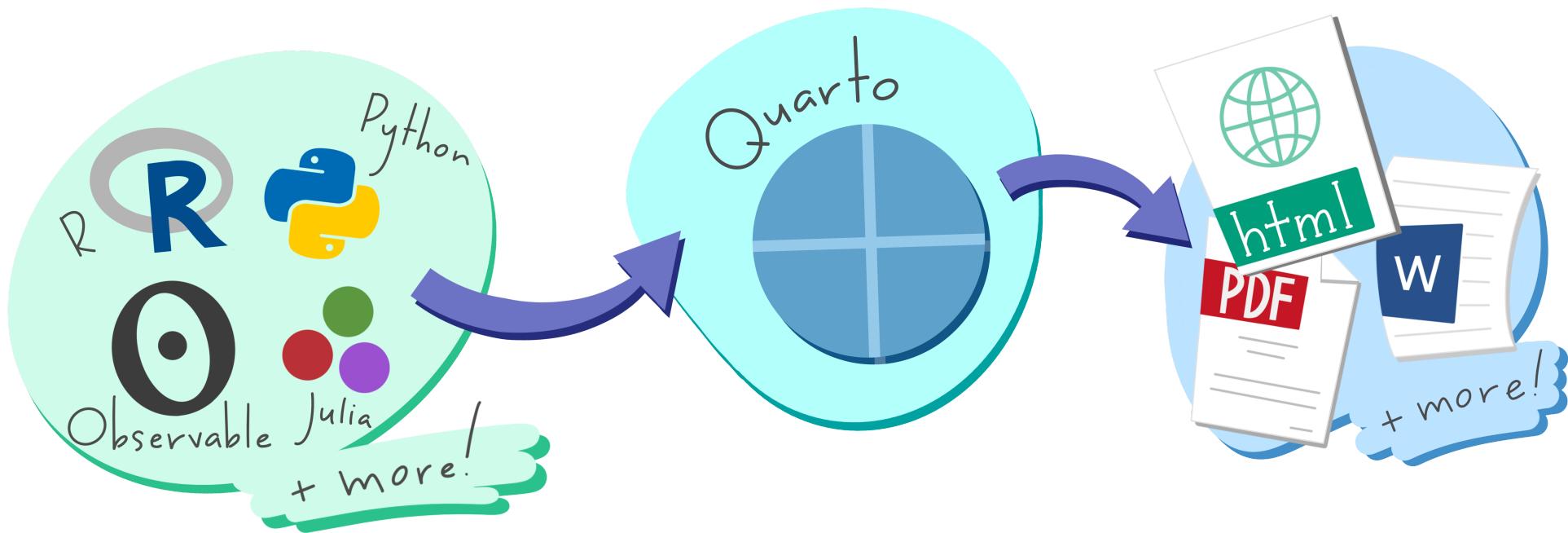


# Dynamic documents in R reproducible research with Quarto

2024-07-18



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

```
---
```

```
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")
)
```

```
```
```

# Source

## ggplot2 demo

Norah Jones

May 22nd, 2021

# Output

## Air Quality

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

### ► Code

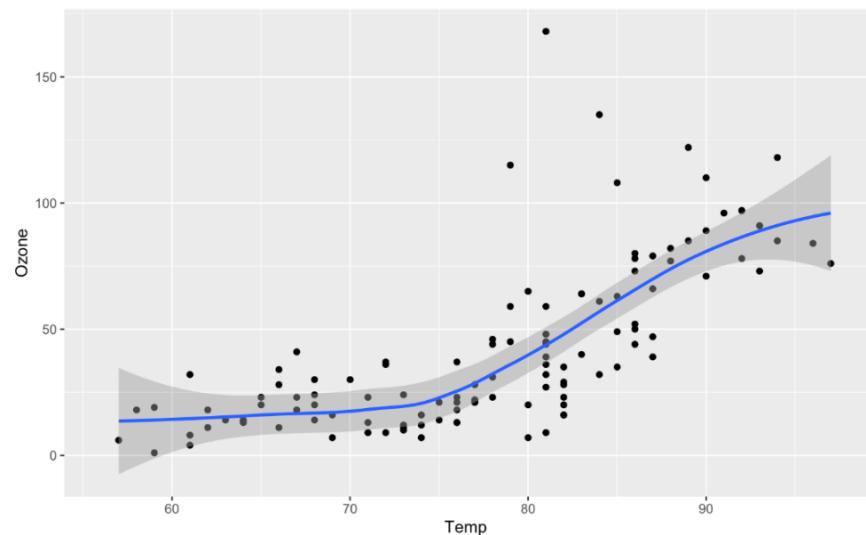
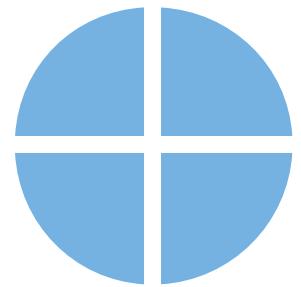


Figure 1: Temperature and ozone level.

# Quarto

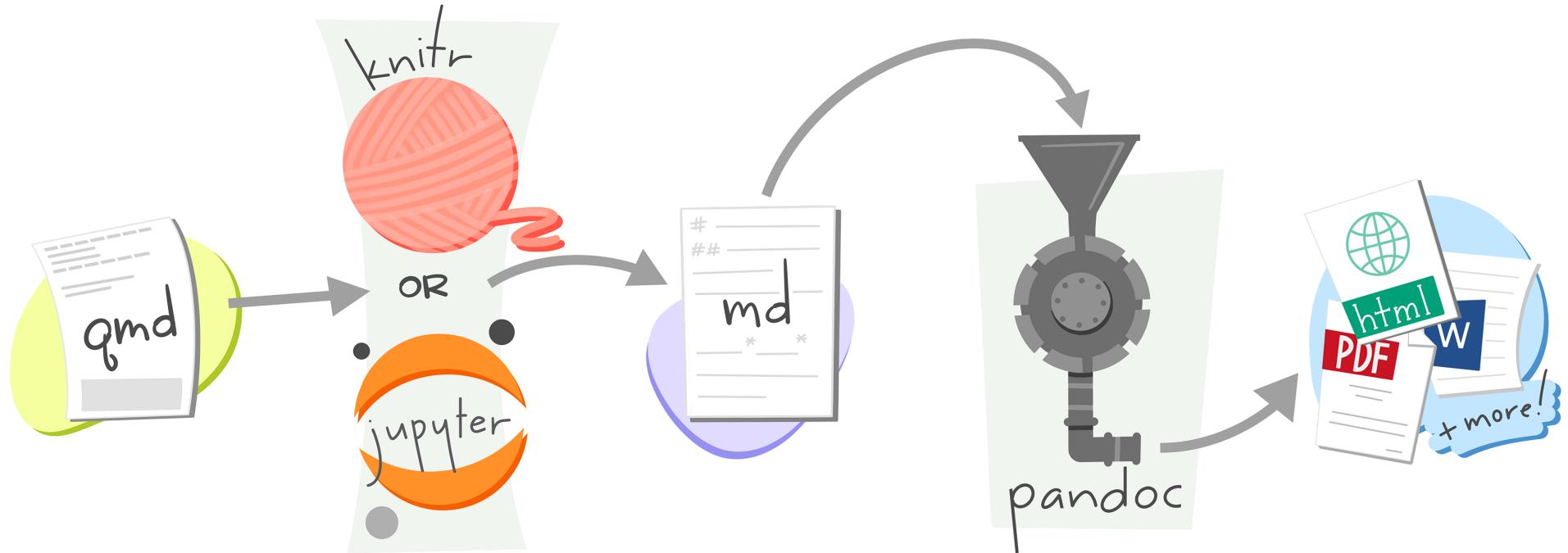


*Dynamic:* code and text in same document

*Reproducible:* re-run your analysis, re-render your document

*Flexible:* output to different formats easily

# Rendering



## *Your Turn 1*

**Create a new Quarto file. Go to File > New File > Quarto Document. Press OK. Save the file and press the “Render” button.**



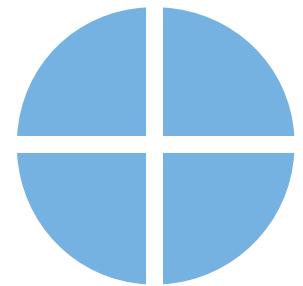
```
1 ---  
2 title: "Untitled" ← document metadata  
3 format: html  
4 ---  
5  
6 ## Quarto  
7  
8 Quarto enables you to weave together content and executable code into a  
finished document.  
9 To learn more about Quarto see <https://quarto.org>.  
10  
11 ## Running Code  
