Project-3

Chitra Karki[*] University of Texas at El Paso (UTEP)

October 11, 2022

Contents

---

1 Kernel PCA

Consider the data on optical recognition of handwritten digits, available from UCI Machine Learning Repository:
https://archive.ics.uci.edu/ml/machine-learning-databases/optdigits/
The following three files are needed: (1) optdigits.names (a description of the data set); (2) optdigits.tra (the training set); and (3) optdigits.tes (the test set).

(a) Bring in the training set **optdigits.tra**, which has sixty-four (p = 64) inputs plus the target variable that indicates the digit 0–9. Examine the data briefly. Remove columns that are unary (i.e., containing only one values) or close to being unary (i.e., nearly all values are the same except a few). And check on possible missing values.

(b) Excluding the target variables, run the ordinary principal components analysis (PCA) with the training set. Output the scree plot of the variances (i.e., eigenvalues) of the principal components. Make a scatter plot of the first two PCs and show the target class variable (i.e., digit number) with different symbols and colors. Recall that this also corresponds to a multidimensional scaling (MDS) analysis of data. Interpret results.

  – careful with the normalization or standardization of the data before performing PCA. Make parallel boxplots of the attributes and inspect for necessity of data normalization.

[*]cbkarki@miners.utep.edu

(c) Run kernel PCA on the input variables only. Explain your choice of kernel function and the choice of parameters involved. Output the scree plot of the variances (i.e., eigenvalues) of the resultant principal components. Plot the first two PCs with scatted points and show the target class variable with different symbols and colors. Compare the kPCA results with the PCA results and comment with interpretations.

(d) Apply both the PCA and kPCA results learned from the training data to the test set **optdigits.tes**, which can be simply done by using the **predict()** function. Obtain the first two principal components in each case and make similar plots as Part (b) & (c) and compare

```
# (a)

setwd("C:/Users/chitr/OneDrive - University of Texas at El Paso/data_science/semesters/sem3-fal

dat.tra = read.csv(file = "optdigits.tra")
names(dat.tra)=paste("x",1:ncol(dat.tra),sep = "")

#class labels
table(dat.tra[,ncol(dat.tra)])
```

```
##
##   0   1   2   3   4   5   6   7   8   9
## 375 389 380 389 387 376 377 387 380 382
```

```
# detecting uniary variablesand nearly uniary variables cut off 90%

# filtering columns which have less then 5 unique values
uni.var = c(which(as.vector(apply(dat.tra[,-ncol(dat.tra)],2,function(x) table(x)[1]/nrow(dat.t
apply(dat.tra[,uni.var],2,table)
```

```
## $x1
##
##    0
## 3822
##
## $x8
##
##    0    1    2    3    4    5    6    7    8    9   10   11   12   14   15   16
## 3708   28   16   12   10   10    8    2   10    3    1    3    6    2    2    1
##
## $x9
##
##    0    1    2    5
## 3819    1    1    1
##
## $x16
##
```

```
##    0    1    2    3    4    5    6    7    8    9   10   11   12   13   15
## 3680   34   27   21   10    9    8   11    6    6    1    5    1    2    1
##
## $x17
##
##    0    1    2    3    5
## 3813    4    2    2    1
##
## $x24
##
##    0    1    2    3    4    5    6    7    8
## 3755   20   16    9   11    4    5    1    1
##
## $x25
##
##    0    1
## 3818    4
##
## $x32
##
##    0    1    2
## 3812    8    2
##
## $x33
##
##    0    1
## 3817    5
##
## $x40
##
##    0
## 3822
##
## $x41
##
##    0    1    2    3    4    5    6    7
## 3783   11   11    6    5    3    2    1
##
## $x48
##
##    0    1    2    4    6
## 3774   32   11    4    1
##
## $x49
##
##    0    1    2    3    4    5    7   10
## 3790   20    4    3    1    2    1    1
##
```

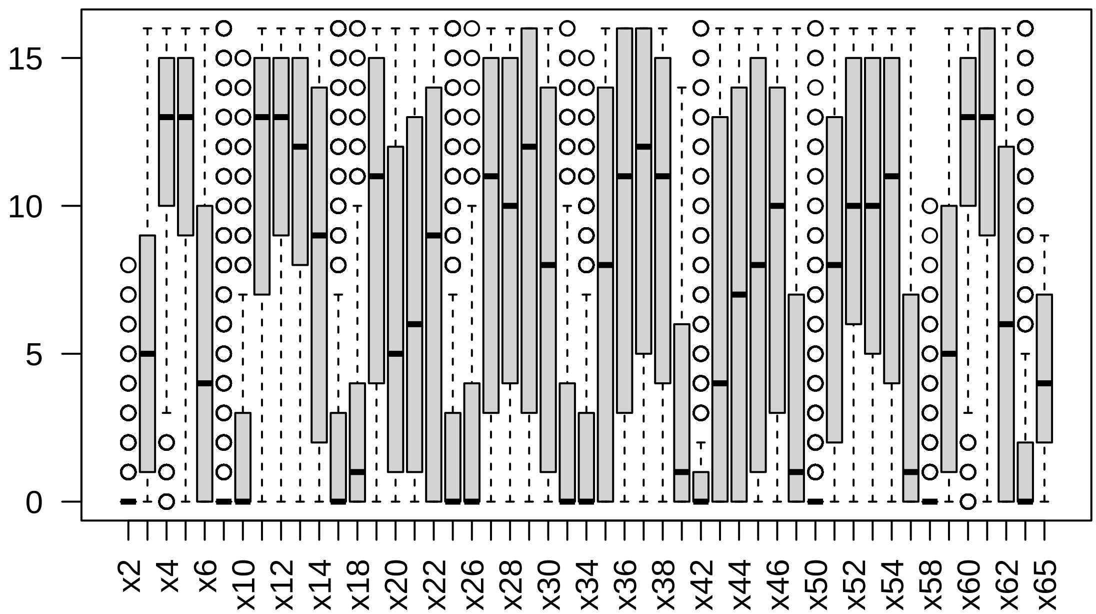
```
## $x56
##
##    0    1    2    3    4    5    6    7    8   10   12
## 3618   62   53   35   17   20    6    4    5    1    1
##
## $x57
##
##    0    1
## 3821    1
##
## $x64
##
##    0    1    2    3    4    5    6    7    8    9   11   12   13   14   15   16
## 3615   67   34   30   12   19   14    7    5    3    5    2    4    2    2    1
```

```
dat.tra = dat.tra[,-uni.var]


# check for na values
sum(is.na(dat.tra))
```

