



Washington University in St. Louis

UNIVERSITY LIBRARIES

Making Web Maps with R

Dorris Scott

Learning Objectives

- To map points with the Leaflet for R package.
- To download a polygon using the tidycensus package.
- To map polygons with the Leaflet for R package.
- To have basic familiarity with R Markdown.
- To embed your map in R Markdown and share it using RPub.

Getting Started

Open up R-Studio

For an RScript: **File -> New File -> RScript**

For a R Markdown document: **File -> New File -> R Markdown**

You don't have to change anything in the **New Markdown Document** options window.. Just click OK and we'll change some things later!

Installing Packages

Install and load these packages before we start!

```
install.packages("leaflet")  
install.packages("tidycensus")  
install.packages("tidyverse")  
install.packages("rgdal")  
install.packages("sf")  
install.packages("rmarkdown")  
install.packages("magrittr")
```

```
library(leaflet)  
library(tidycensus)  
library(tidyverse)  
library(rgdal)  
library(sf)  
library(rmarkdown)  
library(magrittr)
```

Important thing to note!

Make sure to not use absolute paths!!!

Example:

```
C:/Users/DataServices/Workshop/  
Making_Web_Maps_With_R  
/Data/shapefile.shp
```

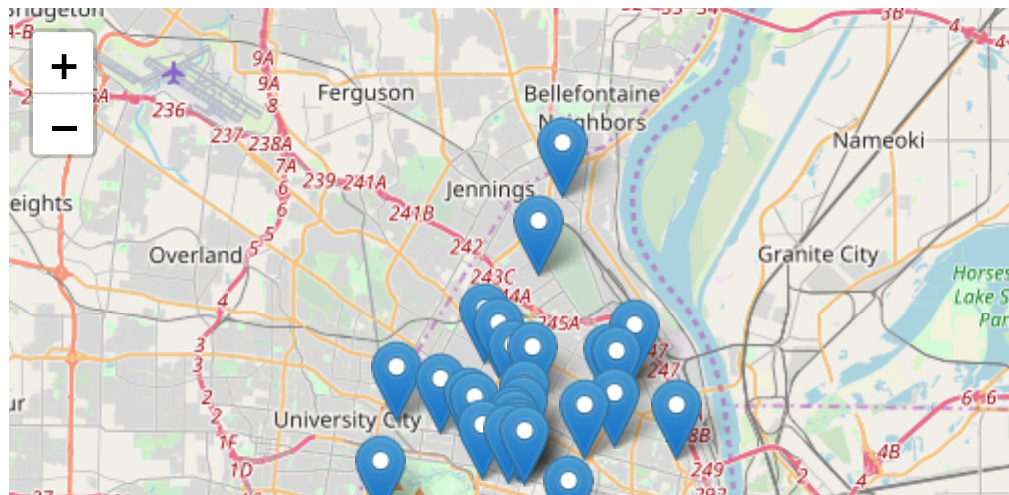
Use relative paths instead.

Example:

```
./Data/shapefile.shp
```

Mapping points with the Leaflet for R Package

```
ltc_facilities <- readOGR("./dataprep/SLC_LTC_Facilities/slc_ltc_...")
leaflet() %>%
  addTiles() %>%
  addMarkers(data = ltc_facilities, lng = ~LONGITUDE, lat = ~LATITUDE,
             popup = as.character(ltc_facilities$FACILITY),
             label = as.character(ltc_facilities$FACILITY))
```

