Assignment-2

Yizhe Qu

February 9, 2019

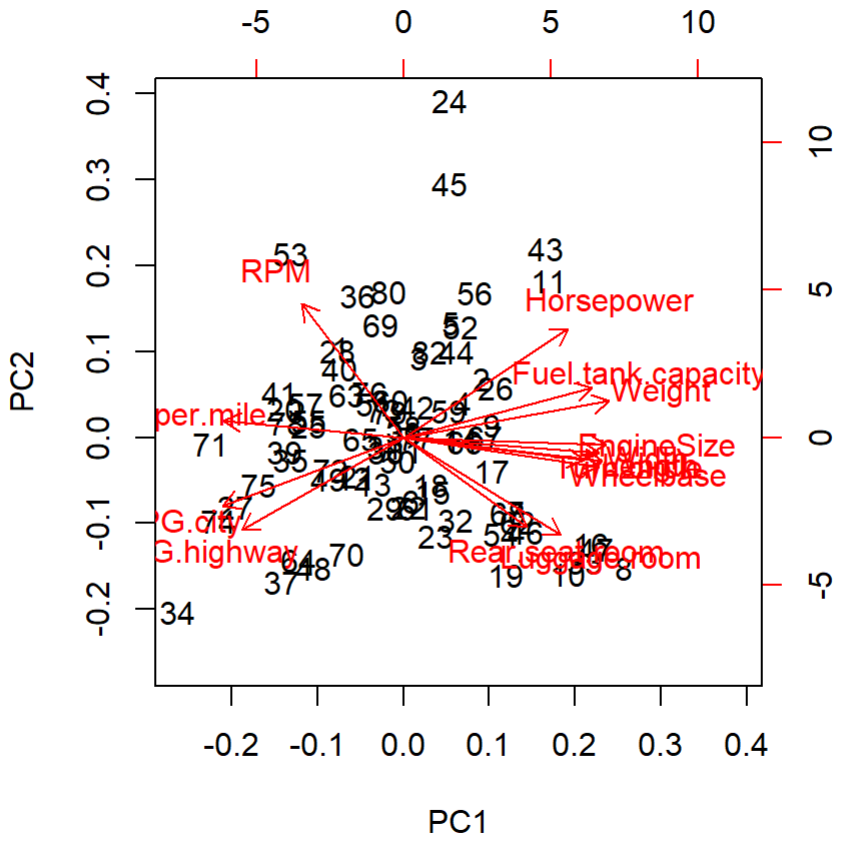
Q1

a.

```
card <- read.csv("C:/Users/yizhe/Desktop/MDS/Term4/data_572/data/car93.csv", stringsAsFactors =
FALSE)
pcard <- prcomp(as.matrix(card[,-c(1,2,3,4)]), scale.=TRUE)
summary(pcard)
```

```
## Importance of components:
##                           PC1    PC2     PC3     PC4     PC5     PC6
## Standard deviation     3.097 1.2112 0.94592 0.76113 0.59294 0.54436
## Proportion of Variance 0.685 0.1048 0.06391 0.04138 0.02511 0.02117
## Cumulative Proportion  0.685 0.7898 0.85374 0.89512 0.92023 0.94140
##                           PC7     PC8     PC9    PC10    PC11    PC12
## Standard deviation     0.48674 0.44913 0.34512 0.30099 0.26044 0.20876
## Proportion of Variance 0.01692 0.01441 0.00851 0.00647 0.00484 0.00311
## Cumulative Proportion  0.95832 0.97273 0.98124 0.98771 0.99255 0.99566
##                           PC13    PC14
## Standard deviation     0.19860 0.14579
## Proportion of Variance 0.00282 0.00152
## Cumulative Proportion  0.99848 1.00000
```

```
biplot(pcard)
```



b. The first principal component is made up of a relatively equal weight of all the variables. However, MPG ratings, RPM, and Rev.per.mile are all negatively associated with the first component. We can somewhat view this component as a measure of the `size' of the car, since large cars will have high scores on all the positively correlated variables and smaller scores on the four negatively associated variables.

c. The second principal component is a bit less clear. We have a high positive loading of price, horsepower, and RPM, along with large negative loadings for MPG and rear seats, luggage. This suggests relatively small, expensive cars with strong engines. It seems we could potentially interpret this as a measure of "luxuriousness".

d.

e. Kaiser criterion states that all PCs where lambda > 1 should be kept. In this case, PC1 and PC2 should be kept.

