Heat Map Examples

sugar_consumption.Rmd

AJR

2024-05-16

Introduction

This material may be of use for this short course:

Session Date / Time: 8/4/2024, 1:00 pm - 4:00 pm (California Time)
Title: Heatmaps, Colour Spaces, and Bubble Plots With R

The term *heatmap* may mean different things to different people. Some people may strongly associate the term with bioinformatics. Generally speaking, we are interested in cases where we have an x- variable and an y- variable plotted to the horizontal and vertical axes, and a third variable that is plotted to colour. We can also include cases where spatial polygons are plotted to colour of fill; such plots seems to be popular (and some people may strongly associate the term *map* with a geographic map).

We might call a scatterplot that maps a third variable to colour a *confetti plot*; the term *confetti plot* does not yet seem to be in general use.

Load some R packages:

```r
suppressPackageStartupMessages(library(sf))
library(sp)
library(ggplot2)
library(maps)
suppressPackageStartupMessages(library(dplyr))
```

Draw a Map of the World

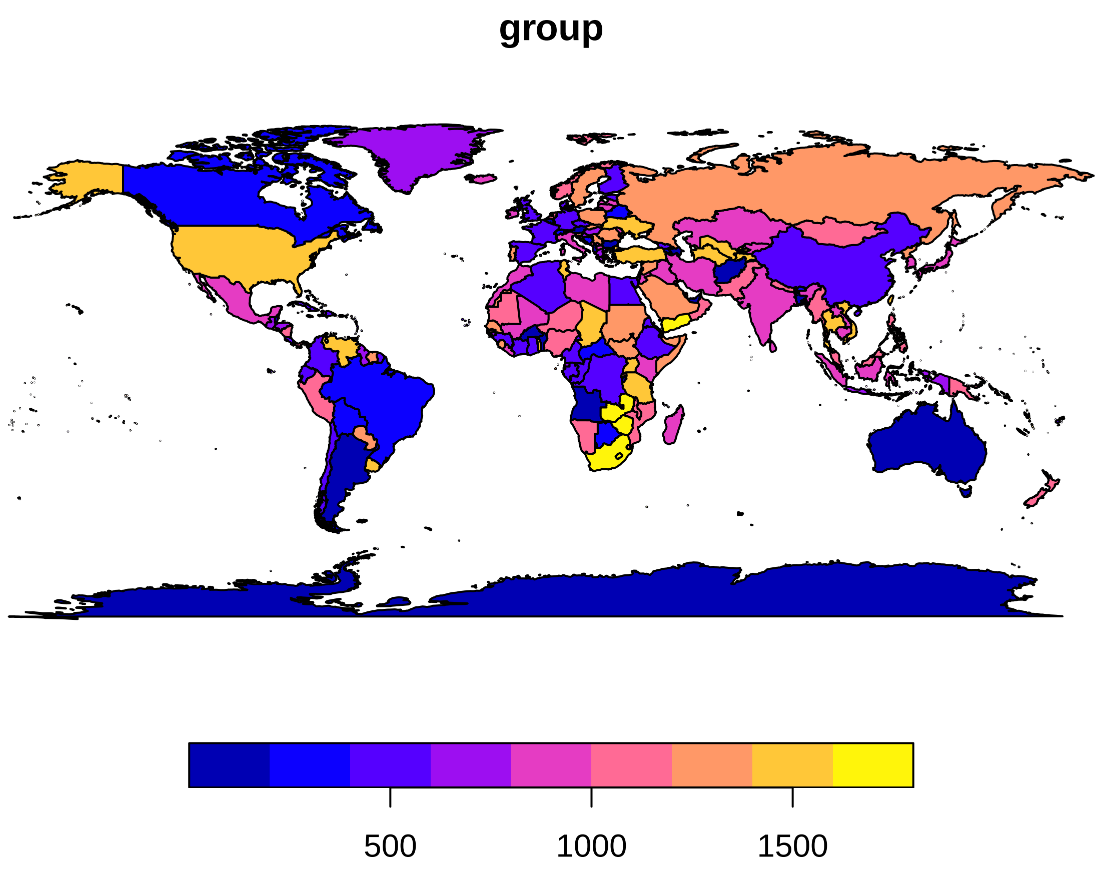
Get some map data:

```r
world_map <- map_data("world")
class(world_map)
```

```
## [1] "data.frame"
```

