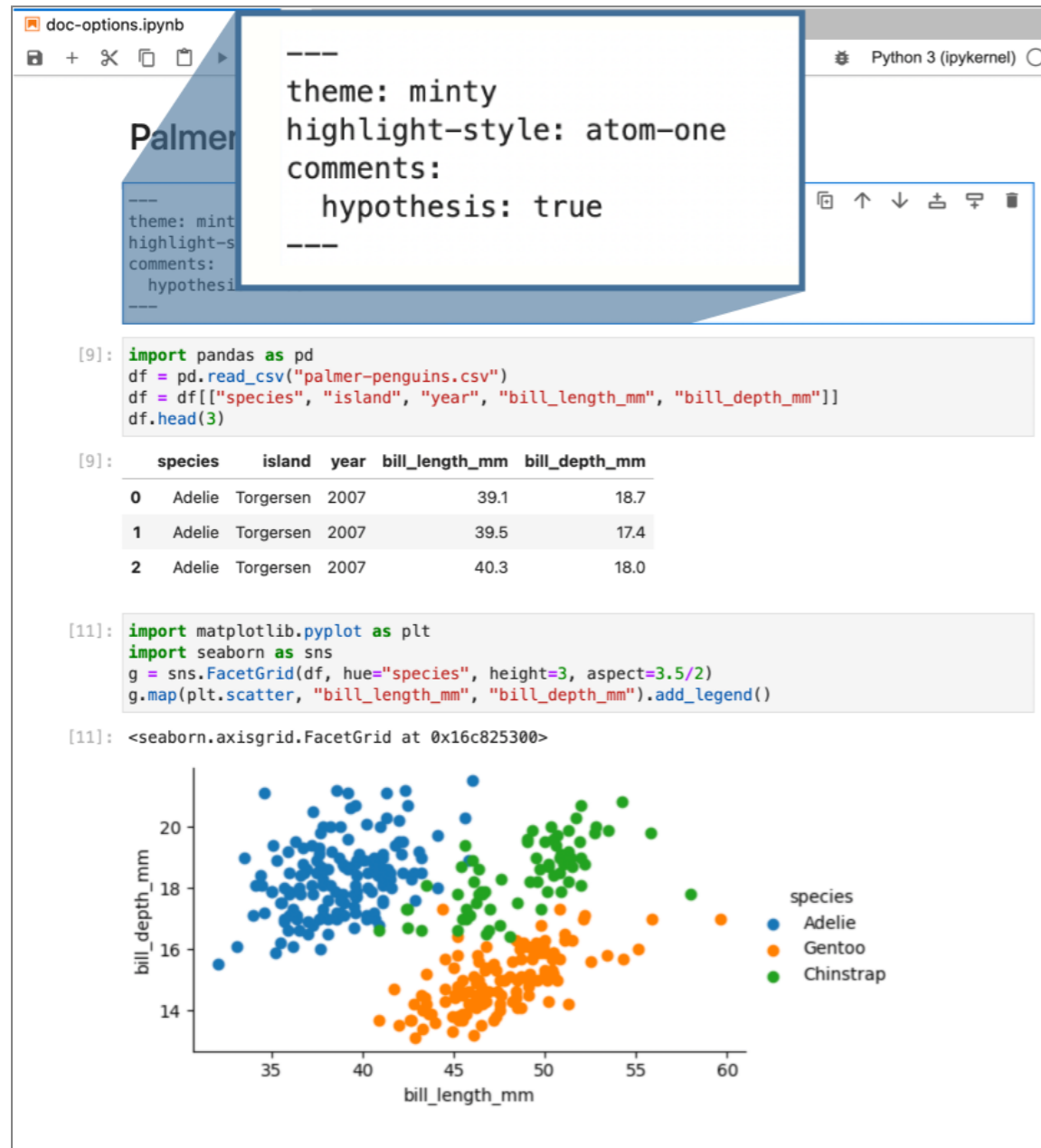
```
import pandas as pd
df = pd.read_csv("palmer-penguins.csv")
df = df[["species", "island", "year", "bill_length_mm", "bill_depth_mm"]]
df.head(3)
```

	species	island	year	bill_length_mm	bill_depth_mm
0	Adelie	Torgersen	2007	39.1	18.7
1	Adelie	Torgersen	2007	39.5	17.4
2	Adelie	Torgersen	2007	40.3	18.0

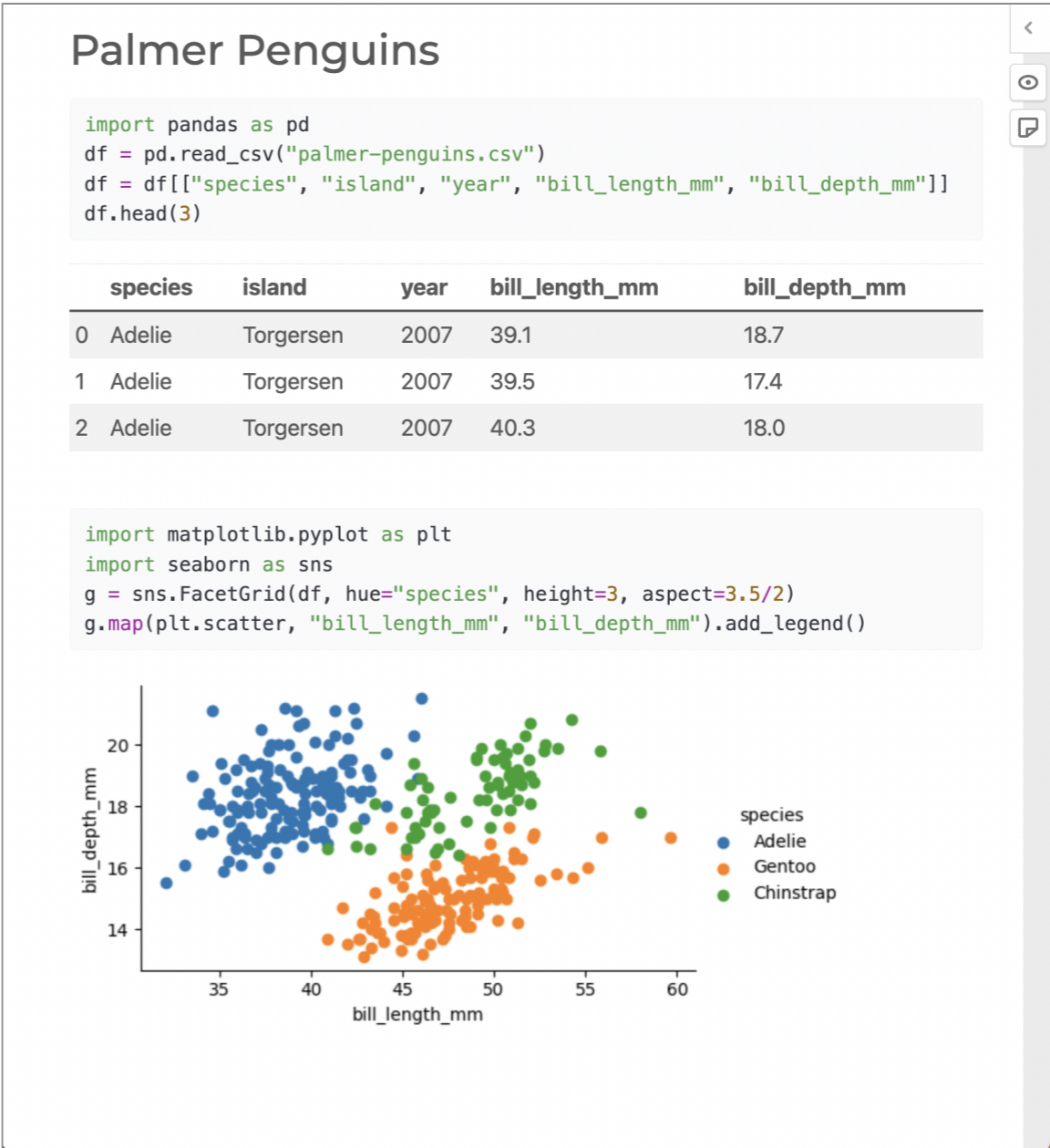
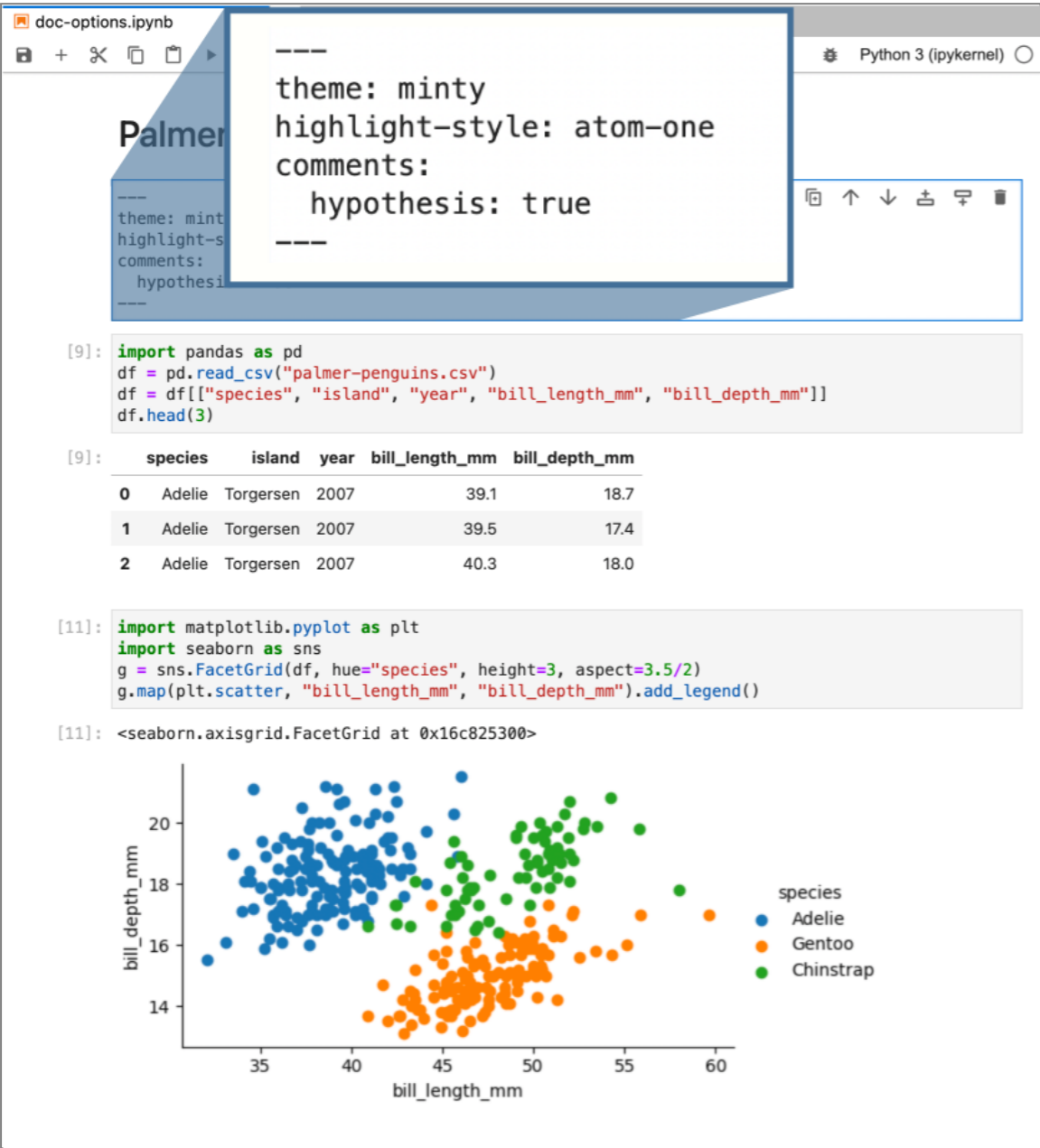
```
import matplotlib.pyplot as plt
import seaborn as sns
g = sns.FacetGrid(df, hue="species", height=3, aspect=3.5/2)
g.map(plt.scatter, "bill_length_mm", "bill_depth_mm").add_legend()
```



Render Notebook to HTML (document level options)



Render Notebook to HTML (document level options)



Render Notebook to HTML (document and cell level options)

cell-options.ipynb

Python 3 (ipykernel)

Palmer

```
author: "Norah Jones"
date: "March 12, 2023"
code-tools: true
code-fold: true
```

```
[1]: #| echo: false
import pandas as pd
df = pd.read_csv("palmer-penguins.csv")
df = df[["species", "island", "year", "bill_length_mm", "bill_depth_mm"]]
```

Exploring

See @fig-bill-s

```
[2]: #| label: fig-bill-sizes
#| fig-cap: "Bill Sizes by Species"
import matplotlib.pyplot as plt
import seaborn as sns
g = sns.FacetGrid(df, hue="species", height=3, aspect=3.5/2)
g.map(plt.scatter, "bill_length_mm", "bill_depth_mm").add_legend()
```

```
[2]: <seaborn.axisgrid.FacetGrid at 0x10612e080>
```



bill_depth_mm

bill_length_mm

species

- Adelie
- Gentoo
- Chinstrap

Render Notebook to HTML (document and cell level options)

cell-options.ipynb

Python 3 (ipykernel)

Palmer

author: "Norah Jones"
date: "March 12, 2023"
code-tools: true
code-fold: true

```
[1]: #| echo: false
import pandas as pd
df = pd.read_csv("palmer-penguins.csv")
df = df[["species", "island", "year", "bill_length_mm", "bill_depth_mm"]]
```

Exploring

See @fig-bill-s

#| label: fig-bill-sizes
#| fig-cap: "Bill Sizes by Species"

```
[2]: #| label: fig-bill-sizes
#| fig-cap: "Bill Sizes by Species"
import matplotlib.pyplot as plt
import seaborn as sns
g = sns.FacetGrid(df, hue="species", height=3, aspect=3.5/2)
g.map(plt.scatter, "bill_length_mm", "bill_depth_mm").add_legend()
```

```
[2]: <seaborn.axisgrid.FacetGrid at 0x10612e080>
```



Palmer Penguins

</> Code

AUTHOR
Norah Jones

PUBLISHED
March 12, 2023

Exploring the Data

See [Figure 1](#) for an exploration of bill sizes by species.

► Code



Figure 1: Bill Sizes by Species

Render Notebook to HTML (document and cell level options)

cell-options.ipynb

Python 3 (ipykernel)

author: "Norah Jones"
date: "March 12, 2023"
code-tools: true
code-fold: true

Palmer

author: "No
date: "Marc
code-tools:
code-fold:

[1]:

```
#| echo: false
import pandas as pd
df = pd.read_csv("palmer-penguins.csv")
df = df[["species", "island", "year", "bill_length_mm", "bill_depth_mm"]]
```

Exploring

See @fig-bill-s

#| label: fig-bill-sizes
#| fig-cap: "Bill Sizes by Species"

import matplotlib.pyplot as plt
import seaborn as sns

[2]:

```
#| label: fig-bill-sizes
#| fig-cap: "Bill Sizes by Species"
import matplotlib.pyplot as plt
import seaborn as sns
g = sns.FacetGrid(df, hue="species", height=3, aspect=3.5/2)
g.map(plt.scatter, "bill_length_mm", "bill_depth_mm").add_legend()
```

[2]:

<seaborn.axisgrid.FacetGrid at 0x10612e080>



Palmer Penguins

AUTHOR
Norah Jones

PUBLISHED
March 12, 2023

Show All Code

Hide All Code

Exploring the Data

See [Figure 1](#) for an exploration of bill sizes by species.

▼ Code

import matplotlib.pyplot as plt
import seaborn as sns
g = sns.FacetGrid(df, hue="species", height=3, aspect=3.5/2)
g.map(plt.scatter, "bill_length_mm", "bill_depth_mm").add_legend()



Figure 1: Bill Sizes by Species

javascript:void(0)

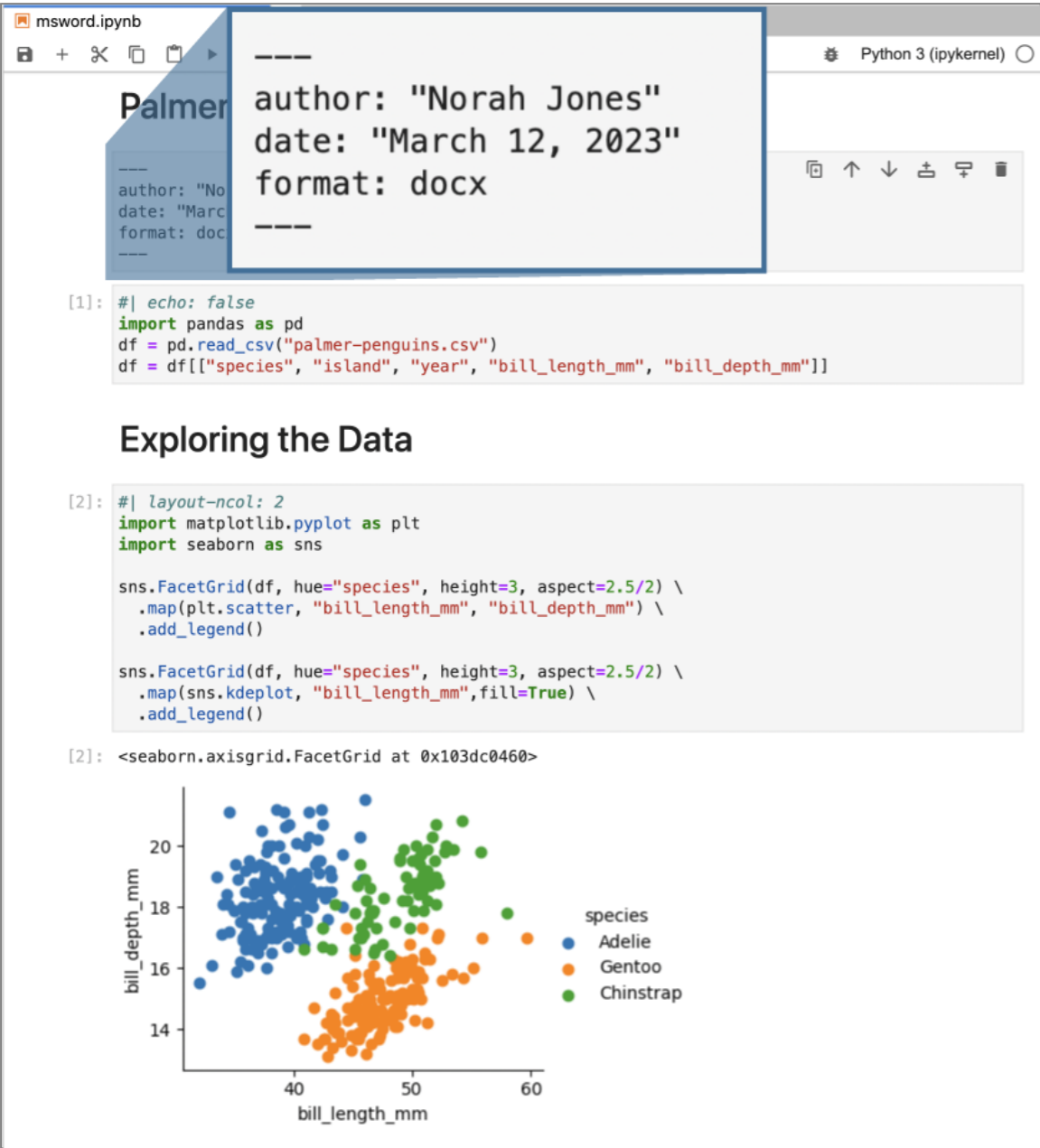
Production Quality Output

You have no doubt seen this sort of conversion before (e.g. [nbconvert](#))

