Assignment-2-573-chi

Chi

2019-03-16

Question 1

a)

```
library(mlbench)
data(HouseVotes84)

HV <- data.frame(lapply(HouseVotes84,as.character),stringsAsFactors=FALSE)
HV[is.na(HV)] <- 'NoVote'

# change to factor
HV <- as.data.frame(unclass(HV))
head(HV)
```

```
##        Class     V1 V2 V3     V4     V5 V6 V7 V8 V9 V10    V11    V12 V13
## 1 republican      n  y  n      y      y  y  n  n  n   y NoVote      y   y
## 2 republican      n  y  n      y      y  y  n  n  n   n      n      y   y
## 3   democrat NoVote  y  y NoVote      y  y  n  n  n   n      y      n   y
## 4   democrat      n  y  y      n NoVote  y  n  n  n   n      y      n   y
## 5   democrat      y  y  y      n      y  y  n  n  n   n      y NoVote   y
## 6   democrat      n  y  y      n      y  y  n  n  n   n      n      n   y
##   V14 V15    V16
## 1   y   n      y
## 2   y   n NoVote
## 3   y   n      n
## 4   n   n      y
## 5   y   y      y
## 6   y   y      y
```

b)

```
library(StatMatch)
```

```
## Loading required package: proxy
```

```
##
## Attaching package: 'proxy'
```

```
## The following objects are masked from 'package:stats':
##
##     as.dist, dist
```

```
## The following object is masked from 'package:base':
##
##     as.matrix
```

```
## Loading required package: clue
```

```
## Loading required package: survey
```

```
## Loading required package: grid
```

```
## Loading required package: Matrix
```

```
## Loading required package: survival
```

```
##
## Attaching package: 'survey'
```

```
## The following object is masked from 'package:graphics':
##
##     dotchart
```

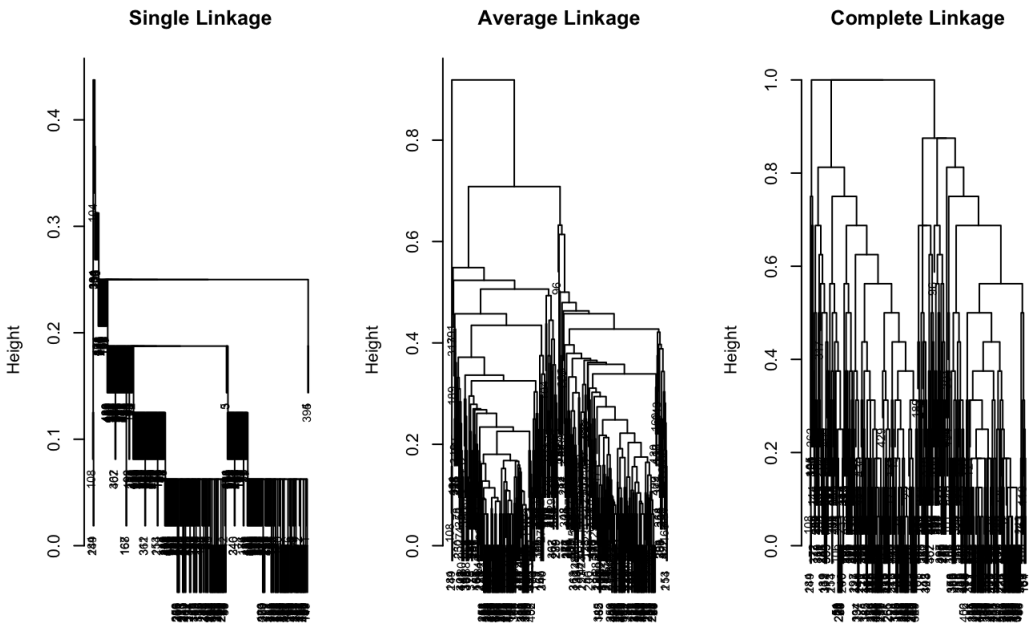
```
## Loading required package: RANN
```

```
## Loading required package: lpSolve
```

```
dat <- HV[c(-1)]

#gow.matrix <- gower.dist(dat) #this will not work since hclust doesnt take this type
#gow.matrix <- as.matrix(gow.matrix)

# Lets try daisy gower distance instead
library(cluster)
gower.dist <- daisy(dat, metric = "gower")
```

c)

```
single <- hclust(gower.dist, method ="single")
average <- hclust(gower.dist, method ="average")
complete <- hclust(gower.dist, method ="complete")

par(mfrow =c(1,3))
plot(single , main= "Single Linkage", xlab="", sub ="",cex =.7)
plot(average , main = "Average Linkage", xlab="", sub ="",cex =.7)
plot(complete ,main = "Complete Linkage", xlab="", sub ="",cex =.7)
```



So lets probably take the Average or complete linkage clustering, but it seems like we can disregard complete linkage as well since it will be cut at its largest jump with three clusters.

In addition, single linkage has quite a lot of chaining so we dont pick.

check misclassification to make sure we are picking the correct one since we do have the response

```
table(HV$Class,cutree(average,2))
```

```
##
##                1   2
##   democrat   266   1
##   republican 166   2
```

```
#hooray it cannot be three because we only have democrats and republicans lets pick average linkage and cut at 2!
table(HV$Class,cutree(complete, 3) )
```

```
##
##                1   2   3
##   democrat    58 204   5
##   republican 160   5   3
```

