Midterm 1 Applied Data Mining

Keith Hickman

November 14, 2017

# Problem 1

## install.packages("data.table")  
library(data.table)  
library(ggplot2)  
mydata <- read.csv("C:/Users/khickman/Desktop/Personal/IUMSDS/AppliedDataMining/Midterm/mydata.csv", sep=",")  
  
summary(mydata)

## V1 V2 V3   
## Min. : 1.0 ? : 4 Min. :-6.7749   
## 1st Qu.: 500.8 -0.001405791: 1 1st Qu.:-2.2878   
## Median :1000.5 -0.002235545: 1 Median :-0.4438   
## Mean :1000.5 -0.003699072: 1 Mean :-0.9815   
## 3rd Qu.:1500.2 -0.006583953: 1 3rd Qu.: 0.4354   
## Max. :2000.0 -0.006972429: 1 Max. : 3.1754   
## (Other) :1991 NA's :8   
## V4 V5 X   
## ? : 6 Min. :-12.342 Min. :1.00   
## -0.00303014 : 1 1st Qu.: -9.420 1st Qu.:1.75   
## -0.012157336: 1 Median : -8.628 Median :2.00   
## -0.017954776: 1 Mean : -6.775 Mean :1.75   
## -0.027248905: 1 3rd Qu.: -4.728 3rd Qu.:2.00   
## -0.031789989: 1 Max. : 3.355 Max. :2.00   
## (Other) :1989

str(mydata)

## 'data.frame': 2000 obs. of 6 variables:  
## $ V1: int 1 2 3 4 5 6 7 8 9 10 ...  
## $ V2: Factor w/ 1997 levels "-0.001405791",..: 1987 1330 1766 1850 1817 1768 1462 1870 1583 1809 ...  
## $ V3: num -3.27 -3.94 -4.92 -2.79 -4.66 ...  
## $ V4: Factor w/ 1995 levels "-0.00303014",..: 745 746 942 889 662 742 809 1855 681 764 ...  
## $ V5: num -9.21 -9.92 -8.66 -10.35 -10.58 ...  
## $ X : int 1 1 1 1 1 1 1 1 1 1 ...

mydata[100:110,]

## V1 V2 V3 V4 V5 X  
## 100 100 2.106898626 -3.673282 ? -10.551039 1  
## 101 101 ? -2.746142 9.093913921 -5.315355 1  
## 102 102 3.326490963 -6.774908 9.702177633 -9.461351 1  
## 103 103 2.33460081 -3.738696 10.36410186 -11.209379 1  
## 104 104 4.167999798 NA 8.183001977 -9.487888 1  
## 105 105 2.836025413 -3.413888 8.467948059 -8.823351 1  
## 106 106 3.672512903 -3.648048 10.37116448 -8.959097 1  
## 107 107 4.265924166 -3.897882 8.79822484 -9.610169 1  
## 108 108 3.288826248 -2.518422 8.479622918 -8.470363 1  
## 109 109 2.860199674 -3.332852 11.25037736 -8.391735 1  
## 110 110 3.748196493 -2.918405 7.787542705 -8.632427 1

## 1.

*How many entries are in the data set?* There are 2000 observations of 6 variables.

## 2.

*How many unknown or missing data are in the data set?* I noticed 8 missing values here, as well as some values as ? which is the same here as NA. Additionally, I've got two of the variables that should be continuous listed as factors.

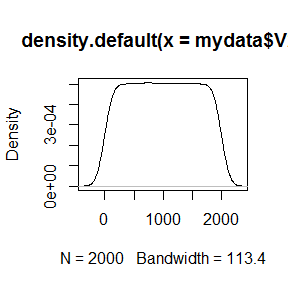
Starting with setting the datatypes factors:

mydata$V2 <- as.numeric(mydata$V2)  
mydata$V4 <- as.numeric(mydata$V4)  
str(mydata)

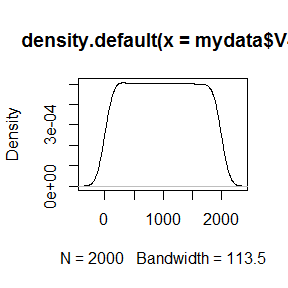
## 'data.frame': 2000 obs. of 6 variables:  
## $ V1: int 1 2 3 4 5 6 7 8 9 10 ...  
## $ V2: num 1987 1330 1766 1850 1817 ...  
## $ V3: num -3.27 -3.94 -4.92 -2.79 -4.66 ...  
## $ V4: num 745 746 942 889 662 ...  
## $ V5: num -9.21 -9.92 -8.66 -10.35 -10.58 ...  
## $ X : int 1 1 1 1 1 1 1 1 1 1 ...

Now that the data columns are of the correct type, we can deal with missing or incorrect values. To impute missing or incorrect values, I'll start with examining the distribution of each variable with missing values. Interestingly, this step also looks like it took care of reverting the "?" to actual values. Not sure why this happened, or whether the values are correct.

plot(density(mydata$V2))



## plot(density(mydata$V3))  
plot(density(mydata$V4))



I learned that only V3 is non-normal, and that V2 and V4 are almost uniformly distributed and symmetric. I'll use median for replacing missing values of V3.

## v3na <- is.na(mydata$V3)  
v3na <- mydata[rowSums(is.na(mydata)) > 0,]  
v3na

## V1 V2 V3 V4 V5 X  
## 50 50 1855 NA 908 -8.659472 1  
## 70 70 1843 NA 653 -10.469044 1  
## 104 104 1938 NA 1078 -9.487888 1  
## 201 201 1912 NA 1923 -9.402905 1  
## 301 301 1586 NA 1179 -9.106679 1  
## 401 401 1631 NA 1658 -9.761055 1  
## 800 800 619 NA 834 -9.125540 2  
## 900 900 1588 NA 1639 -9.802796 2

Great - 8 rows of the V3 variable have NA values. I'll also check again for question marks a bit later on. For now, let's impute the missing values. We'll use the mean because the shape of the variable indicates that thus will be a good representation of the values.

mydata[50, "V3"] <- median(mydata$V3, na.rm = TRUE)  
mydata[70, "V3"] <- median(mydata$V3, na.rm = TRUE)  
mydata[104, "V3"] <- median(mydata$V3, na.rm = TRUE)  
mydata[201, "V3"] <- median(mydata$V3, na.rm = TRUE)  
mydata[301, "V3"] <- median(mydata$V3, na.rm = TRUE)  
mydata[401, "V3"] <- median(mydata$V3, na.rm = TRUE)  
mydata[800, "V3"] <- median(mydata$V3, na.rm = TRUE)  
mydata[900, "V3"] <- median(mydata$V3, na.rm = TRUE)  
  
summary(mydata$V3)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## -6.7750 -2.2680 -0.4438 -0.9793 0.4299 3.1750

Great - looks like that did the trick. It imputed all values to the same number, however, so that's something that we may have to come back to later on.

Let's continue with problem 1.

## 3.

*Calculate mean and median of variable V2.*

mean(mydata$V2)

## [1] 998.6115

median(mydata$V2)

## [1] 997.5

The values are very close together, especially considering the scale. This looks good for a normally distributed variable.

## 4.

*Find variance, standard deviation and interquartile range of variable V4.*

var(mydata$V4)

## [1] 332438.5

sd(mydata$V4)

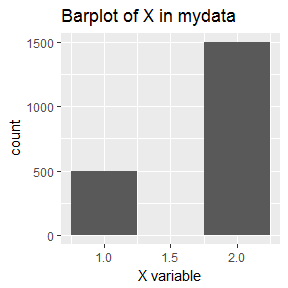
## [1] 576.5748

IQR(mydata$V4)

## [1] 999.5

Moving on to the barplot.

qplot(mydata$X, bins=3, xlab="X variable", main = "Barplot of X in mydata")

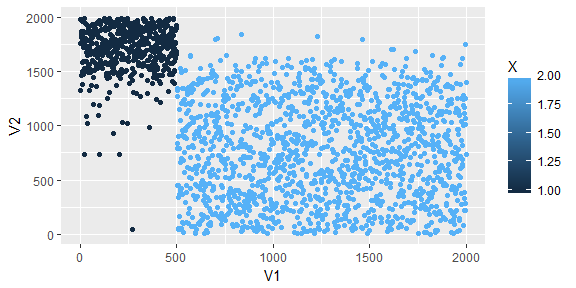


Looks like we have an uneven class distribution between class 1 and 2 in the X variable. This will likely be a factor in fitting models later on.

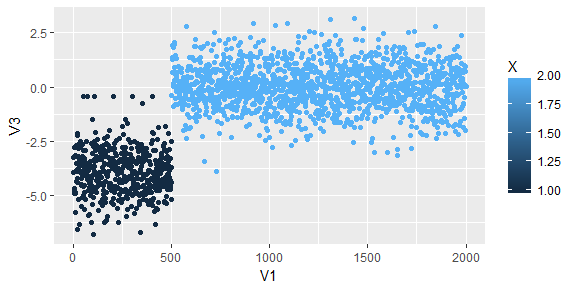
## 5.

*Create a bar plot that shows count of data points for classes " and " (variable 5). Is the data skewed?*

qplot(V1, V2, data=mydata, colour=X)



qplot(V1, V3, data=mydata, colour=X)

 #Problem 2.

## 1.

*How many principal components explain 90% of the variance?* PC1 and PC2 will explain 90% of the variance. In plot 1 (V1 and V2), I get the sense that V1 is going to be a good predictor of class. There is a clear pattern in both of these plots. From observation alone, some initial rules emerge for assigning target variables based on V1 that will handle a large majority of our cases. Where the observations have V1 <500, assign to class X=1. Where V1 > 500, assign to class 2.

In the second plot (V1 and V3) it's apparent that both V1 and V3 are well correlated with the class variable. Let's see whether this bears out in the PCA analysis. We'll use the prcomp function.

mydata.pca <- mydata[,1:4]  
summary(mydata.pca)

## V1 V2 V3 V4   
## Min. : 1.0 Min. : 1.0 Min. :-6.7749 Min. : 1.0   
## 1st Qu.: 500.8 1st Qu.: 500.8 1st Qu.:-2.2679 1st Qu.: 495.8   
## Median :1000.5 Median : 997.5 Median :-0.4438 Median : 995.5   
## Mean :1000.5 Mean : 998.6 Mean :-0.9793 Mean : 996.1   
## 3rd Qu.:1500.2 3rd Qu.:1497.2 3rd Qu.: 0.4299 3rd Qu.:1495.2   
## Max. :2000.0 Max. :1997.0 Max. : 3.1754 Max. :1995.0

princa <- princomp(mydata.pca)  
prca <- prcomp(mydata.pca)  
summary(princa)

## Importance of components:  
## Comp.1 Comp.2 Comp.3 Comp.4  
## Standard deviation 780.6856279 531.8591332 324.3447227 1.337746e+00  
## Proportion of Variance 0.6109697 0.2835702 0.1054583 1.793968e-06  
## Cumulative Proportion 0.6109697 0.8945399 0.9999982 1.000000e+00

summary(prca)

## Importance of components:  
## PC1 PC2 PC3 PC4  
## Standard deviation 780.881 531.9921 324.4258 1.338  
## Proportion of Variance 0.611 0.2836 0.1055 0.000  
## Cumulative Proportion 0.611 0.8945 1.0000 1.000

print(princa)

## Call:  
## princomp(x = mydata.pca)  
##   
## Standard deviations:  
## Comp.1 Comp.2 Comp.3 Comp.4   
## 780.685628 531.859133 324.344723 1.337746   
##   
## 4 variables and 2000 observations.

print(prca)

## Standard deviations:  
## [1] 780.880872 531.992148 324.425839 1.338081  
##   
## Rotation:  
## PC1 PC2 PC3 PC4  
## V1 0.67259513 -0.007526584 -0.739970162 0.0018174482  
## V2 -0.51586849 -0.721692291 -0.461560664 -0.0012991176  
## V3 0.00160239 0.001302478 -0.001012861 -0.9999973550  
## V4 -0.53055907 0.692171865 -0.489290187 0.0005469607

## loadings(princa)  
## loadings(prca)

## 2. Loadings in PCA.

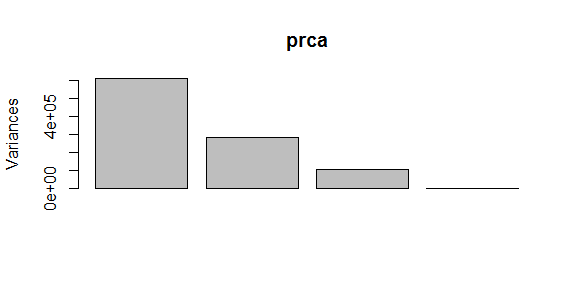
*What are loadings in PCA? Observe loadings and express the principal components using the original variables.* Loadings are the breakdown of how the pricinpal component variables were arrived at (e.g. what linear function was used), and what percentage of the variance each one explains. It appears that the first two components explain a 90% of the variance. PC1 was created by the function V1 \* .673 + V2 \* -.516 + V3 \* .001 + V4 \* -.530. Interestingly, the 4th PC in the PCA did not explain any of the variance. I initially didn't understand the difference betweeen princomp and prcomp but it appears that the use of eigen is the main difference, as well as the output. I can't call the loadings function on the variable transformed with prcomp, as it returns null.

## 3. Scree Plot.

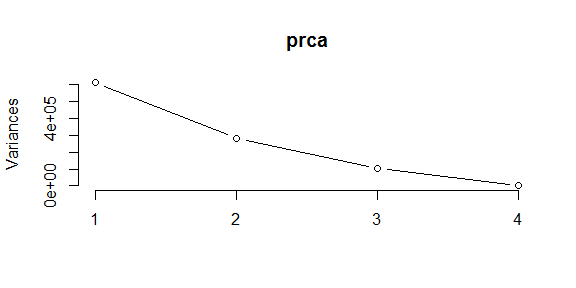
*Make a scree plot. Discuss the plot, i.e., what is a scree plot? What is the optimal number of dimensions based on the plot?*

Let's look at the scree and line plots:

screeplot(prca)



plot(prca,type="l")



If we were concerned with computer performance, we would likely select only the first two variables. Since it doesn't cost of anything, we can select the first three, as the 4th doesn't offer any added benefit.

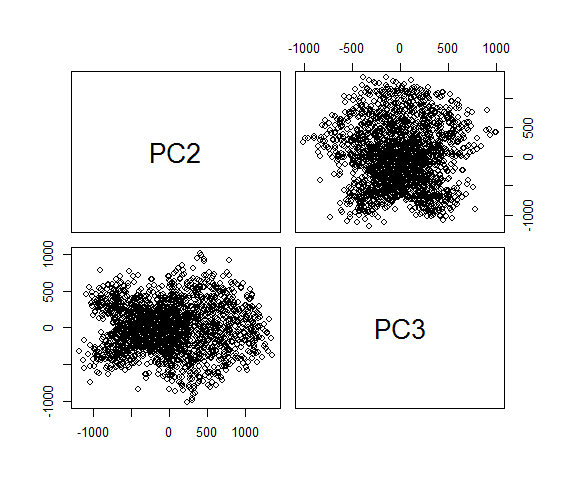
## 4. Scatter plot of PCs

*Make a scatter plot of PC2 and PC3. Do you observe any relationship? i.e., Calculate the correlation between PC2 and PC3? What does it show?*

The transformed variables are normally distributed, but since the underlying variables are closer to uniform distributions, I will try other correlation methods Spearman and Kendall.