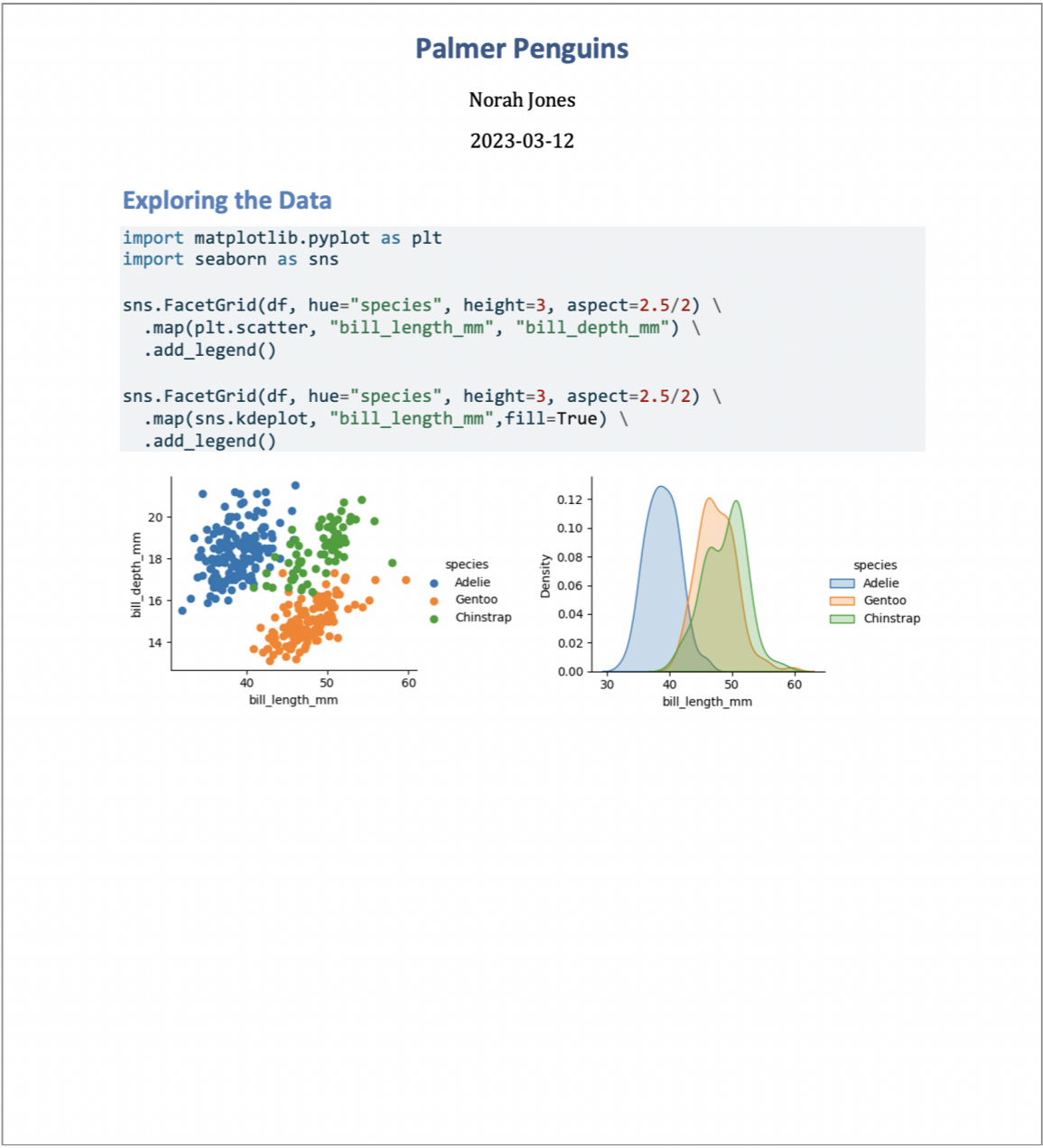
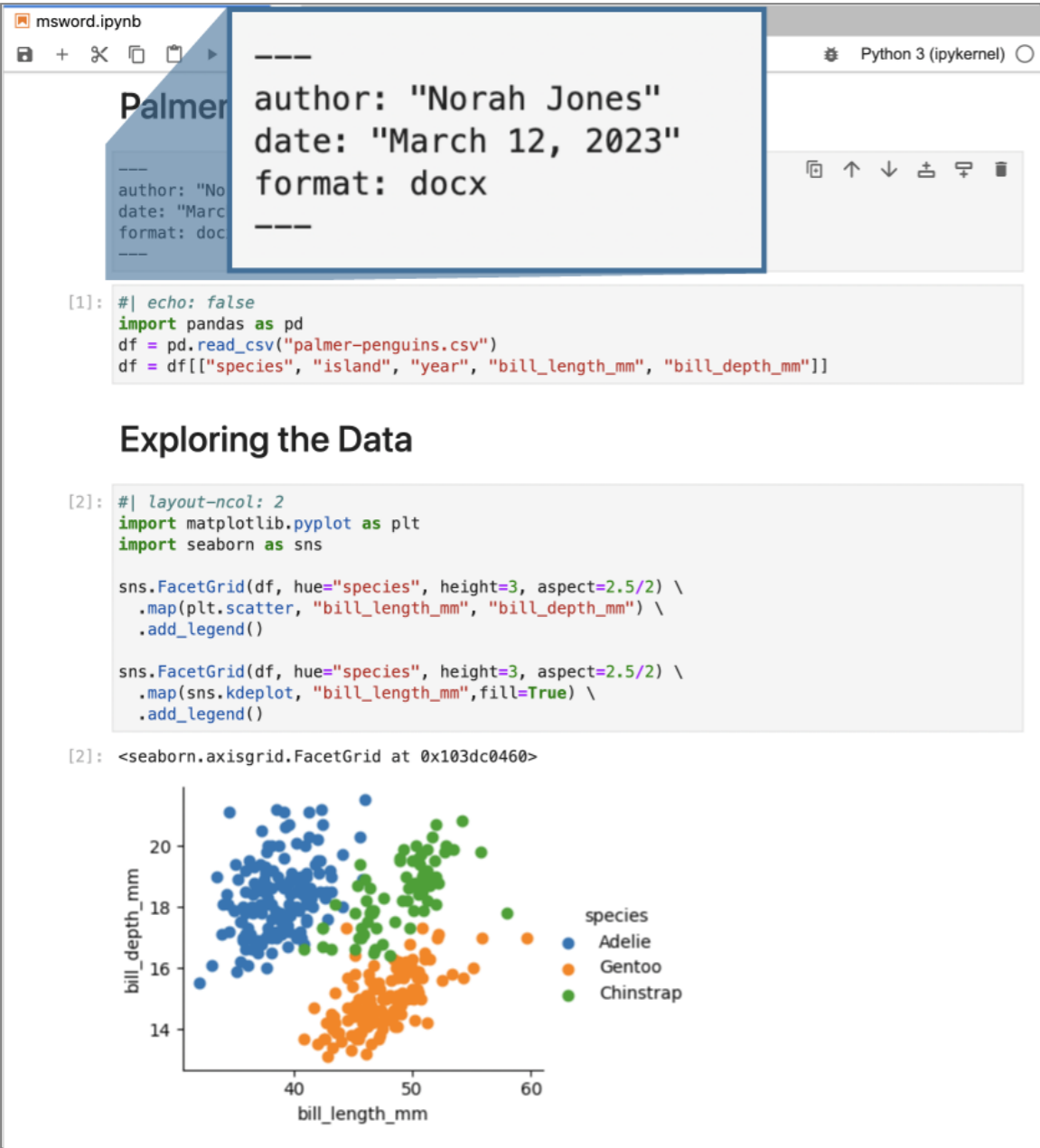
Quarto goes well beyond simple conversion—our goal is to help you produce professional, production-quality output!

- Office Documents
- Technical Manuscripts
- Websites / Blogs
- Multi-Format Books
- Presentations
- And beyond...

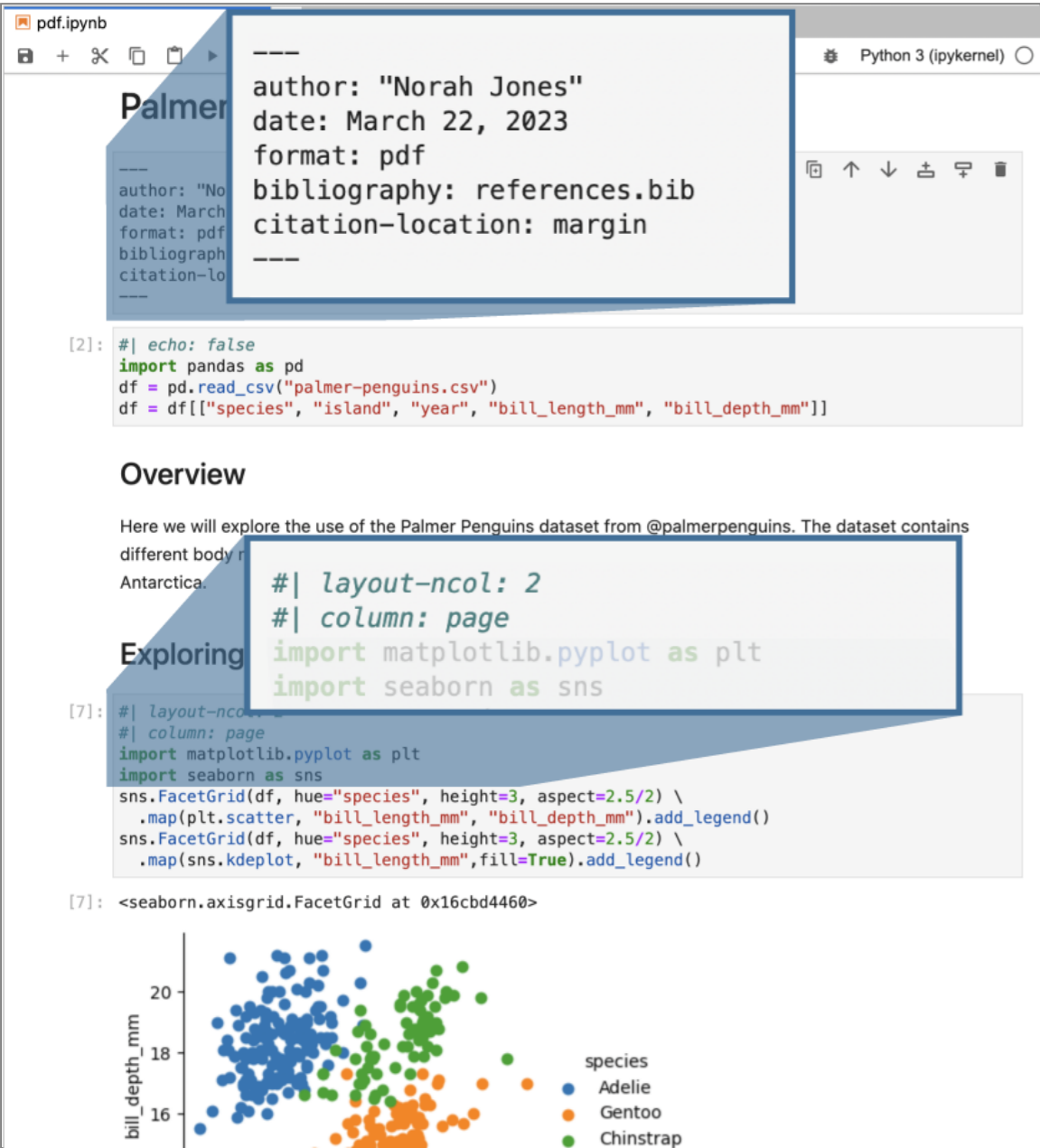
Render Notebook to MS Word



Render Notebook to MS Word



Render Notebook to PDF



Render Notebook to PDF

pdf.ipynb

Python 3 (ipykernel)

author: "Norah Jones"
date: March 22, 2023
format: pdf
bibliography: references.bib
citation-location: margin

```
[2]: #| echo: false
import pandas as pd
df = pd.read_csv("palmer-penguins.csv")
df = df[["species", "island", "year", "bill_length_mm", "bill_depth_mm"]]
```

Overview

Here we will explore the use of the Palmer Penguins dataset from @palmerpenguins. The dataset contains different body measurements for three species of penguins from three islands in the Palmer Archipelago, Antarctica.

Exploring the Data

```
[7]: #| layout-ncol: 2
      #| column: page
import matplotlib.pyplot as plt
import seaborn as sns

sns.FacetGrid(df, hue="species", height=3, aspect=2.5/2) \
    .map(plt.scatter, "bill_length_mm", "bill_depth_mm").add_legend()
sns.FacetGrid(df, hue="species", height=3, aspect=2.5/2) \
    .map(sns.kdeplot, "bill_length_mm", fill=True).add_legend()
```

<seaborn.axisgrid.FacetGrid at 0x16cbd4460>

Palmer Penguins

Norah Jones
2023-03-22

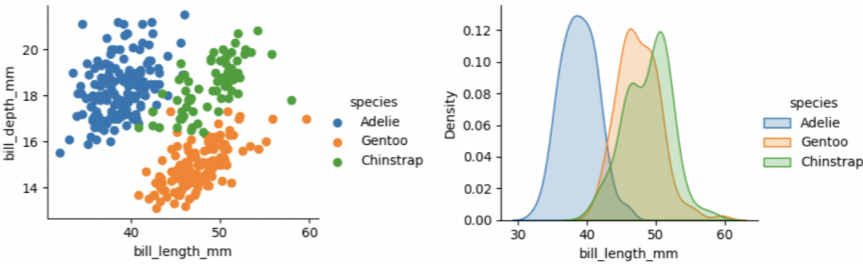
Overview

Here we will explore the use of the Palmer Penguins dataset from Horst, Hill, and Gorman (2020). The dataset contains different body measurements for three species of penguins from three islands in the Palmer Archipelago, Antarctica.

Horst, Allison Marie, Alison Presmanes Hill, and Kristen B Gorman. 2020. *Palmerpenguins: Palmer Archipelago (Antarctica) Penguin Data*. <https://doi.org/10.5281/zenodo.3960218>.

Exploring the Data

```
import matplotlib.pyplot as plt
import seaborn as sns
sns.FacetGrid(df, hue="species", height=3, aspect=2.5/2) \
    .map(plt.scatter, "bill_length_mm", "bill_depth_mm").add_legend()
sns.FacetGrid(df, hue="species", height=3, aspect=2.5/2) \
    .map(sns.kdeplot, "bill_length_mm", fill=True).add_legend()
```



Render Notebook to Revealjs

presentation.ipynb Python 3 (ipykernel)

Palmer

```
author: "Norah Jones"
date: "March 12, 2023"
format:
  revealjs:
    slide-number: c/t
    logo: palmer.png
```

```
[2]: #| echo: false
import pandas as pd
df = pd.read_csv("palmer-penguins.csv")
df = df[["species", "island", "year", "bill_length_mm", "bill_depth_mm"]]
```

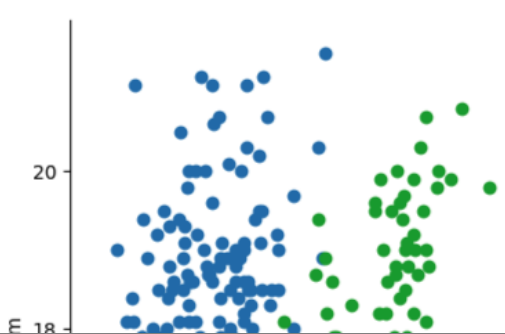
Exploring the Data

```
[7]: #| layout-ncol: 2
import matplotlib.pyplot as plt
import seaborn as sns

sns.FacetGrid(df, hue="species", height=6, aspect=3/4) \
    .map(plt.scatter, "bill_length_mm", "bill_depth_mm") \
    .add_legend()

sns.FacetGrid(df, hue="species", height=6, aspect=3/4) \
    .map(sns.kdeplot, "bill_length_mm", fill=True) \
    .add_legend()
```

```
[7]: <seaborn.axisgrid.FacetGrid at 0x137f56b60>
```



Render Notebook to Revealjs

presentation.ipynb

Python 3 (ipykernel)

↶ ↷ ↵ ↶ ↷ ↵

Palmer

author: "Norah Jones"
date: "March 12, 2023"
format:
 revealjs:
 slide-number: c/t
 logo: palmer.png

[2]:

```
#| echo: false
import pandas as pd
df = pd.read_csv("palmer-penguins.csv")
df = df[["species", "island", "year", "bill_length_mm", "bill_depth_mm"]]
```

Exploring the Data

[7]:

```
#| layout-ncol: 2
import matplotlib.pyplot as plt
import seaborn as sns

sns.FacetGrid(df, hue="species", height=6, aspect=3/4) \
    .map(plt.scatter, "bill_length_mm", "bill_depth_mm") \
    .add_legend()

sns.FacetGrid(df, hue="species", height=6, aspect=3/4) \
    .map(sns.kdeplot, "bill_length_mm", fill=True) \
    .add_legend()
```

[7]:

<seaborn.axisgrid.FacetGrid at 0x137f56b60>

2 / 3

Exploring the Data

bill_depth_mm

bill_length_mm

species

- Adelia
- Gentoo
- Chinstrap

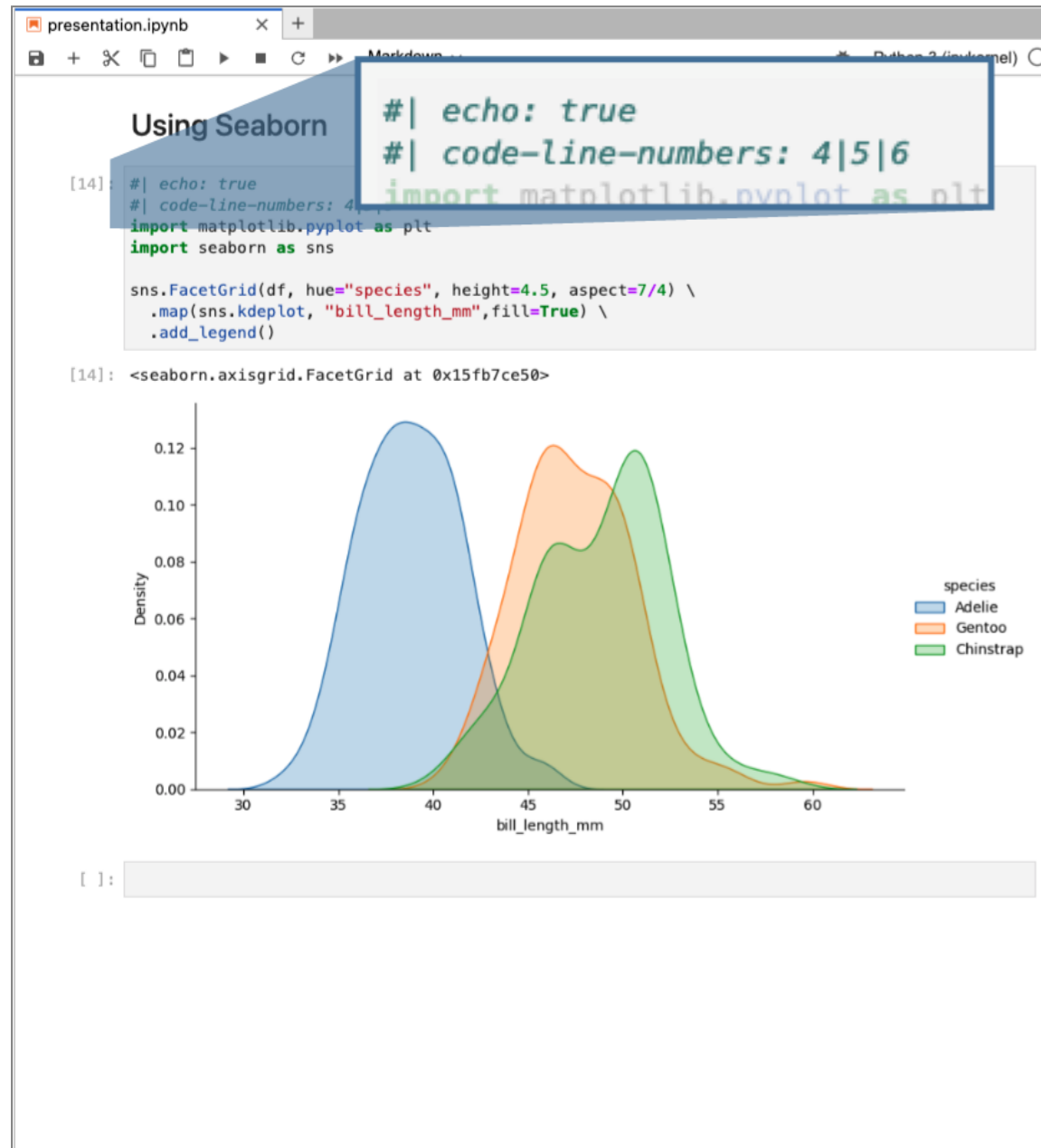
Density

bill_length_mm

species

- Adelia
- Gentoo
- Chinstrap

Render Notebook to Revealjs (show code with line by line highlighting)



Render Notebook to Revealjs (show code with line by line highlighting)

