Project III: Kernel PCA and Association Rules

DS 5494 - Statistical Machine Learning II

Willliam Ofosu Agyapong* University of Texas at El Paso (UTEP)

October 11, 2022

Contents

*woagyapong@miners.utep.edu

1 Ordinary PCA versus Kernel PCA

We first import both the training and the test sets which contain optical recognition of handwritten digits.

```
# Import the training set
train_set <- read.table(file=
"http://archive.ics.uci.edu/ml/machine-learning-databases/optdigits/optdigits.tra",
sep=",", header = F, na.strings = c("NA", "", " "),
col.names = c(paste("I", 1:64, sep=""), "target"))

# Import the test set
test_set <- read.table(file=
"http://archive.ics.uci.edu/ml/machine-learning-databases/optdigits/optdigits.tes",
sep=",", header = F, na.strings = c("NA", "", " "),
col.names = c(paste("I", 1:64, sep=""), "target"))
dim(train_set); dim(test_set)
```

```
## [1] 3823   65
```

```
## [1] 1797   65
```

```
# Combine train and test data for some EDA
combined_set <- dplyr::bind_rows(train_set, test_set)
```

The training and testing data have 3823 and 1797 observations, respectively, with 65 number of columns.

```
combined_set %>%
  slice_sample(n=10) %>%
  kable(booktabs=T, linesep="", align = "c",
        caption = "10 random samples from the combined training and test data") %>%
  kable_styling(latex_options = c("scale_down", "HOLD_position")) %>%
  kable_classic()
```

Table 1: 10 random samples from the combined training and test data

I1	I2	I3	I4	I5	I6	I7	I8	I9	I10	I11	I12	I13	I14	I15	I16	I17	I18	I19	I20	I21	I22	I23	I24	I25	I26	I27	I28	I29	I30	I31	I32	I33	I34	I35	I36	I37	I38	I39	I40	I41	I42	I43	I44	I45	I46	I47	I48	I49	I50	I51	I52	I53	I54	I55	I56	I57	I58	I59	I60	I61	I62	I63	I64	target
0	0	11	10	0	0	0	0	0	0	13	15	0	0	0	0	0	0	12	16	5	0	0	0	0	1	15	16	5	0	0	0	0	0	3	13	10	0	0	0	0	0	0	10	14	0	0	0	0	0	5	11	16	9	5	1	0	0	12	16	16	16	16	12	1
0	0	4	13	13	8	0	0	0	0	0	16	1	1	11	2	0	0	4	13	0	0	6	7	0	0	3	14	0	9	14	2	0	0	0	0	9	15	14	1	0	0	0	0	5	16	9	0	0	0	0	0	8	12	15	2	0	0	0	0	4	15	13	2	0	0	8
0	0	9	16	15	5	0	0	0	0	0	4	7	12	15	0	0	0	0	0	1	9	16	3	0	0	0	0	10	16	13	0	0	0	0	0	3	8	15	9	0	0	0	0	0	0	7	14	0	0	0	0	10	7	4	10	16	1	0	0	7	16	16	16	10	0	3
0	0	9	16	12	3	0	0	0	0	0	9	16	16	7	0	0	0	0	8	16	14	2	0	0	0	0	14	16	14	0	0	0	0	0	0	15	16	16	1	0	0	0	2	15	16	14	1	0	0	0	0	14	16	8	0	0	0	0	0	12	16	10	0	0	0	1
0	0	6	12	14	4	0	0	0	0	2	15	2	1	15	0	0	0	2	14	0	3	16	2	0	0	0	13	5	14	14	6	0	0	0	0	2	8	4	7	9	0	0	0	0	0	0	9	9	0	0	0	0	5	0	5	14	3	0	0	0	7	13	12	4	0	0	9
0	0	0	0	15	11	0	0	0	0	0	0	1	16	8	0	0	0	0	0	12	13	1	0	0	0	0	7	15	2	12	3	0	0	0	1	15	7	2	16	4	0	0	10	16	11	11	16	3	0	0	0	15	12	12	15	16	5	0	0	0	0	0	13	7	0	0	4
0	0	1	9	14	5	0	0	0	0	0	8	15	9	14	1	0	0	0	3	14	0	16	4	0	0	0	0	8	14	16	4	0	0	0	0	0	0	3	13	5	0	0	0	3	0	0	8	7	0	0	0	3	15	6	2	14	6	0	0	0	1	10	14	14	1	0	9
0	0	6	12	14	14	2	0	0	0	12	16	11	4	5	2	0	0	4	14	5	0	0	0	0	0	0	9	13	1	0	0	0	0	0	2	13	15	6	0	0	0	0	0	2	11	15	6	0	0	0	0	8	6	6	15	8	0	0	0	6	16	16	9	0	5
0	0	4	15	14	1	0	0	0	0	2	16	14	16	9	0	0	0	5	16	1	10	16	0	0	0	4	16	16	16	16	1	0	0	0	0	9	12	10	16	6	0	0	0	0	0	0	13	7	0	0	0	3	6	5	14	8	0	0	0	6	16	16	15	2	0	9
0	0	5	14	12	5	0	0	0	0	0	3	16	16	9	0	0	0	0	2	16	16	7	0	0	0	0	5	16	16	3	0	0	0	0	0	10	16	16	4	0	0	0	0	9	16	14	1	0	0	0	0	8	16	16	2	0	0	0	0	8	15	13	4	0	0	1

1.1 Part (a): Examining the data

We inspected the distinct values in each input variable to determine which ones are unary or close to unary by printing out the number of observations that corresponds to every distinct value for all the input variables, but for conciseness the outputs were not reported. The results clearly revealed inputs 1, and 40 as unary (contain only one values) whiles the inputs 8, 9, 16, 17, 24,25, 32, 33, 41, 48, 49, 56, 57, and 64 were found to be reasonably close to unary (with nearly all values, about 99%, being zeros except a few). These inputs variables shall be removed accordingly.

```
cols <- 1:NCOL(train_set)
for (j in cols){
  x <- train_set[,j]
  print(names(train_set)[j])
  # print(sort(unique(x, incomparables=TRUE)))
  print(table(x, useNA="ifany"))
}

# 1, 8, 9, 16, 17, 24,25, 32, 33, 40, 41, 48, 49, 56, 57, 64 (suspect)
```

```
output <- NULL
roles <- c(rep("Input", 64), "Target")
for(i in seq_along(train_set)) {
  output <- rbind(output, c(names(train_set)[i],
                  class(train_set[[i]]),
                  roles[i],
                  as.numeric(length(unique(train_set[[i]]))))
                  )
}
```

Checking for distinct and missing values

```
# inspecting missing values using the "naniar" package
as.data.frame(output) %>%
    bind_cols(naniar::miss_var_summary(combined_set)) %>%
    select(-variable, -n_miss) %>%
  kable(booktabs = T, linesep="", align = "lllcc", longtable=T,
        col.names = c("Variable", "Data type", "Role", "Distinct values", "Percent missing"),
        cap = "Amount of distinct and missing values") %>%
  kable_styling(latex_options = c("HOLD_position","repeat_header")) %>%
  kable_classic()
```

Table 2: Amount of distinct and missing values

Variable	Data type	Role	Distinct values	Percent missing
I1	integer	Input	1	0
I2	integer	Input	9	0
I3	integer	Input	17	0
I4	integer	Input	17	0
I5	integer	Input	17	0
I6	integer	Input	17	0
I7	integer	Input	17	0
I8	integer	Input	16	0
I9	integer	Input	4	0
I10	integer	Input	16	0
I11	integer	Input	17	0
I12	integer	Input	17	0
I13	integer	Input	17	0

Table 2: Amount of distinct and missing values *(continued)*

Variable	Data type	Role	Distinct values	Percent missing
I14	integer	Input	17	0
I15	integer	Input	17	0
I16	integer	Input	15	0
I17	integer	Input	5	0
I18	integer	Input	17	0
I19	integer	Input	17	0
I20	integer	Input	17	0
I21	integer	Input	17	0
I22	integer	Input	17	0
I23	integer	Input	17	0
I24	integer	Input	9	0
I25	integer	Input	2	0
I26	integer	Input	17	0
I27	integer	Input	17	0
I28	integer	Input	17	0
I29	integer	Input	17	0
I30	integer	Input	17	0
I31	integer	Input	17	0
I32	integer	Input	3	0
I33	integer	Input	2	0
I34	integer	Input	16	0
I35	integer	Input	17	0
I36	integer	Input	17	0
I37	integer	Input	17	0
I38	integer	Input	17	0
I39	integer	Input	15	0
I40	integer	Input	1	0
I41	integer	Input	8	0
I42	integer	Input	17	0
I43	integer	Input	17	0
I44	integer	Input	17	0
I45	integer	Input	17	0
I46	integer	Input	17	0
I47	integer	Input	17	0
I48	integer	Input	5	0
I49	integer	Input	8	0
I50	integer	Input	17	0
I51	integer	Input	17	0
I52	integer	Input	17	0
I53	integer	Input	17	0
I54	integer	Input	17	0
I55	integer	Input	17	0
I56	integer	Input	11	0
I57	integer	Input	2	0

