Analytics_assignment3

Question 1 : EDA

a)

Table 1: The dimensions of the data set

Dimensions
5000
2

```
'data.frame':   5000 obs. of  2 variables:
 $ V1: int  624474 673412 647442 532283 646529 647535 644131 521368 688940 592666 ...
 $ V2: int  837604 735362 677000 741384 742844 755101 777721 736923 798967 805244 ...
```

Table: The structure of the data set

Table 2: The number of missing values in each column

	x
V1	0
V2	0

Table 3: A summary of the data statistics

V1	V2
Min. : 89604	Min. : 9597
1st Qu.:370419	1st Qu.:362698
Median :509386	Median :494896

V1	V2
Mean :502998	Mean :497113
3rd Qu.:637848	3rd Qu.:631829
Max. :932954	Max. :977215

```
[1] 0
```

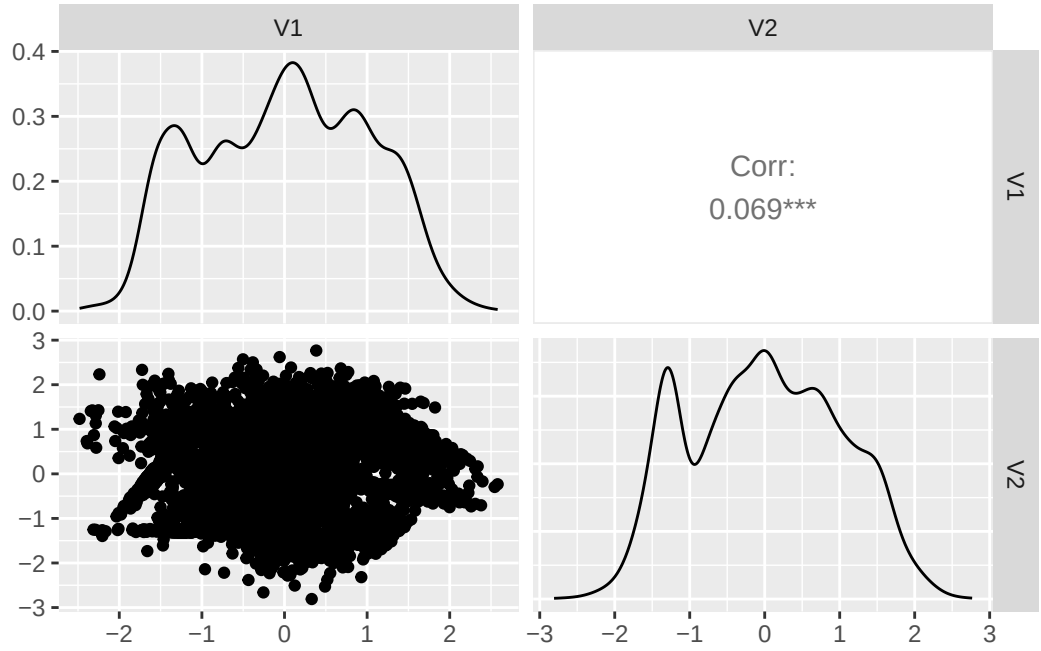
The dataset contains 5000 observations of 2 numeric variables (i.e. values are all integers) . There are no missing values in the data set. Variable 1 values ranges from 89606 to 932954. The median (509386) is slightly above the mean (502998), which hints at a very mild left-skew (a few low values pulling the mean down). For Variable 2, the mean (497113) is very slightly above the median (494896), suggesting a slight right-skew (a few high values pulling the mean up). In both cases, the medians near their means and they have very similar IQRs, indicating the core of the distribution is roughly symmetric.

b)

The variables are of continous nature and have fairly comparable scales besides variable 2 having a much wider range. Therefore, this as well as there being symmetry in the bulk of the data means that Euclidean distances on standardised values will be better.

DON'T PUT CODE HERE?

