Bar Charts

Leah Brooks

February 7, 2018

Today we work on a variety of bar charts. We'll go through some variations, and ideally you'll try some on your own at home.

To achieve bar charts that communicate, you'll frequently need to modify the underlying data frame. We also learn some steps to do this.

A. Basic Bar Chart

Inspired by the North Korean defectors (see lecture 3 notes), I decided to work on small bar chart that I thought might be an improvement.

Here I create a small dataframe and print it. The command `c()` defines columns, and I inputs years and the number of defectors per year. I then print the dataframe to make sure it seems ok.

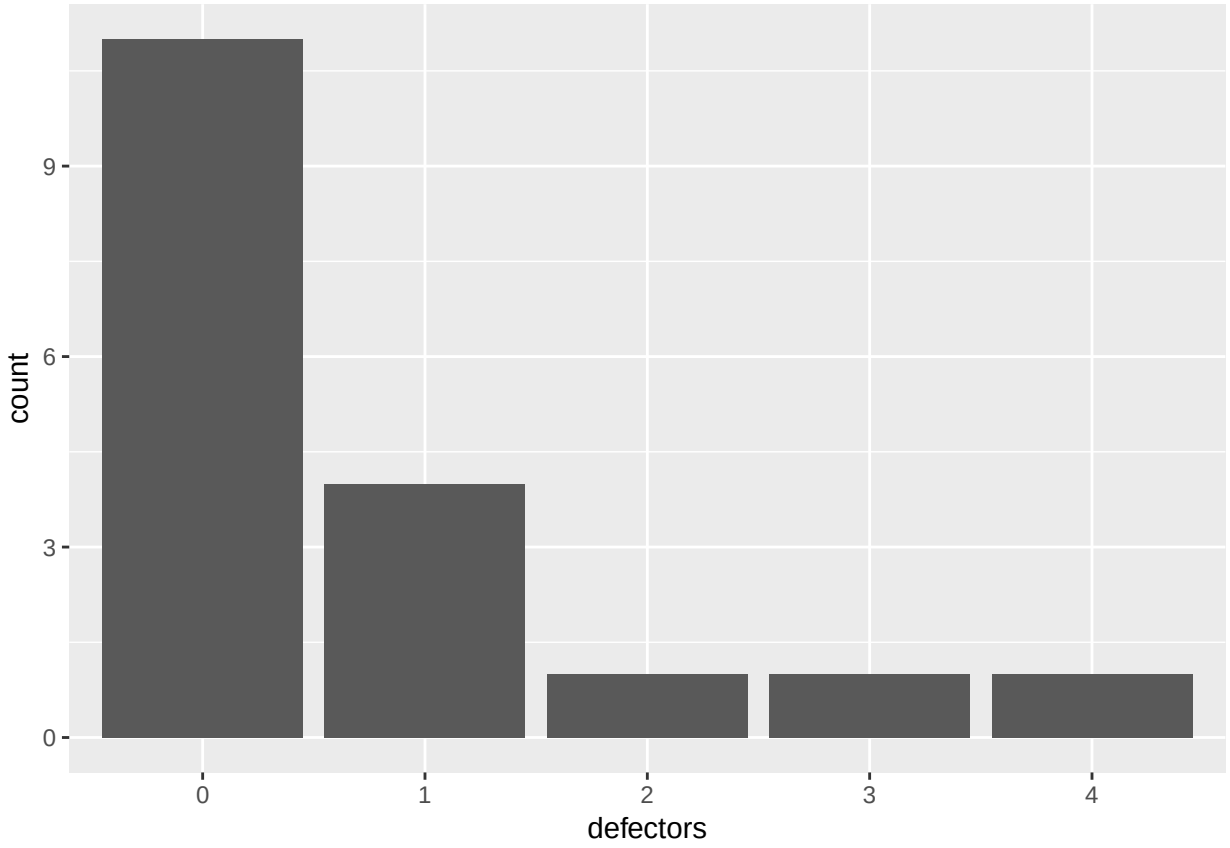
```
# load north korean data
nkd <- data.frame(year = c("2000","2001","2002","2003","2004",
                           "2005","2006","2007","2008","2009",
                           "2010","2011","2012","2013","2014",
                           "2015","2016","2017"),
                  defectors = c("0","0","1","0","0",
                                "0","0","0","2","0",
                                "1","0","3","0","0",
                                "1","1","4"))
nkd
```

```
##    year defectors
## 1  2000         0
## 2  2001         0
## 3  2002         1
## 4  2003         0
## 5  2004         0
## 6  2005         0
## 7  2006         0
## 8  2007         0
## 9  2008         2
## 10 2009         0
## 11 2010         1
## 12 2011         0
## 13 2012         3
## 14 2013         0
## 15 2014         0
## 16 2015         1
## 17 2016         1
## 18 2017         4
```

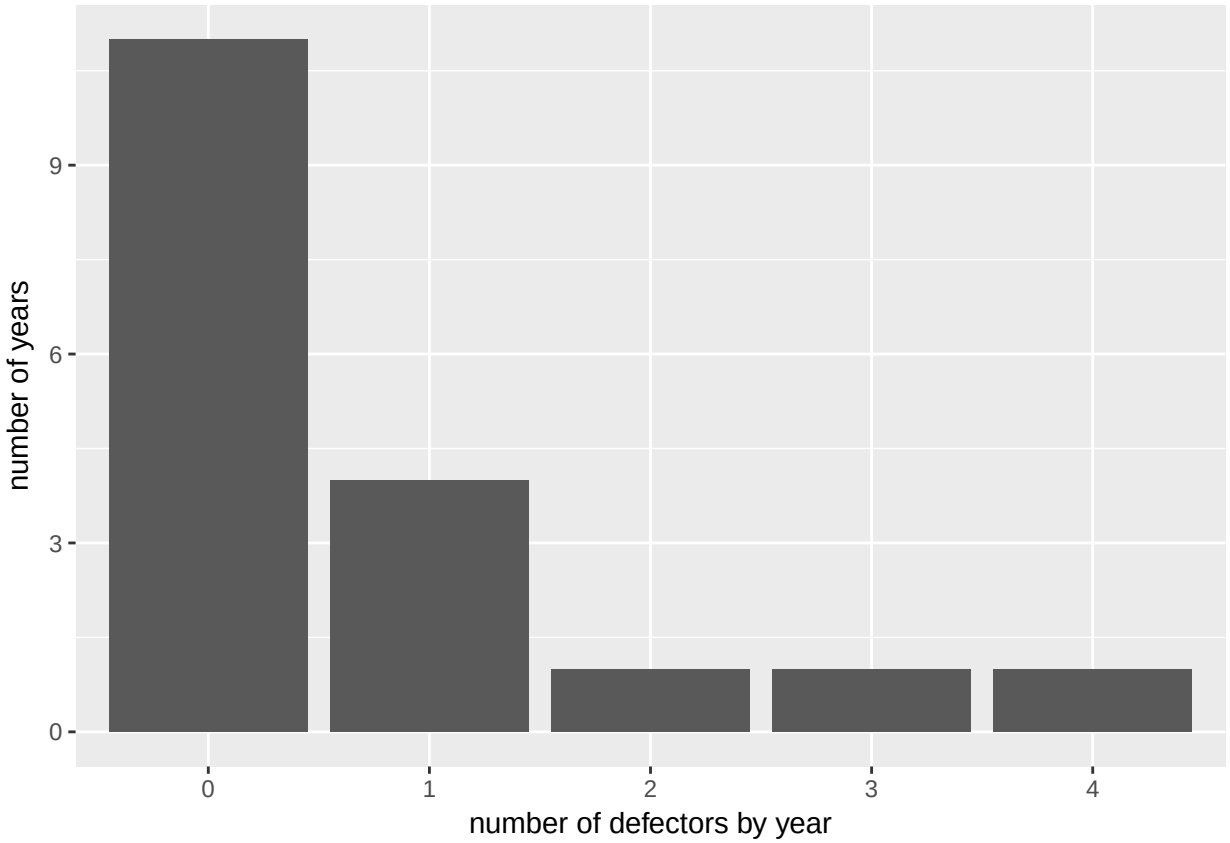
This points out a key part of making a bar chart in R. All the data you'd like in the chart should be in one column – even if there are multiple groups, as we'll see below.

So we start with a very simple bar chart. Load the `ggplot2` library, and like last class, set the data and aesthetics in the `ggplot` command. Use `geom_bar()` to tell R what you'd like on the graph, and `labs()` to label the axes.

```
# make a bar chart
library(ggplot2)
ggplot(nkd, aes(x=defectors)) + geom_bar()
```




```
ggplot(nkd, aes(x=defectors)) + geom_bar() +
  labs(x = "number of defectors by year", y="number of years")
```



But I realize that I can make a way cooler graph (and keep more of the information in the data) by putting the names of the years into each bar. The problem to generate this is two-fold. First, I need to tell R where to put the year numbers inside the bar, and second I need to make a background behind all the year numbers.

After some investigation, I think the only way to do this is a "fake" bar chart. In essence, I make a scatter plot with boxes around the year numbers so they look like a bar. However, I still remain dissatisfied with my final result because I haven't figured out how to get the background to go behind the entire bar.

Hopefully this will illustrate the difference between these two types of charts.

To get the years in the right place, I make a cumulative sum by defector number (a cumulative sum adds up values; the cumulative sum of a vector (1,2,3) is (1,3,6)). First I make a new variable called "marker" that is always equal to one. I then use the `ave ()` function, which creates (in general) group averages over level combinations of factors. Here we are trying to create a group cumulative sum (for marker) over level combinations of factors (defectors). Thus, I sum markers by defectors type, to get a cumulative sum (`FUN=cumsum`).

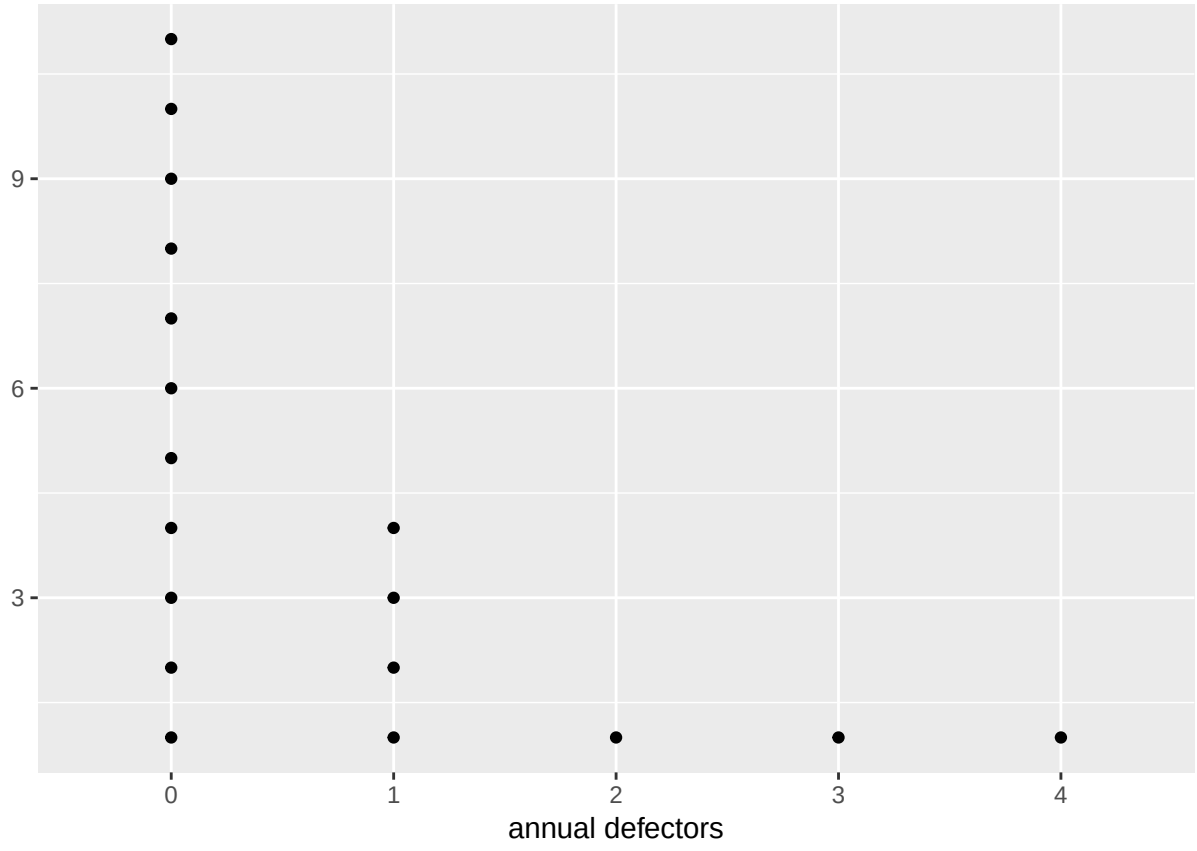
```
# make the dataset to have a cumulative total by defectors
nkd$marker <- 1
nkd$cum.defect <- ave(nkd$marker, nkd$defectors, FUN=cumsum)
nkd
```

```
##    year defectors marker cum.defect
## 1  2000         0      1          1
## 2  2001         0      1          2
## 3  2002         1      1          1
## 4  2003         0      1          3
## 5  2004         0      1          4
## 6  2005         0      1          5
```

```
## 7  2006         0      1          6
## 8  2007         0      1          7
## 9  2008         2      1          1
## 10 2009         0      1          8
## 11 2010         1      1          2
## 12 2011         0      1          9
## 13 2012         3      1          1
## 14 2013         0      1         10
## 15 2014         0      1         11
## 16 2015         1      1          3
## 17 2016         1      1          4
## 18 2017         4      1          1
```

To get a sense of why I need this cumulative sum, look at the chart below, which makes points at the appropriate x and y, and labels them with the year.

