GIS III: Final Project

Rachel Steiner-Dillon

6/5/2020

This project downloads, cleans, and aggregates U.S. domestic migration data from the IRS for a given set of input years. Sample visualizations are shown for the year 2018. The accompanying shiny app allows the user to change the input year for visualizations at the county, state, and regional level.

```
library(tidyverse)
library(tidycensus)
library(tigris)
library(sf)
library(spData)
library(glue)
library(readr)

library(leaflet)
library(ggmap)
```

Step 1: Download County Shapefiles

Import shapefile data from the U.S. Census API using the tidycensus package.

```
#get shapefile from 2014-2018 5-year aCS
counties <- get_acs(geography = 'county',
                    variables = 'B01001_001',
                    geometry = TRUE)

#Filter out Puerto Rico, Alaska, and Hawaii
counties = counties %>%
  select(GEOID, NAME, geometry) %>%
  filter(!grepl('Puerto Rico', NAME)) %>%
  filter(!grepl('Alaska', NAME)) %>%
  filter(!grepl('Hawaii', NAME))

#create a state-region crosswalk from the us_states dataframe in the spData package
data(us_states)
crosswalk <- us_states %>%
  st_drop_geometry() %>%
  select(GEOID, NAME, REGION) %>%
  rename(st_fips = GEOID)

#Separate the state and county names from the counties shapefile, and merge the crosswalk
counties = counties %>% separate(NAME, into = c('county', 'state'), sep=', ')
counties = left_join(counties, crosswalk, by=c('state'='NAME'))
```

Step 2: Prepare Migration Data

Download migration data from the IRS website.
IRS data is recorded as the change from one year to the next. The number of taxpayers who moved out of a county between 2013 and 2014 is included in a table called "outflows1314." So, years of interest must also be given in this 'yryr' format.

Note: this chunk is set not to evaluate, because the files are already in the folder

```
#setwd("C:/Users/rache/Documents/GitHub/GIS_3/final_project")

#specify that we want both inflows and outflows
directions = c('in', 'out')

#specify years of interest
years = c('1314', '1415', '1516', '1617', '1718')

#define function to download files
download_files = function(dir, year){
  url = glue('https://www.irs.gov/pub/irs-soi/county{dir}flow{years}.csv')
  filename = glue('{dir}flow{year}.csv')
  download.file(url, filename)
}

#Iterate function to get inflows and outflows files for all years of interest (10 files in all)
for(dir in directions){
  for(year in years){
    download_files(dir, year)
  }
}
```

Import and and clean inflows data.

```
setwd("C:/Users/rache/Documents/GitHub/GIS_3/final_project")
years = c('1314', '1415', '1516', '1617', '1718')

#define function
load_inflows = function(name, yr){
  df = read_csv(name)

  #eliminate rows that do not refer to unique counties
  df = subset(df, y2_countyfips!='000' & y1_statefips !='96')
  df['year'] <- yr

  #select aggregate row, merge the county and state FIPS colums to create a 5-digit GEOID
  df = df %>%
    filter(grepl('Total Migration-US', y1_countyname)) %>%
    unite('geoid', y2_statefips:y2_countyfips, sep='', remove=TRUE) %>%
    select(geoid, year, n1) %>%
    rename(inflows = n1)
}

#Run load_inflows function for each year of interest
inflow_list <- list()
for(year in years){
  df = load_inflows(glue('inflow{year}.csv'), year)
```

```
  inflow_list[[year]] <- df
}

#Merge the separate year dataframes
inflows = do.call('rbind', inflow_list)

#Restructure the year column to be a 4-digit year beginning with '20'
substring(inflows$year, 1, 2) <- '20'
```

Import and and clean outflows data.

```
setwd("C:/Users/rache/Documents/GitHub/GIS_3/final_project")
years = c('1314', '1415', '1516', '1617', '1718')

#define function
load_outflows = function(name, yr){
  df = read_csv(name)
  df = subset(df, y1_countyfips!='000' & y2_statefips !='96')
  df['year'] <- yr
  df = df %>%
    filter(grepl('Total Migration-US', y2_countyname)) %>%
    unite('geoid', y1_statefips:y1_countyfips, sep='', remove=TRUE) %>%
    select(geoid, year, n1) %>%
    rename(outflows = n1)
}

#Run load_outflows function for each year of interest
outflow_list <- list()
for(year in years){
  df = load_outflows(glue('outflow{year}.csv'), year)
  outflow_list[[year]] <- df
}

#Merge the separate year dataframes
outflows = do.call('rbind', outflow_list)

#Restructure the year column to be a 4-digit year beginning with '20'
substring(outflows$year, 1, 2) <- '20'
```

