Assignment 3

EMATM0061: Statistical Computing and Empirical Methods, TB1, 2024

Introduction

Create an R Markdown for the assignment

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:

```
library(tidyverse)
```

2. Load the Stat2Data package and then the dataset Hawks:

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

1. Visualisation

This part is mainly about visualisation using ggplot2. It covers mainly Lecture 7.

We are going to use the "Hawks" dataset that was used last week.

(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))
```

Use the **dim()** function to check how many rows and how many columns in hawksSmall. Then use the **head()** function to display the first 5 rows of the hawksSmal.

Answer

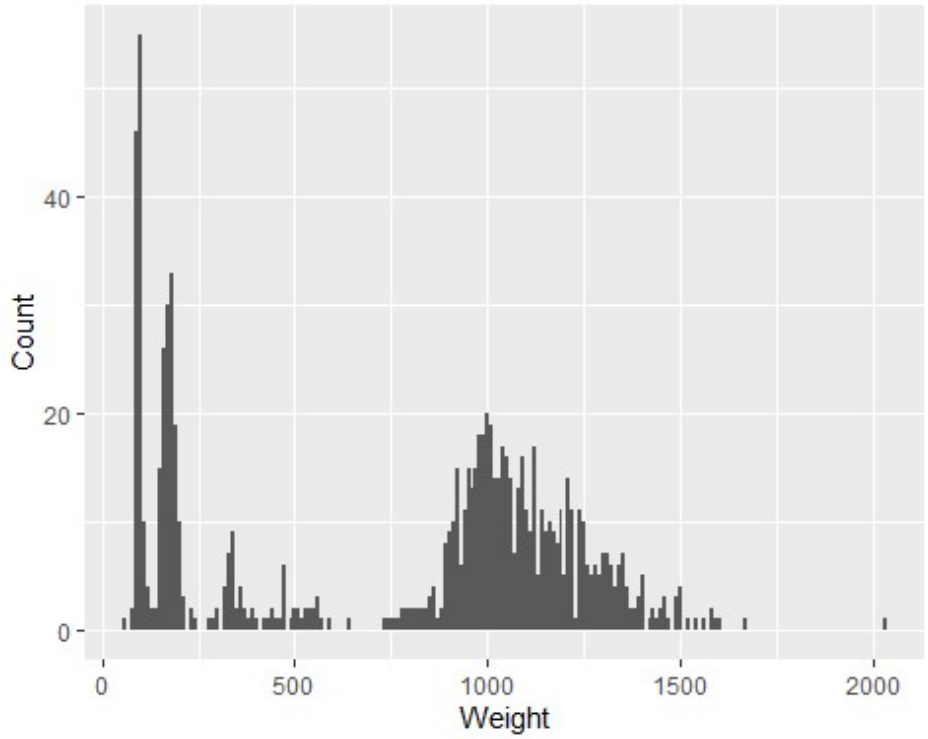
```
  head(hawksSmall, 5)
```

