Stat 240 - Take Home Final Exam - Spring 2019

Due by 6 pm, Wednesday, May 15

Michael Purvis

Instructions

Your fully reproducible Rmd AND compiled pdf solution is due via Moodle by Wednesday, May 15th by 6 pm. NO LATE SUBMISSIONS will be accepted. Submit both files early, and check that your solution compiles well in advance of the deadline!

The solution you provide for the exam must be your own work.

The take home exam may be completed in multiple sittings. I expect it will take you several hours to complete because you will be supplying your own code for the solution. Some problems are longer than others, so I encourage you to read through them all to get a sense of each.

The exam is open class textbooks (recall these are available via the e-reserves link on Moodle), our class materials, our Moodle site, but CLOSED to all external sources for statistical information.

If needed, you may use the R help menu, files, and official package documentation for code assistance. For example, you can do ?functionname, or google the official package documentation pdf (say, for igraph, for example). All other external code references are forbidden. It is inappropriate, for example, to google for the functionname, find a similar analysis done by someone else, and use that code. The help menu should be more than sufficient - you should have examples of all code needed from class. If you encounter an error and cannot resolve it from the help menu, contact me for assistance.

For your solution:

- Be sure your code is READABLE in the pdf (i.e. that it does not go over the ends of the lines)
- That all necessary packages are listed in the top chunk in the Rmd (the package list itself can be hidden in the pdf)
- Show all code necessary to reproduce your work in the pdf (i.e. ONLY the package list can be hidden in the pdf).
- Be sure to submit BOTH the pdf and Rmd files in Moodle

You may NOT consult anyone except the instructor (Amy Wagaman). If a question is asked where the entire class should be notified of the answer, the instructor will handle that notification.

General guidance: If you have a chunk of code that you are using to do something, it is helpful to state what you are using it for. It is a good idea to explain the steps you are taking for your analysis. If I cannot follow your work, it will not earn you partial credit. You won't need this for every part below, but the idea is that I need to be able to follow what you are doing. Basically, shorter code chunks are recommended, with interspersed comments, rather than ALL code and output (several pages), followed by your response.

There are 5 problems, for a total of 125 points. Point breakdown is shown by problem and by part. Partial credit may be given for problems based on your submitted work.

If desired, you may delete this line and the instructions above in your submission.

Honor Code Statement

Please add your full name as your electronic signature for agreeing to abide by the honor code and honor statement provided below.

I have abided by the Amherst College Honor Code in regards to this exam. In particular, I have neither given nor received any unauthorized assistance with this take home exam.

Michael Purvis

Question 1 - 29 points

Letter recognition is a challenging problem with wide-ranging applications. For example, think of all the mail that goes through the post office - numerical recognition of zip code digits is very helpful, and so is letter recognition for addresses. For this problem, you have a filtered (for size) version of a data set from UCI's machine learning repository to work with. A separate pdf of a data dictionary is provided for you. The data set and a few other variants of it are provided for you to use. We are interested in correctly predicting which capital letter we have based on attributes of it when viewed as an image with pixel characteristics.

```r
#setup chunk, do not alter
letters <- read_csv("https://awagaman.people.amherst.edu/stat240/letters2.csv")
letters <- mutate(letters, letter = factor(letter))

#create train/test set
letters <- letters %>% mutate(id = row_number())
set.seed(240)
lettertrain <- letters %>% sample_frac(0.75)
lettertest  <- anti_join(letters, lettertrain, by = 'id')

letters <- select(letters, -id) #drop id variable
lettertrain <- select(lettertrain, -id) #drop id variable
lettertest <- select(lettertest, -id) #drop id variable
```

a. (2 points) What are the classes in the filtered data set? How many are there? Is any class rare? Provide supporting output.

SOLUTION:

```r
summary(letters)
```

```
##  letter       xbox              ybox              width
##  A:789   Min.   : 0.0000   Min.   : 0.0000   Min.   : 1.0000
##  E:768   1st Qu.: 3.0000   1st Qu.: 5.0000   1st Qu.: 4.0000
##  H:734   Median : 4.0000   Median : 7.0000   Median : 5.0000
##  M:792   Mean   : 4.0643   Mean   : 7.0464   Mean   : 5.3434
##  R:758   3rd Qu.: 5.0000   3rd Qu.: 9.0000   3rd Qu.: 6.0000
##  S:748   Max.   :15.0000   Max.   :15.0000   Max.   :15.0000
##  T:796
##      height          onpix             xbar              ybar
##  Min.   : 0.000   Min.   : 0.0000   Min.   : 2.0000   Min.   : 0.0000
##  1st Qu.: 4.000   1st Qu.: 2.0000   1st Qu.: 6.0000   1st Qu.: 6.0000
##  Median : 6.000   Median : 3.0000   Median : 7.0000   Median : 7.0000
##  Mean   : 5.244   Mean   : 3.8137   Mean   : 7.3142   Mean   : 7.3096
##  3rd Qu.: 7.000   3rd Qu.: 5.0000   3rd Qu.: 8.0000   3rd Qu.: 8.0000
##  Max.   :10.000   Max.   :15.0000   Max.   :14.0000   Max.   :15.0000
##
##      x2bar             y2bar             xybar             x2ybar
##  Min.   : 0.0000   Min.   : 0.0000   Min.   : 3.0000   Min.   : 0.0000
##  1st Qu.: 3.0000   1st Qu.: 3.0000   1st Qu.: 7.0000   1st Qu.: 4.0000
##  Median : 4.0000   Median : 5.0000   Median : 7.0000   Median : 6.0000
##  Mean   : 4.5811   Mean   : 4.9339   Mean   : 8.0739   Mean   : 5.8032
##  3rd Qu.: 6.0000   3rd Qu.: 7.0000   3rd Qu.:10.0000   3rd Qu.: 7.0000
##  Max.   :15.0000   Max.   :12.0000   Max.   :14.0000   Max.   :12.0000
```

```
##
##      xy2bar            xege             xegvy              yege
##  Min.   : 2.0000   Min.   : 0.0000   Min.   : 0.0000   Min.   : 0.0000
##  1st Qu.: 7.0000   1st Qu.: 2.0000   1st Qu.: 6.0000   1st Qu.: 2.0000
##  Median : 8.0000   Median : 3.0000   Median : 8.0000   Median : 4.0000
##  Mean   : 8.0949   Mean   : 3.4427   Mean   : 7.6589   Mean   : 4.0479
##  3rd Qu.: 9.0000   3rd Qu.: 5.0000   3rd Qu.: 8.0000   3rd Qu.: 6.0000
##  Max.   :15.0000   Max.   :15.0000   Max.   :14.0000   Max.   :15.0000
##
##      yegvx
##  Min.   : 0.0000
##  1st Qu.: 7.0000
##  Median : 8.0000
##  Mean   : 7.8423
##  3rd Qu.: 9.0000
##  Max.   :13.0000
##
```

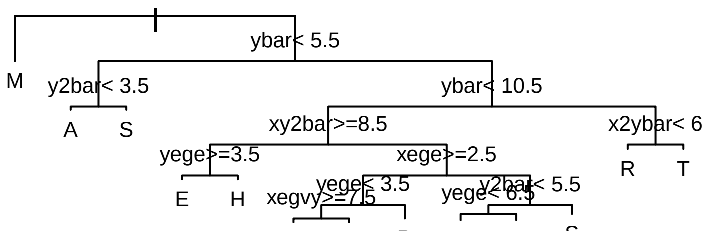
There appear to be 7 classes: the letters A, E, H, M, R, S, and T. There does not seem to be a very large variance in each class's frequency. The most common class, T has 796 observations. The least frequent class is H with 734 observations.

b. (6 points) Use a classification tree with parameter values chosen by you to optimize the resulting solution to obtain an AER and estimated TER. Report your findings, including your chosen parameters, AER, estimated TER, and what method you used to obtain the estimated TER. Using all default parameter values does not demonstrate that you optimized the solution.

SOLUTION:

The optimal solution that I found had an AER of .235 and an Estimated TER of .239. To find the crossvalidated error, which is then multiplied by the root null error to obtain an estimated TER, I set the number of cross-validation folds to 10. In addition, the minsplit was set to 10 and the minbucket was set to 3. I also tried other combinations for the parameters, but this provided the best AER and estimated TER.

```r
g.control <- rpart.control(minsplit = 10, minbucket = 3, xval = 10)
g.treeorig <- rpart(letter ~ ., data = letters, method = "class", control = g.control)
plot(g.treeorig)
text(g.treeorig, cex = 0.7)
```

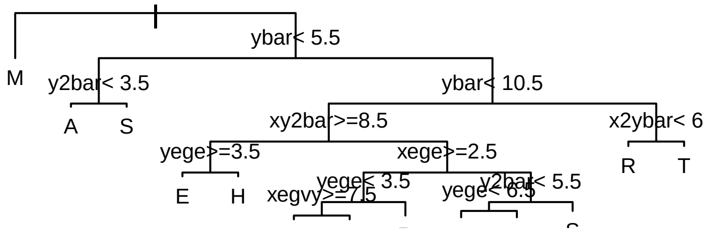


```r
printcp(g.treeorig)
```

```
##
## Classification tree:
## rpart(formula = letter ~ ., data = letters, method = "class",
##     control = g.control)
##
## Variables actually used in tree construction:
## [1] x2ybar xege   xegvy  xy2bar y2bar  ybar   yege
##
```

```
## Root node error: 4589/5385 = 0.852182
##
## n= 5385
##
##            CP nsplit rel error   xerror       xstd
## 1  0.1434953      0  1.000000 1.013728 0.00548354
## 2  0.1205056      2  0.713009 0.713009 0.00780810
## 3  0.0982785      3  0.592504 0.573763 0.00799353
## 4  0.0937023      4  0.494225 0.483548 0.00787088
## 5  0.0427108      5  0.400523 0.389192 0.00752871
## 6  0.0165613      6  0.357812 0.353454 0.00733638
## 7  0.0141643      8  0.324689 0.337982 0.00724138
## 8  0.0122031      9  0.310525 0.319895 0.00712080
## 9  0.0119852     10  0.298322 0.304859 0.00701240
## 10 0.0102419     11  0.286337 0.295707 0.00694263
## 11 0.0100000     12  0.276095 0.288952 0.00688923
```

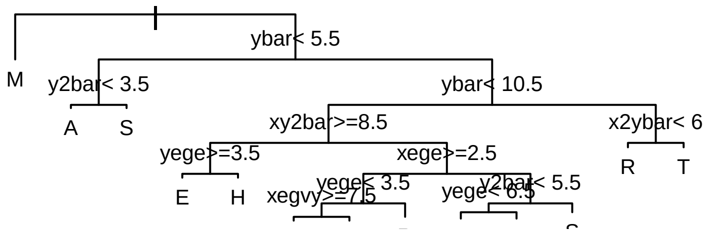
```r
g.control2 <- rpart.control(minsplit = 50, minbucket = 10, xval = 10)
g.treeorig2 <- rpart(letter ~ ., data = letters, method = "class", control = g.control2)
plot(g.treeorig2)
text(g.treeorig2, cex = 0.7)
```



```r
printcp(g.treeorig2)
```

```
##
## Classification tree:
## rpart(formula = letter ~ ., data = letters, method = "class",
##     control = g.control2)
##
## Variables actually used in tree construction:
## [1] x2ybar xege   xegvy  xy2bar y2bar  ybar   yege
##
## Root node error: 4589/5385 = 0.852182
##
## n= 5385
##
##            CP nsplit rel error   xerror       xstd
## 1  0.1434953      0  1.000000 1.016561 0.00544228
## 2  0.1205056      2  0.713009 0.713009 0.00780810
## 3  0.0982785      3  0.592504 0.592504 0.00799510
## 4  0.0937023      4  0.494225 0.493354 0.00789358
## 5  0.0427108      5  0.400523 0.384397 0.00750503
## 6  0.0165613      6  0.357812 0.355197 0.00734662
## 7  0.0141643      8  0.324689 0.336021 0.00722880
## 8  0.0122031      9  0.310525 0.323382 0.00714486
## 9  0.0119852     10  0.298322 0.304642 0.00701077
## 10 0.0102419     11  0.286337 0.295489 0.00694093
## 11 0.0100000     12  0.276095 0.276531 0.00678668
```

```
g.control3 <- rpart.control(minsplit = 100, minbucket = 30, xval = 10)
g.treeorig3 <- rpart(letter ~ ., data = letters, method = "class", control = g.control3)
plot(g.treeorig3)
text(g.treeorig3, cex = 0.7)
```