Step 3: Merge and Aggregate

Merge inflows, outflows, and counties data.
Calculate net migration per county.

```
net_join <- function(inflow, outflow, shape){
  df = inner_join(inflow, outflow, by=c('geoid', 'year'))
  df['net_migration'] = df$inflows - df$outflows
  shape_merge = left_join(shape, df, by=c('GEOID'='geoid'))
}


counties_merged = net_join(inflows, outflows, counties)
```

Aggregate counties data to the state and regional levels.

```
aggregate_geom = function(df, col){
  level <- enquo(col)
  new_df = df %>% group_by((!!level), year) %>%
    summarize(net_migrants = sum(net_migration, na.rm=TRUE))
}

states_merged = aggregate_geom(counties_merged, state)
regions_merged = aggregate_geom(counties_merged, REGION)
```

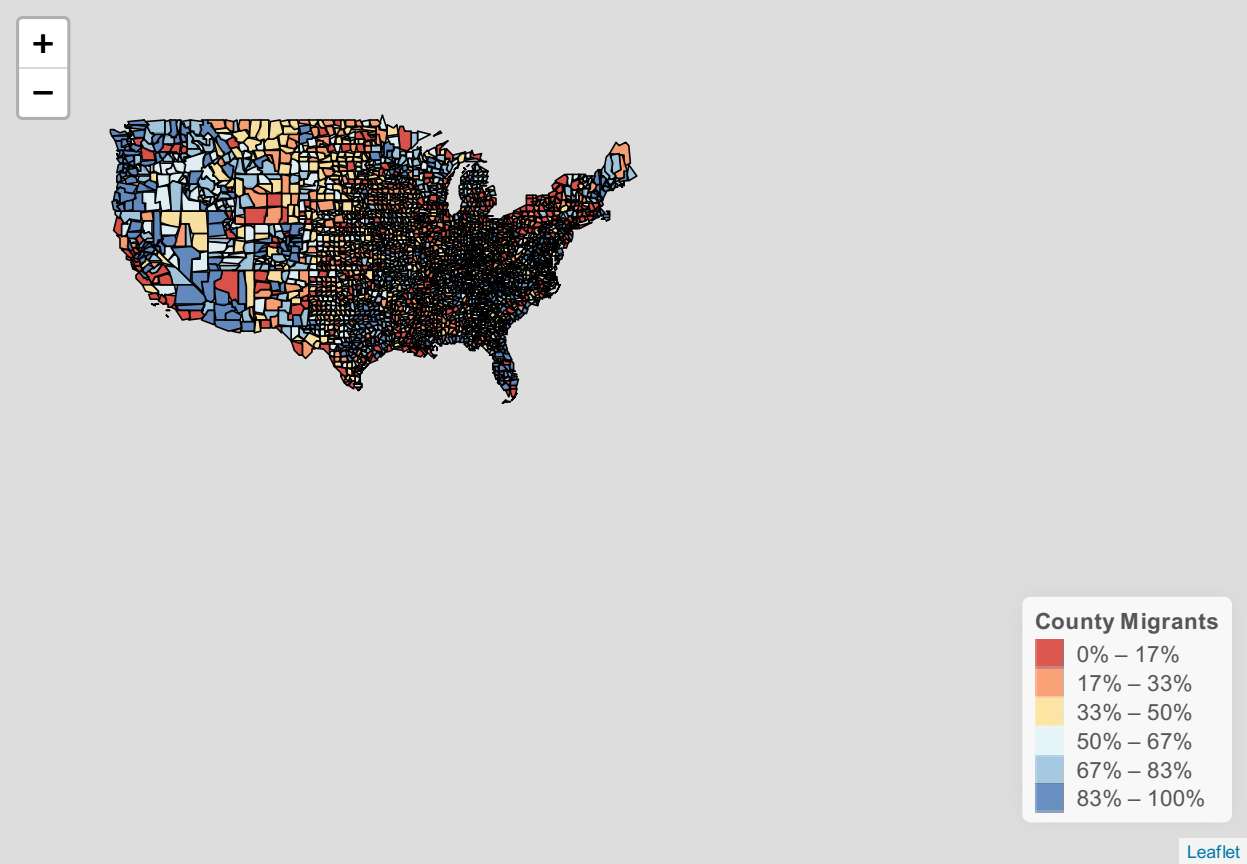
Step 4: Sample Net Migration Maps - 2018

