E392: Problem Set 2

Data Transformation & Exploratory Data Analysis

Spring 2018

Answer Key

Answers are typed below in italic font. I graded the questions ***Create new variables*** *and* ***Covariation****.*

Please work on the following questions and hand in your solutions in groups of at most 2 students. You are asked to answer all questions, but we will only select 2 questions randomly to grade.

Part 1: R questions

Question 1: Data Transformation

The following questions use the `nycflights13` data that we used in class.

Filtering

1. How many flights have a missing `dep_time`? What other variables are missing? What might these rows represent? *There are 8255 flights with no departure time. Probably these indicate canceled flights.*

   ```
   dt_missing = filter(flights, is.na(dep_time))
   nrow(dt_missing)
   ```

   ```
   ## [1] 8255
   ```

2. Why is `NA ^ 0` not missing? Why is `NA | TRUE` not missing? Why is `FALSE & NA` not missing? Can you figure out the general rule? (`NA * 0` is a tricky counterexample!) *This is an important and interesting point. In R, `NA` is best interpreted as* ***`I don't know what this is`****. `NA ^ 0` will not be `NA` because it doesn't matter what `NA` is here: `NA ^ 0` will always be 1. Similarly, for the logical comparisons `NA | TRUE`. Here `NA` stands for either `TRUE` or `FALSE`, we don't know. But here it doesn't matter since not matter what `NA` is, `NA | TRUE` will always be `TRUE`. Analogously, `FALSE & NA` will always be `FALSE`, no matter what `NA` is. Therefore none of the two expressions is going to be `NA`. `NA * 0` is a litle tricky. You would expect that no matter which number you multiply by zero, the outcome will always be zero. However, `NA` could also be infinity, in which case `NA * 0` would not be zero, so the expression is `NA`.*

Arrange data

1. How could you use `arrange()` to sort all missing values to the start? (Hint: use `is.na()`). *The easiest is probably:*

```
flights_na_first <- mutate(flights,dt_missing=is.na(dep_time)) %>%
    arrange(desc(dt_missing))
```

1. Which flights traveled the longest? Which traveled the shortest? *Simply sort data according to distance and extract the first flight(s) and last flight(s).*

```
flights_distance <- arrange(flights, distance) %>%
                    filter(row_number()<=10 | row_number()>n()-10)
```

Select columns

1. What happens if you include the name of a variable multiple times in a `select()` call? *Nothing, the variable will still just be included once in your new data set.*

```
flights_test <- select(flights, year,month,day,day)
```

Create new variables

1. Compare `air_time` with `arr_time - dep_time`. What do you expect to see? What do you see? What do you need to do to fix it? *Our computed variable does not match the* ***`air_time`*** *variable although it should be the same. The problem is that all variables are coded as integers (numbers), not date-times. Therefore, neither of the variables is interpreted in time units. Declaring both variables in date-time format would fix this issue. If you are interested, we can talk about date-times during the March 7 lecture. In addition, the destination might be in a different time zone and all times are store in local time, which we could also handle by working with specific date-time formats.*

```
flights_at_check <- mutate(flights, at_check = arr_time - dep_time) %>%
    select(year,month,day,air_time,arr_time,dep_time, at_check)
```

1. Compare `dep_time`, `sched_dep_time`, and `dep_delay`. How would you expect those three numbers to be related? *Similarly to the previous question,* ***`dep_delay`*** *should be the difference between* ***`dep_time`*** *and* ***`sched_dep_time`****. To check this rigorously, we would first have to convert all variables to appropriate date-time formats.*

```
flights_delay_check <- mutate(flights, delay_check = dep_time - sched_dep_time) %>%
    select(dep_time,sched_dep_time, dep_delay, delay_check)
```

1. What does `1:3 + 1:10` return? Why? *It will return a 10-component vector where the first (shorter) vector will be recycled such that it fits the dimensions of the longer vector. This is another aspect that often causes confusion in base R although it helps*

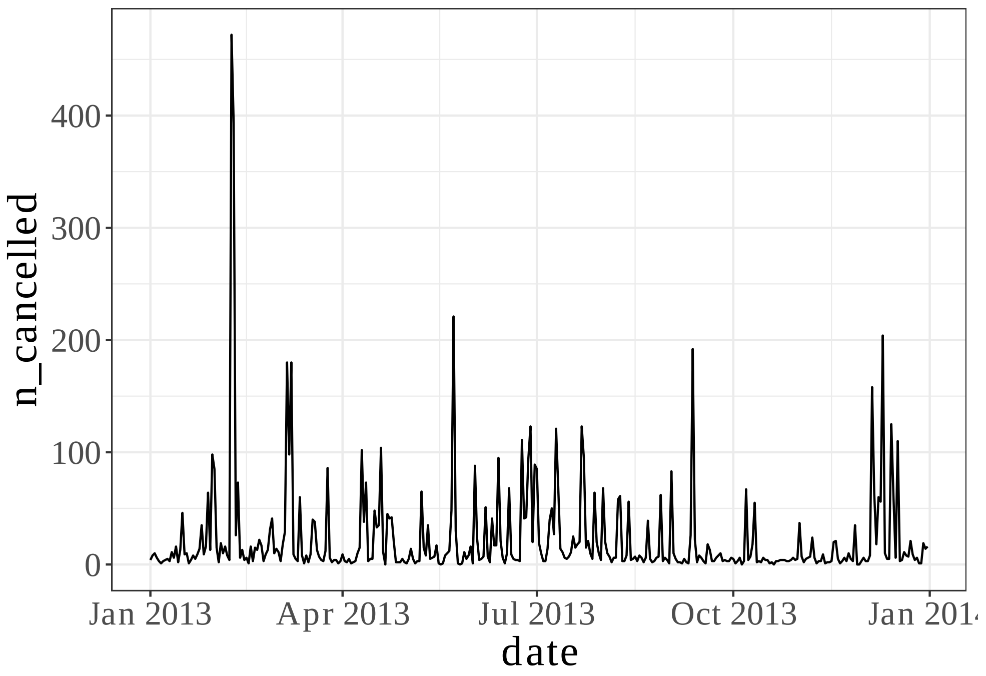
with certain aspects of coding. More modern languages and packages only allow scalars, but not vectors to be recycled which is often all we need.

