



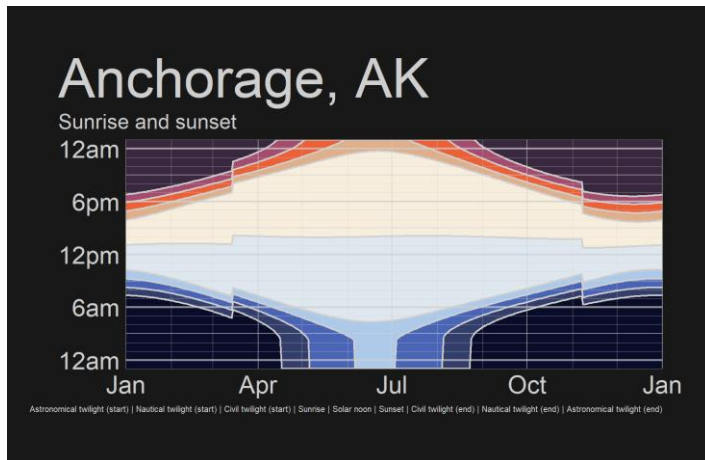
I made an entire e-commerce platform on Shiny

Jacqueline Nolis | @skyetetra | jnolis.com

{ggirl} – make your **GG**plots exist **In Real Life**

```
remotes::install_github("jnolis/ggirl")
```

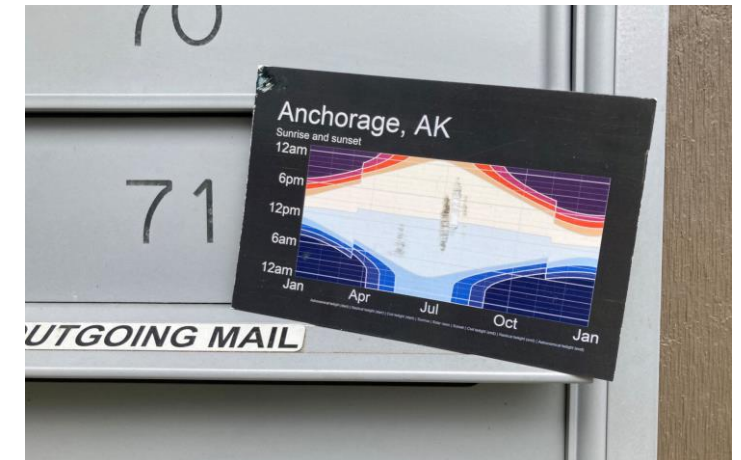
A ggplot2 plot



Paying a modest fee



An actual postcard



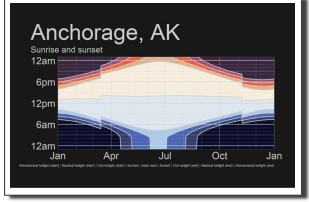
Live demo time!

```
ggpostcard(plot = plot,  
  contact_email = "ggirl@jnlolis.com",  
  messages = "Greetings from R!",  
  send_addresses =  
    address(name = "RStudio::conf",  
      address_line_1 = "165 Waterfront ST",  
      city = "National Harbor",  
      state = "MD",  
      postal_code = "20745",  
      country = "US"))
```

Call R code



Front preview



Order your ggplot2 postcard.

Check that you like the front and the back content of all the postcards. Once you're ready, click the pay and submit button to finalize the order.

- Postcards are printed on 4"x6" cardstock (14pt weight).
- Postcards take 5-10 business days between ordering and delivery for US addresses. International addresses may take several weeks.
- The postal system is a rough and tumble place—the postcard may not arrive in pristine condition.

The price for a single postcard is \$2.50.

Pay and submit

Back content

Message	Address
This plot made me think of you!	Fake Personname 250 North Ave Boston, MA, 22222 US

Preview image on a Shiny webpage



GGIRL

Pay GGIRL

\$2.50

ggplot2 postcard \$2.50
A custom printed postcard of a ggplot2 plot.