Create an sf dataframe:

```r
# Perhaps a better method:
# XY <- st_sfc(st_multipoint(cbind(world_map$long, world_map$lat)))
# XY <- st_cast(XY, "POINT")
# world_map2 <- world_map
# world_map2$geometry <- XY etc.

xy_points <- SpatialPoints(cbind(world_map$long, world_map$lat))
xy_points <- st_as_sfc(xy_points)

ids <- world_map$group

xy_multipoint <- st_cast(xy_points, to = "MULTIPOINT", ids = ids)
xy_polygon <- st_cast(xy_multipoint, to = "POLYGON")

polygon_sf <- st_sf(data.frame(geometry = xy_polygon, group = unique(ids)))

# Not run:
plot(polygon_sf)
```



It seems that there ought to be an easier way to transform the `maps` package data frame into an sf data frame.

Combine Sugar Availability Data with Map Data

Combine the sugar data with the map data:

```r
world_map_summary <- summarize(group_by(world_map, group, region, subregion),
                               .groups = "keep")

polygon_sf <- left_join(polygon_sf, world_map_summary, by = "group")


sugar_data <- read.csv("WHO_sugar_data.csv")


sugar_data$region <- sugar_data$Location


polygon_sf <- left_join(polygon_sf, sugar_data[,c("region", "Value")],
                        by = "region")

ggplot(polygon_sf) +
  geom_sf(aes(fill = Value))
```

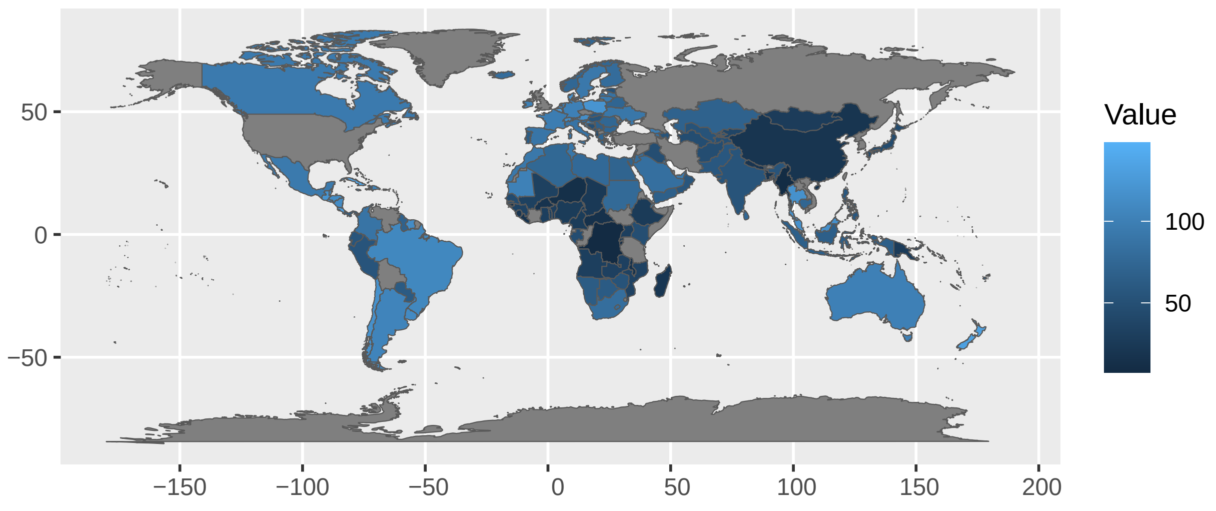


Figure 1: Many of the missing values are due to mismatches in region names.