Grouped summaries

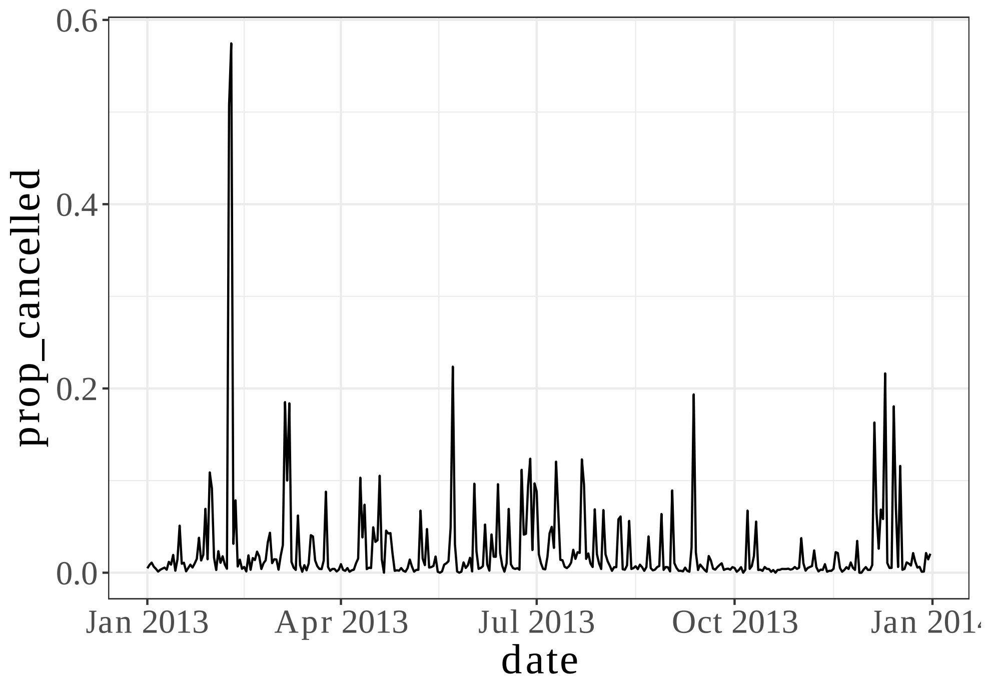
1. Look at the number of cancelled flights per day. Is there a pattern? Is the proportion of cancelled flights related to the average delay? *There are lots of aspects you could look at here. I simply plot the number of cancelled flights over time and we see that there are spikes in cancellations around Christmas, one in February/March and several modest spikes during summer, while the fall months do not have a lot of cancelled flights.*

```r
by_day <- group_by(flights,year,month,day)
flights_cancelled <- summarise(by_day, n_cancelled = sum(is.na(dep_time)),
                               n_total = n(), prop_cancelled = n_cancelled / n_total,
                               dep_delay_mean = mean(dep_delay, na.rm = TRUE)) %>%
    ungroup()
# Create a new date variable to facilitate plotting.
flights_cancelled <- mutate(flights_cancelled, date = make_date(year,month,day))

# Create plot for cancelled flights.
ggplot(data=flights_cancelled, mapping = aes(x = date, y = n_cancelled)) +
    geom_line()
```



```r
# Create plot for proportion of cnacelled flights.
ggplot(data=flights_cancelled, mapping = aes(x = date, y = prop_cancelled)) +
    geom_line()
```



```
# Check for correlation between cancelled flights and average departure delay.
cor(flights_cancelled$prop_cancelled,flights_cancelled$dep_delay_mean)
```

```
## [1] 0.5516646
```

1. Which carrier has the worst delays? Challenge: can you disentangle the effects of bad airports vs. bad carriers? Why/why not? (Hint: think about `flights %>% group_by(carrier, dest) %>% summarise(n())`) *Let's look at the average delay per carrier and then at the average delay by carrier and airport. Frontier Airlines has the worst delays on average. The last part of probably very hard to do without a formal model. One approach could be to compare the average delays of each airline relative to the average delay of flights to that destination. However, you probably would like to compare flights of one airline onlyto the average delays generated by all other airlines. FiveThirtyEight did a similar analysis. You can find it here.*

```
flights_carr_delay <- group_by(flights, carrier) %>%
    summarise(avg_delay = mean(arr_delay, na.rm = TRUE)) %>%
    arrange(avg_delay)
flights_carrdest_delay <- group_by(flights, carrier,dest) %>%
    summarise(avg_delay = mean(arr_delay, na.rm = TRUE)) %>%
    arrange(avg_delay)
flight_cd_n <- flights %>%
    group_by(carrier, dest) %>%
    summarise(n())
```

