Assignment 3

Vishal

2022-10-12

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.3.6      v purrr   0.3.5
## v tibble  3.1.8      v dplyr   1.0.10
## v tidyr   1.2.1      v stringr 1.4.1
## v readr   2.1.3      v forcats 0.5.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(ggplot2)
library(Stat2Data)
data("Hawks")
```

1. Exploratory data analysis

1.1 (Q1)

```
HawksTail <- Hawks$Tail
HawksTail[1:6]
```

```
## [1] 219 221 235 220 157 230
```

```
mean(HawksTail)
```

```
## [1] 198.8315
```

```
median(HawksTail)
```

```
## [1] 214
```

1.2 (Q1)

```
Hawks %>%
  summarise_at(vars(Wing:Weight), c(mean, median), na.rm = T)
```

```
##   Wing_fn1 Weight_fn1 Wing_fn2 Weight_fn2
## 1 315.6375   772.0802      370        970
```

```
Hawks %>%
  summarise(Wing_mean = mean(Wing, na.rm = T),
            Wing_t_mean = mean(Wing, trim = 0.5, na.rm = T),
            Wing_med = median(Wing, na.rm = T),
            Weight_mean = mean(Weight, na.rm = T),
            Weight_t_mean = mean(Weight, trim = 0.5, na.rm = T),
            Weight_med = median(Weight, na.rm = T))
```

```
##   Wing_mean Wing_t_mean Wing_med Weight_mean Weight_t_mean Weight_med
## 1  315.6375         370      370    772.0802           970        970
```

We are calculating mean of 50% trimmed data which essentially means that we are setting aside the top and bottom half and are leaving only a single value in the middle. Which is a median value and that's why mean and median are same when we take a mean with trim value of 0.5.

1.2 (Q2)

```
Hawks %>%
  group_by(Species) %>%
  summarise(Wing_mean = mean(Wing, na.rm = T),
            Wing_t_mean = mean(Wing, trim = 0.5, na.rm = T),
            Wing_med = median(Wing, na.rm = T),
            Weight_mean = mean(Weight, na.rm = T),
            Weight_t_mean = mean(Weight, trim = 0.5, na.rm = T),
            Weight_med = median(Weight, na.rm = T))
```

```
## # A tibble: 3 x 7
##   Species Wing_mean Wing_t_mean Wing_med Weight_mean Weight_t_mean Weight_med
##   <fct>       <dbl>       <dbl>    <dbl>       <dbl>         <dbl>      <dbl>
## 1 CH           244.         240      240        420.          378.       378.
## 2 RT           383.         384      384       1094.         1070       1070
## 3 SS           185.         191      191        148.          155        155
```

1.3 (Q1)

```
X = c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
A = mean(X)

a = 2
b = 3
X_ = (a * X) + b
```

```
A_ = mean(X_)

A
```

```
## [1] 5.5
```

```
A_
```

```
## [1] 14
```

The mean of `X_` can be defined by `A_ = a * A + b`. That means, we can get the mean of newer vector by multiplying the mean of older vector by value of `a` and adding the value of `b` to the the result.

```
HawksTailMean = mean(HawksTail)
HawksTail_ = (HawksTail * a) + b
HawksTail_Mean = mean(HawksTail_)

HawksTailMean
```

```
## [1] 198.8315
```

```
HawksTail_Mean
```

```
## [1] 400.663
```

Hence, it is proved that `A_ = a * A + b`.

1.3 (Q2)

```
X = c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
varX = var(X)
sdX = sd(X)

a = 4
b = 3

X_ = (a * X) + b
varX_ = var(X_)
sdX_ = sd(X_)

varX
```

```
## [1] 9.166667
```

```
sdX
```

```
## [1] 3.02765
```

```
varX_
```

```
## [1] 146.6667
```

```
sdX_
```

```
## [1] 12.1106
```

It is seen that variance of a newer vector is square times the value of `a`. Whereas the value of standard deviation of the newer vector is `a` times the value of standard deviation of older vector.

```
varHawksTail = var(HawksTail)
sdHawksTail = sd(HawksTail)

a = 2
b = 3
HawksTail_ = (HawksTail * a) + b
varHawksTail_ = var(HawksTail_)
sdHawksTail_ = sd(HawksTail_)

varHawksTail
```

```
## [1] 1356.037
```

```
sdHawksTail
```

```
## [1] 36.8244
```

```
varHawksTail_
```

```
## [1] 5424.147
```

```
sdHawksTail_
```

```
## [1] 73.64881
```

1.4 (Q1)

```
hal <- Hawks$Hallux
hal <- hal[!is.na(hal)]
```

```
outlier_val <- 100
num_outliers <- 10
corrupted_hal <- c(hal,rep(outlier_val,times=num_outliers))
```

