Chapter 19: Decision Tree

Exercise 2: Baseball

Yêu cầu: Áp dụng Decision Tree để dự đoán cân nặng dựa trên chiều cao

- Cho dữ liệu baseball_2D.txt
- Tái sử dụng bài Baseball ở phần Linear Regression, sau đó áp dụng Decision Tree để dự đoán cân nặng dựa trên chiều cao
- Cho chiều cao lần lượt là c(1.775, 1.825, 1.925), hỏi cân nặng bằng bao nhiêu?
- Chú ý: khi build model Decsion Tree cho Regression phải sử dụng method là “anova”, khi predict thì chọn type là “vector”

In [1]:
```
# predict weight based on height
# open and read csv file
library(rpart)
data <- read.csv("baseball.csv")
print(head(data))
print(is.data.frame(data))
print(ncol(data))
print(nrow(data))
```

```
             Name Team       Position Height Weight   Age PosCategory
1   Adam_Donachie  BAL        Catcher     74    180 22.99     Catcher
2      Paul_Bako  BAL        Catcher     74    215 34.69     Catcher
3 Ramon_Hernandez  BAL        Catcher     72    210 30.78     Catcher
4    Kevin_Millar  BAL  First_Baseman     72    210 35.43   Infielder
5     Chris_Gomez  BAL  First_Baseman     73    188 35.71   Infielder
6   Brian_Roberts  BAL Second_Baseman     69    176 29.39   Infielder
[1] TRUE
[1] 7
[1] 1015
```

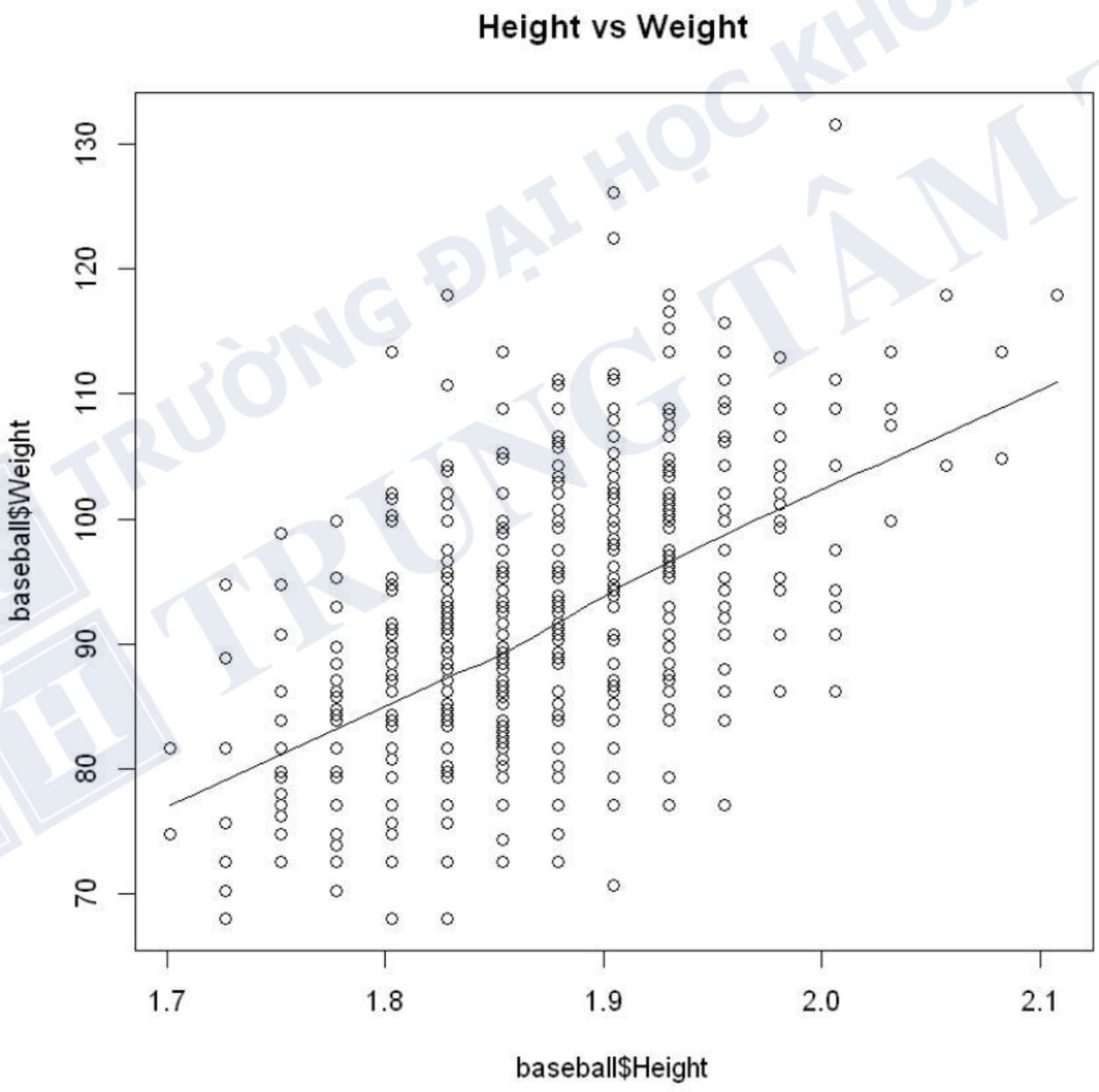
In [2]:
```
baseball <- data[c("Height", "Weight")]
print(head(baseball))
```

```
  Height Weight
1     74    180
2     74    215
3     72    210
4     72    210
5     73    188
6     69    176
```

