skills_pset3

Rabail Sofi

2023-04-10

```
rm(list=ls())
library(tinytex)
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.1.9000     v readr     2.1.4
## v forcats   1.0.0          v stringr   1.5.0
## v ggplot2   3.4.1          v tibble    3.2.1
## v lubridate 1.9.2          v tidyr     1.3.0
## v purrr     1.0.1
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(devtools)
```

```
## Loading required package: usethis
```

```
library(palmerpenguins)
library(ggthemes)
library(rmarkdown)
library(knitr)
library(dplyr)
library(DescTools)
library(binsreg)
knitr::opts_chunk$set(error = TRUE)
knitr::opts_chunk$set(echo = TRUE)
```

Front matter

This submission is my work alone and complies with the 30535 integrity policy.

Add your initials to indicate your agreement: « RS »

Add your collaborators: «___»

Late coins used this pset: 0. Late coins left: 4.

R for Data Science Exercises

First Steps

Flight Data: Part 1

Download BTS Data

Tidy Data

1.a)

```
# set up to create table3 from the lecture slides
table1 <- table1
table2 <- table1 |>
  pivot_longer(cases:population, names_to = "type", values_to = "count")
table3 <- table2 |>
  pivot_wider(names_from = year, values_from = count)


table3
```

```
## # A tibble: 6 x 4
##   country     type            `1999`     `2000`
##   <chr>       <chr>            <dbl>      <dbl>
## 1 Afghanistan cases              745       2666
## 2 Afghanistan population    19987071   20595360
## 3 Brazil      cases            37737      80488
## 4 Brazil      population   172006362  174504898
## 5 China       cases           212258     213766
## 6 China       population  1272915272 1280428583
```

```
table3_rate <- table3 |>
  pivot_longer(cols = `1999`:`2000`,
               names_to = "year",
               values_to = "number")
table3_rate
```

```
## # A tibble: 12 x 4
##    country     type       year      number
##    <chr>       <chr>      <chr>      <dbl>
##  1 Afghanistan cases      1999         745
##  2 Afghanistan cases      2000        2666
##  3 Afghanistan population 1999    19987071
##  4 Afghanistan population 2000    20595360
##  5 Brazil      cases      1999       37737
##  6 Brazil      cases      2000       80488
##  7 Brazil      population 1999   172006362
##  8 Brazil      population 2000   174504898
##  9 China       cases      1999      212258
## 10 China       cases      2000      213766
## 11 China       population 1999  1272915272
## 12 China       population 2000  1280428583
```

1.b.)

```
table3_rate_ <- table3_rate |>
    pivot_wider(
    names_from = "type",
    values_from = "number")
table3_rate_
```

```
## # A tibble: 6 x 4
##   country     year   cases population
##   <chr>       <chr>  <dbl>      <dbl>
## 1 Afghanistan 1999     745   19987071
## 2 Afghanistan 2000    2666   20595360
## 3 Brazil      1999   37737  172006362
## 4 Brazil      2000   80488  174504898
## 5 China       1999  212258 1272915272
## 6 China       2000  213766 1280428583
```

1.c.)

```
table3_cases_and_rates <- table3_rate_ |>
  mutate(rate = cases / population * 10000)
table3_cases_and_rates
```

```
## # A tibble: 6 x 5
##   country     year   cases population  rate
##   <chr>       <chr>  <dbl>      <dbl> <dbl>
## 1 Afghanistan 1999     745   19987071 0.373
## 2 Afghanistan 2000    2666   20595360 1.29
## 3 Brazil      1999   37737  172006362 2.19
## 4 Brazil      2000   80488  174504898 4.61
## 5 China       1999  212258 1272915272 1.67
## 6 China       2000  213766 1280428583 1.67
```

1.d.)

```
table3_converted_back <- table3_cases_and_rates |>
  pivot_longer(cols = cases:rate, names_to = "type", values_to = "count") |>
  pivot_wider(names_from= year, values_from = count)
table3_converted_back
```

```
## # A tibble: 9 x 4
##   country     type        `1999`       `2000`
##   <chr>       <chr>        <dbl>        <dbl>
## 1 Afghanistan cases      7.45e+2       2666
## 2 Afghanistan population 2.00e+7   20595360
## 3 Afghanistan rate       3.73e-1          1.29
## 4 Brazil      cases      3.77e+4      80488
## 5 Brazil      population 1.72e+8  174504898
## 6 Brazil      rate       2.19e+0          4.61
## 7 China       cases      2.12e+5     213766
## 8 China       population 1.27e+9 1280428583
## 9 China       rate       1.67e+0          1.67
```

1.2)

```
knitr::include_graphics("image")
library(reprex)
reprex(table4a %>% pivot_longer(1999:2000, names_to = "year", values_to = "cases"))
```

1.3) The error is saying that the function "%>%" is not found.

1.4) The following tibble does not work with the pivot_wider() function because the name "Phillip Woods" is assigned to multiple different "age" and "height" variables and this confusion is causing errors.

```
people <- tibble(
~name, ~key, ~value,
#-----------------|--------|------
"Phillip Woods", "age",       45,
"Phillip Woods", "height",   186,
"Phillip Woods","age",       50,
"Phillip Woods", "height",   185,
"Jessica Cordero", "age",       37,
"Jessica Cordero", "height",   156,)

people

people |>
  pivot_wider(names_from = "key", values_from = "value")
```

Tidying case study

2.1) NA refers to missing data, while 0 refers to the data that is equal to 0 in numeric form.

2.2) Without using the mutate function to replace the characters "newrel" with "new_rel", we wouldn't be able to clean up our data correctly. As the chapter suggests, we must replace the characters "newrel" with "new_rel" to keep all the names consistent as seen in the data set "who2".

2.3) a)

```
tidyr::who
```

```
## # A tibble: 7,240 x 60
##    country  iso2  iso3   year new_sp_m014 new_sp_m1524 new_sp_m2534 new_sp_m3544
##    <chr>    <chr> <chr> <dbl>       <dbl>        <dbl>        <dbl>        <dbl>
##  1 Afghani~ AF    AFG    1980          NA           NA           NA           NA
##  2 Afghani~ AF    AFG    1981          NA           NA           NA           NA
##  3 Afghani~ AF    AFG    1982          NA           NA           NA           NA
##  4 Afghani~ AF    AFG    1983          NA           NA           NA           NA
##  5 Afghani~ AF    AFG    1984          NA           NA           NA           NA
##  6 Afghani~ AF    AFG    1985          NA           NA           NA           NA
##  7 Afghani~ AF    AFG    1986          NA           NA           NA           NA
##  8 Afghani~ AF    AFG    1987          NA           NA           NA           NA
##  9 Afghani~ AF    AFG    1988          NA           NA           NA           NA
## 10 Afghani~ AF    AFG    1989          NA           NA           NA           NA
## # i 7,230 more rows
## # i 52 more variables: new_sp_m4554 <dbl>, new_sp_m5564 <dbl>,
```

```
## #   new_sp_m65 <dbl>, new_sp_f014 <dbl>, new_sp_f1524 <dbl>,
## #   new_sp_f2534 <dbl>, new_sp_f3544 <dbl>, new_sp_f4554 <dbl>,
## #   new_sp_f5564 <dbl>, new_sp_f65 <dbl>, new_sn_m014 <dbl>,
## #   new_sn_m1524 <dbl>, new_sn_m2534 <dbl>, new_sn_m3544 <dbl>,
## #   new_sn_m4554 <dbl>, new_sn_m5564 <dbl>, new_sn_m65 <dbl>, ...
```

```
who1 <- who %>%
  pivot_longer(
    cols = new_sp_m014:newrel_f65,
    names_to = "key",
    values_to = "cases",
    values_drop_na = TRUE
  )
who1
```

```
## # A tibble: 76,046 x 6
##    country     iso2  iso3   year key           cases
##    <chr>       <chr> <chr> <dbl> <chr>         <dbl>
##  1 Afghanistan AF    AFG    1997 new_sp_m014       0
##  2 Afghanistan AF    AFG    1997 new_sp_m1524     10
##  3 Afghanistan AF    AFG    1997 new_sp_m2534      6
##  4 Afghanistan AF    AFG    1997 new_sp_m3544      3
##  5 Afghanistan AF    AFG    1997 new_sp_m4554      5
##  6 Afghanistan AF    AFG    1997 new_sp_m5564      2
##  7 Afghanistan AF    AFG    1997 new_sp_m65        0
##  8 Afghanistan AF    AFG    1997 new_sp_f014       5
##  9 Afghanistan AF    AFG    1997 new_sp_f1524     38
## 10 Afghanistan AF    AFG    1997 new_sp_f2534     36
## # i 76,036 more rows
```

```
who2 <- who1 %>%
  mutate(key = stringr::str_replace(key, "newrel", "new_rel"))
who2
```

```
## # A tibble: 76,046 x 6
##    country     iso2  iso3   year key           cases
##    <chr>       <chr> <chr> <dbl> <chr>         <dbl>
##  1 Afghanistan AF    AFG    1997 new_sp_m014       0
##  2 Afghanistan AF    AFG    1997 new_sp_m1524     10
##  3 Afghanistan AF    AFG    1997 new_sp_m2534      6
##  4 Afghanistan AF    AFG    1997 new_sp_m3544      3
##  5 Afghanistan AF    AFG    1997 new_sp_m4554      5
##  6 Afghanistan AF    AFG    1997 new_sp_m5564      2
##  7 Afghanistan AF    AFG    1997 new_sp_m65        0
##  8 Afghanistan AF    AFG    1997 new_sp_f014       5
##  9 Afghanistan AF    AFG    1997 new_sp_f1524     38
## 10 Afghanistan AF    AFG    1997 new_sp_f2534     36
## # i 76,036 more rows
```

```
who3 <- who2 %>%
  separate(key, c("new", "type", "sexage"), sep = "_")
who3
```

```
## # A tibble: 76,046 x 8
##    country     iso2  iso3   year new   type  sexage cases
##    <chr>       <chr> <chr> <dbl> <chr> <chr> <chr>  <dbl>
##  1 Afghanistan AF    AFG    1997 new   sp    m014       0
##  2 Afghanistan AF    AFG    1997 new   sp    m1524     10
##  3 Afghanistan AF    AFG    1997 new   sp    m2534      6
##  4 Afghanistan AF    AFG    1997 new   sp    m3544      3
##  5 Afghanistan AF    AFG    1997 new   sp    m4554      5
##  6 Afghanistan AF    AFG    1997 new   sp    m5564      2
##  7 Afghanistan AF    AFG    1997 new   sp    m65        0
##  8 Afghanistan AF    AFG    1997 new   sp    f014       5
##  9 Afghanistan AF    AFG    1997 new   sp    f1524     38
## 10 Afghanistan AF    AFG    1997 new   sp    f2534     36
## # i 76,036 more rows
```

```
who3 %>%
  count(new)
```

```
## # A tibble: 1 x 2
##   new       n
##   <chr> <int>
## 1 new   76046
```

```
who4 <- who3 %>%
  select(-new, -iso2, -iso3)

who5 <- who4 %>%
  separate(sexage, c("sex", "age"), sep = 1)
who5
```

```
## # A tibble: 76,046 x 6
##    country      year type  sex   age   cases
##    <chr>       <dbl> <chr> <chr> <chr> <dbl>
##  1 Afghanistan  1997 sp    m     014       0
##  2 Afghanistan  1997 sp    m     1524     10
##  3 Afghanistan  1997 sp    m     2534      6
##  4 Afghanistan  1997 sp    m     3544      3
##  5 Afghanistan  1997 sp    m     4554      5
##  6 Afghanistan  1997 sp    m     5564      2
##  7 Afghanistan  1997 sp    m     65        0
##  8 Afghanistan  1997 sp    f     014       5
##  9 Afghanistan  1997 sp    f     1524     38
## 10 Afghanistan  1997 sp    f     2534     36
## # i 76,036 more rows
```

```
who %>%
  pivot_longer(
    cols = new_sp_m014:newrel_f65,
    names_to = "key",
    values_to = "cases",
    values_drop_na = TRUE
  ) %>%
```

```r
  mutate(
    key = stringr::str_replace(key, "newrel", "new_rel")
  ) %>%
  separate(key, c("new", "var", "sexage")) %>%
  select(-new, -iso2, -iso3) %>%
  separate(sexage, c("sex", "age"), sep = 1)
```

```
## # A tibble: 76,046 x 6
##    country      year var   sex   age   cases
##    <chr>       <dbl> <chr> <chr> <chr> <dbl>
##  1 Afghanistan  1997 sp    m     014       0
##  2 Afghanistan  1997 sp    m     1524     10
##  3 Afghanistan  1997 sp    m     2534      6
##  4 Afghanistan  1997 sp    m     3544      3
##  5 Afghanistan  1997 sp    m     4554      5
##  6 Afghanistan  1997 sp    m     5564      2
##  7 Afghanistan  1997 sp    m     65        0
##  8 Afghanistan  1997 sp    f     014       5
##  9 Afghanistan  1997 sp    f     1524     38
## 10 Afghanistan  1997 sp    f     2534     36
## # i 76,036 more rows
```

```r
who5_1 <- who5 |>
  group_by(country, year, sex) |>
  summarize(total = sum(cases))
```

```
## `summarise()` has grouped output by 'country', 'year'. You can override using
## the `.groups` argument.
```

```r
who5_1
```

```
## # A tibble: 6,921 x 4
## # Groups:   country, year [3,484]
##    country      year sex   total
##    <chr>       <dbl> <chr> <dbl>
##  1 Afghanistan  1997 f       102
##  2 Afghanistan  1997 m        26
##  3 Afghanistan  1998 f      1207
##  4 Afghanistan  1998 m       571
##  5 Afghanistan  1999 f       517
##  6 Afghanistan  1999 m       228
##  7 Afghanistan  2000 f      1751
##  8 Afghanistan  2000 m       915
##  9 Afghanistan  2001 f      3062
## 10 Afghanistan  2001 m      1577
## # i 6,911 more rows
```

2.3) Using raw data is probably not going to provide clear evidence because we need to identify patterns and trends in data in order to make interpretations. Moreover, large amounts of raw data can be affected by outliers so its important to summarize the data effectively through visualization and analysis.

