Capstone CYO Project

Yaping Zhang

23 July 2020

1. Introduction

In this Harvard's Data Science Capstone CYO Project, we will analyze a large dataset using machine learning algorithms. The goal is to use an example showing learn how to work with big dataset in practice. For instance, how to explore the data, how to evaluate the prediction and which existing machine learning algorithms to choose, etc.

The dataset we choose to be analyzed is the Abalone dataset downloaded from UCI (University of California, Irvine) Machine Learning Repository and provided from Department of Primary Industry and Fisheries of Tasmania. The data includes information from physical characteristics of abalones, such as sizes, weights and sex. We will use these information to predict the age.

Why to predict the age? It is related with the economic value of abalone. The older it is, the higher the price is. The age can be physically determined by drilling its shell and counting the rings on the shell under a microscope. The number of rings plus 1.5 is the age. It is a boring and time consuming procedure. Therefore, we want to use machine learning method to predict the "Rings" using the physical characteristic information.

In this report, we will first give a shot description of the dataset. Thereupon explain the project objective in detail and the key steps to archive it. Secondly, we will explore and visualize the data, if necessary clean the data. As a result, we get insights and set up the modelling approach and evaluate them. Thirdly, we present the RMSE values from the evaluation and determine the best model for this dataset. In the end, we summarize the project and describe the difficulties and limitation of the project, based on which the further work will be concluded.

1.1. Dataset description

Let us first install the necessary packages for the project:

```
###Loading packages
if(!require(tidyverse)) install.packages("tidyverse",
                        repos = "http://cran.us.r-project.org")
if(!require(caret)) install.packages("caret",
                        repos = "http://cran.us.r-project.org")
if(!require(data.table)) install.packages("data.table",
                        repos = "http://cran.us.r-project.org")
if(!require(DataExplorer)) install.packages("DataExplorer",
                        repos = "http://cran.us.r-project.org")
if(!require(GGally)) install.packages("GGgally",
                        repos = "http://cran.us.r-project.org")
if(!require(car)) install.packages("car",
                        repos = "http://cran.us.r-project.org")
```

```r
if(!require(gam)) install.packages("gam",
                        repos = "http://cran.us.r-project.org")
```

Then wrangle the dataset from the following link and see the structure of the original dataset:

```r
#download and wrangle the dataset
varnames<-c("Sex", "Length", "Diameter", "Height", "Whole weight",
            "Shucked weight", "Viscera weight", "Shell weight", "Rings")
abalone <- read.csv(url(
  "https://archive.ics.uci.edu/ml/machine-learning-databases/abalone/abalone.data"),
                header = FALSE, col.names = varnames)

#Data structure
str(abalone)
```

```
## 'data.frame':    4177 obs. of  9 variables:
##  $ Sex           : Factor w/ 3 levels "F","I","M": 3 3 1 3 2 2 1 1 3 1 ...
##  $ Length        : num  0.455 0.35 0.53 0.44 0.33 0.425 0.53 0.545 0.475 0.55 ...
##  $ Diameter      : num  0.365 0.265 0.42 0.365 0.255 0.3 0.415 0.425 0.37 0.44 ...
##  $ Height        : num  0.095 0.09 0.135 0.125 0.08 0.095 0.15 0.125 0.125 0.15 ...
##  $ Whole.weight  : num  0.514 0.226 0.677 0.516 0.205 ...
##  $ Shucked.weight: num  0.2245 0.0995 0.2565 0.2155 0.0895 ...
##  $ Viscera.weight: num  0.101 0.0485 0.1415 0.114 0.0395 ...
##  $ Shell.weight  : num  0.15 0.07 0.21 0.155 0.055 0.12 0.33 0.26 0.165 0.32 ...
##  $ Rings         : int  15 7 9 10 7 8 20 16 9 19 ...
```

As we can see there are 4177 observations with 9 variables. The variable "Rings" is the integer variable we have to predict, called as "output". The other 8 variables are "predictors", which can be separated into three types:

a. Size. It includes three continuous variables(in millimeters): "Length", "Height" and "Diameter".

b. Weight. There are four continuous variables(in grams) included:

   "Whole weight" for the whole abalone,

   "Shucked weight" for the weight of meat,

   "Viscera weight" for the gut weight after bleeding,

   "Shell weight" for its weight after drying.

   We could already expect some logical connections between the predictors. We will discuss it in detail in data exploration.

c. Sex. One predictor called "Sex" and includes three levels: F(female), I(Infant), M(male).

Additionally, the ranges of the continuous variables were already scaled for usage. Next, let us determine the objective and key steps of the project in the following chapter.

1.2. Define the objective of the project

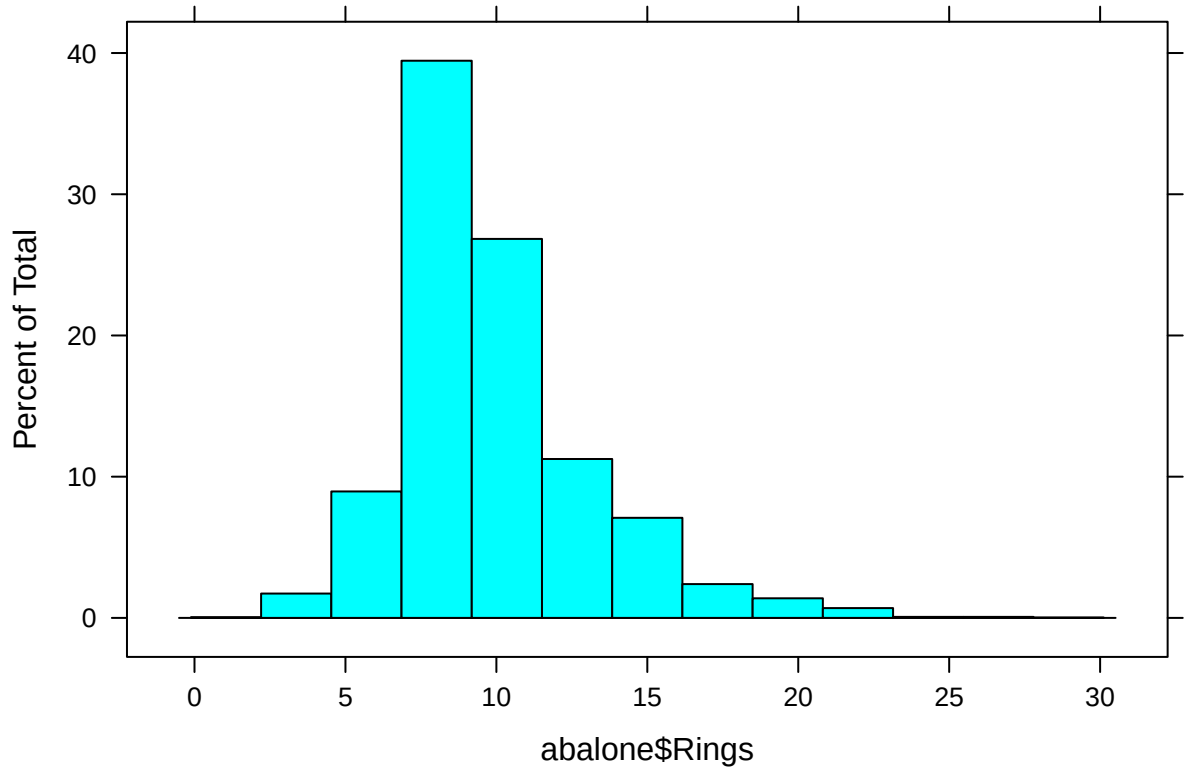
In this chapter we will discuss how to evaluate the performance of the prediction and which model to choose, as the objective of the project.

Let us start with how to evaluate the prediction. We need first to have a look at the output variable "Rings":

```r
#explore the output variable
summary(abalone$Rings)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   8.000   9.000   9.934  11.000  29.000
```

```r
histogram(abalone$Rings)
```




```r
#To check if there is output classes with small count
abalone%>%group_by(Rings)%>%mutate(n=n())%>%filter(n()<5)%>%top_n(10)%>%select(Rings,n)
```

```
## Selecting by n
```

```
## # A tibble: 9 x 2
## # Groups:   Rings [7]
##   Rings     n
##   <int> <int>
## 1     1     1
## 2    26     1
## 3    29     1
## 4     2     1
## 5    27     2
## 6    25     1
## 7    27     2
```

```
## 8    24     2
## 9    24     2
```

From the the above information, it looks like that this project is to solve a classification problem. For classification outcomes we often use confusion matrix to calculate the accuracy. However, there are 29 classes and some of the classes have very little volume of samples. With this amount of classes, the accuracy can be quite low. Furthermore, to implement classification algorithms, the sampling for each class will not be enough.

To compromise it, we could weather categorize the 29 classes into 3 classes(eg., 1-5 as 1st class, 5-15 the second,16-29 as the third), or consider it as a regression problem and use the loss function to calculate the Root Mean Square Error (RMSE) for the evaluation. For demonstration purpose we will choose the second solution here. Therefore, we considered this as a regression problem and write a function for RMSE:

```
#evaluation of models
RMSE <- function(true_Rings, pred){ sqrt(mean((true_Rings - pred)^2))}
```

The RMSE value means: if for example is 3.5, the predicted rings are in average 3.5 away from the real number of rings.

Moreover, which machine learning models can be used? We are using labeled input variables to predict an output variable, thus, supervised machine learning algorithms will be used. Furthermore, we need to choose regression models for the regression problem. Here we will first use a simple linear regression model "lm", then choose more sophisticated and common used regression models such as k-Nearest Neighbors, RandomForest and gamLoess. Additionally, to avoid over-fitting we will use cross validation.

Now the objective of the project is clear, let us summarize the key steps of the project in next chapter.

1.3. Key steps

All things considered, we conclude the key steps of the project:

a. Data exploration and cleaning. This is an important part of the project to guarantee a trustworthy result. We will have a deep look into different predictor types by examining their distribution, detecting and removing the odd data and the correlation between the predictors.

b. Modelling. Separate the dataset into train and test sets. Use the different models to train on the train dataset. Predict on the test set.

c. Resulting. List all the results and determine the best model with the lowest RMSE.

d. Summarizing. It is always essential for a project. In this chapter, we will summarize the project, explain its limitation and future work.

We put the step a and b in the next chapter "Methods" to explain in detail.

2. Methods

In this chapter we will first clean and explore the data and based on the insights carry out the modelling approach.

