Clustering and Randomised Algorithms

*Statistic for Computer Science*

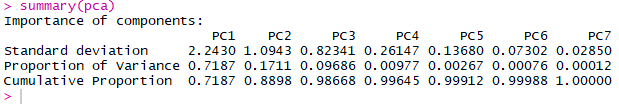
# Part 1

## Section 1

### Task 1

1. PCA results using the first 7 attributes:





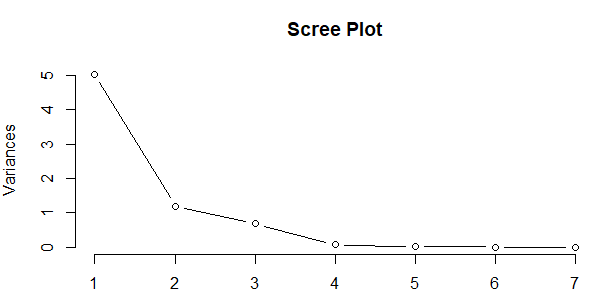
1. Eigenvalues for each of the attributes in descending order:



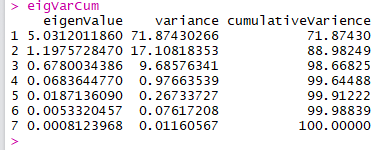
|  |  |
| --- | --- |
| **PC1** | 5.0312011860 |
| **PC2** | 1.1975728470 |
| **PC3** | 0.6780034386 |
| **PC4** | 0.0683644770 |
| **PC5** | 0.0187136090 |
| **PC6** | 0.0053320457 |
| **PC7** | 0.0008123968 |

1. Scree plot using the eigenvalues found above:





1. After performing the PCA the 7 original scaled attributes have been combined to form 4 leaner combinations or Principal Components. However, the first Principal Component is generally the most important as it explains most of the data, followed by Principal Component 2 then 3 and so on. Using the summary function for the PCA we get 3 values; the standard deviation, proportion of variance and the cumulative proportion. The standard deviation is the standard deviation of the data along a single principal component; a measure of the variability across that principal component. The proportion of variance represent the proportion of all the variability in the original data explained by the principal component for instance PC1 explains 71.87% of the data followed by PC2 which explains 17.11%, followed by PC3 which explains 9.69%. The cumulative proportion of PC1 explains 71.87% of the data, PC1 and PC2 combined explain 88.98%, PC1, PC2 and PC3 combined explain 98.67% of the data. This goes on until a 100% of the data is explained after combining all 7 principal components. All this is represented in the scree plot above. The table below shows the eigenvalue, percentage of variance and percentage of cumulative variance for each principal component.



From this table it is apparent that the first principal component is the best one as it explains the data the most from all 7 principal components.

The following table show the first 10 rows of the data set with the scores from the first 2 principal components in the last 2 columns.

## Section 2

### Task 1

Clustering is the process of partitioning a data set into homogeneous groups based on given features such that similar objects are kept in a group while dissimilar objects are placed in a different group. It is the most important unsupervised learning problem as it deals with finding structure in a collection of unlabelled data.

There are many clustering algorithms that can be used in any given scenario. The most popular ones include k-means, hierarchical clustering, fuzzy c-means and mixture of gaussians.

**K-means** is perhaps the most common algorithm used because it is a simple and easy way to classify a given data set through a certain number of clusters. The main idea is to define k centres, one for each cluster. These centres should be placed in a calculated way since different location cause different results. The better choice is to place them as far away as possible from each other. The next step is to take each point belonging to a given data set and associate it to the nearest centre. When no point is pending, the first step is complete, and an early group age is done. At this point we re-calculate k new centroids as barycentre of the clusters resulting from the previous step. After we have these k new centroids, a new binding must be done between the same data set points and the nearest new centre. Here a loop is generated and as a result we notice that the k centres change their location step by step until no more changes are done. This is a randomised clustering algorithm as the centres are randomly selected.

K-means is Fast, robust and easier to understand. It is Relatively efficient and Gives best result when data set are distinct or well separated from each other. However, it requires specification of the number of clusters. it is not invariant to non-linear transformations therefore with different representation of data we get different results. The Euclidean distance measures can unequally weight underlying factors. This algorithm is Unable to handle noisy data and outliers and it fails for non-linear data set

**Hierarchical** clustering is an algorithm that builds a hierarchy of clusters. This algorithm starts with all the data points assigned to a cluster of their own. Then two nearest clusters are merged into the same cluster. The algorithm terminates when there is only a single cluster left. This is a non-randomised clustering algorithm. An advantage to hierarchical clustering is that it does not require information about the number of clusters that are needed. It is also easy to implement and gives best result in some cases. On the downside you can never undo what was previously done and a time complexity of at least O(n2 log n) is required (n being the number of data points). Moreover, no objective function is directly minimized and sometimes it is difficult to identify the correct number of clusters by the dendrogram.