Subtotal \$2.50

Tax @ Enter address to calculate

Total due \$2.50

Pay with card

Email fakeemailforreal@gmail.com

Card information

1234 1234 1234 1234

MM / YY CVC

Name on card

Billing address

United States

Address line 1

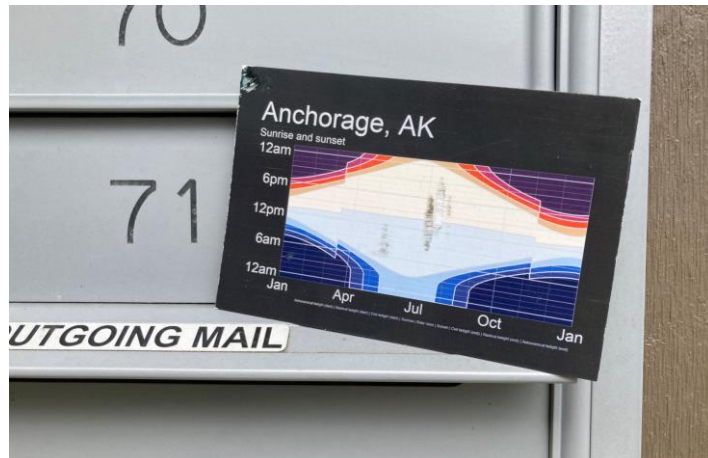
Address line 2

City ZIP

State

Pay \$2.50

Make Stripe payment

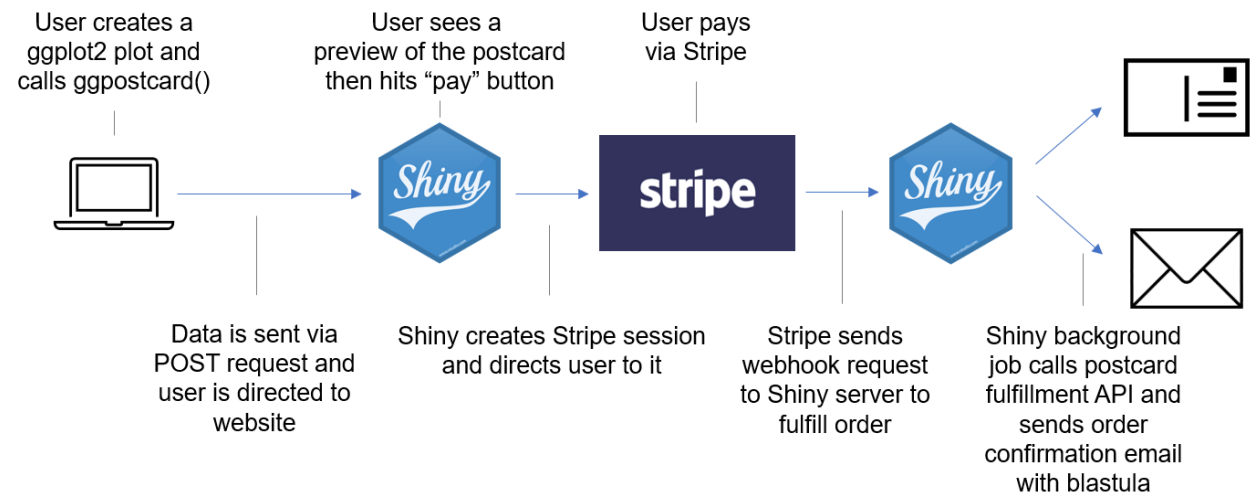


Get a postcard (& confirmation email)

How does it work?

- `{ggirl}`: R package that takes plot and passes it to a Shiny server
- `ggirl-server`: Shiny app that does the work of
 - Showing the user a preview
 - Passing user to Stripe to pay
 - Noticing when user paid on Stripe
 - Fulfilling the postcard order via an API
 - Sending confirmation email (or error email)

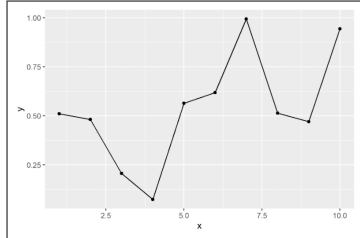
`ggirl-server` *should* be a collection of Shiny/plumber apps, but I wanted just a single Shiny app for ease of maintenance



```
1 # install.packages("ggplot2")
2
3 library(ggplot2)
4
5 plot <-
6   ggplot(data.frame(x=1:10, y=runif(10)), aes(x=x, y=y)) +
7   geom_line() + geom_point() +
8   labs(title = "Hello From Cascadia R Conf!!!")
9
10 plot
11
12 # -----
13 library(ggir)
14
15 contact_email <- "mollis.test@gmail.com"
16 send_address <- address(name = "Fake Personname", address_line_1 = "250 North Ave",
17   city = "Boston", state = "MA",
18   postal_code = "22222", country = "us")
19 message <- "This plot made me think of you!"
20
21 # -----
22
23 ggpostcard(plot, contact_email, messages = message, send_addresses = send_address)
```



Front preview



Back content

Message	Address
This plot made me think of you!	Fake Personname 250 North Ave Boston, MA, 22222 US

Order your ggplot2 postcard.

