Devon Zuegel // 14 Oct 2015 // STATS 202

Problem Set 3

web.stanford.edu/class/stats202/content/viewhw.html?hw3

Problem 1

Chapter 4, Exercise 4 (Sec. 4.7, p. 168)

Part A

- `X` is uniformly distributed on `[0, 1]`.
- When predicting a test observation's response, we look the 10% of the range closest to that observation.
    - If our test observation has value `X = 0.6`, we look at `[0.55, 0.65]`.
    - If our test observation has value `X = 0.02`, we look at `[0.00, 0.10]`.
    - If our test observation has value `X = 0.98`, we look at `[0.90, 1.00]`.

Since at any give point we're looking at 10% of the range and the points are evenly distributed along that range, we'd expect to be looking at 10% of the data on average each time.

Part B

- `X1` is uniformly distributed on `[0, 1]`, and `X2` is uniformly distributed on `[0, 1]`.
- Similar rules as in part a.

Since at any give point we're looking at 10% of `X1`'s range and 10% of `X2`'s range and the points are evenly distributed along those two ranges, we'd expect to be looking at 1% of the data on average each time.

We can think about it as a square:

```
   0   1   2   3   4   5   6   7   8   9
0  - | - | - | - | - | - | - | - | - | -
1  - | - | - | - | - | - | - | - | - | -
2  - | - | - | - | - | - | - | - | - | -
3  - | - | - | - | - | X | - | - | - | -
4  - | - | - | - | - | - | - | - | - | -
5  - | - | - | - | - | - | - | - | - | -
6  - | - | - | - | - | - | - | - | - | -
```

```
  7  - | - | - | - | - | - | - | - | - | -
  8  - | - | - | - | - | - | - | - | - | -
  9  - | - | - | - | - | - | - | - | - | -
```

If we look at just 10% of the x-dimension (let's say the `5`th column) and then also just 10% of the y-dimension (let's say the `3`rd row), we get `1/100 = 1%` of the available cells.

Part C

If we have 100 features (a.k.a. 100 dimensions) and we look at just 10% of the range for each of them, we look at just a tiny portion (10^{-100}) of the data.

Part D

Let's say we have 1 billion (10^9) training observations. That's a lot of data! However, consider trying to predict the response for some test observation `m` with 100 features, where we look at just the observations that fall within 10% of each range from `m`. Of the 1 billion points we started out with, we'd expect to have $10^9 \cdot 10^{-100} = 10^{-91}$ observations to look at. That is still effectively 0, which doesn't help us at all.

Part E

The expected length of the hypercube is:

- hypercube's length is $(\frac{1}{10})^1$ `= 10%` when `p = 1`
- hypercube's length is $(\frac{1}{10})^2$ `= 1%` when `p = 2`
- hypercube's length is $(\frac{1}{10})^{100}$ when `p = 100`

Problem 2

Chapter 4, Exercise 6 (Sec. 4.7, p. 170).

$$p(X) = \frac{e^{\beta_0+\beta_1 X_1+\cdots+\beta_p X_p}}{1+e^{\beta_0+\beta_1 X_1+\cdots+\beta_p X_p}}.$$

Part A

X1 = hours studied, X2 = undergrad GPA, and Y = receive an A. We fit a logistic regression and produce estimated coefficient, β̂0 = −6, β̂1 = 0.05, β̂2 = 1.

$$
\begin{aligned}
x &= \text{“3.5 GPA \&\& studies for 40h”} \\
Pr(x) = 0.5 &= \frac{e^{\beta_0+\beta_1 X_1+\beta_2 X_2}}{1+e^{\beta_0+\beta_1 X_1+\beta_2 X_2}} \\
&= \frac{e^{-6+40\cdot 0.05+3.5\cdot 1.0}}{1+e^{-6+40\cdot 0.05+3.5\cdot 1.0}} \\
&= \boxed{0.377541}
\end{aligned}
$$