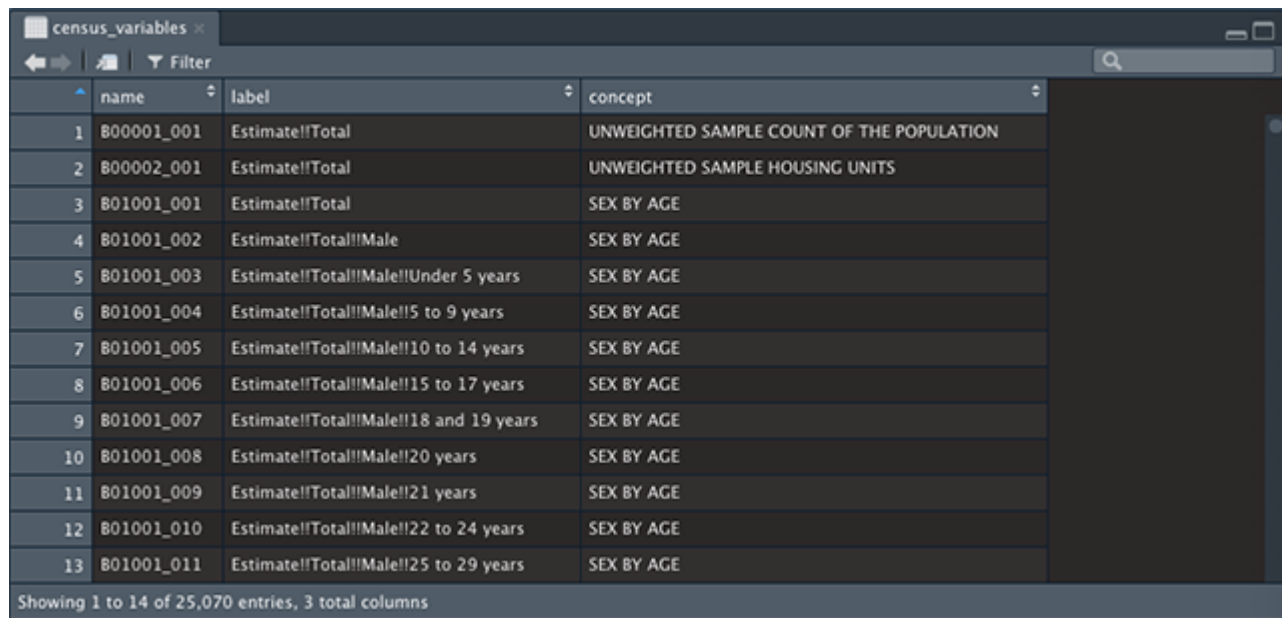


Another Important thing to note!

Make sure to set `verbose = FALSE` to suppress any warning messages. This doesn't fix everything though, for we will have to change another option later on.

Downloading a Polygon with the tidycensus package

```
census_api_key("ENTER API KEY HERE")  
census_variables <- load_variables(2017, "acs5", cache = TRUE)
```



	name	label	concept
1	B00001_001	Estimate!!Total	UNWEIGHTED SAMPLE COUNT OF THE POPULATION
2	B00002_001	Estimate!!Total	UNWEIGHTED SAMPLE HOUSING UNITS
3	B01001_001	Estimate!!Total	SEX BY AGE
4	B01001_002	Estimate!!Total!!Male	SEX BY AGE
5	B01001_003	Estimate!!Total!!Male!!Under 5 years	SEX BY AGE
6	B01001_004	Estimate!!Total!!Male!!5 to 9 years	SEX BY AGE
7	B01001_005	Estimate!!Total!!Male!!10 to 14 years	SEX BY AGE
8	B01001_006	Estimate!!Total!!Male!!15 to 17 years	SEX BY AGE
9	B01001_007	Estimate!!Total!!Male!!18 and 19 years	SEX BY AGE
10	B01001_008	Estimate!!Total!!Male!!20 years	SEX BY AGE
11	B01001_009	Estimate!!Total!!Male!!21 years	SEX BY AGE
12	B01001_010	Estimate!!Total!!Male!!22 to 24 years	SEX BY AGE
13	B01001_011	Estimate!!Total!!Male!!25 to 29 years	SEX BY AGE

Showing 1 to 14 of 25,070 entries, 3 total columns

Downloading a Polygon with the tidycensus package

Here are the variables we are interested in...

	name	label	concept
1	B18101_00	Total Population	Sex and Age by Disability Status
2	B18101_016	Estimate- Total Male 65 to 74 years old with a disability	Sex and Age by Disability Status
3	B18101_019	Estimate - Totale Male 75 years and older with a disability	Sex and Age by Disability Status
4	B18101_035	Estimate - Total Female 65 to 74 years with a disability	Sex and Age by Disability Status
5	B18101_038	Estimate - Total Female 75 years and over with a disability	Sex and Age by Disability Status

Downloading a Polygon with the tidycensus package

```
stlouis_disabled <- get_acs(geography = "tract",  
                             variables = c("B18101_001", "B18101_016", "B18101_035", "B18101_038"), state = "MO",  
                             county = "510", geometry = TRUE, output = "sf")
```