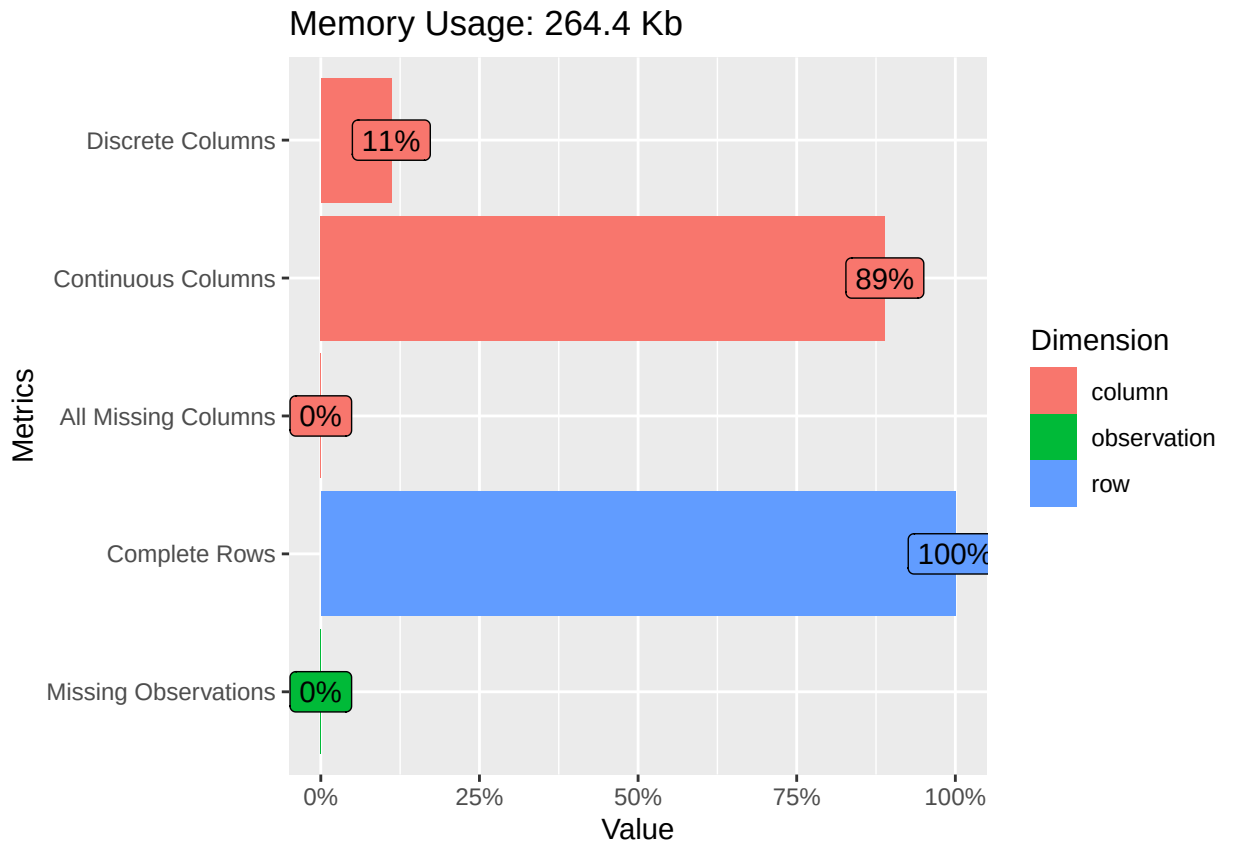
2.1. Data exploration and cleaning

Let us first detect if there are missing or zero values in the whole dataset. Then look into each types to explore the data and to clean the odds based on the physical logic. After that, we will look at the correlation on all predictors.

2.1.1. Missing or zero values

If missing or zero value exists, we need to remove or replace it.

```
#detect any missing values
plot_intro(abalone)
```




```
## detect zero values and show the column which contains 0 values
which(!!colSums(abalone==0))
```

```
## Height
##      4
```

We found out there is no missing values but 0s in the fourth column "Height". Let us have a further look into the observations:

```
#list observations with 0
abalone%>%filter(Height==0)
```

```
##   Sex Length Diameter Height Whole.weight Shucked.weight Viscera.weight
## 1   I  0.430     0.34      0        0.428         0.2065         0.0860
## 2   I  0.315     0.23      0        0.134         0.0575         0.0285
##   Shell.weight Rings
## 1       0.1150     8
## 2       0.3505     6
```

Two observations have height with 0. This could be the reason of missing data. Let us replace the 0s with the median of "Height" values which have the same rings:

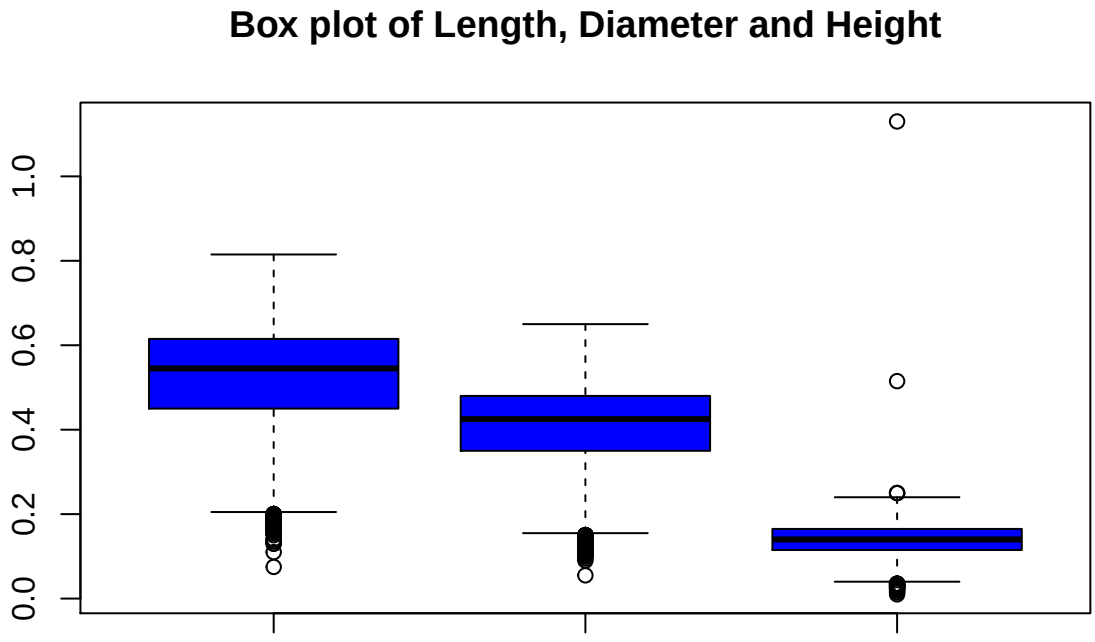
```
# name our data abalone_cleaned as cleaned dataset
abalone_cleaned<-abalone

#use the data-table function to replace 0 with NA
setDT(abalone_cleaned)[Height==0, Height := NA,]

##replace the "NA"s with median value with the same rings
abalone_cleaned[, Height := replace(Height, is.na(Height), median(Height, na.rm = TRUE)),
                by = Rings]
```

The zero values are corrected. We renamed our dataset to "abalone_cleaned" and move on to analyse each type of predictors.

2.1.2. Size predictors

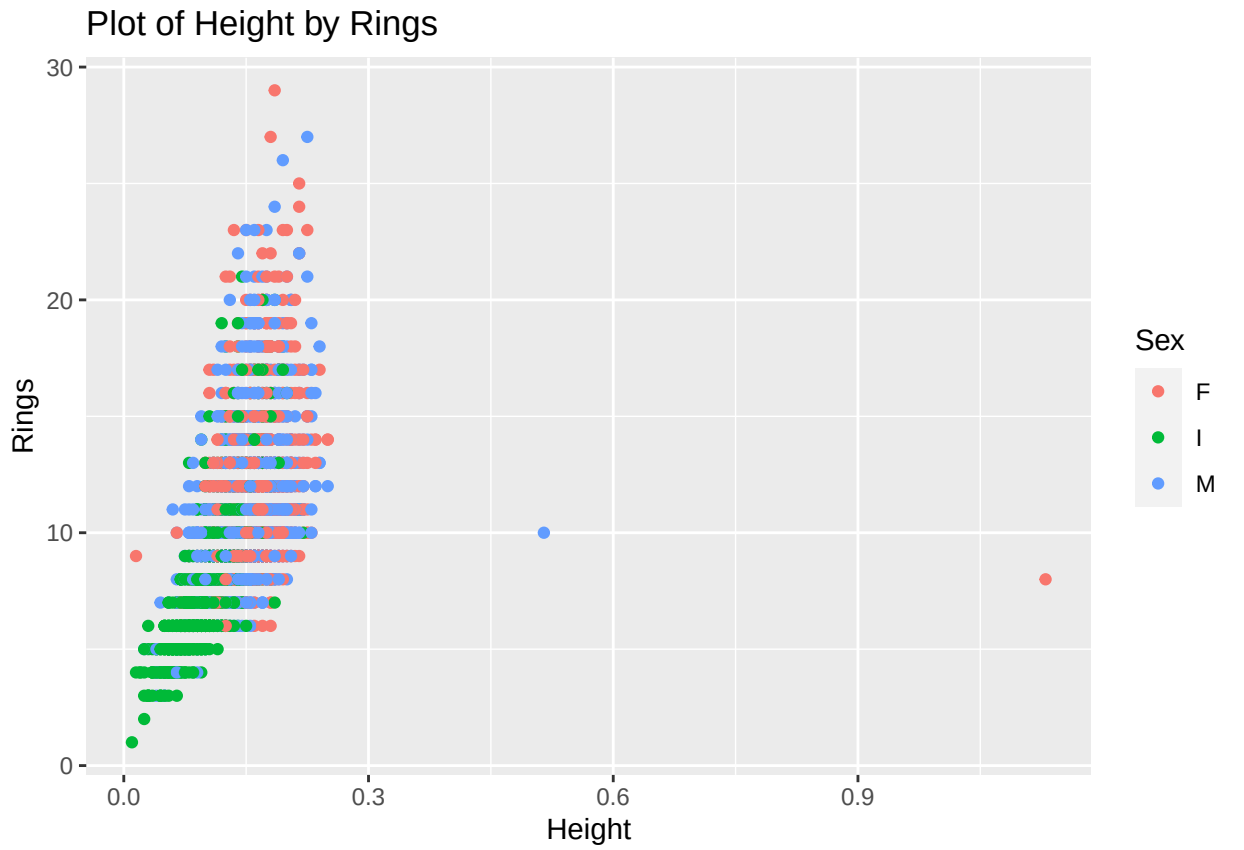
In this chapter, we will explore the outliers using box plot and the data distribution using histogram. In addition, we will color some plots with sex to have a side view on how this type of predictor influences.

```
# detect if there are extreme outliers from the three size predictors
boxplot(abalone_cleaned$Length,abalone_cleaned$Diameter,abalone_cleaned$Height,
        col = "Blue",main="Box plot of Length, Diameter and Height")
```



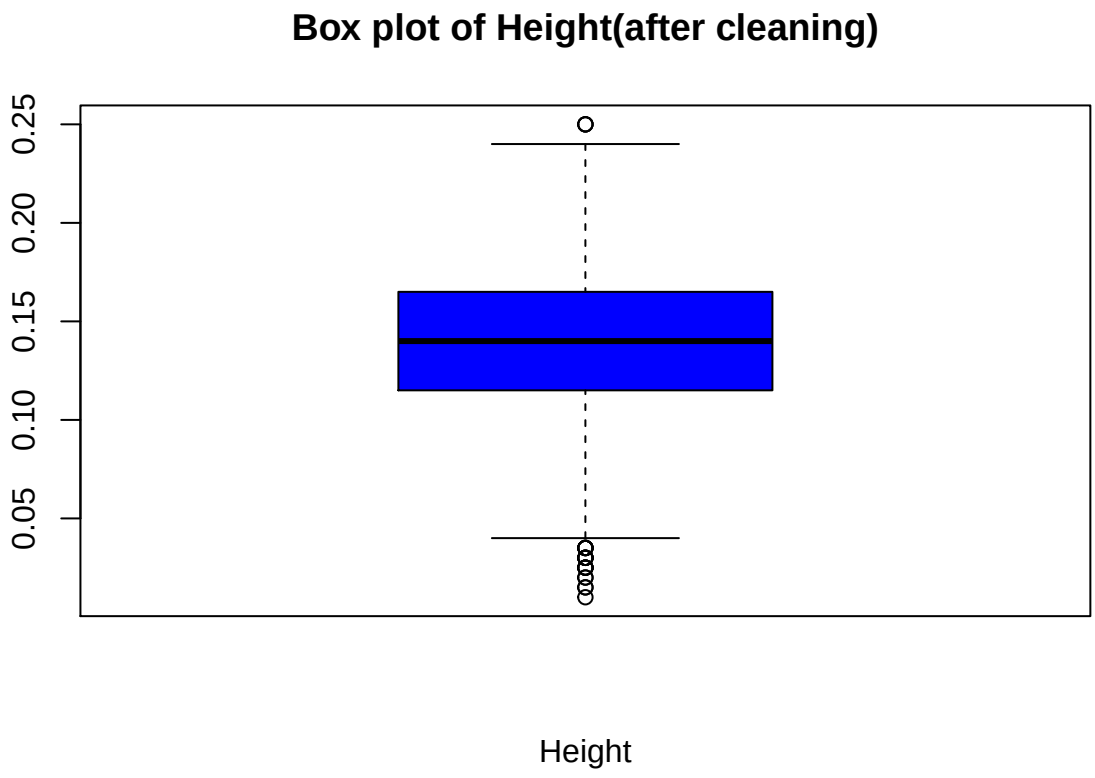
We can visualize two extreme outliers in "Height". Let us have a deeper look into "Height":

```
#plot height ~ rings to detect outlier
abalone_cleaned%>%ggplot(aes(Height,Rings,color=Sex))+geom_point()+
  ggtitle("Plot of Height by Rings")
```



