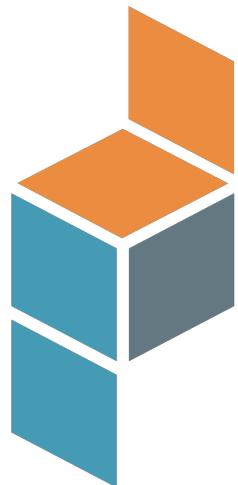


# Quarto Dashboard

*for impactful and visual communication*

Christophe Dervieux  
*Posit PBC*

September 25, 2024



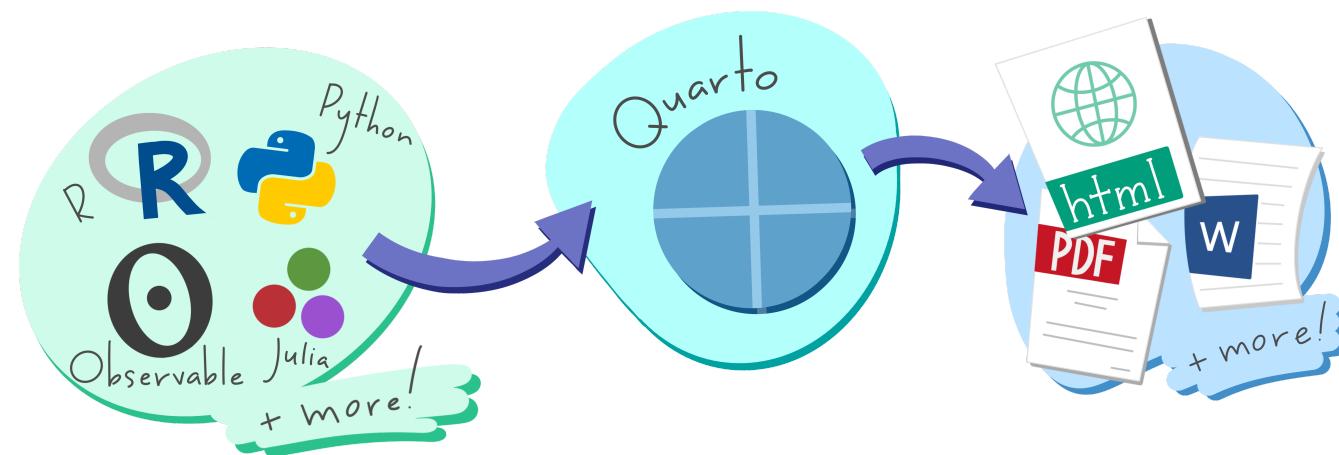
PyData

# What is quarto® ?

An open-source, scientific and technical publishing system

*Building on standard Markdown with features essential for scientific communication*

- Extended Pandoc Markdown for textual content
- Multi-engine support for configurable computations
- Various Output Formats



# With Quarto ...

you can **weave** together **narrative** and **code** to produce elegantly formatted output as documents, web pages, blog posts, books and more...



Follow along at <https://cderv.github.io/pydata-paris-2024-quarto-dashboard/> and <https://quarto.org>



# With Quarto ...

you can **weave** together **narrative** and **code** to produce elegantly formatted output as documents, web pages, blog posts, books and more...

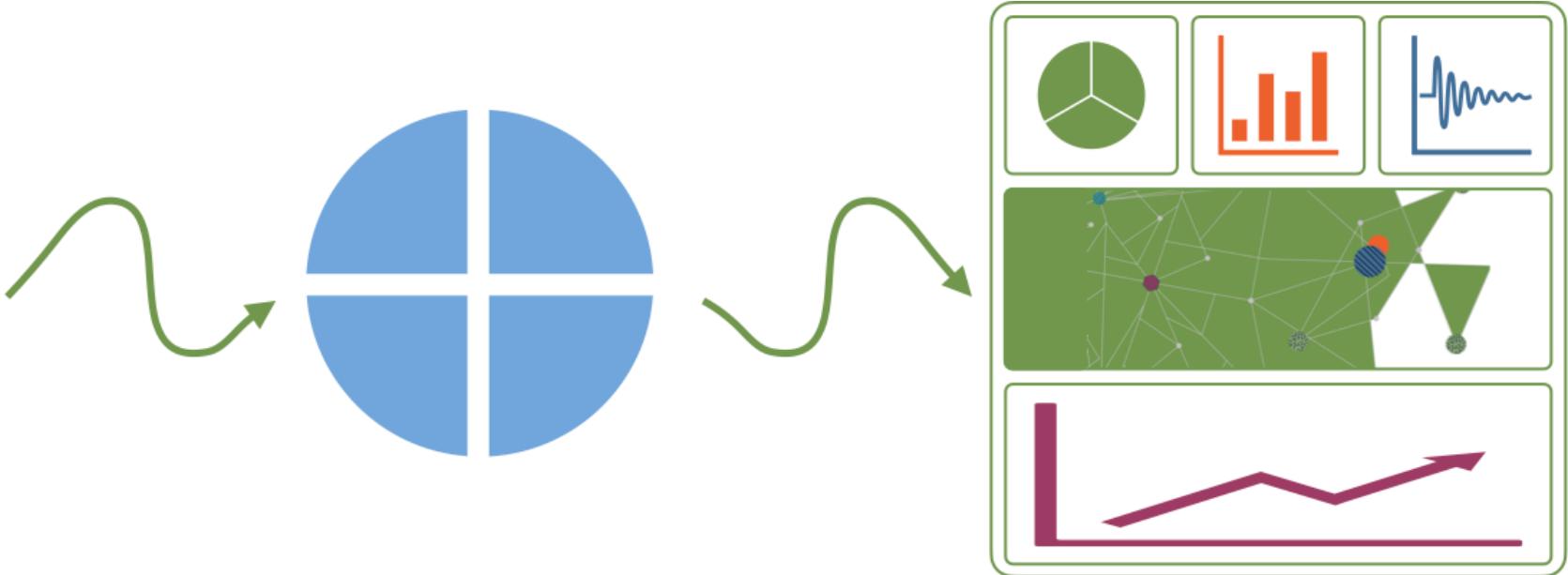
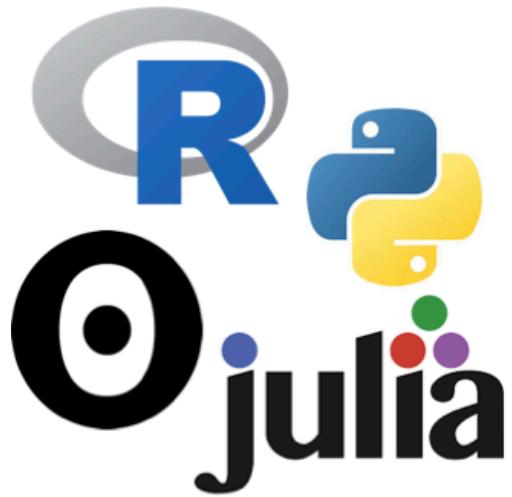
Which includes...



# With Quarto ...

you can **weave** together **narrative** and **code** to produce elegantly formatted output as documents, web pages, blog posts, books and more...

Which includes... **Dashboards**



# How to use Quarto?

Quarto is a [command line interface \(CLI\)](#) that renders Jupyter notebook ([.ipynb](#)) or plain text formats ([.qmd](#), [.md](#)) into reports (PDF, Word, HTML ...), books, websites, presentations (Revealjs, Powerpoint, ...) and more.

```
1 $ quarto --help
2
3 Usage:   quarto
4 Version: 1.5.57
5
6 Description:
7
8   Quarto CLI
9
10 Options:
11
12   -h, --help      - Show this help.
13   -V, --version   - Show the version number for this program.
14
15 Commands:
```



# How to use Quarto?

Quarto is a [command line interface \(CLI\)](#) that renders Jupyter notebook ([.ipynb](#)) or plain text formats ([.qmd](#), [.md](#)) into reports (PDF, Word, HTML ...), books, websites, presentations (Revealjs, Powerpoint, ...) and more.

```
1 $ quarto --help
2
3 Usage:   quarto
4 Version: 1.5.57
5
6 Description:
7
8   Quarto CLI
9
10 Options:
11
12   -h, --help      - Show this help.
13   -V, --version   - Show the version number for this program.
14
15 Commands:
```



# How to use Quarto?

Quarto is a **command line interface (CLI)** that renders Jupyter notebook (`.ipynb`) or plain text formats (`.qmd`, `.md`) into reports (PDF, Word, HTML ...), books, websites, presentations (Revealjs, Powerpoint, ...) and more.

```
1 $ quarto --help
2
3 Usage:   quarto
4 Version: 1.5.57
5
6 Description:
7
8   Quarto CLI
9
10 Options:
11
12   -h, --help      - Show this help.
13   -V, --version   - Show the version number for this program.
14
15 Commands:
```



# How to use Quarto?

Quarto is a **command line interface (CLI)** that renders Jupyter notebook (`.ipynb`) or plain text formats (`.qmd`, `.md`) into reports (PDF, Word, HTML ...), books, websites, presentations (Revealjs, Powerpoint, ...) and more.

```
10 Options:  
11  
12 -h, --help      - Show this help.  
13 -V, --version   - Show the version number for this program.  
14  
15 Commands:  
16  
17 render    [input] [args...]      - Render files or projects to various document types.  
18 preview   [file] [args...]       - Render and preview a document or website project.  
19 serve     [input]                 - Serve a Shiny interactive document.  
20 create    [type] [commands...]   - Create a Quarto project or extension  
21 use       <type> [target]        - Automate document or project setup tasks.  
22 add       <extension>          - Add an extension to this folder or project  
23 update   [target...]           - Updates an extension or global dependency.  
24 remove   [target...]           - Removes an extension.
```



# How to use Quarto?

Quarto is a **command line interface (CLI)** that renders Jupyter notebook (`.ipynb`) or plain text formats (`.qmd`, `.md`) into reports (PDF, Word, HTML ...), books, websites, presentations (Revealjs, Powerpoint, ...) and more.

```
11
12   -h, --help      - Show this help.
13   -V, --version   - Show the version number for this program.
14
15 Commands:
16
17   render      [input] [args...]
18   preview     [file] [args...]
19   serve       [input]
20   create      [type] [commands...]
21   use         <type> [target]
22   add         <extension>
23   update     [target...]
24   remove     [target...]
25   convert     <input>

          - Render files or projects to various document types.
          - Render and preview a document or website project.
          - Serve a Shiny interactive document.
          - Create a Quarto project or extension
          - Automate document or project setup tasks.
          - Add an extension to this folder or project
          - Updates an extension or global dependency.
          - Removes an extension.
          - Convert documents to alternate representations.
```



# How to use Quarto?

Quarto is a **command line interface (CLI)** that renders Jupyter notebook (`.ipynb`) or plain text formats (`.qmd`, `.md`) into reports (PDF, Word, HTML ...), books, websites, presentations (Revealjs, Powerpoint, ...) and more.

15	commands	.	
16			
17	render	<code>[input] [args...]</code>	- Render files or projects to various document types.
18	preview	<code>[file] [args...]</code>	- Render and preview a document or website project.
19	serve	<code>[input]</code>	- Serve a Shiny interactive document.
20	create	<code>[type] [commands...]</code>	- Create a Quarto project or extension
21	use	<code>&lt;type&gt; [target]</code>	- Automate document or project setup tasks.
22	add	<code>&lt;extension&gt;</code>	- Add an extension to this folder or project
23	update	<code>[target...]</code>	- Updates an extension or global dependency.
24	remove	<code>[target...]</code>	- Removes an extension.
25	convert	<code>&lt;input&gt;</code>	- Convert documents to alternate representations.
26	pandoc	<code>[args...]</code>	- Run the version of Pandoc embedded within Quarto.
27	typst	<code>[args...]</code>	- Run the version of Typst embedded within Quarto.
28	run	<code>[script] [args...]</code>	- Run a TypeScript, R, Python, or Lua script.
29	install	<code>[target...]</code>	- Installs a global dependency (TinyTex or Chromium).
30	uninstall	<code>[tool]</code>	- Removes an extension.

# How to use Quarto?

## *Render and Preview*

### Render to output formats:

```
# ipynb notebook  
quarto render notebook.ipynb  
quarto render notebook.ipynb --execute  
quarto render notebook.ipynb --to pdf  
quarto render notebook.ipynb --to dashboard
```

```
# plain text qmd  
quarto render notebook.qmd  
quarto render notebook.qmd --to docx
```

### Live preview server:

```
# ipynb notebook  
quarto preview notebook.ipynb  
quarto preview notebook.ipynb --execute  
quarto preview notebook.ipynb --to revealjs  
  
# plain text qmd  
quarto preview notebook.qmd  
quarto preview notebook.qmd --to revealjs
```

# How to use Quarto?

Quarto integrates with other tools.



# How to use Quarto?



# Quarto integrates with other tools.

The screenshot shows the Quarto Web interface with three main panes. The left pane is the Explorer, listing files and folders. The middle pane is the code editor for 'diagrams.qmd', showing Mermaid and Graphviz syntax. The right pane is the Quarto Preview, displaying the generated flowchart.

**Explorer**

- > OPEN EDITORS
- > QUARTO-WEB
  - > .quarto
  - > .Rproj.user
  - > docs
    - > authoring
      - > images
      - > \_pagebreak.qmd
      - > \_shortcode-escaping.qmd
      - > \_table-crossrefs.md
      - > appendices.qmd
      - > article-layout.qmd
      - > callouts.qmd
      - > conditional.qmd
      - > create-citeable-articles.qmd
      - > cross-references.qmd
      - > diagrams.qmd
      - > elephant.png
      - > figures.qmd
      - > filters.qmd
      - > footnotes-and-citations.qmd
      - > includes.qmd
      - > language.qmd
      - > markdown-basics.qmd
      - > shortcodes.qmd
      - > tables.qmd
      - > title-blocks.qmd
      - > variables.qmd
    - > blog
    - > books
    - > computations
    - > download
    - > faq
    - > gallery
    - > get-started
    - > .Rproj.user
    - > authoring
    - > computations
      - > \_computations-vscode\_files
      - > \_notebooks
      - > .ipynb\_checkpoints
      - > images
      - > \_computations-complete.qmd

# How to use Quarto?

Quarto integrates with other tools.



localhost:8888/lab/tree/quarto-jupyterlab.ipynb

File Edit View Run Kernel Tabs Settings Help

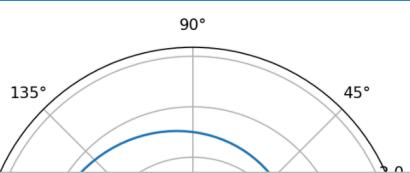
quarto-jupyterlab.ipynb

```
---  
title: Matplotlib Demo  
author: Norah Smith  
date: 'May 22nd, 2021'  
format:  
  html:  
    code-fold: true  
keep-ipynb: true  
---
```

## Polar Axis

For a demonstration of a line plot on a polar axis, see @fig-polar.

```
[2]: #| label: fig-polar  
#| fig.cap: A line plot on a polar axis  
  
import numpy as np  
import matplotlib.pyplot as plt  
  
r = np.arange(0, 2, 0.01)  
theta = 2 * np.pi * r  
fig, ax = plt.subplots(subplot_kw={'projection': 'polar'})  
ax.plot(theta, r)  
ax.set_rticks([0.5, 1, 1.5, 2])  
ax.grid(True)  
plt.show()
```



Terminal 1

```
author: Norah Smith  
date: 'May 22nd, 2021'
```

localhost:6916

## Matplotlib Demo

Norah Smith  
May 22nd, 2021

### Polar Axis

For a demonstration of a line plot on a polar axis, see [fig.1](#).

▶ Code

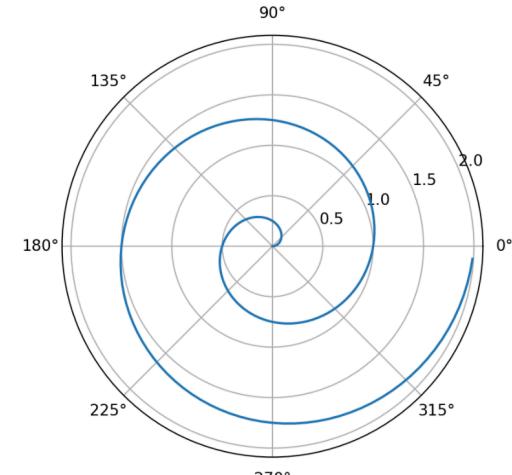


Figure 1: A line plot on a polar axis

# How to use Quarto?



Quarto integrates with other tools.

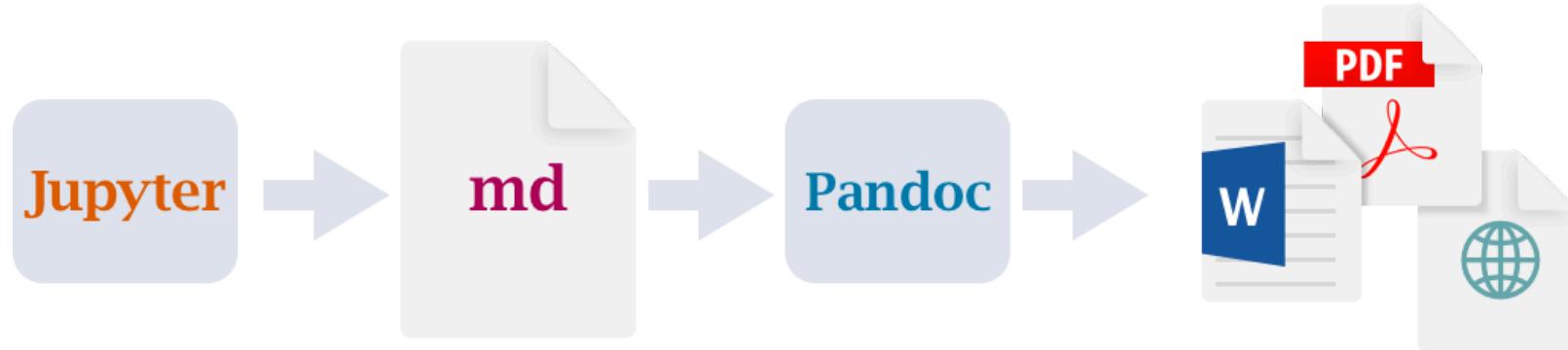
The screenshot shows a Microsoft Visual Studio Code (VS Code) interface with the following details:

- Title Bar:** index.ipynb - paris-metro-dashboard - Positon
- File Explorer:** Shows 'index.ipynb' as the active file.
- Code Editor:** Displays Python code for a Quarto dashboard. The code includes imports for polars, great\_tables, IPython.display, plotnine, numpy, ipyleaflet, and ipywidgets, along with code to read a CSV file named 'parismetro.csv'.
- Output Panel:** Shows the command 'format: dashboard'.
- Bottom Status Bar:** Shows 'Python 3.12.1 (PipEnv: .venv)' and the current working directory as 'C:\Users\DEV\_OTHER\paris-metro-dashboard'.
- Bottom Left:** A small 'PyData' logo.
- Right Side:** A map titled 'Paris Metro Explorer: Ridership trends by line, station, and year'. The map shows frequentation (nb of entering users) for stations in line 3 of Paris. The map includes labels for various Parisian neighborhoods and arrondissements.

