VisuallyExploringEmploymentData

Jingan Qu

March 27, 2017

Introduction

This project will introduce you to the US employment data provided by the Bureau of Labor Statistics (BLS) of the United States government. The BLS is the federal agency responsible for measuring labor market activity and working conditions and prices in the US economy. Its mission is the collection, analysis, and dissemination of essential economic information to support public and private decision-making. In this project, we will use the aggregate annual data on employment and pay, stratified by geography and industry. This data can be downloaded as a compressed comma-separated value (csv) file at 2015 QCEW data, which contains the single file 2015 (the last available full year of data). annual.singlefile.csv . This file has 38 columns and about 3.5 million rows.

Preparing for analysis

```
#load the package
ipak <- function(pkg){
  new.pkg <- pkg[!(pkg %in% installed.packages()[, "Package"])]
  if (length(new.pkg))
    install.packages(new.pkg, dependencies = TRUE)
  sapply(pkg, require, character.only = TRUE)
}
packages <- c("data.table","plyr","dplyr","ggplot2","stringr","maps",
              "bit64","RColorBrewer","gdata","sqldf","choroplethr")
ipak(packages)
```

```
## Warning in doTryCatch(return(expr), name, parentenv, handler): unable to load shared object '/Library
##   dlopen(/Library/Frameworks/R.framework/Resources/modules//R_X11.so, 6): Library not loaded: /opt/X
##   Referenced from: /Library/Frameworks/R.framework/Resources/modules//R_X11.so
##   Reason: image not found
```

```
##   data.table         plyr        dplyr      ggplot2      stringr
##         TRUE         TRUE         TRUE         TRUE         TRUE
##         maps        bit64 RColorBrewer        gdata        sqldf
##         TRUE         TRUE         TRUE         TRUE         TRUE
##  choroplethr
##         TRUE
```

```
# set path
setwd("/Users/jinganqu/Dropbox/Github/PracticeDataScience/chapter5")
```

Importing employment data into R

```
ann2015 <- fread('./data/2015.annual.singlefile.csv', sep=',',
                 colClasses=c('character', 'integer', 'integer', 'integer',
                              'integer', 'integer', 'character',rep('integer',31)))
```

```
##
Read 0.0% of 3576934 rows
Read 10.3% of 3576934 rows
Read 20.4% of 3576934 rows
Read 30.2% of 3576934 rows
Read 40.0% of 3576934 rows
Read 49.8% of 3576934 rows
Read 59.8% of 3576934 rows
Read 69.6% of 3576934 rows
Read 79.7% of 3576934 rows
Read 90.0% of 3576934 rows
Read 3576934 rows and 38 (of 38) columns from 0.475 GB file in 00:00:15
```

We just need the first 15 colomns and convert data types to integer

```
ann2015 <- ann2015[,1:15,with = FALSE]

index <- names(ann2015)[c(2:6,8)]
#convert class
for (i in index) {
  set(ann2015, j=i, value = as.integer(ann2015[[i]]))

}
```

Display the first few lines

```
head(ann2015)
```

```
##    area_fips own_code industry_code agglvl_code size_code year qtr
## 1:     01000        0            10          50         0 2015   A
## 2:     01000        1            10          51         0 2015   A
## 3:     01000        1           102          52         0 2015   A
## 4:     01000        1          1021          53         0 2015   A
## 5:     01000        1          1022          53         0 2015   A
## 6:     01000        1          1023          53         0 2015   A
##    disclosure_code annual_avg_estabs annual_avg_emplvl total_annual_wages
## 1:              NA            119251           1890340        83690526044
## 2:              NA              1185             53014         4199053853
## 3:              NA              1185             53014         4199053853
## 4:              NA               590             11257          741536011
## 5:              NA                 2                12             385630
## 6:              NA                18               151           12141072
##    taxable_annual_wages annual_contributions annual_avg_wkly_wage
## 1:          14289391483            295276826                  851
## 2:                    0                    0                 1523
## 3:                    0                    0                 1523
## 4:                    0                    0                 1267
## 5:                    0                    0                  627
## 6:                    0                    0                 1549
##    avg_annual_pay
## 1:          44273
## 2:          79206
## 3:          79206
## 4:          65876
## 5:          32588
## 6:          80538
```

Obstaining and merging additional data

Finde additional data on the BLS WEBSITE at here under the header Associated Codes and Titles.

Download these csv files (Industries, Areas, Ownerships, Size Classes, Aggregation Levels) to computer, which are used for merging data.

