Mapping

Guy J. Abel and Nayoung Heo

Background

- The `maps` package allowed R users to produce simple maps in R for a long time (first released 2003)
  - Relatively simple mechanism for making outlines and plotting lat-long points and paths on them.
- More recent packages like `sp`, `rgdal`, `raster` and `rgeos` have allowed R users to create more complex maps, acquiring much of the functionality of traditional GIS packages (e.g. ArcGIS).
  - Not always easily accessible for the beginners
  - Requires familiarity with GIS concepts.
  - Will use some basic functions to download shape files and load data.
- A third approach uses `ggplot`
  - Will use this approach to create map plots.
  - Add tiling of detailed base maps from Google Earth, Open Street Maps,. . . using the `ggmap` package.
- There are many other packages to help with spatial data in R.

Shape Files

- The shape-file format is a popular geo-spatial vector data format for geographic information system
- Spatially describes vector features: points, lines, and polygons, representing, for example, water wells, rivers, and regions.
- Each item usually has attributes that describe it, such as name or temperature.
- The term "shape-file" is slightly misleading. The format consists of a collection of files with a common file name prefix, stored in the same directory.
- Three mandatory files:
  - `.shp`: the feature geometry itself
  - `.shx`: positional index of the feature geometry
  - `.dbf`: attributes for each shape

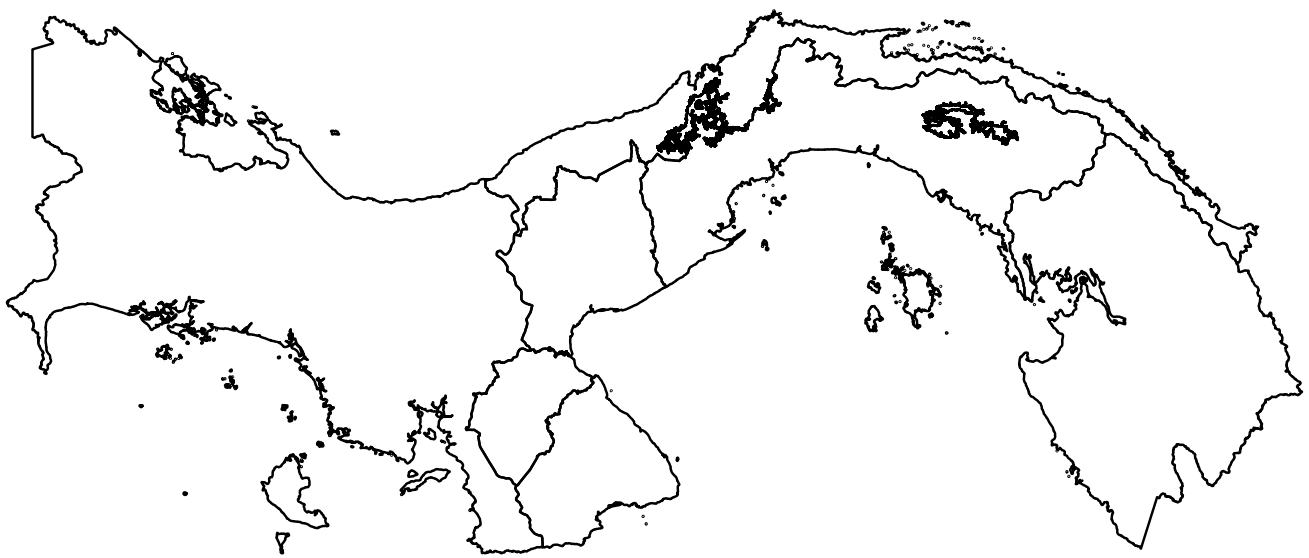
Loading Shape Files

- Data agencies typically provide shape files where there is a demand and resources.
  - IPUMSI provide shape files to match their harmonized geographies.
  - `https://international.ipums.org/international/geography_gis`
  - Example shape files for Panama

```
> list.files("./data/geo1_pa1960_2010")
[1] "geo1_pa1960_2010.cpg"     "geo1_pa1960_2010.dbf"
[3] "geo1_pa1960_2010.prj"     "geo1_pa1960_2010.sbn"
[5] "geo1_pa1960_2010.sbx"     "geo1_pa1960_2010.shp"
[7] "geo1_pa1960_2010.shp.xml" "geo1_pa1960_2010.shx"
```

Loading Shape Files

```
> # install.packages("rgdal")
> library(rgdal)
> pan1 <- readOGR("./data/geo1_pa1960_2010/geo1_pa1960_2010.shp")
OGR data source with driver: ESRI Shapefile
Source: "./data/geo1_pa1960_2010/geo1_pa1960_2010.shp", layer: "geo1_pa1960_2010"
with 8 features
It has 5 fields
> # basic plot
> par(mar = rep(0, 4))
> plot(pan1)
```



Viewing Shape Files

```
> # view the structure of the shape file
> str(pan1, max.level = 2)
Formal class 'SpatialPolygonsDataFrame' [package "sp"] with 5 slots
  ..@ data       :'data.frame': 8 obs. of  5 variables:
  ..@ polygons   :List of 8
  ..@ plotOrder  : int [1:8] 1 3 5 6 2 4 8 7
  ..@ bbox       : num [1:2, 1:2] -83.05 7.2 -77.16 9.65
  .. ..- attr(*, "dimnames")=List of 2
  ..@ proj4string:Formal class 'CRS' [package "sp"] with 1 slot
```

Viewing Shape Files

```
> # view the data slot
> pan1@data
  CNTRY_NAME                                                ADMIN_NAME
0     Panama Bocas de Toro, Chiriquí, Comarca Ngäbe Buglé, Veraguas
1     Panama                                                   Coclé
2     Panama                                  Comarca Emberá, Darién
3     Panama                                               Los Santos
4     Panama                                                   Panamá
5     Panama                        Colón, Comarca Kuna Yala (San Blas)
6     Panama                                                Waterbody
7     Panama                                                  Herrera
  CNTRY_CODE GEOLEVEL1 PARENT
0        591    591004    591
1        591    591002    591
2        591    591005    591
3        591    591007    591
4        591    591008    591
5        591    591003    591
6        591    888888    591
7        591    591006    591
```

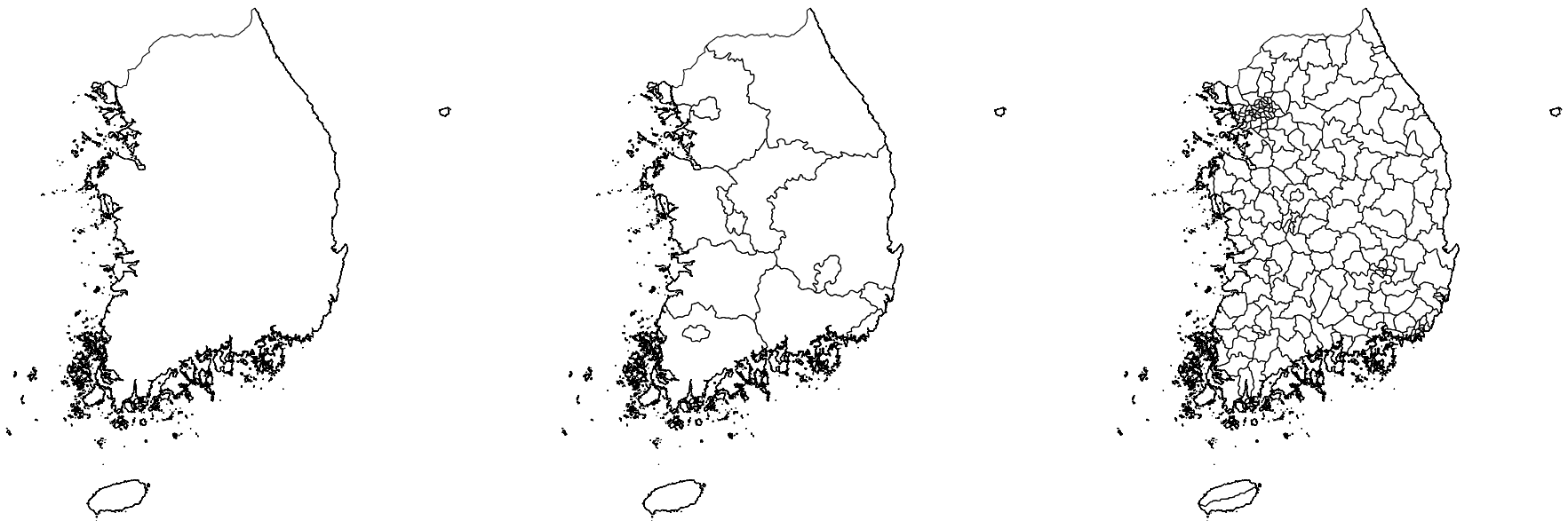
Downloading GADM Shape Files

- There are also resources in R to download shape files, such as administrative data from GADM
  - `http://gadm.org/`
  - Download multiple administrative layers straight from R using the `getData()` function in the `raster` package.
  - Requires the 3 letter ISO code, that can be viewed using `getData('ISO3')`

```
> # install.packages("raster")
> library(raster)
> getData('ISO3')
    ISO3                                  NAME
1    ABW                                 Aruba
2    AFG                           Afghanistan
3    AGO                                Angola
4    AIA                              Anguilla
5    ALA                                 Åland
6    ALB                               Albania
7    AND                               Andorra
8    ARE                  United Arab Emirates
9    ARG                             Argentina
10   ARM                               Armenia
11   ASM                        American Samoa
12   ATA                            Antarctica
```

Downloading GADM Shape Files

```
> kor0 <- getData(name = "GADM", country = "KOR", level = 0)
> kor1 <- getData(name = "GADM", country = "KOR", level = 1)
> kor2 <- getData(name = "GADM", country = "KOR", level = 2)
> par(mar = rep(0, 4))
> plot(kor0); plot(kor1); plot(kor2)
```



Viewing GADM Shape Files

```
> # view the structure of the shape file
> str(kor1, max.level = 2)
Formal class 'SpatialPolygonsDataFrame' [package "sp"] with 5 slots
  ..@ data       :'data.frame': 17 obs. of  13 variables:
  ..@ polygons   :List of 17
  ..@ plotOrder  : int [1:17] 9 6 14 8 10 13 3 2 12 11 ...
  ..@ bbox       : num [1:2, 1:2] 125.1 33.1 130.9 38.6
  .. ..- attr(*, "dimnames")=List of 2
  ..@ proj4string:Formal class 'CRS' [package "sp"] with 1 slot
```

