Oct 2024

R Shiny Masterclass Series - Introduction

Maps and spatial visualisation with Leaflet

EPI-interactive

Agenda

- **Session 1** | 30 September | Getting started with Posit Cloud and your first R Shiny app
- **Session 2** | 01 October | R Shiny core concepts and mobile ready layout
- **Session 3** | 03 October | R Shiny user interface components, reactivity and debugging
- **Session 4** | 07 October | Data sources and data processing in R Shiny
- **Session 5** | 08 October | Maps and spatial visualisation with Leaflet: adding map layers, annotations, pins, filters and legend
- **Session 6** | 10 October | Interactive charts with Plotly: chart types, customising hover boxes and chart styling
- **Session 7** | 14 October | Publishing R Shiny apps, design considerations and case study
- **Session 8** | 15 October | Case study, top 10 tips for data visualisation with R Shiny and wrap-up

Today

Recap: Session 4 challenge

Goals:
Understand leaflet capabilities – AIS Explorer
Create leaflet data visualization

Steps:
Add base map / markers to Shiny app
Add data to map
Add reactive behaviour to shapes on the map
Customise pop-ups

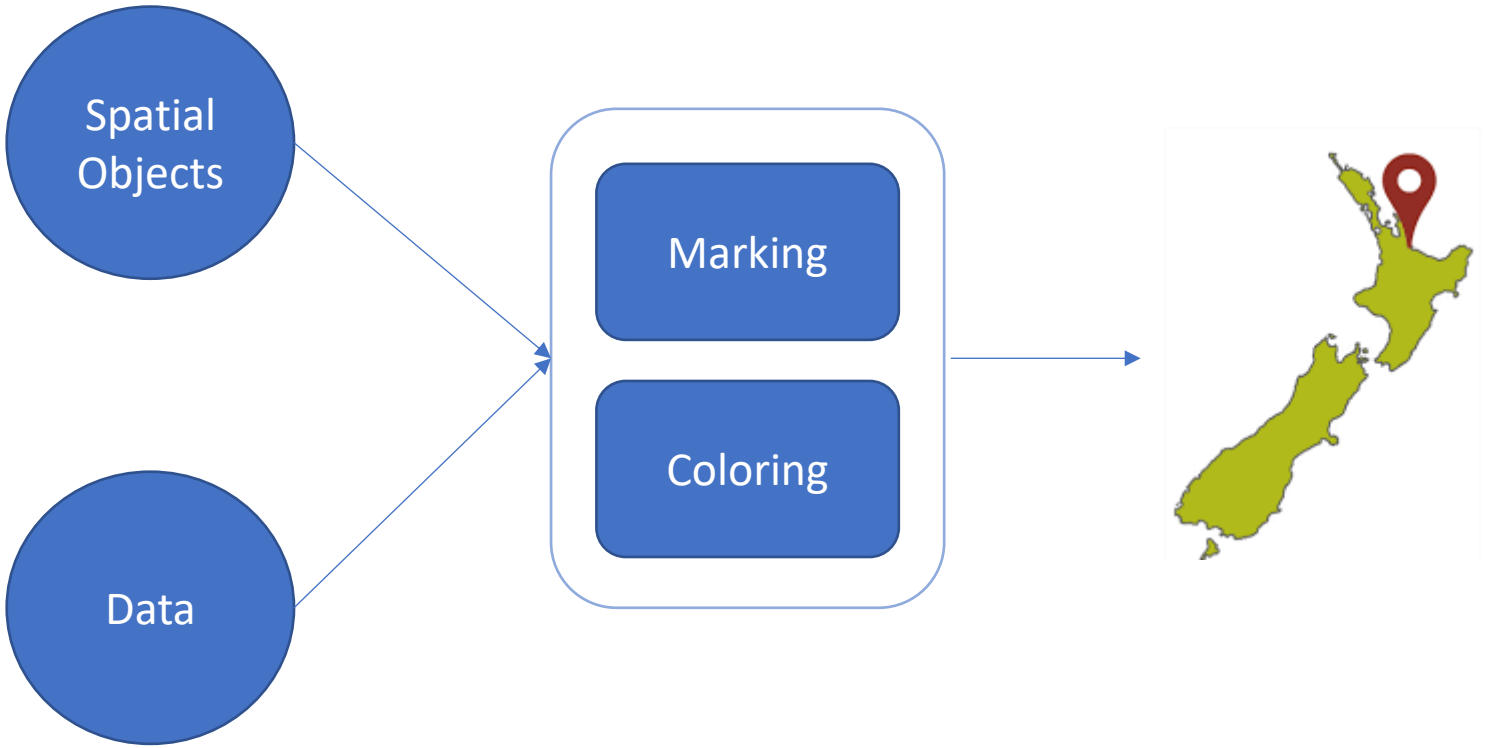
Introduction to Leaflet

Why Leaflet?