County Level

```
counties18 <- filter(counties_merged, year=='2018')

pal <- colorQuantile('RdYlBu', counties18$net_migration, n=6)

leaflet(data = counties18) %>%
  addPolygons(fillColor = ~pal(net_migration),
              weight=1,
              opacity=1,
              color='black',
              fillOpacity=0.8,
              highlight = highlightOptions(
                weight = 2,
                color = 'white',
                fillOpacity=0.9,
                bringToFront=TRUE),
              label = counties18$net_migration) %>%
  addLegend(pal = pal, values = ~net_migration, opacity = 0.8,
            title='County Migrants', position='bottomright')
```

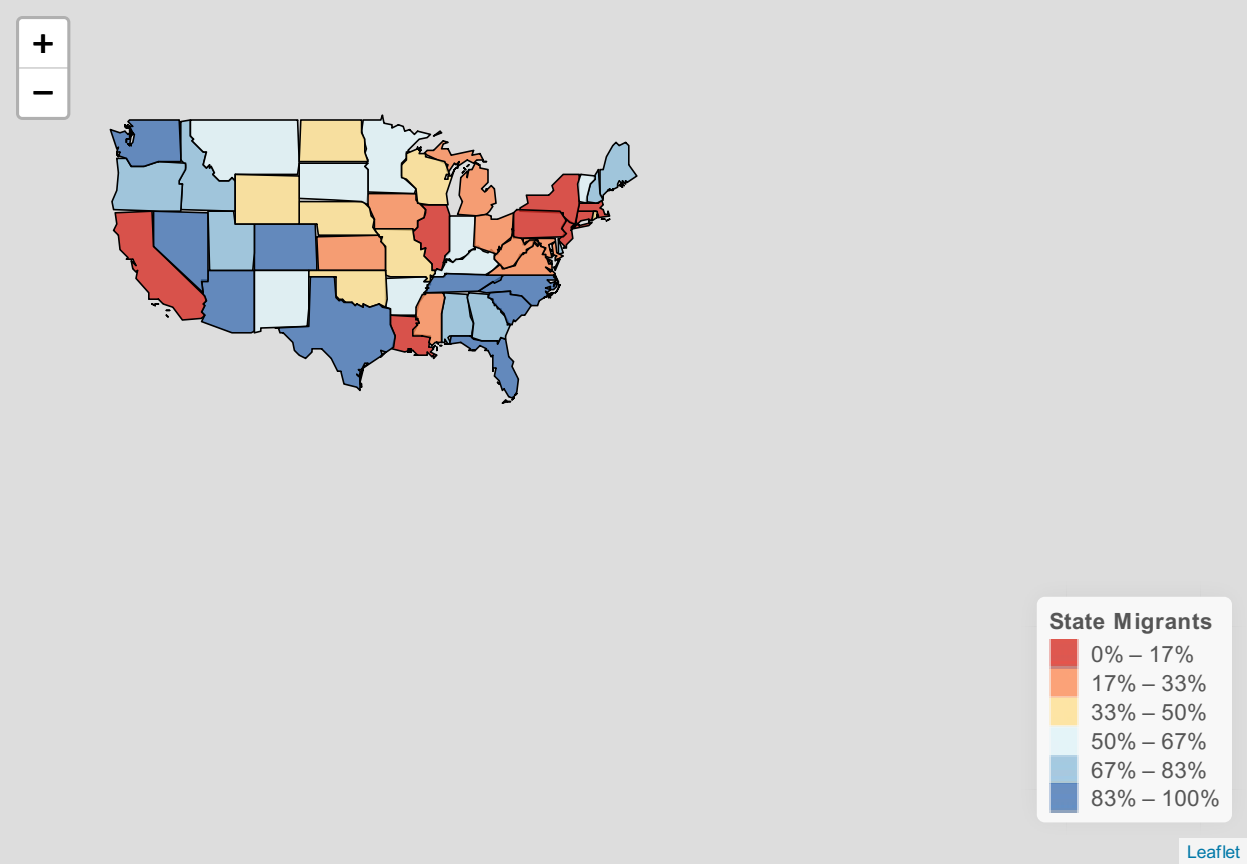


State Level

```
states18 <- filter(states_merged, year=='2018')

pal <- colorQuantile('RdYlBu', states18$net_migrants, n=6)

leaflet(data = states18) %>%
  addPolygons(fillColor = ~pal(net_migrants),
              weight=1,
              opacity=1,
              color='black',
              fillOpacity=0.8,
              highlight = highlightOptions(
                weight = 2,
                color = 'white',
                fillOpacity=0.9,
                bringToFront=TRUE),
              label = states18$net_migrants) %>%
  addLegend(pal = pal, values = ~net_migrants, opacity = 0.8,
            title='State Migrants', position='bottomright')
```

