Tutorial 3: Histograms

Leah Brooks

February 3, 2019

Today's tutorial is an introduction to `ggplot`, the package we'll focus on for most of the rest of the course, and an overview of options for histograms and histogram-like plots.

Histograms are plots that show the distribution of one variable. They are very useful when you want to show details of a variable beyond the mean. For example, if you're interested in showing income inequality, a histogram is one way to visualize income inequality (you can also calculate statistics that summarize inequality; these are more limited descriptions of what you see in a histogram).

In this tutorial, we first work though some histograms with a small annual dataset of hurricanes by year since 1851. After establishing how histograms work, we then turn to a larger dataset of all small neighborhoods in the US, and work on further variations of histograms.

We also introduce some elements of graph legibility such as titles and axis scaling.

A. Load Packages and Small Data

As you did last week, create an R script for this class. Write all your commands in the R script (recall, a file with R commands ending in .R). You can run all of the program at once (code -> run region -> run all), or just selected lines.

Start by loading the ggplot2 package. If you have not already installed it, you must first do that by typing

```r
install.packages("ggplot2", dependencies = TRUE)
```

Once `ggplot2` is loaded, you can tell R (and once per session is enough; I usually put this line at the top of the .R script) to load the package:

```r
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 3.4.4
```

Now you should download the hurricane data, and load them into R using `read.csv()`. These data come from this webpage; look at this page for complete variable definitions. I copied the online table into Excel and saved as a csv file.

```r
# load hurricane data
hurr <- read.csv("H:/pppa_data_viz/2019/tutorial_data/lecture03/2019-02-02_hurricaines_by_year.csv")
```

Take a quick look at these data. What variables does it have? What do the first five observations look like (`head`)? And what types of variables does it have (`str`)?

```r
# look at variables and types
names(hurr)
```

```
## [1] "year"         "Named.Storms" "Hurricanes"   "Major"
```

```r
head(hurr)
```

```
##   year Named.Storms Hurricanes Major
## 1 1851            6          3     1
## 2 1852            5          5     1
## 3 1853            8          4     2
## 4 1854            5           3     1
## 5 1855            5           4     1
## 6 1856            6           4     2
```

```
str(hurr)
```

```
## 'data.frame':    168 obs. of  4 variables:
##  $ year        : int  1851 1852 1853 1854 1855 1856 1857 1858 1859 1860 ...
##  $ Named.Storms: int  6 5 8 5 5 6 4 6 8 7 ...
##  $ Hurricanes  : int  3 5 4 3 4 4 3 6 7 6 ...
##  $ Major       : int  1 1 2 1 1 2 0 0 1 1 ...
```

B. Make a simple histogram

B.1. Check on data

Our first goal is to make a histogram of the number of hurricanes. How many years have which numbers of hurricanes? Before making a histogram, let's start by making a table shows the data we'll use in the histogram. We do this to check on data quality and to make sure that we are graphing something that can be graphed with a histogram.

We use the command `table` that we've used in the last tutorial.

```
# make sure we know what data will tell us
table(hurr$Major)
```

```
## 
##  0  1  2  3  4  5  6  7  8 
## 31 48 45 15  9  9  7  2  1
```

This looks pretty good – except that there is one `NA` observation. This observation will give error messages for graphs and generally cause problems. Let's look into why and where we have this observation. First I use `summary` to tell us about the variable and to be sure that there is a problem. Then I print just a subset of the dataframe `hurr` where the variable `Major` takes on a value of missing.

```
# fix data problem
summary(hurr$Major)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's 
##   0.000   1.000   2.000   1.964   3.000   8.000       1
```

```
hurr[which(is.na(hurr$Major) == TRUE),]
```

```
##     year Named.Storms Hurricanes Major
## 168   NA           NA         NA    NA
```

I notice that this error occurs in a row without a value for the variable `year`. I check that this is the last row of the dataframe by printing the dimensions of the dataframe. Once I learn that this just seems to be an extra row, I use another subsetting command to take the dataframe without the problematic line.

```
dim(hurr)
```

```
## [1] 168   4
```

```
hurr <- hurr[1:167,]
```

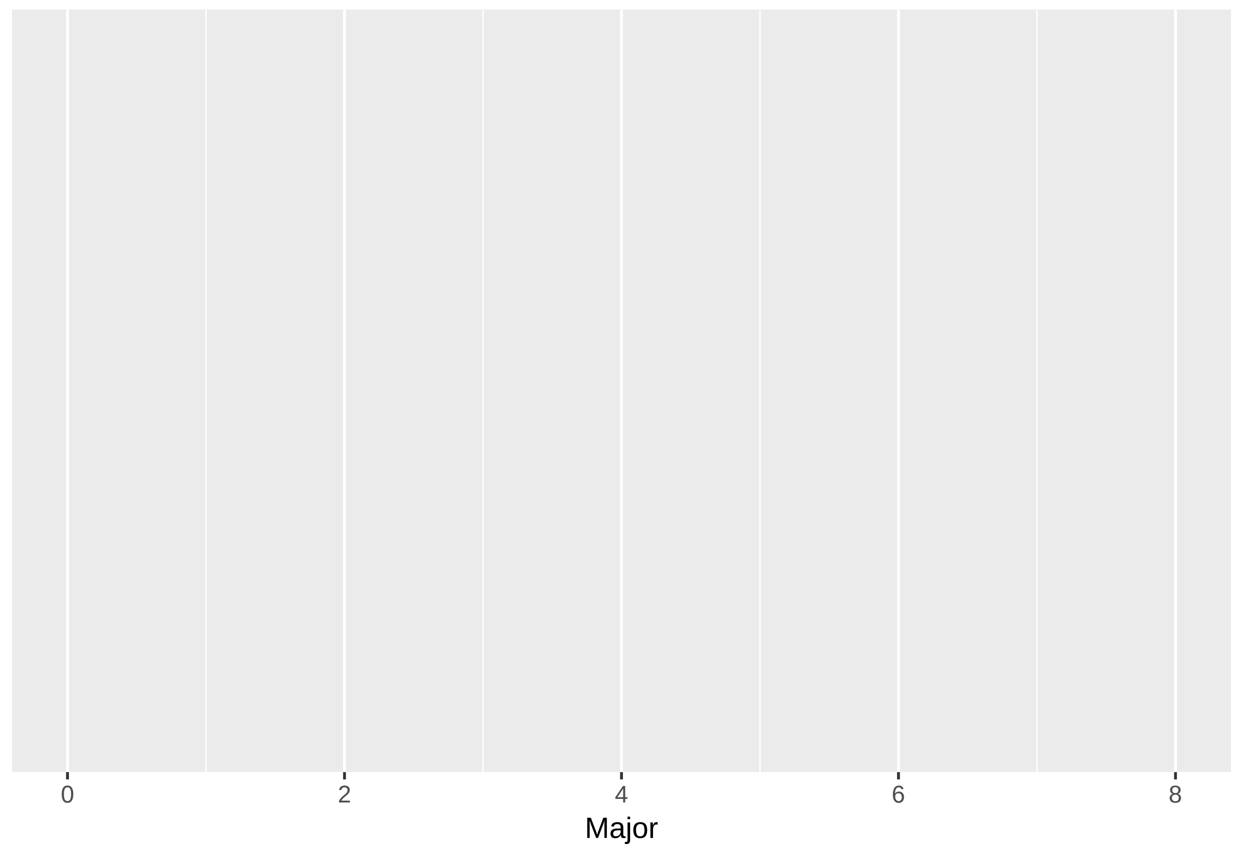
B.2. Basic histograms

We begin by briefly discussing the key elements of making a graph with `ggplot`. The three things R needs to make a graph are (i) the dataframe, (ii) the variable you want to graph and (iii) the type of graph you'd like to make.

R's `ggplot` is set up so that you can either use the same dataframe for all types of graph within one call, or you can name different dataframes in each graph that you combine. We'll begin with the first possibility today and move on to the second later in the class.

If you just tell R that you want to use `ggplot` and specify only the variable of interest, you will get a blank plot:

```r
# ggplot basics
c1 <- ggplot(data = hurr, aes(Major))
c1
```



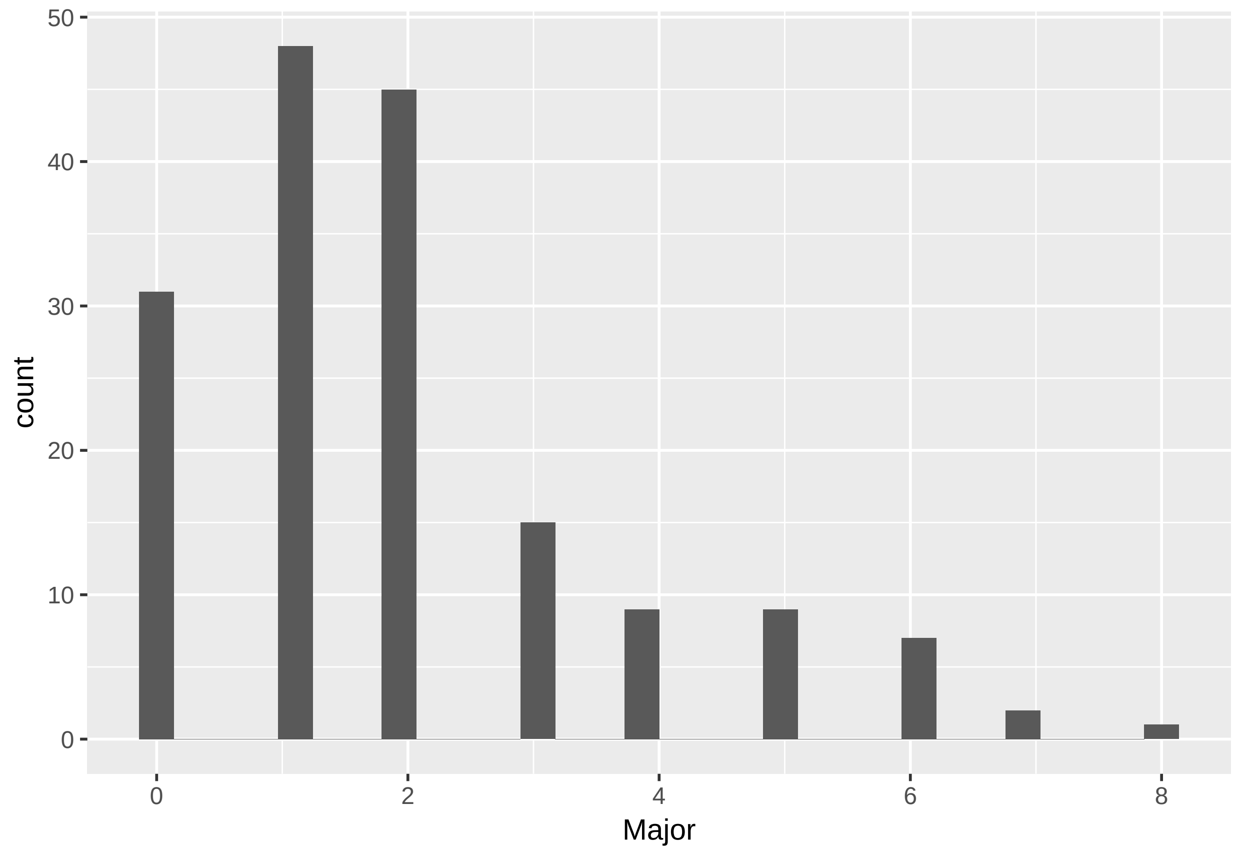
Note that I am creating the chart as an object `c1`, and I can refer to c1 elsewhere in the program if I like. I can add to it, as we'll see later, and I can output it.

