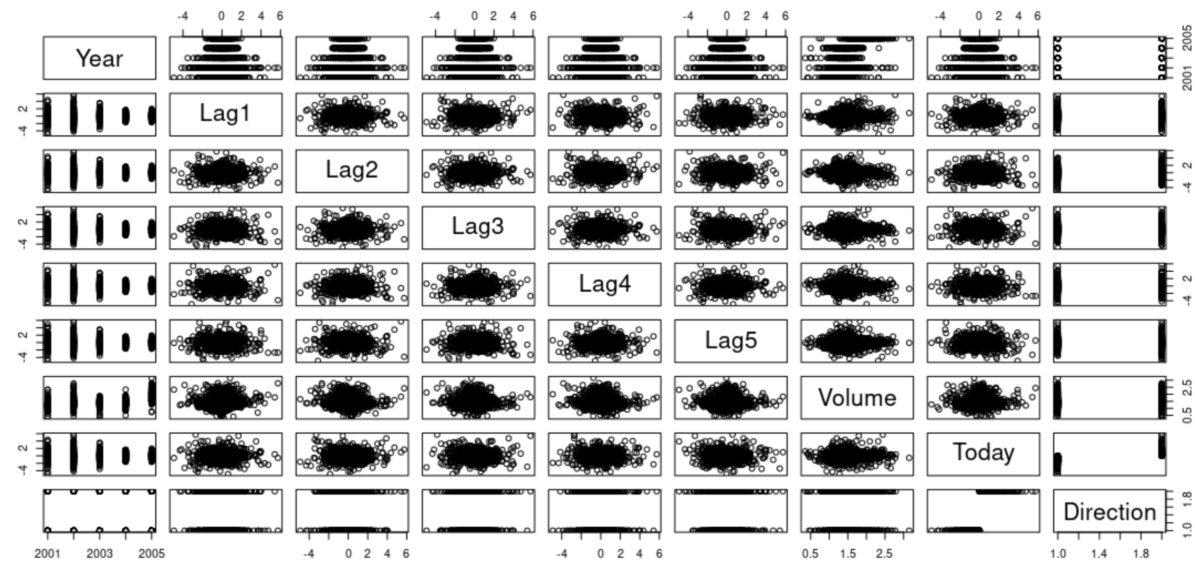
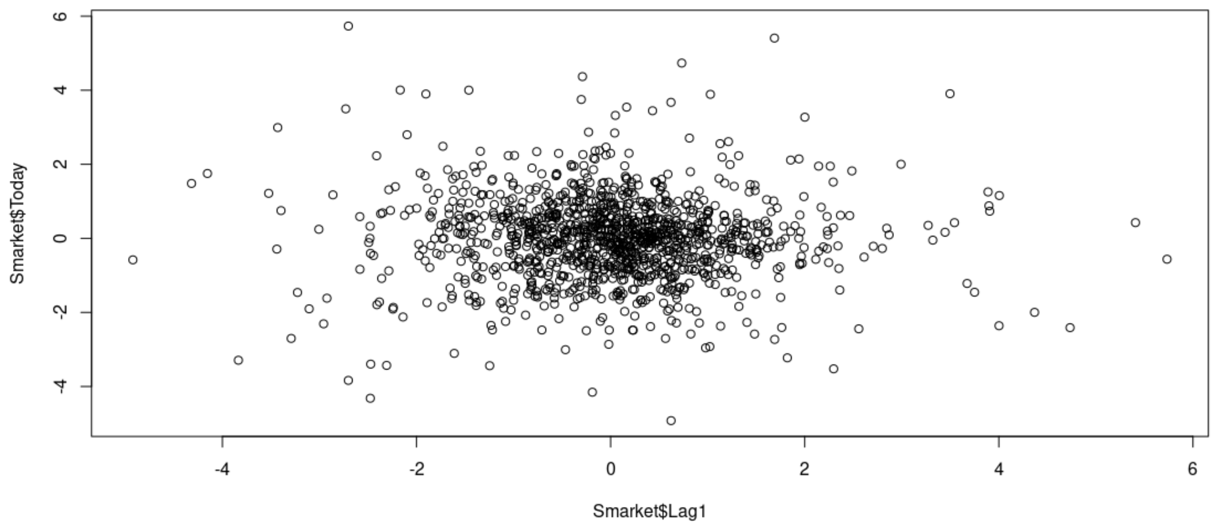
Part 1:

```
> getwd()
[1] "/home/adam/Documents/JH_MBA/BU.510.650.33_Data_Analytics/Module4"
> library(ISLR)
> attach(Smarket)
The following objects are masked from Smarket (pos = 5):

    Direction, Lag1, Lag2, Lag3, Lag4, Lag5, Today, Volume, Year

> head(Smarket)
  Year  Lag1  Lag2  Lag3  Lag4  Lag5 Volume Today Direction
1 2001  0.38 -0.19 -2.62 -1.05  5.01    1.2  0.96        Up
2 2001  0.96  0.38 -0.19 -2.62 -1.05    1.3  1.03        Up
3 2001  1.03  0.96  0.38 -0.19 -2.62    1.4 -0.62      Down
4 2001 -0.62  1.03  0.96  0.38 -0.19    1.3  0.61        Up
5 2001  0.61 -0.62  1.03  0.96  0.38    1.2  0.21        Up
6 2001  0.21  0.61 -0.62  1.03  0.96    1.3  1.39        Up
> tail(Smarket)
     Year   Lag1   Lag2   Lag3   Lag4   Lag5 Volume  Today Direction
1245 2005  0.252 -0.024 -0.584 -0.285 -0.141    2.1  0.422        Up
1246 2005  0.422  0.252 -0.024 -0.584 -0.285    1.9  0.043        Up
1247 2005  0.043  0.422  0.252 -0.024 -0.584    1.3 -0.955      Down
1248 2005 -0.955  0.043  0.422  0.252 -0.024    1.5  0.130        Up
1249 2005  0.130 -0.955  0.043  0.422  0.252    1.4 -0.298      Down
1250 2005 -0.298  0.130 -0.955  0.043  0.422    1.4 -0.489      Down
> summary(Smarket)
      Year          Lag1          Lag2          Lag3          Lag4          Lag5         Volume
 Min.   :2001   Min.   :-4.9   Min.   :-4.9   Min.   :-4.9   Min.   :-4.9   Min.   :-4.9   Min.   :0.36
 1st Qu.:2002   1st Qu.:-0.6   1st Qu.:-0.6   1st Qu.:-0.6   1st Qu.:-0.6   1st Qu.:-0.6   1st Qu.:1.26
 Median :2003   Median : 0.0   Median : 0.0   Median : 0.0   Median : 0.0   Median : 0.0   Median
:1.42
 Mean   :2003   Mean   : 0.0   Mean   : 0.0   Mean   : 0.0   Mean   : 0.0   Mean   : 0.0   Mean
:1.48
 3rd Qu.:2004   3rd Qu.: 0.6   3rd Qu.: 0.6   3rd Qu.: 0.6   3rd Qu.: 0.6   3rd Qu.: 0.6   3rd
Qu.:1.64
 Max.   :2005   Max.   : 5.7   Max.   : 5.7   Max.   : 5.7   Max.   : 5.7   Max.   : 5.7   Max.   :3.15
     Today      Direction
 Min.   :-4.9   Down:602
 1st Qu.:-0.6   Up  :648
 Median : 0.0
 Mean   : 0.0
 3rd Qu.: 0.6
 Max.   : 5.7
> cor(Smarket)
```

