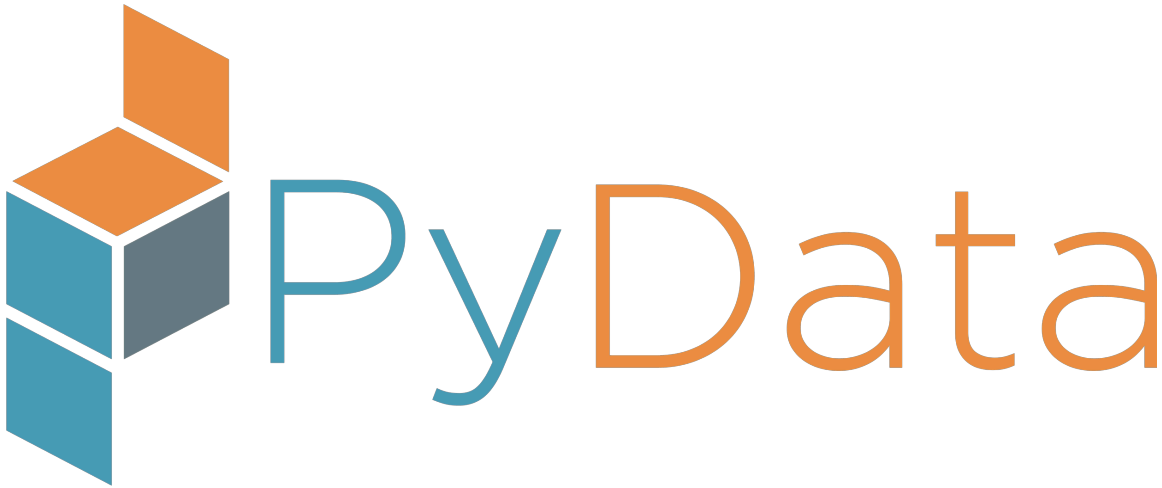
Quarto Dashboard

for impactful and visual communication

Christophe Dervieux

Posit PBC



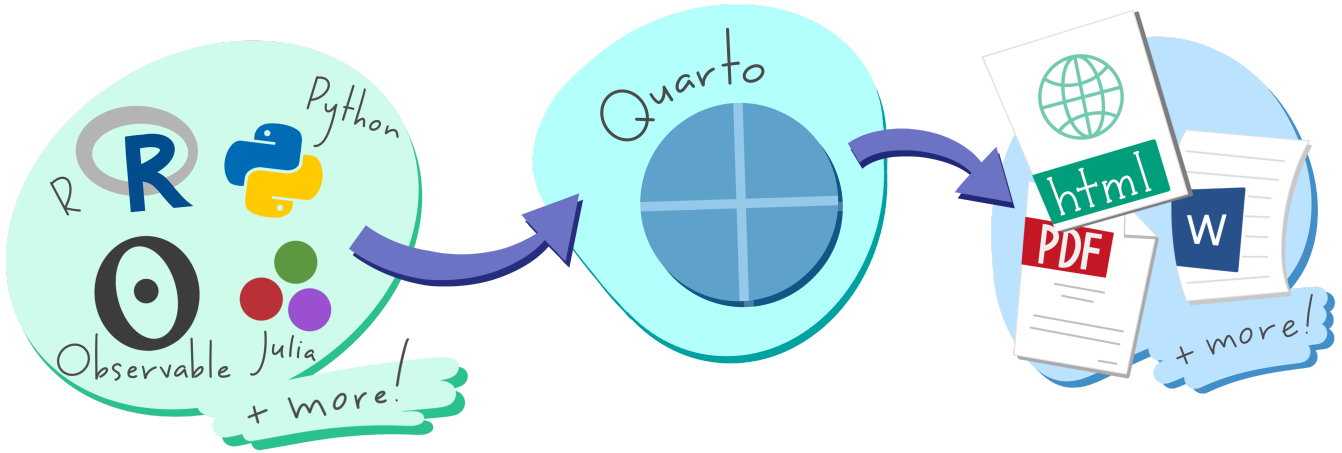
September 25, 2024

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



Artwork from “Hello, Quarto” keynote by Julia Lowndes and Mine Çetinkaya-Rundel, presented at RStudio Conference 2022. Illustrated by Allison Horst.

With Quarto …

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

With Quarto …

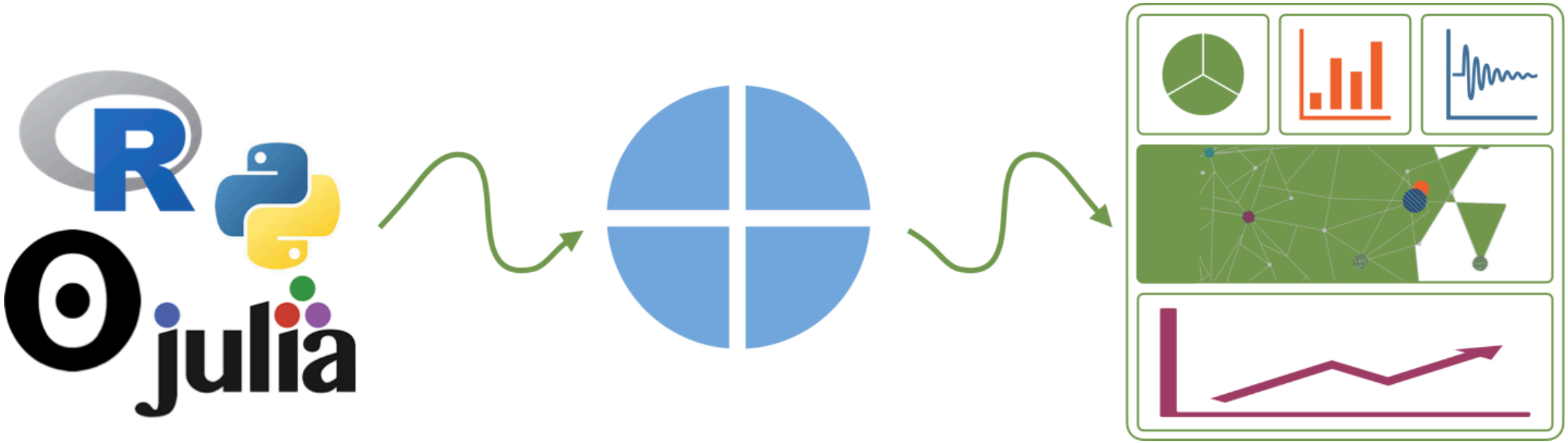
you can **weave** together **narrative** and **code** to produce elegantly formatted output as documents, web pages, blog posts, books and more…

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.

```
$ quarto --help

Usage:   quarto
Version: 1.5.57

Description:

  Quarto CLI

Options:

  -h, --help     - Show this help.
  -V, --version  - Show the version number for this program.

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.

```
$ quarto --help

Usage:   quarto
Version: 1.5.57

Description:

  Quarto CLI

Options:

  -h, --help     - Show this help.
  -V, --version  - Show the version number for this program.

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.

```
$ quarto --help

Usage:   quarto
Version: 1.5.57

Description:

  Quarto CLI

Options:

  -h, --help     - Show this help.
  -V, --version  - Show the version number for this program.

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.

```
Options:

  -h, --help      - Show this help.
  -V, --version   - Show the version number for this program.

Commands:

  render      [input] [args...]     - Render files or projects to various document types.
  preview     [file] [args...]      - Render and preview a document or website project.
  serve       [input]               - Serve a Shiny interactive document.
  create      [type] [commands...]  - Create a Quarto project or extension
  use         <type> [target]       - Automate document or project setup tasks.
  add         <extension>           - Add an extension to this folder or project
  update      [target...]           - Updates an extension or global dependency.
  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...]     - 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.
25    convert     <input>               - 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.

```
Commands:

  render      [input] [args...]      - Render files or projects to various document types.
  preview     [file] [args...]       - Render and preview a document or website project.
  serve       [input]                - Serve a Shiny interactive document.
  create      [type] [commands...]   - Create a Quarto project or extension
  use         <type> [target]        - Automate document or project setup tasks.
  add         <extension>            - Add an extension to this folder or project
  update      [target...]            - Updates an extension or global dependency.
  remove      [target...]            - Removes an extension.
  convert     <input>                - Convert documents to alternate representations.
  pandoc      [args...]              - Run the version of Pandoc embedded within Quarto.
  typst       [args...]              - Run the version of Typst embedded within Quarto.
  run         [script] [args...]     - Run a TypeScript, R, Python, or Lua script.
  install     [target...]            - Installs a global dependency (TinyTex or Chromium).
  uninstall   [tool]                 - 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:

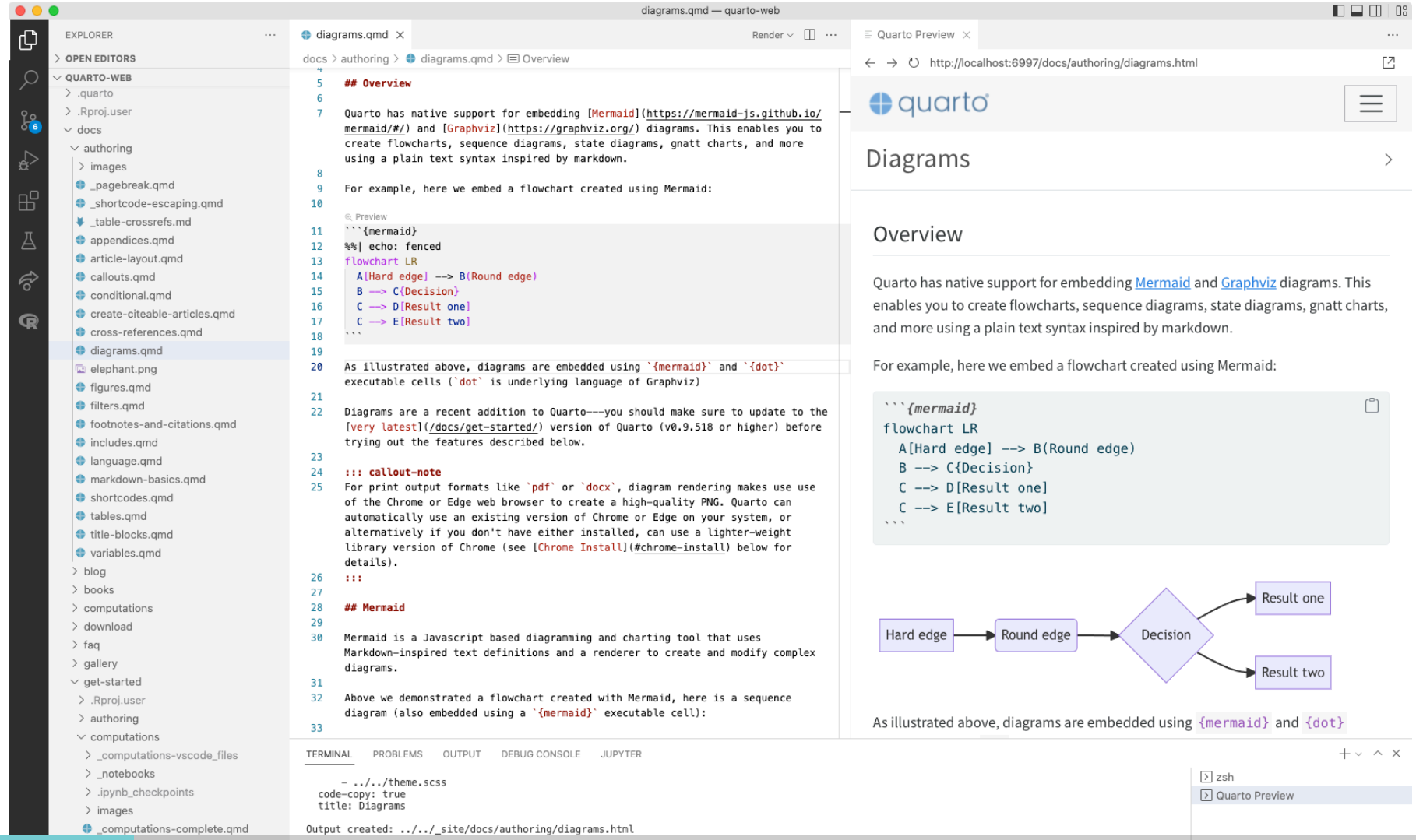
```
# ipynb notebook
quarto preview notebook.ipynb
quarto preview notebook.ipynb --execute
quarto preview notebook.ipynb --to revealjs

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

How to use Quarto?

Quarto integrates with other tools.

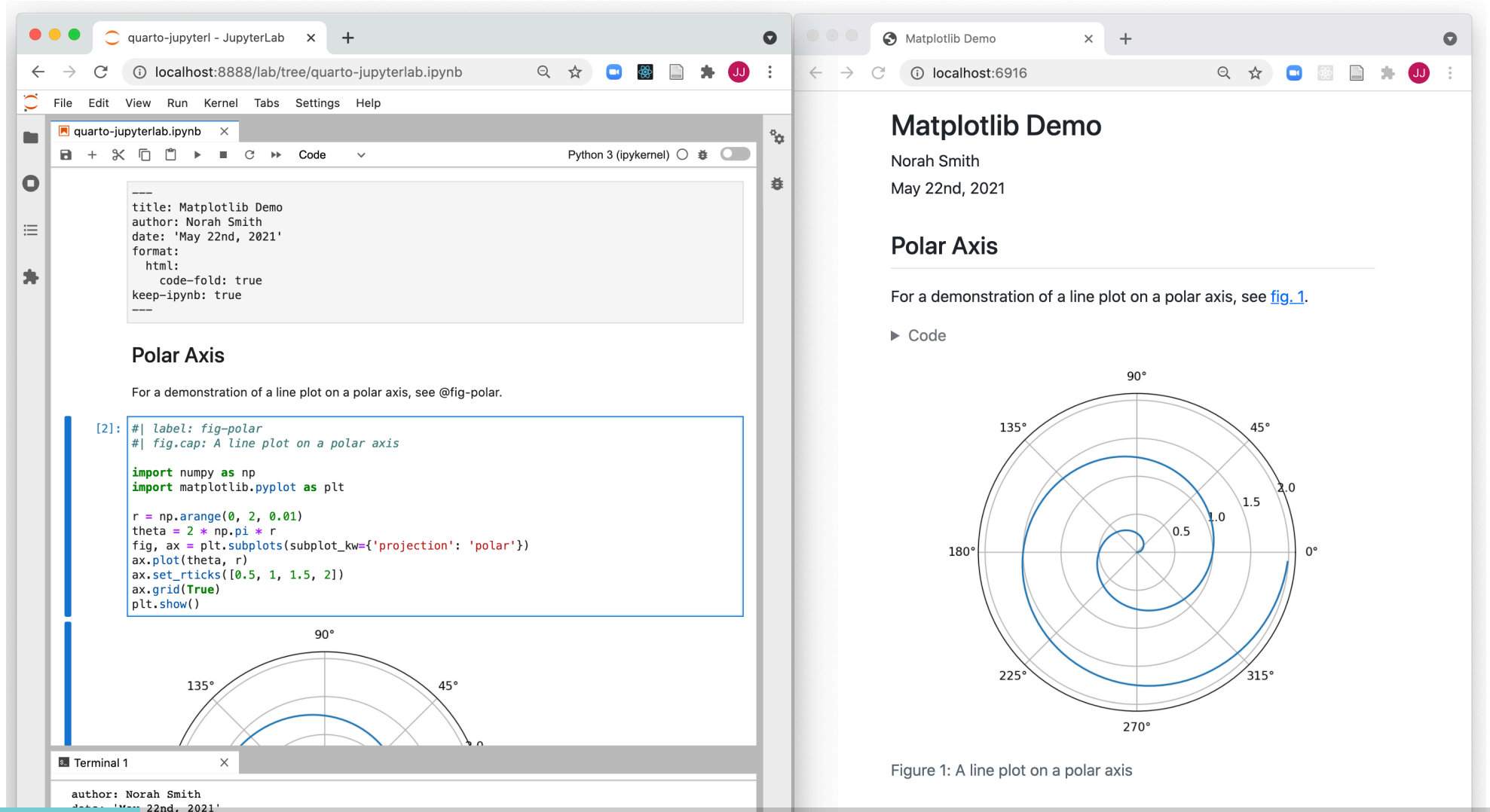
How to use Quarto?

Quarto integrates with other tools.



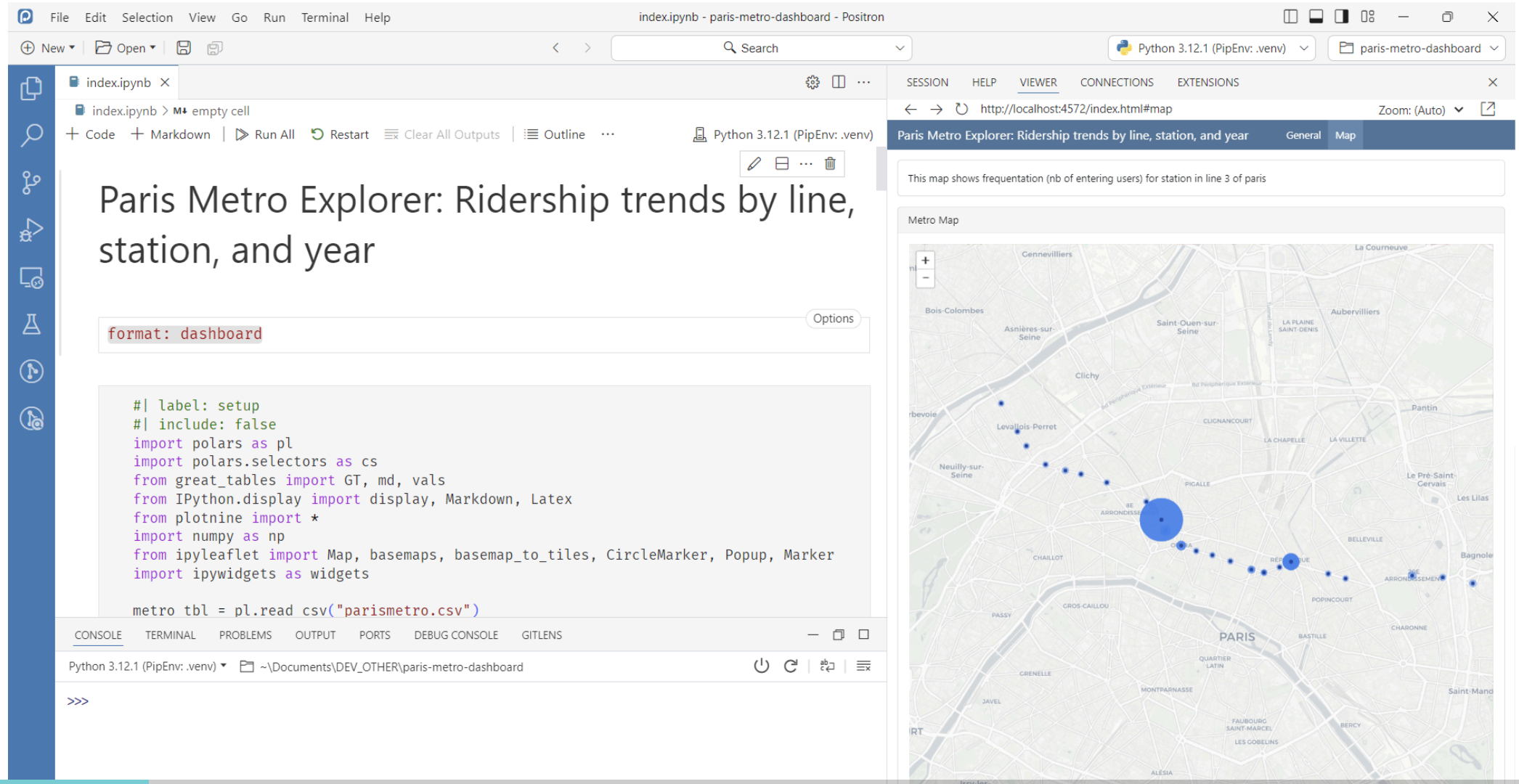
How to use Quarto?