Examining a scatter plot and correlation coefficient of PC2 and PC3:

## hist(prca$x)  
pairs(prca$x[,2:3])



cor(prca$x, prca$x)

## PC1 PC2 PC3 PC4  
## PC1 1.000000e+00 -7.698859e-16 -4.415020e-15 4.553976e-15  
## PC2 -7.698859e-16 1.000000e+00 -3.415184e-16 6.208564e-14  
## PC3 -4.415020e-15 -3.415184e-16 1.000000e+00 2.049505e-15  
## PC4 4.553976e-15 6.208564e-14 2.049505e-15 1.000000e+00

cor(prca$x, prca$x, method= "kendall")

## PC1 PC2 PC3 PC4  
## PC1 1.000000000 0.008620310 0.001432716 0.002543272  
## PC2 0.008620310 1.000000000 0.001158579 -0.014747374  
## PC3 0.001432716 0.001158579 1.000000000 0.002243122  
## PC4 0.002543272 -0.014747374 0.002243122 1.000000000

cor(prca$x, prca$x, method= "spearman")

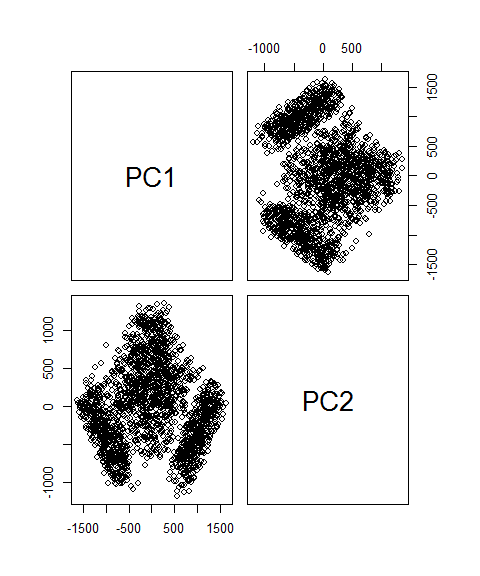
## PC1 PC2 PC3 PC4  
## PC1 1.000000000 0.009710051 -0.012794202 -0.009230396  
## PC2 0.009710051 1.000000000 0.001295745 -0.024909630  
## PC3 -0.012794202 0.001295745 1.000000000 0.004654344  
## PC4 -0.009230396 -0.024909630 0.004654344 1.000000000

??cor

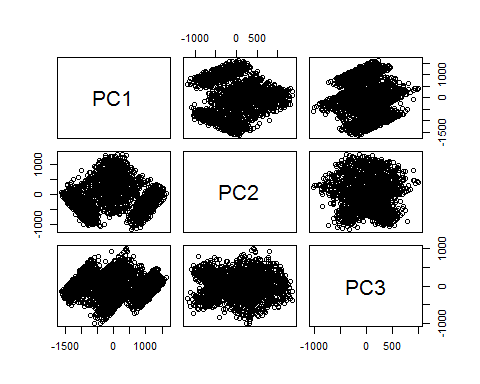
## starting httpd help server ... done

Interesting. There aren't any strong linear correlations in the data, but there are definite clusters present. The correlation matrix shows very small Pearson correlation coefficients. As for the scatter plots, there are two prongs on the left side of the scale, a cluster in the middle, and less dense values of PC2 above 250. I was also curious about the correlation between PC1 and PC2, which is plotted here:

pairs(prca$x[,1:2])

 I found this correlation to be very compelling, and likely better suited to a clustering algorithm we might perform, because the clusters would probably be much clearer, and individual instance values would be easier to classify. However, I don't know whether k-means will perform exceptionally well, as the apparent clusters tend to be oblong and irregularly shaped vs. circular, where k-means performs best. Additionally, we can capture more variance of the original dataset by using PC1 and PC2 as well. What about PC1 and PC3?

pairs(prca$x[,1:3])



Looks like all three pairs of variables have interesting correlations! PC1 and PC3 align into three or possibly four neat clusters.

# Problem 3.

## 1.

*Randomly sample without replacement 300 data points from kmeans.mydata. (call the sampled data mysample). Cluster mysample with K-means. Include the R code and answer the questions below:*

Code:

#Creating the vector  
kmeans.mydata <- mydata[,c(1,2)]  
## V1 appears to be an id variable - it just iterates as n+1 for every observation. Should this be included in the k-means?   
  
