Intro to health data analysis in R

Background

Today we want to demonstrate how R can provide a one-stop-shop for health data analysis.

We'll start with base R today, and show how dplyr makes life easier.

To start, we'll use publicly available data on county-by-county health statistics for 2015.

Follow along

If interested:

- Download the data from here
- Install dplyr. Enter into the command line: install.packages(c("dplyr","choroplethr","choroplethrMaps"))
- Go to RStudio –> File –> New Rscript and copy in the code snippets that follow.
- To get a better understanding, comment out certain lines (using #), and then run the script again.
- Note you can run the entire script via CTRL-SHIFT-ENTER or run the script just to your cursor via CTRL-ALT-b
- If you get stuck, you can pull up documents via `?functionName` like `?read.csv`

Getting started

When you turn to Excel to analyze data, what are you typically trying to do? Perhaps you're looking for

- Easy sorting or filtering
- Pivot tables
- Simple calculations
- Simple plots

Let's show you how to do the same in R.

Note that dataframes are the most popular way in R to work with tabular data, and we'll focus on them throughout this tutorial.

First, let's load in the csv file and store it in a dataframe called df.

```r
df <- read.csv('2015 CHR Analytic Data.csv') #  You may want a path here, using /forward/slashes
```

Looking at data in RStudio

Now that we've read in the data, note that you see a df entry under your Environment tab.

Note that `df` has 3191 obs (i.e., rows, representing counties) and 329 variables (i.e., columns or metrics).

329 columns is a lot. Let's create a smaller dataset that's based only on our columns of interest.

Besides the state and county column, we'll grab county data on

1) The percentage of county births that are under 5 lbs 8 oz (i.e., low-birth weight or LBW).
2) Median household income
3) Percentage of individuals drinking heavily in the last 30 days.

```r
# Remove state summary rows
df <- subset(df, COUNTYCODE != 0)

# Remove commas from income column
df$Median.household.income.Value <- as.numeric(gsub(",", "", df$Median.household.income.Value))

# Create a smaller dataframe, with just a few cols
colList <- c('State',
             'County',
             'Low.birthweight.Value',
             'Median.household.income.Value',
             'Excessive.drinking.Value')

dfSmall <- subset(df, select = colList)
```

Look again in the Environment tab, double-click on `df_small`, and you'll see an Excel-like view open. You'll notice that you can easily

1) Sort the dataset in both directions by each column
2) Filter the rows displayed based on sliders or an input value for a particular column (i.e., counties in Nevada)

Note: you can also bring up the interactive tab via `View(df)`

Sorting and filtering in RStudio

NOW YOU TRY:

Use the df interactive tab to find the counties in Utah with the highest rate of low-birth weight (LBW) births.

Besides the interactive route, note that there's always a code-based way of doing the same thing.

Let's list the five counties with the highest rate of LBW in Utah (using base R):

```r
# Order the dataframe by LBW (descending)
dfUtah <- subset(dfSmall, State == 'UT')

dfUtahOrdered <- dfUtah[with(dfUtah, order(Low.birthweight.Value, decreasing = TRUE)),]

# Print first five rows of dataframe
head(dfUtahOrdered, n = 5)
```

```
##      State          County Low.birthweight.Value
## 2823    UT   Carbon County                  0.09
## 2827    UT    Emery County                  0.09
## 2840    UT   Sevier County                  0.09
## 2841    UT   Summit County                  0.09
## 2826    UT Duchesne County                  0.08
##      Median.household.income.Value Excessive.drinking.Value
## 2823                         44594                    0.114
## 2827                         52070                    0.105
## 2840                         49877                    0.096
## 2841                         81907                    0.191
## 2826                         61386                    0.093
```