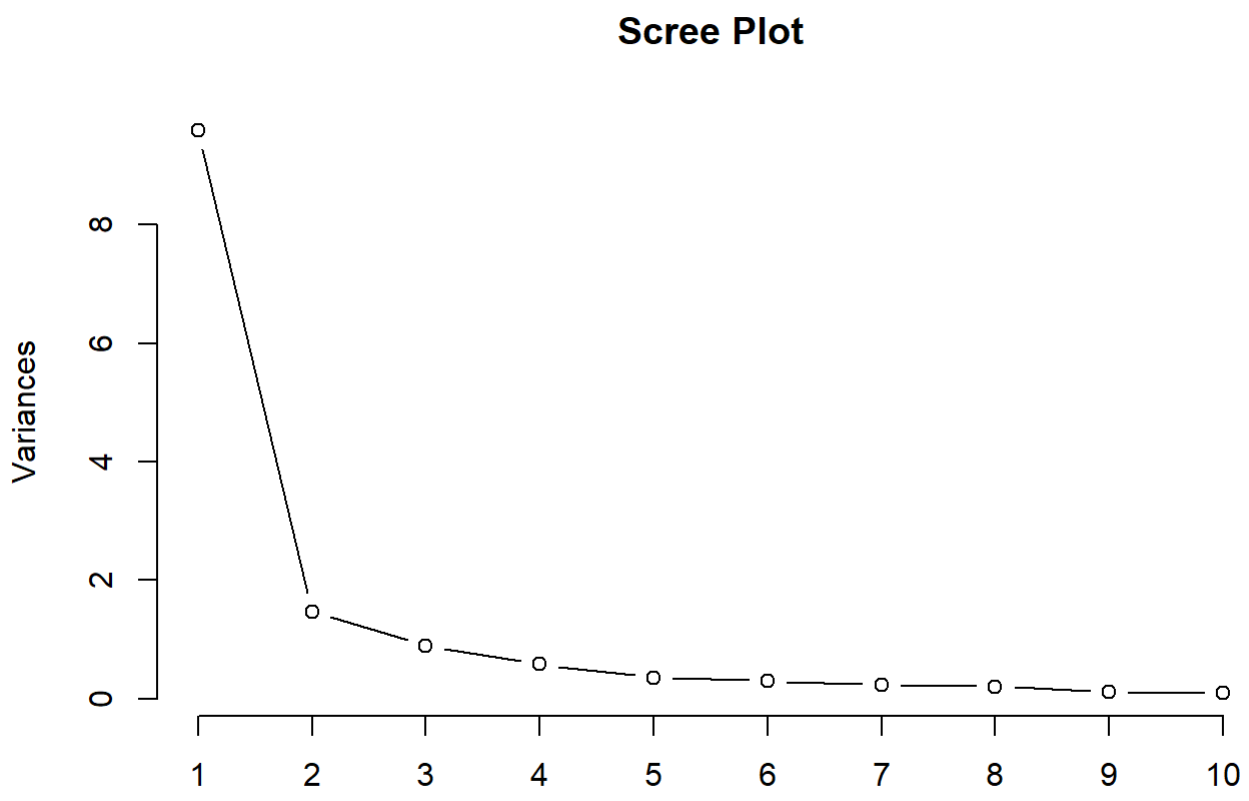
ii. We can keep the first 4 principal components to retain at least 90% of the variance in the data.

```
cum_var <- cumsum(pcard$sdev^2/sum(pcard$sdev^2))
cum_var
```

```
##  [1] 0.6850394 0.7898256 0.8537371 0.8951171 0.9202301 0.9413967 0.9583193
##  [8] 0.9727279 0.9812358 0.9877069 0.9925517 0.9956646 0.9984818 1.0000000
```

iii. According to scree plot, we keep the first two principal components since the "elbow" is found at PC2.

```
plot(pcard, type="lines", main="Scree Plot")
```



e.

f.

```
pccomps <- pcard$x[,1:2]
small <- card$Type=="Small"
cvres <- list()
predprob <- NA
for(i in 1:nrow(pccomps)){
  dum <- data.frame(small=small[-i], PC1=pccomps[-i,1], PC2=pccomps[-i,2])
  cvres[[i]] <- glm(small~., data=dum, family="binomial")
  predprob[i] <- predict(cvres[[i]], newdata=data.frame(pccomps), type="response")[i]
}
library(MLmetrics)
```

```
## Warning: package 'MLmetrics' was built under R version 3.5.2
```

```
##
## Attaching package: 'MLmetrics'
```

```
## The following object is masked from 'package:base':
##
##     Recall
```

```
table(small, predprob>0.5)
```

```
##
## small   FALSE TRUE
##   FALSE    58    3
##   TRUE      4   17
```

```
LogLoss(predprob, small)
```

```
## [1] 0.2496668
```