Several of the `region` names in the map data do not match the `Location` names in the sugar data. Will will fix some of these mismatches manually.

```
sugar_data[sugar_data$Location == "United States of America",]$region <-
  "USA"

sugar_data[sugar_data$Location == "United Kingdom of Great Britain and Northern Ireland"
  "UK"

sugar_data[sugar_data$Location == "Russian Federation",]$region <- "Russia"
sugar_data[sugar_data$Location =="Bolivia (Plurinational State of)",]$region <-
  "Bolivia"

sugar_data[sugar_data$Location =="Venezuela (Bolivarian Republic of)",]$region <-
  "Venezuela"
```

Remove previous `Value` column to avoid a duplicated column:

```
polygon_sf <- polygon_sf[,names(polygon_sf) != "Value"]
```

Re-join the sugar data:

```
polygon_sf <- left_join(polygon_sf, sugar_data[,c("region", "Value")],
                        by = "region")
```

Re-draw the map:

```
ggplot(polygon_sf) +
  geom_sf(aes(fill = Value))
```

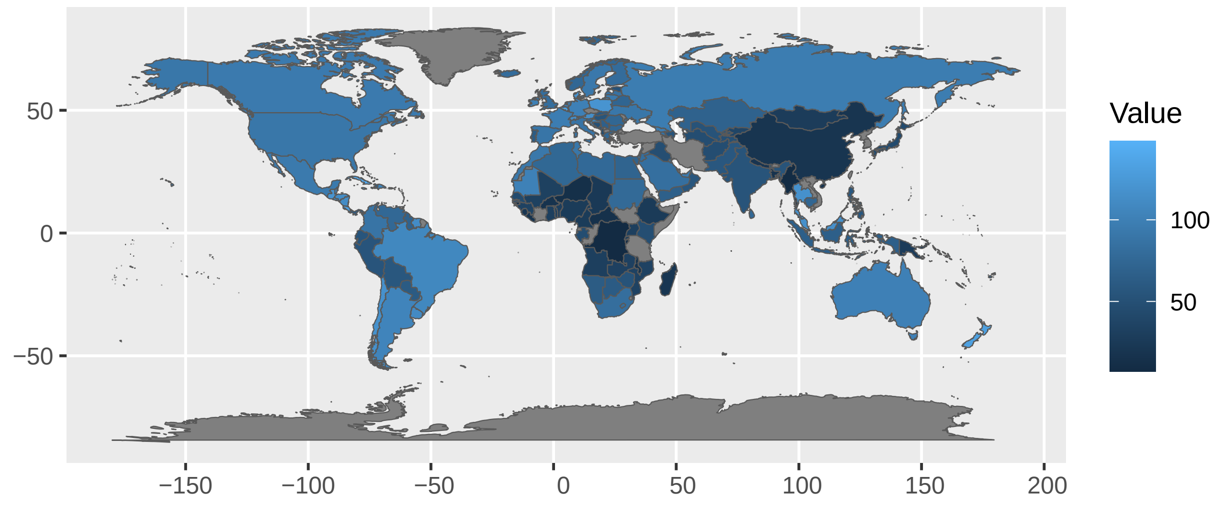


Figure 2: Now fewer countries have data missing.

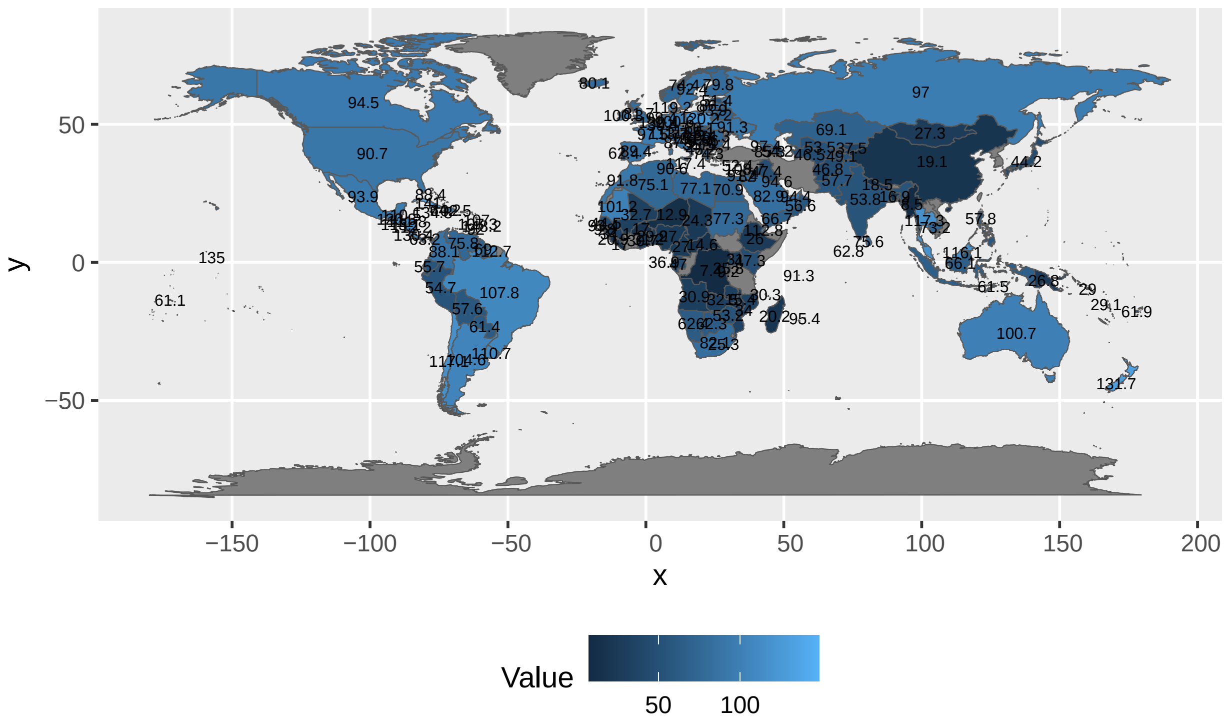
The `left_join()` function will not automatically replace the values in the `Value` column; thus that column has been dropped before re-joining the sugar data values.

