

Introduction to Data Science

Interactive Maps With Leaflet

Florian Winkler and Jiayu Yang

November 4, 2021



Outline



Introduction to Leaflet



Use cases for Leaflet maps



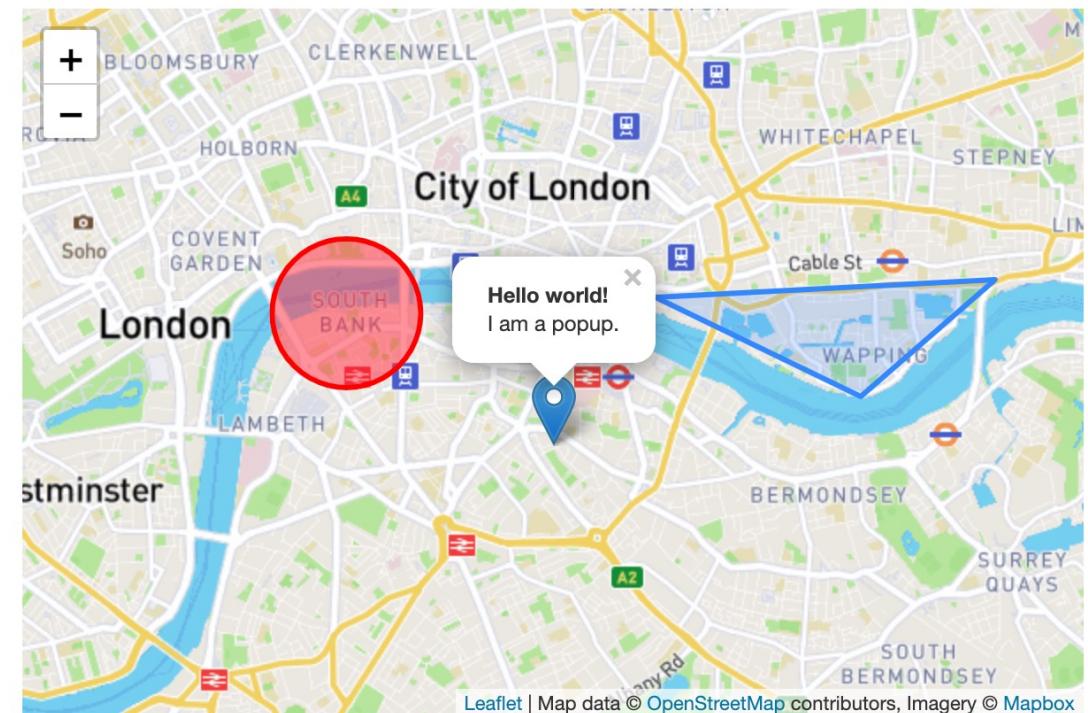
Leaflet workflow & key features



Further references and resources

What is Leaflet?

- One of the most popular open-source JavaScript libraries for interactive maps
- Creates embedding maps with tiled base layers, panning and zooming, and layer features
- Used by websites from the New York Times and Washington Post to GitHub and Flickr
- No JavaScript knowledge required: R package makes it easy to create Leaflet maps in R



Why should you use Leaflet?

Leaflet [website](#): "*Leaflet is designed with **simplicity, performance and usability** in mind.*"

- Free and open-source
- Easy to learn: Uses familiar R framework (e.g., pipe operator `%>%`, similar syntax to `ggplot2`)
- Clear and comprehensive R [documentation](#)
- Flexible: Draws on powerful JavaScript library (can be extended with plugins and connected to API)
- Most importantly, Leaflet maps are interactive.



A few caveats before you start:

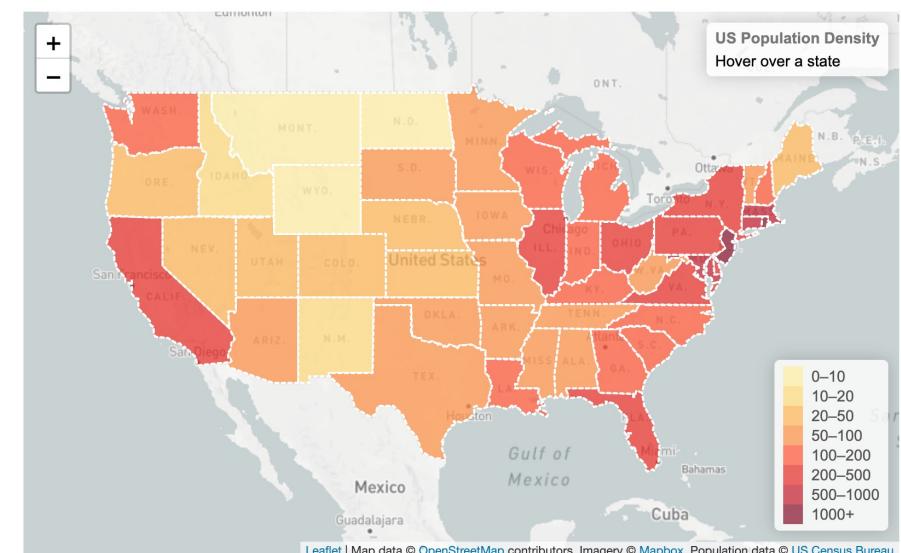
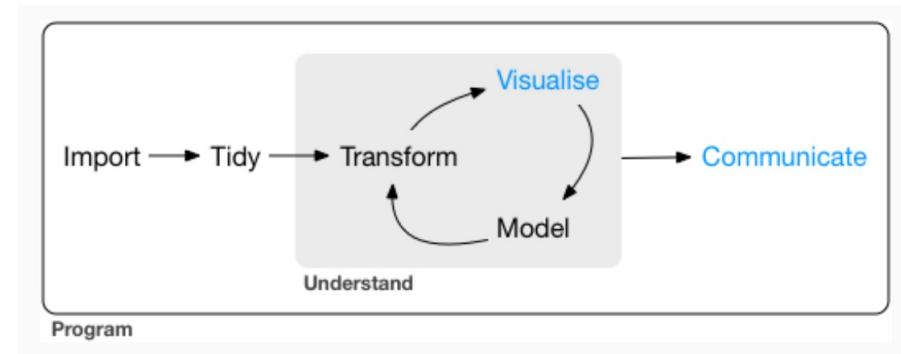
- Data collection: Spatial data required (e.g., latitudes and longitudes, GeoJSON data, or [other formats](#))
- Designed for interactive environments such as web apps (not pasting images in papers & presentations)
- JavaScript knowledge is necessary for more advanced Leaflet features, API and plugins.

Use cases for Leaflet

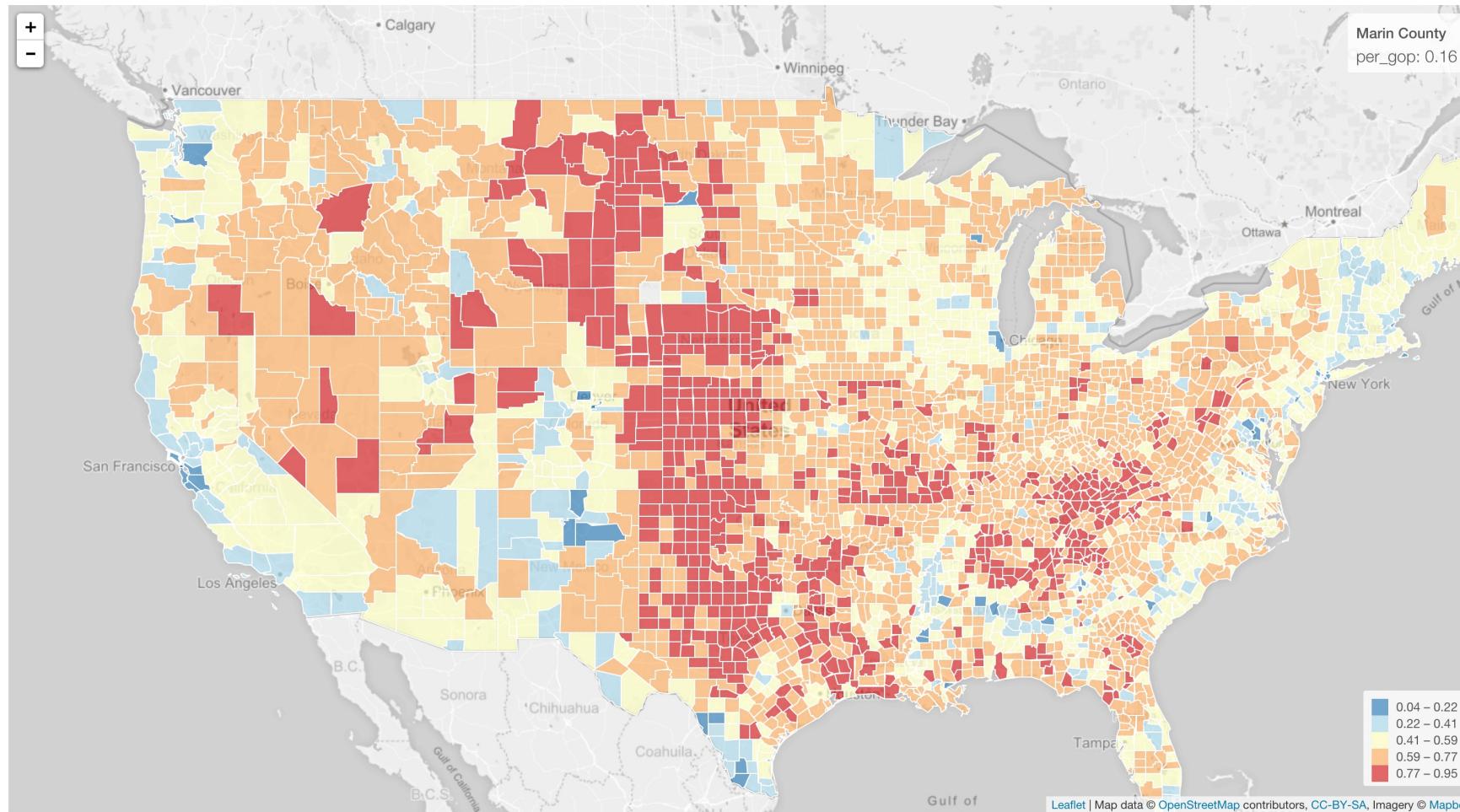
Leaflet maps in the data science workflow: To **visualize and communicate** your findings.