- A popular open-source JavaScript library
- Easy to integrate and control Leaflet maps in R
- Interactive panning/zooming
- Easily composing maps using combinations of Map tiles/ Markers / Polygons / Lines / Popups / GeoJSON
- Easily render spatial objects from the *sp* or *sf* packages, or data frames with latitude/longitude columns

Reference: https://rstudio.github.io/leaflet/

Leaflet working flow chart

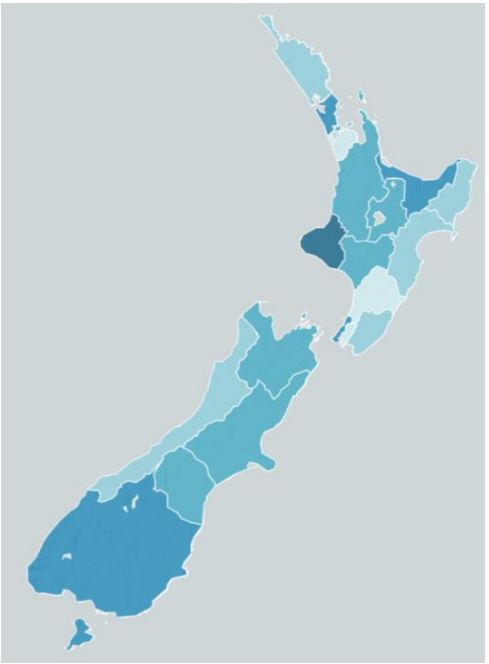


Leaflet object - a spatial vector

The sp/sf package:

To handle geographic vector data

An sf example: District Health Board, New Zealand



```
Simple feature collection with 20 features and 2 fields
geometry type:  MULTIPOLYGON
dimension:      XY
bbox:           xmin: 166.4293 ymin: -47.28706 xmax: 178.5497 ymax: -34.41455
epsg (SRID):    4326
proj4string:    +proj=longlat +datum=WGS84 +no_defs
First 10 features:
  DHB12               NAME                       geometry
0     6             Lakes MULTIPOLYGON (((176.6977 -3...
1    10        Hawke's Bay MULTIPOLYGON (((177.1222 -3...
2     3           Auckland MULTIPOLYGON (((174.8427 -3...
3    18         Canterbury MULTIPOLYGON (((174.0488 -4...
4     2          Waitemata MULTIPOLYGON (((174.6907 -3...
5    16 Nelson Marlborough MULTIPOLYGON (((173.9564 -4...
6    17         West Coast MULTIPOLYGON (((172.479 -42...
7     5            Waikato MULTIPOLYGON (((175.9401 -3...
8     9           Taranaki MULTIPOLYGON (((174.6152 -3...
9    13               Hutt MULTIPOLYGON (((175.1509 -4...
```

EPI-interactive

File Structure

- UI:
  `leafletOutput("map_id")`

- Server:

```
output$map_id <- renderLeaflet({
        leaflet(…)
})
```

Common server functions

- **leaflet():** Base of the map; creates the overall map projection. Data can be provided to this
- **addTiles() / addProviderTiles:** Creates a base layer of information to be presented on the map. Different tiles can be selected for different mapping purposes
- **addMarkers / addCircles / addLines:** Renders simple features onto the map. Requires lat / long coordinates.
- **addPolygons():** Renders shape files onto the map as complex polygons. Details such as the border colour, fill colour, opacity and pop-up behaviours can be customised. Requires a spatial data frame.
- **clearGroup():** Removes a previously rendered group of data from the map

