

# Visualizing and mapping Twitter data

---

Connor Gilroy

2017-10-24

# Why visualize?

*“You should look at your data.”*

—Kieran Healy, Data Visualization for Social Science

Visualization is a powerful tool for *exploratory data analysis*.

It helps you spot patterns and get a descriptive sense for the data you're working with.

## Packages

The key package here is `leaflet`:

```
library(countrycode)
library(forcats)
library(leaflet)
library(stringr)
library(tidyverse)
```

## Data: tweets from Africa

```
tweets_unfiltered <- readRDS("data/tweets_africa.rds")
```

These African tweets were collected using a rectangular “bounding box” around Africa. Because of this, the data include tweets from the Middle East and Mediterranean. (See `collect_tweets.R` for the collection code.)

We want to filter down to tweets from African countries only.

## countrycode

We can use the `countrycode` package to get a list of countries in Africa and their 2-letter country codes

```
cc_africa <-  
  countrycode_data %>% filter(continent == "Africa")  
  
tweets_africa <-  
  tweets_unfiltered %>%  
  filter(country_code %in% cc_africa$iso2c)
```

## Data join

We can use some of the information in the countrycode data in our plots and maps, so let's join the two together.

```
tweets_africa <-  
  left_join(tweets_africa, cc_africa,  
            by = c("country_code" = "iso2c"))
```

## Example 1: Non-spatial representation

How many tweets are there from each country?

A good way to represent this is with a simple dotplot.

## Summarize the data

```
tweets_africa_summarized <-
  tweets_africa %>%
  group_by(country_code, country.name.en, region) %>%
  count() %>%
  ungroup()
```

We also shorten some country names (code not shown in slides)

## Build the plot

```
n_tweets_plot <-  
  ggplot(tweets_africa_summarized,  
          aes(x = n, y = fct_reorder(country.name.en, n))) +  
  geom_point(size = 2) +  
  facet_wrap(~region, scales = "free_y", ncol = 2) +  
  labs(y = NULL, x = NULL) +  
  theme_minimal(base_size = 20) +  
  # make the base_size smaller on your laptop!  
  # it's big for the projector  
  theme(axis.text.y = element_text(size = rel(.6)),  
        strip.text = element_text(size = rel(1.2)),  
        plot.title = element_text(size = rel(1.5)))
```

# Number of tweets by country

## Western Africa

Nigeria  
Ghana  
Senegal  
Côte d'Ivoire  
Burkina Faso  
Mali  
Togo  
Guinea  
Cabo Verde  
Guinea-Bissau  
Sierra Leone  
Mauritania  
Alder  
Guinea-Bissau  
St. Helena



## Northern Africa

Egypt  
Algeria  
Libya  
Morocco  
Sudan  
Tunisia  
Western Sahara



## Middle Africa

Cameroon  
Angola  
DRC  
Congo  
Gabon  
Central African Republic  
Chad  
Sao Tome and Principe  
Equatorial Guinea



## Eastern Africa

Kenya  
Tanzania  
Zimbabwe  
Mozambique  
Malawi  
Rwanda  
Burundi  
Somalia  
Ethiopia  
Moldova  
Mongolia  
Serbia  
Burundi  
Croatia  
Eritrea  
Mayotte



## Southern Africa

South Africa  
Botswana  
Namibia  
Swaziland  
Lesotho



## Interlude: Static map

```
library(ggthemes)
map_data_africa <-
  map_data("world",
           region = cc_africa$country.name.en.regex)
```

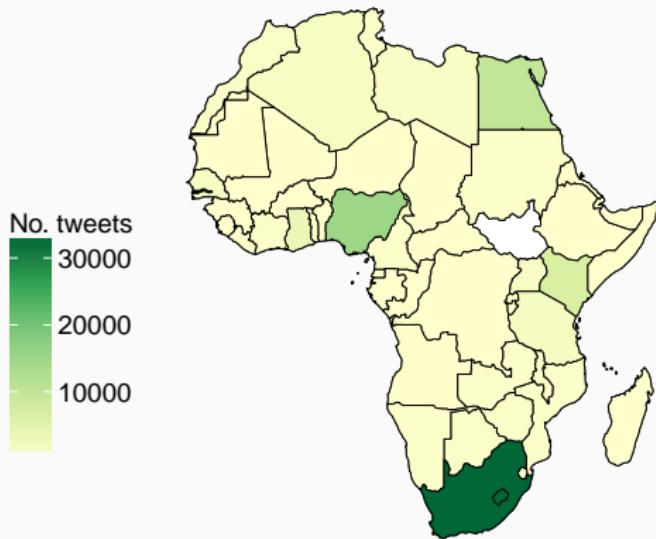
Again, we need to modify some country names (code not shown in slides).  
We must also rename the `country.name.en.regex` to `region`.