Viewing GADM Shape Files

```
> # view the data slot
> kor1@data
   OBJECTID ID_0 ISO      NAME_0 ID_1           NAME_1 HASC_1 CCN_1 CCA_1
1         1  213 KOR South Korea    1            Busan  KR.PU    NA
2         2  213 KOR South Korea    2 Chungcheongbuk-do  KR.GB    NA
3         3  213 KOR South Korea    3 Chungcheongnam-do  KR.GN    NA
4         4  213 KOR South Korea    4            Daegu  KR.TG    NA
5         5  213 KOR South Korea    5          Daejeon  KR.TJ    NA
6         6  213 KOR South Korea    6       Gangwon-do  KR.KW    NA
7         7  213 KOR South Korea    7           Gwangju  KR.KJ    NA
8         8  213 KOR South Korea    8      Gyeonggi-do  KR.KG    NA
9         9  213 KOR South Korea    9 Gyeongsangbuk-do  KR.KB    NA
10       10  213 KOR South Korea   10 Gyeongsangnam-do  KR.KN    NA
11       11  213 KOR South Korea   11          Incheon  KR.IN    NA
12       12  213 KOR South Korea   12             Jeju  KR.CJ    NA
13       13  213 KOR South Korea   13     Jeollabuk-do  KR.CB    NA
14       14  213 KOR South Korea   14     Jeollanam-do  KR.CN    NA
15       15  213 KOR South Korea   15           Sejong  KR.SJ    NA
16       16  213 KOR South Korea   16            Seoul  KR.SO    NA
17       17  213 KOR South Korea   17            Ulsan  KR.UL    NA
            TYPE_1                ENGTYPE_1
1       Gwangyeoksi Metropolitan City
2               Do         Province
3               Do         Province
```

Plotting Shape Files

- Using ggplot2 to plot the shape files allows for easier customization and the ability to add geoms.
- Data imported from shape files are SpatialPolygonsDataFrame objects which are not suitable for the ggplot() function.
- Need to run the data through the tidy() function in the broom package. (Note: Previous standard way was to use the fortify() function in ggplot2)

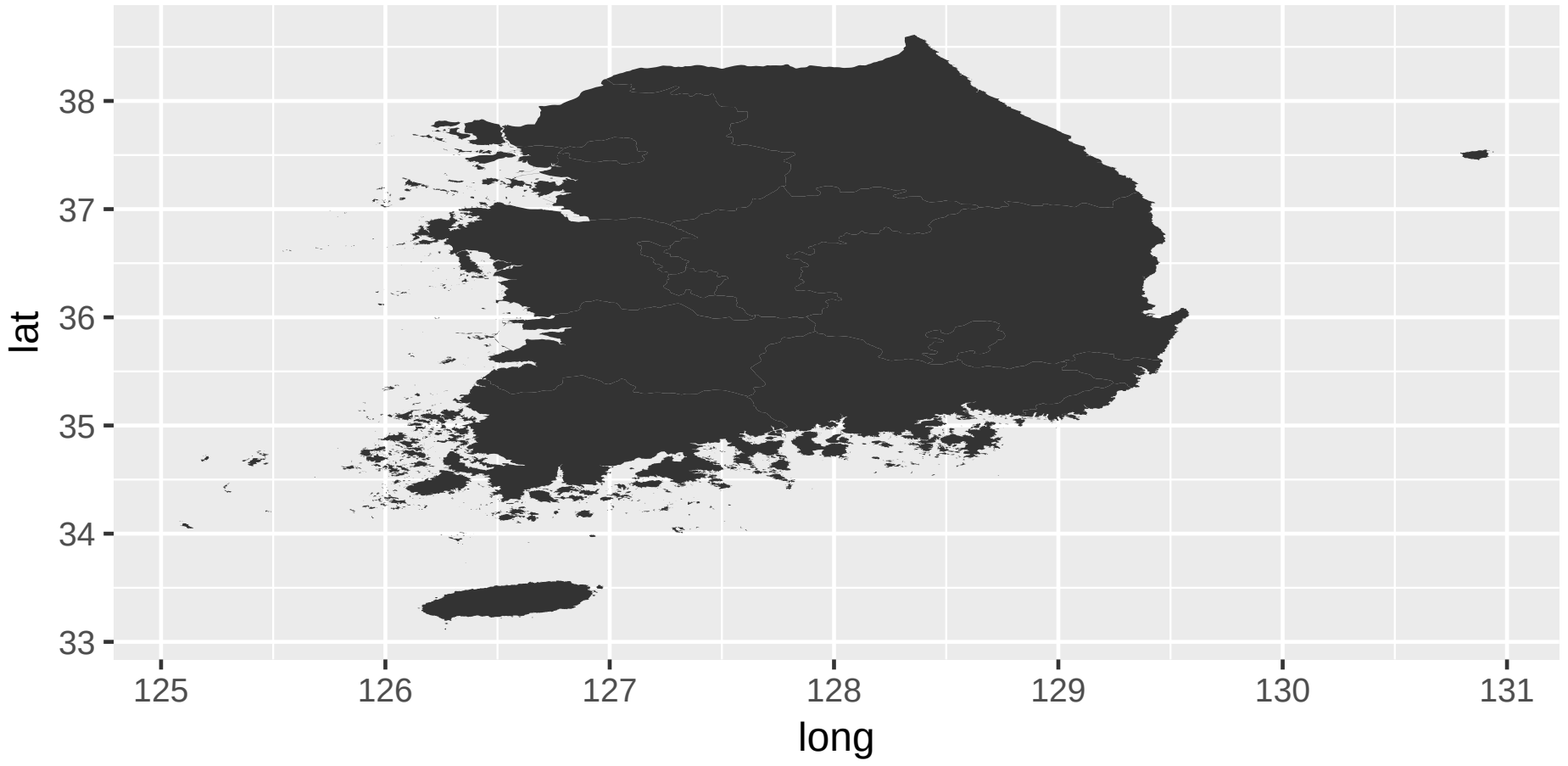
```
> library(tidyverse)
> library(broom)
> sd0 <- tidy(kor1)
> head(sd0)
      long      lat order  hole piece group id
1 129.2997 35.38611     1 FALSE     1   1.1  1
2 129.3084 35.37564     2 FALSE     1   1.1  1
3 129.3077 35.36646     3 FALSE     1   1.1  1
4 129.3139 35.36151     4 FALSE     1   1.1  1
5 129.3199 35.35071     5 FALSE     1   1.1  1
6 129.3249 35.34599     6 FALSE     1   1.1  1
```

Plotting Shape Files

- Use the `group` argument with `geom_polygon()` to identify regions
- The `coord_map()` can be used to set the projection and for zooming.
  - Default `"mercator"`.
  - Many more choices, see `?mapproject`
  - Some alternative projections require extra parameters
- Zoom into particular coordinates using `xlim` and `ylim` arguments in `coord_map()`
  - Setting limits elsewhere, such as with `xlim()` and `ylim()` functions may clip polygons and hence distort the map.
- Pass data to `fill` to `geom_polygon()` to show different data

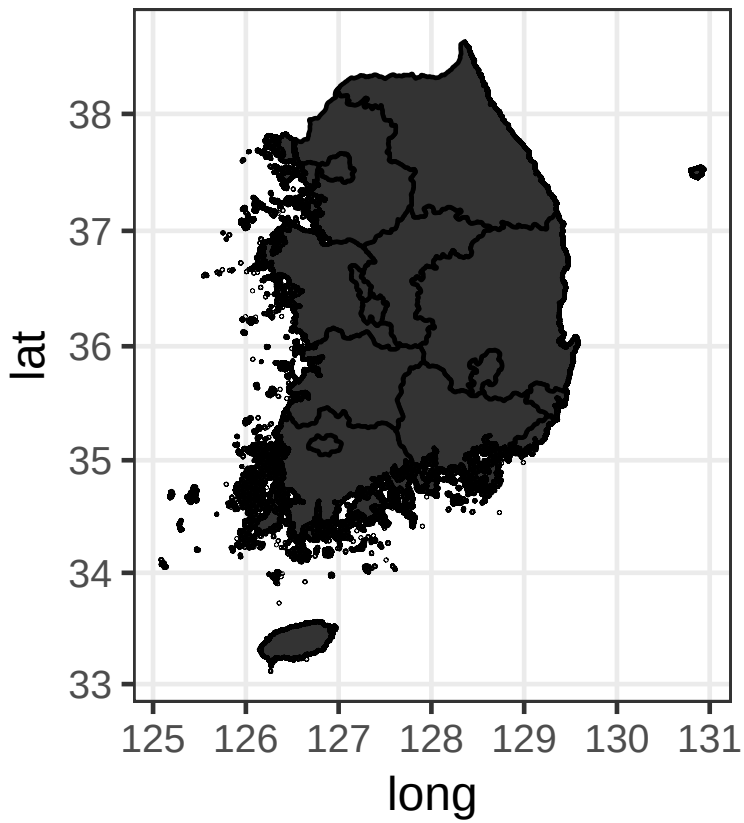
Plotting Shape Files

```
> # default plot
> ggplot(data = sd0, mapping = aes(x = long, y = lat, group = group))  +
+   geom_polygon()
```



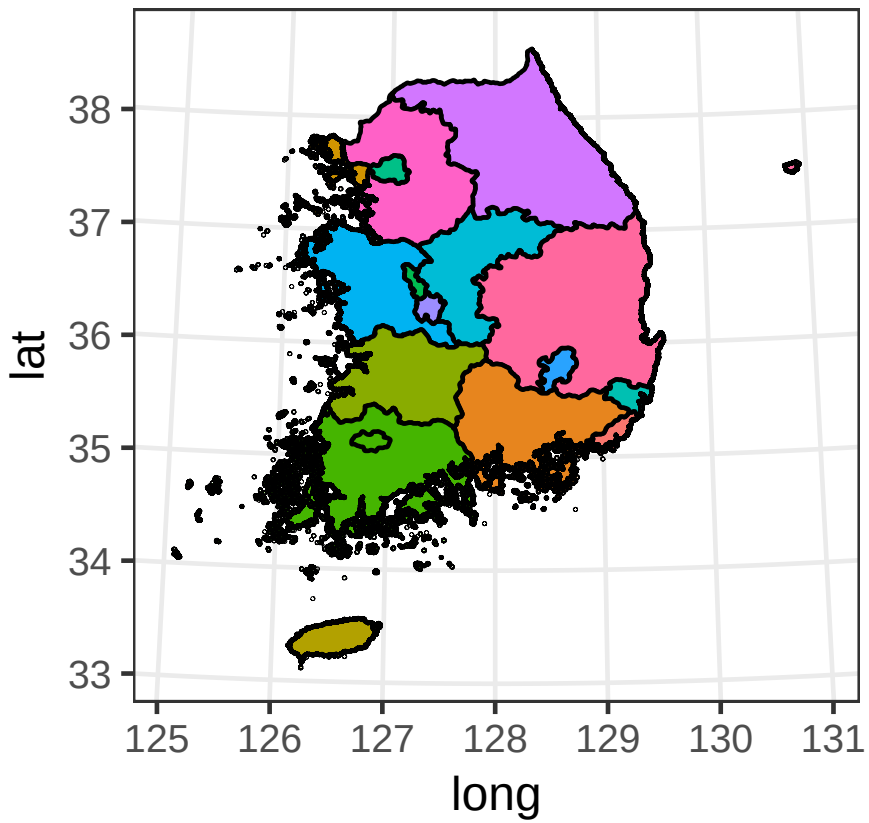
Fixing Coordinates

```
> # setting border colour to black, use coord_map() to avoid streching to plot windo
> ggplot(data = sd0, mapping = aes(x = long, y = lat, group = group))  +
+   geom_polygon(colour = "black") +
+   coord_map() +
+   theme_bw()
```



