GIS 2: Final Project

Josh Sulkin

3/18/2022

PART 1: MILITARY POINTS

Extract

Environment setup

```
library(sf)
library(tmap)
library(terra)
```

```
## Warning: package 'terra' was built under R version 4.1.2
```

```
library(dplyr)
library(spData)
library(tidyverse)
library(tidygeocoder)
library(vtable)
library(ggmap)
library(tmaptools)
```

Bring CSV data into environment and view

After having created an Excel file with the locations of significant (recorded in secondary literature) locations of military occupations, I ensured that there was a join key, "dept," so that I could later join this data set to the Guerry data set using "dept." Thereafter, I bring in the CSV-exported document into R using "read.csv" and view the data using "head()".

```
troopPts <- read.csv("/Users/jls/GEOG28402/Final\ Project/GIS2_JLS_FinalProject_TroopPts_Attempt2.csv")
head(troopPts)
```

```
##         City                Address   Zip Country dept
## 1     Nantes       4 Pl. Marc Elder 44000  France   44
## 2   Bordeaux       16 Rue de Tivoli 33000  France   33
## 3       Lyon 25 Rue du Premier Film 69008  France   69
## 4      Lille        9 Rue Princesse 59000  France   59
## 5    Damigny     112 Rue Principale 61250  France   61
## 6 Strasbourg 4 Rue de Koenigshoffen 67000  France   67
```

Transform

Change structure and add column to prepare for geocoding

After loading in the data, I had to ensure that prepare for a smooth geocoding process, which first required me to create a new column combining the necessary location information for OpenStreetMap to geocode the data. The new column, "full add" contains the address, city, zip code, and department number, all of which became characters using "as.character":

```
troopPts$fullAdd <- paste(as.character(troopPts$Address),
                          as.character(troopPts$City),
                          as.character(troopPts$Zip),
                          as.character(troopPts$Country))
```

Geocode points

For the more exciting part, I used geocode_OSM() since my data are not from the United States (so geocode() would not have worked since it uses US Census Bureau data), specifying the Coordinate Reference System as EPSG:27572 (NTF (Paris) / Lambert zone II) because this projection limits the distortion for France.

```
geocodedPts <- geocode_OSM(troopPts$fullAdd,
                           projection = 27572,
                           return.first.only = TRUE,
                           keep.unfound = FALSE,
                           details = FALSE,
                           as.sf = FALSE,
                           geometry = c("point", "bbox"),
                           server = "https://nominatim.openstreetmap.org")
```

```
## No results found for "2 Rue Paul Gauguin Loos-en-Gohelle 62750 France".
```

```
## No results found for "115 Rue Raoul Briquet Vieux-Conde 62710 France".
```

Convert geocoded points to spatial data

Since the geocoded data still do are not spatially enabled, I enabled them using the st_as_sf() function and viewed the data to ensure they were properly enabled spatially.

```
countMilitary <- st_as_sf(geocodedPts, coords = c("x", "y"), crs = 27572)
head(data.frame(countMilitary))
```

```
##                                         query   y_min   y_max    x_min
## 1           4 Pl. Marc Elder Nantes 44000 France 2253460 2253619 305722.3
## 2         16 Rue de Tivoli Bordeaux 33000 France 1987990 1988002 368690.6
## 3       25 Rue du Premier Film Lyon 69008 France 2085911 2085922 797158.3
## 4             9 Rue Princesse Lille 59000 France 2628203 2628214 651194.4
## 5        112 Rue Principale Damigny 61250 France 2386185 2386434 432389.9
## 6 4 Rue de Koenigshoffen Strasbourg 67000 France 2411587 2411598 997710.2
##      x_max                 geometry
## 1 305900.9 POINT (305746.7 2253538)
```

```
## 2 368698.9 POINT (368694.8 1987996)
## 3 797166.4 POINT (797162.3 2085917)
## 4 651201.6   POINT (651198 2628209)
## 5 432516.0 POINT (432488.1 2386325)
## 6 997718.4 POINT (997714.3 2411593)
```