kmeans.mydata <- mydata[,c(1,2)]  
kmeans.mydata

## V1 V2  
## 1 1 1987  
## 2 2 1330  
## 3 3 1766  
## 4 4 1850  
## 5 5 1817  
## 6 6 1768  
## 7 7 1462  
## 8 8 1870  
## 9 9 1583  
## 10 10 1809  
## 11 11 1762  
## 12 12 1380  
## 13 13 1506  
## 14 14 1970  
## 15 15 1599  
## 16 16 1851  
## 17 17 1729  
## 18 18 1474  
## 19 19 1871  
## 20 20 1997  
## 21 21 1812  
## 22 22 740  
## 23 23 1530  
## 24 24 1792  
## 25 25 1744  
## 26 26 1813  
## 27 27 1951  
## 28 28 1943  
## 29 29 1864  
## 30 30 1677  
## 31 31 1597  
## 32 32 1087  
## 33 33 1722  
## 34 34 1732  
## 35 35 1933  
## 36 36 1027  
## 37 37 1652  
## 38 38 1789  
## 39 39 1505  
## 40 40 1493  
## 41 41 1723  
## 42 42 1897  
## 43 43 1571  
## 44 44 1547  
## 45 45 1724  
## 46 46 1733  
## 47 47 1568  
## 48 48 1327  
## 49 49 1705  
## 50 50 1855  
## 51 51 1659  
## 52 52 1785  
## 53 53 1497  
## 54 54 1702  
## 55 55 1526  
## 56 56 1839  
## 57 57 1869  
## 58 58 1373  
## 59 59 1835  
## 60 60 1811  
## 61 61 1940  
## 62 62 1761  
## 63 63 1605  
## 64 64 1820  
## 65 65 1930  
## 66 66 1449  
## 67 67 1201  
## 68 68 1690  
## 69 69 1917  
## 70 70 1843  
## 71 71 1834  
## 72 72 1513  
## 73 73 1662  
## 74 74 1868  
## 75 75 1590  
## 76 76 1934  
## 77 77 1937  
## 78 78 1673  
## 79 79 1885  
## 80 80 1701  
## 81 81 1900  
## 82 82 1763  
## 83 83 1546  
## 84 84 1707  
## 85 85 1368  
## 86 86 1698  
## 87 87 1844  
## 88 88 1535  
## 89 89 1630  
## 90 90 1952  
## 91 91 1664  
## 92 92 1565  
## 93 93 1814  
## 94 94 1726  
## 95 95 1955  
## 96 96 1095  
## 97 97 1694  
## 98 98 1188  
## 99 99 1975  
## 100 100 1541  
## 101 101 740  
## 102 102 1815  
## 103 103 1602  
## 104 104 1938  
## 105 105 1713  
## 106 106 1882  
## 107 107 1948  
## 108 108 1804  
## 109 109 1721  
## 110 110 1893  
## 111 111 1564  
## 112 112 1671  
## 113 113 1668  
## 114 114 1764  
## 115 115 1529  
## 116 116 1647  
## 117 117 1651  
## 118 118 1681  
## 119 119 1740  
## 120 120 1972  
## 121 121 1770  
## 122 122 1667  
## 123 123 1574  
## 124 124 1697  
## 125 125 1985  
## 126 126 1731  
## 127 127 1993  
## 128 128 1743  
## 129 129 1787  
## 130 130 1755  
## 131 131 1582  
## 132 132 1909  
## 133 133 1878  
## 134 134 1458  
## 135 135 1899  
## 136 136 1683  
## 137 137 1484  
## 138 138 1956  
## 139 139 1703  
## 140 140 1894  
## 141 141 1566  
## 142 142 1254  
## 143 143 1437  
## 144 144 1672  
## 145 145 1769  
## 146 146 1655  
## 147 147 1716  
## 148 148 1895  
## 149 149 1489  
## 150 150 1994  
## 151 151 1949  
## 152 152 1451  
## 153 153 1310  
## 154 154 1591  
## 155 155 1739  
## 156 156 1914  
## 157 157 1841  
## 158 158 1644  
## 159 159 1772  
## 160 160 1969  
## 161 161 1798  
## 162 162 1492  
## 163 163 1559  
## 164 164 1790  
## 165 165 1616  
## 166 166 1824  
## 167 167 1944  
## 168 168 1537  
## 169 169 1680  
## 170 170 930  
## 171 171 1485  
## 172 172 1780  
## 173 173 1524  
## 174 174 1539  
## 175 175 1920  
## 176 176 1747  
## 177 177 1615  
## 178 178 1699  
## 179 179 1791  
## 180 180 1575  
## 181 181 1494  
## 182 182 1799  
## 183 183 1788  
## 184 184 1584  
## 185 185 1637  
## 186 186 1376  
## 187 187 1954  
## 188 188 1587  
## 189 189 1968  
## 190 190 1793  
## 191 191 1715  
## 192 192 1856  
## 193 193 1982  
## 194 194 1781  
## 195 195 1669  
## 196 196 1471  
## 197 197 1734  
## 198 198 1881  
## 199 199 1988  
## 200 200 1964  
## 201 201 1912  
## 202 202 740  
## 203 203 1866  
## 204 204 1876  
## 205 205 1941  
## 206 206 1666  
## 207 207 1797  
## 208 208 1440  
## 209 209 1500  
## 210 210 1854  
## 211 211 1760  
## 212 212 1908  
## 213 213 1991  
## 214 214 1522  
## 215 215 1487  
## 216 216 1617  
## 217 217 1657  
## 218 218 1636  
## 219 219 1737  
## 220 220 1853  
## 221 221 1036  
## 222 222 1983  
## 223 223 1891  
## 224 224 1911  
## 225 225 1816  
## 226 226 1974  
## 227 227 1932  
## 228 228 1939  
## 229 229 1883  
## 230 230 1802  
## 231 231 1613  
## 232 232 1502  
## 233 233 1959  
## 234 234 1777  
## 235 235 1965  
## 236 236 1960  
## 237 237 1612  
## 238 238 1510  
## 239 239 1826  
## 240 240 1718  
## 241 241 1625  
## 242 242 1704  
## 243 243 1984  
## 244 244 1852  
## 245 245 1992  
## 246 246 1978  
## 247 247 1023  
## 248 248 1823  
## 249 249 1633  
## 250 250 1942  
## 251 251 1867  
## 252 252 1598  
## 253 253 1682  
## 254 254 1921  
## 255 255 1782  
## 256 256 1849  
## 257 257 1392  
## 258 258 1896  
## 259 259 1862  
## 260 260 1455  
## 261 261 1831  
## 262 262 1860  
## 263 263 1910  
## 264 264 1469  
## 265 265 1947  
## 266 266 1736  
## 267 267 1689  
## 268 268 1417  
## 269 269 1794  
## 270 270 1480  
## 271 271 1527  
## 272 272 1693  
## 273 273 43  
## 274 274 1847  
## 275 275 1311  
## 276 276 1746  
## 277 277 1827  
## 278 278 1604  
## 279 279 1927  
## 280 280 1957  
## 281 281 1634  
## 282 282 1490  
## 283 283 1661  
## 284 284 1622  
## 285 285 1886  
## 286 286 1656  
## 287 287 1846  
## 288 288 1749  
## 289 289 1298  
## 290 290 1696  
## 291 291 1840  
## 292 292 1695  
## 293 293 1822  
## 294 294 1858  
## 295 295 1946  
## 296 296 1977  
## 297 297 1916  
## 298 298 1646  
## 299 299 1931  
## 300 300 1735  
## 301 301 1586  
## 302 302 1727  
## 303 303 1892  
## 304 304 1561  
## 305 305 1692  
## 306 306 1879  
## 307 307 1775  
## 308 308 1650  
## 309 309 1848  
## 310 310 1756  
## 311 311 1268  
## 312 312 1875  
## 313 313 1684  
## 314 314 1784  
## 315 315 1907  
## 316 316 1670  
## 317 317 1863  
## 318 318 1632  
## 319 319 1464  
## 320 320 1902  
## 321 321 1643  
## 322 322 1548  
## 323 323 1821  
## 324 324 1771  
## 325 325 1578  
## 326 326 1980  
## 327 327 1810  
## 328 328 1989  
## 329 329 1874  
## 330 330 1750  
## 331 331 1663  
## 332 332 1563  
## 333 333 1753  
## 334 334 1962  
## 335 335 1658  
## 336 336 1973  
## 337 337 1976  
## 338 338 1880  
## 339 339 1966  
## 340 340 1687  
## 341 341 1889  
## 342 342 1778  
## 343 343 1915  
## 344 344 1806  
## 345 345 1700  
## 346 346 1416  
## 347 347 1302  
## 348 348 1606  
## 349 349 1926  
## 350 350 1554  
## 351 351 1758  
## 352 352 1877  
## 353 353 1950  
## 354 354 1888  
## 355 355 1609  
## 356 356 1828  
## 357 357 987  
## 358 358 1519  
## 359 359 1837  
## 360 360 1945  
## 361 361 1865  
## 362 362 1935  
## 363 363 1645  
## 364 364 1861  
## 365 365 1709  
## 366 366 1923  
## 367 367 1712  
## 368 368 1901  
## 369 369 1990  
## 370 370 1558  
## 371 371 1728  
## 372 372 1394  
## 373 373 1553  
## 374 374 1686  
## 375 375 1961  
## 376 376 1711  
## 377 377 1600  
## 378 378 1808  
## 379 379 1515  
## 380 380 1452  
## 381 381 1953  
## 382 382 1478  
## 383 383 1675  
## 384 384 1498  
## 385 385 1725  
## 386 386 1905  
## 387 387 1929  
## 388 388 1742  
## 389 389 1629  
## 390 390 1570  
## 391 391 1603  
## 392 392 1857  
## 393 393 1247  
## 394 394 1685  
## 395 395 1748  
## 396 396 1443  
## 397 397 1751  
## 398 398 1825  
## 399 399 1520  
## 400 400 1786  
## 401 401 1631  
## 402 402 1544  
## 403 403 1465  
## 404 404 1918  
## 405 405 1714  
## 406 406 1638  
## 407 407 1967  
## 408 408 1678  
## 409 409 1979  
## 410 410 1679  
## 411 411 1408  
## 412 412 1936  
## 413 413 1884  
## 414 414 1665  
## 415 415 1218  
## 416 416 1767  
## 417 417 1922  
## 418 418 1396  
## 419 419 1642  
## 420 420 1818  
## 421 421 1466  
## 422 422 1608  
## 423 423 1717  
## 424 424 1610  
## 425 425 1783  
## 426 426 1924  
## 427 427 1838  
## 428 428 1468  
## 429 429 1958  
## 430 430 1730  
## 431 431 1639  
## 432 432 1873  
## 433 433 1516  
## 434 434 1819  
## 435 435 1963  
## 436 436 1836  
## 437 437 1626  
## 438 438 1557  
## 439 439 1830  
## 440 440 1919  
## 441 441 1773  
## 442 442 1913  
## 443 443 1801  
## 444 444 1776  
## 445 445 1754  
## 446 446 1986  
## 447 447 1796  
## 448 448 1538  
## 449 449 1540  
## 450 450 1720  
## 451 451 1928  
## 452 452 1585  
## 453 453 1805  
## 454 454 1618  
## 455 455 1323  
## 456 456 1779  
## 457 457 1525  
## 458 458 1832  
## 459 459 1971  
## 460 460 1996  
## 461 461 1741  
## 462 462 1641  
## 463 463 1745  
## 464 464 1473  
## 465 465 1757  
## 466 466 1859  
## 467 467 1842  
## 468 468 1738  
## 469 469 1581  
## 470 470 1872  
## 471 471 1674  
## 472 472 1890  
## 473 473 1795  
## 474 474 1903  
## 475 475 1488  
## 476 476 1476  
## 477 477 1649  
## 478 478 1573  
## 479 479 1409  
## 480 480 1759  
## 481 481 1981  
## 482 482 1708  
## 483 483 1833  
## 484 484 1995  
## 485 485 1676  
## 486 486 1596  
## 487 487 1504  
## 488 488 1514  
## 489 489 1719  
## 490 490 1393  
## 491 491 1620  
## 492 492 1765  
## 493 493 1887  
## 494 494 1774  
## 495 495 1898  
## 496 496 1925  
## 497 497 1906  
## 498 498 1660  
## 499 499 1904  
## 500 500 1691  
## 501 501 1345  
## 502 502 445  
## 503 503 1332  
## 504 504 1273  
## 505 505 842  
## 506 506 788  
## 507 507 899  
## 508 508 453  
## 509 509 343  
## 510 510 49  
## 511 511 39  
## 512 512 793  
## 513 513 366  
## 514 514 164  
## 515 515 42  
## 516 516 1155  
## 517 517 1055  
## 518 518 660  
## 519 519 420  
## 520 520 401  
## 521 521 1534  
## 522 522 1204  
## 523 523 426  
## 524 524 1495  
## 525 525 56  
## 526 526 702  
## 527 527 363  
## 528 528 955  
## 529 529 105  
## 530 530 908  
## 531 531 1348  
## 532 532 1211  
## 533 533 1344  
## 534 534 677  
## 535 535 19  
## 536 536 101  
## 537 537 1235  
## 538 538 566  
## 539 539 755  
## 540 540 901  
## 541 541 1242  
## 542 542 891  
## 543 543 1512  
## 544 544 1199  
## 545 545 294  
## 546 546 961  
## 547 547 973  
## 548 548 880  
## 549 549 452  
## 550 550 387  
## 551 551 1119  
## 552 552 110  
## 553 553 1320  
## 554 554 993  
## 555 555 1291  
## 556 556 614  
## 557 557 1295  
## 558 558 1205  
## 559 559 797  
## 560 560 1354  
## 561 561 195  
## 562 562 1147  
## 563 563 95  
## 564 564 239  
## 565 565 525  
## 566 566 1015  
## 567 567 1333  
## 568 568 605  
## 569 569 422  
## 570 570 1654  
## 571 571 1195  
## 572 572 1640  
## 573 573 705  
## 574 574 1079  
## 575 575 518  
## 576 576 73  
## 577 577 1371  
## 578 578 1307  
## 579 579 597  
## 580 580 158  
## 581 581 1035  
## 582 582 818  
## 583 583 1223  
## 584 584 1335  
## 585 585 360  
## 586 586 1033  
## 587 587 1030  
## 588 588 801  
## 589 589 1388  
## 590 590 918  
## 591 591 71  
## 592 592 607  
## 593 593 882  
## 594 594 1071  
## 595 595 15  
## 596 596 5  
## 597 597 324  
## 598 598 627  
## 599 599 456  
## 600 600 556  
## 601 601 41  
## 602 602 598  
## 603 603 341  
## 604 604 945  
## 605 605 1383  
## 606 606 348  
## 607 607 1077  
## 608 608 438  
## 609 609 717  
## 610 610 639  
## 611 611 701  
## 612 612 1430  
## 613 613 97  
## 614 614 936  
## 615 615 906  
## 616 616 1436  
## 617 617 984  
## 618 618 964  
## 619 619 1088  
## 620 620 85  
## 621 621 417  
## 622 622 1086  
## 623 623 1532  
## 624 624 1198  
## 625 625 121  
## 626 626 455  
## 627 627 1399  
## 628 628 855  
## 629 629 186  
## 630 630 477  
## 631 631 581  
## 632 632 134  
## 633 633 494  
## 634 634 474  
## 635 635 776  
## 636 636 315  
## 637 637 1292  
## 638 638 168  
## 639 639 1425  
## 640 640 405  
## 641 641 989  
## 642 642 613  
## 643 643 467  
## 644 644 3  
## 645 645 888  
## 646 646 354  
## 647 647 1309  
## 648 648 839  
## 649 649 440  
## 650 650 284  
## 651 651 1054  
## 652 652 1099  
## 653 653 269  
## 654 654 35  
## 655 655 1017  
## 656 656 654  
## 657 657 150  
## 658 658 339  
## 659 659 11  
## 660 660 1329  
## 661 661 1435  
## 662 662 649  
## 663 663 91  
## 664 664 1189  
## 665 665 699  
## 666 666 939  
## 667 667 236  
## 668 668 84  
## 669 669 571  
## 670 670 314  
## 671 671 468  
## 672 672 104  
## 673 673 207  
## 674 674 77  
## 675 675 1562  
## 676 676 530  
## 677 677 781  
## 678 678 1483  
## 679 679 1101  
## 680 680 1116  
## 681 681 858  
## 682 682 1444  
## 683 683 184  
## 684 684 404  
## 685 685 729  
## 686 686 1509  
## 687 687 594  
## 688 688 416  
## 689 689 809  
## 690 690 1098  
## 691 691 443  
## 692 692 1313  
## 693 693 1040  
## 694 694 1475  
## 695 695 896  
## 696 696 551  
## 697 697 181  
## 698 698 995  
## 699 699 192  
## 700 700 1106  
## 701 701 1803  
## 702 702 1193  
## 703 703 1413  
## 704 704 338  
## 705 705 1037  
## 706 706 926  
## 707 707 999  
## 708 708 1459  
## 709 709 1807  
## 710 710 527  
## 711 711 1080  
## 712 712 1128  
## 713 713 243  
## 714 714 1081  
## 715 715 592  
## 716 716 962  
## 717 717 167  
## 718 718 731  
## 719 719 725  
## 720 720 953  
## 721 721 723  
## 722 722 986  
## 723 723 874  
## 724 724 140  
## 725 725 722  
## 726 726 337  
## 727 727 948  
## 728 728 1607  
## 729 729 446  
## 730 730 90  
## 731 731 187  
## 732 732 913  
## 733 733 138  
## 734 734 683  
## 735 735 1183  
## 736 736 166  
## 737 737 434  
## 738 738 1161  
## 739 739 311  
## 740 740 350  
## 741 741 1285  
## 742 742 1237  
## 743 743 661  
## 744 744 769  
## 745 745 165  
## 746 746 1229  
## 747 747 659  
## 748 748 693  
## 749 749 1325  
## 750 750 593  
## 751 751 128  
## 752 752 524  
## 753 753 641  
## 754 754 1140  
## 755 755 1277  
## 756 756 1246  
## 757 757 894  
## 758 758 1653  
## 759 759 1083  
## 760 760 823  
## 761 761 180  
## 762 762 700  
## 763 763 1026  
## 764 764 1442  
## 765 765 1131  
## 766 766 1251  
## 767 767 193  
## 768 768 870  
## 769 769 252  
## 770 770 370  
## 771 771 325  
## 772 772 866  
## 773 773 1412  
## 774 774 515  
## 775 775 398  
## 776 776 220  
## 777 777 313  
## 778 778 310  
## 779 779 419  
## 780 780 1342  
## 781 781 837  
## 782 782 388  
## 783 783 291  
## 784 784 656  
## 785 785 1018  
## 786 786 942  
## 787 787 93  
## 788 788 535  
## 789 789 611  
## 790 790 1401  
## 791 791 1178  
## 792 792 1461  
## 793 793 573  
## 794 794 789  
## 795 795 74  
## 796 796 1283  
## 797 797 356  
## 798 798 347  
## 799 799 266  
## 800 800 619  
## 801 801 1457  
## 802 802 250  
## 803 803 403  
## 804 804 626  
## 805 805 1094  
## 806 806 1555  
## 807 807 1340  
## 808 808 203  
## 809 809 1517  
## 810 810 202  
## 811 811 89  
## 812 812 208  
## 813 813 81  
## 814 814 1358  
## 815 815 1149  
## 816 816 1363  
## 817 817 328  
## 818 818 800  
## 819 819 1460  
## 820 820 1194  
## 821 821 1148  
## 822 822 406  
## 823 823 277  
## 824 824 197  
## 825 825 188  
## 826 826 813  
## 827 827 542  
## 828 828 1378  
## 829 829 1418  
## 830 830 856  
## 831 831 376  
## 832 832 688  
## 833 833 21  
## 834 834 1375  
## 835 835 1503  
## 836 836 1845  
## 837 837 879  
## 838 838 1208  
## 839 839 60  
## 840 840 808  
## 841 841 115  
## 842 842 30  
## 843 843 655  
## 844 844 213  
## 845 845 259  
## 846 846 828  
## 847 847 716  
## 848 848 950  
## 849 849 708  
## 850 850 300  
## 851 851 69  
## 852 852 719  
## 853 853 368  
## 854 854 1000  
## 855 855 273  
## 856 856 523  
## 857 857 832  
## 858 858 270  
## 859 859 814  
## 860 860 1400  
## 861 861 805  
## 862 862 176  
## 863 863 242  
## 864 864 704  
## 865 865 342  
## 866 866 63  
## 867 867 1143  
## 868 868 979  
## 869 869 355  
## 870 870 301  
## 871 871 251  
## 872 872 1165  
## 873 873 1551  
## 874 874 1454  
## 875 875 770  
## 876 876 256  
## 877 877 217  
## 878 878 539  
## 879 879 391  
## 880 880 802  
## 881 881 1381  
## 882 882 990  
## 883 883 709  
## 884 884 111  
## 885 885 333  
## 886 886 502  
## 887 887 385  
## 888 888 365  
## 889 889 1244  
## 890 890 517  
## 891 891 451  
## 892 892 1410  
## 893 893 62  
## 894 894 1232  
## 895 895 629  
## 896 896 759  
## 897 897 588  
## 898 898 1282  
## 899 899 316  
## 900 900 1588  
## 901 901 364  
## 902 902 329  
## 903 903 296  
## 904 904 919  
## 905 905 36  
## 906 906 1415  
## 907 907 1096  
## 908 908 1191  
## 909 909 1628  
## 910 910 1264  
## 911 911 120  
## 912 912 276  
## 913 913 909  
## 914 914 965  
## 915 915 22  
## 916 916 96  
## 917 917 1250  
## 918 918 450  
## 919 919 1219  
## 920 920 72  
## 921 921 475  
## 922 922 648  
## 923 923 40  
## 924 924 1507  
## 925 925 1230  
## 926 926 635  
## 927 927 637  
## 928 928 344  
## 929 929 1550  
## 930 930 1111  
## 931 931 742  
## 932 932 487  
## 933 933 1321  
## 934 934 1156  
## 935 935 890  
## 936 936 173  
## 937 937 58  
## 938 938 1433  
## 939 939 630  
## 940 940 1176  
## 941 941 696  
## 942 942 935  
## 943 943 367  
## 944 944 431  
## 945 945 1580  
## 946 946 441  
## 947 947 174  
## 948 948 707  
## 949 949 23  
## 950 950 332  
## 951 951 157  
## 952 952 493  
## 953 953 302  
## 954 954 227  
## 955 955 732  
## 956 956 636  
## 957 957 674  
## 958 958 1374  
## 959 959 1624  
## 960 960 409  
## 961 961 1196  
## 962 962 400  
## 963 963 695  
## 964 964 1286  
## 965 965 146  
## 966 966 1576  
## 967 967 237  
## 968 968 997  
## 969 969 10  
## 970 970 959  
## 971 971 974  
## 972 972 712  
## 973 973 928  
## 974 974 70  
## 975 975 510  
## 976 976 1215  
## 977 977 1241  
## 978 978 604  
## 979 979 565  
## 980 980 785  
## 981 981 949  
## 982 982 784  
## 983 983 318  
## 984 984 615  
## 985 985 214  
## 986 986 933  
## 987 987 143  
## 988 988 1404  
## 989 989 1024  
## 990 990 827  
## 991 991 288  
## 992 992 260  
## 993 993 873  
## 994 994 131  
## 995 995 54  
## 996 996 153  
## 997 997 1253  
## 998 998 1011  
## 999 999 238  
## 1000 1000 835  
## 1001 1001 845  
## 1002 1002 563  
## 1003 1003 1352  
## 1004 1004 1222  
## 1005 1005 1481  
## 1006 1006 663  
## 1007 1007 106  
## 1008 1008 1227  
## 1009 1009 1221  
## 1010 1010 579  
## 1011 1011 253  
## 1012 1012 1594  
## 1013 1013 102  
## 1014 1014 651  
## 1015 1015 603  
## 1016 1016 1041  
## 1017 1017 424  
## 1018 1018 694  
## 1019 1019 286  
## 1020 1020 509  
## 1021 1021 200  
## 1022 1022 912  
## 1023 1023 885  
## 1024 1024 1406  
## 1025 1025 116  
## 1026 1026 730  
## 1027 1027 664  
## 1028 1028 1346  
## 1029 1029 497  
## 1030 1030 522  
## 1031 1031 429  
## 1032 1032 1181  
## 1033 1033 1339  
## 1034 1034 1365  
## 1035 1035 529  
## 1036 1036 1062  
## 1037 1037 1206  
## 1038 1038 617  
## 1039 1039 1357  
## 1040 1040 64  
## 1041 1041 1252  
## 1042 1042 883  
## 1043 1043 897  
## 1044 1044 686  
## 1045 1045 736  
## 1046 1046 1238  
## 1047 1047 245  
## 1048 