

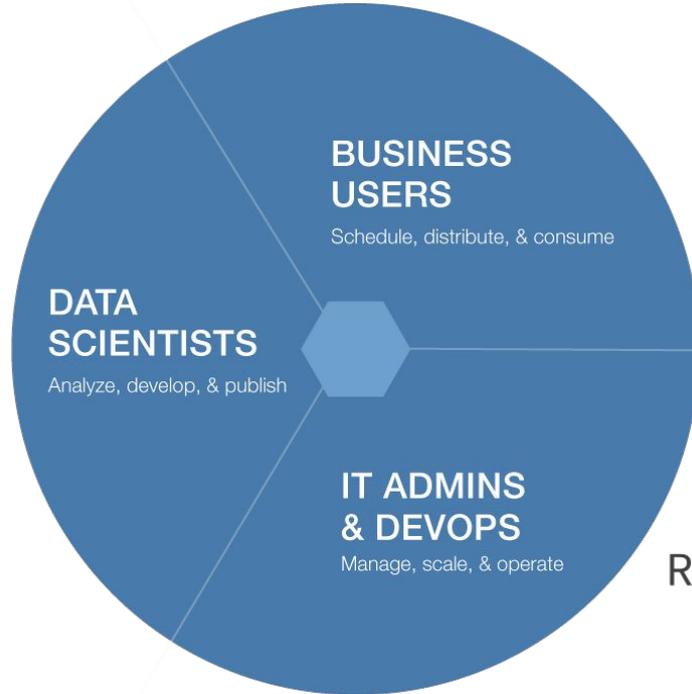
Art of the Feature Toggle

Patterns for maintaining and improving Shiny applications over time



Solutions Engineering at RStudio


RStudio Server Pro




RStudio Connect


RStudio Package Manager

Shiny in Production Workshop - 2019



rstudio::conf 2019

1 Shiny in Production Workshop

2 Introduction to Shiny in Production

2.1 Workshop Objectives

2.2 Workshop Infrastructure

3 Introduction to the Application

3.1 Activity: Explore the Application

4 Application Testing: shinytest

4.1 Testing Options

4.2 Activity: shinytest

5 Profiling: "The most important thing"

5.1 Activity: Profiling

6 Deployment

6.1 RStudio Connect

6.2 Activity: Initial Deployment

6.3 Application Settings

7 Connecting to Data in Production

7.1 The config package

7.2 Environment Variables

7.3 Activity: Databases

Supplement to Shiny in Production

2019-01-15

Chapter 1 Shiny in Production Workshop



rstudio::conf 2019

This document is full of supplemental resources and content from the Shiny in Production Workshop delivered at rstudio::conf 2019.



Chapter 12 - DevOps Philosophy and Tooling

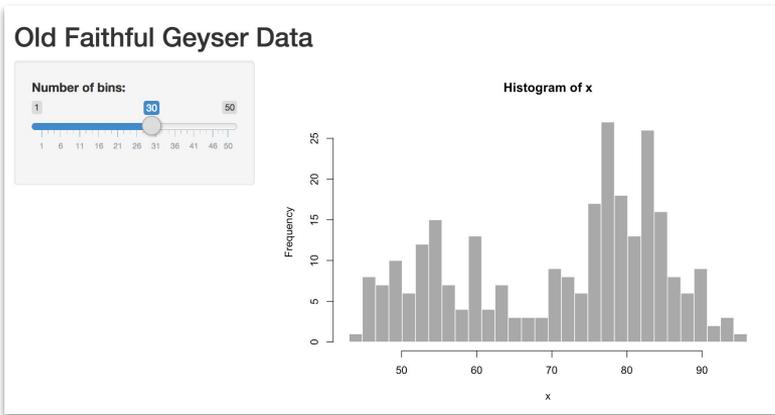
- “Production” has many meanings
- Are there lessons can be learned from the DevOps community?

[Bookdown: Shiny in Production](#) (RStudio Conference 2019)

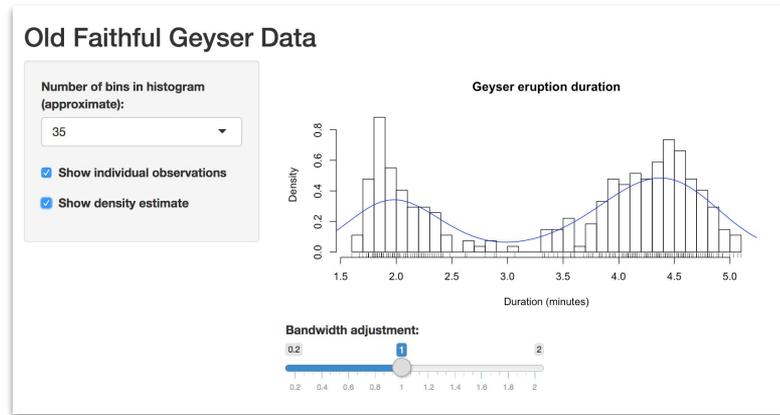
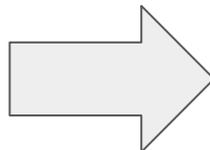
Introducing new features over time

Shiny Applications in Production

Adding new features to Shiny in Production

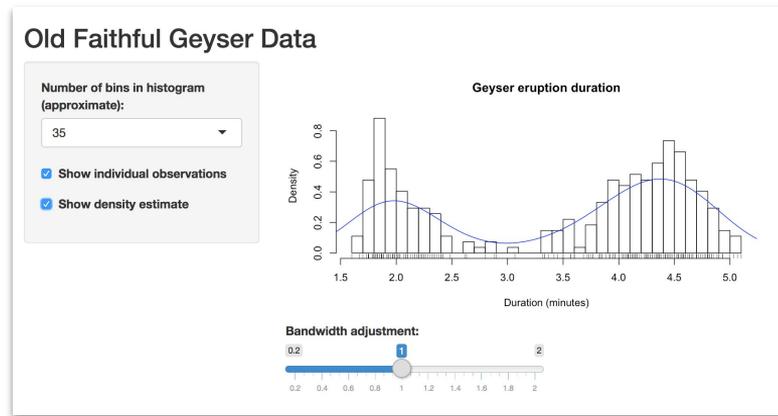
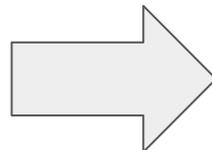
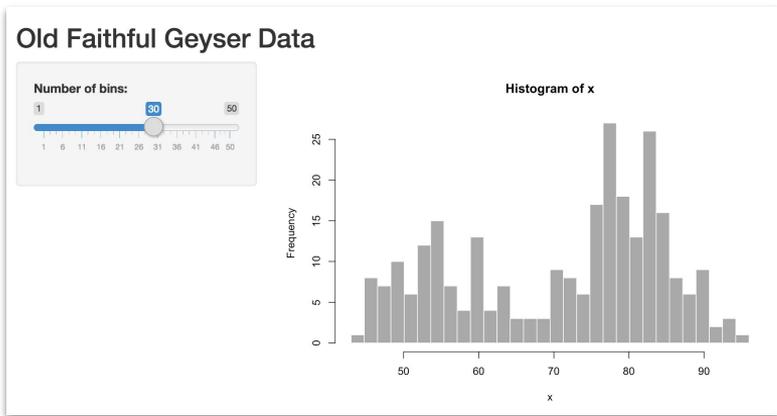


