

Portfolio project: K-modes analysis of fighting game tier list data

Dr. Terence Vockerodt

March 2022

Contents

I	Introduction	3
1.1	Background	4
1.2	Tier lists	4
1.3	Project premise	5
II	Algorithms and methods	6
2.1	The problem	7
2.2	What is K-modes clustering?	7
2.2.1	Dissimilarity measure	7
2.2.2	Main algorithm	8
III	Results	10
3.1	Exploratory data analysis	11
3.2	What happens when K is varied?	14
3.3	Suggested tier list	20
3.4	Conclusion	21
IV	Data cleaning by presorting tier lists	25
4.1	The need for data cleaning	26
4.2	Data cleaning method	27

4.3	Preliminary tests	27
4.4	Conclusion	31
V	Iterative procedures and project summary	33
5.1	Addressing convergence	34
5.2	THE tier list	34
5.3	Project summary	35

Part I

Introduction

1.1 Background

E-sports are video game competitions that are played with spectatorship in mind. Competitions are visual spectacles set in otherworldly environments, with competitors being able to express themselves with superior tactics as they try to win. They are a relatively accessible medium - training for e-sports carries significantly fewer physical risks as compared to athletic sports. E-sports also remove the need for judges, since the win conditions for the games can be accurately determined within the confines of the game's code, without human bias affecting decisions. Much like athletes, players have their own drives, desires and goals. Some want to be the best, others want to turn their hobby into a lucrative career, and some just want to have fun with their friends. E-sports mark another stage in human competition, with many people heavily invested in winning with their team. Whilst some games remain relatively niche, there are other games that have a huge establishment and following such as chess, which has been historically significant since the 13th century.

Among many of my favourite competitive oriented games is a small platform fighting game called Rivals of Aether. Rivals distills down many of the fun aspects of the platform fighting genre in order to deliver fast paced combat, with an emphasis on using superior movement options to navigate through your opponents attacks.

1.2 Tier lists

The characters that players control in Rivals have different attributes and abilities. Due to these differences, typically there is a hierarchy associated with the characters called a 'tier list'. Traditional tier lists aim to categorise the characters that are similar into tiers, which gives a rough idea of their strength and power level. Tier lists are typically designed with the highest level of play in mind, and thus are typically made by top players of the game. A character is considered high tier if it is strong relative to the other members of the cast of characters.

Due to the differences between characters, some characters over others may be better suited to fight a particular character. For example, a character with ranged abilities can typically win more than they lose against slower, close ranged characters. If character A wins more matches on average than character B when they fight, then this is known as a winning matchup against B in favour of character A. The best character in the game is typically regarded as the character with the most relevant winning matchups. A relevant winning matchup would be a winning matchup against other strong characters.

This does mean that tier lists should not be taken at face value. A high tier character may have a losing matchup to a low tier character, but the low tier character is low tier because on average they have too few relevant winning matchups. Perhaps this hypothetical low tier character beats the best character in the game, but loses to the

remainder of the top tier characters, which overall solidifies it as a low tier. Nevertheless, tier lists are still used to gauge the relative strengths of characters compared to one another, which does help newer players make a more informed decision of what character they want to play.

1.3 Project premise

In June 2021, 37 of the top players of Rivals of Aether were polled to submit their individual ordered tier lists [1]. An ordered tier list essentially assigns a ranking from best to worst of all the characters in the game. The data was aggregated using K -means clustering in order to generate a definitive tier list which was both ordered and categorised [1]. I initially wanted to try and replicate their data using K -means as a part of this project. However, looking closer at the data, I thought that K -means was an inappropriate algorithm to use in this case. This is because the submitted ordered tier lists are categorical data and not continuous. From my research, it is not recommended to use a method like K -means on categorical data, since it is difficult to interpret the results. For example, if one of the clusters has a cluster mean of 2.345, what does that mean statistically in terms of a tier placement? A reasonable assumption would be that 2.345 is somewhere between the 2nd and 3rd placement, with it being closer to the 2nd, but this makes little sense since these placements are categorical (there is no in between, a character is either 2nd or 3rd). This creates some ambiguity in interpreting the results, and it might manifest itself in the data as potential errors.

Luckily for us, there are methods based off of the K -means principle but for categorical data. This is where the K -modes clustering method can be applied [2]. The K -modes clustering method will be explained in more detail later in this document, but in short instead of using proximity to the mean as a metric of cluster assignment, it uses similarity to the cluster mode [2]. Since all of the intermediate data remains categorical in this method, we do not run into the issue of having badly interpreted intermediate and auxiliary results.

The aim of this project is to compare and contrast the results that were generated using K -means and my results using K -modes. Part II will go over the basic theory of K -modes, as well as my decisions to alter the algorithm design so it can produce results. Part III showcases EDA and results from the K -modes method. Part IV explores using K -modes clustering to help data cleaning, which can be used to ensure that lists used represent a high skill level even if the sample size is increased. Part V attempts to find a convergent, model independent tier list by increasing the sample size of the data using a small handful of generated K -modes tier lists. I have decided to do the majority of this project in R, since I need more R experience. Data storage and some of the plotting will also be handled in Excel.

Part II

Algorithms and methods

2.1 The problem

The tier list problem can be described as follows. We want to take an aggregate of individual tier list statistics from top players, and create ordered tiers that group the game characters by their relative strength. The strength of the characters relative to one another is typically determined via matchup analysis, which requires top player expertise in order to determine. Once the matchups are analysed, the ordering of the characters can be determined. K -means clustering has been used before [1], but there is ambiguity in using the means of character placings as explained in section 1.3. Therefore, I opt to use K -modes clustering since this is more suited for categorical data. This section will detail the basic theory behind K -modes clustering, as well as changes in my numerical implementation due to the nature of the data set.

2.2 What is K-modes clustering?

As alluded to in part I, K -modes clustering is the analogue for K -means clustering for categorical data. Generally speaking, we consider a set of n objects, and each of the n objects have m categorical values associated with them. The K -modes clustering method will attempt to sort the n objects into K clusters, using the categorical attributes of the entire data set. A way of visualising the data is shown in Table 2.1, which has the m categorical values (denoted by A_1, A_2, \dots, A_m) as column headings, and the rows represent the n objects.

Table 2.1: A visual representation of the type of data used, which explains the terminology from reference [2].

i	A_1	A_2	\dots	A_m
1	\dots	\dots	\dots	\dots
2	\dots	\dots	\dots	\dots
\vdots	\dots	\dots	\dots	\dots
n	\dots	\dots	\dots	\dots

In our case, the rows are the characters, and the columns are each player's rankings of said characters. This is shown in Table 2.2.

2.2.1 Dissimilarity measure

We will now define a cluster mode Q_k as a set of values $\{q_{k,1}, q_{k,2}, \dots, q_{k,m}\}$. Here, k can range from 1 to the number of clusters K . We can measure the dissimilarity of an object X_i to a cluster Q_k using the following steps:

Table 2.2: Changing the abstract headings from Table 2.1 into the types of headings used in this project. The column headings are the player names, and the rows are the characters. The data that goes into the table are the ratings for each character from each player.

Character	player ₁	player ₂	...	player _m
character ₁
character ₂
⋮
character _n

1. Iterate j from 1 to m , and compare the j^{th} value of object X_i to the j^{th} value of Q_k . The j^{th} value of X_i is denoted as $x_{i,j}$, and the j^{th} value of Q_k is denoted as $q_{k,j}$.
2. If $x_{i,j} == q_{k,j}$, do nothing. Else, add 1 to the dissimilarity.

The dissimilarity quantity will range in value from 0 to m , with 0 meaning the object and cluster mode are the same, and m meaning that all the values of X_i are different from all the corresponding values of Q_k . The dissimilarity measure is the analogue to the proximity to the cluster mean used in K -means clustering.

2.2.2 Main algorithm

The typical algorithm used for clustering is as follows:

1. Initialise the cluster modes Q_k .
2. For each object X_i , calculate the dissimilarities for each cluster, and assign the object to the cluster with the least dissimilarity. If it's a tie, it can be allocated at random.
3. Re-calculate the cluster modes. This involves iterating over the objects in each cluster, and calculating the modes of each of the j in $1:m$ categories. Again, ties can be decided at random.
4. Repeat the above two steps until no object changes cluster.

