Contents

Juweria Ali

25/02/2022

1.1 Clearing the workspace and setting the working directory.

```
rm(list=ls())
```

1.2 Set working directory

```
#library(here)
#setwd(here::here())
```

1.3 Loading libraries

```
library(tidyr)
library(stringr)
library(lubridate)

library(caret)

library(Hmisc)

library(ggplot2)
library(rattle)
```

1.4 Loading & reading the datasets - Analysis and Escapes

```
Escapes <- read.csv("escapes.csv", header=T, stringsAsFactors=F)
Analysis <- read.csv("analysis.csv", header=T, stringsAsFactors=T)
```

Q1) Preparing the data sets

##Removing unnecessary columns

```
Escapes$Aquaculture.Type <- NULL # Removing Aquaculture Type, as it has no va
riation (i.e. factors with one level)

Escapes$Escape.Grid.Reference <- NULL # Removing Escape Grid Reference, as re
ference points are not useful in analysis

Escapes$Escape.ID <- NULL #Removing Escape ID,  the value is unique to all th
e instances and does not add value to the overall analysis and can be deleted

Escapes$Escape.Water.Type <- NULL #This column is an abbreviation of the colu
mn Water.type, hence not required
Escapes$Marine.Scotland.Site.ID <-NULL
Escapes$Initial.Escape.Reason <-NULL
Escapes$National.Grid.Reference <-NULL
Escapes$Species <- NULL
Escapes$Easting <-NULL
Escapes$Northing <-NULL
Escapes$Site.Address.1 <-NULL
Escapes$Site.Address.2 <-NULL
Escapes$Site.Address.3 <-NULL
Escapes$Escape.Start.Time <-NULL
Escapes$MS.Management.Area <-NULL
Escapes$Date.Registered <-NULL
Escapes$Producing.in.Last.3.Years <-NULL
```

##Cleaning up column Age

https://www.rdocumentation.org/packages/tidyr/versions/1.2.0/topics/separate
https://stackoverflow.com/questions/53738857/dplyr-mutate-unlist-issue

##Replacing 'unknown' with NA

```
Escapes$Age[Escapes$Age == "unknown"] <- NA
```

##Replacing the instances with a string value to NA's

```
Escapes$Age[Escapes$Age == "unknown"]<-NA
Escapes$Age[Escapes$Age == "post smolt"]<-NA
Escapes$Age[Escapes$Age == "parr/presmt"]<-NA
Escapes$Age[Escapes$Age == "not report"]<-NA
Escapes$Age[Escapes$Age == "parr"]<-NA
Escapes$Age[Escapes$Age == "parr/presm"]<-NA
```

##Extracting the numeric values from the column age

```
Escapes$Age01<- str_extract_all(Escapes$Age, "[0-9]+")
head(Escapes$Age01)
```

```
## [[1]]
## [1] "28"
##
## [[2]]
## [1] "18"
##
## [[3]]
## [1] "30"
##
## [[4]]
## [1] "9"
##
## [[5]]
## [1] "11"
##
## [[6]]
## [1] NA
```

##Stripping value of instances from the new column into three columns namely New_Age

```
Escapes1<- Escapes%>% separate(Age01, into = c("New_Age"), sep = "_", remove
= TRUE)
```

##Removing columns Age,Age02 and Age03 as they are no longer useful

```
Escapes2<-subset(Escapes1,select=-c(Age))
```

##There are instances in the column that are a list. The function below identifies and replaces/imputes them with NA's

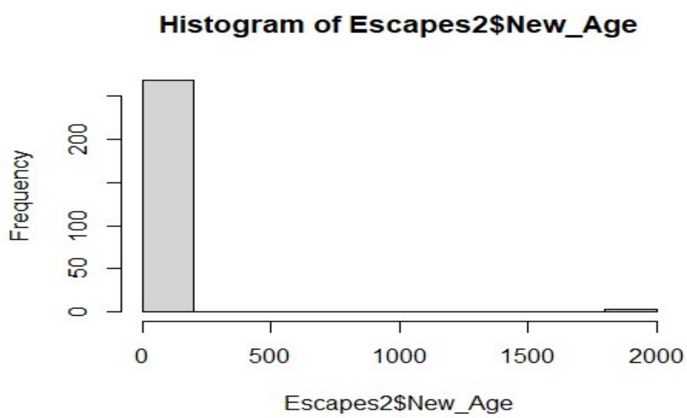
https://r-lang.com/how-to-convert-list-to-numeric-value-in-r/#:~:text=To%20convert%20R%20List%20to%20Numeric%20value%2C%20use%20the%20combination,contains%20all%20the%20atomic%20components

```
Escapes2$New_Age<-as.numeric(unlist(Escapes2$New_Age))

## Warning: NAs introduced by coercion
```

##Histogram to check for any outliers

```
hist(Escapes2$New_Age)
```



##Outliers noticed, deal with them

```
Escapes2$New_Age[Escapes2$New_Age == 1999] <-NA
Escapes2$New_Age[Escapes2$New_Age == 2000] <-NA
```

##Imputing NA's in the column New_Age with median

```
Escapes2$New_Age <- impute((Escapes2$New_Age), median)
```

##Cleaning the column Average Weight column

##Replacing the instances with string values i.e. unknown and post smolt to NA's

```
Escapes2$Average.Weight[Escapes2$Average.Weight == "unknown"]<-NA
Escapes2$Average.Weight[Escapes2$Average.Weight == "post smolt"]<-NA
```

##We only need the numeric values from the Average.Weight column. In order to do this we separate the column into kg and g column,then, filter the columns as shown below.

```
Escapes3 <- Escapes2%>% filter(grepl('kg|kilos', Average.Weight))  %>% filter
(!grepl('-', Average.Weight))

Escapes4 <- Escapes2%>% filter(!grepl('k', Average.Weight)) %>% filter(grepl(
'g', Average.Weight))  %>% filter(!grepl('-', Average.Weight))
```