```
##   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
```

(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.

Answer

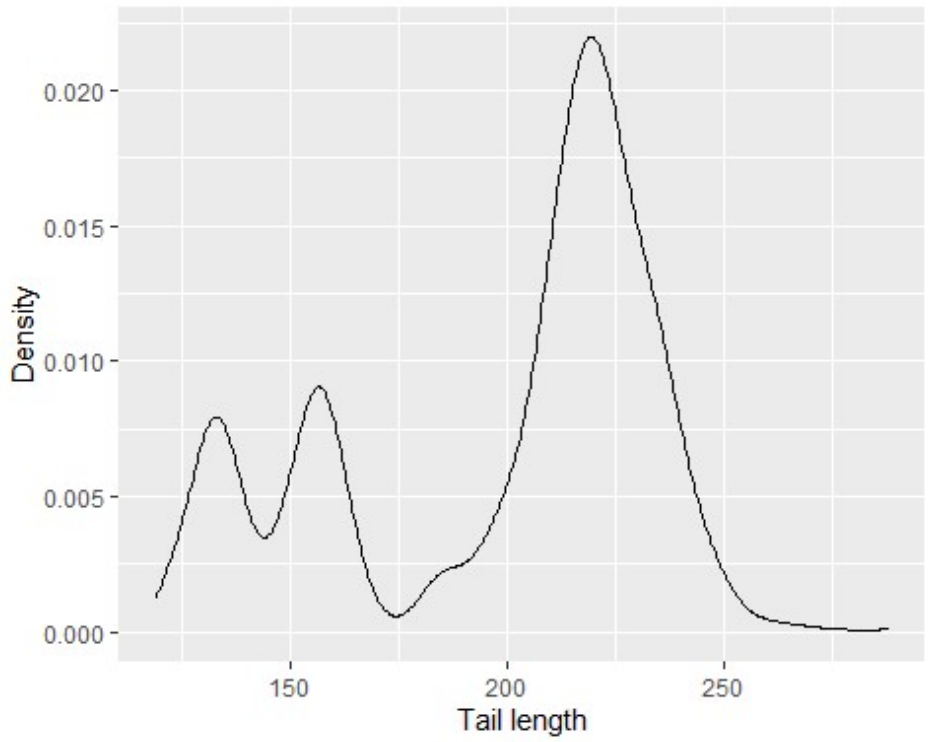
```
ggplot(data=hawksSmall,aes(x=Weight))+geom_histogram(binwidth=10)+
xlab("Weight")+ylab("Count")
```



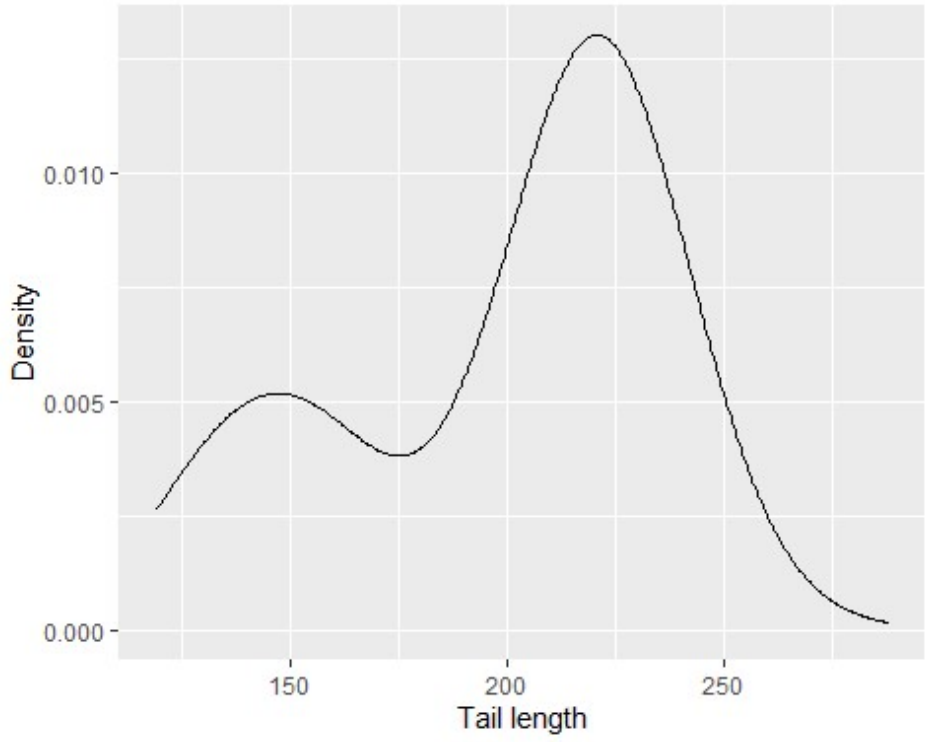
(Q3) Similar to (Q2), use the “geom_density()” function to create two density plots of the tail lengths of Hawks within the hawksSmall data frame, with parameters “adjust=0.5” and “adjust=2”, respectively.

Answer