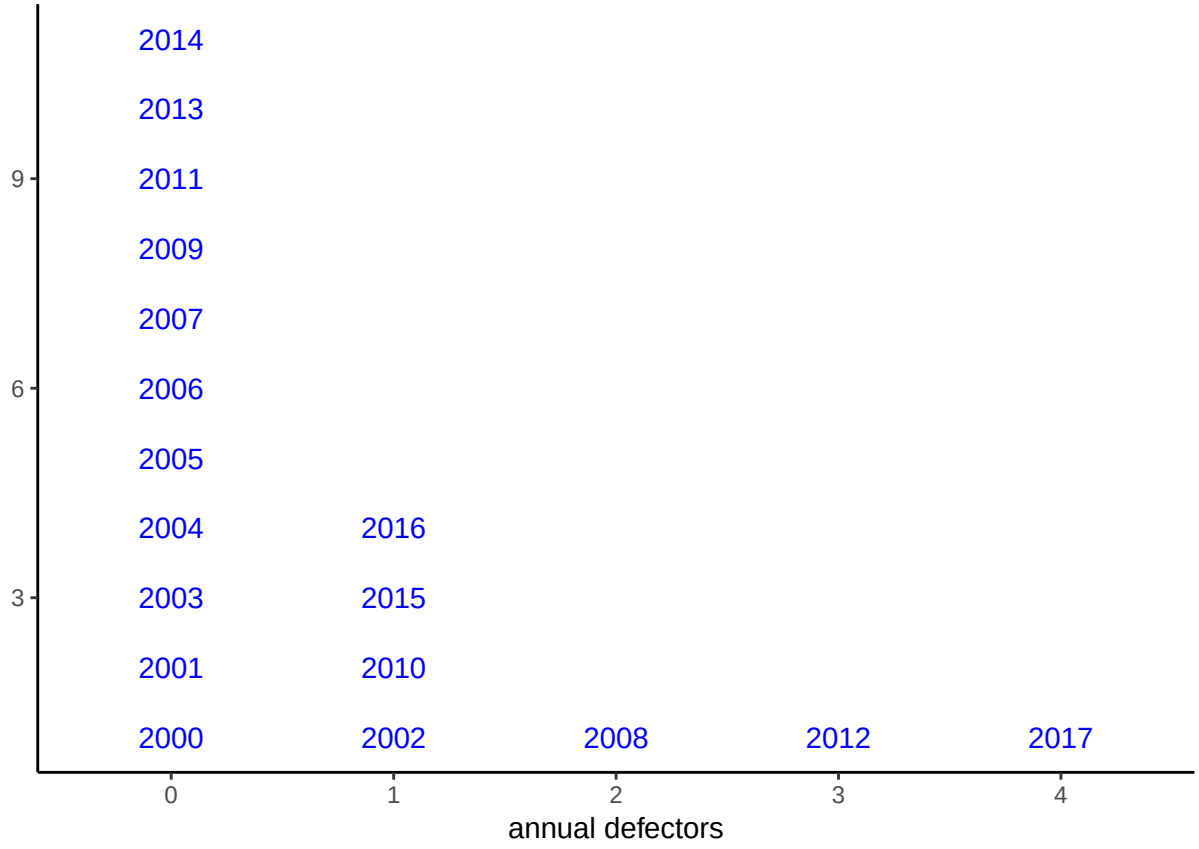
```
ggplot(nkd, aes(x=defectors, y=cum.defect, label=paste(year))) +
  geom_point(fill = "blue", color="black") +
  labs(x="annual defectors", y="")
```



Now I'm ready to make a fake bar chart. I do that below and then get rid of a lot of background stuff by setting theme elements to missing (`element_blank()`).

```
# make a scatter plot with markers
ggplot(nkd, aes(x=defectors, y=cum.defect, label=paste(year))) +
  geom_text(color = "blue") +
  labs(x="annual defectors", y="") +
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
```

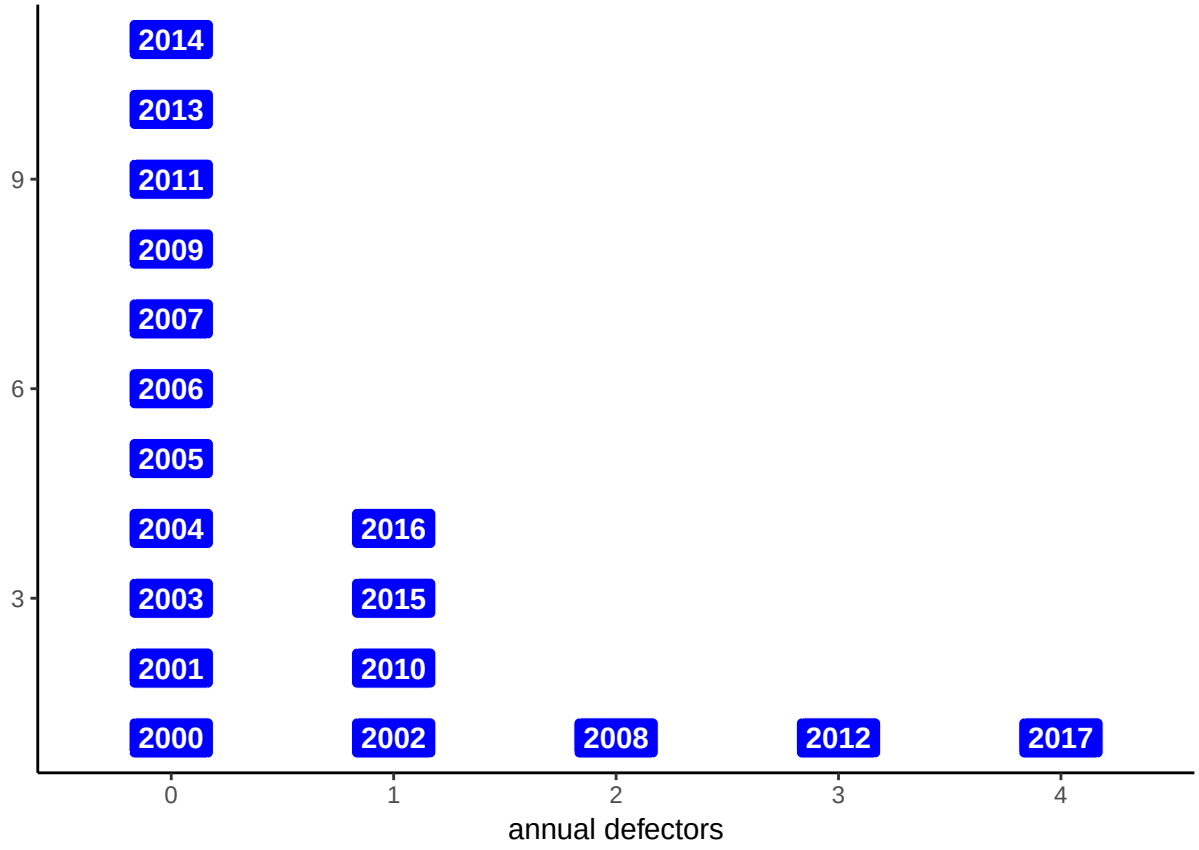
```
        panel.background = element_blank(), axis.line = element_line(colour = "black"))
```



I don't like that this doesn't really look like a bar.

I try to fix this by trying `geom_label()`, which puts a box behind whatever text you put on the graph.

```
# make a scatter plot with markers
ggplot(nkd, aes(x=defectors, y=cum.defect, label=paste(year))) +
  geom_label(fill = "blue", color="white", fontface = "bold", label.size = 0.005) +
  labs(x="annual defectors", y="") +
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
        panel.background = element_blank(), axis.line = element_line(colour = "black"))
```



I remain dissatisfied with the white areas within each quasi-bar, but haven't figured out how to fix this.

B. Summarizing data and creating bar charts

In this section, we rely on the county data, which we summarize into the nine census divisions. Using these division-level data, we create bar charts. Our goal is to discover whether division differ in average county size (measured by population).

The first step is to load the data and assign each county to a division. The Census reports states by division here. I use this information in conjunction with R's `ifelse` command.

The syntax for `ifelse` is `ifelse(condition, [value if condition is true],[value if condition is not true])`. You can nest these, as I do below. This code is so long it gets cut off – but you should have enough of the basic idea to re-create it yourself.

```
# load the data
counties <- read.csv("h:/pppa_data_viz/2018/tutorials/lecture01/counties_1910to2010_20180115.csv")

## you assigned division or region in the tutorial for lecture 1
## assign census division
counties$division <- ifelse(counties$statefips == 9 | counties$statefips == 23 |
                              counties$statefips == 25 | counties$statefips == 33 |
                              counties$statefips == 44 | counties$statefips == 50, 1,
  ifelse(counties$statefips == 34 | counties$statefips == 36 |
           counties$statefips == 42, 2,
    ifelse(counties$statefips == 18 | counties$statefips == 17 |
             counties$statefips == 26 | counties$statefips == 39 |
             counties$statefips == 55, 3,
      ifelse(counties$statefips == 19 | counties$statefips == 20 |
```

```
            counties$statefips == 27 | counties$statefips == 29 |
            counties$statefips == 31 | counties$statefips == 38 |
            counties$statefips == 46, 4,
          ifelse(counties$statefips == 10 | counties$statefips == 11 |
                 counties$statefips == 12 | counties$statefips == 13 |
                 counties$statefips == 24 | counties$statefips == 37 |
                 counties$statefips == 45 | counties$statefips == 51 |
                 counties$statefips == 54, 5,
            ifelse(counties$statefips == 1 | counties$statefips == 21 |
                   counties$statefips == 28 | counties$statefips == 47,6,
              ifelse(counties$statefips == 5 | counties$statefips == 22 |
                     counties$statefips == 40 | counties$statefips == 48, 7,
                ifelse(counties$statefips == 4 | counties$statefips == 8 |
                       counties$statefips == 16 | counties$statefips == 35 |
                       counties$statefips == 30 | counties$statefips == 49 |
                       counties$statefips == 32 | counties$statefips == 56, 8,
                  ifelse(counties$statefips == 2 | counties$statefips == 6 |
                         counties$statefips == 15 | counties$statefips == 41 |
                         counties$statefips == 53,9,0)))))))))
# did it work?
table(counties$division)
```

```
##
##    1    2    3    4    5    6    7    8    9
##  737 1649 4801 6799 6370 4000 5155 2994 1644
```

We are going to use just 2010, so let's subset to 2010.

```
# just keep 2010
counties.2010 <- subset(counties, year == 2010)
```

Here is the key summarizing step (see notes from Lecture 3 if you have set-up questions). We want to sum the county data (about 3,000 obs) into 9 census divisions. We add up the values of variables such as population that are reasonable to add to the division level. Were we interested in division-level median income, we would need to average that variable, perhaps weighted by population, to get a sensible number.

The command below finds the total number of counties by division (`length(cv1)`, which measures the length of a vector) and totals other variables including population (cv1) and education things (the other ones).

You'll see that the data below are missing quite a bit for division 9. I don't know what, but you'd like to use these data for yorur brief, let me know and I'll fix the problem.

```
# now count the number of counties by division and other division totals
library(plyr)
divisions.2010 <- ddply(counties.2010, "division", summarize,
                                        total.cntes = length(cv1),
                                        cv1 = sum(cv1),
                                        cv8 = sum(cv8),
                                        cv9 = sum(cv9),
                                        cv10 = sum(cv10),
                                        cv11 = sum(cv11),
                                        cv12 = sum(cv12),
                                        cv13 = sum(cv13),
                                        cv14 = sum(cv14),
                                        cv15 = sum(cv15),
                                        cv16 = sum(cv16),
                                        cv17 = sum(cv17),
```

```
                                        cv18 = sum(cv18),
                                        cv19 = sum(cv19),
                                        cv20 = sum(cv20),
                                        cv21 = sum(cv21),
                                        cv22 = sum(cv22),
                                        cv23 = sum(cv23),
                                        cv24 = sum(cv24),
                                        cv25 = sum(cv25),
                                        cv26 = sum(cv26),
                                        cv27 = sum(cv27), na.rm = TRUE)
```

To find county size, I calculate people per county, using `transform` below.

```
# find average # of ppl/county
divisions.2010$ppl.by.cnty <- divisions.2010$cv1 / divisions.2010$total.cntes
divisions.2010
```

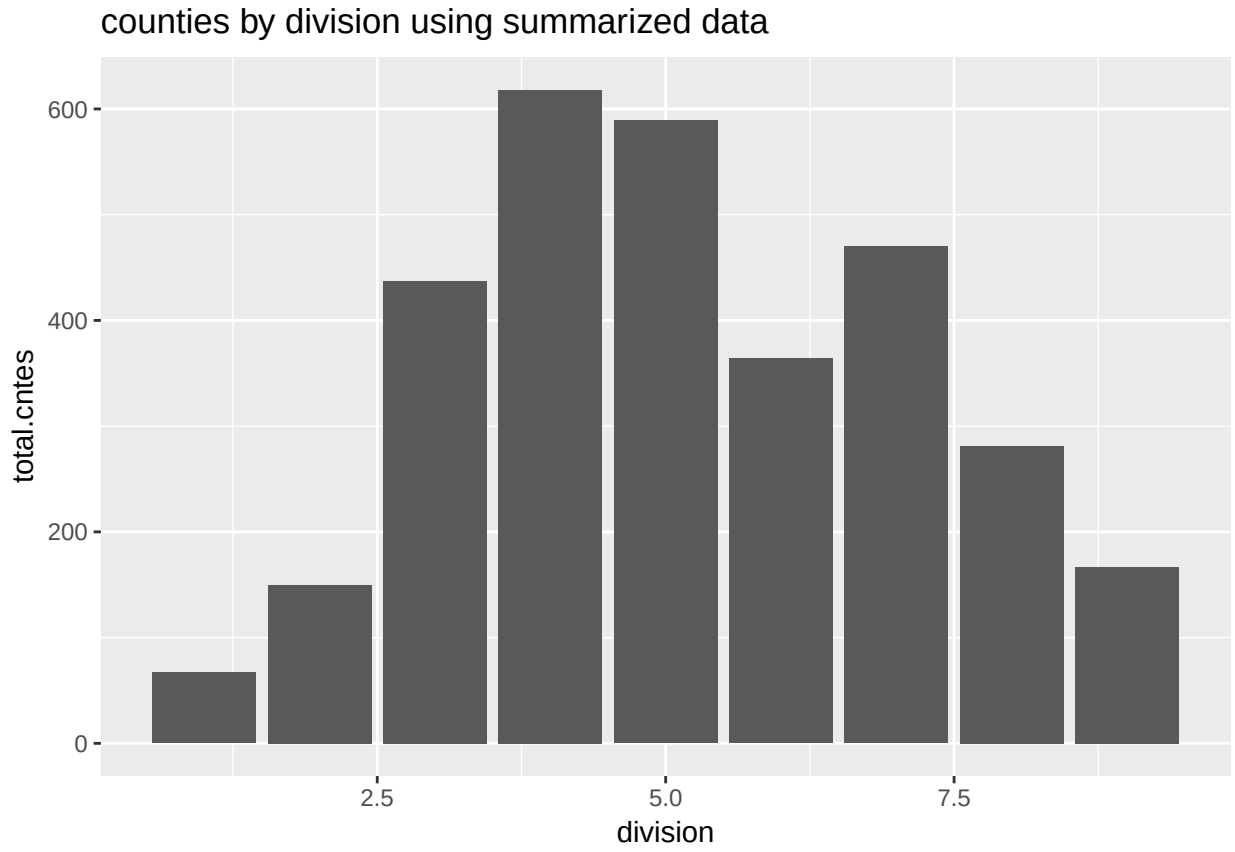
```
##   division total.cntes      cv1    cv8    cv9   cv10     cv11     cv12
## 1        1          67 14444865  50015  31459  43047  45330.5  45330.5
## 2        2         150 40872375 170343  90841 196522 138535.5 138535.5
## 3        3         437 46421564 142867  59876 146415 144157.5 144157.5
## 4        4         618 20505437  49492  23534  48272  72378.0  72378.0
## 5        5         589 59777037 240216 160840 314431 219085.0 219085.0
## 6        6         364 18432505  75197  46370  82533 105235.0 105235.0
## 7        7         470 36346202 184457 160807 335394 148163.5 148163.5
## 8        8         281 22065451  71067  56576 129916  60772.0  60772.0
## 9        9         167 49880102     NA     NA     NA       NA       NA
##      cv13    cv14    cv15    cv16    cv17   cv18   cv19   cv20     cv21
## 1  244172 1427360 1115164  983937  714468  58134  35261  43405  46261.0
## 2  806523 4304214 3100256 2502804 1694042 196197 111256 206721 155073.5
## 3  943420 5029409 4211704 2478762 1501889 144969  61195 133066 150669.5
## 4  344397 2108464 1995873 1208256  639790  53870  20118  40790  73122.0
## 5 1473673 5899337 5127538 3385554 2185153 240476 134061 276707 212496.5
## 6  549461 2048112 1534946  809108  467693  69765  33092  67593 107059.5
## 7  953058 3336683 3107160 1830797  979822 192414 158140 321792 148528.5
## 8  407631 1907276 2251548 1295973  766905  73044  51193 121447  58277.5
## 9      NA      NA      NA      NA      NA     NA     NA     NA       NA
##       cv22    cv23    cv24    cv25    cv26    cv27 na.rm ppl.by.cnty
## 1  46261.0  233840 1465703 1408382 1077808  769126  TRUE   215595.00
## 2 155073.5  834389 4756566 3619443 2693377 1859658  TRUE   272482.50
## 3 150669.5  942072 5262801 4924157 2682289 1529243  TRUE   106227.84
## 4  73122.0  330927 2092468 2297914 1324066  641171  TRUE    33180.32
## 5 212496.5 1379984 6343560 6357387 3739689 2193027  TRUE   101489.03
## 6 107059.5  549549 2121770 1928399  888830  527452  TRUE    50638.75
## 7 148528.5  926176 3493625 3585059 1982965  922973  TRUE    77332.34
## 8  58277.5  386155 1934043 2520593 1323491  675570  TRUE    78524.74
## 9       NA      NA      NA      NA      NA      NA  TRUE   298683.25
```