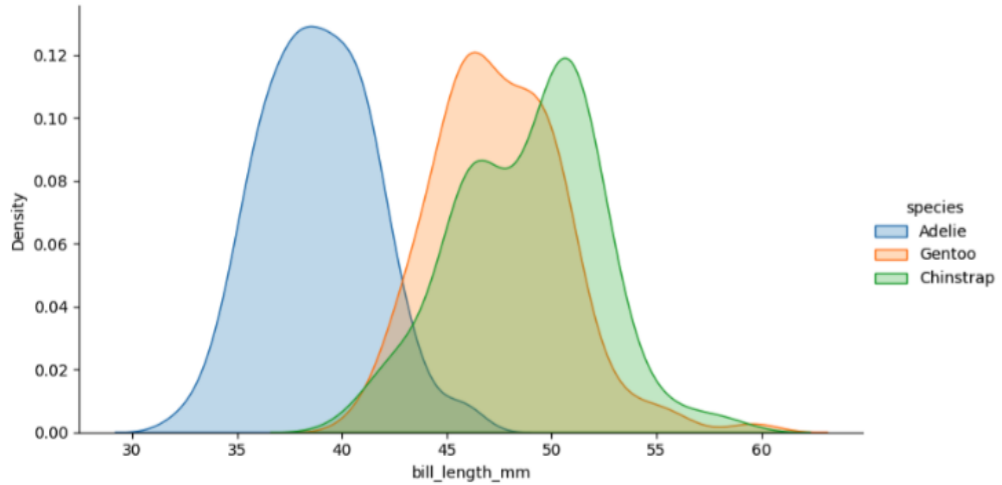
presentation.ipynb

Using Seaborn

```
[14]: #| echo: true
      #| code-line-numbers: 4,5,6
      import matplotlib.pyplot as plt
      import seaborn as sns

      sns.FacetGrid(df, hue="species", height=4.5, aspect=7/4) \
        .map(sns.kdeplot, "bill_length_mm", fill=True) \
        .add_legend()

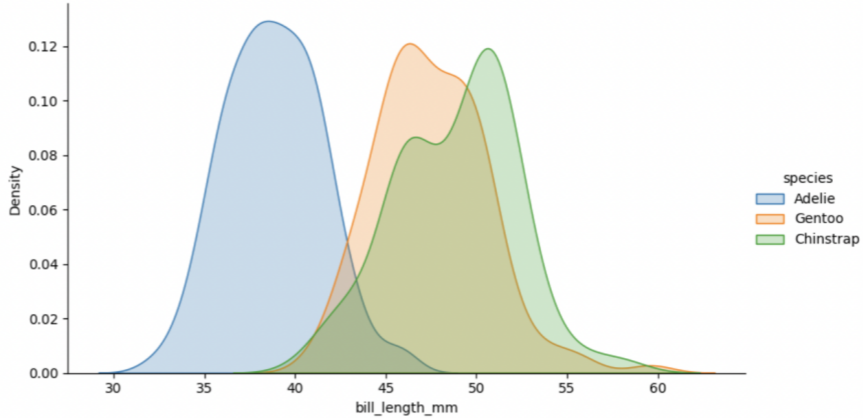
[14]: <seaborn.axisgrid.FacetGrid at 0x15fb7ce50>
```




```
[ ]:
```

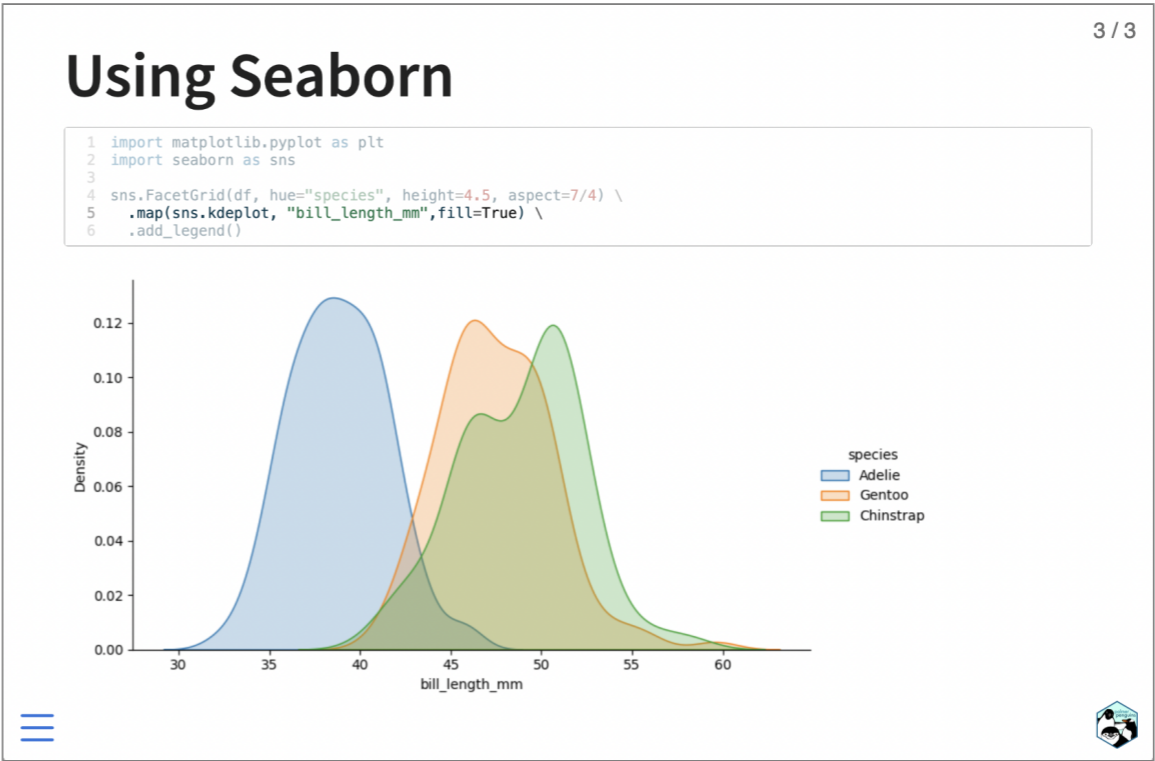
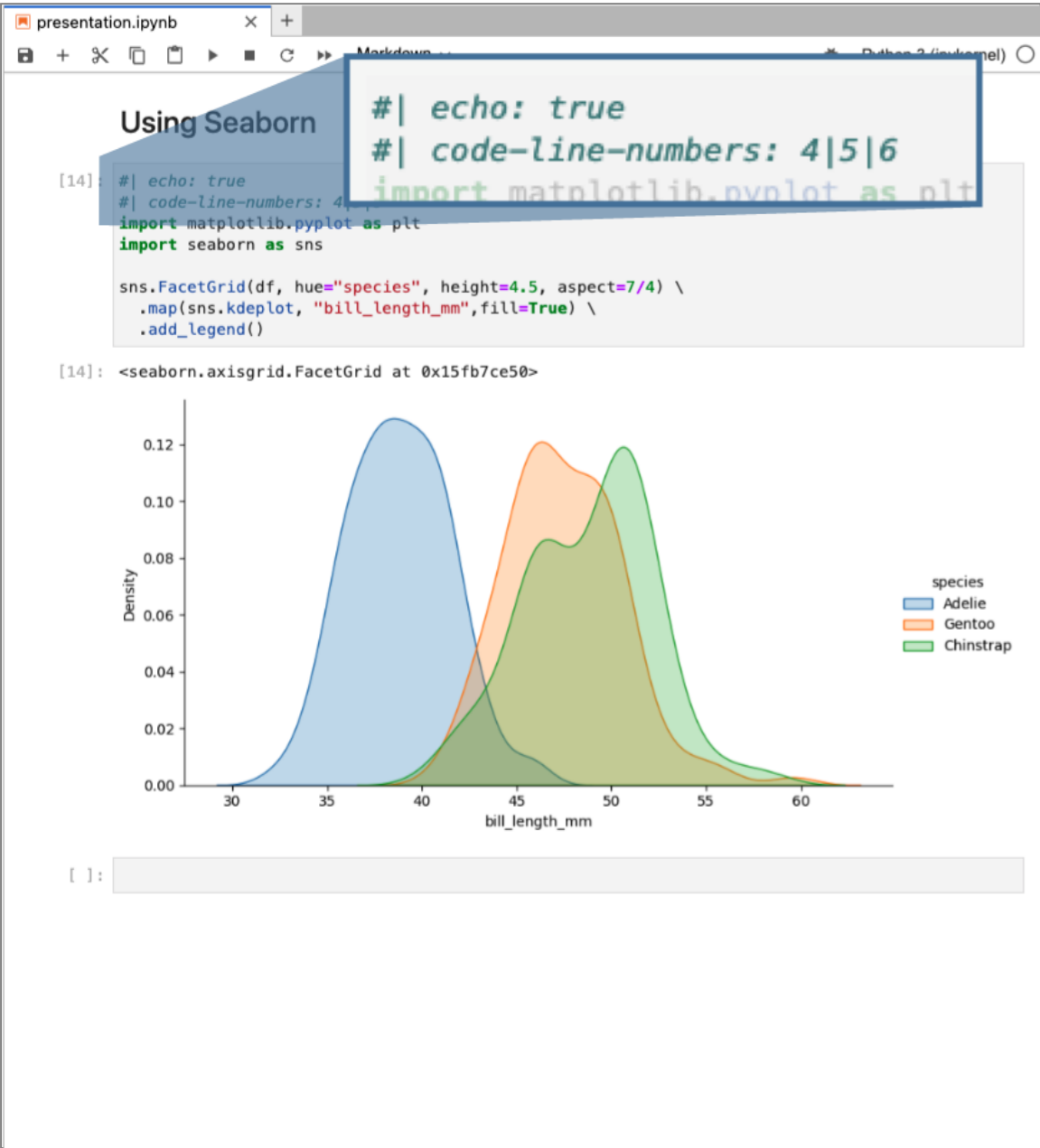
Using Seaborn

```
1 import matplotlib.pyplot as plt
2 import seaborn as sns
3
4 sns.FacetGrid(df, hue="species", height=4.5, aspect=7/4) \
5   .map(sns.kdeplot, "bill_length_mm", fill=True) \
6   .add_legend()
```





Render Notebook to Revealjs (show code with line by line highlighting)



Quarto Projects

```
_quarto.yml
1 project:
2   type: website
3
4 website:
5   title: "Acme"
6   navbar:
7     left:
8       - href: index.qmd
9         text: Home
10      - about.qmd
11
12 format:
13   html:
14     theme: cosmo
15     css: styles.css
```

- So far our examples have been single documents or presentations
- Quarto has a project system that enables you to produce collections of documents in various formats (websites, blogs, books, etc.)
- `_quarto.yml` config file defines the behavior of projects

Quarto Projects

_quarto.yml

```
1 project:
2   type: website
3
4 website:
5   title: "Acme"
6   navbar:
7     left:
8       - href: index.qmd
9         text: Home
10      - about.qmd
11
12 format:
13   html:
14     theme: cosmo
15     css: styles.css
```

- So far our examples have been single documents or presentations
- Quarto has a project system that enables you to produce collections of documents in various formats (websites, blogs, books, etc.)
- `_quarto.yml` config file defines the behavior of projects

Quarto Projects

_quarto.yml

```
1 project:
2   type: website
3
4 website:
5   title: "Acme"
6   navbar:
7     left:
8       - href: index.qmd
9         text: Home
10      - about.qmd
11
12 format:
13   html:
14     theme: cosmo
15     css: styles.css
```

- So far our examples have been single documents or presentations
- Quarto has a project system that enables you to produce collections of documents in various formats (websites, blogs, books, etc.)
- `_quarto.yml` config file defines the behavior of projects

Quarto Projects

_quarto.yml

```
1 project:
2   type: website
3
4 website:
5   title: "Acme"
6   navbar:
7     left:
8       - href: index.qmd
9         text: Home
10      - about.qmd
11
12 format:
13   html:
14     theme: cosmo
15     css: styles.css
```

- So far our examples have been single documents or presentations
- Quarto has a project system that enables you to produce collections of documents in various formats (websites, blogs, books, etc.)
- `_quarto.yml` config file defines the behavior of projects

Website: Fastai Course

_quarto.yml

```
1 project:
2   type: website
3   resources:
4     - "www/*"
5
6 format:
7   html:
8     theme: cosmo
9     css: styles.css
10    toc: true
11
12 website:
13   title: "Practical Deep Learning for Coders"
14   description: "Learn Deep Learning with fastai and PyTorch"
15   twitter-card: true
16   open-graph: true
17   reader-mode: true
18   page-navigation: true
19   repo-branch: master
20   repo-url: https://github.com/fastai/course22
21   repo-actions: [issue]
22   navbar:
23     search: true
24     right:
25       - icon: github
26         href: https://github.com/fastai/course22
27   sidebar:
28     style: "floating"
29
30 metadata-files:
31   - sidebar.yml
```

Website: Fastai Course

```
_quarto.yml
1 project:
2   type: website
3   resources:
4     - "www/*"
5
6 format:
7   html:
8     theme: cosmo
9     css: styles.css
10    toc: true
11
12 website:
13   title: "Practical Deep Learning for Coders"
14   description: "Learn Deep Learning with fastai and PyTorch"
15   twitter-card: true
16   open-graph: true
17   reader-mode: true
18   page-navigation: true
19   repo-branch: master
20   repo-url: https://github.com/fastai/course22
21   repo-actions: [issue]
22   navbar:
23     search: true
24     right:
25       - icon: github
26         href: https://github.com/fastai/course22
27   sidebar:
28     style: "floating"
29
30 metadata-files:
31   - sidebar.yml
```

Practical Deep Learning for Coders

Practical Deep Learning Lessons

1: Getting started

2: Deployment

3: Neural net foundations

4: Natural Language (NLP)

5: From-scratch model

6: Random forests

7: Collaborative filtering

8: Convolutions (CNNs)

9: Data ethics

Summaries

Resources

The book


Forums

Kaggle

Testimonials

Practical Deep Learning

A free course designed for people with some coding experience, who want to learn how to apply deep learning and machine learning to practical problems.




This free course is designed for people (and bunnies!) with some coding experience who want to learn how to apply deep learning and machine learning to practical problems.

Deep learning can do all kinds of amazing things. For instance, all illustrations throughout this website are made with deep learning, using [DALL-E 2](#).

Welcome!

[Practical Deep Learning for Coders 2022](#), recorded at the [University of Queensland](#), covers topics such as how to:



- Build and train deep learning models for computer vision, natural language processing, tabular analysis, and collaborative filtering problems
- Create random forests and regression models
- Deploy models
- Use PyTorch, the world's fastest growing deep learning software, plus popular libraries like fastai and Hugging Face

There are 9 lessons, and each lesson is around 90 minutes long. The course is based on our [5-star rated book](#), which is [freely available](#) online.

You don't need any special hardware or software — we'll show you how to use free resources for both building and deploying models. You don't need any university math either — we'll teach you the calculus and linear algebra you need during the course.

On this page

Welcome!

Real results

Your teacher

Is this course for me?

The software you will be using

Why deep learning?

What you will learn

How do I get started?

Report an issue

Blog: Aayush Agrawal

_quarto.yml

```
1 project:
2   type: website
3
4 website:
5   title: "Aayush Agrawal"
6   description: "Aayush's personal website"
7   repo-url: https://github.com/aayushmnit/aayushmnit.github
8   repo-actions: [edit, issue]
9   repo-branch: main
10  open-graph: true
11  google-analytics: "G-7QN8N70N41"
12  twitter-card:
13    creator: "@aayushmnit"
14    card-style: summary_large_image
15  navbar:
16    collapse-below: lg
17    left:
18      - icon: newspaper
19        href: blog.qmd
20        text: Blog
21    right:
22      - icon: github
23        href: https://github.com/aayushmnit/
24      - icon: rss
25        href: blog.xml
26
27 format:
28   html:
29     theme: sandstone
30     mainfont: Roboto
31     css: styles.css
```

Blog: Aayush Agrawal


_quarto.yml

```
1 project:
2   type: website
3
4 website:
5   title: "Aayush Agrawal"
6   description: "Aayush's personal website"
7   repo-url: https://github.com/aayushmnit/aayushmnit.github
8   repo-actions: [edit, issue]
9   repo-branch: main
10  open-graph: true
11  google-analytics: "G-7QN8N70N41"
12  twitter-card:
13    creator: "@aayushmnit"
14    card-style: summary_large_image
15  navbar:
16    collapse-below: lg
17    left:
18      - icon: newspaper
19        href: blog.qmd
20        text: Blog
21    right:
22      - icon: github
23        href: https://github.com/aayushmnit/
24      - icon: rss
25        href: blog.xml
26
27 format:
28   html:
29     theme: sandstone
30     mainfont: Roboto
31     css: styles.css
```

Aayush Agrawal

HOMEBLOGTALKSABOUTOTHER

🔍




Python OOPs fundamentals
9 min

PROGRAMMING

An introduction to Object Oriented programming using Python.

DEC 20, 2022




Stable diffusion using 🤖
Hugging Face - DiffEdit paper implementation
8 min

STABLE DIFFUSIONRESEARCH

An implementation of DIFFEDIT: DIFFUSION-BASED SEMANTIC IMAGE EDITING WITH MASK GUIDANCE using 🤖 hugging face diffusers library.

NOV 17, 2022




Stable diffusion using 🤖
Hugging Face - Variations of Stable Diffusion
4 min

STABLE DIFFUSION

An introduction to negative prompting and image to image stable diffusion pipeline using 🤖 hugging face diffusers library.

NOV 11, 2022




Stable diffusion using 🤖
Hugging Face - Putting everything together
3 min

STABLE DIFFUSION

An introduction to the diffusion process using 🤖 hugging face diffusers library.

NOV 7, 2022




Stable diffusion using 🤖
Hugging Face - Looking under the hood
9 min

STABLE DIFFUSION

An introduction into what goes on in the pipe function of 🤖 hugging face diffusers library

StableDiffusionPipeline function.

NOV 5, 2022




Stable diffusion using 🤖
Hugging Face - Introduction
2 min


STABLE DIFFUSION

A brief introduction to start generating images from text prompts using 🤖 hugging face - Diffusers library.

NOV 2, 2022

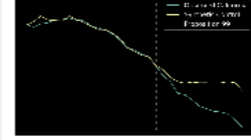


Model calibration for classification tasks using Python
6 min



Mixing art into the science of model explainability
9 min

EXPLAINABILITYMACHINE LEARNING



Causal inference with Synthetic Control using Python and SparseSC
7 min

Categories

All (17)

Business (1)

Causal Inference (1)

Deep Learning (6)

Explainability (1)

FastAI (4)

Machine Learning (5)

Model Calibration (1)

Programming (1)

Pytorch (3)

Recommender System (1)

Research (1)

Stable Diffusion (5)

Synthetic Control (1)

Video (1)

Vision (4)

announcement (1)

launch (1)

Book: Geocomputation with Python

_quarto.yml

```
1 project:
2   type: book
3
4 book:
5   title: "Geocomputation with Python"
6   author: |
7     Michael Dorman, Anita Graser,
8     Jakub Nowosad, Robin Lovelace
9   description: |
10     An introductory resource for working with geographic
11     data in Python
12   cover-image: https://geocompx.org/static/img/book_cover_p
13   site-url: https://py.geocompx.org
14   repo-url: https://github.com/geocompx/geocompy/
15   repo-actions: [edit]
16   sharing: [twitter, facebook, linkedin]
17   chapters:
18     - index.qmd
19     - preface.qmd
20     - 02-spatial-data.qmd
21     - 03-attribute-operations.qmd
22     - 04-spatial-operations.qmd
23     - 05-geometry-operations.qmd
24     - 06-raster-vector.qmd
25     - 07-reproj.qmd
26
27 format:
28   html:
29     theme: flatly
30     template-partials: [toc.html,title-block.html]
31     code-overflow: wrap
```

_quarto.yml

```
1 project:
2   type: book
3
4 book:
5   title: "Geocomputation with Python"
6   author: |
7     Michael Dorman, Anita Graser,
8     Jakub Nowosad, Robin Lovelace
9   description: |
10     An introductory resource for working with geographic
11     data in Python
12   cover-image: https://geocompx.org/static/img/book_cover_p
13   site-url: https://py.geocompx.org
14   repo-url: https://github.com/geocompx/geocompy/
15   repo-actions: [edit]
16   sharing: [twitter, facebook, linkedin]
17   chapters:
18     - index.qmd
19     - preface.qmd
20     - 02-spatial-data.qmd
21     - 03-attribute-operations.qmd
22     - 04-spatial-operations.qmd
23     - 05-geometry-operations.qmd
24     - 06-raster-vector.qmd
25     - 07-reproj.qmd
26
27 format:
28   html:
29     theme: flatly
30     template-partials: [toc.html,title-block.html]
31     code-overflow: wrap
```

Geocomputation with Python



Welcome

Preface

1 Geographic data in Python

2 Attribute data operations

3 Spatial data operations

4 Geometry operations

5 Raster-vector interactions

6 Reprojecting geographic data

7 Geographic data I/O

8 Making maps with Python

3.3.5 Non-overlapping joins

Sometimes two geographic datasets do not touch but still have a strong geographic relationship. The datasets `cycle_hire` and `cycle_hire_osm`, provide a good example. Plotting them shows that they are often closely related but they do not touch, as shown in [Figure 3.6](#), which is created with the code below:

```
base = cycle_hire.plot(edgecolor='blue', color='none')
cycle_hire_osm.plot(ax=base, edgecolor='red',
                    color='none');
```

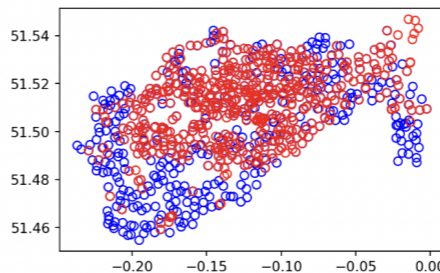


Figure 3.6: The spatial distribution of cycle hire points in London based on official data (blue) and OpenStreetMap data (red).

