PART I:

1. ***Suppose we are interested in estimating the number of home runs based on other numerical variables in the dataset. Use Principal Component Analysis on all numerical predictors. How many components should be extracted?***

To be able to run a principal component analysis, the first set taken was to standardise the numerical variables. I then create a test and train data to be able to validate my model (90% training, 10% test.

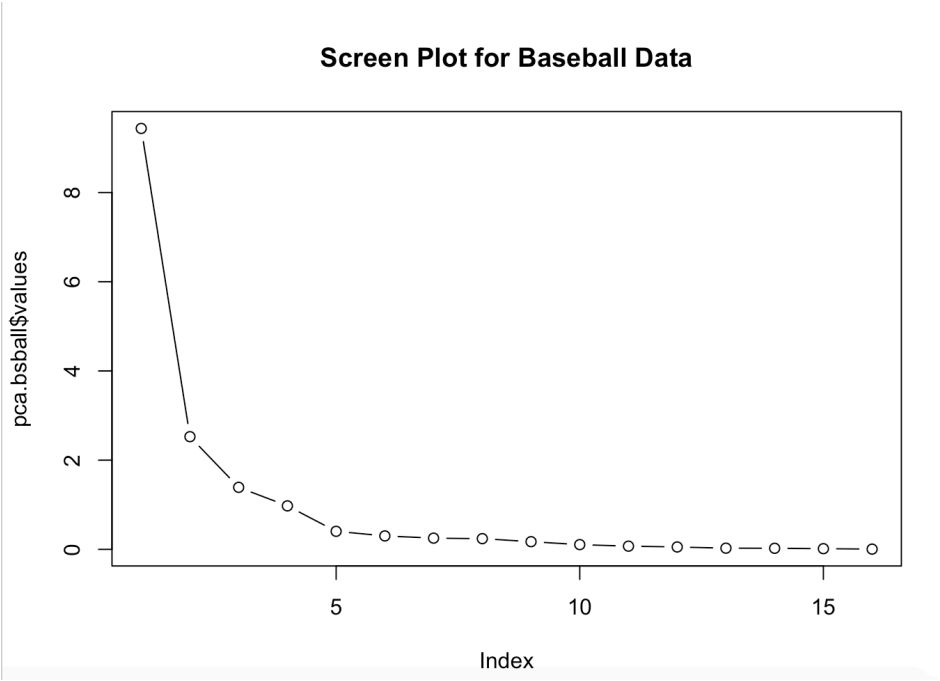
```
Loadings:
                PC1    PC2    PC3    PC4    PC5    PC6    PC7    PC8    PC9    PC10   PC11   PC12   PC13   PC14   PC15   PC16
age              0.143  0.142 -0.248  0.943
games            0.955 -0.125                -0.102        -0.109                      -0.117
at_bats          0.974 -0.110                              -0.112
runs             0.980                                                                              -0.113
hits             0.974                                     -0.126
doubles          0.944                       -0.153        -0.101  0.103                0.213
triples          0.643 -0.163  0.569        -0.392  0.191  0.169
homeruns         0.855        -0.378 -0.144  0.132         0.173  0.106         0.157
RBIs             0.941        -0.265
walks            0.891        -0.190         0.219  0.137        -0.214 -0.135 -0.140
strikeouts       0.899 -0.114 -0.147                             -0.207  0.300
bat_ave          0.158  0.937  0.194                      -0.135                0.122         0.139
on_base_pct      0.302  0.901  0.137                             -0.187                      -0.146
slugging_pct     0.428  0.819                              0.243  0.190  0.127 -0.145
stolen_bases     0.624 -0.170  0.600  0.144  0.294  0.250 -0.127  0.176
Caught_stealing  0.659 -0.187  0.550  0.117  0.183 -0.389  0.166

                PC1   PC2   PC3   PC4   PC5   PC6   PC7   PC8   PC9  PC10  PC11  PC12  PC13  PC14  PC15  PC16
SS loadings    9.437 2.526 1.390 0.975 0.404 0.301 0.252 0.240 0.172 0.106 0.072 0.053 0.027 0.024 0.015 0.005
Proportion Var 0.590 0.158 0.087 0.061 0.025 0.019 0.016 0.015 0.011 0.007 0.004 0.003 0.002 0.002 0.001 0.000
Cumulative Var 0.590 0.748 0.835 0.896 0.921 0.940 0.955 0.970 0.981 0.988 0.992 0.996 0.997 0.999 1.000 1.000
```

The initial loadings are seen in the image above. From this we can see that all variables except age load the highest in the first factor. One can also see that only this factor explains 60% of the variability by itself.