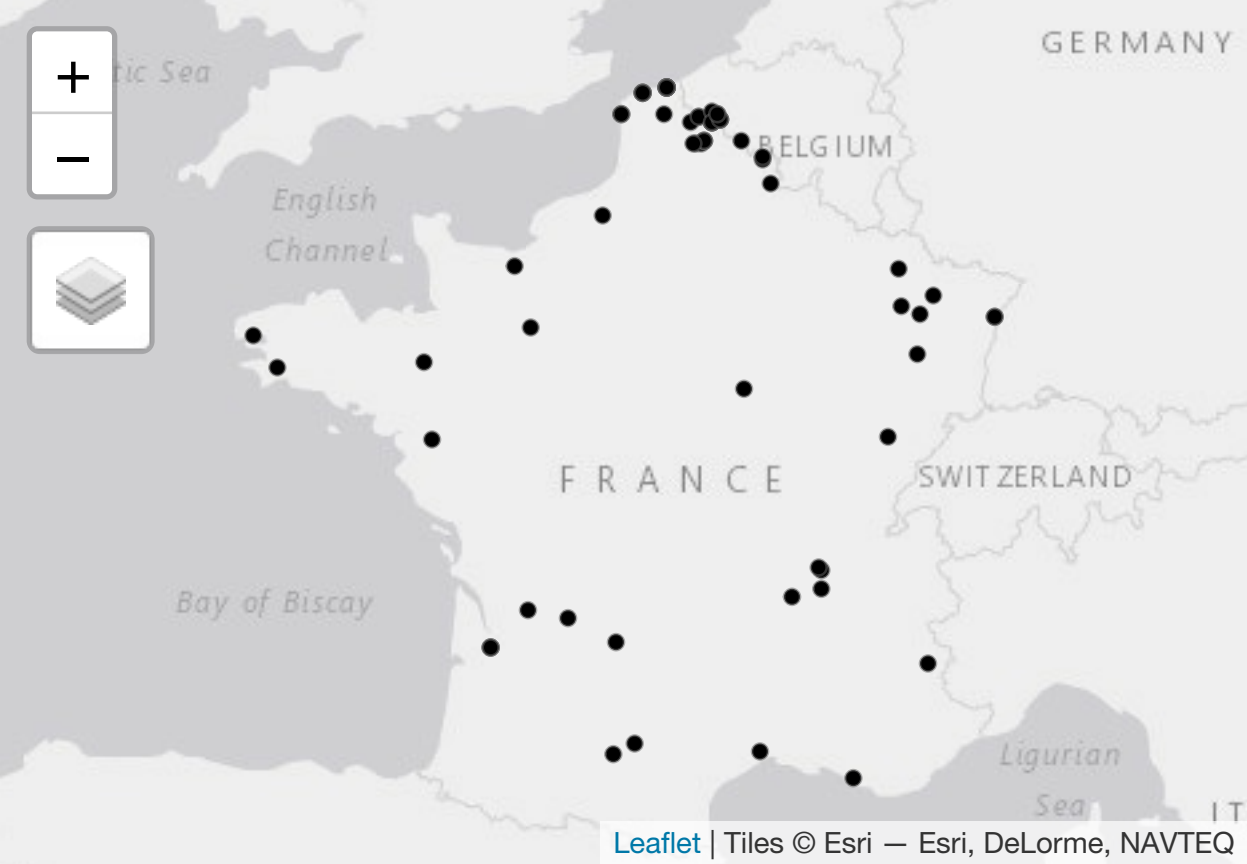
Visualize points

To visualize the points, I used a combination of tmap() functions, ensuring that the points would overlay the interactive OSM base map.

```
tmap_mode("view")
```

```
## tmap mode set to interactive viewing
```

```
Military_Map <- tm_shape(countMilitary) + tm_dots()
Military_Map
```