```r
mean(hal)
```

```
## [1] 26.41086
```

```r
mean(corrupted_hal)
```

```
## [1] 27.21776
```

```r
num_outliers_vect <- seq(0, 1000)
means_vect <- c()
for(num_outliers in num_outliers_vect){
  corrupted_hal <- c(hal,rep(outlier_val, times = num_outliers))
  means_vect <- c(means_vect, mean(corrupted_hal))
}
```

Sample Median

```r
num_outliers_vect <- seq(0, 1000)
medians_vect <- c()
for(num_outliers in num_outliers_vect) {
  corrupted_hal <- c(hal, rep(outlier_val, times = num_outliers))
  medians_vect <- c(medians_vect, median(corrupted_hal))
}
```

1.4 (Q2)

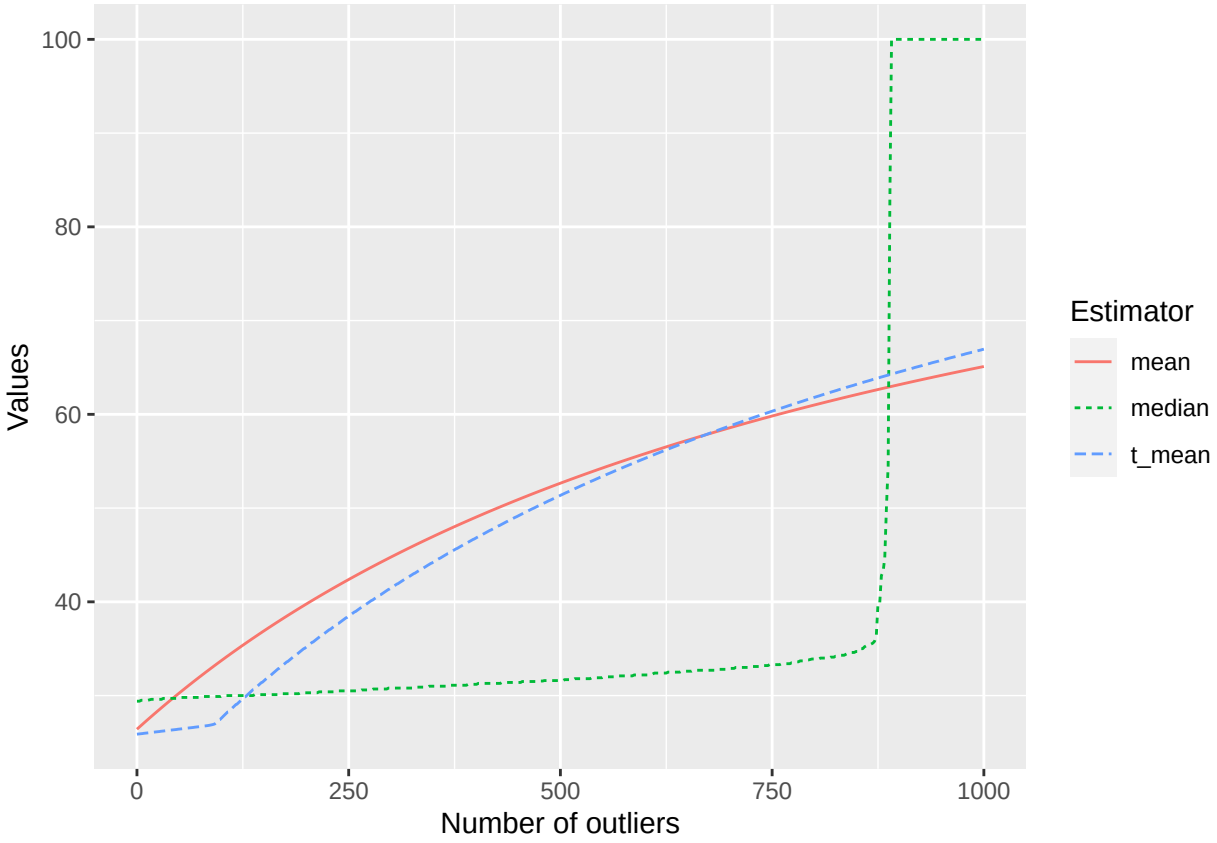
Sample trimmed mean

```r
num_outliers_vect <- seq(0, 1000)
t_means_vect <- c()
for(num_outliers in num_outliers_vect){
  corrupted_hal <- c(hal,rep(outlier_val, times = num_outliers))
  t_means_vect <- c(t_means_vect, mean(corrupted_hal, trim = 0.1))
}
```

1.4 (Q3)