```
ggplot(data=hawksSmall,aes(x=Tail))+geom_density(adjust=0.5)+
xlab("Tail length")+ylab("Density")
```




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



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.

Answer

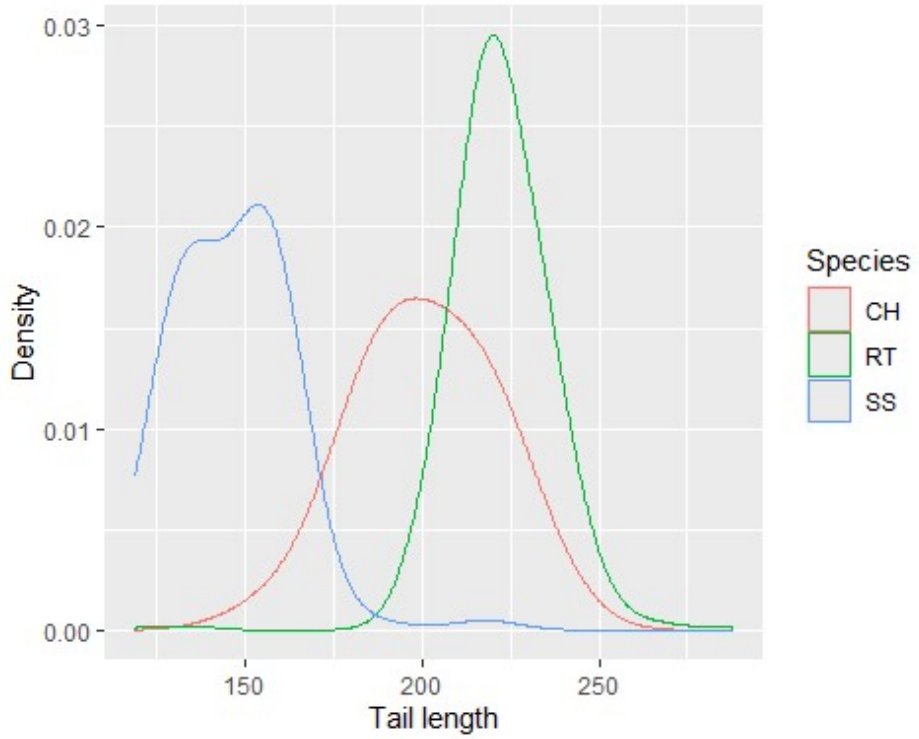
The second plot has a curve that is a smoother version of the first plot. This is because the adjust argument in the geom_density() function adjusts the bandwidth of the density plot. A larger "adjust" corresponds to a larger bandwidth, hence a smoother function in the plot, whilst a smaller "adjust" create a less smooth density plot which is more sensitive to the local behaviour 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?

Answer

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

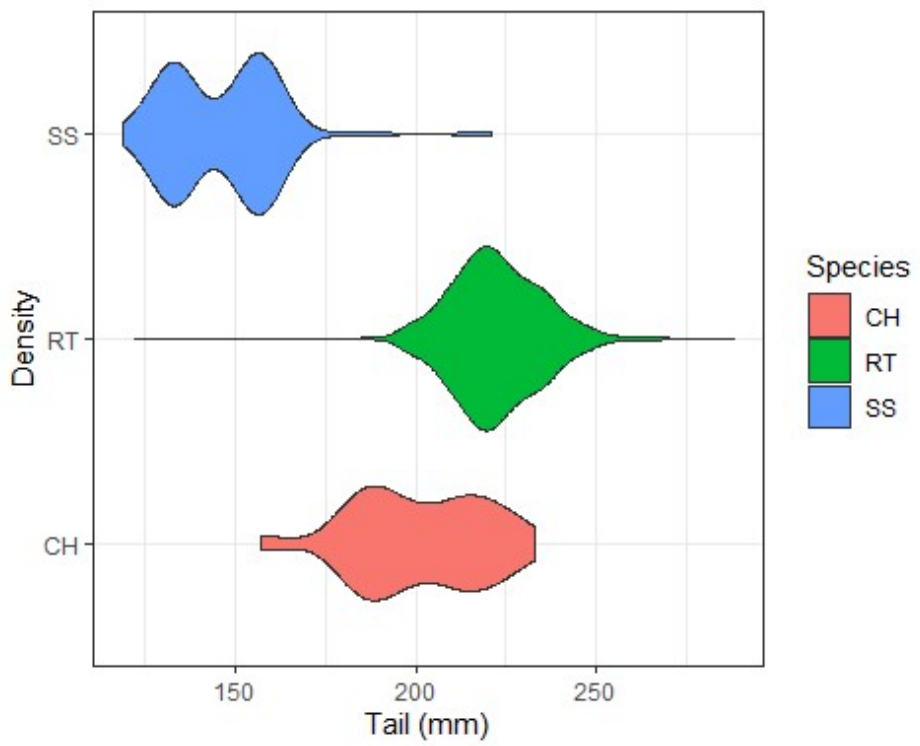


(Q5) Violin plot:

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

Answer

