STA 230 HW 1 Solutions

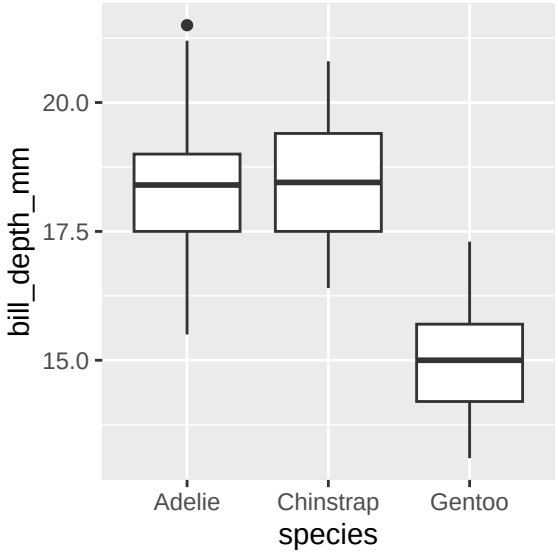
Chapter 1

Section 1.2.5

Question 4: What happens if you make a scatterplot of species vs. bill_depth_mm? What might be a better choice of geom?

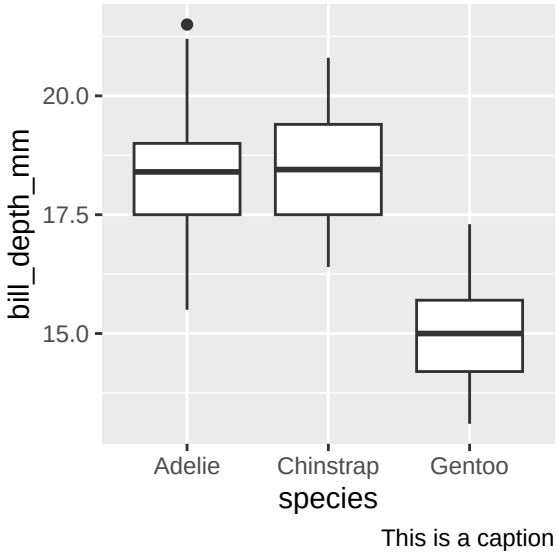
This is a categorical and quantitative variable, so a scatterplot is going to look dumb. I would use geom_point OR a boxplot

```
ggplot(penguins, aes(species, bill_depth_mm)) +
  geom_boxplot()
```



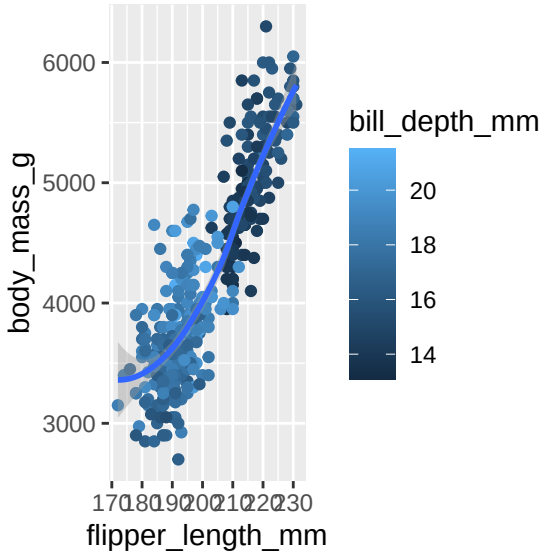
Question 7: Add the following caption to the plot you made in the previous exercise: "Data come from the palmerpenguins package." Hint: Take a look at the documentation for labs().

```
ggplot(penguins, aes(species, bill_depth_mm)) +
  geom_boxplot() +
  labs(caption = "This is a caption")
```



Question 8: Recreate the following visualization. What aesthetic should bill_depth_mm be mapped to? And should it be mapped at the global level or at the geom level?

```
ggplot(penguins, aes(x = flipper_length_mm, y = body_mass_g)) +
  geom_point(aes(color = bill_depth_mm)) +
  geom_smooth()
```



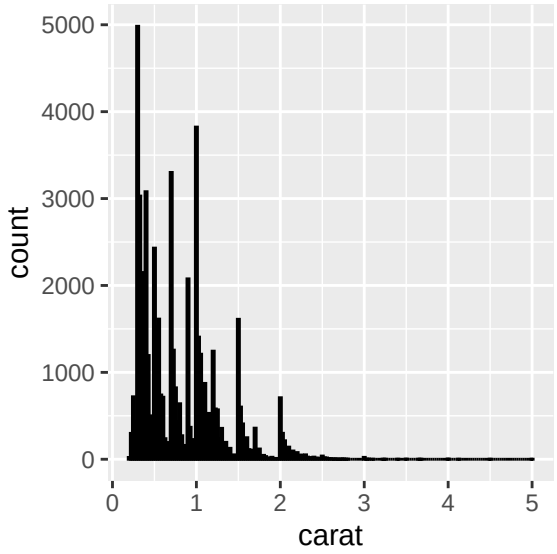
Question 10: Will these two graphs look different? Why/why not?