Quarto integrates with other tools.



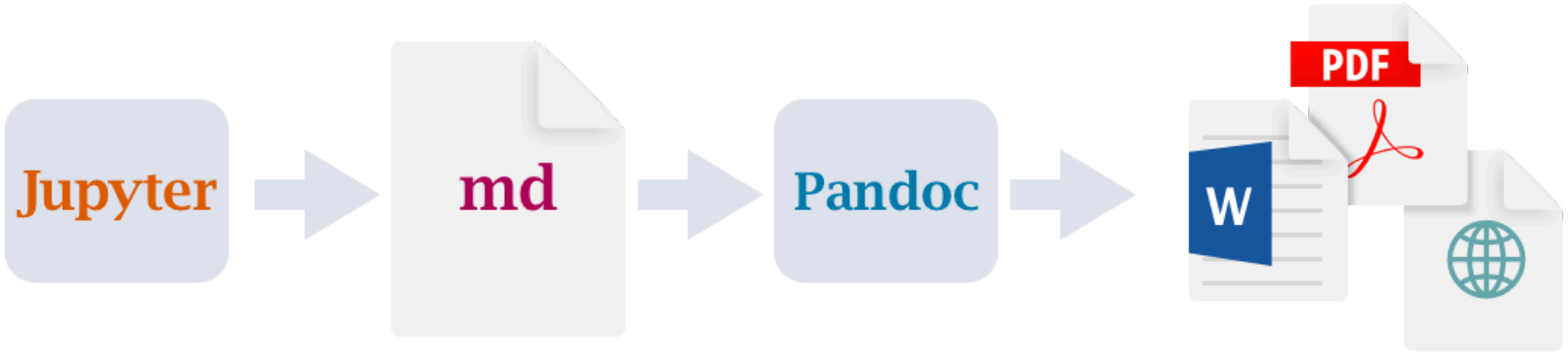
How to use Quarto?

Quarto integrates with other tools.

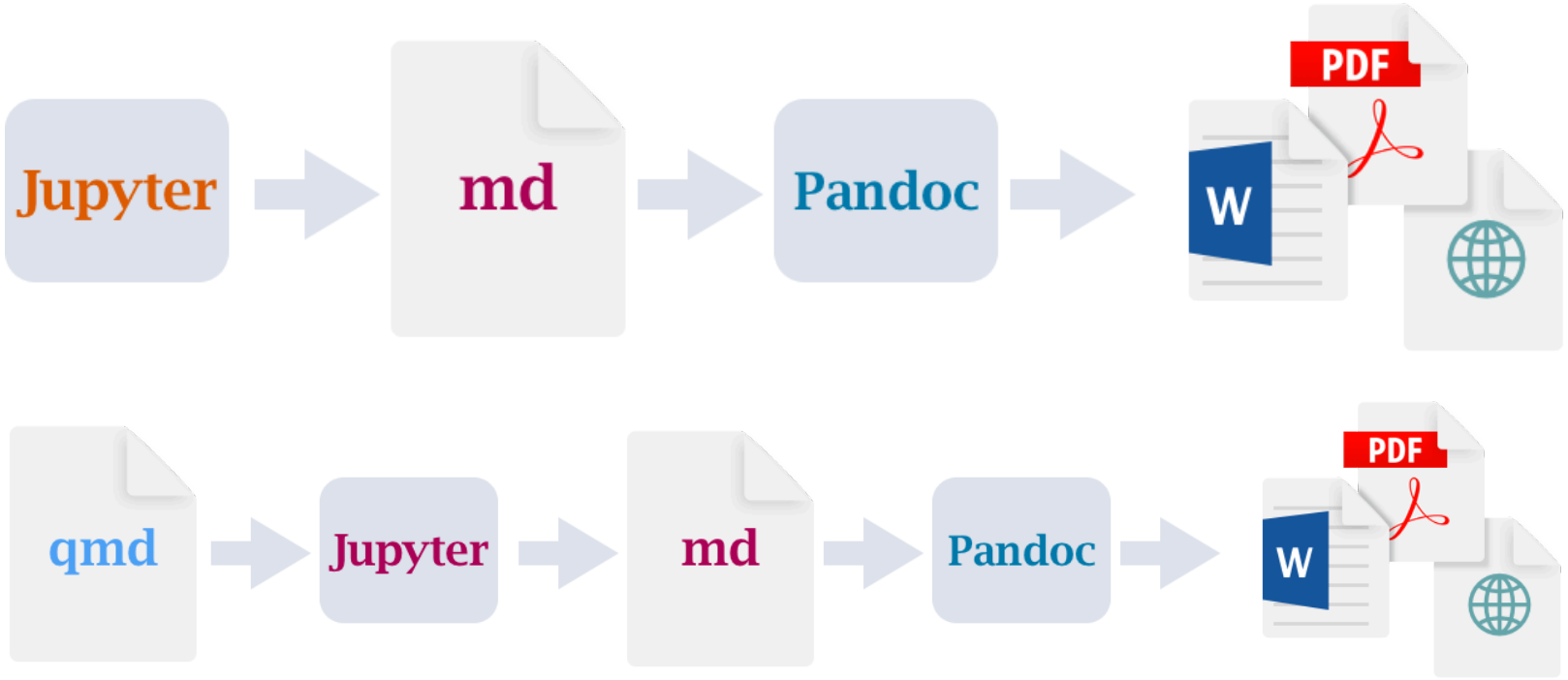


Quarto Workflow

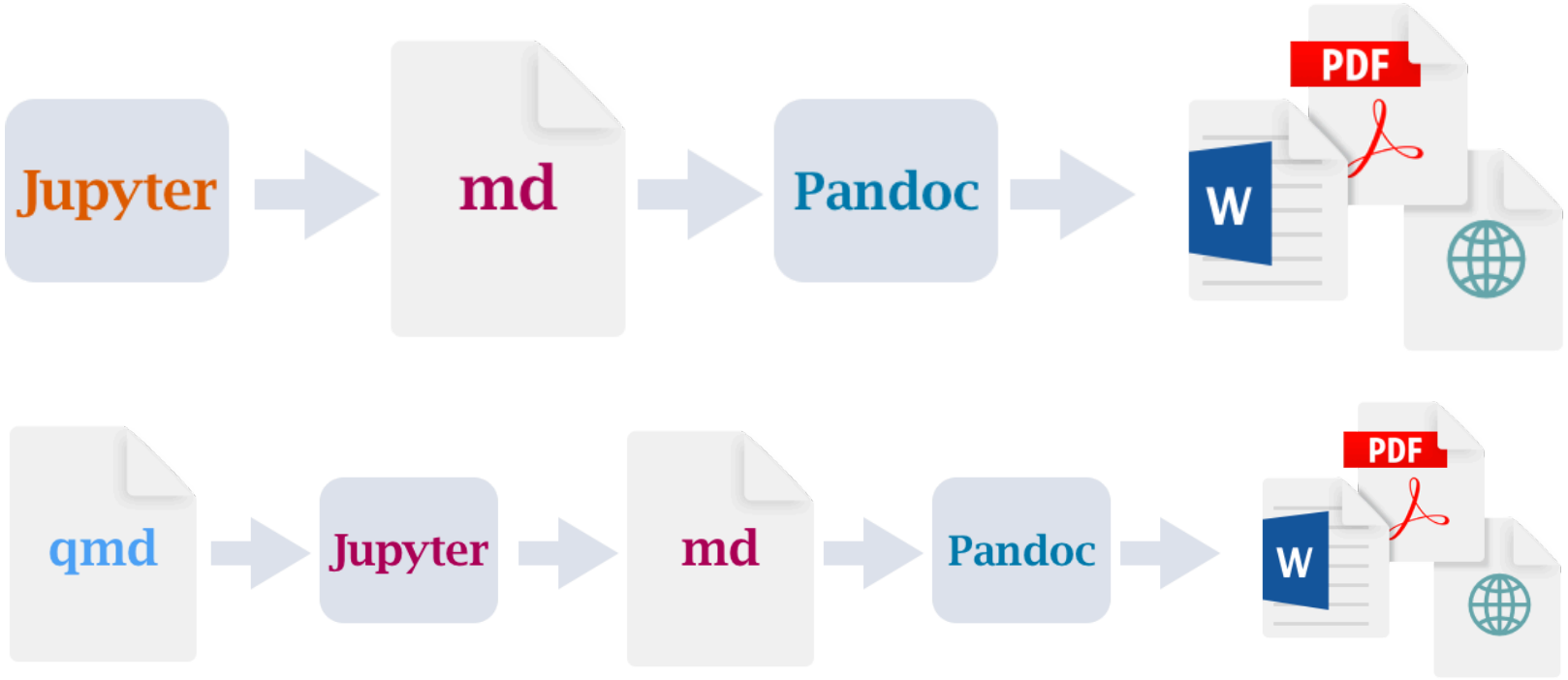
Quarto Workflow



Quarto Workflow

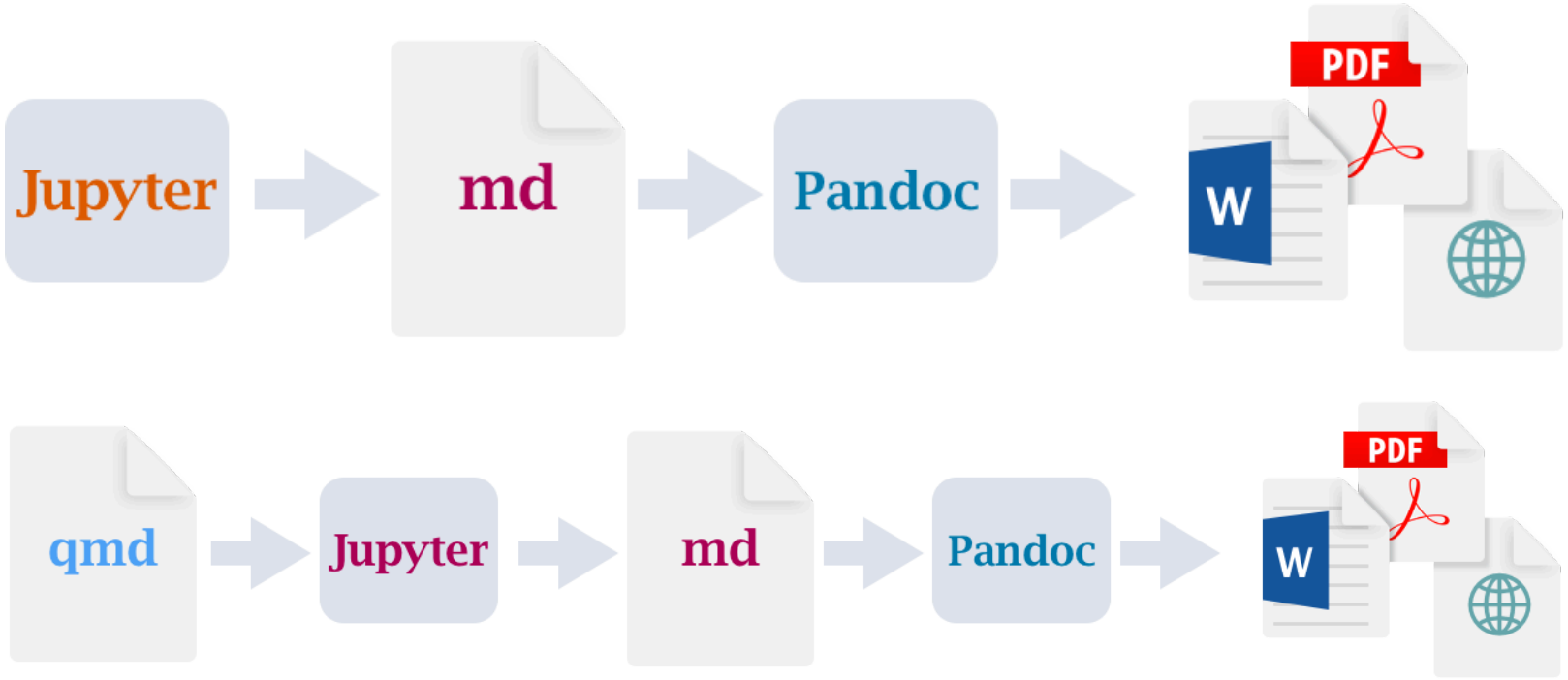


Quarto Workflow



Computations: Jupyter (and Knitr and ObservableJS)

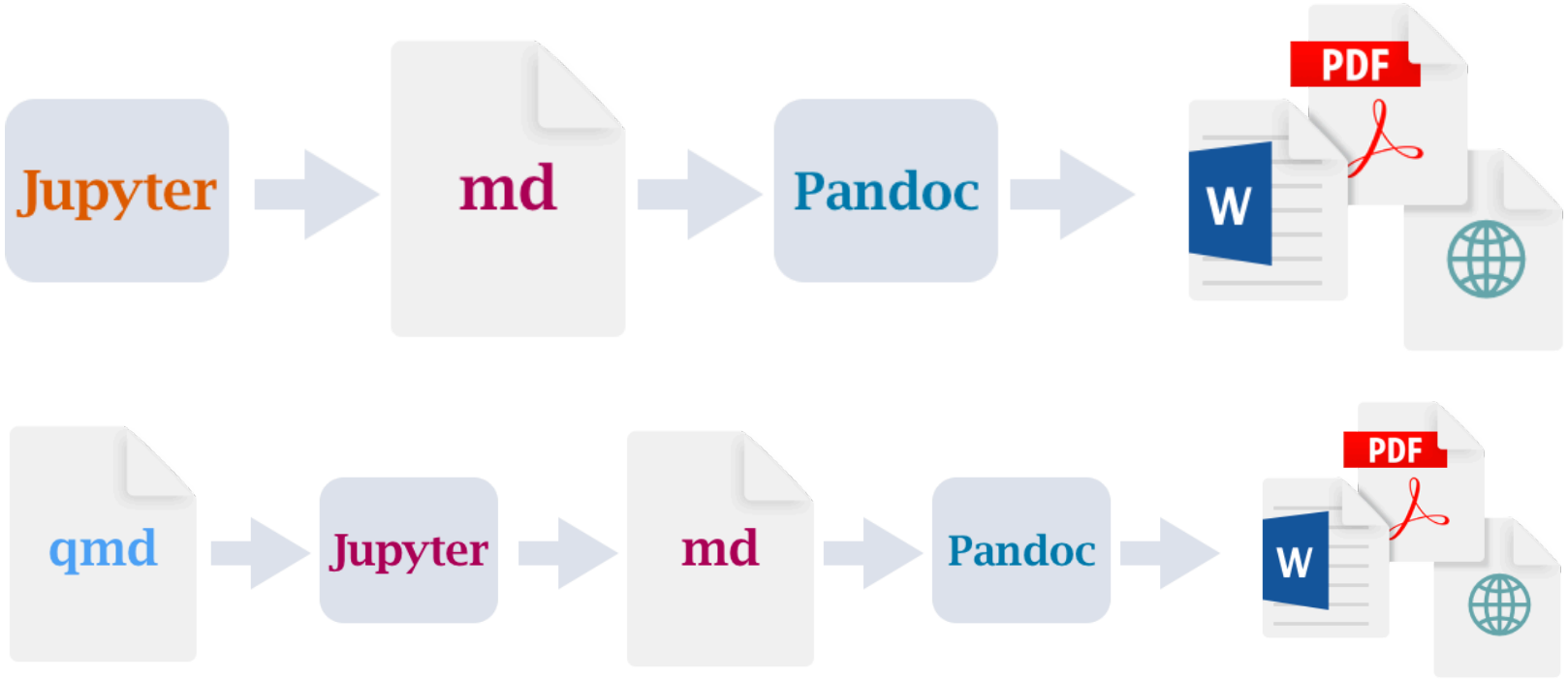
Quarto Workflow



Computations: Jupyter (and Knitr and ObservableJS)