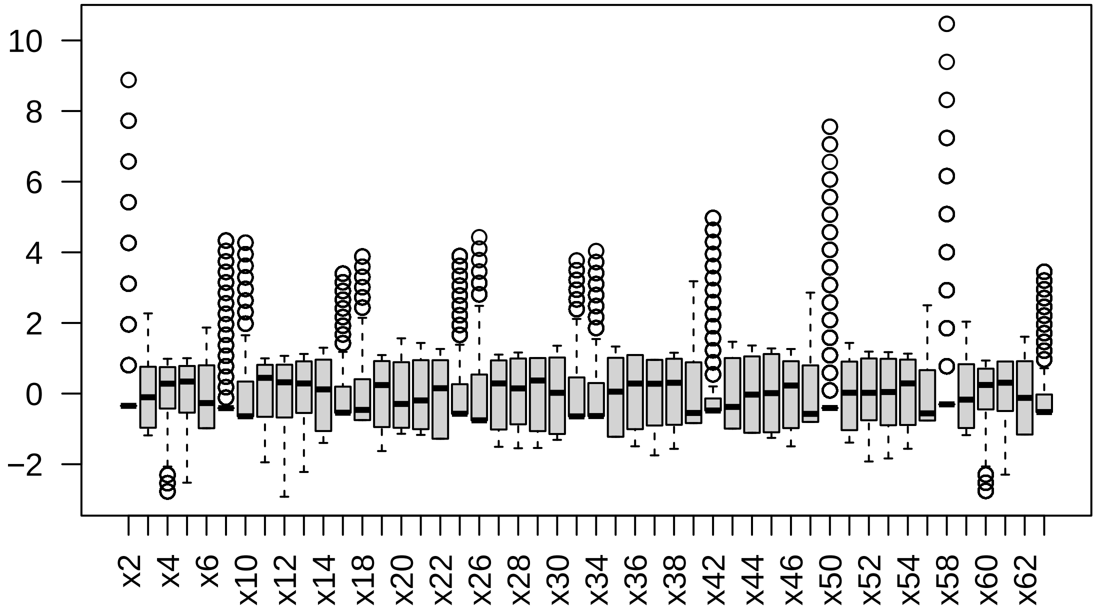
```
## [1] 0
```

There are no missing values.For detecting uniary attributes, the relative frequency of high occurring unique value in each attribute was calculated. Attribute for which relative frequency is greater then 90 % was considered as uniary attribute. The cutoff of 90 % was defined intuitively. in this way we knocked out 16 attributes.

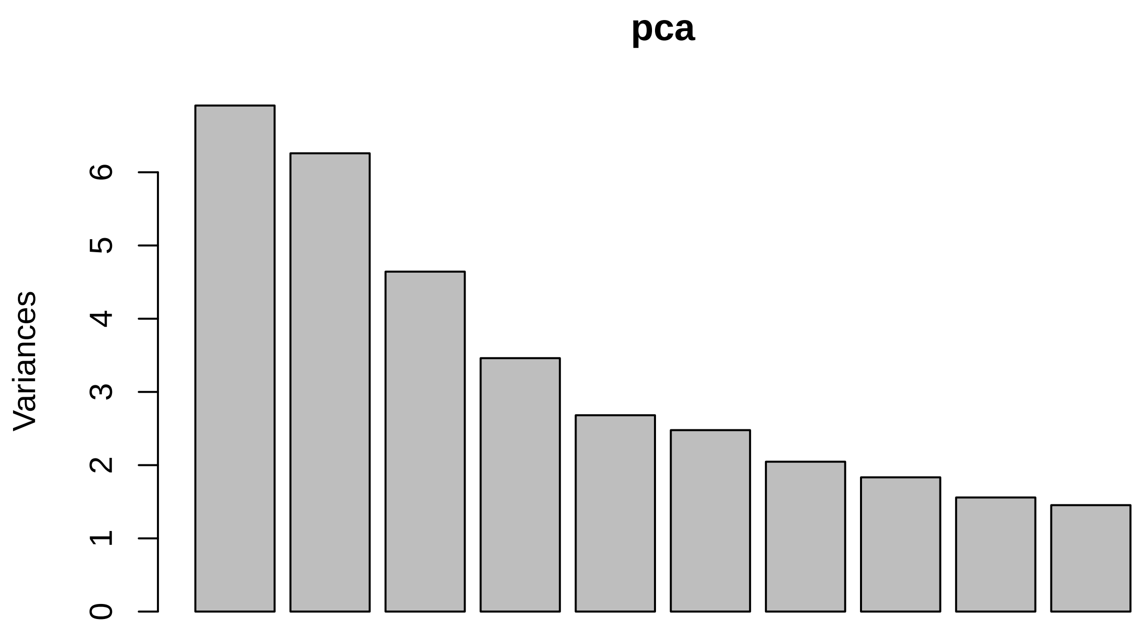
```
# (b) ordinary pca

# boxplot to see if scaling is required.
boxplot(dat.tra,las=2)
```



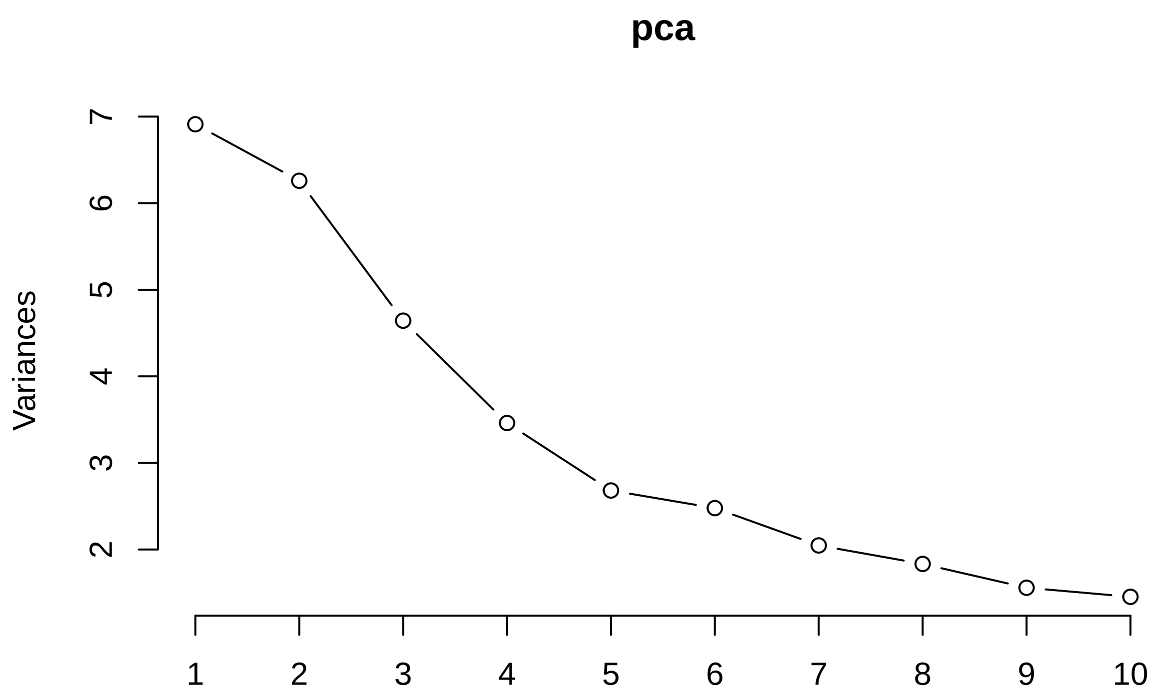

```
dat.tra.scaled = scale(dat.tra[,-ncol(dat.tra)],scale = T,center = T)
boxplot(dat.tra.scaled,las=2)
```



