

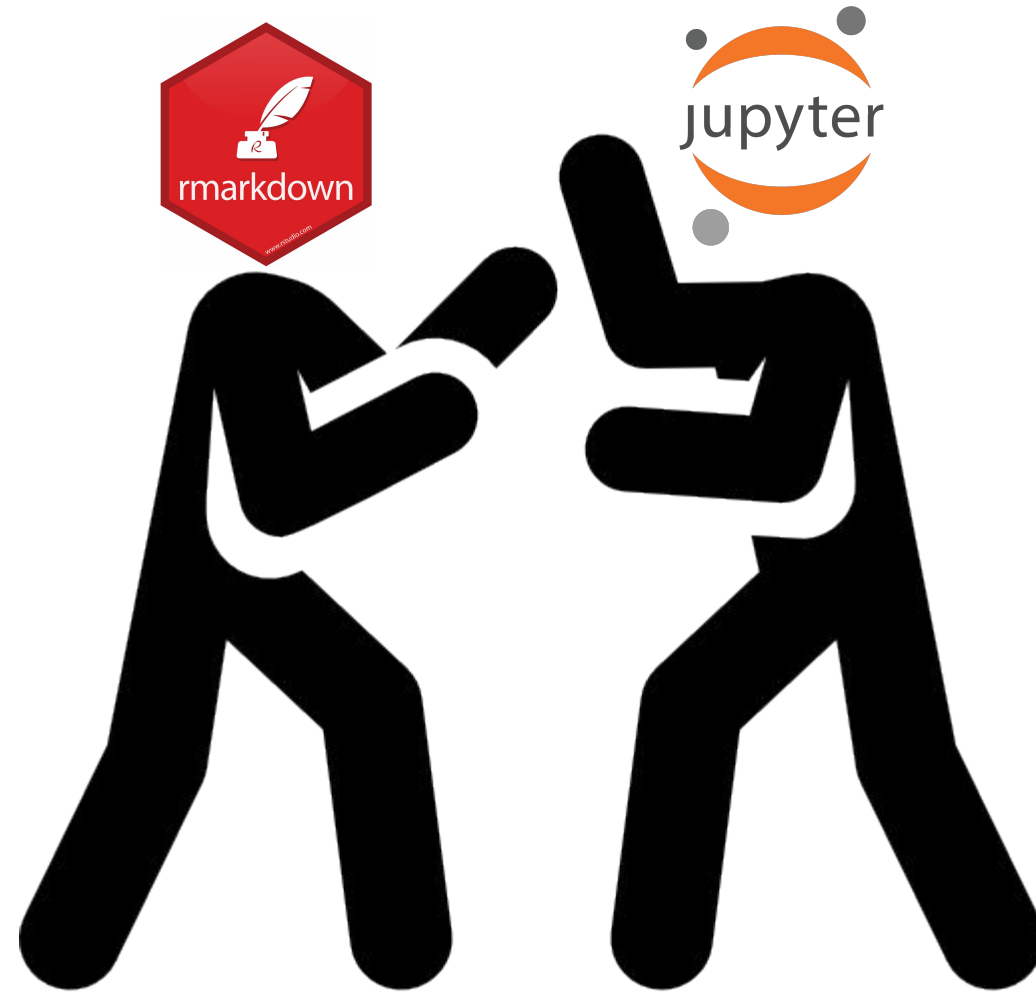
Alternative to Jupyter notebooks

Thomas Denecker

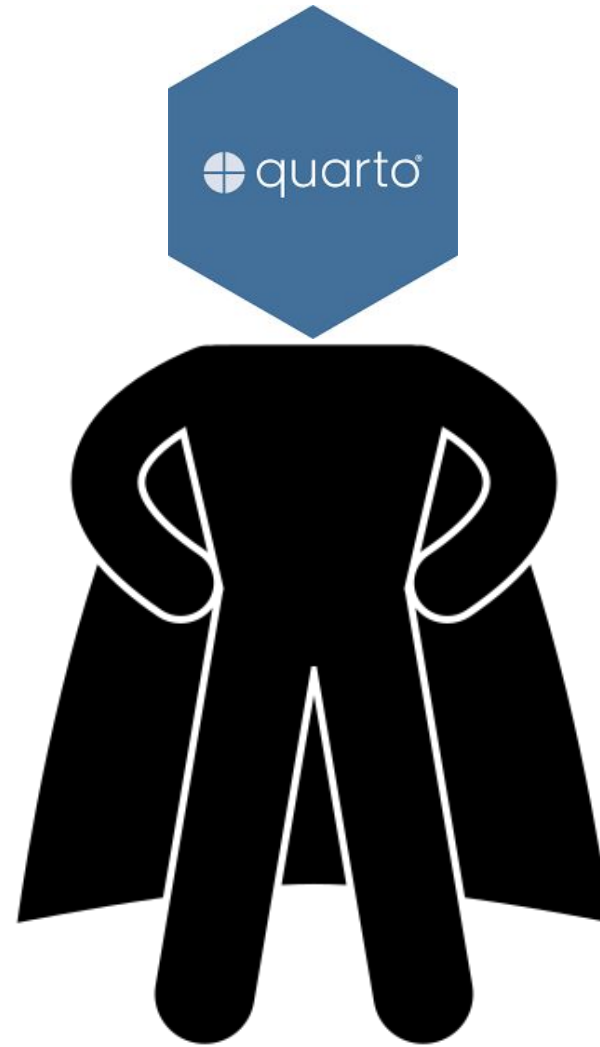
IFB, <https://orcid.org/0000-0003-1421-7641>

BiGEst





A replacement?