2.3) c)

```r
who5_2 <- who5_1 |>
  pivot_wider(names_from = sex, values_from = total) |>
  mutate(male_and_female = f + m) |>
  mutate(female_ratio = f/male_and_female) |>
  mutate(male_ratio = m/male_and_female)
who5_2
```

```
## # A tibble: 3,484 x 7
## # Groups:   country, year [3,484]
##    country      year     f     m male_and_female female_ratio male_ratio
##    <chr>       <dbl> <dbl> <dbl>           <dbl>        <dbl>      <dbl>
##  1 Afghanistan  1997   102    26             128        0.797      0.203
##  2 Afghanistan  1998  1207   571            1778        0.679      0.321
##  3 Afghanistan  1999   517   228             745        0.694      0.306
##  4 Afghanistan  2000  1751   915            2666        0.657      0.343
##  5 Afghanistan  2001  3062  1577            4639        0.660      0.340
##  6 Afghanistan  2002  4418  2091            6509        0.679      0.321
##  7 Afghanistan  2003  4423  2105            6528        0.678      0.322
##  8 Afghanistan  2004  5587  2658            8245        0.678      0.322
##  9 Afghanistan  2005  6818  3131            9949        0.685      0.315
## 10 Afghanistan  2006  8520  3949           12469        0.683      0.317
## # i 3,474 more rows
```

```r
who5_2 <- who5_1 |>
    pivot_wider(names_from = sex, values_from = total) |>
  mutate(male_female_ratio = m/f)
who5_2
```

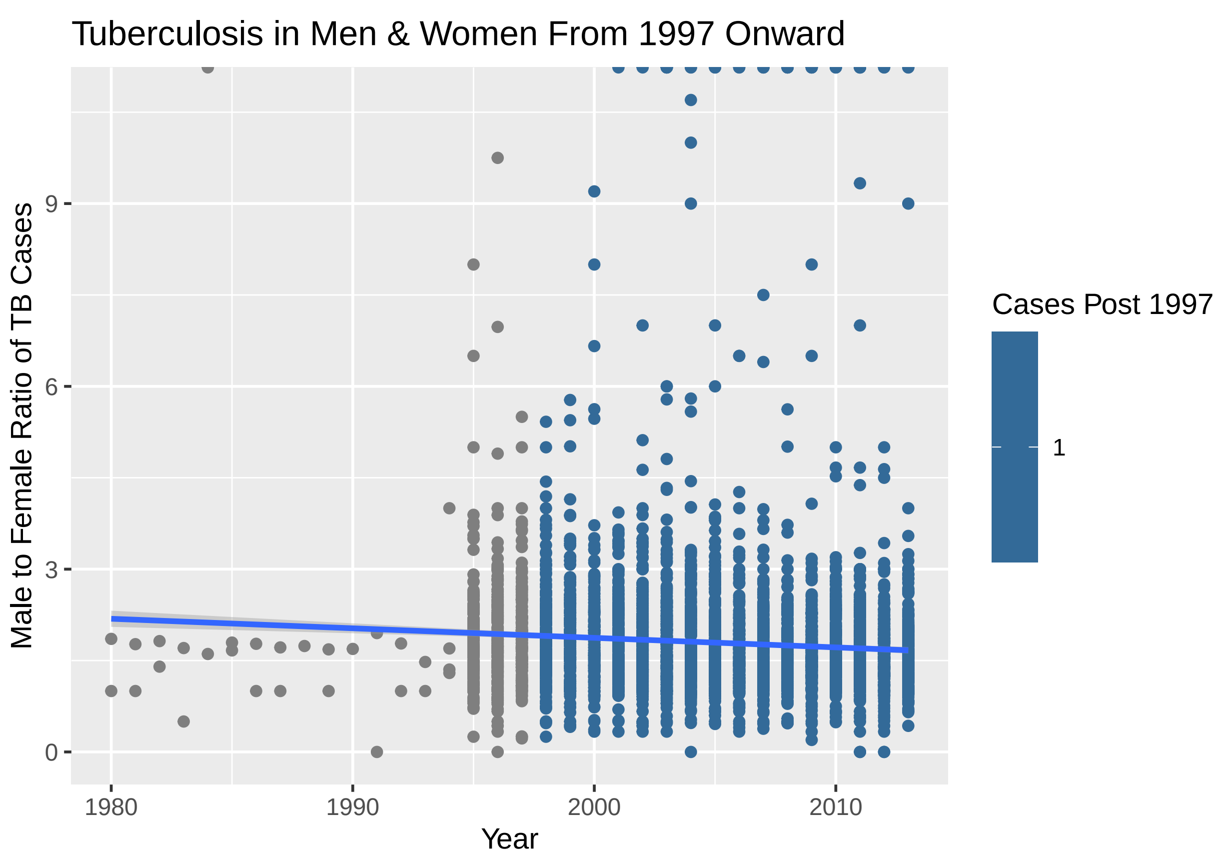
```
## # A tibble: 3,484 x 5
## # Groups:   country, year [3,484]
##    country      year     f     m male_female_ratio
##    <chr>       <dbl> <dbl> <dbl>             <dbl>
##  1 Afghanistan  1997   102    26             0.255
##  2 Afghanistan  1998  1207   571             0.473
##  3 Afghanistan  1999   517   228             0.441
##  4 Afghanistan  2000  1751   915             0.523
##  5 Afghanistan  2001  3062  1577             0.515
##  6 Afghanistan  2002  4418  2091             0.473
##  7 Afghanistan  2003  4423  2105             0.476
##  8 Afghanistan  2004  5587  2658             0.476
##  9 Afghanistan  2005  6818  3131             0.459
## 10 Afghanistan  2006  8520  3949             0.463
## # i 3,474 more rows
```

2.3) d) It is a bad idea to ignore “country” when computing ratios because the men to women ratios can differ greately in countries across years that have severe gender inequality, lack of access to medical facilities for either gender, or other health concerns that effect one gender disproportionately. Hence, including the years and countries for these ratios help paint a better picture of TB cases and how they effect the respective populations based on what part of the world they live in.

2.3) e)

```
who5_2 |>
  mutate(post97 = if_else(year < 1998, NA, 1)) |>
  ggplot(aes(x = year, y = male_female_ratio, color = post97)) +
  geom_point() + geom_smooth(method = "lm") +
  labs(title = "Tuberculosis in Men & Women From 1997 Onward",
       x = "Year", y = "Male to Female Ratio of TB Cases",
        color = "Cases Post 1997")
```

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

## Warning: Removed 136 rows containing non-finite values (`stat_smooth()`).

## Warning: The following aesthetics were dropped during statistical transformation: colour
## i This can happen when ggplot fails to infer the correct grouping structure in
##   the data.
## i Did you forget to specify a `group` aesthetic or to convert a numerical
##   variable into a factor?

## Warning: Removed 95 rows containing missing values (`geom_point()`).
```



2.3) f) I learned that it is crucial to include concise and specific aesthetics for summarizing raw data to avoid confusion. Adding arguments to address the Q speficially (such as creating a variable just for cases post 1997) is important because those specific additions will help explain the plot more effectively. Although adding the variable "country" was not needed, I struggled with added countries to the plot because there were many countries in the data frame and it was difficult for me to organize them without making the plot unreadable.

The title for this would be: Men see a significant increase in TB cases as compared to women post 1997.

Sub points: After the year 1997, the cases of TB increased significantly for men. By filtering analyzing the data, one can infer that men in lower income countries were more impacted by TB than women. The reason could be that men are more exposed to TB-causing agents than women in those countries or that men are disadvantaged from seeking medical help for TB.

Citation: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5012571/ & https://jrnold.github.io/r4ds-exercise-solutions/factors.html

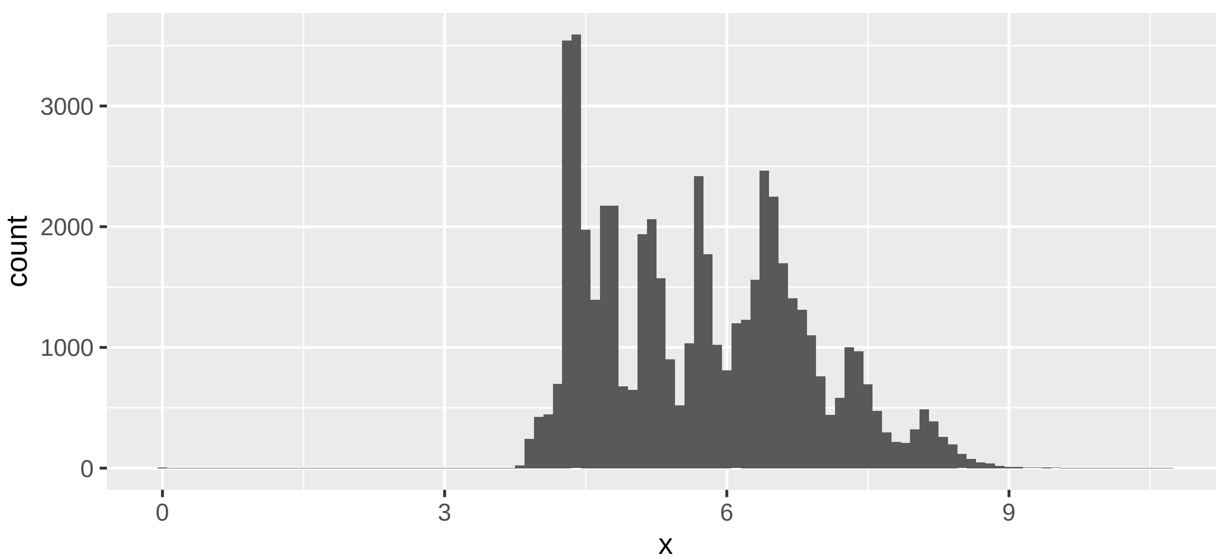
EDA (Exploring variation, including NAs)

3.1) X's distribution is between -0.25 and 10.8 and the variable surpasses a count of about 3,000.

Y's distribution ranges between -0.25 and 59.2 with majority of the distribution being between 3.5 and 7.5. The total count for this variable surpasses 3,000 as well.

Z's distribution ranges between -0.25 and 32, with most of the distribution being between 2.25 and 5.7. The total count for this variable surpasses 6,000.

```
ggplot(diamonds, aes(x = x)) +
  geom_histogram(binwidth = 0.1)
```

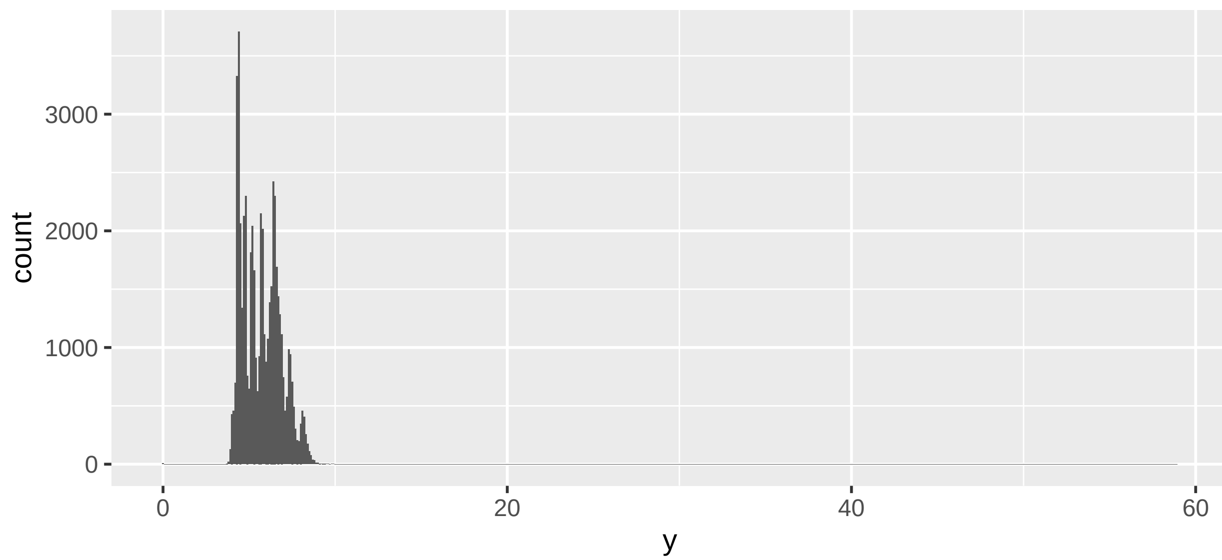