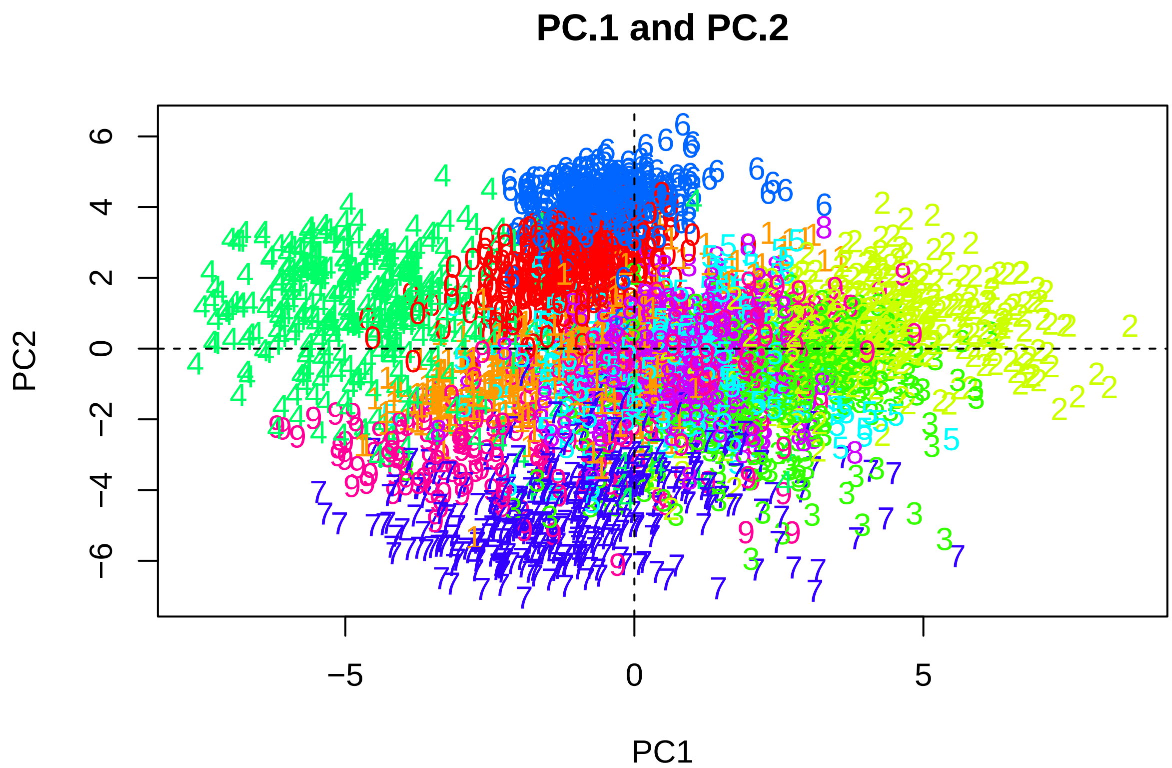

```
# ------------------
# ORDINARY PCA
# ------------------
pca <- prcomp(dat.tra.scaled, retx=TRUE, center=F, scale=F)
#pca

# screeplot:
#plot(pca)
screeplot(pca);
```



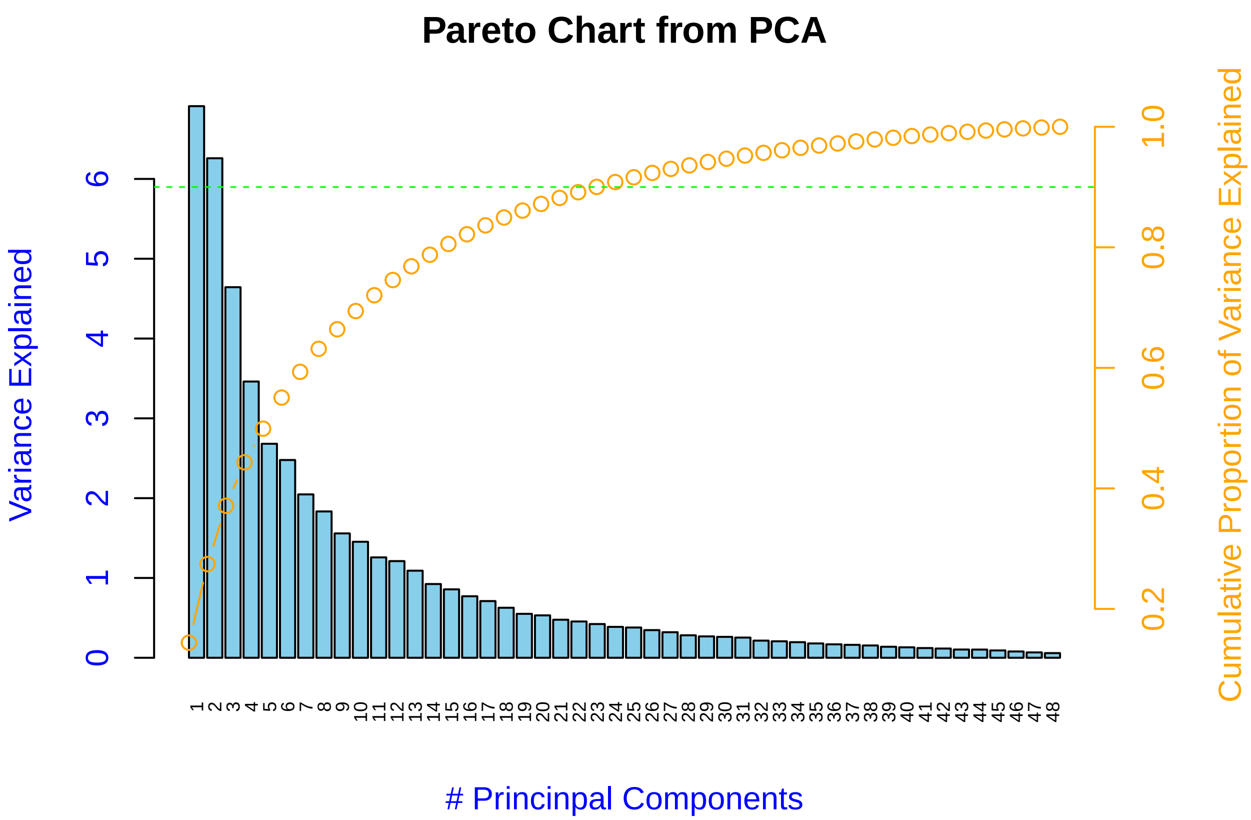

```r
screeplot(pca, type="lines")
```




```
# PLOT FIRST TWO PCs
colors = rainbow(length(unique(dat.tra[,ncol(dat.tra)])))
plot(pca$x[,1:2], pch="", main="PC.1 and PC.2")
text(pca$x[,1:2], labels=dat.tra[,ncol(dat.tra)],
     col=colors[factor(dat.tra[,ncol(dat.tra)])])
abline(v=0, lty=2)
abline(h=0, lty=2)
```



