

Guidelines on Building a Shiny Application

Environmental Impact Data Collaborative (EIDC)

Last Updated on July, 2023

Table of contents

Learning Objectives	3
Instructions	4
I. Getting Started: Loading Shiny Apps & Data Processing	5
1. Create a new shiny web application in RStudio	5
2. Install and load packages	5
3. Read in and merge the datasets	5
II. Data Visualization: An Interactive Map of the U.S.	7
1. Define a custom function that generates an interactive map	7
2. Customize the layout of the map and store the map object	7
III. Setting up the Shiny Dashboard	9
1. Create the main skeleton of the Shiny application	9
2. Apply a ready-made template to the Shiny app	9
3. Add a page title and 2 tab panels for displaying the interactive map	9
4. Display a header for each landing page	10
5. Place the map within each tab and adjust the size of the output	11
6. Render the map object within the <i>server</i>	11
7. Test the application	12
IV. Publishing the App on the Cloud	12
1. Click on ‘ <i>Publish the application or document</i> ’ on the top right-hand corner of the source editor	12
2. Connect RStudio to shinyapps.io	12

If you have any questions, please e-mail us at eidc@georgetown.edu.

Learning Objectives

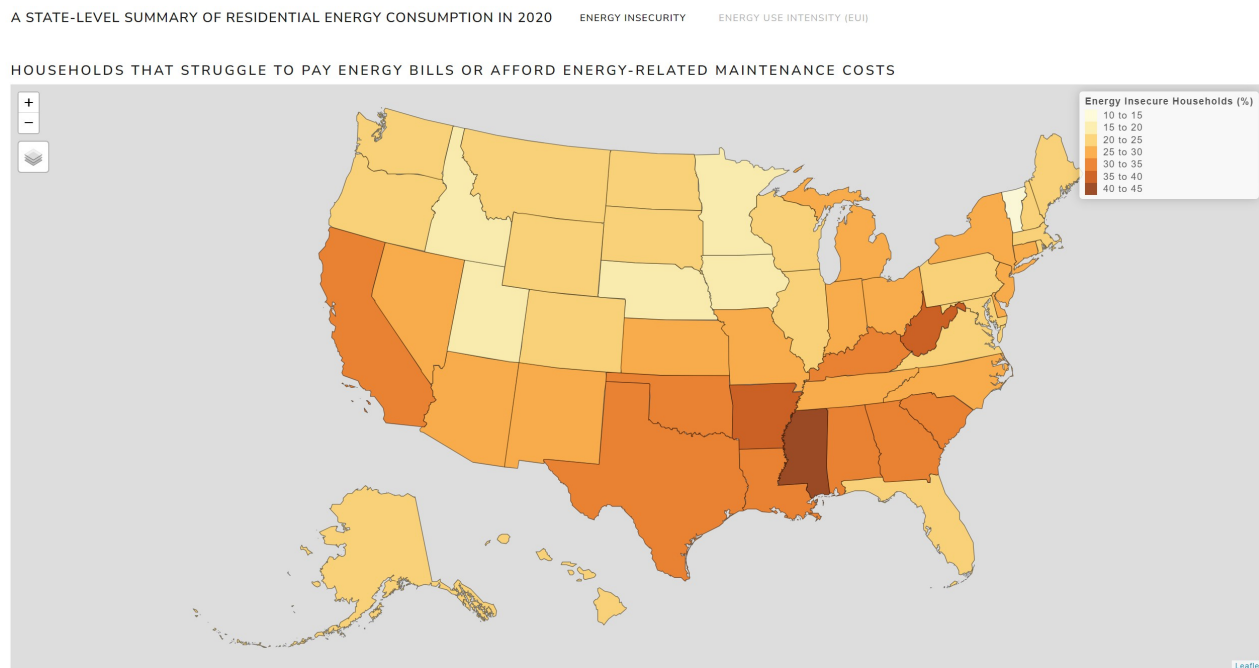
- Build an interactive web application
- Plot an interactive map with custom pop-up tables
- Compare the extent of **energy insecurity** across the U.S.
 - **Energy insecurity** refers to the inability of a household to meet basic energy needs, which includes the following conditions:
 - * Reducing or forgoing basic necessities in order to pay energy bills
 - * Keeping one's home at unhealthy temperatures
 - * Receiving disconnection notice from energy providers
 - * Being unable to use A/C or heating equipment in the home, due to
 - High costs of repair
 - Failure to pay utility bills
- Examine state-level differences in annual household energy consumption, expressed in terms of the size of the housing unit
 - Compare **Energy Use Intensity** (EUI; Btu/square foot), which is an indication of energy efficiency of the home
- Transform and merge data of various formats (**shp.** and **csv.** data types)
 - Create user-defined functions to write clean and efficient blocs of code