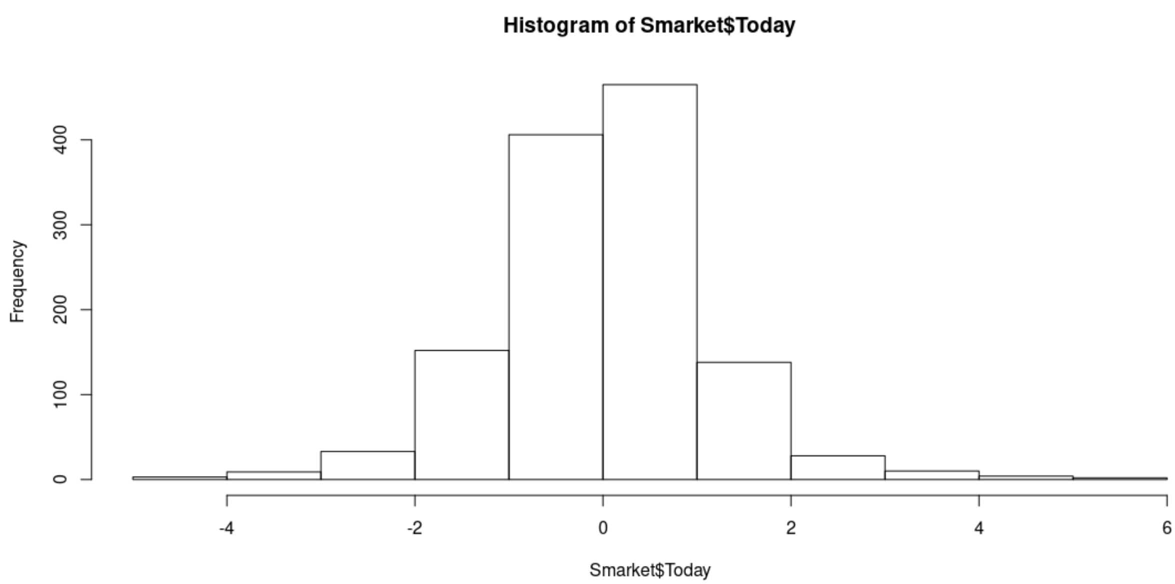
```
Error in cor(Smarket) : 'x' must be numeric
> cor(Smarket[,-9])
       Year    Lag1    Lag2    Lag3    Lag4    Lag5 Volume   Today
Year   1.000  0.0297  0.0306  0.0332  0.0357  0.0298  0.539  0.0301
Lag1   0.030  1.0000 -0.0263 -0.0108 -0.0030 -0.0057  0.041 -0.0262
Lag2   0.031 -0.0263  1.0000 -0.0259 -0.0109 -0.0036 -0.043 -0.0103
Lag3   0.033 -0.0108 -0.0259  1.0000 -0.0241 -0.0188 -0.042 -0.0024
Lag4   0.036 -0.0030 -0.0109 -0.0241  1.0000 -0.0271 -0.048 -0.0069
Lag5   0.030 -0.0057 -0.0036 -0.0188 -0.0271  1.0000 -0.022 -0.0349
Volume 0.539  0.0409 -0.0434 -0.0418 -0.0484 -0.0220  1.000  0.0146
Today  0.030 -0.0262 -0.0103 -0.0024 -0.0069 -0.0349  0.015  1.0000
> #### plots
> plot(Smarket)
```



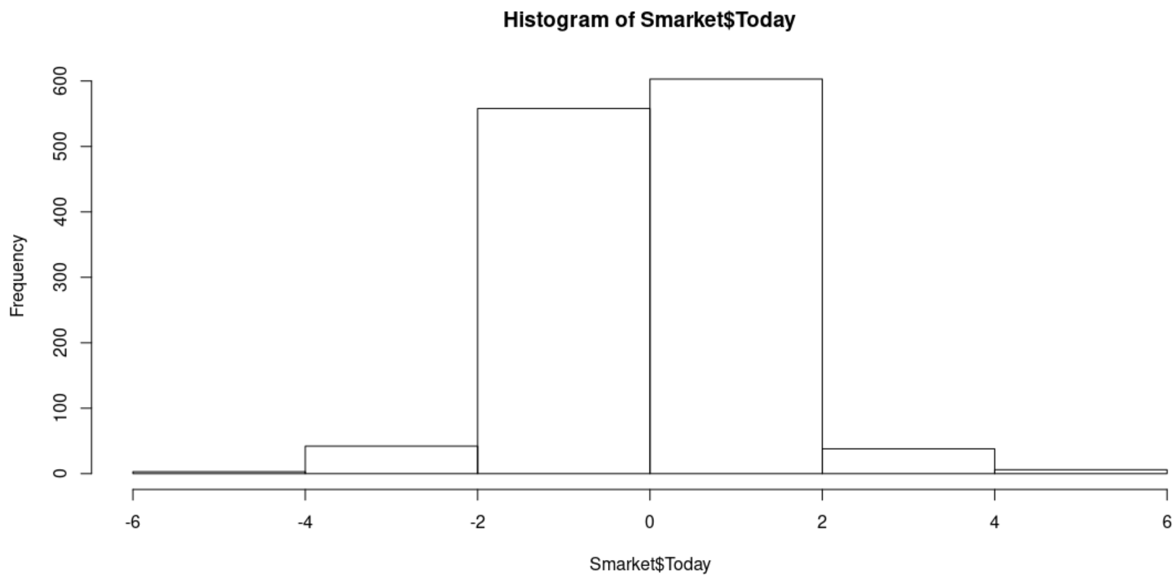

```
> plot(Smarket$Today~Smarket$Lag1
```



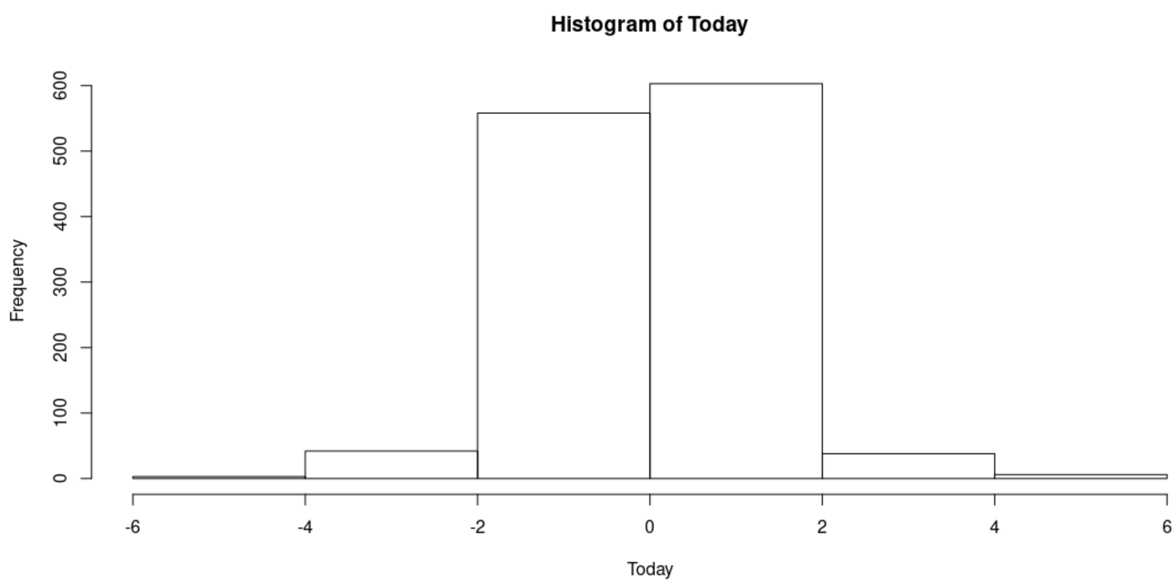
)
> plot(Smarket$Lag1,Smarket$Today)
> hist(Smarket$Today)



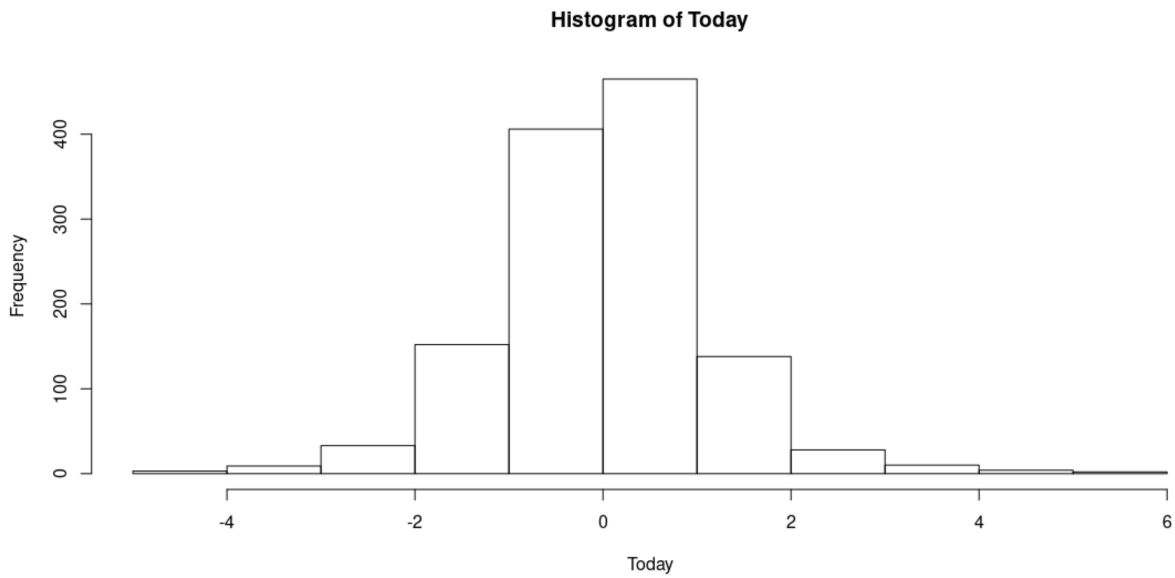
> hist(Smarket$Today,breaks=5)



