Climate App

Build an interactive application that explores climate projection data from CMIP5[[1]](#footnote-1) and the Norwegian Climate Centre NorESM1-M Climate model.

Make a copy of the ClimateApp folder. This folder contains the base files (ui.R, server.R, and R project file) that you will use to create an application in shiny.

# Step 1 – Read in data and load packages

Create a new script in the app/project directory called “global.R”. You can load any packages or data files in this file that can be used in your ui.R and server.R scripts.

In the gloabal.R script, you will want to load the shiny package, and the following data files to start:

load('data/snoteldata.Rdata')

load('data/prcp\_proj.Rdata')

These are datasets with observed SNOTEL data and projected precipitation (inches/day). Projected temperature data files (degrees Celsius) are also included for additional exploration/customization

# Step 2 – Add user input control widget for the site selection

Add the code allowing the user to select the site via a dropdown menu (selectInput) to the sidebarPanel (ui.R):

selectInput(inputId='site',

label=’Choose SNOTEL Site’,

choices=unique(snoteldata$Station),

selected = NULL,

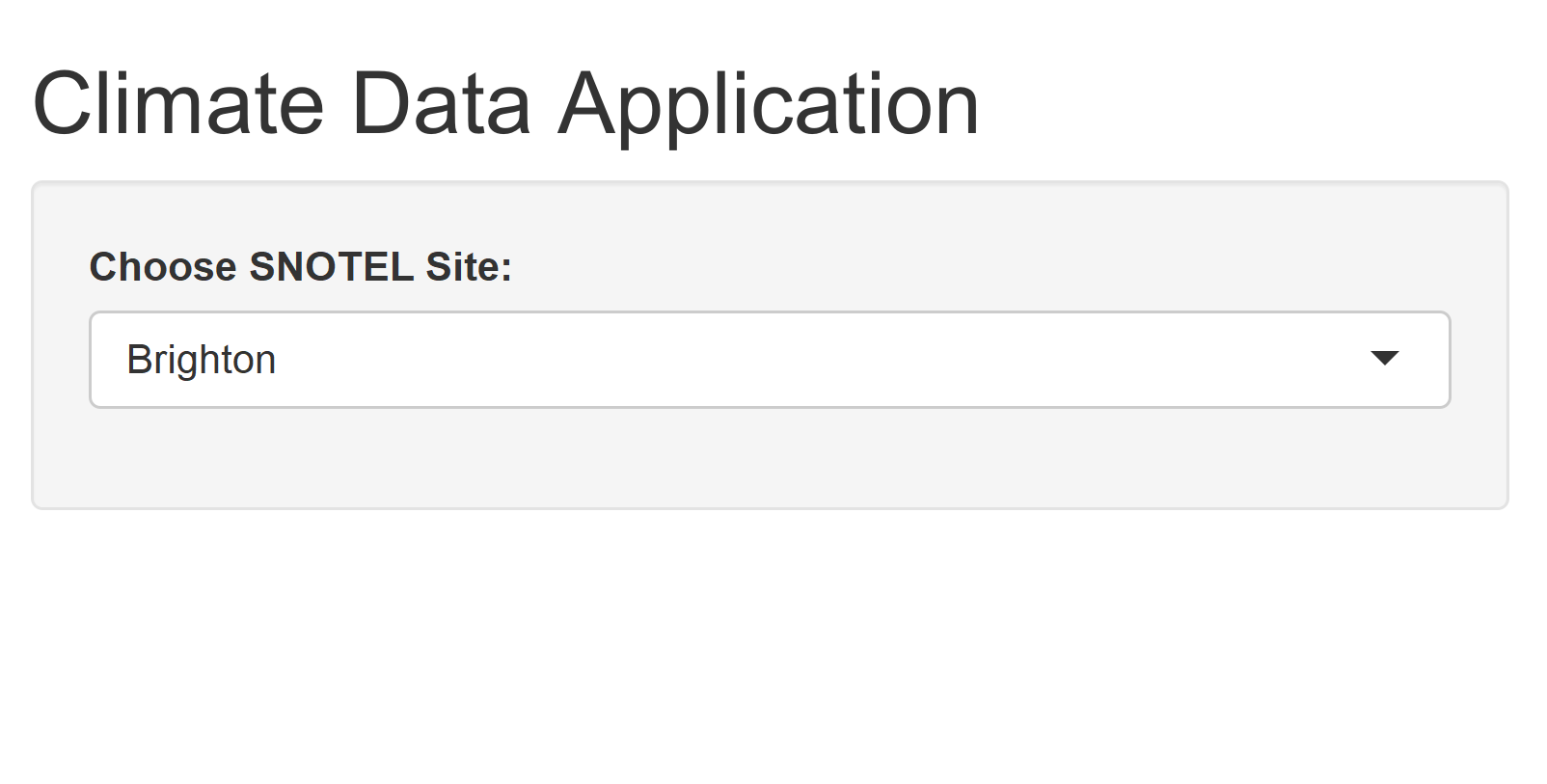
multiple = FALSE,

selectize = TRUE,

width = NULL,

size = NULL)

Run your app. It should look something like this:

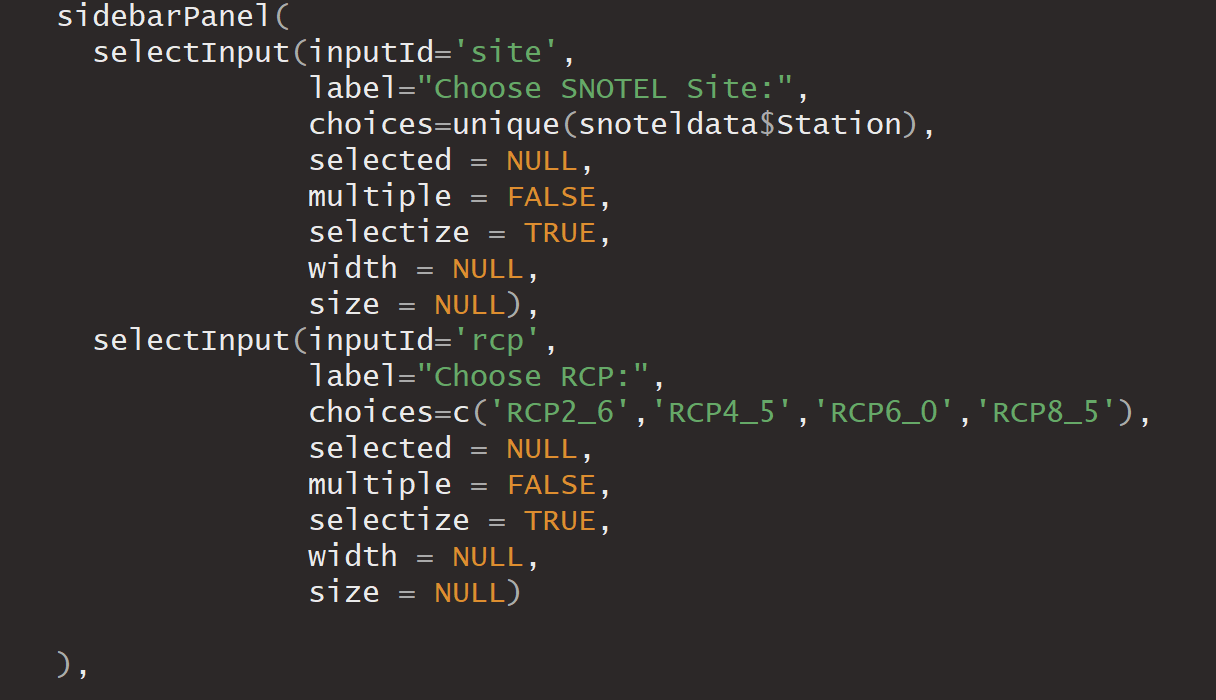


# Step 3 – Add user input control widget for the RCP[[2]](#footnote-2) selection

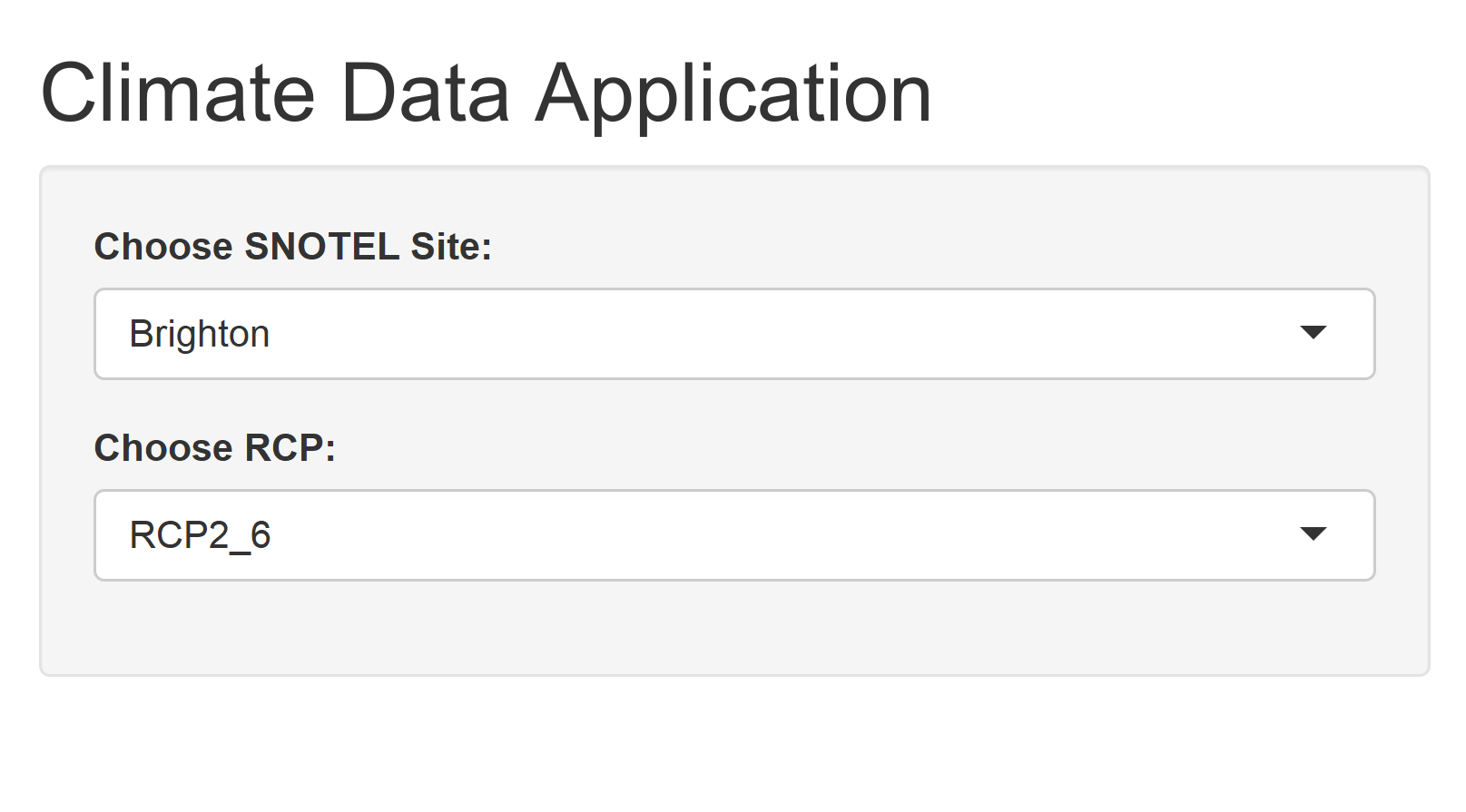
Add this code to the sidebarPanel, below the site widget created in Step 2.