Differences between the above algorithms:

* Hierarchical clustering can’t handle big data well unlike K Means clustering since the time complexity of K Means is linear (O(n)) while that of hierarchical clustering is quadratic (O(n2))
* Since K Means clustering starts with a random choice of clusters, the results produced by running the algorithm multiple times might differ. While results in Hierarchical clustering are reproducible
* K Means works well when the shape of the clusters is hyper spherical (2D circle, 3D sphere)
* K Means requires prior knowledge of K i.e. no. of clusters you want to divide your data into. But, you can stop at whatever number of clusters you find appropriate in hierarchical clustering by interpreting the dendrogram

The **fuzzy c-means** clustering algorithm works by assigning membership to each data point corresponding to each cluster centre based on distance between the cluster centre and the data point. The closer the data is to the cluster centre, the closer is its membership towards that particular cluster centre. Fuzzy c-means allows one piece of data to belong to two or more clusters. This method is frequently used in pattern recognition. Fuzzy c-means gives best result for overlapped data set and is comparatively better then k-means algorithm. Unlike k-means where data point must exclusively belong to one cluster centre here a data point is assigned membership to each cluster centre as a result of which a data point may belong to more than one cluster. Disadvantages to this algorithm include the fact that the number of clusters need to be specified. Also, with lower value of β (the termination criterion between [0, 1]) we get better results but at the expense of more number of iteration. The Euclidean distance measures can unequally weight underlying factors.

### Task 2

K-means clustering is a type of unsupervised learning, used when you have unlabelled data (data without defined categories or groups). The goal of the algorithm is to find groups in the data, with the number of groups represented by the variable K. The algorithm works iteratively to assign each data point to one of K groups based on the features provided. Data points are clustered based on feature similarity. The results of the K-means clustering algorithm are the centroids of the K clusters, which can be used to label new data and labels for the training data (each data point is assigned to a single cluster). Clustering allows you to find and analyse the groups that have formed organically. Each centroid of a cluster is a collection of feature values which define the resulting groups. Examining the centroid feature weights can be used to qualitatively interpret what kind of group each cluster represents.

In the k-mean algorithm the first thing to do is to randomly initialize k points, also called means. Then categorize each item to its closest mean and update the mean’s coordinates, which are the averages of the items categorized in that mean so far. This process is repeated for a given number of iterations and at the end, we get the clusters.

K points are called means because they hold the mean values of the items categorized in it. To initialize these there are many options. One method is to initialize the means at random items in the data set. Another method is to initialize the means at random values between the boundaries of the data set. The following is the pseudocode for the process explained above:

initialize k with randomly;

for (a number of iterations){

foreach(iterate through items){

find the mean closest to the item;

assign item to mean;

update mean;

}

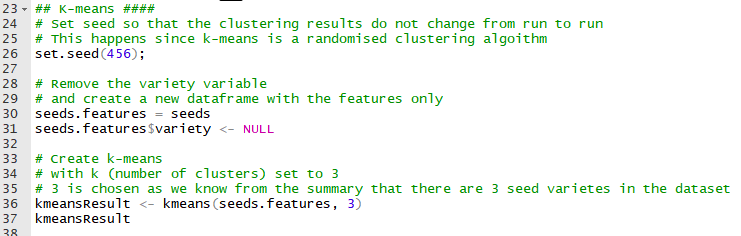
}

K-means algorithm works well when the data fits the cluster model where clusters are spherical; the data points in a cluster are centred around that cluster and where the spread/variance of the clusters is similar therefore each data point belongs to the closest cluster. If any of these principles does not hold then the local minimal of the k-means algorithm will be counter-intuitive, which is an issue that can occur when using this algorithm.