Nope these will be the same because they share the same aesthetics

Section 1.4.3

Question 4: I think the idea is there are a huge volume near the integer values that then taper off in between, though be generous in what people put

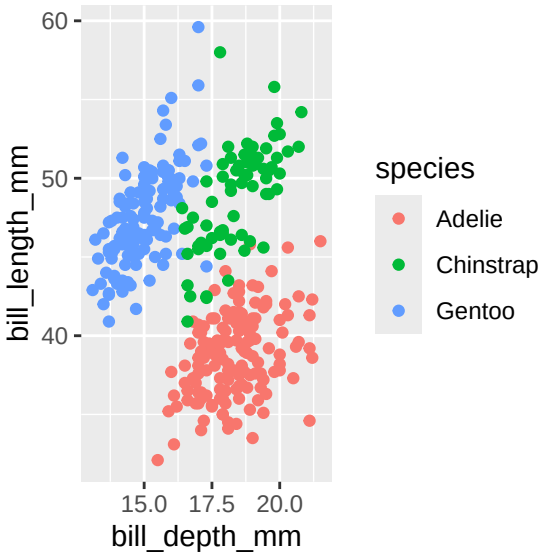
```
ggplot(diamonds, aes(carat)) +
  geom_histogram(color = "black", fill = "gray80", binwidth = 0.025)
```



Section 1.5.5

Question 5: Without color, there is no apparent relation between the two but we see when adding the color that the relationship is linear within a species

```
ggplot(penguins, aes(bill_depth_mm, bill_length_mm, color = species)) +
  geom_point()
```

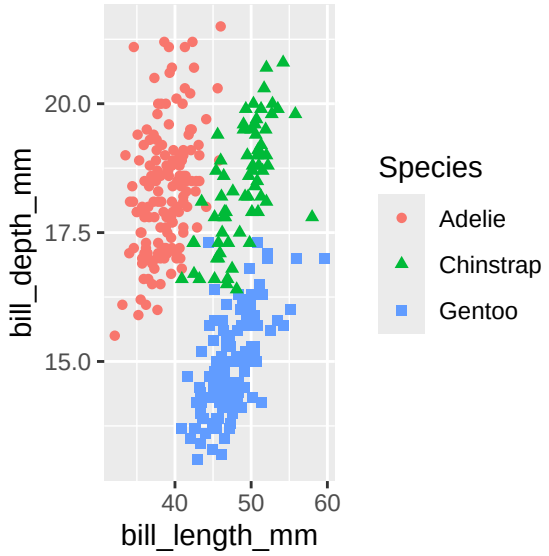


Question 6: Why does the following yield two separate legends? How would you fix it to combine the two legends?

It's because color was changed to "Species" while shape was left with lowercase

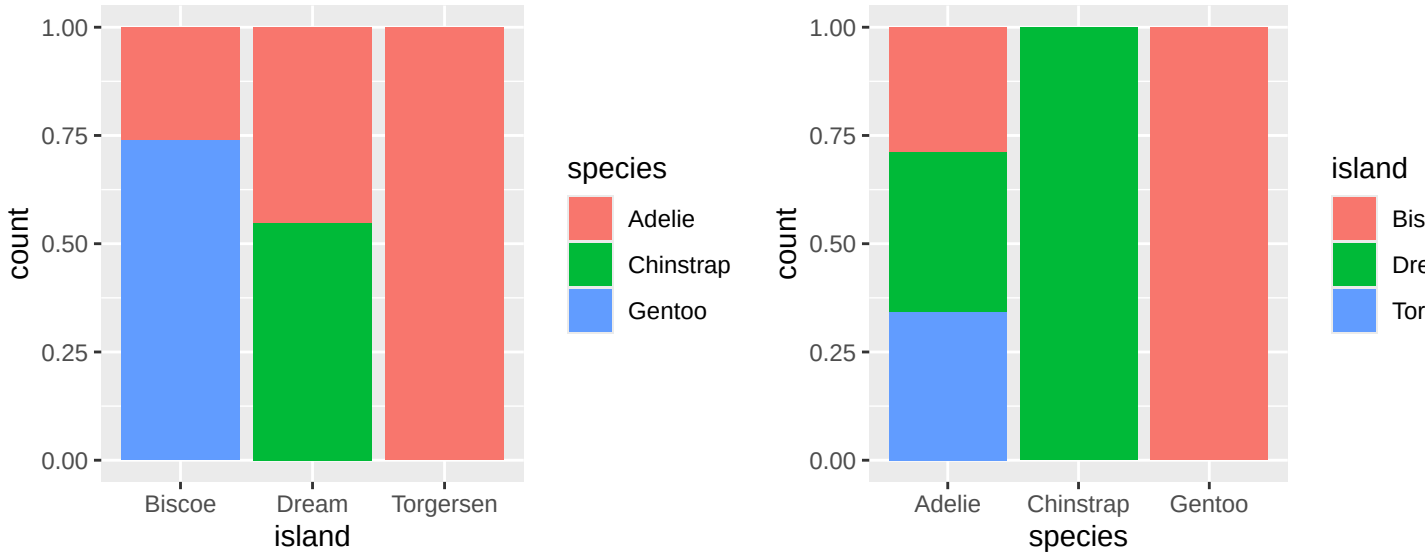
```
ggplot(
  data = penguins,
  mapping = aes(
    x = bill_length_mm, y = bill_depth_mm,
    color = species, shape = species
```

```
  )
) +
  geom_point() +
  labs(color = "Species", shape = "Species")
```



