431 Class 03 Demo: Exploring the Short Survey

Thomas E. Love, Ph.D.

2020-09-01

Contents


```
knitr::opts_chunk$set(comment=NA)
options(width = 70)

## add additional libraries/packages here, as needed
## leaving the tidyverse as the last package loaded
library(magrittr); library(tidyverse)
```

Load the Data

We will read in the .csv file of the data using `read_csv` to turn the data frame into a nicely organized *tibble*.

- Since we've carefully stored the data file in the same directory as our R Project, we can read it in directly.

```
## if you want to load in a data set called namebeta.csv
## and then create a tibble from it called namealpha
## then uncomment the next line by removing the #

day1 <- read_csv("surveyday1_2020.csv")
```

```
Parsed with column specification:
cols(
  .default = col_double(),
  sex = col_character(),
  glasses = col_character(),
  english = col_character(),
  favcolor = col_character()
)
See spec(...) for full column specifications.
```

Key Verbs in the Tidyverse for Data Wrangling

Print your tibble by typing its name

```
day1
```

```
# A tibble: 382 x 21
   student sex   glasses english statsofar ageguess smoke h.left
     <dbl> <chr> <chr>   <chr>       <dbl>    <dbl> <dbl>  <dbl>
 1  202001 <NA>  y       n              NA       NA     1     NA
 2  202002 <NA>  y       y              NA       NA     1     NA
 3  202003 <NA>  y       y              NA       NA     1     NA
 4  202004 <NA>  y       n              NA       NA     1     NA
 5  202005 <NA>  y       y              NA       NA     1     NA
 6  202006 <NA>  n       y              NA       NA     1     NA
 7  202007 <NA>  y       y              NA       NA     1     NA
 8  202008 <NA>  n       y              NA       NA     1     NA
 9  202009 <NA>  y       n              NA       NA     1     NA
10  202010 <NA>  y       n              NA       NA     1     NA
# ... with 372 more rows, and 13 more variables: h.right <dbl>,
#   handedness <dbl>, statfuture <dbl>, haircut <dbl>, lecture <dbl>,
#   alone <dbl>, height.in <dbl>, hand.span <dbl>, favcolor <chr>,
#   lastsleep <dbl>, pulse <dbl>, year <dbl>, lovetrueage <dbl>
```

What are the column names?

```
names(day1)
```

```
 [1] "student"     "sex"         "glasses"     "english"
 [5] "statsofar"   "ageguess"    "smoke"       "h.left"
 [9] "h.right"     "handedness"  "statfuture"  "haircut"
[13] "lecture"     "alone"       "height.in"   "hand.span"
```

```
[17] "favcolor"    "lastsleep"   "pulse"       "year"
[21] "lovetrueage"
```

Use select to pick columns / variables from your tibble

```
day1 %>%
    select(favcolor, haircut)
```

```
# A tibble: 382 x 2
   favcolor haircut
   <chr>      <dbl>
 1 blue           5
 2 blue           0
 3 purple        45
 4 blue           3
 5 purple         0
 6 silver        45
 7 green          0
 8 blue           0
 9 purple        20
10 green         72
# ... with 372 more rows
```

Use filter to pick rows / subjects from your tibble

```
day1 %>%
    filter(year == 2020)
```

```
# A tibble: 67 x 21
   student sex   glasses english statsofar ageguess smoke h.left
     <dbl> <chr> <chr>   <chr>       <dbl>    <dbl> <dbl>  <dbl>
 1  202001 <NA>  y       n              NA       NA     1     NA
 2  202002 <NA>  y       y              NA       NA     1     NA
 3  202003 <NA>  y       y              NA       NA     1     NA
 4  202004 <NA>  y       n              NA       NA     1     NA
 5  202005 <NA>  y       y              NA       NA     1     NA
 6  202006 <NA>  n       y              NA       NA     1     NA
 7  202007 <NA>  y       y              NA       NA     1     NA
 8  202008 <NA>  n       y              NA       NA     1     NA
 9  202009 <NA>  y       n              NA       NA     1     NA
10  202010 <NA>  y       n              NA       NA     1     NA
# ... with 57 more rows, and 13 more variables: h.right <dbl>,
#   handedness <dbl>, statfuture <dbl>, haircut <dbl>, lecture <dbl>,
#   alone <dbl>, height.in <dbl>, hand.span <dbl>, favcolor <chr>,
#   lastsleep <dbl>, pulse <dbl>, year <dbl>, lovetrueage <dbl>
```

Use count to count the number of observations meeting a criterion

```
day1 %>%
    count(favcolor == "red")
```

```
# A tibble: 3 x 2
  `favcolor == "red"`     n
  <lgl>               <int>
1 FALSE                 337
2 TRUE                   39
3 NA                      6
```

Or to provide a cross-classification:

```
day1 %>%
    count(favcolor == "blue", glasses)
```

```
# A tibble: 8 x 3
  `favcolor == "blue"` glasses     n
  <lgl>                <chr>   <int>
1 FALSE                n          19
2 FALSE                y          62
3 FALSE                <NA>      152
4 TRUE                 n          18
5 TRUE                 y          27
6 TRUE                 <NA>       98
7 NA                   y           2
8 NA                   <NA>        4
```

Use arrange to arrange the rows of a tibble

```
day1 %>%
    count(smoke) %>%
    arrange(desc(n))
```

```
# A tibble: 4 x 2
  smoke     n
  <dbl> <int>
1     1   358
2     2    18
3     3     4
4    NA     2
```

Add new variables with mutate

```
day1 %>%
    mutate(guess_error = ageguess - lovetrueage) %>%
    select(ageguess, lovetrueage, guess_error) %>%
    summary()
```

```
    ageguess     lovetrueage     guess_error      
 Min.   :21.0   Min.   :47.50   Min.   :-31.500  
 1st Qu.:45.0   1st Qu.:49.50   1st Qu.: -6.500  
 Median :48.0   Median :50.50   Median : -2.500  
 Mean   :47.3   Mean   :50.73   Mean   : -2.837  
 3rd Qu.:52.0   3rd Qu.:52.50   3rd Qu.:  0.500  
 Max.   :70.0   Max.   :53.50   Max.   : 20.500  
 NA's   :73                     NA's   :73       
```

Get grouped summaries with group_by and summarize

```
day1 %>%
    group_by(year) %>%
    summarize(n = n(),
              average_sleep = mean(lastsleep),
              min_sleep = min(lastsleep))
```

```
`summarise()` ungrouping output (override with `.groups` argument)
```

```
# A tibble: 7 x 4
   year     n average_sleep min_sleep
  <dbl> <int>         <dbl>     <dbl>
1  2014    42         NA           NA
2  2015    49          7.32         4
3  2016    64         NA           NA
4  2017    48          6.71         2
5  2018    51          6.78         4
6  2019    61         NA           NA
7  2020    67          6.57         2
```

Whoops - looks like we have some missing `lastsleep` values. We could filter our data to only include the subjects who provided data on that variable...

```
day1 %>%
    filter(complete.cases(lastsleep)) %>%
    group_by(year) %>%
    summarize(n = n(),
              average_sleep = mean(lastsleep),
              min_sleep = min(lastsleep))
```

```
`summarise()` ungrouping output (override with `.groups` argument)
```

```
# A tibble: 7 x 4
   year     n average_sleep min_sleep
  <dbl> <int>         <dbl>     <dbl>
1  2014    41          6.90         4
2  2015    49          7.32         4
3  2016    63          7.00         4
4  2017    48          6.71         2
5  2018    51          6.78         4
6  2019    60          7.11         4
7  2020    67          6.57         2
```

