The K-Means Algorithm, a Solution to the Problem of Unsupervised Clustering

Gregory Pollard, Emily Hird, Nowell Crooks

January 16, 2021

Contents

1 First Meeting Summary

Prior to any development of the report, we entered a voice call to delegate tasks between members of the group. The report was to be split into sections regarding:

• An introduction, in which the contents of the report, background information on the k-means algorithm, and some alternative solutions to the problem of clustering were to be detailed. (Written by all members).

• A detailed summary on the problem of clustering with unsupervised algorithms, and in addition the benefits and drawbacks of the k-means algorithm. (Written by Nowell).

• A thorough, step-by-step explanation on how the k-means algorithm generates clusters and how to choose an optimal value for k. (Written by Gregory).

• The application of the k-means algorithm to image compression in order to reduce file size or save disk space. (Written by Emily).

• A conclusion of the report regarding the results we arrived at and in turn assessing the limitations of the algorithm. (Written by all members).

2 Introduction

In this report, we discuss the k-means algorithm, how it works and how it can be used. The k-means algorithm is a way of clustering numerical data so that the target categorical variable can be classified for any new data points we introduce. Grouping data has many uses, such as in numerical taxonomy. Numerical taxonomy looks to create classification systems with numerical methods based on characteristic states, creating the motivation to develop cluster analysis.[12] The book "Principles of numerical taxonomy", by Sokal and Sneath (1963) in particular started worldwide research on clustering methods.[4] The SSQ clustering criterion was developed as a method to cluster data points by minimizing the distance between the data points and the centroid. Extensions of the SSQ criterion then lead to the k-means algorithm.[12] Alternatives to k-means clustering were also developed such as:

Density-Based Clustering
In general, density-based clustering algorithms are used in databases of immense size; they can find and omit anomalies, identify clusters of arbitrary size and shape, and do not require a specification on the number of clusters prior to execution. For example, Density-Based Spatial Clustering of Applications with Noise (DBSCAN) is a density-based clustering algorithm proposed in 1996 which classifies spacial data.[7]

Hierarchical Clustering
Another solution to classifying data into clusters is hierarchical clustering, a process in which the desired metric (a measure of distance between pairs of observations) is used to split/merge clusters together in the following ways:

• **Agglomerative**: A method in which we assign each data point to its own cluster and then merge each group in pairs until we have a singular cluster containing all of the data; Ward's method is a well known way of doing this.[5]

• **Divisive**: Also known as DIANA (DIvisive ANAlysis); this is a polar opposite solution in which we start with a model of all data points and recursively split the most appropriate data into pairs of clusters.[6] The process continues until a stopping criterion (a pre-defined number of k clusters) is achieved.[1]

3 The Problem of Clustering

3.1 Unsupervised Algorithms

Primarily, issues with unsupervised learning from algorithms stem from two things. Firstly, the fact there are no real answers to verify results against. Results are subject to interpretation and choices without an objective answer like distance measures or initialisation methods have a significant impact on results. Secondly, real world data sets won't fall into neat clusters. Some samples will be empty or have overlapping clusters, some will take odd sizes or shapes; creating one algorithm capable of dealing with all this would be a very complex algorithm. These problems combined mean it can be hard to tell if your algorithm is working as it should.

3.2 Identifying problems

Checking clustering quality isn't a rigorous process because unsupervised clustering lacks objective results to verify against. However there are steps you can take to confirm the quality of your clustering:
• **Visual Check**: Are the results far different than what you'd expected? Are the samples you consider similar in the same cluster? Would the inter-cluster magnitude (sum of distances between clusters for your distance measure) be smaller with a different distance measure?
• **Cluster Magnitude VS Cardinality**: Calculate the intracluster magnitude (sum of distances for your distance measure) and count the cardinality for all clusters. Plot the clusters' cardinality and investigate outliers then plot the magnitudes and do the same. Finally plot the two measures against each other and investigate outliers.
• **Performance of Downstream System**: Clustering is often used in downstream ML systems.[2] Checking if the system's performance improves when the clustering process changes provides a real world test for the quality of the

[1] Further reading on the comparison between these approaches can be found here.[8]

clustering. This is difficult and slow to perform with more complex algorithms. Overfitting[2] is a problem as the ML system is meant to be able to predict the test data but it will not generalize well if it's only tuned for the training data specifically.