```
ggplot(hawksSmall, aes(x=Tail, y=Species, fill=Species)) +
geom_violin() +
  xlab('Tail (mm)') + ylab('Density') + theme_bw()
```



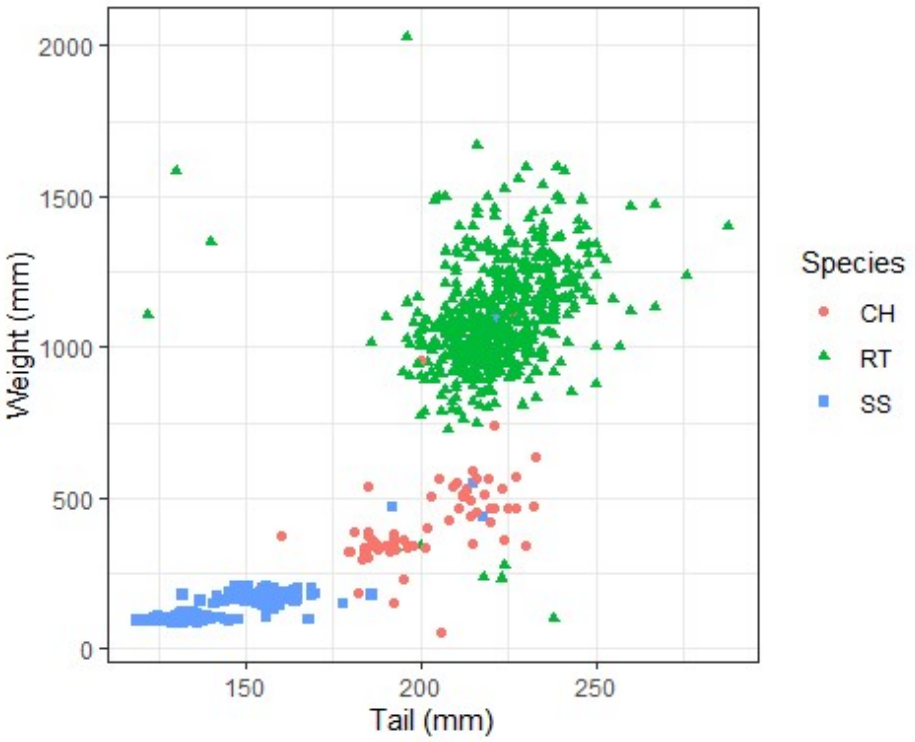
(Q6) Scatter plot

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

Answer

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



Answer

1. 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. The glyphs are the small elements (in this particular case represented by the shapes) corresponding to individual cases.
3. The visual cues include horizontal and vertical position, shape, color.

(Q7) Trend lines and facet wraps:

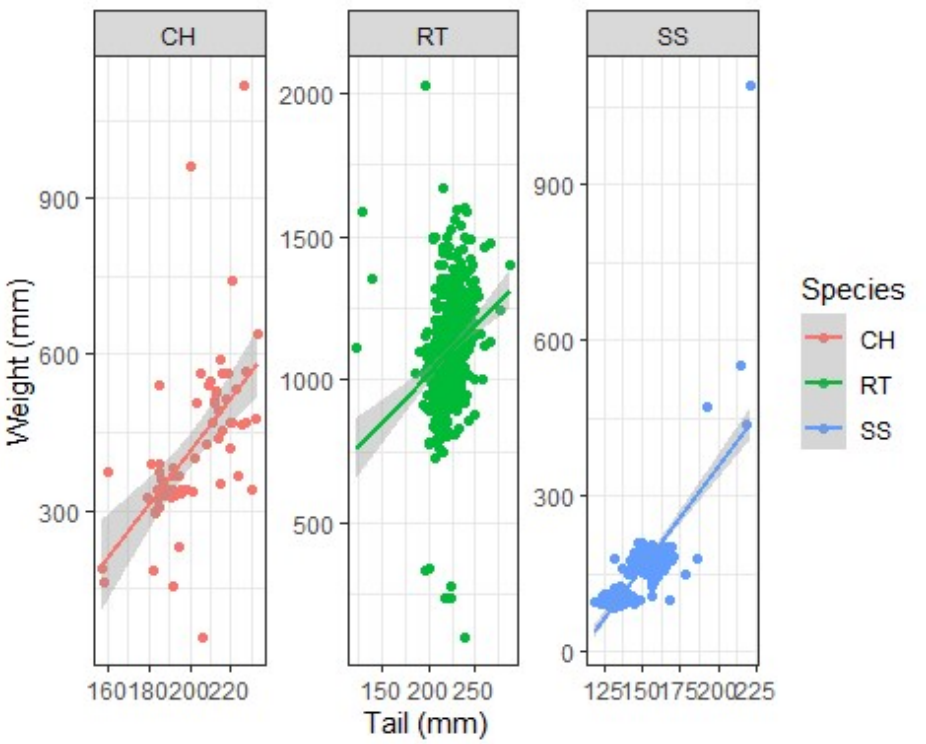
Generate the following plot using the ggplot(), geom_point(), geom_smooth() and facet_wrap() functions. Note that in the facet plot, the three panels use different scales.

1. What are the visual cues being used within this plot?
2. Based on the plot below, what can we say about the relationship between the weight of the hawks and their tail lengths?

Answer

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