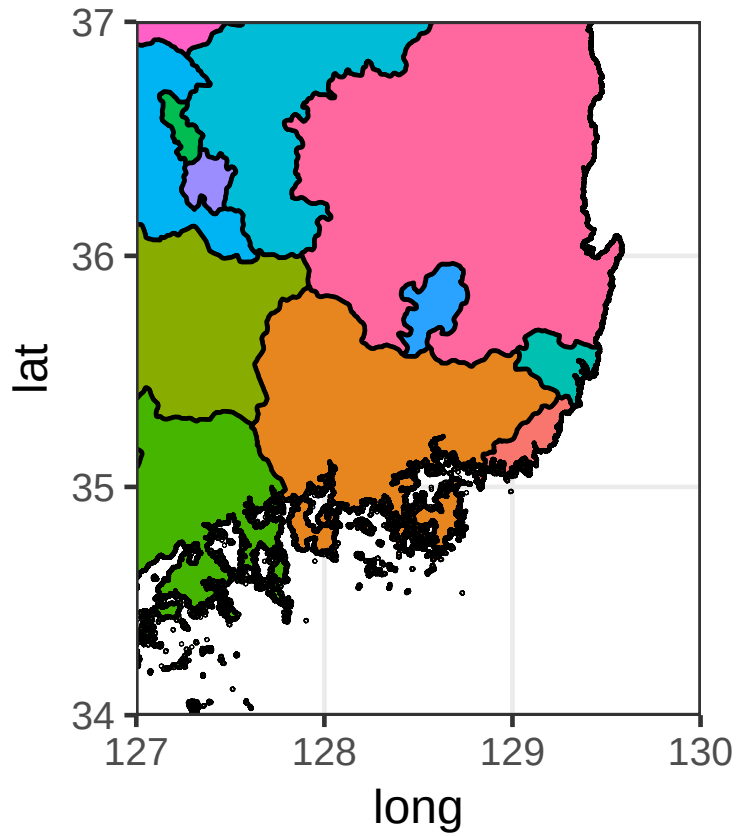
Fixing Coordinates

```
> # fill regions based on the id number, azequidistant projection
> ggplot(data = sd0, mapping = aes(x = long, y = lat, group = group, fill = id))  +
+   geom_polygon(colour = "black") +
+   coord_map(projection = "azequidistant") +
+   theme_bw() +
+   guides(fill = FALSE)
```



Zooming

```
> # zoom in using xlim and ylim within coord function, use id column for fill
> ggplot(data = sd0, mapping = aes(x = long, y = lat, group = group, fill = id))  +
+   geom_polygon(colour = "black") +
+   coord_map(xlim = c(127, 130), ylim = c(37, 34)) +
+   theme_bw() +
+   guides(fill = FALSE)
```



Exercise 1 (ex61.R)

```
# 0. a) set your working directory to the course folder
#####(dir = "#####")
```

Adding Data

- In order to map results we usually need to do some joins. Typically we have
  - Tidied spatial data (e.g. `sd1` below) with an `id` column for each polygon
  - Regional data (e.g. `df1` below) with some geocode or name for each region
- Can use the `@data` slot to join the two data frames once we are sure the columns that represent
  - The polygon `id` variable matching what is in the tidied spatial data. Usually starts from 0 or 1. Sometimes hidden in the `rownames`
  - A geocode or name matching that of our regional data. Must be in the same format (e.g. `as.integer()`) as in the tidied spatial data. Formatted as `md1` below.
- Use `left_join()` in the `dplyr` package twice
  1. Add the id column from the merging data (`md1`) to the regional data (`df1`)
  2. Add the regional data (`df1`) to the tidied shape file (`sd1`) using the `id` column.

Tidy Spatial Data

```
> pan1 <- pan1[pan1$GEOLEVEL1 != 888888, ]
>
> # tidy the panama data and turn into tibble
> sd1 <- tidy(pan1) %>%
+   tbl_df()
> sd1
# A tibble: 87,087 x 7
        long      lat order  hole  piece  group     id
       <dbl>    <dbl> <int> <lgl> <fctr> <fctr> <chr>
 1 -82.37736 9.255870     1 FALSE      1    0.1     0
 2 -82.37522 9.255750     2 FALSE      1    0.1     0
 3 -82.37239 9.252584     3 FALSE      1    0.1     0
 4 -82.37016 9.252500     4 FALSE      1    0.1     0
 5 -82.36758 9.250944     5 FALSE      1    0.1     0
 6 -82.36359 9.250500     6 FALSE      1    0.1     0
 7 -82.36203 9.249361     7 FALSE      1    0.1     0
 8 -82.36193 9.248474     8 FALSE      1    0.1     0
 9 -82.36178 9.247194     9 FALSE      1    0.1     0
10 -82.36314 9.245475    10 FALSE      1    0.1     0
# ... with 87,077 more rows
```

Regional Data

```
> # ipumsi data for panama
> df1 <- read_csv("./data/pan_summary1.csv") %>%
+   tbl_df()
> df1
# A tibble: 42 x 6
    year geolev1    pop      age          fb   low_edu
   <int>   <int>  <int>    <dbl>       <dbl>     <dbl>
 1  1960  591002  92960 21.25194 0.002151463 0.6086957
 2  1960  591003 105960 24.03511 0.215364288 0.3593246
 3  1960  591004 346420 21.01068 0.113446106 0.5514370
 4  1960  591005  20320 20.03051 0.285433071 0.4206897
 5  1960  591006  61100 22.38789 0.001636661 0.6442843
 6  1960  591007  68900 22.41190 0.001451379 0.6505080
 7  1960  591008 375400 24.49030 0.038190740 0.3937256
 8  1970  591002 125940 21.39495 0.004687003 0.5719622
 9  1970  591003 141870 23.60337 0.068084804 0.4830993
10  1970  591004 460030 21.10673 0.012089585 0.6112803
# ... with 32 more rows
```

Linking Shape File to Regional Data

```
> # merging data set from the @data slot
> md1 <- pan1@data %>%
+   tbl_df() %>%
+   rownames_to_column(var = "id") %>%
+   select(id, GEOLEVEL1, ADMIN_NAME) %>%
+   mutate(GEOLEVEL1 = as.character(GEOLEVEL1),
+          GEOLEVEL1 = as.integer(GEOLEVEL1))
> # notice no 888888 Waterbody anymore
> md1
# A tibble: 7 x 3
     id GEOLEVEL1                                                ADMIN_NAME
  <chr>     <int>                                                   <fctr>
1     0    591004 Bocas de Toro, Chiriquí, Comarca Ngäbe Buglé, Veraguas
2     1    591002                                                    Coclé
3     2    591005                                    Comarca Emberá, Darién
4     3    591007                                               Los Santos
5     4    591008                                                   Panamá
6     5    591003                        Colón, Comarca Kuna Yala (San Blas)
7     7    591006                                                  Herrera
```

Linking Shape File to Data

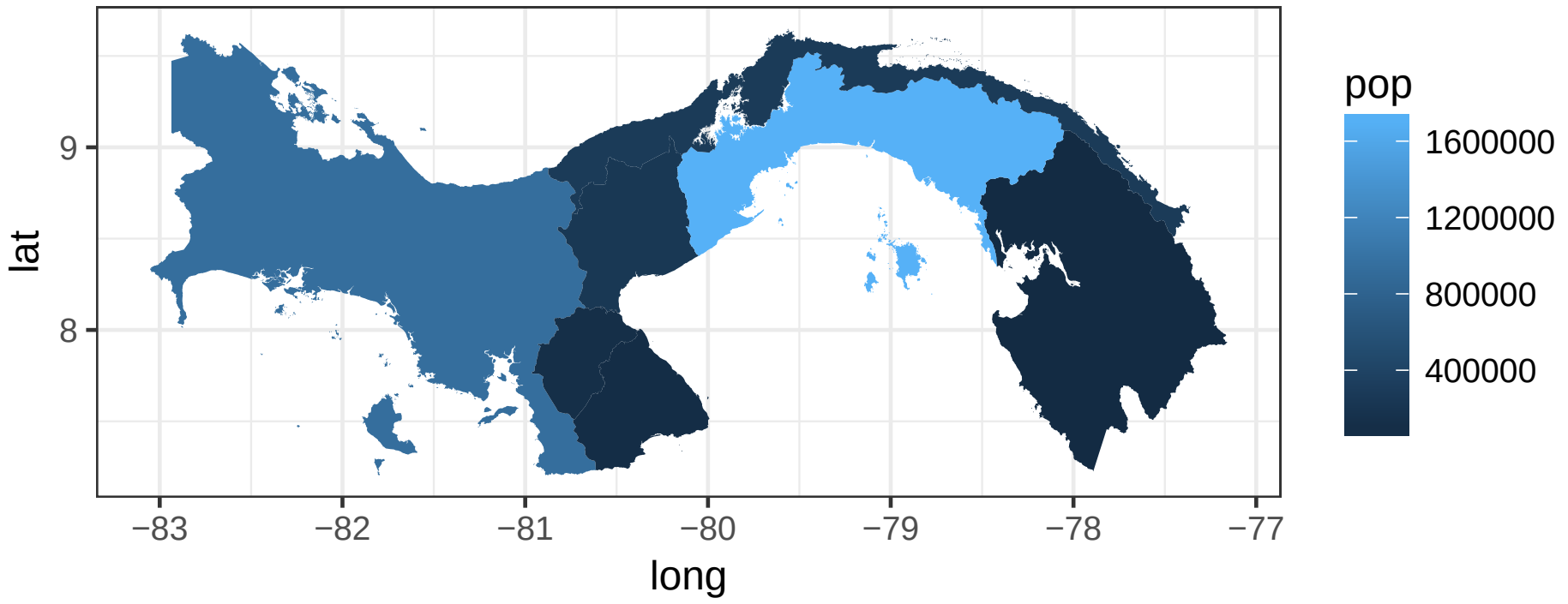
```
> # add the id column from the linking data (md1) to the regional data (df1)
> df2 <- df1 %>%
+   filter(year == 2010) %>%
+   left_join(md1, by = c("geolev1" = "GEOLEVEL1"))
> # everything() selects all other remaining variables
> select(df2, id, geolev1, everything())
# A tibble: 7 x 8
     id geolev1  year     pop      age         fb   low_edu
  <chr>   <int> <int>   <int>    <dbl>      <dbl>     <dbl>
1     1  591002  2010  232480 30.58455 0.01996214 0.2809774
2     5  591003  2010  274880 28.24742 0.03979447 0.2668801
3     0  591004  2010  929900 28.09281 0.01414241 0.3772352
4     2  591005  2010   57880 26.23134 0.04694266 0.4822609
5     7  591006  2010  110220 34.14544 0.01216302 0.2788942
6     3  591007  2010   91060 36.76203 0.01022203 0.2786344
7     4  591008  2010 1714760 30.71708 0.06431776 0.1969602
# ... with 1 more variables: ADMIN_NAME <fctr>
```