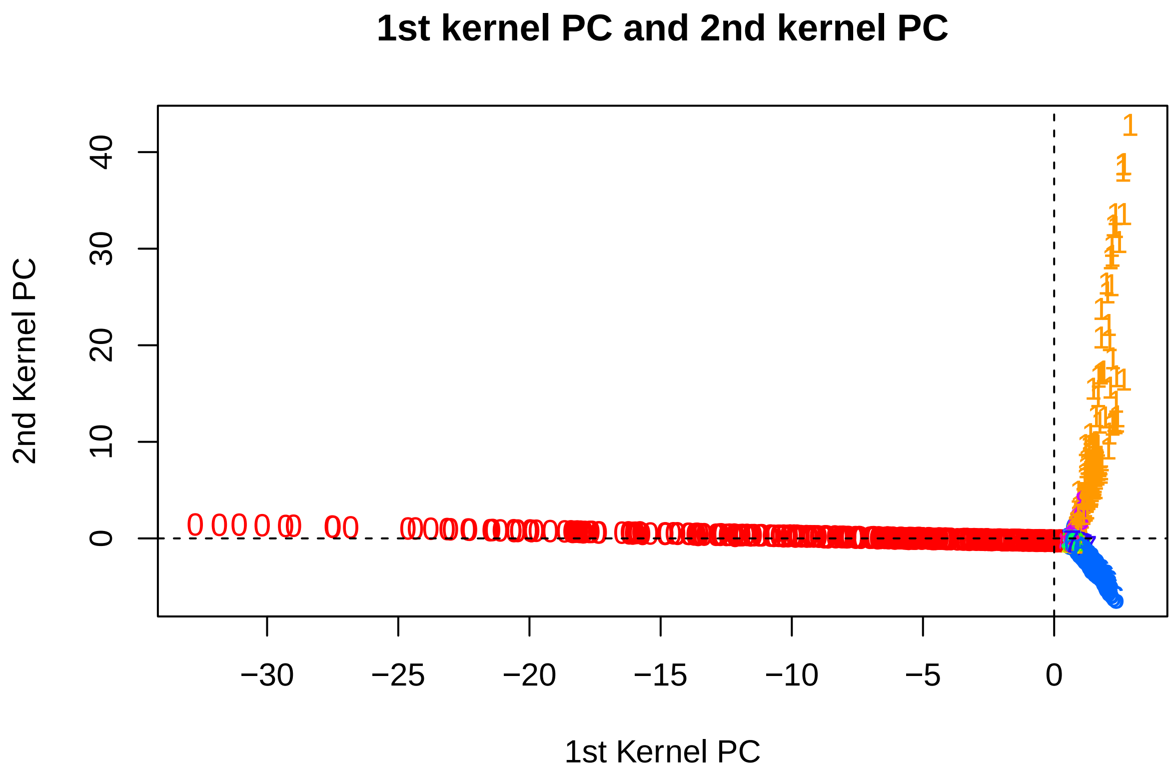

```r
#  PUTS VARIANCE AND CUMULATIVE PROPORTIONS TOGETHER -- THE PARETO PLOT
sd.pc <- pca$sdev
var.pc <- sd.pc^2
prop.pc <- var.pc/sum(var.pc)

par(mar=c(4, 4, 4, 4))
bar <- barplot(var.pc, ylab="Variance Explained", col="skyblue",
    xlab="# Princinpal Components", col.axis="blue", col.lab="blue")
mtext(1:length(var.pc),side=1, line=1,at=bar,col="black",las=2,cex = 0.60)

par(new=T)
plot(bar, cumsum(prop.pc),axes=F,xlab="", ylab="", col="orange", type="b",
    col.lab="orange", col.lab="orange")
axis(4,col="orange", col.ticks="orange", col.axis="orange")
abline(h=.90, lty=2, col="green", lwd=.8)
mtext("Cumulative Proportion of Variance Explained",side=4,line=3,col="orange")
title(main = 'Pareto Chart from PCA')
```



Observing boxplots, it can be concluded that the scaling is needed because the variance in each columns is different and will affect the PC analysis.Its observed that the pca didn't separate the handwritten digits well because the overlap are present.First five principal components explain most of the variation, after that the variation explained progresses in a slow rate.

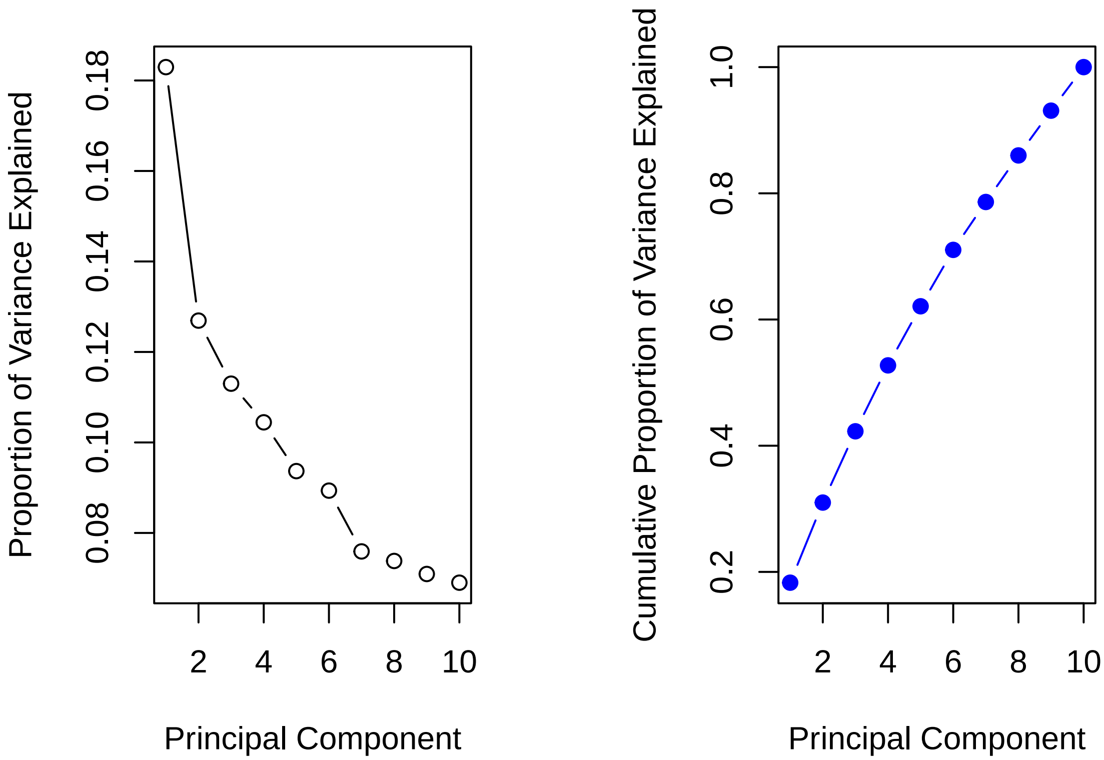
```
# (c) kernel pca
# ------------------
# KERNEL PCA
# ------------------
library("kernlab")
kpc <- kpca(~., data=as.data.frame(dat.tra.scaled), kernel="rbfdot",
    kpar=list(sigma=0.2), features=10)

# ?kpca
#eig(kpc)        # returns the eigenvalues
#kernelf(kpc)    # returns the kernel used when kpca was performed
PCV <- pcv(kpc)        # returns the principal component vectors (BE CAREFUL!)
#dim(PCV); head(PCV);
PC <- rotated(kpc)    # returns the data projected in the (kernel) pca space
#dim(PC); head(PC);

# Plot THE DATA PROJECTION ON THE KERNEL PCS
plot(PC[,1:2], pch="", main="1st kernel PC and 2nd kernel PC",
```

```
     xlab="1st Kernel PC", ylab="2nd Kernel PC")
text(PC[,1:2], labels=dat.tra[,ncol(dat.tra)],
     col=colors[factor(dat.tra[,ncol(dat.tra)])])
abline(v=0, lty=2)
abline(h=0, lty=2)
```

