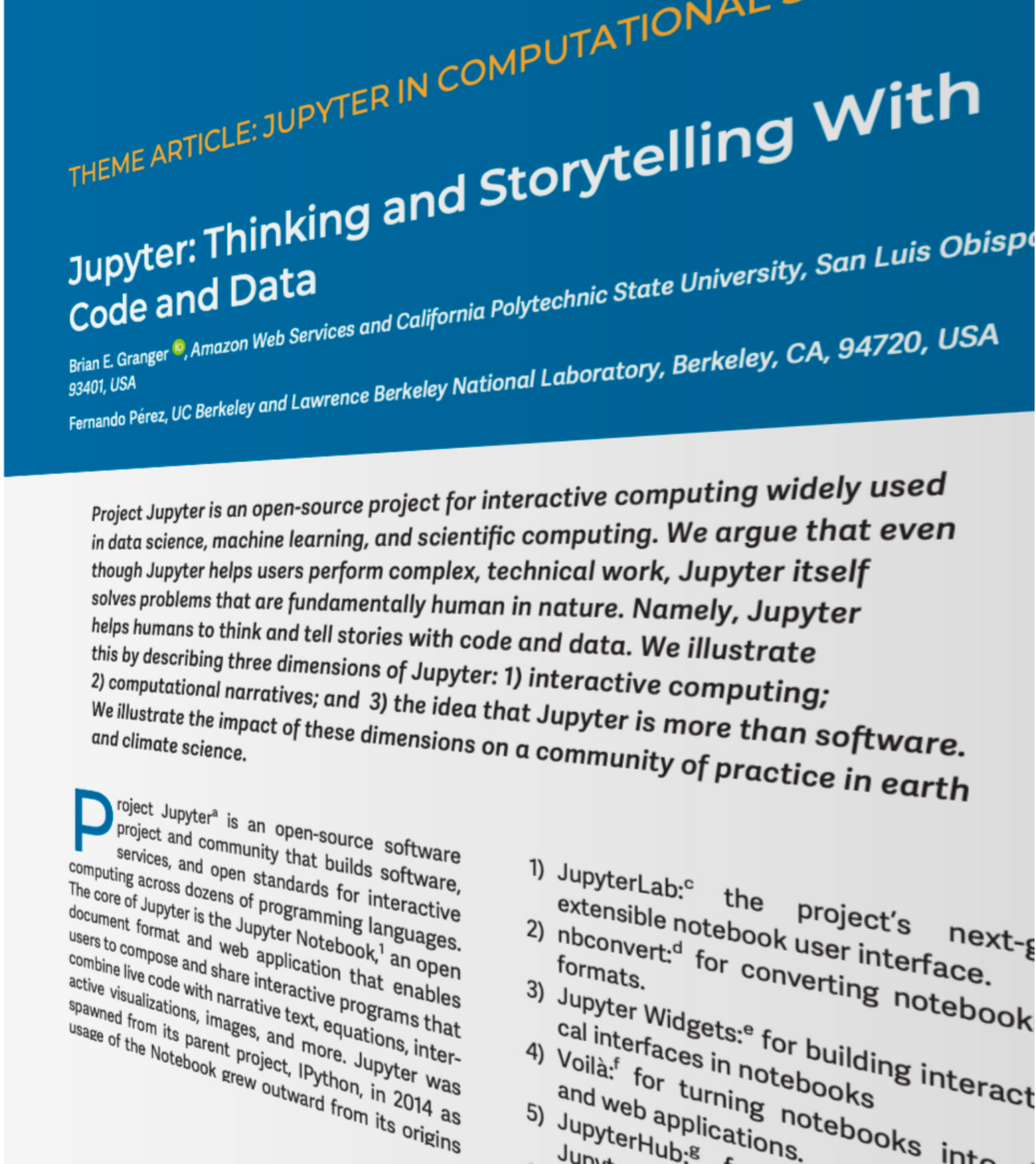
Publishing Jupyter Notebooks with Quarto

J.J. Allaire — Data Council 2023

posit

> “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



THEME ARTICLE: JUPYTER IN COMPUTATIONAL

Jupyter: Thinking and Storytelling With Code and Data

Brian E. Granger, 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.

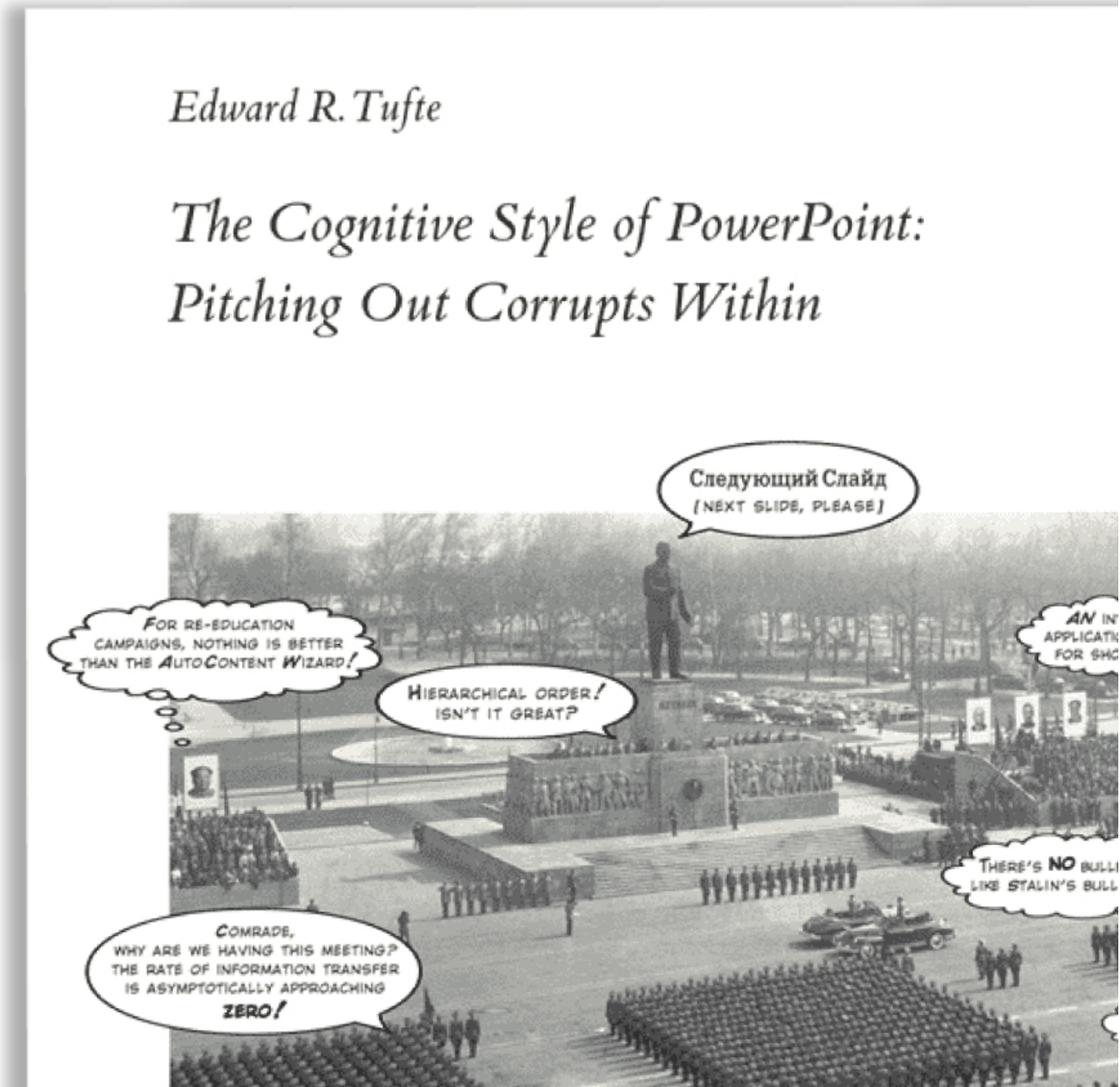
Project Jupyter[a] is an open-source software project and community that builds software, services, and open standards for interactive computing across dozens of programming languages. The core of Jupyter is the Jupyter Notebook,[1] an open document format and web application that enables users to compose 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-g extensible notebook user interface.
2) nbconvert:[d] for converting notebook formats.
3) Jupyter Widgets:[e] for building interact cal interfaces in notebooks
4) Voilà:[f] for turning notebooks into and web applications.
5) JupyterHub:[g]

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



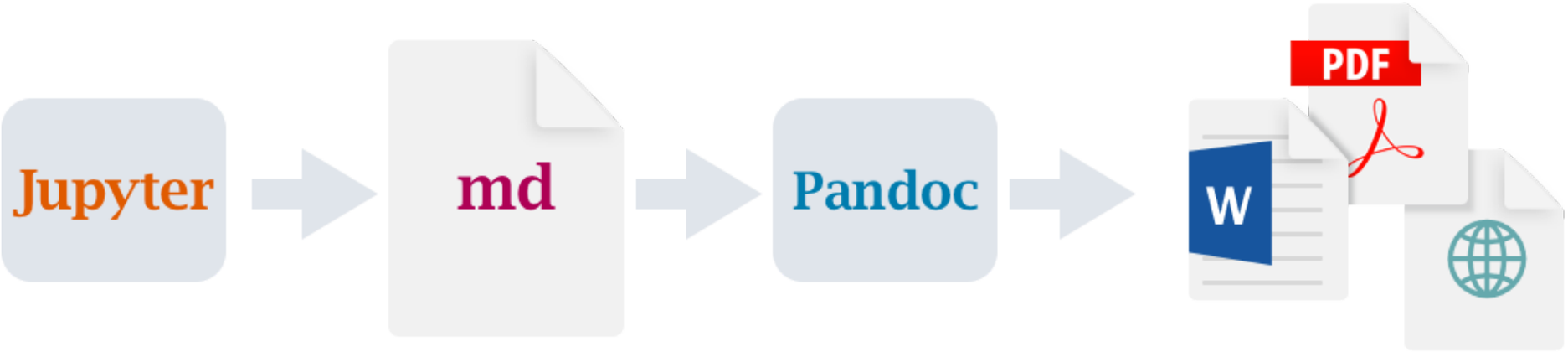
Some History

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

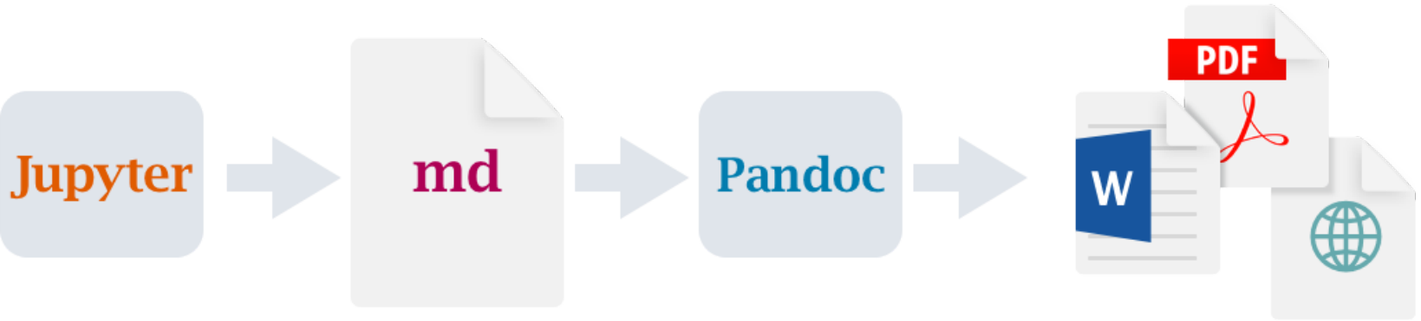
Variety of tools available: nbconvert, Jupyter Book, Myst-JS, Quarto

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:

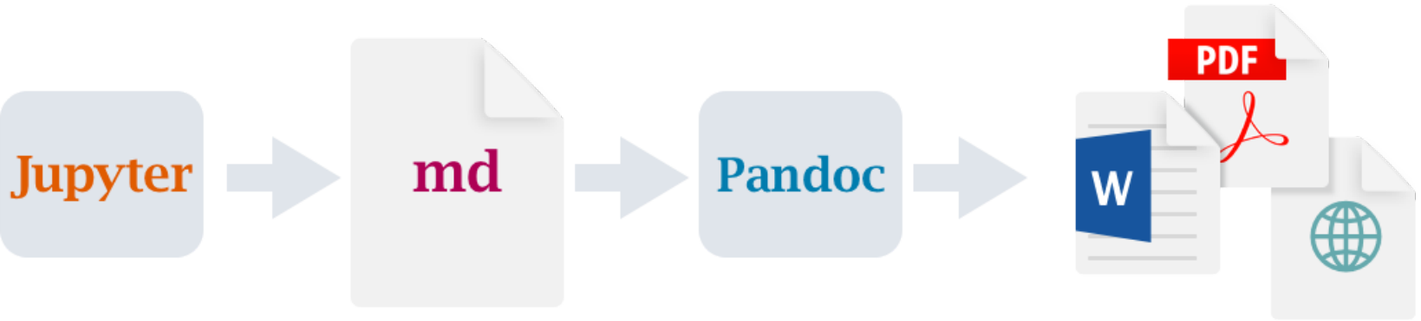
```
$ quarto render notebook.ipynb
```



How Does Quarto Work?

Quarto *renders* notebooks into various output formats:

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

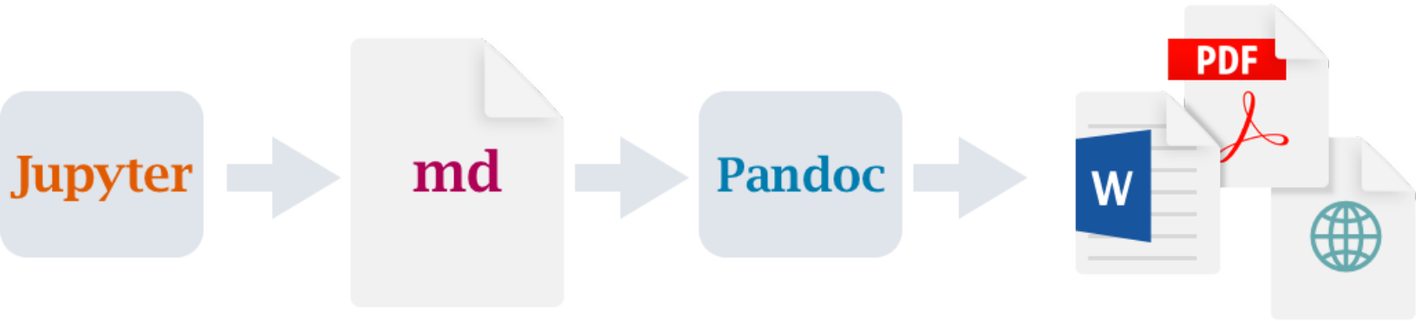


- Start with a Jupyter notebook (executed or not)

How Does Quarto Work?

Quarto *renders* notebooks into various output formats:

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

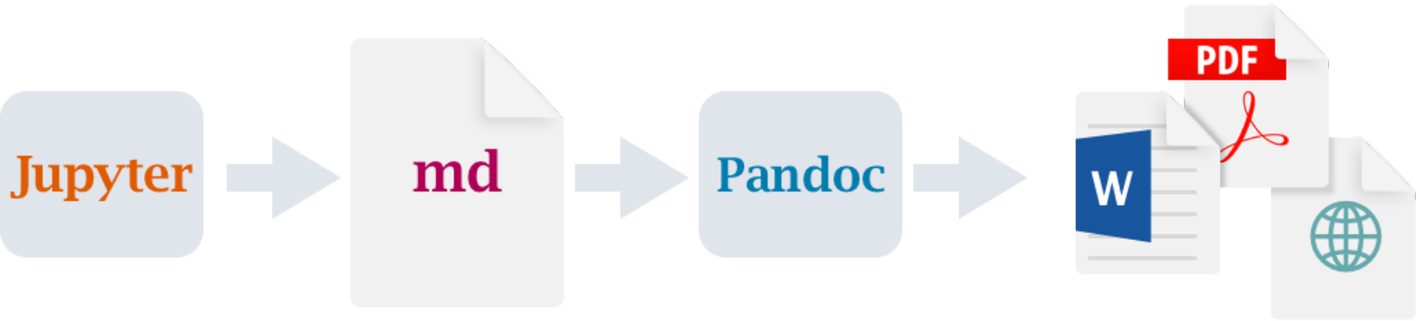


- Start with a Jupyter notebook (executed or not)
- Add document and cell level output options using YAML

How Does Quarto Work?