```
  facet_wrap(~Species, scales='free') + geom_smooth(method='lm') +
  xlab('Tail (mm)') + ylab('Weight (mm)') + theme_bw()
```



1. The visual cues include horizontal and vertical position, shape, color and direction.
2. Based on this sample, longer tail lengths appear to be predictive of larger weights within each species.

(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.

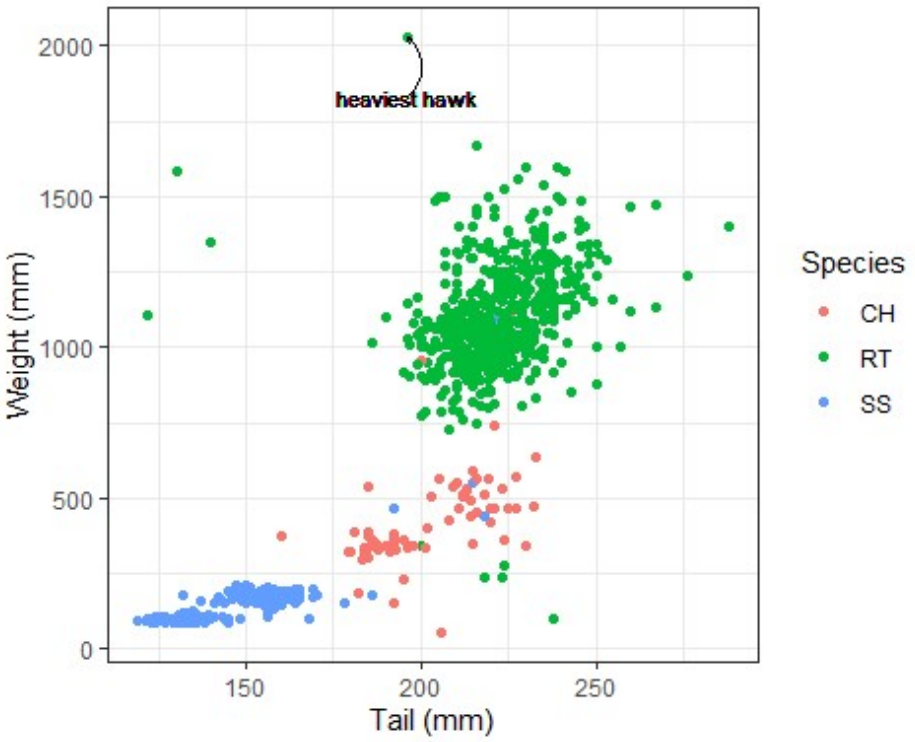
Answer

```
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('Weight (mm)') + theme_bw() +
  geom_curve(x=max_weight_tail,xend=max_weight_tail,y=max_weight-
200,yend=max_weight,
```

```
            arrow=arrow(length=unit(0.1,"cm")), color="black")+
  geom_text(x=max_weight_tail,y=max_weight-200,label='heaviest
hawk',size=3,color="black")
```



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

Recall that 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:

```
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 $z \in \{0,1,\cdots,22\}$).