However, we have to amend the cluster mode re-calculation step given our data. Essentially, step 3 boils down to looking at Table 2.1, selecting only the objects in the particular cluster, and then looking at each column in this new table to find the mode for that column. Given the data that we have, there is no mode in the columns. This is because the columns are the individual player's tier lists, which are ordered from 1 to 14, without any ties. Therefore, my amendment to the cluster re-calculation is

as follows: for each cluster, find a single mode value from all the player's ratings of the characters in the cluster, and assign $q_{k,1} = q_{k,2} = \dots = q_{k,m} = q_k$, where q_k is the new cluster mode. It is sensible to fix the cluster mode for each player, since the cluster centers should be in the same ranking for each player. Also, provided that character voting distributions are reasonably distributed, the mode of the cluster characters is a sensible metric for determining the cluster mode.

Part III

Results

3.1 Exploratory data analysis

I will firstly remark that the sample size of tier lists used is very small, which makes accurate statistical analysis difficult. One remedy to this is to poll more players, but then there is a risk of introducing more biases to the data due to the inclusion of less experienced player's lists. The top players that contributed their lists to this collection have very specialist knowledge about the game, which can be difficult for less experienced players to account for when they make their lists. It would be interesting to see the results with some compromise reached.

The main part of the EDA is to look at the distributions of the tier list placings for each character. I will comment on the modes of each character to see what challenges may arise. Starting with the fire characters in Fig. 3.1, we can see that there is a strong consensus for the character placings for all of the characters. In an ideal world, the placings would be strictly decreasing from the modal value, such that one can visualise a bell curve centered at the modal value and tailing at the ends. Despite a great modal consensus for these characters, we do not see this strictly decreasing behaviour from the mode for any of the characters, which is a symptom of not having enough data. Hopefully, the Rival's community can identify more professional level players and get their lists to improve this data, but for now at least the data is not completely unusable. A character's distribution is described as reasonable in this document if it has a single mode with strong consensus, and if it is decreasing from the modal value for most of the placings.

Fig. 3.2 show the air characters, where we see our first problematic character - Elliana has two modes. Again, this is indicative of not enough data used, but presents a more challenging technical aspect to cluster assignment. In the previous part, I mentioned that if cluster assignment was tied, then the cluster can be assigned randomly. Not wanting to have issues of characters that borderline two clusters hopping to and from one another, I decided to use another method. In the case of a tie, I assign the character to the cluster which has a mode closest in value to the character's mode. However, since Elliana has two modes, how is Elliana's mode decided? As a simple assumption, when calculating the character's modes, if the character has a list of modes I picked the mid-point of the list if it's an odd number of modes, or the right most middle value if it's an even number of modes. In Elliana's case, this corresponds to a mode of 7. Later on, I will showcase results that look at choosing Elliana's mode equal to 5. As for the other air characters, Absa has a reasonable looking distribution. Wrastor looks like the best character in the game so far, with a strong consensus and a mode of 1.

Fig. 3.3 shows the earth characters. There is little to talk about in terms of their placings, since the distributions at least have a strong consensus for the mode. Fig. 3.4 on the other hand shows the water characters, and our second problematic character is Etalus, which again has two modes. Like before, Etalus' mode is chosen to be 14, and I will also showcase some results with Etalus' mode chosen to be 11. Since K -modes clustering uses modal analysis and not the mean, it means that I

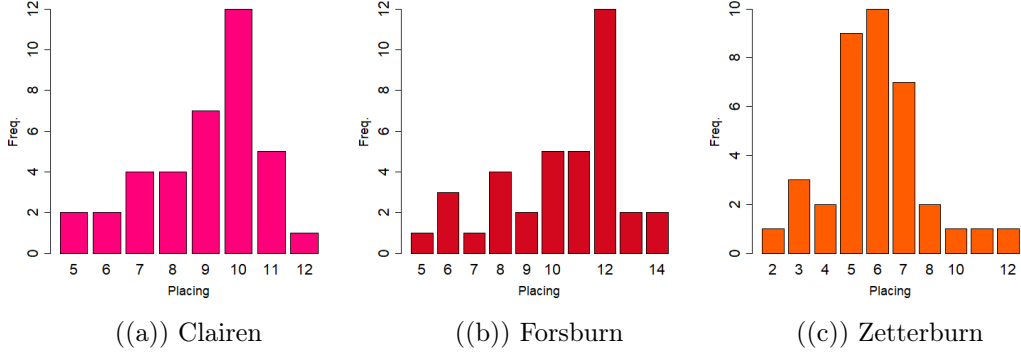


Figure 3.1: Frequency plots of the placings of the fire characters.

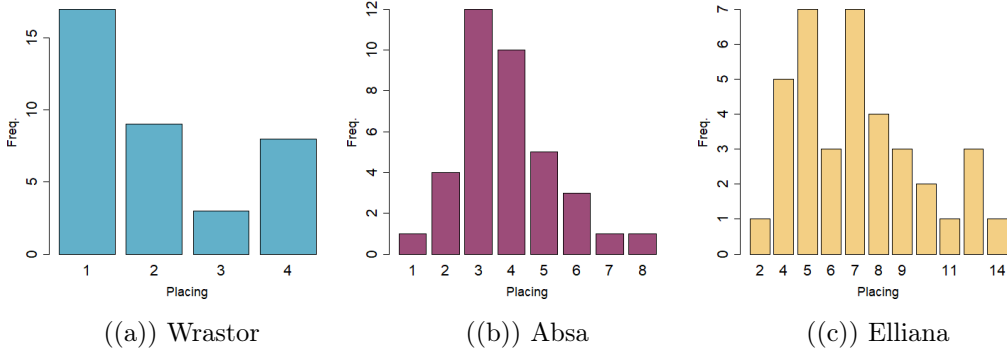


Figure 3.2: Frequency plots of the placings of the air characters.

did not have to throw out any outliers. This simplifies the data cleaning phase. Orcane and Ranno have reasonable looking distributions. Fig. 3.5 shows the third party characters, with Ori having a solid placing near the top of the tier list, and the opposite for Shovel Knight.

Ordering the modes, we get a tentative tier list order of Wrastor, Ori (Maypul), Absa, Zetterburn (Ranno), Elliana, Orcane, Clairen, Forsburn, Shovel Knight (Kragg), and Etalus (Sylvanos). If we chose the other mode for Elliana and Etalus, then they move before Zetterburn and Forsburn respectively. We can use this tentative ordering to check the results of our clustered tier list, which should follow this order closely. If we just needed the ordering of the tier list, then we could stop here with this simple modal analysis. However, because we require the clusters, we need to use the K -modes method on this data.

To summarise, most of the characters had reasonable distributions that looked like what we expected, except for Elliana and Etalus which had two modes each. Since I opted to use the character modes for tie-breaking in cluster assignment to avoid randomness, I chose the mode to be the lowest placing of the two. I predict that this simplification/assumption will not be needed with a larger sample size of tier lists, but at this stage this is difficult to get. Note that the mode tie-breaker scenario

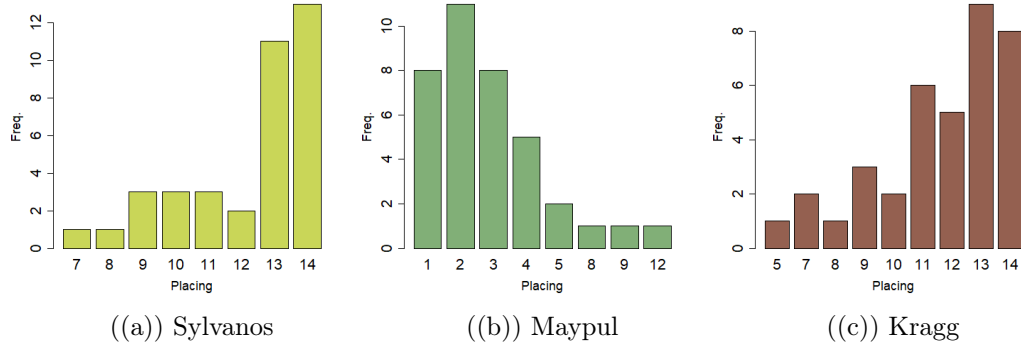


Figure 3.3: Frequency plots of the placings of the earth characters.