Filling Polygons

```
> # add the regional data with the ids (df2) to the tidied shape file (sd1)
> sd2 <- left_join(sd1, df2)
> sd2
# A tibble: 87,087 x 14
       long      lat order  hole  piece  group     id  year geolev1    pop
      <dbl>    <dbl> <int> <lgl> <fctr> <fctr> <chr> <int>   <int>  <int>
 1 -82.37736 9.255870     1 FALSE      1    0.1     0  2010  591004 929900
 2 -82.37522 9.255750     2 FALSE      1    0.1     0  2010  591004 929900
 3 -82.37239 9.252584     3 FALSE      1    0.1     0  2010  591004 929900
 4 -82.37016 9.252500     4 FALSE      1    0.1     0  2010  591004 929900
 5 -82.36758 9.250944     5 FALSE      1    0.1     0  2010  591004 929900
 6 -82.36359 9.250500     6 FALSE      1    0.1     0  2010  591004 929900
 7 -82.36203 9.249361     7 FALSE      1    0.1     0  2010  591004 929900
 8 -82.36193 9.248474     8 FALSE      1    0.1     0  2010  591004 929900
 9 -82.36178 9.247194     9 FALSE      1    0.1     0  2010  591004 929900
10 -82.36314 9.245475    10 FALSE      1    0.1     0  2010  591004 929900
# ... with 87,077 more rows, and 4 more variables: age <dbl>, fb <dbl>,
#   low_edu <dbl>, ADMIN_NAME <fctr>
```

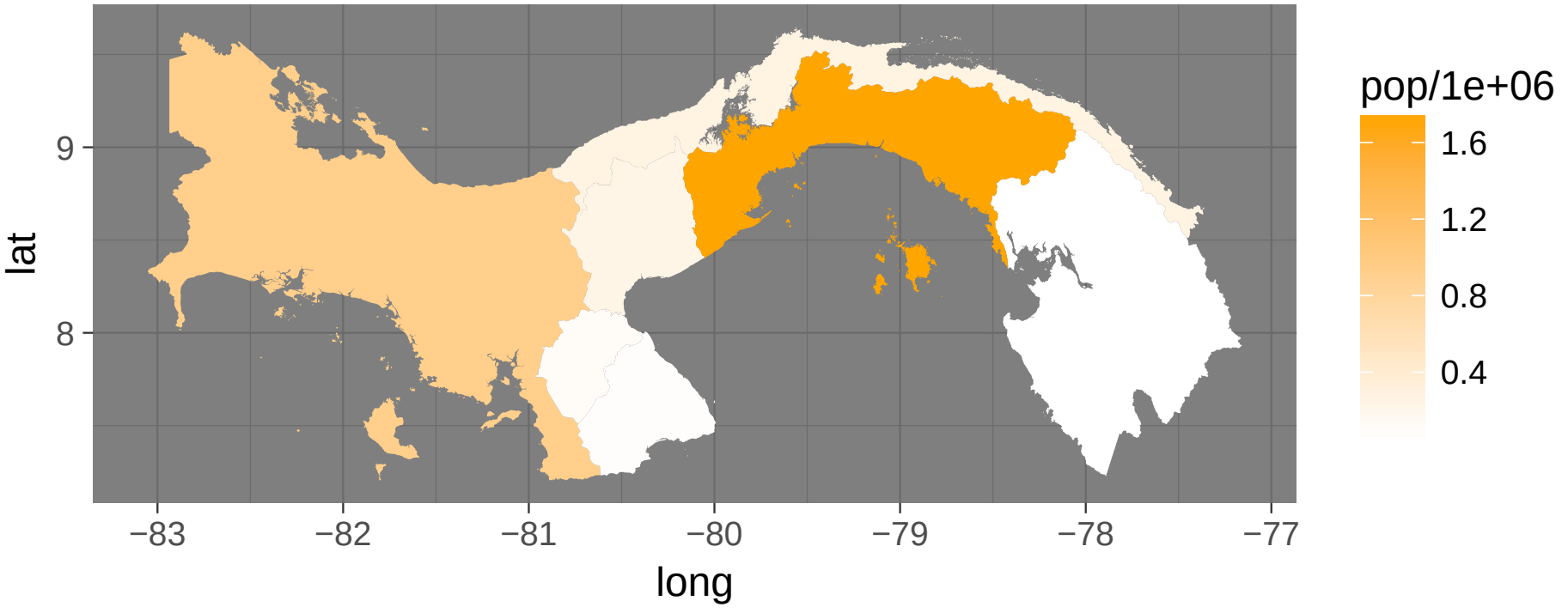
Filling Polygons

```
> ggplot(data = sd2,
+        mapping = aes(x = long, y = lat, group = group, fill = pop))  +
+   geom_polygon() +
+   coord_equal() +
+   theme_bw()
```



Filling Polygons

```
> ggplot(data = sd2,
+        mapping = aes(x = long, y = lat, group = group, fill = pop/1e06)) +
+   geom_polygon() +
+   coord_equal() +
+   scale_fill_gradient(low = "white", high = "orange") +
+   theme_dark()
```



Plotting Points

- Can add other types of geoms to the `ggplot`, such as points.
- Requires a new data frame (`cd2` below) with coordinates corresponding to location of points
  - Determine the center of a polygons using the `gCentroid()` function in `rgeos` package
  - Alternatively might have an external data source or shape file of specific points (e.g. survey sights, city locations, GPS tracks)

Plotting Points

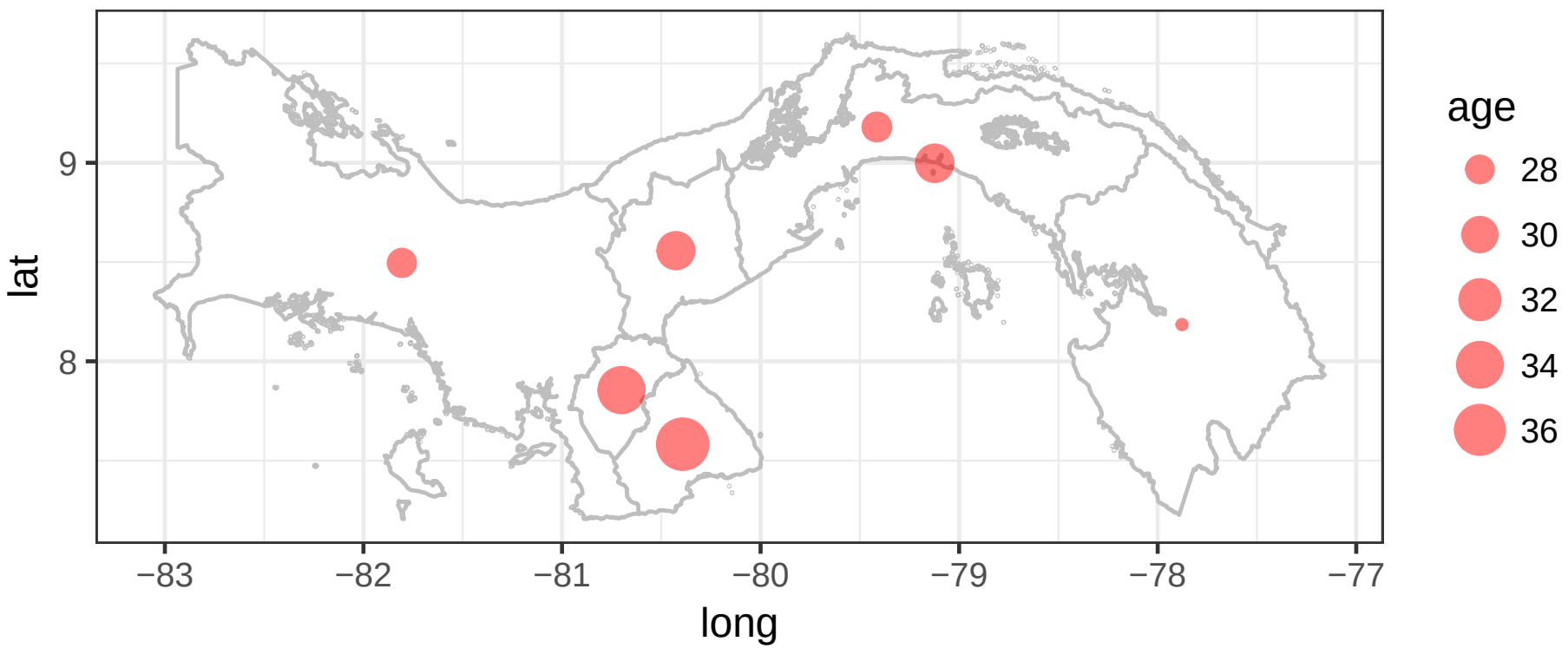
```
> # install.packages("rgeos")
> library(rgeos)
rgeos version: 0.3-26, (SVN revision 560)
 GEOS runtime version: 3.6.1-CAPI-1.10.1 r0
 Linking to sp version: 1.2-5
 Polygon checking: TRUE
> cd1 <- gCentroid(pan1, byid = TRUE) %>%
+   as.data.frame() %>%
+   rownames_to_column(var = "id")
> cd1
  id         x        y
1  0 -81.80643 8.495707
2  1 -80.42614 8.557500
3  2 -77.87756 8.185172
4  3 -80.39213 7.582804
5  4 -79.12272 8.997573
6  5 -79.41397 9.180383
7  7 -80.70050 7.854864
```

Plotting Points

```
> # add regional data
> cd2 <- left_join(cd1, df2)
Joining, by = "id"
> cd2
  id          x         y year geolev1     pop      age          fb   low_edu
1  0 -81.80643 8.495707 2010  591004  929900 28.09281 0.01414241 0.3772352
2  1 -80.42614 8.557500 2010  591002  232480 30.58455 0.01996214 0.2809774
3  2 -77.87756 8.185172 2010  591005   57880 26.23134 0.04694266 0.4822609
4  3 -80.39213 7.582804 2010  591007   91060 36.76203 0.01022203 0.2786344
5  4 -79.12272 8.997573 2010  591008 1714760 30.71708 0.06431776 0.1969602
6  5 -79.41397 9.180383 2010  591003  274880 28.24742 0.03979447 0.2668801
7  7 -80.70050 7.854864 2010  591006  110220 34.14544 0.01216302 0.2788942
                                                 ADMIN_NAME
1 Bocas de Toro, Chiriquí, Comarca Ngäbe Buglé, Veraguas
2                                                   Coclé
3                                  Comarca Emberá, Darién
4                                               Los Santos
5                                                   Panamá
6                        Colón, Comarca Kuna Yala (San Blas)
7                                                 Herrera
```

Plotting Points

```
> ggplot(data = sd2,
+        mapping = aes(x = long, y = lat, group = group)) +
+   geom_polygon(colour = "grey", fill = "white") +
+   coord_equal() +
+   geom_point(data = cd2,
+              mapping = aes(x = x, y = y, group = id, size = age),
+              colour = "red", alpha = 0.5) +
+   theme_bw()
```



Plotting Names

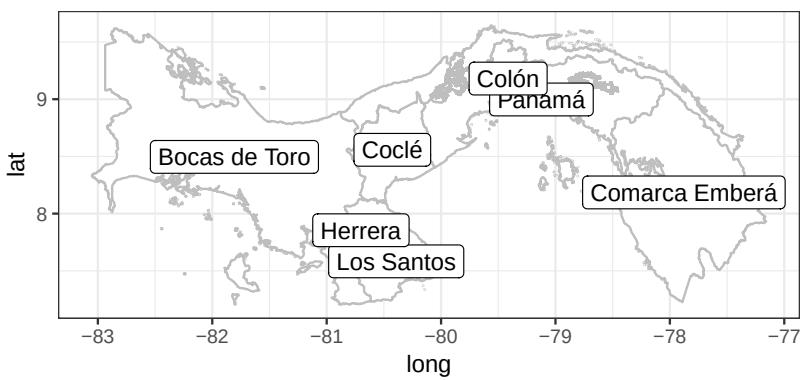
- Labels from the shape file are very long.
- Can put over multiple line by inserting \n for line breaks