Check that you like the front and the back content of all the postcards. Once you're ready, click the pay and submit button to finalize the order.

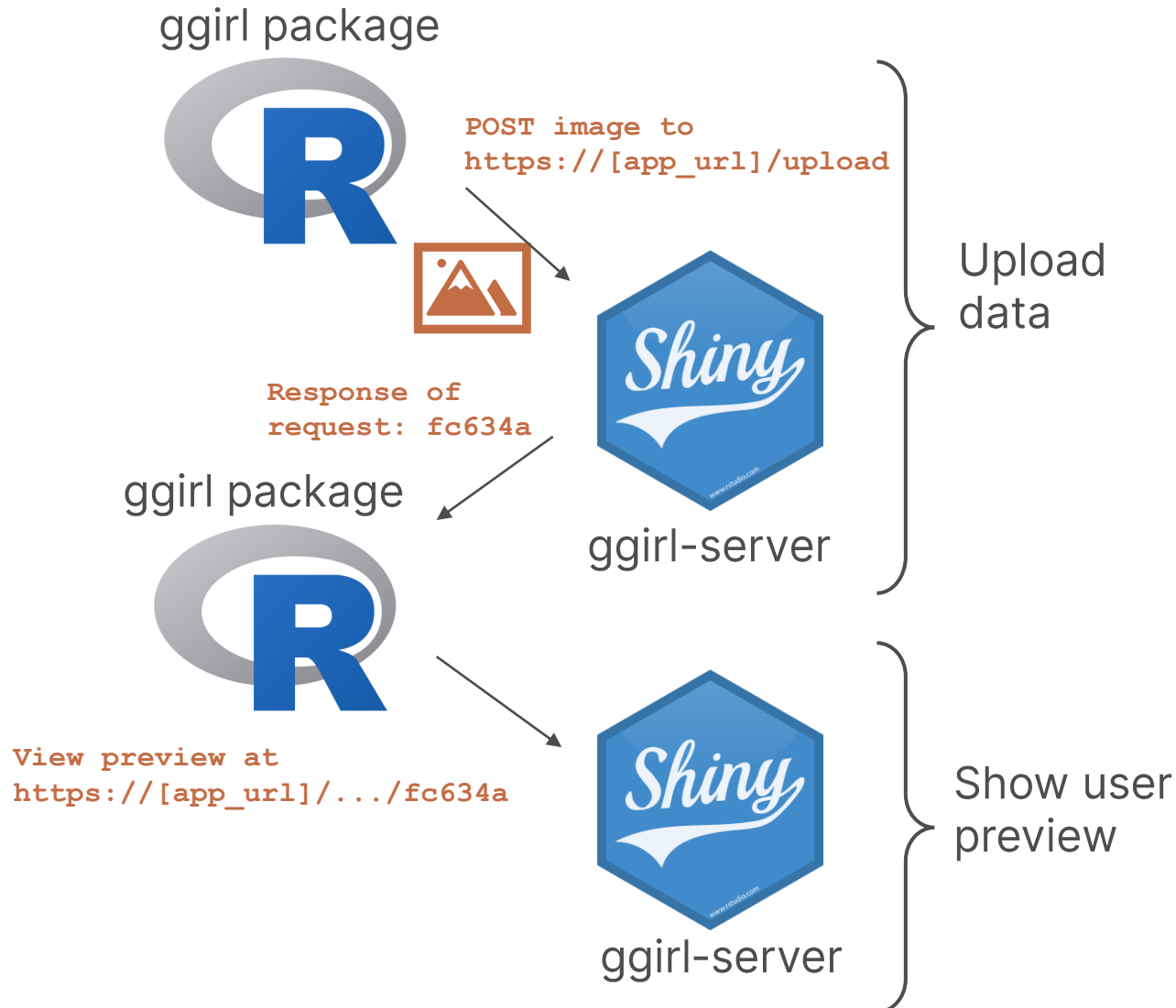
- Postcards are printed on 4"x6" cardstock (14pt weight).
- Postcards take 5-10 business days between ordering and delivery for US addresses. International addresses may take several weeks.
- The postal system is a rough and tumble place—the postcard may not arrive in pristine condition.