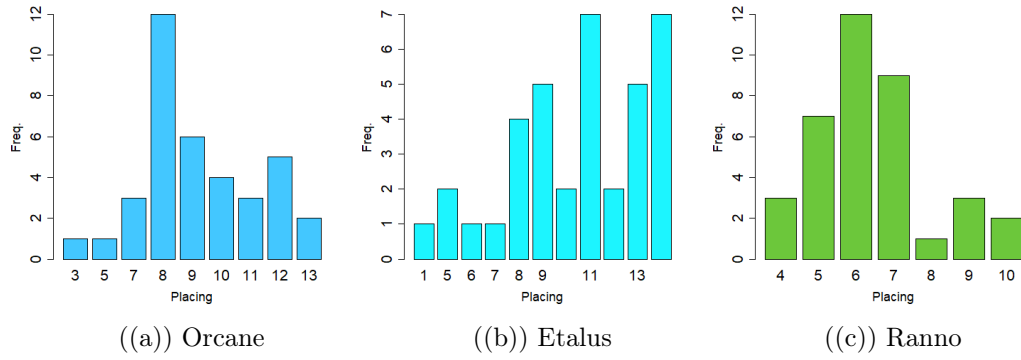


Figure 3.4: Frequency plots of the placings of the water characters.

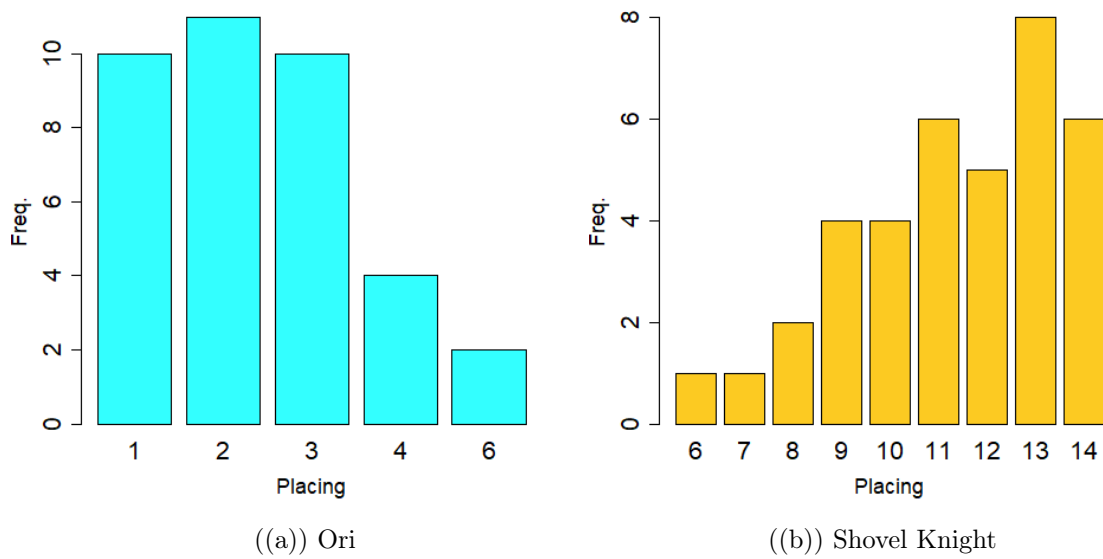


Figure 3.5: Frequency plots of the placings of the third party characters.

described above is also used when determining cluster modes. The data required little cleaning since the K -modes method uses the mode over the mean, which means that outliers did not have to be removed from the data-set. If the number of characters or the number of tier lists increase in the future, this means that K -modes analysis could be useful for getting results quickly, since it requires less data cleaning than K -means.

3.2 What happens when K is varied?

I would like to explore what happens when K is varied. Using a simple schema for the initial cluster modes, I varied K and looked at what happens to the dissimilarity metric for each character. The schema is shown in Table 3.1. I expect that the lower the dissimilarity for a character, the better the clustering is. For the sake of the investigation, I iterate to a K value much higher than what is practically needed.

Figs. 3.6 - 3.10 show the effect of varying K on each character pool. Starting with Fig. 3.6, we can see that the dissimilarity appears converged for Zetterburn and Clairen at $K = 5$. This is not true for all other characters, but it is a good starting point. The character dissimilarities that have not converged at $K = 5$ are Forsburn, Elliana, Absa, Maypul and Orcane, which indicates that these characters change clusters as K is increased. I will investigate if these variances are significant later in the text.

Table 3.1: The initial cluster modes for each K used.

K	Initial config.
1	1
2	1, 14
3	1, 7, 14
4	1, 5, 9, 14
5	1, 4, 7, 10, 14
6	1, 3, 6, 8, 11, 14
7	1, 3, 5, 7, 9, 11, 14
8	1, 2, 4, 6, 8, 10, 12, 14

Once the clusters had converged, they were ordered using the character modes from the EDA. Some of the character modes tied, so in this ordering, I also included the number of people who ‘voted’ for that character’s mode as a means of distinction. For example, if two characters A and B had a mode of 6, but 10 people voted for character A’s mode compared to 15 for character B, then character B would take the 6th placing, and character A would be placed in the next available slot. Fig. 3.11 shows the effect of varying K on the tier list results. Starting from $K = 4$ to $K = 5$, Absa gets put into her own cluster. From $K = 5$ to $K = 6$, the only change is that Elliana and Orcane break from their clusters and form their own. Increasing K further is unproductive, because K is nearing the number of

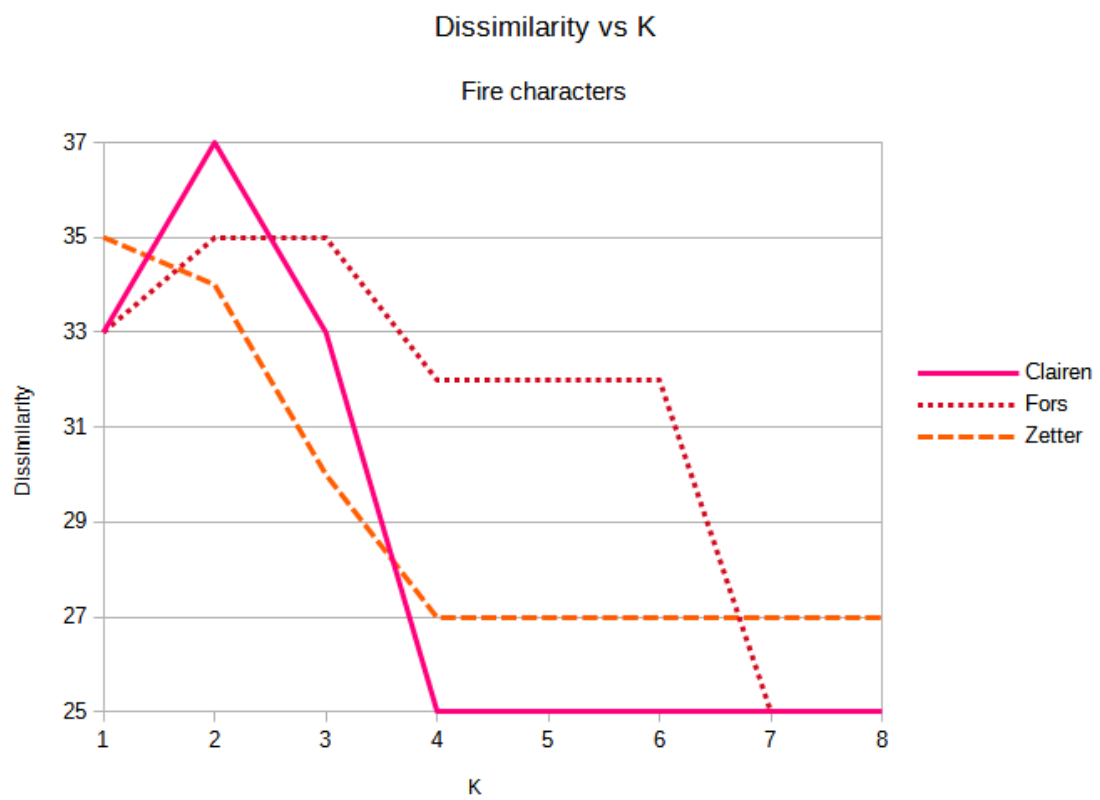


Figure 3.6: Dissimilarity vs K for the fire characters, using the initial conditions in Table 3.1.

