Week 12: *Interactive Charts*

EMSE 4572: Exploratory Data Analysis

John Paul Helveston

November 16, 2022

Week 12: *Interactive Charts*

1. Interactive charts
2. Interactive tables

Intermission

3. Interactive maps

Week 12: *Interactive Charts*

1. Interactive charts

2. Interactive tables

Intermission

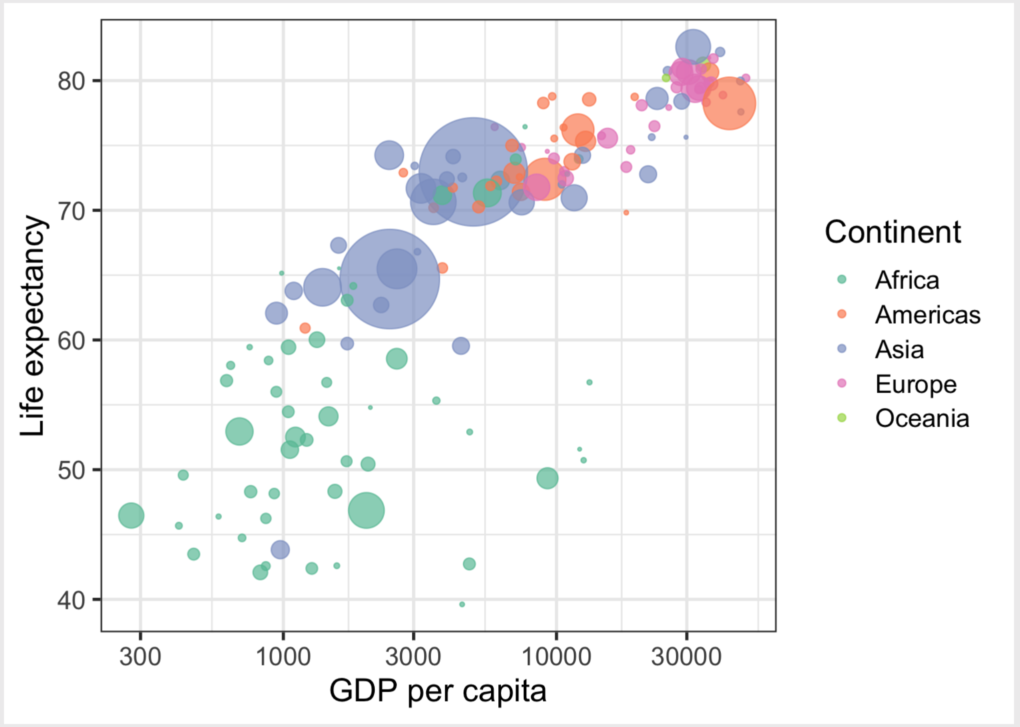
3. Interactive maps

Plotly uses JavaScript to create interactive charts

But you don't have to know JavaScript to use it! 🎉

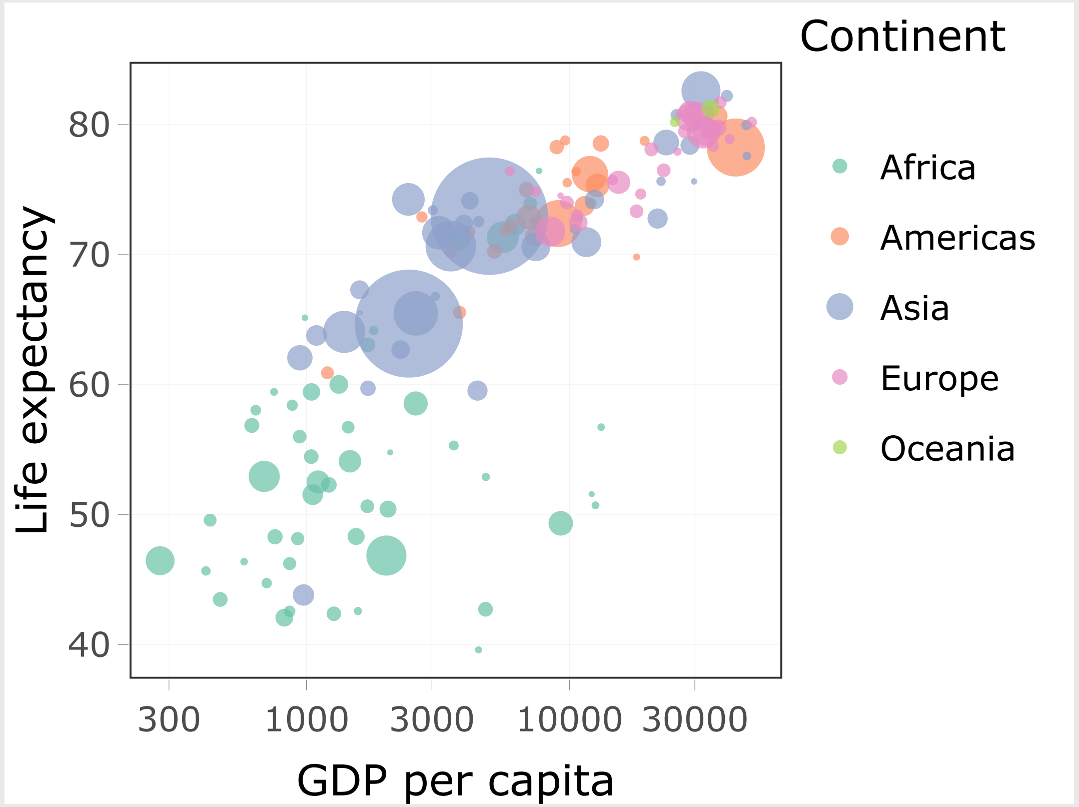
Turn any ggplot into an interactive chart with `ggplotly()`

```r
plot <- gapminder %>%
  filter(year == 2007) %>%
  ggplot(aes(x = gdpPercap, y = lifeExp,
             size = pop, color = continent,
             label = country)) +
  geom_point(alpha = 0.7) +
  scale_color_brewer(palette = 'Set2') +
  scale_size_area(
    guide = FALSE, max_size = 25) +
  scale_x_log10() +
  theme_bw(base_size = 16) +
  labs(x = 'GDP per capita',
       y = 'Life expectancy',
       color = 'Continent')

plot
```



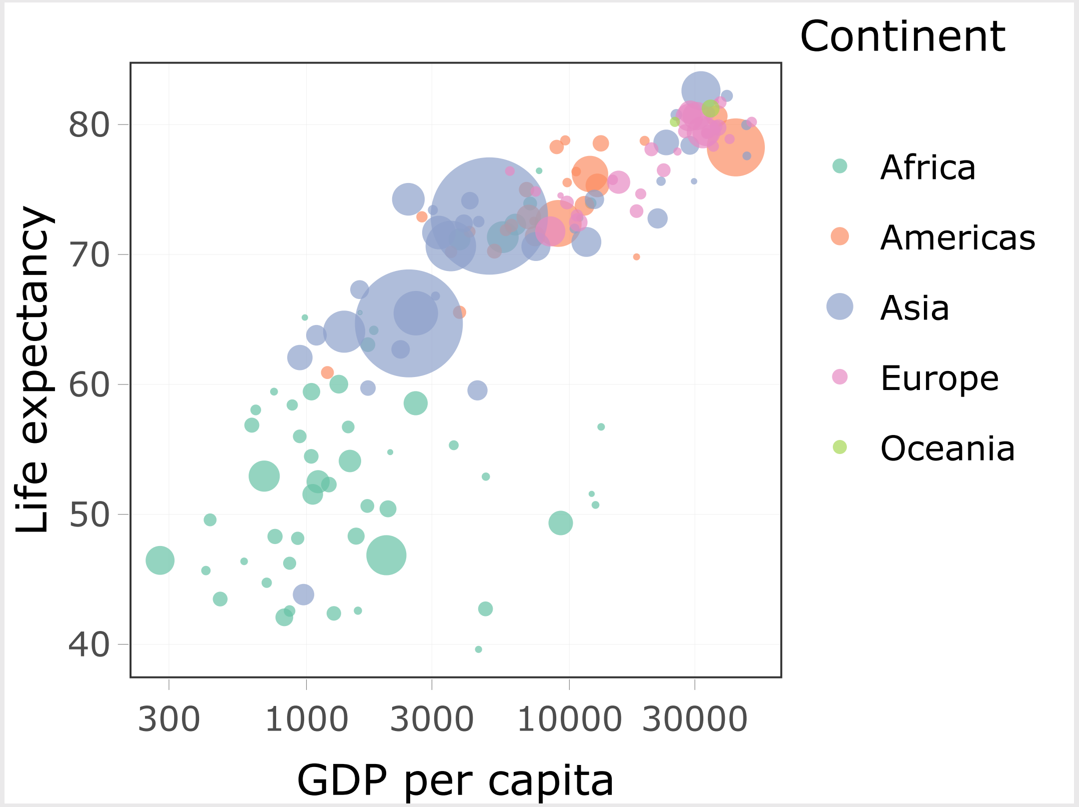
Turn any ggplot into an interactive chart with `ggplotly()`

```
ggplotly(plot)
```



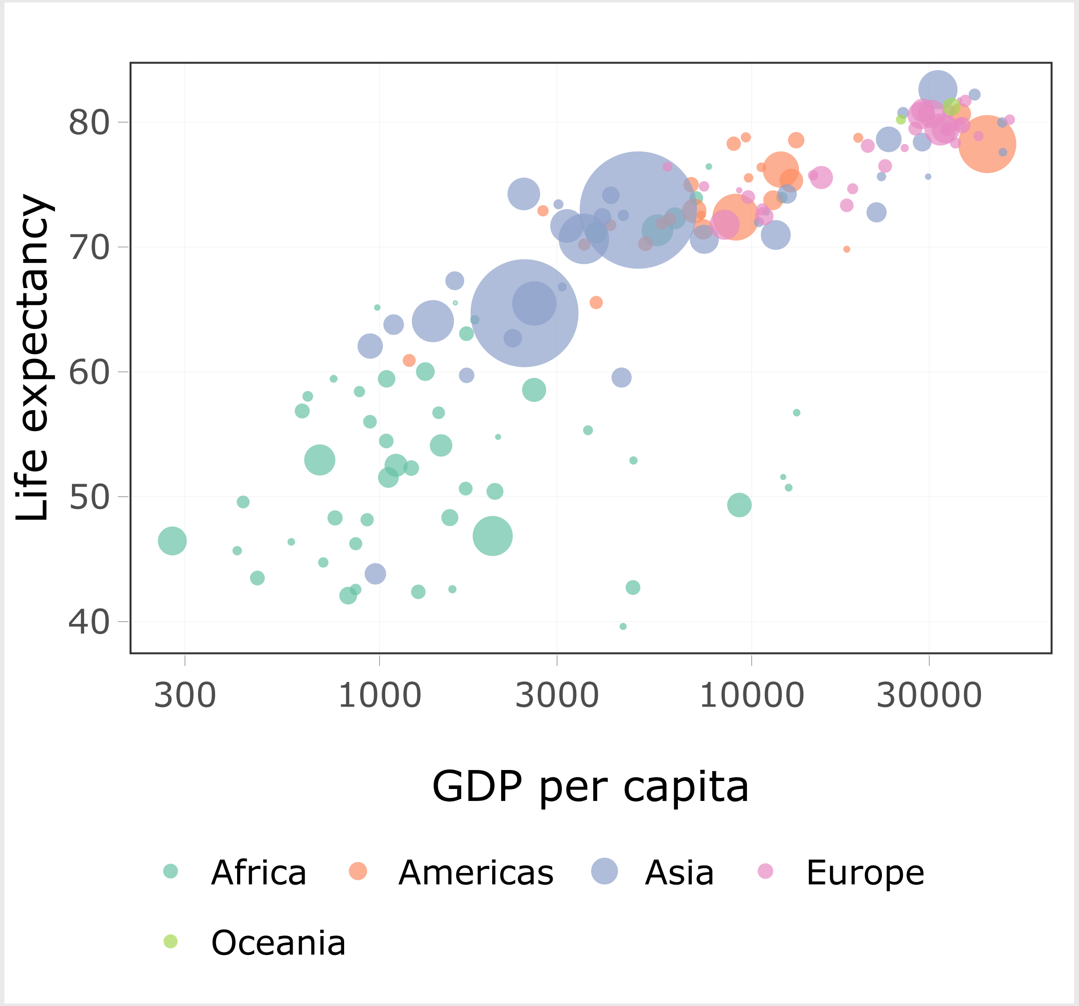
Modify the data shown with `tooltip` argument

```r
ggplotly(
  plot,
  tooltip = c("country", "pop")
)
```



Modify other features by piping on `plotly` functions

```
ggplotly(
  plot,
  tooltip = c("country", "pop")
) %>%
  layout(legend = list(
    orientation = "h", x = 0, y = -0.3))
```

