Lecture 3: Histograms

Leah Brooks

January 30, 2018

The goals of this tutorial are to

- introduce ggplot
- make some histograms
- give you some understanding of where the histograms come from

For Problem Set 3, turn in

- code and output from tutorial
- answers to all questions in red throughout
- the additional open-ended assignment at the end

A. Load Data

We will rely on the datasets from the two previous classes. Please see the previous two tutorials for explanations of those datasets. We will focus today on the block group data.

To load the data, write

```
# load the data
block.groups <- read.csv("h:/pppa_data_viz/2018/assignments/lecture02/acs_bgs20082012_dmv_20180123.csv"

# check to make sure size is reasonable
dim(block.groups)
```

```
## [1] 9708 3063
```

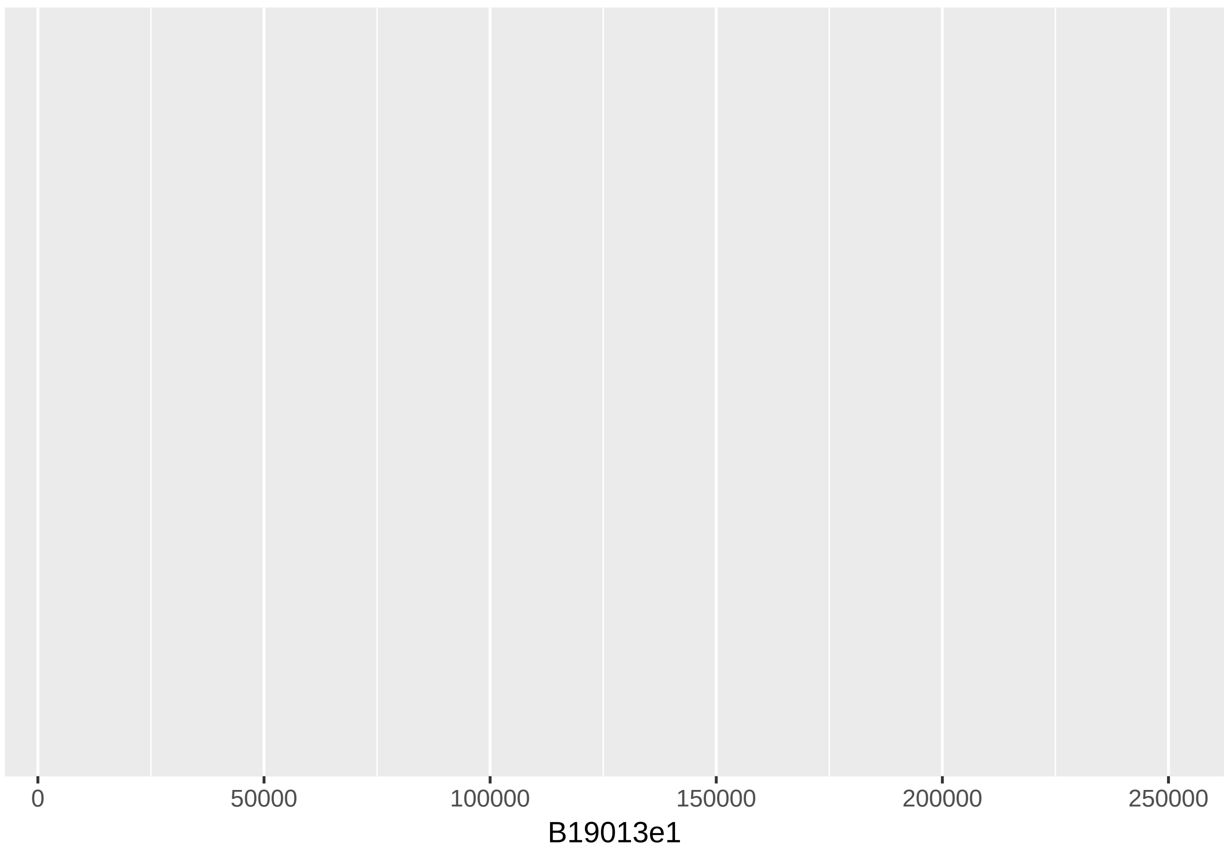
B. Get a sense of ggplot

We'll have discussed ggplot in the lecture, so I don't provide too much detail here.

Remember that nothing will work unless you install the library. (Do not re-install the package unless you fail to see it in the "packages" window.) You need only install the library once per session or program.

The ggplot package is best understood by example. Here are a set of plots to create and see what R is doing. Recall that variable `B19013e1` is median household income in the past 12 months.

```
## load ggplot library
library(ggplot2)

## note that this doesnt make anything: no geom_* command
ggplot(block.groups, aes(x=B19013e1))
```

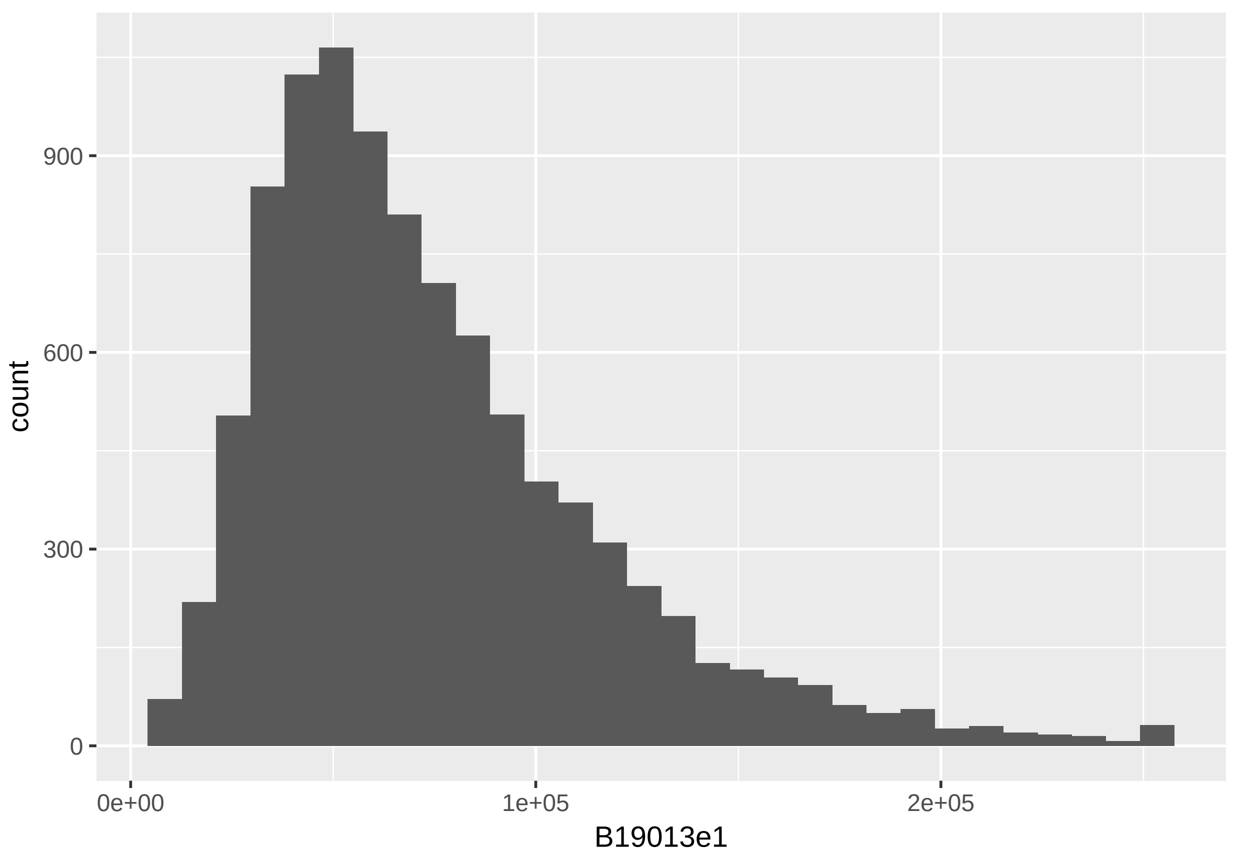


This makes nothing, because there is no `geom` command. R knows the appropriate x axis, but doesn't know what kind of dots or bars or lines to use.

