Project2

James Wu

January 23, 2018

Contents

1 Background

1.1 Data

The data I'm using is a generated features list derived from a single match in Heroes of the Storm. Using the 1260 relational distance matrices (1 matrix for every second the game lasted), I extracted the minimum, maximum, and median values for the Blue Team, Red Team, and the opposing team members. For now, I'm omitting all the rows of data that has NAs (times when only 1 or 0 people are alive in the same team). [will give more information here]

1.2 Goal

The goal of this study is to properly identify timepoints when teamfights and skirmishes occur. Given that there is no actual classification or labelling given to me, I watched the replay myself and determined when the skirmishes and teamfights are based on my own intuition. As a disclaimer, I found it very difficult sometimes to determine the difference between skirmishes and teamfights, and thought that most player interactions in the game I watched were more skirmishes than teamfights. As such, for now I will have all the timepoints I identified as 'teamfights' rather than skirmishes. Further refinement to my definition and the study will need to happen.

2 Exploring the bivariate plots

First, we explore the pairwise plots to see whether there are any clusters that can be seen in the feature pairs. Then the densities of the pairs will be explored to see whether any feature is very skewed or need other transformations.

2.1 Pairwise Plot

```
> pairs(PositionFeatures[,2:9],pch=".")
```

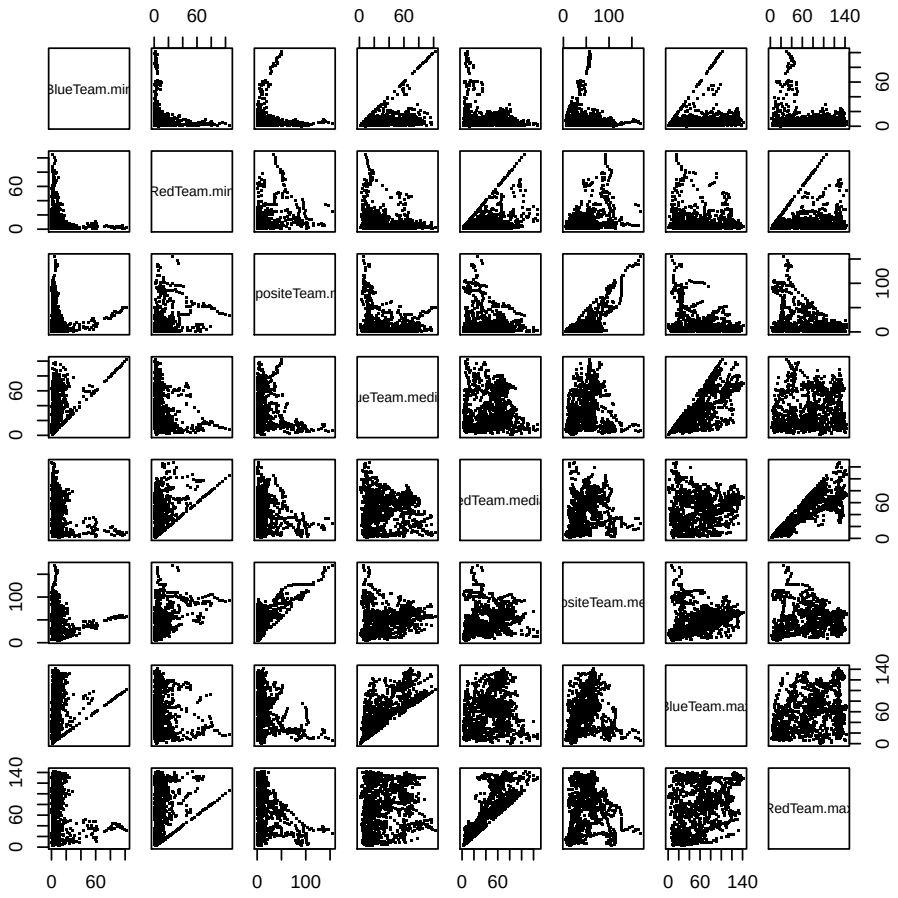


Figure 1: Feature Pair Plots

Here, we do not see any separation to indicate clear clusters.