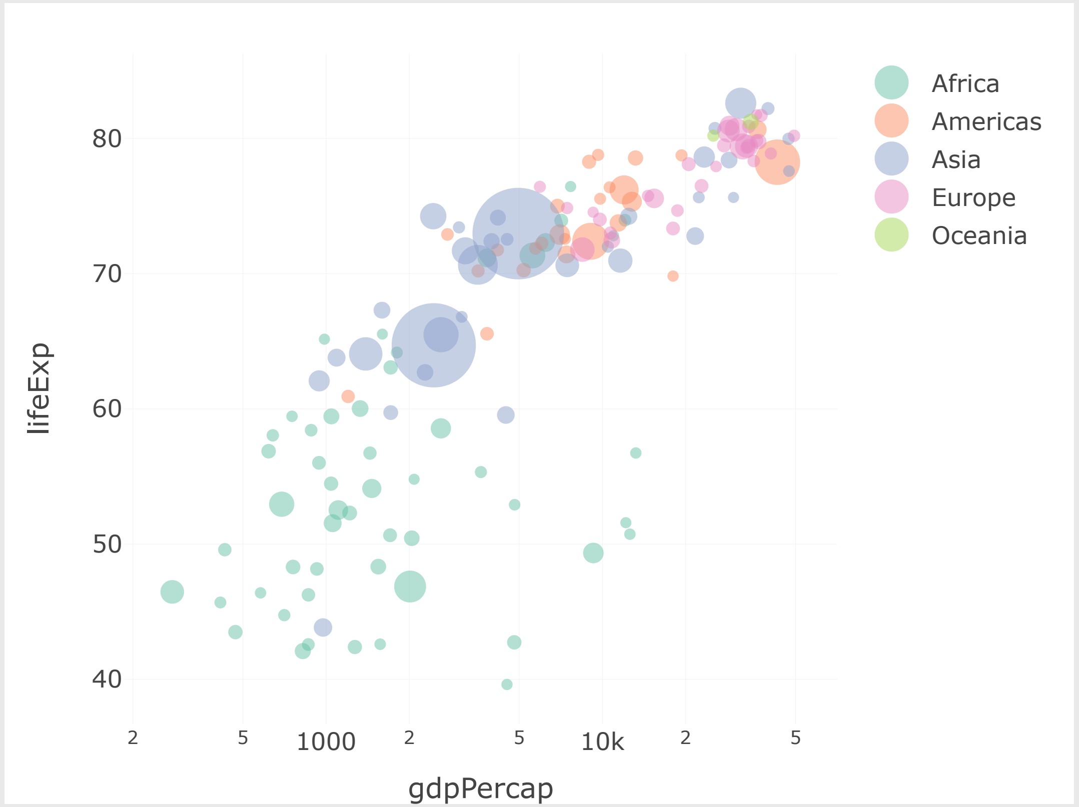


Reference guide: https://plotly.com/ggplot2/

Make interactive charts with `plot_ly()`

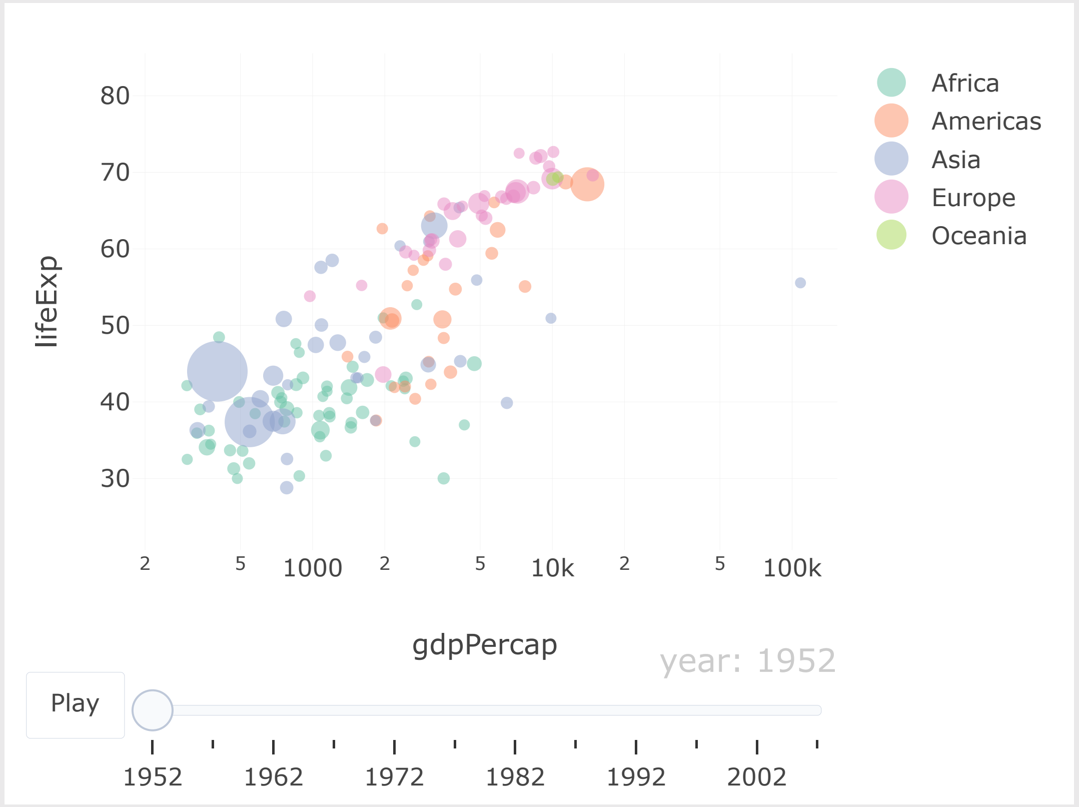
(More examples here: https://plotly.com/r/)

```r
plot_ly(
  data = gapminder %>% filter(year == 2007),
  type = 'scatter',
  x = ~gdpPercap,
  y = ~lifeExp,
  size = ~pop,
  color = ~continent,
  text = ~country,
  mode = "markers",
  sizes = c(10, 1000),
  marker = list(opacity = 0.5),
  hoverinfo = "text"
  ) %>%
  layout(xaxis = list(type = "log"))
```



Animation is relatively easy with `plot_ly()`

```r
plot_ly(
  data = gapminder,
  type = 'scatter',
  x = ~gdpPercap,
  y = ~lifeExp,
  size = ~pop,
  color = ~continent,
  text = ~country,
  frame = ~year,
  mode = "markers",
  sizes = c(10, 1000),
  marker = list(opacity = 0.5),
  hoverinfo = "text"
  ) %>%
  layout(xaxis = list(type = "log"))
```



Save as html page

```r
htmlwidgets::saveWidget(
  ggplotly(plot),
  file = here::here('figs', 'gapminder.html')
)
```

Insert using iframe

```r
htmltools::tags$iframe(
  src         = here::here('figs', 'gapminder.html'),
  width       = "100%",
  height      = "400",
  scrolling   = "no",
  seamless    = "seamless",
  frameBorder = "0"
)
```

One more option: https://g2r.opifex.org/index.html

Your Turn: Interactive Charts

10:00

1. Open your reflection from this past week
2. Take turns sharing your interactive chart
3. With a classmate, go back to a chart we made in a previous class and make it interactive using either `ggplotly()` or `plot_ly()`

When 10 minutes is up, we will share 3 examples.

Week 12: *Interactive Charts*

1. Interactive charts

2. Interactive tables

Intermission

3. Interactive maps

Make pretty static tables with `kable()`

```
library(knitr)

gapminder %>%
  kable()
```

country	continent	year	lifeExp	pop	gdpPercap
Afghanistan	Asia	1952	28.80100	8425333	779.4453
Afghanistan	Asia	1957	30.33200	9240934	820.8530
Afghanistan	Asia	1962	31.99700	10267083	853.1007
Afghanistan	Asia	1967	34.02000	11537966	836.1971
Afghanistan	Asia	1972	36.08800	13079460	739.9811
Afghanistan	Asia	1977	38.43800	14880372	786.1134
Afghanistan	Asia	1982	39.85400	12881816	978.0114
Afghanistan	Asia	1987	40.82200	13867957	852.3959
Afghanistan	Asia	1992	41.67400	16317921	649.3414
Afghanistan	Asia	1997	41.76300	22227415	635.3414
Afghanistan	Asia	2002	42.12900	25268405	726.7341

Behind the scenes:

`kable()` generates the code to make a pretty table

```r
gapminder %>%
  kable(format = "pipe")
```

```
|country |continent | year| lifeExp| pop| gdpPercap| |:-------------------------|:---------
|----:|--------:|----------:|-----------:| |Afghanistan |Asia | 1952| 28.80100| 8425333|
779.4453| |Afghanistan |Asia | 1957| 30.33200| 9240934| 820.8530| |Afghanistan
|Asia | 1962| 31.99700| 10267083| 853.1007| |Afghanistan |Asia | 1967| 34.02000|
11537966| 836.1971| |Afghanistan |Asia | 1972| 36.08800| 13079460| 739.9811|
|Afghanistan |Asia | 1977| 38.43800| 14880372| 786.1134| |Afghanistan |Asia | 1982|
39.85400| 12881816| 978.0114| |Afghanistan |Asia | 1987| 40.82200| 13867957|
852.3959| |Afghanistan |Asia | 1992| 41.67400| 16317921| 649.3414| |Afghanistan
|Asia | 1997| 41.76300| 22227415| 635.3414| |Afghanistan |Asia | 2002| 42.12900|
25268405| 726.7341| |Afghanistan |Asia | 2007| 43.82800| 31889923| 974.5803|
|Albania |Europe | 1952| 55.23000| 1282697| 1601.0561| |Albania |Europe | 1957|
```

Behind the scenes:

`kable()` generates the code to make a pretty table

```
gapminder %>%
  kable(format = "html")
```

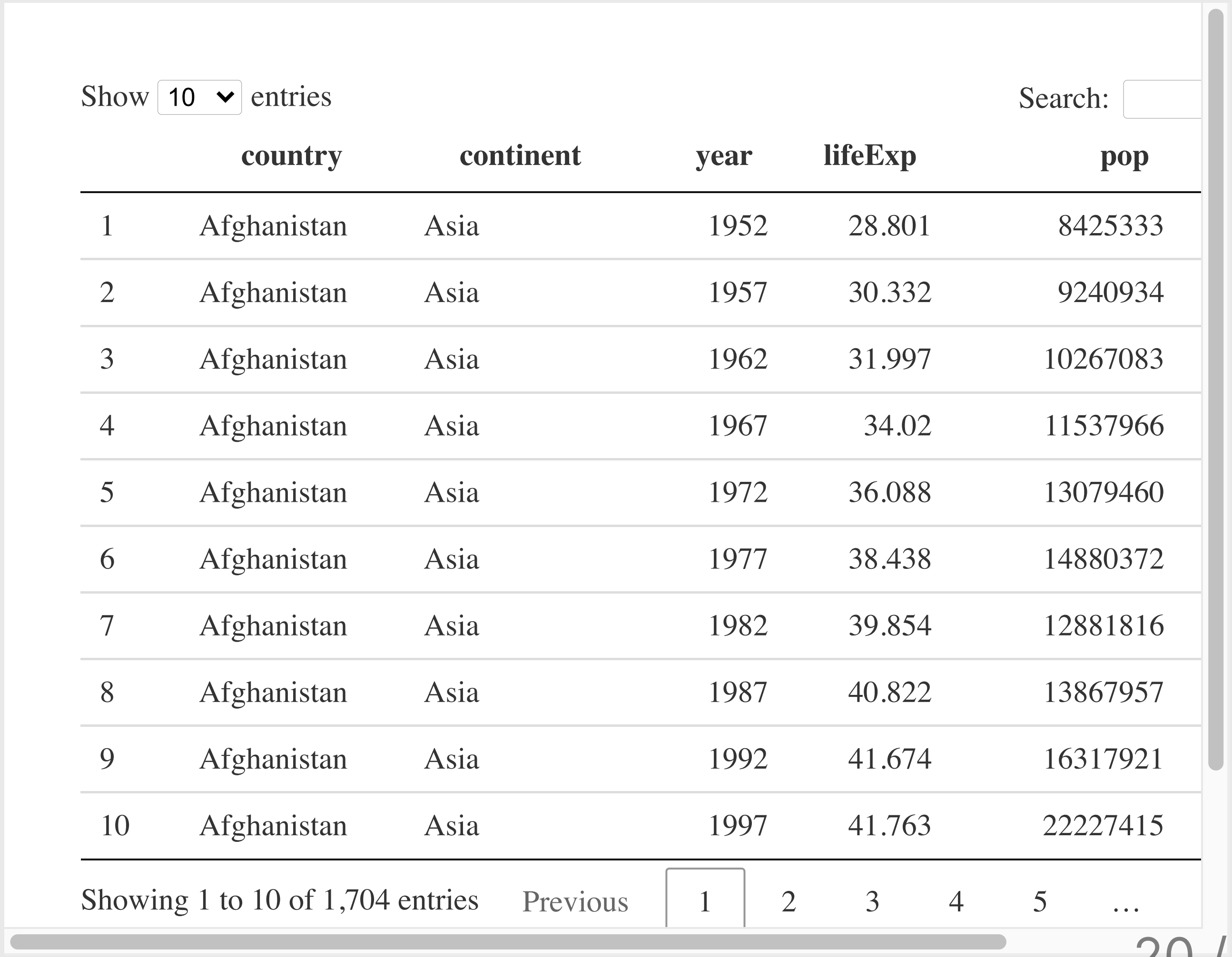
```
#> <table>
#>  <thead>
#>   <tr>
#>    <th style="text-align:left;"> country </th>
#>    <th style="text-align:left;"> continent </th>
#>    <th style="text-align:right;"> year </th>
#>    <th style="text-align:right;"> lifeExp </th>
#>    <th style="text-align:right;"> pop </th>
#>    <th style="text-align:right;"> gdpPercap </th>
#>   </tr>
#>  </thead>
#> <tbody>
#>   <tr>
#>    <td style="text-align:left;"> Afghanistan </td>
#>    <td style="text-align:left;"> Asia </td>
```

Make *interactive* tables with:

`DT::datatable()`

Make *interactive* tables with `datatable()`

```r
library(DT)

gapminder %>%
  datatable()
```



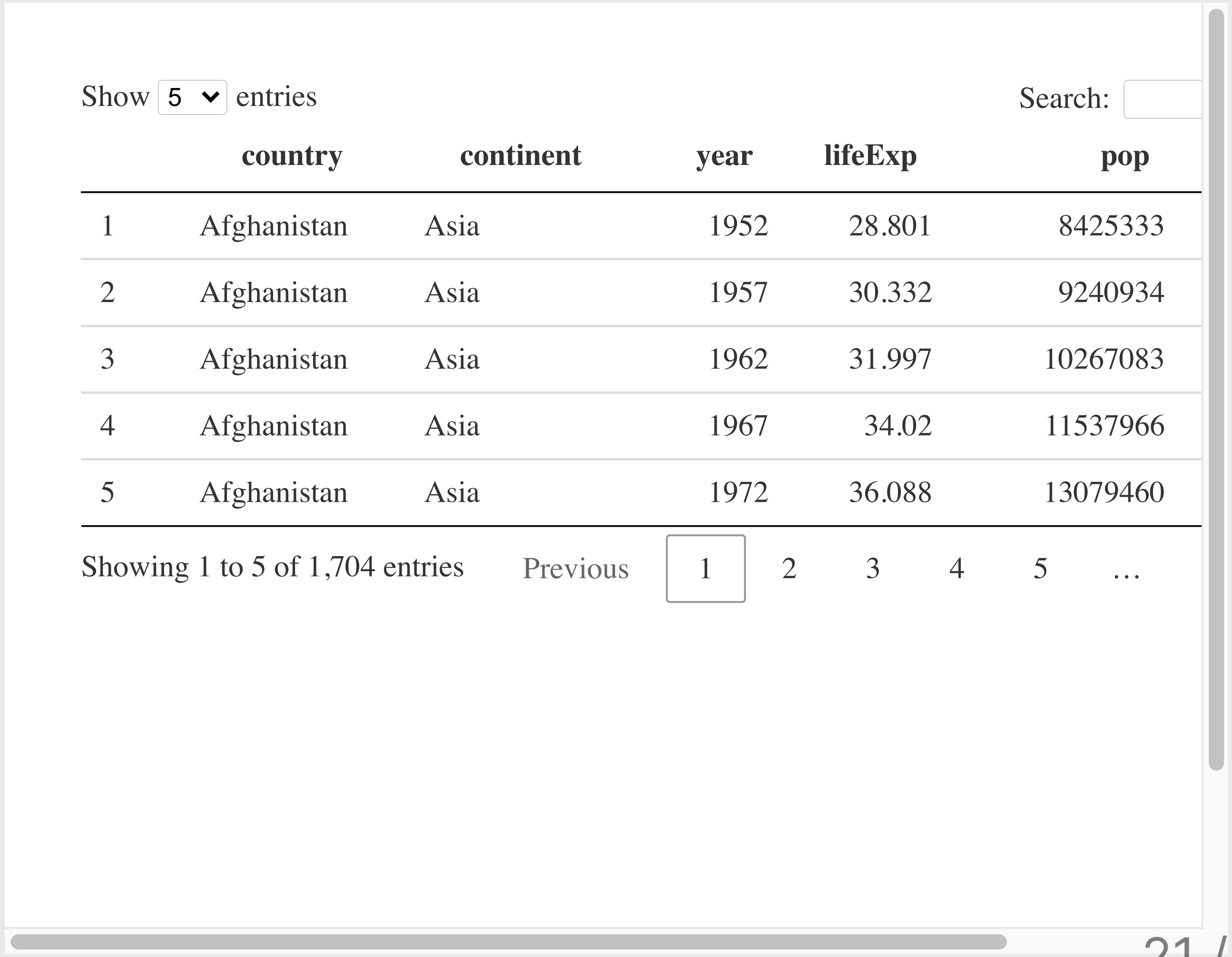
Show 10 entries Search:

	country	continent	year	lifeExp	pop
1	Afghanistan	Asia	1952	28.801	8425333
2	Afghanistan	Asia	1957	30.332	9240934
3	Afghanistan	Asia	1962	31.997	10267083
4	Afghanistan	Asia	1967	34.02	11537966
5	Afghanistan	Asia	1972	36.088	13079460
6	Afghanistan	Asia	1977	38.438	14880372
7	Afghanistan	Asia	1982	39.854	12881816
8	Afghanistan	Asia	1987	40.822	13867957
9	Afghanistan	Asia	1992	41.674	16317921
10	Afghanistan	Asia	1997	41.763	22227415