We can check if any of the points are the same by creating a pairwise boolean matrix of `.intersects` relations, then evaluating whether any of the values in it is `True`. Note that the `.to_numpy` method is applied to go from a `DataFrame` to a `numpy` array, for which `.any` gives a global rather than a row-wise summary, which is what we want in this case:

```
m = cycle_hire['geometry'].apply(
    lambda x: cycle_hire_osm['geometry'].intersects(x)
)
m.to_numpy().any()
```

False

Imagine that we need to join the capacity variable in `cycle_hire_osm` onto the official 'target' data contained in `cycle_hire`. This is when a non-overlapping join is needed. Spatial join (`gpd.sjoin`) along with buffered geometries can be used to do that. This is demonstrated below, using a threshold distance of 20 m. Note that we transform the data to a projected CRS (`27700`) to use real buffer distances, in meters.

```
crs = 27700
```

Note: The book is under construction 🚧

[Open an issue ?](#)

[Chat on Discord](#) 🗨️

On this page

3.1 Prerequisites

3.2 Introduction

3.3 Spatial operations on vector data

3.3.1 Spatial subsetting

3.3.2 Topological relations

3.3.3 DE-9IM strings

3.3.4 Spatial joining

3.3.5 Non-overlapping joins

3.3.6 Spatial aggregation

3.3.7 Joining incongruent layers

3.3.8 Distance relations

3.4 Spatial operations on raster data

3.4.1 Spatial subsetting

3.4.2 Map algebra

3.4.3 Local operations

3.4.4 Focal operations

3.4.5 Zonal operations

3.4.6 Global operations and distances

3.4.7 Map algebra counterparts in vector processing

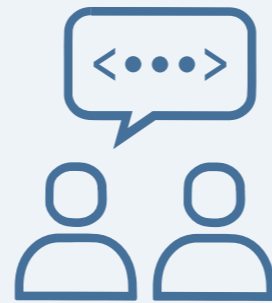
3.4.8 Merging rasters

3.5 Exercises

[Edit this page](#)

Books

- Inherit features of Quarto websites (navigation, search, mobile, etc.)
 - Support cross references across chapters
 - Produce multiple book formats from a single source
- HTML
 - PDF (LaTeX)
 - MS Word
 - ePub
 - AsciiDoc



Technical Communication

What is different?

- Sophisticated presentation of source code
- Figures, sub-figures, and figure panels
- Use of citations and cross references
- Content written in specialized languages and rendered into visual form (e.g. equations and diagrams)
- Specialized regions (e.g. callouts) and layout (e.g. use of margins)

Google Workspace Updates

Easily format and display code in Google Docs

Wednesday, December 14, 2022

What's changing

Currently, when working in Google Docs, collaborators who want to present code have to paste it in the document and then manually apply styles by highlighting syntax.

We're adding a [new smart canvas](#) feature that makes this process much easier by enabling you to format and display code in Docs with code blocks.

Source Code Annotation

Code blocks and executable code cells may include annotations, which provide a way to attach explanations to code (much like footnotes)

```
1  ```python
2  word_index = imdb.get_word_index()           # <1>
3  reverse_word_index = dict(
4      [(value, key) for (key, value) in word_index.items()]) # <2>
5  decoded_review = ' '.join(
6      [reverse_word_index.get(i - 3, '?') for i in train_data[0]]) # <3>
7  ```
8
9  1. `word_index` is a dictionary mapping words to an integer index
10
11  2. Reverses it, mapping integer indices to words
12
13  3. Decodes the review. Indices are offset by 3 because 0, 1, and 2
14     are reserved for "padding", "start of sequence" and "unknown".
15
```

Code Annotation

```
word_index = imdb.get_word_index() ①
reverse_word_index = dict(
    [(value, key) for (key, value) in word_index.items()]) ②
decoded_review = ' '.join(
    [reverse_word_index.get(i - 3, '?') for i in train_data[0]]) ③
```

- ① `word_index` is a dictionary mapping words to an integer index
- ② Reverses it, mapping integer indices to words
- ③ Decodes the review. Note that the indices are offset by 3 because 0, 1, and 2 are reserved indices for "padding", "start of sequence" and "unknown".

Source Code Annotation

Code blocks and executable code cells may include annotations, which provide a way to attach explanations to code (much like footnotes)

```
1  ```python
2  word_index = imdb.get_word_index()                # <1>
3  reverse_word_index = dict(
4      [(value, key) for (key, value) in word_index.items()]) # <2>
5  decoded_review = ' '.join(
6      [reverse_word_index.get(i - 3, '?') for i in train_data[0]]) # <3>
7  ```
8
9  1. `word_index` is a dictionary mapping words to an integer index
10
11  2. Reverses it, mapping integer indices to words
12
13  3. Decodes the review. Indices are offset by 3 because 0, 1, and 2
14     are reserved for "padding", "start of sequence" and "unknown".
15
```

Code Annotation

```
word_index = imdb.get_word_index() ①
reverse_word_index = dict(
    [(value, key) for (key, value) in word_index.items()]) ②
decoded_review = ' '.join(
    [reverse_word_index.get(i - 3, '?') for i in train_data[0]]) ③
```

- ① `word_index` is a dictionary mapping words to an integer index
- ② Reverses it, mapping integer indices to words
- ③ Decodes the review. Note that the indices are offset by 3 because 0, 1, and 2 are reserved indices for "padding", "start of sequence" and "unknown".

Source Code Annotation

Code blocks and executable code cells may include annotations, which provide a way to attach explanations to code (much like footnotes)

```
1  ```python
2  word_index = imdb.get_word_index()           # <1>
3  reverse_word_index = dict(
4      [(value, key) for (key, value) in word_index.items()]) # <2>
5  decoded_review = ' '.join(
6      [reverse_word_index.get(i - 3, '?') for i in train_data[0]]) # <3>
7  ```
8
9  1. `word_index` is a dictionary mapping words to an integer index
10
11  2. Reverses it, mapping integer indices to words
12
13  3. Decodes the review. Indices are offset by 3 because 0, 1, and 2
14     are reserved for "padding", "start of sequence" and "unknown".
15
```

Code Annotation

```
word_index = imdb.get_word_index() ①
reverse_word_index = dict(
    [(value, key) for (key, value) in word_index.items()]) ②
decoded_review = ' '.join(
    [reverse_word_index.get(i - 3, '?') for i in train_data[0]]) ③
```

- ① `word_index` is a dictionary mapping words to an integer index
- ② Reverses it, mapping integer indices to words
- ③ Decodes the review. Note that the indices are offset by 3 because 0, 1, and 2 are reserved indices for "padding", "start of sequence" and "unknown".

Source Code Annotation

Code blocks and executable code cells may include annotations, which provide a way to attach explanations to code (much like footnotes)

```
1  ```python
2  word_index = imdb.get_word_index()           # <1>
3  reverse_word_index = dict(
4      [(value, key) for (key, value) in word_index.items()]) # <2>
5  decoded_review = ' '.join(
6      [reverse_word_index.get(i - 3, '?') for i in train_data[0]]) # <3>
7  ```
8
9  1. `word_index` is a dictionary mapping words to an integer index
10
11  2. Reverses it, mapping integer indices to words
12
13  3. Decodes the review. Indices are offset by 3 because 0, 1, and 2
14     are reserved for "padding", "start of sequence" and "unknown".
15
```

Code Annotation

```
word_index = imdb.get_word_index() ①
reverse_word_index = dict(
    [(value, key) for (key, value) in word_index.items()]) ②
decoded_review = ' '.join(
    [reverse_word_index.get(i - 3, '?') for i in train_data[0]]) ③
```

- ① `word_index` is a dictionary mapping words to an integer index
- ② Reverses it, mapping integer indices to words
- ③ Decodes the review. Note that the indices are offset by 3 because 0, 1, and 2 are reserved indices for "padding", "start of sequence" and "unknown".

Diagrams

Native support for embedding [Mermaid](#) and [Graphviz](#) diagrams.

```
```{mermaid}
sequenceDiagram
 participant Alice
 participant Bob
 Alice->>John: Hello John, how are you?
 loop Healthcheck
 John->>John: Fight against hypochondria
 end
 Note right of John: Rational thoughts
prevail!
 John-->>Alice: Great!
 John->>Bob: How about you?
 Bob-->>John: Jolly good!
```
```

```
```{dot}
graph G {
 layout=neato
 run -- intr;
 intr -- runbl;
 runbl -- run;
 run -- kernel;
 kernel -- zombie;
}
```
```

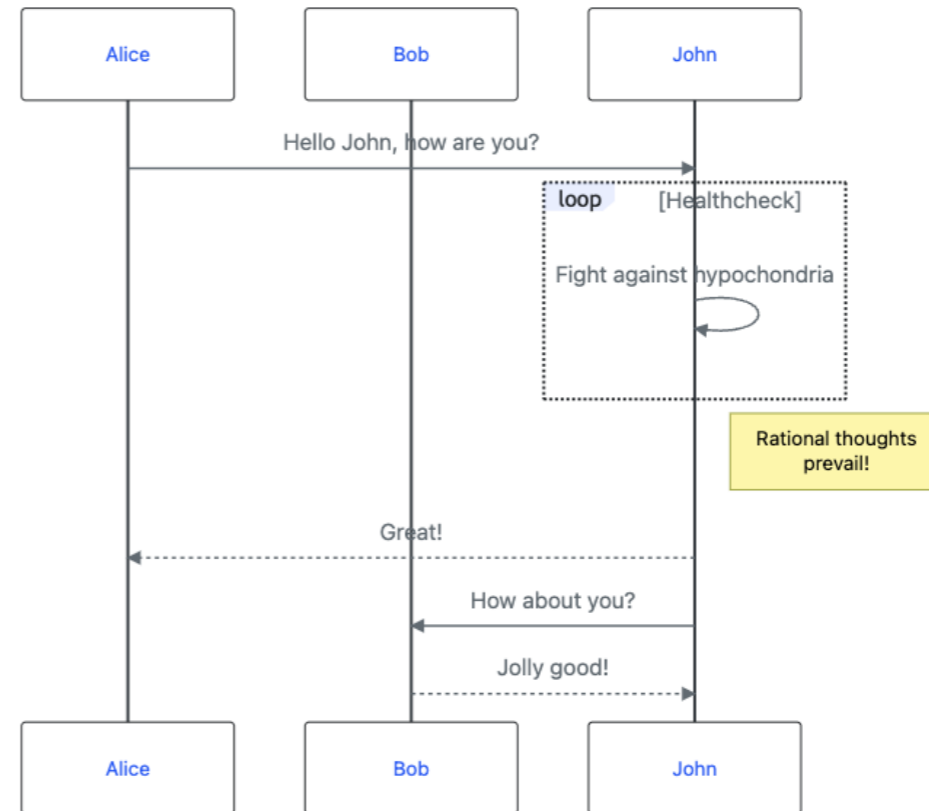
Diagrams

Native support for embedding [Mermaid](#) and [Graphviz](#) diagrams.

```
```{mermaid}
sequenceDiagram
 participant Alice
 participant Bob
 Alice->>John: Hello John, how are you?
 loop Healthcheck
 John->>John: Fight against hypochondria
 end
 Note right of John: Rational thoughts
prevail!
 John-->>Alice: Great!
 John->>Bob: How about you?
 Bob-->>John: Jolly good!
```
```

```
```{dot}
graph G {
 layout=neato
 run -- intr;
 intr -- runbl;
 runbl -- run;
 run -- kernel;
 kernel -- zombie;
}
```
```

Mermaid



Equations

LaTeX Equations (supported for all output formats)

```
$$  
\Delta C \approx \frac{\partial C}{\partial v_1} \Delta v_1 +  
\frac{\partial C}{\partial v_2} \Delta v_2.  
$$
```

```
$$  
\nabla C \equiv \left( \frac{\partial C}{\partial v_1},  
\frac{\partial C}{\partial v_2} \right)^T.  
$$
```

Equations

LaTeX Equations (supported for all output formats)

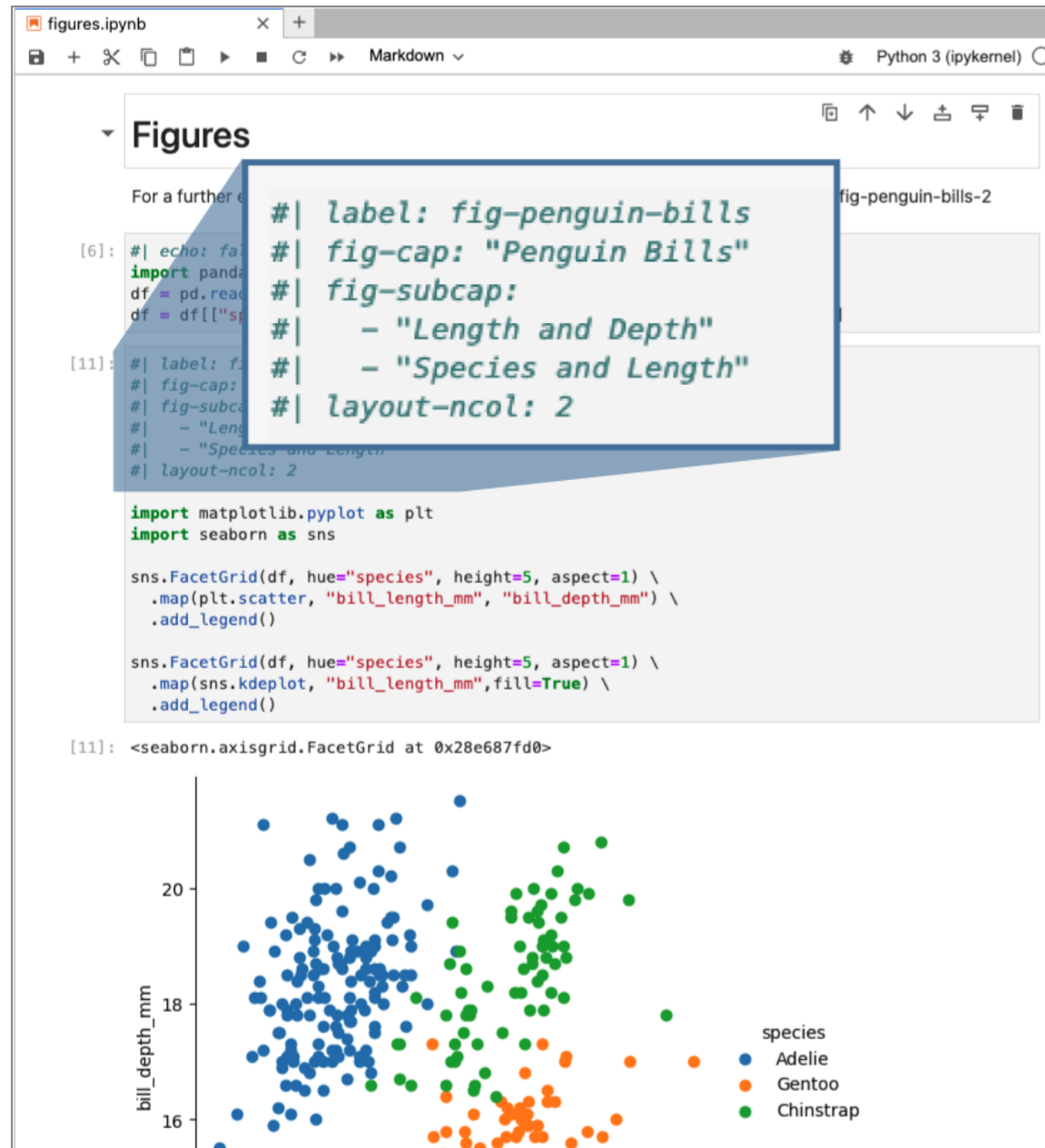
```
$$  
\Delta C \approx \frac{\partial C}{\partial v_1} \Delta v_1 +  
\frac{\partial C}{\partial v_2} \Delta v_2.  
$$
```

```
$$  
\nabla C \equiv \left( \frac{\partial C}{\partial v_1},  
\frac{\partial C}{\partial v_2} \right)^T.  
$$
```

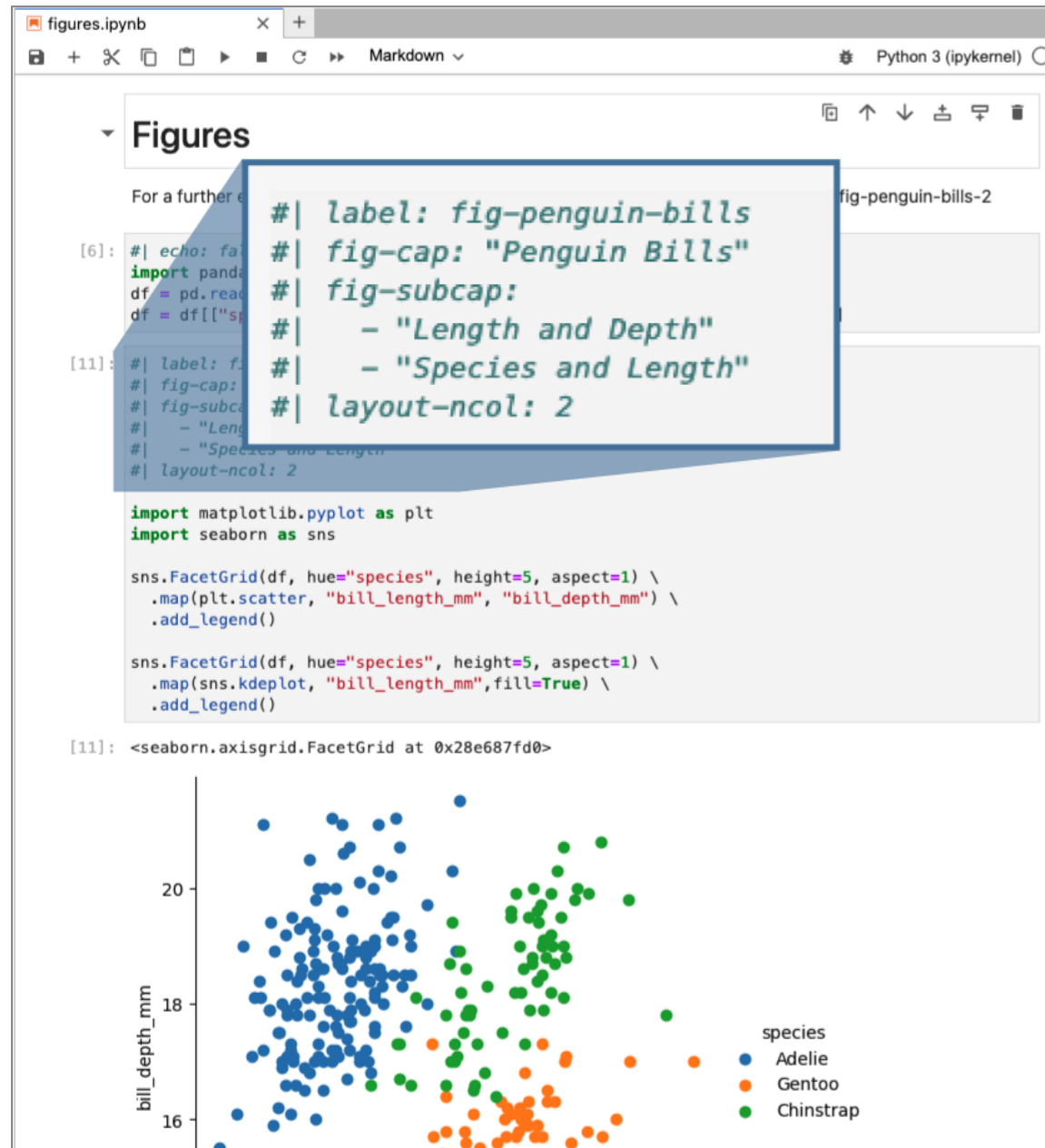
$$\Delta C \approx \frac{\partial C}{\partial v_1} \Delta v_1 + \frac{\partial C}{\partial v_2} \Delta v_2.$$

$$\nabla C \equiv \left(\frac{\partial C}{\partial v_1}, \frac{\partial C}{\partial v_2} \right)^T.$$

Figures and Cross References



Figures and Cross References



Figures

For a further exploration of these relationships, see [Figure 1](#) and especially [Figure 1\(b\)](#).

```
import matplotlib.pyplot as plt
import seaborn as sns

sns.FacetGrid(df, hue="species", height=5, aspect=1) \
    .map(plt.scatter, "bill_length_mm", "bill_depth_mm") \
    .add_legend()

sns.FacetGrid(df, hue="species", height=5, aspect=1) \
    .map(sns.kdeplot, "bill_length_mm", fill=True) \
    .add_legend()
```

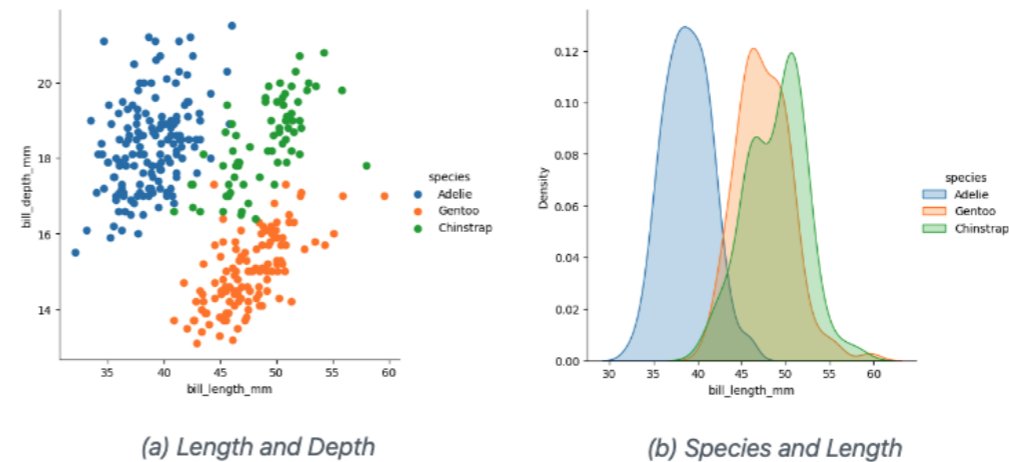


Figure 1: Penguin Bills

Citations

Pandoc includes robust support for citations in a wide variety of formats including [BibTeX](#), [CSL](#), and [RIS](#). More than [10,000 citation output styles](#) supported via CSL.

