**Introduction**

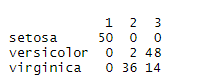
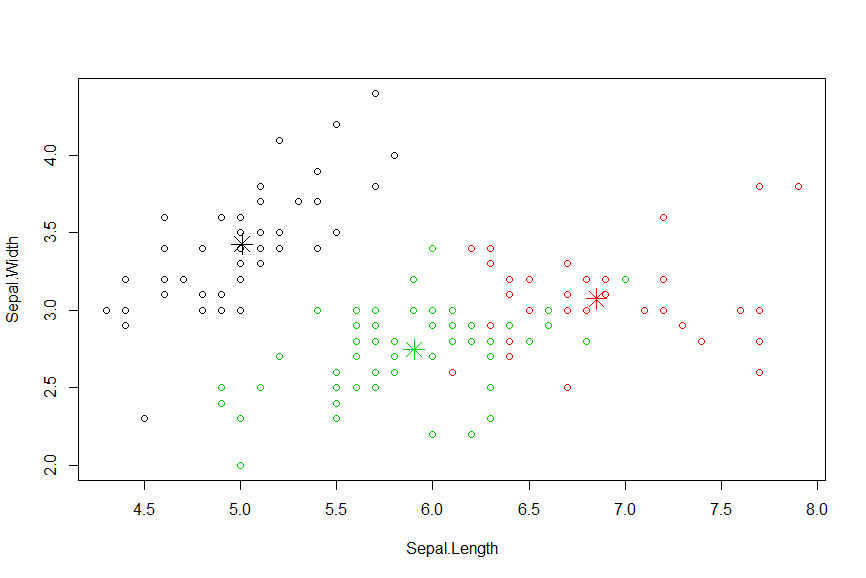
KMeans clustering is an unsupervised machine learning algorithm. I will summarise how it works, before examining the results obtained when running it on the iris dataset within R and consider how the algorithm is working on this dataset.

**KMeans Clustering (Background)**

Clustering is plotting datapoints across multiple dimensions, in order to form distinct “clusters” of data. The aim of this is to enable classification of new data points, by determining which cluster they are closest to. Real world applications include fraud detection in insurance and cheat detection in online video games. KMeans finds those clusters by first defining a “K” number of centroids declared by the user, and iteratively adjusting their positions so to minimise the within cluster sum of squares [Garbade, 2018]. The centroids are randomly selected from the data. The algorithm starts by assigning each point to the centroid with the lowest squared Euclidean distance from it. The means of all the data points assigned to each cluster are calculated for each centroid (hence the “means” in KMeans) and the positions of the centroids are adjusted based off of this calculation. This process is repeated until either the within cluster sum of squares (variance) is minimised and no more data points move between clusters, or the maximum number of iterations is reached, whichever comes sooner [Trevino, 2016]. If the algorithm stops as a result of no further inter-cluster movement, then convergence has been achieved. Although this is not guaranteed to be the optimum solution.

**Running the Code**.

Ideally, all three clusters should all have 50 values all belonging to one species of iris. When we run it, we see that all 50 data points corresponding to the Setosa species are confined to a single cluster. Versicolor is distributed across two clusters, and so is virginica. Fig 1.0 is a table obtained from a specimen run, and fig 2.0 is the plot obtained from the same run. The plot shows a distinct cluster on the top left of the plot, with a mean sepal length and width corresponding to those of Setosa. We can safely conclude that this is cluster 1. Below that and to the right are 2 overlapping clusters. The leftmost has a centroid around the mean sepal length of versicolor. Since cluster 3 is biased towards that species, we can safely conclude that this is cluster 3.