##As there are some instances that do not have a kg or g,now we merge the two columns created above and get the difference of what was left after separating the kg and g.

```
Escapes_join<-merge(Escapes3,Escapes4,all=TRUE)
difference <- setdiff(Escapes2,Escapes_join)
```

##Substituting the grams and kilograms with blank spaces and converting the grams column to kilogram by dividing with 1000.

```
Escapes3<-Escapes3 %>%  mutate(Average.Weight=as.double(gsub('kg','',as.chara
cter(Escapes3$Average.Weight))))
```

```
## Warning in mask$eval_all_mutate(quo): NAs introduced by coercion
```

```
Escapes4<- Escapes4%>%  mutate(Average.Weight=as.double(gsub('g','',as.charac
ter(Escapes4$Average.Weight))))
```

```
## Warning in mask$eval_all_mutate(quo): NAs introduced by coercion
```

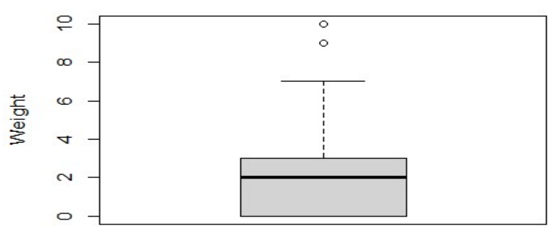
```
Escapes4$Average.Weight<- Escapes4$Average.Weight / 1000
difference$Average.Weight <- NA
```

##Merging all the three above data frames back to one data frame Escapes6

```
Escapes5 <- union(Escapes3,Escapes4)
Escapes6 <- union(Escapes5,difference)
```

Imputing NA's in the column Average Weight with median

```
Escapes6$Average.Weight <- impute((Escapes6$Average.Weight), median)
Escapes6$Average.Weight<-as.integer(Escapes6$Average.Weight)
boxplot(Escapes6$Average.Weight, ylab = "Weight")
```



Preparing the datasets fro merge

##Extracting the numeric values from the colummns Initial Number Escaped,Final Number Escaped,Final Number Recovered

```
Escapes6$Initial.Number.Escaped <- as.integer(str_extract(Escapes6$Initial.Nu
mber.Escaped, "[0-9]+"))
Escapes6$Final.Number.Escaped <- as.integer(str_extract(Escapes6$Final.Number
.Escaped, "[0-9]+"))
Escapes6$Final.Number.Recovered <- as.integer(str_extract(Escapes6$Final.Numb
er.Recovered, "[0-9]+"))
```

##Creating a new dataframe with only the columns Initial.Number Escaped,Final.Number.Escaped, Final.Number.Recovered,New_Age,Average.Weight to apply bag impute

```
Escapes7<-Escapes6 %>% select(Initial.Number.Escaped, Final.Number.Escaped, F
inal.Number.Recovered,New_Age,Average.Weight)
```

##Bag Impute applied to replace NA's

```
bagImp <- preProcess(Escapes7, method = c("bagImpute"))
bagimputedEscapes <- predict(bagImp, Escapes7)
```

##Creating a new dataframe by deleting the columns Initial.Number.Escaped,Final.Number.Escaped,Final.Number.Recovered,New_Age,Average.Weight in order to merge it with the dataframe with bag imputed values

```
EscapesnoCol<-subset(Escapes6,select=-c(Initial.Number.Escaped, Final.Number.
Escaped, Final.Number.Recovered,New_Age,Average.Weight))
```

##Combining the dataframe with EscapesnoCol and bagimputedEscapes, assigning that to a new dataframe EscapesCol, then assigning that new dataframe to EscapesFinal

```
EscapesFinal<-bind_cols(EscapesnoCol,bagimputedEscapes)
```

#Removing unnecessary columns

```
EscapesFinal$Initial.Number.Escaped <-NULL
EscapesFinal$Final.Number.Recovered <- NULL
EscapesFinal$Initial.Date.of.Escape <- NULL
EscapesFinal$Final.Date.of.Escape <- NULL
EscapesFinal$Escape.End.Time <- NULL
```

##Formatting the column Escape start date

###the dates are in character format, here we are parsing them in a date format as 'dmy' https://cran.r-project.org/web/packages/lubridate/lubridate.pdf

```
EscapesFinal$Escape.Start.Date <-
  dmy(EscapesFinal$Escape.Start.Date)
```

##Extracting month and year as new columns

```
EscapesFinal$StartingMonth <- as.character(as.integer(format(EscapesFinal$Esc
ape.Start.Date, "%m")))
EscapesFinal$StartingYear<- as.character(as.integer(format(EscapesFinal$Escap
e.Start.Date, "%Y")))
```

##Changing site name in both datasets to lower case in order make them uniform

```
EscapesFinal$Site.Name <- tolower(EscapesFinal$Site.Name)
Analysis$Site.Name <- tolower(Analysis$Site.Name)
```

##Converting the month and year columns of the Analysis dataset to character type so to match the EscapesFinal dataset column type

```
Analysis$month <- as.character(Analysis$month)
Analysis$year<- as.character(Analysis$year)
```

##Creating a new column Grp in the Analysis dataframe by combining month,year and Site.Name,to then compare and see if there are any duplicates and then everything that is not a duplicate stays in Analysis dataframe.

```
Analysis$Grp <- paste(Analysis$month,Analysis$year,Analysis$Site.Name)
Analysis<-Analysis[!duplicated(Analysis$Grp), ]
```

##Deleting the column Grp as it is no longer needed

```
AnalysisFinal <- subset(Analysis, select = -c(Grp))
```

##Removing other columns in the Analysis dataset that are not useful

```
AnalysisFinal <- subset(AnalysisFinal, select = -c(c2,c3,c4,c5,c6,c7))
```

Q2) Integrate the 2 datasets together into a merged dataset

```
escapesPlus <- EscapesFinal %>% left_join(AnalysisFinal, AnalysisFinal, by=c(
'Site.Name'='Site.Name', 'StartingYear'='year','StartingMonth'='month'))

write.csv(escapesPlus, "escapesPlus.csv")
```