Markdown: Pandoc with many enhancements

Quarto Workflow

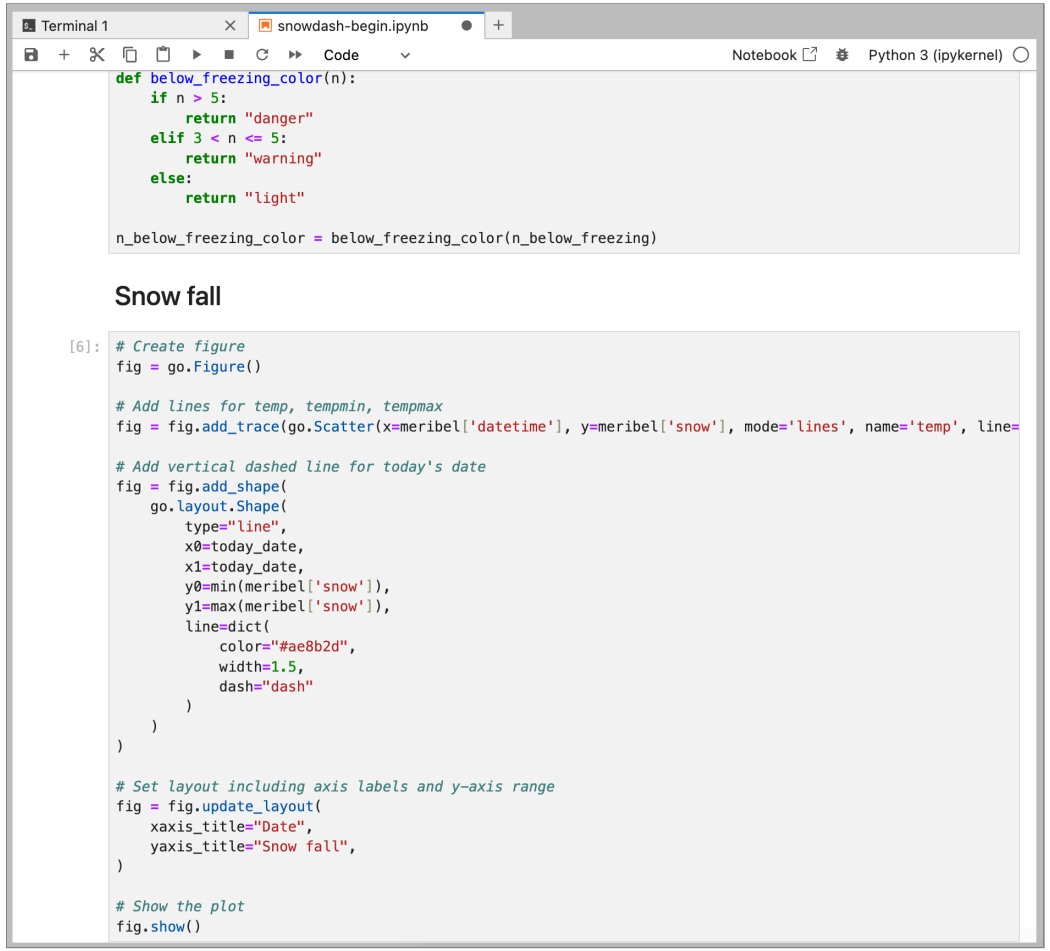


Computations: Jupyter (and Knitr and ObservableJS)

Markdown: Pandoc with many enhancements

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

Render Notebook to HTML (default options)



Render Notebook to HTML (default options)

Terminal 1 | snowdash-begin.ipynb

Code | Notebook | Python 3 (ipykernel)

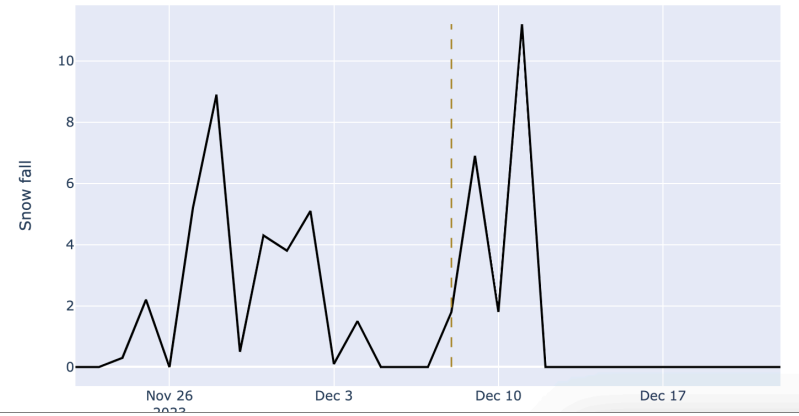
```
def below_freezing_color(n):
    if n > 5:
        return "danger"
    elif 3 < n <= 5:
        return "warning"
    else:
        return "light"

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()
```

```
        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()
```



Render Notebook to HTML (document level options)

Terminal 1 | snowdash-begin.ipynb

Code | Notebook | Python 3 (ipykernel)

```
---
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)
```

Snow fall

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

Render Notebook to HTML (document level options)

Terminal 1 | snowdash-begin.ipynb

Code | Notebook | Python 3 (ipykernel)

```
---
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)
```

Snow fall

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

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

# Add vertical dashed line for today's date
fig = fig.add_shape(
    go.layout.Shape(
```

Render Notebook to HTML (document and cell level options)

Terminal 1 | snowdash-begin.ipynb

Raw | Notebook | Python 3 (ipykernel)

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

Render Notebook to HTML (document and cell level options)

Terminal 1 | snowdash-begin.ipynb

Raw | Notebook | Python 3 (ipykernel)

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

Dashing through the snow ❄

</> Code

Show All Code
Hide All Code

▶ Code
▶ Code
▶ Code
▶ Code
▶ Code

Snow fall

▶ Code

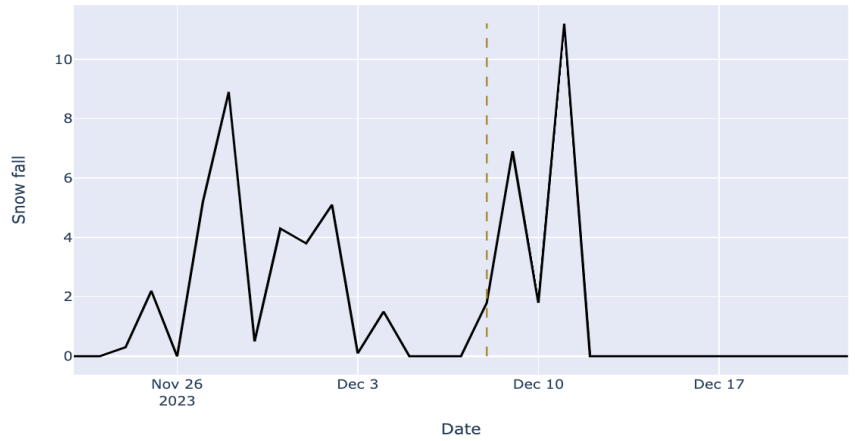


Figure 1: Snow fall in Meribel

Data

▶ Code

Show 10 entries

Search:

Render Notebook to Revealjs — https://quarto.org/docs/presentations/revealjs/

Terminal 1 | snowdash-begin.ipynb

Raw | Notebook | Python 3 (ipykernel)

```
---
title: "Dashing through the snow ❄"
format: revealjs
---
```

[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)
```

Render Notebook to Revealjs — https://quarto.org/docs/presentations/revealjs/

Terminal 1 | snowdash-begin.ipynb

Raw | Notebook | Python 3 (ipykernel)

```
---
title: "Dashing through the snow ❄"
format: revealjs
---
```

[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

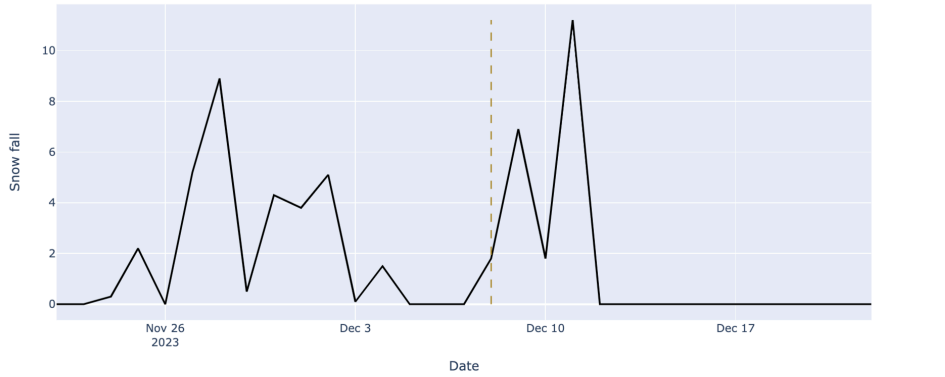


Figure 1: Snow fall in Meribel

Peeking at Dashboard Examples

DEMOCO Customer Churn | Churn (Standard) | Data

Churn rate 2022

17%

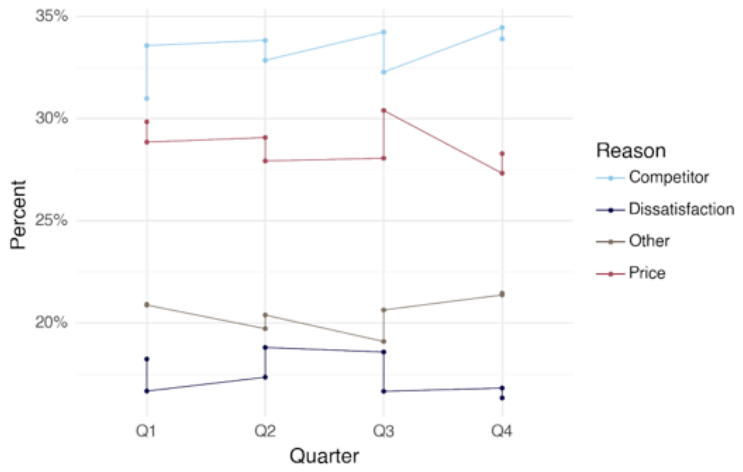
Current churn rate

14%

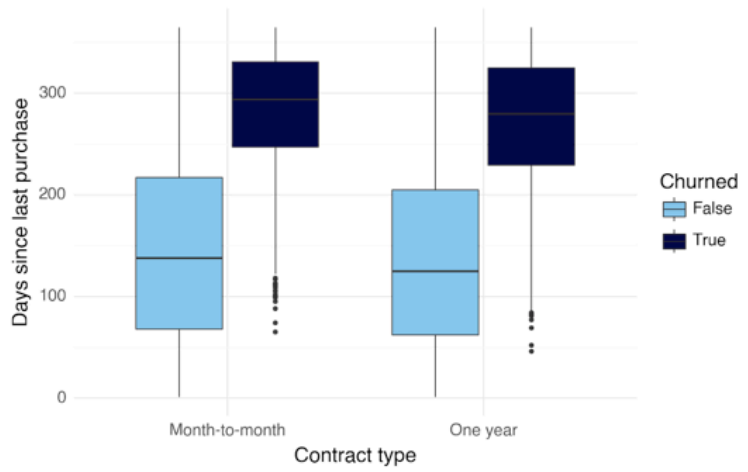
Churn rate goal

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

Dashing through the snow

ALL FORECAST HISTORICAL DATA

This dashboard displays weather data for:

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

The data were gathered from the Virtual Crossing.

Measurement units

Today

2023-12-08

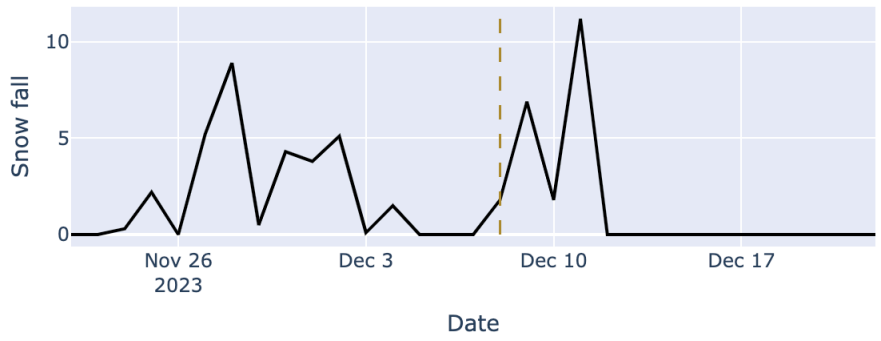
Snowy days

14

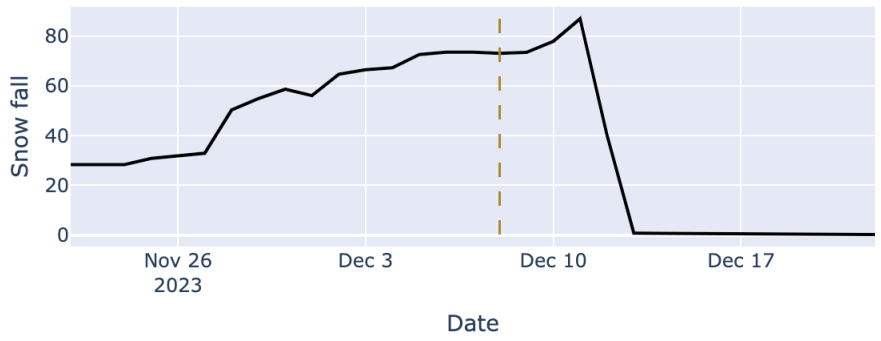
Number of days below freezing

7

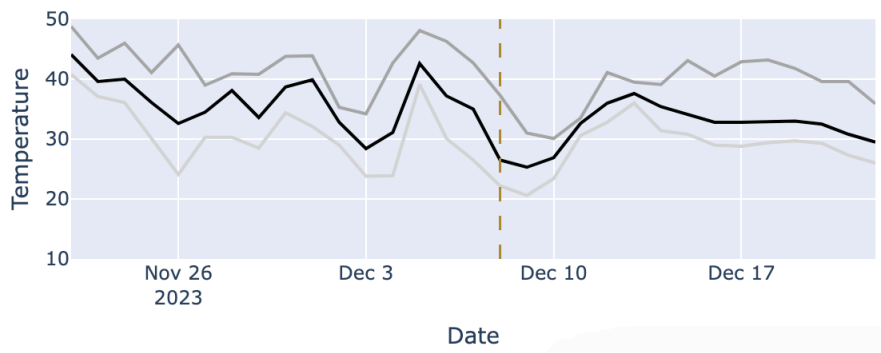
Snow fall (in)



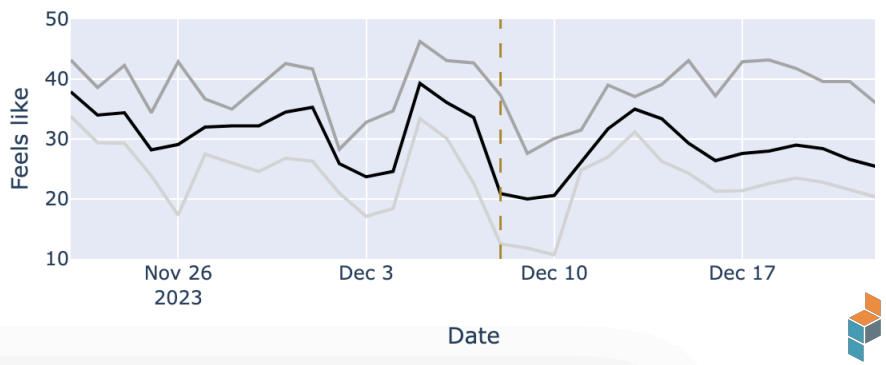
Snow depth (in)



Temperature (F)



Feels like (F)