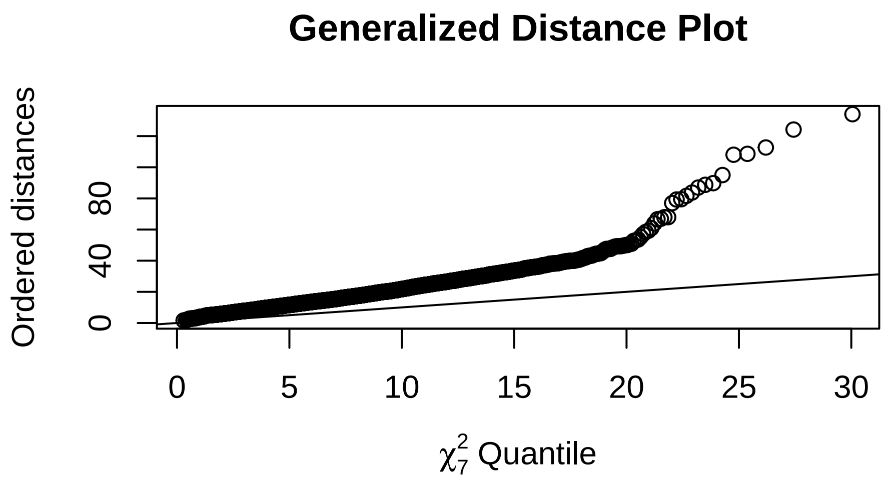


```
printcp(g.treeorig3)
```

```
##
## Classification tree:
## rpart(formula = letter ~ ., data = letters, method = "class",
##     control = g.control3)
##
## Variables actually used in tree construction:
## [1] x2ybar xege   xegvy  xy2bar y2bar  ybar   yege
##
## Root node error: 4589/5385 = 0.852182
##
## n= 5385
##
##           CP nsplit rel error   xerror       xstd
## 1  0.1434953      0  1.000000 1.010678 0.00552732
## 2  0.1205056      2  0.713009 0.713009 0.00780810
## 3  0.0982785      3  0.592504 0.592504 0.00799510
## 4  0.0937023      4  0.494225 0.520811 0.00794486
## 5  0.0427108      5  0.400523 0.388320 0.00752446
## 6  0.0165613      6  0.357812 0.355633 0.00734917
## 7  0.0141643      8  0.324689 0.341251 0.00726206
## 8  0.0122031      9  0.310525 0.321421 0.00713138
## 9  0.0119852     10  0.298322 0.312268 0.00706677
## 10 0.0102419     11  0.286337 0.294400 0.00693242
## 11 0.0100000     12  0.276095 0.285247 0.00685924
```

c. (6 points) Investigate the applicability of linear discriminant analysis (LDA) to this classification problem. Does an LDA solution seem viable using all possible predictors? Any subset of predictors? Report on your findings, including how you assessed LDA's viability with appropriate support. If LDA is viable, perform an LDA and obtain an AER.

```
x <- letters[, -1]
cm <- colMeans(x)
S <- cov(x)
d <- apply(x, MARGIN = 1, function(x) t(x - cm) %*% solve(S) %*% (x - cm))
plot(qchisq((1:nrow(x) - 1/2) / nrow(x), df = 7), sort(d),
     xlab = expression(paste(chi[7]^2, " Quantile")),  ylab = "Ordered distances", main = "Generalized 
abline(a = 0, b = 1)
```

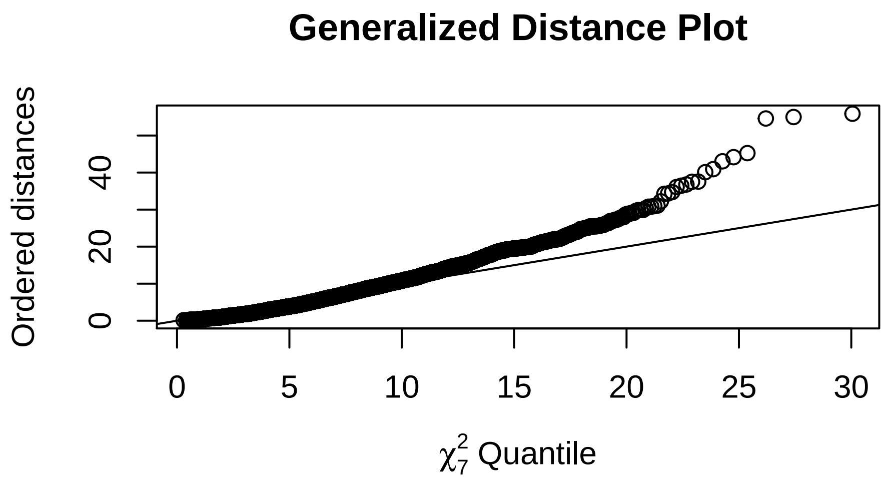


```
#this creates a generalized distance plot so that we can check multivariate normality.
```

This does not appear to be multivariate normal, so we cannot run LDA on the full set of variables. In order to find out what variables to remove. I will look to see which variables do not seem to be normally distributed and remove them.

```
#ggpairs(letters)
```

```
x <- letters[, -c(1, 3, 5, 6, 9, 10, 11, 12, 14, 16)]
cm <- colMeans(x)
S <- cov(x)
d <- apply(x, MARGIN = 1, function(x) t(x - cm) %*% solve(S) %*% (x - cm))
plot(qchisq((1:nrow(x) - 1/2) / nrow(x), df = 7), sort(d),
     xlab = expression(paste(chi[7]^2, " Quantile")),  ylab = "Ordered distances", main = "Generalized D
abline(a = 0, b = 1)
```



After taking out the non-normal variables, the generalized distance plot is still not showing multivariater normality. I believe that LDA should not be run on the subset of variables because the multivariate normality condition is not met.

d. (6 points) Apply nearest neighbor methods with parameter values chosen by you to optimize the solution to find an AER and estimated TER. Report your findings, including your chosen parameters, AER, estimated TER, and what method you used to obtain the estimated TER. Using k=1 is not recommended.

SOLUTION:

I will run 3 knn with k = 2, 3, and 4. I will then choose the model that has the lowest estimated TER.

```
set.seed(240)
g.knn <- class::knn(lettertrain[, 2:17], lettertrain[, 2:17], lettertrain$letter, k = 2, prob = T)
temptable<- table(lettertrain$letter, g.knn)
tempsum <- as.numeric(sum(as.matrix(temptable)) - sum(diag(as.matrix(temptable))))
AER <- tempsum/(sum(as.matrix(temptable)))
AER
```

```
## [1] 0.0054468928
```

```
g.knn <- class::knn(lettertrain[, 2:17], lettertest[, 2:17], lettertrain$letter, k = 2, prob = T)
temptable<- table(lettertest$letter, g.knn)
tempsum <- as.numeric(sum(as.matrix(temptable)) - sum(diag(as.matrix(temptable))))
eTER <- tempsum/(sum(as.matrix(temptable)))
eTER
```

```
## [1] 0.018573551
```

```
#if you get an error about unused arguments, try class:: before the function call
```

```
set.seed(240)
g.knn <- class::knn(lettertrain[, 2:17], lettertrain[, 2:17], lettertrain$letter, k = 3, prob = T)
temptable<- table(lettertrain$letter, g.knn)
tempsum <- as.numeric(sum(as.matrix(temptable)) - sum(diag(as.matrix(temptable))))
AER <- tempsum/(sum(as.matrix(temptable)))
AER
```

```
## [1] 0.0076751671
```

```
g.knn <- class::knn(lettertrain[, 2:17], lettertest[, 2:17], lettertrain$letter, k = 3, prob = T)
temptable<- table(lettertest$letter, g.knn)
tempsum <- as.numeric(sum(as.matrix(temptable)) - sum(diag(as.matrix(temptable))))
eTER <- tempsum/(sum(as.matrix(temptable)))
eTER
```

```
## [1] 0.014858841
```

```
set.seed(240)
g.knn <- class::knn(lettertrain[, 2:17], lettertrain[, 2:17], lettertrain$letter, k = 4, prob = T)
temptable<- table(lettertrain$letter, g.knn)
tempsum <- as.numeric(sum(as.matrix(temptable)) - sum(diag(as.matrix(temptable))))
AER <- tempsum/(sum(as.matrix(temptable)))
AER
```

```
## [1] 0.0084179252
```

```
g.knn <- class::knn(lettertrain[, 2:17], lettertest[, 2:17], lettertrain$letter, k = 4, prob = T)
temptable<- table(lettertest$letter, g.knn)
tempsum <- as.numeric(sum(as.matrix(temptable)) - sum(diag(as.matrix(temptable))))
eTER <- tempsum/(sum(as.matrix(temptable)))
eTER
```

```
## [1] 0.017087667
```

Apparently running knn with k = 3 is the best model of knn because it gives us the lowest estimated TER of .014.

e. (6 points) Use boosting with parameter values chosen by you to optimize the resulting solution to obtain an AER and estimated TER. Report your findings, including your chosen parameters, AER, estimated TER, and what method you used to obtain the estimated TER. Using all default parameter values does not demonstrate that you optimized the solution.

I will runt diiferent boosting models using different parameters and choose the one that has the lowest estimated TER.

SOLUTION:

```r
set.seed(240)
letter.boost <- gbm(letter ~ ., data = lettertrain, distribution = "multinomial",
                    n.trees = 2500, interaction.depth = 2, shrinkage = 0.1)
boost_estimate <- predict(letter.boost, newdata = lettertrain,
                          n.trees = 2500, type = "response")
pred_letter <- apply(boost_estimate, 1, which.max)
temptable<- table(lettertrain$letter, pred_letter)
tempsum <- as.numeric(sum(as.matrix(temptable)) - sum(diag(as.matrix(temptable))))
AER <- tempsum/(sum(as.matrix(temptable)))
AER
```

```
## [1] 0
```

```r
boost_estimate <- predict(letter.boost, newdata = lettertest,
                          n.trees = 2500, type = "response")
pred_letter <- apply(boost_estimate, 1, which.max)
temptable<- table(lettertest$letter, g.knn)
tempsum <- as.numeric(sum(as.matrix(temptable)) - sum(diag(as.matrix(temptable))))
eTER <- tempsum/(sum(as.matrix(temptable)))
eTER
```

```
## [1] 0.017087667
```

```r
set.seed(240)
letter.boost <- gbm(letter ~ ., data = lettertrain, distribution = "multinomial",
                    n.trees = 1000, interaction.depth = 4, shrinkage = 0.05)
boost_estimate <- predict(letter.boost, newdata = lettertrain,
                          n.trees = 1000, type = "response")
pred_letter <- apply(boost_estimate, 1, which.max)
temptable<- table(lettertrain$letter, pred_letter)
tempsum <- as.numeric(sum(as.matrix(temptable)) - sum(diag(as.matrix(temptable))))
AER <- tempsum/(sum(as.matrix(temptable)))
AER
```

```
## [1] 0
```

```r
boost_estimate <- predict(letter.boost, newdata = lettertest,
                          n.trees = 1000, type = "response")
pred_letter <- apply(boost_estimate, 1, which.max)
temptable<- table(lettertest$letter, g.knn)
tempsum <- as.numeric(sum(as.matrix(temptable)) - sum(diag(as.matrix(temptable))))
eTER <- tempsum/(sum(as.matrix(temptable)))
eTER
```

```
## [1] 0.017087667
```

Apparently having a boosting model with n.trees= 2500, depth= 2 and shrinkage = .1 gives the same AER and Estimated TER as a model with n.trees= 1000, interactiondepth = 4, and shrinkage = .05. Therefore I believe it is best to choose the model that is less complex, so I believe the ideal model would be the the boosting model with 1000 tress.

f. (3 points) Consider the models you fit above. Choose one as your optimal model for predicting the capital letters, and state your rationale/supporting evidence for your choice.

SOLUTION: The optimal model would be my knn model with k =3. Because we are trying to predict the letter, we want the most accurate model, and this knn model has the lowest estimated TER, of .014, than all of the other models that were created.

Question 2 - 20 points

You previously saw part of the avocado data set used on midterm 2. In this problem, you will be working with a smaller subset (only 500 of the roughly 18000 observations) in order to help avocado growers with some visualizations. The original data set is from Kaggle, and a data dictionary has been provided for you in a separate pdf document.

```
#load the subset of the data available
avocado <- read_csv("https://awagaman.people.amherst.edu/stat240/avocadosubset.csv")
```

Your task for this problem is to create two multivariate (not just bivariate) visualizations to share with avocado growers that point out interesting items of note in the data set. Present your well-constructed visuals as well as your “story” for what you found that is interesting in a few paragraphs. You may use univariate and bivariate visuals as additional support as needed for your story.