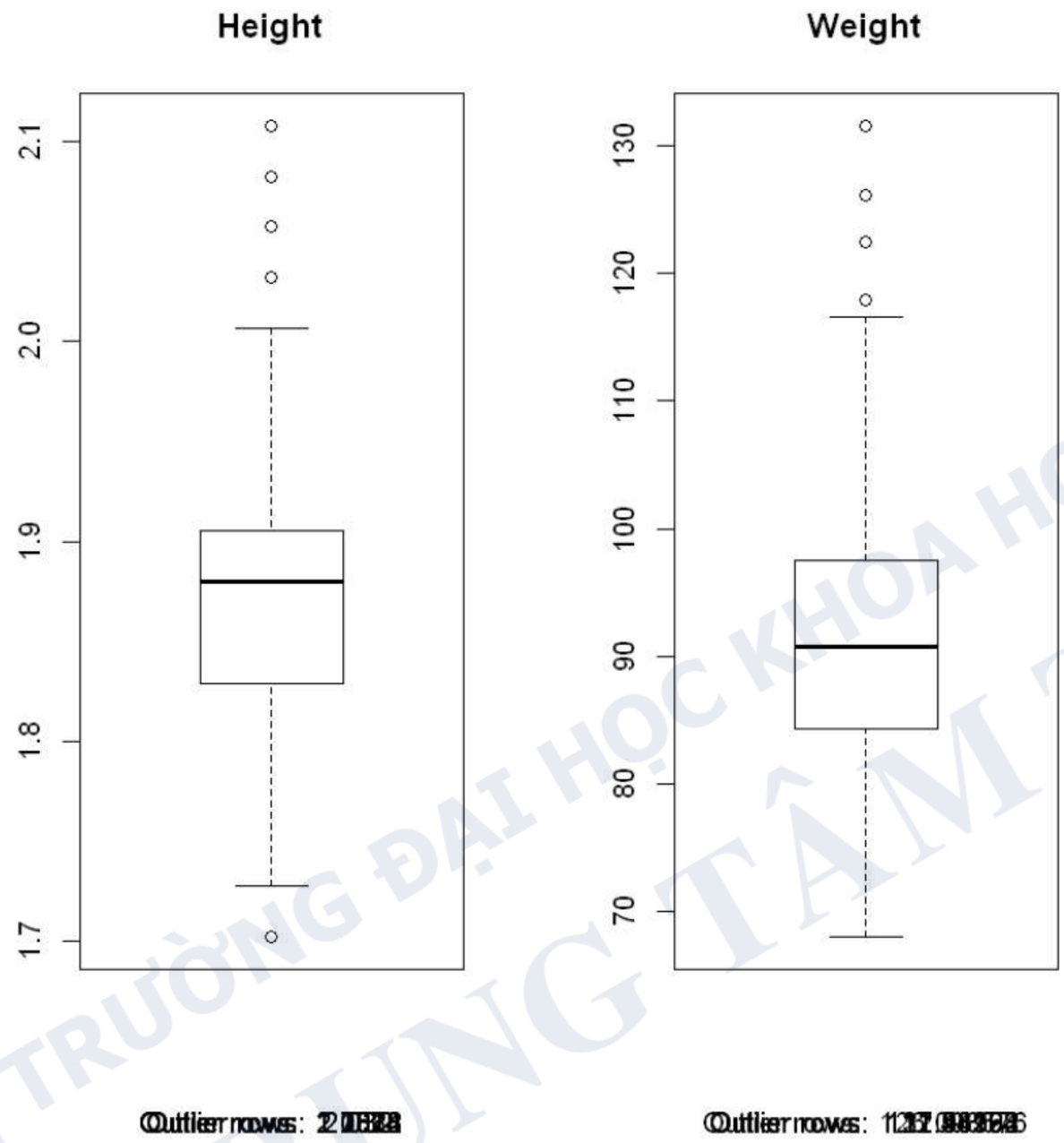
In [3]:
```
baseball["Height"] <- baseball["Height"] * 0.0254
baseball["Weight"] <- baseball["Weight"] * 0.453592

print("After changing data:")
print(head(baseball))
```

```
[1] "After changing data:"
  Height   Weight
1 1.8796 81.64656
2 1.8796 97.52228
3 1.8288 95.25432
4 1.8288 95.25432
5 1.8542 85.27530
6 1.7526 79.83219
```