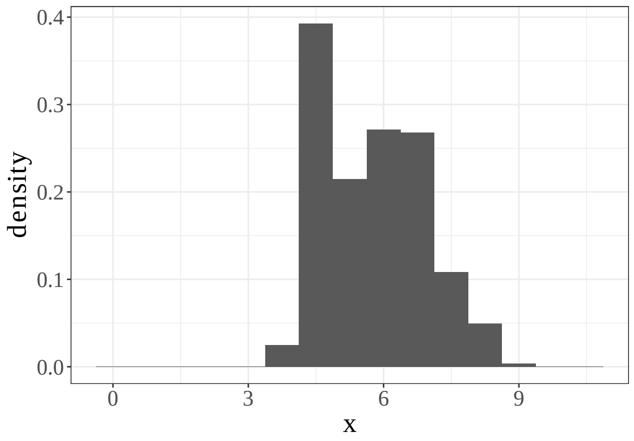
Question 2: Exploratory Data Analysis

The following questions use the `diamonds` data set that comes with the `tidyverse` library.

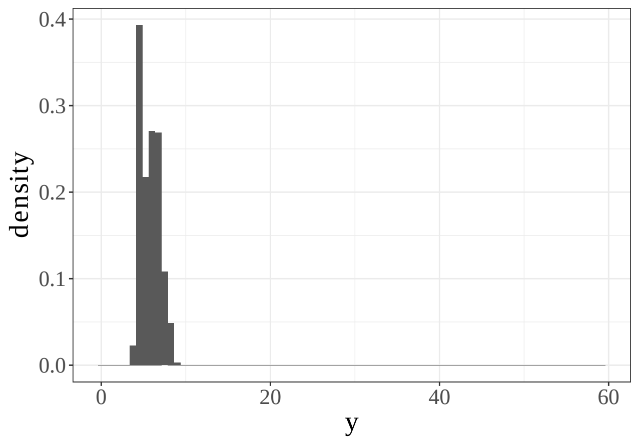
Typical and atypical values

1. Explore the distribution of each of the `x`, `y`, and `z` variables in `diamonds`. What do you learn? Think about a diamond and how you might decide which dimension is the length, width, and depth. *Again there are plenty of things you could investigate about this data. One striking feature is that there seem to be a few significant outliers, mostly in the y and z-dimension. These deserve further investigation. If we zoom into the distribution using a smaller bandwidth, we see that there are spikes in the distribution in more or less fixed intervals. These are most likely due to measuring/rounding errors as discussed in class.*

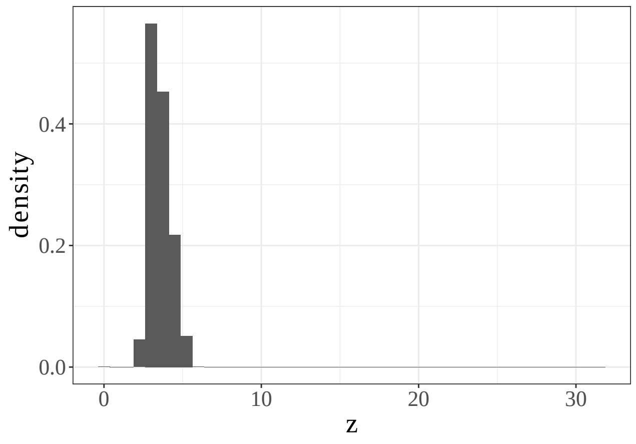
```r
rm(list=ls())
ddata <- diamonds
# Explore distribution with a large bandwith.
# Explore x-dimension.
ggplot(diamonds) +
    geom_histogram(aes(x = x, y = ..density..), binwidth=0.75)
```



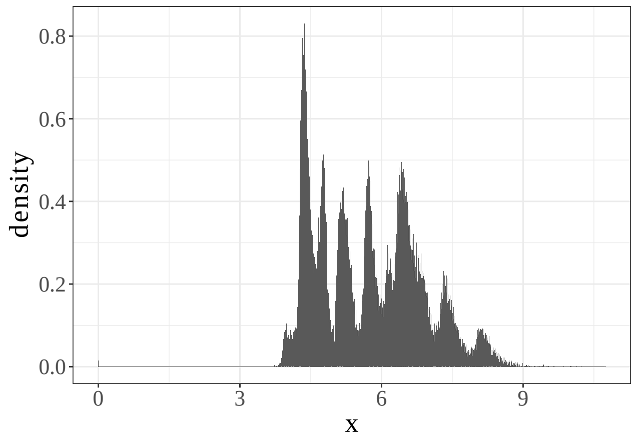
```r
# Explore y-dimension.
ggplot(diamonds) +
    geom_histogram(aes(x = y, y = ..density..), binwidth=0.75)
```



