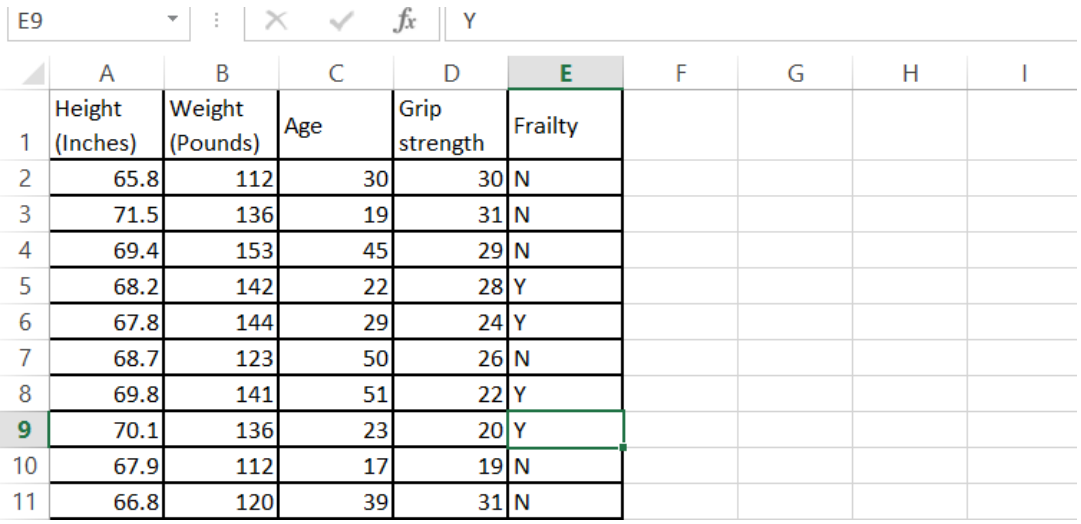
PDS Assignment

Name : Atluri Chanikya

Student Id: 16325264

Works in Stage-1 (Data Collection):

All project files are often compiled into a single directory, which is then further divided into subdirectories for data, source code, analytical results, etc.



E9 | Y

	A	B	C	D	E	F	G	H	I
1	Height (Inches)	Weight (Pounds)	Age	Grip strength	Frailty				
2	65.8	112	30	30	N				
3	71.5	136	19	31	N				
4	69.4	153	45	29	N				
5	68.2	142	22	28	Y				
6	67.8	144	29	24	Y				
7	68.7	123	50	26	N				
8	69.8	141	51	22	Y				
9	70.1	136	23	20	Y				
10	67.9	112	17	19	N				
11	66.8	120	39	31	N				

Folder Structure:

| - - Frailty_Project

| | - - raw_data

| | | - - dataset.csv

| | | - - README.txt

| | - - clean_data

| | - - results

| | - - src

Works in stage-2 (Data processing):

We can easily develop a small script that will read the raw table, eliminate the rows with NA yields and those with a field code of N, and save the resulting processed data.

Folder Structure:

| - - Frailty_Project

| | - - raw_data

| | | - - raw_dataset.csv

| | | - - README.txt

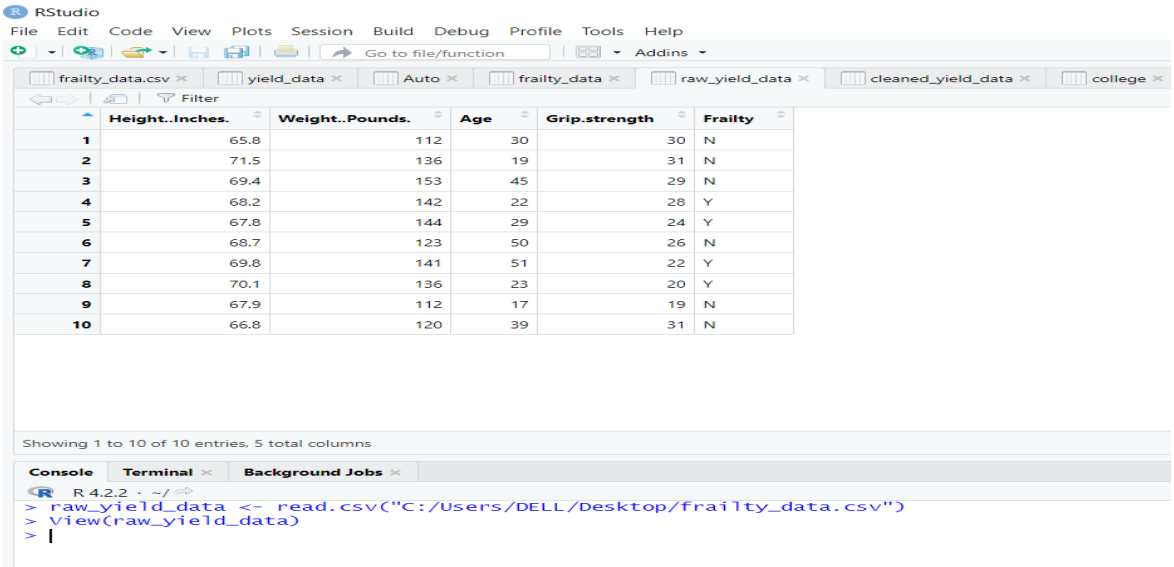
| |- - clean_data

| | | - - cleaned_data.csv

| | - - results

| | - - src

| | | - - clean_data.R



RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Go to file/function | Addins

frailty_data.csv × | yield_data × | Auto × | frailty_data × | raw_yield_data × | cleaned_yield_data ×

Filter

	Height..Inches.	Weight..Pounds.	Age	Grip.strength	Frailty
4	68.2	142	22	28	Y
5	67.8	144	29	24	Y
7	69.8	141	51	22	Y
8	70.1	136	23	20	Y

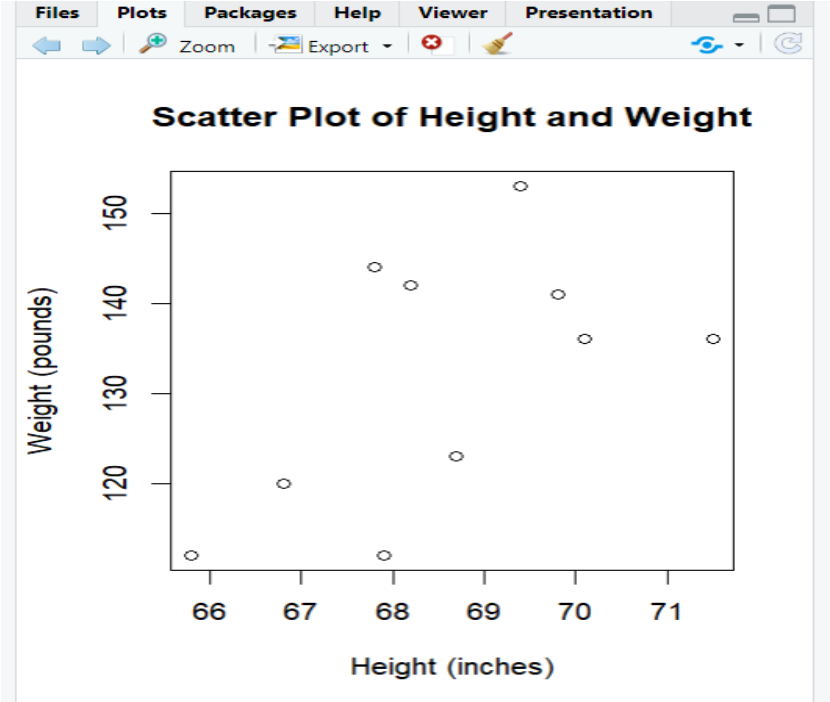
Showing 1 to 4 of 4 entries, 5 total columns

Console | Terminal × | Background Jobs ×

R 4.2.2 · ~/

```
> raw_yield_data <- read.csv("C:/Users/DELL/Desktop/frailty_data.csv")
> View(raw_yield_data)
> cleaned_yield_data <- na.omit(raw_yield_data[raw_yield_data$Frailty !="N", ])
> print(cleaned_yield_data)
  Height..Inches. Weight..Pounds. Age Grip.strength Frailty
4            68.2             142  22            28       Y
5            67.8             144  29            24       Y
7            69.8             141  51            22       Y
8            70.1             136  23            20       Y
> View(cleaned_yield_data)
```

```
> plot(raw_yield_data$Height..Inches., raw_yield_data$Weight..Pounds.,
+      xlab = "Height (inches)", ylab = "Weight (pounds)",
+      main = "Scatter Plot of Height and Weight")
```



There are no missing values after visualization (scatter plot), therefore we may utilize raw data as input to train a prediction model directly.