At the beginning, there was nothing. Then came Sweave.

Vol. 2/3, December 2002 28

Sweave, Part I: Mixing R and L^AT_EX

A short introduction to the Sweave file format and corresponding R functions

by Friedrich Leisch

This is the first article in a two part mini series on Sweave (Leisch, 2002), a tool that allows to embed the R code for complete data analyses in L^AT_EX documents. In this issue we will introduce the Sweave file format and R functions to process it, and demonstrate how to use Sweave as a reporting tool for literate statistical practice (Rossini, 2001). The companion article scheduled for the next issue of R News will concentrate on how to use files in Sweave format to write primers or manuals for R packages that can be automatically checked for syntax errors in the code or inconsistencies between examples and implementation.

The traditional way of writing a report as part of a statistical data analysis project uses two separate steps: First, the data are analyzed, and afterwards the results of the analysis (numbers, graphs, ...) are used as the basis for a written report. In larger projects the two steps may be repeated alternately, but the basic procedure remains the same. R supports this in a number of ways: graphs can be saved as PDF, EPS, or WMF which in turn can be included in L^AT_EX or Word documents. L^AT_EX tables can be created by specifying the columns and row separators in `write.table()` or using the package `xtable`. The basic paradigm is to write the report around the results of the analysis.

The purpose of Sweave is to create dynamic reports, which can be updated automatically if data or analysis change. Instead of inserting a prefabricated graph or table into the report, the master document contains the R code necessary to obtain it. When run through R, all data analysis output (tables, graphs, ...) is created on the fly and inserted into a final L^AT_EX document. The report can be automatically updated if data or analysis change, which allows for truly reproducible research.

A small example

Sweave source files are regular noweb files (Ramsey, 1998) with some additional syntax that allows control over the final output. Noweb is a simple literate programming tool which allows to combine program source code and the corresponding documentation into a single file. These consist of a sequence of code and documentation segments, called *chunks*. Different command line programs are used to extract the code ("`literate`") or typeset documentation to-

gether with the code ("`noweb`").

A small Sweave file is shown in Figure 1, which contains four code chunks embedded in a simple L^AT_EX document. "`<<...>>`" at the beginning of a line marks the start of a code chunk, while a "`%`" at the beginning of a line marks the start of a documentation chunk. Sweave translates this into a regular L^AT_EX document, which in turn can be compiled by latex to Figure 2.

The code chunks

The main work of Sweave is done on the code chunks. All code chunks are evaluated by R in the order they appear in the document¹. Within the double angle brackets we can specify options that control how the code and the corresponding output are rendered in the final document. The first code chunk (lines 5-8 in Figure 1) declares that neither the R code (`echo=false`) nor output (`results=hide`) shall be included. The purpose of this chunk is to initialize R by loading packages and data, we want to hide these technical details from the reader.

Let us skip the text in lines 10-19 for the moment and go directly to the next code chunk in lines 20-22. It uses the default settings for all options (nothing is specified within the double angle brackets): both input and output are shown to the user (see Figure 2), the chunk is rendered such that it emulates the R console when the code is typed at the prompt. All input and output are automatically encapsulated in verbatim-like environments.

The next code chunk can be found at lines 30-31. It uses the package `xtable` to pretty-print the coefficient matrix of the linear regression model. By specifying `results=tex` we tell Sweave that the output of this code chunk is regular T_EX code and hence needs no protection by a verbatim environment.

The last code chunk in lines 36-38 is marked as a figure chunk (`fig=true`) such that Sweave creates EPS and PDF files corresponding to the plot created by the commands in the chunk. Furthermore, an `\includegraphics()` statement is inserted into the L^AT_EX file. Options width and height are passed to R's graphics devices and determine the size of the figure in the EPS and PDF files.

In line 28 we use `\SweaveOpts{echo=false}` to modify the default for option `echo` to the value of `false` for all code chunks following, hence the code for the last two chunks is not shown in Figure 2. It has exactly the same effect as if we had included `echo=false` within the double angle brackets of the two chunks.

¹There are ways to suppress evaluation or re-use chunks, which is beyond the scope of this article.

```

\documentclass[a4paper]{article}

\begin{document}

5 <<echo=false,results=hide>>=
library(lattice)
library(xtable)
data(cats, package="MASS")
@

10 \section*{The Cats Data}

Consider the \texttt{cats} regression example from Venables \& Ripley
(1997). The data frame contains measurements of heart and body weight
of \Sexpr{nrow(cats)} cats (\Sexpr{sum(cats$Sex=="F")} female,
\Sexpr{sum(cats$Sex=="M")} male).

A linear regression model of heart weight by sex and gender can be
fitted in R using the command
<<>=
20 lm = lm(Hwt~Bwt*Sex, data=cats)
lm
@
Tests for significance of the coefficients are shown in
Table~\ref{tab:coef}, a scatter plot including the regression lines is
shown in Figure~\ref{fig:cats}.

\SweaveOpts{echo=false}

30 <<results=tex>>=
xtable(lm, caption="Linear regression model for cats data.", label="tab:coef")
@

\begin{figure}
35 \centering
<<fig=true,width=12,height=6>>=
lset(col.whitebg())
print(xyplot(Hwt~Bwt|Sex, data=cats, type=c("p", "r")))
@
\caption{The cats data from package MASS.}
\label{fig:cats}
\end{figure}

\end{document}
    
```

Figure 1: A minimal Sweave file: `example.Snw`.