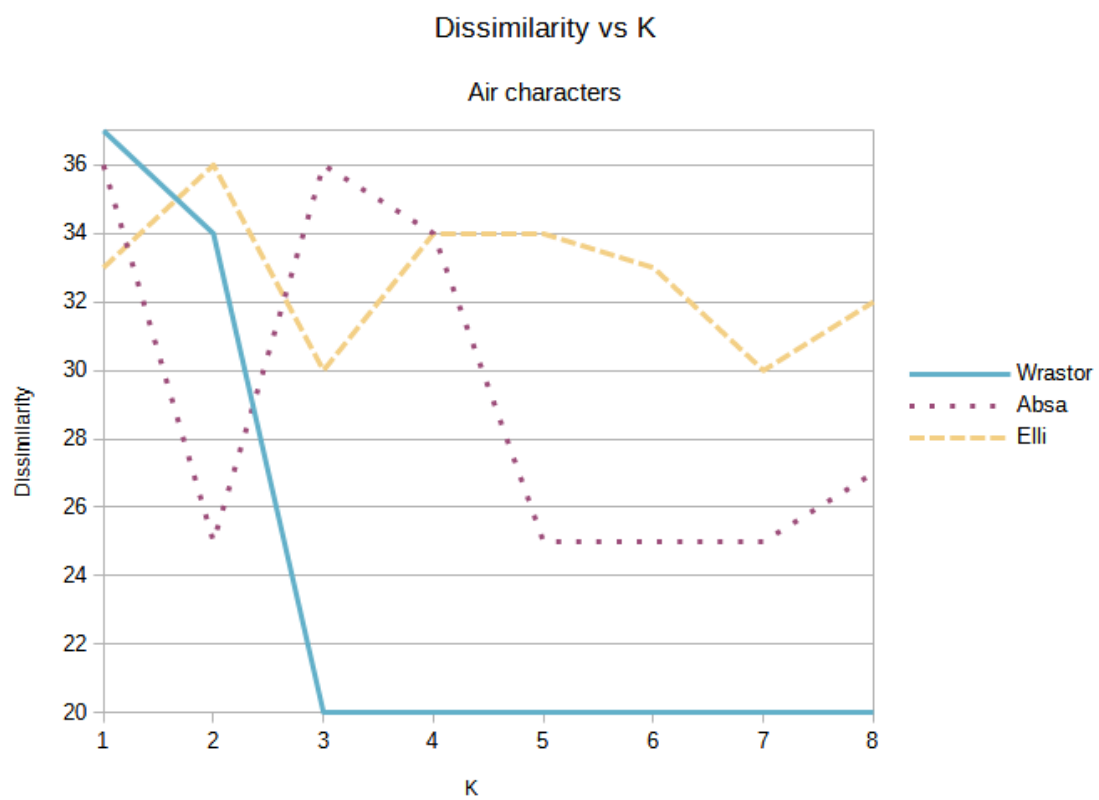


Figure 3.7: Same as Fig. 3.6, but for the air characters.

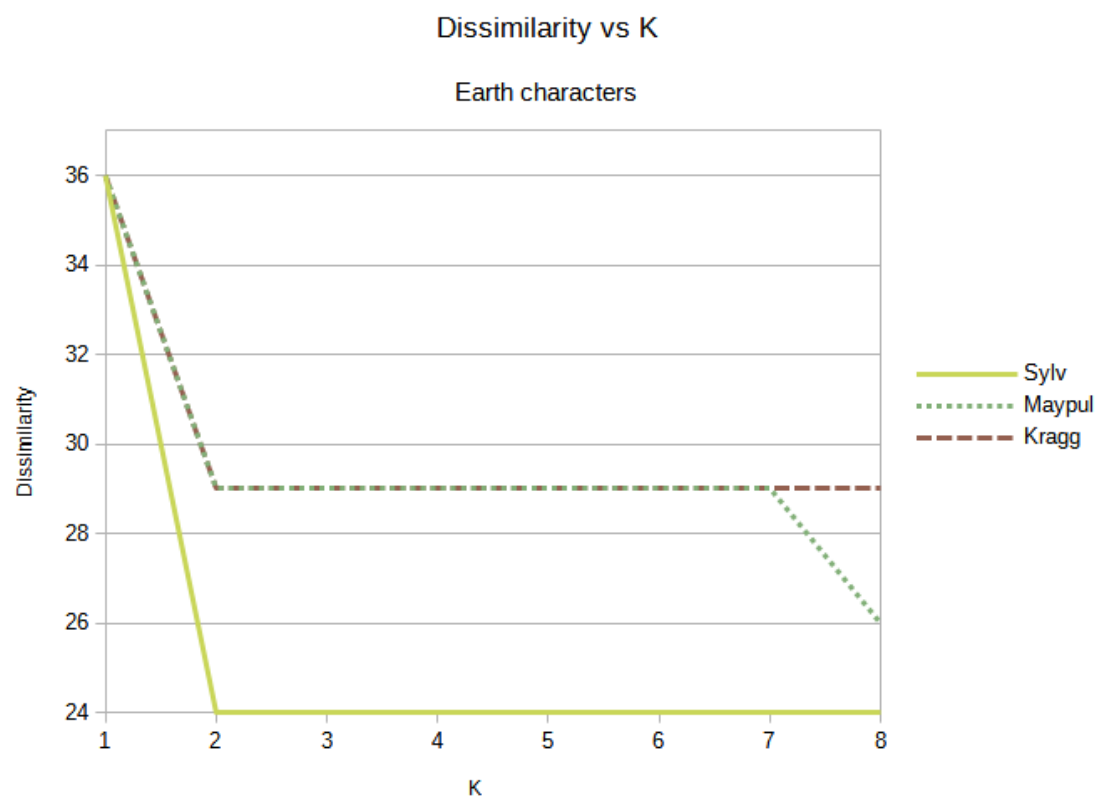


Figure 3.8: Same as Fig. 3.6, but for the earth characters.

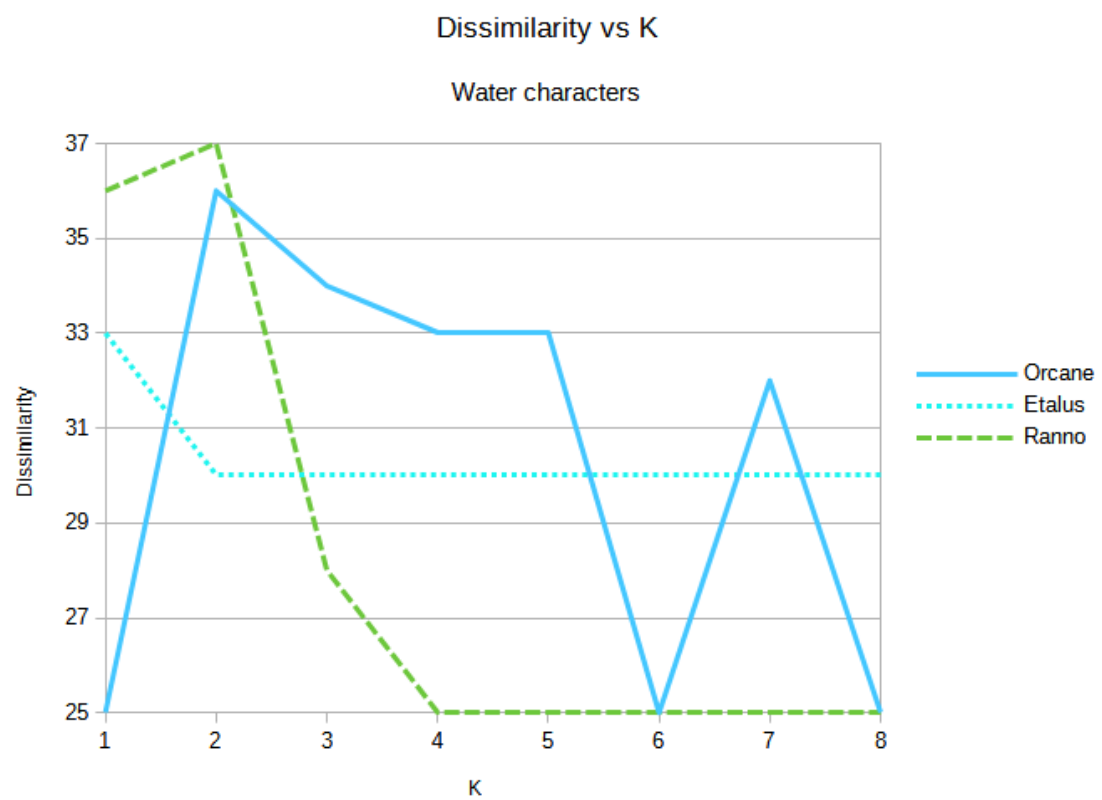


Figure 3.9: Same as Fig. 3.6, but for the water characters.