Version 1

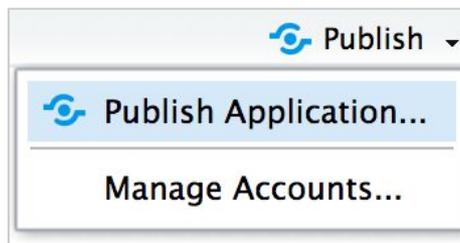


Version 2

Adding new features to Shiny in Production

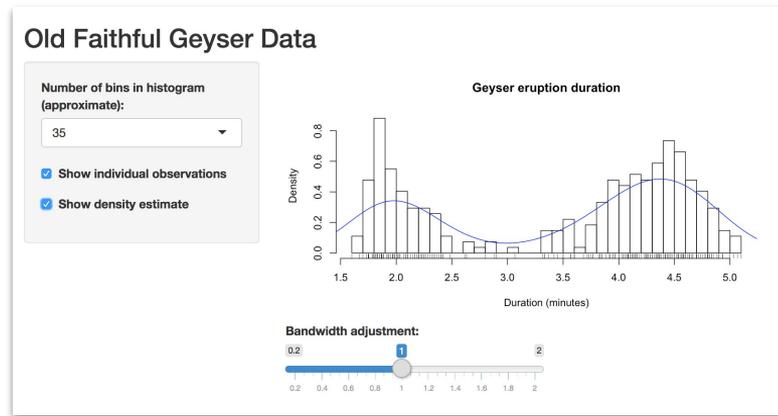
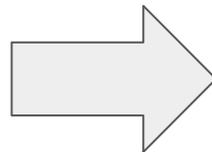
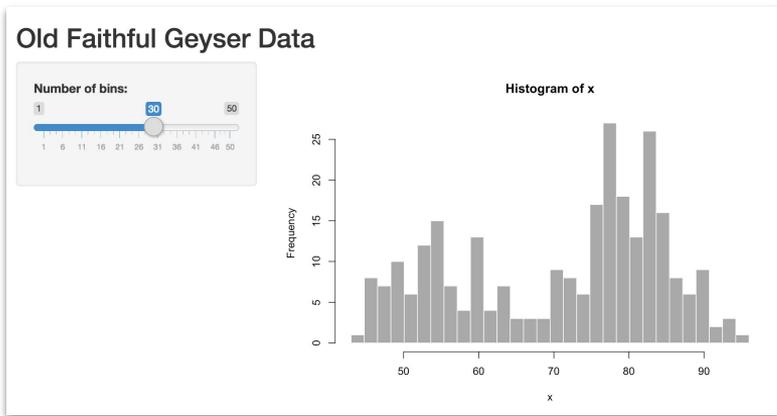


Version 1



Version 2

Adding new features to Shiny in Production



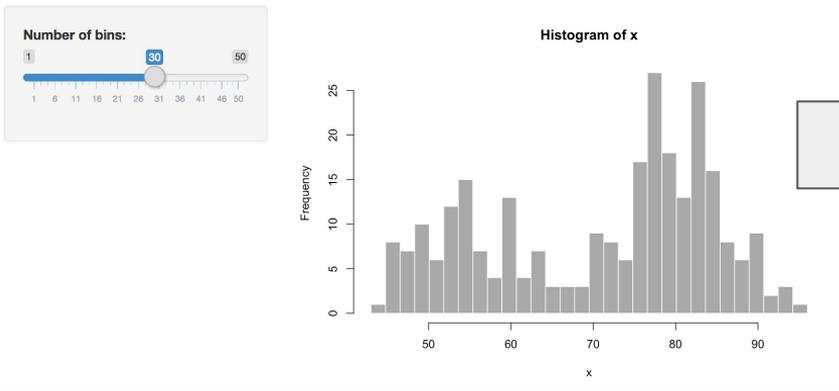
Version 1

Version 2



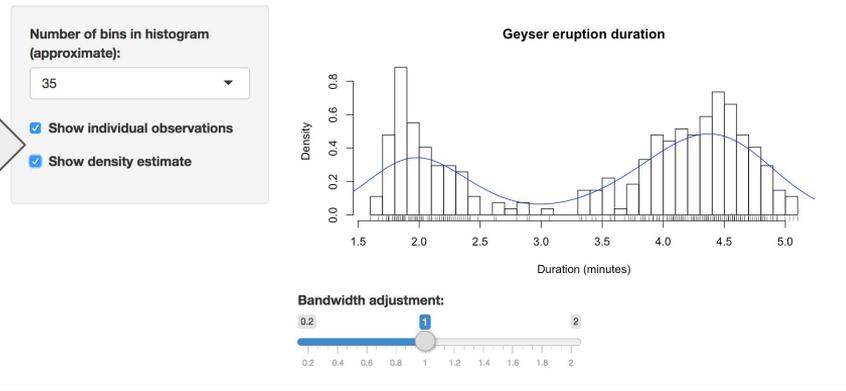


Old Faithful Geyser Data



Version 1

Old Faithful Geyser Data



Version 2

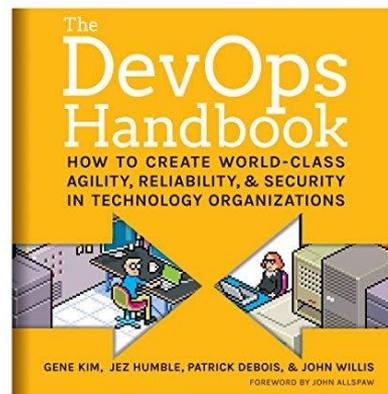
Does pushing a code change feel risky?