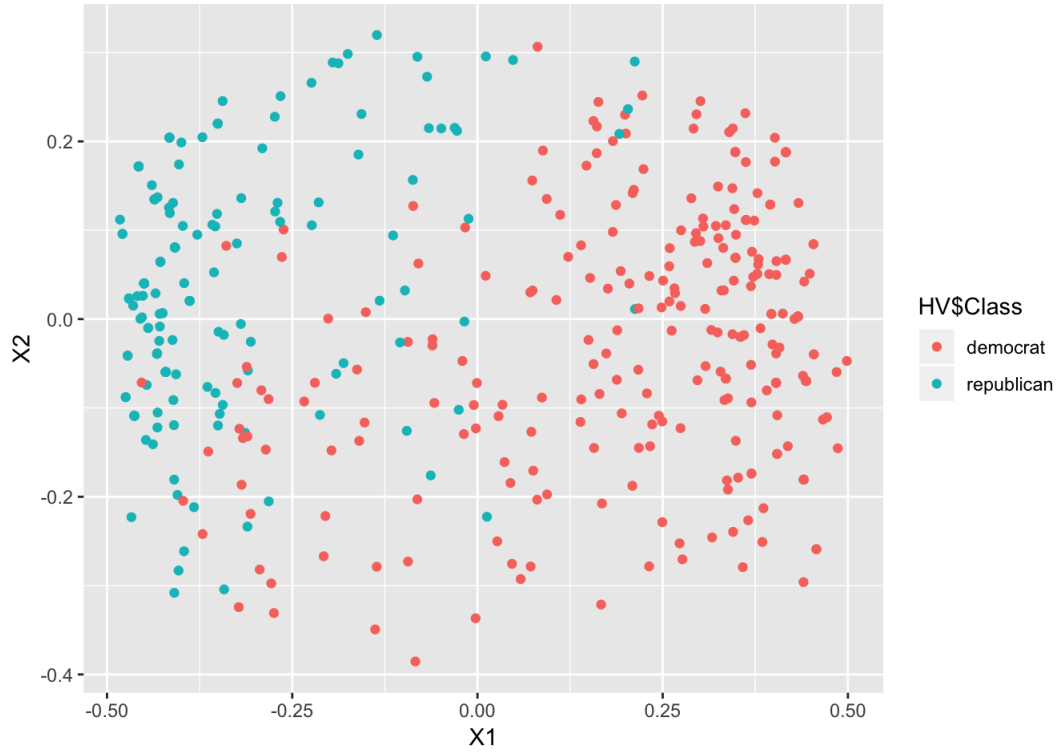
So as mentioned we will pick Average linkage with a cut at 2 at its biggest jump.

It also makes more sense to pick average linkage since it cuts at 2 and we know our labels are republicans and democrats

The misclassification is (166+1)/(435) = 0.384

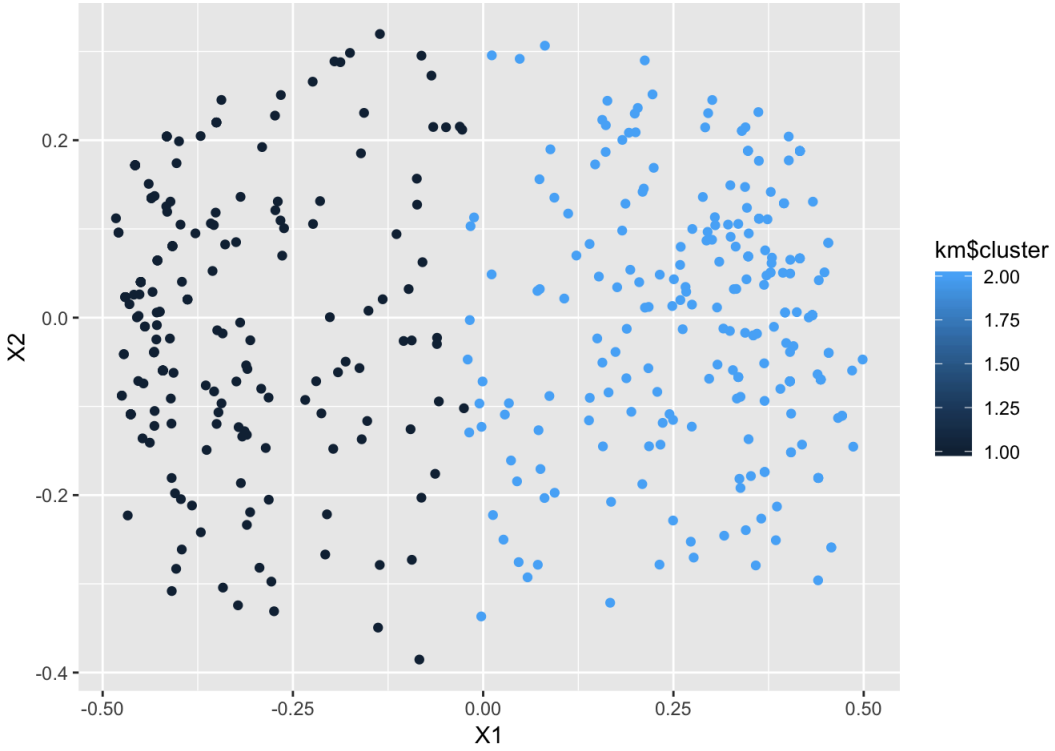
d)

```
library(ggplot2)
MDS <- cmdscale(gower.dist,eig=TRUE, k=2)
MDSframe <- data.frame(MDS$points)
ggplot(MDSframe, aes(X1, X2)) + geom_point(aes(fill=HV$Class,col=HV$Class))
```



e)

```
library(ggplot2)
km <- kmeans(as.matrix(MDS$points), 2)
ggplot(MDSframe, aes(X1, X2)) + geom_point(aes(fill=km$cluster,col=km$cluster))
```




```
table(HV$Class, km$cluster)
```

```
##
##                1   2
##   democrat    42 225
##   republican 159   9
```

f)

```
library(mclust)
```

```
## Package 'mclust' version 5.4.2
## Type 'citation("mclust")' for citing this R package in publications.
```