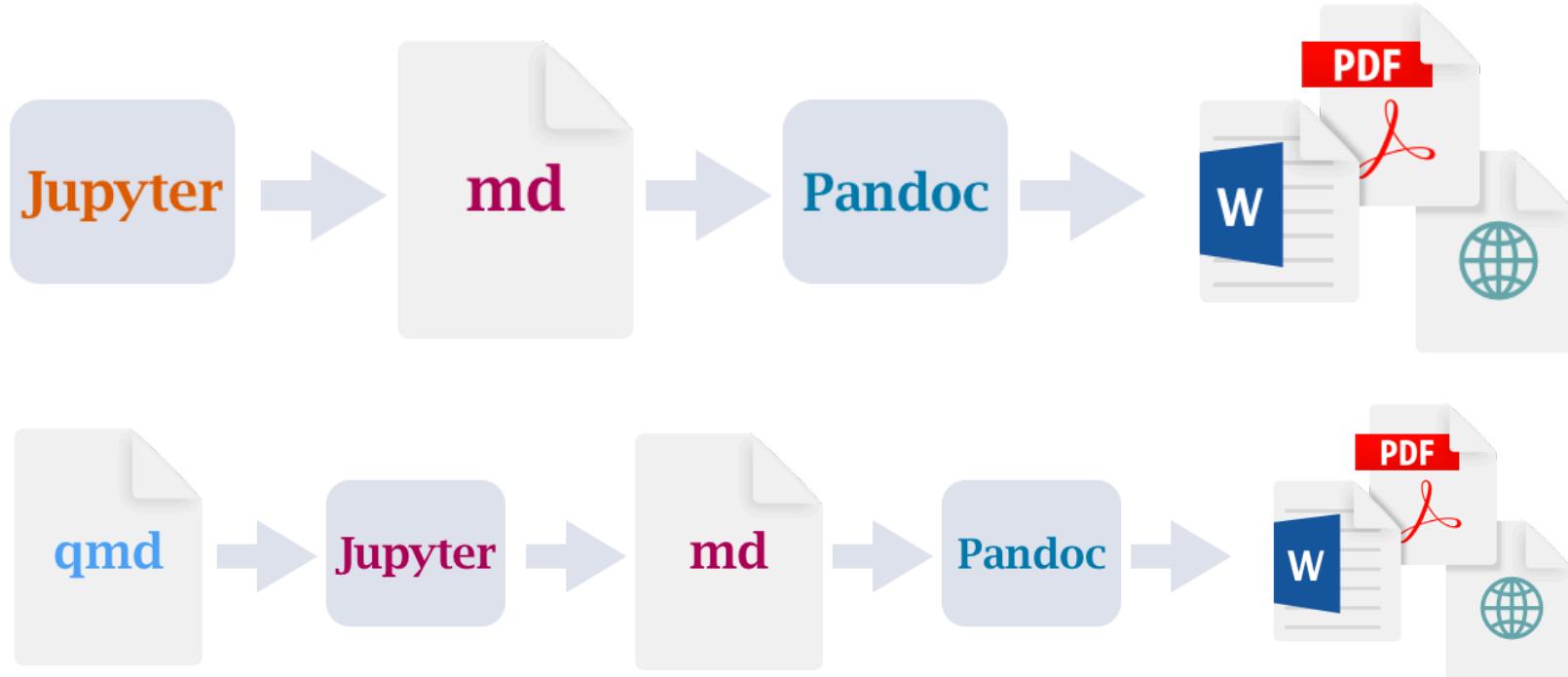
# Quarto Workflow



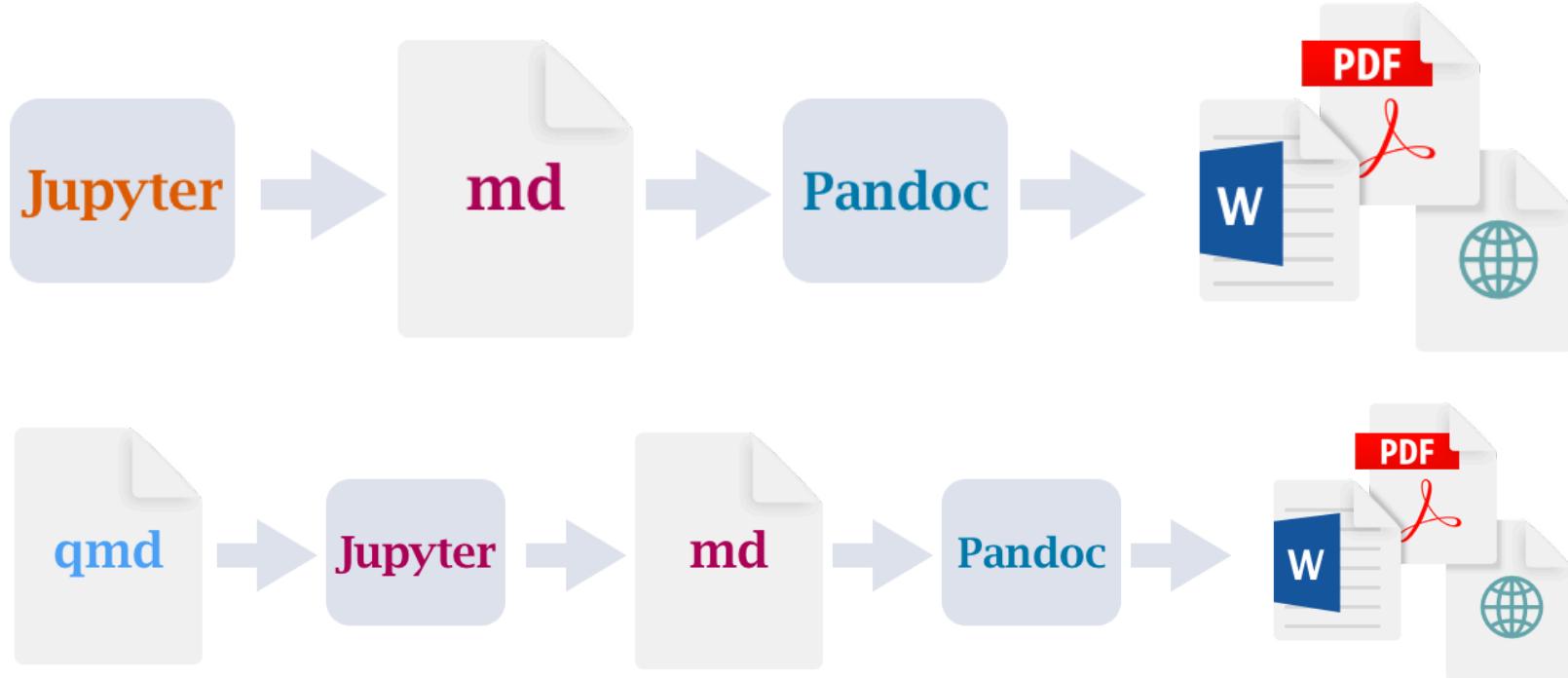
# Quarto Workflow



# Quarto Workflow

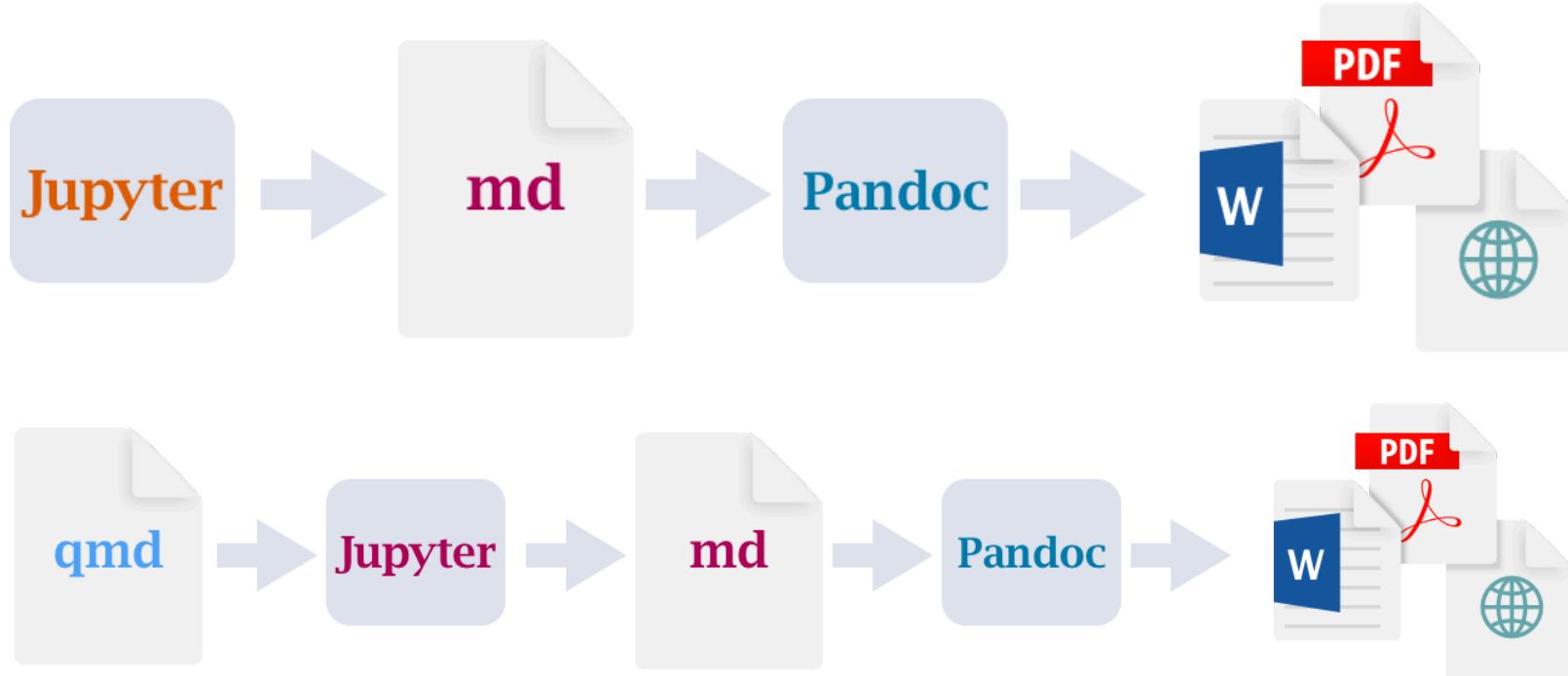


# Quarto Workflow



**Computations:** Jupyter  
(and Knitr and  
ObservableJS)

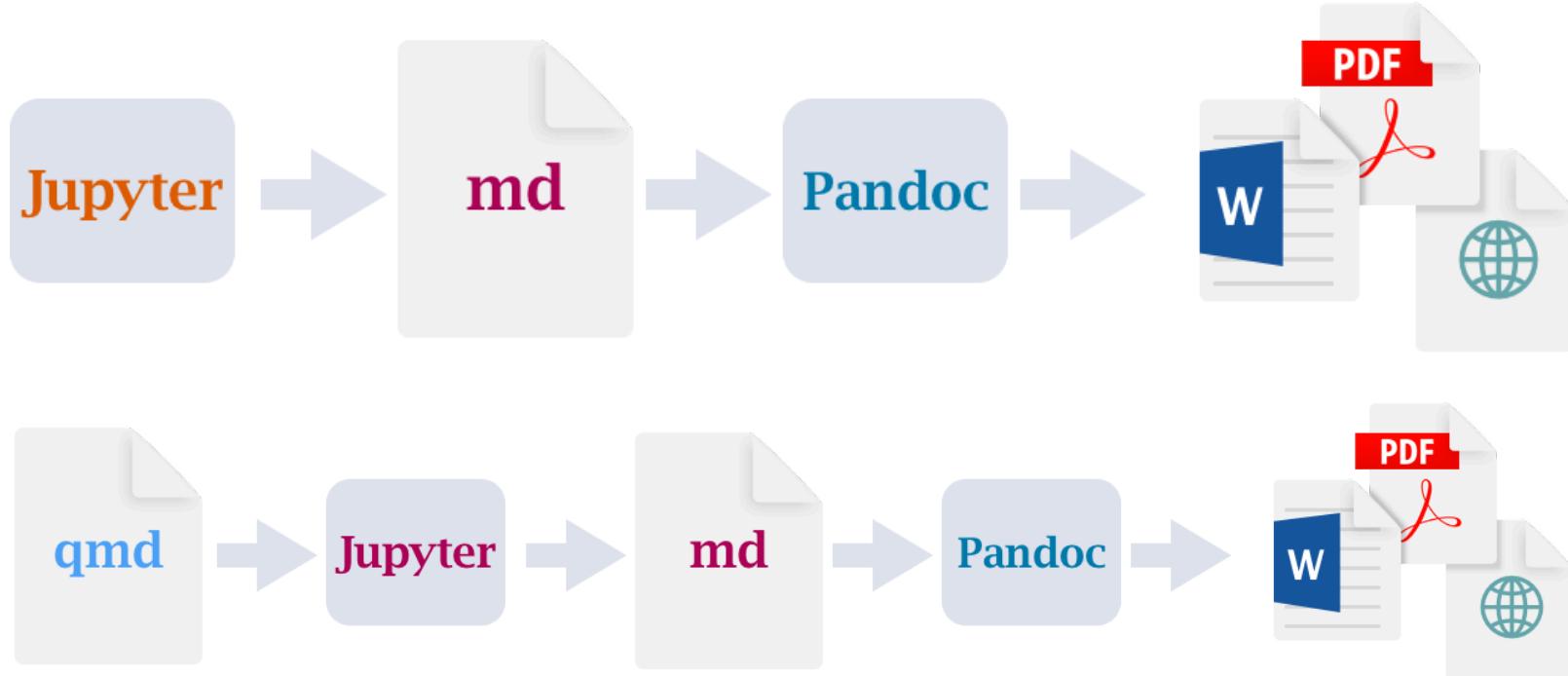
# Quarto Workflow



**Computations:** [Jupyter](#)  
(and [Knitr](#) and  
[ObservableJS](#))

**Markdown:** [Pandoc](#)  
with many enhancements

# Quarto Workflow

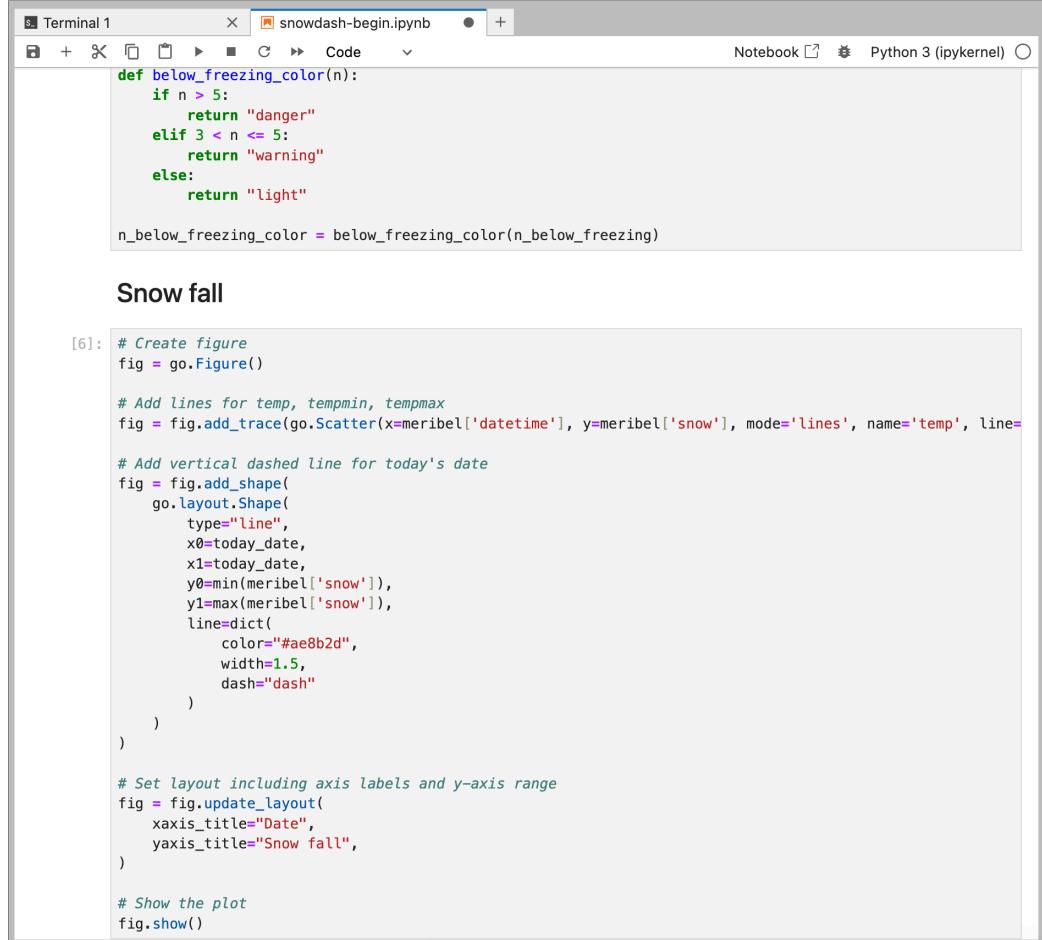


**Computations:** [Jupyter](#)  
(and [Knitr](#) and  
[ObservableJS](#))

**Markdown:** [Pandoc](#)  
with many enhancements

**Output:** Documents,  
presentations, websites,  
books, blogs, ...

## Render Notebook to HTML (default options)



The screenshot shows a Jupyter Notebook interface with two tabs: "Terminal 1" and "snowdash-begin.ipynb". The "snowdash-begin.ipynb" tab is active, displaying a Python script. The script defines a function `below_freezing_color` which returns "danger" if `n > 5`, "warning" if `3 < n <= 5`, and "light" otherwise. It then calls this function with `n_below_freezing`. Below this, a cell titled "Snow fall" contains code to create a plot. The code uses Plotly's `Figure` class to add a scatter plot of snow depth over time, and a vertical dashed line at today's date. It sets axis titles and shows the plot.

```
def below_freezing_color(n):
    if n > 5:
        return "danger"
    elif 3 < n <= 5:
        return "warning"
    else:
        return "light"

n_below_freezing_color = below_freezing_color(n_below_freezing)

Snow fall

[6]: # Create figure
fig = go.Figure()

# Add lines for temp, tempmin, tempmax
fig = fig.add_trace(go.Scatter(x=meribel['datetime'], y=meribel['snow'], mode='lines', name='temp', line=
# Add vertical dashed line for today's date
fig = fig.add_shape(
    go.layout.Shape(
        type="line",
        x0=today_date,
        x1=today_date,
        y0=min(meribel['snow']),
        y1=max(meribel['snow']),
        line=dict(
            color="#ae8b2d",
            width=1.5,
            dash="dash"
        )
    )
)
# Set layout including axis labels and y-axis range
fig = fig.update_layout(
    xaxis_title="Date",
    yaxis_title="Snow fall",
)
# Show the plot
fig.show()
```

Follow along at <https://cderv.github.io/pydata-paris-2024-quarto-dashboard/> and <https://quarto.org>

## Render Notebook to HTML (default options)

```
Terminal 1 x snowdash-begin.ipynb + Notebook Python 3 (ipykernel)
+ X □ ▶ C ▶ Code v

def below_freezing_color(n):
    if n > 5:
        return "danger"
    elif 3 < n <= 5:
        return "warning"
    else:
        return "light"

n_below_freezing_color = below_freezing_color(n_below_freezing)

Snow fall

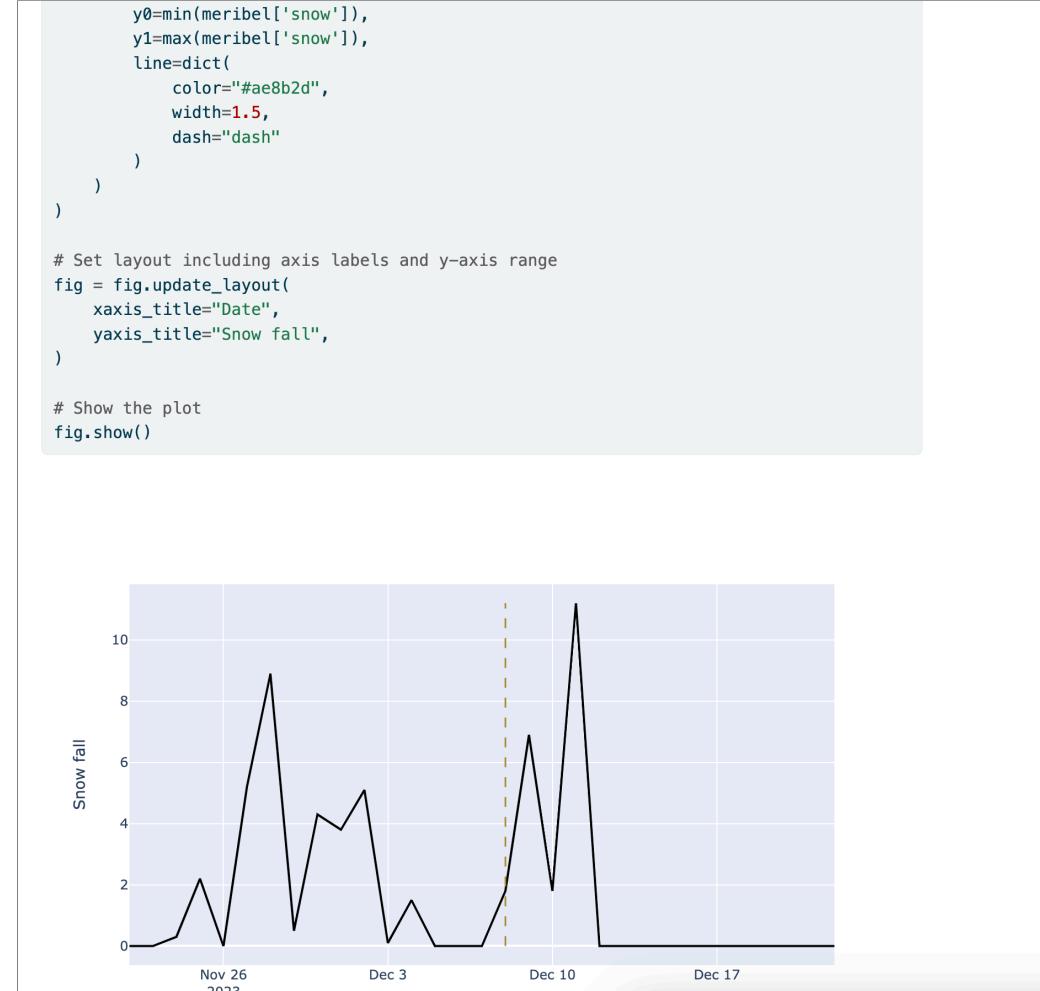
[6]: # Create figure
fig = go.Figure()

# Add lines for temp, tempmin, tempmax
fig = fig.add_trace(go.Scatter(x=meribel['datetime'], y=meribel['snow'], mode='lines', name='temp', line=))

# Add vertical dashed line for today's date
fig = fig.add_shape(
    go.layout.Shape(
        type="line",
        x0=today_date,
        x1=today_date,
        y0=min(meribel['snow']),
        y1=max(meribel['snow']),
        line=dict(
            color="#ae8b2d",
            width=1.5,
            dash="dash"
        )
    )
)

# Set layout including axis labels and y-axis range
fig = fig.update_layout(
    xaxis_title="Date",
    yaxis_title="Snow fall",
)

# Show the plot
fig.show()
```



Follow along at <https://cderv.github.io/pydata-paris-2024-quarto-dashboard/> and <https://quarto.org>

