More on ggplot

Md Johirul Islam

2/28/2022

Grouped data and the group aesthetic

First, we need to load libraries

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.5     v dplyr   1.0.7
## v tidyr   1.1.4     v stringr 1.4.0
## v readr   2.0.2     v forcats 0.5.1
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(socviz)
```

```
## Warning: package 'socviz' was built under R version 4.1.2
```

```
library(gapminder)
```

```
## Warning: package 'gapminder' was built under R version 4.1.2
```

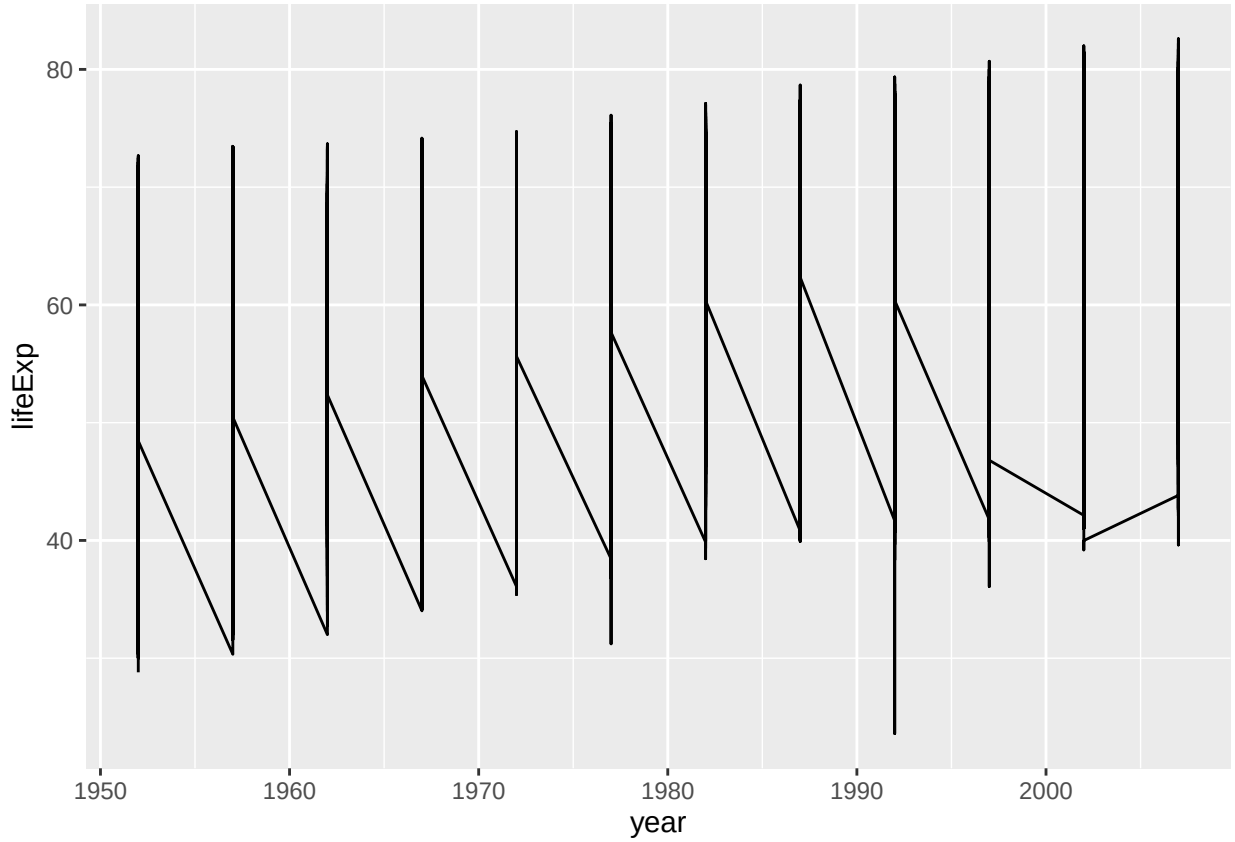
```
#load the data
gapminder
```

```
## # A tibble: 1,704 x 6
##    country     continent  year lifeExp      pop gdpPercap
##    <fct>       <fct>     <int>   <dbl>    <int>     <dbl>
##  1 Afghanistan Asia       1952    28.8  8425333      779.
##  2 Afghanistan Asia       1957    30.3  9240934      821.
##  3 Afghanistan Asia       1962    32.0 10267083      853.
##  4 Afghanistan Asia       1967    34.0 11537966      836.
##  5 Afghanistan Asia       1972    36.1 13079460      740.
##  6 Afghanistan Asia       1977    38.4 14880372      786.
##  7 Afghanistan Asia       1982    39.9 12881816      978.
##  8 Afghanistan Asia       1987    40.8 13867957      852.
##  9 Afghanistan Asia       1992    41.7 16317921      649.
## 10 Afghanistan Asia       1997    41.8 22227415      635.
## # ... with 1,694 more rows
```

We can draw geom_line to show the trajectory of life expectancy.

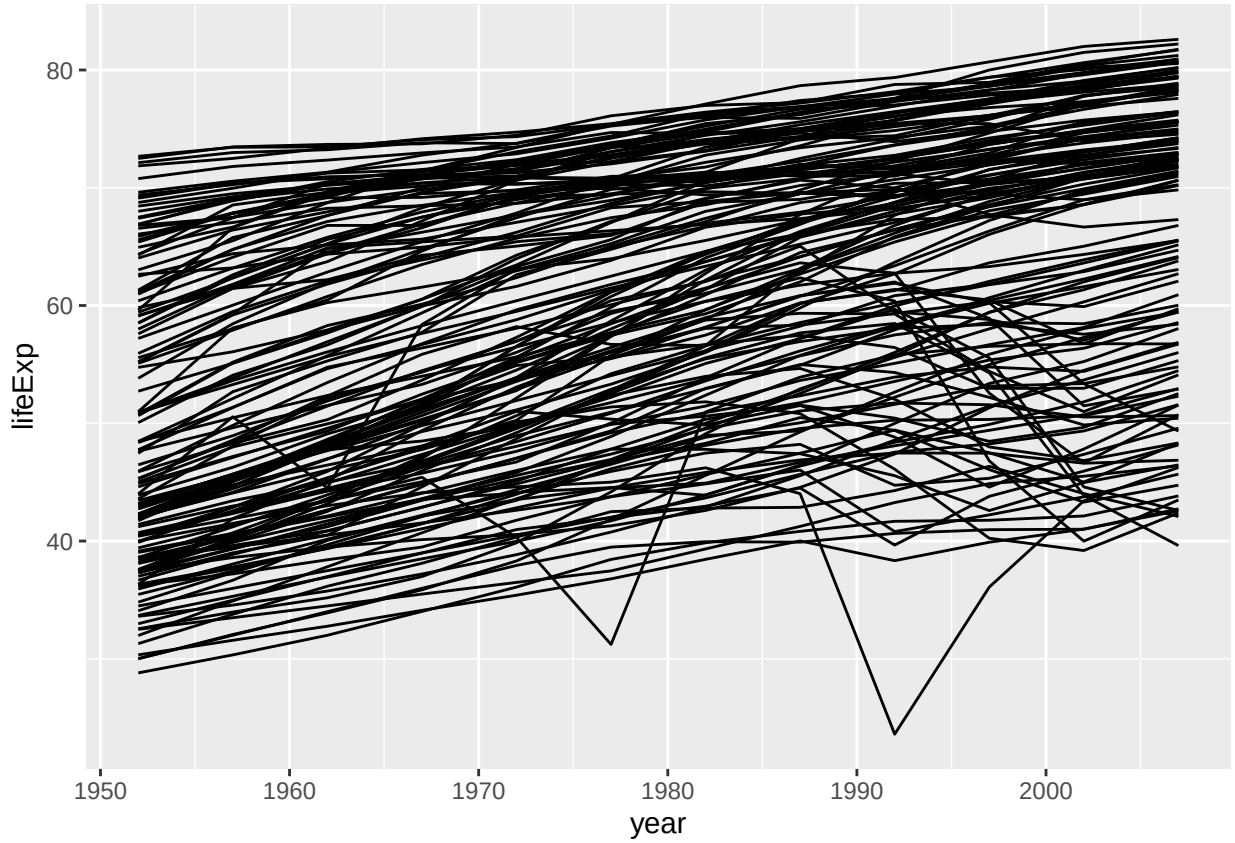
```
p<- ggplot(data = gapminder,
           mapping = aes(x = year,
```

```
                          y = lifeExp))
p+ geom_line()
```



Something bizarre happens. Here, we can use the group aesthetic to tell ggplot explicitly about this country-level structure.

```
p <- ggplot(data = gapminder,
            mapping = aes(x = year,
                          y = lifeExp))
p + geom_line(aes(group = country))
```

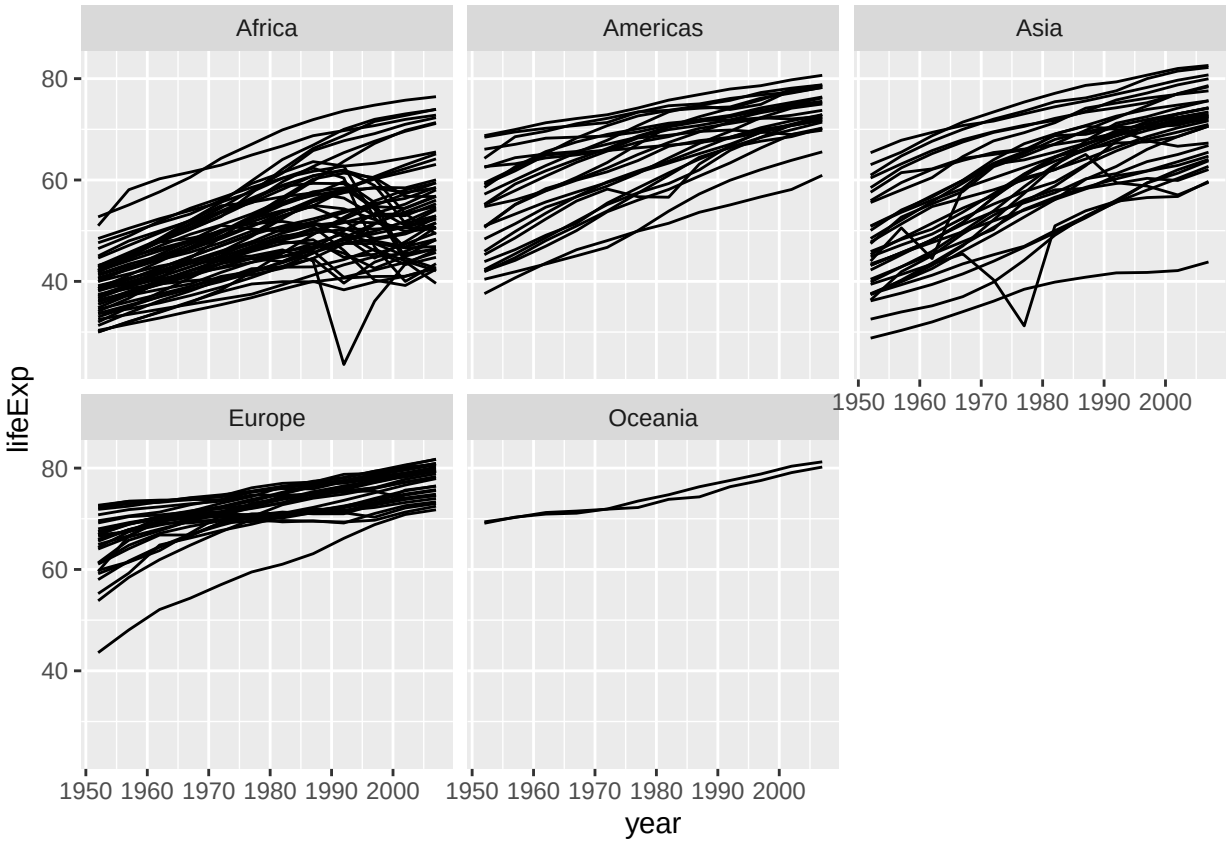


The group aesthetic is usually only needed when the grouping information you need to tell ggplot about is not built-in to the variables being mapped. For example, when we were plotting the points by continent, mapping color to continent was enough to get the right answer, because continent is already a categorical variable, so the grouping is clear. When mapping the x to year, however, there is no information in the year variable itself to let ggplot know that it is grouped by country for the purposes of drawing lines with it. So we need to say that explicitly.

Facet to make small multiples

We will use facet_wrap() to split our plot by a categorical variable since the plot looks messy. In this case, we split our plot by continent.

