Assignment 3

Thrisha Rajkumar

2024-10-23

Load Packages

We need to load two packages, namely "Stat2Data" and "tidyverse", before answering the questions. If they haven't been installed on your computer, please use "install.packages()" to install them first. 1. Load the tidyverse package: 2. Load the Stat2Data package and then the dataset Hawks:

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

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

Visualization

using ggplot2

(Q1) After the dataset Stat2Data was loaded sucesfully, create a smaller dataset with the code below

```
hawksSmall<-
drop_na(select(Hawks,Age,Day,Month,Year,CaptureTime,Species,Wing,Weight,Tail))
dim(hawksSmall)
```

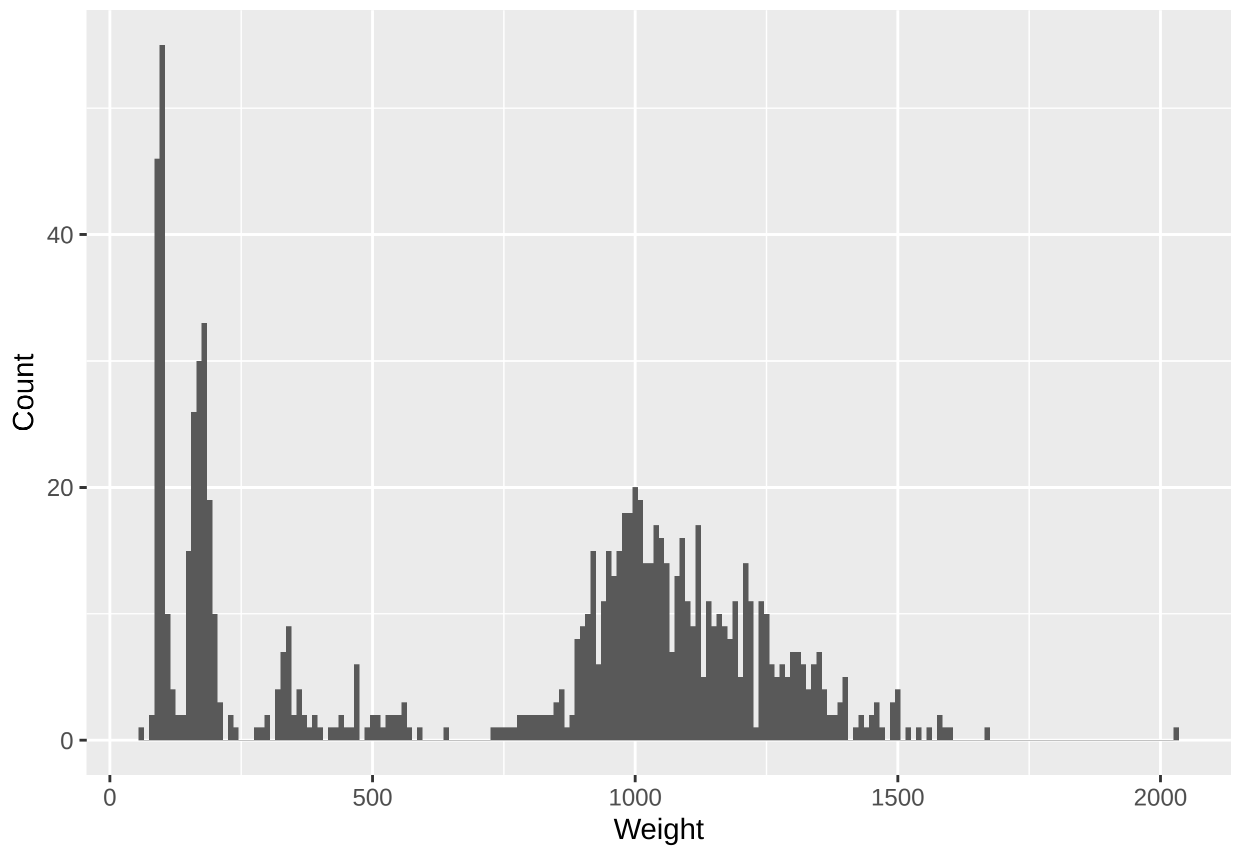
```
## [1] 897   9
```

```r
head(hawksSmall)
```

```
##   Age Day Month Year CaptureTime Species Wing Weight Tail
## 1   I  19     9 1992       13:30      RT  385    920  219
## 2   I  22     9 1992       10:30      RT  376    930  221
## 3   I  23     9 1992       12:45      RT  381    990  235
## 4   I  23     9 1992       10:50      CH  265    470  220
## 5   I  27     9 1992       11:15      SS  205    170  157
## 6   I  28     9 1992       11:25      RT  412   1090  230
```