Markdown Syntax

Output

Blah Blah [see @knuth1984, pp. 33–35;
also @wickham2015, chap. 1]

Blah Blah (see [Knuth 1984, 33–35](#);
also [Wickham 2015, chap. 1](#))

Blah Blah [@knuth1984, pp. 33–35,
38–39 and passim]

Blah Blah ([Knuth 1984, 33–35, 38–39](#)
and passim)

Blah Blah [@wickham2015; @knuth1984].

Blah Blah ([Wickham 2015; Knuth 1984](#)).

Callouts

Supported for HTML, PDF, MS Word, Revealjs, ePub, JATS, AsciiDoc, Docusaurus, and Confluence formats

```
::: {.callout-note}
Note that there are five types of callouts
:::
```

Note

Note that there are five types of callouts, including: `note`, `tip`, `warning`, `caution`, and `important`.

Tip With Caption

This is an example of a callout with a caption.

This is Important

Danger, callouts will really improve your writing.

Warning

Callouts provide a simple way to attract attention, for example, to this warning.

Expand To Learn About Collapse

This is an example of a 'collapsed' caution callout that can be expanded by the user. You can use `collapse="true"` to collapse it by default or `collapse="false"` to make a collapsible callout that is expanded by default.

Margin Layout

margin.ipynb

Python 3 (ipykernel)

Margin Layout

author: Norah Jones
date: March 22, 2023

Abritrary Content

You can place arbitrary content in the margin using a div with the `.column-margin` class.

```
 ::: {.column-margin}  
 We know from *the first fundamental theorem of calculus* that for  $x$  in  $[a, b]$ :  
  
 
$$\frac{d}{dx} \left( \int_a^x f(u) du \right) = f(x).$$
  
 :::
```

Figures and Tables

Figures the option.

```
[2]: #| column:  
 import matplotlib.pyplot as plt  
 plt.plot([1,23,2,4])  
 plt.show()
```



Margin Layout

margin.ipynb

Python 3 (ipykernel)

Margin Layout

author: Norah Jones
date: March 22, 2023

Abritrary Content

You can place arbitrary content in the margin using a div with the `.column-margin` class.

```
::: {.column-margin}
We know from *the first fundamental theorem of calculus* that for  $x$  in  $[a, b]$ :

$$\frac{d}{dx} \left( \int_a^x f(u) du \right) = f(x).$$

:::
```

Figures and Tables

Figures that you create using code cells can be placed in the margin by using the `column: margin` cell option.

```
[2]: #| column: margin
import matplotlib.pyplot as plt
plt.plot([1,23,2,4])
plt.show()
```



Margin Layout

AUTHOR
Norah Jones

PUBLISHED
March 22, 2023

Abritrary Content

You can place arbitrary content in the margin using a div with the `.column-margin` class.

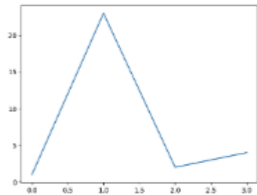
We know from *the first fundamental theorem of calculus* that for x in $[a, b]$:

$$\frac{d}{dx} \left(\int_a^x f(u) du \right) = f(x).$$

Figures and Tables

Figures that you create using code cells can be placed in the margin by using the `column: margin` cell option.

```
import matplotlib.pyplot as plt
plt.plot([1,23,2,4])
plt.show()
```

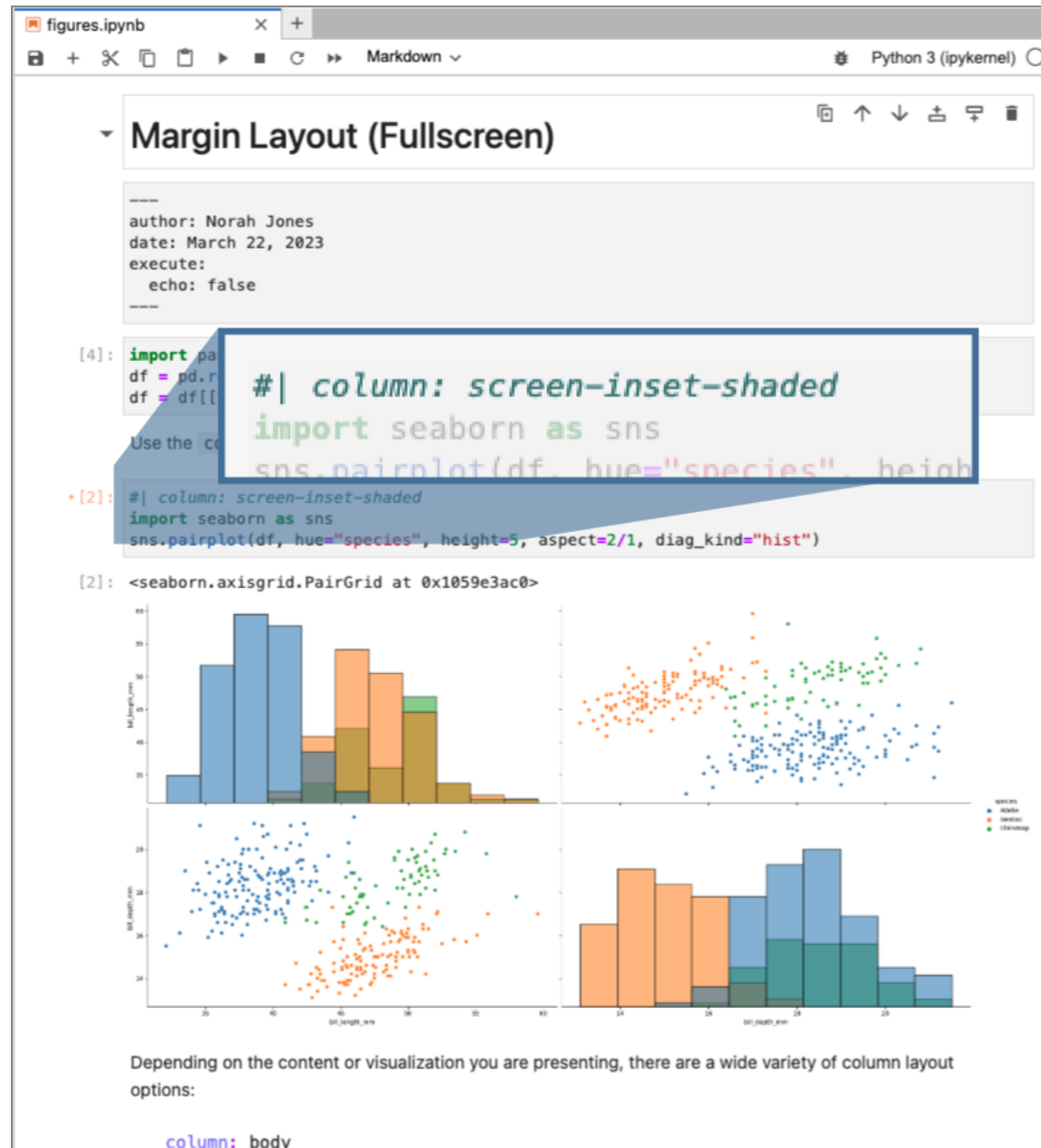


You can also use `column: margin` with tables:

```
from IPython.display import Markdown
from tabulate import tabulate
table = [
    ["Sun", 696000],
    ["Earth", 6371.5973.6],
    ["Moon", 1737.73.5],
    ["Mars", 3390.641.85]
]
Markdown(tabulate(
    table,
    headers=["Planet", "R (km)"]
))
```

| Planet | R (km) |
|--------|--------|
| Sun | 696000 |
| Earth | 6371 |
| Moon | 1737 |
| Mars | 3390 |

Margin Layout (Fullscreen)



Margin Layout (Fullscreen)

figures.ipynb

Python 3 (ipykernel)

Margin Layout (Fullscreen)

```
author: Norah Jones
date: March 22, 2023
execute:
  echo: false
```

```
[4]: import pandas as pd
df = pd.read_csv('data.csv')
df = df[['length', 'depth', 'species']]

Use the column layout to create a pair plot.

[2]: #| column: screen-inset-shaded
import seaborn as sns
sns.pairplot(df, hue="species", height=5, aspect=2/1, diag_kind="hist")

[2]: <seaborn.axisgrid.PairGrid at 0x1059e3ac0>
```

Depending on the content or visualization you are presenting, there are a wide variety of column layout options:

Margin Layout (Fullscreen)

AUTHOR

Norah Jones

PUBLISHED

March 22, 2023

Use the `column: screen-inset` option to create full bleed visualizations:

Depending on the content or visualization you are presenting, there are a wide variety of column layout options:

```
column: body
column: body-outset
column: page
column: page-left
column: page-right
column: screen-inset
column: screen-inset-shaded
```

See the [Article Layout](#) docs for additional details.



Semantic Authoring

Literal Authoring

- Writing a dissertation using LaTeX

L^AT_EX

- A proposal using MS Word



- Adding articles to a Hugo website



- Creating a presentation with Keynote



- Publishing to a Confluence Wiki



Challenges

Challenges

- Each of these has their own proprietary format, making it awkward and time consuming to re-purpose content across mediums.

Challenges

- Each of these has their own proprietary format, making it awkward and time consuming to re-purpose content across mediums.
- Format native authoring tools have variable (sometimes non-existent) support for technical content like code, math, diagrams, figures, crossrefs, etc.

Challenges

- Each of these has their own proprietary format, making it awkward and time consuming to re-purpose content across mediums.
- Format native authoring tools have variable (sometimes non-existent) support for technical content like code, math, diagrams, figures, crossrefs, etc.
- Generally there is no mechanism for including live code and its output (normally done via copy and paste or manually managed files)

Challenges

- Each of these has their own proprietary format, making it awkward and time consuming to re-purpose content across mediums.
- Format native authoring tools have variable (sometimes non-existent) support for technical content like code, math, diagrams, figures, crossrefs, etc.
- Generally there is no mechanism for including live code and its output (normally done via copy and paste or manually managed files)
- No straightforward way to automate / reproduce computationally derived content

Semantic Authoring

semantic.md

```
1 ---
2 title: "My Document"
3 ---
4
5 ## Code Blocks
6
7 This is a code block:
8
9 ```python
10 def add(x, y):
11     return x + 1
12 ```
13
14 ## Block Quotes
15
16 > This is a block quote. Block quotes are
17 > specified by proceeding lines with `>`
18
```

Semantic Authoring

semantic.md

```
1 ---
2 title: "My Document"
3 ---
4
5 ## Code Blocks
6
7 This is a code block:
8
9 ```python
10 def add(x, y):
11     return x + 1
12 ```
13
14 ## Block Quotes
15
16 > This is a block quote. Block quotes are
17 > specified by proceeding lines with `>`
18
```

- Compose with semantic structure (heading, emphasis, code, etc.) that is output independent

Semantic Authoring

semantic.md

```
1 ---
2 title: "My Document"
3 ---
4
5 ## Code Blocks
6
7 This is a code block:
8
9 ```python
10 def add(x, y):
11     return x + 1
12 ```
13
14 ## Block Quotes
15
16 > This is a block quote. Block quotes are
17 > specified by proceeding lines with `>`
18
```

- Compose with semantic structure (heading, emphasis, code, etc.) that is output independent
- Document is parsed into an AST (*abstract syntax tree*) that can be easily computed on! (key to supporting arbitrary output formats)

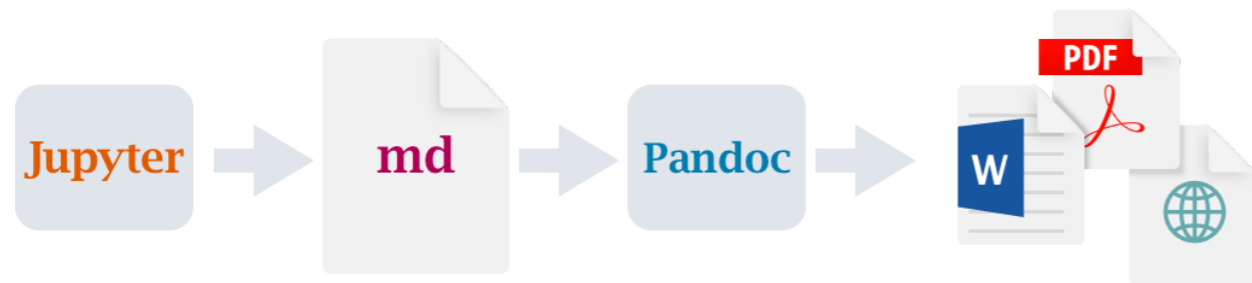
Semantic Authoring

semantic.md

```
1 ---
2 title: "My Document"
3 ---
4
5 ## Code Blocks
6
7 This is a code block:
8
9 ```python
10 def add(x, y):
11     return x + 1
12 ```
13
14 ## Block Quotes
15
16 > This is a block quote. Block quotes are
17 > specified by proceeding lines with `>`
18
```

- Compose with semantic structure (heading, emphasis, code, etc.) that is output independent
- Document is parsed into an AST (*abstract syntax tree*) that can be easily computed on! (key to supporting arbitrary output formats)
- Production quality output can be created for any format required

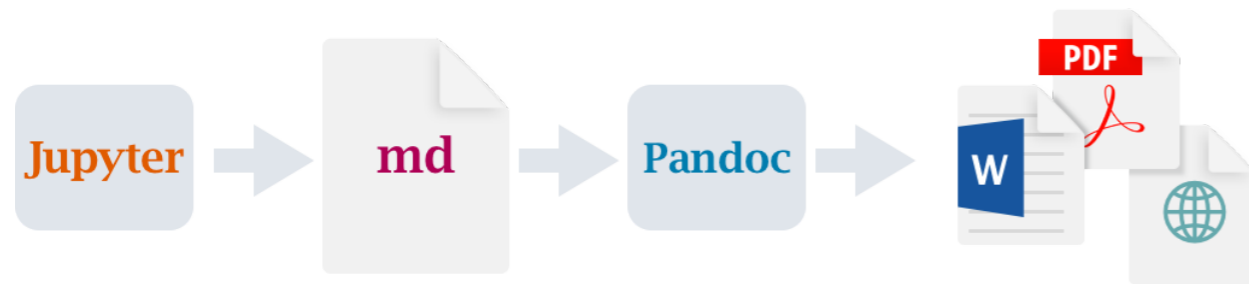
Semantic Authoring with Jupyter



Jupyter executes cells and produces a document with markdown cells and markdown executable output

Pandoc renders markdown into a variety of formats

Semantic Authoring with Jupyter

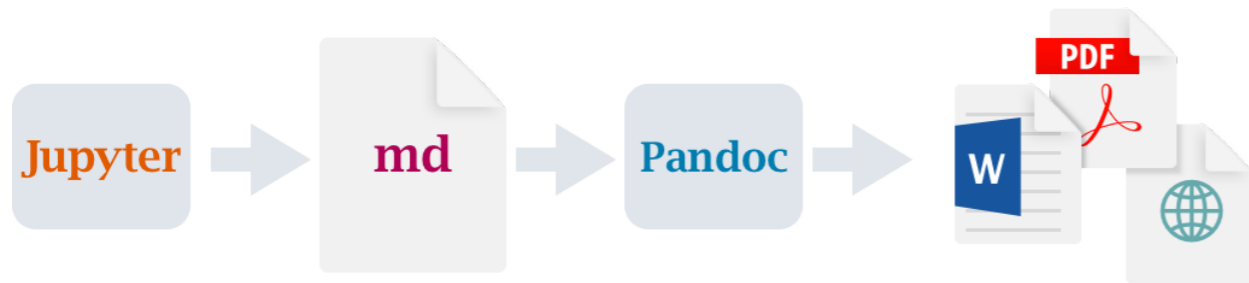


- Markdown is translated into document AST

Jupyter executes cells and produces a document with markdown cells and markdown executable output

Pandoc renders markdown into a variety of formats

Semantic Authoring with Jupyter

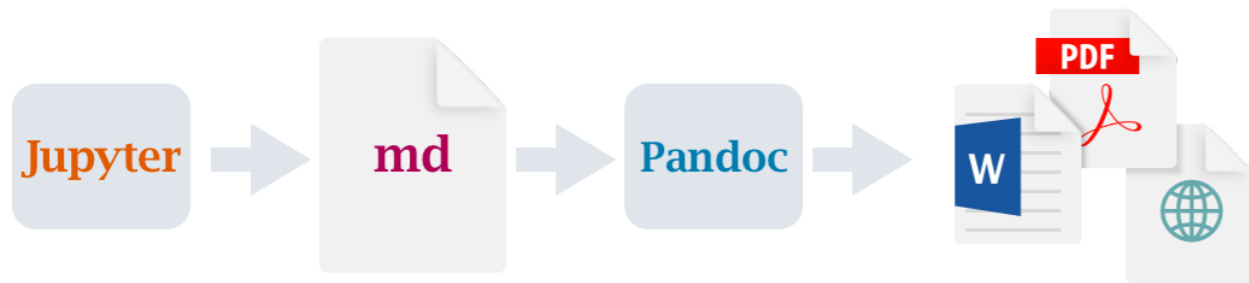


Jupyter executes cells and produces a document with markdown cells and markdown executable output

Pandoc renders markdown into a variety of formats

- Markdown is translated into **document AST**
- Filters transform the AST (e.g. to implement layout, crossrefs, code folding, etc.)

Semantic Authoring with Jupyter



Jupyter executes cells and produces a document with markdown cells and markdown executable output

Pandoc renders markdown into a variety of formats

- Markdown is translated into **document AST**
- Filters transform the AST (e.g. to implement layout, crossrefs, code folding, etc.)
- Final output rendered from the AST

Pandoc Formats

Documents

- HTML
- PDF
- MS Word
- Open Office
- ePub

Presentations

- Revealjs
- PowerPoint
- Beamer

Markdown

- CommonMark
- GitHub (GFM)
- Markua

Pandoc Formats (cont.)