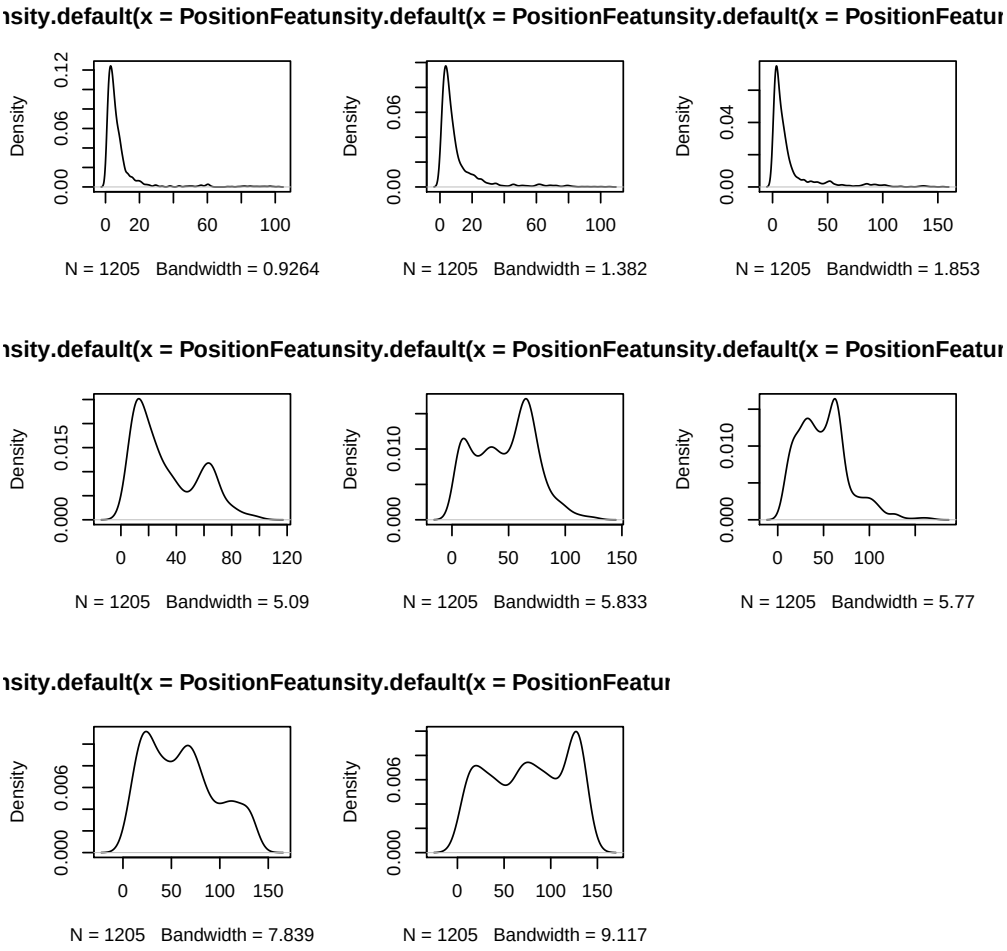


Figure 2: Feature Densities

The minimum features are right skewed, and the median and maximum features seem to be multimodal. This is possibly be a good thing, and could mean that there may be clusters in those variables. Given that all of these variables are on the same scale, I will not be making any transformations.

2.2 Principal Component Analysis

I want to see whether any clusters could be identified via PCA. We will naturally see strings of points due to time-dependent nature of the points. The minimums and medians of a single timepoint will be similar to the timepoint before it. For now, I will be only looking at the minimums and medians because it's difficult to justify what maximums could represent without it being relative to something else.

```
> pc.positions <- princomp(PositionFeatures[,c(2:7),])$scores
> pairs(pc.positions,pch=".")
```

