COEN 281, HW4

Yutong Li

1.

library(gdata)

## gdata: read.xls support for 'XLS' (Excel 97-2004) files ENABLED.

##

## gdata: read.xls support for 'XLSX' (Excel 2007+) files ENABLED.

##   
## Attaching package: 'gdata'

## The following object is masked from 'package:stats':  
##   
## nobs

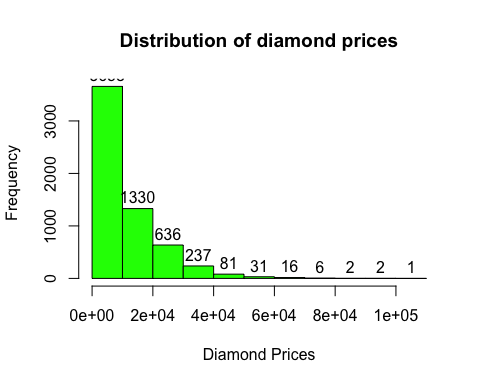
## The following object is masked from 'package:utils':  
##   
## object.size

## The following object is masked from 'package:base':  
##   
## startsWith

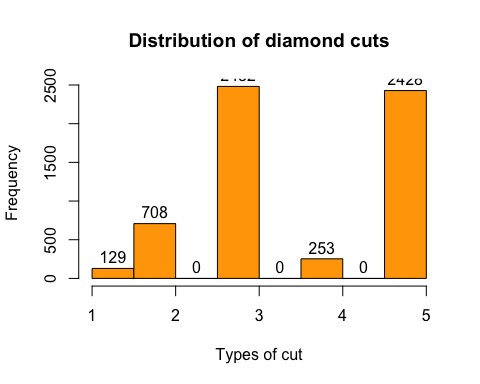
library(pre)  
set.seed(123)  
data <- read.xls("Diamond\_data/Diamond\_Data.xls", sheet="Raw Data", perl="/usr/bin/perl")

(a)

hist(data$Price, col = "green", xlab = "Diamond Prices", ylab = "Frequency", main = "Distribution of diamond prices", labels = TRUE)



hist(as.numeric(data$Cut), col = "orange", xlab = "Types of cut", ylab = "Frequency", main = "Distribution of diamond cuts", labels = TRUE)



(b)

train\_in = sample(1:nrow(data), 5000, replace = FALSE)  
train\_data = data[train\_in,]  
test\_data = data[-train\_in,]  
  
regression\_model = pre(Price ~., data = train\_data, family = "gaussian", nfolds = 10)  
  
summary(regression\_model)

##   
## Final ensemble with cv error within 1se of minimum:   
## lambda = 4.796746  
## number of terms = 238  
## mean cv error (se) = 1230678 (95149.05)  
##   
## cv error type : Mean-Squared Error

library(Metrics)

##   
## Attaching package: 'Metrics'

## The following object is masked from 'package:gdata':  
##   
## ll

pred\_train = predict(regression\_model, newdata = train\_data)  
rmse(train\_data$Price, pred\_train)

## [1] 939.7673

(c)

print(regression\_model$rules[1:10, ])

## rule  
## rule1 rule1  
## rule2 rule2  
## rule3 rule3  
## rule4 rule4  
## rule5 rule5  
## rule6 rule6  
## rule7 rule7  
## rule8 rule8  
## rule9 rule9  
## rule10 rule10  
## description  
## rule1 Carat.Weight <= 1.71  
## rule2 Carat.Weight <= 1.71 & Carat.Weight <= 1.23  
## rule3 Carat.Weight <= 1.71 & Carat.Weight <= 1.23 & Carat.Weight <= 0.93  
## rule4 Carat.Weight <= 1.71 & Carat.Weight <= 1.23 & Carat.Weight > 0.93  
## rule5 Carat.Weight <= 1.71 & Carat.Weight > 1.23  
## rule6 Carat.Weight <= 1.71 & Carat.Weight > 1.23 & Color %in% c("D", "E", "F", "G")  
## rule7 Carat.Weight <= 1.71 & Carat.Weight > 1.23 & Color %in% c("H", "I")  
## rule8 Carat.Weight > 1.71 & Clarity %in% c("FL", "IF", "VVS1", "VVS2")  
## rule9 Carat.Weight > 1.71 & Clarity %in% c("FL", "IF", "VVS1", "VVS2") & Color %in% c("D", "E", "F")  
## rule10 Carat.Weight > 1.71 & Clarity %in% c("FL", "IF", "VVS1", "VVS2") & Color %in% c("G", "H", "I")