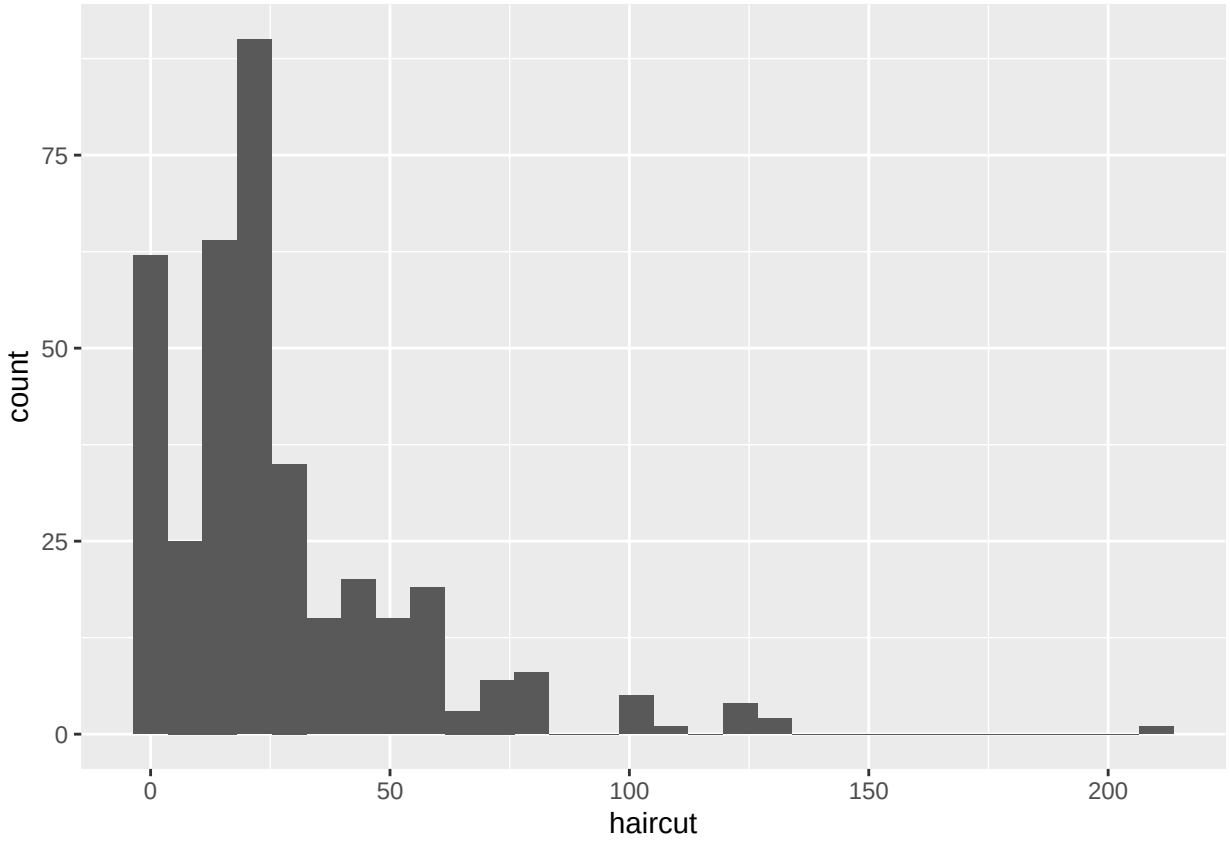
Using ggplot: A Histogram of Haircut Prices

Default Version

```
ggplot(data = day1, aes(x = haircut)) +
    geom_histogram()
```

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

```
Warning: Removed 6 rows containing non-finite values (stat_bin).
```