Table 2: Amount of distinct and missing values *(continued)*

Variable	Data type	Role	Distinct values	Percent missing
I58	integer	Input	11	0
I59	integer	Input	17	0
I60	integer	Input	17	0
I61	integer	Input	17	0
I62	integer	Input	17	0
I63	integer	Input	17	0
I64	integer	Input	16	0
target	integer	Target	10	0

- We do observe that some of the inputs have fewer distinct values. 10 inputs have less than 10 unique values.
- Also, the output suggests that the combined data comprising the train set and test set has no missing values.

```
# Heat Map on the Train Data
dat1 <- data.matrix(train_set[order(train_set$target), -65])
n <- NROW(dat1)
color <- rainbow(n, alpha = 0.8)
heatmap(dat1, col=color, scale="column", Rowv=NA, Colv=NA,
labRow=FALSE, margins=c(4,4), xlab="Image Variables", ylab="Samples",
main="Heatmap of Handwritten Digit Data")
```

```
# Heat Map on the Test Data
dat0 <- data.matrix(test_set[order(test_set$target), -65])
n <- NROW(dat0)
color <- rainbow(n, alpha = 0.8)
heatmap(dat0, col=color, scale="column", Rowv=NA, Colv=NA,
labRow=FALSE, margins=c(4,4), xlab="Image Variables", ylab="Samples",
main="Heatmap of Handwritten Digit Data")
```

Removing unary and close to unary columns

A total of 16 input variables were removed due to the fact that they had almost all observations coming from a single unique value.

```
column_ids <- c(1, 8, 9, 16, 17, 24,25, 32, 33, 40, 41, 48, 49, 56, 57, 64)
new_train <- train_set[-column_ids]
new_test <- test_set[-column_ids]
dim(new_train)
```

```
## [1] 3823   49
```

```
dim(new_test)
```

```
## [1] 1797   49
```

1.2 Part (b): Ordinary Principal Component Analysis

```
# excluding target variable
train_inputs <- new_train[-ncol(new_train)] # Train
test_inputs <- new_test[-ncol(new_test)] # Test
```

Parallel boxplots

```
train_inputs %>%
  tidyr::pivot_longer(everything(), names_to="variable", values_to="value") %>%
  ggplot(aes(variable, value, fill=variable)) +
  geom_boxplot() +
  labs(x='Input variables', y='Values') +
  theme(legend.position = "none",
        axis.text.x = element_text(angle = 90, vjust = .5, hjust = 1))
```

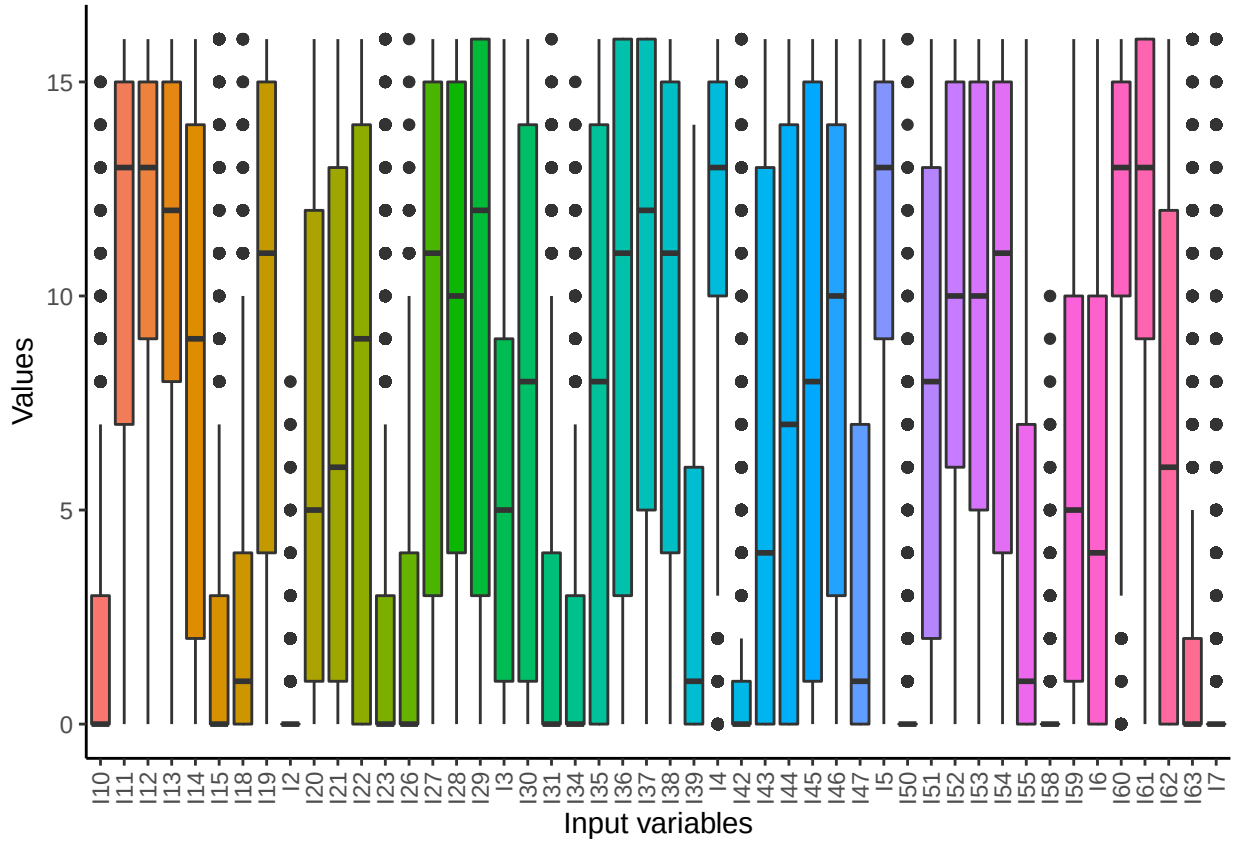

Figure 1: Boxplots of input variables from the training set

From Figure 1, we observe that the input variables show differing range and variations which justifies the standardization of the data as a necessary next step prior to performing the PCA. Thus, we standardize the input variables as follows by first normalizing the inputs from the training set with their respective means and standard deviations and subsequently normalizing the test set with the means and standard deviations computed from the training set:

```r
# Normalize train inputs
scaledTrainInputs <- scale(train_inputs)
scaledTestInputs <- scale(test_inputs, center = attributes(scaledTrainInputs)$`scaled:center`,
                          scale = attributes(scaledTrainInputs)$`scaled:scale`)
scaledTrainInputs <- as.data.frame(scaledTrainInputs)
scaledTestInputs <- as.data.frame(scaledTestInputs)
```

1.2.1 Runing the Ordinary PCA

```r
ord_pca <- prcomp(scaledTrainInputs, retx=TRUE, center=F, scale=F)

# veiw summary of pca results
# summary(ord_pca)
```

1.2.2 Variances explained by the ordinary PCs

```r
par(mfrow=c(1,2))
screeplot(ord_pca, type="lines", main = "", npcs = 20, col="skyblue")
var.pc <- (ord_pca$sdev)^2
prop.pc <- var.pc/sum(var.pc)
plot(prop.pc, type='o',
     ylab="Proportion of variance explained", col="skyblue",
          xlab="Number of princinpal components")
```