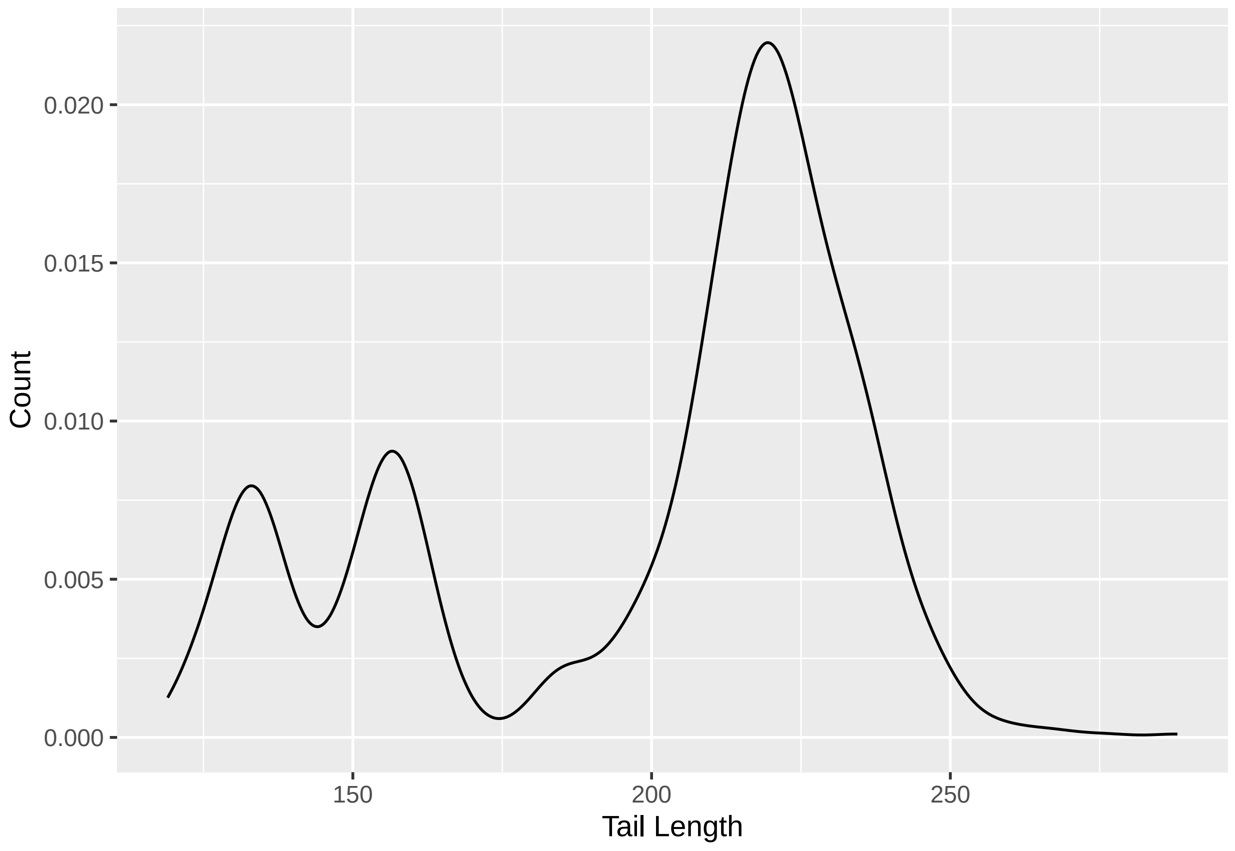
(Q2) Use a combination of the functions "ggplot()" and "geom_histogram" to create a histogram plot of the weights of Hawks within the hawksSmall data frame with bin widths of 10. Your result should look something like the following.

```r
ggplot(data=hawksSmall,aes(x=Weight))+geom_histogram(binwidth=10)+
xlab("Weight")+ylab("Count")
```

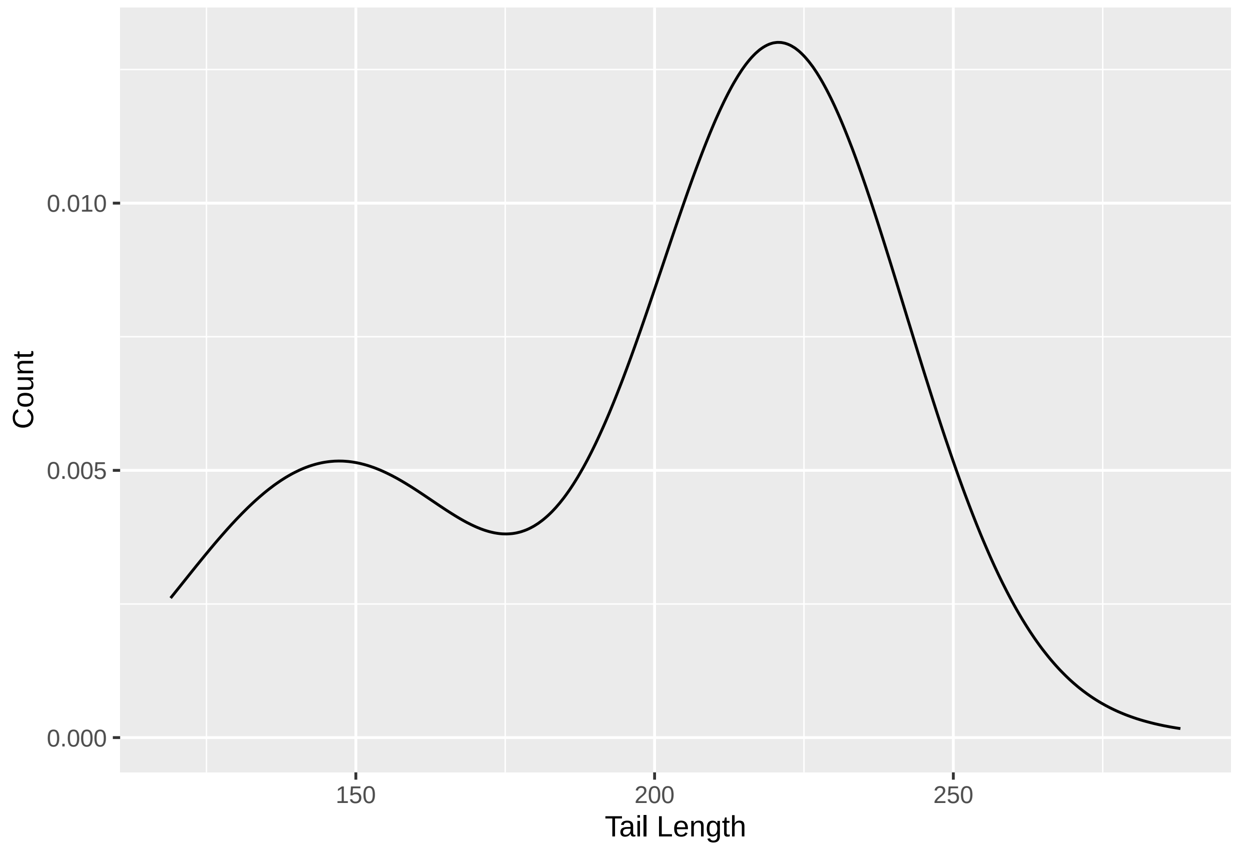


(Q3) Similar to (Q2), use the "geom_density()" function to create two density plots of the tail lengths of Hawks within the hawksSmall data frame, with parameters "adjust=0.5" and "adjust=2", respectively. Your results should look like this:

```r
ggplot(data = hawksSmall, aes(x=Tail))+geom_density(adjust=0.5)+xlab("Tail Length")+ylab("Count")
```



```r
ggplot(data = hawksSmall, aes(x=Tail))+geom_density(adjust=2)+xlab("Tail Length")+ylab("Count")
```