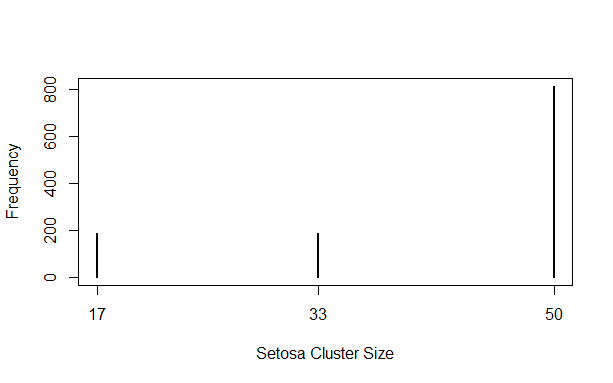
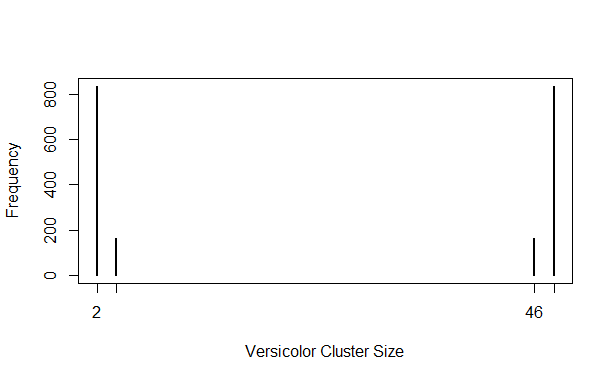
**Fig 1.0: The distribution among clusters of the**

**different species when we run KMeans on the Iris Dataset**

**Fig 2.0: A plot obtained from running KMeans on the**

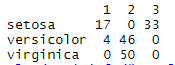
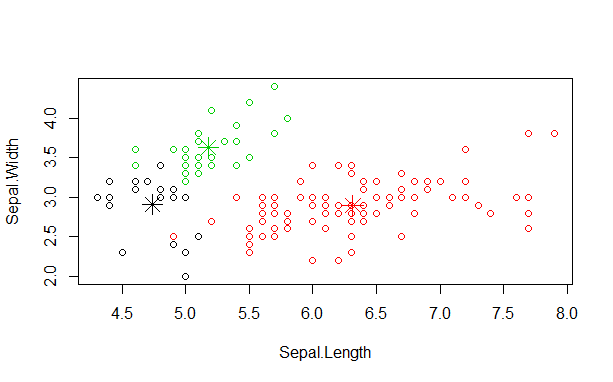
**iris dataset.**

**Monte Carlo Simulation**

In order to see if this is the distribution we get every time, I converted the code into a Monte-Carlo simulation (MCS) in order to run the algorithm multiple times and plot the distribution of cluster sizes for each species. If this is the result that the algorithm outputs every time, we would expect to see one large spike at 50 for the Setosa plot, spikes at 2 and 48 for versicolor and at 36 and 14 for virginica. When we run the MCS the expected spikes are the largest ones present on all the plots, but they are not the only ones. Setosa has two smaller spikes at 17 and 33, versicolor has a smaller spike at 4 and another smaller one at 46 and virginica has a smaller spike at 50. These plots are depicted in Fig 3.0

**Fig3.0 Plots obtained showing the distributions of cluster sizes per species when the Monte Carlo simulation is run.**

It is clear from the monte Carlo simulation that the results obtained in fig 1.0 is what occurs the majority of the time, but another result is also obtained many a time albeit not to the same frequency. I manually ran the original code in order to obtain the recurring alternative distribution that accounts for this. Fig 4.0 depicts this distribution and fig 5.0 is the plot obtained. The algorithm manages to confine virginica to a single cluster, but also puts 92% of the versicolor within the same cluster. Although not all of the Setosas are in one cluster, there is a cluster dedicated to that species. The algorithm still fails to distinguish versicolor and virginica from one another. In fact, it thinks that they are almost the same.