And now let's do the same thing with dplyr. (See here for a cheat sheet.)

```r
library(dplyr)

dfSmall %>%
  filter(State == "UT") %>%
  arrange(desc(Low.birthweight.Value)) %>%
  top_n(5)
```

```
##   State           County Low.birthweight.Value
## 1    UT    Carbon County                  0.09
## 2    UT    Summit County                  0.09
## 3    UT    Uintah County                  0.08
## 4    UT     Grand County                  0.07
## 5    UT Salt Lake County                  0.07
##   Median.household.income.Value Excessive.drinking.Value
## 1                         44594                    0.114
## 2                         81907                    0.191
## 3                         62028                    0.128
## 4                         42368                    0.161
## 5                         61716                    0.120
```

So if Carbon, Emery, Summit, and Sevier counties were listed for highest LBW, you were correct!

Notice how simple the dplyr syntax is!

Pivot tables in R

Since we have counties from multiple states in this dataset, we can easily use dplyr to get the LBW mean for each state.

```r
dfSmall %>%
  group_by(State) %>%
  summarize(LBWMean = mean(Low.birthweight.Value, na.rm = TRUE)) %>%
  arrange(LBWMean)
```

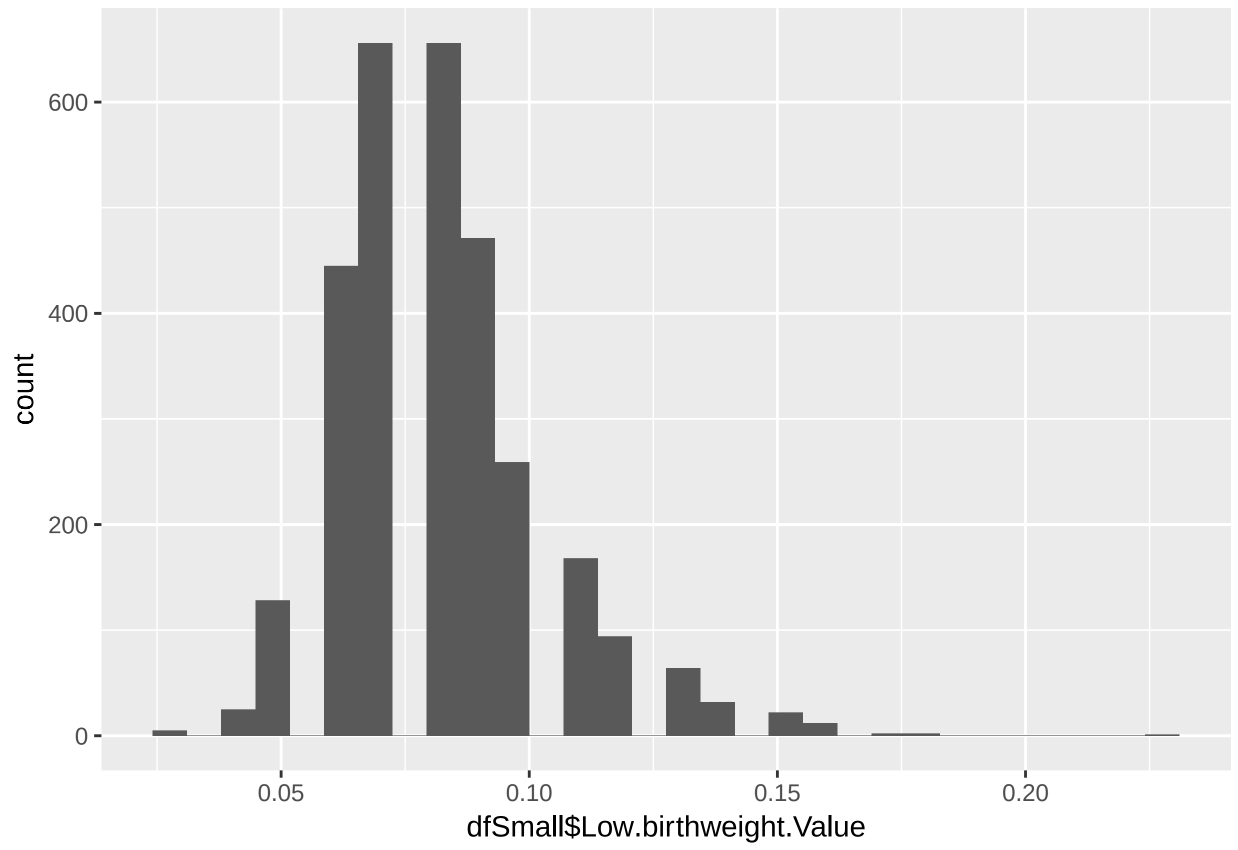
```
## # A tibble: 51 × 2
##     State    LBWMean
##    <fctr>      <dbl>
## 1      AK 0.05523810
## 2      MN 0.05954023
## 3      WA 0.06153846
## 4      WI 0.06281690
## 5      CA 0.06315789
## 6      SD 0.06410714
## 7      OR 0.06411765
## 8      ND 0.06487179
## 9      ME 0.06500000
## 10     NE 0.06500000
## # ... with 41 more rows
```