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	2.9813	1.8428	1.62	0.1080
Bwt	2.6364	0.7759	3.40	0.0009
SexM	-4.1654	2.0618	-2.02	0.0453
Bwt:SexM	1.6763	0.8373	2.00	0.0472

Table 1: Linear regression model for cats data.

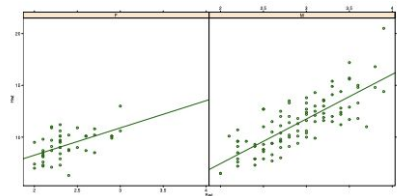


Figure 1: The cats data from package MASS.

The Cats Data

Consider the `cats` regression example from Venables & Ripley (1997). The data frame contains measurements of heart and body weight of 144 cats (47 female, 97 male).

A linear regression model of heart weight by sex and gender can be fitted in R using the command

```

> lm1 = lm(Hwt ~ Bwt * Sex, data = cats)
> lm1
    
```

```

Call:
lm(formula = Hwt ~ Bwt * Sex, data = cats)
    
```

Coefficients:

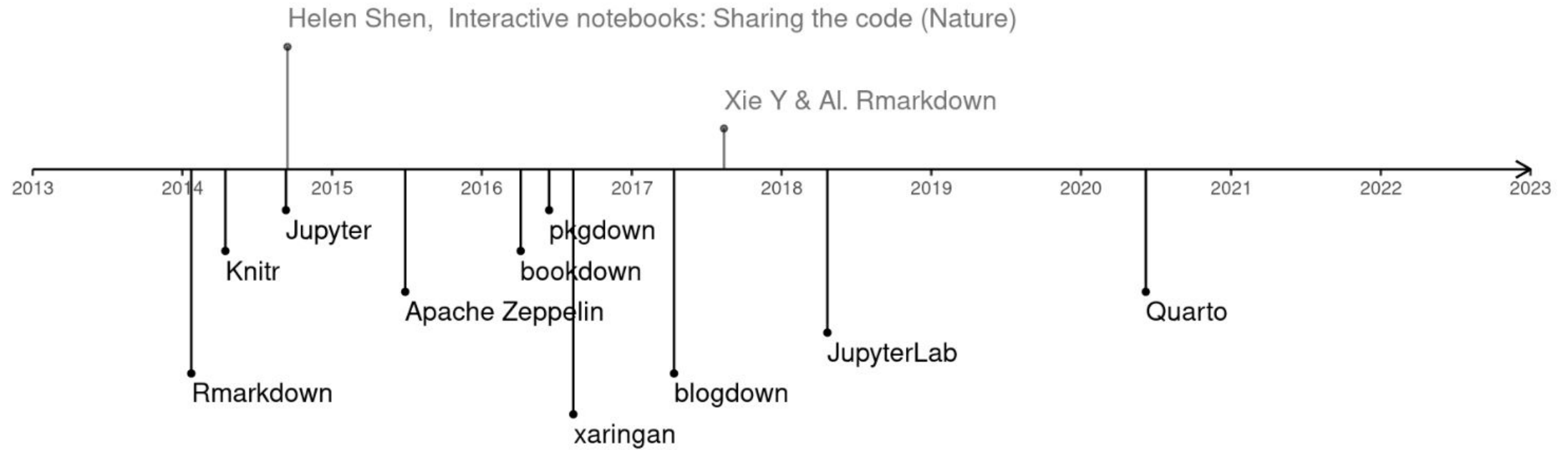
(Intercept)	Bwt	SexM	Bwt:SexM
2.981	2.636	-4.165	1.676

Tests for significance of the coefficients are shown in Table 1, a scatter plot including the regression lines is shown in Figure 1.

Figure 2: The final document is created by running `latex` on the intermediate file `example.tex` created by Sweave ("`example.Snw`").

And people saw that the path would be long...

Appearance of packages allowing the creation of notebooks

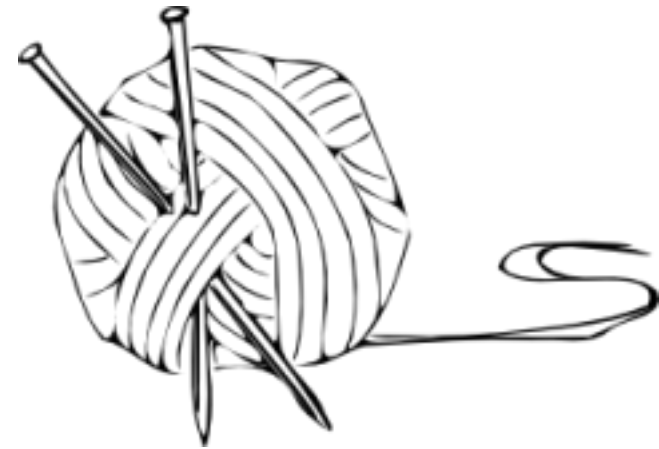


<https://camembr.quarto.pub/hello-quarto/#/les-packages>



”The knitR package was designed to be a transparent engine for dynamic report generation with R, solve some long-standing problems in Sweave, and combine features in other add-on packages into one package”

<https://yihui.org/knitr/>





”When you run render, R Markdown feeds the .Rmd file to knitr, which executes all of the code chunks and creates a new markdown (.md) document which includes the code and its output.