Instructions

In this tutorial, we will build a Shiny web application that visualizes findings from the [2020 Residential Energy Consumption Survey](#) (RECS). The app will display 2 maps:

1. One that summarizes the extent of household energy insecurity across U.S. states; and
2. Another that compares the average energy use intensity of U.S. homes.

The final dashboard will look like the following:

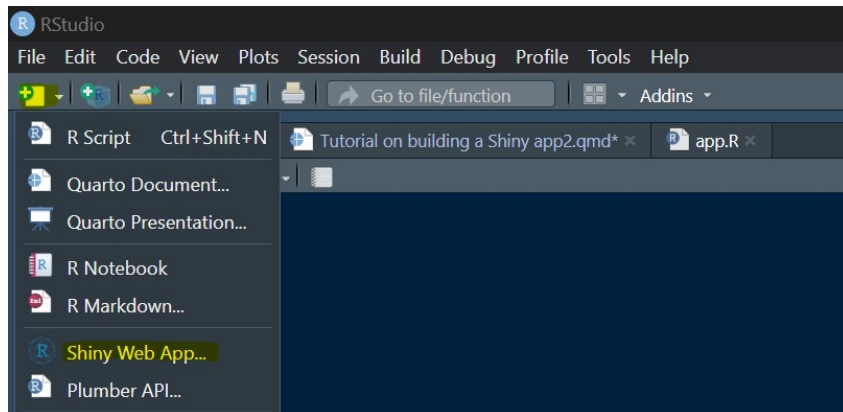


The complete R script for building a Shiny app can be found [here](#).

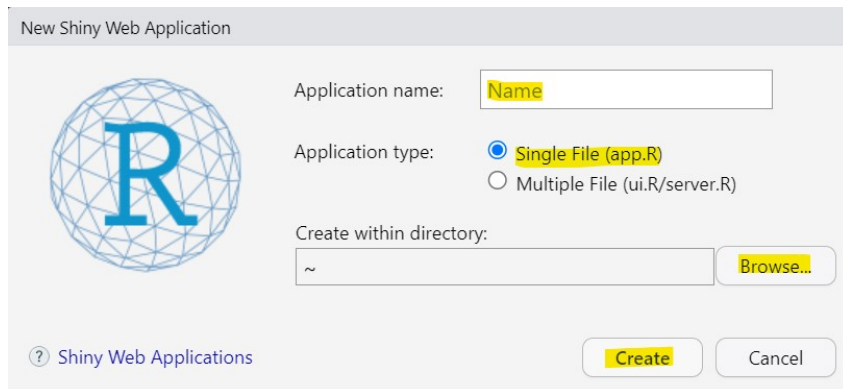
I. Getting Started: Loading Shiny Apps & Data Processing

1. Create a new shiny web application in RStudio

- Click on ‘new file’ > ‘shiny web app’



- Fill out the ‘Application name’ and select the directory in which to save the file.



Note: The file name should be saved as `app.R`.

2. Install and load packages

```
# p_load() function of the pacman package installs & loads packages simultaneously
pacman::p_load(tidyverse, # for data wrangling
               shiny, shinydashboard, shinythemes, bslib, rsconnect, # for shiny app
               sf, tigris, tmap) # for spatial mapping
```

