# Nearest foreign borders within a selected country

*Carl Schmertmann*

*12 July 2020*

## Source

This is a modified version of Arthur Welle's idea and code. The original is at

https://raw.githubusercontent.com/arthurwelle/VIS/master/closest_neighbour_git.Rmd

```
library(tidyverse)
library(sf)
library(geobr)
library(ggthemes)
```

### Data

The working directory contains shapefiles for international boundaries, downloaded from

https://www.naturalearthdata.com/downloads/50m-cultural-vectors/50m-admin-0-countries-2/

There are also custom shapefiles for German state borders in this repository. You could add internal borders for other countries as desired.

### Procedure

Select a *focal country*, get the world map, and save the polygons for the focal country plus any other countries that touch its (land) borders.

The current code works well for countries whose closest neighbors are contiguous (e.g. Brazil or Germany). It will not work for cases in which nearby countries do not "touch" the focal country (e.g. UK's neighbors, Ireland's, Iceland's, . . . ).

Improvements are still necessary to make the code universal, but the current version will work correctly for many countries.

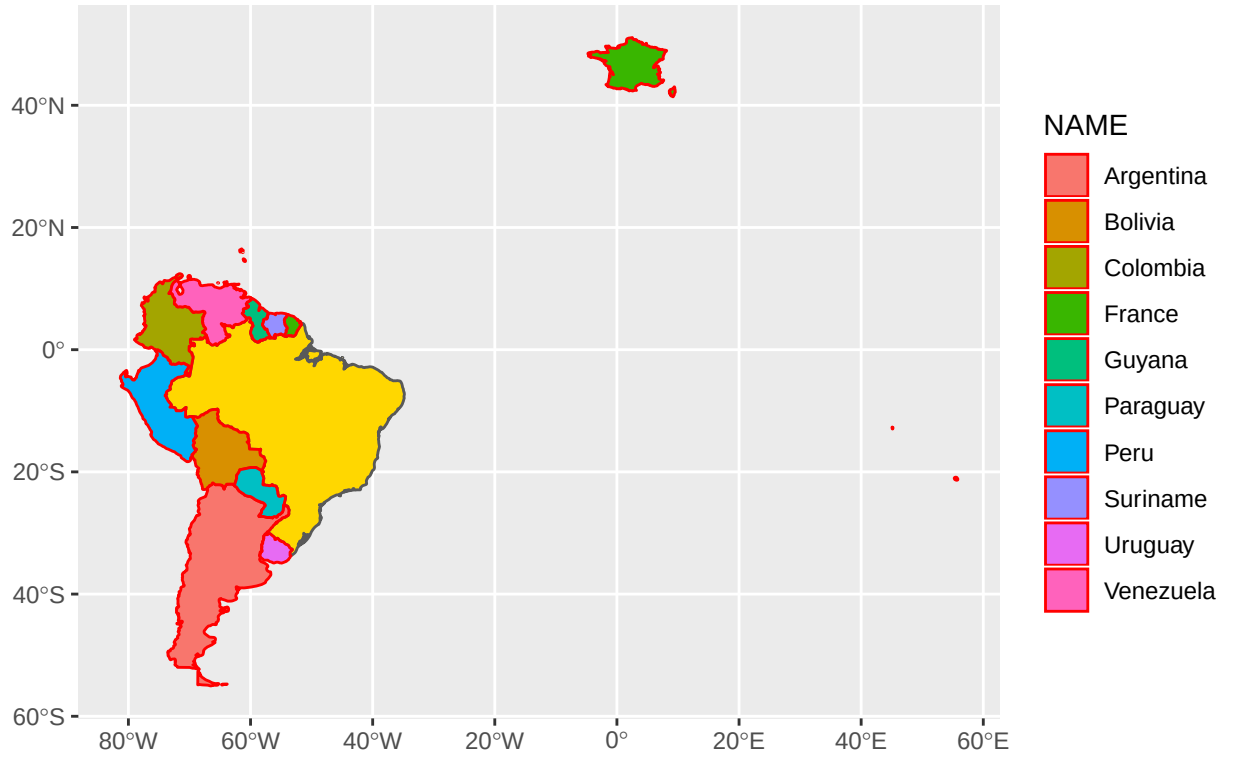
```
focal_country = 'Brazil'

world_map = st_read(dsn='.', 'ne_50m_admin_0_countries')
```

```
## Reading layer `ne_50m_admin_0_countries' from data source `C:\Users\Carl\Documents\GitHub\bonecave\n
## Simple feature collection with 241 features and 94 fields
## geometry type:  MULTIPOLYGON
## dimension:      XY
## bbox:           xmin: -180 ymin: -89.99893 xmax: 180 ymax: 83.59961
## geographic CRS: WGS 84
```

```
national_map = filter(world_map, NAME==focal_country)

touching = st_touches(world_map, national_map) %>%
            sapply(length)

neighbors = filter(world_map, touching==1)

neighbor_codes = unique(neighbors$ADM0_A3)
```

Display a map of the focal country + (contiguous) neighboring countries.

```
M = ggplot() +
      geom_sf(data=national_map, fill='gold') +
      geom_sf(data=neighbors, aes(fill=NAME), color='red')

print(M)
```