##Summary of the new merged dataset

```
summary(escapesPlus)

##  Operator.at.Time.of.Escape Escape.Start.Date    Escaped.Species
##  Length:356                 Min.   :1995-11-01   Length:356
##  Class :character           1st Qu.:2005-01-11   Class :character
##  Mode  :character           Median :2008-12-10   Mode  :character
##                             Mean   :2009-10-01
##                             3rd Qu.:2015-01-30
##                             Max.   :2020-12-04
##     Stage            Final.Escape.Reason  Site.Name          Local.Authority
##  Length:356         Length:356          Length:356         Length:356
##  Class :character   Class :character    Class :character   Class :characte
r
##  Mode  :character   Mode  :character    Mode  :character   Mode  :characte
r
##
##
##
##  Site.Post.Code     Site.Contact.Number  Water.Type         Health.Surveill
ance
##  Length:356         Length:356          Length:356         Length:356
##  Class :character   Class :character    Class :character   Class :characte
r
##  Mode  :character   Mode  :character    Mode  :character   Mode  :characte
r
##
##
```

```
##
##     Region             Operator          Final.Number.Escaped    New_Age
##  Length:356         Length:356         Min.   :     0.0     Min.   : 1.00
##  Class :character   Class :character   1st Qu.:     0.0     1st Qu.:12.00
##  Mode  :character   Mode  :character   Median :    28.5     Median :13.00
##                                        Mean   :  3958.1     Mean   :14.28
##                                        3rd Qu.:  1263.8     3rd Qu.:17.00
##                                        Max.   :258000.0     Max.   :50.00
##  Average.Weight   StartingMonth      StartingYear
##  Min.   : 0.000   Length:356         Length:356
##  1st Qu.: 0.000   Class :character   Class :character
##  Median : 2.000   Mode  :character   Mode  :character
##  Mean   : 1.837
##  3rd Qu.: 3.000
##  Max.   :10.000
```

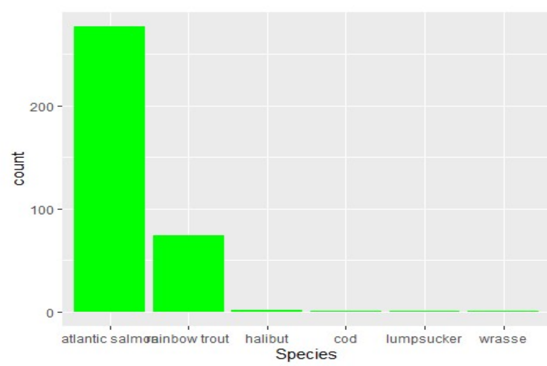
Q3) Exploratory data analysis of the dataset

##Univariate Analysis

##Checking class distribution for Escaped.Sepcies, Final.Escape.Reason in order to identify any interesting insights.

##Plotting a bar chart on the Escaped.Species to check the class distribution.

```
plot1 <- ggplot(escapesPlus, aes(x=reorder(Escaped.Species, Escaped.Species,
function(x)-length(x)))) +
geom_bar(fill='Green') +  labs(x='Species')
plot1
```



From the above bar plot we see that Atlantic salmon and the rainbow trout are the two species that are predominant. The plot shows about 245 instances of class 'atlantic salmon', 65 instances of class 'rainbow trout' and the other classes are very less comparitvely.
https://www.r-bloggers.com/2021/08/how-to-plot-categorical-data-in-r-quick-guide/

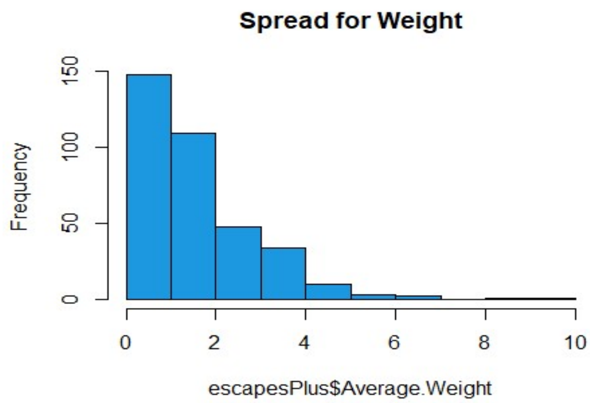
##Average Weight is a continuous variable

###Spread

```
summary(escapesPlus$Average.Weight)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##   0.000   0.000   2.000   1.837   3.000  10.000

hist(escapesPlus$Average.Weight, main = "Spread for Weight",col = "#1b98e0")
```



Here we have Average Weight on the x-axis and frequency on the y-axis. Looking at the analysis and graph we can say that the minimum weight is 0 kg and maximum is 9 kg . We can also see the mean i.e. the average weight is 1.78 kg. Also, we observe from the histogram that the most frequently escaping fish are of an average weight of 0kg - 4kg.

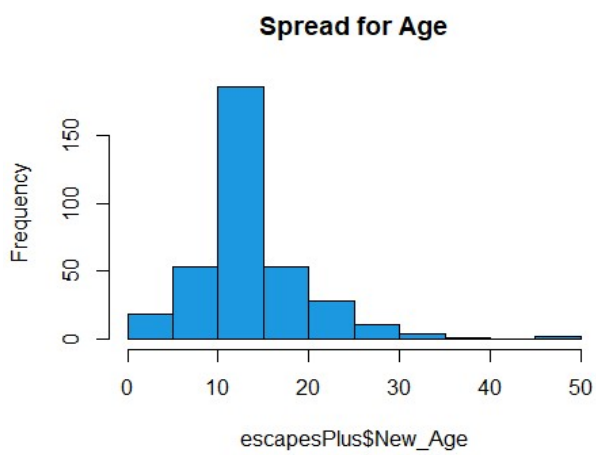
##New_Age is a continuous variable

###Spread

```
summary(escapesPlus$New_Age)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##    1.00   12.00   13.00   14.28   17.00   50.00

hist(escapesPlus$New_Age, main = "Spread for Age",col = "#1b98e0")
```