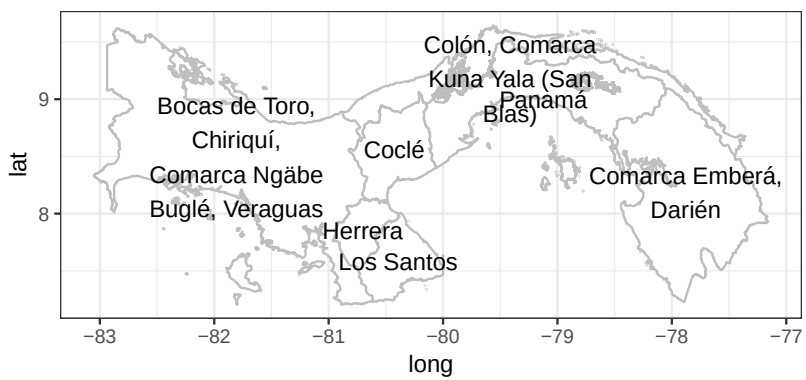
```
> library(stringr)
> #use str_wrap to put labels on multiple rows
> cd2 <- cd2 %>%
+   mutate(name1 = as.character(ADMIN_NAME),
+          name1 = str_wrap(name1, width = 15)) %>%
+   separate(col = ADMIN_NAME, into = c("name2", "waste"), sep = c(", "),
+            remove = FALSE, fill = "right", extra = "drop")
> select(cd2, name1, name2)
                                                    name1           name2
1 Bocas de Toro,\nChiriquí,\nComarca Ngäbe\nBuglé, Veraguas  Bocas de Toro
2                                                   Coclé           Coclé
3                                  Comarca Emberá,\nDarién Comarca Emberá
4                                              Los Santos     Los Santos
5                                                  Panamá         Panamá
6                        Colón, Comarca\nKuna Yala (San\nBlas)         Colón
7                                                 Herrera        Herrera
```

Plotting Names

- Can plot using `geom_label()` or `geom_text()`

```
> ggplot(data = sd2, mapping = aes(x = long, y = lat, group = group)) +
+   geom_polygon(colour = "grey", fill = "white") +
+   coord_equal() +
+   geom_label(data = cd2,
+              mapping = aes(x = x, y = y, group = id, label = name2)) +
+   theme_bw()
>
> ggplot(data = sd2, mapping = aes(x = long, y = lat, group = group)) +
+   geom_polygon(colour = "grey", fill = "white") +
+   coord_equal() +
+   geom_text(data = cd2,
+             mapping = aes(x = x, y = y, group = id, label = name1)) +
+   theme_bw()
```





Maps in Facets

- To use facets we need to add multiple values for the same variables to each row of the tidied shape file (`df1`)
  - Multiplies up the size of the spatial data frame by the number of categories in the facet variable(s) (e.g 522,522 in `sd3` below from 128,768 in `sd2`)

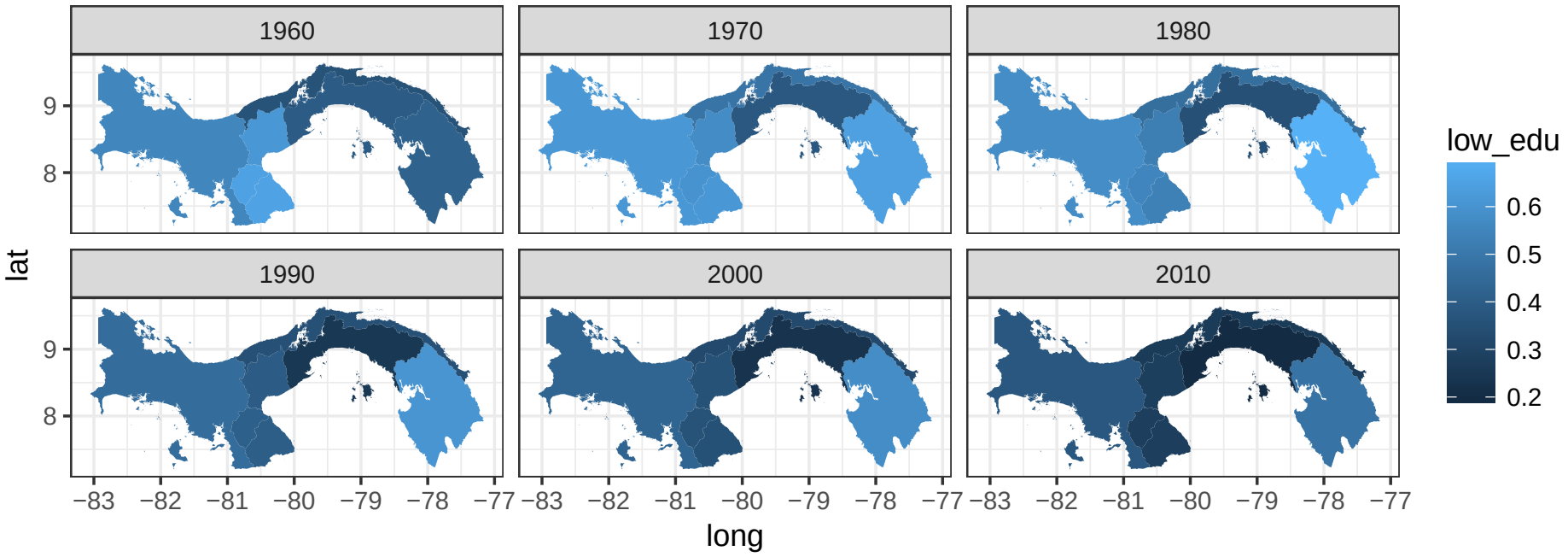
```
> # add the id column from the merging data (md1) to the whole regional data (df1)
> df3 <- left_join(df1, md1, by = c("geolev1" = "GEOLEVEL1"))
> select(df3, id, geolev1, everything())
# A tibble: 42 x 8
      id geolev1  year    pop      age          fb   low_edu
   <chr>   <int> <int>  <int>    <dbl>       <dbl>     <dbl>
 1     1  591002  1960  92960 21.25194 0.002151463 0.6086957
 2     5  591003  1960 105960 24.03511 0.215364288 0.3593246
 3     0  591004  1960 346420 21.01068 0.113446106 0.5514370
 4     2  591005  1960  20320 20.03051 0.285433071 0.4206897
 5     7  591006  1960  61100 22.38789 0.001636661 0.6442843
 6     3  591007  1960  68900 22.41190 0.001451379 0.6505080
 7     4  591008  1960 375400 24.49030 0.038190740 0.3937256
 8     1  591002  1970 125940 21.39495 0.004687003 0.5719622
 9     5  591003  1970 141870 23.60337 0.068084804 0.4830993
10     0  591004  1970 460030 21.10673 0.012089585 0.6112803
# ... with 32 more rows, and 1 more variables: ADMIN_NAME <fctr>
```

Maps in Facets

```
> # add the regional data in all years with the ids (df3) to the tidied shape file (
> sd3 <- left_join(sd1, df3)
> sd3
# A tibble: 522,522 x 14
        long     lat order  hole  piece  group     id  year geolev1    pop
       <dbl>   <dbl> <int> <lgl> <fctr> <fctr> <chr> <int>   <int>  <int>
 1 -82.37736 9.25587     1 FALSE      1    0.1     0  1960  591004 346420
 2 -82.37736 9.25587     1 FALSE      1    0.1     0  1970  591004 460030
 3 -82.37736 9.25587     1 FALSE      1    0.1     0  1980  591004 554940
 4 -82.37736 9.25587     1 FALSE      1    0.1     0  1990  591004 666130
 5 -82.37736 9.25587     1 FALSE      1    0.1     0  2000  591004 778810
 6 -82.37736 9.25587     1 FALSE      1    0.1     0  2010  591004 929900
 7 -82.37522 9.25575     2 FALSE      1    0.1     0  1960  591004 346420
 8 -82.37522 9.25575     2 FALSE      1    0.1     0  1970  591004 460030
 9 -82.37522 9.25575     2 FALSE      1    0.1     0  1980  591004 554940
10 -82.37522 9.25575     2 FALSE      1    0.1     0  1990  591004 666130
# ... with 522,512 more rows, and 4 more variables: age <dbl>, fb <dbl>,
#   low_edu <dbl>, ADMIN_NAME <fctr>
```

Maps in Facets

```
> ggplot(data = sd3,
+        mapping = aes(x = long, y = lat, group = group, fill = low_edu)) +
+   geom_polygon() +
+   coord_equal() +
+   facet_wrap("year") +
+   theme_bw()
```



Exercise 2 (ex62.R)

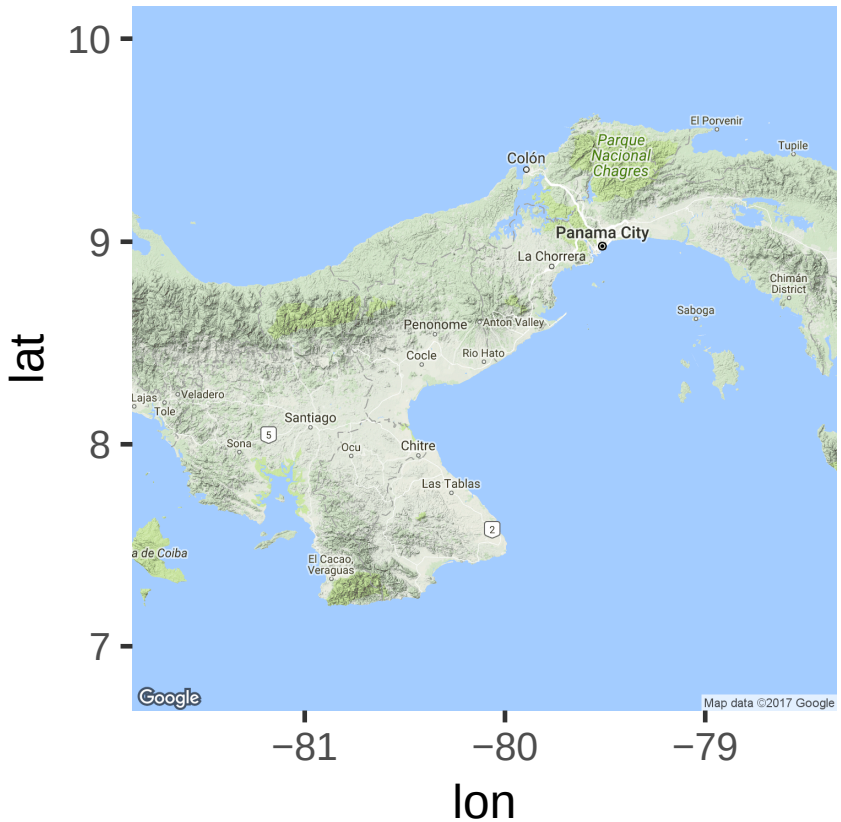
```
# 0. a) set your working directory to the course folder
```

Tiling

- The ggmap package downloads background tiles for your plot using the `get_map()` function. Requires
  - `location`: four coordinates for the background
  - `map_type`: choose from 16, including `"terrain"`, `"satellite"`, `"roadmap"`
  - `source`: choose from `"google"`, `"osm"`, `"stamen"`, ...
- Downloads tiles from internet.
  - Use the `ggmap()` function to initialise the plot with the tile (rather than `ggplot()`)
  - Add geoms on-top of the tiles.