3. Read in and merge the datasets

- Define a function that:
 - Loads the following datasets

- * Two state-level summaries of the 2020 Residential Energy Consumption Survey (RECS)
 - `energy_insecurity.csv`, which provides data on the number of energy insecure households for each state; and
 - `energy_consumption.csv`, which includes data on annual household energy consumption (per square foot)
- * A geospatial data of U.S. states
- Transforms the spatial data
 - * Rotate, scale, and transpose Alaska and Hawaii to locate them closer to the continental United States
 - * Project the map according to an *Albers equal-area conic reference system*, to preserve the correct proportion of the states
- Merges spatial data with the RECS, by *state name*
 - * Convert the names of each state to upper case, for consistency across datasets
 - * Perform a left join, using the *state name* variables as key:
 - `STATE` variable for the RECS dataframe
 - `NAME` variable for the US geospatial dataframe
- Store the merged data as `energy_insecurity` and `energy_consumption`, respectively

```
##### I. Create a user defined function that wrangles the data
merge_the_data <- function(file_name) {

### 1) Read in the RECS summary data
# Define the path directory
directory <-
  "https://raw.githubusercontent.com/quinnei/Residential-Energy-Consumption-2020/main/RECS-Shiny-App/1_Dataset/"
# Specify the location of the csv file
file_path <- paste0(directory, file_name, ".csv")
# Read the csv dataset
RECS <- read.csv(file_path)

### 2) Read in the geospatial data of the US, using the tigris package
US <- states(cb = TRUE, year = 2020) %>%
# Position Alaska & Hawaii below the continental US
  shift_geometry(position = "below") %>%
# For a conic representation of the U.S.
  sf::st_transform("+proj=aea +lat_1=29.5 +lat_2=45.5 +lat_0=37.5 +lon_0=-96")
# Capitalize the state names
US$NAME <- toupper(US$NAME)

### 3) Join the 2 datasets, by the state variable
merged_data <- left_join(RECS, US, by = c("STATE" = "NAME"))
# Make sure that the merged dataframe is recognized as a spatial object
st_geometry(merged_data) <- merged_data$geometry

  return(merged_data)
}

##### II. Call the function and store the merged, spatial dataset
energy_insecurity <- merge_the_data("energy_insecurity")
energy_consumption <- merge_the_data("energy_consumption")
```

II. Data Visualization: An Interactive Map of the U.S.

1. Define a custom function that generates an interactive map

- Use the `tmap` package for visualization
- The function takes in 4 arguments:
 - dataset;
 - main variable of interest, which to indicate in varying degrees of colors;
 - title of the legend; and
 - pop-up labels, for displaying supplementary information
- Delineate state boundaries in black and apply a line width of 0.5
- Fill in the state geometries, using the default yellow-orange color scheme; Set the transparency of the colors to 0.85
- Remove the external borders of the map, which comes as a default
- Place the legend on the top right-hand corner, to prevent it from overlapping with the map
- Make sure that the map is represented according to the *Albers equal-area conic reference system*

```
##### III. Define a function that plots an interactive map

create_map <- function(data, colored_variable, legend_title, popup_labels) {
  tm_shape(data) +
    tm_borders(col = 'black', alpha = 0.5) + # Black outlines for representing the states
    tm_fill(col = colored_variable, title = legend_title,
            popup.vars = popup_labels, # Add popup with relevant info
            id = "STATE", # The first piece of info to display in the popup
            alpha = 0.85) + # Set the transparency of the colors
    tm_layout(frame = FALSE, # Remove the black frame that surrounds the map
              legend.outside = TRUE, # Move the legend
              legend.title.size = 1, legend.text.size = 0.7) +
  # Plot the interactive map in the original CRS (in conic form)
  tm_view(projection = 0) + tm_basemap(NULL)
}
```