The markdown file generated by knitR is then processed by pandoc which is responsible for creating the finished format.”

<https://rmarkdown.rstudio.com>



```
1 ---
2 title: "Viridis Demo"
3 output: html_document
4 ---
5
6 ```{r include = FALSE}
7 library(viridis)
8 ```
9
10 The code below demonstrates two color palettes in the
11 [viridis](https://github.com/sjmgarnier/viridis) package. Each
12 plot displays a contour map of the Maunga Whau volcano in
13 Auckland, New Zealand.
14
15 ## Viridis colors
16
17 ```{r}
18 image(volcano, col = viridis(200))
19 ```
20
21 ## Magma colors
22
23 ```{r}
24 image(volcano, col = viridis(200, option = "A"))
25 ```
26
27
```



The screenshot shows the RStudio interface with a file named "1-example.Rmd". The code editor contains the following R code:

```
12- ## Viridis colors
13-
14- ```{r}
15- image(volcano, col = viridis(200))
16- ```
```

The plot area displays a heatmap of the "volcano" dataset, colored using the "viridis" color palette. The x and y axes both range from 0.0 to 1.0. The plot shows a central bright yellow-green region that transitions to dark purple towards the edges.

```
17-
18- ## Magma colors
19-
20- ```{r}
21- image(volcano, col = viridis(200, option = "A"))
22- ```
```

The bottom of the window shows the "Console" area, which is currently empty. The status bar at the bottom indicates "1:1 Viridis Demo" and "R Markdown".



Markdown Basics

Output Formats

Notebooks

Slide Presentations

Dashboards

Websites

Interactive Documents

Cheatsheets

file below, which is available [here](#) on RStudio Cloud.

The screenshot shows the RStudio interface. The left pane displays the R code for a presentation slide. The right pane shows the rendered HTML output, which includes the title 'PLASMA COLORS' and a contour plot of the Maunga Whau volcano using the Plasma color palette.

```
1 ---
2 title: "Viridis Presentation"
3 output:
4   revealjs::revealjs_presentation:
5     theme: league
6 ---
7
8 ```{r include = FALSE}
9 knitr::opts_chunk$set(echo = FALSE)
10 library(viridis)
11 ```
12
13 The [viridis](https://github.com/sjmgarnier/viridis)
14 package contains four color palettes, revealed in the
15 plots that follow.
16
17 >- Viridis
18 >- Magma
19 >- Inferno
20 >- Plasma
21
22 Each plot displays a contour map of the Maunga Whau
23 volcano in Auckland, New Zealand.
24
25 ## Viridis colors
26
27 ```{r}
28 image(volcano, col = viridis(200))
29 ```
```

PLASMA COLORS

The plot shows a contour map of the Maunga Whau volcano. The x and y axes both range from 0.0 to 1.0. The plot uses the Plasma color palette, with colors ranging from dark purple (low values) to bright yellow (high values). The volcano's shape is clearly visible as a bright yellow/orange area in the center-left of the plot.