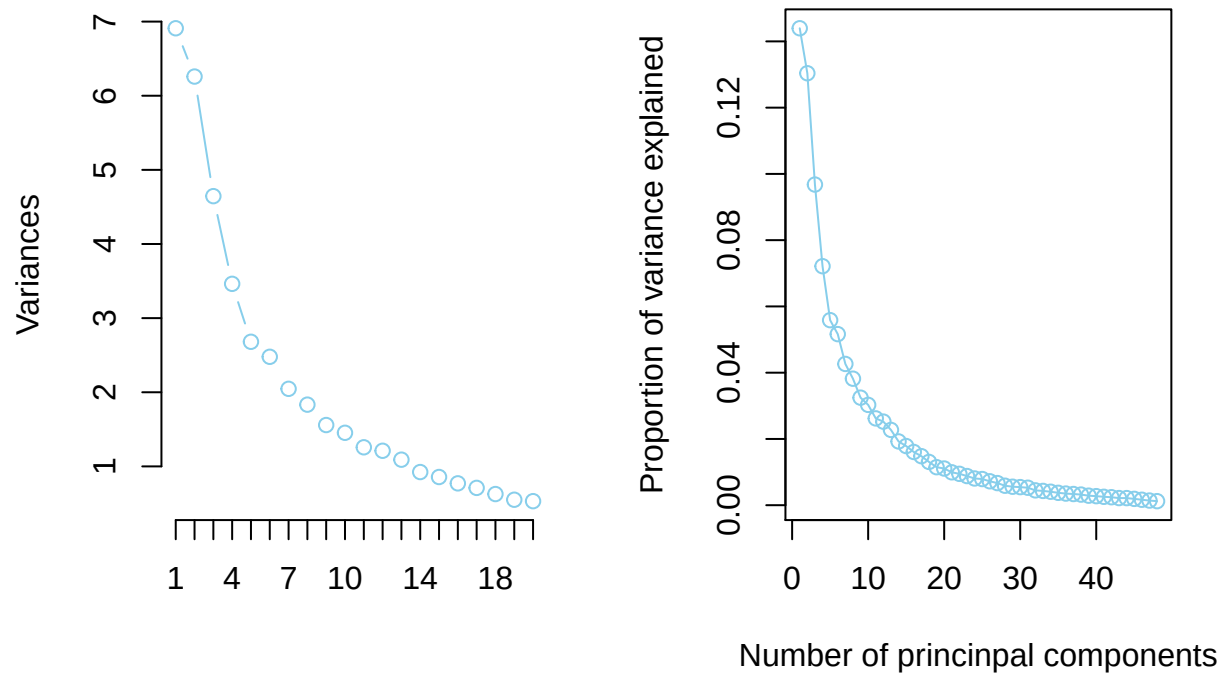

Figure 2: Scree plots depicting the variances (eigenvalues) as well as the proportion of variance explained by the principal components from the ordinary PCA

Looking at the proportion of variance explained from Figure 2 or the variances, we can see that the first two PCs, relative to the other PCs, explain a great amount of the total variation.

```
pca_scatterplot <- data.frame(ord_pca$x[,1:2], target = train_set$target) %>%
    mutate(target = as.factor(target)) %>%
    ggplot(aes(PC1, PC2)) +
    geom_text(aes(label = target, color=target), show.legend = F) +
    geom_vline(xintercept = 0, linetype='dashed', color='grey') +
    geom_hline(yintercept = 0, linetype='dashed', color='grey') +
    labs(x="1st PC", y="2nd PC")
pca_scatterplot
```

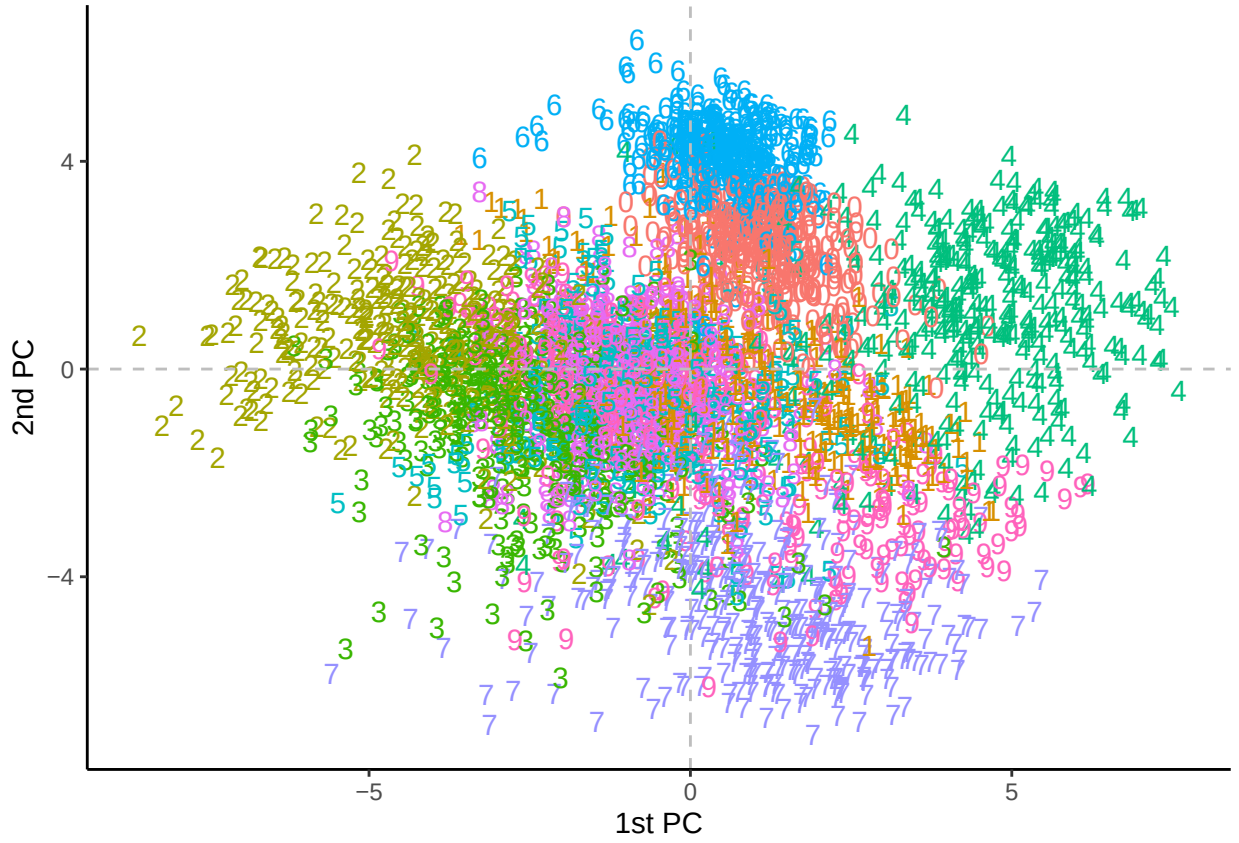

Figure 3: Scatter plot of the first two PCs with the target class variable

Figure 3 shows that the first two PCs try to separate the data into the various 10 handwritten digits. The digits, 2, 4, and 6, appear more separated than the others. Different digits clustered together are most likely to be similar to one another than those that are further apart. For example, the digits 2, 3, 5, and 8, may look alike since they have much overlaps. This supports our observation from the screen plots that the first two components contain substantial information from the data.

1.3 Part (c): Performing Kernel PCA

A polynomial kernel function with degree 1 was chosen to perform the kernel PCA. We were able to arrive at this choice of kernel function after comparing its results to the radial basis kernel

function by trying different degrees and sigma values for the polynomial kernel and radial basis kernel, respectively. The best results from each kernel based on the proportion of variance explained (see figure 4 in this section and figure 11 in the Appendix) were compared, and the kernel used here emerged the best.

```
# kernPCA <- kpca(~., data=scaledTrainInputs, kernel="rbfdot",
#   kpar=list(sigma=0.01), features=ncol(scaledTrainInputs))

kernPCA <- kpca(~., data=scaledTrainInputs, kernel="polydot",
    kpar=list(degree=1), features=ncol(scaledTrainInputs))
```

1.3.1 Variances explained by the kPCs

```
sdev <- sqrt(eig(kernPCA))
names(sdev) <- NULL
par(mfrow=c(1,2))
screeplot(list(sdev=sdev), type="lines", main = "", npcs = 20, col="skyblue"
          )

var.pc <- sdev^2
prop.pc <- var.pc/sum(var.pc)
plot(prop.pc, type='o',
     ylab="Proportion of variance explained", col="skyblue"
     ,xlab="Number of princinpal components")
```

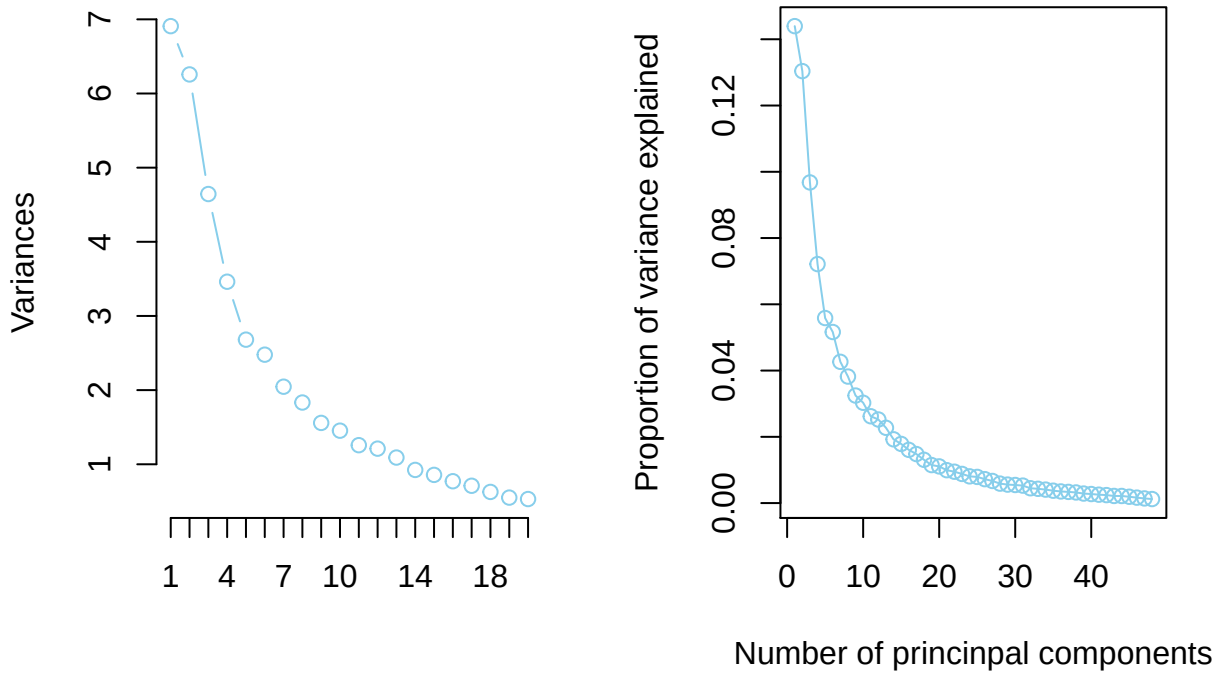

Figure 4: Scree plots depicting the variances (eigenvalues) as well as the proportion of variance explained by the principal components from the kernel PCA.