To get an actual plot, I need to add a `geom`. The ggplot world is based on this `geom` command, which you add to the ggplot command to get a graph. There are tons of kinds of `geoms` that you can use, and we'll explore ones for scatter plots, bar plots and many more in this class.

A small programming note: R will fail if you put the `+` on the second line. So make sure that you never start a continuing line of `ggplot` with a plus. See here for details.

```r
# how many hurricanes by year?
c1 <- ggplot(data = hurr) +
       geom_histogram(aes(Major))
c1
```

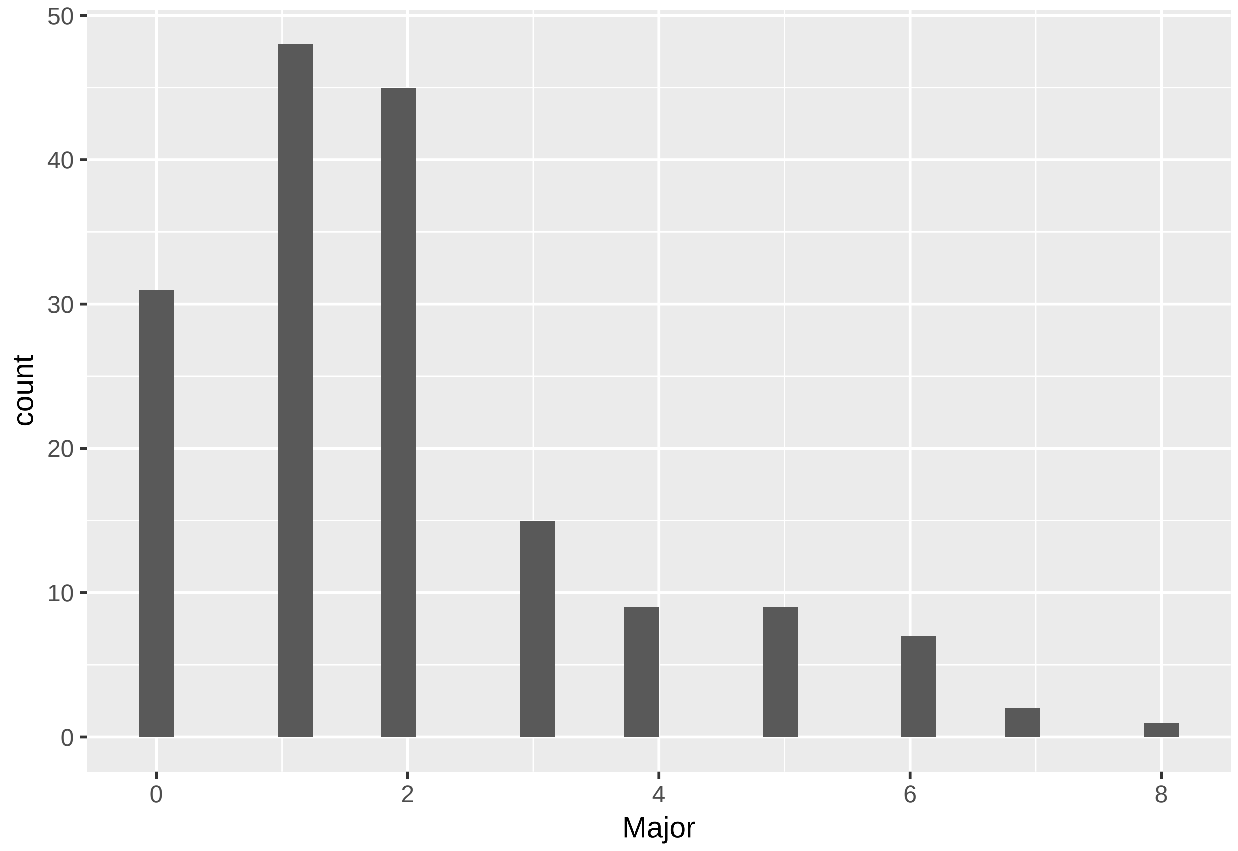
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



It is entirely equvalent to write the graph this way:

```
# how many hurricanes by year?
c1 <- ggplot() +
       geom_histogram(data = hurr, aes(Major))
c1
```

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



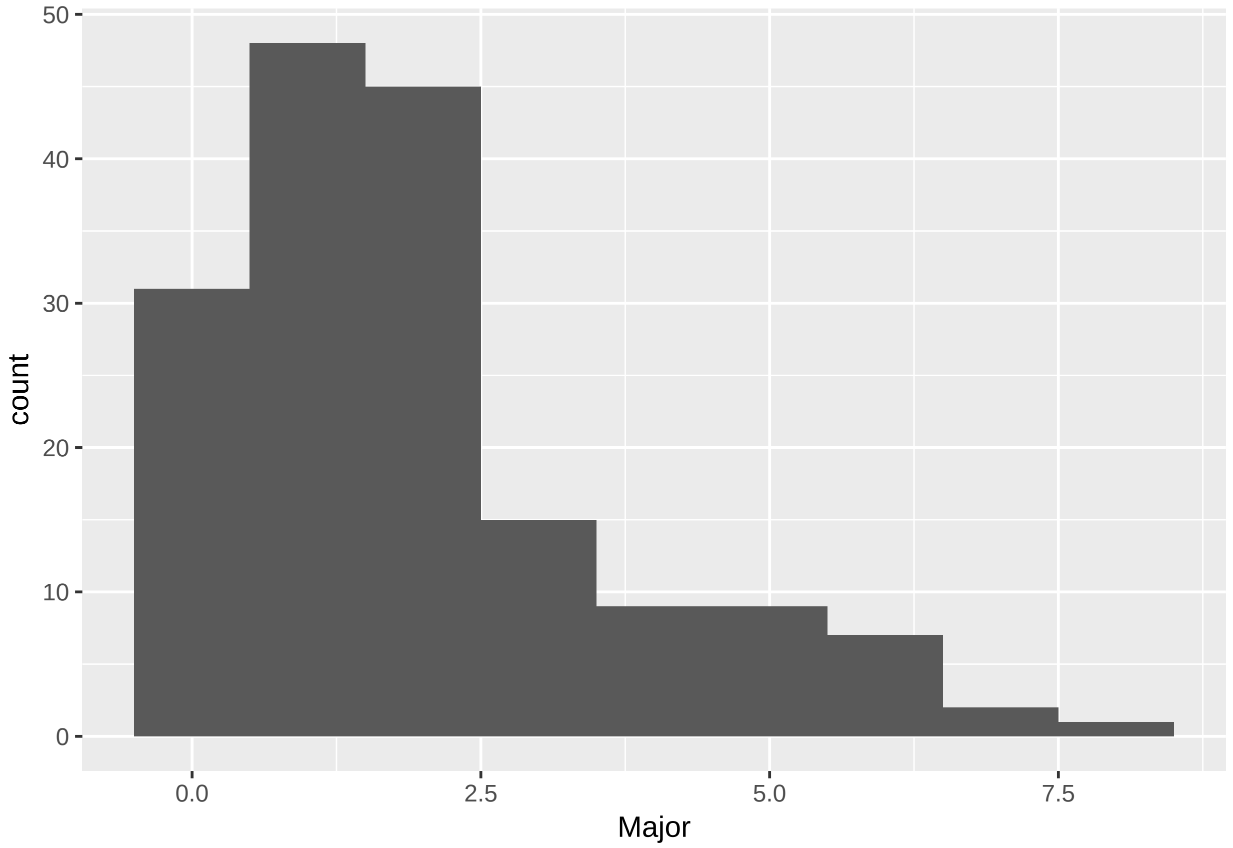
This second format is preferred if you think you'll be using multiple dataframes.

Note that you can also just write the ggplot command directly without assigning to an object, as below. I usually don't do this, because I eventually want to save my graph (something we'll learn in a future class) and to tell R to save something, you need to have an object to save.

```
# how many hurricanes by year?
ggplot() +
       geom_histogram(data = hurr, aes(Major))
```

All of those charts look odd. There are only 9 categories, not the strange number that the x axis above reflects. We can change this by telling R the the number of bins we'd like to use as below.

```
# we know there are only 9 bins. fix to reflect
c1 <- ggplot(data = hurr) +
        geom_histogram(aes(Major), bins = 9)
c1
```



However, the axis labels in the above are entirely unhelpful. We now do two things to make the axes more legible. First, we modify axis labels by using the `labs` command to add in x and y axis labels. You can also use this command to get rid of labels.

Second, to change the numeric labeling on the x axis, I use the `scale_x_continuous(breaks = seq(0,8,1))` to tell R to use numbers 0 to 8 as the labeled breakpoints on the horizontal axis. The command `seq(0,8,1)` means "create a sequence starting at 0, going to 8, by 1." Alternatively, I could have written `scale_x_contintuous(breaks = c(0,1,2,3,4,5,6,7,8))`, but the first one is shorter, cleaner and easier to modify if you want to later make changes.

The sequence framework is easily modifiable. For example, `seq(0,1,0.25)` means "create a sequences starting at 0, going to 1, by 1/4." Try it and see:

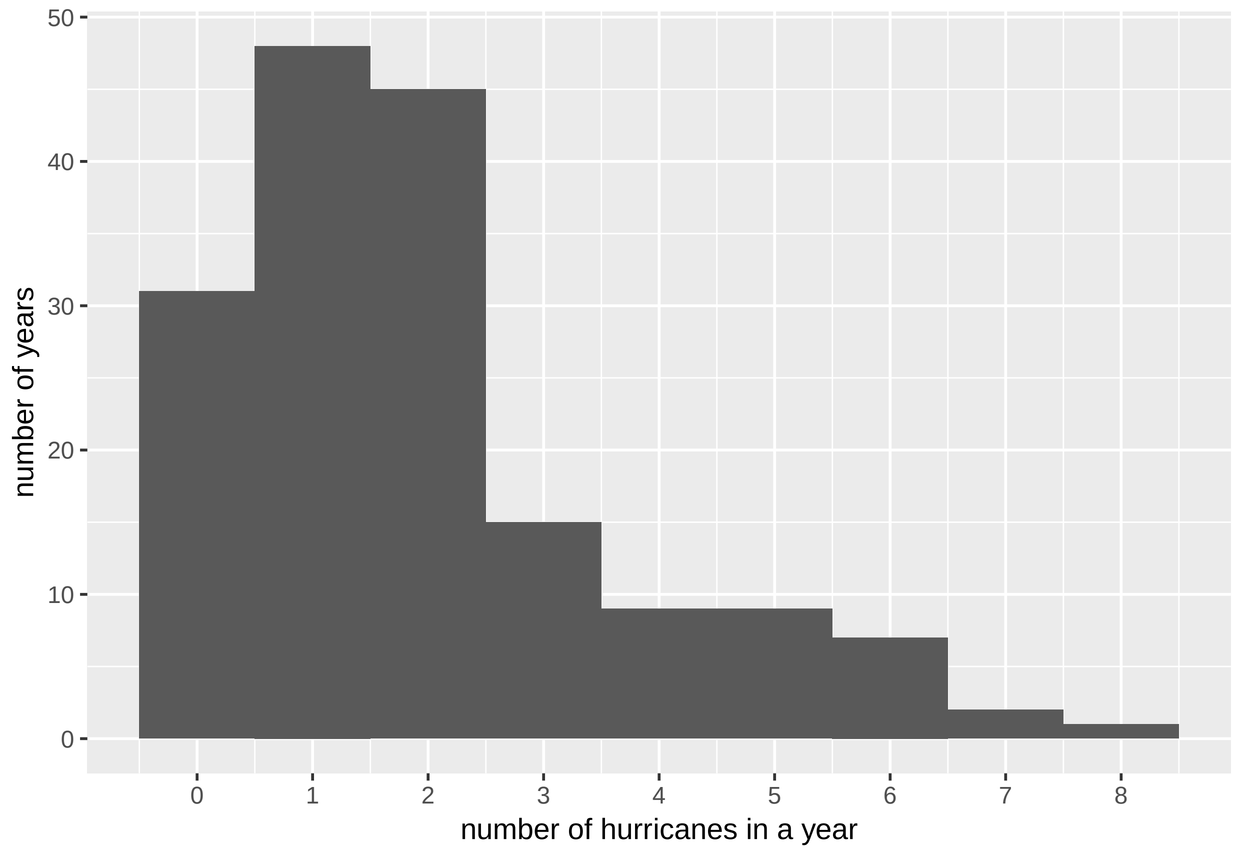
```
my.seq <- seq(0,1,0.25)
my.seq
```

```
## [1] 0.00 0.25 0.50 0.75 1.00
```

Does the sequence look as you expected?