```
p<- ggplot(data = gapminder,
           mapping = aes(x = year,
                         y = lifeExp))
p + geom_line(aes(group = country)) + facet_wrap(~ continent)
```

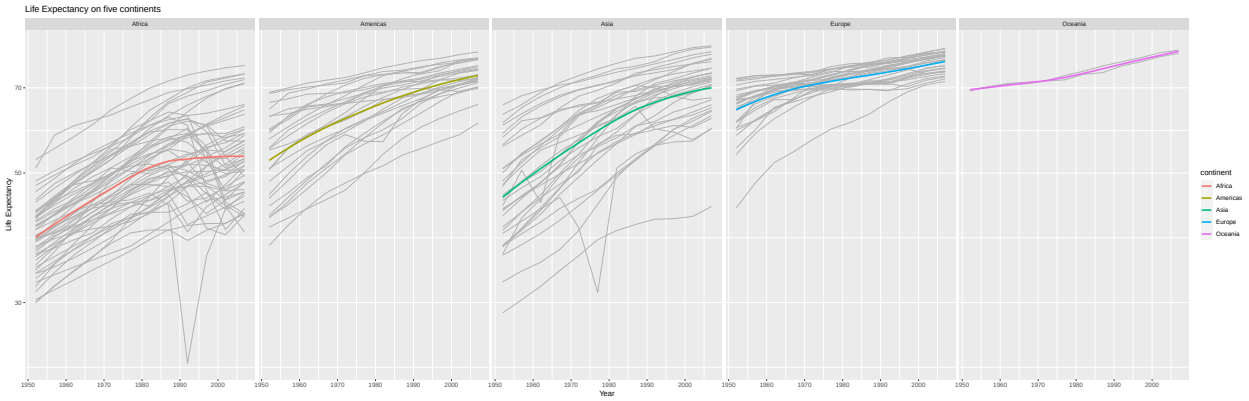


We can also use the ncol argument to facet_wrap() to control the number of columns used to lay out the facets. Because we have only five continents it might be worth seeing if we can fit them on a single row (which means we will have five columns)

So, full figure is as follows:

```
p <- ggplot(data = gapminder,
            mapping = aes(x = year,
                          y = lifeExp,
                          color = continent,
                          fill = continent))
p+ geom_line(aes(group = country), color = "grey70") +
  geom_smooth(method = "loess", size = 1.1, se = F) +
  scale_y_log10() +
  facet_wrap(~ continent, ncol = 5) +
  labs(x = "Year",
       y = "Life Expectancy",
       title = "Life Expectancy on five continents")
```

```
## `geom_smooth()` using formula 'y ~ x'
```



Facets can be more complex than this. For instance, you might want to cross-classify some data by two categorical variables. In that case you should try facet_grid() instead.

Faceting on two categorical variables. Here, we will use gss_sm dataset.

```
gss_sm
```

```
## # A tibble: 2,867 x 32
##     year    id ballot       age childs sibs   degree race  sex   region income16
##    <dbl> <dbl> <labelled> <dbl>  <dbl> <labe> <fct>  <fct> <fct> <fct>  <fct>
##  1  2016     1 1             47      3 2      Bache~ White Male  New E~ $170000~
##  2  2016     2 2             61      0 3      High ~ White Male  New E~ $50000 ~
##  3  2016     3 3             72      2 3      Bache~ White Male  New E~ $75000 ~
##  4  2016     4 1             43      4 3      High ~ White Fema~ New E~ $170000~
##  5  2016     5 3             55      2 2      Gradu~ White Fema~ New E~ $170000~
##  6  2016     6 2             53      2 2      Junio~ White Fema~ New E~ $60000 ~
##  7  2016     7 1             50      2 2      High ~ White Male  New E~ $170000~
##  8  2016     8 3             23      3 6      High ~ Other Fema~ Middl~ $30000 ~
##  9  2016     9 1             45      3 5      High ~ Black Male  Middl~ $60000 ~
## 10  2016    10 3             71      4 1      Junio~ White Male  Middl~ $60000 ~
## # ... with 2,857 more rows, and 21 more variables: relig <fct>, marital <fct>,
## #   padeg <fct>, madeg <fct>, partyid <fct>, polviews <fct>, happy <fct>,
## #   partners <fct>, grass <fct>, zodiac <fct>, pres12 <labelled>,
## #   wtssall <dbl>, income_rc <fct>, agegrp <fct>, ageq <fct>, siblings <fct>,
## #   kids <fct>, religion <fct>, bigregion <fct>, partners_rc <fct>, obama <dbl>
```

```
#get a glimpse of the data
glimpse(gss_sm)
```

```
## Rows: 2,867
## Columns: 32
## $ year        <dbl> 2016, 2016, 2016, 2016, 2016, 2016, 2016, 2016, 2016, 2016~
## $ id          <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17,~
## $ ballot      <labelled> 1, 2, 3, 1, 3, 2, 1, 3, 1, 3, 2, 1, 2, 3, 2, 3, 3, 2,~
## $ age         <dbl> 47, 61, 72, 43, 55, 53, 50, 23, 45, 71, 33, 86, 32, 60, 76~
## $ childs      <dbl> 3, 0, 2, 4, 2, 2, 2, 3, 3, 4, 5, 4, 3, 5, 7, 2, 6, 5, 0, 2~
## $ sibs        <labelled> 2, 3, 3, 3, 2, 2, 2, 6, 5, 1, 4, 4, 3, 6, 0, 1, 3, 8,~
## $ degree      <fct> Bachelor, High School, Bachelor, High School, Graduate, Ju~
## $ race        <fct> White, White, White, White, White, White, White, Other, Bl~
## $ sex         <fct> Male, Male, Male, Female, Female, Female, Male, Female, Ma~
## $ region      <fct> New England, New England, New England, New England, New En~
## $ income16    <fct> $170000 or over, $50000 to 59999, $75000 to $89999, $17000~
## $ relig       <fct> None, None, Catholic, Catholic, None, None, None, Catholic~
```

```
## $ marital     <fct> Married, Never Married, Married, Married, Married, Married~
## $ padeg       <fct> Graduate, Lt High School, High School, NA, Bachelor, NA, H~
## $ madeg       <fct> High School, High School, Lt High School, High School, Hig~
## $ partyid     <fct> "Independent", "Ind,near Dem", "Not Str Republican", "Not ~
## $ polviews    <fct> Moderate, Liberal, Conservative, Moderate, Slightly Libera~
## $ happy       <fct> Pretty Happy, Pretty Happy, Very Happy, Pretty Happy, Very~
## $ partners    <fct> NA, "1 Partner", "1 Partner", NA, "1 Partner", "1 Partner"~
## $ grass       <fct> NA, Legal, Not Legal, NA, Legal, Legal, NA, Not Legal, NA,~
## $ zodiac      <fct> Aquarius, Scorpio, Pisces, Cancer, Scorpio, Scorpio, Capri~
## $ pres12      <labelled> 3, 1, 2, 2, 1, 1, NA, NA, NA, 2, NA, NA, 1, 1, 2, 1, ~
## $ wtssall     <dbl> 0.9569935, 0.4784968, 0.9569935, 1.9139870, 1.4354903, 0.9~
## $ income_rc   <fct> Gt $170000, Gt $50000, Gt $75000, Gt $170000, Gt $170000, ~
## $ agegrp      <fct> Age 45-55, Age 55-65, Age 65+, Age 35-45, Age 45-55, Age 4~
## $ ageq        <fct> Age 34-49, Age 49-62, Age 62+, Age 34-49, Age 49-62, Age 4~
## $ siblings    <fct> 2, 3, 3, 3, 2, 2, 2, 6+, 5, 1, 4, 4, 3, 6+, 0, 1, 3, 6+, 2~
## $ kids        <fct> 3, 0, 2, 4+, 2, 2, 2, 3, 3, 4+, 4+, 4+, 3, 4+, 4+, 2, 4+, ~
## $ religion    <fct> None, None, Catholic, Catholic, None, None, None, Catholic~
## $ bigregion   <fct> Northeast, Northeast, Northeast, Northeast, Northeast, Nor~
## $ partners_rc <fct> NA, 1, 1, NA, 1, 1, NA, 1, NA, 3, 1, NA, 1, NA, 0, 1, 0, N~
## $ obama       <dbl> 0, 1, 0, 0, 1, 1, NA, NA, NA, 0, NA, NA, 1, 1, 0, 1, 0, 1,~
```

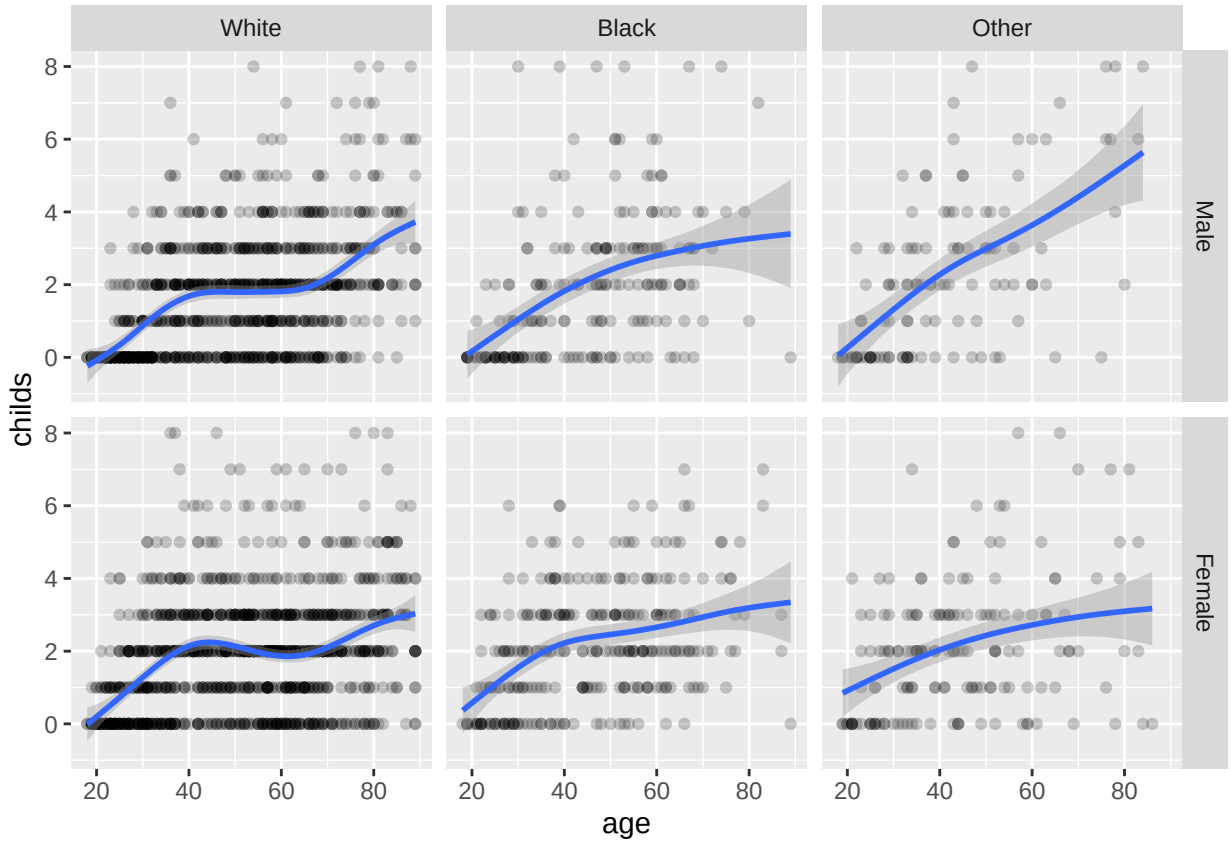
We use R's formula notation in the facet_grid function to facet sex and race. This time, because we are cross-classifying our results, the formula is two-sided: facet_grid(sex ~ race).