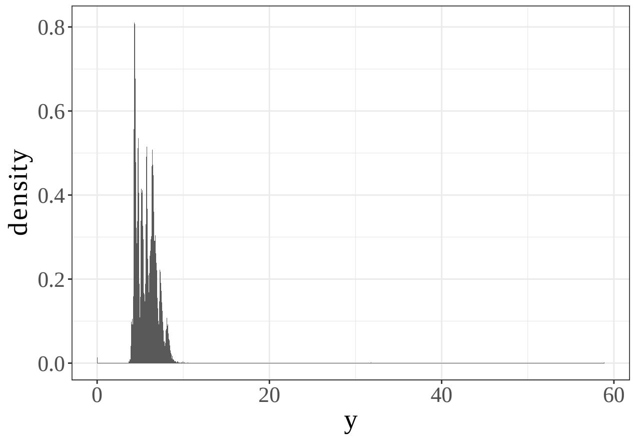
```r
# Explore z-dimension.
ggplot(diamonds) +
    geom_histogram(aes(x = z, y = ..density..), binwidth=0.75)
```



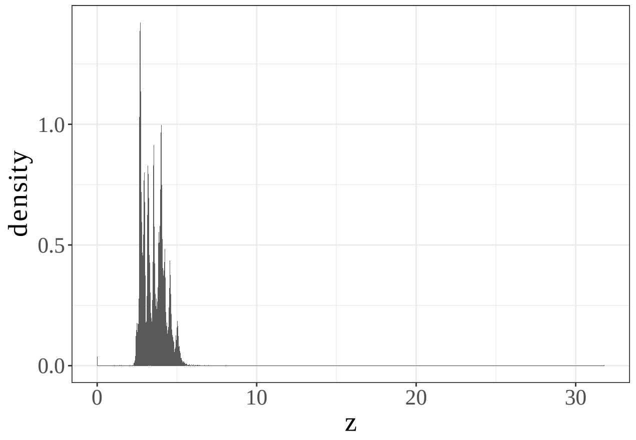
```r
# Explore distribution with a smaller bandwith.
# Explore x-dimension.
ggplot(diamonds) +
    geom_histogram(aes(x = x, y = ..density..), binwidth=0.01)
```



```
# Explore y-dimension.
ggplot(diamonds) +
    geom_histogram(aes(x = y, y = ..density..), binwidth=0.01)
```

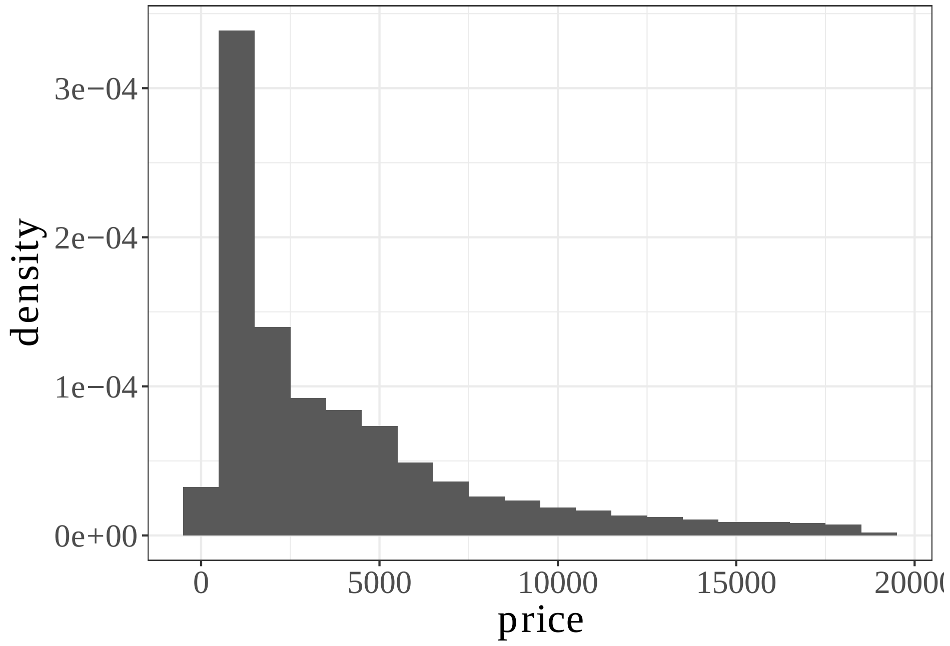


```
# Explore z-dimension.
ggplot(diamonds) +
    geom_histogram(aes(x = z, y = ..density..), binwidth=0.01)
```