Real-world examples for inspiration:

- Electoral maps (e.g., election results)
- COVID-19 data (e.g., cases across countries)
- Socioeconomic data (e.g., development indicators)
- Climate impact data (e.g., meteorological maps)
- ... the sky's the limit.

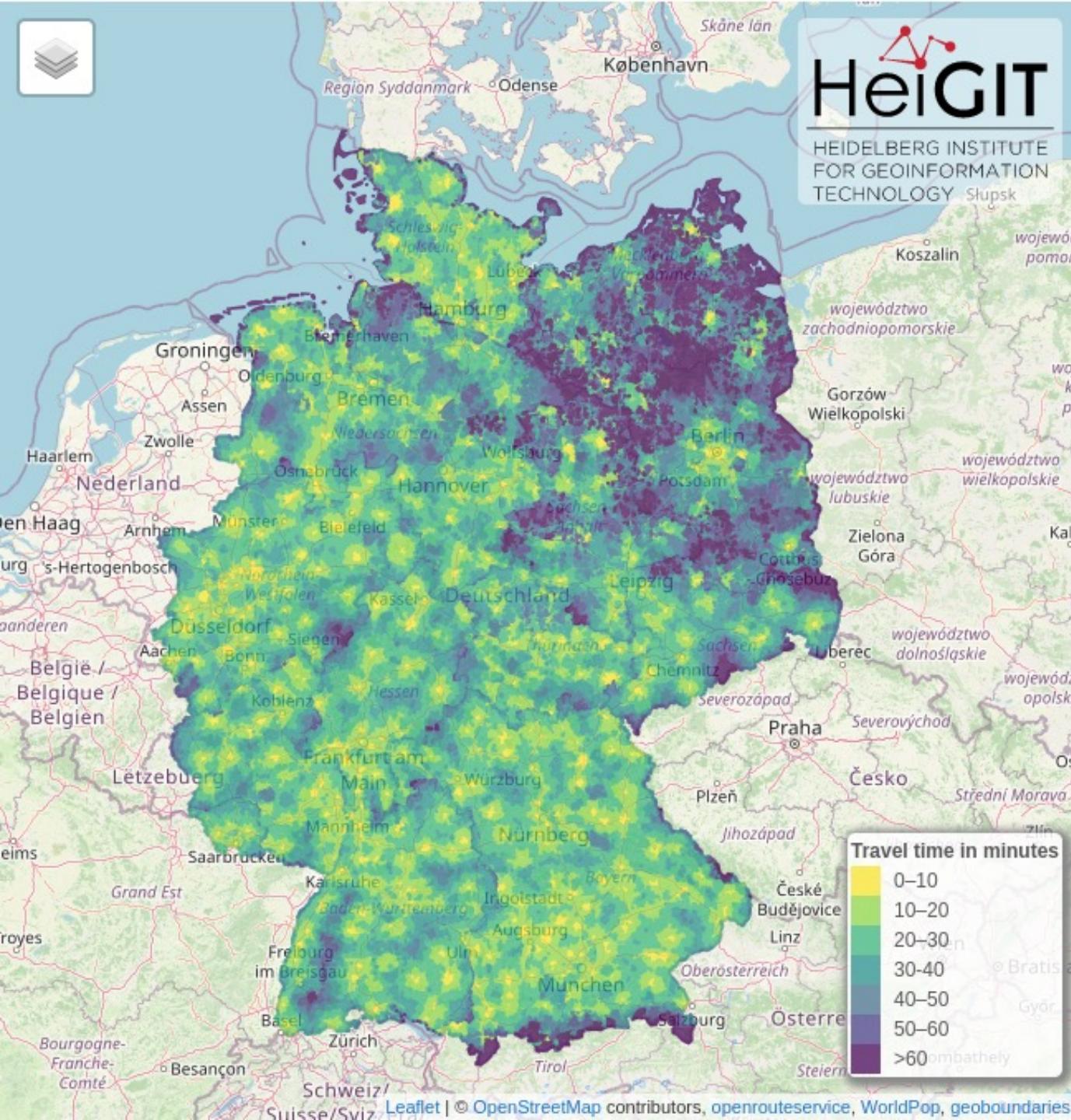


Use cases: Electoral maps



2016 US Presidential Election Results by County
(percentage of Republican voters)

Image source: <https://blog.exploratory.io/creating-geojson-out-of-shapefile-in-r-40bc0005857d>