## Option A – Dropdown menu:

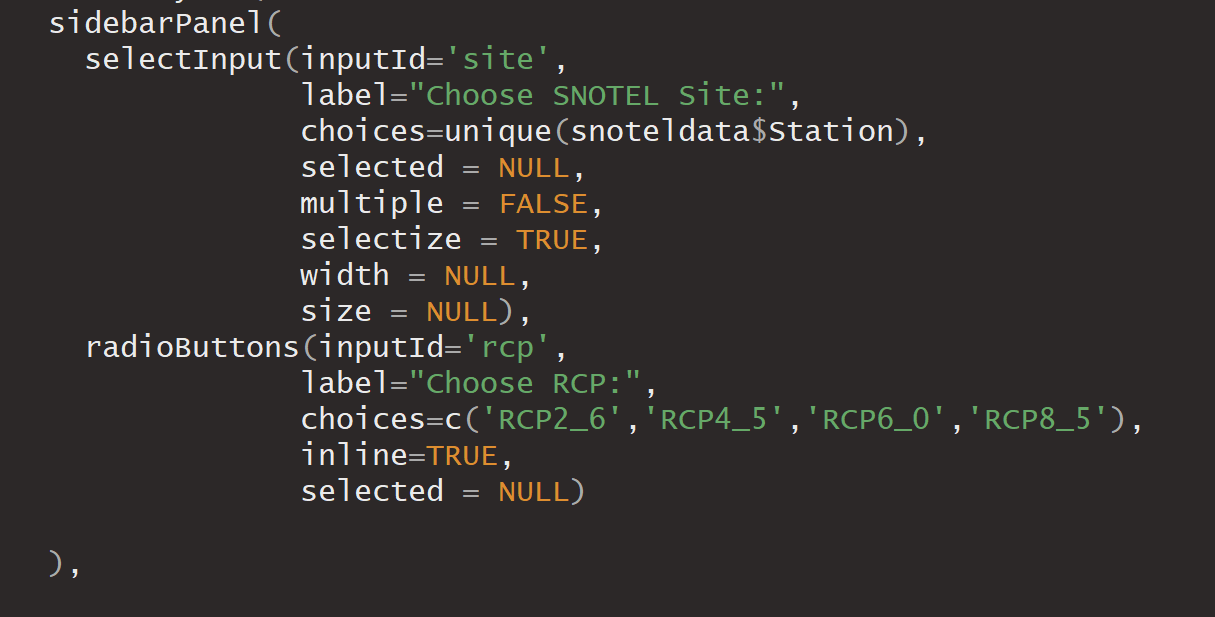
List the four possible RCPs (2.6, 4.5, 6.0, and 8.5). Code will be similar to the dropdown menu (selectInput) created in Step 2, except for the choices.



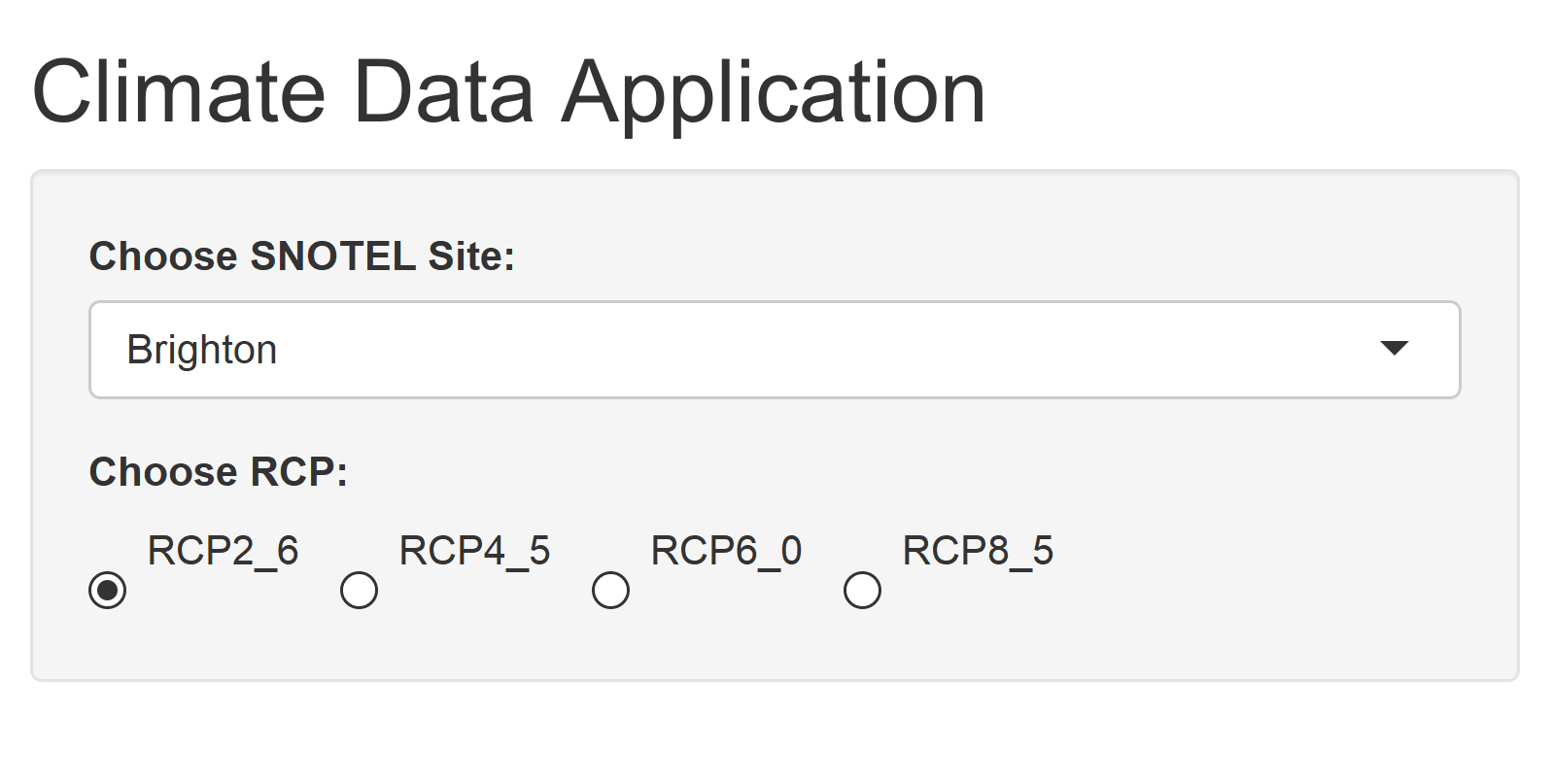
Run your app. It should look something like this:



## Option B – Radio Buttons:

Provide a list of the four possible RCPs (2.6, 4.5, 6.0, and 8.5) for the user to select from: 

Run your app. It should look something like this:



# Step 4 – Add user input for date ranges

Go to <https://shiny.rstudio.com/gallery/widget-gallery.html>, and look at the example for the Date range widget. Click “See Code” and copy the line (up to but not including the comma) for dateRangeInput() in the ui.R tab. Paste this below the other user inputs from Step 2 and 3 in the ui.R script. Don’t forget to put a comma after the previous input widget!

Note: You’ll notice the label is formatted using an “h3” wrapper:

label = h3("Date range")

h3 indicates the label should be stylized as “header 3” style (think HTML header styles)[[3]](#footnote-3).

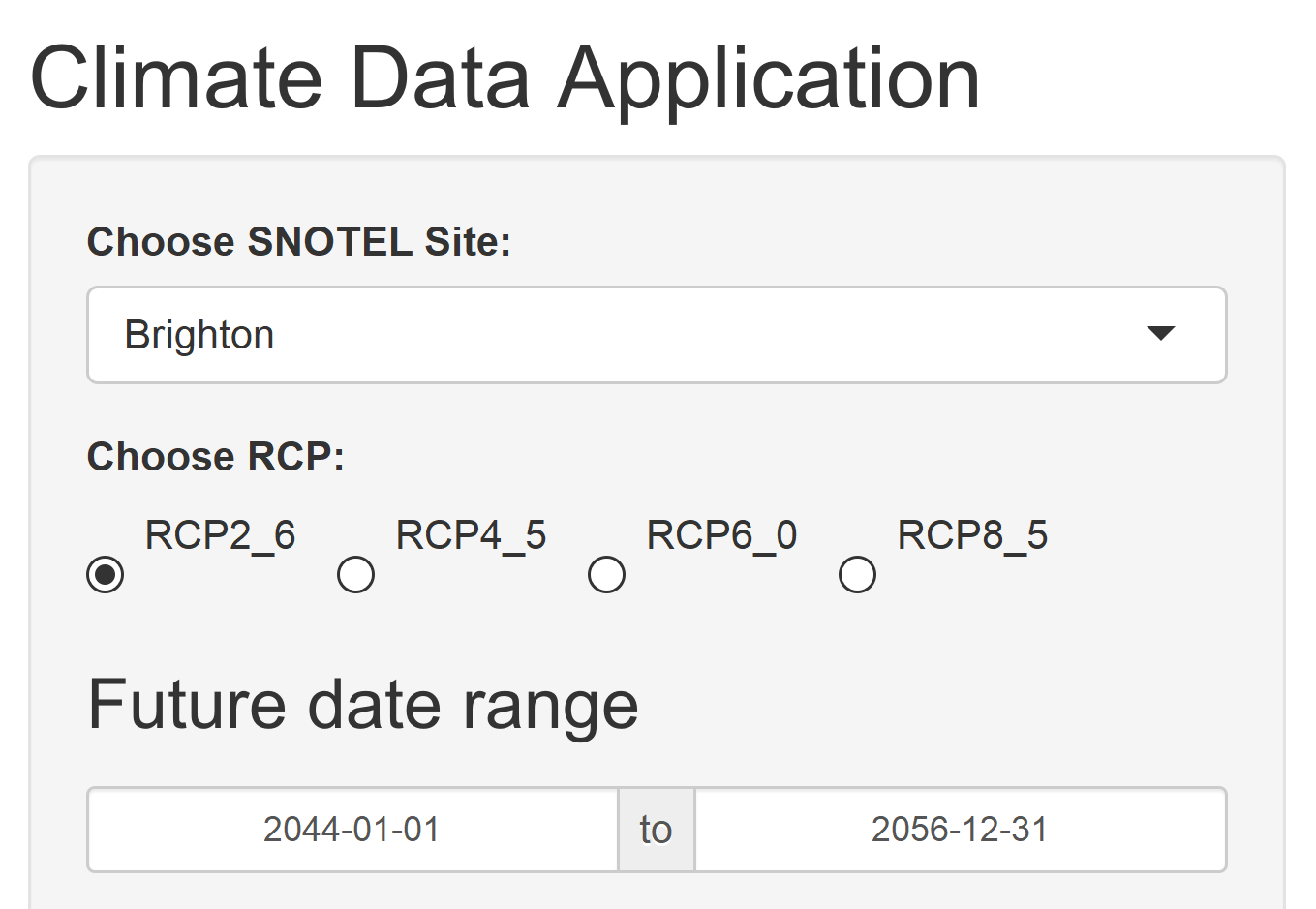
You can either remove this wrapper so that it reads:

dateRangeInput("dates", label = "Date range")

Change the inputId for the dateRangeInput from “dates” to “futuredates”, and the label to “Future date range”.

