Tidyverse and Data Visualization Exercises

Juan R Gonzalez

1 March 2019

Exercises (tibbles)

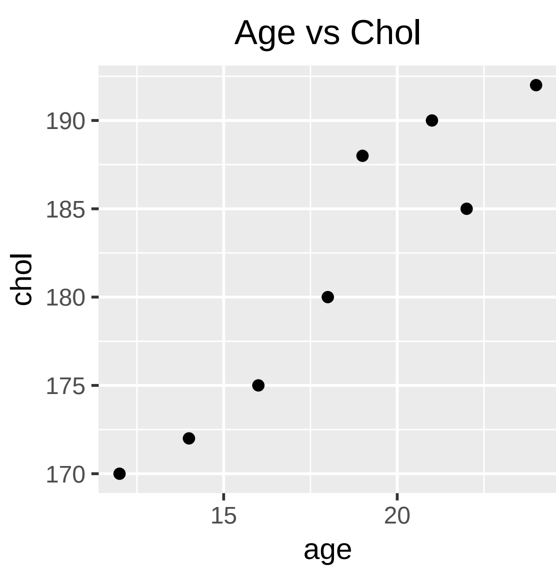
1. You can tell whether an obect is a tibble simply by printing it. When printing regular data frames the full content of the data frame is printed to screen and the data types of columns are not shown. In contrast printing a tibble only prints the first 10 rows and also shows the data type of each column.
2. You can extract the reference variable either by object$var or object[[“var”]]
3. The “n_extra” option in the print function controls how many additional columns are printed at the footer of a tibble.
4.

```
tbl <- tibble(
  age = c(14, 18, 22, 12, 16, 19, 21, 24),
  chol = c(172, 180, 185, 170, 175, 188, 190, 192),
  sex = c("male", "male", "female", "female", "female", "male", "male", "male")
)

tbl[["sex"]]
```

```
[1] "male"   "male"   "female" "female" "female" "male"   "male"   "male"
```

```
ggplot(data = tbl, mapping = aes(x = age, y = chol)) + geom_point() +
  labs(title = "Age vs Chol") + theme(plot.title = element_text(hjust=0.5))
```




```
mutate(tbl, chol2 = chol^2)
```

```
# A tibble: 8 x 4
    age  chol sex    chol2
  <dbl> <dbl> <chr>  <dbl>
1    14   172 male   29584
```

```
2     18    180 male   32400
3     22    185 female 34225
4     12    170 female 28900
5     16    175 female 30625
6     19    188 male   35344
7     21    190 male   36100
8     24    192 male   36864
```

```r
rename(tbl,  one = age,  two = chol, three = sex)
```

```
# A tibble: 8 x 3
    one   two three
  <dbl> <dbl> <chr>
1     14    172 male
2     18    180 male
3     22    185 female
4     12    170 female
5     16    175 female
6     19    188 male
7     21    190 male
8     24    192 male
```

Exercises (data transform)

1.

```r
filter(flights, arr_delay >= 120)
```

```
# A tibble: 10,200 x 19
    year month   day dep_time sched_dep_time dep_delay arr_time
   <int> <int> <int>    <int>          <int>     <dbl>    <int>
 1  2013     1     1      811            630       101     1047
 2  2013     1     1      848           1835       853     1001
 3  2013     1     1      957            733       144     1056
 4  2013     1     1     1114            900       134     1447
 5  2013     1     1     1505           1310       115     1638
 6  2013     1     1     1525           1340       105     1831
 7  2013     1     1     1549           1445        64     1912
 8  2013     1     1     1558           1359       119     1718
 9  2013     1     1     1732           1630        62     2028
10  2013     1     1     1803           1620       103     2008
# ... with 10,190 more rows, and 12 more variables: sched_arr_time <int>,
#   arr_delay <dbl>, carrier <chr>, flight <int>, tailnum <chr>,
#   origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
#   minute <dbl>, time_hour <dttm>
```

```r
filter(flights, (dest == "IAH" |  dest == "HOU") & arr_delay > 120 & dep_delay <= 0)
```

```
# A tibble: 0 x 19
# ... with 19 variables: year <int>, month <int>, day <int>,
#   dep_time <int>, sched_dep_time <int>, dep_delay <dbl>, arr_time <int>,
#   sched_arr_time <int>, arr_delay <dbl>, carrier <chr>, flight <int>,
#   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>,
#   distance <dbl>, hour <dbl>, minute <dbl>, time_hour <dttm>
```

```r
filter(flights, dep_delay >= 60 & arr_delay < 30)
```

```
# A tibble: 206 x 19
    year month   day dep_time sched_dep_time dep_delay arr_time
   <int> <int> <int>    <int>          <int>     <dbl>    <int>
 1  2013     1     3     1850           1745        65     2148
 2  2013     1     3     1950           1845        65     2228
 3  2013     1     3     2015           1915        60     2135
 4  2013     1     6     1019            900        79     1558
 5  2013     1     7     1543           1430        73     1758
 6  2013     1    11     1020            920        60     1311
 7  2013     1    12     1706           1600        66     1949
 8  2013     1    12     1953           1845        68     2154
 9  2013     1    19     1456           1355        61     1636
10  2013     1    21     1531           1430        61     1843
# ... with 196 more rows, and 12 more variables: sched_arr_time <int>,
#   arr_delay <dbl>, carrier <chr>, flight <int>, tailnum <chr>,
#   origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
#   minute <dbl>, time_hour <dttm>
```