2. Customize the layout of the map and store the map object

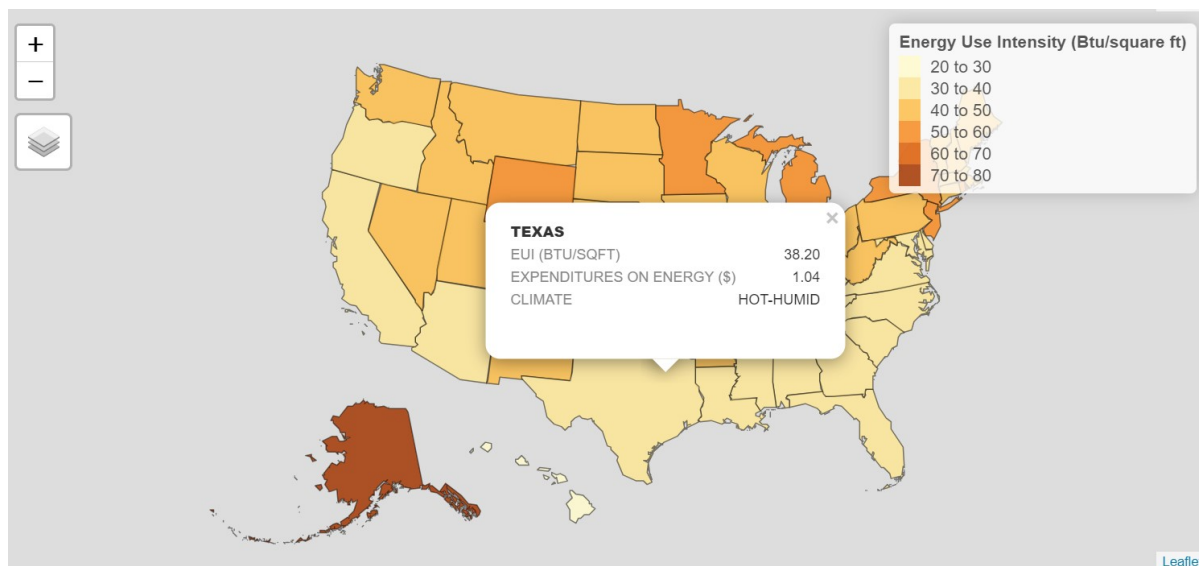
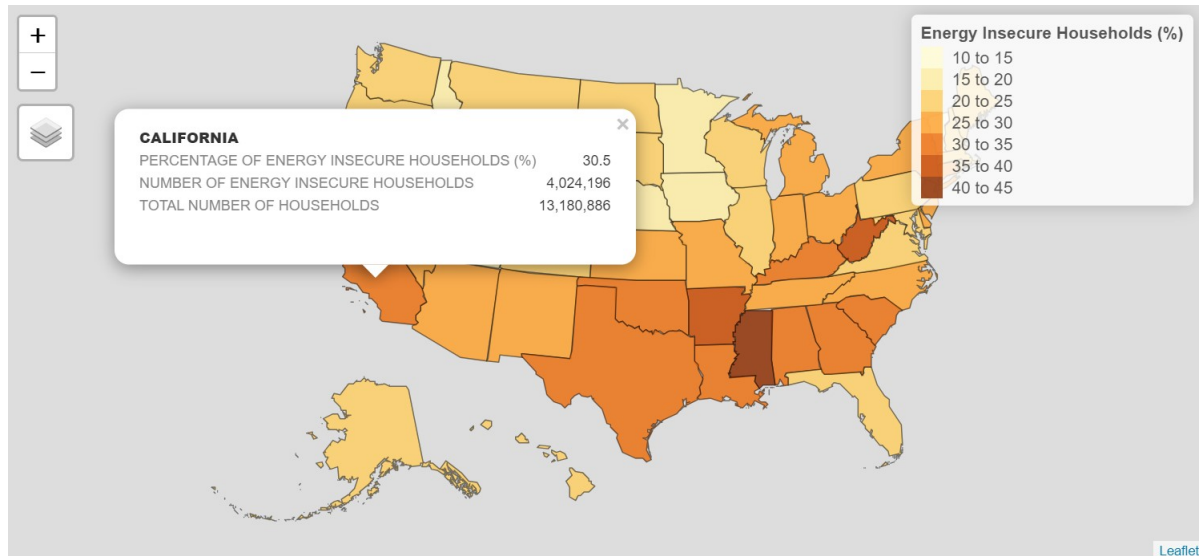
- Create a pair of interactive maps based on the 1) `energy_insecurity` and 2) `energy_consumption` dataframe
 - Store the output as `map_energy_insecurity` and `map_energy_consumption`
- Apply the following layout to each output:

	Energy Insecurity	Energy Use Intensity
Variable used to fill in state geometries	PERCENTAGE	ENERGY_CONSUMPTION_PER_SQFT
Legend title	Energy Insecure Households (%)	Energy Use Intensity (Btu/square ft)
Info to display in the pop-up table and their respective variable names	<ul style="list-style-type: none">• PERCENTAGE OF ENERGY INSECURE HOUSEHOLDS (%) - PERCENTAGE• NUMBER OF ENERGY INSECURE HOUSEHOLDS - NUM_ENERGY_INSECURE_HOUSEHOLDS• TOTAL NUMBER OF HOUSEHOLDS - TOTAL_HOUSEHOLDS	<ul style="list-style-type: none">• EUI (BTU/SQFT) - ENERGY_CONSUMPTION_PER_SQFT• EXPENDITURES ON ENERGY (\$) - ENERGY_EXPENDITURE_PER_SQFT• CLIMATE - CLIMATE

```
##### IV. Create an interactive map of 1) energy insecurity 2) Energy Use Intensity (EUI)

map_energy_insecurity <- create_map(
  energy_insecurity, 'PERCENTAGE', "Energy Insecure Households (%)",
  # Indicate % and absolute values of energy insecurity & total number of households in each state
  c("PERCENTAGE OF ENERGY INSECURE HOUSEHOLDS (%)" = "PERCENTAGE",
    "NUMBER OF ENERGY INSECURE HOUSEHOLDS" = "NUM_ENERGY_INSECURE_HOUSEHOLDS",
    "TOTAL NUMBER OF HOUSEHOLDS" = "TOTAL_HOUSEHOLDS"))

map_energy_consumption <- create_map(
  energy_consumption, 'ENERGY_CONSUMPTION_PER_SQFT', "Energy Use Intensity (Btu/square ft)",
  # For each state, indicate EUI, total average expenditures on energy, and climate region
  c("EUI (BTU/SQFT)" = "ENERGY_CONSUMPTION_PER_SQFT",
    "EXPENDITURES ON ENERGY ($)" = "ENERGY_EXPENDITURE_PER_SQFT",
    "CLIMATE" = "CLIMATE"))
```



III. Setting up the Shiny Dashboard

Note that *Steps III-2 to III-5* are about designing the user interface (ui), whereas *III-6* relates to creating the `server` logic.

1. Create the main skeleton of the Shiny application

```
##### V. Create a user interface that defines the visual elements of the app
ui <- fluidPage()

##### VI. Create a server, which consists of an 1) input and 2) output
##### Tells the server how to generate outputs from the inputs provided
server <- function(input, output){}

##### VII. Run the Shiny app
shinyApp(ui = ui, server = server)
```

2. Apply a ready-made template to the Shiny app

- Select any of the themes provided by the `bslib` package
 - Run `bootswatch_themes()` to check out the [full list of themes available](#)

```
##### V. Create a user interface that defines the visual elements of the app
ui <- fluidPage(
##### 1) Apply a minimalist theme
  theme = bs_theme(bootswatch = "lux")
)
```

3. Add a page title and 2 tab panels for displaying the interactive map

The application should look like the following:

A STATE-LEVEL SUMMARY OF RESIDENTIAL ENERGY CONSUMPTION IN 2020
PAGE TITLE

ENERGY INSECURITY ENERGY USE INTENSITY (EUI)
TAB NAME

HOUSEHOLDS THAT STRUGGLE TO PAY ENERGY BILLS OR LIVE IN POTENTIALLY DANGEROUS THERMAL CONDITIONS
HEADER

- Create a navigation bar at the top of the page
- Add the title of the web page: A STATE-LEVEL SUMMARY OF RESIDENTIAL ENERGY CONSUMPTION IN 2020
- Create 2 tabs, which will display an interactive map of 1) energy insecurity and 2) energy use intensity
 - Assign names to each tab; Introduce the topic/main theme of the visualized output

	Energy Insecurity	Energy Use Intensity
Tab name	ENERGY INSECURITY	ENERGY USE INTENSITY (EUI)
Header	HOUSEHOLDS THAT STRUGGLE TO PAY ENERGY BILLS OR AFFORD ENERGY-RELATED MAINTENANCE COSTS	ANNUAL HOUSEHOLD ENERGY CONSUMPTION (PER SQUARE FOOTAGE)
Output ID	energy_insecurity_map	energy_consumption_map
Output size	width = "100%", height = "865px"	width = "100%", height = "865px"