Part B

$$
\begin{aligned}
x &= \text{“3.5 GPA \&\& studies for 40h”} \\
Pr(x) = 0.5 &= \frac{e^{\beta_0+\beta_1 X_1+\beta_2 X_2}}{1+e^{\beta_0+\beta_1 X_1+\beta_2 X_2}} \\
&= \frac{e^{-6+0.05\cdot \texttt{num\_hours}+3.5\cdot 1.0}}{1+e^{-6+0.05\cdot \texttt{num\_hours}+3.5\cdot 1.0}} \\
&= \boxed{50\text{ hours}}
\end{aligned}
$$

Input interpretation:

solve	$0.5 = \dfrac{e^{-6+0.05\,x+3.5}}{1+e^{-6+0.05\,x+3.5}}$	for	x

Result:

$x = 50 + (125.664\, i)\, n$ and $n \in \mathbb{Z}$

Problem 3

Chapter 4, Exercise 8 (Sec. 4.7, p. 170).

We prefer to use the logistic regression, despite the fact that the 1-nearest neighbors method gives us a lower average error rate than the logistic regression (`18%` vs $\frac{20+30}{2}$ `= 25%`). However, since KNN with `k = 1` gives us a training error rate of `0%` , we know that its tests error rate must be `36%` (since $\frac{0+x}{2} = 18$).

Problem 4

Chapter 4, Exercise 10 (Sec. 4.7, p. 171). In part (i), please be concise; only describe and provide the output of your best prediction.

```
##               Year        Lag1        Lag2        Lag3        Lag4
## Year    1.00000000 -0.032289274 -0.03339001 -0.03000649 -0.031127923
## Lag1   -0.03228927  1.000000000 -0.07485305  0.05863568 -0.071273876
## Lag2   -0.03339001 -0.074853051  1.00000000 -0.07572091  0.058381535
## Lag3   -0.03000649  0.058635682 -0.07572091  1.00000000 -0.075395865
## Lag4   -0.03112792 -0.071273876  0.05838153 -0.07539587  1.000000000
## Lag5   -0.03051910 -0.008183096 -0.07249948  0.06065717 -0.075675027
## Volume  0.84194162 -0.064951313 -0.08551314 -0.06928771 -0.061074617
## Today  -0.03245989 -0.075031842  0.05916672 -0.07124364 -0.007825873
##                Lag5      Volume        Today
## Year   -0.030519101  0.84194162 -0.032459894
## Lag1   -0.008183096 -0.06495131 -0.075031842
## Lag2   -0.072499482 -0.08551314  0.059166717
## Lag3    0.060657175 -0.06928771 -0.071243639
## Lag4   -0.075675027 -0.06107462 -0.007825873
## Lag5    1.000000000 -0.05851741  0.011012698
## Volume -0.058517414  1.00000000 -0.033077783
## Today   0.011012698 -0.03307778  1.000000000
```

Part B

Only the `Lag2` predictor appears statistically significant.

```
## (Intercept)        Lag1        Lag2        Lag3        Lag4        Lag5
## 0.001898848 0.118144368 0.029601361 0.546923890 0.293653342 0.583348244
##      Volume
## 0.537674762
```

Part C

```
attach(Weekly)
probs = predict(fit, type = 'response')
pred = rep('Down', nrow(Weekly))
pred[probs > .5] = 'Up'

table(pred, Direction)
```

```
##       Direction
## pred   Down  Up
##   Down   54  48
##   Up    430 557
```

```
mean(pred==Direction)  # Fraction of correct predictions
```

```
## [1] 0.5610652
```

This tells us that we are making the correct prediction about 56% of the time. In particular, we often wrongly predict “Up” when we should have predicted “Down”.

Part D

```
train=(Year<=2008)
Weekly.2009and10 = Weekly[!train,]
dim(Weekly.2009and10)  #  0  9
```

```
## [1] 104   9
```

```
Direction.2009and10 = Direction[!train]

fit = glm(Direction ~ Lag2, data = Weekly, family = binomial, subset = train)
probs = predict(fit, Weekly.2009and10, type  = 'response')

pred = rep('Down', nrow(Weekly.2009and10))
pred[probs > .5] = 'Up'
table(pred, Direction.2009and10)
```

```
##       Direction.2009and10
## pred   Down Up
##   Down    9  5
##   Up     34 56
```

```
mean(pred==Direction.2009and10)
```

```
## [1] 0.625
```