Just like in the case of the ordinary PCA, the first two principal components appear to explain a large amount of the total variation.

```
kpcs <- rotated(kernPCA)
kpca_scatterplot <- data.frame(kpcs[,1:2], target = train_set$target) %>%
    mutate(target = as.factor(target)) %>%
    ggplot(aes(X1, X2)) +
    geom_text(aes(label = target, color=target), show.legend = F) +
    geom_vline(xintercept = 0, linetype='dashed', color='grey') +
    geom_hline(yintercept = 0, linetype='dashed', color='grey')+
    labs(x="1st kPC", y="2nd kPC")
kpca_scatterplot
```

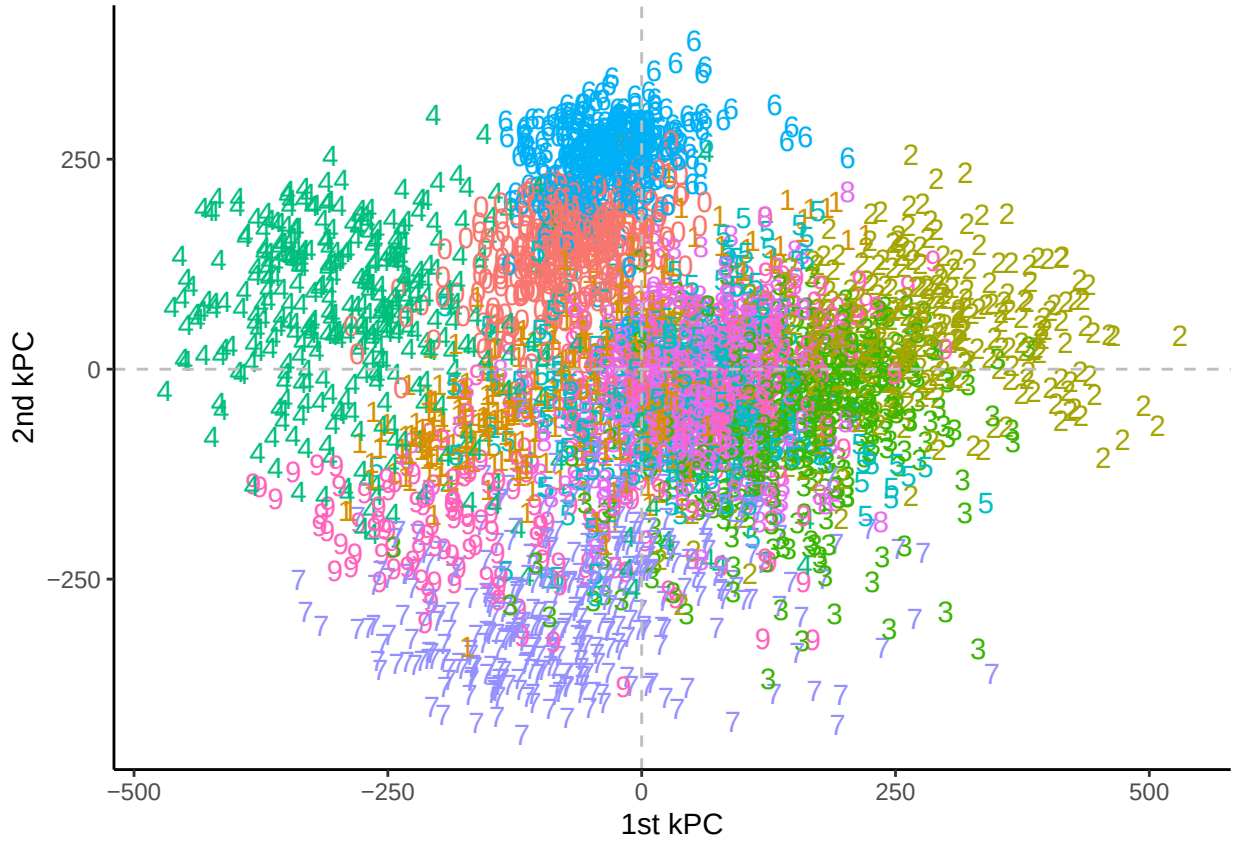

Figure 5: Scatter plot of the first two kernel PCs with the target class variable

The scatter plot in figure 5 is just the mirror image of the one for ordinary PCA in figure 3. This shows that the results from the two PCA methods are identical. Thus, the observations we made earlier about 5 also hold true here that the two kernel principal components hold a good amount of information from the data.

1.3.2 Comparing kPCA results with the PCA results

```
p1 <- pca_scatterplot + ggtitle("Results from ordinary PCA")
p2 <- kpca_scatterplot + ggtitle("Results from kernel PCA")
p1 + p2
```

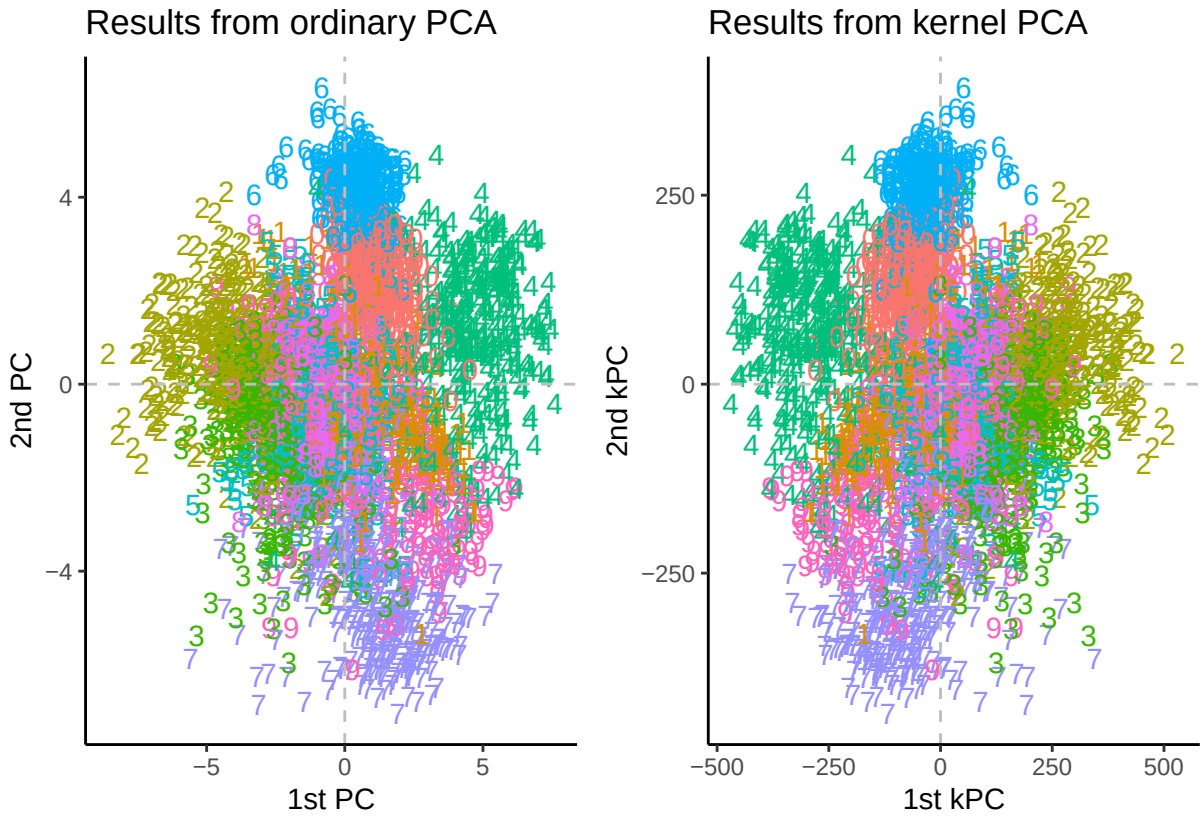

Figure 6: A graph comparing kPCA results with the ordinary PCA results

Clearly, we observe that the results obtained does not differ significantly between the two types of PCA methods performed. The two principal components from the two methods appear to hold almost the same amount of information from the data.