Import these data files into R:

```
for (u in c('agglevel','area','industry', 'ownership','size')) {
  assign(u, read.csv(paste('data/',u,'_titles.csv',sep=''), stringsAsFactors = F
                     ))

}
```

Each of these datasets has exactly one variable in common with our original data (ann2015). So join four of these datasets with ann2015 now.

```
codes <- c('agglevel','industry','area', 'ownership','size')
ann2015full <- ann2015

for (i in 1:length(codes)) {
  tmp <- eval(parse(text=codes[i]))
  code <- names(tmp)[1]
  if (class(ann2015full[[code]]) != class(tmp[,1])) {
    c <- class(tmp[,1])
    eval(parse(text = paste0("set(ann2015full, j=code, value = as.",
                             c, "(ann2015full[[code]]))")))
  }
  eval(parse(text=paste('ann2015full <- left_join(ann2015full,',codes[i],')', sep='')))

}
```

Display the first few rows again:

```
head(ann2015full)
```

```
##   area_fips own_code industry_code agglvl_code size_code year qtr
## 1     01000        0            10          50         0 2015   A
## 2     01000        1            10          51         0 2015   A
## 3     01000        1           102          52         0 2015   A
## 4     01000        1          1021          53         0 2015   A
## 5     01000        1          1022          53         0 2015   A
## 6     01000        1          1023          53         0 2015   A
##   disclosure_code annual_avg_estabs annual_avg_emplvl total_annual_wages
## 1              NA            119251           1890340        83690526044
## 2              NA              1185             53014         4199053853
## 3              NA              1185             53014         4199053853
## 4              NA               590             11257          741536011
## 5              NA                 2                12             385630
## 6              NA                18               151           12141072
##   taxable_annual_wages annual_contributions annual_avg_wkly_wage
## 1          14289391483            295276826                  851
## 2                    0                    0                 1523
## 3                    0                    0                 1523
## 4                    0                    0                 1267
## 5                    0                    0                  627
## 6                    0                    0                 1549
##   avg_annual_pay                                   agglvl_title
## 1          44273                          State, Total Covered
## 2          79206          State, Total -- by ownership sector
## 3          79206      State, by Domain -- by ownership sector
## 4          65876 State, by Supersector -- by ownership sector
## 5          32588 State, by Supersector -- by ownership sector
## 6          80538 State, by Supersector -- by ownership sector
##                           industry_title          area_title
## 1                Total, all industries Alabama -- Statewide
## 2                Total, all industries Alabama -- Statewide
## 3                    Service providing Alabama -- Statewide
## 4 Trade, transportation, and utilities Alabama -- Statewide
## 5                          Information Alabama -- Statewide
## 6                 Financial activities Alabama -- Statewide
##            own_title              size_title
## 1      Total Covered All establishment sizes
## 2 Federal Government All establishment sizes
## 3 Federal Government All establishment sizes
## 4 Federal Government All establishment sizes
## 5 Federal Government All establishment sizes
## 6 Federal Government All establishment sizes
```

Adding geographical information

First take a look at the area data:

```
head(area)
```

```
##   area_fips                                    area_title
## 1     US000                                    U.S. TOTAL
## 2     USCMS     U.S. Combined Statistical Areas (combined)
## 3     USMSA U.S. Metropolitan Statistical Areas (combined)
## 4     USNMS  U.S. Nonmetropolitan Area Counties (combined)
## 5     01000                          Alabama -- Statewide
## 6     01001                       Autauga County, Alabama
```

Capitalize all the namesm according to the conventions. Here is a small function to do this:

```
simpleCap <- function(x) {
  if (!is.na(x)) {
    s <- strsplit(x, " ")[[1]]

    result <- paste(toupper(substring(s,1,1)), substring(s,2),
                  sep='', collapse = " ")
    return(result)

  } else {return(NA)}

}
```

The maps package contains two datasets that we will use; they are county.fips and state.fips. First do some transformations because there is missing a leading 0 on the left for some of the codes. All the codes in our data comprise five digits.

```r
data("county.fips")
head(county.fips)
```

```
##   fips        polyname
## 1 1001 alabama,autauga
## 2 1003 alabama,baldwin
## 3 1005 alabama,barbour
## 4 1007    alabama,bibb
## 5 1009  alabama,blount
## 6 1011 alabama,bullock
```

The stringr package will help us out here:

```r
county.fips$fips <- str_pad(county.fips$fips , width = 5, pad = 0)
```

Separate the county names from the polyname column in county.fips. And get the state names form state.fips

```r
county.fips$polyname <- as.character(county.fips$polyname)
county.fips$polyname <- sapply(
  gsub("[a-z\ ]+,([a-z\ ]+)", "\\1", county.fips$polyname),
  simpleCap)

county.fips <- county.fips[!duplicated(county.fips$fips),]
```

The state.fips data invovles a lot of details:

```r
data("state.fips")
```

Show the first few row of states.fips:

```r
head(state.fips)
```

```
##   fips ssa region division abb    polyname
## 1    1   1      3        6  AL     alabama
## 2    4   3      4        8  AZ     arizona
## 3    5   4      3        7  AR    arkansas
## 4    6   5      4        9  CA  california
## 5    8   6      4        8  CO    colorado
## 6    9   7      1        1  CT connecticut
```