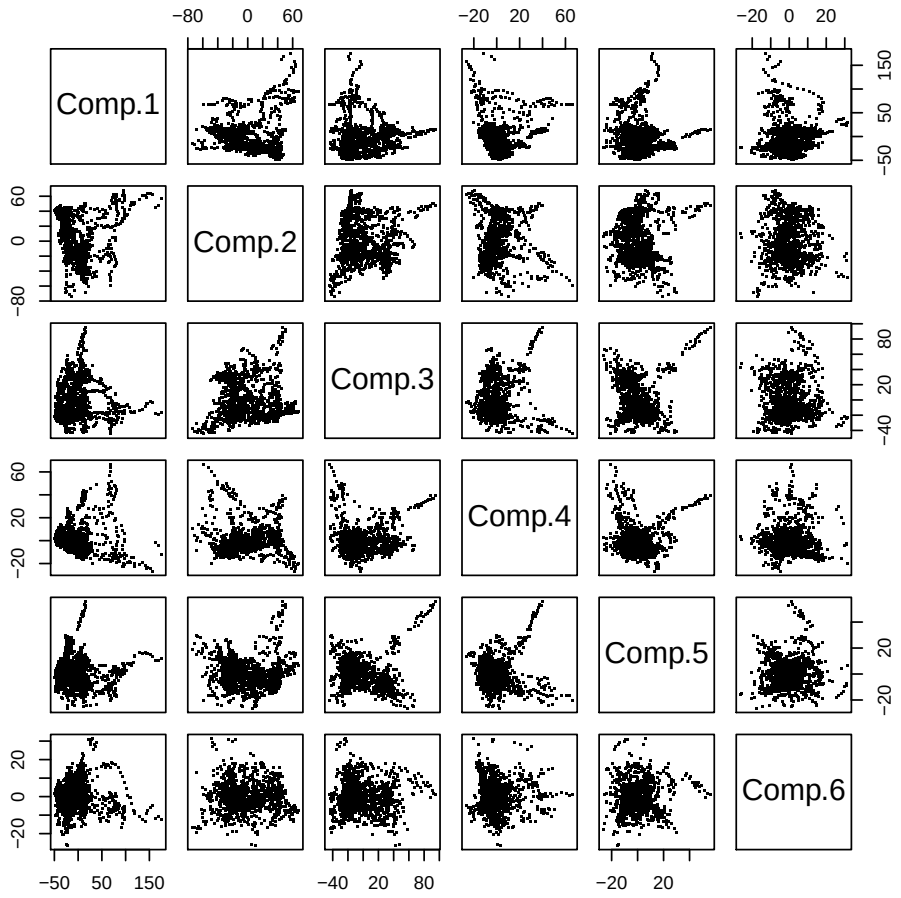


Figure 3: Pairs in Principle Component Space

```
> princomp(PositionFeatures[,c(2:7)])$loading

Loadings:
                    Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6
BlueTeam.min                 0.124  0.300  0.454  0.735  0.383
RedTeam.min          0.186 -0.127 -0.189  0.868 -0.395
OppositeTeam.min     0.600  0.343                0.335 -0.634
BlueTeam.median             -0.160  0.899        -0.196 -0.350
RedTeam.median       0.118 -0.903 -0.148         0.340 -0.175
OppositeTeam.median  0.768         0.187 -0.184 -0.193  0.542

                Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6
SS loadings      1.000  1.000  1.000  1.000  1.000  1.000
Proportion Var   0.167  0.167  0.167  0.167  0.167  0.167
```

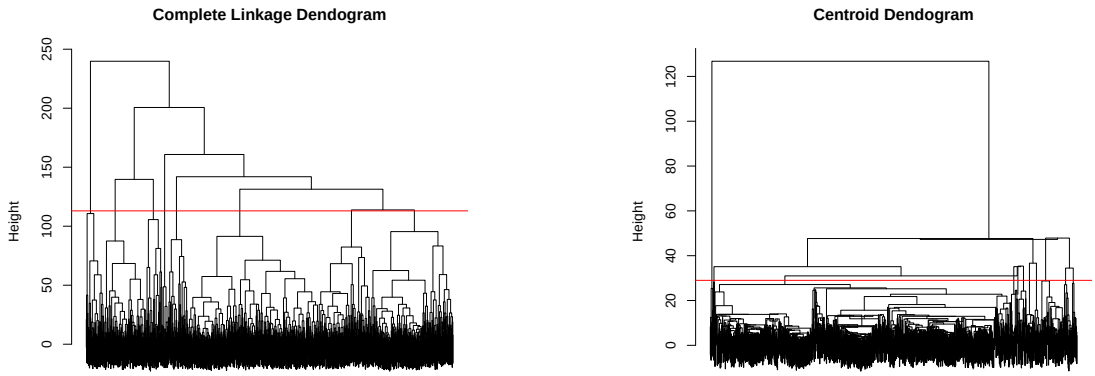
```
Cumulative Var  0.167  0.333  0.500  0.667  0.833  1.000
```

Just looking at the PCA pairs, it seems there may be some clusters that could be unraveled. This will be explored in the following section. The loadings tell us that there isn't any space that only uses one specific metric (median or minimum), which is probably good.

3 Clustering

In this section, I will be exploring different clustering methods to use as a possible guide for the number of labels I may want to include for the classification analysis. Since each timepoint's features should be highly correlated with the previous timepoint, it may be useful to reduce the number of points by skipping every n points. Doing so may improve the separation of the clusters and improve the results from the clustering methods.

3.1 Hierarchal Clustering



(a) Complete Linkage Dendrogram

(b) Centroid Clustering Dendrogram

Figure 4: Complete and Centroid Dendrograms

Both the complete linkage method and centroid clustering show many possible clusters - around 9/10 clusters from just looking at their dendrograms. The single linkage method predictably put most of the points in a single cluster (Appendix A).

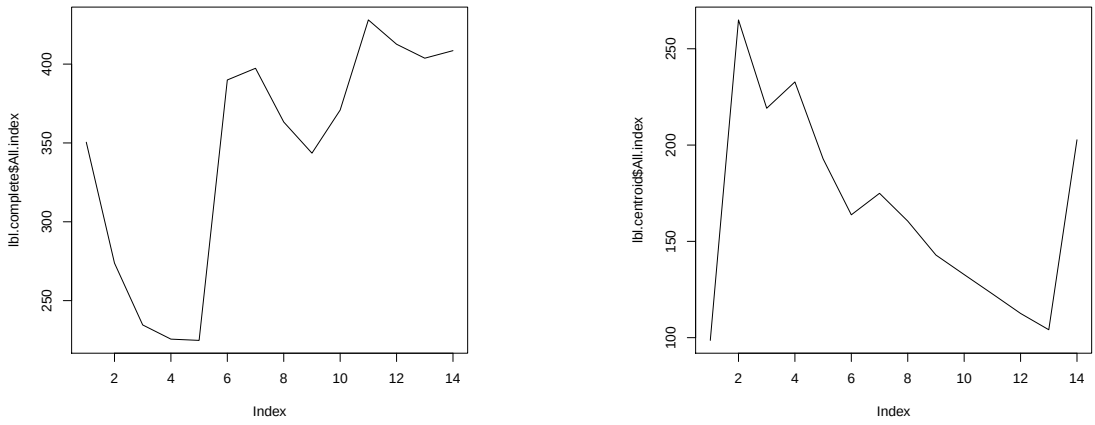
When I only include every 5th point, the complete linkage's dendrogram did not change much, but the centroid dendogram drastically changed (Appendix B).

