

Lecture 6: Scatter Plots

Leah Brooks

February 27, 2018

Today's agenda

1. Explore a new dataset: DC crime data
2. Data manipulation
 - find things in strings using `substr()`
 - make bins, take averages by bins: `cut()` and `ddply()`
3. Make some scatter plots
 - color dots by type
 - change shape of dots
 - add trend lines

A. Small Data Example with Homicides

We begin by using a new dataset: 2017 crime incidents from Washington, DC.

A.1. Prepare Data

Download these data from [here](#). Take a look at the documentation on this page to get a sense of what these data are.

Save these data somewhere you'll remember and then we will load them in. We'll use the familiar `read.csv()` command and make sure the variable names look ok.

```
crimes.2017 <-  
  read.csv("H:/pppa_data_viz/2018/tutorials/lecture06/dc_crime_2017/Crime_Incidents_in_2017.csv")  
names(crimes.2017)
```

```
## [1] "i..X"          "Y"             "CCN"  
## [4] "REPORT_DAT"    "SHIFT"         "METHOD"  
## [7] "OFFENSE"       "BLOCK"         "XBLOCK"  
## [10] "YBLOCK"        "WARD"          "ANC"  
## [13] "DISTRICT"      "PSA"           "NEIGHBORHOOD_CLUSTER"  
## [16] "BLOCK_GROUP"   "CENSUS_TRACT"  "VOTING_PRECINCT"  
## [19] "LATITUDE"      "LONGITUDE"     "BID"  
## [22] "START_DATE"    "END_DATE"      "OBJECTID"
```

When I loaded the data, one of these names looked fishy. If you look closely, the `x` variable is an `i` with two dots on top, followed by two periods and a `x`. Strange letters like this cause trouble, so let's create a new variable that is equal to this variable but that has a normal name.

(Alternativley, Rosa had a similar problem, but my solution below did not work for her. Instead, she used `names(crimes.2017) <- sub("i..X", "X", names(crimes.2017))`, which finds `i..X` in `names(crimes.2017)` and replaces with `X`.)

```
crimes.2017$X <- crimes.2017$i..X
```

Now let's see what kind of offenses are in this dataframe:

```
table(crimes.2017$OFFENSE)
```

```
##
##              ARSON ASSAULT W/DANGEROUS WEAPON
##              5                               1853
##          BURGLARY                               HOMICIDE
##              1527                               115
##      MOTOR VEHICLE THEFT                       ROBBERY
##              2411                               2167
##          SEX ABUSE                             THEFT F/AUTO
##              290                               10256
##      THEFT/OTHER
##              14451
```

To make the graphing easier to see, we'll just keep homicides. Keeping only a portion of the data is "taking a subset" of the data. In past classes we've used the `subset` command, but I have become nervous about that command from further reading, and instead I'm going to rely on `which` and matrix notation.

In the command below, I name rows where the criminal offense is homicide with `which(crimes.2017$OFFENSE == "HOMICIDE")`, and I select all columns `,]`. I check the dimensions of this new dataframe. It should be as long as the number of homicides we found above in the table. If the dimension does not match, something has gone wrong. Finally, I used `summary()` to take a look at the distribution of all variables in this dataframe.

```
# lets keep only homicides
```

```
homicides <- crimes.2017[which(crimes.2017$OFFENSE == "HOMICIDE"),]
dim(homicides)
```

```
## [1] 115  25
```

```
summary(homicides)
```

```
##      i..X      Y      CCN
##  Min.   :-77.07  Min.   :38.82  Min.    : 9251904
## 1st Qu. :-77.01  1st Qu.:38.85  1st Qu.:17046654
## Median :-76.99  Median :38.89  Median :17116708
## Mean   :-76.98  Mean   :38.89  Mean   :17005983
## 3rd Qu. :-76.97  3rd Qu.:38.91  3rd Qu.:17172014
## Max.    :-76.91  Max.    :38.98  Max.    :17222986
##
##      REPORT_DAT  SHIFT  METHOD
## 2017-07-30T00:00:00.000Z: 4 DAY      : 0 GUN      :87
## 2017-01-17T00:00:00.000Z: 2 EVENING : 0 KNIFE    : 9
## 2017-03-11T00:00:00.000Z: 2 MIDNIGHT:115 OTHERS:19
## 2017-03-17T00:00:00.000Z: 2
## 2017-05-10T00:00:00.000Z: 2
## 2017-06-26T00:00:00.000Z: 2
## (Other)              :101
##      OFFENSE
## HOMICIDE              :115
## ARSON                  : 0
## ASSAULT W/DANGEROUS WEAPON: 0
## BURGLARY                : 0
## MOTOR VEHICLE THEFT      : 0
```

```

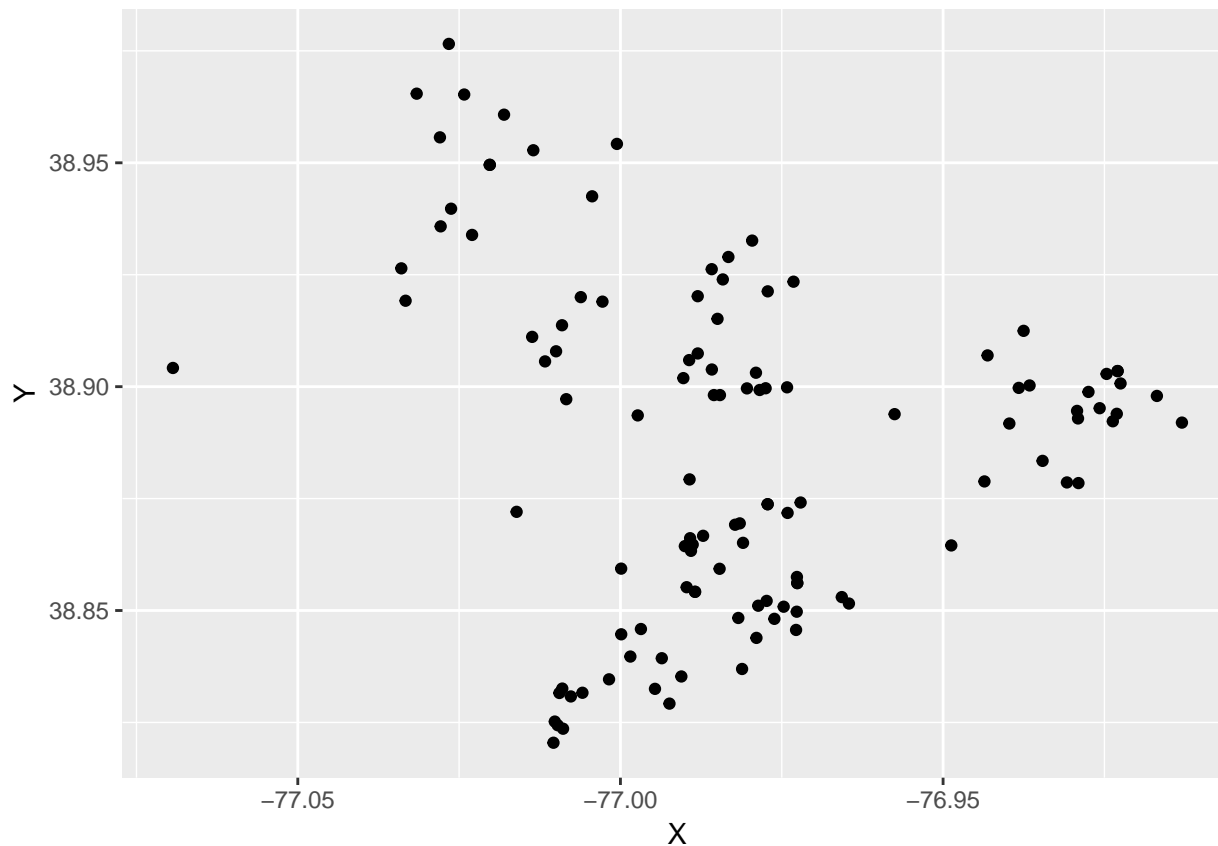
## ROBBERY : 0
## (Other) : 0
##
## BLOCK XBLOCK
## 1903 - 2099 BLOCK OF FAIRLAWN AVENUE SE: 2 Min. :393985
## 2400 - 2599 BLOCK OF ELVANS ROAD SE : 2 1st Qu.:399555
## 2700 2899 BLOCK OF LANGSTON PLACE SE : 2 Median :401257
## 4820 - 4899 BLOCK OF KANSAS AVENUE NW : 2 Mean :401526
## 934 - 1099 BLOCK OF EASTERN AVENUE NE : 2 3rd Qu.:402373
## 1 - 99 BLOCK OF HANOVER PLACE NW : 1 Max. :407549
## (Other) :104
## YBLOCK WARD ANC DISTRICT
## Min. :128076 Min. :1.000 7C :13 Min. :1.000
## 1st Qu.:131875 1st Qu.:5.000 8A :12 1st Qu.:5.000
## Median :136119 Median :7.000 8B :10 Median :6.000
## Mean :135403 Mean :6.461 8E : 9 Mean :5.513
## 3rd Qu.:137755 3rd Qu.:8.000 8D : 8 3rd Qu.:7.000
## Max. :145403 Max. :8.000 5D : 7 Max. :7.000
## (Other):56
## PSA NEIGHBORHOOD_CLUSTER BLOCK_GROUP CENSUS_TRACT
## Min. :103.0 Cluster 39:16 007503 1: 5 Min. : 202
## 1st Qu.:502.5 Cluster 31:13 007601 1: 4 1st Qu.: 7406
## Median :605.0 Cluster 23: 9 007403 1: 3 Median : 7806
## Mean :556.0 Cluster 38: 9 007406 2: 3 Mean : 7643
## 3rd Qu.:702.5 Cluster 22: 7 007502 1: 3 3rd Qu.: 9301
## Max. :708.0 Cluster 28: 7 007807 2: 3 Max. :11100
## (Other) :54 (Other) :94 NA's :1
## VOTING_PRECINCT LATITUDE LONGITUDE BID
## Precinct 114: 7 Min. :38.82 Min. : -77.07 :111
## Precinct 116: 5 1st Qu.:38.85 1st Qu.: -77.01 ANACOSTIA : 2
## Precinct 124: 5 Median :38.89 Median : -76.99 DOWNTOWN : 1
## Precinct 115: 4 Mean :38.89 Mean : -76.98 GEORGETOWN : 1
## Precinct 126: 4 3rd Qu.:38.91 3rd Qu.: -76.97 ADAMS MORGAN: 0
## Precinct 72 : 4 Max. :38.98 Max. : -76.91 CAPITOL HILL: 0
## (Other) :86 (Other) : 0
## START_DATE END_DATE
## 2017-07-30T01:59:18.000Z: 2 : 8
## 2009-02-13T12:03:45.000Z: 1 2017-07-30T02:05:03.000Z: 2
## 2014-05-01T19:32:00.000Z: 1 2009-02-13T12:03:43.000Z: 1
## 2016-10-02T02:33:43.000Z: 1 2014-05-01T19:32:00.000Z: 1
## 2016-12-25T11:59:35.000Z: 1 2016-10-02T03:03:13.000Z: 1
## 2017-01-07T16:13:07.000Z: 1 2016-12-25T20:36:17.000Z: 1
## (Other) :108 (Other) :101
## OBJECTID X
## Min. :125524302 Min. : -77.07
## 1st Qu.:125729760 1st Qu.: -77.01
## Median :125739163 Median : -76.99
## Mean :125765301 Mean : -76.98
## 3rd Qu.:125829406 3rd Qu.: -76.97
## Max. :125867304 Max. : -76.91
##

```

A.2. Initial Plots

While it's not standard, you can make a scatterplot with geographic coordinates, and that's what we're going to do with this dataset (in future classes, we'll use more sophisticated map techniques). We begin with a simple point graph that maps the location of all homicides in 2017, using `geom_point()`.