COVID-19

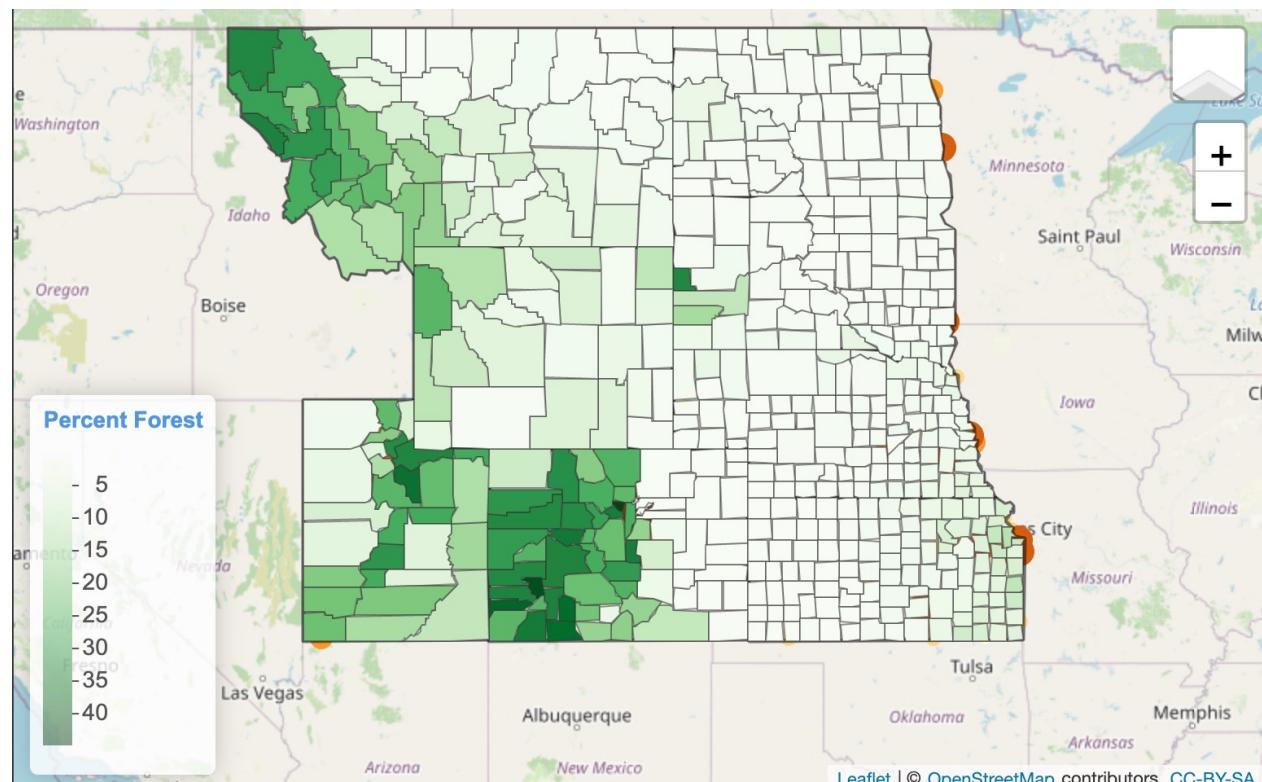
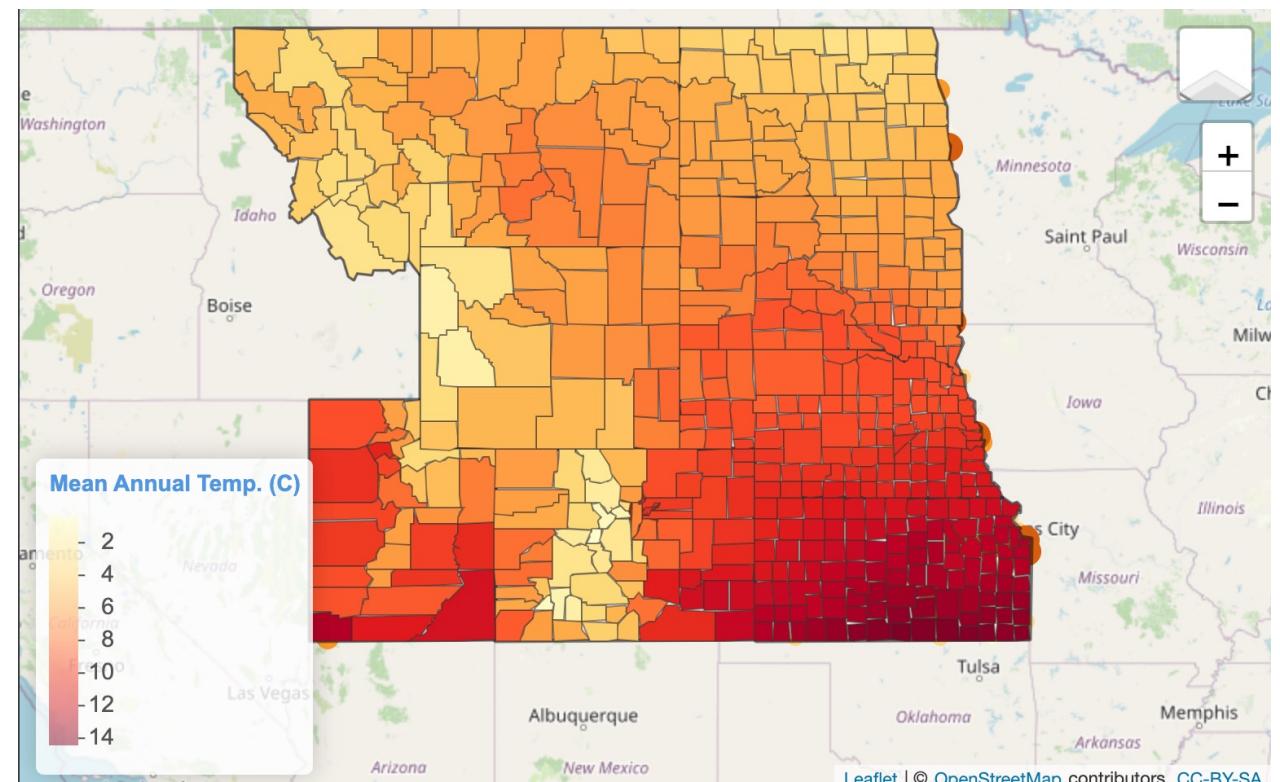
Accessibility of COVID-19 vaccination centers in Germany

(motorized travel time in minutes)

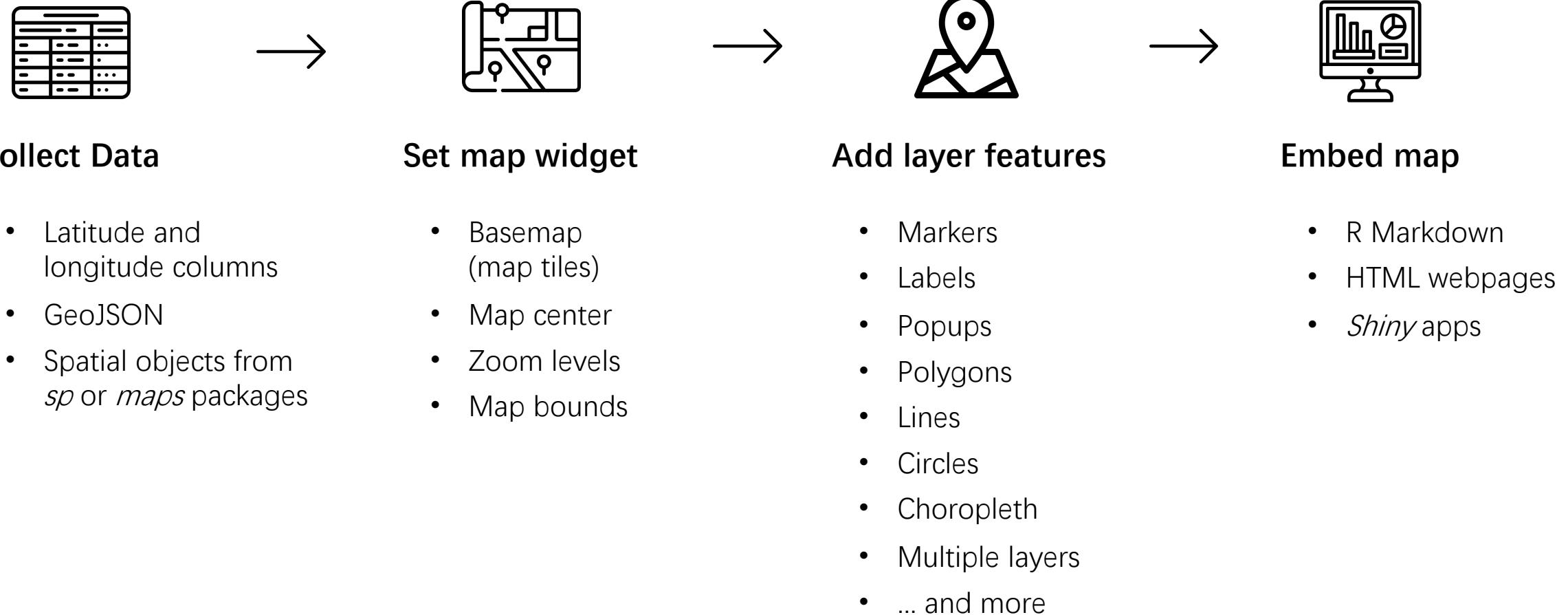
Image source: <http://k1z.blog.uni-heidelberg.de/2021/04/02/german-mass-vaccination-sites-in-open-healthcare-access-map/>

Use cases: Climate data

Mean annual temperatures and forest cover in the High Plains states in the United States