Now the data are ready to make pictures. Notice that key variables are in one column. Here I walk through a number of steps to produce better looking graphs.

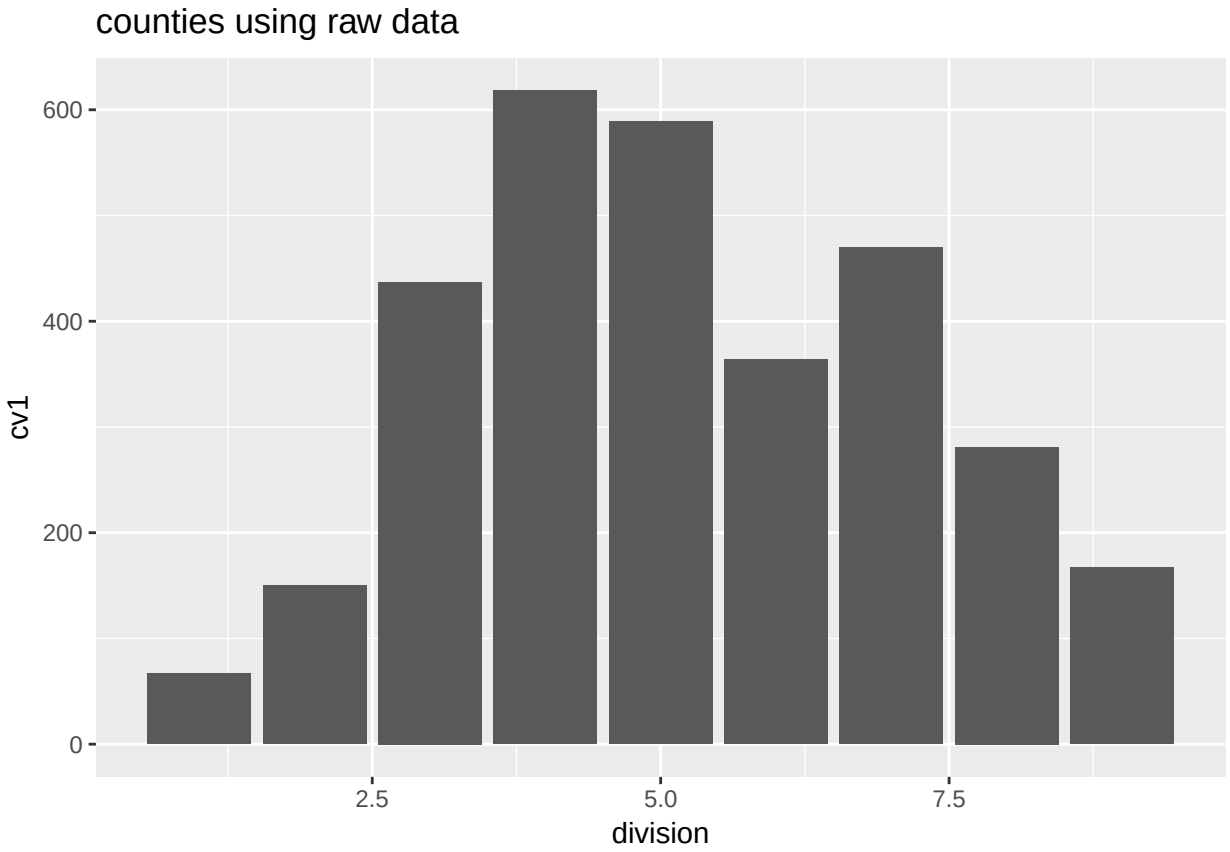
We start by charting total counties per division. I point out in the code that you can get to this same picture two ways: by summarizing the data as we did (first version) or by using the original `counties.2010` data and asking R to add up the number of counties. I am not sure this latter method will work for counties per capita, but you can check.

Like before, we rely on `geom_bar`. In the first case, we say that the statistic is `identity`, which means use the value of the y variable `total.cntes`. In the second case, we say that the statistic should be a total, which we indicate with `stat="summary"` of the y variable (cv1). The x variable is the division, which we want to range along the horizontal axis.

```
# counties per division
ggplot(data = divisions.2010, aes(x = division, y=total.cntes)) +
  geom_bar(stat = "identity") +
  ggtitle("counties by division using summarized data")
```

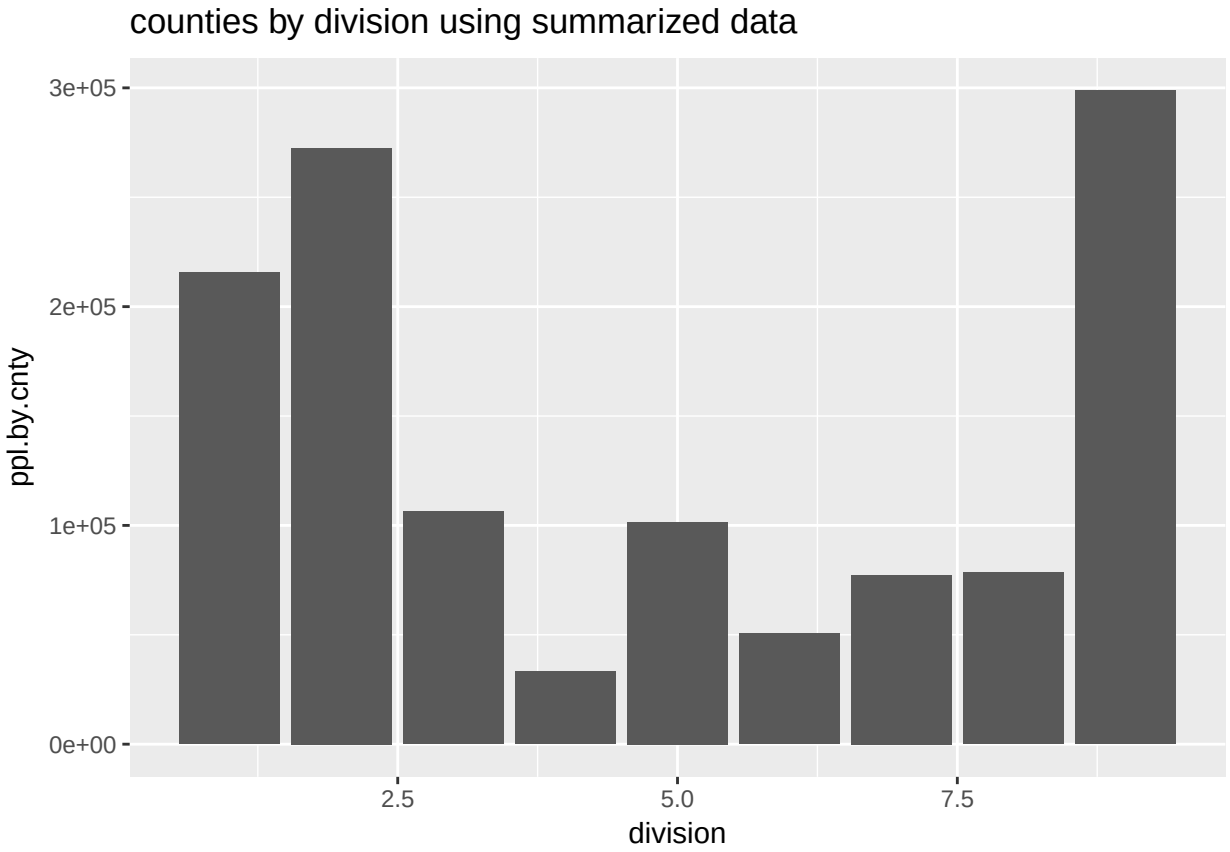



```
ggplot(data = counties.2010, aes(x = division, y=cv1)) +
  geom_bar(stat="summary", fun.y = "length") +
  ggtitle("counties using raw data")
```



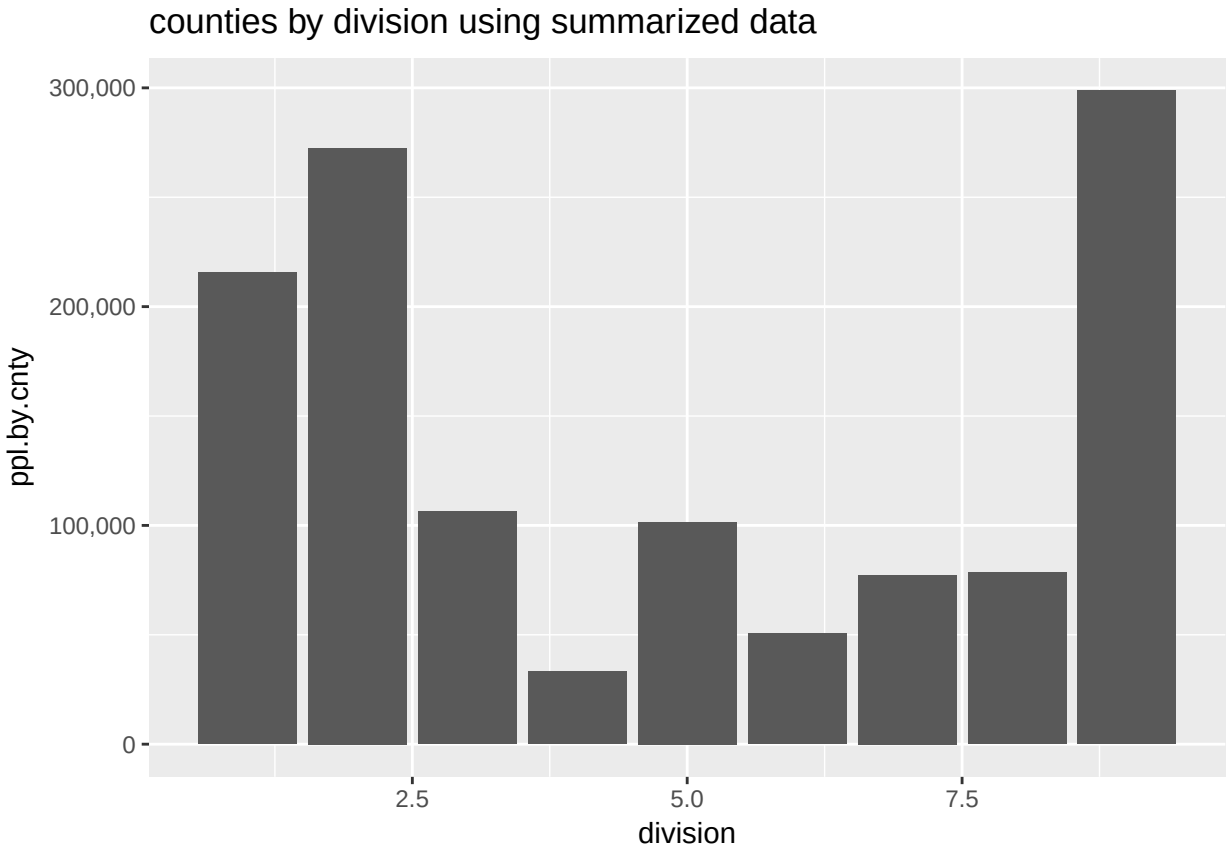
Of course, divisions could have differing number of counties based on population, so we now consider whether counties are systematically different, in terms of population, by division. To do this, we use the `ppl.by.cnty` variable we created.

```
# people per county per division
ggplot(data = divisions.2010, aes(x = division, y=ppl.by.cnty)) +
  geom_bar(stat = "identity") +
  ggtitle("counties by division using summarized data")
```



