# Shinylive is SO EASY

Max Kuhn

https://topepo.github.io/2024_05_NYR

# Quarto

- It is the next generation of what Rmarkdown, bookdown, blogdown (and all of the “downs”) started.
- Code chunks can target R, python, SQl, bash, juila, and so on.
- This is an open source technical publishing system that is independent of the target language (i.e., R, python, etc.).

# Shiny and Servers

Historically, to let {people who are not you} run a shiny app, you'd need:

- A shiny server of your own
- A hosted server (e.g. `shinyapps.io`, Posit Connect)

The server runs your code and renders the results on your computer/phone/tv etc.

To use a Shiny app in Quarto, you would need to embed a frame that would show the app (from the external shiny server)

An example app…

# Understanding UMAP

https://topepo.github.io/2024_05_NYR

# webR

R can be compiled to WebAssembly (aka wasm)

- R and packages are made into a binary format that can be embedded in the website.
- JavaScript is the interface that the browser uses.

See George Stagg's video for more information.

You can see which R packages can be used at the repository site. In mid May:

- Built R packages: 19,531 (about 94% of CRAN).
- Available R packages: 13,707 (about 66% of CRAN).

# webR

Loading webR...

```
1 mean(mtcars$mpg)
```

https://topepo.github.io/2024_05_NYR

# shinylive

A project for deploying Shiny applications that will run completely in the browser via webR

- Computations occur *locally*!

shinylive does *almost* all of the heavy lifting for you.

I'll focus on adding shiny apps to Quarto documents (a book, in my case)

shiny server: runs your code and renders the results on your client.

# shinylive: downloads R & packages to client then runs R/shiny on the client

https://topepo.github.io/2024_05_NYR

# Setup

## Download the extension for your Quarto project

```
quarto add quarto-ext/shinylive
```

## You also have to add a Quarto/Pandoc *filter* reference in your `_quarto.yml` file:

```
filters:
  - shinylive
```

# Code Chunks

Each shiny app goes in a single code chunk.

Instead of using `{r}` in the header, it is `{shinylive-r}`.

Also use the `standalone` option:

````
```{shinylive-r}
#| label: fig-shiny-spline
#| standalone: true

# shiny app code here

```
````

# Package declarations

To declare packages, use `library()` calls

- The renv package is used to figure this out.

Currently

- Shinylive grabs the latest built version of a package from the wasm repository at render-time

Upcoming

- Shinylive will bundle package files when the app is first built and tries to match package versions used at that time.

# Ingesting Data and Code

The one rule is:

https://topepo.github.io/2024_05_NYR

# If you live in JavaScript's house, you have to live by JavaScript's rules

https://topepo.github.io/2024_05_NYR

# Data access

You *don't* get automatic access to your local objects in your workspace

Opening raw network sockets is not permitted from the WebAssembly sandbox (no curl).

webR patches `download.file()` to use an XHR requests (JS API for HTTP requests).

It's a work in progress. (GH issue: Add local folder support).

# As always, be careful…

From Gordon Shotwell:

- All the source code goes to the client
- All of the data goes to the client
- Don’t include credentials
- Don’t include sensitive data

https://topepo.github.io/2024_05_NYR

# An Example Pattern

````
```{shinylive-r}
#| label: fig-potato
#| standalone: true

library(tidymodels)
1 # other library() calls

2 load("https://raw.githubusercontent.com/{user}/{repo}/{branch}/{file}")

3 source("https://raw.githubusercontent.com/{user}/{repo}/{branch}/{file}")

4 app
```
````

1. Packages should be listed in the code chunk.
2. Load the data in the code chunk.
3. `source()` with a local file won't work.
4. Return the shiny app in the code chunk.

https://topepo.github.io/2024_05_NYR

# Why would Posit support this?

For many applications, shinylive is not the answer:

- Client computing power may not be great.
- Moving data to the client may be expensive.
- Data/Code security is paramount to many.
- Niceties of authentication, parameterization, support, etc that Connect offers.

# More Information

Must watch: Running R-Shiny without a Server - posit::conf(2023)

Examples:

- All R examples on GitHub
- All Python examples on GitHub

Specific examples by George Stagg:

- Reproducible data science with webR and Shinylive 2024
- Shinylive example with duckplyr and parquet files

# Thanks

Thanks for the invitation to speak again!

Thanks to George Stagg, Joe Cheng, Winston Chang, Gordon Shotwell, and everyone else who made `webR` and `shinylive` happen.