2. The dplyr “between” function determines if a numeric vector falls within a specified range. It would not help in simplifying the code in the previous exercises due to the fact that we are never checking if a variable falls between two values. Rather, for all the exercises we check if a variable is greater/less than or equal to just one value.

3.

```r
arrange(flights, air_time)
```

```
# A tibble: 336,776 x 19
    year month   day dep_time sched_dep_time dep_delay arr_time
   <int> <int> <int>    <int>          <int>     <dbl>    <int>
 1  2013     1    16     1355           1315        40     1442
 2  2013     4    13      537            527        10      622
 3  2013    12     6      922            851        31     1021
 4  2013     2     3     2153           2129        24     2247
 5  2013     2     5     1303           1315       -12     1342
 6  2013     2    12     2123           2130        -7     2211
 7  2013     3     2     1450           1500       -10     1547
 8  2013     3     8     2026           1935        51     2131
 9  2013     3    18     1456           1329        87     1533
10  2013     3    19     2226           2145        41     2305
# ... with 336,766 more rows, and 12 more variables: sched_arr_time <int>,
#   arr_delay <dbl>, carrier <chr>, flight <int>, tailnum <chr>,
#   origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
#   minute <dbl>, time_hour <dttm>
```

4.

```r
(dep_info <- select(flights, contains("dep")))
```

```
# A tibble: 336,776 x 3
   dep_time sched_dep_time dep_delay
      <int>          <int>     <dbl>
 1      517            515         2
 2      533            529         4
 3      542            540         2
 4      544            545        -1
 5      554            600        -6
```

```
 6      554            558        -4
 7      555            600        -5
 8      557            600        -3
 9      557            600        -3
10      558            600        -2
# ... with 336,766 more rows
```

5.

```
(dep_hr_min <- transmute(flights, dep_hour = dep_time%/%100, dep_min = dep_time%%100))
```

```
# A tibble: 336,776 x 2
   dep_hour dep_min
      <dbl>   <dbl>
 1        5      17
 2        5      33
 3        5      42
 4        5      44
 5        5      54
 6        5      54
 7        5      55
 8        5      57
 9        5      57
10        5      58
# ... with 336,766 more rows
```

6.

```
delay_data <- flights %>%
  group_by(carrier) %>%
  summarise(
    no_of_flights = n(),
    average_delay = mean(dep_delay + arr_delay, na.rm = TRUE),
    median_delay = median(dep_delay + arr_delay, na.rm = TRUE),
    delay_IQR = IQR(dep_delay + arr_delay, na.rm = TRUE),
    delay_variance = sd(dep_delay + arr_delay, na.rm = TRUE)
)

arrange(delay_data, average_delay, median_delay, delay_variance)
```

```
# A tibble: 16 x 6
   carrier no_of_flights average_delay median_delay delay_IQR
   <chr>           <int>         <dbl>        <dbl>     <dbl>
 1 AS                714         -4.10          -20      40
 2 HA                342         -2.01          -17      33.8
 3 US              20536          5.87          -10      29
 4 AA              32729          8.93          -12      37
 5 DL              48110         10.9           -10      34
 6 VX               5162         14.5            -9      38
 7 UA              58665         15.6            -5      42
 8 MQ              26397         21.2            -4      44
 9 B6              54635         22.4            -4      45
10 9E              18460         23.8            -9      55
11 OO                 32         24.5           -13      30
12 WN              12275         27.3            -1      46
13 YV                601         34.5            -4      67.2
```