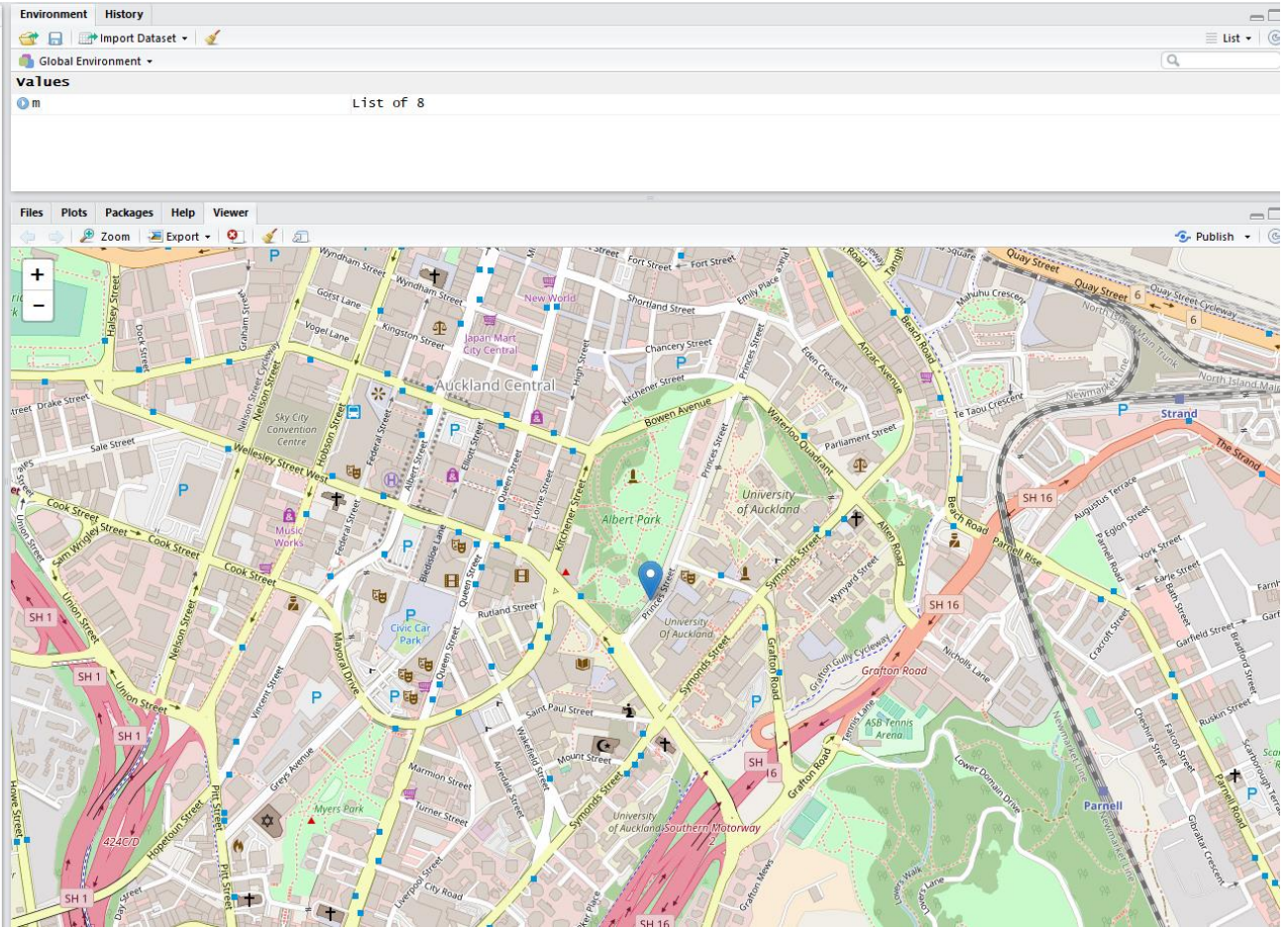
Basic Usage

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

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

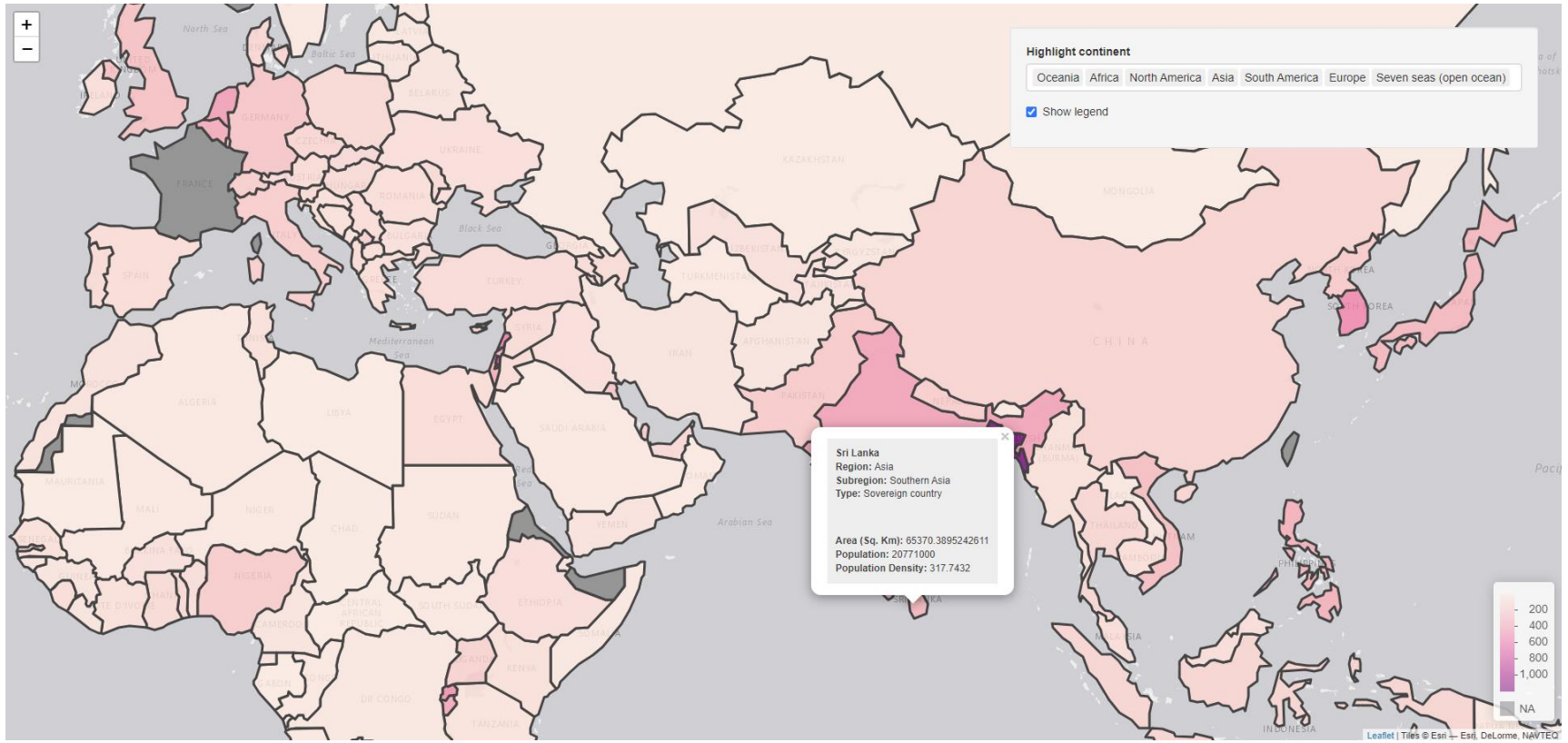
Basic Usage

```
Console ~/

R version 3.4.1 (2017-06-30) -- "Single Candle"
Copyright (C) 2017 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> library(leaflet)
warning message:
package 'leaflet' was built under R version 3.4.3
>
> m <- leaflet() %>%
+     addTiles() %>%  # Add default OpenStreetMap map tiles
+     addMarkers(lng=174.768, lat=-36.852, popup="The birthplace of R")
> m  # Print the map
>
```