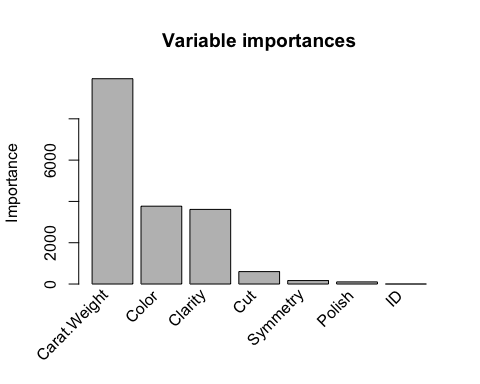
coef(regression\_model)[1:10,]

## rule coefficient  
## 3698 Carat.Weight 8600.643  
## 1740 rule2187 3919.619  
## 614 rule776 2761.764  
## 3696 (Intercept) 2715.202  
## 796 rule1019 2359.231  
## 1686 rule2129 2205.546  
## 2169 rule2699 2163.425  
## 1491 rule1903 2037.291  
## 211 rule250 1806.588  
## 2656 rule3274 1717.079  
## description  
## 3698 0.77 <= Carat.Weight <= 2.37  
## 1740 Clarity %in% c("FL", "IF") & Color %in% c("D")  
## 614 Carat.Weight > 1.95 & Clarity %in% c("FL", "IF", "VVS1", "VVS2") & Color %in% c("D", "E", "F")  
## 3696 1  
## 796 Carat.Weight > 1.61 & Clarity %in% c("FL", "IF", "VVS1", "VVS2") & Carat.Weight > 2.48  
## 1686 Carat.Weight > 2.04 & Clarity %in% c("FL", "IF", "VVS1", "VVS2") & Color %in% c("D", "E")  
## 2169 Carat.Weight > 2.24 & Color %in% c("D", "E", "F", "G") & Carat.Weight > 2.6  
## 1491 Carat.Weight > 2.38 & Color %in% c("D")  
## 211 Carat.Weight > 1.76 & Clarity %in% c("FL", "IF", "VVS1", "VVS2") & Color %in% c("D")  
## 2656 Carat.Weight > 2.39 & Carat.Weight > 2.7

# Yes, Carat Weight is almost linear. It's coefficient is 8604.378  
# rule2063: if the Clarith is 'FL' or 'IF' and Color is D, the price will add to 8604.378

(d)

importance(regression\_model)



# Carat Weight, Clarity and Color

(e)

pred\_test = predict(regression\_model, newdata = test\_data)  
mae(test\_data$Price, pred\_test)

## [1] 634.4471

(f)

library(rpart)  
train\_data[,3] <- as.factor(train\_data[,3])  
train\_data[,4] <- as.factor(train\_data[,4])  
train\_data[,5] <- as.factor(train\_data[,5])  
train\_data[,6] <- as.factor(train\_data[,6])  
train\_data[,7] <- as.factor(train\_data[,7])  
train\_data[,8] <- as.factor(train\_data[,8])  
  
test\_data[,3] <- as.factor(test\_data[,3])  
test\_data[,4] <- as.factor(test\_data[,4])  
test\_data[,5] <- as.factor(test\_data[,5])  
test\_data[,6] <- as.factor(test\_data[,6])  
test\_data[,7] <- as.factor(test\_data[,7])  
test\_data[,8] <- as.factor(test\_data[,8])  
  
diamond\_tree <- rpart(Price~., method = "anova", data = train\_data, cp=0.0001)  
  
cp\_value <- diamond\_tree$cptable  
  
diamond\_pruned <- prune(diamond\_tree, cp = 0.0001)  
  
pred\_test <- predict(diamond\_pruned, test\_data, type = "vector")  
# mean absolute error  
mae(test\_data$Price, pred\_test)

## [1] 1141.227

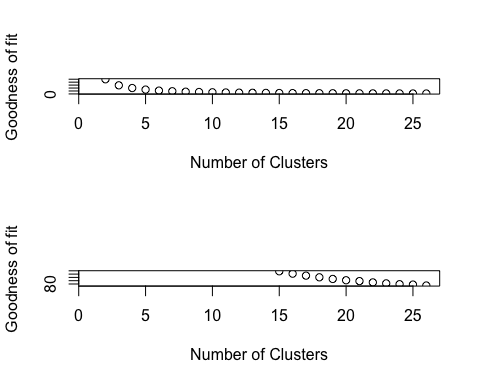
2.