1.4 Part (d): Applying the trained PCA and kPCA to the test set

```
pca_preds <- predict(ord_pca, scaledTestInputs)
pcaplot <- pca_scatterplot +  ggtitle("Based on training set")
pcapred_scatterplot <- data.frame(pca_preds[,1:2], target = test_set$target) %>%
    mutate(target = as.factor(target)) %>%
    ggplot(aes(PC1, PC2)) +
    geom_text(aes(label = target, color=target), show.legend = F) +
    geom_vline(xintercept = 0, linetype='dashed', color='grey') +
    geom_hline(yintercept = 0, linetype='dashed', color='grey') +
    labs(x="1st PC", y="2nd PC", title = "Based on predictions from test set")

pcaplot + pcapred_scatterplot
```

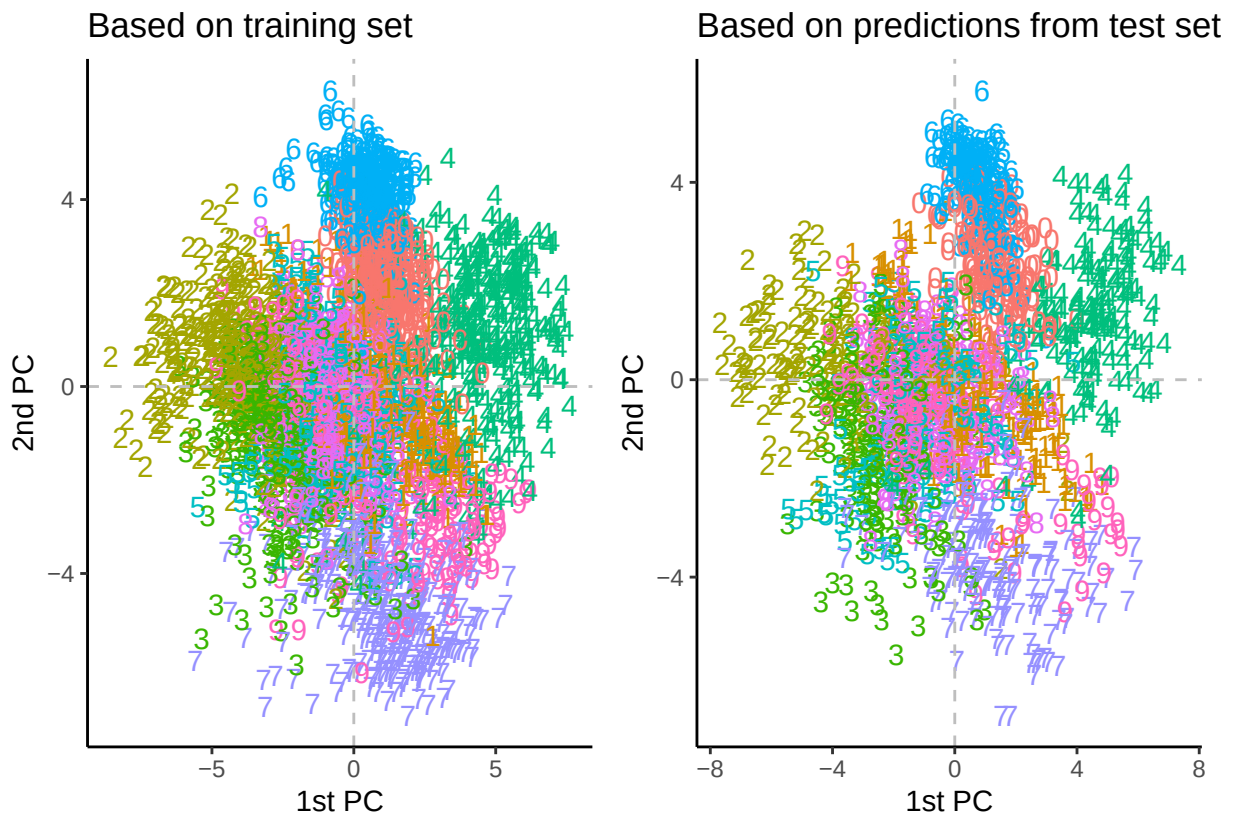

Figure 7: Scatter plot of the first two PCs with the target class variable

From figure 7, except that there are fewer observations in the test set, we can see that the patterns formed by the predicted principal components are identical to the patterns from the trained PCA. This shows how good the results learned from the PCA are.

```
kpca_preds <- predict(kernPCA, scaledTestInputs)
kpcaplot <- kpca_scatterplot +  ggtitle("Based on training set")
kpcapred_scatterplot <- data.frame(kpca_preds[,1:2], target = test_set$target) %>%
    mutate(target = as.factor(target)) %>%
    ggplot(aes(X1, X2)) +
    geom_text(aes(label = target, color=target), show.legend = F) +
    geom_vline(xintercept = 0, linetype='dashed', color='grey') +
    geom_hline(yintercept = 0, linetype='dashed', color='grey') +
    labs(x="1st PC", y="2nd PC", title = "Based on predictions from test set")

kpcaplot + kpcapred_scatterplot
```

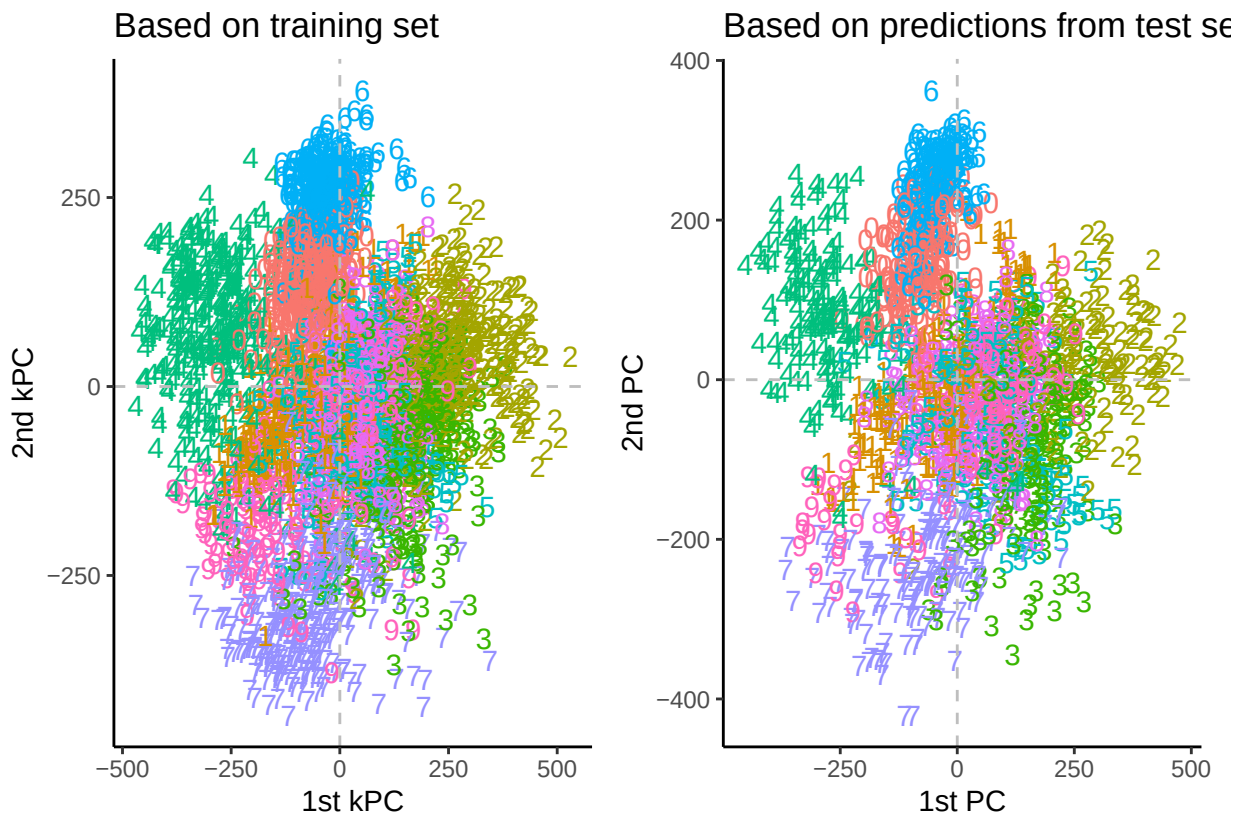

Figure 8: Scatter plot of the first two PCs with the target class variable

Similarly, the two plots in figure 8 are identical, indicating that the kPCA also did a good job.