Recent Earthquakes in Aotearoa New Zealand Of Weak Intensity or Greater

Hours since last earthquake

13

Earthquakes in the last 24 hours

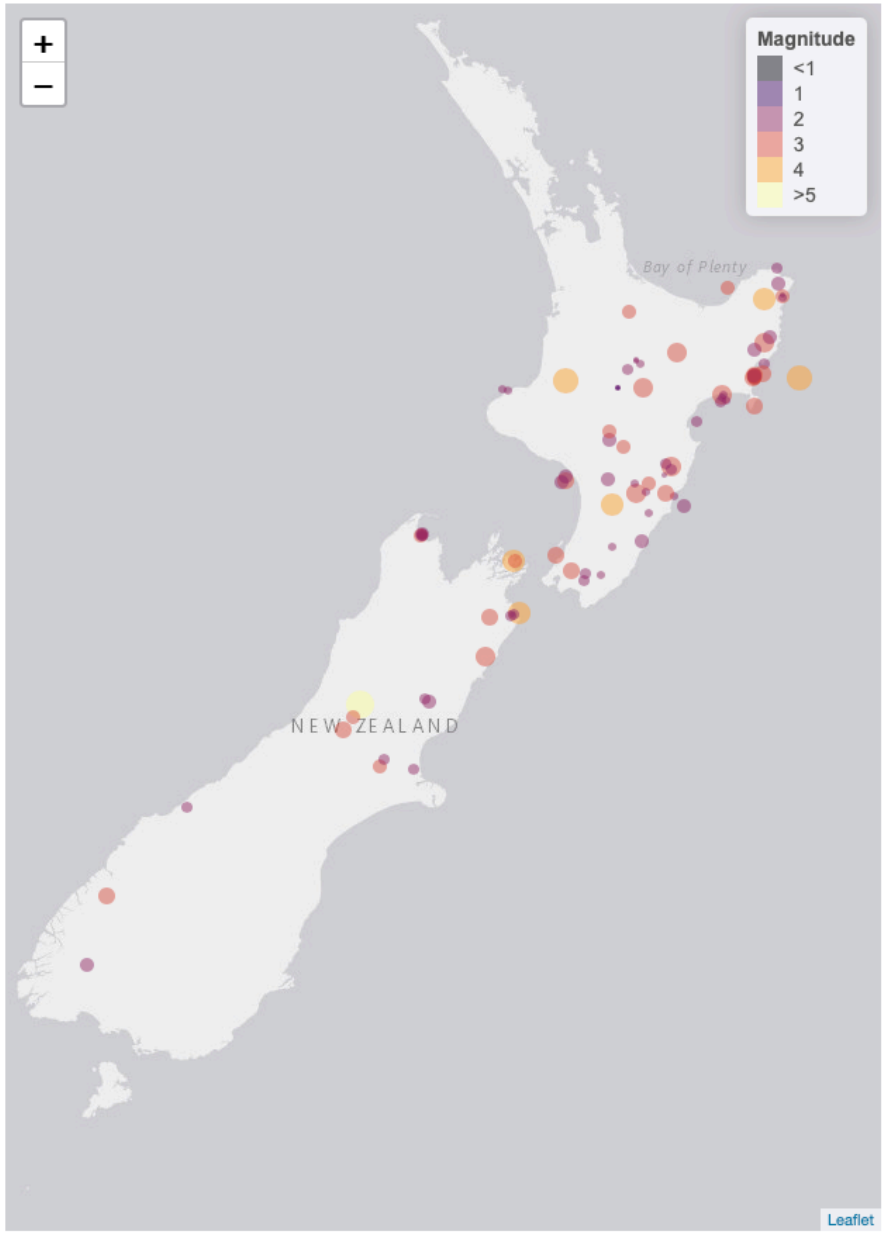
2

Last 10 Earthquakes

Magnitude			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

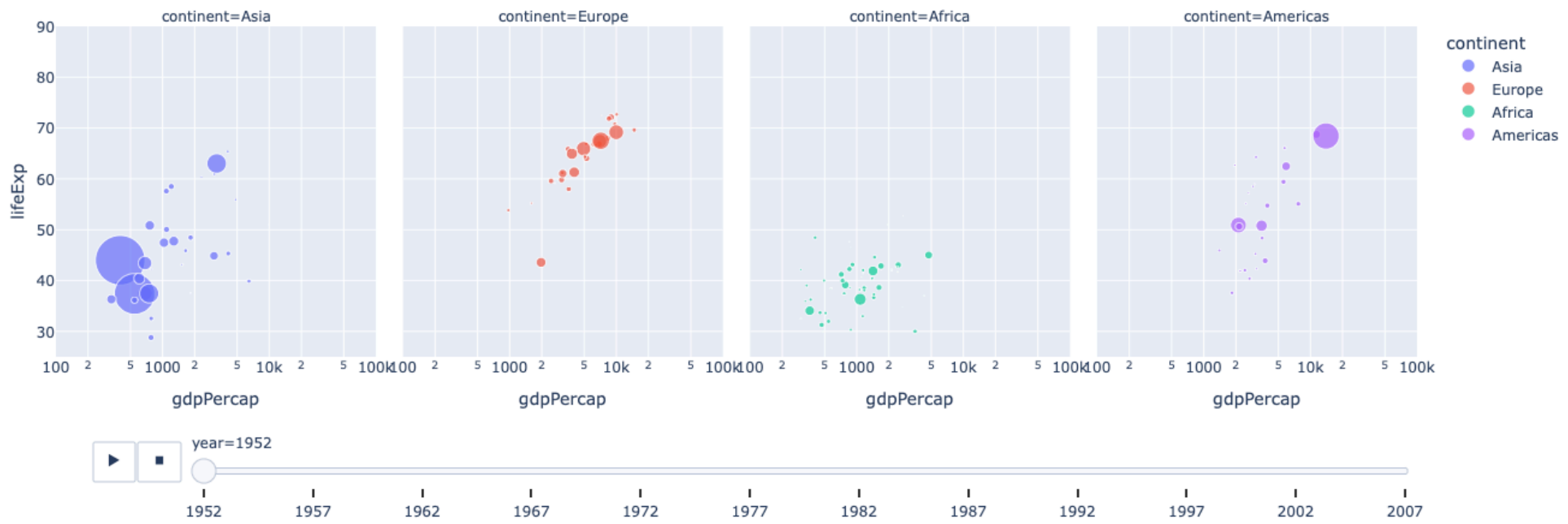
100 Most Recent Earthquakes



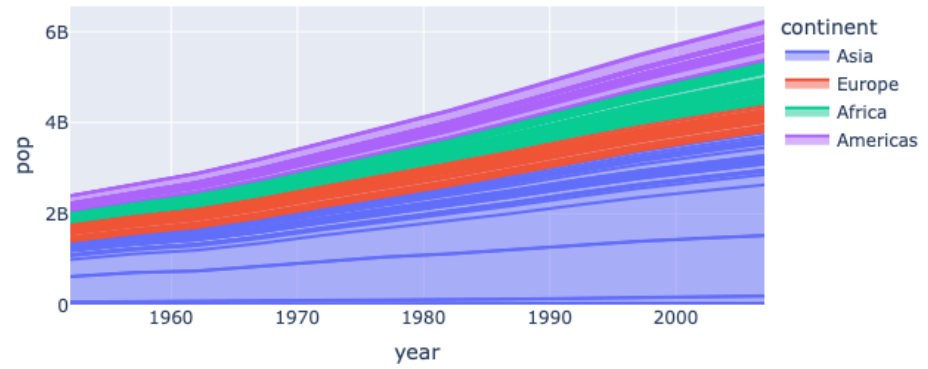
PyData

Worldwide Development Gapminder Analytics | Indicators | Data

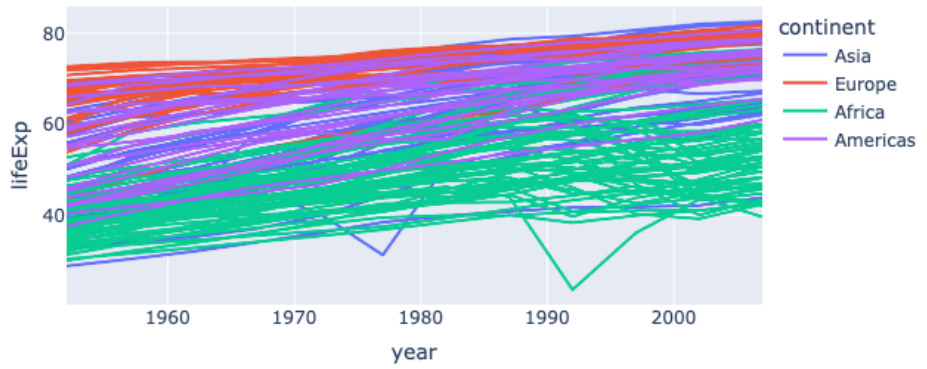
GDP and Life Expectancy



Population



Life Expectancy



PyData

7.79%

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

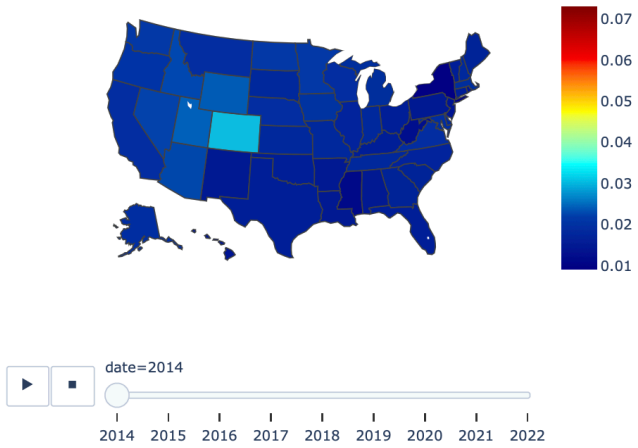
7.03%

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

$388k

National median home price

Mortgage Originations per Capita

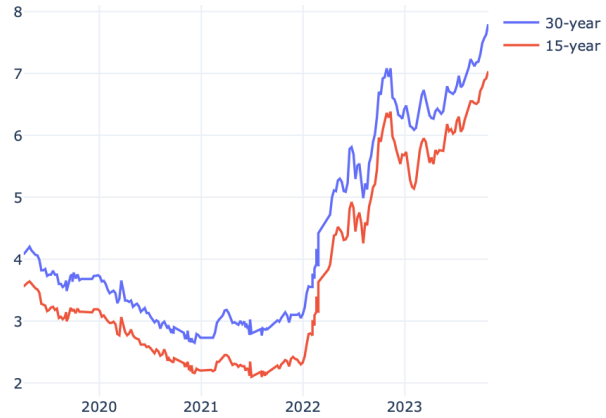


Mortage Originations

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

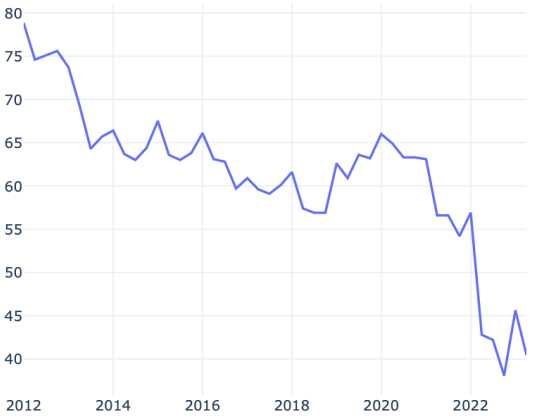
Housing Economics | Median Price

Interest Rates 15- and 30-Year



Housing Opportunity Index

Percentage of homes affordable to median-income families



Labor and Delivery Dashboard

ALL | VAGINAL | CAESAREAN | DATA

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%

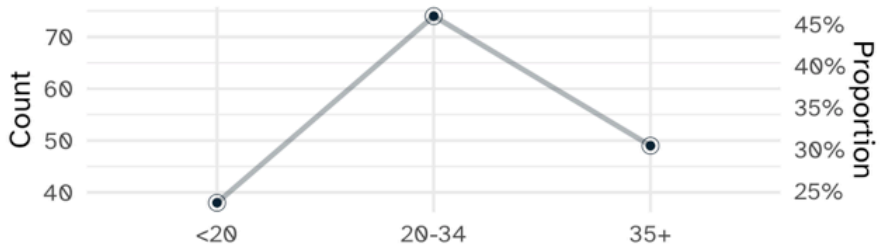
Pre-term births

18.6%

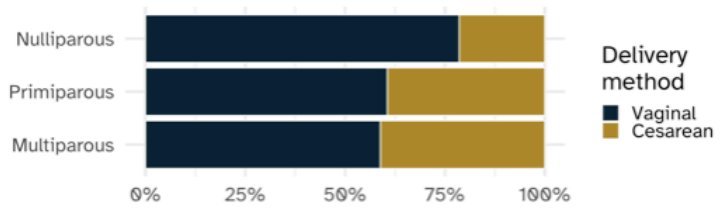
Delivery method

	Number of deliveries	Proportion of deliveries
Cesarean	54	33.5%
Vaginal	107	66.5%

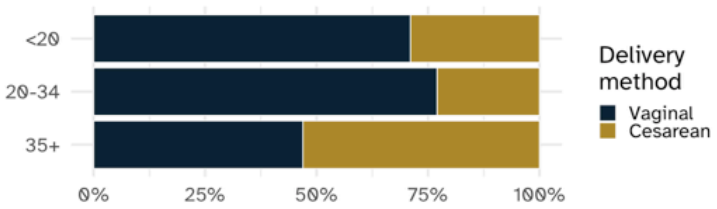
Maternal age



Delivery method and parity



Delivery method and maternal age



PyData

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'},
 'total_violations': {'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.

MyNorfolk Requests

Today | Week | About

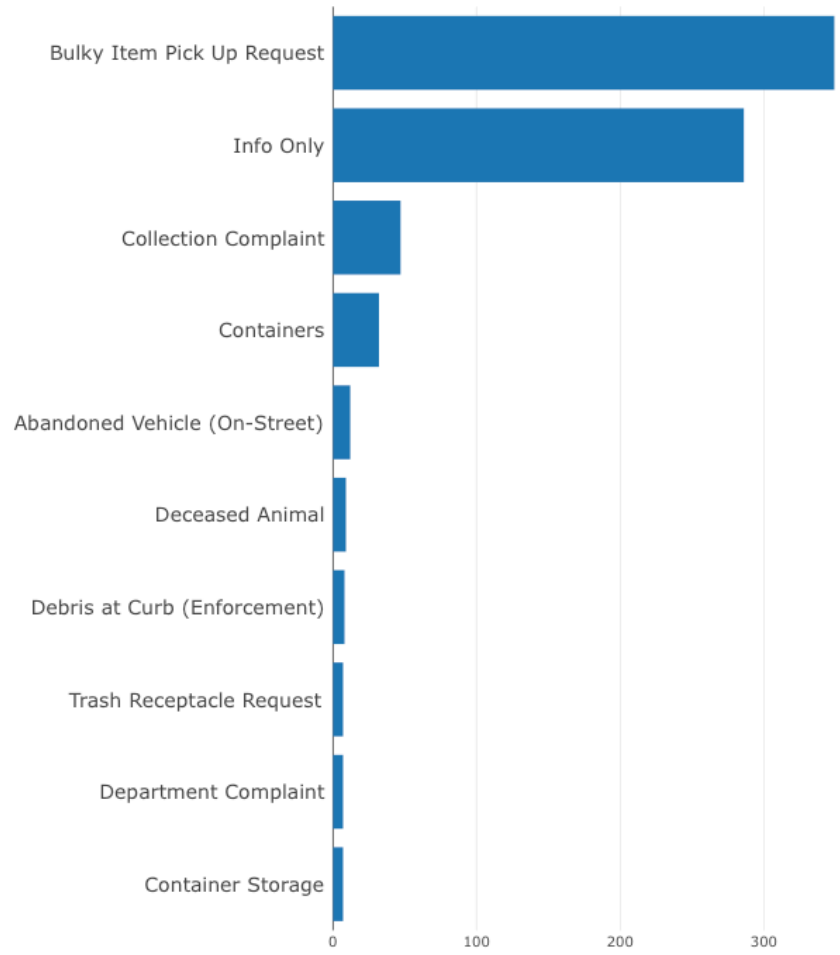
In Progress
13,008
▲ 176 were started today

Opened
836
▲ 752 more than yesterday

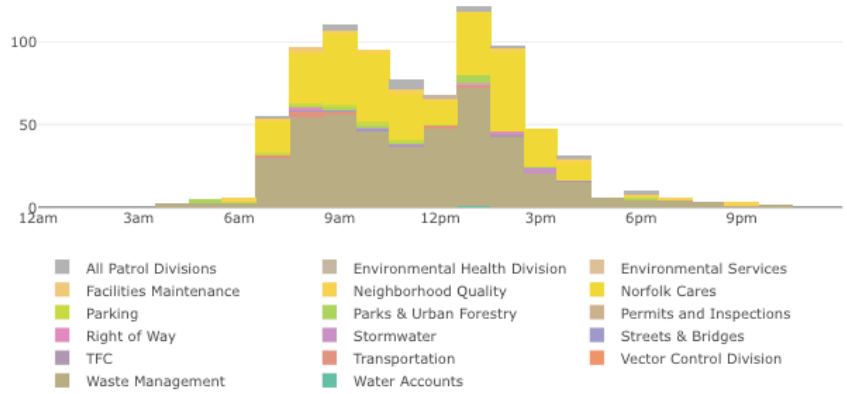
Closed
340
▲ 339 more than yesterday

Cancelled
26
▲ 26 more than yesterday

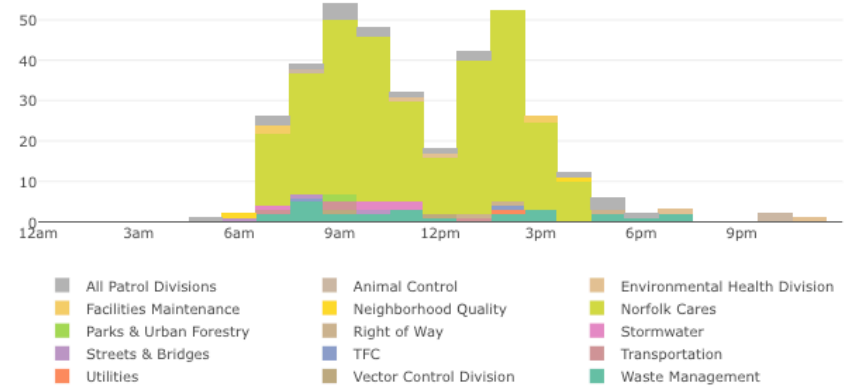
Top New Requests | Top In Progress | Top Closed Today