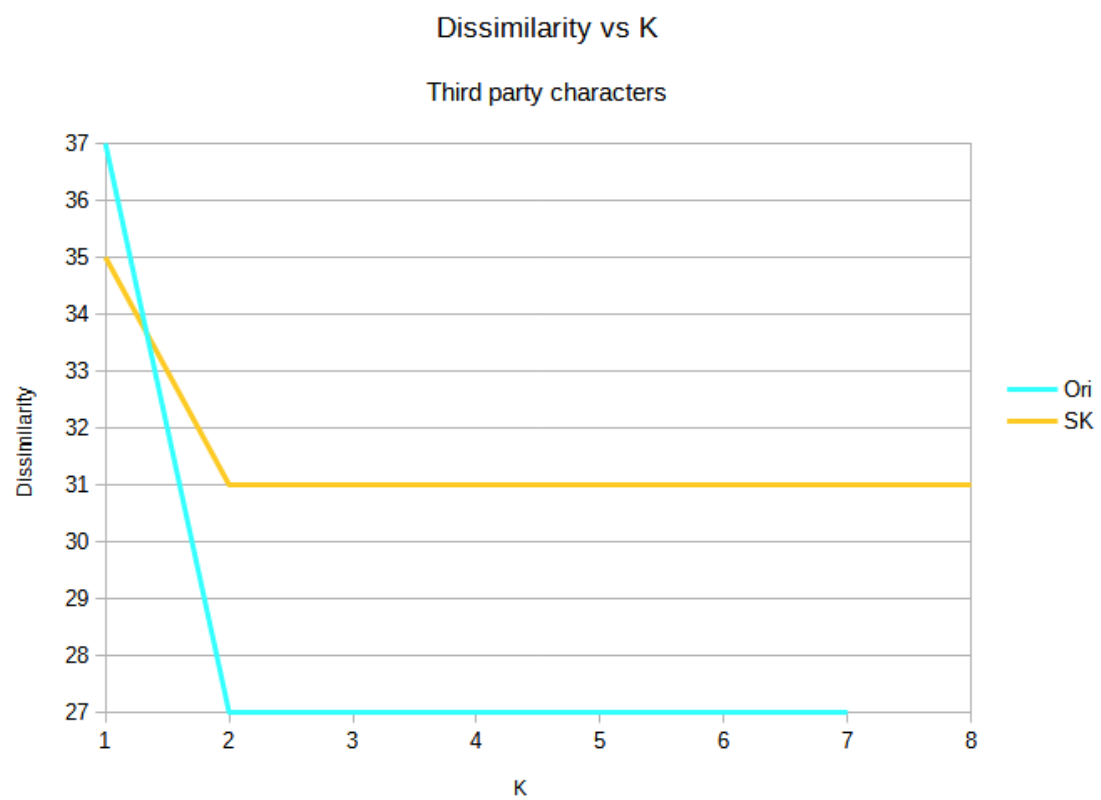


Figure 3.10: Same as Fig. 3.6, but for the third party characters.



Figure 3.11: Showing the tier list results from $K = 4$ to $K = 6$. Clusters are in order, with the first cluster representing the top tier. Increasing K mainly has the effect of sub-dividing previous tiers, with some characters occasionally changing tiers. Graphic made using resource in reference [3].

individual characters ($K = 14$ should simply reproduce the ordering that was found in the EDA). Past $K = 5$, increasing K either subdivides an existing cluster, or introduces a new cluster in between two clusters. Whilst the dissimilarity is not optimised for all characters at $K = 5$, further divisions are counter-productive and introduce unnecessary complexity. The top and bottom clusters remain the same throughout varying K . As reported in reference [1], there is a gap between Maypul and Absa, and Absa and Zetterburn, which is represented with Absa being in her own tier. Absa's solo tier is also consistent with the K -means analysis in reference [1].

Finally, Fig. 3.12 shows the variation of the cost function for K -modes with K . For K -modes, the cost function is calculated by summing all the dissimilarities associated with each cluster [2]. Using the elbow method, we can see that $K = 4$ clusters appears optimal.

3.3 Suggested tier list

My list is shown in Fig. 3.13, with the list from reference [1] shown in Fig. 3.14. The only difference between the lists is the placement of Absa and Etalus, which is due to the difference in the selection of the mode I have used in this K -modes implementation, and how K -means uses the means. Absa's distribution in Fig. 3.2((b)) is more similar to her companions in S-, despite having a higher modal value. As mentioned in reference [1], there is a gap between Maypul and Absa, and Absa and Zetterburn. In this work, I selected the worst of the two modes for Elliana and Etalus as 7 and 14 respectively. If we look at the mean of these two characters (calculations provided by reference [1], see description of the video for the spreadsheet), then Elliana is valued at approximately 7th place, and Etalus at approximately 10th place. This means that my mode selection for Elliana is close to her mean, but this is not the case for Etalus. Again, this analysis will benefit more from an increased sample size, since the distributions of Elliana's and Etalus' placings are not reasonably distributed.

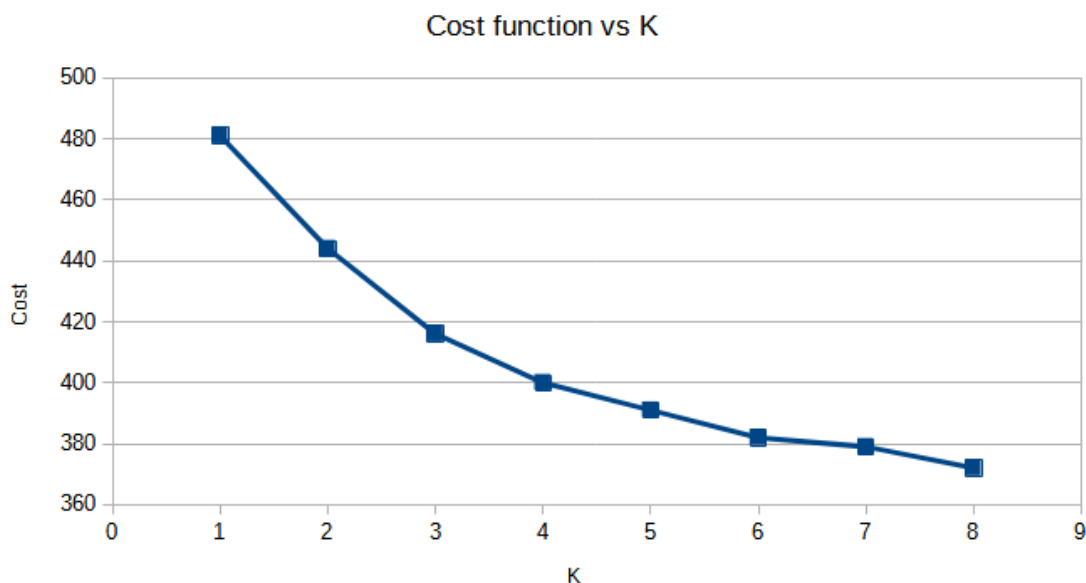


Figure 3.12: Relationship between the cost function and K . Elbow method suggests around $K = 4$ is sensible.

We can see the results of choosing the second modes in Fig. 3.15. This correctly predicts the ordering of Etalus, but keeps him in the bottom tier. However, it also puts Elliana higher than Zetterburn and Ranno. Again, a stronger census is required to accurately place these two characters, but otherwise the K -modes and K -means methods perform comparably.

3.4 Conclusion

The K -modes method provides consistent results with the K -means method employed by top players, with the added benefits of less ambiguity of intermediate results for the clusters, and less data cleaning due to using modal analysis. Requiring less data cleaning can be useful if the aforementioned weakness of the low sample size is addressed. K -modes is more appropriate for categorical data such as list placings, but requires a special treatment of calculating the cluster modes (i.e.: cluster modes are just single values, as opposed to being a value per player). In this work, I chose a simple treatment of calculating the cluster modes, but more experienced statisticians may know of a more sensible and rigorous approach for this particular problem. The tier lists produced are largely similar, but the K -modes method has less ambiguity in the intermediate results than K -means.

S+	
S-	
A+	
A-	

Figure 3.13: Suggested tier list from this work. Generated using $K = 4$. Tier labels used are the same as in reference [1]. Graphic made using resource in reference [3].







S+				
S-				
A+				
A-				

Figure 3.14: Suggested tier list from reference [1]. Differs from Fig. 3.13 by varying the position of Absa and Etalus. Graphic made using resource in reference [3].