DevOps Learning: Decouple **deployment** from **release**

- **Deployment** is any push of code to an environment (test, prod)
- **Release** is when that code (feature) is made available to users or customers

Deployment on demand and thoughtful release strategies allow more control (and more success) over the delivery of features to end users.

- Application-based release patterns (today!)
- Environment-based release patterns (tomorrow)



Application-Based Release Patterns

Feature Toggle



A mechanism to selectively enable and disable features, or control which features are visible to specific user segments.

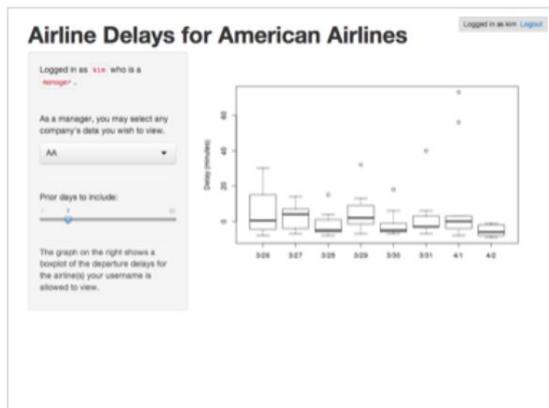
Enable **dark launch**: deploy a change to production and then perform testing while it's invisible to most users.

- Modify the code to make calls to new functions, log results without displaying
- Have 1% of users make invisible calls to new feature, fix issues, progressively increase users to test production load

Starting from the Gallery

Professional Features

These examples demonstrate some of the unique features of RStudio Connect.



Authentication and database

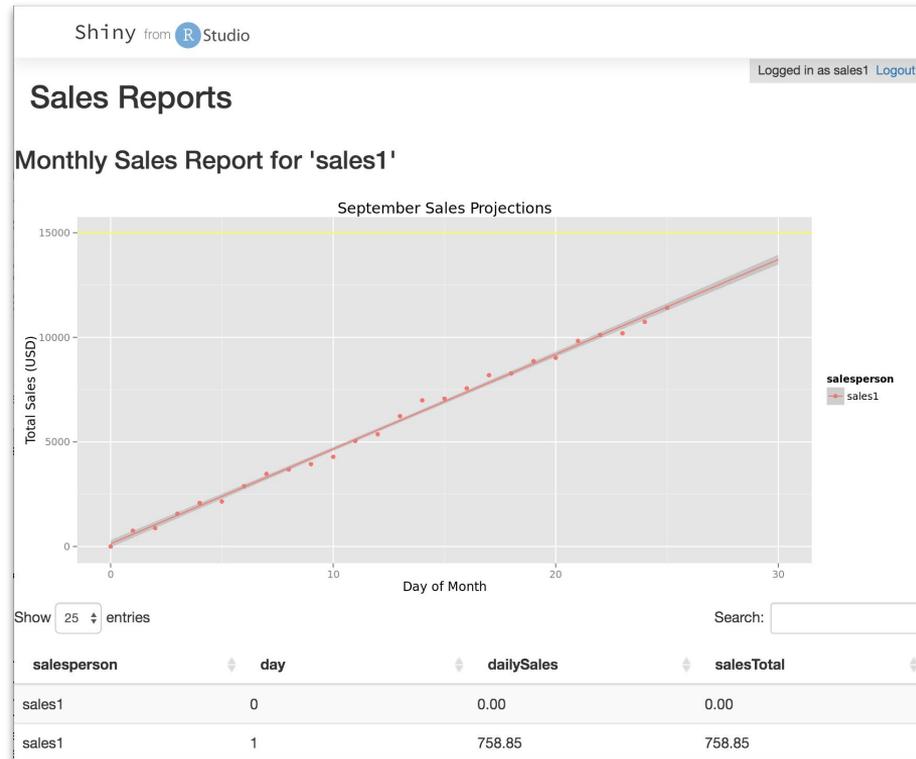
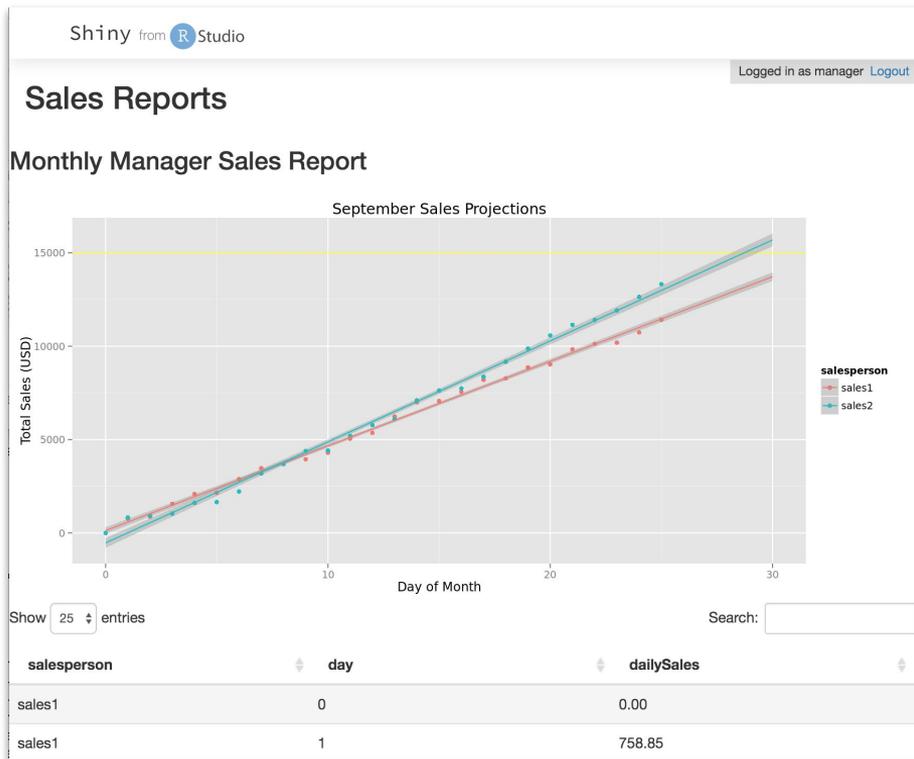


Personalized UI

Smells like feature toggles!