Downloading a Polygon with the tidycensus package

Creating a new variable with the mutate function

```
stlouis_disabled <- get_acs(geography = "tract",  
                             variables = c("B18101_001", "B18101_016", "B18101_019",  
                                           "B18101_035", "B18101_038"), state = "MO",  
                             county = "510", geometry = TRUE, output = "json",  
                             mutate(totalpop = B18101_016E + B18101_019E + B18101_035E + B18101_038E,  
                                     percent_disb = (totalpop/B18101_001E)*100))
```

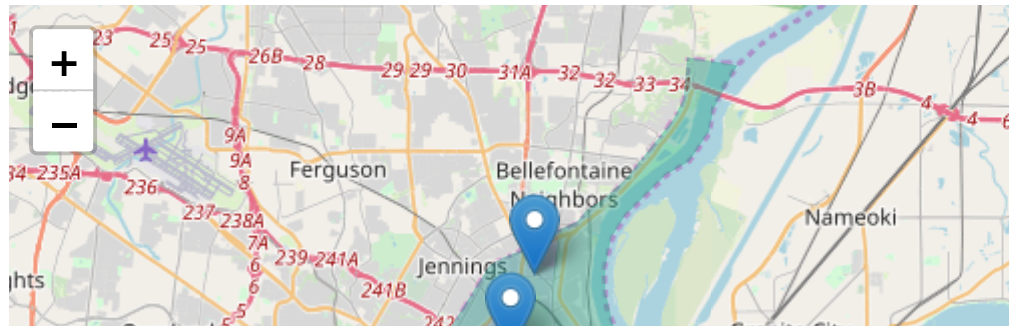
The pipe operation above (`%>%`) moves the variables from the previous operation to the next operation which is to sum up all of the total populations by sex.

Mapping points with the Leaflet for R Package

class: center

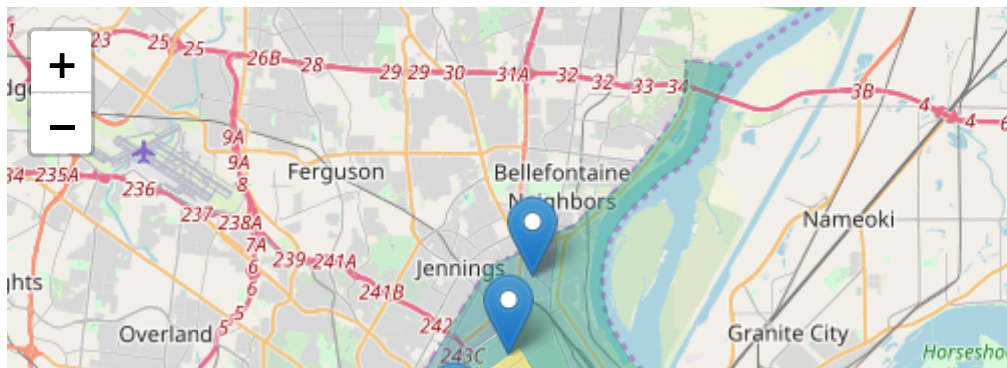
```
#setting the color palette for the polygon
pal = colorNumeric(palette = "viridis", domain = stlouis_disabled)

map <- leaflet() %>%
  addTiles() %>%
  addMarkers(data = ltc_facilities, lng = ~LONGITUDE, lat = ~LATITUDE)
  addPolygons(data = stlouis_disabled, fillOpacity = 0.4, stroke = "red")
map
```



Adding a legend to your leaflet map

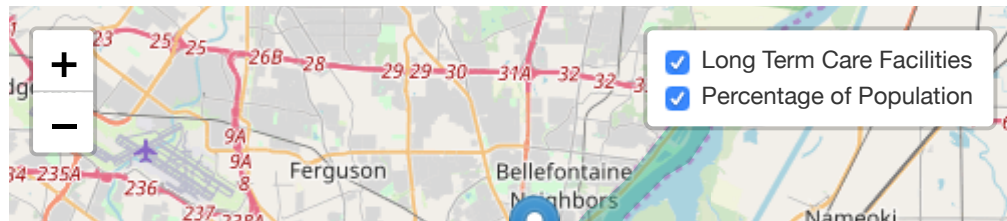
```
map <- leaflet() %>%  
  addTiles() %>%  
  addMarkers(data = ltc_facilities, lng = ~LONGITUDE, lat = ~LATITUDE) %>%  
  addPolygons(data = stlouis_disabled, fillOpacity = 0.4, stroke = "red",  
              popup = stlouis_disabled$NAME) %>%  
  addLegend(data = stlouis_disabled, pal = pal, values = ~stlouis_disabled$NAME,  
            position = "bottomright", title = "Disabled Population",  
            labFormat = labelFormat(suffix = "%"))  
map
```