```r
df_means_medians <- data.frame(num_outliers = num_outliers_vect, mean = means_vect,
                               t_mean = t_means_vect, median = medians_vect)
```

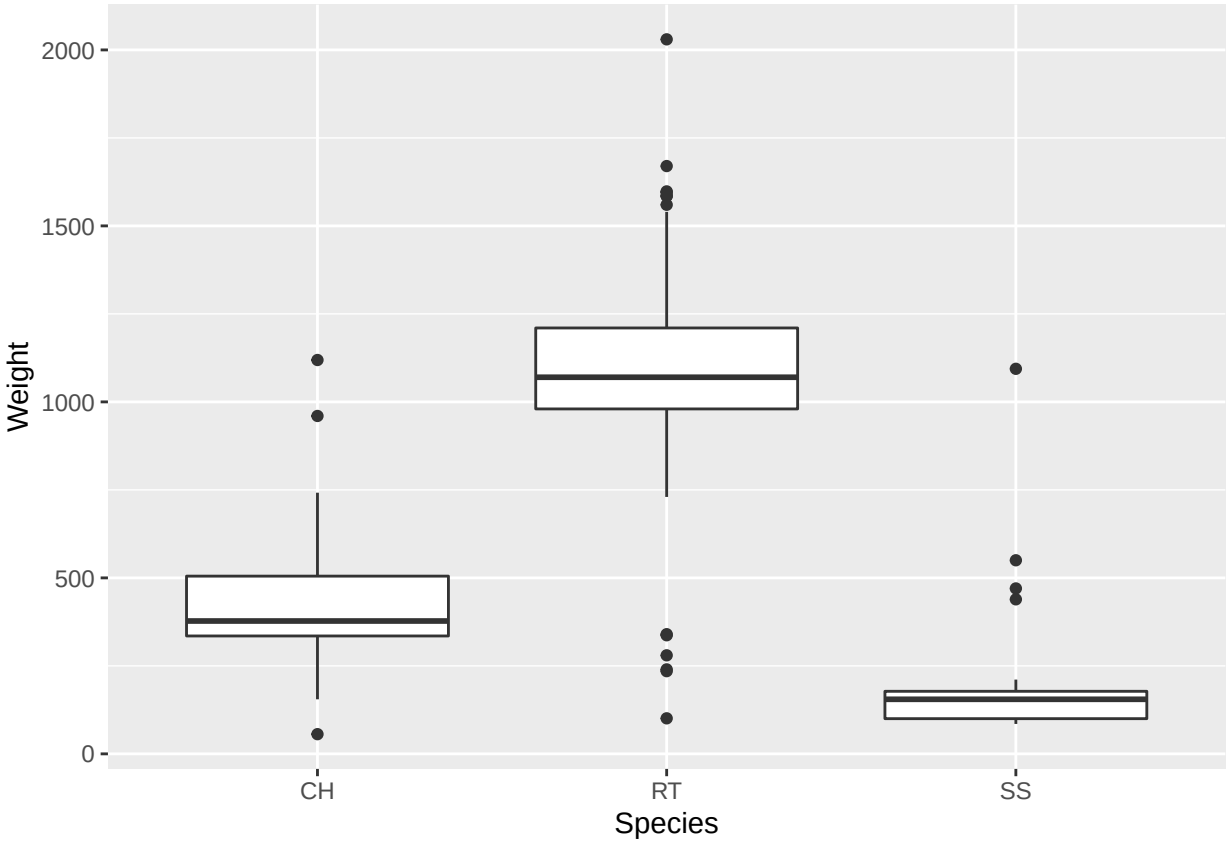
```r
df_means_medians %>%
  pivot_longer(!num_outliers, names_to = "Estimator", values_to = "Values") %>%
  ggplot(aes(x = num_outliers, color = Estimator, linetype = Estimator, y = Values)) +
  geom_line() +
  xlab("Number of outliers")
```



1.5 (Q1)

```
Hawks %>%
  group_by(Species) %>%
  ggplot(aes(x = Species, y = Weight)) +
  geom_boxplot()
```

```
## Warning: Removed 10 rows containing non-finite values (stat_boxplot).
```



1.5 (Q2)

```
q = c(.25, .50, .75)

Hawks %>%
  group_by(Species) %>%
  summarise(quantile025 = quantile(Weight, probs = q[1], na.rm = T),
            quantile050 = quantile(Weight, probs = q[2], na.rm = T),
            qunatile075 = quantile(Weight, probs = q[3], na.rm = T))
```

```
## # A tibble: 3 x 4
##   Species quantile025 quantile050 qunatile075
##   <fct>         <dbl>       <dbl>       <dbl>
## 1 CH              335        378.        505
## 2 RT              980       1070        1210
## 3 SS              100        155         178.
```

1.5 (Q3)

```
num_outliers <- function(x) {
```

```
  x<-x[!is.na(x)]

  OutliersCount = 0
  q25 = quantile(x, probs = 0.25, na.rm = T)
  q75 = quantile(x, probs = 0.75, na.rm = T)

  IQR = q75 - q25


  for (i in x) {
    if ((i < q25 - (1.5 * IQR)) || (i > q75 + (1.5 * IQR))) {
      OutliersCount = OutliersCount + 1
    }
  }
  return(OutliersCount)
}
num_outliers( c(0, 40,60,185))
```

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

1.5 (Q4)

```
Hawks %>%
  group_by(Species) %>%
  summarise(num_outliers_weight = num_outliers(Weight))
```