## Render Notebook to HTML (document level options)

The screenshot shows a Jupyter Notebook interface with several code cells and a rendered section. The top bar includes tabs for Terminal 1, snowdash-begin.ipynb, Notebook, and Python 3 (ipykernel). The code cells contain Python code for data processing and visualization, including imports for pandas, datetime, and plotly, and logic for calculating snowfall and temperature thresholds. A rendered section titled "Snow fall" is visible below the code.

```
---  
title: "Dashing through the snow *"  
format: html  
theme: minty  
highlight-style: atom-one  
---  
[1]: import pandas as pd  
from datetime import datetime  
import itables as itables  
import plotly.express as px  
import plotly.graph_objects as go  
[2]: today_string = "2023-12-08"  
[3]: meribel = pd.read_csv("data/Meribel.csv")  
meribel['datetime'] = pd.to_datetime(meribel['datetime'])  
  
stations = pd.read_csv("data/stations.csv")  
[4]: today_date = pd.to_datetime(today_string)  
[5]: n_snow = meribel[meribel['snow'] > 0].shape[0]  
  
n_below_freezing = meribel[meribel['temp'] < 32].shape[0]  
  
def below_freezing_color(n):  
    if n > 5:  
        return "danger"  
    elif 3 < n <= 5:  
        return "warning"  
    else:  
        return "light"  
  
n_below_freezing_color = below_freezing_color(n_below_freezing)  
[6]: # Create figure  
fig = go.Figure()  
[7]: fig  
[8]: fig  
[9]: fig  
[10]: fig  
[11]: fig  
[12]: fig  
[13]: fig  
[14]: fig  
[15]: fig  
[16]: fig  
[17]: fig  
[18]: fig  
[19]: fig  
[20]: fig  
[21]: fig  
[22]: fig  
[23]: fig  
[24]: fig  
[25]: fig  
[26]: fig  
[27]: fig  
[28]: fig  
[29]: fig  
[30]: fig  
[31]: fig  
[32]: fig  
[33]: fig  
[34]: fig  
[35]: fig  
[36]: fig  
[37]: fig  
[38]: fig  
[39]: fig  
[40]: fig  
[41]: fig  
[42]: fig  
[43]: fig  
[44]: fig  
[45]: fig  
[46]: fig  
[47]: fig  
[48]: fig  
[49]: fig  
[50]: fig  
[51]: fig  
[52]: fig  
[53]: fig  
[54]: fig  
[55]: fig  
[56]: fig  
[57]: fig  
[58]: fig  
[59]: fig  
[60]: fig  
[61]: fig  
[62]: fig  
[63]: fig  
[64]: fig  
[65]: fig  
[66]: fig  
[67]: fig  
[68]: fig  
[69]: fig  
[70]: fig  
[71]: fig  
[72]: fig  
[73]: fig  
[74]: fig  
[75]: fig  
[76]: fig  
[77]: fig  
[78]: fig  
[79]: fig  
[80]: fig  
[81]: fig  
[82]: fig  
[83]: fig  
[84]: fig  
[85]: fig  
[86]: fig  
[87]: fig  
[88]: fig  
[89]: fig  
[90]: fig  
[91]: fig  
[92]: fig  
[93]: fig  
[94]: fig  
[95]: fig  
[96]: fig  
[97]: fig  
[98]: fig  
[99]: fig  
[100]: fig  
[101]: fig  
[102]: fig  
[103]: fig  
[104]: fig  
[105]: fig  
[106]: fig  
[107]: fig  
[108]: fig  
[109]: fig  
[110]: fig  
[111]: fig  
[112]: fig  
[113]: fig  
[114]: fig  
[115]: fig  
[116]: fig  
[117]: fig  
[118]: fig  
[119]: fig  
[120]: fig  
[121]: fig  
[122]: fig  
[123]: fig  
[124]: fig  
[125]: fig  
[126]: fig  
[127]: fig  
[128]: fig  
[129]: fig  
[130]: fig  
[131]: fig  
[132]: fig  
[133]: fig  
[134]: fig  
[135]: fig  
[136]: fig  
[137]: fig  
[138]: fig  
[139]: fig  
[140]: fig  
[141]: fig  
[142]: fig  
[143]: fig  
[144]: fig  
[145]: fig  
[146]: fig  
[147]: fig  
[148]: fig  
[149]: fig  
[150]: fig  
[151]: fig  
[152]: fig  
[153]: fig  
[154]: fig  
[155]: fig  
[156]: fig  
[157]: fig  
[158]: fig  
[159]: fig  
[160]: fig  
[161]: fig  
[162]: fig  
[163]: fig  
[164]: fig  
[165]: fig  
[166]: fig  
[167]: fig  
[168]: fig  
[169]: fig  
[170]: fig  
[171]: fig  
[172]: fig  
[173]: fig  
[174]: fig  
[175]: fig  
[176]: fig  
[177]: fig  
[178]: fig  
[179]: fig  
[180]: fig  
[181]: fig  
[182]: fig  
[183]: fig  
[184]: fig  
[185]: fig  
[186]: fig  
[187]: fig  
[188]: fig  
[189]: fig  
[190]: fig  
[191]: fig  
[192]: fig  
[193]: fig  
[194]: fig  
[195]: fig  
[196]: fig  
[197]: fig  
[198]: fig  
[199]: fig  
[200]: fig  
[201]: fig  
[202]: fig  
[203]: fig  
[204]: fig  
[205]: fig  
[206]: fig  
[207]: fig  
[208]: fig  
[209]: fig  
[210]: fig  
[211]: fig  
[212]: fig  
[213]: fig  
[214]: fig  
[215]: fig  
[216]: fig  
[217]: fig  
[218]: fig  
[219]: fig  
[220]: fig  
[221]: fig  
[222]: fig  
[223]: fig  
[224]: fig  
[225]: fig  
[226]: fig  
[227]: fig  
[228]: fig  
[229]: fig  
[230]: fig  
[231]: fig  
[232]: fig  
[233]: fig  
[234]: fig  
[235]: fig  
[236]: fig  
[237]: fig  
[238]: fig  
[239]: fig  
[240]: fig  
[241]: fig  
[242]: fig  
[243]: fig  
[244]: fig  
[245]: fig  
[246]: fig  
[247]: fig  
[248]: fig  
[249]: fig  
[250]: fig  
[251]: fig  
[252]: fig  
[253]: fig  
[254]: fig  
[255]: fig  
[256]: fig  
[257]: fig  
[258]: fig  
[259]: fig  
[260]: fig  
[261]: fig  
[262]: fig  
[263]: fig  
[264]: fig  
[265]: fig  
[266]: fig  
[267]: fig  
[268]: fig  
[269]: fig  
[270]: fig  
[271]: fig  
[272]: fig  
[273]: fig  
[274]: fig  
[275]: fig  
[276]: fig  
[277]: fig  
[278]: fig  
[279]: fig  
[280]: fig  
[281]: fig  
[282]: fig  
[283]: fig  
[284]: fig  
[285]: fig  
[286]: fig  
[287]: fig  
[288]: fig  
[289]: fig  
[290]: fig  
[291]: fig  
[292]: fig  
[293]: fig  
[294]: fig  
[295]: fig  
[296]: fig  
[297]: fig  
[298]: fig  
[299]: fig  
[300]: fig  
[301]: fig  
[302]: fig  
[303]: fig  
[304]: fig  
[305]: fig  
[306]: fig  
[307]: fig  
[308]: fig  
[309]: fig  
[310]: fig  
[311]: fig  
[312]: fig  
[313]: fig  
[314]: fig  
[315]: fig  
[316]: fig  
[317]: fig  
[318]: fig  
[319]: fig  
[320]: fig  
[321]: fig  
[322]: fig  
[323]: fig  
[324]: fig  
[325]: fig  
[326]: fig  
[327]: fig  
[328]: fig  
[329]: fig  
[330]: fig  
[331]: fig  
[332]: fig  
[333]: fig  
[334]: fig  
[335]: fig  
[336]: fig  
[337]: fig  
[338]: fig  
[339]: fig  
[340]: fig  
[341]: fig  
[342]: fig  
[343]: fig  
[344]: fig  
[345]: fig  
[346]: fig  
[347]: fig  
[348]: fig  
[349]: fig  
[350]: fig  
[351]: fig  
[352]: fig  
[353]: fig  
[354]: fig  
[355]: fig  
[356]: fig  
[357]: fig  
[358]: fig  
[359]: fig  
[360]: fig  
[361]: fig  
[362]: fig  
[363]: fig  
[364]: fig  
[365]: fig  
[366]: fig  
[367]: fig  
[368]: fig  
[369]: fig  
[370]: fig  
[371]: fig  
[372]: fig  
[373]: fig  
[374]: fig  
[375]: fig  
[376]: fig  
[377]: fig  
[378]: fig  
[379]: fig  
[380]: fig  
[381]: fig  
[382]: fig  
[383]: fig  
[384]: fig  
[385]: fig  
[386]: fig  
[387]: fig  
[388]: fig  
[389]: fig  
[390]: fig  
[391]: fig  
[392]: fig  
[393]: fig  
[394]: fig  
[395]: fig  
[396]: fig  
[397]: fig  
[398]: fig  
[399]: fig  
[400]: fig  
[401]: fig  
[402]: fig  
[403]: fig  
[404]: fig  
[405]: fig  
[406]: fig  
[407]: fig  
[408]: fig  
[409]: fig  
[410]: fig  
[411]: fig  
[412]: fig  
[413]: fig  
[414]: fig  
[415]: fig  
[416]: fig  
[417]: fig  
[418]: fig  
[419]: fig  
[420]: fig  
[421]: fig  
[422]: fig  
[423]: fig  
[424]: fig  
[425]: fig  
[426]: fig  
[427]: fig  
[428]: fig  
[429]: fig  
[430]: fig  
[431]: fig  
[432]: fig  
[433]: fig  
[434]: fig  
[435]: fig  
[436]: fig  
[437]: fig  
[438]: fig  
[439]: fig  
[440]: fig  
[441]: fig  
[442]: fig  
[443]: fig  
[444]: fig  
[445]: fig  
[446]: fig  
[447]: fig  
[448]: fig  
[449]: fig  
[450]: fig  
[451]: fig  
[452]: fig  
[453]: fig  
[454]: fig  
[455]: fig  
[456]: fig  
[457]: fig  
[458]: fig  
[459]: fig  
[460]: fig  
[461]: fig  
[462]: fig  
[463]: fig  
[464]: fig  
[465]: fig  
[466]: fig  
[467]: fig  
[468]: fig  
[469]: fig  
[470]: fig  
[471]: fig  
[472]: fig  
[473]: fig  
[474]: fig  
[475]: fig  
[476]: fig  
[477]: fig  
[478]: fig  
[479]: fig  
[480]: fig  
[481]: fig  
[482]: fig  
[483]: fig  
[484]: fig  
[485]: fig  
[486]: fig  
[487]: fig  
[488]: fig  
[489]: fig  
[490]: fig  
[491]: fig  
[492]: fig  
[493]: fig  
[494]: fig  
[495]: fig  
[496]: fig  
[497]: fig  
[498]: fig  
[499]: fig  
[500]: fig  
[501]: fig  
[502]: fig  
[503]: fig  
[504]: fig  
[505]: fig  
[506]: fig  
[507]: fig  
[508]: fig  
[509]: fig  
[510]: fig  
[511]: fig  
[512]: fig  
[513]: fig  
[514]: fig  
[515]: fig  
[516]: fig  
[517]: fig  
[518]: fig  
[519]: fig  
[520]: fig  
[521]: fig  
[522]: fig  
[523]: fig  
[524]: fig  
[525]: fig  
[526]: fig  
[527]: fig  
[528]: fig  
[529]: fig  
[530]: fig  
[531]: fig  
[532]: fig  
[533]: fig  
[534]: fig  
[535]: fig  
[536]: fig  
[537]: fig  
[538]: fig  
[539]: fig  
[540]: fig  
[541]: fig  
[542]: fig  
[543]: fig  
[544]: fig  
[545]: fig  
[546]: fig  
[547]: fig  
[548]: fig  
[549]: fig  
[550]: fig  
[551]: fig  
[552]: fig  
[553]: fig  
[554]: fig  
[555]: fig  
[556]: fig  
[557]: fig  
[558]: fig  
[559]: fig  
[560]: fig  
[561]: fig  
[562]: fig  
[563]: fig  
[564]: fig  
[565]: fig  
[566]: fig  
[567]: fig  
[568]: fig  
[569]: fig  
[570]: fig  
[571]: fig  
[572]: fig  
[573]: fig  
[574]: fig  
[575]: fig  
[576]: fig  
[577]: fig  
[578]: fig  
[579]: fig  
[580]: fig  
[581]: fig  
[582]: fig  
[583]: fig  
[584]: fig  
[585]: fig  
[586]: fig  
[587]: fig  
[588]: fig  
[589]: fig  
[590]: fig  
[591]: fig  
[592]: fig  
[593]: fig  
[594]: fig  
[595]: fig  
[596]: fig  
[597]: fig  
[598]: fig  
[599]: fig  
[600]: fig  
[601]: fig  
[602]: fig  
[603]: fig  
[604]: fig  
[605]: fig  
[606]: fig  
[607]: fig  
[608]: fig  
[609]: fig  
[610]: fig  
[611]: fig  
[612]: fig  
[613]: fig  
[614]: fig  
[615]: fig  
[616]: fig  
[617]: fig  
[618]: fig  
[619]: fig  
[620]: fig  
[621]: fig  
[622]: fig  
[623]: fig  
[624]: fig  
[625]: fig  
[626]: fig  
[627]: fig  
[628]: fig  
[629]: fig  
[630]: fig  
[631]: fig  
[632]: fig  
[633]: fig  
[634]: fig  
[635]: fig  
[636]: fig  
[637]: fig  
[638]: fig  
[639]: fig  
[640]: fig  
[641]: fig  
[642]: fig  
[643]: fig  
[644]: fig  
[645]: fig  
[646]: fig  
[647]: fig  
[648]: fig  
[649]: fig  
[650]: fig  
[651]: fig  
[652]: fig  
[653]: fig  
[654]: fig  
[655]: fig  
[656]: fig  
[657]: fig  
[658]: fig  
[659]: fig  
[660]: fig  
[661]: fig  
[662]: fig  
[663]: fig  
[664]: fig  
[665]: fig  
[666]: fig  
[667]: fig  
[668]: fig  
[669]: fig  
[670]: fig  
[671]: fig  
[672]: fig  
[673]: fig  
[674]: fig  
[675]: fig  
[676]: fig  
[677]: fig  
[678]: fig  
[679]: fig  
[680]: fig  
[681]: fig  
[682]: fig  
[683]: fig  
[684]: fig  
[685]: fig  
[686]: fig  
[687]: fig  
[688]: fig  
[689]: fig  
[690]: fig  
[691]: fig  
[692]: fig  
[693]: fig  
[694]: fig  
[695]: fig  
[696]: fig  
[697]: fig  
[698]: fig  
[699]: fig  
[700]: fig  
[701]: fig  
[702]: fig  
[703]: fig  
[704]: fig  
[705]: fig  
[706]: fig  
[707]: fig  
[708]: fig  
[709]: fig  
[710]: fig  
[711]: fig  
[712]: fig  
[713]: fig  
[714]: fig  
[715]: fig  
[716]: fig  
[717]: fig  
[718]: fig  
[719]: fig  
[720]: fig  
[721]: fig  
[722]: fig  
[723]: fig  
[724]: fig  
[725]: fig  
[726]: fig  
[727]: fig  
[728]: fig  
[729]: fig  
[730]: fig  
[731]: fig  
[732]: fig  
[733]: fig  
[734]: fig  
[735]: fig  
[736]: fig  
[737]: fig  
[738]: fig  
[739]: fig  
[740]: fig  
[741]: fig  
[742]: fig  
[743]: fig  
[744]: fig  
[745]: fig  
[746]: fig  
[747]: fig  
[748]: fig  
[749]: fig  
[750]: fig  
[751]: fig  
[752]: fig  
[753]: fig  
[754]: fig  
[755]: fig  
[756]: fig  
[757]: fig  
[758]: fig  
[759]: fig  
[760]: fig  
[761]: fig  
[762]: fig  
[763]: fig  
[764]: fig  
[765]: fig  
[766]: fig  
[767]: fig  
[768]: fig  
[769]: fig  
[770]: fig  
[771]: fig  
[772]: fig  
[773]: fig  
[774]: fig  
[775]: fig  
[776]: fig  
[777]: fig  
[778]: fig  
[779]: fig  
[780]: fig  
[781]: fig  
[782]: fig  
[783]: fig  
[784]: fig  
[785]: fig  
[786]: fig  
[787]: fig  
[788]: fig  
[789]: fig  
[790]: fig  
[791]: fig  
[792]: fig  
[793]: fig  
[794]: fig  
[795]: fig  
[796]: fig  
[797]: fig  
[798]: fig  
[799]: fig  
[800]: fig  
[801]: fig  
[802]: fig  
[803]: fig  
[804]: fig  
[805]: fig  
[806]: fig  
[807]: fig  
[808]: fig  
[809]: fig  
[810]: fig  
[811]: fig  
[812]: fig  
[813]: fig  
[814]: fig  
[815]: fig  
[816]: fig  
[817]: fig  
[818]: fig  
[819]: fig  
[820]: fig  
[821]: fig  
[822]: fig  
[823]: fig  
[824]: fig  
[825]: fig  
[826]: fig  
[827]: fig  
[828]: fig  
[829]: fig  
[830]: fig  
[831]: fig  
[832]: fig  
[833]: fig  
[834]: fig  
[835]: fig  
[836]: fig  
[837]: fig  
[838]: fig  
[839]: fig  
[840]: fig  
[841]: fig  
[842]: fig  
[843]: fig  
[844]: fig  
[845]: fig  
[846]: fig  
[847]: fig  
[848]: fig  
[849]: fig  
[850]: fig  
[851]: fig  
[852]: fig  
[853]: fig  
[854]: fig  
[855]: fig  
[856]: fig  
[857]: fig  
[858]: fig  
[859]: fig  
[860]: fig  
[861]: fig  
[862]: fig  
[863]: fig  
[864]: fig  
[865]: fig  
[866]: fig  
[867]: fig  
[868]: fig  
[869]: fig  
[870]: fig  
[871]: fig  
[872]: fig  
[873]: fig  
[874]: fig  
[875]: fig  
[876]: fig  
[877]: fig  
[878]: fig  
[879]: fig  
[880]: fig  
[881]: fig  
[882]: fig  
[883]: fig  
[884]: fig  
[885]: fig  
[886]: fig  
[887]: fig  
[888]: fig  
[889]: fig  
[890]: fig  
[891]: fig  
[892]: fig  
[893]: fig  
[894]: fig  
[895]: fig  
[896]: fig  
[897]: fig  
[898]: fig  
[899]: fig  
[900]: fig  
[901]: fig  
[902]: fig  
[903]: fig  
[904]: fig  
[905]: fig  
[906]: fig  
[907]: fig  
[908]: fig  
[909]: fig  
[910]: fig  
[911]: fig  
[912]: fig  
[913]: fig  
[914]: fig  
[915]: fig  
[916]: fig  
[917]: fig  
[918]: fig  
[919]: fig  
[920]: fig  
[921]: fig  
[922]: fig  
[923]: fig  
[924]: fig  
[925]: fig  
[926]: fig  
[927]: fig  
[928]: fig  
[929]: fig  
[930]: fig  
[931]: fig  
[932]: fig  
[933]: fig  
[934]: fig  
[935]: fig  
[936]: fig  
[937]: fig  
[938]: fig  
[939]: fig  
[940]: fig  
[941]: fig  
[942]: fig  
[943]: fig  
[944]: fig  
[945]: fig  
[946]: fig  
[947]: fig  
[948]: fig  
[949]: fig  
[950]: fig  
[951]: fig  
[952]: fig  
[953]: fig  
[954]: fig  
[955]: fig  
[956]: fig  
[957]: fig  
[958]: fig  
[959]: fig  
[960]: fig  
[961]: fig  
[962]: fig  
[963]: fig  
[964]: fig  
[965]: fig  
[966]: fig  
[967]: fig  
[968]: fig  
[969]: fig  
[970]: fig  
[971]: fig  
[972]: fig  
[973]: fig  
[974]: fig  
[975]: fig  
[976]: fig  
[977]: fig  
[978]: fig  
[979]: fig  
[980]: fig  
[981]: fig  
[982]: fig  
[983]: fig  
[984]: fig  
[985]: fig  
[986]: fig  
[987]: fig  
[988]: fig  
[989]: fig  
[990]: fig  
[991]: fig  
[992]: fig  
[993]: fig  
[994]: fig  
[995]: fig  
[996]: fig  
[997]: fig  
[998]: fig  
[999]: fig  
[1000]: fig
```

## Snow fall

```
[6]: # Create figure  
fig = go.Figure()
```

Follow along at <https://cderv.github.io/pydata-paris-2024-quarto-dashboard/> and <https://quarto.org>



## Render Notebook to HTML (document level options)

The screenshot shows a Jupyter Notebook interface with several code cells and a rendered output section.

**Code Cells:**

- [1]:

```
---  
title: "Dashing through the snow *"  
format: html  
theme: minty  
highlight-style: atom-one  
---
```
- [2]:

```
import pandas as pd  
from datetime import datetime  
import itables as itables  
import plotly.express as px  
import plotly.graph_objects as go
```
- [3]:

```
today_string = "2023-12-08"
```
- [4]:

```
meribel = pd.read_csv("data/Meribel.csv")  
meribel['datetime'] = pd.to_datetime(meribel['datetime'])
```
- [5]:

```
stations = pd.read_csv("data/stations.csv")
```
- [6]:

```
today_date = pd.to_datetime(today_string)
```
- [7]:

```
n_snow = meribel[meribel['snow'] > 0].shape[0]
```
- [8]:

```
n_below_freezing = meribel[meribel['temp'] < 32].shape[0]
```
- [9]:

```
def below_freezing_color(n):  
    if n > 5:  
        return "danger"  
    elif 3 < n <= 5:  
        return "warning"  
    else:  
        return "light"
```
- [10]:

```
n_below_freezing_color = below_freezing_color(n_below_freezing)
```
- [11]:

```
# Create figure  
fig = go.Figure()
```

**Output Section:**

### Snow fall

```
# Create figure
fig = go.Figure()

# Add lines for temp, tempmin, tempmax
fig = fig.add_trace(go.Scatter(x=meribel['datetime'], y=meribel['snow'], mode='lines', n
```

## Dashing through the snow ❄️

```
import pandas as pd
from datetime import datetime
import itables as itables
import plotly.express as px
import plotly.graph_objects as go

today_string = "2023-12-08"

meribel = pd.read_csv("data/Meribel.csv")
meribel['datetime'] = pd.to_datetime(meribel['datetime'])

stations = pd.read_csv("data/stations.csv")

today_date = pd.to_datetime(today_string)

n_snow = meribel[meribel['snow'] > 0].shape[0]

n_below_freezing = meribel[meribel['temp'] < 32].shape[0]

def below_freezing_color(n):
    if n > 5:
        return "danger"
    elif 3 < n <= 5:
        return "warning"
    else:
        return "light"

n_below_freezing_color = below_freezing_color(n_below_freezing)



### Snow fall



```
# Create figure
fig = go.Figure()

# Add lines for temp, tempmin, tempmax
fig = fig.add_trace(go.Scatter(x=meribel['datetime'], y=meribel['snow'], mode='lines', n
```


```

Follow along at <https://cderv.github.io/pydata-paris-2024-quarto-dashboard/> and <https://quarto.org>

## Render Notebook to HTML (document and cell level options)

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

- Terminal 1**: A terminal window showing configuration settings:

```
---  
title: "Dashing through the snow *"  
format: html  
theme: minty  
highlight-style: atom-one  
code-fold: true  
code-tools: true  
---
```
- Cells:**
  - [1]:

```
import pandas as pd  
from datetime import datetime  
import ipandas as ipandas  
import plotly.express as px  
import plotly.graph_objects as go
```
  - [2]:

```
today_string = "2023-12-08"
```
  - [3]:

```
meribel = pd.read_csv("data/Meribel.csv")  
meribel['datetime'] = pd.to_datetime(meribel['datetime'])  
  
stations = pd.read_csv("data/stations.csv")
```
  - [4]:

```
today_date = pd.to_datetime(today_string)
```
  - [5]:

```
n_snow = meribel[meribel['snow'] > 0].shape[0]  
  
n_below_freezing = meribel[meribel['temp'] < 32].shape[0]  
  
def below_freezing_color(n):  
    if n > 5:  
        return "danger"  
    elif 3 < n <= 5:  
        return "warning"  
    else:  
        return "light"  
  
n_below_freezing_color = below_freezing_color(n_below_freezing)
```
  - Snow fall**: A rendered output cell containing the text "Snow fall".
  - [6]:

```
#| label: fig-snow-fall  
#| fig-cap: "Snow fall in Meribel"
```

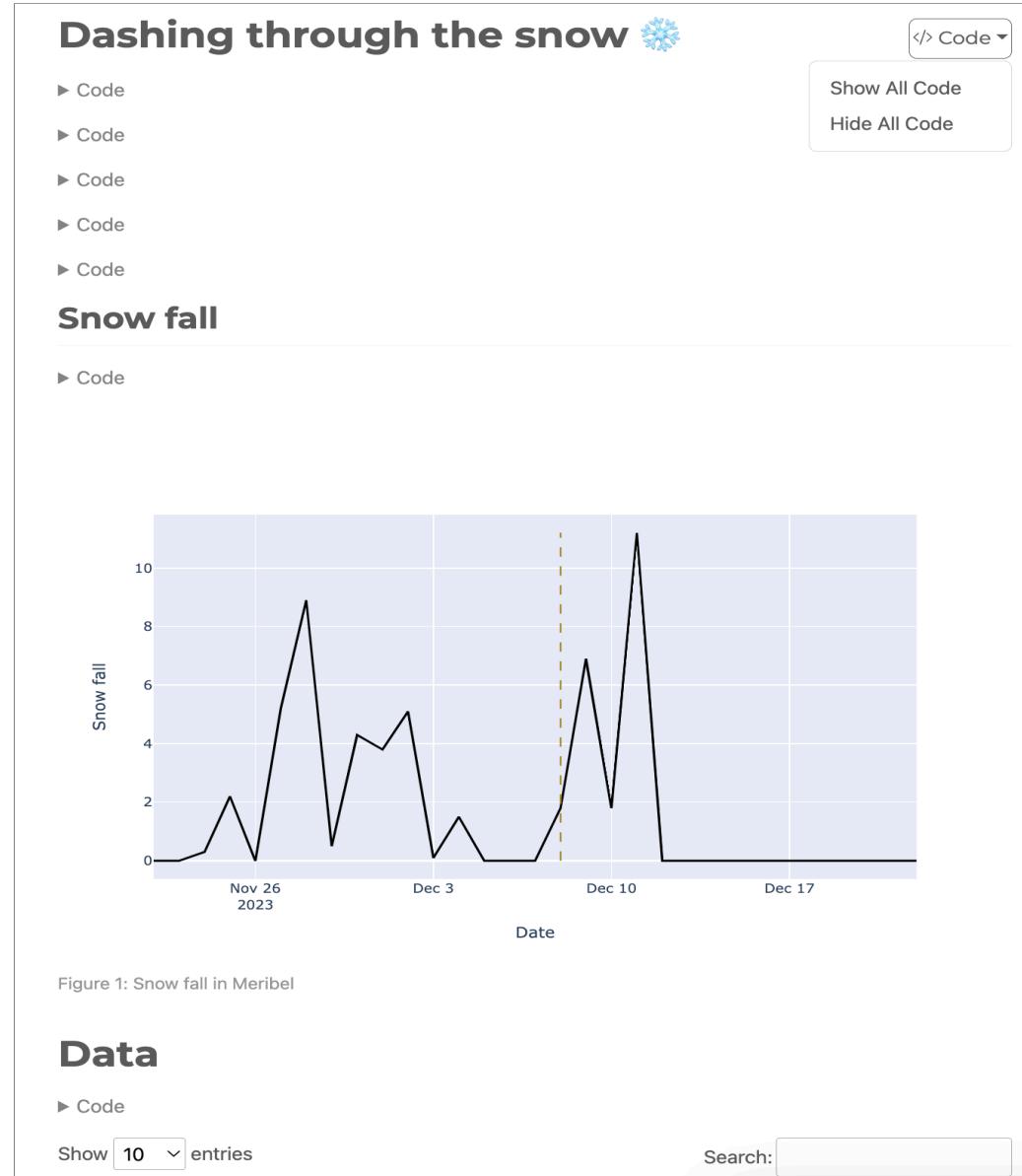
Follow along at <https://cderv.github.io/pydata-paris-2024-quarto-dashboard/> and <https://quarto.org>

## Render Notebook to HTML (document and cell level options)

The screenshot shows a Jupyter Notebook interface with several code cells and one rendered output cell.