ii.

```
library(MASS)
```

```
## Warning: package 'MASS' was built under R version 3.5.2
```

```
ldabin <- lda(small~pccomps, CV=TRUE)
table(small, ldabin$class)
```

```
##
## small   FALSE TRUE
##   FALSE    59    2
##   TRUE      4   17
```

```
LogLoss(ldabin$posterior[,2], small)
```

```
## [1] 0.2224975
```

iii.

```
ldamult <- lda(card$Type~pccomps, CV=TRUE)
table(card$Type, ldamult$class)
```

```
##
##           Compact Large Midsize Small Sporty
##   Compact      13     0       2     0      1
##   Large         0     9       2     0      0
##   Midsize       4     2      15     0      1
##   Small         2     0       0    16      3
##   Sporty        3     0       3     3      3
```

```
MultiLogLoss(ldamult$posterior, card$Type)
```

```
## [1] 0.8001237
```

f. The fact that LDA and Logistic regression can find the structure of Small vs Not-Small cars in the first two fits suggests to some extent that our interpretation of the first component (size of car) may well hold true. In terms of LDA using all car types, we can see Compact, Small, Midsize, and Large all being largely distinguishable through the first two components. The only category of car that is poorly classified is Sporty.

Q2

a.

```
library(nnet)
car_numeric <- card[,-c(1,2,3)]
scar <- apply(car_numeric, 2, function(v) (v-min(v))/(max(v)-min(v)))
set.seed(4521)
nncar <- nnet(Price~., data=scar, size=5)
```

```
## # weights:  81
## initial  value 29.930870
## final  value 6.529868
## converged
```

```
res <- predict(nncar)
mse <- mean((scar[,1]-res)^2)
mse
```

```
## [1] 0.07963253
```

b.

```
set.seed(217)
ind <- sample(1:nrow(scar), 41)
train <- scar[ind,]
test <- scar[-ind,]
nncar <- nnet(Price~., data=train, size=5)
```

```
## # weights:  81
## initial  value 3.490432
## iter  10 value 0.777098
## iter  20 value 0.381318
## iter  30 value 0.187845
## iter  40 value 0.096033
## iter  50 value 0.074861
## iter  60 value 0.056126
## iter  70 value 0.043923
## iter  80 value 0.030166
## iter  90 value 0.024096
## iter 100 value 0.020959
## final  value 0.020959
## stopped after 100 iterations
```

```
yhat <- predict(nncar, test[,-1])
mse <- mean((yhat-test[,1])^2)
mse
```

```
## [1] 0.01140137
```

Not marked (or bonus if the student notices the same thing and provides at least some further investigation): Note there is something funny going on here. In fact, our train test setup is providing better results than our original run on the full data.time for some investigation, after playing around with some settings, we can get.

```
set.seed(4521)
nncar3 <- nnet(Price~., data=scar, size=2, maxit=3000, linout=TRUE, trace=FALSE)
res <- predict(nncar3)
mse <- mean((scar[,1]-res)^2)
mse
```

```
## [1] 0.001586173
```

```
nncar4 <- nnet(Price~., data=train, size=2, maxit=3000, linout=TRUE, trace=FALSE)
yhat2 <- predict(nncar, test[,-1])
mse <- mean((yhat2-test[,1])^2)
mse
```

```
## [1] 0.01140137
```

Which makes more sense. I believe the predominant issue was the number of weights we were estimating given quite a small sample size (my apologies for setting up the question that way).

c. Now we first need to transfer back to the original scale, then to the dollars unit. I will provide the answer both in terms of sqrt(MSE) and MAE (mean absolute error), and I will use the original runs even though they don't make much sense. Square root of the MSE does not provide you "on average, how far off is your model," but the important practice I'm looking for in this question is transforming back to the original scale.

```
#first get my predictions back to the original scale
osyhat <- yhat*(max(card$Price)-min(card$Price)) + min(card$Price)
#then to the dollars unit
osyhat <- 1000*osyhat
#then use the indexing for my testing set on the original price
osy <- card$Price[-ind]*1000
#now calculate sqrt(MSE) for this...
sqrt(mean((osyhat-osy)^2))
```

```
## [1] 5819.358
```

```
#now for the better version
mean(abs(osyhat-osy))
```

```
## [1] 3589.563
```