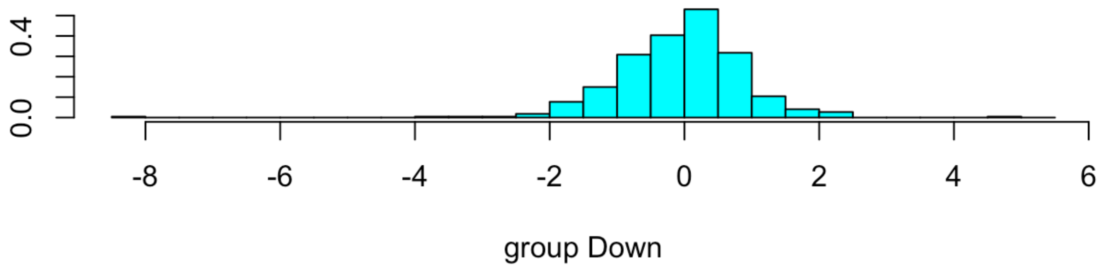
Using just `Lag2` gives us a better result of `62.5%`.

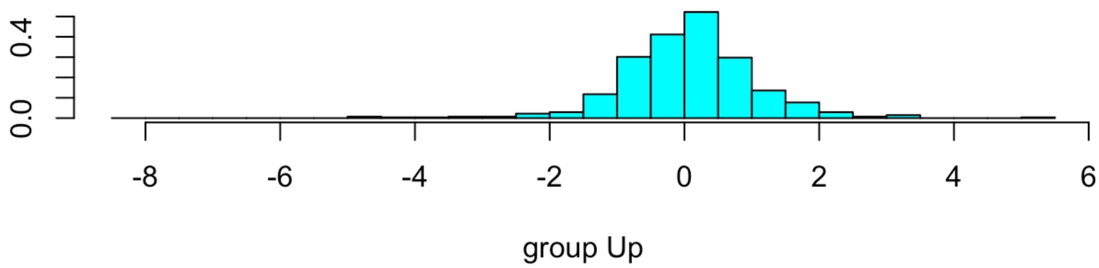
Part E

```
library(MASS)
lda.fit = lda(Direction ~ Lag2, data = Weekly, subset = train)
lda.fit
```

```
## Call:
## lda(Direction ~ Lag2, data = Weekly, subset = train)
##
## Prior probabilities of groups:
##      Down        Up
## 0.4477157 0.5522843
##
## Group means:
##             Lag2
## Down -0.03568254
## Up    0.26036581
##
## Coefficients of linear discriminants:
##            LD1
## Lag2 0.4414162
```

```
plot(lda.fit)
```






```
lda.pred = predict(lda.fit, Weekly.2009and10)
names(lda.pred)
```

```
## [1] "class"     "posterior" "x"
```

```
lda.class = lda.pred$class
table(lda.class, Direction.2009and10)
```

```
##          Direction.2009and10
## lda.class Down Up
##      Down    9  5
##      Up     34 56
```

```
mean(lda.class == Direction.2009and10)
```

```
## [1] 0.625
```

```
sum(lda.pred$posterior[,1] >= .5)
```

```
## [1] 14
```

```
sum(lda.pred$posterior[,1] < .5)
```

```
## [1] 90
```

Part F

```
qda.fit = qda(Direction ~ Lag2, data = Weekly, subset = train)
qda.fit
```

```
## Call:
## qda(Direction ~ Lag2, data = Weekly, subset = train)
##
## Prior probabilities of groups:
##      Down        Up
## 0.4477157 0.5522843
##
## Group means:
##              Lag2
## Down -0.03568254
## Up    0.26036581
```

```
qda.class = predict(qda.fit, Weekly.2009and10)$class
table(qda.class, Direction.2009and10)
```

```
##          Direction.2009and10
## qda.class Down Up
##      Down    0  0
##      Up     43 61
```

```
mean(qda.class == Direction.2009and10)
```

```
## [1] 0.5865385
```

Part G

```
library(class)
train.X = as.matrix(Lag2[train])
test.X  = as.matrix(Lag2[!train])
train.Direction = Direction[train]

set.seed(1)
knn.pred = knn(train.X, test.X, train.Direction, k = 1)
table(knn.pred, Direction.2009and10)
```

```
##         Direction.2009and10
## knn.pred Down Up
##     Down   21 30
##     Up     22 31
```

```
mean(knn.pred == Direction.2009and10)
```

```
## [1] 0.5
```

Part H

Logistic Regression and Linear Discriminant Analysis tied for the best test results, both resulting in a `62.5%` success rate.