(a)

char\_data = read.table("az-5000.txt" , TRUE)  
new\_char\_data <- char\_data[,-1]  
  
fit <- vector()  
  
for (i in 2:26)  
{   
 kmeans\_out <- kmeans(new\_char\_data, centers = i, iter.max = 26)  
 fit[i] <- (1/i)\*sum(kmeans(new\_char\_data, centers = i)$withinss)  
}

(b)

par(mfrow = c(2,1))  
plot(1:26, fit, type = "b", xlab = "Number of Clusters", ylab = "Goodness of fit")  
  
fit2 <- vector()  
for (i in 15:26)  
{  
 fit2[i] <- (1/i)\*sum(kmeans(new\_char\_data, centers = i)$withinss)  
}  
  
plot(1:26, fit2, type = "b", xlab = "Number of Clusters", ylab = "Goodness of fit")

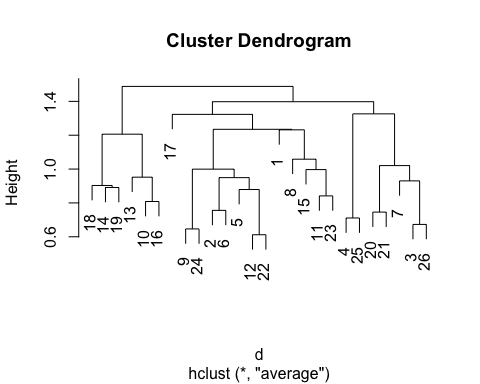


## There is a "step" at 23 cluster. The number of "natural" cluster is 23.

3.

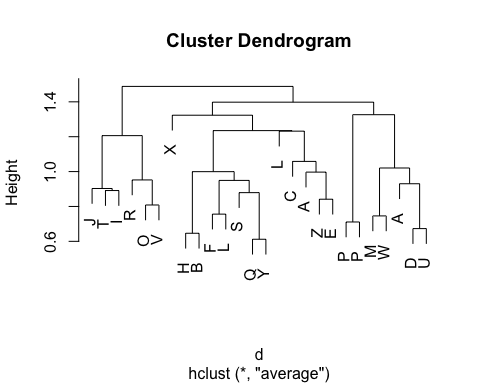
(a)

d <- dist(kmeans\_out$centers, method = "euclidean")  
fit <- hclust(d, method = "average")  
plot(fit)



(b)

letter\_mat <- char\_data[,1]  
num\_cluster <- kmeans\_out$cluster  
initial\_mat <- matrix(0,26,26)  
rownames(initial\_mat) <- LETTERS  
  
for(k in 1:5000)  
{  
 initial\_mat[letter\_mat[k], num\_cluster[k]] <- initial\_mat[letter\_mat[k], num\_cluster[k]] + 1  
}  
  
common\_letter <- c()  
for(i in 1:26)  
{  
 common\_letter[i] <- which.max(initial\_mat[,i])  
}  
  
plot(fit, labels = LETTERS[common\_letter])



(c)

1. Some letters appear more than once.

2. At height = 1.0, there are 8 clusters. Below height = 1.0 and above height = 0.8, there are 18 clusters. The number doubled.

(d)

common\_cluster <- data.frame(c(0) \*26)  
for(i in 1:26)  
{  
 common\_cluster[i] <- which.max(initial\_mat[i,])  
}  
colnames(common\_cluster) = LETTERS  
common\_cluster[c('D', 'G', 'K', 'N', 'Y', 'Z')]

## D G K N Y Z  
## 1 3 22 9 26 22 11

4.

(a)

library(arules)

## Loading required package: Matrix

##   
## Attaching package: 'arules'

## The following objects are masked from 'package:base':  
##   
## abbreviate, write

movies\_data <- read.transactions("Movies\_data/ratingsAsBasket.txt", format = "basket", sep = NULL)  
summary(movies\_data)