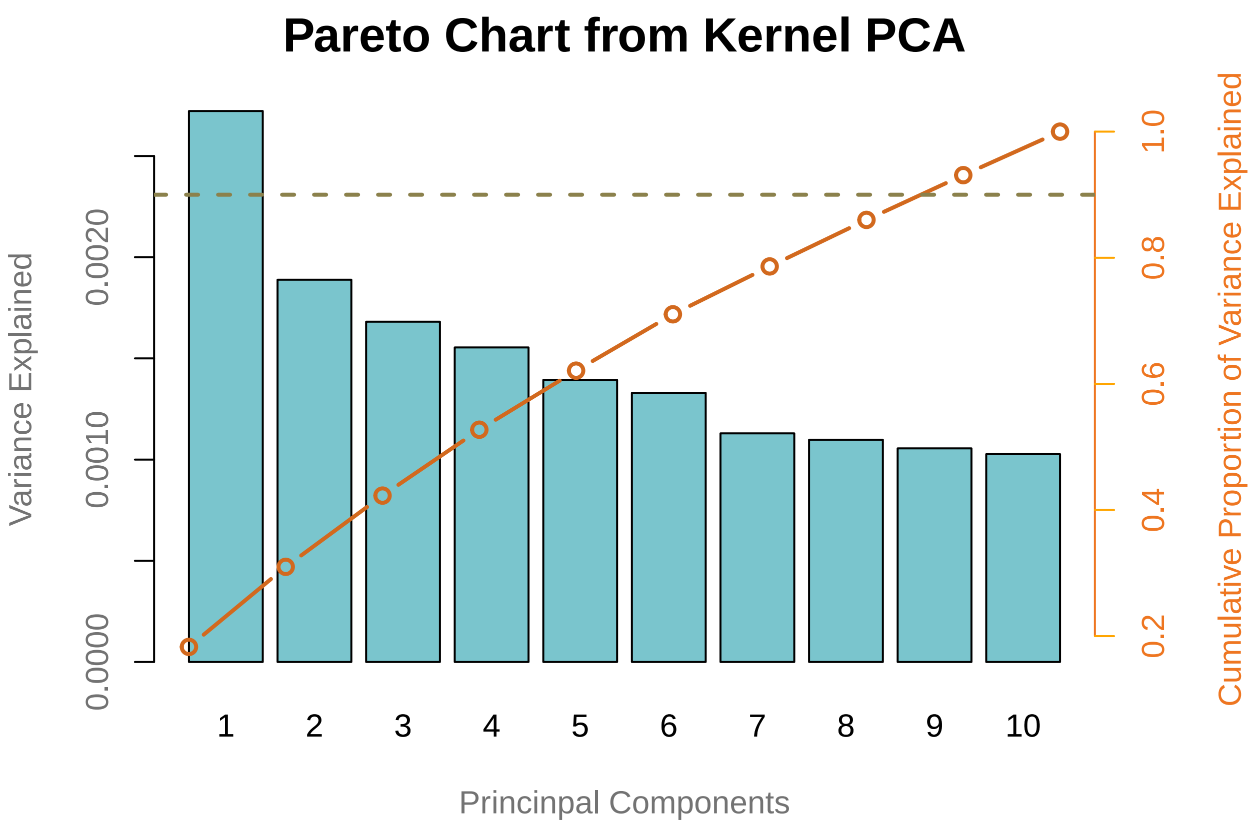



```
# EMBED WITH PREDICTION ON NEW DATA
# pred <- predict(kpc, bumpus.scaled);
# head(pred); head(rotated(kpc))

# COMPUTE NONCUMULATIVE/CUMULATIVE PROPORTIONS OF VARIATION EXPLAINED
var.pc <- eig(kpc)
names(var.pc) <- 1:length(var.pc)
prop.pc <- var.pc/sum(var.pc)

par(mfrow=c(1,2), mar=rep(4,4))
# NONCUMULATIVE
plot(prop.pc, xlab = "Principal Component",
    ylab = "Proportion of Variance Explained", type = "b")
# CUMULATIVE
plot(cumsum(prop.pc), xlab = "Principal Component", col="blue",
               ylab = "Cumulative Proportion of Variance Explained",
```

```
              type = "b", pch=19)
```




```
#  PUTS VARIANCE AND CUMULATIVE PROPORTIONS TOGETHER -- THE PARETO PLOT
par(mar=c(4, 4, 4, 4), mfrow=c(1,1))
bar <- barplot(var.pc, ylab="Variance Explained", col="cadetblue3",
    xlab="Princinpal Components", col.axis="gray45", col.lab="gray45")
mtext(1:length(var.pc),side=1, line=1,at=bar,col="black")
# mtext(" ",side=1,line=3,col="black")
par(new=T)
plot(bar, cumsum(prop.pc),axes=F,xlab="", ylab="", col="chocolate", type="b",
    col.lab="orange", col.lab="orange", lwd=2)
axis(4, col="chocolate2", col.ticks="orange", col.axis="chocolate2")
abline(h=.90, lty=2, col="lightgoldenrod4", lwd=2)
mtext("Cumulative Proportion of Variance Explained", side=4,
      line=3,col="chocolate2")
title(main = 'Pareto Chart from Kernel PCA', cex.main=1.5)
```