Create a summary statistics table

To detail a statistical summary, I used sumtable() to document the number, mean, standard deviation, minimum, first quartile, third quartile, and maximum for each variable in the countMilitary data set.

```
summary(countMilitary)
```

```
##     query               y_min               y_max               x_min
##  Length:74          Min.   :1814890   Min.   :1814910   Min.   : 93867
##  Class :character   1st Qu.:2351504   1st Qu.:2351657   1st Qu.:571184
##  Mode  :character   Median :2605721   Median :2605735   Median :643248
##                     Mean   :2447051   Mean   :2447136   Mean   :631164
##                     3rd Qu.:2634075   3rd Qu.:2634126   3rd Qu.:691650
##                     Max.   :2669936   Max.   :2669947   Max.   :997710
##      x_max                 geometry
##  Min.   : 93876   POINT        :74
##  1st Qu.:571334   epsg:27572   : 0
##  Median :643270   +proj=lcc ...: 0
##  Mean   :631246
##  3rd Qu.:691895
##  Max.   :997718
```

Load

To use these data in the next steps of the analysis, I saved the data using write_sf().

```
write_sf(countMilitary, "/Users/jls/GEOG28402/Final\ Project/GIS2_JLS_FinalProject_countMilitary.shp")
```

PART 2: THE OTHER VARIABLES: LITERACY, WEALTH, AND DISTANCE

Extract

Environment Set-Up

It is not necessary to call-in libraries again since they were called in Part 1.

Load-in data

To load the literacy, wealth, and distance data, I will read in the Guerry shapefile using st_read().

```
FR_Guerry = st_read("/Users/jls/GEOG28402/Final\ Project/guerry/guerry.shp")
```

```
## Reading layer ‘guerry’ from data source
##   ‘/Users/jls/GEOG28402/Final Project/guerry/guerry.shp’ using driver ‘ESRI Shapefile’
## Simple feature collection with 85 features and 23 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: 47680 ymin: 1703258 xmax: 1031401 ymax: 2677441
## Projected CRS: NTF (Paris) / Lambert zone II
```

Transform

Inspect data

I will now inspect the data using head() and st_crs() to investigate the CRS, structure, and other elements of the data.

```
st_crs(FR_Guerry)
```

```
## Coordinate Reference System:
##   User input: NTF (Paris) / Lambert zone II
##   wkt:
## PROJCRS["NTF (Paris) / Lambert zone II",
##     BASEGEOGCRS["NTF (Paris)",
##         DATUM["Nouvelle Triangulation Francaise (Paris)",
##             ELLIPSOID["Clarke 1880 (IGN)",6378249.2,293.466021293627,
##                 LENGTHUNIT["metre",1]]],
##         PRIMEM["Paris",2.5969213,
##             ANGLEUNIT["grad",0.0157079632679489]],
##         ID["EPSG",4807]],
##     CONVERSION["Lambert zone II",
##         METHOD["Lambert Conic Conformal (1SP)",
##             ID["EPSG",9801]],
##         PARAMETER["Latitude of natural origin",52,
##             ANGLEUNIT["grad",0.0157079632679489],
##             ID["EPSG",8801]],
##         PARAMETER["Longitude of natural origin",0,
##             ANGLEUNIT["grad",0.0157079632679489],
##             ID["EPSG",8802]],
##         PARAMETER["Scale factor at natural origin",0.99987742,
##             SCALEUNIT["unity",1],
##             ID["EPSG",8805]],
##         PARAMETER["False easting",600000,
##             LENGTHUNIT["metre",1],
##             ID["EPSG",8806]],
##         PARAMETER["False northing",2200000,
##             LENGTHUNIT["metre",1],
##             ID["EPSG",8807]]],
##     CS[Cartesian,2],
##         AXIS["easting (X)",east,
##             ORDER[1],
##             LENGTHUNIT["metre",1]],
##         AXIS["northing (Y)",north,
##             ORDER[2],
##             LENGTHUNIT["metre",1]],
##     USAGE[
##         SCOPE["Engineering survey, topographic mapping."],
##         AREA["France mainland onshore between 50.5 grads and 53.5 grads North (45°27'N to 48°09'N). A
##         BBOX[42.33,-4.87,51.14,8.23]],
##     ID["EPSG",27572]]
```