```
p <- ggplot(data = gss_sm,
            mapping = aes(x = age, y = childs))
p + geom_point(alpha = 0.2) +
    geom_smooth() +
    facet_grid(sex ~ race)
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```

```
## Warning: Removed 18 rows containing non-finite values (stat_smooth).
```

```
## Warning: Removed 18 rows containing missing values (geom_point).
```



We are not limited to two-way comparison. Further categorical variables can be added to the formula, too, (e.g. sex ~ race + degree) for more complex multi-way plots

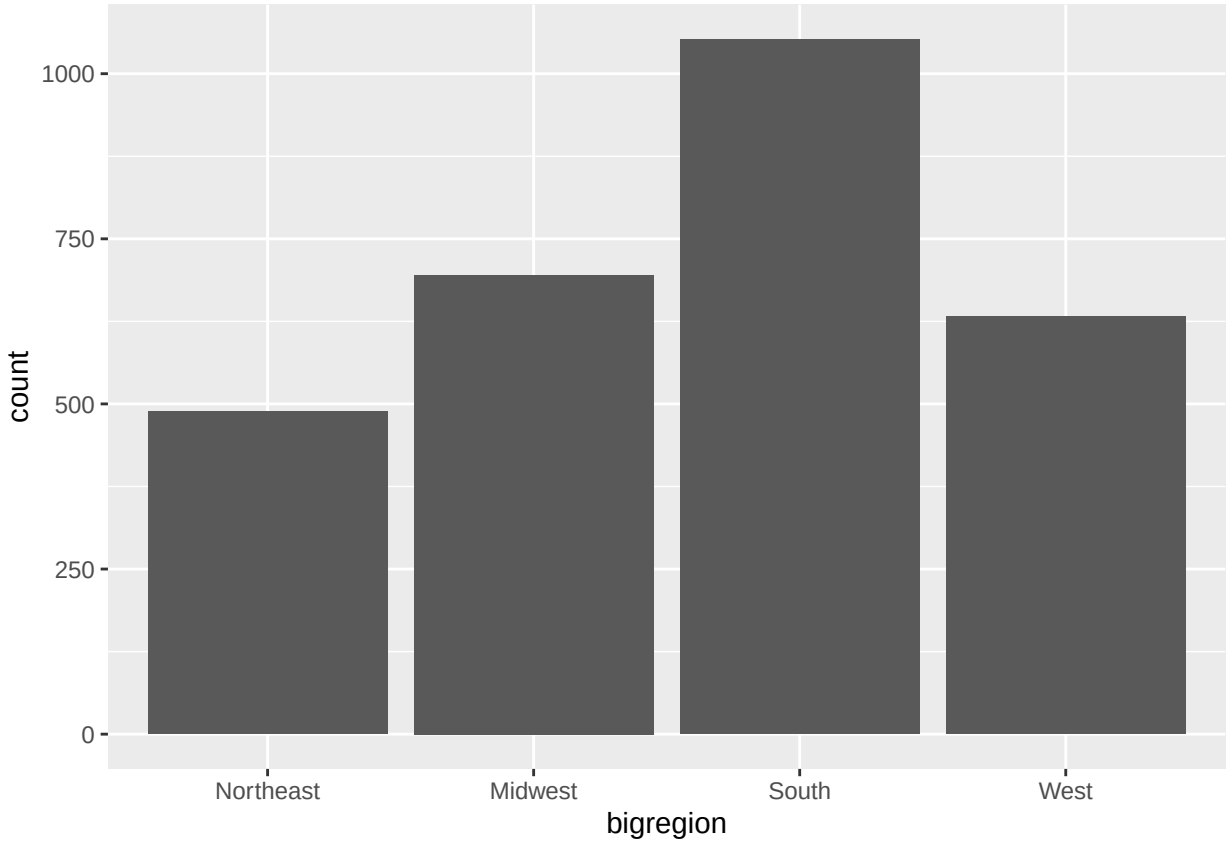
#Geoms can transform data#

To transfrom data we use stat_smooth fn.

A bar chart. We plot bar chart by geom_bar fn.

```
p <- ggplot(data = gss_sm,
            mapping = aes(x = bigregion))

p + geom_bar()
```

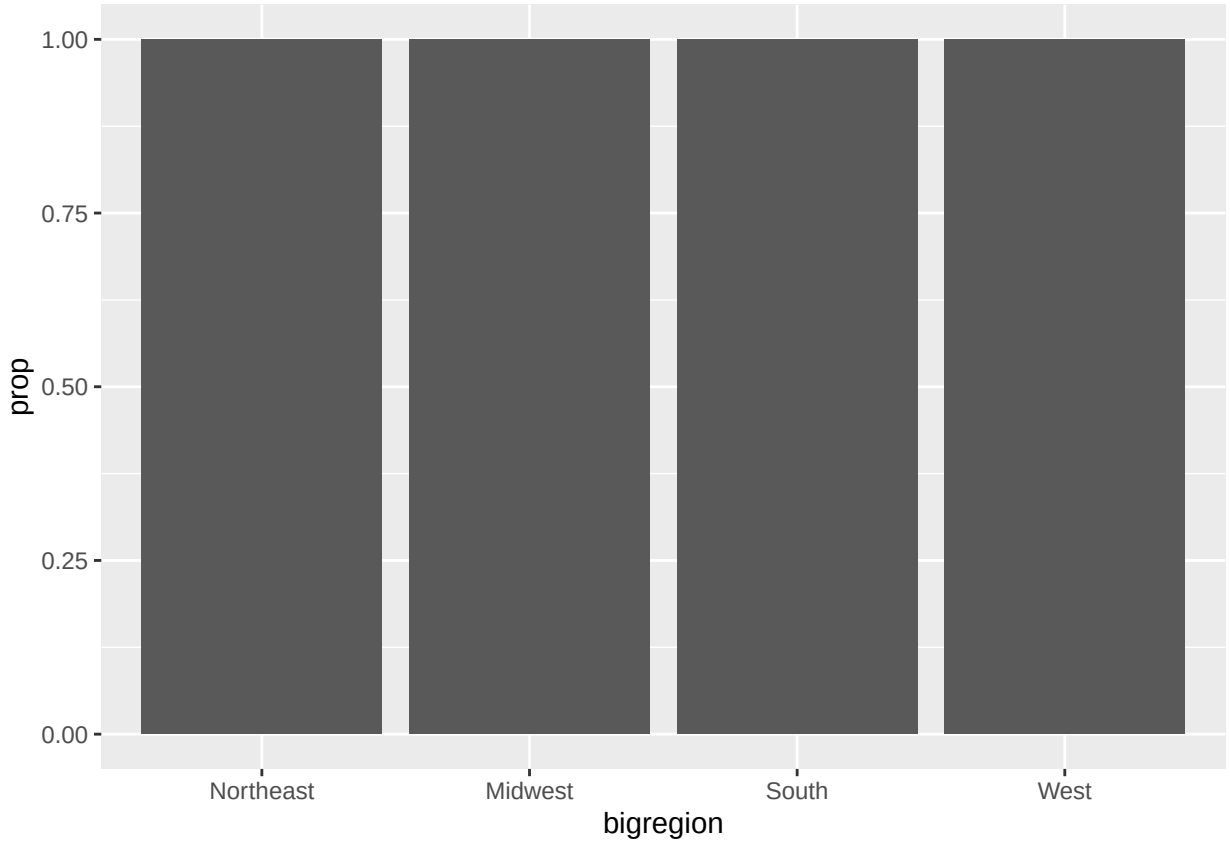


This bar chart gives the count on the vertical axis. Suppose, we want frequency rather than count.

If we want a chart of relative frequencies rather than counts, we will need to get the prop statistic instead. When ggplot calculates the count or the proportion, it returns temporary variables that we can use as mappings in our plots. The relevant statistic is called ..prop.. rather than prop. To make sure these temporary variables won't be confused with others we are working with, their names begin and end with two periods. (This is because we might already have a variable called count or prop in our dataset.) So our calls to it from the aes() function will generically look like this: = <..statistic..>. In this case, we want y to use the calculated proportion, so we say aes(y = ..prop..).

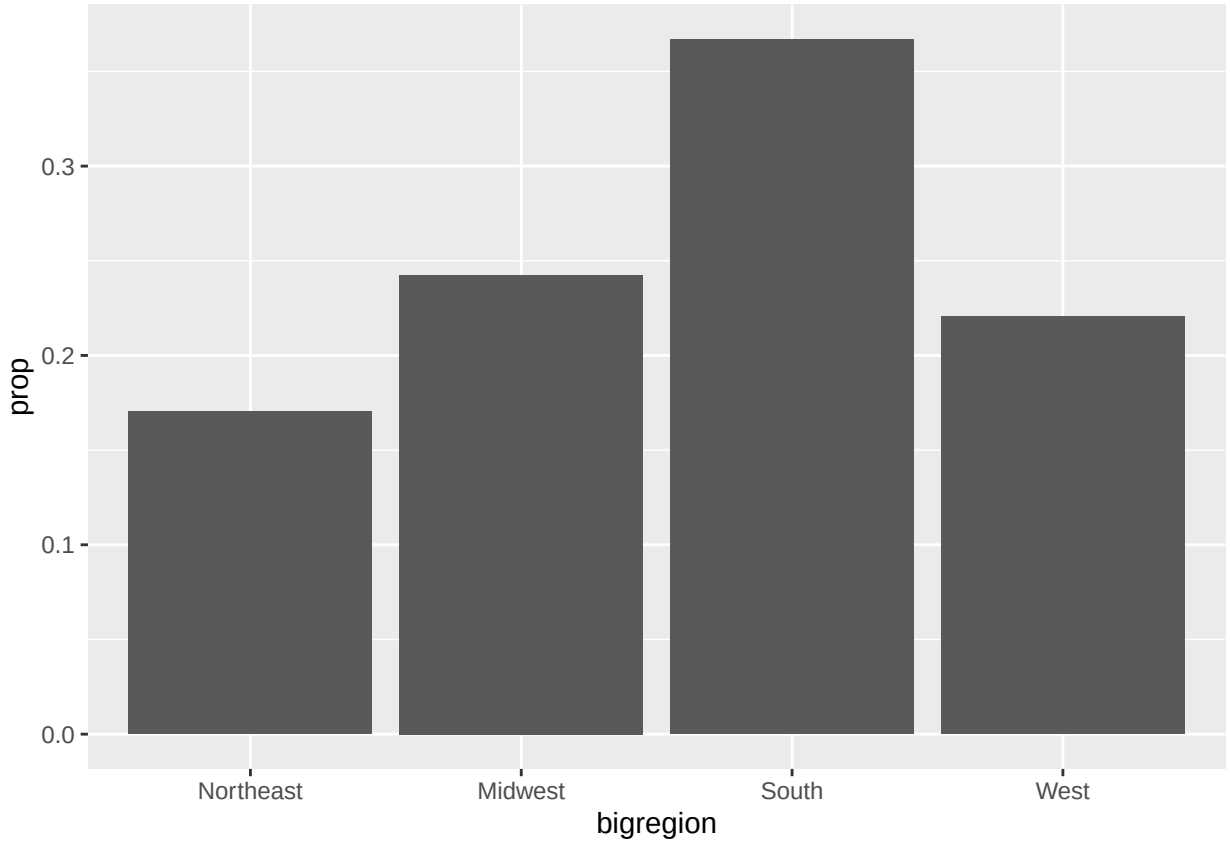
```
p <- ggplot(data = gss_sm,
            mapping = aes(x = bigregion))

p + geom_bar(mapping = aes(y = ..prop..))
```



The resulting plot is still not right. We no longer have a count on the y-axis, but the proportions of the bars all have a value of 1, so all the bars are the same height. This is a grouping issue again. In this case, we need to tell ggplot to ignore the x-categories when calculating denominator of the proportion, and use the total number observations instead. To do so we specify group = 1 inside the aes() call. The value of 1 is just a kind of "dummy group" that tells ggplot to use the whole dataset when establishing the denominator for its prop calculations.

```
p <- ggplot(data = gss_sm,
            mapping = aes(x = bigregion))
p + geom_bar(mapping = aes( y = ..prop.., group = 1))
```