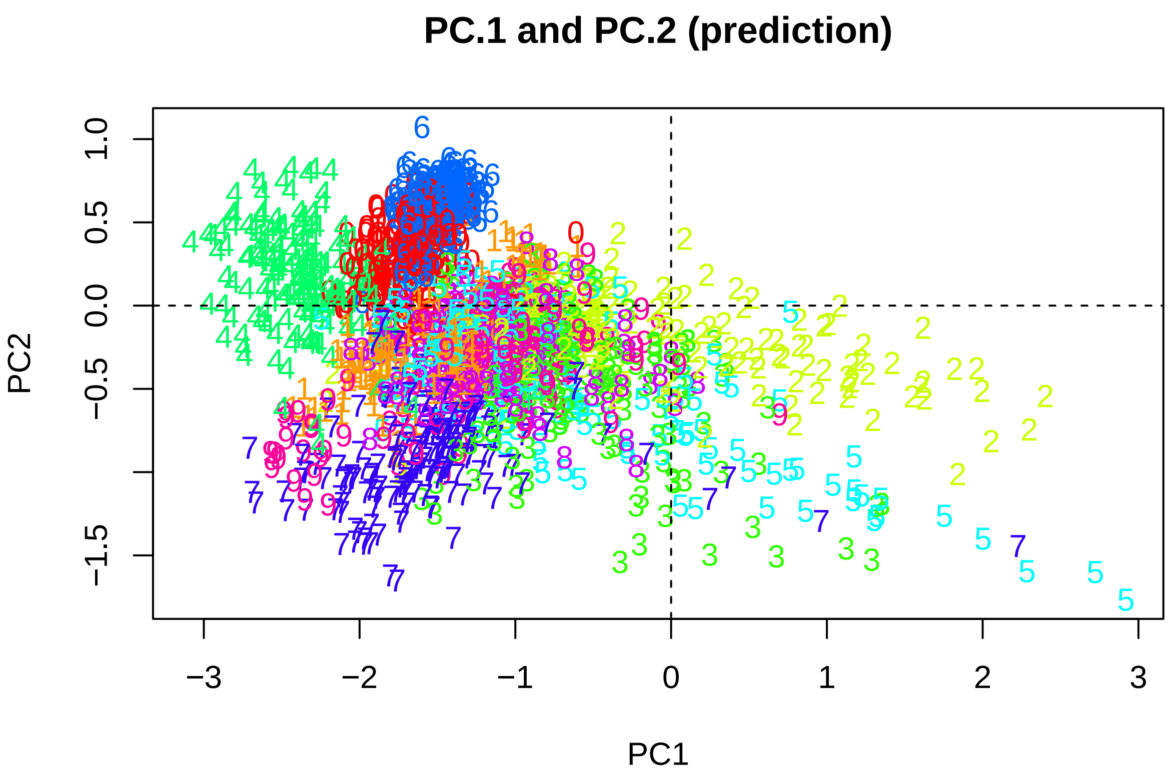
For kernel PCA rbf kernel function was used with sigma 0.2. 10 features were outputted. First two kernel PCA captured variations of 1, 6 and 0, but other digits are not distinguished.

```
# (d)
dat.test = read.csv("optdigits.tes")
names(dat.test)=paste("x",1:ncol(dat.test),sep = "")

table(dat.test[,ncol(dat.test)])
```

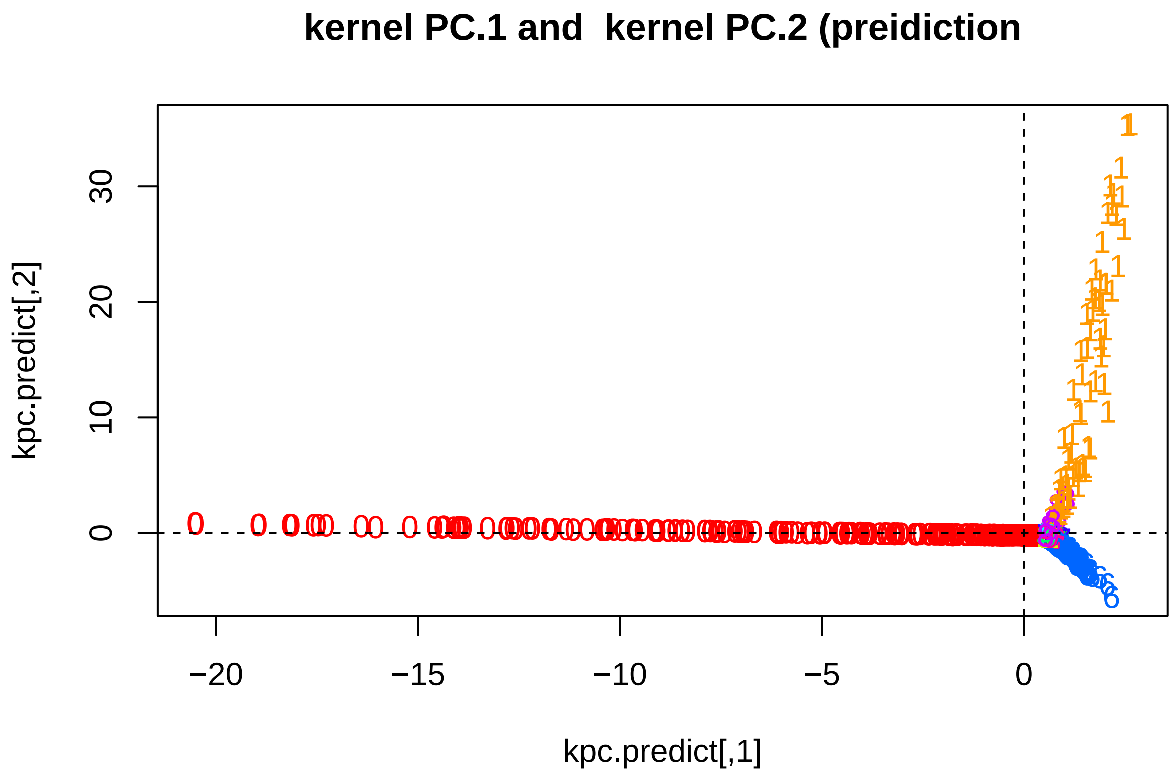
```
##
##   0   1   2   3   4   5   6   7   8   9
## 177 182 177 183 181 182 181 179 174 180
```

```
# eliminating the columns that were eliminated during the process of cleaning data for trainin
dat.test = dat.test[,-uni.var]

# scaling the testing data with means and sd of training set.
means = apply(dat.tra[,-ncol(dat.tra)], 2, mean)
sds = apply(dat.tra[,-ncol(dat.tra)], 2, sd)

dat.test.scaled = scale(dat.test[,-ncol(dat.test)],center = means,scale = sds)

# prediction for ordinary PCA
```

```
pca.predict = predict(pca,newdata=dat.test.scaled)[,1:2]

colors = rainbow(length(unique(dat.test[,ncol(dat.test)])))
plot(pca.predict, pch="", main="PC.1 and PC.2 (prediction)")
text(pca.predict, labels=dat.test[,ncol(dat.test)],
     col=colors[factor(dat.test[,ncol(dat.test)])])
abline(v=0, lty=2)
abline(h=0, lty=2)
```




```
# predication for kernel PCA
kpc.predict = predict(kpc,dat.test.scaled)[,1:2]

colors = rainbow(length(unique(dat.test[,ncol(dat.test)])))
plot(kpc.predict, pch="", main="kernel PC.1 and  kernel PC.2 (preidiction")
text(kpc.predict, labels=dat.test[,ncol(dat.test)],
     col=colors[factor(dat.test[,ncol(dat.test)])])
abline(v=0, lty=2)
abline(h=0, lty=2)
```



The predictions from pca and kernel pca both didn't separate the digits well. though both methods seperate some of the digits in some extend, most of them have overlaps.

2 Association Rules

A parsed-version (with punctuation and stop words removed) of the King James Bible1 is provided in:
http://snap.stanford.edu/class/cs246-data/AV1611Bible.txt
Here, each line is a sentence in the document. We are interested in finding which words commonly occur together in sentences.