3.3 K-Means Benefits

The k-means algorithm is a widely used algorithm for many reasons:

• **Intuitive**: Most data mining software includes a k-means package.[1] The algorithm is intuitive to understand and just as easy to implement. This makes it easier to identify and correct problems as they arise because people know how it should work.

• **Versatility**: The algorithm has dozens of modifications and alterations that make it suitable to different tasks dependent on your need. You can clip outliers, use different initialisation methods, or use different distance measures for different cluster shapes [1]. Additionally, principal component analysis can help when clustering data with too many dimensions.[3]

• **Fast to Cluster**: K-means has relatively low time complexity so it can be run quickly several times. Consequently, the algorithm scales well to large data sets where more complex (hierarchical) algorithms take too long to compute.[1]

• **Guarantees Convergence**: The centroids will always converge to optimal positions.[4] It can be sent to a bad local optima if the wrong initialization method is used.

3.4 K-Means Drawbacks

The k-means algorithm has the problems typical of a partition clustering algorithm:

• **Elbow Method**: There's no standardised way to describe a best k even though the results depend on the choice. This invites subjectivity and makes the method less scientific. There's often no way of knowing how many clusters exist or their size. This problem can be solved by running the algorithm several times as described in section 4.1. We use the elbow point to avoid over fitting the model to the training data whilst still lowering the error function.

• **Being Dependent/Sensitive to Initial Values**: k has to be chosen manually beforehand and has a big impact.[11] For a low k, you can mitigate this dependence by running k-means several times with different initial values and picking the best result. As k increases, you need advanced versions of the k-means algorithm to pick better values of the initial centroids.[5]

• **Outliers**: Outliers might get their own cluster or drag centroids instead of being ignored. Remove outliers before clustering or use a different distance measure than Euclidean.

• **Clusters of Varying Size and Density**: K-means has trouble clustering data where clusters are of varying sizes and density. The algorithm produces circular-like clusters but wouldn't work as well for arbitrarily shaped clusters. The algorithm also assumes the data produces similarly sized clusters when this might not be the case with real world data sets. This exacerbates the susceptibility to outliers. To cluster such data, you need to adapt k-means or use a different type of clustering such as those mentioned in section 2.

4 K-Means Algorithm

4.1 Step 1: Initialize K

The k in the name k-means is the number of clusters we want to split our data up into, unfortunately however, we don't know the optimal number of clusters we need that will best represent the data prior to analysis. So we need to find the best k for our data, a popular method of doing this is to execute the algorithm with increasing numbers of $k \in \{1, 2, 3...\}$ and then find what is known as the "elbow point", which was briefly discussed in 3.4.[14] Figure 1 shows a graph of the

[2]Overfitting occurs when we train our model too tightly on the training data, this means that the model will predict values poorly when new data is introduced.

[3]Further reading on dimension reduction with PCA can be found here.[3]

[4]Full proof can be found here.[10]

[5]More can be read on the different initialisation methods here. [1]

average distance[6] from all data points to their assigned centroid after running the algorithm.[7] The aforementioned "elbow point" is the value of k for which the gradient of the line in Figure 1 changes the most, in this example, $k = 5$. In doing this, we are selecting the value of k which decreases the size of the clusters from one value of k to the next the most without choosing a value for k that is too high.[8]

Intuitively, we can see that our elbow point graph will always look similar to Figure 1 because the distance that each point is to its cluster's centre will naturally decrease as the number of clusters increase. This conclusion may become easier to see as we detail the next steps of the algorithm in 4.2, 4.3, and 4.4. This method of finding our optimal value for k is quite unsurprisingly called the elbow method. So now we know how to find the elbow point, but we need to run the algorithm to actually create this graph in the first place, so where do we start?

4.2 Step 2: Randomly Place Centroids

For each value of k that we run the algorithm for, we place k random points on our scatter plot, these points are called centroids and will eventually become the centre point for each cluster. It is important to note that it doesn't matter where we initially place these points, this is because our algorithm will (through a recursive process) move our centroids to the most optimal position anyway, no matter where they are placed at the start.