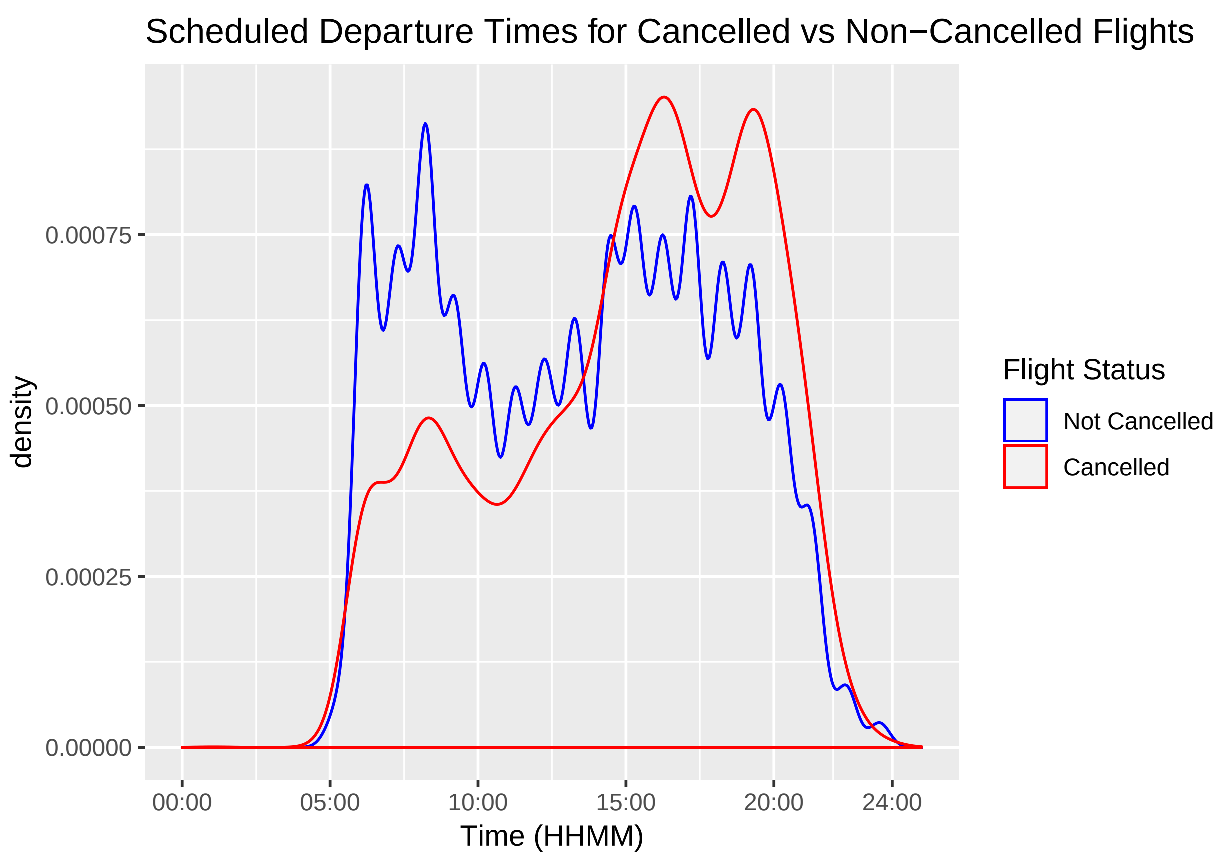
```
14 EV                54173          35.6              -1       68
15 FL                 3260          38.7               6       50
16 F9                  685          42.1               8       55
# ... with 1 more variable: delay_variance <dbl>
```

Based on the summary statistics generated, the AS carrier is the best airline.