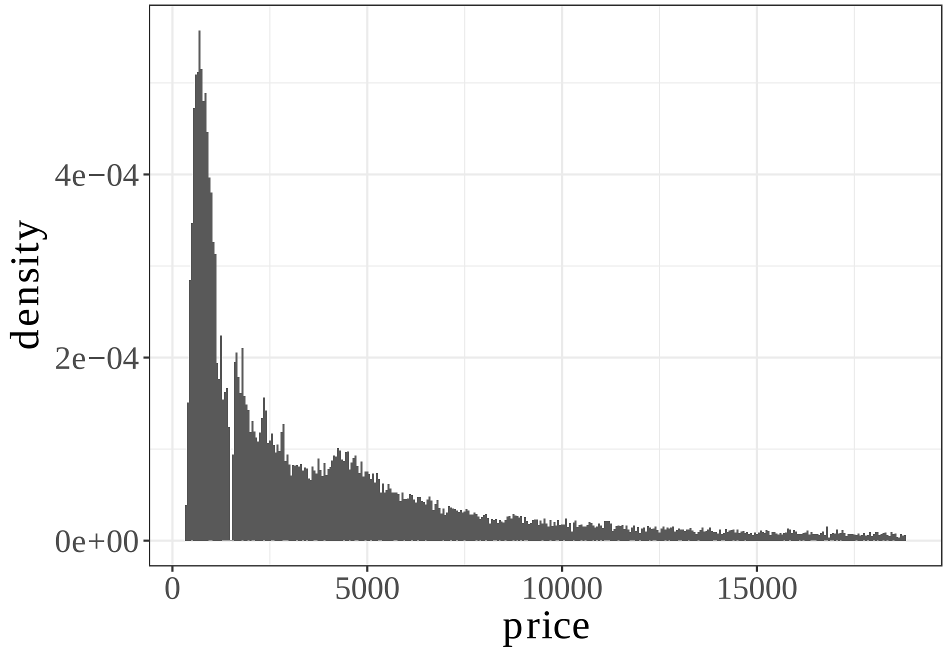


1. Explore the distribution of `price`. Do you discover anything unusual or surprising? (Hint: Carefully think about the `binwidth` and make sure you try a wide range of values.) *When we start with a large bandwidth to see the distribution on a very aggregate level. We see that the price distribution has a large right tail. Most diamonds cost less than 2500 USD, but there are a few diamonds that cost up to almost 20000 USD. When we zoom in, by using a smaller bin width, we see spikes in the distribution at more or less regular intervals. One way to proceed is to focus on the range of most common prices and zoom in in great detail. A striking observation is that while the price distributions looks generally very continuous, there is a big gap around 1500 USD. It seems like there are no diamonds that are sold for prices between 1454 USD and 1546 USD.*

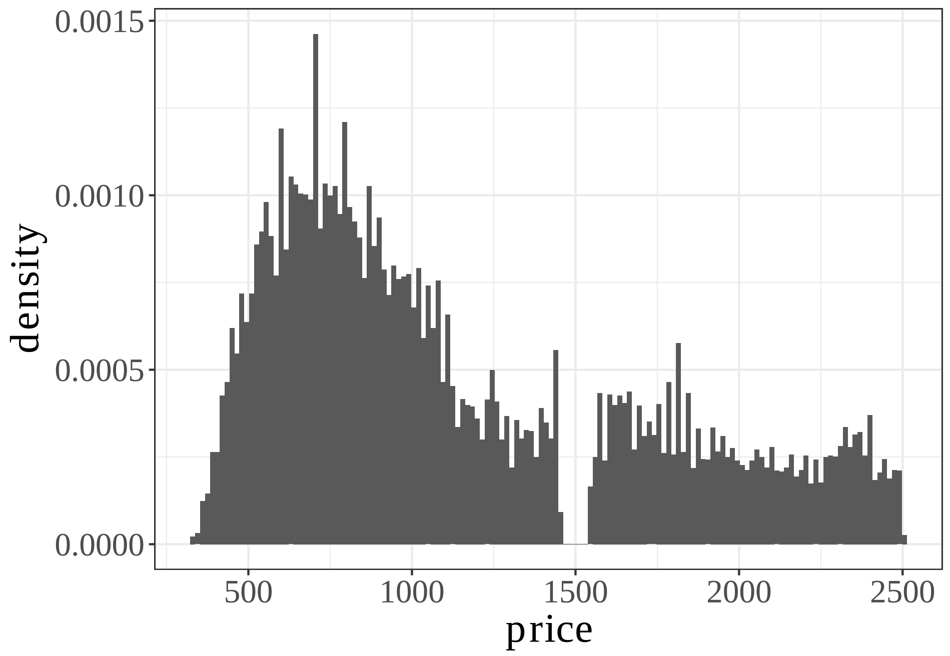
```
ggplot(data=diamonds) +
    geom_histogram(aes(x=price, y= ..density..), binwidth=1000)
```



```
ggplot(data=diamonds) +
    geom_histogram(aes(x=price, y= ..density..), binwidth=50)
```



```
diamonds_cheap <- filter(diamonds, price < 2500)
ggplot(data=diamonds_cheap) +
    geom_histogram(aes(x=price, y= ..density..), binwidth=15)
```



1. How many diamonds are 0.99 carat? How many are 1 carat? What do you think is the cause of the difference? *There are more than 1500 diamonds with exactly one carat, but only 23 with 0.99 carat. This is most likely due to when measuring diamonds, people tend to round the weight to a full number. Interestingly, there are more than 2000 diamonds with exactly 1.01 carat. So people don't seem to round down, but only up.*

```
diamonds_1carat <- filter(diamonds,carat==1)
# Number of 1-carat diamonds.
nrow(diamonds_1carat)
```

```
## [1] 1558
```

```
diamonds_099carat <- filter(diamonds,carat==0.99)
# Number of 0.99 carat diamonds.
nrow(diamonds_099carat)
```

```
## [1] 23
```

```
diamonds_101carat <- filter(diamonds,carat==1.01)
# Number of 1.01 carat diamonds.
nrow(diamonds_101carat)
```

```
## [1] 2242
```

Missing values

1. What happens to missing values in a histogram? What happens to missing values in a bar chart? Why is there a difference? *In a histogram, `NA`s are removed when the number of observations in each bin are computed. In a bar chart, missing values are*