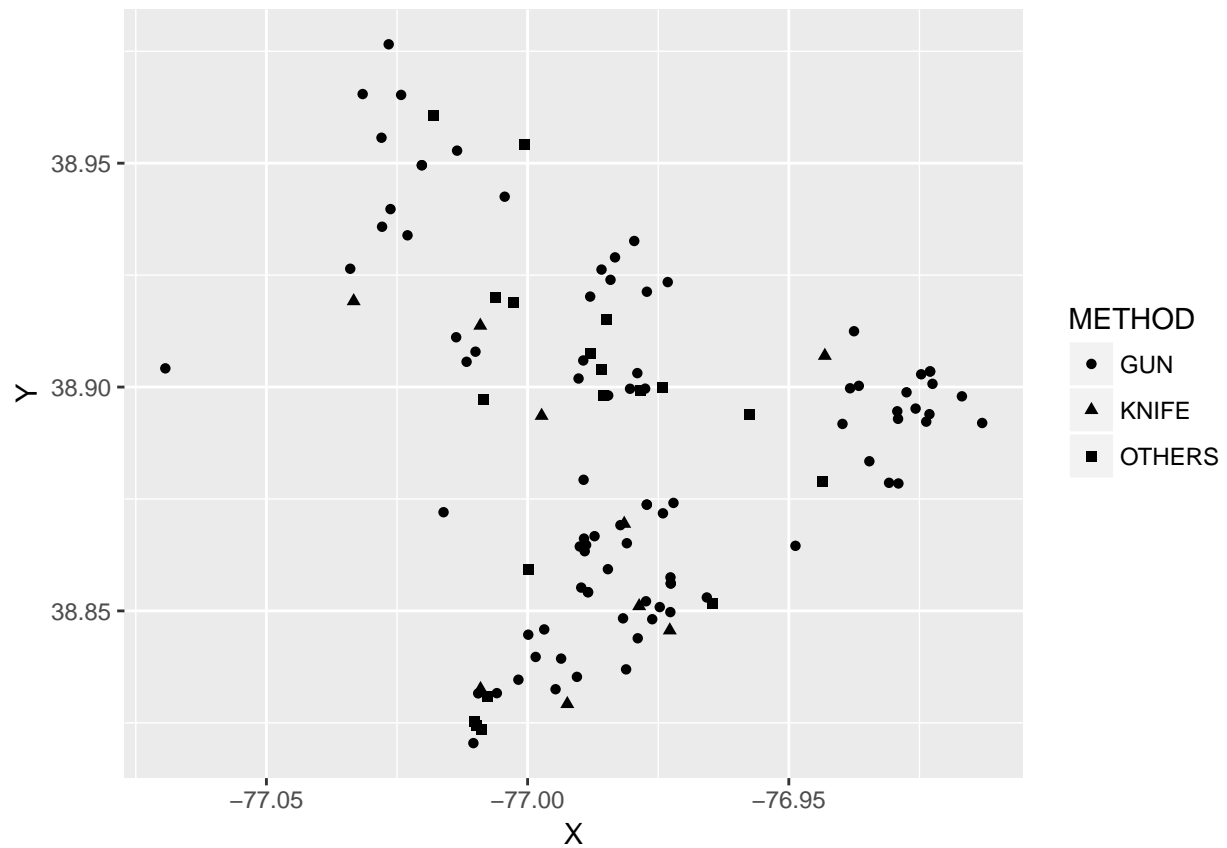
```
# lets make a plot where we show lat and long of homicides
# pay attention to capitalization of variables
library(ggplot2)
ggplot(homicides, aes(x = X, y = Y)) +
  geom_point()
```



This chart is ok. You know where the homicides are in space (though it would be helpful to have a DC map around the points).

We can use the power of R to distinguish homicides by method. To do this, we tell R that the data are in groups by METHOD (see the `group=METHOD` in the `ggplot` call), and let R know it should distinguish these groups with shapes in the point layer (`geom_point(aes(shape=METHOD))`).

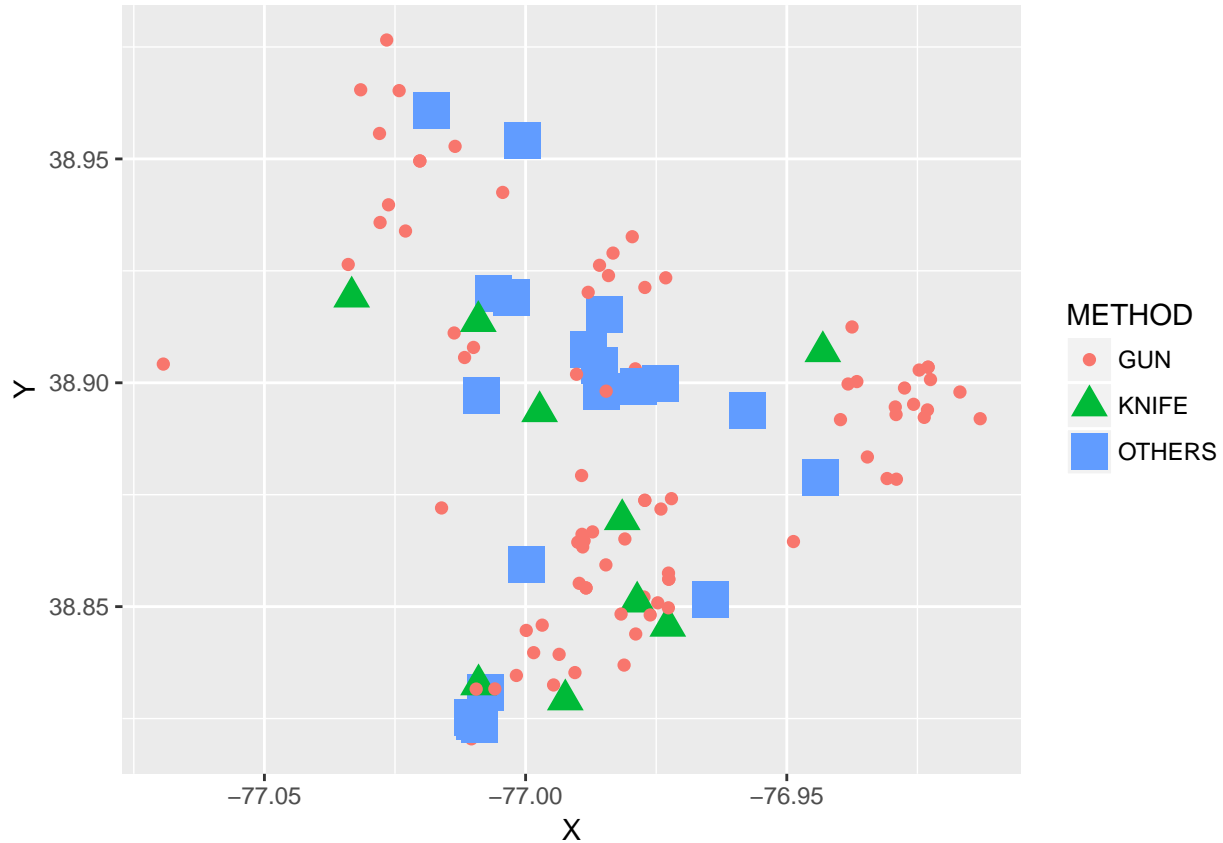
```
# lets distinguish offenses by method
ggplot(homicides, aes(x=X, y=Y, group = METHOD)) +
  geom_point(aes(shape=METHOD))
```



Not because I recommend this, but as a proof of concept, you can actually differentiate between points in at least three dimensions: shape, color and size. See below.

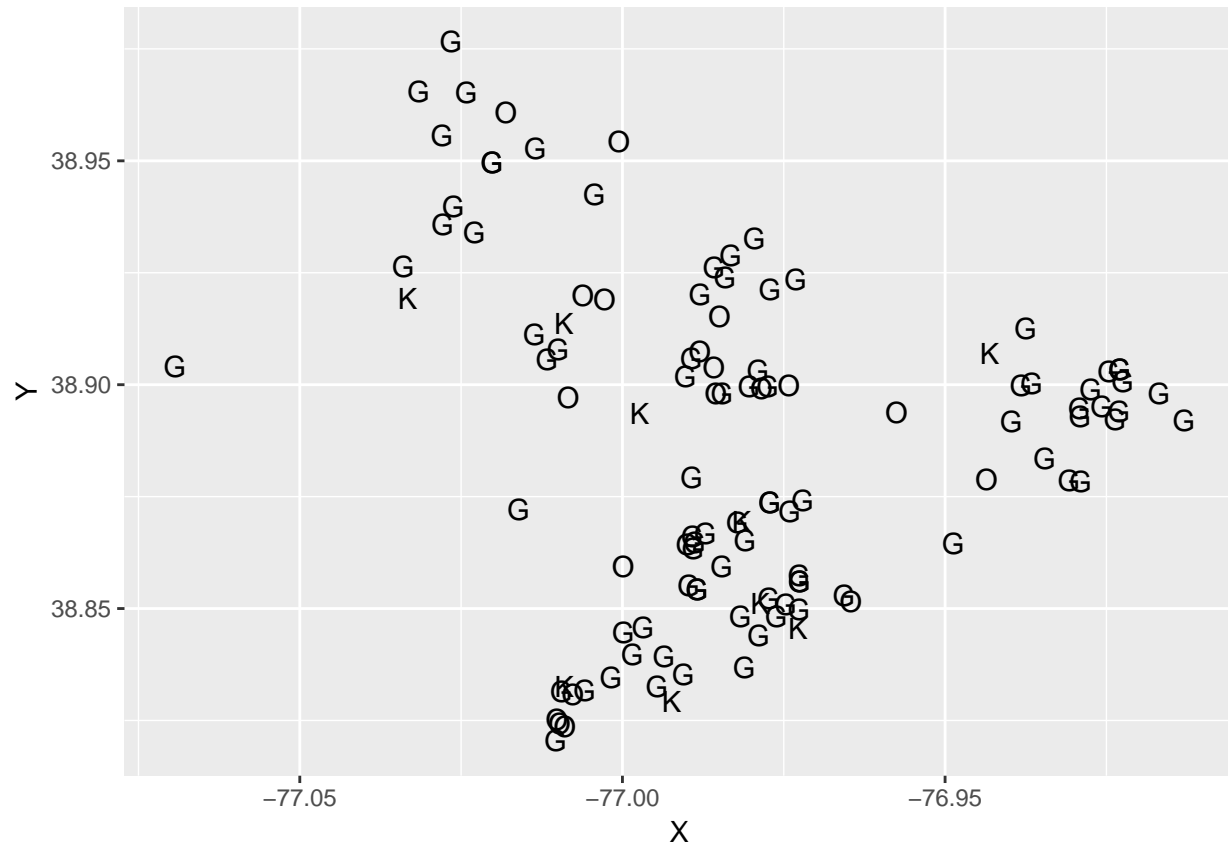
```
# you can use even three types of differentiation
ggplot(homicides, aes(x=X, y=Y, group = METHOD)) +
  geom_point(aes(shape=METHOD, color = METHOD, size = METHOD))
```

```
## Warning: Using size for a discrete variable is not advised.
```