Opened by Hour of Day



Resolved by Hour of Day



PyData

NFL Injuries — 10 Weeks Max Bade

Stats | Data

Average Injured

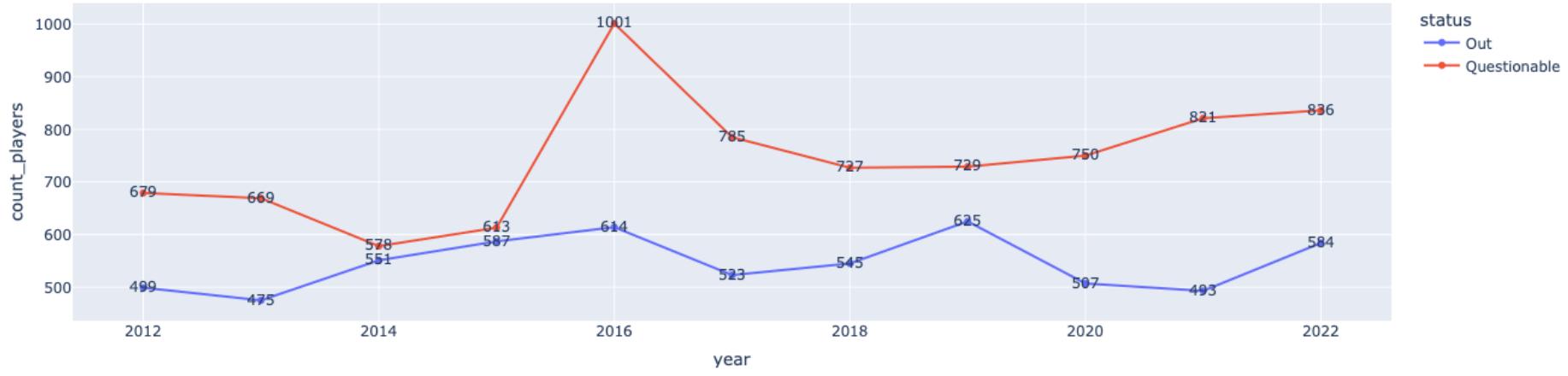
1289

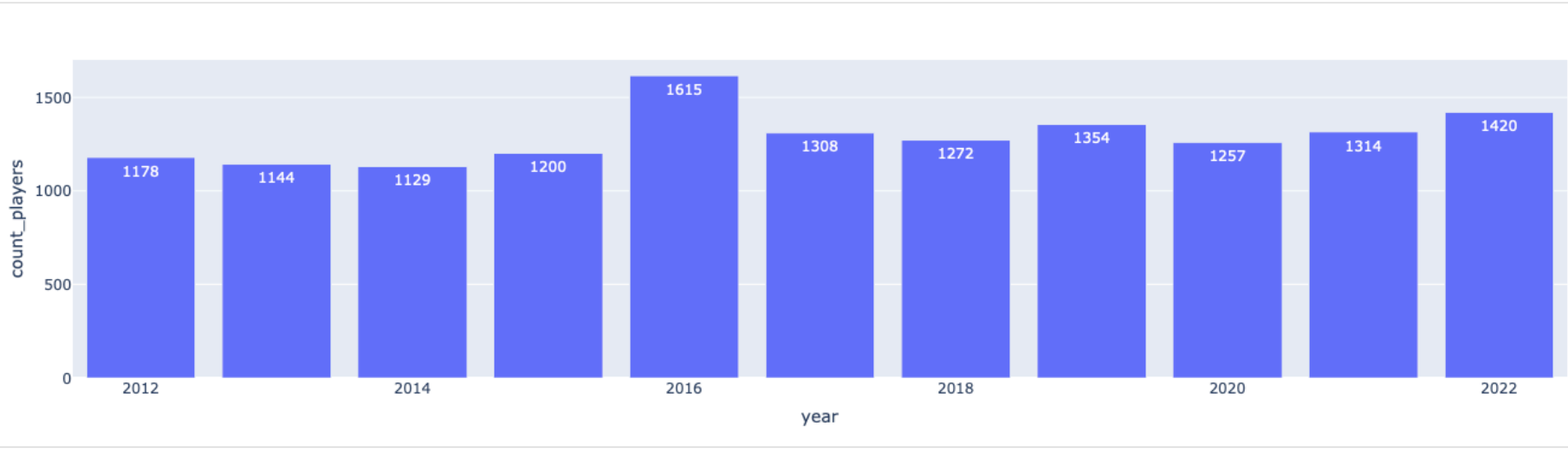
Average Out

545

Average Questionable

744





PyData

Retirement: simulating wealth with random returns, inflation and withdrawals

Scenario A

Initial investment

$100,000 | $2,000,000 | $10,000,000

Average annual investment return

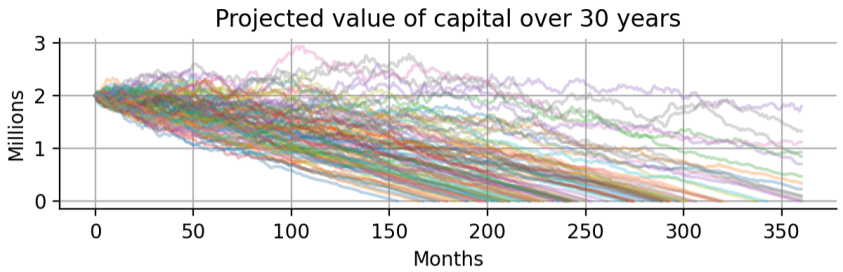
0% | 5% | 30%

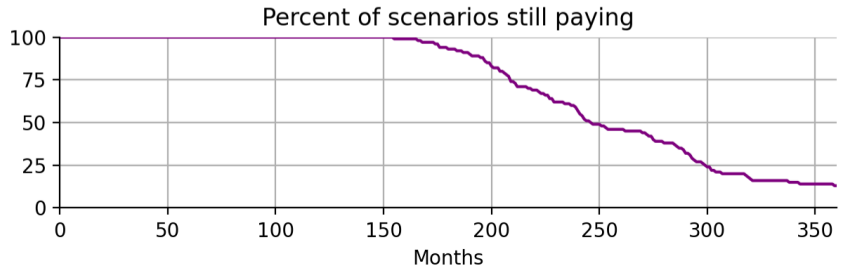
Average annual inflation

0% | 2.5% | 20%

Monthly withdrawals

$0 | $10,000 | $50,000





Scenario B

Initial investment

$100,000 | $2,000,000 | $10,000,000

Average annual investment return

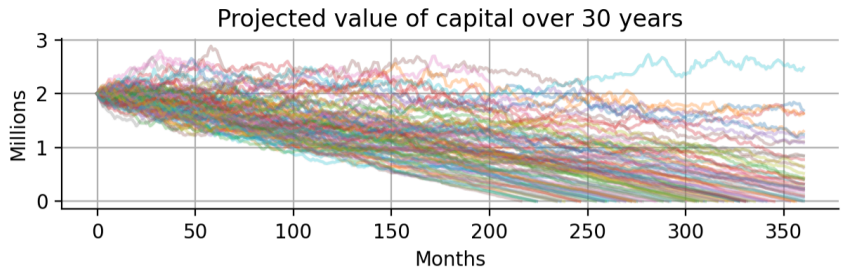
0% | 5% | 30%

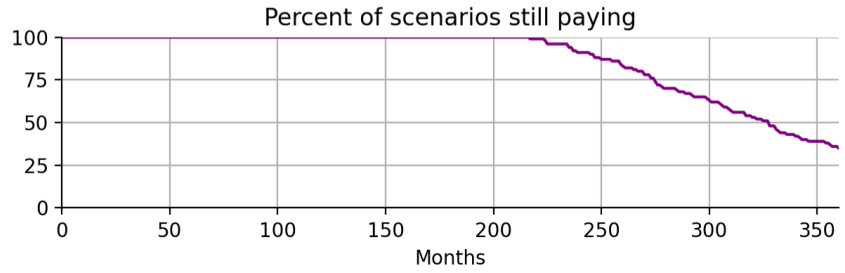
Average annual inflation

0% | 2.5% | 20%

Monthly withdrawals

$0 | $8,000 | $50,000





PyData

Palmer Penguins Cobblepot Analytics

Plots Data

Species:

- [x] Adelie
- [x] Gentoo
- [x] Chinstrap

Islands:

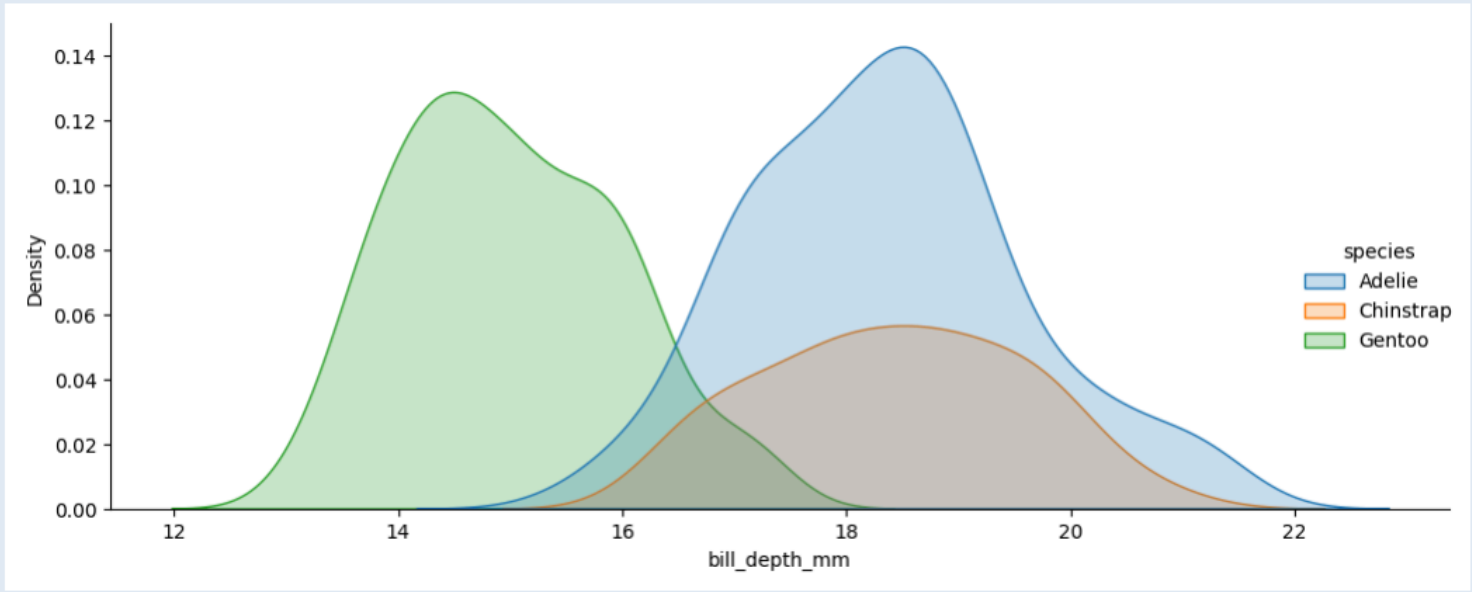
- [x] Biscoe
- [x] Dream
- [x] Torgersen

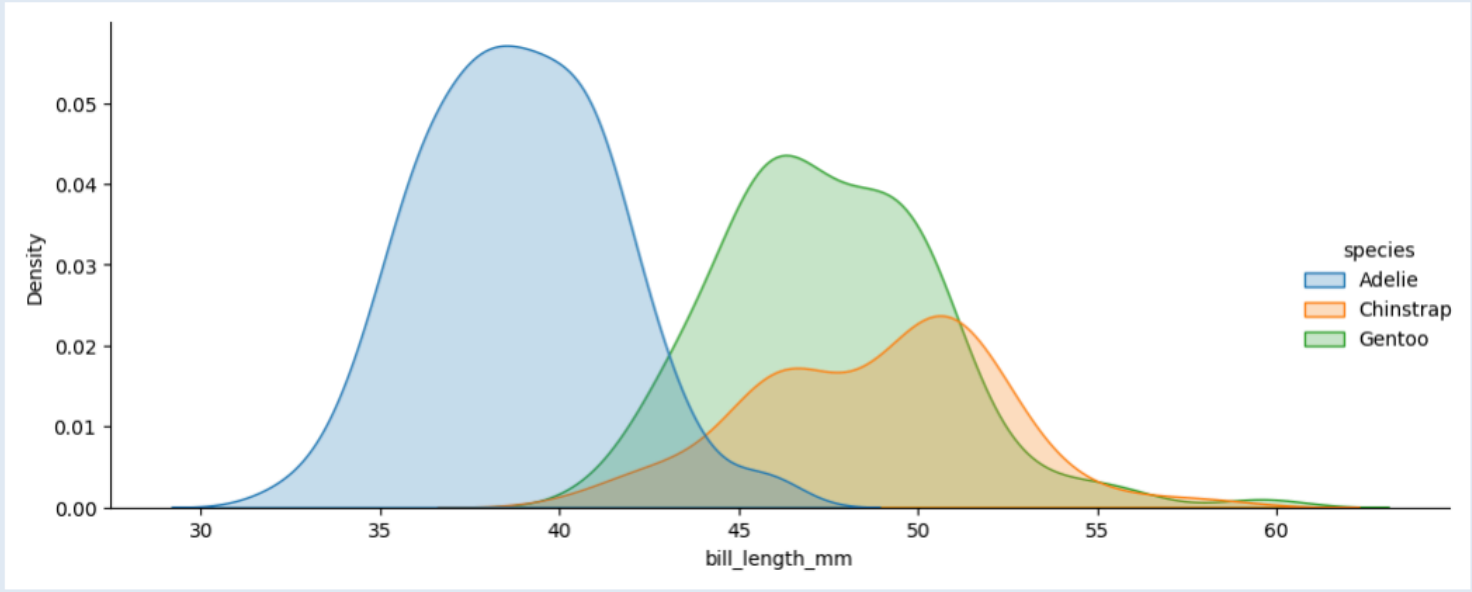
Distribution:

kde

- [ ] Show rug marks

Learn more about the Palmer Penguins dataset.





PyData

DEMOCO Stock Explorer Ty Coon

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

Current Price	Change	Percent Change
$ 179.69	↑ $0.60	% 0.34

Price History

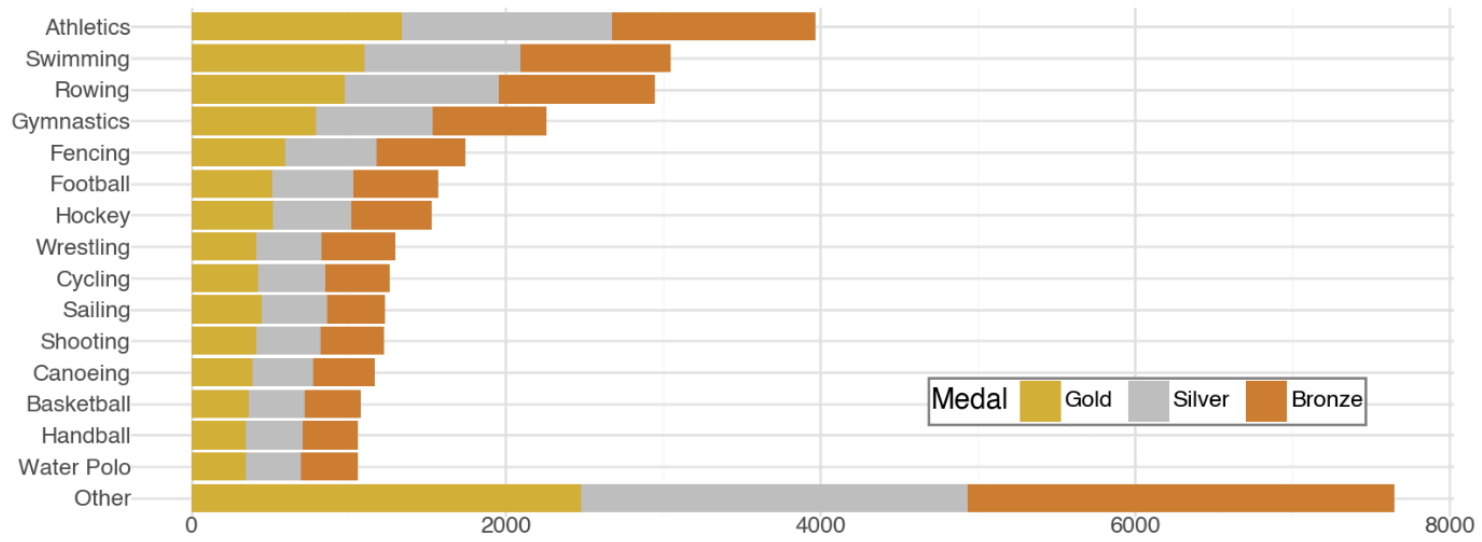
Last Close: 2023-10-27

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

PyData

Olympic Games | Summer Olympics | Winter Olympics

Medals by sport



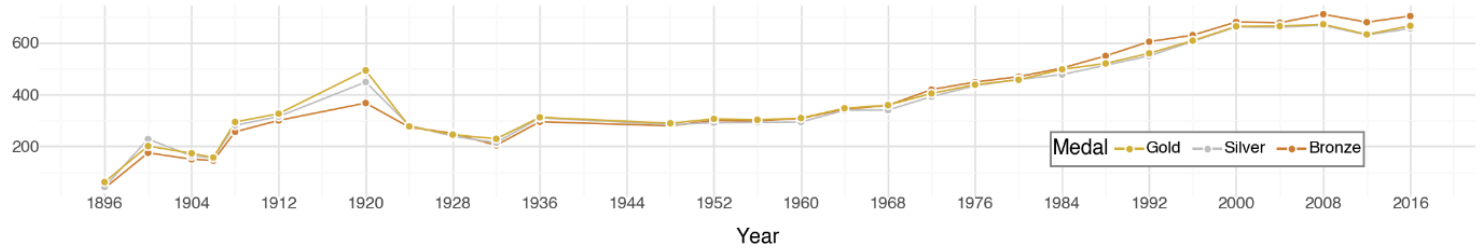
Most golds:
2363
United States

Most silvers:
1251
United States

Most bronzes:
1126
United States

Medals by year

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



Top 30 total medals | Bottom 30 total medals

Teams sorted in descending order of total medals.

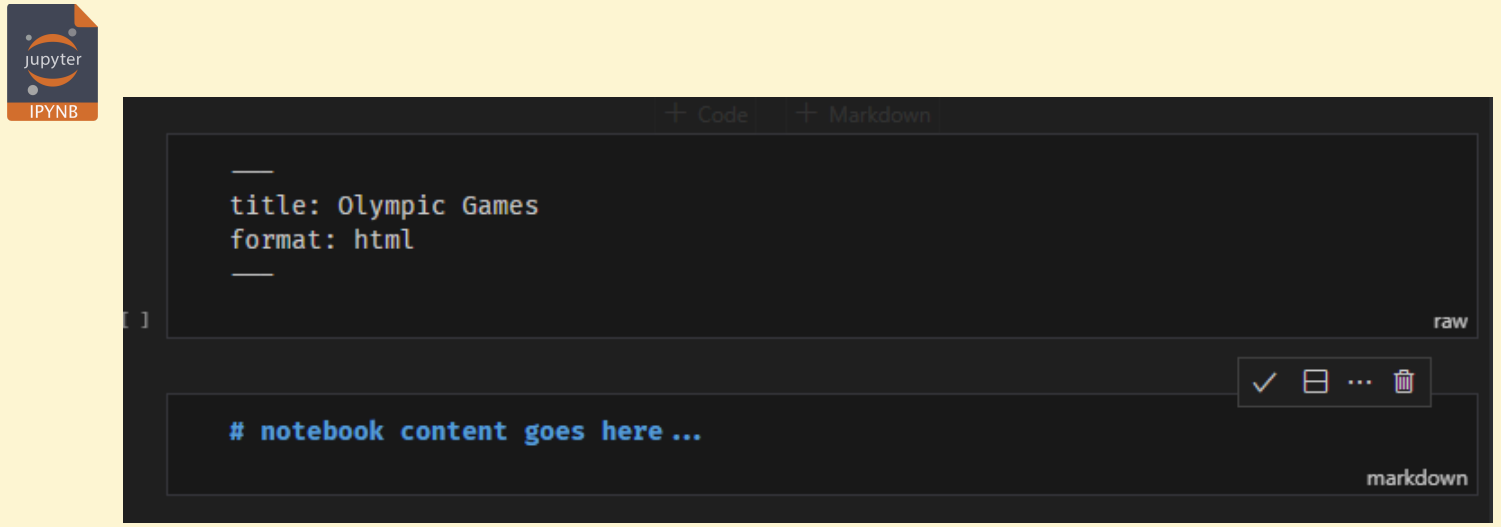
Team	Gold	Silver	Bronze
United States	2363.0	1251.0	1126.0
Soviet Union	830.0	635.0	596.0
Germany	568.0	515.0	614.0
Great Britain	501.0	594.0	563.0
France	425.0	490.0	515.0
Italy	492.0	469.0	425.0
Australia	338.0	450.0	506.0
Hungary	432.0	328.0	363.0
Sweden	326.0	356.0	332.0
China	334.0	317.0	258.0
Russia	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

How to build a Quarto Dashboard ?