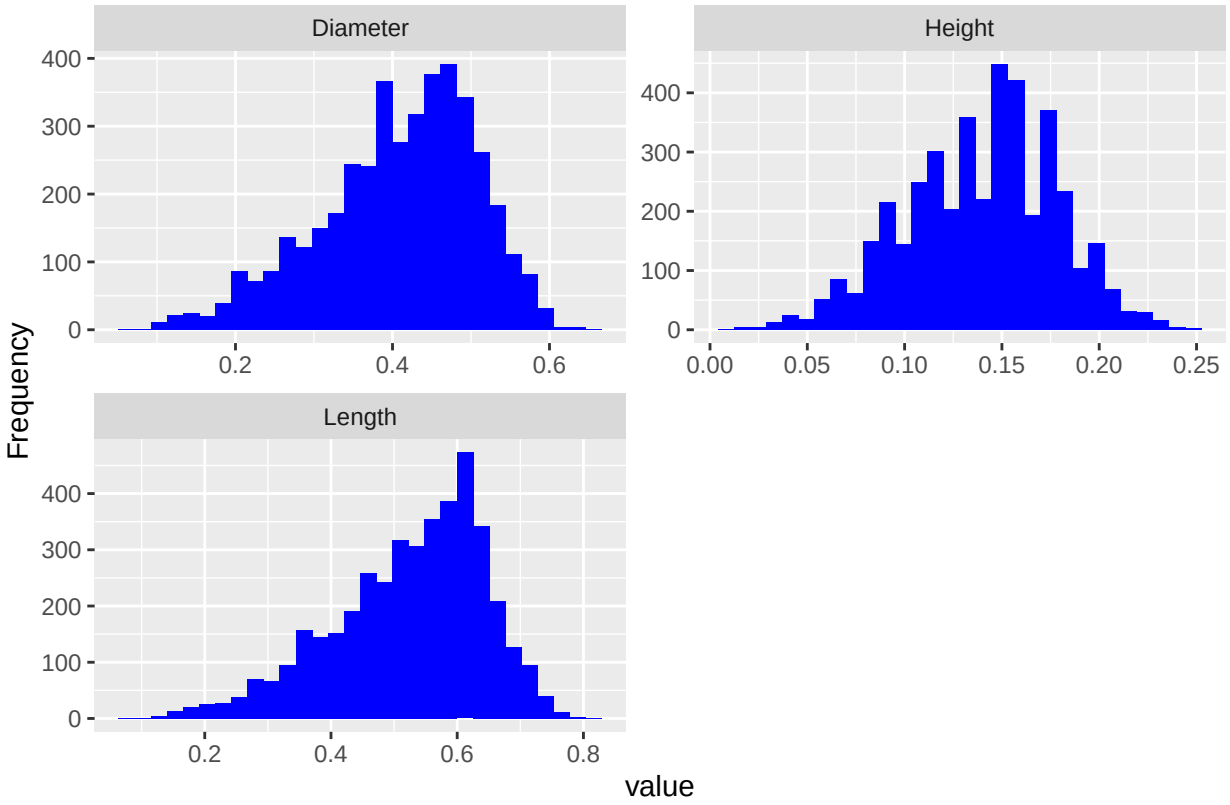
Nearly all abalone have height smaller than 0.3, except two extreme values(one male and one female). We remove them out of the data. Furthermore, we see that the height is positively related to rings. Moreover, the infant abalone has relatively smaller height than the adult while the male and female have similar values.

```
#remove the observation with extreme height value(>0.3) and
abalone_cleaned<-abalone_cleaned%>%filter(Height<0.3)
##box plot the height afterwards
boxplot(abalone_cleaned$Height,xlab="Height",col = "Blue",
        main="Box plot of Height(after cleaning)")
```



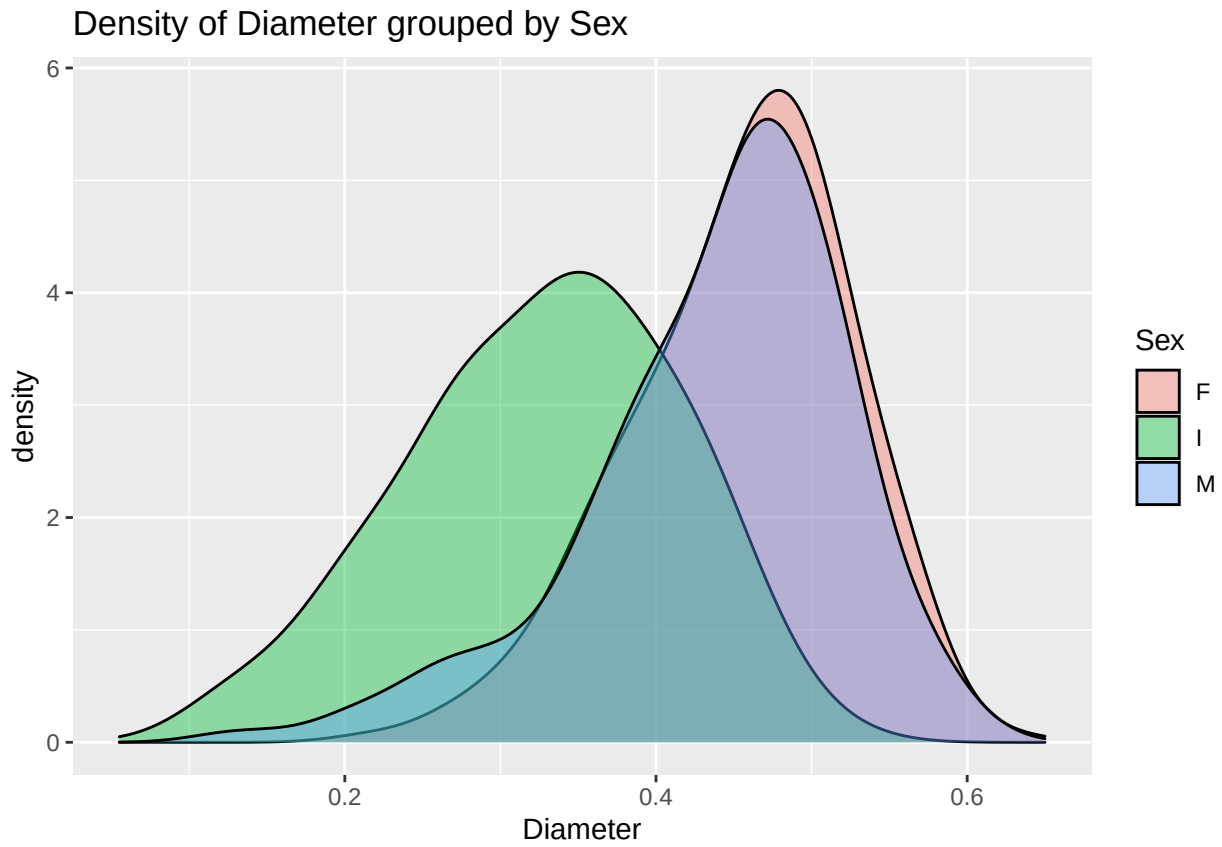
The next is to have a look at the distribution. This biological data is expected to be approximately normal distributed. The confirmation of normal distribution is fundamental as some machine learning models are based on assumption of approximate normal distribution of the predictors.

```
#see the distribution of the size predictors
plot_histogram(abalone_cleaned[,2:4],geom_histogram_args = list("fill" = "blue",bins=30),
               nrow = 2L, ncol = 2L)
```



We see the "Diameter" and "Length" are very similar left skewed distributions(negative distribution). Thus, there is a suspicion of multicollinearity. We will explore this issue in the chapter "Correlation Exploration". Furthermore, the negative distribution could be from the influence of different, let us plot the Diameter grouped with sex as an example to have a deeper look:

```
#let us see histogram
ggplot(data=abalone_cleaned, aes(x=Diameter, group=Sex, fill=Sex)) +
    geom_density(adjust=1.5, alpha=.4)+ggtitle("Density of Diameter grouped by Sex")
```



We see the adults are still left skewed but infant is close to normal distribution. In addition we detect the distributions of male and female abalone are very similar. Now, let us move on to the next type.

2.1.3. Weight predictors

Same like the size predictors, we check on outliers and distribution. However, weight predictors have extra logical relationships with each other:

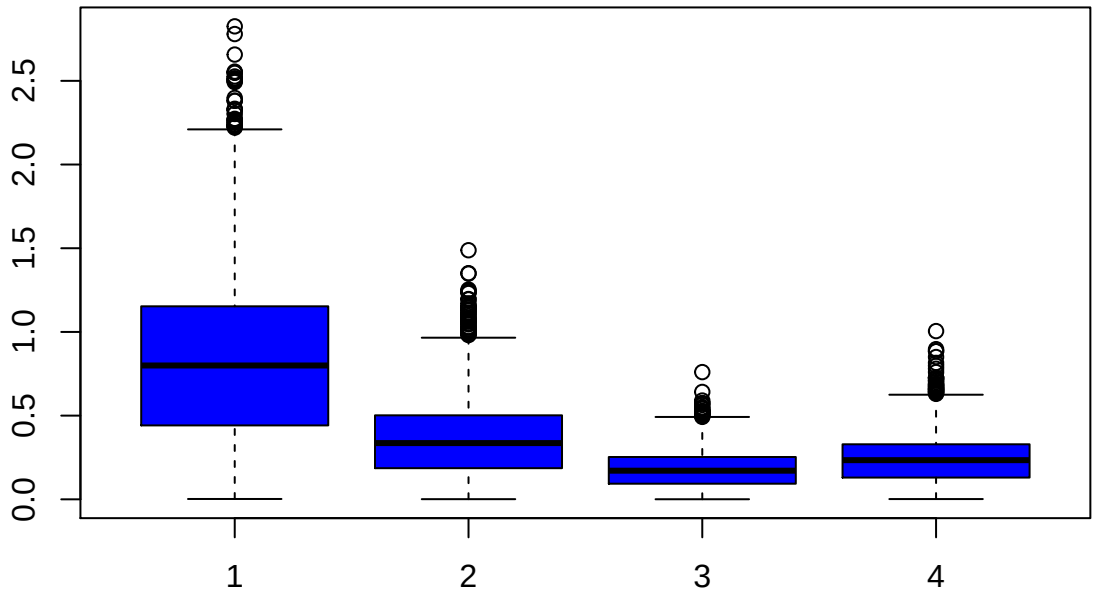
a. Whole weight > Shucked weight. The whole weight is the sum of shuck weight and weight of shell.

b. Shucked weight >= Viscera weight. The meat includes the organic.

c. Shell weight <= (Whole weight - Shucked weight). Shell weight is dried weight, while the right side result is the wet shell weight.