12  
13 When you click the **Render** button a document will be generated that includes  
both content and the output of embedded code.  
14 You can embed code like this:  
15  
16 ```{r} ← code chunks  
17 1 + 1  
18 ````  
19  
20 You can add options to executable code like this  
21  
22 ```{r} ← code chunks  
23 #| echo: false  
24 2 * 2  
25 ````  
26  
27 The `echo: false` option disables the printing of code (only output is  
displayed).  
28
```

# Quarto

## Prose

## Code

## Metadata

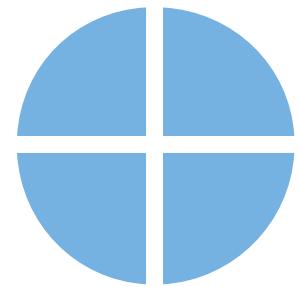


# Quarto

**Prose** = *Markdown*

Code

Metadata



# the Visual Editor

relational-data.qmd x

Source Visual B I Normal Format Insert Table Outline #filtering\_joins ...

## Filtering Joins

Filtering joins match observations in the same way as [mutating joins](#), but affect the observations, not the variables. There are two types:

|                 |                      |  |
|-----------------|----------------------|--|
| semi_join(x, y) | $x \ltimes y$        | Keeps all observations in x that have a match in y |
| anti_join(x, y) | $x \triangleright y$ | Drops all observations in x that have a match in y |

Graphically, a semi-join looks like this:

```
{r.join-semi}~  
#| echo: false~  
#| out-width: "6"~  
~  
knitr::include_graphics("diagrams/join-semi.png")~
```

| key | val_x |
|-----|-------|
| 1   | x1    |
| 2   | x2    |

Only the existence of a match is important; it doesn't matter which observation is matched. This means that filtering joins never duplicate rows like mutating joins do:

Chunk 1: join-semi Quarto

# Basic Markdown Syntax

\*italic\*    \*\*bold\*\*

\_italic\_    \_\_bold\_\_

# Basic Markdown Syntax

```
# Header 1
```

```
## Header 2
```

```
### Header 3
```

# Basic Markdown Syntax

`http://example.com`

`[linked phrase](http://example.com)`

*Learn more about Markdown Syntax with the  
ten-twenty minute tutorial on markdown at  
<https://commonmark.org/help/tutorial>.*

## **Your Turn 2 (open exercises.qmd)**

**Read this short introduction to the Visual Editor:**

**<https://quarto.org/docs/visual-editor/>**

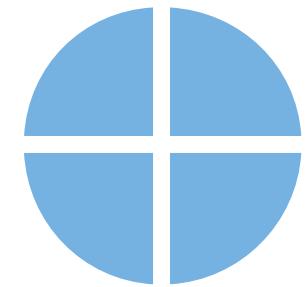
**Use the Visual Editor to stylize the text from the  
Gapminder website below. Experiment with  
bolding, italicizing, making lists, etc.**

# Quarto

Prose

**Code = *R code chunks***

Metadata



# Code chunks

```
```{r}
#| label: howl
#| echo: false
my_generation |>
  select(best_minds) |>
  filter(destroyed == "by madness")
```
```

# Code chunks



# Code chunks

```
chunk options
```{r}
#| label: howl
#| echo: false
my_generation |>
  select(best_minds) |>
  filter(destroyed == "by madness")
```
option values
```

# Code chunks

code chunk comments