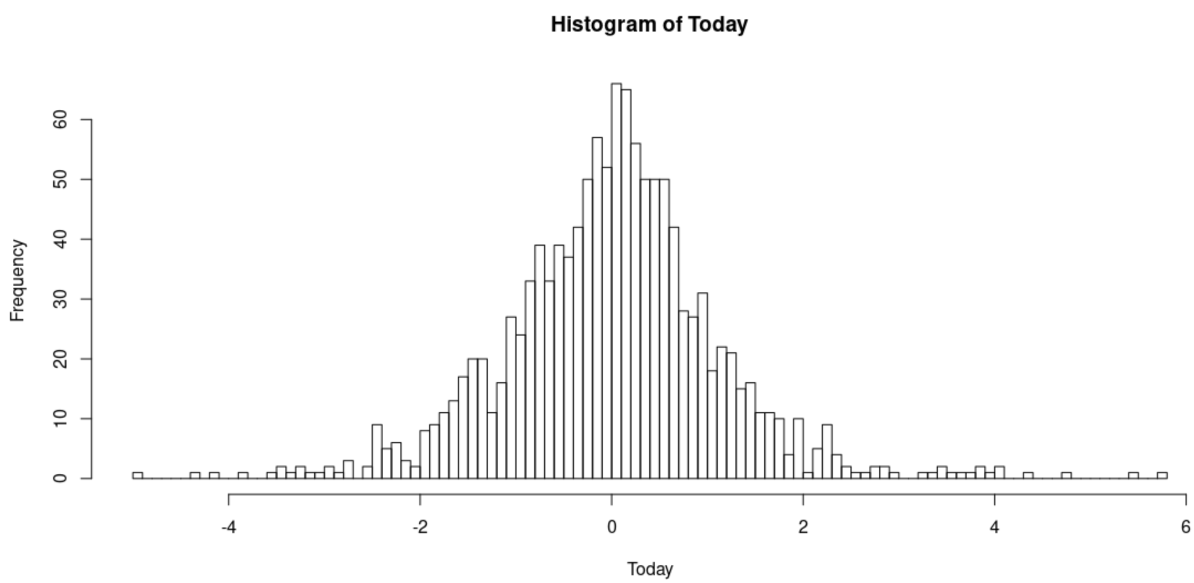
> hist(Today,breaks=5)



> hist(Today,breaks=10)



> hist(Today,breaks=100)

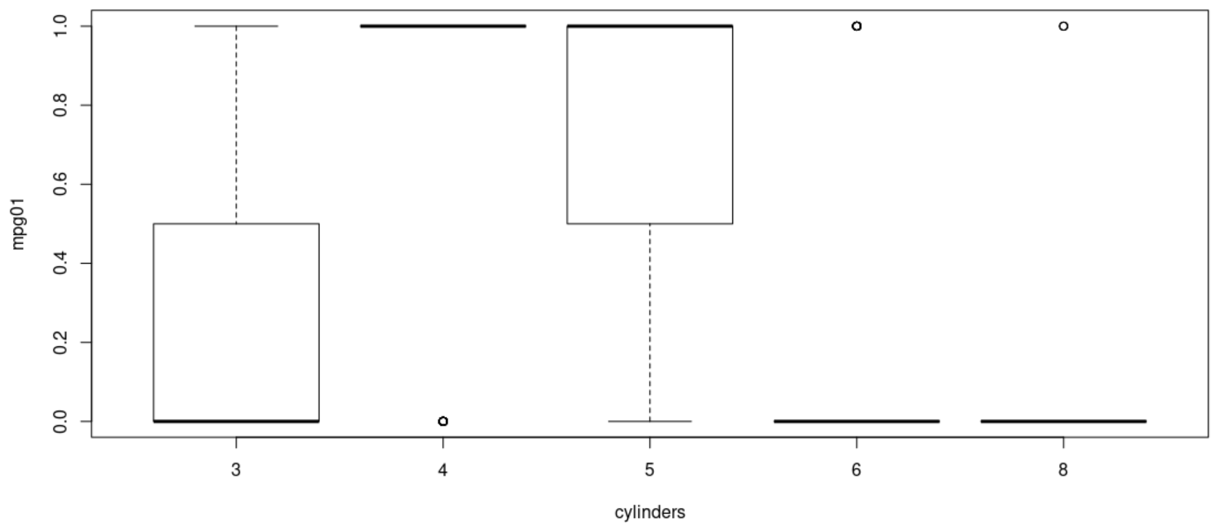



```
> #### logistic regression
> glm.fit=glm(Direction~Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume,
+           family=binomial,data=Smarket)
> summary(glm.fit)

Call:
glm(formula = Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 +
   Volume, family = binomial, data = Smarket)

Deviance Residuals:
```

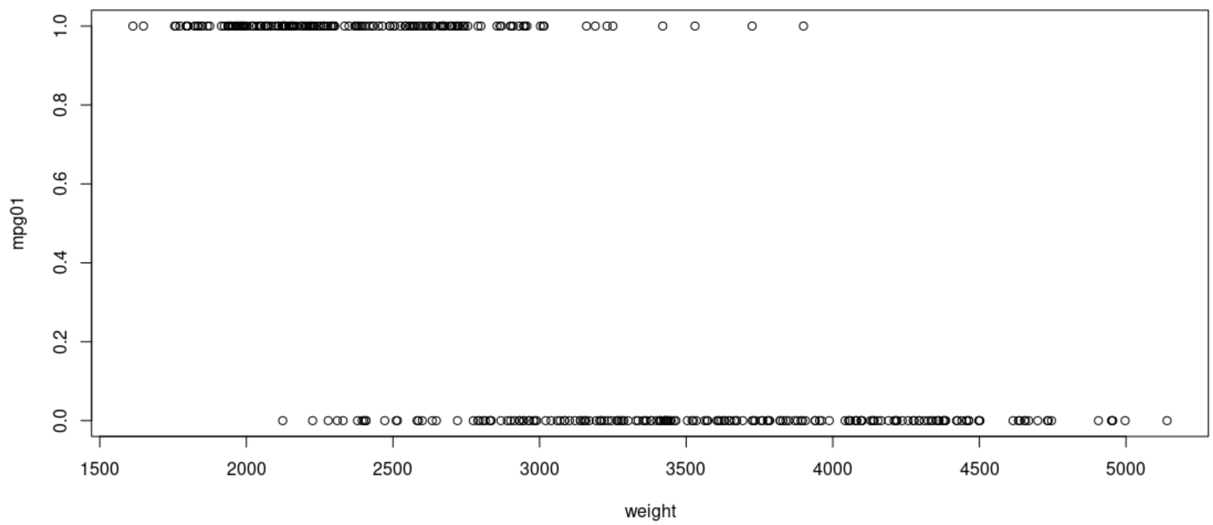
```
   Min      1Q  Median      3Q     Max
 -1.45   -1.20    1.07    1.15    1.33

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -0.12600    0.24074   -0.52     0.60
Lag1        -0.07307    0.05017   -1.46     0.15
Lag2        -0.04230    0.05009   -0.84     0.40
Lag3         0.01109    0.04994    0.22     0.82
Lag4         0.00936    0.04997    0.19     0.85
Lag5         0.01031    0.04951    0.21     0.83
Volume       0.13544    0.15836    0.86     0.39

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1731.2  on 1249  degrees of freedom
Residual deviance: 1727.6  on 1243  degrees of freedom
AIC: 1742

Number of Fisher Scoring iterations: 3

> predict(glm.fit,type="response")->glm.probs
> head(glm.probs)
   1    2    3    4    5    6
0.51 0.48 0.48 0.52 0.51 0.51
> options("digits"=2)
> glm.probs[1:6]
   1    2    3    4    5    6
0.51 0.48 0.48 0.52 0.51 0.51
> contrasts(Direction)
     Up
Down  0
Up    1
> glm.pred=rep("Down",1250)
> glm.pred[glm.probs>.5]="Up"
> head(glm.pred)
[1] "Up"   "Down" "Down" "Up"   "Up"   "Up"
> head(Direction)
[1] Up   Up   Down Up   Up   Up
Levels: Down Up
> table(glm.pred,Direction)
        Direction
glm.pred Down  Up
    Down  145 141
```

```
   Up    457 507
> (145+507)/1250
[1] 0.52
> mean(glm.pred==Direction)
[1] 0.52
> levels(Smarket$Year)
NULL
> class(Smarket$Year)
[1] "numeric"
> train=(Year<2005)
> Smarket.2005=Smarket[!train,]
> dim(Smarket.2005)
[1] 252   9
> Direction.2005=Direction[!train]
> head(Direction.2005)
[1] Down Down Down Up   Down Up
Levels: Down Up
> length(Direction.2005)
[1] 252
> glm.fit=glm(Direction~Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume,
+             family=binomial,data=Smarket,subset=train)
> glm.probs=predict(glm.fit,newdata=Smarket.2005,type="response")
> glm.pred[glm.probs>.5]="Up"
> glm.pred=rep("Down",252)
> glm.pred[glm.probs>.5]="Up"
> mean(glm.pred==Direction.2005)
[1] 0.48
> # improve prediction accuracy
> glm.fit=glm(Direction ~ Lag1+Lag2, data=Smarket, family=binomial,subset=train)
> glm.probs=predict(glm.fit,Smarket.2005,type="response")
> glm.pred=rep("Down",252)
> glm.pred[glm.probs>.5]="Up"
> table(glm.pred,Direction.2005)
        Direction.2005
glm.pred Down  Up
   Down   35  35
   Up     76 106
> mean(glm.pred==Direction.2005)
[1] 0.56
> ############## Auto Data Set
> Auto=read.csv("Auto.csv",header=T,na.strings="?")
> Auto2=na.omit(Auto) # remove missing values
> mpg.median=median(Auto2$mpg)
> mpg01 <- ifelse(Auto2$mpg > mpg.median, 1, 0)
```