SOLUTION:

```
cor(avocado[, -c(1, 11, 13)])
```

```
##              AveragePrice  TotalVolume        V4046       V4225
## AveragePrice  1.000000000 -0.217791918 -0.219757283 -0.20277125
## TotalVolume  -0.217791918  1.000000000  0.981685249  0.98266407
## V4046        -0.219757283  0.981685249  1.000000000  0.94404185
## V4225        -0.202771251  0.982664070  0.944041848  1.00000000
## V4770        -0.232433541  0.873716078  0.861379284  0.88368879
## TotalBags    -0.212115515  0.963497681  0.918848976  0.92859674
## SmallBags    -0.211748581  0.971812937  0.933905985  0.93880833
## LargeBags    -0.198084702  0.856890304  0.796806390  0.81818222
## XLargeBags   -0.099029154  0.622531895  0.559601763  0.61078840
## year          0.097637430 -0.035344495 -0.057359145 -0.05382384
##                    V4770    TotalBags    SmallBags   LargeBags
## AveragePrice -0.23243354 -0.212115515 -0.211748581 -0.19808470
## TotalVolume   0.87371608  0.963497681  0.971812937  0.85689030
## V4046         0.86137928  0.918848976  0.933905985  0.79680639
## V4225         0.88368879  0.928596744  0.938808328  0.81818222
## V4770         1.00000000  0.779092238  0.800426837  0.64538701
## TotalBags     0.77909224  1.000000000  0.993457008  0.93517913
## SmallBags     0.80042684  0.993457008  1.000000000  0.88895912
## LargeBags     0.64538701  0.935179127  0.888959122  1.00000000
## XLargeBags    0.52163273  0.681054176  0.657465883  0.66268238
## year         -0.10447032  0.033336858  0.028122925  0.04376332
##                XLargeBags         year
## AveragePrice -0.099029154  0.097637430
## TotalVolume   0.622531895 -0.035344495
## V4046         0.559601763 -0.057359145
## V4225         0.610788401 -0.053823840
## V4770         0.521632726 -0.104470320
## TotalBags     0.681054176  0.033336858
## SmallBags     0.657465883  0.028122925
## LargeBags     0.662682382  0.043763320
## XLargeBags    1.000000000  0.072654984
## year          0.072654984  1.000000000
```

```
#add chunks and reorganize as needed
```

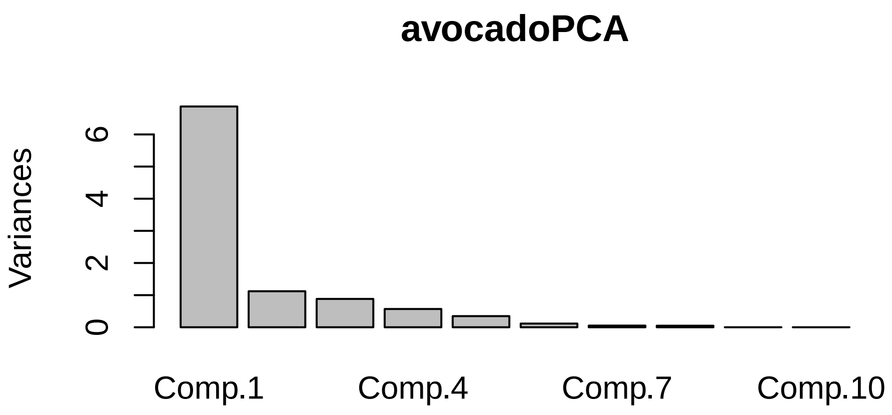
```
eigen(cor(avocado[, -c(1, 11, 13)]))
```

```
## eigen() decomposition
## $values
##  [1] 6.8695154e+00 1.1207110e+00 8.8172317e-01 5.6790082e-01 3.4614739e-01
##  [6] 1.1469596e-01 5.1140481e-02 4.8165767e-02 3.4830091e-12 1.2116924e-15
##
## $vectors
##               [,1]         [,2]          [,3]         [,4]          [,5]
##  [1,]  0.0967454632  0.525908426  0.8393536965  0.087168422 -0.0147460548
##  [2,] -0.3775247776 -0.018644622  0.0507755859  0.149012055 -0.0059993443
##  [3,] -0.3649257754 -0.055763020  0.0612932253  0.236456462  0.0327957508
##  [4,] -0.3697123903 -0.034179156  0.0769196671  0.162307513  0.0996484511
##  [5,] -0.3314026456 -0.139302940  0.0762520319  0.283987396  0.6136376009
##  [6,] -0.3739053926  0.072140884 -0.0083243925 -0.029122013 -0.2681510794
##  [7,] -0.3735171014  0.055681190  0.0017029063  0.042049598 -0.1786730830
##  [8,] -0.3422649219  0.107242957 -0.0393418001 -0.199469822 -0.5675943650
##  [9,] -0.2681709960  0.216735428 -0.0065276328 -0.831133956  0.4131741359
## [10,]  0.0069529555  0.795367631 -0.5251254730  0.272221826  0.1244378550
##              [,6]          [,7]         [,8]          [,9]
##  [1,] -0.033977736 -0.0089446703  0.022265259  1.6262381e-07
##  [2,]  0.193630225 -0.0664352113 -0.151900182 -8.7597301e-01
##  [3,]  0.419185742 -0.7133395525  0.041269268  3.4161737e-01
##  [4,]  0.100469922  0.4213181350 -0.732806698  3.0878070e-01
##  [5,] -0.586725328 -0.0104466902  0.250831332  2.1921646e-02
##  [6,]  0.036468769  0.2458238843  0.325329042  8.9064032e-02
##  [7,]  0.234443041  0.4087640199  0.482613003  1.0536100e-01
##  [8,] -0.600386742 -0.2764792103 -0.185078537  3.3238058e-02
##  [9,]  0.119817604 -0.0667914079 -0.024461462  2.1617670e-03
## [10,] -0.013466239 -0.0248129450 -0.034622276  1.2020733e-07
##               [,10]
##  [1,]  8.0716924e-10
##  [2,]  5.7904320e-05
##  [3,] -2.2582358e-05
##  [4,] -2.0413208e-05
##  [5,] -1.4486279e-06
##  [6,]  7.7886926e-01
##  [7,] -5.9801474e-01
##  [8,] -1.8865443e-01
##  [9,] -1.2270093e-02
## [10,] -1.6662802e-09
```

```
avocadoPCA<- princomp((avocado[, -c(1, 11, 13)]), cor = TRUE, scores = TRUE)
summary(avocadoPCA)
```

```
## Importance of components:
##                           Comp.1     Comp.2      Comp.3      Comp.4
## Standard deviation     2.62097604 1.05863638 0.939001156 0.753591944
## Proportion of Variance 0.68695154 0.11207110 0.088172317 0.056790082
## Cumulative Proportion  0.68695154 0.79902264 0.887194958 0.943985040
##                             Comp.5      Comp.6       Comp.7       Comp.8
## Standard deviation     0.588342922 0.338667925 0.2261426113 0.2194670077
## Proportion of Variance 0.034614739 0.011469596 0.0051140481 0.0048165767
## Cumulative Proportion  0.978599779 0.990069375 0.9951834233 1.0000000000
##                               Comp.9       Comp.10
## Standard deviation     1.8660097e-06 4.0056422e-08
## Proportion of Variance 3.4819922e-13 1.6045170e-16
## Cumulative Proportion  1.0000000e+00 1.0000000e+00
```

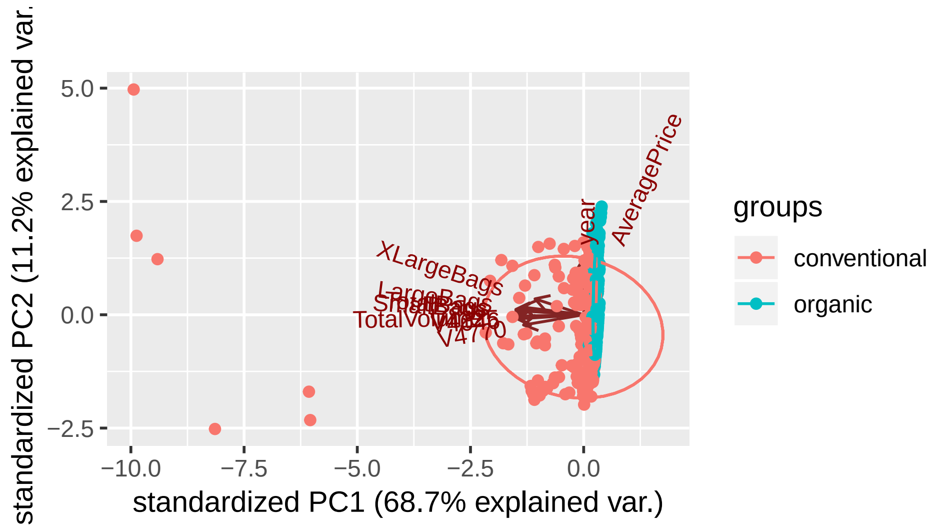
```
plot(avocadoPCA)
```



```
library(ggbiplot)
```

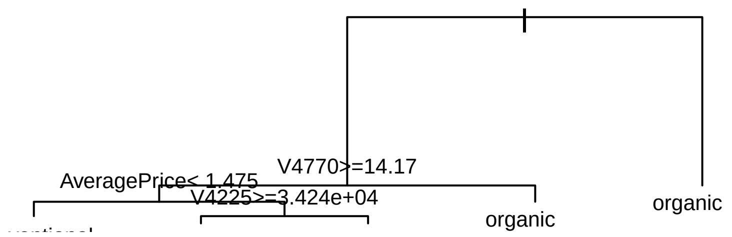
```
## Loading required package: plyr

## ------------------------------------------------------------------------------

## You have loaded plyr after dplyr - this is likely to cause problems.
## If you need functions from both plyr and dplyr, please load plyr first, then dplyr:
## library(plyr); library(dplyr)

## ------------------------------------------------------------------------------

## 
## Attaching package: 'plyr'

## The following object is masked from 'package:mosaic':
## 
##     count

## The following objects are masked from 'package:dplyr':
## 
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize

## Loading required package: scales

## 
## Attaching package: 'scales'

## The following object is masked from 'package:readr':
## 
##     col_factor

## The following object is masked from 'package:mosaic':
## 
##     rescale
## Loading required package: grid
```

```
ggbiplot(avocadoPCA, ellipse = TRUE, groups = factor(avocado$type), var.axes = TRUE, choices = c(1,2), 
```



According to this PCA analysis of the avocados it appears that the firt PCA describes the "smallness" of the avocado's sold, while the second PC focuses more on the overall price. In this visual, we can see that organic avocados tend to be smaller and more expensive than conventional avocados.

```
g.controlavo <- rpart.control(minsplit = 10, minbucket = 3, xval = 500)
g.treeorigavo <- rpart(type ~ ., data = avocado[, -c(1, 13)], method = "class", control = g.controlavo)
plot(g.treeorigavo)
text(g.treeorigavo, cex = 0.7)
```



```
printcp(g.treeorigavo)
```

```
##
## Classification tree:
## rpart(formula = type ~ ., data = avocado[, -c(1, 13)], method = "class",
##     control = g.controlavo)
##
## Variables actually used in tree construction:
## [1] AveragePrice TotalVolume  V4225        V4770
##
## Root node error: 230/500 = 0.46
##
## n= 500
##
##          CP nsplit rel error   xerror      xstd
## 1 0.8086957      0 1.0000000 1.000000 0.0484544
## 2 0.0782609      1 0.1913043 0.217391 0.0291661
## 3 0.0173913      2 0.1130435 0.143478 0.0241381
## 4 0.0100000      4 0.0782609 0.100000 0.0203662
```

According to this tree, we can predict with aproximately 4.6% error rate whether or not an avocado is conventional or organic. The variables taken into account here are Average Price, Total Volume, V4770, and V4225. This will help both consumers and supermarkets determine what kind of avocado they want to buy and sell.

Question 3 - 28 points

For this problem, we consider data on dolphins. The data is presented as an undirected social network of frequent associations between 62 dolphins in a community living off Doubtful Sound, New Zealand. The data set is from D. Lusseau, K. Schneider, O. J. Boisseau, P. Haase, E. Slooten, and S. M. Dawson, Behavioral Ecology and Sociobiology 54, 396-405 (2003).

```
dolphins <- read_graph("https://awagaman.people.amherst.edu/stat240/dolphins.gml", format = "gml")
```

a. (6 points) Obtain some appropriate descriptive statistics to describe basic properties of the network. Use your statistics to write a paragraph description of the network that could be used as an introduction to it for the data section of an article.

SOLUTION:

```
plot(dolphins)
```

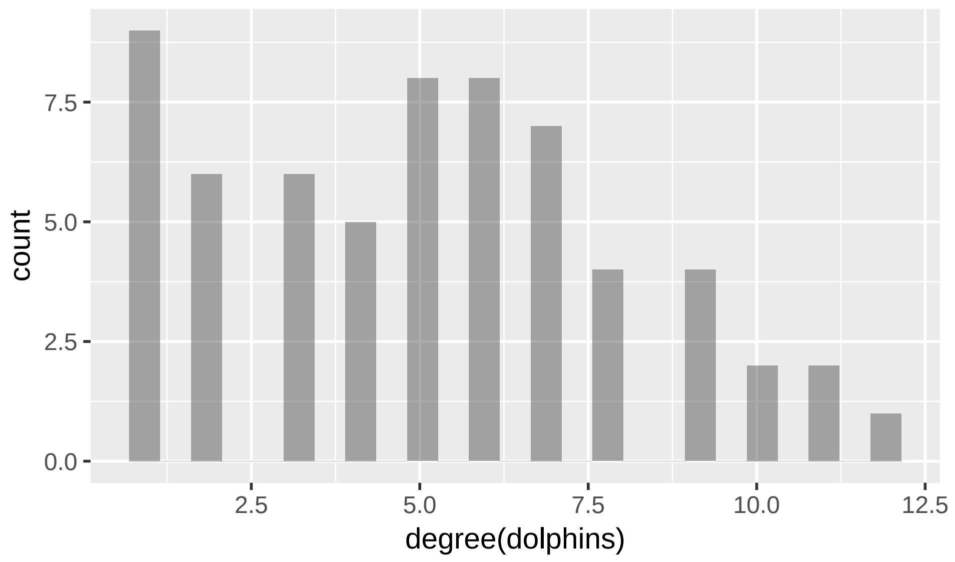
```
vcount(dolphins)
```

```
## [1] 62
```

```
ecount(dolphins)
```

```
## [1] 159
```

```
gf_histogram( ~ degree(dolphins))
```



```
tally(~ degree(dolphins))
```

```
## degree(dolphins)
##  1  2  3  4  5  6  7  8  9 10 11 12
##  9  6  6  5  8  8  7  4  4  2  2  1
```

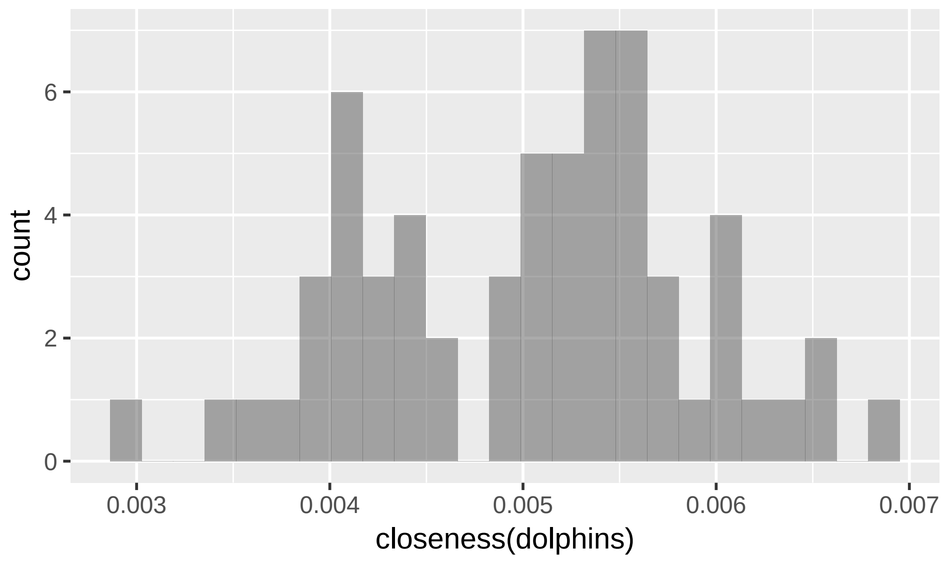
For this network, we consider data the frequent assossiation of members of the dolphin community in Doubtful Sound, New Zealand. In this network we have 62 dolphins with 159 connections. The distribution seems to be a bit skewed right and the average degree of the connections in this network is 5 connections. The most connections that a dolphin has is twevlve while the least, and most common number of connections is 1.

b. (6 points) Which dolphins are most central to the network? Use at least two centrality measures to determine which dolphins are most central. Are there clear "central" dolphins here? Report on your findings, including whether the centrality measures agree.