Now let's add a variety of `geom` commands.

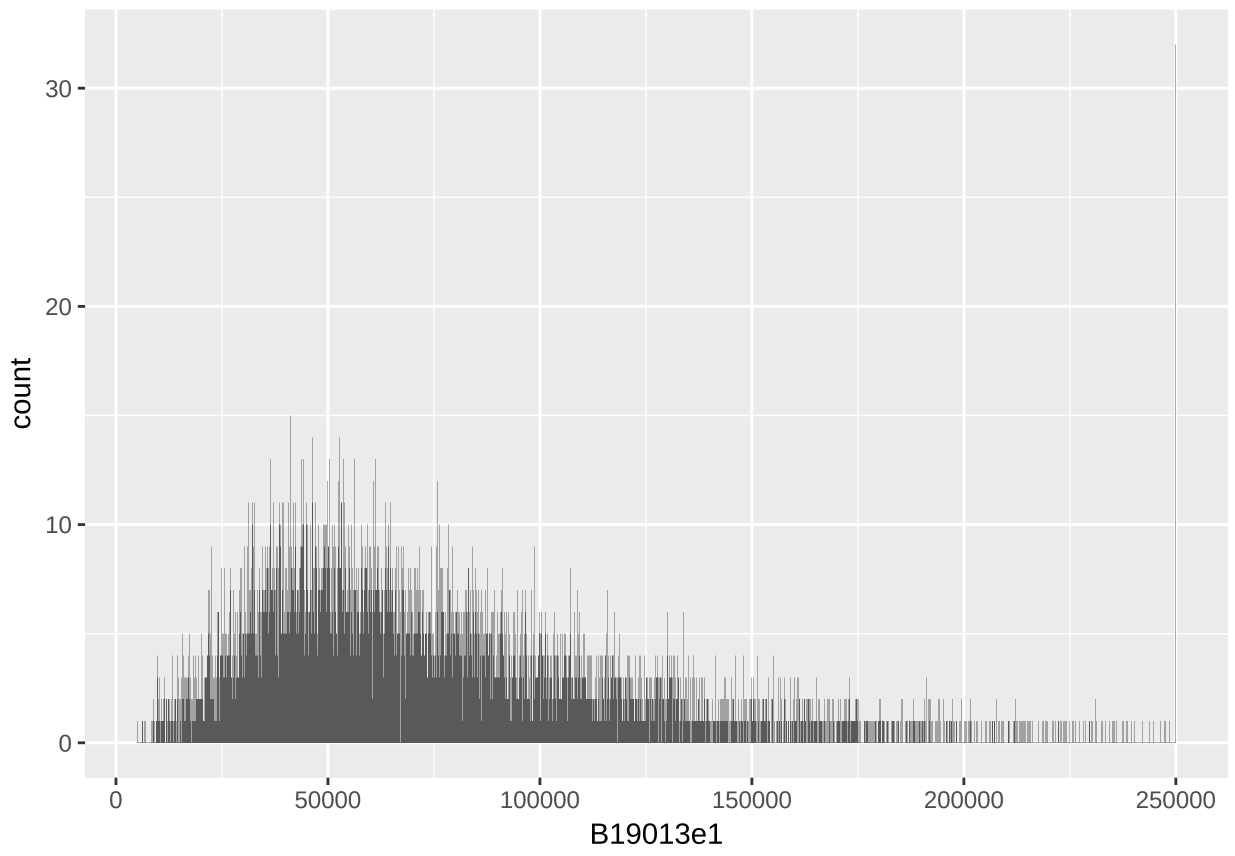
```
## this makes a basic histogram
ggplot(block.groups, aes(x=B19013e1)) + geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

## Warning: Removed 108 rows containing non-finite values (stat_bin).
```



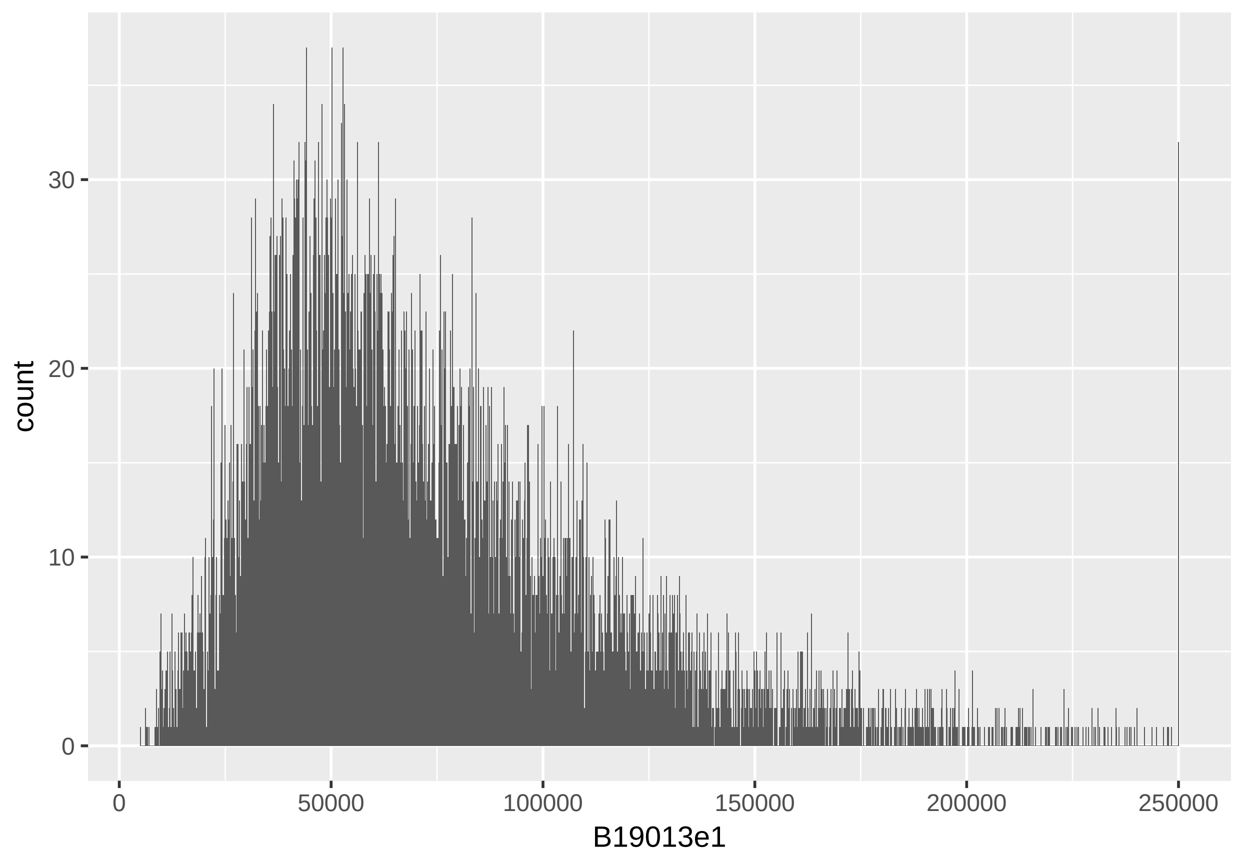
```
## here we begin changing the width of the histogram bins
ggplot(block.groups, aes(x=B19013e1)) + geom_histogram(binwidth = 50)
```

```
## Warning: Removed 108 rows containing non-finite values (stat_bin).
```