“Personalized Data Access” `session$user`



Feature Toggles

Shiny applications can access the username and groups of the current user through the session parameter of the shinyServer function.

Your application could use this information to display customized messages or to enable functionality for a specific subset of users.

```
shinyServer(function(input, output, session) {  
  output$username <- reactive({  
    session$user  
  })  
  
  output$groups <- reactive({  
    session$groups  
  })  
})
```

Implementation

```
# Render the subtitle of the page according to what user is logged in.
output$subtitle <- renderText({
  if (is.null(user())){
    return("You must log in to use this application using Shiny Server Pro.")
  }

  if (isManager()){
    return("Monthly Manager Sales Report")
  } else {
    return(paste0("Monthly Sales Report for '", user(), "'"))
  }
})
```

Implementation

UI built through conditional `session$user` tests within server-side output objects

```
output$salesTbl <- renderDataTable({  
  # If no user is logged in, don't show any data.  
  if (is.null(user())){  
    return()  
  }  
  
  # Otherwise return all data that should be visible to this user.  
  myData()  
})
```

`output$subtitle` `output$salesPlot` `output$salesTbl`

Dynamic UI Feature Toggles

The conditionalPanel function

The `conditionalPanel` function, is used in `ui.R` and wraps a set of UI elements that need to be dynamically shown/hidden.

Creates a panel that shows and hides its contents depending on the value of a JavaScript expression, usually input-based.

This example shows how to define an output variable in the server code that you can use in the UI. →

`session$user` `session$groups`

```
Branch: master | advanced-shiny / server-to-ui-variable / app.R
daattali add many readmes and a few more examples
1 contributor

18 lines (15 sloc) | 391 Bytes

1 library(shiny)
2
3 ui <- fluidPage(
4   selectInput("num", "Choose a number", 1:10),
5   conditionalPanel(
6     condition = "output.square",
7     "That's a perfect square!"
8   )
9 )
10
11 server <- function(input, output, session) {
12   output$square <- reactive({
13     sqrt(as.numeric(input$num)) %% 1 == 0
14   })
15   outputOptions(output, 'square', suspendWhenHidden = FALSE)
16 }
17
18 shinyApp(ui = ui, server = server)
```

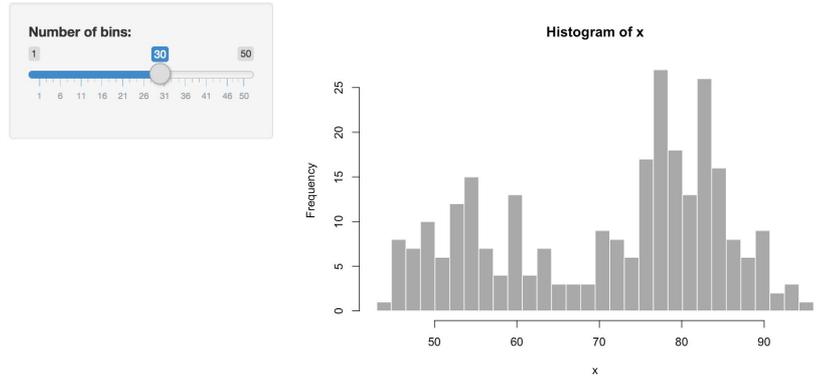
[Output condition example: Dean Attali](#)

```
output$isBeta <- reactive({
  [Test for group membership]
})
```

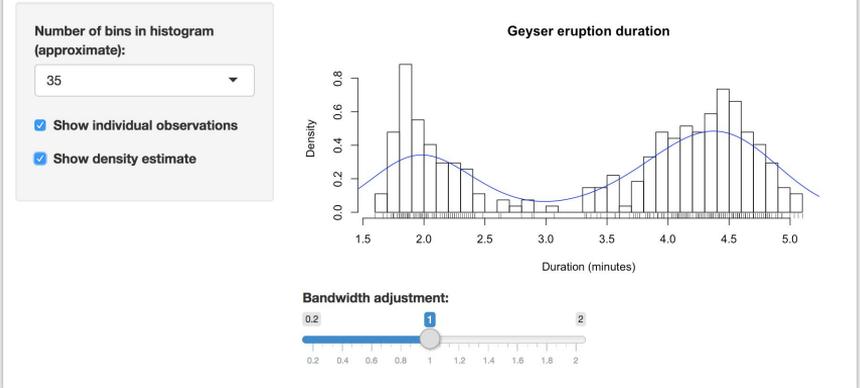
```
conditionalPanel(
  condition = "output.isBeta ==
false", ... UI Elements)
```

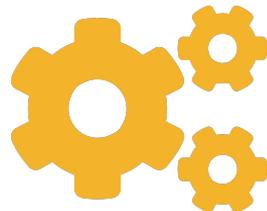
```
conditionalPanel(
  condition = "output.isBeta == true",
  ... UI Elements)
```

Old Faithful Geyser Data



Old Faithful Geyser Data





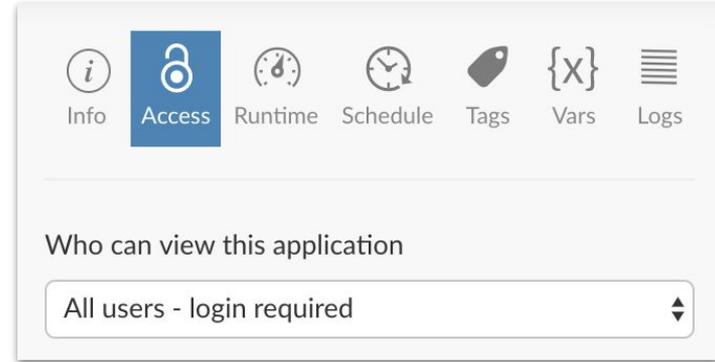
Feature Toggles in Production?

Will I (personally) use feature toggles for Shiny application deployment and releases? **No, not likely.**

- Not easy to manage, automate, test and ultimately “toggle”
 - Alternative: “Environment-Based Release Patterns”
- Still a cool concept, useful for solving other types of problems
 - RStudio Connect Feature Hacks!