In [4]:
```
scatter.smooth(x=baseball$Height,
               y=baseball$Weight,
               main="Height vs Weight")
```



In [5]:
```
# BoxPlot to Check for outliers
par(mfrow=c(1, 2))  # divide graph area in 2 columns
boxplot(baseball$Height, main="Height",
        sub=paste("Outlier rows: ",
                  boxplot.stats(baseball$Height)$out))
boxplot(baseball$Weight, main="Weight",
        sub=paste("Outlier rows: ",
                  boxplot.stats(baseball$Weight)$out))
```



In [6]:
```
# calculate correlation between Width and Length
print(cor(baseball$Height, baseball$Weight))
```

```
[1] 0.5315393
```

In [7]:
```
wt_outliers <- c(boxplot.stats(baseball$Weight)$out)
print("wt_outliers: ")
print(wt_outliers)

ht_outliers <- c(boxplot.stats(baseball$Height)$out)
print("ht_outliers: ")
print(ht_outliers)

#drop rows have outliers
print(paste("Before drop:", nrow(baseball)))
for (record in wt_outliers){
  baseball <- baseball[baseball$Weight != record,]
}
for (record in ht_outliers)
{
  baseball <- baseball[baseball$Height != record,]
}
print(paste("After drop:", nrow(baseball)))
```

```
[1] "wt_outliers: "
[1] 117.9339 122.4698 131.5417 126.0986 117.9339 117.9339 117.9339
[1] "ht_outliers: "
 [1] 2.0574 2.0320 2.0320 2.0320 2.0320 2.0828 2.0320 2.0574 2.0828 2.1082
[11] 1.7018 1.7018
[1] "Before drop: 1015"
[1] "After drop: 998"
```

In [8]:
```
# Create the training (development) and test (validation) data.
set.seed(42)  # setting seed to reproduce results of random sampling
trainingRowIndex <- sample(1:nrow(baseball), 0.7*nrow(baseball))
print("Selected training row indexes:")
# print(trainingRowIndex)
trainingData <- baseball[trainingRowIndex, ] # training data
testData  <- baseball[-trainingRowIndex, ]   # test data
print("Rows of training data and test data:")
print(nrow(trainingData))
print(nrow(testData))
```