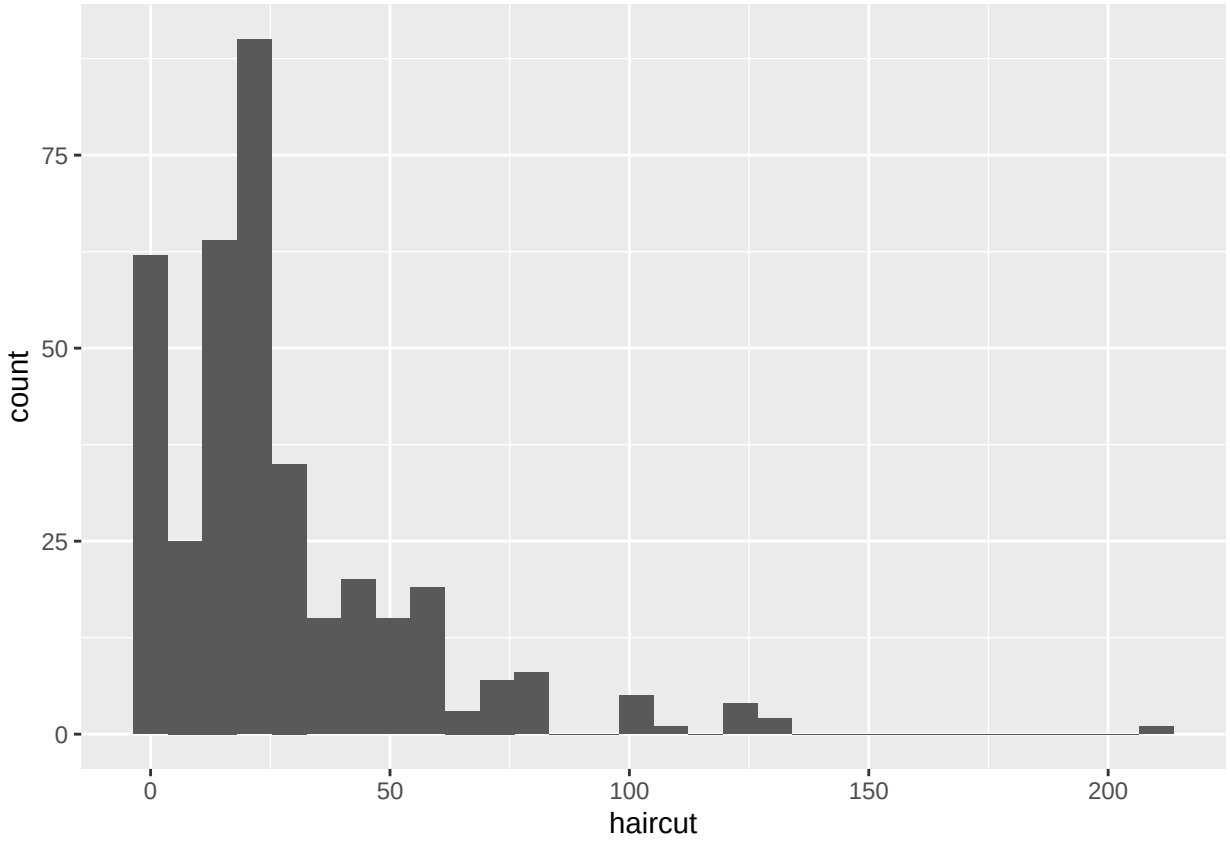


Improvements

1. We'll filter the rows of the `day1` tibble to include only those subjects who gave us a haircut price.

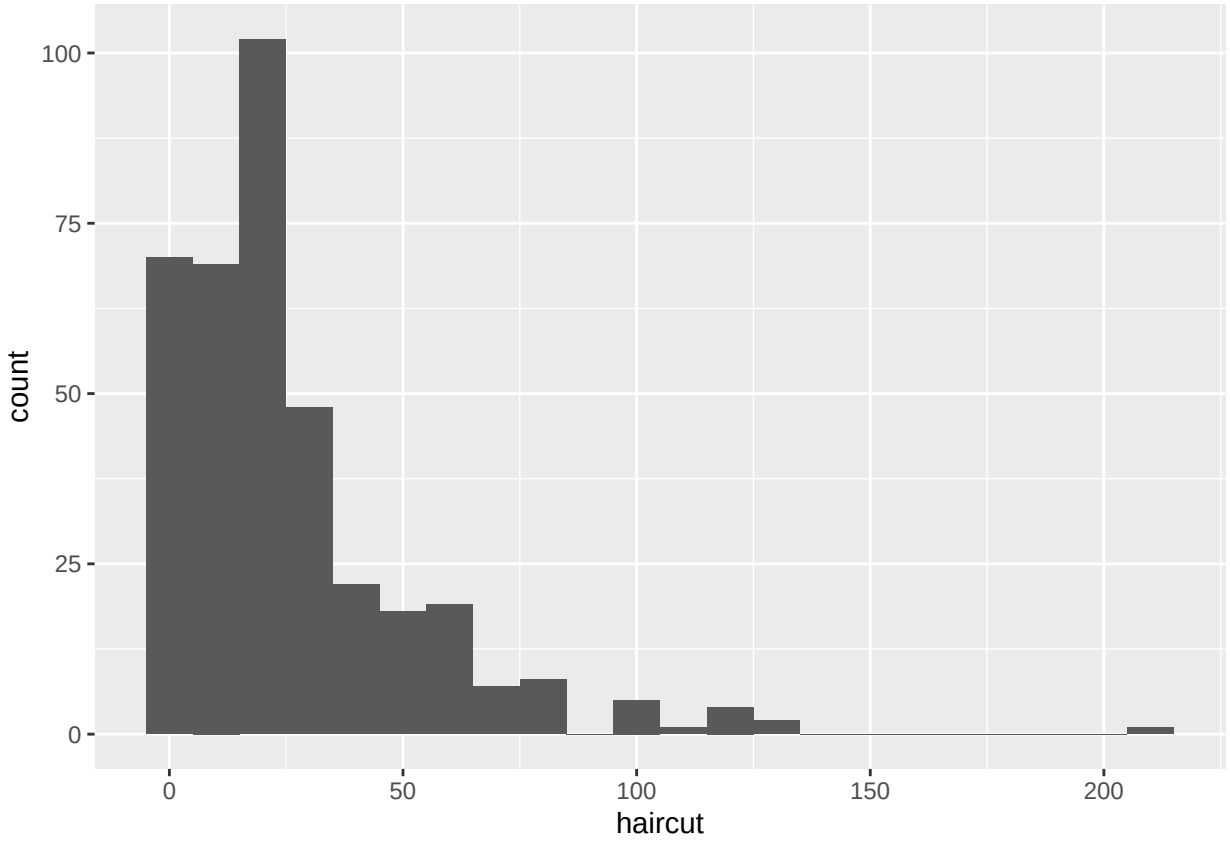
```
day1 %>%
    filter(complete.cases(haircut)) %>%
    ggplot(data = ., aes(x = haircut)) +
    geom_histogram()
```

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



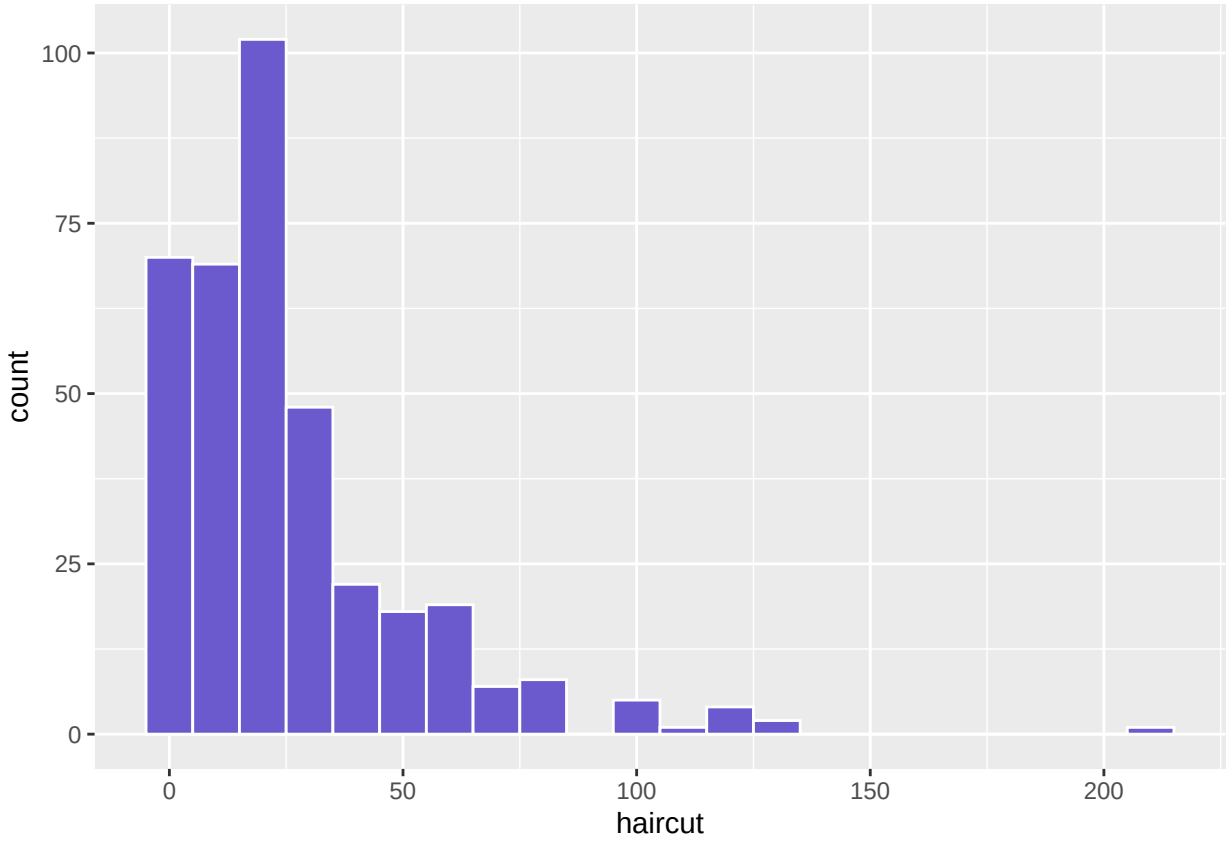
2. We'll specify that R should create bins of width $10 (rather than the default, which creates 30 bins) for the haircut prices to fall in.

```
day1 %>%
    filter(complete.cases(haircut)) %>%
    ggplot(data = ., aes(x = haircut)) +
    geom_histogram(binwidth = 10)
```