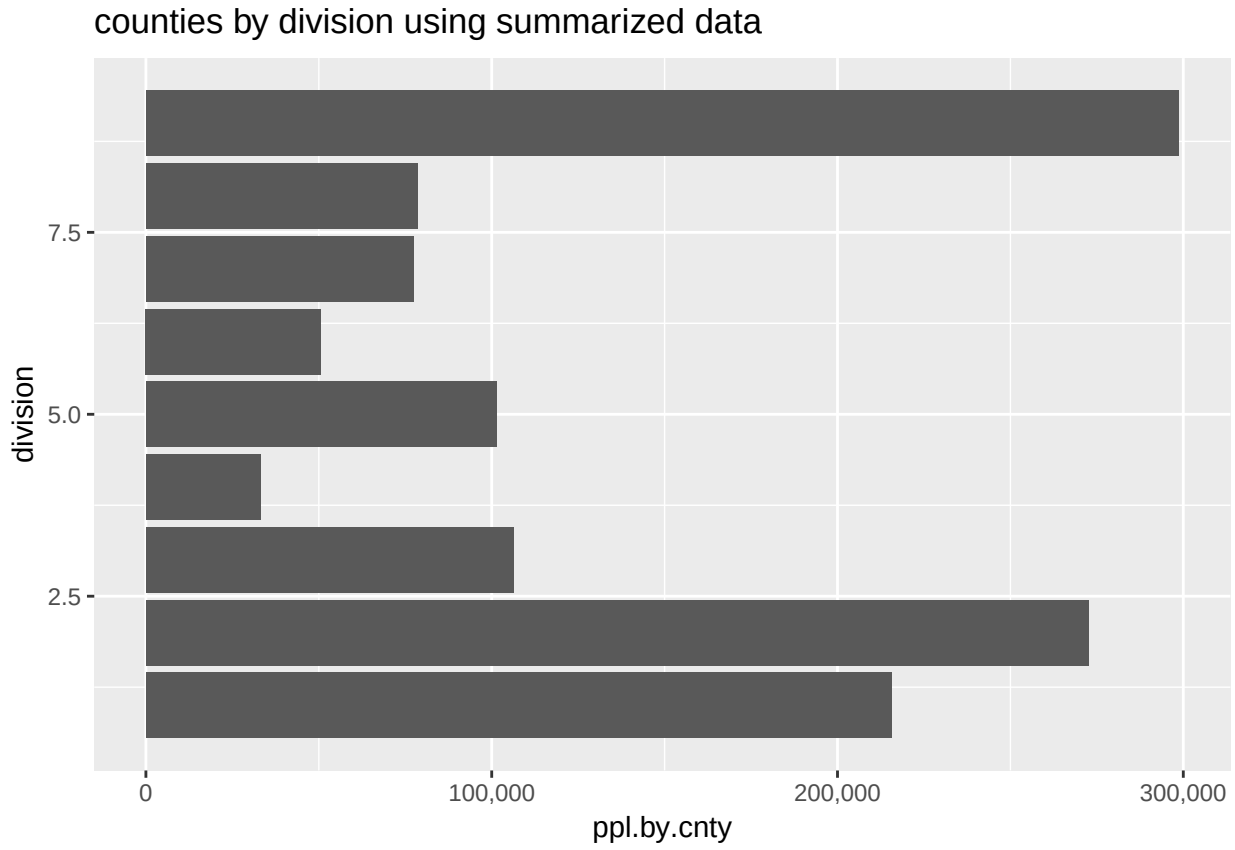
You may not be able to understand what this graph is saying due to the y labels. So add a command to modify the y axis labels: `scale_y_continuous(labels = comma)`. This command requires the `scales` library.

```
# people per county per division
library(scales)
ggplot(data = divisions.2010, aes(x = division, y=ppl.by.cnty)) +
  geom_bar(stat = "identity") +
  scale_y_continuous(label = comma) +
  ggtitle("counties by division using summarized data")
```



However, I think that this looks better horizontally, so we can add longer names. To make horizontal, add `coord_flip()`, though I believe all further references to axes should still line up with how you've called them in the `ggplot` command.

```
# people per county per division, horizontal plot
ggplot(data = divisions.2010, mapping = aes(x = division, y=ppl.by.cnty)) +
  geom_bar(stat = "identity") +
  scale_y_continuous(labels = comma) +
  ggtitle("counties by division using summarized data") +
  coord_flip()
```

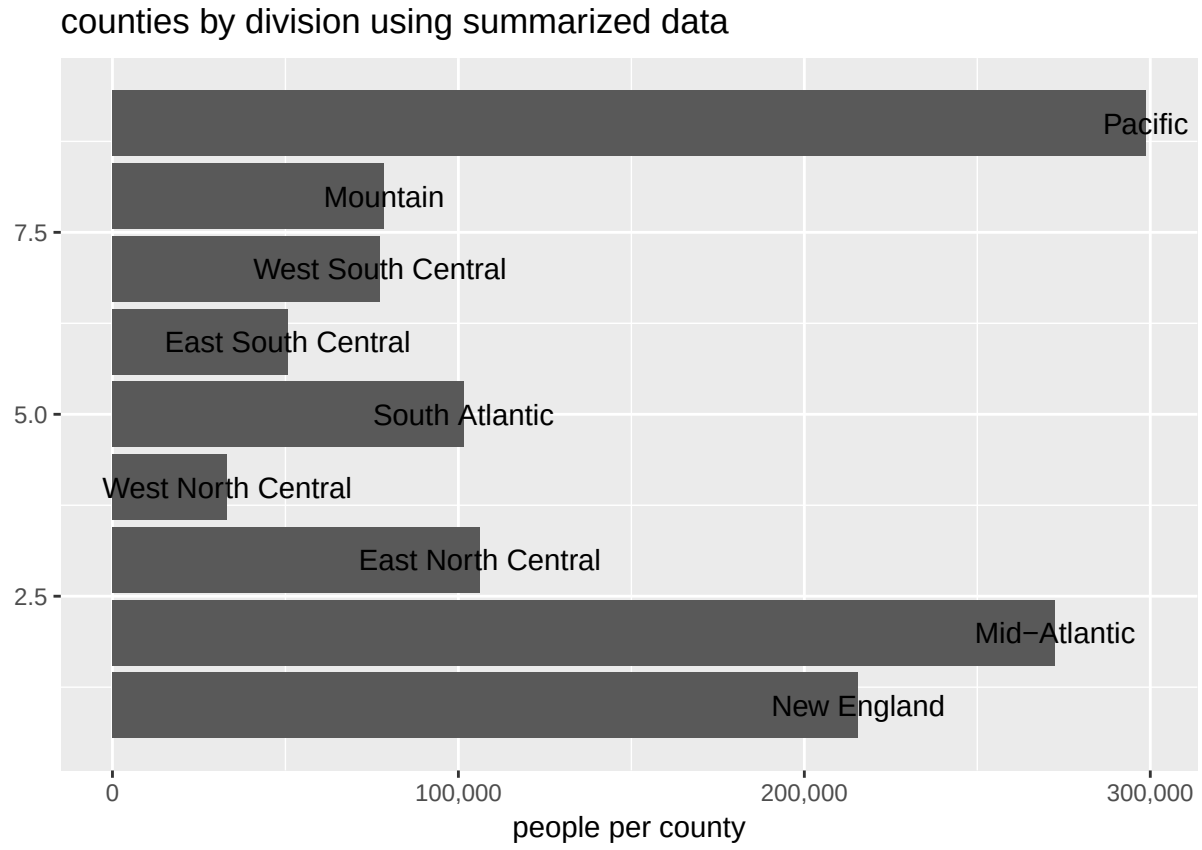


Now we need to add some text so we can add in division names. First I create a variable with division names:

```
# make a column with division names
divisions.2010$div.name <- c("New England", "Mid-Atlantic", "East North Central",
                         "West North Central", "South Atlantic", "East South Central",
                         "West South Central", "Mountain", "Pacific")
```

Now I add text to the chart with `geom_text`, telling R that the mapping for labels is `divisions$div.name`.

```
ggplot(data = divisions.2010, aes(x = division, y=ppl.by.cnty)) +
  geom_bar(stat = "identity") +
  scale_y_continuous(labels = comma) +
  ggtitle("counties by division using summarized data") +
  coord_flip() +
  labs(x="", y="people per county") +
  geom_text(mapping = aes(label=div.name))
```



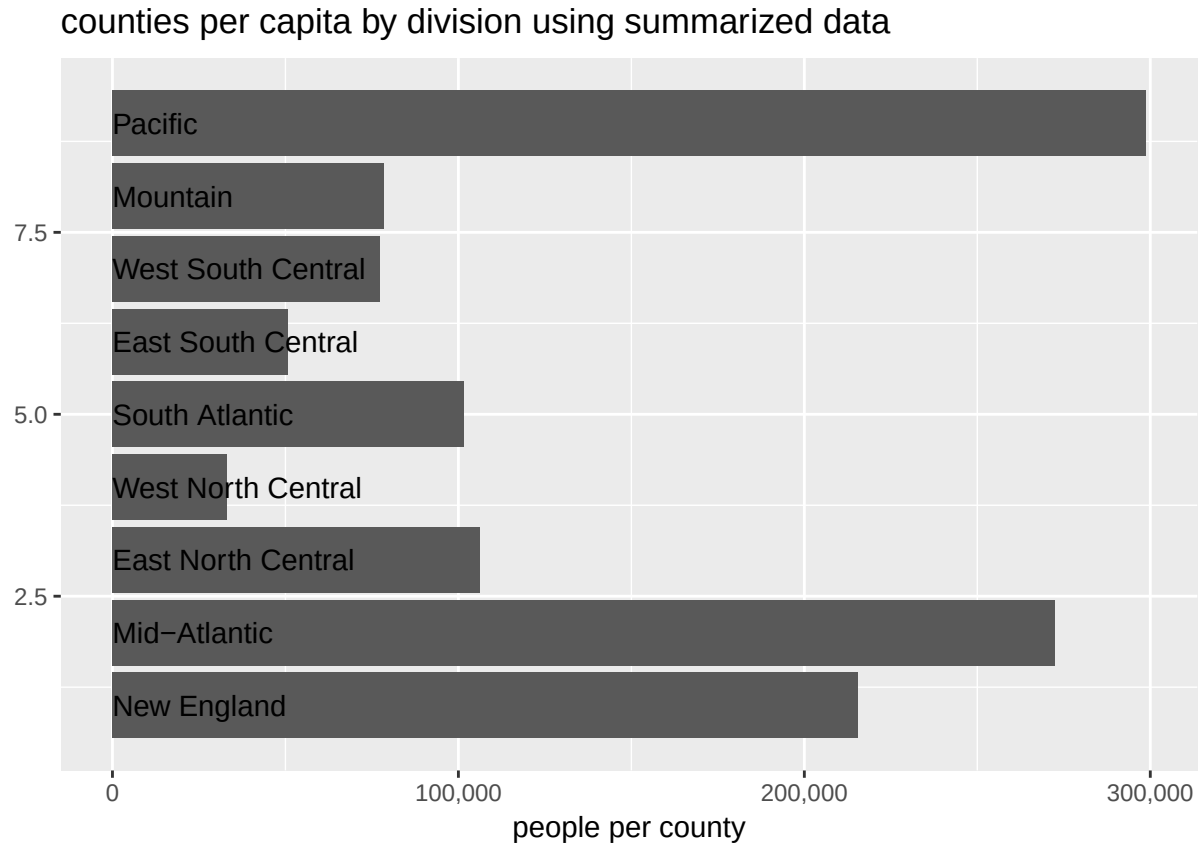
Unfortunately, this looks odd. So I try to fix by creating a variable that tells R where to put the labels. The command `rep()` means repeat, so I am telling R to name a new column in the dataframe `divisions` called `nada`, where it should repeat 0 for as many times as the length of the division name column (I could use any dataframe column here, I believe). I also include `hjust=0` to tell R that we want horizontal left justification (left is 0; right is 1; everything else is nonsense).

```
# make labels legible
divisions.2010$nada <- c(rep(0, length(divisions.2010$div.name)))
divisions.2010
```

```
##   division total.cntes      cv1    cv8    cv9   cv10     cv11     cv12
## 1        1          67 14444865  50015  31459  43047  45330.5  45330.5
## 2        2         150 40872375 170343  90841 196522 138535.5 138535.5
## 3        3         437 46421564 142867  59876 146415 144157.5 144157.5
## 4        4         618 20505437  49492  23534  48272  72378.0  72378.0
## 5        5         589 59777037 240216 160840 314431 219085.0 219085.0
## 6        6         364 18432505  75197  46370  82533 105235.0 105235.0
## 7        7         470 36346202 184457 160807 335394 148163.5 148163.5
## 8        8         281 22065451  71067  56576 129916  60772.0  60772.0
## 9        9         167 49880102     NA     NA     NA       NA       NA
##      cv13    cv14    cv15    cv16    cv17   cv18   cv19   cv20     cv21
## 1  244172 1427360 1115164  983937  714468  58134  35261  43405  46261.0
## 2  806523 4304214 3100256 2502804 1694042 196197 111256 206721 155073.5
## 3  943420 5029409 4211704 2478762 1501889 144969  61195 133066 150669.5
## 4  344397 2108464 1995873 1208256  639790  53870  20118  40790  73122.0
## 5 1473673 5899337 5127538 3385554 2185153 240476 134061 276707 212496.5
## 6  549461 2048112 1534946  809108  467693  69765  33092  67593 107059.5
```

```
## 7  953058 3336683 3107160 1830797  979822 192414 158140 321792 148528.5
## 8  407631 1907276 2251548 1295973  766905  73044  51193 121447  58277.5
## 9      NA      NA      NA      NA      NA     NA     NA     NA       NA
##       cv22    cv23    cv24    cv25    cv26    cv27 na.rm ppl.by.cnty
## 1  46261.0  233840 1465703 1408382 1077808  769126  TRUE   215595.00
## 2 155073.5  834389 4756566 3619443 2693377 1859658  TRUE   272482.50
## 3 150669.5  942072 5262801 4924157 2682289 1529243  TRUE   106227.84
## 4  73122.0  330927 2092468 2297914 1324066  641171  TRUE    33180.32
## 5 212496.5 1379984 6343560 6357387 3739689 2193027  TRUE   101489.03
## 6 107059.5  549549 2121770 1928399  888830  527452  TRUE    50638.75
## 7 148528.5  926176 3493625 3585059 1982965  922973  TRUE    77332.34
## 8  58277.5  386155 1934043 2520593 1323491  675570  TRUE    78524.74
## 9       NA      NA      NA      NA      NA      NA  TRUE   298683.25
##             div.name nada
## 1         New England    0
## 2        Mid-Atlantic    0
## 3 East North Central    0
## 4 West North Central    0
## 5     South Atlantic    0
## 6 East South Central    0
## 7 West South Central    0
## 8            Mountain    0
## 9             Pacific    0
```

```
ggplot(data = divisions.2010, aes(x = division, y=ppl.by.cnty)) +
  geom_bar(stat = "identity") +
  scale_y_continuous(labels = comma) +
  ggtitle("counties per capita by division using summarized data") +
  coord_flip() +
  labs(x="", y="people per county") +
  geom_text(mapping = aes(y=nada, label=div.name), hjust = 0)
```