Question 7: Create the two following stacked bar plots. Which question can you answer with the first one? Which question can you answer with the second one?

```
library(gridExtra)
p1 <- ggplot(penguins, aes(x = island, fill = species)) +
  geom_bar(position = "fill")
p2 <- ggplot(penguins, aes(x = species, fill = island)) +
  geom_bar(position = "fill")
grid.arrange(p1, p2, nrow = 1)
```



From the first, you can see the composition of the islands e.g., Biscoe is mostly Gentoo. From the second, we

see that Adelie penguins are split evenly across islands, which isn't something that can be learned from plot one. Basically, we can ask questions that condition on whatever is on the x-axis

Chapter 3

Section 3.2.5

Question 1:

In a single pipeline for each condition, find all flights that meet the condition:

```
Had an arrival delay of two or more hours
Flew to Houston (IAH or HOU)
Were operated by United, American, or Delta
Departed in summer (July, August, and September)
Arrived more than two hours late but didn't leave late
Were delayed by at least an hour, but made up over 30 minutes in flight
```

```
## None of these will run, but they are given in order
filter(flights, arr_delay >= 120)
filter(flights, dest %in% c("IAH", "HOU"))
filter(flights, carrier %in% c("UA", "AA", "DL"))
filter(flights, month %in% c(7, 8, 9))
filter(flights, arr_delay > 120 & dep_delay <= 0)
filter(flights, dep_delay > 60) # and something, ill come back to this (maybe)
```

Question 2: Sort flights to find the flights with the longest departure delays. Find the flights that left earliest in the morning.

Unclear if they wanted two separate things here, kind of a stupid question tbth.

```
arrange(flights, desc(dep_delay), hour, minute)
```

```
## # A tibble: 336,776 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
##  1  2013     1     9      641            900      1301     1242           1530
##  2  2013     6    15     1432           1935      1137     1607           2120
##  3  2013     1    10     1121           1635      1126     1239           1810
##  4  2013     9    20     1139           1845      1014     1457           2210
##  5  2013     7    22      845           1600      1005     1044           1815
##  6  2013     4    10     1100           1900       960     1342           2211
##  7  2013     3    17     2321            810       911      135           1020
##  8  2013     6    27      959           1900       899     1236           2226
##  9  2013     7    22     2257            759       898      121           1026
## 10  2013    12     5      756           1700       896     1058           2020
## # i 336,766 more rows
## # i 11 more variables: 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>
```

Question 3: Sort flights to find the fastest flights. (Hint: Try including a math calculation inside of your function.)

Another dumb question because fastest is poorly defined. It could be least airtime (shortest flights) or those that were shortest relative to their scheduled flight time (as a fraction)

Because of format you cannot just subtract scheduled arrival or departure times because they are in HHMM format. In that case, let's do air time divided by distance to do speed

```
arrange(flights, air_time / distance)
```

```
## # A tibble: 336,776 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
##  1  2013     5    25     1709           1700         9     1923           1937
##  2  2013     7     2     1558           1513        45     1745           1719
##  3  2013     5    13     2040           2025        15     2225           2226
##  4  2013     3    23     1914           1910         4     2045           2043
##  5  2013     1    12     1559           1600        -1     1849           1917
##  6  2013    11    17      650            655        -5     1059           1150
##  7  2013     2    21     2355           2358        -3      412            438
##  8  2013    11    17      759            800        -1     1212           1255
##  9  2013    11    16     2003           1925        38       17             36
## 10  2013    11    16     2349           2359       -10      402            440
## # i 336,766 more rows
## # i 11 more variables: 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>
```

Question 4: Was there a flight on every day of 2013?

