

mapping_R Markup

Geospatial R Packages

Each example below starts by loading the dependant libraries. Make sure you have installed the packages below that you will be using:

```
install.packages(c("acs", "choroplethr", "choroplethrMaps", "maptools", "rgeos", "mapproj", "RColorBrewer", "maps"))
```

- **ACS & choroplethr** - can be used together to easily make choroplethr maps using data from the American Community Survey (ACS), yearly census data collected by the U.S. Census Bureau. To access ACS data you need an API key. Visit http://api.census.gov/data/key_signup.html, request a key, and paste it into the line below:

```
api.key.install("<ACS API key>")
```

- **choroplethrMaps** - contains maps of the USA used by the choroplethr package.
- **maptools** - contains functions for reading and manipulating geographic data, including ESRI shapefiles.
- **rgeos** - contains functions for performing geometric analysis. For example `gLength()` calculates the length of input geometry, while `gBuffer()` adds a buffer to an input feature.
- **mapproj** - simple package for converting from latitude and longitude into projected coordinates.
- **RColorBrewer** - provides color schemes that are especially useful for creating thematic maps.
- **maps** another simple set of tools for creating maps, with links to several databases of spatial data.
- **geosphere** - supports trigonometric calculations for geographic applications. For example, computing distance to the horizon from a given location and altitude.
- **rgdal** - geospatial data abstraction and projection / transformation.
- **reshape** - ???
- **ggplot2** - package for creating and customizing graphics in R.

Example 1: Mapping Census Data using ACS and Choroplethr

```
library(acs)
library(choroplethr)
```

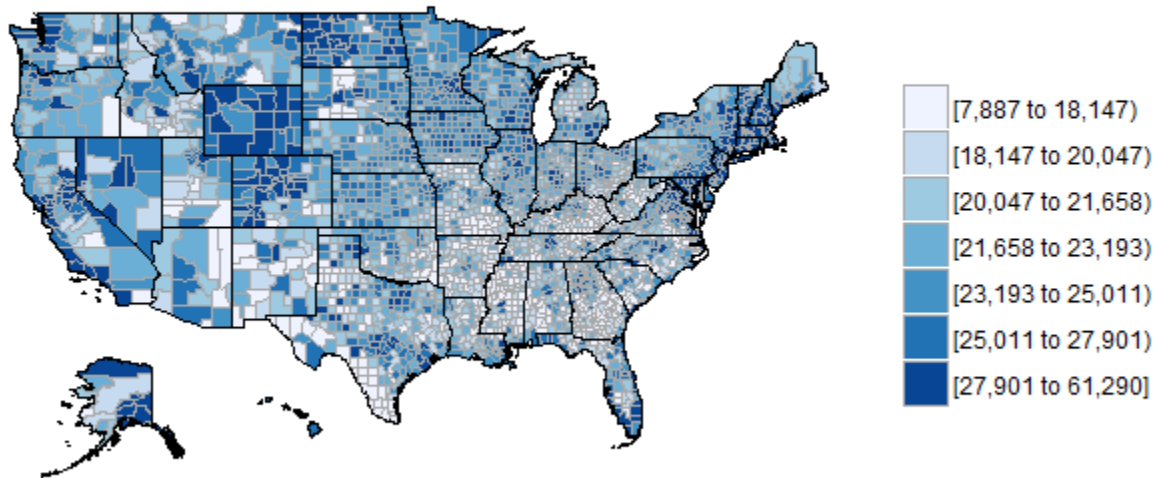
We need an api key to access the ACS data. Visit http://api.census.gov/data/key_signup.html, request a key, and paste it into the line below:

```
api.key.install("e3dd607b83adce3268ef2bb723da22c68001e6f0")
```

Great, now we have access to the census data. Table B19301 contains per capita income for the year of 2011. Lets plot it!

```
county_choropleth_acs(tableId="B19301")
```

Per capita income in the past 12 months (in 2011 inflation-adjusted dollars)



To see the description of a function and its arguments in R, place a “?” before its name:

```
?county_choropleth_acs
```

Reading in Shapefiles and Setting Projections

Load `maptools`, a library for reading and manipulating geographic data, including ESRI shapefiles.

```
library(maptools)
```

The following prompts you to select the provided county census shapefiles at the path `... county_census/Count_2010Census_DP`

Note: You will have to unzip the folder `county_census` first!

“+proj=longlat +datum=WGS84” = unprojected proj4

```
counties <- readShapeSpatial(file.choose(),proj4string=CRS("+proj=longlat +datum=WGS84"))
```

Let’s inspect the first few rows of the counties data to get a feel for its structure:

```
head(counties@data)
```

Census data assigns codes to counties using the Federal Information Processing Standard (FIPS). A FIPS code starts with two digits representing the state, and is followed by three digits representing the county. For example, Florida is 12 and Clay County Florida is 12019. So to select all the counties in Florida, we can use a regular expression matching all codes that start with “12”:

```
florida <- counties[substring(counties$GEOID10,1,2)=="12",]  
plot(florida)
```



You can look up other state and county codes using the U.S. Census Bureau site: <https://www.census.gov/geo/reference/codes/cou.html>

Projection and Layering with RGDAL

Next we'll work with library `rgdal`, a package for working with projections and transformations of geospatial data. We're going to read in a shape file of cultural points in Florida from the supplied data using function `readShapeSpatial()`. This function takes a `prj4`, a string containing the projection information of the shapefile, as an argument. We know already that our shapefile uses NAD83(HARN) / Florida GDL Albers. We can make the EPSG data frame of projections to find the `prj4` string for this projection (use `?make_EPSG()` to find out more about this table):

```
library(rgdal)
EPSG <- make_EPSG()
```