```
diamonds |>
  count(cut_width(x, 0.5))
```

```
## # A tibble: 16 x 2
##    `cut_width(x, 0.5)`     n
##    <fct>               <int>
##  1 [-0.25,0.25]            8
##  2 (3.25,3.75]             3
##  3 (3.75,4.25]          1834
##  4 (4.25,4.75]         12680
```

```
##  5 (4.75,5.25]          7502
##  6 (5.25,5.75]          6448
##  7 (5.75,6.25]          6031
##  8 (6.25,6.75]          9381
##  9 (6.75,7.25]          4193
## 10 (7.25,7.75]          3437
## 11 (7.75,8.25]          1620
## 12 (8.25,8.75]           699
## 13 (8.75,9.25]            79
## 14 (9.25,9.75]            18
## 15 (9.75,10.2]             6
## 16 (10.2,10.8]             1
```

```
ggplot(diamonds, aes(x = y)) +
  geom_histogram(binwidth = 0.1)
```

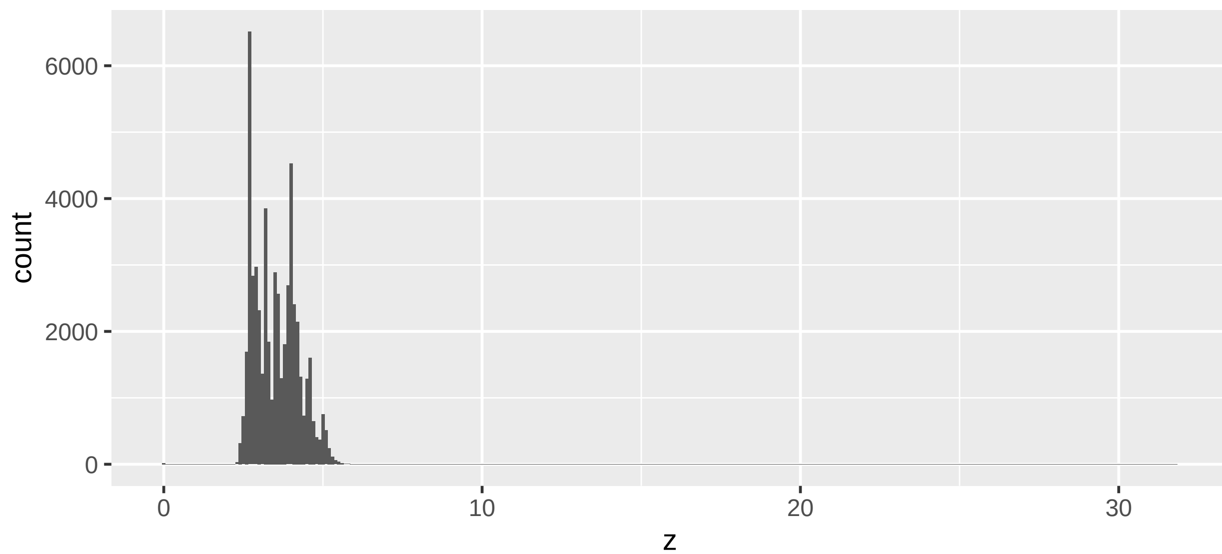



```
diamonds |>
  count(cut_width(y, 0.5))
```

```
## # A tibble: 18 x 2
##    `cut_width(y, 0.5)`     n
##    <fct>               <int>
##  1 [-0.25,0.25]            7
##  2 (3.25,3.75]             6
##  3 (3.75,4.25]          1730
##  4 (4.25,4.75]         12566
##  5 (4.75,5.25]          7556
##  6 (5.25,5.75]          6272
##  7 (5.75,6.25]          6464
##  8 (6.25,6.75]          9382
##  9 (6.75,7.25]          4176
## 10 (7.25,7.75]          3425
## 11 (7.75,8.25]          1612
## 12 (8.25,8.75]           654
```

```
## 13 (8.75,9.25]            67
## 14 (9.25,9.75]            14
## 15 (9.75,10.2]             6
## 16 (10.2,10.8]             1
## 17 (31.8,32.2]             1
## 18 (58.8,59.2]             1
```

```r
ggplot(diamonds, aes(x = z)) +
  geom_histogram(binwidth = 0.1)
```

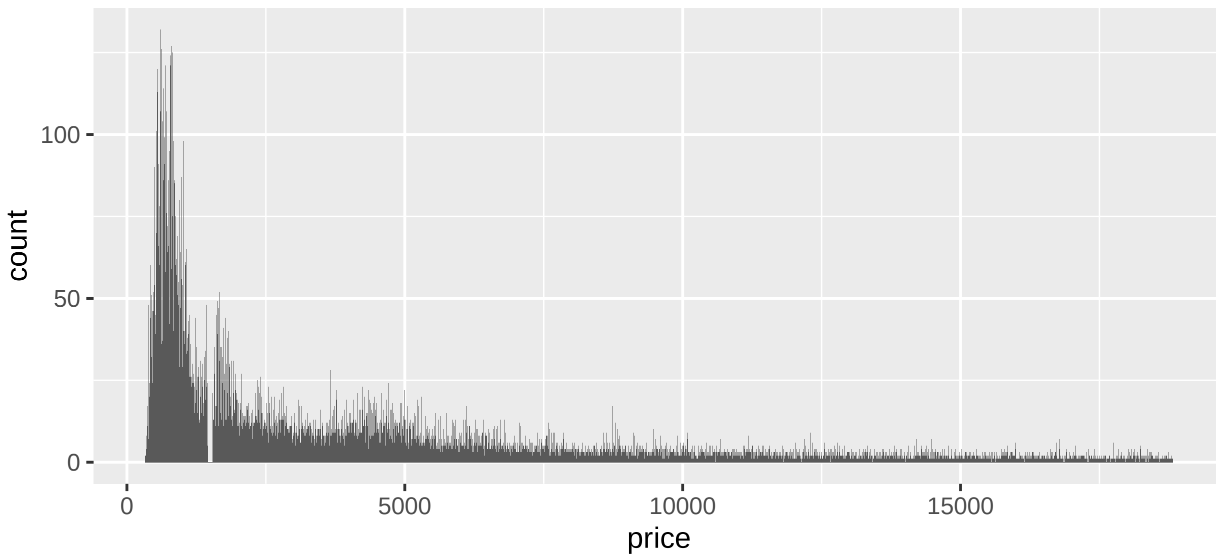



```r
diamonds |>
  count(cut_width(z, 0.5))
```

```
## # A tibble: 16 x 2
##    `cut_width(z, 0.5)`     n
##    <fct>               <int>
##  1 [-0.25,0.25]           20
##  2 (0.75,1.25]             1
##  3 (1.25,1.75]             2
##  4 (1.75,2.25]             3
##  5 (2.25,2.75]          9276
##  6 (2.75,3.25]         13340
##  7 (3.25,3.75]          9572
##  8 (3.75,4.25]         13584
##  9 (4.25,4.75]          5589
## 10 (4.75,5.25]          2288
## 11 (5.25,5.75]           238
## 12 (5.75,6.25]            19
## 13 (6.25,6.75]             5
## 14 (6.75,7.25]             1
## 15 (7.75,8.25]             1
## 16 (31.8,32.2]             1
```

3.2) As we try various different bin_widths, we can see that the default bin-width of 0.5, the price looks like it gradually decreases as count increases. However, once we increase the bin-width, we squint less and we can get better insight into the distribution of price in the data set and see that there is a dip before focal numbers and bulge after focal numbers. Since price is a continuous variable, adjusting bin-width allows us to get more clarity on the correlation between diamond size and price. As discussed in lecture, the different ranges in bin-width raises insight into marketing (people will buy more at a certain point of sale, consumer preferences, etc.) about a pattern of people buying more diamonds right on the cusp.

```
diamonds <- diamonds

ggplot(diamonds) +
  geom_histogram(aes(x = price), binwidth = 0.5)
```



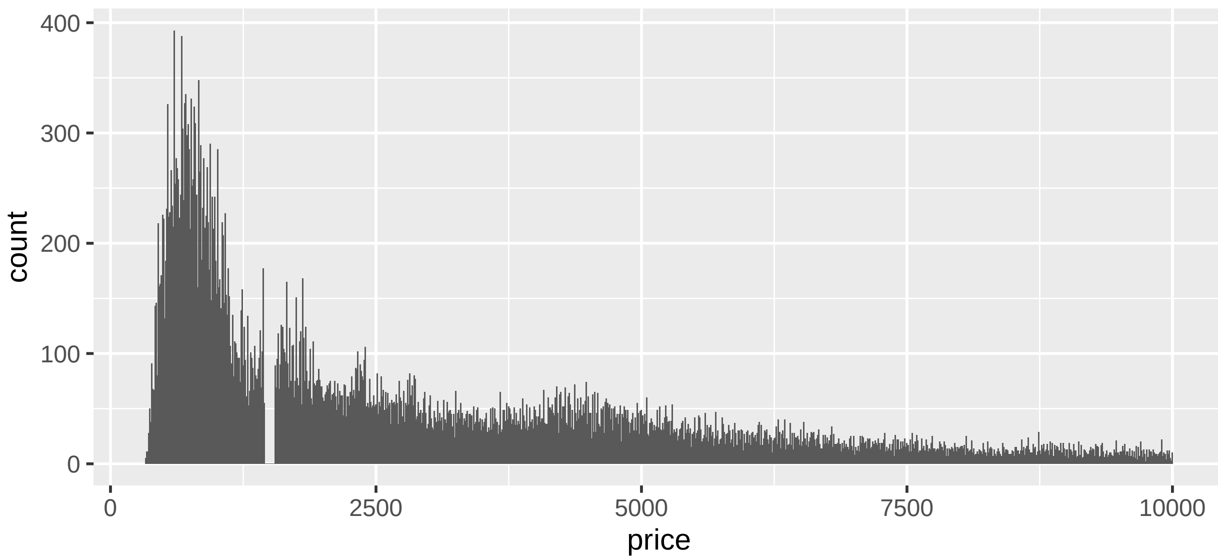

```
diamonds |>
  count(cut_width(price, 0.5))
```

```
## # A tibble: 11,602 x 2
##    `cut_width(price, 0.5)`     n
##    <fct>                   <int>
##  1 [325.75,326.25]             2
##  2 (326.75,327.25]             1
##  3 (333.75,334.25]             1
##  4 (334.75,335.25]             1
##  5 (335.75,336.25]             2
##  6 (336.75,337.25]             2
##  7 (337.75,338.25]             1
##  8 (338.75,339.25]             1
##  9 (339.75,340.25]             1
## 10 (341.75,342.25]             1
## # i 11,592 more rows
```

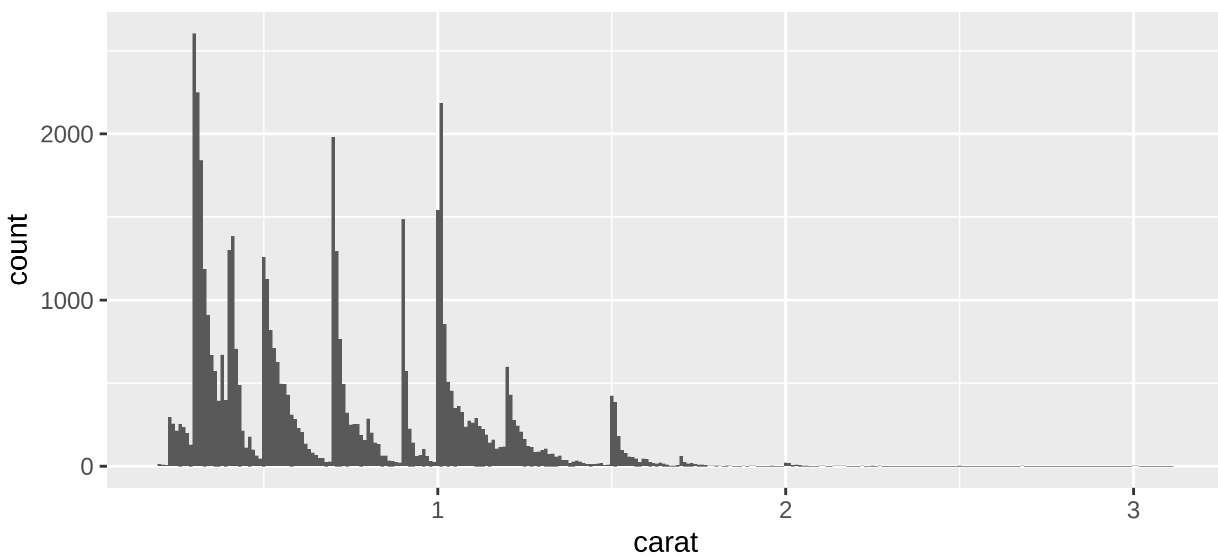
```
mean(diamonds$price)
```