**Build a grid of tesselating hexagons over the map of the selected country. Find which neighbor is closest to each.**

```
# grid of hexagons over national map
# clip parts of any hexagons that fall outside of national map

hex_grid = st_make_grid(national_map,
              n=c(100,100),square=FALSE) %>%
          st_sf() %>%
          st_intersection(., national_map)


hex_centroids = st_centroid(hex_grid) %>%
                st_coordinates()

# the "L3" column will contain the index of the neighboring country
edges = st_coordinates(neighbors$geometry)
```

```
national_bbox = st_bbox(national_map)

keep = sapply(1:nrow(edges), function(i) {
              ((edges[i,'X'] >= national_bbox['xmin'])
            &  (edges[i,'X'] <= national_bbox['xmax'])
            &  (edges[i,'Y'] >= national_bbox['ymin'])
            &  (edges[i,'Y'] <= national_bbox['ymax']))
             })

edges = edges[keep,c('X','Y','L3')]
```

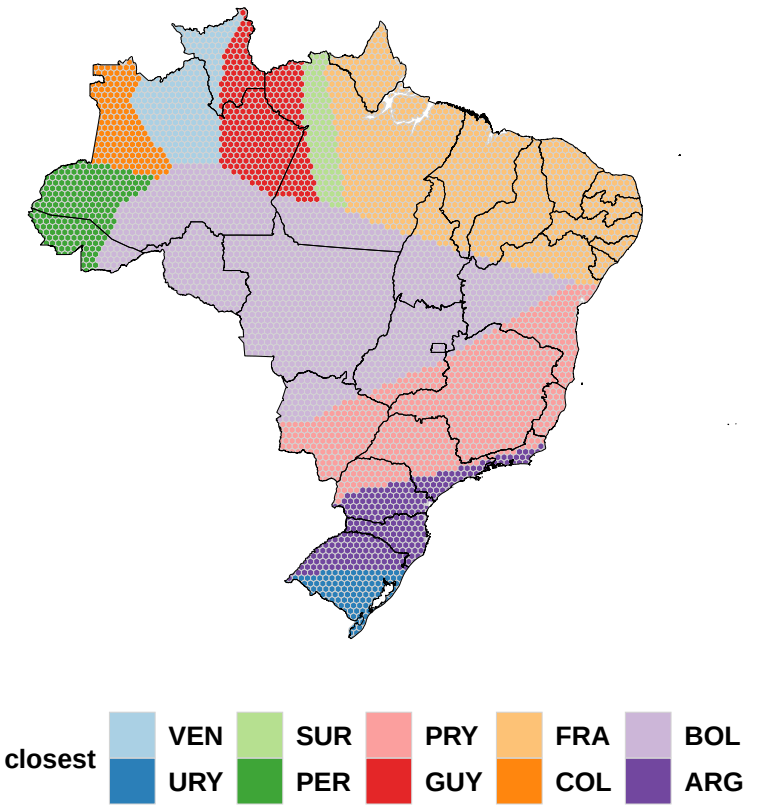
Do a brute-force check of distances: find the subset of neighbor's boundaries that are within the bounding box of the selected country. Then check the pairwise Euclidean distances between each of those points and the center of each hexagon.

```
# brute force, Euclidean
# for each hexagon in the grid
# find the country corresponding to the nearest edge point

i_closest = sapply(1:nrow(hex_centroids), function(i) {
         d = sqrt( (edges[,'X'] - hex_centroids[i,'X'])^2 +
                   (edges[,'Y'] - hex_centroids[i,'Y'])^2)
         ix = which.min(d)
         return(edges[ix,'L3'])
       })

# add a "closest" column to the hex grid
hex_grid$closest = factor(i_closest,
                          levels = seq(neighbor_codes),
                          labels = neighbor_codes)
```

Now construct the map, shading each hexagonal cell based on the closest neighbor.

```
my_palette = c('#a6cee3','#1f78b4','#b2df8a','#33a02c',
               '#fb9a99','#e31a1c','#fdbf6f','#ff7f00',
               '#cab2d6','#6a3d9a','#ffff99','#b15928')

H = ggplot() +
  geom_sf( data=hex_grid, aes(fill=closest), alpha=.95, size=.05, color='lightgrey') +
  scale_fill_manual(values=my_palette[seq(neighbor_codes)]) +
  ggthemes::theme_map() +
  theme(legend.position = 'bottom',
        legend.text = element_text(size=10,face='bold'),
        legend.title = element_text(size=10,face='bold'))


# custom addition for Brazil: add state boundaries
# Thanks, {geobr}!
if (focal_country == 'Brazil') {

    state_map = read_state('all',
                           simplified=TRUE,
                           showProgress = FALSE)

    H = H +
```

```
      geom_sf(data=state_map, color='black',
              size=.05, fill=NA)
}

# custom addition for Germany: add state boundaries
# from a downloaded shapefile
# see https://gadm.org for a catalog

if (focal_country == 'Germany') {

    state_map = readRDS('gadm36_DEU_1_sf.rds')

    H = H +
      geom_sf(data=state_map, color='black',
              size=.05, fill=NA)
}



print(H)
```




```
ggsave(filename=paste0('nearest-country-map-',
                       focal_country,'.pdf'), plot=H,
       width=11, height=8.5)
```

```
ggsave(filename=paste0('nearest-country-map-',
                       focal_country,'.png'), plot=H,
       width=11, height=8.5)
```