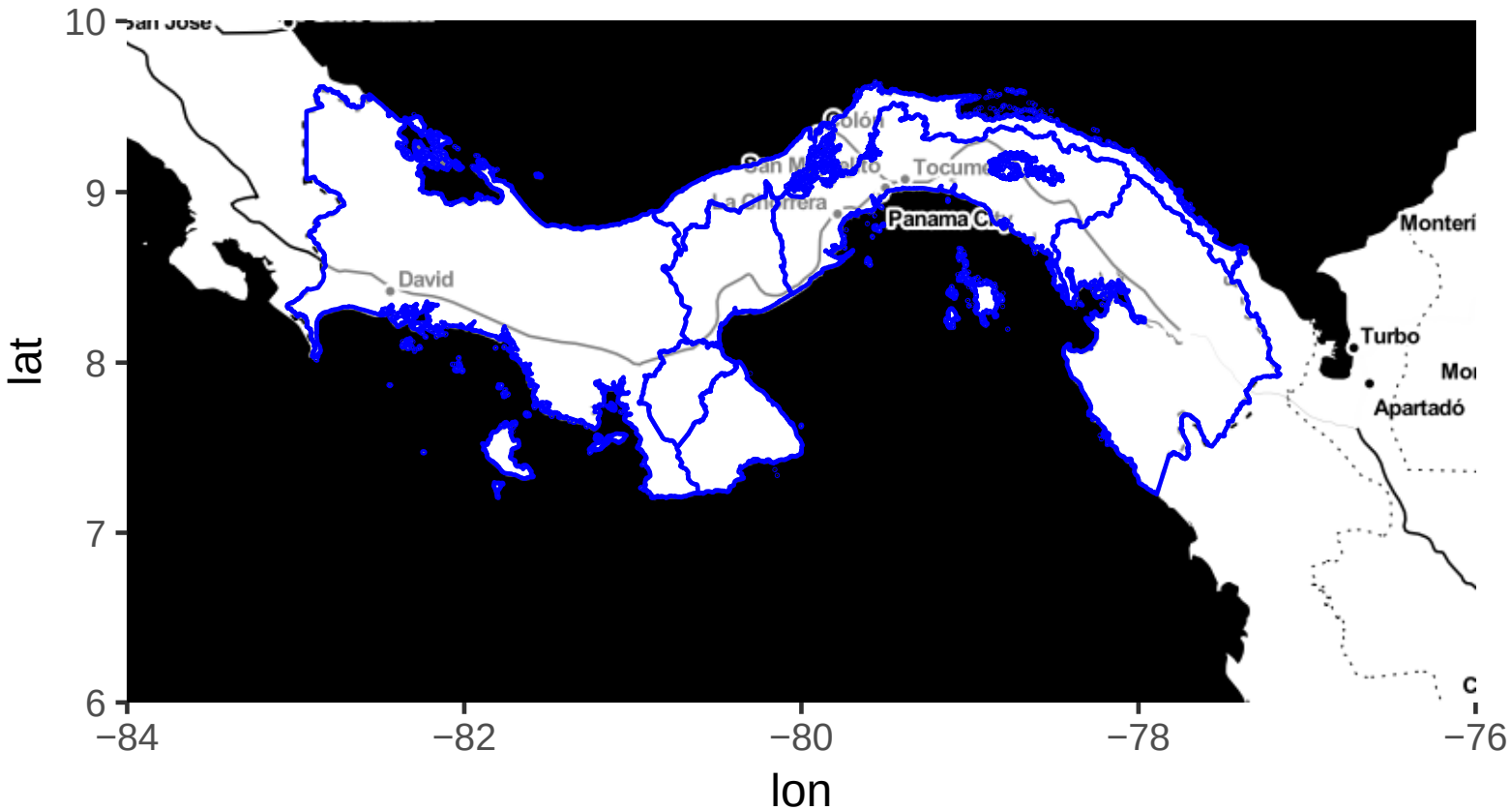
Tiling

```
> # install.packages("ggmap")
> library(ggmap)
> pan_t1 <- get_map(location = pan1@bbox,
+                   maptype = "terrain", source = "google")
> ggmap(pan_t1)
```



Tiling

```
> pan_t2 <- get_map(location = c(-84, 6, -76, 10), #left/bottom/right/top
+                   maptype = "toner", source = "stamen")
> ggmap(pan_t2) +
+   geom_polygon(data = sd2,
+                mapping = aes(x = long, y = lat, group = group),
+                colour = "blue", alpha = 0.5, fill = "transparent")
```



Geofacet

- Geofacets are an increasingly popular way to show simplified spatial-temporal patterns.
- The `geofacet` package in R allows users to create a facet arrangement for regions similar to a geographic arrangement regions - The `grid_design()` function is used to arrange the facets. Main arguments are
  - `~`: indicates the level of units by which you want to facet your plot (e.g. ~state)
  - `grid=`: the name of the grid you will be using. Use either built-in or user-defined grids (e.g. "us_state_grid2")
  - `label=`: input a column name of the grid data file to label the facets (e.g. "name")

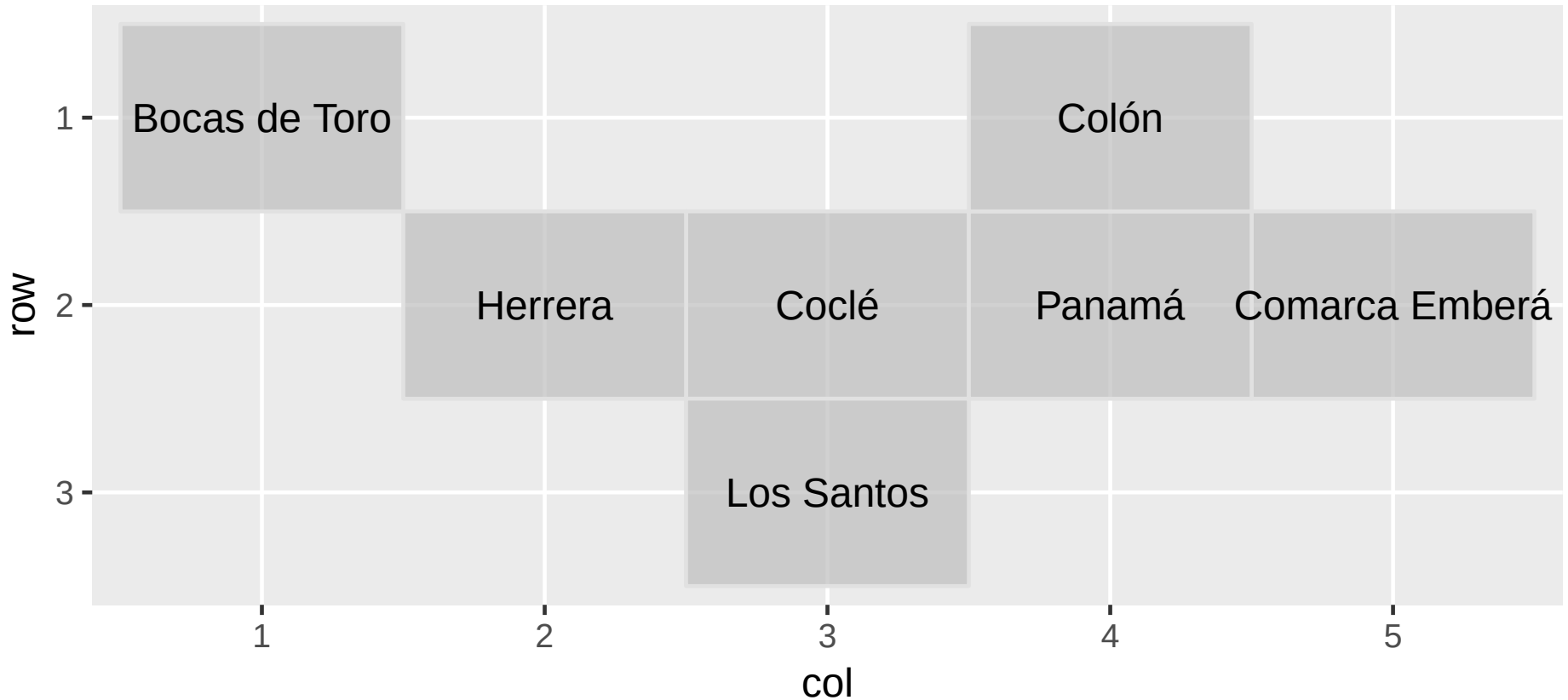
Creating a Geofacet

- To plot the panama data (`df1`) on a grid, you need to create your own grid for the first-level geographic units.

```
> # install.packages("geofacet")
> library(geofacet)
>
> pan1_grid <- cd2 %>%
+   select(geolev1, name2) %>%
+   # must have column names code and name
+   rename(code = geolev1, name = name2)
> pan1_grid
   code           name
1 591004  Bocas de Toro
2 591002          Coclé
3 591005 Comarca Emberá
4 591007     Los Santos
5 591008         Panamá
6 591003          Colón
7 591006        Herrera
```

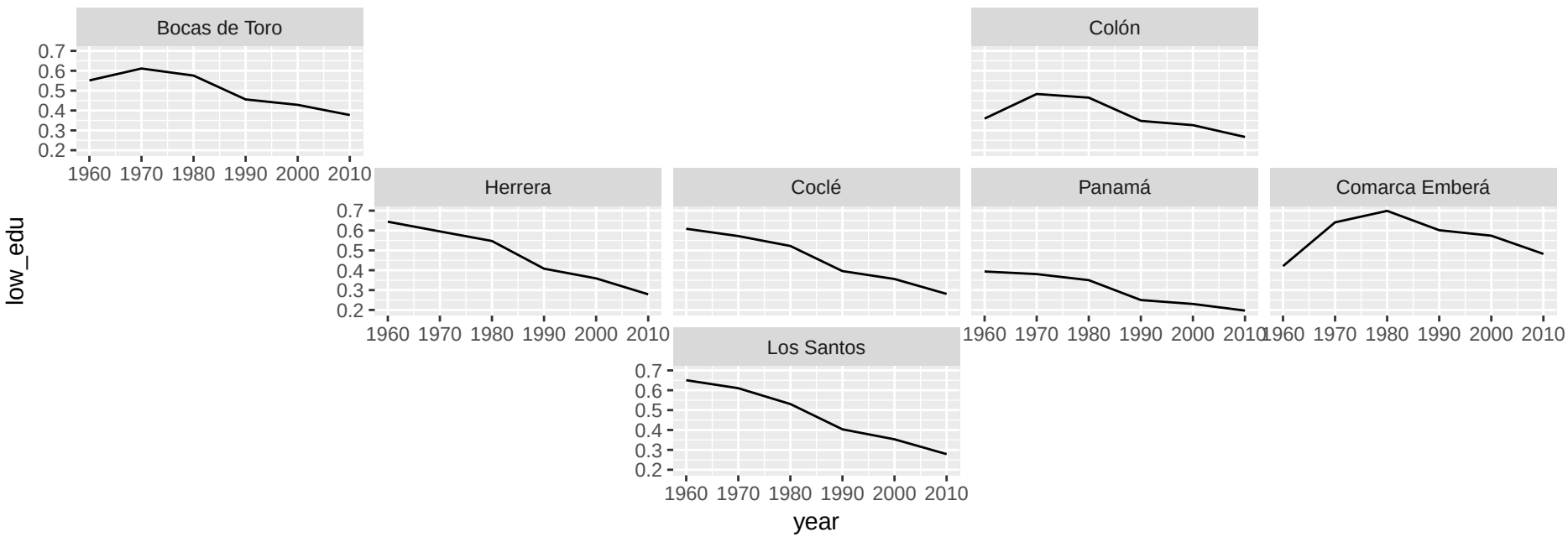
Preview Grid

```
> pan1_grid <- pan1_grid %>%
+   mutate(row = c(1, 2, 2, 3, 2, 1, 2),
+          col = c(1, 3, 5, 3, 4, 4, 2))
>
> grid_preview(pan1_grid, label = "name")
```



