Getting started with ShinyApps.io

ADDED: 18 MAR 2014
BY: ANDY KIPP

ShinyApps.io is a platform as a service (PaaS) for hosting Shiny web apps (applications). This guide will show you how to create a ShinyApps.io account and deploy your first application to the cloud.

Before you get started with ShinyApps.io, you will need:

- An R development environment, such as the RStudio IDE
- (*for Windows users only*) RTools (http://cran.r-project.org/bin/windows/Rtools/) for building packages
- (*for Mac users only*) XCode Command Line Tools for building packages
- (*for Linux users only*) GCC
- The devtools (https://github.com/hadley/devtools) R package (version 1.4 or later)
- The latest version of the shinyapps (https://github.com/rstudio/shinyapps) R package

How to install `devtools`

ShinyApps.io uses the latest improvements to the `devtools` package. To use ShinyApps.io, you must update `devtools` to version 1.4 or later. To install `devtools` from CRAN, run the code below. Then restart your R session.

```
install.packages('devtools')
```

How to install `shinyapps`

The `shinyapps` package deploys applications to the ShinyApps.io service. Currently, you need to install the `shinyapps` package from its development page at Github. You can do this by running the R command:

```
devtools::install_github('rstudio/shinyapps')
```

After the `shinyapps` package has been installed, load it into your R session:

```
library(shinyapps)
```

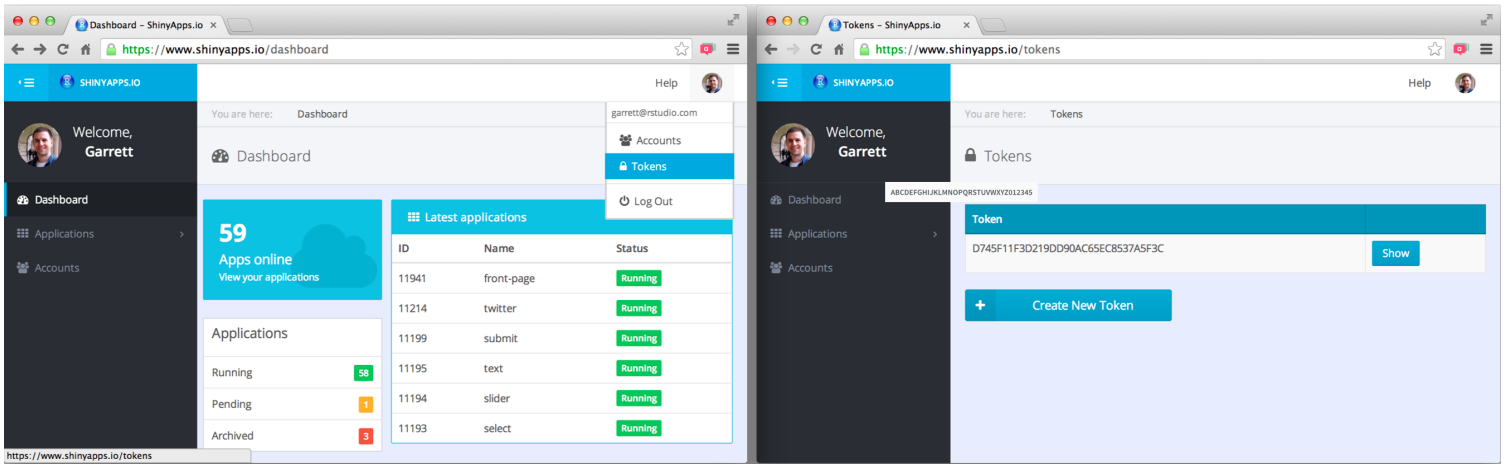
Create a ShinyApps.io account

Go to shinyapps.io (https://www.shinyapps.io) and click “Sign In.” The site will ask you to sign in using your Google Account.

The first time you sign in, ShinyApps.io prompts you to setup your account. ShinyApps.io uses the account name as the domain name for all your apps. Account names must be between four and 63 characters and can contain only letters, numbers, and dashes (-). Account names may not begin with a number or a dash, and they can not end with a dash (see RFC 952 (http://tools.ietf.org/html/rfc952)). Some account names may be reserved.