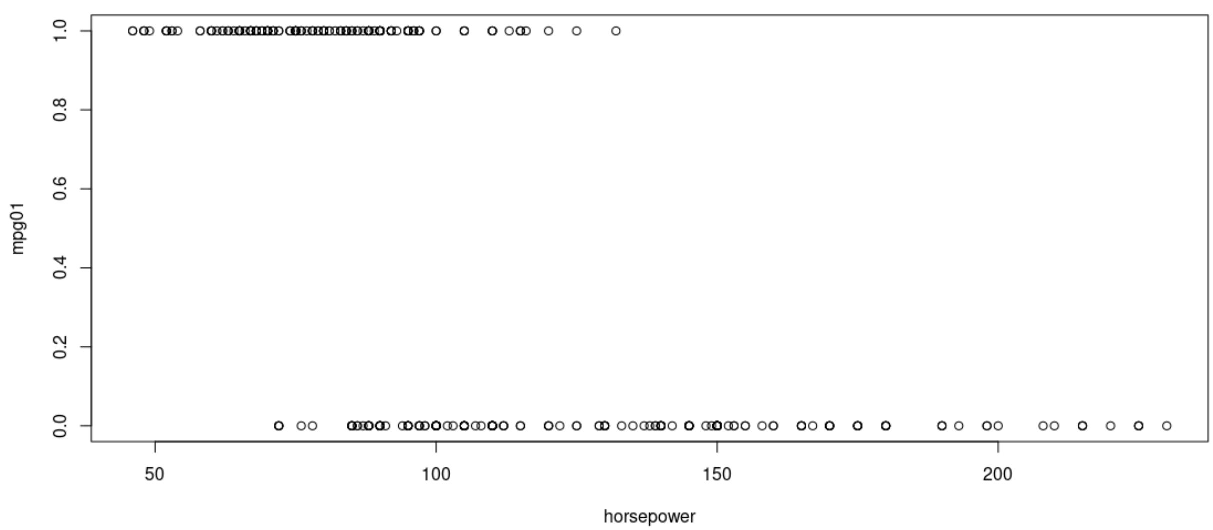
```
> Auto3=data.frame(mpg01,Auto2[,-1]) # create a new data frame
> head(Auto3)
  mpg01 cylinders displacement horsepower weight acceleration year origin                      name
1     0         8          307        130   3504           12   70      1 chevrolet chevelle malibu
2     0         8          350        165   3693           12   70      1         buick skylark 320
3     0         8          318        150   3436           11   70      1        plymouth satellite
4     0         8          304        150   3433           12   70      1             amc rebel sst
5     0         8          302        140   3449           10   70      1               ford torino
6     0         8          429        198   4341           10   70      1          ford galaxie 500
> # explore the data w.r.t. mgp01
> boxplot(mpg01~cylinders, data=Auto3)
```



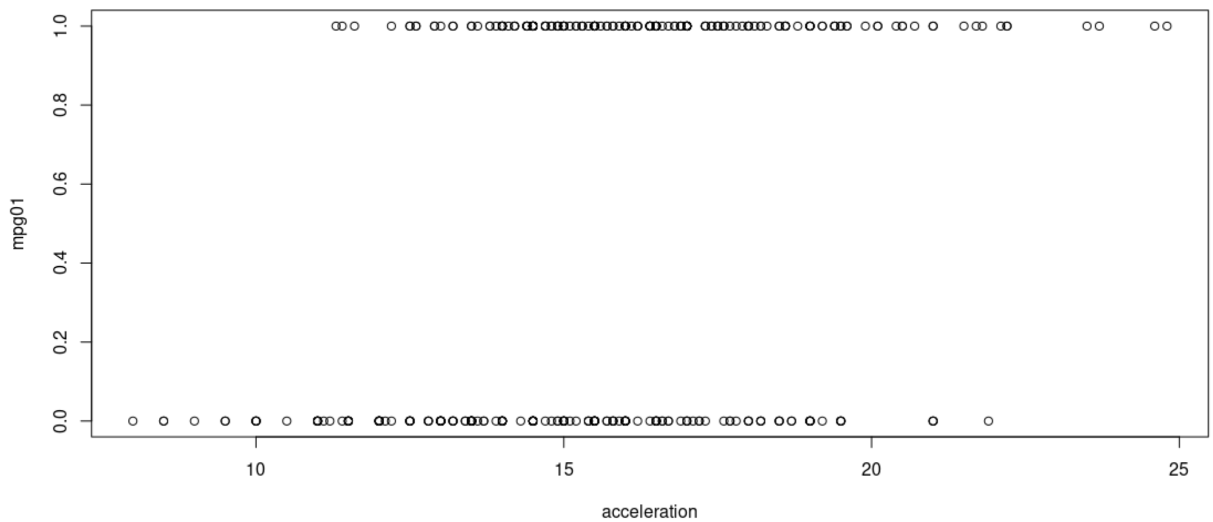

```
> plot(mpg01~weight, data=Auto3)
```



> plot(mpg01~horsepower, data=Auto3)



> plot(mpg01~acceleration, data=Auto3)




```
> train= seq(1,nrow(Auto3)/2) # create indeces for training data set
> train
  [1]   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25
 [26]  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49
50
 [51]  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74
75
 [76]  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99
100
[101] 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121
122 123 124 125
[126] 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145
146 147 148 149 150
[151] 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170
171 172 173 174 175
[176] 176 177 178 179 180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195
196
> test= seq(nrow(Auto3)/2+1, nrow(Auto3))
> test
  [1] 197 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217
218 219 220 221
 [26] 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240 241
242 243 244 245 246
 [51] 247 248 249 250 251 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266
267 268 269 270 271
 [76] 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 288 289 290 291
292 293 294 295 296
```

```
[101] 297 298 299 300 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316
317 318 319 320 321
[126] 322 323 324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341
342 343 344 345 346
[151] 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362 363 364 365 366
367 368 369 370 371
[176] 372 373 374 375 376 377 378 379 380 381 382 383 384 385 386 387 388 389 390 391
392
> # perform logistic regression
> logit.auto=glm(mpg01~ cylinders+weight+horsepower +acceleration,
family="binomial",data=Auto3)
> summary(logit.auto)

Call:
glm(formula = mpg01 ~ cylinders + weight + horsepower + acceleration,
    family = "binomial", data = Auto3)

Deviance Residuals:
   Min      1Q  Median      3Q     Max
-2.461  -0.193   0.050   0.380   3.250

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  14.020986   2.757000    5.09  3.7e-07 ***
cylinders    -0.460909   0.230505   -2.00   0.0455 *
weight       -0.002373   0.000744   -3.19   0.0014 **
horsepower   -0.044918   0.020100   -2.23   0.0254 *
acceleration -0.042621   0.123279   -0.35   0.7295
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 543.43  on 391  degrees of freedom
Residual deviance: 209.68  on 387  degrees of freedom
AIC: 219.7

Number of Fisher Scoring iterations: 7

> # p-value for acceleration is not significant, re-perform logistic regression without it
> logit.auto=glm(mpg01~ cylinders+weight+horsepower, family="binomial",data=Auto3)
> summary(logit.auto)

Call:
```