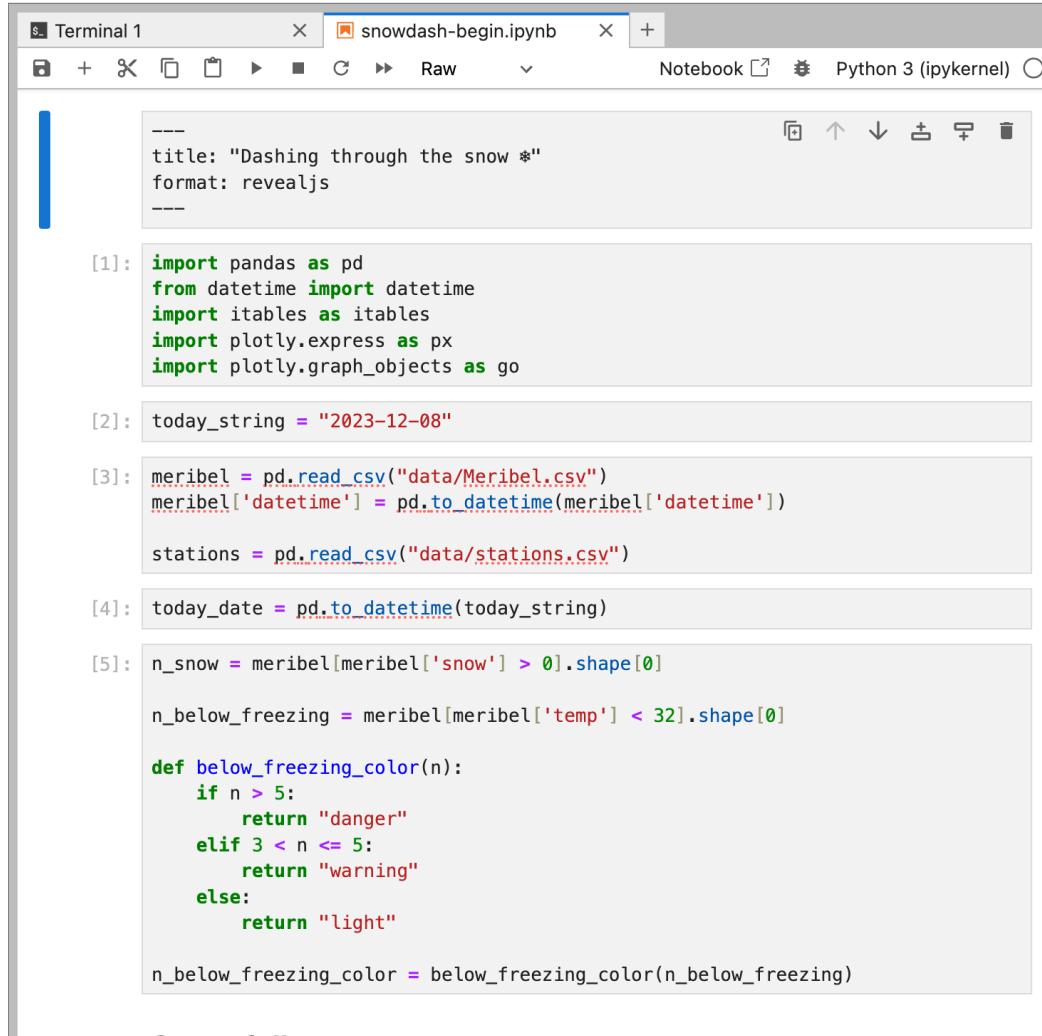
Cells 1-5 contain standard Python code for reading CSV files, creating dataframes, and defining a function to color snowfall based on temperature. Cell 6 contains a single-line comment starting with '#|'. Cell 7 contains the rendered output of the code in Cell 6, which is a title 'Snow fall' followed by a plot.

```
---  
title: "Dashing through the snow *"  
format: html  
theme: minty  
highlight-style: atom-one  
code-fold: true  
code-tools: true  
---  
[1]:  
import pandas as pd  
from datetime import datetime  
import itables as itables  
import plotly.express as px  
import plotly.graph_objects as go  
  
[2]: today_string = "2023-12-08"  
  
[3]: meribel = pd.read_csv("data/Meribel.csv")  
meribel['datetime'] = pd.to_datetime(meribel['datetime'])  
  
stations = pd.read_csv("data/stations.csv")  
  
[4]: today_date = pd.to_datetime(today_string)  
  
[5]: n_snow = meribel[meribel['snow'] > 0].shape[0]  
  
n_below_freezing = meribel[meribel['temp'] < 32].shape[0]  
  
def below_freezing_color(n):  
    if n > 5:  
        return "danger"  
    elif 3 < n <= 5:  
        return "warning"  
    else:  
        return "light"  
  
n_below_freezing_color = below_freezing_color(n_below_freezing)  
  
Snow fall  
  
[6]: #| label: fig-snow-fall  
#| fia-cap: "Snow fall in Meribel"
```



Follow along at <https://cderv.github.io/pydata-paris-2024-quarto-dashboard/> and <https://quarto.org>

## Render Notebook to Revealjs – <https://quarto.org/docs/presentations/revealjs/>



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

- Toolbar:** Includes icons for file operations (New, Open, Save, etc.), cell selection, and kernel management.
- Title Bar:** Shows "Terminal 1" and "snowdash-begin.ipynb".
- Cell 1:** Metadata cell containing:

```
---
```

```
title: "Dashing through the snow *"
format: revealjs
---
```
- Cell 2:** Python code cell containing:

```
[1]: import pandas as pd
from datetime import datetime
import ipandas as ipandas
import plotly.express as px
import plotly.graph_objects as go
```
- Cell 3:** Python code cell containing:

```
[2]: today_string = "2023-12-08"
```
- Cell 4:** Python code cell containing:

```
[3]: meribel = pd.read_csv("data/Meribel.csv")
meribel['datetime'] = pd.to_datetime(meribel['datetime'])

stations = pd.read_csv("data/stations.csv")
```
- Cell 5:** Python code cell containing:

```
[4]: today_date = pd.to_datetime(today_string)

n_snow = meribel[meribel['snow'] > 0].shape[0]

n_below_freezing = meribel[meribel['temp'] < 32].shape[0]

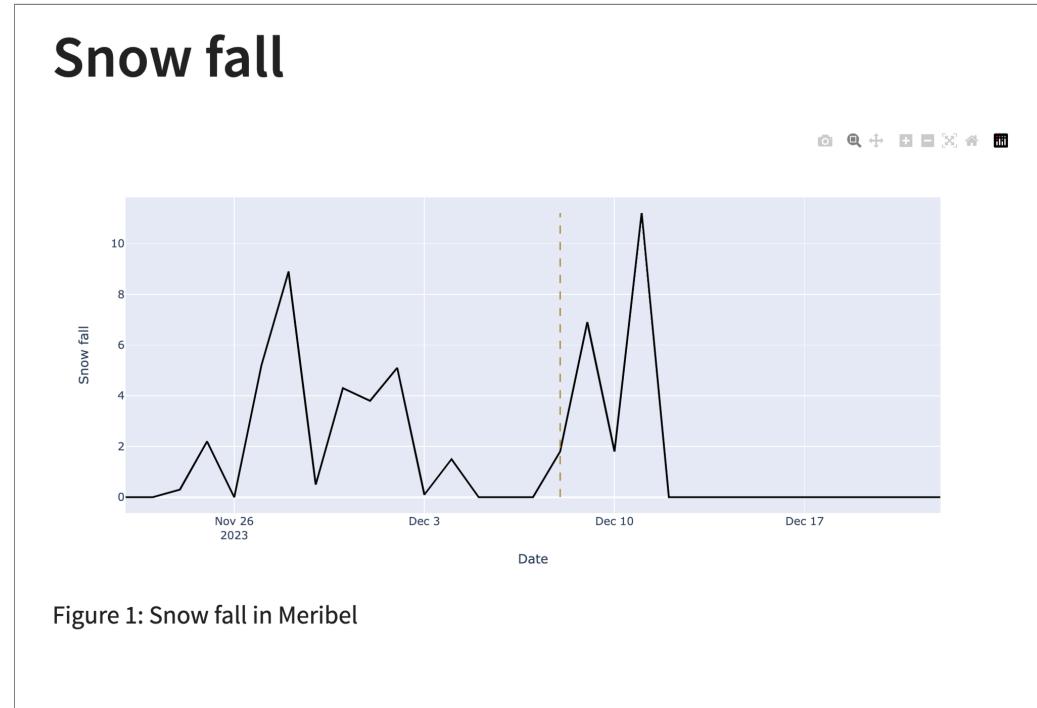
def below_freezing_color(n):
    if n > 5:
        return "danger"
    elif 3 < n <= 5:
        return "warning"
    else:
        return "light"

n_below_freezing_color = below_freezing_color(n_below_freezing)
```

## Render Notebook to Revealjs – <https://quarto.org/docs/presentations/revealjs/>

The screenshot shows a Jupyter Notebook interface with the title "snowdash-begin.ipynb". The notebook contains the following code:

```
---  
title: "Dashing through the snow *"  
format: revealjs  
---  
  
[1]:  
import pandas as pd  
from datetime import datetime  
import ipandas as ipandas  
import plotly.express as px  
import plotly.graph_objects as go  
  
[2]:  
today_string = "2023-12-08"  
  
[3]:  
meribel = pd.read_csv("data/Meribel.csv")  
meribel['datetime'] = pd.to_datetime(meribel['datetime'])  
  
stations = pd.read_csv("data/stations.csv")  
  
[4]:  
today_date = pd.to_datetime(today_string)  
  
[5]:  
n_snow = meribel[meribel['snow'] > 0].shape[0]  
  
n_below_freezing = meribel[meribel['temp'] < 32].shape[0]  
  
def below_freezing_color(n):  
    if n > 5:  
        return "danger"  
    elif 3 < n <= 5:  
        return "warning"  
    else:  
        return "light"  
  
n_below_freezing_color = below_freezing_color(n_below_freezing)
```



# Peeking at Dashboard Examples

Follow along at <https://cderv.github.io/pydata-paris-2024-quarto-dashboard/> and <https://quarto.org>





Churn rate 2022

17%



Current churn rate

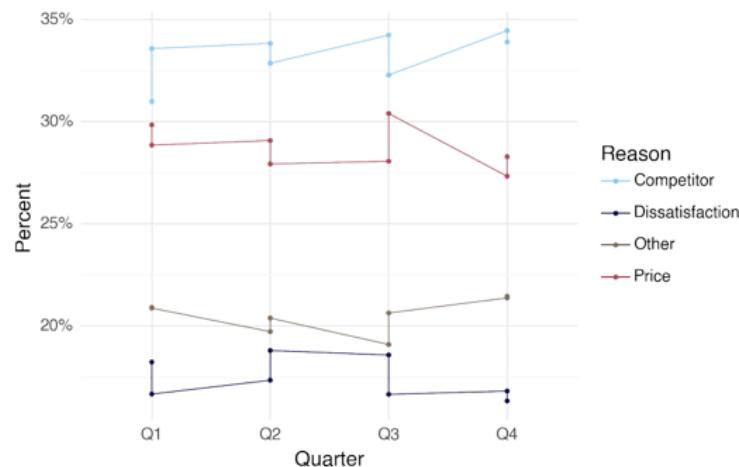
14%



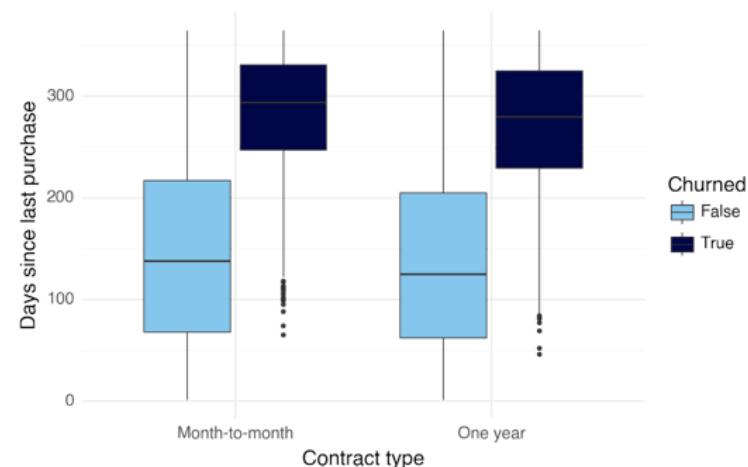
Churn rate goal

10%

Reason for churn by quarter



Churn by contract type and days since purchase



Purchase characteristics

index	Contract type	Churn	Average purchase	Total transactions	Days since last purchase
2	Month-to-month	Did not churn	50.14	19	138
5	Month-to-month	Churned	50.01	16	294
8	One year	Did not churn	50.05	2	125
11	One year	Churned	50.15	2	280



This dashboard displays weather data for:

Méribel, Les Alluessa, Auvergne-Rhône-Alpes, France

The data were gathered from the Virtual Crossing.

Today

2023-12-08



Snowy days

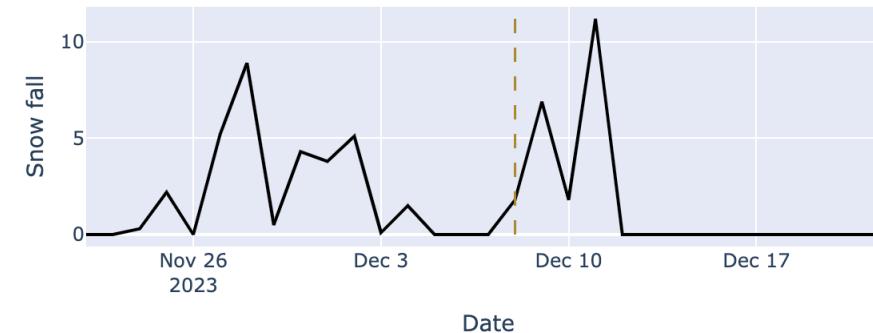
14



Number of days below freezing

7

Snow fall (in)



Snow depth (in)

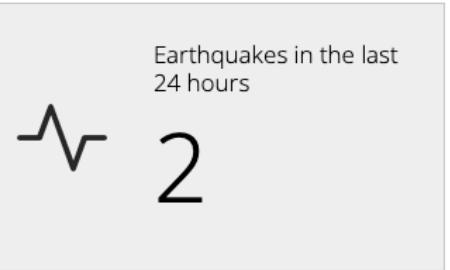


Temperature (F)



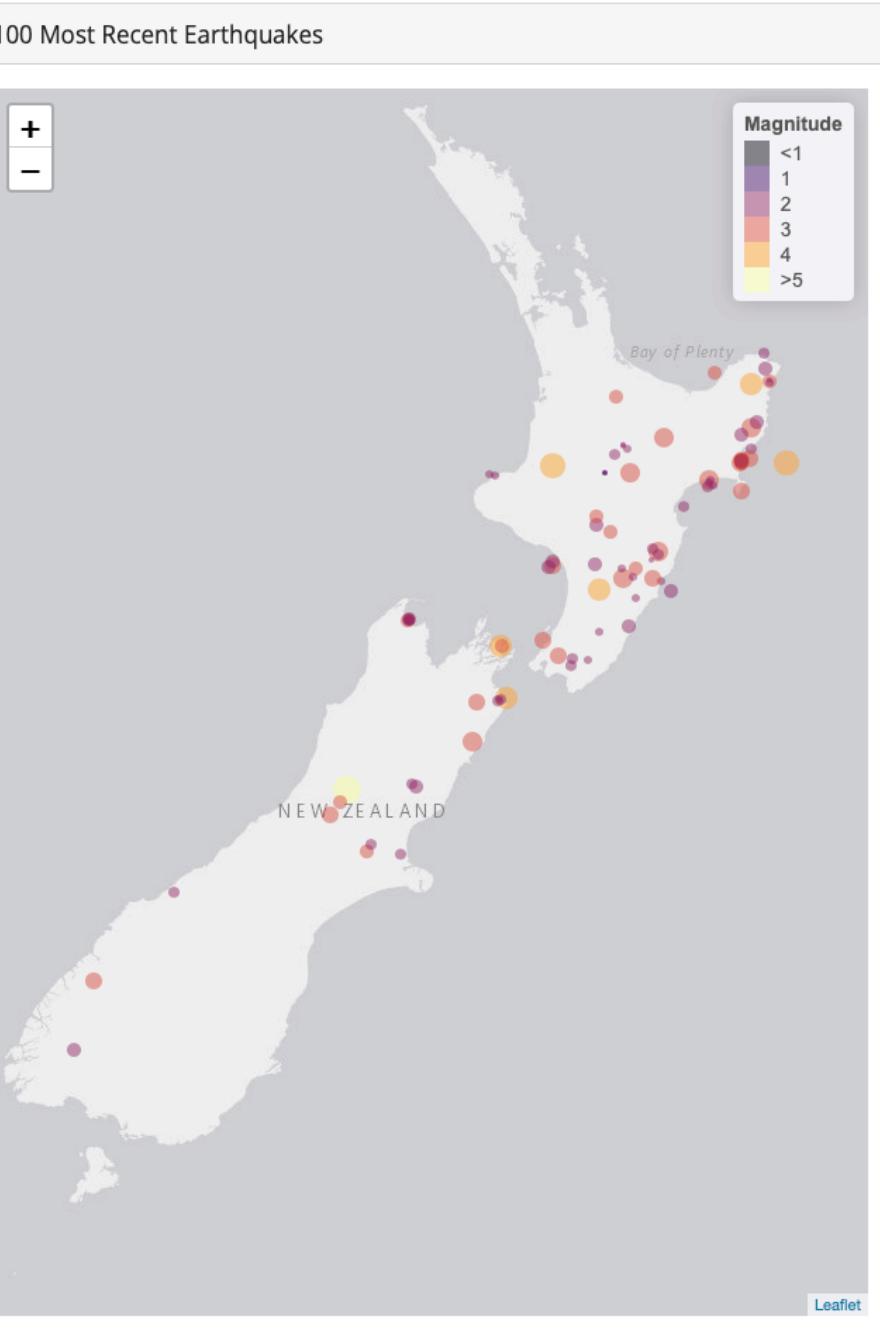
Feels like (F)



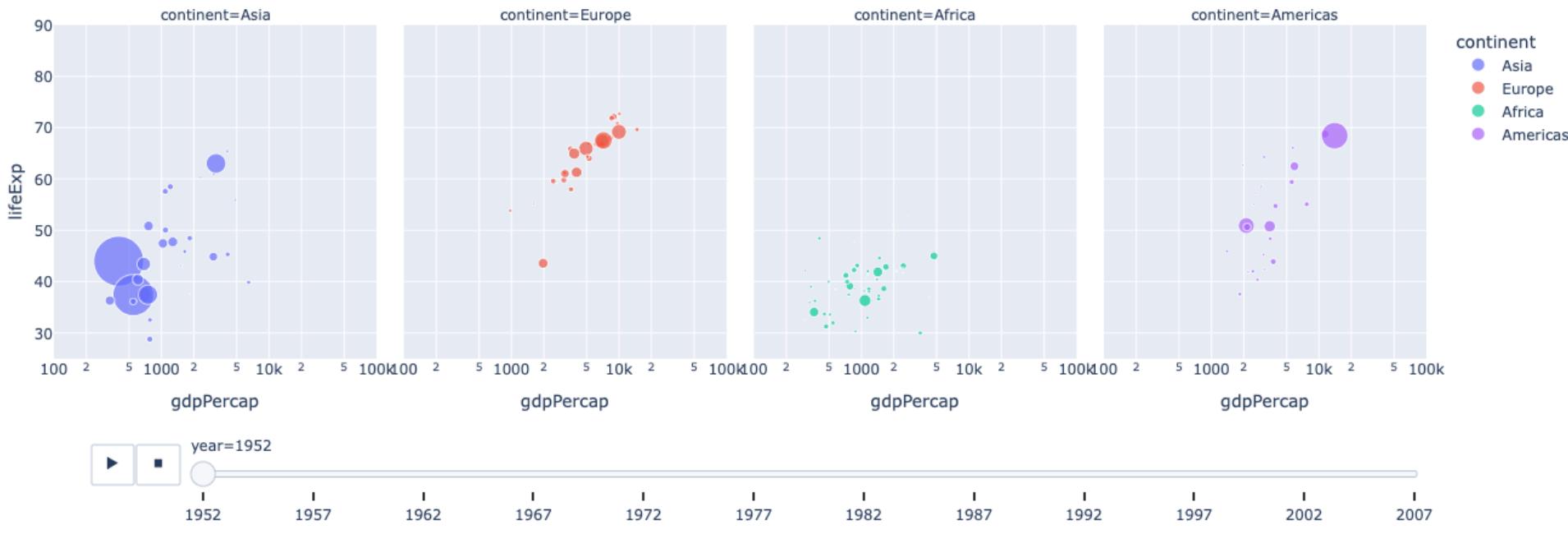


Last 10 Earthquakes					
Magnitude	Date	Time	Location	Depth	
2.4	Yesterday	03:37 PM	5 km north-west of New Plymouth	8 km	
3.6	Yesterday	11:18 AM	Within 5 km of Wairoa	6 km	
3.1	Yesterday	12:43 AM	15 km north-east of Tokoroa	10 km	
3.0	2 days ago	03:38 AM	20 km south-east of Porangahau	38 km	
4.2	2 days ago	02:44 AM	15 km east of Seddon	9 km	
3.2	4 days ago	06:18 AM	20 km south of Whanganui	17 km	
2.3	5 days ago	04:17 PM	10 km north-east of Pongaroa	15 km	
2.7	5 days ago	02:06 AM	5 km south-east of Gisborne	28 km	
3.7	6 days ago	03:23 AM	5 km south of Dannevirke	6 km	
3.0	6 days ago	03:02 AM	10 km south-west of Te Araroa	10 km	

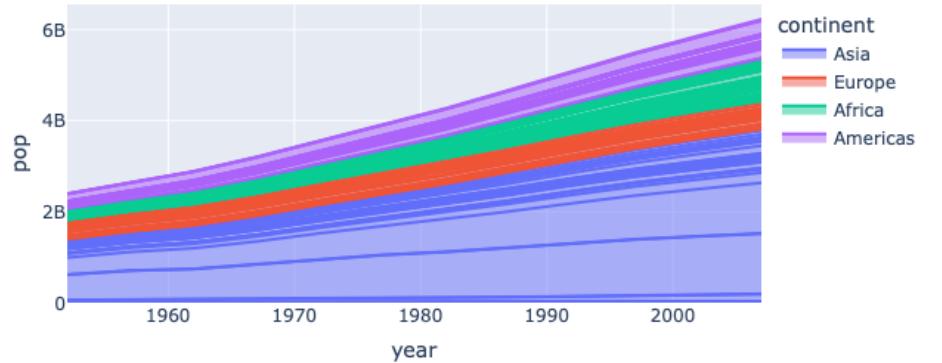
Retrieved from the [GeoNet API](#) at 2023/11/01 04:50 NZDT



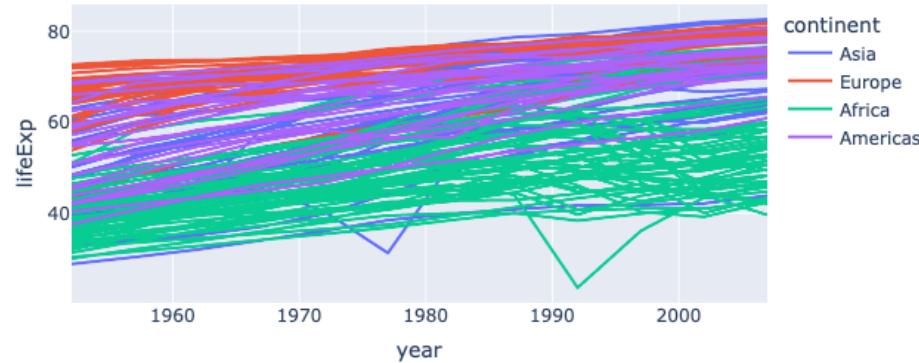
## GDP and Life Expectancy



## Population



## Life Expectancy



**7.79%**

Average annual rate  
for a 30-year fixed  
mortgage in Oct. 2023

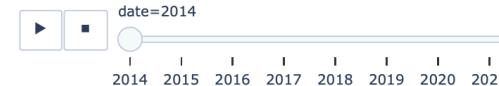
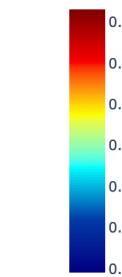
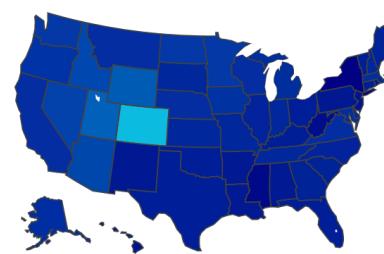
**7.03%**