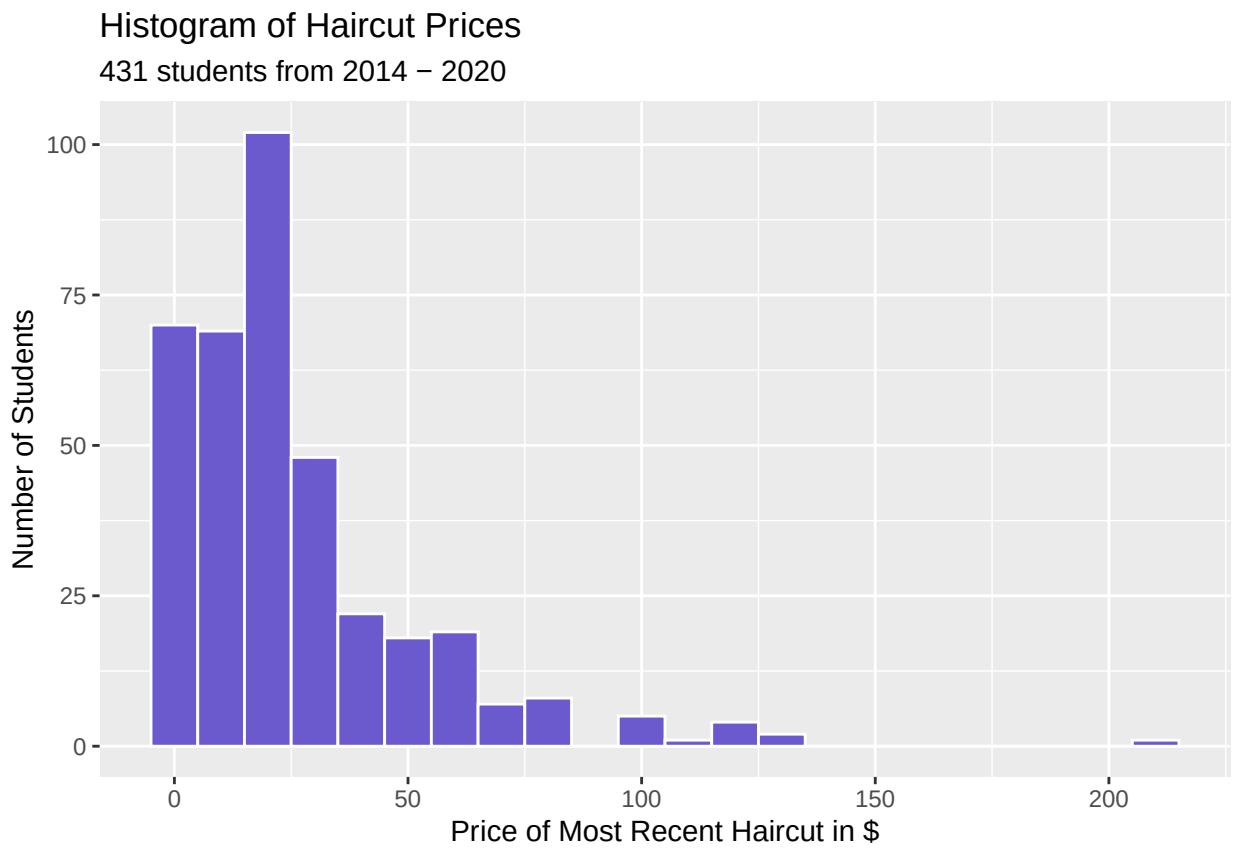
3. We'll set the fill to be a better color - a nice resource for this is to google **Colors in R**. I'll pick "slateblue". We'll also color the outlines of the bars "white".

```
day1 %>%
    filter(complete.cases(haircut)) %>%
    ggplot(data = ., aes(x = haircut)) +
    geom_histogram(binwidth = 10,
                   fill = "slateblue", col = "white")
```



4. We'll build a main title, subtitle and proper axis titles.

```
day1 %>%
    filter(complete.cases(haircut)) %>%
    ggplot(data = ., aes(x = haircut)) +
    geom_histogram(binwidth = 10,
                   fill = "slateblue", col = "white") +
    labs(x = "Price of Most Recent Haircut in $",
         y = "Number of Students",
         title = "Histogram of Haircut Prices",
         subtitle = "431 students from 2014 - 2020")
```



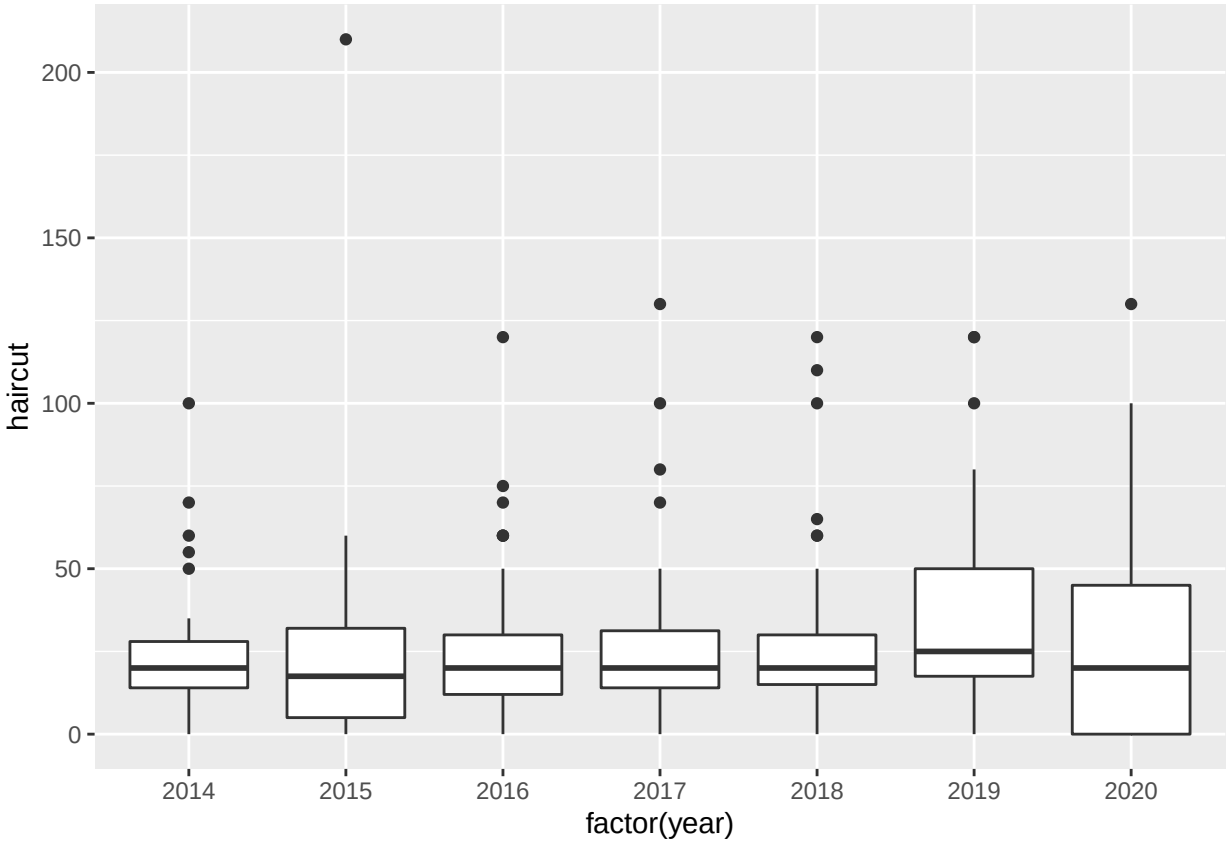
Separate histograms for each year with faceting?

```
day1 %>%
    filter(!is.na(haircut)) %>%
    ggplot(., aes(x = haircut)) +
        geom_histogram(binwidth = 10,
                       fill = "slateblue", color = "white") +
        guides(fill = FALSE) +
        labs(x = "Price of Last Haircut in $",
                 y = "Number of Students") +
        facet_wrap(~ year)
```

Building a Comparison Boxplot

We could use a comparison boxplot. A trick here is to specify `year` as a factor...

```
day1 %>%
    filter(complete.cases(haircut)) %>%
    ggplot(data = .,
           aes(x = factor(year), y = haircut)) +
    geom_boxplot()
```



Numerical Summaries

Detailed Numerical Summary of Haircut Prices

```
day1 %>%
    select(haircut) %>%
    summary()
```

```
    haircut
 Min.   :  0.00
 1st Qu.: 12.00
 Median : 20.00
 Mean   : 27.28
 3rd Qu.: 35.00
 Max.   :210.00
 NA's   :6
```

which can also be done with

```
summary(day1$haircut)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
   0.00   12.00   20.00   27.28   35.00  210.00       6
```