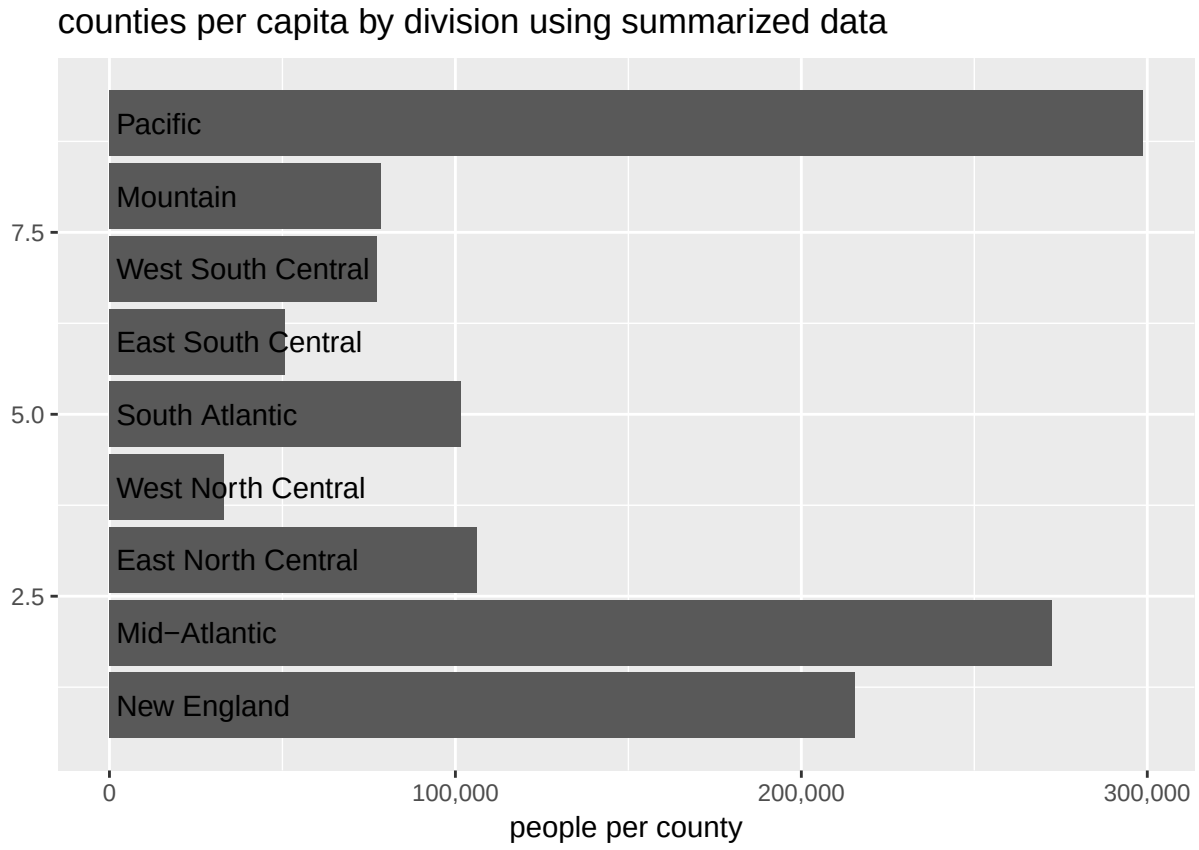
But this isn't actually pleasing becase the names are so far to the left. To fix this, I change the location of the names from 0 to 2000.

```
# make labels legible -- why is this one wacky?
divisions.2010$nada <- c(rep(2000, length(divisions.2010$div.name)))
divisions.2010
```

```
##   division total.cntes      cv1    cv8    cv9   cv10     cv11     cv12
## 1        1          67 14444865  50015  31459  43047  45330.5  45330.5
## 2        2         150 40872375 170343  90841 196522 138535.5 138535.5
## 3        3         437 46421564 142867  59876 146415 144157.5 144157.5
## 4        4         618 20505437  49492  23534  48272  72378.0  72378.0
## 5        5         589 59777037 240216 160840 314431 219085.0 219085.0
## 6        6         364 18432505  75197  46370  82533 105235.0 105235.0
## 7        7         470 36346202 184457 160807 335394 148163.5 148163.5
## 8        8         281 22065451  71067  56576 129916  60772.0  60772.0
## 9        9         167 49880102     NA     NA     NA       NA       NA
##      cv13    cv14    cv15    cv16    cv17   cv18   cv19   cv20     cv21
## 1  244172 1427360 1115164  983937  714468  58134  35261  43405  46261.0
## 2  806523 4304214 3100256 2502804 1694042 196197 111256 206721 155073.5
## 3  943420 5029409 4211704 2478762 1501889 144969  61195 133066 150669.5
## 4  344397 2108464 1995873 1208256  639790  53870  20118  40790  73122.0
## 5 1473673 5899337 5127538 3385554 2185153 240476 134061 276707 212496.5
## 6  549461 2048112 1534946  809108  467693  69765  33092  67593 107059.5
## 7  953058 3336683 3107160 1830797  979822 192414 158140 321792 148528.5
## 8  407631 1907276 2251548 1295973  766905  73044  51193 121447  58277.5
## 9      NA      NA      NA      NA      NA     NA     NA     NA       NA
```

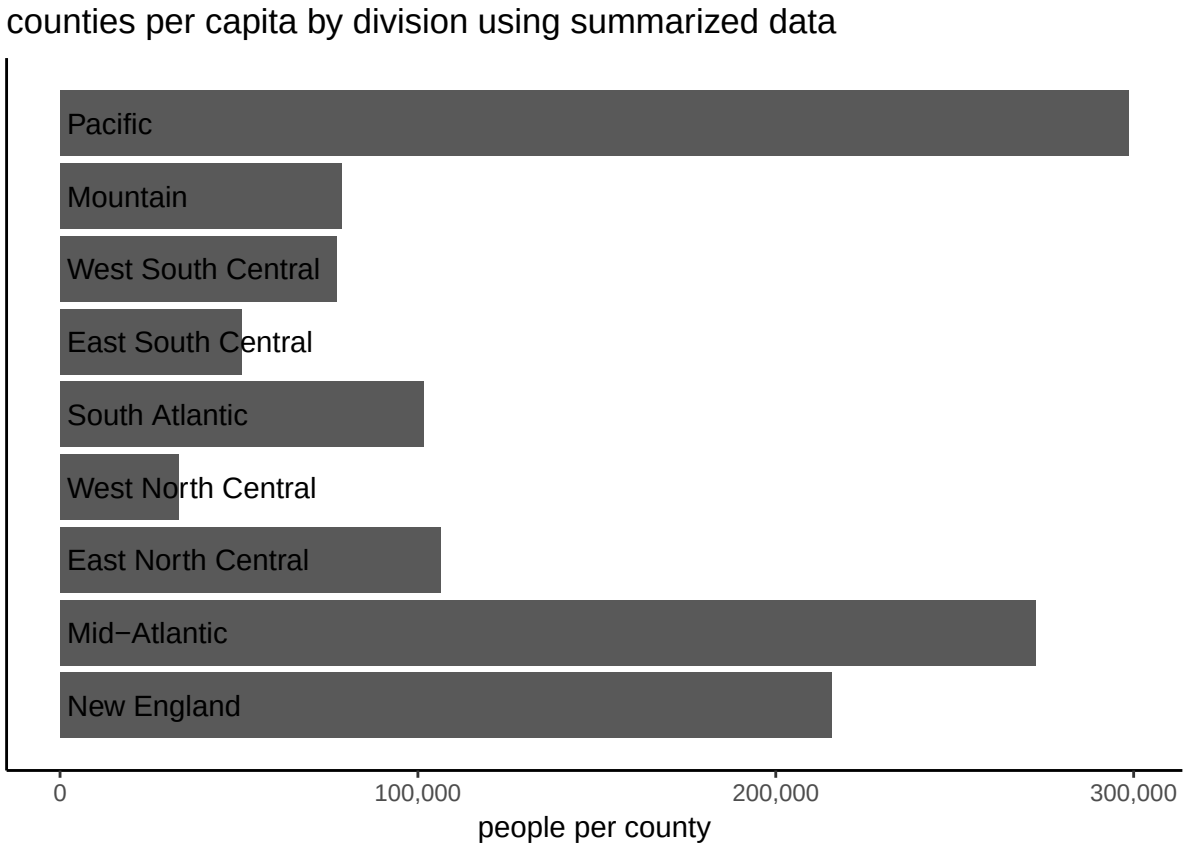
```
##       cv22    cv23    cv24    cv25    cv26    cv27 na.rm ppl.by.cnty
## 1  46261.0  233840 1465703 1408382 1077808  769126  TRUE   215595.00
## 2 155073.5  834389 4756566 3619443 2693377 1859658  TRUE   272482.50
## 3 150669.5  942072 5262801 4924157 2682289 1529243  TRUE   106227.84
## 4  73122.0  330927 2092468 2297914 1324066  641171  TRUE    33180.32
## 5 212496.5 1379984 6343560 6357387 3739689 2193027  TRUE   101489.03
## 6 107059.5  549549 2121770 1928399  888830  527452  TRUE    50638.75
## 7 148528.5  926176 3493625 3585059 1982965  922973  TRUE    77332.34
## 8  58277.5  386155 1934043 2520593 1323491  675570  TRUE    78524.74
## 9       NA      NA      NA      NA      NA      NA  TRUE   298683.25
##             div.name nada
## 1        New England 2000
## 2       Mid-Atlantic 2000
## 3 East North Central 2000
## 4 West North Central 2000
## 5     South Atlantic 2000
## 6 East South Central 2000
## 7 West South Central 2000
## 8           Mountain 2000
## 9            Pacific 2000
```

```r
ggplot(data = divisions.2010, mapping = aes(x = division, y=ppl.by.cnty)) +
  geom_bar(stat = "identity") +
  scale_y_continuous(labels = comma) +
  ggtitle("counties per capita by division using summarized data") +
  coord_flip() +
  labs(x="", y="people per county") +
  geom_text(mapping = aes(y=nada, label=div.name), hjust = 0)
```



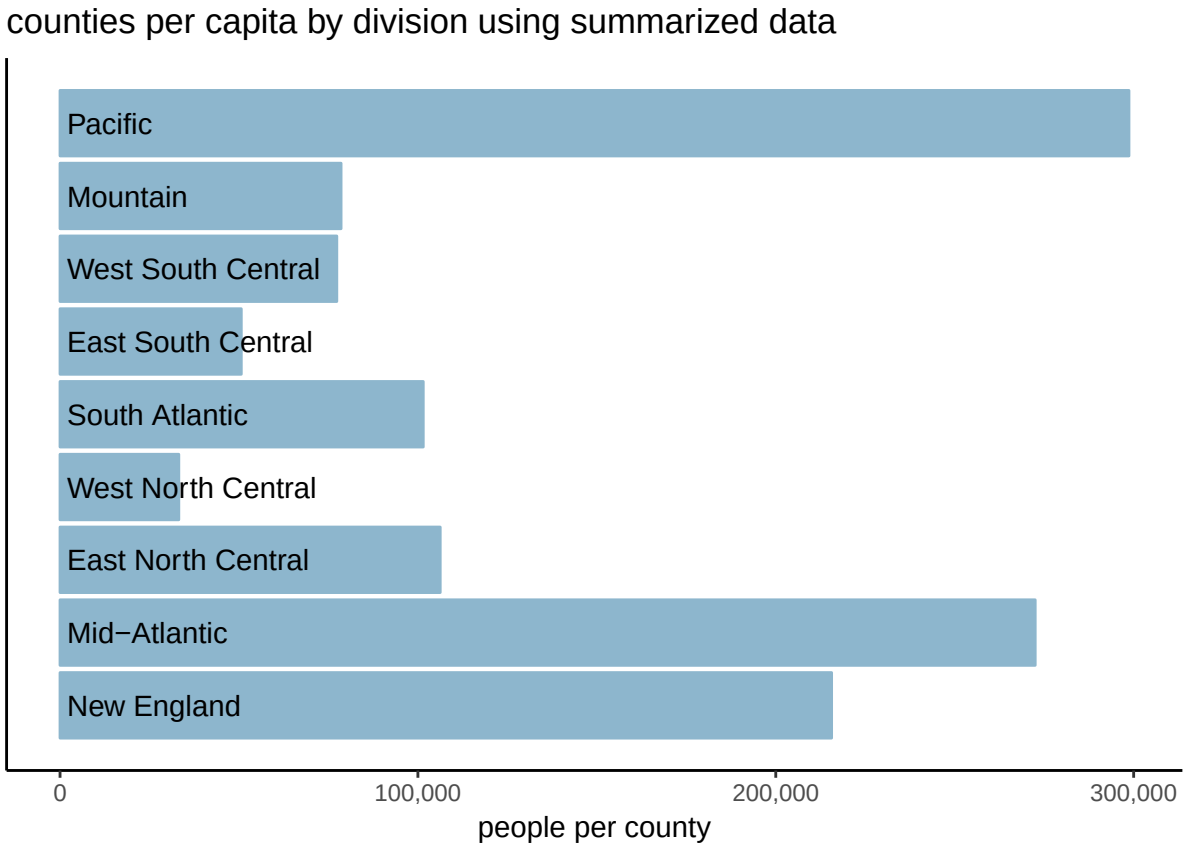
Not too bad, but we can get rid of a lot of extra formatting. First get rid of the y axis and other background features by adjusting the theme.

```
# get rid of the y axis, which is not helpful
ggplot(data = divisions.2010, aes(x = division, y=ppl.by.cnty)) +
  geom_bar(stat = "identity") +
  scale_y_continuous(labels = comma) +
  ggtitle("counties per capita by division using summarized data") +
  coord_flip() +
  labs(x="", y="people per county") +
  geom_text(mapping = aes(y=nada, label=div.name), hjust = 0) +
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
        panel.background = element_blank(), axis.line = element_line(colour = "black"),
        axis.text.y = element_blank(),
        axis.ticks.y=element_blank())
```



It's hard to see dark text on gray bars, so let's change the color of the bars. A full description of R colors is here.

```
# make bars more pleasant with text
ggplot(data = divisions.2010, aes(x = division, y=ppl.by.cnty)) +
  geom_bar(stat = "identity", color ="lightskyblue3", fill = "lightskyblue3") +
  scale_y_continuous(labels = comma) +
  ggtitle("counties per capita by division using summarized data") +
  coord_flip() +
  labs(x="", y="people per county") +
  geom_text(mapping = aes(y=nada, label=div.name), hjust = 0) +
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
        panel.background = element_blank(), axis.line = element_line(colour = "black"),
        axis.text.y = element_blank(),
        axis.ticks.y=element_blank())
```



Not too bad. Here are my remaining gripes

- should drop black vertical axis
- maybe need only ticks or vertical lines for horizontal axis

C. More variations on bars: paired and stacked

Now we'll use bar charts to show income distributions. The block group data from the census reports the number of households by income category in variable B19001e1-B19001e17 (look at the documentation to see what these mean).