Add Text Labels to the Map

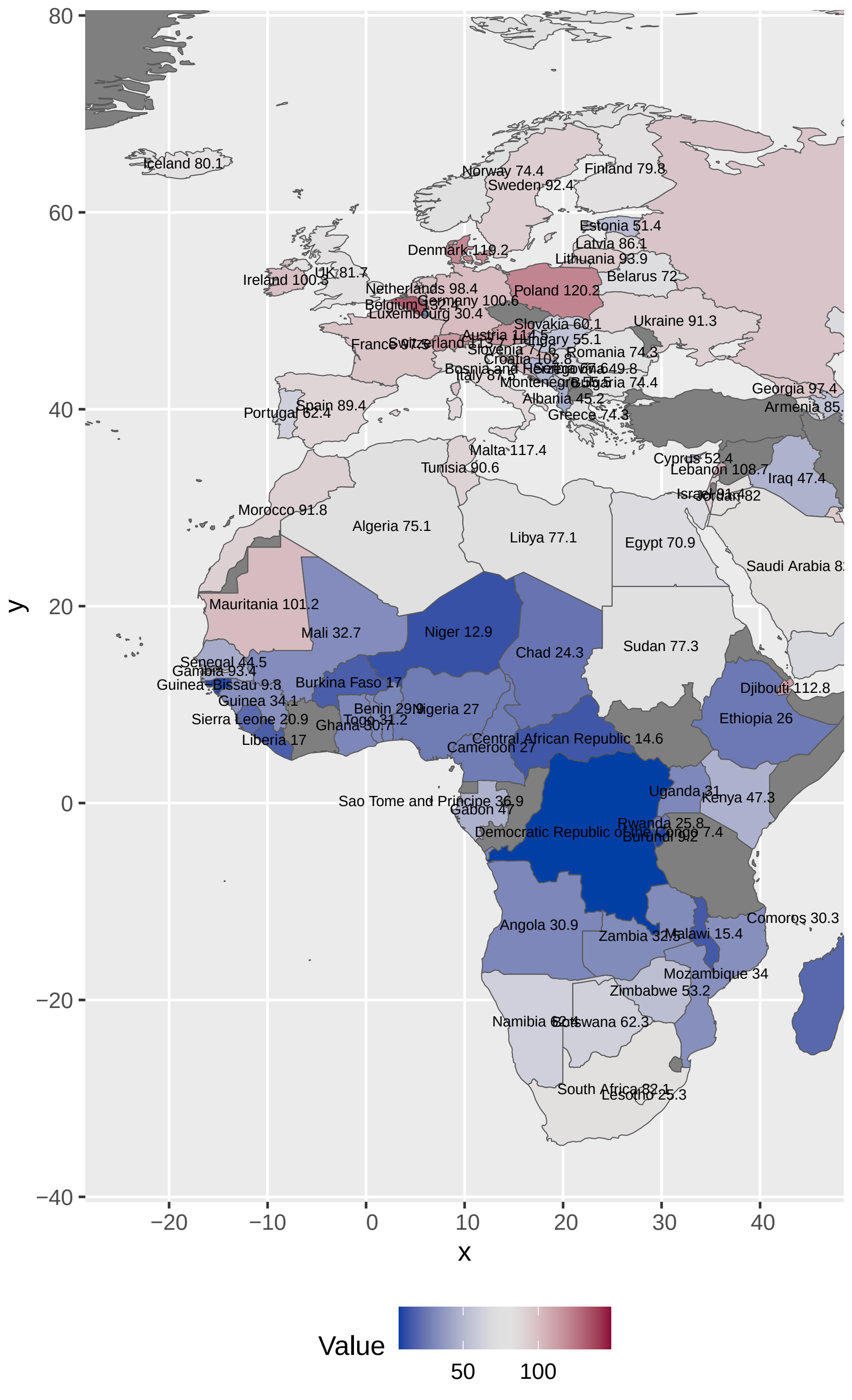
The colour scale may let us seem some general patterns; however, the colour scale does not make it easy to read off a data value for any particular country.