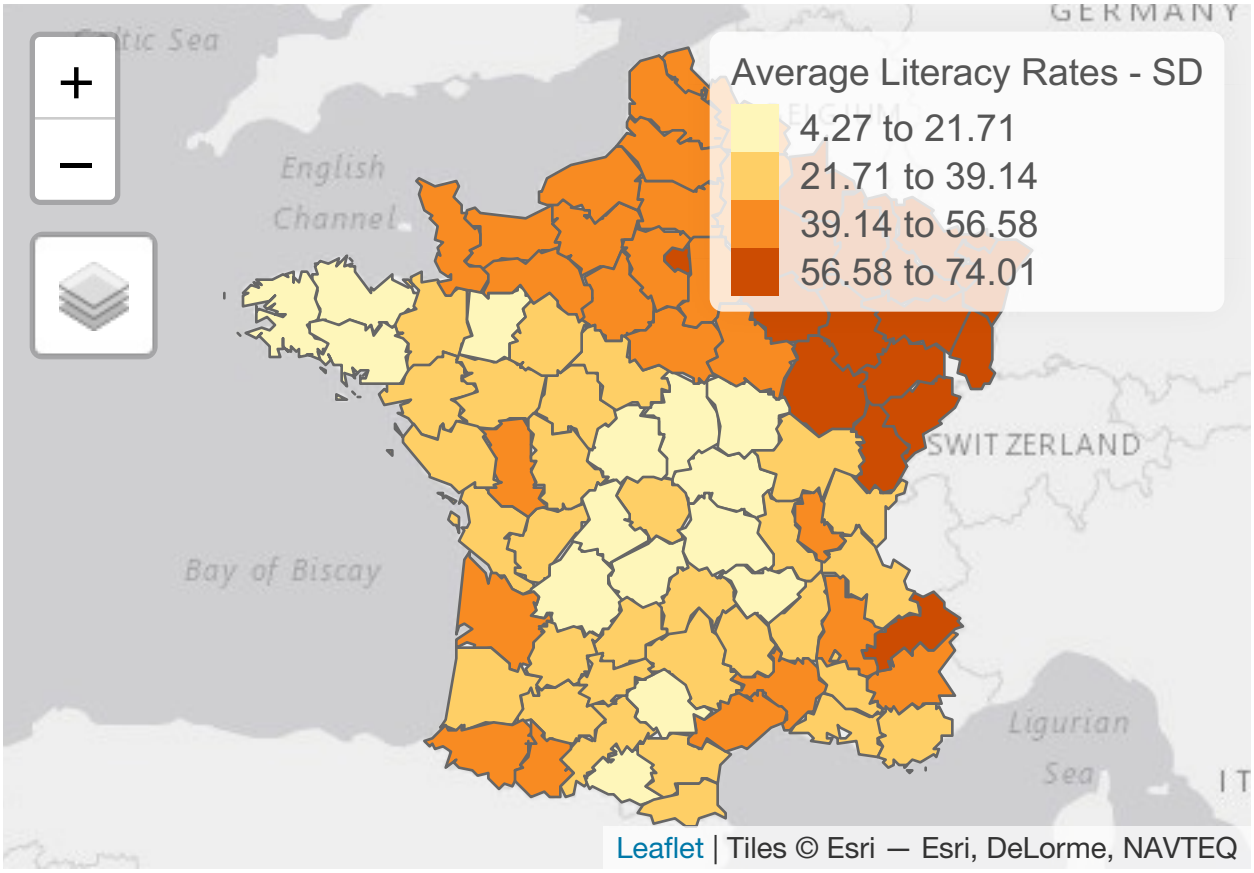
```
head(FR_Guerry)
```

```
## Simple feature collection with 6 features and 23 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: 595532 ymin: 1858801 xmax: 975716 ymax: 2564568
## Projected CRS: NTF (Paris) / Lambert zone II
##   dept Region      Dprtmnt Crm_prs Crm_prp Litercy Donatns Infants Suicids
## 1    1      E          Ain   28870   15890      37    5098   33120   35039
## 2    2      N        Aisne   26226    5521      51    8901   14572   12831
## 3    3      C       Allier   26747    7925      13   10973   17044  114121
## 4    4      E Basses-Alpes   12935    7289      46    2733   23018   14238
## 5    5      E Hautes-Alpes   17488    8174      69    6962   23076   16171
## 6    7      S      Ardeche    9474   10263      27    3188   42117   52547
##   MainCty Wealth Commerc Clergy Crm_prn Infntcd Dntn_cl Lottery Desertn Instrct
## 1       2     73      58     11      71      60      69      41      55      46
## 2       2     22      10     82       4      82      36      38      82      24
## 3       2     61      66     68      46      42      76      66      16      85
## 4       1     76      49      5      70      12      37      80      32      29
## 5       1     83      65     10      22      23      64      79      35       7
## 6       1     84       1     28      76      47      67      70      19      62
##   Prsttts Distanc Area Pop1831                        geometry
## 1      13 218.372 5762  346.03 MULTIPOLYGON (((801150 2092...
## 2     327  65.945 7369  513.00 MULTIPOLYGON (((729326 2521...
## 3      34 161.927 7340  298.26 MULTIPOLYGON (((710830 2137...
## 4       2 351.399 6925  155.90 MULTIPOLYGON (((882701 1920...
## 5       1 320.280 5549  129.10 MULTIPOLYGON (((886504 1922...
## 6       1 279.413 5529  340.73 MULTIPOLYGON (((747008 1925...
```