Add Data

```
> # need columns in regional data to match names in pan1_grid.
> df3 <- df3 %>%
+   left_join(pan1_grid, by = c("geolev1" = "code")) %>%
+   mutate(code = geolev1)
>
> # use standard ggplot2 code, with facet_geo (rather than facet_wrap)
> ggplot(data = df3,
+        mapping = aes(x = year, y = low_edu)) +
+   geom_line() +
+   facet_geo("code", grid = pan1_grid, label = "name")
```



Exercise 3 (ex63.R)

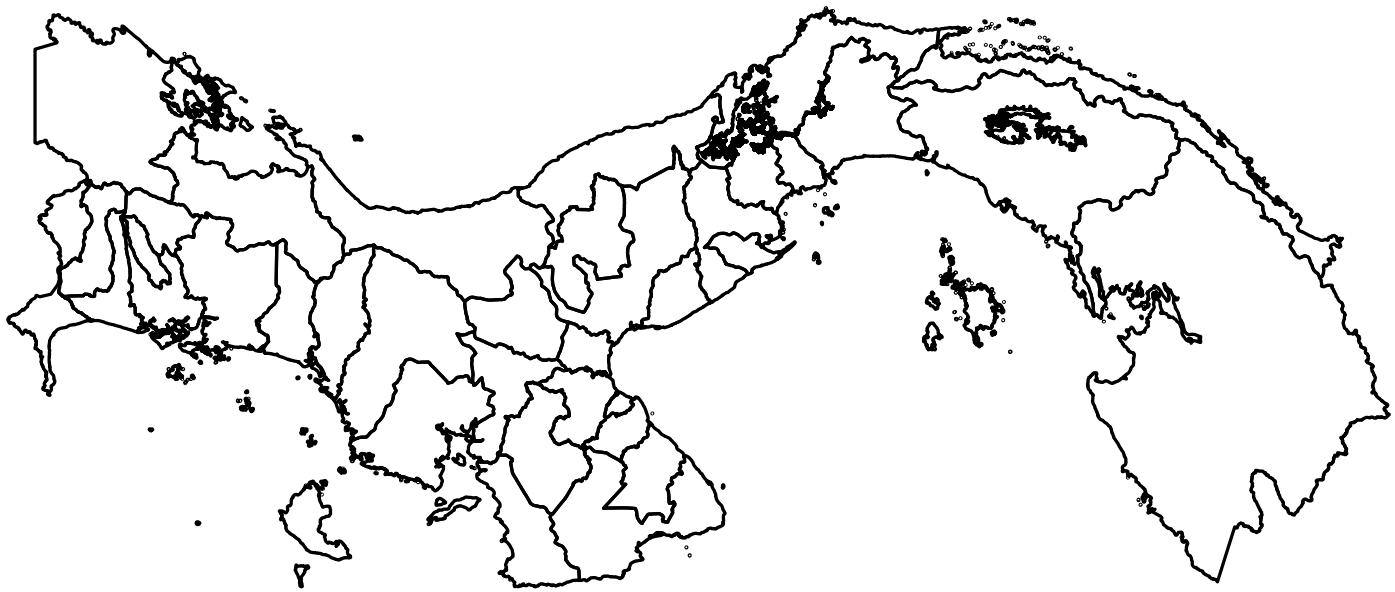
```
# 0. a) set your working directory to the course folder

#    b) read in the sweden1993.csv data

##
##
##
# 1. commplete the following code to calcualte the female survivorship
d0 <- d0 %>%
  mutate(sx_f = lead(#####)/Lx_f,
         sx_f = ifelse(test = row_number() == n() - #####, yes = lead(#####) / (###
         sx_f = ifelse(test = ##### == n(), yes = lag(sx_f), no = #####))
# 2. create a matrices called LL and ss for time specific mortality age schedules
#    for 18 age groups over 50 years
LL <- matrix(#####$Lx, nrow = 18, ncol = 10)
ss <- matrix(d0$#####, nrow = 18, ncol = #####)
# 3. create a sequances of alpha to move female Sweden 1993 life expectancies from
#    current value (alpha = 0) to 86 over 50 years
alpha <- #####(from = 0, to = -0.518, length.out = #####)
alpha
# 4. adjust Lx and sx in LL and ss for the increase in life expectancies using alpha
for(i in 2:10){
  #adjust for change at time i
  LL[, i] <- brass_mort(model = d0$Lx_f, x = d0$x, alpha = #####, beta = 1)
```

Cartograms

- A cartogram is a map in which a variable is substituted for land area or distance.
  - The geometry or space of the map is distorted in order to convey the information of this alternate variable.
  - geofacts are a type of cartogram
- The `hexmapr` can generate square or hexagonal grids to map a variable by geographic units
  - Still under development on Github
- Can generate grids (unlike `geofacet` package)
  - Useful with many regions.
- The generation of grids has three steps
  1. Read geospatial polygons from a shape file.
  2. Generate some potential grids and their sizes using the `calculate_cell_size()` function
     - Need to specify the learning rate (e.g. 0.03) and type of plot (e.g. `regular` or `hexagonal`).
  3. Assigns the original polygons to their new locations on the generated grid using `assign_polygons()`.

Step 1: Read in Polygon

```
> # read geospatial polygons from a shapefile.
> pan2 <- readOGR("./data/geo2_pa1960_2010/geo2_pa1960_2010.shp")
OGR data source with driver: ESRI Shapefile
Source: "./data/geo2_pa1960_2010/geo2_pa1960_2010.shp", layer: "geo2_pa1960_2010"
with 36 features
It has 5 fields
> # drop lake polygons
> pan2 <- pan2[pan2$GEOLEVEL2 != 591888888, ]
> par(mar = rep(0, 4))
> plot(pan2)
```

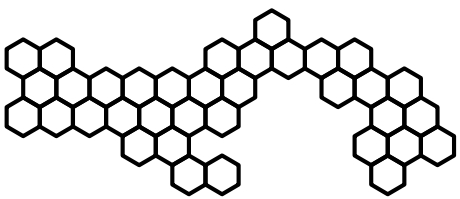


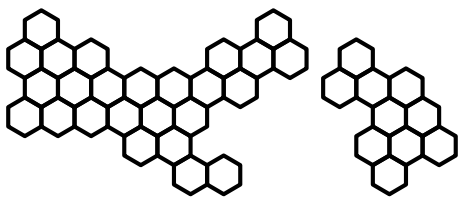
Step 2: Generating Potential Grids

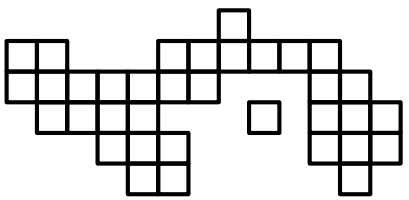
```
> # install.packages("devtools")
> # library(devtools)
> # install_github("sassalley/hexmapr")
> library(hexmapr)
>
> pan_hex100 <- calculate_cell_size(shape = pan2, seed = 100,
+                                   shape_details = get_shape_details(pan2),
+                                   learning_rate = 0.03, grid_type = 'hexagonal')
>
> pan_hex101 <- calculate_cell_size(shape = pan2, seed = 101,
+                                   shape_details = get_shape_details(pan2),
+                                   learning_rate = 0.05, grid_type = 'hexagonal')
>
> pan_reg102 <- calculate_cell_size(shape = pan2, seed = 102,
+                                   shape_details = get_shape_details(pan2),
+                                   learning_rate = 0.03, grid_type = 'regular')
```

Step 2: Generating Potential Grids

```
> #outputs a list with two components, e.g.;
> str(pan_hex100, max.level = 1)
List of 2
 $ :Formal class 'SpatialPoints' [package "sp"] with 3 slots
 $ :Formal class 'SpatialPolygons' [package "sp"] with 4 slots
> par(mar = rep(0,4))
> plot(pan_hex100[[2]])
> plot(pan_hex101[[2]])
> plot(pan_reg102[[2]])
```







Step 3: Assigns Polygons

```
> pan_hex <- assign_polygons(shape = pan2, new_polygons = pan_hex100)
>
> # use ipumsi regional data for panama at the second administrative level
> df4 <- read_csv("./data/pan_summary2.csv") %>%
+   filter(year == 2010)
> df4
# A tibble: 35 x 6
    year   geolev2    pop      age          fb   low_edu
   <int>     <int>  <int>    <dbl>       <dbl>     <dbl>
 1  2010 591002001 103900 29.75207 0.008375048 0.2856729
 2  2010 591002002  53990 30.77325 0.052426825 0.2521989
 3  2010 591002003  42930 32.61868 0.019100862 0.2264987
 4  2010 591002004  31660 30.23658 0.003791469 0.3871478
 5  2010 591003001 218930 28.68291 0.047146061 0.2174738
 6  2010 591003002  23740 25.67186 0.003369840 0.3675105
 7  2010 591003003  32210 27.18566 0.016770186 0.5273070
 8  2010 591004001 155500 23.01640 0.019874814 0.4465956
 9  2010 591004002 144930 31.91492 0.017045062 0.2160985
10  2010 591004003  99420 31.22913 0.015492958 0.2223456
# ... with 25 more rows
```