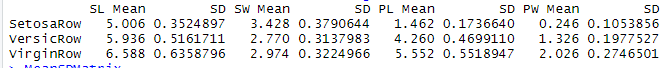
**Fig 4.0. Another distribution that KMeans**

**keeps returning from the data**

**Fig 5.0. The plot obtained on fig 4.0.**

**Overlaps in the data and how this affects centroid placement:**

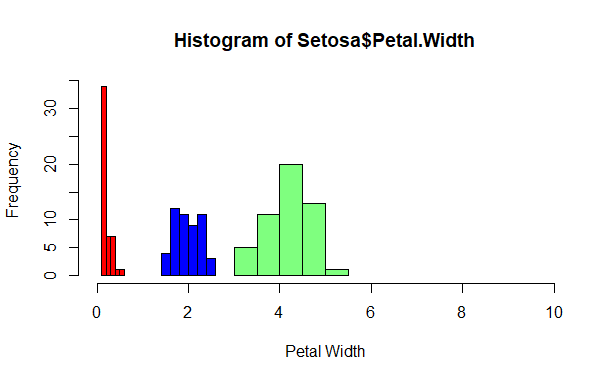
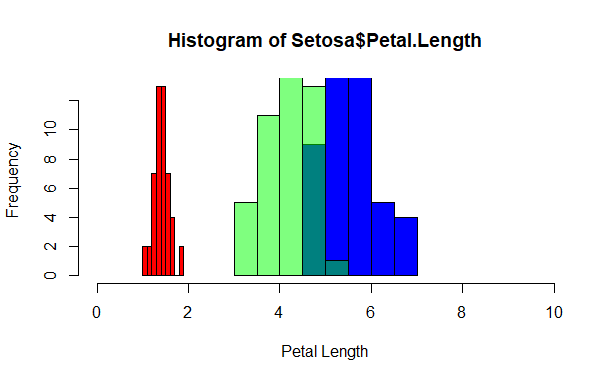
So why is the algorithm better at classifying Setosa than the other two species, and why does virginica sometimes get its own cluster? To answer those questions, we need to first consider that KMeans in R initially selects a K number of random points from the data to set as the initial centroids [Jaiswa, 2018]. If a particular centroid is initially set far away from the other two, then it is more likely to be part of a cluster containing only one species. Because it is less likely that a data point of another species will be closer to it than the centroid of the cluster that it should belong to during the means calculation step. This is more likely to happen if the two centroids are closer together. To get an idea of where the centroids are likely to be set, we should examine the means and standard deviations of all the dimensions for each species relative to each other. Empirical examination indicates that the mean Sepal and Petal lengths and widths of Setosa, are set apart from the other two species. Setosa is usually lower than the other two species for all dimensions, except for sepal width. Table 1 shows these values, it was obtained from splitting the Iris dataset up into separate tables for each species within R. This indicates that Setosa’s data forms an outlying cluster. The closest centroid to all the Setosa points is likely to be far away from the other points, which is why Setosa is most often confined to one cluster and no other species falls into the same cluster. In order to verify the statistical soundness of this hypothesis, t-tests were run between each species for each dimension. While the t test is usually used to determine if there is a significant difference between two samples, I am going to use it as a measure of how likely the two dimensions are to overlap. The results which are depicted in table 2 show that when Setosa is T Tested against the other species, the p values are multiple orders of magnitude smaller than the p values obtained when virginica and versicolor are tested against each other. This is true for all dimension, but the greatest differences are noted in the petal lengths and widths. Fig 6.0 provides visualisations for petal lengths and widths, where this pattern is most noticeable. It is clear that the data points for Setosa is distant from those of virginica and versicolor.



**Table 1.0. The means and standard deviations of all dimensions by species of the iris dataset.**

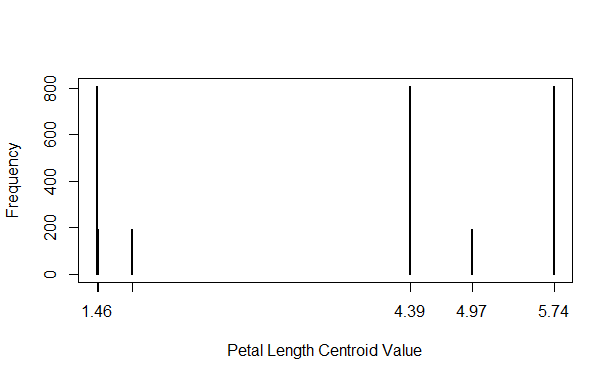


**Table 2.0.** **The results of the t-tests by dimensions and species pairs.**



**Fig 6.0. Histograms showing the distributions of petal lengths and widths for Setosa(red) versicolor (green) and virginica (blue). The Setosa is set apart from the other two which even overlap in petal lengths.**