The `mosaic` package has a useful `favstats` function...

```
mosaic::favstats(day1$haircut)
```

```
Registered S3 method overwritten by 'mosaic':
  method                           from
  fortify.SpatialPolygonsDataFrame ggplot2
```

```
 min Q1 median Q3 max     mean       sd   n missing
   0 12     20 35 210 27.28056 26.52167 376       6
```

But to get this in a pipeline, you'd need the `%$%` operator from the `magrittr` package...

```
day1 %$%
    mosaic::favstats(haircut)
```

```
 min Q1 median Q3 max     mean       sd   n missing
   0 12     20 35 210 27.28056 26.52167 376       6
```

The `psych` package has a useful `describe` function...

```
day1 %$%
    psych::describe(haircut)
```

```
   vars   n  mean    sd median trimmed   mad min max range skew
X1    1 376 27.28 26.52     20    23.2 14.83   0 210   210 2.16
   kurtosis   se
X1     7.63 1.37
```

The `Hmisc` package also has a useful `describe` function...

```
day1 %$%
    Hmisc::describe(haircut)
```

```
haircut
       n  missing distinct     Info     Mean      Gmd      .05
     376        6       53    0.992    27.28    26.19        0
     .10      .25      .50      .75      .90      .95
       0       12       20       35       60       80

lowest :   0.0   1.0   3.0   3.5   5.0, highest: 100.0 110.0 120.0 130.0 210.0
```

Numerical Summary by Year?

```
day1 %>%
    filter(!is.na(haircut)) %>%
    group_by(year) %>%
    summarize(n = n(), mean = mean(haircut),
              sd = sd(haircut), median = median(haircut))
```

```
`summarise()` ungrouping output (override with `.groups` argument)
```

```
# A tibble: 7 x 5
   year     n  mean    sd median
  <dbl> <int> <dbl> <dbl>  <dbl>
1  2014    41  23.7  19.8   20
2  2015    49  24.7  32.8   17.5
3  2016    63  23.8  21.5   20
4  2017    48  25.9  25.3   20
5  2018    50  28.3  26.1   20
```

```
6  2019    60  35.8  29.1   25
7  2020    65  27.1  27.5   20
```

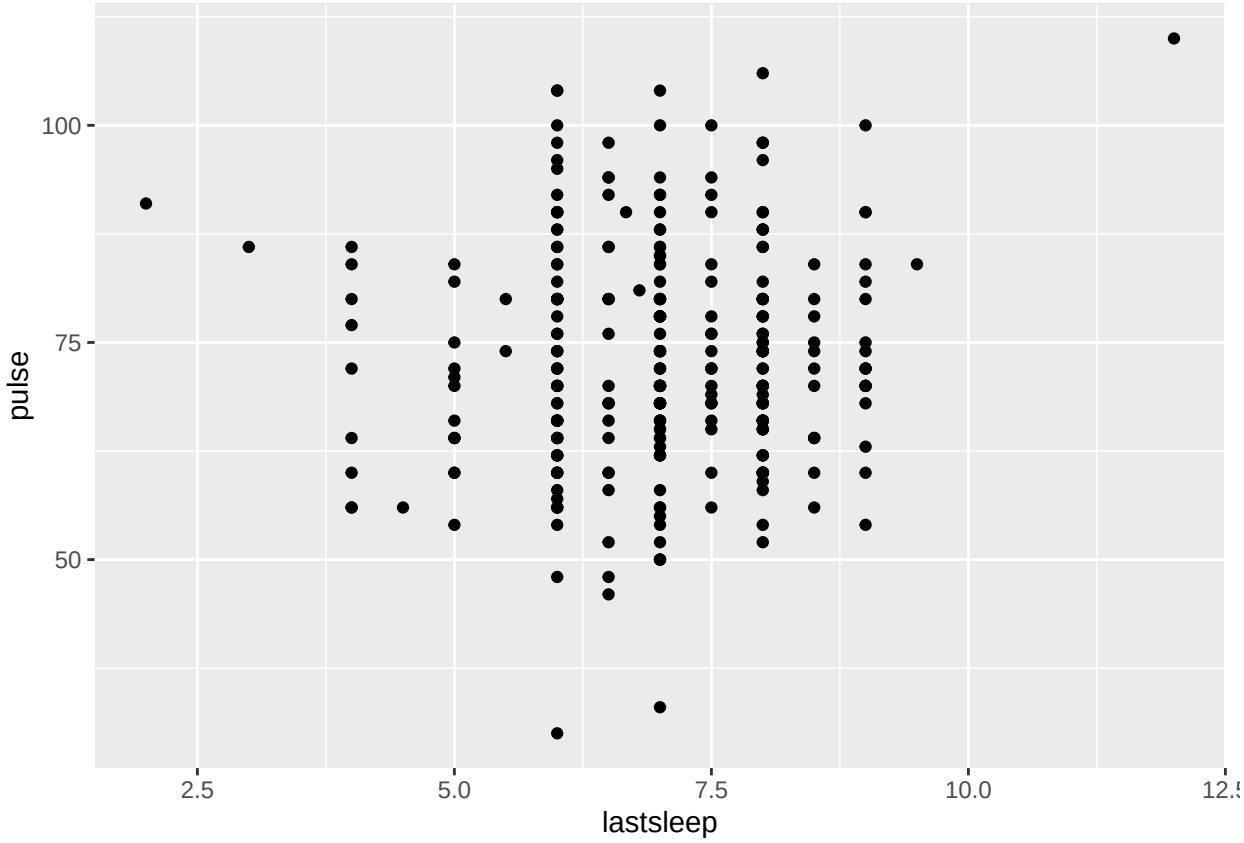
Using ggplot scatterplots

What is the relationship between 431 students' pulse rate and hours of sleep the prior night?

Here, we're looking at two quantitative variables. A scatterplot is usually the best choice.

```
ggplot(data = day1, aes(x = lastsleep, y = pulse)) +
    geom_point()
```

```
Warning: Removed 70 rows containing missing values (geom_point).
```