4.3 Step 3: Recursively Move Centroids to Optimal Positions

Now that we have our centroids, we need to move them to the positions that will eventually become the centre of their corresponding clusters. To break this process down, we shall describe it in 4 stages. In addition, we'll use graphics to explain this visually, where $k = 3$ for an arbitrary set of a data points:

• **Calculate Distance (Figure 2):** To begin, we find the distance of each data point to every centroid individually. We use our chosen metric to do this, and from this data we can then determine which points are similar to others later on.

• **Assign Data Points (Figure 4):** Secondly, we utilize these distances to assign each data point to its closest centroid. By doing this we have partitioned the data into clusters.

• **Re-locate Centroids (Figure 3):** Following this, we move our centroids to the centre of their assigned data points. This "centre" is the position given when we take the average of the (x, y) coordinates for each data point in a cluster.

• **Repeat Until Centroids Stabilize (Figure 5):** Finally, we repeat the prior 3 stages until we reach a point where all of our centroids fail to move when compared to the previous iteration. Our centroids are now correctly placed and we have our optimal clusters for the given k value.

4.4 Step 4: Take Averages

Upon executing the algorithm with each value of $k \in \{1, 2, 3...\}$, we can then find our optimal value for how many clusters we need (as described in Step 1) by finding the average distance of all data points to their assigned centroid. The clusters that we define using our algorithm at this optimal value for k are our desired solution to the problem of unsupervised clustering using the k-means algorithm. If a new data point is introduced, we can easily classify it by running the algorithm again, there are many packages in common programming languages than can run this algorithm in seconds with ease[9]. It is important to also note that we can evaluate how good our model is at classifying new data points through many avenues. This can be done in many ways, but is out of scope for this report and so further reading can be found here.[13] Now that we've described the algorithm fully, let's look at a popular way in which it is utilised.

[6] When we say distance, it just means the difference between observations based on our desired metric. Some common metrics for dissimilarity between data points when using K-means are: Euclidean, Manhattan (city-block), and Cosine distance.

[7] A centroid is simply the point that we designate to be the centre of a given cluster, this is detailed further in Step 2 (4.2).

[8] Choosing a value for k that is too small means our data will not be partitioned into enough clusters to draw any conclusions from, on the other hand, choosing a value that is too high will over fit the data and make the concept of a cluster meaningless.

[9] For example, R has functions that can perform this algorithm in the "stats" package, and Python has similar functions in the "scikit learn" package.

5 Application of K-means to Image Compression

Images are groups of small pixels, each being a colour that can be represented as an RGB value. RGB values can be given in the form (r,g,b) where r,g, and b are integers between 0 and 255. We can use the integer values in the k-means clustering algorithm to convert the closest colours to the same singular colour.

The code in section 7.1.3 shows that we apply the k-means algorithm to the image by treating the R, G, B values as dimensions to find k colours that can recreate the image. Since k relates to the number of colours remaining after compression, it follows that the smaller k, the smaller the image size. However, we lose image quality in the process.

5.1 Lossy vs Lossless Compression

K-means clustering is a form of lossy compression as information is lost when it's compressed, meaning quality is reduced and the compression cannot be reversed. Another example of lossy compression is chroma subsampling because it reduces the chroma information.[9]

Lossless compression is reversible and does not remove any information. While lossless may keep the image at a higher quality than lossy there is a limit to how much lossless compression can reduce an image by. An example of lossless compression is Run Length Encoding (RLE) which reduces the size of images by recording the number of colours along with the colour. For example, if four red pixels were in a row, they would be recorded as 'four red' rather than 'red red red red'. Lossless and lossy compression are often used in combination as it would be pointless to use k-means clustering if every pixel still had it's colour specified.

5.2 Example

We can use the source code in 7.1.3 to compress Figure 6 into Figures 7 and 8. For Figure 7 we have reduced the image from 37KB to 20KB, in Figure 8 we take this Further reducing the image to 9KB but significantly reducing the quality by doing so. We can see that the image size can be significantly reduced using the k-means method. However, the quality decreases with size. We can plot colours based on their RGB values, as shown in Figure 9, we can see the size of the groupings and the variation within groups by doing this. Comparing Figure 10, where each point is coloured in its true colour, to the clustered Figure 9 we can observe how much the colours in the same cluster vary. Furthermore, we can see that the central green colour in Figure 9 can hardly be seen in Figure 10. The k-means algorithm takes the mean point which can lead to a colour being used that is not in the original image due to k including a range of colours.