View the LBW as a histogram.

Similar to dplyr, ggplot2 is a fantastic add-on package that makes life easier. While dplyr focuses on analysis, ggplot focuses on plotting.

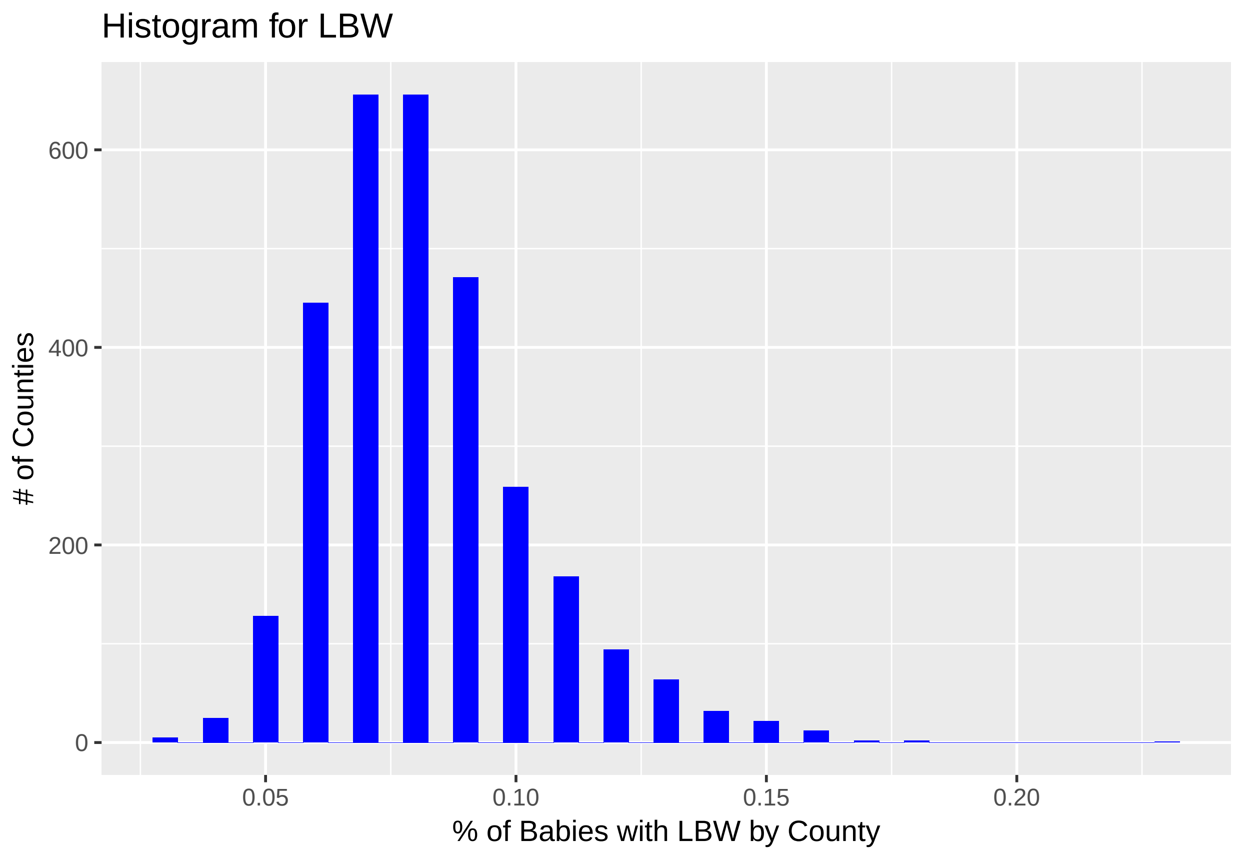
First, the most basic thing possible.

```r
library(ggplot2)
#qplot is used for really quick and dirty plots. There's less customization than ggplot.
qplot(dfSmall$Low.birthweight.Value, geom="histogram")
```