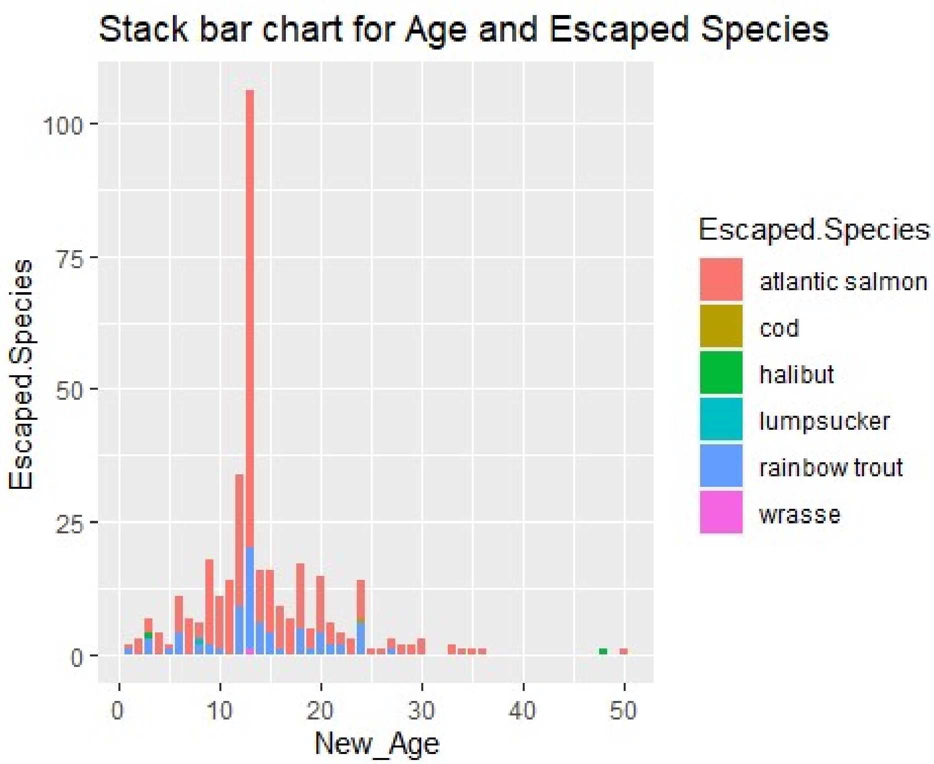


Here we have New_Age on the x-axis and frequency on the y-axis. Looking at the analysis and graph we can say that the minimum age of the fish is 1 month and maximum age of the escaped fish 50 months We can also see the mean i.e. the average age is 14 months.From the above histogram we see that the age of fish that escape more frequently were between the age of 10 -15 months.

#Bivariate analysis to see impact of New_Age on Escaped Species

##Stacked bar chart

```
ggplot(escapesPlus,
       aes(x = New_Age,
           fill = Escaped.Species)) +
  geom_bar(position = "stack")+
labs(title = "Stack bar chart for Age and Escaped Species",x="New_Age",y="Esc
aped.Species")
```

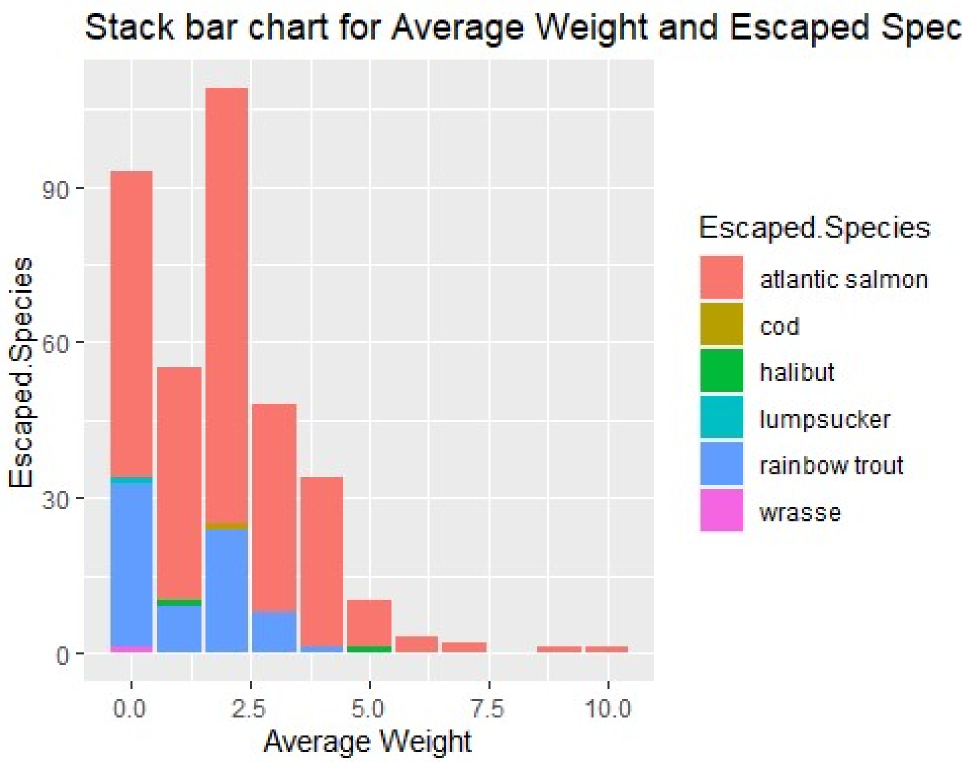


From the chart above we can say that atlantic salmon is the species that had the highest number of escapes Also, we observe that 13 months being the age when most of the fishes escaped.