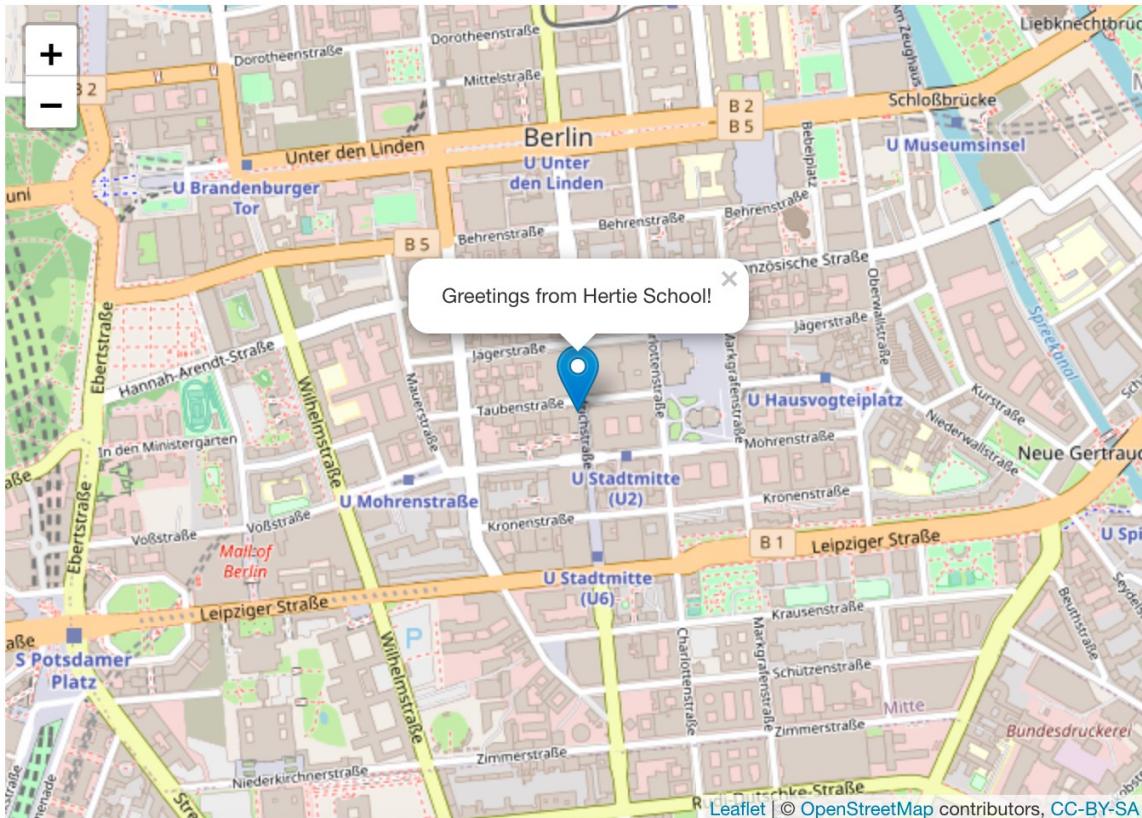


Basic Leaflet workflow



Basic Leaflet syntax

```
install.packages("leaflet")
```



```
library(leaflet)
```

```
first_map <- leaflet() %>%  
  addTiles() %>% # Add default OpenStreetMap map tiles  
  addMarkers(lng=13.3893204,  
            lat=52.5128055,  
            popup="Greetings from Hertie School!")  
first_map # Print the map
```

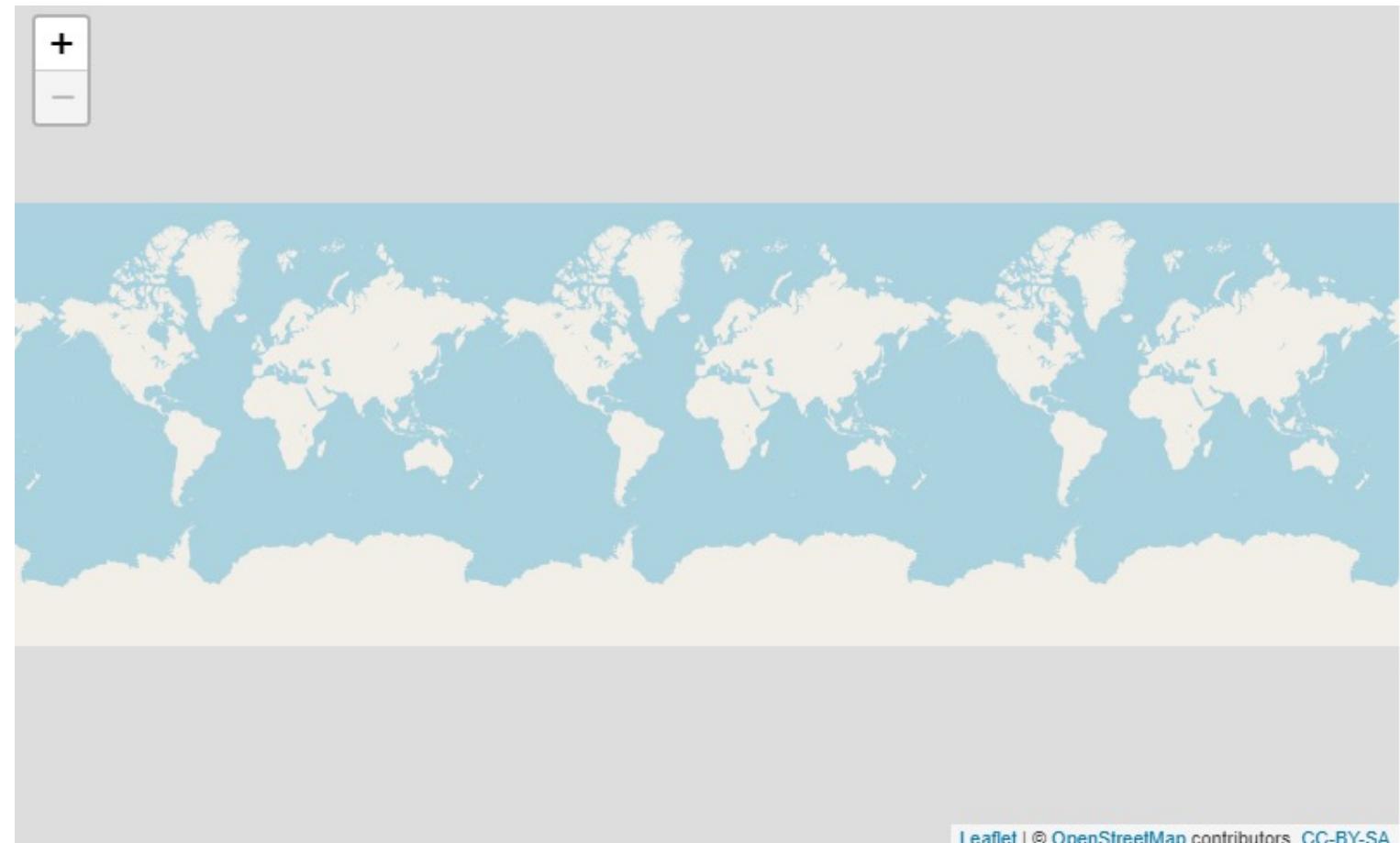
1. Create a map widget by calling `leaflet()`.
2. Add layers (i.e., features) to the map by using layer functions (e.g. `addTiles`, `addMarkers`, `addPolygons`).
3. Repeat step 2 as desired.
4. Print the map widget to display it.

Basemaps: Default map tiles

Similar to Google Maps, Leaflet supports basemaps (i.e. the underlying map) using map tiles.

We can use the default basemap, which is *OpenStreetMap*, using ***addTiles()*** with no arguments:

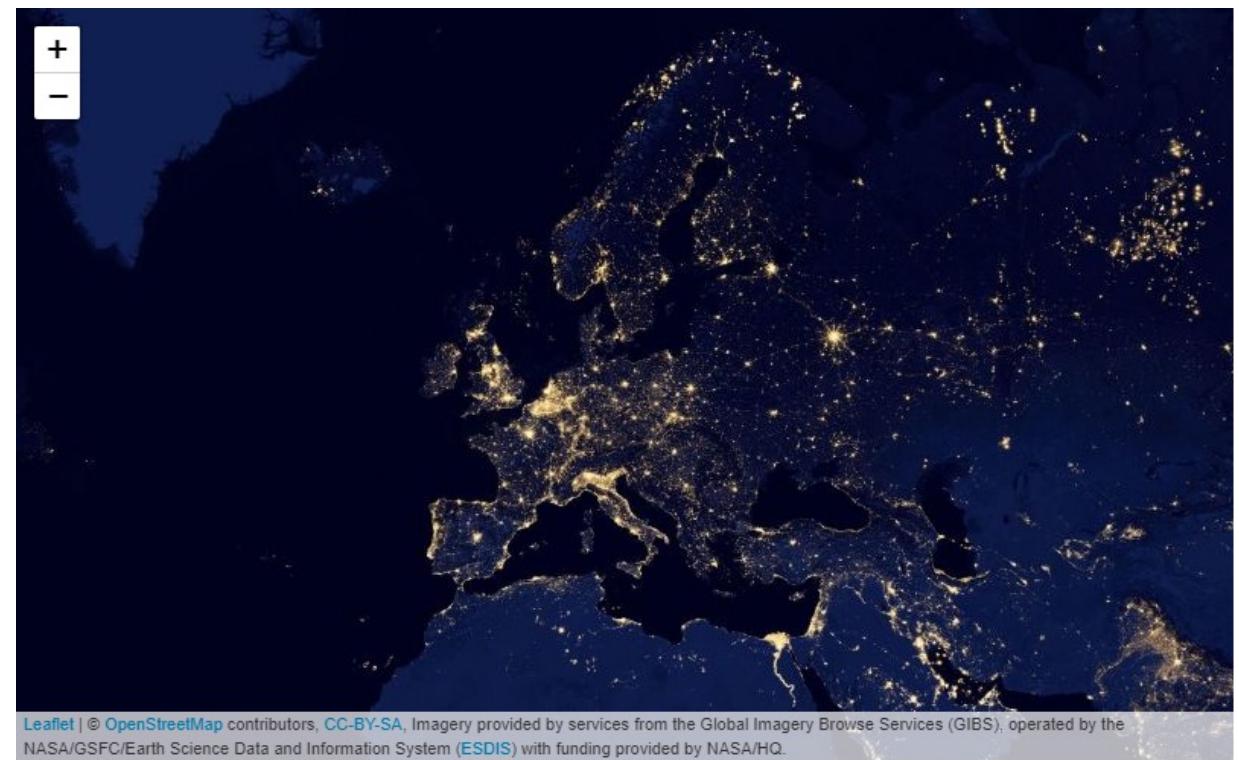
```
leaflet() %>%  
  addTiles()
```



Basemaps: Third-party map tiles

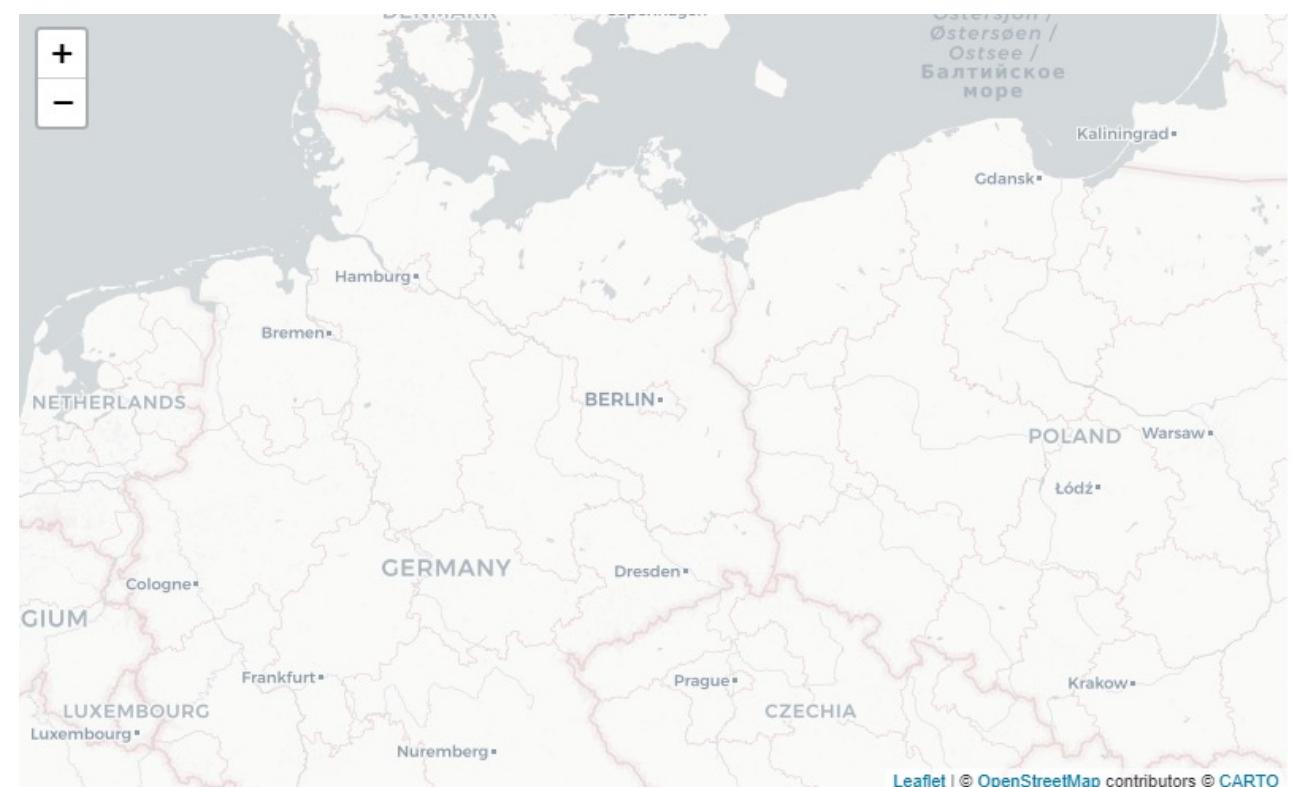
You can also use one of many useful and interesting [third-party tiles](#) with the *addProviderTiles()* function:

```
leaflet() %>%  
  setView(lat=52.5, lng=13.3, zoom=3) %>%  
  addTiles() %>%  
  addProviderTiles("NASAGIBS.ViirsEarthAtNight2012")
```



Basemaps: Third-party map tiles

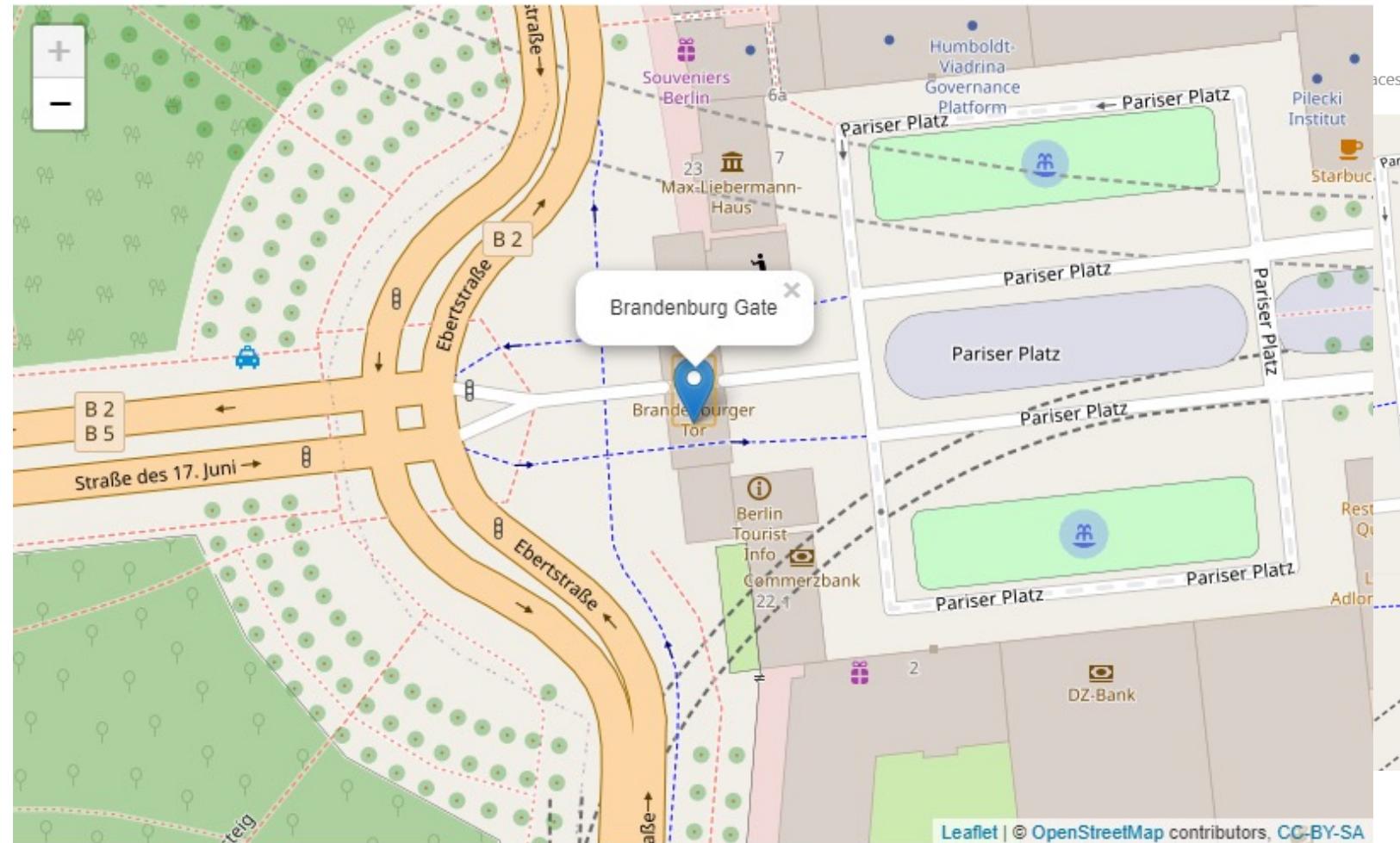
```
leaflet() %>%
  setView(lat=52.5, lng=13.3, zoom=3) %>%
  addTiles() %>%
  addProviderTiles("providers$CartoDB.Positron")
```



Layer features: Markers & Popups

We can use ***addMarkers()*** to specify locations on a map with **markers** and add **popups** to show text when markers are clicked.