```
## [1] 3932.8
```

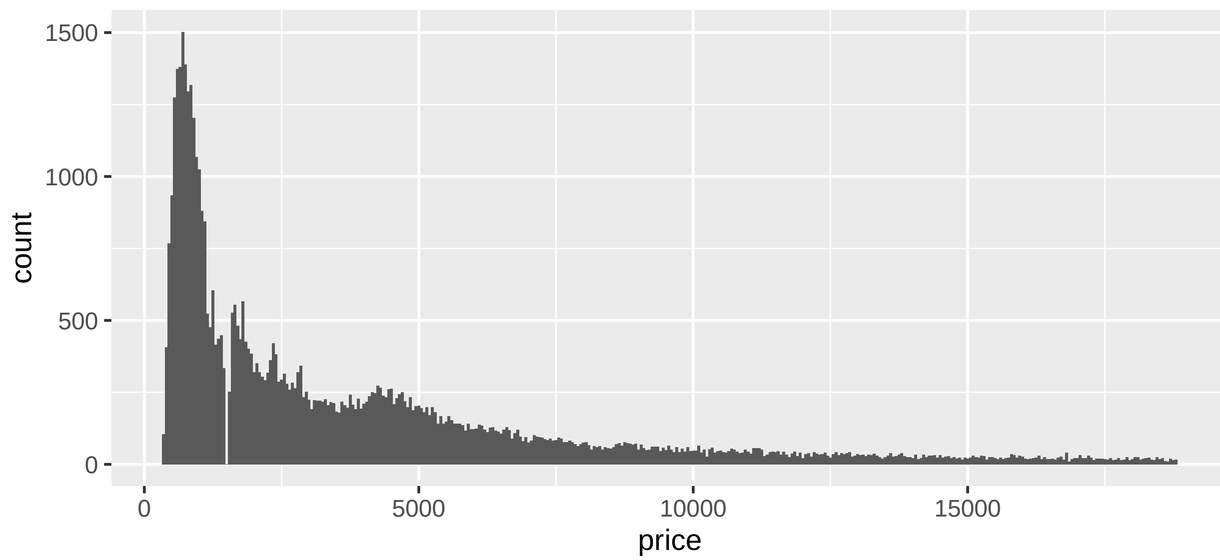
```
diamonds_small <-
  diamonds |>
  filter(price < 10000)

ggplot(diamonds_small) +
  geom_histogram(aes(x = price), binwidth = 10)
```



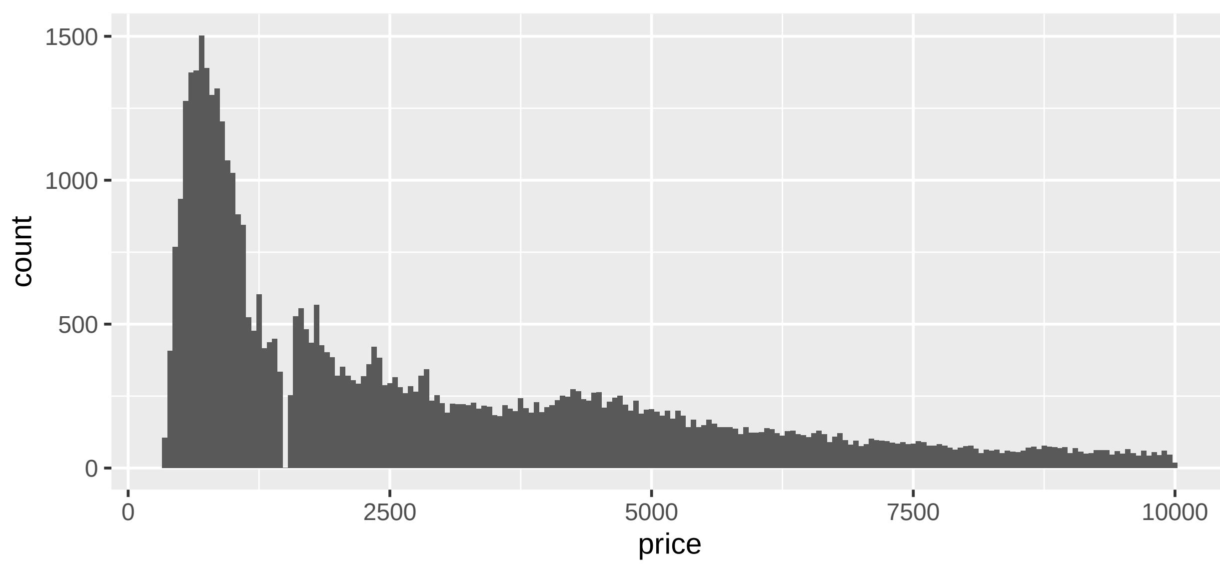

```
ggplot(diamonds_small, aes(x = carat)) +
  geom_histogram(binwidth = 0.01)
```




```
ggplot(diamonds) +
  geom_histogram(aes(x = price), binwidth = 50)
```




```
ggplot(diamonds_small) +
  geom_histogram(aes(x = price), binwidth = 50)
```



3.3) - In a bar chart of a factor variable, missing values are usually treated as a separate category and are shown as an additional bar on the plot, often labeled as "NA". This can be seen in the plot below where there is a new bar titled "NA" for the penguin data frame.

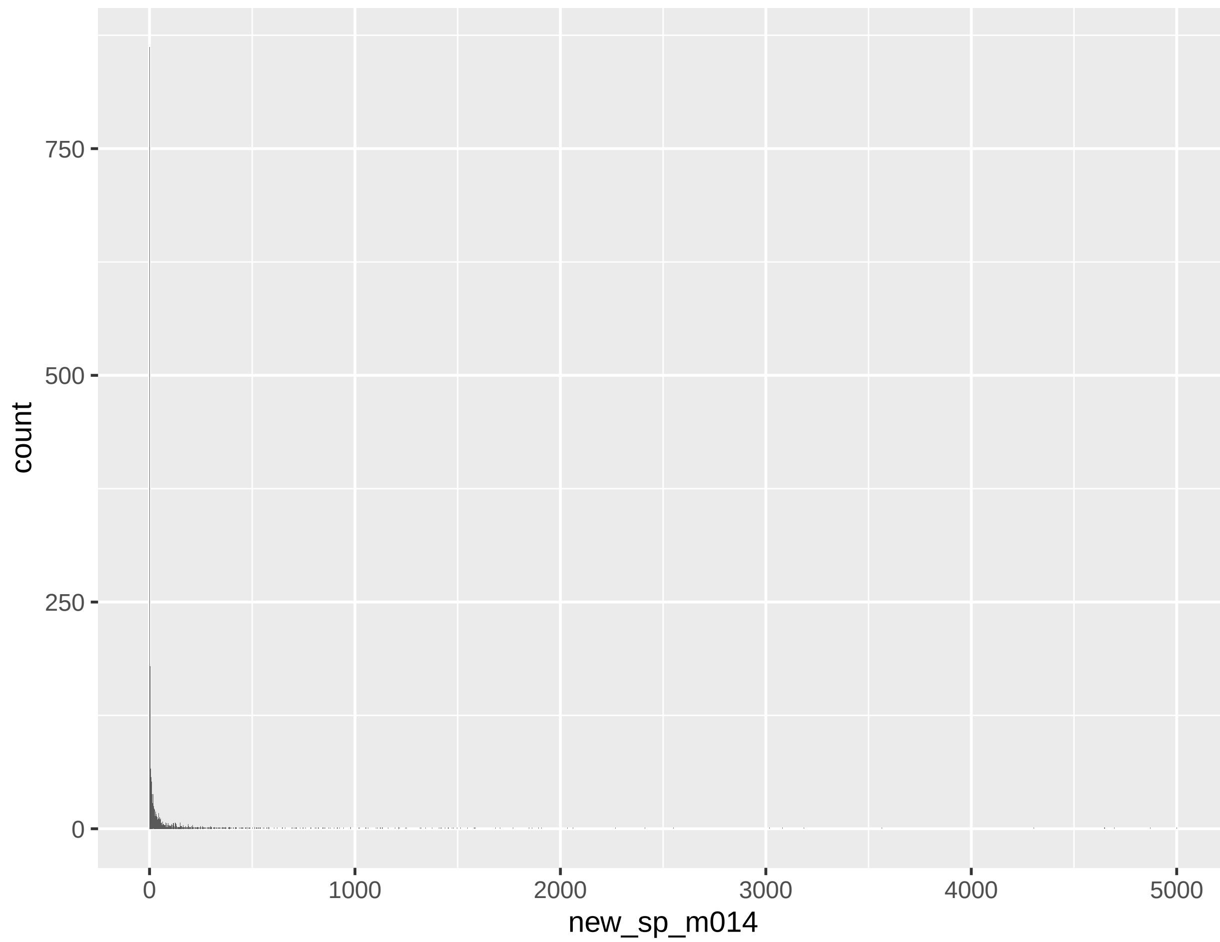
- In a bar chart of a numeric variable, missing values are typically excluded from the plot entirely, and only the non-missing values are displayed. This can be seen in the plot below where the missing values are entirely excluded for the numeric variable chart.
- The difference in missing values between factor and numeric variables is due to the nature of the variables. Factors represent discrete categories, and missing values are treated as a distinct category. In contrast, numeric variables represent continuous values, and missing values do not have a meaningful value to display in a chart and this is the default behavior in R.

Source: help from chatGPT

```
who <- who

### Missing values in a bar chart of a numeric variable:
ggplot(who, aes(new_sp_m014))+
  geom_bar(binwidth = 200)
```

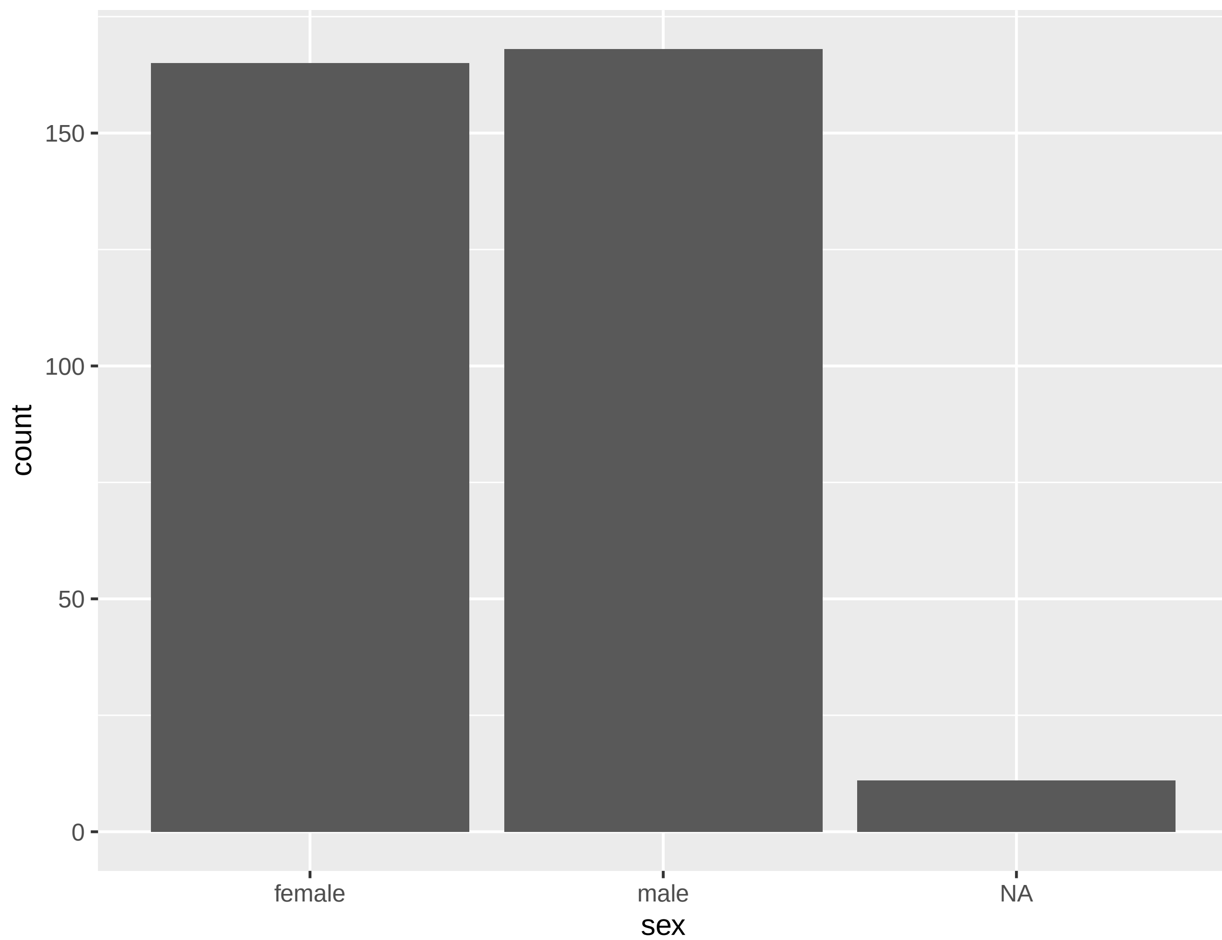
```
## Warning in geom_bar(binwidth = 200): Ignoring unknown parameters: ‘binwidth‘
```

```
## Warning: Removed 4067 rows containing non-finite values (‘stat_count()‘).
```




```
### Missing values in a bar chart of a factor variable:
penguins <- penguins
ggplot(penguins, aes(sex))+
  geom_bar(binwidth = 200)
```