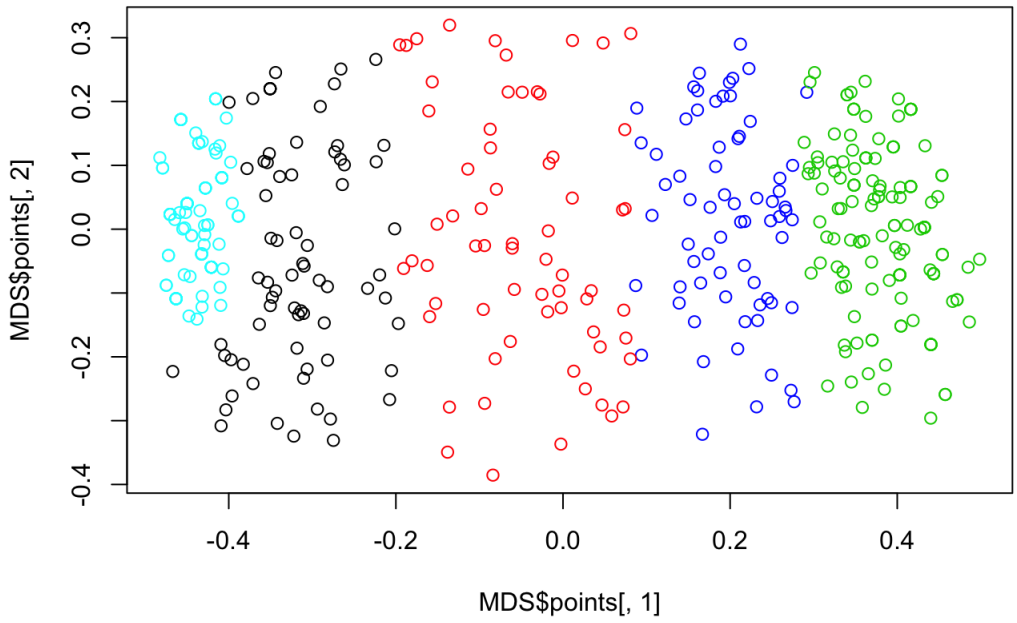
```
library(mvtnorm)
mixclust <- Mclust(scale(MDS$points))

#plot(mclustBIC(MDS$points)) # could look at this to see plot similar to a scree plot
summary(mixclust$BIC) #suggest 5 clusters
```

```
## Best BIC values:
##              VEI,5        VEE,5       VVI,5
## BIC      -2119.481 -2121.041469 -2124.74433
## BIC diff     0.000    -1.560895    -5.26376
```

```
table(HV$Class, mixclust$classification) #classification table
```

```
##
##                1   2   3   4   5
##   democrat    26  38 138  64   1
##   republican  45  27   0   4  92
```

```
plot(MDS$points[,1],MDS$points[,2], col = mixclust$classification) # 2D scatter plot coloured by group membe
rship
```



So BIC suggets 5 clusters.

Question 2

a)

```
AC<- ability.cov$cov
ACfac <- factanal(covmat = AC, factors = 1,rotation = "none", n.obs = 112)
ACfac
```

```
##
## Call:
## factanal(factors = 1, covmat = AC, n.obs = 112, rotation = "none")
##
## Uniquenesses:
## general picture  blocks    maze reading   vocab
##   0.535   0.853   0.748   0.910   0.232   0.280
##
## Loadings:
##         Factor1
## general 0.682
## picture 0.384
## blocks  0.502
## maze    0.300
## reading 0.877
## vocab   0.849
##
##                Factor1
## SS loadings      2.443
## Proportion Var   0.407
##
## Test of the hypothesis that 1 factor is sufficient.
## The chi square statistic is 75.18 on 9 degrees of freedom.
## The p-value is 1.46e-12
```

This is not suitable since the p-value is 1.46e-12, which means that we have to reject the null hypothesis: assuming 1 factor is sufficient.

However, we want to fail to reject null hypothesis to have a suitable model!

In addition the uniqueness row has quite high values => the variance not explained in that particular variable

Lastly, we notice that the variance explained of the factor is only 0.407. This is probaby not sufficient enough to explain the data. By adding more factors the cummulative distribution may increase to a point where we think is enough of variance explained.

b)

```
ACcorfac <- factanal(covmat = cov2cor(AC), factors = 1,rotation = "none", n.obs=112)
ACcorfac
```

```
##
## Call:
## factanal(factors = 1, covmat = cov2cor(AC), n.obs = 112, rotation = "none")
##
## Uniquenesses:
## general picture  blocks    maze reading   vocab
##   0.535   0.853   0.748   0.910   0.232   0.280
##
## Loadings:
##         Factor1
## general 0.682
## picture 0.384
## blocks  0.502
## maze    0.300
## reading 0.877
## vocab   0.849
##
##                Factor1
## SS loadings      2.443
## Proportion Var   0.407
##
## Test of the hypothesis that 1 factor is sufficient.
## The chi square statistic is 75.18 on 9 degrees of freedom.
## The p-value is 1.46e-12
```

Are there any differences in the model to the previous question?

No differences because factor analysis is scale invariant!

So as mentioned in part a) we reject the null hypothesis that 1 factor is sufficient.

c)

```
ACfac2 <- factanal(covmat = AC, factors = 2,rotation = "none", n.obs = 112)
ACfac2
```

```
##
## Call:
## factanal(factors = 2, covmat = AC, n.obs = 112, rotation = "none")
##
## Uniquenesses:
## general picture  blocks    maze reading   vocab
##   0.455   0.589   0.218   0.769   0.052   0.334
##
## Loadings:
##         Factor1 Factor2
## general  0.648   0.354
## picture  0.347   0.538
## blocks   0.471   0.748
## maze     0.253   0.408
## reading  0.964  -0.135
## vocab    0.815
##
##                Factor1 Factor2
## SS loadings      2.420   1.162
## Proportion Var   0.403   0.194
## Cumulative Var   0.403   0.597
##
## Test of the hypothesis that 2 factors are sufficient.
## The chi square statistic is 6.11 on 4 degrees of freedom.
## The p-value is 0.191
```

Two factors is enough since the pvalue 0.191 is > 0.05. This tells us that we fail to reject the null hypothesis and that 2 factors is sufficient. Other than that, our cummulative distribution increased to 0.597 and our row of uniqueness also decreased quite a lot in each of the

variables.