EPI-interactive

Visualising world data with Leaflet



Creating the base map

Creating the base map

In Posit Cloud, open **Session-5**, then /stage1

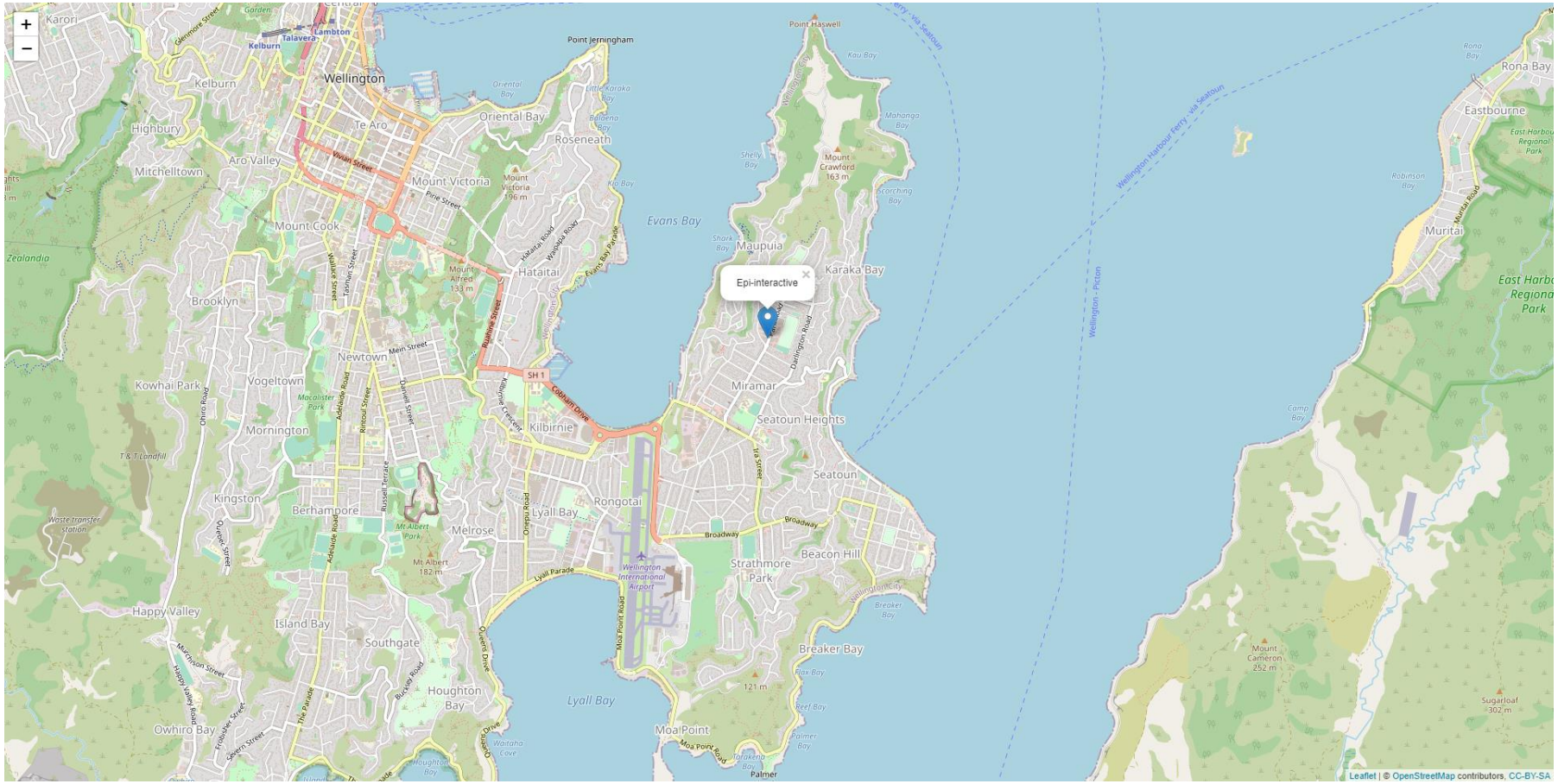
- Create a leafletOutput for our map:

```
leafletOutput("map", width = "100%", height = "100%")
```

- Use render function in the server side:

```
output$map <- renderLeaflet({
    leaflet() %>% addTiles() %>%
    addMarkers(lng=174.821029, lat=-41.309602, popup="Epi-interactive")
 })
```

Creating the base map



Creating the base map

- addProviderTiles() can be used instead of addTiles() to customise the background of leaflet maps.
- Change style and purpose of leaflet maps
- Many options available: http://leaflet-extras.github.io/leaflet-providers/preview/index.html

```
output$map <- renderLeaflet({
    leaflet() %>%
    addProviderTiles(providers$Esri.WorldGrayCanvas) %>%
     addMarkers(lng=174.821029, lat=-41.309602, popup="Epi-interactive")
 })
```

Rendering spatial data

Rendering spatial data

- For simple features:
  - addCircles(lng, lat, radius, layerId, group, stroke, color, weight, …)
  - addMarkers(lng, lat, layerId, group, popup, label, …)
  - AddLines()

- For more complex features:
  - addPolygons()
  - addPolyLines
  - addRasterImage

Rendering spatial data

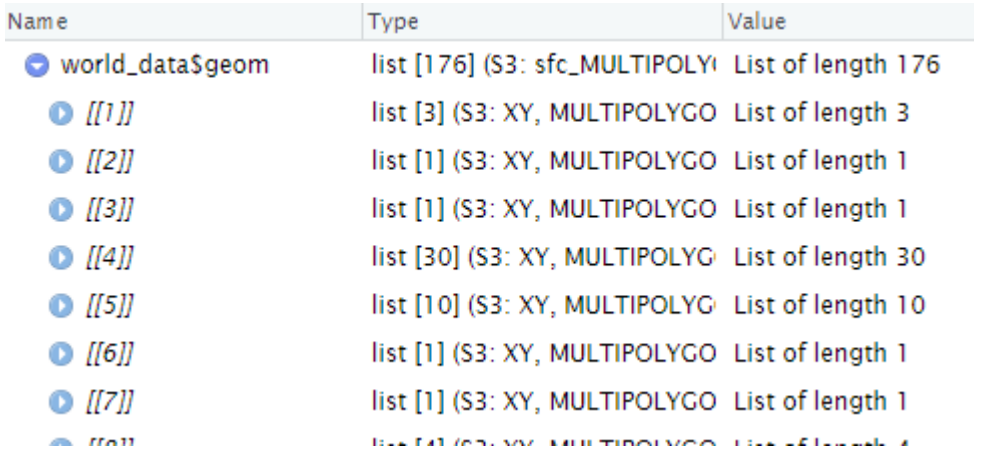
- To remove layers:
  - clearShapes()
  - clearGroup(“group_name”)
  - clearBounds()
  - clearTiles()