Configure `shinyapps`

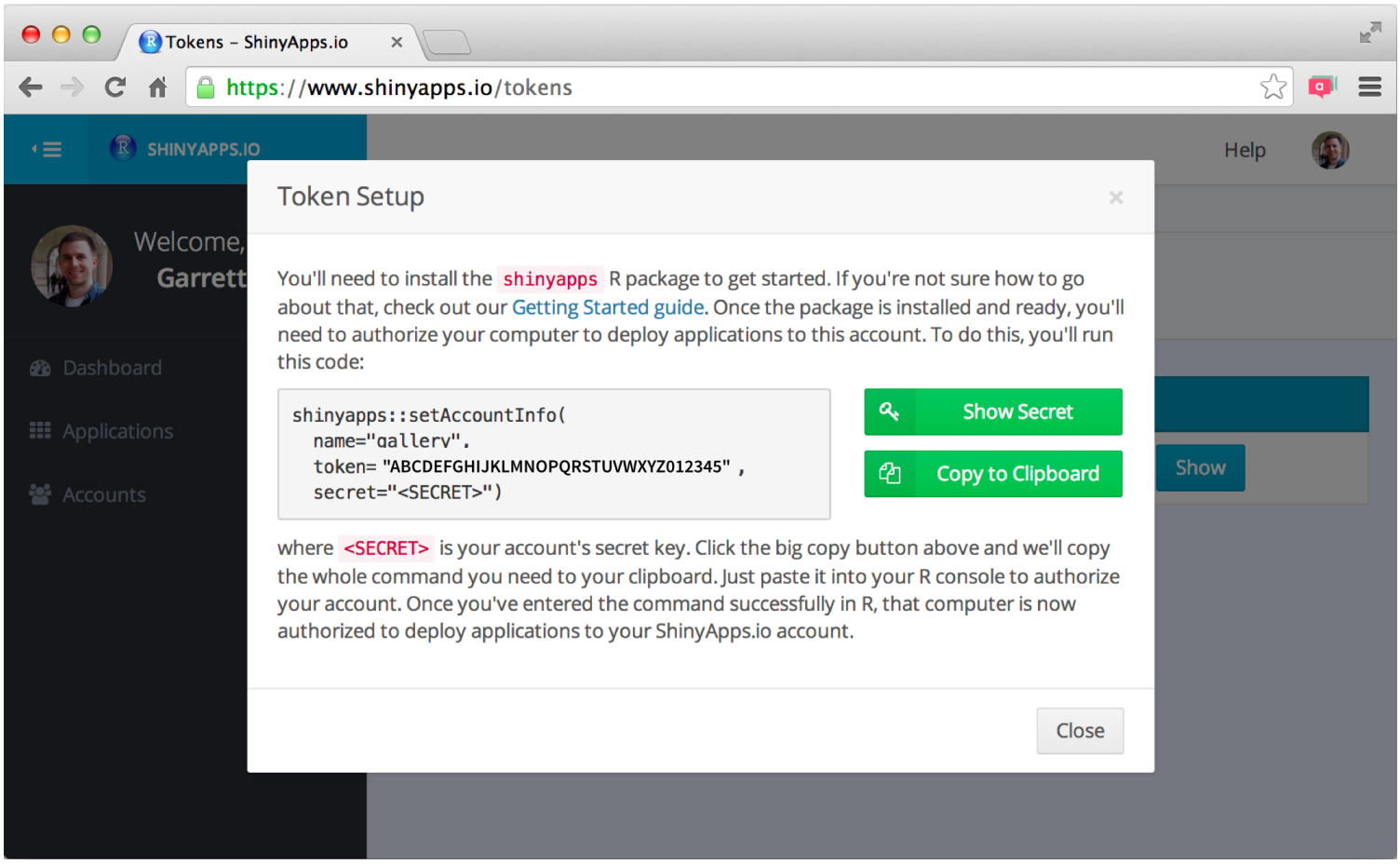
Once you set up your account in ShinyApps.io, you can configure the `shinyapps` package to use your account. ShinyApps.io automatically generates a token and secret for you, which the `shinyapps` package can use to access your account. Retrieve your token from the ShinyApps.io dashboard. Tokens are listed under `Tokens` in the menu at the top right of the ShinyApps dashboard (under your avatar).