Wikis

- MediaWiki
- DocuWiki
- ZimWiki
- Jira Wiki
- XWiki

Other

- JATS
- ConTeXt
- reST
- AsciiDoc
- Org-mode

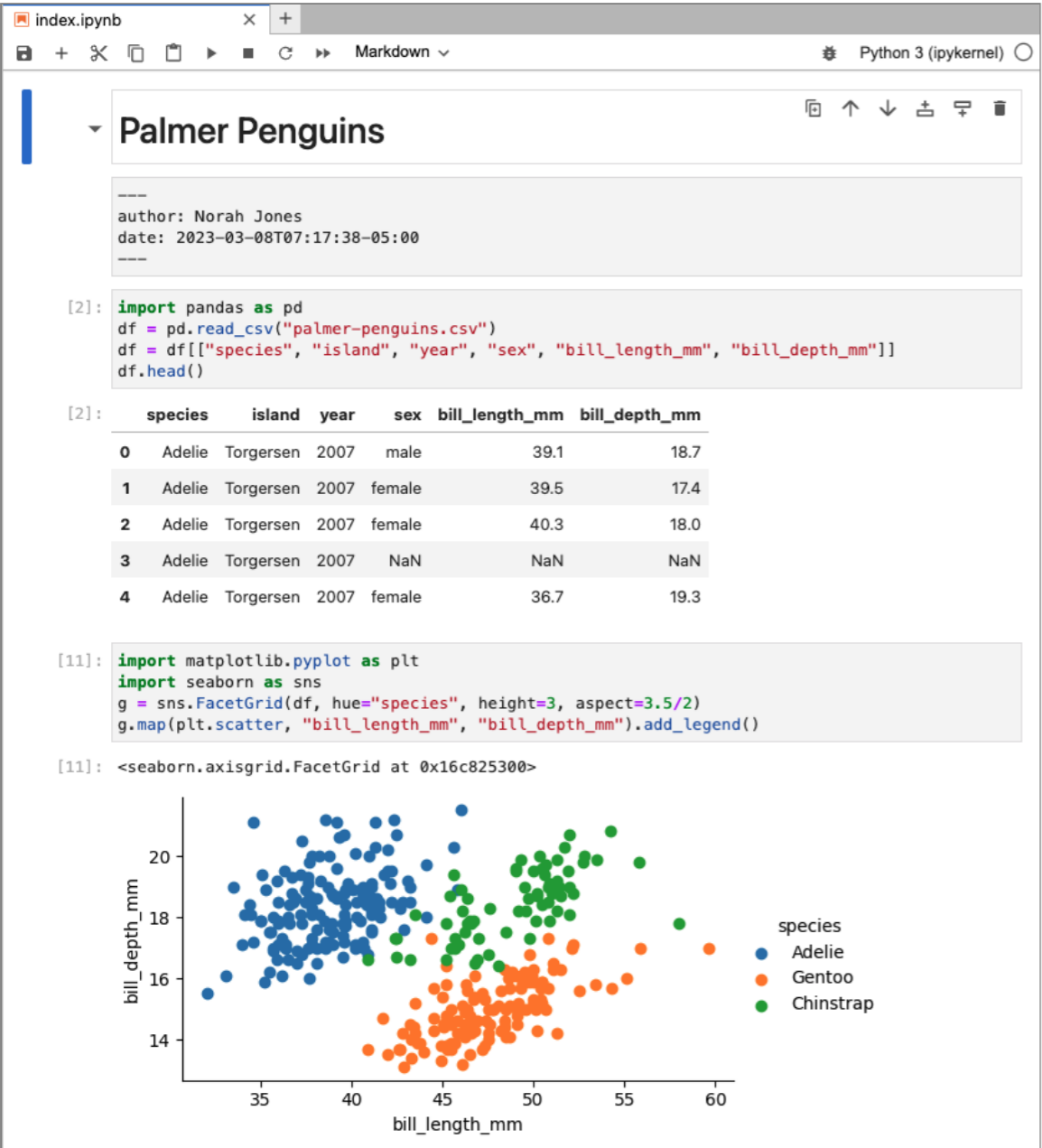
- Textile
- DocBook
- InDesign
- GNU Texinfo
- FictionBook

Content Management Systems

Because Quarto and Pandoc are based on a semantic AST, we can also publish to any content management system we need to. For example:

- **Hugo** (Goldmark Markdown)
- **Docusaurus** (MDX Markdown)
- **Confluence** (Confluence XML)
- **O'Reilly Atlas** (Asciidoc)

Hugo: Goldmark Markdown



Hugo: Goldmark Markdown

index.ipynb

Python 3 (ipykernel)

Palmer Penguins

author: Norah Jones

date: 2023-03-08T07:17:38-05:00

```
[2]: import pandas as pd
df = pd.read_csv("palmer-penguins.csv")
df = df[["species", "island", "year", "sex", "bill_length_mm", "bill_depth_mm"]]
df.head()
```

| | species | island | year | sex | bill_length_mm | bill_depth_mm |
|---|---------|-----------|------|--------|----------------|---------------|
| 0 | Adelie | Torgersen | 2007 | male | 39.1 | 18.7 |
| 1 | Adelie | Torgersen | 2007 | female | 39.5 | 17.4 |
| 2 | Adelie | Torgersen | 2007 | female | 40.3 | 18.0 |
| 3 | Adelie | Torgersen | 2007 | NaN | NaN | NaN |
| 4 | Adelie | Torgersen | 2007 | female | 36.7 | 19.3 |

```
[11]: import matplotlib.pyplot as plt
import seaborn as sns
g = sns.FacetGrid(df, hue="species", height=3, aspect=3.5/2)
g.map(plt.scatter, "bill_length_mm", "bill_depth_mm").add_legend()
```

```
[11]: <seaborn.axisgrid.FacetGrid at 0x16c825300>
```

My New Hugo Site

POSTS

Palmer Penguins

By **Norah Jones**

March 8, 2023

```
import pandas as pd
df = pd.read_csv("palmer-penguins.csv")
df = df[["species", "island", "year", "sex", "bill_length_mm", "bill_depth_mm"]]
df.head()
```

| | species | island | year | sex | bill_length_mm | bill_depth_mm |
|---|---------|-----------|------|--------|----------------|---------------|
| 0 | Adelie | Torgersen | 2007 | male | 39.1 | 18.7 |
| 1 | Adelie | Torgersen | 2007 | female | 39.5 | 17.4 |
| 2 | Adelie | Torgersen | 2007 | female | 40.3 | 18.0 |
| 3 | Adelie | Torgersen | 2007 | NaN | NaN | NaN |
| 4 | Adelie | Torgersen | 2007 | female | 36.7 | 19.3 |

```
import matplotlib.pyplot as plt
import seaborn as sns
g = sns.FacetGrid(df, hue="species", height=3, aspect=3.5/2)
g.map(plt.scatter, "bill_length_mm", "bill_depth_mm").add_legend()
```

Docusaurus: MDX Markdown

2023-03-02-palmer-pengu

+

Python 3 (ipykernel)

Markdown

Palmer Penguins

author: Norah Jones

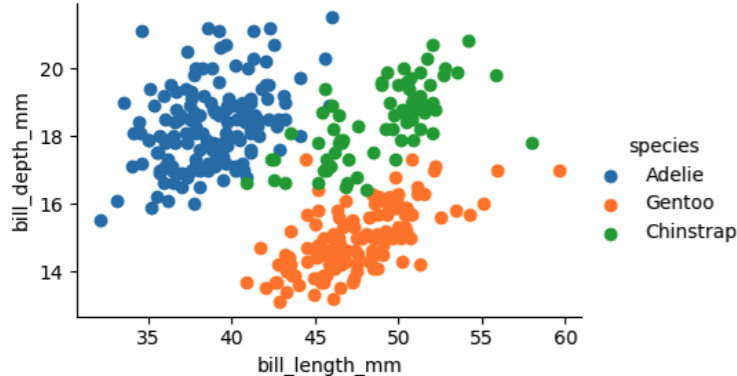
```
[7]: import pandas as pd
df = pd.read_csv("palmer-penguins.csv")
df = df[["species", "island", \
        "bill_length_mm", "bill_depth_mm"]]
df.head(3)
```

```
[7]:
```

| | species | island | bill_length_mm | bill_depth_mm |
|---|---------|-----------|----------------|---------------|
| 0 | Adelie | Torgersen | 39.1 | 18.7 |
| 1 | Adelie | Torgersen | 39.5 | 17.4 |
| 2 | Adelie | Torgersen | 40.3 | 18.0 |

```
[5]: import matplotlib.pyplot as plt
import seaborn as sns
g = sns.FacetGrid(df, hue="species", height=3, aspect=3/2)
g.map(plt.scatter, "bill_length_mm", "bill_depth_mm") \
.add_legend()
```

```
[5]: <seaborn.axisgrid.FacetGrid at 0x17fe7bd00>
```



Docusaurus: MDX Markdown

2023-03-02-palmer-pengu

Python 3 (ipykernel)

Palmer Penguins

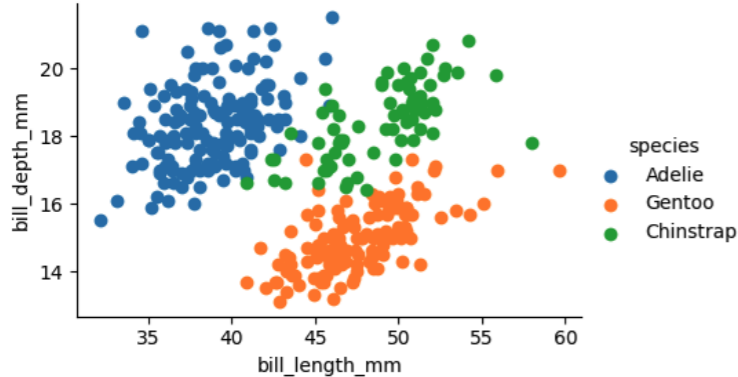
author: Norah Jones

```
[7]: import pandas as pd
df = pd.read_csv("palmer-penguins.csv")
df = df[["species", "island", \
        "bill_length_mm", "bill_depth_mm"]]
df.head(3)
```

```
[7]: species  island  bill_length_mm  bill_depth_mm
0  Adelie  Torgersen         39.1         18.7
1  Adelie  Torgersen         39.5         17.4
2  Adelie  Torgersen         40.3         18.0
```

```
[5]: import matplotlib.pyplot as plt
import seaborn as sns
g = sns.FacetGrid(df, hue="species", height=3, aspect=3/2)
g.map(plt.scatter, "bill_length_mm", "bill_depth_mm") \
.add_legend()
```

```
[5]: <seaborn.axisgrid.FacetGrid at 0x17fe7bd00>
```



My Site

Tutorial

Blog

GitHub

Recent posts

Palmer Penguins

Welcome

MDX Blog Post

Long Blog Post

First Blog Post

Palmer Penguins

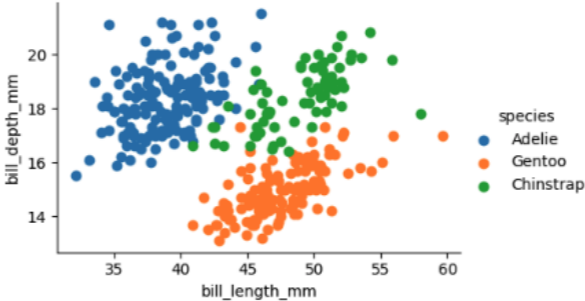
March 2, 2023 · One min read

Norah Jones

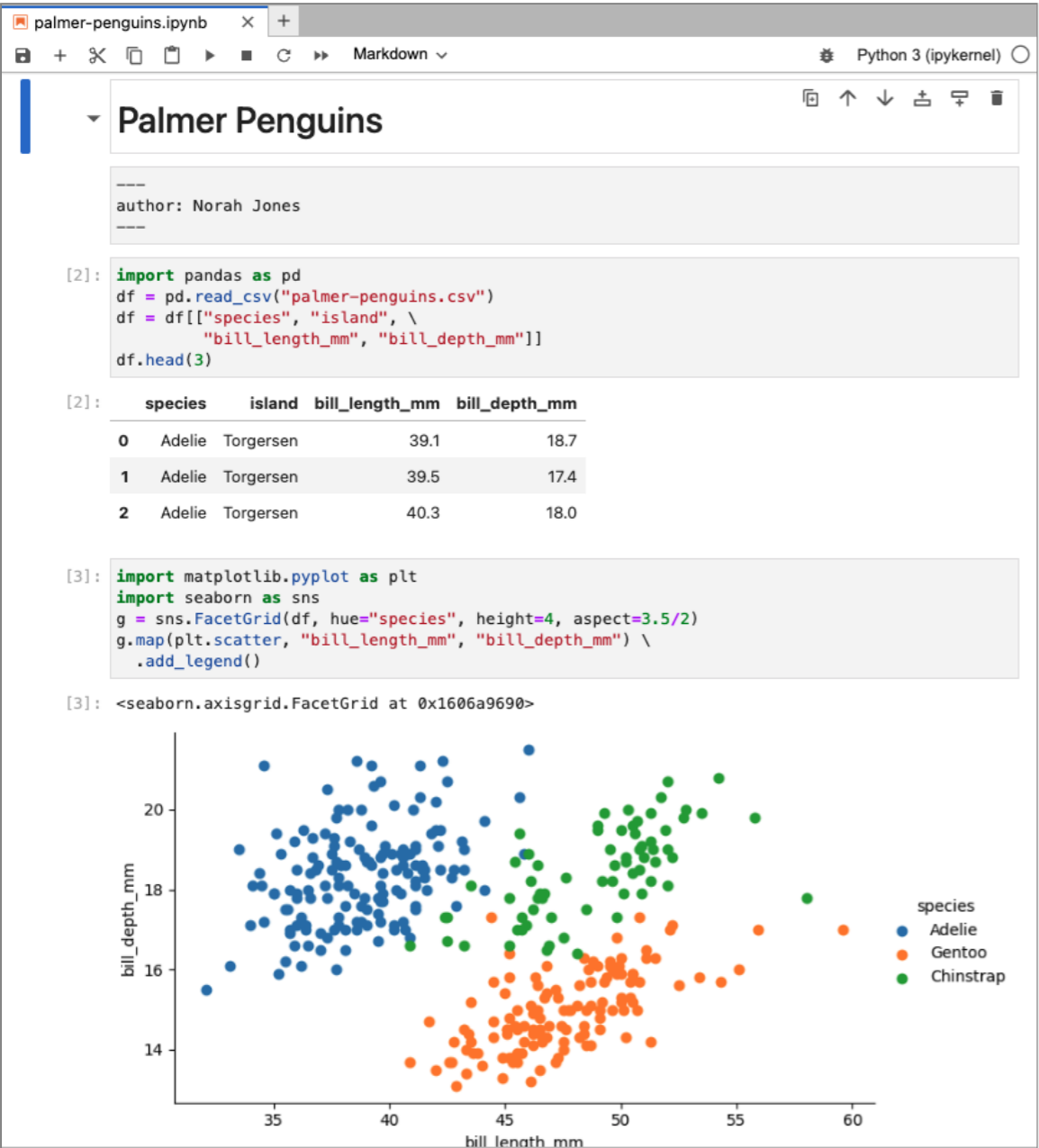
```
import pandas as pd
df = pd.read_csv("palmer-penguins.csv")
df = df[["species", "island", \
        "bill_length_mm", "bill_depth_mm"]]
df.head(3)
```

| | species | island | bill_length_mm | bill_depth_mm |
|---|---------|-----------|----------------|---------------|
| 0 | Adelie | Torgersen | 39.1 | 18.7 |
| 1 | Adelie | Torgersen | 39.5 | 17.4 |
| 2 | Adelie | Torgersen | 40.3 | 18.0 |

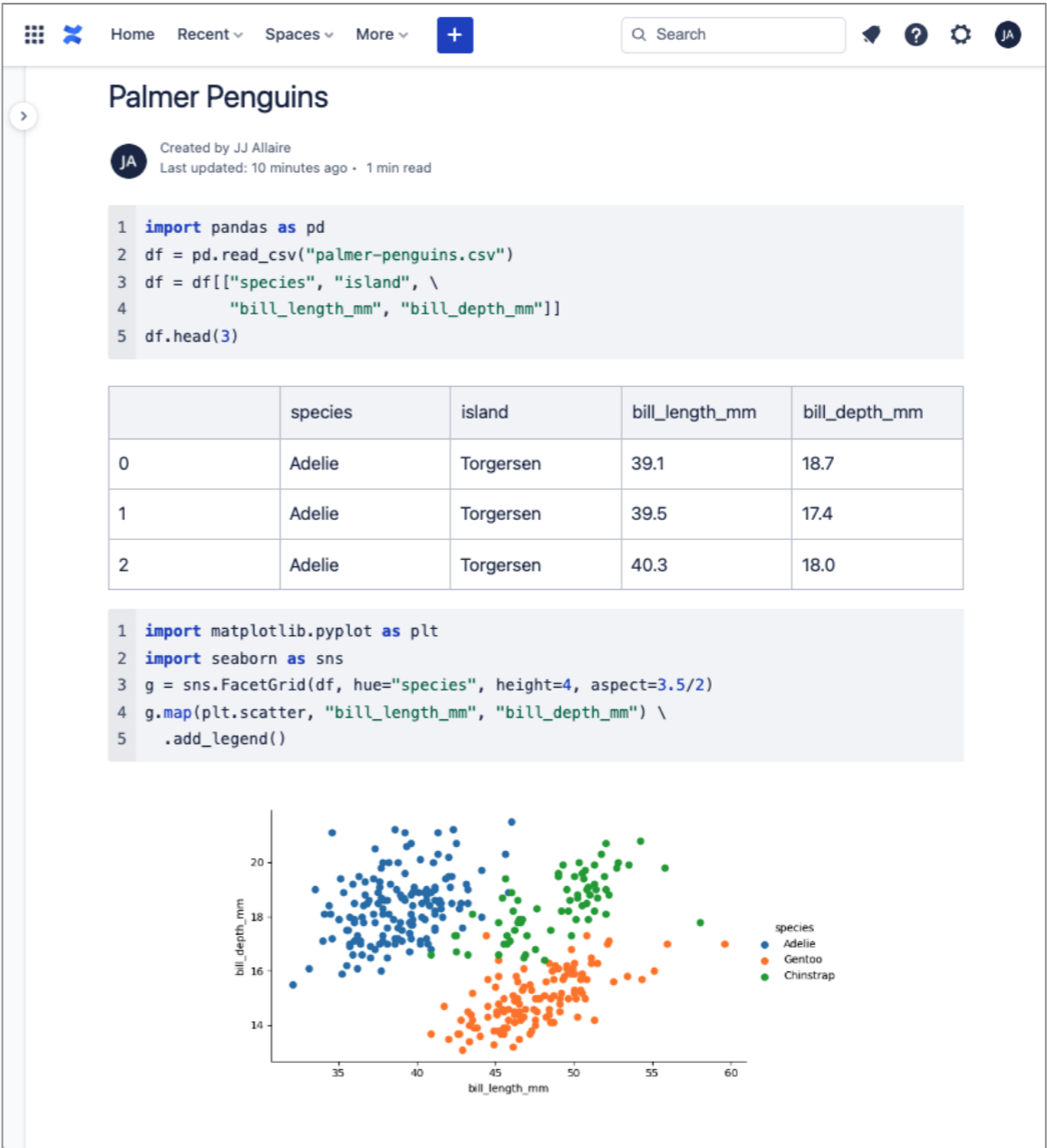
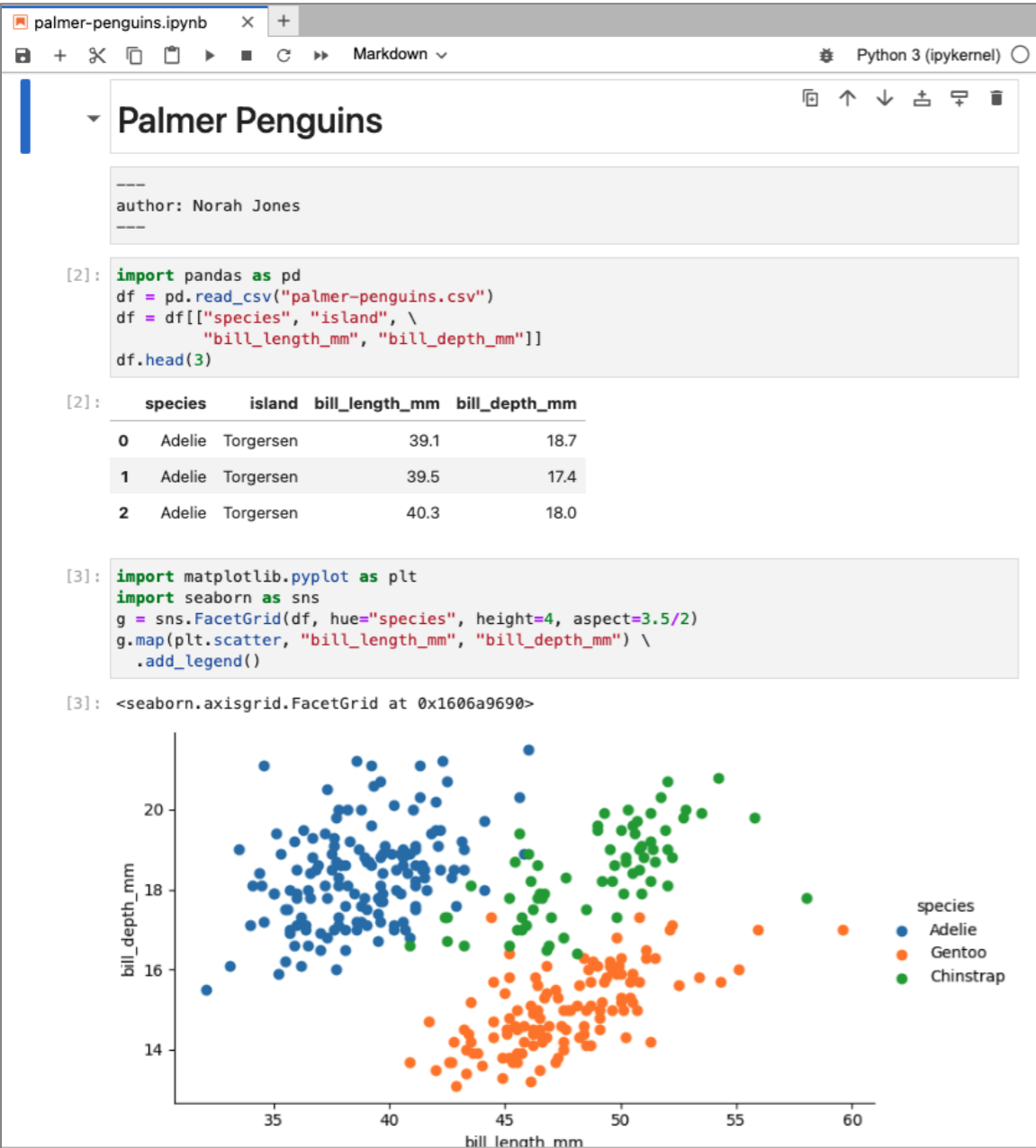
```
import matplotlib.pyplot as plt
import seaborn as sns
g = sns.FacetGrid(df, hue="species", height=3, aspect=3/2)
g.map(plt.scatter, "bill_length_mm", "bill_depth_mm") \
.add_legend()
```



Confluence: Confluence XML



Confluence: Confluence XML



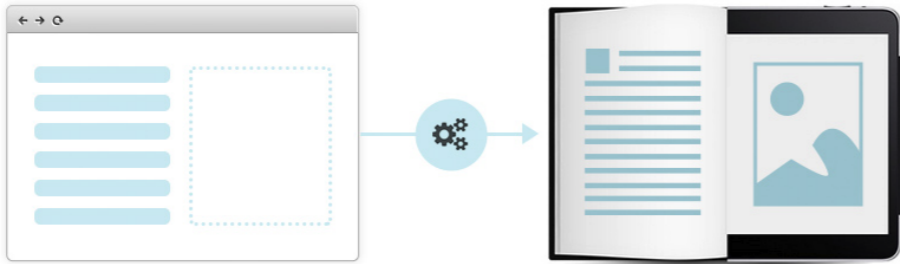
O'REILLY

Sign up with code Sign In


Atlas

Push-Button Publishing


Write. Build. Publish.