## transactions as itemMatrix in sparse format with  
## 10000 rows (elements/itemsets/transactions) and  
## 15500 columns (items) and a density of 0.009911529   
##   
## most frequent items:  
## M.4712.R.High M.3749.R.High M.5407.R.High M.4275.R.High M.538.R.High   
## 4729 4610 4162 4152 4010   
## (Other)   
## 1514624   
##   
## element (itemset/transaction) length distribution:  
## sizes  
## 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34   
## 64 110 77 71 81 71 77 100 96 85 108 112 99 100 110   
## 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49   
## 93 83 84 95 115 80 110 80 99 83 84 72 69 82 66   
## 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64   
## 88 71 66 60 67 71 76 61 81 76 60 58 55 74 48   
## 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79   
## 57 68 54 48 44 48 56 60 50 63 43 56 56 52 53   
## 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94   
## 44 51 43 36 34 64 34 41 42 43 39 46 27 42 40   
## 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109   
## 36 42 38 31 37 36 36 40 41 36 42 38 36 35 40   
## 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124   
## 33 38 37 34 32 35 34 30 41 32 38 33 39 24 29   
## 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139   
## 20 37 31 32 20 40 20 33 25 26 24 26 30 24 29   
## 140 141 142 143 144 145 146 147 148 149 150 151 152 153 154   
## 26 18 22 30 24 19 28 23 11 24 26 27 27 20 35   
## 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169   
## 21 27 21 19 23 21 11 18 24 24 22 15 29 13 17   
## 170 171 172 173 174 175 176 177 178 179 180 181 182 183 184   
## 26 16 20 16 10 13 22 23 16 19 15 14 10 24 20   
## 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199   
## 21 16 17 15 12 14 16 15 13 13 13 19 15 12 7   
## 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214   
## 14 24 13 15 9 11 17 14 12 13 19 12 10 14 9   
## 215 216 217 218 219 220 221 222 223 224 225 226 227 228 229   
## 5 14 16 12 15 17 16 12 14 8 13 11 14 7 14   
## 230 231 232 233 234 235 236 237 238 239 240 241 242 243 244   
## 11 15 18 12 8 15 12 16 8 2 7 7 10 13 16   
## 245 246 247 248 249 250 251 252 253 254 255 256 257 258 259   
## 15 8 11 10 7 8 12 13 9 11 9 6 9 15 11   
## 260 261 262 263 264 265 266 267 268 269 270 271 272 273 274   
## 11 11 9 5 3 9 9 7 8 8 5 5 9 7 2   
## 275 276 277 278 279 280 281 282 283 284 285 286 287 288 289   
## 4 12 6 13 8 6 6 6 8 5 5 6 10 12 10   
## 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304   
## 4 9 11 4 5 9 7 11 2 4 5 6 8 3 5   
## 305 306 307 308 309 310 311 312 313 314 315 316 317 318 319   
## 8 8 8 8 4 10 5 4 10 6 8 5 6 3 6   
## 320 321 322 323 324 325 326 327 328 329 330 331 332 333 334   
## 6 7 5 5 6 5 3 3 5 4 7 6 9 5 2   
## 335 336 337 338 339 340 341 342 343 344 345 346 347 348 349   
## 11 1 5 5 6 5 4 5 4 3 6 6 4 5 3   
## 350 351 352 353 354 355 356 357 358 359 360 361 362 363 364   
## 8 3 5 5 8 6 4 3 5 8 4 3 1 3 5   
## 365 366 367 368 369 370 371 372 373 374 375 376 377 378 379   
## 8 1 2 6 5 9 5 2 5 8 5 3 8 3 1   
## 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394   
## 5 7 4 2 1 4 6 4 4 5 3 3 1 4 2   
## 396 397 398 399 400 401 402 403 404 405 406 407 408 409 410   
## 5 4 3 5 3 8 3 3 6 3 6 4 2 4 4   
## 411 412 413 414 415 416 417 419 420 421 422 423 425 426 427   
## 4 6 4 6 2 3 3 1 3 5 3 2 5 2 5   
## 428 429 430 431 432 434 435 436 437 438 439 440 441 442 443   
## 4 1 1 2 5 3 3 3 4 4 5 1 5 4 4   
## 445 446 448 450 451 453 454 455 456 457 458 459 460 461 462   
## 3 3 4 2 1 3 4 3 3 2 1 4 3 1 1   
## 464 466 467 468 469 470 471 472 473 474 475 476 477 478 479   
## 2 1 4 2 1 5 2 2 1 3 6 1 3 3 6   
## 480 481 482 483 484 485 486 488 489 490 491 492 493 494 496   
## 5 3 3 1 3 2 2 1 1 2 3 3 1 1 2   
## 497 498 500 501 502 503 505 506 507 508 509 510 511 512 514   
## 1 2 1 5 1 1 3 1 4 4 5 3 1 3 3   
## 515 516 517 518 519 520 521 522 524 525 526 527 528 529 530   
## 3 2 4 2 4 1 3 1 2 1 4 3 2 1 1   
## 531 532 534 535 536 537 538 540 542 543 544 545 546 547 548   
## 2 1 1 1 2 6 1 1 2 3 1 2 3 1 3   
## 552 553 554 555 556 557 560 561 562 563 564 565 567 568 572   
## 4 1 3 5 2 2 2 1 1 1 3 2 1 2 5   
## 574 575 577 578 579 580 582 583 586 588 589 590 592 593 595   
## 7 1 1 1 3 3 2 4 1 1 1 2 1 1 3   
## 597 599 600 602 603 604 606 608 609 610 611 612 613 614 615   
## 1 1 2 2 2 1 1 3 1 2 1 1 1 2 2   
## 616 617 618 620 621 622 624 625 628 629 630 632 634 635 637   
## 2 2 3 2 1 1 1 2 1 3 3 1 1 1 1   
## 638 639 640 641 642 643 644 645 646 647 650 652 653 655 657   
## 3 3 1 1 2 1 1 1 2 2 1 1 2 2 2   
## 658 659 660 661 662 663 665 667 668 669 672 675 676 677 679   
## 1 2 3 1 1 1 1 1 2 1 1 1 1 2 3   
## 681 686 687 692 694 695 697 698 701 703 705 707 708 709 712   
## 1 1 1 1 1 1 2 1 2 1 1 3 1 1 3   
## 715 716 718 721 722 723 727 728 729 730 731 732 733 734 736   
## 1 1 1 5 1 1 1 3 1 1 1 1 1 1 1   
## 737 738 741 742 745 746 750 751 753 754 755 758 759 761 762   
## 1 3 1 1 1 2 1 1 1 1 1 1 3 3 2   
## 763 764 766 769 771 774 775 776 783 787 788 789 792 793 794   
## 2 2 1 1 1 1 1 2 1 2 1 2 1 1 2   
## 797 806 810 811 815 818 819 820 821 823 830 832 834 835 837   
## 1 2 2 1 1 2 1 1 1 1 1 1 1 1 1   
## 838 840 842 843 845 847 849 853 855 857 860 863 864 871 878   
## 1 1 1 1 1 1 2 1 1 1 1 2 1 1 2   
## 885 892 896 901 906 907 912 923 924 925 927 932 934 937 946   
## 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1   
## 952 953 962 964 982 985 986 988 994 995 999 1000 1002 1006 1017   
## 1 1 1 4 1 1 2 1 1 1 1 1 1 1 3   
## 1018 1019 1023 1025 1028 1040 1042 1050 1056 1058 1061 1062 1068 1082 1083   
## 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1   
## 1086 1097 1101 1105 1117 1120 1121 1130 1132 1156 1162 1163 1164 1176 1179   
## 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1   
## 1181 1182 1186 1207 1208 1209 1212 1216 1219 1225 1230 1245 1255 1272 1285   
## 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1   
## 1292 1306 1315 1325 1339 1359 1366 1392 1443 1482 1493 1513 1539 1558 1565   
## 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1   
## 1648 1653 1658 1666 1709 1852 1945 1972 2003 2027 2087 2106 2267 2289   
## 1 1 1 1 1 1 1 1 1 1 1 1 1 1   
##   
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 20.0 47.0 92.0 153.6 183.0 2289.0   
##   
## includes extended item information - examples:  
## labels  
## 1 M.1.R.High  
## 2 M.1.R.Low  
## 3 M.1.R.Med