```
## Warning in geom_bar(binwidth = 200): Ignoring unknown parameters: ‘binwidth‘
```

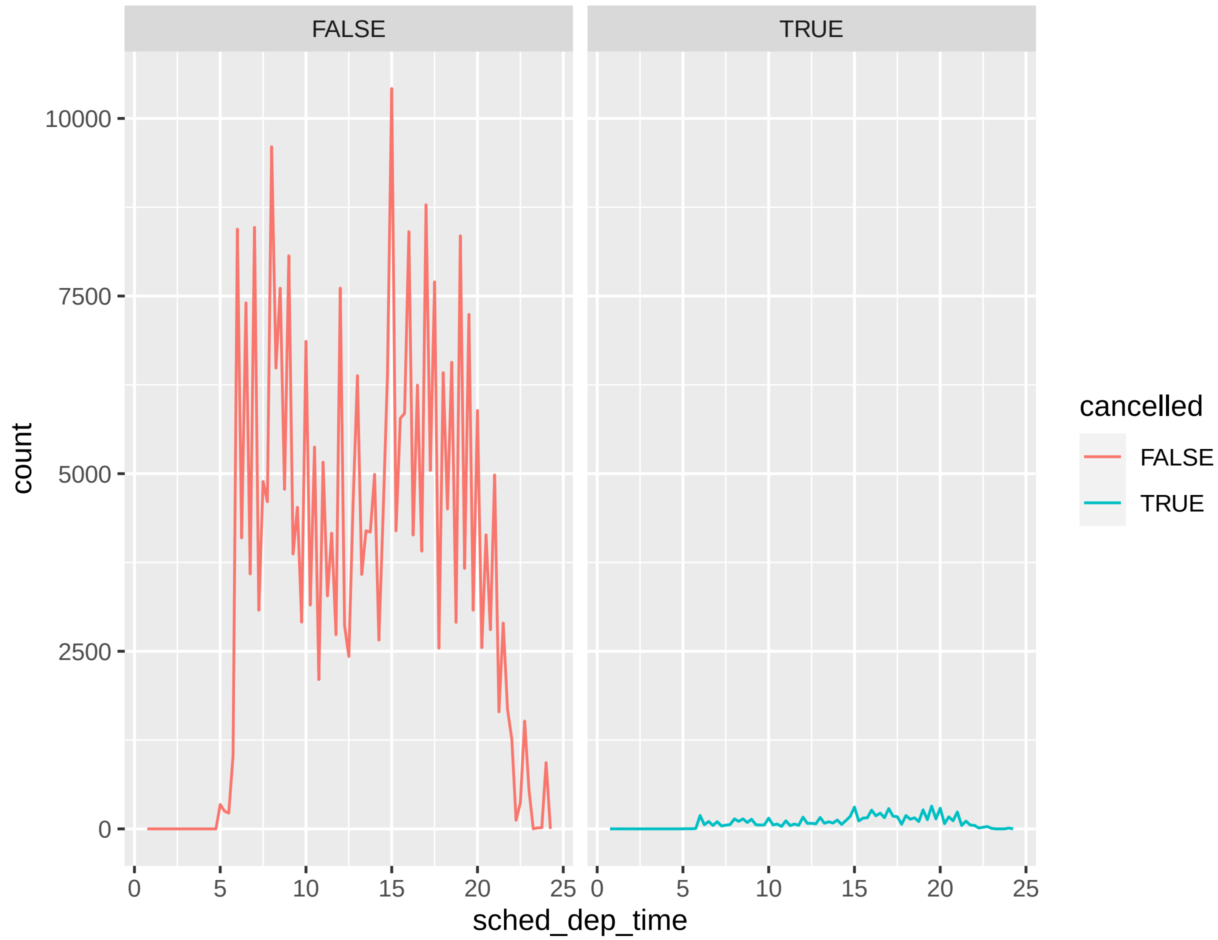


EDA: Compare two distributions

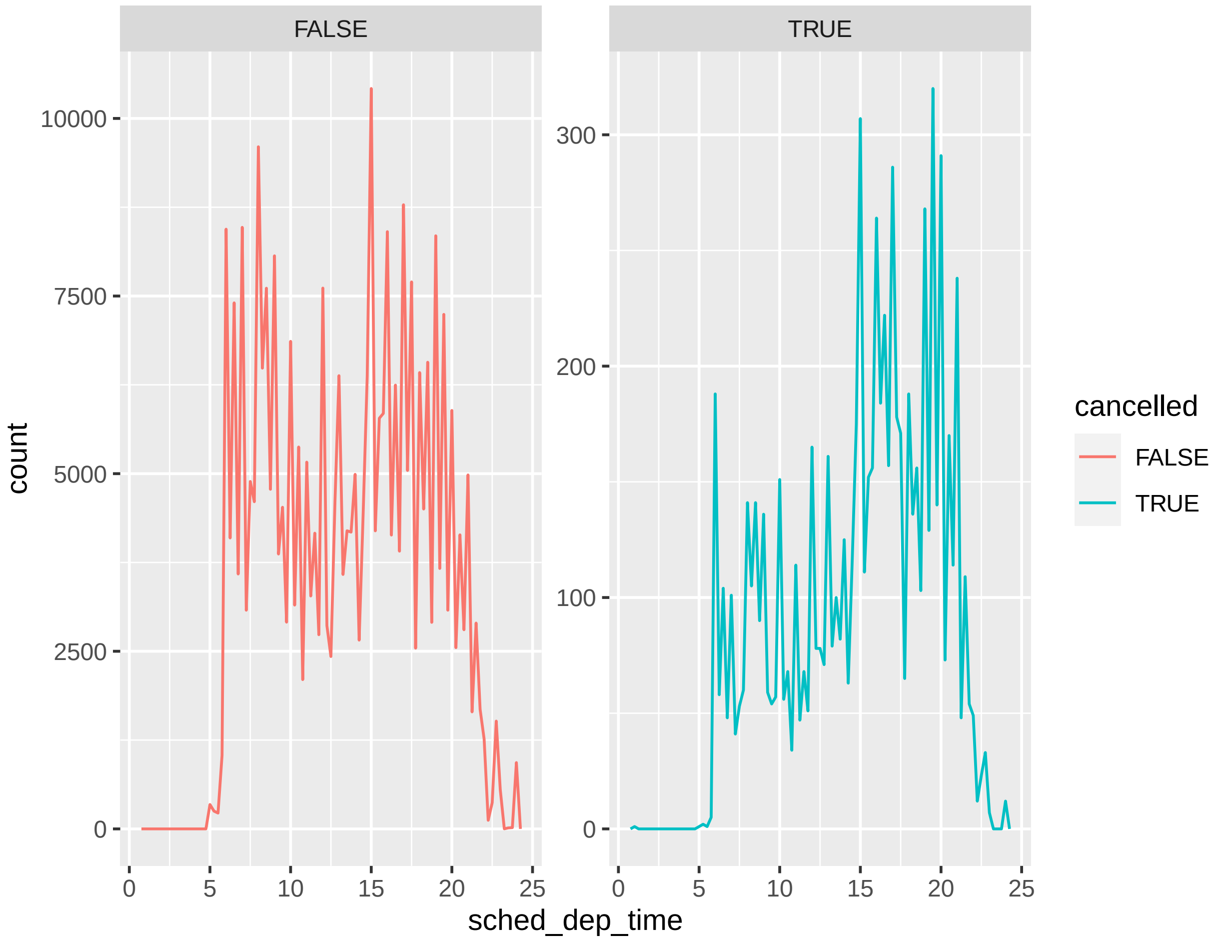
4.1) In R, the scale() function is used to standardize or normalize data. Standardization involves transforming the data so that it has a mean of 0 and a standard deviation of 1, while normalization involves scaling the data to a specified range, such as between 0 and 1.

In the first graph, with the facet_wrap (~cancelled) we can use it to assess the total count of flights that were actually cancelled. In the second graph with the facet_wrap(~cancelled, scales = "free_y"), we can use that to assess what what going on at different times for flights that were cancelled as well as flights that did make it.

```
library(ggplot2)
nycflights13::flights |>
  mutate(
    cancelled = is.na(dep_time),
    sched_hour = sched_dep_time %/% 100,
    sched_min = sched_dep_time %% 100,
    sched_dep_time = sched_hour + (sched_min / 60)
  ) |>
  ggplot(aes(x = sched_dep_time)) +
  geom_freqpoly(aes(color = cancelled), binwidth = 1/4) +
  facet_wrap (~cancelled)
```



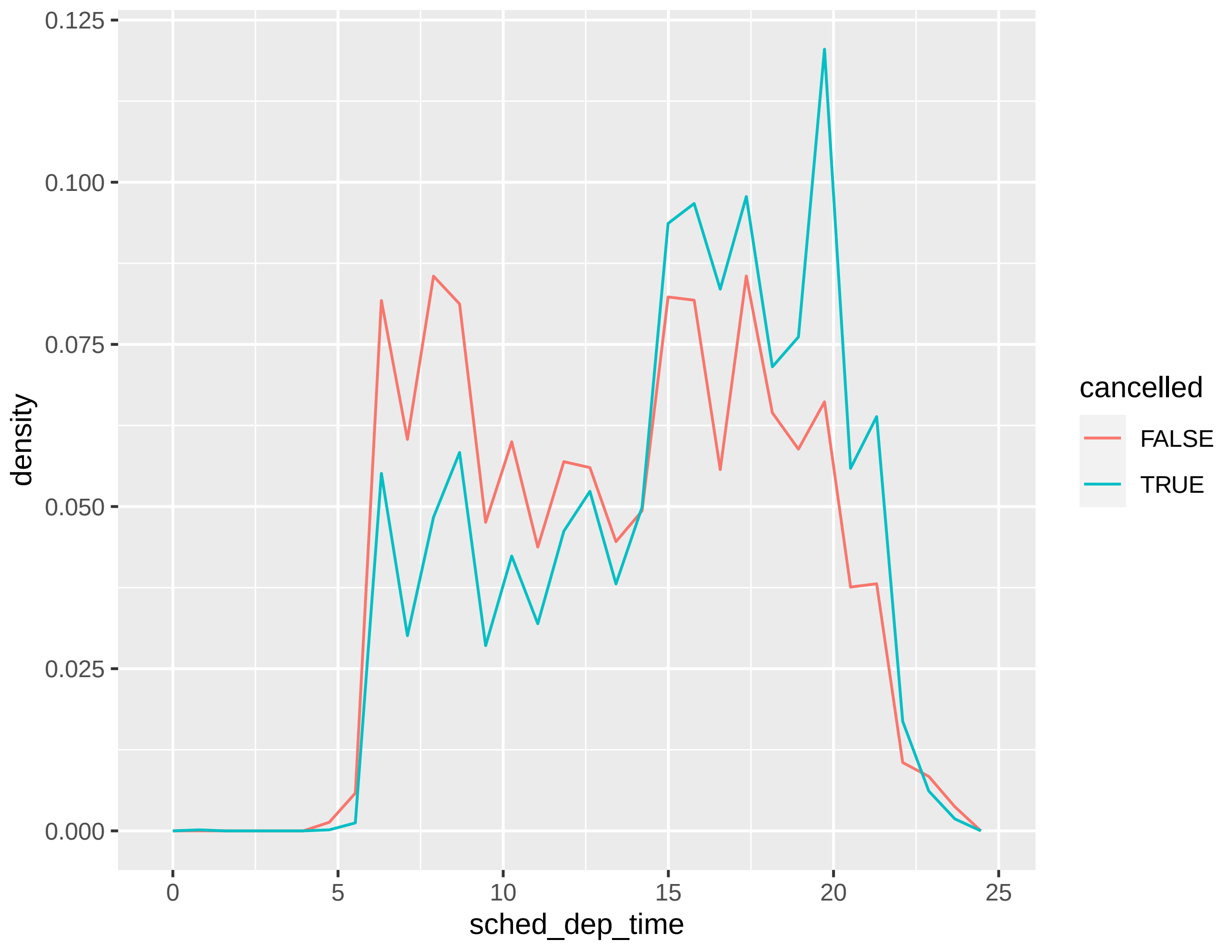

```r
nycflights13::flights |>
  mutate(
    cancelled = is.na(dep_time),
    sched_hour = sched_dep_time %/% 100,
    sched_min = sched_dep_time %% 100,
    sched_dep_time = sched_hour + (sched_min / 60)
  ) |>
  ggplot(aes(x = sched_dep_time)) +
  geom_freqpoly(aes(color = cancelled), binwidth = 1/4) +
  facet_wrap(~cancelled, scales = "free_y")
```



4.2)