Features

 Version Control


Atlas is based on Git, the powerful version-control system that tracks every change in your content, who made it, and when it happened—and allows you to instantly revert to any previous version.

 Semantic Markup


Atlas uses HTMLBook—a specification of HTML—as the canonical source format, and template designs are maintained in CSS. Atlas also supports books written in AsciiDoc and DocBook XML.

 Digital Formats

Atlas exports to PDF, EPUB, and Web at the push of a button.

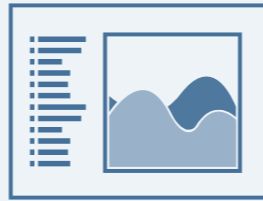
 STEM Content

Atlas has built-in support for MathML, LaTeX, and syntax highlighting.

 Interactivity

Embed audio and video for digital format output. For print output, these features are automatically converted to images.

Books can be rendered to `asciidoc`, which is fully compatible with the production requirements of O'Reilly Atlas (used for Print, ePub, and Web books)



Jupyter Notebooks

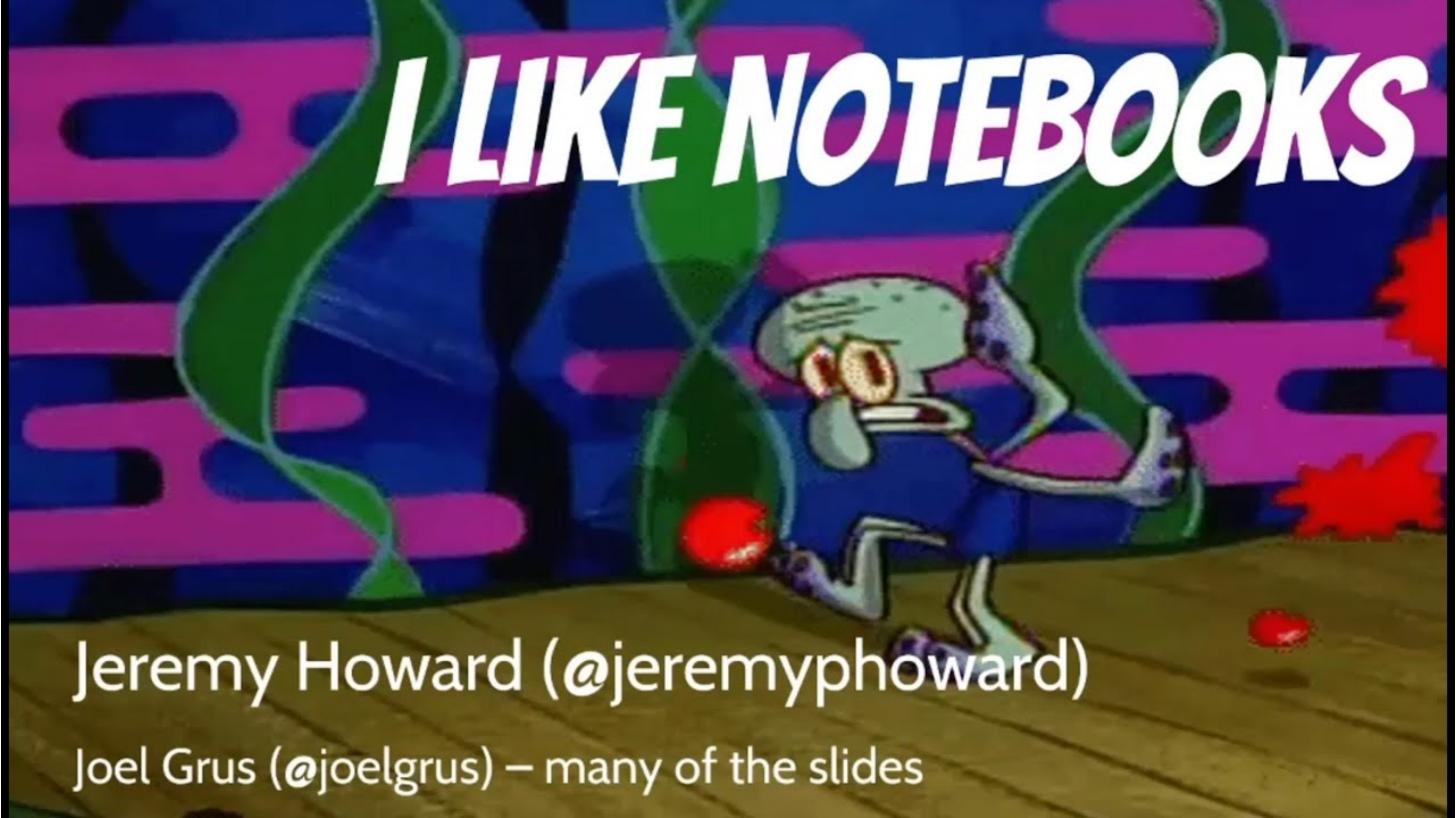
I DON'T LIKE NOTEBOOKS

Joel Grus (@joelgrus)

#JupyterCon 2018

(audience booing)

I LIKE NOTEBOOKS

A still from the animated show The Simpsons. Squidward Tentacles is running away from a red bomb on the ground. He has a shocked expression with wide eyes and an open mouth. The background shows a blue sky with pink clouds and green foliage. The ground is brown dirt.

Jeremy Howard (@jeremyphoward)

Joel Grus (@joelgrus) – many of the slides

Notebooks: The Coin of the Realm

The importance of having a standard container for code, output, and related narrative cannot be overestimated!

- Authoring tool (ipynb or plain text)
- Source of content for embedding in other documents
- Publishing format (static or interactive)

Notebook as Authoring Tool

Data science REPL with embedded narrative

Standard file format that is widely produced and consumed

A huge variety of notebook authoring tools are available...

- Jupyter Lab
- VS Code
- PyCharm
- Google CoLab
- Kaggle

Plain Text Authoring

[Jupyter](#) supports 10 different plain text formats for notebooks!

Markdown Formats

- Jupyter Markdown
- R Markdown
- MyST Markdown
- Pandoc Markdown
- Quarto Markdown

Script Formats (.py)

- light
- nomarker
- percent
- hydrogen
- Sphinx-gallery

Example: Quarto `.qmd` Files

penguins.qmd

```
1 ---
2 title: "Palmer Penguins"
3 author: Norah Jones
4 date: March 12, 2023
5 format: html
6 jupyter: python3
7 ---
8
9 ```{python}
10 #| echo: false
11
12 import pandas as pd
13 df = pd.read_csv("palmer-penguins.csv")
14 df = df[["species", "island", "year", \
15         "bill_length_mm", "bill_depth_mm"]]
16 ```
17
18 ## Exploring the Data
19
20 See @fig-bill-sizes for an exploration of bill sizes.
21
22 ```{python}
23 #| label: fig-bill-sizes
24 #| fig-cap: Bill Sizes by Species
25
26 import matplotlib.pyplot as plt
27 import seaborn as sns
28 g = sns.FacetGrid(df, hue="species", height=3)
29 g.map(plt.scatter, "bill_length_mm", "bill_depth_mm") \
30     .add_legend()
31 ```
```

- Editable with any text editor (extensions for VS Code, Neovim, and Emacs)
- Cells always run in the same order
- Integrates well with version control
- Cache output with Jupyter Cache or Quarto freezer
- Lots of pros and cons visa-vi traditional `.ipynb` format/editors, use the right tool for each job

Example: Quarto `.qmd` Files

penguins.qmd

```
1 ---
2 title: "Palmer Penguins"
3 author: Norah Jones
4 date: March 12, 2023
5 format: html
6 jupyter: python3
7 ---
8
9 ```{python}
10 #| echo: false
11
12 import pandas as pd
13 df = pd.read_csv("palmer-penguins.csv")
14 df = df[["species", "island", "year", \
15         "bill_length_mm", "bill_depth_mm"]]
16 ```
17
18 ## Exploring the Data
19
20 See @fig-bill-sizes for an exploration of bill sizes.
21
22 ```{python}
23 #| label: fig-bill-sizes
24 #| fig-cap: Bill Sizes by Species
25
26 import matplotlib.pyplot as plt
27 import seaborn as sns
28 g = sns.FacetGrid(df, hue="species", height=3)
29 g.map(plt.scatter, "bill_length_mm", "bill_depth_mm") \
30     .add_legend()
31 ```
```

- Editable with any text editor (extensions for VS Code, Neovim, and Emacs)
- Cells always run in the same order
- Integrates well with version control
- Cache output with Jupyter Cache or Quarto freezer
- Lots of pros and cons visa-vi traditional `.ipynb` format/editors, use the right tool for each job

Example: Quarto `.qmd` Files

penguins.qmd

```
1 ---
2 title: "Palmer Penguins"
3 author: Norah Jones
4 date: March 12, 2023
5 format: html
6 jupyter: python3
7 ---
8
9 ```{python}
10 #| echo: false
11
12 import pandas as pd
13 df = pd.read_csv("palmer-penguins.csv")
14 df = df[["species", "island", "year", \
15         "bill_length_mm", "bill_depth_mm"]]
16 ```
17
18 ## Exploring the Data
19
20 See @fig-bill-sizes for an exploration of bill sizes.
21
22 ```{python}
23 #| label: fig-bill-sizes
24 #| fig-cap: Bill Sizes by Species
25
26 import matplotlib.pyplot as plt
27 import seaborn as sns
28 g = sns.FacetGrid(df, hue="species", height=3)
29 g.map(plt.scatter, "bill_length_mm", "bill_depth_mm") \
30     .add_legend()
31 ```
```

- Editable with any text editor (extensions for VS Code, Neovim, and Emacs)
- Cells always run in the same order
- Integrates well with version control
- Cache output with Jupyter Cache or Quarto freezer
- Lots of pros and cons visa-vi traditional `.ipynb` format/editors, use the right tool for each job

Example: Quarto `.qmd` Files

penguins.qmd

```
1 ---
2 title: "Palmer Penguins"
3 author: Norah Jones
4 date: March 12, 2023
5 format: html
6 jupyter: python3
7 ---
8
9 ```{python}
10 #| echo: false
11
12 import pandas as pd
13 df = pd.read_csv("palmer-penguins.csv")
14 df = df[["species", "island", "year", \
15         "bill_length_mm", "bill_depth_mm"]]
16 ```
17
18 ## Exploring the Data
19
20 See @fig-bill-sizes for an exploration of bill sizes.
21
22 ```{python}
23 #| label: fig-bill-sizes
24 #| fig-cap: Bill Sizes by Species
25
26 import matplotlib.pyplot as plt
27 import seaborn as sns
28 g = sns.FacetGrid(df, hue="species", height=3)
29 g.map(plt.scatter, "bill_length_mm", "bill_depth_mm") \
30     .add_legend()
31 ```
```

- Editable with any text editor (extensions for VS Code, Neovim, and Emacs)
- Cells always run in the same order
- Integrates well with version control
- Cache output with Jupyter Cache or Quarto freezer
- Lots of pros and cons visa-vi traditional `.ipynb` format/editors, use the right tool for each job

Example: Quarto `.qmd` Files

penguins.qmd

```
1 ---
2 title: "Palmer Penguins"
3 author: Norah Jones
4 date: March 12, 2023
5 format: html
6 jupyter: python3
7 ---
8
9 ```{python}
10 #| echo: false
11
12 import pandas as pd
13 df = pd.read_csv("palmer-penguins.csv")
14 df = df[["species", "island", "year", \
15         "bill_length_mm", "bill_depth_mm"]]
16 ```
17
18 ## Exploring the Data
19
20 See @fig-bill-sizes for an exploration of bill sizes.
21
22 ```{python}
23 #| label: fig-bill-sizes
24 #| fig-cap: Bill Sizes by Species
25
26 import matplotlib.pyplot as plt
27 import seaborn as sns
28 g = sns.FacetGrid(df, hue="species", height=3)
29 g.map(plt.scatter, "bill_length_mm", "bill_depth_mm") \
30     .add_legend()
31 ```
```

- Editable with any text editor (extensions for VS Code, Neovim, and Emacs)
- Cells always run in the same order
- Integrates well with version control
- Cache output with Jupyter Cache or Quarto freezer
- Lots of pros and cons visa-vi traditional `.ipynb` format/editors, use the right tool for each job

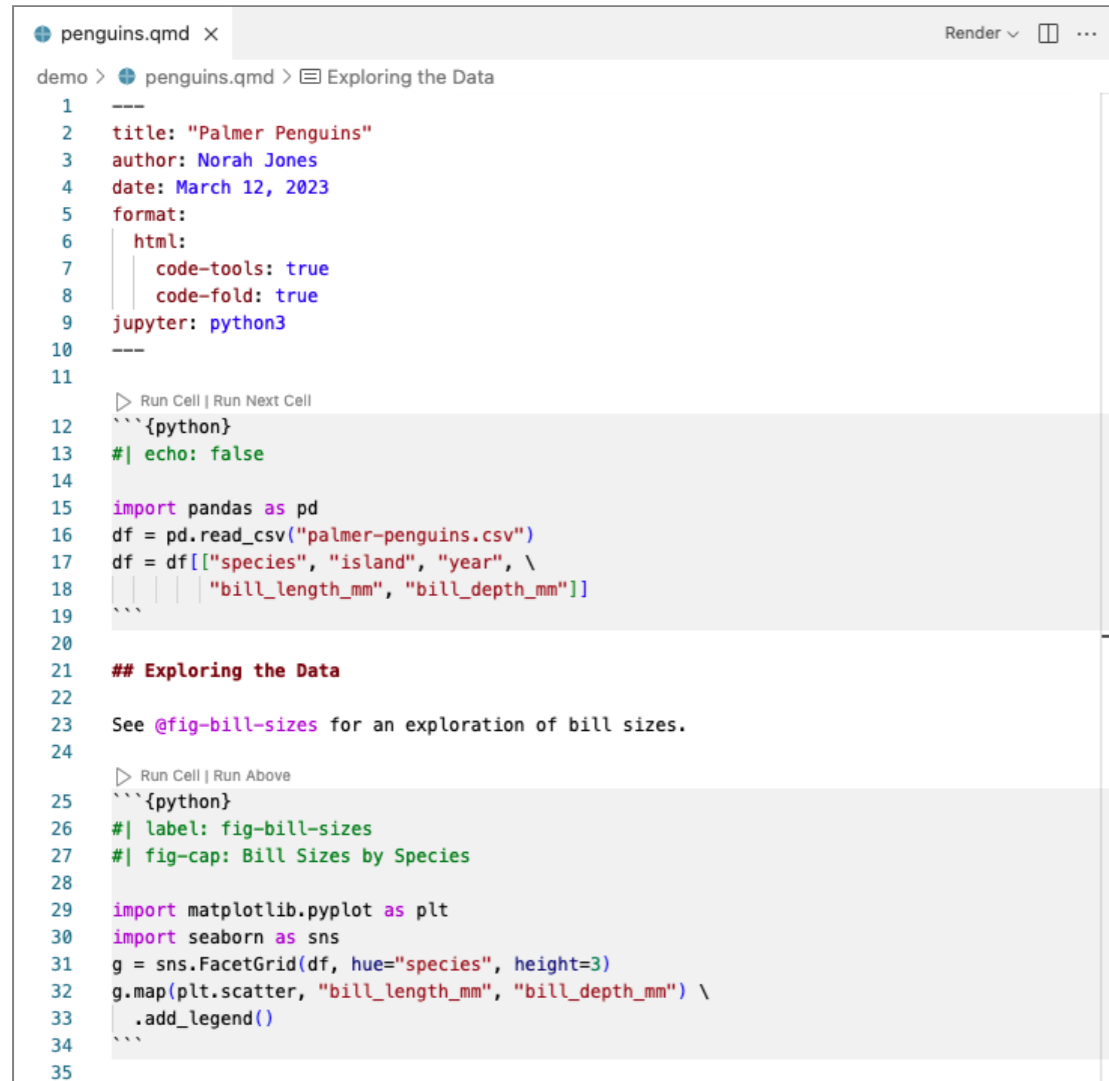
Example: Quarto `.qmd` Files

penguins.qmd

```
1 ---
2 title: "Palmer Penguins"
3 author: Norah Jones
4 date: March 12, 2023
5 format: html
6 jupyter: python3
7 ---
8
9 ```{python}
10 #| echo: false
11
12 import pandas as pd
13 df = pd.read_csv("palmer-penguins.csv")
14 df = df[["species", "island", "year", \
15         "bill_length_mm", "bill_depth_mm"]]
16 ```
17
18 ## Exploring the Data
19
20 See @fig-bill-sizes for an exploration of bill sizes.
21
22 ```{python}
23 #| label: fig-bill-sizes
24 #| fig-cap: Bill Sizes by Species
25
26 import matplotlib.pyplot as plt
27 import seaborn as sns
28 g = sns.FacetGrid(df, hue="species", height=3)
29 g.map(plt.scatter, "bill_length_mm", "bill_depth_mm") \
30     .add_legend()
31 ```
```

- Editable with any text editor (extensions for VS Code, Neovim, and Emacs)
- Cells always run in the same order
- Integrates well with version control
- Cache output with Jupyter Cache or Quarto freezer
- Lots of pros and cons visa-vi traditional `.ipynb` format/editors, use the right tool for each job

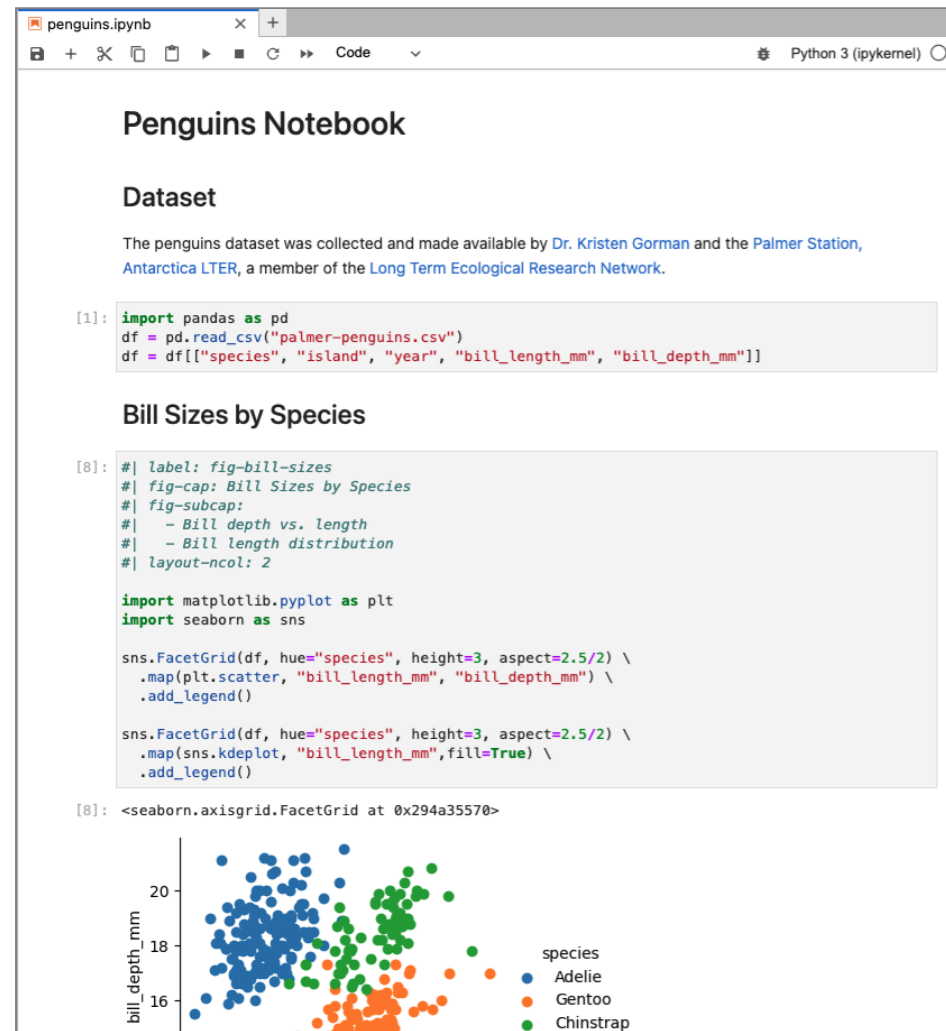
Quarto VS Code Extension



```
1  ---
2  title: "Palmer Penguins"
3  author: Norah Jones
4  date: March 12, 2023
5  format:
6    html:
7      code-tools: true
8      code-fold: true
9  jupyter: python3
10 ---
11
12 > Run Cell | Run Next Cell
13 ```{python}
14 #| echo: false
15
16 import pandas as pd
17 df = pd.read_csv("palmer-penguins.csv")
18 df = df[["species", "island", "year", \
19         "bill_length_mm", "bill_depth_mm"]]
20 ```
21
22 ## Exploring the Data
23
24 See @fig-bill-sizes for an exploration of bill sizes.
25
26 > Run Cell | Run Above
27 ```{python}
28 #| label: fig-bill-sizes
29 #| fig-cap: Bill Sizes by Species
30
31 import matplotlib.pyplot as plt
32 import seaborn as sns
33 g = sns.FacetGrid(df, hue="species", height=3)
34 g.map(plt.scatter, "bill_length_mm", "bill_depth_mm") \
35     .add_legend()
36 ```
```

- Render with integrated preview
- Syntax highlighting for markdown and embedded languages
- Completion for embedded languages (e.g. Python, R, Julia, LaTeX, etc.)
- Completion for YAML options
- Commands and key-bindings for running cells and selected line(s)
- Live preview for diagrams

Notebooks as a Content Source



- This notebook will not be the final document consumed by readers.
- Rather, it includes cells that will be incorporated by reference into another report or article.
- This is done using the **embed** shortcode, e.g.:

```
{{< embed penguins.ipynb#fig-bill-sizes >}}
```
- Links back to the original notebook are preserved.