- Common parameters
  - **lng / lat:** longitude and latitude of data
  - **layerId:** a unique identifier for objects in the layer (e.g. iso_a2)*
  - **group:** a named identifier for the current layer of data
  - **data:** which data should be used for this layer

AddPolygons

- Requires a *geom* (geometry) column in the provided data

- Customisations:
  - stroke, color, weight, opacity
  - fill, fillColor, fillOpacity
  - smoothFactor
  - popup



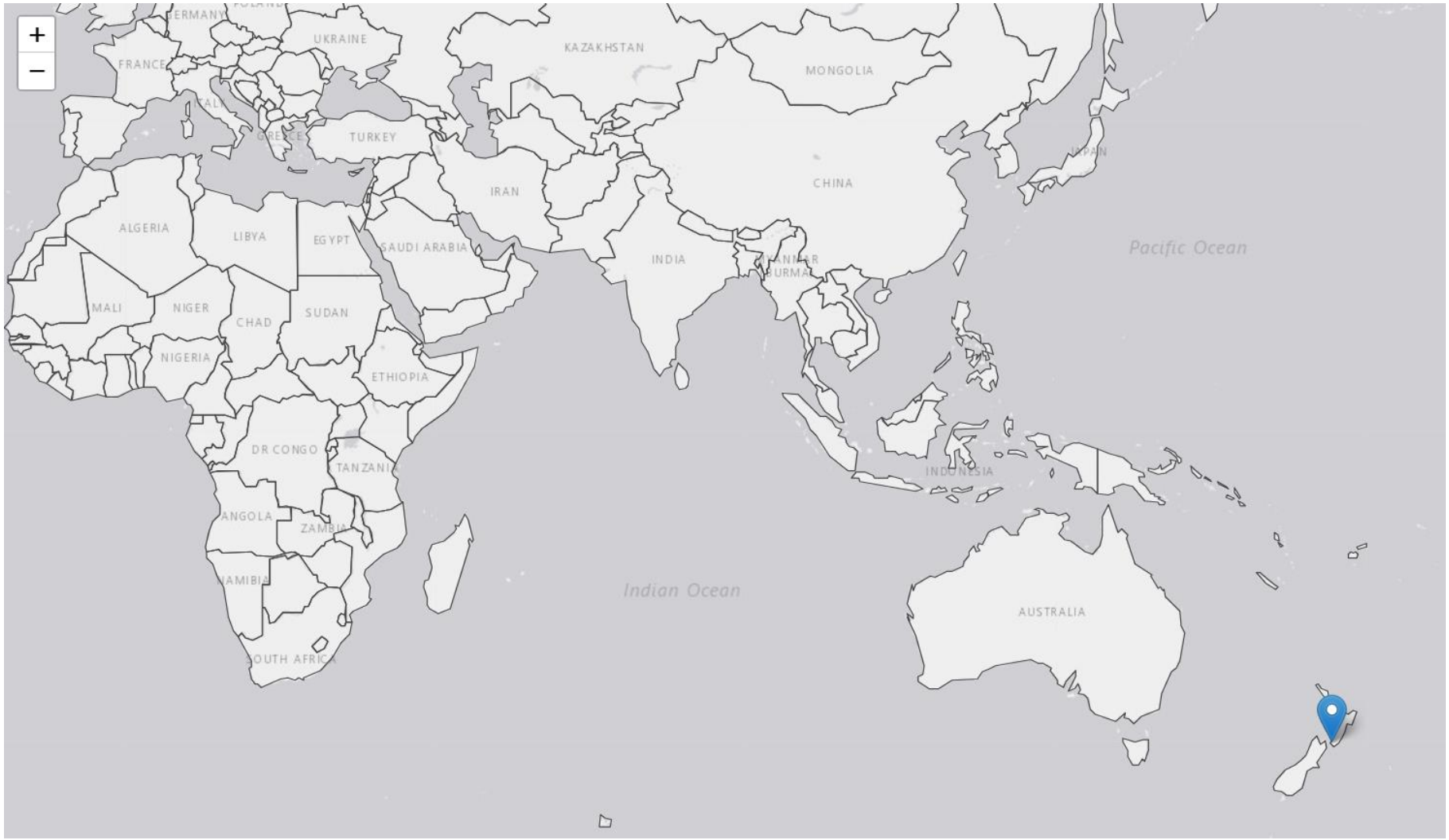
Name	Type	Value
world_data$geom	list [176] (S3: sfc_MULTIPOLY(	List of length 176
[[1]]	list [3] (S3: XY, MULTIPOLYGO	List of length 3
[[2]]	list [1] (S3: XY, MULTIPOLYGO	List of length 1
[[3]]	list [1] (S3: XY, MULTIPOLYGO	List of length 1
[[4]]	list [30] (S3: XY, MULTIPOLYG	List of length 30
[[5]]	list [10] (S3: XY, MULTIPOLYG	List of length 10
[[6]]	list [1] (S3: XY, MULTIPOLYGO	List of length 1
[[7]]	list [1] (S3: XY, MULTIPOLYGO	List of length 1