The price for a single postcard is \$2.50.

Pay and submit

1. How do you pass the plot from R to the Shiny app?

Get data to Shiny with HTTP POST



- Should be done with a HTTP POST request
 - Shiny does not have that documented functionality
 - It should!
- There are actually a number of undocumented ways to do this

The {brochure} package

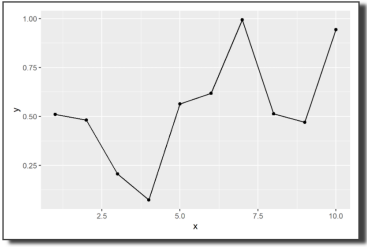
- Experimental {brochure} package from Colin Fay lets you:
 - **Connect multiple Shiny apps** together in a single bigger app as different endpoints
 - **Listen for requests besides just GET**
- Solves two of my problems!
 - Removes need for multiple apps/APIs
 - Removes need for Plumber for POST requests



github.com/ColinFay/brochure

Code for brochure and two other ways to POST to Shiny available at github.com/jnolis/ggml-examples

Front preview



Order your ggplot2 postcard.

Check that you like the front and the back content of all the postcards. Once you're ready, click the pay and submit button to finalize the order.

- Postcards are printed on 4"x6" cardstock (14pt weight).
- Postcards take 5-10 business days between ordering and delivery for US addresses. International addresses may take several weeks.
- The postal system is a rough and tumble place—the postcard may not arrive in pristine condition.

The price for a single postcard is \$2.50.

Pay and submit

Back content

Message	Address
This plot made me think of you!	Fake Personname 250 North Ave Boston, MA, 22222 US

2. How to show the image preview page

Showing a user preview

This is a Shiny app with one button.

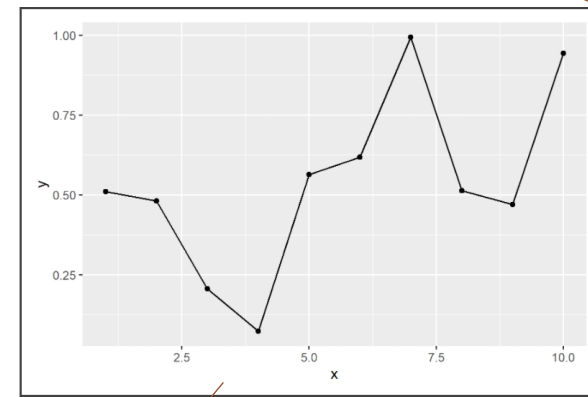
The ID of the particular ggplot is a query parameter in the URL.

```
https://[app_url]/postcard?token=[token]
```

Making Shiny apps look good with HTML and CSS is cool and fun and I gave a whole different talk about it at Shiny Conf 2022:

link.jnolis.com/shiny-conf-2022

Front preview



Shadow and frame uses a few lines of custom css

Column layout via bootstrap grid

Order your ggplot2 postcard.

Check that you like the front and the back content of all the postcards. Once you're ready, click the pay and submit button to finalize the order.

- Postcards are printed on 4"x6" cardstock (14pt weight).
- Postcards take 5-10 business days between ordering and delivery for US addresses. International addresses may take several weeks.
- The postal system is a rough and tumble place—the postcard may not arrive in pristine condition.

The price for a single postcard is \$2.50.

Pay and submit

Back content

Message

Address

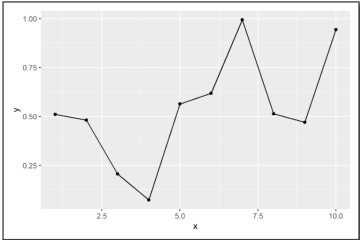
Fake Personname
250 North Ave
Boston, MA, 22222
US

Margin added to image so printer doesn't cut off plot

The important button

{gt} table

Front preview



Order your ggplot2 postcard.

Check that you like the front and the back content of all the postcards. Once you're ready, click the pay and submit button to finalize the order.