Notebook → HTML

olympicdash-py.qmd

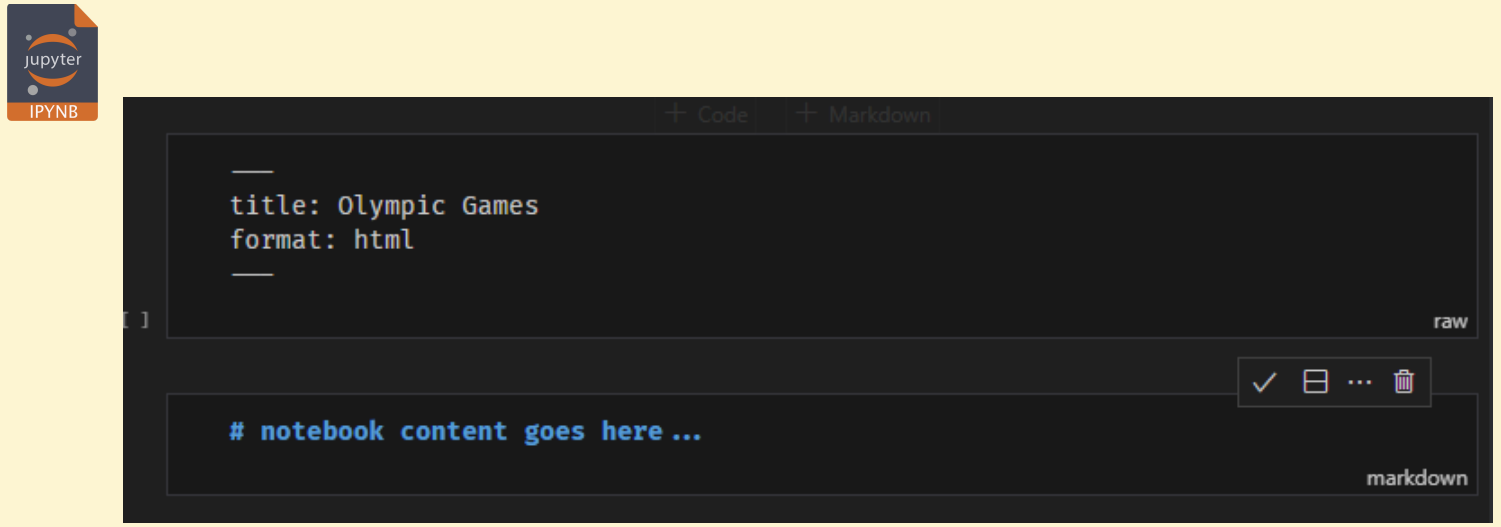
```
---
title: "Olympic Games"
format: html
---

# notebook content goes here...
```



Notebook → HTML

olympicdash-py.qmd

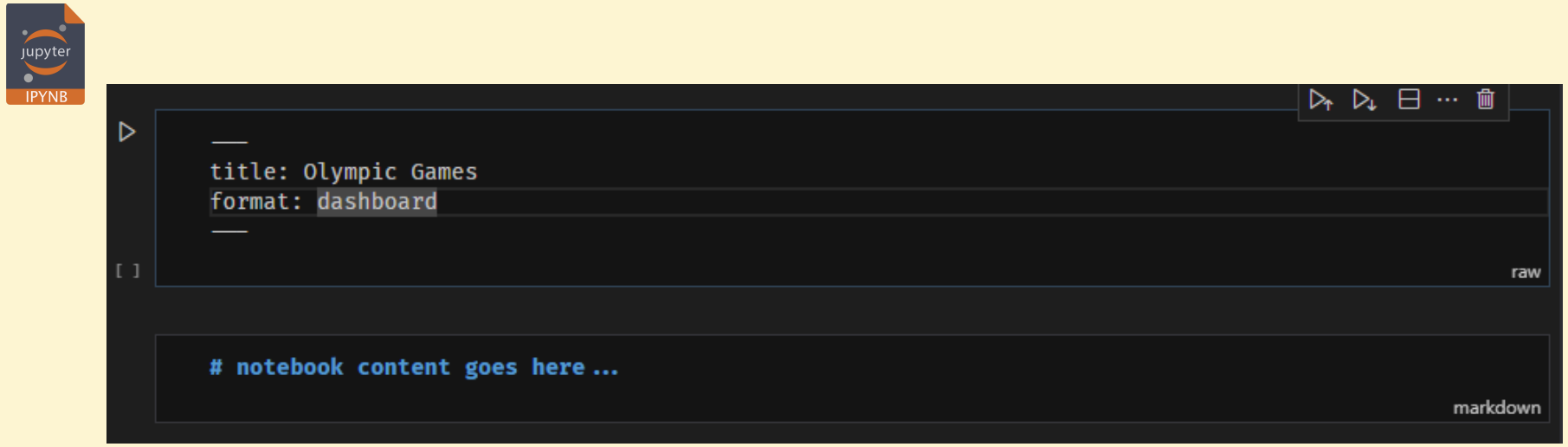
```yaml
---
title: "Olympic Games"
format: html
---

# notebook content goes here...
```



Notebook → Dashboard

olympicdash-py.qmd

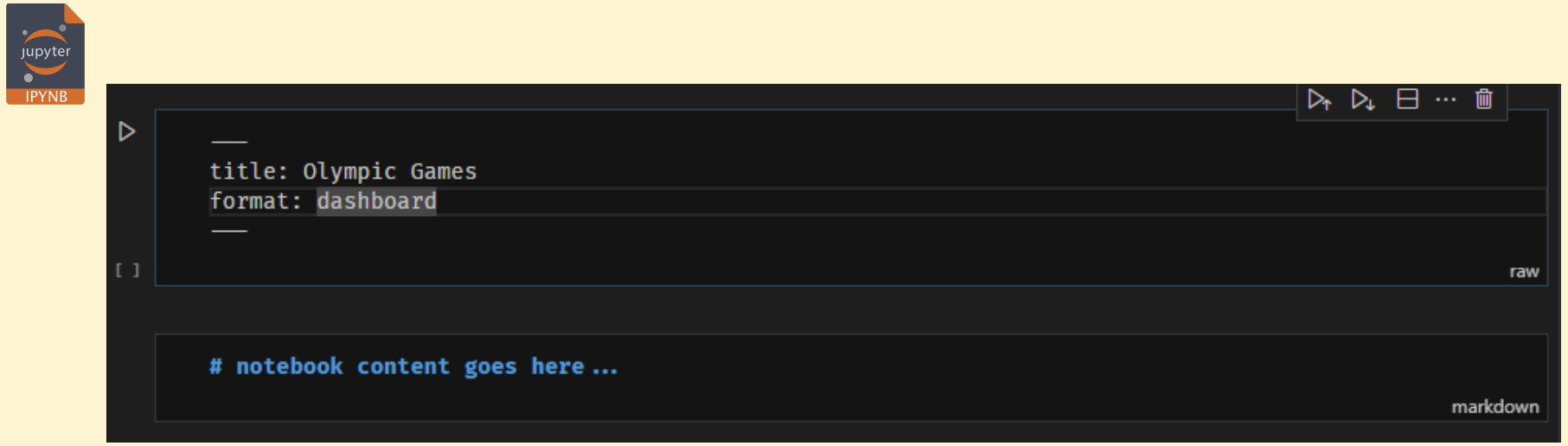
```yaml
---
title: "Olympic Games"
format: dashboard
---

# notebook content goes here...
```



Notebook → Dashboard

olympicdash-py.qmd

```yaml
---
title: "Olympic Games"
format: dashboard
---

# notebook content goes here...
```



Layout

Layout - Cards

Dashboards are composed of **cards**.

Layout - Rows and columns

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

Layout - Advanced

Sidebars, **tabsets**, 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 Tabsets**
   — Rows and columns using markdown heading (with optional attributes to control height, width, etc.). Sidebars for interactive inputs. Tabsets 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 Tabsets**
   — Rows and columns using markdown heading (with optional attributes to control height, width, etc.). Sidebars for interactive inputs. Tabsets 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


```yaml
---
title: "Olympic Games"
format:
  dashboard:
    nav-buttons: [github]
    github: https://github.com/posit-conf-2024/olympicdash
logo: images/olympics-logo.svg
logo-alt: "Olympics logo with multicolored circles."
---

# 🌞 Summer Olympics

# ❄️ Winter Olympics
```

Navigation Bar and Pages


```yaml
---
title: "Olympic Games"
format:
  dashboard:
    nav-buttons: [github]
    github: https://github.com/posit-conf-2024/olympicdash
logo: images/olympics-logo.svg
logo-alt: "Olympics logo with multicolored circles."
---

# 🌞 Summer Olympics

# ❄️ Winter Olympics
```

Navigation Bar and Pages


```yaml
---
title: "Olympic Games"
format:
  dashboard:
    nav-buttons: [github]
    github: https://github.com/posit-conf-2024/olympicdash
logo: images/olympics-logo.svg
logo-alt: "Olympics logo with multicolored circles."
---

# 🌞 Summer Olympics

# ❄️ Winter Olympics
```

Navigation Bar and Pages


```yaml
---
title: "Olympic Games"
format:
  dashboard:
    nav-buttons: [github]
    github: https://github.com/posit-conf-2024/olympicdash
logo: images/olympics-logo.svg
logo-alt: "Olympics logo with multicolored circles."
---

# 🌞 Summer Olympics

# ❄️ Winter Olympics
```

Navigation Bar and Pages


```yaml
---
title: "Olympic Games"
format:
  dashboard:
    nav-buttons: [github]
    github: https://github.com/posit-conf-2024/olympicdash
logo: images/olympics-logo.svg
logo-alt: "Olympics logo with multicolored circles."
---

# 🌞 Summer Olympics

# ❄️ Winter Olympics
```

Sidebars: Page Level

````
---
title: "Sidebar"
format: dashboard
---

# Page 1

## {.sidebar}

```{python}
```

## Column

```{python}
````
````

Sidebars: Page Level

````
format: dashboard
---

# Page 1

## {.sidebar}

```{python}
```

## Column

```{python}
```
````

Sidebars: Page Level

````
format: dashboard
---

# Page 1

## {.sidebar}

```{python}
```

## Column

```{python}
```
````

Sidebars: Global

````
---
title: "Global Sidebar"
format: dashboard
---

# {.sidebar}

Sidebar content (e.g. inputs)

# Plot

```{python}
```

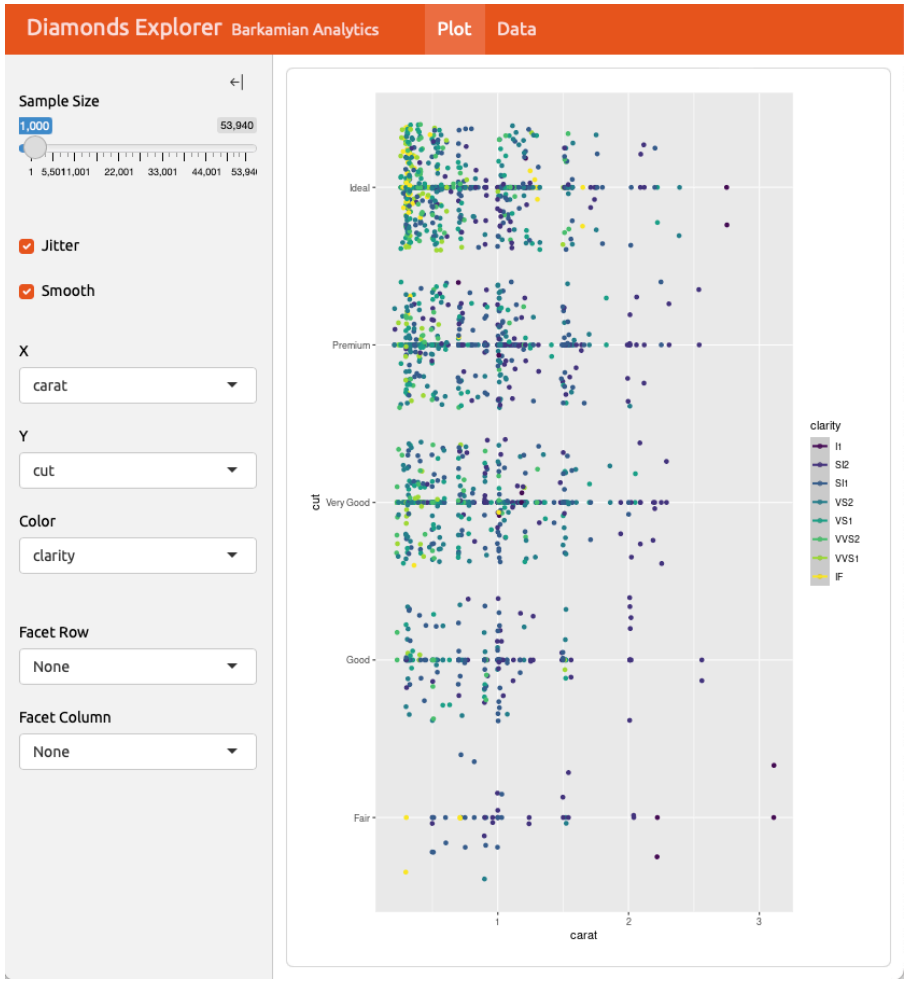
# Data
````

Sidebars: Global

````
---
title: "Global Sidebar"
format: dashboard
---

# {.sidebar}

Sidebar content (e.g. inputs)

# Plot

```{python}
```

# Data
````

Sidebars: Global

```
---
title: "Global Sidebar"
format: dashboard
---

# {.sidebar}

Sidebar content (e.g. inputs)

# Plot

```{python}
```

Data
```



Layout: Rows

````
---
title: "By Rows"
format: dashboard
---

## Row {height=70%}

```{python}
```

## Row {height=30%}

```{python}
```
````

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

Layout: Rows

````
---
title: "By Rows"
format: dashboard
---

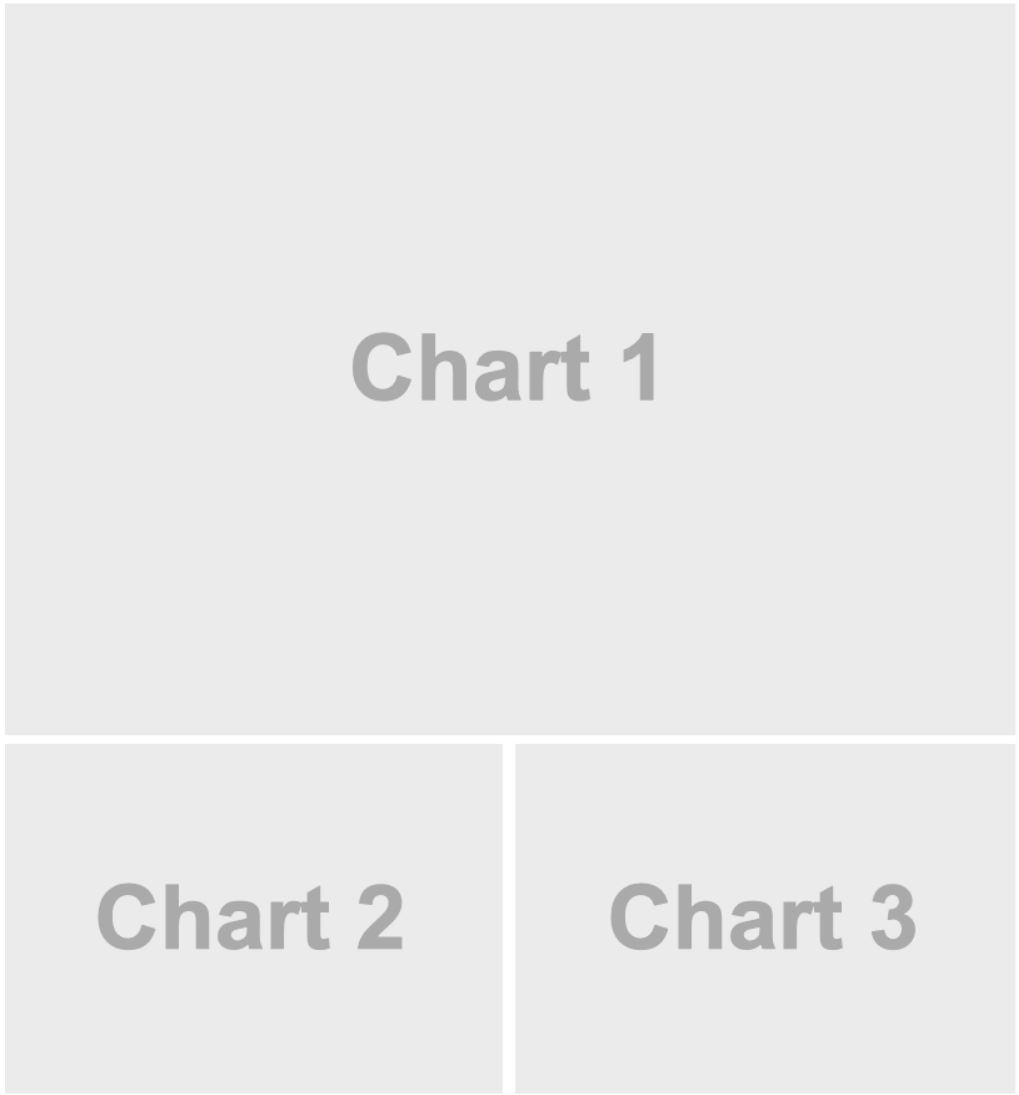
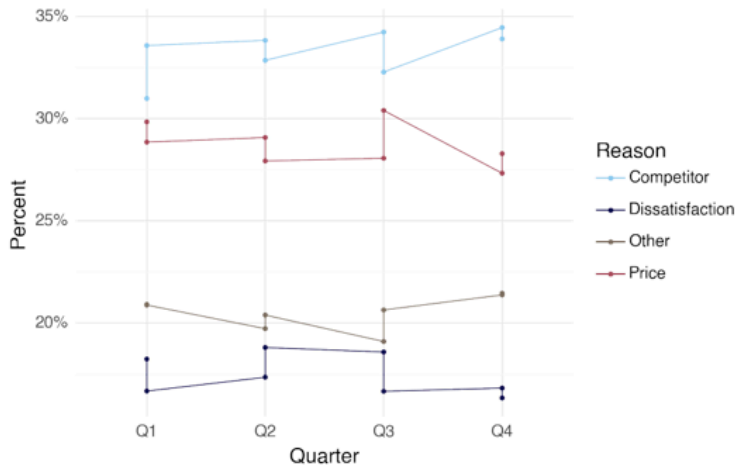
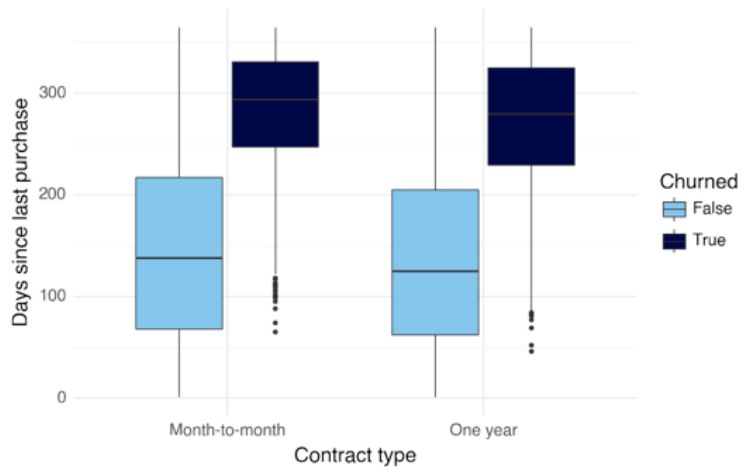
## Row {height=70%}