```
## # A tibble: 3 x 2
##   Species num_outliers_weight
##   <fct>                 <dbl>
## 1 CH                        3
## 2 RT                       13
## 3 SS                        4
```

1.6 (Q1)

```
covWeightWing <- cov(Hawks$Weight, Hawks$Wing, use = "complete.obs")
corrWeightWing <- cor(Hawks$Weight, Hawks$Wing, use = "complete.obs")

covWeightWing
```

```
## [1] 41174.39
```

```
corrWeightWing
```

```
## [1] 0.9348575
```

1.6 (Q2)

```
X <- seq(1, 10, by = 1)
Y <- seq(3, 5.7, by = 0.3)
Y
```

```
##  [1] 3.0 3.3 3.6 3.9 4.2 4.5 4.8 5.1 5.4 5.7
```

```
S <- cov(X, Y)
R <- cor(X, Y)

a = 2
b = 3
c = 5
d = 4
X_ = (a * X) + b
Y_ = (c * Y) + d

S_ = cov(X_, Y_)
R_ = cor(X_, Y_)

S
```

```
## [1] 2.75
```

```
R
```

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

```
S_
```

```
## [1] 27.5
```

```
R_
```

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

The covariance of a newer vector is `a * c` the value of covariance of a older vector. That means `S_ = S * a * c`. Whereas, the correlation stays the same although it becomes negative based on the signs of a and c.

```
a = 2.4
b = 7.1
c = -1
d = 3

X = Hawks$Weight
Y = Hawks$Wing

S = cov(X, Y, use = "complete.obs")
```

```r
R = cor(X, Y, use = "complete.obs")

X_ = (a * X) + b
Y_ = (c * Y) + d

S_ = cov(X_, Y_, use = "complete.obs")
R_ = cor(X_, Y_, use = "complete.obs")