You may want to set a start/end and min/max dates, since climate data are only provided from 2005-2017 and 2044-2056. See <https://shiny.rstudio.com/reference/shiny/latest/dateRangeInput.html> for how to do this.

Run your app. It should look something like this:

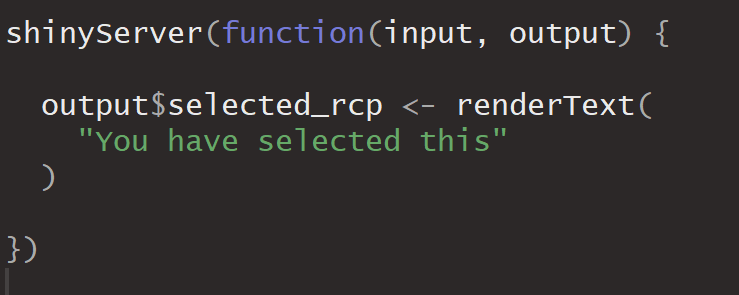


# Step 5 – Create a reactive text output

Go to <https://shiny.rstudio.com/tutorial/written-tutorial/lesson4/>.

Add a textOutput function in the mainPanel (following Step 1 of the example), but instead of calling it “selected\_var”, call it “selected\_rcp”.

Continue following the example (Step 2 of the example) to create an object in the server.R script called “output$selected\_rcp”. This will be a renderText() function.



Replace the example “You have selected this” text with the following:

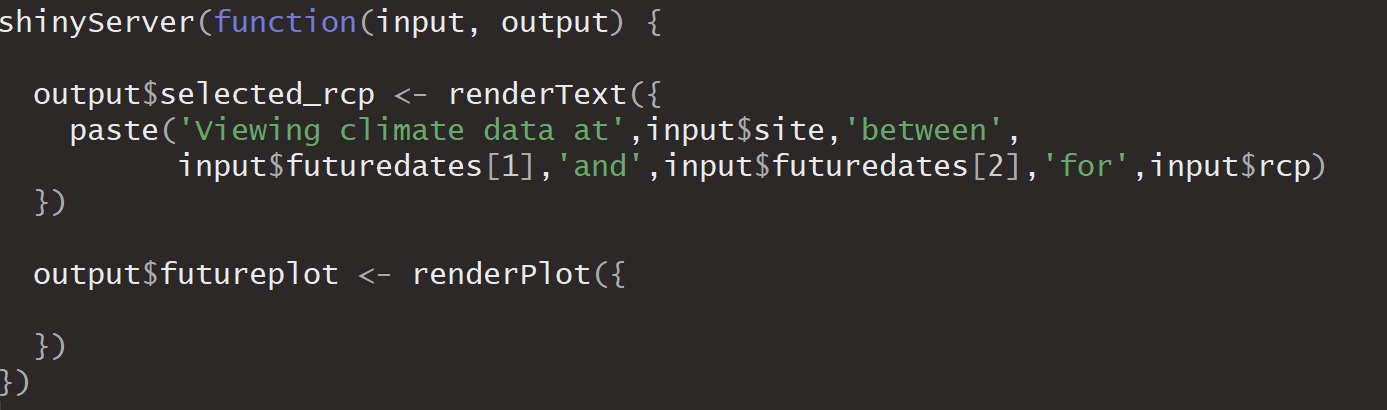
paste('Viewing climate data at',input$site,'between', input$futuredates[1],'and',input$futuredates[2],'for',input$rcp)

Run your app. Note that the text updates dynamically based on the user inputs.

# Step 6 – Add a Plot

Add a line of code to load the ggplot2 package in the global.R script.

Create another output, this time a renderPlot output, called “output$futureplot” after the renderText function. You do not need a comma to separate the outputs in the shinyServer function. This plot will show the projected climate data versus time.



Subset the data to reflect the user inputs. Within the renderPlot function, add the following code:

plotdata <- subset(prcp\_proj,Station==input$site &

Date >= input$futuredates[1] &

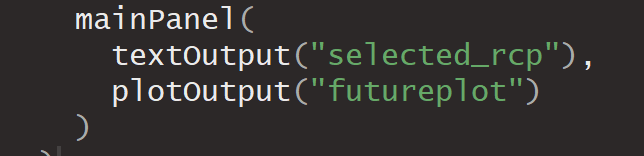
Date<= input$futuredates[2])

Then create a plot by adding the following code within the renderPlot function.

ggplot()+ geom\_line(data=plotdata,aes(x=plotdata$Date,y=plotdata[,input$rcp]))

Customize the plot (add a title, labels, change color, change theme, etc.)

Add the plot object to the user interface, below the textOutput function in the mainPanel.