SOLUTION:

In order to find which dolphins are the most important, we will look at the dolphins with the highest closeness and betweenness measures. Looking at the scores for closeness, the top four dolphins are dolphins number 37, 41, 38, and 21. The top four dolphins for betweenness were 37, 2, 41, and 38. Seeing as to how 37, 41, and 38 appear in both top four, I would say that they are the most important dolphins.

```
gf_histogram(~ closeness(dolphins))
```



```
favstats(~ closeness(dolphins))
```

```
##          min           Q1       median           Q3          max
##  0.0029239766 0.0042879806 0.0051813472 0.0055555556 0.0068493151
##         mean            sd  n missing
##  0.0050367209 0.00085289563 62       0
```

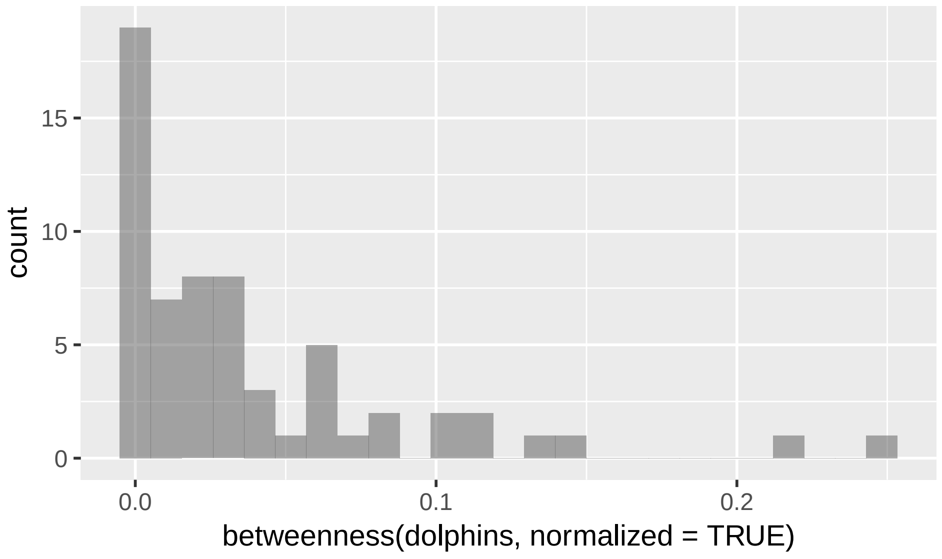
```
length(unique(closeness(dolphins)))
```

```
## [1] 43
```

```
sort(closeness(dolphins), method = "shell", index.return = TRUE, decreasing = TRUE)
```

```
## $x
##  [1] 0.0068493151 0.0066225166 0.0065359477 0.0064102564 0.0061728395
##  [6] 0.0060975610 0.0059880240 0.0059880240 0.0059880240 0.0059523810
## [11] 0.0057471264 0.0056818182 0.0056818182 0.0056179775 0.0056179775
## [16] 0.0055555556 0.0055555556 0.0055248619 0.0055248619 0.0054945055
## [21] 0.0054644809 0.0054644809 0.0054054054 0.0054054054 0.0054054054
## [26] 0.0054054054 0.0053191489 0.0052910053 0.0052910053 0.0051813472
## [31] 0.0051813472 0.0051813472 0.0051282051 0.0051282051 0.0051020408
## [36] 0.0050761421 0.0050505051 0.0049504950 0.0049504950 0.0048780488
## [41] 0.0046296296 0.0045454545 0.0044444444 0.0044444444 0.0044052863
## [46] 0.0043859649 0.0042553191 0.0042016807 0.0041841004 0.0041322314
## [51] 0.0040816327 0.0040816327 0.0040816327 0.0040485830 0.0040160643
## [56] 0.0039062500 0.0038910506 0.0038910506 0.0038167939 0.0035460993
## [61] 0.0034965035 0.0029239766
##
## $ix
##  [1] 37 41 38 21 15  2  8 29 34  9 51  1 46 53 60 16 48 19 44 40 22 24 17
## [24] 39 43 52 55 30 31 20 28 35 11 25 45 18  4 58 62 42  3 27 14 56 13  7
## [47] 54 47 26 10  5 12 59 50 36  6 23 32 49 33 57 61
```

```
gf_histogram(~ betweenness(dolphins, normalized = TRUE))
```



```r
favstats(~ betweenness(dolphins, normalized = TRUE))
```

```
##  min           Q1      median          Q3       max        mean
##    0 0.0030825277 0.021630059 0.056086412 0.2482372 0.039282567
##           sd  n missing
##  0.050964657 62       0
```

```r
length(unique(betweenness(dolphins, normalized = TRUE)))
```

```
## [1] 54
```

```r
sort(betweenness(dolphins, normalized = TRUE), method = "shell", index.return = TRUE, decreasing = TRUE
```

```
## $x
##  [1] 0.24823719603 0.21332443553 0.14314951834 0.13856978866 0.11823861927
##  [6] 0.11430016292 0.10264573972 0.09912164676 0.08467725475 0.08420468343
## [11] 0.07051677854 0.06675695466 0.06552928250 0.06283060443 0.06197200485
## [16] 0.05716644012 0.05284632844 0.04535223884 0.04218278564 0.04067044060
## [21] 0.03341103493 0.03329222098 0.03305046077 0.03278688525 0.03269475970
## [26] 0.02937253675 0.02923686049 0.02915795194 0.02325160067 0.02320147612
## [31] 0.02236573760 0.02089438036 0.02033277469 0.01923434489 0.01908259621
## [36] 0.01609202091 0.01485489972 0.01419498261 0.01331439417 0.01270065393
## [41] 0.01203780585 0.00907281243 0.00738304349 0.00438030018 0.00436247723
## [46] 0.00330470986 0.00300846694 0.00237379651 0.00164411484 0.00119307832
## [51] 0.00092896175 0.00087746953 0.00013661202 0.00000000000 0.00000000000
## [56] 0.00000000000 0.00000000000 0.00000000000 0.00000000000 0.00000000000
## [61] 0.00000000000 0.00000000000
##
## $ix
##  [1] 37  2 41 38  8 18 21 55 52 58 40 29 30 44 15 34 14 39 24 46 51 16 31
## [24] 33 35  7 28 43 42 48  9 10 60 53  1 11 19 62 20 22 45  3 25  6 27 17
## [47] 47  4 26 54 50 56 57  5 12 13 23 32 36 49 59 61
```

c. (4 points) How dense and connected is the network? Use at least 2 appropriate descriptive statistics to write sentences to address this question.

SOLUTION:

The network has a desity of .08, showing that it is infact rather dense. The average path length is 3. 35

telling us that the average number of edges from one dolphin to another is about 3. The diameter is the shortest, longest path. This tells us that the longest number of edges between any two dolphins is 8 edges.

```
graph.density(dolphins)
```

```
## [1] 0.084082496
```

```
average.path.length(dolphins)
```

```
## [1] 3.356954
```

```
diameter(dolphins)
```

```
## [1] 8
```

d. (8 points) What communities are present in the dolphin network? Use at least two algorithms to address this question. Present your findings and compare the solutions the two algorithms give you.

SOLUTION:

The leading eigenvector community produces five communities ranging in size from 8 members to 17 members. The fast and greedy community found 4 communities, one of which had only two members. The other tree ranged from 15 to 23 members. There is a good amount of overlap in the two communities, but they do seem to be diffeent enough to call to different communitiy clusters.

```
set.seed(240)
sp1 <- leading.eigenvector.community(dolphins)
length(sp1)
```

```
## [1] 5
```

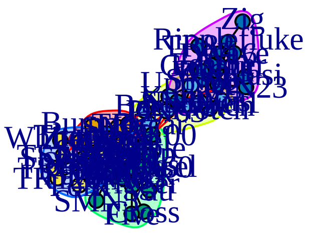
```
sizes(sp1)
```

```
## Community sizes
##  1  2  3  4  5
##  8  9 17 14 14
```

```
membership(sp1)
```

```
##  [1] 1 2 1 3 3 5 5 2 3 5 1 3 4 5 4 3 4 5 3 2 1 3 5 3 3 2 2 2 2 3 2 5 5 4 4
## [36] 3 3 3 4 5 4 5 1 4 1 3 4 1 5 4 4 3 4 4 2 3 5 5 4 3 5 1
```

```
plot(sp1, dolphins)
```



```
set.seed(240)
fg1 <- fastgreedy.community(dolphins)
length(fg1)
```

```
## [1] 4
```

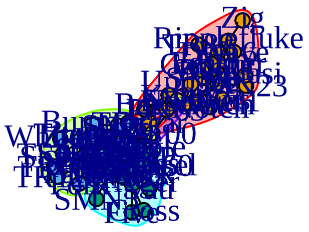
```
sizes(fg1)
```

```
## Community sizes
##  1  2  3  4
## 22 23 15  2
```

```
membership(fg1) #this is the clustering vector you are used to
```

```
##  [1] 2 1 2 3 3 1 1 1 3 1 2 3 2 1 2 3 2 1 3 1 2 3 1 3 3 1 1 1 1 3 1 1 1 2 2
## [36] 3 4 2 2 4 2 1 2 2 2 3 2 2 1 2 2 3 2 2 1 3 1 1 2 3 1 2
```

```
plot(fg1, dolphins) #how to plot the solution
```



e. (4 points) Which community structure from part d do you prefer for the dolphin network? Why? Explain using at least one appropriate statistic as support for your choice.

SOLUTION:

Modularity is a statistic that can be used in the clustering to find how well it does. Large values indicate a better fit. The modualrities for the fast and greedy and the eigen communities are almost the same, however the eigen communities seem to do slightly better, so I believe that it is the better community structure.

```
modularity(dolphins, fg1$membership)
```

```
## [1] 0.49549068
```

```
modularity(dolphins, sp1$membership)
```

```
## [1] 0.49119892
```

Question 4 - 26 points

In this question, we will examine course evaluation data from the University of Texas. A separate pdf with a data dictionary has been provided for you to review. Analysts have a few questions they want you to address using the data.

```
courseeval <- read_csv("https://awagaman.people.amherst.edu/stat240/courseeval.csv")
```

a. (6 points) In what ways are student evaluations typically the most different? In other words, the analysts want to know how student evaluations are most likely to vary from one to the next. Perform a PCA to help address this question. Report your findings, list choices you made in performing the PCA such as the number of PCs kept, and associated rationale.

I ran a PCA on the covariance matrix and it outputted 12 different PCs The covariance matrix was chosen because all of the variables were on the same scale, rangeing from 1-5. Looking at the skree plor, it appears as though there is an elbow at 2 PCs so I used only those two.