Exercises (data visualization)

1.

```
ggplot(flights, aes(sched_dep_time, colour = is.na(dep_time))) + geom_density() +
  labs(title = "Scheduled Departure Times for Cancelled vs Non-Cancelled Flights",
       colour = "Flight Status") + scale_colour_manual(labels = c("Not Cancelled", "Cancelled"),
       values = c("blue", "red")) + scale_x_continuous("Time (HHMM)", c(0, 500, 1000, 1500, 2000, 2400)
       labels = c("00:00", "05:00", "10:00", "15:00", "20:00", "24:00"), limits = c(0, 2500))
```



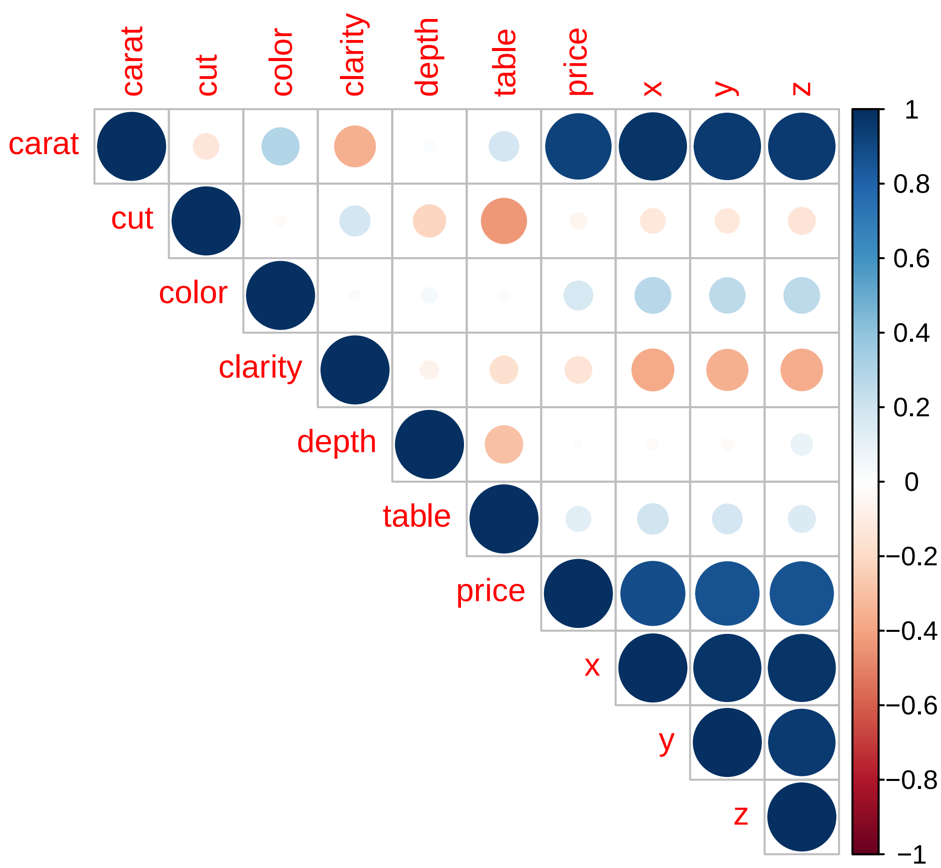
2.

```
cor_mat <- round(cor(mutate(diamonds, cut = as.numeric(cut), color = as.numeric(color),
          clarity = as.numeric(clarity))), 4)
cor_mat
```

```
          carat     cut   color clarity   depth   table   price       x
carat    1.0000 -0.1350  0.2914 -0.3528  0.0282  0.1816  0.9216  0.9751
cut     -0.1350  1.0000 -0.0205  0.1892 -0.2181 -0.4334 -0.0535 -0.1256
color    0.2914 -0.0205  1.0000  0.0256  0.0473  0.0265  0.1725  0.2703
clarity -0.3528  0.1892  0.0256  1.0000 -0.0674 -0.1603 -0.1468 -0.3720
```

```
depth    0.0282 -0.2181  0.0473 -0.0674  1.0000 -0.2958 -0.0106 -0.0253
table    0.1816 -0.4334  0.0265 -0.1603 -0.2958  1.0000  0.1271  0.1953
price    0.9216 -0.0535  0.1725 -0.1468 -0.0106  0.1271  1.0000  0.8844
x        0.9751 -0.1256  0.2703 -0.3720 -0.0253  0.1953  0.8844  1.0000
y        0.9517 -0.1215  0.2636 -0.3584 -0.0293  0.1838  0.8654  0.9747
z        0.9534 -0.1493  0.2682 -0.3670  0.0949  0.1509  0.8612  0.9708
               y       z
carat    0.9517  0.9534
cut     -0.1215 -0.1493
color    0.2636  0.2682
clarity -0.3584 -0.3670
depth   -0.0293  0.0949
table    0.1838  0.1509
price    0.8654  0.8612
x        0.9747  0.9708
y        1.0000  0.9520
z        0.9520  1.0000
```

```r
corrplot(cor_mat, type = "upper")
```