Notebook Embedding

```
index.qmd U X Render v ...
demo > penguins > index.qmd > Penguin Species
1 ---
2 title: "Penguins"
3 author: "Norah Jones"
4 date: March 22, 2023
5 toc: true
6 ---
7
8 ## Introduction
9
10 The penguins dataset was collected and made available by [Dr. Kristen Gorman](https://www.uafl.edu/cfos/people/faculty/detail/kristen-gorman.php). It was collected at the Palmer [Long-Term Ecological Research Station](https://lter.palmer.edu/). The data was collected from 1977 to 2007, and is available in the penguins dataset.
11
12 ## Bills by Species
13
14 In @fig-bill-sizes we examine penguin bill dimensions by species.
15
16 {{< embed penguins.ipynb#fig-bill-sizes >}}
17
18 ## Bill Dimensions
19
20 The culmen is the upper ridge of a bird's bill. In the simplified penguins data, culmen length and depth are renamed as variables bill_length_mm and bill_depth_mm to be more intuitive.
21
22 For this penguin data, the culmen (bill) length and depth are measured as shown below:
23
24 ![(culmen_depth.png){width="80%"}
25
26 ## Penguin Species
27
28 There have been three penguin species observed on three islands in the Palmer Archipelago, Antarctica over a study period of three years:
29
30 ![(species.jpeg){width="80%"}
31
32
33
34
35
36
37
```

Notebook Embedding

```
index.qmd U X
demo > penguins > index.qmd > Penguin Species
1
2 title: "Penguins"
3 author: "Norah Jones"
4 date: March 22, 2023
5 toc: true
6
7
8 ## Introduction
9
10 The penguins dataset was collected and made available by [Dr. Kristen Gorman](https://www.uaf.edu/cfos/people/faculty/detail/kristen-gorman.php). It was collected at the Palmer [Long-Term Ecological Research Station](https://lter.palmer.edu/) located to the west of the Antarctic Peninsula extending from onshore to several hundred kilometers off shore.
11
12 ## Bills by Species
13
14 In @fig-bill-sizes we examine penguin bill dimensions by species.
15
16 {{< embed penguins.ipynb#fig-bill-sizes >}}
17
18 ## Bill Dimensions
19
20 The culmen is the upper ridge of a bird's bill. In the simplified penguins data, culmen length and depth are renamed as variables bill_length_mm and bill_depth_mm to be more intuitive.
21
22 For this penguin data, the culmen (bill) length and depth are measured as shown below:
23
24 ![(culmen_depth.png){width="80%"}
25
26 ## Penguin Species
27
28 There have been three penguin species observed on three islands in the Palmer Archipelago, Antarctica over a study period of three years:
29
30 ![(species.jpeg){width="80%"}
31
32
33
34
35
36
37
```

Penguins

AUTHOR
Norah Jones

PUBLISHED
March 22, 2023

On this page

[Introduction](#)
[Bills by Species](#)
[Bill Dimensions](#)

Notebooks

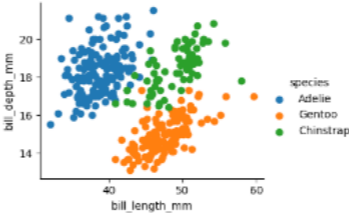
[penguins.ipynb](#)

Introduction

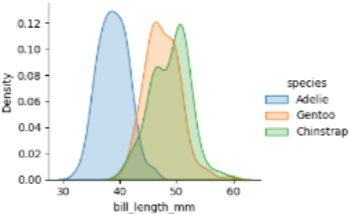
The penguins dataset was collected and made available by [Dr. Kristen Gorman](https://www.uaf.edu/cfos/people/faculty/detail/kristen-gorman.php). It was collected at the Palmer [Long-Term Ecological Research Station](https://lter.palmer.edu/) located to the west of the Antarctic Peninsula extending from onshore to several hundred kilometers off shore.

Bills by Species

In [Figure 1](#) we examine penguin bill dimensions by species:



(a) Bill depth vs. length



(b) Bill length distribution


Figure 1: Bill Sizes by Species

Source: penguins.ipynb

Bill Dimensions

The culmen length and depth are renamed as variables bill_length_mm and bill_depth_mm to be more intuitive.

For this penguin data, the culmen (bill) length and depth are measured as shown below:



Source: penguins.ipynb

Bill Dimensions

The culmen length and depth are renamed as variables bill_length_mm and bill_depth_mm to be more intuitive.

For this penguin data, the culmen (bill) length and depth are measured as shown below:

Notebook Embedding

```
index.qmd U X Render v ...
demo > penguins > index.qmd > Penguin Species
1 ---
2 title: "Penguins"
3 author: "Norah Jones"
4 date: March 22, 2023
5 toc: true
6 ---
7
8 ## Introduction
9
10 The penguins dataset was collected and made available by [Dr. Kristen Gorman](https://www.uaf.edu/cfos/people/faculty/detail/kristen-gorman.php). It was collected at the Palmer [Long-Term Ecological Research Station](https://www.lter.psu.edu/), a field station on the west coast of Antarctica, onshore to the west of the Antarctic Peninsula.
11
12 ## Bills by Species
13
14 In @fig-bill-sizes we examine penguin bill dimensions by species.
15
16 {{< embed penguins.ipynb#fig-bill-sizes >}}
17
18 ## Bill Dimensions
19
20 The culmen is the upper ridge of a bird's bill. In the simplified penguins data, culmen length and depth are renamed as variables bill_length_mm and bill_depth_mm to be more intuitive.
21
22 For this penguin data, the culmen (bill) length and depth are measured as shown below:
23
24 ![(culmen_depth.png){width="80%"}
25
26 ## Penguin Species
27
28 There have been three penguin species observed on three islands in the Palmer Archipelago, Antarctica over a study period of three years:
29
30 ![(species.jpeg){width="80%"}
31
32
33
34
35
36
37
```

penguins.ipynb

Download Notebook

Penguins Notebook

Dataset

The penguins dataset was collected and made available by [Dr. Kristen Gorman](#) and the [Palmer Station, Antarctica LTER](#), a member of the [Long Term Ecological Research Network](#).

In [1]:

```
import pandas as pd
df = pd.read_csv("palmer-penguins.csv")
df = df[["species", "island", "year", "bill_length_mm", "bill_depth_mm"]]
```

Bill Sizes by Species

In [8]:

```
import matplotlib.pyplot as plt
import seaborn as sns

sns.FacetGrid(df, hue="species", height=3, aspect=2.5/2) \
    .map(plt.scatter, "bill_length_mm", "bill_depth_mm") \
    .add_legend()

sns.FacetGrid(df, hue="species", height=3, aspect=2.5/2) \
    .map(sns.kdeplot, "bill_length_mm", fill=True) \
    .add_legend()
```

(a) Bill depth vs. length

(b) Bill length distribution

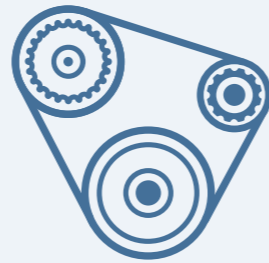
Figure 1: Bill Sizes by Species

Notebooks Now

<https://data.agu.org/notebooks-now/>



- Collaboration among participants in the open-science community, scientific publishers, and the developers of Jupyter Book, Myst-JS, and Quarto to create a standard for including notebooks in scientific publications.
- Aim is to define a standard for scholarly articles that include notebooks, enabling them to be considered as part of peer review and included in archives.



Under the Hood

Jupyter Kernels

- Quarto executes code cells (whether in `.ipynb` or `.qmd` files) using Jupyter
- All Jupyter kernels are supported (Python and Julia are the most widely used)
- To preserve interactive response times, kernels are kept alive for up to 5 minutes across renders (especially important for Julia)

Jupyter Kernels

- Quarto executes code cells (whether in `.ipynb` or `.qmd` files) using Jupyter
- All Jupyter kernels are supported (Python and Julia are the most widely used)
- To preserve interactive response times, kernels are kept alive for up to 5 minutes across renders (especially important for Julia)

i Quarto execution is extensible so other engines besides Jupyter are possible (e.g. Knitr engine for compatibility with R Markdown documents). Other engines may be added in the future as the landscape evolves.

Cell Execution

Cell Execution

- Authoring in `.ipynb` enables you to control exactly when code execution occurs (and cache results in the notebook)

Cell Execution

- Authoring in `.ipynb` enables you to control exactly when code execution occurs (and cache results in the notebook)
- Authoring in `.qmd` will execute cells on every render (but see below for caching strategies)

Cell Execution

- Authoring in `.ipynb` enables you to control exactly when code execution occurs (and cache results in the notebook)
- Authoring in `.qmd` will execute cells on every render (but see below for caching strategies)
- [Jupyter Cache](#) (from the [Executable Book Project](#)) provides caching of cell outputs for a document (change to one cell triggers re-rendering of all cells)

Cell Execution

- Authoring in `.ipynb` enables you to control exactly when code execution occurs (and cache results in the notebook)
- Authoring in `.qmd` will execute cells on every render (but see below for caching strategies)
- [Jupyter Cache](#) (from the [Executable Book Project](#)) provides caching of cell outputs for a document (change to one cell triggers re-rendering of all cells)
- Quarto's [Freeze](#) feature enables you to permanently save and re-use computational outputs (which are updated only when input files change)

Filters

- Filters transform the document AST before final rendering
- Can be used to modify, remove, or generate content
- Can include target format specific logic / output
- Example: Use the [panflute](#) library to increase the level of headings in a document.

Filters

- Filters transform the document AST before final rendering
- Can be used to modify, remove, or generate content
- Can include target format specific logic / output
- Example: Use the **panflute** library to increase the level of headings in a document.

Terminal

```
$ quarto render nb.ipynb --filter headers.py
```

headers.py

```
1 from panflute import *
2
3 def increase_header_level(elem, doc):
4     if type(elem) == Header:
5         if elem.level < 6:
6             elem.level += 1
7
8 def main():
9     return run_filter(increase_header_level)
10
11 if __name__ == "__main__":
12     main()
```

Filters

- Filters transform the document AST before final rendering
- Can be used to modify, remove, or generate content
- Can include target format specific logic / output
- Example: Use the **panflute** library to increase the level of headings in a document.

Terminal

```
$ quarto render nb.ipynb --filter headers.py
```

headers.py

```
1 from panflute import *
2
3 def increase_header_level(elem, doc):
4     if type(elem) == Header:
5         if elem.level < 6:
6             elem.level += 1
7
8 def main():
9     return run_filter(increase_header_level)
10
11 if __name__ == "__main__":
12     main()
```

What Can Filters Do?

- Embedded languages (e.g. PlantUML, GraphViz)
- Macro substitution (environment variables, config files, etc.)
- Cross references and citations
- Image conversion and filtering
- Advanced formatting (e.g. callouts)

What Can Filters Do?

- Embedded languages (e.g. PlantUML, GraphViz)
- Macro substitution (environment variables, config files, etc.)
- Cross references and citations
- Image conversion and filtering
- Advanced formatting (e.g. callouts)

Quarto includes dozens of filters that implement its core functionality, but the system is open so you can add whatever features you require.

Filter Examples

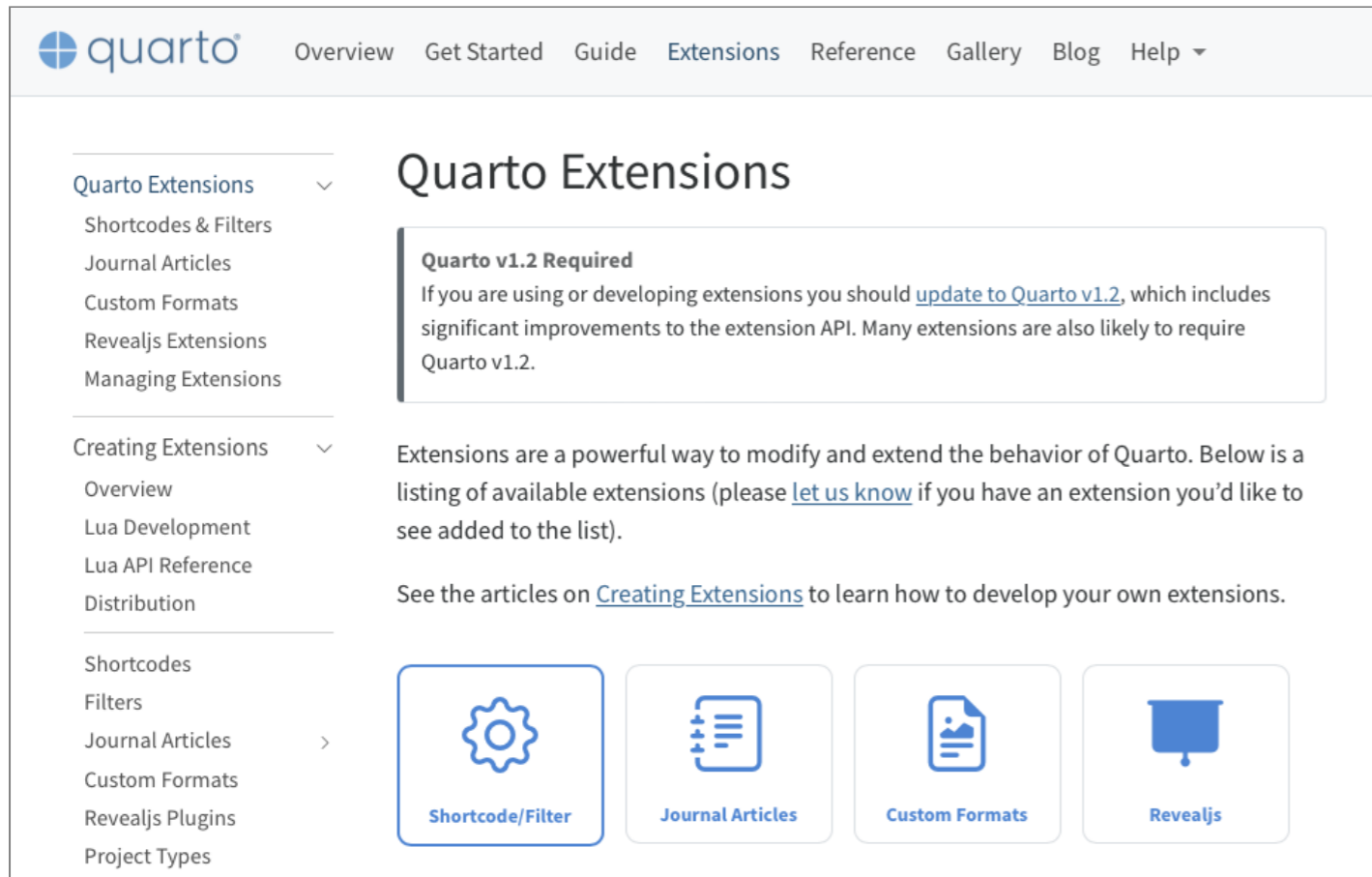
| Filter | Description |
|--------------------------------|---|
| <code>lightbox</code> | Create lightbox treatments for images in your HTML documents. |
| <code>molstar</code> | Shortcode to embed proteins and trajectories with Mol* . |
| <code>social-share</code> | Add buttons to share articles on various social media platforms. |
| <code>latex-environment</code> | Output divs as custom LaTeX environments. |
| <code>qrcode</code> | Shortcode to embed QR codes using qrcodejs . |
| <code>code-visibility</code> | Filter code and stream output included within a document. |
| <code>authors-block</code> | Add author-related header block when rendering docx-documents. |

Writing Filters

- **pandocfilters** Python library from the creator of Pandoc
- **panflute** Python library with improved API and more batteries included
- **Lua Filters** Pandoc includes an embedded Lua interpreter for fast, zero-dependency filters
- **JSON Filters** Write filters in any language via JSON representation over stdin/stdout

Extensions

<https://quarto.org/docs/extensions/>



The screenshot shows the Quarto website's 'Extensions' page. The top navigation bar includes links for Overview, Get Started, Guide, Extensions (active), Reference, Gallery, Blog, and Help. The left sidebar lists categories: Quarto Extensions (with a dropdown arrow), Creating Extensions (with a dropdown arrow), Shortcodes, Filters, Journal Articles, Custom Formats, Revealjs Plugins, and Project Types. The main content area is titled 'Quarto Extensions' and features a 'Quarto v1.2 Required' warning box. Below this, it explains that extensions are used to modify Quarto's behavior and provides a link to 'let us know' if users have extensions to add. It also directs users to 'Creating Extensions' articles for development guidance. At the bottom, four icons represent different extension types: Shortcode/Filter (gear icon), Journal Articles (document icon), Custom Formats (document with image icon), and Revealjs (presentation icon).

quarto Overview Get Started Guide Extensions Reference Gallery Blog Help ▾

Quarto Extensions ▾

- Shortcodes & Filters
- Journal Articles
- Custom Formats
- Revealjs Extensions
- Managing Extensions

Quarto v1.2 Required

If you are using or developing extensions you should [update to Quarto v1.2](#), which includes significant improvements to the extension API. Many extensions are also likely to require Quarto v1.2.

Creating Extensions ▾

- Overview
- Lua Development
- Lua API Reference
- Distribution

Extensions are a powerful way to modify and extend the behavior of Quarto. Below is a listing of available extensions (please [let us know](#) if you have an extension you'd like to see added to the list).

See the articles on [Creating Extensions](#) to learn how to develop your own extensions.

Shortcodes

Filters

Journal Articles >

Custom Formats

Revealjs Plugins

Project Types

Shortcode/Filter

Journal Articles

Custom Formats

Revealjs

- Filters
- Shortcodes (macros)
- Custom Formats
- Revealjs Plugins
- Project Types
- Project Templates

Integration w/ nbdev

nbdev+Quarto: A new secret weapon for productivity

Our favorite tool for software engineering productivity—nbdev, now re-written with Quarto

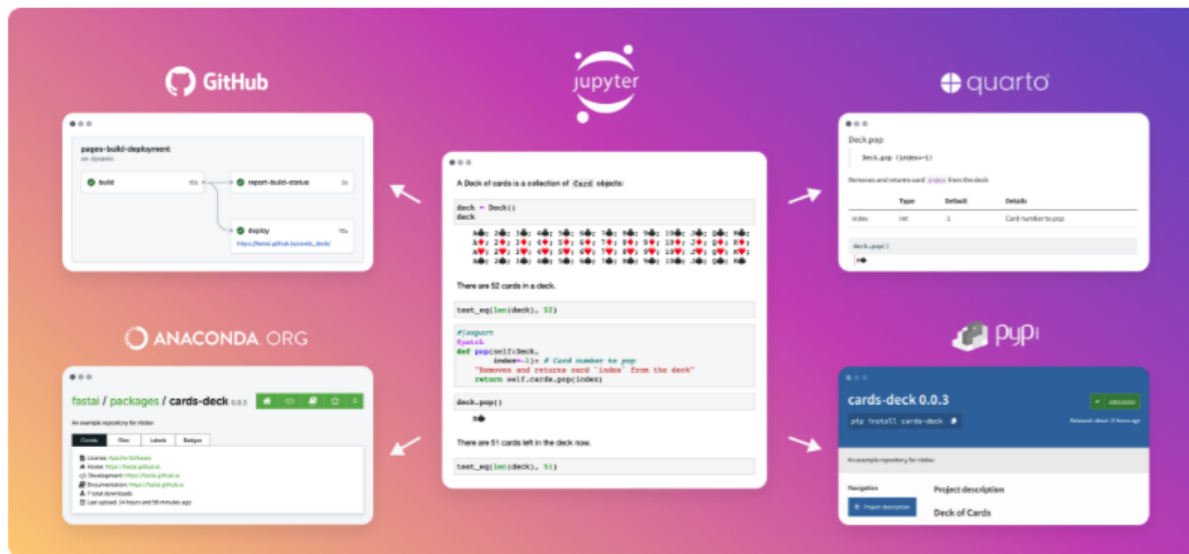
TECHNICAL

AUTHOR

Hamel Husain and Jeremy Howard

PUBLISHED

July 28, 2022



A single notebook can create a python module, tests, CI, pypi/conda packages, and more.

<https://nbdev.fast.ai>

- Interactively develop Python packages within Jupyter, including embedded tests/docs, CI, pypi and conda publishing
- Version 2 of **nbdev** uses Quarto to produce documentation websites

Thank You!

Slides: <https://jjallaire.quarto.pub/data-council-2023/>

Resources

| | |
|-----------------|---|
| Getting Started | https://quarto.org/ |
| User Guide | https://quarto.org/docs/guide/ |
| Extensions | https://quarto.org/docs/extensions/ |
| Awesome Quarto | https://github.com/mcanouil/awesome-quarto |

Questions?