Homework 3

# 5.6.7 Exercises

## (1) 5.6.7 Exercise 1

Brainstorm at least 5 different ways to assess the typical delay characteristics of a group of flights. Consider the following scenarios:

A flight is 15 minutes early 50% of the time, and 15 minutes late 50% of the time.

A flight is always 10 minutes late.

A flight is 30 minutes early 50% of the time, and 30 minutes late 50% of the time.

99% of the time a flight is on time. 1% of the time it’s 2 hours late.

Which is more important: arrival delay or departure delay? Why?

# A flight is 15 minutes early 50% of the time, and 15 minutes late 50% of the time  
flights %>%  
 group\_by(flight) %>%  
 summarize(early=sum(arr\_delay<=-15, na.rm=TRUE)/n(), late=sum(arr\_delay>=15, na.rm=TRUE)/n()) %>%  
 filter(early==0.5, late==0.5)

## # A tibble: 18 × 3  
## flight early late  
## <int> <dbl> <dbl>  
## 1 107 0.5 0.5  
## 2 2072 0.5 0.5  
## 3 2366 0.5 0.5  
## 4 2500 0.5 0.5  
## 5 2552 0.5 0.5  
## 6 3495 0.5 0.5  
## 7 3518 0.5 0.5  
## 8 3544 0.5 0.5  
## 9 3651 0.5 0.5  
## 10 3705 0.5 0.5  
## 11 3916 0.5 0.5  
## 12 3951 0.5 0.5  
## 13 4273 0.5 0.5  
## 14 4313 0.5 0.5  
## 15 5297 0.5 0.5  
## 16 5322 0.5 0.5  
## 17 5388 0.5 0.5  
## 18 5505 0.5 0.5

# A flight is always 10 minutes late  
flights %>%  
 group\_by(flight) %>%  
 summarize(late=sum(arr\_delay==10, na.rm=TRUE)/n()) %>%  
 filter(late==1)

## # A tibble: 4 × 2  
## flight late  
## <int> <dbl>  
## 1 2254 1  
## 2 3656 1  
## 3 3880 1  
## 4 5854 1

# A flight is 30 minutes early 50% of the time, and 30 minutes late 50% of the time  
flights %>%  
 group\_by(flight) %>%  
 summarize(early=sum(arr\_delay<=-30, na.rm=TRUE)/n(), late=sum(arr\_delay>=30, na.rm=TRUE)/n()) %>%  
 filter(early==0.5, late == 0.5)

## # A tibble: 3 × 3  
## flight early late  
## <int> <dbl> <dbl>  
## 1 3651 0.5 0.5  
## 2 3916 0.5 0.5  
## 3 3951 0.5 0.5

# 99% of the time a flight is on time. 1% of the time it's 2 hours late.  
flights %>%  
 group\_by(flight) %>%  
 summarize(on\_time=sum(arr\_delay==0, na.rm=TRUE)/n(), late=sum(arr\_delay>=120, na.rm=TRUE)/n()) %>%  
 filter(on\_time==.99, late == .01)

## # A tibble: 0 × 3  
## # ... with 3 variables: flight <int>, on\_time <dbl>, late <dbl>

This is really a subjective question to answer since it depends on the one's personal preferences to answer honestly. I can see arguments for both sides.

## (2) 5.6.7 Exercise 3

Our definition of cancelled flights (is.na(dep\_delay) | is.na(arr\_delay)) is slightly suboptimal. Why? Which is the most important column?

All flights that arrived also departed, so we can use !is.na(dep\_delay).

## (3) 5.6.7 Exercise 4

Look at the number of cancelled flights per day. Is there a pattern? Is the proportion of cancelled flights related to the average delay?

flights %>%  
 filter(is.na(dep\_delay)) %>%  
 count(day)

## # A tibble: 31 × 2  
## day n  
## <int> <int>  
## 1 1 246  
## 2 2 250  
## 3 3 109  
## 4 4 82  
## 5 5 226  
## 6 6 296  
## 7 7 318  
## 8 8 921  
## 9 9 593  
## 10 10 535  
## # ... with 21 more rows

flights %>%  
 group\_by(day) %>%  
 summarize(prop\_can=sum(is.na(dep\_delay))/n(), avg\_del=mean(dep\_delay, na.rm=TRUE))