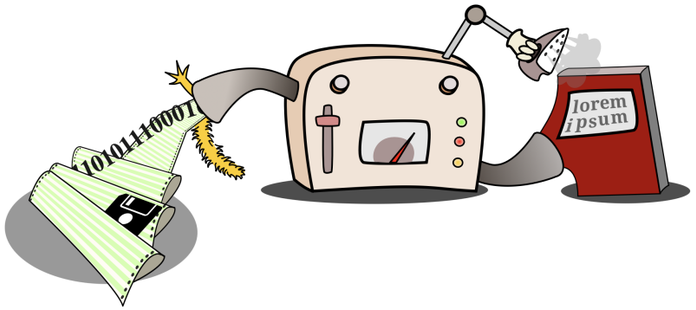
Quarto *renders* notebooks into various output formats:

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



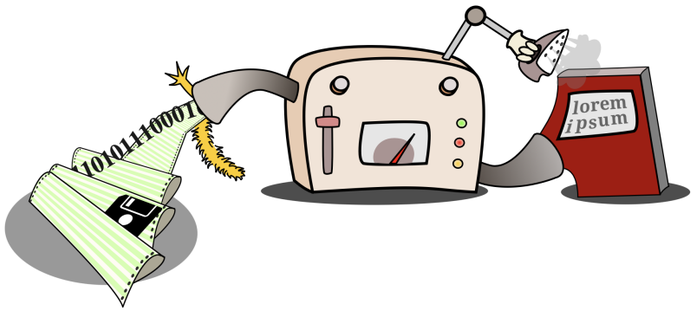
- Start with a Jupyter notebook (executed or not)
- Add document and cell level output options using YAML
- `quarto render` to the desired output format

Rendering with Pandoc



Universal document converter

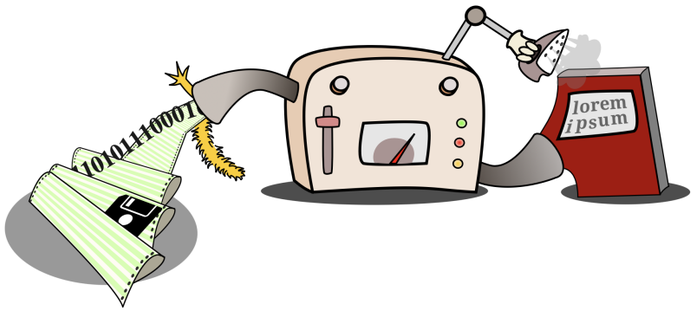
Rendering with Pandoc



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)

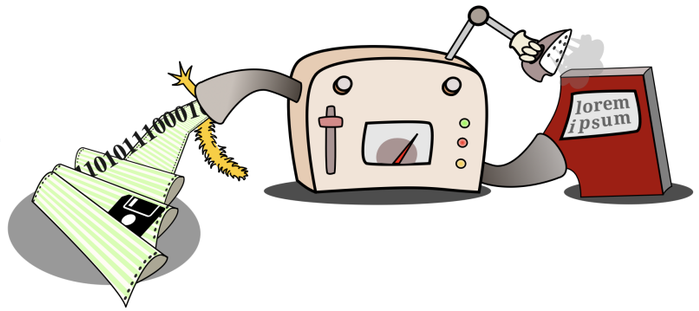
Rendering with Pandoc



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

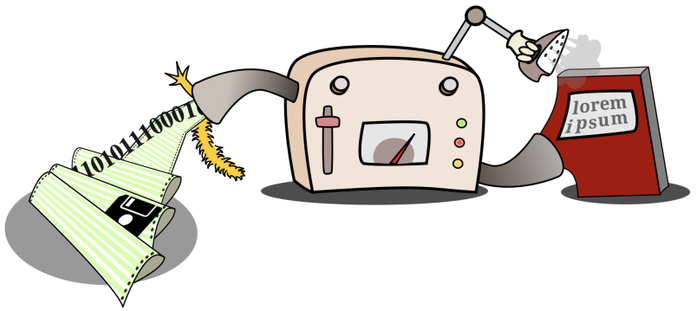
Rendering with Pandoc



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)

basics.ipynb

Markdown Python 3 (ipykernel)

Palmer Penguins

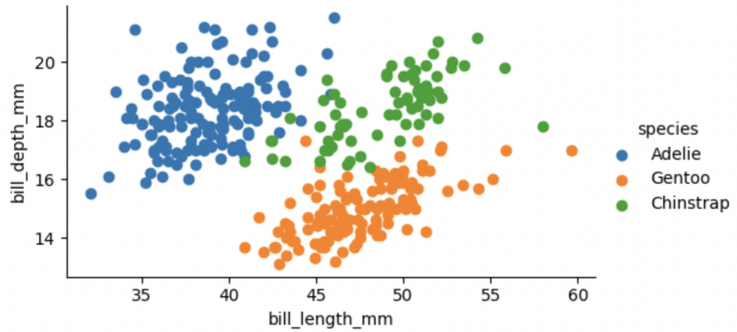
[9]:
```
import pandas as pd
df = pd.read_csv("palmer-penguins.csv")
df = df[["species", "island", "year", "bill_length_mm", "bill_depth_mm"]]
df.head(3)
```

[9]:

	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

[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>
```



Render Notebook to HTML (default options)

basics.ipynb

Markdown Python 3 (ipykernel)

Palmer Penguins

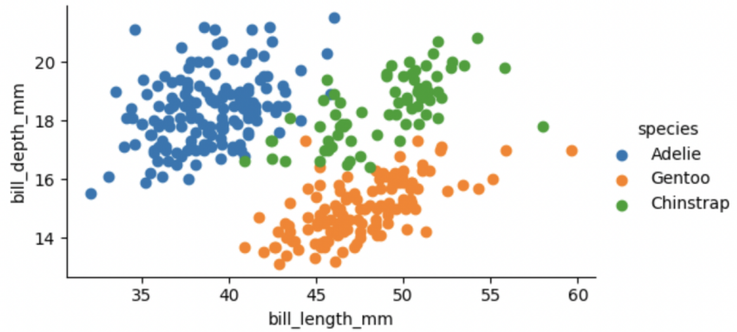
[9]:
```
import pandas as pd
df = pd.read_csv("palmer-penguins.csv")
df = df[["species", "island", "year", "bill_length_mm", "bill_depth_mm"]]
df.head(3)
```

[9]:

	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

[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>
```

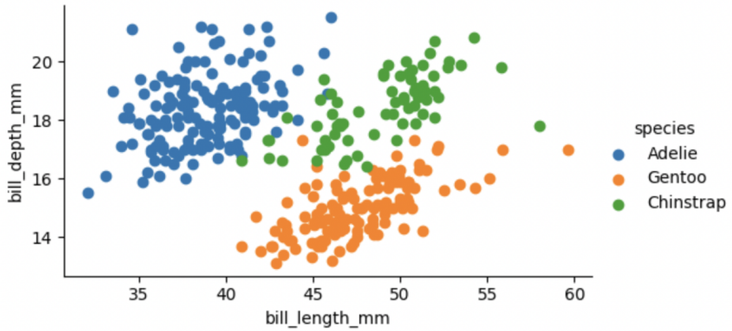


Palmer Penguins

```
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)

doc-options.ipynb

Python 3 (ipykernel)

```
---
theme: minty
highlight-style: atom-one
comments:
  hypothesis: true
---
```

Palmer

```
---
theme: mint
highlight-s
comments:
  hypothesi
---
```

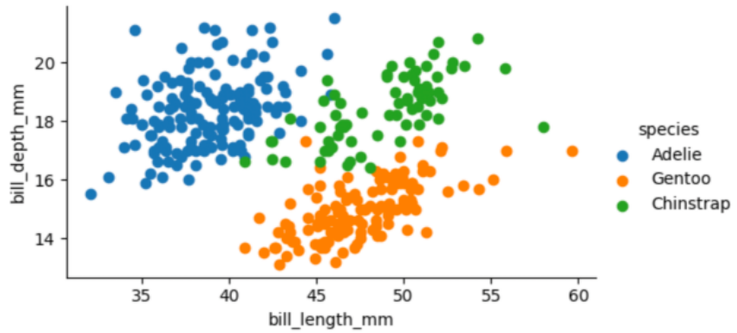
[9]:
```
import pandas as pd
df = pd.read_csv("palmer-penguins.csv")
df = df[["species", "island", "year", "bill_length_mm", "bill_depth_mm"]]
df.head(3)
```

[9]:

	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

[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>
```



Render Notebook to HTML (document level options)

doc-options.ipynb

Python 3 (ipykernel)

```
---
theme: minty
highlight-style: atom-one
comments:
  hypothesis: true
---
```

Palmer

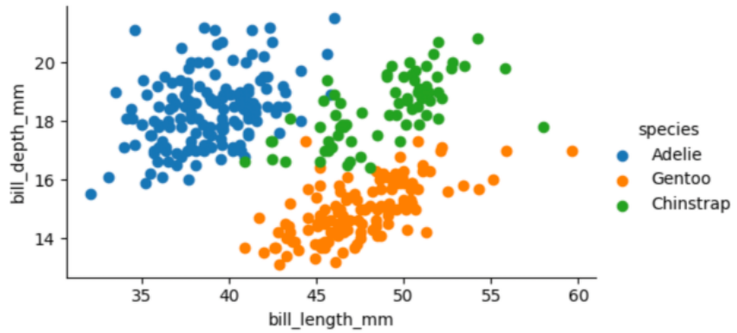
[9]:
```
import pandas as pd
df = pd.read_csv("palmer-penguins.csv")
df = df[["species", "island", "year", "bill_length_mm", "bill_depth_mm"]]
df.head(3)
```

[9]:

	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

[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>

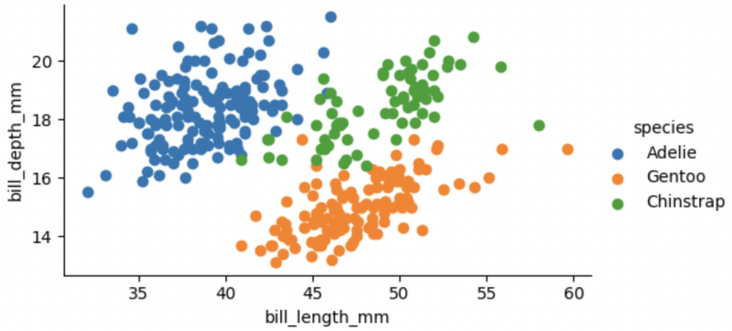


Palmer Penguins

```
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 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"]]
```

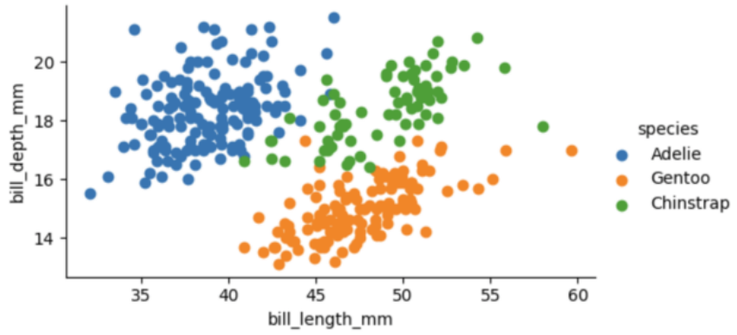
Exploring

See @fig-bill-si

```
#| label: fig-bill-sizes
#| fig-cap: "Bill Sizes by Species"
import matplotlib.pyplot as plt
import seaborn as sns
```

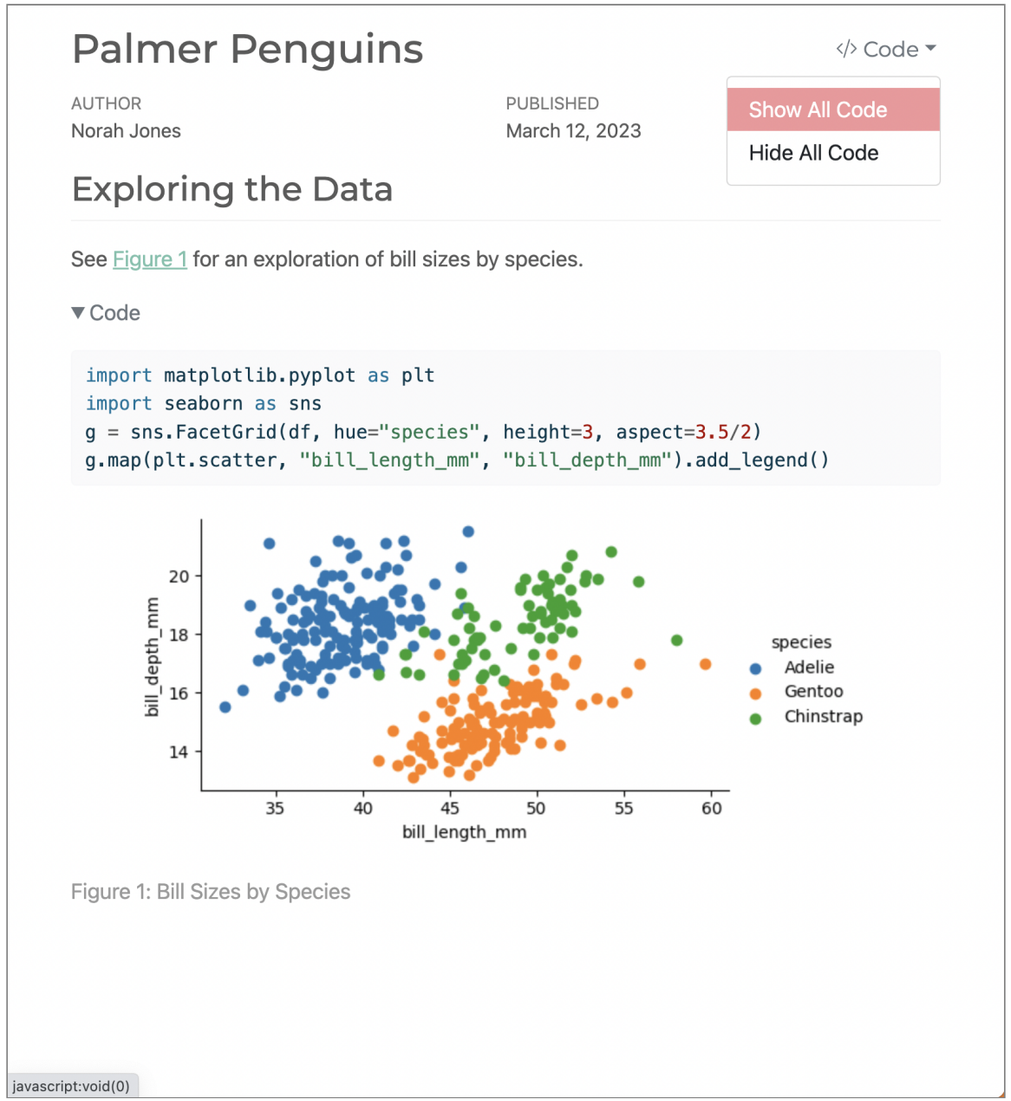
[2]:
```
#| label: fig
#| fig-cap: "
import matplotlib.pyplot as plt
import seaborn as sns
g = sns.FacetGrid(df, hue="species", height=3, aspect=3.5/2)
g.map(plt.scatter, "bill_length_mm", "bill_depth_mm").add_legend()
```

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



Render Notebook to HTML (document and cell level options)

Render Notebook to HTML (document and cell level options)



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

msword.ipynb

Python 3 (ipykernel)

```
---
author: "Norah Jones"
date: "March 12, 2023"
format: docx
---
```

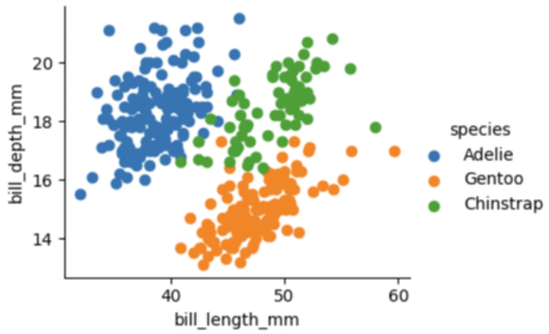
Palmer

[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 the Data

[2]:
```
#| layout-ncol: 2
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()
```

[2]:
```
<seaborn.axisgrid.FacetGrid at 0x103dc0460>
```



Render Notebook to MS Word

msword.ipynb

Python 3 (ipykernel)

```
---
author: "Norah Jones"
date: "March 12, 2023"
format: docx
---
```

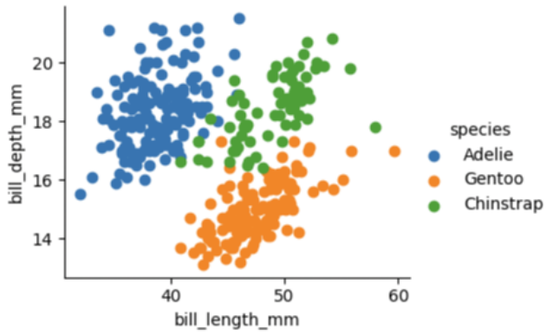
Palmer

[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 the Data

[2]:
```
#| layout-ncol: 2
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()
```

[2]:
```
<seaborn.axisgrid.FacetGrid at 0x103dc0460>
```



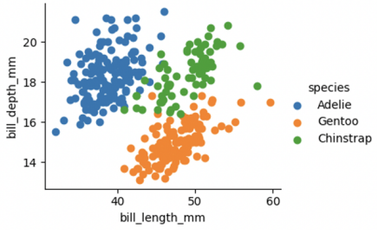
Palmer Penguins

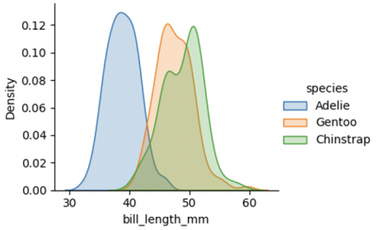
Norah Jones

2023-03-12

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 PDF

pdf.ipynb

Python 3 (ipykernel)

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

Palmer

[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

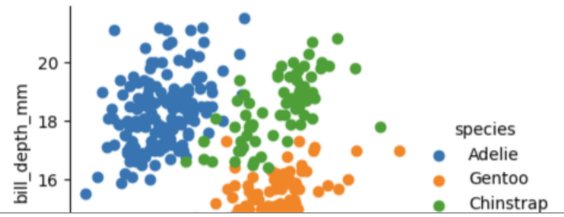
Here we will explore the use of the Palmer Penguins dataset from @palmerpenguins. The dataset contains different body
Antarctica.