#Bivariate analysis to see impact of Average weight on Escaped Species ##stacked bar chart

```
ggplot(escapesPlus,
       aes(x = Average.Weight,
           fill = Escaped.Species)) +
  geom_bar(position = "stack")+
```

```
labs(title = "Stack bar chart for Average Weight and Escaped Species",x="Aver
age Weight",y="Escaped.Species")
```



From the chart we can see that atlantic salmon species are the heaviest.There are instances which reflect that there are other species like cod, halibut and lumpsucker which are also heavier. We may also say that fishes with the highest number of escapes weighed about 1.25 to 2.5 kg regardless of which species they belonged to.

Checking for any correlation between the age of the fish and their weight

```
escapesPlus$New_Age <- as.integer(as.character(escapesPlus$New_Age))
byEscapeSpecies <- group_by(escapesPlus, Escaped.Species)
groupedDetails <- summarise(byEscapeSpecies,
                    count = n(),
                    averageAge = mean(New_Age, na.rm=T),
                    medianAge = median(New_Age, na.rm=T),
                    oldest = max(New_Age, na.rm=T),
                )
groupedDetails

## # A tibble: 6 x 5
##   Escaped.Species count averageAge medianAge oldest
##   <chr>           <int>      <dbl>     <dbl>  <int>
## 1 atlantic salmon   277       14.2      13       50
## 2 cod                 1       24        24       24
## 3 halibut             2       25.5      25.5     48
## 4 lumpsucker          1        8         8        8
```

```
## 5 rainbow trout      74       14.1       13       27
## 6 wrasse              1       13         13       13
```

```
escapesPlus$Average.Weight <- as.integer(as.character(escapesPlus$Average.Wei
ght))
byEscapeSpecies <- group_by(escapesPlus, Escaped.Species)
groupedDetails <- summarise(byEscapeSpecies,
                    count = n(),
                    averageWeight = mean(Average.Weight, na.rm=T),
                    medianWeight = median(Average.Weight, na.rm=T),
                    heavy = max(Average.Weight, na.rm=T),

                    )
groupedDetails
```

```
## # A tibble: 6 x 5
##   Escaped.Species count averageWeight medianWeight heavy
##   <chr>           <int>         <dbl>        <dbl> <int>
## 1 atlantic salmon   277          2.03            2    10
## 2 cod                 1          2               2     2
## 3 halibut             2          3               3     5
## 4 lumpsucker          1          0               0     0
## 5 rainbow trout      74          1.15            1     4
## 6 wrasse              1          0               0     0
```

From the above tables we can see the atlantic salmon and cod are the oldest and the heaviest species. This reflects that the weight of the fish is most probably correlated to the age.

Q4) Preparing the dataset for learning

##We want to predict the feature Average.Weight, using the rest of the data in the NewEscapesPlus dataset

Selecting columns that are useful for learning

```
escapesPlus <- subset(escapesPlus, select = c(Escape.Start.Date,Average.Weigh
t,New_Age,Escaped.Species,Stage,Final.Escape.Reason))
```

##Selecting instnaces with only atlantic salmon and rainbow trout as species

```
NewEscapesPlus <- escapesPlus[ which( escapesPlus$Escaped.Species=="atlantic
salmon"| escapesPlus$Escaped.Species == "rainbow trout") , ]
```

Atlantic salmon and rainbow trout are the predominant species and there are not enough instances to put forward for the learning task, hence we do not consider them.

##Removing missing values if any

```
NewEscapesPlus = NewEscapesPlus[complete.cases(NewEscapesPlus), ]
```

##Applying learning tasks

##Categorical target with rpart, rf and xgboost

###Setting train controls (repeated cv)

```
trControl2 <- trainControl(method = "repeatedcv", number=10, repeats=1)
```

The train control uses repeated 10-fold cross validation.The repeats are set to 1 to keep the model building time low.The number is the number of folds and repeats is the number of repetitions.

Decision tree model

```
set.seed(1234)
rpart.model <- train(Escaped.Species~.,
    data = NewEscapesPlus,
    method = "rpart",
    metric = "Accuracy",
    trControl = trControl2)

print(rpart.model)

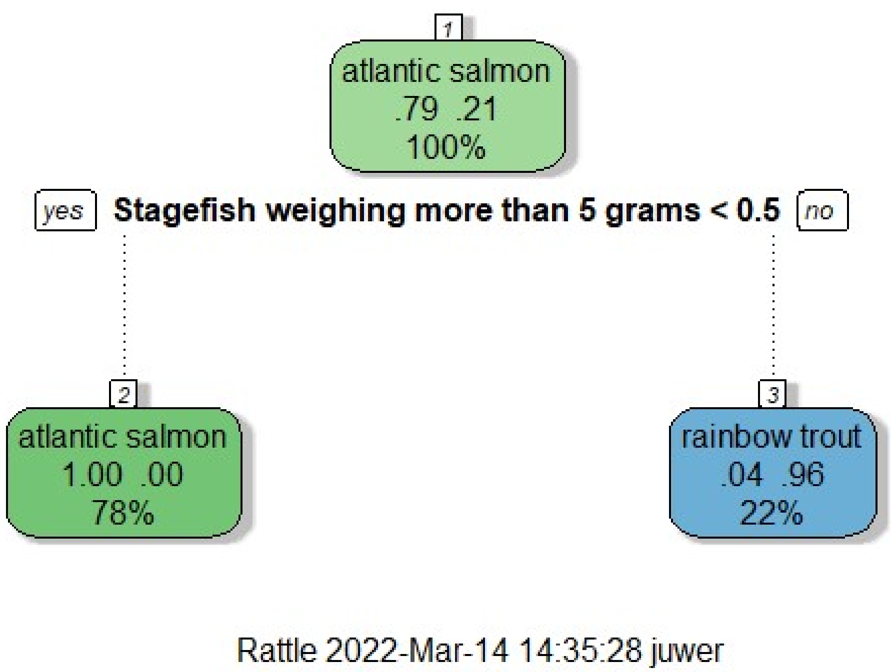
## CART
##
## 346 samples
##   5 predictor
##   2 classes: 'atlantic salmon', 'rainbow trout'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 1 times)
```

```
## Summary of sample sizes: 311, 311, 311, 312, 310, 311, ...
## Resampling results across tuning parameters:
##
##   cp         Accuracy   Kappa
##   0.0000000  0.9913445  0.9759362
##   0.4797297  0.9913445  0.9759362
##   0.9594595  0.8439122  0.2759362
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.4797297.
```

```
confusionMatrix.train(rpart.model)
```

```
## Cross-Validated (10 fold, repeated 1 times) Confusion Matrix
##
## (entries are percentual average cell counts across resamples)
##
##                   Reference
## Prediction        atlantic salmon rainbow trout
##   atlantic salmon            77.7           0.0
##   rainbow trout               0.9          21.4
##
##  Accuracy (average) : 0.9913
```

```
fancyRpartPlot(rpart.model$finalModel)
```