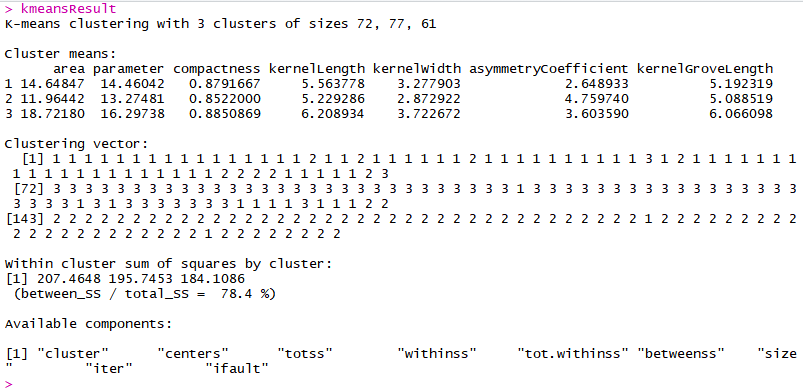
To find the value of K, or the number of clusters in the data, the K-means clustering algorithm must be run for a range of K values and then the results are compare. A common used metrics to compare results of K is the mean distance between data points and their cluster centroid. Since increasing the number of clusters will always reduce the distance to data points, increasing K will hence decrease this metric, until zero is reached when K is the same as the number of data points. Thus, this metric cannot be used as the sole target. Instead, mean distance to the centroid as a function of K is plotted and the "elbow point," where the rate of decrease sharply shifts, can be used to roughly determine K. Other techniques for validating K include cross-validation, information criteria, the information theoretic jump method, the silhouette method, and the G-means algorithm.

The regular k-means algorithm will be used for the dataset. Using R libraries, the K-means algorithms can be initialised and used on any data set. In R studio, we can import and scale the dataset and implement the k-means function on the data.

### Task 3



The screenshot above shows the k-means implementation to the seeds data set. Every step of the implementation is justified and described by comments. The following screenshot is the output of ‘kmeansResult’.

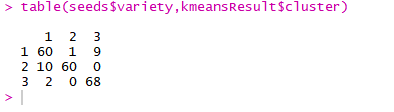


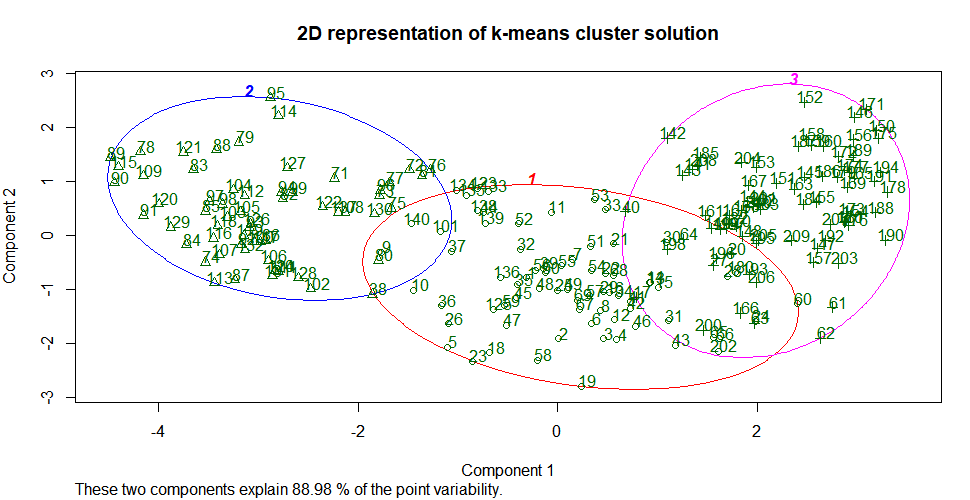
These results where obtained using a set seed of 456 and with k = 3, the predefined number of clusters. The dataset has a total of 120 instances which got divided as follows: 72 in cluster 1, 77 in cluster 2 and 61 in cluster 3.

The results also show the mean value for each attribute in each cluster, for instance, the mean area in cluster 1 if 14.65 while in cluster 2 it is 11.96 (the full results are found within the orange box).

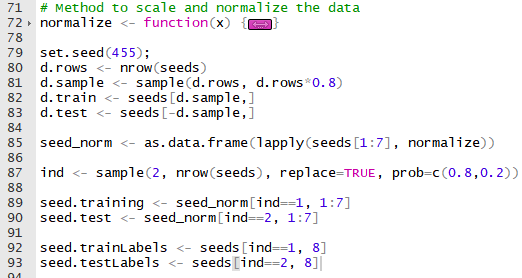
The next section (within the blue box) is a representation of all the observations where the first 16 instances belongs to the first cluster a the 17th belongs to the second cluster. These may not be the actual results as this k-means algorithm is not always accurate.