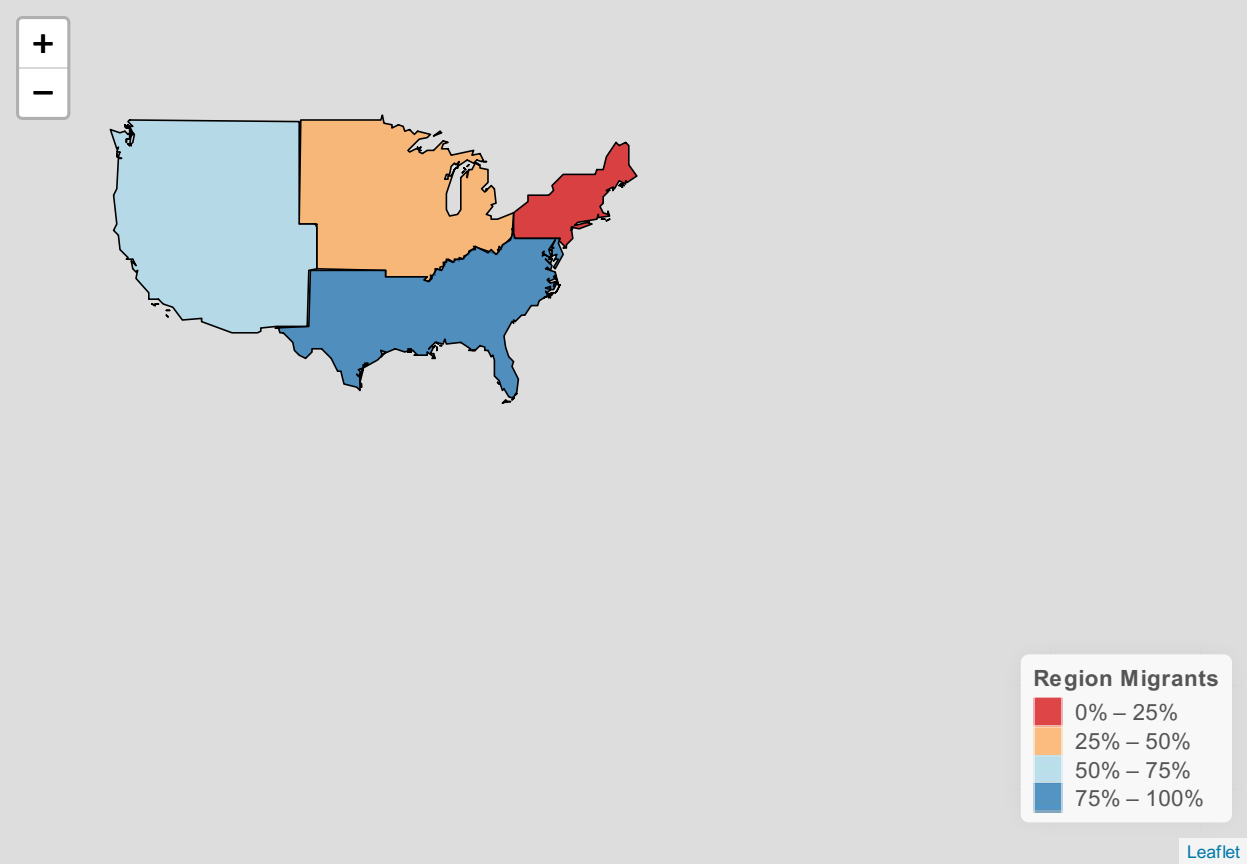


Region Level

```
regions18 <- filter(regions_merged, year=='2018')

pal <- colorQuantile('RdYlBu', regions18$net_migrants, n=4)

leaflet(data = regions18) %>%
  addPolygons(fillColor = ~pal(net_migrants),
              weight=1,
              opacity=1,
              color='black',
              fillOpacity=0.8,
              highlight = highlightOptions(
                weight = 2,
                color = 'white',
                fillOpacity=0.9,
                bringToFront=TRUE),
              label = regions18$net_migrants) %>%
  addLegend(pal = pal, values = ~net_migrants, opacity = 0.8,
            title='Region Migrants', position='bottomright')
```



Step 5: Origin-Destination Flows

Reload outflows data, preserving origin and destination values.

```
setwd("C:/Users/rache/Documents/GitHub/GIS_3/final_project")

#Make a more comprehensive crosswalk from the counties table.
xwalk_all = counties %>% st_drop_geometry() %>%
  select(GEOID, county, state, REGION)

#Define function
load_OD <- function(name, yr){
  df = read_csv(name)
  df['year'] <- yr

  #Eliminate Puerto Rico, Alaska, Hawaii, and non-state FIPS codes
  df = subset(df, y1_statefips!='00' & y1_statefips!='02' &
                y1_statefips!='15' & y2_statefips!="57" &
                y2_statefips!='58' & y2_statefips!='59' &
                y2_statefips!='72' & y2_statefips!="96" &
                y2_statefips!="98" & y2_statefips!="97" &
                y2_statefips!='02' & y2_statefips!='15')