(a) First read the data into R as transaction data type. This can be done using R function **read.transactions()** in the arules package:

- library(arules)
  bible <- read.transactions(file="AV1611Bible.txt",
  format = "basket", sep =" ", rm.duplicates =F,
  quote="") # DOUBLE/SINGLE QUOTE ISSUE
  dat <- bible; dim(dat)
  inspect(dat[1:5,])

(b) Set up the parameters in R function **arules** appropriately with your own choices and then perform frequent itemsets and association rule analysis

(c) List the top 5 rules in decreasing order of confidence (conf) for item sets of size/length 2 or 3 which satisfy the support threshold that you have specified. Are they interesting rules within the problem context?

(d) List the top 5 rules in decreasing order of the lift measure for item sets of size 2 or 3. Always interpret the results.

```
library(arules)
```

```
## Loading required package: Matrix

##
## Attaching package: 'arules'

## The following object is masked from 'package:kernlab':
##
##     size

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

```
bible <- read.transactions(file="AV1611Bible.txt",
format = "basket", sep =" ", rm.duplicates =F,
quote="") # DOUBLE/SINGLE QUOTE ISSUE

dim(bible)
```

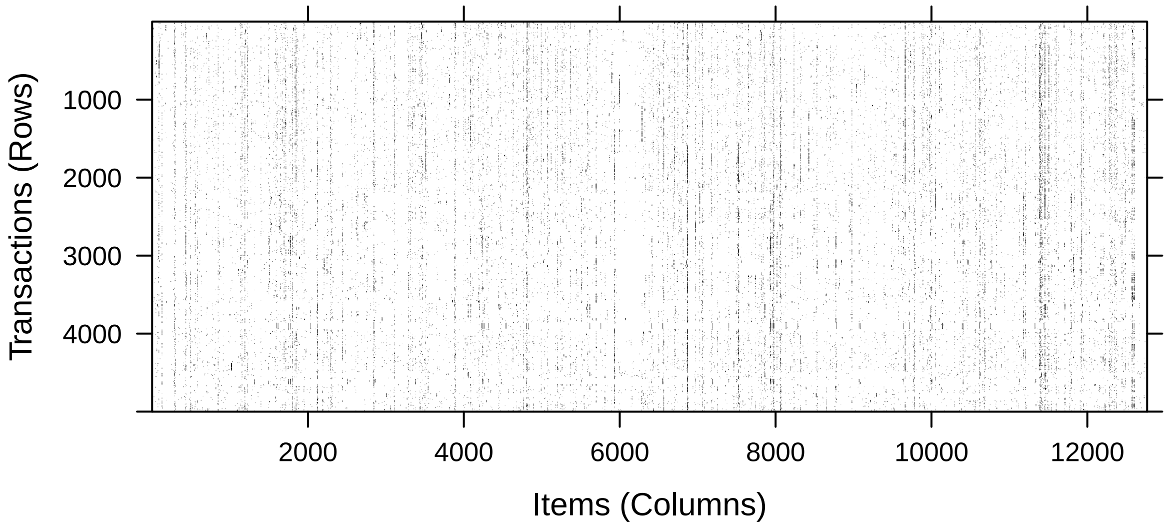
```
## [1] 31101 12767
```

```
inspect(bible[1:5, ])
```

```
##     items
## [1] {beginning,
##      created,
##      earth,
##      god,
##      heaven}
## [2] {darkness,
##      deep,
##      earth,
##      face,
```

```
##      form,
##      god,
##      moved,
##      spirit,
##      upon,
##      void,
##      waters,
##      without}
## [3] {god,
##      let,
##      light,
##      said,
##      there}
## [4] {darkness,
##      divided,
##      god,
##      good,
##      light,
##      saw}
## [5] {called,
##      darkness,
##      day,
##      evening,
##      first,
##      god,
##      light,
##      morning,
##      night}
```

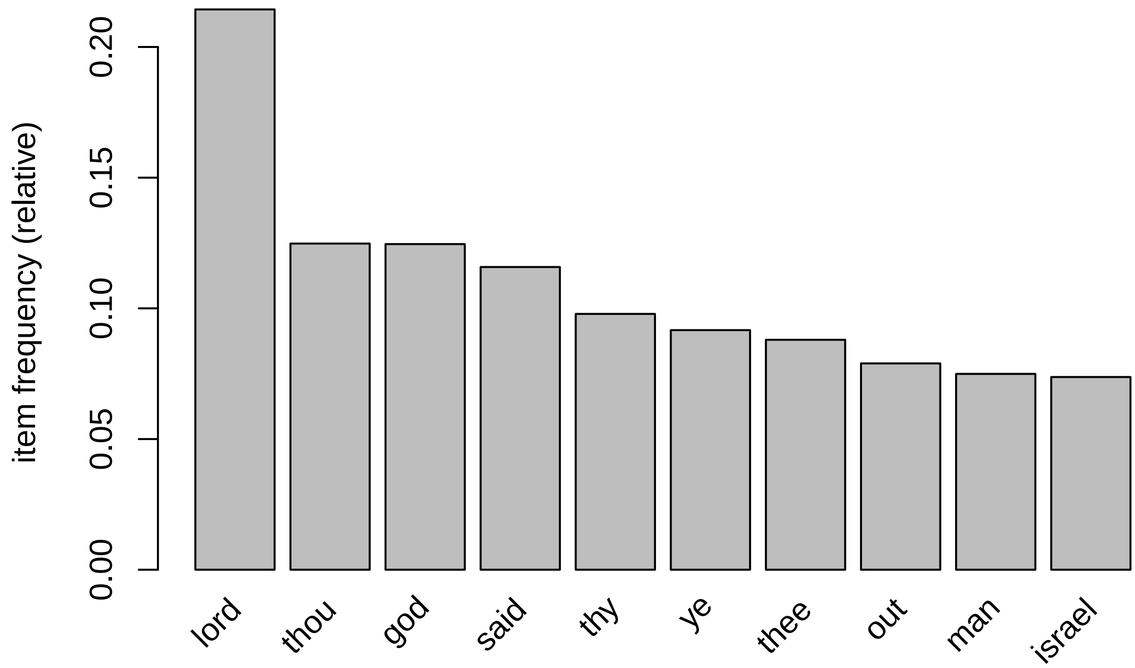
```
image(bible[1:5000,]) # first 2500 transactions or rows,
```




```r
#shows the sparsity of the data

itemFrequency(bible[,1:10],type="relative") # supprot/frequency of the first 10
```

```
##        aaron      aaron's    aaronites      abaddon     abagtha        abana
## 9.742452e-03 9.967525e-04 6.430661e-05 3.215331e-05 3.215331e-05 3.215331e-05
##       abarim        abase       abased      abasing
## 1.286132e-04 1.286132e-04 1.286132e-04 3.215331e-05
```

```r
#distinct items or the words in the bible
# itemFrequency(bible[,1:10],type="absolute")


#itemFrequencyPlot(bible[,1:10],type = "absolute")
itemFrequencyPlot(bible,type = "relative",support=0.01,topN=10)
```