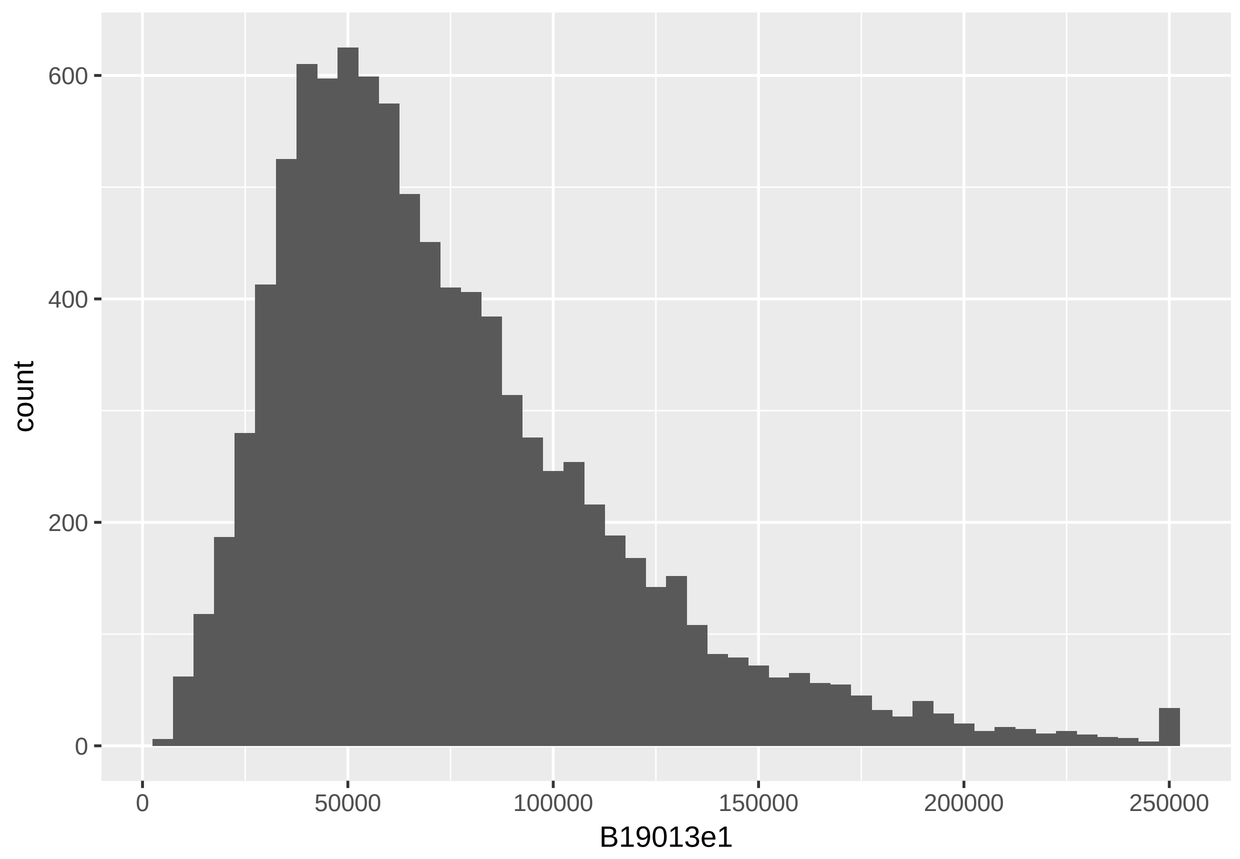
```
## again
ggplot(block.groups, aes(x=B19013e1, na.rm = TRUE)) + geom_histogram(binwidth = 200)
```

```
## Warning: Removed 108 rows containing non-finite values (stat_bin).
```



```
## and again
ggplot(block.groups, aes(x=B19013e1, na.rm = TRUE)) + geom_histogram(binwidth = 5000)
```

```
## Warning: Removed 108 rows containing non-finite values (stat_bin).
```



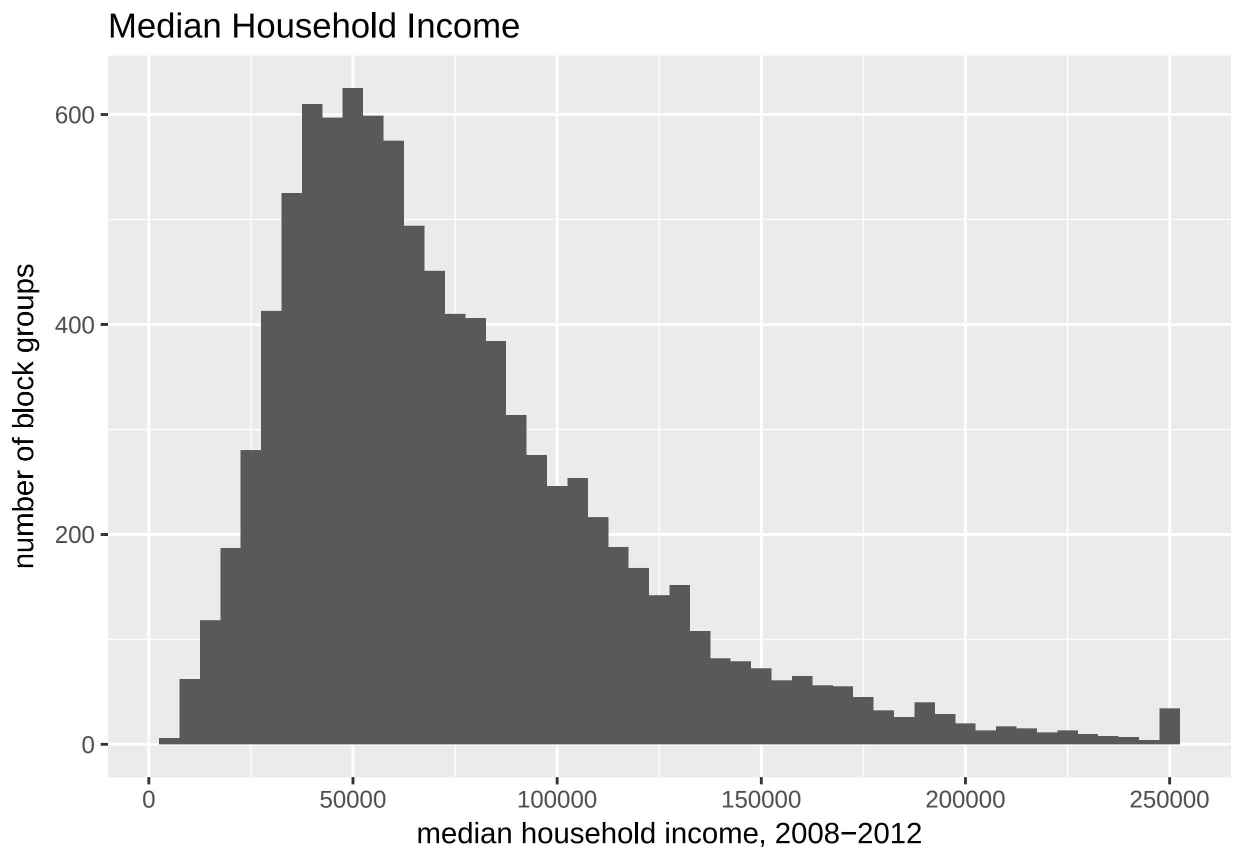
Note that the choice of bin is very important to the final look of the graph. Also notice that very small bins make the top-coded final category (for all block groups with a median income greater than 250,000, the Census reports 250,000) look big. Why is this?