  #Unite state and county FIPS codes to create 5-digit geoids
  df = df %>%
```

```
    unite('orig_id', y1_statefips:y1_countyfips, sep='', remove=TRUE) %>%
    unite('dest_id', y2_statefips:y2_countyfips, sep='', remove=TRUE) %>%
    select(orig_id, dest_id, year, n1) %>%
    rename(migrants = n1)

  #Join withe the crosswalk to get county, state, and region names for origin and destination
  df = left_join(df, xwalk_all, by=c('orig_id'='GEOID')) %>%
    rename(orig_cty=county, orig_st=state, orig_reg = REGION)
  df = left_join(df, xwalk_all, by=c('dest_id'='GEOID')) %>%
    rename(dest_cty=county, dest_st=state, dest_reg=REGION)
}

#Run function for each year
od_list <- list()
for(year in years){
  df = load_OD(glue('outflow{year}.csv'), year)
  od_list[[year]] <- df
}

#Merge the separate year dataframes
od_cty = do.call('rbind', od_list)

#Restructure the year column to be a 4-digit year beginning with '20'
substring(od_cty$year, 1, 2) <- '20'
```

Calculate Centroids

```
cent_coords <- function(df, col){

  #define the level (county, state, region) to use for calculation
  level <- enquo(col)
  grouped <- df %>% group_by(!!level) %>% summarize()

  #calculate centroid
  cent = st_centroid(grouped)

  #separate the lat/lon coordinates into separate columns
  coords <- do.call(rbind, st_geometry(cent)) %>%
    as_tibble() %>% setNames(c('lon', 'lat'))

  #merge coordinates with county ids
  new_df = merge(cent, coords, by='row.names')
  new_df = new_df %>% st_drop_geometry() %>%
    select((!!level), lon, lat)
}

cty_cent = cent_coords(counties, GEOID)
```