Explanation of factors:

Description of data: Six tests were given to 112 individuals. The covariance matrix is given in this object.

Factor 1 scores high on reading and vocab (and maybe general). We can probably explain this as the English ability.

Factor 2 scores high on blocks, which probably means ability with block design (maybe logic thinking).

d)

```
ACfacVar <- factanal(covmat = AC, factors = 2,rotation = "varimax", n.obs = 112)
ACfacVar
```

```
##
## Call:
## factanal(factors = 2, covmat = AC, n.obs = 112, rotation = "varimax")
##
## Uniquenesses:
## general picture  blocks    maze reading   vocab
##   0.455   0.589   0.218   0.769   0.052   0.334
##
## Loadings:
##         Factor1 Factor2
## general 0.499   0.543
## picture 0.156   0.622
## blocks  0.206   0.860
## maze    0.109   0.468
## reading 0.956   0.182
## vocab   0.785   0.225
##
##                Factor1 Factor2
## SS loadings      1.858   1.724
## Proportion Var   0.310   0.287
## Cumulative Var   0.310   0.597
##
## Test of the hypothesis that 2 factors are sufficient.
## The chi square statistic is 6.11 on 4 degrees of freedom.
## The p-value is 0.191
```

Which elements of the output change? (The factor loadings have changed)

The “varimax” rotation reduces the loadings of some factors and provide more interpretable factors. As explained in lecture it seeks for a gamma that loads heavily on only one factor.

Explanation of factors:

Factor 1 scores high on reading (decreased a little) and vocab (decreased a little). Probably still explaining the English ablity

Factor 2 scores high on blocks (increased). Probaly still explaining the ability to block design (logic thinking).

So what we see is that most of the factors that had high value without the varimax rotations, are still around the same but factor 2 increased significantly for all the loadings.

e)

```
ACfacPro <- factanal(covmat = AC, factors = 2,rotation = "promax", n.obs = 112)
ACfacPro
```

```
##
## Call:
## factanal(factors = 2, covmat = AC, n.obs = 112, rotation = "promax")
##
## Uniquenesses:
## general picture  blocks    maze reading   vocab
##   0.455   0.589   0.218   0.769   0.052   0.334
##
## Loadings:
##         Factor1 Factor2
## general  0.364   0.470
## picture          0.671
## blocks           0.932
## maze             0.508
## reading  1.023
## vocab    0.811
##
##                Factor1 Factor2
## SS loadings      1.853   1.807
## Proportion Var   0.309   0.301
## Cumulative Var   0.309   0.610
##
## Factor Correlations:
##         Factor1 Factor2
## Factor1   1.000   0.557
## Factor2   0.557   1.000
##
## Test of the hypothesis that 2 factors are sufficient.
## The chi square statistic is 6.11 on 4 degrees of freedom.
## The p-value is 0.191
```

What element is added to the output? the Factor Correlations

What assumptions have we relaxed that necessitates that output? We allow correlation among the factors => relaxed the assumption that the 2 factors are orthogonal to each other.

Note: Promax is a non-orthogonal rotation (called oblique), popular for increasing the amount of variance explained by small q (allows correlation among factors)

Explanation of Factors:

Factor 1: Now it is very clear that Factor 1 explains the ability in English in terms of scoring high on vocabulary tests and reading comprehension tests.(reading and vocab loadings increased significantly)

Factor 2: It is also clear to us Factor two is most likely explaining the ability to block design (logical thinking). It is slighly possible that being well on the block also implies pretty ok with picutre completion (another logical test?). (blocks loading increased significantly)

Question 3)

```
# from lab
#####Get images
to.read = file("t10k-images-idx3-ubyte", "rb")
readBin(to.read, integer(), n=4, endian="big")
```

```
## [1]  2051 10000    28    28
```