Average annual rate  
for a 15-year fixed  
mortgage in Oct. 2023

**\$388k**

National median home  
price

Mortgage Originations per Capita



Mortgage Originations

myd	fixed_30	fixed_15	date
Apr-19-05	4.08	3.56	2019-04-05
Apr-19-12	4.12	3.6	2019-04-12
Apr-19-19	4.17	3.62	2019-04-19
Apr-19-26	4.2	3.64	2019-04-26
May-19-03	4.14	3.6	2019-05-03
May-19-10	4.1	3.57	2019-05-10
May-19-17	4.07	3.53	2019-05-17
May-19-24	4.06	3.51	2019-05-24
May-19-31	3.99	3.46	2019-05-31
Jun-19-07	3.82	3.28	2019-06-07
Jun-19-14	3.82	3.26	2019-06-14
Jun-19-21	3.84	3.25	2019-06-21
Jun-19-28	3.73	3.16	2019-06-28

Housing Economics

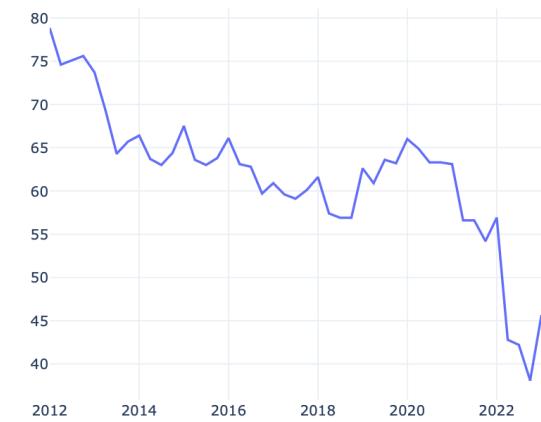
Median Price

Interest Rates 15- and 30-Year



Housing Opportunity Index

Percentage of homes affordable to median-income families



This dashboard displays statistics for:  
Hospital Grey Sloan Memorial  
Unit Labor and Delivery  
Month October 2023

In October 2023 the staff breakdown in the unit was as follows:

Attending physicians	14
Residents	21
Nurses	12

[Disclaimer](#) >

Total births



161

Cesarean deliveries



33.5%

Pre-term births



18.6%

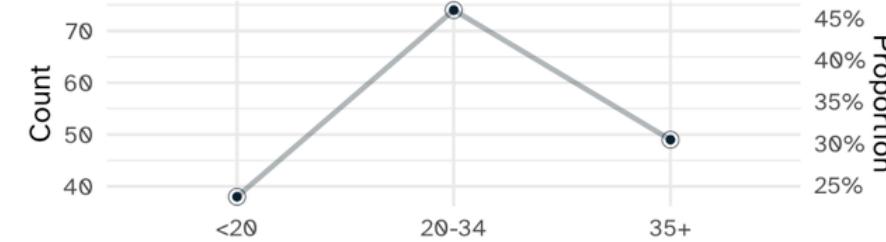
Delivery method

	Number of deliveries	Proportion of deliveries
--	----------------------	--------------------------

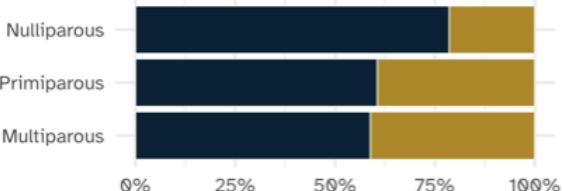
Cesarean	54	33.5%
----------	----	-------

Vaginal	107	66.5%
---------	-----	-------

Maternal age

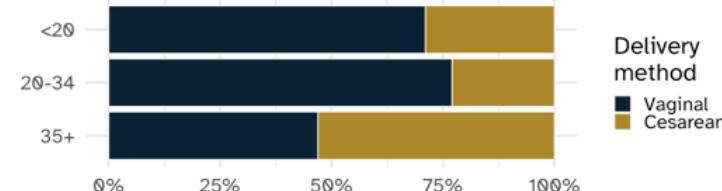


Delivery method and parity



Delivery method  
■ Vaginal  
■ Cesarean

Delivery method and maternal age



Delivery method  
■ Vaginal  
■ Cesarean



A scikit-learn Pipeline model

# chicago-model-python

Model age

# 54 days old

## Model details

- The city of Chicago offers access to health code inspections of restaurants, available from the [Chicago Department of Public Health](#). This model looks to predict inspection outcome from `['facility_type', 'risk', 'total_violations', 'month', 'year']` features.
- The model deployed is a [scikit-learn](#) Pipeline involving an encoder for categorical variables and a RandomForestClassifier to make predictions.

## Intended use

- The primary intended users of this model are people who are interested in health inspection data from Chicago
- Some use cases are out of scope for this model, such as using this model for real world health inspection prediction

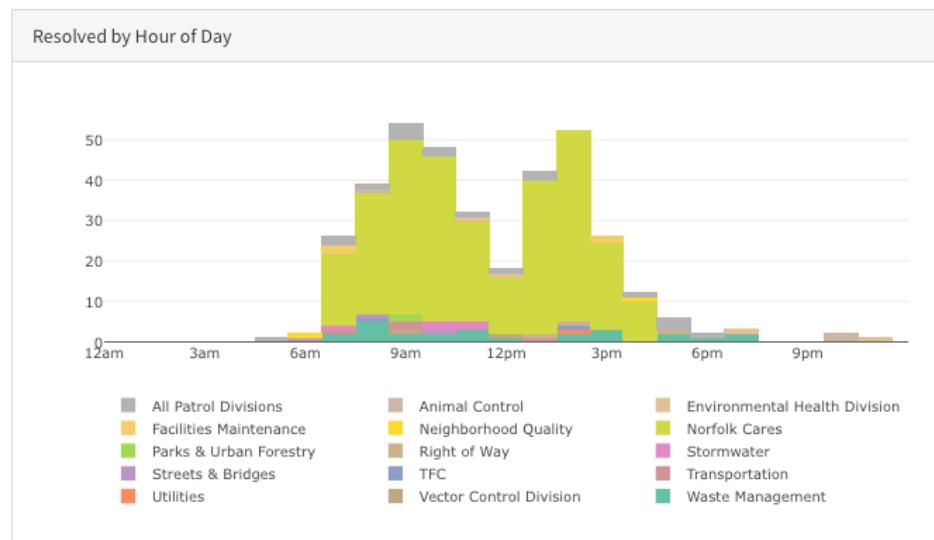
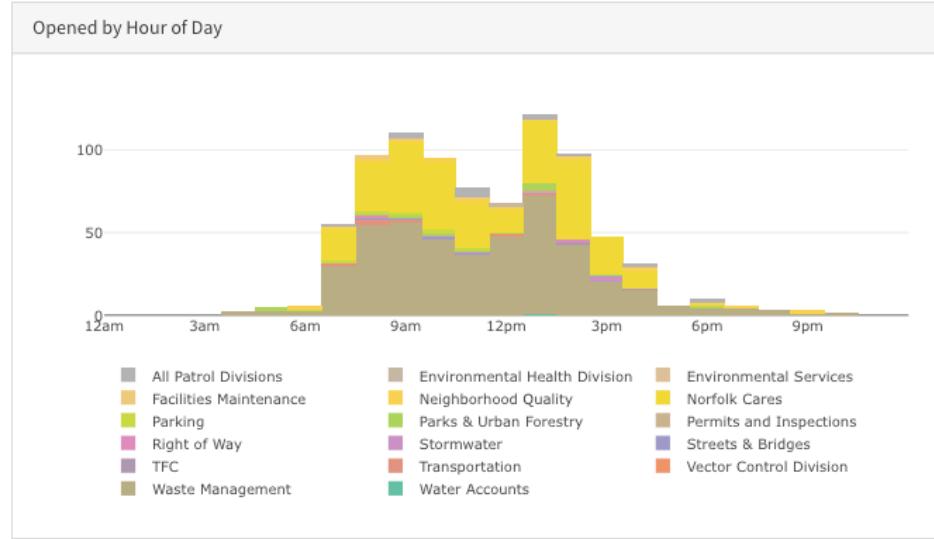
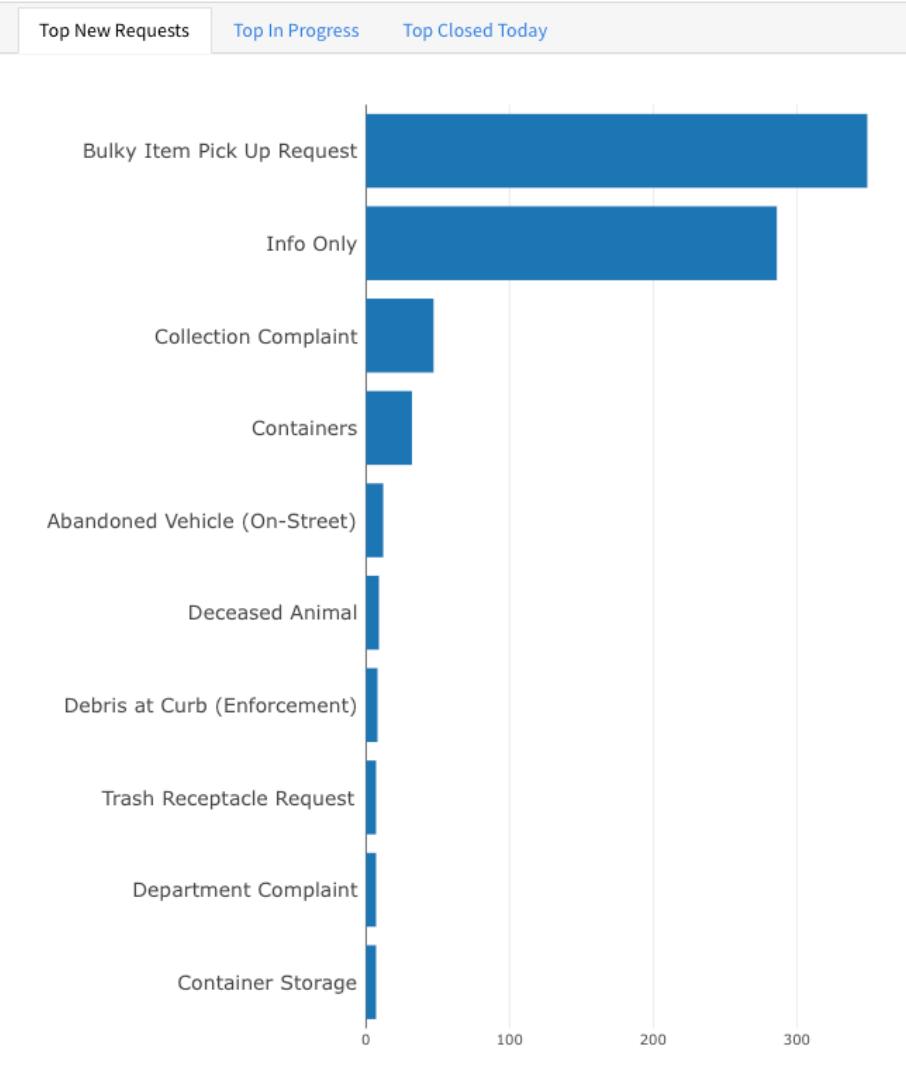
## Training data & evaluation data

- The training dataset for this model has the prototype:

```
{'facility_type': {'default': 'RESTAURANT',
  'title': 'Facility Type',
  'type': 'string'},
'risk': {'default': 'RISK 1 (HIGH)', 'title': 'Risk', 'type': 'string'},
'totalViolations': {'default': 31.0,
  'title': 'Total Violations',
  'type': 'number'},
'month': {'default': 11, 'title': 'Month', 'type': 'integer'},
'year': {'default': 2019, 'title': 'Year', 'type': 'integer'}}
```

## Ethical considerations

- This model does not have personal data and is not used for production. However, health inspections have other inputs that affect failure that are not tracked in this model that should be taken into account for production use cases.





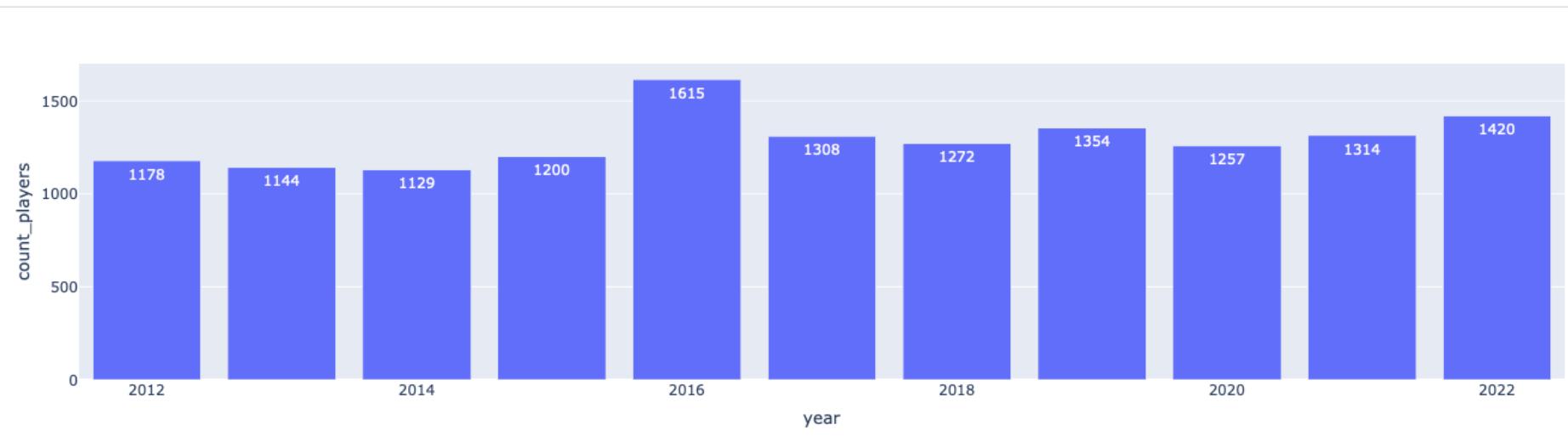
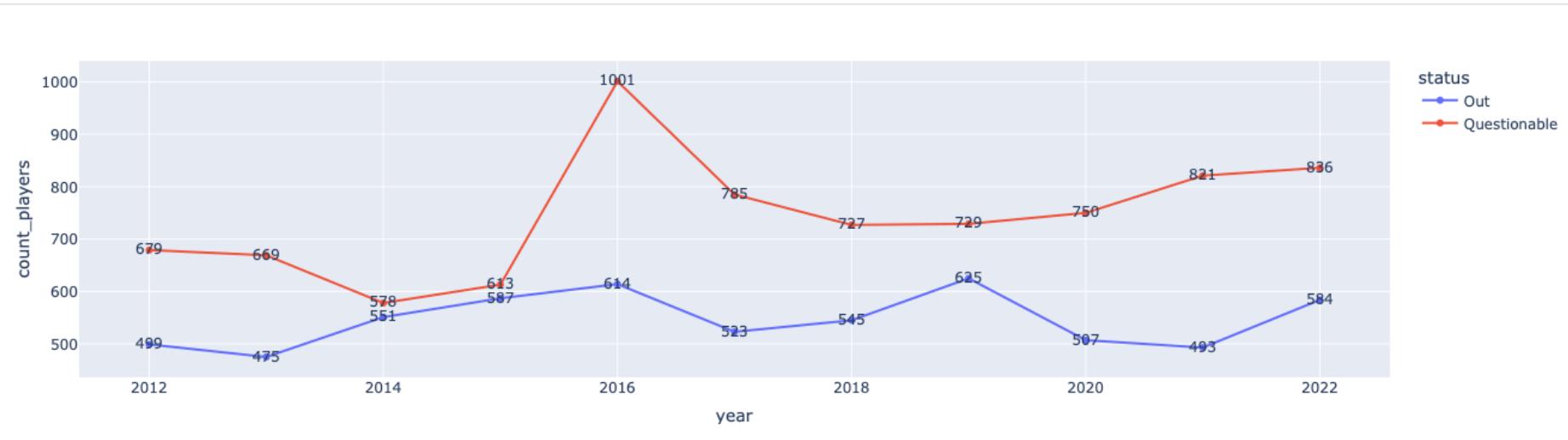
Average Injured

**1289**

Average Out

**545**

Average Questionable

**744**



## Scenario A

## Initial investment



## Average annual investment return



## Average annual inflation



## Monthly withdrawals



## Scenario B

## Initial investment



## Average annual investment return



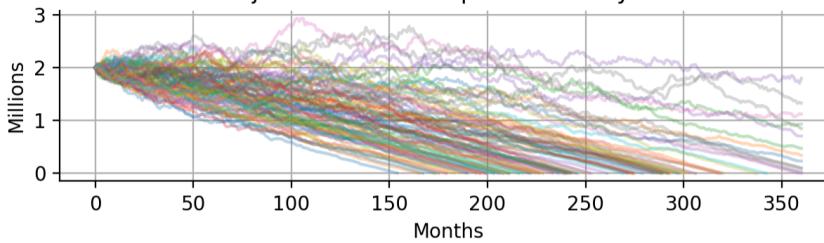
## Average annual inflation



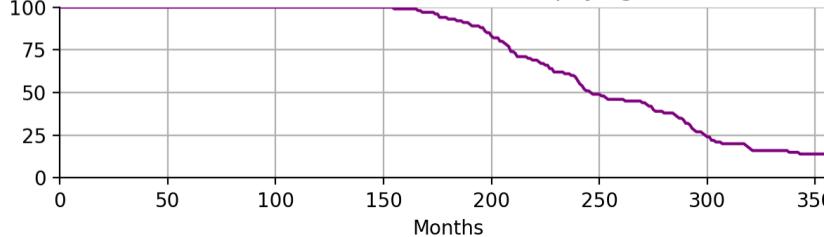
## Monthly withdrawals



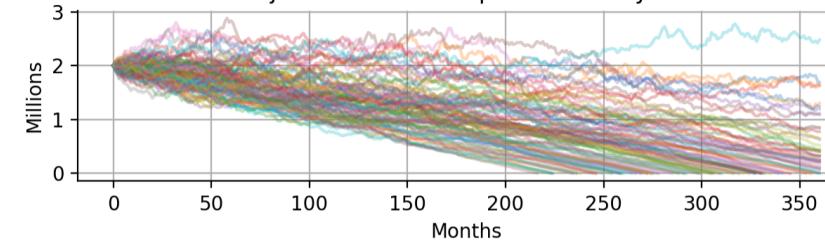
Projected value of capital over 30 years



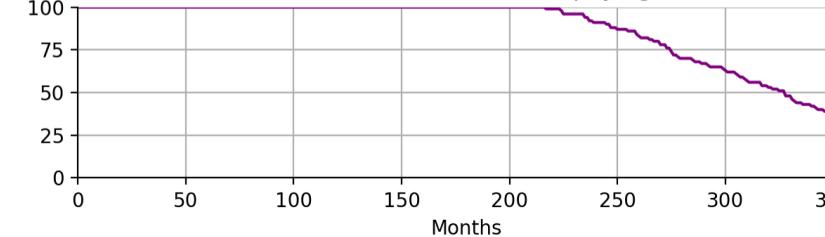
Percent of scenarios still paying



Projected value of capital over 30 years



Percent of scenarios still paying





←

## Species:

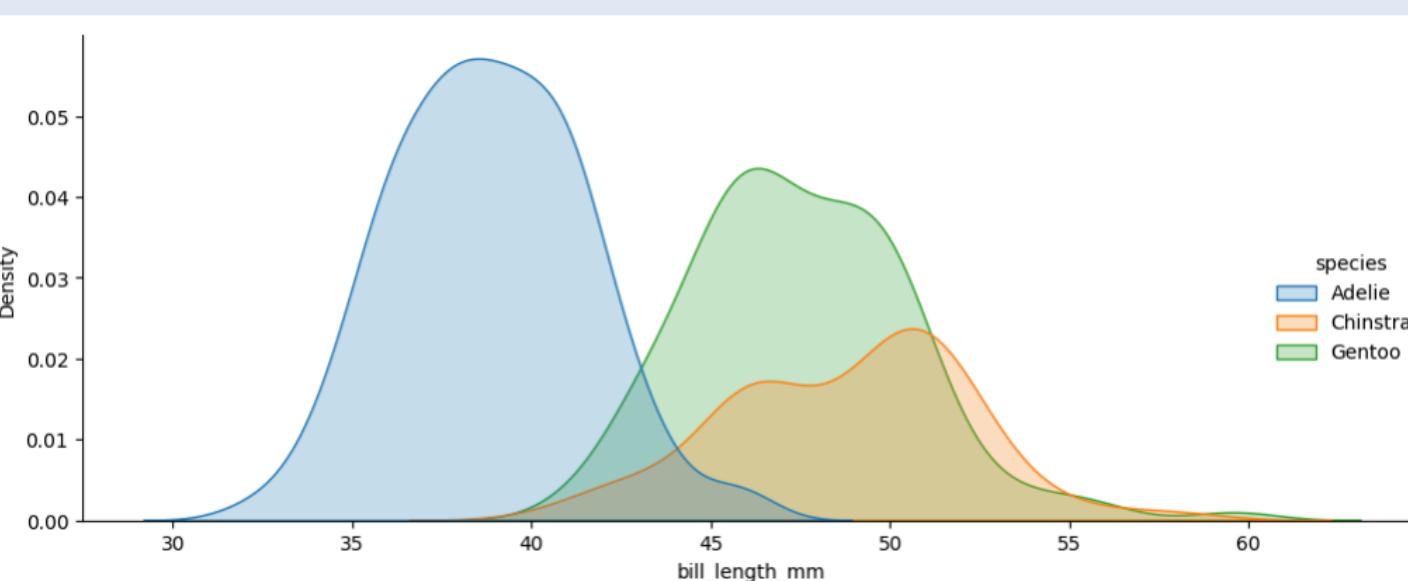
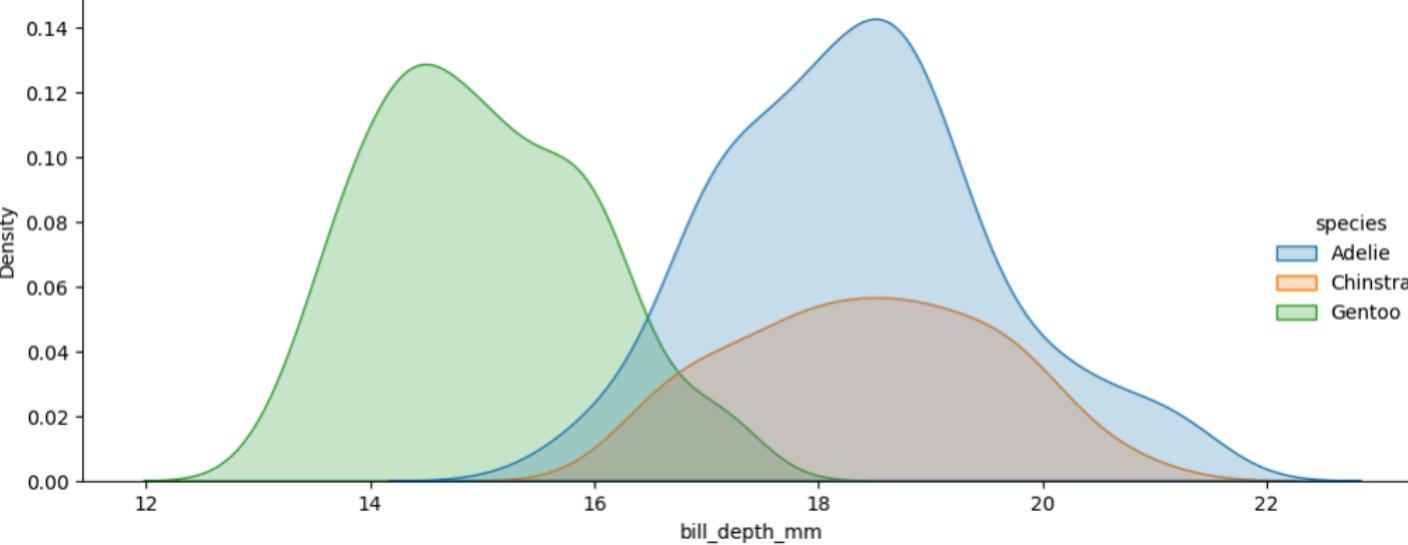
- Adelie
- Gentoo
- Chinstrap