# number of buskets = 10000  
# most frequent item: M.4712 is rated high  
# Minimum/Maximum/Average number of movies rated by one rater: 20/2289/153.6

(b)

rules <- apriori(movies\_data, parameter = list(supp = 0.1, conf = 0.7, target = "rules"))

## Apriori  
##   
## Parameter specification:  
## confidence minval smax arem aval originalSupport maxtime support minlen  
## 0.7 0.1 1 none FALSE TRUE 5 0.1 1  
## maxlen target ext  
## 10 rules FALSE  
##   
## Algorithmic control:  
## filter tree heap memopt load sort verbose  
## 0.1 TRUE TRUE FALSE TRUE 2 TRUE  
##   
## Absolute minimum support count: 1000   
##   
## set item appearances ...[0 item(s)] done [0.00s].  
## set transactions ...[15500 item(s), 10000 transaction(s)] done [0.60s].  
## sorting and recoding items ... [253 item(s)] done [0.01s].  
## creating transaction tree ... done [0.01s].  
## checking subsets of size 1 2 3 4 5 6 done [0.08s].  
## writing ... [2318 rule(s)] done [0.00s].  
## creating S4 object ... done [0.01s].

top\_ten <- head(rules, 10)  
inspect(top\_ten)

## lhs rhs support confidence lift count  
## [1] {M.1212.R.High} => {M.4712.R.High} 0.1013 0.7303533 1.544414 1013   
## [2] {M.265.R.High} => {M.3749.R.High} 0.1442 0.7166998 1.554663 1442   
## [3] {M.4021.R.High} => {M.1870.R.High} 0.1007 0.7249820 1.848501 1007   
## [4] {M.4021.R.High} => {M.4275.R.High} 0.1096 0.7890569 1.900426 1096   
## [5] {M.3670.R.High} => {M.5407.R.High} 0.1008 0.7093596 1.704372 1008   
## [6] {M.3670.R.High} => {M.1870.R.High} 0.1121 0.7888811 2.011425 1121   
## [7] {M.3670.R.High} => {M.4275.R.High} 0.1062 0.7473610 1.800002 1062   
## [8] {M.4230.R.High} => {M.4712.R.High} 0.1143 0.7326923 1.549360 1143   
## [9] {M.3816.R.High} => {M.3749.R.High} 0.1230 0.8698727 1.886926 1230   
## [10] {M.3816.R.High} => {M.4275.R.High} 0.1039 0.7347949 1.769737 1039

# User who rates "Tomorrow Never Dies" high also rates "The Matrix" high. One possible reason is that they are scientific fiction.