1. Load and prep data

For purposes of learning, we'll aggregate the block group data to the state level. I then check that my three-observation dataframe looks ok.

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

# aggregate to state
state.demo <- ddply(block.groups, "STATE", summarize,
                    b19001e_2 = sum(B19001e2),
                    b19001e_3 = sum(B19001e3),
                    b19001e_4 = sum(B19001e4),
                    b19001e_5 = sum(B19001e5),
                    b19001e_6 = sum(B19001e6),
                    b19001e_7 = sum(B19001e7),
                    b19001e_8 = sum(B19001e8),
                    b19001e_9 = sum(B19001e9),
```

```
                    b19001e_10 = sum(B19001e10),
                    b19001e_11 = sum(B19001e11),
                    b19001e_12 = sum(B19001e12),
                    b19001e_13 = sum(B19001e13),
                    b19001e_14 = sum(B19001e14),
                    b19001e_15 = sum(B19001e15),
                    b19001e_16 = sum(B19001e16),
                    b19001e_17 = sum(B19001e17)
)
state.demo
```

```
##   STATE b19001e_2 b19001e_3 b19001e_4 b19001e_5 b19001e_6 b19001e_7
## 1    11     27304     11975      9726      9771      9231      9489
## 2    24    109214     70865     70989     75347     77904     80693
## 3    51    171152    126284    126786    128776    127852    132415
##   b19001e_8 b19001e_9 b19001e_10 b19001e_11 b19001e_12 b19001e_13
## 1      9651     10683       8606      16914      21497      29172
## 2     77491     84600      75707     161400     212685     292053
## 3    127293    132236     118297     232775     298797     392128
##   b19001e_14 b19001e_15 b19001e_16 b19001e_17
## 1      22457      14978      19794      29944
## 2     227291     160405     183781     178381
## 3     276323     185930     208976     220199
```

This summarizing worked, but I can't make a bar chart when the data are in separate columns. To fix this, we need a long dataset rather than a wide one. As we discussed in class today, we turn to the reshape command to do this. The R reshape command wants to call the new variable it creates `time` and since ours is not `time`, I rename it to `income.cat` for income categories.

```
# reshape to long
state.demo.l <- reshape(state.demo,
                        varying= c("b19001e_2","b19001e_3","b19001e_4","b19001e_5",
                                   "b19001e_6","b19001e_7","b19001e_8","b19001e_9",
                                   "b19001e_10","b19001e_11","b19001e_12","b19001e_13",
                                   "b19001e_14","b19001e_15","b19001e_16","b19001e_17"),
                        direction = "long",
                        idvar="STATE", sep="_")
# rename -time- b/c it isn't time
names(state.demo.l)[names(state.demo.l) == "time"] <- "income.cat"
names(state.demo.l)
```

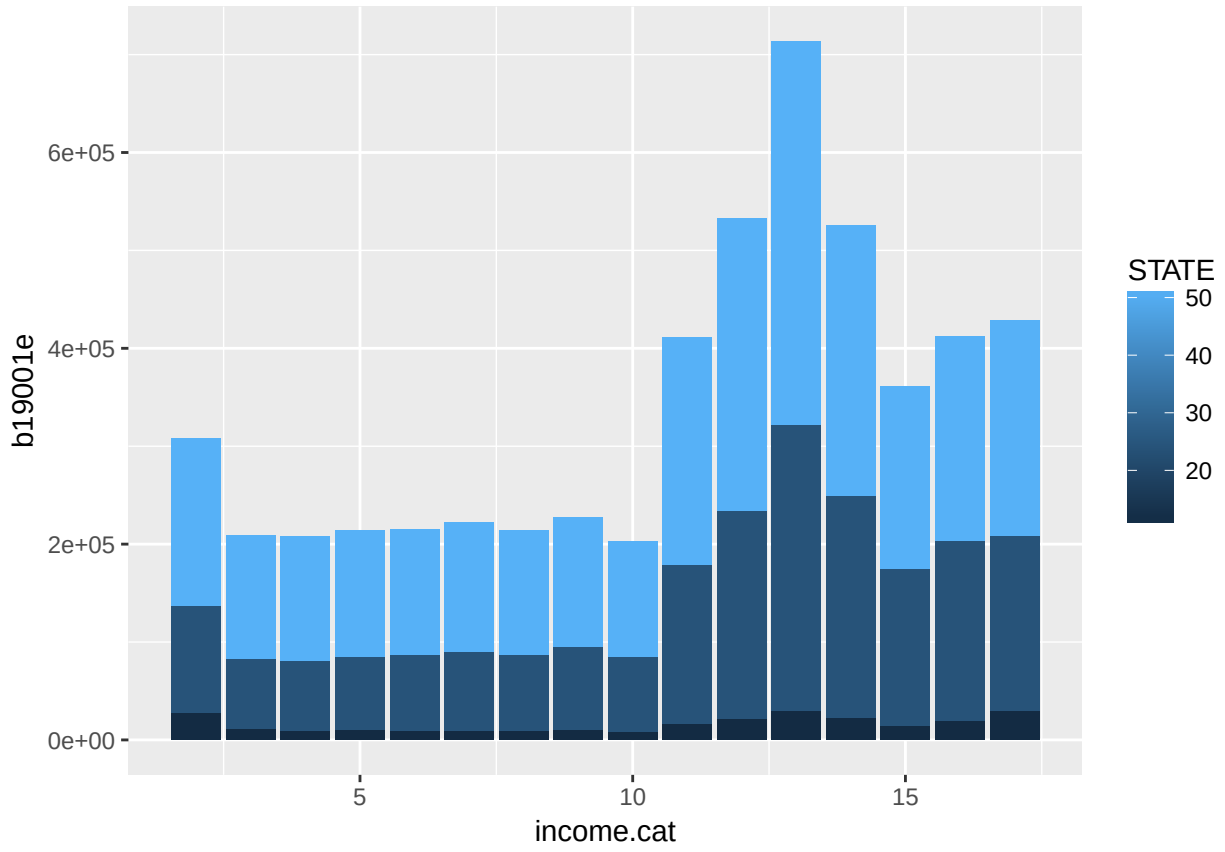
```
## [1] "STATE"      "income.cat" "b19001e"
```

2. Make charts

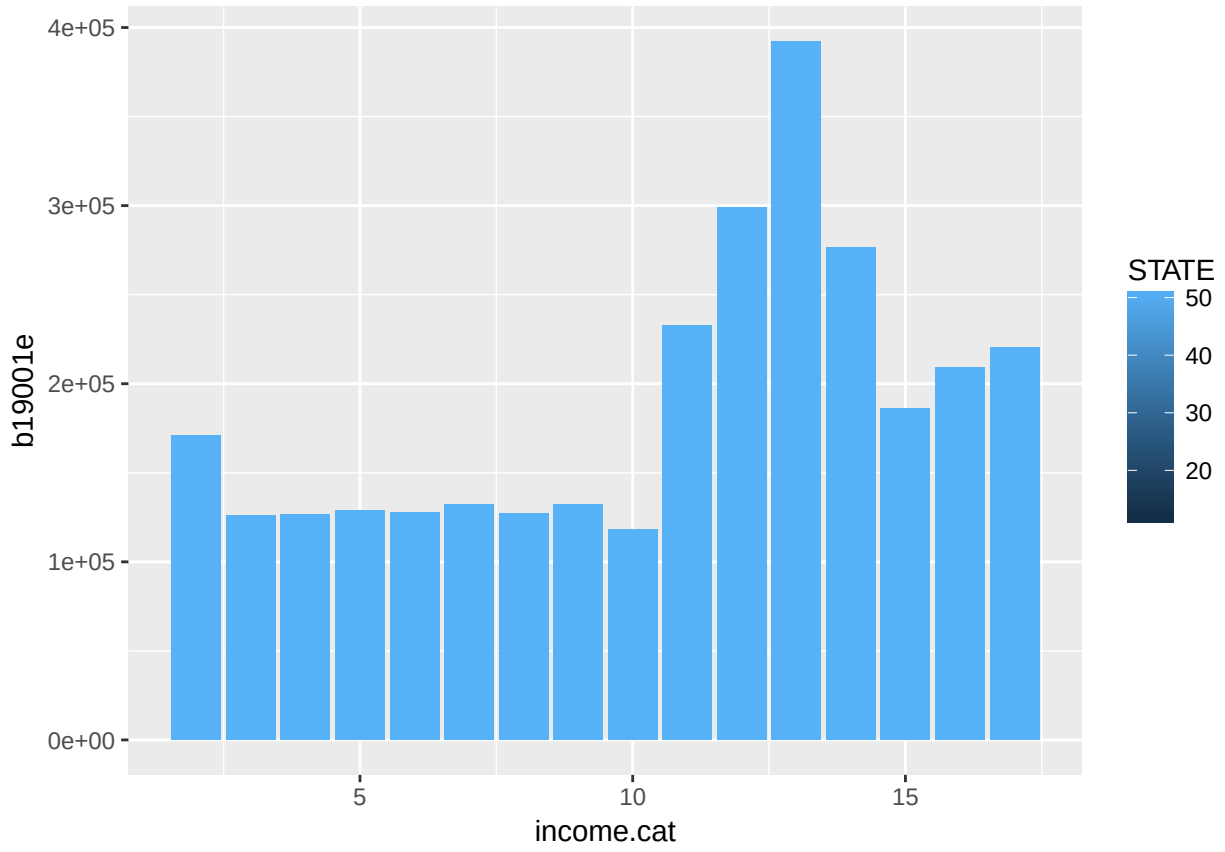
Fixed data in hand, we can now make charts.

I begin by making three bar charts that illustrate the types of graph possible; none of them are very helpful in illuminating our data.

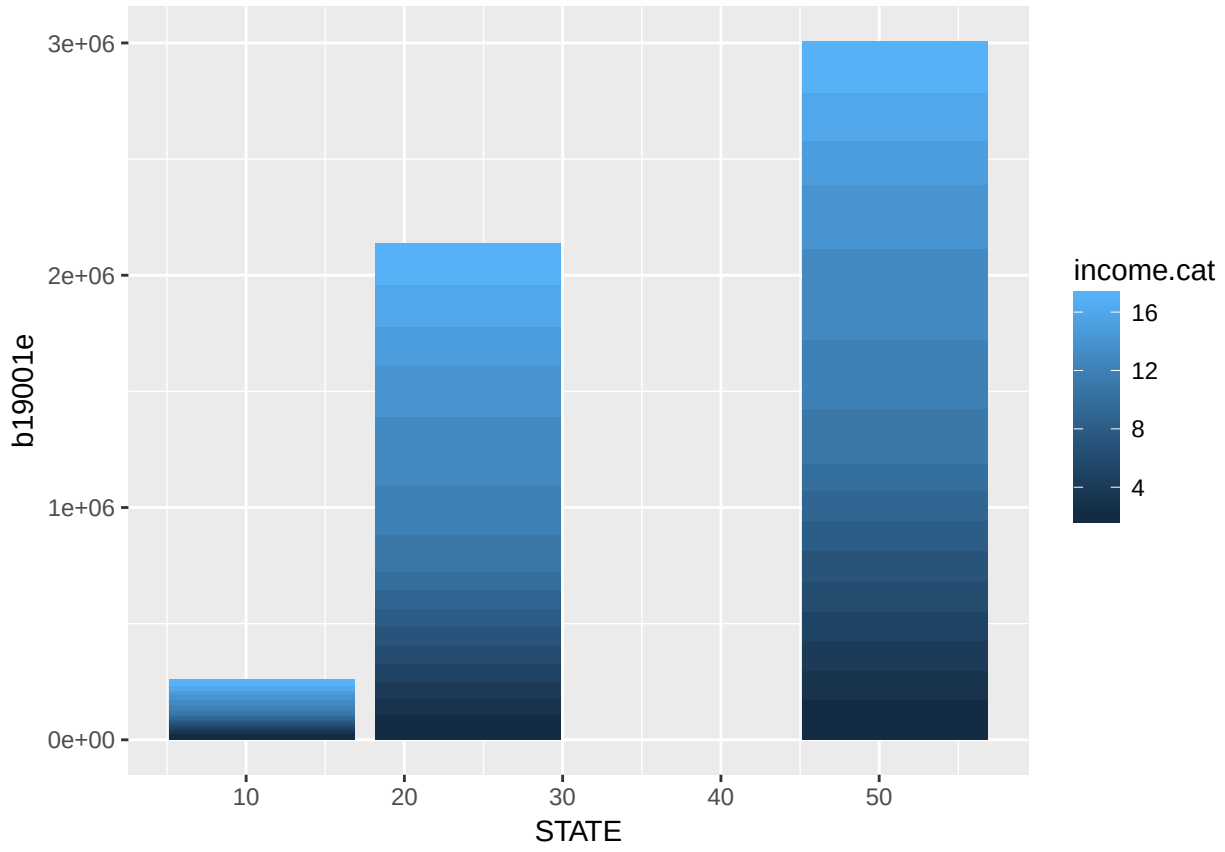
```
# stacked bars -- not too helpful
ggplot(state.demo.l, aes(fill=STATE, x=income.cat, y=b19001e)) +
  geom_bar(stat="identity")
```