2 Association Rules

The data used in this section consist of a parsed-version of the King James Bible, with punctuation and stop words removed. You can think of stop words as words such as "the", "a", "is", "are", "be" among others which are so commonly used in a way that they tend to carry very little useful information. As a result, each line is a sentence in the document. Our goal is to ***find words which commonly occur together in sentences***.

From here forward, we shall simply use the word Bible to refer to the King James version of the Bible.

2.1 Part (a): Importing the data

To begin, we use the R function `read.transactions()` available in the `arules` package to import the data into R as a transaction data type. We set the format to "basket" to be able to perform a **Market Basket Analysis** Association mining.

```
library(arules)
bible <- read.transactions(file="http://snap.stanford.edu/class/cs246-data/AV1611Bible.txt",
                           format = "basket", sep =" ", rm.duplicates =F, quote="")
dim(bible)
```

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

```
# itemLabels(bible) # making sure that the items were properly separated
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}
```

The output shows that the first sentence in the Bible begins with the word "beginning", not considering stop words.

We begin our exploration and analysis by presenting a summary of the entire bible transaction data available to us. This gives us useful information about our transaction object.

```r
summary(bible)
```

```
## transactions as itemMatrix in sparse format with
##  31101 rows (elements/itemsets/transactions) and
##  12767 columns (items) and a density of 0.0009591
##
## most frequent items:
##    lord    thou     god    said     thy (Other)
##    6667    3881    3875    3602    3044  359755
##
## element (itemset/transaction) length distribution:
## sizes
##    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16   17
##   13  235  550  840 1554 2258 2536 2611 2428 2465 2283 2139 1925 1751 1490 1248
##   18   19   20   21   22   23   24   25   26   27   28   29   30   31   32   33
## 1084  876  666  574  412  321  258  182  129  107   57   47   22   14    2    8
##   34   35   36   37
##    5    4    5    2
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     2.0     8.0    12.0    12.2    15.0    37.0
##
## includes extended item information - examples:
##      labels
## 1     aaron
## 2   aaron's
## 3 aaronites
```

There are **31101** transactions which gives us a count of the total number of sentences in the Bible and **12767** items denoting the number of words. Note that the "quote" parameter we specified in the `read.transactions()` function to deal with double/single quotes had the effect of reducing the number of items from 13978 to the 12767 that we have.

With a density of `0.0009591`, representing the proportion of non-zero entries, we know that our transaction data is extremely sparse. We learn from this summary output that "**lord**" is the most frequent item with **6667** occurrences, which is about twice the records for the second most frequent item "**thou**". The item "**god**" comes in the third position with **3875** occurrences. We do see that the item set sizes ranges from 2 to 37, with item sets of size 8 as the most prevalent having **2536** occurrences.

2.2 Part (b)

2.2.1 Item Frequency Analysis

To identify the first and last words with their frequencies, the `itemFrequency()` function with type set to "absolute" was used.

```r
itemfreq <- itemFrequency(bible, type="absolute")
first10_items <- head(itemfreq, 10)
last10_items <- tail(itemfreq, 10)
```

```
first10_plt <- data.frame(item = names(first10_items), freq=first10_items) %>%
  ggplot(aes(item, freq, fill = item)) +
  geom_bar(stat = "identity",position = "dodge", show.legend=F) +
  scale_y_continuous(breaks = seq(0, 300, by=50)) +
  labs(x="Words", y="Absolute frequency", title = "First 10 items") +
  theme(axis.text.x = element_text(angle = 90, vjust = .5, hjust = 1),
        plot.title = element_text(hjust = .5))

last10_plt <- data.frame(item = names(last10_items), freq=last10_items) %>%
  ggplot(aes(item, freq, fill = item)) +
  geom_bar(stat = "identity",position = "dodge", show.legend=F)  +
  labs(x="Words", y="Absolute frequency", title = "Last 10 items") +
  theme(axis.text.x = element_text(angle = 90, vjust = .5, hjust = 1),
        plot.title = element_text(hjust = .5))

first10_plt + last10_plt
```

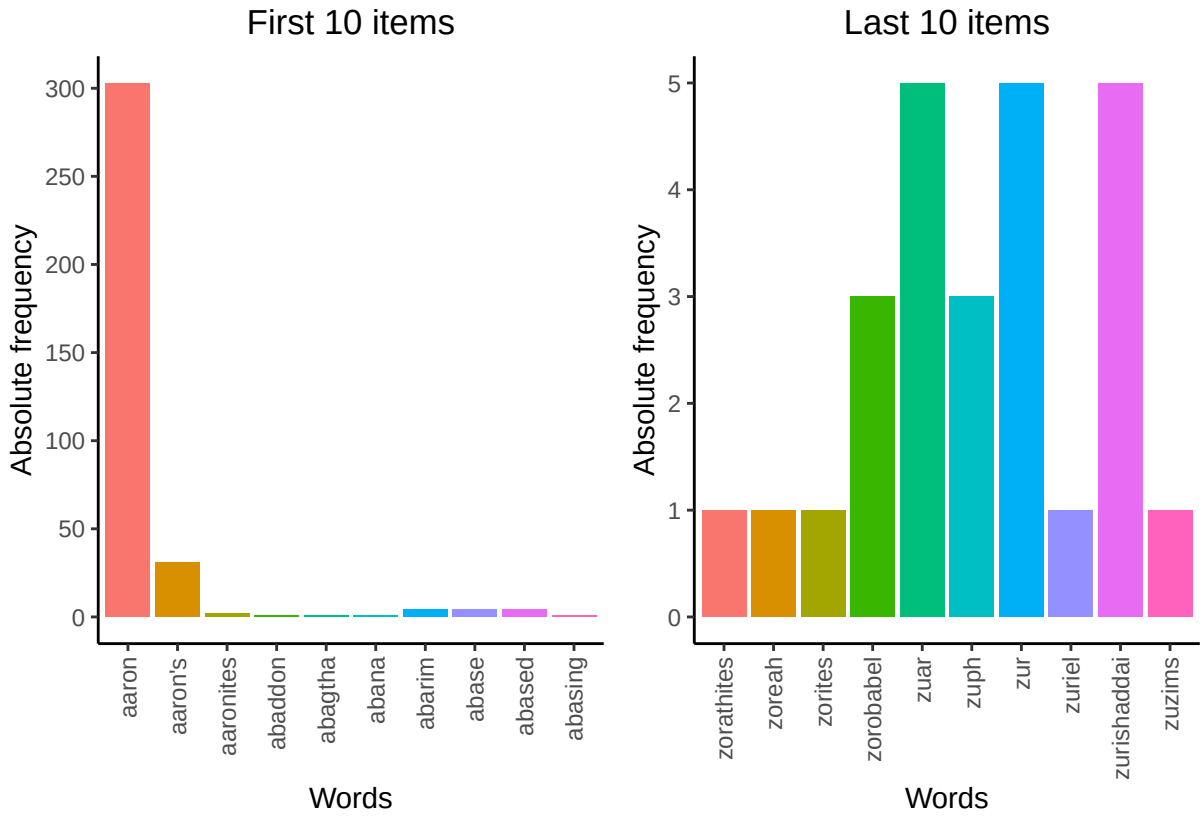

Figure 9: First 10 versus last 10 items with their absolute frequencies

These **absolute frequency** plots present numeric frequencies of each word independently. Looks like our Bible transaction data was sorted in alphabetical order from a - z. Interestingly, all the first 10 words begin with letter "a" while all the last 10 begin with letter "z". Apart from the word "aaron" all the words have very low frequencies; none of the last 10 even went above 10.

Next is a bar graph depicting the top 20 most frequent items. This was obtained by means of the `itemFrequencyPlot()` function available in the `arules` package with **1%** minimum support. The relative frequencies (support) show how many times these words have appeared as compared to others.

```
itemFrequencyPlot(bible, type="relative",  topN = 20, support=0.01,
                  cex.names = 0.8, col="dodgerblue",main="Relative item frequency plot")
```