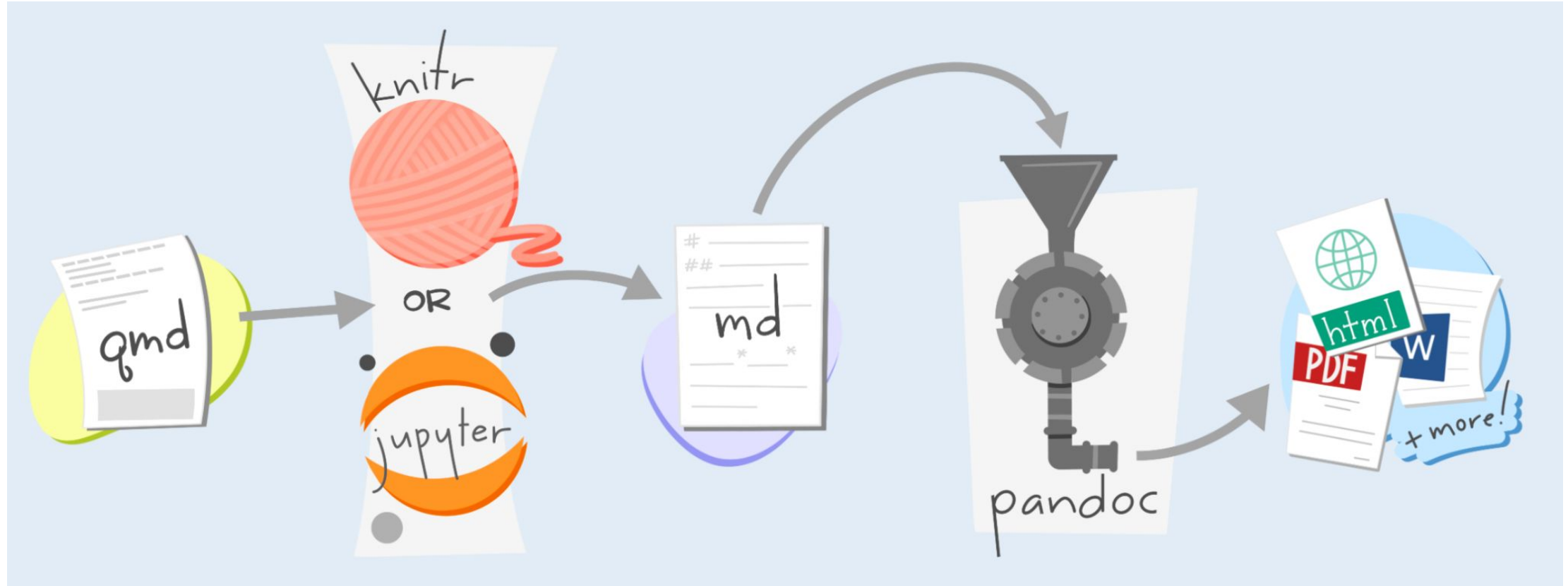




Quarto is an open-source scientific and technical publishing system where authors :

- Can use Jupyter notebooks or with plain text markdown in your favorite editor.
- Create dynamic content with Python, R, Julia, and Observable.
- Publish reproducible, production quality articles, presentations, websites, blogs, and books in HTML, PDF, MS Word, ePub, and more.
- Share results in a lot of publishing systems like GitHub.





With R

```

---
title: "ggplot2 demo"
author: "Norah Jones"
date: "5/22/2021"
format:
  html:
    fig-width: 8
    fig-height: 4
    code-fold: true
---

## Air Quality

@fig-airquality further explores the impact of
temperature on ozone level.

```{r}
#| label: fig-airquality
#| fig-cap: "Temperature and ozone level."
#| warning: false

library(ggplot2)

ggplot(airquality, aes(Temp, Ozone)) +
 geom_point() +
 geom_smooth(method = "loess"
)
```

```

ggplot2 demo

Norah Jones
May 22nd, 2021

Air Quality

[Figure 1](#) further explores the impact of temperature on ozone level.

► Code

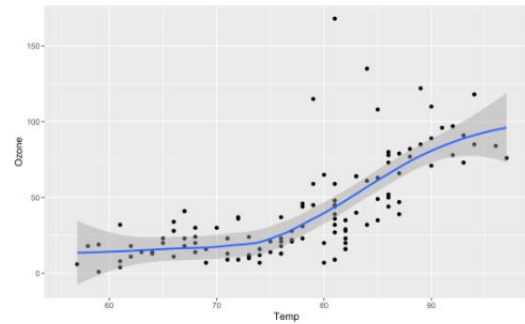


Figure 1: Temperature and ozone level.

With Jupyter

```

cell-options.ipynb
Python 3 (ipykernel)

Palmer Penguins

---
author: Norah Jones
format:
  html:
    code-tools: true
    code-fold: true
---

[3]: #| echo: false
import pandas as pd
df = pd.read_csv("palmer-penguins.csv")

Exploring the Data

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

[6]: #| label: fig-bill-sizes
#| fig-cap: Bill Sizes by Species
import matplotlib.pyplot as plt
import seaborn as sns
g = sns.FacetGrid(df, hue="species", height=3, aspect=3.5/1.5)
g.map(plt.scatter, "bill_length_mm", "bill_depth_mm").add_legend()

[6]: <seaborn.axisgrid.FacetGrid at 0x2946720e0>

```

Palmer Penguins

author: Norah Jones
format: html
code-tools: true
code-fold: true

```

[3]: #| echo: false
import pandas as pd
df = pd.read_csv("palmer-penguins.csv")

```

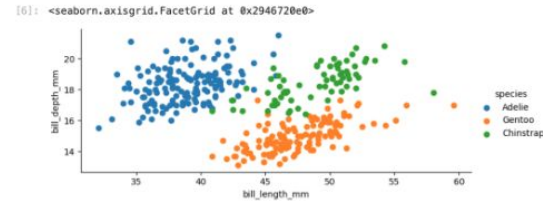
Exploring the Data

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

```

[6]: #| label: fig-bill-sizes
#| fig-cap: Bill Sizes by Species
import matplotlib.pyplot as plt
import seaborn as sns
g = sns.FacetGrid(df, hue="species", height=3, aspect=3.5/1.5)
g.map(plt.scatter, "bill_length_mm", "bill_depth_mm").add_legend()

```



Palmer Penguins

AUTHOR: Norah Jones
PUBLISHED: March 12, 2023

Code

Show All Code

Hide All Code

Exploring the Data

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

▼ Code

```

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

```

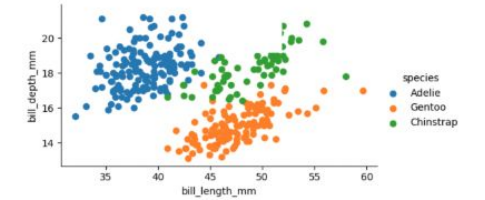


Figure 1: Bill Sizes by Species



Rmarkdown

```
quarto render code/supplementary_material.Rmd --to html  
quarto render code/supplementary_material.Rmd --to docx
```