Works in stage-3 (Data Analysis):

To predict frailty, we fitted several models (logistic regression, support vector machine, and decision tree) to cleaned and preprocessed data. We divided the data into training and testing sets, fitted the models to the training set, then predicted on the testing set. The models' performance was then tested using confusion matrices.

Folder Structure:

| - - Frailty_Project

| | - - raw_data

| | | - - raw_dataset.csv

| | | - - README.txt

| | - - clean_data

| | | - - cleaned_data.csv

| | - - results

| | | - - test_results.txt

| | - - src

| | | - - analysis.R

| | | - - clean_data.R

R Snippet:

```
# Load necessary libraries

library(caret)

# Load data

raw_yield_data <- read.csv("C:/Users/DELL/Desktop/frailty_data.csv")

# Remove rows with missing values

cleaned_yield_data <- na.omit(raw_yield_data)

# Convert Frailty column to a factor

cleaned_yield_data$Frailty <- as.factor(cleaned_yield_data$Frailty)

# Split data into training and testing sets

set.seed(123)

trainIndex <- createDataPartition(cleaned_yield_data$Frailty, p = .7, list = FALSE)

train <- cleaned_yield_data[trainIndex, ]

test <- cleaned_yield_data[-trainIndex, ]

# Fit logistic regression model

lr_model <- train(Frailty ~ ., data = train, method = "glm", family = "binomial")


# Fit support vector machine model

svm_model <- train(Frailty ~ ., data = train, method = "svmRadial")


# Fit decision tree model
```

dt_model <- train(Frailty ~ ., data = train, method = "rpart")

 # Make predictions on test set

lr_pred <- predict(lr_model, newdata = test)

svm_pred <- predict(svm_model, newdata = test)

dt_pred <- predict(dt_model, newdata = test)

 # Evaluate performance of models

confusionMatrix(lr_pred, test$Frailty)

confusionMatrix(svm_pred, test$Frailty)

confusionMatrix(dt_pred, test$Frailty)

Results:

```
Confusion Matrix and Statistics

          Reference
Prediction N Y
         N 1 0
         Y 0 1

               Accuracy : 1
                 95% CI : (0.1581, 1)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : 0.25

                  Kappa : 1

 Mcnemar's Test P-Value : NA

            Sensitivity : 1.0
            Specificity : 1.0
         Pos Pred Value : 1.0
         Neg Pred Value : 1.0
             Prevalence : 0.5
         Detection Rate : 0.5
   Detection Prevalence : 0.5
      Balanced Accuracy : 1.0

       'Positive' Class : N

> confusionMatrix(svm_pred, test$Frailty)
Confusion Matrix and Statistics

          Reference
Prediction N Y
         N 1 1
```

```
         Y 0 0

               Accuracy : 0.5
                 95% CI : (0.0126, 0.9874)
    No Information Rate
    : 0.5 P-Value [Acc >
    NIR] : 0.75

                  Kappa

 : 0 Mcnemar's Test P-

 Value : 1.00

       Sensitivity : 1.0
             Specificity
           : 0.0Pos Pred
          Value : 0.5Neg
            Pred Value :
           NaNPrevalence
                    : 0.5
          Detection Rate
                    : 0.5
   Detection Prevalence
          : 1.0Balanced
         Accuracy : 0.5

       'Positive' Class : N


> confusionMatrix(dt_pred,
test$Frailty)Confusion Matrix and
Statistics

          Refe
rence
Prediction N Y
         N 1 1
         Y 0 0

               Accuracy : 0.5
                 95% CI : (0.0126, 0.9874)
    No Information Rate
    : 0.5 P-Value [Acc >
    NIR] : 0.75
```

```
                 Kappa

: 0 Mcnemar's Test P-

Value : 1.00


      Sensitivity : 1.0

            Specificity
          : 0.0Pos Pred
         Value : 0.5Neg
           Pred Value :
          NaNPrevalence
                  : 0.5
         Detection Rate
                  : 0.5

   Detection Prevalence
          : 1.0Balanced
         Accuracy : 0.5


      'Positive' Class : N
```