In an ideal clustering situation, we would observe small within group variance and large between group variance. An adequacy index was developed (C(g)) to better choose the optimal number of clusters. The larger the index is, the larger the ratio of the between group variance and the within,

which represents the 'ideal clustering situation' mentioned earlier.

These are the values of C(g) for different values of groups for these two methods:

```
> lbl.complete <- NbClust(PositionFeatures[,2:7],method='complete',index='ch')
> lbl.centroid <- NbClust(PositionFeatures[,2:7],method='centroid',index='ch')

> plot(lbl.complete$All.index,type='l')

> plot(lbl.centroid$All.index,type='l')
```



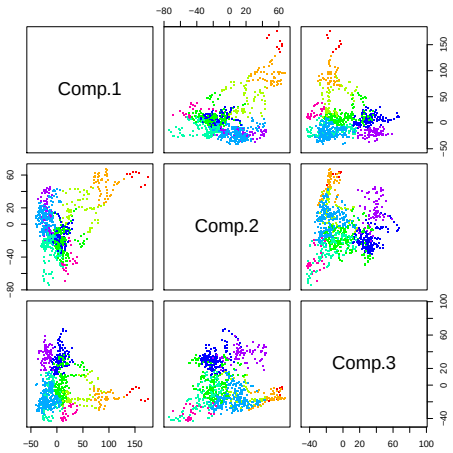
(a) Complete Linkage C(g)

(b) Centroid Clustering C(g)

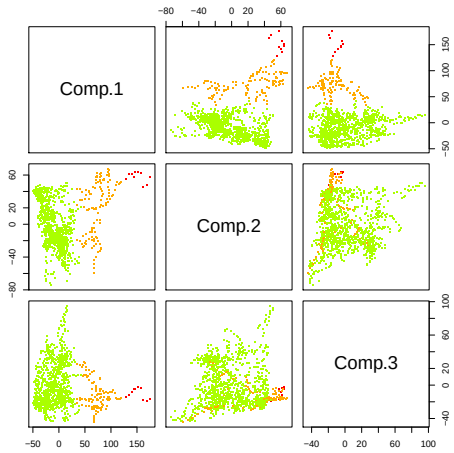
Figure 5: C(g) Criterion Plots

We see a maximal C(g) at 12 cluster in the Complete Linkage method, and a maximal C(g) at 3 clusters in the Centroid Clustering method. I cannot say which one is better for sure, but it seems as if the 3 cluster solution separates decently well in Principle Component Spaces 1 and 2. The clusters revealed here could be interpreted as my theorized 3 things that happen at any one point: 'nothing', 'skirmish', and 'teamfight'. However, it's also very likely these are not the only things, which the complete linkage method reveals.

```
> pairs(pc.positions[,1:3],col=rainbow(9)[lbl.complete$Best.partition],pch=".")

> pairs(pc.positions[,1:3],col=rainbow(9)[lbl.centroid$Best.partition],pch=".")
```



(a) Complete Linkage maximal C(g) in PC space



(b) Centroid Clustering maximal C(g) in PC space

Figure 6: Maximal C(g) labels pairs plot in PC space

3.2 k-means

k-means clustering is a popular clustering method aiming to partition n observations into k clusters, where each observation belongs to a cluster with the nearest mean. By applying this method to my dataset, I hope to compare its results to the centroid/complete clustering methods used earlier to see which method found similar clusters.

```
> lbl.kmeans <- NbClust(PositionFeatures[,2:7],method='kmeans',index='ch')
> #lbl.kmeans.2 <- kmeans(PositionFeatures[,2:7],2)$cluster
> plot(lbl.kmeans$All.index,type='l')
```

The most optimal number of clusters according to the criterion, is at 4 clusters.

```
> #plot(pc.positions[,2:3],col=rainbow(9)[lbl.kmeans$Best.partition],cex=.75, main = "kmeans 4-
> pairs(pc.positions[,1:3],col=rainbow(9)[lbl.kmeans$Best.partition],pch=".")
```