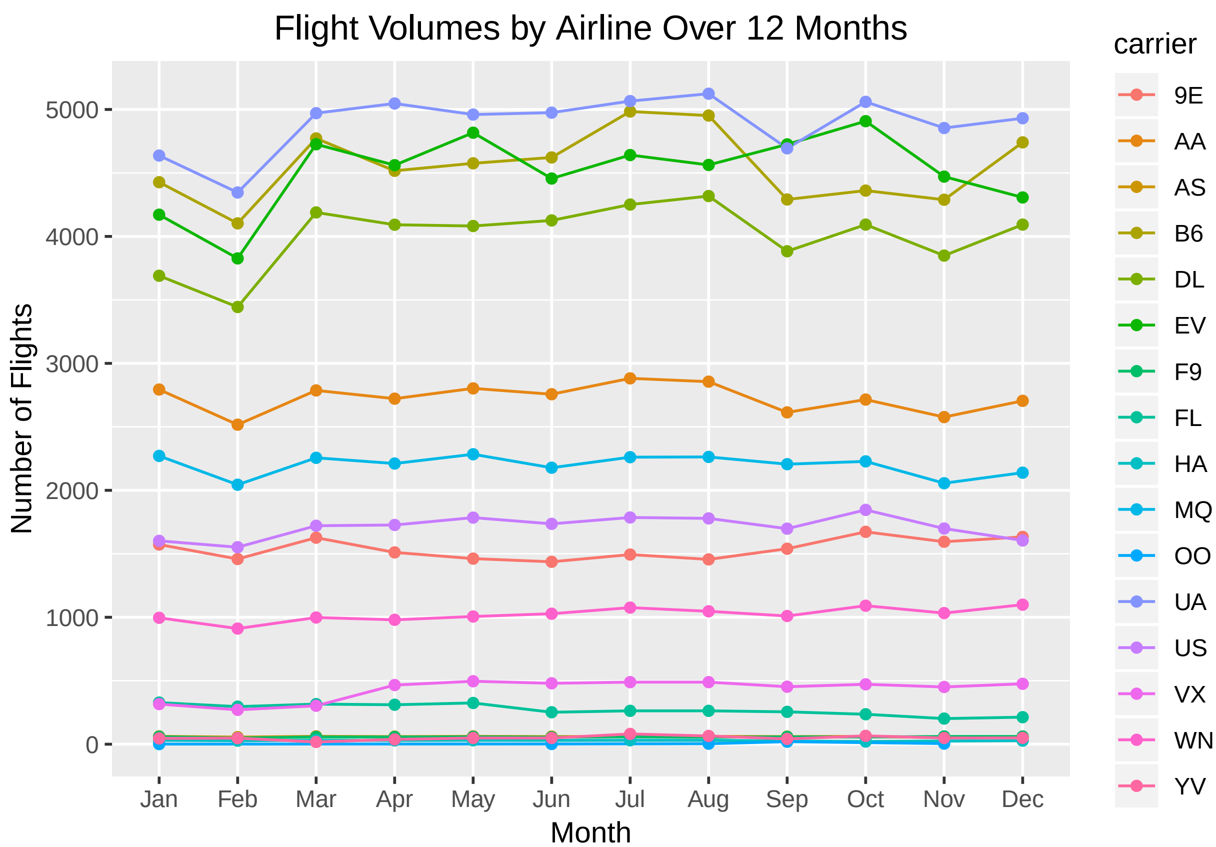
The carat of a diamond is the most important variable for predicting its price. Carat has a weak negative correlation with the cut of a diamond. As a result some diamonds which have a higher carat value (weigh more) but have a less valuable cut (lower quality) may be more expensive than lower carat diamonds with a significantly better cut.

3.

```
flights_by_month <- flights%>%
  group_by(month, carrier) %>%
  summarise(
    no_of_flights = n()

  )

ggplot(data = flights_by_month, mapping = aes(x = month.abb[month] , y = no_of_flights,
      group = carrier)) + geom_line(aes(colour = carrier)) + geom_point(aes(colour = carrier)) +
      labs(title = "Flight Volumes by Airline Over 12 Months", y = "Number of Flights", x = "Month") +
      scale_x_discrete(limits = month.abb) + theme(plot.title = element_text(hjust=0.5))
```



4.