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

Exploring

[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()
```

[7]: <seaborn.axisgrid.FacetGrid at 0x16cbd4460>



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

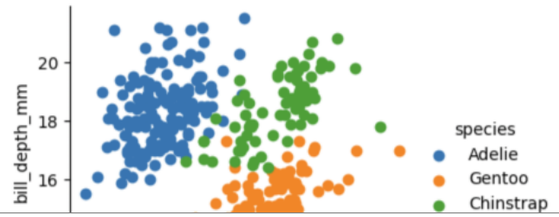
Here we will explore the use of the Palmer Penguins dataset from @palmerpenguins. The dataset contains different body m
Antarctica.

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

Exploring

[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()
```

[7]:
```
<seaborn.axisgrid.FacetGrid at 0x16cbd4460>
```



Palmer Penguins

Norah Jones

2023-03-22

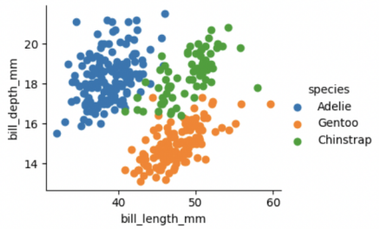
Overview

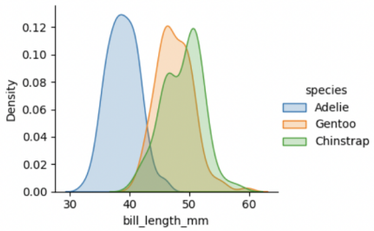
Here we will explore the use of the Palmer Penguins dataset from Horst, Hill, and Gorman (2020). The dataset contains different body measurements for three species of penguins from three islands in the Palmer Archipelago, Antarctica.

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)

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

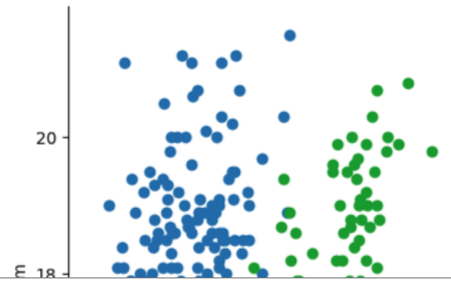
Palmer

[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)

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

Palmer

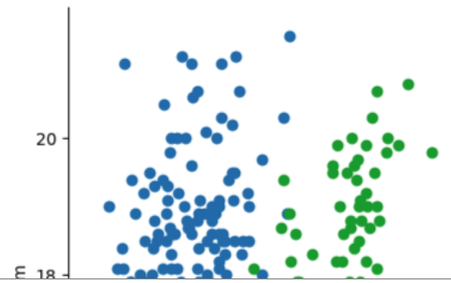
```
---
author: "No
date: "Marc
format:
  revealjs:
    slide-n
    logo: p
---
```

[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

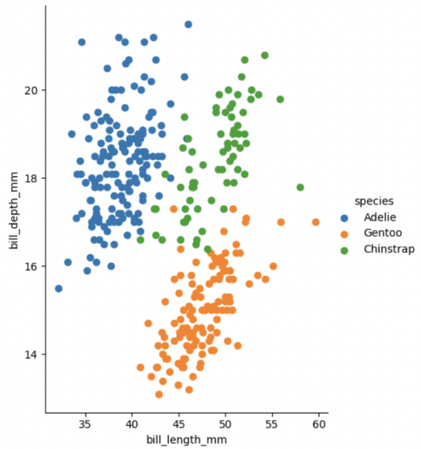
[7]:
```
#| layout-ncol: 2
import matplotlib.pyplot as plt
import seaborn as sns

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()
```

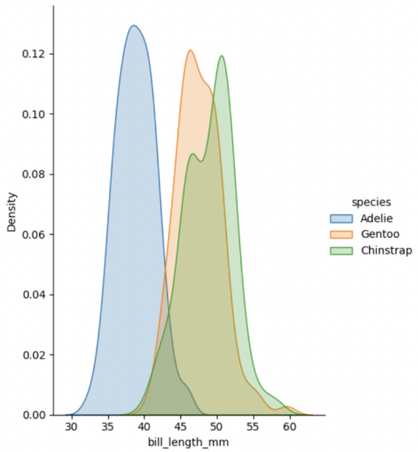
[7]:
```
<seaborn.axisgrid.FacetGrid at 0x137f56b60>
```



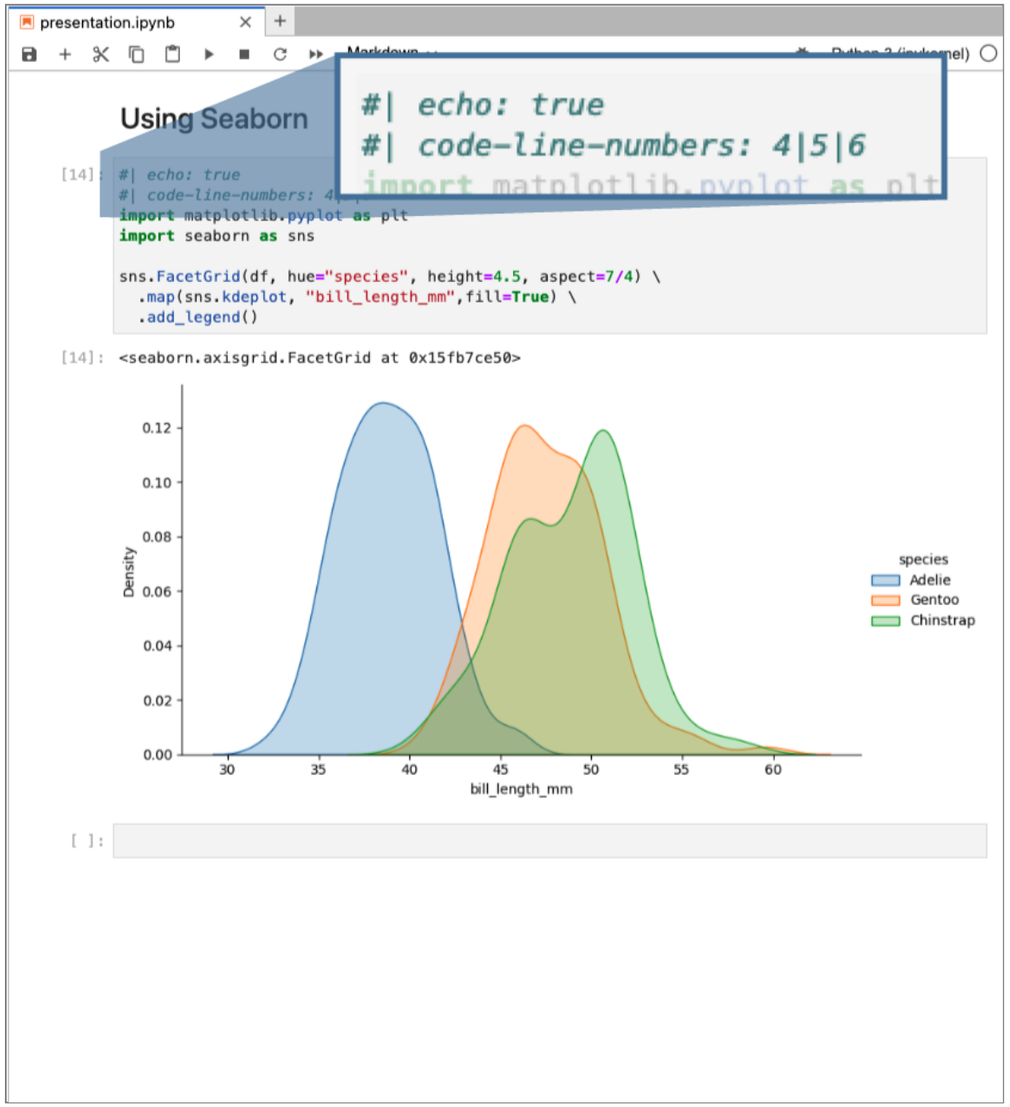
2 / 3

Exploring the Data

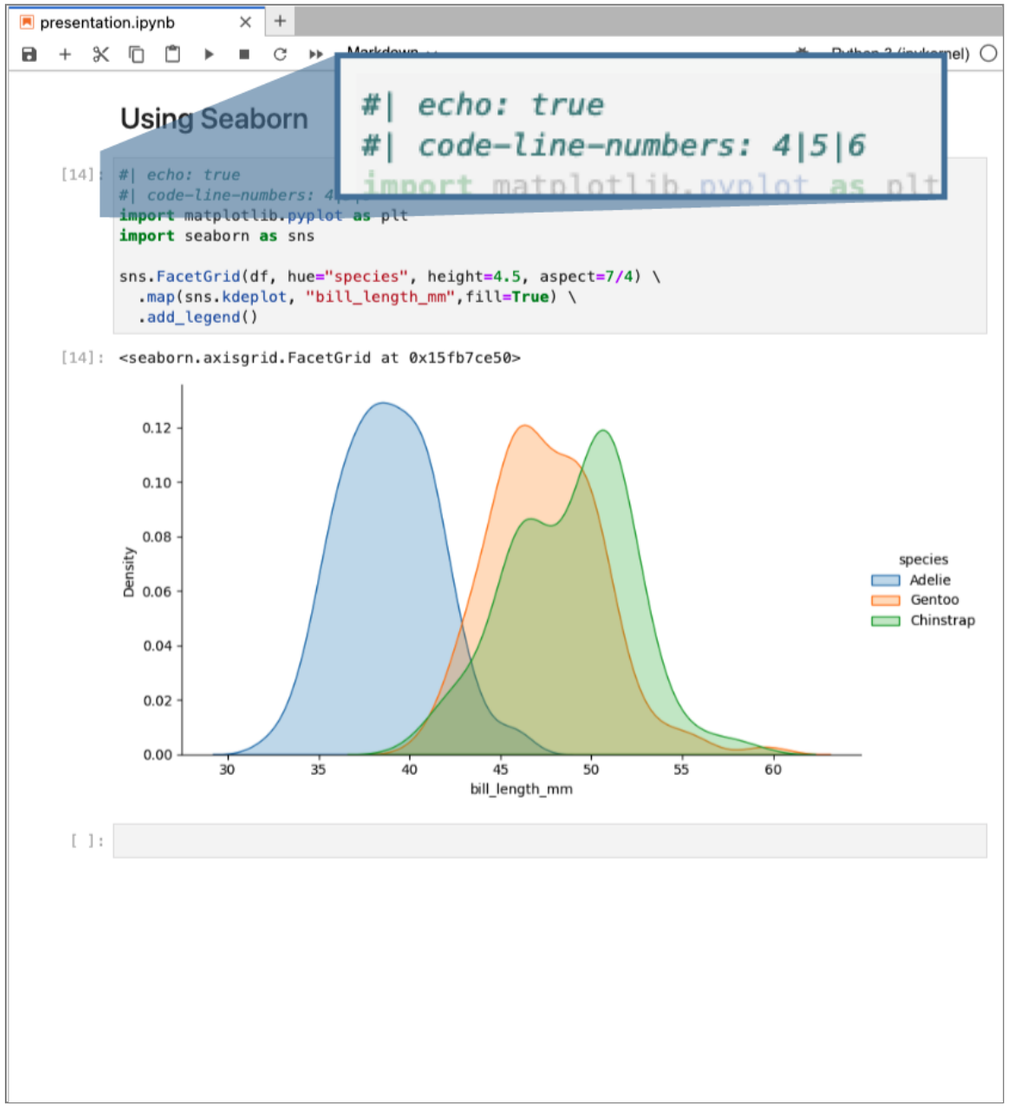


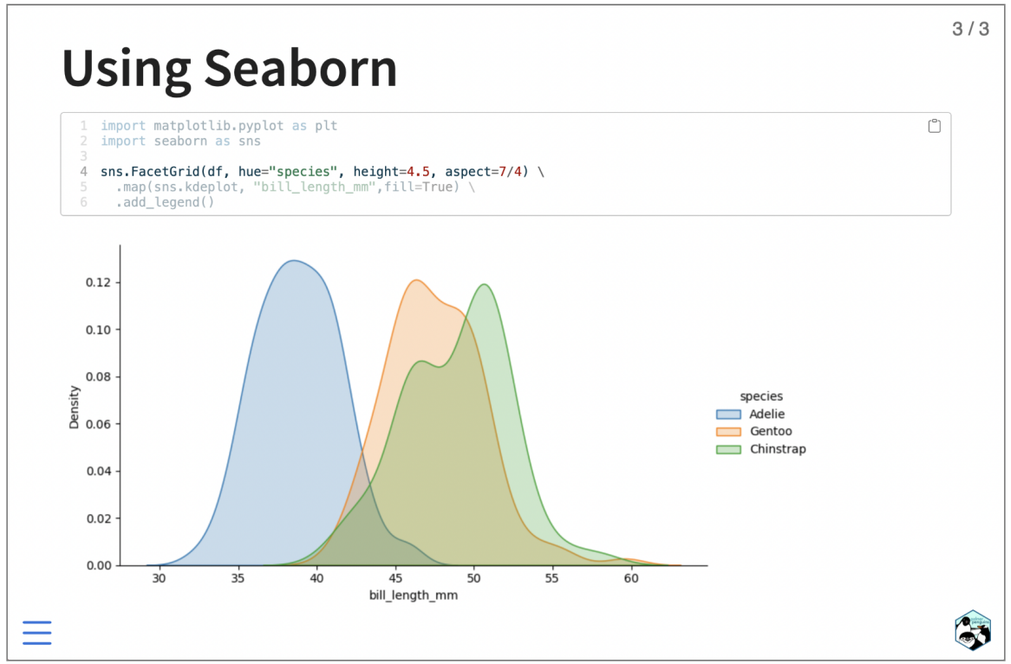


Render Notebook to Revealjs (show code with line by line highlighting)

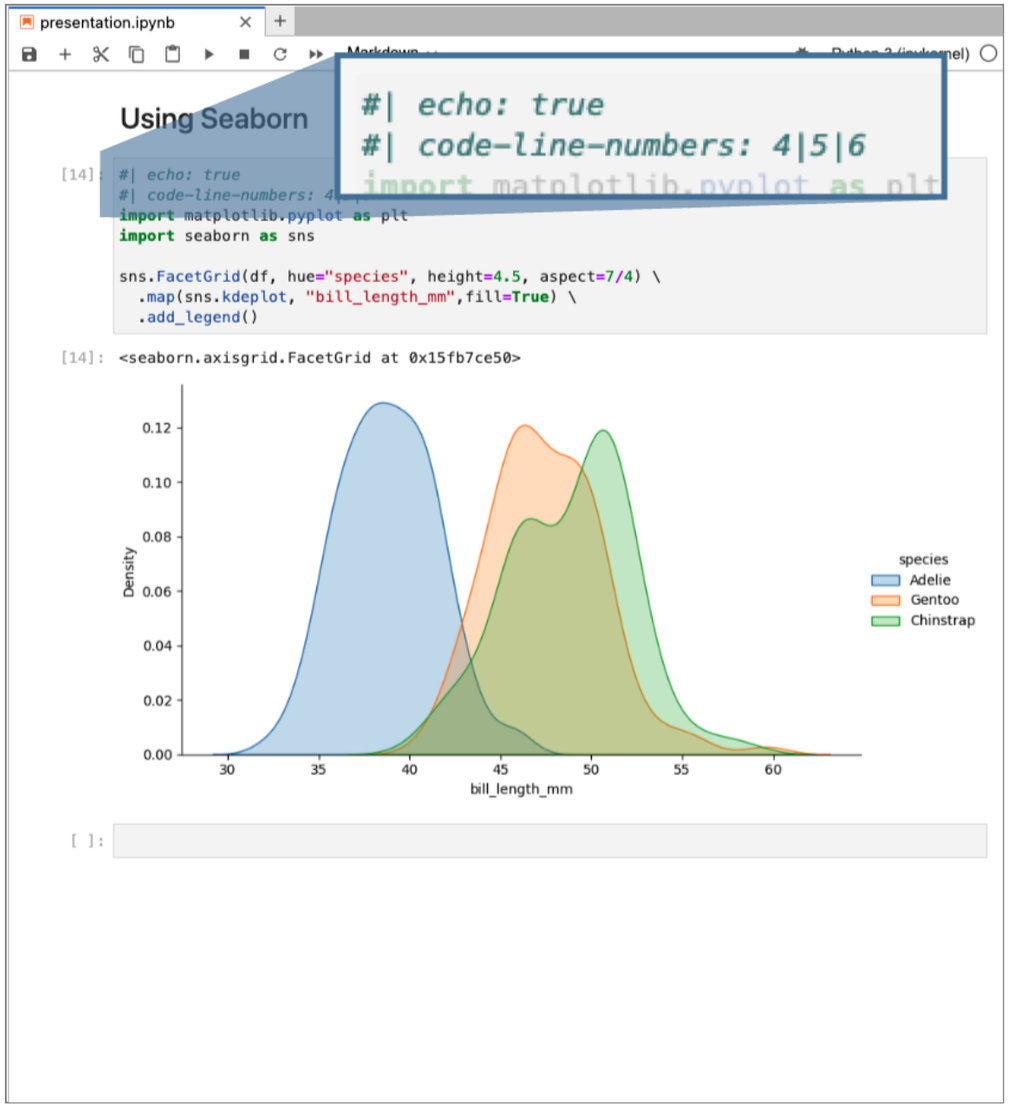


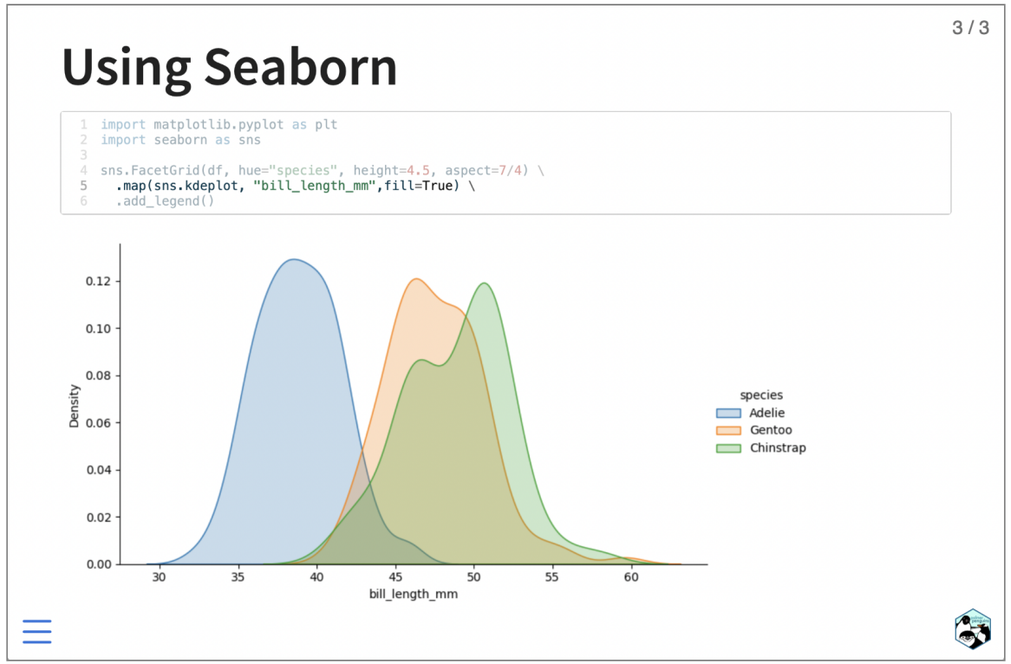
Render Notebook to Revealjs (show code with line by line highlighting)