```
nycflights13::flights |>
  mutate(
    cancelled = is.na(dep_time),
    sched_hour = sched_dep_time %/% 100,
    sched_min = sched_dep_time %% 100,
    sched_dep_time = sched_hour + (sched_min / 60)
  ) |>
  ggplot(aes(x = sched_dep_time, y = after_stat(density))) +
  geom_freqpoly(aes(color = cancelled))
```

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



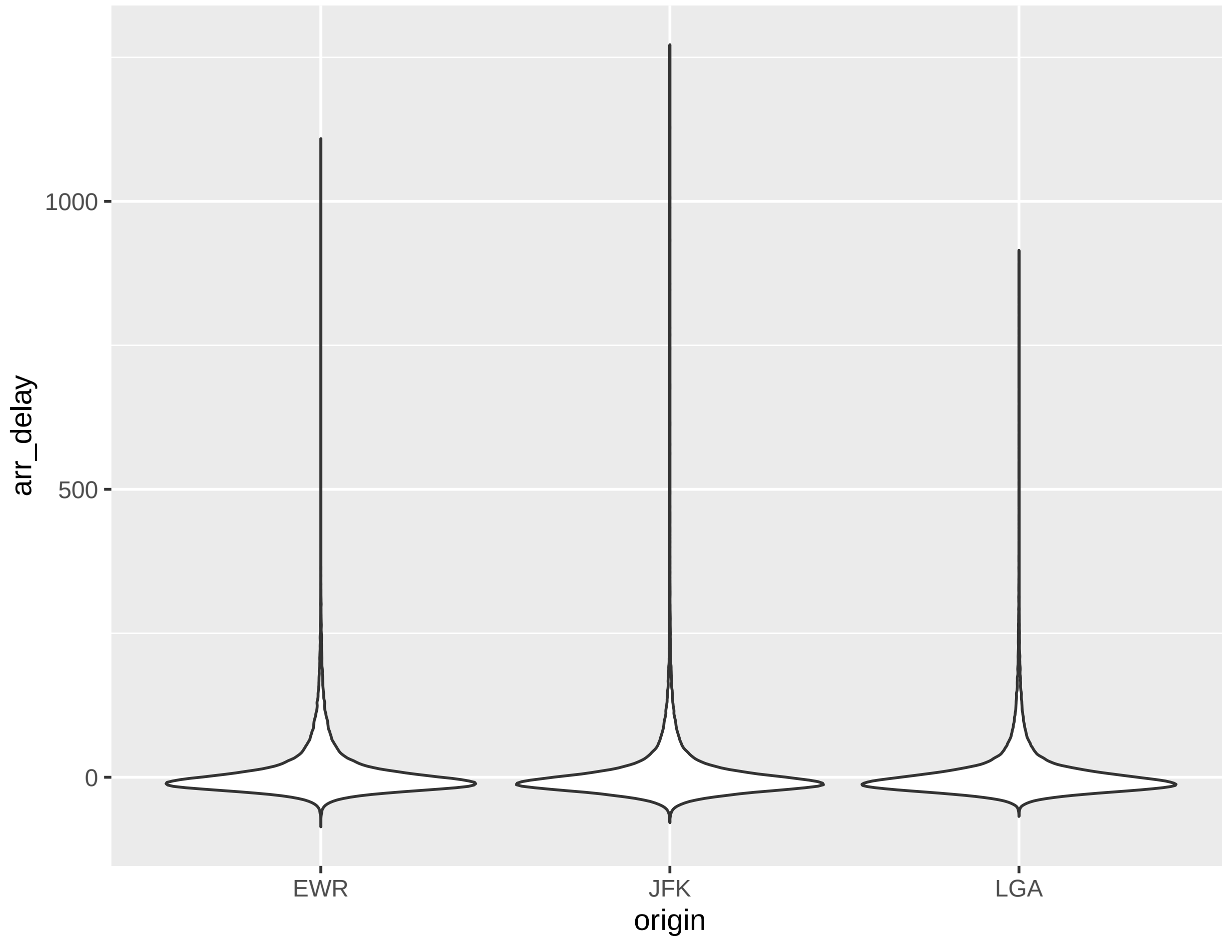
4.3) Geom_freqpoly: this is best for a quick look up. Given the scheduled departure time, you can easily tell which origin is best. The colors make the plot easy to read as well! The downside is that the lines in the plot overlap and we don't have much insight into distribution and how the categorical variables affect each other.

Geom_violin: provides a visual distribution of each variable and shows the density of the data well, but the plot layout is a bit odd looking and might confuse an ordinary person who is not familiar with R's geom_violin plots.

Geom_histogram: Can visualize the distribution of a single variable, making it a useful tool for exploring the shape of the data. It allows for the specification of bins, making it possible to control the level of detail in the histogram. The downside here is that it can be sensitive to the choice of bin size and bin width and it is difficult to compare the data across bin-widths.

```
nycflights13::flights |>
  ggplot(aes(x = origin, y = arr_delay)) +
  geom_violin()
```

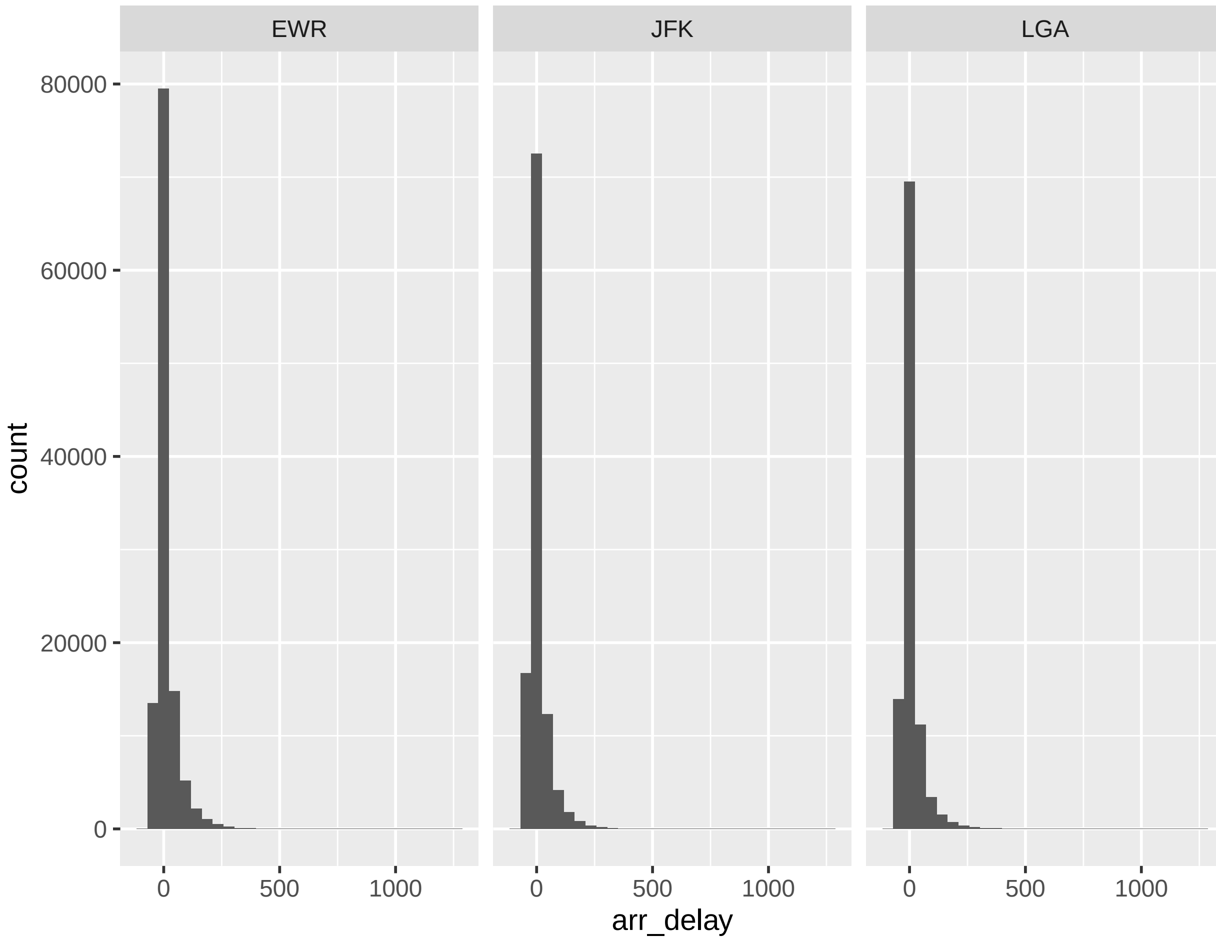
```
## Warning: Removed 9430 rows containing non-finite values (`stat_ydensity()`).
```



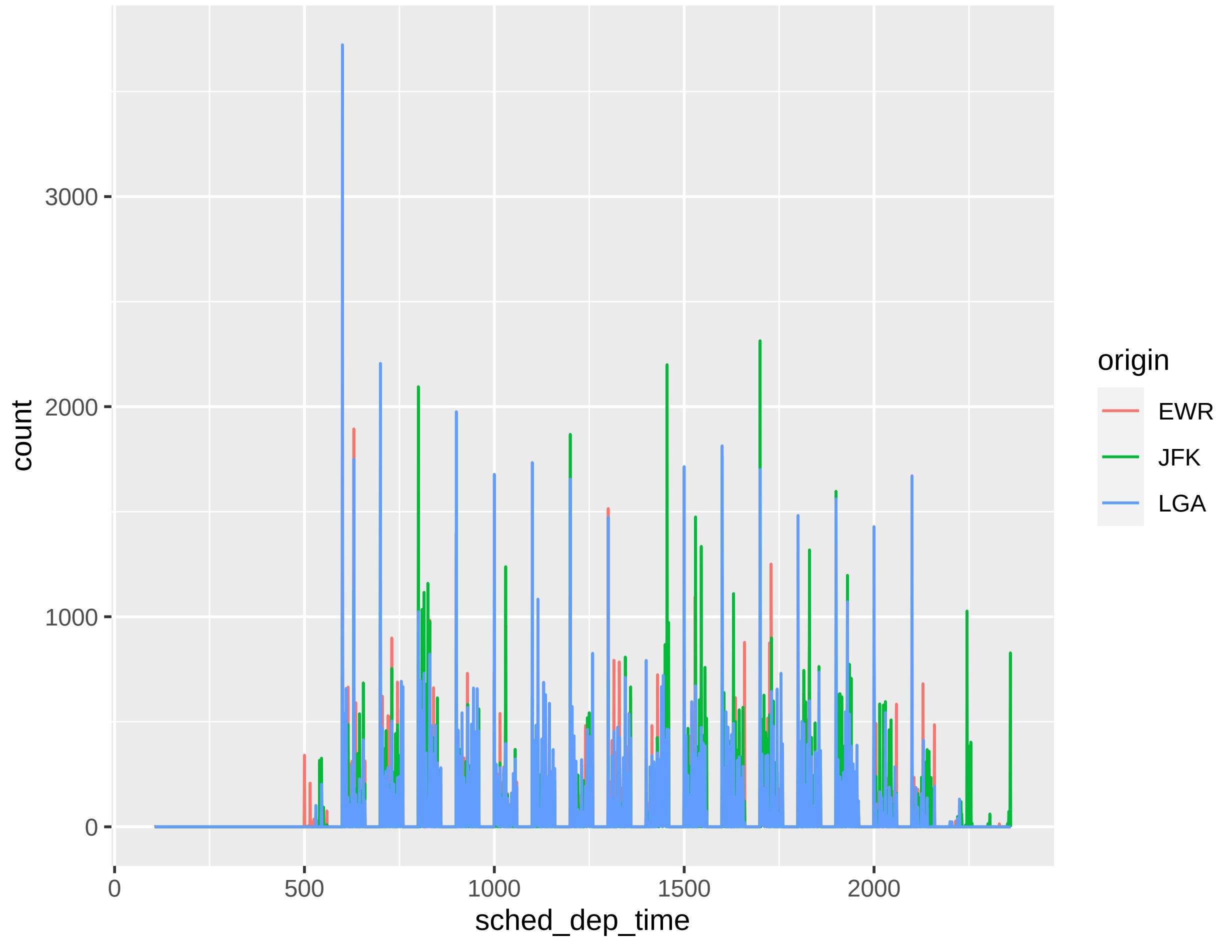

```
nycflights13::flights |>
ggplot(mapping = aes(x = arr_delay)) +
  geom_histogram() + facet_wrap(~origin)
```

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

```
## Warning: Removed 9430 rows containing non-finite values (`stat_bin()`).
```

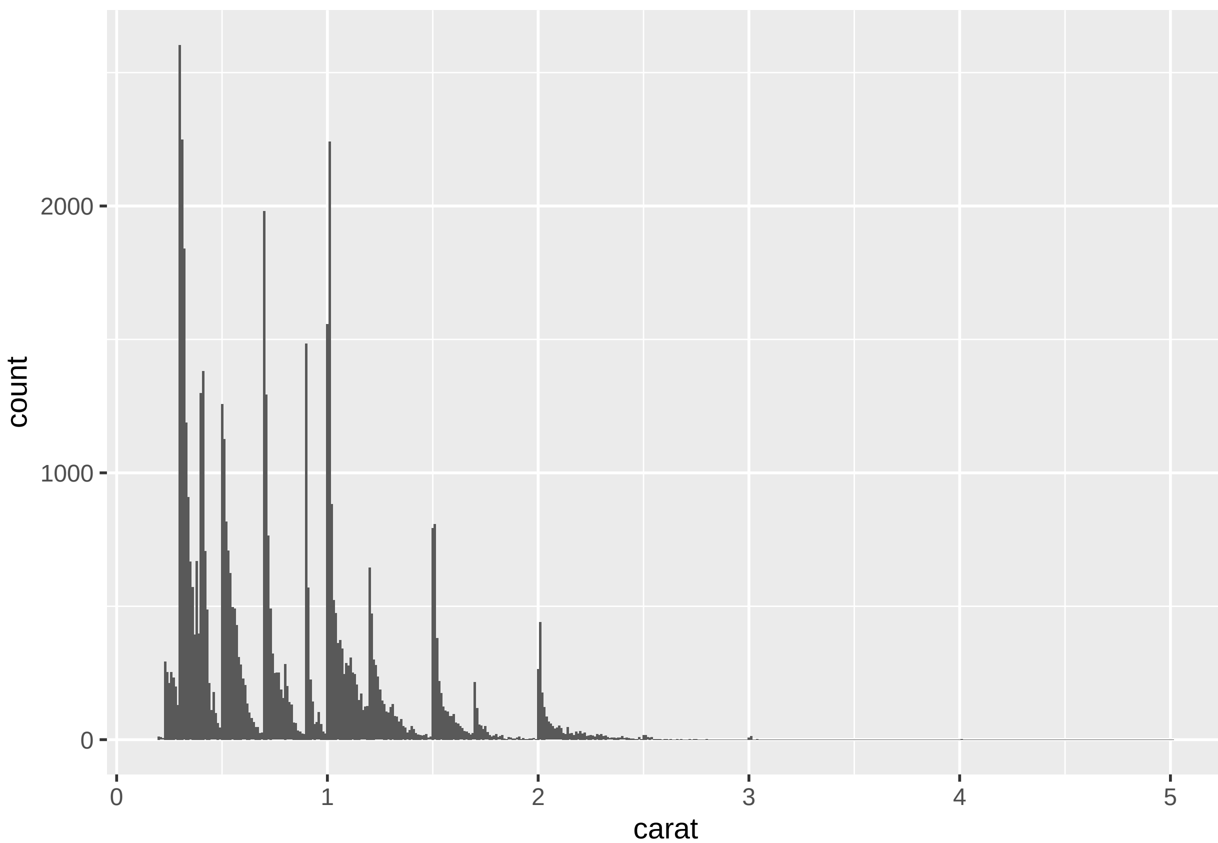



```
nycflights13::flights |>
  ggplot(aes(sched_dep_time)) +
  geom_freqpoly(aes(colour = origin), binwidth = 1)
```



4.4) The distribution among small diamonds has much more range is very vast. On the other hand, the distribution among large diamonds is much smaller. There is a significant cut off at 2.1 carats. I found this understandable because of consumer behavior and buying patterns; most people will invest in smaller diamonds with smaller carats as opposed to larger diamonds with larger carrots. To accommodate for this pattern, diamond producing companies see benefit in adding more range to diamonds with fewer carats since they are MUCH more common.