### Task 4



The table above shows how the clustering algorithm did in comparison to the actual result. The columns represent the 3 clusters while the rows represent the seed varieties. As we can see the algorithm wasn’t accurate in any cluster as there is some overlap. This can be seen better in the following visualisation.

The accuracy of the cluster based on the table is calculated by adding the true positive results and dividing that by the total number of observation (in this case 210); **((60 +60 +68) / 210) \* 100 = *89.5%***



After normalising the data, setting a seed value to 455 and splitting the dataset into 80% training and 20% testing, the k-means clustering algorithm was applied. First it was calculated on the training data, then on the testing data which is based on the result of the training set.

|  |  |  |
| --- | --- | --- |
|  | K-means Code | Matrix Result |
| Training Data |  |  |
| Testing Data |  |  |

**Confusion Matrix on the testing data results:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Original Matrix | | | | |
|  |  | **Clusters** | | |
| **Varieties** |  | **1** | **2** | **3** |
| **1** | 11 | 2 | 2 |
| **2** | 2 | 12 | 0 |
| **3** | 1 | 0 | 13 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Confusion Matrix on Variety 1 | | | | |
|  |  | **Clusters** | | |
| **Varieties** |  | **1** | **2** | **3** |
| **1** | TP | FN | FN |
| **2** | FP | TN | TN |
| **3** | FP | TN | TN |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Confusion Matrix on Variety 2 | | | | |
|  |  | **Clusters** | | |
| **Varieties** |  | **1** | **2** | **3** |
| **1** | TN | FP | TN |
| **2** | FN | TP | FN |
| **3** | TN | FP | TN |

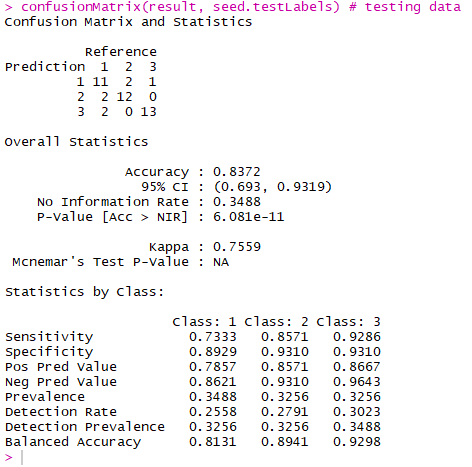
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Confusion Matrix on Variety 3 | | | | |
|  |  | **Clusters** | | |
| **Varieties** |  | **1** | **2** | **3** |
| **1** | TN | TN | FP |
| **2** | TN | TN | FP |
| **3** | FN | FN | TP |

*The sum of all TPs (correct classifications) is 11+12+13 =* ***36***

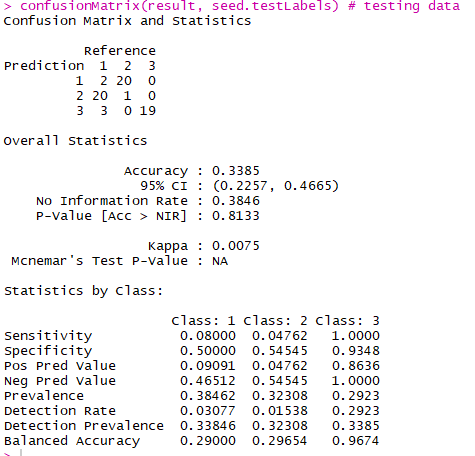
*The total number of classifications is* ***43***

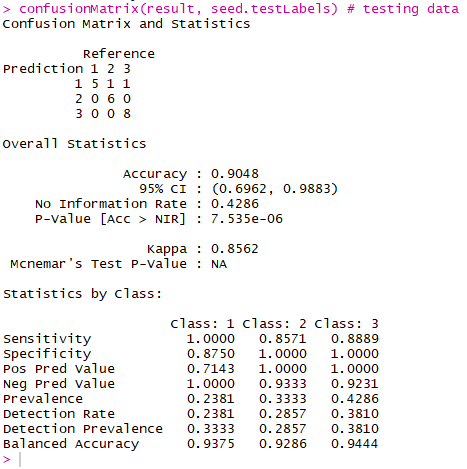
|  |  |  |  |
| --- | --- | --- | --- |
|  | Sensitivity | Specificity | overall Accuracy |
| *formula* |  |  |  |
| Variety 1 |  | *tN = 12+ 0 + 0 + 13 = 26* |  |
| Variety 2 |  | *tN = 11+ 2 + 1 + 13 = 27* |  |
| Variety 3 |  | *tN = 11+ 2 + 2 + 12 = 27* |  |

Apart from the statistics shown above, there are other methods to evaluate cluster results. This can be done through libraries and function is R studio such as the library “caret” which calculates not only overall statistics like the accuracy but also by class, in this case by varieties. This was applied to the clusters calculated previously (on testing data) and the results can be found in the screenshot below.