The accuracy was 0.9913445 and a corresponding kappa value of 0.9759362.From the confusion matrix we see that 77.7% of the instances for class atlantic salmon were correctly classified,21.4% of instances for class rainbow trout are correctly classified.0.9% of the instances of class atlantic salmon were classified incorrectly i.e.(0.9/77.7+0.9) = 0.011% which is a low prediction error.

Random forest

```
set.seed(1234)
rf.model <- train(Escaped.Species~.,
    data = NewEscapesPlus,
    method = "rf",
    metric = "Accuracy",
    trControl = trControl2)
print(rf.model)

## Random Forest
##
## 346 samples
##   5 predictor
##   2 classes: 'atlantic salmon', 'rainbow trout'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 1 times)
## Summary of sample sizes: 311, 311, 311, 312, 310, 311, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9913445  0.9759362
##   12    0.9913445  0.9759362
##   23    0.9971429  0.9922395
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 23.

confusionMatrix.train(rf.model)

## Cross-Validated (10 fold, repeated 1 times) Confusion Matrix
##
## (entries are percentual average cell counts across resamples)
##
##                     Reference
## Prediction           atlantic salmon rainbow trout
##   atlantic salmon              78.3           0.0
##   rainbow trout                 0.3          21.4
##
##  Accuracy (average) : 0.9971
```

The accuracy was 0.9971429 and a corresponding kappa value of 0.9922395.From the confusion matrix we see that 78.3% of the instances for class atlantic salmon were correctly classified,21.4% of instances for class rainbow trout are correctly classified .0.3% of the instances of class atlantic salmon were classified incorrectly i.e.(0.3/78.3+0.3) = 0.004% which is a very low prediction error.0.5% of the instances of class halibut were classified incorrectly i.e.0.5/0.5+0.5 = 0.5% which is low prediction error.

Boosted trees

```
set.seed(1234)
# Run the model

xgboost.model <- train(Escaped.Species ~ .,
                       data=NewEscapesPlus,
                       metric = "Accuracy",
                       method="xgbTree",
                       trControl = trControl2)

#print(xgboost.model)
confusionMatrix.train(xgboost.model)

## Cross-Validated (10 fold, repeated 1 times) Confusion Matrix
##
## (entries are percentual average cell counts across resamples)
##
##                   Reference
## Prediction         atlantic salmon rainbow trout
##   atlantic salmon            78.3           0.0
##   rainbow trout               0.3          21.4
##
##  Accuracy (average) : 0.9971
```

The accuracy was 0.9971.From the confusion matrix we see that 78.3% of the instances for class atlantic salmon were correctly classified,21.4% of instances for class rainbow trout are correctly classified.0.3% of the instances of class atlantic salmon were classified incorrectly i.e.0.3/78.3+0.3 = 0.004% which is a very low prediction error.

In all of the above models it is interesting to note that the percentage of class rainbow trout correctly classified has always been the same .

Comparing results of the three models

```
results <- resamples(list(CART=rpart.model, randomForest = rf.model, xgboost
= xgboost.model))
results

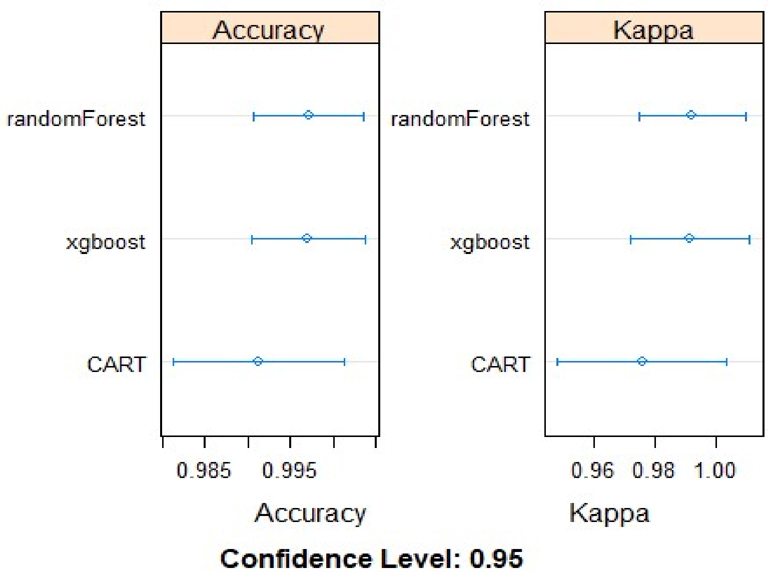
##
## Call:
## resamples.default(x = list(CART = rpart.model, randomForest = rf.model,
##  xgboost = xgboost.model))
##
## Models: CART, randomForest, xgboost
## Number of resamples: 10
## Performance metrics: Accuracy, Kappa
## Time estimates for: everything, final model fit
```

```
summary(results)

##
## Call:
## summary.resamples(object = results)
##
## Models: CART, randomForest, xgboost
## Number of resamples: 10
##
## Accuracy
##                   Min.   1st Qu. Median      Mean 3rd Qu. Max. NA's
## CART         0.9705882 0.9785714      1 0.9913445       1    1    0
## randomForest 0.9714286 1.0000000      1 0.9971429       1    1    0
## xgboost      0.9705882 1.0000000      1 0.9970588       1    1    0
##
## Kappa
##                   Min.  1st Qu. Median      Mean 3rd Qu. Max. NA's
## CART         0.9145729 0.941796      1 0.9759362       1    1    0
## randomForest 0.9223947 1.000000      1 0.9922395       1    1    0
## xgboost      0.9145729 1.000000      1 0.9914573       1    1    0
```