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





Quarto Projects

_quarto.yml

```yaml
project:
  type: website

website:
  title: "Acme"
  navbar:
    left:
      - href: index.qmd
        text: Home
      - about.qmd

format:
  html:
    theme: cosmo
    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

```yaml
project:
  type: website

website:
  title: "Acme"
  navbar:
    left:
      - href: index.qmd
        text: Home
      - about.qmd

format:
  html:
    theme: cosmo
    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

```yaml
project:
  type: website

website:
  title: "Acme"
  navbar:
    left:
      - href: index.qmd
        text: Home
      - about.qmd

format:
  html:
    theme: cosmo
    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

```yaml
project:
  type: website

website:
  title: "Acme"
  navbar:
    left:
      - href: index.qmd
        text: Home
      - about.qmd

format:
  html:
    theme: cosmo
    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

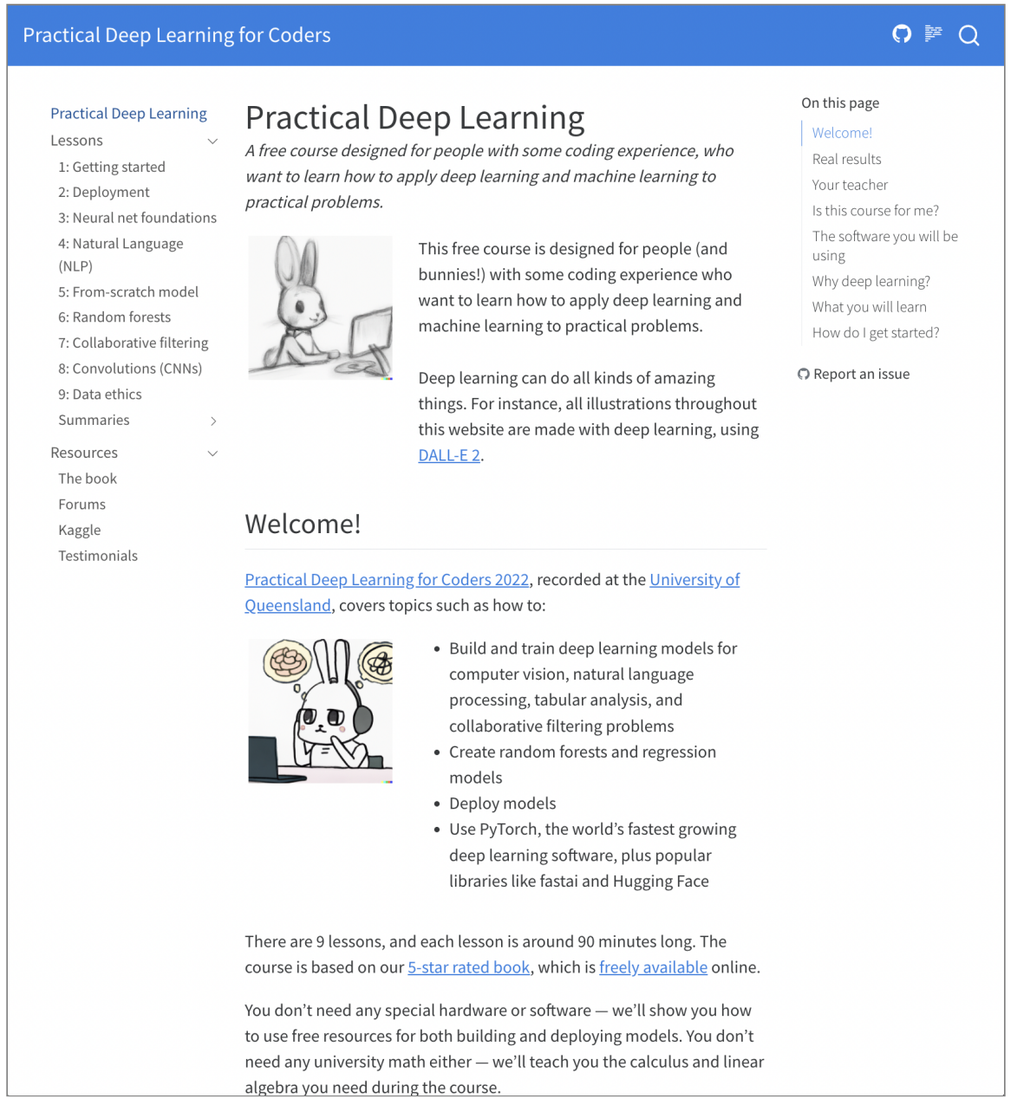
_quarto.yml

```yaml
project:
  type: website
  resources:
    - "www/*"

format:
  html:
    theme: cosmo
    css: styles.css
    toc: true

website:
  title: "Practical Deep Learning for Coders"
  description: "Learn Deep Learning with fastai and PyTorch
  twitter-card: true
  open-graph: true
  reader-mode: true
  page-navigation: true
  repo-branch: master
  repo-url: https://github.com/fastai/course22
  repo-actions: [issue]
  navbar:
    search: true
    right:
      - icon: github
        href: https://github.com/fastai/course22
  sidebar:
    style: "floating"

metadata-files:
  - sidebar.yml
```

Website: Fastai Course

_quarto.yml

```yaml
project:
  type: website
  resources:
    - "www/*"

format:
  html:
    theme: cosmo
    css: styles.css
    toc: true

website:
  title: "Practical Deep Learning for Coders"
  description: "Learn Deep Learning with fastai and PyTorch
  twitter-card: true
  open-graph: true
  reader-mode: true
  page-navigation: true
  repo-branch: master
  repo-url: https://github.com/fastai/course22
  repo-actions: [issue]
  navbar:
    search: true
    right:
      - icon: github
        href: https://github.com/fastai/course22
  sidebar:
    style: "floating"

metadata-files:
  - sidebar.yml
```



Blog: Aayush Agrawal

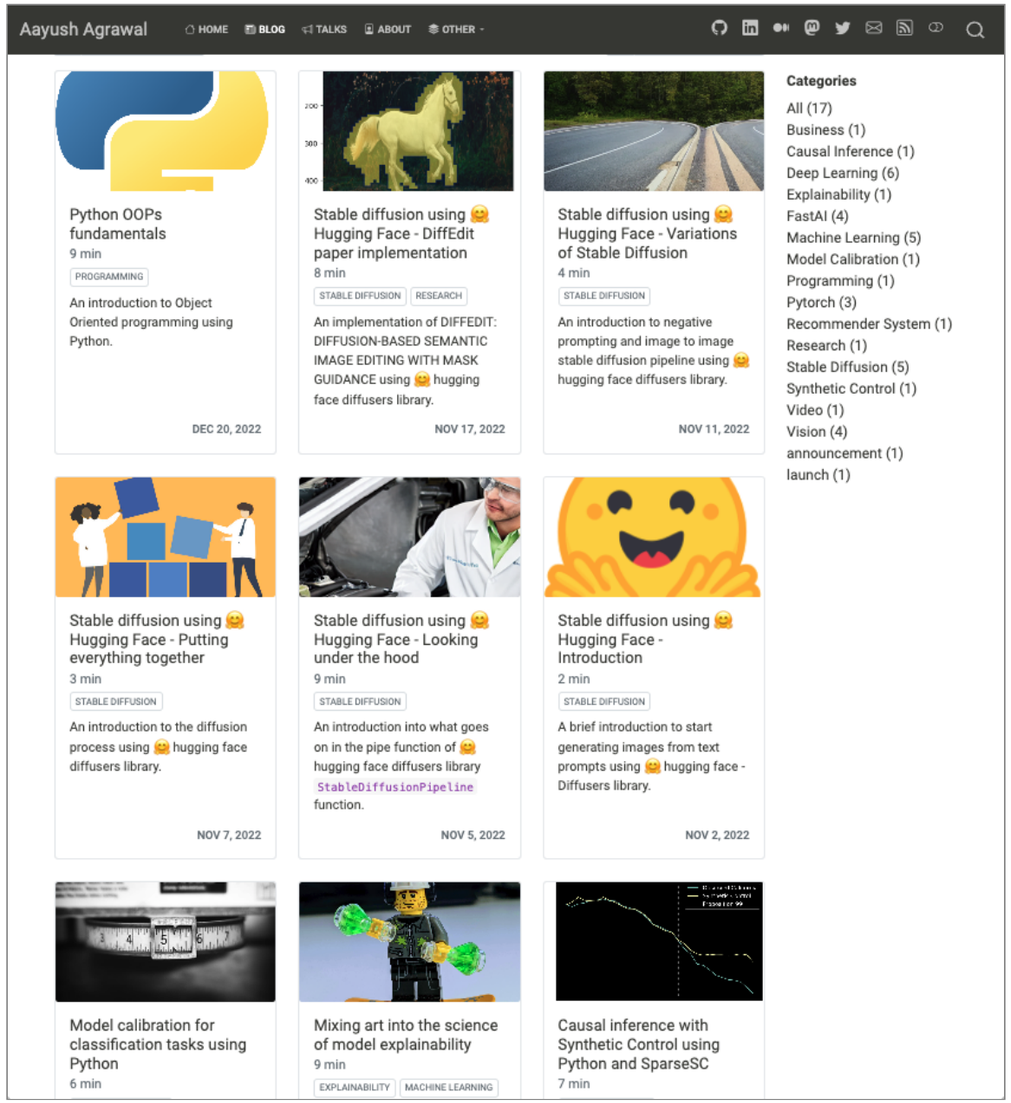
_quarto.yml

```yaml
project:
  type: website

website:
  title: "Aayush Agrawal"
  description: "Aayush's personal website"
  repo-url: https://github.com/aayushmnit/aayushmnit.github
  repo-actions: [edit, issue]
  repo-branch: main
  open-graph: true
  google-analytics: "G-7QN8N70N41"
  twitter-card:
    creator: "@aayushmnit"
    card-style: summary_large_image
  navbar:
    collapse-below: lg
    left:
      - icon: newspaper
        href: blog.qmd
        text: Blog
    right:
      - icon: github
        href: https://github.com/aayushmnit/
      - icon: rss
        href: blog.xml

format:
  html:
    theme: sandstone
    mainfont: Roboto
    css: styles.css
```

Blog: Aayush Agrawal

_quarto.yml

```yaml
project:
  type: website

website:
  title: "Aayush Agrawal"
  description: "Aayush's personal website"
  repo-url: https://github.com/aayushmnit/aayushmnit.github
  repo-actions: [edit, issue]
  repo-branch: main
  open-graph: true
  google-analytics: "G-7QN8N70N41"
  twitter-card:
    creator: "@aayushmnit"
    card-style: summary_large_image
  navbar:
    collapse-below: lg
    left:
      - icon: newspaper
        href: blog.qmd
        text: Blog
    right:
      - icon: github
        href: https://github.com/aayushmnit/
      - icon: rss
        href: blog.xml

format:
  html:
    theme: sandstone
    mainfont: Roboto
    css: styles.css
```



Book: Geocomputation with Python

_quarto.yml

```yaml
project:
  type: book

book:
  title: "Geocomputation with Python"
  author: |
    Michael Dorman, Anita Graser,
    Jakub Nowosad, Robin Lovelace
  description: |
    An introductory resource for working with geographic
    data in Python
  cover-image: https://geocompx.org/static/img/book_cover_p
  site-url: https://py.geocompx.org
  repo-url: https://github.com/geocompx/geocompy/
  repo-actions: [edit]
  sharing: [twitter, facebook, linkedin]
  chapters:
    - index.qmd
    - preface.qmd
    - 02-spatial-data.qmd
    - 03-attribute-operations.qmd
    - 04-spatial-operations.qmd
    - 05-geometry-operations.qmd
    - 06-raster-vector.qmd
    - 07-reproj.qmd

format:
  html:
    theme: flatly
    template-partials: [toc.html,title-block.html]
    code-overflow: wrap
```

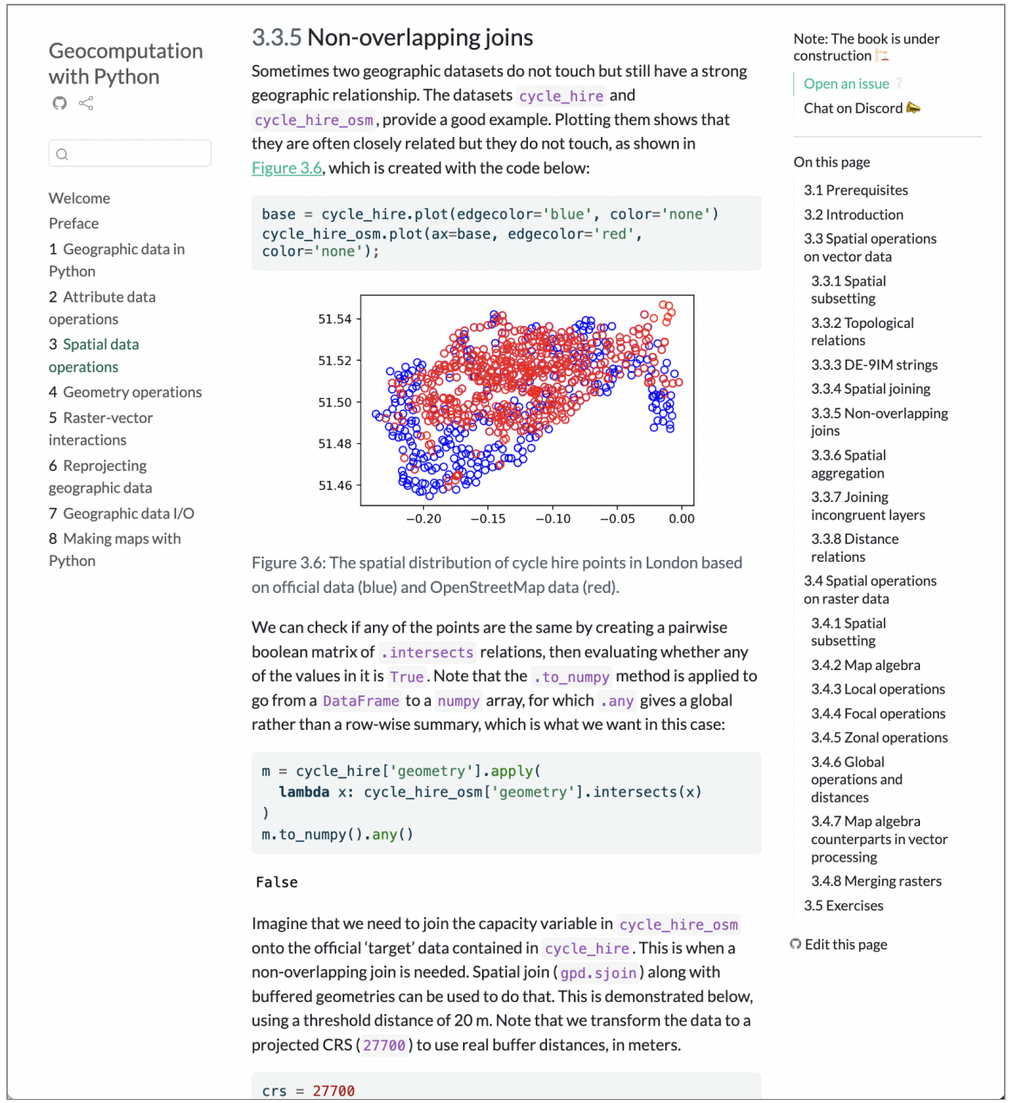
Book: Geocomputation with Python

_quarto.yml

```yaml
project:
  type: book

book:
  title: "Geocomputation with Python"
  author: |
    Michael Dorman, Anita Graser,
    Jakub Nowosad, Robin Lovelace
  description: |
    An introductory resource for working with geographic
    data in Python
  cover-image: https://geocompx.org/static/img/book_cover_p
  site-url: https://py.geocompx.org
  repo-url: https://github.com/geocompx/geocompy/
  repo-actions: [edit]
  sharing: [twitter, facebook, linkedin]
  chapters:
    - index.qmd
    - preface.qmd
    - 02-spatial-data.qmd
    - 03-attribute-operations.qmd
    - 04-spatial-operations.qmd
    - 05-geometry-operations.qmd
    - 06-raster-vector.qmd
    - 07-reproj.qmd

format:
  html:
    theme: flatly
    template-partials: [toc.html,title-block.html]
    code-overflow: wrap
```



Geocomputation 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

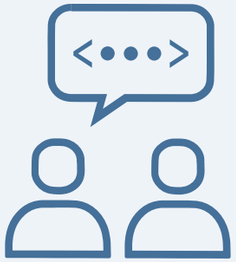
On this page

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)

```
```python
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. Indices are offset by 3 because 0, 1, and 2
    are reserved for "padding", "start of sequence" and "unknown".
```

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)

```python
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. Indices are offset by 3 because 0, 1, and 2 are reserved for "padding", "start of sequence" and "unknown".