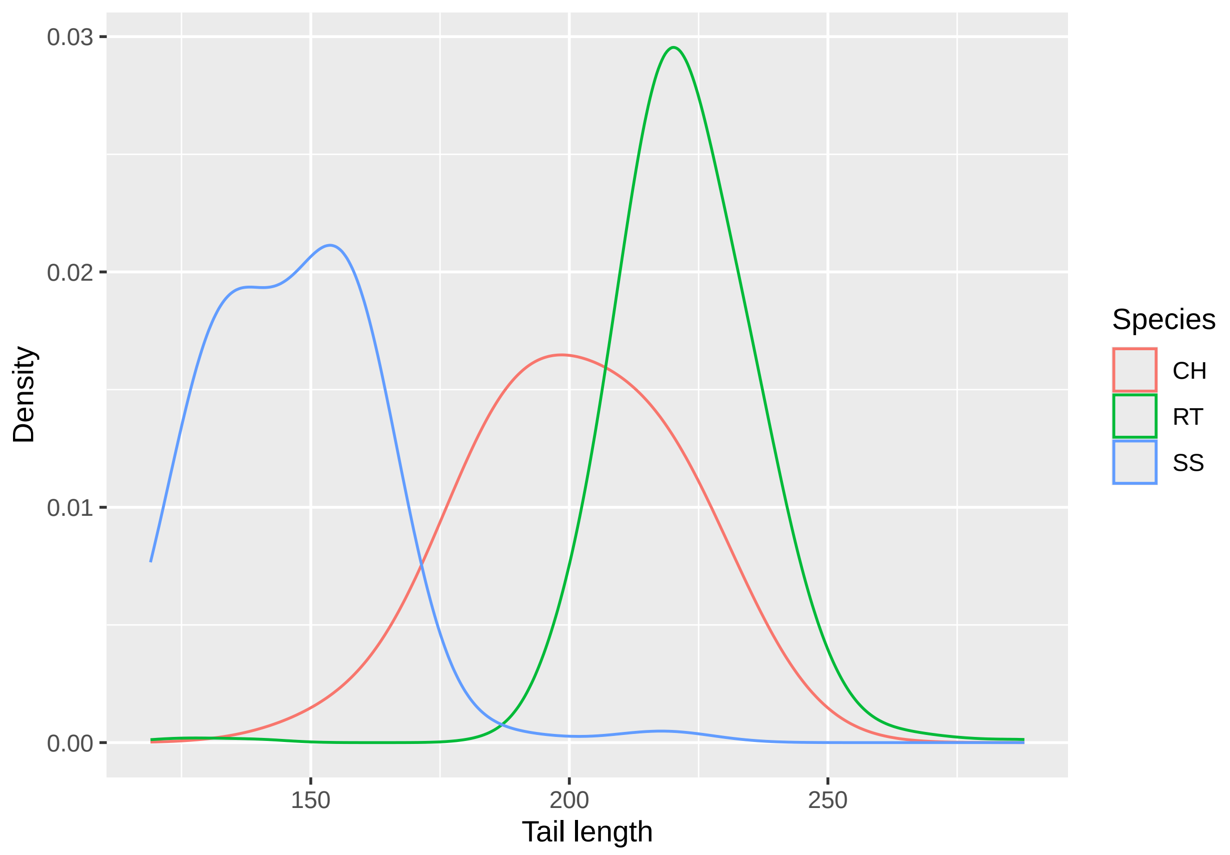
Can you explain the difference between the two plots? How many modes does the distribution of Hawk tail lengths have in each of the plots.

The second plot has a curve that is smoother version of the first plot. This is because the adjust argument in the geom_density() function adjusts the bandwidth of the density plot.A larger "adjust" corressponds to a larger bandwidth, hencce a smoother function in the plot, whilst a smaller "adjust" create a less smooth density plot which is mor esensitive to the local behavior of the data.

Three modes are visible in the first plot. Two modes are visible in the second plot.