Adding interactivity to the map

```
map <- leaflet() %>%  
  addTiles() %>%  
  addMarkers(data = ltc_facilities, lng = ~LONGITUDE, lat = ~LATITUDE,  
             group = "Long Term Care Facilities") %>%  
  addPolygons(data = stlouis_disabled, fillOpacity = 0.4, stroke = ~stroke_color,  
             group = "Percentage of Population", popup = stlouis_disabled_popup) %>%  
  addLegend(data = stlouis_disabled, pal = pal, values = ~stlouis_disabled_values,  
            position = "bottomright", title = "Disabled Population",  
            labFormat = labelFormat(suffix = "%")) %>%  
  #We will add the layer control here  
  addLayersControl(  
    overlayGroups = c("Long Term Care Facilities", "Percentage of Population"),  
    options = layersControlOptions(collapsed = FALSE))
```

map



Introduction to R Markdown

The image shows a side-by-side comparison of an R Markdown document in its source and rendered states. On the left, the RStudio editor shows the source code of an R Markdown file named 'Untitled.Rmd'. The code includes a YAML header with title, author, date, output type, and runtime. The main content is a text block explaining Shiny, followed by a section titled 'Inputs and Outputs' which describes how to embed Shiny inputs and outputs. The code then defines an input panel with a 'Number of bins' select input and a 'Bandwidth adjustment' slider input, and a 'renderPlot' function. On the right, the rendered output is shown in a browser window. It displays the same text content, but the 'Inputs and Outputs' section is replaced by an interactive Shiny interface. This interface includes a 'Number of bins' dropdown menu set to 20 and a 'Bandwidth adjustment' slider set to 1. Below these inputs is a histogram titled 'Geyser eruption duration' showing the density of eruption durations. The histogram has two distinct peaks, one around 2.0 and another around 4.5. A blue density curve is overlaid on the histogram bars.

```
1 ---
2 title: "Untitled"
3 author: "Garrett"
4 date: "July 10, 2014"
5 output: html_document
6 runtime: shiny
7 ---
8
9 This R Markdown document is made interactive using Shiny. Unlike the more
10 traditional workflow of creating static reports, you can now create
11 documents that allow your readers to change the assumptions underlying
12 your analysis and see the results immediately.
13
14 To learn more, see [Interactive Documents](http://rmarkdown.rstudio.com/
15 authoring_shiny.html).
16
17 ## Inputs and Outputs
18
19 You can embed Shiny inputs and outputs in your document. Outputs are
20 automatically updated whenever inputs change. This demonstrates how a
21 standard R plot can be made interactive by wrapping it in the Shiny
22 'renderPlot' function. The 'selectInput' and 'sliderInput' functions
23 create the input widgets used to drive the plot.
24
25 ```{r, echo=FALSE}
26 inputPanel(
27   selectInput("n_breaks", label = "Number of bins:",
28     choices = c(10, 20, 35, 50), selected = 20),
29   sliderInput("bw_adjust", label = "Bandwidth adjustment:",
30     min = 0.2, max = 2, value = 1, step = 0.2)
31 )
32
33 renderPlot({
```

Untitled
Garrett
July 10, 2014

This R Markdown document is made interactive using Shiny. Unlike the more traditional workflow of creating static reports, you can now create documents that allow your readers to change the assumptions underlying your analysis and see the results immediately.

To learn more, see [Interactive Documents](http://rmarkdown.rstudio.com/authoring_shiny.html).

Inputs and Outputs

You can embed Shiny inputs and outputs in your document. Outputs are automatically updated whenever inputs change. This demonstrates how a standard R plot can be made interactive by wrapping it in the Shiny `renderPlot` function. The `selectInput` and `sliderInput` functions create the input widgets used to drive the plot.

Number of bins: 20 Bandwidth adjustment: 1

Geyser eruption duration

The histogram shows the density of eruption durations. The x-axis ranges from 1.5 to 5.0, and the y-axis (Density) ranges from 0.0 to 0.7. There are two main clusters of data: one between 1.5 and 2.5, and another between 3.5 and 5.0. A blue density curve is overlaid on the histogram bars.