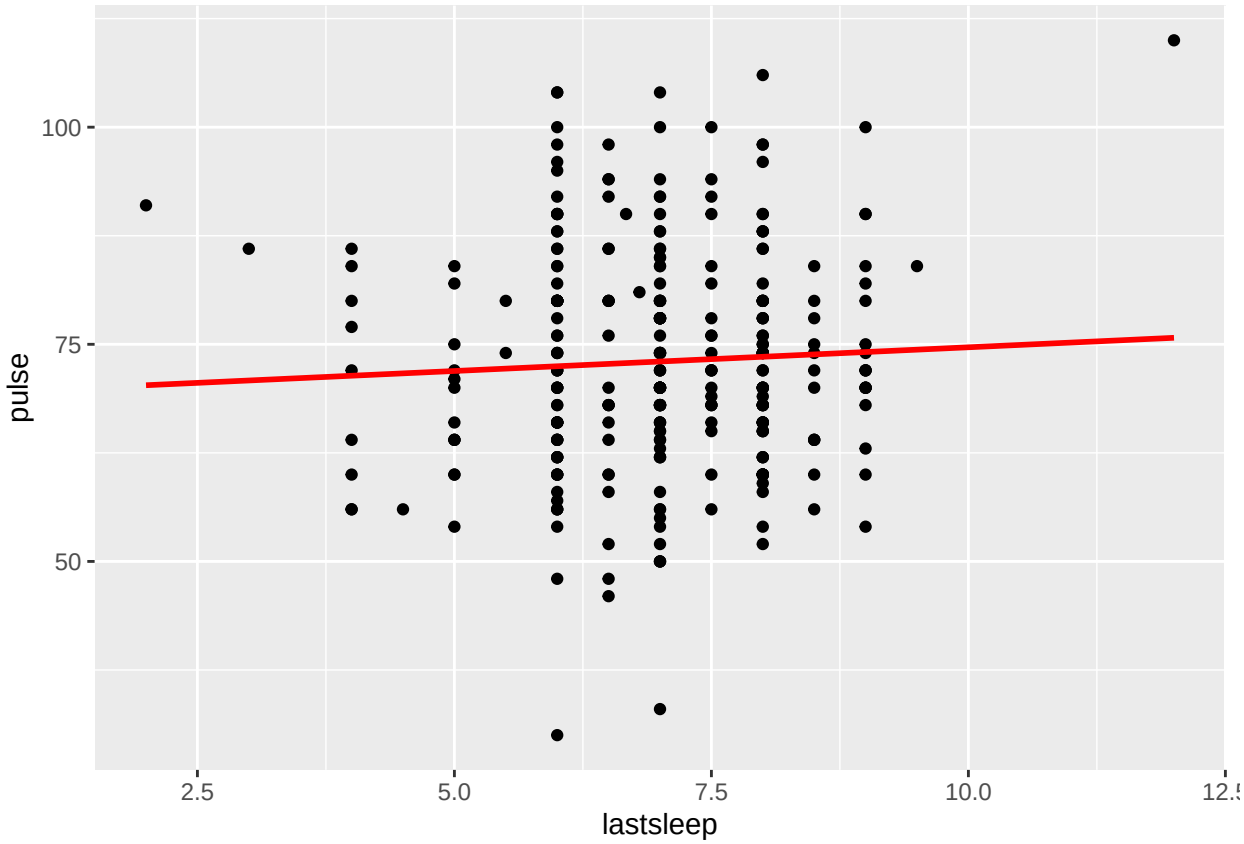
Improving the Scatterplot

Let's filter to include only those cases with known `pulse` and known `lastsleep`, and also add a line from a linear regression model to predict pulse rate on the basis of hours of sleep the prior night.

```
day1 %>%
    filter(complete.cases(pulse, lastsleep)) %>%
    ggplot(data = ., aes(x = lastsleep, y = pulse)) +
```

```
    geom_point() +
    geom_smooth(method = "lm", se = FALSE, col = "red")
```

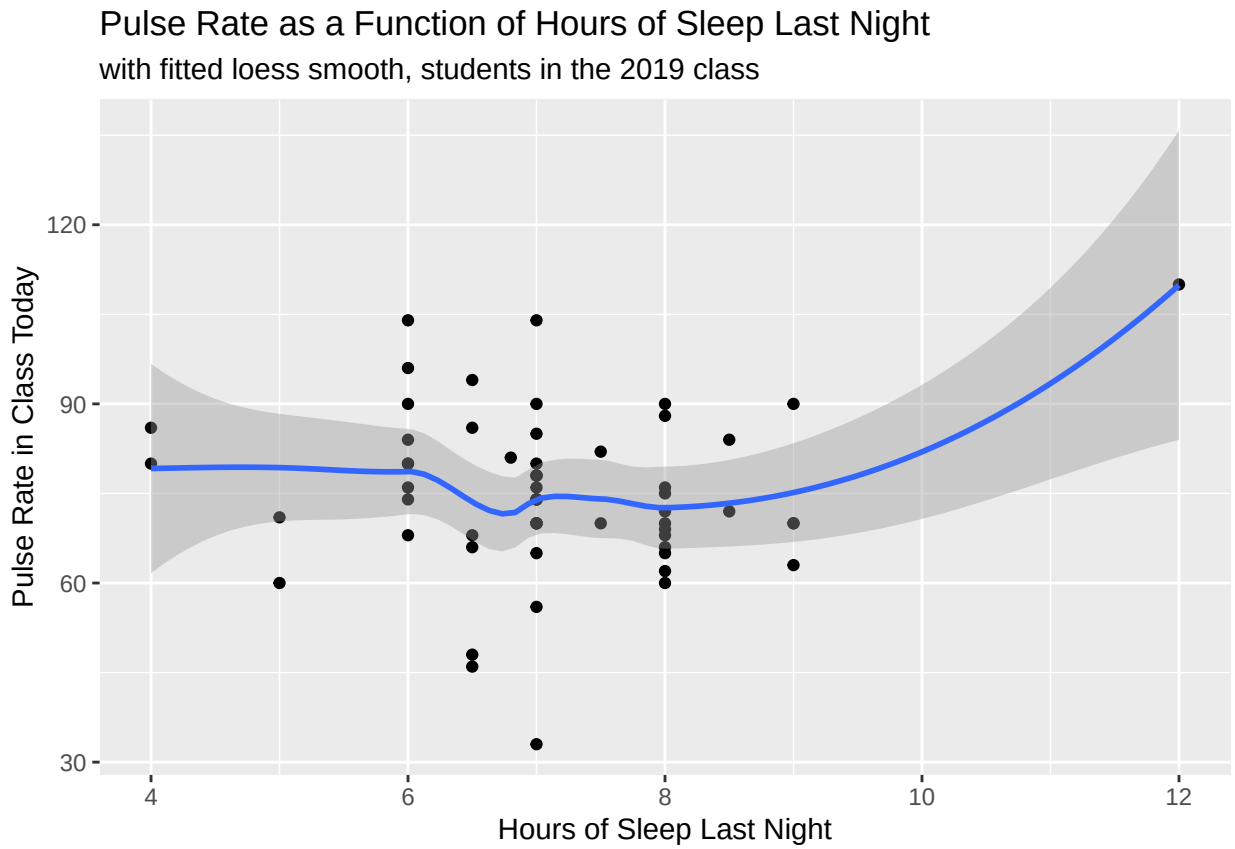
```
`geom_smooth()` using formula 'y ~ x'
```



Let's look at the 2019 data only, and fit a curved (loess) smooth to predict pulse rate on the basis of hours of sleep the prior night. We'll also add a title and subtitle and retitle the axes

```
day1 %>%
    filter(year == "2019") %>%
    filter(complete.cases(pulse, lastsleep)) %>%
    ggplot(data = ., aes(x = lastsleep, y = pulse)) +
    geom_point() +
    geom_smooth(method = "loess") +
    labs(title = "Pulse Rate as a Function of Hours of Sleep Last Night",
         subtitle = "with fitted loess smooth, students in the 2019 class",
         x = "Hours of Sleep Last Night",
         y = "Pulse Rate in Class Today")
```