- Postcards are printed on 4"x6" cardstock (14pt weight).
- Postcards take 5-10 business days between ordering and delivery for US addresses. International addresses may take several weeks.
- The postal system is a rough and tumble place—the postcard may not arrive in pristine condition.

The price for a single postcard is \$2.50.

Pay and submit

Back content

Message	Address
This plot made me think of you!	Fake Personname 250 North Ave Boston, MA, 22222 US



Pay ggplot2

\$2.50

ggplot2 postcard

A custom printed postcard of a ggplot2 plot.

Subtotal

\$2.50

Tax ID

Enter address to calculate

Total due

\$2.50

Pay with card

Email

nolis.test@gmail.com

Card information

1234 1234 1234 1234

MM / YY

CVC

Name on card

Billing address

United States

Address line 1

Address line 2

City

ZIP

State

☐ Save information to pay faster next time

Pay \$2.50

3. How to pass the user from Shiny to Stripe?

Having users pay with Stripe

- Surprisingly easy
- Steps in payment flow:
 - Shiny UI makes POST request for a new Stripe session which has the list of items to buy and prices
 - Stripe JavaScript library makes button send user to Stripe
 - Add library in Shiny with `tags$script` and `Shiny.addCustomMessageHandler`

User presses
button

**1. Shiny send
POST of cart info**

**2. Shiny triggers
Stripe JavaScript**

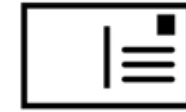
User is redirected
to stripe



Code for this is available at github.com/jnolis/ggirl-examples

The screenshot shows a Stripe checkout page for a payment of \$2.50. The order is for a 'ggirl' postcard. The payment method is 'Pay with card'. The card information field is highlighted with a blue border. The billing address is 'United States'. The page is powered by Stripe.

Item	Amount
ggirl2 postcard	\$2.50
Subtotal	\$2.50
Total due	\$2.50



4. How to fulfill the order?

Have Stripe send a POST telling Shiny to order postcards

Steps:

1. Stripe sends a POST request (aka webhook) saying “Order xyz has been paid for” (+ keys to ensure authentic)
2. Shiny receives the request and sends a POST to the postcard printing company
3. Shiny send an email to the user saying the order is confirmed

Feels like it should be straightforward...



Fulfill orders in background process

- Stripe needs a response to know the message was received
 - If the response doesn't come REALLY QUICKLY, Stripe retries
 - Shiny won't respond to a request until it's work is done
- Use {callr} package to spawn a separate parallel workflow to do the fulfillment
- I love this so much I gave a whole talk about it: link.jnolis.com/r-parallel-talk

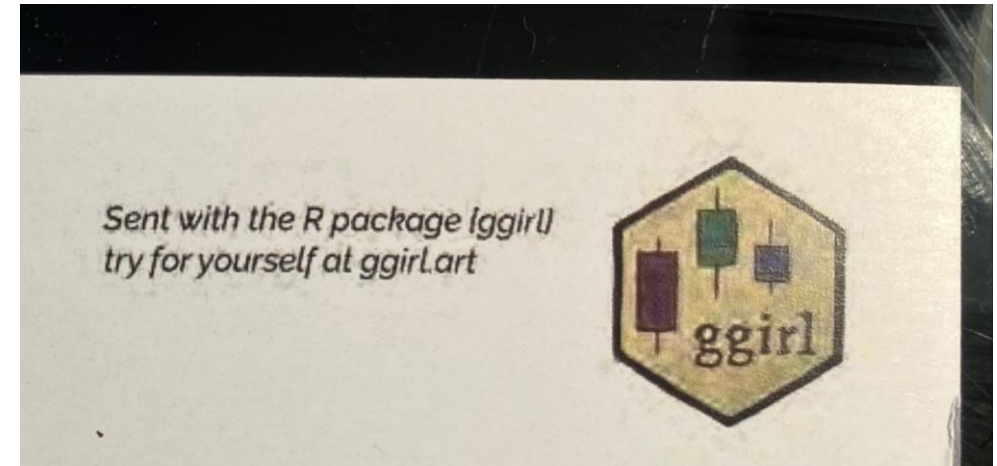
```
jobs <- list()
run_token <- function(token){
  Sys.sleep(10)
  write.csv(data.frame(),paste0(token,".csv"))
}
server <- function(input, output, session) {
  observe({
    if(input$startjob == 1){
      token <- UUIDgenerate()
      message(paste0("running task for token: ", token))
      if(is.null(jobs[[token]])){
        jobs[[token]] <-
          callr::r_bg(run_token, args = list(token = token))
      }
    }
  })
}
```