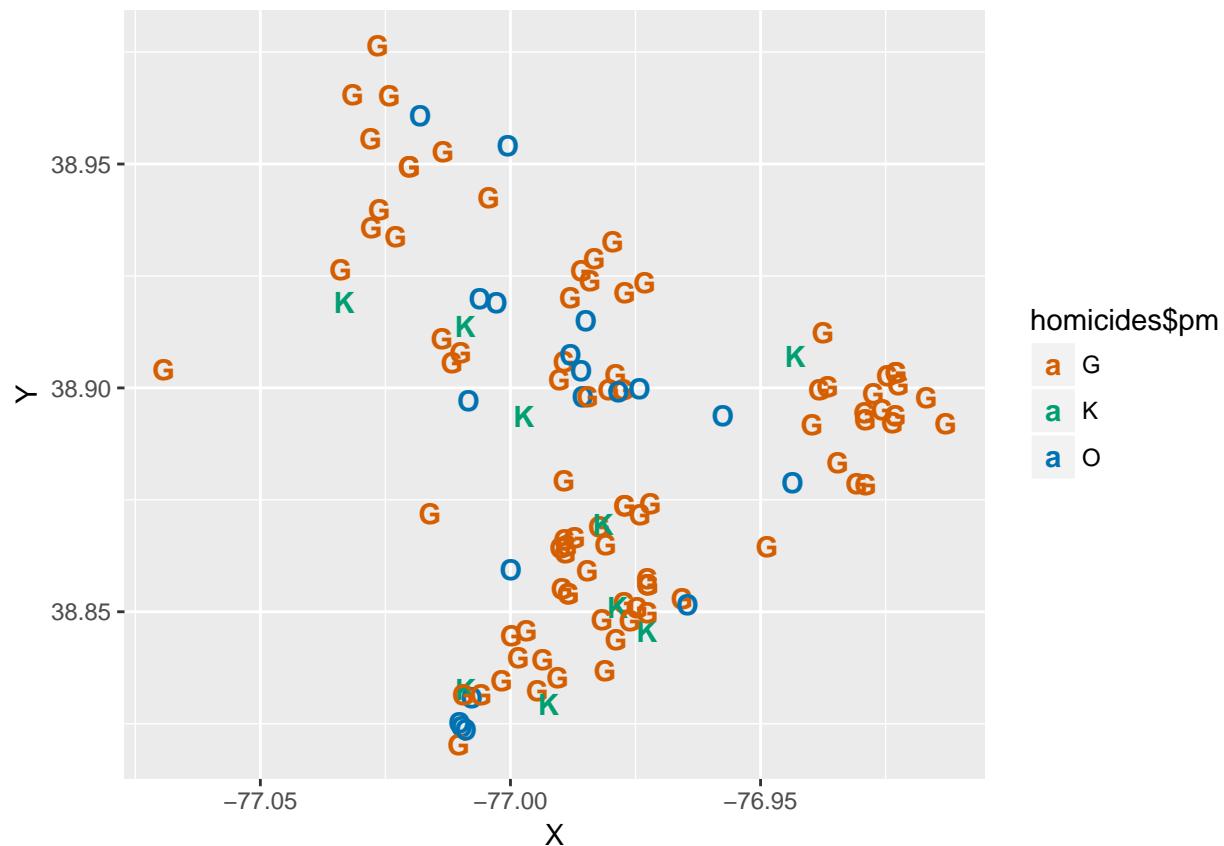
For conveying information, we're probably better off with a symbol that conveys a little more meaning. I use the `ifelse()` command to create a new variable called `pm` that is the letter we'd like to plot. I then change the plot type from `geom_point()` to `geom_text()` and plot these letters.

```
# but maybe lets try something meaningful
homicides$pm <- ifelse(homicides$METHOD == "GUN", "G",
                      ifelse(homicides$METHOD == "KNIFE", "K",
                             ifelse(homicides$METHOD == "OTHERS", "O", "NA")))
ggplot(homicides, aes(x=X, y=Y, group = METHOD)) +
  geom_text(label=homicides$pm)
```



This could stand for a few improvements. The letters are quite hard to distinguish, and I'm going to modify this by changing the color and making the font bold. I found very helpful color suggestions here.


```
ggplot(homicides, aes(x=X, y=Y, group = METHOD)) +
  geom_text(aes(label=homicides$pm, color = homicides$pm), fontface = "bold") +
  scale_color_manual(values = c("#D55E00", "#009E73", "#0072B2"))
```



A.3. Looking at data by ward

All charts to date have used the individual level data. If we want to make comparisons across wards, it is helpful (virtually necessary) to take averages by ward. We are going to * count the number of homicides * find the average latitude and longitude * find the average time of day of murders

We begin by printing to the screen the first five rows of the table, writing `homicides[1:5,]`. You can see that the hour is always at the same location in the text string.

The next step is manipulating the data to access the date stored in the `START_DATE` variable (look below the output from the dataset). At the moment, `START_DATE` is a big long mess of text and numbers. The next step is two nested commands. You'll see

- the interior command grabs the hour using the `substr()` command. This command goes to the 12th character in `homicides$START_DATE` and keeps everything until the 13th character.
- look at your printed values of `homicides$START_DATE` to see why I chose those particular locations. The command is generalizable to any variable and start and stop location.
- the exterior command (`as.numeric()`) takes that string output and makes it into a number

Finally, I look at this output to see if it looks reasonable using the `table()` command. It does: the top row of the table lists the hours we see in the dataset. The bottom row reports the number of homicides in each hour. Most homicides occur between 8 pm and midnight.

```
# find average hour of the day by ward
# use substring function to get time out of start_date variable
homicides[1:5,]
```

```
##          i..X          Y          CCN          REPORT_DAT    SHIFT METHOD
## 25  -77.00972 38.82439 16217337 2017-04-03T00:00:00.000Z MIDNIGHT OTHERS
## 53  -76.92296 38.90348 17222616 2017-12-28T00:00:00.000Z MIDNIGHT   GUN
## 68  -76.94359 38.87883 17222848 2017-12-28T00:00:00.000Z MIDNIGHT OTHERS
## 198 -76.97318 38.92344 17221838 2017-12-27T00:00:00.000Z MIDNIGHT   GUN
## 209 -76.97717 38.92129 17222010 2017-12-27T00:00:00.000Z MIDNIGHT   GUN
##      OFFENSE                                BLOCK XBLOCK YBLOCK WARD ANC
## 25  HOMICIDE      2 - 153 BLOCK OF GALVESTON PLACE SW 399156 128511    8  8D
## 53  HOMICIDE    934 - 1099 BLOCK OF EASTERN AVENUE NE 406683 137294    7  7C
## 68  HOMICIDE     810 - 845 BLOCK OF ADRIAN STREET SE 404895 134556    7  7E
## 198 HOMICIDE   2200 - 2399 BLOCK OF DOUGLAS STREET NE 402326 139507    5  5C
## 209 HOMICIDE   1815 - 1999 BLOCK OF BRYANT STREET NE 401980 139268    5  5C
##      DISTRICT PSA NEIGHBORHOOD_CLUSTER BLOCK_GROUP CENSUS_TRACT
## 25          7 708                Cluster 39    009807 2          9807
## 53          6 608                Cluster 31    007807 2          7807
## 68          6 605                Cluster 33    007707 2          7707
## 198         5 503                Cluster 22    011100 3          11100
## 209         5 505                Cluster 22    011100 3          11100
##      VOTING_PRECINCT LATITUDE LONGITUDE BID          START_DATE
## 25      Precinct 126 38.82438 -77.00972    2016-12-25T11:59:35.000Z
## 53      Precinct 95 38.90348 -76.92295    2017-12-28T11:20:09.000Z
## 68      Precinct 106 38.87882 -76.94359    2017-12-28T19:18:51.000Z
## 198     Precinct 72 38.92343 -76.97318    2017-12-26T17:15:33.000Z
## 209     Precinct 72 38.92128 -76.97717    2017-12-27T02:52:01.000Z
##      END_DATE OBJECTID          X pm
## 25 2016-12-25T20:36:17.000Z 125525367 -77.00972  0
## 53 2017-12-28T11:23:03.000Z 125524302 -76.92296  G
## 68 2017-12-28T19:25:57.000Z 125524317 -76.94359  0
## 198 2017-12-26T19:54:42.000Z 125524699 -76.97318  G
## 209 2017-12-27T02:54:22.000Z 125524738 -76.97717  G
```

```
homicides$hour <- as.numeric(substr(homicides$START_DATE,12,13))
table(homicides$hour)
```

```
##
##  0  1  2  3  4  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
##  6 10  6  3  4  1  3  6  1  4  4  4  2  4  3  1  4  3  8  8 12  8 10
```

Now that the data are prepared, we can take averages by ward using the `ddply` command we've used before. See earlier tutorials for syntax.

I then round the hour of the day so it will look ok in pictures using the `round()` command. This command rounds the variable you choose to the number of significant digits you set; here I choose one, which means that it will generate 12.3 instead of 12.277568, for example.

```
# lets use transpose and take average by ward
# im also taking average of x and y
library(plyr)
ward.hom <- ddply(homicides,
                  "WARD",
                  summarize,
                  mean.start.hour = mean(hour),
```

```

av.x = mean(X),
av.y = mean(Y),
num.hom = sum(METHOD != "")
ward.hom$mean.start.hour <- round(ward.hom$mean.start.hour,digits = 1)

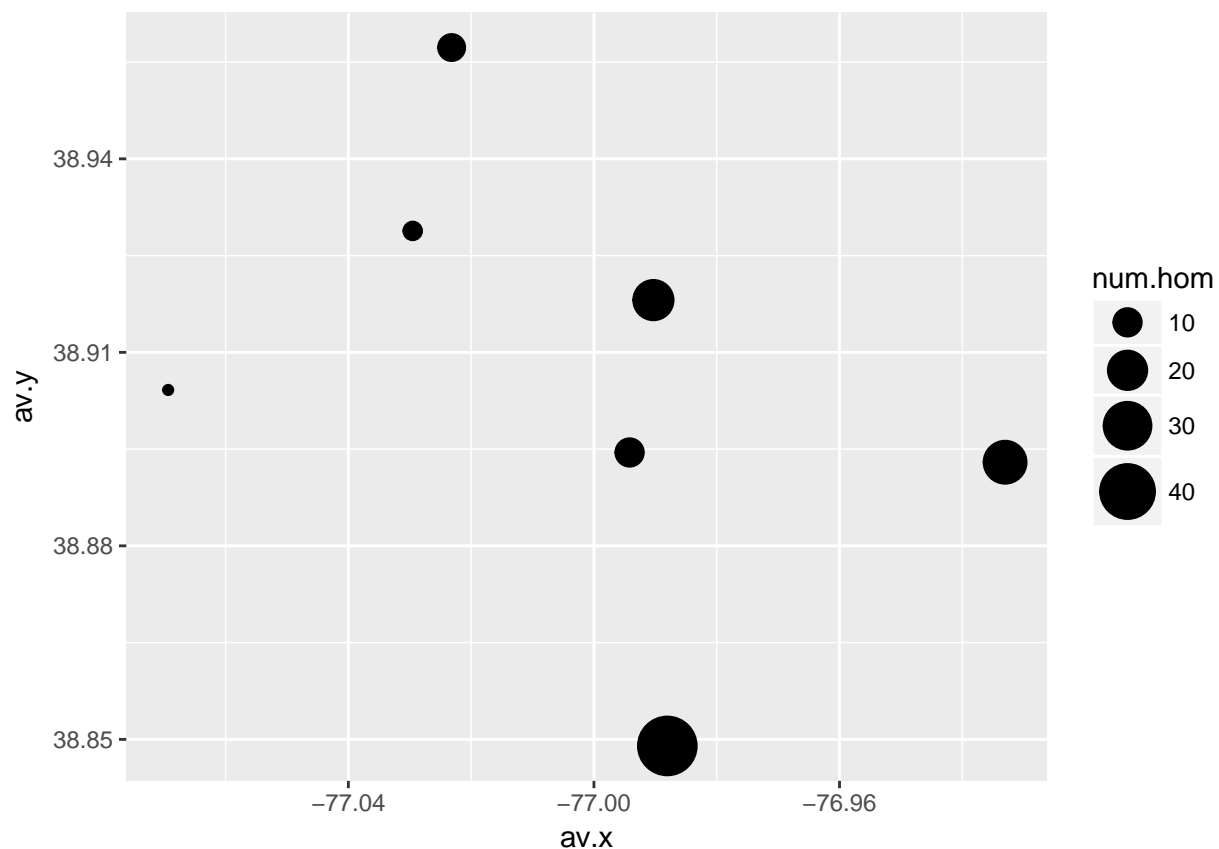
```