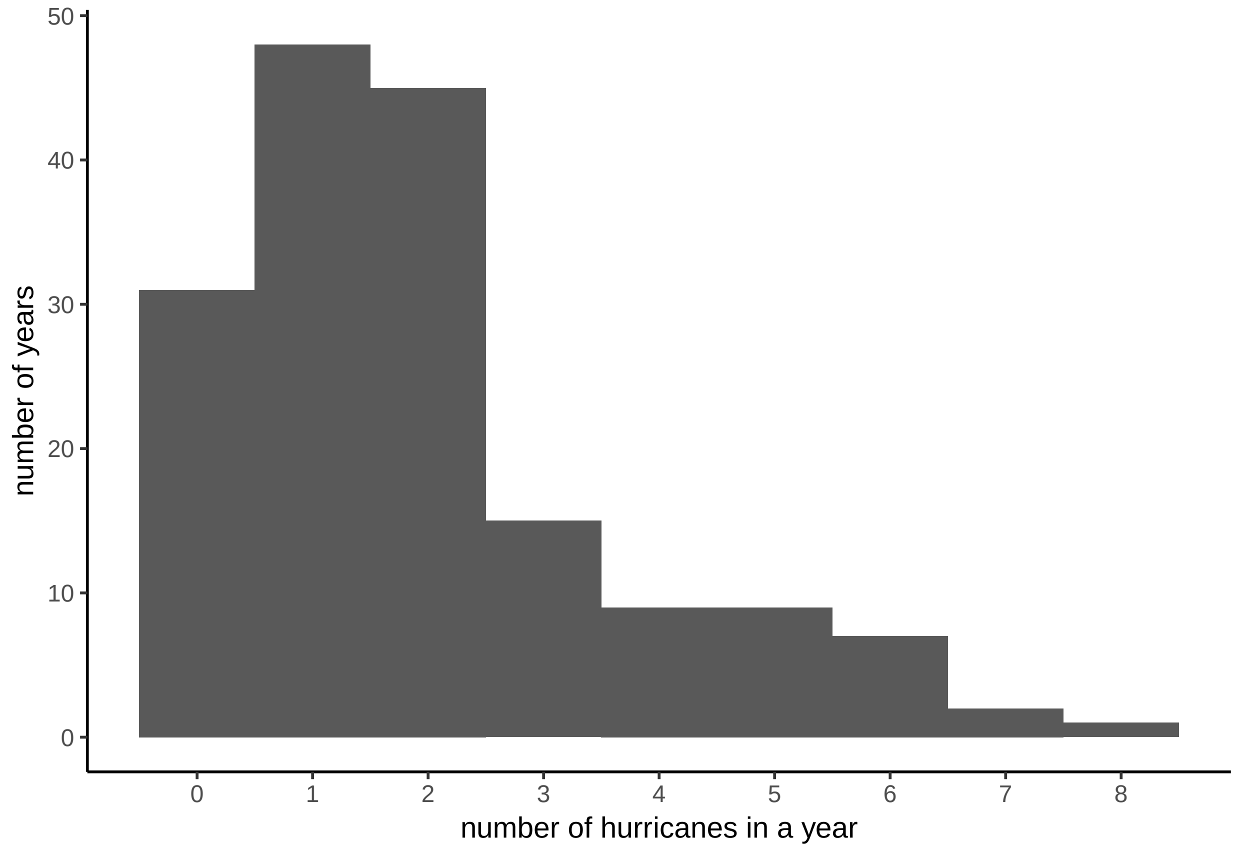
And now for the new graph with both additions:

```r
# yes, but nutty labels!
c1 <- ggplot(data = hurr) +
        geom_histogram(aes(Major), bins = 9) +
        scale_x_continuous(breaks = seq(0,8,1)) +
        labs(x = "number of hurricanes in a year",
             y = "number of years")
c1
```



I find the background in the above chart distracting and unhelpful. Like almost everything in `ggplot`, the background is also modifiable. The background and the types of axes, and many other things are parts of the "theme" of the graph, and you modify them with the `theme()` command. You can see the zillions of full options for modification here. Right now, we'll just get rid of the grid and the background color. Here I use `panel.grid`, but you can adjust both the major and minor grids with `panel.grid.major` and `panel.grid.minor`. "Getting rid" means setting to `element_blank()`. I also set the axis lines black.

```
# get rid of crazy background that is confusing
c1 <- ggplot(data = hurr) +
        geom_histogram(aes(Major), bins = 9) +
        scale_x_continuous(breaks = seq(0,8,1)) +
        labs(x = "number of hurricanes in a year",
             y = "number of years") +
        theme(panel.grid = element_blank(),
              panel.background = element_blank(),
              axis.line = element_line(colour = "black"))
c1
```



This histogram above is doing one job: telling us the overall distribution of number of major hurricaines by year. Suppose we want to convey some additional information – perhaps we'd like to show that the number of hurricaines by century is on the rise.

Below is an interesting but not-perfect technique for doing this. We color in the bars to reflect the number of observations that come from each century. This is visually clear, but somewhat misleading because each century does not have the same number of observations, so three observations from the 2000s should carry more weight than three observations from the 1900s. We are not going to deal with this concern here, but return to ways to compare distributions later in this tutorial.

To color bars by century, we first need a variable that marks century. I do this with an `ifelse()` command. I then check my work with a `table()` command, making sure that I have roughly the number of observations for each century that I think I should.

```
# color sections of bars?
summary(hurr$year)
```

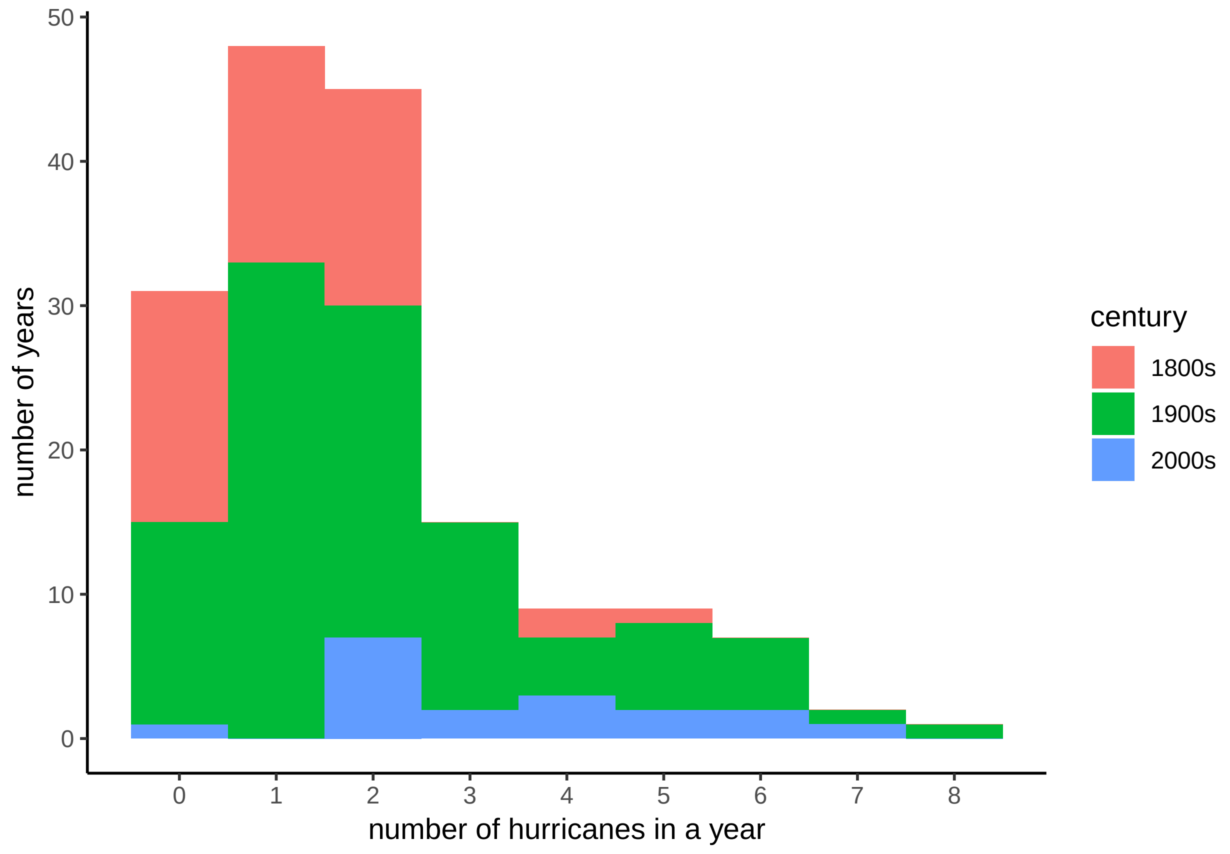
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##    1851    1892    1934    1934    1976    2017
```

```
hurr$century <- ifelse(hurr$year < 1900,"1800s",
                       ifelse(hurr$year >= 1900 & hurr$year < 2000,"1900s","2000s"))
table(hurr$century)
```