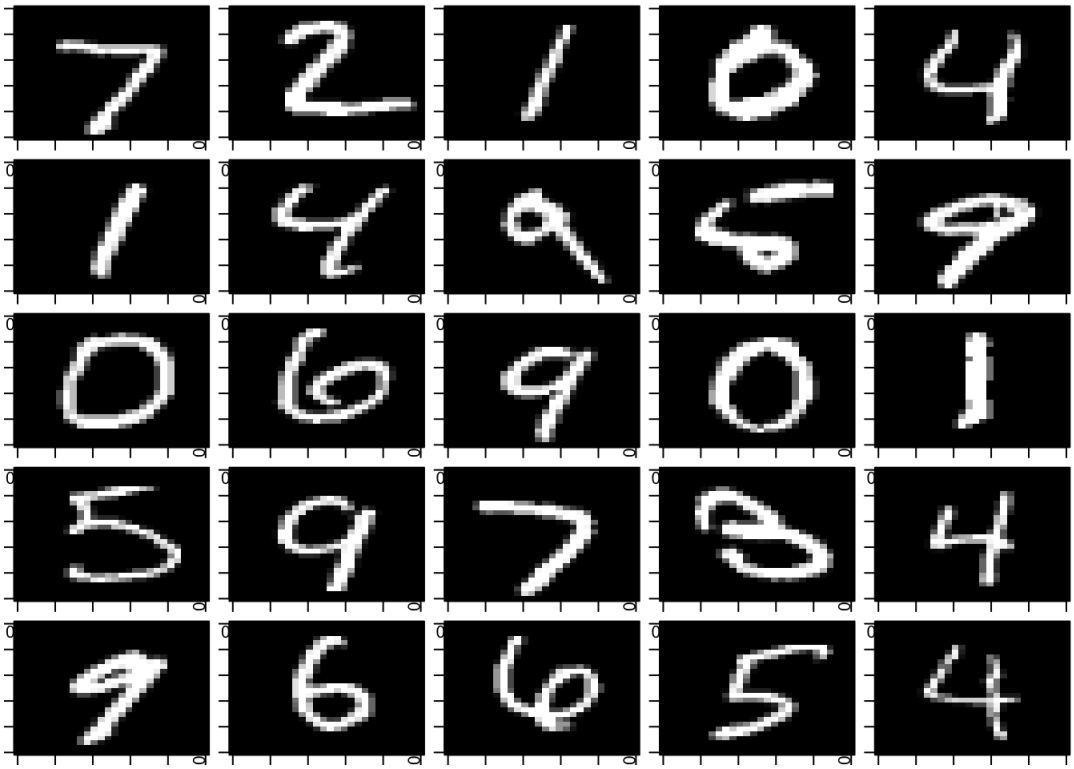
```
####Build an image array (like the pain data)
imarr <- array(0, dim=c(28,28,10000))
for(i in 1:10000){
  imarr[,,i] <- matrix(readBin(to.read,integer(), size=1, n=28*28, endian="big", signed="F"),28,28)[,28:1]
}
close(to.read)
####Build a flattened image matrix
immat <- t(apply(imarr, 3, as.vector))

####Get Labels
lab.read <- file("t10k-labels-idx1-ubyte", "rb")
readBin(lab.read, 'integer', n = 1, size = 4, endian = 'big')
```

```
## [1] 2049
```

```
n = readBin(lab.read, 'integer', n = 1, size = 4, endian = 'big')
labls = readBin(lab.read, 'integer', n = n, size = 1, signed = FALSE)
close(lab.read)
```

a) Provide a plot of the first 25 images in the data

```
par(mfrow=c(5,5))
par(mar=c(.5,.5,.5,.5))

for (i in 1:25){
  image(imarr[,,i], col = gray((0:32)/32))
  }
```



b) Run principal components (with scaling) on the images. What is the maximum number of components are permittable?

```
# scale. = FALSE since there  are  0's
prnum<- prcomp(immat) #principal component of the number images

dim(prnum$rotation)
```

```
## [1] 784 784
```

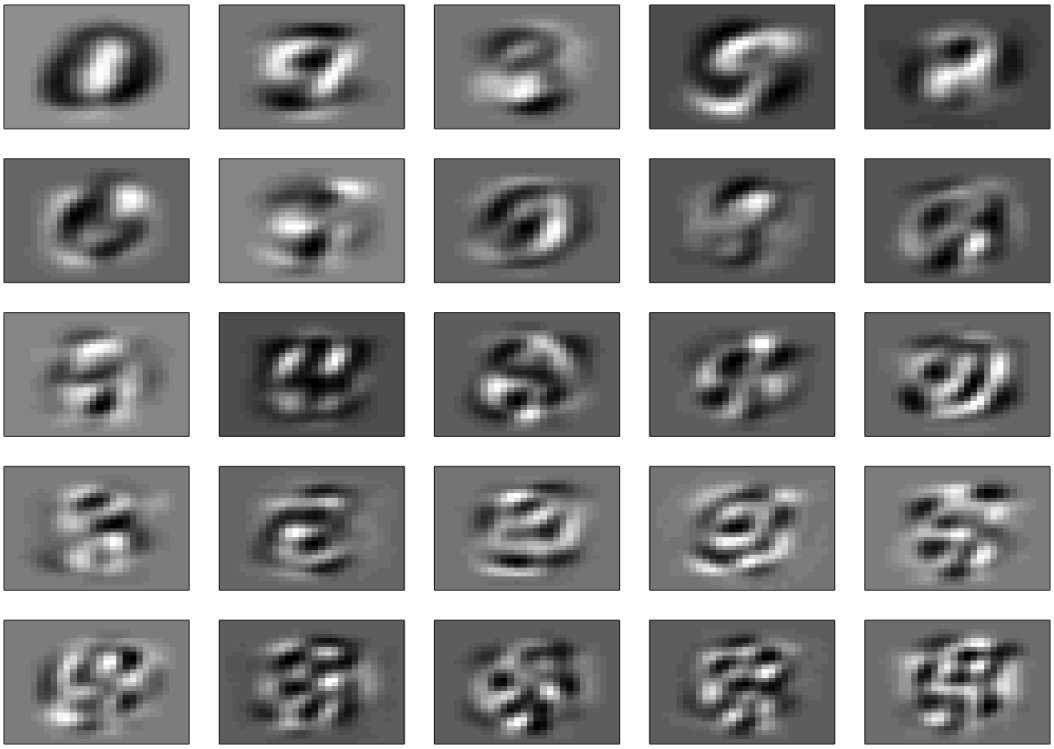
```
dim(immat)
```

```
## [1] 10000   784
```

The dimension of pca$rotation and immat tells us that the maximum permittable number of components is 784 .

c) Plot the first 25 resulting eigenvectors as images. What percentage of the original variation in the pixels is explained by the first 25 PCs?

```
par(mfrow=c(5,5), omi=c(0,0,0,0), mai=c(0.1,0.1,0.1,0.1))

for (i in 1:25){
  image(matrix(prnum$rotation[,i], 28,28), col = gray((0:32)/32), xaxt="n", yaxt="n")
  }
```



continue part c) to find the variation explained in the pixles explained by the first 25 PCs