(c)

subset\_rules <- subset(rules, subset = lift > 3)  
inspect(subset\_rules)

## lhs rhs support confidence lift count  
## [1] {M.1814.R.High} => {M.1817.R.High} 0.1379 0.7182292 3.103843 1379  
## [2] {M.1814.R.High,   
## M.4275.R.High} => {M.1817.R.High} 0.1104 0.7645429 3.303988 1104  
## [3] {M.1814.R.High,   
## M.4712.R.High} => {M.1817.R.High} 0.1071 0.7526353 3.252529 1071  
## [4] {M.1817.R.High,   
## M.647.R.High} => {M.646.R.High} 0.1026 0.8234350 3.057687 1026  
## [5] {M.1817.R.High,   
## M.646.R.High} => {M.647.R.High} 0.1026 0.7349570 3.068714 1026  
## [6] {M.2936.R.High,   
## M.647.R.High} => {M.646.R.High} 0.1164 0.8185654 3.039604 1164  
## [7] {M.2250.R.High,   
## M.646.R.High} => {M.647.R.High} 0.1407 0.7252577 3.028216 1407  
## [8] {M.2250.R.High,   
## M.2936.R.High,   
## M.647.R.High} => {M.646.R.High} 0.1025 0.8464079 3.142993 1025  
## [9] {M.2250.R.High,   
## M.2936.R.High,   
## M.646.R.High} => {M.647.R.High} 0.1025 0.7465404 3.117079 1025  
## [10] {M.2250.R.High,   
## M.2749.R.High,   
## M.647.R.High} => {M.646.R.High} 0.1006 0.8293487 3.079646 1006  
## [11] {M.2250.R.High,   
## M.2749.R.High,   
## M.646.R.High} => {M.647.R.High} 0.1006 0.7558227 3.155836 1006  
## [12] {M.2526.R.High,   
## M.2749.R.High,   
## M.647.R.High} => {M.646.R.High} 0.1007 0.8440905 3.134387 1007  
## [13] {M.2526.R.High,   
## M.2749.R.High,   
## M.646.R.High} => {M.647.R.High} 0.1007 0.7453738 3.112208 1007  
## [14] {M.2250.R.High,   
## M.2526.R.High,   
## M.647.R.High} => {M.646.R.High} 0.1158 0.8324946 3.091328 1158  
## [15] {M.2250.R.High,   
## M.2526.R.High,   
## M.646.R.High} => {M.647.R.High} 0.1158 0.7509728 3.135586 1158  
## [16] {M.2250.R.High,   
## M.5407.R.High,   
## M.647.R.High} => {M.646.R.High} 0.1038 0.8166798 3.032602 1038  
## [17] {M.2250.R.High,   
## M.5407.R.High,   
## M.646.R.High} => {M.647.R.High} 0.1038 0.7424893 3.100164 1038  
## [18] {M.1870.R.High,   
## M.2250.R.High,   
## M.647.R.High} => {M.646.R.High} 0.1084 0.8181132 3.037925 1084  
## [19] {M.1870.R.High,   
## M.2250.R.High,   
## M.646.R.High} => {M.647.R.High} 0.1084 0.7475862 3.121446 1084  
## [20] {M.2250.R.High,   
## M.4275.R.High,   
## M.647.R.High} => {M.646.R.High} 0.1157 0.8390138 3.115536 1157  
## [21] {M.2250.R.High,   
## M.4275.R.High,   
## M.646.R.High} => {M.647.R.High} 0.1157 0.7392971 3.086836 1157  
## [22] {M.2250.R.High,   
## M.4712.R.High,   
## M.647.R.High} => {M.646.R.High} 0.1130 0.8242159 3.060586 1130  
## [23] {M.2250.R.High,   
## M.4712.R.High,   
## M.646.R.High} => {M.647.R.High} 0.1130 0.7558528 3.155962 1130  
## [24] {M.2526.R.High,   
## M.5407.R.High,   
## M.647.R.High} => {M.646.R.High} 0.1012 0.8214286 3.050236 1012  
## [25] {M.2526.R.High,   
## M.5407.R.High,   
## M.646.R.High} => {M.647.R.High} 0.1012 0.7397661 3.088794 1012  
## [26] {M.1870.R.High,   
## M.2526.R.High,   
## M.647.R.High} => {M.646.R.High} 0.1072 0.8195719 3.043341 1072  
## [27] {M.1870.R.High,   
## M.2526.R.High,   
## M.646.R.High} => {M.647.R.High} 0.1072 0.7253045 3.028411 1072  
## [28] {M.2526.R.High,   
## M.4275.R.High,   
## M.647.R.High} => {M.646.R.High} 0.1119 0.8369484 3.107866 1119  
## [29] {M.2526.R.High,   
## M.4275.R.High,   
## M.646.R.High} => {M.647.R.High} 0.1119 0.7205409 3.008521 1119  
## [30] {M.2526.R.High,   
## M.4712.R.High,   
## M.647.R.High} => {M.646.R.High} 0.1075 0.8231240 3.056532 1075  
## [31] {M.2526.R.High,   
## M.4712.R.High,   
## M.646.R.High} => {M.647.R.High} 0.1075 0.7460097 3.114863 1075  
## [32] {M.1870.R.High,   
## M.5407.R.High,   
## M.646.R.High} => {M.647.R.High} 0.1030 0.7192737 3.003231 1030  
## [33] {M.4275.R.High,   
## M.5407.R.High,   
## M.647.R.High} => {M.646.R.High} 0.1066 0.8149847 3.026308 1066  
## [34] {M.4712.R.High,   
## M.5407.R.High,   
## M.646.R.High} => {M.647.R.High} 0.1037 0.7282303 3.040628 1037  
## [35] {M.1870.R.High,   
## M.4275.R.High,   
## M.647.R.High} => {M.646.R.High} 0.1085 0.8238421 3.059198 1085  
## [36] {M.1870.R.High,   
## M.4712.R.High,   
## M.646.R.High} => {M.647.R.High} 0.1034 0.7364672 3.075020 1034  
## [37] {M.4275.R.High,   
## M.4712.R.High,   
## M.647.R.High} => {M.646.R.High} 0.1112 0.8261516 3.067774 1112  
## [38] {M.4275.R.High,   
## M.4712.R.High,   
## M.646.R.High} => {M.647.R.High} 0.1112 0.7202073 3.007128 1112