To add text labels to the map, we will want labelling points:

```
label_sf <-
  reframe(group_by(polygon_sf[polygon_sf$region != "Antarctica",], region),
          Value = unique(Value), geometry = st_union(geometry))

label_sf$centroid <- st_centroid(label_sf$geometry, of_largest_polygon = TRUE)
label_sf$point_on_surface <- st_point_on_surface(label_sf$geometry)

ggplot(polygon_sf) +
  geom_sf(aes(fill = Value)) +
  geom_sf_text(aes(label = Value, geometry = centroid), data = label_sf,
               size = 2, na.rm = TRUE) +
  theme(legend.position = "bottom")
```



```r
pal <- colorspace::diverge_hcl(n = 20)

ggplot(polygon_sf) +
  geom_sf(aes(fill = Value)) +
  geom_sf_text(aes(label = paste(region, Value), geometry = centroid),
               data = label_sf[!is.na(label_sf$Value), ], size = 2, na.rm = TRUE) +
  xlim(-25, 45) + ylim(-35, 75) +
  theme(legend.position = "bottom") +
  scale_fill_gradientn(colours = pal)
```


80
60
40
20
0
−20
−40
y
−20
−10
0
10
20
30
40
x
Iceland 80.1
Norway 74.4
Finland 79.8
Sweden 92.4
Estonia 51.4
Latvia 86.1
Denmark 119.2
Lithuania 93.9
UK 81.7
Ireland 100.3
Belarus 72
Netherlands 98.4
Poland 120.2
Belgium 132.4
Germany 100.6
Luxembourg 30.4
Slovakia 60.1
Ukraine 91.3
Austria 114.5
France 97.5
Switzerland 113.7
Hungary 55.1
Slovenia 77.6
Romania 74.3
Croatia 102.8
Bosnia and Herzegovina 67.6
Serbia 49.8
Italy 87.5
Montenegro 55.5
Bulgaria 74.4
Georgia 97.4
Albania 45.2
Spain 89.4
Portugal 62.4
Armenia 85.
Greece 74.3
Malta 117.4
Cyprus 52.4
Lebanon 108.7
Tunisia 90.6
Iraq 47.4
Israel 91.4
Jordan 82
Morocco 91.8
Algeria 75.1
Libya 77.1
Egypt 70.9
Saudi Arabia 8
Mauritania 101.2
Mali 32.7
Niger 12.9
Chad 24.3
Sudan 77.3
Senegal 44.5
Gambia 93.4
Guinea-Bissau 9.8
Burkina Faso 17
Djibouti 112.8
Guinea 34.1
Benin 29.9
Nigeria 27
Sierra Leone 20.9
Ghana 30.7
Togo 31.2
Ethiopia 26
Central African Republic 14.6
Liberia 17
Cameroon 27
Uganda 31
Kenya 47.3
Sao Tome and Principe 36.9
Gabon 47
Rwanda 25.8
Democratic Republic of the Congo 7.4
Burundi 9.2
Comoros 30.3
Angola 30.9
Zambia 32.5
Malawi 15.4
Mozambique 34
Zimbabwe 53.2
Namibia 62.4
Botswana 62.3
South Africa 82.1
Lesotho 25.3
Value
50
100

9