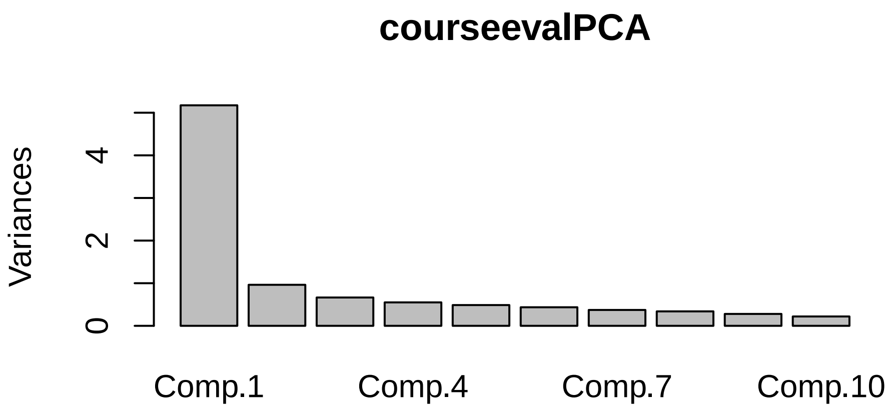
```
eigen(cov(courseeval))
```

```
## eigen() decomposition
## $values
##  [1] 5.17773039 0.96318778 0.66515383 0.55041560 0.48700147 0.43563548
##  [7] 0.37285689 0.33937079 0.27945420 0.21954245 0.20602088 0.16317499
##
## $vectors
##              [,1]          [,2]          [,3]         [,4]          [,5]
##  [1,] -0.21319324 -0.3386215866 -0.0329355858 -0.189166645 -0.0484891574
##  [2,] -0.20340784 -0.2837535965 -0.0593538487 -0.139417716 -0.0236746676
##  [3,] -0.22337013 -0.2509579941 -0.0755951894 -0.165492177 -0.0213309386
##  [4,] -0.23588432 -0.2873792534 -0.0525017411 -0.407150455 -0.0811537998
##  [5,] -0.31042600 -0.2133025415 -0.0946022638 -0.191185537  0.0322394528
##  [6,] -0.36048017  0.3054526717 -0.2817549810  0.071912183  0.1714614379
##  [7,] -0.28872312  0.4287584808 -0.3035089516 -0.038606299  0.4852864123
##  [8,] -0.24388618  0.2404639902 -0.4511851787  0.148749286 -0.7789598417
##  [9,] -0.33735429 -0.0059754329  0.0049605987 -0.052368888  0.2407546934
## [10,] -0.34796442  0.4476086375  0.7173835567 -0.304224552 -0.2361566761
## [11,] -0.34375475 -0.2091612878  0.1452712568  0.426217063  0.0594107897
## [12,] -0.29228754 -0.1861688770  0.2590995610  0.639783867 -0.0080733853
##              [,6]         [,7]         [,8]          [,9]         [,10]
##  [1,]  0.109809153 -0.148676813  0.171706487  0.1874372561  0.2677420498
##  [2,]  0.167720897 -0.271787272  0.295817767  0.1182937275  0.2931973497
##  [3,]  0.047024168 -0.276988963  0.304908691 -0.0328658611 -0.0606547317
##  [4,]  0.169273372  0.563540708 -0.288690182  0.3756048243 -0.3155571602
##  [5,] -0.044634747  0.119774378 -0.274271691 -0.8368841762  0.1178068283
##  [6,]  0.310349420 -0.439276777 -0.577893838  0.1889828783  0.0388323535
##  [7,]  0.162941517  0.368473250  0.484523378 -0.0491901745  0.0208945669
##  [8,] -0.153323732  0.088916076  0.131858660 -0.0042485345 -0.0093593201
##  [9,] -0.870420207 -0.049901577 -0.083760189  0.2256740384  0.0681236902
## [10,]  0.077644299 -0.055054056  0.068456150 -0.0364685807  0.0214979076
## [11,]  0.044375275 -0.169190498  0.160341719 -0.0897655538 -0.7158316728
## [12,]  0.122272795  0.352974128 -0.089163114  0.0993517018  0.4535037263
##               [,11]          [,12]
##  [1,]  0.54055391777  0.58137858883
##  [2,]  0.09982274952 -0.74508994936
##  [3,] -0.76807208239  0.29490882230
##  [4,] -0.07956241414 -0.10049268396
##  [5,]  0.06914317378 -0.03161277329
##  [6,] -0.04061038365  0.02612726779
##  [7,]  0.05132270541  0.03344309659
##  [8,]  0.03574129842 -0.00630243190
##  [9,]  0.00056681496 -0.05214634066
## [10,]  0.00821547157  0.00032104165
## [11,]  0.22925041952 -0.04957025327
## [12,] -0.19648655634  0.04187115912
```

```
courseevalPCA<- princomp(courseeval, cor = FALSE, scores = TRUE)
summary(courseevalPCA)
```

```
## Importance of components:
##                            Comp.1      Comp.2      Comp.3      Comp.4
## Standard deviation     2.27462902 0.981061747 0.815270835 0.741628188
## Proportion of Variance 0.52514903 0.097690898 0.067462935 0.055825661
## Cumulative Proportion  0.52514903 0.622839931 0.690302866 0.746128527
##                            Comp.5      Comp.6      Comp.7      Comp.8
## Standard deviation     0.69759923 0.659785063 0.610396376 0.582341968
## Proportion of Variance 0.04939391 0.044184137 0.037816847 0.034420534
## Cumulative Proportion  0.79552244 0.839706575 0.877523421 0.911943955
##                            Comp.9     Comp.10     Comp.11     Comp.12
## Standard deviation     0.52844060 0.468381912 0.453728938 0.403801245
## Proportion of Variance 0.02834352 0.022266997 0.020895577 0.016549952
## Cumulative Proportion  0.94028747 0.962554471 0.983450048 1.000000000
```

```r
plot(courseevalPCA)
```



```r
courseevalPCA$loadings
```

```
##
## Loadings:
##        Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6 Comp.7 Comp.8 Comp.9
## ITEM13  0.213  0.339         0.189         0.110  0.149  0.172  0.187
## ITEM14  0.203  0.284         0.139         0.168  0.272  0.296  0.118
## ITEM15  0.223  0.251         0.165                0.277  0.305
## ITEM16  0.236  0.287         0.407         0.169 -0.564 -0.289  0.376
## ITEM17  0.310  0.213         0.191               -0.120 -0.274 -0.837
## ITEM18  0.360 -0.305  0.282        -0.171  0.310  0.439 -0.578  0.189
## ITEM19  0.289 -0.429  0.304        -0.485  0.163 -0.368  0.485
## ITEM20  0.244 -0.240  0.451 -0.149  0.779 -0.153         0.132
## ITEM21  0.337                      -0.241 -0.870                0.226
## ITEM22  0.348 -0.448 -0.717  0.304  0.236
## ITEM23  0.344  0.209 -0.145 -0.426                0.169  0.160
## ITEM24  0.292  0.186 -0.259 -0.640         0.122 -0.353
##        Comp.10 Comp.11 Comp.12
## ITEM13  0.268   0.541   0.581
## ITEM14  0.293          -0.745
## ITEM15         -0.768   0.295
## ITEM16 -0.316          -0.100
## ITEM17  0.118
## ITEM18
## ITEM19
## ITEM20
## ITEM21
## ITEM22
## ITEM23 -0.716   0.229
## ITEM24  0.454  -0.196
##
##                Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6 Comp.7 Comp.8
## SS loadings     1.000  1.000  1.000  1.000  1.000  1.000  1.000  1.000
## Proportion Var  0.083  0.083  0.083  0.083  0.083  0.083  0.083  0.083
## Cumulative Var  0.083  0.167  0.250  0.333  0.417  0.500  0.583  0.667
##                Comp.9 Comp.10 Comp.11 Comp.12
## SS loadings     1.000   1.000   1.000   1.000
## Proportion Var  0.083   0.083   0.083   0.083
## Cumulative Var  0.750   0.833   0.917   1.000
```

SOLUTION:

b. (4 points) Interpret the PCs you have chosen to keep.

SOLUTION: PC1 has small, positive loadings on all variables. This may just be an indicator of how good the coure was overall. PC2 had positive loadings on all variables except for items 18, 20, and 22. These instead had negative loadings. These variables seem to indicate how well the teacher interacts with the students, so this PC, because they are negative indicates howe poorly a teacher interacts with students.

c. (6 points) Are students picking up on some underlying instructor or course characteristic in their evaluations? Perform a factor analysis to help address this question. Report your findings, list choices you made in performing the factor analysis including the final number of factors and chosen rotation, and associated rationale.

We concluded that the maximum number of factors that we could use was 7, however, after doi9ng hypothesis tests, we determined that we did not need more than 5 because it had a p-value of .351.

SOLUTION:

```
q <- 12 #number of variables
k <- 7 #number of factors
df <- (q*(q+1)/2)-(q*(k+1)-(k*(k-1)/2))
df
```

```
## [1] 3
```

```
FAeval <- factanal(courseeval, factors = 5)
print(FAeval)
```

```
##
## Call:
## factanal(x = courseeval, factors = 5)
##
## Uniquenesses:
## ITEM13 ITEM14 ITEM15 ITEM16 ITEM17 ITEM18 ITEM19 ITEM20 ITEM21 ITEM22
##  0.355  0.307  0.393  0.071  0.378  0.274  0.438  0.629  0.334  0.557
## ITEM23 ITEM24
##  0.268  0.209
##
## Loadings:
##        Factor1 Factor2 Factor3 Factor4 Factor5
## ITEM13  0.185   0.703   0.211   0.251
## ITEM14  0.225   0.767   0.183   0.143
## ITEM15  0.294   0.664   0.175   0.168   0.147
## ITEM16  0.230   0.392   0.159   0.832
## ITEM17  0.428   0.492   0.258   0.288   0.217
## ITEM18  0.775   0.275   0.195   0.105
## ITEM19  0.713   0.147   0.123   0.107
## ITEM20  0.542   0.203   0.149   0.103
## ITEM21  0.509   0.343   0.259   0.156   0.444
## ITEM22  0.546   0.180   0.258   0.137   0.165
## ITEM23  0.407   0.475   0.553           0.160
## ITEM24  0.303   0.272   0.774   0.152
##
##                Factor1 Factor2 Factor3 Factor4 Factor5
## SS loadings      2.625   2.507   1.314   0.996   0.345
## Proportion Var   0.219   0.209   0.110   0.083   0.029
## Cumulative Var   0.219   0.428   0.537   0.620   0.649
```

```
##
## Test of the hypothesis that 5 factors are sufficient.
## The chi square statistic is 17.55 on 16 degrees of freedom.
## The p-value is 0.351
```

d. (4 points) Interpret the factors you have chosen to keep. If you felt no solution was appropriate above, chose one to use for practice interpreting.

SOLUTION: Factor 1 has high ;loadings on items 17-23, which allows it to be interpretted as how good the course was. Factor 2 has high loadings on 13,14,15,17, and 23, allowing it to be interpretted as to how good the instructor was. Factor 3 has high loadings on items 23 and 24, allowing it to be interpretted as quality in comparison to other classes. Factor 4 has high loadings on item 16, allowing it to be interpretted as how focused the instructoer was. Factor 5 has its highest, though not very high, loading on item 21, allowing it to be interpretted as how aware the insructor was.

e. (6 points) Compare and contrast your PCA and FA solutions. What do you learn from each? Which do you think provides more insights into this data set? Explain your response.

SOLUTION: PCA gave us 2 PCs and FA gave us 5 factors. It seems as though the first PC agrees in interpretation with the first factor. However, the interprettation varies with PC and Factor 2. PC tells us about how the instructor gets along with students while factor analysis tell howgood the instructor was. I believe that the factor analysis is more useful in this situation because it tells us more about the profeesor, however, PC2 can be hlepful if one were curious as to the relationship between intructor and students.

Question 5 - 22 points

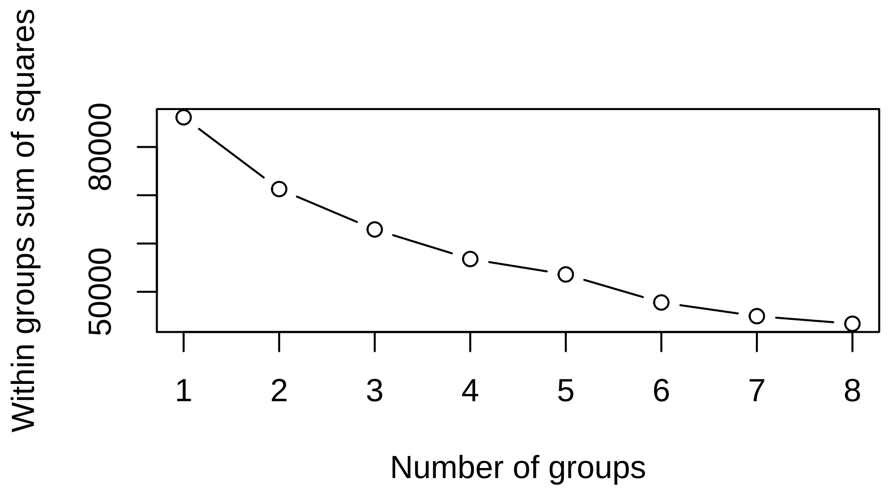
This problem uses the letters data set from question 1. Investigators want to know which letters, if any, naturally group together based on their pixel-based characteristics.

a. (10 points) Use two different algorithms to find natural groupings of the letters in the data set. Do you find strong natural groupings? Report your findings, including what algorithms you used, parameter values for those algorithms (and how you chose them), how many groups you find, and compare the solutions from the two algorithms. Which solution do you prefer? Why?