Given this data preparation, we are ready to make charts. Below we make another scatter plot by location, where this is now the average latitude and longitude of homicides by ward. I size the point of the scatter by the number of homicides using the `size=num.hom` option inside the `aes()` command. The relative size of the points is determined by the relative number of homicides, but we can scale all the points by using the `scale_size_area()` command. Here I set `max_size = 10` – you should play around with variations on this.

```

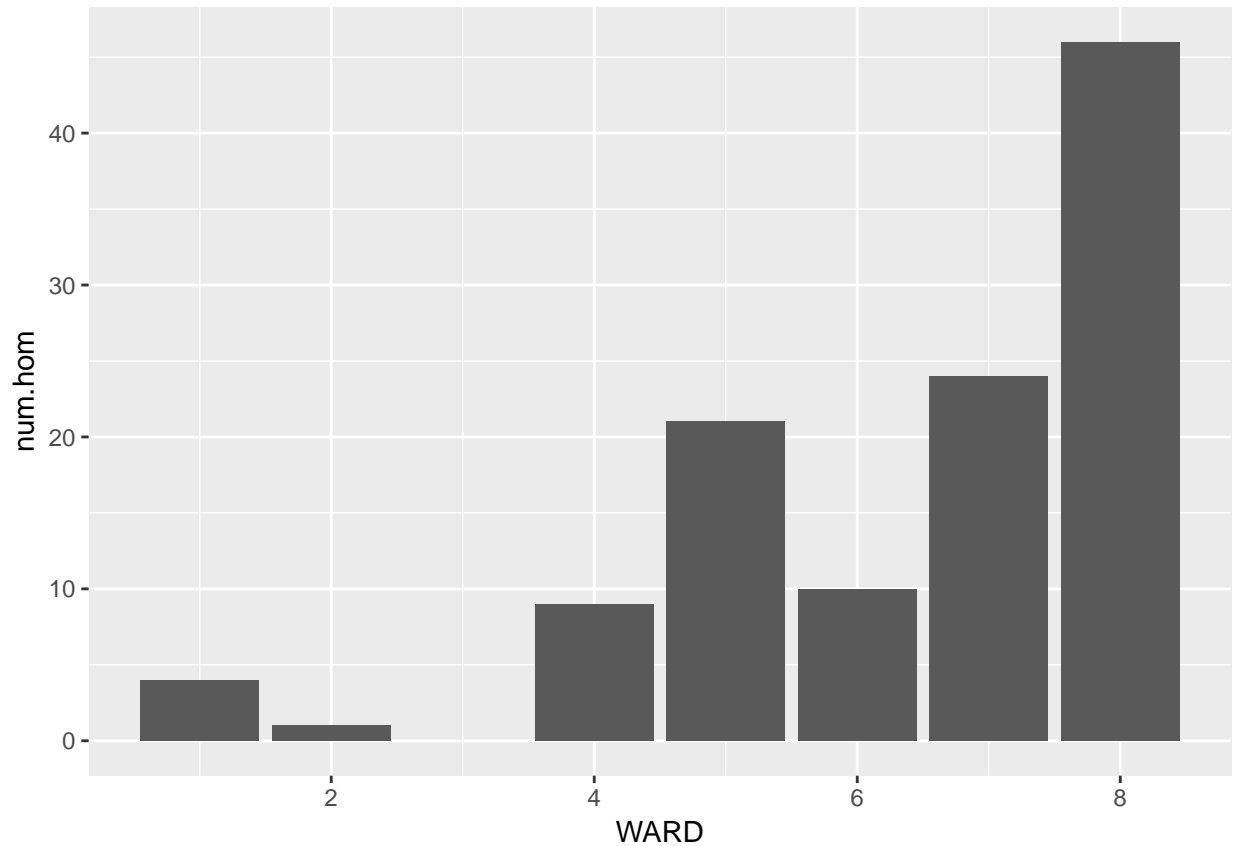
# now make a chart with bubbles by the number of homicides
# and text with the hour of the day
# at location (x,y)
# am pretty sure this suffers from the 2-d size problem
ggplot(ward.hom, aes(x=av.x, y=av.y, size = num.hom)) +
  geom_point() +
  scale_size_area(max_size = 10)

```



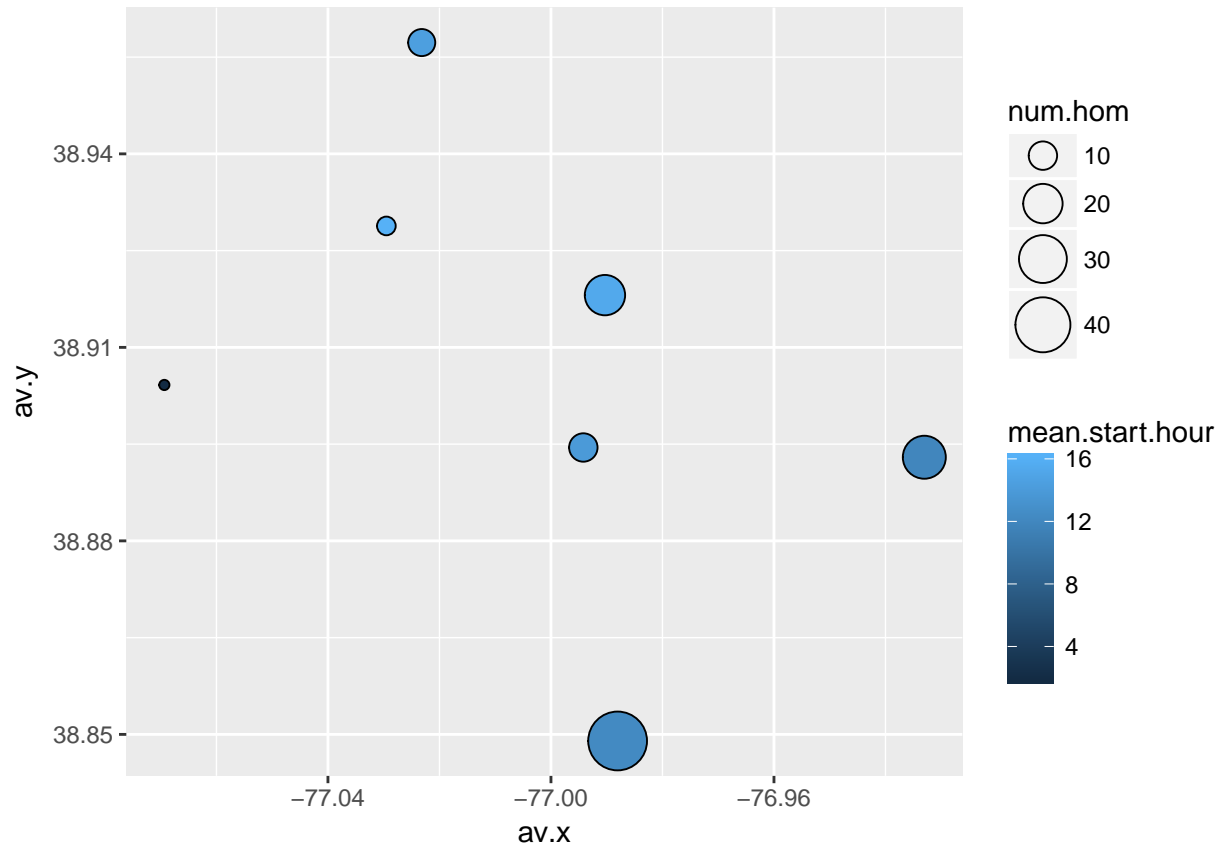
After some reflection, I think this bubble plot is not a good way to make comparisons. Fundamentally, it's going against Few's very reasonable rule about not using 2-D surfaces to represent quantity. Compare what you learn from the picture with the bar chart below, and I think you'll see that the size is misleading.

```
# compare with this one  
ggplot(ward.hom, aes(x=WARD, y=num.hom)) +  
  geom_bar(stat="identity")
```



Just to round things out, we'll color the points by the average hour of the day that the homicides took place. I add the `fill()` option to the overall `aes()` command, and change the shape to be shape number 21, which is a circle filled in with a color. You can see all the R point markers by number ([here](http://sape.inf.usi.ch/quick-reference/ggplot2/shape))[\[http://sape.inf.usi.ch/quick-reference/ggplot2/shape\]](http://sape.inf.usi.ch/quick-reference/ggplot2/shape). I don't care much for the chart below – just including for completeness.

```
# color of circles by hour of day
# see shapes here
ggplot(ward.hom, aes(x=av.x, y=av.y, size = num.hom, fill=mean.start.hour)) +
  geom_point(shape=21) +
  scale_size_area(max_size = 10)
```



B. More Scatters With Block Group Data

Now we'll try a few more scatter plots with the block group data. Please refer to the tutorial from Lecture 2 for complete details about these data. Some of you were confused about the meaning of the variables in this dataset; the tutorial links to variable coverage and definition information.

B.1. Set Up Data

We begin by setting up data: loading and creating shares. This first block of code should be familiar.

```
# bring in block group data
block.groups <- read.csv(
  "h:/pppa_data_viz/2018/tutorials/lecture02/acs_bgs20082012_dmv_20180123.csv")

## create some variables of interest

# share in this location one year ago
block.groups$shr.here.1yr.ago <- block.groups$B07201e2/block.groups$B07201e1
```

In a previous lecture, one of you asked how to sum a list of similarly named variables. We will now use code to do this to create shares of education by block group. We want to find the share of people who have dropped out of high school or otherwise have less than a high school education. This requires summing B15002e3 to B15002e10 and B15002e20 through B15002e27. We divide these sums by the total number of people for whom information about education is available (B15002e1).

To do this, we will rely on the `apply` command, which is a very powerful and confusing command in R. I encourage you to understand how it works in the current example. As we work on more tutorials, we will likely see it again.

The general syntax for `apply` is `(data.frame, 1 for rows or 2 for columns, function to apply)`. Here we create a new variable in the block groups data frame, `block.groups$t1`, using only a subset of columns (if the subset command with `[]` seems confusing, see the discussion earlier in this tutorial). Here we are selecting a set of columns (with `c`, as usual) using a command that finds the column number: `match(variable name, in set of variable names)`. We find the number of the first and last columns of interest and use a colon to say all columns from the first to the last. The `margin = 1` (you can also just write `1`) means apply on rows, and the final `sum` means take the sum of the named columns.

After this, I take the sum of men and women with low education and divide by the total number of people for whom education is defined.

I check the result using `summary`. Is it a share? Is it reasonable?