Clean up the plot

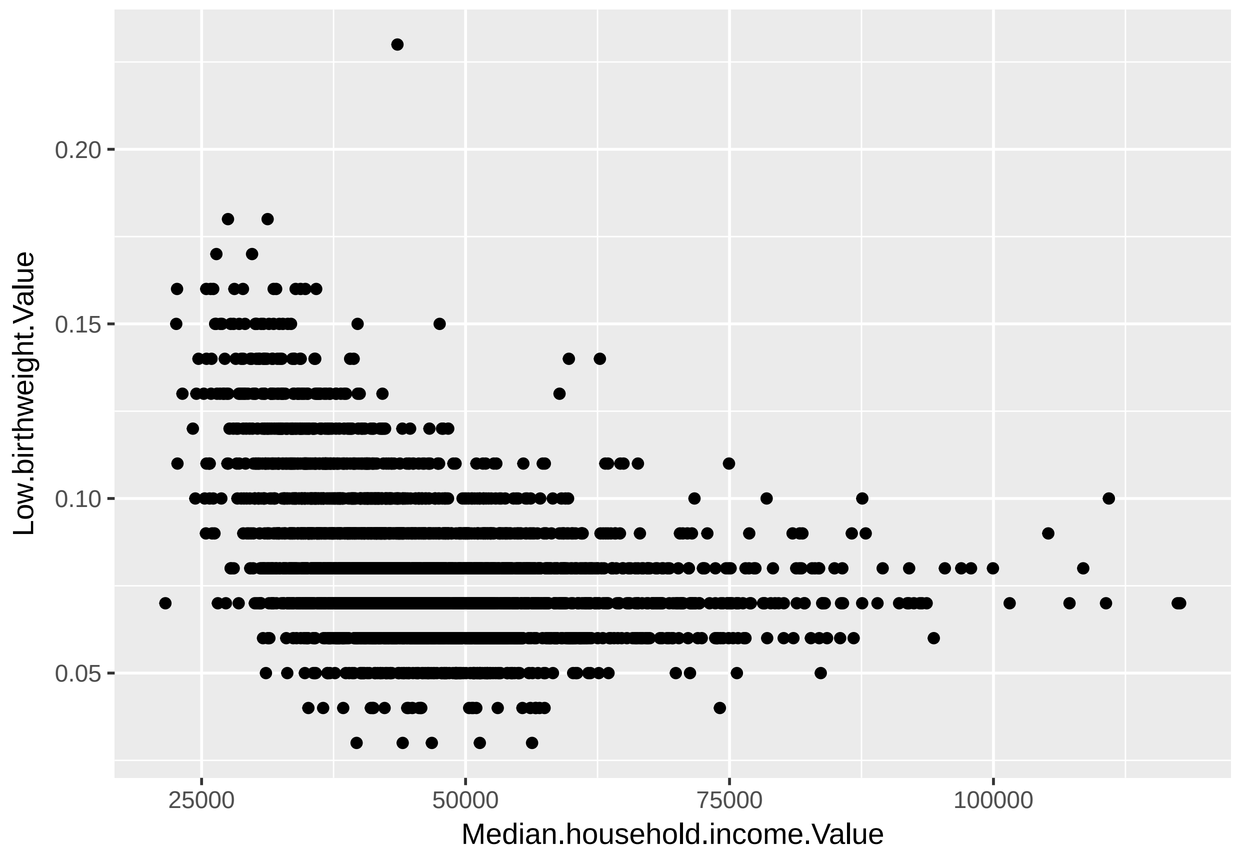
```r
#qplot is used for really quick and dirty plots. There's less customization than ggplot.
qplot(dfSmall$Low.birthweight.Value,
      binwidth = 0.005,
      main = "Histogram for LBW",
      xlab = "% of Babies with LBW by County",
      ylab = "# of Counties",
      fill = I("blue"),
      geom="histogram")
```