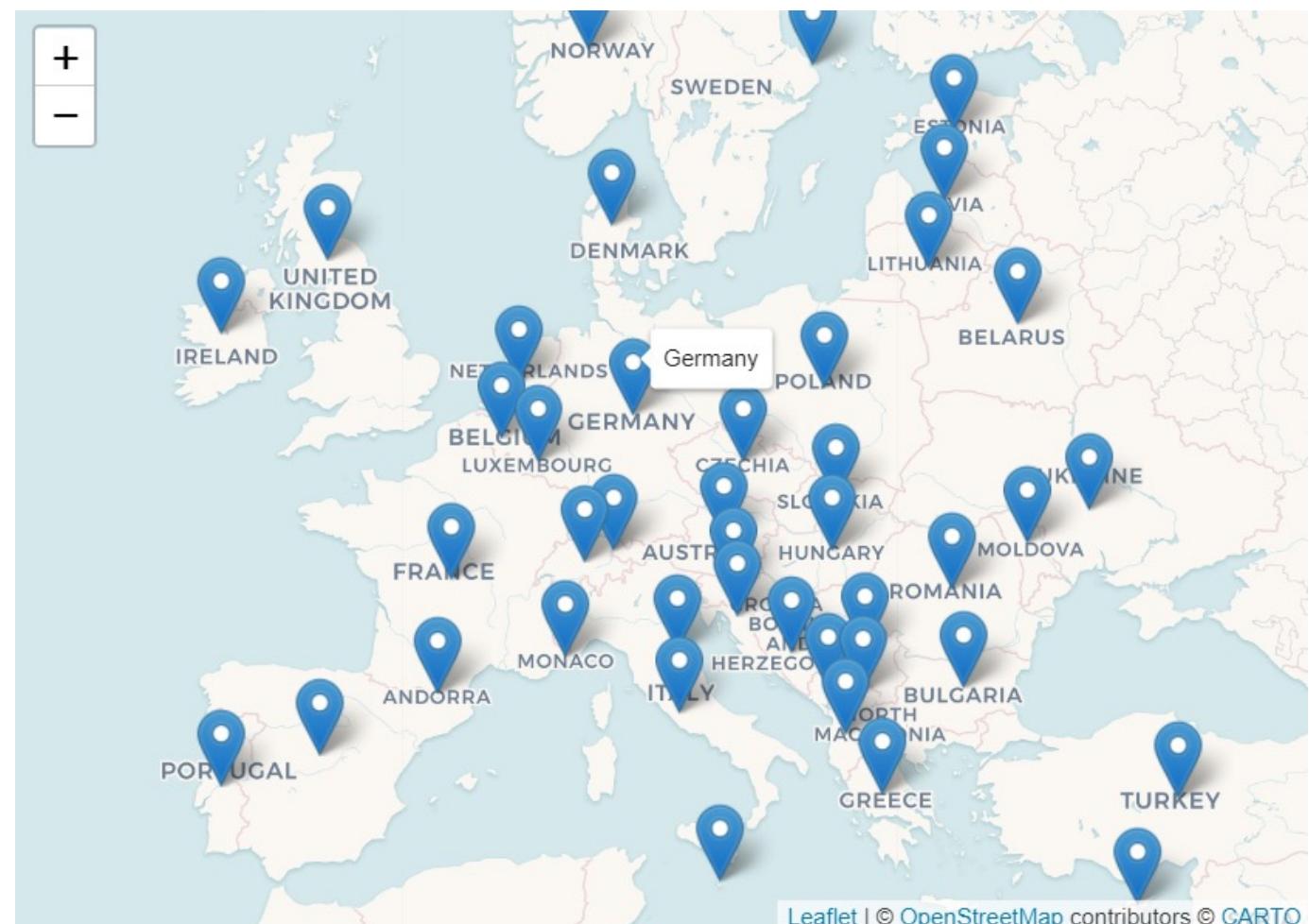
```
leaflet() %>%  
  addTiles() %>%  
  addMarkers(lng=13.377775,  
            lat=52.516266,  
            popup="Brandenburg Gate")
```



Layer features: Labels

We can also use ***addMarkers()*** to add **labels**. Unlike popups, you don't need to click a marker for the label to be shown. Instead, labels are displayed on mouse over.

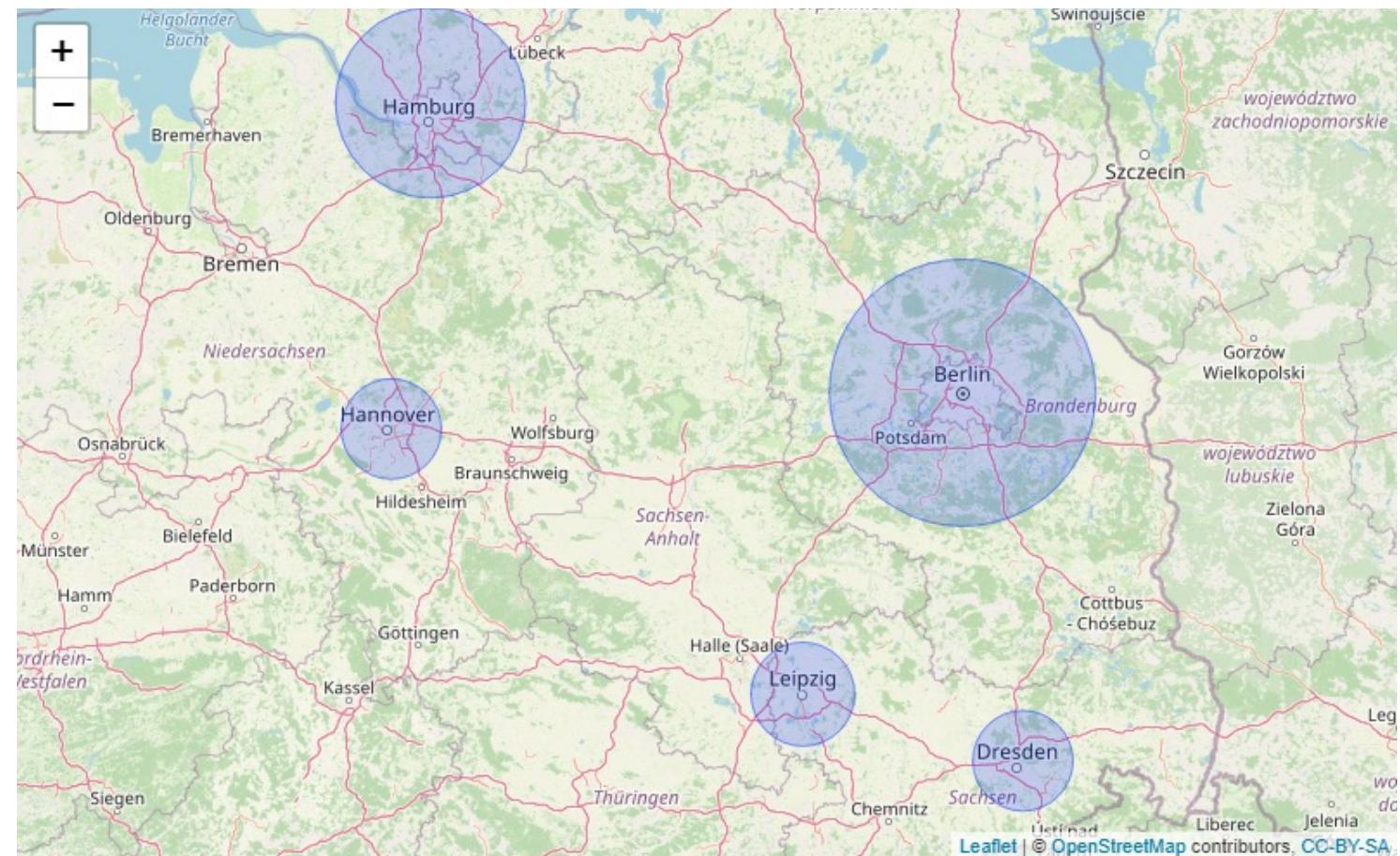
```
data %>%  
  leaflet() %>%  
  addProviderTiles(providers$CartoDB.Voyager) %>%  
  
  addMarkers(lng = ~Longitude,  
            lat = ~Latitude,  
            label = ~country)
```