6 Conclusion

Throughout this report we have described the problem of clustering, the k-means algorithm itself, and demonstrated a common way in which it is used. We've also highlighted the key strengths and weaknesses of the algorithm itself, and in which situations these are emphasized. In particular, a large disadvantage to this algorithm is that it can only be used when we wish to classify categorical data given that we have only numerical data. It also assumes that the data will produce clusters containing roughly equal numbers of observations, and that they are spherical in shape; this is unreasonable to assume for many real data sets.

However, given these assumptions are true, the k-means algorithm can be rather useful as a tool for introducing one to the world of machine learning as it is fairly simple to understand and can be executed with only a few lines of code. As a solution to the problem of unsupervised learning, the k-means algorithm may not be the most complex or accurate answer for many sets of data; but as a tool to demonstrate the power and potential of machine learning for predicting data, it can be very useful indeed.

7 Appendix

7.1 K-Means Algorithm

7.1.1 Choosing K Graph (Figure: 1)

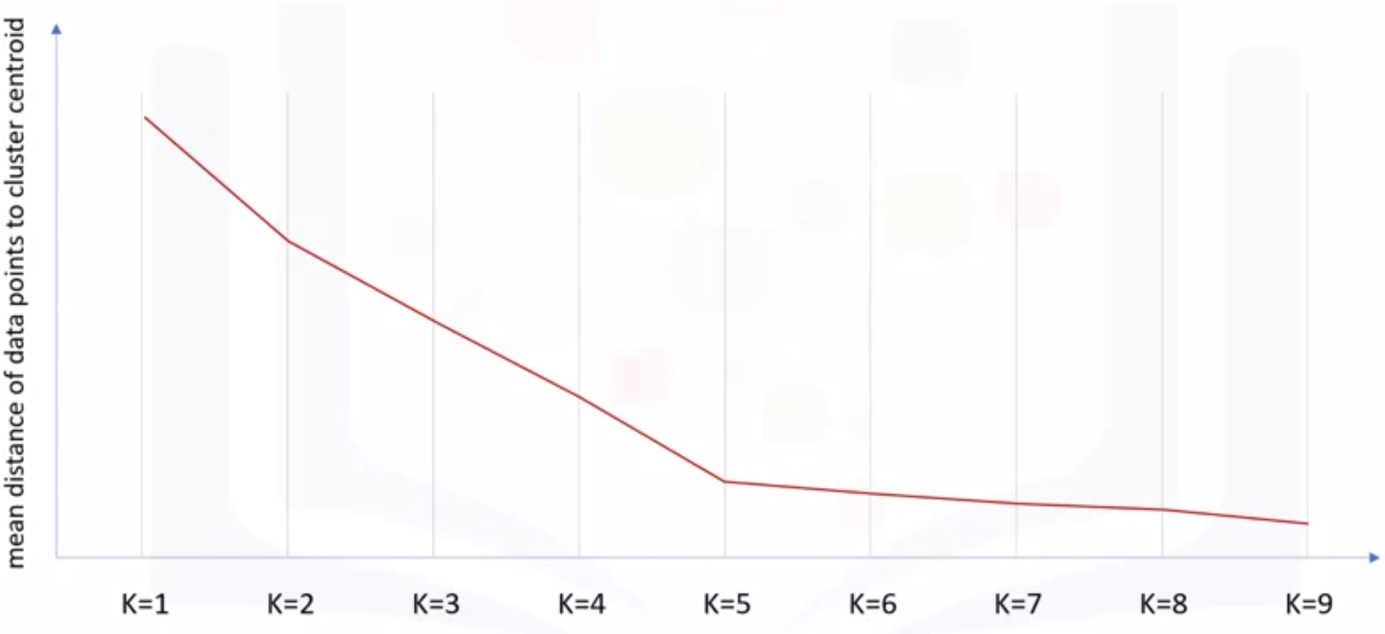

Figure 1: A general graph showing the mean distance of all data points to their cluster's centroid as K increases.