A simple scatter plot

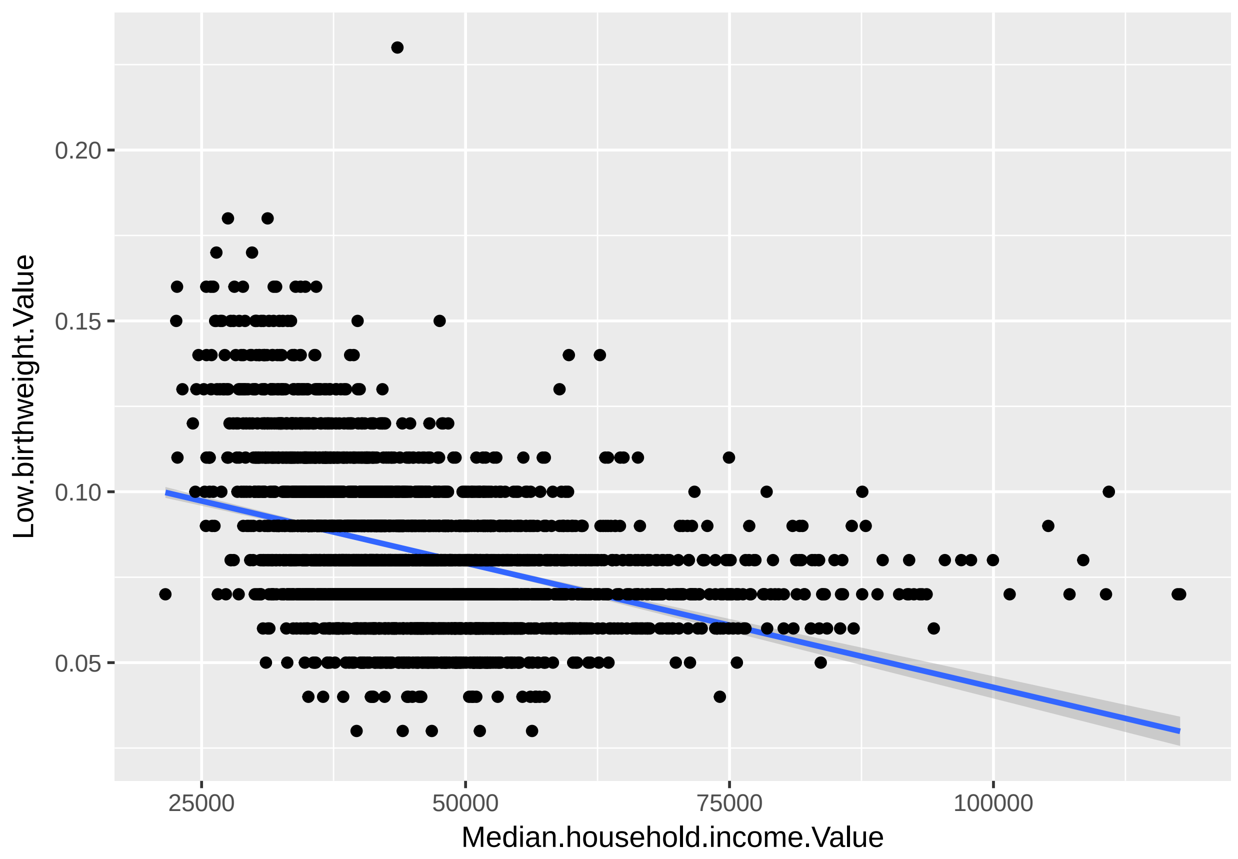
GGplot can also be used to look at the relationship between LBW and income.

```r
library(ggplot2)
# We can use ggplot for this one.
ggplot(dfSmall, aes(x = Median.household.income.Value, y = Low.birthweight.Value)) + geom_point()
```