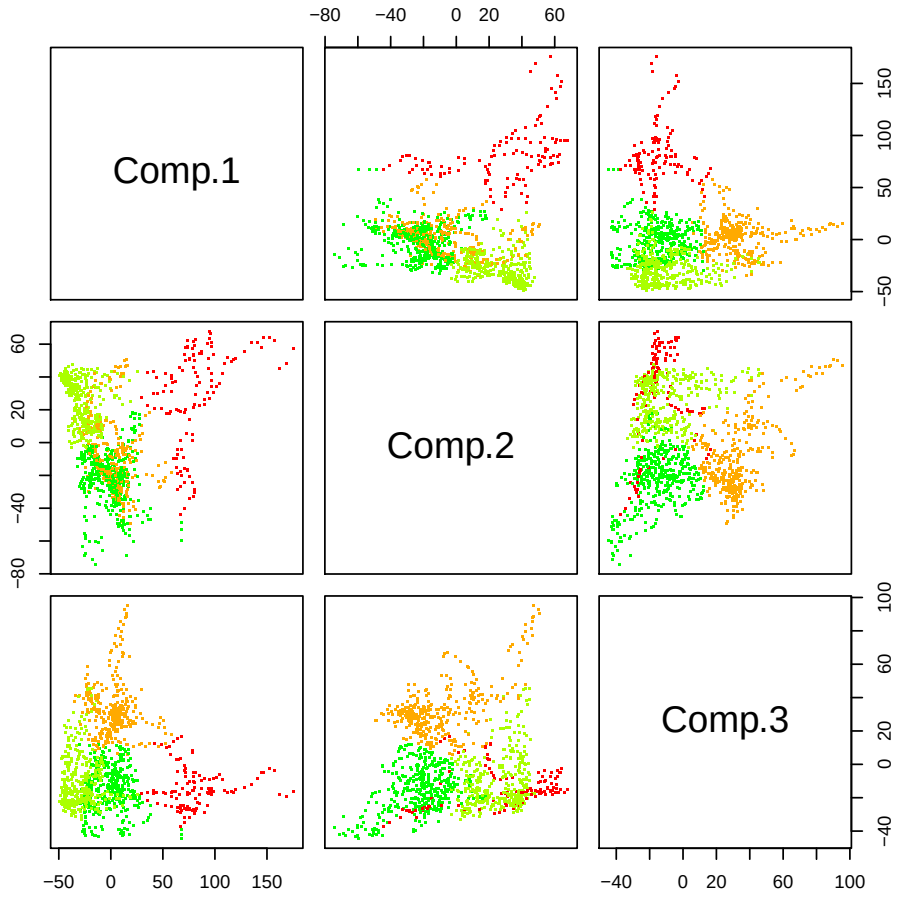


Figure 7: Kmeans labels in Principle Component Space

4 Exploration of Results

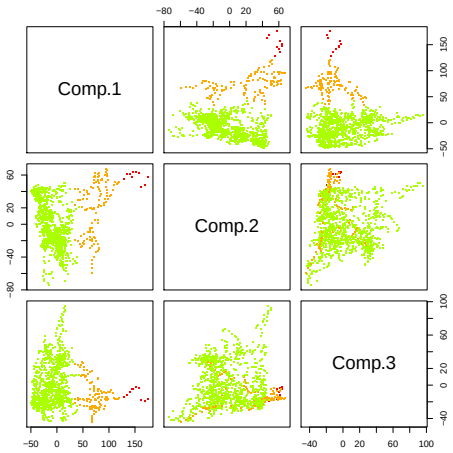
The next logical step in clustering here is seeing whether the clusters are actually clustered in any meaningful way. To do this, I will explore several position plots corresponding to the timepoints that were clustered in the optimal C(g). I will be focusing on the centroid clustering results and the kmeans, as they both have similar maximal g (3 and 4 respectively). I did a brief analysis on the Complete Linkage results, and you can find that in Appendix C
The optimal number of timepoints these two methods agreed upon is the following:

```
> optLabel(lbl.centroid$Best.partition,lbl.kmeans$Best.partition)$best.tbl
```

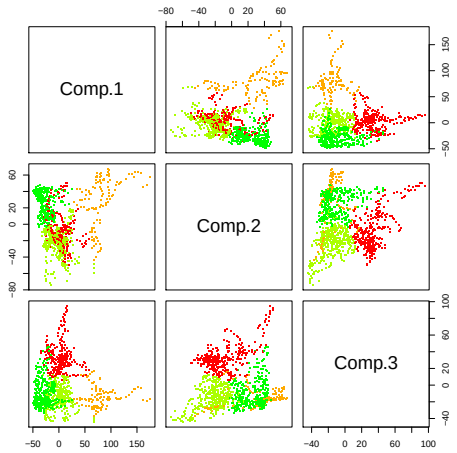
```
   trg
src   2   1   4   3
  1   0  10   0   0
  2  11 106   3   0
  3 328   2 396 349
```

Here we see only 502 out of 1205 points are put in the same clusters by these two methods. While this is not great, in the two clusters they matched the most in, had only a total of 15 differences. It also may be worthwhile to see whether the methods produced clusters that could be interpretable for me.
Here is the pairs graph after relabelling for optimal number of matches:

```
> lbl.kmeans.opt <- lbl.kmeans$Best.partition
> lbl.kmeans.opt[which(lbl.kmeans$Best.partition == 1)] <- 2
> lbl.kmeans.opt[which(lbl.kmeans$Best.partition == 2)] <- 1
> lbl.kmeans.opt[which(lbl.kmeans$Best.partition == 3)] <- 4
> lbl.kmeans.opt[which(lbl.kmeans$Best.partition == 4)] <- 3

> pairs(pc.positions[,1:3],col=rainbow(9)[lbl.kmeans.opt],pch=".")
```