7.1.2 Stages of Step 3 (Figure: 2, 3, 4 and 5)

Stage 1

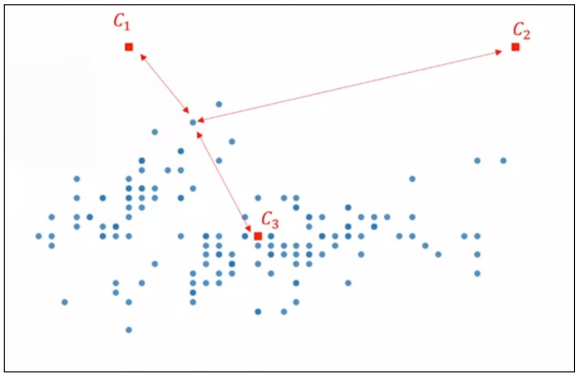

Figure 2: Calculate distance.

Stage 2

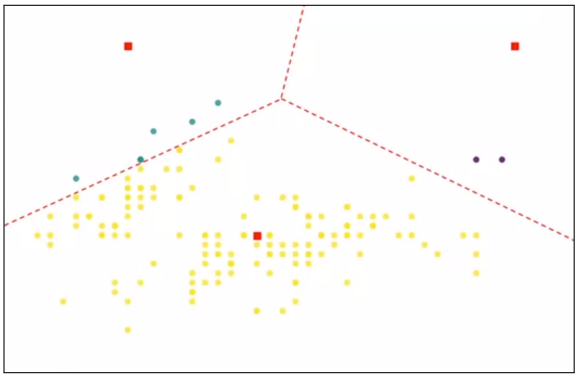

Figure 4: Assign data points.

Stage 3

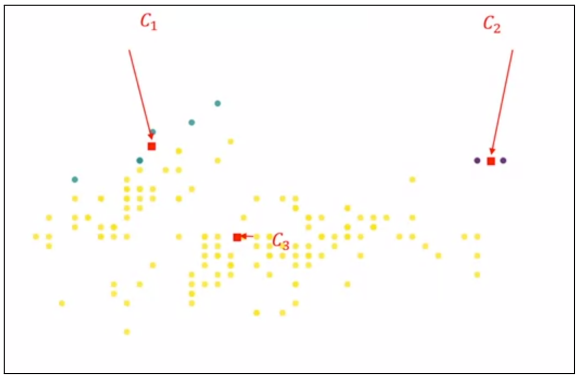

Figure 3: Re-locate centroids.

Stage 4

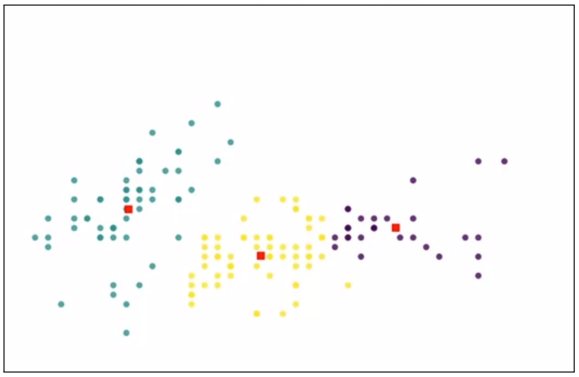

Figure 5: Repeat until centroids stabilize.

7.1.3 Compression Code

```
# Obtain Image  dimension
imgDm <- dim(img)

# Assign RGB channels to data frame
imgRGB <- data.frame(
  x = rep(1:imgDm[2], each = imgDm[1]),
  y = rep(imgDm[1]:1, imgDm[2]),
  R = as.vector(img[,,1]),
  G = as.vector(img[,,2]),
  B = as.vector(img[,,3])
)


# Compress the image using k-means clustering
k <- 3 # Number of clusters, you can play with it to obtain different compression levels!
kMeans <- kmeans(imgRGB[, c("R", "G", "B")], centers = k)
num.of.colours <- rgb(kMeans$centers[kMeans$cluster,])
```