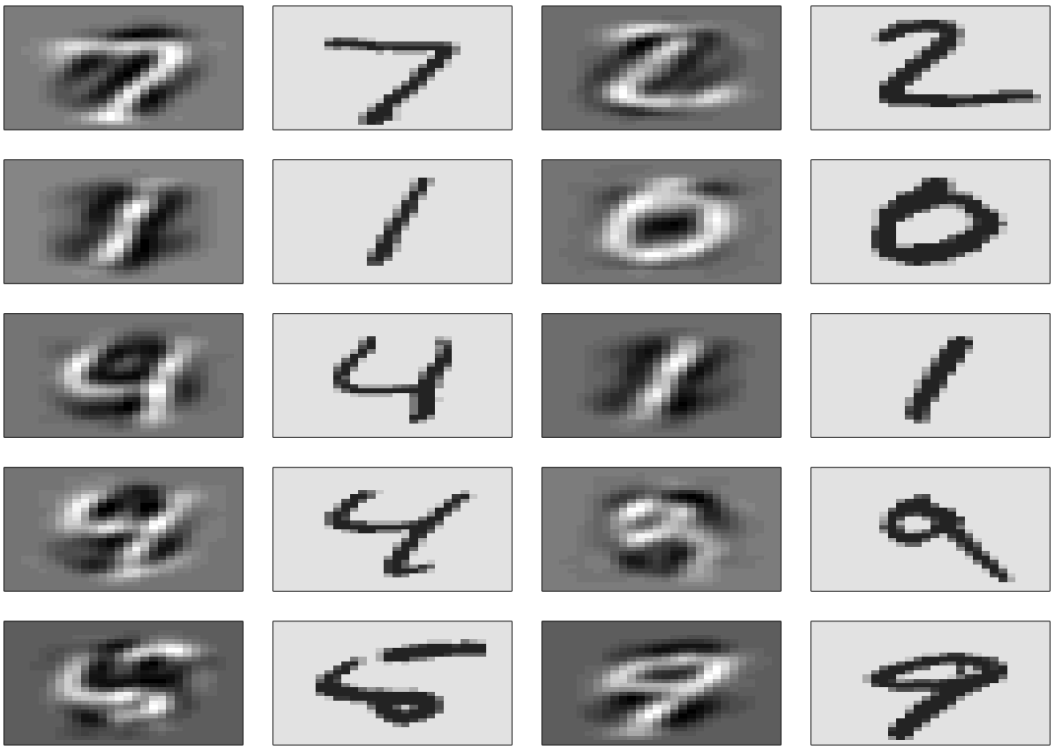
```
# source:https://stackoverflow.com/questions/23866765/getting-cumulative-proportion-in-pca
variance <- apply(prnum$x, 2, var)
props <- variance / sum(variance)
cumsum(props)[25]
```

```
##       PC25
## 0.7018987
```

PC25 tells us the that cummulative variance is 0.701898, which is 78% pretty good actually.

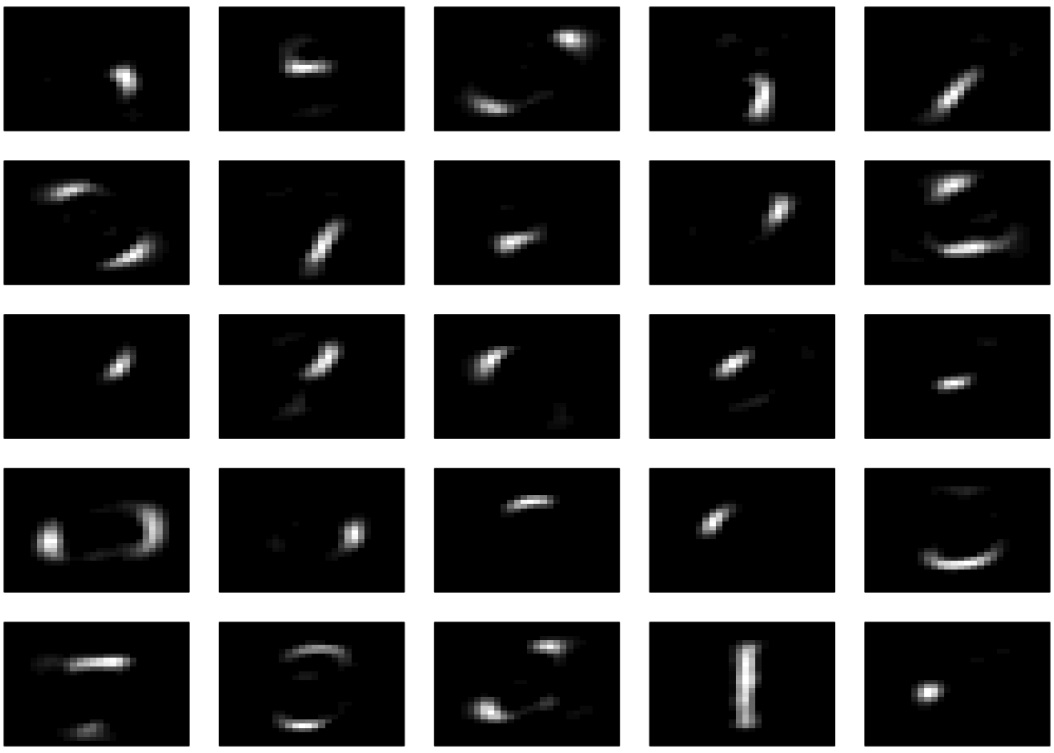
d) Reconstruct approximations of the original observations using 25 PCs. Plot side-bysides for the first 10 digits of the reconstructions and originals in a 5x4 matrix of images.

```
par(mfrow=c(5,4), omi=c(0,0,0,0), mai=c(0.1,0.1,0.1,0.1))

for (j in 1:10){
  reconst <- (t(prnum$rotation[,1:25] %*% t(prnum$x[,1:25])))
  reconstructed <- (matrix(reconst[j,], 28,28))
  image(reconstructed, col=gray((0:32)/32), xaxt="n", yaxt="n")
  image(imarr[,,j], col = gray(10:2/11),xaxt="n", yaxt="n")
}
```



e)

```
load("nmfres.Rdata")
par(mfrow=c(5,5), omi=c(0,0,0,0), mai=c(0.1,0.1,0.1,0.1))

for(i in 1:25){
    image((matrix(nmfres$h[i,], 28,28)),col = gray((0:32)/32), xaxt="n", yaxt="n")
}
```