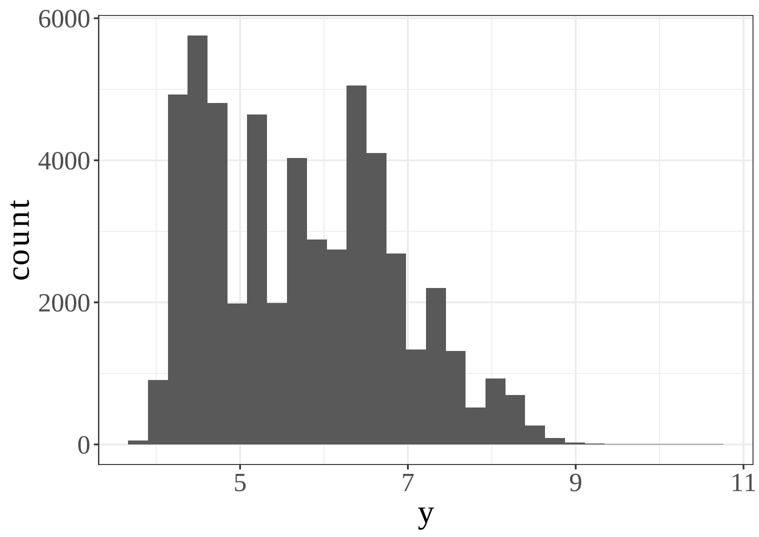
just another category. In histograms, the grouping variables needs to be numeric, since the value of NA is unknown, these observations cannot be placed in any bin. In contrast, geom_bar() splits the data in categories that are not necessarily numerical.

2. What does `na.rm = TRUE` do in `mean()` and `sum()`? *These arguments tell R whether to ignore missing values when computing the summary statistics, such as mean or the sum. This is almost always what you want since missing values are contagious, i.e. when computing for example a mean without removing missing values, we would always get NA as the result.*

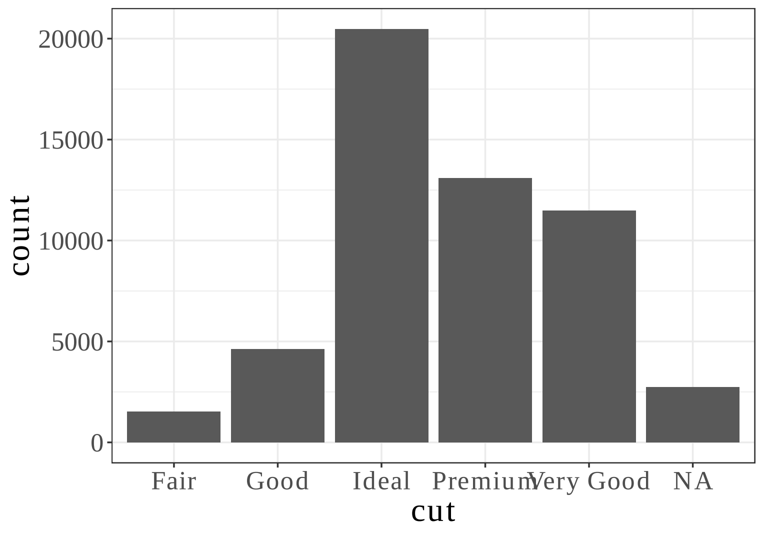
```
diamonds_clean <- diamonds %>%
  mutate(y = ifelse(y < 3 | y > 20, NA, y))
# A histogram will simply signore missing values.
ggplot(diamonds_clean, aes(x = y)) +
  geom_histogram()
```

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

```
## Warning: Removed 9 rows containing non-finite values (stat_bin).
```



```
diamonds %>%
# Here I just create some random missing values to illustrate how a bar chart handles
  mutate(cut = if_else(runif(n()) < 0.05, NA_character_, as.character(cut))) %>%
  ggplot() +
  geom_bar(mapping = aes(x = cut))
```