You can configure the `shinyapps` package to use your account with two methods:

Method 1

Click the show button on the token page. A window will pop up that shows the full command to configure your account using the appropriate parameters for the `shinyapps::setAccountInfo` function. Copy this command to your clip board, and then paste it into the command line of RStudio and click enter.



Method 2

Run the 'setAccountInfo' function from the `shinyapps` package passing in the token and secret from the Profile / Tokens page.

```
shinyapps::setAccountInfo(name="<ACCOUNT>", token="<TOKEN>", secret="<SECRET>")
```

Once you have configured your `shinyapps` installation, you can use it to upload applications to ShinyApps.io. In the second part of this guide, we will build a demo application, upload it to ShinyApps.io, and create a password for the application.

A Demo app

For this guide, we created an RStudio project named "demo" that contains a Shiny application to upload to ShinyApps.io. Follow these steps to create your own Shiny app.

Install application dependencies

The demo application we will deploy requires the `ggplot2` package and the `shiny` package. Ensure that any package required by your application is installed locally before you deploy your application:

```
install.packages(c('ggplot2', 'shiny'))
```

ui.R and server.R

We placed two Shiny source files, `ui.R` and `server.R`, in our demo application. You can cut and paste the code below to make your own Shiny application:

server.R

```
library(shiny)
library(ggplot2)

shinyServer(function(input, output) {

  dataset <- reactive(function() {
    diamonds[sample(nrow(diamonds), input$sampleSize),]
  })

  output$plot <- reactivePlot(function() {

    p <- ggplot(dataset(), aes_string(x=input$x, y=input$y)) + geom_point()

    if (input$color != 'None')
      p <- p + aes_string(color=input$color)

    facets <- paste(input$facet_row, '~', input$facet_col)
    if (facets != '. ~ .')
      p <- p + facet_grid(facets)

    if (input$jitter)
      p <- p + geom_jitter()
    if (input$smooth)
      p <- p + geom_smooth()

    print(p)

  }, height=700)

})
```

ui.R

```
library(shiny)
library(ggplot2)

dataset <- diamonds

shinyUI(pageWithSidebar(

  headerPanel("Diamonds Explorer"),

  sidebarPanel(

    sliderInput('sampleSize', 'Sample Size', min=1, max=nrow(dataset),
                value=min(1000, nrow(dataset)), step=500, round=0),

    selectInput('x', 'X', names(dataset)),
    selectInput('y', 'Y', names(dataset), names(dataset)[[2]]),
    selectInput('color', 'Color', c('None', names(dataset))),

    checkboxInput('jitter', 'Jitter'),
    checkboxInput('smooth', 'Smooth'),

    selectInput('facet_row', 'Facet Row', c(None='.', names(dataset))),
    selectInput('facet_col', 'Facet Column', c(None='.', names(dataset)))
  ),

  mainPanel(
    plotOutput('plot')
  )
))
```

Test your application

Test that your application works by running it locally. Set your working directory (http://www.rstudio.com/ide/docs/using/workspaces) to your app directory, and then run:

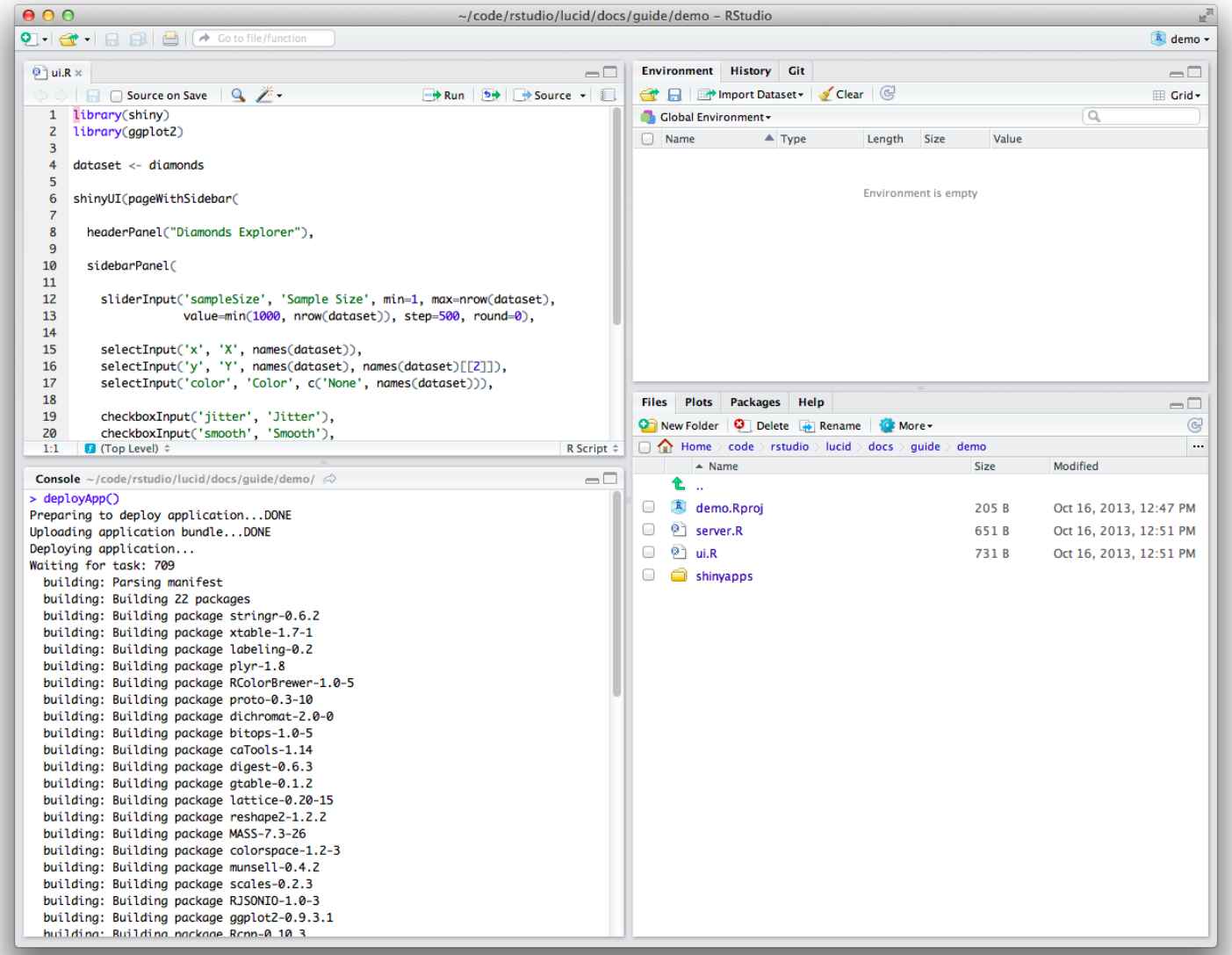
```
library(shiny)
runApp()
```

Now that the application works, let's upload it to ShinyApps.io.

Deploying apps

To deploy your application, use the `deployApp` command from the `shinyapps` packages.

```
library(shinyapps)
deployApp()
```



Once the deployment finishes, your browser should open automatically to your newly deployed application.

Congratulations! You've deployed your first application. :-)

Feel free to make changes to your code and run `deployApp` again. `shinyapps` can deploy an app much more quickly after the first deployment.

Package dependencies

When you deploy your application, the `shinyapps` package attempts to detect the packages that your application uses. `shinyapps` sends this list of packages and their dependencies along with your application to the ShinyApps.io service. Then ShinyApps.io builds and installs the packages into the R library for your application. The first time you deploy your application, it may take some time to build these packages (depending on how many packages are used). However, you will not wait for these packages to build during future deployments (unless you upgrade or downgrade a package).