Rendering spatial data

If required, open /stage2 to continue

- Use addPolygons to render the country shapes from world_data onto the map
- Modify the shape borders to have a weight of 1px
- Disable the shape fill so that only the outline is drawn

Rendering spatial data



Reactive controls

Leaflet Proxy

- Rendering data onto a map is expensive (takes a lot of resources)
- Shouldn’t redraw entire map any time data changes
- Leaflet Proxy allows data to update without changing whole map.
- Need to decide when we draw certain elements

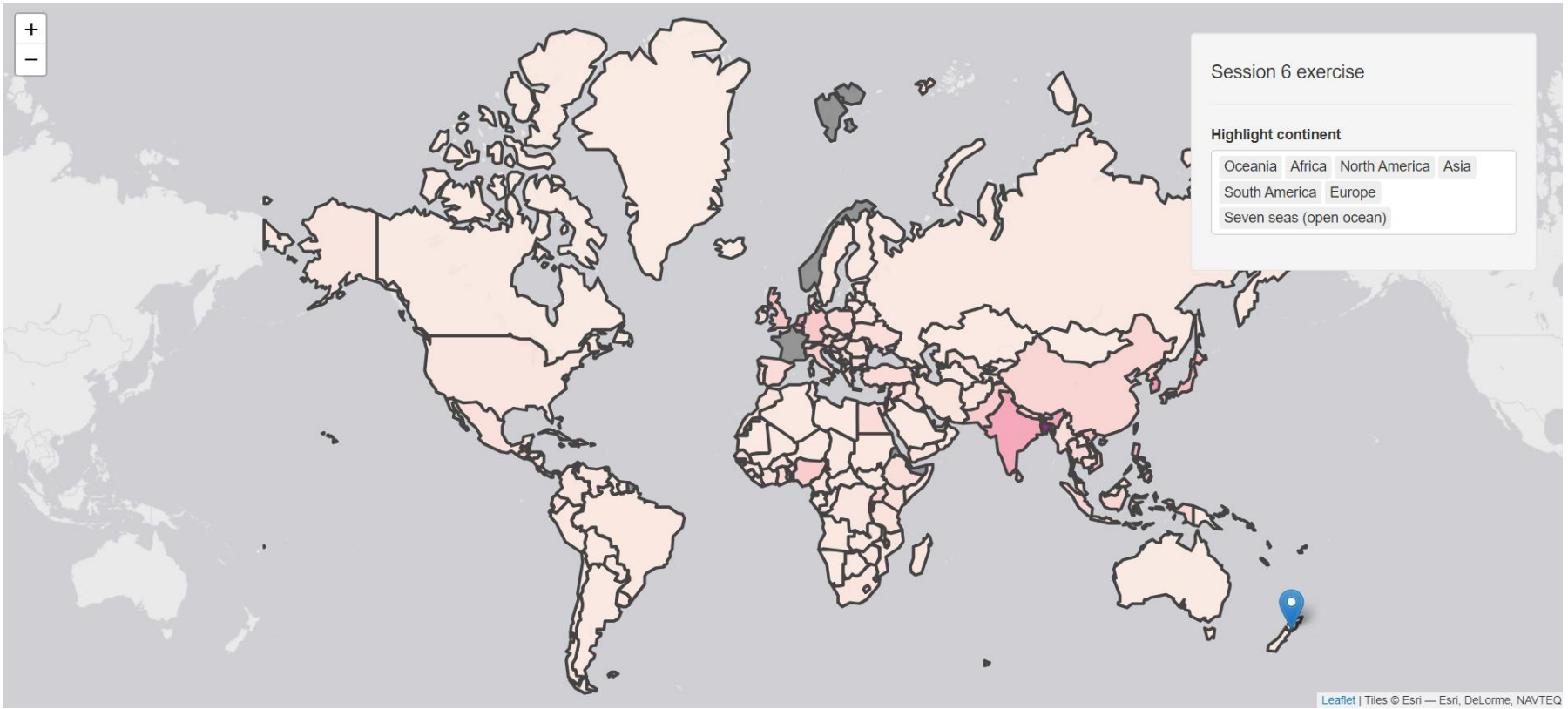
```
observe({
   pal <- colorpal()
   leafletProxy("map") %>%
      clearGroup("highlight") %>%
      addPolygons(data = continent_data(), group = "highlight",  …)
})
```