SOLUTION:

To find natural groupings I used hierarchical clustering and k-means. There doesnt seem to se any naturally occuring groups, however,the groups that were made no not seem to overlap too much. Aftr looking at at the elbow plot, the "elbow" appeared to be at 6 clusters. Therfore, for both k means and average hieracrhical clustering, I seperaterd them into 6 clusters.

```r
set.seed(240)
n <- nrow(letters)
wss <- rep(0, 8)
for(i in 1:8){wss[i] <- sum(kmeans(scale(letters[, -1]), centers = i)$withinss)}
plot(1:8, wss, type = "b", xlab = "Number of groups", ylab = "Within groups sum of squares")
```



```r
Ksol1 <- kmeans(scale(letters[, -1]), centers = 6)
Ksol1$centers
```

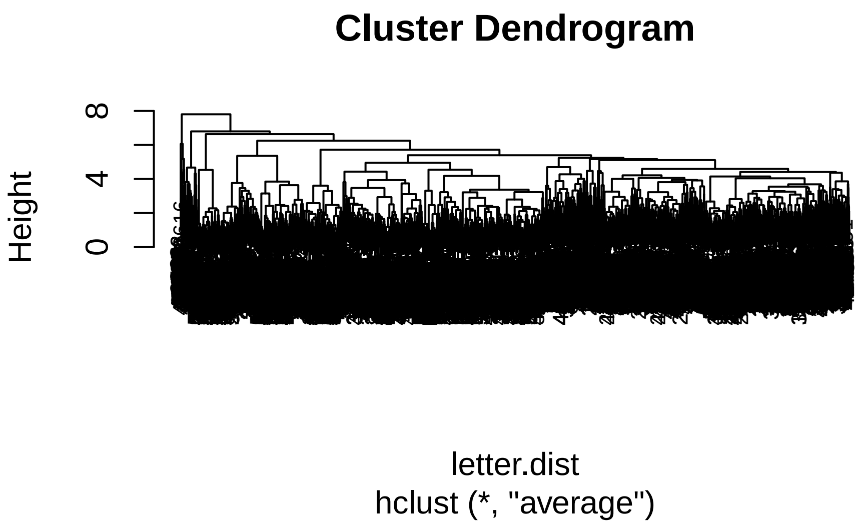
```
##          xbox        ybox       width       height       onpix
## 1 -0.52507917 -0.47738607 -0.60813896 -0.442342790 -0.39861051
## 2 -0.78757387 -0.31369649 -0.72160920 -0.213147048 -0.87444480
## 3  0.98749951  0.77832046  1.12336257  0.774397958  1.18317055
## 4 -0.45154292 -0.11062307 -0.20175255 -0.209615763 -0.51005832
## 5  0.47761057  0.27513769  0.10033148  0.072515337 -0.26086482
## 6 -0.28640687 -0.51231755 -0.48305206 -0.335541702 -0.39287560
##           xbar        ybar       x2bar        y2bar        xybar
## 1 -0.353367157  0.12188589  0.17879323  0.570632559 -0.241551882
## 2 -0.021693230  2.11427258 -1.04792950  0.388861607 -0.629006370
## 3  0.169831142 -0.10913211 -0.13898016  0.051987259  0.055532573
## 4  1.002564226 -1.39300647 -0.67984346 -1.062534650  0.098520896
## 5 -0.976518212  1.08099665 -0.53701155  0.765035147  1.580310581
## 6 -0.011036372 -0.25535992  2.04096057 -1.221736316 -0.596355613
##         x2ybar       xy2bar        xege        xegvy        yege
## 1 -0.058572009  0.340823547 -0.50678126  0.149808661  0.62863633
## 2  1.915787365 -0.078365853 -1.03294245  1.009409212 -1.33974466
## 3 -0.111810136  0.012422992  0.81687216 -0.056045123  0.48991704
## 4 -1.462509336  0.088514926 -0.24520345 -0.891297626 -0.57079319
## 5  1.041169808 -1.250618855 -0.56680997  1.218196787 -0.28552437
## 6  0.587949501 -0.041339601  0.91522090 -0.688949178 -1.29779787
##          yegvx
## 1  0.383799422
## 2 -0.010792660
## 3  0.070891370
## 4  0.114311663
## 5 -1.676920973
## 6 -0.028034729
```

```r
list(Ksol1)
```