Layer features: Circles

Circles can be added using *addCircles()*.

```
data %>%
  leaflet() %>%
  addTiles() %>%
  addCircles(lng = ~Long,
             lat = ~Lat,
             weight = 1,
             radius = ~sqrt(Pop) * 30,
             popup = ~City)
```

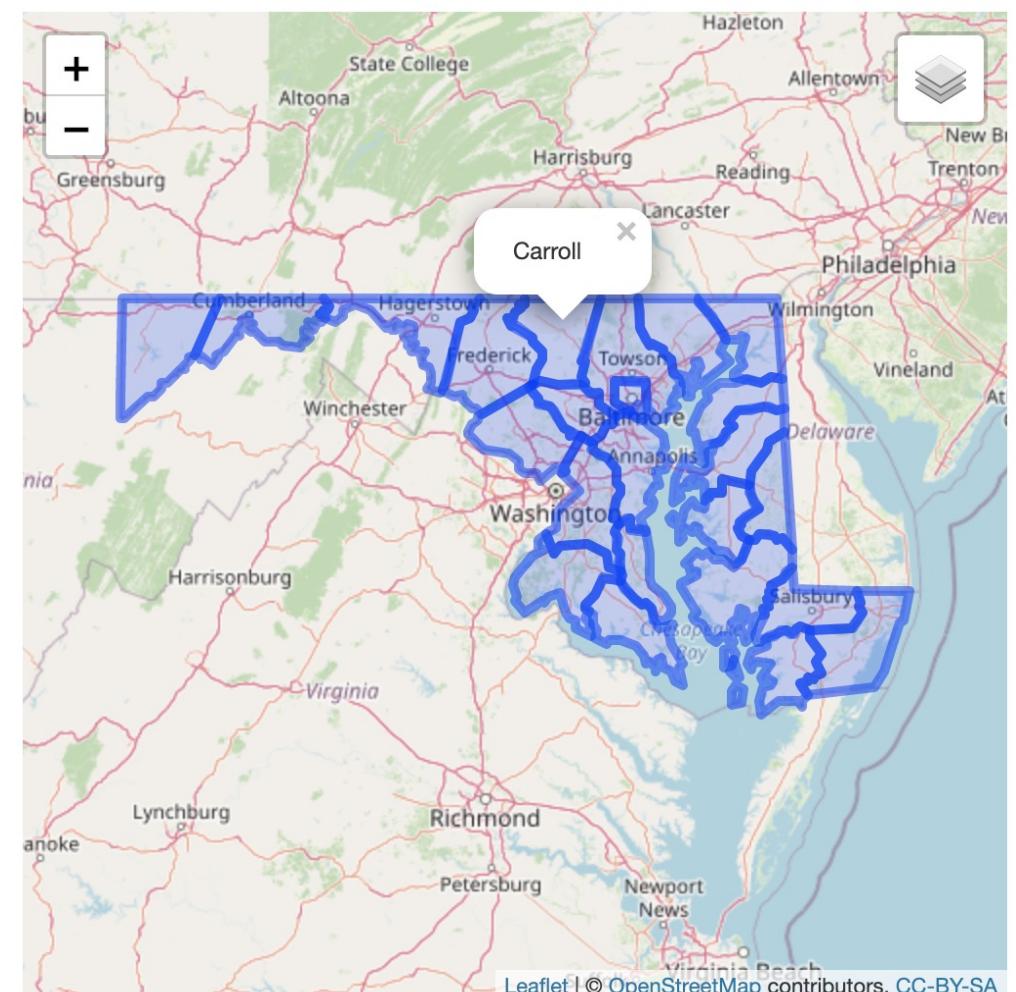


Layer features: Polygons

In mathematics, a polygon is defined as a **two-dimensional** figure with **three or more sides** (e.g., triangles or octagons).

In the GIS world, polygons often have often **irregular lines** (e.g., country borders, lakes).

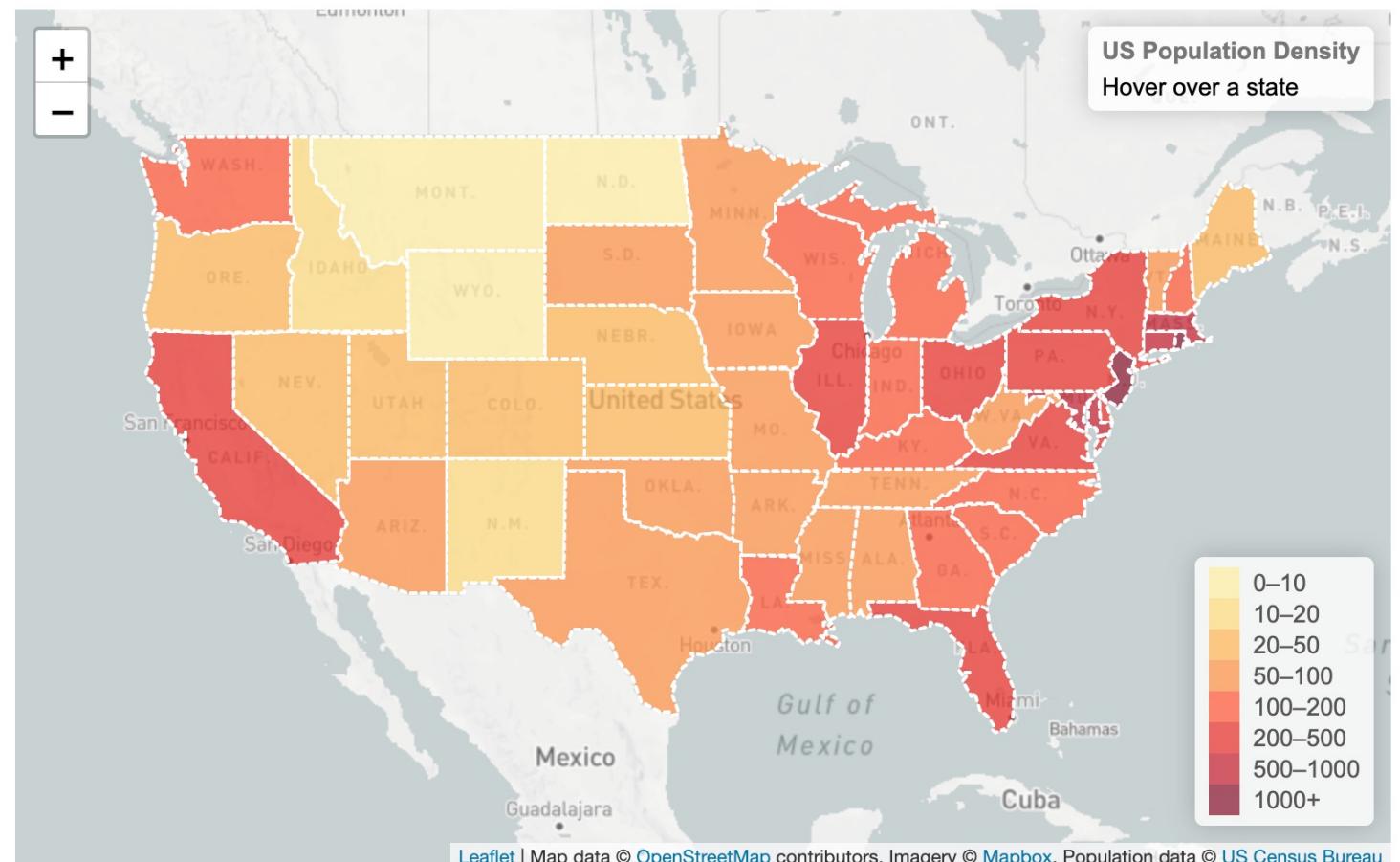
Polygons lay the groundwork for creating **choropleth maps** (see next slide).



Layer features: Choropleths

Choropleth maps display geographical areas or regions that are **colored or shaded according to a variable**.

Using **color progression**, choropleth maps provide a way visualize **variation or patterns** across locations.



Further references and resources (1)

Leaflet for R

Introduction

Leaflet is one of the most popular open-source JavaScript libraries for interactive maps. It's used by websites ranging from [The New York Times](#) and [The Washington Post](#) to [GitHub](#) and [Flickr](#), as well as GIS specialists like OpenStreetMap, Mapbox, and CartoDB.

This R package makes it easy to integrate and control Leaflet maps in R.