To be able to determine the ideal amounts of components to be extracted, my initial guess was 4. With such a value, more than 85.5% of the variance is explained and from there on the additional variance decreases rapidly actually. Furthermore, when looking at the upper panel we can see that with 4 factors, each variable has a communality over the 50% threshold.

```
> pca.bsball$values
 [1] 9.43704426 2.52604352 1.39014053 0.97488452 0.40446231 0.30129377 0.25155004 0.24006607
 [9] 0.17194061 0.10642078 0.07175875 0.05317983 0.02685641 0.02407863 0.01511656 0.00516342
```

The next possible test was the eigenvalue test. The above output confirms the initial hypothesis that only the first 4 factors should be taken as they are the only factors that explain more than one variable's worth of variability (value is greater than 1).



Finally, when looking at the above plot, one can see that the elbow is at 5, indicating that we should not retain more than 5 factors and therefore, as the other test seem to indicate 4, the ideal number of components to be extracted in 4.

```
Loadings:
                PC1    PC2    PC3    PC4
age              0.185  0.200  0.821  0.104
games            0.934 -0.175  0.104  0.115
at_bats          0.980
runs             0.979
hits             0.988
doubles          0.961
triples          0.622 -0.156 -0.206  0.672
homeruns         0.877               -0.414
RBIs             0.975         0.122
walks            0.768         0.509  0.142
strikeouts       0.918         0.114 -0.280
bat_ave          0.175  0.890 -0.245  0.112
on_base_pct            0.912         0.133
slugging_pct     0.364  0.748 -0.122 -0.130
stolen_bases     0.724 -0.103 -0.541
Caught_stealing  0.675        -0.493

                 PC1   PC2   PC3   PC4
SS loadings    9.406 2.321 1.643 0.828
Proportion Var 0.588 0.145 0.103 0.052
Cumulative Var 0.588 0.733 0.836 0.887
```

I then validated my hypothesis by running a PCA with 4 components on the test data and as can be seen above, 88.7% of the variability explained.

2. Write up a short profile of the components extracted in the first question. Is it appropriate to use PCA for this analysis? Why?

```
Loadings:
                PC1    PC2    PC3    PC4    PC5    PC6    PC7    PC8    PC9    PC10   PC11   PC12   PC13   PC14   PC15   PC16
age              0.143  0.142 -0.248  0.943
games            0.955 -0.125               -0.102        -0.109                      -0.117
at_bats          0.974 -0.110                             -0.112
runs             0.980                                                                          -0.113
hits             0.974                                    -0.126
doubles          0.944                      -0.153        -0.101  0.103                0.213
triples          0.643 -0.163  0.569        -0.392  0.191  0.169
homeruns         0.855        -0.378 -0.144  0.132         0.173  0.106         0.157
RBIs             0.941        -0.265
walks            0.891        -0.190         0.219  0.137        -0.214 -0.135 -0.140
strikeouts       0.899 -0.114 -0.147                             -0.207  0.300
bat_ave          0.158  0.937  0.194                    -0.135                0.122         0.139
on_base_pct      0.302  0.901  0.137                           -0.187                      -0.146
slugging_pct     0.428  0.819                            0.243  0.190  0.127 -0.145
stolen_bases     0.624 -0.170  0.600  0.144  0.294  0.250 -0.127  0.176
Caught_stealing  0.659 -0.187  0.550  0.117  0.183 -0.389  0.166

                 PC1   PC2   PC3   PC4   PC5   PC6   PC7   PC8   PC9  PC10  PC11  PC12  PC13  PC14  PC15  PC16
SS loadings    9.437 2.526 1.390 0.975 0.404 0.301 0.252 0.240 0.172 0.106 0.072 0.053 0.027 0.024 0.015 0.005
Proportion Var 0.590 0.158 0.087 0.061 0.025 0.019 0.016 0.015 0.011 0.007 0.004 0.003 0.002 0.002 0.001 0.000
Cumulative Var 0.590 0.748 0.835 0.896 0.921 0.940 0.955 0.970 0.981 0.988 0.992 0.996 0.997 0.999 1.000 1.000
```

As seen in the image above, the first component is the most important one. It explains about 60% of the variance and is composed of the variables such as games, at_bats, rus, hits, doubles, homeruns, RBI's, walks and strikeouts. Seeing as it incorporates so many of the variables it makes sense that it has such high variance explanation.