1048 210  
## 1049 1049 399  
## 1050 1050 1543  
## 1051 1051 834  
## 1052 1052 601  
## 1053 1053 844  
## 1054 1054 850  
## 1055 1055 893  
## 1056 1056 1006  
## 1057 1057 645  
## 1058 1058 1284  
## 1059 1059 796  
## 1060 1060 586  
## 1061 1061 358  
## 1062 1062 851  
## 1063 1063 1118  
## 1064 1064 449  
## 1065 1065 248  
## 1066 1066 931  
## 1067 1067 1104  
## 1068 1068 240  
## 1069 1069 290  
## 1070 1070 1477  
## 1071 1071 1556  
## 1072 1072 706  
## 1073 1073 830  
## 1074 1074 230  
## 1075 1075 371  
## 1076 1076 516  
## 1077 1077 691  
## 1078 1078 1623  
## 1079 1079 1288  
## 1080 1080 1174  
## 1081 1081 1262  
## 1082 1082 713  
## 1083 1083 938  
## 1084 1084 460  
## 1085 1085 557  
## 1086 1086 773  
## 1087 1087 1175  
## 1088 1088 1110  
## 1089 1089 1063  
## 1090 1090 219  
## 1091 1091 622  
## 1092 1092 745  
## 1093 1093 1179  
## 1094 1094 872  
## 1095 1095 1536  
## 1096 1096 351  
## 1097 1097 754  
## 1098 1098 1173  
## 1099 1099 1141  
## 1100 1100 1317  
## 1101 1101 482  
## 1102 1102 209  
## 1103 1103 570  
## 1104 1104 631  
## 1105 1105 852  
## 1106 1106 392  
## 1107 1107 390  
## 1108 1108 541  
## 1109 1109 924  
## 1110 1110 1390  
## 1111 1111 67  
## 1112 1112 13  
## 1113 1113 393  
## 1114 1114 752  
## 1115 1115 287  
## 1116 1116 1299  
## 1117 1117 689  
## 1118 1118 840  
## 1119 1119 1138  
## 1120 1120 774  
## 1121 1121 867  
## 1122 1122 922  
## 1123 1123 151  
## 1124 1124 7  
## 1125 1125 727  
## 1126 1126 384  
## 1127 1127 905  
## 1128 1128 662  
## 1129 1129 458  
## 1130 1130 911  
## 1131 1131 862  
## 1132 1132 1446  
## 1133 1133 1377  
## 1134 1134 98  
## 1135 1135 212  
## 1136 1136 1482  
## 1137 1137 1407  
## 1138 1138 1085  
## 1139 1139 481  
## 1140 1140 679  
## 1141 1141 678  
## 1142 1142 1316  
## 1143 1143 211  
## 1144 1144 436  
## 1145 1145 8  
## 1146 1146 1296  
## 1147 1147 231  
## 1148 1148 1589  
## 1149 1149 293  
## 1150 1150 681  
## 1151 1151 379  
## 1152 1152 826  
## 1153 1153 599  
## 1154 1154 407  
## 1155 1155 892  
## 1156 1156 1084  
## 1157 1157 51  
## 1158 1158 1303  
## 1159 1159 744  
## 1160 1160 1214  
## 1161 1161 846  
## 1162 1162 317  
## 1163 1163 25  
## 1164 1164 1002  
## 1165 1165 735  
## 1166 1166 998  
## 1167 1167 1542  
## 1168 1168 1523  
## 1169 1169 1032  
## 1170 1170 268  
## 1171 1171 533  
## 1172 1172 780  
## 1173 1173 547  
## 1174 1174 1601  
## 1175 1175 28  
## 1176 1176 1337  
## 1177 1177 806  
## 1178 1178 1359  
## 1179 1179 1042  
## 1180 1180 1224  
## 1181 1181 1531  
## 1182 1182 838  
## 1183 1183 1203  
## 1184 1184 132  
## 1185 1185 1012  
## 1186 1186 1240  
## 1187 1187 1234  
## 1188 1188 353  
## 1189 1189 1334  
## 1190 1190 31  
## 1191 1191 650  
## 1192 1192 94  
## 1193 1193 1133  
## 1194 1194 1261  
## 1195 1195 319  
## 1196 1196 471  
## 1197 1197 860  
## 1198 1198 1508  
## 1199 1199 190  
## 1200 1200 1434  
## 1201 1201 819  
## 1202 1202 1614  
## 1203 1203 1322  
## 1204 1204 1364  
## 1205 1205 1427  
## 1206 1206 1045  
## 1207 1207 958  
## 1208 1208 1090  
## 1209 1209 1312  
## 1210 1210 470  
## 1211 1211 1486  
## 1212 1212 1528  
## 1213 1213 499  
## 1214 1214 4  
## 1215 1215 37  
## 1216 1216 836  
## 1217 1217 552  
## 1218 1218 478  
## 1219 1219 1091  
## 1220 1220 257  
## 1221 1221 345  
## 1222 1222 162  
## 1223 1223 1225  
## 1224 1224 1351  
## 1225 1225 183  
## 1226 1226 1157  
## 1227 1227 1829  
## 1228 1228 205  
## 1229 1229 1428  
## 1230 1230 408  
## 1231 1231 330  
## 1232 1232 1068  
## 1233 1233 100  
## 1234 1234 335  
## 1235 1235 191  
## 1236 1236 59  
## 1237 1237 574  
## 1238 1238 562  
## 1239 1239 44  
## 1240 1240 1467  
## 1241 1241 737  
## 1242 1242 1431  
## 1243 1243 1389  
## 1244 1244 307  
## 1245 1245 171  
## 1246 1246 228  
## 1247 1247 795  
## 1248 1248 505  
## 1249 1249 558  
## 1250 1250 479  
## 1251 1251 1010  
## 1252 1252 585  
## 1253 1253 137  
## 1254 1254 1414  
## 1255 1255 322  
## 1256 1256 1123  
## 1257 1257 1028  
## 1258 1258 977  
## 1259 1259 170  
## 1260 1260 804  
## 1261 1261 568  
## 1262 1262 1306  
## 1263 1263 531  
## 1264 1264 24  
## 1265 1265 1021  
## 1266 1266 934  
## 1267 1267 26  
## 1268 1268 1269  
## 1269 1269 1075  
## 1270 1270 177  
## 1271 1271 375  
## 1272 1272 423  
## 1273 1273 1289  
## 1274 1274 272  
## 1275 1275 1171  
## 1276 1276 1567  
## 1277 1277 932  
## 1278 1278 1217  
## 1279 1279 46  
## 1280 1280 87  
## 1281 1281 602  
## 1282 1282 810  
## 1283 1283 241  
## 1284 1284 490  
## 1285 1285 1151  
## 1286 1286 944  
## 1287 1287 491  
## 1288 1288 576  
## 1289 1289 1105  
## 1290 1290 1146  
## 1291 1291 863  
## 1292 1292 536  
## 1293 1293 292  
## 1294 1294 1499  
## 1295 1295 1167  
## 1296 1296 549  
## 1297 1297 1552  
## 1298 1298 503  
## 1299 1299 871  
## 1300 1300 575  
## 1301 1301 1595  
## 1302 1302 994  
## 1303 1303 996  
## 1304 1304 1048  
## 1305 1305 1265  
## 1306 1306 1270  
## 1307 1307 782  
## 1308 1308 1007  
## 1309 1309 670  
## 1310 1310 751  
## 1311 1311 760  
## 1312 1312 1422  
## 1313 1313 1272  
## 1314 1314 513  
## 1315 1315 112  
## 1316 1316 1611  
## 1317 1317 1129  
## 1318 1318 306  
## 1319 1319 1479  
## 1320 1320 484  
## 1321 1321 14  
## 1322 1322 76  
## 1323 1323 687  
## 1324 1324 1047  
## 1325 1325 960  
## 1326 1326 1239  
## 1327 1327 61  
## 1328 1328 414  
## 1329 1329 1331  
## 1330 1330 249  
## 1331 1331 1164  
## 1332 1332 172  
## 1333 1333 1126  
## 1334 1334 129  
## 1335 1335 734  
## 1336 1336 68  
## 1337 1337 1019  
## 1338 1338 334  
## 1339 1339 282  
## 1340 1340 459  
## 1341 1341 600  
## 1342 1342 1163  
## 1343 1343 1152  
## 1344 1344 298  
## 1345 1345 1243  
## 1346 1346 1361  
## 1347 1347 386  
## 1348 1348 1260  
## 1349 1349 1139  
## 1350 1350 1185  
## 1351 1351 1429  
## 1352 1352 136  
## 1353 1353 1379  
## 1354 1354 1593  
## 1355 1355 572  
## 1356 1356 638  
## 1357 1357 1210  
## 1358 1358 537  
## 1359 1359 1158  
## 1360 1360 1142  
## 1361 1361 519  
## 1362 1362 480  
## 1363 1363 1070  
## 1364 1364 486  
## 1365 1365 532  
## 1366 1366 761  
## 1367 1367 261  
## 1368 1368 1092  
## 1369 1369 79  
## 1370 1370 555  
## 1371 1371 179  
## 1372 1372 790  
## 1373 1373 685  
## 1374 1374 1315  
## 1375 1375 149  
## 1376 1376 169  
## 1377 1377 199  
## 1378 1378 682  
## 1379 1379 382  
## 1380 1380 1278  
## 1381 1381 1398  
## 1382 1382 267  
## 1383 1383 684  
## 1384 1384 223  
## 1385 1385 457  
## 1386 1386 623  
## 1387 1387 224  
## 1388 1388 1275  
## 1389 1389 1073  
## 1390 1390 1150  
## 1391 1391 595  
## 1392 1392 340  
## 1393 1393 1328  
## 1394 1394 326  
## 1395 1395 117  
## 1396 1396 771  
## 1397 1397 442  
## 1398 1398 946  
## 1399 1399 1290  
## 1400 1400 444  
## 1401 1401 1305  
## 1402 1402 889  
## 1403 1403 53  
## 1404 1404 1020  
## 1405 1405 1039  
## 1406 1406 483  
## 1407 1407 383  
## 1408 1408 550  
## 1409 1409 38  
## 1410 1410 743  
## 1411 1411 1102  
## 1412 1412 538  
## 1413 1413 492  
## 1414 1414 80  
## 1415 1415 1350  
## 1416 1416 618  
## 1417 1417 321  
## 1418 1418 485  
## 1419 1419 816  
## 1420 1420 747  
## 1421 1421 625  
## 1422 1422 983  
## 1423 1423 559  
## 1424 1424 427  
## 1425 1425 675  
## 1426 1426 247  
## 1427 1427 50  
## 1428 1428 139  
## 1429 1429 632  
## 1430 1430 859  
## 1431 1431 710  
## 1432 1432 119  
## 1433 1433 1013  
## 1434 1434 943  
## 1435 1435 1533  
## 1436 1436 1577  
## 1437 1437 609  
## 1438 1438 1197  
## 1439 1439 265  
## 1440 1440 411  
## 1441 1441 1419  
## 1442 1442 463  
## 1443 1443 907  
## 1444 1444 546  
## 1445 1445 703  
## 1446 1446 512  
## 1447 1447 876  
## 1448 1448 1200  
## 1449 1449 1287  
## 1450 1450 750  
## 1451 1451 1001  
## 1452 1452 1100  
## 1453 1453 1319  
## 1454 1454 589  
## 1455 1455 857  
## 1456 1456 869  
## 1457 1457 884  
## 1458 1458 65  
## 1459 1459 1281  
## 1460 1460 564  
## 1461 1461 52  
## 1462 1462 1441  
## 1463 1463 1800  
## 1464 1464 1249  
## 1465 1465 1403  
## 1466 1466 1439  
## 1467 1467 799  
## 1468 1468 410  
## 1469 1469 1426  
## 1470 1470 1082  
## 1471 1471 175  
## 1472 1472 1360  
## 1473 1473 1060  
## 1474 1474 720  
## 1475 1475 591  
## 1476 1476 672  
## 1477 1477 1463  
## 1478 1478 1059  
## 1479 1479 642  
## 1480 1480 511  
## 1481 1481 425  
## 1482 1482 975  
## 1483 1483 1372  
## 1484 1484 82  
## 1485 1485 787  
## 1486 1486 783  
## 1487 1487 6  
## 1488 1488 971  
## 1489 1489 1326  
## 1490 1490 923  
## 1491 1491 1511  
## 1492 1492 496  
## 1493 1493 196  
## 1494 1494 1127  
## 1495 1495 130  
## 1496 1496 878  
## 1497 1497 34  
## 1498 1498 951  
## 1499 1499 1355  
## 1500 1500 141  
## 1501 1501 766  
## 1502 1502 1135  
## 1503 1503 606  
## 1504 1504 1066  
## 1505 1505 421  
## 1506 1506 1058  
## 1507 1507 308  
## 1508 1508 1231  
## 1509 1509 232  
## 1510 1510 952  
## 1511 1511 1044  
## 1512 1512 824  
## 1513 1513 577  
## 1514 1514 1592  
## 1515 1515 465  
## 1516 1516 99  
## 1517 1517 312  
## 1518 1518 222  
## 1519 1519 758  
## 1520 1520 587  
## 1521 1521 1016  
## 1522 1522 887  
## 1523 1523 147  
## 1524 1524 161  
## 1525 1525 1137  
## 1526 1526 698  
## 1527 1527 811  
## 1528 1528 954  
## 1529 1529 514  
## 1530 1530 976  
## 1531 1531 1300  
## 1532 1532 1267  
## 1533 1533 346  
## 1534 1534 580  
## 1535 1535 665  
## 1536 1536 667  
## 1537 1537 937  
## 1538 1538 280  
## 1539 1539 1184  
## 1540 1540 323  
## 1541 1541 763  
## 1542 1542 1053  
## 1543 1543 1153  
## 1544 1544 1008  
## 1545 1545 1226  
## 1546 1546 898  
## 1547 1547 985  
## 1548 1548 652  
## 1549 1549 1216  
## 1550 1550 561  
## 1551 1551 271  
## 1552 1552 733  
## 1553 1553 1545  
## 1554 1554 362  
## 1555 1555 263  
## 1556 1556 155  
## 1557 1557 305  
## 1558 1558 145  
## 1559 1559 303  
## 1560 1560 956  
## 1561 1561 488  
## 1562 1562 109  
## 1563 1563 1579  
## 1564 1564 658  
## 1565 1565 778  
## 1566 1566 1470  
## 1567 1567 297  
## 1568 1568 299  
## 1569 1569 1074  
## 1570 1570 917  
## 1571 1571 1125  
## 1572 1572 1022  
## 1573 1573 738  
## 1574 1574 1014  
## 1575 1575 1447  
## 1576 1576 921  
## 1577 1577 982  
## 1578 1578 608  
## 1579 1579 992  
## 1580 1580 1052  
## 1581 1581 1078  
## 1582 1582 1572  
## 1583 1583 1109  
## 1584 1584 621  
## 1585 1585 680  
## 1586 1586 775  
## 1587 1587 466  
## 1588 1588 501  
## 1589 1589 1271  
## 1590 1590 476  
## 1591 1591 118  
## 1592 1592 27  
## 1593 1593 473  
## 1594 1594 548  
## 1595 1595 55  
## 1596 1596 653  
## 1597 1597 159  
## 1598 1598 1336  
## 1599 1599 847  
## 1600 1600 1107  
## 1601 1601 972  
## 1602 1602 233  
## 1603 1603 620  
## 1604 1604 1341  
## 1605 1605 45  
## 1606 1606 1169  
## 1607 1607 88  
## 1608 1608 432  
## 1609 1609 469  
## 1610 1610 1706  
## 1611 1611 669  
## 1612 1612 1182  
## 1613 1613 767  
## 1614 1614 47  
## 1615 1615 229  
## 1616 1616 113  
## 1617 1617 877  
## 1618 1618 108  
## 1619 1619 791  
## 1620 1620 970  
## 1621 1621 1005  
## 1622 1622 895  
## 1623 1623 553  
## 1624 1624 1276  
## 1625 1625 1710  
## 1626 1626 1212  
## 1627 1627 1391  
## 1628 1628 543  
## 1629 1629 1518  
## 1630 1630 941  
## 1631 1631 657  
## 1632 1632 765  
## 1633 1633 278  
## 1634 1634 160  
## 1635 1635 1166  
## 1636 1636 1162  
## 1637 1637 903  
## 1638 1638 1280  
## 1639 1639 668  
## 1640 1640 154  
## 1641 1641 643  
## 1642 1642 148  
## 1643 1643 1301  
## 1644 1644 914  
## 1645 1645 1213  
## 1646 1646 1236  
## 1647 1647 753  
## 1648 1648 349  
## 1649 1649 1382  
## 1650 1650 1347  
## 1651 1651 57  
## 1652 1652 544  
## 1653 1653 798  
## 1654 1654 1065  
## 1655 1655 380  
## 1656 1656 1170  
## 1657 1657 92  
## 1658 1658 462  
## 1659 1659 590  
## 1660 1660 690  
## 1661 1661 739  
## 1662 1662 1064  
## 1663 1663 1456  
## 1664 1664 402  
## 1665 1665 258  
## 1666 1666 75  
## 1667 1667 1501  
## 1668 1668 506  
## 1669 1669 624  
## 1670 1670 1114  
## 1671 1671 156  
## 1672 1672 394  
## 1673 1673 1069  
## 1674 1674 560  
## 1675 1675 1384  
## 1676 1676 352  
## 1677 1677 854  
## 1678 1678 281  
## 1679 1679 714  
## 1680 1680 772  
## 1681 1681 1220  
## 1682 1682 1266  
## 1683 1683 1136  
## 1684 1684 724  
## 1685 1685 1180  
## 1686 1686 1549  
## 1687 1687 1308  
## 1688 1688 1076  
## 1689 1689 762  
## 1690 1690 48  
## 1691 1691 1619  
## 1692 1692 963  
## 1693 1693 1314  
## 1694 1694 123  
## 1695 1695 794  
## 1696 1696 1450  
## 1697 1697 1049  
## 1698 1698 289  
## 1699 1699 940  
## 1700 1700 792  
## 1701 1701 875  
## 1702 1702 1370  
## 1703 1703 430  
## 1704 1704 437  
## 1705 1705 969  
## 1706 1706 966  
## 1707 1707 1113  
## 1708 1708 644  
## 1709 1709 711  
## 1710 1710 107  
## 1711 1711 1362  
## 1712 1712 981  
## 1713 1713 495  
## 1714 1714 927  
## 1715 1715 1192  
## 1716 1716 498  
## 1717 1717 126  
## 1718 1718 822  
## 1719 1719 279  
## 1720 1720 534  
## 1721 1721 1154  
## 1722 1722 1438  
## 1723 1723 978  
## 1724 1724 1304  
## 1725 1725 125  
## 1726 1726 1445  
## 1727 1727 1046  
## 1728 1728 1112  
## 1729 1729 369  
## 1730 1730 749  
## 1731 1731 295  
## 1732 1732 1029  
## 1733 1733 201  
## 1734 1734 1688  
## 1735 1735 1397  
## 1736 1736 244  
## 1737 1737 1293  
## 1738 1738 1294  
## 1739 1739 841  
## 1740 1740 1202  
## 1741 1741 1411  
## 1742 1742 395  
## 1743 1743 1187  
## 1744 1744 1259  
## 1745 1745 647  
## 1746 1746 1117  
## 1747 1747 1421  
## 1748 1748 1089  
## 1749 1749 359  
## 1750 1750 9  
## 1751 1751 554  
## 1752 1752 807  
## 1753 1753 886  
## 1754 1754 980  
## 1755 1755 1025  
## 1756 1756 904  
## 1757 1757 361  
## 1758 1758 1003  
## 1759 1759 1072  
## 1760 1760 1177  
## 1761 1761 578  
## 1762 1762 715  
## 1763 1763 235  
## 1764 1764 418  
## 1765 1765 1256  
## 1766 1766 868  
## 1767 1767 264  
## 1768 1768 374  
## 1769 1769 634  
## 1770 1770 320  
## 1771 1771 988  
## 1772 1772 1274  
## 1773 1773 1432  
## 1774 1774 947  
## 1775 1775 1420  
## 1776 1776 504  
## 1777 1777 843  
## 1778 1778 275  
## 1779 1779 1056  
## 1780 1780 396  
## 1781 1781 1061  
## 1782 1782 746  
## 1783 1783 1521  
## 1784 1784 12  
## 1785 1785 929  
## 1786 1786 815  
## 1787 1787 226  
## 1788 1788 234  
## 1789 1789 849  
## 1790 1790 246  
## 1791 1791 829  
## 1792 1792 1424  
## 1793 1793 66  
## 1794 1794 1207  
## 1795 1795 16  
## 1796 1796 1491  
## 1797 1797 472  
## 1798 1798 304  
## 1799 1799 671  
## 1800 1800 1324  
## 1801 1801 507  
## 1802 1802 78  
## 1803 1803 825  
## 1804 1804 1160  
## 1805 1805 372  
## 1806 1806 225  
## 1807 1807 526  
## 1808 1808 1297  
## 1809 1809 331  
## 1810 1810 178  
## 1811 1811 1050  
## 1812 1812 1343  
## 1813 1813 1453  
## 1814 1814 1496  
## 1815 1815 1  
## 1816 1816 1124  
## 1817 1817 255  
## 1818 1818 336  
## 1819 1819 381  
## 1820 1820 103  
## 1821 1821 1145  
## 1822 1822 1057  
## 1823 1823 821  
## 1824 1824 373  
## 1825 1825 1648  
## 1826 1826 915  
## 1827 1827 861  
## 1828 1828 500  
## 1829 1829 428  
## 1830 1830 1387  
## 1831 1831 786  
## 1832 1832 991  
## 1833 1833 721  
## 1834 1834 1402  
## 1835 1835 831  
## 1836 1836 198  
## 1837 1837 1172  
## 1838 1838 412  
## 1839 1839 569  
## 1840 1840 666  
## 1841 1841 17  
## 1842 1842 29  
## 1843 1843 254  
## 1844 1844 182  
## 1845 1845 1031  
## 1846 1846 1338  
## 1847 1847 1356  
## 1848 1848 820  
## 1849 1849 520  
## 1850 1850 33  
## 1851 1851 1423  
## 1852 1852 1097  
## 1853 1853 902  
## 1854 1854 916  
## 1855 1855 583  
## 1856 1856 1448  
## 1857 1857 848  
## 1858 1858 812  
## 1859 1859 768  
## 1860 1860 206  
## 1861 1861 582  
## 1862 1862 1132  
## 1863 1863 612  
## 1864 1864 489  
## 1865 1865 124  
## 1866 1866 194  
## 1867 1867 1318  
## 1868 1868 777  
## 1869 1869 1108  
## 1870 1870 1560  
## 1871 1871 508  
## 1872 1872 1263  
## 1873 1873 633  
## 1874 1874 1130  
## 1875 1875 764  
## 1876 1876 616  
## 1877 1877 1621  
## 1878 1878 1122  
## 1879 1879 448  
## 1880 1880 1635  
## 1881 1881 461  
## 1882 1882 152  
## 1883 1883 1051  
## 1884 1884 144  
## 1885 1885 726  
## 1886 1886 728  
## 1887 1887 1043  
## 1888 1888 464  
## 1889 1889 900  
## 1890 1890 925  
## 1891 1891 584  
## 1892 1892 756  
## 1893 1893 133  
## 1894 1894 1367  
## 1895 1895 447  
## 1896 1896 1134  
## 1897 1897 262  
## 1898 1898 881  
## 1899 1899 285  
## 1900 1900 357  
## 1901 1901 910  
## 1902 1902 1190  
## 1903 1903 1386  
## 1904 1904 596  
## 1905 1905 1233  
## 1906 1906 646  
## 1907 1907 545  
## 1908 1908 957  
## 1909 1909 1369  
## 1910 1910 864  
## 1911 1911 528  
## 1912 1912 1121  
## 1913 1913 567  
## 1914 1914 1228  
## 1915 1915 853  
## 1916 1916 1472  
## 1917 1917 1366  
## 1918 1918 833  
## 1919 1919 218  
## 1920 1920 114  
## 1921 1921 413  
## 1922 1922 1034  
## 1923 1923 18  
## 1924 1924 1209  
## 1925 1925 1245  
## 1926 1926 439  
## 1927 1927 1349  
## 1928 1928 1353  
## 1929 1929 189  
## 1930 1930 757  
## 1931 1931 2  
## 1932 1932 433  
## 1933 1933 122  
## 1934 1934 135  
## 1935 1935 185  
## 1936 1936 163  
## 1937 1937 748  
## 1938 1938 676  
## 1939 1939 86  
## 1940 1940 204  
## 1941 1941 435  
## 1942 1942 673  
## 1943 1943 1120  
## 1944 1944 378  
## 1945 1945 803  
## 1946 1946 397  
## 1947 1947 415  
## 1948 1948 1159  
## 1949 1949 1395  
## 1950 1950 741  
## 1951 1951 221  
## 1952 1952 309  
## 1953 1953 967  
## 1954 1954 1257  
## 1955 1955 697  
## 1956 1956 20  
## 1957 1957 920  
## 1958 1958 1144  
## 1959 1959 540  
## 1960 1960 968  
## 1961 1961 779  
## 1962 1962 127  
## 1963 1963 1093  
## 1964 1964 32  
## 1965 1965 454  
## 1966 1966 142  
## 1967 1967 1103  
## 1968 1968 718  
## 1969 1969 521  
## 1970 1970 1569  
## 1971 1971 610  
## 1972 1972 1385  
## 1973 1973 865  
## 1974 1974 377  
## 1975 1975 1038  
## 1976 1976 274  
## 1977 1977 1258  
## 1978 1978 640  
## 1979 1979 817  
## 1980 1980 692  
## 1981 1981 1627  
## 1982 1982 628  
## 1983 1983 83  
## 1984 1984 1279  
## 1985 1985 1004  
## 1986 1986 1248  
## 1987 1987 1186  
## 1988 1988 389  
## 1989 1989 1115  
## 1990 1990 215  
## 1991 1991 1168  
## 1992 1992 327  
## 1993 1993 1255  
## 1994 1994 1067  
## 1995 1995 1752  
## 1996 1996 283  
## 1997 1997 216  
## 1998 1998 1009  
## 1999 1999 1405  
## 2000 2000 740