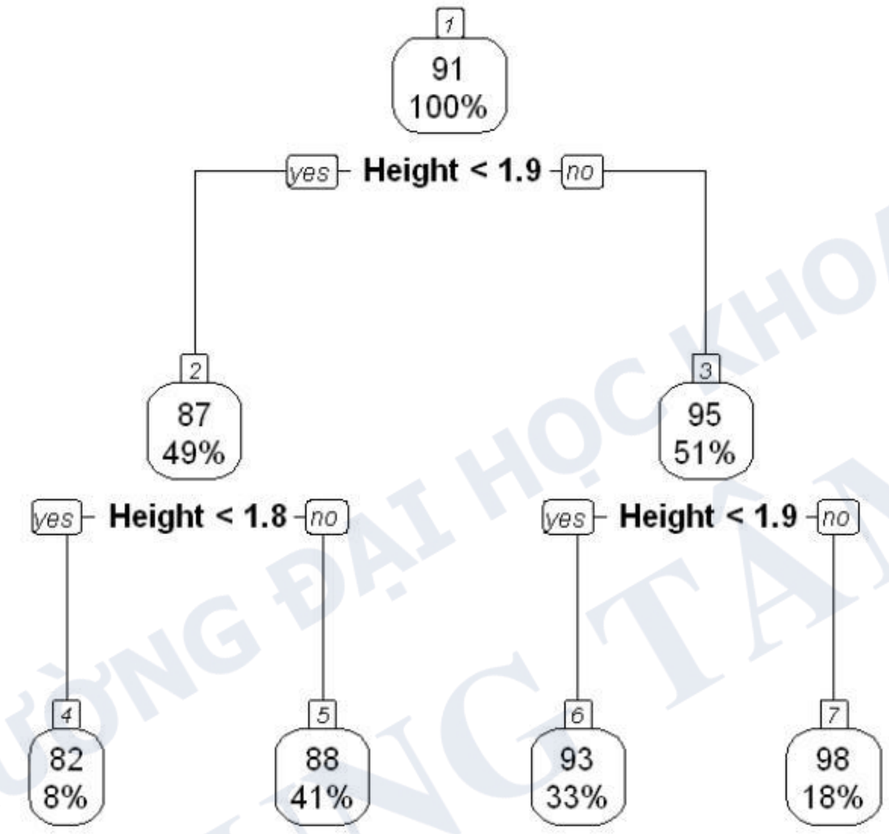
```
[1] "Selected training row indexes:"
[1] "Rows of training data and test data:"
[1] 698
[1] 300
```

In [9]:
```
# Use Decision Tree
# Build model
baseball.tree <- rpart(Weight~Height,
                       data = trainingData,
                           method="anova")
print(summary(baseball.tree))
```

```
Call:
rpart(formula = Weight ~ Height, data = trainingData, method = "anova")
  n= 698

          CP nsplit rel error    xerror       xstd
1 0.18727791      0 1.0000000 1.0015896 0.04984781
2 0.03704596      1 0.8127221 0.8408814 0.04377180
3 0.03223072      2 0.7756761 0.8136477 0.04243592
4 0.01000000      3 0.7434454 0.7693579 0.04083492

Variable importance
Height
   100

Node number 1: 698 observations,    complexity param=0.1872779
  mean=91.09076, MSE=83.66086
  left son=2 (339 obs) right son=3 (359 obs)
  Primary splits:
      Height < 1.8669 to the left,  improve=0.1872779, (0 missing)

Node number 2: 339 observations,    complexity param=0.03223072
  mean=87.01741, MSE=60.56898
  left son=4 (53 obs) right son=5 (286 obs)
  Primary splits:
      Height < 1.7907 to the left,  improve=0.09166379, (0 missing)

Node number 3: 359 observations,    complexity param=0.03704596
  mean=94.93718, MSE=75.00348
  left son=6 (230 obs) right son=7 (129 obs)
  Primary splits:
      Height < 1.9177 to the left,  improve=0.08034201, (0 missing)

Node number 4: 53 observations
  mean=81.54386, MSE=39.88859

Node number 5: 286 observations
  mean=88.03174, MSE=57.82052

Node number 6: 230 observations
  mean=93.09877, MSE=65.82509

Node number 7: 129 observations
  mean=98.21497, MSE=74.59822

n= 698

node), split, n, deviance, yval
      * denotes terminal node

1) root 698 58395.280 91.09076
```

```
2) Height< 1.8669 339 20532.890 87.01741
  4) Height< 1.7907 53  2114.095 81.54386 *
  5) Height>=1.7907 286 16536.670 88.03174 *
3) Height>=1.8669 359 26926.250 94.93718
  6) Height< 1.9177 230 15139.770 93.09877 *
  7) Height>=1.9177 129  9623.170 98.21497 *
```

In [10]:
```
# draw tree
library(rpart.plot)
prp(baseball.tree, type=2, extra="auto",
    nn = TRUE, branch=1, varlen=0, yesno=2)
```



In [11]:
```
#test model
pred_new = predict(baseball.tree, testData, type = "vector")

# mean square error of testData
mse_test = mean((testData$Weight - pred_new)^2)
print(paste("mse in test: ", mse_test))
```

```
[1] "mse in test:  59.6827254267076"
```

In [12]:
```
# new predictions
x <- c(1.775, 1.825, 1.925)
y1 <- predict(baseball.tree, data.frame(Height = x), type = "vector")
print("Solution 2 - results:")
print(y1)
```

```
[1] "Solution 2 - results:"
       1        2        3 
81.54386 88.03174 98.21497 
```