S+	
S-	
A+	
A-	

Figure 3.15: Same as Fig. 3.13, but with selecting the mode for Elliana and Etalus and 5 and 11 respectively. Graphic made using resource in reference [3].

Part IV

Data cleaning by presorting tier lists

4.1 The need for data cleaning

In the previous parts of this work, we saw that there were a lot of provisions needed for tie-breaking modes for some of the characters. In this example, it did not produce results too dissimilar from what the top players produced in reference [1], but it may be problematic for other data sets. One of the solutions to bad looking data (i.e.: not reasonably distributed) is to increase the sample size. However, for the task in question increasing the sample size produces bias in the data. This is because tier lists intend to represent the top level of play, and there are more mid to low level players than top players.

That being said, there may be mid level players who understand conceptually what the top level play should play like, but just do not perform at that level. This can be due to a variety of factors such as genetics, injuries or conditions (which can limit reaction times, hand to eye coordination, hand speed, etc.), or the lack of motivation to compete for the top level spots. They could even be players that play other platform fighters that do not compete in Rivals, but through their understanding of the platform fighting genre can place the characters reasonably accurately. If we just census the top players, we leave out this group of players, whose lists can help make the statistics more palpable for the analysis.

On the other hand, there may be biases within the top players that need to be identified. Highly anomalous lists did not affect the modal analysis in the previous parts of this work, since their placings are so far from the mode. However, more medium anomalies which are more agreeable could misrepresent the situation, and may impact the modal analysis of the data. Even if we limit the sample set to top players only, there are still biases that can present themselves.

It is clear that for whatever sample set we use, there is a benefit for presorting the data by game knowledge, which is not always correlated with the player's ranking. Luckily, we can use K -modes analysis again, only this time by sorting the player's lists into clusters. We can then use some number of clusters that correlate with the more skilled players. For example, if some tier list is clustered with the top 10 player's tier lists, then there is at least a reasonable likelihood that that player understands the game to a similar level to those players, regardless of their ranking. By using player placements as indicators of understanding of the game, we can choose to only use tier lists belonging to clusters that include the top players. At the very least, we begin to address a scalable method of analysing tier lists, which can favour the lists that demonstrate higher levels of game understanding over those that do not.

4.2 Data cleaning method

For the data cleaning, I will use K -modes again to cluster the types of tier list, and look at the different clusters to see which ones could be removed. The removal criteria will be determined from finding the cluster(s) containing the tier lists that were removed in reference [1]. If a larger data set is considered, we might not have these markers for removal. However, if the tier lists are associated with some quantitative measure of skill level such as matchmaking rating (MMR), then one ought to remove clusters that have a large correlation with the lower MMR. Player rankings may also be used in this instance, although the game may not track player rankings too closely below the top 100 say.

Since we are now looking at clustering the tier lists, there will now be a modal value for each column. This is because each column represents all the player's ratings of a single character. This means that we can use the original method proposed in [2] (i.e.: the method described in section 2.2.2 of this work). After the clustering is complete, I will experiment with excluding certain clusters that may contain anomalies, and use the K -modes method from part III on the non-rejected clusters to determine tier lists.

4.3 Preliminary tests

For these tests, the number of clusters will be varied. The total dissimilarity of all the characters (a.k.a.: the K modes cost function) will be calculated. Since tie-breakers are decided at random, the average of 5 cost functions per value of K will be taken. The optimal K will be chosen using the elbow method and our intuition. Then, any clusters that contain the anomalous tier lists (according to the spreadsheet in reference [1]) will be removed, and the results will be ran through the previous code to generate tier lists for the characters.

I denote K_1 as the K value for clustering the tier lists, and K_2 as the K value for clustering the characters from the clusters that remain after removing anomalies. Firstly, before any clusters are removed, I use the elbow method to determine the optimal number of clusters. The initial cluster modes correspond to player ids, such that the range of player ids included are $1 : K_1$. Fig. 4.1 shows that the elbow is located at $K_1 = 3$, but removing the anomalous clusters at $K_1 = 3$ results in removing the tier list for the 2nd best player (at the time). Therefore, I opt to use $K_1 = 4$. Interestingly, two of the anomalous tier lists were consistently clustered together, with the other two having their own clusters.

With the anomalous clusters removed, K_2 is varied to see the optimal value for clustering the characters. Fig. 4.2 shows that for $K_1 = 4$, $K_2 = 4$ is a sensible choice.

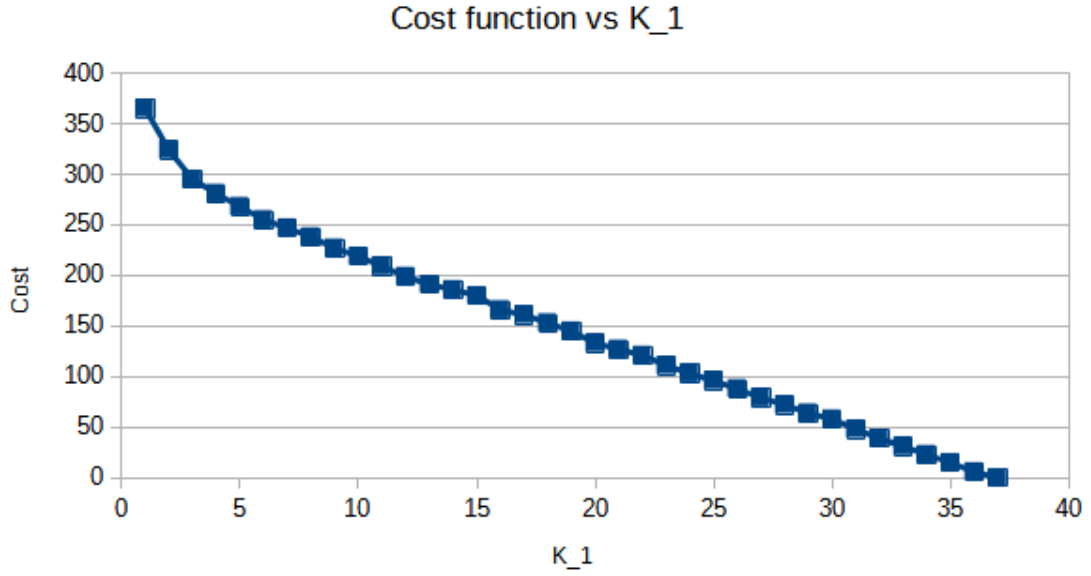


Figure 4.1: Cost function vs. K_1 , which is the cluster number for clustering the tier lists. The cost function is calculated before clusters that contain anomalous tier lists are removed. $K_1 = 3$ is the location of the elbow. Due to random cluster allocations for tie-breaking, each data point is the average of 5 runs.

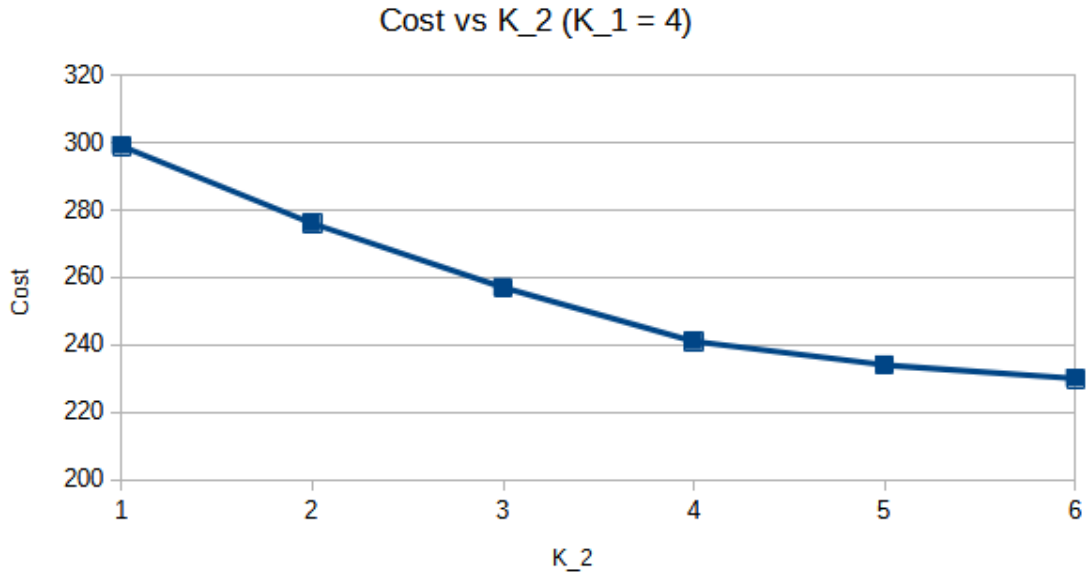


Figure 4.2: Cost function vs. K_2 for $K_1 = 4$. K_2 is the cluster number for clustering the characters, which results in a tier list. $K_2 = 4$ is the location of the elbow.