Let us start with box plot to see if there are extreme outliers:

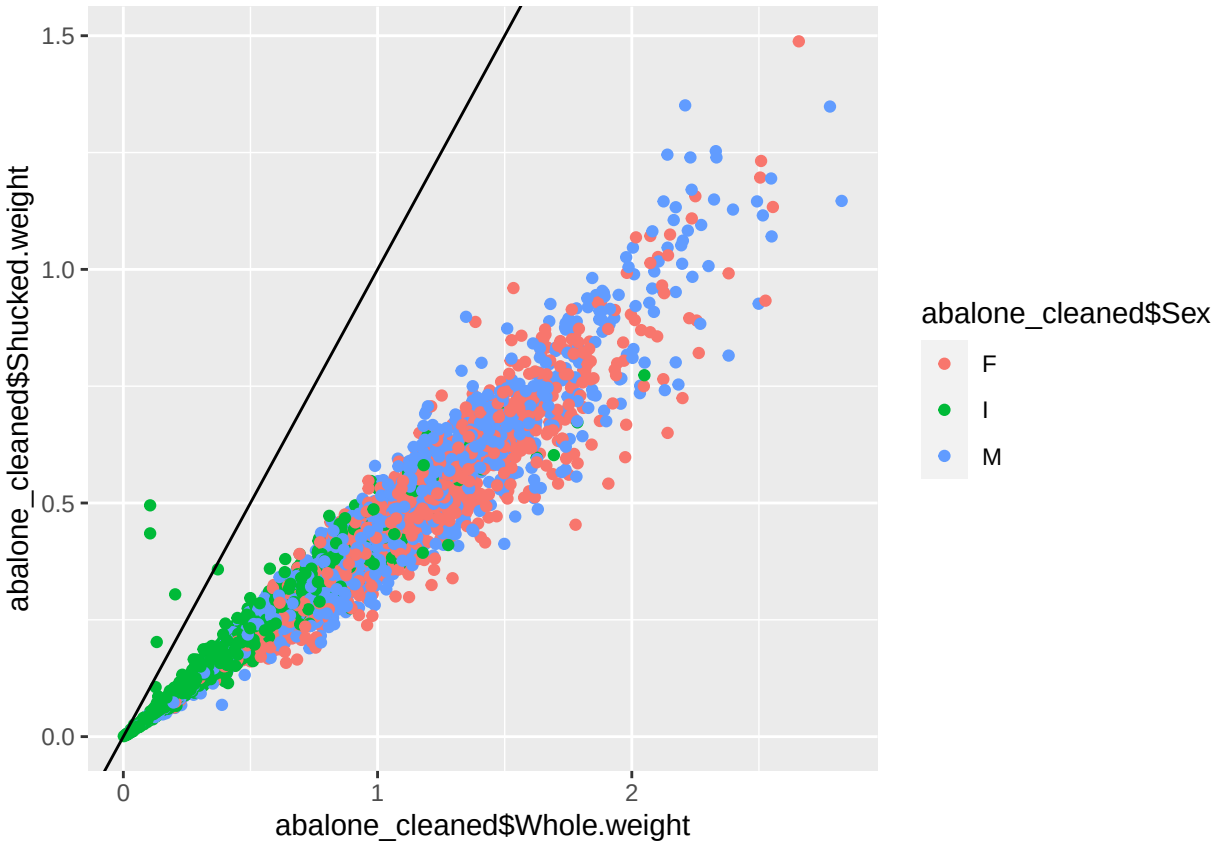
```
# detection of outliers from weight predictors
boxplot(abalone_cleaned$Whole.weight,abalone_cleaned$Shucked.weight,
        abalone_cleaned$Viscera.weight, abalone_cleaned$Shell.weight,col = "Blue")
```



We see no extreme outliers here.

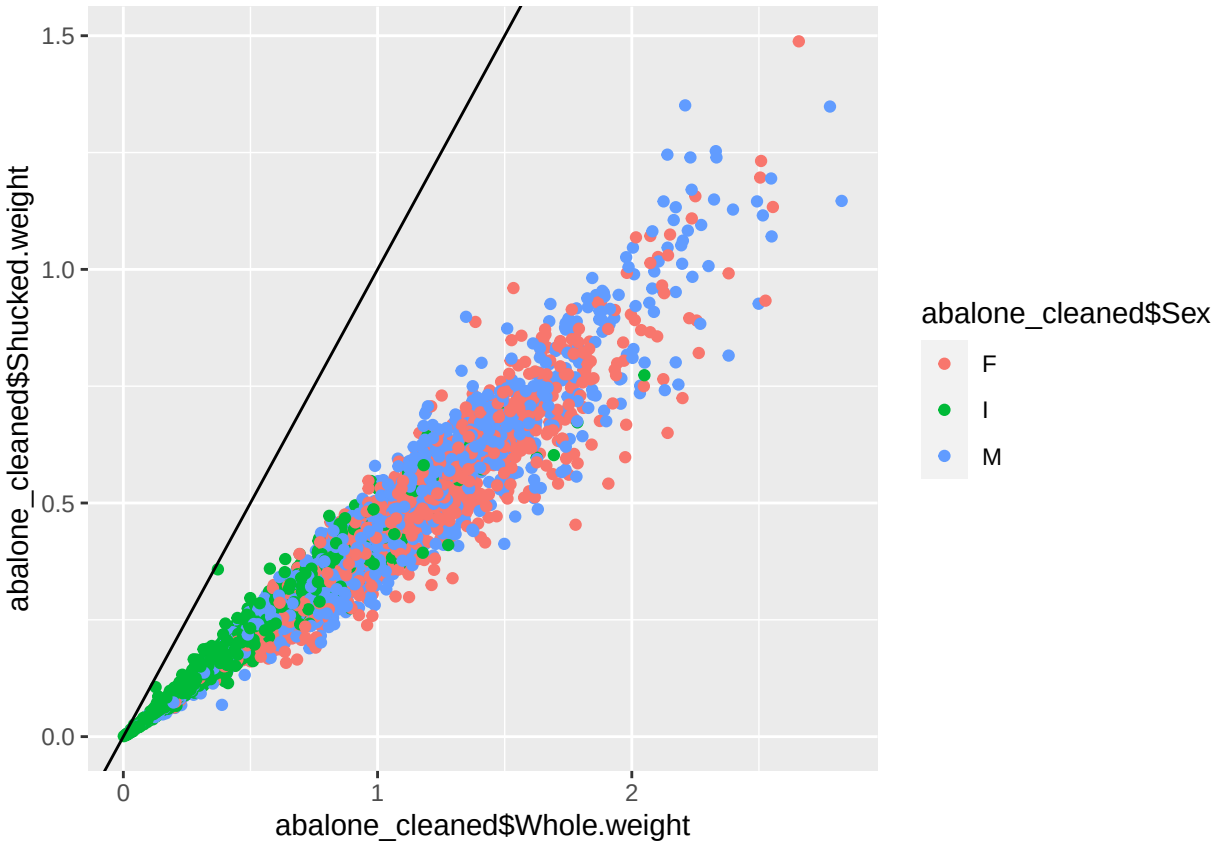
Then we check the characteristic "a" with scatter plot with ab-line:

```
# detection if whole weight > shucked weight with an ab-line with slope=1
qplot(abalone_cleaned$Whole.weight,abalone_cleaned$Shucked.weight,
      col=abalone_cleaned$Sex)+geom_abline(slope=1)
```



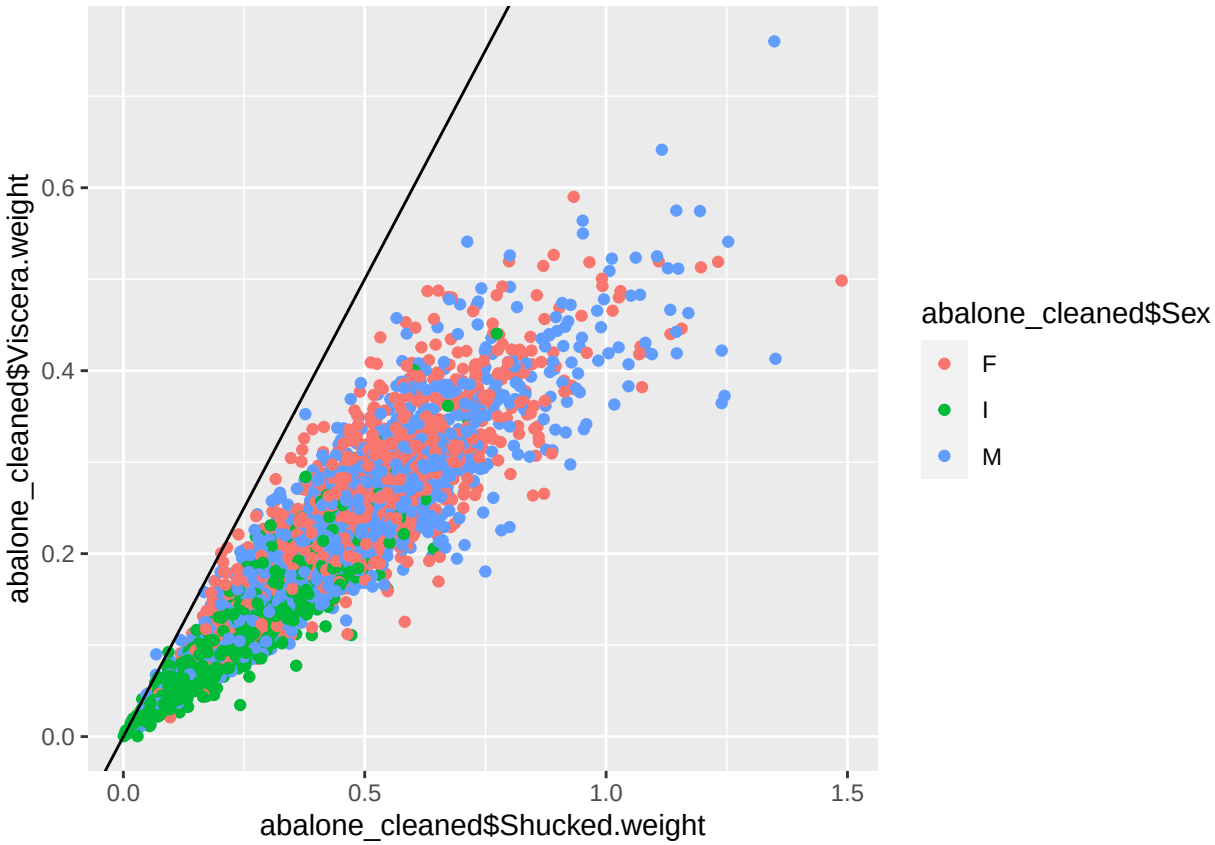
There are four points above the ab-line which means that these four observations have higher shucked weight than whole weight which is not logically possible. As we don't know which predictor has the wrong value, we will remove the observations:

```
# remove observations which whole weight smaler than shucked weight
abalone_cleaned<-abalone_cleaned%>%filter(Whole.weight>Shucked.weight)
#plot if outliers are removed
qplot(abalone_cleaned$Whole.weight,abalone_cleaned$Shucked.weight,
      col=abalone_cleaned$Sex)+geom_abline(slope=1)
```



Next, examine the characteristic "b" in the same way:

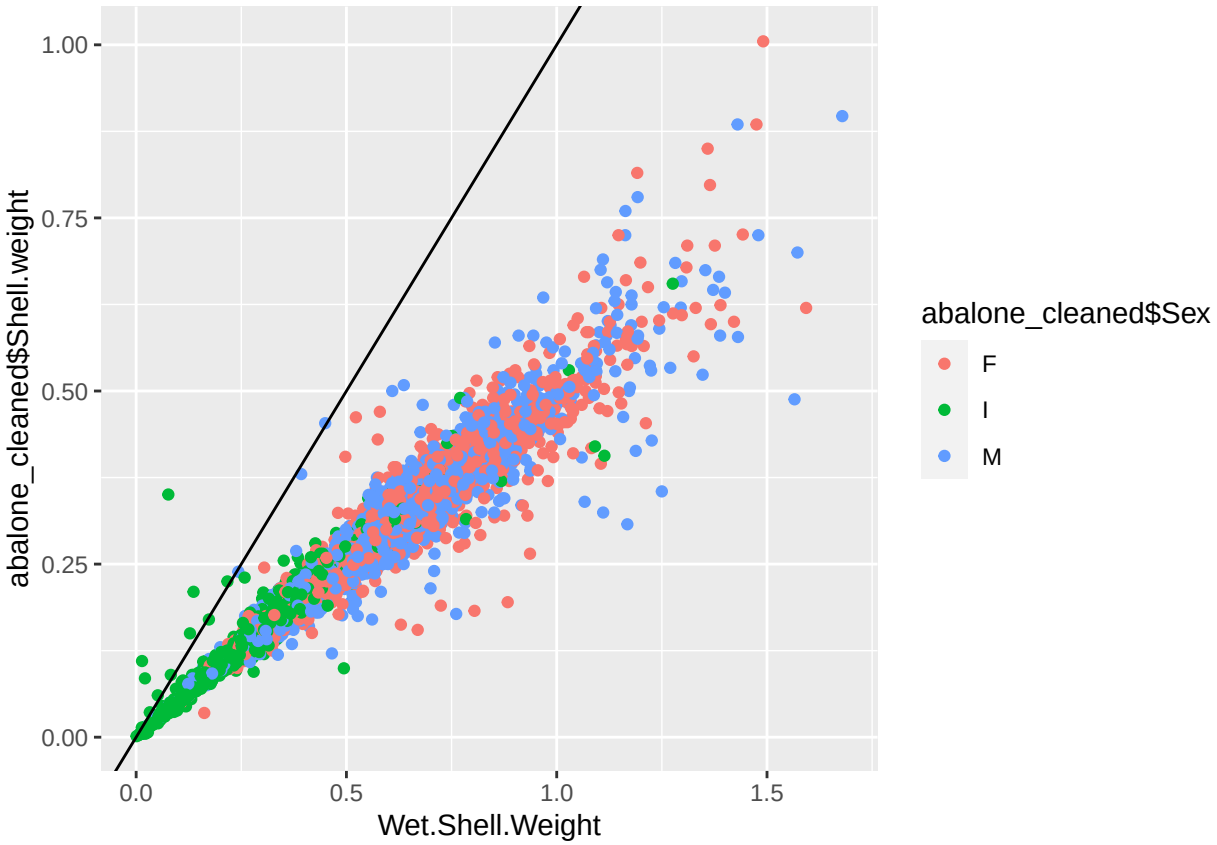
```
#plot to see if all shucked weight > viscera weight
qplot(abalone_cleaned$Shucked.weight,abalone_cleaned$Viscera.weight,
      col=abalone_cleaned$Sex)+geom_abline(slope=1)
```




```
#Remove the observation with shucked weight < viscera weight
abalone_cleaned<-abalone_cleaned%>%filter(Shucked.weight>=Viscera.weight)
```

Finally with characteristic "c":

```
#detect if all wet shell weight > dry shell weight
Wet.Shell.Weight<-abalone_cleaned$Whole.weight-abalone_cleaned$Shucked.weight
qplot(Wet.Shell.Weight,abalone_cleaned$Shell.weight,col=abalone_cleaned$Sex)+
  geom_abline(slope=1)
```




```
## remove the observations which donot fulfill the requirement
abalone_cleaned<-abalone_cleaned%>%filter((Whole.weight-Shucked.weight)>=Shell.weight)
```

We already cleaned out the observations with erroneous weight predictors. We can also see that the infant has smaller weight than the male and female. From the different plots we already get the side view of sex distribution. Let us summarize it in the next chapter.

2.1.4. Sex predictor

As we already saw and analysed from the above plots with sex grouped, we won't plot them here again. We can summarize two insights from above:

a. The infants have relatively smaller size and weight comparing to adult. This is logical.

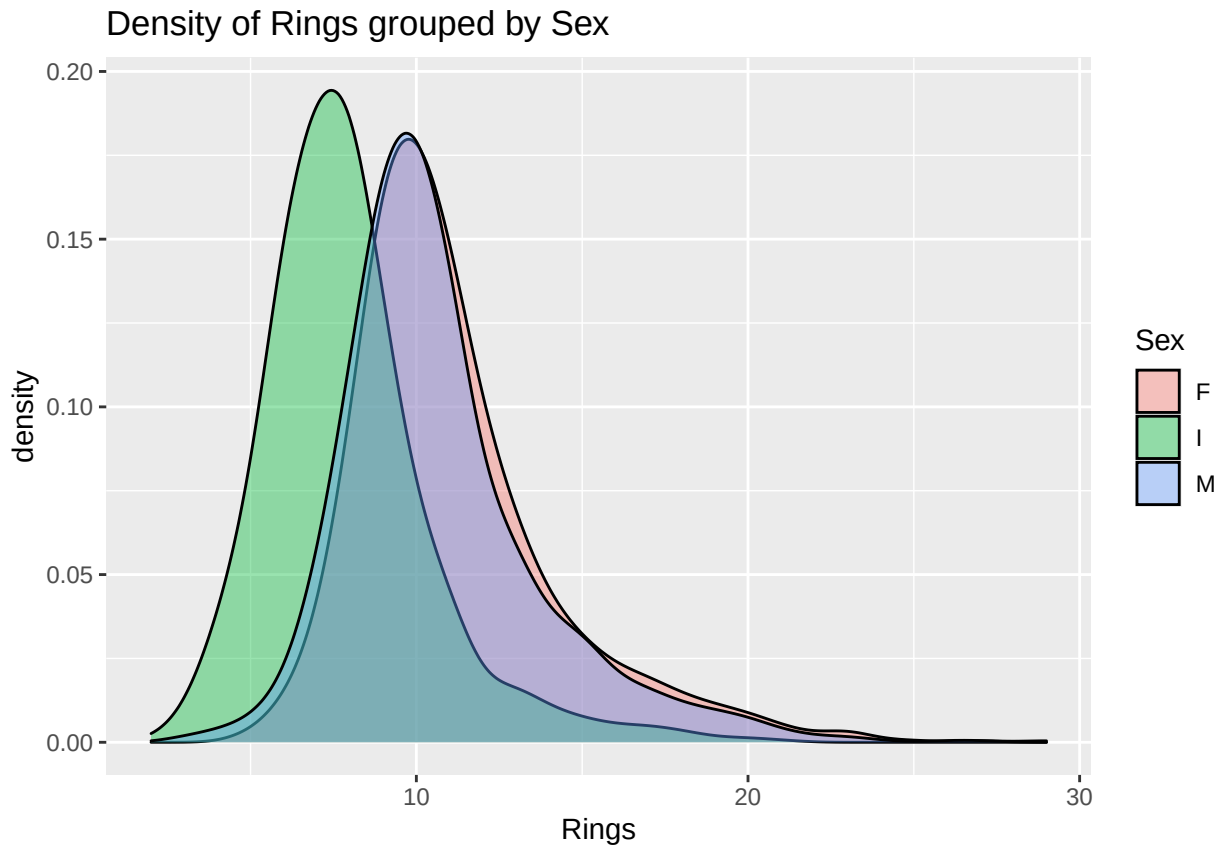
b. Between male and female there is no significant differences. This character does not supporting the prediction but adding instability of its result. Therefore, we can consider to reduce the level of this predictor to infant and adult using binary variable.

Let us then have a look if there is prevalence existing and how the distribution looks like:

```
# detection of prevalence
summary(abalone_cleaned$Sex)
```

```
##    F    I    M
## 1306 1325 1525
```

```
##plot the distribution of sex to rings
ggplot(data=abalone_cleaned, aes(x=Rings, group=Sex, fill=Sex)) +
    geom_density(adjust=1.5, alpha=.4)+ggtitle("Density of Rings grouped by Sex")
```



From the plots above, we see Infant has smaller median value of Rings than the adult while male and female are approximatly normal distributed with the rings. Additionally, there is no prevalence.

2.1.5. Correlation exploration

From the plots before, we noticed, that the predictors in same category are highly correlated. Therefore, the multicollinearity is suspected in the dataset. The multicollinearity occurs when two or more predictors are highly correlated. This can increase the standard error, unstable the regression coefficients and make the estimates sensitive to minor changes. In this chapter, we will detect if this problem really exists and solve it.

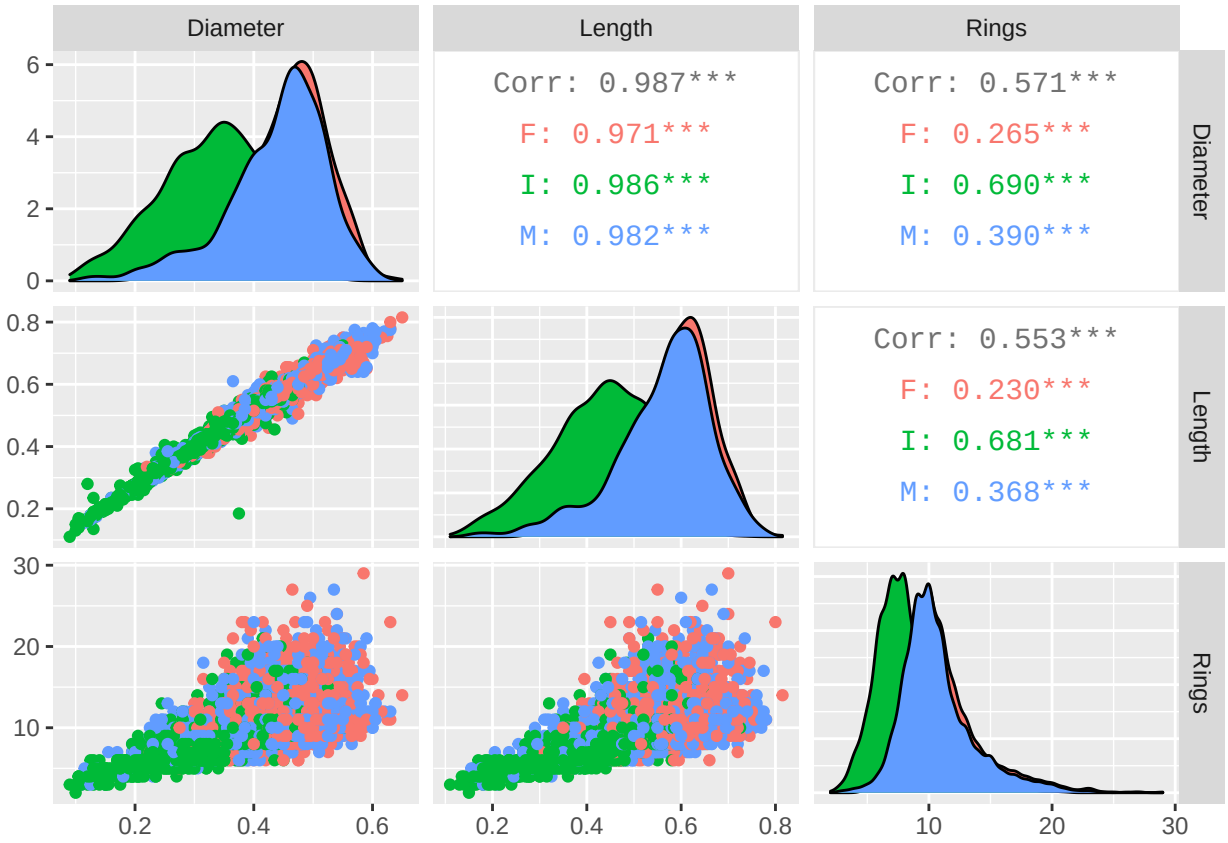
First let us have a look at the correlation matrix among all predictors:

```
#correlation coefficients between all predictors
ggcorr(abalone_cleaned[,-1],label = TRUE, hjust=0.75,label_size = 3, label_round = 2) +
  scale_alpha_manual(values = c("TRUE" = 0.25, "FALSE" = 0)) +
  guides(color = FALSE, alpha = FALSE)
```