summary(subset\_rules)

## set of 38 rules  
##   
## rule length distribution (lhs + rhs):sizes  
## 2 3 4   
## 1 6 31   
##   
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 2.000 4.000 4.000 3.789 4.000 4.000   
##   
## summary of quality measures:  
## support confidence lift count   
## Min. :0.1006 Min. :0.7182 Min. :3.003 Min. :1006   
## 1st Qu.:0.1027 1st Qu.:0.7394 1st Qu.:3.041 1st Qu.:1027   
## Median :0.1074 Median :0.7558 Median :3.077 Median :1074   
## Mean :0.1090 Mean :0.7783 Mean :3.087 Mean :1090   
## 3rd Qu.:0.1119 3rd Qu.:0.8234 3rd Qu.:3.115 3rd Qu.:1119   
## Max. :0.1407 Max. :0.8464 Max. :3.304 Max. :1407   
##   
## mining info:  
## data ntransactions support confidence  
## movies\_data 10000 0.1 0.7

# Life is the importance of a rule  
# User who rates "Die Hard", "Lethal Weapon", and "Paper" high also rates "Papillon" high.

5.

(a)

data = scan("Movies\_data/ratings.txt", what = list(integer(), integer(), integer()), sep = "|")  
library('Matrix')  
sparse\_mat = sparseMatrix(i = data[[1]], j = data[[2]], x = data[[3]])  
dim(sparse\_mat)

## [1] 10000 7223

columns = c(seq(1, dim(sparse\_mat)[2]))  
rows = c(seq(1, dim(sparse\_mat)[1]))  
dimnames(sparse\_mat) = list(rows, columns)

(b)

library(recommenderlab)

## Loading required package: proxy

##   
## Attaching package: 'proxy'

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

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

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

## Loading required package: registry

rating\_data = new("realRatingMatrix", data = sparse\_mat)  
ubcf = Recommender(rating\_data, "UBCF")  
prediction = predict(ubcf, rating\_data[10000], n = 5)  
as(prediction, "list")

## $`10000`  
## [1] "5407" "4712" "4716" "1401" "538"

(c)

prediction\_2 = predict(ubcf, rating\_data[500,], n = 1)  
as(prediction\_2, "list")

## $`500`  
## [1] "4716"

6.

(a)

library(tm)

## Loading required package: NLP

##   
## Attaching package: 'tm'

## The following object is masked from 'package:arules':  
##   
## inspect

news\_corpus = Corpus(DirSource(c("Newsgroup\_data/rec.autos", "Newsgroup\_data/rec.motorcycles")), readerControl = list(reader = readPlain))  
length(news\_corpus)

## [1] 1986

which(names(news\_corpus) == "103806")

## [1] 980

(b)

news\_corpus = tm\_map(news\_corpus, removePunctuation)  
news\_corpus[980]$content