Showing 1 to 10 of 1,704 entries Previous 1 2 3 4 5 …

Make *interactive* tables with `datatable()`

```
gapminder %>%
  datatable(
    options = list(
      pageLength = 5,
      lengthMenu = c(5, 10, 15, 20
  )
```



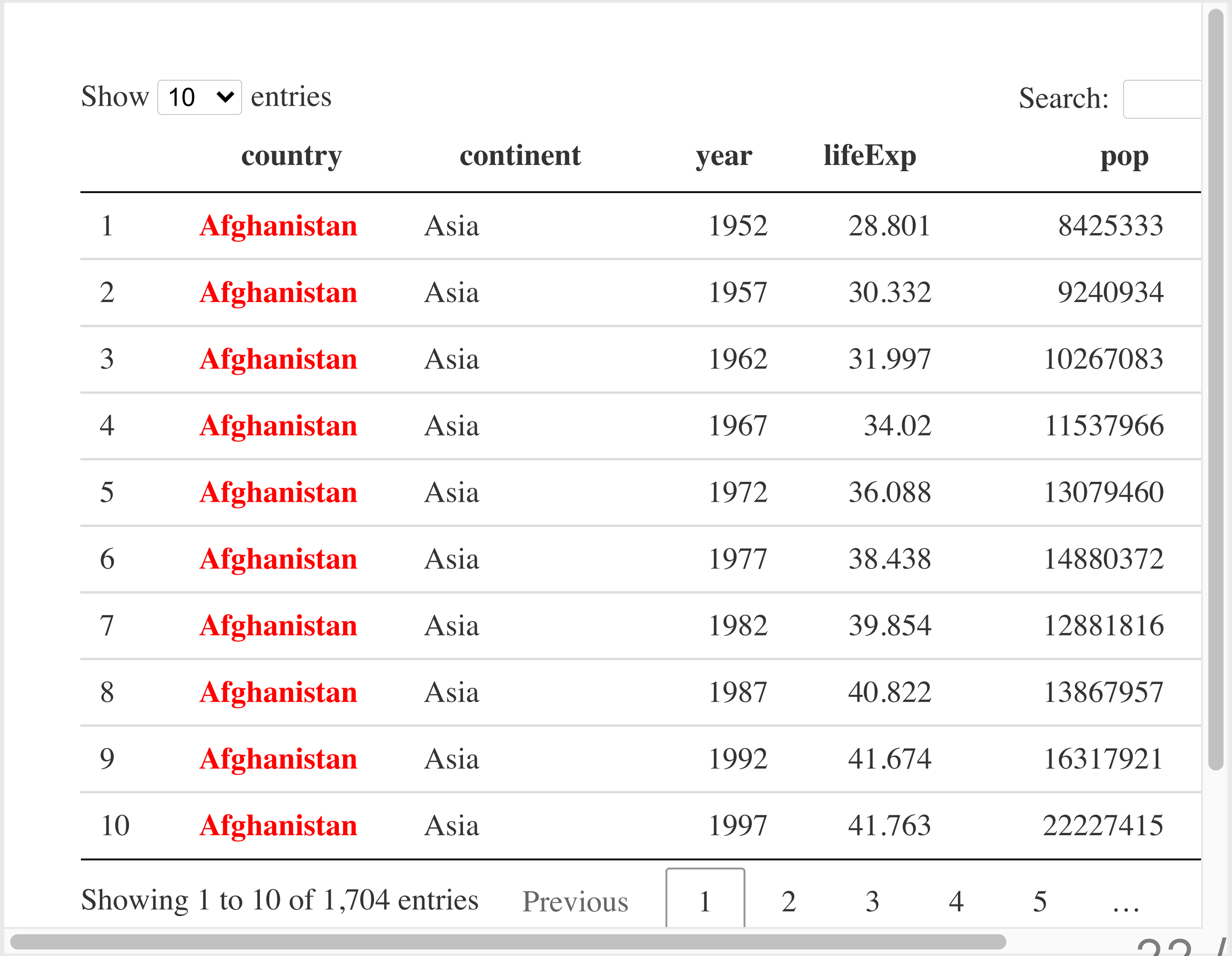
Show 5 entries Search:

	country	continent	year	lifeExp	pop
1	Afghanistan	Asia	1952	28.801	8425333
2	Afghanistan	Asia	1957	30.332	9240934
3	Afghanistan	Asia	1962	31.997	10267083
4	Afghanistan	Asia	1967	34.02	11537966
5	Afghanistan	Asia	1972	36.088	13079460

Showing 1 to 5 of 1,704 entries Previous 1 2 3 4 5 …

Modify features by piping on functions

```r
gapminder %>%
  datatable() %>%
  formatCurrency('gdpPercap') %>%
  formatStyle(
    'country',
    color = 'red',
    backgroundColor = 'black',
    fontWeight = 'bold')
```



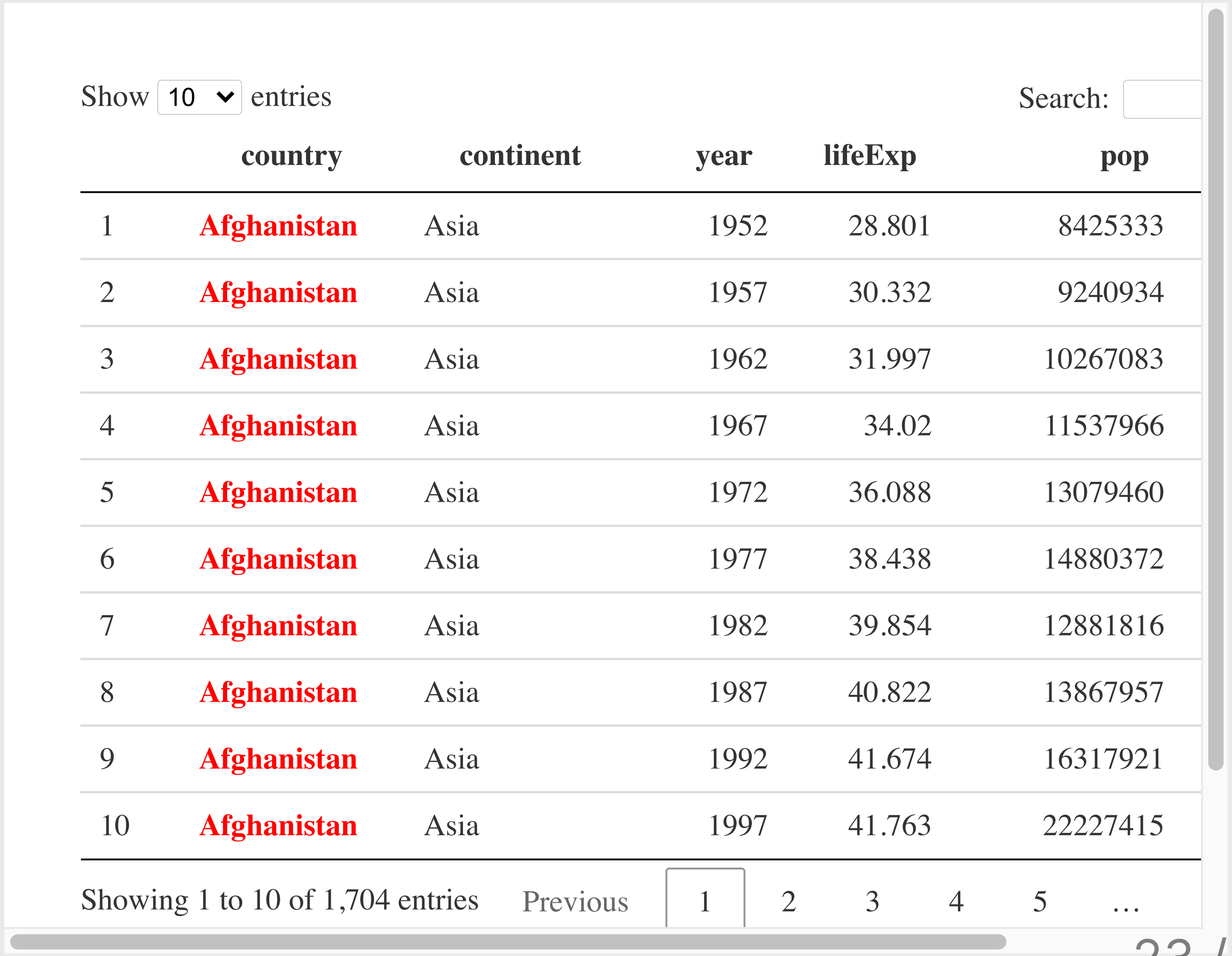
Show 10 entries   Search:

	country	continent	year	lifeExp	pop
1	**Afghanistan**	Asia	1952	28.801	8425333
2	**Afghanistan**	Asia	1957	30.332	9240934
3	**Afghanistan**	Asia	1962	31.997	10267083
4	**Afghanistan**	Asia	1967	34.02	11537966
5	**Afghanistan**	Asia	1972	36.088	13079460
6	**Afghanistan**	Asia	1977	38.438	14880372
7	**Afghanistan**	Asia	1982	39.854	12881816
8	**Afghanistan**	Asia	1987	40.822	13867957
9	**Afghanistan**	Asia	1992	41.674	16317921
10	**Afghanistan**	Asia	1997	41.763	22227415

Showing 1 to 10 of 1,704 entries   Previous 1 2 3 4 5 …

Modify features by piping on functions

```
gapminder %>%
  datatable() %>%
  formatCurrency('gdpPercap') %>%
  formatStyle(
    'country',
    color = 'red',
    backgroundColor = 'black',
    fontWeight = 'bold') %>%
  formatStyle(
    'lifeExp',
    background = styleColorBar(
      gapminder$lifeExp, 'dodgerbl
    backgroundSize = '100% 90%',
    backgroundRepeat = 'no-repeat'
    backgroundPosition = 'center')
```


Show 10 entries Search:

	country	continent	year	lifeExp	pop
1	**Afghanistan**	Asia	1952	28.801	8425333
2	**Afghanistan**	Asia	1957	30.332	9240934
3	**Afghanistan**	Asia	1962	31.997	10267083
4	**Afghanistan**	Asia	1967	34.02	11537966
5	**Afghanistan**	Asia	1972	36.088	13079460
6	**Afghanistan**	Asia	1977	38.438	14880372
7	**Afghanistan**	Asia	1982	39.854	12881816
8	**Afghanistan**	Asia	1987	40.822	13867957
9	**Afghanistan**	Asia	1992	41.674	16317921
10	**Afghanistan**	Asia	1997	41.763	22227415

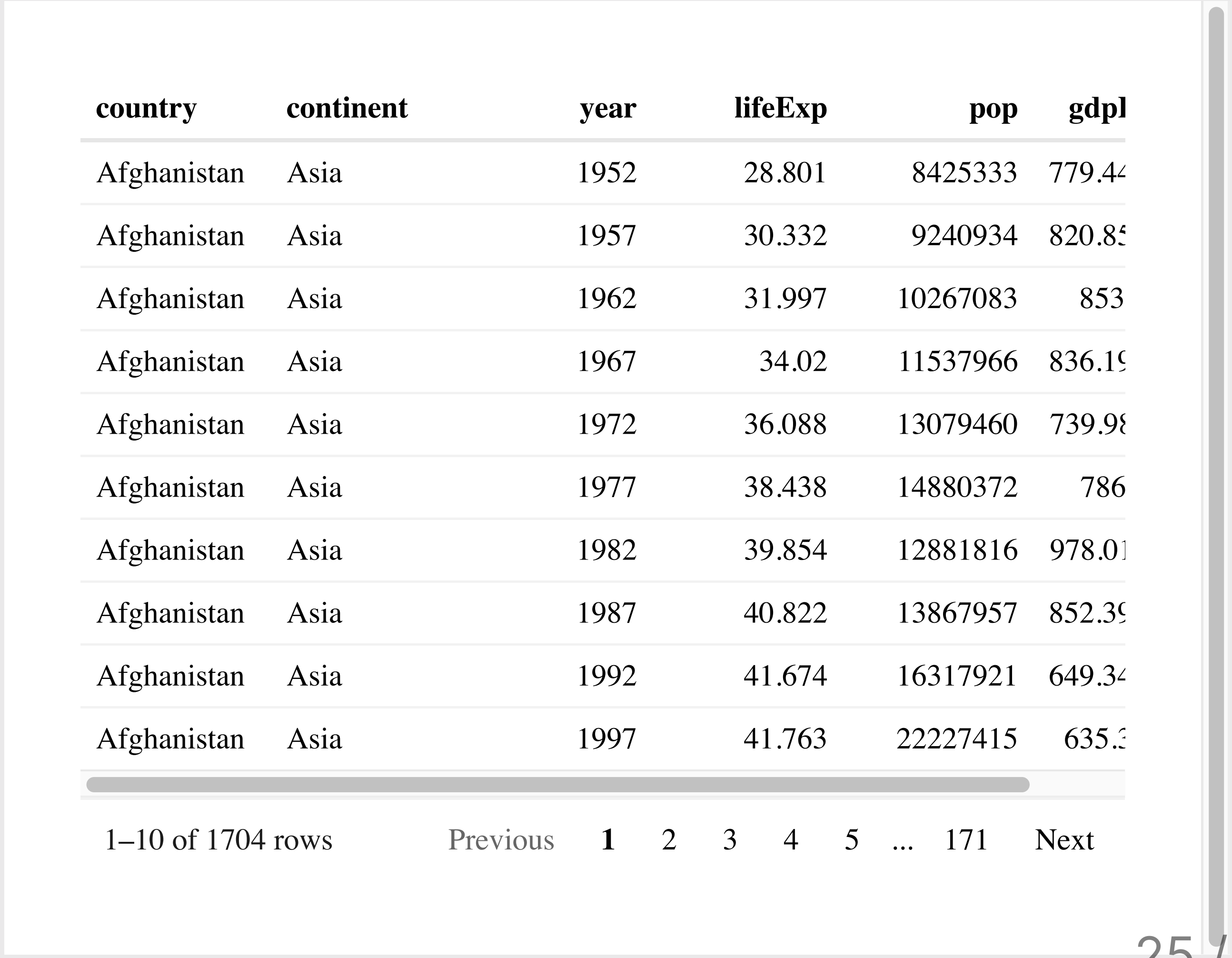
Showing 1 to 10 of 1,704 entries Previous 1 2 3 4 5 …

Make *interactive* tables with:

`reactable::reactable()`

Make *interactive* tables with `reactable()`

```r
library(reactable)

gapminder %>%
  reactable()
```