Source Code 1: Image compression code

7.1.4 Image Prior to Compression (Figure 6)

Figure 6: Image of flower

7.1.5 Compressed Image 1 (Figure 7)

Figure 7: K-means clustering applied with $k = 24$

7.1.6 Compressed Image 2 (Figure 8)

Figure 8: K-means clustering applied with $k = 6$

7.1.7 Cluster Plot (Figure 9)

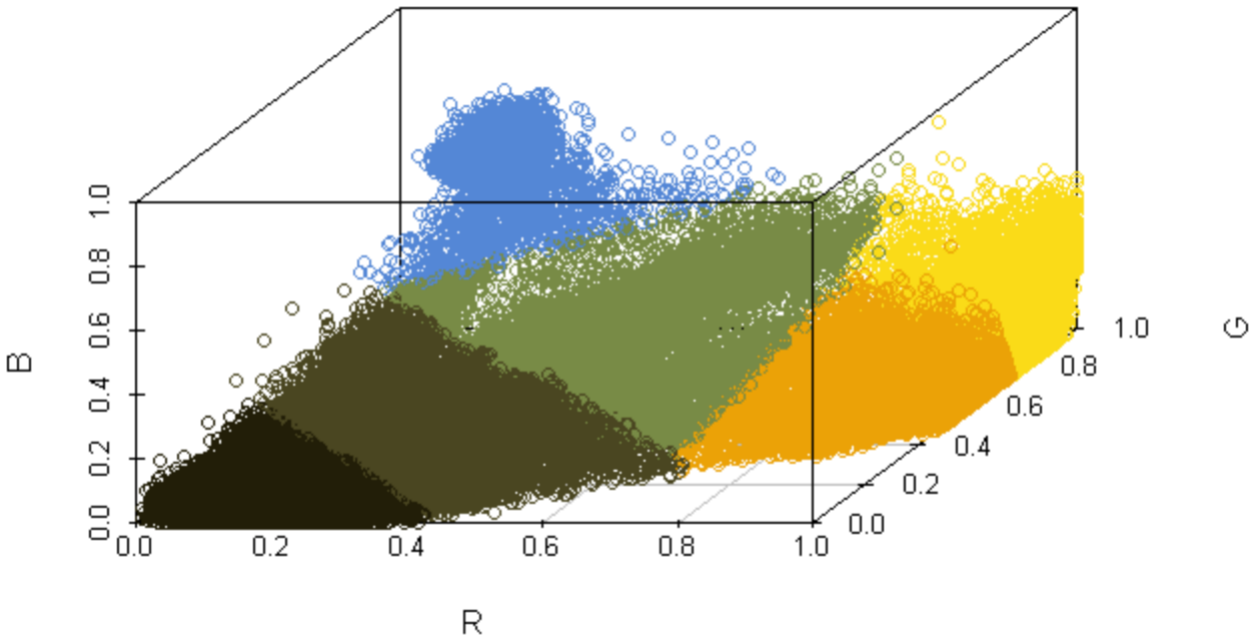

Figure 9: Clusters when $k = 6$

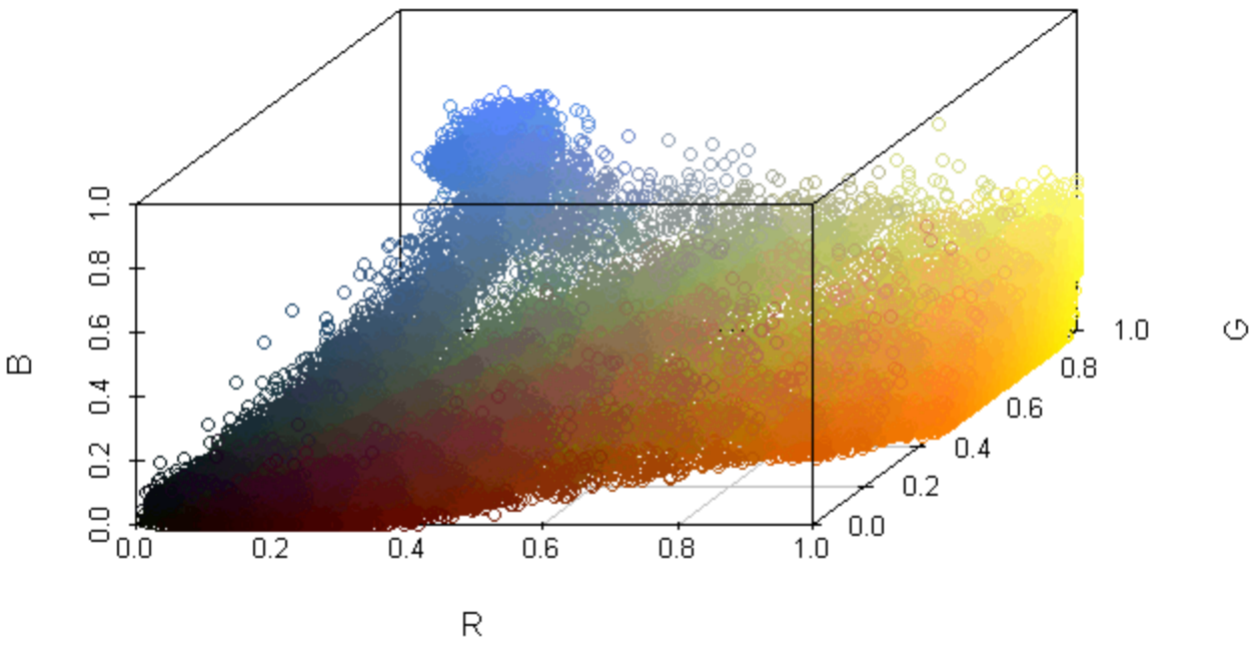

Figure 10: Each colour plotted with its original colour

References

[1] M. Emre Celebi, Hassan A. Kingravi, and Patricio A. Vela. "A comparative study of efficient initialization methods for the k-means clustering algorithm". In: *Expert Systems with Applications* 40.1 (2013), pp. 200–210. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2012.07.021. URL: http://www.sciencedirect.com/science/article/pii/S0957417412008767.

[2] C. Darken and J. Moody. "Fast adaptive k-means clustering: some empirical results". In: *1990 IJCNN International Joint Conference on Neural Networks*. 1990, 233–238 vol.2. DOI: 10.1109/IJCNN.1990.137720.

[3] Chris Ding and Xiaofeng He. "K-Means Clustering via Principal Component Analysis". In: *Proceedings of the Twenty-First International Conference on Machine Learning*. ICML '04. Banff, Alberta, Canada: Association for Computing Machinery, 2004, p. 29. ISBN: 1581138385. DOI: 10.1145/1015330.1015408. URL: https://doi.org/10.1145/1015330.1015408.

[4] Bock HH. *Clustering Methods: A History of k-Means Algorithms*. 2007. DOI: https://doi.org/10.1007/978-3-540-73560-1_15.

[5] Joe Ward Jr. "Hierarchical Grouping to Optimize an Objective Function". In: (1963), pp. 236–244. DOI: https://amstat.tandfonline.com/doi/pdf/10.1080/01621459.1963.10500845?needAccess=true.