Again pad the fips column with a 0, if necessary, so that they have two digits, and capitalize the state names from polyname to create a new state column.

```r
state.fips$fips <- str_pad(state.fips$fips, width=2, pad="0")
state.fips$state <- as.character(state.fips$polyname)
state.fips$state <- gsub("([a-z\ ]+):[a-z\ \\']+",'\\1',state.fips$state)
state.fips$state <- sapply(state.fips$state, simpleCap)
```

Make sure rows are unique.

```r
mystatefips <- unique(state.fips[, c("fips", "abb", "state")])
```

Filter the data to look only at these states:

```r
lower48 <- setdiff(unique(state.fips$state),
                   c('Hawaii','Alaska'))
```

Finally, put all this information together into a single dataset, myarea:

```r
myarea <- merge(area, county.fips, by.x='area_fips',by.y='fips', all.x=T)
myarea$state_fips <- substr(myarea$area_fips, 1,2)
myarea <- merge(myarea, mystatefips,by.x='state_fips',by.y='fips', all.x=T)
```

Lastly, join the geographical information with our dataset, and filter it to keep only data on the lower 48 states:

```r
ann2015full_1 <- left_join(ann2015full, myarea, by = "area_fips")
ann2015full <- filter(ann2015full_1, state %in% lower48)
```

Store the final data set in an R data file (rda) on disk.

```r
save(ann2015full, file = "data/ann2015full.rda", compress = T)
```

Extracting state- and county-level wage and employment information

First extract data from ann2014full at the state-level.

Look at the aggregate state-level data. A peek at agglevel tells us that the code for the level of data that we want is 50. Also, look at the average annual pay (avg_annual_pay) and the average annual employment level (annual_avg_emplvl), and not the other variables:

```r
d.state <- ann2015full %>%
  filter(agglvl_code==50) %>%
  select(state, avg_annual_pay,
                 annual_avg_emplvl)
```

Create two new variables, wage and empquantile, which discretizes the pay and employment variables.

```r
d.state$wage <- cut(d.state$avg_annual_pay,
                    quantile(d.state$avg_annual_pay, c(seq(0,.8, by=.2), .9, .95,
                                                       .99, 1)),
                    include.lowest=TRUE)

d.state$empquantile <- cut(d.state$annual_avg_emplvl,
                           quantile(d.state$annual_avg_emplvl,
                                    c(seq(0,.8,by=.2),.9,.95,.99,1)),
                           include.lowest=TRUE)
```

Label the variable wage and empquantile

```r
x <- quantile(d.state$avg_annual_pay, c(seq(0,.8,by=.2), .9,
                                        .95, .99, 1))
xx <- paste(round(x/1000),'K',sep='')
Labs <- paste(xx[-length(xx)],xx[-1],sep='-')
levels(d.state$wage) <- Labs

x <- quantile(d.state$annual_avg_emplvl,
              c(seq(0,.8,by=.2),.9, .95, .99, 1))
xx <- ifelse(x>1000, paste(round(x/1000),'K',sep=''),
             round(x))
Labs <- paste(xx[-length(xx)],xx[-1],sep='-')
levels(d.state$empquantile) <- Labs
```

We repeat this process at the county-level. We will find that the appropriate aggregation level code is 70 (agglvl_code==70). Everything else will be the same. Let's try to be a bit smarter this time around. First of all, we will discretize our variables the same way, and then change the labels to match. A function might be a good idea! The following command lines depict this:

```
Discretize <- function(x, breaks=NULL){
  if(is.null(breaks)){
    breaks <- quantile(x, c(seq(0,.8,by=.2),.9, .95, .99, 1))
    if (sum(breaks==0)>1) {
      temp <- which(breaks==0, arr.ind=TRUE)
      breaks <- breaks[max(temp):length(breaks)]
    }
  }
  x.discrete <- cut(x, breaks, include.lowest=TRUE)
  breaks.eng <- ifelse(breaks > 1000,
                       paste0(round(breaks/1000),'K'),
                       round(breaks))
  Labs <- paste(breaks.eng[-length(breaks.eng)], breaks.eng[-1],
                sep='-')
  levels(x.discrete) <- Labs
  return(x.discrete)
}
```

```
d.cty <- filter(ann2015full, agglvl_code==70) %>%
  select(state,polyname,abb, avg_annual_pay, annual_avg_emplvl) %>%
  mutate(wage=Discretize(avg_annual_pay),
         empquantile=Discretize(annual_avg_emplvl))
```

Visualizing geographical distributions of pay

We first need to get some data on the map itself. The ggplot2 package provides a convenient function, map_data, to extract this from data bundled in the maps package:

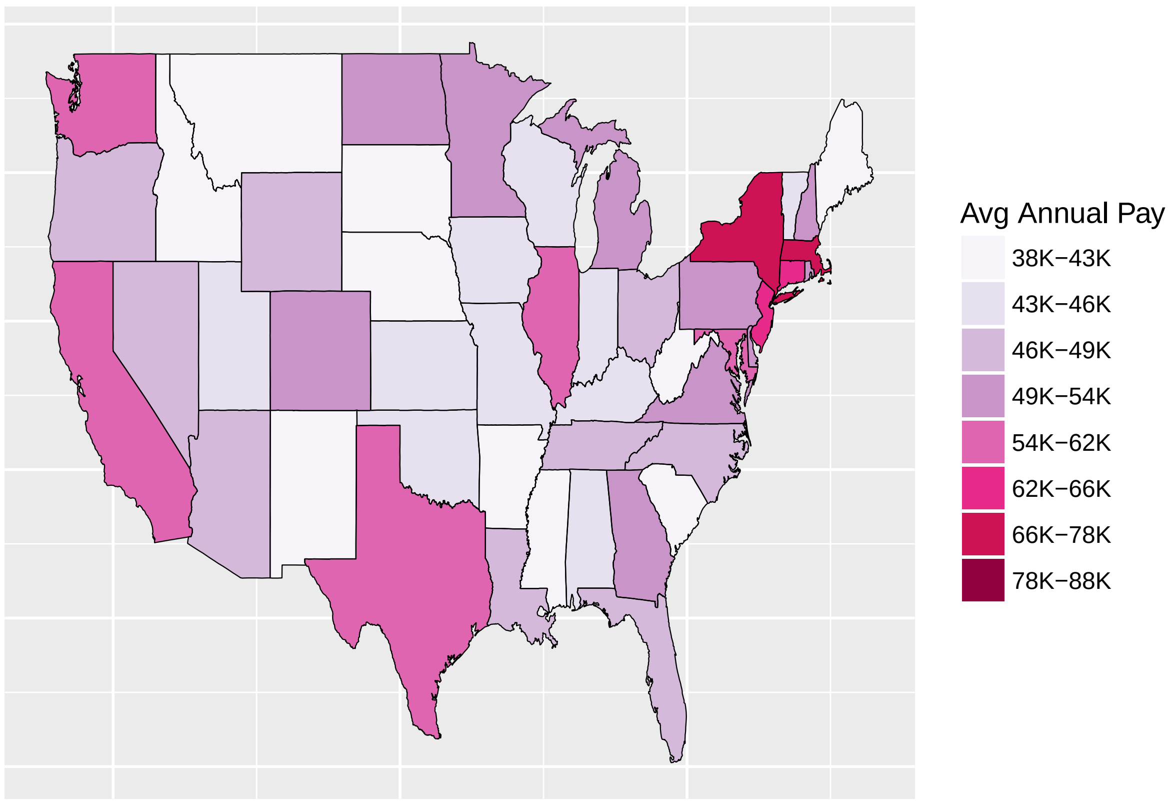
```
state_df <- map_data("state")
county_df <- map_data("county")
```

We now do a bit of transforming to make this data conform to our data:

```
transform_mapdata <- function(x) {
  names(x)[5:6] <- c('state', 'county')
  for (u in c('state', 'county')) {
    x[,u] <- sapply(x[,u],simpleCap)
  }
  return(x)
}

state_df <- transform_mapdata(state_df)
county_df <- transform_mapdata(county_df)
```