Part I

2-means

```
train.X = as.matrix(Lag2[train])
test.X  = as.matrix(Lag2[!train])
train.Direction = Direction[train]

set.seed(1)
knn.pred = knn(train.X, test.X, train.Direction, k = 2)
table(knn.pred, Direction.2009and10)
```

```
##         Direction.2009and10
## knn.pred Down Up
##     Down   19 27
##     Up     24 34
```

```
mean(knn.pred == Direction.2009and10)
```

```
## [1] 0.5096154
```

3-means

```
train.X = as.matrix(Lag2[train])
test.X  = as.matrix(Lag2[!train])
train.Direction = Direction[train]

set.seed(1)
knn.pred = knn(train.X, test.X, train.Direction, k = 3)
table(knn.pred, Direction.2009and10)
```

```
##          Direction.2009and10
## knn.pred Down Up
##     Down   16 20
##     Up     27 41
```

```
mean(knn.pred == Direction.2009and10)
```

```
## [1] 0.5480769
```

4-means

```
train.X = as.matrix(Lag2[train])
test.X  = as.matrix(Lag2[!train])
train.Direction = Direction[train]

set.seed(1)
knn.pred = knn(train.X, test.X, train.Direction, k = 4)
table(knn.pred, Direction.2009and10)
```

```
##          Direction.2009and10
## knn.pred Down Up
##     Down   20 17
##     Up     23 44
```

```
mean(knn.pred == Direction.2009and10)
```

```
## [1] 0.6153846
```

5-means

```
train.X = as.matrix(Lag2[train])
test.X  = as.matrix(Lag2[!train])
train.Direction = Direction[train]

set.seed(1)
knn.pred = knn(train.X, test.X, train.Direction, k = 5)
table(knn.pred, Direction.2009and10)
```

```
##         Direction.2009and10
## knn.pred Down Up
##     Down   16 21
##     Up     27 40
```

```
mean(knn.pred == Direction.2009and10)
```

```
## [1] 0.5384615
```

Problem 5

```
library(MASS)
library(nnet)
library(ggplot2)

rosters <- read.csv('rosters.csv')
summary(rosters)
```

```
##        X             gender        height        homestate
##  Min.   :  0.00   male:204   Min.   :62.00   CA     :70
##  1st Qu.: 50.75              1st Qu.:71.00   WA     :12
##  Median :101.50              Median :74.00   TX     :11
##  Mean   :101.63              Mean   :73.33   GA     : 9
##  3rd Qu.:152.25              3rd Qu.:76.00   AZ     : 8
##  Max.   :206.00              Max.   :84.00   FL     : 8
##                                              (Other):86
##                  name             sport        weight
##  Alabi, Adrian      :  1   Baseball   :29   Min.   :125.0
##  Alexander, Terrence:  1   Basketball :12   1st Qu.:176.5
##  Alfieri, Joey      :  1   Football   :95   Median :197.0
##  Allen, Malcolm     :  1   Soccer     :25   Mean   :205.4
##  Allen, Marcus      :  1   Tennis     :10   3rd Qu.:229.2
##  Allen, Rosco       :  1   Wrestling  :33   Max.   :321.0
##  (Other)            :198
```