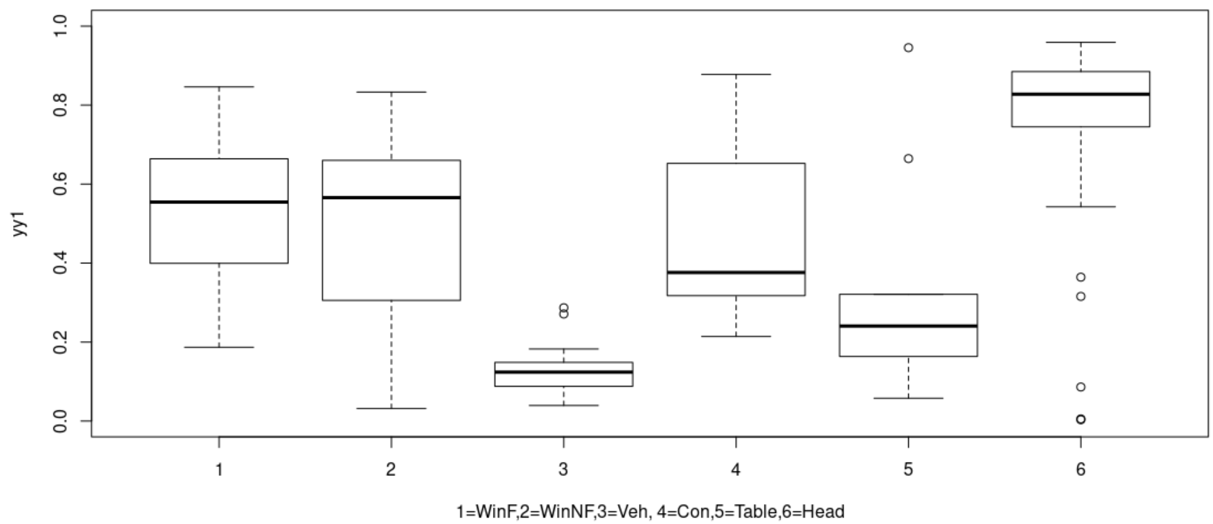
```
glm(formula = mpg01 ~ cylinders + weight + horsepower, family = "binomial",
    data = Auto3)

Deviance Residuals:
   Min      1Q  Median      3Q     Max
-2.440  -0.195   0.050   0.375   3.221

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) 13.22467    1.46848    9.01  < 2e-16 ***
cylinders   -0.44140    0.22273   -1.98   0.0475 *
weight      -0.00254    0.00058   -4.37  1.2e-05 ***
horsepower  -0.03987    0.01370   -2.91   0.0036 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 543.43  on 391  degrees of freedom
Residual deviance: 209.80  on 388  degrees of freedom
AIC: 217.8

Number of Fisher Scoring iterations: 7

> # you may also want to remove cylinders
> logit.auto=glm(mpg01~ weight+horsepower, family="binomial",data=Auto3)
> summary(logit.auto)

Call:
glm(formula = mpg01 ~ weight + horsepower, family = "binomial",
    data = Auto3)

Deviance Residuals:
   Min      1Q  Median      3Q     Max
-2.312  -0.207   0.036   0.351   3.092

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) 13.632842   1.503203    9.07  < 2e-16 ***
weight      -0.003245   0.000473   -6.86    7e-12 ***
horsepower  -0.045980   0.013147   -3.50  0.00047 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

```
(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 543.43  on 391  degrees of freedom
Residual deviance: 213.83  on 389  degrees of freedom
AIC: 219.8

Number of Fisher Scoring iterations: 7

> # perform logistic regression with training data, use testing data for accuracy
> logit.auto2=glm(mpg01~ weight+horsepower, family="binomial",data=Auto3,subset=train)
> auto.probs=predict(logit.auto2,newdata=Auto3[test,],type="response")
> auto.pred=rep("Down",length(test))
> auto.pred[auto.probs>.5]="Up"
> table(auto.pred,Auto3$mpg01[test]) ->test.table
> prediction.accuracy <- (test.table[1,1]+test.table[2,2])/sum(test.table)
> prediction.accuracy
[1] 0.8
> ##################################################
> #### multinomial logistic regression
> ##################################################
> library(VGAM) ## VGAM to estimate multinomial logistic regression
Loading required package: stats4
Loading required package: splines
> library(textir)## to standardize the features
Loading required package: distrom
Loading required package: Matrix
Loading required package: gamlr

Attaching package: ‘gamlr’

The following object is masked from ‘package:VGAM’:

    AICc

Loading required package: parallel
> library(MASS)## a library of example datasets
> data(fgl)## loads the data into R; see help(fgl)
> fgl[1:3,]
     RI Na  Mg  Al Si    K  Ca Ba Fe type
1  3.01 14 4.5 1.1 72 0.06 8.8  0  0 WinF
2 -0.39 14 3.6 1.4 73 0.48 7.8  0  0 WinF
3 -1.82 14 3.5 1.5 73 0.39 7.8  0  0 WinF
> gg <- vglm(type ~ Na+Mg+Al,multinomial,data=fgl)
> summary(gg)
```

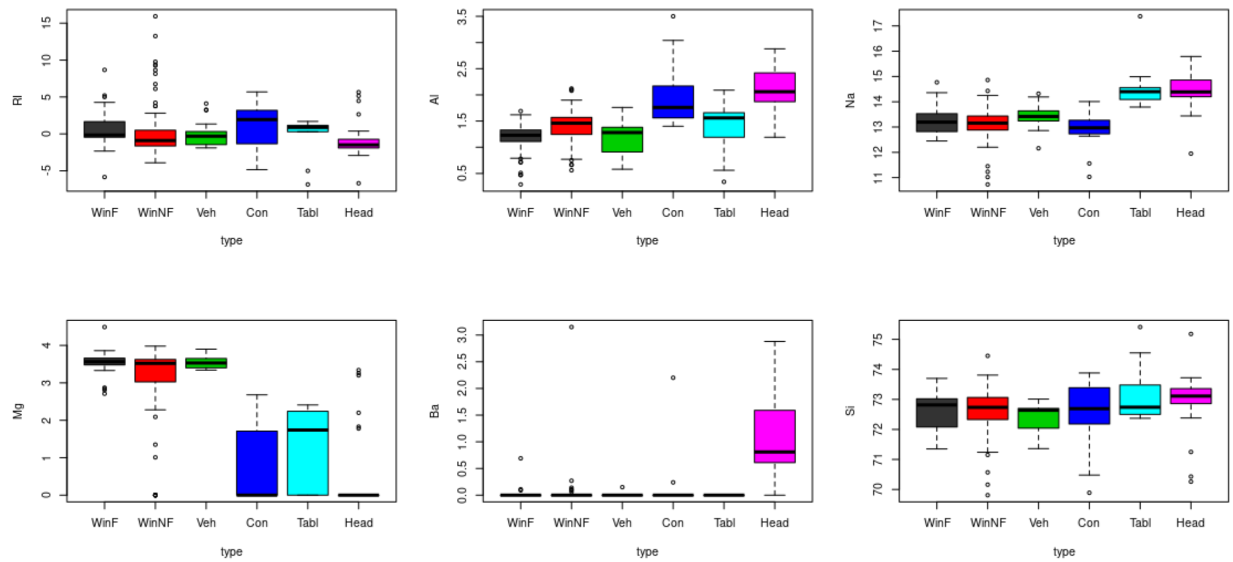
```
Call:
vglm(formula = type ~ Na + Mg + Al, family = multinomial, data = fgl)

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept):1  46.0044    10.9336    4.21  2.6e-05 ***
(Intercept):2  38.5771     9.4632    4.08  4.6e-05 ***
(Intercept):3  26.7163    12.7079    2.10  0.03552 *
(Intercept):4  38.3292     9.7985    3.91  9.2e-05 ***
(Intercept):5   0.5872     9.6285    0.06  0.95137
Na:1           -3.0413     0.7999   -3.80  0.00014 ***
Na:2           -2.4880     0.6784   -3.67  0.00025 ***
Na:3           -1.7261     0.8905      NA       NA
Na:4           -2.9177     0.7278   -4.01  6.1e-05 ***
Na:5            0.1855     0.6534    0.28  0.77650
Mg:1            2.6642     0.5316    5.01  5.4e-07 ***
Mg:2            1.1766     0.3310    3.55  0.00038 ***
Mg:3            2.2818     0.7097    3.22  0.00130 **
Mg:4            0.0357     0.3486    0.10  0.91848
Mg:5            0.4848     0.3490    1.39  0.16486
Al:1           -7.4505     1.3630   -5.47  4.6e-08 ***
Al:2           -3.4144     1.0977   -3.11  0.00187 **
Al:3           -6.0210     1.5133   -3.98  6.9e-05 ***
Al:4            0.5278     0.8014    0.66  0.51018
Al:5           -2.7932     1.0278   -2.72  0.00657 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Number of linear predictors:  5

