

# Shing

showmeshiny.com







# Data in Shiny Apps

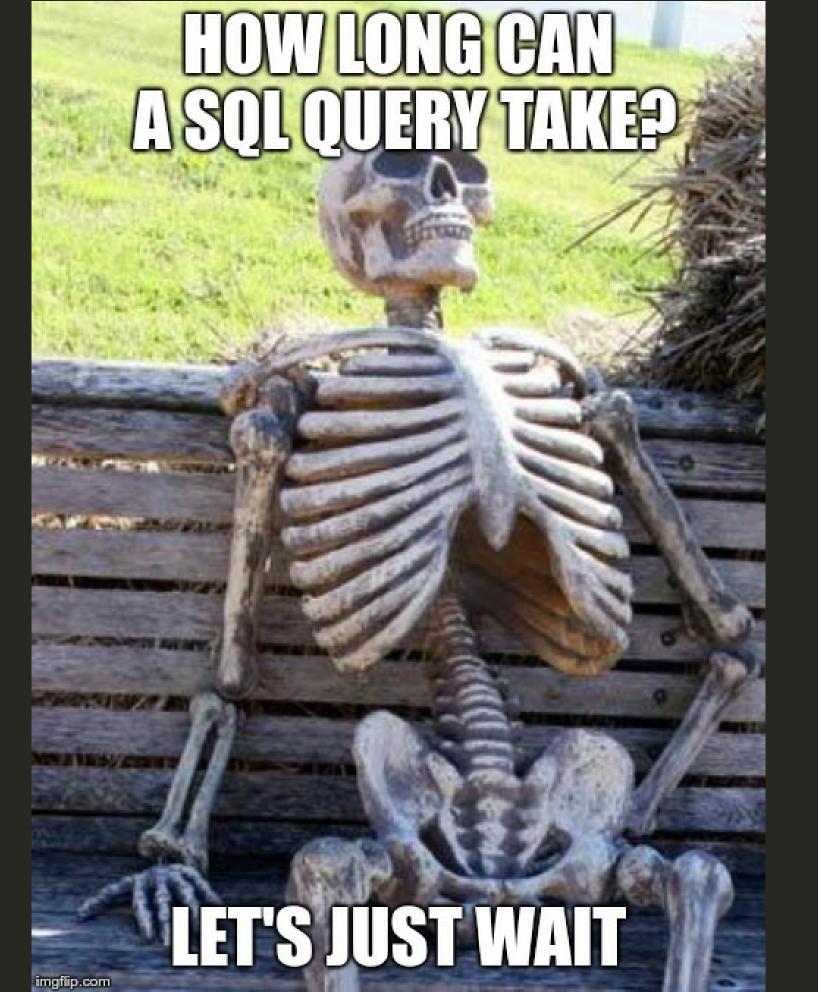
- Typically static
  - Maybe CSV
  - Maybe database
  - Maybe an API





# Updating data in a shiny app

- Add data to the database
- Refresh the page



### What could we do?

• reactiveTimer() or equivalent





# Whats the problem?

- How often should I make a request? aka Goldilocks principal
  - Not enough slow updates in dashboard
  - Too muc shiny server spends loads of resource making requests
- Is there a rate limit?
  - Is the API charged per request?
- If nothing has changed, I'm wasting time
  - Not quite real time (I know it's close enough to not matter but it irks me)





#### What do we want?

- Something that lets me send data directly to my Shiny app
- Shiny web server responds to GET requests
- As far as I know there is no direct support for adding POST endpoints to an app



# Hacky solution

```
shinyServer(function(input, output, session) {
  api_url <- session$registerDataObj (</pre>
           = 'api', # an arbitrary but unique name
    name
    data = list(), # you can bind some data here
    filter = function(data, req) {
      if (req$REQUEST_METHOD == "GET") {
        query <- parseQueryString(req$QUERY_STRING)</pre>
        # etc...
      if (req$REQUEST_METHOD == "POST") {
        # handle POST requests here
        reqInput <- req$rook.input</pre>
        # ...
  ) #stackoverflow.com/q/25297489/
  # because the API entry is UNIQUE, we need to send it to the client
  # we can create a custom pipeline to convey this message
  session$sendCustomMessage("api_url", list(url=api_url))
})
```





# Hands up

- I don't know if my approach is any less hacky than this
- But we can get around the communicating the endpoint info to a client browser
- I also don't really know anything about networking
- Maybe it looks less hacky (?)





#### Websockets

- Communication protocol
- Persistent connection between a client and a server
- Two way communication
- Internal guts of shiny use them

#### In R:

- httpuv
- websocket





#### Version 0.1

- Kind of works
- Except when it doesn't
- Can't share graphs across sessions
- No data persistence





## Version 0.7

- I decided I had to rethink things
- Terrible pun incoming

#### Rethink

When your app polls for data, it becomes slow, unscalable, and cumbersome to maintain.

RethinkDB is the open-source, scalable database that makes building realtime apps dramatically easier.

- rethinker
- shiny.collections





# shiny.collection

- Has an example of sharing data between shiny sessions
- Last library update 2017
- Still doesn't quite do what I want
- Can't insert from a non reactive context (i.e outside a shiny app)
- What's the answer

# Fork







# Changes

- Only needed a few small changes
- Essentially taking away reactive structure from things that would never change
  - database name
  - table handle
  - connection info





# Voila!

- Send any type of data from anywhere
- Share across multiple sessions
- Persistence





#### Issues

- Seem to have to run the socket server and the shiny server from the same environment
- Would be much better to have this separate, but it stops auto updating
- Currently no idea why



#### What to do next

- Try using plumber to create endpoints to send things to
- Support for a single structure that lets you send other types of information