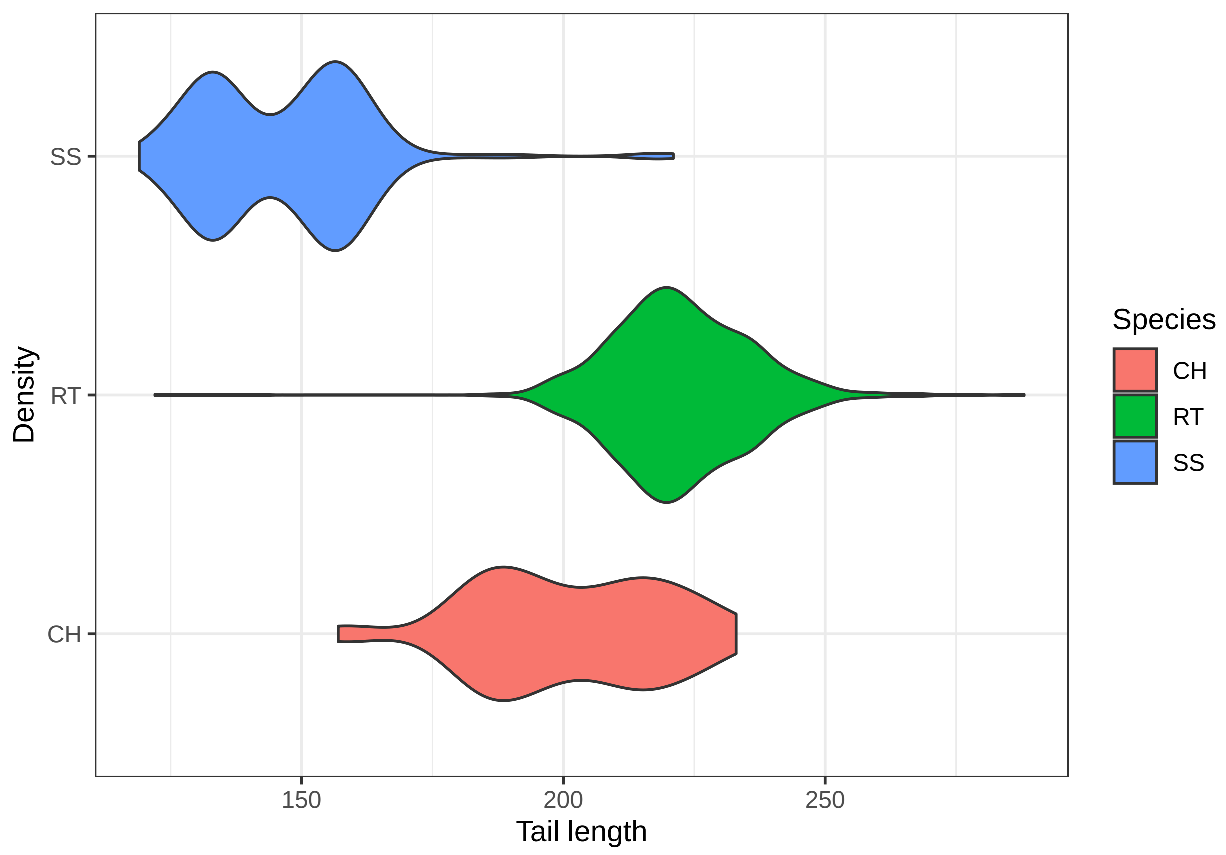
##(Q4) Based on the answer from (Q3), can you create the following plot?

```
ggplot(data=hawksSmall, aes(x=Tail, color=Species))+geom_density(adjust=2)+xlab("Tail length")+ylab("De
```



(Q5) Violin plot: Use the ggplot and geom_violin() functions to create the following violin plot for the three species.

```
ggplot(data=hawksSmall, aes(x=Tail, y=Species, fill=Species))+geom_violin()+xlab("Tail length")+ylab("De
```



(Q6) Scatter plot

Generate a plot similar to the following plot using the ggplot() and geom_point()functions.

1. How many aesthetics are present within the following plot?

There are four aesthetics: (1) The tail length is mapped to the horizontal position, (2) The weight is mapped to the vertical position, (3) The species is mapped to the colour and (4) The species is mapped to shape.

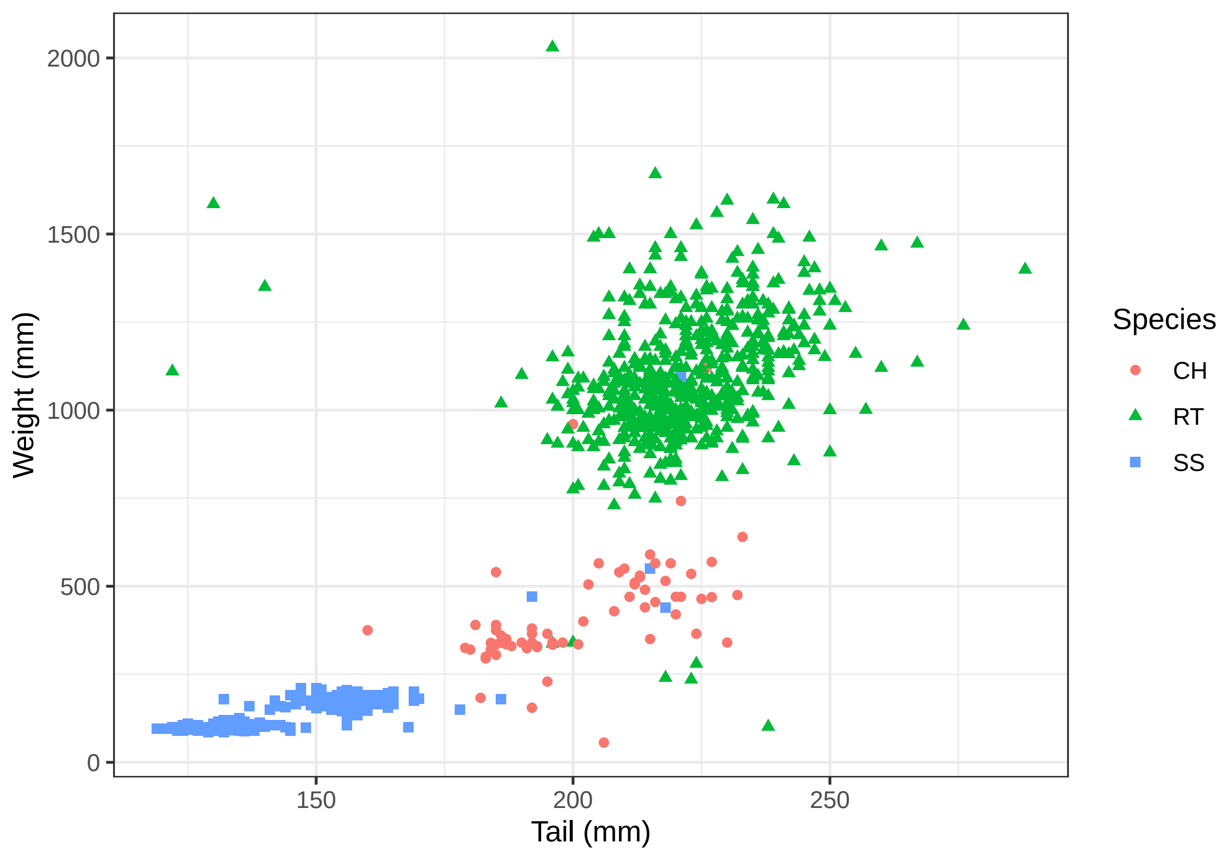
2. What are the glyphs within this plot?