## Islands:

- Biscoe
- Dream
- Torgersen

## Distribution:

kde

 Show rug marks[Learn more about the Palmer Penguins dataset.](#)



BA — The Boeing Company — <https://www.boeing.com>

\$ 179.69  
Current Price

↑ \$0.60  
Change

% 0.34  
Percent Change

#### Price History

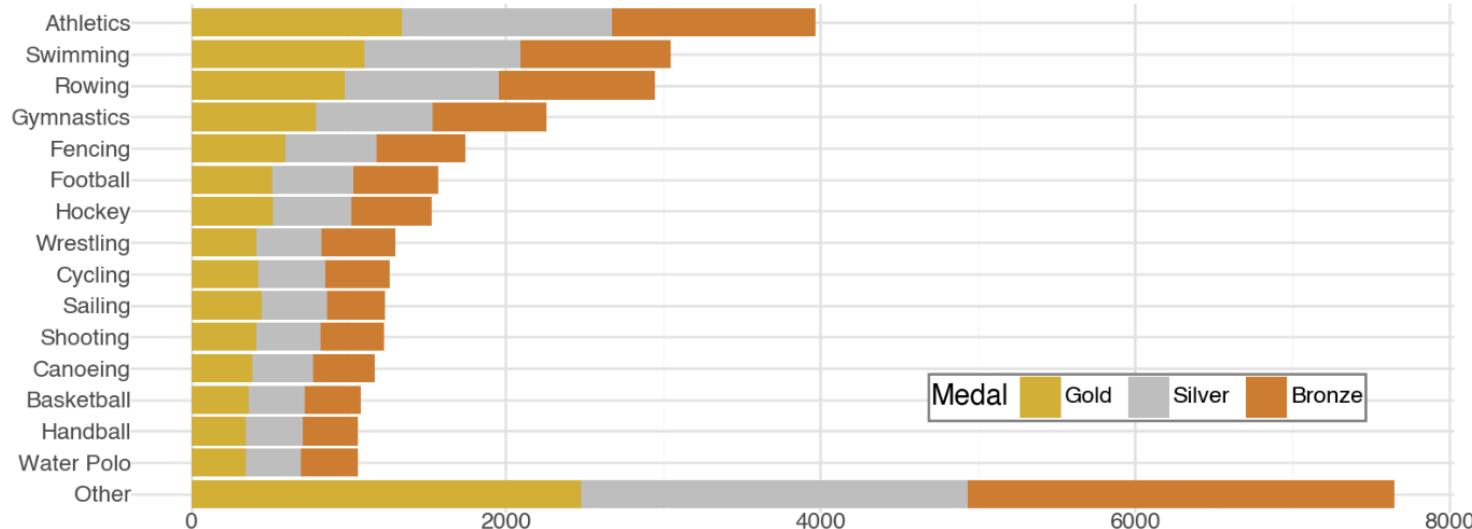


Last Close: 2023-10-27

Open	\$180.00
High	\$182.33
Low	\$179.01
Close	\$179.69
Volume	4,601,300



## Medals by sport



Medal    Gold    Silver    Bronze

0    2000    4000    6000    8000



Top 30 total medals    Bottom 30 total medals

Teams sorted in descending order of total medals.

Team	Gold	Silver	Bronze
United States	2363.0	1251.0	1126.0
Soviet Union	830.0	635.0	596.0
Germany	568.0	515.0	614.0
Great Britain	501.0	594.0	563.0
France	425.0	490.0	515.0
Italy	492.0	469.0	425.0
Australia	338.0	450.0	506.0
Hungary	432.0	328.0	363.0
Sweden	326.0	356.0	332.0
China	334.0	317.0	258.0
Russia	296.0	278.0	324.0
Netherlands	235.0	283.0	350.0
Japan	230.0	287.0	333.0
East Germany	339.0	277.0	227.0
Canada	133.0	224.0	313.0
Romania	161.0	200.0	290.0
Denmark	168.0	218.0	166.0
South Korea	171.0	206.0	175.0

## Medals by year

Due to World War II, no olympic games were held in 1940 and 1944.



Medal    Gold    Silver    Bronze

# How to build a Quarto Dashboard ?

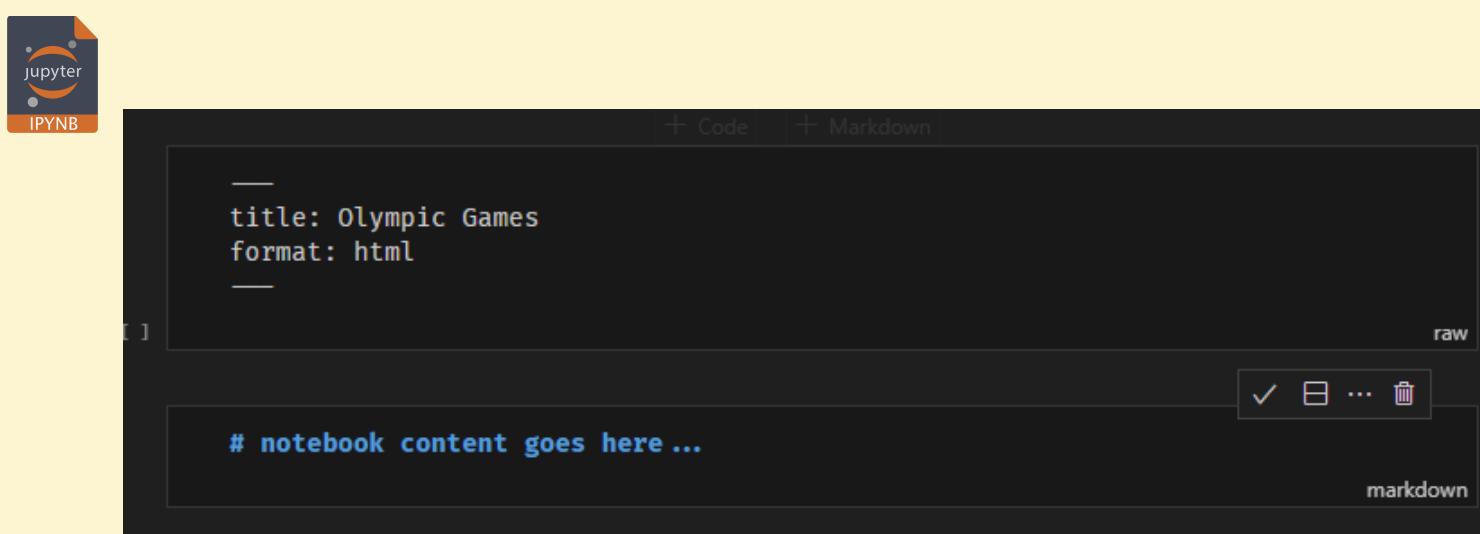
Follow along at <https://cderv.github.io/pydata-paris-2024-quarto-dashboard/> and <https://quarto.org>



# Notebook → HTML

```
olympicdash-py.qmd
```

```
1 ---  
2 title: "Olympic Games"  
3 format: html  
4 ---  
5  
6 # notebook content goes here...
```



The screenshot shows a Jupyter Notebook interface. On the left, there's a sidebar with the 'jupyter' logo and 'IPYNB' text. The main area has two cells. The top cell contains the following code:

```
title: Olympic Games  
format: html
```

The bottom cell contains the text:

```
# notebook content goes here ...
```

Below the cells are standard Jupyter controls: a 'raw' button, a toolbar with a checkmark, a bold icon, a ellipsis icon, and a delete icon, and a 'markdown' button.

# Notebook → HTML

olympicdash-py.qmd

```
1 ---  
2 title: "Olympic Games"  
3 format: html  
4 ---  
5  
6 # notebook content goes here...
```

The screenshot shows a Jupyter Notebook interface. On the left, there's a sidebar with the Jupyter logo and the text "IPYNB". The main area has two cells. The top cell is a code cell containing the following YAML configuration:

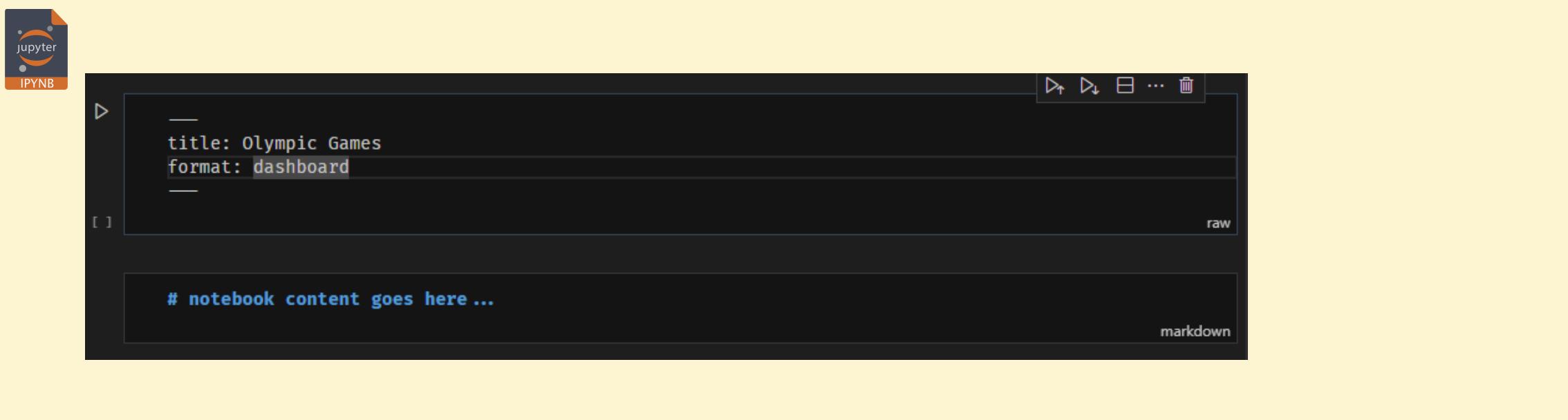
```
title: Olympic Games  
format: html
```

The bottom cell is a markdown cell containing the placeholder text "# notebook content goes here ...". There are standard Jupyter notebook controls like a checkmark, a refresh icon, an ellipsis, and a delete icon above the markdown cell, along with "raw" and "markdown" buttons.

# Notebook → Dashboard

```
olympicdash-py.qmd
```

```
1 ---  
2 title: "Olympic Games"  
3 format: dashboard  
4 ---  
5  
6 # notebook content goes here...
```



The screenshot shows a Jupyter Notebook interface with a yellow header bar. On the left, there's a dark sidebar with a 'jupyter' logo and an 'IPYNB' button. The main area displays a code cell containing the following content:

```
title: Olympic Games  
format: dashboard
```

Below this cell is a text cell containing the instruction:

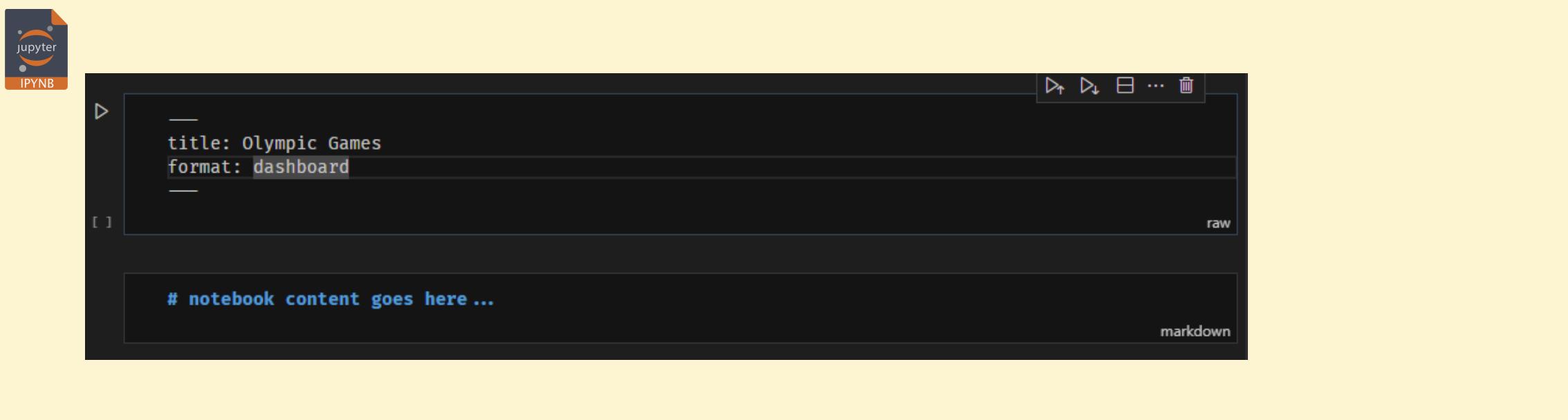
```
# notebook content goes here ...
```

The interface includes standard Jupyter Notebook controls at the top right, such as arrows for navigation, a search icon, and a trash bin icon. Below the code cell, there are 'raw' and 'markdown' buttons. The overall layout is clean and modern, typical of a Jupyter Notebook environment.

# Notebook → Dashboard

```
olympicdash-py.qmd
```

```
1 ---  
2 title: "Olympic Games"  
3 format: dashboard  
4 ---  
5  
6 # notebook content goes here...
```



The screenshot shows a Jupyter Notebook interface with a yellow header bar. On the left, there's a dark sidebar with a 'jupyter' logo and an 'IPYNB' button. The main area displays a code cell containing the following content:

```
title: Olympic Games  
format: dashboard
```

Below this cell is a text cell containing the instruction:

```
# notebook content goes here ...
```

The interface includes standard Jupyter Notebook controls at the top right, such as arrows for navigation, a search icon, and a trash bin icon. Below the code cell, there are 'raw' and 'markdown' buttons. The overall layout is clean and modern, typical of a Jupyter Notebook environment.

# Layout

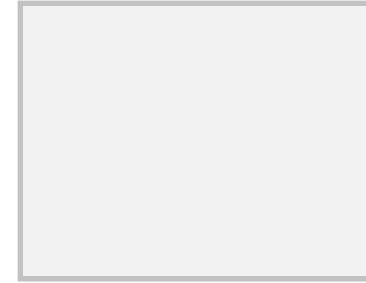


Follow along at <https://cderv.github.io/pydata-paris-2024-quarto-dashboard/> and <https://quarto.org>



# Layout - Cards

Dashboards are composed of **cards**.



# Layout - Rows and columns

Cards are arranged into **rows** and **columns**.



# Layout - Advanced

**Sidebars, tabssets, and pages** allow for more advanced layouts.



# Dashboard Components

## 1. Navigation Bar and Pages

- Icon, title, and author along with links to sub-pages (if more than one page is defined).

## 2. Sidebars, Rows & Columns, and Tabssets

- Rows and columns using markdown heading (with optional attributes to control height, width, etc.). Sidebars for interactive inputs. Tabssets to further divide content.

## 3. Cards (Plots, Tables, Value Boxes, Content)

- Cards are containers for cell outputs and free form markdown text. The content of cards typically maps to *cells* in your notebook or source document.



# Dashboard Components

## 1. Navigation Bar and Pages

- Icon, title, and author along with links to sub-pages (if more than one page is defined).

## 2. Sidebars, Rows & Columns, and Tabssets

- Rows and columns using markdown heading (with optional attributes to control height, width, etc.). Sidebars for interactive inputs. Tabssets to further divide content.

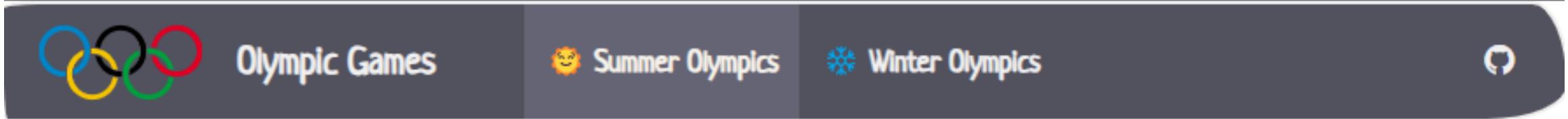
## 3. Cards (Plots, Tables, Value Boxes, Content)

- Cards are containers for cell outputs and free form markdown text. The content of cards typically maps to *cells* in your notebook or source document.

*All of these components can be authored and customized within notebook UI or plain text qmd.*



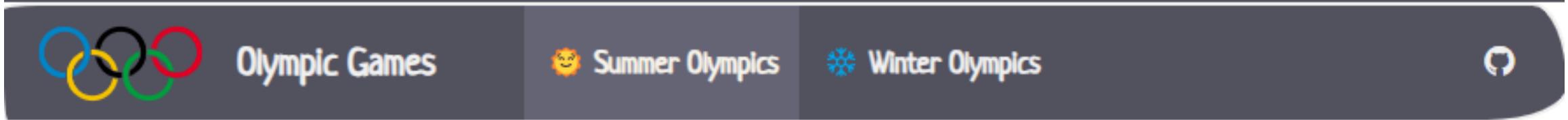
# Navigation Bar and Pages



```
1 ---  
2 title: "Olympic Games"  
3 format:  
4   dashboard:  
5     nav-buttons: [github]  
6     github: https://github.com/posit-conf-2024/olympicdash  
7 logo: images/olympics-logo.svg  
8 logo-alt: "Olympics logo with multicolored circles."  
9 ---  
10  
11 # ☀ Summer Olympics  
12  
13 # ❄ Winter Olympics
```



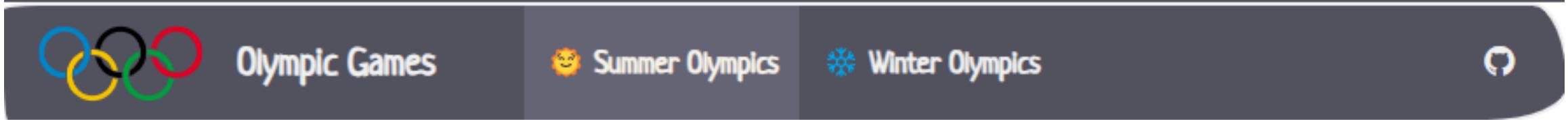
# Navigation Bar and Pages



```
1 ---  
2 title: "Olympic Games"  
3 format:  
4   dashboard:  
5     nav-buttons: [github]  
6     github: https://github.com/posit-conf-2024/olympicdash  
7 logo: images/olympics-logo.svg  
8 logo-alt: "Olympics logo with multicolored circles."  
9 ---  
10  
11 # ☀ Summer Olympics  
12  
13 # ❄ Winter Olympics
```



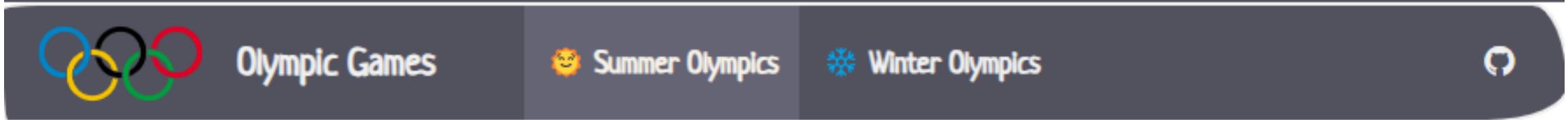
# Navigation Bar and Pages



```
1 ---  
2 title: "Olympic Games"  
3 format:  
4   dashboard:  
5     nav-buttons: [github]  
6     github: https://github.com/posit-conf-2024/olympicdash  
7   logo: images/olympics-logo.svg  
8   logo-alt: "Olympics logo with multicolored circles."  
9 ---  
10  
11 # ☀ Summer Olympics  
12  
13 # ❄ Winter Olympics
```



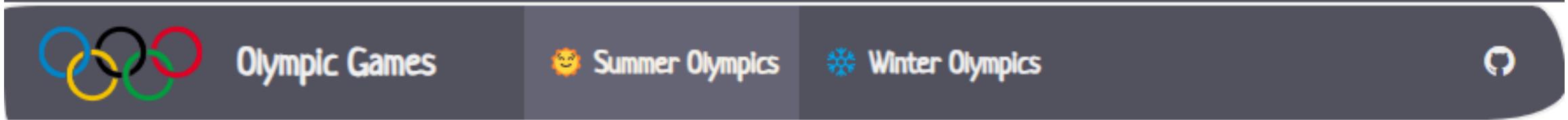
# Navigation Bar and Pages



```
1 ---  
2 title: "Olympic Games"  
3 format:  
4   dashboard:  
5     nav-buttons: [github]  
6     github: https://github.com/posit-conf-2024/olympicdash  
7 logo: images/olympics-logo.svg  
8 logo-alt: "Olympics logo with multicolored circles."  
9 ---  
10  
11 # ☀ Summer Olympics  
12  
13 # ❄ Winter Olympics
```



# Navigation Bar and Pages



```
1 ---  
2 title: "Olympic Games"  
3 format:  
4   dashboard:  
5     nav-buttons: [github]  
6     github: https://github.com/posit-conf-2024/olympicdash  
7 logo: images/olympics-logo.svg  
8 logo-alt: "Olympics logo with multicolored circles."  
9 ---  
10  
11 # ☀ Summer Olympics  
12  
13 # ❄ Winter Olympics
```



Follow along at <https://cderv.github.io/pydata-paris-2024-quarto-dashboard/> and <https://quarto.org>



# Sidebars: Page Level

```
1 ---  
2 title: "Sidebar"  
3 format: dashboard  
4 ---  
5  
6 # Page 1  
7  
8 ## {.sidebar}  
9  
10 ````{python}  
11 ````  
12  
13 ## Column  
14  
15 ````{python}
```



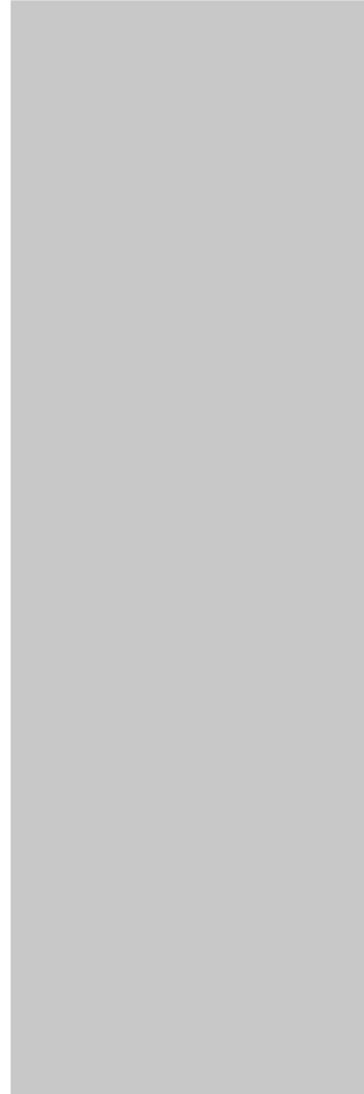
# Sidebars: Page Level

```
3 format: dashboard
4 ---
5
6 # Page 1
7
8 ## {.sidebar}
9
10 ````{python}
11 ```
12
13 ## Column
14
15 ````{python}
16 ```
17 ````
```



# Sidebars: Page Level

```
3 format: dashboard  
4 ---  
5  
6 # Page 1  
7  
8 ## {.sidebar}  
9  
10 ````{python}  
11 ``  
12  
13 ## Column  
14  
15 ````{python}  
16 ``  
17 ``
```