## 103806   
## "From cheekeentartarusuwaeduau Desmond Chan\nSubject Re Honda clutch chatter\nOrganization The University of Western Australia\nLines 8\nNNTPPostingHost tartarusuwaeduau\nXNewsreader NN version 6419 1\n\n I also experience this kinda problem in my 89 BMW 318 During cold\nstart ups the clutch seems to be sticky and everytime i drive out for\nabout 5km the clutch seems to stick onto somewhere that if i depress\nthe clutch the whole chassis moves along But after preheating it\nbecomes smooth again I think that your suggestion of being some\nhumudity is right but there should be some remedy I also found out that\nmy clutch is already thin but still alright for a couple grand more\n"

news\_corpus = tm\_map(news\_corpus, removeNumbers)  
news\_corpus[980]$content

## 103806   
## "From cheekeentartarusuwaeduau Desmond Chan\nSubject Re Honda clutch chatter\nOrganization The University of Western Australia\nLines \nNNTPPostingHost tartarusuwaeduau\nXNewsreader NN version \n\n I also experience this kinda problem in my BMW During cold\nstart ups the clutch seems to be sticky and everytime i drive out for\nabout km the clutch seems to stick onto somewhere that if i depress\nthe clutch the whole chassis moves along But after preheating it\nbecomes smooth again I think that your suggestion of being some\nhumudity is right but there should be some remedy I also found out that\nmy clutch is already thin but still alright for a couple grand more\n"

news\_corpus = tm\_map(news\_corpus, tolower)  
news\_corpus[980]$content

## 103806   
## "from cheekeentartarusuwaeduau desmond chan\nsubject re honda clutch chatter\norganization the university of western australia\nlines \nnntppostinghost tartarusuwaeduau\nxnewsreader nn version \n\n i also experience this kinda problem in my bmw during cold\nstart ups the clutch seems to be sticky and everytime i drive out for\nabout km the clutch seems to stick onto somewhere that if i depress\nthe clutch the whole chassis moves along but after preheating it\nbecomes smooth again i think that your suggestion of being some\nhumudity is right but there should be some remedy i also found out that\nmy clutch is already thin but still alright for a couple grand more\n"

news\_corpus = tm\_map(news\_corpus, removeWords, stopwords("english"))  
news\_corpus[980]$content

## 103806   
## " cheekeentartarusuwaeduau desmond chan\nsubject re honda clutch chatter\norganization university western australia\nlines \nnntppostinghost tartarusuwaeduau\nxnewsreader nn version \n\n also experience kinda problem bmw cold\nstart ups clutch seems sticky everytime drive \n km clutch seems stick onto somewhere depress\n clutch whole chassis moves along preheating \nbecomes smooth think suggestion \nhumudity right remedy also found \n clutch already thin still alright couple grand \n"

(c)

dtm = DocumentTermMatrix(news\_corpus, control = list(weighting = weightTfIdf, minWordLength = 1, minDocFreq = 1))  
dim(dtm)

## [1] 1986 22213

terms <- c('bmw', 'clutch', 'mother')  
inspect(dtm[980,intersect(colnames(dtm), terms)])

## <<DocumentTermMatrix (documents: 1, terms: 3)>>  
## Non-/sparse entries: 2/1  
## Sparsity : 33%  
## Maximal term length: 6  
## Weighting : term frequency - inverse document frequency (normalized) (tf-idf)  
## Sample :  
## Terms  
## Docs bmw clutch mother  
## 103806 0.05710895 0.4289088 0

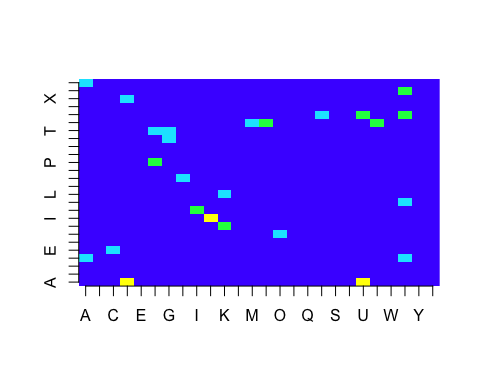
7.

(a)

require("MASS")

## Loading required package: MASS

raw\_data <- read.table("az-5000.txt", header = TRUE)  
smp\_size <- floor(0.8\*nrow(raw\_data))  
train\_ind <- sample(seq\_len(nrow(raw\_data)), size = smp\_size)  
priors <- c(rep(1/26, 26))  
data\_lda <- lda(char ~., raw\_data, subset = train\_ind, prior = priors)  
confusion\_table <- table(raw\_data[-train\_ind,]$char, predict(data\_lda, raw\_data[-train\_ind,])$class)  
  
diag(confusion\_table) <- 0  
  
labelpos <- 0:25  
  
labelpos\_std <- labelpos/25  
  
image(confusion\_table, col = topo.colors(4), axes = FALSE)  
  
axis(1, labelpos\_std, labels = LETTERS[1:26])  
axis(2, labelpos\_std, labels = LETTERS[1:26])



(b)

The color coresponding to the worst confusion is the yellow. The corresponding character pairs are (U,V),(A,D),(D,X).