```
## 
## 1800s 1900s 2000s 
##    49   100    18
```

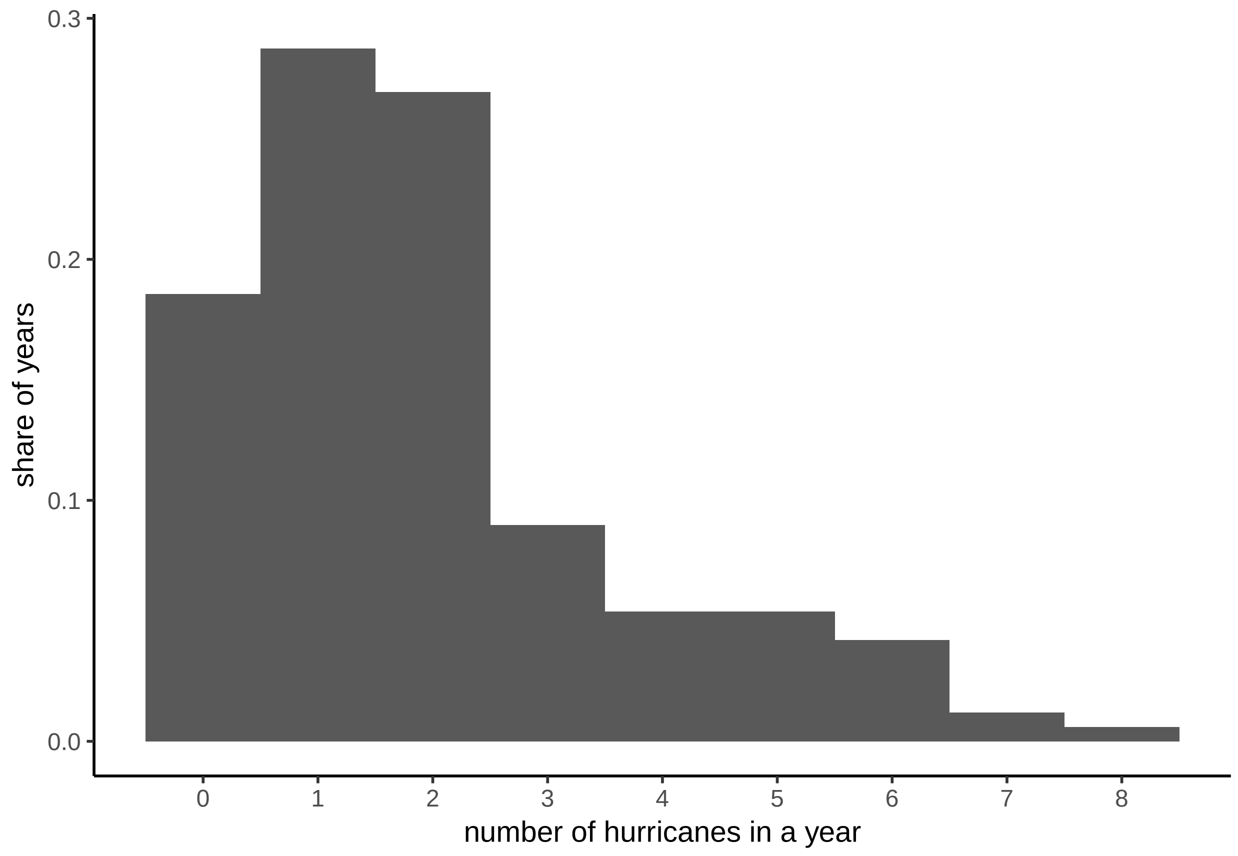
To add color to the graph, I add to the "aesthetics" command for the graph. I tell R to fill in the bars by the century, using `fill = century`.

```r
# color bars by century
c1 <- ggplot(data = hurr) +
  geom_histogram(aes(x = Major, fill = century), bins = 9) +
  scale_x_continuous(breaks = seq(0,8,1)) +
  labs(x = "number of hurricanes in a year",
       y = "number of years") +
  theme(panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.background = element_blank(),
        axis.line = element_line(colour = "black"))
c1
```



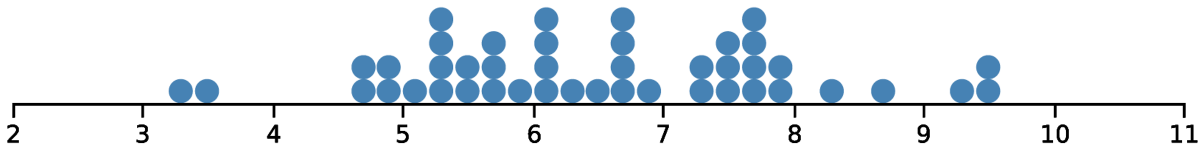
All the graphs we've made till now use the number of observations on the vertical axis. Sometimes it is more useful to show shares rather than numbers. You can do this easily by telling R that the y axis is a share: `y = stat(count / sum(count))`. This will change the numbering on the y axis, but not the height of the bars (why?).

```r
# show as percentage, rather than number
c1 <- ggplot(data = hurr) +
  geom_histogram(aes(y = stat(count / sum(count)), x = Major), bins = 9) +
  scale_x_continuous(breaks = seq(0,8,1)) +
  labs(x = "number of hurricanes in a year",
       y = "share of years") +
  theme(panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.background = element_blank(),
        axis.line = element_line(colour = "black"))
c1
```



I wanted to also show you how you could make this plot with dots instead of bars, sort of like the example below. However, this requires using a new `geom` command, so we'll save it for later in the semester.

```r
knitr::include_graphics("https://camo.githubusercontent.com/a78925c179daccb698eccc463af675627f65dbf1/68
```



C. Load and examine a new dataset

Now we are going to do more histogram examples with a larger dataset where you cannot see all the values. We do this to work on your skills with larger datasets and to have enough data to make interesting histogram comparisons.

Specifically, we are using data on neighborhoods called block groups. A block group is a neighborhood of typically between 600 to 3,000 people. You can find examples of block groups by looking here.

For each block group, we observe data on people and housing. These data come from surveys conducted by the Census Bureau and are 5-year averages from 2008-2012.

For each block group, we observe a variety of characteristics. Use this Census provided dictionary to understand variables. Look at this file and you will see what the Census calls “tables.” Table B00001 is total population, and the variable in the dataset that relates to this table is called B00001e1 (e is for estimate). Similarly, B00002e1 is the number of housing units, and B01001e2 is number of males under 5 years of age.

Be aware that

- the file I created does not include all of these variables (there are more than 10,000)
- I loaded variables from sequence numbers 1, 4, 9, 19, 41, 43, 58, 59, 62, 63, 64, 78, 81, 83, 105, and 106
- not all variables are available at the block group level. This file tells you whether variables are available at the block group level.

For further information, consult documentation from the American Community Survey.

A block group is uniquely identified by the variables state + county + tract + blkgrp.

This file contains just data from VA, MD and DC so as to be of a manageable size. If at some point you want the whole file (6G), let me know.

Download the data from here

Remember where you saved your data, and use that path in the `read.csv()` command below. We also use some of the commands we’ve seen before to explore the data.

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

# How big is it?
dim(block.groups)
```

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

```
# What variables does it have?
str(block.groups)
```

```
## 'data.frame':    9708 obs. of  3063 variables:
##  $ FILEID     : Factor w/ 1 level "ACSSF": 1 1 1 1 1 1 1 1 1 1 ...
##  $ STUSAB     : Factor w/ 3 levels "dc","md","va": 1 1 1 1 1 1 1 1 1 1 ...
##  $ SUMLEVEL   : int  150 150 150 150 150 150 150 150 150 150 ...
##  $ COMPONENT  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ LOGRECNO   : int  367 368 369 370 371 372 373 374 375 376 ...
##  $ US         : logi  NA NA NA NA NA NA ...
##  $ REGION     : logi  NA NA NA NA NA NA ...
##  $ DIVISION   : logi  NA NA NA NA NA NA ...
##  $ STATECE    : logi  NA NA NA NA NA NA ...
##  $ STATE      : int  11 11 11 11 11 11 11 11 11 11 ...
##  $ COUNTY     : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ COUSUB     : logi  NA NA NA NA NA NA ...
##  $ PLACE      : logi  NA NA NA NA NA NA ...
##  $ TRACT      : int  100 100 100 100 201 202 202 202 202 300 ...
##  $ BLKGRP     : int  1 2 3 4 1 1 2 3 4 1 ...
##  $ CONCIT     : logi  NA NA NA NA NA NA ...
##  $ CSA        : logi  NA NA NA NA NA NA ...
##  $ METDIV     : logi  NA NA NA NA NA NA ...
##  $ UA         : logi  NA NA NA NA NA NA ...
##  $ UACP       : logi  NA NA NA NA NA NA ...
##  $ VTD        : logi  NA NA NA NA NA NA ...
##  $ ZCTA3      : logi  NA NA NA NA NA NA ...
##  $ SUBMCD     : logi  NA NA NA NA NA NA ...
##  $ SDELM      : logi  NA NA NA NA NA NA ...
##  $ SDSEC      : logi  NA NA NA NA NA NA ...
##  $ SDUNI      : logi  NA NA NA NA NA NA ...
##  $ UR         : logi  NA NA NA NA NA NA ...
##  $ PCI        : logi  NA NA NA NA NA NA ...
##  $ TAZ        : logi  NA NA NA NA NA NA ...
##  $ UGA        : logi  NA NA NA NA NA NA ...
##  $ GEOID      : Factor w/ 9708 levels "15000US110010001001",..: 1 2 3 4 5 6 7 8 9 10 ...
##  $ NAME       : Factor w/ 9708 levels "Block Group 0, Census Tract 751.01, Suffolk city, Virginia",.
##  $ AIANHH     : logi  NA NA NA NA NA NA ...
##  $ AIANHHFP   : logi  NA NA NA NA NA NA ...
##  $ AIHHTLI    : logi  NA NA NA NA NA NA ...
##  $ AITSCE     : logi  NA NA NA NA NA NA ...
##  $ AITS       : logi  NA NA NA NA NA NA ...
##  $ ANRC       : logi  NA NA NA NA NA NA ...
##  $ CBSA       : logi  NA NA NA NA NA NA ...
##  $ MACC       : logi  NA NA NA NA NA NA ...
##  $ MEMI       : logi  NA NA NA NA NA NA ...
##  $ NECTA      : logi  NA NA NA NA NA NA ...
##  $ CNECTA     : logi  NA NA NA NA NA NA ...
##  $ NECTADIV   : logi  NA NA NA NA NA NA ...
##  $ CDCURR     : logi  NA NA NA NA NA NA ...
##  $ SLDU       : logi  NA NA NA NA NA NA ...
##  $ SLDL       : logi  NA NA NA NA NA NA ...
##  $ ZCTA5      : logi  NA NA NA NA NA NA ...
##  $ PUMA5      : logi  NA NA NA NA NA NA ...
##  $ PUMA1      : logi  NA NA NA NA NA NA ...
##  $ BTTR       : logi  NA NA NA NA NA NA ...
##  $ BTBG       : logi  NA NA NA NA NA NA ...
##  $ FILETYPE   : num  2.01e+08 2.01e+08 2.01e+08 2.01e+08 2.01e+08 ...
##  $ CHARITER   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ SEQUENCE   : int  106 106 106 106 106 106 106 106 106 106 ...
##  $ B00001e1   : int  63 75 80 54 423 101 60 76 204 65 ...
##  $ B00002e1   : int  40 52 33 30 0 46 29 37 44 33 ...
##  $ B02001e1   : int  1296 1322 1430 992 4074 1268 869 976 1863 1209 ...
##  $ B02001e2   : int  1155 1204 1241 924 3079 1131 800 922 1599 1137 ...
##  $ B02001e3   : int  0 27 0 0 323 12 0 0 96 0 ...
##  $ B02001e4   : int  0 0 81 0 0 0 0 0 14 12 ...
##  $ B02001e5   : int  44 46 108 68 500 81 59 7 88 40 ...
##  $ B02001e6   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ B02001e7   : int  97 22 0 0 16 32 0 0 0 20 ...
##  $ B02001e8   : int  0 23 0 0 156 12 10 47 66 0 ...
##  $ B02001e9   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ B02001e10  : int  0 23 0 0 156 12 10 47 66 0 ...
##  $ B02005e1   : logi  NA NA NA NA NA NA ...
##  $ B02005e2   : logi  NA NA NA NA NA NA ...
##  $ B02005e3   : logi  NA NA NA NA NA NA ...
##  $ B02005e4   : logi  NA NA NA NA NA NA ...
##  $ B02005e5   : logi  NA NA NA NA NA NA ...
##  $ B02005e6   : logi  NA NA NA NA NA NA ...
##  $ B02005e7   : logi  NA NA NA NA NA NA ...
##  $ B02005e8   : logi  NA NA NA NA NA NA ...
##  $ B02005e9   : logi  NA NA NA NA NA NA ...
##  $ B02005e10  : logi  NA NA NA NA NA NA ...
##  $ B02005e11  : logi  NA NA NA NA NA NA ...
##  $ B02005e12  : logi  NA NA NA NA NA NA ...
##  $ B02005e13  : logi  NA NA NA NA NA NA ...
##  $ B02005e14  : logi  NA NA NA NA NA NA ...
##  $ B02005e15  : logi  NA NA NA NA NA NA ...
##  $ B02005e16  : logi  NA NA NA NA NA NA ...
##  $ B02005e17  : logi  NA NA NA NA NA NA ...
##  $ B02005e18  : logi  NA NA NA NA NA NA ...
##  $ B02005e19  : logi  NA NA NA NA NA NA ...
##  $ B02005e20  : logi  NA NA NA NA NA NA ...
##  $ B02005e21  : logi  NA NA NA NA NA NA ...
##  $ B02005e22  : logi  NA NA NA NA NA NA ...
##  $ B02005e23  : logi  NA NA NA NA NA NA ...
##  $ B02005e24  : logi  NA NA NA NA NA NA ...
##  $ B02005e25  : logi  NA NA NA NA NA NA ...
##  $ B02005e26  : logi  NA NA NA NA NA NA ...
##  $ B02005e27  : logi  NA NA NA NA NA NA ...
##  $ B02005e28  : logi  NA NA NA NA NA NA ...
##  $ B02005e29  : logi  NA NA NA NA NA NA ...
##  $ B02005e30  : logi  NA NA NA NA NA NA ...
##  $ B02005e31  : logi  NA NA NA NA NA NA ...
##  $ B02005e32  : logi  NA NA NA NA NA NA ...
##   [list output truncated]
```