- For a preview of the Shiny app, click on ‘Run app’ on the top right-hand corner of the source editor.

```
##### V. Create a user interface that defines the visual elements of the app
ui <- fluidPage(
##### 1) Apply a minimalist theme
  theme = bs_theme(bootswatch = "lux"),

##### 2) Create a navigation bar at the top of the page
  navbarPage(
# 2-1) Specify the title of the web page
  title = "A STATE-LEVEL SUMMARY OF RESIDENTIAL ENERGY CONSUMPTION IN 2020",

# 2-2) Create 2 main tabs
# 2-2-a) title of tab 1
    tabPanel("ENERGY INSECURITY"),

# 2-2-a') title of tab 2
    tabPanel("ENERGY USE INTENSITY (EUI)")
  )
)
```

4. Display a header for each landing page

The header will be displayed above the interactive map, underneath every tab.

- Apply a **h4** [HTML style heading](#) to the header
 - Tags that define HTML headings range from **h1** to **h6**
 - **h1** defines the most important heading and is the largest in size.
 - **h6** defines the least important heading and is the smallest in size

```
##### V. Create a user interface that defines the visual elements of the app
ui <- fluidPage(
##### 1) Apply a minimalist theme
  theme = bs_theme(bootswatch = "lux"),

##### 2) Create a navigation bar at the top of the page
  navbarPage(
# 2-1) Specify the title of the web page
  title = "A STATE-LEVEL SUMMARY OF RESIDENTIAL ENERGY CONSUMPTION IN 2020",

# 2-2) Create 2 main tabs
# 2-2-a) title of tab 1
    tabPanel("ENERGY INSECURITY",
# 2-2-b) header for the landing page
      h4("HOUSEHOLDS THAT STRUGGLE TO PAY ENERGY BILLS OR AFFORD ENERGY-RELATED MAINTENANCE COSTS")),

# 2-2-a') title of tab 2
    tabPanel("ENERGY USE INTENSITY (EUI)",
# 2-2-b') header for the landing page
      h4("ANNUAL HOUSEHOLD ENERGY CONSUMPTION (PER SQUARE FOOTAGE)")
    )
  )
)
```

5. Place the map within each tab and adjust the size of the output

- Instruct Shiny to display 2 outputs: the interactive maps to be defined in *Step III-6*
 - Tell Shiny *where* to display the map object by listing `tmapOutput(outputId)` within `ui`
 - * Nest the map object under each tab, using `tabPanel(..., tmapOutput(outputId))`
- Set the width of the output to 100% and height to 865px , using `tmapOutput(outputId, width = ..., height = ...)`

```
##### V. Create a user interface that defines the visual elements of the app
ui <- fluidPage(
##### 1) Apply a minimalist theme
  theme = bs_theme(bootswatch = "lux"),

##### 2) Create a navigation bar at the top of the page
  navbarPage(
# 2-1) Specify the title of the web page
    title = "A STATE-LEVEL SUMMARY OF RESIDENTIAL ENERGY CONSUMPTION IN 2020",

# 2-2) Create 2 main tabs
# 2-2-a) title of tab 1
    tabPanel("ENERGY INSECURITY",
# 2-2-b) title of the landing page
      h4("HOUSEHOLDS THAT STRUGGLE TO PAY ENERGY BILLS OR AFFORD ENERGY-RELATED MAINTENANCE COSTS"),
# 2-2-c) Define the size of the map
      tmapOutput("energy_insecurity_map", width = "100%", height = "865px")
    ),

# 2-2-a') title of tab 2
    tabPanel("ENERGY USE INTENSITY (EUI)",
# 2-2-b') title of the landing page
      h4("ANNUAL HOUSEHOLD ENERGY CONSUMPTION (PER SQUARE FOOTAGE)"),
# 2-2-c') Define the size of the map
      tmapOutput("energy_consumption_map", width = "100%", height = "865px")
    )
  )
)
```