You can try writing down your mathematical expression using "LaTex" code in your R markdown file, making use of the LaTex functions "binom{"{}} and "frac{"{}}. For information about writing mathematical formulae in Rmarkdown documents, find illustrative examples in "Assignment_R MarkdownMathformulasandSymbolsExamples.rmd" (under the resource list tab on blackboard course webpage) or visit: https://bookdown.org/yihui/bookdown/markdown-extensions-by-bookdown.html#equations

Answer:

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

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

The LaTex code is as follows:

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

Answer:

```
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))
}
```

Test your function as follows:

```
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.

Answer:

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

Display the first 3 rows of your data frame:

```
prob_by_num_reds %>% head(3)
```

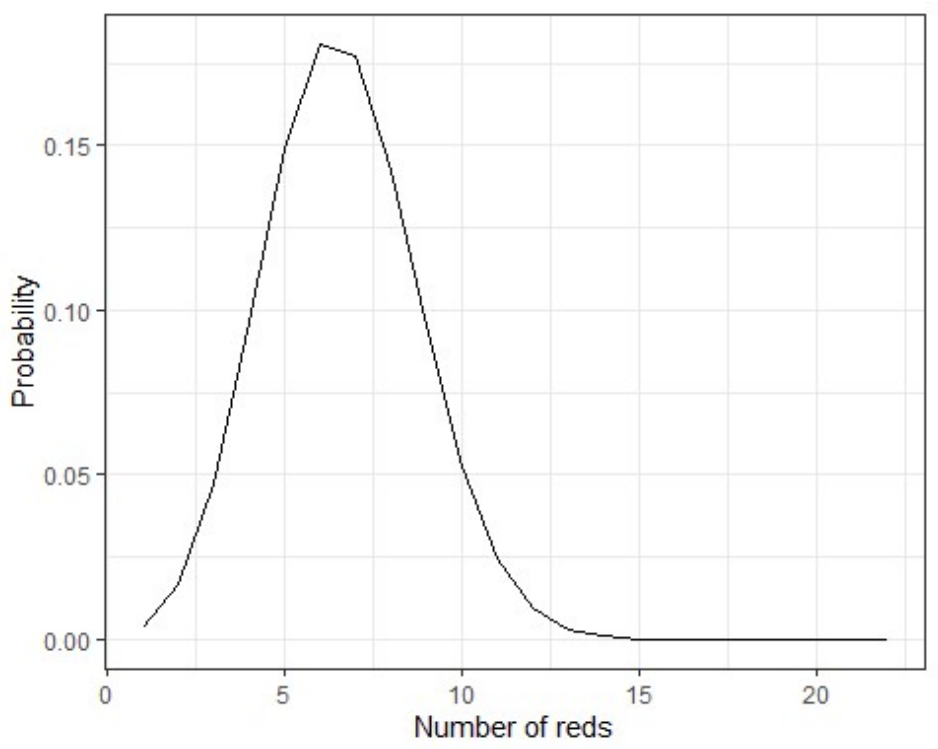
```
##   num_reds        prob
## 1        1 0.003686403
## 2        2 0.016588812
## 3        3 0.047396606
```

(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:

Answer:

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



(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.

We can do this by calling:

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

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

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:

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

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

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

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
```

The function “itermap” and “itermap_dbl” defined above are similar to the “map()” and “map_dbl” functions in R that will be introduced next week.

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:

```
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
```

Now add a new column called “num_reds” such that, for each row, “num_reds” contains an integer which gives the number of items within the sample for that row (i.e., the sample represented by the entry of the “sample_balls” column) which are less than or equal to three (assuming that the three red balls are labelled by {1,2,3}). For example, suppose that for some row of the data frame, the “sample_balls” column contains a entry which is the following list:

4 10 4 10 8 3 5 5 5 5 10 7 1 2 1 10 5 6 5 7 1 10

Then the corresponding row of the “num_reds” column should contain the number 5, since 5 of these values are less than equal to 3. You may want to use the functions “mutate()”, “itermap_dbl” and “sum()”. After performing this step, the data frame “sampling_with_replacement_simulation” should contain a new column called “num_reds”.