Code Annotation

```python
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)

```python
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. Indices are offset by 3 because 0, 1, and 2 are reserved for "padding", "start of sequence" and "unknown".

Code Annotation

```python
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)

```python
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. Indices are offset by 3 because 0, 1, and 2 are reserved for "padding", "start of sequence" and "unknown".

Code Annotation

```python
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 <br/>prevail!
  John-->>Alice: Great!
  John->>Bob: How about you?
  Bob-->>John: Jolly good!
```

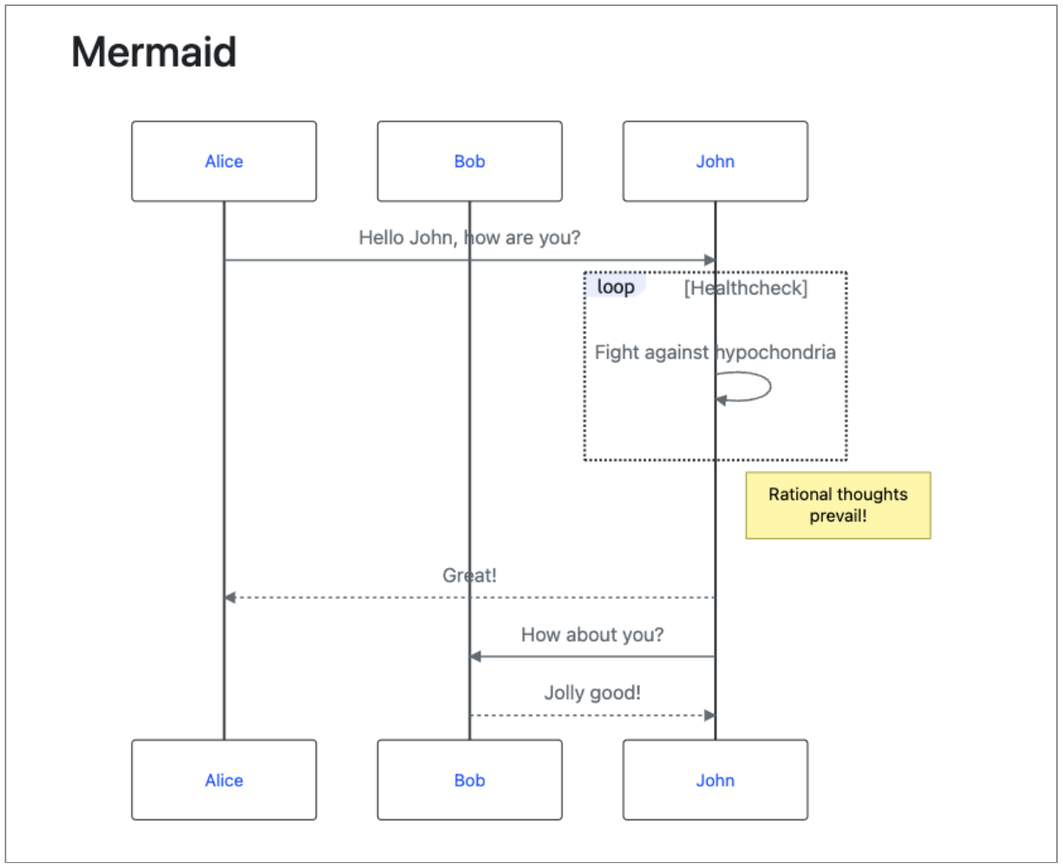
```{dot}
graph G {
  layout=neato
  run -- intr;
  intr -- runbl;
  runbl -- run;
  run -- kernel;
  kernel -- zombie;
}
```

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 <br/>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;
}
```



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)

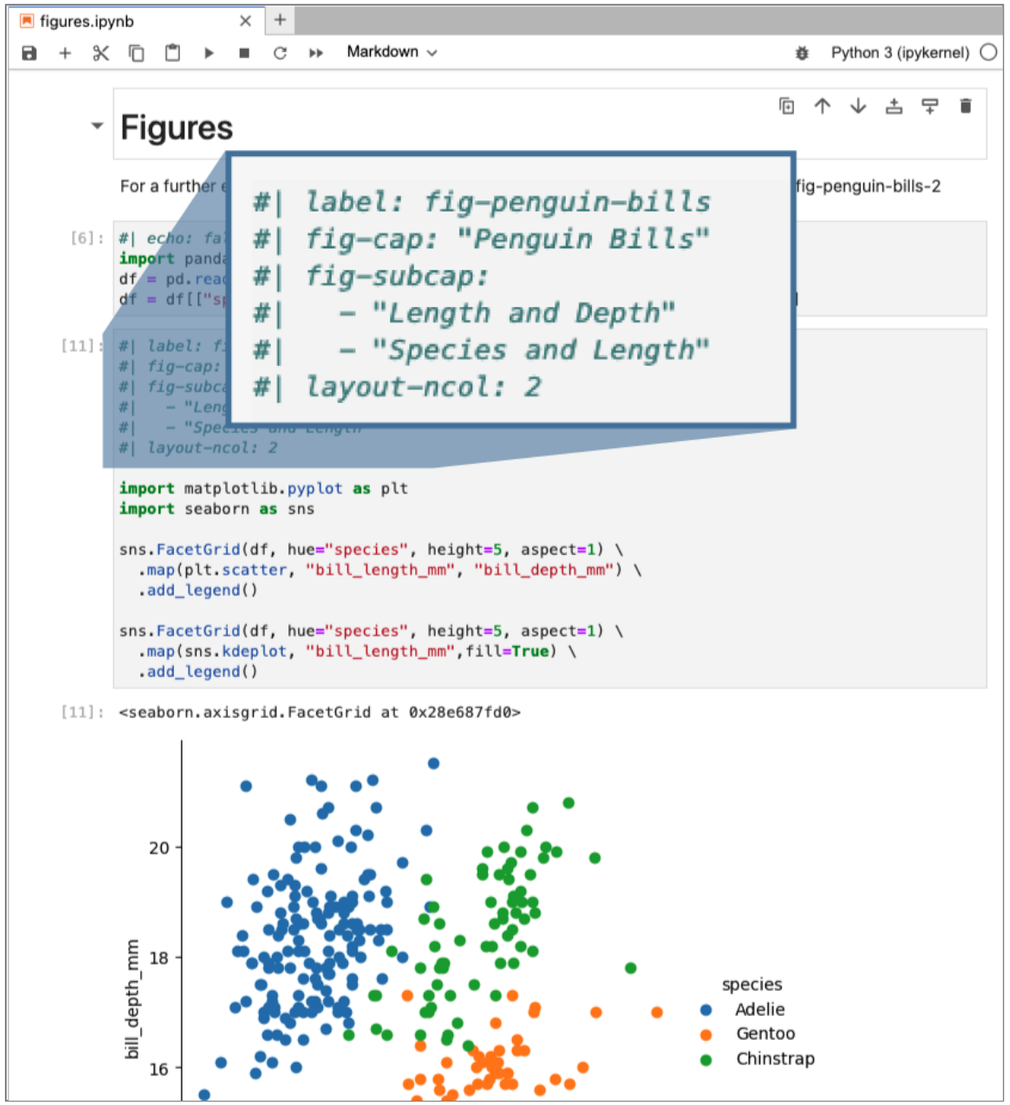
```
$$
\Delta C \approx \frac{\partial C}{\partial v_1} \Delta v_1 +
\frac{\partial C}{\partial v_2} \Delta v_2.
$$


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

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

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

Figures and Cross References



Figures and Cross References

figures.ipynb

Markdown

Python 3 (ipykernel)

Figures

For a further e… fig-penguin-bills-2

```
#| label: fig-penguin-bills
#| fig-cap: "Penguin Bills"
#| fig-subcap:
#|   - "Length and Depth"
#|   - "Species and Length"
#| layout-ncol: 2
```

[6]:
```
#| echo: fa
import panda
df = pd.read
df = df[["sp
```

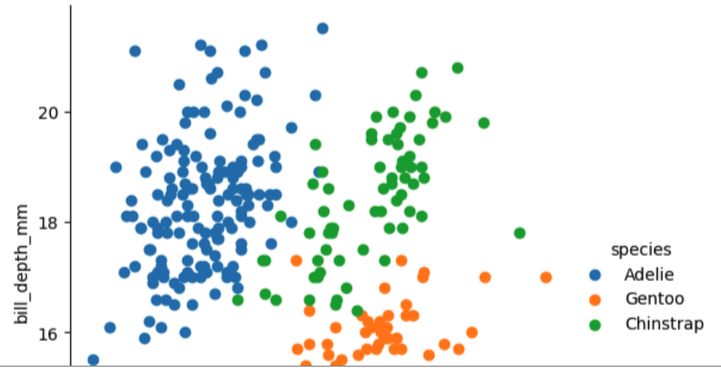
[11]:
```
#| label: f
#| fig-cap:
#| fig-subca
#|   - "Leng
#|   - "Species and Length"
#| layout-ncol: 2

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()
```

[11]: <seaborn.axisgrid.FacetGrid at 0x28e687fd0>



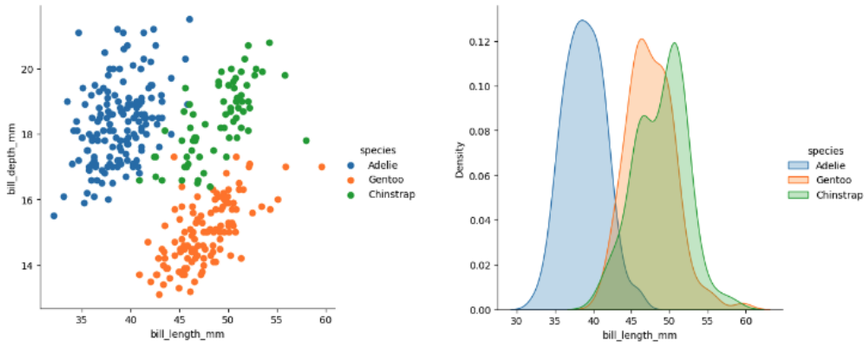
Figures

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()
```



(a) Length and Depth *(b) Species and Length*

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

Markdown

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 $[a, b]$:

$$\frac{d}{dx}\left( \int_{a}^{x} f(u)\,du\right)=f(x).$$
:::
```

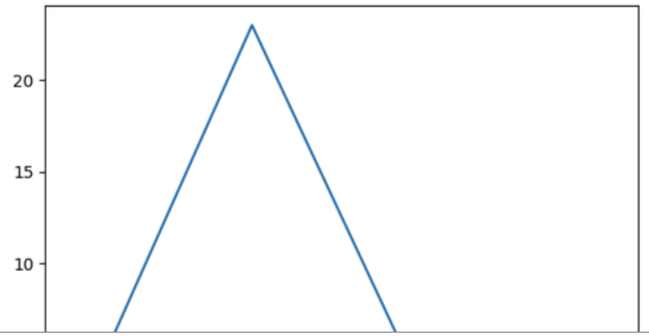
Figures and Tables

Figures th

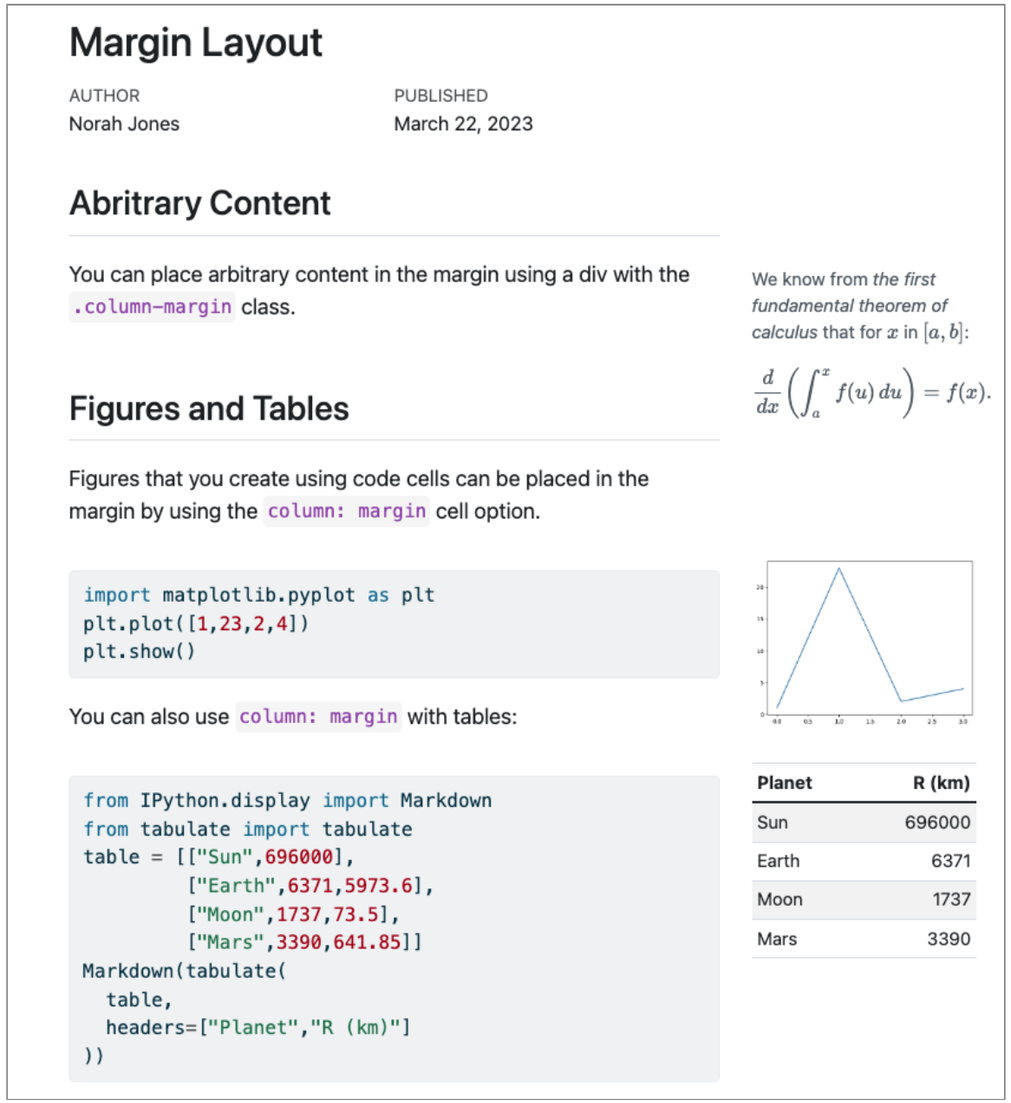
option.