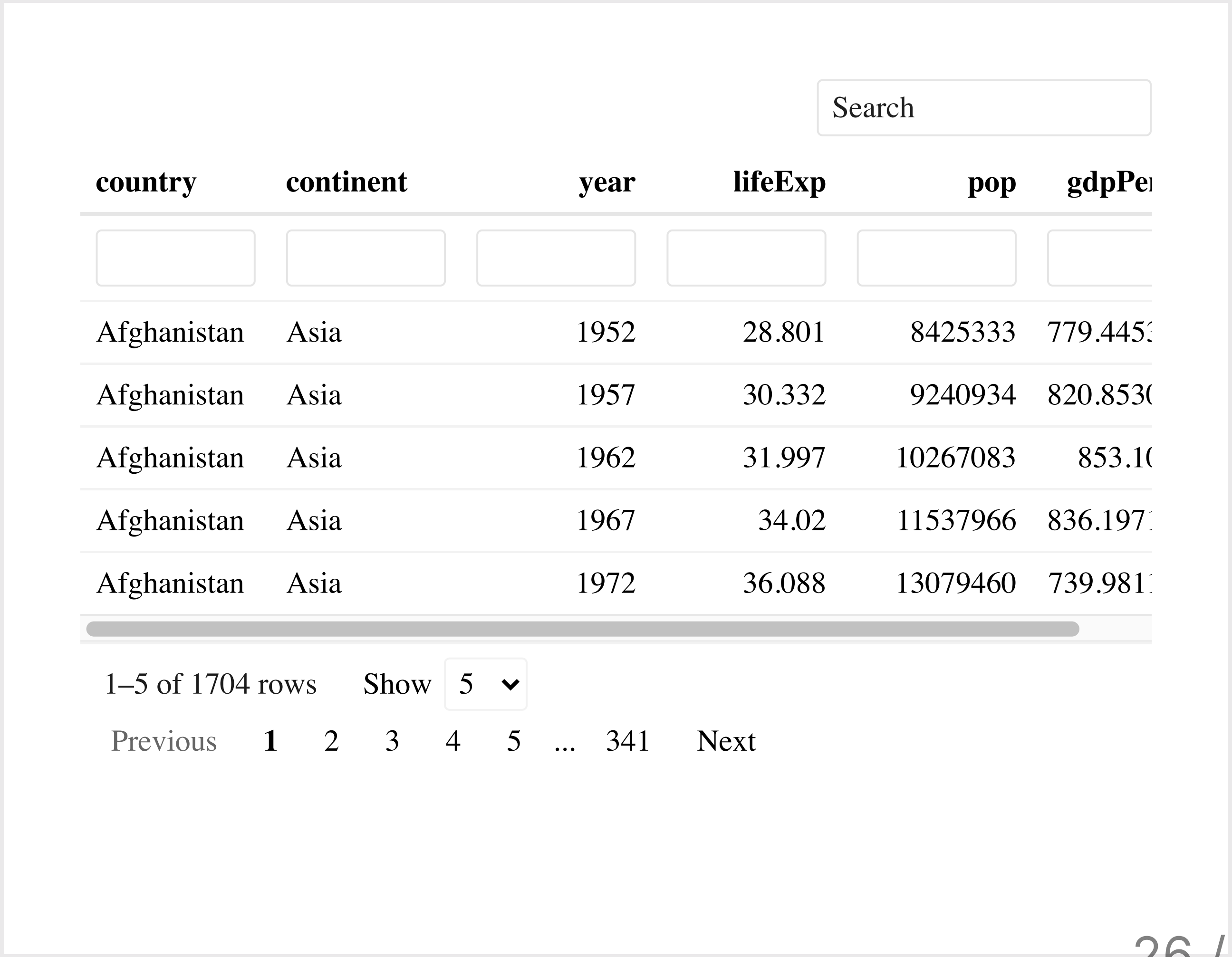


country	continent	year	lifeExp	pop	gdpl
Afghanistan	Asia	1952	28.801	8425333	779.4
Afghanistan	Asia	1957	30.332	9240934	820.8
Afghanistan	Asia	1962	31.997	10267083	853
Afghanistan	Asia	1967	34.02	11537966	836.1
Afghanistan	Asia	1972	36.088	13079460	739.9
Afghanistan	Asia	1977	38.438	14880372	786
Afghanistan	Asia	1982	39.854	12881816	978.0
Afghanistan	Asia	1987	40.822	13867957	852.3
Afghanistan	Asia	1992	41.674	16317921	649.3
Afghanistan	Asia	1997	41.763	22227415	635.

1–10 of 1704 rows   Previous **1** 2 3 4 5 ... 171 Next

`reactable()` has some nice options!

```r
library(reactable)

gapminder %>%
  reactable(
    searchable = TRUE,
    highlight = TRUE,
    filterable = TRUE,
    defaultPageSize = 5,
    showPageSizeOptions = TRUE,
    pageSizeOptions = c(5, 10, 15)
  )
```



Search

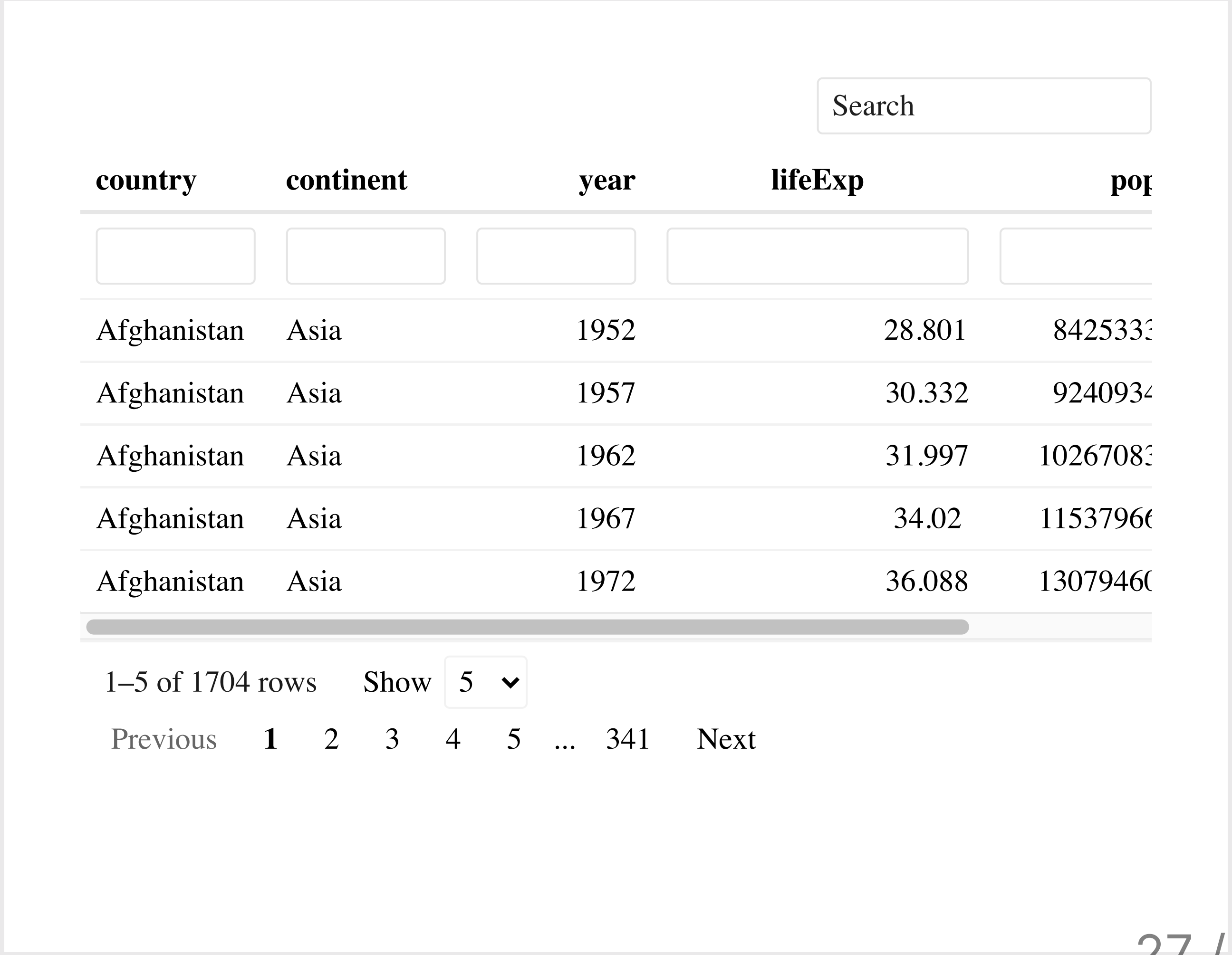
country	continent	year	lifeExp	pop	gdpPe
Afghanistan	Asia	1952	28.801	8425333	779.445
Afghanistan	Asia	1957	30.332	9240934	820.853
Afghanistan	Asia	1962	31.997	10267083	853.1
Afghanistan	Asia	1967	34.02	11537966	836.197
Afghanistan	Asia	1972	36.088	13079460	739.981

1–5 of 1704 rows Show 5

Previous **1** 2 3 4 5 ... 341 Next

Add more features with `reactablefmtr` library

```r
library(reactable)
library(reactablefmtr)

gapminder %>%
  reactable(
    searchable = TRUE,
    highlight = TRUE,
    filterable = TRUE,
    defaultPageSize = 5,
    showPageSizeOptions = TRUE,
    pageSizeOptions = c(5, 10, 15)
    columns = list(
      lifeExp = colDef(cell = data_
        gapminder,
        colors = c("#d7191c", "#ff
      align = "center")) ## align 
  )
```



Search

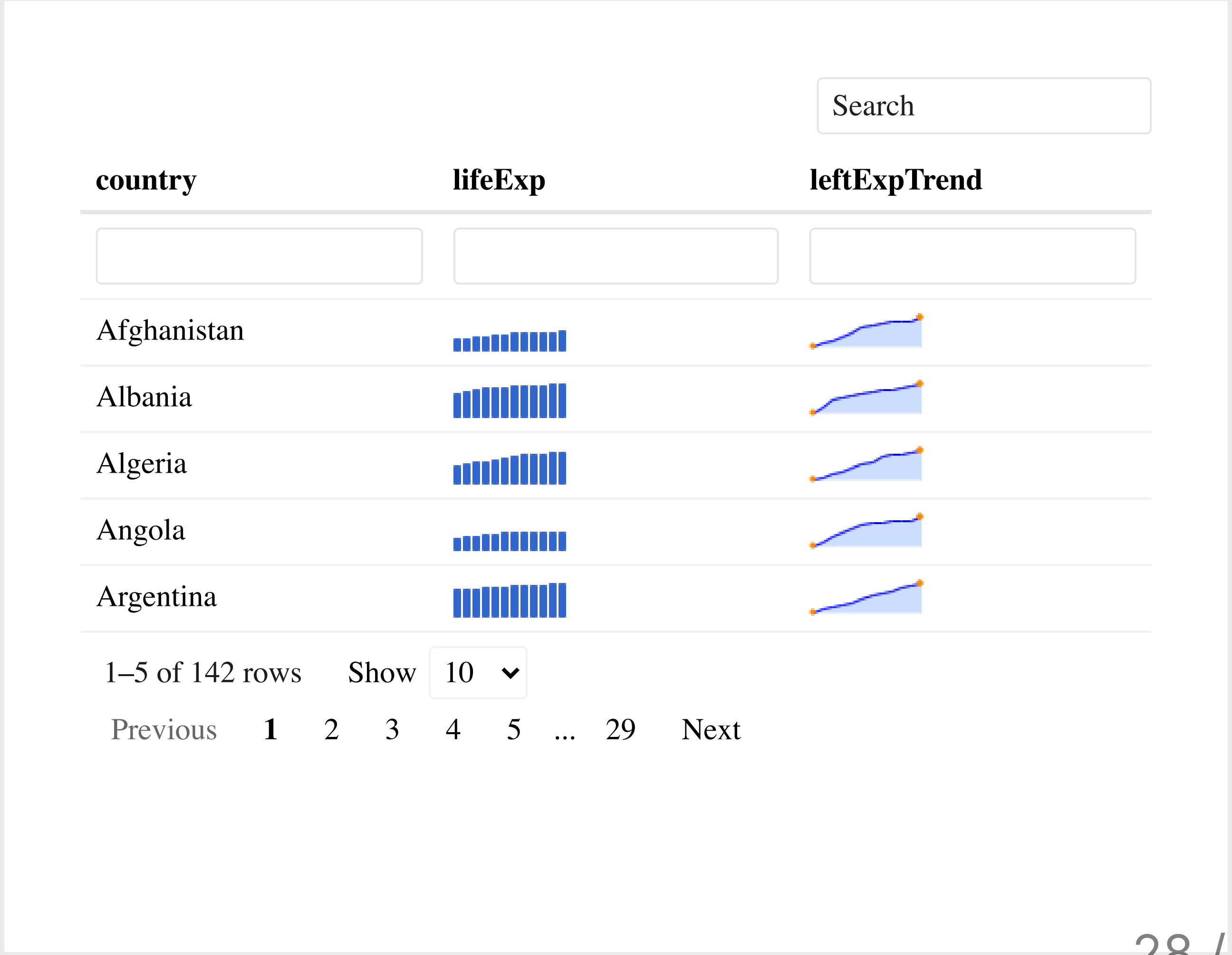
country	continent	year	lifeExp	pop
Afghanistan	Asia	1952	28.801	8425333
Afghanistan	Asia	1957	30.332	9240934
Afghanistan	Asia	1962	31.997	10267083
Afghanistan	Asia	1967	34.02	11537966
Afghanistan	Asia	1972	36.088	13079460

1–5 of 1704 rows Show 5

Previous **1** 2 3 4 5 ... 341 Next

Add more features with `sparkline` library (example)

```r
library(reactable)
library(sparkline)

gapminder_summary <- gapminder %>%
  group_by(country) %>%
  summarise(lifeExp = list(lifeExp)) %>%
  mutate(leftExpTrend = NA)

gapminder_reactable_sparkline <- gapminder_
  reactable(
    searchable = TRUE,
    highlight = TRUE,
    filterable = TRUE,
    defaultPageSize = 5,
    showPageSizeOptions = TRUE,
    columns = list(
      lifeExp = colDef(
        cell = function(values) {
          sparkline(
            values, type = "bar", chartRang
            chartRangeMax = max(gapminder$l
        }),
      leftExpTrend = colDef(
        cell = function(value, index) {
          sparkline(gapminder_summary$lifeE
        })
  ))
```



Search

country	lifeExp	leftExpTrend
Afghanistan		
Albania		
Algeria		
Angola		
Argentina		

1–5 of 142 rows Show 10

Previous 1 2 3 4 5 ... 29 Next

References:

- https://rstudio.github.io/DT/

- https://glin.github.io/reactable/

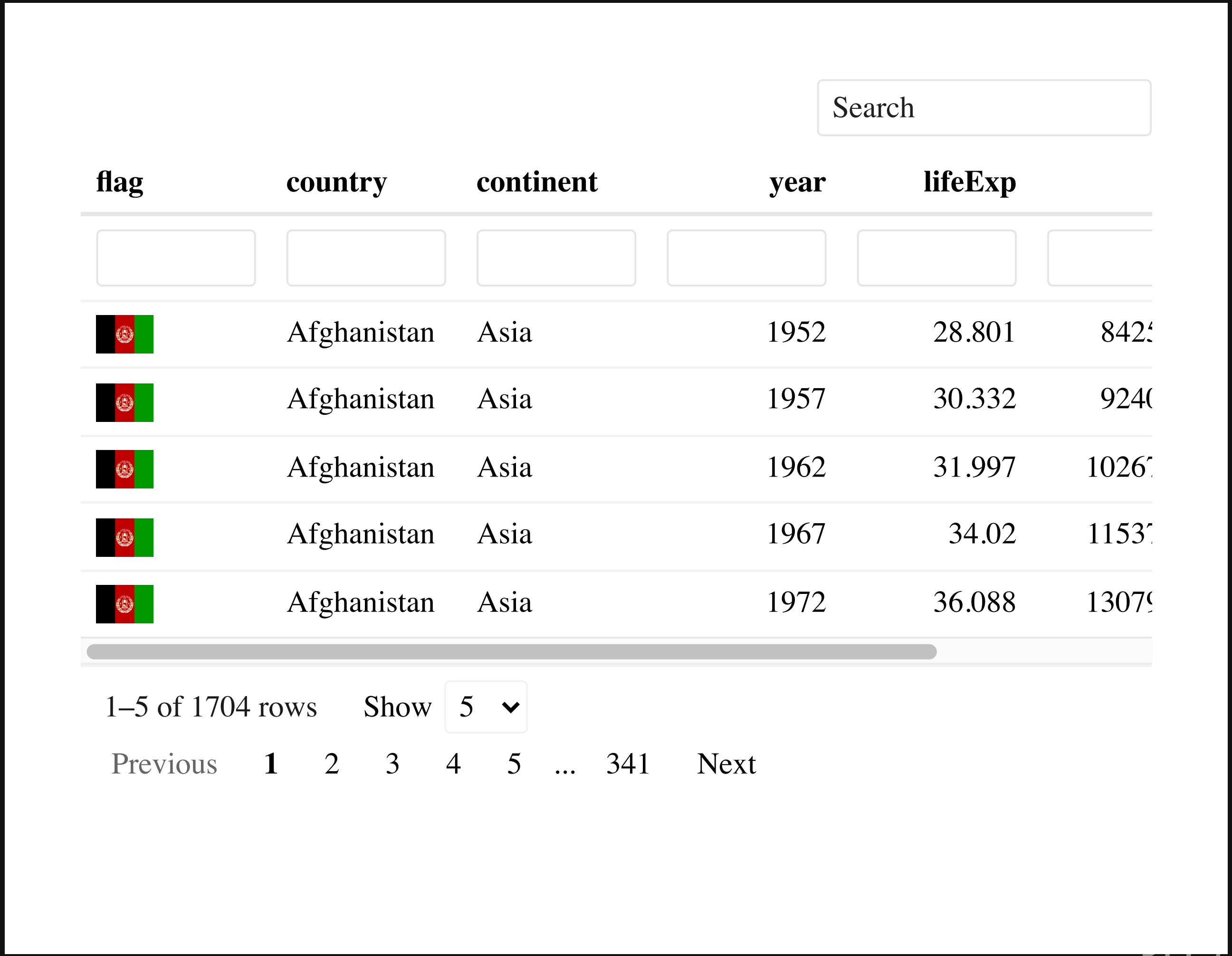
- https://kcuilla.github.io/reactablefmtr/

Your Turn: Interactive Tables

20:00

Use `reactable()` to make the following interactive table

Read [this example](#) on how to embed images in table cells, then use the `gapminder_flags` data frame to make the interactive table.



Search

flag	country	continent	year	lifeExp	
🇦🇫	Afghanistan	Asia	1952	28.801	842
🇦🇫	Afghanistan	Asia	1957	30.332	924
🇦🇫	Afghanistan	Asia	1962	31.997	1026
🇦🇫	Afghanistan	Asia	1967	34.02	1153
🇦🇫	Afghanistan	Asia	1972	36.088	1307

1–5 of 1704 rows Show 5

Previous **1** 2 3 4 5 ... 341 Next

Intermission

05:00

Week 12: *Interactive Charts*

1. Interactive charts

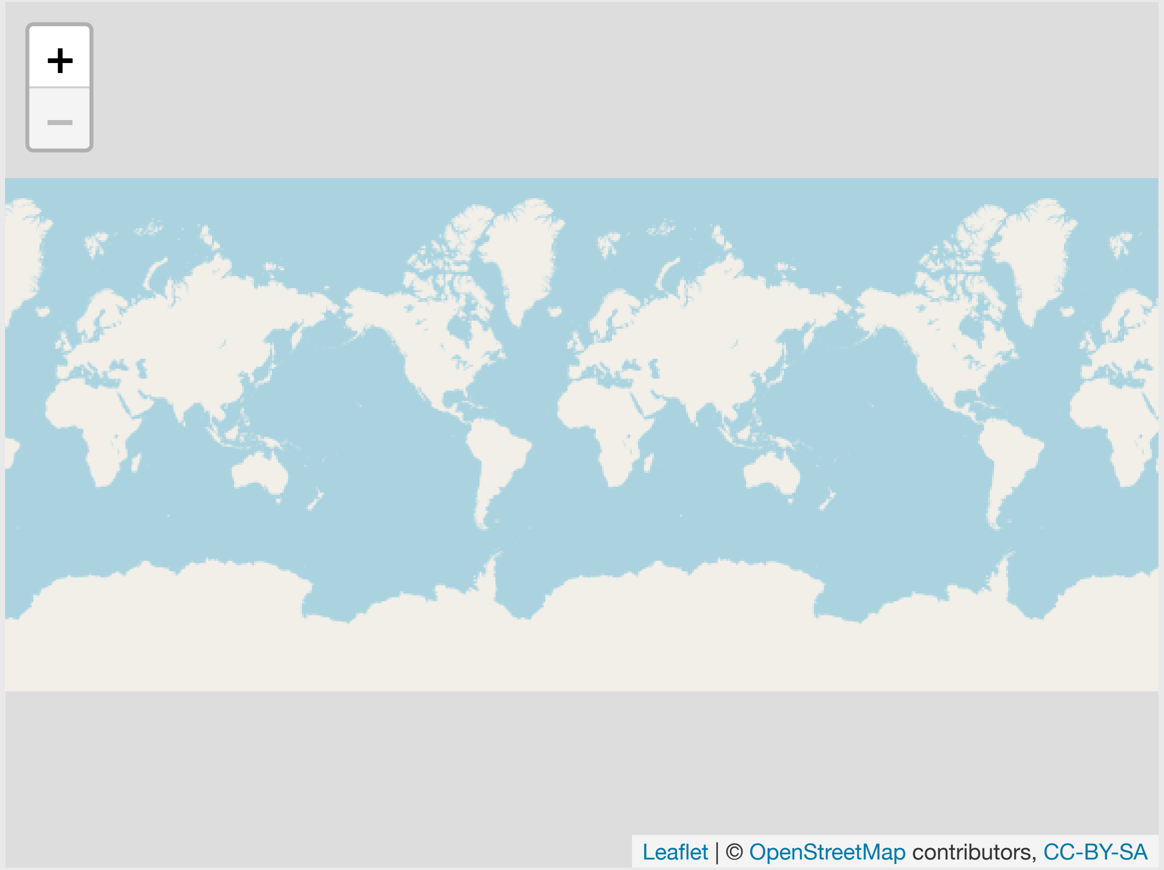
2. Interactive tables

Intermission

3. Interactive maps

Make interactive maps with `leaflet`

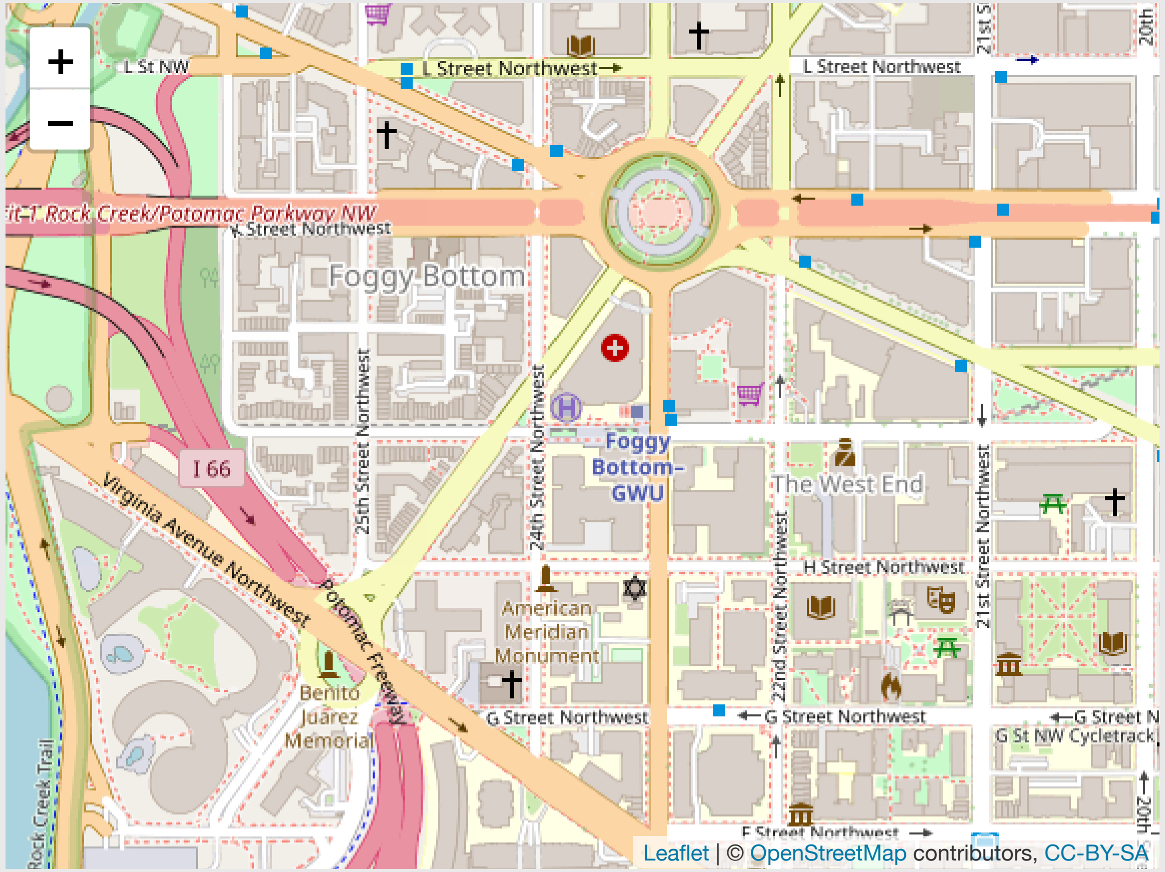
```r
library(leaflet)

# Default tiles: OpenStreetMap
leaflet() %>%
  addTiles()
```



Use `setView()` to set a start location

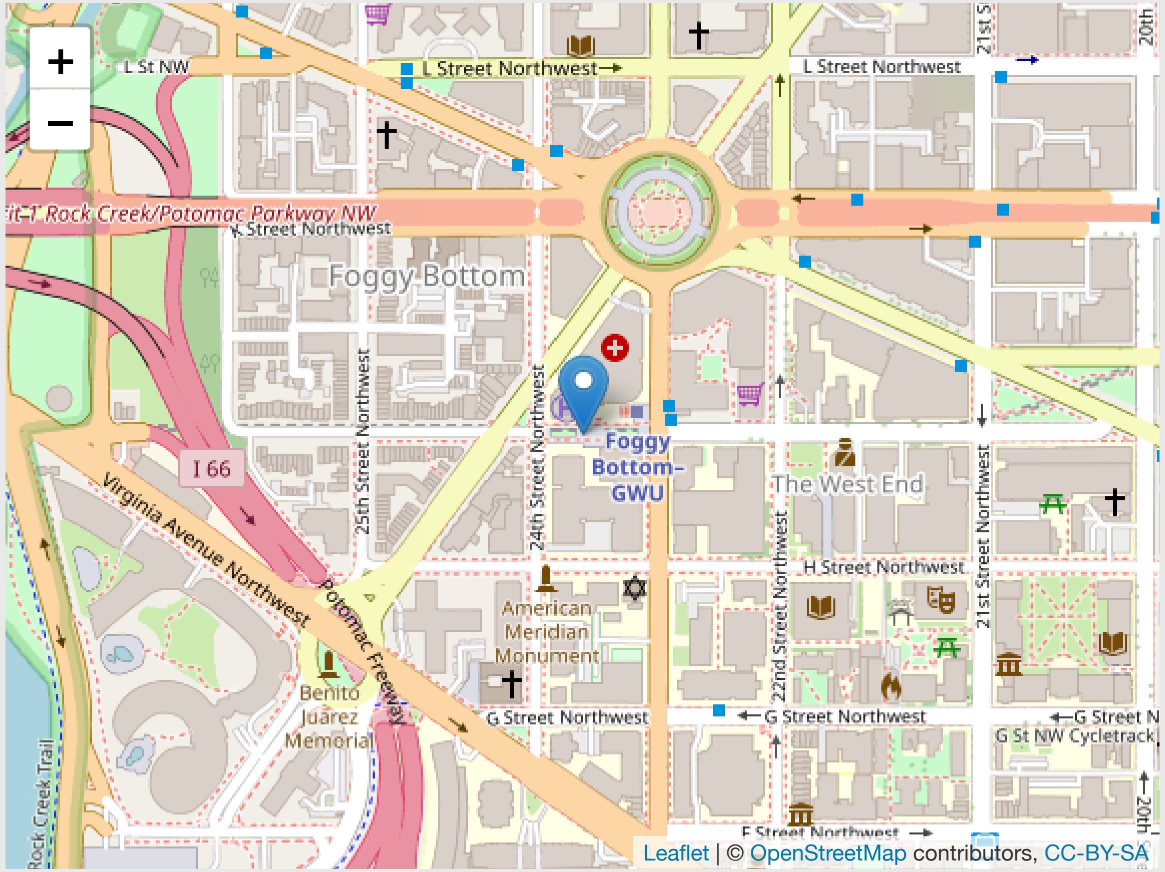
```r
leaflet() %>%
  addTiles() %>%
  setView(
    lat = 38.900671142379586,
    lng = -77.05094820047492,
    zoom = 16)
```