D. Basic histograms, bigger data

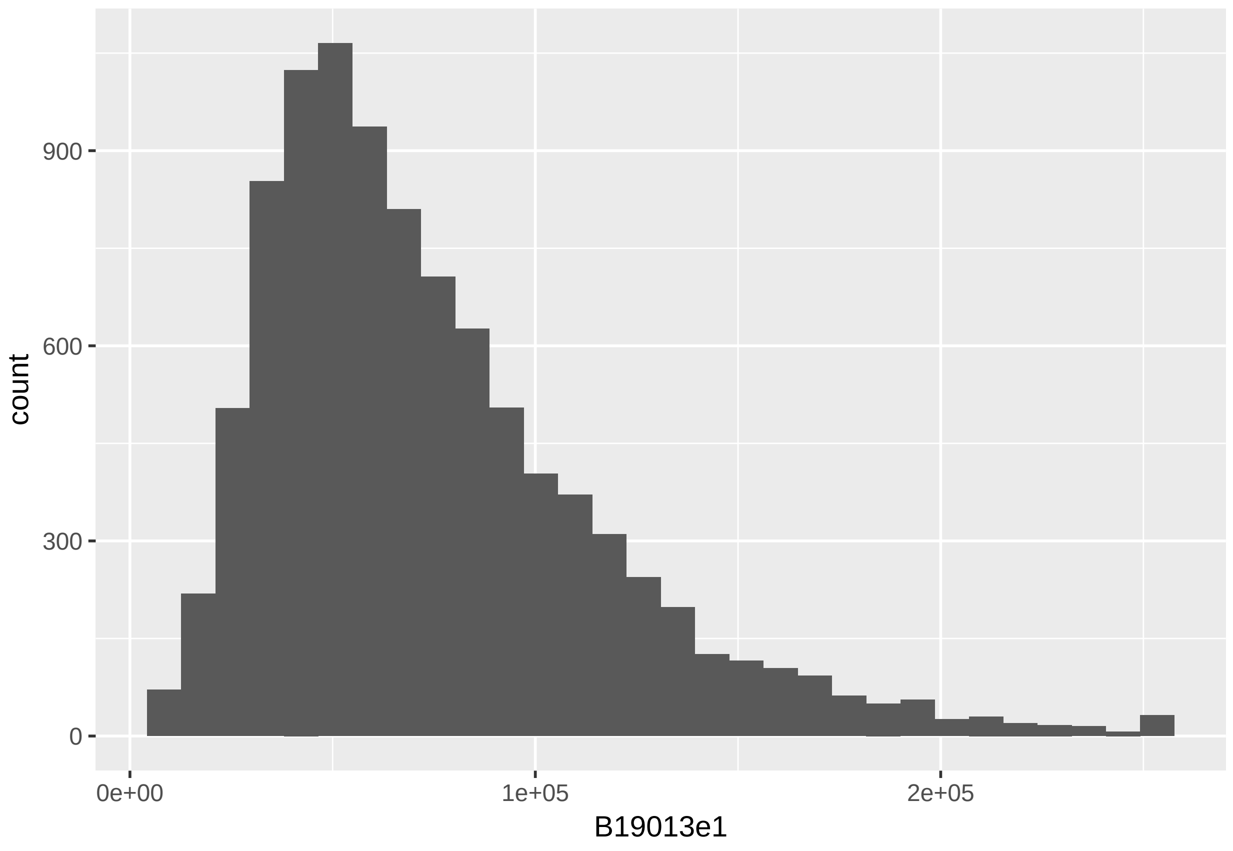
Let's return to histograms and plot the distribution of median household income over the past 12 months, or `B19013e1`.

Here is what R does with the most basic command:

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

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

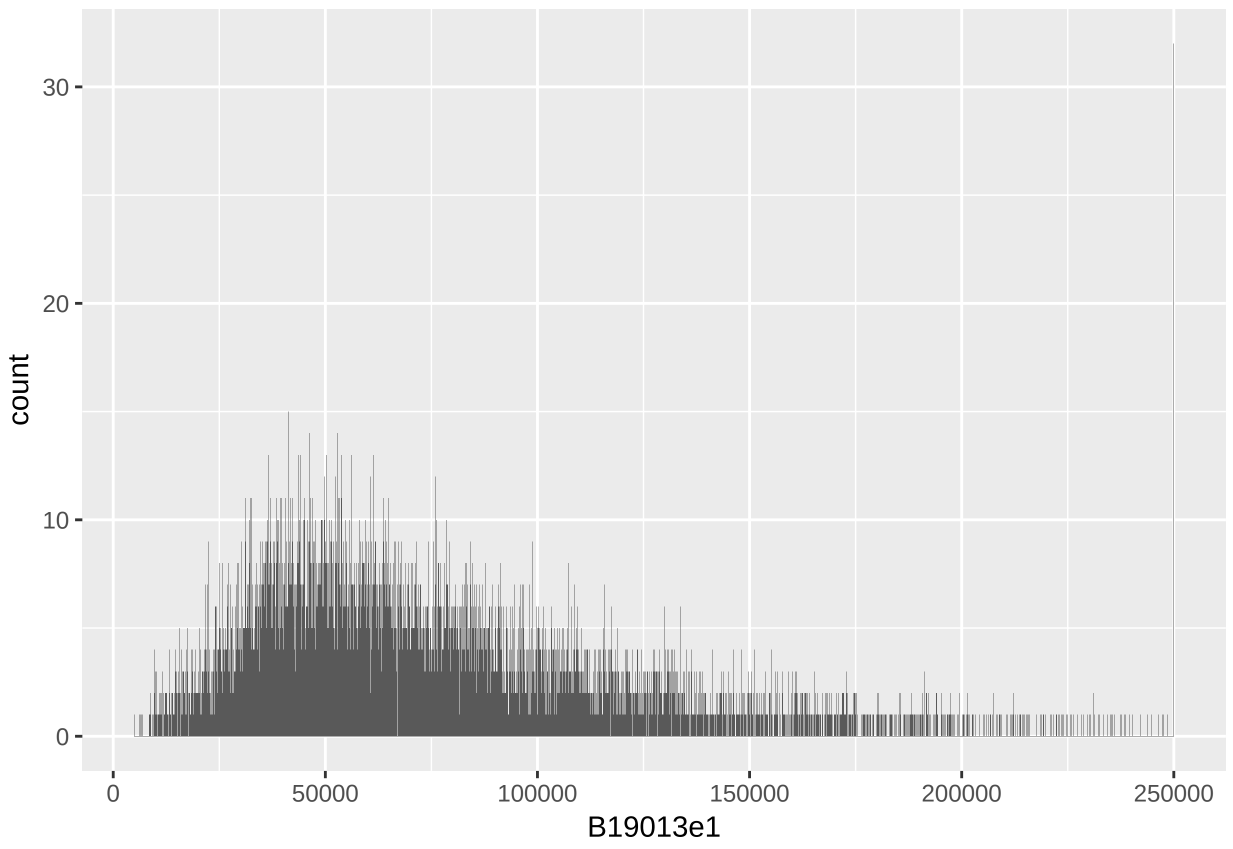
```
## Warning: Removed 108 rows containing non-finite values (stat_bin).
```



Now that we have a wealth of data, we can explore how changing the bin width (the width of the vertical bar, or how many values of income you group together) impacts the look of the graph.

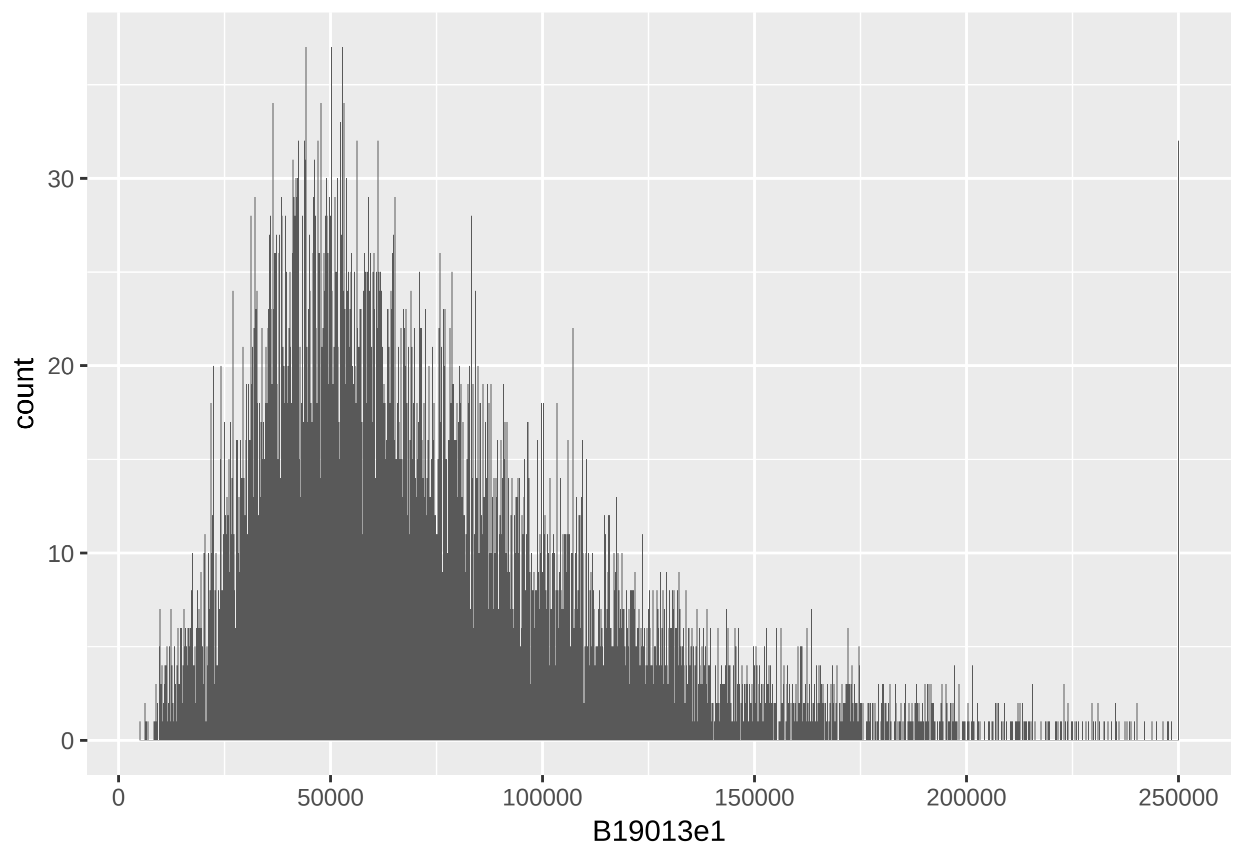
```
## here we begin changing the width of the histogram bins
ggplot(data = block.groups) + geom_histogram(aes(x=B19013e1), binwidth = 50)
```