C. Make things a bit more legible: titles and axis labels

Even with the small number of plots we just made, we can get lost without titles. (Though frequently I end up omitting the title in the very final product because I put the title on with other software.)

```
## make things legible: titles and axes
ggplot(block.groups, aes(x=B19013e1, na.rm = TRUE)) + geom_histogram(binwidth = 5000)  +
  ggtitle("Median Household Income")  +
  labs(y="number of block groups", x="median household income, 2008-2012")
```

```
## Warning: Removed 108 rows containing non-finite values (stat_bin).
```



A word of warning: you may wish to divide up long lines of code so that they are legible. This seems perfectly reasonable. However, don't put the + character at the beginning of a line that continues a command. R will go crazy, and you may go crazy, too. See here for details.

We will talk more later about the power of titles and axis labels. You should consider them key to any decent final product.

D. Other ggplot options for types of histograms

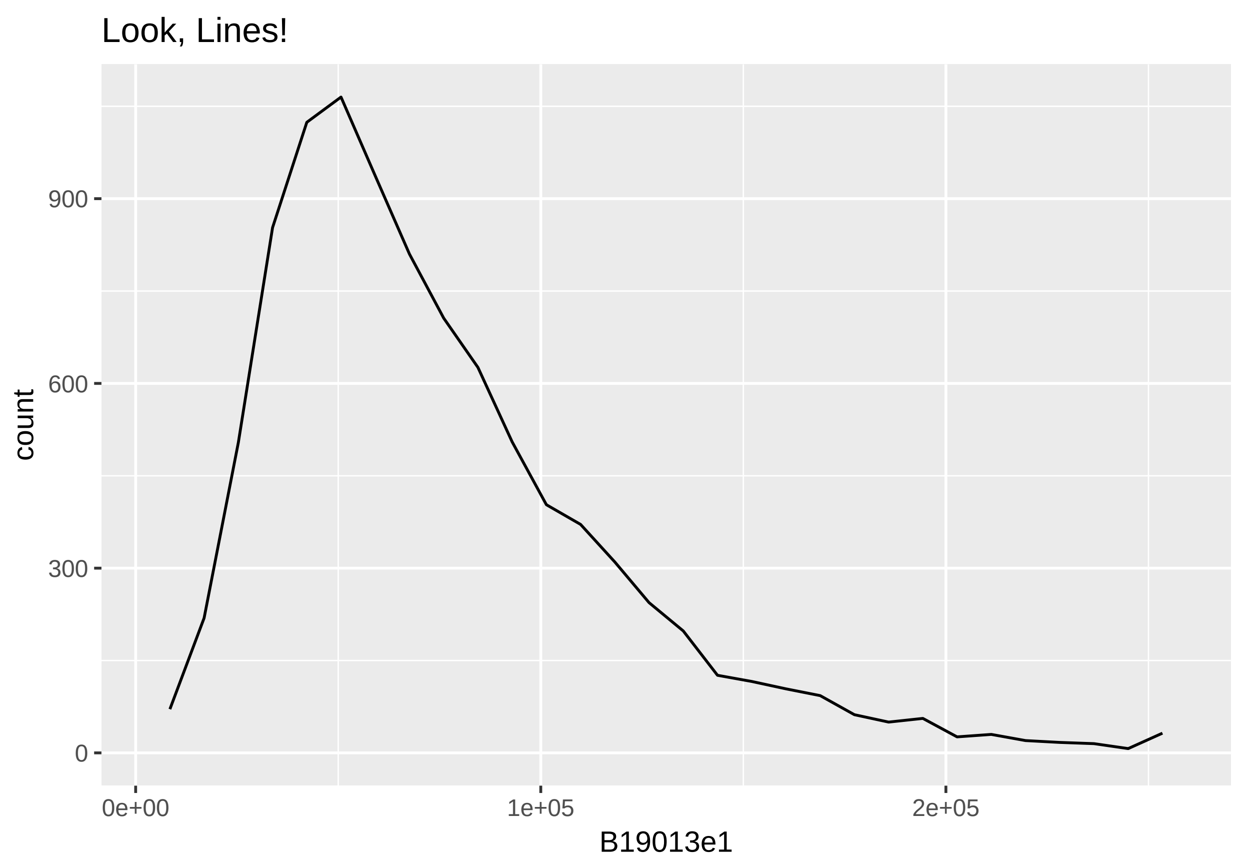
Here we explore options beyond bars for histograms.