PC2 on the other hand explains about 16% of the variance and is mainly composed of bate_ave, on_base_pct and slugging_pct.

PC3 explains 8.7% of the variance and is mainly composed of triples, stolen bases and caught_stealing.

Finally, PC4 explains 6% of the data but is mainly composed of age. This makes sense as age was seen to be very independent to the other variables and thus it makes sense that this variable has its "own" component.

In this case, PCA is an appropriate technique to use. First, all the variables expect for age seem to be correlated, therefore dimension reduction seems appropriate. Furthermore, the first PCA only explain 60% of the variation, leaving enough variability to work with and perform PCA. Finally, when checking for the normality of the variables, there does not seem to be any skewness.

PART II:

3. Develop a C5.0 model to classify the loan decision without misclassification cost.

Before being able to run such a tree, I had to change the levels of the training and testing dataset to make sure the approval3 column matches (one had 3 – disapproved and the other 3 – denied). The next step was the standardize the columns (only the interest column had to be) and then the C50 tree could be built using only the debt to income ration, FICO score and request amount to predict the approval3 column. The interest was removed as it is very correlated to the request amount. The following is the line of code to build such a tree:

```
nocostmodel <- C5.0(Loans_Training[,3:5],as.factor(Loans_Training$Approval3))
```

I then used a confusion matrix and the testing dataset to test how well the tree worked. As can been seen below, it was a success as the accuracy was of 98.95% and the dataset is relatively balanced (there is the more or less same amount of case 1, 2 and 3). Furthermore, the p-value is low meaning the model seems to be significant.

```
Confusion Matrix and Statistics

                  Reference
Prediction          1 - Denied 2 - Approve Half 3 - Approve Whole
  1 - Denied             15566              131                 0
  2 - Approve Half         121            13847               114
  3 - Approve Whole          0              157             19762

Overall Statistics

               Accuracy : 0.9895
                 95% CI : (0.9885, 0.9904)
    No Information Rate : 0.3999
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.984
 Mcnemar's Test P-Value : NA
```

4. ***Derive a cost matrix from the scenario above and develop a C5.0 model using the derived cost matrix.***

Direct Cost Matrix

		Reference		
		1	2	3
Predicted	1	0	0	0
	2	6713,5	-3021	0
	3	13427	3692,5	-6042

Opportunity Cost Matrix

		Reference		
		1	2	3
Predicted	1	-13427	3021	6042
	2	0	3021	3021
	3	0	0	0

Direct Cost Matrix

		Reference		
		1	2	3
Predicted	1	-13427	3021	6042
	2	6713,5	0	3021
	3	13427	3692,5	-6042

Ajusted Cost Matrix

		Reference		
		1	2	3
Predicted	1	0	16448	19469
	2	6713,5	0	3021
	3	19469	9734,5	0

Final Cost Matrix (div by 3021)

		Reference		
		1	2	3
Predicted	1	-	5,44	6,44
	2	2,22	-	1,00
	3	6,44	3,22	-

The above is the development of the cost matrix. The final one was then inserted into R

```
> confusionMatrix(c50cost.pred, Loans_Test$Approval3)
Confusion Matrix and Statistics

                  Reference
Prediction         1 - Denied 2 - Approve Half 3 - Approve Whole
  1 - Denied            15561              125                 0
  2 - Approve Half        126            13866               124
  3 - Approve Whole         0              144             19752

Overall Statistics

               Accuracy : 0.9896
                 95% CI : (0.9886, 0.9904)
    No Information Rate : 0.3999
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9842
 Mcnemar's Test P-Value : NA
```

After having built the model, (c50cost <- C5.0 (x=Loans_Training[,3:5], as.factor(Loans_Training$Approval3), costs = cost.matrix), I tested it compared to the test data and the above results show how each observation is classified. Once again, the accuracy is very high, 95.96% and the p-value is very low.