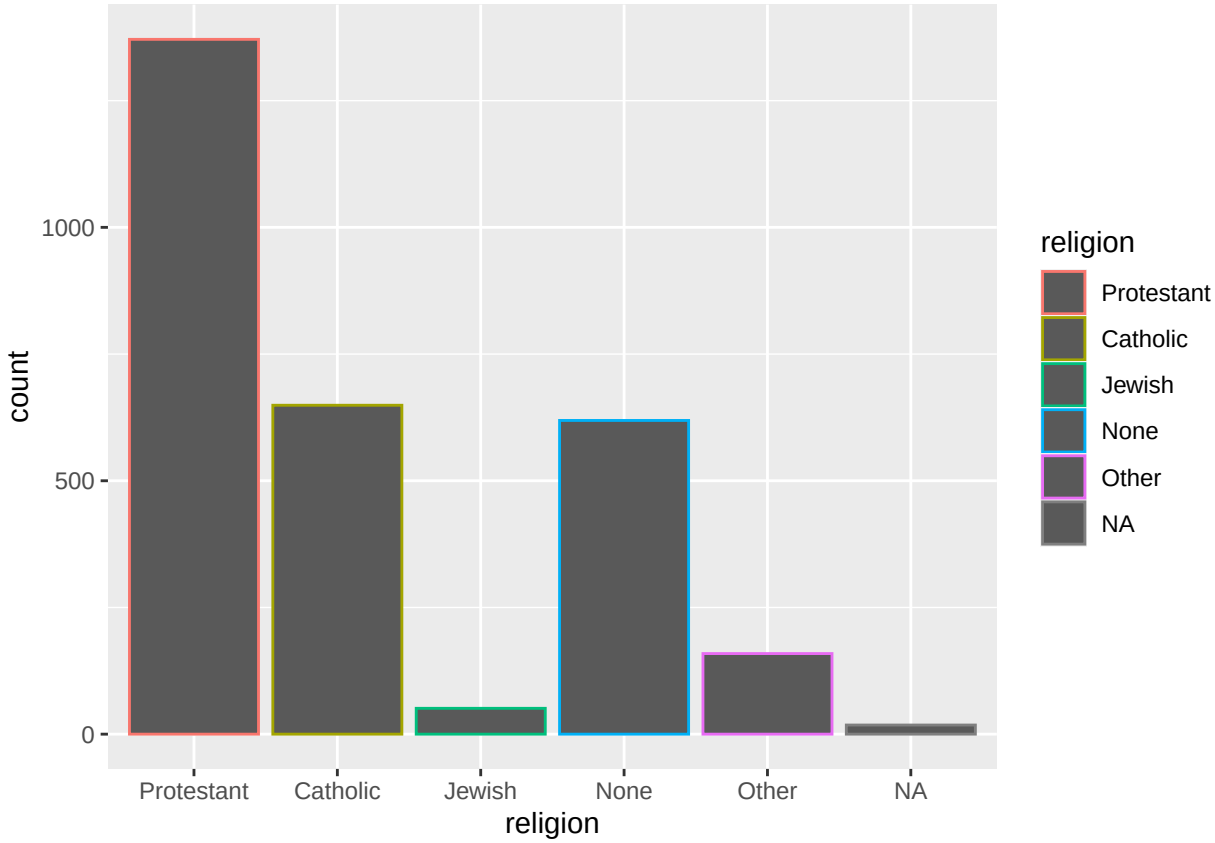
making table from a categorical variable

```
table(gss_sm$religion)
```

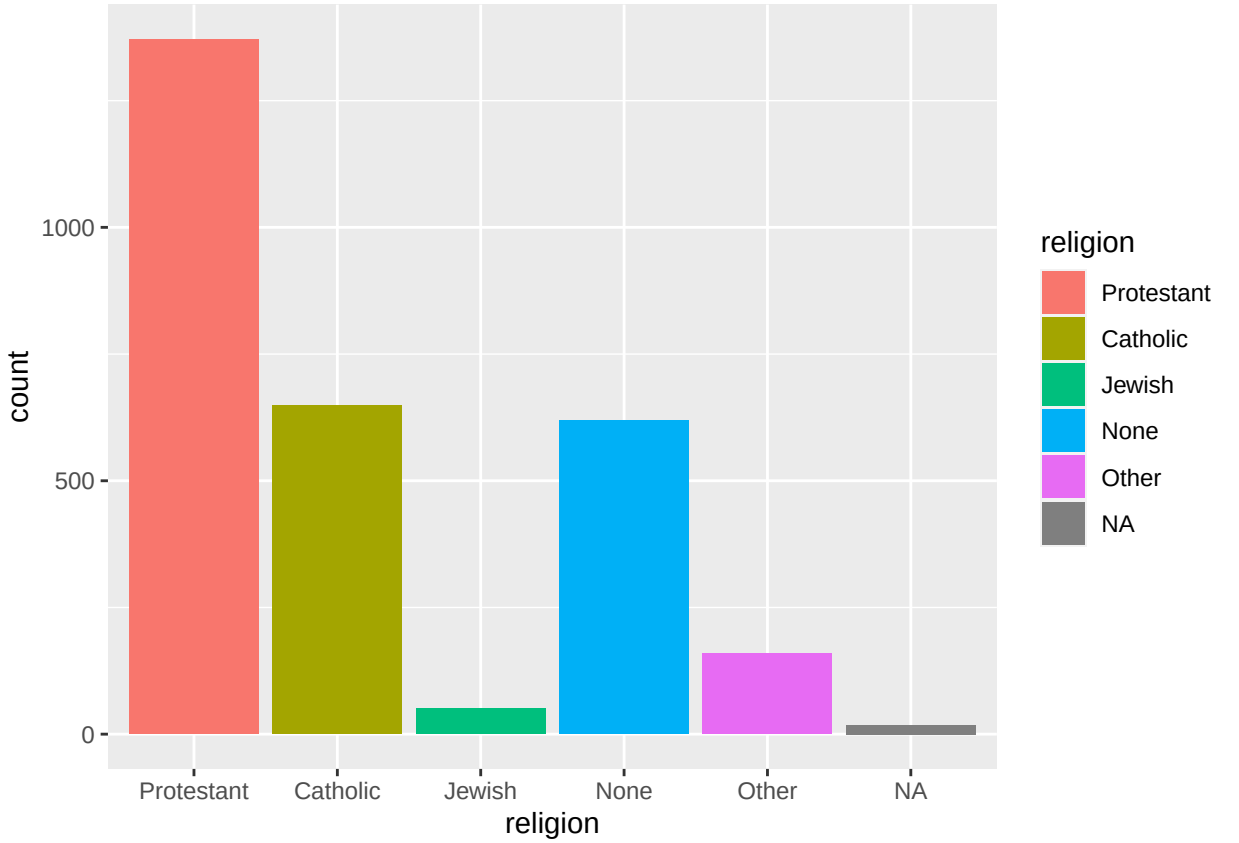
```
##
## Protestant   Catholic     Jewish       None      Other
##       1371        649         51        619        159
```

We can make a bar plot for religion (categorical variable) with color and fill

```
p<- ggplot(data = gss_sm,
           mapping = aes(religion, color = religion))
p + geom_bar()
```

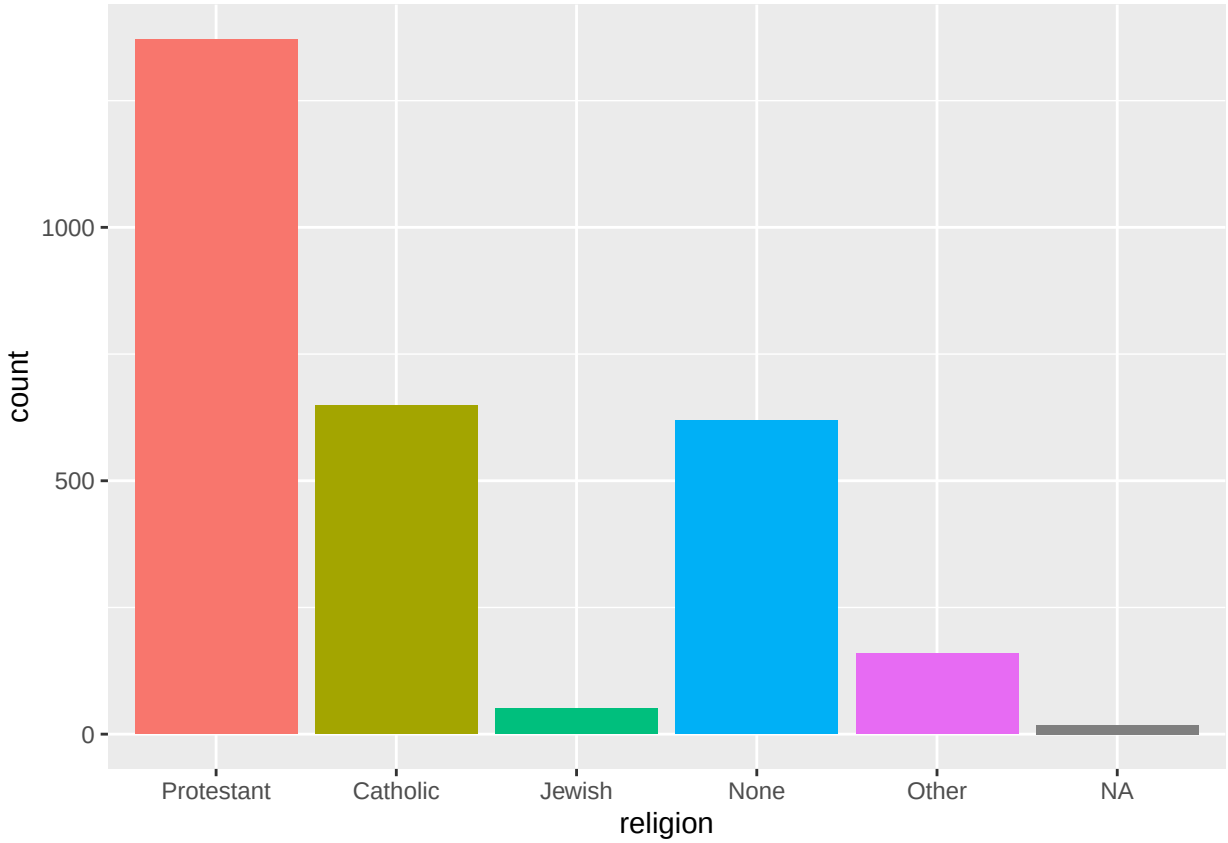



```
p<- ggplot(data = gss_sm,
           mapping = aes(religion))
p + geom_bar(mapping = aes(x = religion, fill = religion))
```




```
#since legends are redundant, we can remove them.
p + geom_bar(mapping = aes(x = religion, fill = religion)) +
    guides(fill = F) #use guides fn to remove legend, F means remove legends
```

```
## Warning: `guides(<scale> = FALSE)` is deprecated. Please use `guides(<scale> =
## "none")` instead.
```