## # A tibble: 31 × 3  
## day prop\_can avg\_del  
## <int> <dbl> <dbl>  
## 1 1 0.022290685 14.172660  
## 2 2 0.023131014 14.115552  
## 3 3 0.009722594 10.811295  
## 4 4 0.007414775 5.789651  
## 5 5 0.020814146 7.820824  
## 6 6 0.026765530 6.990244  
## 7 7 0.028948566 14.339177  
## 8 8 0.081714134 21.760773  
## 9 9 0.054619140 14.644680  
## 10 10 0.047652979 18.300505  
## # ... with 21 more rows

## (4) 5.6.7 Exercise 6

For each plane, count the number of flights before the first delay of greater than 1 hour.

flights %>%  
 group\_by(tailnum) %>%  
 mutate(row\_num=row\_number()) %>%  
 filter(arr\_delay>60) %>%  
 summarize(first\_hr\_del=first(row\_num)-1)

## # A tibble: 3,371 × 2  
## tailnum first\_hr\_del  
## <chr> <dbl>  
## 1 D942DN 0  
## 2 N0EGMQ 0  
## 3 N10156 9  
## 4 N102UW 33  
## 5 N104UW 6  
## 6 N10575 0  
## 7 N105UW 34  
## 8 N107US 29  
## 9 N108UW 9  
## 10 N109UW 15  
## # ... with 3,361 more rows

This groups by the tailnum to begin with, then establishes a variable for the row number. From there, I filter to only the flights with an arr\_delay greater than an hour. I then take all the rows before this hour delayed flight.

# 5.7.1 Exercises

## (5) 5.7.1 Exercise 2

Which plane (tailnum) has the worst on-time record?

(within 5 minutes is what I say is on time)

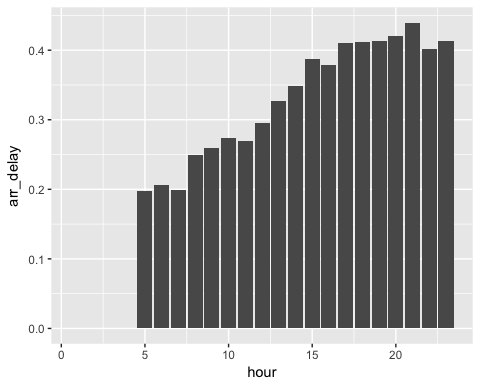
flights %>%  
 group\_by(tailnum) %>%  
 summarize(on\_time=sum(arr\_delay<=5, na.rm=TRUE)/n(), avg\_arr\_del=mean(arr\_delay, na.rm=TRUE), flights=n()) %>%  
 arrange(on\_time, desc(avg\_arr\_del))

## # A tibble: 4,044 × 4  
## tailnum on\_time avg\_arr\_del flights  
## <chr> <dbl> <dbl> <int>  
## 1 N844MH 0 320 1  
## 2 N911DA 0 294 1  
## 3 N922EV 0 276 1  
## 4 N587NW 0 264 1  
## 5 N851NW 0 219 1  
## 6 N928DN 0 201 1  
## 7 N7715E 0 188 1  
## 8 N654UA 0 185 1  
## 9 N427SW 0 157 1  
## 10 N136DL 0 146 1  
## # ... with 4,034 more rows

## (6) 5.7.1 Exercise 3

What time of day should you fly if you want to avoid delays as much as possible?

flights %>%  
 group\_by(hour) %>%  
 summarize(arr\_delay=sum(arr\_delay>5, na.rm=TRUE)/n()) %>%  
 ggplot(aes(x=hour, y=arr\_delay)) +  
 geom\_col()



Evening flights seems to have the most delay, so flying at any other time is recommended.