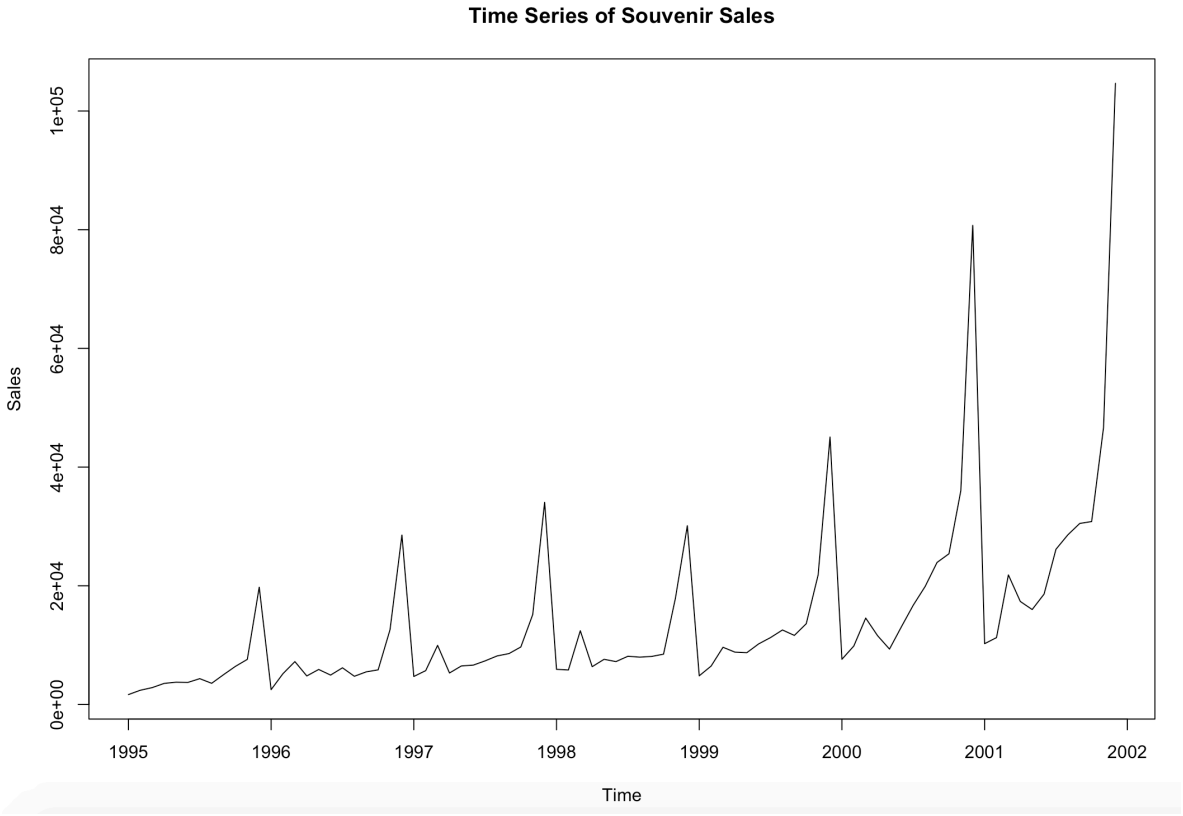
5. What is the increase/decrease in revenue when the cost matrix is incorporated?

No Cost Model						
		Reference				
		1	2	3		
Predicted	1	15566	131	0		
	2	121	13847	114		
	3	0	157	19762	**Total Revenue**	$ 159.841.735,00
With Cost Model						
		Reference				
		1	2	3		
Predicted	1	15569	125	0		
	2	126	13866	124		
	3	0	144	19752	**Total Revenue**	$ 159.853.149,00
					Difference in Revenue	$ 11.414,00

The above shows the difference in revenue between a no cost and cost. Implementing the cost matrix will allow the company to save about 11 000$