(a) Centroid Clustering maximal C(g) in PC space



(b) Kmeans maximal C(g) in PC space

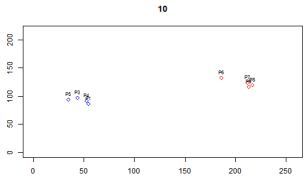
Figure 8: Maximal C(g) labels pairs plots in PC space

4.1 Centroid Clustering Results

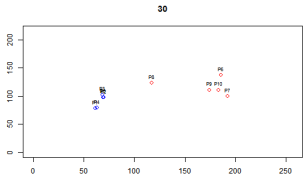
C(g) had its absolute maximum at g = 3 clusters here.

I will take 4 randomly-drawn sample of the TimePoints in each cluster:

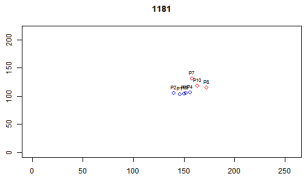
```
> ClusterSample <- function(methodlabels, numSample=1){
+   maxClust <- max(methodlabels)
+   Clustnum <- vector()
+   Sample <- matrix(rep(0,numSample*maxClust),nrow = numSample,ncol = maxClust)
+
+   for(i in 1:maxClust){
+     Sample[,i] <- sample(names(methodlabels[methodlabels == i]),numSample,replace=F)
+   }
+   return(Sample)
+ }

> set.seed(5)
> ClusterSample(lbl.centroid$Best.partition,4)
```



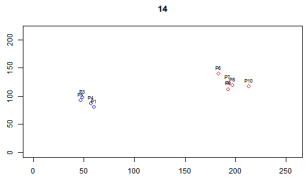
(a) Cluster 1 - Frame 10



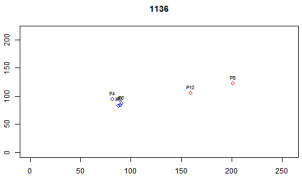
(b) Cluster 2 - Frame 30



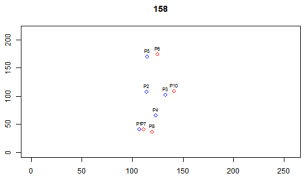
(c) Cluster 3 - Frame 1181



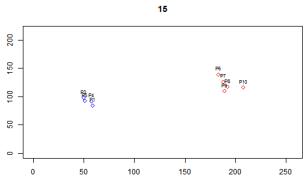
(d) Cluster 1 - Frame 14



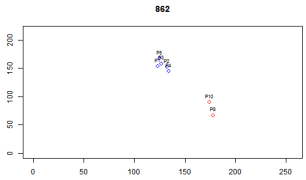
(e) Cluster 2 - Frame 1136



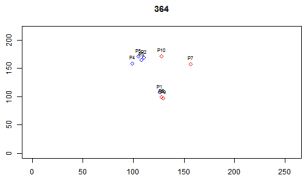
(f) Cluster 3 - Frame 158



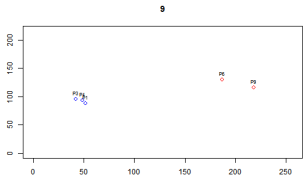
(g) Cluster 1 - Frame 15



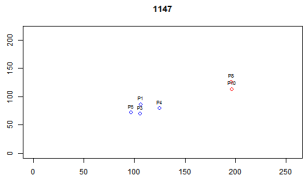
(h) Cluster 2 - Frame 862



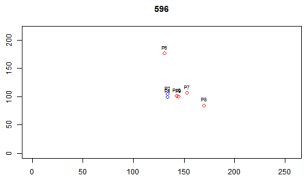
(i) Cluster 3 - Frame 364



(j) Cluster 1 - Frame 9



(k) Cluster 2 - Frame 1147



(l) Cluster 3 - Frame 596

Figure 9

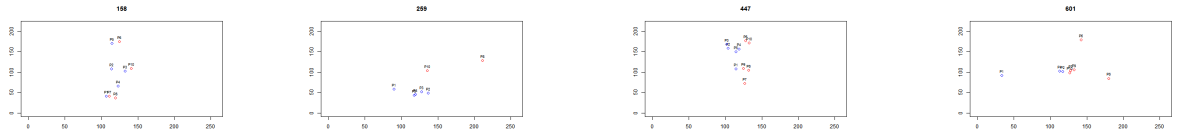
It almost seems here that cluster 1 is when players are furthest away, cluster 3 is when they are the closest, and cluster 2 is the middle ground.