## (7) 5.7.1 Exercise 5

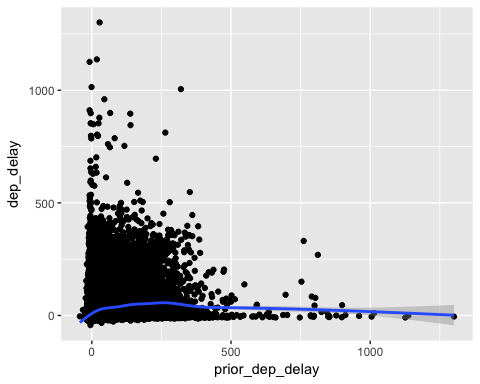
Delays are typically temporally correlated: even once the problem that caused the initial delay has been resolved, later flights are delayed to allow earlier flights to leave. Using lag(), explore how the delay of a flight is related to the delay of the immediately preceding flight.

flights %>%  
 group\_by(origin) %>%  
 arrange(year, month, day, hour, minute) %>%  
 mutate(prior\_dep\_delay=lag(dep\_delay)) %>%  
 ggplot(aes(x = prior\_dep\_delay, y=dep\_delay)) +  
 geom\_point() +  
 geom\_smooth()

## `geom\_smooth()` using method = 'gam'

## Warning: Removed 14383 rows containing non-finite values (stat\_smooth).

## Warning: Removed 14383 rows containing missing values (geom\_point).



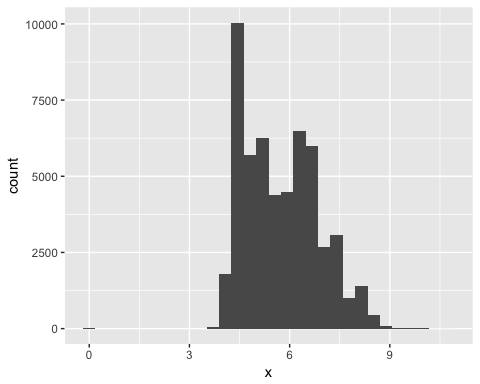
# 7.3.4 Exercises

## (8) 7.3.4 Exercise 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.

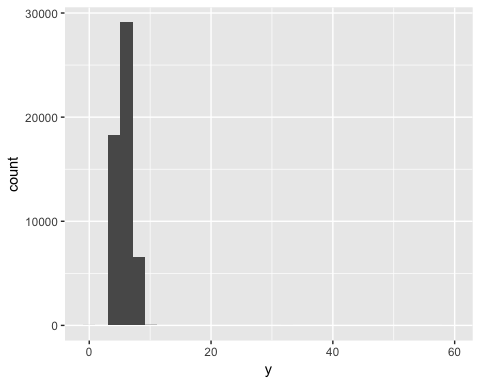
ggplot(diamonds, aes(x)) +  
 geom\_histogram()

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



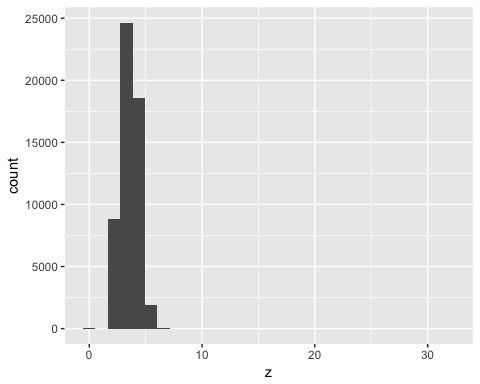
ggplot(diamonds, aes(y)) +  
 geom\_histogram()

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



ggplot(diamonds, aes(z)) +  
 geom\_histogram()

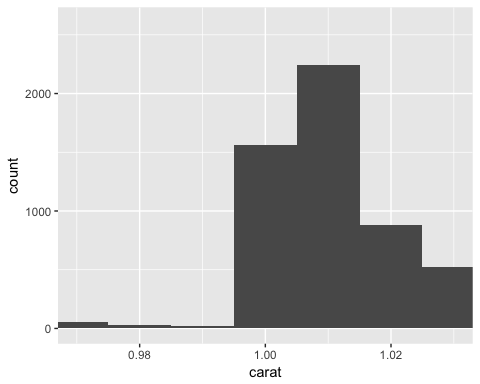
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



## (9) 7.3.4 Exercise 3

How many diamonds are 0.99 carat? How many are 1 carat? What do you think is the cause of the difference?

ggplot(diamonds, aes(carat)) +  
 geom\_histogram(binwidth=.01) +  
 coord\_cartesian(xlim=c(.97, 1.03))