#To plot the results

```
scales <- list(x=list(relation="free"), y=list(relation= "free"))
dotplot(results, scales=scales, conf.level = 0.95)
```



Discussion

In the above plot we can see that the intervals, including the margin of error in each of the results overlap.So we can say that the at a confidence interval of 95% the results cannot be said to be statistically significant.

```
set.seed(1234)
partIndex <- createDataPartition(NewEscapesPlus$Average.Weight, p=0.75, list=
F)
trainData <- NewEscapesPlus[partIndex,]
testData <- NewEscapesPlus[-partIndex,]

set.seed(1234)
rpart.model.tr <- train(Average.Weight ~ ., data=trainData,
                        method="rpart",
    trControl = trControl2, na.action=na.omit)

CART



260 samples

  5 predictor



No pre-processing

Resampling: Cross-Validated (10 fold, repeated 1 times)

Summary of sample sizes: 233, 234, 234, 233, 234, 234, ...

Resampling results across tuning parameters:



  cp          RMSE      Rsquared   MAE

  0.05322534  1.466489  0.2839557  1.158104

  0.08491852  1.500595  0.2757492  1.191706

  0.23193914  1.601411  0.1829848  1.262075



RMSE was used to select the optimal model using the smallest value.

The final value used for the model was cp = 0.05322534.

#Testing rpart

```{r}
```
```

```
rpartRes <- predict(rpart.model.tr, newdata = testData)

table(rpartRes, testData$Average.Weight)

```
```

```
set.seed(1234)
rf.model.tr <- train(Average.Weight ~ ., data=trainData,
                    method="rf",
    trControl = trControl2, na.action=na.omit)
```

Random Forest

260 samples

5 predictor

No pre-processing

Resampling: Cross-Validated (10 fold, repeated 1 times)

Summary of sample sizes: 233, 234, 234, 233, 234, 234, ...

Resampling results across tuning parameters:

mtry	RMSE	Rsquared	MAE
2	1.378596	0.3816698	1.0688007
12	1.328665	0.4229677	0.9895231
23	1.359293	0.4021556	1.0133490

RMSE was used to select the optimal model using the smallest value.

The final value used for the model was mtry = 12.

```
rfRes <- predict(rf.model.tr, newdata = testData)
table(rfRes, testData$Average.Weight)

set.seed(1234)
xgb.model.tr <- train(Average.Weight ~ ., data=trainData,
```
```

```
                     method="xgbTree",
    trControl = trControl2, na.action=na.omit)
```

eta	max_depth	colsample_bytree	subsample	nrounds	RMSE	Rsquared	MAE
0.3	1	0.6	0.50	50	1.368376	0.3848469	1.029030
0.3	1	0.6	0.50	100	1.385443	0.3919067	1.035499
0.3	1	0.6	0.50	150	1.381722	0.3829764	1.042923
0.3	1	0.6	0.75	50	1.376100	0.3953277	1.031956
0.3	1	0.6	0.75	100	1.404899	0.3741640	1.042240
0.3	1	0.6	0.75	150	1.417445	0.3645452	1.047232
0.3	1	0.6	1.00	50	1.357249	0.3993699	1.013992
0.3	1	0.6	1.00	100	1.376256	0.3909685	1.016343
0.3	1	0.6	1.00	150	1.389334	0.3852609	1.018801
0.3	1	0.8	0.50	50	1.388396	0.3765759	1.040668
0.3	1	0.8	0.50	100	1.396085	0.3853923	1.055072
0.3	1	0.8	0.50	150	1.398264	0.3767857	1.062312
0.3	1	0.8	0.75	50	1.397386	0.3536539	1.034648
0.3	1	0.8	0.75	100	1.376231	0.3894760	1.014558
0.3	1	0.8	0.75	150	1.400321	0.3753852	1.034080
0.3	1	0.8	1.00	50	1.364350	0.3971603	1.023067
0.3	1	0.8	1.00	100	1.380333	0.3871975	1.019456
0.3	1	0.8	1.00	150	1.392275	0.3841518	1.021860
0.3	2	0.6	0.50	50	1.339500	0.4061160	1.006504
0.3	2	0.6	0.50	100	1.443757	0.3464275	1.068601
0.3	2	0.6	0.50	150	1.448477	0.3550407	1.079058
0.3	2	0.6	0.75	50	1.392692	0.3830041	1.031707
0.3	2	0.6	0.75	100	1.422977	0.3743910	1.066740
0.3	2	0.6	0.75	150	1.435191	0.3699580	1.067471

0.3	2	0.6	1.00	50	1.398462	0.3860300	1.020048
0.3	2	0.6	1.00	100	1.446823	0.3695851	1.053378
0.3	2	0.6	1.00	150	1.462734	0.3701208	1.062351
0.3	2	0.8	0.50	50	1.404369	0.3862770	1.048037
0.3	2	0.8	0.50	100	1.461722	0.3501235	1.108731
0.3	2	0.8	0.50	150	1.477060	0.3447612	1.114500
0.3	2	0.8	0.75	50	1.380314	0.4089919	1.017105
0.3	2	0.8	0.75	100	1.423865	0.4003031	1.044204
0.3	2	0.8	0.75	150	1.434735	0.3938744	1.056391
0.3	2	0.8	1.00	50	1.402978	0.3887387	1.025927
0.3	2	0.8	1.00	100	1.422826	0.3882850	1.049303
0.3	2	0.8	1.00	150	1.455049	0.3738384	1.071236
0.3	3	0.6	0.50	50	1.441545	0.3625392	1.064001
0.3	3	0.6	0.50	100	1.514557	0.3350915	1.110508
0.3	3	0.6	0.50	150	1.557197	0.3198989	1.139892
0.3	3	0.6	0.75	50	1.426391	0.3631518	1.065950
0.3	3	0.6	0.75	100	1.458873	0.3708681	1.074764
0.3	3	0.6	0.75	150	1.496250	0.3525912	1.106975
0.3	3	0.6	1.00	50	1.410118	0.3799732	1.040408
0.3	3	0.6	1.00	100	1.449081	0.3733872	1.073111
0.3	3	0.6	1.00	150	1.475047	0.3638738	1.093480
0.3	3	0.8	0.50	50	1.471837	0.3424982	1.093770
0.3	3	0.8	0.50	100	1.480535	0.3402804	1.105545
0.3	3	0.8	0.50	150	1.501970	0.3343383	1.110129
0.3	3	0.8	0.75	50	1.432063	0.3909052	1.071585
0.3	3	0.8	0.75	100	1.500126	0.3589215	1.112840
0.3	3	0.8	0.75	150	1.532791	0.3498373	1.144372