```{python}
```

## Row {height=30%}

```{python}
```

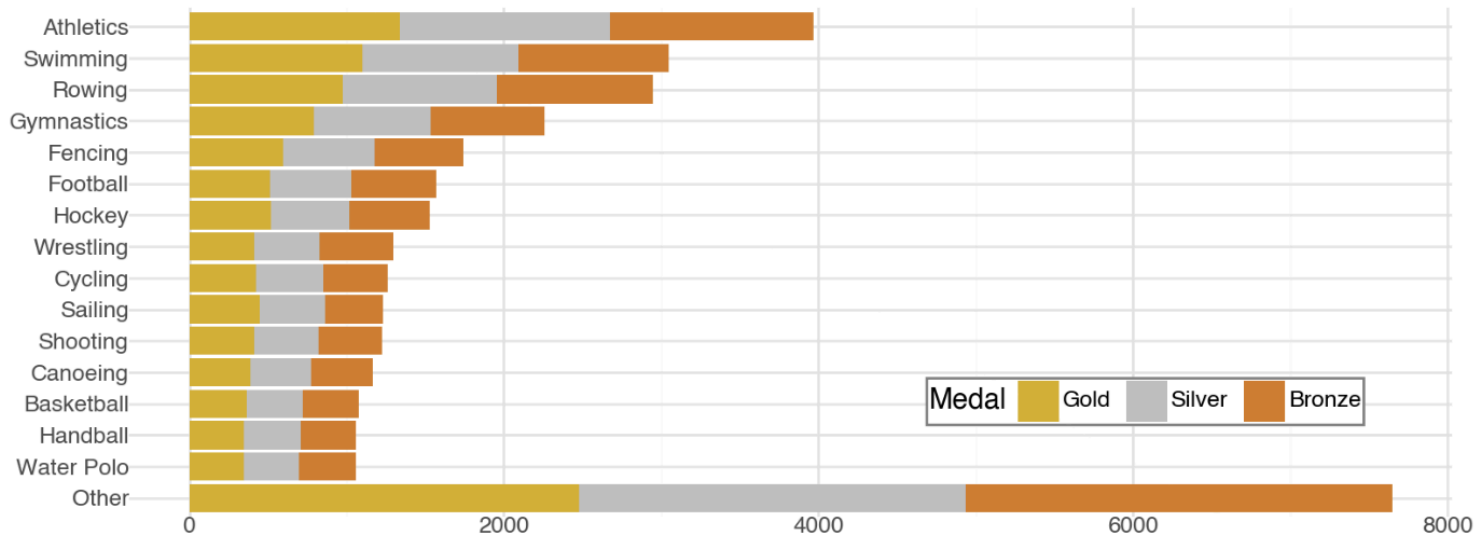
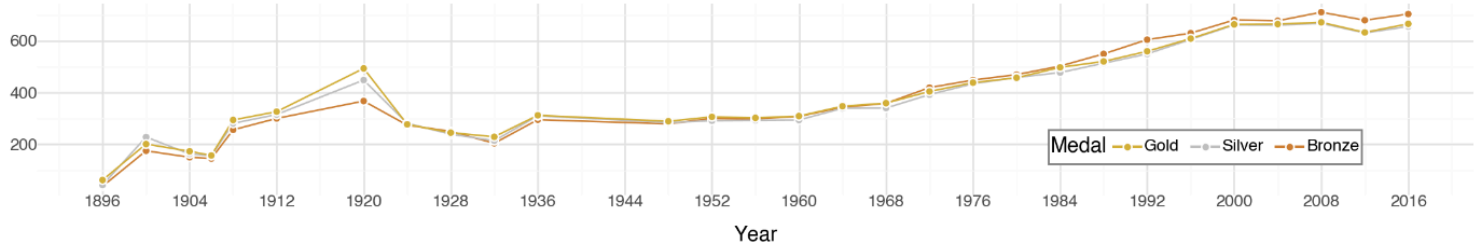
```{python}
````

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

Layout: Rows

```
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.

DEMOCO Customer Churn | Churn (Standard) | Data

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

````
---
title: "By Columns"
format:
  dashboard:
    orientation: columns
---

## Column {width=60%}

```{python}
```

## Column {width=40%}

```{python}
````
````

Layout: Columns

```
---
title: "By Columns"
format:
  dashboard:
    orientation: columns
---

## Column {width=60%}

```{python}
```

Column {width=40%}

```{python}
```
```

Layout: Columns

```
format:
  dashboard:
    orientation: columns
---

## Column {width=60%}

```{python}
```

Column {width=40%}

```{python}
```

```{python}
```
```

Layout: Columns

```
    orientation: columns
---

## Column {width=60%}

```{python}
```

Column {width=40%}

```{python}
```

```{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 ```
```

Olympic Games | Summer Olympics | Winter Olympics

Medals by sport



Most golds:
2363
United States

Most silvers:
1251
United States

Most bronzes:
1126
United States

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.



PyData

Cards

Automatically created from cell's output or Markdown content

````
## Column - Medals by sport and year {width=65%}

### Row - Medals by sport {height=60%}

```{python}
#| label: summer-medals-by-sport
#| title: Medals by sport

# Lump the sport column to top 15 categories, grouping others as Other
top_15_sports = summer_olympics["sport"].value_counts().nlargest(15).index
summer_olympics["sport"] = summer_olympics["sport"].apply(lambda x: x if x in top_15_sports

# Convert the sport column to a categorical type with order based on frequency, and reverse
summer_olympics["sport"] = pd.Categorical(summer_olympics["sport"], categories = summer_oly

````
````

Cards

Automatically created from cell's output or Markdown content

````
## Column - Medals by sport and year {width=65%}

### Row - Medals by sport {height=60%}

```{python}
#| label: summer-medals-by-sport
#| title: Medals by sport

# Lump the sport column to top 15 categories, grouping others as Other
top_15_sports = summer_olympics["sport"].value_counts().nlargest(15).index
summer_olympics["sport"] = summer_olympics["sport"].apply(lambda x: x if x in top_15_sports

# Convert the sport column to a categorical type with order based on frequency, and reverse
summer_olympics["sport"] = pd.Categorical(summer_olympics["sport"], categories = summer_oly

````
````

Cards

Automatically created from cell's output or Markdown content

````
## Column - Medals by sport and year {width=65%}

### Row - Medals by sport {height=60%}

```{python}
#| label: summer-medals-by-sport
#| title: Medals by sport

# Lump the sport column to top 15 categories, grouping others as Other
top_15_sports = summer_olympics["sport"].value_counts().nlargest(15).index
summer_olympics["sport"] = summer_olympics["sport"].apply(lambda x: x if x in top_15_sports

# Convert the sport column to a categorical type with order based on frequency, and reverse
summer_olympics["sport"] = pd.Categorical(summer_olympics["sport"], categories = summer_oly

````
````

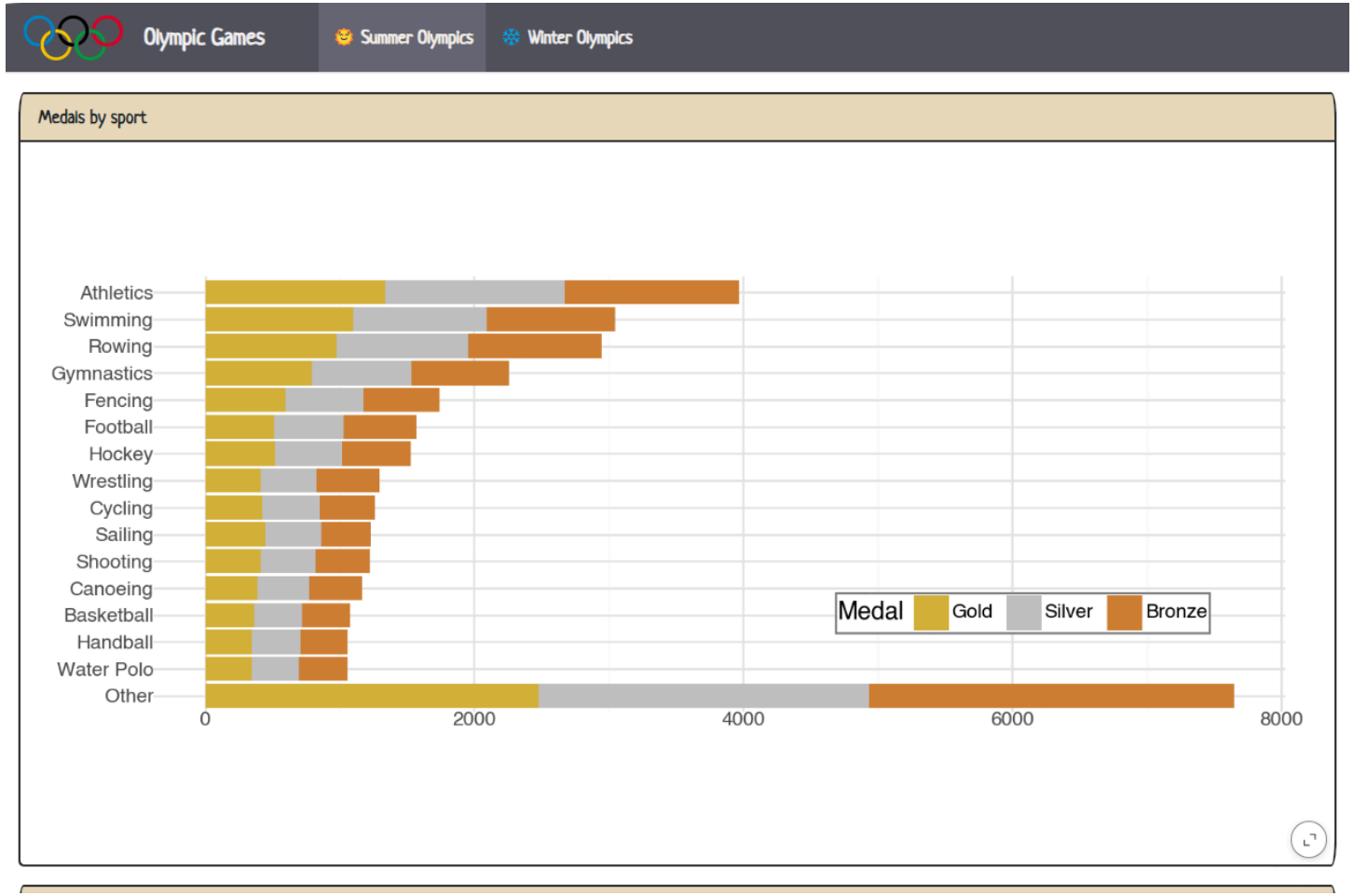
Cards

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

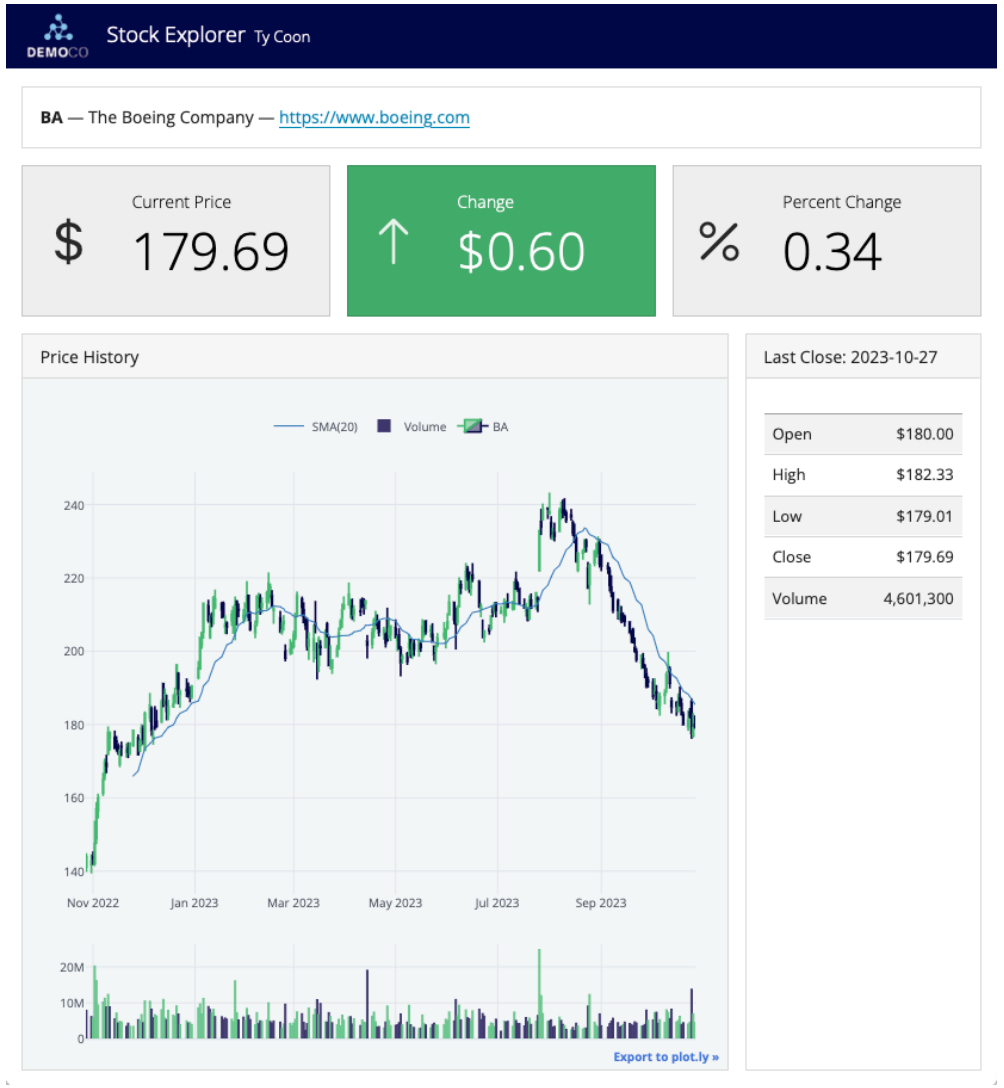


Value Boxes

```
## Row

```{python}
#| component: valuebox
#| title: "Current Price"
dict(icon = "currency-dollar",
     color = "secondary",
     value = get_price(data))
```

```{python}
#| component: valuebox
#| title: "Change"
change = get_change(data)
dict(value = change['amount'],
```

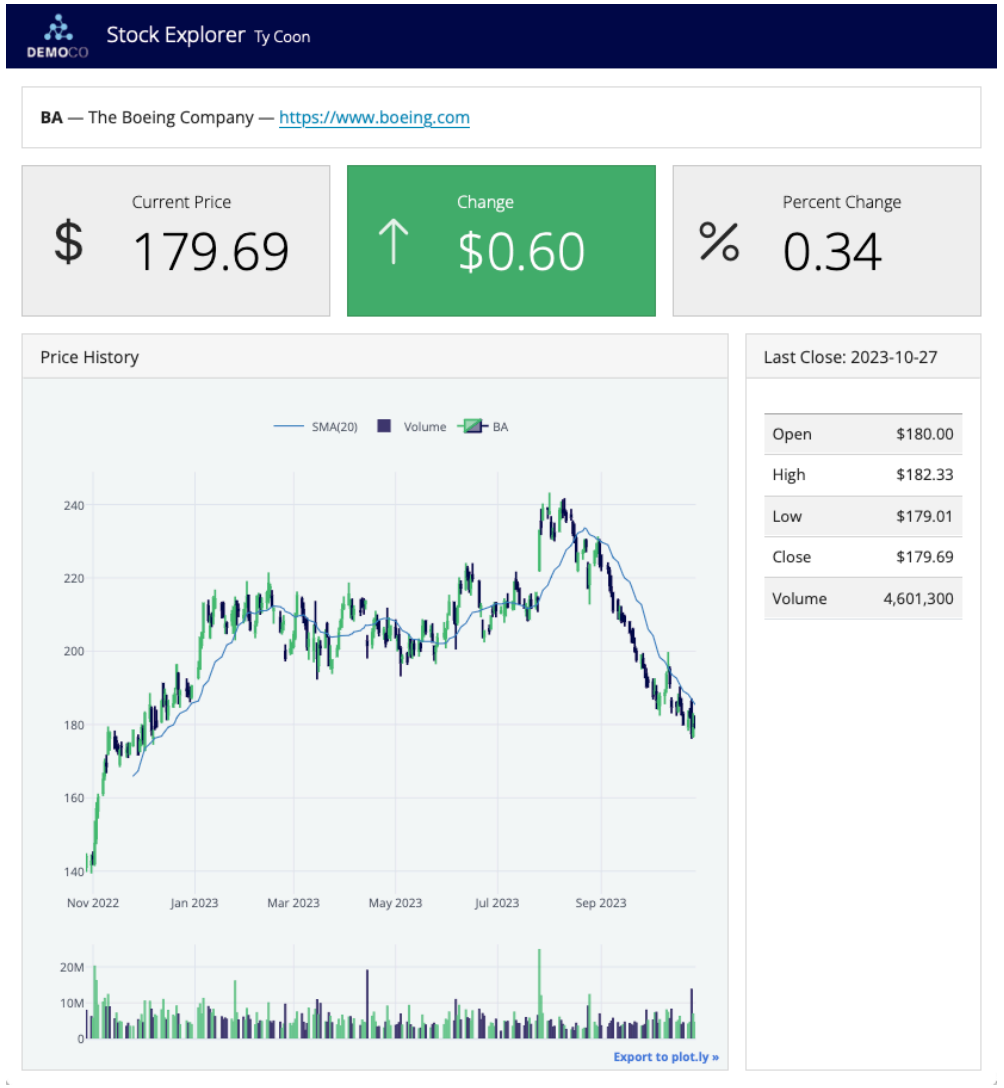

```

Value Boxes

```
## Row

```{python}
#| component: valuebox
#| title: "Current Price"
dict(icon = "currency-dollar",
     color = "secondary",
     value = get_price(data))
```