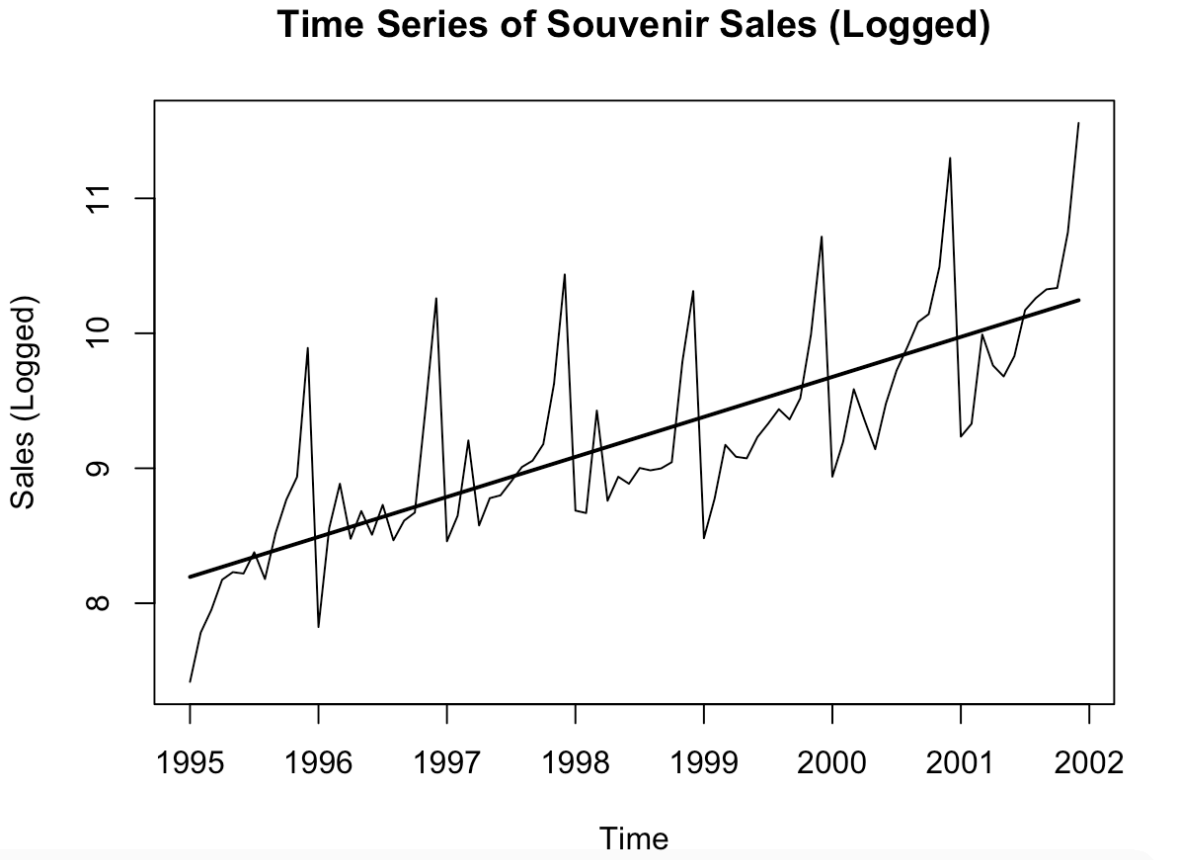
PART III:

6. Create a well-formatted time plot of the data.



The above plot shows the time series of the sales of souvenirs. There seems to be some yearly seasonality, which makes sense as the souvenir sales probably depend on the number of tourists that are present. Furthermore, there seems to be a positive trend probably indicating more and more tourists come to buy souvenirs.

7. Change the scale on the x-axis, or on the y-axis, or on both to log-scale in order to achieve a linear relationship. Select the time plot that seems most linear.