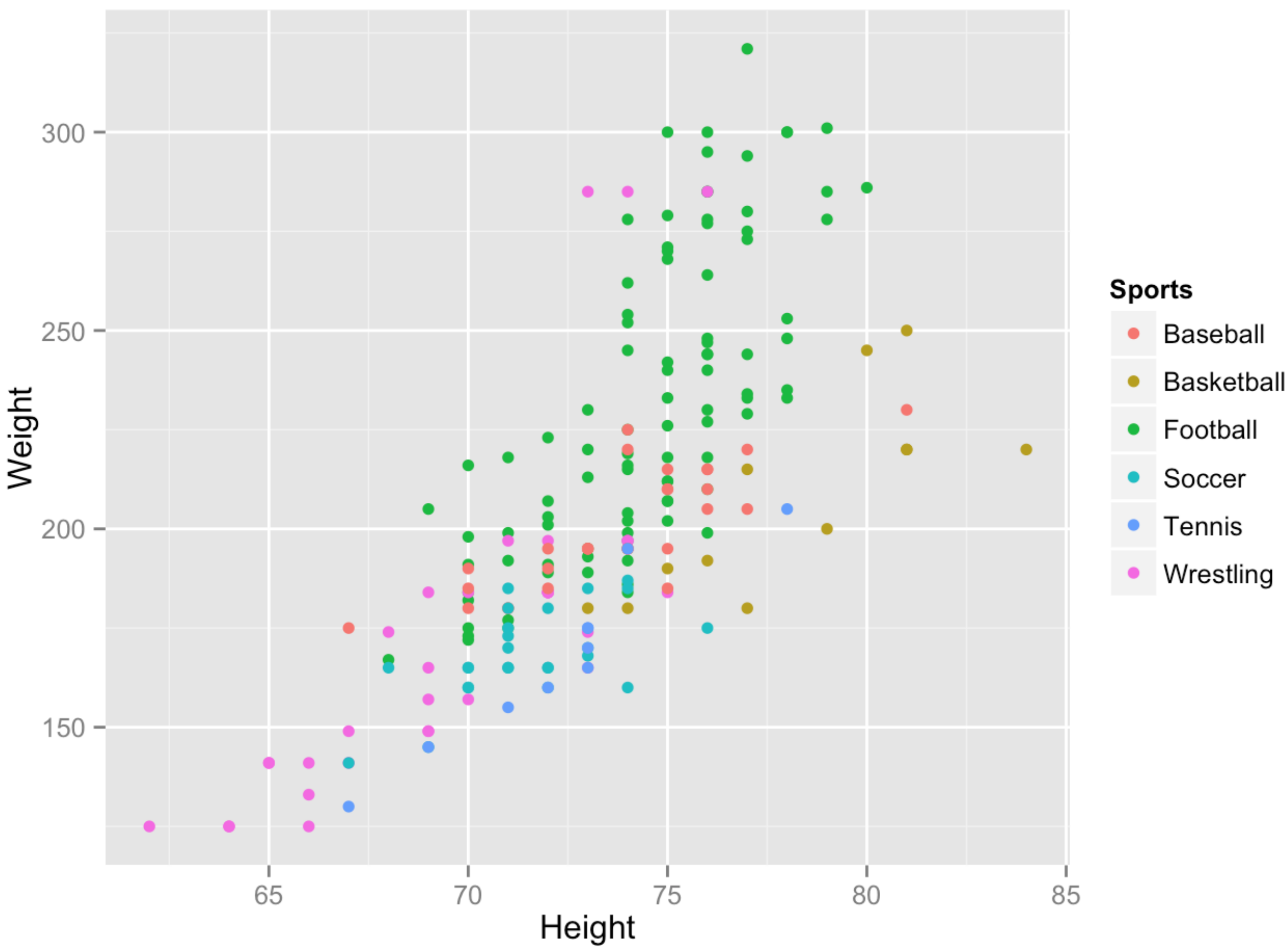
```
fit = multinom(sport ~ height + weight, rosters)
```

```
## # weights:  24 (15 variable)
## initial  value 365.518932
## iter  10 value 266.164885
## iter  20 value 207.817029
## iter  30 value 200.902009
## iter  40 value 200.198801
## iter  50 value 199.808004
## iter  60 value 199.680438
## iter  70 value 199.653822
## iter  80 value 199.647074
## iter  90 value 199.645164
## final  value 199.644727
## converged
```

```
summary(fit)
```

```
## Call:
## multinom(formula = sport ~ height + weight, data = rosters)
##
## Coefficients:
##             (Intercept)     height      weight
## Basketball   -71.754980  1.1940557 -0.09567987
## Football      16.366376 -0.3500011  0.05057577
## Soccer        -6.939543  0.3534022 -0.10321321
## Tennis       -30.636220  0.8123257 -0.16392309
## Wrestling     44.531678 -0.6709520  0.02033305
##
## Std. Errors:
##             (Intercept)     height     weight
## Basketball  0.140277235 0.06577436 0.02461660
## Football    4.953425402 0.08643443 0.01180643
## Soccer      0.083352009 0.04979394 0.01992204
## Tennis      0.004621523 0.07325690 0.03054070
## Wrestling   3.278747021 0.06415213 0.01411842
##
## Residual Deviance: 399.2895
## AIC: 429.2895
```