However, this does not produce a stable/converged tier list. This is due to lots of tie-breaking, which removes lists inconsistently. Since I am reducing the sample size which is already low at this point, this means that the inconsistent removal of certain lists between runs has a strong effect on the results. This method may be useful if applied to a larger sample size as I explained above, but it cannot be used solely on the top player's lists due to these inconsistent results. I also tried using the K -modes function from the klaR library [4], but it had worse average cost as well as inconsistent results.

In order to address this, I am extending the concept that was used before for the character clustering. If we encounter a tie in the dissimilarity, we can calculate the proximity of object X_i to the cluster mode by iterating j from 1 to m , and summing the absolute value of $x_{i,j} - q_{k,j}$. This is similar to the K -means method, but in this instance it is only invoked on tie-breaking. This was used previously in part III, but in that instance the character's mode and the singular cluster mode were compared. Using this proximity metric does enable the code to converge. This proximity to the cluster is not a bad metric to use in this instance. This is because rankings, whilst categorical, are related to one another. For example, a 3rd place character is closer performance wise to a 1st place character than a 14th place character, so the proximity metric for ranking data is meaningful because the ranks are related to one another with a scale. On the other hand, if one of the columns was a player's favourite fruit out of apples, oranges and grapes, then proximity metric has no bearing here. One cannot say that apples are closer to oranges than they are to grapes - they are totally unrelated categories.

Using the new proximity metric for tie-breaking, the cost function vs. K_1 is shown in Fig. 4.3. The elbow in this case is located at $K_1 = 4$, but we can investigate further than that. Due to the low sample size, I want to retain as many players as possible for the tier list calculation. This may not be the case for larger sample sizes, in which case one should use the elbow value of K_1 for removal. One can investigate the relationship between the remaining players and K_1 in order to determine which value to use, which is shown in Fig. 4.4. Whilst $K_1 = 4$ does retain a lot of the players in the calculation, we see some interesting features after $K_1 = 8$. There is a plateau in the number of players remaining after we remove the clusters. This basically arises because the size of the clusters that are being removed have converged, and so adding clusters is merely splitting up the remaining clusters into sub-clusters. Using this graph, $K_1 = 9$ is a good value to use, since beyond that the remaining clusters are splitting into smaller clusters, which is not giving us more information.

Using $K_1 = 9$, the cost function vs. K_2 is shown in Fig. 4.5, with an elbow value of $K_2 = 4$ like before. Using these values, the resulting character tier list is shown in Fig. 4.6((a)), which has several differences from Fig. 3.13. Firstly, Maypul and Ori's position are swapped. In the list in Fig. 3.13, they were both holding the 2nd place simultaneously in that cluster, but in Fig. 4.6((a)) Maypul is firmly a place ahead of Ori. Elliana has improved her placing to be behind Absa, and Shovel Knight has increased a tier to be ahead of Forsburn. For the last tier, Sylvanos and Kragg have swapped places. Between Fig. 3.13 and Fig. 4.6((a)), only Shovel Knight moved

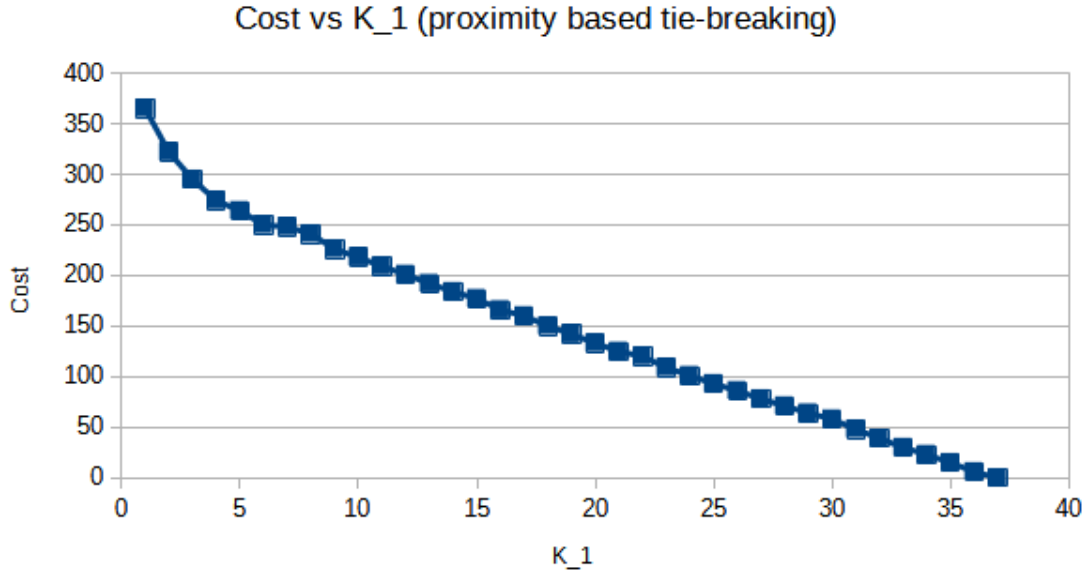


Figure 4.3: Same as Fig. 4.1, but using the proximity to the cluster for tie-breaking in cluster allocation. Elbow located at $K_1 = 4$.

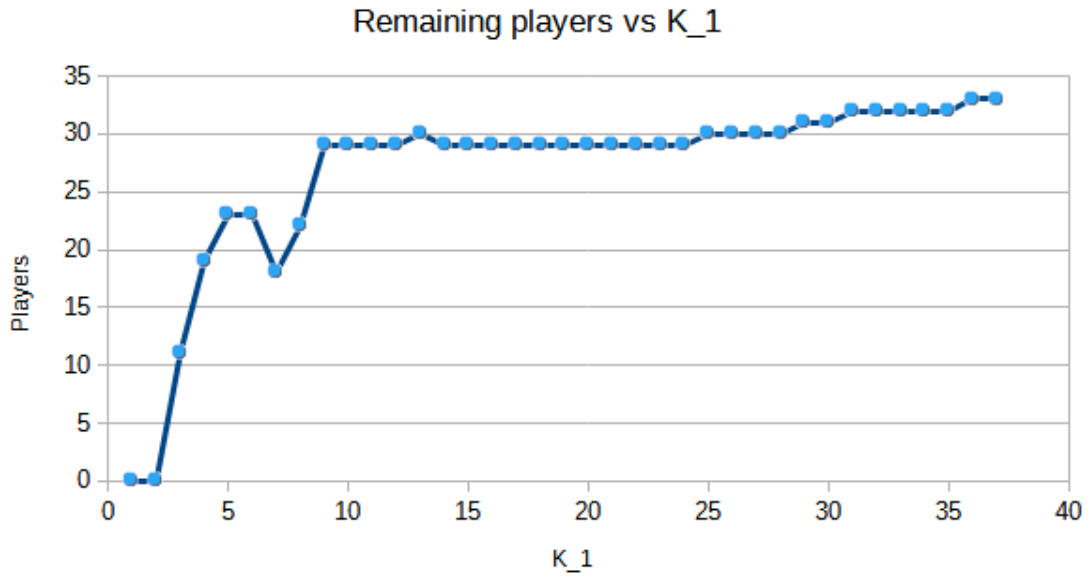


Figure 4.4: Remaining players after anomalous cluster removal vs K_1 . At and beyond $K_1 = 9$, the remaining players plateau, which indicates at the occurrence of sub-clustering processes (see text). Using this and the elbow method, $K_1 = 9$ is a sensible choice.