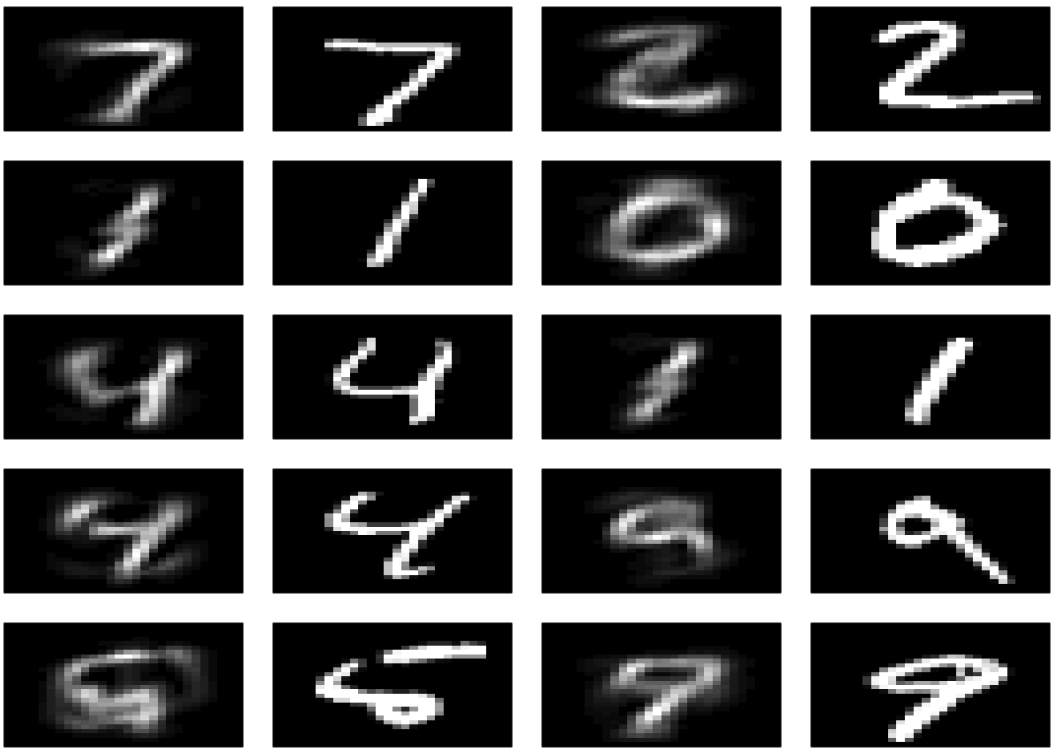
Comment on the differences between these and the eigenvectors from part (b).

We know that PCA has both negative and positive loadings (Eigenvectors), but NMF only has postive loadings (Eigenvectors). This is why the pictures from NMF is much darker than PCA. Note that PCA allows to add and subtract images from each other but not in NMF.

(From lecture we know that PCA can subtract the average face and diffrenent parts or what so ever and NMF restricts such a thing since in some particular data like spectral data this isn't allowed)

f) Reconstruct approximations of the original observations using 25 NMF bases. Plot side-by-sides for the first 10 digits of the reconstructions and originals in a 5x4 matrix of images

```
par(mfrow=c(5,4), omi=c(0,0,0,0), mai=c(0.1,0.1,0.1,0.1))
nmfrec <- nmfres$w %*% nmfres$h

for(i in 1:10){
  image((matrix(nmfrec[i,], 28, 28)),col = gray((0:32)/32), xaxt="n", yaxt="n")
  image(imarr[,,i],col = gray((0:32)/32),xaxt="n", yaxt="n")
}
```



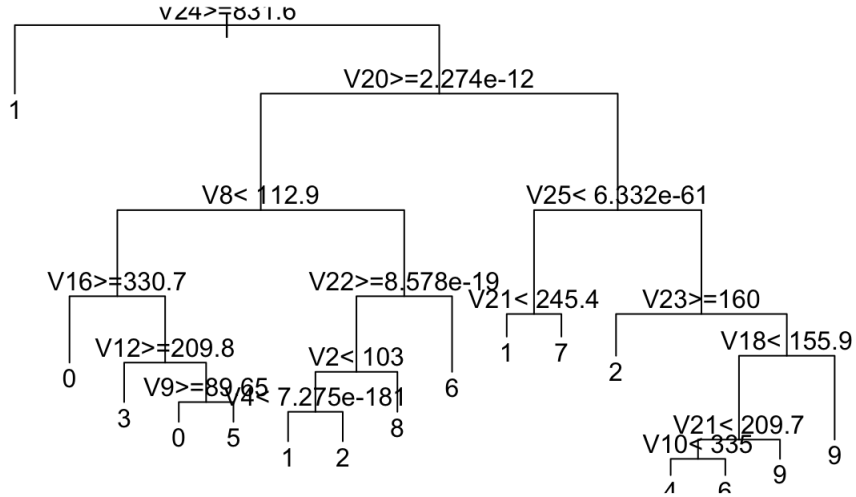
g) Fit a classification tree with labels as the response variable and the NMF ‘scores’ as the predictors. Plot the tree.

```
library(rpart)
```

```
##
## Attaching package: 'rpart'
```

```
## The following object is masked from 'package:survival':
##
##     solder
```

```
library(tree)

df <- as.data.frame(cbind(nmfres$w,labls))
df$labls<- as.factor(df$labls)

tree1<- rpart(labls~.,data=df, method="class",control = rpart.control(minsplit = 20, xval = 10, cp = 0.01))

par(mfrow=c(1,1))
plot(tree1)
text(tree1)
```



h) Use ‘cv.tree’ with ‘prune.misclass’ as the function, how many nodes are suggested to be removed? What is the cross-validated misclassification rate of the best tree?