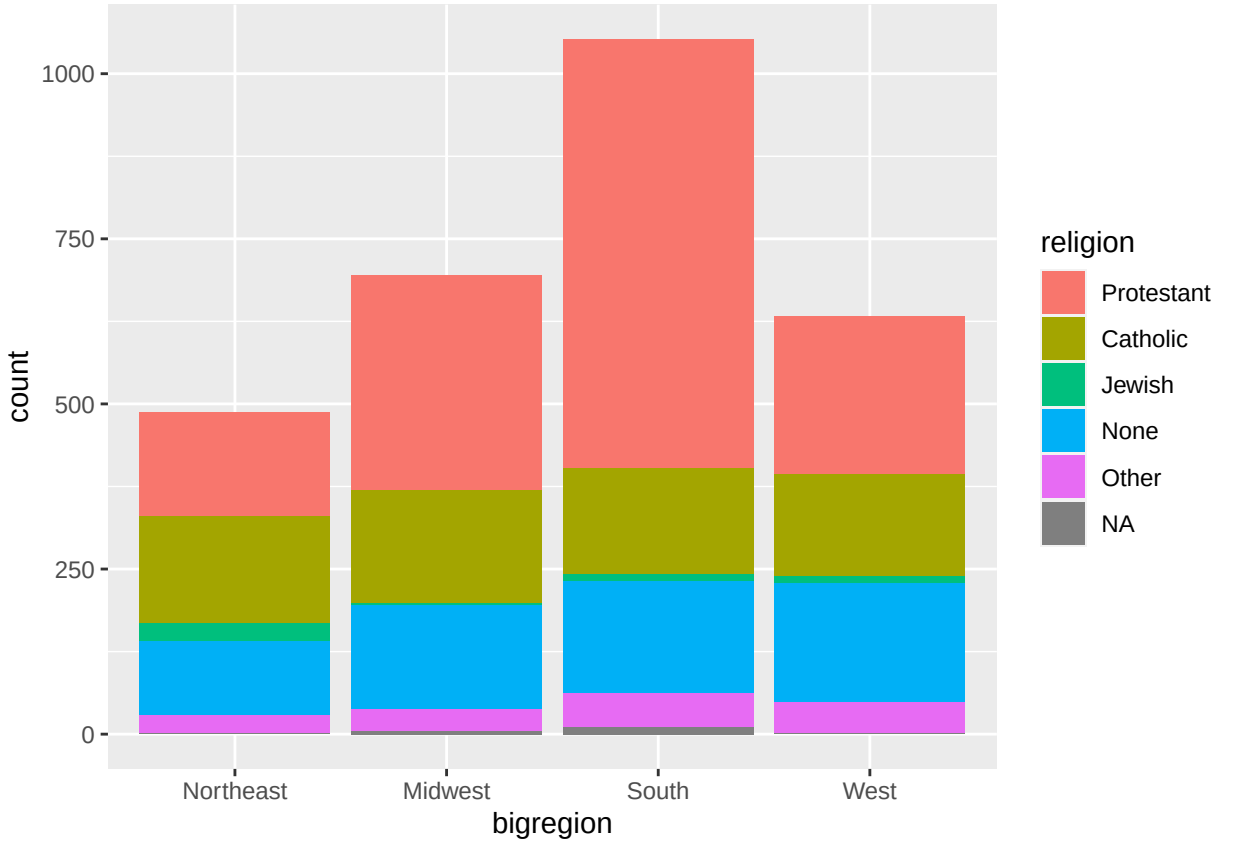


#Frequency plots the slightly awkward way#

cross-classify two categorical variables. W do it by ataked barplot

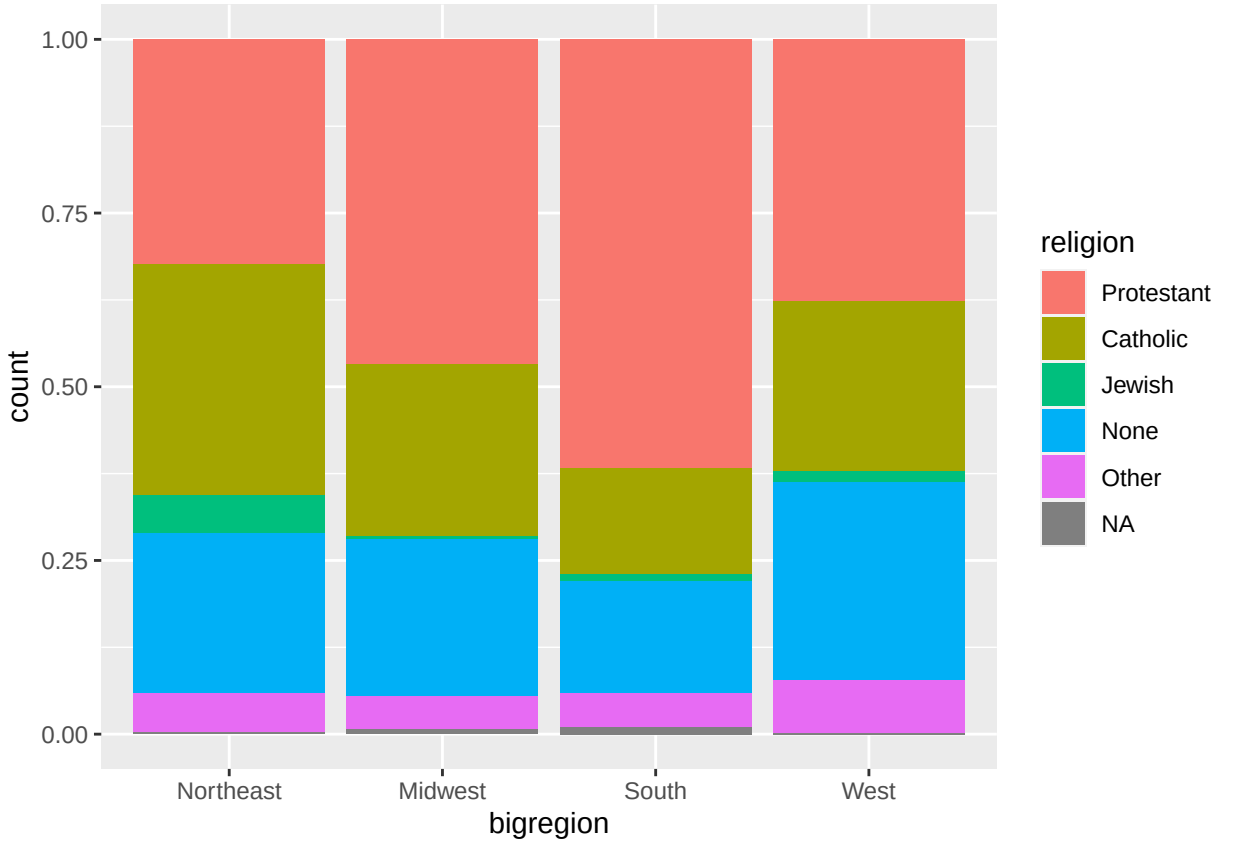
This is the graphical equivalent of a frequency table of counts or proportions. Using the GSS data, for instance, we might want to examine the distribution of religious preferences within different regions of the United States.

```
p<- ggplot(data = gss_sm,
           mapping = aes(x = bigregion,
                         fill = religion))
p + geom_bar()
```



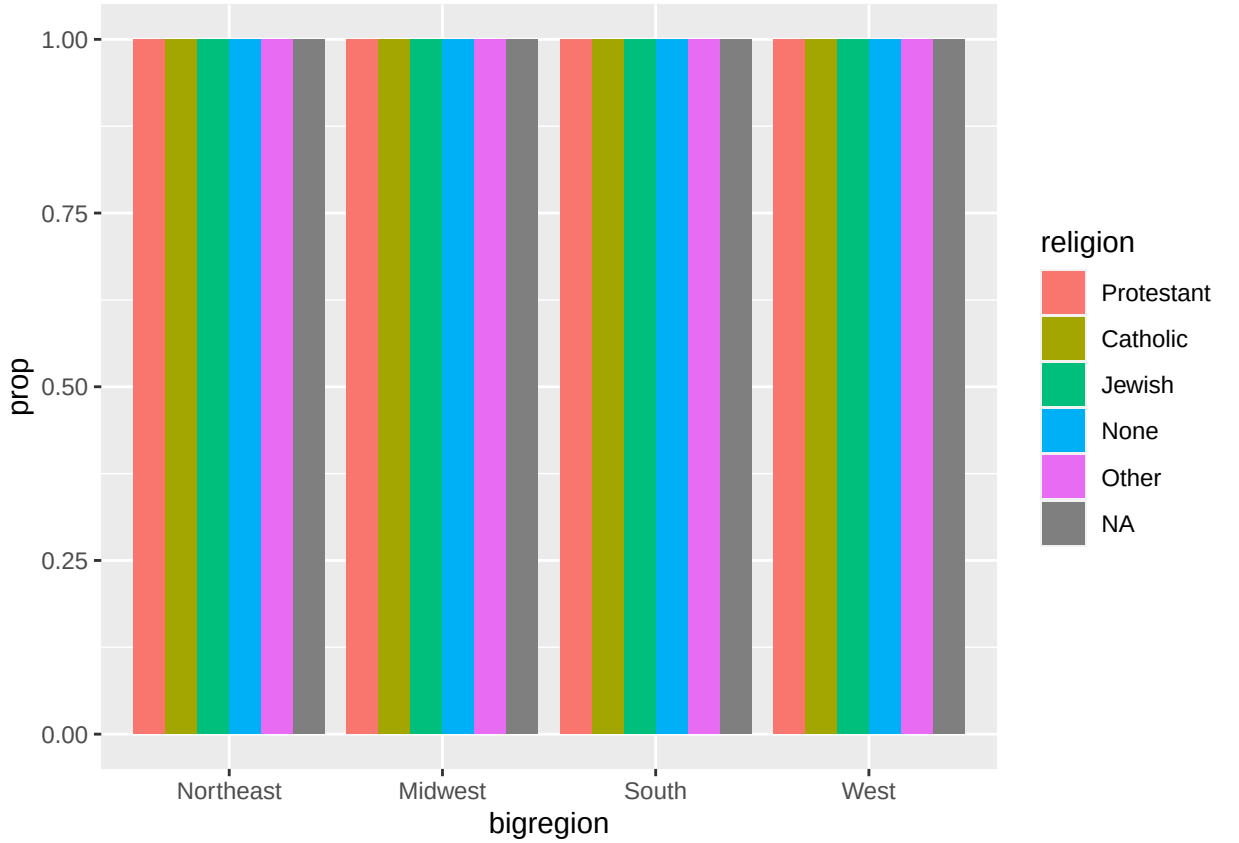
Now, we plot staked barplot with relative frequency

```
p<- ggplot(data = gss_sm,
           mapping = aes(x = bigregion,
                         fill  = religion))
p + geom_bar(position = "fill")
```



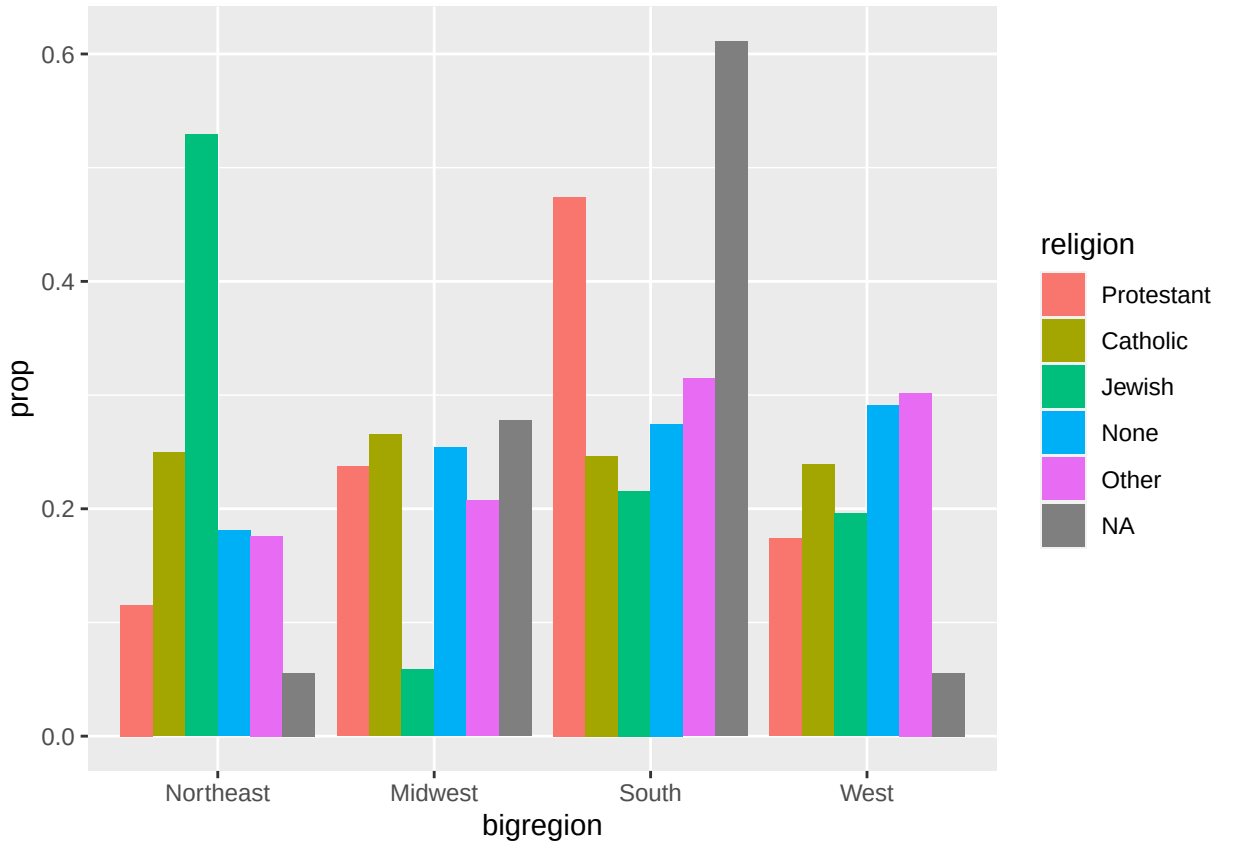
A first go at a dodged bar chart with proportional bars

```
p <- ggplot(data = gss_sm,
            mapping = aes(x = bigregion,
                          fill = religion))
p + geom_bar(position = "dodge",
             mapping = aes(y = ..prop.. )) # to make y-axis as proportion
```



there seems to be an issue with the grouping. When we just wanted the overall proportions for one variable, we mapped group = 1 to tell ggplot to calculate the proportions with respect to the overall N. In this case our grouping variable is religion, so we might try mapping that to the group aesthetic.