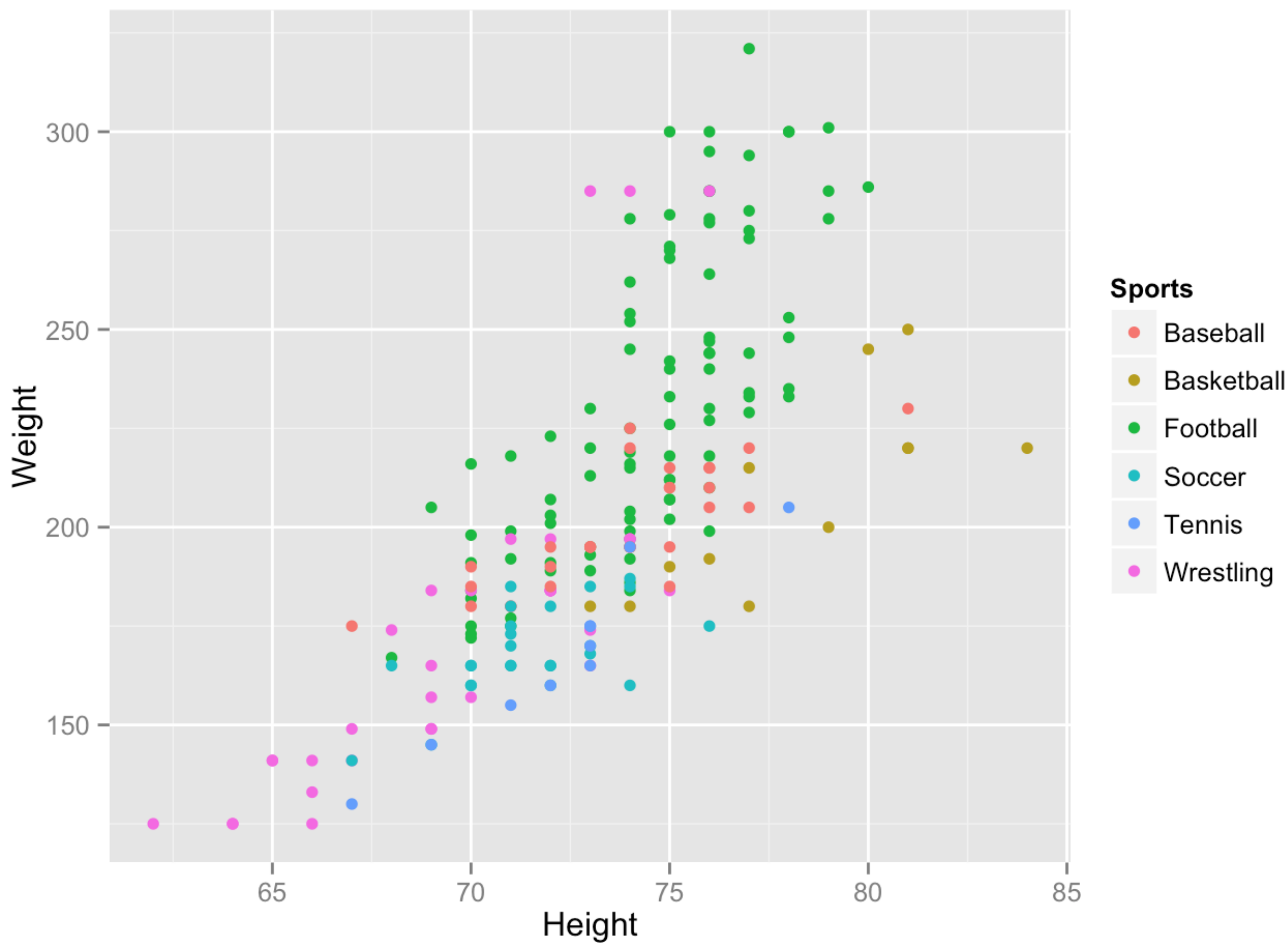
```
Height = rosters$height
Weight = rosters$weight
Sports = rosters$sport
qplot(Height, Weight, rosters, color=Sports)
```




```
pred = predict(fit, rosters)
pred
```

```
##   [1] Wrestling   Football    Football    Football    Football  
##   [6] Football    Soccer      Football    Football    Football  
##  [11] Football    Wrestling   Football    Football    Football  
##  [16] Football    Football    Football    Football    Football  
##  [21] Football    Football    Football    Baseball    Football  
##  [26] Football    Football    Football    Football    Football  
##  [31] Football    Wrestling   Football    Football    Football  
##  [36] Wrestling   Football    Football    Soccer      Football  
##  [41] Football    Football    Football    Football    Football  
##  [46] Football    Football    Wrestling   Football    Football  
##  [51] Wrestling   Football    Football    Football    Football  
##  [56] Football    Football    Football    Football    Football  
##  [61] Football    Football    Football    Football    Football  
##  [66] Football    Football    Football    Football    Football  
##  [71] Football    Football    Football    Football    Football  
##  [76] Football    Football    Football    Football    Football  
##  [81] Football    Football    Wrestling   Football    Football  
##  [86] Wrestling   Football    Football    Football    Football  
##  [91] Football    Football    Football    Football    Football  
##  [96] Football    Soccer      Wrestling   Football    Soccer    
## [101] Football    Football    Wrestling   Wrestling   Wrestling 
## [106] Wrestling   Soccer      Football    Football    Wrestling 
## [111] Soccer      Football    Football    Wrestling   Soccer    
## [116] Soccer      Football    Wrestling   Wrestling   Football  
## [121] Soccer      Wrestling   Wrestling   Football    