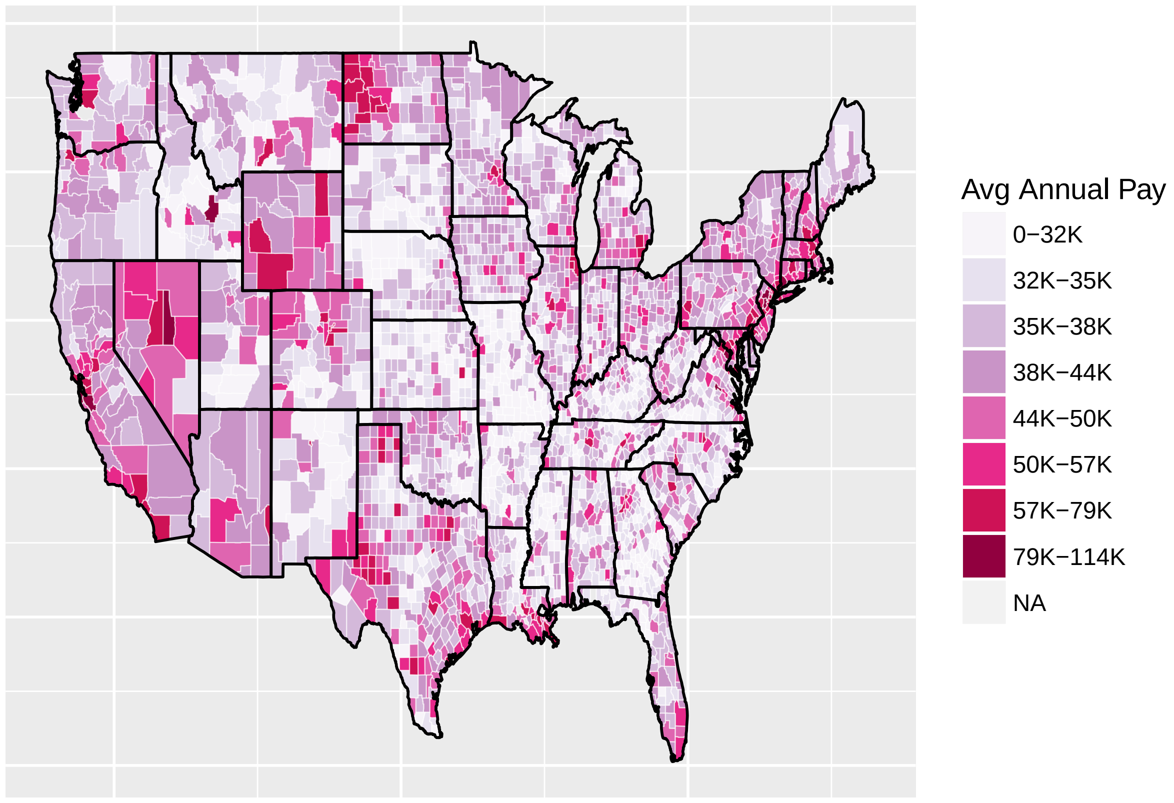
The data.frame objects, state_df and county_df, contain the latitude and longitude of points. These are our primary graphical data and need to be joined with the data we created in the previous recipe, which contains what is in effect the color information for the map:

```
chor1 <- left_join(state_df, d.state, by='state')

ggplot(chor1, aes(long,lat,group=group)) +
  geom_polygon(aes(fill=wage))+geom_path(color='black',size=0.2) +
  scale_fill_brewer(palette='PuRd') +
  theme(axis.text.x=element_blank(),
        axis.text.y=element_blank(), axis.ticks.x=element_blank(),
        axis.ticks.y=element_blank())  +
  labs(x='',y='', fill='Avg Annual Pay')
```



We can similarly create a visualization of the average annual pay by county, which will give us a much more granular information about the geographical distribution of wages:

```
d.cty <- rename(d.cty, county = polyname)
chor2 <- left_join(county_df, d.cty)

ggplot(chor2, aes(long,lat, group=group))+
  geom_polygon(aes(fill=wage))+
  geom_path( color='white',alpha=0.5,size=0.2)+
  geom_polygon(data=state_df, color='black',fill=NA)+
  scale_fill_brewer(palette='PuRd')+
  labs(x='',y='', fill='Avg Annual Pay')+
  theme(axis.text.x=element_blank(), axis.text.y=element_blank(),
        axis.ticks.x=element_blank(), axis.ticks.y=element_blank())
```



Exploring where the jobs are, by industry

The employment dataset has more granular data, divided by public/private sectors and types of jobs. The types of jobs in this data follow a hierarchical coding system called North American Industry Classification System (NIACS). Now consider four particular industries and look at visualizing the geographical distribution of employment in these industries, restricted to private sector jobs.

We will look at four industrial sectors: * Agriculture, forestry, fishing, and hunting (NIACS 11) * Mining, quarrying, and oil and gas extraction (NIACS 21) * Finance and insurance (NIACS 52) * Professional and technical services (NIACS 54)

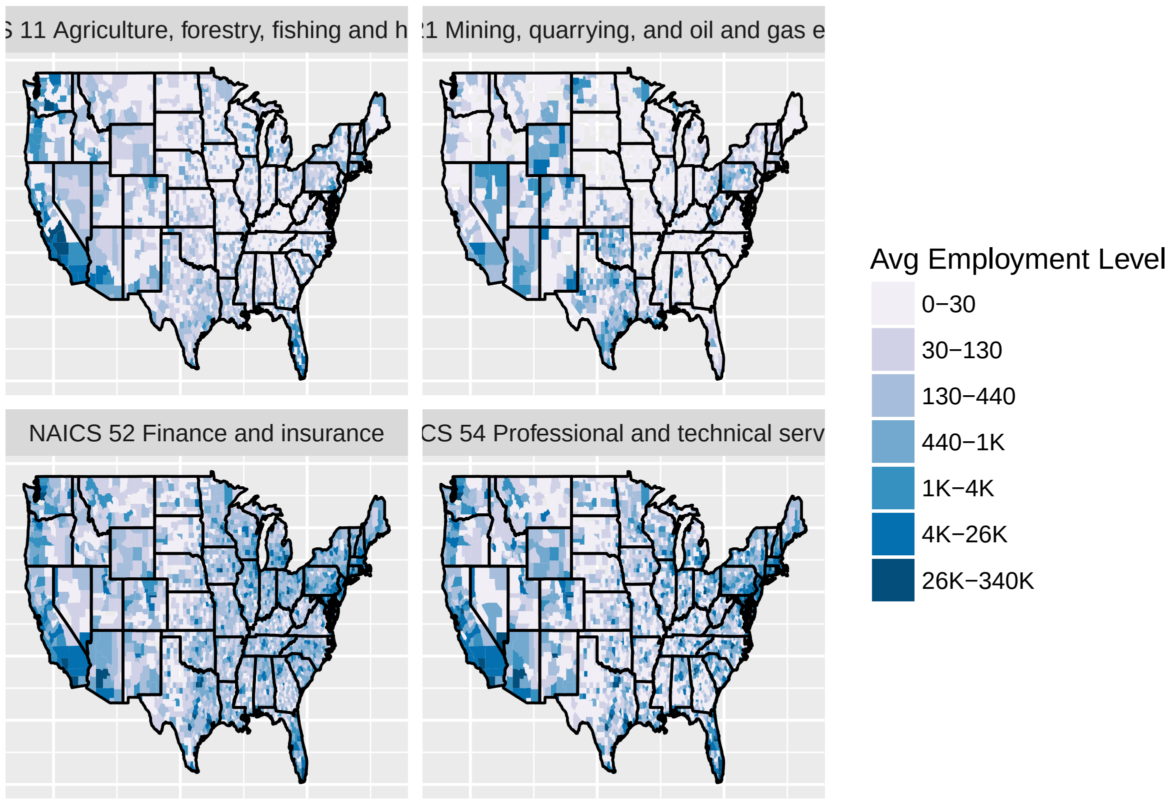
Create a subset of the employment data, including the data for industrial sectors, but restricting it to the private sector, by performing the following steps:

Start by filtering the data by the conditions we are imposing on the industry and private sectors, and keep only relevant variables:

```
d.sectors <- filter(ann2015full, industry_code %in%
                      c(11,21,54,52),
                    own_code==5, # Private sector
                    agglvl_code == 74 # county-level
                    ) %>%
  rename(county = polyname) %>%
  select(state, county, industry_code, own_code, agglvl_code,
         industry_title, own_title, avg_annual_pay,
         annual_avg_emplvl) %>%
  mutate(wage=Discretize(avg_annual_pay),
         emplevel=Discretize(annual_avg_emplvl)) %>%
  filter(!is.na(industry_code))
```

Create the visualization using ggplot2. This visualization will be an array of four panels, one for each industrial sector. Each panel will have a county-level map of the US, with colors signifying the level of employment in each county in 2015 in each particular industry.

```r
chor3 <- left_join(county_df, d.sectors) %>%
  filter(!is.na(industry_code))

ggplot(chor3, aes(long,lat,group=group))+
  geom_polygon(aes(fill=emplevel))+
  geom_polygon(data=state_df, color='black',fill=NA)+
  scale_fill_brewer(palette='PuBu')+
  facet_wrap(~industry_title, ncol=2, as.table=T)+
  labs(fill='Avg Employment Level',x='',y='')+
  theme(axis.text.x=element_blank(),
        axis.text.y=element_blank(),
        axis.ticks.x=element_blank(),
        axis.ticks.y=element_blank())
```



S 11 Agriculture, forestry, fishing and h
21 Mining, quarrying, and oil and gas e
NAICS 52 Finance and insurance
CS 54 Professional and technical serv
Avg Employment Level
0−30
30−130
130−440
440−1K
1K−4K
4K−26K
26K−340K

Animating maps for a geospatial time series

The QCEW site provides data from 2000 to 2015. We will look at the overall average annual pay by county for each of these years and create an animation that displays the changes in the pay pattern over this period.

Import the data for all the years from 2000 through 2015 and extract data for the county-level (agglvl_code==70) average annual pay (avg_annual_pay) for each county, plot it, and then string the pay values together in an animation. Since we basically need to do the same things for each year's data, we can do this in a for loop, and we can create functions to encapsulate the repeated actions. We start by writing code for a single year.

Import the data from the ZIP file. In reality, the file names are of the pattern 2000_annual_singlefile.zip, and the CSV files in them are of the pattern 2000.annual.singlefile.csv. We will use the common patterns in the ZIP and CSV files in our code to automate the process.

Encapsulate the actions of unzipping data, loading data and joining data in one function, the input of the function is year.

```
get_data <- function(zipfile) {

  #unzip data
  unzip(file.path('data',zipfile),exdir = 'data')

  #load data
  csvfile <- gsub('zip','csv',zipfile)
  csvfile <- gsub('_', '.', csvfile)
  raw_data <- fread(file.path('data',csvfile))

  #join the data with area data
  dat <- left_join(raw_data, myarea) %>%
    filter(agglvl_code==70) %>%
    rename(county = polyname) %>%
    select(state, county, avg_annual_pay)


  return(dat)
}
```