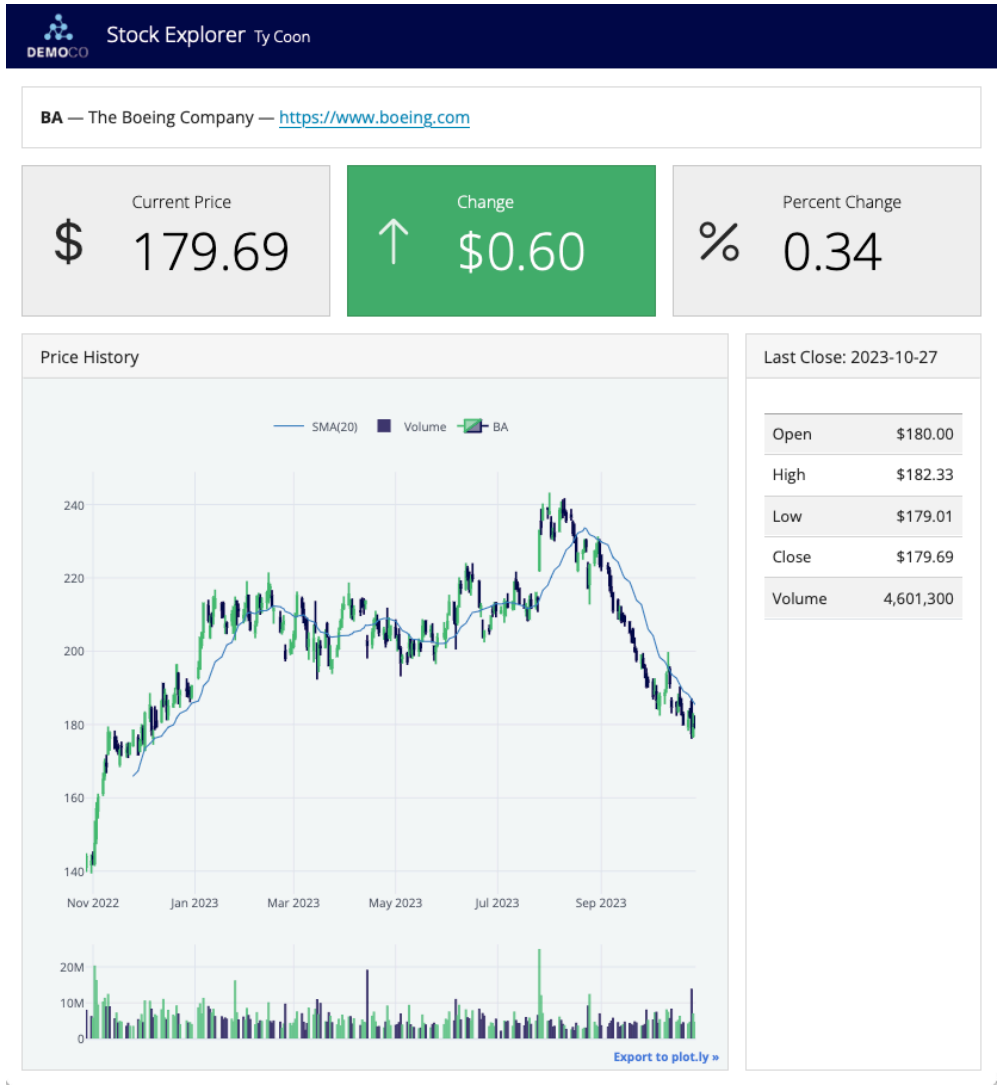
```{python}
#| component: valuebox
#| title: "Change"
change = get_change(data)
dict(value = change['amount'],
```


```

Value Boxes

````
#| component: valuebox
#| title: "Current Price"
dict(icon = "currency-dollar",
     color = "secondary",
     value = get_price(data))
```

```{python}
#| component: valuebox
#| title: "Change"
change = get_change(data)
dict(value = change['amount'],
     icon = change['icon'],
     color = change['color'])
```
````

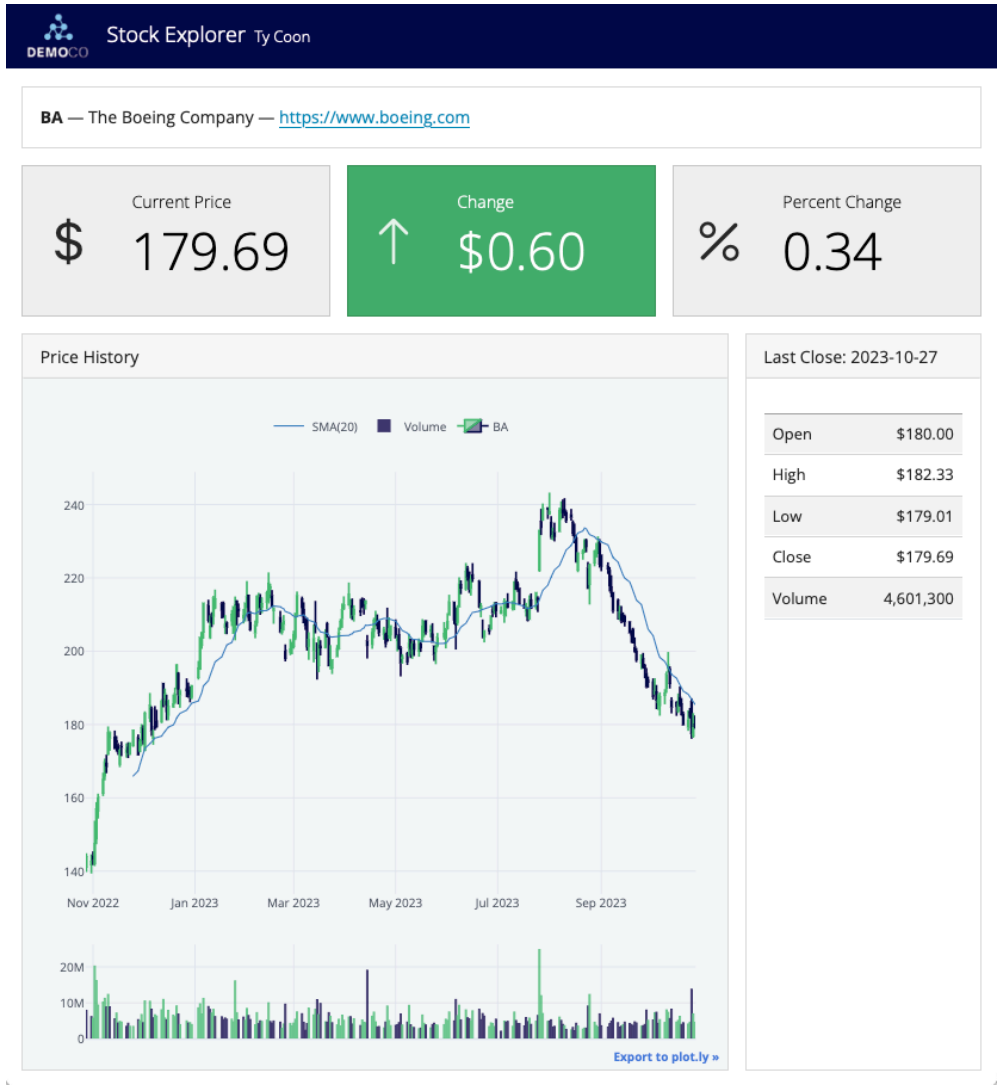

````

Value Boxes

```
## Row

```{python}
#| component: valuebox
#| title: "Current Price"
dict(icon = "currency-dollar",
     color = "secondary",
     value = get_price(data))
```

```{python}
#| component: valuebox
#| title: "Change"
change = get_change(data)
dict(value = change['amount'],
```


```

Value Boxes

```
## Column - Medals by country {width=35%}

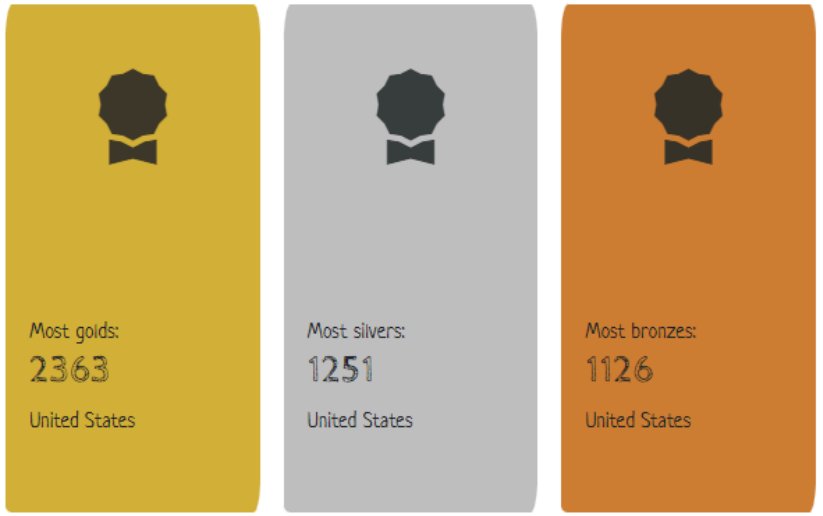
### Row - Value boxes {height=30%}

::: {.valuebox icon="award-fill" color="#d4af37"}
Most golds:

`{python} str(count_most_gold_medals)`

`{python} most_gold_medals`
:::

::: {.valuebox icon="award-fill" color="#c0c0c0"}
Most silvers:

```



Value Boxes

```
## Column - Medals by country {width=35%}

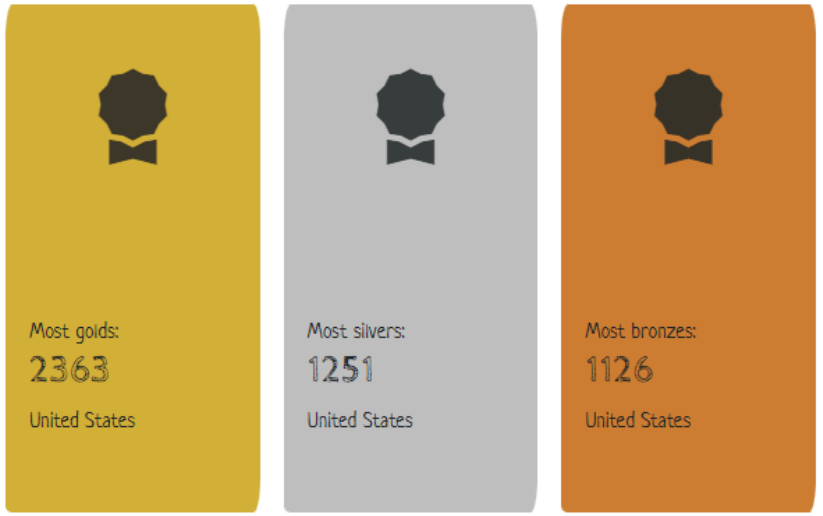
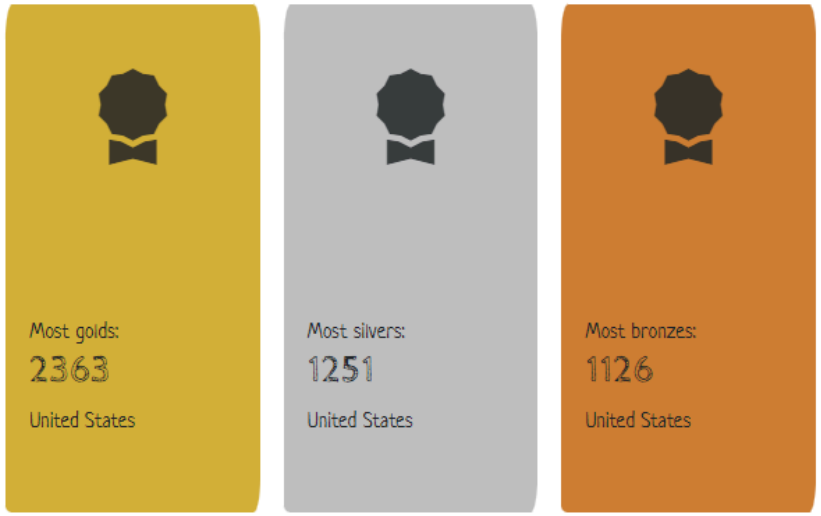
### Row - Value boxes {height=30%}

::: {.valuebox icon="award-fill" color="#d4af37"}
Most golds:

`{python} str(count_most_gold_medals)`

`{python} most_gold_medals`
:::

::: {.valuebox icon="award-fill" color="#c0c0c0"}
Most silvers:

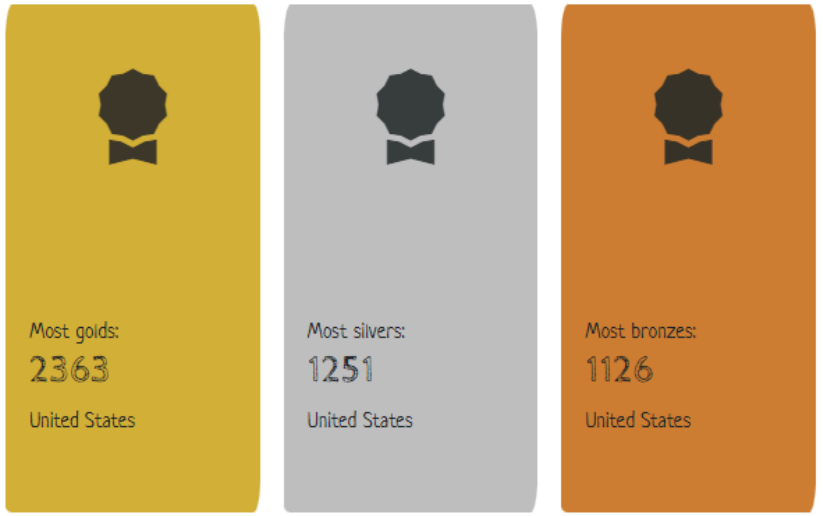
```



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

```
::: {.valuebox icon="award-fill" color="#c0c0c0"}
Most silvers:

`{python} str(count_most_silver_medals)`

`{python} most_silver_medals`
:::

::: {.valuebox icon="award-fill" color="#cd7f32"}
Most bronzes:

`{python} str(count_most_bronze_medals)`

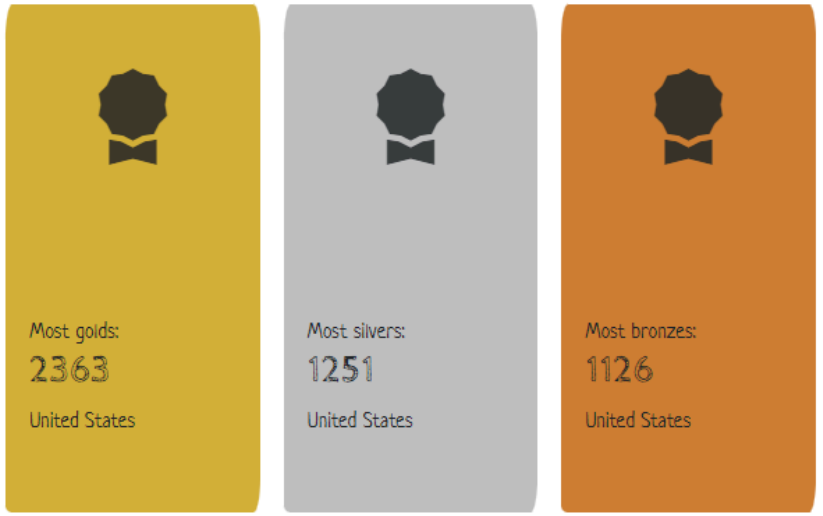
`{python} most_bronze_medals`
:::
```



Value Boxes

```
::: {.valuebox icon="award-fill" color="#c0c0c0"}
Most silvers:

`{python} str(count_most_silver_medals)`

`{python} most_silver_medals`
:::

::: {.valuebox icon="award-fill" color="#cd7f32"}
Most bronzes:

`{python} str(count_most_bronze_medals)`

`{python} most_bronze_medals`
:::
```



`{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.

Tabsets

````
---
title: "Palmer Penguins"
format: dashboard
---

## Row

```{python}
```

## Row {.tabset}

```{python}
#| title: Chart 2
```
````

Tabsets

````
---

## Row

```{python}
```

## Row {.tabset}

```{python}
#| title: Chart 2
```

```{python}
#| title: Chart 3
```
````

Tabsets

````
## Row

```{python}
```

## Row {.tabset}

```{python}
#| title: Chart 2
```

```{python}
#| title: Chart 3
```
````

Tabsets

````
## Row

```{python}
```

## Row {.tabset}

```{python}
#| title: Chart 2
```

```{python}
#| 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 ```
```

Styling a Dashboard

```yaml
---
title: "Olympic Games"
format:
  dashboard:
    orientation: columns
    nav-buttons: [github]
    github: https://github.com/posit-conf-2024/olympicdash
    theme:
      - sketchy
      - style/olympicdash.scss
logo: images/olympics-logo.svg
logo-alt: "Olympics logo with multicolored circles."
---
```

Details at https://quarto.org/docs/dashboards/theming.html

Styling a Dashboard

```yaml
---
title: "Olympic Games"
format:
  dashboard:
    orientation: columns
    nav-buttons: [github]
    github: https://github.com/posit-conf-2024/olympicdash
    theme:
      - sketchy
      - style/olympicdash.scss
logo: images/olympics-logo.svg
logo-alt: "Olympics logo with multicolored circles."
---
```

Bootstrap & Bootswatch Themes:

Quarto Dashboard is based on Bootstrap

with +20 themes from Bootswatch available

Details at https://quarto.org/docs/dashboards/theming.html

PyData

Styling a Dashboard

```yaml
---
title: "Olympic Games"
format:
  dashboard:
    orientation: columns
    nav-buttons: [github]
    github: https://github.com/posit-conf-2024/olympicdash
    theme:
      - sketchy
      - style/olympicdash.scss
logo: images/olympics-logo.svg
logo-alt: "Olympics logo with multicolored circles."
---
```

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

Details at https://quarto.org/docs/dashboards/theming.html

Quarto Theme File

olympicdash.scss

```scss
/*-- scss:defaults --*/

// colors

$navbar-bg: #52515e;
$navbar-fg: #F0F0F0;

$link-color: #ae8b2d;
$hover-color: lighten($link-color, 40%);

/*-- scss:rules --*/
.card-header {
  background-color: #ae8b2d50;
}

```

Quarto Theme File

olympicdash.scss

```
/*-- scss:rules --*/
.card-header {
  background-color: #ae8b2d50;
}

.nav-item > a:hover {
  color: #ae8b2d;
}

.nav-link {
  color: #F0F0F0;
}

.tabset .nav-link {
  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