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

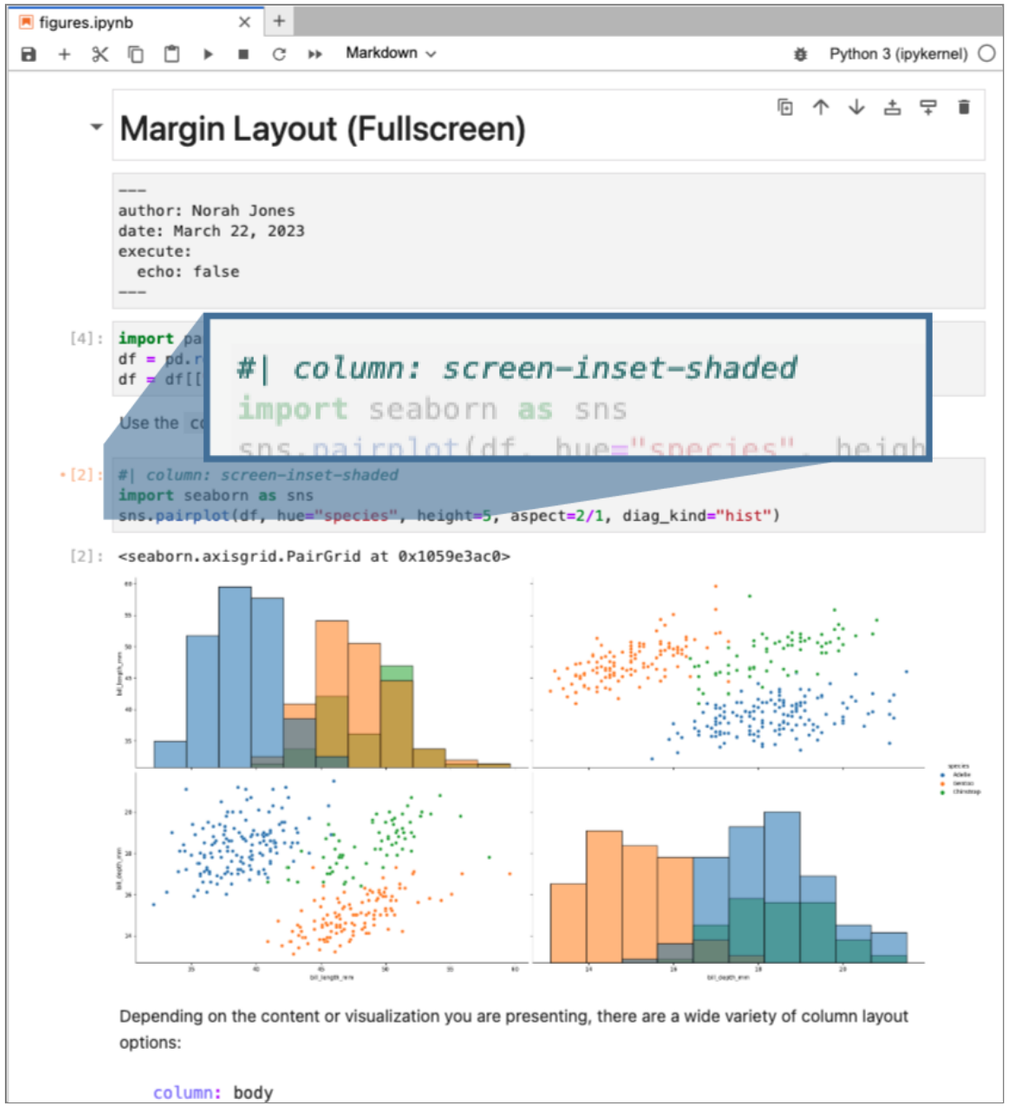
[2]:
```
#| column
import matplotlib.pyplot as plt
plt.plot([1,23,2,4])
plt.show()
```



Margin Layout



Margin Layout (Fullscreen)



Margin Layout (Fullscreen)

figures.ipynb

Markdown

Python 3 (ipykernel)

Margin Layout (Fullscreen)

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

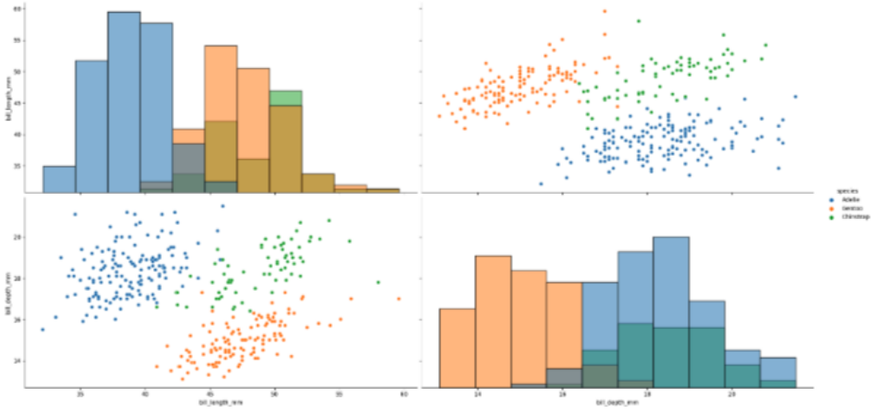
[4]:
```
import pa
df = pd.r
df = df[[
```

Use the c

```
#| column: screen-inset-shaded
import seaborn as sns
sns.pairplot(df, hue="species", height
```

[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:

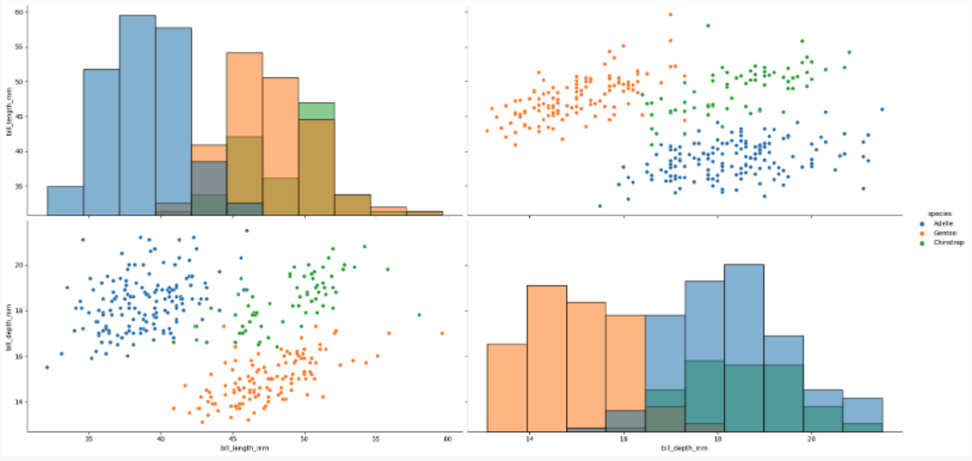
```
column: body
```

Margin Layout (Fullscreen)

AUTHOR
Norah Jones

PUBLISHED
March 22, 2023

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



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
- 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

```
---
title: "My Document"
---

## Code Blocks

This is a *code block*:

```python
def add(x, y):
  return x + 1
```

Block Quotes

> This is a block quote. Block quotes are
> specified by proceeding lines with `>`

```

# Semantic Authoring

semantic.md

```md
---
title: "My Document"
---

## Code Blocks

This is a *code block*:

```python
def add(x, y):
  return x + 1
```

Block Quotes

> This is a block quote. Block quotes are
> specified by proceeding lines with `>`

```

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

# Semantic Authoring

semantic.md

```
---
title: "My Document"

Code Blocks

This is a *code block*:

```python
def add(x, y):
  return x + 1
```

Block Quotes

> This is a block quote. Block quotes are
> specified by proceeding lines with `>`

```

- 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

```md
---
title: "My Document"
---

## Code Blocks

This is a *code block*:

```python
def add(x, y):
  return x + 1
```

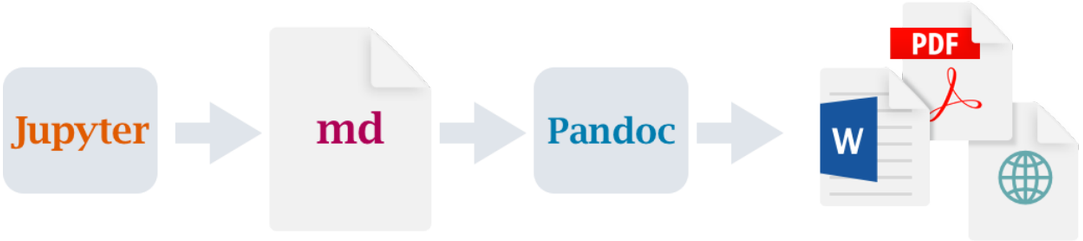
Block Quotes

> This is a block quote. Block quotes are
> specified by proceeding lines with `>`

```

- 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

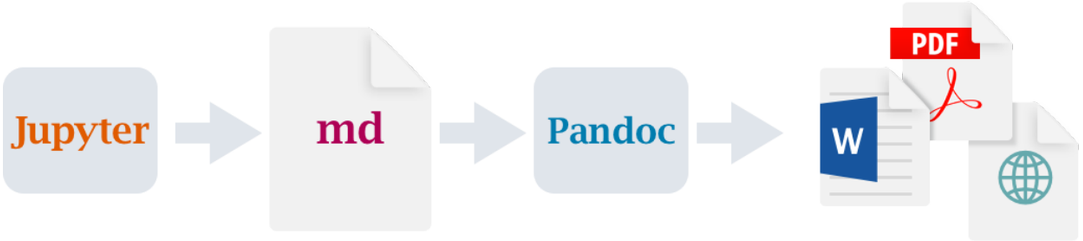
# Semantic Authoring with Jupyter




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

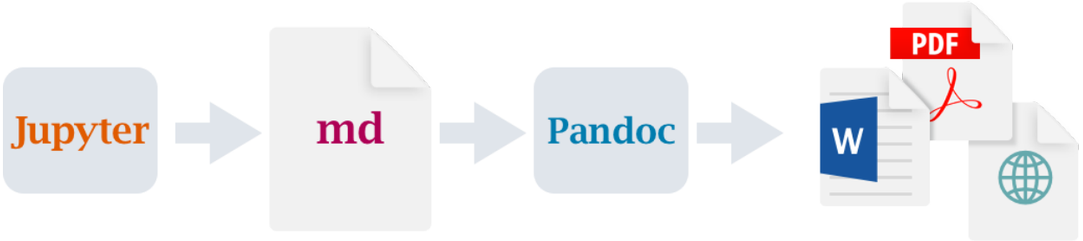



Jupyter executes cells and produces a document with markdown cells and markdown executable output

Pandoc renders markdown into a variety of formats

- Markdown is translated into **document AST**

# Semantic Authoring with Jupyter

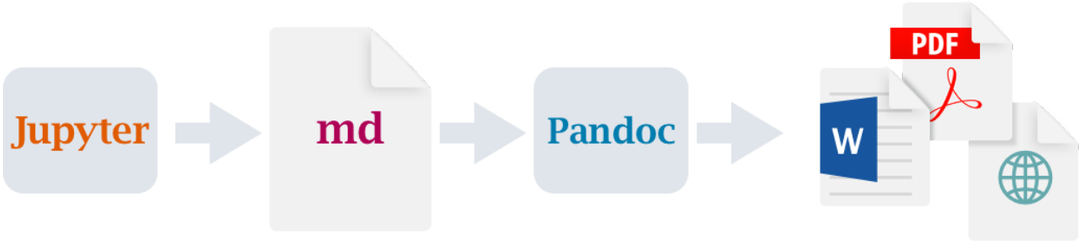



Jupyter executes cells and produces a document with markdown cells and markdown executable output

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

index.ipynb

Markdown  Python 3 (ipykernel)

## Palmer Penguins

```
---
author: Norah Jones
date: 2023-03-08T07:17:38-05:00

```

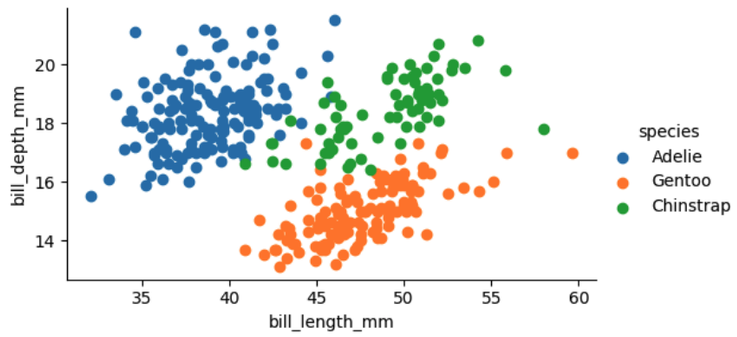
[2]:
```
import pandas as pd
df = pd.read_csv("palmer-penguins.csv")
df = df[["species", "island", "year", "sex", "bill_length_mm", "bill_depth_mm"]]
df.head()
```

[2]:

	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>
```



Hugo: Goldmark Markdown

index.ipynb

Markdown Python 3 (ipykernel)

Palmer Penguins

```
---
author: Norah Jones
date: 2023-03-08T07:17:38-05:00
---
```

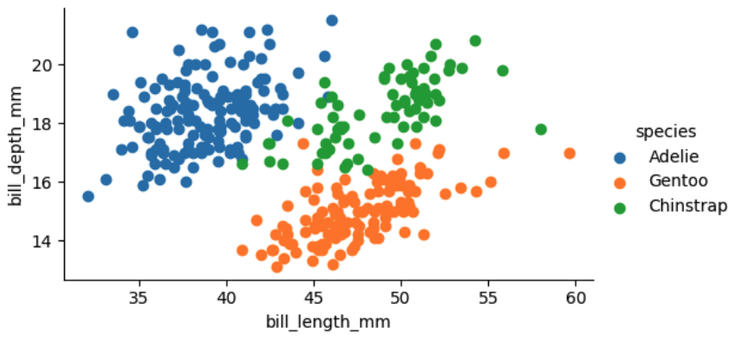
[2]:
```python
import pandas as pd
df = pd.read_csv("palmer-penguins.csv")
df = df[["species", "island", "year", "sex", "bill_length_mm", "bill_depth_mm"]]
df.head()
```

[2]:

	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]:
```python
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

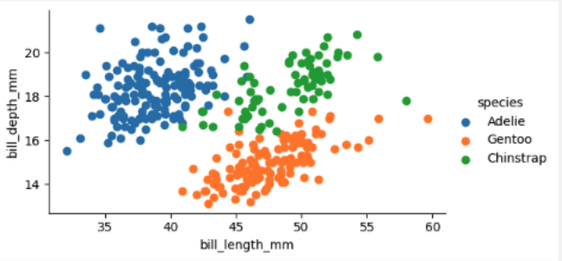
By **Norah Jones**

March 8, 2023

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

```python
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

Markdown

Python 3 (ipykernel)

Palmer Penguins

```
---
author: Norah Jones
---
```

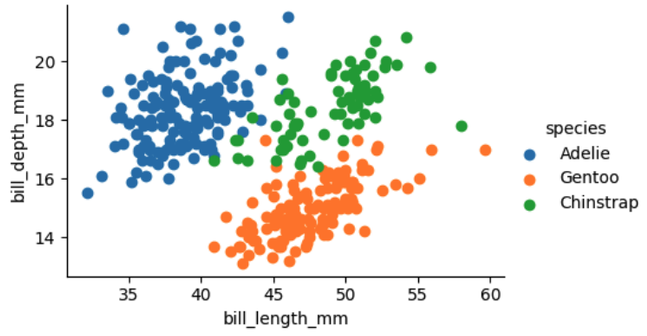
[7]:
```
import pandas as pd
df = pd.read_csv("palmer-penguins.csv")
df = df[["species", "island", \
         "bill_length_mm", "bill_depth_mm"]]
df.head(3)
```

[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>
```



Docusaurus: MDX Markdown

2023-03-02-palmer-pengu

Markdown Python 3 (ipykernel)

Palmer Penguins

```
---
author: Norah Jones
---
```

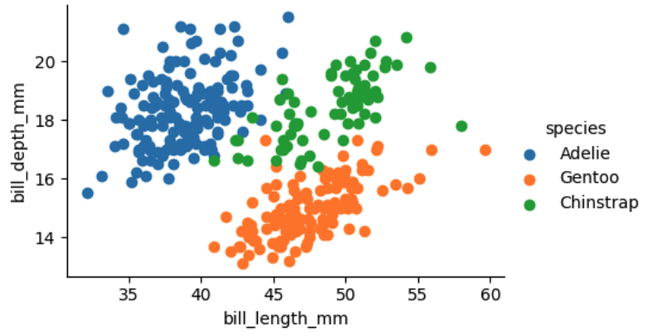
[7]:
```
import pandas as pd
df = pd.read_csv("palmer-penguins.csv")
df = df[["species", "island", \
         "bill_length_mm", "bill_depth_mm"]]
df.head(3)
```

[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

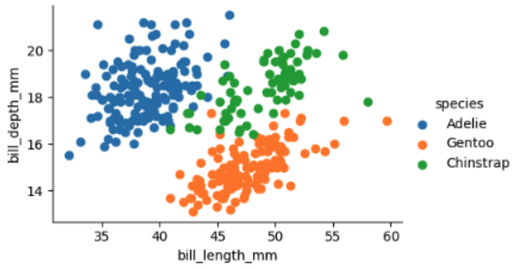
March 2, 2023 · One min read

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

palmer-penguins.ipynb

Markdown

Python 3 (ipykernel)

Palmer Penguins

```
---
author: Norah Jones
---
```

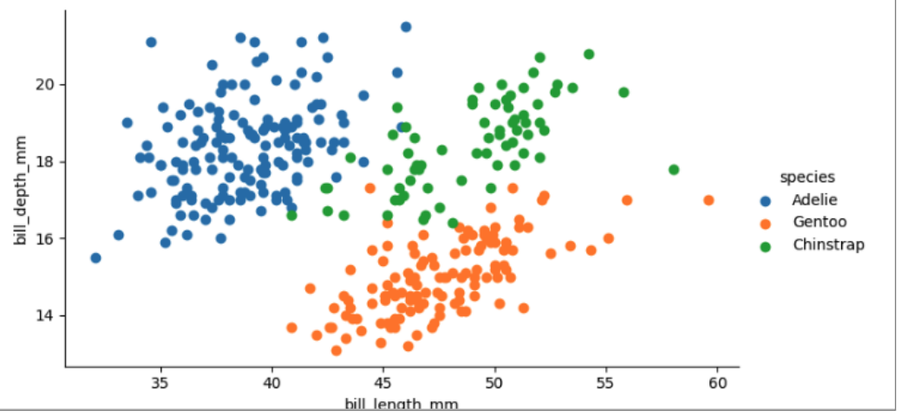
[2]:
```
import pandas as pd
df = pd.read_csv("palmer-penguins.csv")
df = df[["species", "island", \
         "bill_length_mm", "bill_depth_mm"]]
df.head(3)
```

[2]:

	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

[3]:
```
import matplotlib.pyplot as plt
import seaborn as sns
g = sns.FacetGrid(df, hue="species", height=4, aspect=3.5/2)
g.map(plt.scatter, "bill_length_mm", "bill_depth_mm") \
  .add_legend()
```

[3]:
```
<seaborn.axisgrid.FacetGrid at 0x1606a9690>
```



Confluence: Confluence XML

palmer-penguins.ipynb

Markdown | Python 3 (ipykernel)

Palmer Penguins

```
---
author: Norah Jones
---
```

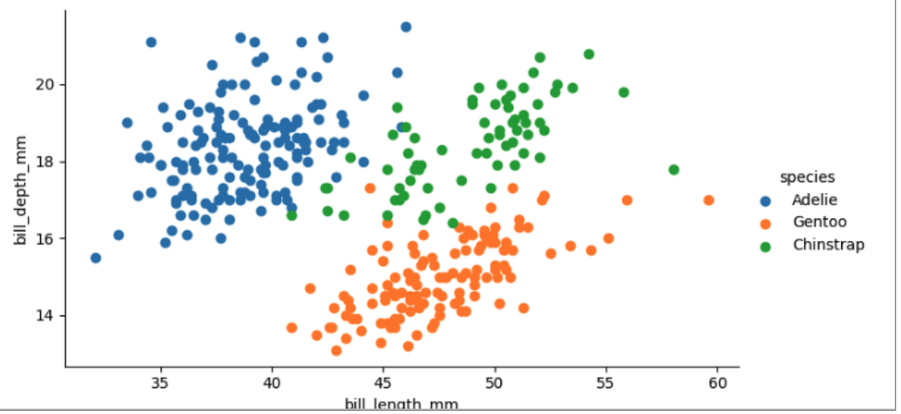
[2]:
```
import pandas as pd
df = pd.read_csv("palmer-penguins.csv")
df = df[["species", "island", \
         "bill_length_mm", "bill_depth_mm"]]
df.head(3)
```