```
```{r}
#| label: howl
#| echo: false
my_generation |>
  select(best_minds) |>
  filter(destroyed == "by madness")
```
```

# Chunk options

| Option                           | Effect                                       |
|----------------------------------|--|
| <code>include: false</code>      | run the code but don't print it or results   |
| <code>eval: false</code>         | don't evaluate the code                      |
| <code>echo: false</code>         | run the code and output but don't print code |
| <code>message: false</code>      | don't print messages (e.g. from a function)  |
| <code>warning: false</code>      | don't print warnings                         |
| <code>fig.cap: "Figure 1"</code> | caption output plot with "Figure 1"          |

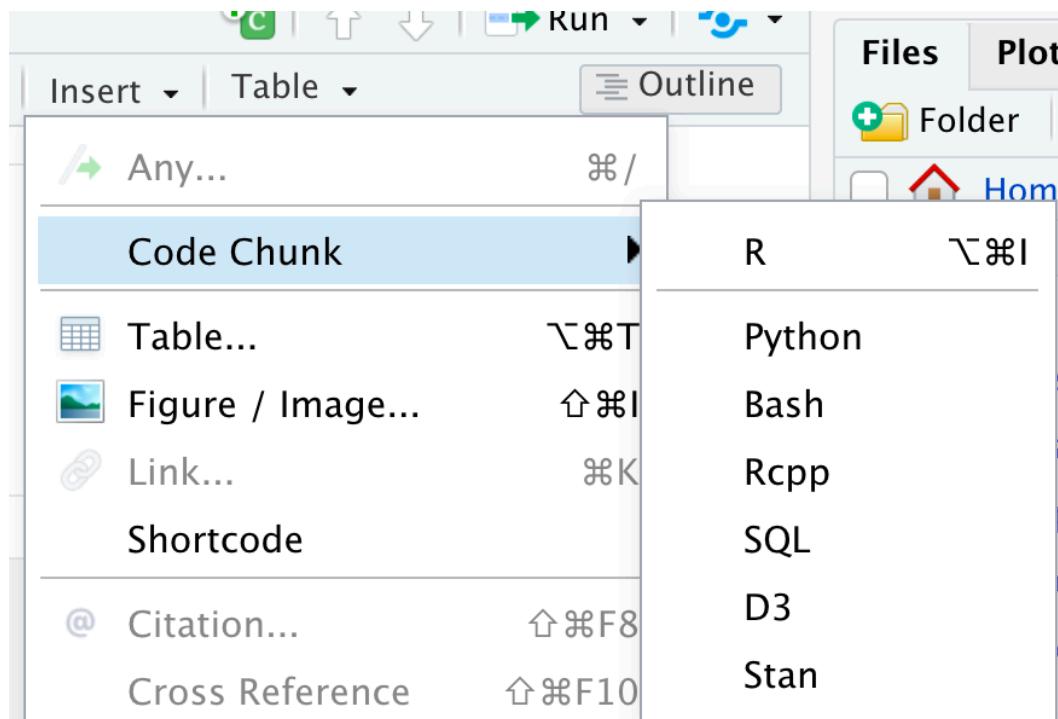
# Other languages

First-class support for Python, Julia, and  
Observable JS

Supports Jupyter notebooks

57 knitr engines

# Insert code chunks with cmd/ctrl + alt/option + I or Visual Editor



# Press **tab** to autocomplete chunk options

```
```{r}
#| echo: false
#
#| knitr:::knit_hooks$set(
#|   label = function(x, options) {
#|     if (!is.null(options$label)) {
#|       cat("label:", options$label, "\n")
#|     }
#|   }
#| )
## Op
```

label  
Unique label for code cell

You can type it in manually, use the keyboard short-cut (Cmd/ "Insert" button above. Put the following code in it:

```
```{r}
#| echo: | 
#| out.h | 
#| out.w | 
knitr:::i
```
true
false
fenced
```

## Your Turn 3

Create a code chunk. You can type it in manually, use the keyboard short-cut (Cmd/Ctrl + Option/Alt + I), or use the “Insert” button above. Put the following code in it:

```
1 gapminder |>
2   slice(1:5) |>
3     gt()
```

Render the document

## *Your Turn 4*

Add `echo: false` to the code chunk you created and re-render. What's the difference in the output?