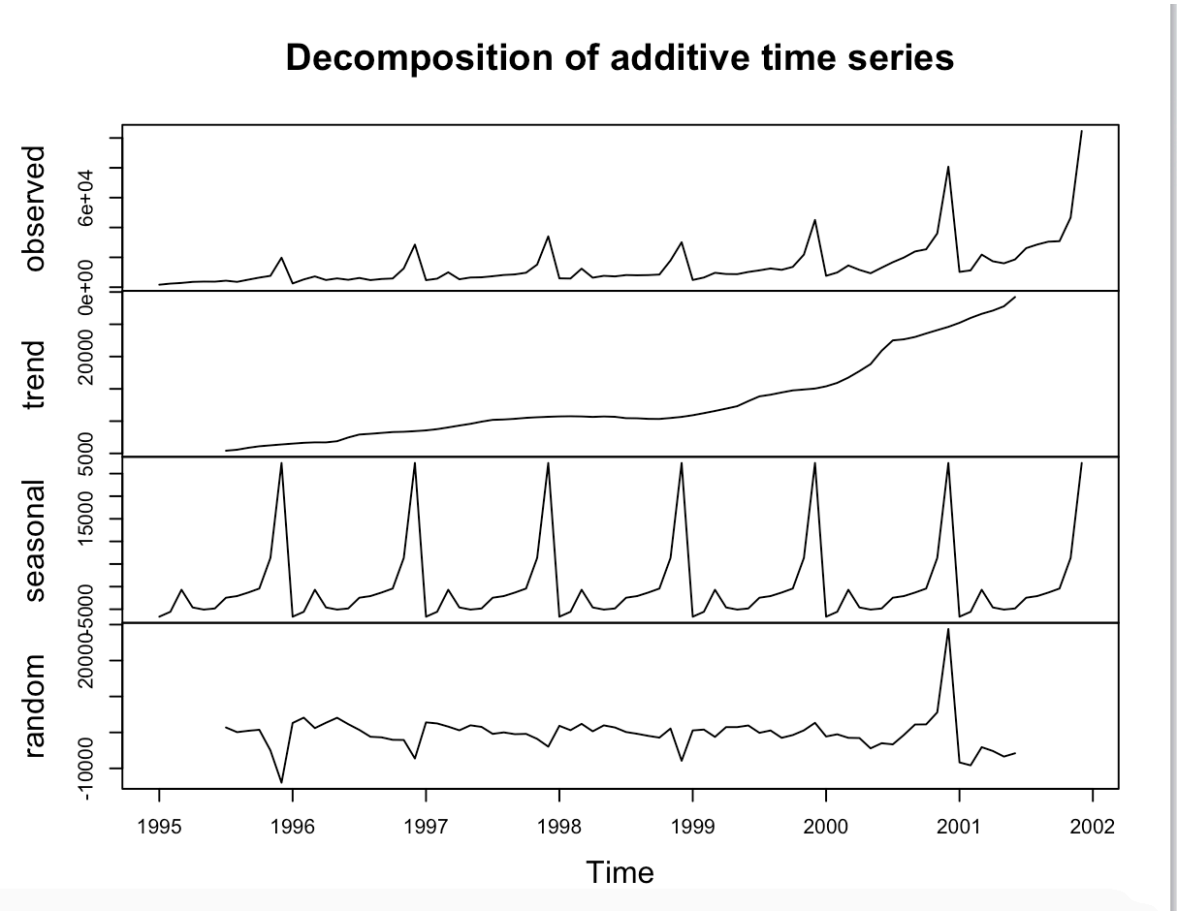


This time plot shows the logged sales for souvenirs since 1995. After having logged the values, the relationship went from looking exponential to looking linear.

8. If the goal is forecasting sales for the next 12 months (year 2002), should we partition the data into the training set and test set? If so, how? If not, why?

Yes, you should partition your data. Like in most cases it is always a good idea to be able to test your model on unseen data to know how it will perform. However, this is a time series issue so the data split cannot be random.

The data needs to be trimmed into two periods: early periods for the training and later periods for the testing. The key part however is that in the testing data there must be enough time to cover a whole period if the data shows signs of seasonality.



As the sales of souvenirs do show signs of yearly seasonality (see above graph), then we must make sure that there is one year of data in the testing (for example 2001). The test data should represent the last year.

However, before doing the forecasting, the test set should be recombined and then the model should be rerun before predicting the values of 2002.

Finally, when splitting the data, I to do the forecasting I would perform a test to confirm the exponential trend that the sales seems to show to make sure we predict right.

PART IV:

9. ***Create a term-document matrix and a concept matrix. Limit the number of concepts to 20. Is the term- document matrix spares or dense? Find two non-zero entries and briefly interpret their meaning.***

To be able to perform such matrices, the first step was the transform the text into a corpus. Then the "–" were replaced to by spaces and then came the pre-processing. I removed white spaces, punctuation, numbers, stem words and finally stemming. Once that was done the document term matrix was created.

When analysing the TDM, I got a sparsity of 100%. This is extremely high and means that most of the cells are empty and that most terms being kept do not appear in most of the documents.

When looking at 2 non-zero entries, we get:

- The term free appears 3 times in document 2055
- The term rabbit appears 2 times in document 2603

10. ***Using logistic regression, develop a model to classify the ads as relevant or non-relevant based on the matrices developed above. Comment on its efficacy.***

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 585  62
         1 188 823

               Accuracy : 0.8492
                 95% CI : (0.8311, 0.8661)
    No Information Rate : 0.5338
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.6939
 Mcnemar's Test P-Value : 2.664e-15

            Sensitivity : 0.7568
            Specificity : 0.9299
         Pos Pred Value : 0.9042
         Neg Pred Value : 0.8140
             Prevalence : 0.4662
         Detection Rate : 0.3528
   Detection Prevalence : 0.3902
      Balanced Accuracy : 0.8434

       'Positive' Class : 0
```

After having ran a logistic regression to predict the adds as relevant or not, using the LSA, the result of the confusion matrix is above. There is a very high accuracy, 84.92% and the p-value is low, indicating the model is significant. Finally, the fact that the positive class is 0 means that the model is very good at predicting when adds are not significant.