The above predictions where made on a training set of 80% and a testing set of 20%. For further testing the process was repeated with the data set still into 70% and 30% and again splitting it into 90% and 10%.

Using the same code as above, only changing the splitting ratio to 70% : 30%, the result of the confusion matrix changed a lot, as can be seen in the screenshot to the right. The accuracy went from 84% to 34%. However, it is important to note that many variables affect the answers. For instance, the seed was set to 455 for every scenario, but it that were to change the results will also show a dramatic change. K-means was also set an ‘nstart’ of 750, which when changed causes a huge change. 455 and 750 where chosen after a lot of trial and error. Many other values where tested but the best results where obtained using those exact values.

As mentioned the script was also run using a ratio of 90% : 10% and the results are pasted below. From these tests we can see that by increasing the size of training data set, more precise predictions can be made using the testing data. In fact, the overall accuracy is the best one yet with a value of 90%.

# Part 2

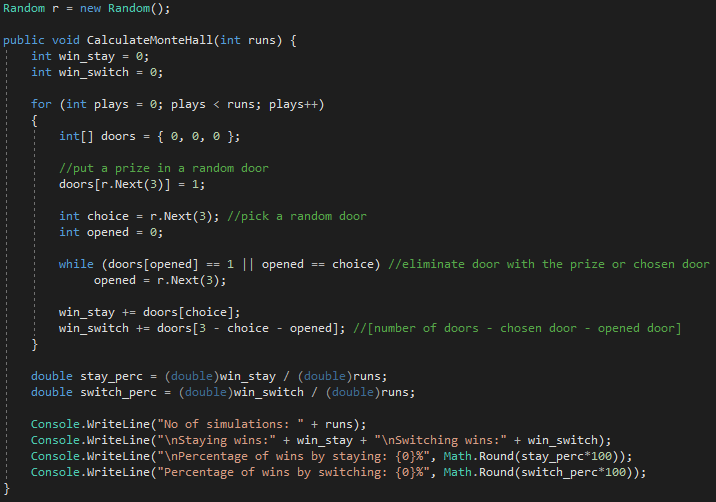
## Section 1

### Task 1

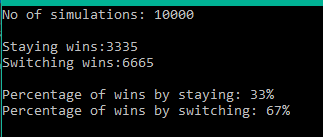
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Chosen Door | Prize Location | Opened Door | Switch / Stay |  | Total Possible Choices | Switch | Stay |
| 1 | 1 | 2 or 3 | Stay | **9** |  |  |
| 1 | 2 | 3 | Switch |
| 1 | 3 | 2 | Switch |
| 2 | 1 | 3 | Switch |
| 2 | 2 | 1 or 3 | Stay |
| 2 | 3 | 1 | Switch |
| 3 | 1 | 2 | Switch |
| 3 | 2 | 1 | Switch |
| 3 | 3 | 1 or 2 | Stay |

Since there are 3 doors to chose from, the initial possibility of finding a prize is 1/3. After chosen a door the player is left with a 2/3 chances that the prize is in the remaining doors. The gameshow host opens one of those remaining doors, and behind it there will never be the prize. At this point there is still a 1/3 chance that the prize is behind the chosen door, but the remaining door now inherits the 2/3 chances of finding the prize. With this logic the player should always switch the door since s/he is twice as likely to find the prize.

### Task 2

**Monte Hall Game Implementation:**

**Results after running the simulation 10,000 times:**



### Task 3

Using the techniques presented in the Monte Carlo Algorithms section of the assignment, find the variance of the random variable representing the Monte Hall problem (with a 1 representing a car being found, 0 otherwise).

Use this variance to find the sample size for a RMSE error of 0.1.