## Base map

```
base_map_africa <-
  ggplot(map_data_africa, aes(map_id = region)) +
  geom_map(map = map_data_africa,
           fill = "white", color = "black", size = .25) +
  expand_limits(x = map_data_africa$long,
                y = map_data_africa$lat) +
  coord_map() +
  theme_map() +
  theme(legend.background =
        element_rect(fill = alpha("white", 0))))
```

## Map with fill gradient

```
static_map_africa <-
base_map_africa +
geom_map(map = map_data_africa,
         data = tweets_africa_summarized2,
         aes(fill = n), color = "black", size = .25) +
scale_fill_distiller(palette = "YlGn", direction = 1) +
labs(fill = "No. tweets") +
theme(legend.position = "left",
      legend.justification = c("right", "center"),
      legend.key.size = unit(2, "line"),
      legend.text = element_text(size = rel(2)),
      legend.title = element_text(size = rel(2)))
```



## Example 2: Interactive map

For exploratory purposes, being able to click and zoom on a map is useful. The **leaflet** R package provides an interface to the Leaflet JavaScript library, which lets us do just that.

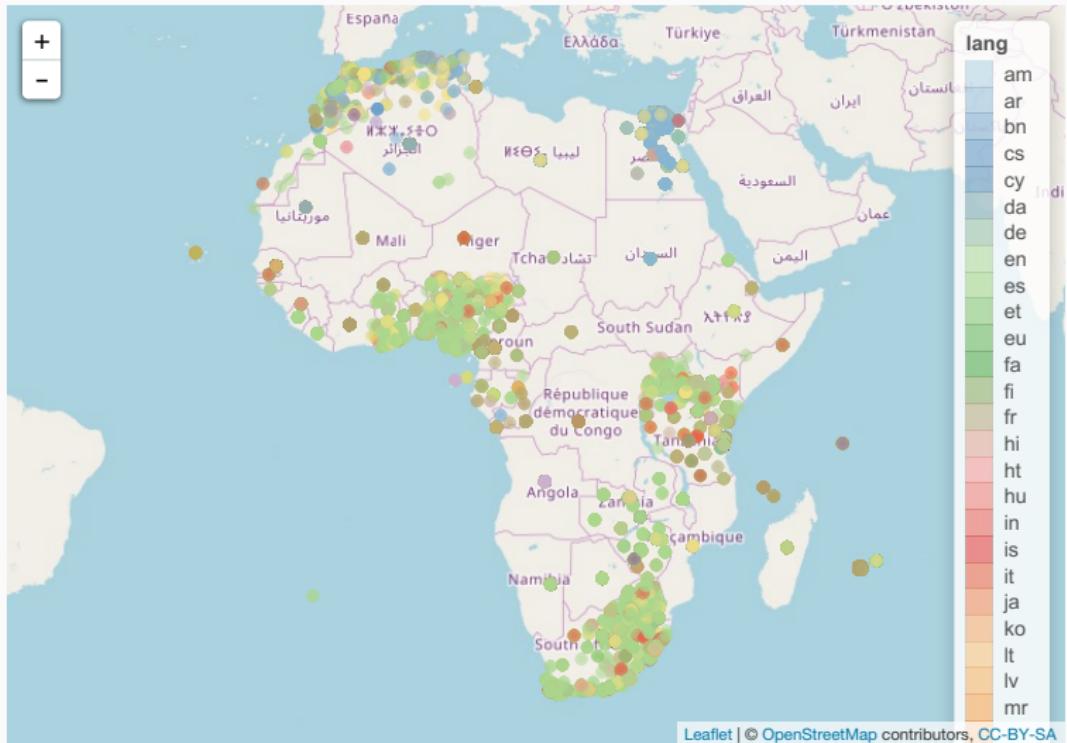
We'll map individual tweets, with two additional innovations:

1. Popup markers containing the text of the tweet
2. Tweets color-coded by language

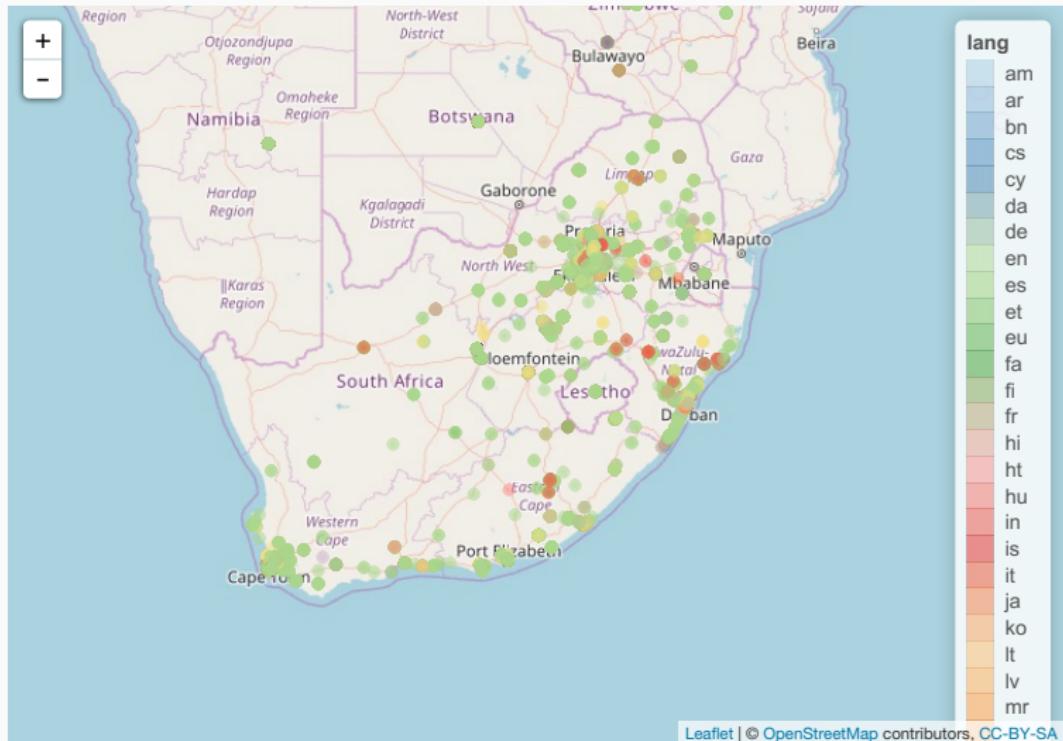
## Interactive mapping with Leaflet

`leaflet()` creates the widget; *tiles* are the map background, and *markers* are the points based on the data.

```
pal <- colorFactor("Paired", tweets_africa$lang)
map_africa <-
  leaflet(data = tweets_africa) %>%
  addTiles() %>%
  addCircleMarkers(lat = ~place_lat, lng = ~place_lon,
                  popup = ~str_c(screen_name, ": ", text),
                  radius = 2,
                  color = ~pal(lang)) %>%
  addLegend(pal = pal, values = ~lang)
```



# Zoom in on South Africa



## Exercises

**Exercise 1:** Click on a few markers and look at the text of the tweets. How accurate does the assigned language generally seem to be?

**Exercise 2:** Try plotting tweets you've collected using `leaflet`.

**Challenge exercise:** Some countries have tweets from many locations, while others show all tweets in the same spot. What's the difference? Look at the tweets data to see if you can figure it out. (Hint: we used `place_lat` and `place_lon`, so the difference has to do with places.)

## Resources

Leaflet documentation:

<https://rstudio.github.io/leaflet/>

Kieran Healy's online book, *Data Visualization for Social Science*:

<http://socviz.co/>

The examples in this module were inspired by Chapter 7, “Draw Maps”