[2]:

	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

[3]:
```
import matplotlib.pyplot as plt
import seaborn as sns
g = sns.FacetGrid(df, hue="species", height=4, aspect=3.5/2)
g.map(plt.scatter, "bill_length_mm", "bill_depth_mm") \
  .add_legend()
```

[3]: <seaborn.axisgrid.FacetGrid at 0x1606a9690>



Home | Recent | Spaces | More | Search

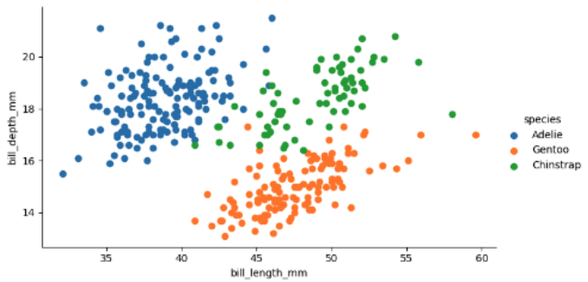
Palmer Penguins

Created by JJ Allaire
Last updated: 10 minutes ago • 1 min read

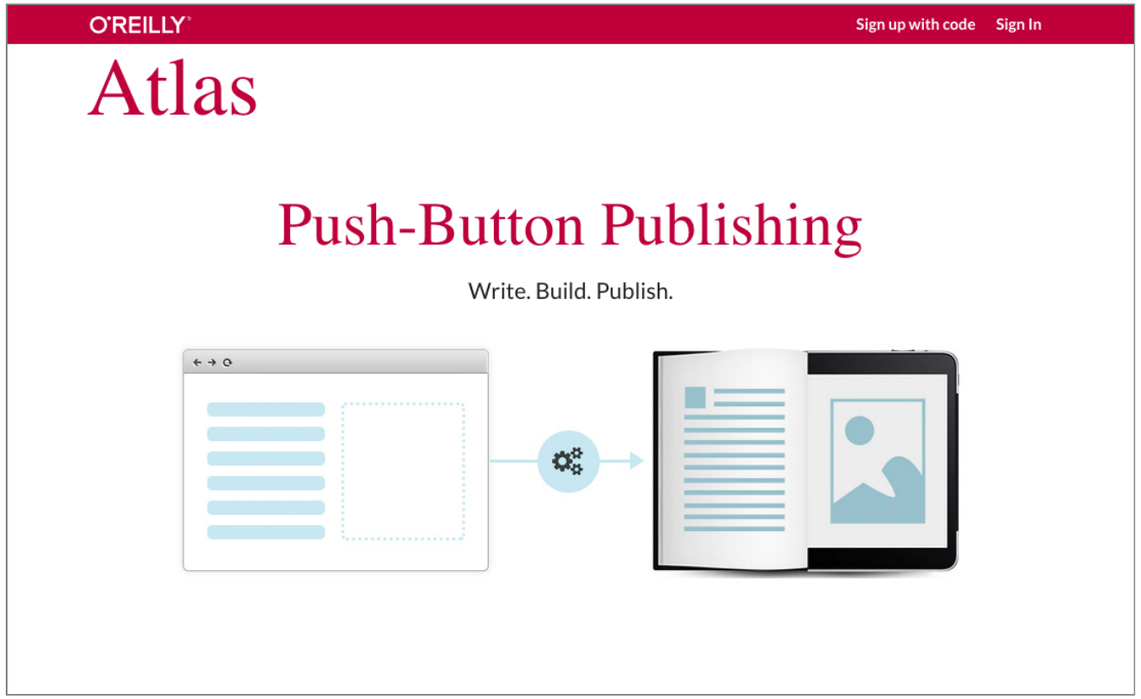
```
import pandas as pd
df = pd.read_csv("palmer-penguins.csv")
df = df[["species", "island", \
         "bill_length_mm", "bill_depth_mm"]]
df.head(3)
```

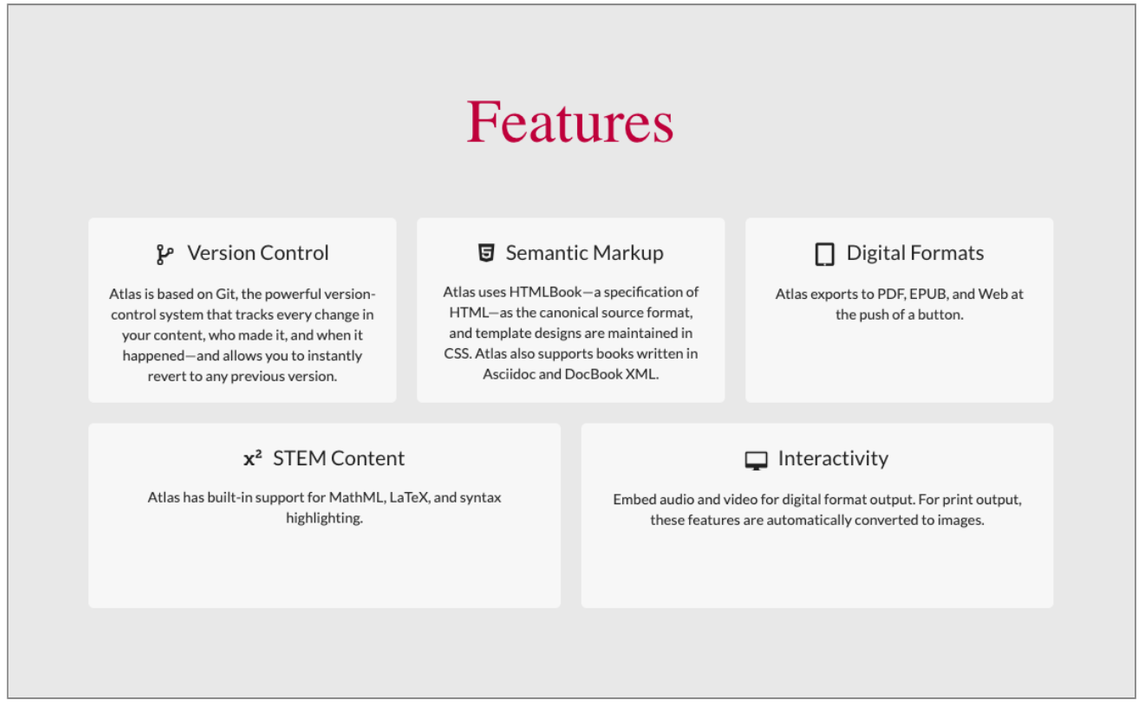
	species	island	bill_length_mm	bill_depth_mm
0	Adelie	Torgersen	39.1	18.7
1	Adelie	Torgersen	39.5	17.4
2	Adelie	Torgersen	40.3	18.0

```
import matplotlib.pyplot as plt
import seaborn as sns
g = sns.FacetGrid(df, hue="species", height=4, aspect=3.5/2)
g.map(plt.scatter, "bill_length_mm", "bill_depth_mm") \
  .add_legend()
```



O’Reilly Atlas





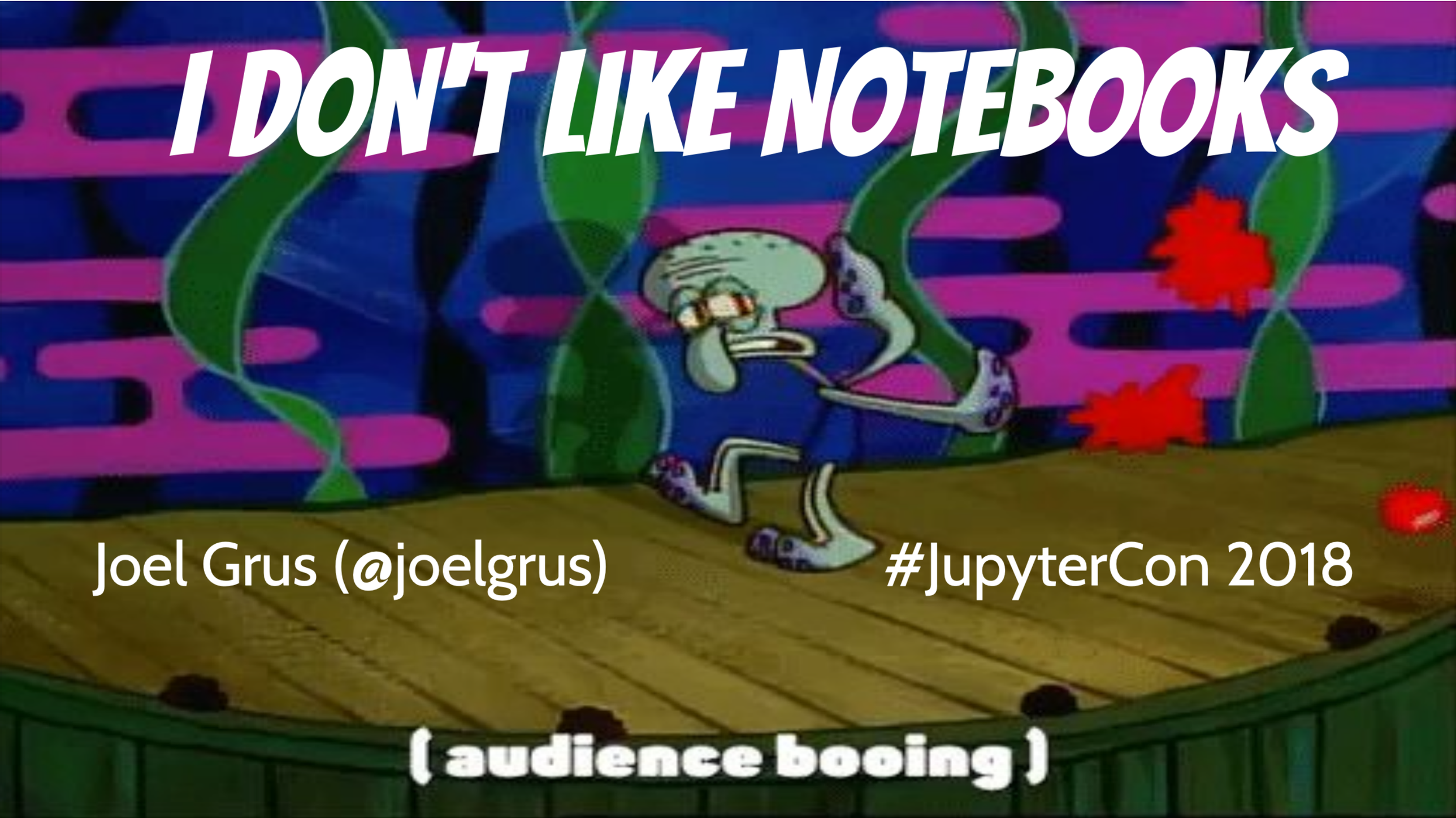
Books can be rendered to `asciidoc`, which is fully compatible with the production requirements of O’Reilly Atlas (used for Print, ePub, and Web books)

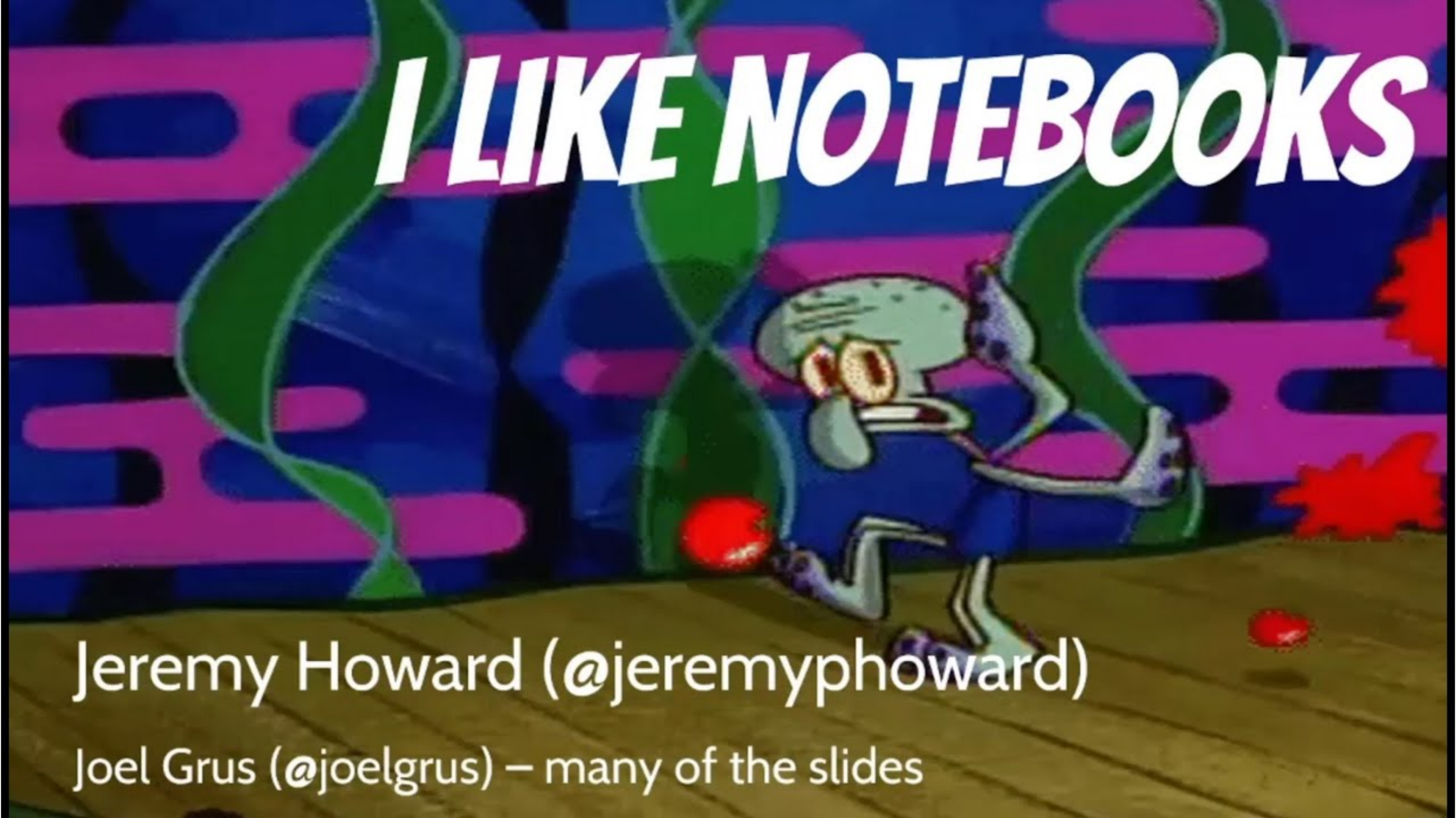
Jupyter Notebooks

I DON'T LIKE NOTEBOOKS

Joel Grus (@joelgrus)

#JupyterCon 2018




I LIKE NOTEBOOKS
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

Jupytext supports 10 different plain text formats for notebooks!

Markdown Formats

- Jupytext Markdown
- R Markdown
- MyST Markdown
- Pandoc Markdown
- Quarto Markdown

Script Formats (.py)

- light
- nomarkder
- percent
- hydrogen
- Sphinx-gallery

Example: Quarto `.qmd` Files

penguins.qmd

```
---
title: "Palmer Penguins"
author: Norah Jones
date: March 12, 2023
format: html
jupyter: python3
---

```{python}
#| 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

See @fig-bill-sizes for an exploration of bill sizes.

```{python}
#| 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)
g.map(plt.scatter, "bill_length_mm", "bill_depth_mm") \
  .add_legend()
```
```

- 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

```qmd
---
title: "Palmer Penguins"
author: Norah Jones
date: March 12, 2023
format: html
jupyter: python3
---

```{python}
#| 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

See @fig-bill-sizes for an exploration of bill sizes.

```{python}
#| 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)
g.map(plt.scatter, "bill_length_mm", "bill_depth_mm") \
  .add_legend()
```
```

- 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

```qmd
---
title: "Palmer Penguins"
author: Norah Jones
date: March 12, 2023
format: html
jupyter: python3
---

```{python}
#| 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

See @fig-bill-sizes for an exploration of bill sizes.

```{python}
#| 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)
g.map(plt.scatter, "bill_length_mm", "bill_depth_mm") \
  .add_legend()
```
```

- 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

```
---
title: "Palmer Penguins"
author: Norah Jones
date: March 12, 2023
format: html
jupyter: python3
---

```{python}
#| 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

See @fig-bill-sizes for an exploration of bill sizes.