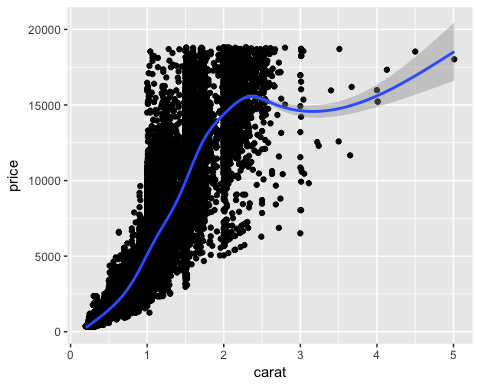
There is a much larger number of 1.00 carat diamonds purchased (1500) than there are .99 carat or close to that (32). This could be a reporting error just because it is so close or it could be because people just buy at even intervals more often.

## (10) 7.3.4 Exercise 4

Compare and contrast coord\_cartesian() vs xlim() or ylim() when zooming in on a histogram. What happens if you leave binwidth unset? What happens if you try and zoom so only half a bar shows?

# full plot  
ggplot(diamonds, aes(carat, price)) +  
 geom\_point() +  
 geom\_smooth()

## `geom\_smooth()` using method = 'gam'

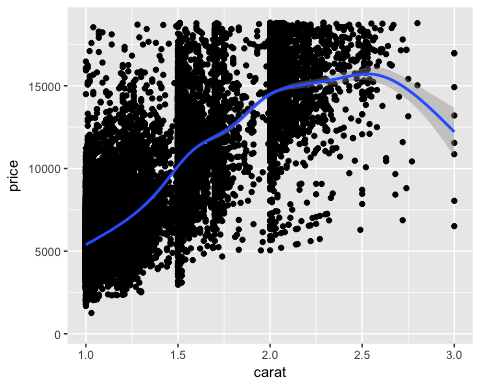


# xlim  
ggplot(diamonds, aes(carat, price)) +  
 geom\_point() +  
 geom\_smooth() +  
 xlim(1, 3)

## `geom\_smooth()` using method = 'gam'

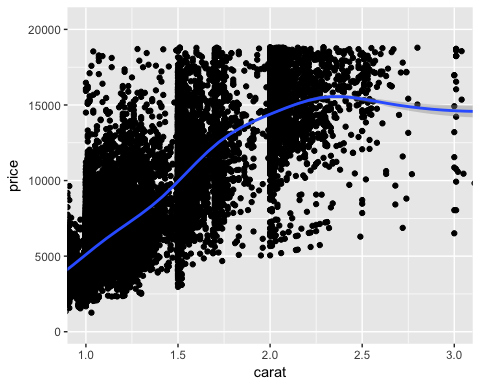
## Warning: Removed 34912 rows containing non-finite values (stat\_smooth).

## Warning: Removed 34912 rows containing missing values (geom\_point).



# coord\_cartesian  
ggplot(diamonds, aes(carat, price)) +  
 geom\_point() +  
 geom\_smooth() +  
 coord\_cartesian(xlim = c(1, 3))

## `geom\_smooth()` using method = 'gam'



xlim() and ylim() both remove observations that are not within a set interval, but coord\_cartesian() these values are all included but will be cut out if one zooms in.

# 7.4.1 Exercises

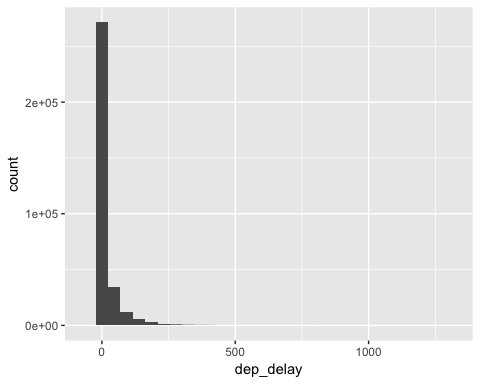
## (11) 7.4.1 Exercise 1

What happens to missing values in a histogram? What happens to missing values in a bar chart? Why is there a difference?