```
ggplot(diamonds, aes(x = carat)) +
  geom_histogram(binwidth = 0.01)
```



EDA: Covariation

5.1) Despite spending lots of time adjusting height and width, this graph is hard to read because of the number of variables on the X axis. Because there are so many destinations in the data frame, it is not easy to compute all the destinations in one plot. I would suggest grouping the destinations into regions to make the graph easier to read and comprehend.

```
flights |>
  group_by(month, dest) |>
  mutate(total_ave_delay = mean(arr_delay + dep_delay, na.rm=TRUE)) |>
  ggplot(aes(x=month, y=dest)) +
  geom_tile(aes(fill = total_ave_delay)) +
  scale_x_continuous(breaks = 1:12) +
  labs(x= "Month", y = "Destination")
```

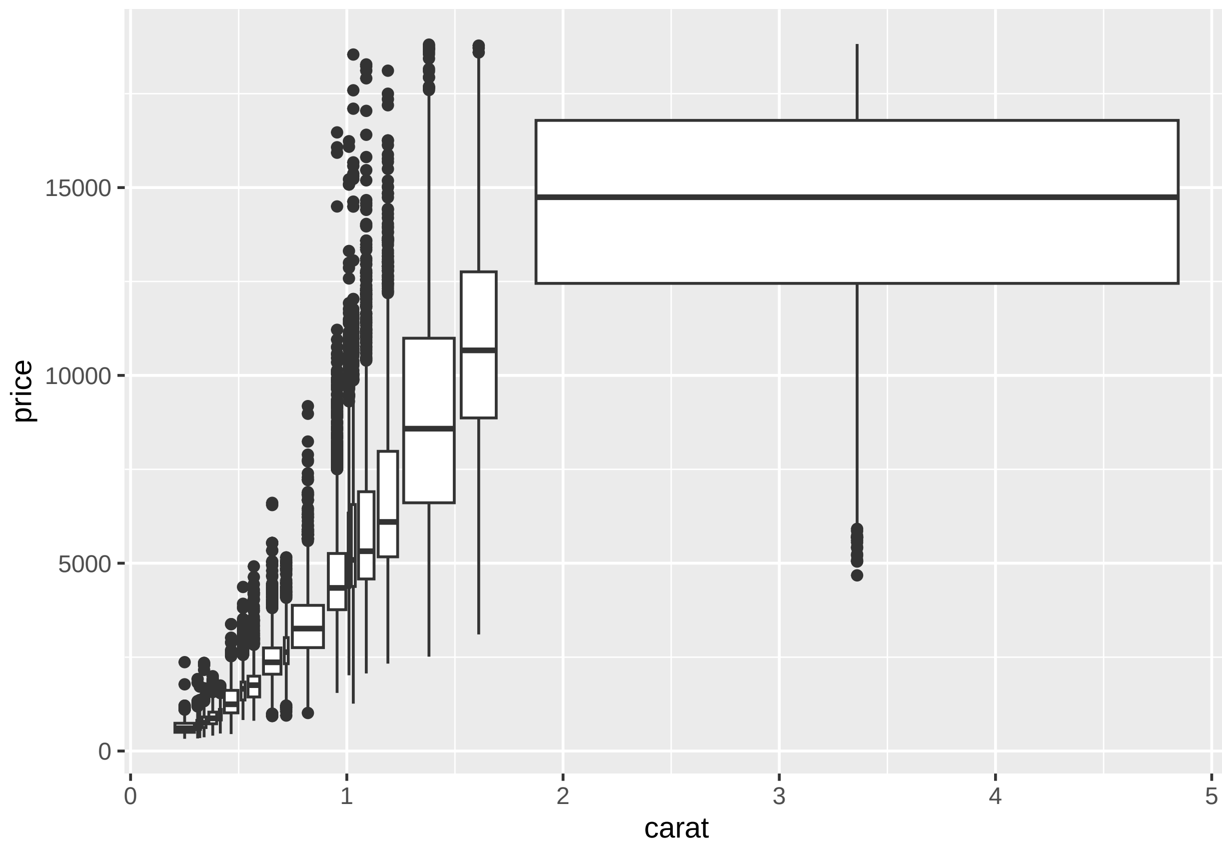
```
## Error in group_by(flights, month, dest): object 'flights' not found
```

5.2) Cut_width() divides the range of the data into equally spaced (bins) of a set width, regardless of how many observations fall into each bin. This is helpful to keep the data even based on widths, but it won't be helpful when we want to analyze patterns. On the other hand, cut_number() divides the data into a specified number of bins of (approximately) equal size, based on the number of observations which is helpful if we want to visualize the size in groups, but will not be as even as the cut_width() function.

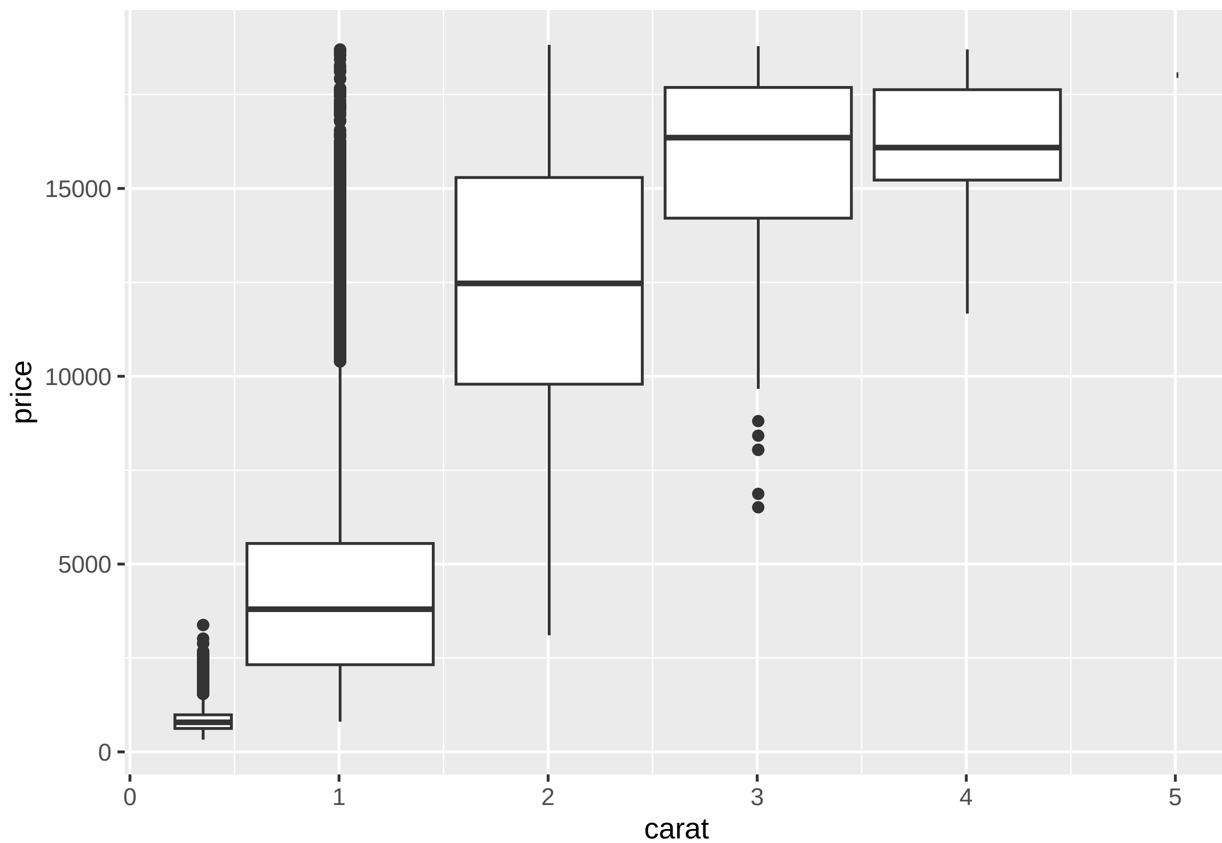
In summary, cut_width() creates bins of a fixed width, while cut_number() creates bins with roughly the same number of observations in each bin.

Citation: help from chat GPT.

```
diamonds <- diamonds

ggplot(diamonds, aes(x = carat, y = price, group = cut_number(carat, 20))) + geom_boxplot()
```



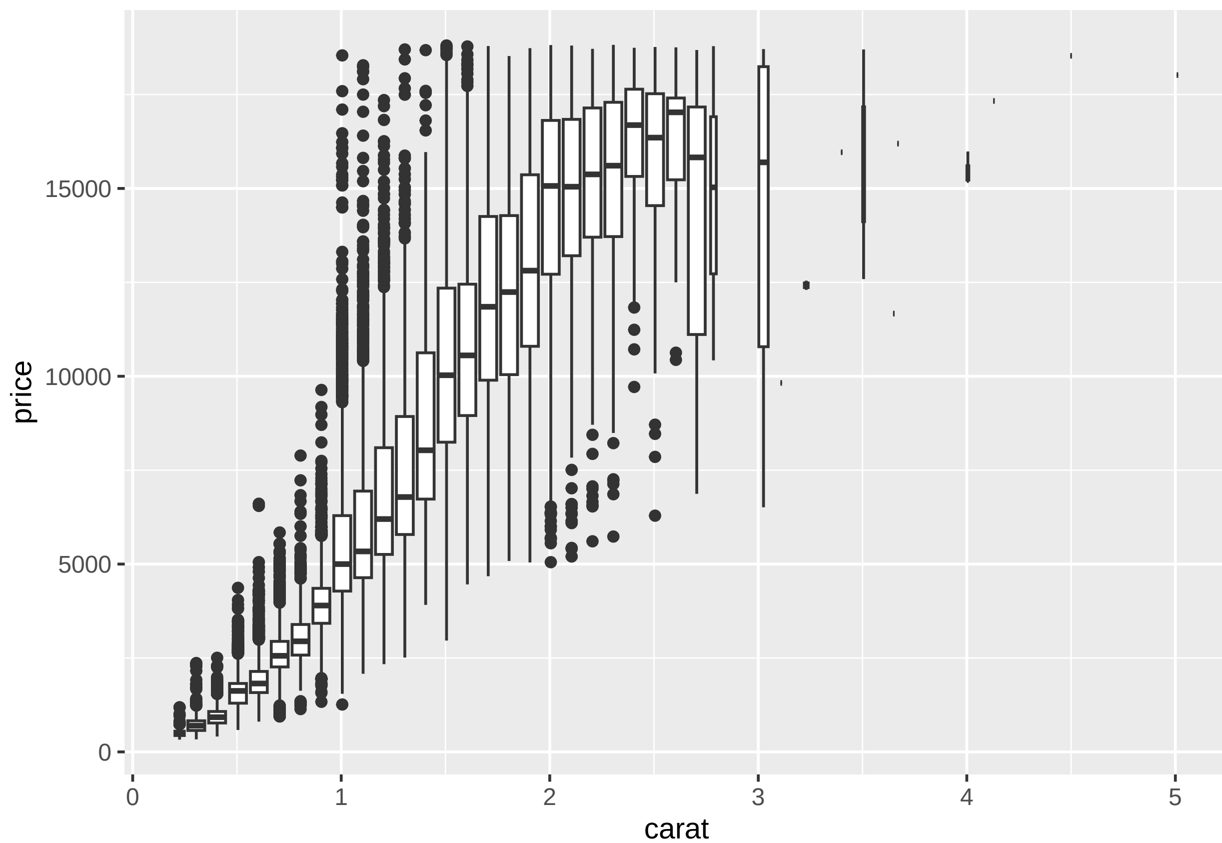