S
```

```
## [1] 41174.39
```

```r
R
```

```
## [1] 0.9348575
```

```r
S_
```

```
## [1] -98818.54
```

```r
R_
```

```
## [1] -0.9348575
```

2. Random experiments, events and sample spaces, and the set theory

2.1 (Q1)

- **Random Experiment:** A random experiment is a procedure (real or imagined) which:

1. has a well-defined set of possible outcomes;
2. could (at least in principle) be repeated arbitrarily many times.

- **Event:** An event is a set (i.e. a collection) of possible outcomes of an experiment.
- **Sample space:** A sample space is the set of all possible outcomes of interest for a random experiment

2.1 (Q2)

Event: One or few of {1, 2, 3, 4, 5, 6}. **Sample space:** The whole set of {1, 2, 3, 4, 5, 6} **Number of different events:** {1}, {2}, {3}, {4}, {5}, {6} (In case of rolling a single dice! - 6 different events) Empty set is considered as an event. It is an impossible event.

2.2 (Q1)

1. $A \cup B = \{1, 2, 3, 4, 6\}$ $A \cup C = \{1, 2, 3, 4, 5, 6\}$
2. $A \cap B = \{2\}$ $A \cap C = \{\}$
3. $A \setminus B = \{1, 3\}$ $A \setminus C = \{1, 2, 3\}$
4. A and B are not disjoint. A and C are disjoint.
5. Yes, B and $A \setminus B$ are disjoint.
6. $\{1, 2, 3\}, \{4, 5, 6\}$ $\{1, 2\}, \{3, 4\}, \{5, 6\}$

2.2 (Q2)

1. Yes, double complement of a set produces the original set. $(A^c)^c = A$
2. $\Omega^c = \emptyset$
3. Let $A \subset B \Leftrightarrow A \cup B = B \Rightarrow (A \cup B)^{\complement} = B^{\complement}$ Using De Morgan's Law, $\Rightarrow A^{\complement} \cap B^{\complement} = B^{\complement} \Rightarrow B^{\complement} \subset A^{\complement}$
4. Let $P = (A \cap B)^{\complement}$ and $Q = A^{\complement} \cup B^{\complement}$ Let, element y belong to P. $y \in P \Rightarrow y \in (A \cap B)^{\complement} \Rightarrow y \notin (A \cap B)$ $\Rightarrow y \notin A or y \notin B \Rightarrow y \in A^{\complement} or y \in B^{\complement} \Rightarrow A^{\complement} \cup B^{\complement} \Rightarrow y \in P$ This implies that $P \subset Q$. Similarly we can show that $Q \subset P$. And combining these results will give us, $(A \cap B)^{\complement} = A^{\complement} \cup B^{\complement}$ General expression: $(A_1 \cap A_2 \cap ... \cap A_k) = (A_1^{\complement} \cup A_2^{\complement} \cap ... A_k^{\complement})$
5. We know that, $(X^{\complement})^{\complement} = X$ $A \cup B = (A^{\complement})^{\complement} \cup (B^{\complement})^{\complement}) = (A^{\complement} \cap B^{\complement})^{\complement}$ Therefore, $(A \cup B)^{\complement} = A^{\complement} \cap B^{\complement}$
6. $\emptyset$

2.2 (Q3)

$n(\Omega) = n(w_1) + n(w_2) + ... + n(w_k)$

2.2 (Q4)

If any of the set is $\emptyset$ then it is disjoint from every other set and also other conditions hold true in that case. $A \cap B = \emptyset$ for all $B \subset \Omega$.

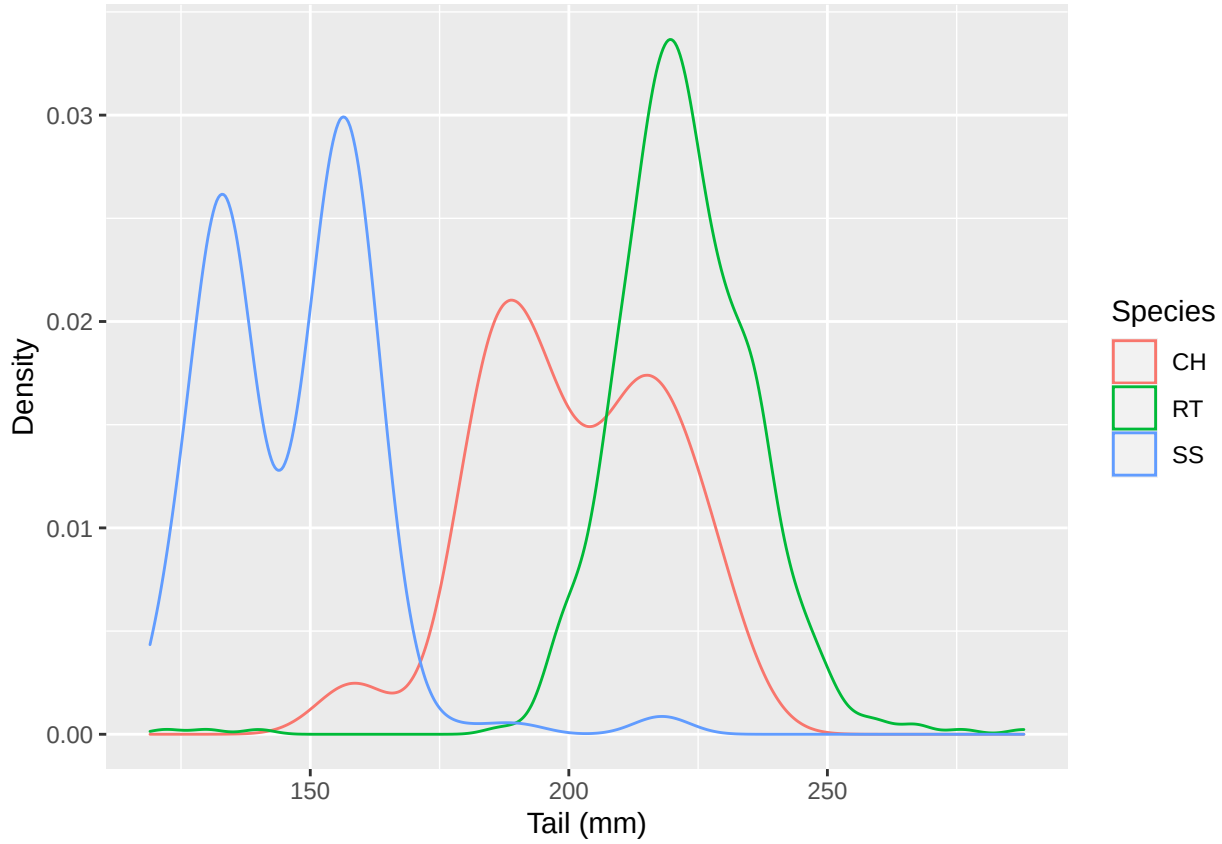
2.2 (Q5)

1. $1_A`(x) = 1 - 1_A(x)$
2. Yes, $B = \Omega - A$ or $A \cap B = \emptyset$
3. $1 - 1_{A \cup B} = (1 - 1_A)(1 - 1_B)$ i.e. $(A \cap B)^{\complement} = A^{\complement} \cup B^{\complement}$

3. Visualisation

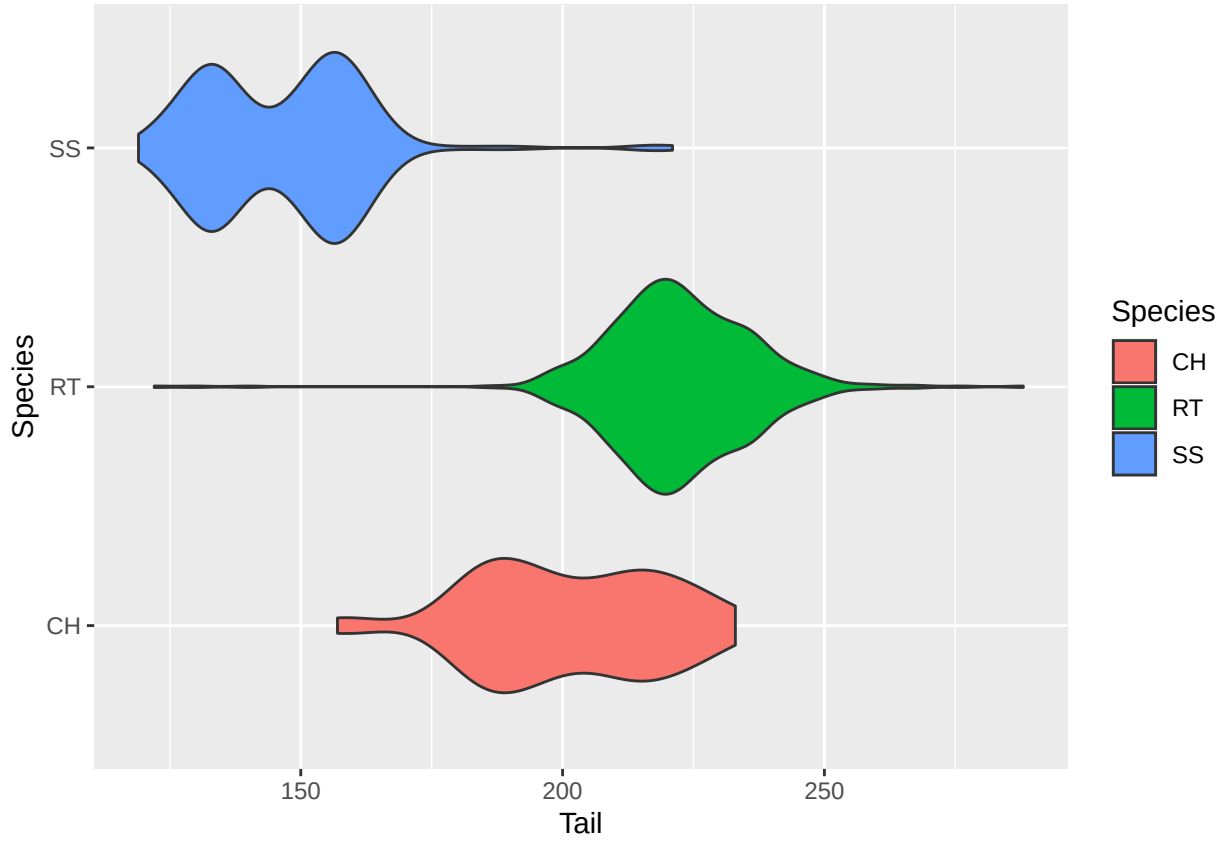
3 (Q1)