Header 1 # Header 1

Header 2 ## Header 2

header 3 ### Header 3

This is italicized *This is italicized*

This is bold **This is bold**

Inserting R Code Chunks

Insert -> R

For today's workshop, we will create two code chunks.

Pre-processing code chunk

Mapping code chunk

Code chunk options

`include = TRUE/FALSE`

Prevents the code and results from appearing from the finished file. Other code chunks can use it though.

`echo = TRUE/FALSE`

Whether to display the code along with its results

`message = TRUE/FALSE`

Whether to display messages.

Code chunk options

The preprocessing code chunk will have the option of **include = FALSE**

- Loading packages
- Loading the points shapefile
- Downloading the polygon shapefile
- Setting the color palette for the polygon

Code chunk options

The second code chunk will have the options of **eval = TRUE**, **echo = FALSE**, **message = FALSE**, **warning = FALSE**

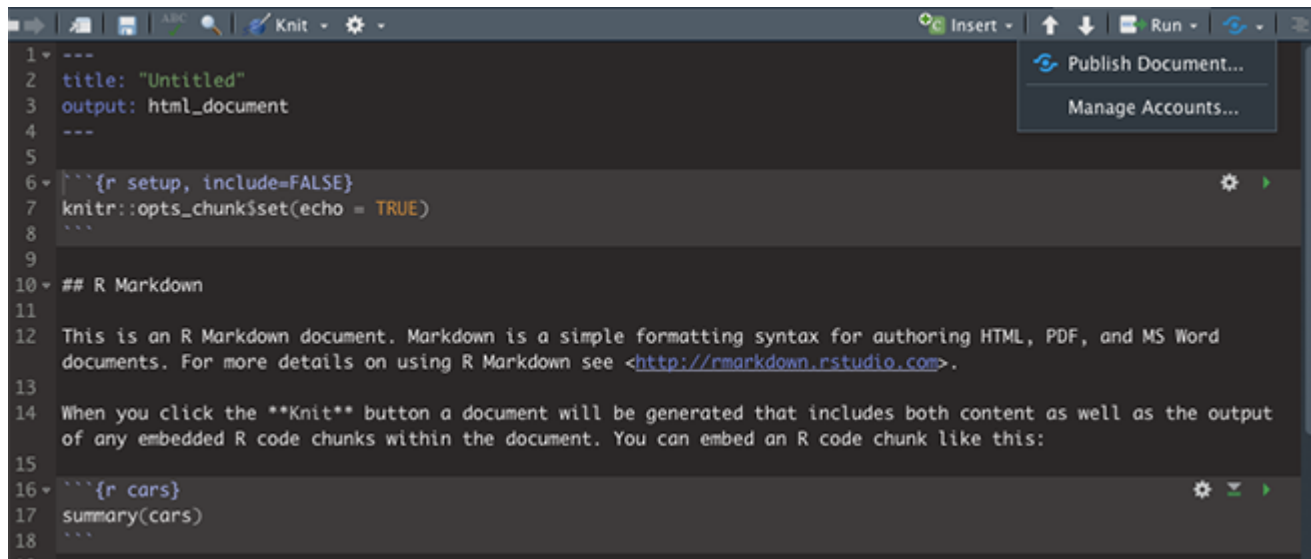
- Mapping of points and polygons
- Adding a legend
- Adding a layer control panel

Final touches!

- Adding a title and author above your code chunks
- Adding a brief description of your map
- Preview your map by clicking the **Knit** button

Uploading your map online

- Go to the R Pubs website and register for an account.
- Publish your document from R Studio



The screenshot shows the R Studio interface. The top toolbar includes buttons for 'Insert', 'Run', and 'Publish Document...'. The 'Publish Document...' button is highlighted, and a dropdown menu is visible with the option 'Manage Accounts...'. The main editor area displays R Markdown code. The code starts with a YAML header: `---`, `title: "Untitled"`, `output: html_document`, `---`. This is followed by an R code chunk: ````{r setup, include=FALSE}`, `knitr::opts_chunk$set(echo = TRUE)`, `````. Below the code is a section titled `## R Markdown` containing text about R Markdown and a link to <http://rmarkdown.rstudio.com>. Another R code chunk is shown at the bottom: ````{r cars}`, `summary(cars)`, `````.

Now it's time to make your own
leaflet map!