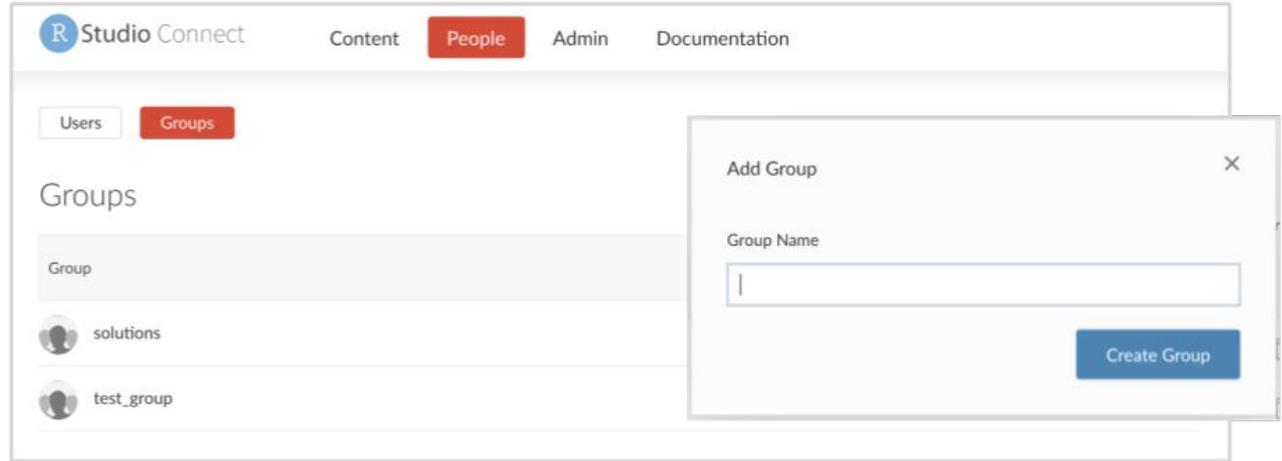
Mixing Access Controls & Feature Toggles on RStudio Connect

Content **Access** Control Options on RStudio Connect

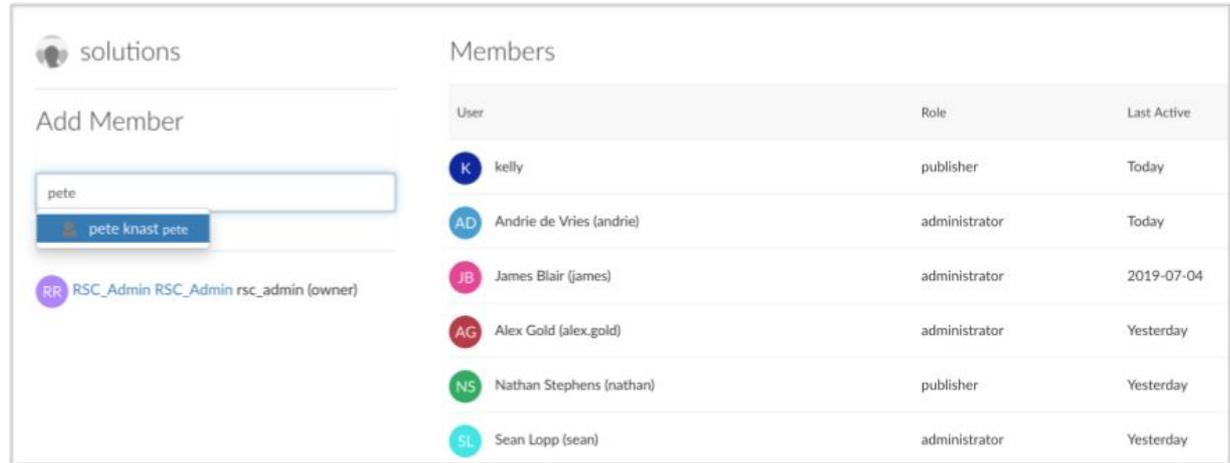


Access Control Setting	Description
Anyone, no login required	Allows for anonymous viewer access
All users, login required	Everyone with an RStudio Connect account
Specific users or groups	Only users or groups you specify can access
Collaborators & you	Only you and your collaborators can access

People > Group Management UI in RStudio Connect (Admin Tools)



Solutions Engineering group setup



Problem: Can you add me as a collaborator on this?

1. Self service code access

```
- install.packages("gitlink")
```

2. Slack Incoming Webhook integration

```
If session$groups != solutions {  
  Display an action button that will  
  send me a slack alert when clicked.  
}
```

The screenshot shows a user interface for managing application access. At the top right, there is a user profile for 'kelly' and a settings gear icon. Below this is a navigation bar with icons for 'Info', 'Access' (which is highlighted in blue), 'Runtime', 'Schedule', 'Tags', 'Vars', and 'Logs'. Below the navigation bar, there are two sections: 'Who can view this application' with a dropdown menu set to 'All users - login required', and 'Who can change this application' with a list of users including 'kelly' and 'solutions'. At the bottom, there is a button labeled 'Add collaborator'.

“Slack me if you need access”