Coordinates from Google Maps



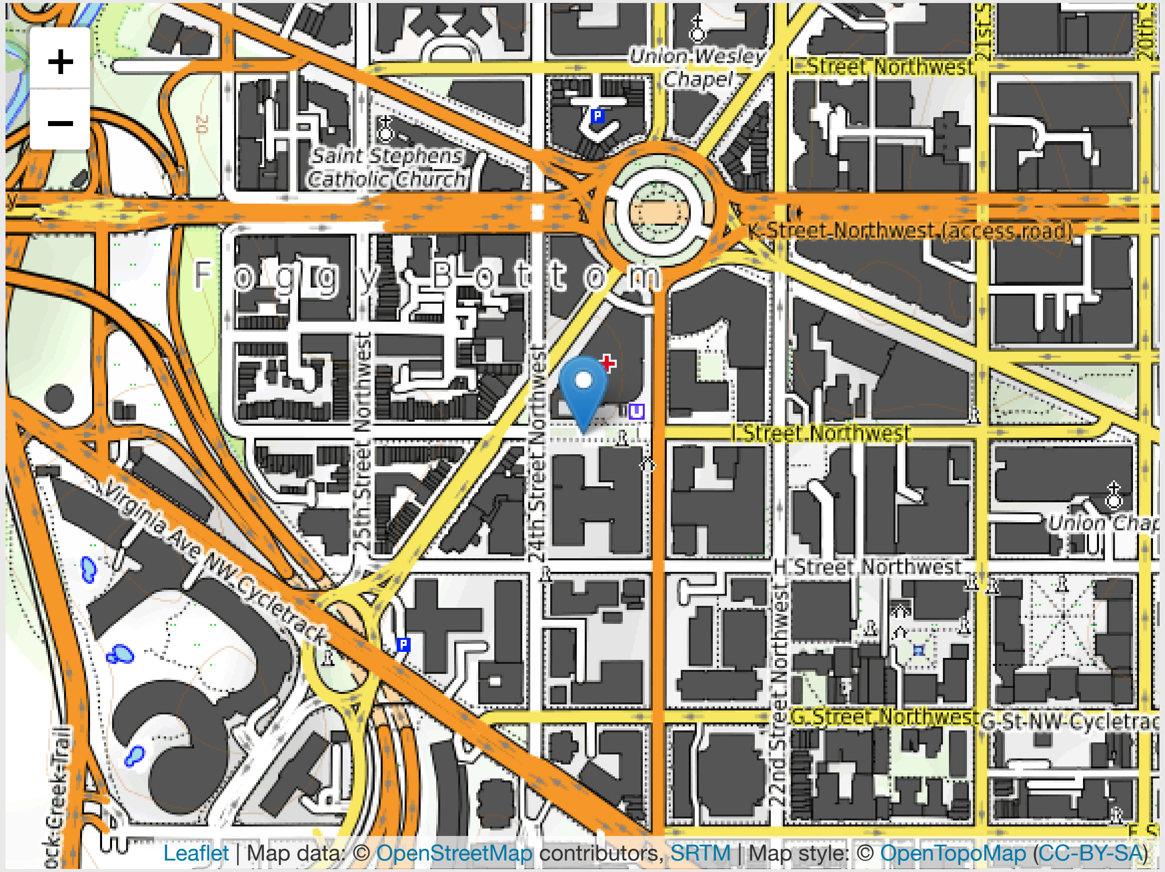
Use `addMarkers()` to add markers

```r
leaflet() %>%
  addTiles() %>%
  setView(
    lat = 38.90067114237958б,
    lng = -77.05094820047492,
    zoom = 16) %>%
  addMarkers(
    lat = 38.900671142379586,
    lng = -77.05094820047492,
    popup = "GWU!!!")
```



Change the tiles with `addProviderTiles()`

```
leaflet() %>%
  # addTiles() %>%
  addProviderTiles(providers$OpenTopoMap) %>
  setView(
    lat = 38.900671142379586,
    lng = -77.05094820047492,
    zoom = 16) %>%
  addMarkers(
    lat = 38.900671142379586,
    lng = -77.05094820047492,
    popup = "GWU!!!")
```



Go here to get other tiles

Overlaying data with leaflet

- Points

- Choropleth maps

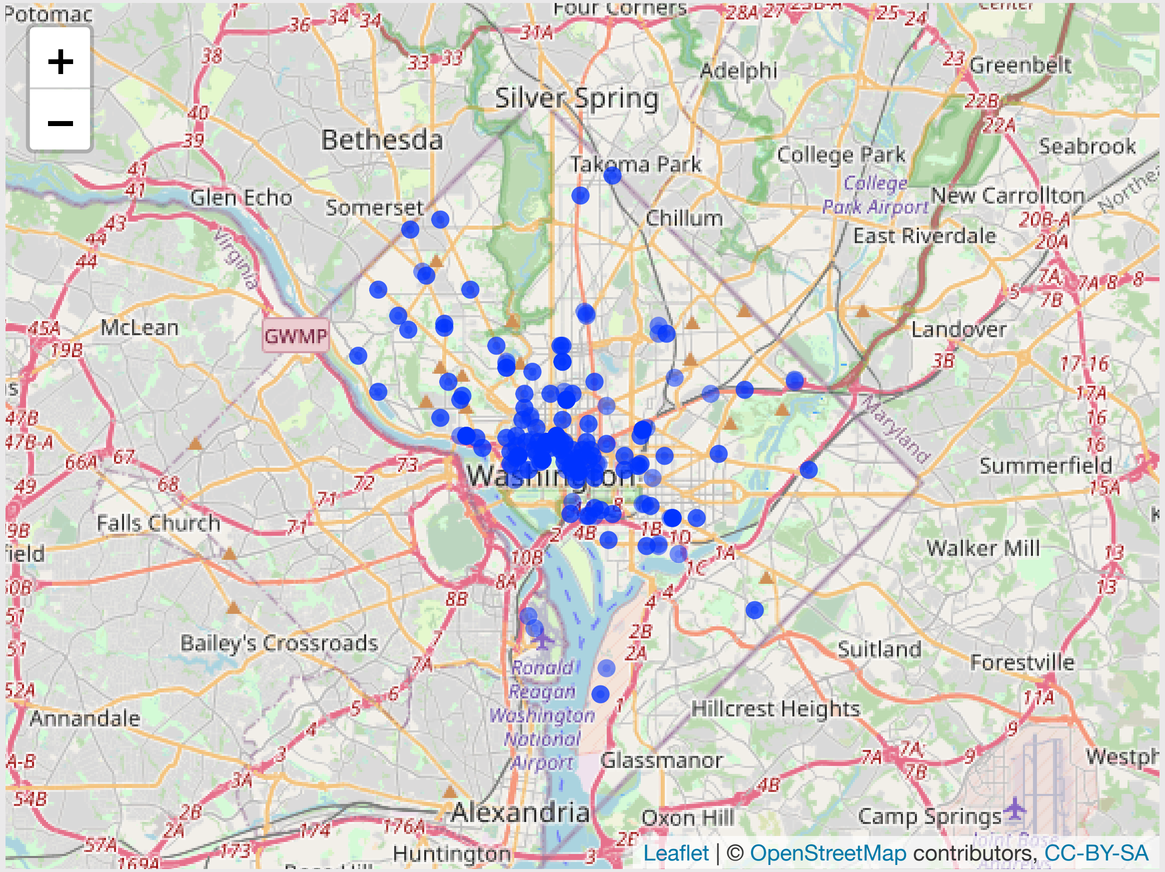
Points example: coffee data

```r
coffee_shops <- read_csv(here::here("data", "us_coffee_shops.csv"))

# Let's just look at MD
dc_coffee_shops <- coffee_shops %>%
  filter(state == "District of Columbia")

head(dc_coffee_shops)
```

```
#> # A tibble: 6 × 8
#>   name              lat  long unique_id city       state_abb zip   state
#>   <chr>           <dbl> <dbl>     <dbl> <chr>      <chr>     <chr> <chr>
#> 1 Baskin Robbins   38.9 -77.1   3303629 Washington DC        20008 District of Columbia
#> 2 Baskin Robbins   38.9 -76.9   3303741 Washington DC        20019 District of Columbia
#> 3 Baskin Robbins   38.9 -77.0   3303173 Washington DC        20002 District of Columbia
#> 4 Baskin Robbins   38.9 -77.0   3303939 Washington DC        20003 District of Columbia
#> 5 Baskin Robbins   38.9 -77.0   3302548 Washington DC        20009 District of Columbia
#> 6 Dunkin' Donuts   38.9 -77.0  13589373 Washington DC        20024 District of Columbia
```

Overlay points with `addCircleMarkers()`

```r
leaflet(data = dc_coffee_shops) %>%
  addTiles() %>%
  addCircleMarkers(
    lng = ~long,
    lat = ~lat,
    popup = ~name,
    radius = 2)
```



Make a color pallete with `colorFactor()`

Make the palette

```r
pal <- colorFactor(
  palette = "Set2",
  levels  = c(
    "Starbucks",
    "Dunkin' Donuts",
    "Peet's Coffee & Tea",
    "Baskin Robbins",
    "The Coffee Bean & Tea Leaf"))
```

`pal()` links the shop *name* to a *color*:

```r
pal("Starbucks")
```

```
#> [1] "#66C2A5"
```

```r
pal("Dunkin' Donuts")
```

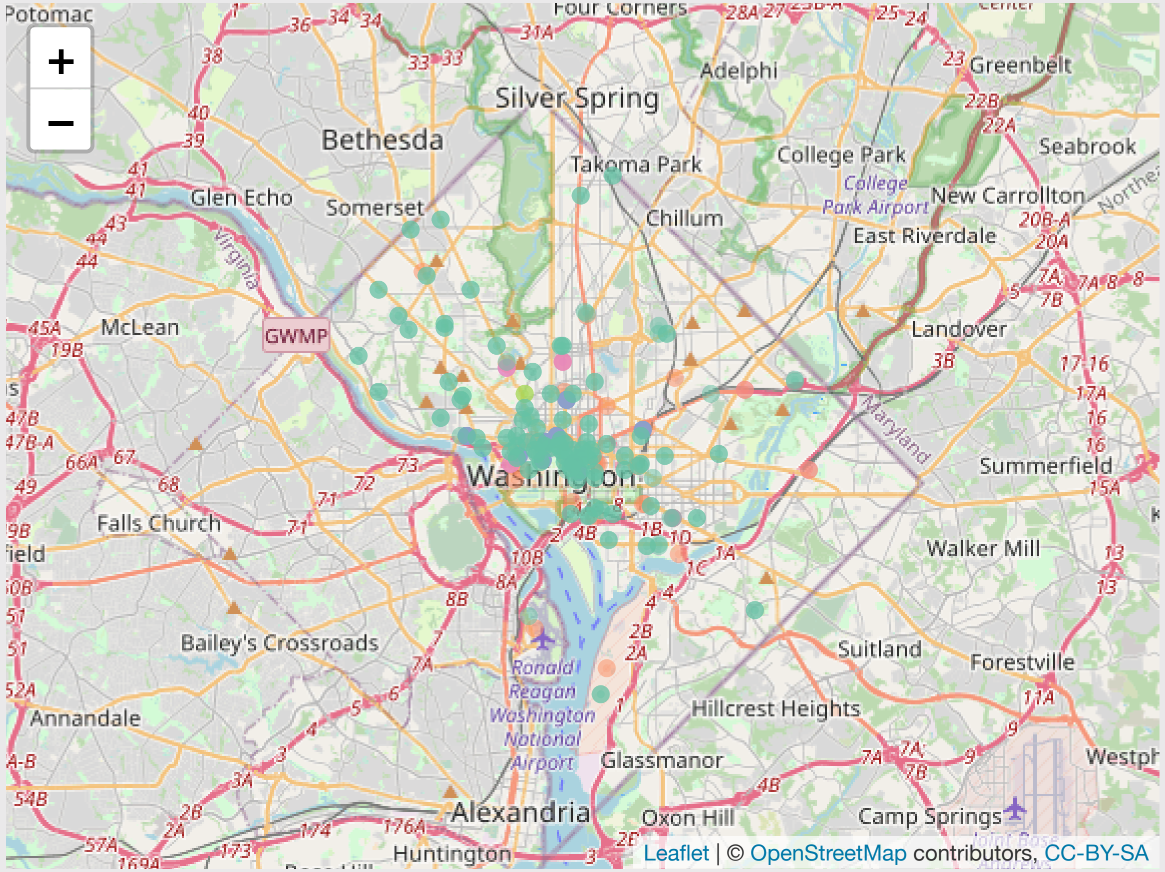
```
#> [1] "#FC8D62"
```

```r
pal("Baskin Robbins")
```

```
#> [1] "#E78AC3"
```

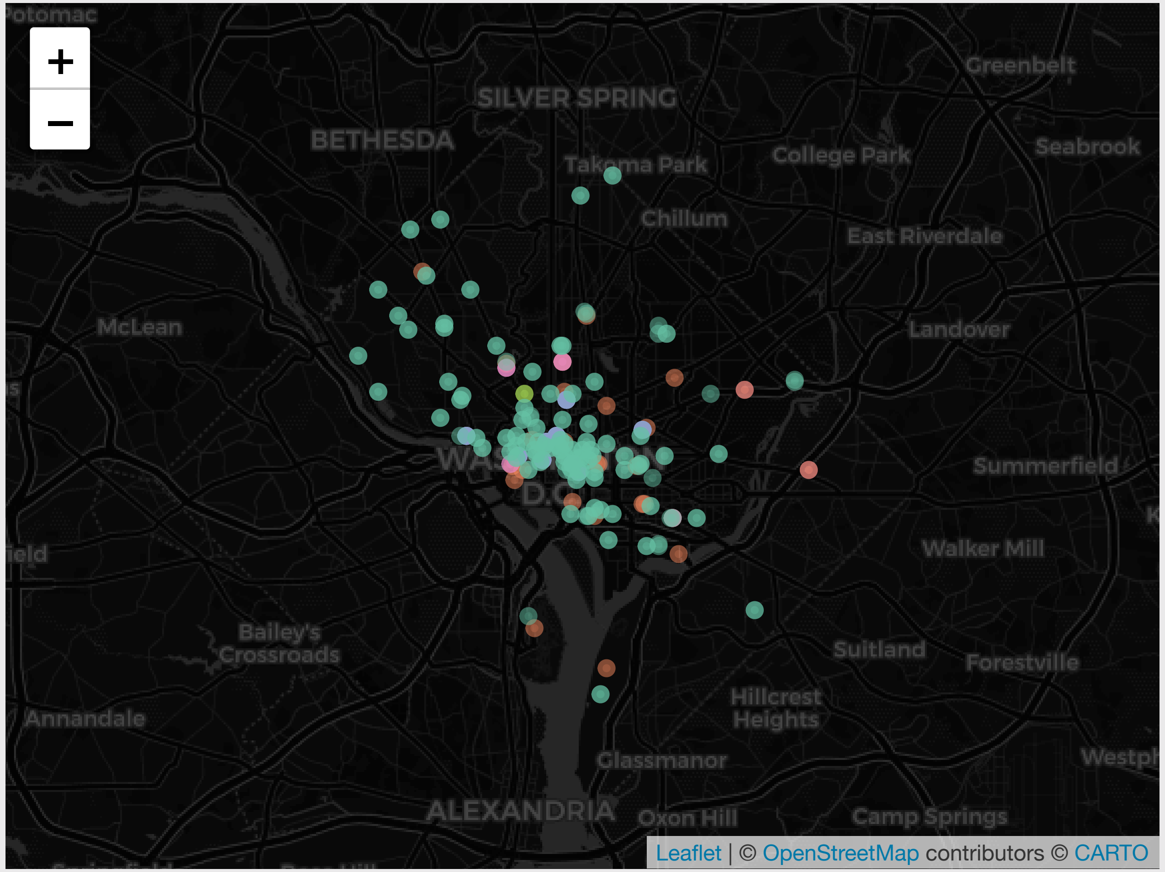
Use `pal()` to color the points

```r
leaflet(data = dc_coffee_shops) %>%
  addTiles() %>%
  addCircleMarkers(
    lng = ~long,
    lat = ~lat,
    popup = ~name,
    color = ~pal(name),
    radius = 2)
```