Answer:

```
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 z out of a total of the 22 balls selected are red (for $z = 1,2,\cdots,22$). We can do this as follows:

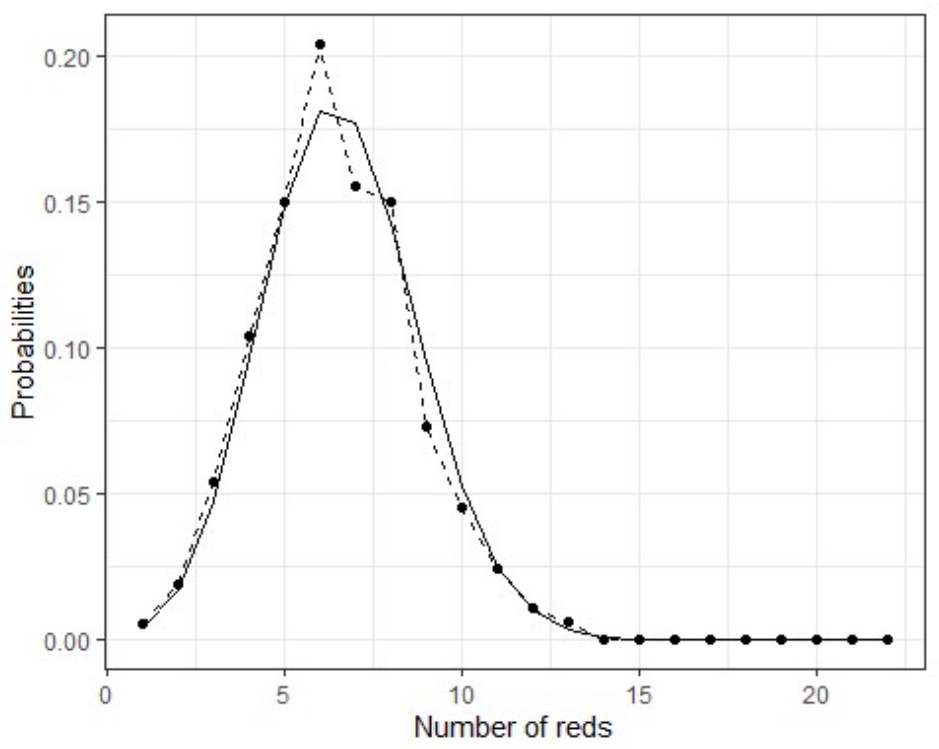
```
num_reds_in_simulation<-sampling_with_replacement_simulation %>%
  pull(num_reds)
# we extract a vector corresponding to the number of reds in each trial
prob_by_num_reds<-prob_by_num_reds %>%
  mutate(predicted_prob=itermap_dbl(.x=num_reds, function(.x)
sum(num_reds_in_simulation==.x))/num_trials)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

Here “prob_by_num_reds” was initially defined in (Q3).

(Q7) Finally, create a plot which compares the results of your simulation with your probability formula.

Your result should look something like the plot below. Of course, since this is a random simulation, your result may well look slightly different.

```
prob_by_num_reds %>%
  rename(TheoreticalProbability=prob,
EstimatedProbability=predicted_prob) %>%
  ggplot() + geom_line(aes(x=num_reds, y=TheoreticalProbability)) +
  geom_line(aes(x=num_reds, y=EstimatedProbability), linetype='dashed')
+
  geom_point(aes(x=num_reds, y=EstimatedProbability)) +
  theme_bw() + xlab("Number of reds") + ylab("Probabilities")
```



Try to make sense of each line in the above code.

Notice the close relationship between probabilities and long-run frequency behaviour. We shall explore this connection over the next few lectures.

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.

(Q1) What is the probability that one or more colours are missing from your selection? First aim to answer this question via a simulation study using ideas from the previous question.

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(
(.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 $\binom{n}{k}$ 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 $= \{1, \cdots, 50\}$, Blues $= \{51, \cdots, 80\}$, Greens $= \{81, \cdots, 100\}$. Then compute the number of subsets of size 10 from 100 which miss out two of the subsets Red, Blues, Green. Be careful not 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, blues balls or green balls, respectively. So

$$|A_1| = \binom{50}{10}, \quad |A_2| = \binom{70}{10}, \quad \text{and } |A_3| = \binom{80}{10}.$$

Let $A_{1,2}$ be the events that the sample does not contain any red balls or blue balls. And similarly, we can define $A_{1,3}$ and $A_{2,3}$. So

$$|A_{1,2}| = \binom{20}{10}, \quad |A_{1,3}| = \binom{30}{10}, \quad \text{and } |A_{2,3}| = \binom{50}{10}.$$

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

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

Finally, the probability for the event A is

$$\mathbb{P}(A) = \frac{\binom{50}{10} + \binom{70}{10} + \binom{80}{10} - \binom{20}{10} - \binom{30}{10} - \binom{50}{10}}{\binom{100}{10}}$$

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

Answer:

```
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
```