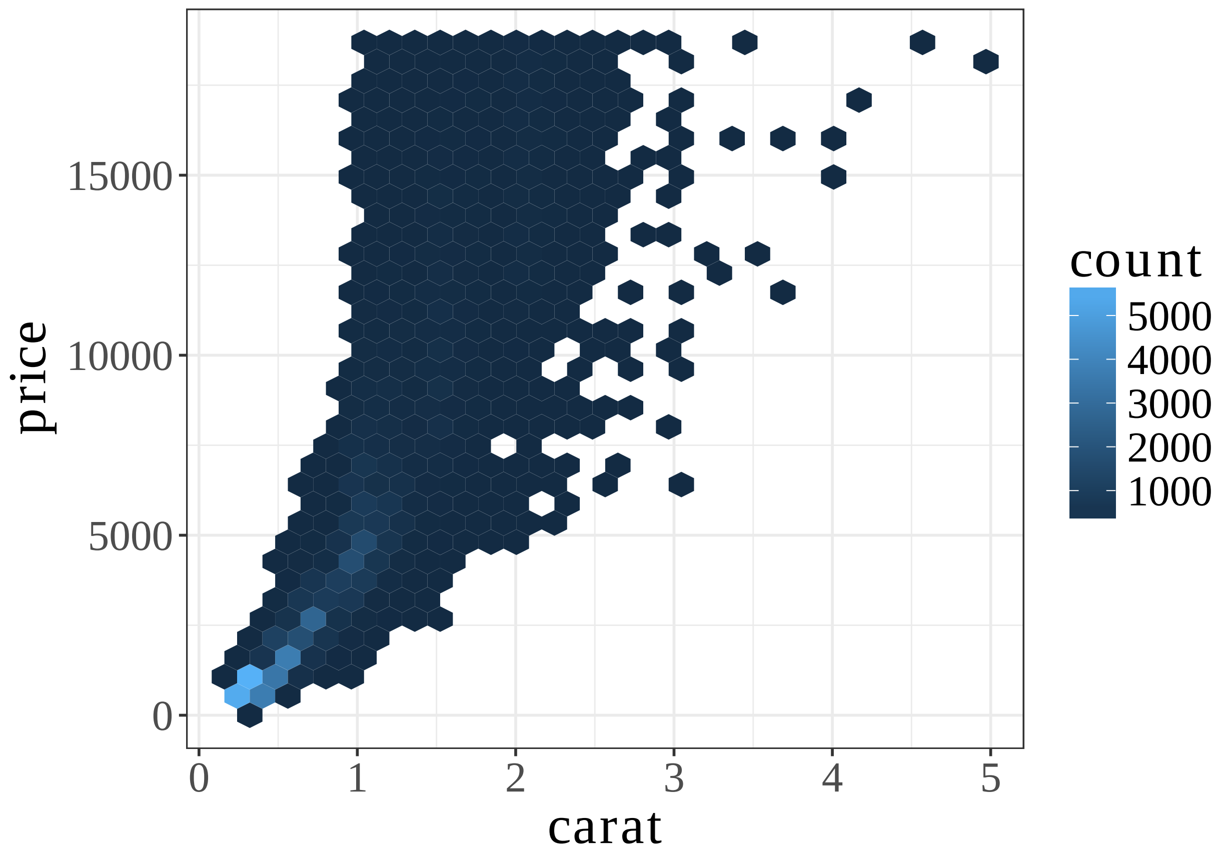


Covariation

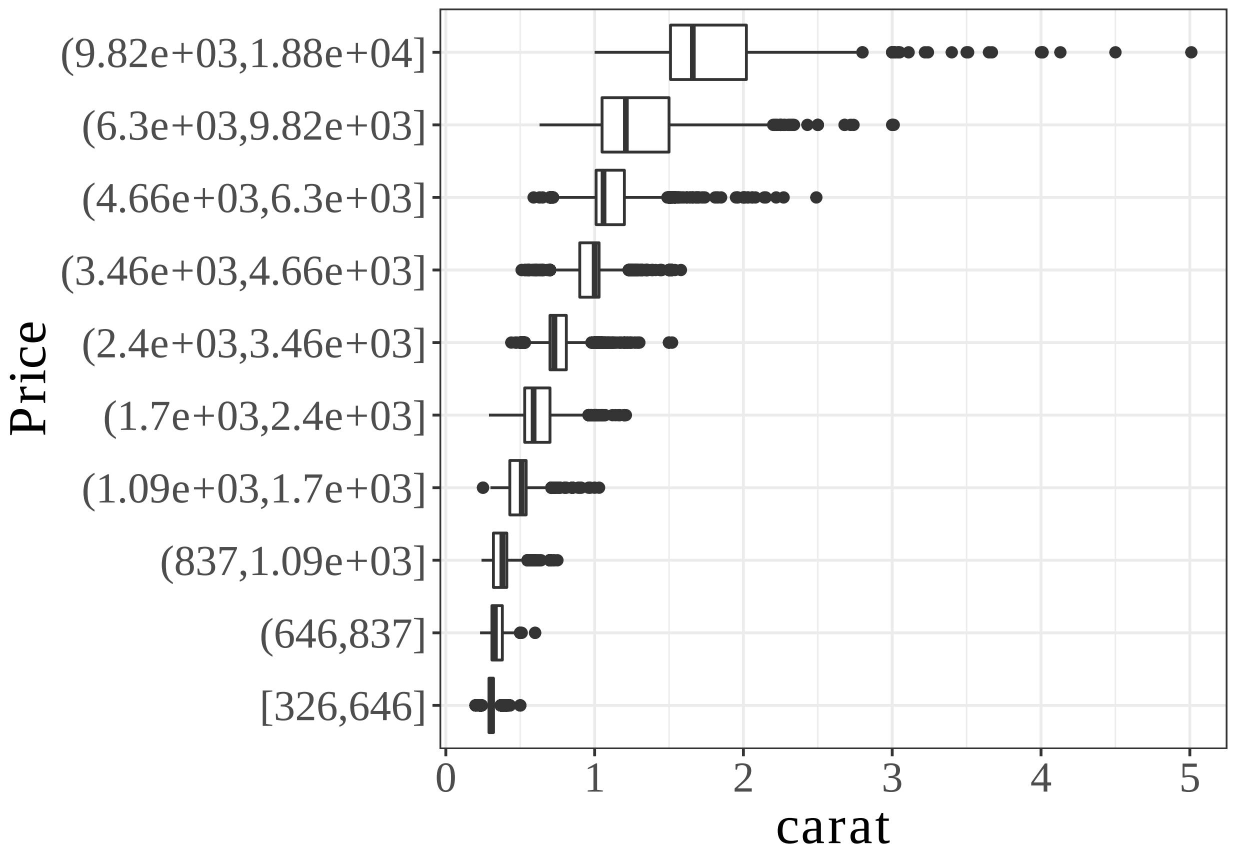
1. Visualize the distribution of carat, partitioned by price. *The easiest is to use two-dimensional bins as discussed in class. However, this graph does not convey a lot of information about the carat distribution. Boxplots, partitioned by price ranges, could be more useful. Since price is a continuous variable, we can use the* ***`cut_number()`*** *function to split the prices into, for example 10 different equally-spaced categories. We see that for low prices, there is only little variation in the weight of the diamond. Not surprisingly, more expensive diamonds tend to be larger, but we also see that there is substantially more variation and more outliers in the group of very expensive diamonds.*

```
# A heatmap-style graph.
ggplot(data = diamonds) +
geom_hex(mapping = aes(x = carat, y = price))
```

```
## Warning: package 'hexbin' was built under R version 3.4.3
```