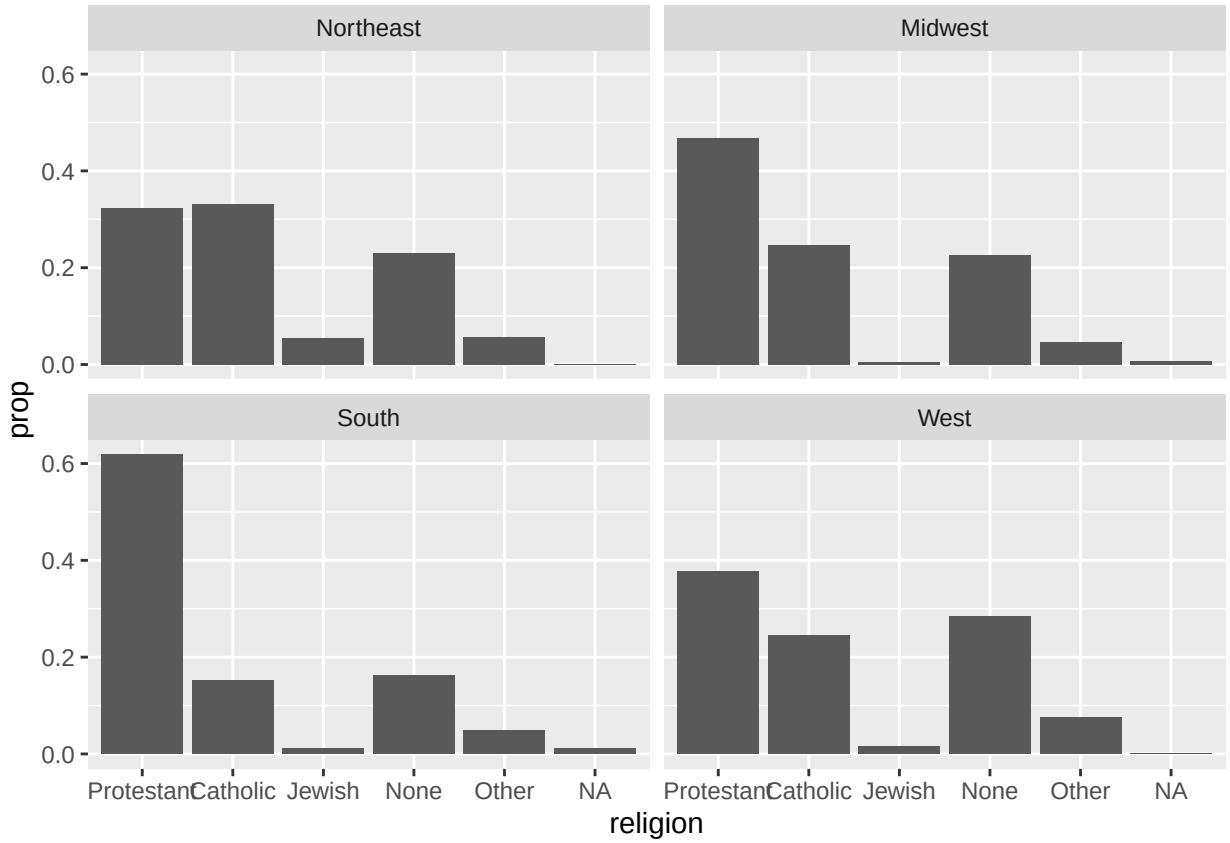
```
p <- ggplot(data = gss_sm,
            mapping = aes(x = bigregion, fill = religion))
p + geom_bar(position = "dodge",
             mapping = aes(y = ..prop.., group = religion))
```



Faceting proportions within region. We can plot different region in different panels bby faceting the plot.

```
p <- ggplot(data = gss_sm,
            mapping = aes(x = religion))

p + geom_bar(position = "dodge",
             mapping = aes(y = ..prop.., group = bigregion))  +
    facet_wrap(~ bigregion, ncol = 2) #make it as four diff plot
```



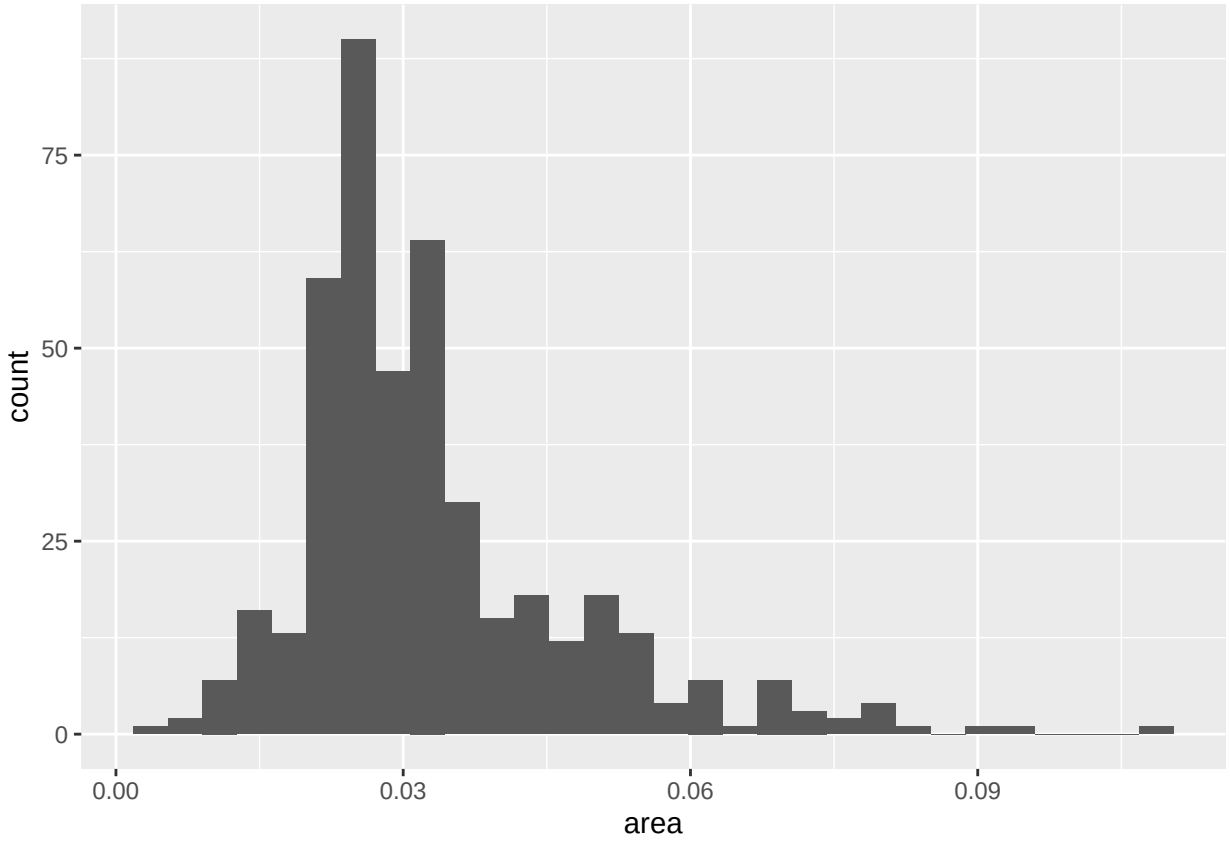
#Histograms and density plots.#

A histogram is a way of summarizing a continuous variable by chopping it up into segments or "bins" and counting how many observations are found within each bin. Because we are summarizing a continuous variable using a series of bars, we need to divide the observations into groups, or bins, and count how many are in each one. By default, the geom_histogram() function will choose a bin size for us based on a rule of thumb.

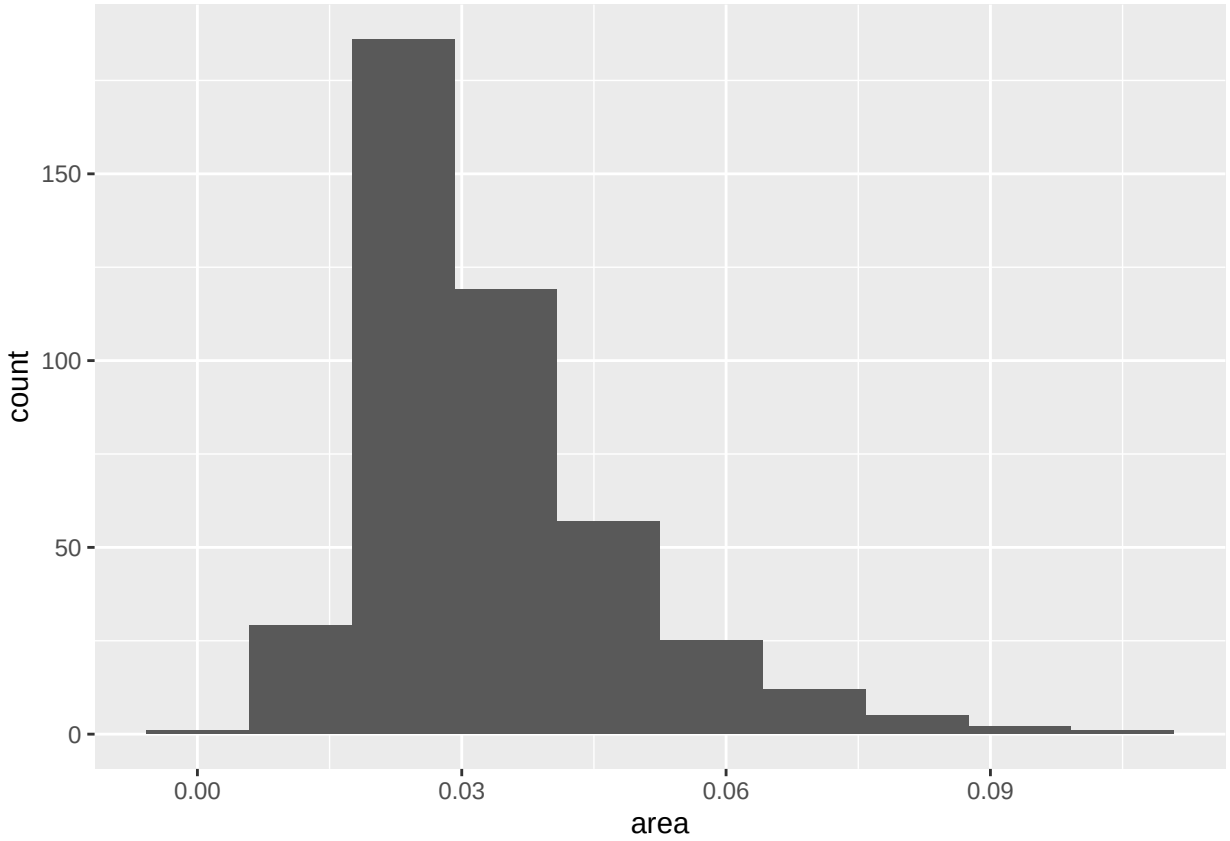
```
p <- ggplot(data = midwest,
            mapping = aes(x = area))

p + geom_histogram()
```