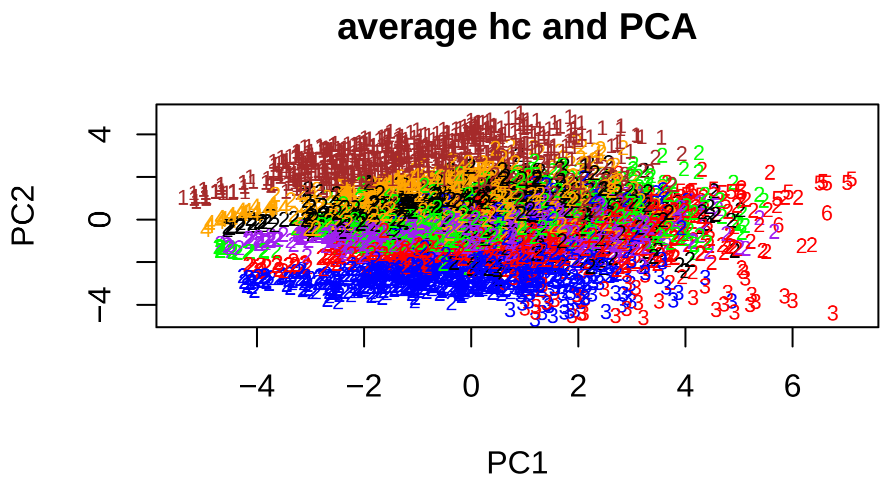
```
## [[1]]
## K-means clustering with 6 clusters of sizes 1689, 345, 1475, 841, 495, 540
##
## Cluster means:
##          xbox        ybox       width       height       onpix
## 1 -0.52507917 -0.47738607 -0.60813896 -0.442342790 -0.39861051
## 2 -0.78757387 -0.31369649 -0.72160920 -0.213147048 -0.87444480
## 3  0.98749951  0.77832046  1.12336257  0.774397958  1.18317055
## 4 -0.45154292 -0.11062307 -0.20175255 -0.209615763 -0.51005832
## 5  0.47761057  0.27513769  0.10033148  0.072515337 -0.26086482
## 6 -0.28640687 -0.51231755 -0.48305206 -0.335541702 -0.39287560
##          xbar        ybar       x2bar        y2bar        xybar
## 1 -0.353367157  0.12188589  0.17879323  0.570632559 -0.241551882
## 2 -0.021693230  2.11427258 -1.04792950  0.388861607 -0.629006370
## 3  0.169831142 -0.10913211 -0.13898016  0.051987259  0.055532573
## 4  1.002564226 -1.39300647 -0.67984346 -1.062534650  0.098520896
## 5 -0.976518212  1.08099665 -0.53701155  0.765035147  1.580310581
## 6 -0.011036372 -0.25535992  2.04096057 -1.221736316 -0.596355613
##        x2ybar       xy2bar        xege        xegvy        yege
## 1 -0.058572009  0.340823547 -0.50678126  0.149808661  0.62863633
## 2  1.915787365 -0.078365853 -1.03294245  1.009409212 -1.33974466
## 3 -0.111810136  0.012422992  0.81687216 -0.056045123  0.48991704
## 4 -1.462509336  0.088514926 -0.24520345 -0.891297626 -0.57079319
## 5  1.041169808 -1.250618855 -0.56680997  1.218196787 -0.28552437
## 6  0.587949501 -0.041339601  0.91522090 -0.688949178 -1.29779787
##         yegvx
## 1  0.383799422
## 2 -0.010792660
## 3  0.070891370
## 4  0.114311663
## 5 -1.676920973
## 6 -0.028034729
## 
## Clustering vector:
##    [1] 2 1 4 3 3 3 5 1 1 3 3 3 1 1 3 1 1 1 1 1 1 3 1 6 4 3 3 1 1 6 5 3 2 5
##   [35] 1 3 4 4 3 6 4 1 4 3 2 6 4 2 4 3 3 1 5 4 1 6 1 1 3 1 1 5 1 1 1 3 3 1
##   [69] 3 4 1 1 4 4 1 1 1 3 1 3 1 3 2 5 2 1 1 3 3 3 3 1 3 1 4 6 3 4 1 3 2 1
##  [103] 4 1 3 5 4 3 1 3 1 3 3 5 1 3 2 4 2 3 5 6 6 6 1 6 1 1 2 6 4 4 4 3 1 1
##  [137] 3 4 1 3 4 6 1 1 3 1 2 1 3 3 3 1 3 4 3 3 5 3 3 6 3 6 3 5 4 3 5 1 2 3
##  [171] 5 3 5 4 4 3 1 1 3 1 6 5 1 2 4 1 1 3 1 4 3 1 5 1 3 1 1 1 1 4 3 1 4 1
##  [205] 6 3 3 1 4 4 1 3 3 3 1 6 2 4 4 6 3 3 2 1 1 4 1 1 5 1 1 1 4 2 1 6 3 1
##  [239] 6 1 6 5 1 3 3 3 3 5 6 5 4 3 3 3 1 3 3 2 1 1 1 1 1 6 3 1 1 3 5 2 3 4
##  [273] 6 3 3 1 2 3 1 1 3 3 1 2 1 1 1 1 5 4 6 2 3 4 6 4 4 4 6 3 3 5 4 4 6 3
##  [307] 3 6 1 3 4 3 6 4 1 3 6 1 1 1 3 1 6 1 6 3 5 1 3 4 1 5 1 3 2 3 1 5 1 1
##  [341] 5 6 1 1 1 1 3 3 2 2 6 3 3 4 3 1 6 4 3 5 4 3 3 6 3 4 1 1 3 6 1 1 2 6
##  [375] 1 1 1 4 5 3 6 3 1 3 1 4 4 6 3 2 1 4 2 5 1 3 3 5 2 4 1 6 1 4 3 2 1 3
##  [409] 4 5 5 1 4 1 1 3 3 4 3 3 5 2 6 1 1 4 1 3 1 6 4 1 4 4 2 1 1 4 1 5 5 5
##  [443] 6 5 1 6 1 4 4 3 5 1 4 6 6 5 1 3 4 6 5 5 6 5 3 5 1 4 3 2 1 6 4 3 3 3
##  [477] 4 4 6 1 1 4 2 6 1 1 1 1 1 2 1 3 3 2 5 4 1 3 6 1 3 3 5 1 3 1 2 3 1 4
##  [511] 3 3 5 1 2 6 6 1 4 3 4 3 1 6 6 3 3 6 5 6 2 1 4 3 6 3 3 3 6 1 1 5 4 4
##  [545] 1 4 6 5 5 1 1 6 1 1 6 1 4 4 1 4 4 2 3 3 1 3 1 5 4 3 4 1 3 5 3 1 1 1
##  [579] 5 3 1 1 1 1 1 5 3 3 1 3 1 3 3 2 1 6 1 3 4 3 3 3 4 3 4 1 5 4 4 1 1 1
##  [613] 4 6 1 5 1 2 3 6 6 4 1 4 1 3 1 3 3 1 3 1 1 1 6 1 1 6 3 3 1 3 4 2 4 4
##  [647] 5 3 6 2 6 1 4 3 6 4 1 1 1 5 1 4 1 3 2 4 4 5 3 4 1 1 3 3 1 3 4 4 4 6
##  [681] 3 1 3 5 4 3 2 4 1 1 3 1 2 3 3 4 4 3 2 6 3 3 1 4 3 3 4 2 1 1 3 6 3 3
##  [715] 4 3 1 1 3 1 1 4 3 3 4 1 1 5 3 3 3 3 4 3 1 3 6 5 3 6 4 1 3 1 3 1 6 1
##  [749] 1 1 5 1 4 3 1 4 4 6 3 6 3 6 3 3 5 3 5 1 1 5 1 1 1 2 4 1 6 2 4 5 3 6
##  [783] 6 2 1 1 3 1 1 1 1 4 5 1 1 3 1 3 3 5 3 3 1 6 3 3 4 4 3 5 4 3 1 6 1 1
##  [817] 5 6 1 4 5 3 6 4 3 4 2 3 3 6 1 2 4 1 1 3 5 3 4 4 1 1 1 1 3 4 4 6 6 6
##  [851] 2 2 1 1 1 4 4 6 1 5 1 2 5 5 2 3 3 1 6 4 5 4 1 3 3 1 1 3 3 3 3 2 1 1
##  [885] 1 3 1 2 1 2 4 4 6 5 4 2 4 6 1 1 3 3 6 6 1 1 6 3 2 3 2 1 5 3 3 6 1 1
##  [919] 3 5 3 1 4 6 3 1 2 1 4 3 1 3 3 5 1 1 5 1 5 4 1 6 6 1 3 5 3 1 1 4 4 1
##  [953] 6 3 3 3 4 1 1 5 6 3 1 4 1 3 6 3 3 3 3 6 1 6 6 2 1 1 4 2 5 5 3 6 1 1
##  [987] 3 4 3 2 1 3 3 1 5 1 2 5 3 4 3 3 3 1 3 1 6 5 1 1 1 1 1 1 4 5 6 6 3 3
## [1021] 1 5 4 2 3 1 3 4 4 6 3 1 1 4 4 4 2 4 3 2 4 6 3 4 5 5 6 3 1 3 2 4 4 1
## [1055] 6 3 4 1 2 4 1 6 1 5 5 1 6 6 2 6 3 1 6 5 3 1 3 1 3 1 1 1 3 4 3 1 1 3
## [1089] 3 1 1 1 2 6 1 1 2 6 6 1 1 5 5 4 1 3 2 3 1 3 1 4 3 1 3 6 6 4 2 1 1 1
## [1123] 3 5 4 5 6 4 3 1 6 3 1 2 3 1 1 3 4 2 4 3 5 1 4 1 3 4 1 3 2 5 3 6 6 3
## [1157] 6 3 1 2 6 1 3 1 4 2 3 1 5 5 1 1 5 3 3 6 5 1 3 1 4 1 6 3 1 3 4 4 1 1
## [1191] 5 6 4 3 1 5 5 4 4 6 5 3 1 1 1 6 5 3 6 1 3 1 3 6 6 3 5 1 4 1 1 3 1 1
## [1225] 1 4 1 1 6 3 5 3 1 2 6 1 6 1 3 1 3 4 4 3 3 5 6 6 2 6 2 1 4 6 3 4 6 5
## [1259] 3 3 1 5 6 1 1 1 5 3 1 4 1 5 3 1 5 3 3 6 1 3 2 4 1 3 2 1 1 4 5 1 1 2
## [1293] 2 2 1 2 1 1 1 1 3 3 5 3 5 1 4 1 2 3 1 3 4 1 2 3 3 3 4 5 3 1 3 3 3 1
## [1327] 1 3 3 6 4 1 3 1 6 4 6 2 1 3 3 1 1 1 1 3 3 4 3 3 1 3 1 1 1 1 3 3 1 1
## [1361] 6 3 3 1 2 5 4 2 3 1 1 6 3 3 3 4 4 4 3 1 3 3 3 5 3 2 6 4 1 1 6 3 4 3
## [1395] 4 3 4 5 3 6 1 1 4 1 4 5 5 6 1 1 3 3 2 2 5 2 3 3 1 2 5 5 1 6 3 2 1 1
## [1429] 4 3 3 1 4 3 3 4 1 1 1 5 6 1 1 1 3 4 1 1 3 3 3 5 6 2 1 1 3 2 2 2 4 6
## [1463] 3 4 1 3 3 1 6 6 2 3 3 3 4 3 3 3 1 3 4 1 2 3 2 1 3 1 3 6 1 1 4 1 4 3
## [1497] 3 3 3 3 3 3 3 1 1 3 1 3 3 6 6 1 1 2 1 1 6 3 1 5 1 1 1 4 5 6 6 3 3 5
## [1531] 1 1 4 3 2 1 4 1 1 3 4 3 6 1 6 5 1 1 3 4 6 3 3 4 4 3 1 1 1 5 3 1 5 1
## [1565] 5 2 2 6 4 6 4 1 3 1 1 4 5 5 4 6 1 5 3 3 3 1 4 3 5 3 3 1 3 1 3 3 1 2
## [1599] 1 4 1 3 6 1 3 5 2 1 3 6 1 6 5 1 4 4 4 6 4 3 4 4 4 5 3 4 6 2 1 3 3 1
## [1633] 3 1 1 3 1 3 6 1 5 1 5 1 2 1 3 4 3 3 3 1 1 1 2 1 1 3 3 3 1 3 3 1 1 1
## [1667] 3 5 5 1 1 1 6 5 1 3 1 3 1 1 6 3 1 6 1 6 2 5 1 3 1 1 1 1 1 4 1 3 1 4
## [1701] 4 6 6 4 3 1 1 5 1 1 4 1 4 3 1 5 1 1 6 1 1 3 1 1 5 3 1 1 1 2 1 1 4 3
## [1735] 3 1 3 2 1 1 2 1 2 3 6 6 3 5 4 4 1 3 5 1 1 3 1 5 1 1 3 3 1 5 4 6 5 6
## [1769] 4 3 6 2 4 4 5 3 2 2 1 1 3 1 4 4 3 3 1 3 3 1 5 3 3 3 1 1 5 3 2 4 4 3
## [1803] 2 2 5 4 1 3 5 3 3 4 2 1 1 1 6 3 1 1 1 3 3 3 6 5 6 3 5 1 1 5 1 1 4 1
## [1837] 1 1 3 4 3 1 6 6 3 3 4 4 5 6 3 3 3 5 5 5 1 3 5 2 4 3 3 1 4 6 1 1 4 3
## [1871] 3 3 1 3 3 4 1 5 3 3 5 3 2 3 3 3 3 1 3 1 1 6 1 3 5 3 3 6 4 1 1 4 5 1
## [1905] 6 4 1 3 1 3 1 3 3 1 4 3 6 2 1 3 3 4 1 1 1 4 1 3 3 4 6 6 3 4 2 1 6 4
## [1939] 2 6 5 1 1 3 3 1 4 1 4 3 4 4 4 5 1 3 3 3 3 1 2 5 1 3 2 1 6 5 5 1 3 1
## [1973] 3 1 3 3 3 6 4 3 3 1 6 3 3 3 1 3 1 1 3 5 3 4 1 5 3 5 3 2 3 4 6 2 5 3
## [2007] 4 1 5 4 2 1 3 1 1 1 3 1 3 5 3 2 3 3 3 1 2 1 1 2 1 1 3 3 3 4 1 3 3 3
## [2041] 3 3 3 3 4 1 5 4 3 2 6 1 1 2 4 1 1 6 4 3 1 1 4 4 1 5 1 4 3 4 4 1 4 3
## [2075] 6 6 2 1 2 1 1 3 3 1 5 4 4 3 2 1 1 3 1 4 1 1 1 1 1 6 4 1 5 4 3 5 6 6
## [2109] 1 3 1 1 4 3 4 1 1 6 3 4 6 3 4 3 3 3 3 1 5 2 3 1 1 3 4 3 6 5 3 3 4 1
## [2143] 5 2 5 3 6 1 4 4 4 3 1 3 1 3 5 2 4 4 4 3 1 1 1 1 3 5 3 3 1 4 3 1 2 5
## [2177] 1 3 4 4 3 1 1 2 1 6 1 1 1 4 4 1 1 6 2 3 3 4 3 5 3 5 6 4 1 4 2 5 1 3
## [2211] 3 5 5 3 6 5 1 1 5 5 3 6 4 3 3 1 4 1 1 4 3 3 1 3 5 3 1 3 1 5 4 1 1 1
## [2245] 1 1 3 1 4 3 4 1 6 3 3 3 3 2 4 1 4 1 3 5 3 4 2 1 1 6 4 6 6 6 1 5 4 5
## [2279] 4 3 4 1 3 1 2 1 3 4 3 4 6 4 1 4 3 4 5 4 1 3 4 3 4 1 3 4 3 3 5 4 3 3
## [2313] 1 1 3 3 4 5 1 6 1 2 4 1 1 1 2 1 1 3 3 4 1 3 2 3 6 3 1 2 1 6 5 1 3 3
## [2347] 3 6 1 3 1 3 3 3 3 1 1 5 1 1 5 3 1 1 4 4 3 3 4 4 4 5 1 1 6 2 3 3 4 6
## [2381] 1 1 2 1 5 3 5 1 1 3 3 3 3 5 1 6 3 5 1 4 4 1 1 4 5 3 5 3 1 6 1 1 4 2
## [2415] 1 6 3 4 5 3 4 3 3 6 3 3 3 1 2 5 2 6 4 6 3 3 1 5 4 1 3 3 3 3 3 3 4 3
## [2449] 3 1 1 5 1 6 5 1 6 2 6 6 6 1 3 3 3 5 1 3 3 1 1 3 3 1 4 3 1 3 4 4 6 2
## [2483] 4 3 5 2 4 3 3 4 5 6 3 1 3 5 4 6 3 3 2 1 4 4 2 4 3 1 2 1 4 6 5 4 4 1
## [2517] 2 3 1 2 5 6 3 3 3 4 3 6 5 6 2 3 1 4 3 1 3 2 4 1 3 3 6 3 3 6 1 1 4 4
## [2551] 4 3 1 1 5 3 1 5 3 5 3 1 3 3 1 3 2 1 5 3 6 3 3 1 3 3 2 3 4 4 1 3 3 1
## [2585] 5 6 6 3 2 4 1 4 6 3 4 4 6 3 1 1 4 6 4 3 1 1 2 1 1 1 1 6 2 3 1 3 1 1
## [2619] 1 1 3 1 6 3 4 6 1 4 5 3 1 3 4 3 6 1 3 6 4 1 3 6 6 6 1 2 4 1 3 3 3 3
## [2653] 5 1 6 4 1 3 3 6 4 4 3 1 3 1 5 4 1 5 1 6 1 1 3 1 1 1 6 2 1 1 3 1 1 3
## [2687] 6 6 3 3 4 1 4 1 1 1 3 3 1 1 6 1 3 4 1 2 3 6 6 3 6 4 4 3 1 3 1 5 3 4
## [2721] 1 2 3 3 3 3 3 5 4 1 3 1 5 1 3 1 2 3 6 1 4 4 5 3 1 1 1 4 4 5 3 1 2 6
## [2755] 3 1 1 6 1 6 4 3 3 4 1 3 6 3 3 1 1 4 1 3 1 6 1 2 2 5 2 6 3 1 6 2 1 6
## [2789] 3 3 6 4 4 3 4 2 4 3 5 3 4 1 3 5 5 4 2 1 1 1 1 5 3 1 4 1 1 3 1 3 1 3
## [2823] 1 3 1 6 3 3 2 6 3 1 3 4 1 1 2 6 3 5 3 6 6 3 3 5 4 3 3 3 3 3 1 5 6 3
## [2857] 3 6 1 4 6 1 6 3 2 3 1 1 5 3 1 1 4 3 3 4 2 3 1 1 3 1 3 3 1 4 1 1 3 3
## [2891] 1 5 6 5 3 3 5 1 1 3 4 3 3 3 1 1 6 1 2 1 1 2 1 3 3 4 6 5 4 4 6 4 3 1
## [2925] 3 6 1 1 1 4 3 3 5 1 4 1 3 3 1 3 4 4 5 6 5 3 1 6 1 6 1 1 1 3 4 3 3 4
## [2959] 4 4 1 5 3 1 1 1 5 4 3 4 3 3 4 5 6 6 3 3 1 4 1 5 6 4 1 1 2 6 3 6 1 1
## [2993] 1 6 1 1 3 2 1 6 3 4 1 1 1 3 3 3 5 4 3 4 4 3 5 2 1 4 3 3 1 5 1 1 4 1
## [3027] 1 3 2 3 1 4 1 2 1 4 1 4 5 1 4 3 6 6 3 6 4 4 1 1 4 1 5 3 4 6 3 3 6 3
## [3061] 4 2 6 3 1 3 1 3 5 1 3 6 1 3 3 3 1 4 2 1 1 1 3 6 3 3 6 1 3 2 3 1 3 1
## [3095] 3 6 3 1 1 6 1 2 4 3 1 1 1 3 3 3 2 4 5 3 3 6 5 1 5 3 5 3 1 5 4 2 1 3
## [3129] 3 3 3 3 3 3 3 4 1 1 5 3 6 5 6 1 1 1 4 3 2 5 4 3 5 3 1 1 1 3 1 3 4 1
## [3163] 4 3 1 4 2 4 3 1 1 1 1 3 4 3 2 3 6 1 3 1 4 2 1 5 3 4 3 1 1 4 3 3 4 1
## [3197] 3 3 1 1 3 1 1 1 3 3 3 2 1 1 2 1 3 1 1 1 4 5 1 6 1 3 6 5 1 4 6 4 1 3
## [3231] 3 1 1 4 6 3 1 3 4 3 1 4 5 4 3 6 3 1 1 1 4 6 4 1 1 5 6 1 3 1 4 3 2 1
## [3265] 1 6 4 4 1 5 4 1 1 3 4 4 6 6 1 4 3 5 3 3 4 6 6 1 5 6 1 6 3 1 2 5 3 3
## [3299] 1 1 2 1 2 1 2 2 3 1 4 3 1 2 6 6 4 1 5 1 1 3 3 1 1 1 6 1 1 1 3 1 1 3
## [3333] 6 1 4 1 6 4 4 1 4 3 3 5 6 3 2 4 2 4 1 4 1 3 3 2 4 1 1 1 1 3 3 1 1 1
## [3367] 6 1 1 5 3 6 3 1 4 1 3 1 4 3 5 4 4 1 4 3 1 1 3 1 1 4 3 1 5 6 1 3 1 3
## [3401] 2 1 1 1 2 1 1 2 1 5 1 3 1 4 2 6 3 3 1 1 4 4 1 1 3 4 4 1 4 6 4 1 1 3
## [3435] 1 5 4 4 3 1 6 5 3 6 1 1 1 6 4 1 1 3 1 3 1 2 1 1 3 4 4 1 3 6 1 3 4 4
## [3469] 5 4 6 1 5 6 5 4 4 1 1 5 3 2 1 1 2 1 1 3 4 6 1 1 3 4 1 3 2 1 4 3 1 1
## [3503] 6 5 3 3 1 3 4 1 3 6 5 3 1 2 1 1 4 3 6 1 3 4 1 6 3 6 1 4 3 6 1 3 5 3
## [3537] 1 5 4 4 2 2 1 4 3 3 1 5 3 4 1 5 1 2 1 3 5 3 1 2 2 1 4 3 2 5 2 4 3 1
## [3571] 5 1 6 4 3 4 4 3 2 2 1 3 2 3 4 1 6 4 4 5 4 4 1 3 3 1 4 2 1 6 2 1 3 1
## [3605] 3 1 5 1 1 2 5 3 3 4 1 3 1 4 1 1 6 3 4 3 1 3 4 3 3 1 3 1 1 3 6 3 3 1
## [3639] 3 4 4 4 4 5 3 3 4 3 1 1 1 3 4 3 5 1 1 1 1 4 3 3 1 1 6 1 1 4 3 4 3 5
## [3673] 1 2 3 3 3 3 1 4 6 1 4 4 6 1 1 2 4 1 1 5 3 1 3 4 4 4 6 1 1 5 4 3 4 1
## [3707] 1 3 3 1 1 1 1 4 3 1 6 5 3 1 5 5 4 3 5 1 1 4 5 1 3 5 3 3 4 4 1 1 3 4
## [3741] 1 4 3 5 4 6 1 4 1 4 6 4 4 3 1 4 1 3 1 4 3 1 3 1 1 1 1 5 3 3 3 1 3 1
## [3775] 4 1 1 5 5 3 3 4 1 3 6 1 3 1 1 3 2 5 1 3 6 1 3 4 5 3 1 3 2 5 3 5 1 4
## [3809] 3 2 1 1 1 1 4 5 4 5 3 1 3 4 4 3 3 1 2 5 1 3 6 5 3 6 1 1 1 4 3 3 3 5
## [3843] 6 3 3 6 5 4 1 1 4 5 4 5 6 2 2 1 6 3 3 4 3 2 4 1 1 3 3 1 4 5 2 5 5 3
## [3877] 5 4 4 6 1 3 4 3 5 1 1 5 1 1 6 1 2 4 1 1 1 6 3 1 3 3 1 4 3 1 1 6 6 4
## [3911] 3 1 1 6 5 1 3 4 6 1 4 1 5 4 2 1 1 3 1 6 4 3 1 3 3 4 1 6 1 4 5 6 2 2
## [3945] 1 4 5 3 3 4 4 3 5 1 3 3 4 1 1 1 5 1 1 3 3 1 3 1 4 6 6 3 1 4 4 1 2 2
## [3979] 6 1 5 3 4 1 3 4 3 4 4 1 1 1 4 5 4 1 5 3 1 4 4 4 3 2 1 1 5 1 3 1 1 2
## [4013] 3 6 1 6 1 3 4 1 3 6 2 4 1 1 3 4 3 5 2 3 3 5 3 3 4 1 1 1 2 4 2 4 6 1
## [4047] 3 1 6 3 5 2 3 6 1 1 4 4 3 1 1 1 3 1 2 3 1 6 3 3 1 3 3 3 6 1 4 5 6 2
## [4081] 4 1 4 6 4 1 1 3 5 3 4 1 5 1 1 3 6 5 1 1 6 3 3 4 4 4 4 3 1 2 5 1 6 3
## [4115] 1 6 1 1 4 5 4 3 3 2 3 5 3 5 3 1 3 3 3 6 4 4 6 2 5 3 3 3 1 4 3 5 1 4
## [4149] 3 3 3 2 1 3 3 1 3 1 2 4 3 4 4 3 6 5 1 6 4 5 4 1 1 1 1 1 1 1 1 1 3 1
## [4183] 6 2 6 5 3 3 1 3 4 1 1 1 3 1 4 1 1 4 4 6 1 1 1 1 2 5 3 6 1 4 4 3 6 3
## [4217] 1 1 4 5 1 2 1 3 3 5 6 3 1 2 4 1 3 4 1 1 3 1 6 3 3 4 6 3 1 1 2 6 3 3
## [4251] 6 4 1 4 4 4 1 3 6 3 4 1 1 4 3 4 4 4 3 6 3 5 4 4 3 1 6 3 2 1 1 3 3 1
## [4285] 4 4 1 1 1 6 1 3 2 3 1 1 1 5 5 3 5 1 3 1 2 4 1 3 1 3 1 1 1 3 1 1 1 3
## [4319] 3 6 1 1 4 6 1 4 6 1 5 5 4 6 2 5 3 1 1 3 1 5 1 3 1 1 5 4 1 3 4 4 3 3
## [4353] 5 1 1 3 4 4 3 1 5 6 5 1 1 6 5 3 3 6 3 3 4 1 4 3 3 4 3 3 1 2 5 5 1 6
## [4387] 3 3 3 4 3 6 3 1 4 4 1 1 1 3 4 4 3 3 1 1 1 1 2 4 1 4 1 6 6 3 1 3 1 1
## [4421] 4 3 4 3 4 6 1 1 3 1 4 1 1 4 4 4 5 5 4 1 1 1 1 1 5 3 1 1 3 4 1 1 5 3
## [4455] 1 3 5 5 3 4 5 5 4 4 1 1 1 3 1 1 3 3 4 4 3 4 3 3 3 3 1 3 1 1 4 4 1 5
## [4489] 3 2 6 3 3 4 1 5 1 1 6 3 3 2 4 5 6 3 1 6 2 2 3 1 4 4 3 1 1 3 4 6 3 4
## [4523] 1 4 3 3 6 1 1 5 1 4 3 1 5 1 4 1 1 6 5 3 1 1 6 1 6 1 2 6 3 2 3 1 4 3
## [4557] 6 6 6 1 3 1 1 3 3 1 1 1 1 6 4 5 5 4 1 1 5 4 1 4 1 4 6 3 5 3 5 1 6 1
## [4591] 3 3 3 3 5 1 3 3 1 1 3 4 3 6 5 1 1 1 4 4 3 5 1 1 1 3 4 1 1 1 1 6 3 1
## [4625] 5 4 4 4 1 4 2 6 1 6 3 2 3 1 3 1 1 1 5 6 4 3 4 1 1 6 3 1 3 1 3 3 4 3
## [4659] 6 1 3 1 5 1 3 1 3 1 1 1 4 6 1 3 6 3 1 3 5 3 3 3 5 3 3 1 1 3 3 1 3 6
## [4693] 2 1 2 3 4 6 1 1 1 3 1 4 4 3 1 1 3 1 6 3 1 3 1 4 1 6 3 1 1 4 1 1 3 1
## [4727] 3 3 3 3 1 1 4 3 1 1 1 1 6 5 3 1 1 1 3 4 3 1 1 1 1 1 2 5 1 3 6 1 1 3 1
## [4761] 1 2 1 2 1 4 4 3 3 5 6 5 3 5 6 2 1 3 1 6 3 1 5 1 3 3 5 5 1 4 1 1 4 5
## [4795] 3 4 1 5 5 6 2 4 1 1 1 1 3 3 6 6 3 1 4 3 1 1 4 4 1 3 4 1 1 1 3 4 6 3
## [4829] 1 1 5 1 5 1 4 6 1 3 6 4 4 3 4 3 6 3 5 3 3 6 5 1 5 1 6 2 3 6 1 1 2 4
## [4863] 1 4 5 1 1 1 1 1 3 1 2 1 2 3 1 6 3 3 5 3 1 3 1 6 1 6 4 1 1 3 3 3 1 3
## [4897] 3 4 4 4 3 4 1 1 6 5 3 3 6 3 3 4 1 3 2 4 1 4 1 1 1 1 4 5 2 3 2 2 2 1
## [4931] 3 3 3 5 3 5 6 1 6 1 4 6 4 5 1 2 3 3 3 1 3 3 2 1 5 4 3 3 3 1 1 3 4 3
## [4965] 1 3 4 3 6 3 1 2 6 5 3 1 2 1 4 3 4 4 3 5 4 4 1 1 5 4 1 3 3 4 3 4 3 1
## [4999] 1 3 3 3 3 3 1 4 1 5 3 6 1 1 1 4 4 5 3 4 2 5 1 1 5 1 3 3 1 3 5 1 1 3
## [5033] 4 3 3 3 1 3 6 4 4 6 4 6 1 1 3 4 1 3 1 5 2 1 5 5 3 2 1 5 6 3 3 1 1 1
## [5067] 6 3 1 3 6 3 1 3 3 4 1 1 6 1 1 1 1 4 4 1 1 1 1 1 3 5 1 3 4 4 4 1 3 3 1
## [5101] 5 3 2 3 3 4 4 3 1 1 6 3 3 2 3 1 3 6 4 4 4 4 6 6 5 4 1 1 5 3 3 3 1 4
## [5135] 1 5 3 2 5 4 1 3 3 3 4 6 3 3 3 3 3 3 4 3 3 1 2 1 6 4 5 2 5 1 3 6 1 6
## [5169] 2 1 4 1 3 5 1 3 1 3 6 1 3 1 3 3 1 3 4 3 1 3 1 5 3 2 4 4 3 3 3 4 1 1
## [5203] 3 3 4 1 1 3 6 3 6 5 3 5 5 2 3 3 1 1 3 1 1 3 1 1 1 4 3 3 1 3 6 4 1 1 6
## [5237] 2 1 2 3 1 2 1 1 2 1 3 3 1 3 6 3 6 1 6 1 5 1 3 2 6 3 1 1 4 3 1 3 2 3
## [5271] 3 3 4 1 2 3 1 3 5 1 1 6 2 3 1 4 2 3 3 4 3 4 2 1 1 1 5 1 3 1 4 1 2 1
## [5305] 2 3 1 5 3 3 1 5 1 6 6 1 5 3 5 1 1 4 1 3 3 1 4 3 4 3 1 1 6 3 1 6 4 5
## [5339] 5 1 6 3 1 5 1 3 2 2 1 3 4 4 4 6 5 5 3 3 6 5 1 5 1 5 3 4 2 3 1 1 4 3
## [5373] 5 1 4 3 1 5 1 3 5 3 5 1 4
##
## Within cluster sum of squares by cluster:
## [1] 15326.4116  2018.5820 16154.2001  6684.1596  3805.8334  4322.2895
##  (between_SS / total_SS =  43.9 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
## [5] "tot.withinss" "betweenss"    "size"         "iter"
## [9] "ifault"
```