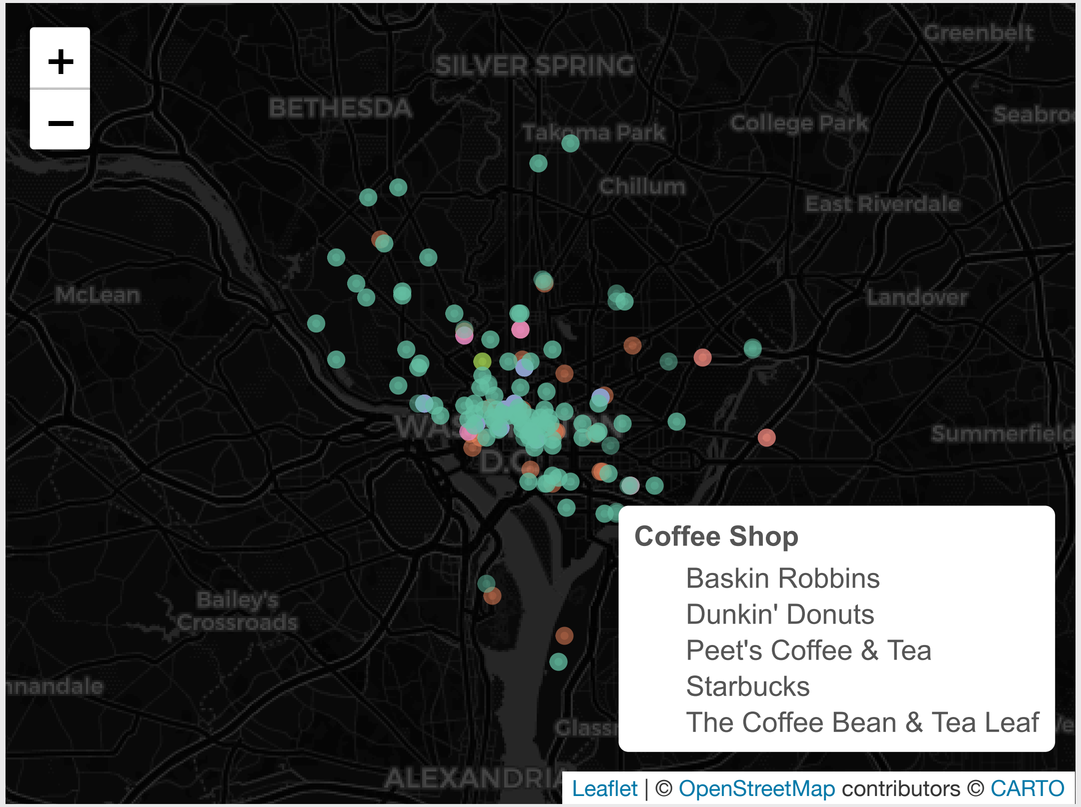
Use a dark tile scheme to make colors pop

```r
leaflet(data = dc_coffee_shops) %>%
  # addTiles() %>%
  addProviderTiles(providers$CartoDB.DarkMatter) %>%
  addCircleMarkers(
    lng = ~long,
    lat = ~lat,
    popup = ~name,
    color = ~pal(name),
    radius = 2)
```



Add a legend with `addLegend()`

```
leaflet(data = dc_coffee_shops) %>%
  addProviderTiles(providers$CartoDB.DarkMatter) %>%
  addCircleMarkers(
    lng = ~long,
    lat = ~lat,
    popup = ~name,
    color = ~pal(name),
    radius = 2) %>%
  addLegend(
    position = "bottomright",
    pal = pal,
    values = ~name,
    title = "Coffee Shop",
    opacity = 1)
```



Coffee Shop
- Baskin Robbins
- Dunkin' Donuts
- Peet's Coffee & Tea
- Starbucks
- The Coffee Bean & Tea Leaf

Leaflet | © OpenStreetMap contributors © CARTO

Overlaying data with leaflet

- Points

- Choropleth maps

How to make a choropleth leaflet map

Get the "fill" data

```r
milk_production <- read_csv(here::here(
  'data', 'milk_production.csv'))

milk_2017 <- milk_production %>%
  filter(year == 2017) %>%
  select(name = state, milk_produced) %>%
  mutate(milk_produced = milk_produced / 10^9)
```

Join to my "map" data

```r
library(rnaturalearth)

state_milk <- ne_states(
  country = 'united states of america',
  returnclass = 'sf') %>%
  left_join(milk_2017, by = 'name')
```

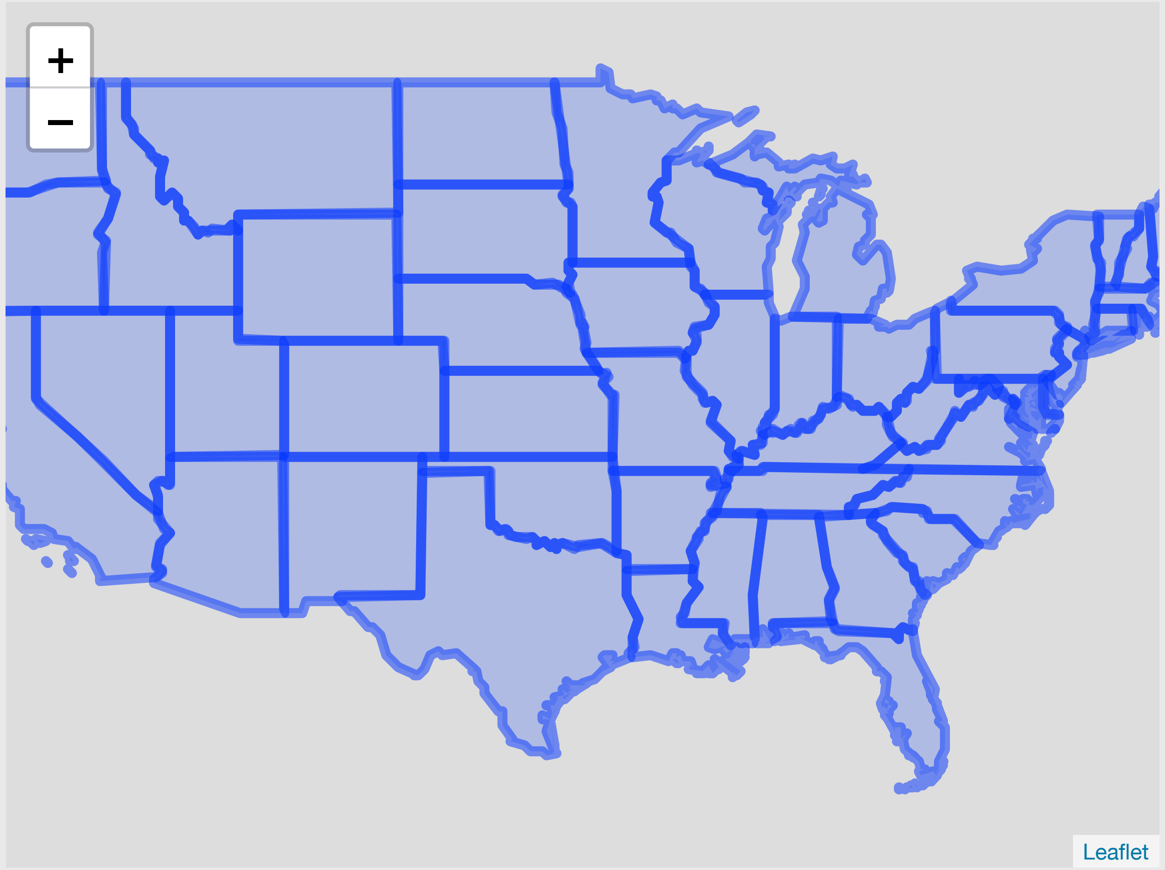
```r
state_milk %>%
  select(name, milk_produced) %>%
  head()
```

```
#> Simple feature collection with 6
#> Geometry type: MULTIPOLYGON
#> Dimension:     XY
#> Bounding box:  xmin: -124.7346 y
#> CRS:           +proj=longlat +da
#>             name milk_produced
#> 1    Minnesota          9.864 MUL
#> 2   Washington          6.526 MUL
#> 3        Idaho         14.627 MUL
#> 4      Montana          0.288 MUL
#> 5 North Dakota          0.345 MUL
#> 6     Michigan         11.231 MUL
```

Add state shapes with addPolygons()

Notice that I didn't include addTiles()

```r
leaflet(data = state_milk) %>%
  setView(-96, 37.8, 4) %>%
  addPolygons()
```



Make a color pallete with `colorBin()`

Make the palette

```
pal <- colorBin(
  palette = "YlOrRd",
  domain = state_milk$milk_produced)
```

`pal()` links the milk produced color:

```
pal(1)
```

```
#> [1] "#FFFFCC"
```

```
pal(10)
```

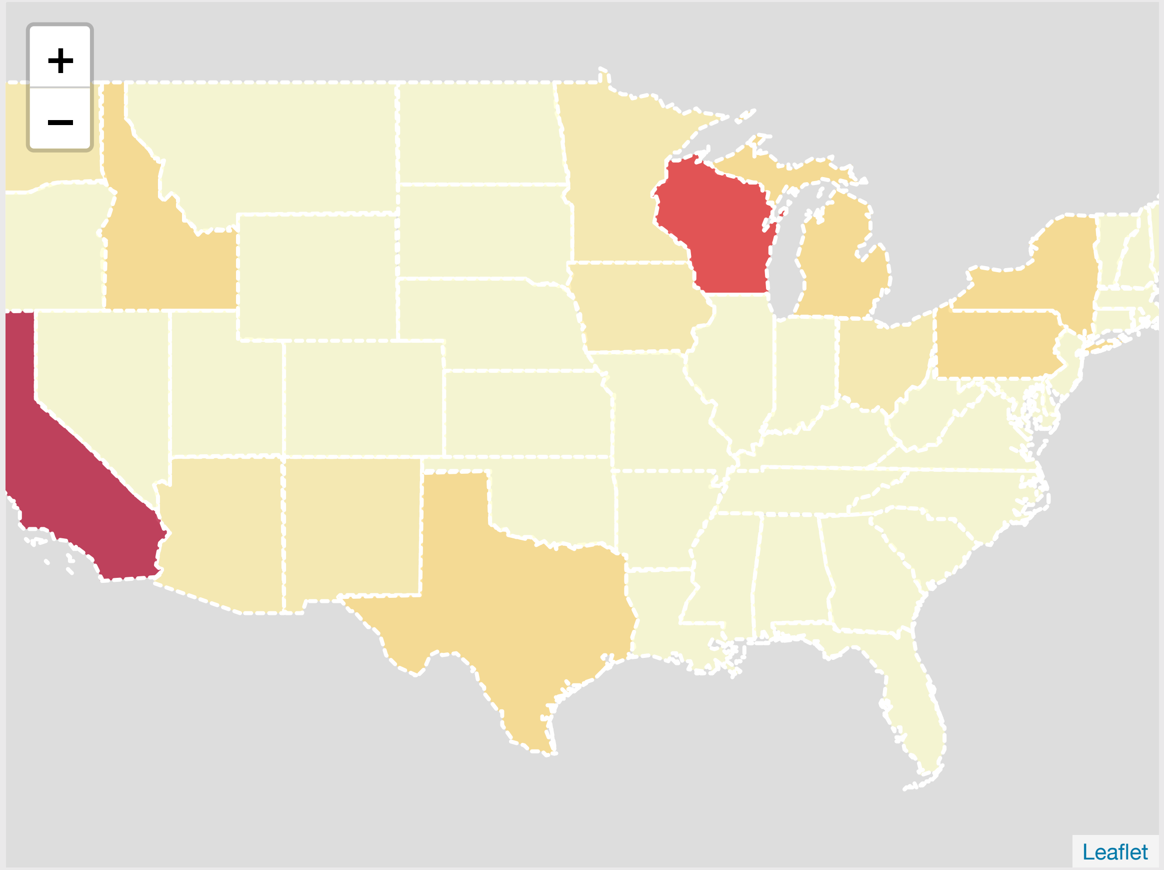
```
#> [1] "#FED976"
```

```
pal(20)
```

```
#> [1] "#FD8D3C"
```

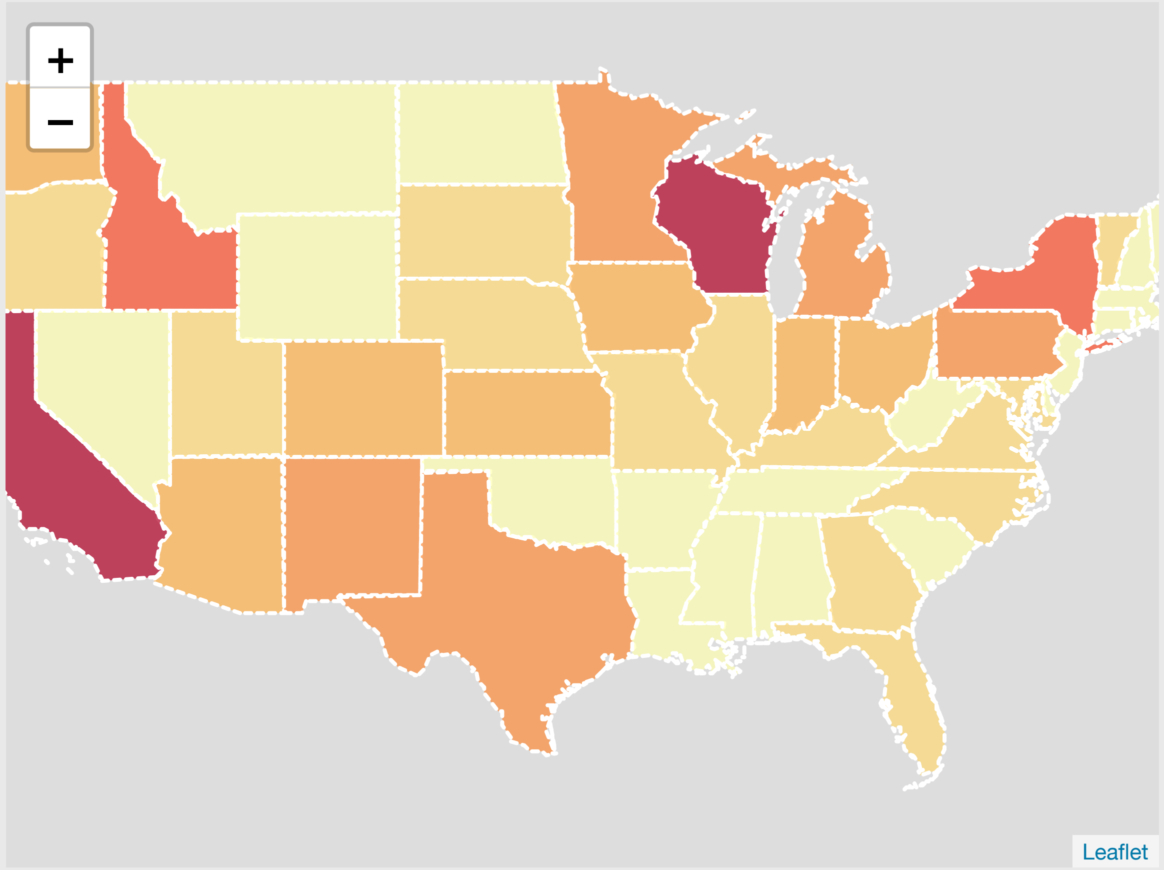
Use `pal()` to fill the polygons

```r
pal <- colorBin(
  palette = "YlOrRd",
  domain = state_milk$milk_produced)

leaflet(data = state_milk) %>%
  setView(-96, 37.8, 4) %>%
  addPolygons(
    fillColor = ~pal(milk_produced),
    weight = 2,
    opacity = 1,
    color = "white",
    dashArray = "3",
    fillOpacity = 0.7)
```