Reactive controls

*If required, open **/stage3** to continue:*

- In ui.R, add to the absolutePanel a selectInput to choose ***multiple*** continents (all selected by default)
- In server.R, create a reactive to filter world_data by your continent input
- Create an observe() and use leafletProxy with addPolygons to render the selected continent shapes with a thicker border

Reactive controls



Legends and Popups

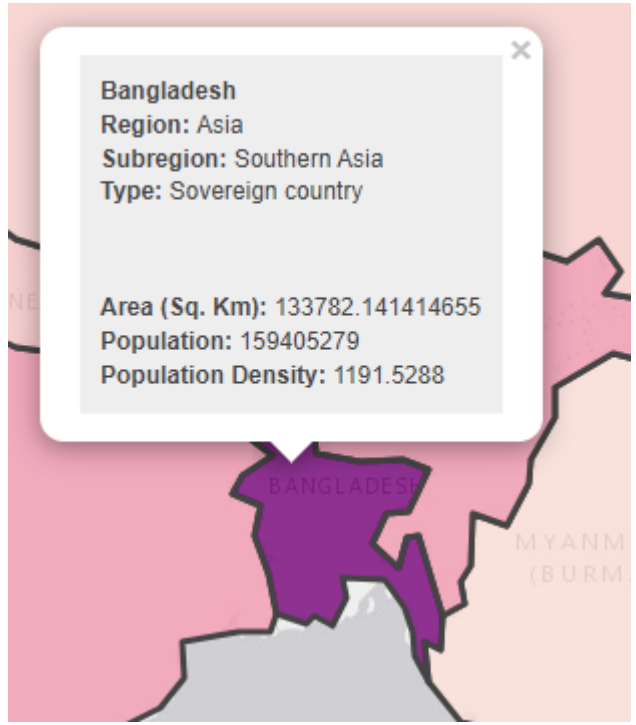
Legends

- Leaflet legends require some manual set-up
  - `addLegend(map, position, pal, values)`
  - `clearControls()`

```
observe({
   proxy <- leafletProxy("map", data = filteredData())

    pal <- colorpal()  #colorpal() is the colour palette to render map
       proxy %>% addLegend(position = "bottomright",
          pal = pal, values = ~world_data$popDensity )
  }
 })
```

Popups

- We can create HTML formatted pop-up text for each shape
- Use dplyr::mutate to create a popupText column
- Use paste / paste0 / sprintf to construct the text



Legends and Popups

*If required, open **/stage4** to continue:*

- Add to ui.R a checkboxInput to show / hide the legend
- Add to server.R an observe() which uses a leafletProxy to add / remove the legend from the map

Legends and Popups

Recap

1. Use *leaflet* to create base map
2. Customise look and purpose of map with provider tiles
3. Render shapes onto the map with addPolygons.
4. Reactively modify existing maps with *leafletProxy.* Filter data with Shiny inputs and reactivity
5. Customise extra information such as colours, legend, popups

Next time

- Data visualisation with Plotly

- **Challenge: using your existing project (or the /result folder):**
- Create a new selectInput and use this to change the category variable
- Create a new selectInput to change the providerTiles and colours used to fill the shapes on the map
- Customise the shapes to have a highlight effect when you hover with your mouse
  - Hint: see https://rstudio.github.io/leaflet/articles/choropleths.html for an example
- Change the shape popup text into a label that appears on hover (check the 'label' attribute)
- Add a downloadHandler to export the data currently shown on the map
- Share your results on the session 6 forum.