Names of linear predictors: log(mu[,1]/mu[,6]), log(mu[,2]/mu[,6]), log(mu[,3]/mu[,6]),
log(mu[,4]/mu[,6]), log(mu[,5]/mu[,6])

Residual deviance: 380 on 1050 degrees of freedom

Log-likelihood: -190 on 1050 degrees of freedom

Number of Fisher scoring iterations: 7

Warning: Hauck-Donner effect detected in the following estimate(s):
'(Intercept):2', 'Na:3', 'Na:4', 'Al:3'
```

Reference group is level 6 of the response

```
> dWinF=fgl$type=="WinF"
> dWinNF=fgl$type=="WinNF"
> dVeh=fgl$type=="Veh"
> dCon=fgl$type=="Con"
> dTable=fgl$type=="Tabl"
> dHead=fgl$type=="Head"
> yy1=c(fitted(gg)[dWinF,1],fitted(gg)[dWinNF,2],fitted(gg)[dVeh,3],
fitted(gg)[dCon,4],fitted(gg)[dTable,5],fitted(gg)[dHead,6])
> xx1=c(fgl$type[dWinF],fgl$type[dWinNF],fgl$type[dVeh], fgl$type[dCon], fgl$type[dTable],
fgl$type[dHead])
> boxplot(yy1~xx1,ylim=c(0,1),xlab="1=WinF,2=WinNF,3=Veh, 4=Con,5=Table,6=Head")
```




```
> # more boxplots
> par(mfrow=c(2,3))
> plot(RI ~ type, data=fgl, col=c(grey(.2),2:6))
> plot(Al ~ type, data=fgl, col=c(grey(.2),2:6))
> plot(Na ~ type, data=fgl, col=c(grey(.2),2:6))
> plot(Mg ~ type, data=fgl, col=c(grey(.2),2:6))
> plot(Ba ~ type, data=fgl, col=c(grey(.2),2:6))
> plot(Si ~ type, data=fgl, col=c(grey(.2),2:6))
```




```
> par(mfrow=c(1,1))
> #####################################################
> #### K-Nearest Neighbors: Example
> #####################################################
> library(ISLR)
> attach(Smarket)
The following objects are masked from Smarket (pos = 11):

    Direction, Lag1, Lag2, Lag3, Lag4, Lag5, Today, Volume, Year

The following objects are masked from Smarket (pos = 14):

    Direction, Lag1, Lag2, Lag3, Lag4, Lag5, Today, Volume, Year

> data(Smarket)
> train=(Year<2005)
> Smarket.2005=Smarket[!train,]
> dim(Smarket.2005)
[1] 252   9
> ###### k-nearest neighbor
> library(class)
> train.X=cbind(Lag1,Lag2)[train,]
> test.X=cbind(Lag1,Lag2)[!train,]
> train.Direction=Direction[train]
> Direction.2005=Direction[!train]
> set.seed(1)
> knn.pred=knn(train.X,test.X,train.Direction,k=1)
> table(knn.pred,Direction.2005)
```

```
        Direction.2005
knn.pred Down Up
    Down   43 58
    Up     68 83
> mean(knn.pred==Direction.2005)
[1] 0.5
> knn.pred=knn(train.X,test.X,train.Direction,k=3)
> mean(knn.pred==Direction.2005)
[1] 0.54
```

Part 2:

The accuracy using Lags1-5 and Volume is 0.48. The accuracy using Lags1-2 is 0.56

```
> train=(Year<2005)
> Smarket.2005=Smarket[!train,]
> Direction.2005=Direction[!train]
> glm.fit=glm(Direction~Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume,
+             family=binomial,data=Smarket,subset=train)
> glm.probs=predict(glm.fit,newdata=Smarket.2005,type="response")
> glm.pred=rep("Down",252)
> glm.pred[glm.probs>.5]="Up"
> mean(glm.pred==Direction.2005)
[1] 0.48
> glm.fit2=glm(Direction~Lag1 + Lag2,
+             family=binomial,data=Smarket,subset=train)
> glm.probs2=predict(glm.fit2,newdata=Smarket.2005,type="response")
> glm.pred2=rep("Down",252)
> glm.pred2[glm.probs2>.5]="Up"
> mean(glm.pred2==Direction.2005)
[1] 0.56
```

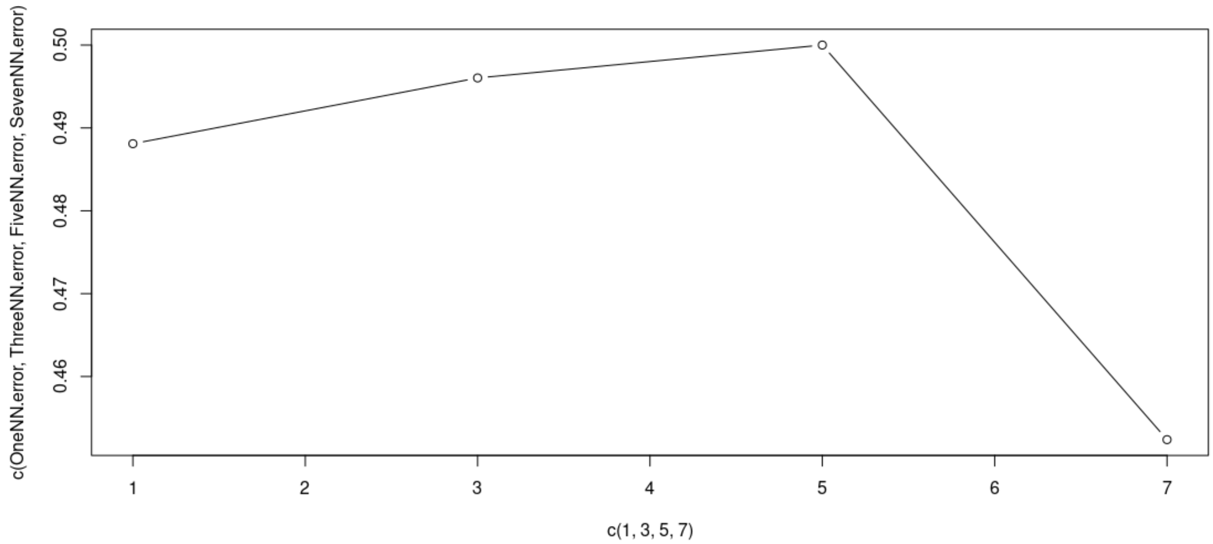
Part 3:

```
> ###########################################################
> # We can analyze th Smarket data in many ways.
> # Here we present 2 examples
> require(ISLR)
> Smarket=Smarket
> names(Smarket)
[1] "Year"      "Lag1"      "Lag2"      "Lag3"      "Lag4"      "Lag5"      "Volume"    "Today"
[9] "Direction"
> head(Smarket)
```

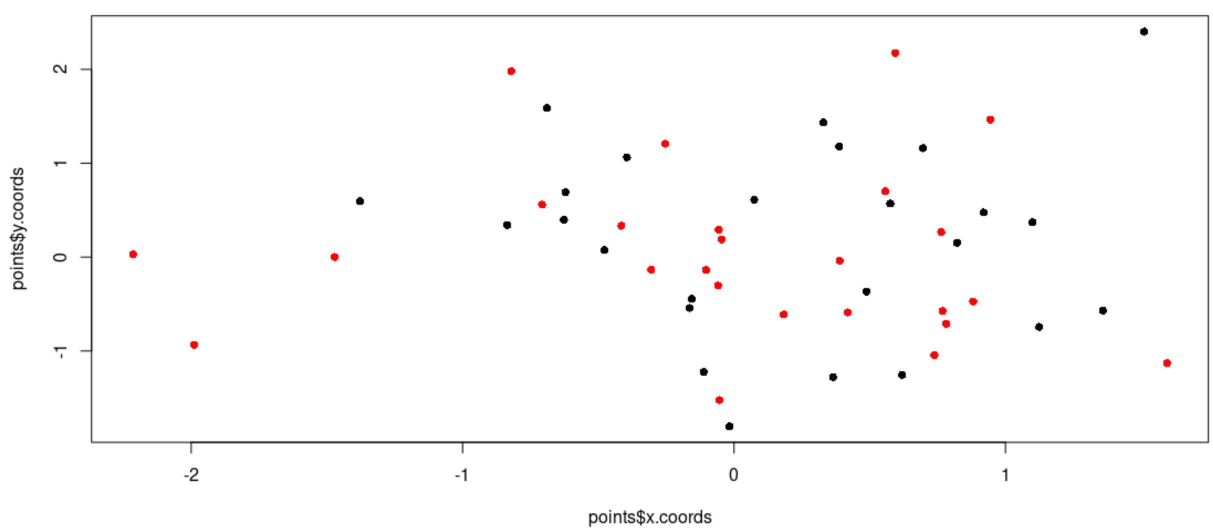
```
  Year  Lag1  Lag2  Lag3  Lag4  Lag5 Volume Today Direction
1 2001  0.38 -0.19 -2.62 -1.05  5.01    1.2  0.96        Up
2 2001  0.96  0.38 -0.19 -2.62 -1.05    1.3  1.03        Up
3 2001  1.03  0.96  0.38 -0.19 -2.62    1.4 -0.62      Down
4 2001 -0.62  1.03  0.96  0.38 -0.19    1.3  0.61        Up
5 2001  0.61 -0.62  1.03  0.96  0.38    1.2  0.21        Up
6 2001  0.21  0.61 -0.62  1.03  0.96    1.3  1.39        Up
> attach(Smarket)
The following objects are masked from Smarket (pos = 3):

    Direction, Lag1, Lag2, Lag3, Lag4, Lag5, Today, Volume, Year

The following objects are masked from Smarket (pos = 4):

    Direction, Lag1, Lag2, Lag3, Lag4, Lag5, Today, Volume, Year

The following objects are masked from Smarket (pos = 5):

    Direction, Lag1, Lag2, Lag3, Lag4, Lag5, Today, Volume, Year

The following objects are masked from Smarket (pos = 6):

    Direction, Lag1, Lag2, Lag3, Lag4, Lag5, Today, Volume, Year

The following objects are masked from Smarket (pos = 7):

    Direction, Lag1, Lag2, Lag3, Lag4, Lag5, Today, Volume, Year

The following objects are masked from Smarket (pos = 16):

    Direction, Lag1, Lag2, Lag3, Lag4, Lag5, Today, Volume, Year

The following objects are masked from Smarket (pos = 19):

    Direction, Lag1, Lag2, Lag3, Lag4, Lag5, Today, Volume, Year

> ##########################################################
> # Logistic Model
> logistic.fit=glm(Direction~Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume, family = binomial,
data=Smarket)
> summary(logistic.fit)

Call:
glm(formula = Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 +
```

```
    Volume, family = binomial, data = Smarket)

Deviance Residuals:
   Min      1Q  Median      3Q     Max
 -1.45   -1.20    1.07    1.15    1.33

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -0.12600    0.24074   -0.52     0.60
Lag1        -0.07307    0.05017   -1.46     0.15
Lag2        -0.04230    0.05009   -0.84     0.40
Lag3         0.01109    0.04994    0.22     0.82
Lag4         0.00936    0.04997    0.19     0.85
Lag5         0.01031    0.04951    0.21     0.83
Volume       0.13544    0.15836    0.86     0.39

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1731.2  on 1249  degrees of freedom
Residual deviance: 1727.6  on 1243  degrees of freedom
AIC: 1742

Number of Fisher Scoring iterations: 3

> pchisq(1731.2-1727.6, 1249-1243)
[1] 0.27
> names(logistic.fit)
 [1] "coefficients"      "residuals"         "fitted.values"     "effects"           "R"
 [6] "rank"              "qr"                "family"            "linear.predictors" "deviance"
[11] "aic"               "null.deviance"     "iter"              "weights"           "prior.weights"
[16] "df.residual"       "df.null"           "y"                 "converged"         "boundary"
[21] "model"             "call"              "formula"           "terms"             "data"
[26] "offset"            "control"           "method"            "contrasts"         "xlevels"
> pchisq(logistic.fit$null.deviance-logistic.fit$deviance, logistic.fit$df.null-logistic.fit$df.residual)
[1] 0.27
> train.data=subset(Smarket, Year<2005)
> test.data=subset(Smarket, Year==2005)
> train.mod=glm(Direction~Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume, family = binomial,
data=train.data)
> test.probs=predict(train.mod, test.data, type="response")
> head(test.probs)
 999 1000 1001 1002 1003 1004
0.53 0.52 0.52 0.51 0.50 0.50
> require(psych)
```

```
Loading required package: psych

Attaching package: 'psych'

The following objects are masked from 'package:VGAM':

    fisherz, logistic, logit

> describe(test.probs)
   vars   n mean   sd median trimmed  mad  min  max range skew kurtosis se
X1    1 252 0.49 0.01   0.49    0.49 0.01 0.44 0.53  0.09 -0.4      0.2  0
> pred.directions.test=ifelse(test.probs<=0.5, "Down", "Up")
> head(pred.directions.test)
   999   1000   1001   1002   1003   1004
  "Up"   "Up"   "Up"   "Up" "Down"   "Up"
> mean(pred.directions.test==test.data$Direction)
[1] 0.48
> table(pred.directions.test, test.data$Direction)

pred.directions.test Down Up
                Down   77 97
                Up     34 44
> ########################################################
> # KNN Approach
> require(class)
> knn1.fit=knn(train.data[,c(2:7)], test.data[,c(2:7)], train.data$Direction, 1)
> head(knn1.fit)
[1] Down Up   Down Down Down Down
Levels: Down Up
> OneNN.error = 1 - mean(knn1.fit==test.data$Direction);OneNN.error
[1] 0.49
> # Consider higher values of k
> knn3.fit=knn(train.data[,c(2:7)], test.data[,c(2:7)], train.data$Direction, 3)
> ThreeNN.error = 1 - mean(knn3.fit==test.data$Direction); ThreeNN.error
[1] 0.5
> knn5.fit = knn(train.data[,c(2:7)], test.data[,c(2:7)], train.data$Direction, 5)
> FiveNN.error = 1 - mean(knn5.fit==test.data$Direction); FiveNN.error
[1] 0.5
> knn7.fit = knn(train.data[,c(2:7)], test.data[,c(2:7)], train.data$Direction, 7)
> SevenNN.error = 1 - mean(knn7.fit==test.data$Direction); SevenNN.error
[1] 0.45
> plot(c(1,3,5,7), c(OneNN.error, ThreeNN.error, FiveNN.error, SevenNN.error), type="b")
```

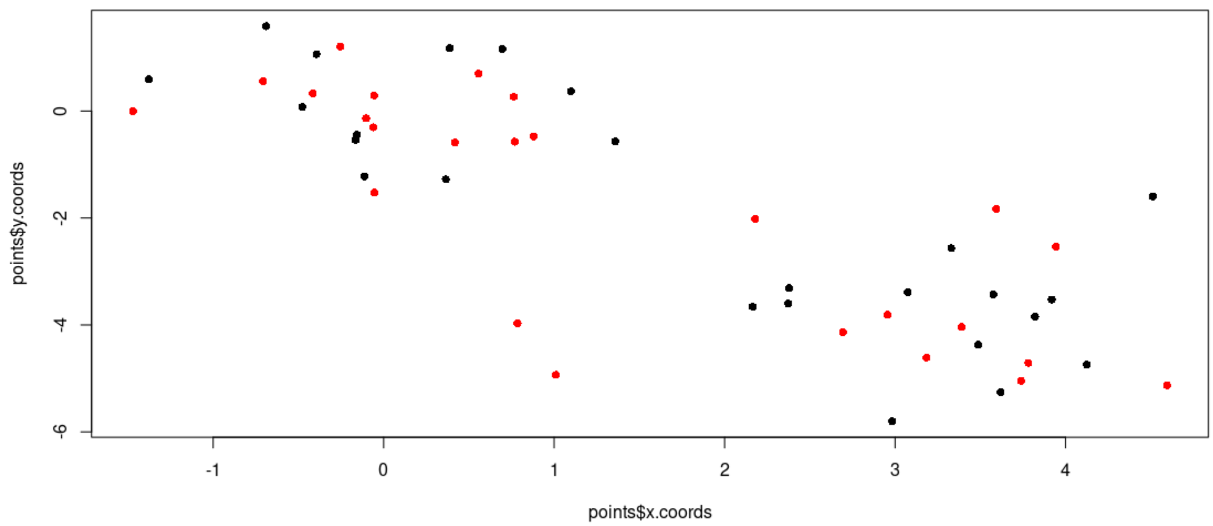