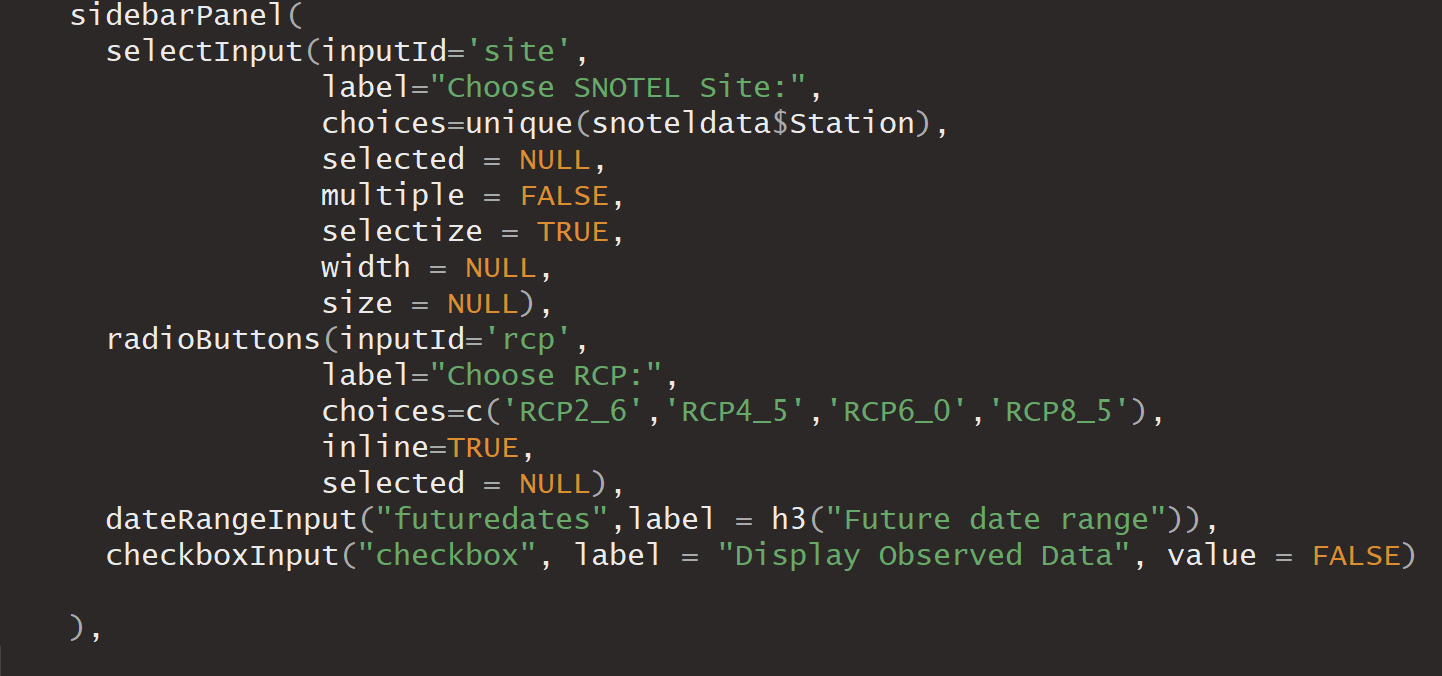


Save and run your app. Both the text and the plot should update based on your input.

# Step 7 – Add a button option to plot observed data

Follow the example in the widget gallery for adding a single checkbox: <http://shiny.rstudio.com/gallery/widget-gallery.html>

This user input can be added beneath the other user inputs.



In the server.R script, modify the plot with an if statement, as follows:

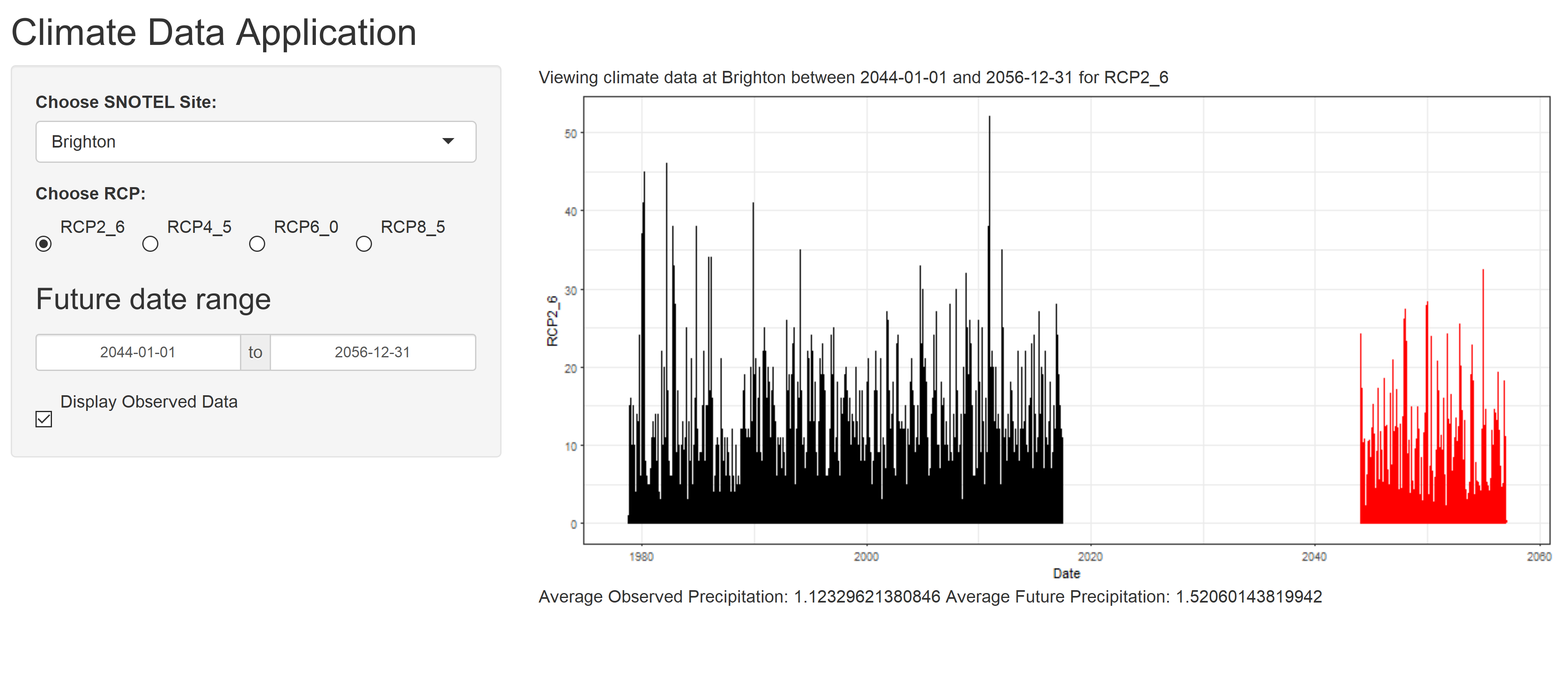
if(input$checkbox==TRUE){

ggplot()+ geom\_line(data=plotdata,aes(x=plotdata$Date,y=plotdata[,input$rcp]),color='red')+ geom\_line(data=snoteldata,aes(x=Date,y=DailyPrecip),color='black'

}else{

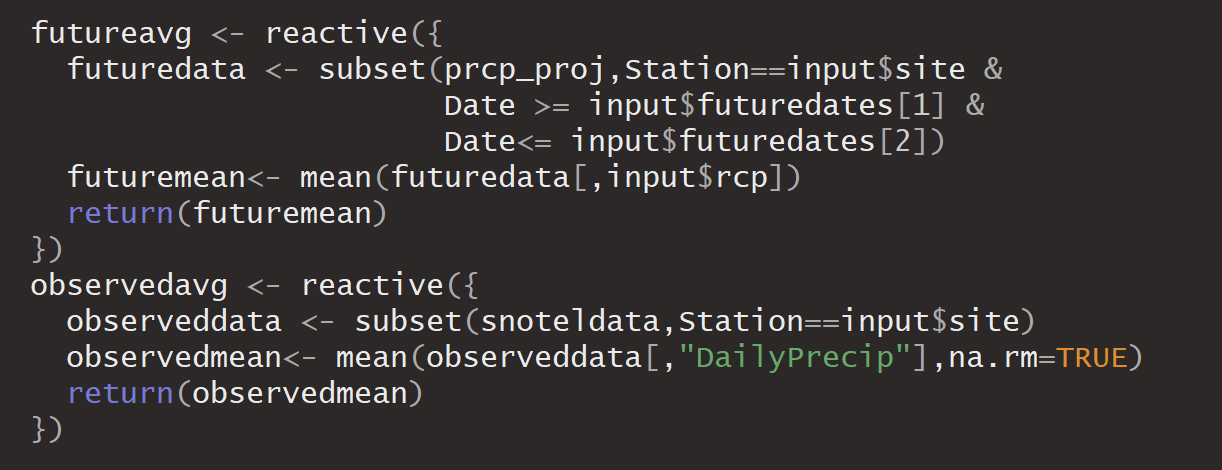
ggplot()+ geom\_line(data=plotdata,aes(x=plotdata$Date,y=plotdata[,input$rcp])) }

Run your app. It should look something like this:



# Step 8 - Add a reactive function that reports summary statistics

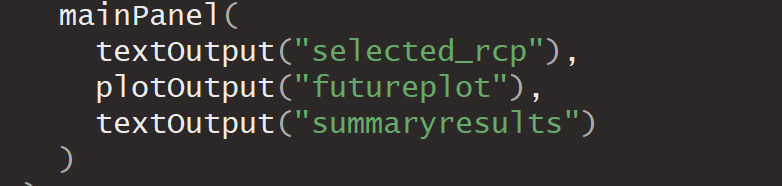
Create reactive functions within the shinyServer function that will calculate summary statistics of the data as the user changes the inputs. Below is an example calculating the mean precipitation:



Add an output within the shinyServer function that will display a result of these calculations.



Add a textOutput to the mainPanel (in the ui.R script), similar to when you added the textOutput in step 5.



Save and test your app.

# Step 9 – Customize!

## Change the title!

## Use different widgets:

* Check out other user inputs in the gallery for other user input widget options: <http://shiny.rstudio.com/gallery/widget-gallery.html>
* The shinyWidgets package has additional options for user inputs: <https://dreamrs.github.io/shinyWidgets/>