We know that only 2013 is included in this dataset

```
## 365, yo
distinct(flights, month, day) %>%
  count()
```

```
## # A tibble: 1 x 1
##       n
##   <int>
## 1   365
```

Question 6: Does it matter what order you used filter() and arrange() if you're using both? Why/why not? Think about the results and how much work the functions would have to do.

Yes it matters – we want to filter first to reduce the amount of computation needed to sort.

Section 3.3.5

Question 4: What does the any_of() function do? Why might it be helpful in conjunction with this vector?

any_of() is a selection function that tries to grab "any of" the elements in its argument. No idea why it's useful with that vector

Question 7: Why doesn't the following work, and what does the error mean?

```
flights |>
  select(tailnum) |>
  arrange(arr_delay)
#> Error in `arrange()`:
#> In argument: `..1 = arr_delay`.
#> Caused by error:
#> object 'arr_delay' not found
```

This doesn't work because select() removes all other columns from the data frame, meaning arr_delay doesn't exist

Section 3.5.7

Honestly, I should have investigated these questions more before assigning them, some are not interesting or clear as to what they should be. Grade this section easily

Question 1: Which carrier has the worst average delays?

```
group_by(flights, carrier) %>%
  summarize(meandep = mean(dep_delay, na.rm = TRUE)) %>%
  arrange(desc(meandep)) %>%
  slice_head(n = 5)
```

```
## # A tibble: 5 x 2
##   carrier meandep
##   <chr>     <dbl>
## 1 F9         20.2
## 2 EV         20.0
## 3 YV         19.0
## 4 FL         18.7
## 5 WN         17.7
```

Question 2: Find the flights that are most delayed upon departure to each destination.

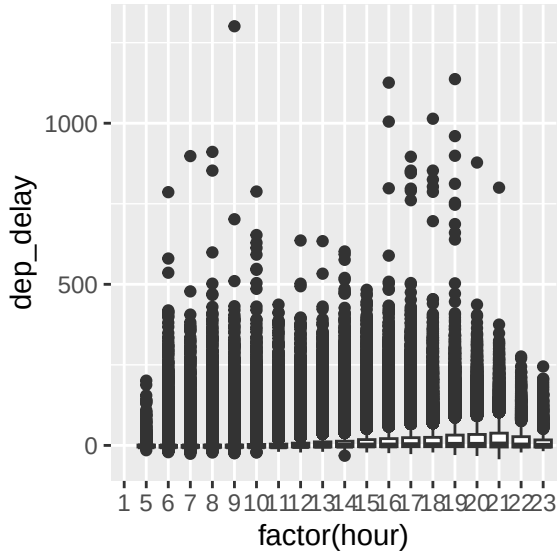
```
group_by(flights, dest) %>%
  arrange(desc(dep_delay)) %>%
  slice_head(n = 1) %>%
  select(dest, tailnum, dep_delay) %>%
  head(n = 5)
```

```
## # A tibble: 5 x 3
## # Groups:   dest [5]
##   dest  tailnum dep_delay
##   <chr> <chr>       <dbl>
## 1 ABQ   N659JB        142
## 2 ACK   N192JB        219
## 3 ALB   N13908        323
## 4 ANC   N528UA         75
## 5 ATL   N6716C        898
```

Question 3: How do delays vary over the course of the day? Illustrate your answer with a plot.

Well, if we look at distribution by hour we see that departure delays start to escalate throughout the day

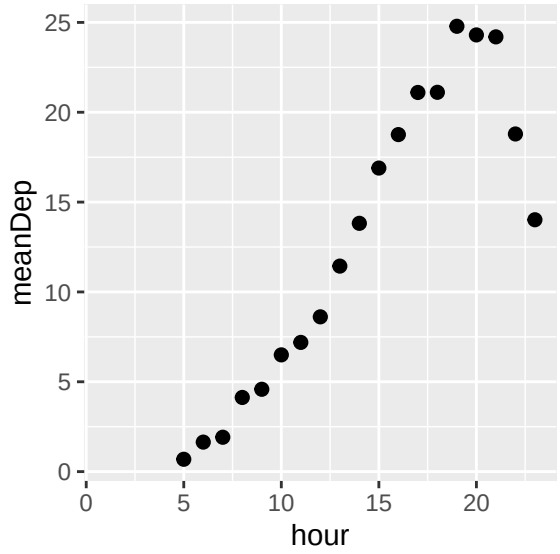
```
ggplot(flights, aes(factor(hour), dep_delay)) + geom_boxplot()
```



Could also do this, which is easier to see

```
ff <- group_by(flights, hour) %>%
  summarize(meanDep = mean(dep_delay, na.rm = TRUE))

ggplot(ff, aes(hour, meanDep)) +
  geom_point(size = 2)
```



Question 6: Whatever they put for this is fine