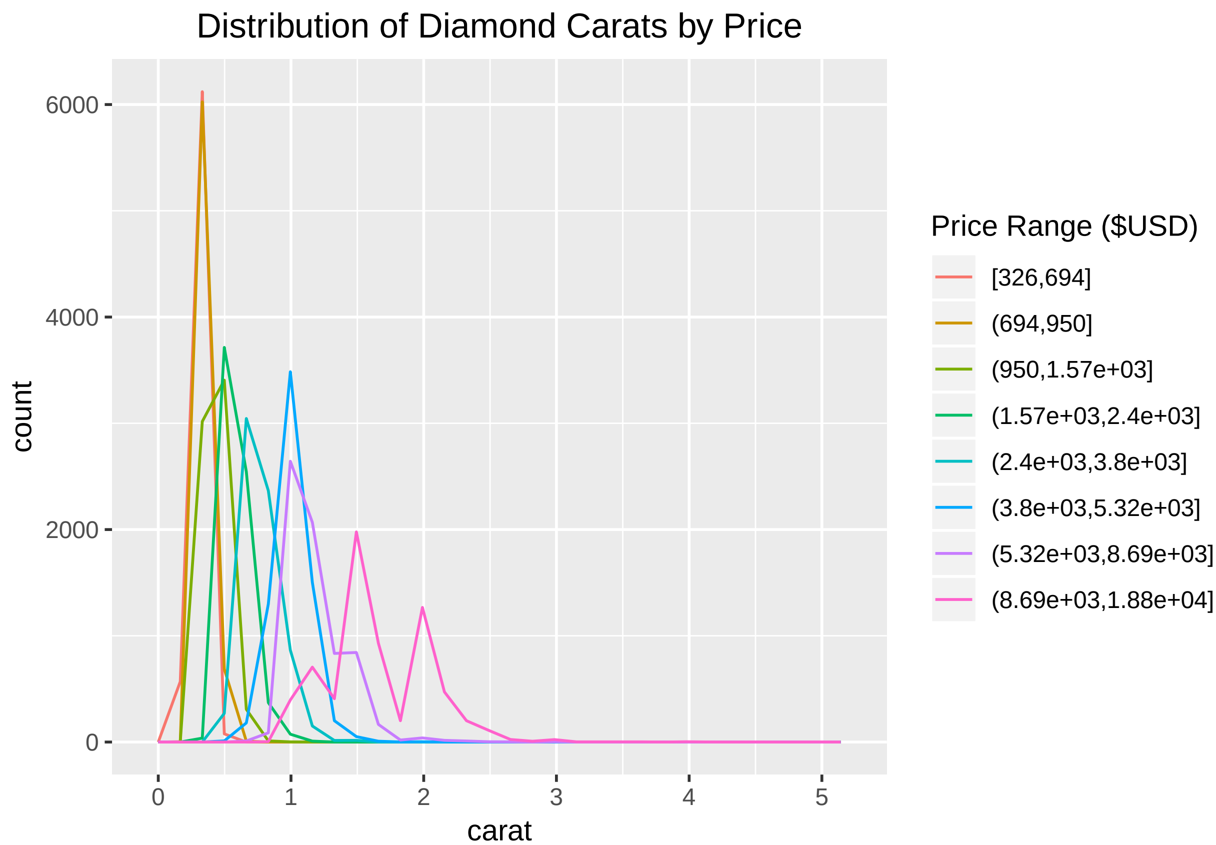
When using cut_width() vs cut_number() you need to consider the density distribution of the data. This impacts the 2D visualization of data as areas of varying data density on the graph will be more or less prominent depending on whether cut_width() or cut_data() is used.

For example if there are outlying clusters of more sparse data these will be displayed disproportionately large on the graph when using cut_number() as it forms the specified number of groups with an approximately equal number of data points. In contrast when using cut_width() these areas will be less prominent on the graph than the densely populated areas.

5.

```
ggplot(data = diamonds,  mapping = aes(x = carat)) +
  geom_freqpoly(mapping = aes(colour = cut_number(price, 8))) +
  labs(title = "Distribution of Diamond Carats by Price", colour = "Price Range ($USD)") +
```

```r
  theme(plot.title = element_text(hjust=0.5))
```



6.