6. Render the map object within the server

- Render the map object defined in *Step II-2* (i.e. `map_energy_insecurity` and `map_energy_consumption`)
- Toggle on the interactive mode
 - The map can be plotted either as a static image or an interactive, zoomable output, using the "plot" or "view" mode
- Create a list of outputs (i.e. `energy_insecurity_map` and `energy_consumption_map`) inside the `server`, to make these outputs visible
 - **Note:** The *output names* in both the `server` (i.e. `output$name`) and the `ui` (i.e. `tmapOutput("name")`) should be identical to each another
 - * `output$energy_insecurity_map` in the `server` *should match* the `tmapOutput("energy_insecurity_map")` in the `ui`
 - * `output$energy_consumption_map` in the `server` *should match* the `tmapOutput("energy_consumption_map")` in the `ui`

```
##### VI. Create a server, which consists of input and output
server <- function(input, output) {
##### 1) Generate the 'energy_insecurity_map' output
  output$energy_insecurity_map <- renderTmap({
# 1-a) Turn on interactive feature of the map
    tmap_mode("view")
# 1-b) Return the output that was defined in Part I.
    map_energy_insecurity
  })

##### 2) Generate the 'energy_consumption_map' output
  output$energy_consumption_map <- renderTmap({
# 2-a) Turn on interactive feature of the map
    tmap_mode("view")
# 2-b) Return the output that was defined in Part I.
    map_energy_consumption
  })
}
```

7. Test the application

- Click on each tab and check whether the map has been rendered successfully
- Make sure that the information displayed in the pop-up table is without error

IV. Publishing the App on the Cloud

Subscribers of shinyapps.io can host their web applications for free, without having to set up a Shiny Server from scratch. Under the free plan, users can deploy up to 5 applications, which supports a monthly usage of up to 25 active hours.

1. Click on ‘*Publish the application or document*’ on the top right-hand corner of the source editor

2. Connect RStudio to shinyapps.io

- Sign in/Sign up for a free account at shinyapps.io
- (For new subscribers only) Authorize the shinyapps.io account
 - Copy the token by clicking on ‘*Show secret*’ > ‘*Copy to clipboard*’
 New users can skip the installation of `rsconnect`, as it has already been done in *Step I-2*

Start with R

Start with Python

STEP 1 – INSTALL RSCONNECT

The `rsconnect` package can be installed directly from CRAN. To make sure you have the latest version run following code in your R console:

```
install.packages('rsconnect')
```

STEP 2 – AUTHORIZE ACCOUNT

The `rsconnect` package must be authorized to your account using a token and secret. To do this, click the copy button below and we'll copy the whole command you need to your clipboard. Paste it into your R console to authorize your account. Once you've entered the command successfully in R, that computer is now authorized to deploy Shiny for R applications to your shinyapps.io account.

```
rsconnect::setAccountInfo(name='[REDACTED]',
  token='[REDACTED]',
  secret='<SECRET>')
```

Show secret

Copy to clipboard

In the future, you can manage your tokens from the [Tokens](#) page the settings menu.

STEP 3 – DEPLOY

Once the `rsconnect` package has been configured, you're ready to deploy your first application. If you haven't written any applications yet, you can also checkout the [Getting Started Guide](#) for instructions on how to deploy our demo application. Run the following code in your R console.


```
library(rsconnect)
rsconnect::deployApp('path/to/your/app')
```

- Paste the token in the space provided. Then proceed to ‘*Connect Account*’

Connect Account

Back

Connect ShinyApps.io Account



Go to [your account on ShinyApps](#) and log in.

Click your name, then choose **Tokens** from your account menu.

Click **Show** on the token you want to use, then **Show Secret** and **Copy to Clipboard**. Paste the result here:

```
rsconnect::setAccountInfo(name=[REDACTED],
  token='[REDACTED]',
  secret='[REDACTED]')
```

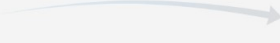

Need a ShinyApps.io account? [Get started here](#).

Connect Account


Cancel

- Select 1) the `app.R` file, 2) your shinyapps.io account, and 3) provide a title for the Shiny app. Then click ‘*Publish*’

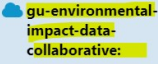
Publish to Server



Publish Files From: .../RECS-Shiny-App/2-2_R
script_Energy_Insecurity_and_EUI

☒  app.R

Publish From Account: [Add New Account](#)

 shinyapps.io

Title:

RECS-Energy-Insecurity-and-EUI

☒ Launch browser

Publish

Cancel