We begin by using curves.

```
## B.2. density curve instead of bars
ggplot(block.groups, aes(x=B19013e1)) + stat_bin(geom = "line") +
  ggtitle("Look, Lines!")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

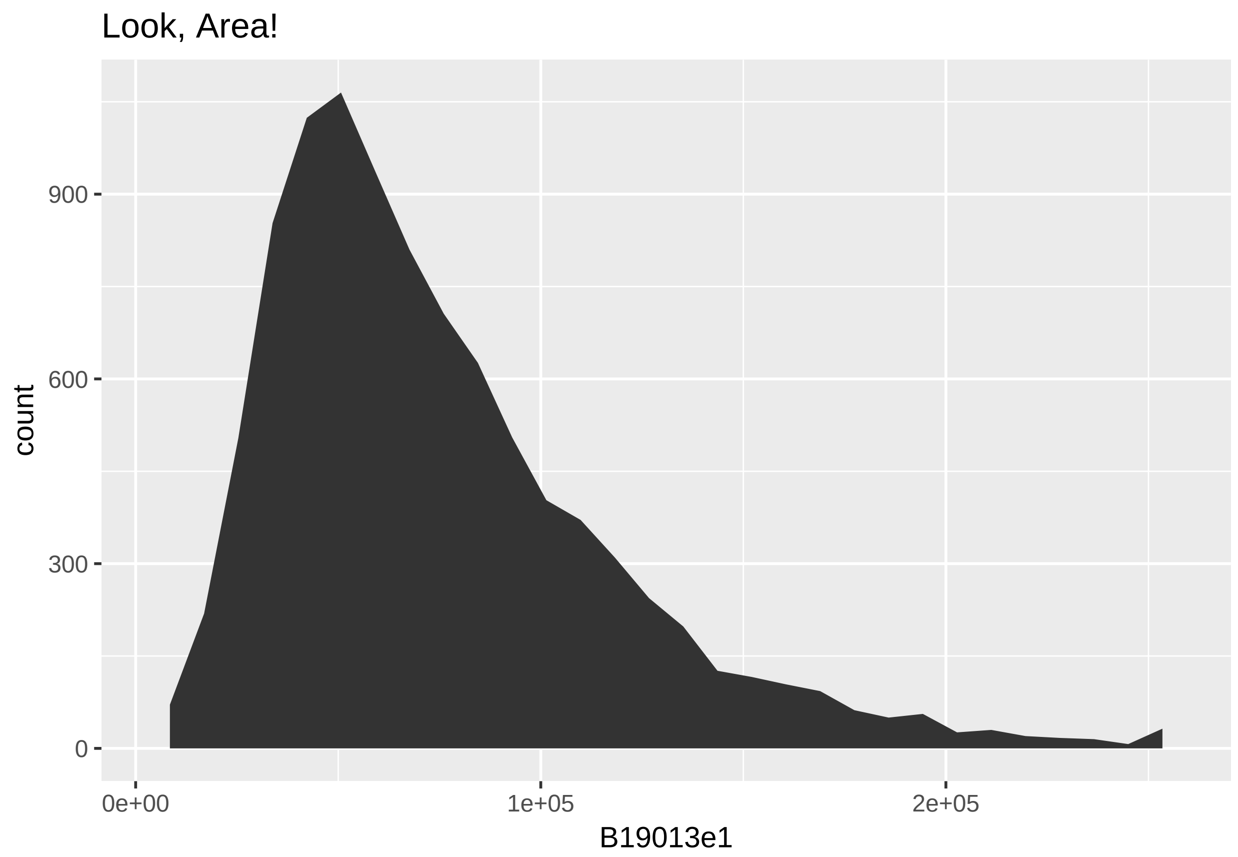
```
## Warning: Removed 108 rows containing non-finite values (stat_bin).
```



```
ggplot(block.groups, aes(x=B19013e1)) + stat_bin(geom = "area") +
  ggtitle("Look, Area!")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
## Warning: Removed 108 rows containing non-finite values (stat_bin).
```

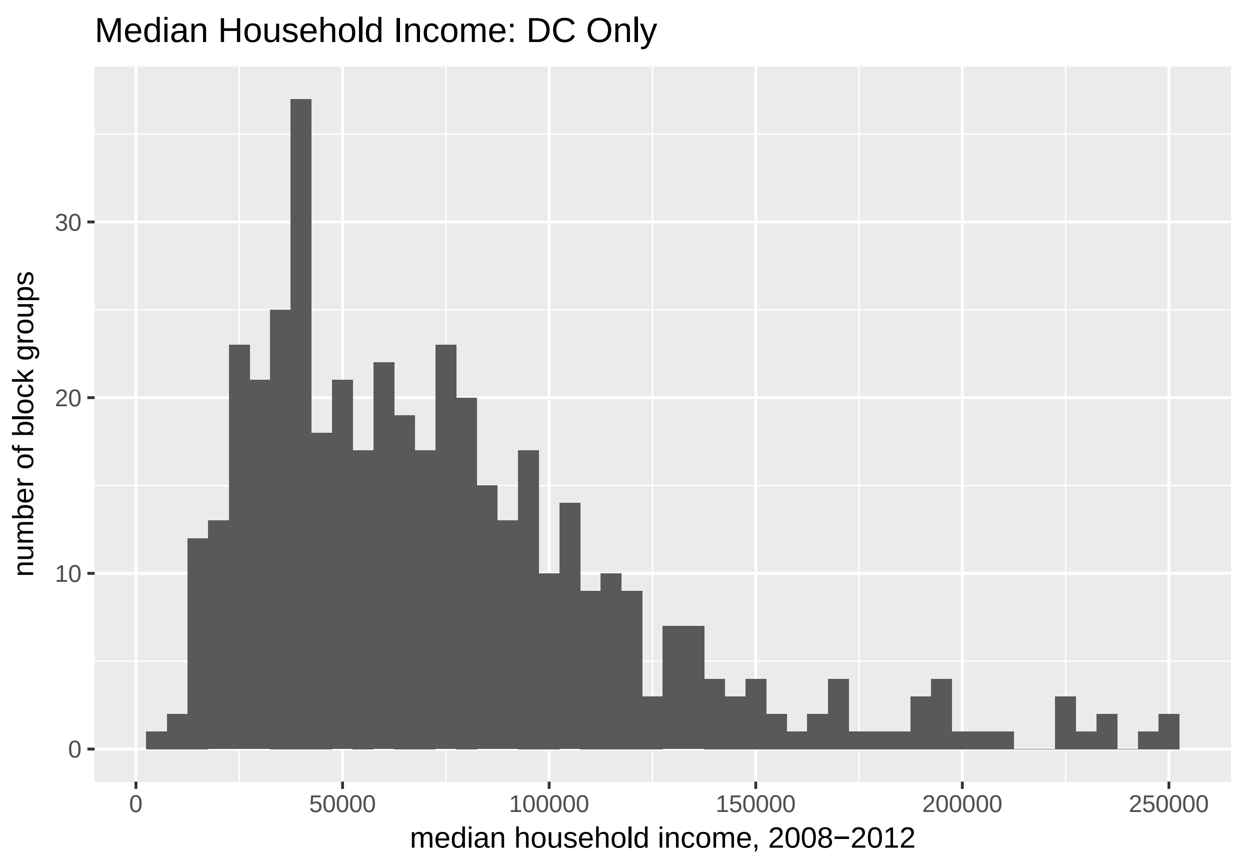


I don't view these as superior to the previous final histogram because to me they seem more difficult to understand without any other benefit.

We can limit the analysis only to a subgroup – this can frequently be useful and enlightening. Note that we use the double equals sign for evaluating (here we evaluating, not assigning).