Jupyter

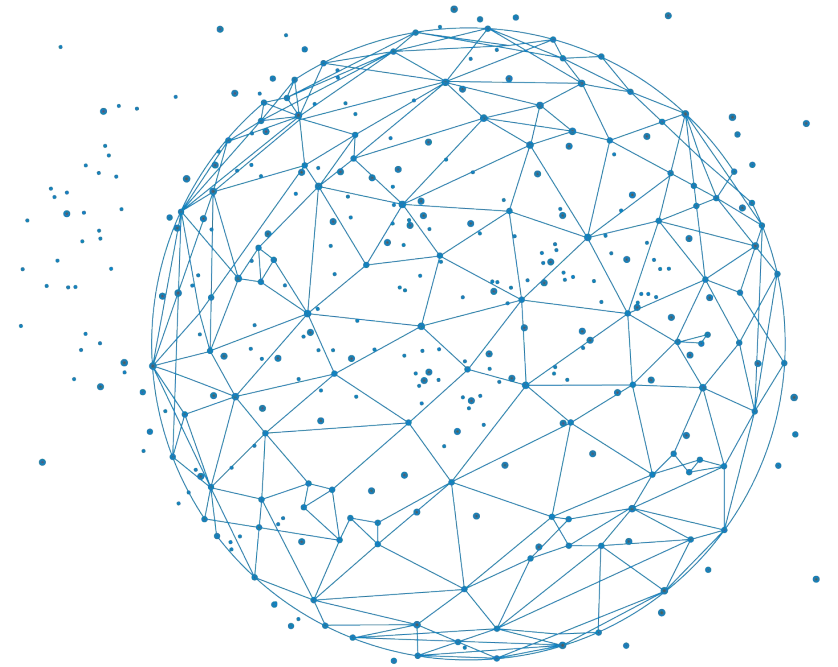
```
quarto render code/supplementary_material.ipynb --to html  
quarto render code/supplementary_material.ipynb --to docx
```



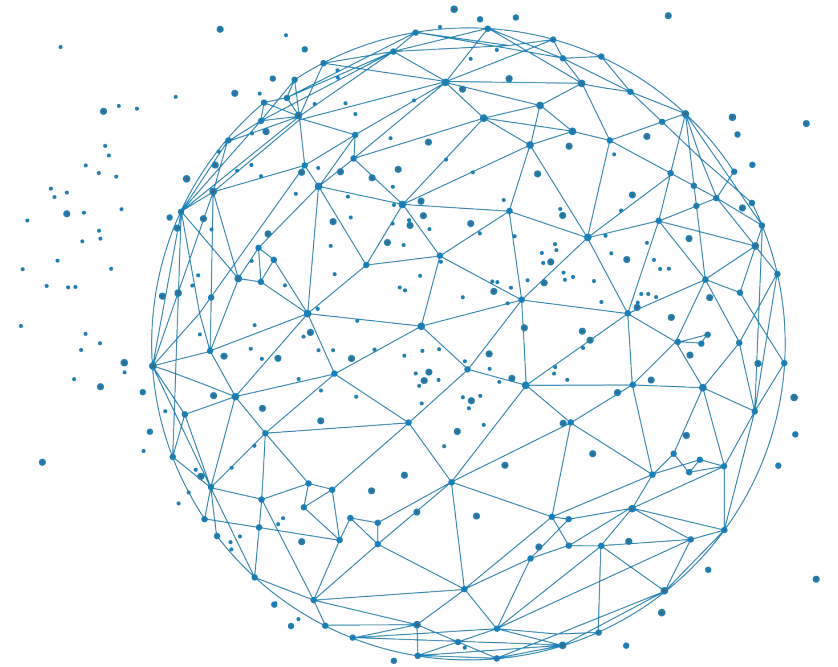


| Method | Jupyter | Rmarkdown | Quarto |
|-------------------------|---|--|---|
| IDE | JupyterHub, JupyterLab | R, Rstudio | VS Code, Jupyter, Rstudio, Neovim, TextEditor |
| Code mixing | Limited | Yes | Yes |
| Format | ipynb | Rmd | Qmd, Rmd, ipynb |
| Output | Asciidoc, HTML, LaTeX, MD, PDF, RST, Slide (Reveal) | HTML, PDF, Docx, ODT, RTF, MD, Slides (Powerpoint, Reveal,...), Dashboard, ... | HTML, PDF, Docx, ODT, Epub, RTF, MD, Slides (Powerpoint, Reveal,...), Wiki (MediaWiki, ...), Book, and more ! |
| Reproductibility | Easy | Easy | Yes (if done from the start) |

And now we try?