We now have to repeat this for each years and store the data. For this type of data, a list object usually makes sense:

```
files <- dir("data", pattern = "annual_singlefile.zip") #files names
n <- length(files)
data_list <- vector('list',n) #Initialize the list
for (i in 1:n) {
  data_list[[i]] <- get_data(files[i])

  names(data_list)[i] <- substr(files[i], 1, 4)
}
```

```
##
Read 47.0% of 999212 rows
Read 75.1% of 999212 rows
Read 999212 rows and 38 (of 38) columns from 0.158 GB file in 00:00:04
##
Read 0.0% of 3579633 rows
Read 12.3% of 3579633 rows
Read 24.9% of 3579633 rows
Read 38.3% of 3579633 rows
Read 50.8% of 3579633 rows
Read 63.4% of 3579633 rows
Read 75.7% of 3579633 rows
Read 87.7% of 3579633 rows
Read 99.7% of 3579633 rows
Read 3579633 rows and 38 (of 38) columns from 0.421 GB file in 00:00:11
##
Read 0.0% of 3580607 rows
Read 10.3% of 3580607 rows
Read 19.8% of 3580607 rows
Read 30.4% of 3580607 rows
Read 41.9% of 3580607 rows
Read 53.1% of 3580607 rows
Read 64.5% of 3580607 rows
Read 75.4% of 3580607 rows
Read 84.9% of 3580607 rows
Read 94.1% of 3580607 rows
Read 3580607 rows and 38 (of 38) columns from 0.470 GB file in 00:00:14
##
Read 0.0% of 3586768 rows
Read 10.3% of 3586768 rows
Read 20.4% of 3586768 rows
Read 30.7% of 3586768 rows
Read 42.1% of 3586768 rows
Read 53.3% of 3586768 rows
Read 64.4% of 3586768 rows
Read 75.6% of 3586768 rows
Read 86.7% of 3586768 rows
Read 97.6% of 3586768 rows
Read 3586768 rows and 38 (of 38) columns from 0.473 GB file in 00:00:12
##
Read 2.2% of 3591082 rows
Read 11.4% of 3591082 rows
Read 20.9% of 3591082 rows
Read 30.6% of 3591082 rows
Read 41.8% of 3591082 rows
Read 53.2% of 3591082 rows
Read 63.8% of 3591082 rows
Read 74.1% of 3591082 rows
Read 84.1% of 3591082 rows
Read 94.1% of 3591082 rows
Read 3591082 rows and 38 (of 38) columns from 0.478 GB file in 00:00:12
##
Read 0.0% of 3603234 rows
Read 10.3% of 3603234 rows
Read 20.8% of 3603234 rows
Read 31.9% of 3603234 rows
Read 42.5% of 3603234 rows
Read 52.7% of 3603234 rows
Read 63.3% of 3603234 rows
Read 73.3% of 3603234 rows
Read 83.0% of 3603234 rows
Read 93.0% of 3603234 rows
Read 3603234 rows and 38 (of 38) columns from 0.477 GB file in 00:00:12
##
Read 1.7% of 3613495 rows
Read 12.2% of 3613495 rows
Read 22.7% of 3613495 rows
Read 32.9% of 3613495 rows
Read 42.9% of 3613495 rows
Read 52.6% of 3613495 rows
Read 62.8% of 3613495 rows
Read 72.8% of 3613495 rows
Read 83.0% of 3613495 rows
Read 92.7% of 3613495 rows
Read 3613495 rows and 38 (of 38) columns from 0.478 GB file in 00:00:12
##
Read 4.2% of 3609616 rows
Read 15.0% of 3609616 rows
Read 26.3% of 3609616 rows
Read 37.1% of 3609616 rows
Read 47.1% of 3609616 rows
Read 57.3% of 3609616 rows
Read 67.6% of 3609616 rows
Read 77.8% of 3609616 rows
Read 88.1% of 3609616 rows
Read 98.1% of 3609616 rows
Read 3609616 rows and 38 (of 38) columns from 0.479 GB file in 00:00:12
##
Read 4.1% of 3622718 rows
Read 15.2% of 3622718 rows
Read 25.9% of 3622718 rows
Read 36.2% of 3622718 rows
Read 46.7% of 3622718 rows
Read 57.1% of 3622718 rows
Read 67.4% of 3622718 rows
Read 77.6% of 3622718 rows
Read 88.1% of 3622718 rows
Read 98.0% of 3622718 rows
Read 3622718 rows and 38 (of 38) columns from 0.481 GB file in 00:00:12
##
Read 1.7% of 3608838 rows
Read 10.8% of 3608838 rows
Read 21.6% of 3608838 rows
Read 32.1% of 3608838 rows
Read 42.1% of 3608838 rows
Read 52.9% of 3608838 rows
Read 63.2% of 3608838 rows
Read 74.0% of 3608838 rows
Read 84.5% of 3608838 rows
Read 94.8% of 3608838 rows
Read 3608838 rows and 38 (of 38) columns from 0.481 GB file in 00:00:12
##
Read 4.2% of 3600171 rows
Read 15.0% of 3600171 rows
Read 26.1% of 3600171 rows
Read 37.2% of 3600171 rows
Read 48.3% of 3600171 rows
Read 58.9% of 3600171 rows
Read 69.4% of 3600171 rows
Read 80.0% of 3600171 rows
Read 91.1% of 3600171 rows
Read 3600171 rows and 38 (of 38) columns from 0.478 GB file in 00:00:11
##
Read 4.5% of 3560725 rows
Read 15.4% of 3560725 rows
Read 26.7% of 3560725 rows
Read 37.9% of 3560725 rows
Read 48.6% of 3560725 rows
Read 59.3% of 3560725 rows
Read 69.9% of 3560725 rows
Read 80.3% of 3560725 rows
Read 91.3% of 3560725 rows
Read 3560725 rows and 38 (of 38) columns from 0.472 GB file in 00:00:11
##
Read 4.8% of 3556289 rows
Read 16.0% of 3556289 rows
Read 27.3% of 3556289 rows
Read 38.5% of 3556289 rows
Read 49.2% of 3556289 rows
Read 59.9% of 3556289 rows
Read 70.6% of 3556289 rows
Read 81.3% of 3556289 rows
Read 92.2% of 3556289 rows
Read 3556289 rows and 38 (of 38) columns from 0.472 GB file in 00:00:11
##
Read 4.5% of 3567316 rows
Read 15.4% of 3567316 rows
Read 26.9% of 3567316 rows
Read 38.4% of 3567316 rows
Read 49.3% of 3567316 rows
Read 59.7% of 3567316 rows
Read 70.1% of 3567316 rows
Read 80.2% of 3567316 rows
Read 91.1% of 3567316 rows
Read 3567316 rows and 38 (of 38) columns from 0.473 GB file in 00:00:12
##
Read 0.0% of 3569127 rows
Read 10.9% of 3569127 rows
Read 21.9% of 3569127 rows
Read 32.8% of 3569127 rows
Read 43.4% of 3569127 rows
Read 54.1% of 3569127 rows
Read 64.4% of 3569127 rows
Read 75.1% of 3569127 rows
Read 85.2% of 3569127 rows
Read 95.8% of 3569127 rows
Read 3569127 rows and 38 (of 38) columns from 0.474 GB file in 00:00:12
##
Read 4.2% of 3576934 rows
Read 13.4% of 3576934 rows
Read 24.6% of 3576934 rows
Read 35.2% of 3576934 rows
Read 45.6% of 3576934 rows
Read 55.9% of 3576934 rows
Read 66.5% of 3576934 rows
Read 77.4% of 3576934 rows
Read 88.1% of 3576934 rows
Read 98.4% of 3576934 rows
Read 3576934 rows and 38 (of 38) columns from 0.475 GB file in 00:00:12
```