summary(kmeans.mydata)

## V1 V2   
## Min. : 1.0 Min. : 1.0   
## 1st Qu.: 500.8 1st Qu.: 500.8   
## Median :1000.5 Median : 997.5   
## Mean :1000.5 Mean : 998.6   
## 3rd Qu.:1500.2 3rd Qu.:1497.2   
## Max. :2000.0 Max. :1997.0

#Creating the sample  
mysample <- sample(nrow(kmeans.mydata),300,replace = FALSE)  
summary(mysample)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 1.0 579.5 1074.0 1044.0 1557.0 1999.0

km.outHW <- kmeans(mysample,centers=2,nstart=20, algorithm = "Hartigan-Wong")  
km.outLl <- kmeans(mysample,centers=2,nstart=20, algorithm = "Lloyd")

Hartigan-Wong and Lloyd appear to give the same results, paying attention to Within Sum and Between/Total ratio, as well as the areas in the center of the graph where observations might slip from one class to another between algorithms. I'll use the default Hartigan Wong.

## 2.

*Explain iter:max and algorithm parameters of kmeans function in R and run k-means on mysample data set where nstart = 35 and k = 2. Report total within squares error and within squares error for each cluster.*

set.seed(1234)  
km.out2 <- kmeans(mysample,centers=2,nstart=35)  
km.out2$cluster

## [1] 1 2 2 2 1 1 1 2 1 1 1 2 2 2 2 2 2 2 1 1 2 1 2 2 2 1 1 2 1 2 2 2 2 1 1  
## [36] 2 2 1 1 1 2 1 2 2 1 1 1 1 1 2 2 2 2 2 1 2 2 2 2 2 1 1 1 2 1 2 2 1 2 2  
## [71] 2 2 1 1 1 1 1 1 1 2 1 1 1 2 1 2 1 1 2 1 1 1 1 2 2 2 2 2 2 2 1 2 2 2 1  
## [106] 2 2 2 2 2 1 1 2 2 1 2 1 2 2 1 1 2 1 2 2 2 1 1 1 2 2 1 2 1 2 2 1 2 2 1  
## [141] 2 2 2 1 2 2 2 2 2 2 2 1 2 2 1 2 1 1 1 1 1 1 2 1 2 2 1 1 1 1 2 1 2 2 1  
## [176] 1 1 1 1 2 2 1 1 2 1 2 1 1 1 2 2 2 1 1 1 1 2 2 1 1 1 2 2 2 2 1 2 1 2 2  
## [211] 1 1 2 1 2 1 2 1 2 2 2 2 1 1 1 2 2 2 1 1 2 2 2 1 1 1 2 1 1 2 1 1 1 2 2  
## [246] 1 2 1 2 2 2 1 1 2 2 1 2 2 1 2 2 1 1 2 1 1 2 1 1 1 2 1 1 1 2 2 1 2 1 1  
## [281] 2 1 2 2 2 2 2 1 1 1 2 1 2 1 1 1 2 2 1 2

#Total within-cluster sum of squares  
km.out2$tot.withinss

## [1] 24318617

# within-cluster sum of squares for each cluster  
km.out2$withinss

## [1] 11826520 12492097

km.out2

## K-means clustering with 2 clusters of sizes 144, 156  
##   
## Cluster means:  
## [,1]  
## 1 518.4792  
## 2 1530.0321  
##   
## Clustering vector:  
## [1] 1 2 2 2 1 1 1 2 1 1 1 2 2 2 2 2 2 2 1 1 2 1 2 2 2 1 1 2 1 2 2 2 2 1 1  
## [36] 2 2 1 1 1 2 1 2 2 1 1 1 1 1 2 2 2 2 2 1 2 2 2 2 2 1 1 1 2 1 2 2 1 2 2  
## [71] 2 2 1 1 1 1 1 1 1 2 1 1 1 2 1 2 1 1 2 1 1 1 1 2 2 2 2 2 2 2 1 2 2 2 1  
## [106] 2 2 2 2 2 1 1 2 2 1 2 1 2 2 1 1 2 1 2 2 2 1 1 1 2 2 1 2 1 2 2 1 2 2 1  
## [141] 2 2 2 1 2 2 2 2 2 2 2 1 2 2 1 2 1 1 1 1 1 1 2 1 2 2 1 1 1 1 2 1 2 2 1  
## [176] 1 1 1 1 2 2 1 1 2 1 2 1 1 1 2 2 2 1 1 1 1 2 2 1 1 1 2 2 2 2 1 2 1 2 2  
## [211] 1 1 2 1 2 1 2 1 2 2 2 2 1 1 1 2 2 2 1 1 2 2 2 1 1 1 2 1 1 2 1 1 1 2 2  
## [246] 1 2 1 2 2 2 1 1 2 2 1 2 2 1 2 2 1 1 2 1 1 2 1 1 1 2 1 1 1 2 2 1 2 1 1  
## [281] 2 1 2 2 2 2 2 1 1 1 2 1 2 1 1 1 2 2 1 2  
##   
## Within cluster sum of squares by cluster:  
## [1] 11826520 12492097  
## (between\_SS / total\_SS = 75.9 %)  
##   
## Available components:  
##   
## [1] "cluster" "centers" "totss" "withinss"   
## [5] "tot.withinss" "betweenss" "size" "iter"   
## [9] "ifault"

## km.out3

The itermax parameter sets the number of iterations the algorithm will perform. The algorithm selects one of at least four different algorithms to use. H The total within squares error and total squares errors are included for both clusters. Within sum of squares error indicates the total distance between each point in a variable and the center point of that variable. The between sum of squares indicates how far the two variables are from each other.

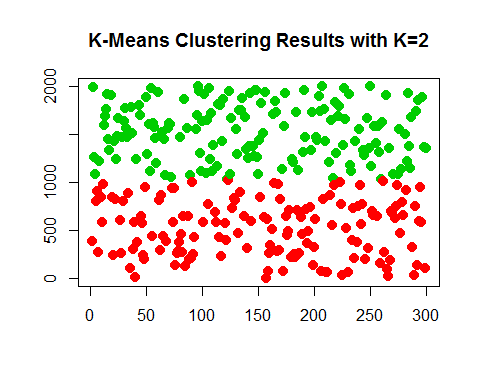
Total Within SS = 25,101,542

Within SS for each cluster = 13454956 11646587

## 3.

*Make a plot of data points and color the observation according to the cluster labels obtained.*

plot(mysample, col=(km.out2$cluster+1), main="K-Means Clustering Results with K=2",  
 xlab="", ylab="", pch=20, cex=2)



## 4.

Run k-means on mysample data set where nstart = 35 and k = 4. Report total within squares error and within squares error for each cluster.

km.out4 <- kmeans(mysample,centers=4,nstart=35)  
km.out4

## K-means clustering with 4 clusters of sizes 81, 76, 68, 75  
##   
## Cluster means:  
## [,1]  
## 1 1765.0370  
## 2 1272.8158  
## 3 254.1324  
## 4 751.5067  
##   
## Clustering vector:  
## [1] 3 1 2 2 4 4 3 2 4 4 4 1 1 1 1 2 2 1 4 3 2 4 2 2 1 4 3 2 4 1 1 1 2 4 3  
## [36] 1 2 3 4 3 2 3 1 1 4 4 3 3 4 1 2 1 2 1 3 1 1 2 2 1 4 3 4 1 3 2 2 3 1 1  
## [71] 1 2 4 4 4 3 3 3 3 2 3 3 4 1 3 1 4 3 2 3 4 3 3 1 1 1 1 2 2 1 4 1 2 1 4  
## [106] 1 2 1 2 2 4 4 2 1 3 1 3 2 1 4 3 2 2 1 2 1 4 4 4 1 2 3 1 4 1 1 4 1 2 3  
## [141] 2 2 1 4 2 2 1 1 2 2 2 4 1 2 4 1 3 4 3 3 3 4 1 4 1 1 3 4 3 4 2 3 1 1 4  
## [176] 3 3 4 3 2 2 3 3 1 4 2 3 4 3 2 1 2 4 3 3 4 2 1 3 3 4 2 2 1 1 3 1 4 1 2  
## [211] 3 3 2 4 1 4 2 4 1 2 1 1 4 4 3 2 1 1 4 3 1 2 2 3 4 3 2 3 4 1 4 4 4 2 2  
## [246] 3 1 3 2 1 2 4 4 1 2 4 1 2 3 2 1 4 3 1 3 3 2 3 4 4 2 4 4 4 2 1 3 2 4 4  
## [281] 2 4 2 1 2 2 1 3 3 4 1 3 1 4 4 4 1 2 3 2  
##   
## Within cluster sum of squares by cluster:  
## [1] 1674599 1579109 1255556 1498017  
## (between\_SS / total\_SS = 94.0 %)  
##   
## Available components:  
##   
## [1] "cluster" "centers" "totss" "withinss"   
## [5] "tot.withinss" "betweenss" "size" "iter"   
## [9] "ifault"

km.out4$tot.withinss

## [1] 6007281

km.out4$withinss

## [1] 1674599 1579109 1255556 1498017

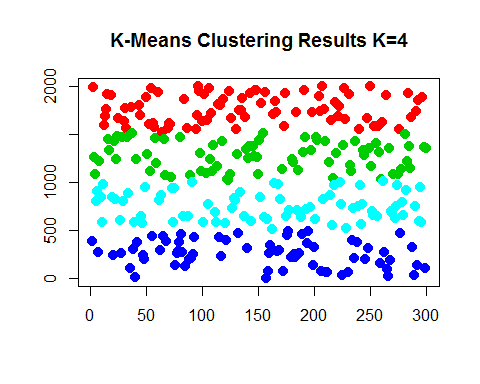
The Within sum of squares by cluster is as follows: - C1: 1245029 - C2: 1343345 - C3: 980807 - C4: 2702241

Total within squares error is 6,271,425

## 5.

Make a plot of data points and color the observation according to the cluster labels obtained.

plot(mysample, col=(km.out4$cluster+1), main="K-Means Clustering Results K=4",  
 xlab="", ylab="", pch=20, cex=2)

 ##6. Compare (2) and (4). With a higher number of clusters, we obviously have a lower total within sum of square error, as there are more centroids, thus a shorter distance and less overall error. There is an interesting function in the text that describes how to determine the optimum number of clusters between two and six:

library(cluster)  
  
set.seed(1234)  
d <- dist(mydata[,-5])  
avgS <- c()  
  
for(k in 2:6) {  
 cl <- kmeans(mydata[,-5],centers=k,iter.max=200)  
 s <- silhouette(cl$cluster,d)  
 avgS <- c(avgS,mean(s[,3]))  
}  
data.frame(nClus=2:6,Silh=avgS)

## nClus Silh  
## 1 2 0.3724321  
## 2 3 0.4216897  
## 3 4 0.3795938  
## 4 5 0.3583321  
## 5 6 0.3645822

This appears to indicate that the optimum number of clusters is 3, and it appears that four clusters is slightly better than two clusters in this case.