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



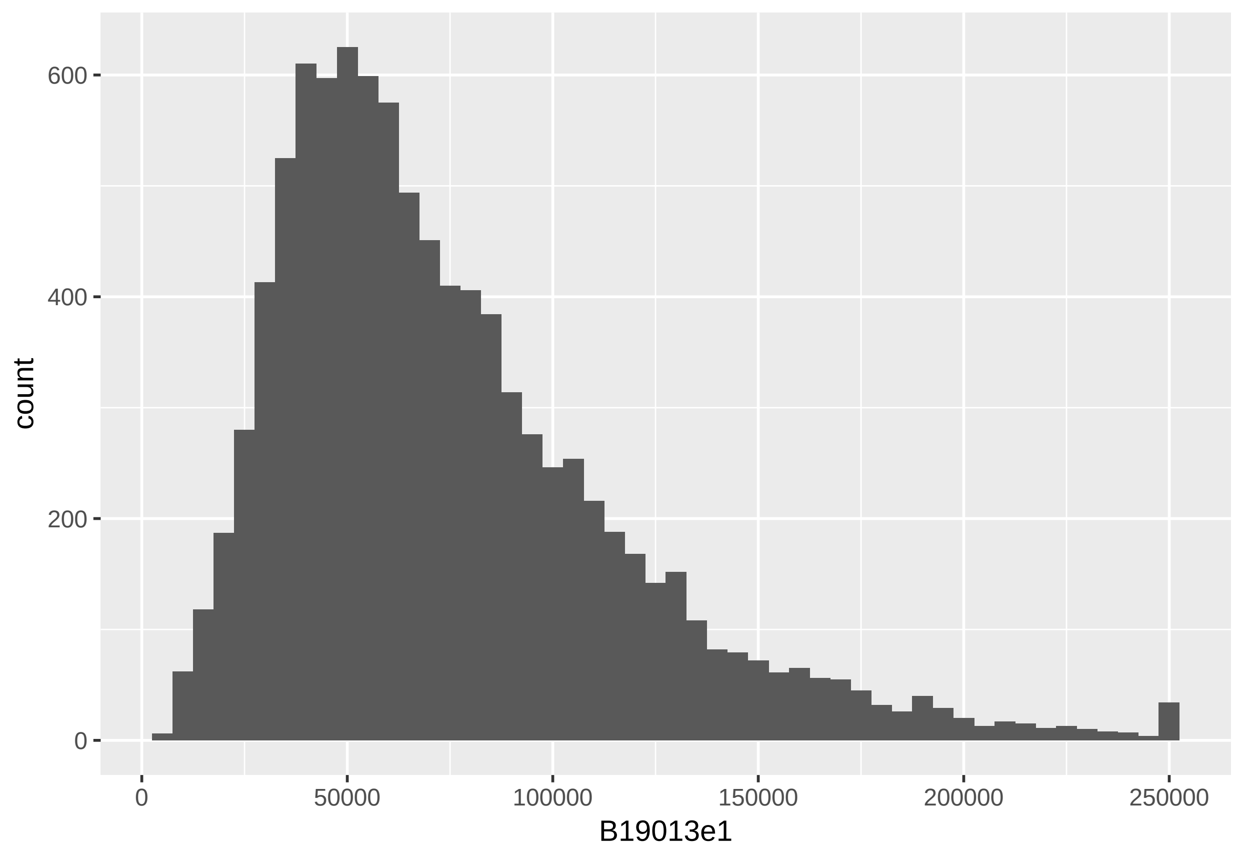
```
## again
ggplot(data = block.groups) + geom_histogram(aes(x=B19013e1), binwidth = 200)
```

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



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

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



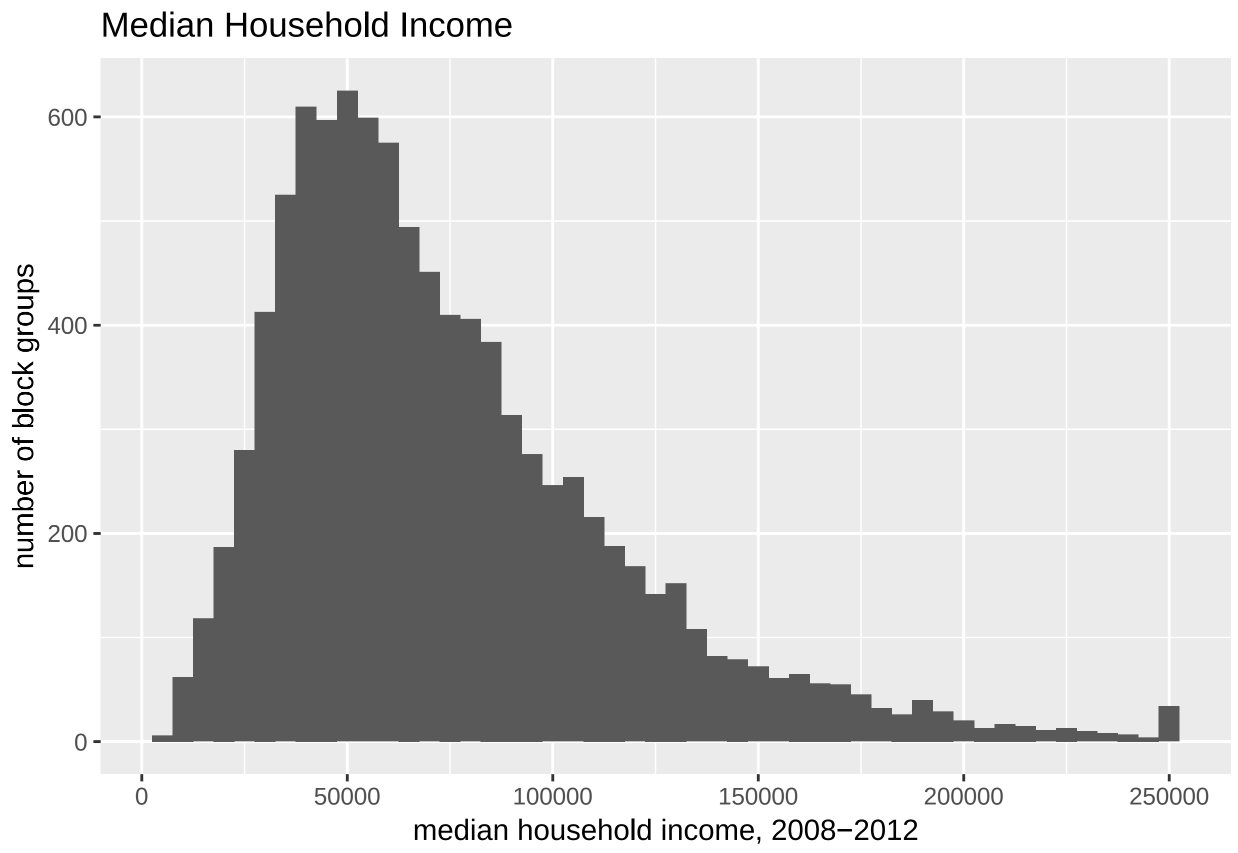
Note that the choice of bin is very important to the final look of the graph. Also notice that very small bins make the top-coded final category (for all block groups with a median income greater than 250,000, the Census reports 250,000) look big. Why is this?

E. 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.) Here again we use `labs` for labels, and I introduce `ggtitle` for the main plot title.

```
## make things legible: titles and axes
ggplot(data = block.groups) + geom_histogram(aes(x=B19013e1), 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).
```



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

F. Other ggplot options for types of histograms

Here we explore density curves and subsets.

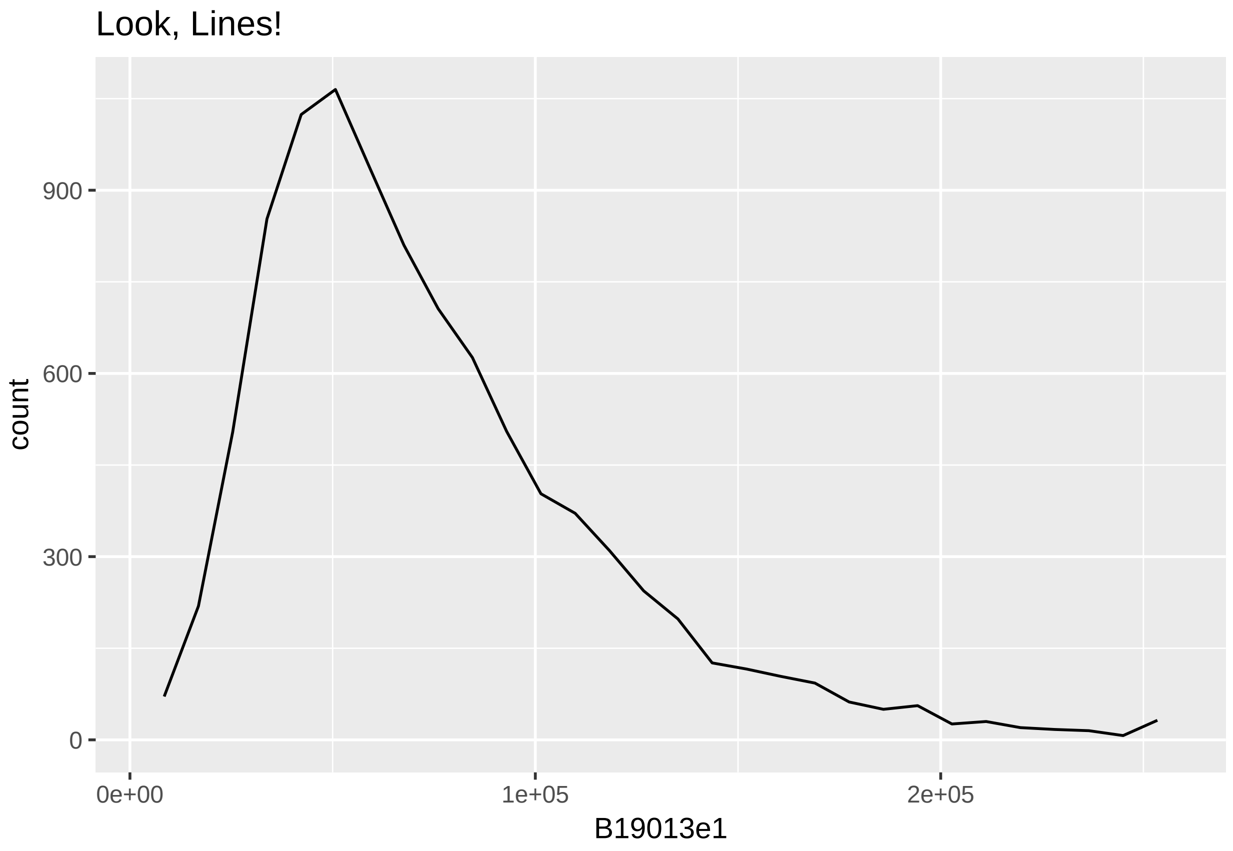
F.1. Density curves

We begin by using curves. One way to get a curve is to use `stat_bin()`, which calculates statistics by bin.