# Inline Code

who wept at the romance of the  
streets with their pushcarts full of  
onions and bad music,  
who sat in boxes breathing in the  
darkness under the bridge, and rose  
up to build `r select(instruments,  
**harpsichord**)` in their lofts,  
who coughed on the `r length(floors)`  
floor of Harlem crowned with flame  
under the tubercular sky surrounded  
by orange crates of theology,  
who scribbled all night rocking and  
rolling over lofty incantations which  
in the yellow morning were stanzas of  
gibberish,

# Inline Code

who wept at the romance of the  
streets with their pushcarts full of  
onions and bad music,  
who sat in boxes breathing in the  
darkness under the bridge, and rose  
up to build `r select(instruments,  
single  
backtick + r harpsichord)` in their lofts,  
who coughed on the `r length(floors)`  
floor of Harlem crowned with flame  
under the tubercular sky surrounded  
by orange crates of theology,  
who scribbled all night rocking and  
rolling over lofty incantations which  
in the yellow morning were stanzas of  
gibberish,

**any R code**



## Your Turn 5

Remove `eval: false` so that Quarto evaluates the code.

Use `summarize()` and `n_distinct()` to get the number of unique years in `gapminder` and save the results as `n_years`.

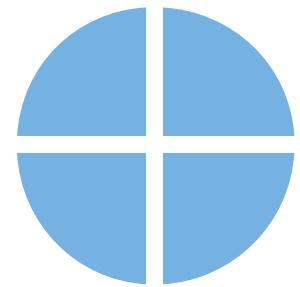
Use inline code to describe the data set in the text below the code chunk and re-render.

# Quarto

Prose

Code

**Metadata = YAML**



# YAML Metadata

```
1  ---
2  title: "Howl"
3  author: "Allen Ginsberg"
4  date: "March 18, 1956"
5  format:
6    html: default
7    pdf:
8      toc: true
9      number-sections: true
10 ---
```

# Output formats

| Format   | Outputs                  |
|----------|--------------------------|
| html     | HTML                     |
| pdf      | PDF                      |
| word     | Word .docx               |
| odt      | OpenOffice .odt          |
| gfm      | Github-flavored Markdown |
| revealjs | Reveal Slides (HTML)     |
| beamer   | Beamer Slides (PDF)      |
| pptx     | Powerpoint Slides        |

## Your Turn 6

Set figure chunk options to the code chunk below, such as `dpi`, `fig.width`, and `fig.height`. Run `knitr::opts_chunk$get()` in the console to see the defaults.

Add your name to the YAML header using `author: Your Name`.

Change `format: html` to use the `toc: true` and `code-fold: true` options and re-render

# Parameters

```
1 ---  
2 params:  
3   param1: x  
4   param2: y  
5   data: df  
6 ---
```

## *Calling parameters in R*

```
1 params$param1  
2 params$param2  
3 params$data
```

## *From the Console*

```
1 quarto::quarto_render(  
2   "document.qmd",  
3   execute_params = list(param1 = 0.2, param2 = 0.3)  
4 )
```

## Your Turn 7

Change the **params** option in the YAML header to use a different continent. Re-render.

```
1 gapminder |>
2   filter(continent == params$continent) |>
3   ggplot(aes(x = year, y = lifeExp, group = country, co:
4     geom_line(lwd = 1, show.legend = FALSE) +
5     scale_color_manual(values = country_colors) +
6     theme_minimal(14) +
7     theme(strip.text = element_text(size = rel(1.1))) +
8     ggtitle(paste("Continent:", params$continent))
```

# Bibliographies and citations

*Bibliography files:* .bib, Zotero, others

*Citation styles:* .csl

[@citation-label]

*Visual Editor's citation wizard can help!*

# Including bibliography files in YAML

```
1  ---
2  bibliography: file.bib
3  csl: file.csl
4  ---
```

*the Visual Editor can also manage this for you.*

# Your turn 8

Cite Causal Inference in text below. Using the citation wizard, find the right citation under My sources > Bibliography.

Add the American Journal of Epidemiology CSL to the YAML using csl: [aje.csl](#)

Re-render

# Make cool stuff in Quarto!

- Books
- Blogs
- These slides!

See the Gallery for inspiration

# Resources

**Quarto Documentation:** A comprehensive but friendly introduction to Quarto. Written in Quarto!

**R for Data Science:** A comprehensive but friendly introduction to the tidyverse. Free online.  
Written in Quarto!

**Posit Recipes:** Common code patterns in R (with some comparisons to SAS)