Merge flow data with centroids for origin and destination.

```
od_cty = left_join(od_cty, cty_cent, by=c('orig_id'='GEOID')) %>%
  rename(orig_lon=lon, orig_lat=lat)
od_cty = left_join(od_cty, cty_cent, by=c('orig_id'='GEOID')) %>%
  rename(dest_lon=lon, dest_lat=lat)
od_cty = subset(od_cty, orig_id!=dest_id)
```

Repeat data preparation for states.

```
od_st = od_cty %>% group_by(year, orig_st, dest_st) %>% summarize(migrants = sum(migrants))
st_cent = cent_coords(counties, state)

od_st = left_join(od_st, st_cent, by=c('orig_st'='state')) %>%
  rename(orig_lon=lon, orig_lat=lat)
od_st = left_join(od_st, st_cent, by=c('dest_st'='state')) %>%
  rename(dest_lon=lon, dest_lat=lat)
od_st = subset(od_st, orig_st!=dest_st)
```

Repeat data preparation for regions.

```
od_reg = od_cty %>% group_by(year, orig_reg, dest_reg) %>% summarize(migrants = sum(migrants))
reg_cent = cent_coords(counties, REGION)

od_reg = left_join(od_reg, reg_cent, by=c('orig_reg'='REGION')) %>%
  rename(orig_lon=lon, orig_lat=lat)
od_reg = left_join(od_reg, reg_cent, by=c('dest_reg'='REGION')) %>%
  rename(dest_lon=lon, dest_lat=lat)
od_reg = subset(od_reg, orig_reg!=dest_reg)

#Shift origin points down two degrees and left one degree to improve flow visibility
od_reg$orig_lat = od_reg$orig_lat-2
od_reg$orig_lon = od_reg$orig_lon-1
```

Step 6: Sample Flow Maps - 2018

County Level

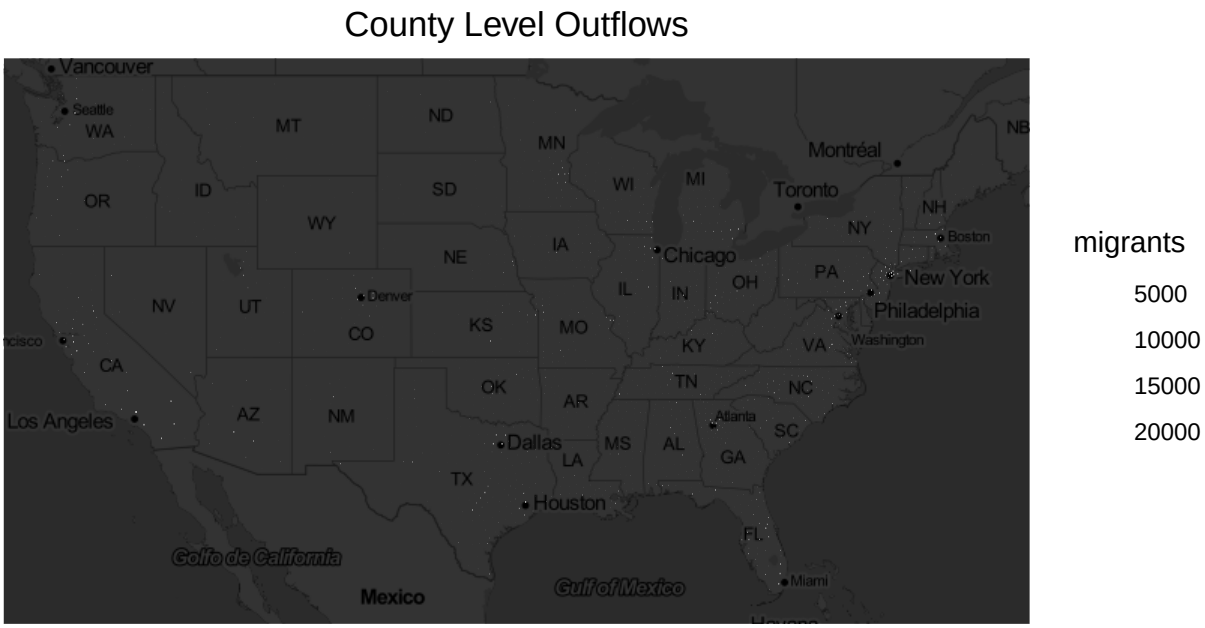
```
#define US bounding box and download basemap
us_bbox = c(left=-125.94, bottom=23.48, right=-65.84, top=49.61)
base = get_stamenmap(us_bbox, zoom=4, maptype='toner-lite')
```

```
## Source : http://tile.stamen.com/toner-lite/4/2/5.png
## Source : http://tile.stamen.com/toner-lite/4/3/5.png
## Source : http://tile.stamen.com/toner-lite/4/4/5.png
## Source : http://tile.stamen.com/toner-lite/4/5/5.png
## Source : http://tile.stamen.com/toner-lite/4/2/6.png
## Source : http://tile.stamen.com/toner-lite/4/3/6.png
## Source : http://tile.stamen.com/toner-lite/4/4/6.png
## Source : http://tile.stamen.com/toner-lite/4/5/6.png
```

```
cty18 <- filter(od_cty, year=='2018')

ggmap(base, darken=0.8) +
  geom_segment(data=cty18,
               aes(y=orig_lat, x=orig_lon, yend=dest_lat, xend=dest_lon, alpha=migrants),
               size=0.3, color='white') +
  scale_alpha_continuous(range=c(0.01, 0.5)) +
  theme_minimal() +
  theme(axis.text=element_blank(),
```

```
        axis.title=element_blank(),
        axis.ticks=element_blank(),
        plot.title = element_text(hjust=0.5)) +
  ggtitle('County Level Outflows')
```