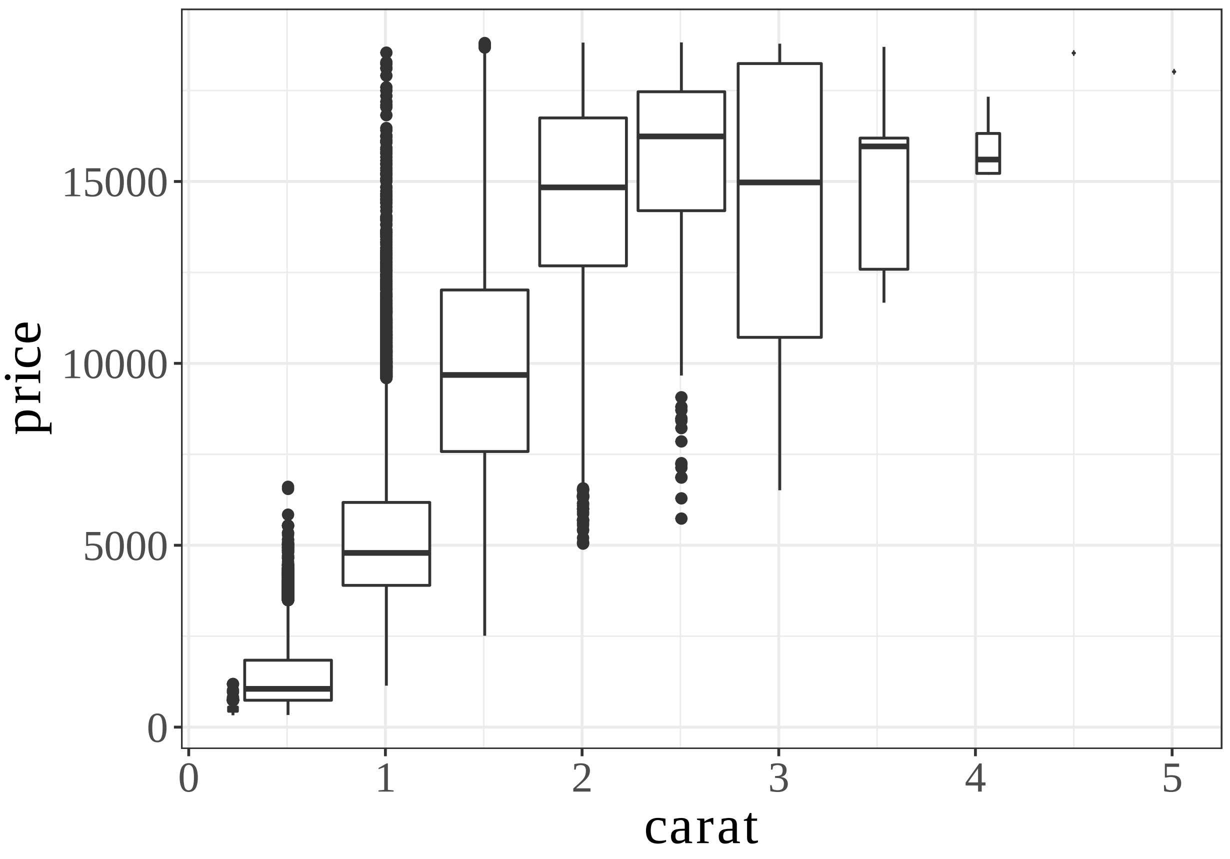


```r
# Boxplots partitioned by price.
ggplot(diamonds, aes(x = cut_number(price, 10), y = carat)) +
  geom_boxplot() +
    coord_flip() +
    xlab("Price")
```



1. How does the price distribution of very large diamonds compare to small diamonds. Is it as you expect, or does it surprise you? *Above a certian threshold (at around 2 carat), prices do not seem to increase a lot with size, most likely because other factors become more important such as cut quality and color. Maybe large diamonds simply tend to be of lower quality. In addition, the variance of prices increases substantially with carat size which is not surprising since large diamonds are probably much more hetergeneous than smaller diamonds.*

```r
ggplot(data = filter(diamonds,x>0), mapping = aes(x = carat, y = price)) +
  geom_boxplot(aes(group = cut_width(carat, 0.5)))
```



Part 2: Your project

Question 3

Continue working on your project! Reconsider the 2 or 3 most promising ideas or data sets that you came up with last week. Focus on one of them and think more about the data aspect.

If the data is not freely available, check carefully which steps you have to take in order to acquire the data. In at most one page, outline the next steps you plan on taking to acquire the data together with a timeline (allow for some buffer time in each step). Make a realistic assessment on whether you will be able to get the data in time for your course project.

If you already have access to the data, start looking into the documentation, in particular at those variables that are most important for your analysis. Try to figure out whether the data looks good enough for your project. Describe which features of the data you looked at to judge its quality and why you think it is good or bad data.