Literacy Choropleth Map

Beginning with the literacy variable, I will create a choropleth map by using tm_shape() to specify the dataset. Then I use tm_fill() to specify the column R will pull data from– the literacy ("litercy") column– the title of the key on the right side, the style (which in this case uses standard deviation because it shows the most significant variation that captures the relatively high literacy rates of Paris and the northeast), and other details for the map if I could have figured how to display them. I then named all of that as "Literacy_Map" so that I could then call "Literacy_Map" at the very end and for ease of use. Finally, I create a summary statistics table using summary().

```
Literacy_Map <- tm_shape(FR_Guerry) +
  tm_fill('Litercy',
          title = 'Average Literacy Rates - SD',
          style = 'sd',
          n = 5) +
  tm_borders() +
  tm_layout(frame = FALSE, legend.outside = TRUE,
            legend.outside.position = 'right', legend.title.size =0.9,
            main.title = 'Average Literacy Rate, by Department, 1830, by Josh Sulkin',
            main.title.size = 0.9)
Literacy_Map
```




```
summary(FR_Guerry$Litercy)
```

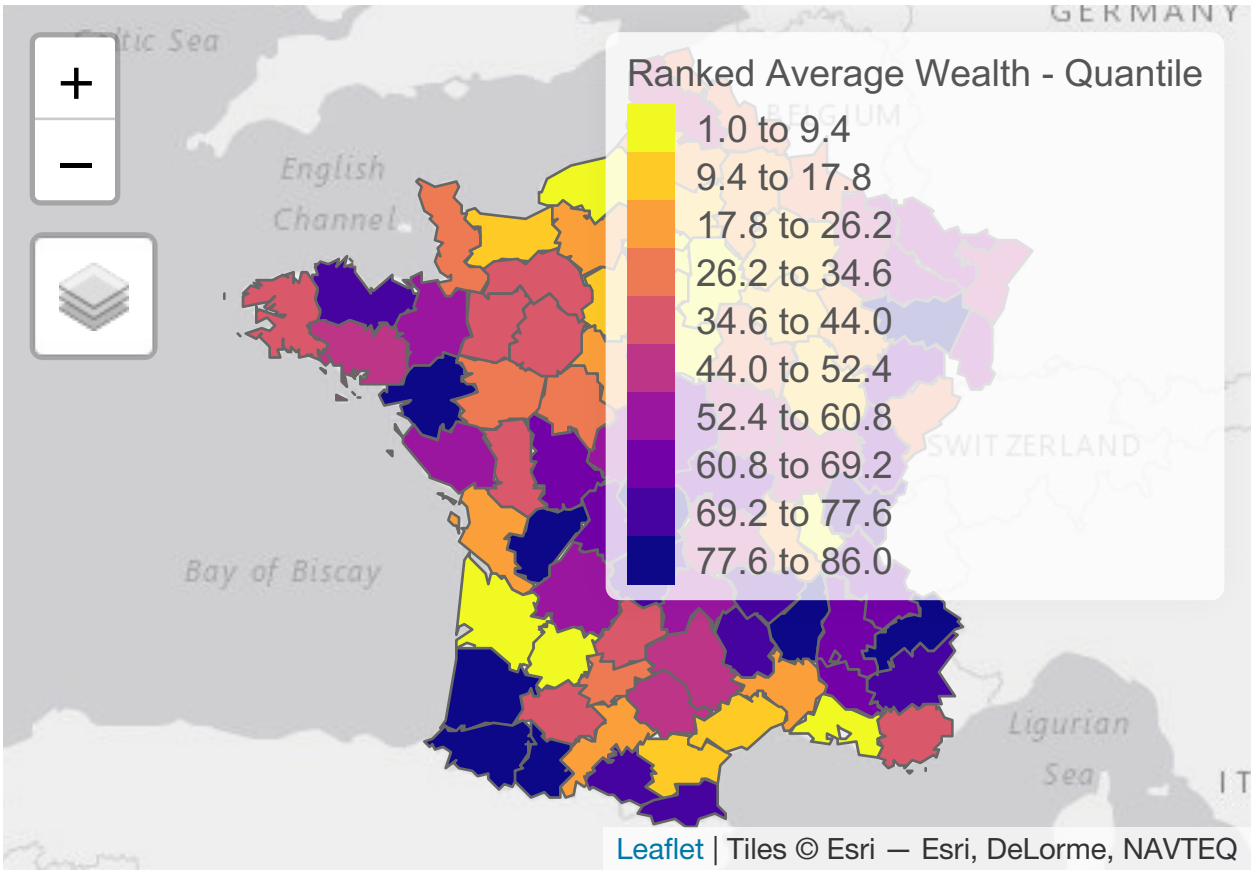
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   12.00   25.00   38.00   39.14   52.00   74.00
```

Wealth Choropleth Map

Turning to the literacy variable, I will create a choropleth map by using tm_shape() to specify the dataset. Then I use tm_fill() to specify the column R will pull data from– the wealth ("wealth") column– the title of the key on the right side, the style (which in this case uses the "quantile" because quantiles work best for ordinal data), the colors (which I inverted to better show how higher numbers mean more wealth using lighter colors), and other details for the map if I could have figured how to display them. I then named all of that as "Wealth_Map" so that I could then call "Wealth_Map" at the very end and for ease of use. As a note, Guerry ranked the departments from most wealthy to least wealthy, with the most wealthy areas assigned smaller numbers and poorer areas assigned the larger numbers.I do not create a statistics table for this variable because it would not make sense to find them for ordinal data.

```
Wealth_Map <- tm_shape(FR_Guerry) +
  tm_fill('Wealth',
          title = 'Ranked Average Wealth - Quantile',
          style = 'quantile',
          n = 10) +
  tm_borders() +
  tm_layout(frame = FALSE,
```