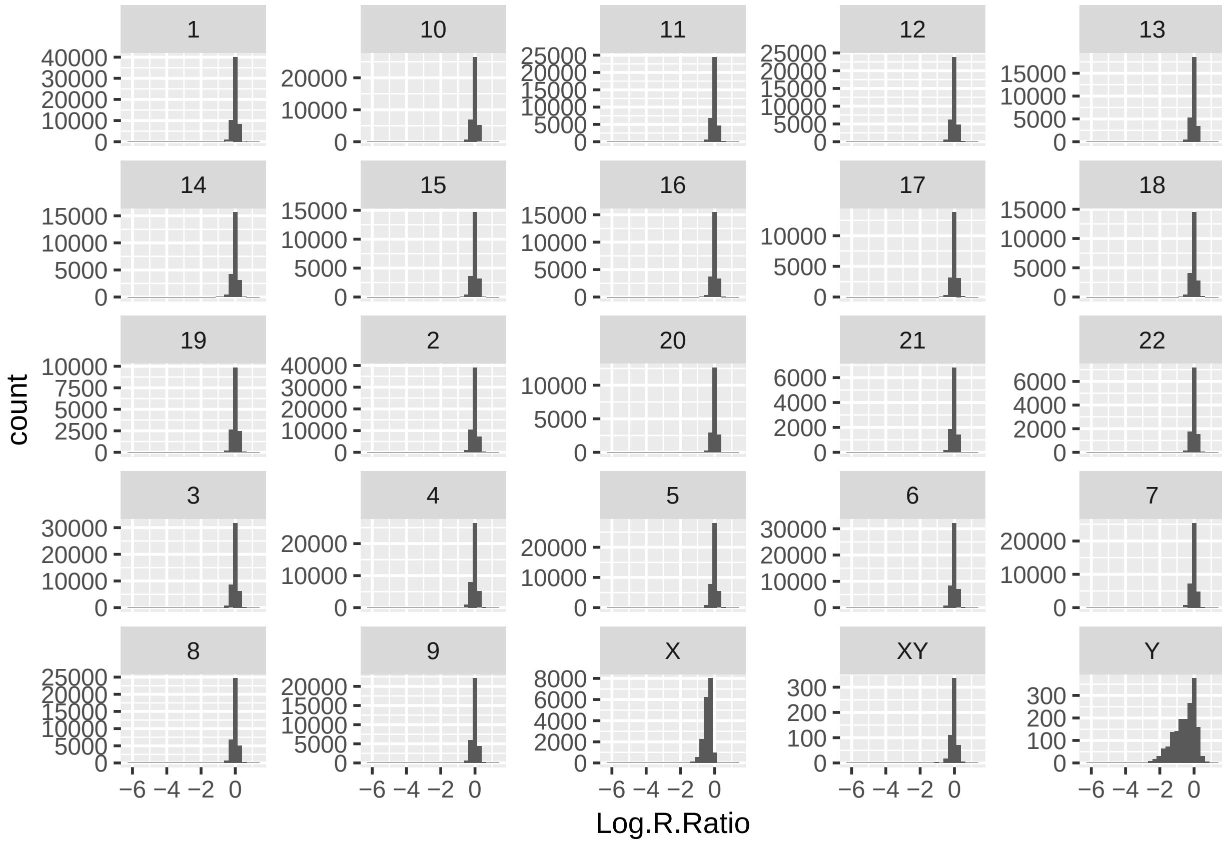
```r
gd <- read_delim("../../Master_Modelling/data/genome.txt", delim = "\t")

#Log.R.Ratio and B.Allele.Freq expected values by chromosome
gd %>%
  group_by(Chr) %>%
  summarise(
    ex.log.ratio <- mean(Log.R.Ratio, na.rm = TRUE),
    ex.allele.frew <- mean(B.Allele.Freq, na.rm = TRUE)
    )
```