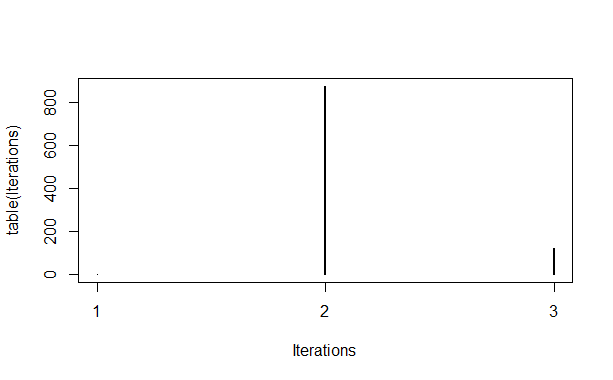
Since the initial setting of centroids is random, but still based off the data points, there is a high probability that there will be an outlying centroid in the region of Setosa’s plots, which will eventually form part of the cluster containing only Setosa. This is reflected in the results of the Monte Carlo simulation. But why do we sometimes see virginica getting its own cluster? If you examine the standard deviations in table 1, you will notice that the standard deviations of virginica being the largest in all the dimensions. A hypothesis that could explain this is that on some occasions, a centroid is being placed near the outliers of virginica and all of virginica end up there. Considering that the differences in SDs are less than those of the means, this is not going to happen very often, and the Monte Carlo simulation reflects this. In order to get an idea of how the centroids are being placed, we should plot the distribution of KMeans centre positions. It would be excessive to plot all the dimensions, so in remaining consistent I will plot the distribution of the petal length centroid values. When we run this plot, we see a big spike around 1.46, which corresponds to the actual mean petal length of Setosa. We also see two smaller spikes, one of which is a7 4.97, which is within one standard deviation from the mean peta length of virginica. This plot can be found in Fig 7.0. This of course, reflects the position of the centroids after they are iteratively moved in the means part of KMeans, but considering the 4.97 placement has been repeated 200 times it is reasonable to assume that the initial placement of the centroid won’t be that far off. This finding therefore supports the hypothesis, that a centroid is sometimes initially set near an outlier of the virginica data. A large spike is also seen at 1.46, which is the true mean petal length of the Setosa species within the dataset, confirming that most of the time Setosa is being correctly classified.



**Fig 7.0. A plot of the distribution of petal length values of the centroids obtained from the Monte Carlo simulation.**

**Maximum Iterations Reached?**

We have established that there is some overlap between the versicolor and virginica species, which explains why this particular implementation of KMeans cannot distinguish between the two. But is this because the maximum number of iterations is reached before the within cluster sums of squares are minimised? If so, would it be possible to distinguish between those two species if more iterations were run? It is possible to obtain the number of iterations used for each run of the KMeans algorithm. When the distribution of iterations was obtained from the Monte Carlo simulation, there was a large spike at 2 and a smaller spike at 3. Fig 8.0 depicts this. Since the default maximum number of iterations in the R implementation of KMeans is 10, and those numbers are far below this, then it is safe to assume that the within cluster sums of square have been minimised in all 1000 runs of the algorithm. It is clear that the overlap between the versicolor and virginica datasets is too great for them to be fully distinguished by KMeans in R.



**Fig 8.0. Distribution of numbers of iterations when the KMeans algorithm was run 1000 times in the MCS.**

**Conclusion**

A clear pattern emerges when running KMeans on the iris dataset in R, where it can on most runs distinguish Setosa but not Virginica and Versicolor. This is explained by Setosa’s data being set apart from the other two. Anomalous readouts are often obtained as a result of a centroid being placed near the outliers of the Virginica data due to its high standard deviation. KMeans is a heuristic algorithm, which doesn’t always provide the perfect results. The reliability of the result seems to be affected by to what extent the different clusters overlap.