The glyphs are the small elements (in this particular case represented by the shapes) corresponding to individual cases

3. What are the visual cues being used within this plot?

The visual cues include horizontal and vertical position, shape, color.

```
ggplot(hawksSmall, aes(x=Tail, y=Weight, color=Species, shape=Species)) + geom_point() + xlab('Tail (mm
```



(Q7) Trend lines and facet wraps:

##Generate the following plot using the ggplot(), geom_point(), geom_smooth() and facet_wrap() functions. Note that in the facet plot, the three panels use different scales.

1. What are the visual cues being used within this plot?

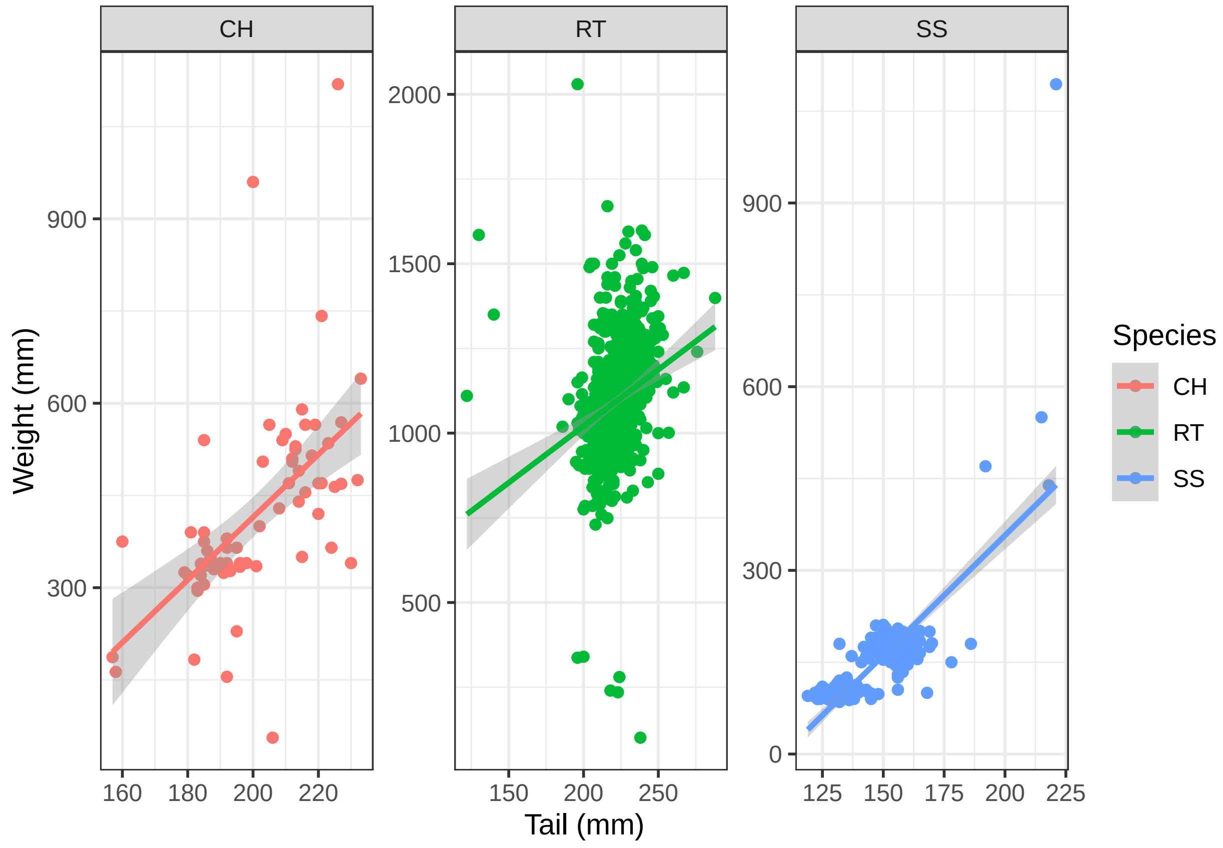
The visual cues include horizontal and vertical position, shape, color and direction.

2. Based on the plot below, what can we say about the relationship between the weight of the hawks and their tail lengths?

Based on this sample, longer tail lengths appear to be predictive of larger weights within each species.

```r
ggplot(hawksSmall, aes(x=Tail, y=Weight, color=Species)) +
  geom_point() + facet_wrap(~Species, scales = 'free') + geom_smooth(method = 'lm') + xlab('Tail (mm)')
```