Wrestling 
## [126] Football    Wrestling   Wrestling   Basketball  Soccer    
## [131] Soccer      Basketball  Football    Basketball  Baseball  
## [136] Basketball  Basketball  Football    Basketball  Basketball
## [141] Football    Wrestling   Football    Football    Wrestling 
## [146] Football    Football    Football    Football    Wrestling 
## [151] Football    Football    Football    Football    Football  
## [156] Football    Football    Football    Soccer      Football  
## [161] Football    Football    Baseball    Football    Basketball
## [166] Football    Basketball  Football    Football    Football  
## [171] Wrestling   Wrestling   Soccer      Soccer      Soccer    
## [176] Wrestling   Football    Football    Soccer      Soccer    
## [181] Soccer      Football    Soccer      Soccer      Tennis    
## [186] Soccer      Soccer      Wrestling   Wrestling   Tennis    
## [191] Soccer      Wrestling   Baseball    Soccer      Basketball
## [196] Soccer      Soccer      Soccer      Football    Soccer    
## [201] Soccer      Soccer      Soccer      Soccer    
## Levels: Baseball Basketball  Football Soccer Tennis Wrestling
```

```
qplot(Height, Weight, pred, color=Sports)
```




```
table(rosters$sport, pred)
```

```
##              pred
##               Baseball Basketball  Football Soccer Tennis Wrestling
##    Baseball          1          2        22      1      0         3
##    Basketball        1          7         2      2      0         0
##    Football          1          0        84      2      0         8
##    Soccer            1          0         4     12      2         6
##    Tennis            0          1         1      8      0         0
##    Wrestling         0          0        12      7      0        14
```

```
success_rate = mean(pred == rosters$sport)
```

```
## Our success rate is 0.5784314
```

```
## Our 0-1 loss error rate is 0.4215686
```