```
# grouped bars -- not helpful
ggplot(state.demo.l, aes(fill=STATE, x=income.cat, y=b19001e)) +
  geom_bar(position="dodge", stat="identity")
```

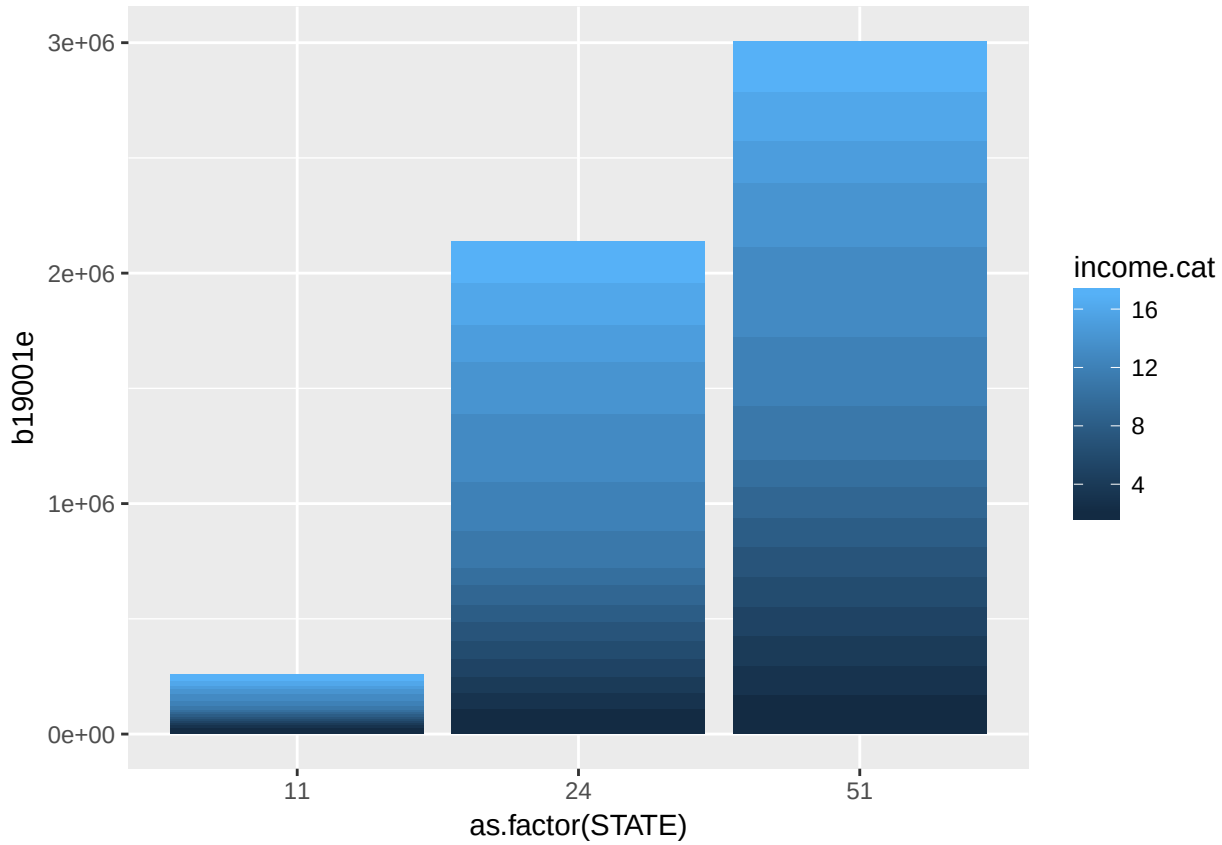



```
# bars by state
ggplot(state.demo.l, aes(fill=income.cat, x=STATE, y=b19001e)) +
  geom_bar(stat="identity")
```



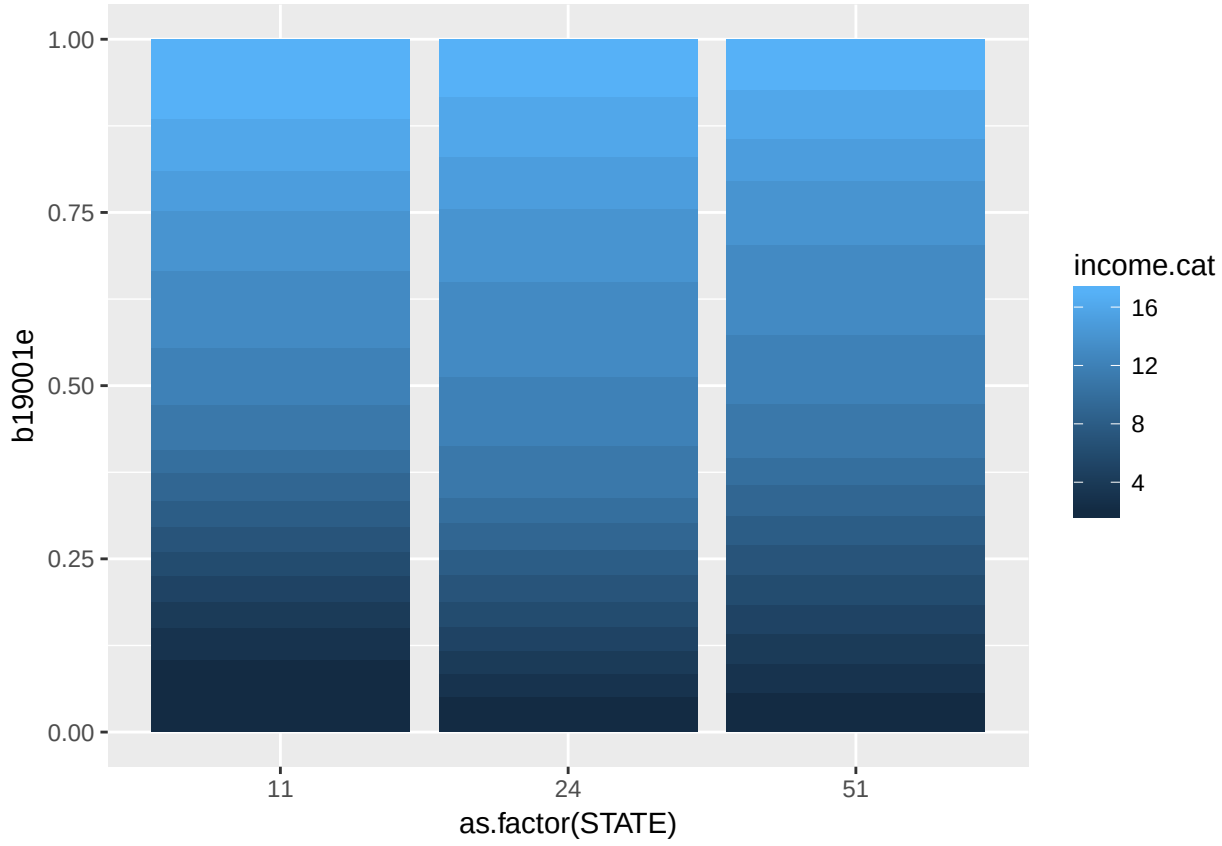
R thinks that `STATE` is some kind of continuous variable when in fact it is not. We can correct this by telling R that `STATE` is a factor using `as.factor` in the axis assignment.

```
# bars by state, but state is a factor
ggplot(state.demo.1, aes(fill=income.cat, x=as.factor(STATE), y=b19001e)) +
  geom_bar(stat="identity")
```



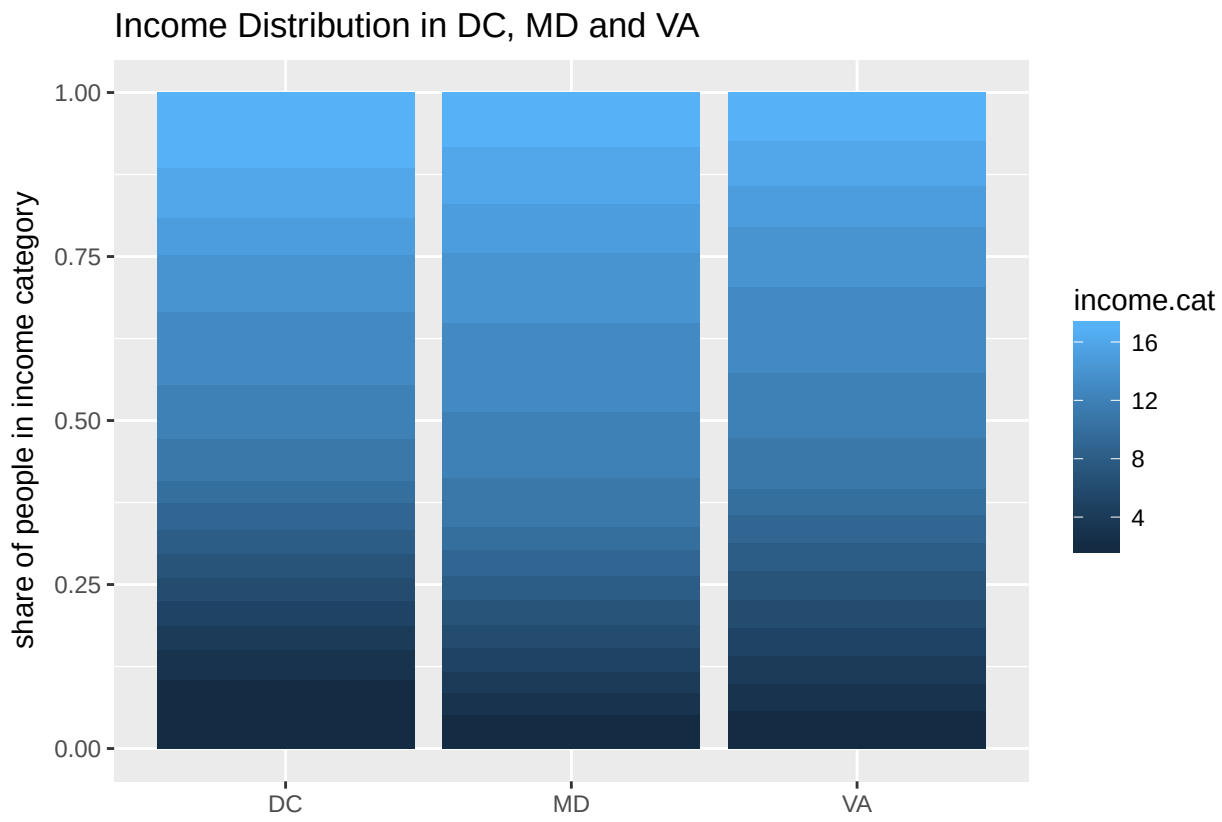
While raw numbers are sometimes useful, for cross-state comparisons shares are usually more helpful. We can get to shares by setting `position = "fill"` in the `geom_bar` command.

```
# percentage bars by state
ggplot(state.demo.l, aes(fill=income.cat, x=as.factor(STATE), y=b19001e)) +
  geom_bar(position="fill", stat="identity")
```



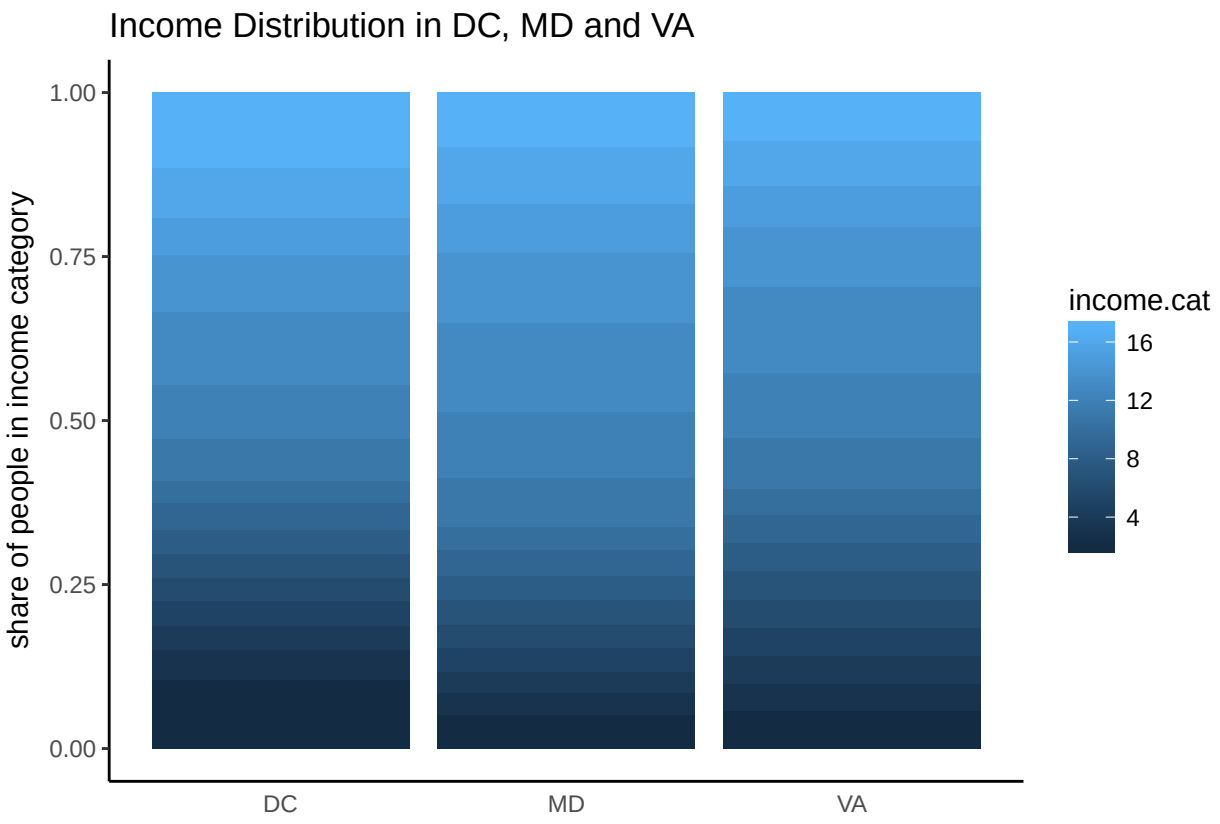
Make the chart more legible by adding labels and a title.