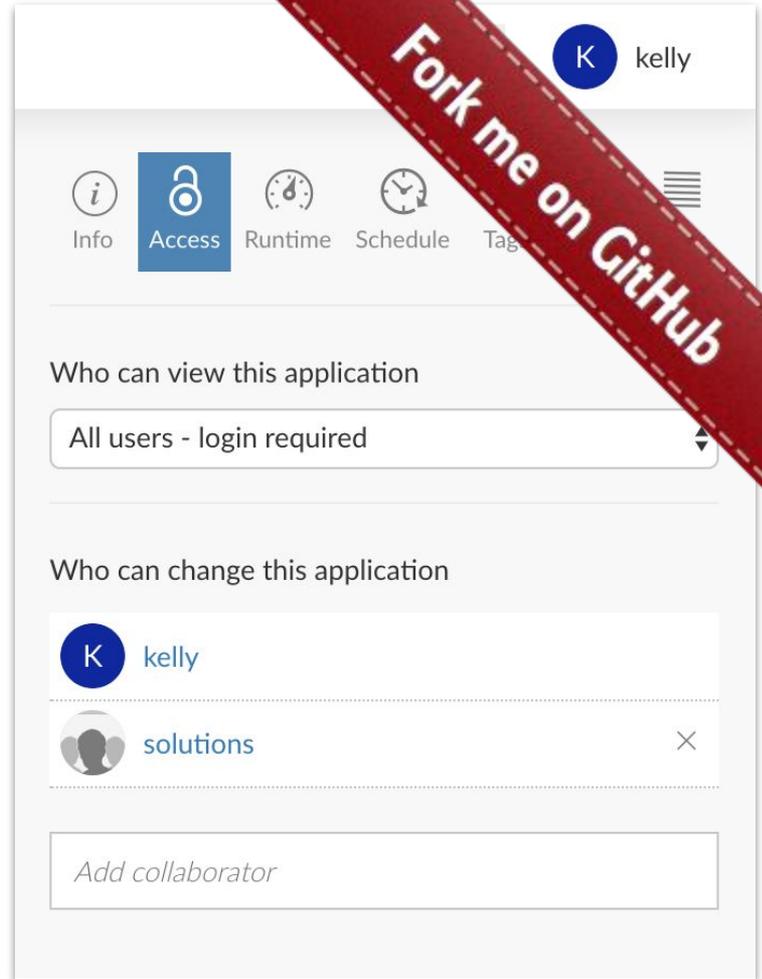
Problem: Can you add me as a collaborator on this?

1. [Self service code access](#)

```
- install.packages("gitlink")
```

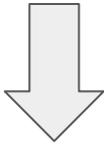
2. Slack Incoming Webhook integration

```
If session$groups != solutions {  
  Display an action button that will  
  send me a slack alert when clicked.  
}
```

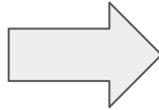


“Slack me if you need (collaborator) access”

```
conditionalPanel(  
  condition = "output.isCollab == false",  
  actionButton("request", "Request Collaborator Access")  
)
```



Request Collaborator Access



colorado.rstudio.com says

"Your Collaboration Request Has Been Sent."

OK

Action Button - Command Pattern

Pattern 1 - Command

Use `observeEvent()` to trigger a command with an action button.

Example

Click Me

Action Button - Pattern 1 by RStudio, Inc.

```
app.R www ▾ ↑ show with app

library(shiny)

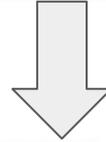
ui <- fluidPage(
  tags$head(tags$script(src = "message-handler.js")),
  actionButton("do", "Click Me")
)

server <- function(input, output, session) {
  observeEvent(input$do, {
    session$sendCustomMessage(type = 'testmessage',
      message = 'Thank you for clicking!')
  })
}

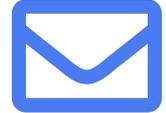
shinyApp(ui, server)
```

Code license: MIT

Request Collaborator Access



slack



colorado.rstudio.com says

"Your Collaboration Request Has Been Sent."

OK

Recommendation: [Create a Slack App](#)



There are a number of R packages and legacy methods you could use to talk to slack from your R code.

I recommend creating a Slack App.

Send data to Slack in real-time

- [Incoming Webhook](#)

Creating an Incoming Webhook gives you a unique URL to which you send a [JSON](#) payload with the message text and some options.

Create a Slack App ✕

App Name

Don't worry; you'll be able to change this later.

Development Slack Workspace

Your app belongs to this workspace—leaving this workspace will remove your ability to manage this app. Unfortunately, this can't be changed later.

By creating a Web API Application, you agree to the [Slack API Terms of Service](#).

Add features and functionality

Choose and configure the tools you'll need to create your app (or review all [our documentation](#)).

Incoming Webhooks

Post messages from external sources into Slack.

Interactive Components

Add buttons to your app's messages, and create an interactive experience for users.

Slash Commands

Allow users to perform app actions by typing commands in Slack.

Event Subscriptions

Make it easy for your app to respond to activity in Slack.

Bots

Add a bot to allow users to exchange messages with your app.

Permissions

Configure permissions to allow your app to interact with the Slack API.

Install your app to your workspace

Install your app to your Slack workspace to test your app and generate the tokens you need to interact with the Slack API. You will be asked to authorize this app after clicking **Install App to Workspace**.

Reinstall App

App Name



Connect Collaborate Notifier

Display Information

This information will be shown in the Slack App Directory and in the Slack App. For more information, view our [App Detail Guidelines](#).

App name

Connect Collaborate |

Short description

A Webhook for Collaboration Requests on RStudio Cor

App icon & Preview



Connect Collaborate Notifier APP

A Webhook for Collaboration Requests on RStudio Connect Shiny Apps

Background color

 #004492

“Slack `_me_` if you need access”

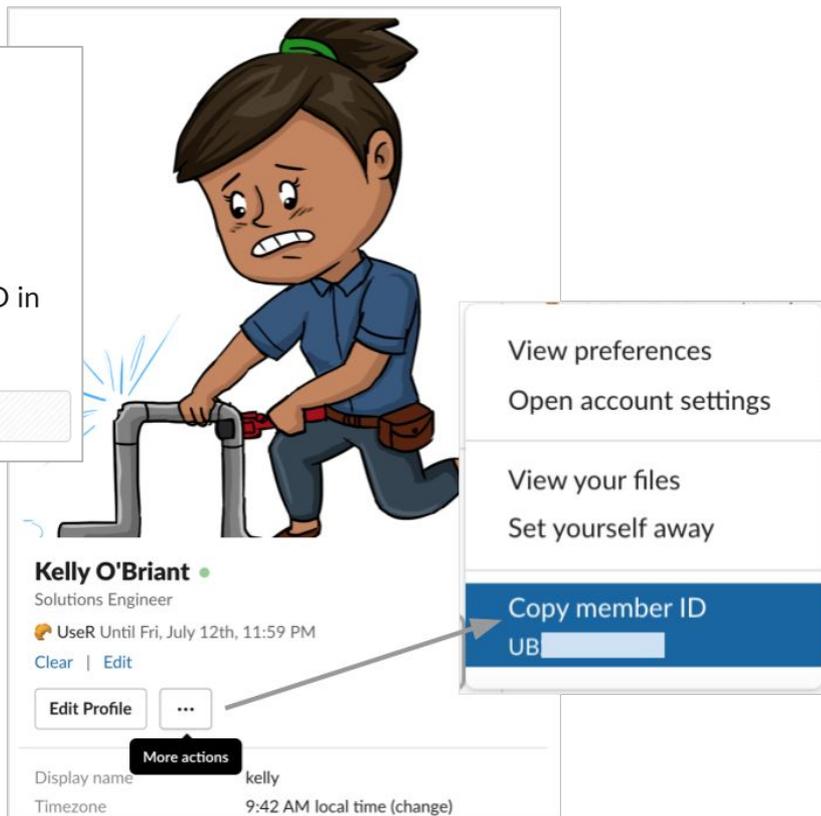
Mentioning users

A [mention](#) is a special type of reference that will provide a link to the mentioned user's profile, and also *notify* them about the reference. The [help center page for mentions](#) describes what that notification process looks like.