```
## subsetting to only certain data
ggplot(subset(block.groups, STATE == "11"), aes(x=B19013e1, na.rm = TRUE)) +
  geom_histogram(binwidth = 5000) +
  ggtitle("Median Household Income: DC Only")+
  labs(y="number of block groups", x="median household income, 2008-2012")
```

```
## Warning: Removed 3 rows containing non-finite values (stat_bin).
```



Compare this distribution to the previous to see how DC's relative distribution. Of course, looking across two graphs is not an ideal comparison method. We'll work on this as we go along.

(As an aside, you can accomplish the same thing with the code below.

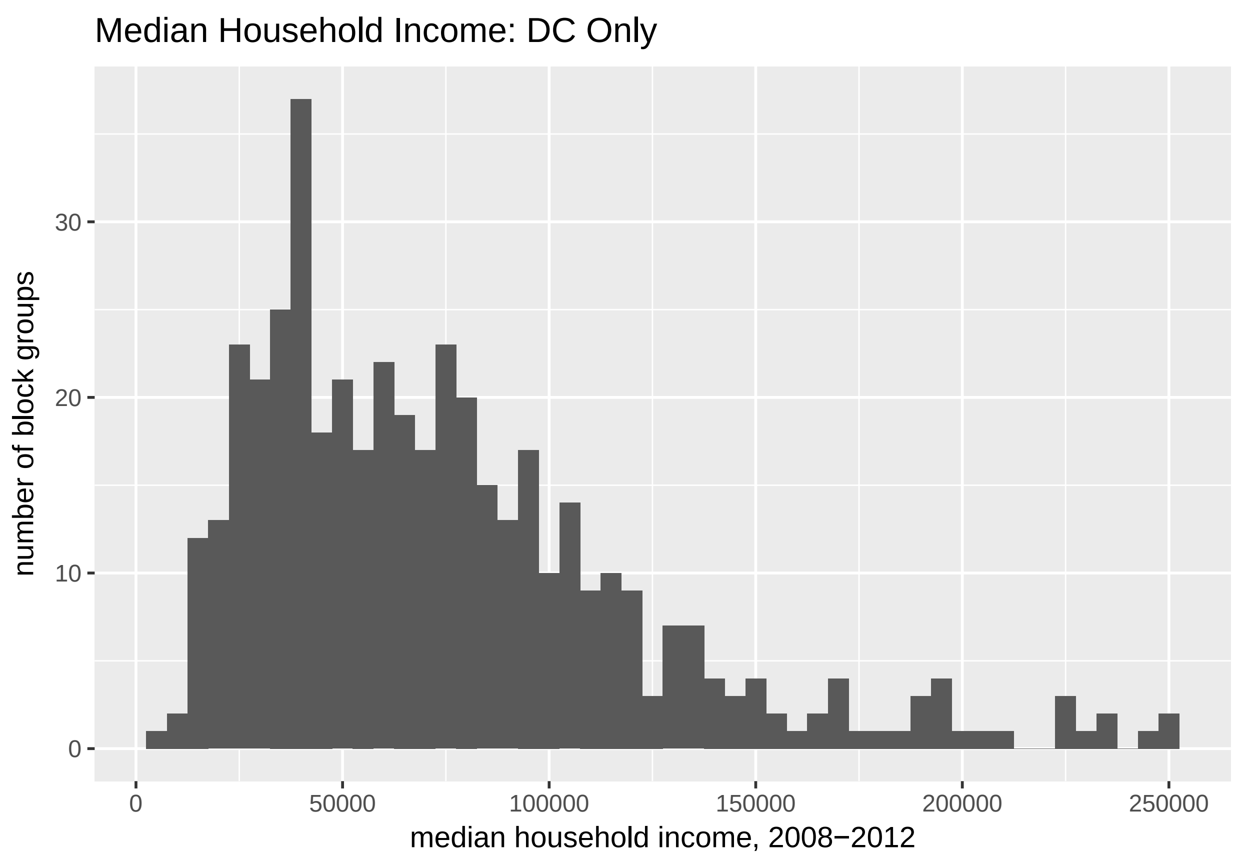
```
block.groups.dc <- block.groups%>% filter(STATE == 11)
ggplot(block_groups_dc, aes(x=B19013e1, na.rm = TRUE)) +
  geom_histogram(binwidth = 5000) +
  ggtitle("Median Household Income: DC Only")+
  labs(y="number of block groups", x="median household income, 2008-2012")
```

The first code is more efficient, since we don't make a new data frame, but it may be less clear.)

Finally, it may be useful to save the graph as an object that you can use later. You can do this with the below. The second line prints the object.

```
## B.4. you can make this an object
stdc <- ggplot(subset(block.groups, STATE == "11"), aes(x=B19013e1, na.rm = TRUE)) +
  geom_histogram(binwidth = 5000) +
  ggtitle("Median Household Income: DC Only")+
  labs(y="number of block groups", x="median household income, 2008-2012")
stdc
```