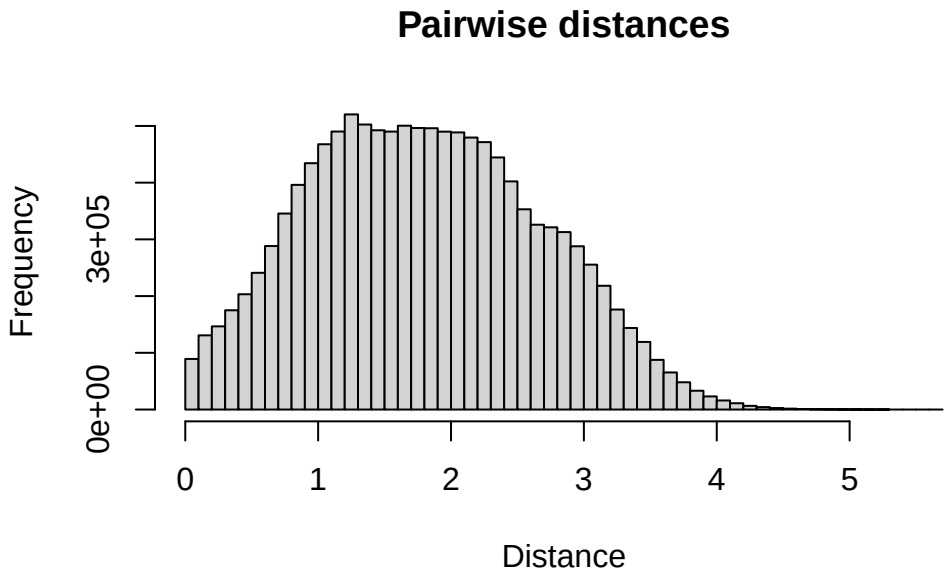
c)



We've already standardized the data set as a result of choosing the euclidean distance as our metric in the previous question, so it is just a linear rescaling of the axes and won't really change the shape of the graphs/plots.

The marginal density for V1 is clearly multi-modal, with at least two distinct "humps" separated by shallow troughs. One peak below zero and a larger one around +0.5 to +1.0 which suggests there are two or even three natural subgroups in V1. In contrast, V2's density is largely uni-modal but can be said to be right-skewed, with most observations clustered near the center and a long tail stretching out past +2. When you plot these together, the scatterplot forms a roughly elliptical, symmetric cloud centered at (0,0) with no clear diagonal stretch and as expected the correlation is nearly zero, so points group in vertically aligned bands corresponding to V1's modes which is not clearly shown in the plot. V1's multiple peaks point to splitting clusters along that axis, while V2 adds spread but not distinct clusters, which justifies using a Euclidean distance on the standardized data and expecting roughly spherical clusters.

d)



The distance histogram clealry shows a single, broad peak centered around 1–2.2. This suggests the typical distance between any two observations once both variables are on the same scale. It suggests that if you pick two random data points, they're most likely to be about two standard deviations apart overall. The bars taper off into a long right-hand tail reaching out to about 5, which indicates a handful of point-pairs that are far apart (either true outliers or members of very distinct subgroups). At the extreme left, the histogram shape indiactes that there are a few nearly zero distances corresponding to observations that are virtually identical on V1 and

V2, meaning they would always merge immediately in a hierarchical tree or sit in the same k-means cluster. Altogether, this moderate spread, neither all bunched up or wildly dispersed, suggests clustering algorithms will have enough contrast to pull apart dense regions.

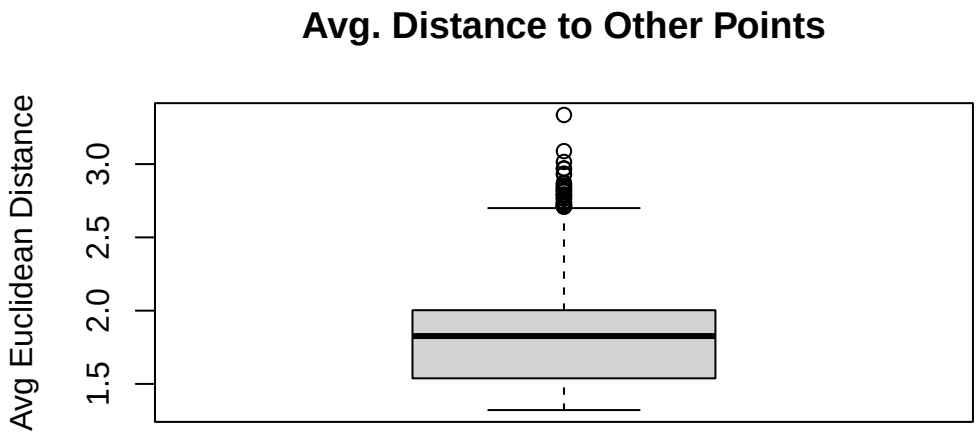
e)