**References:**

Gabarde M.J. (2018) *Understanding K-means Clustering in Machine Learning* Available at: <https://towardsdatascience.com/understanding-k-means-clustering-in-machine-learning-6a6e67336aa1> (accessed 9/3/20)

Trevino A. (2016) *Introduction to K-means Clustering* Available at: <https://blogs.oracle.com/datascience/introduction-to-k-means-clustering> (accessed 9/3/20)

Jaiswa J. (2018) *K-Means Clustering in R Tutorial* Available at: <https://www.datacamp.com/community/tutorials/k-means-clustering-r> (accessed 9/3/20)

**See appendices on next page for code**

**Appendices**

**Appendix 1, the code used to run the Monte Carlo Simulation and generate figs 3.0. 7.0. and 8.0.:**

Runs = 1:1000

SetosaDistVect = vector()

VersicDistVect = vector()

VirginiDistVect = vector()

CentPetLength = vector()

Iterations = vector()

for (i in Runs){

iris

newiris <- iris

newiris$Species <- NULL

newiris

kc <- kmeans(newiris, 3)

kc

output = table(iris$Species, kc$cluster)

set = output['setosa',]

setsel = set != 0

setosa = set[setsel]

SetosaDistVect = c(SetosaDistVect,setosa)

vers = output['versicolor',]

verssel = vers !=0

versicolor = vers[verssel]

versicolor

VersicDistVect = c(VersicDistVect, versicolor)

virg = output['virginica',]

virgsel = virg !=0

virginica = virg[virgsel]

virginica

VirginiDistVect = c(VirginiDistVect, virginica)

Cents = kc$centers

CPL = Cents[,'Petal.Length']

CentPetLength = c(CentPetLength,CPL)

Iterations = c(Iterations, kc$iter)

}

plot(table(SetosaDistVect),xlab = "Setosa Cluster Size",ylab = "Frequency")

plot(table(VersicDistVect),xlab = "Versicolor Cluster Size",ylab = "Frequency")

plot(table(VirginiDistVect),xlab = "Virginica Cluster Size",ylab = "Frequency")

CPL = format(round(CentPetLength,2))

plot(table(CPL),xlab = "Petal Length Centroid Value",ylab = "Frequency")

plot(table(Iterations))

**Appendix 2, the code used to obtain the mean, standard deviations and T Test values of the Iris dataset, and generate Tables 1.0. and 2.0. and fig 6.0.**

MeanSDMatrix = matrix(data = NA, nrow = 0, ncol =8, byrow = TRUE)

SpeciesVect = iris$Species

SetSel = SpeciesVect == 'setosa'

Setosa = iris[SetSel,]

SetosaSL = c(mean(Setosa$Sepal.Length),sd(Setosa$Sepal.Length))

SetosaSW = c(mean(Setosa$Sepal.Width),sd(Setosa$Sepal.Width))

SetosaPL = c(mean(Setosa$Petal.Length),sd(Setosa$Petal.Length))

SetosaPW = c(mean(Setosa$Petal.Width),sd(Setosa$Petal.Width))

SetosaRow = c(SetosaSL,SetosaSW,SetosaPL,SetosaPW)

VersSel = SpeciesVect == 'versicolor'

Versicolor = iris[VersSel,]

VersicSL = c(mean(Versicolor$Sepal.Length),sd(Versicolor$Sepal.Length))

VersicSW = c(mean(Versicolor$Sepal.Width),sd(Versicolor$Sepal.Width))

VersicPL = c(mean(Versicolor$Petal.Length),sd(Versicolor$Petal.Length))

VersicPW = c(mean(Versicolor$Petal.Width),sd(Versicolor$Petal.Width))

VersicRow = c(VersicSL,VersicSW,VersicPL,VersicPW)