```
            legend.outside = TRUE,
            legend.outside.position = 'right',
            legend.title.size = 0.9,
            main.title = 'Ranked Average of Wealth, by Department, 1830, by Josh Sulkin',
            main.title.size = 0.9,
            aes.palette = list(seq = "-plasma"))
Wealth_Map
```

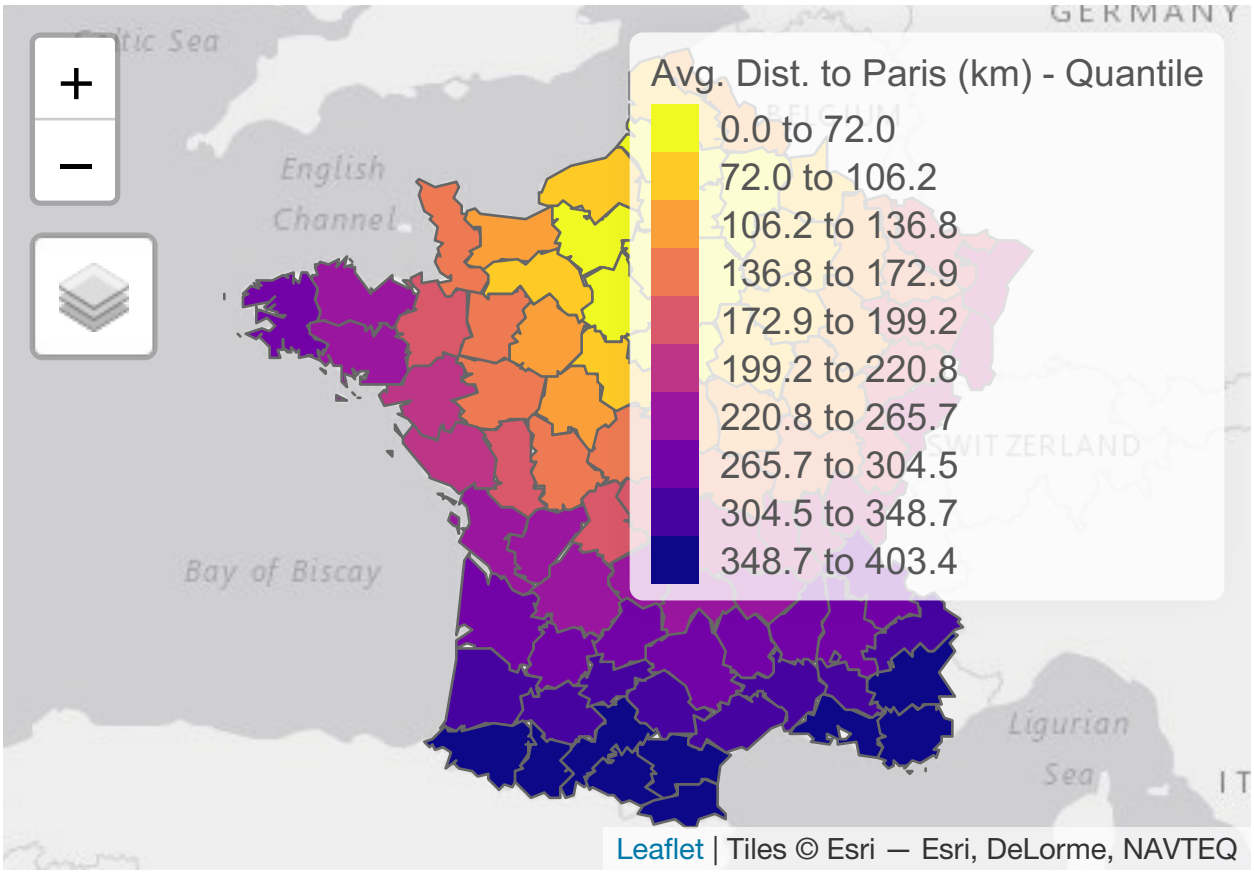


Distance to Paris Choropleth Map

Culminating with the distance to Paris variable, I will create a choropleth map by using tm_shape() to specify the dataset. Then I use tm_fill() to specify the column R will pull data from– the wealth ("distanc") column– the title of the key on the right side, the style (which in this case uses quantile because it shows the gradual, equal variation of distance), the colors (which I inverted to better show how the closer one is to Paris, the brighter the color is), and other details for the map if I could have figured how to display them. I then named all of that as "Distance_Map" so that I could then call "Distance_Map" at the very end and for ease of use. I do not create a statistics table for this variable because it would not make sense to find them for average distances to Paris.

```
Distance_Map <- tm_shape(FR_Guerry) +
  tm_fill('Distanc',
          title = 'Avg. Dist. to Paris (km) - Quantile',
          style = 'quantile',
          n = 10) +
```

```
  tm_borders() +
  tm_layout(frame = FALSE, legend.outside = TRUE,
            legend.outside.position = 'right',
            legend.title.size = 0.9,
            main.title = 'Average Absolute Distance from Department
            Centroid to Paris, by Josh Sulkin',
            main.title.size = 0.9,
            aes.palette = list(seq = "-plasma"))
Distance_Map
```



Load

Since I did not manipulate the data from the Guerry shapefile (I failed to join the military data set with the Guerry set), I will not save it again as a shapefile and instead will export each of the maps as .jpeg files.