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

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

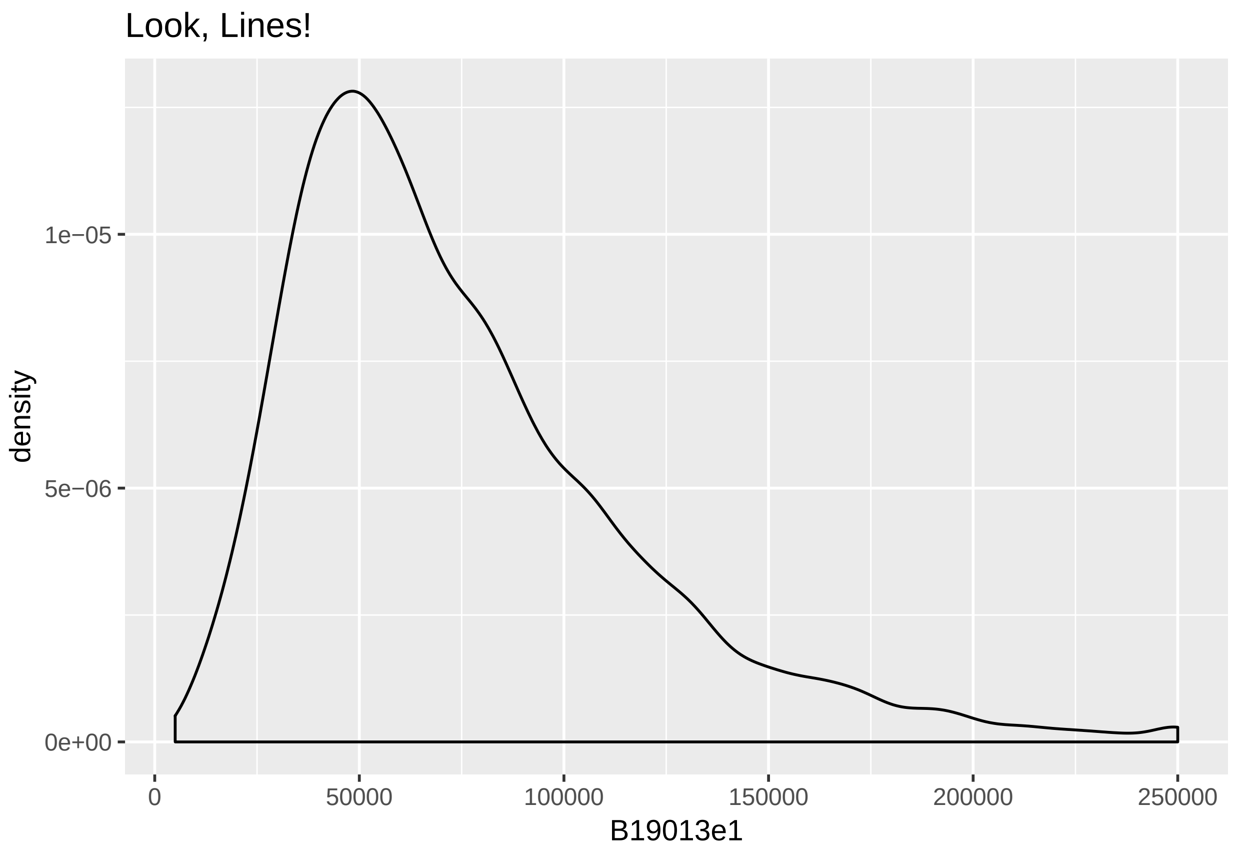
```
## Warning: Removed 108 rows containing non-finite values (stat_bin).
```



Alternatively, you can use `geom_density()`, which makes a smoothed density curve (very close to a continuous histogram). Whether or not smoothing is appropriate depends on your data and goals.

```r
# with geom_density
ggplot(data = block.groups) +
  geom_density(aes(x=B19013e1)) +
  ggtitle("Look, Lines!")
```

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

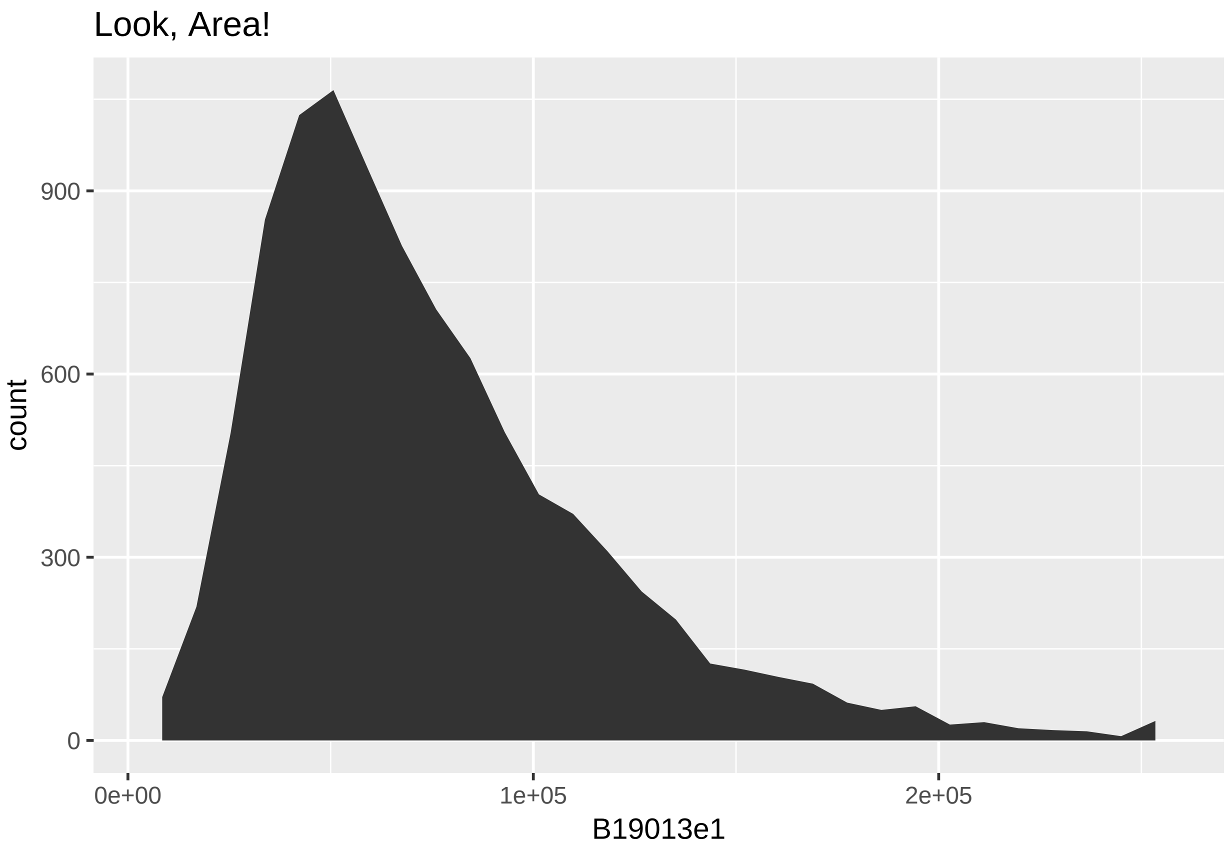


It is sometimes useful to color in the area under the line, which you can do by changing `geom="line"` to `geom="area"`:

```
ggplot(data = block.groups) +
  stat_bin(aes(x=B19013e1), 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).
```



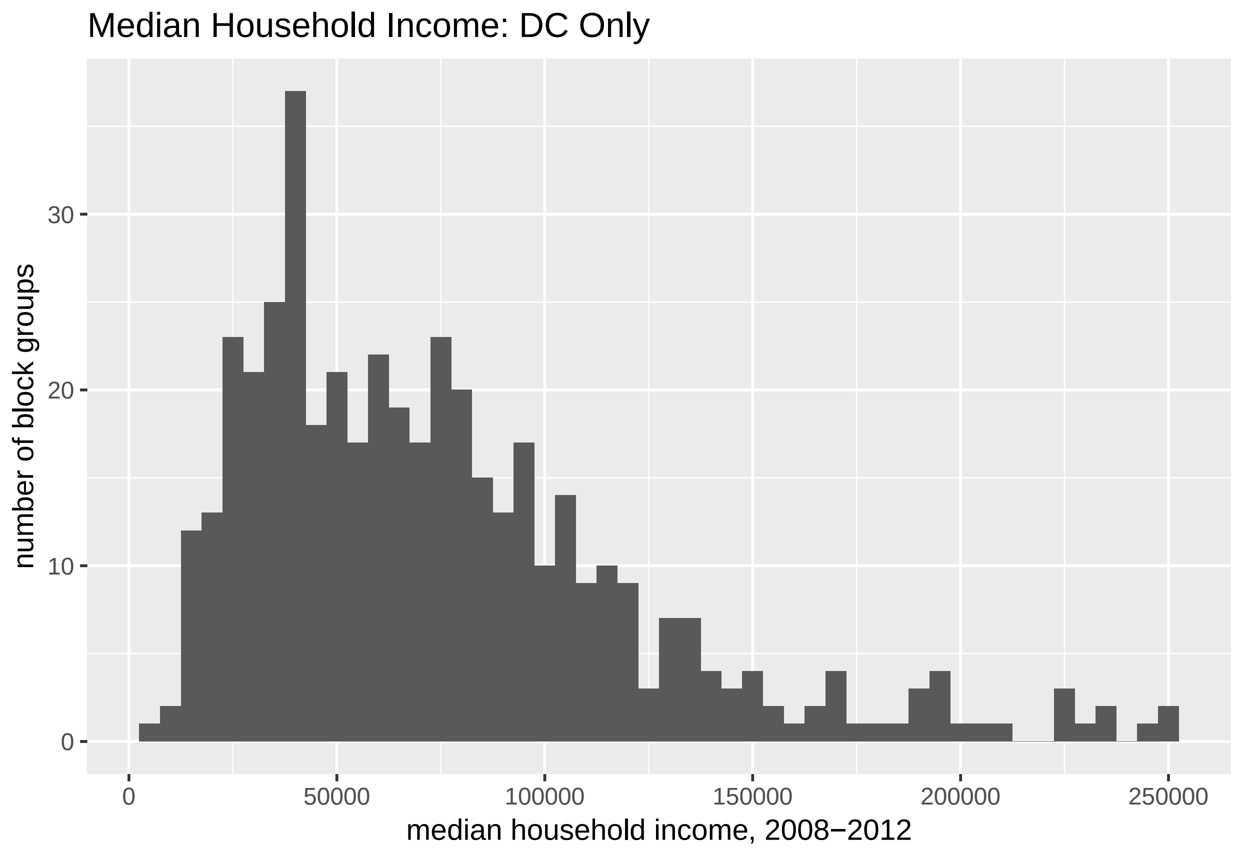
I don't view these as superior to the previous final histogram because to me they seem more difficult to understand without any other benefit.