text(cerPCA*sscores*[, 1 : 2], *col* = *y*target, labels = mod1$classification, cex = 0.7)

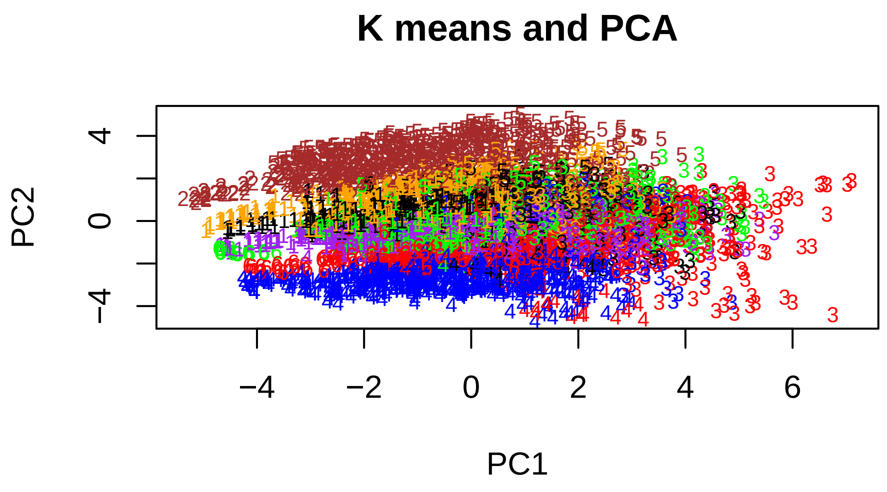
```r
set.seed(240)
x <- array(letters$letter)
y <- x
for(i in 1:5385) {
  if(x[i] == "A"){y[i] <- "blue"}
  if(x[i] == "M"){y[i] <- "red"}
  if(x[i] == "H"){y[i] <- "green"}
  if(x[i] == "E"){y[i] <- "orange"}
  if(x[i] == "R"){y[i] <- "purple"}
  if(x[i] == "S"){y[i] <- "black"}
  if(x[i] == "T"){y[i] <- "brown"}
}

letter.dist <- dist(scale(letters[, -1]))
hcavg <- hclust(letter.dist, method = "average")
plot(hcavg, cex = 0.7)
```



```
avgSol <- (cutree(hcavg, k = 6))
letterPCA <- princomp(letters[, -1], cor = TRUE)

plot(letterPCA$scores[, 1:2], type = "n", xlab = "PC1", ylab = "PC2", main = "average hc and PCA") #bla
text(letterPCA$scores[, 1:2], col = y, labels = avgSol, cex = 0.7)
```



```
plot(letterPCA$scores[, 1:2], type = "n", xlab = "PC1", ylab = "PC2", main = "K means and PCA") #blank!
text(letterPCA$scores[, 1:2], col = y, labels = Ksol1$cluster, cex = 0.7)
```



b. (4 points) Do either of your solutions seem to recover specific letters or letter groupings? If so, what letters are grouped together?

SOLUTION: It seems as though mant of the letters are mixed into the different groups, however, it seems that some groups such as clusters 1 and 2 in the hierarcical clustering and 5 and 4 in the k means clustering were mostly made up of a single letter. It appears as though 1 in the hc and 5 in the kmeans wer mostly made up of letter T. However, cluster 2 in the hc and cluster 4 in the kmeans seems to be mostly made up of A's.

c. (8 points) Are your preferred natural groups really present? We can assess this by running a different algorithm to see if the groups you found are well-recovered. Use a random forest with parameter values chosen by you to optimize the resulting solution to assess how well your groups are recovered. Report your findings, including AER and estimated TER. How well are your natural groups recovered?

SOLUTION:

It seems as though the clusters found in the k means clustering are really present because according to the randomn forests 93.33% of the variance in the clusters are explained.

```
set.seed(240)
bf.rf <- randomForest( Ksol1$cluster ~ ., data = letters, mtry = 3, ntree = 100, importance = F, proxim
bf.rf
```

```
##
## Call:
##  randomForest(formula = Ksol1$cluster ~ ., data = letters, mtry = 3,      ntree = 100, importance = 
##                Type of random forest: regression
##                      Number of trees: 100
## No. of variables tried at each split: 3
##
##           Mean of squared residuals: 0.18302277
##                     % Var explained: 93.33
```