We can use regular expressions to search the note field of EPSG for any that refer to Florida:

```
EPSG[grep("florida", EPSG$note, ignore.case=TRUE), 1:2]
```

code	note
2236	# NAD83 / Florida East (ftUS)
2237	# NAD83 / Florida West (ftUS)
2238	# NAD83 / Florida North (ftUS)
2777	# NAD83(HARN) / Florida East
2778	# NAD83(HARN) / Florida West
2779	# NAD83(HARN) / Florida North
2881	# NAD83(HARN) / Florida East (ftUS)
2882	# NAD83(HARN) / Florida West (ftUS)
2883	# NAD83(HARN) / Florida North (ftUS)
3086	# NAD83 / Florida GDL Albers
3087	# NAD83(HARN) / Florida GDL Albers
3511	# NAD83(NSRS2007) / Florida East
3512	# NAD83(NSRS2007) / Florida East (ftUS)
3513	# NAD83(NSRS2007) / Florida GDL Albers
3514	# NAD83(NSRS2007) / Florida North

We see the code is 3087. Extract the `prj4` string from this dataframe:

```
subset(EPSG, code==3087)
prjstring <- subset(EPSG, code==3087)$prj4
```

Inspect our `prjstring` variable if you want to see the format of the `prj4` variable.

select cultural shapefile in cultural_centers

Now that we have the appropriate `prj4` we can read in the cultural centers data. The following prompts you to select the shape file. Select the actual `.shp` file in the provided data from `...cultural_centers/gc_culturecenter_oct15.shp`.

```
cultural <- readShapeSpatial(file.choose(),proj4string=CRS(prjstring))
```

Before we overlay the cultural points, we need to transform this layer to match that of the Florida counties layer - simple longitude and latitude in WGS84:

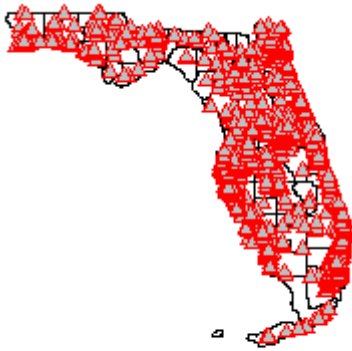
```
cultural_proj <- spTransform(cultural, CRS("+proj=longlat +datum=WGS84"))
```

```
plot(florigida)
points(cultural_proj)
```



You can play around with the symbology for your map with some additional arguments in the `points` function. For example:

```
plot(florigida)
points(cultural_proj, cex=.8, pch=24, col=554, bg="grey")
```



join polygon data to points

```
county_data <- over(cultural_proj,florida)
cultural_proj$pop <- county_data$DP0010001
```

set colors

```
library(RColorBrewer)

brks <- c(.5,1,1.5,2) * 1000000
cols <- brewer.pal(5,"Greens")

mapcols <- cols[findInterval(cultural_proj$pop, vec=brks)]
plot(cultural_proj,col=mapcols,pch=20)
```

base R instructions for choropleth

```
brks <- c(25,30,35,40,45,50,55,60,65)
cols <- brewer.pal(8,"Purples")

mapcols <- cols[findInterval(florida$DP0020001, vec=brks)]
plot(florida,col=mapcols,border="white")

legend("bottomleft", legend = levels(cut(florida$DP0020001, brks)), fill = cols, title = "Median Age")
```

using ggplot2

```
library(ggplot2)

fl_shapes <- fortify(florida,region="GEOID10")

ggplot() + geom_map(data=as.data.frame(florida),aes(map_id = GEOID10,fill=DP0020001), map=fl_shapes) +
```

networky type example

```
library(maps)
library(geosphere)
library(reshape)
```

select - state_shapes/tl_2014_us_state.shp

```
state <- readShapeSpatial(file.choose())
```

select - /state_migrations_2014.csv

```
migration <- read.csv(file.choose())

centrs <- data.frame(as.character(state@data$NAME),coordinates(state))
colnames(centrs) <- c("name","long","lat")
```

```
migration <- migration[c(1,6:56)]
long_mig <- melt(migration,id.vars="from_state")

map("state")
```

define draw__from__state function

```
draw_from_state <- function(centrs, migrations, state_name, color=rgb(0,0,0,alpha=0.5)) {
  migrations$variable <- sub(".", " ",migrations$variable,fixed=TRUE)
  migrations <- migrations[migrations$variable==state_name & migrations$from_state != state_name,]
  for(i in 1:nrow(migrations)){
    if (nrow(centrs[centrs$name==as.character(migrations[i,]$from_state),]) > 0){
      from_long <- centrs[centrs$name==as.character(migrations[i,]$from_state),]$long
      from_lat <- centrs[centrs$name==as.character(migrations[i,]$from_state),]$lat
      to_long <- centrs[centrs$name==as.character(migrations[i,]$variable),]$long
      to_lat <- centrs[centrs$name==as.character(migrations[i,]$variable),]$lat
      number <- migrations[i,]$value
      lines(gcIntermediate(c(from_long, from_lat), c(to_long, to_lat), n=50, addStartEnd=TRUE),lw
    }
  }
}

draw_from_state(centrs, long_mig, "Florida", rgb(0,0,1,0.5))

xlim <- c(-171.738281, -56.601563)
ylim <- c(12.039321, 71.856229)
map("world", col="#f2f2f2", fill=TRUE, bg="white", lwd=0.05, xlim=xlim, ylim=ylim)

draw_from_state(centrs, long_mig, "Wyoming", rgb(1,0,0,.5))
```