```
> ##########################################################
> # K-means clusters with simulated data
> set.seed(1)
> x.coords=(rnorm(50))
> y.coords=(rnorm(50))
> group.num=c(1,2)
> points=data.frame(cbind(group.num, x.coords, y.coords))
> View(points)
> plot(points$x.coords, points$y.coords, col=group.num, pch=16)
```

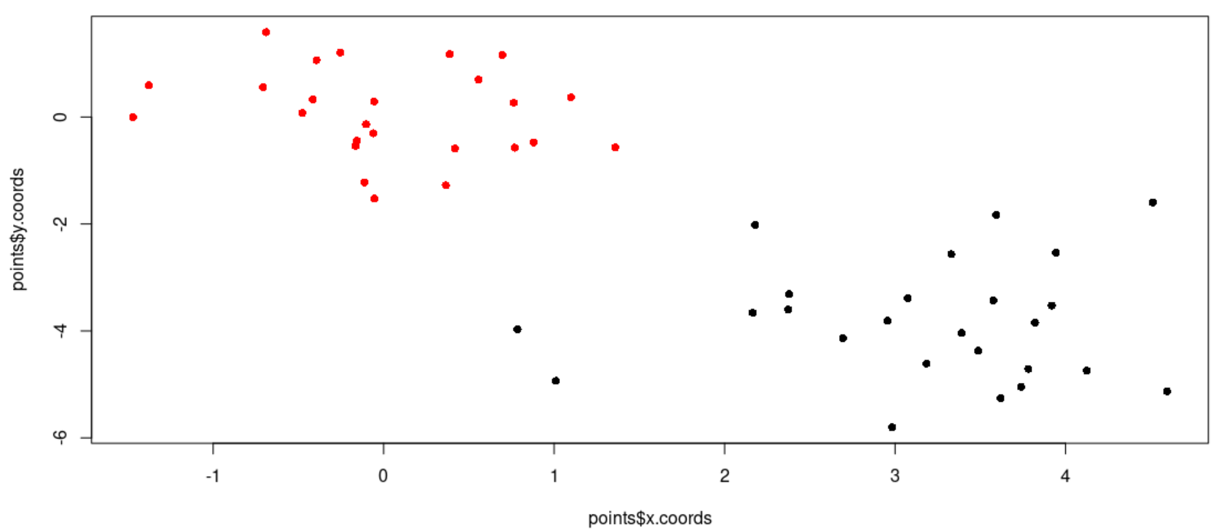



```
> points$x.coords[c(1:25)] = points$x.coords[c(1:25)]+3
> points$y.coords[c(1:25)] = points$y.coords[c(1:25)]-4
```

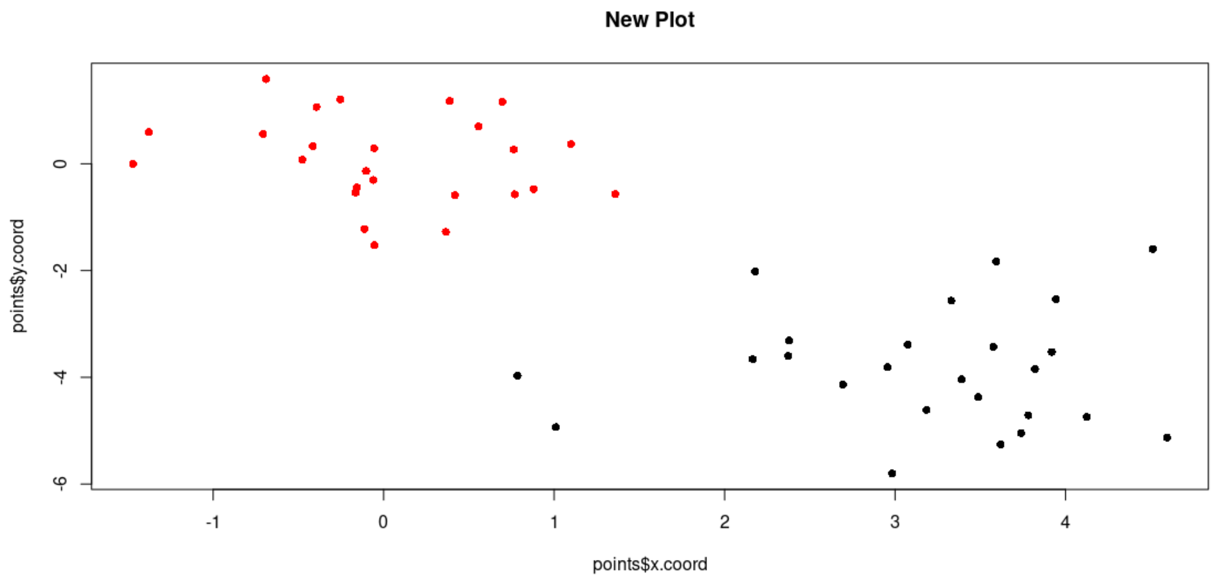
```
> plot(points$x.coords, points$y.coords, col=points$group.num, pch=16)
```



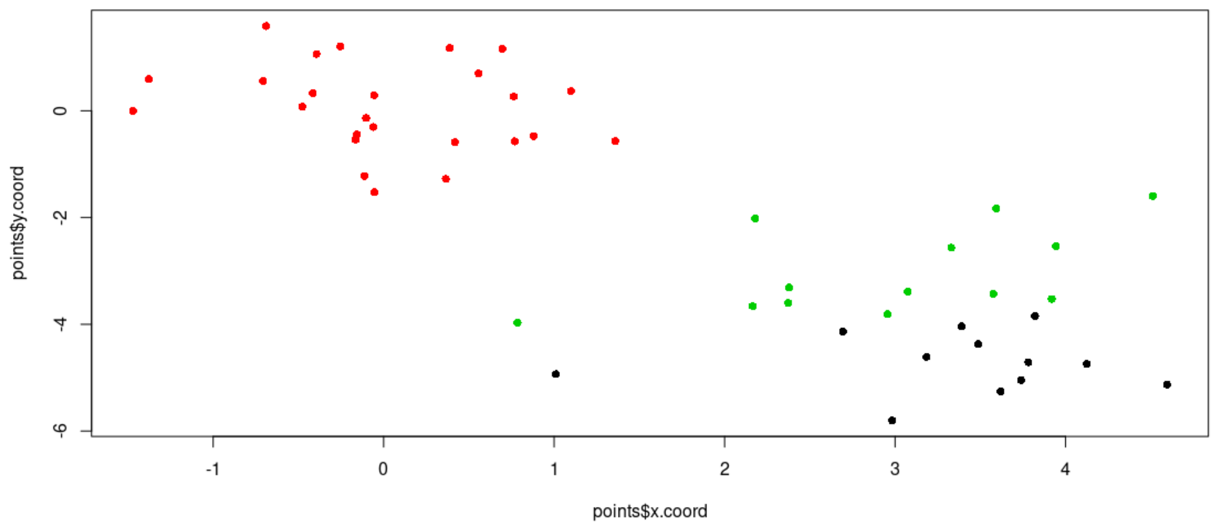

```
> points$group.num[c(1:25)]=1
> points$group.num[c(26:50)]=2
> plot(points$x.coords, points$y.coords, col=points$group.num, pch=16)
```




```
> km.out=kmeans(points, 2, nstart=20)
> plot(points$x.coord, points$y.coord, col=km.out$cluster, pch=16, main="New Plot")
```




```
> names(km.out)
[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"
[7] "size"         "iter"         "ifault"
> km.out$cluster
 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
2
> km.out$totss
[1] 405
> km.out$withinss
[1] 51 29
> km.out$tot.withinss
[1] 80
> km.out$betweenss
[1] 326
> right = mean(points$group.num==km.out$cluster); c(right, 1-right)
[1] 1 0
> # Consider 3 clusters
> km3.out=kmeans(points, 3, nstart=20)
> plot(points$x.coord, points$y.coord, col=km3.out$cluster, pch=16)
```




```
> c(km.out$totss, km.out$tot.withinss, km.out$betweenss)
[1] 405  80 326
> c(km3.out$totss, km3.out$tot.withinss, km3.out$betweenss)
[1] 405  61 345
```

Part 4:

K = 7 provides the lowest error. Based on the size of the dataset, 7 is also sufficiently small to be usable as a good value of K.