**Chart 1**

**Chart 2**



# Sidebars: Global

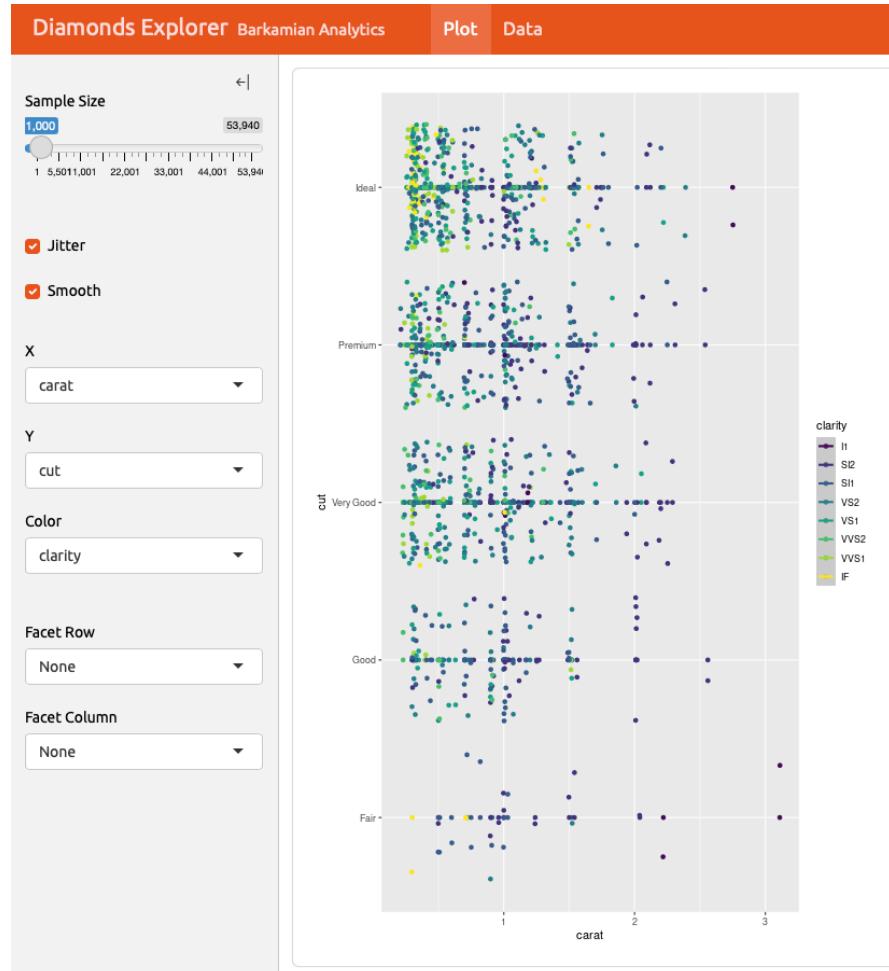
```
1 ---  
2 title: "Global Sidebar"  
3 format: dashboard  
4 ---  
5  
6 # {.sidebar}  
7  
8 Sidebar content (e.g. inputs)  
9  
10 # Plot  
11  
12 ```{python}  
13 ...  
14  
15 # Data
```

# Sidebars: Global

```
1 ---  
2 title: "Global Sidebar"  
3 format: dashboard  
4 ---  
5  
6 # {.sidebar}  
7  
8 Sidebar content (e.g. inputs)  
9  
10 # Plot  
11  
12 ```{python}  
13 ...  
14  
15 # Data
```

# Sidebars: Global

```
1 ---  
2 title: "Global Sidebar"  
3 format: dashboard  
4 ---  
5  
6 # {.sidebar}  
7  
8 Sidebar content (e.g. inputs)  
9  
10 # Plot  
11  
12 ````{python}  
13 ````  
14  
15 # Data
```



# Layout: Rows

```
1 ---  
2 title: "By Rows"  
3 format: dashboard  
4 ---  
5  
6 ## Row {height=70%}  
7  
8 ````{python}  
9 ``  
10  
11 ## Row {height=30%}  
12  
13 ````{python}  
14 ``  
15
```

 **Row** for header content is used as example here.  
Header name can be anything.



# Layout: Rows

```
1
2 title: "By Rows"
3 format: dashboard
4 ---
5
6 ## Row {height=70%}
7
8 ````{python}
9 ``
10
11 ## Row {height=30%}
12
13 ````{python}
14 ``
15
16 ````{python}
```

 **Row** for header content is used as example here.  
Header name can be anything.

# Layout: Rows

```
1
2 title: "By Rows"
3 format: dashboard
4 ---
5
6 ## Row {height=70%}
7
8 ````{python}
9 ``
10
11 ## Row {height=30%}
12
13 ````{python}
14 ``
15
16 ````{python}
```

 **Row** for header content is used as example here.  
Header name can be anything.

**Chart 1**

**Chart 2**

**Chart 3**



Churn rate 2022

17%



Current churn rate

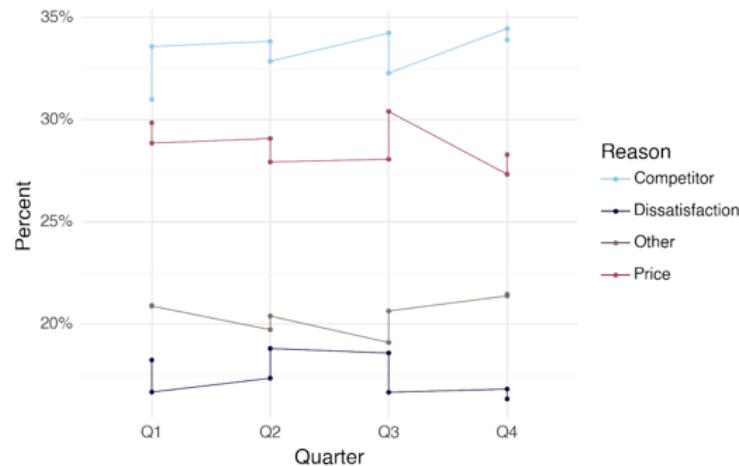
14%



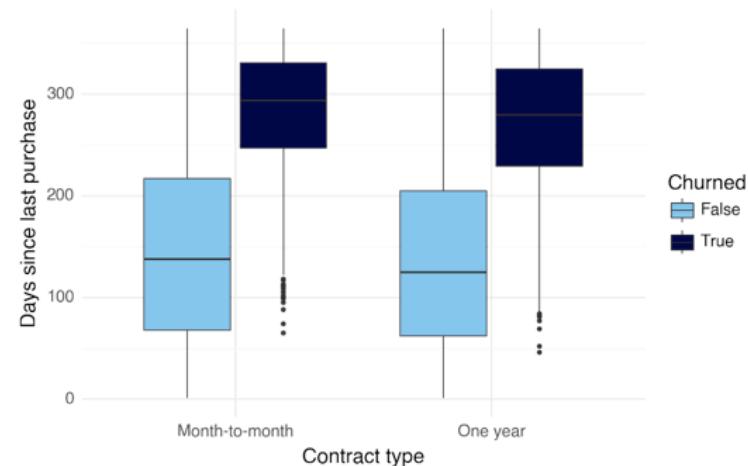
Churn rate goal

10%

Reason for churn by quarter



Churn by contract type and days since purchase



Purchase characteristics

index	Contract type	Churn	Average purchase	Total transactions	Days since last purchase
2	Month-to-month	Did not churn	50.14	19	138
5	Month-to-month	Churned	50.01	16	294
8	One year	Did not churn	50.05	2	125
11	One year	Churned	50.15	2	280

# Layout: Columns

```
1 ---  
2 title: "By Columns"  
3 format:  
4   dashboard:  
5     orientation: columns  
6 ---  
7  
8 ## Column {width=60%}  
9  
10 ````{python}  
11 ````  
12  
13 ## Column {width=40%}  
14  
15 ````{python}
```

# Layout: Columns

```
1 ---  
2 title: "By Columns"  
3 format:  
4   dashboard:  
5     orientation: columns  
6 ---  
7  
8 ## Column {width=60%}  
9  
10 ````{python}  
11 ````  
12  
13 ## Column {width=40%}  
14  
15 ````{python}
```

# Layout: Columns

```
3 format.
4   dashboard:
5     orientation: columns
6 ---
7
8 ## Column {width=60%}
9
10 ````{python}
11 ```
12
13 ## Column {width=40%}
14
15 ````{python}
16 ```
17
18 ````{python}
```

# Layout: Columns

```
5     orientation: columns
6     ---
7
8 ## Column {width=60%}
9
10 ````{python}
11 ``
12
13 ## Column {width=40%}
14
15 ````{python}
16 ``
17
18 ````{python}
19 ````
```

# Layout: Columns

```
5     orientation: columns
6     ---
7
8 ## Column {width=60%}
9
10 ````{python}
11 ```
12
13 ## Column {width=40%}
14
15 ````{python}
16 ```
17
18 ````{python}
19 ````
```

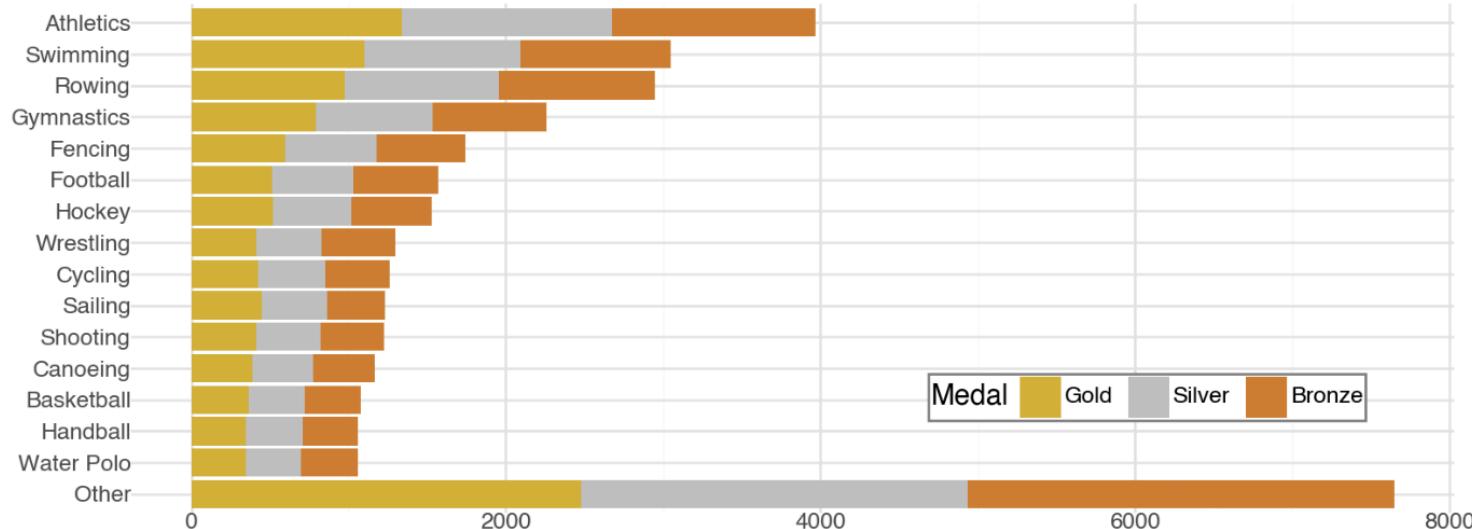
Chart 1

Chart 2

Chart 3



## Medals by sport



Medal    Gold    Silver    Bronze

0    2000    4000    6000    8000



Top 30 total medals    Bottom 30 total medals

Teams sorted in descending order of total medals.

Team	Gold	Silver	Bronze
United States	2363.0	1251.0	1126.0
Soviet Union	830.0	635.0	596.0
Germany	568.0	515.0	614.0
Great Britain	501.0	594.0	563.0
France	425.0	490.0	515.0
Italy	492.0	469.0	425.0
Australia	338.0	450.0	506.0
Hungary	432.0	328.0	363.0
Sweden	326.0	356.0	332.0
China	334.0	317.0	258.0
Russia	296.0	278.0	324.0
Netherlands	235.0	283.0	350.0
Japan	230.0	287.0	333.0
East Germany	339.0	277.0	227.0
Canada	133.0	224.0	313.0
Romania	161.0	200.0	290.0
Denmark	168.0	218.0	166.0
South Korea	171.0	206.0	175.0

## Medals by year

Due to World War II, no olympic games were held in 1940 and 1944.



Medal    Gold    Silver    Bronze

# Cards

Automatically created from cell's output or Markdown content

```
1 ## Column - Medals by sport and year {width=65%}
2
3 ### Row - Medals by sport {height=60%}
4
5 ````{python}
6 #| label: summer-medals-by-sport
7 #| title: Medals by sport
8
9 # Lump the sport column to top 15 categories, grouping others as Other
10 top_15_sports = summer_olympics["sport"].value_counts().nlargest(15).index
11 summer_olympics["sport"] = summer_olympics["sport"].apply(lambda x: x if x in top_15_sports
12
13 # Convert the sport column to a categorical type with order based on frequency, and reverse
14 summer_olympics["sport"] = pd.Categorical(summer_olympics["sport"], categories = summer_oly
15
```

# Cards

Automatically created from cell's output or Markdown content

```
1 ## Column - Medals by sport and year {width=65%}
2
3 ### Row - Medals by sport {height=60%}
4
5 ````{python}
6 #| label: summer-medals-by-sport
7 #| title: Medals by sport
8
9 # Lump the sport column to top 15 categories, grouping others as Other
10 top_15_sports = summer_olympics["sport"].value_counts().nlargest(15).index
11 summer_olympics["sport"] = summer_olympics["sport"].apply(lambda x: x if x in top_15_sports
12
13 # Convert the sport column to a categorical type with order based on frequency, and reverse
14 summer_olympics["sport"] = pd.Categorical(summer_olympics["sport"], categories = summer_oly
15
```



# Cards

Automatically created from cell's output or Markdown content

```
1 ## Column - Medals by sport and year {width=65%}
2
3 ### Row - Medals by sport {height=60%}
4
5 ````{python}
6 #| label: summer-medals-by-sport
7 #| title: Medals by sport
8
9 # Lump the sport column to top 15 categories, grouping others as Other
10 top_15_sports = summer_olympics["sport"].value_counts().nlargest(15).index
11 summer_olympics["sport"] = summer_olympics["sport"].apply(lambda x: x if x in top_15_sports
12
13 # Convert the sport column to a categorical type with order based on frequency, and reverse
14 summer_olympics["sport"] = pd.Categorical(summer_olympics["sport"], categories = summer_oly
15
```



# Cards

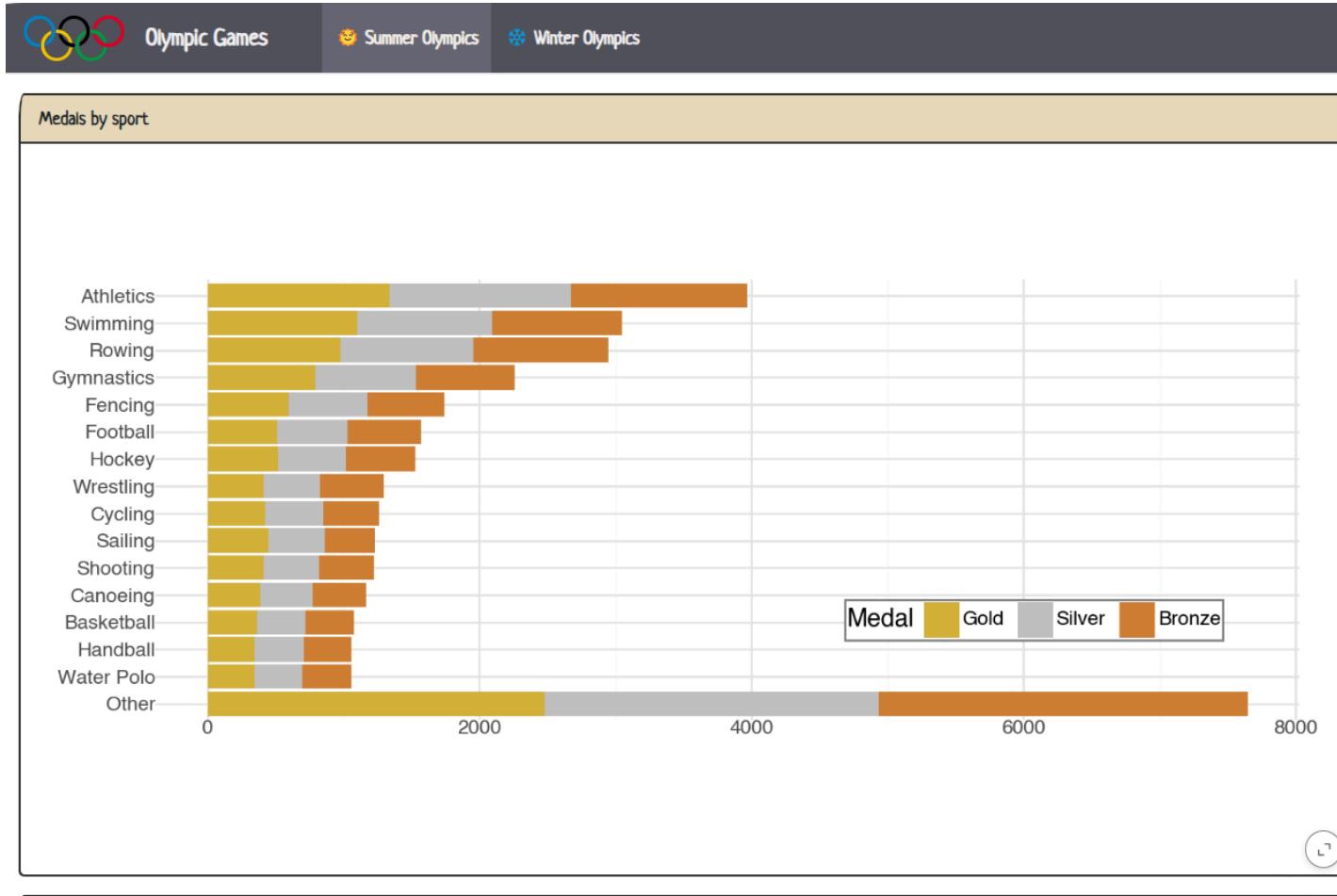
Automatically created from cell's output or Markdown content

```
20 # Plot
21 (
22   ggplot(summer_olympics, aes(x = "sport", fill = "medal")) +
23   geom_bar() +
24   coord_flip() +
25   guides(fill = guide_legend(reverse = True)) +
26   scale_fill_manual(
27     values = {"Gold": "#d4af37", "Silver": "#c0c0c0", "Bronze": "#cd7f32"})
28 ) +
29 labs(
30   x = "",
31   y = "",
32   fill = "Medal"
33 ) +
34 theme_minimal() +
```



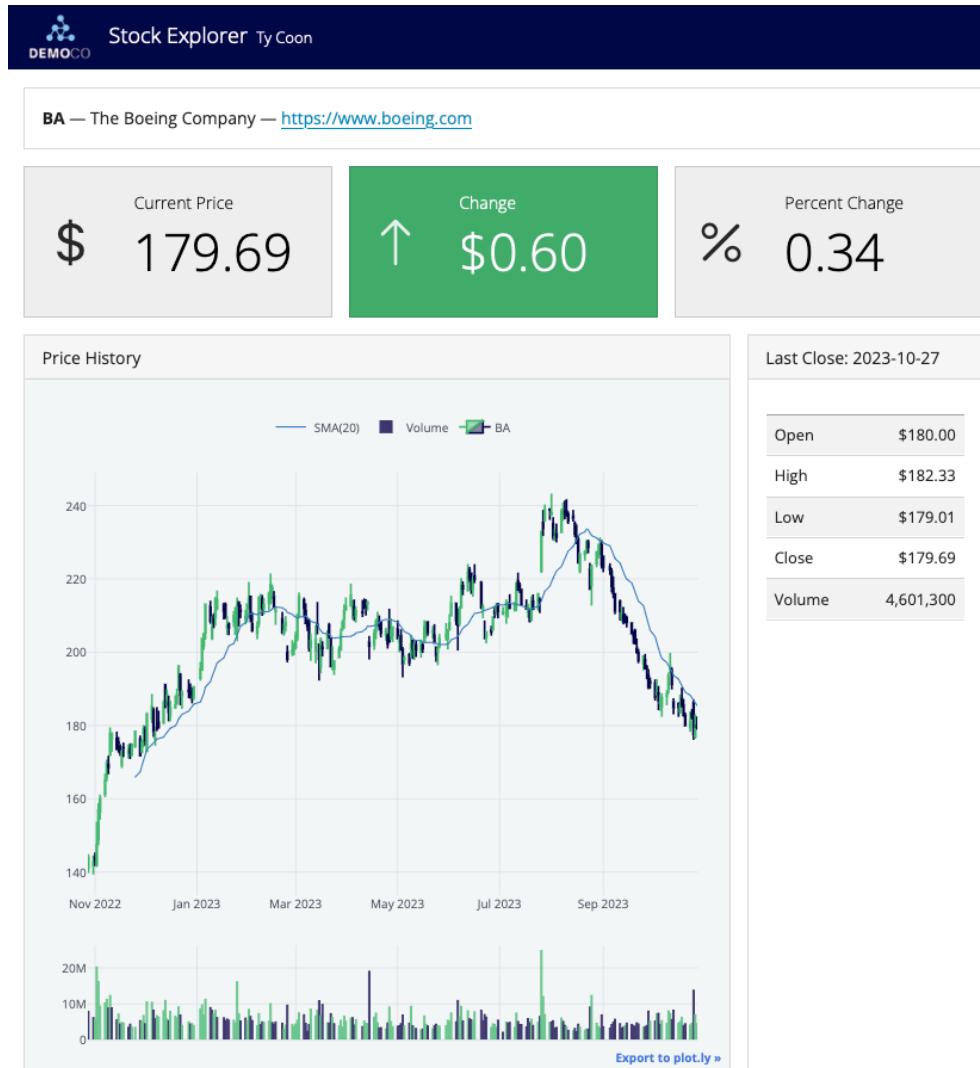
# Cards

Automatically created from cell's output or Markdown content



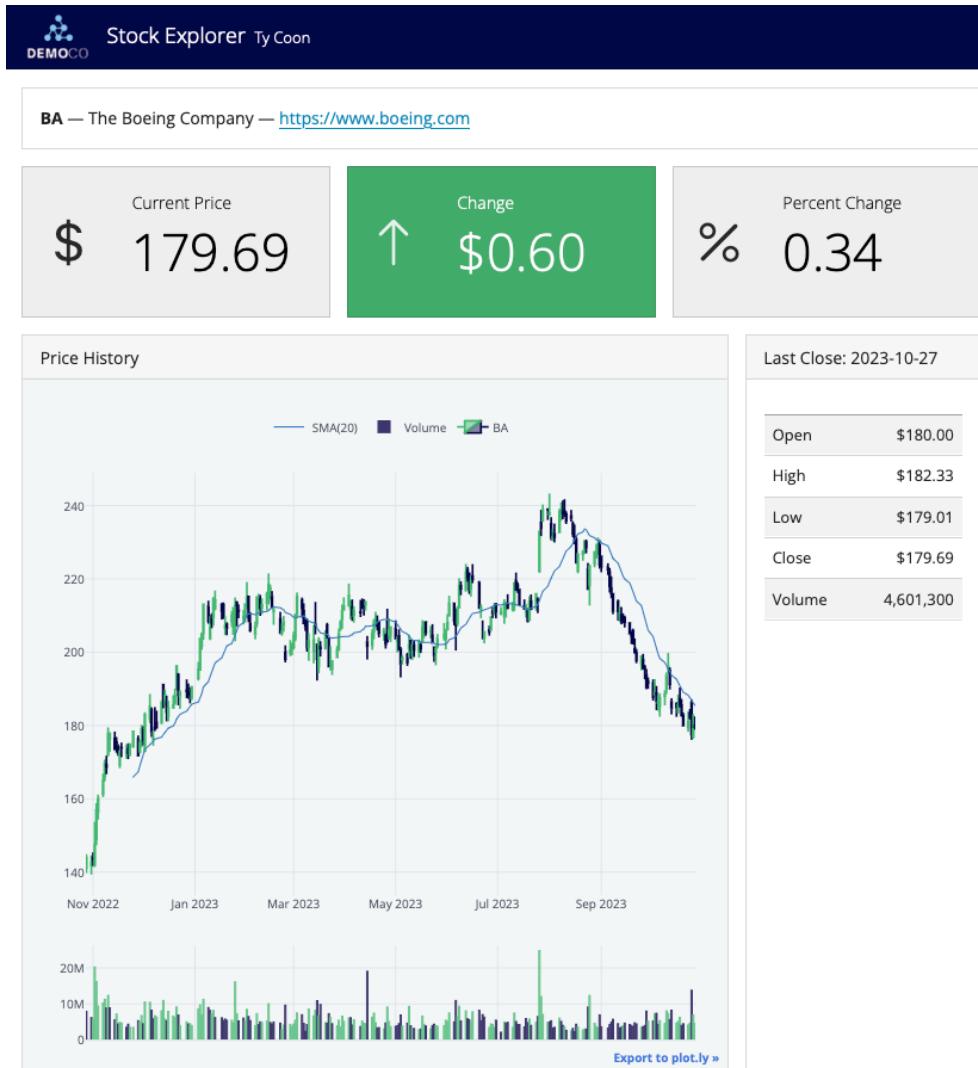
# Value Boxes

```
1 ## Row
2
3 ````{python}
4 #| component: valuebox
5 #| title: "Current Price"
6 dict(icon = "currency-dollar",
7      color = "secondary",
8      value = get_price(data))
9
10 ````{python}
11 #| component: valuebox
12 #| title: "Change"
13 change = get_change(data)
14 dict(value = change['amount'],
```