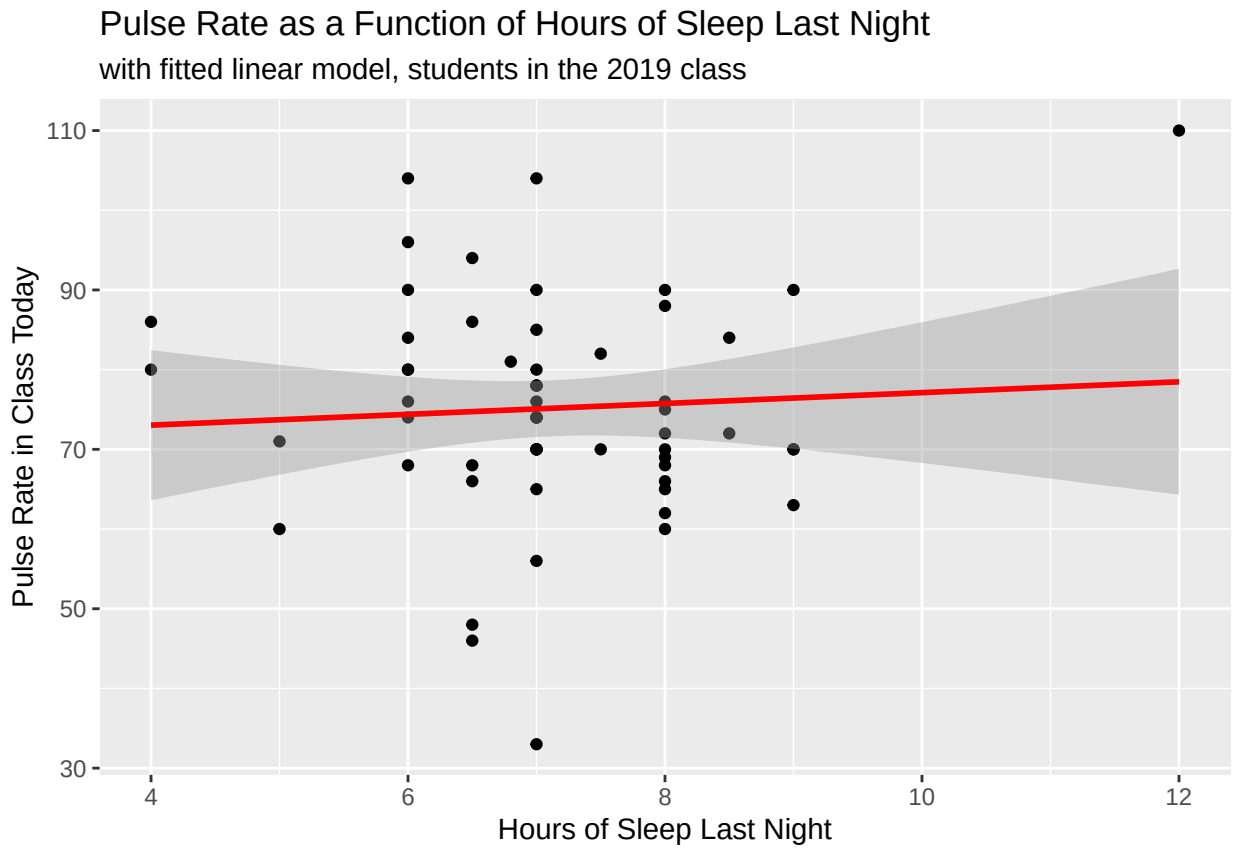
```
`geom_smooth()` using formula 'y ~ x'
```



We could instead restrict ourselves to a linear model for the 2019 group.

```
day1 %>%
    filter(year == "2019") %>%
    filter(complete.cases(pulse, lastsleep)) %>%
    ggplot(data = ., aes(x = lastsleep, y = pulse)) +
    geom_point() +
    geom_smooth(method = "lm", col = "red") +
    labs(title = "Pulse Rate as a Function of Hours of Sleep Last Night",
         subtitle = "with fitted linear model, students in the 2019 class",
         x = "Hours of Sleep Last Night",
         y = "Pulse Rate in Class Today")
```

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



The correlation of `lastsleep` and `pulse` is likely to be of some interest. Note the use of both the `%>%` and `%$%` pipes in this case.

```
day1 %>%
    filter(year == "2019") %>%
    filter(complete.cases(pulse, lastsleep)) %$%
    cor(pulse, lastsleep)
```

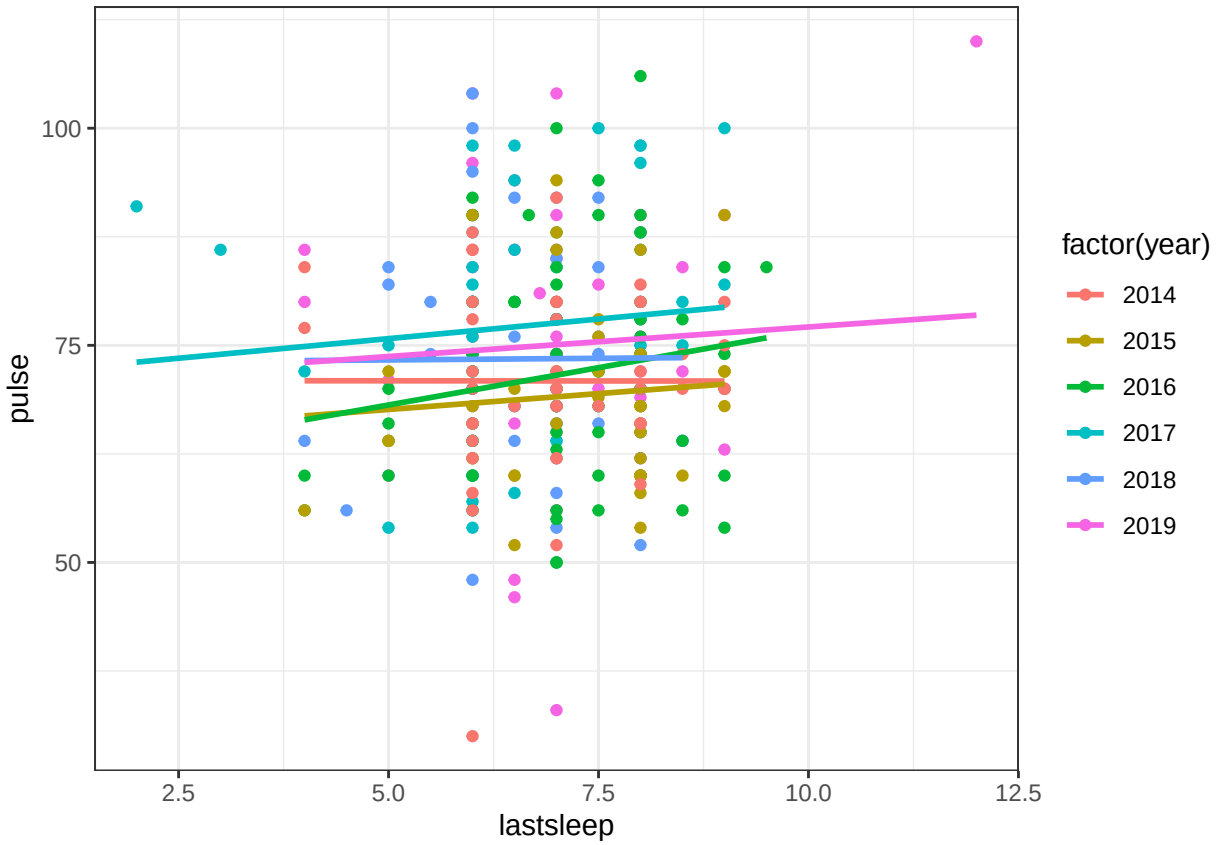
```
[1] 0.06356228
```

Does the linear model change much by year?