Finally, I get rid of the totals I created that I don't need. I do this by making a subset of `block.groups` selecting all columns where the name is not `t1` or `t2` (`!(names(block.groups) %in% c("t1","t2"))`).

```
# share high school drop out
# with thanks to
# https://stackoverflow.com/questions/29006056/efficiently-sum-across-multiple-columns-in-r

block.groups$t1 <- apply(block.groups[, c(match("B15002e3",names(block.groups)):
                                          match("B15002e10",names(block.groups)))],
                        1, sum)

block.groups$t2 <- apply(block.groups[, c(match("B15002e20",names(block.groups)):
                                          match("B15002e27",names(block.groups)))],
                        1, sum)

# find share
block.groups$shr.ed.lt.hs <- ( (block.groups$t1 + block.groups$t2) / block.groups$B15002e1)
# does it look ok?
summary(block.groups$shr.ed.lt.hs)

##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.     NA's
## 0.00000 0.04381 0.10581 0.13210 0.19200 0.90678      62

# get rid of totals
block.groups <- block.groups[, !(names(block.groups) %in% c("t1","t2"))]
```

Now repeat where education is the share with some college or more.

```
# share some college or more
block.groups$t3 <- apply(block.groups[, c(match("B15002e12",names(block.groups)):
                                          match("B15002e18",names(block.groups)))],
                        1, sum, na.rm = TRUE)
block.groups$t4 <- apply(block.groups[, c(match("B15002e29",names(block.groups)):
                                          match("B15002e35",names(block.groups)))],
                        1, sum, na.rm = TRUE)
block.groups$shr.ed.ge.sc <- ( (block.groups$t3 + block.groups$t4) / block.groups$B15002e1)
summary(block.groups$shr.ed.ge.sc)
```

```
##      Min. 1st Qu.  Median      Mean 3rd Qu.      Max.      NA's
## 0.0000 0.4505 0.6070 0.6073 0.7739 1.0000      62
```

```
# get rid of totals
```

```
block.groups <- block.groups[, !(names(block.groups) %in% c("t3", "t4"))]
```

```
# lets check the final category to make sure it adds up to 1
```

```
block.groups$shr.ed.hs <- (block.groups$B15002e11 + block.groups$B15002e28) / block.groups$B15002e1
summary(block.groups$shr.ed.hs)
```

```
##      Min. 1st Qu.  Median      Mean 3rd Qu.      Max.      NA's
## 0.0000 0.1554 0.2643 0.2606 0.3585 1.0000      62
```

Finally, I pull out median household income.

```
# lets grab income
```

```
block.groups$med.hh.inc <- block.groups$B19013e1
summary(block.groups$med.hh.inc)
```

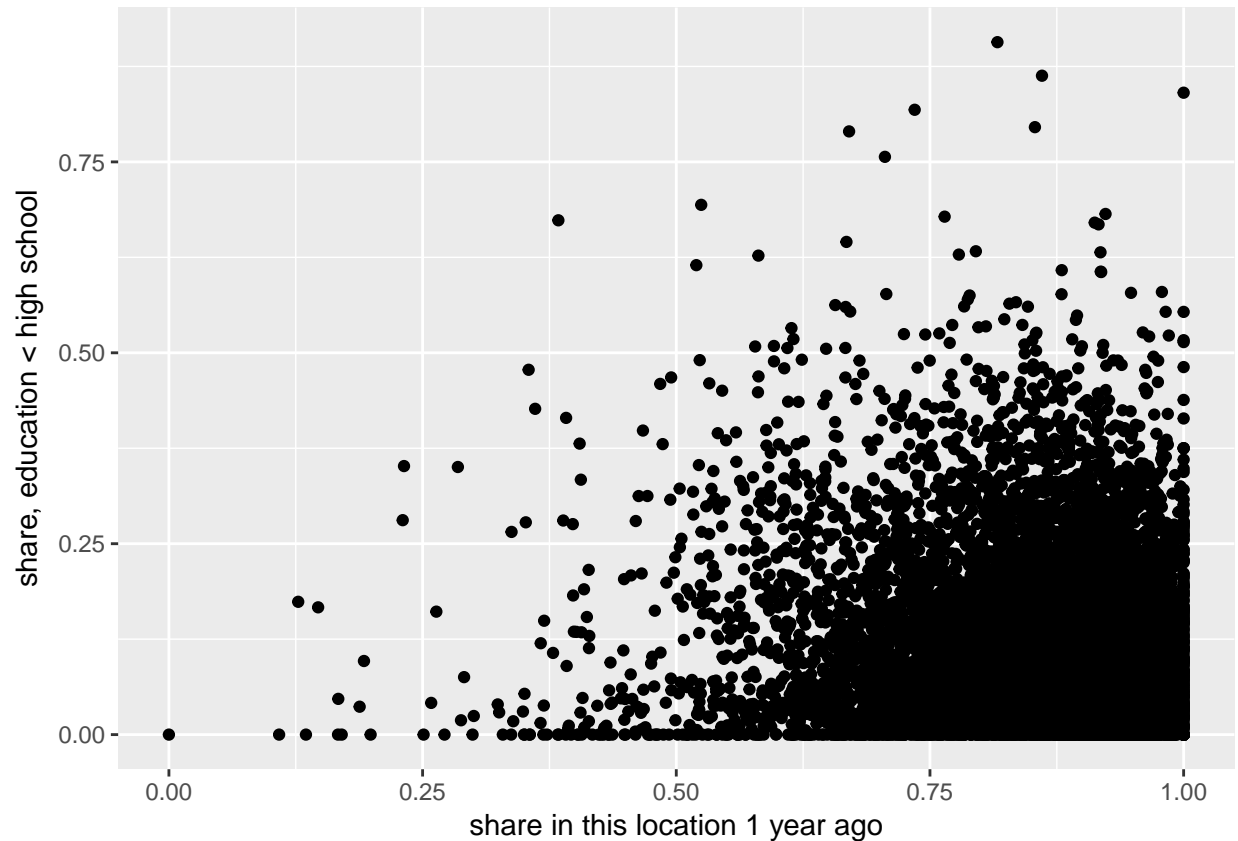
```
##      Min. 1st Qu.  Median      Mean 3rd Qu.      Max.      NA's
## 5000 44217 64511 74523 95246 250001      108
```

B.2. Education and probability of moving

Now we're ready to look at these educational shares and the probability of moving. We begin with the simplest possible scatter.

```
# relationship between probability of moving and share low education  
ggplot(block.groups, aes(x=shr.here.1yr.ago, y=shr.ed.lt.hs)) +  
  labs(x = "share in this location 1 year ago", y = "share, education < high school") +  
  geom_point()
```

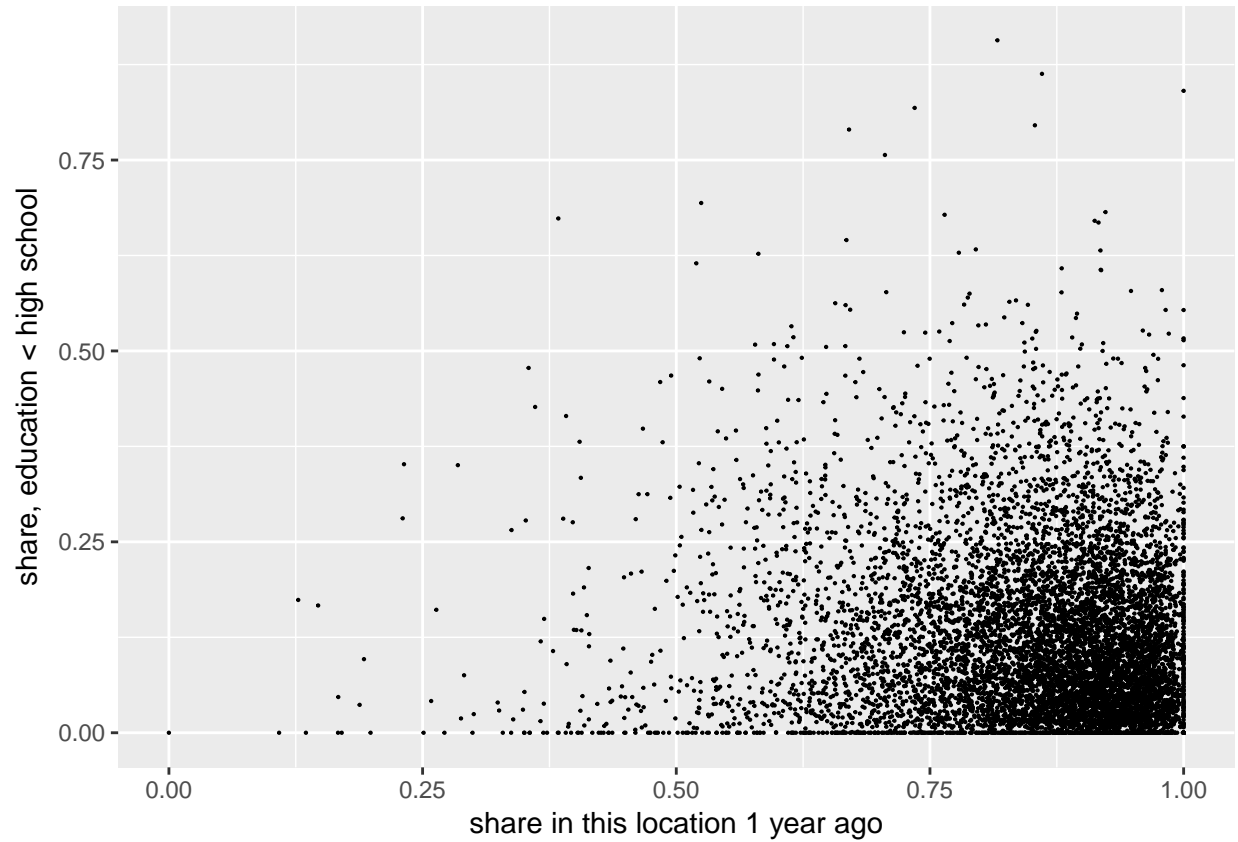
```
## Warning: Removed 1154 rows containing missing values (geom_point).
```



How about a smaller point to make things more visible? Set `size=0.1` in the `geom_point()` command.

```
# relationship between probability of moving and share low education  
ggplot(block.groups, aes(x=shr.here.1yr.ago, y=shr.ed.lt.hs)) +  
  labs(x = "share in this location 1 year ago", y = "share, education < high school") +  
  geom_point(size = 0.1)
```

```
## Warning: Removed 1154 rows containing missing values (geom_point).
```



I notice that there is very little data on the left-hand side of the chart, and I see whether my eyes are deceiving me by looking at the statistics below. The first shows the overall distribution (look at the 25th percentile, suggesting my eyes did not deceive me), and the second shows the distribution of a variable that takes on the values 1 when `block.groups$shr.here.1yr.ago > 0.5`.

```
summary(block.groups$shr.here.1yr.ago)
```