The visualization shows high correlation values between all predictors(>0.8), while all predictors are moderate correlated with the output rings(between 0.4 and 0.7). Let us analyse it in detail:

a. Length and Diameter are most correlated pair(0.99). Let us plot them to confirm:

```
# Correlation coefficients between Length and Diameter including Rings
pair<-abalone_cleaned%>%select(Sex,Diameter,Length,Rings)
ggpairs(pair,columns = 2:4,ggplot2::aes(colour=Sex))
```

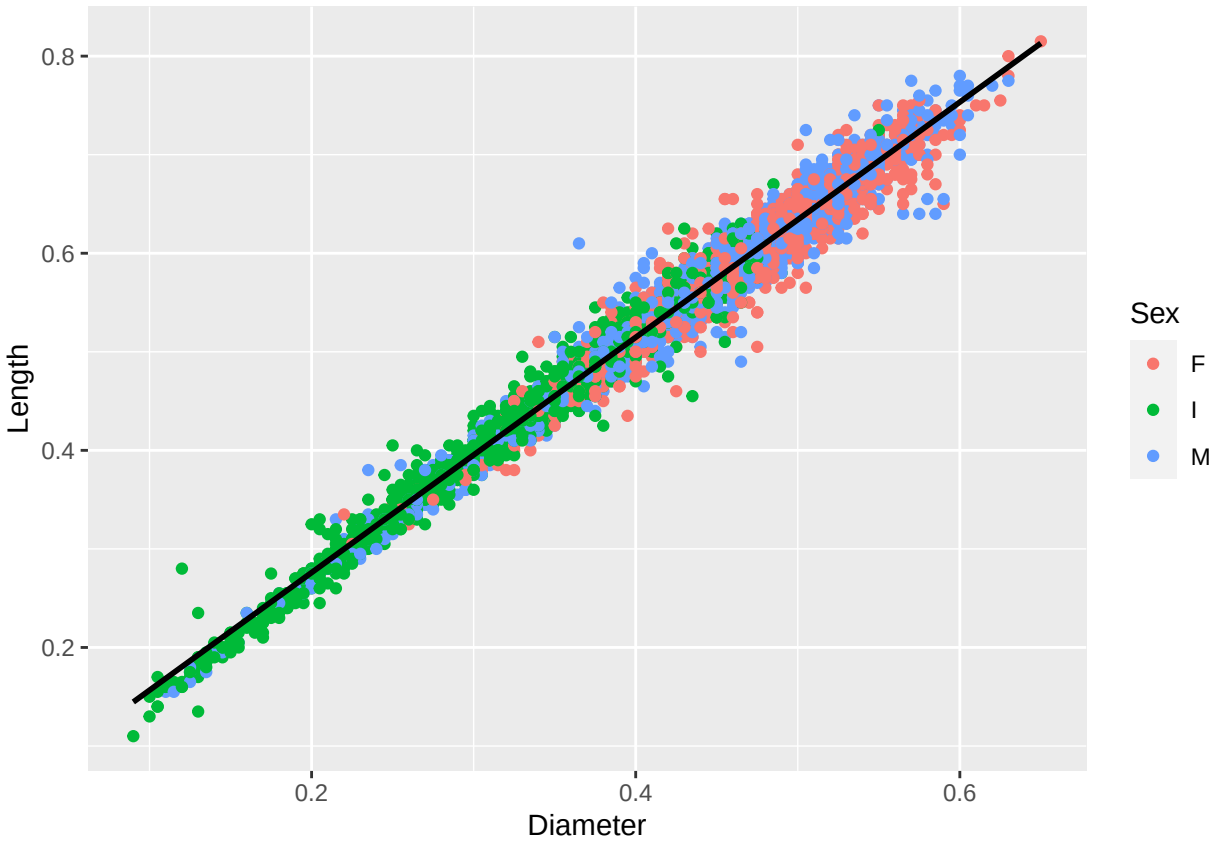


Except the truly high correlation between predictors, we also noticed again that female and male have no significant differences. Furthermore, comparing to infant, their Diameter and Length are less correlated with Rings.

In addition, one infant observation could be visualized as outlier from the point plot between Length and Diameter. It has larger diameter with smaller length(Diameter>0.3 and Length<0.2). We can remove it:

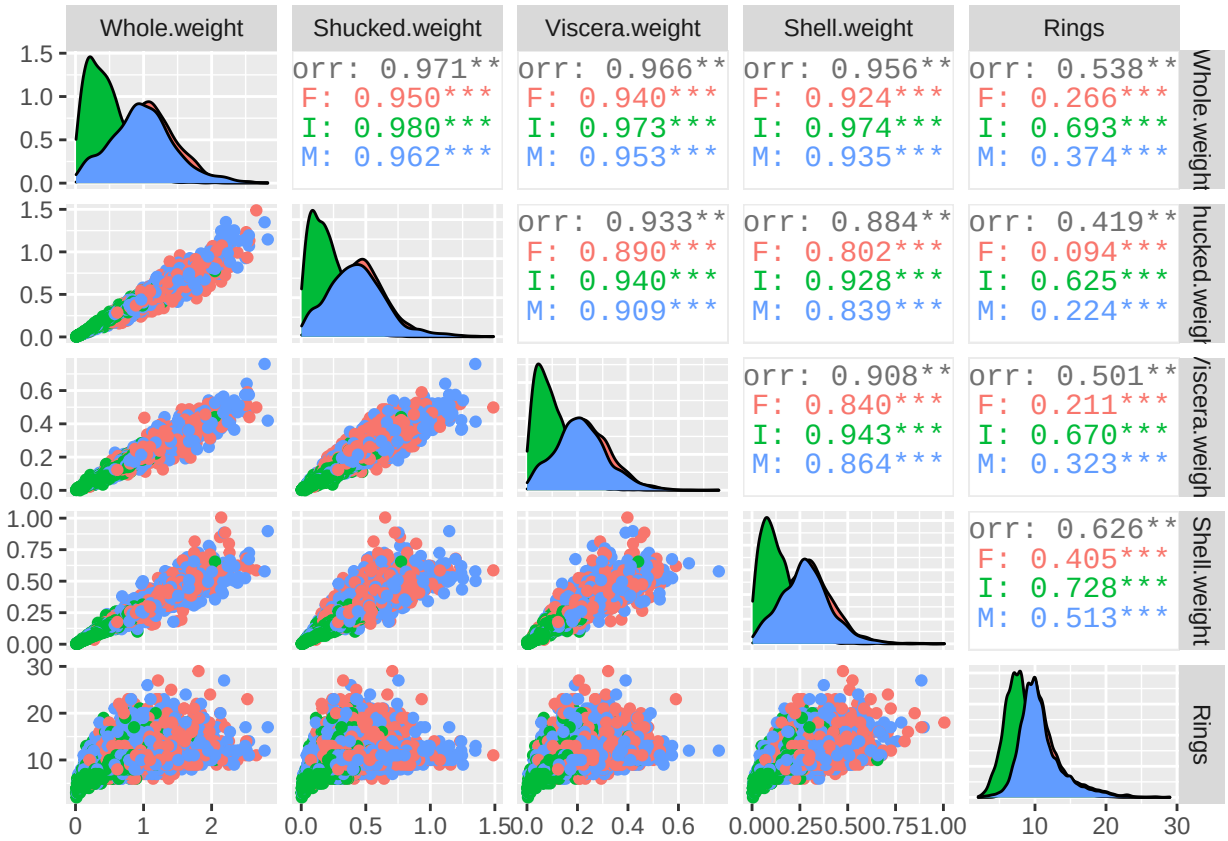
```
# remove odd value with diamter and height
abalone_cleaned<-abalone_cleaned%>%filter(!(Diameter>0.3&Length<0.2))
##plot diameter and height
qplot(Diameter,Length,data=abalone_cleaned,color=Sex)+geom_smooth(method="lm",col="Black")
```

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



b. The 4 weight predictors are highly correlated with each other(>0.95), lets plot them to confirm it:

```
# correlation between weight predictors including Rings
weight<-abalone_cleaned%>%select(Sex,Whole.weight,Shucked.weight,Viscera.weight,
                                 Shell.weight,Rings)
ggpairs(weight,columns = 2:6,ggplot2::aes(colour=Sex))
```



This is expected because of the logical relationship between the weights. From the point plot we could confirm the correlation. Here we noticed again that adult has less correlation with rings than infant. Therefore, we should reduce the level of sex to infant and adult.

c. Determine multicollinearity by VIF and variable Variance

```
#VIF and variance of predictors
vif(lm(Rings~.,data=abalone_cleaned))
```

```
##                       GVIF Df GVIF^(1/(2*Df))
## Sex                1.539552  2        1.113906
## Length            43.089502  1        6.564259
## Diameter          44.549801  1        6.674564
## Height             6.795232  1        2.606767
## Whole.weight     122.108403  1       11.050267
## Shucked.weight    31.952392  1        5.652645
## Viscera.weight    17.942170  1        4.235820
## Shell.weight      23.616017  1        4.859631
```

```
var(abalone_cleaned[,-1])
```

```
##                     Length    Diameter      Height Whole.weight Shucked.weight
## Length         0.014220980 0.011615634 0.004102032   0.05403243    0.023777703
## Diameter       0.011615634 0.009732208 0.003415391   0.04467796    0.019552281
## Height         0.004102032 0.003415391 0.001460014   0.01660401    0.007093548
```

```
## Whole.weight   0.054032425 0.044677965 0.016604013   0.23921367    0.105121397
## Shucked.weight 0.023777703 0.019552281 0.007093548   0.10512140    0.049030213
## Viscera.weight 0.011785212 0.009706296 0.003618898   0.05167359    0.022576567
## Shell.weight   0.014907877 0.012430649 0.004736391   0.06499473    0.027206160
## Rings          0.211961519 0.181194590 0.074603148   0.84611848    0.298675828
##                Viscera.weight Shell.weight       Rings
## Length            0.011785212  0.014907877  0.21196152
## Diameter          0.009706296  0.012430649  0.18119459
## Height            0.003618898  0.004736391  0.07460315
## Whole.weight      0.051673595  0.064994731  0.84611848
## Shucked.weight    0.022576567  0.027206160  0.29867583
## Viscera.weight    0.011953789  0.013796780  0.17628104
## Shell.weight      0.013796780  0.019325336  0.28015999
## Rings             0.176281040  0.280159991 10.35659040
```

Except Height all the other continuous numerous predictors have VIF >10, while variances between predictors are <0.10. These all indicate the existence of multicollinearity.

To deal with this problem we have three options: removing some of the predictors, converting correlated predictors into one, and using principle component analysis(PCA). As the first option is also a big topic and nearly each predictor is highly correlated with the other. We will choose the third option: PCA.

In the end, we reduce the levels of sex as we concluded from the last chapters. One easier way, which also support regression models in the next chapter, is reversing this variable into binary variable: Infant(1) and Not Infant(0). We rename the predictor as "Infant":

```
# reduce levels of Sex to infant(1) and not infant(0)
abalone_cleaned<-abalone_cleaned%>%mutate(Infant=ifelse(Sex=="I",1,0))%>%select(-Sex)
```

2.2. Summary of data cleaning and exploration

In summary, we did the following steps to prepare the data:

a. removed observations with Height=0.

b. removed observations with impossible weights combination.

c. removed observation with odd Height and Diameter combination.

d. found out the multicollinearity in the dataset and will solve this problem with PCA.

e. Renamed "Sex" to "Infant" and changed it to binary variable: 1(infant) and 0(adult).