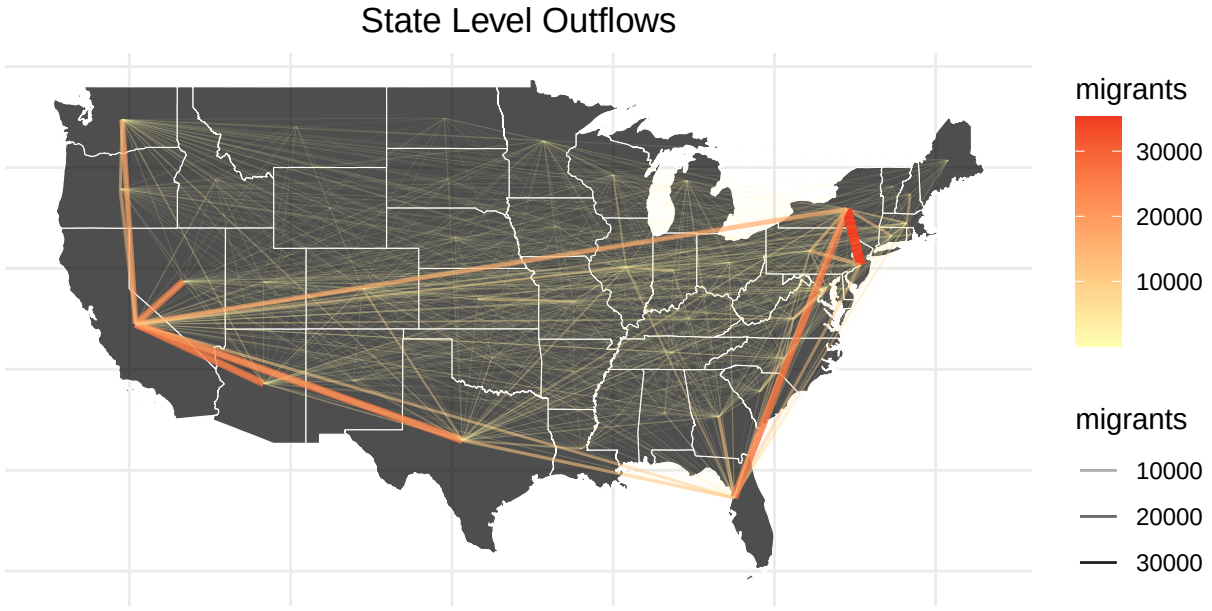


State Level

```
state_base = counties %>% group_by(state) %>% summarize()

st18 = filter(od_st, year=='2018')

ggplot(data=state_base) +
  geom_sf(color='white', alpha=0.7, size = 0.2, fill='black') +
  geom_segment(data=st18,
               aes(y=orig_lat, x=orig_lon, yend=dest_lat, xend=dest_lon,
                   alpha=migrants, color=migrants), size=st18$migrants/20000) +
  scale_alpha_continuous(range=c(0.05, 1)) +
  scale_colour_gradient(low='#ffffb2', high='#f03b20') +
  theme_minimal() +
  theme(axis.text=element_blank(),
        axis.title=element_blank(),
        axis.ticks=element_blank(),
        plot.title = element_text(hjust=0.5)) +
  ggtitle('State Level Outflows')
```