```
# add some labels, title
ggplot(state.demo.l, aes(fill=income.cat, x=as.factor(STATE), y=b19001e)) +
  geom_bar(position="fill", stat="identity") +
  labs(x="", y="share of people in income category") +
  scale_x_discrete(labels = c("DC","MD","VA")) +
  ggtitle("Income Distribution in DC, MD and VA")
```



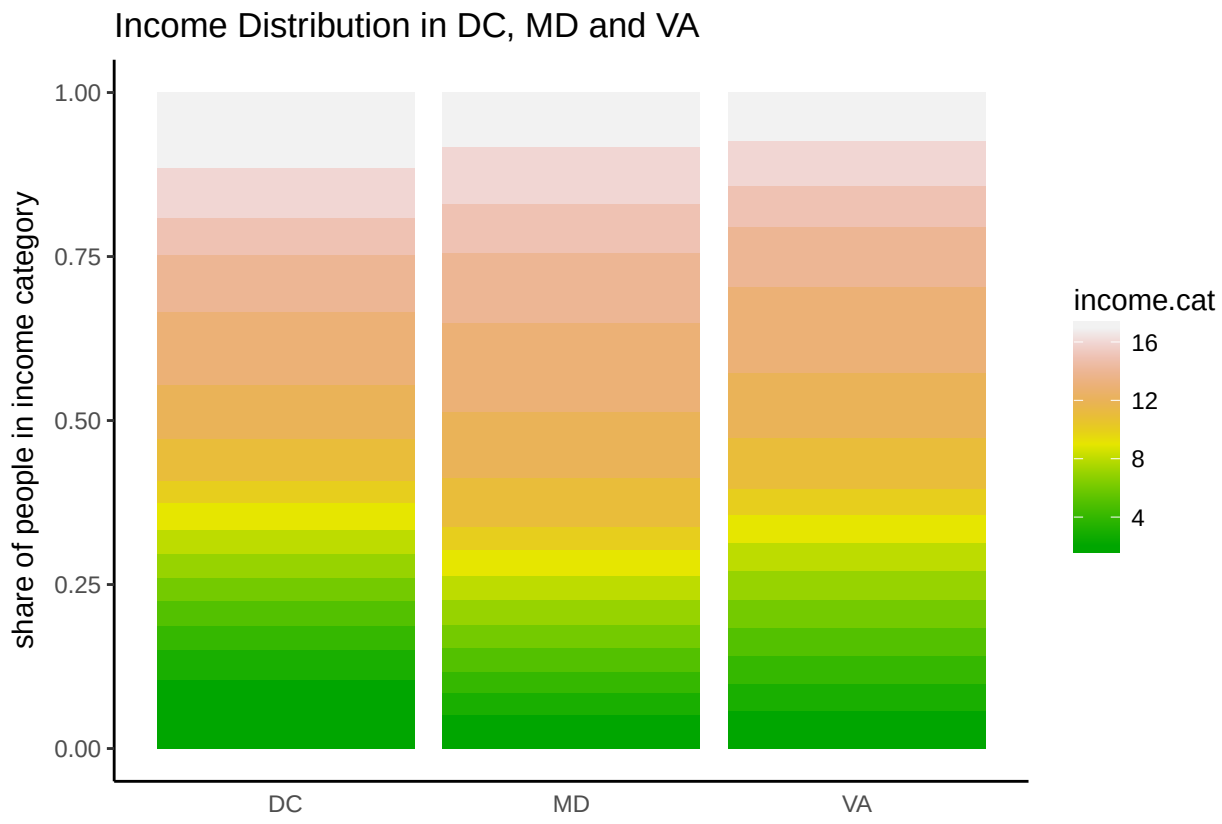
The axis ticks serve no useful role here, so I drop them, along with the axis ticks and grey background.

```
# get rid of grey background, x axis ticks
ggplot(state.demo.l, aes(fill=income.cat, x=as.factor(STATE), y=b19001e)) +
  geom_bar(position="fill", stat="identity") +
  labs(x="", y="share of people in income category") +
  scale_x_discrete(labels = c("DC","MD","VA")) +
  ggtitle("Income Distribution in DC, MD and VA") +
  theme(panel.grid.major = element_blank(),
      panel.grid.minor = element_blank(),
      panel.background = element_blank(),
      axis.line = element_line(colour = "black"),
      axis.ticks.x=element_blank())
```



To make this chart legible, we need to change the fill color so readers can discriminate between categories. Depending on the story I'd like to tell, I would seriously consider collapsing the number of categories, since 16 different colors are too hard to differentiate.

```
# change the gradient of the fill
ggplot(state.demo.l, aes(fill=income.cat, x=as.factor(STATE), y=b19001e)) +
  geom_bar(position="fill", stat="identity") +
  labs(x="", y="share of people in income category") +
  scale_x_discrete(labels = c("DC","MD","VA")) +
  ggtitle("Income Distribution in DC, MD and VA") +
  theme(panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.background = element_blank(),
        axis.line = element_line(colour = "black"),
        axis.ticks.x=element_blank())  +
  scale_fill_gradientn(colors = terrain.colors(16))
```



As is clear, the labels make no sense in this graph. To replace them, we create a new variable with the text we'd like to appear. (We did something similar in step B above.) The first steps assign values to the variable `state.demo.l$income.name` based on the condition in the square bracket.

We call this variable in the program, filling by `income.name` rather than `income.cat`.

We also label the states with the `scale_x_discrete()` command.

```
# check what reverse order looks like
test <- c(17:2)
test
```

```
##  [1] 17 16 15 14 13 12 11 10  9  8  7  6  5  4  3  2
```

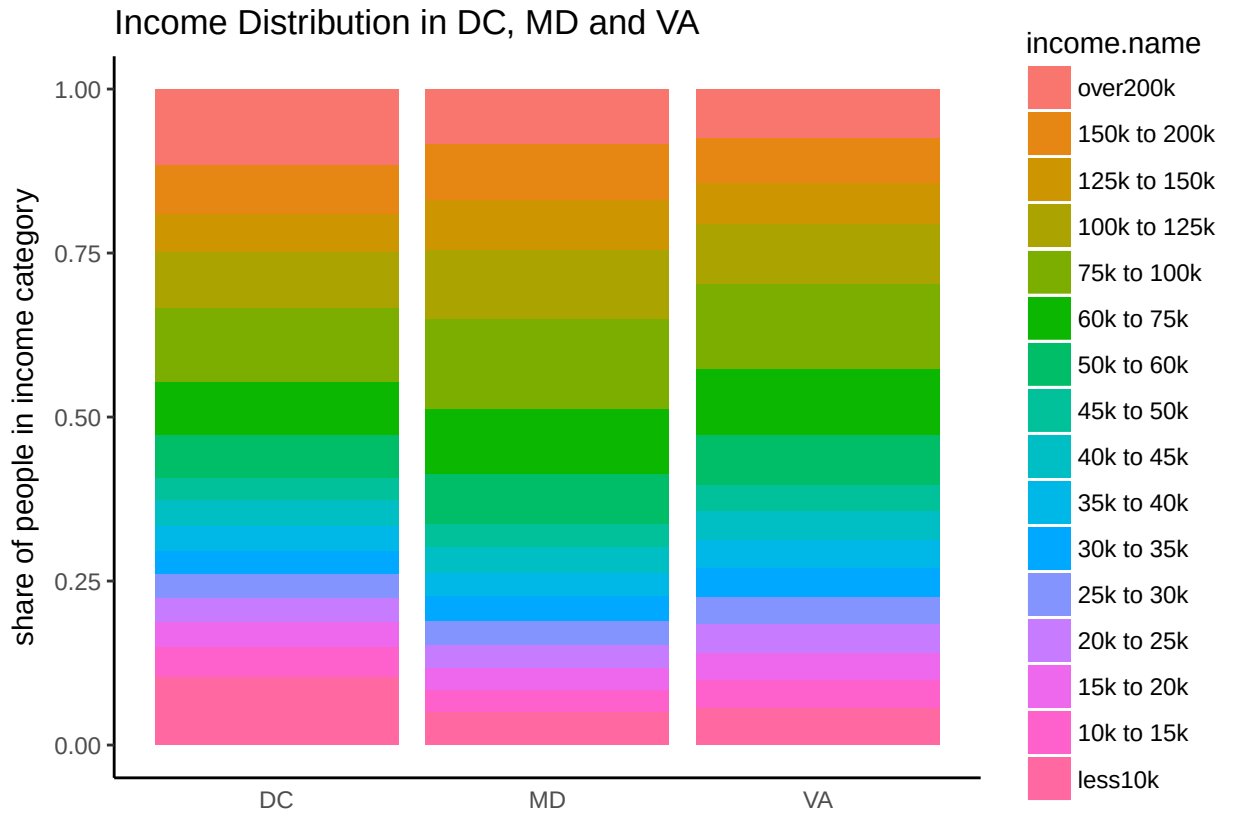
```
# add text so label makes sense
# Making a new factor variable with labels (Reverse order)
state.demo.l$income.name <- factor(state.demo.l$income.cat, levels=c(17:2),
                                   labels=c("over200k", "150k to 200k", "125k to 150k", "100k to 125k",
                                            "75k to 100k", "60k to 75k", "50k to 60k", "45k to 50k",
                                            "40k to 45k", "35k to 40k", "30k to 35k",  "25k to 30k",
                                            "20k to 25k", "15k to 20k", "10k to 15k", "less10k"))

levels(state.demo.l$income.name)
```

```
##  [1] "over200k"     "150k to 200k" "125k to 150k" "100k to 125k"
##  [5] "75k to 100k"  "60k to 75k"   "50k to 60k"   "45k to 50k"
##  [9] "40k to 45k"   "35k to 40k"   "30k to 35k"   "25k to 30k"
## [13] "20k to 25k"   "15k to 20k"   "10k to 15k"   "less10k"
```

```
table(state.demo.l$income.name)
```

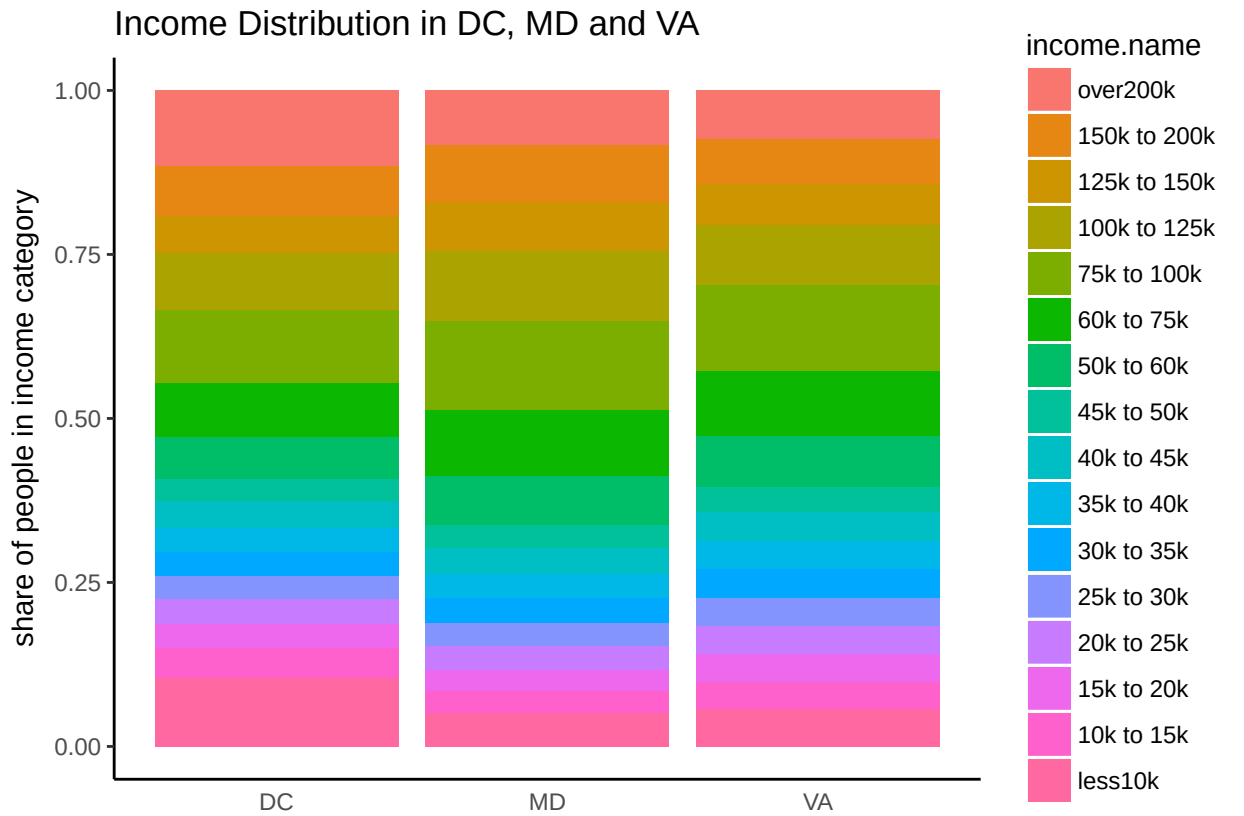
```
##
##     over200k 150k to 200k 125k to 150k 100k to 125k  75k to 100k
##            3            3            3            3            3
##   60k to 75k   50k to 60k   45k to 50k   40k to 45k   35k to 40k
##            3            3            3            3            3
##   30k to 35k   25k to 30k   20k to 25k   15k to 20k   10k to 15k
##            3            3            3            3            3
##      less10k
##            3
```

```
# now make the graph
ggplot(state.demo.l, aes(fill=income.name, x=as.factor(STATE), y=b19001e)) +
  geom_bar(position="fill", stat="identity") +
  labs(x="", y="share of people in income category") +
  ggtitle("Income Distribution in DC, MD and VA") +
  theme(panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.background = element_blank(),
        axis.line = element_line(colour = "black"),
        axis.ticks.x=element_blank())  +
  scale_x_discrete(labels = c("DC","MD","VA"))
```



This sort of works, but the income variable is not sorted, and is therefore very different to follow. Colors are not always ranked by size. To fix this, I tell R that `income.name` is a factor with the levels as listed below (I do big to small so that the highest income gets listed at the top of the graph).

```
## set the levels in order we want
#state.demo.l <- within(state.demo.l,
#                    income.name <- factor(income.name,
#                                          levels=c("gt 200","150 to 200","125 to 150",
#                                                   "100 to 125","75 to 100","60 to 75",
#                                                   "50 to 60","45 to 50","40 to 45",
#                                                   "35 to 40","30 to 35","25 to 30",
#                                                   "20 to 25","15 to 20","10 to 15",
#                                                  "lt 10")))

ggplot(state.demo.l,
       aes(fill=income.name, x=as.factor(STATE), y=b19001e)) +
  geom_bar(position="fill", stat="identity") +
  labs(x="", y="share of people in income category") +
  ggtitle("Income Distribution in DC, MD and VA") +
  theme(panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.background = element_blank(),
        axis.line = element_line(colour = "black"),
        axis.ticks.x=element_blank())  +
  scale_x_discrete(labels = c("DC","MD","VA"))
```



D. Homework

1. Make a bar chart with just block groups of one county, using horizontal bars to show the income distribution. Feel free to limit the number of categories if you think that would be helpful.

2. Make a bar chart of your choice based on another dataset.