Now the structure of the dataset is like the following and we are ready to fix our strategy and then start the modelling:

```
str(abalone_cleaned)
```

```
## 'data.frame':    4155 obs. of  9 variables:
##  $ Length        : num  0.455 0.35 0.53 0.44 0.33 0.425 0.53 0.545 0.475 0.55 ...
##  $ Diameter      : num  0.365 0.265 0.42 0.365 0.255 0.3 0.415 0.425 0.37 0.44 ...
##  $ Height        : num  0.095 0.09 0.135 0.125 0.08 0.095 0.15 0.125 0.125 0.15 ...
##  $ Whole.weight  : num  0.514 0.226 0.677 0.516 0.205 ...
##  $ Shucked.weight: num  0.2245 0.0995 0.2565 0.2155 0.0895 ...
```

```
##  $ Viscera.weight: num  0.101 0.0485 0.1415 0.114 0.0395 ...
##  $ Shell.weight  : num  0.15 0.07 0.21 0.155 0.055 0.12 0.33 0.26 0.165 0.32 ...
##  $ Rings         : int  15 7 9 10 7 8 20 16 9 19 ...
##  $ Infant        : num  0 0 0 0 1 1 0 0 0 0 ...
```

2.3. Insights and modelling approach

From the data exploration and cleaning step we get the insight that there is multicollinearity existing and decided to use PCA to eliminate the problem. Now we can set up the approaching strategy.

2.3.1. General strategy from insights

We will firstly separate the dataset to training and testing sets, then impliment cross validation on training to avoid over-fitting. Secondly, use PCA to eliminate the multicollinearity. Thirdly, we will implement the simple regression model(lm) and several sophisticated models(kNN, GamLoess, RandomForest) as discussed before. We will tune the parameters using train function. In this way, the optimized parameter is automatically used for prediction.

2.3.2. Modelling

This chapter includes train/test split, PCA process and implementation of models.

2.3.2.1. Train and test split

We will separate the data for training and final testing in the ratio of 80 to 20. For the splitting ratio there are the followings considered. There is no fix rules of how to split the data and it is also related with the size of dataset. Basically, with less portion of training data, the tuning variance is greater; with less portion of testing, the performance variance is greater. This dataset has around 4000 Observations, using 80 to 20 is on one hand not taking too long time of training, and on the other hand, the performance has a certain stability.

```
#split into the train(80%) and test set(20%)
index<-createDataPartition(abalone_cleaned$Rings,times = 1, p=0.2,list = FALSE)
train_set<-abalone_cleaned[-index,]
test_set<-abalone_cleaned[index,]
```

Furthermore, the cross validation with 5 folds is used instead of further splitting training dataset. The reason are the followings: the cross validation is more sophisticated method and optimizing parameters. Additionally, in some algorithms such as Random Forest, the computation time with the 80% of 4000 observations is still acceptable with 5 folds.

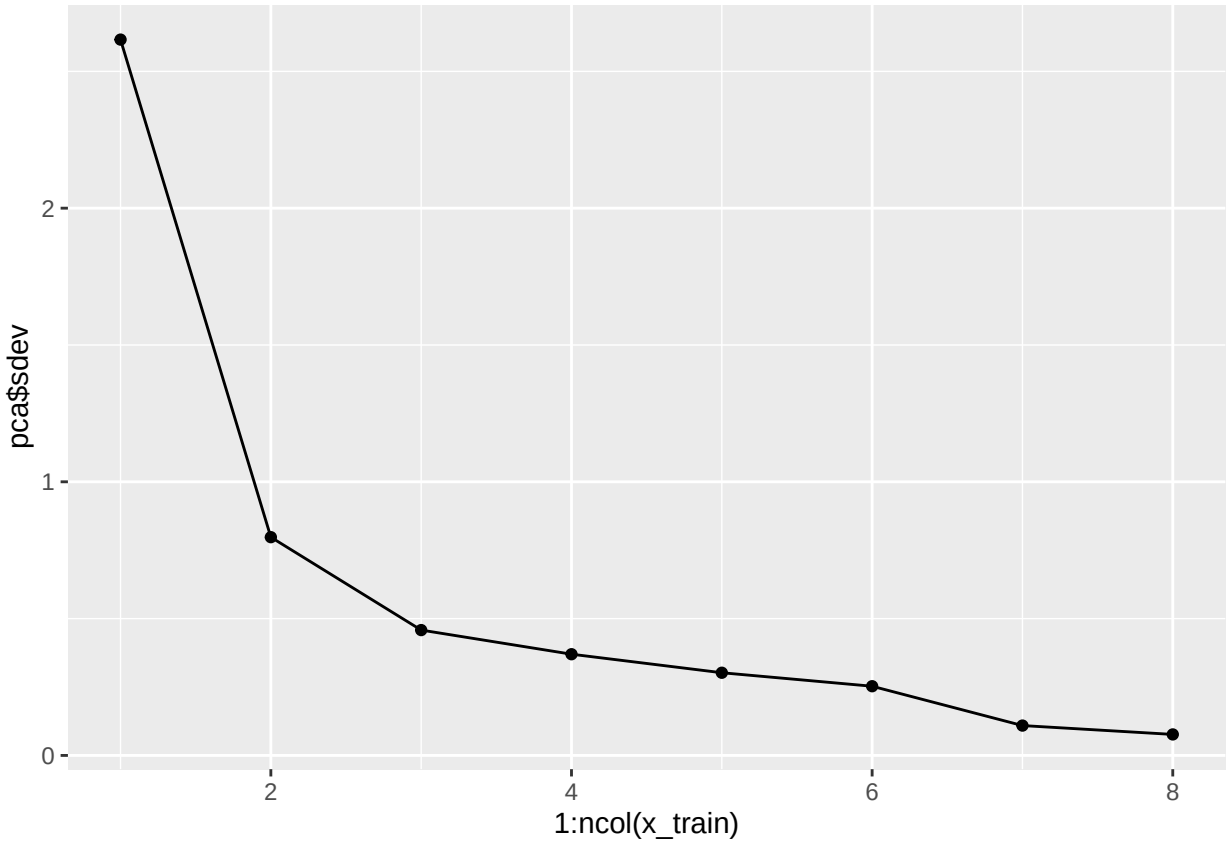
2.3.2.2. PCA on the train and test set

Let us first do the PCA on train set and choose how many principle components to use:

```
# x_train as the train set without output column Rings
x_train<-train_set[,!names(train_set)%in%"Rings"]
# pca the x_train with scaling the predictor first
pca<-prcomp(x_train,scale. = T)
##Examine how many pinciple components should be used for modelling
summary(pca)
```

```
## Importance of components:
##                           PC1     PC2     PC3     PC4     PC5    PC6     PC7
## Standard deviation     2.6161 0.79765 0.45814 0.36994 0.30222 0.2530 0.10917
## Proportion of Variance 0.8555 0.07953 0.02624 0.01711 0.01142 0.0080 0.00149
## Cumulative Proportion  0.8555 0.93501 0.96125 0.97836 0.98977 0.9978 0.99926
##                            PC8
## Standard deviation     0.07682
## Proportion of Variance 0.00074
## Cumulative Proportion  1.00000
```

```
qplot(1:ncol(x_train),pca$sdev)+geom_line()
```



The first 6 principle components are good to choose, as representing around 99% of the variance.Then further transform the test set with it.

```
#choose the representitive components
x_train<-pca$x[,1:5]
#transform the test set
##x_test as the new test set and removing output Rings
x_test<-test_set[,!names(test_set)%in%"Rings"]
##stantardize the columns of test set and keep the first 6 columns
x_test<-sweep(as.matrix(x_test),2,colMeans(x_test))%*%pca$rotation
x_test<-x_test[,1:5]
```

Now we can start to impliment the different model on the dataset and calculate the RMSE.

2.3.2.3. Naive Modelling

Use mean as predicted value to calculate a RMSE:

```
#mean value of Rings
mean<-mean(train_set$Rings)
#RMSE value using average Rings
naive_rmse<-RMSE(test_set$Rings,mean)
naive_rmse
```

```
## [1] 3.191154
```

This should be the largest RMSE value comparing using algorithmus.

2.3.2.4. Simple linear regression lm

Use the lm to get the RMSE value, there is no tuned parameter here:

```
# train on x_train, no parameter is tuned
fit_lm<-train(x_train,train_set$Rings,method="lm",
              trControl=trainControl(method="cv",number=5,p=0.9))
## make the predition on x_test
pred_lm<-predict(fit_lm,x_test)
## calculate the RMSE
lm_rmse<-RMSE(test_set$Rings,pred_lm)
lm_rmse
```

```
## [1] 3.031933
```

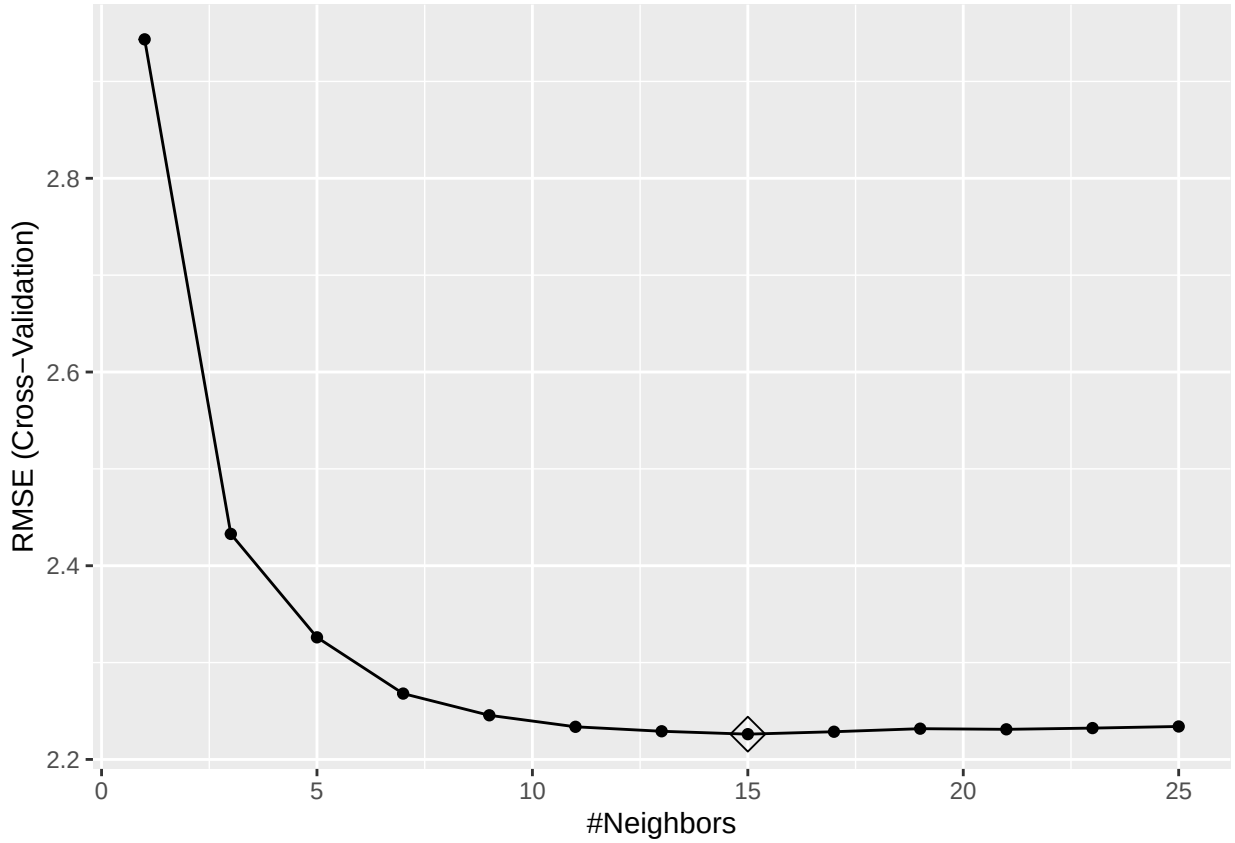
2.3.2.5. k-Nearest Neighbor(kNN)