```
Hawks %>%
  group_by(Species) %>%
  ggplot(aes(x = Tail, color = Species)) +
  geom_density() +
  xlab("Tail (mm)") +
  ylab("Density")
```



3 (Q2)

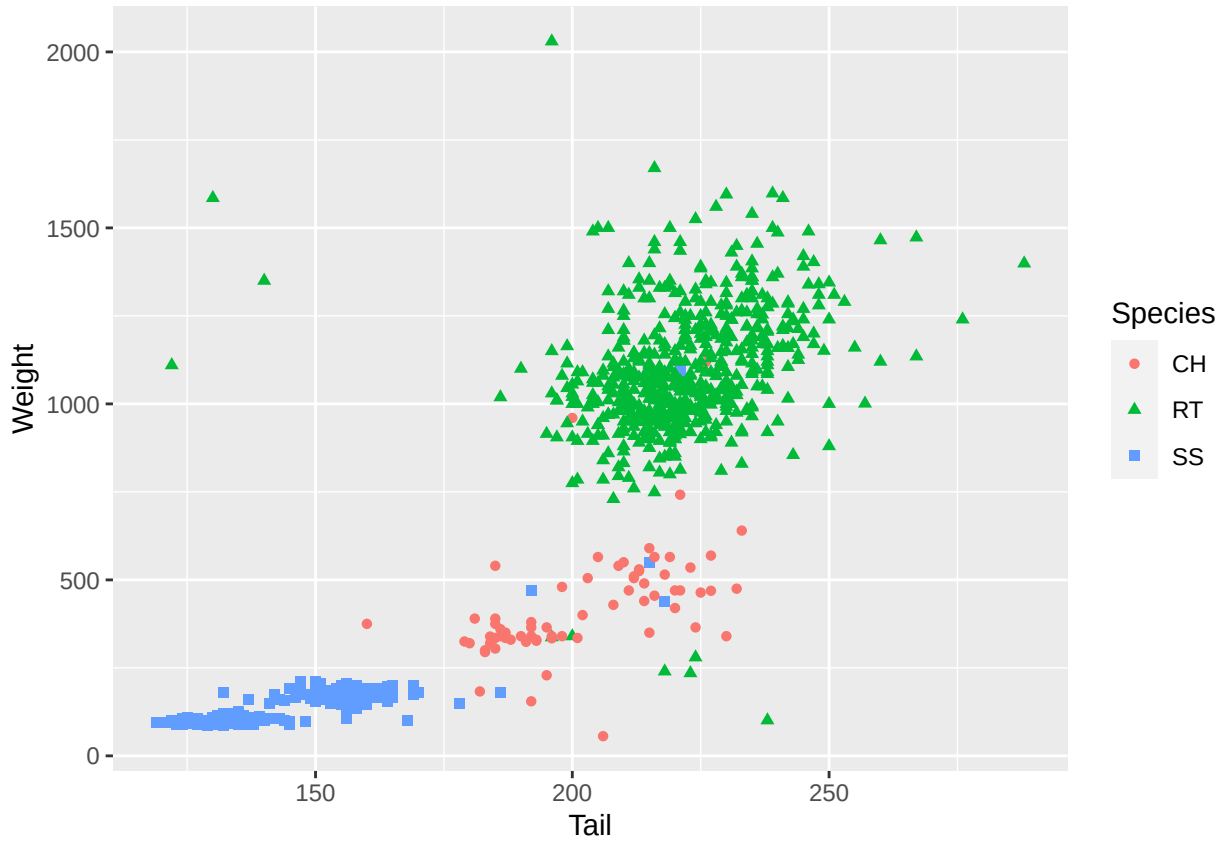
```
Hawks %>%
  group_by(Species) %>%
  ggplot(aes(x = Tail, fill = Species, y = Species)) +
  geom_violin(stat = "ydensity")
```



3 (Q3)