For more information on application package dependencies, see the documentation for the shinyapps (https://github.com/rstudio/shinyapps) package.

Package sources

Currently the ShinyApps.io service supports deploying packages installed from CRAN and GitHub. Soon it will support BioConductor and R-Forge packages.

Important note on GitHub packages

Only packages installed from GitHub with `devtools::install_github` in version 1.4 (or later) of `devtools` are supported. Packages installed with an earlier version of `devtools` must be reinstalled before you can deploy your application. **If you get an error such as "PackageSourceError" when you attempt to deploy, check that you have installed any package from Github with `devtools` 1.4 or later.**

Application instances

ShinyApps.io hosts each app on its own virtualized server, called an instance. Each instance runs an identical copy of the code and packages that you deployed (called the image).

When you deploy an app, ShinyApps.io creates a new image with the updated code and packages, and starts one or more instances with the new image. If the app was previously deployed, ShinyApps.io shuts down and destroys the old instances. Consider a few implications of this arrangement:

1) **Data written by an application to the local filesystem of an instance will be lost when you re-deploy the app.** Additionally, the distributed nature of the ShinyApps.io platform means that instances may be shut down and re-created at any time for maintenance or to recover from server failures.

2) It is possible to have more than one instance of an application. This situation means that **multiple instances of an application do not share a local filesystem**. A file written to one instance will not be available to another instance.

ShinyApps.io limits the amount of system resources an instance can consume. The amount of resources available to an instance will depend on its type. The table below outlines the various instance types and how much memory is allowed. **By default, ShinyApps.io deploys all applications on 'small' instances, which are allowed to use 256 MB of memory.**

Instance Type	Memory
small (default)	256 MB
medium	512 MB

large	1024 MB
xlarge	2048 MB
xxlarge	4096 MB

Note: Instance types and limits are not finalized; RStudio may change them in the future.

Configuring applications

You can change the instance type used by an application with the `configureApp` function from the `shinyapps` package. To change the instance type of your application (here from small to medium), run:

```
shinyapps::configureApp("APPNAME", size="medium")
```

This change will redeploy your application using the medium instance type.

Application authentication (passwords)

With ShinyApps.io, you can limit the access to your application by configuring authentication. ShinyApps.io will prompt each visitor to your app for a username and password when authentification is enabled.Only users who log-in with valid credentials will be able to view or use the app. ShinyApps.io enables authentication automatically when you add the first authorized user to an app.

To get started, you need to prepare your system to build native packages. Before continuing, please ensure you installed these tools for your system:

- Windows: RTools (http://cran.r-project.org/bin/windows/Rtools/)
- Mac OSX: XCode Command Line Tools
- Linux: GCC

You will also need the `scrypt` package used for encrypting passwords.To install the `scrypt` package using `devtools`:

```
devtools::install_github('rstudio/rscrypt')
```

You should restart your R session at this point, if you already had the `shinyapps` package installed. **Note: that authentification requires `shinyapps` version 0.3 or greater**.

Add an authorized user

The `shinyapps` package provides a number of functions for managing an application's authorized users. Before you use them, change your working directory to where your application code resides:

```
setwd("/path/to/my/shiny/app")
```

To add an authorized user, use the `shinyapps::addAuthorizedUser` function. The code below will add an authorized user whose username is “andy”.

```
addAuthorizedUser("andy")
```

After you run `addAuthorizedUser`, R will prompt you to enter a password for the user. Passwords must be at least four characters in length, and may not contain: `\t`, `\n`, `$` or `:`.

Please remember that passwords are stored using a scrypt (http://en.wikipedia.org/wiki/Scrypt) hash for security. Once stored, passwords can not be retrieved (but they can always be reset).

You can add multiple authorized users to an app by running `addAuthorizedUser` multiple times.

After adding or removing a user, you need to deploy your application using the `shinyapps::deployApp` function.

```
deployApp()
```

That’s it. ShinyApps.io will now prompt your users for a username and password when they visit your app.

If a user forgets his or her password, you can reset it using the `shinyapps::addAuthorizedUser` function. You will be prompted to confirm you want to reset the user’s password.