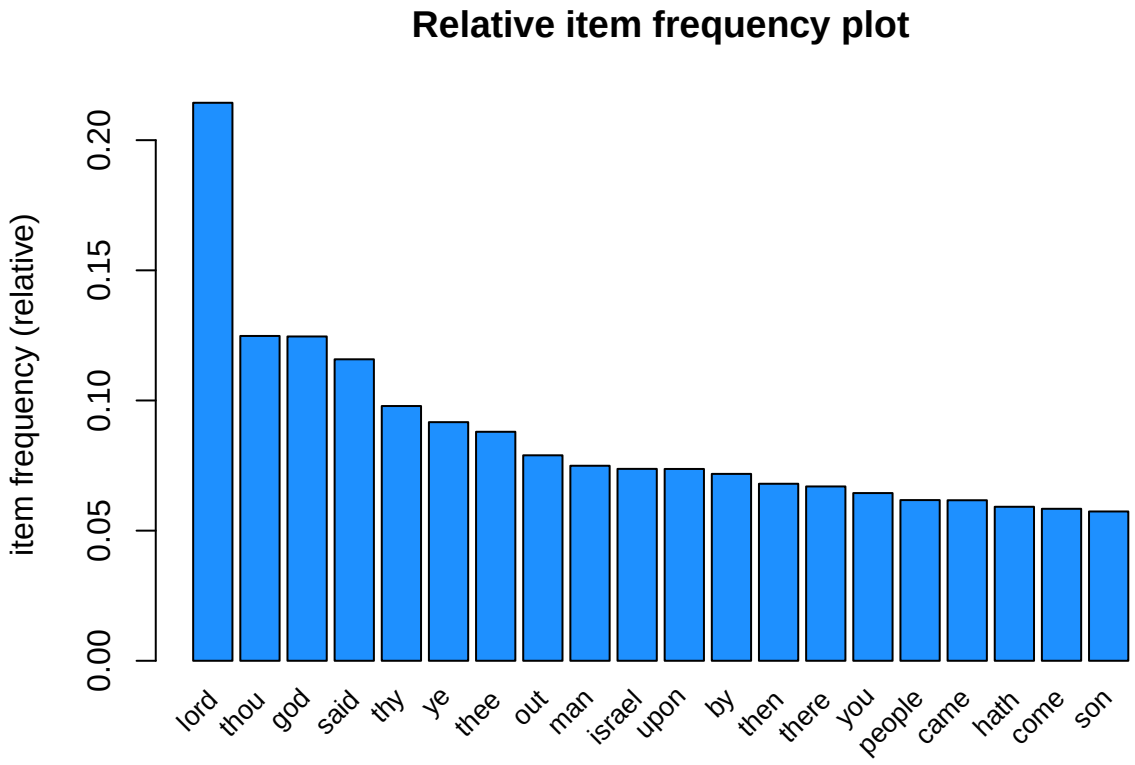

Figure 10: Top 20 most frequent words

Requiring at least 1% support, the word "lord" is the most frequent item from the Bible data under consideration, followed by the words "thou" and "god" with almost the same frequency of occurrences as already observed. The word "son" is the least frequent among the top 20 items. It comes as no surprise for anyone familiar with the Bible to see the words "lord", "god", "israel", and "people" appearing among the most frequent words since a major portion of the Bible concerns the ***God (sometimes refered to as lord)*** of the Bible and his dealings with the ***people*** of ***Israel***. Having a better understanding of the common meanings of these most occurring words, as used in the Bible, would be very beneficial to users and students of the Bible.

2.2.2 Association Rule Analysis

With the help of the **Apriori** algorithm implementation in the `arules` package, we mined association rules from the constructed Bible transaction data using **0.01** minimum support and **0.5** minimum confidence with **5** items as the maximum items in a rule. Varying these parameters, it was observed that a 10% minimum support was too strict to result in zero rules when a 50% minimum confidence was enforced, hence our choice of 1% support with 50% confidence seemed reasonable thresholds

resulting in 29 rules for exploration. Note that lowering the support threshold to **0.001** with the same values for the other parameters gave rise to 3566 rules (too many rules).

```
# obtain the association rules with apriori algorithm
association_rules <- apriori(bible, parameter = list(support = 0.01, confidence = 0.5,
    target = "rules", maxlen=5))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.5    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 [0.12s].
## sorting and recoding items ... [230 item(s)] done [0.01s].
## creating transaction tree ... done [0.01s].
## checking subsets of size 1 2 3 4 done [0.01s].
## writing ... [29 rule(s)] done [0.00s].
## creating S4 object  ... done [0.01s].
```

```
summary(association_rules)
```

```
## set of 29 rules
##
## rule length distribution (lhs + rhs):sizes
##  2  3
## 15 14
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    2.00    2.00    2.00    2.48    3.00    3.00
##
## summary of quality measures:
##     support         confidence       coverage           lift
##  Min.   :0.0106   Min.   :0.501   Min.   :0.0113   Min.   : 2.52
##  1st Qu.:0.0119   1st Qu.:0.576   1st Qu.:0.0155   1st Qu.: 3.34
##  Median :0.0139   Median :0.678   Median :0.0224   Median : 5.12
##  Mean   :0.0155   Mean   :0.726   Mean   :0.0221   Mean   : 6.56
##  3rd Qu.:0.0151   3rd Qu.:0.950   3rd Qu.:0.0263   3rd Qu.: 7.99
##  Max.   :0.0390   Max.   :1.000   Max.   :0.0415   Max.   :22.18
##      count
```

```
##  Min.   : 328
##  1st Qu.: 370
##  Median : 432
##  Mean   : 481
##  3rd Qu.: 471
##  Max.   :1213
##
## mining info:
##   data ntransactions support confidence
##  bible         31101    0.01        0.5
##
##  apriori(data = bible, parameter = list(support = 0.01, confidence = 0.5, target = "rules",
```

- Per our configuration settings for the `apriori` algorithm, a total of 29 rules were obtained split between only two item set of sizes 2 and 3, with a very close representation for both (15 rules of size 2 and 14 rules of size 3).
- The support measures are generally very low with the maximum occurring at 3.9% and a minimum of 1.06% (obviously constrained by our minimum threshold).
- We also obtained the highest possible confidence value of 100%.
- With a minimum lift value of 2.52 (greater than 1), there is likely to be some interesting positive associations among the words in the sentences of the Bible. This tells us that the presence of the *antecedents* (lhs) have positive effects on the corresponding *consequence* (rhs) in our document.

```
# inspect(rules)
rules_df <- as(association_rules, "data.frame")
kable(rules_df[1:5,],booktabs=T,linesep ="", caption="First 5 association rules")%>%
  kable_styling(latex_options = c("HOLD_position")) %>%
  kable_classic()
```

Table 3: First 5 association rules

rules	support	confidence	coverage	lift	count
{answered} => {said}	0.0107	0.6776	0.0158	5.850	332
{art} => {thou}	0.0143	0.9867	0.0145	7.907	446
{word} => {lord}	0.0119	0.5498	0.0216	2.565	370
{moses} => {lord}	0.0149	0.5977	0.0249	2.788	462
{she} => {her}	0.0141	0.6016	0.0234	15.685	438

```
tail(rules_df, 5) %>%
  kable(booktabs=T,linesep ="",  caption = "Last 5 association rules") %>%
  kable_styling(latex_options = c("HOLD_position")) %>%
  kable_classic()
```

Table 4: Last 5 association rules

	rules	support	confidence	coverage	lift	count
25	{god,thy} => {thou}	0.0114	0.5913	0.0192	4.738	353
26	{god,thou} => {thy}	0.0114	0.5065	0.0224	5.175	353
27	{god,thy} => {lord}	0.0134	0.6985	0.0192	3.258	417
28	{lord,thy} => {thou}	0.0153	0.5157	0.0297	4.133	476
29	{god,thou} => {lord}	0.0131	0.5825	0.0224	2.717	406

According to Tables 3 and 4, we can make the following observations:

- The first five rules are all of length or size 2 while the last five rules are all of size 3. Among the last five rules, the word "god" is an inevitable part of the antecedents.
- The last five rules tend to have approximately the same magnitudes of the different association measures, suggesting that those rules appear to carry the same meaning. This makes sense when one considers the fact that "god" and "lord" can be used interchangeably to refer to the the same personality in the Bible, while thou is the archaic form of you and thy is the archaic form of your.
- Among the first five rules we observe that 67.76% of the time that the word "answered" is used it is followed by the word "said" and these two words have positive association with lift 5.850. Similarly, the word "thou" mostly comes after the word "art" 98.67% of the time with a positive correlation (lift of 7.907). The rule, {she} => {her} confirms our English grammar knowledge that the word "she" is the *antecedent* of "her".
- The credibility of the rules generated is somewhat questionable since the rules have fairly large confidence values (at least 50%) and lift values greater than 1, but low levels of support (ranging between 1.06% and 3.9%).

2.3 Part (c): The top 5 rules in decreasing order of confidence (conf) for item sets of size 2 or 3

```
# get the sizes of each rule
new_rules_df <- data.frame(matrix(unlist(strsplit(as.character(rules_df$rules), split="=>")), r
colnames(new_rules_df) <- c("LHS", "RHS")# LHS=Left hand side, RHS= Right hand side.
rule_size <- function(x){length(unlist(strsplit(as.character(x), split=",")))}
sizes <- apply(new_rules_df, 1, rule_size)

# display the top 5 in decreasing order of confidence
data.frame(rules_df, size=sizes) %>%
    filter(size %in% c(2,3)) %>%
    arrange(desc(confidence)) %>%
    dplyr::slice_head(n=5) %>%
  kable(booktabs=T,linesep ="",
    caption = "top 5 rules in decreasing order of confidence for itemsets of size 2 or 3") %>%
  kable_styling(latex_options = c("HOLD_position")) %>%
  kable_classic()
```

Table 5: top 5 rules in decreasing order of confidence for itemsets of size 2 or 3

rules	support	confidence	coverage	lift	count	size
{shalt,thee} => {thou}	0.0127	1.0000	0.0127	8.014	395	3
{shalt,thy} => {thou}	0.0151	1.0000	0.0151	8.014	471	3
{shalt} => {thou}	0.0390	0.9992	0.0390	8.007	1213	2
{lord,shalt} => {thou}	0.0123	0.9974	0.0123	7.993	381	3
{art} => {thou}	0.0143	0.9867	0.0145	7.907	446	2

- There are two rules of size 2 and three rules of size 3.
- It is not surprising that the first two rules have the same confidence and lift values, and almost the same support. This is because the two rules are basically the same, since thee and thy are both the archaic forms of you. By their confidence values, these two rules imply that in 100% of the sentences where **shall** and **thee/thy** are used together, the word **thou** always follows.