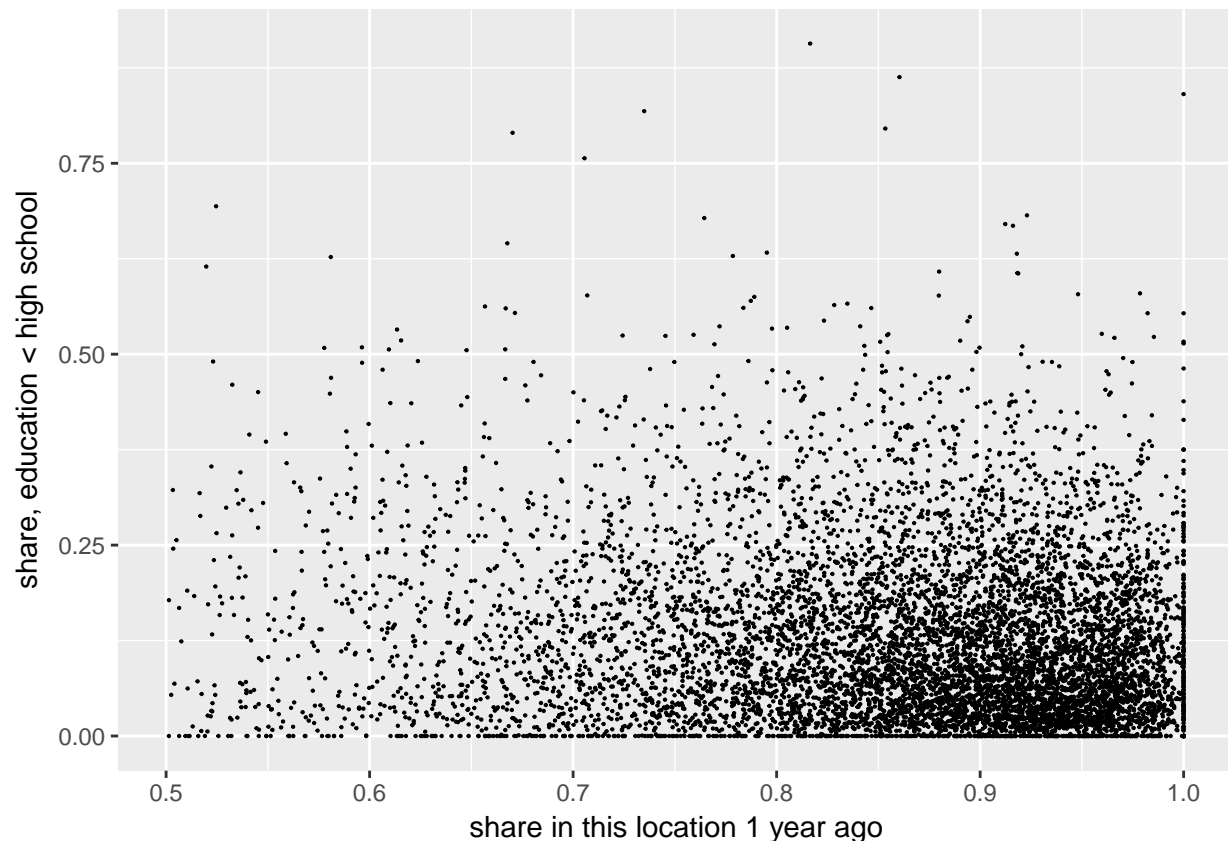
```
##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.   NA's
## 0.0000 0.7974 0.8821 0.8509 0.9376 1.0000 1151
```

```
summary(block.groups$shr.here.1yr.ago > 0.5)
```

```
##      Mode  FALSE    TRUE   NA's
## logical    140   8417  1151
```

Given this, let's focus on the block groups where the share of people who lived here a year ago is more than 50 percent. I do this by subsetting the data frame when I call it in the ggplot command. This is more efficient than subsetting outside of the command since it does not create another object for R to remember.

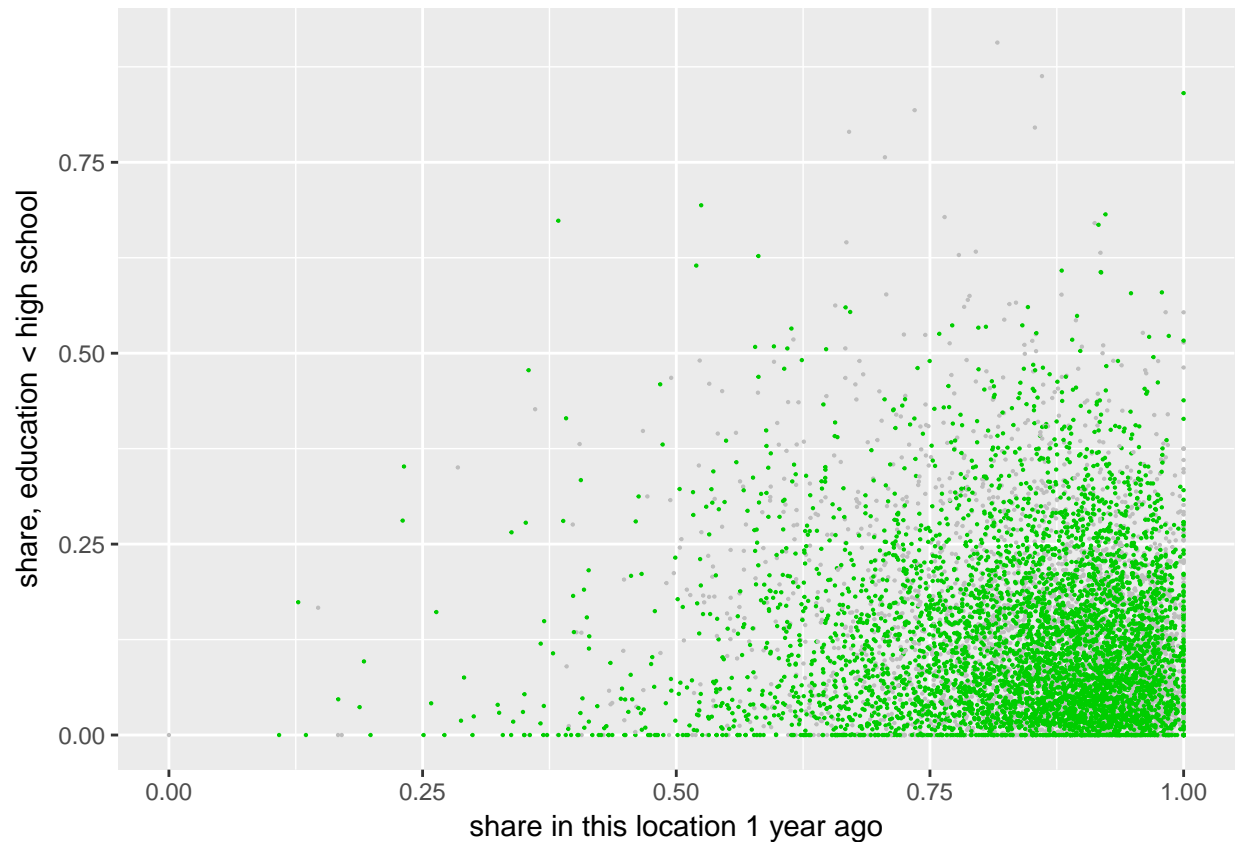
```
# relationship between probability of moving and share low education, where this is > 0.5
ggplot(block.groups[which(block.groups$shr.here.1yr.ago > 0.5),],
       aes(x=shr.here.1yr.ago, y=shr.ed.lt.hs)) +
  labs(x = "share in this location 1 year ago", y = "share, education < high school") +
  geom_point(size = 0.1)
```



Does this relationship vary by state? Let's make points in different colors by state to see. I do this by telling ggplot that there are groups by state in the ggplot command (`group = STATE`), and by coloring the points by state in the `geom_point()` command.

```
# different relationship between variables for dc, md, va?
ggplot(block.groups, aes(x=shr.here.1yr.ago, y=shr.ed.lt.hs), group = STATE) +
  labs(x = "share in this location 1 year ago", y = "share, education < high school") +
  geom_point(color=block.groups$STATE, size = 0.1)
```

```
## Warning: Removed 1154 rows containing missing values (geom_point).
```

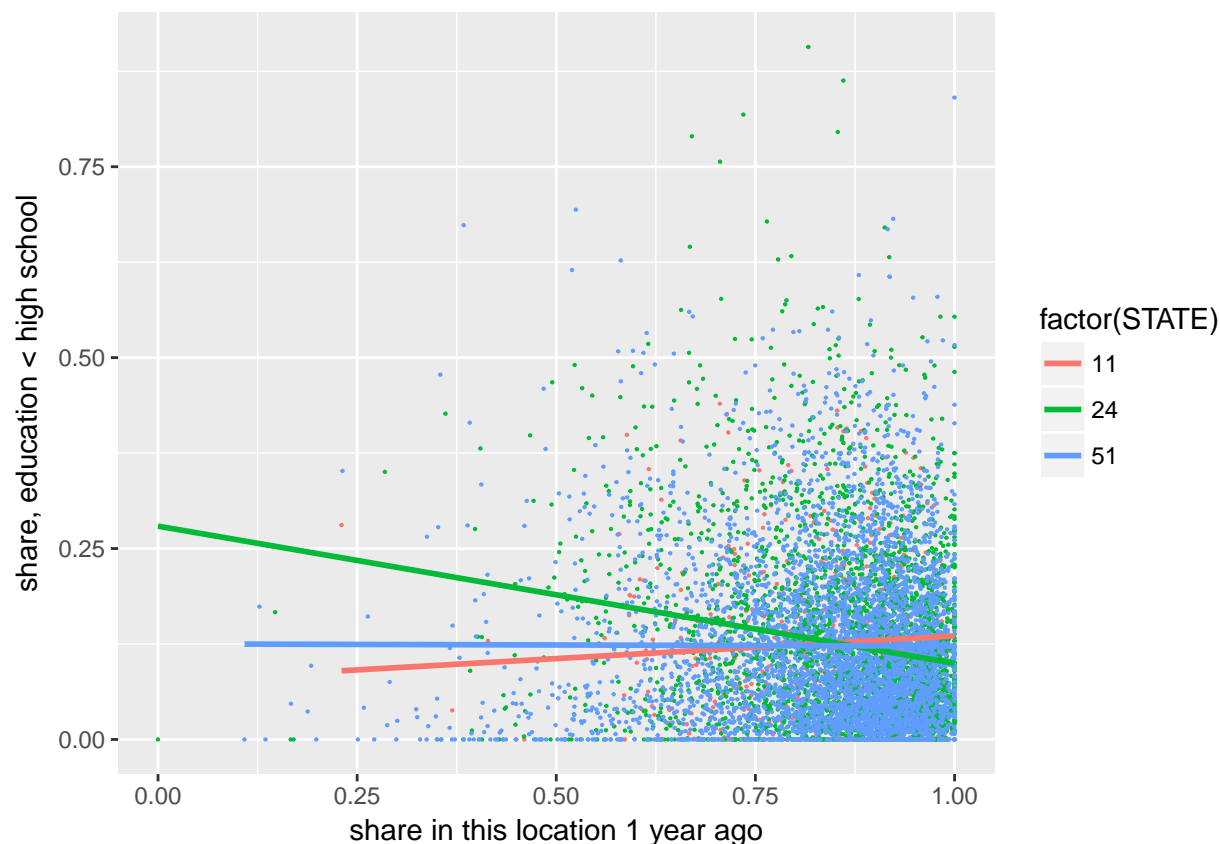


There are so many points it's hard to figure out what's going on! One (imperfect) solution is to add a trend, which I do below with the `geom_smooth(method = lm, se = FALSE)` command. The `method` option says "use a linear model", and `se = FALSE` means don't put confidence bounds on the line.

```
# add a state-specific trend line
ggplot(block.groups, aes(x=shr.here.1yr.ago, y=shr.ed.lt.hs, color = factor(STATE)),
      group = factor(STATE)) +
  labs(x = "share in this location 1 year ago", y = "share, education < high school") +
  geom_point(size = 0.1) +
  geom_smooth(method = lm, se = FALSE)
```

```
## Warning: Removed 1154 rows containing non-finite values (stat_smooth).
```

```
## Warning: Removed 1154 rows containing missing values (geom_point).
```



In two subsections, we'll explore other options for making this type of graph legible.