```
## Warning: Removed 3 rows containing non-finite values (stat_bin).
```



E. Comparing the three jurisdictions

I had hoped to use this class to introduce loops. In working on this, I realized that loops are not very R-like. Almost everything that a loop does is better done with a matrix or list in R. You sometimes have to do heroic coding to make a loop work like you'd like, and heroics are not needed.

So, instead, to compare the three jurisdictions, you should not use a loop.

In principle, you could layer three plots of distributions on top of each other, which is what I tried (and never succeeded in doing) with the first piece of commented out code. (If you figure out my error, let me know!)

However, this method is not a good one because `ggplot` already has built-in tools to make graphs by group.

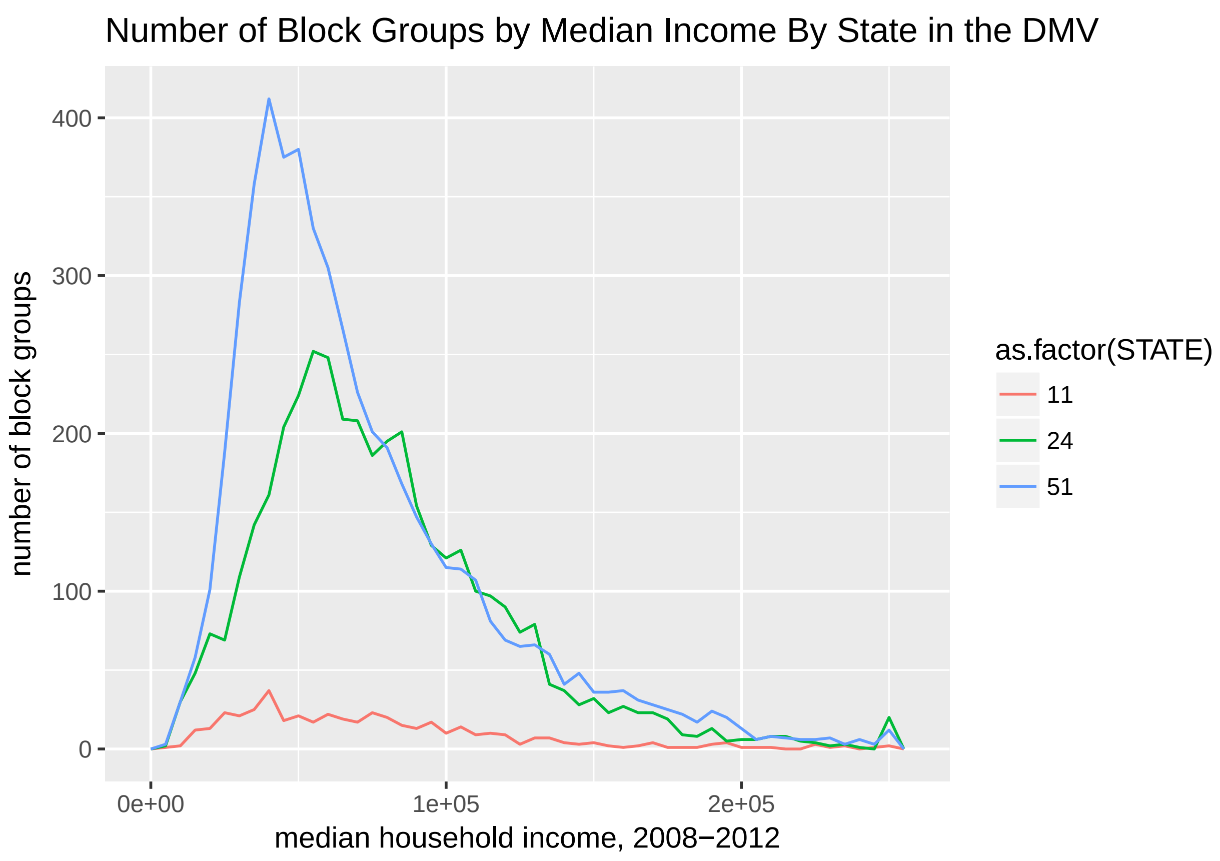
The important missing ingredient before we do this is to understand R's "factor" variables. In essence, a factor variable is a variable that takes on a limited number of values. This may also be known as a categorical variable. Give two examples of things that could be factor variables and one example of something that could not be a factor.

We care about factors here because we can only make graphs by factor variables. If something is not already a factor variable, but it takes on a limited set of values, we can pretend it is by putting `as.factor()` around it.

So, to the code. The first block doesn't work, so don't try it! The second block shows the number of block groups by state and median income and the third the share (setting the y axis to be density rather than counts with the `..density..` command) of block groups by state and median income. Why is the third the most appropriate for cross-state comparisons?

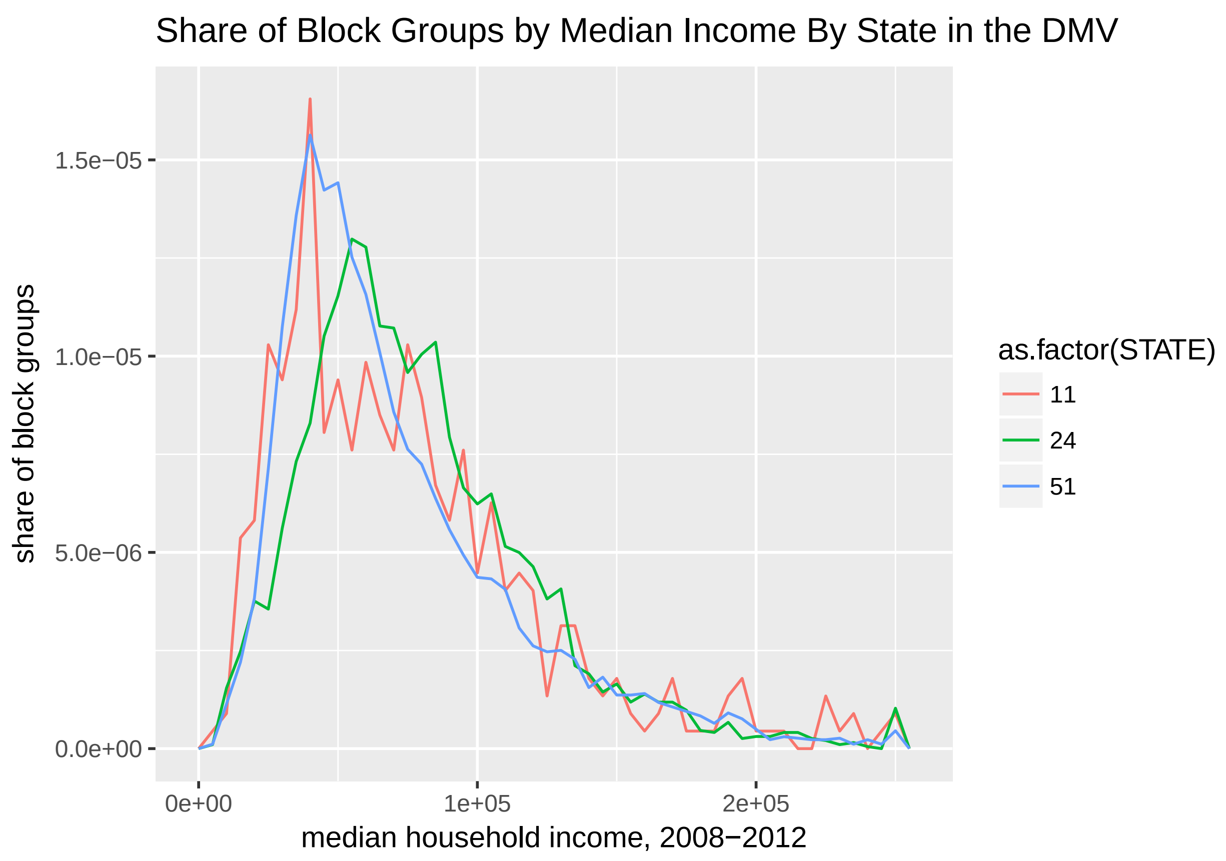
```
# B.5.1 bad way to combine three different states
# in fact, its so bad i can't get it to work!
#ggplot(block.groups, aes(x=B19013e1, na.rm = TRUE)) +
#   geom_histogram(subset(block.groups, STATE = "11") ,binwidth = 5000) +
#   geom_histogram(subset(block.groups, STATE = "24") ,binwidth = 5000) +
#   geom_histogram(subset(block.groups, STATE = "51") ,binwidth = 5000) +
#   ggtitle("Median Household Income: MD, VA, and DC")+
#   labs(y="number of block groups", x="median household income, 2008-2012")


# B.5.2. better way
ggplot(block.groups, aes(x=B19013e1, colour = as.factor(STATE))) +
  geom_freqpoly(binwidth=5000) +
  ggtitle("Number of Block Groups by Median Income By State in the DMV") +
  labs(y="number of block groups", x="median household income, 2008-2012")
```

```
## Warning: Removed 108 rows containing non-finite values (stat_bin).
```



```
# B.5.3. still better way
ggplot(block.groups, aes(x=B19013e1, y = ..density..,  colour = as.factor(STATE))) +
  geom_freqpoly(binwidth=5000) +
  ggtitle("Share of Block Groups by Median Income By State in the DMV") +
  labs(y="share of block groups", x="median household income, 2008-2012")
```

```
## Warning: Removed 108 rows containing non-finite values (stat_bin).
```



There are still plenty of unpleasant things about this graph. In this list, I would include

- illegible numbers on the horizontal axis
- probably bad smoothing making the DC line more jerky than the others
- density in illegible units
- hard to read legend
- unhelpful color scheme
- potentially useless grey background
- poor axis labels
- lines too jerky to convey desired idea

To my mind, the most immediate obstacle to comprehension are the axis numbers; we will hold off on this until next class.

F. Other ways of cutting the data

We've just looked at median income by state, but one could easily think to examine median income by other variables. To give you one other example, suppose I am interested in median income by the share of homes in the tract that have mortgages. Since this second variable is continuous, we can't look at the entire distribution of median income by the entire distribution of the share of homes with a mortgage. But we can find quartiles of the share of homes with a mortgage, and then compare the distribution of median income across these quartiles.

To do this, first I need to make a variable that measures the share of homes in each block group with a mortgage. I do this with familiar commands below, and then I check the output using the `summary` function to make sure what should be a share really is a share, and that the values are reasonable.