cv.tree doesnt work for rpart and when Matthias and I did cv.tree for tree() it didn’t work either so we took on a different method

```
printcp(tree1)
```

```
##
## Classification tree:
## rpart(formula = labls ~ ., data = df, method = "class", control = rpart.control(minsplit = 20,
##     xval = 10, cp = 0.01))
##
## Variables actually used in tree construction:
##  [1] V10 V12 V16 V18 V2  V20 V21 V22 V23 V24 V25 V4  V8  V9
##
## Root node error: 8865/10000 = 0.8865
##
## n= 10000
##
##          CP nsplit rel error  xerror      xstd
## 1  0.068697      0   1.00000 1.00000 0.0035782
## 2  0.063959      3   0.78545 0.73232 0.0053832
## 3  0.056176      4   0.72149 0.67693 0.0055260
## 4  0.049295      5   0.66531 0.64005 0.0055887
## 5  0.046644      6   0.61602 0.60541 0.0056249
## 6  0.029667      8   0.52273 0.53119 0.0056306
## 7  0.029329      9   0.49306 0.51506 0.0056189
## 8  0.021884     10   0.46373 0.48584 0.0055857
## 9  0.016808     11   0.44185 0.46080 0.0055449
## 10 0.014664     12   0.42504 0.44219 0.0055070
## 11 0.014100     13   0.41038 0.42854 0.0054750
## 12 0.012070     14   0.39628 0.41309 0.0054345
## 13 0.010000     15   0.38421 0.40090 0.0053991
```

```
tree1$cptable
```

```
##            CP nsplit rel error    xerror        xstd
## 1  0.06869712      0 1.0000000 1.0000000 0.003578150
## 2  0.06395939      3 0.7854484 0.7323181 0.005383197
## 3  0.05617597      4 0.7214890 0.6769318 0.005525973
## 4  0.04929498      5 0.6653130 0.6400451 0.005588679
## 5  0.04664411      6 0.6160180 0.6054146 0.005624944
## 6  0.02966723      8 0.5227298 0.5311901 0.005630598
## 7  0.02932882      9 0.4930626 0.5150592 0.005618871
## 8  0.02188381     10 0.4637338 0.4858432 0.005585722
## 9  0.01680767     11 0.4418500 0.4608009 0.005544912
## 10 0.01466441     12 0.4250423 0.4421884 0.005507013
## 11 0.01410039     13 0.4103779 0.4285392 0.005475033
## 12 0.01206994     14 0.3962775 0.4130852 0.005434462
## 13 0.01000000     15 0.3842076 0.4009024 0.005399147
```

```
prunedtree <- prune(tree1, cp=tree1$cptable[which.min(tree1$cptable[,"xerror"]),"CP"])
prunedtree$cptable
```

```
##            CP nsplit rel error    xerror        xstd
## 1  0.06869712      0 1.0000000 1.0000000 0.003578150
## 2  0.06395939      3 0.7854484 0.7323181 0.005383197
## 3  0.05617597      4 0.7214890 0.6769318 0.005525973
## 4  0.04929498      5 0.6653130 0.6400451 0.005588679
## 5  0.04664411      6 0.6160180 0.6054146 0.005624944
## 6  0.02966723      8 0.5227298 0.5311901 0.005630598
## 7  0.02932882      9 0.4930626 0.5150592 0.005618871
## 8  0.02188381     10 0.4637338 0.4858432 0.005585722
## 9  0.01680767     11 0.4418500 0.4608009 0.005544912
## 10 0.01466441     12 0.4250423 0.4421884 0.005507013
## 11 0.01410039     13 0.4103779 0.4285392 0.005475033
## 12 0.01206994     14 0.3962775 0.4130852 0.005434462
## 13 0.01000000     15 0.3842076 0.4009024 0.005399147
```

```
printcp(prunedtree)
```

```
##
## Classification tree:
## rpart(formula = labls ~ ., data = df, method = "class", control = rpart.control(minsplit = 20,
##     xval = 10, cp = 0.01))
##
## Variables actually used in tree construction:
##  [1] V10 V12 V16 V18 V2  V20 V21 V22 V23 V24 V25 V4  V8  V9
##
## Root node error: 8865/10000 = 0.8865
##
## n= 10000
##
##          CP nsplit rel error  xerror      xstd
## 1  0.068697      0   1.00000 1.00000 0.0035782
## 2  0.063959      3   0.78545 0.73232 0.0053832
## 3  0.056176      4   0.72149 0.67693 0.0055260
## 4  0.049295      5   0.66531 0.64005 0.0055887
## 5  0.046644      6   0.61602 0.60541 0.0056249
## 6  0.029667      8   0.52273 0.53119 0.0056306
## 7  0.029329      9   0.49306 0.51506 0.0056189
## 8  0.021884     10   0.46373 0.48584 0.0055857
## 9  0.016808     11   0.44185 0.46080 0.0055449
## 10 0.014664     12   0.42504 0.44219 0.0055070
## 11 0.014100     13   0.41038 0.42854 0.0054750
## 12 0.012070     14   0.39628 0.41309 0.0054345
## 13 0.010000     15   0.38421 0.40090 0.0053991
```

```
0.8865*0.40226
```

```
## [1] 0.3566035
```

The cross validated misclassification rate 0.8865*0.40226 = 0.3566035.