Okay, it appears that that higher income counties have smaller rates of LBW. Let's fit a line to check:

```
c <- ggplot(dfSmall, aes(x = Median.household.income.Value, y = Low.birthweight.Value))
c + stat_smooth(method=lm) + geom_point()
```



Let's do a calculation to check the LBW-income relationship

Up till now, we've been sorting, filtering, and creating pivot tables. Let's use a two-sample t-test to test the likelihood that these two vectors of LBW values come from the same distribution. In other words, we want to test if income has a significant effect on LBW rates.

Now let's imagine that we want two vectors of LBW. One for richer states and one for poorer states. How can we do this? First, let's create the LBW vector–think of it as a list–for states richer than the median state.

```
richStatesLBW <-
dfSmall %>%
  group_by(State) %>%
  summarize(LBWMean = mean(Low.birthweight.Value, na.rm = TRUE), IncomeMean = mean(Median.household.inco
  filter(IncomeMean > median(IncomeMean)) %>%
  .$LBWMean
```

Now let's create a vector–think of a list–of LBW values for poorer states:

```
poorStatesLBW <-
dfSmall %>%
  group_by(State) %>%
  summarize(LBWMean = mean(Low.birthweight.Value, na.rm = TRUE), IncomeMean = mean(Median.household.inco
  filter(IncomeMean < median(IncomeMean)) %>%
  .$LBWMean
```

Vectors are ready. Run the T-Test

```r
t.test(richStatesLBW, poorStatesLBW)
```

```
##
##  Welch Two Sample t-test
##
## data:  richStatesLBW and poorStatesLBW
## t = -2.5115, df = 44.129, p-value = 0.01576
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.018691341 -0.002049278
## sample estimates:
##  mean of x  mean of y
## 0.07453657 0.08490688
```

Recall that typically p-values below 0.05 allow one to reject the null hypothesis that income does not affect LBW rates. Since, p-value here is ~0.02, that means we can conclude that county income does have a statistically significant affect on county rates of LBW.