Code for this is available at github.com/jnolis/ggml-examples

Actually ordering the postcards

- There are **many** companies that have APIs to mail postcards for you via POST request
- Time between sending the order and getting the postcard was generally 2 weeks
- This took forever to find a good one



It's not your monitor: this actual real test postcard looked like a bad Facebook meme IRL. I did not go with them.

Send confirmation emails

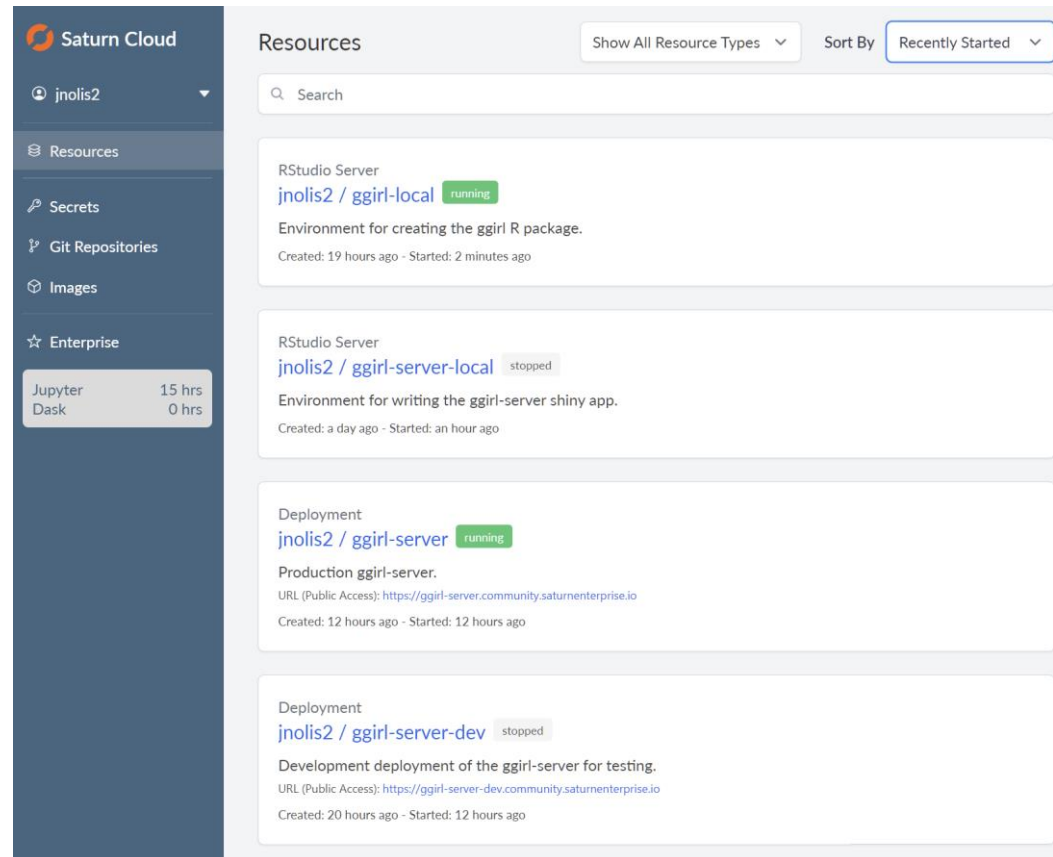
- Use blastula to send email (easy!)
- Thoughtfully resize image with {imagemagick}
- If there is an error in fulfillment function, have it email me the error message



How to deploy it?