We will compute the average distance for each observation i to all other points in the distance matrix:

```
#####USING AVG DIST AND 2 STAND DEV TO GET OUTLIERS
avg_d   <- rowMeans(as.matrix(dist_matrix))
mu      <- mean(avg_d)
sigma   <- sd(avg_d)
outliers <- which(avg_d > mu + 2*sigma)       # those beyond 2 SD above the mean
outliers_top10 <- head(order(avg_d, decreasing=TRUE), 10)

#####USING BOXPLOTS HERE, NOT SURE WHICH WAY TO DO IT
# 1. Compute each point's average distance
d_mat     <- as.matrix(dist(df_scaled))
avg_d     <- rowMeans(d_mat)

# 2. Boxplot the average distances
boxplot(avg_d, main="Avg. Distance to Other Points",
        ylab="Avg Euclidean Distance")
```



```r
# 3. Identify those beyond the upper whisker (= Q3 + 1.5*IQR)
stats     <- boxplot.stats(avg_d)$stats    # [1]=min [2]=Q1 [3]=med [4]=Q3 [5]=max
upper_cut <- stats[4] + 1.5*(stats[4] - stats[2])
outliers  <- which(avg_d > upper_cut)
length(outliers)  # how many we found
```

```
[1] 32
```

```r
# 4. If more than 10, take the top 10 by avg_d
outliers_top10 <- head(order(avg_d[outliers], decreasing=TRUE), 10)
outliers[outliers_top10]
```

```
 461  340 2540 4722  442  486  501  555 2317  535
 461  340 2540 4722  442  486  501  555 2317  535
```

The 10 observations with the largest average distance are our outliers. These points are much further from the bulk of the data. Each of these observations are on average, more than 2 standard deviations farther away from all other observations than a "normal" observation. This could influence the clustering algorithms because the k-means centroids will be pulled towards any of these extreme values and hierarchical merges will bind them at very high heights.

f)

```r
# 1. Compute the correlation matrix on the standardized data
cor_mat <- cor(df_scaled)
kable(cor_mat, caption="The correlation matrix on the standardized data")
```

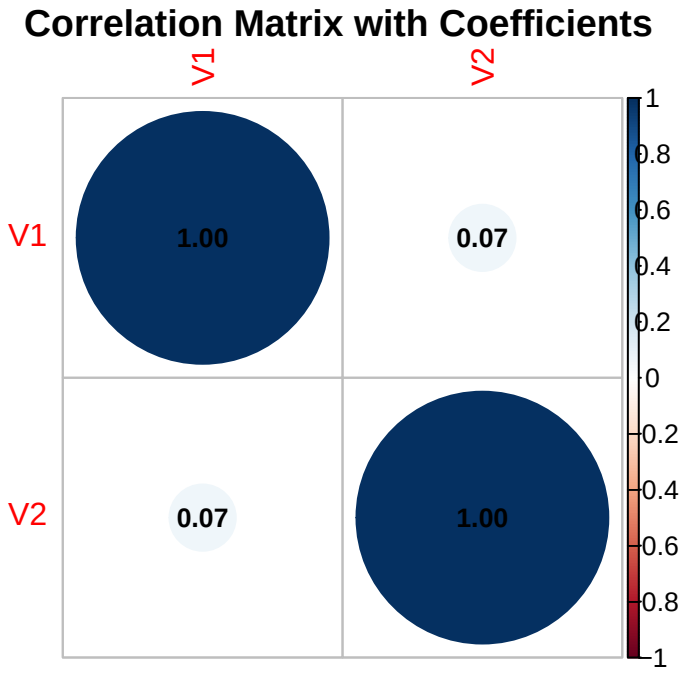
Table 4: The correlation matrix on the standardized data

	V1	V2
V1	1.0000000	0.0691441
V2	0.0691441	1.0000000

```r
# 2. (Optional) Visualise
library(corrplot)
```

```
corrplot 0.95 loaded
```

```r
corrplot(
  cor_mat,
  method     = "circle",        # or "shade"/"pie" etc for the shapes
  addCoef.col= "black",         # draw the correlation numbers in black
  number.cex = 0.8,             # shrink the text size if needed
  tl.cex     = 1.0,             # text size for variable names
  title      = "Correlation Matrix with Coefficients",
  mar        = c(0,0,1,0)       # make room for the title
)
```



The diagonals are 1 because each variable is perfectly correlated with itself. The off-diagonals represent the association between the 2 variables. As we can see, the correlation between variable 1 and 2 is 0.07. The low correlation suggests that each variable has their own, independent information which influences the data set and suggests no redundancy. These variables won't dominate each other when clustering. Due to the variables being slightly uncorrelated, there would be no need to drop one or decorrelate them as the euclidean distance on the standardized data will treat the variables fairly.