B.3 Levels vs Logs

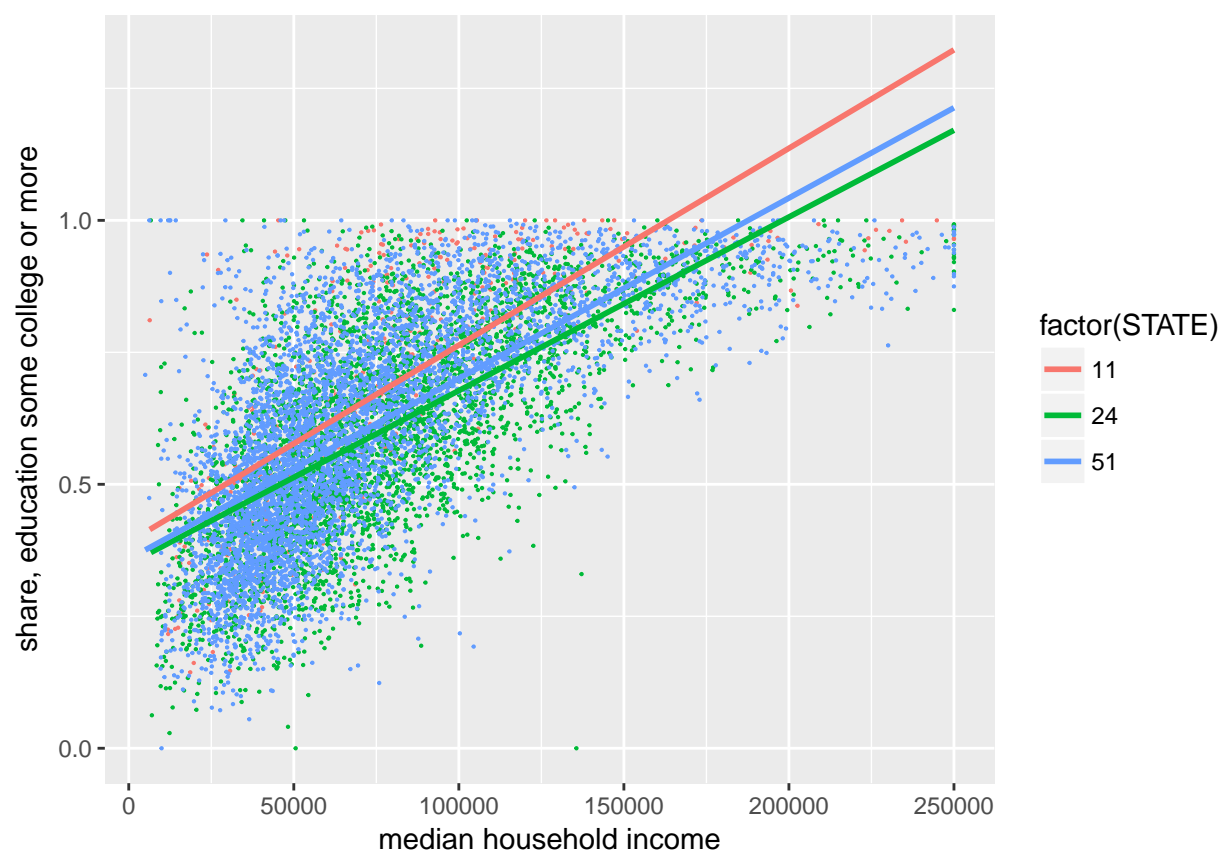
As a reinforcement from the previous lecture, let's look at the value of using logged variables.

First, look at the correlation between income and the block group's share of people with an education of some college or more by state.

```
# income and education by state, levels
ggplot(block.groups, aes(x=med.hh.inc, y=shr.ed.ge.sc, color = factor(STATE)),
      group = factor(STATE)) +
  labs(x = "median household income", y = "share, education some college or more") +
  geom_point(size = 0.1) +
  geom_smooth(method = lm, se = FALSE)
```

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

```
## Warning: Removed 108 rows containing missing values (geom_point).
```

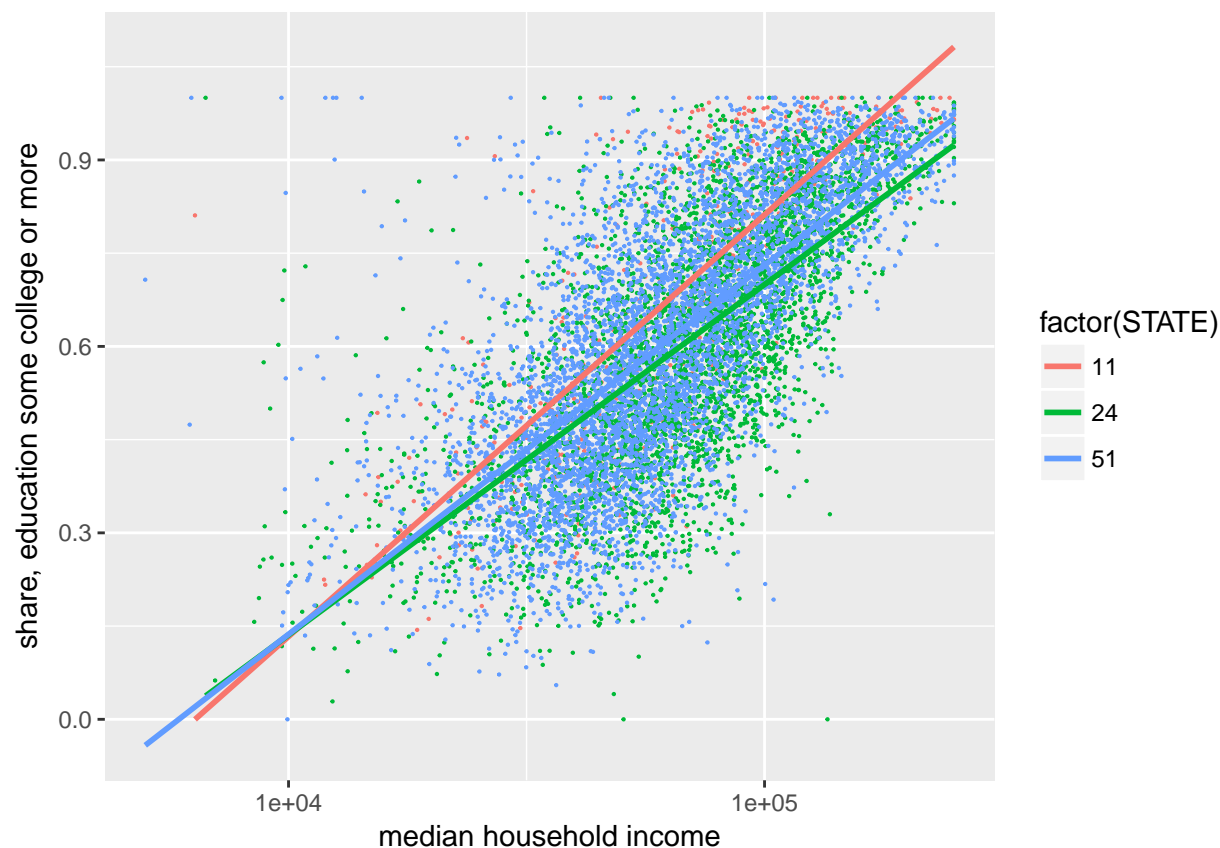


Notice how quite a bit of the income is on the left third of the graph. This clumping is called “bunching,” and taking a log of the data can often solve this problem. Let’s use R’s built in command to scale the x axis with a base 10 log: `scale_x_log10()`. Nicely, this labels the x axis values with non-logged numbers.

```
# income and education by state, base 10 log income
ggplot(block.groups, aes(x=med.hh.inc, y=shr.ed.ge.sc, color = factor(STATE)), group = factor(STATE)) +
  labs(x = "median household income", y = "share, education some college or more") +
  scale_x_log10() +
  geom_point(size = 0.1) +
  geom_smooth(method = lm, se = FALSE)
```

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

```
## Warning: Removed 108 rows containing missing values (geom_point).
```

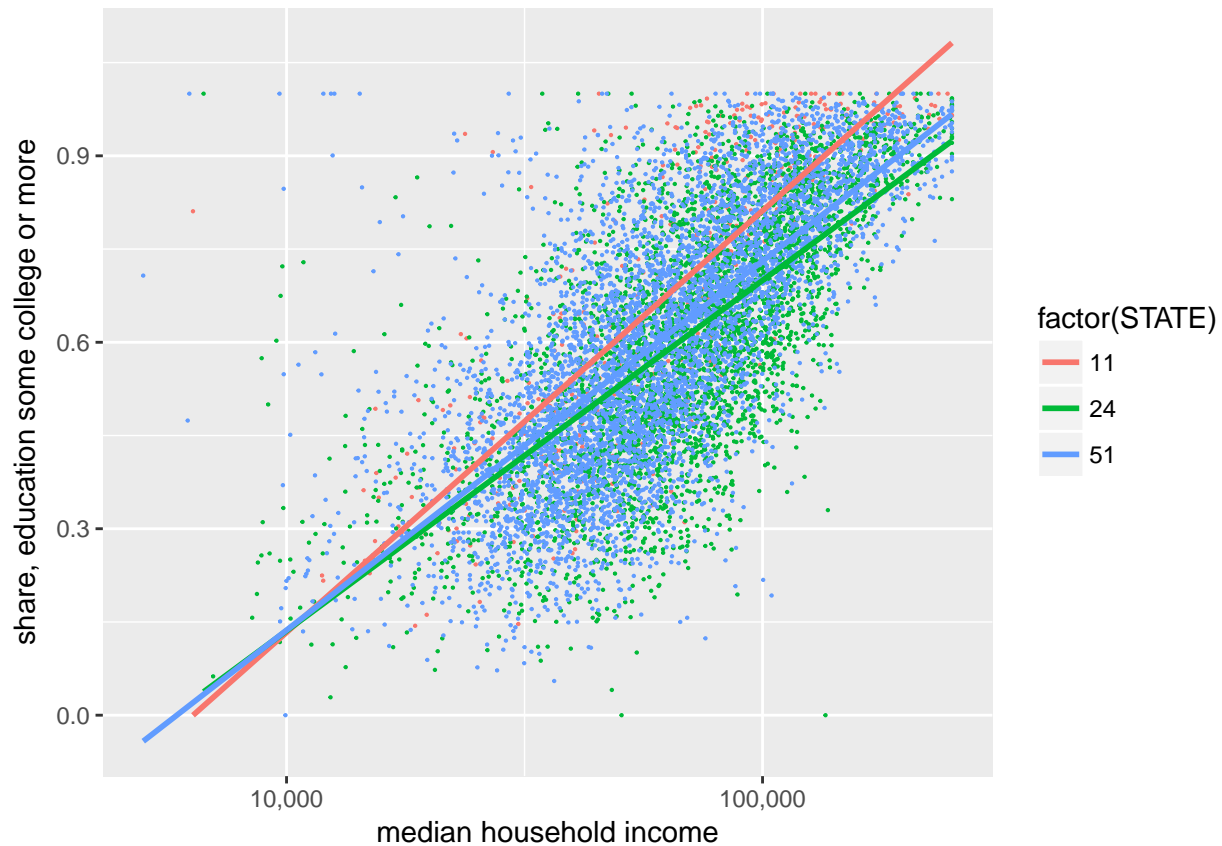


As we've done before, we'll use `label = comma` to get rid of the horrible scientific notation. Recall that this requires the library `scales`.

```
# income and education by state, base 10 log income
library(scales)
ggplot(block.groups, aes(x=med.hh.inc, y=shr.ed.ge.sc, color = factor(STATE)),
      group = factor(STATE)) +
  labs(x = "median household income", y = "share, education some college or more") +
  scale_x_log10(label=comma) +
  geom_point(size = 0.1) +
  geom_smooth(method = lm, se = FALSE)
```