DevOps is easy with Saturn Cloud

- Saturn Cloud is a great data science platform
 - Write code in RStudio Server
 - Deploy Shiny apps to Saturn Cloud
 - Free to use for 30 hours a month
- I had environments for
 - Writing ggir (R package)
 - Writing ggir-server (Shiny app)
 - Deploying dev ggir-server
 - Deploying prod ggir-server
- Ran multiple beta tests with real users to iron out the issues



The screenshot displays the Saturn Cloud web interface. On the left is a dark blue sidebar with navigation links: 'jnlis2' (selected), 'Resources', 'Secrets', 'Git Repositories', 'Images', and 'Enterprise'. Below these are resource usage statistics: 'Jupyter Dask' with '15 hrs' and '0 hrs'. The main panel is titled 'Resources' and includes a search bar, a dropdown for 'Show All Resource Types', and a 'Sort By' dropdown set to 'Recently Started'. The resource list contains four entries:

- RStudio Server**: `jnlis2 / ggir-local` (status: **running**). Description: 'Environment for creating the ggir R package.' Created: 19 hours ago - Started: 2 minutes ago.
- RStudio Server**: `jnlis2 / ggir-server-local` (status: **stopped**). Description: 'Environment for writing the ggir-server shiny app.' Created: a day ago - Started: an hour ago.
- Deployment**: `jnlis2 / ggir-server` (status: **running**). Description: 'Production ggir-server.' URL (Public Access): <https://ggir-server.community.saturnenterprise.io>. Created: 12 hours ago - Started: 12 hours ago.
- Deployment**: `jnlis2 / ggir-server-dev` (status: **stopped**). Description: 'Development deployment of the ggir-server for testing.' URL (Public Access): <https://ggir-server-dev.community.saturnenterprise.io>. Created: 20 hours ago - Started: 12 hours ago.

R package
development

Shiny app
development

Prod
deployment

Dev
deployment

Wrapping it up

- {ggirl} is a cool package for ordering cool stuff
- Wild things I didn't know if they could be done in Shiny but I did anyway:
 - Receiving data from an R package
 - Handling POST requests in Shiny
 - Sending/receiving Stripe API calls
 - Spawning background processes



Thank you!

Jacqueline Nolis | @skyetetra | jnolis.com

Install the package `remotes::install_github("jnolis/ggirl")`

Code for the rstudio::conf(2022) postcard link.jnolis.com/rstudio22-code

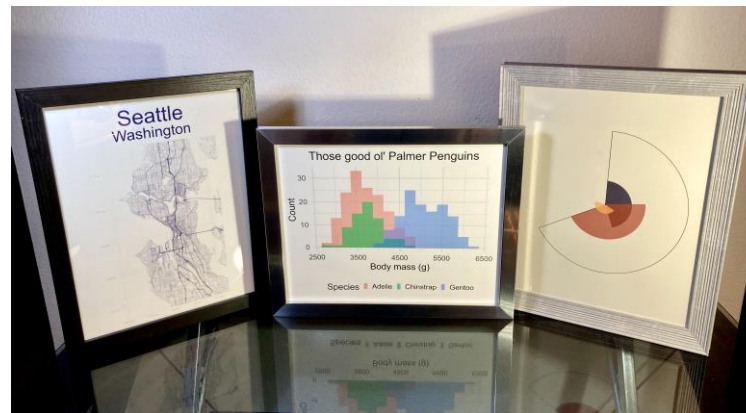
Code examples from gggirl-server github.com/jnolis/ggirl-examples

Slides from this talk link.jnolis.com/rstudio22-slides

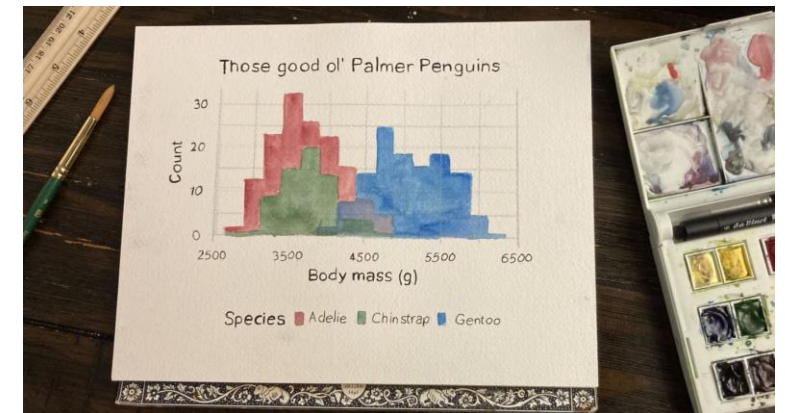
Come talk to me in person for a 80% off a postcard coupon!



ggpostcard()



ggartprint()



ggwatercolor()