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



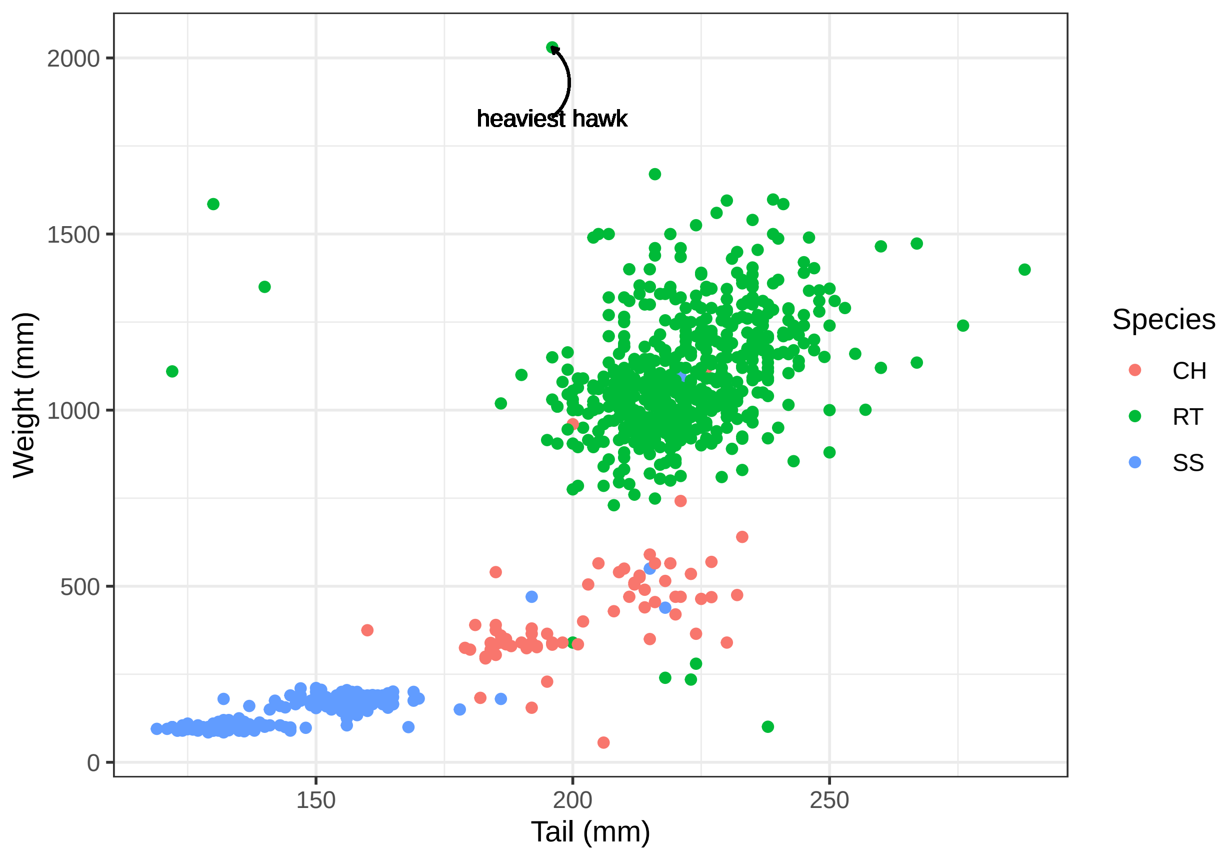
(Q8) Adding annotations

First, compute the "Weight" and the "Tail" of the heaviest hawk in the dataset. You can use filter() and select() function to select proper data.

```
max_weight<-max(hawksSmall$Weight, na.rm=TRUE)
max_weight_row = head(hawksSmall %>% filter(Weight>=max_weight), 1)
max_weight_tail = max_weight_row$Tail
```

Second, reuse the code that you create from Q(3), adding an arrow and an annotation to indicate the heaviest hawk. Your result should look similar to this:

```
ggplot(hawksSmall, aes(x=Tail, y=Weight, color=Species)) + geom_point() + xlab('Tail (mm)') + ylab('Wei
```



2. Finite probability spaces

Now, let's move on to consider probability on finite sample spaces, which is covered in Lecture 8.

2.1 Sampling with replacement

for positive integers n and k,

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

gives the number of subsets of size k from a set of n objects.

You can compute this number straightforwardly within "R" via the choose function "choose()". For example, if we want to compute the number of different subsets of size 3 from a collection of size 8 we would compute:

```r
choose(8,3)
```

```
## [1] 56
```

Suppose we have a bag containing 10 spheres. This includes 3 red spheres and 7 blue spheres.

Let's suppose that we draw a sphere at random from the bag (all spheres have equal probability of being drawn). We record its colour and then return the sphere to the bag. This process is repeated 22 times. This is an example of sampling with replacement since the spheres are replaced after each draw.

(Q1) Write down a mathematical expression for the probability that z out of the 22 selections were red spheres (here z   {0,1,   ,22}).

Let A be the event that z out of the 22 selections were red spheres. The probability is

$$P(A) = \binom{22}{z}\left(\frac{3}{10}^{10}\right)\left(\frac{7}{10}^{22-x}\right)$$

(Q2) Next write an R function called "prob_red_spheres()" which takes z as an argument and computes the probability that   out of a total of the 22 balls selected are red.

```r
num_red_balls<-3
num_blue_balls<-7
total_draws<-22
prob_red_spheres<-function(z){
  total_balls<-num_red_balls+num_blue_balls
  log_prob<-log(choose(total_draws,z))+
    z*log(num_red_balls/total_balls)+(total_draws-z)*log(num_blue_balls/total_balls)
  return(exp(log_prob))
}

prob_red_spheres(10)
```

```
## [1] 0.05285129
```

(Q3) Generate a data frame called "prob_by_num_reds" with two columns "num_reds" and "prob". The "num_reds" column should contain numbers 1 through 22 and the "prob" column should give the associated probability of selecting that many reds out of a total number of 22 selections.