# 4

## 1.

In the listing below, which line number eectively reads the data into an R data.frame?

This is the correct method for reading this data into R: teach <???? read.table( f i l e ="c:/R/rainfalldataraw.txt", header=TRUE, sep=",")

teach <- read.table("C:\\Users\\khickman\\Desktop\\Personal\\IUMSDS\\AppliedDataMining\\Midterm\\rainfalldataraw.txt", header = TRUE, sep=",")  
teach

## SEEDED SEASON A B C D E  
## 1 S AUTUMN 1.69 3.730 1.65 1.80 3.33  
## 2 U AUTUMN 0.74 0.780 1.09 0.79 1.59  
## 3 S WINTER 0.81 0.860 2.39 0.36 2.06  
## 4 U WINTER 1.44 2.010 2.96 1.27 4.05  
## 5 S WINTER 2.48 4.610 4.16 2.16 6.00  
## 6 U WINTER 0.84 2.390 2.76 0.87 4.17  
## 7 U WINTER 0.37 1.370 1.08 0.85 3.45  
## 8 S WINTER 0.37 0.840 0.26 0.47 0.90  
## 9 U SPRING 1.33 2.310 2.53 1.08 3.65  
## 10 S SPRING 3.38 5.560 2.76 3.10 5.06  
## 11 S SPRING 0.69 1.460 1.07 0.64 1.95  
## 12 U SPRING 1.42 2.790 1.42 1.08 1.22  
## 13 S SPRING 0.44 1.050 0.24 0.44 0.94  
## 14 U SPRING 0.76 1.240 0.70 0.67 0.94  
## 15 S SUMMER 1.13 2.280 0.97 1.66 2.21  
## 16 U SUMMER 0.88 1.580 1.06 1.13 1.46  
## 17 S SUMMER 0.17 0.550 0.13 0.27 0.35  
## 18 U SUMMER 0.25 0.770 0.10 0.30 0.34  
## 19 U SUMMER 0.78 1.450 0.38 0.58 0.67  
## 20 S SUMMER 0.40 0.340 0.45 0.43 0.44  
## 21 S AUTUMN 0.52 0.790 0.42 0.47 0.53  
## 22 U AUTUMN 2.73 2.090 2.24 4.02 2.52  
## 23 U AUTUMN 0.90 2.450 0.52 1.32 2.18  
## 24 S AUTUMN 1.62 2.540 0.94 1.59 1.73  
## 25 U AUTUMN 0.93 2.110 1.19 0.85 2.31  
## 26 S AUTUMN 0.63 1.310 0.76 0.71 1.28  
## 27 S WINTER 0.42 1.230 0.13 0.59 0.91  
## 28 U WINTER 0.64 0.430 1.50 0.24 1.15  
## 29 U WINTER 0.30 0.690 1.03 0.22 1.88  
## 30 S WINTER 0.88 1.320 1.87 0.58 2.97  
## 31 WINTER 0.76 1.250 1.85 1.36 2.17  
## 32 S WINTER 1.25 1.000 2.04 0.71 2.22  
## 33 U WINTER 1.08 0.990 1.44 1.00 1.64  
## 34 S WINTER 1.11 0.800 1.46 1.48 0.40  
## 35 S SPRING 3.43 2.550 5.08 1.77 4.20  
## 36 U SPRING 0.54 0.430 0.66 0.73 0.91  
## 37 S SPRING 0.39 0.440 0.49 0.55 0.51  
## 38 U SPRING 2.53 3.180 3.27 2.68 3.60  
## 39 U SPRING 0.81 0.890 1.33 0.43 2.18  
## 40 S SPRING 0.39 1.220 0.25 0.46 0.89  
## 41 S SUMMER 0.86 1.240 0.69 0.49 0.69  
## 42 U SUMMER 2.16 2.290 2.12 0.95 1.82  
## 43 U SPRING 1.70 2.180 1.45 1.47 2.20  
## 44 S SPRING 1.22 2.000 2.13 1.13 2.33  
## 45 S SPRING 0.07 0.220 0.02 0.08 0.24  
## 46 U SPRING 0.49 1.070 0.36 0.87 0.57  
## 47 U SPRING 0.71 1.730 0.72 0.99 0.98  
## 48 S SPRING 1.67 3.460 1.02 1.89 2.47  
## 49 U SUMMER 0.73 1.510 0.18 1.42 0.71  
## 50 S SUMMER 1.79 3.130 1.83 1.82 3.11  
## 51 U SUMMER 0.19 1.050 0.08 0.40 0.57  
## 52 S SUMMER 0.00 0.150 0.00 0.04 0.04  
## 53 S SUMMER 0.44 0.890 0.83 0.38 0.70  
## 54 U SUMMER 0.31 1.150 0.01 0.44 0.66  
## 55 S SUMMER 0.96 0.880 2.65 0.85 1.48  
## 56 U SUMMER 1.04 1.200 1.27 1.39 1.20  
## 57 S AUTUMN 0.05 0.060 0.01 200.30 0.10  
## 58 U AUTUMN 0.04 0.200 0.35 0.75 0.20  
## 59 S AUTUMN 1.83 2.930 1.80 1.62 3.02  
## 60 U AUTUMN 2.24 2.170 4.44 1.05 3.59  
## 61 S AUTUMN 2.50 3.990 2.84 2.44 4.48  
## 62 U AUTUMN 1.10 1.710 2.05 1.30 4.04  
## 63 S AUTUMN 1.83 3.870 3.01 1.66 4.56  
## 64 U AUTUMN 1.41 -0.034 2.58 1.21 3.95  
## 65 U WINTER 0.74 1.360 2.22 0.61 2.68  
## 66 S WINTER 1.09 3.560 0.07 2.26 2.08  
## 67 S WINTER 0.79 1.430 1.62 1.16 2.87  
## 68 U WINTER 4.06 6.710 4.34 3.29 6.40  
## 69 U WINTER 0.40 0.640 1.03 0.58 1.77  
## 70 S WINTER 0.76 1.830 1.50 0.41 2.56  
## 71 S SPRING 1.53 3.620 1.52 1.62 2.86  
## 72 U SPRING 0.56 2.880 0.37 1.25 1.74  
## 73 U SPRING 1.74 3.450 2.14 1.00 4.39  
## 74 S SPRING 1.59 3.190 2.36 1.53 3.03  
## 75 U SPRING 1.91 4.740 1.71 2.03 3.24  
## 76 S SPRING 2.09 5.230 2.12 2.77 4.44  
## 77 U SUMMER 1.59 3.920 1.38 2.11 3.01  
## 78 S SUMMER 0.66 2.220 0.21 1.41 0.80  
## 79 U SUMMER 0.68 0.420 0.48 0.59 0.68  
## 80 S SUMMER 0.46 1.080 0.01 0.65 0.48  
## 81 S SUMMER 0.22 0.620 0.15 0.13 0.42  
## 82 U SUMMER 1.11 1.700 1.32 0.57 1.54  
## 83 S SUMMER 1.76 1.190 2.26 1.04 1.27  
## 84 U SUMMER 5.12 5.250 5.95 3.97 5.37  
## 85 U AUTUMN 0.12 0.600 0.19 0.28 0.70  
## 86 S AUTUMN 0.37 NA 0.31 0.23 0.83  
## 87 S AUTUMN 4.97 3.030 1.44 3.14 0.86  
## 88 U AUTUMN 0.57 1.530 0.30 0.72 1.38  
## 89 S AUTUMN 0.13 0.540 0.11 0.14 0.58  
## 90 U AUTUMN 2.47 4.700 3.66 1.84 5.36  
## 91 U AUTUMN 1.01 2.320 1.14 0.81 2.09  
## 92 S AUTUMN 0.55 1.130 1.30 NA 2.45  
## 93 S WINTER 0.24 0.610 0.05 0.38 0.90  
## 94 U WINTER 2.36 1.150 1.84 1.73 2.33  
## 95 S WINTER 2.35 4.290 4.24 1.67 5.48  
## 96 U WINTER 2.23 4.300 1.99 1.90 3.67  
## 97 U WINTER 1.16 3.060 2.44 1.52 4.01  
## 98 S WINTER 1.63 3.310 2.21 2.36 3.25  
## 99 S WINTER 1.08 3.170 0.80 2.25 2.79  
## 100 U WINTER 6.00 6.150 9.42 3.60 7.84  
## 101 S SPRING 2.67 6.930 NA 3.03 6.39  
## 102 U SPRING 0.36 0.150 0.00 0.19 0.06  
## 103 S SPRING 0.58 1.410 0.96 0.64 1.24  
## 104 U SPRING 1.36 3.430 1.38 1.86 2.91  
## 105 S SPRING 1.17 1.650 1.22 2.28 1.58  
## 106 U SPRING 2.37 1.940 2.46 2.47 2.39  
## 107 S 0.02 0.080 0.05 0.02 0.09  
## 108 U SPRING 0.92 2.090 0.61 0.87 1.35  
## 109 S SPRING 3.43 2.550 5.08 1.77 4.20

str(teach)

## 'data.frame': 109 obs. of 7 variables:  
## $ SEEDED: Factor w/ 3 levels "","S","U": 2 3 2 3 2 3 3 2 3 2 ...  
## $ SEASON: Factor w/ 5 levels "","AUTUMN","SPRING",..: 2 2 5 5 5 5 5 5 3 3 ...  
## $ A : num 1.69 0.74 0.81 1.44 2.48 0.84 0.37 0.37 1.33 3.38 ...  
## $ B : num 3.73 0.78 0.86 2.01 4.61 2.39 1.37 0.84 2.31 5.56 ...  
## $ C : num 1.65 1.09 2.39 2.96 4.16 2.76 1.08 0.26 2.53 2.76 ...  
## $ D : num 1.8 0.79 0.36 1.27 2.16 0.87 0.85 0.47 1.08 3.1 ...  
## $ E : num 3.33 1.59 2.06 4.05 6 4.17 3.45 0.9 3.65 5.06 ...

## 2.

Give a select operation on the data.frame that gives the rows whose E variable values are greater than 4, but less than 5.

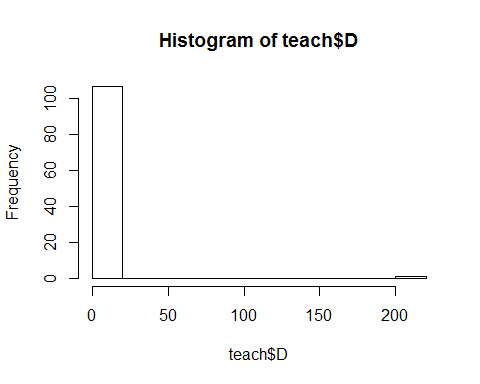
??subset  
teach.sub <- teach[which(teach$E > 4 & teach$E < 5),]  
teach.sub

## SEEDED SEASON A B C D E  
## 4 U WINTER 1.44 2.01 2.96 1.27 4.05  
## 6 U WINTER 0.84 2.39 2.76 0.87 4.17  
## 35 S SPRING 3.43 2.55 5.08 1.77 4.20  
## 61 S AUTUMN 2.50 3.99 2.84 2.44 4.48  
## 62 U AUTUMN 1.10 1.71 2.05 1.30 4.04  
## 63 S AUTUMN 1.83 3.87 3.01 1.66 4.56  
## 73 U SPRING 1.74 3.45 2.14 1.00 4.39  
## 76 S SPRING 2.09 5.23 2.12 2.77 4.44  
## 97 U WINTER 1.16 3.06 2.44 1.52 4.01  
## 109 S SPRING 3.43 2.55 5.08 1.77 4.20

## 3.

Give the code that produces the histogram of variable D.

hist(teach$D)



## 4.

How many tuples (or records) are in the data?

summary(teach)

## SEEDED SEASON A B C   
## : 1 : 1 Min. :0.000 Min. :-0.034 Min. :0.000   
## S:55 AUTUMN:24 1st Qu.:0.520 1st Qu.: 0.890 1st Qu.:0.410   
## U:53 SPRING:32 Median :0.920 Median : 1.555 Median :1.285   
## SUMMER:24 Mean :1.253 Mean : 2.036 Mean :1.528   
## WINTER:28 3rd Qu.:1.690 3rd Qu.: 2.955 3rd Qu.:2.132   
## Max. :6.000 Max. : 6.930 Max. :9.420   
## NA's :1 NA's :1   
## D E   
## Min. : 0.0200 Min. :0.040   
## 1st Qu.: 0.5775 1st Qu.:0.890   
## Median : 1.0200 Median :1.950   
## Mean : 3.0679 Mean :2.211   
## 3rd Qu.: 1.7400 3rd Qu.:3.110   
## Max. :200.3000 Max. :7.840   
## NA's :1

There are 109 observations of 7 variables.

## 5.

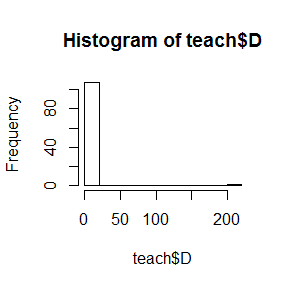
*Identify the data that is either missing or likely corrupted:*  Most of the variables here have at least one missing or corrupted value. Seeded, Season, B, C, and D all have missing or NA values.