Observable





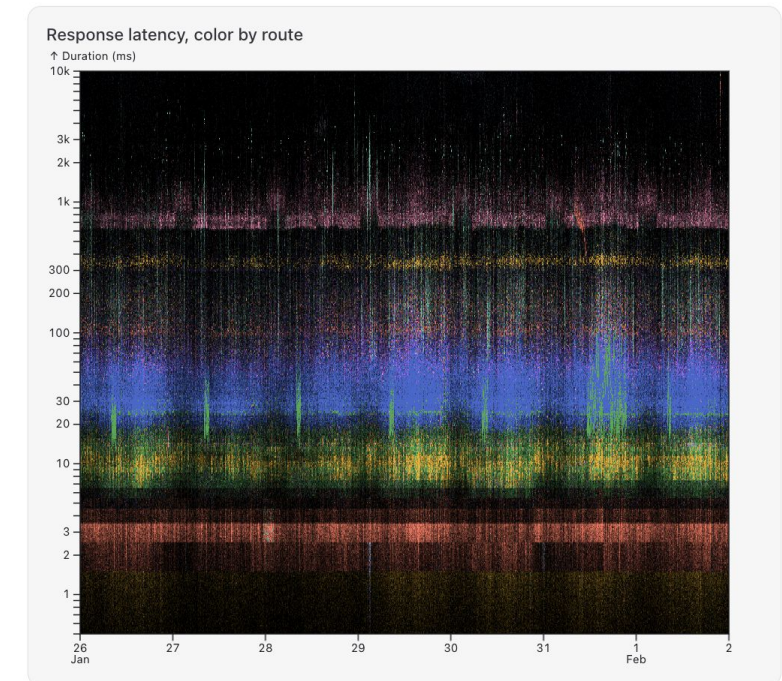
Observable Framework

Observable is an online platform that enables users to create, execute, and share interactive notebooks. These notebooks on Observable are based on JavaScript and allow users to combine code, visualizations, and text within a single interactive document.

Analyzing web logs

Web logs capture traffic metadata, such as the request time and route, how long the server took to respond, the response size, and so on. Analyzing web logs sheds light on both server performance and client behavior. Yet the common practice of summary statistics (e.g., 95th-percentile latency) often hides interesting patterns! This is because performance varies wildly based on the nature of the request, and unusual clients such as bots can easily hide in a sea of “natural” traffic.

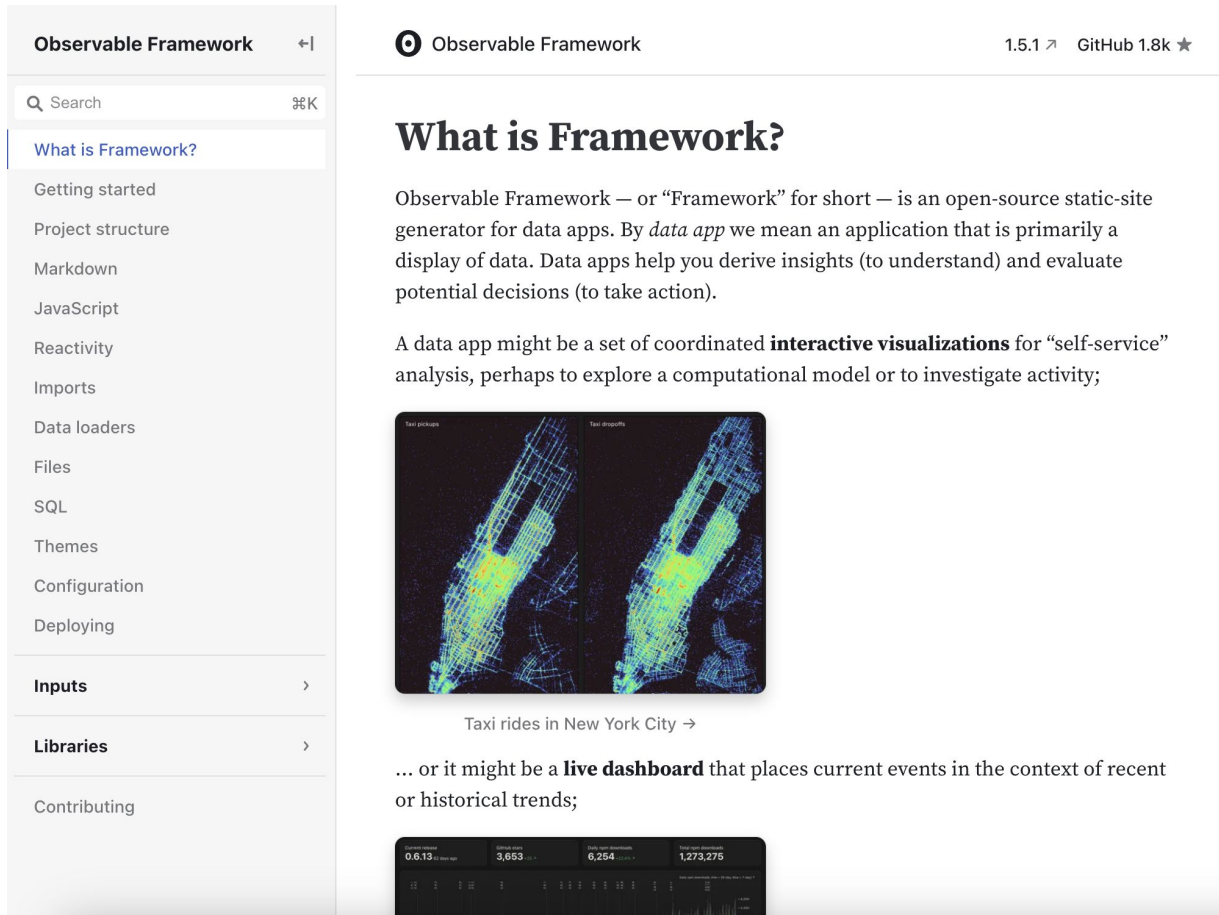
What if – instead of summarizing – we plotted *every* request as a dot with time along $x \rightarrow$ and latency (on a log scale) along $y \uparrow$?