To mention a user in an app-published message, you need to provide their user ID in the following syntax:

```
Hey <@U024BE7LH>, thanks for submitting your report.
```

- Find your slack member ID
- Construct your message
- Send JSON payload to Webhook URL



The image shows a Slack user profile for Kelly O'Briant, a Solutions Engineer. The profile includes a header with the name and role, a status indicator (green dot), and a message: "UseR Until Fri, July 12th, 11:59 PM". Below the message are "Clear" and "Edit" buttons. A "More actions" menu is open, showing options: "Edit Profile", "View preferences", "Open account settings", "View your files", "Set yourself away", and "Copy member ID". The "Copy member ID" option is highlighted in blue, and an arrow points to it from the text box above. The user's display name is "kelly" and their timezone is "9:42 AM local time (change)".

Action Button - Command Pattern

```
conditionalPanel(  
  condition = "output.isCollab == false",  
  actionButton("request", "Request Collaborator Access")  
)
```

ui.R

```
observeEvent(input$request, {  
  collab_alert <- glue('{{"text":"Hey <{slack_owner_id}>, {session$user} is requesting collaborator access to {vanity_url}"}}')  
  
  POST(slack_webhook, body = collab_alert, add_headers('Content-Type' = 'application/json'))  
  
  session$sendCustomMessage(type = 'confirm-request',  
    message = 'Your Collaboration Request Has Been Sent.')  
})
```

server.R

```
// This recieves messages of type "confirm-request" from the server.  
Shiny.addCustomMessageHandler("confirm-request",  
  function(message) {  
    alert(JSON.stringify(message));  
  }  
);
```

message-handler.js

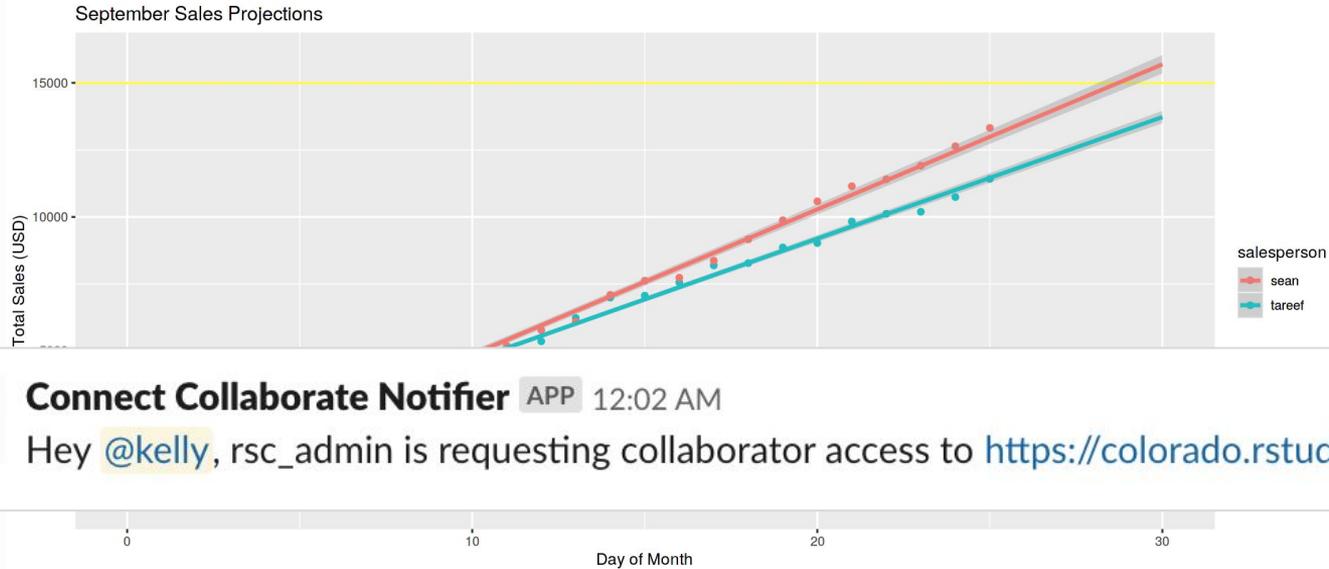
Sales Reports

Fork me on GitHub



Monthly Sales Report

Request Collaborator Access



Connect Collaborate Notifier APP 12:02 AM

Hey @kelly, rsc_admin is requesting collaborator access to <https://colorado.rstudio.com/rsc/collab-notify/>

Access Control Setting	Group or User Conditional	Solution
All users, login required	Group isSolutions() FALSE	Element for requesting collaborator access