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



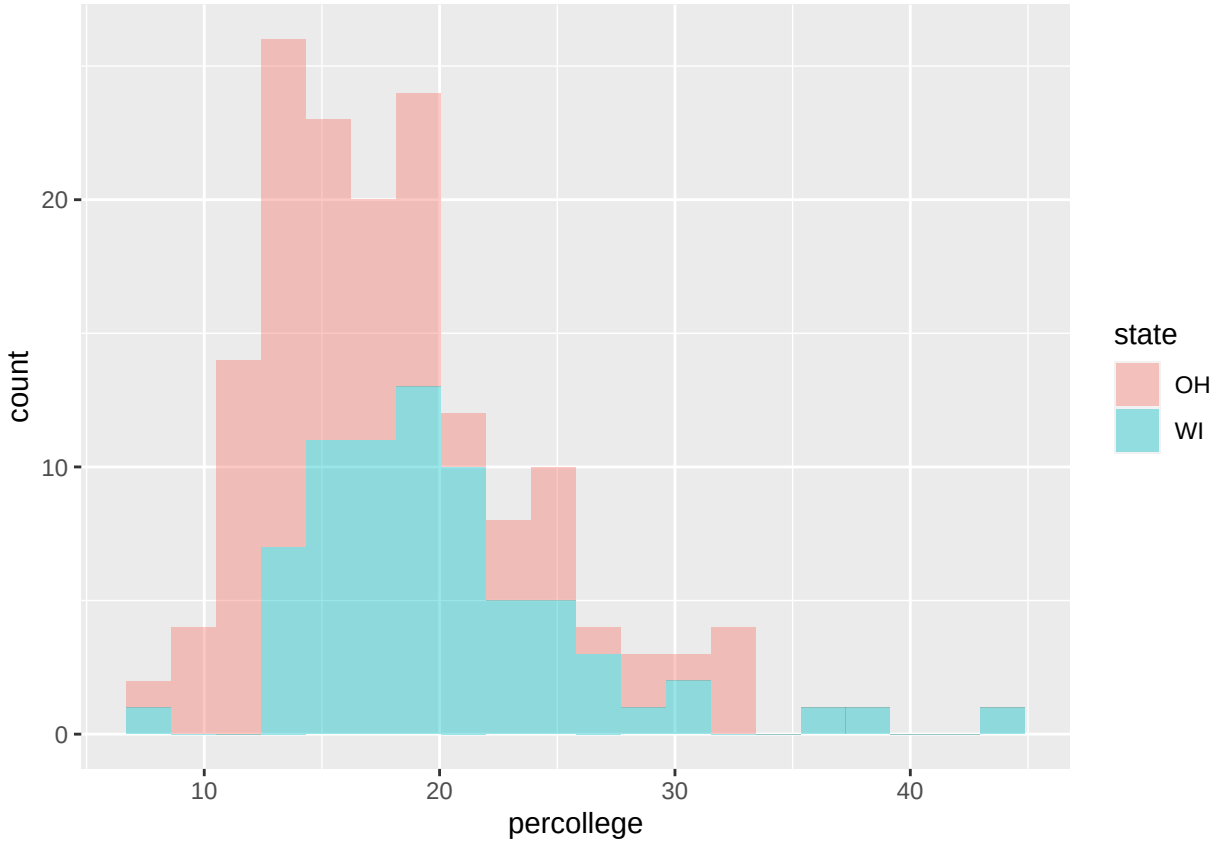

```
p <- ggplot(data = midwest,
              mapping = aes(x = area))
p + geom_histogram(bins = 10)
```



While histograms summarize single variables, it's also possible to use several at once to compare distributions. We can facet histograms by some variable of interest, or as here we can compare them in the same plot using the fill mapping.

```
oh_wi <- c("OH", "WI")

p <- ggplot(data = subset(midwest, subset = state %in% oh_wi), #to subset data
            mapping = aes(x = percollege, fill = state))
p + geom_histogram(alpha = 0.4, bins = 20)
```

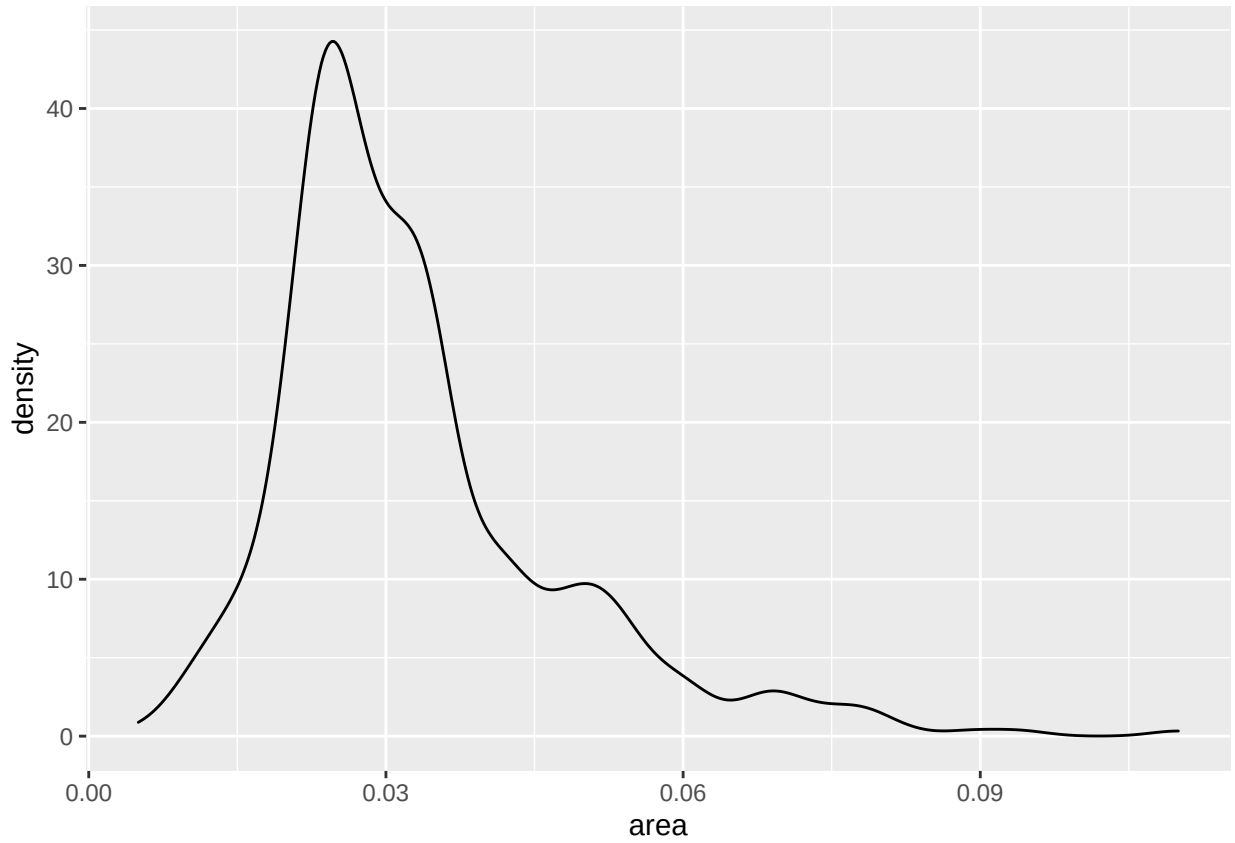


We subset the data here to pick out just two states. To do this we create a character vector with just two elements, "OH" and "WI". Then we use the subset() function to take our data and filter it so that we only select rows whose state name is in this vector. The %in% operator is a convenient way to filter on more than one term in a variable when using subset().

Kernel Density Plot

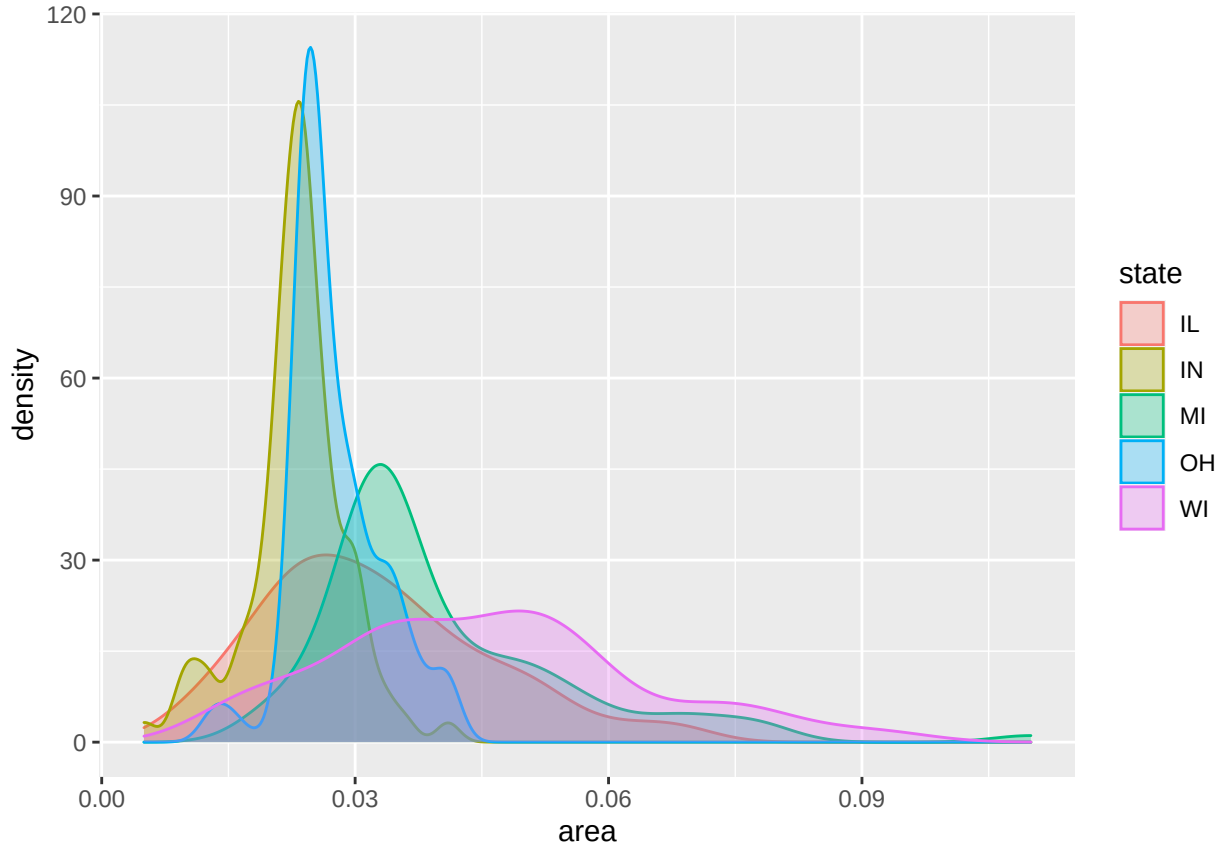
We draw Density plot by 'geom_density' fn.

```
p <- ggplot(data = midwest,
            mapping = aes(x = area))
p + geom_density()
```