```
Hawks %>%
  ggplot(aes(x = Tail, y = Weight, color = Species, shape = Species)) +
  geom_point()
```

```
## Warning: Removed 10 rows containing missing values (geom_point).
```



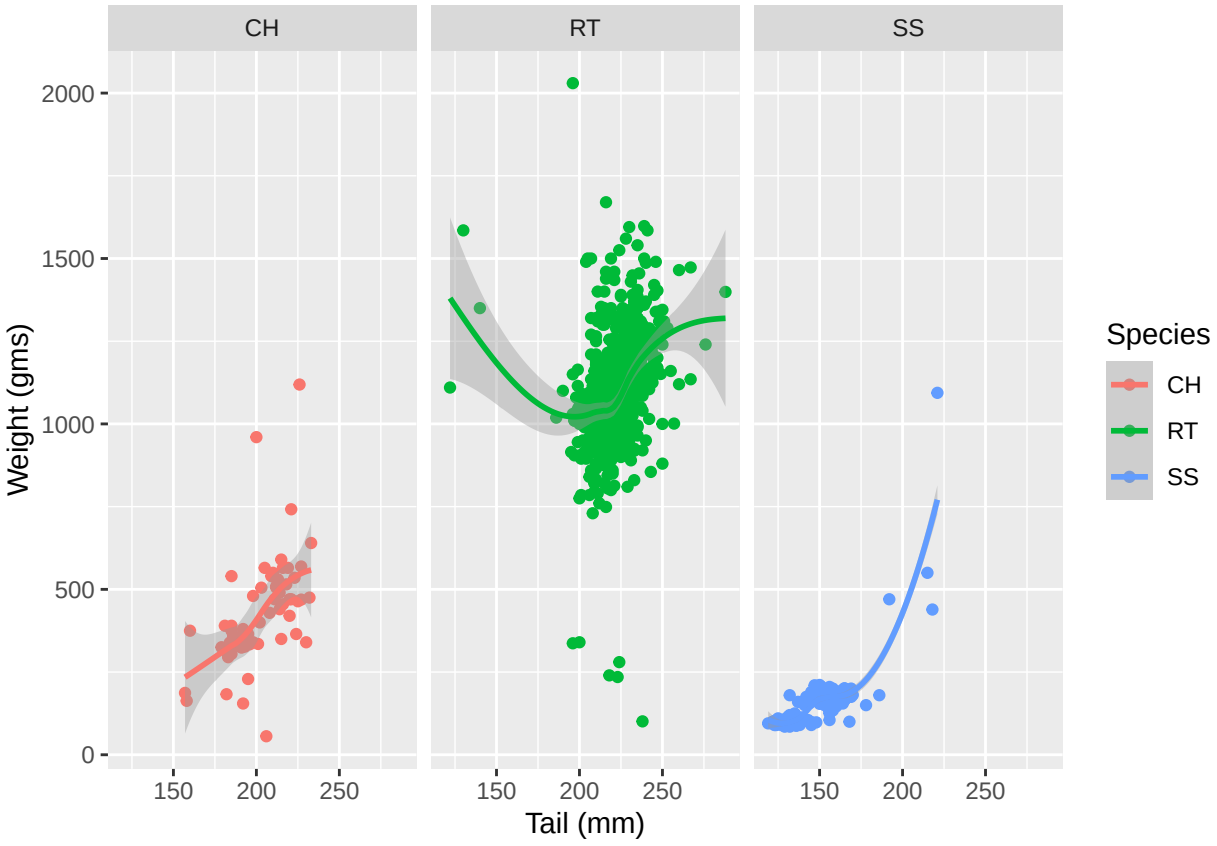
3 (Q4)