4.2 k-means Results

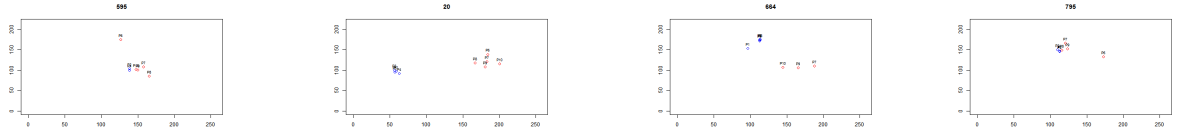
C(g) had its absolute maximum at g = 4 clusters here.

I will take 3 randomly-drawn sample of the TimePoints in each cluster:

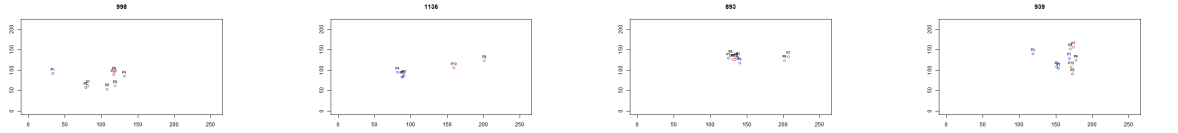
```
> set.seed(5)
> ClusterSample(lbl.kmeans.opt,3)
```



(a) Cluster 1 - Frame 158 (b) Cluster 2 - Frame 259 (c) Cluster 3 - Frame 447 (d) Cluster 4 - Frame 601



(e) Cluster 1 - Frame 595 (f) Cluster 2 - Frame 20 (g) Cluster 3 - Frame 664 (h) Cluster 4 - Frame 795



(i) Cluster 1 - Frame 998 (j) Cluster 2 - Frame 1136 (k) Cluster 3 - Frame 893 (l) Cluster 4 - Frame 939

Figure 10

The position plots here make it a little harder to determine what the clusters may represent. It's worth noting that cluster 1 here does not have a optimal pairing with the centroid clustering's results, nor does cluster 4. Only clusters 2 and 3, and they do seem to just different gradients of closeness.

5 Conclusions

I cannot say which clustering method was objectively better than the other between complete linkage, centroid clustering, and kmeans without further study, but centroid clustering and kmeans produced the most similar results. Single linkage had almost every point in a single cluster, so it was probably not a good method to use for my dataset.

Clustering may have been improved by using a different method (such as PAM), or by including more relevant features or even just our maximum feature. One relevant feature may be hero damage rate. Teamfights are naturally times when players are dealing the most damage to each other in a short period of time, so including the rate of Hero (or player) damage may significantly separate teamfights versus other events. Unfortunately, I would need to do more parsing of the original replay file to find that information, but it's a goal I could work towards.

A Single Linkage Dendrogram

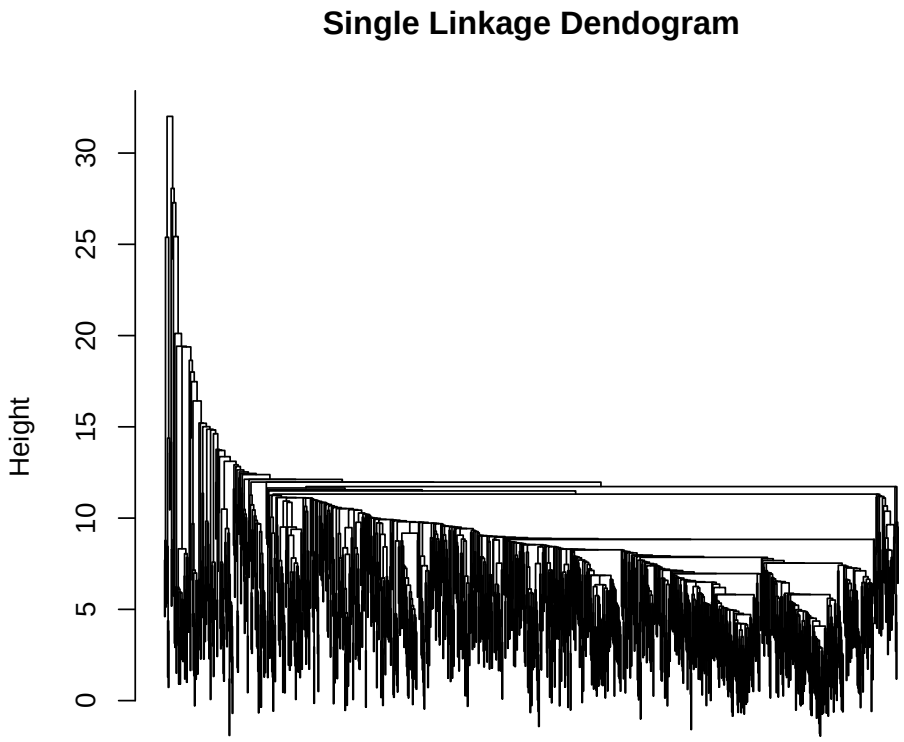
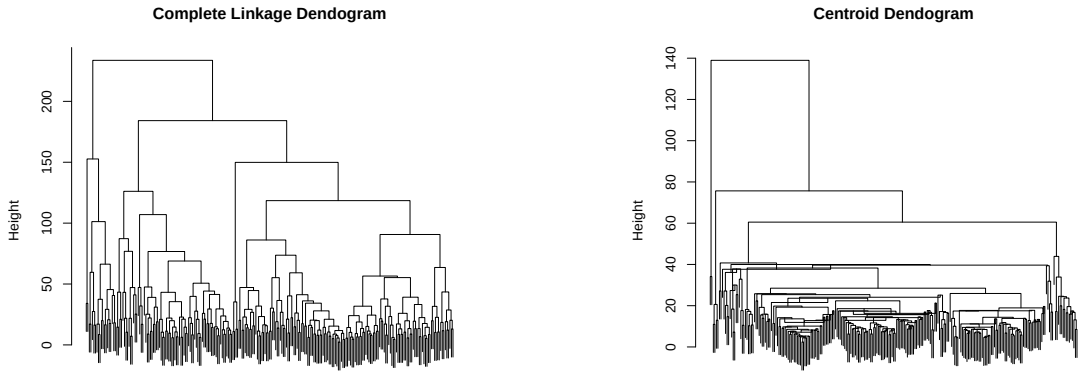