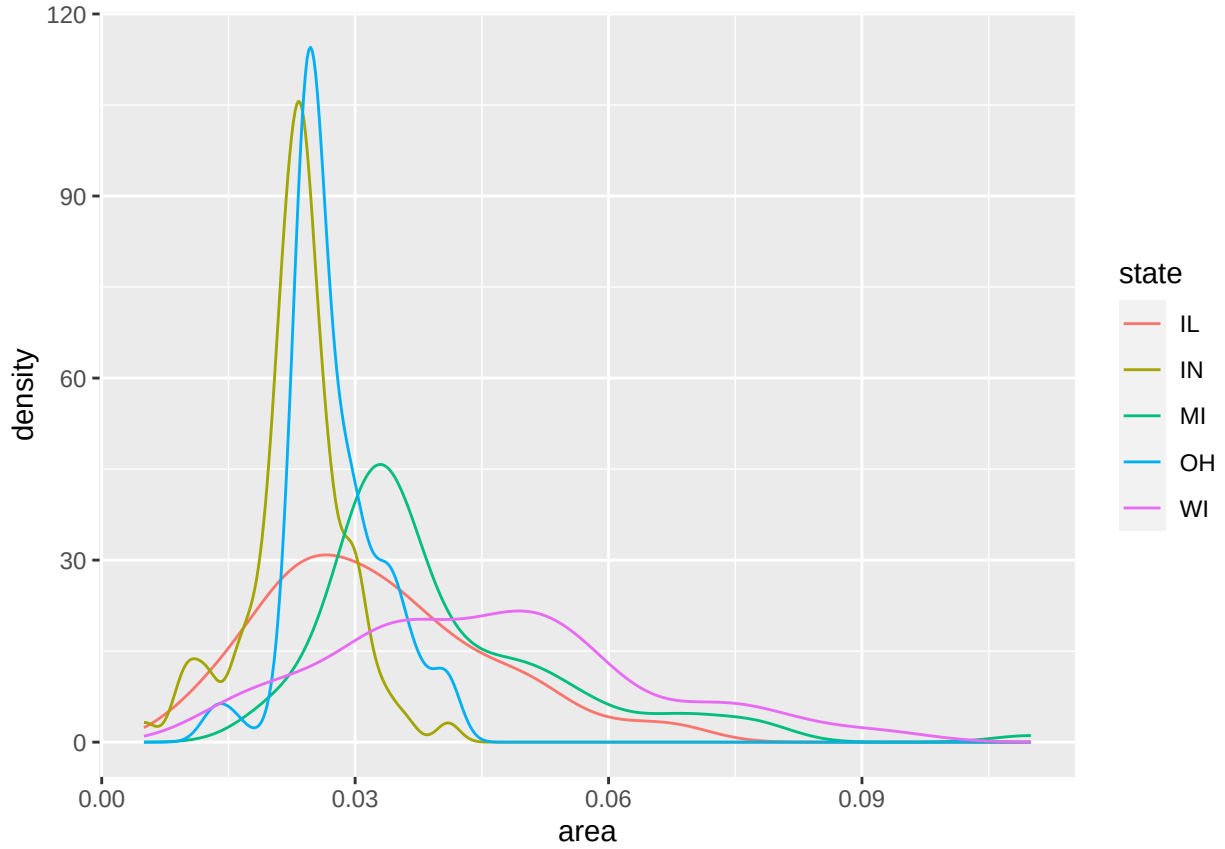
We can use color and fill for this plot to compare more than one distributions.

```
p <- ggplot(data = midwest,
            mapping = aes(x = area,
                          fill = state,
                          color = state))
p + geom_density(alpha = 0.3)
```



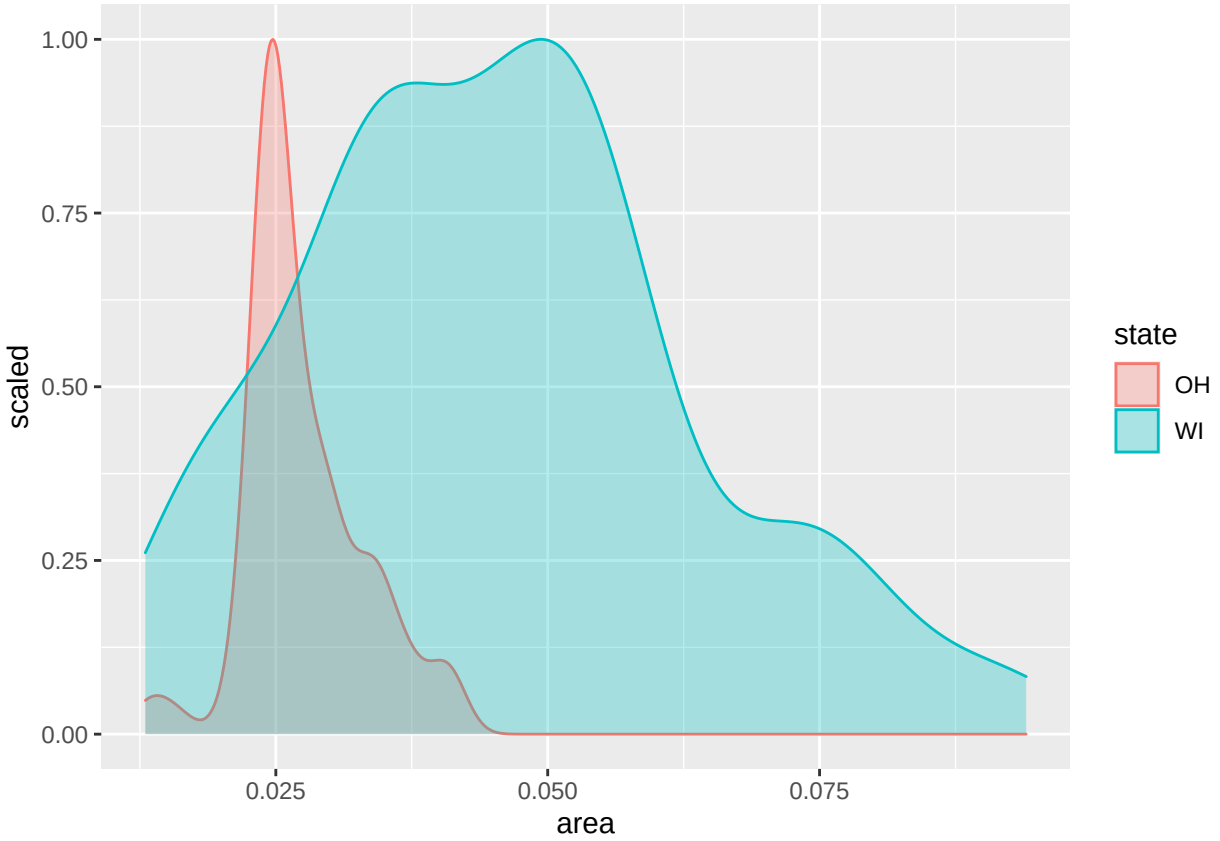
If you want to make the baselines of the density curves go away, you can use geom_line(stat = “density”) instead.

```
p <- ggplot(data = midwest,
            mapping = aes(x = area,
                          fill = state,
                          color = state))
p + geom_line(stat = "density")
```



Just like geom_bar(), the count-based defaults computed by the stat_ functions used by geom_histogram() and geom_density() will return proportional measures if we ask them. For geom_density(), the stat_density() function can return its default ..density.. statistic, or ..scaled.., which will give a proportional density estimate. It can also return a statistic called ..count.., which is the density times the number of points. This can be used in stacked density plots.

```
p <- ggplot(data = subset(midwest, subset = state %in% oh_wi),
            mapping = aes(x = area, fill = state, color = state))
p + geom_density(alpha = 0.3, mapping = (aes(y = ..scaled..)))  #scaled density
```



###Avoid transformations when necessary###

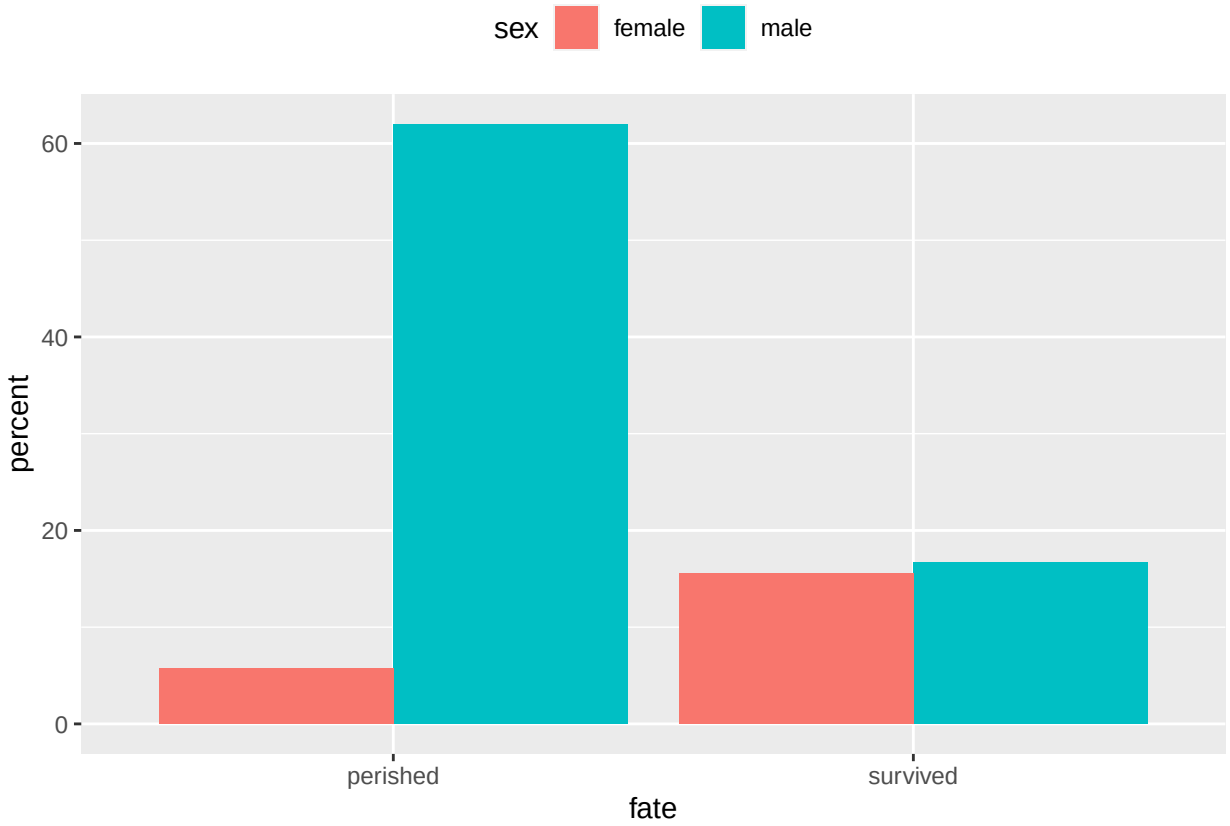
When possible, we avoid transformation bcz sometimes our data is already in summary table. Let's consider the codes below. we do not need the services of any stat_ functions that geom_bar() would normally call. We can tell geom_bar() not to do any work on the variable before plotting it. To do this we say stat = 'identity' in the geom_bar() call. We'll also move the legend to the top of the chart.

```
#load data
titanic
```

```
##       fate    sex    n percent
## 1 perished   male 1364    62.0
## 2 perished female  126     5.7
## 3 survived   male  367    16.7
## 4 survived female  344    15.6
```

```
p <- ggplot(data = titanic,
            mapping = aes(x = fate, y = percent, fill = sex)) #make fill by sex

p+ geom_bar(position = "dodge", stat = "identity")  + #don't transform data to table
            theme(legend.position = "top")   #make legend on the top of fig
```



For convenience ggplot also provides a related geom, geom_col(), which has exactly the same effect but assumes that stat = "identity". We will use this form in future when we don't need any calculations done on the plot.

###Bar chart to compare groups for a threshold.###

This allows us to do things like, for example, plot a flow of positive and negative values in a bar chart. This sort of graph is an alternative to a line plot and is often seen in public policy settings where changes relative to some threshold level or baseline are of interest. For example, the oecd_sum table in socviz contains information on average life expectancy at birth within the United States, and across other OECD countries. We will plot the difference over time, and use the hi_lo variable to color the columns in the chart.

```
#load data
oecd_sum
```

```
## # A tibble: 57 x 5
## # Groups:   year [57]
##     year other   usa  diff hi_lo
##    <int> <dbl> <dbl> <dbl> <chr>
##  1  1960  68.6  69.9 1.30  Below
##  2  1961  69.2  70.4 1.20  Below
##  3  1962  68.9  70.2 1.30  Below
##  4  1963  69.1  70   0.900 Below
##  5  1964  69.5  70.3 0.800 Below
##  6  1965  69.6  70.3 0.700 Below
##  7  1966  69.9  70.3 0.400 Below
##  8  1967  70.1  70.7 0.600 Below
##  9  1968  70.1  70.4 0.300 Below
```

```
## 10  1969  70.1  70.6 0.5   Below
## # ... with 47 more rows
```

```
p <- ggplot(data = oecd_sum,
            mapping = aes(x = year, y = diff, fill = hi_lo ))
p+ geom_col() + guides(fill = F) + #tells ggplot to drop the unnecessary legend
      labs(x = NULL, y = "Difference in years",
          title = "Life expectancy of US",
          subtitle = "Difference between US and OECD in life expectancy",
          caption = "Data: OECD")
```

```
## Warning: `guides(<scale> = FALSE)` is deprecated. Please use `guides(<scale> =
## "none")` instead.
```

```
## Warning: Removed 1 rows containing missing values (position_stack).
```