[6] Leonard Kaufman and Peter J. Rousseeuw. "Finding Groups in Data: An Introduction to Cluster Analysis". In: (1990). DOI: https://onlinelibrary.wiley.com/doi/book/10.1002/9780470316801.

[7] Jörg Sander Martin Ester Hans-Peter Kriegel and Xiaowei Xu. "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise". In: (1996). DOI: https://www.aaai.org/Papers/KDD/1996/KDD96-037.pdf.

[8] Maurice Roux. "A Comparative Study of Divisive and Agglomerative Hierarchical Clustering Algorithms". In: (1963). DOI: https://hal.archives-ouvertes.fr/hal-02085844/document.

[9] C. J. van den Branden Lambrecht S. Winkler and M. Kunt. *Vision and Video: Models and Applications*. 2001.

[10] S. Z. Selim and M. A. Ismail. "K-Means-Type Algorithms: A Generalized Convergence Theorem and Characterization of Local Optimality". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-6.1 (1984), pp. 81–87. DOI: 10.1109/TPAMI.1984.4767478.

[11] Xin-She Yang Simon Fong Suash Deb and Yan Zhuang. "Towards Enhancement of Performance of K-Means Clustering Using Nature-Inspired Optimization Algorithms". In: (2014). DOI: https://doi.org/10.1155/2014/564829.

[12] Sokal & Sneath. *Principles of Numerical Taxonomy*. 1963.

[13] M A Syakur. "Data Mining and Knowledge Discovery". In: (2016), pp. 891–927. DOI: https://doi.org/10.1007/s10618-015-0444-8.

[14] M A Syakur. "Integration K-Means Clustering Method and Elbow Method For Identification of The Best Customer Profile Cluster". In: (2018), pp. 3–4. DOI: https://iopscience.iop.org/article/10.1088/1757-899X/336/1/012017/pdf.