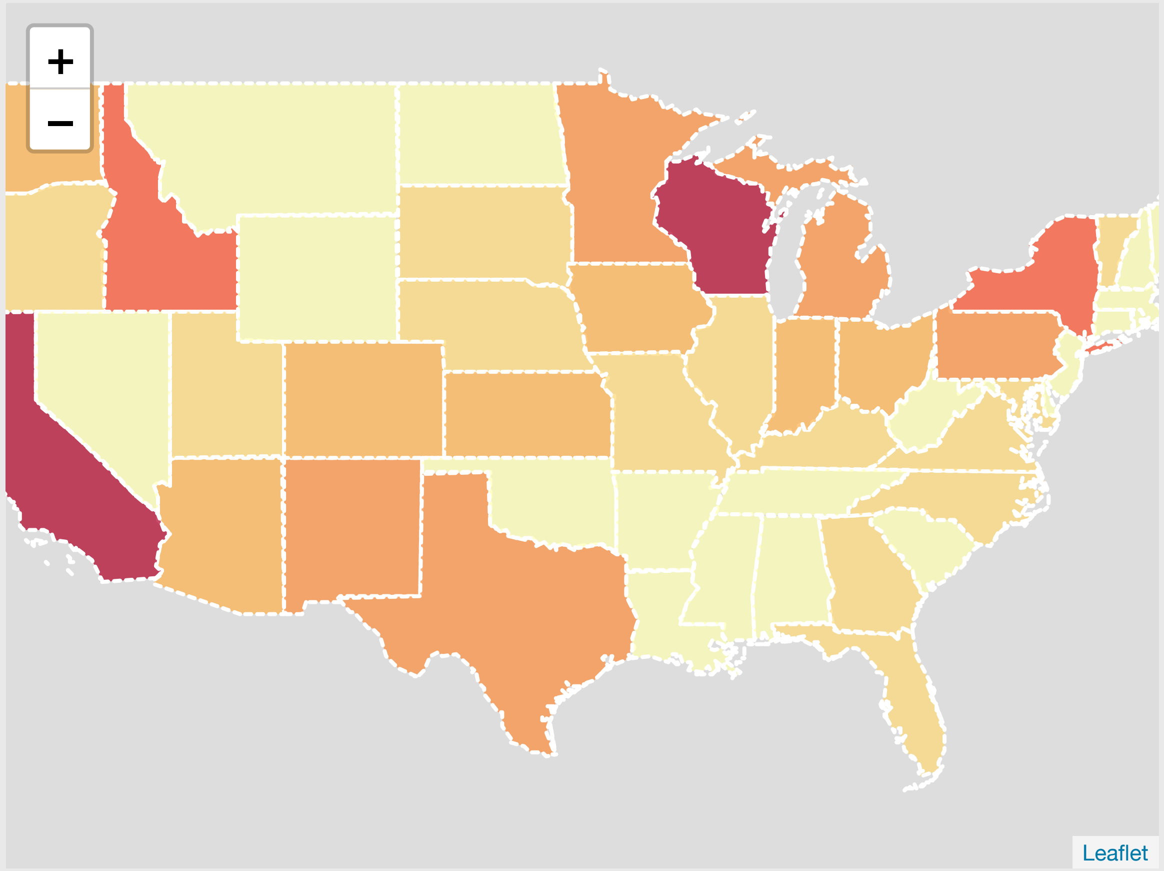
Manually set bins in `pal()`

```r
pal <- colorBin(
  palette = "YlOrRd",
  bins = round(seq(0, sqrt(40), length.out =
  domain = state_milk$milk_produced)

leaflet(data = state_milk) %>%
  setView(-96, 37.8, 4) %>%
  addPolygons(
    fillColor = ~pal(milk_produced),
    weight = 2,
    opacity = 1,
    color = "white",
    dashArray = "3",
    fillOpacity = 0.7)
```



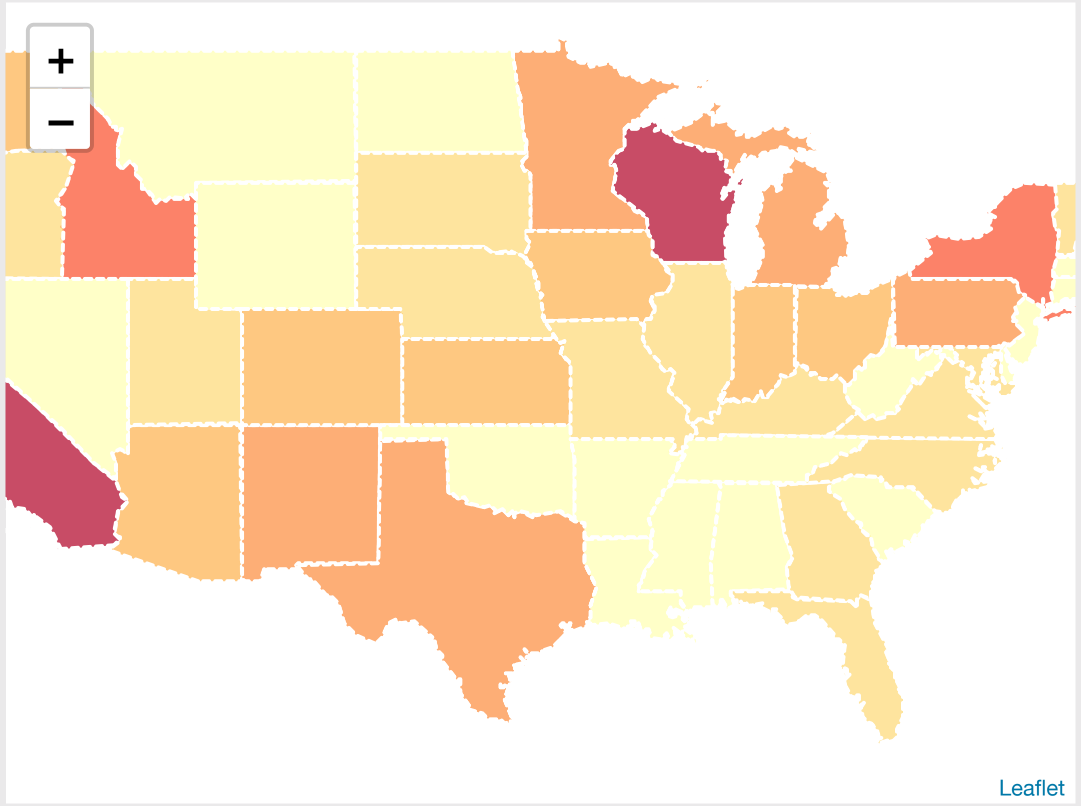
Make it interactive with `highlight` argument

```r
leaflet(data = state_milk) %>%
  setView(-96, 37.8, 4) %>%
  addPolygons(
    fillColor = ~pal(milk_produced),
    weight = 2,
    opacity = 1,
    color = "white",
    dashArray = "3",
    fillOpacity = 0.7,
    highlight = highlightOptions(
      weight = 5,
      color = "#666",
      dashArray = "",
      fillOpacity = 0.7,
      bringToFront = TRUE))
```



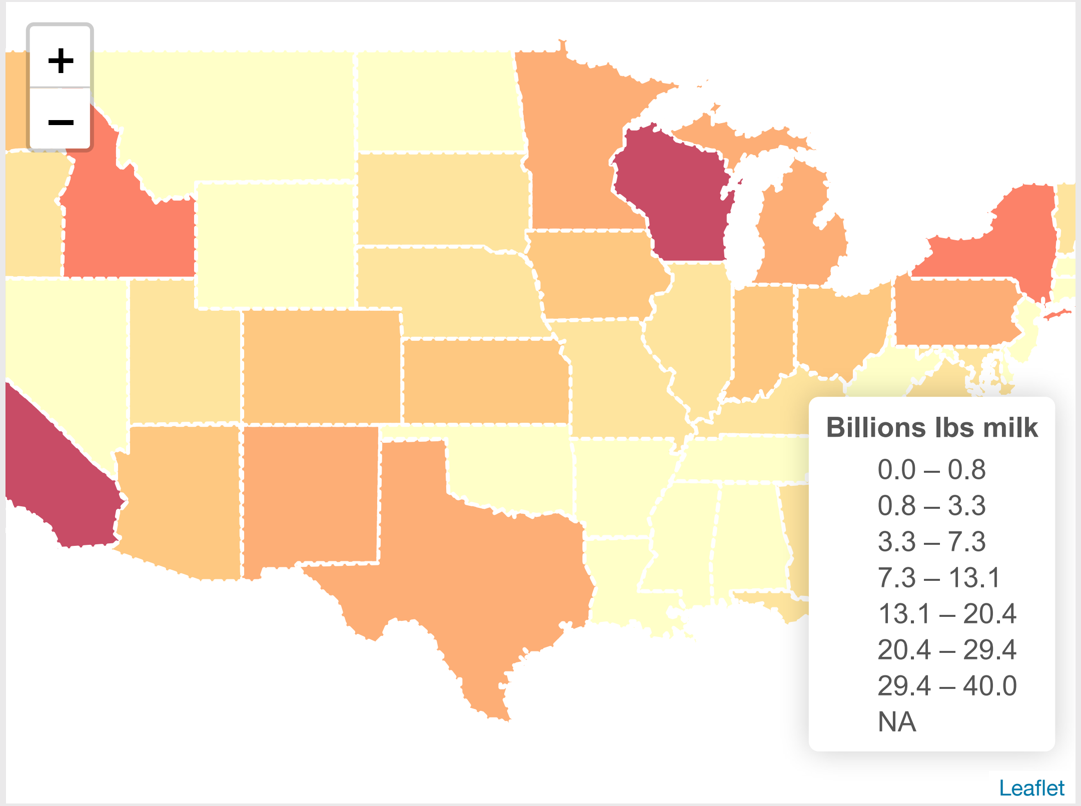
Add labels

```r
state_milk <- state_milk %>%
  mutate(label = paste(name, ": ",
    round(milk_produced, 2), " B lbs", sep = ""))

leaflet(data = state_milk) %>%
  setView(-96, 37.8, 4) %>%
  addPolygons(
    fillColor = ~pal(milk_produced),
    weight = 2,
    opacity = 1,
    color = "white",
    dashArray = "3",
    fillOpacity = 0.7,
    highlight = highlightOptions(
      weight = 5,
      color = "#666",
      dashArray = "",
      fillOpacity = 0.7,
      bringToFront = TRUE),
    label = state_milk$label)
```



Add a legend with `addLegend()`

```r
leaflet(data = state_milk) %>%
  setView(-96, 37.8, 4) %>%
  addPolygons(
    fillColor = ~pal(milk_produced),
    weight = 2,
    opacity = 1,
    color = "white",
    dashArray = "3",
    fillOpacity = 0.7,
    highlight = highlightOptions(
      weight = 5,
      color = "#666",
      dashArray = "",
      fillOpacity = 0.7,
      bringToFront = TRUE),
    label = state_milk$label) %>%
  addLegend(
    pal = pal, values = ~milk_produced, opacity = 0.7,
    title = "Billions lbs milk",
    position = "bottomright")
```



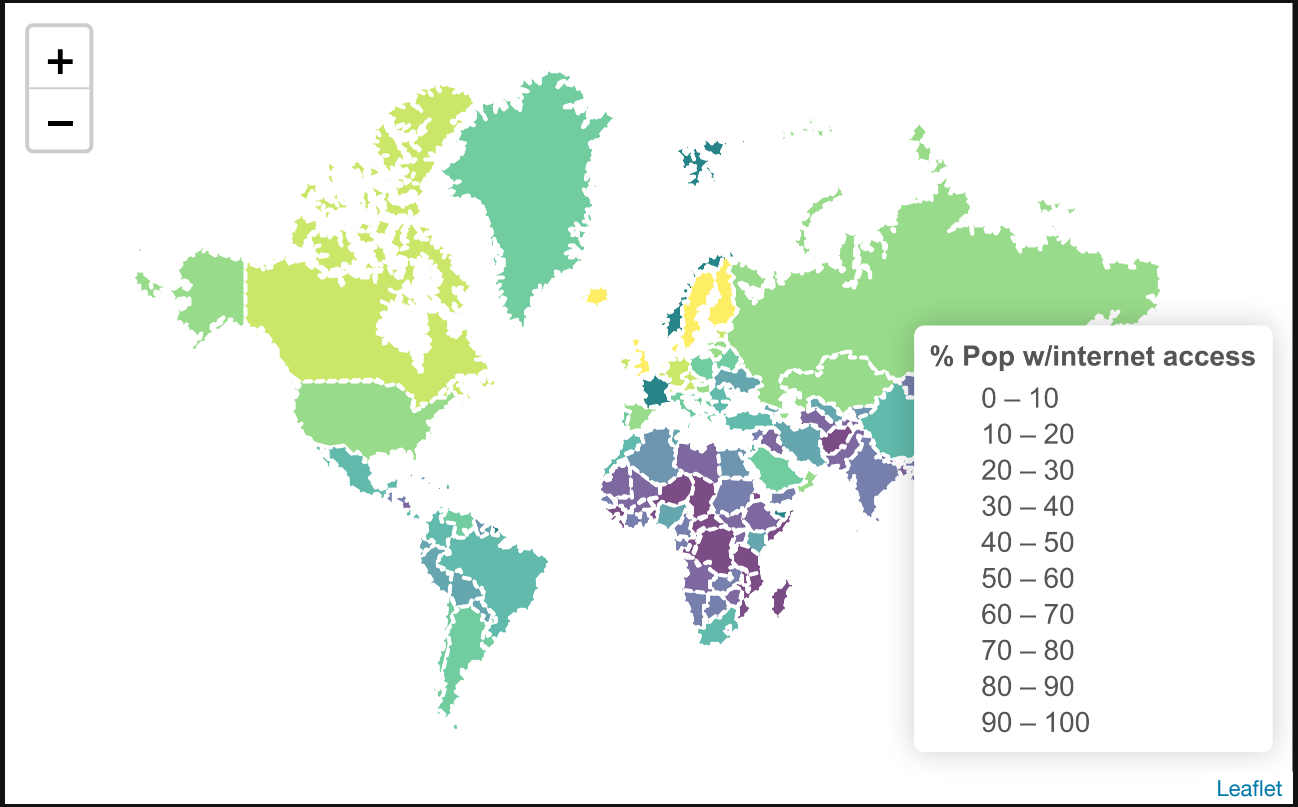
Reference guide:
https://rstudio.github.io/leaflet/

Your Turn: Interactive Maps

20:00

Use the `world_internet_2015` data frame to create this interactive leaflet map of internet access by country in 2015.

Note: I've already created the `world_internet_2015` data frame by joining the `internet_users` data frame to the `world` data frame from the **rnaturalearth** library.


+
−
% Pop w/internet access
0 – 10
10 – 20
20 – 30
30 – 40
40 – 50
50 – 60
60 – 70
70 – 80
80 – 90
90 – 100
Leaflet