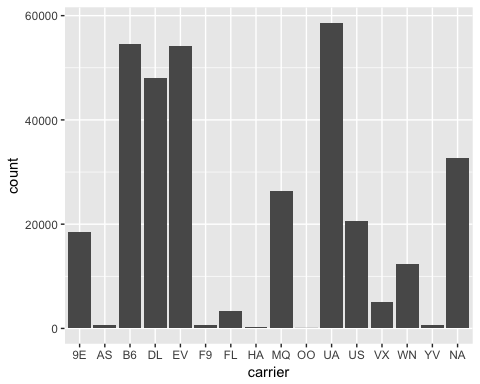
ggplot(flights, aes(dep\_delay)) +  
 geom\_histogram()

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.

## Warning: Removed 8255 rows containing non-finite values (stat\_bin).



# change AA to NA  
flights %>%  
 mutate(carrier=ifelse(carrier=="AA", NA, carrier)) %>%  
 ggplot(aes(carrier)) +  
 geom\_bar()



Bar charts will draw missing values separately where histograms will just elimate these values from the graph. There isn't much value for a missing continuous value in a histogram. Bar charts map categorical variables and thus have more purpose to graph these missing values. These values can be drawn wherever specified.

## (12) 7.4.1 Exercise 2

What does na.rm = TRUE do in mean() and sum()?

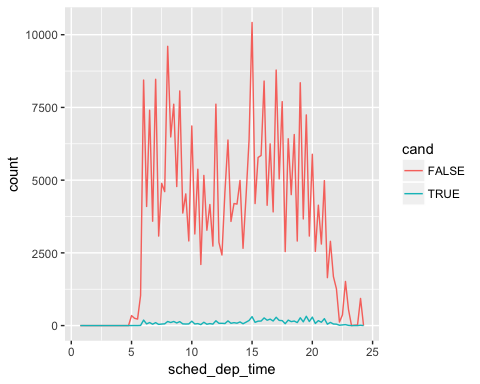
It eliminates values with NA (missing values) before calculation.

# 7.5.1.1 Exercises

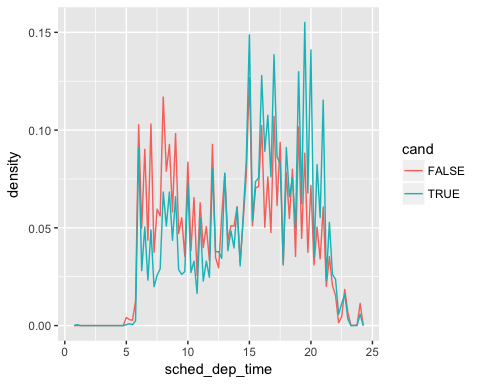
## (13) 7.5.1.1 Exercise 1

Use what you’ve learned to improve the visualization of the departure times of cancelled vs. non-cancelled flights.

# original chart  
flights %>%   
 mutate(  
 cand=is.na(dep\_time),  
 schhour=sched\_dep\_time %/% 100,  
 schmin = sched\_dep\_time %% 100,  
 sched\_dep\_time = schhour+schmin/60  
 ) %>%  
 ggplot(mapping=aes(sched\_dep\_time)) +   
 geom\_freqpoly(mapping=aes(colour=cand), binwidth=1/4)



# revised chart  
flights %>%   
 mutate(  
 cand=is.na(dep\_time),  
 schhour=sched\_dep\_time %/% 100,  
 schmin=sched\_dep\_time %% 100,  
 sched\_dep\_time=schhour+schmin/60  
 ) %>%  
 ggplot(aes(x=sched\_dep\_time, y=..density.., color=cand)) +   
 geom\_freqpoly(binwidth=1/4)



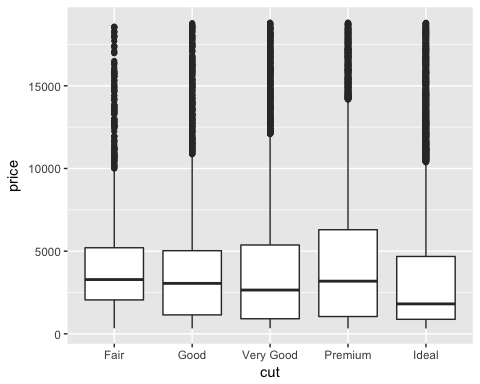
## (14) 7.5.1.1 Exercise 4

One problem with boxplots is that they were developed in an era of much smaller datasets and tend to display a prohibitively large number of “outlying values”. One approach to remedy this problem is the letter value plot. Install the lvplot package, and try using geom\_lv() to display the distribution of price vs cut. What do you learn? How do you interpret the plots?