```
# make share of homes with a mortgage
block.groups <- transform(block.groups, shr.mort = B25081e2/B25081e1, na.rm = TRUE)
```

```
summary(block.groups$shr.mort)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's 
##  0.0000  0.6193  0.7477  0.7222  0.8551  1.0000     289
```

Now I make a new variable that tells us, for each observation (which is a block group), the quartile of the share of homes with a mortgage into which it falls. In other words, block groups with very few homes with mortgages will be in the bottom quartile, and block groups with almost all homes with mortgages will be in the top quartile. Each block group will be assigned a number 1 through 4 to denote these quartiles (you could plausible also do quintiles or terciles, or the -tile of your choice).

Making quartiles requires a bunch of functions! The most interior function is the `quantile` function. In this function, the first argument is the variable of which you would like to find quantiles. The second argument is `probs=`, which tells R how you want to divide the line [0,1] into quantiles. Here we are dividing into 4, starting at 0 and ending at 1. This function returns a vector of four values.

Surrounding this I have the R function `cut`, which "cuts" the data by the numbers you've inputted, assigning numbers to each cut. So this assigns values 1 through 4 to each block group, by the values of the quartiles you just found.

Then we use the `as.integer()` to make sure this new number is an integer (which will make it easier to use).

And then (finally) the within command does all this "within" the current `block.groups` data frame, rather than creating a stand-alone vector.

In the code below, I do each of these steps individually and then put them all together – this is so you can see what's going on and understand the logic. The final step is really all that's needed from a programming standpoint.

After all that work, I check whether the calculation of quartiles delivers numbers one through four with groups of about equal size by using the `table` function we learned in the first class. I also use `typeof` to see what kind of variable I've created.

```
# divide into quartiles
quants <- quantile(block.groups$shr.mort, probs=0:4/4, na.rm = TRUE)
quants
```

```
##        0%       25%       50%       75%      100% 
## 0.0000000 0.6193365 0.7477064 0.8551165 1.0000000
```

```
cut.quants <- cut(block.groups$shr.mort, quantile(block.groups$mort,
                                                   probs=0:4/4, na.rm = TRUE),
                  include.lowest=TRUE)
```

```
## Warning in is.na(x): is.na() applied to non-(list or vector) of type 'NULL'
```

```
cut.quants[1:10]
```

```
##  [1] <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
## Levels: [,]
```

```
cut.quants.int <- as.integer(cut(block.groups$shr.mort,
                                 quantile(block.groups$shr.mort,
                                          probs=0:4/4, na.rm = TRUE),
                                 include.lowest=TRUE))
cut.quants.int[1:10]
```

```
##  [1]  2  2  2  3 NA  3  2  4  4  4
```

```
block.groups <- within(block.groups, mort.qtile <-
                         as.integer(cut(shr.mort,
                                        quantile(shr.mort, probs=0:4/4,
                                                 na.rm = TRUE),
                                        include.lowest=TRUE)))
table(block.groups$mort.qtile)
```

```
##
##    1    2    3    4
## 2355 2357 2352 2355
```

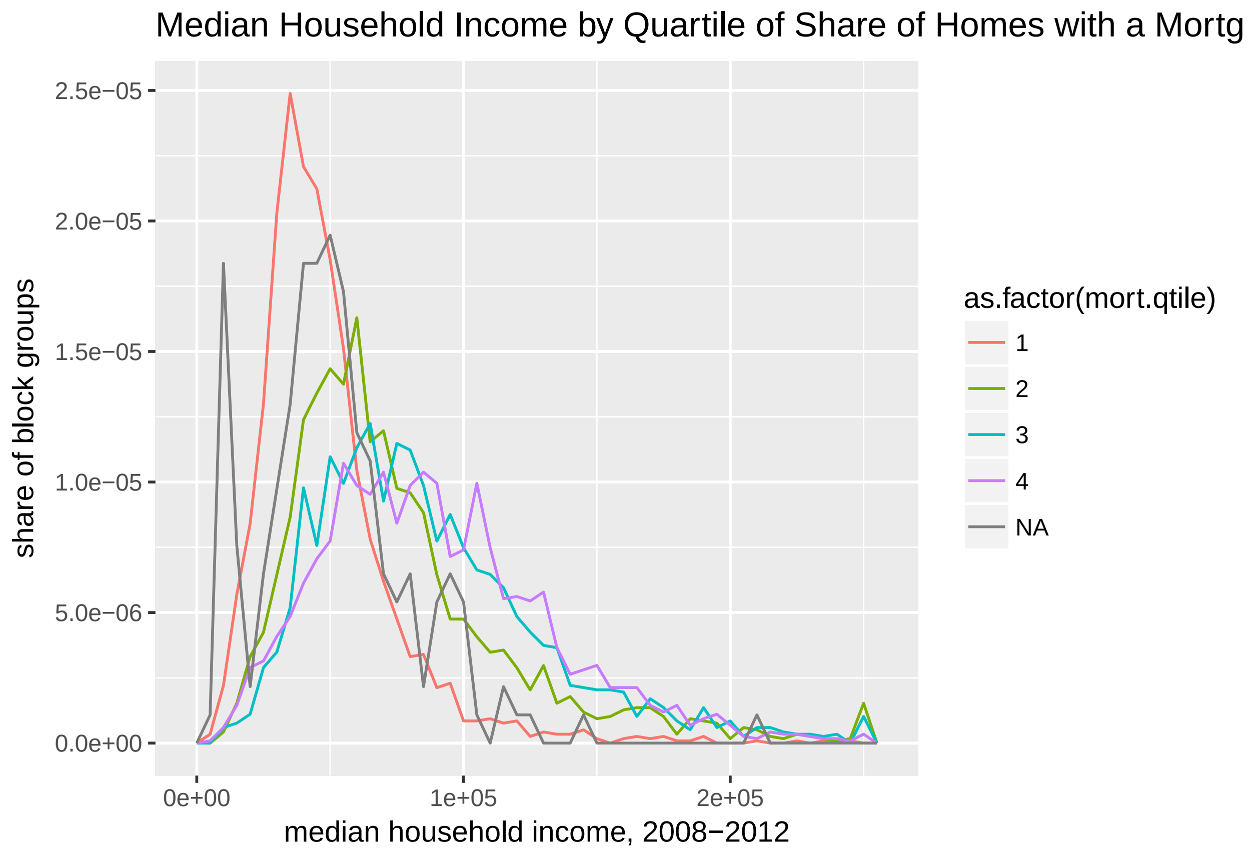
```
typeof(block.groups$mort.qtile)
```

```
## [1] "integer"
```

Having calculated quartiles, I can then plot median household income by share with a mortgage using code very similar to the previous section.

```
# plot by quartiles
ggplot(block.groups, aes(x=B19013e1, y=..density..,  color = as.factor(mort.qtile))) +
  geom_freqpoly(binwidth=5000) +
  ggtitle("Median Household Income by Quartile of Share of Homes with a Mortgage") +
  labs(y="share of block groups", x="median household income, 2008-2012")
```

```
## Warning: Removed 108 rows containing non-finite values (stat_bin).
```



Interpret this final picture. What does it tell you about neighborhoods by share of homes with a mortgage?

G. Try it yourself

1. Choose a new variable for which to make a histogram from the block group data (not income!)

    - plot it by the three states
    - then based on a share you calculate in the data (recall that last class we calculated shares)
2. Make two histograms from a dataset (one dataset is sufficient) we have not used in this class
    - if you want to try an open data DC dataset, great – but use whatever you please
    - write 2 to 3 sentences describing what you find