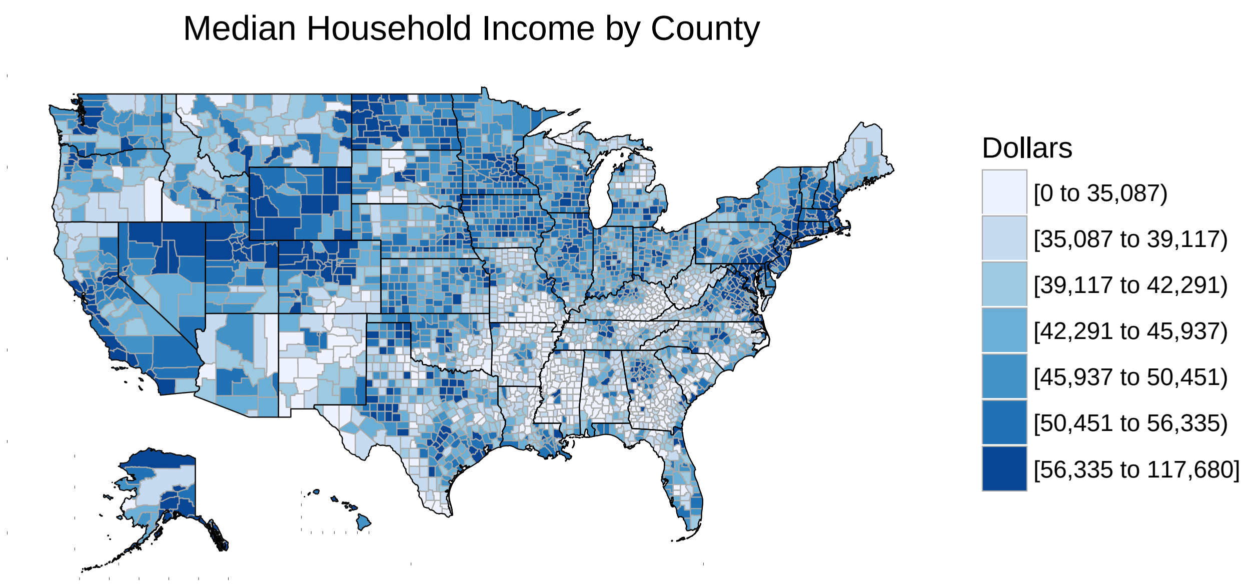
A more advanced example: data cleaning and choropleths.

Now we've fit a line and confidence interval to our data, and run a T-Test. Excel can do all of that (arguably more easily than R), but now let's explore an example a more complicated feature that R can still do easily. This is a lot of code, but most of it is unnecessary for a single choropleth. We use it so that we can make several choropleths efficiently.

```r
removeCommasInNumber <- function(column) {
  column <- as.numeric(gsub(",", "", column))
  column
}

createChoropleth <- function(df,
                             colToPlot,
                             title,
                             legend,
                             numColors=1,
                             NAReplace=NULL) {

  library(choroplethr)
  library(choroplethrMaps)
  library(ggplot2)

  # Remove commas from column of interest
  if (!is.numeric(df[[colToPlot]])) {
    df[[colToPlot]] <- removeCommasInNumber(df[[colToPlot]])
  }

  # Remove state rows from dataset
  df <- subset(df, COUNTYCODE != 0)

  df$STATECODE <- as.integer(df$STATECODE)
  df$COUNTYCODE <- as.integer(df$COUNTYCODE)

  # Pad county digits
  df$COUNTYCODE <- sprintf("%03d", df$COUNTYCODE)
  # Concatenate and create FIPS
  df$FIPSCODE <- as.numeric(paste0(df$STATECODE,df$COUNTYCODE))

  # Reduce dataset and rename cols for county_choropleth func
  df <- subset(df, select = c('FIPSCODE', colToPlot))
  colnames(df) <- c("region","value")

  # Fill NA cells with something (so choropleth works)
  if (!is.null(NAReplace)) {
    df["value"][is.na(df["value"])] <- NAReplace
  }

  # Plot data on a US map
  county_choropleth(df, num_colors = numColors, legend = legend) +
    ggtitle(title) +
    theme(plot.title = element_text(hjust = 0.5))
}

createPercentiles <- function(x) {
  (x - min(x, na.rm = TRUE))/(max(x, na.rm = TRUE) - min(x, na.rm = TRUE))
}
```

Now, call that function on Median Household Income

```
# Plot choropleth
createChoropleth(df,
                 colToPlot = 'Median.household.income.Value',
                 #colToPlot = 'Premature.death.Value',
                 #colToPlot = 'Low.birthweight.Value',
                 title = 'Median Household Income by County',
                 legend = 'Dollars',
                 numColors = 7,
                 NAReplace = 0)
```


Median Household Income by County
Dollars
[0 to 35,087)
[35,087 to 39,117)
[39,117 to 42,291)
[42,291 to 45,937)
[45,937 to 50,451)
[50,451 to 56,335)
[56,335 to 117,680]