0.3	3	0.8	1.00	50	1.417559	0.3801350	1.058977
0.3	3	0.8	1.00	100	1.456679	0.3744384	1.092791
0.3	3	0.8	1.00	150	1.484961	0.3685025	1.112810
0.4	1	0.6	0.50	50	1.414337	0.3591685	1.066964
0.4	1	0.6	0.50	100	1.433562	0.3489345	1.076257
0.4	1	0.6	0.50	150	1.444180	0.3491761	1.094311
0.4	1	0.6	0.75	50	1.369402	0.3957474	1.009106
0.4	1	0.6	0.75	100	1.398214	0.3814774	1.028117
0.4	1	0.6	0.75	150	1.413976	0.3759848	1.047162
0.4	1	0.6	1.00	50	1.376029	0.3892959	1.022409
0.4	1	0.6	1.00	100	1.393083	0.3834684	1.024676
0.4	1	0.6	1.00	150	1.407912	0.3779248	1.031598
0.4	1	0.8	0.50	50	1.361923	0.3874296	1.022568
0.4	1	0.8	0.50	100	1.402668	0.3635669	1.045564
0.4	1	0.8	0.50	150	1.401556	0.3626868	1.056185
0.4	1	0.8	0.75	50	1.374434	0.3946535	1.029557
0.4	1	0.8	0.75	100	1.389488	0.3786122	1.045483
0.4	1	0.8	0.75	150	1.414481	0.3687376	1.050070
0.4	1	0.8	1.00	50	1.362614	0.4012859	1.010415
0.4	1	0.8	1.00	100	1.388859	0.3882815	1.017508
0.4	1	0.8	1.00	150	1.398268	0.3856913	1.026041
0.4	2	0.6	0.50	50	1.477092	0.3407976	1.093920
0.4	2	0.6	0.50	100	1.507836	0.3336704	1.125682
0.4	2	0.6	0.50	150	1.531059	0.3207360	1.137852
0.4	2	0.6	0.75	50	1.434199	0.3667160	1.061933
0.4	2	0.6	0.75	100	1.479033	0.3361377	1.096826
0.4	2	0.6	0.75	150	1.504600	0.3330562	1.116864

0.4	2	0.6	1.00	50	1.446781	0.3646615	1.059287
0.4	2	0.6	1.00	100	1.474686	0.3585495	1.076673
0.4	2	0.6	1.00	150	1.509657	0.3438382	1.097991
0.4	2	0.8	0.50	50	1.455250	0.3218004	1.071431
0.4	2	0.8	0.50	100	1.493445	0.3290639	1.109872
0.4	2	0.8	0.50	150	1.512767	0.3236078	1.131125
0.4	2	0.8	0.75	50	1.422063	0.3696841	1.062860
0.4	2	0.8	0.75	100	1.475640	0.3467806	1.101613
0.4	2	0.8	0.75	150	1.517327	0.3314358	1.136382
0.4	2	0.8	1.00	50	1.431399	0.3743968	1.048001
0.4	2	0.8	1.00	100	1.459031	0.3619477	1.075102
0.4	2	0.8	1.00	150	1.488165	0.3509430	1.089662
0.4	3	0.6	0.50	50	1.465299	0.3509270	1.070650
0.4	3	0.6	0.50	100	1.552511	0.3059979	1.145691
0.4	3	0.6	0.50	150	1.598321	0.2879912	1.197918
0.4	3	0.6	0.75	50	1.386253	0.3744212	1.052495
0.4	3	0.6	0.75	100	1.434749	0.3610614	1.099155
0.4	3	0.6	0.75	150	1.469491	0.3519630	1.125004
0.4	3	0.6	1.00	50	1.434426	0.3776929	1.063101
0.4	3	0.6	1.00	100	1.482020	0.3656771	1.095744
0.4	3	0.6	1.00	150	1.521938	0.3546417	1.127669
0.4	3	0.8	0.50	50	1.488370	0.3539212	1.087042
0.4	3	0.8	0.50	100	1.508912	0.3533567	1.100931
0.4	3	0.8	0.50	150	1.572936	0.3293630	1.147178
0.4	3	0.8	0.75	50	1.428705	0.3615452	1.053594
0.4	3	0.8	0.75	100	1.501534	0.3333249	1.109316
0.4	3	0.8	0.75	150	1.549664	0.3148096	1.142809

```
 0.4  3      0.8         1.00     50    1.444965  0.3680257  1.057087

 0.4  3      0.8         1.00    100    1.495527  0.3471696  1.098086

 0.4  3      0.8         1.00    150    1.519397  0.3431425  1.110849
```

```
results <- resamples(list(CART=rf.model.tr, randomForest = rf.model.tr, xgboo
st = xgb.model.tr))
results
```

Discussion:

`R part 1.466489`

`Random Forest 1.328665`

`XGBoost 1.444965`

From the above RMSE values for the three models we may say that Random Forest model performs better in our scenario.