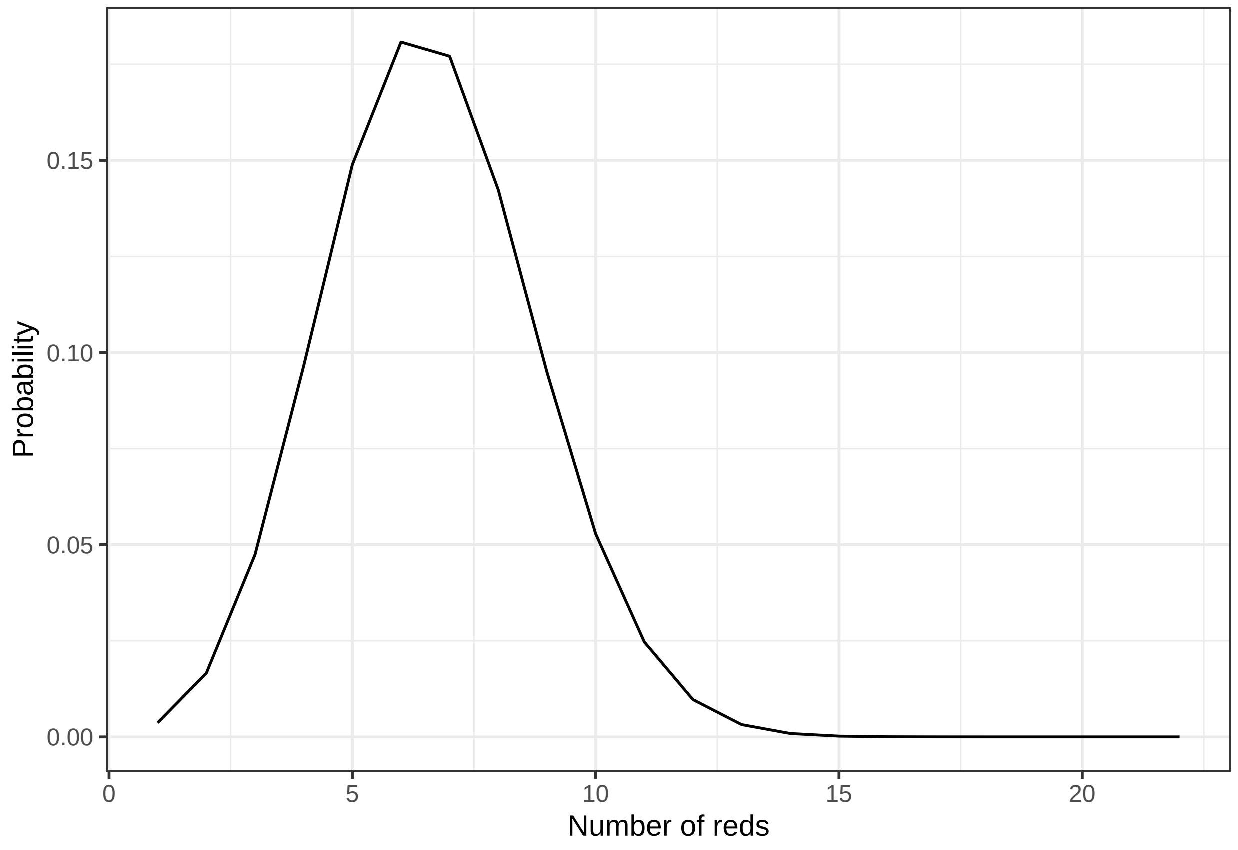
```r
prob_by_num_reds <- data.frame(num_reds=seq(22)) %>%
  mutate(prob=prob_red_spheres(num_reds))

prob_by_num_reds %>% head(3)
```

```
##   num_reds        prob
## 1        1 0.003686403
## 2        2 0.016588812
## 3        3 0.047396606
```

(Q4) Now use the "geom_line()" function within the "ggplot2" library, in conjunction with your data frame to display a plot of the probability as a function of the number of reds.

Your plot should look as follows:

```
ggplot(prob_by_num_reds, aes(x=num_reds, y=prob)) + geom_line() + xlab('Number of reds') + ylab('Probabi
```



(Q5) Next we shall explore the "sample()" function within R. Let's suppose we want to simulate a random experiment in which we sample with replacement from a collection of 10 objects, and repeat this process 22 times.

```
sample(10,22, replace = TRUE)
```

```
##  [1] 10  3  5  7  7  2  3  5  6  8  2  9  3  4  7  9  1 10  2  8  6  2
```

Try this out for yourself. The output should be a vector of length 22 consisting entirely of numbers between 1 and 10. Since this is sampling with replacements and the number of samples exceeds the number of elements, there will be repetitions. Try rerunning the function. You probably get a different sample.This is to be expected, and even desirable, since the function simulates a random sample. However, the fact that we get a different answer every time we run the code is problematic from the perspective of reproducibility.

To avoid this process we can set a random seed via the function set.seed(). By doing so we should get the same output every time. Try the following out for yourself:

```r
## case 1: Setting the random seed just once
set.seed(0)

for(i in 1:5){
  print(sample(100,5, replace=FALSE))
  # result may well differ every time
}
```

```
## [1] 14 68 39  1 34
## [1] 87 43 14 82 59
## [1] 51 97 85 21 54
## [1] 74  7 73 79 85
## [1]  37  89 100  34  99
```

```r
## case 2: Resetting the random seed every time
set.seed(1)
print(sample(100,5,replace=FALSE))
```

```
## [1] 68 39  1 34 87
```

```r
set.seed(1)
print(sample(100,5,replace=FALSE))
```

```
## [1] 68 39  1 34 87
```

```r
set.seed(1)
print(sample(100,5,replace=FALSE))
```

```
## [1] 68 39  1 34 87
```

```r
# The result should not change

## case 3: reproducing case 1 if we set a random seed at the beginning.
set.seed(0)
for(i in 1:5){
  print(sample(100,5,replace=FALSE))
}
```

```
## [1] 14 68 39  1 34
## [1] 87 43 14 82 59
## [1] 51 97 85 21 54
## [1] 74  7 73 79 85
## [1]  37  89 100  34  99
```

```r
# The result will be 5 samples exactly the same as in case 1 (why?).
```

Next, try to understand what the following code does

```r
itermap <- function(.x, .f) {
  result <- list()
  for (item in .x) {
    result <- c(result, list(.f(item)))
  }
  return(result)
}

itermap( c(1,2,3), function(x){ return(c(x,x^2)) } )
```

```
## [[1]]
## [1] 1 1
##
## [[2]]
## [1] 2 4
##
## [[3]]
## [1] 3 9
```

```r
## [[1]]
## [1] 1 1
##
## [[2]]
## [1] 2 4
##
## [[3]]
## [1] 3 9
itermap_dbl <- function(.x, .f) {
  result <- numeric(length(.x))
  for (i in 1:length(.x)) {
    result[i] <- .f(.x[[i]])
  }
  return(result)
}

itermap_dbl( c(1,2,3), function(x){ return(x^3) } )
```

```
## [1]  1  8 27
```