Merging Data

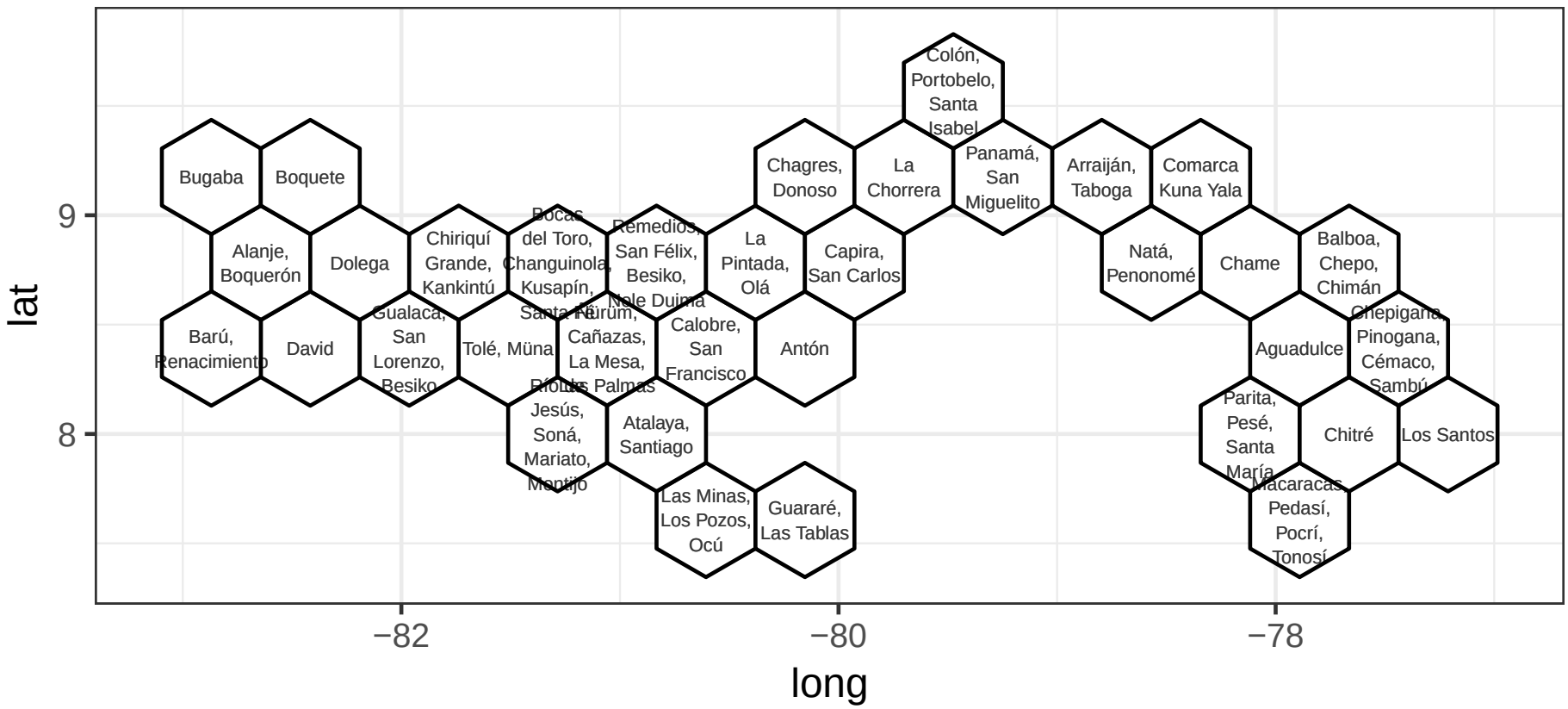
```
> # create merging data (use rowid_to_column() not rownames_to_column())
> md4 <- pan_hex@data %>%
+   tbl_df() %>%
+   rowid_to_column(var = "id") %>%
+   mutate(GEOLEVEL2 = as.character(GEOLEVEL2),
+          GEOLEVEL2 = as.integer(GEOLEVEL2),
+          id = paste0("ID", id))
> md4
# A tibble: 35 x 14
      id                                key_new        V1       V2
   <chr>                                  <chr>     <dbl>    <dbl>
 1   ID1  -80.606191715552_7.60665285118536 -80.60619 7.606653
 2   ID2 -80.1535345550276_7.60665285118536 -80.15353 7.606653
 3   ID3 -77.8902487524058_7.60665285118536 -77.89025 7.606653
 4   ID4 -81.2851774563385_7.99866545140439 -81.28518 7.998665
 5   ID5 -80.8325202958141_7.99866545140439 -80.83252 7.998665
 6   ID6  -78.116577332668_7.99866545140439 -78.11658 7.998665
 7   ID7 -77.6639201721436_7.99866545140439 -77.66392 7.998665
 8   ID8 -77.2112630116192_7.99866545140439 -77.21126 7.998665
 9   ID9 -82.8694775181738_8.39067805162342 -82.86948 8.390678
10  ID10 -82.4168203576494_8.39067805162342 -82.41682 8.390678
# ... with 25 more rows, and 10 more variables: key_orig <chr>,
#   CNTRY_NAME <fctr>, ADMIN_NAME <fctr>, CNTRY_CODE <fctr>,
#   GEOLEVEL2 <int>, PARENT <fctr>, CENTROIX <dbl>, CENTROIY <dbl>,
```

Merge Data and Grid

```
> sd4 <- pan_hex %>%
+   tidy() %>%
+   tbl_df() %>%
+   left_join(md4, by = "id") %>%
+   left_join(df4, by = c("GEOLEVEL2" = "geolev2"))
Regions defined for each Polygons
> sd4
# A tibble: 245 x 25
       long      lat order  hole  piece  group    id
      <dbl>    <dbl> <int> <lgl> <fctr> <fctr> <chr>
 1 -80.83252 7.737324     1 FALSE      1   ID1.1   ID1
 2 -80.60619 7.867995     2 FALSE      1   ID1.1   ID1
 3 -80.37986 7.737324     3 FALSE      1   ID1.1   ID1
 4 -80.37986 7.475982     4 FALSE      1   ID1.1   ID1
 5 -80.60619 7.345311     5 FALSE      1   ID1.1   ID1
 6 -80.83252 7.475982     6 FALSE      1   ID1.1   ID1
 7 -80.83252 7.737324     7 FALSE      1   ID1.1   ID1
 8 -80.37986 7.737324     1 FALSE      1   ID2.1   ID2
 9 -80.15353 7.867995     2 FALSE      1   ID2.1   ID2
10 -79.92721 7.737324     3 FALSE      1   ID2.1   ID2
# ... with 235 more rows, and 18 more variables: key_new <chr>, V1 <dbl>,
#   V2 <dbl>, key_orig <chr>, CNTRY_NAME <fctr>, ADMIN_NAME <fctr>,
#   CNTRY_CODE <fctr>, GEOLEVEL2 <int>, PARENT <fctr>, CENTROIX <dbl>,
#   CENTROIY <dbl>, row <int>, col <int>, year <int>, pop <int>,
```

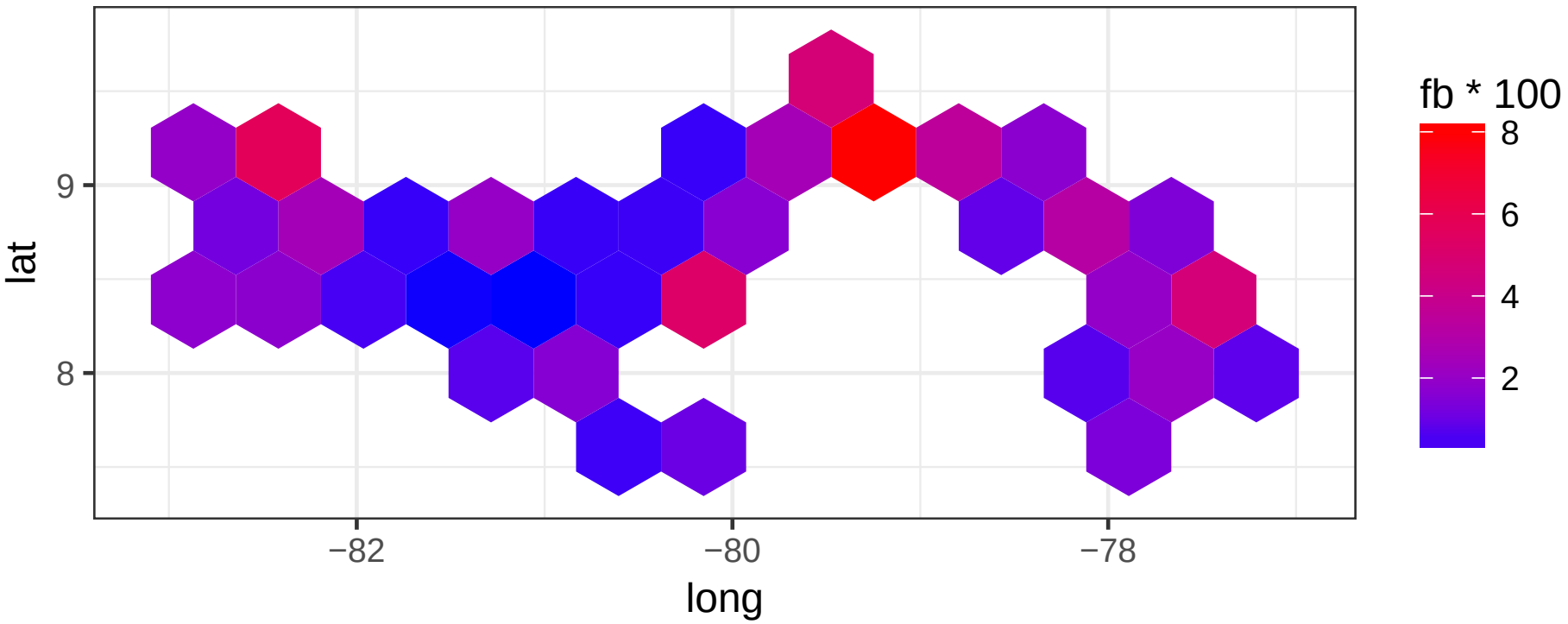
Double Check Layout

```
> ggplot(data = sd4,
+        mapping = aes(x = long, y = lat, group = id,
+                      label = str_wrap(ADMIN_NAME, width = 10))) +
+   geom_polygon(fill = "white", colour = "black") +
+   geom_text(mapping = aes(x = V1, y = V2), size = 1.6, alpha = 0.2) +
+   coord_equal() +
+   theme_bw()
```



Plot Data

```
> ggplot(data = sd4,
+        mapping = aes(x = long, y = lat, group = group, fill = fb*100)) +
+   geom_polygon() +
+   coord_equal() +
+   scale_fill_gradient(low = "blue", high = "red") +
+   theme_bw()
```



Exercise 4 (ex64.R)

```
# 0. a) clear your workspace and set your working directory to the course folder


#    b) read in the austria.csv data

#    c) load the tidyverse, shiny and demoproj packages



#    d) run the code below to load the data and caculate the survivorships
# d1 <- filter(d0, Year == 2014) %>%
#   mutate(sx_f = lead(Lx_f)/Lx_f,
#          sx_f = ifelse(test = row_number() == n() - 1, yes = lead(Lx_f) / (lead(Lx
#          sx_f = ifelse(test = row_number() == n(), yes = lag(sx_f), no = sx_f),
#          sx_m = lead(Lx_m)/Lx_m,
#          sx_m = ifelse(test = row_number() == n() - 1, yes = lead(Lx_m) / (lead(Lx
#          sx_m = ifelse(test = row_number() == n(), yes = lag(sx_m), no = sx_m)) %>
#   replace_na(list(Age = 110))
##
##
##
# 1. calculate survivorships of newborn for females and males
snewborn_f <- #####$Lx_f[#####]/#####
snewborn_m <- df1$#####[1]/#####
```