F.2. Subgroups

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(data = block.groups[which(block.groups$STATE == "11"),]) +
  geom_histogram(aes(x=B19013e1), 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. Use the `tidyverse` command of `filter` to make a new dataframe, then justs plot the new dataframe.)

```
block.groups.dc <- block.groups %>% filter(STATE == 11)
ggplot(data = block_groups_dc) +
  geom_histogram(aes(x=B19013e1), 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.

G. Comparing the three jurisdictions

If you want to look at three groups and have used other programming languages, you might think about using a loop. However, 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, to compare the three jurisdictions, you should not use a loop. Instead, use `ggplot`'s built-in tools to make graphs by group.

The important missing ingredient before we do this is to recall our discussion of 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.

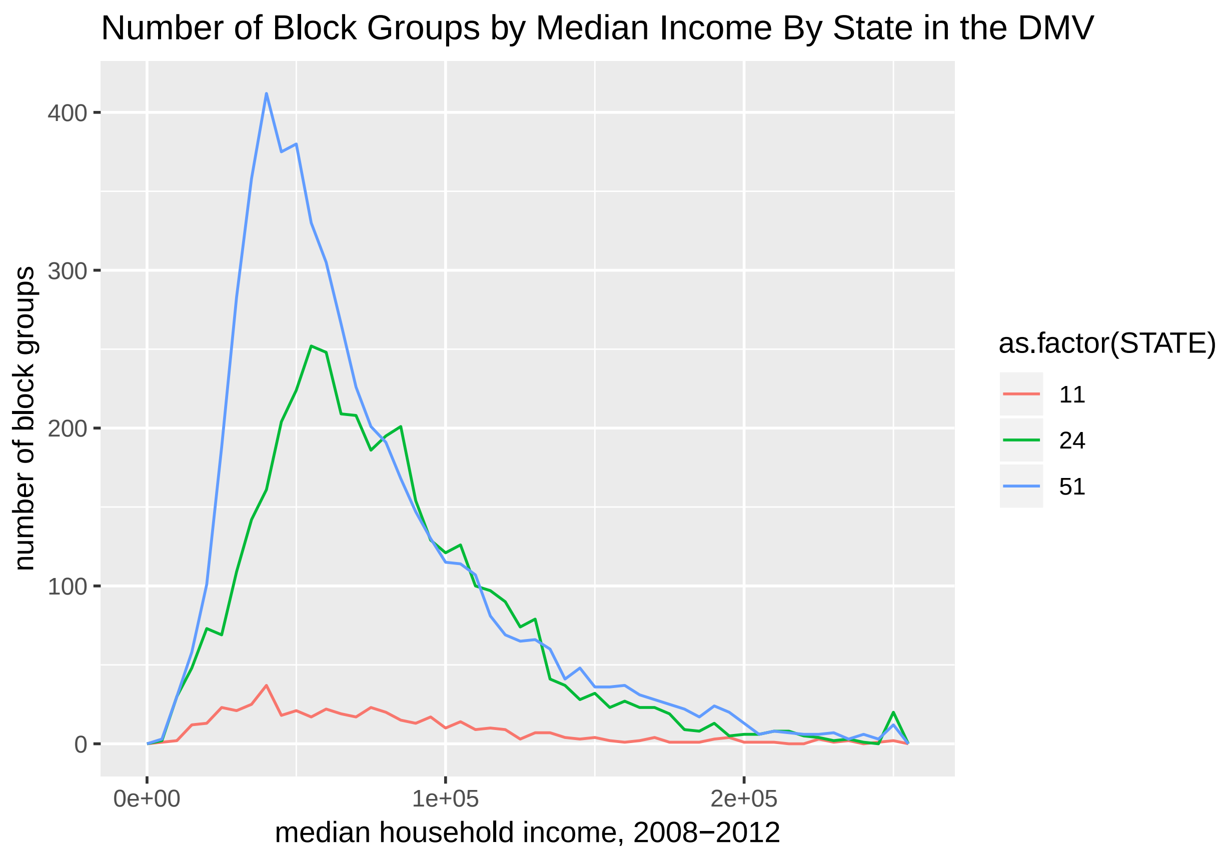
We care about factors here because we can only make graphs by factor variables. If something is not already a factor variable, but it takes on a limited set of values, we can pretend it is by putting `as.factor()` around it.

Here we use yet another potential command for creating histograms: `geom_freqpoly()`. This command also generates a density curve, but does not smoothy like `geom_density()`.

The code below shows the number of block groups by state for median income:

```
ggplot(data = block.groups) +
  geom_freqpoly(aes(x=B19013e1, color = as.factor(STATE)), 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")
```

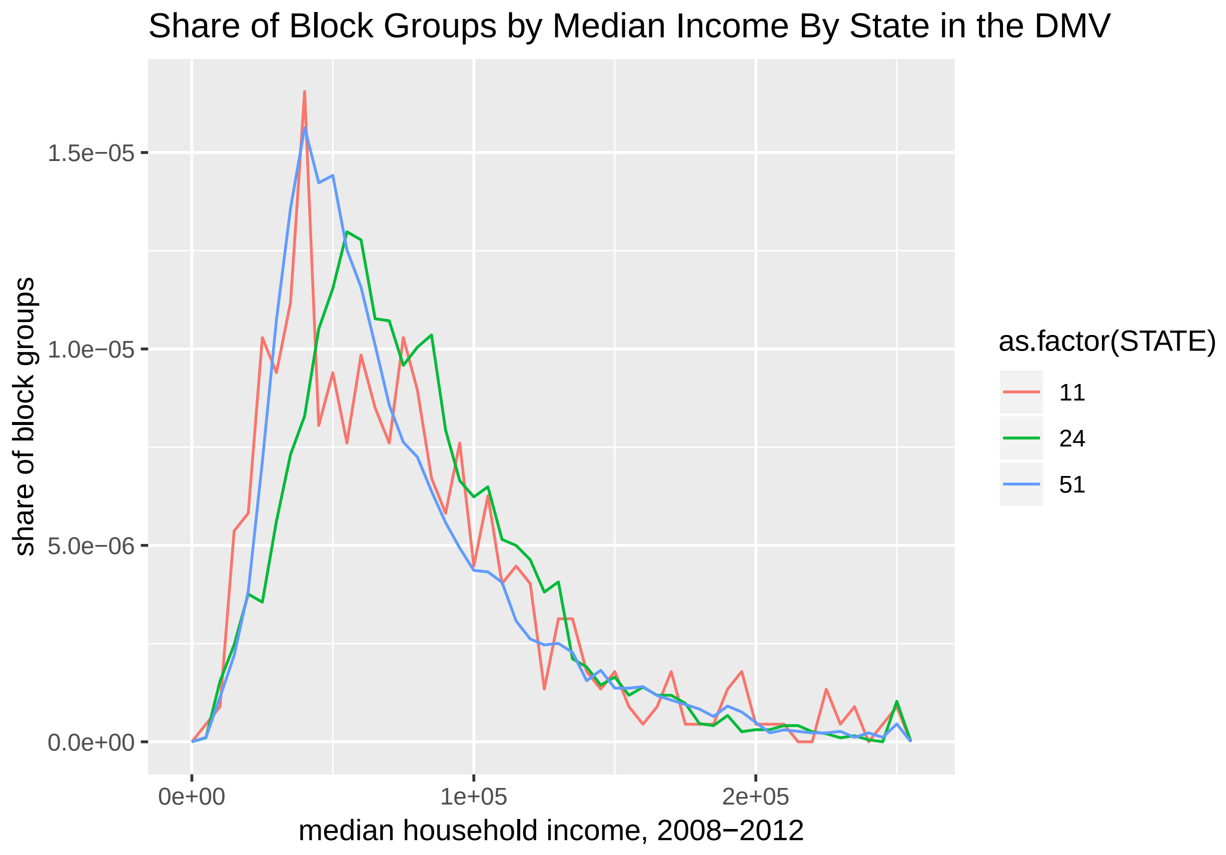
```
## Warning: Removed 108 rows containing non-finite values (stat_bin).
```



One might prefer to show the share of block groups, rather than the number when making this cross-state comparison. To do so, tell R that we want `density`. Using the code we did above calculates the share relative to all observations – not what we want. Why is this density formulation more appropriate for cross-state comparisons?

```r
ggplot(data = block.groups) +
  geom_freqpoly(aes(x=B19013e1, stat(density),
                    color = as.factor(STATE)), 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")
```

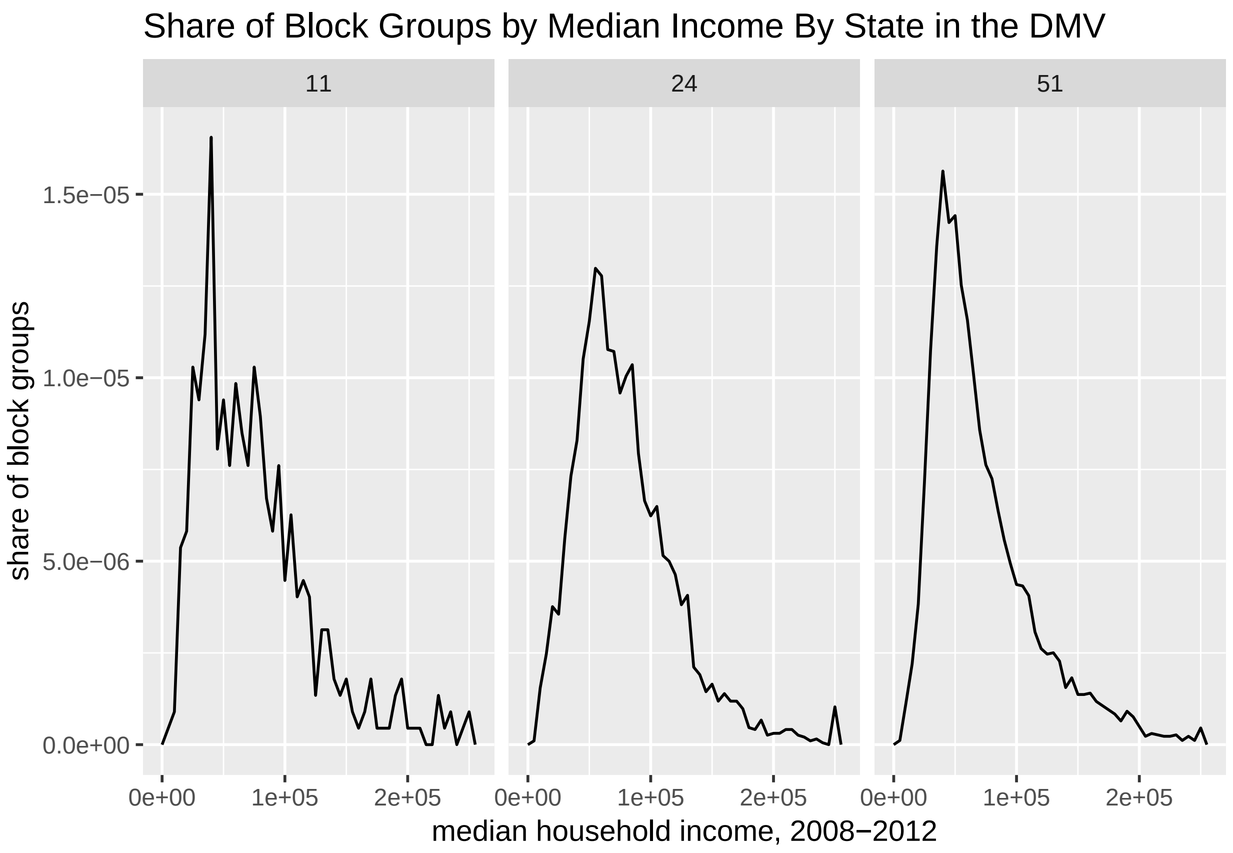
```
## Warning: Removed 108 rows containing non-finite values (stat_bin).
```



Alternatively, you can show multiple small versions of the same chart using `fact_wrap()`:

```r
ggplot(data = block.groups) +
  geom_freqpoly(aes(x=B19013e1, stat(density)), binwidth=5000) +
  facet_wrap(block.groups$STATE) +
  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 these graphs. 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 return to this next class.

H. Homework: Try it yourself

1. Answer the red question(s) in the text.
2. Create a sequence of odd numbers from 1 to 11, inclusive. (Not a graph, just a list of numbers.)
3. Use something from Few's rules about preattentive processing to highlight the larger share of years with large numbers of hurricanes in the 2000s relative to the other centuries.
4. 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)

5. Make two histograms from a dataset (one dataset is sufficient) we have not used in this class. A good place to start if you have no favorite data is Open Data DC here.

   - use whatever you please
   - write 2 to 3 sentences describing what you find