We set the tuning of k between 1 and 30:

```
#train with kNN and tune the k value between 1 and 15:
fit_knn<-train(x_train,train_set$Rings,method="knn", tuneGrid=data.frame(k=seq(1,25,2)),
               trControl=trainControl(method="cv",number=5,p=0.9))
##show the optimized k
fit_knn$bestTune
```

```
##    k
## 8 15
```

```
ggplot(fit_knn,highlight = TRUE)
```




```
##make the prediction on test set
pred_knn<-predict(fit_knn,x_test)
## calculate the RMSE
knn_rmse<-RMSE(test_set$Rings,pred_knn)
knn_rmse
```

```
## [1] 2.94968
```

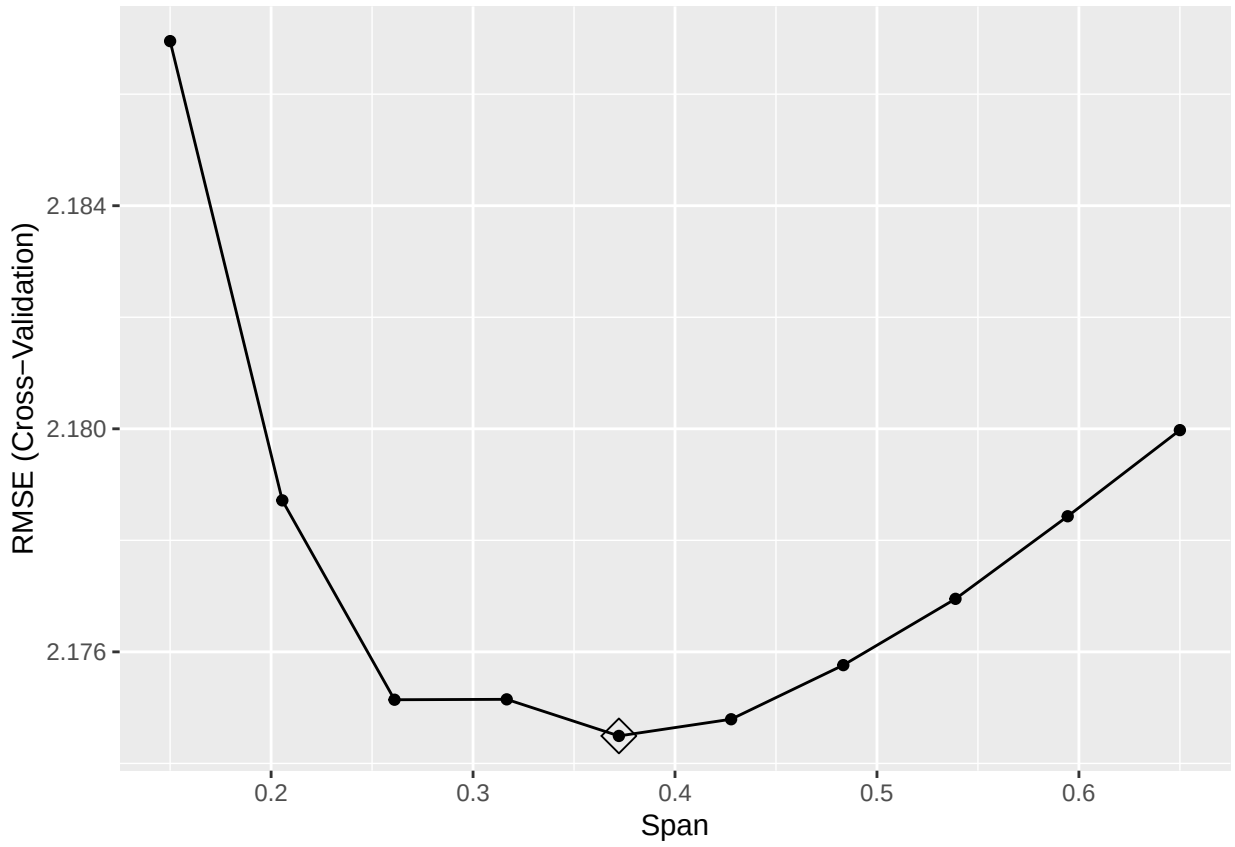
2.3.2.6. GamLoess

GamLoess is a method using a kernel. We train with tuning the parameter "span":

```
#train with gamLoess with tuning the span
fit_gam<-train(x_train,train_set$Rings,method="gamLoess",
               tuneGrid=expand.grid(span=seq(0.15,0.65,len=10),degree=1),
               trControl=trainControl(method="cv",number=5,p=0.9))
## with the optimized span value
fit_gam$bestTune
```

```
##        span degree
## 5 0.3722222      1
```

```
ggplot(fit_gam,highlight = TRUE)
```



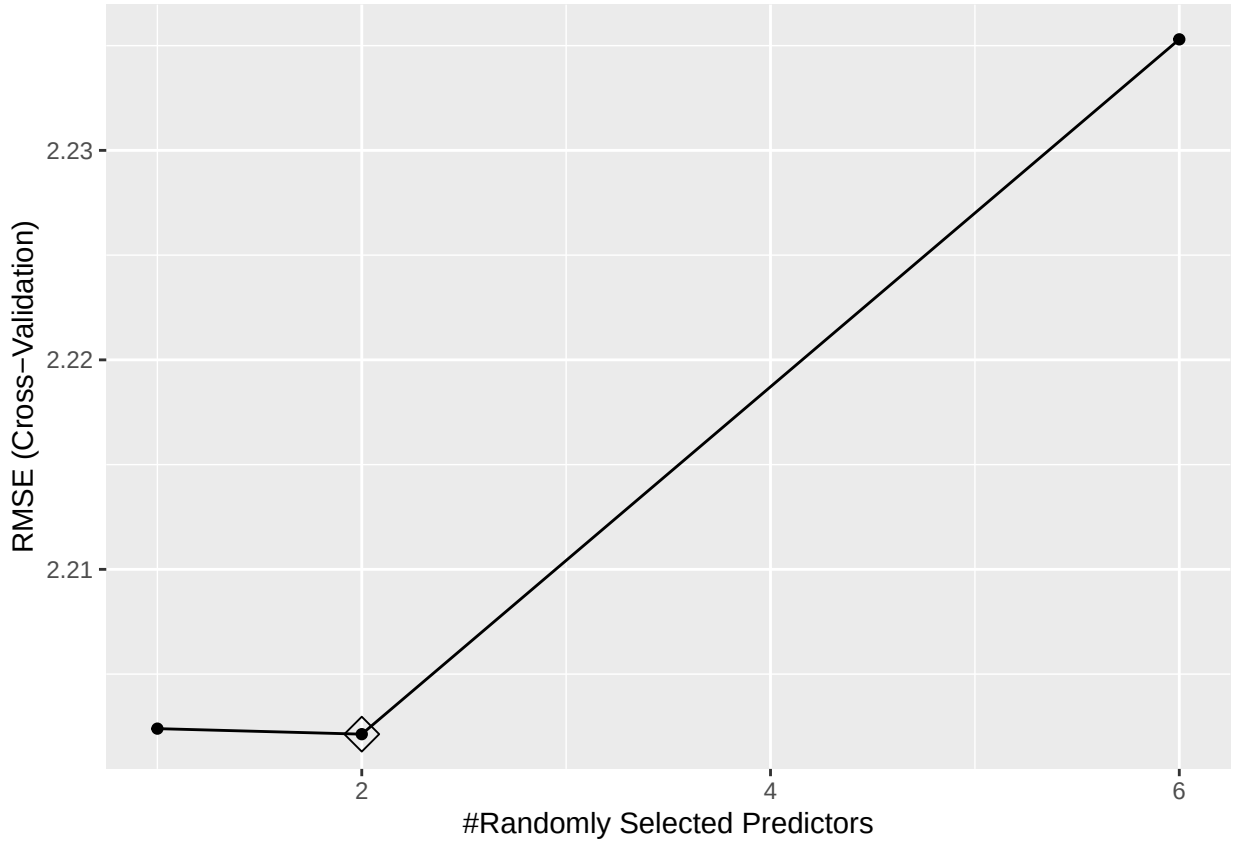

```
## predict on test set
pred_gam<-predict(fit_gam,x_test)
## calculate the RMSE
gam_rmse<-RMSE(test_set$Rings,pred_gam)
gam_rmse
```

```
## [1] 3.027072
```

2.3.2.7. RandomForest

Train with RandomForest with tuning of mtry. It might takes several minutes.

```
#train with random forest model with tuning the parameter mtry from 1 to 6
fit_rf<-train(x_train,train_set$Rings,method="rf",importance=TRUE,
              tuneGrid=data.frame(mtry=c(1,6,2)),trControl=trainControl(method="cv",number=5,p=0.9))
##the optimized mtry with plot and value
ggplot(fit_rf,highlight = TRUE)
```




```
fit_rf$bestTune
```

```
##   mtry
## 2    2
```

```
##predict on test set
pred_rf<-predict(fit_rf,x_test)
#RMSE value with the optimized parameter
rf_rmse<-RMSE(test_set$Rings,pred_rf)
rf_rmse
```

```
## [1] 3.160379
```

3. Results

In this chapter, we will list and compare the RMSE result from last chapter. Based on the lowest value, we will define the best model.

```
# Listing the RMSE result of all model
rmse_result<-t(tibble(naive_rmse,lm_rmse,knn_rmse,gam_rmse,rf_rmse))
rmse_result
```

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

```
## naive_rmse 3.191154
## lm_rmse    3.031933
## knn_rmse   2.949680
## gam_rmse   3.027072
## rf_rmse    3.160379
```

The final model is using the kNN model with the following k value and the RMSE:

```
## with optimized k value
fit_knn$bestTune
```

```
##    k
## 8 15
```

```
# final RMSE
knn_rmse
```

```
## [1] 2.94968
```

The knn_rmse value indicates the performance of the model: in which distance(in average) the predicted rings is away from the real value of rings.

4. Conclusion

Going through the whole project, the most time consuming part of this project is data exploration and cleaning. In this part, we detected the outliers and odd values and furthermore, the multicollinearity issue. Therefore, the PCA is implemented on the predictors. For the modelling we implemented not only the simple linear regression but also some complexer models. Furthermore, cross validation is used to avoid over-fitting and tuning of parameter to increase the accuracy of prediction.

However, the main challenge of the project is to confirm the multicollinearity in dataset and how to solve it. There are different parameters other than correlation value need to be considered before confirming the multicollinearity, such as VIF and variable variance. There are also more possibilities to solve it. Which one is the most suitable for this dataset is a lot to research. For this dataset, we chose PCA method. However, other method such as combine two variables into one using z-score, or just remove some predictors, could also be an option. This can be the future work to implement those different methods and to compare the result.

Another point for future work is to go deeper in the physical and ecnomical meaning of the data. The predictors provided in the dataset include different physical characteristics. To measure some of them(such as viscera weight, shell weight, etc), we need to kill the abalone first. It might be meaningful to create a dataset containing only predictors(such as diameter, whole weight, etc.), which can be measured from alive abalone.

References

1. https://rafalab.github.io/dsbook/
2. https://www.theanalysisfactor.com/eight-ways-to-detect-multicollinearity/
3. https://www.analyticsvidhya.com/blog/2016/03/pca-practical-guide-principal-component-analysis-python/