g)

Yes, we would standardize the data before computing any euclidean distances which is the distance metric we picked in the beginning. We would do this because the raw scales of the variables span hundreds of thousands, so this will drastically influence our clustering algorithms. The euclidean distance is calculated as the square root of the sum of the squares of the

differences between corresponding coordinates. Without standardisation, whichever variable has the largest absolute spread will dominate and lead to skewed results. By standardising, each variable has equal contribution. Since both variables showed similar IQRs and no extreme skew in their middle ranges, standardising simply aligns their scales without distorting the multimodal pattern in V1 or the slight skew in V2, ensuring our clustering sees genuine structure, not scale artifacts.

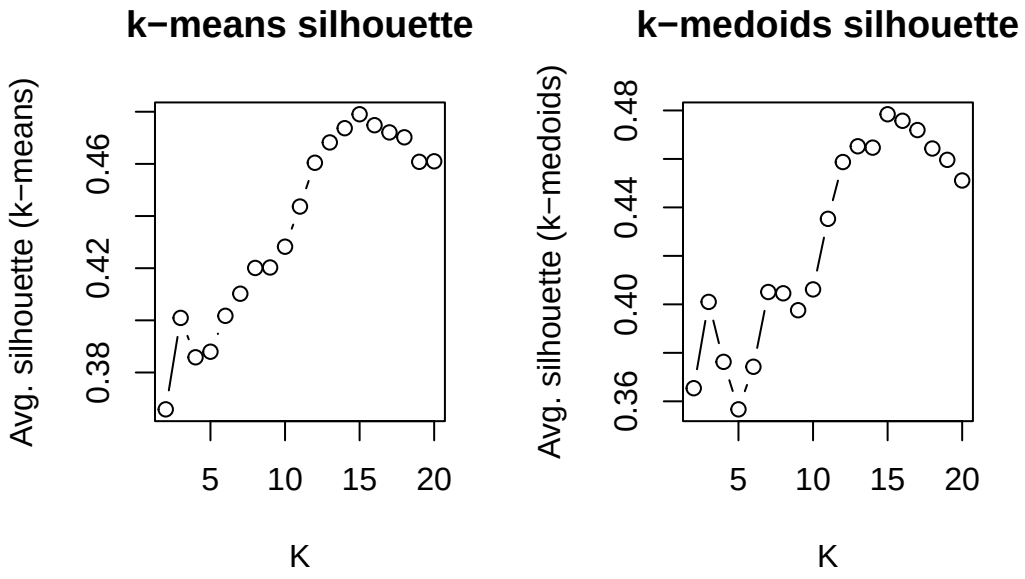
Question 2

a)

```
library(cluster)
#Compute average silhouette for k-means
sil_km <- sapply(2:20, function(K) {
  km  <- kmeans(df_scaled, centers = K, nstart = 50, iter.max = 100)
  sil <- silhouette(km$cluster, dist_matrix)
  mean(sil[, "sil_width"])
})

#Compute average silhouette for k-medoids (PAM)
sil_pam <- sapply(2:20, function(K) {
  pamc <- pam(dist_matrix, k = K, diss = TRUE)
  mean(pamc$silinfo$widths[, "sil_width"])
})

#Plot them side by side
par(mfrow = c(1,2))
plot(2:20, sil_km, type = "b",
     xlab = "K", ylab = "Avg. silhouette (k-means)",
     main = "k-means silhouette")
plot(2:20, sil_pam, type = "b",
     xlab = "K", ylab = "Avg. silhouette (k-medoids)",
     main = "k-medoids silhouette")
```




```
par(mfrow = c(1,1))
```

Both the k-means and k-medoids silhouette curves exhibit two natural "peaks" in the range K=2–20. First, there is a clear local maximum at k=3 (average silhouette 0.40 for both methods), indicating that three clusters capture the most prominent division in the data, splitting into two clusters loses cohesion, while four or five clusters start to overfragment the main groups. Beyond k=8, the silhouette steadily rises again, peaking around k=15–16 (0.47 for k-means, 0.48 for k-medoids), which suggests a much finer partition yields the tightest, most well-separated clusters, but that many clusters may be over-splitting. Therefore, based on average silhouette alone, two sensible choices emerge for both algorithms: a coarse segmentation at k=3 and a fine-grained segmentation at k=15 (or 16), depending on whether we want to prioritize broader overview or maximum cluster compactness.