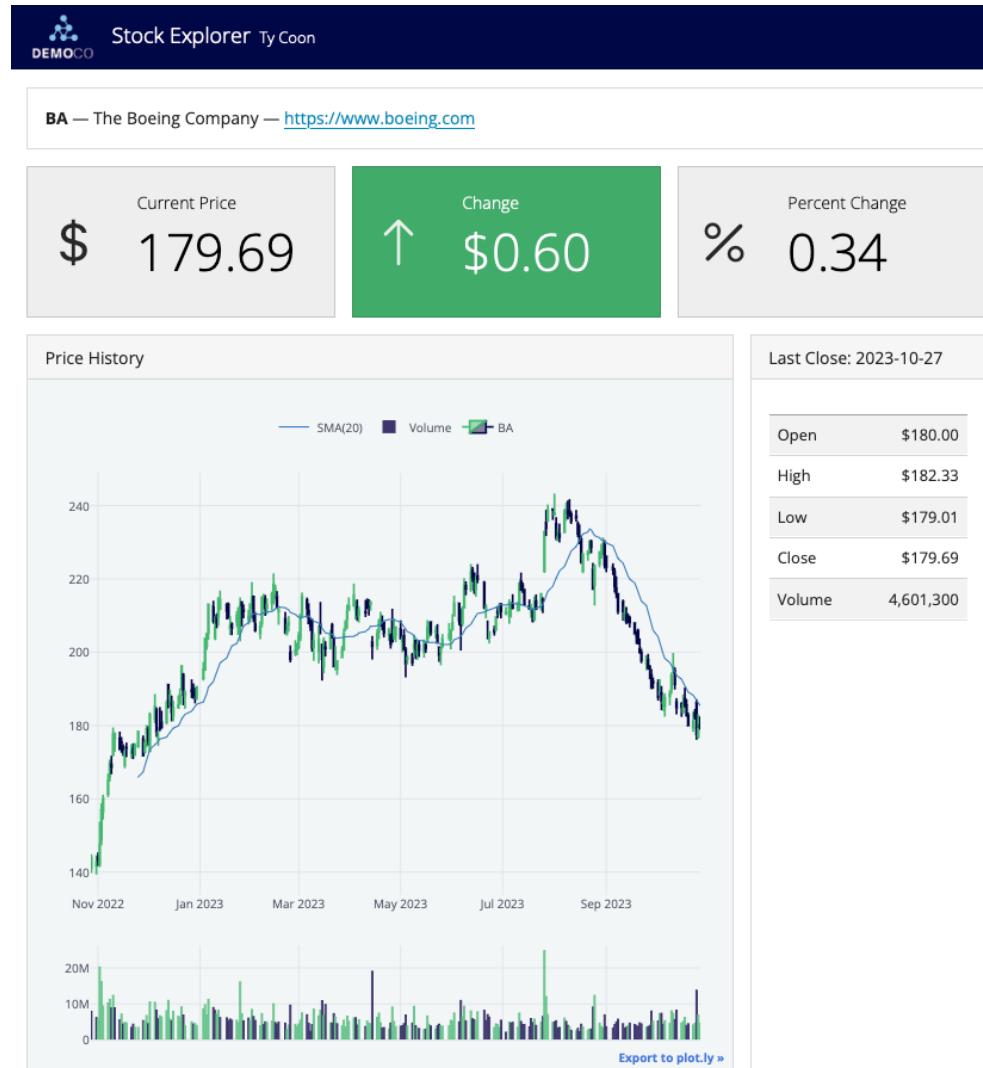
# Value Boxes

```
1 ## Row
2
3 ````{python}
4 #| component: valuebox
5 #| title: "Current Price"
6 dict(icon = "currency-dollar",
7      color = "secondary",
8      value = get_price(data))
9
10 ````{python}
11 #| component: valuebox
12 #| title: "Change"
13 change = get_change(data)
14 dict(value = change['amount'],
```



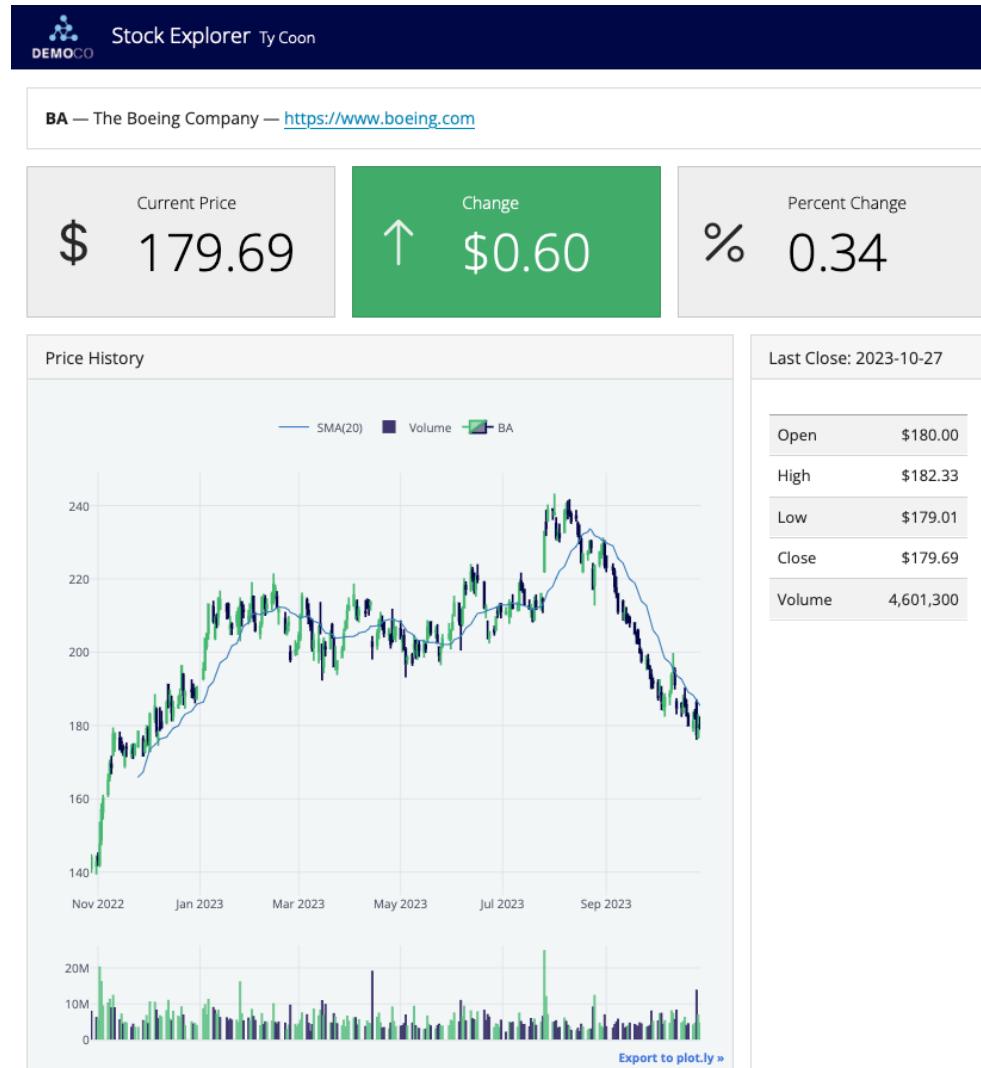
# Value Boxes

```
4 #| component: valuebox
5 #| title: "Current Price"
6 dict(icon = "currency-dollar",
7      color = "secondary",
8      value = get_price(data))
9 ``
10 ``
11 ````{python}
12 #| component: valuebox
13 #| title: "Change"
14 change = get_change(data)
15 dict(value = change['amount'],
16      icon = change['icon'],
17      color = change['color'])
18 ````
```



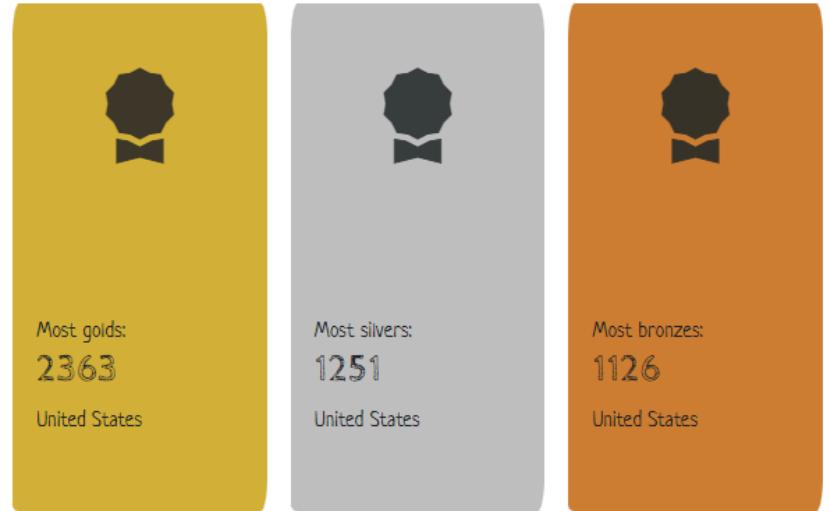
# Value Boxes

```
1 ## Row
2
3 ````{python}
4 #| component: valuebox
5 #| title: "Current Price"
6 dict(icon = "currency-dollar",
7      color = "secondary",
8      value = get_price(data))
9
10 ````{python}
11 #| component: valuebox
12 #| title: "Change"
13 change = get_change(data)
14 dict(value = change['amount'],
```



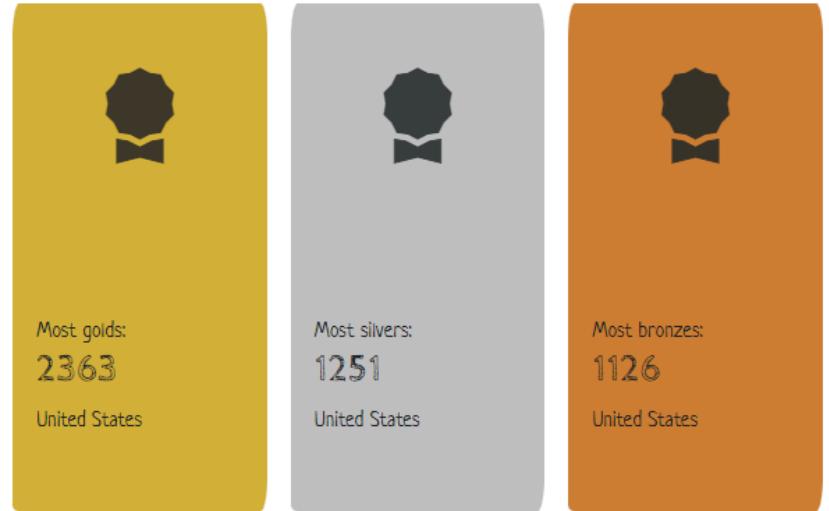
# Value Boxes

```
1 ## Column - Medals by country {width=35%}  
2  
3 ### Row - Value boxes {height=30%}  
4  
5 :::{.valuebox icon="award-fill" color="#d4af37"}  
6 Most golds:  
7  
8 `{python} str(count_most_gold_medals)`  
9  
10 `{python} most_gold_medals`  
11 :::  
12  
13 :::{.valuebox icon="award-fill" color="#c0c0c0"}  
14 Most silvers:  
15
```



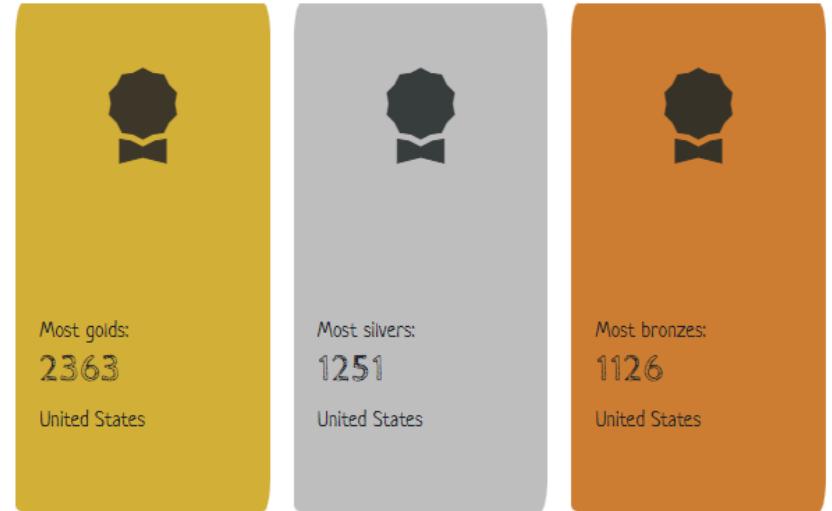
# Value Boxes

```
1 ## Column - Medals by country {width=35%}  
2  
3 ### Row - Value boxes {height=30%}  
4  
5 :::.valuebox icon="award-fill" color="#d4af37"}  
6 Most golds:  
7  
8 `{python} str(count_most_gold_medals)`  
9  
10`{python} most_gold_medals`  
11:::  
12  
13 :::.valuebox icon="award-fill" color="#c0c0c0"}  
14 Most silvers:  
15
```



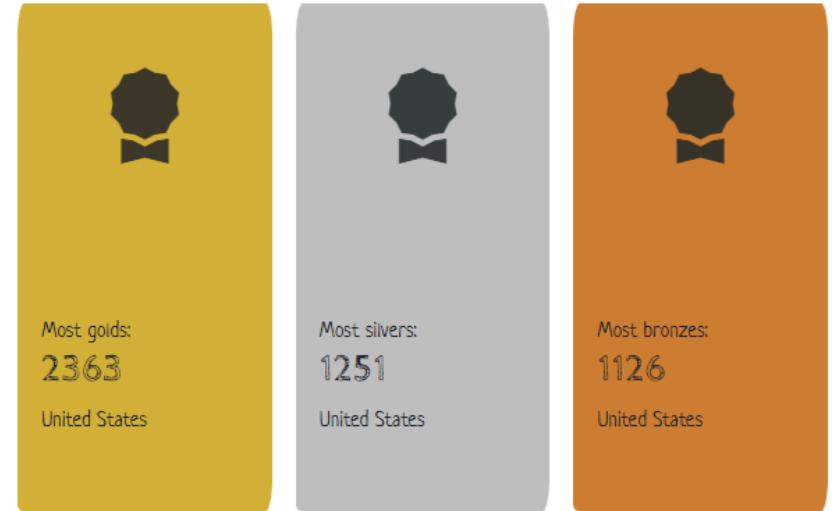
# Value Boxes

```
9
10 `.{python} most_gold_medals`
11 :::
12
13 :::. {.valuebox icon="award-fill" color="#c0c0c0"}
14 Most silvers:
15
16 `.{python} str(count_most_silver_medals)`
17
18 `.{python} most_silver_medals`
19 :::
20
21 :::. {.valuebox icon="award-fill" color="#cd7f32"}
22 Most bronzes:
23
```



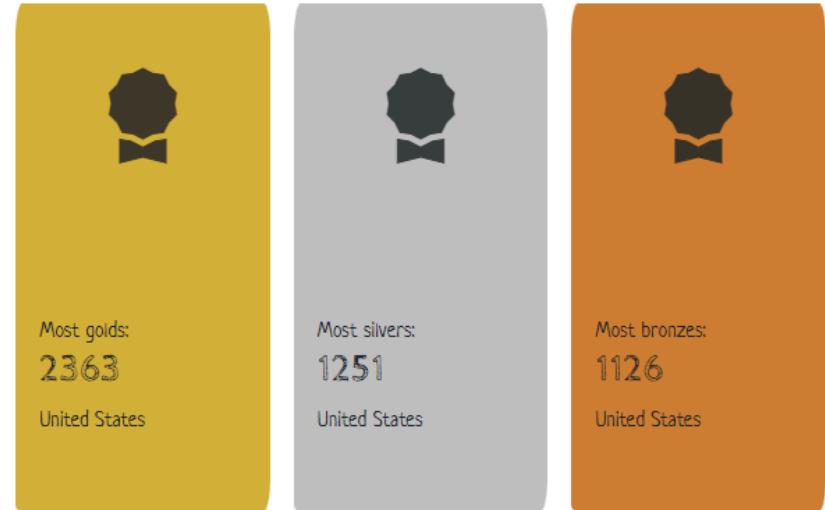
# Value Boxes

```
13 :::.valuebox icon="award-fill" color="#c0c0c0"}  
14 Most silvers:  
15  
16 `{python} str(count_most_silver_medals)`  
17  
18 `{python} most_silver_medals`  
19 :::  
20  
21 :::.valuebox icon="award-fill" color="#cd7f32"}  
22 Most bronzes:  
23  
24 `{python} str(count_most_bronze_medals)`  
25  
26 `{python} most_bronze_medals`  
27 :::
```



# Value Boxes

```
13 :::.valuebox icon="award-fill" color="#c0c0c0"}  
14 Most silvers:  
15  
16 `{python} str(count_most_silver_medals)`  
17  
18 `{python} most_silver_medals`  
19 :::  
20  
21 :::.valuebox icon="award-fill" color="#cd7f32"}  
22 Most bronzes:  
23  
24 `{python} str(count_most_bronze_medals)`  
25  
26 `{python} most_bronze_medals`  
27 :::
```



 `{python} str(count\_most\_gold\_medals)` is an inline code syntax that is supported in Quarto to easily mix Markdown text and computation output values.

# Tabssets

```
1 ---  
2 title: "Palmer Penguins"  
3 format: dashboard  
4 ---  
5  
6 ## Row  
7  
8 ```{python}  
9 ...  
10  
11 ## Row {.tabset}  
12  
13 ```{python}  
14 #| title: Chart 2  
15 ...
```

# Tabssets

```
4  ---
5
6  ## Row
7
8  ````{python}
9  ```
10
11 ## Row {.tabset}
12
13 ````{python}
14 #| title: Chart 2
15 ```
16
17 ````{python}
18 #| title: Chart 3
````
```



# Tabsets

```
5  
6 ## Row  
7  
8 ````{python}  
9 ...  
10  
11 ## Row {.tabset}  
12  
13 ````{python}  
14 #| title: Chart 2  
15 ...  
16  
17 ````{python}  
18 #| title: Chart 3  
19 ...
```



# Tabsets

```
5  
6 ## Row  
7  
8 ````{python}  
9 ...  
10  
11 ## Row {.tabset}  
12  
13 ````{python}  
14 #| title: Chart 2  
15 ...  
16  
17 ````{python}  
18 #| title: Chart 3  
19 ...
```



# Tabsets

```
5  
6 ## Row  
7  
8 ````{python}  
9 ``  
10  
11 ## Row {.tabset}  
12  
13 ````{python}  
14 #| title: Chart 2  
15 ``  
16  
17 ````{python}  
18 #| title: Chart 3  
19 ````
```

**Chart 1**

**Chart 2**

**Chart 3**

# Styling a Dashboard

```
1  ---
2  title: "Olympic Games"
3  format:
4    dashboard:
5      orientation: columns
6      nav-buttons: [github]
7      github: https://github.com/posit-conf-2024/olympicdash
8      theme:
9        - sketchy
10       - style/olympicdash.scss
11 logo: images/olympics-logo.svg
12 logo-alt: "Olympics logo with multicolored circles."
13 ---
```

# Styling a Dashboard

```
1  ---
2  title: "Olympic Games"
3  format:
4    dashboard:
5      orientation: columns
6      nav-buttons: [github]
7      github: https://github.com/posit-conf-2024/olympicdash
8      theme:
9        - sketchy
10       - style/olympicdash.scss
11 logo: images/olympics-logo.svg
12 logo-alt: "Olympics logo with multicolored circles."
13 ---
```

## Bootstrap & Bootswatch Themes:

Quarto Dashboard is based on Bootstrap  
with +20 themes from [Bootswatch](#) available

# Styling a Dashboard

```
1  ---
2  title: "Olympic Games"
3  format:
4    dashboard:
5      orientation: columns
6      nav-buttons: [github]
7      github: https://github.com/posit-conf-2024/olympicdash
8      theme:
9        - sketchy
10       - style/olympicdash.scss
11 logo: images/olympics-logo.svg
12 logo-alt: "Olympics logo with multicolored circles."
13 ---
```

## Bootstrap & Bootswatch Themes:

Quarto Dashboard is based on Bootstrap with +20 themes from [Bootswatch](#) available

## Customization through Quarto themes files:

Based on SASS, an easy way to declare new themes or variations to be layered into SCSS to be compiled to CSS

# Quarto Theme File

```
olympicdash.scss
```

```
1 /*-- scss:defaults --*/
2
3 // colors
4
5 $navbar-bg: #52515e;
6 $navbar-fg: #F0F0F0;
7
8 $link-color: #ae8b2d;
9 $hover-color: lighten($link-color, 40%);
10
11 /*-- scss:rules --*/
12 .card-header {
13   background-color: #ae8b2d50;
14 }
15
```



# Quarto Theme File

```
olympicdash.scss
```

```
11  /*-- scss:rules --*/
12  .card-header {
13    background-color: #ae8b2d50;
14  }
15
16  .nav-item > a:hover {
17    color: #ae8b2d;
18  }
19
20  .nav-link {
21    color: #F0F0F0;
22  }
23
24  .tabset .nav-link {
25    color: #52515e;
```



# Learning More

A lot more can be done with **Quarto Dashboard**:

- Deep dive at <https://quarto.org/docs/dashboards/>

# Learning More

A lot more can be done with **Quarto Dashboard**:

- Deep dive at <https://quarto.org/docs/dashboards/>

Advanced features:

- Interactivity with shiny for python
- Variation of a Dashboard using Parameters
- Easy Dashboard Deployment



# Learning More

A lot more can be done with **Quarto Dashboard**:

- Deep dive at <https://quarto.org/docs/dashboards/>

Advanced features:

- Interactivity with shiny for python
- Variation of a Dashboard using Parameters
- Easy Dashboard Deployment

And about Quarto ?

- Getting started with Quarto
- User guide
- Gallery of Examples

# Learning More

A lot more can be done with **Quarto Dashboard**:

- Deep dive at <https://quarto.org/docs/dashboards/>

Advanced features:

- Interactivity with shiny for python
- Variation of a Dashboard using Parameters
- Easy Dashboard Deployment

And about Quarto ?

- Getting started with Quarto
- User guide
- Gallery of Examples

Thank  
you!

Follow along at <https://cderv.github.io/pydata-paris-2024-quarto-dashboard/> and <https://quarto.org>