Features

- Interactive panning/zooming
- Compose maps using arbitrary combinations of:
 - Map tiles
 - Markers
 - Polygons
 - Lines
 - Popups
 - GeoJSON
- Create maps right from the R console or RStudio
- Embed maps in knitr/R Markdown documents and Shiny apps
- Easily render spatial objects from the sp or sf packages, or data frames with latitude/longitude columns
- Use map bounds and mouse events to drive Shiny logic
- Display maps in non spherical mercator projections
- Augment map features using chosen plugins from [leaflet plugins repository](#)

Installation

To install this R package, run this command at your R prompt:

```
install.packages("leaflet")
# to install the development version from Github, run
# devtools::install_github("rstudio/leaflet")
```

Once installed, you can use this package at the R console, within R Markdown documents, and within Shiny applications.

Basic Usage

You create a Leaflet map with these basic steps:

- Create a map widget by calling `leaflet()`.
- Add layers (i.e., features) to the map by using layer functions (e.g., `addTiles`, `addMarkers`, `addPolygons`) to modify the map widget.
- Repeat step 2 as desired.
- Print the map widget to display it.

Here's a basic example:

```
library(leaflet)

m <- leaflet() %>%
  addTiles() %># Add default OpenStreetMap map tiles
  addMarkers(lat=174.768, lon=-36.852, popup="The birthplace of R")
m # Print the map
```



- Official documentation and tutorial for [Leaflet in R](#)
- Leaflet [cheat sheet](#)

Leaflet Cheat Sheet

an open-source JavaScript library for mobile-friendly interactive maps

Quick Start

Installation

```
Use install.packages("leaflet") to install the package or directly from GitHub devtools::install_github("rstudio/leaflet").
```

First Map

```
m <- leaflet() %>%
  addTiles() %># Set the map with the pipe operator
  addTiles()
  addMarker(lat = -36.852, lon = 174.768, popup = "The birthplace of R")
  # add a single point layer
```



Markers

Use markers to call out points, express locations with latitude/longitude coordinates, appear as icons or as circles.

Icon Markers

```
markerOptions(..., icon=icon) # Set the icon for the marker
markerOptions(..., iconColor=iconColor, iconSize=iconSize, iconAnchor=iconAnchor, shadowColor=shadowColor, shadowSize=shadowSize, shadowOffset=shadowOffset, iconWidth=iconWidth, iconHeight=iconHeight, iconAngle=iconAngle, iconLabel=iconLabel) # Customize marker icons
```

Awesome Icons

Awesome icons customizable with colors and icons

addAwesomeMarkers(..., markerOptions, awesomeIcons, awesomeIconList)

clusterOptions = markerClusterOptions()

freezeAtZoom Freeze the cluster at assigned zoom level

Circle Markers

```
addCircleMarkers(color, radius, stroke, opacity, fillOpacity...) # Customise their color, radius, stroke, opacity
```

GeoJSON and TopoJSON

There are two options to use the GeoJSON/TopoJSON data.

- To read into sp objects with the `geojsonio` or `rgdal` package:
`geojsonio::geojson_read("what-ever.rgtl"; rgdal::readOGR(..., "OGGeoJSON"))`
- To use the `sf` package:
`sf::st_read("what-ever.geojson", encoding="utf-8", driver="GeoJSON", weight="color", color, fill, opacity, fillOpacity...)`

Styles can also be tuned separately with a style: (...) object.

Other packages including `geojsonio` and `jsonlite` can help fast parse or generate the data needed.

Shiny Integration

To integrate a Leaflet map into an app:

- In the UI, call `leafletOutput("name")`
- On the server side, assign a `renderLeaflet(...)` call to the output
- Inside the `renderLeaflet` expression, return a Leaflet map object

Modification

To modify an existing map or add incremental changes to the map, you can use `leafletProxy()`. This should be performed in an observer on the server side.

Popups and Labels

Other useful functions to edit your map:

```
fitBounds(..., ...), fitView() # Set the view to contain these bounds
addCircles(..., ..., ..., ..., ..., ..., ...) # Create circles with layers of "A", "B", "C"...
removeShape(..., ...) # Remove some of the circles
clearShapes() # Clear all circles and other shapes
```

Inputs/Events

Object event names generally use this pattern:

```
inputs$MAPID.$OBJECTCATEGORY_EVENTNAME
```

Triggered when the value of the \$NAME input at this variable valid values for \$OBJECTCATEGORY are click, mouseover and mouseout. Valid values for EVENTNAME are click, mouseover and mouseout.

All of these events are set to either `NULL` if the event has never happened, or a `list` that includes:

- lat: The latitude of the object, if available; otherwise, the mouse cursor's latitude if the object, if available; otherwise, the mouse cursor's latitude if the object, if any
- lon: The longitude of the object, if available; otherwise, the mouse cursor's longitude if the object, if available; otherwise, the mouse cursor's longitude if the object, if any

GeoJSON events also include additional properties:

- * `featureId`: The feature ID, if any
- * `properties`: The feature properties

Map Events

inputs\$MAPID.click when the map background or basemap is clicked
value = > list with lat and lon

inputs\$MAPID_bounds provide the listing bounds of the visible map area
value = > list with north, east, south and west

inputs\$MAPID_zoom an integer indicates the zoom level

Keja Shi © Data Science Institute, Columbia University in the City of New York, Keja.Shi@Columbia.edu

Adapted from RStudio materials <https://rstudio.github.io/leaflet/> 2019-01-01

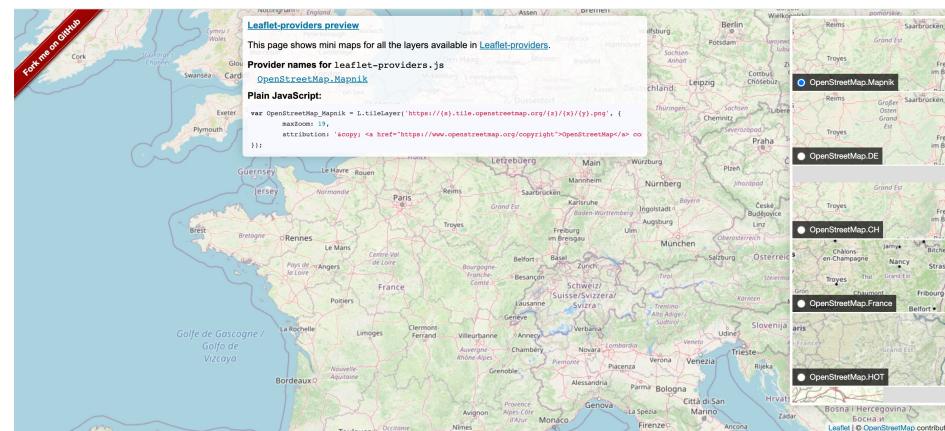
Further references and resources (2)

- Hands-on [YouTube tutorial](#) for basic Leaflet features
- Collection of free [third-party basemaps](#) (map tiles) for Leaflet
- Leaflet website for [JavaScript](#)

A screenshot of an RStudio session titled "Leaflet Mapping in RStudio - Introduction". The code editor shows a script with Leaflet imports and a map view. The map displays a coastal area with green land and blue water.

```
1 # Load the leaflet package
2 library(leaflet)
3
4 # Create a map object
5 m <- leaflet()
6 addTiles(m)
7
8 # Add a marker
9 m %>% addMarker(lat=51.5, lng=-0.1, title="London")
```

Leaflet Mapping in RStudio - Introduction
14.264 Aufrufe · 31.01.2017
225 6 TEILEN SPEICHERN ...
Brent Thorne 452 Abonnenten
ABONNIEREN



A screenshot of the Leaflet website homepage. It features a large map of London with a callout bubble pointing to a specific location. The page includes a header with the Leaflet logo, social media links, and navigation links for Overview, Tutorials, Docs, Download, Plugins, and Blog. A news banner at the bottom left announces "Sep 4, 2020 — Leaflet 1.7.1 has been released!"

Leaflet
an open-source JavaScript library for mobile-friendly interactive maps
Overview Tutorials Docs Download Plugins Blog
Sep 4, 2020 — Leaflet 1.7.1 has been released!
Leaflet is the leading open-source JavaScript library for mobile-friendly interactive maps. Weighing just about 39 KB of JS, it has all the mapping features most developers ever need.
Leaflet is designed with simplicity, performance and usability in mind. It works efficiently across all major desktop and mobile platforms, can be extended with lots of plugins, has a beautiful, easy to use and well-documented API and a simple, readable source code that is a joy to contribute to.

A pretty CSS3 pop-up. Easily customizable.

Coming up: Live tutorial