Remove an authorized user

To remove an authorized user, use the `shinyapps::removeAuthorizedUser` function.

```
removeAuthorizedUser("andy")
```

The remember to re-deploy your app with `shinyapps::deployApp`.

Terminate an app

You can remove an app on ShinyApps.io from the web with the `terminateApp` command. To use it, run

```
terminateApp("<your app's name>")
```

`terminateApp` requires one argument, the name of the app that you would like to terminate (as a character string). This name should correspond with one of the apps in your ShinyApps.io account.

When you run `terminateApp` ShinyApps.io will close your app, but the app will remain archived in your ShinyApps.io account. This creates efficiencies if you later decide to redeploy your app with `deployApp`.

Getting help

To seek and share advice about ShinyApps.io, please visit the ShinyApps google group (https://groups.google.com/forum/?hl=en#!forum/shinyapps-users).

Recap

ShinyApps.io is an online service for hosting Shiny apps in the cloud. RStudio takes care of all of the details of hosting the app and maintaining the server, which lets you focus on writing great apps!

To use ShinyApps.io

- Install the shinyapps (https://github.com/rstudio/shinyapps) R package from github
- Create an account at shinyapps.io (https://www.shinyapps.io)
- Use the tokens generated by ShinyApps.io to configure your `shinyapps` package.
- Deploy apps with `shinyapps::deployApp`
- Terminate apps with `shinyapps::terminateApp`

You can also use ShinyApps.io to create password protected apps, and manage your authorized users.

I'm not sure what went wrong, but I've compiled some links that can help you find someone who would know here. In general, I'd like to avoid using the comment section as a support forum.

Reply • Share ›

Macchiato Macc • 12 days ago

How long does it normally take to deploy an application? I've been waiting about 20 minutes so far. There's just "rollforward: Activating new instances" in R studio console.

Reply • Share ›

Jeff Allen ➦ Macchiato Macc • 11 days ago

It depends on the number of packages you take a dependency on, and the first time will always take longer than subsequent builds. But on average, we'd expect 3-5 minutes for the first build; certainly not twenty. Go ahead and post a message on the mailing list (https://groups.google.com/foru... with the URL or task/app ID and we'll be able to investigate.

Reply • Share ›

singco • 14 days ago

Hi, I try to authorize my account, and I paste the code in my R console, it tells me this: error: /v1/users/current/ 403 - bad signature, how can I get through? thanks

Reply • Share ›

Dave Guarino ➦ singco • 12 days ago

Hey Singco,

I got this error too — try clicking the "Copy to clipboard" button and then pasting that into R.

(If you were actually highlighting the code snippet there, like I did at first, it gives you that error because of the line breaks and spaces.)

Cheers!
Dave

Reply • Share ›

Garrett Mod ➦ singco • 13 days ago

singco, it sounds like you might have an incorrect token. We're also working on a known bug for non-english locales. The best place to get help for technical issues with ShinyApps.io is https://groups.google.com/forum/#!forum/shinyapps-users

Reply • Share ›

Raymond • 16 days ago

Hi, how do I remove an app from Shinyapps.io once deployed? Thank you.

Regards, Raymond

• Reply • Share ›

Garrett Mod ➚ Raymond • 15 days ago

"You can remove an app on ShinyApps.io from the web with the `terminateApp` command. To use it, run `terminateApp("your app's name")`."

• Reply • Share ›

Raymond • a month ago

Hi, I had deployed my app using your sample code showing the map of 2010 US census data. I can run the app on my R desktop but I got the error: could not find function "mapproject" in shinyapps.io. I had included the library calls to shiny and maps within my ui.R and server.R. What did I do wrong? Thanks!

Regards, Raymond

• Reply • Share ›

Raymond ➚ Raymond • a month ago

Sorry, found my mistake... should had included library mapproj as well.

By the way, should we include these libraries in ui.R or server.R or either one? Thanks!

• Reply • Share ›

Garrett Mod ➚ Raymond • 24 days ago

You should put them in server.R. They set the server/your computer up to run the rest of the code in server.R.

• Reply • Share ›