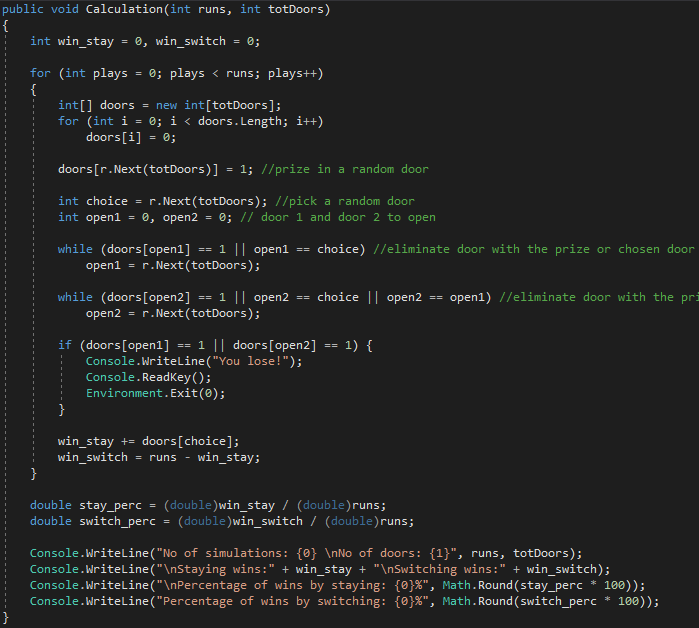
What is the estimated probability of finding a car, using this sample size and both strategies? (2 marks)

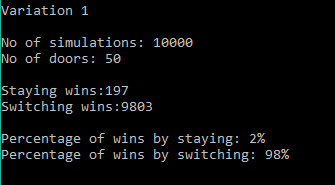
Ans

## Section 2

### Task 1

#### A

**Implementation of Variation 1:**

 **Results using 50 doors, after 10000 simulations:**

#### B

Plot the probability of winning the car, if the player uses the optimal strategy, for n = 3 to 8, where n is the number of boxes. The probabilities error must have a RMSE of 0.1. (3 marks)

|  |  |  |
| --- | --- | --- |
| No of Doors | Switching Wins | % |
| 3 | 6643 | 66 |
| 4 | 7376 | 74 |
| 5 | 8008 | 80 |
| 6 | 8373 | 84 |
| 7 | 8527 | 85 |
| 8 | 8758 | 88 |

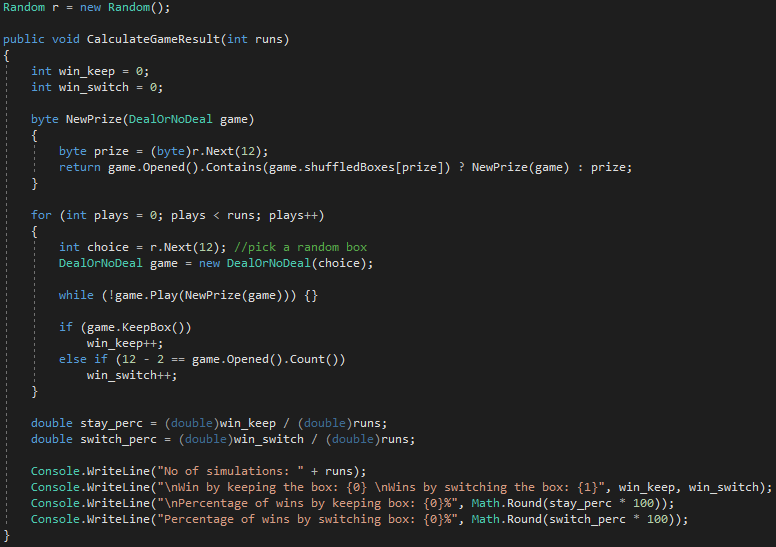
#### C

Find the smallest value of n, such that the probability of winning the car, using an optimal strategy, is smaller than 0.2. (1 mark)

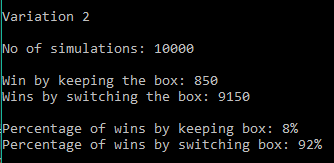
Ans

### Task 2

#### A

**Implementation of Variation 2:**

**Results after 10000 simulations:**



#### B

Using your implementation, perform a simulation of the problem and identify whether an optimal strategy exists and if it does, what is the optimal strategy? Show how you arrived at your solution. (3 marks)

Ans

#### C

Using the optimal strategy found, if any, model the system as a random variable. Find the expected winnings of the game to a RMSE of €1. (3 marks)

Ans

# References