VirgSel = SpeciesVect == 'virginica'

Virginica = iris[VirgSel,]

VirginSL = c(mean(Virginica$Sepal.Length),sd(Virginica$Sepal.Length))

VirginSW = c(mean(Virginica$Sepal.Width),sd(Virginica$Sepal.Width))

VirginPL = c(mean(Virginica$Petal.Length),sd(Virginica$Petal.Length))

VirginPW = c(mean(Virginica$Petal.Width),sd(Virginica$Petal.Width))

VirginRow = c(VirginSL,VirginSW,VirginPL,VirginPW)

colnames(MeanSDMatrix) = c('SL Mean', 'SD','SW Mean','SD','PL Mean','SD','PW Mean','SD')

rbind(MeanSDMatrix,SetosaRow,VersicRow,VirginRow)

MeanSDMatrix

SVeSepL = t.test(Setosa$Sepal.Length,Versicolor$Sepal.Length)

SViSepL = t.test(Setosa$Sepal.Length,Virginica$Sepal.Length)

ViVeSepL = t.test(Versicolor$Sepal.Length,Virginica$Sepal.Length)

SVeSepW = t.test(Setosa$Sepal.Width,Versicolor$Sepal.Width)

SViSepW = t.test(Setosa$Sepal.Width,Virginica$Sepal.Width)

ViVeSepW = t.test(Versicolor$Sepal.Width,Virginica$Sepal.Width)

SVePetL = t.test(Setosa$Petal.Length,Versicolor$Petal.Length)

SViPetL = t.test(Setosa$Petal.Length,Virginica$Petal.Length)

ViVePetL = t.test(Versicolor$Petal.Length,Virginica$Petal.Length)

SVePetW = t.test(Setosa$Petal.Width,Versicolor$Petal.Width)

SViPetW = t.test(Setosa$Petal.Width,Virginica$Petal.Width)

ViVePetW = t.test(Versicolor$Petal.Width,Virginica$Petal.Width)

Sepal\_length = c(SVeSepL$p.value,SViSepL$p.value,ViVeSepL$p.value)

Sepal\_width = c(SVeSepW$p.value,SViSepW$p.value,ViVeSepW$p.value)

Petal\_Length = c(SVePetL$p.value,SViPetL$p.value,ViVePetL$p.value)

Petal\_Width = c(SVePetW$p.value,SViPetW$p.value,ViVePetW$p.value)

TTestMatrix = cbind(Sepal\_length,Sepal\_width,Petal\_Length,Petal\_Width)

rownames(TTestMatrix) = c("Setosa Versicolor","Setosa Virginica","Virginica Versicolor")

TTestMatrix

hist(Setosa$Sepal.Length, xlim= c(0,10),xlab = "Sepal Length",col = "red")

hist(Virginica$Sepal.Length, add=T,xlim= c(0,10), col="blue")

hist(Versicolor$Sepal.Length, add=T, xlim= c(0,10), col=rgb(0, 1, 0, 0.5))

hist(Setosa$Sepal.Width, xlim= c(0,10),xlab = "Sepal Width",col = "red")

hist(Virginica$Sepal.Width, add=T,xlim= c(0,10), col="blue")

hist(Versicolor$Sepal.Width, add=T, xlim= c(0,10), col=rgb(0, 1, 0, 0.5))

hist(Setosa$Petal.Length, xlim= c(0,10),xlab = "Petal Length",col = "red")

hist(Virginica$Petal.Length, add=T,xlim= c(0,10), col="blue")

hist(Versicolor$Petal.Length, add=T, xlim= c(0,10), col=rgb(0, 1, 0, 0.5))

hist(Setosa$Petal.Width, xlim= c(0,10),xlab = "Petal Width",col = "red")

hist(Virginica$Petal.Width, add=T,xlim= c(0,10), col="blue")

hist(Versicolor$Petal.Length, add=T, xlim= c(0,10), col=rgb(0, 1, 0, 0.5))

mean(as.matrix(Setosa[,1:4]))