We shall now use the “sample()” to construct a simulation study to explore the probability of selecting z red balls from a bag of size 10, with 3 red and 7 blue balls, when sampling 22 balls with replacement. First set a random seed. Then create a data frame called “sampling_with_replacement_simulation” consisting of two columns. The first is called “trial” and contains numbers 1 through 1000. The second is called “sample_balls” and corresponds to random samples of size 22 from a bag of size 10, with replacement. We can do this as follows:

```r
num_trials<-1000 # set the number of trials
set.seed(0) # set the random seed
sampling_with_replacement_simulation<-data.frame(trial=1:num_trials) %>%
mutate(sample_balls = itermap(.x=trial, function(x){sample(10,22,
replace = TRUE)}))
# generate collection of num_trials simulations
```

```
sampling_with_replacement_simulation <-
sampling_with_replacement_simulation %>%
mutate(num_reds = itermap_dbl( .x=sample_balls, function(.x)
sum(.x<=3) ) )
```

(Q6) Next we shall add a new column called "predicted_prob" to our existing data frame "prob_by_num_reds" which gives the number of times (that we observed the corresponding number of reds within our simulation) divided by the total number of observations. This aims to estimate the probability of the event that out of a total of the 22 balls selected are red (for = 1,2, ,22). We can do this as follows:

```
num_reds_in_simulation<-sampling_with_replacement_simulation %>%
pull(num_reds)
# we extract a vector corresponding to the number of reds in each trial
prob_by_num_reds<-prob_by_num_reds %>%
mutate(predicted_prob=itermap_dbl(.x=num_reds, function(.x)
sum(num_reds_in_simulation==.x))/num_trials)
# add a column which gives the number of trials with a given number of reds
```

2.2 Sampling without replacement

This is a more challenging question, but the ideas from the last question are useful here.

Let's suppose we have a large bag containing 100 spheres. There are 50 red spheres, 30 blue spheres and 20 green spheres. Suppose that we sample 10 spheres from the bag without replacement.

You may want to use the following steps:

1. First set a random seed;

2. Next set a number of trials (e.g., 10 or 1000. Try this initially with a small number of simulations. Increase your number of simulations to about a relatively large number to get a more accurate answer, once everything seems to be working well), and a sample size (10);

3. Now use a combination of the functions "sample()", "mutate()" and "itermap()" to generate your samples. Here you are creating a sample of size 10 from a collection of 100 balls - the sampling is done without replacement;

4. Now compute the number of "reds", "greens" and "blues" in your sample using the "itermap_dbl()" and "mutate()" functions.

5. Compute the minimum of the three counts using the "pmin()" function. When this minimum is zero, then one of the three colours is missing. It is recommended that you look up the difference between "pmin()" and "min()" here;

6. Compute the proportion of rows for which the minimum number of the three counts is zero.

Answer

```
# Step 1
set.seed(0) # set the random seed
# Step 2
num_trials <- 1000 # set the number of trials
sample_size <- 10
# Step 3
sampling_without_replacement_simulation <-
data.frame(trial=1:num_trials) %>%
  mutate(sample_balls=itermap(.x=trial,function(.x)
sample(100,sample_size,replace = FALSE)))
# generate collection of num_trials simulations
# Step 4
sampling_without_replacement_simulation <-
sampling_without_replacement_simulation %>%
  mutate(num_reds=itermap_dbl(.x=sample_balls, function(.x) sum(
```

```r
(.x<=50) ) ),
         num_blues=itermap_dbl(.x=sample_balls, function(.x) sum(
(.x>50)*(.x<=80) ) ),
         num_greens=itermap_dbl(.x=sample_balls, function(.x) sum(
(.x>80) ) ),
         )
# Step 5
sampling_without_replacement_simulation <-
sampling_without_replacement_simulation %>%
  mutate(mun_minimum=pmin(num_reds, num_blues, num_greens))
# Step 6
proportion_zero =
mean(sampling_without_replacement_simulation$mun_minimum==0)
print(proportion_zero)
```

```
## [1] 0.124
```

(Q2) (*Optional) This question is optional and may be omitted if you are short on time.

Let's derive a mathematical expression for the probability that we considered in (Q1): You can try and use "combinations" with
to compute the probability directly. First aim to compute the number of subsets of size 10 from 100 which either entirely miss out one of the subsets Red = {1, ,50}, Blues = {51, ,80}, Greens = {81, ,100}. Then compute the number of subsets of size 10 from 100 which miss out two of the subsets Red, Blues, Green. Be carefulnot to double count some of these subsets! Once you have computed all such subsets, combine them with the formula for the total number of subsets of size 10 from a set of 100, to compute the probability of missing a colour.

Answer:

Let A_1, A_2, and A_3 be the events at which the sample does not contain any red balls, blue balls, or green balls, respectively. So

$$|A_1| = \frac{50}{10}, \quad |A_2| = \frac{70}{10}, \quad |A_3| = \frac{80}{10}$$

Let $A_{1,2}$, $A_{1,3}$, and $A_{2,3}$ be the events that the sample does not contain any red balls or blue balls, red balls or green balls, and blue balls or green balls, respectively. So

$$|A_{1,2}| = \frac{20}{10}, \quad |A_{1,3}| = \frac{30}{10}, \quad |A_{2,3}| = \frac{50}{10}$$

Let A be the event that one or more colors is missing from the sample. Then:

$$|A| = |A_1| + |A_2| + |A_3| - |A_{1,2}| - |A_{1,3}| - |A_{2,3}|$$

Now, substituting the values:

$$|A| = \frac{50}{10} + \frac{70}{10} + \frac{80}{10} - \frac{20}{10} - \frac{30}{10} - \frac{50}{10}$$

Simplifying:

$$|A| = 5 + 7 + 8 - 2 - 3 - 5 = 10$$

Once you have the mathematical expression for the probability, you can check how close the probability computed in (Q1) to the theoretical values.

Answer:

```r
num1 <- choose(50,10) + choose(70,10) + choose(80,10) -
choose(20,10) - choose(30,10) - choose(50,10)
num2 <- choose(100,10)
probs_A <- num1/num2
print(probs_A)
```

```
## [1] 0.1180318
```