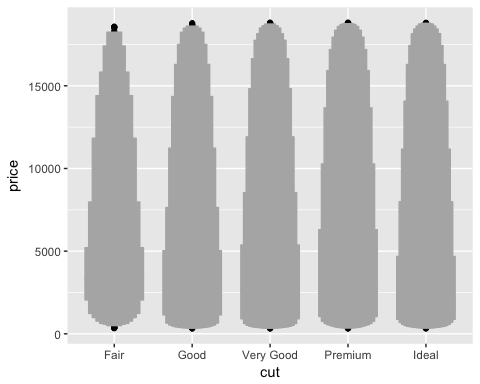
devtools::install\_github("hadley/lvplot")

## Skipping install of 'lvplot' from a github remote, the SHA1 (8ce61c77) has not changed since last install.  
## Use `force = TRUE` to force installation

library(lvplot)  
  
# with boxplot  
ggplot(diamonds, aes(cut, price)) +  
 geom\_boxplot()



# with lvplot  
ggplot(diamonds, aes(cut, price)) +  
 geom\_lv()



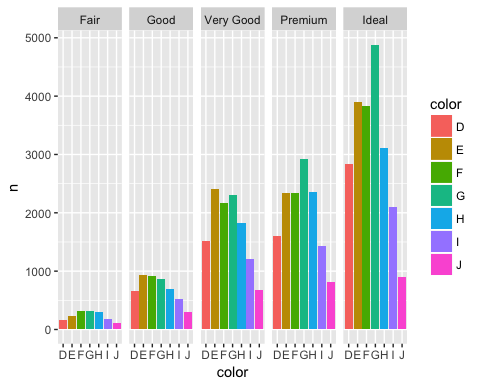
# 7.5.2.1 Exercises

## (15) 7.5.2.1 Exercise 1

How could you rescale the count dataset above to more clearly show the distribution of cut within color, or color within cut?

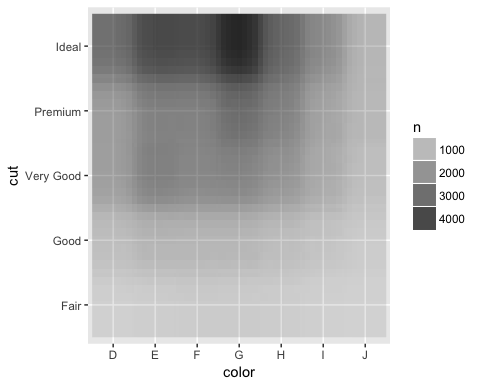
We could use a bar graph to show how the distributions across color and cut compare to one another. This is a very easy way to visualize the difference across this dataset.

diamonds %>%  
 count(color, cut) %>%  
 ggplot(mapping = aes(x = color, y = n, fill=color)) +  
 geom\_col() + facet\_grid( ~cut )



We could also use geom\_raster as it is very similar to geom\_tile and add some aesthetic changes that will change the look of the data, but it will not change much about how we interpret it.

diamonds %>%   
 count(color, cut) %>%   
 ggplot(mapping = aes(x = color, y = cut)) +  
 geom\_raster(mapping = aes(alpha = n), interpolate = TRUE)



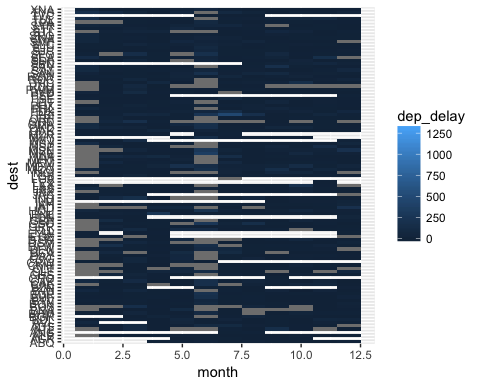
## (16) 7.5.2.1 Exercise 2

Use geom\_tile() together with dplyr to explore how average flight delays vary by destination and month of year. What makes the plot difficult to read? How could you improve it?

flights %>%  
 select(dep\_delay, month, dest)