```
Hawks %>%
  ggplot(aes(x = Tail, y = Weight, color = Species)) +
  geom_point() +
  facet_wrap(vars(Species)) +
  geom_smooth() +
  xlab("Tail (mm)") +
  ylab("Weight (gms)")
```

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

```
## Warning: Removed 10 rows containing non-finite values (stat_smooth).
```

```
## Warning: Removed 10 rows containing missing values (geom_point).
```



3 (Q5)

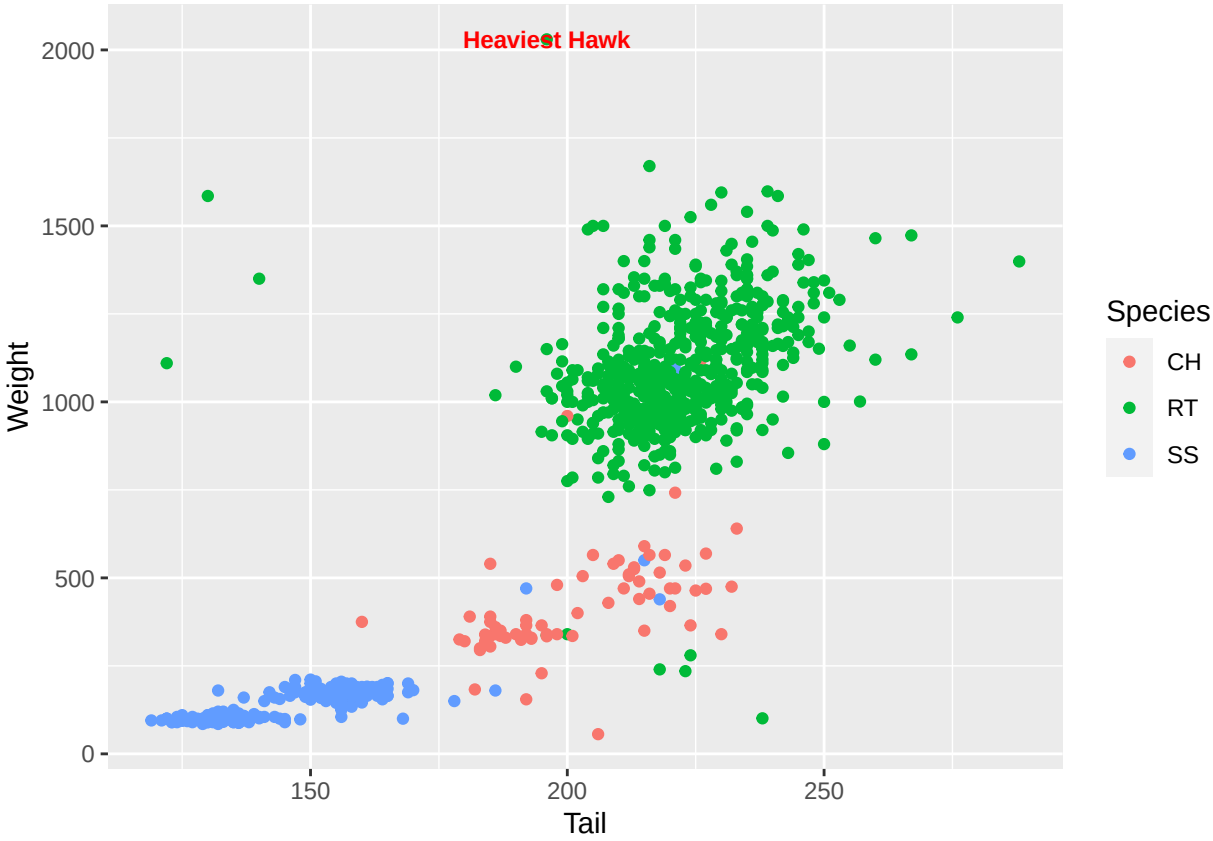
```
HeaviestHawk <- Hawks %>%
  top_n(1, Weight)
```

```
annotation <- data.frame(
   x = HeaviestHawk$Tail,
   y = HeaviestHawk$Weight,
   label = c("Heaviest Hawk")
)


p <- Hawks %>%
  ggplot(aes(x = Tail, y = Weight, color = Species)) +
  geom_point()

p + geom_text(data = annotation, aes(x=x, y=y, label=label),                 ,
           color="red",
           size=3 , angle=0, fontface="bold" )
```

```
## Warning: Removed 10 rows containing missing values (geom_point).
```




```
#p + annotate("segment", x = annotation$x, xend = annotation$x, y = annotation$y-100, yend = #annotatio
```