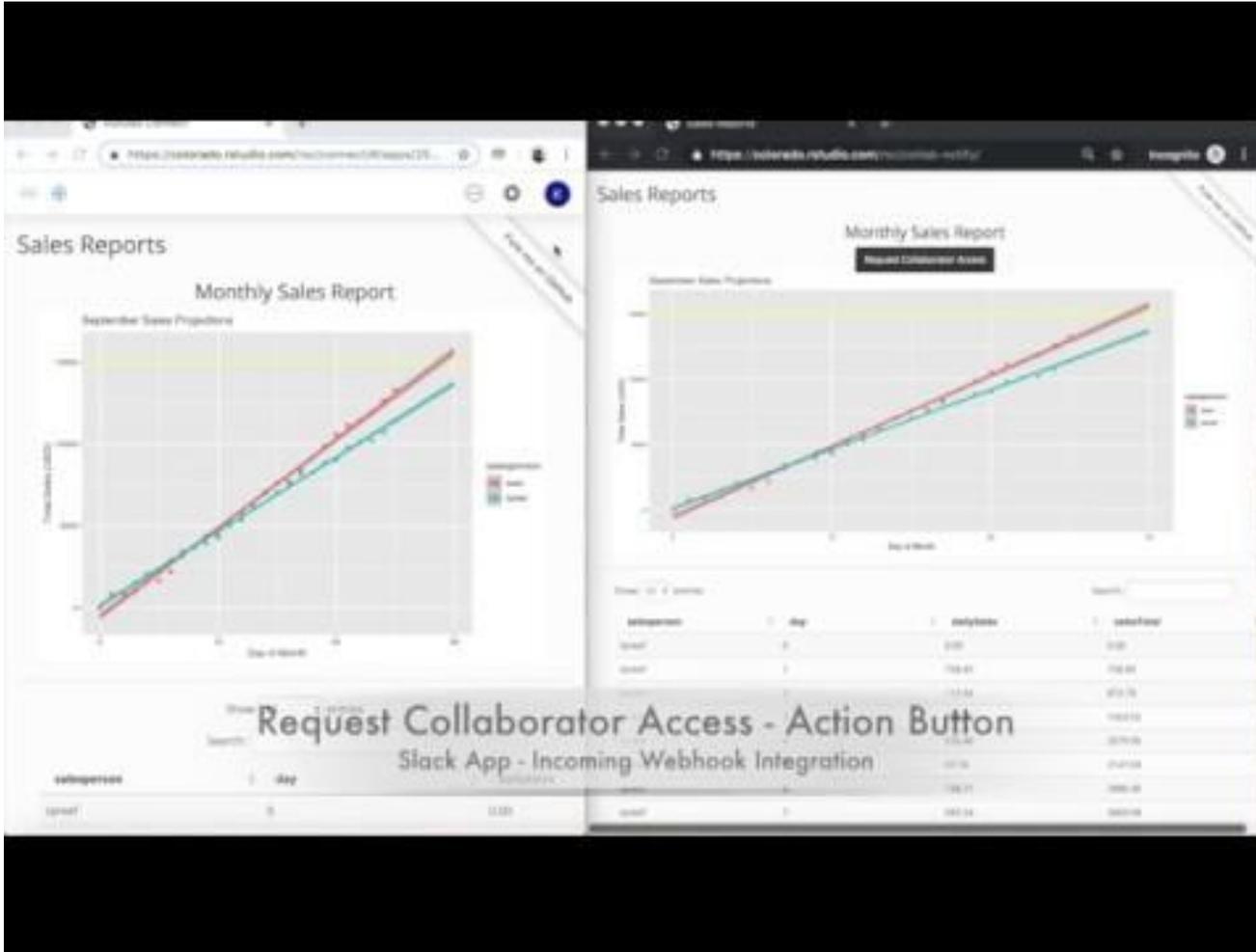
Left:

Publisher view with access controls on RStudio Connect

Signed in as Kelly, who is a member of solutions

No button displayed

[Watch on YouTube](#)



Right:

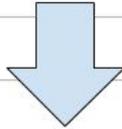
Vanity URL app access by a different user

User is not a member of the solutions group

Request Access button displayed

You are not allowed to access this content because you are not logged in.

Log In



Old Faithful Geyser Data

This application is hosted on our internal RStudio Connect server

Contact the Data Science team to request access if you do not have an account.

Request Access

You must be logged in to see this application.

Return to RStudio Connect

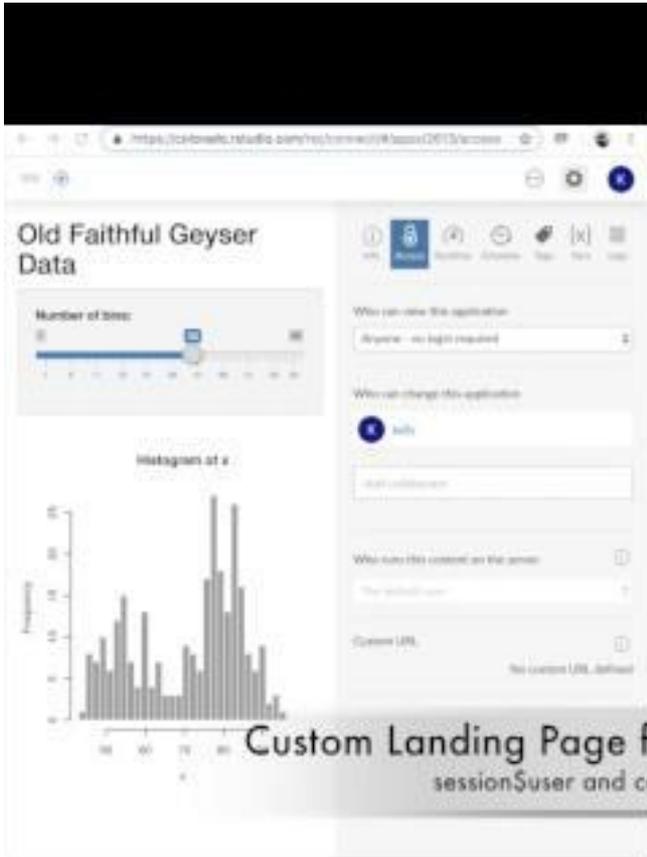


More Hacks for session\$user

Create a custom landing page for anonymous visitors and logged out users

Access Control Setting	Group or User Conditional	Solution
Anyone, no login required	is.null(user()) TRUE	Custom UI for anonymous viewers

Left:
Publisher view
with access
controls on
RStudio Connect



The left panel shows the publisher view for 'Old Faithful Geyser Data'. It includes a 'Number of tiles' slider, a histogram titled 'Histogram of x', and a sidebar with access controls. The sidebar contains sections for 'Who can view this application' (set to 'Anyone - no login required'), 'Who can change this application' (set to 'Self'), 'Who can edit the content on this panel' (set to 'The default user'), and 'Custom LURL' (set to 'No custom LURL defined').

Custom Landing Page for Anonymous Viewers
sessionUser and conditionalPanel() UI

Right:
Custom UI for
anonymous and
logged-out visitors
to the app

[Watch on
YouTube](#)



Code:

github.com/kellobri/shiny-feature-toggle



Slides: **bit.ly/shiny-feature-toggle**

- solutions.rstudio.com
- community.rstudio.com
- kelly@rstudio.com