## # A tibble: 336,776 × 3  
## dep\_delay month dest  
## <dbl> <int> <chr>  
## 1 2 1 IAH  
## 2 4 1 IAH  
## 3 2 1 MIA  
## 4 -1 1 BQN  
## 5 -6 1 ATL  
## 6 -4 1 ORD  
## 7 -5 1 FLL  
## 8 -3 1 IAD  
## 9 -3 1 MCO  
## 10 -2 1 ORD  
## # ... with 336,766 more rows

ggplot(data = flights, mapping = aes(x=month, y=dest)) +  
 geom\_tile(mapping=aes(fill=dep\_delay))



This is difficult to read because there are so many destinations that it is difficult to decipher between them. We could improve this by filtering this dataset to only those destinations that met a significant average of departure or arrival delays.

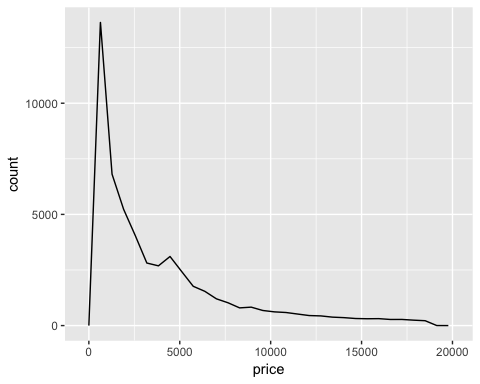
# 7.5.3.1 Exercises

## (17) 7.5.3.1 Exercise 1

Instead of summarizing the conditional distribution with a boxplot, you could use a frequency polygon. What do you need to consider when using cut\_width() vs cut\_number()? How does that impact a visualization of the 2D distribution of carat and price?

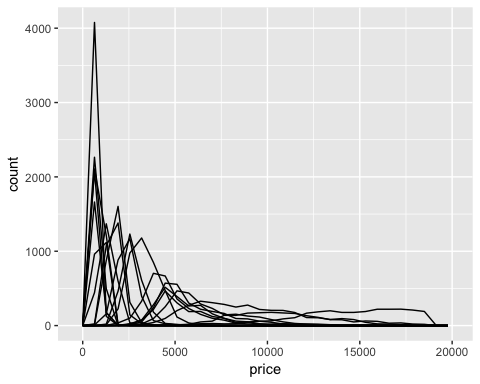
smaller <- diamonds %>% filter(carat < 3)  
ggplot(data = smaller) +   
 geom\_freqpoly(mapping = aes(x=price, group=cut\_width(carat, 20)))

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



ggplot(data = smaller) +   
 geom\_freqpoly(mapping = aes(x=price, group=cut\_number(carat, 20)))

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



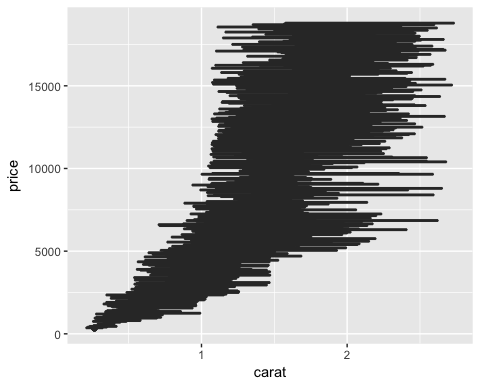
The difference between cut\_width and cut\_number is that cut\_width defines the width of the scale that the groups will be cut into whereas cut\_number defines the number of observations that will be in each group. In this case cut\_width is giving a width of 20 and cut\_number is dividing the dataset into 20 groups of equal number of observations.

## (18) 7.5.3.1 Exercise 2

Visualize the distribution of carat, partitioned by price.

ggplot(data = smaller, mapping = aes(x = carat, price)) +   
 geom\_boxplot(mapping = aes(group = cut\_width(price, 50)))

## Warning: position\_dodge requires non-overlapping x intervals



## (19) 7.5.3.1 Exercise 3

How does the price distribution of very large diamonds compare to small diamonds. Is it as you expect, or does it surprise you?

It is as I would expect as the carat gets larger the price also gets larger. As the carat size maximizes, the price drops slightly which is a little surprising.