## Add text explaining what the app does/how to use it:

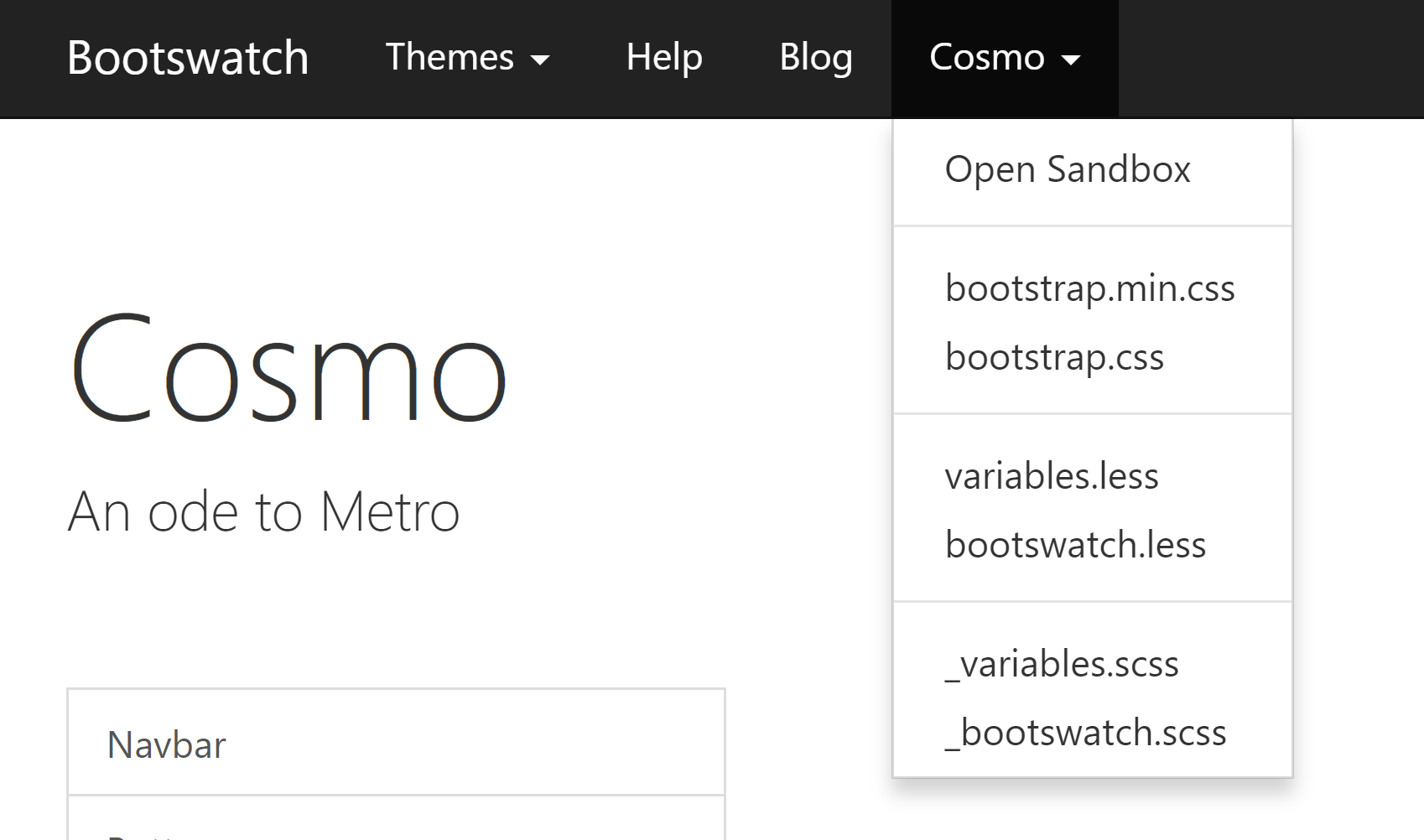
* Use the helpText widget to add (dynamic or non-dynamic) text to your interface: <https://shiny.rstudio.com/reference/shiny/0.11/helpText.html>
* or add text within a panel (see the “Formatted Text” section of <https://shiny.rstudio.com/tutorial/written-tutorial/lesson2/>)

## Add a map:

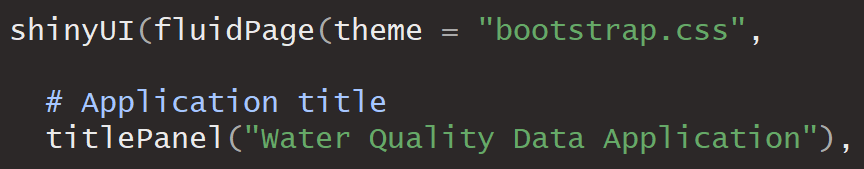
* Use ggmap or leaflet to add a map (static or interactive) to your shiny app
* <https://github.com/dkahle/ggmap>
* <https://rstudio.github.io/leaflet/shiny.html>

## Style your app:

* Add a logo/image: <https://stackoverflow.com/questions/21996887/embedding-image-in-shiny-app>
* Use html-style wrappers to style the text in your app. See the “HTML content” section of <https://shiny.rstudio.com/tutorial/written-tutorial/lesson2/>
* Use css to apply a “custom stylesheet” to your app. Create a subdirectory named “www” in your Shiny app directory. This subdirectory name www is special. Shiny makes every file in www available to your user’s browser. The www subdirectory is a great place to put CSS files, images, and other things a browser needs to build your Shiny App. Check out <https://bootswatch.com/> for free .css files that will change the look of your app (especially if you’ve styled it using the html style tags). Just browse the themes, then select the theme name > bootstrap.css.



This will open a text file in your browser. Save the file in your www folder (right click and save as ‘.css’ file in your www folder). Then, in your ui.R script, assign the theme within the fluidPage object.



## Other ideas:

* Add other parameters (create a user input widget for parameter selection and use your app to look at more than just precipitation. Max and Min temperature files are also provided).
* Instead of choosing a single site, allow users to compare data across different sites.
* Create different kinds of plots.
* Look into the plotly package for an even more interactive application: <https://plot.ly/r/shiny-tutorial/>
* Display subset data in an interactive data table: <https://shiny.rstudio.com/gallery/basic-datatable.html>

1. <https://gdo-dcp.ucllnl.org/downscaled_cmip_projections/dcpInterface.html#About> [↑](#footnote-ref-1)
2. RCP: Representative Concentration Pathway. This corresponds to a specific emissions scenario. [↑](#footnote-ref-2)
3. <https://shiny.rstudio.com/tutorial/written-tutorial/lesson2/>

   [↑](#footnote-ref-3)