```r
# (b)
# ?apriori
rules <- apriori(bible, parameter = list(support = 0.01, confidence = 0.6,
    target = "rules", maxlen=5))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.6    0.1    1 none FALSE            TRUE       5    0.01      1
##  maxlen target  ext
##       5  rules TRUE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 311
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[12767 item(s), 31101 transaction(s)] done [0.20s].
## sorting and recoding items ... [230 item(s)] done [0.00s].
```

```
## creating transaction tree ... done [0.01s].
## checking subsets of size 1 2 3 4 done [0.01s].
## writing ... [17 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
summary(rules)
```

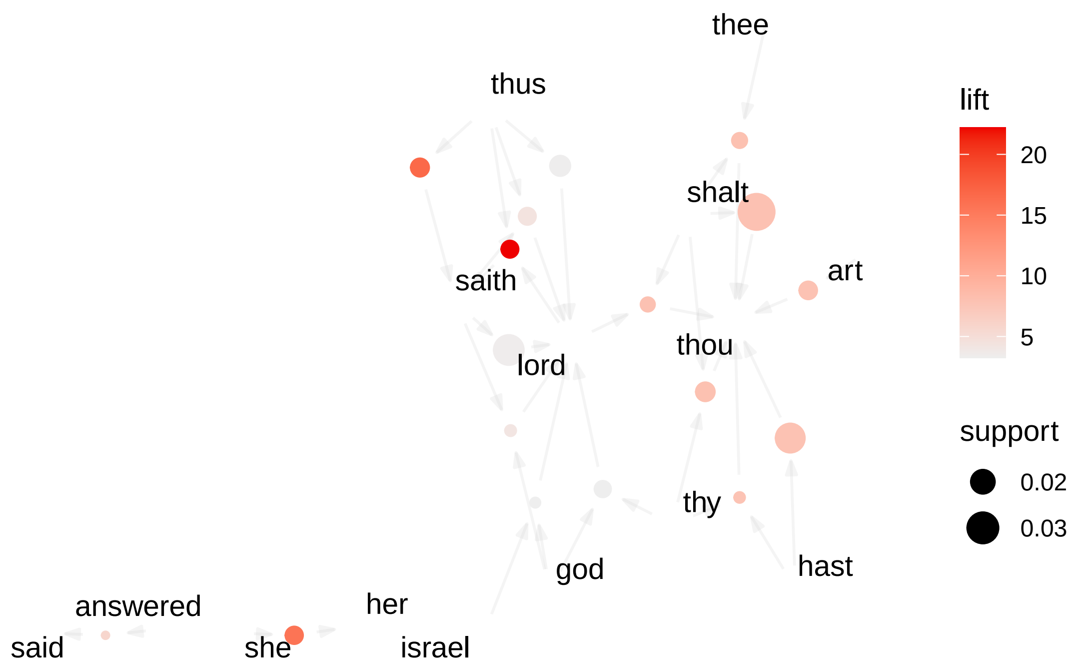
```
## set of 17 rules
##
## rule length distribution (lhs + rhs):sizes
## 2 3
## 8 9
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   2.000   2.000   3.000   2.529   3.000   3.000
##
## summary of quality measures:
##     support          confidence        coverage            lift
##  Min.   :0.01067   Min.   :0.6016   Min.   :0.01132   Min.   : 3.258
##  1st Qu.:0.01225   1st Qu.:0.7033   1st Qu.:0.01453   1st Qu.: 4.191
##  Median :0.01389   Median :0.8984   Median :0.01576   Median : 7.882
##  Mean   :0.01637   Mean   :0.8480   Mean   :0.01961   Mean   : 8.116
##  3rd Qu.:0.01514   3rd Qu.:0.9867   3rd Qu.:0.02273   3rd Qu.: 8.014
##  Max.   :0.03900   Max.   :1.0000   Max.   :0.03903   Max.   :22.183
##      count
##  Min.   : 332.0
##  1st Qu.: 381.0
##  Median : 432.0
##  Mean   : 509.2
##  3rd Qu.: 471.0
##  Max.   :1213.0
##
## mining info:
##   data ntransactions support confidence
##  bible         31101    0.01        0.6
##
##  apriori(data = bible, parameter = list(support = 0.01, confidence = 0.6, target = "rules",
```

```
library(arulesViz)

plot(rules, method="graph", control=list(type="items"))
```

```
## Warning: Unknown control parameters: type
```

```
## Available control parameters (with default values):
## layout    =  stress
```

```
## circular  =  FALSE
## ggraphdots    =  NULL
## edges     =  <environment>
## nodes     =  <environment>
## nodetext  =  <environment>
## colors    =  c("#EE0000FF", "#EEEEEEFF")
## engine    =  ggplot2
## max   =  100
## verbose   =  FALSE
```



Support was assigned 0.01 ,confidence; 0.6 with maximum length of rule to be 5 producing 17 rules. 8 rules are of length 2 and 9 rules are of length 3.

```
# (c)
#rules.by.confidence = sort(rules,by="confidence",decreasing = T)
inspect(head(rules,n=5,by="confidence"))
```

```
##     lhs                rhs     support    confidence coverage   lift     count
## [1] {shalt, thee} => {thou} 0.01270056 1.0000000  0.01270056 8.013656  395
## [2] {shalt, thy}  => {thou} 0.01514421 1.0000000  0.01514421 8.013656  471
## [3] {shalt}       => {thou} 0.03900196 0.9991763  0.03903411 8.007055 1213
## [4] {lord, shalt} => {thou} 0.01225041 0.9973822  0.01228256 7.992678  381
## [5] {art}         => {thou} 0.01434037 0.9867257  0.01453329 7.907280  446
```

Yes, it makes sense in the context because, person using words shalt, thee, thy for your in a certain dialect also tend to use word thou for your.

```
# (d)
inspect(head(rules,n=5,by="lift"))
```

```
##     lhs               rhs     support    confidence coverage   lift      count
## [1] {lord, thus}   => {saith} 0.01389023 0.8537549  0.01626957 22.182650 432
## [2] {thus}         => {saith} 0.01462975 0.6435644  0.02273239 16.721383 455
## [3] {she}          => {her}   0.01408315 0.6016484  0.02340761 15.684715 438
## [4] {shalt, thee} => {thou}  0.01270056 1.0000000  0.01270056  8.013656 395
## [5] {shalt, thy}  => {thou}  0.01514421 1.0000000  0.01514421  8.013656 471
```

For all the rules, when sorted by lift, have lift greater then 1, meaning that the lhs and rhs of the rules have positive relation and chances of using them together also increases. Somehow, lhs and rhs of the rules are also realted words.

```
# (e)

interestMeasure(head(rules,n=5,by="lift"), c( "conviction"),
                transactions = bible)
```

```
## [1] 6.574666 2.697577 2.414051      Inf      Inf
```

In association rule with confidence and lift the directionality of the rules can't be distinguished. With conviction, directions of the rules matter.The conviction infinite means the confidence of the rule is 1. We have two rules with infinity and other three are greater then 1, meaning rhs of the rules is depending on lhs of the rule.