Figure 11: Single Linkage Dendrogram

B Only Every 5th Point



(a) Complete Linkage Dendrogram

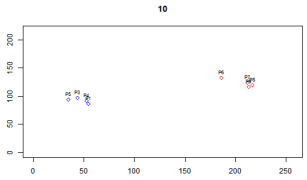
(b) Centroid Clustering Dendrogram

Figure 12: Complete and Centroid Dendrograms: With only every 5th point included

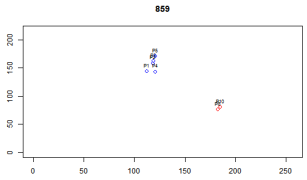
C Complete Linkage Results

When using the adequacy index C(g), we found that the 'optimal' number of clusters is 12.

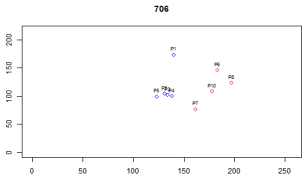
```
> table(lbl.complete$Best.partition)

  1   2   3   4   5   6   7   8   9  10  11  12
  9  51  65 172 138 266 190  79  31  22 163  19
```



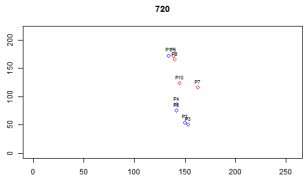
(a) Cluster 1 - Frame 10



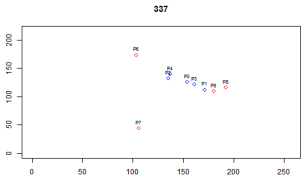
(b) Cluster 2 - Frame 859



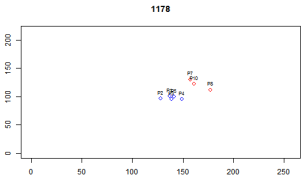
(c) Cluster 3 - Frame 706



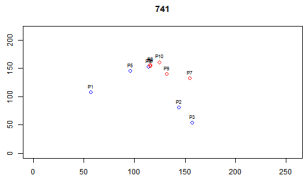
(d) Cluster 4 - Frame 720



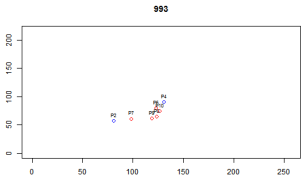
(e) Cluster 5 - Frame 337



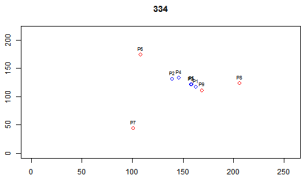
(f) Cluster 6 - Frame 1178



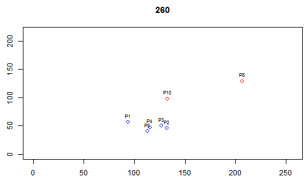
(g) Cluster 7 - Frame 741



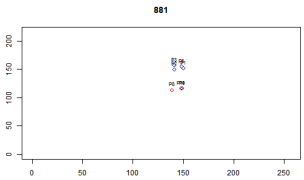
(h) Cluster 8 - Frame 993



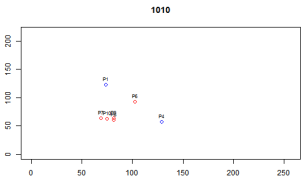
(i) Cluster 9 - Frame 334



(j) Cluster 10 - Frame 260



(k) Cluster 11 - Frame 881



(l) Cluster 12 - Frame 1010

Figure 13

This is curious, I find myself able to label most of these clusters (given the one sample) with something. It may be interesting to see whether the revealed structures here are actually better than my 3 theorized groups.