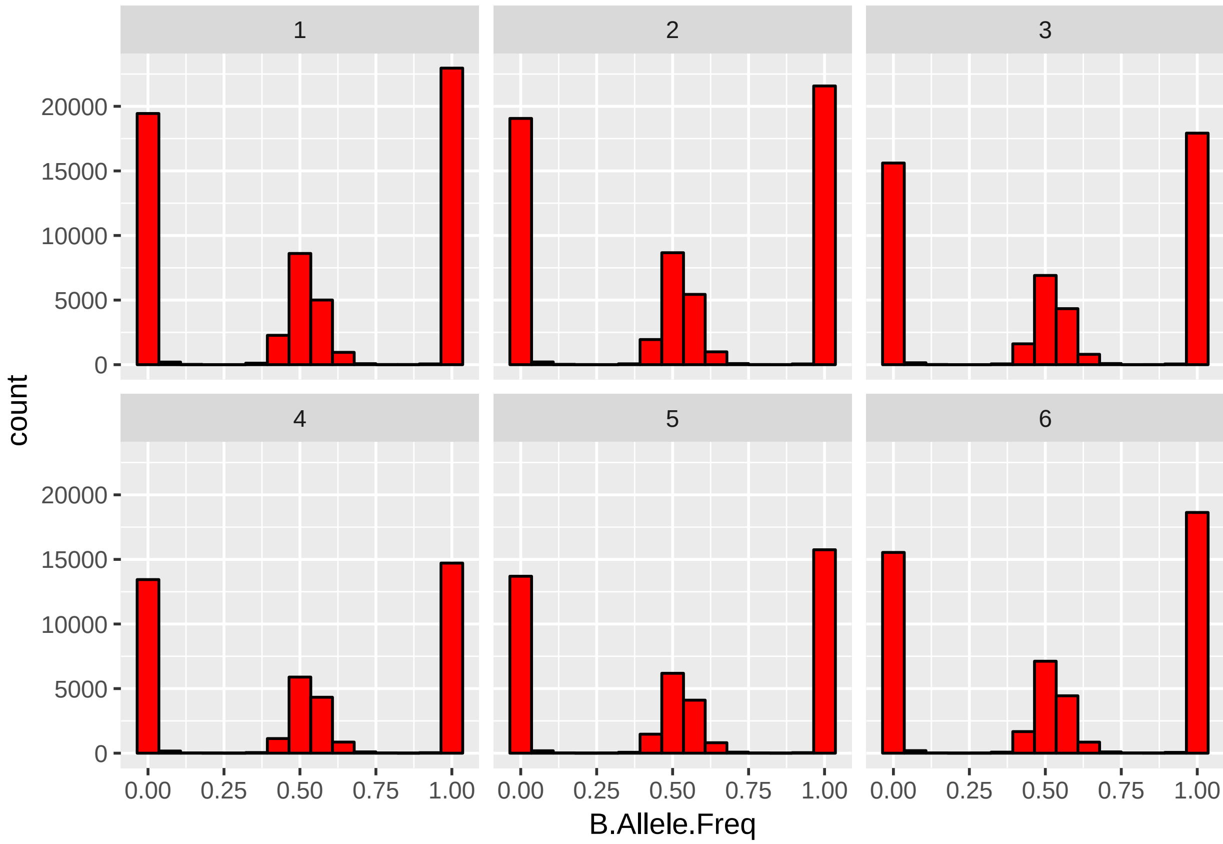
```
# A tibble: 25 x 3
   Chr   `ex.log.ratio <- mean(Log.R.Rat~ `ex.allele.frew <- mean(B.Allel~
   <chr>                            <dbl>                            <dbl>
 1 1                              -0.0171                            0.534
 2 10                             -0.0208                            0.542
 3 11                             -0.0250                            0.544
 4 12                             -0.0207                            0.533
 5 13                             -0.0281                            0.535
 6 14                             -0.0220                            0.534
 7 15                             -0.0152                            0.521
 8 16                             -0.0140                            0.542
 9 17                             -0.00849                           0.546
10 18                             -0.0243                            0.530
```

```
# ... with 15 more rows
```

```r
#Facet plot of Log.R.Ratio for each chromosome
ggplot(gd, aes(x = Log.R.Ratio)) + geom_histogram(bins = 30) +
  facet_wrap(. ~ Chr, ncol = 5, scales = "free_y")
```




```r
#Facet plot of B.Allele.Freq for chromosomes 1-6
ggplot(filter(gd, Chr %in% c(as.character(1:6))), aes(x = B.Allele.Freq)) +
  geom_histogram(colour = "black", fill = "red", bins =15) + facet_wrap(. ~ Chr, ncol = 3)
```



1
2
3
4
5
6
count
20000
15000
10000
5000
0
0.00
0.25
0.50
0.75
1.00
B.Allele.Freq