The plot above shows a sample of 7,633,176 requests to Observable servers over a 7-day period. Color encodes the associated route. Hover to see the route.

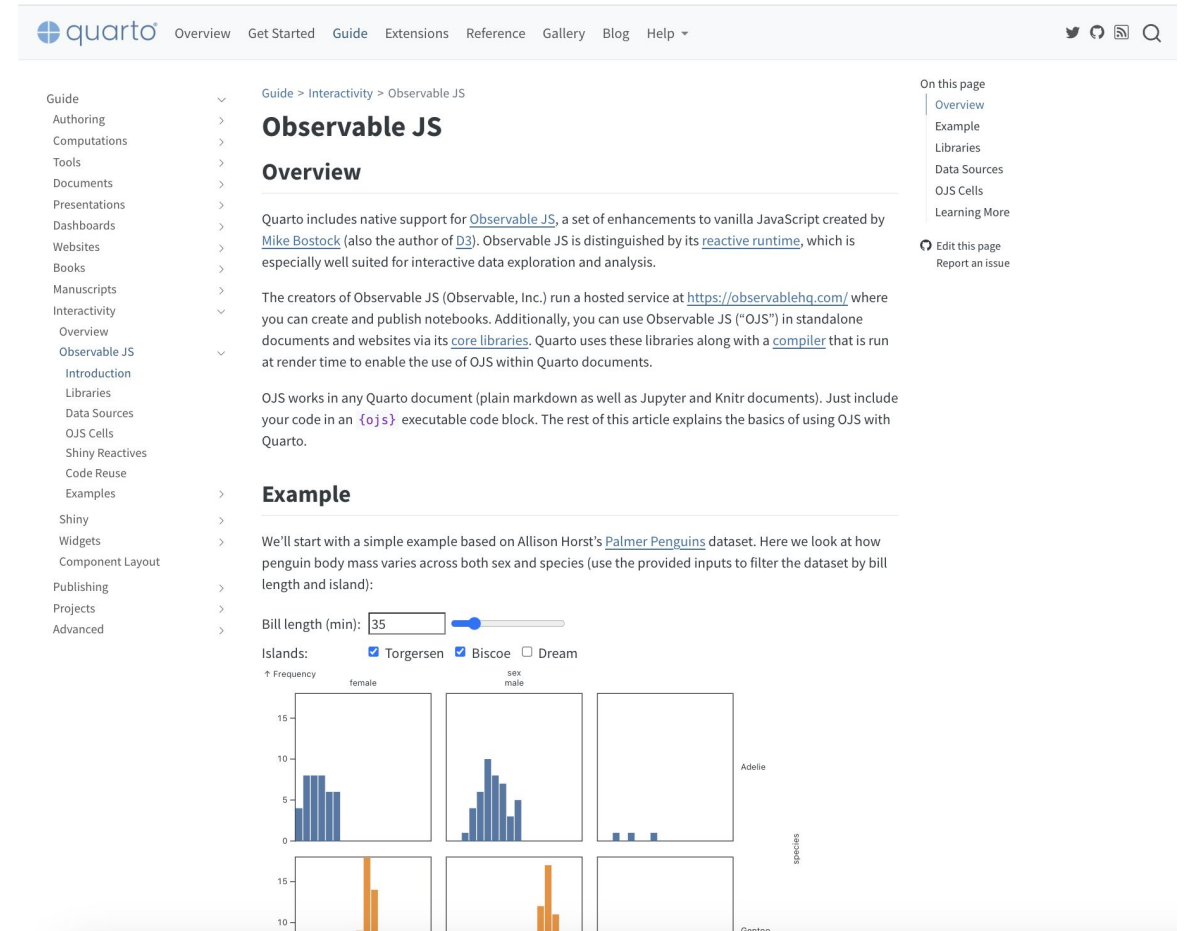
<https://observablehq.com/framework/examples/api/>

Official documentation



The screenshot shows the Observable Framework documentation page. On the left is a navigation sidebar with sections: Observable Framework, Search, What is Framework?, Getting started, Project structure, Markdown, JavaScript, Reactivity, Imports, Data loaders, Files, SQL, Themes, Configuration, Deploying, Inputs, Libraries, and Contributing. The main content area has the title "Observable Framework" with version "1.5.1" and "GitHub 1.8k". The main heading is "What is Framework?". The text explains that Observable Framework is an open-source static-site generator for data apps. Below the text is a visualization titled "Taxi rides in New York City" showing a heatmap of taxi routes. At the bottom, there is a snippet of a live dashboard with numerical data points.

Via quarto



The screenshot shows the Observable JS documentation page on the Quarto website. The top navigation bar includes "quarto" and links for Overview, Get Started, Guide, Extensions, Reference, Gallery, Blog, and Help. The left sidebar lists various Quarto guides. The main content area is titled "Observable JS Overview" and explains that Quarto includes native support for Observable JS, a set of enhancements to vanilla JavaScript. It mentions that Observable JS is distinguished by its reactive runtime. Below the text is an interactive example titled "Example" based on the Palmer Penguins dataset. It features a slider for "Bill length (min)" set to 35 and checkboxes for "Torgersen", "Biscoe", and "Dream" islands. The example displays a grid of histograms showing the frequency of penguin body mass across different species (Adelie and Gentoo) and sexes (female and male).