Region Level

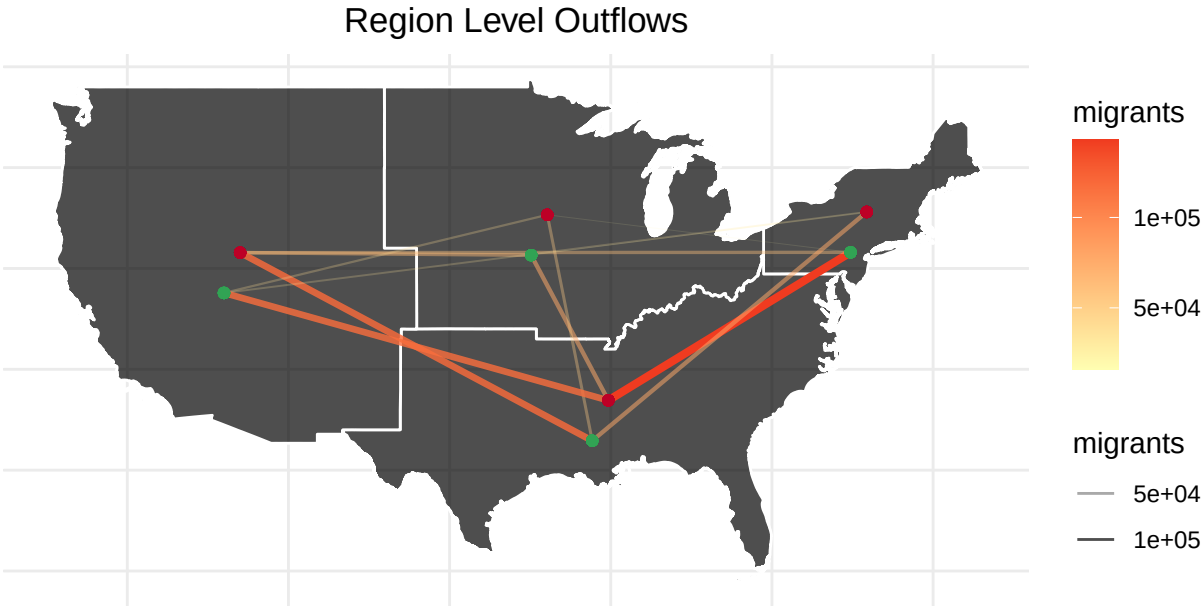
```
reg_base = counties %>% group_by(REGION) %>% summarize()

reg18 = filter(od_reg, year=='2018')

ggplot(data=reg_base) +
  geom_sf(color='white', alpha=0.7, fill = 'black') +
  geom_segment(data=reg18,
               aes(y=orig_lat, x=orig_lon, yend=dest_lat, xend=dest_lon,
                   alpha=migrants, color=migrants),
               size=reg18$migrants/100000) +
  scale_colour_gradient(low='#ffffb2', high='#f03b20') +
  geom_point(data=od_reg, aes(orig_lon, orig_lat), color='#31a354', size=1.5, name='origin') +
  geom_point(data=od_reg, aes(dest_lon, dest_lat), color='#bd0026', size=1.5, name='destination') +
  theme_minimal() +
  theme(axis.text=element_blank(),
        axis.title=element_blank(),
        axis.ticks=element_blank(),
        plot.title = element_text(hjust=0.5)) +
  ggtitle('Region Level Outflows')
```

```
## Warning: Ignoring unknown parameters: name

## Warning: Ignoring unknown parameters: name
```


Region Level Outflows
migrants
1e+05
5e+04
migrants
5e+04
1e+05