Next, start creating the visualizations. The colors need to mean the same thing on all of them for comparison purposes. So, the discretization needs to be the same for all the years:

```r
annpay <- ldply(data_list) #put all the data together
breaks <- quantile(annpay$avg_annual_pay,
                   c(seq(0,.8,.2),.9,.95,.99,1)) #make a common set of breaks
```

We will create the same visualization for each year, using the same breaks. Let's create a function for this common visualization to be produced. We will use ggplot2 for the visualizations. The input values are the data that we create using get_data, and the output is a plot object that can create the visualization:

```r
mychoro <- function(d, fill_label='Avg Annual Pay'){
  # d has a variable "outcome" that
  # is plotted as the fill measure
  chor <- left_join(county_df, d) %>%
    filter(!is.na(outcome))
  plt <- ggplot(chor, aes(long,lat, group=group))+
    geom_polygon(aes(fill=outcome))+
    geom_path(color='white',alpha=0.5,size=0.2)+
    geom_polygon(data=state_df, color='black',fill=NA)+
    scale_fill_brewer(palette='PuRd')+
    labs(x='',y='', fill=fill_label)+
    theme(axis.text.x=element_blank(),
          axis.text.y=element_blank(),
          axis.ticks.x=element_blank(),axis.ticks.y=element_blank())
  return(plt)
}
```

Now create plot objects for each year using a for loop. Store these objects in a list, with each element corresponding to each year. In the process, we create a new variable, outcome, which is the discretized pay variable, using the common breaks.

```r
plt_list <- vector('list',n)
for(i in 1:n){
  data_list[[i]] <- mutate(data_list[[i]],
                           outcome=Discretize(avg_annual_pay,breaks=breaks))
  plt_list[[i]] <-
    mychoro(data_list[[i]]) + ggtitle(names(data_list)[i])
}
```

The choroplethr package has the utility function, choroplethr_animate, which takes a list of plot objects created with ggplot2 and makes a web page with an animated GIF, layering the plots we created in order. The default web file is animated_choropleth.html:

```r
choroplethr_animate(plt_list)
```

```
## [1] "All files will be written to the current working directory: /Users/jinganqu/Dropbox/Github/Prac
## [1] "Now writing individual choropleth files there as 'choropleth_1.png', 'choropleth_2.png', etc."

## [1] "Now writing code to animate all images in 'animated_choropleth.html'.  Please open that file wi
```