```{python}
#| 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)
g.map(plt.scatter, "bill_length_mm", "bill_depth_mm") \
  .add_legend()
```
```

- 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

```
---
title: "Palmer Penguins"
author: Norah Jones
date: March 12, 2023
format: html
jupyter: python3
---

```{python}
#| 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

See @fig-bill-sizes for an exploration of bill sizes.

```{python}
#| 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)
g.map(plt.scatter, "bill_length_mm", "bill_depth_mm") \
  .add_legend()
```
```

- 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

```
---
title: "Palmer Penguins"
author: Norah Jones
date: March 12, 2023
format: html
jupyter: python3
---

```{python}
#| 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

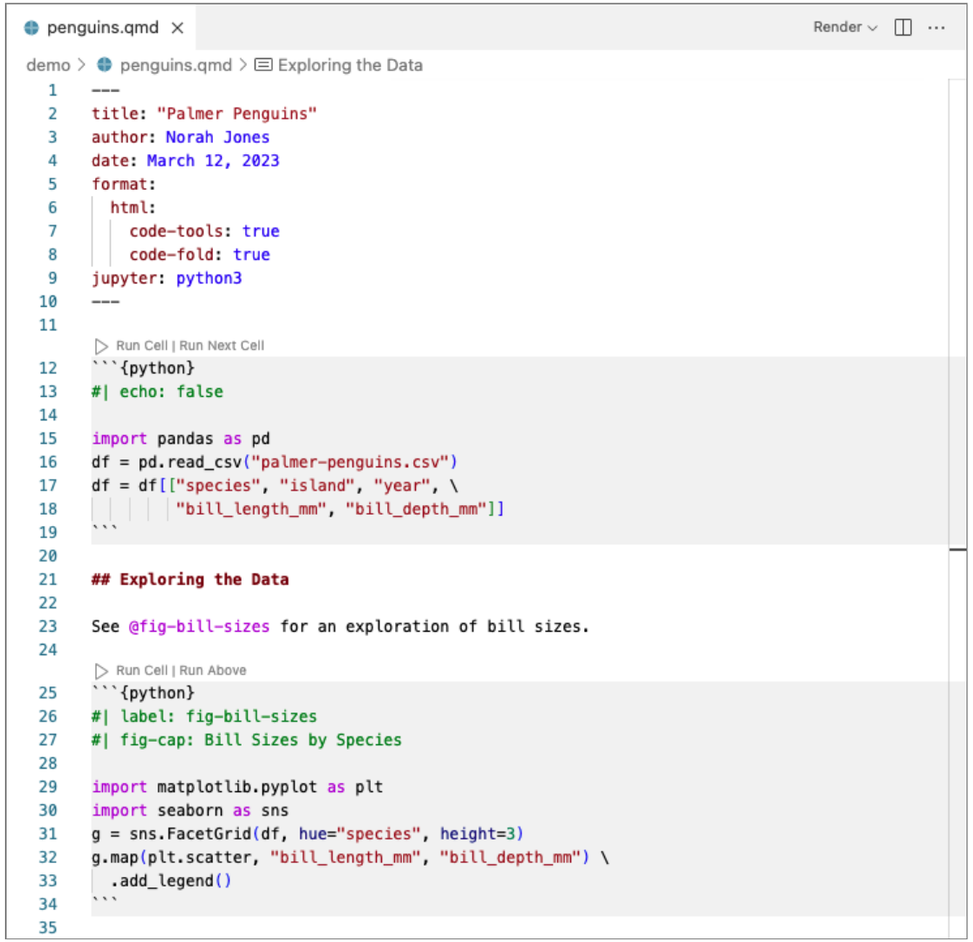
See @fig-bill-sizes for an exploration of bill sizes.

```{python}
#| 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)
g.map(plt.scatter, "bill_length_mm", "bill_depth_mm") \
  .add_legend()
```
```

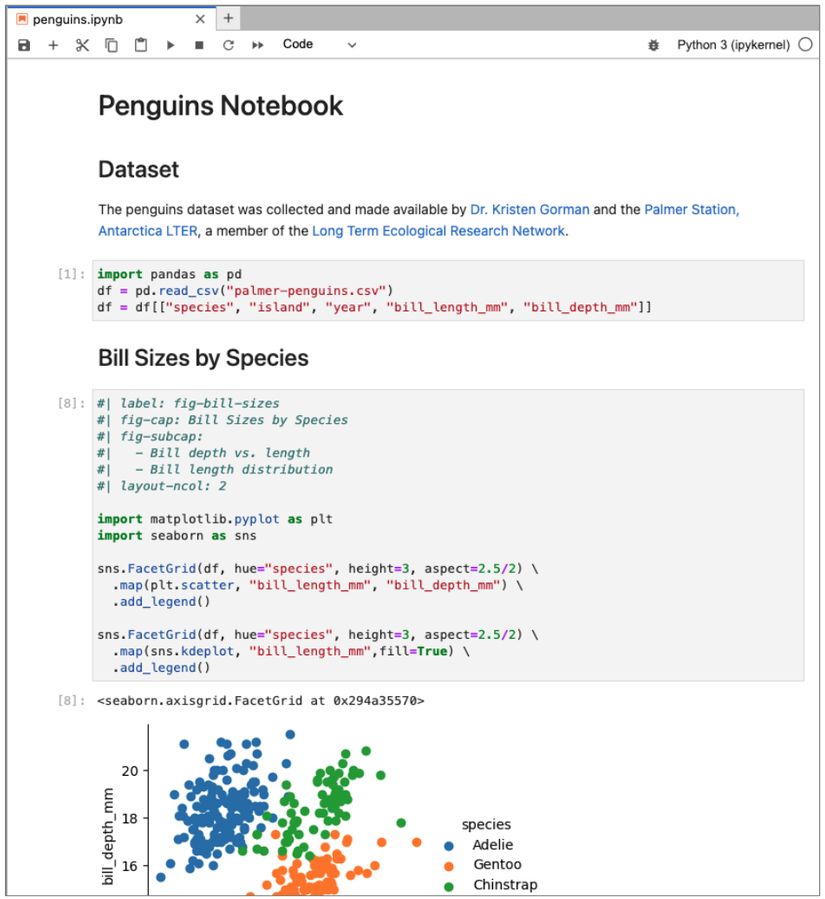
- Editable with any text editor (extensions for VS Code, Neovim, and Emacs)
- Cells always run in the same order
- Integrates well with version control
- Cache output with Jupyter Cache or Quarto freezer
- Lots of pros and cons visa-vi traditional `.ipynb` format/editors, use the right tool for each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

Quarto VS Code Extension



- 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

Render

demo > penguins > index.qmd > Penguin Species

```
---
title: "Penguins"
author: "Norah Jones"
date: March 22, 2023
toc: true
---

## 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
to the west
onshore to

## Bills by

In @fig-bill-sizes we examine penguin bill dimensions by species.

{{< embed penguins.ipynb#fig-bill-sizes >}}

## Bill Dimensions

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.

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

![](culmen_depth.png){width="80%"}

## Penguin Species

There hjave been three penguin species observed on three islands in the Palmer Archipelago,
Antarctica over a study period of three years:

![](species.jpeg){width="80%"}
```

{{< embed penguins.ipynb#fig-bill-sizes >}}

Notebook Embedding

index.qmd U

Render

demo > penguins > index.qmd > Penguin Species

```
---
title: "Penguins"
author: "Norah Jones"
date: March 22, 2023
toc: true
---

## 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
to the west
onshore to

## Bills by

In @fig-bill

{{< embed penguins.ipynb#fig-bill-sizes >}}

## Bill Dimensions

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.

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

![](culmen_depth.png){width="80%"}

## Penguin Species

There hjave been three penguin species observed on three islands in the Palmer Archipelago, Antarctica over a study period of three years:

![](species.jpeg){width="80%"}
```

{{< embed penguins.ipynb#fig-bill-sizes >}}

Penguins

AUTHOR
Norah Jones

PUBLISHED
March 22, 2023

On this page
- Introduction
- Bills by Species
- Bill Dimensions

Notebooks
- penguins.ipynb

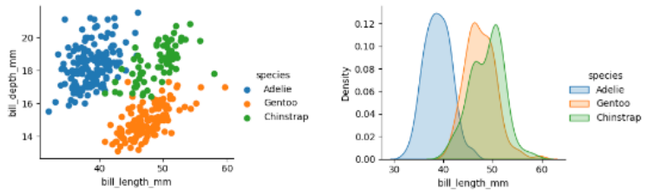
Introduction

The penguins dataset was collected and made ava
collected at the Palmer Long-Term Ecological Rese
located to the west of the Antarctic Peninsula exte
Basin from onshore to several hundred kilometers off shore.

penguins.ipynb

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 D

The cu 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

Notebook Embedding

index.qmd U

Render

demo > penguins > index.qmd > Penguin Species

```
---
title: "Penguins"
author: "Norah Jones"
date: March 22, 2023
toc: true
---

## 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
to the west
onshore to

## Bills by

In @fig-bill

{{< embed penguins.ipynb#fig-bill-sizes >}}

## Bill Dimensions

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.

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

![](culmen_depth.png){width="80%"}

## Penguin Species

There hjave been three penguin species observed on three islands in the Palmer Archipelago,
Antarctica over a study period of three years:

![](species.jpeg){width="80%"}
```

{{< embed penguins.ipynb#fig-bill-sizes >}}

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.

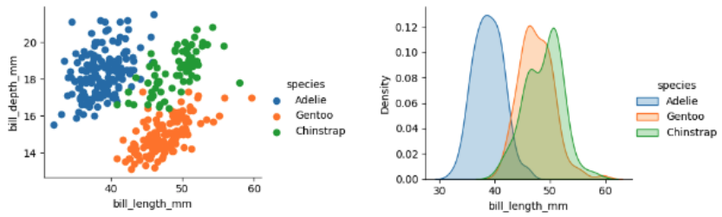
In [1]:
```
import pandas as pd
df = pd.read_csv("palmer-penguins.csv")
df = df[["species", "island", "year", "bill_length_mm", "bill_depth_mm"]]
```

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()
```

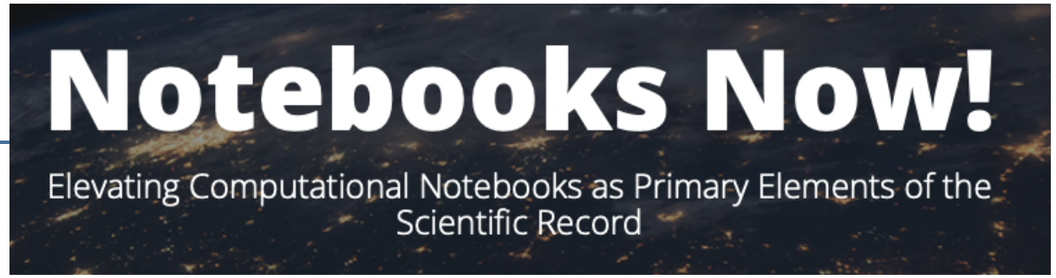


(a) Bill depth vs. length *(b) Bill length distribution*

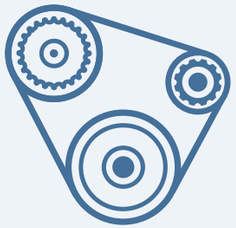
Figure 1: Bill Sizes by Species

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)

> ⓘ 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

```
from panflute import *

def increase_header_level(elem, doc):
    if type(elem) == Header:
        if elem.level < 6:
            elem.level += 1

def main():
    return run_filter(increase_header_level)

if __name__ == "__main__":
    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

```python
from panflute import *

def increase_header_level(elem, doc):
    if type(elem) == Header:
        if elem.level < 6:
            elem.level += 1

def main():
    return run_filter(increase_header_level)

if __name__ == "__main__":
    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

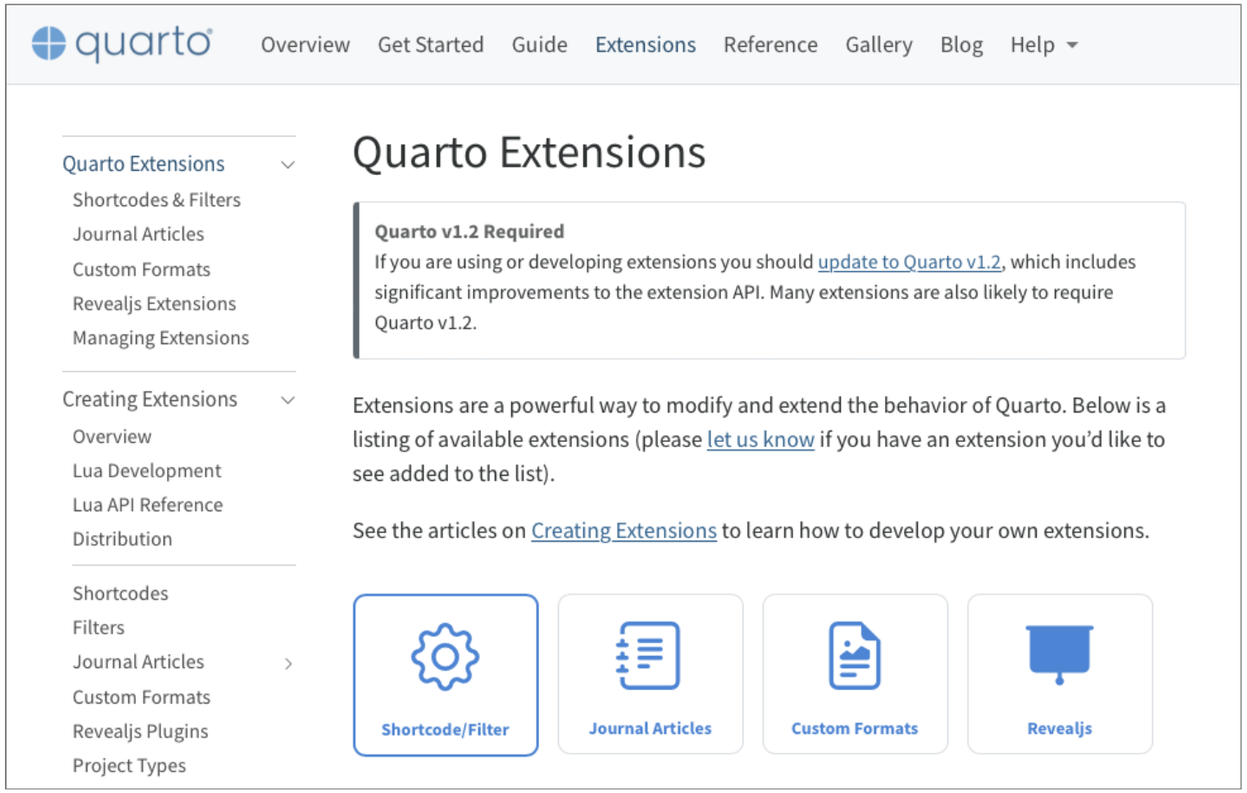
Filter	Description
lightbox	Create lightbox treatments for images in your HTML documents.
molstar	Shortcode to embed proteins and trajectories with Mol*.
social-share	Add buttons to share articles on various social media platforms.
latex-environment	Output divs as custom LaTeX environments.
qrcode	Shortcode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

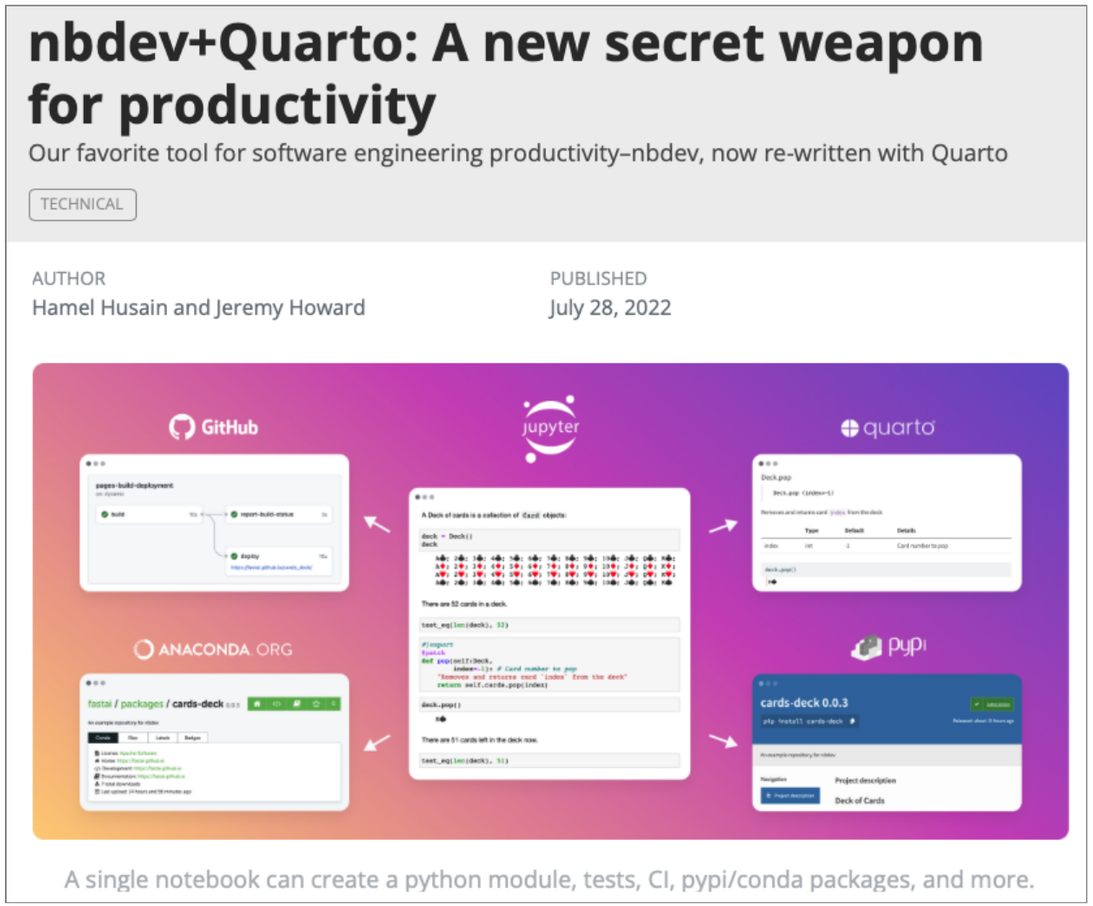
Extensions

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



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

Integration w/ nbdev



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?