## 6.

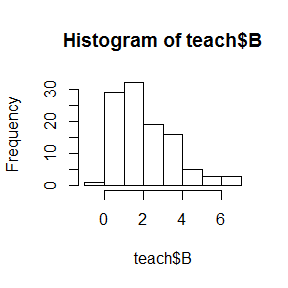
*Preprocess the data, addressing the problems above and save the file as rainfixed.txt as a .csv file. Explain explicitly what you have done in preprocessing this file.*

Since the number of missing values are relatively small, we can either remove the cases with NA, or we can impute the values using a statistic of centrality like mean. We're dealing with two types of variables here as well so we might use two different methods. Examining the distribution of each variable:

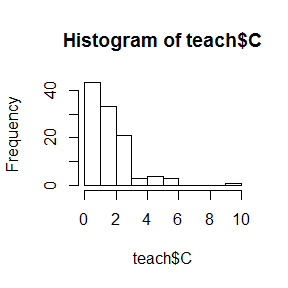
hist(teach$D)



hist(teach$B)



hist(teach$C)



Since none of the three variables is normally distributed, the median is the preferred imputation statistic.

na.var <- teach[rowSums(is.na(teach)) > 0,]  
na.var

## SEEDED SEASON A B C D E  
## 86 S AUTUMN 0.37 NA 0.31 0.23 0.83  
## 92 S AUTUMN 0.55 1.13 1.30 NA 2.45  
## 101 S SPRING 2.67 6.93 NA 3.03 6.39

teach[86, "B"] <- median(teach$B, na.rm = TRUE)  
na.var

## SEEDED SEASON A B C D E  
## 86 S AUTUMN 0.37 NA 0.31 0.23 0.83  
## 92 S AUTUMN 0.55 1.13 1.30 NA 2.45  
## 101 S SPRING 2.67 6.93 NA 3.03 6.39

teach[92,"D"] <- median(teach$D, na.rm = TRUE)  
na.var

## SEEDED SEASON A B C D E  
## 86 S AUTUMN 0.37 NA 0.31 0.23 0.83  
## 92 S AUTUMN 0.55 1.13 1.30 NA 2.45  
## 101 S SPRING 2.67 6.93 NA 3.03 6.39

teach[101,"C"] <- median(teach$C, na.rm = TRUE)  
  
na.var

## SEEDED SEASON A B C D E  
## 86 S AUTUMN 0.37 NA 0.31 0.23 0.83  
## 92 S AUTUMN 0.55 1.13 1.30 NA 2.45  
## 101 S SPRING 2.67 6.93 NA 3.03 6.39

summary(teach)

## SEEDED SEASON A B C   
## : 1 : 1 Min. :0.000 Min. :-0.034 Min. :0.000   
## S:55 AUTUMN:24 1st Qu.:0.520 1st Qu.: 0.890 1st Qu.:0.420   
## U:53 SPRING:32 Median :0.920 Median : 1.555 Median :1.285   
## SUMMER:24 Mean :1.253 Mean : 2.032 Mean :1.526   
## WINTER:28 3rd Qu.:1.690 3rd Qu.: 2.930 3rd Qu.:2.130   
## Max. :6.000 Max. : 6.930 Max. :9.420   
## D E   
## Min. : 0.020 Min. :0.040   
## 1st Qu.: 0.580 1st Qu.:0.890   
## Median : 1.020 Median :1.950   
## Mean : 3.049 Mean :2.211   
## 3rd Qu.: 1.730 3rd Qu.:3.110   
## Max. :200.300 Max. :7.840

It appears that all of our unknown numeric variables have been replaced. Now on to the categorical variables.

teach$SEEDED

## [1] S U S U S U U S U S S U S U S U S U U S S U U S U S S U U S S U S S  
## [36] U S U U S S U U S S U U S U S U S S U S U S U S U S U S U U S S U U S  
## [71] S U U S U S U S U S S U S U U S S U S U U S S U S U U S S U S U S U S  
## [106] U S U S  
## Levels: S U

teach$SEASON

## [1] AUTUMN AUTUMN WINTER WINTER WINTER WINTER WINTER WINTER SPRING SPRING  
## [11] SPRING SPRING SPRING SPRING SUMMER SUMMER SUMMER SUMMER SUMMER SUMMER  
## [21] AUTUMN AUTUMN AUTUMN AUTUMN AUTUMN AUTUMN WINTER WINTER WINTER WINTER  
## [31] WINTER WINTER WINTER WINTER SPRING SPRING SPRING SPRING SPRING SPRING  
## [41] SUMMER SUMMER SPRING SPRING SPRING SPRING SPRING SPRING SUMMER SUMMER  
## [51] SUMMER SUMMER SUMMER SUMMER SUMMER SUMMER AUTUMN AUTUMN AUTUMN AUTUMN  
## [61] AUTUMN AUTUMN AUTUMN AUTUMN WINTER WINTER WINTER WINTER WINTER WINTER  
## [71] SPRING SPRING SPRING SPRING SPRING SPRING SUMMER SUMMER SUMMER SUMMER  
## [81] SUMMER SUMMER SUMMER SUMMER AUTUMN AUTUMN AUTUMN AUTUMN AUTUMN AUTUMN  
## [91] AUTUMN AUTUMN WINTER WINTER WINTER WINTER WINTER WINTER WINTER WINTER  
## [101] SPRING SPRING SPRING SPRING SPRING SPRING SPRING SPRING  
## Levels: AUTUMN SPRING SUMMER WINTER

teach.clean <- teach[-c(31, 107), ]  
summary(teach.clean)

## SEEDED SEASON A B C   
## : 0 : 0 Min. :0.000 Min. :-0.034 Min. :0.000   
## S:54 AUTUMN:24 1st Qu.:0.530 1st Qu.: 0.940 1st Qu.:0.435   
## U:53 SPRING:32 Median :0.930 Median : 1.580 Median :1.285   
## SUMMER:24 Mean :1.269 Mean : 2.057 Mean :1.537   
## WINTER:27 3rd Qu.:1.695 3rd Qu.: 2.980 3rd Qu.:2.135   
## Max. :6.000 Max. : 6.930 Max. :9.420   
## D E   
## Min. : 0.040 Min. :0.040   
## 1st Qu.: 0.580 1st Qu.:0.895   
## Median : 1.020 Median :1.950   
## Mean : 3.093 Mean :2.231   
## 3rd Qu.: 1.750 3rd Qu.:3.175   
## Max. :200.300 Max. :7.840

We only eliminated two rows of data where the SEEDED and SEASON variables were missing, while imputing values to three other rows using median.

Now, we can write the resulting matrix to a .csv file.

write.csv(teach.clean, file = "rainfixed.txt")

## 7.

*Using any techniques you've learned, answer this question to a policy maker*... I think we can dive in and explore the average rainfall for Seeded and Unseeded areas both as a group and individually. I want to examine this alternative hypothesis first - "there is a difference in rainfall bewteen seeded and unseeded areas" (null is that that there is no difference).

First, Let's create a new variable that averages the area rainfall for each row Then I can set up my variables Seeded and Unseeded, which is our variable of interest.

teach$avg <- rowMeans(teach[,3:7], na.rm = FALSE, dims = 1)  
teach$avg

## [1] 2.4400 0.9980 1.2960 2.3460 3.8820 2.2060 1.4240 0.5680  
## [9] 2.1800 3.9720 1.1620 1.5860 0.6220 0.8620 1.6500 1.2220  
## [17] 0.2940 0.3520 0.7720 0.4120 0.5460 2.7200 1.4740 1.6840  
## [25] 1.4780 0.9380 0.6560 0.7920 0.8240 1.5240 1.4780 1.4440  
## [33] 1.2300 1.0500 3.4060 0.6540 0.4760 3.0520 1.1280 0.6420  
## [41] 0.7940 1.8680 1.8000 1.7620 0.1260 0.6720 1.0260 2.1020  
## [49] 0.9100 2.3360 0.4580 0.0460 0.6480 0.5140 1.3640 1.2200  
## [57] 40.1040 0.3080 2.2400 2.6980 3.2500 2.0400 2.9860 1.8232  
## [65] 1.5220 1.8120 1.5740 4.9600 0.8840 1.4120 2.2300 1.3600  
## [73] 2.5440 2.3400 2.7260 3.3300 2.4020 1.0600 0.5700 0.5360  
## [81] 0.3080 1.2480 1.5040 5.1320 0.3780 0.6590 2.6880 0.9000  
## [89] 0.3000 3.6060 1.4740 1.2900 0.4360 1.8820 3.6060 2.8180  
## [97] 2.4380 2.5520 2.0180 6.6020 4.0610 0.1520 0.9660 2.1880  
## [105] 1.5800 2.3260 0.0520 1.1680 3.4060

seeded <- subset(teach, SEEDED=="S")  
seeded

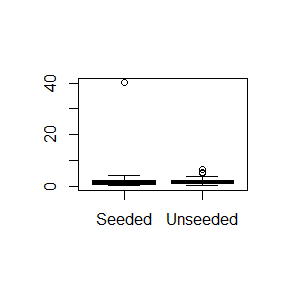
## SEEDED SEASON A B C D E avg  
## 1 S AUTUMN 1.69 3.730 1.650 1.80 3.33 2.440  
## 3 S WINTER 0.81 0.860 2.390 0.36 2.06 1.296  
## 5 S WINTER 2.48 4.610 4.160 2.16 6.00 3.882  
## 8 S WINTER 0.37 0.840 0.260 0.47 0.90 0.568  
## 10 S SPRING 3.38 5.560 2.760 3.10 5.06 3.972  
## 11 S SPRING 0.69 1.460 1.070 0.64 1.95 1.162  
## 13 S SPRING 0.44 1.050 0.240 0.44 0.94 0.622  
## 15 S SUMMER 1.13 2.280 0.970 1.66 2.21 1.650  
## 17 S SUMMER 0.17 0.550 0.130 0.27 0.35 0.294  
## 20 S SUMMER 0.40 0.340 0.450 0.43 0.44 0.412  
## 21 S AUTUMN 0.52 0.790 0.420 0.47 0.53 0.546  
## 24 S AUTUMN 1.62 2.540 0.940 1.59 1.73 1.684  
## 26 S AUTUMN 0.63 1.310 0.760 0.71 1.28 0.938  
## 27 S WINTER 0.42 1.230 0.130 0.59 0.91 0.656  
## 30 S WINTER 0.88 1.320 1.870 0.58 2.97 1.524  
## 32 S WINTER 1.25 1.000 2.040 0.71 2.22 1.444  
## 34 S WINTER 1.11 0.800 1.460 1.48 0.40 1.050  
## 35 S SPRING 3.43 2.550 5.080 1.77 4.20 3.406  
## 37 S SPRING 0.39 0.440 0.490 0.55 0.51 0.476  
## 40 S SPRING 0.39 1.220 0.250 0.46 0.89 0.642  
## 41 S SUMMER 0.86 1.240 0.690 0.49 0.69 0.794  
## 44 S SPRING 1.22 2.000 2.130 1.13 2.33 1.762  
## 45 S SPRING 0.07 0.220 0.020 0.08 0.24 0.126  
## 48 S SPRING 1.67 3.460 1.020 1.89 2.47 2.102  
## 50 S SUMMER 1.79 3.130 1.830 1.82 3.11 2.336  
## 52 S SUMMER 0.00 0.150 0.000 0.04 0.04 0.046  
## 53 S SUMMER 0.44 0.890 0.830 0.38 0.70 0.648  
## 55 S SUMMER 0.96 0.880 2.650 0.85 1.48 1.364  
## 57 S AUTUMN 0.05 0.060 0.010 200.30 0.10 40.104  
## 59 S AUTUMN 1.83 2.930 1.800 1.62 3.02 2.240  
## 61 S AUTUMN 2.50 3.990 2.840 2.44 4.48 3.250  
## 63 S AUTUMN 1.83 3.870 3.010 1.66 4.56 2.986  
## 66 S WINTER 1.09 3.560 0.070 2.26 2.08 1.812  
## 67 S WINTER 0.79 1.430 1.620 1.16 2.87 1.574  
## 70 S WINTER 0.76 1.830 1.500 0.41 2.56 1.412  
## 71 S SPRING 1.53 3.620 1.520 1.62 2.86 2.230  
## 74 S SPRING 1.59 3.190 2.360 1.53 3.03 2.340  
## 76 S SPRING 2.09 5.230 2.120 2.77 4.44 3.330  
## 78 S SUMMER 0.66 2.220 0.210 1.41 0.80 1.060  
## 80 S SUMMER 0.46 1.080 0.010 0.65 0.48 0.536  
## 81 S SUMMER 0.22 0.620 0.150 0.13 0.42 0.308  
## 83 S SUMMER 1.76 1.190 2.260 1.04 1.27 1.504  
## 86 S AUTUMN 0.37 1.555 0.310 0.23 0.83 0.659  
## 87 S AUTUMN 4.97 3.030 1.440 3.14 0.86 2.688  
## 89 S AUTUMN 0.13 0.540 0.110 0.14 0.58 0.300  
## 92 S AUTUMN 0.55 1.130 1.300 1.02 2.45 1.290  
## 93 S WINTER 0.24 0.610 0.050 0.38 0.90 0.436  
## 95 S WINTER 2.35 4.290 4.240 1.67 5.48 3.606  
## 98 S WINTER 1.63 3.310 2.210 2.36 3.25 2.552  
## 99 S WINTER 1.08 3.170 0.800 2.25 2.79 2.018  
## 101 S SPRING 2.67 6.930 1.285 3.03 6.39 4.061  
## 103 S SPRING 0.58 1.410 0.960 0.64 1.24 0.966  
## 105 S SPRING 1.17 1.650 1.220 2.28 1.58 1.580  
## 107 S 0.02 0.080 0.050 0.02 0.09 0.052  
## 109 S SPRING 3.43 2.550 5.080 1.77 4.20 3.406

unseeded <- subset(teach, SEEDED=="U")  
unseeded

## SEEDED SEASON A B C D E avg  
## 2 U AUTUMN 0.74 0.780 1.09 0.79 1.59 0.9980  
## 4 U WINTER 1.44 2.010 2.96 1.27 4.05 2.3460  
## 6 U WINTER 0.84 2.390 2.76 0.87 4.17 2.2060  
## 7 U WINTER 0.37 1.370 1.08 0.85 3.45 1.4240  
## 9 U SPRING 1.33 2.310 2.53 1.08 3.65 2.1800  
## 12 U SPRING 1.42 2.790 1.42 1.08 1.22 1.5860  
## 14 U SPRING 0.76 1.240 0.70 0.67 0.94 0.8620  
## 16 U SUMMER 0.88 1.580 1.06 1.13 1.46 1.2220  
## 18 U SUMMER 0.25 0.770 0.10 0.30 0.34 0.3520  
## 19 U SUMMER 0.78 1.450 0.38 0.58 0.67 0.7720  
## 22 U AUTUMN 2.73 2.090 2.24 4.02 2.52 2.7200  
## 23 U AUTUMN 0.90 2.450 0.52 1.32 2.18 1.4740  
## 25 U AUTUMN 0.93 2.110 1.19 0.85 2.31 1.4780  
## 28 U WINTER 0.64 0.430 1.50 0.24 1.15 0.7920  
## 29 U WINTER 0.30 0.690 1.03 0.22 1.88 0.8240  
## 33 U WINTER 1.08 0.990 1.44 1.00 1.64 1.2300  
## 36 U SPRING 0.54 0.430 0.66 0.73 0.91 0.6540  
## 38 U SPRING 2.53 3.180 3.27 2.68 3.60 3.0520  
## 39 U SPRING 0.81 0.890 1.33 0.43 2.18 1.1280  
## 42 U SUMMER 2.16 2.290 2.12 0.95 1.82 1.8680  
## 43 U SPRING 1.70 2.180 1.45 1.47 2.20 1.8000  
## 46 U SPRING 0.49 1.070 0.36 0.87 0.57 0.6720  
## 47 U SPRING 0.71 1.730 0.72 0.99 0.98 1.0260  
## 49 U SUMMER 0.73 1.510 0.18 1.42 0.71 0.9100  
## 51 U SUMMER 0.19 1.050 0.08 0.40 0.57 0.4580  
## 54 U SUMMER 0.31 1.150 0.01 0.44 0.66 0.5140  
## 56 U SUMMER 1.04 1.200 1.27 1.39 1.20 1.2200  
## 58 U AUTUMN 0.04 0.200 0.35 0.75 0.20 0.3080  
## 60 U AUTUMN 2.24 2.170 4.44 1.05 3.59 2.6980  
## 62 U AUTUMN 1.10 1.710 2.05 1.30 4.04 2.0400  
## 64 U AUTUMN 1.41 -0.034 2.58 1.21 3.95 1.8232  
## 65 U WINTER 0.74 1.360 2.22 0.61 2.68 1.5220  
## 68 U WINTER 4.06 6.710 4.34 3.29 6.40 4.9600  
## 69 U WINTER 0.40 0.640 1.03 0.58 1.77 0.8840  
## 72 U SPRING 0.56 2.880 0.37 1.25 1.74 1.3600  
## 73 U SPRING 1.74 3.450 2.14 1.00 4.39 2.5440  
## 75 U SPRING 1.91 4.740 1.71 2.03 3.24 2.7260  
## 77 U SUMMER 1.59 3.920 1.38 2.11 3.01 2.4020  
## 79 U SUMMER 0.68 0.420 0.48 0.59 0.68 0.5700  
## 82 U SUMMER 1.11 1.700 1.32 0.57 1.54 1.2480  
## 84 U SUMMER 5.12 5.250 5.95 3.97 5.37 5.1320  
## 85 U AUTUMN 0.12 0.600 0.19 0.28 0.70 0.3780  
## 88 U AUTUMN 0.57 1.530 0.30 0.72 1.38 0.9000  
## 90 U AUTUMN 2.47 4.700 3.66 1.84 5.36 3.6060  
## 91 U AUTUMN 1.01 2.320 1.14 0.81 2.09 1.4740  
## 94 U WINTER 2.36 1.150 1.84 1.73 2.33 1.8820  
## 96 U WINTER 2.23 4.300 1.99 1.90 3.67 2.8180  
## 97 U WINTER 1.16 3.060 2.44 1.52 4.01 2.4380  
## 100 U WINTER 6.00 6.150 9.42 3.60 7.84 6.6020  
## 102 U SPRING 0.36 0.150 0.00 0.19 0.06 0.1520  
## 104 U SPRING 1.36 3.430 1.38 1.86 2.91 2.1880  
## 106 U SPRING 2.37 1.940 2.46 2.47 2.39 2.3260  
## 108 U SPRING 0.92 2.090 0.61 0.87 1.35 1.1680