```
ggplot(diamonds, aes(x = carat, y = price, group = cut_width(carat, 1))) + geom_boxplot()
```



5.3)

a.) In this data set, the variable "carat" is most important for predicting the price of a diamond. When I plotted price against the other variables, I did not notice much of a correlation with price as much as I did when I plotted price against the variable "carat". As suggested in the book, I put carats in bins due to the large amount of data points in the dataframe.

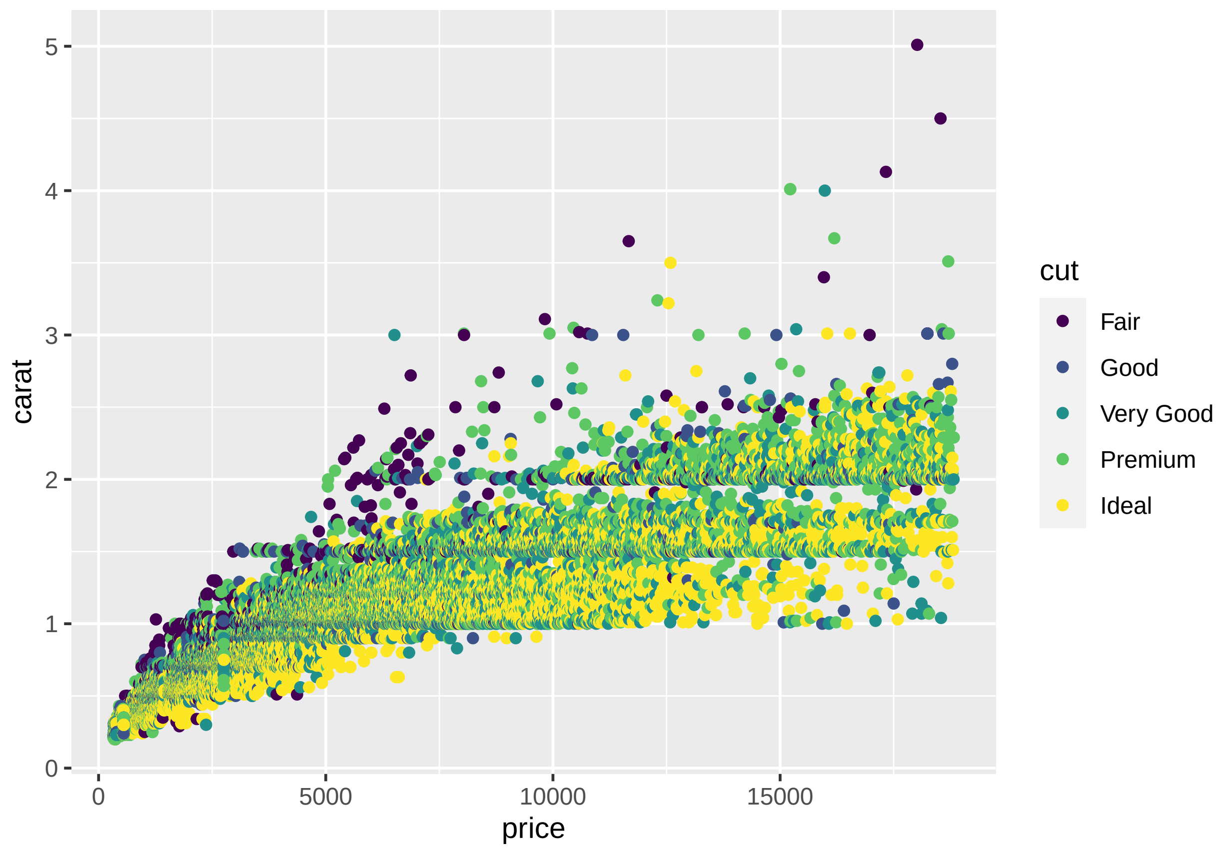
```
ggplot(data = diamonds, mapping = aes(x = carat, y = price)) +
  geom_boxplot(mapping = aes(group = cut_width(carat, 0.1)))
```



b.) By running the lm() function, we know that the correlation between cut and carat is slightly negative. Diamonds with higher carat weight have lower cut ratings, hence, the two variables are negatively correlated.

```
ggplot(diamonds, aes(x= price, y=carat)) +
  geom_point(aes(color = cut), binwidth = 1000)
```

```
## Warning in geom_point(aes(color = cut), binwidth = 1000): Ignoring unknown
## parameters: ‘binwidth‘
```



c.) Larger cut diamonds can be slowed for higher prices with a lower cut quality while smaller diamonds can be sold with a higher cut quality. The table in Question 5.3 is the complete opposite of that. It is misleading because it gives off the impression that price is dictated by cut, while its actually dictated by carat. As seen in the plot in 5.3 (b), different cut types exist across the board so cut type doesn't necessarily give accurate estimates of the price.

5.4.)

a.)In the output below, we can see which cut is most common in every color category. Color G –> ideal cut Color E –> ideal cut Color F –> ideal cut Color H –> ideal cut Color D –> ideal cut Color I –> ideal cut

```
diamonds |>
  count(color, cut)
```

```
## # A tibble: 35 x 3
##    color cut           n
##    <ord> <ord>     <int>
##  1 D     Fair        163
##  2 D     Good        662
##  3 D     Very Good  1513
##  4 D     Premium    1603
##  5 D     Ideal      2834
##  6 E     Fair        224
##  7 E     Good        933
##  8 E     Very Good  2400
##  9 E     Premium    2337
## 10 E     Ideal      3903
## # i 25 more rows
```

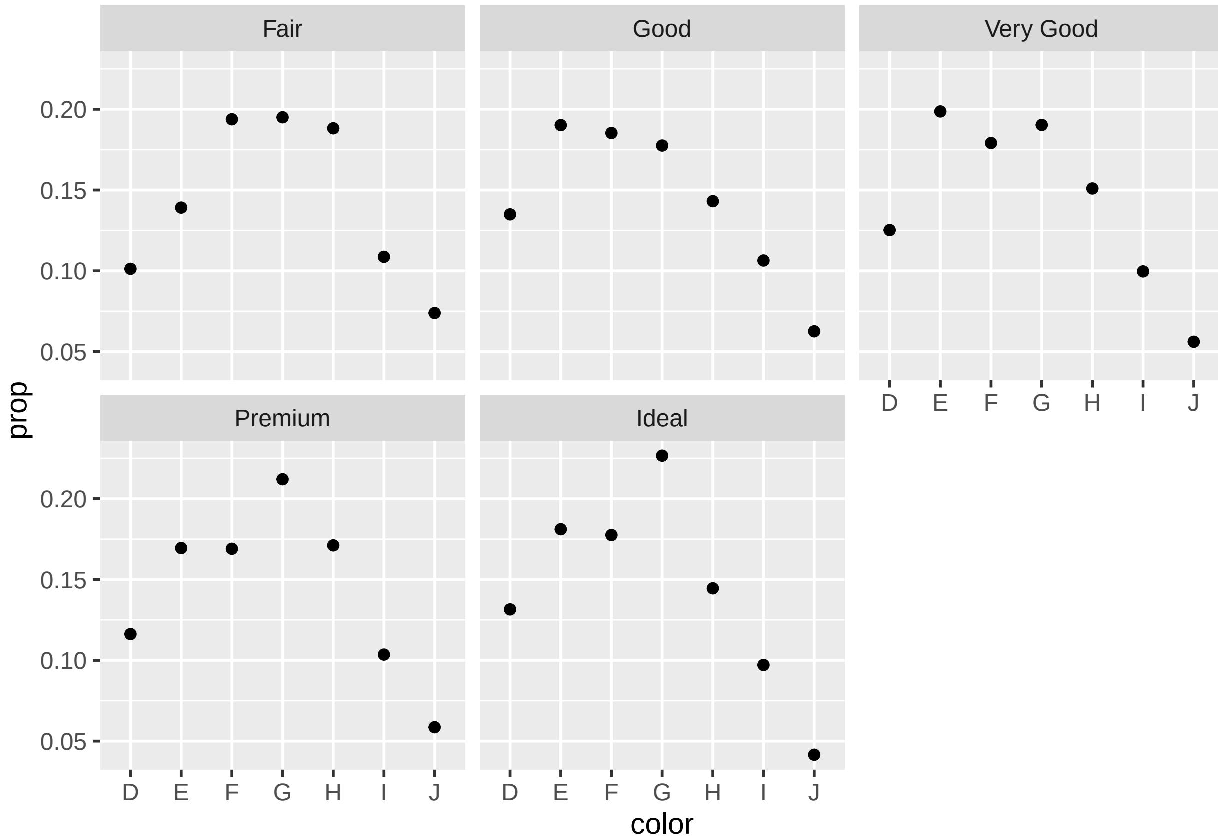
5.4.b.) In the table below, the "prop" column shows distribution of color within cut.

```
diamonds |>
  count(color, cut) |>
  group_by(cut) |>
  mutate(prop = n / sum(n))
```

```
## # A tibble: 35 x 4
## # Groups:   cut [5]
##    color cut           n  prop
##    <ord> <ord>     <int> <dbl>
##  1 D     Fair        163 0.101
##  2 D     Good        662 0.135
##  3 D     Very Good  1513 0.125
##  4 D     Premium    1603 0.116
##  5 D     Ideal      2834 0.132
##  6 E     Fair        224 0.139
##  7 E     Good        933 0.190
##  8 E     Very Good  2400 0.199
##  9 E     Premium    2337 0.169
## 10 E     Ideal      3903 0.181
## # i 25 more rows
```

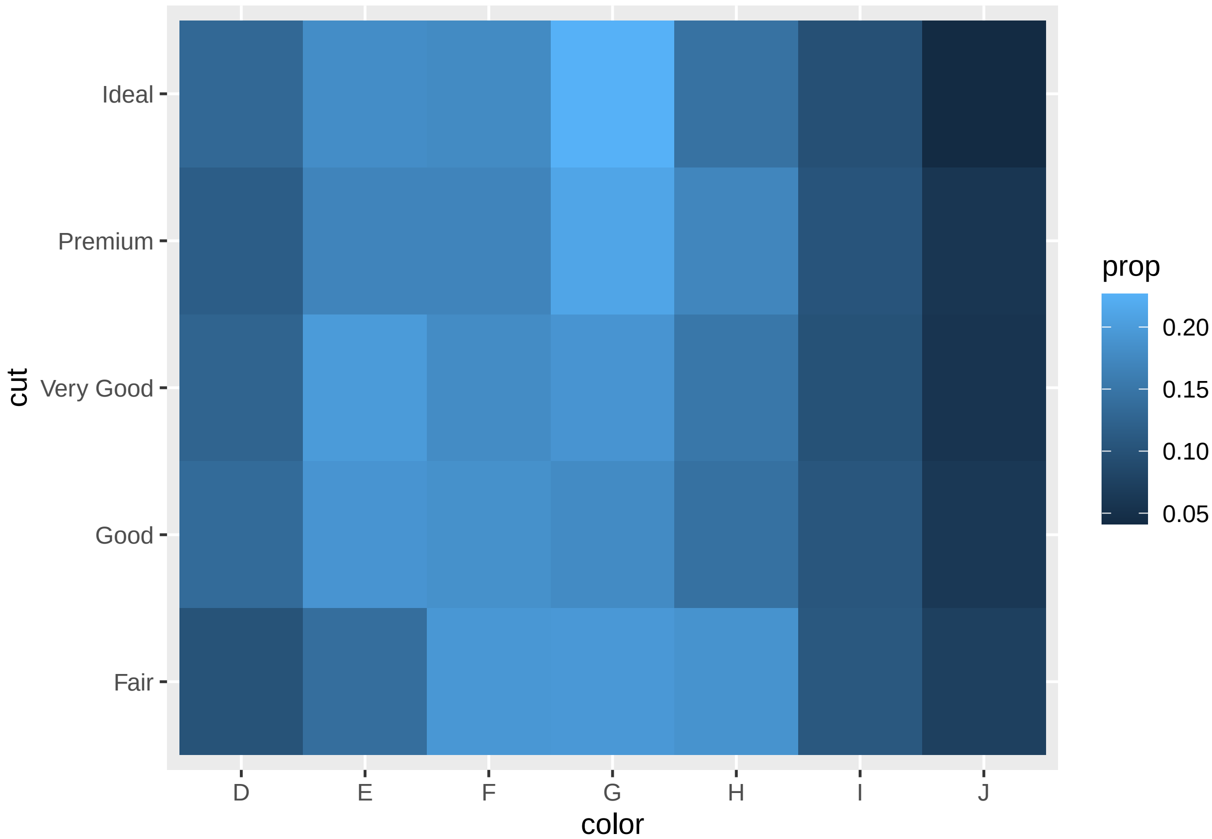
5.4.c.)

```
diamonds |>
  count(color, cut) |>
  group_by(cut) |>
  mutate(prop = n / sum(n)) |>
  ggplot(mapping = aes(x = color, y = prop)) +
  geom_point() + facet_wrap(~cut)
```



5.5)

```
diamonds |>
  count(color, cut) |>
  group_by(cut) |>
  mutate(prop = n / sum(n)) |>
  ggplot(mapping = aes(x = color, y = cut)) +
  geom_tile(mapping = aes(fill = prop))
```


Ideal
Premium
Very Good
Good
Fair
cut
D
E
F
G
H
I
J
color
prop
0.20
0.15
0.10
0.05