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

```
## Warning: Removed 108 rows containing missing values (geom_point).
```



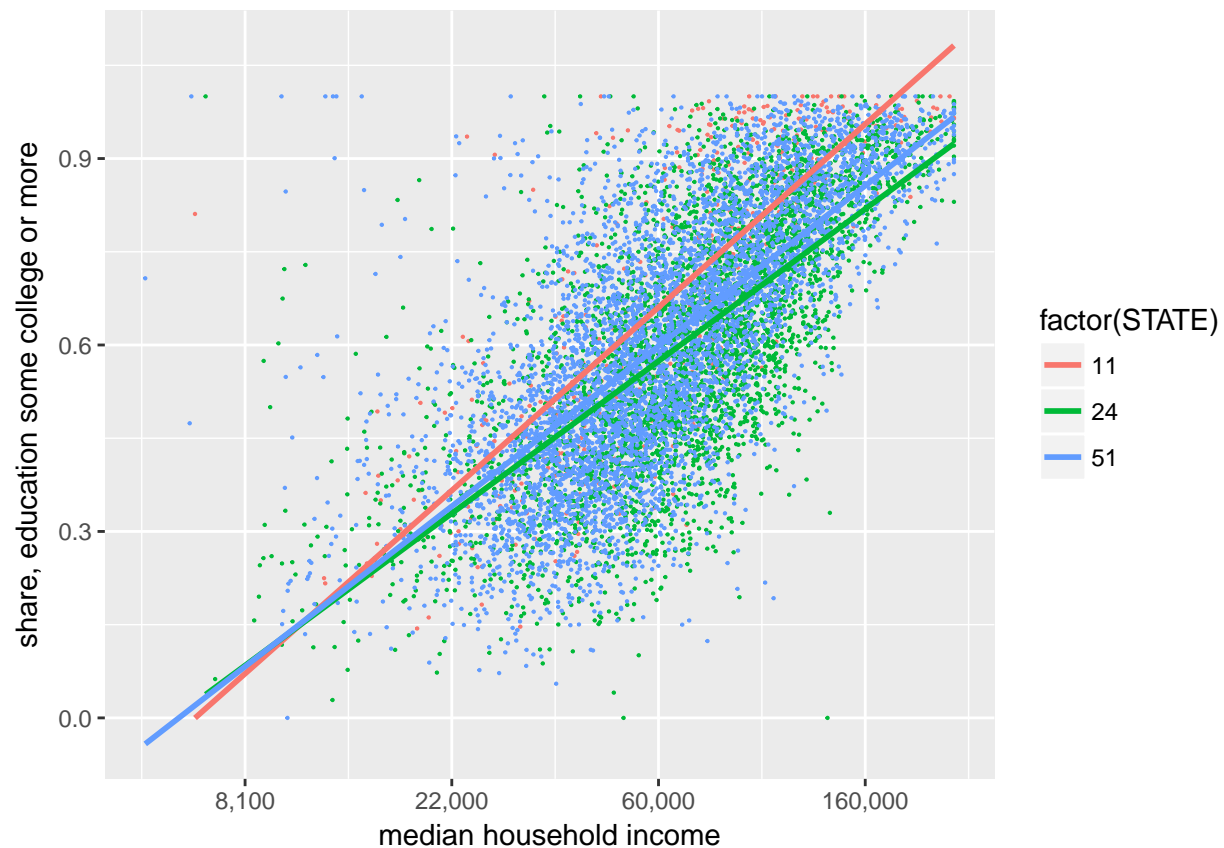
But taking a base 10 log may be too big – we've only got two integers to look at. Instead, labor economists of use the natural log, which has a base of the number e (look this number up if you don't know it).

To do this, I create this natural log in the first command below. Confusingly, `log()` is the natural log. Further, I label the breaks with the (rounded) version of their true values using the `scale_x_continuous()` below.

```
# income and education by state, natural log of income
block.groups$ln.med.hh.inc <- log(block.groups$med.hh.inc)
ggplot(block.groups, aes(x=ln.med.hh.inc, y=shr.ed.ge.sc, color = factor(STATE)),
      group = factor(STATE)) +
  labs(x = "median household income", y = "share, education some college or more") +
  geom_point(size = 0.1) +
  geom_smooth(method = lm, se = FALSE) +
  scale_x_continuous(breaks=c(9,10,11,12), labels=c("8,100", "22,000", "60,000", "160,000"))
```

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

```
## Warning: Removed 108 rows containing missing values (geom_point).
```



B.4 Binning data

Binning data (taking averages across small bits of the x variable) can greatly improve the communication of data. We go through an example below. First, we look at the variable to decide what kind of bins would be reasonable. This is a Goldilocks kind of problem. Too few bins and the story is obscured by limited data. Too many bins and the data are obscured by too many points. Your goal is choose something in the happy middle.

I use the next command to create a vector called `breaks` that is a sequence of numbers from 8.4 to 12.5 (based on min and max from `summary`) by intervals of 0.2.

```
# make bins
summary(block.groups$ln.med.hh.inc)
```



```
##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.     NA's
##      8.517  10.697  11.075  11.066  11.464  12.429     108
```

```
# set up boundaries for intervals/bins
breaks <- c(seq(8.4,12.5,0.2))
breaks
```

```
## [1] 8.4 8.6 8.8 9.0 9.2 9.4 9.6 9.8 10.0 10.2 10.4 10.6 10.8 11.0
## [15] 11.2 11.4 11.6 11.8 12.0 12.2 12.4
```

Now I use the `cut` function to assign each block group to a break, which shows up in the new variable `block.groups$ln.inc.bin`. The command cuts the variable `block.groups$ln.med.hh.inc` at locations given in `breaks`, and we include values even if equal to the lowest break (`include.lowest = T`) and include values even if equal to the highest break (`right = FALSE`).

I check whether it seems like R has done what I'd like by making a table of this bin variable.

```
# bucketing data points into bins
block.groups$ln.inc.bin <- cut(block.groups$ln.med.hh.inc, breaks, include.lowest = T, right=FALSE)
table(block.groups$ln.inc.bin)
```

```
##
##      [8.4,8.6)  [8.6,8.8)  [8.8,9)  [9,9.2)  [9.2,9.4)  [9.4,9.6)
##              1           3           2           20          34          48
##      [9.6,9.8)  [9.8,10)  [10,10.2) [10.2,10.4) [10.4,10.6) [10.6,10.8)
##              88          142          272          495          778          1100
##      [10.8,11)  [11,11.2) [11.2,11.4) [11.4,11.6) [11.6,11.8) [11.8,12)
##             1307          1298          1264          1055          809          452
##      [12,12.2) [12.2,12.4]
##             285           109
```

Now that we have assigned each block group to a bin, we can take the average of the variables of interest using `ddply` (used earlier in this tutorial). The notable difference here is that we are taking an average by both the state and the income bin. In other words, we're creating one mean for DC for income bin 1 and a separate mean for Virginia for income bin 1.

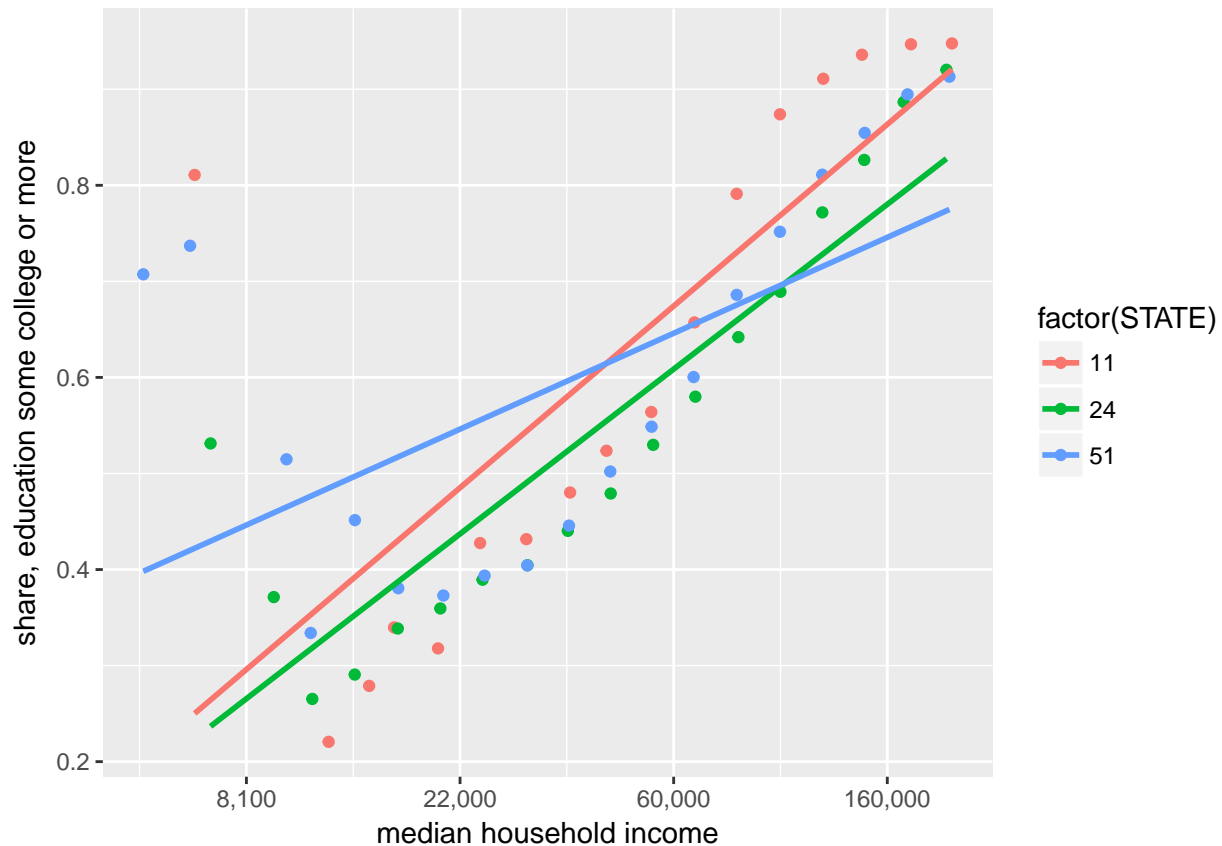
```
# take means by state and bin
inc.bins <- ddply(block.groups,
                  c("STATE", "ln.inc.bin"),
                  summarize,
                  av.ln.med.hh.inc = mean(ln.med.hh.inc),
                  av.shr.ed.ge.sc = mean(shr.ed.ge.sc))
```

Finally, we plot it. I think it is much better than the previous.

```
# plot it
# income and education by state, natural log of income
ggplot(inc.bins, aes(x=av.ln.med.hh.inc, y=av.shr.ed.ge.sc, color = factor(STATE)),
      group = factor(STATE)) +
  labs(x = "median household income", y = "share, education some college or more") +
  geom_point() +
  geom_smooth(method = lm, se = FALSE) +
  scale_x_continuous(breaks=c(9,10,11,12), labels=c("8,100", "22,000", "60,000", "160,000"))
```

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

```
## Warning: Removed 3 rows containing missing values (geom_point).
```



C. Homework

1. Improve one of the scatters from this tutorial. Almost all of them could be improved along multiple dimensions!
2. Find a data source of your own and make a scatter plot using binning.