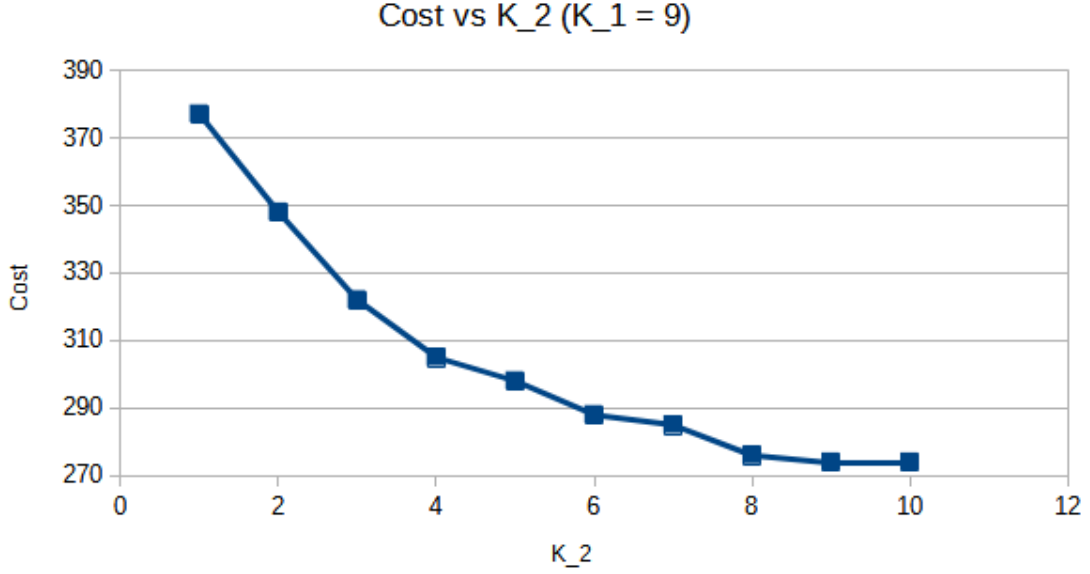


Figure 4.5: Same as Fig. 4.2, but using the proximity to the cluster for tie-breaking in cluster allocation. Elbow located at $K_2 = 4$.

tiers, so there is reasonable consensus amongst the two lists from this work.

For a more direct comparison, Fig. 4.6((b)) shows the tier list generated with solely the anomalies removed. The consensus is reasonable, with Elliana’s and Shovel Knight’s positions being the only differences. This suggests that if the sample size increases, the cluster removal method can be used to curate the sample set effectively, to ensure that the top level of play is mostly represented.

4.4 Conclusion

The small sample size was a limiting factor in the previous study, since there were ties in the modal values for some characters. Ideally, the sample size should be increased, but this can introduce heavy biases that favour lower skilled players. Tier lists need to represent the top level of play, but increasing the sample size increases the amount of tier lists provided by players with skill less than said top level. In order to weight the top level tier lists more than the lower level ones, I opted to use K -modes again, this time on the set of tier lists. Tier lists similar to those identified as anomalous in reference [1] were removed, by removing the whole cluster associated with them.

However, because there were a lot of ties in cluster allocation (which were tie-broken randomly), player’s tier lists did not converge to the same cluster each time, which means that the tier lists being removed each run were not consistent. The change in the results due to different lists being removed each run was strong because the



Figure 4.6: Character tier list using ((a)) $K_1 = 9$ and $K_2 = 4$, with anomalous clusters removed ((b)) $K_2 = 4$, with no anomalous lists. Differs from Fig. 3.13 in the positions of several characters.

original sample size is small. Whilst the elbow method showed optimal values for K_1 and K_2 , neither of these values produced consistent results for the tier list. This meant that I had to employ a more deterministic tie-breaking metric, which was chosen to be the proximity to the cluster mode. This produced consistent auxiliary results, which produced a tier list that was different to the previously calculated lists.

Initially, I used K -modes due to its more appropriate treatment for the categorical data. However, due to the specific nature of the problem I have developed cluster allocation and cluster mode calculation techniques that I have not seen or researched before, that had some logic behind them at the very least. This makes it difficult to fully trust either set of results. My K -modes method is more suitable for categorical data, but on the other hand the results of my work are difficult to verify as ‘correct’, since they depart from the research. This being said, the lists are all reasonably similar to one another and are serviceable tier lists.

Using K -modes or some clustering method to select tier lists that are closer to the top player’s lists may prove very useful, provided that the sample size of tier lists increases, which requires a larger census. If one looks at the ELO/MMR distribution per cluster, then they can filter out lower skilled players from affecting the results, whilst still getting better sample sizes for the data analysis.

Part V

Iterative procedures and project summary

5.1 Addressing convergence

In the previous parts, I devised several ways to generate tier lists from the top player data. This included just using the raw data, removing data associated with anomalous tier lists, and simply removing the anomalous tier lists. Each method has specific advantages and applications, with the first method requiring the least specialist knowledge of the game. Whilst the method used in part IV utilised the specialist knowledge of the anomalous tier lists from reference [1], it could easily be adapted to use another metric such as average cluster ELO/MMR, which requires less insight into the top player's lists. This cluster removal method was designed to allow the sample size to be increased, without diluting the top player results. For each of these three main methods, there are two methods associated with it, which are the harsh and fair mode selection for modes that tie. This was necessary to introduce in order to reduce randomness in the data. This means that this work has used six total methods to address the problem.

It is quite disappointing that there was no convergent tier list with these six total methods. The procedure was thoroughly analysed using the elbow method and other metrics, and still there is not one definitive tier list that is method independent for this sample set. Since clustering tier lists based off of character ordering is relatively niche, it is difficult to get help on this issue. However, we have a very unique scenario where generated tier lists can be converted into a character ordering, which can be converted into a list. This begins an iterative process that we can at the very least investigate for convergence.

There are some rules of thumb that we should consider. The sample set should mainly consist of top player's tier lists. Their lists are significantly more important than generated lists since they possess the game expertise to generate the orderings in the first place. The player tier list orderings are generated from character match-up analysis, which our method does not consider. Therefore, if the generated tier lists converge only after a significant number of previously generated tier lists are added, then it is not a serviceable list. These added tier lists can be interpreted as an approximation of tier lists that top players would produce, if more were censused. Generally speaking, the generated tier lists should be agreeable and represent the top player's views due to them being derived from that data set.

5.2 THE tier list

The top player data was used to generate six K -modes based tier lists, which correspond to the methods described in the above section. As we have seen in this document, a consensus between these six methods pertaining to the ordering and the clustering of the characters was not reached. For simplicity, I used $K_1 = 9$ and $K_2 = 4$ for all lists, which is chosen based on the analysis in the previous parts. The six K -modes tier lists plus the K -means tier list from reference [1] were then added

S+	
S-	
A+	
A-	

Figure 5.1: The definitive tier list from this work. Generated by adding six K -modes and one K -means generated tier lists to the original sample set. This list is the modal list of a set of six generated lists, and it closely follows Fig. 4.6((b))

to the data set, and the six methods were used again to generate another six tier lists. Again, these additional tier lists can be interpreted as an approximation of the tier lists that top level players would provide if censused. This time, the consensus was much stronger, with only Elliana and Ranno switching places. Since iterating through more tier lists breaks the rule of thumb described in the above section, the modal list of the six was taken. This, which is THE tier list as far as this project is concerned, is shown in Fig. 5.1.

5.3 Project summary

This has been a thoroughly enjoyable project for me. I started by attempting to improve the data analysis of the data, by using K -modes clustering to remove ambiguities. This lead to having to adapt and customise the method using simple assumptions in order to properly cluster the data.

The EDA showed that the base sample set was riddled with troublesome data, which prompted me to investigate a double K -modes method that can remove anomalies

and curate the data before generating a tier list. This method looks promising for situations where the sample size is increased, since it can be used to remove tier lists from lower skilled players, which is vital in order to produce an accurate tier list. The double K -modes method produced a tier list that closely followed the K -modes tier list with anomalous lists removed.

By using a small handful of the generated lists from these methods, a stable tier list was generated. I imagine that if the double K -modes method is employed on a large enough sample set, then one would not need to iteratively add generated tier lists to the data set. However, since the sample size has not increased, or if it cannot, this can be useful in stabilising the lists if there is no consensus amongst methods.

I may revisit this project if a new census is released, since Rivals of Aether just got 4 new characters added to the roster. Hopefully the methods developed for this data set can be used to address the tier list problem for the new census data. Also, an interesting project to consider is to link the matchup knowledge into a character ordering.

Bibliography

- [1] “The OFFICIAL Rivals of Aether Tier List.”
https://www.youtube.com/watch?v=tL_gXjGkEU4
- [2] Z. Huang, “Extensions to the k-means algorithm for clustering large data sets with categorical values,” *Data Min. Knowl. Discov.*, p. 283, 1998.
- [3] “Tier Maker website.”
<https://tiermaker.com/create/rivals-of-aether>
- [4] “kmodes function - RDocumentation.”
<https://www.rdocumentation.org/packages/klaR/versions/1.7-0/topics/kmodes>