2.4 Part (d): Top 5 rules in decreasing order of lift for itemsets of size 2 or 3

```
# display the top 5 in decreasing order of lift
top5_liftrules_df <- data.frame(rules_df, size = sizes) %>%
    filter(size %in% c(2,3)) %>%
    arrange(desc(lift)) %>%
    dplyr::slice_head(n=5)

kable(top5_liftrules_df,booktabs=T,linesep ="",
  caption = "top 5 rules in decreasing order of lift for itemsets of size 2 or 3") %>%
  kable_styling(latex_options = c("HOLD_position")) %>%
  kable_classic()
```

Table 6: top 5 rules in decreasing order of lift for itemsets of size 2 or 3

rules	support	confidence	coverage	lift	count	size
{lord,thus} => {saith}	0.0139	0.8538	0.0163	22.183	432	3
{thus} => {saith}	0.0146	0.6436	0.0227	16.721	455	2
{she} => {her}	0.0141	0.6016	0.0234	15.685	438	2
{pass} => {came}	0.0149	0.5759	0.0259	9.338	463	2
{shalt,thee} => {thou}	0.0127	1.0000	0.0127	8.014	395	3

- There are three rules of size 2 and two rules of size 3. The top two rules have some meaning even without other supporting words in the sentences where they occurred.
- Having the highest lift value of **22.183**, the rule {lord,thus} => {saith} is the rule with the highest strength of correlation or association within the Bible document.
- The rule {she} => {her} shows up here because there should obviously be high association between their usage based on our grammatical knowledge. It is however surprising that we don't have the rule {he} => {his/him}.

2.5 Part (e): Conviction measures for the top 5 rules identified in part (d)

By our own design, the top 5 lift rules in part (d) is a data frame, so we first extract it's corresponding rules object from the entire association rules object that we have and then use the resulting object in the `interestMeasures()` function to obtain the desired conviction measures. We had to remove the sixth observation because there is a tie between the last rule in part (d) and the rule **{shalt, thy} => {thou}** according to their lift values.

```
# subset the corresponding rules object from the entire association rules object
top5_liftrules <- subset(association_rules, subset =
(size(lhs) + size(rhs) %in% top5_liftrules_df$size) & lift %in% top5_liftrules_df$lift)

top5_liftrules <- sort(top5_liftrules, decreasing = T, by='lift') # to maintain the order
#inspect(top5_liftrules) # 6 because there is a tie between {shalt, thee} => {thou} and
                  # {shalt, thy} => {thou}
conviction <- interestMeasure(top5_liftrules[-6, ], c( "conviction"), transactions=bible)

data.frame(top5_liftrules_df, conviction = conviction) %>%
  kable(booktabs=T,linesep ="",
  caption = "Conviction measures for the top 5 rules identified in part (d)") %>%
  kable_styling(latex_options = c("HOLD_position")) %>%
  kable_classic()
```

Table 7: Conviction measures for the top 5 rules identified in part (d)

rules	support	confidence	coverage	lift	count	size	conviction
{lord,thus} => {saith}	0.0139	0.8538	0.0163	22.183	432	3	6.575
{thus} => {saith}	0.0146	0.6436	0.0227	16.721	455	2	2.698
{she} => {her}	0.0141	0.6016	0.0234	15.685	438	2	2.414
{pass} => {came}	0.0149	0.5759	0.0259	9.338	463	2	2.212
{shalt,thee} => {thou}	0.0127	1.0000	0.0127	8.014	395	3	Inf

Just like the lift values, all the conviction values are greater than 1 indicating that these are interesting rules. Note that the conviction for the last rule is ∞ (`Inf`) because the confidence value is `1`, signifying logical implication.

Finally, we explain how conviction aids in avoiding the problems associated with both the confidence and the lift measures.

As observed by Azevedo and Jorge (2007), conviction is roughly speaking, the best predictive measure of association rules. Unlike confidence, conviction favors low frequency classes and produces different rule orderings. Also, compared to lift, conviction is not a symmetric measure, which means it can account for the significance of rule direction. That is, conviction is capable of discriminating between the strength of the rules A => B and B => A.

3 Appendix

```
kernPCA <- kpca(~., data=scaledTrainInputs, kernel="rbfdot",
    kpar=list(sigma=0.01), features=ncol(scaledTrainInputs))
```

```
sdev <- sqrt(eig(kernPCA))
names(sdev) <- NULL
par(mfrow=c(1,2))
screeplot(list(sdev=sdev), type="lines", main = "", npcs = 20, col="skyblue")

var.pc <- sdev^2
prop.pc <- var.pc/sum(var.pc)
plot(prop.pc, type='o',
     ylab="Proportion of variance explained", col="skyblue"
     ,xlab="Number of princinpal components")
```

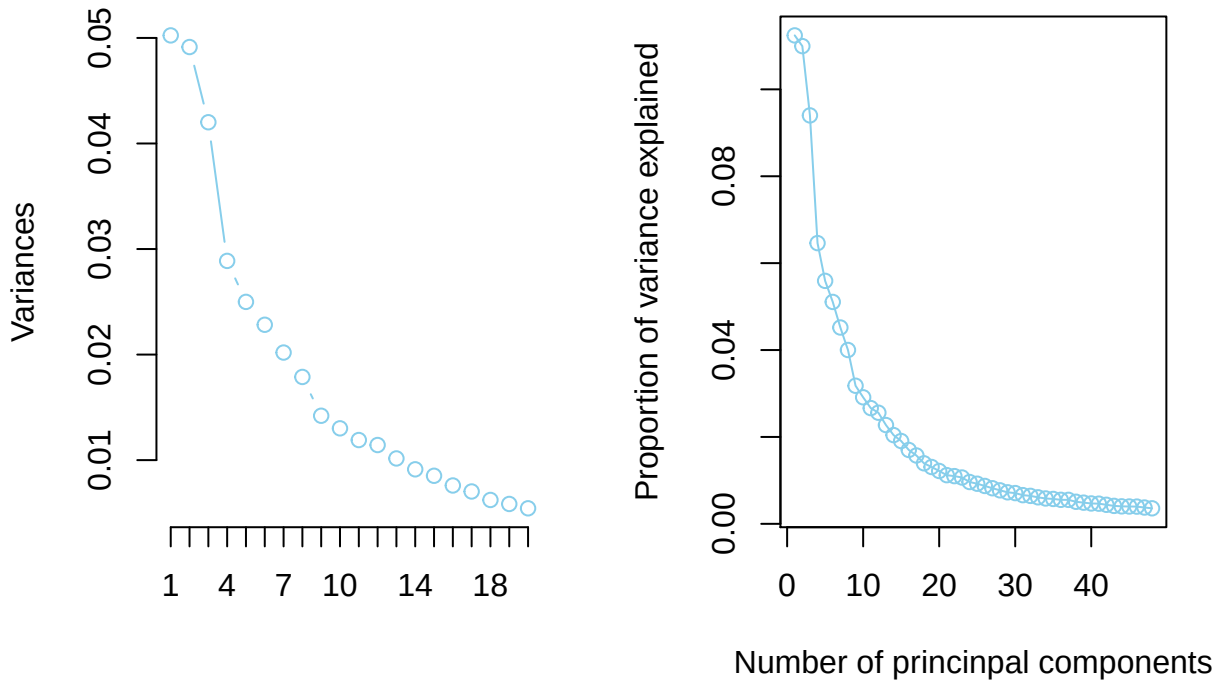

Figure 11: Scree plots depicting the variances (eigenvalues) as well as the proportion of variance explained by the principal components from the kernel PCA with radial basis kernel function.

References

- Azevedo, P. J., & Jorge, A. M. (2007, September). Comparing rule measures for predictive association rules. In European Conference on Machine Learning (pp. 510-517). Springer, Berlin, Heidelberg.
- Optical recognition of handwritten digits, available from UCI Machine Learning Repository: https://archive.ics.uci.edu/ml/machine-learning-databases/optdigits/

- Association Rules' Data retrieved from: http://snap.stanford.edu/class/cs246-data/AV1611Bible.txt