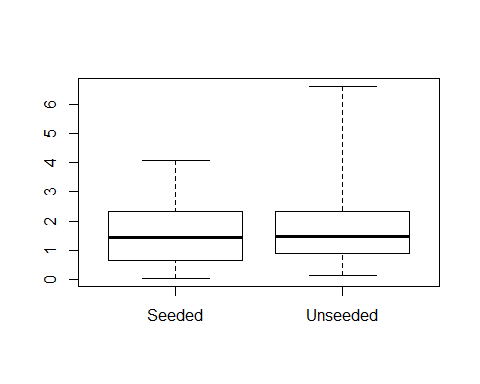
Seeded has 52 observations and Unseeded has 53. This is good, as we have a fair class balance between the two. Let's compare the averages for Seeded A and Unseeded A using a boxplot.

boxplot(seeded$avg, unseeded$avg, names=c("Seeded","Unseeded"))



Looks like there's one outlier in the seeded that could be raising the average of all the data for that category. Let's filter it out and continue on. Since all but one of the values in our variables are below 10, we'll set the filter at less than 10.

seededleq10 <- subset(seeded, avg<10)  
unseededleq10 <- subset(unseeded, avg<10)  
boxplot(seededleq10$avg, unseededleq10$avg, range=0, names=c("Seeded", "Unseeded"))



There appears to be no significant difference in the median rainfall, especially when controlling for outliers, which can greatly skew a statistic of centrality like the mean. Thus, we could not reject the null hypothesis here. I would advise the policy maker that cloud-seeding does not appear to work based on the available data. In fact, unseeded areas seem to have slightly higher rainfall on average.

We could further explore statistical measures such as Welch's t-test, compare the differences in mean, and create p-values, and actually test the null vs. alternative. We would probably get the same result.

We could additionally explore and compare average rainfall between areas, between seasons, etc. to find any interesting trends or correlations. I would use k-means, or random forest.

# Problem 5

## 1.

Assume four pieces of data x1 = (.5; 2000;????100); x2 = (:2; 3000;????200); x3 = (4; 4000;????100); x4 = (:14; 4400;????140). You've been hired to datamine this data using Euclidean distance. How would you preprocess this before datamining and explain why. What are the two closest data points?

Let's create the variables:

x1 <- c(.5, 2000, 100)  
x2 <- c(.2, 300, 200)  
x3 <- c(4, 4000, 100)  
x4 <- c(.14, 4400, 140)

If we are comparing our variables, we need them to be on the same scale. Variable x3 is not, with a value or 4 that will make comparison difficult. The variables need to be combined into a dataframe and normalized first.

df <- data.frame(x1, x2, x3, x4)  
df

## x1 x2 x3 x4  
## 1 5e-01 0.2 4 0.14  
## 2 2e+03 300.0 4000 4400.00  
## 3 1e+02 200.0 100 140.00

df.scale <- scale(df)  
df.scale

## x1 x2 x3 x4  
## [1,] -0.6209393 -1.0909958 -0.5982760 -0.6050868  
## [2,] 1.1535745 0.8730587 1.1544446 1.1542490  
## [3,] -0.5326352 0.2179371 -0.5561686 -0.5491622  
## attr(,"scaled:center")  
## x1 x2 x3 x4   
## 700.1667 166.7333 1368.0000 1513.3800   
## attr(,"scaled:scale")  
## x1 x2 x3 x4   
## 1126.7875 152.6434 2279.8842 2500.8641

We can now measure the Euclidian Distance between the points.

dist(df.scale)

## 1 2  
## 2 3.629559   
## 3 1.313775 3.016669

Thus, x1 and x3 are the closest points. We could compare the unscaled matrix as well.

# Problem 6

You're given a sample of data: 15,2,44,21,40,20,19,18. Calculate the sample mean and sample variance.

x <- c(15,2,44,21,40,20,19,18)  
mean(x)

## [1] 22.375

var(x)

## [1] 183.6964

The sample mean is 22.375 and the sample variance is 183.7.

# Problem 7

Choose all that apply. Which of the following statistical measures can be observed on a box plot?

1. Median
2. Outliers
3. Maximum element
4. Minimum element
5. Variance or covariance (No actual number, but the var or cov is depicted by the distance between the upper and lower whiskers and box edges.)

# Problem 8

Choose all that apply. The most common methods of removing outliers are: (a) Removing tuples with missing values. (c) Observing the probability of existing values in (?).

I have used the following techniques in this paper (from the text):

Remove the cases with unknowns. Fill in the unknown values with the most frequent values. Fill in the unknown values by exploring the correlations between variables. Fill in the unknown values by exploring the similarity between cases.

# Problem 9

Swiss bank data contains various lengths measurements on 200 Swiss bank notes. Load the Swiss bank data as follows:

## install.packages("alr3")  
library("alr3")

## Loading required package: car

head(banknote)

## Length Left Right Bottom Top Diagonal Y  
## 1 214.8 131.0 131.1 9.0 9.7 141.0 0  
## 2 214.6 129.7 129.7 8.1 9.5 141.7 0  
## 3 214.8 129.7 129.7 8.7 9.6 142.2 0  
## 4 214.8 129.7 129.6 7.5 10.4 142.0 0  
## 5 215.0 129.6 129.7 10.4 7.7 141.8 0  
## 6 215.7 130.8 130.5 9.0 10.1 141.4 0

First, summary exploration.

str(banknote)

## 'data.frame': 200 obs. of 7 variables:  
## $ Length : num 215 215 215 215 215 ...  
## $ Left : num 131 130 130 130 130 ...  
## $ Right : num 131 130 130 130 130 ...  
## $ Bottom : num 9 8.1 8.7 7.5 10.4 9 7.9 7.2 8.2 9.2 ...  
## $ Top : num 9.7 9.5 9.6 10.4 7.7 10.1 9.6 10.7 11 10 ...  
## $ Diagonal: num 141 142 142 142 142 ...  
## $ Y : int 0 0 0 0 0 0 0 0 0 0 ...

We have 200 observations of 7 variables. 6 of the variables are numbers, but Y is listed as an integer. Let's explore Y more thoroughly.

banknote$Y

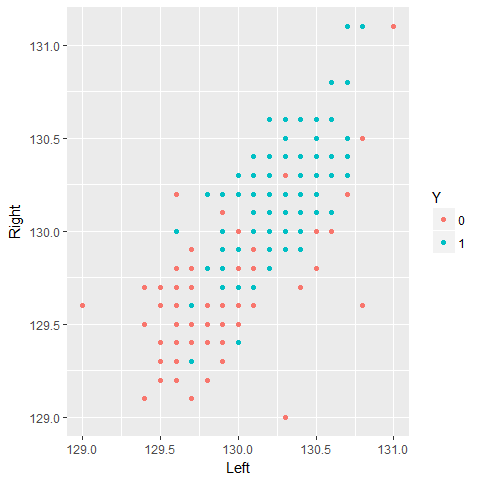
## [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [36] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [71] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1  
## [106] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1  
## [141] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1  
## [176] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

It's listed as an integer, but appears to be a factor, and possibly a class indicator. Let's recode the datatype as a factor. My initial hunch is that the Y variable is denoting whether the note is counterfeit or not, as the remaining variables are all measurements of a physical note.

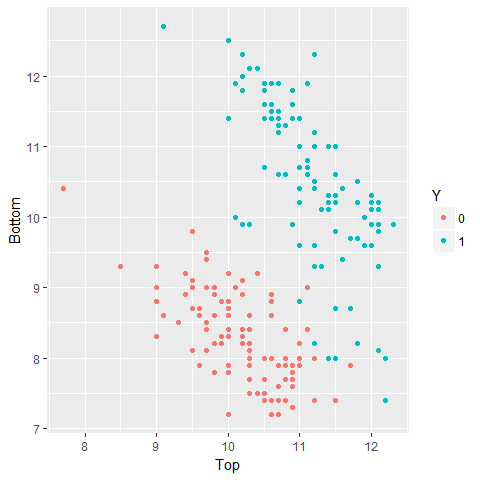
banknote$Y <- as.factor(banknote$Y)  
str(banknote)

## 'data.frame': 200 obs. of 7 variables:  
## $ Length : num 215 215 215 215 215 ...  
## $ Left : num 131 130 130 130 130 ...  
## $ Right : num 131 130 130 130 130 ...  
## $ Bottom : num 9 8.1 8.7 7.5 10.4 9 7.9 7.2 8.2 9.2 ...  
## $ Top : num 9.7 9.5 9.6 10.4 7.7 10.1 9.6 10.7 11 10 ...  
## $ Diagonal: num 141 142 142 142 142 ...  
## $ Y : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...

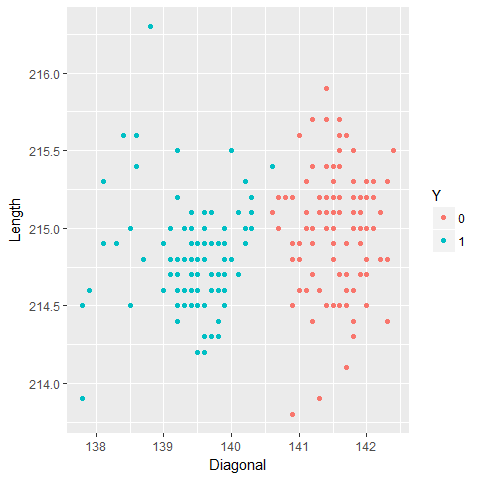
qplot(Left, Right, data=banknote, color=Y)



qplot(Top, Bottom, data=banknote, color=Y)



qplot(Diagonal, Length, data=banknote, color=Y)



There's a pretty clear pattern here in the relationship between Length, Diagonal, and Y variables. This pattern looks suitable for k-means clustering. We could further explore the relationships between the variables, but the plot above shows strong evidence that the Length and Diagonal variables conditioned by Y will make a decent plot.

bn3 <- kmeans(banknote[,c(1,6)], centers=3, iter.max=200)  
bn3

## K-means clustering with 3 clusters of sizes 13, 100, 87  
##   
## Cluster means:  
## Length Diagonal  
## 1 215.0231 138.3154  
## 2 214.9740 141.5270  
## 3 214.7874 139.6080  
##   
## Clustering vector:  
## [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2  
## [36] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3  
## [71] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3  
## [106] 3 3 3 3 3 3 3 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 3 3  
## [141] 3 3 3 3 3 3 3 1 3 3 3 3 3 3 3 3 3 3 3 1 1 1 3 3 3 3 1 1 3 3 1 3 3 3 3  
## [176] 3 3 3 3 1 3 1 3 3 3 3 1 3 3 3 3 1 3 1 3 3 3 3 3 3  
##   
## Within cluster sum of squares by cluster:  
## [1] 6.00000 31.98950 15.92046  
## (between\_SS / total\_SS = 81.6 %)  
##   
## Available components:  
##   
## [1] "cluster" "centers" "totss" "withinss"   
## [5] "tot.withinss" "betweenss" "size" "iter"   
## [9] "ifault"

bn3$tot.withinss

## [1] 53.90996

bn3$withinss

## [1] 6.00000 31.98950 15.92046

table(bn3$cluster, banknote$Y)

##   
## 0 1  
## 1 0 13  
## 2 99 1  
## 3 1 86

We have pretty good performance with three clusters. Only two notes were mis-classified. A Type 1 error is the more egregious error, as we wouldn't want any counterfeit notes to be passed as legitimate notes. Let's experiment with two and four clusters and compare.

bn2 <- kmeans(banknote[,c(1,6)], centers=2, iter.max=200)  
bn2$tot.withinss

## [1] 73.4371

table(bn2$cluster, banknote$Y)

##   
## 0 1  
## 1 99 1  
## 2 1 99

Again, we get decent performance and misclassify only two instances, but our total sum of square errors is higher. Let's look at our 4 clusters and move on after that.

bn4 <- kmeans(banknote[,c(1,6)], centers=4, iter.max = 200)  
table(bn4$cluster, banknote$Y)

##   
## 0 1  
## 1 4 29  
## 2 0 13  
## 3 1 58  
## 4 95 0

bn3$tot.withinss

## [1] 53.90996

The lowest Total SS we've seen so far, but we're not getting any increased performance on classifying our two instances. Other methods we could use would involve splitting the data into train and test sets, possibly scaling the data, etc. Neural nets, random forests, etc... might give better results.

# 10. Bonus question

## install.packages("CORElearn")  
library(CORElearn)  
  
LocData <- read.csv("C:\\Users\\khickman\\Desktop\\Personal\\IUMSDS\\AppliedDataMining\\Midterm\\entropy.csv", header = TRUE)  
LocData$User <- as.factor(LocData$User)  
LocData

## User Location Clicks  
## 1 1 UL 3  
## 2 1 LR 1  
## 3 1 M 2  
## 4 1 LL 0  
## 5 1 UR 0  
## 6 2 UL 1  
## 7 2 LR 1  
## 8 2 M 2  
## 9 2 LL 0  
## 10 2 UR 0  
## 11 3 UL 0  
## 12 3 LR 0  
## 13 3 M 2  
## 14 3 LL 1  
## 15 3 UR 1

attrEval(Clicks ~., LocData, estimator = "GainRatio")

## Changing dependent variable to factor with levels: 0 1 2 3

## Warning in attrEval(Clicks ~ ., LocData, estimator = "GainRatio"): Possibly  
## this is an error caused by regression formula and classification attribute  
## estimator or vice versa.

## User Location   
## 0.07992343 0.39362419

attrEval(Clicks ~., LocData, estimator = "Gini")

## Changing dependent variable to factor with levels: 0 1 2 3

## Warning in attrEval(Clicks ~ ., LocData, estimator = "Gini"): Possibly  
## this is an error caused by regression formula and classification attribute  
## estimator or vice versa.

## User Location   
## 0.01777778 0.28444444

attrEval(Clicks ~., LocData, estimator = "InfGain")

## Changing dependent variable to factor with levels: 0 1 2 3

## Warning in attrEval(Clicks ~ ., LocData, estimator = "InfGain"): Possibly  
## this is an error caused by regression formula and classification attribute  
## estimator or vice versa.

## User Location   
## 0.1266756 0.9139671

attrEval(Clicks ~., LocData, estimator = "MDL")

## Changing dependent variable to factor with levels: 0 1 2 3

## Warning in attrEval(Clicks ~ ., LocData, estimator = "MDL"): Possibly  
## this is an error caused by regression formula and classification attribute  
## estimator or vice versa.

## User Location   
## -0.1469251 0.1326246

The Information Gain estimator indicates that 91% of the click variable can be explained by the location variable. All four metrics agreed that the location is relatively important.