Here's the plot, color coding the models by year (note the use of the group as well as the color aesthetic here), and also incorporating the black-and-white theme, rather than the default.

```
day1 %>%
    filter(complete.cases(pulse, lastsleep)) %>%
    ggplot(., aes(x = lastsleep, y = pulse,
                                color = factor(year),
                                group = factor(year))) +
    geom_point() +
    geom_smooth(method = "lm", se = FALSE) +
      theme_bw()
```

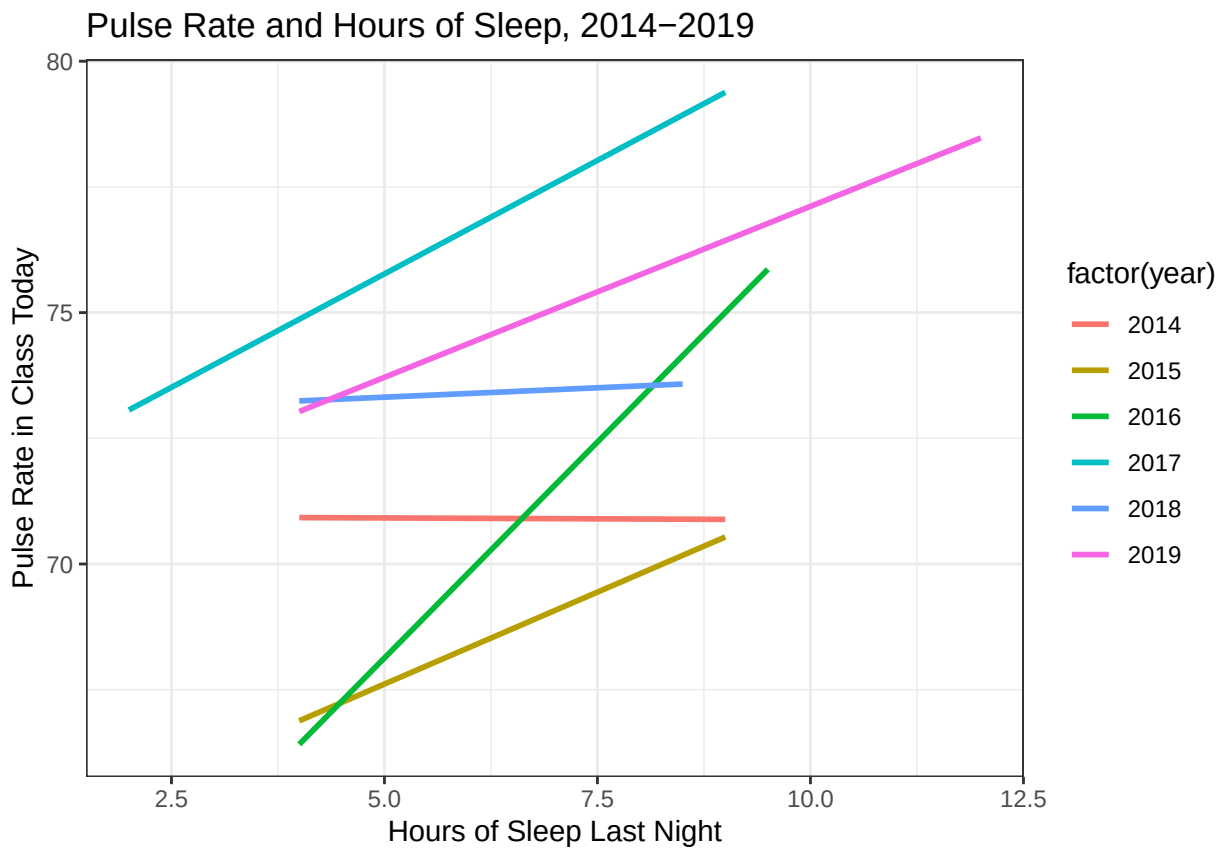
```
`geom_smooth()` using formula 'y ~ x'
```



Here's the same plot of the models alone, and not showing the data (commenting out the line of code that draws the points.) We'll also improve the labeling.

```
day1 %>%
    filter(complete.cases(pulse, lastsleep)) %>%
    ggplot(., aes(x = lastsleep, y = pulse,
                                color = factor(year),
                                group = factor(year))) +
#    geom_point() +
    geom_smooth(method = "lm", se = FALSE) +
    labs(title = "Pulse Rate and Hours of Sleep, 2014-2019",
         x = "Hours of Sleep Last Night",
         y = "Pulse Rate in Class Today") +
      theme_bw()
```

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

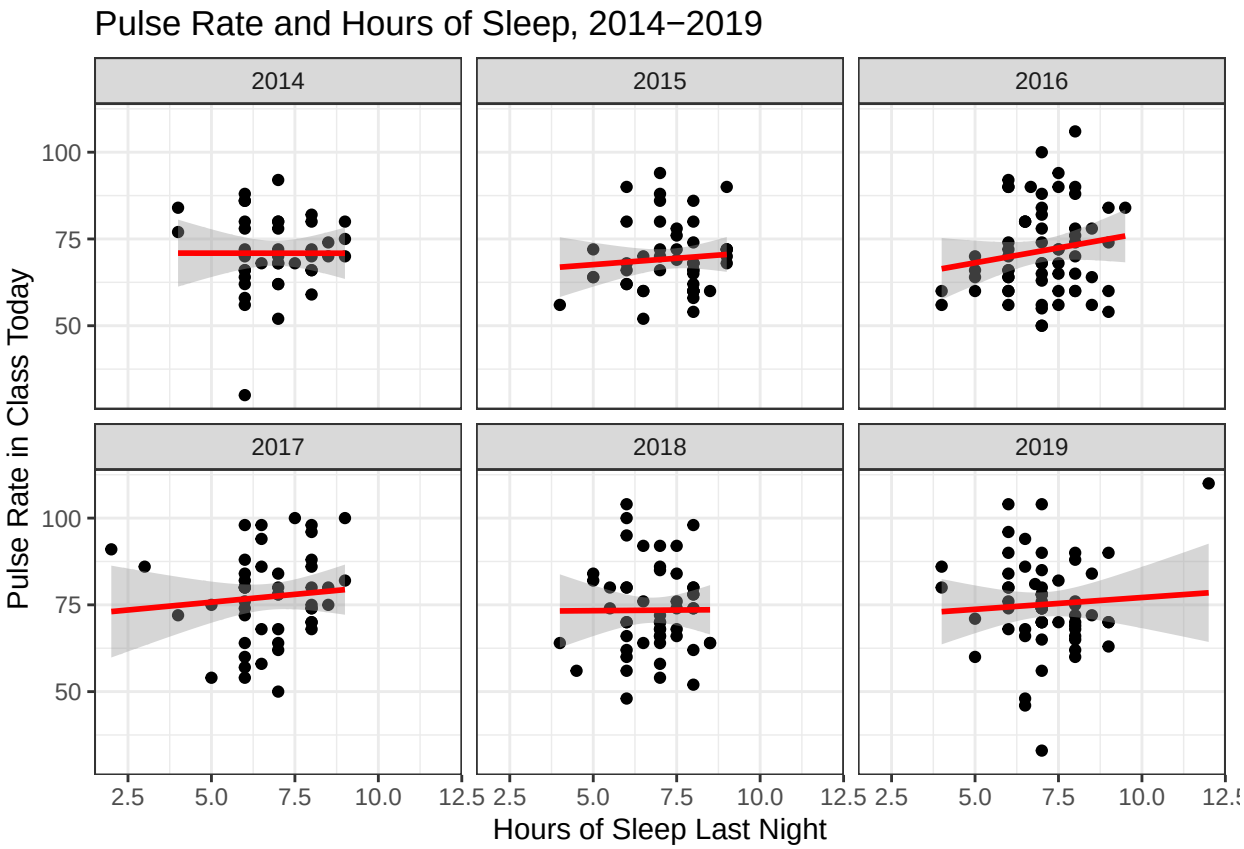


Faceting a Scatterplot

Here's the same basic information, but faceted by year.

```
day1 %>%
    filter(complete.cases(pulse, lastsleep)) %>%
    ggplot(data = ., aes(x = lastsleep, y = pulse,
                                group = factor(year))) +
    geom_point() +
    geom_smooth(method = "lm", color = "red") +
    facet_wrap(~ year) +
    labs(title = "Pulse Rate and Hours of Sleep, 2014-2019",
         x = "Hours of Sleep Last Night",
         y = "Pulse Rate in Class Today") +
    theme_bw()
```

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


Pulse Rate and Hours of Sleep, 2014−2019
2014
2015
2016
2017
2018
2019
Pulse Rate in Class Today
100
75
50
2.5
5.0
7.5
10.0
12.5
Hours of Sleep Last Night