Assignment 2

```
library(tidyverse)

library(magrittr)

library(FactoMineR)
library(factoextra)

library(uwot)
```

Loading the dataset and just taking a glipmse to see if it looks right - and it does.

```
pokemon <- read.csv("https://sds-aau.github.io/SDS-master/00_data/pokemon.csv")
glimpse(pokemon)

## Rows: 800
## Columns: 13
## $ Number         <int> 1, 2, 3, 3, 4, 5, 6, 6, 6, 7, 8, 9, 9, 10, 11, 12, 13,…
## $ Name           <chr> "Bulbasaur", "Ivysaur", "Venusaur", "VenusaurMega Venu…
## $ Type1          <chr> "Grass", "Grass", "Grass", "Grass", "Fire", "Fire", "F…
## $ Type2          <chr> "Poison", "Poison", "Poison", "Poison", "", "", "Flyin…
## $ Total          <int> 318, 405, 525, 625, 309, 405, 534, 634, 634, 314, 405,…
## $ HitPoints      <int> 45, 60, 80, 80, 39, 58, 78, 78, 78, 44, 59, 79, 79, 45…
## $ Attack         <int> 49, 62, 82, 100, 52, 64, 84, 130, 104, 48, 63, 83, 103…
## $ Defense        <int> 49, 63, 83, 123, 43, 58, 78, 111, 78, 65, 80, 100, 120…
## $ SpecialAttack  <int> 65, 80, 100, 122, 60, 80, 109, 130, 159, 50, 65, 85, 1…
## $ SpecialDefense <int> 65, 80, 100, 120, 50, 65, 85, 85, 115, 64, 80, 105, 11…
## $ Speed          <int> 45, 60, 80, 80, 65, 80, 100, 100, 100, 43, 58, 78, 78,…
## $ Generation     <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, …
## $ Legendary      <chr> "False", "False", "False", "False", "False", "False", …
```

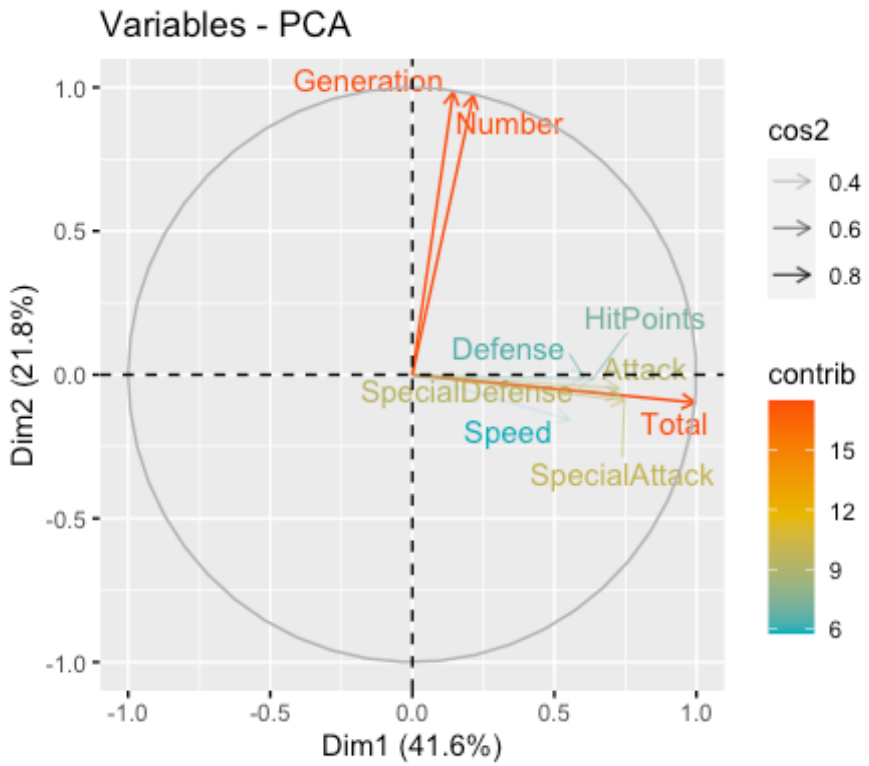
1: The dataset is containing different features about the pokemons. There is 800 different pokemons and the data contains 13 different variables, where the 11 variables is features such as “type”, “attack”, “generation” and if they are legendary or not. The variables is coded as intergers or characters. Looking at the scale of the data, it is not scaled yet. Even though it seems as if the variables are listed in the same unit it should be scaled to make sure that they are comparable.

2: The idea in the PCA is to reduce the diminality by creating linear combinations of the varaibles and thereby reduce the number of explanatory variables and in this way the principal components are uncorrelated/orthogonal. Performing a PCA on the numeric variables and scaling the data at the same time.

```
res_pca <- pokemon %>%
  select_if(is_numeric) %>%
  PCA(scale.unit = TRUE, graph = FALSE)
```

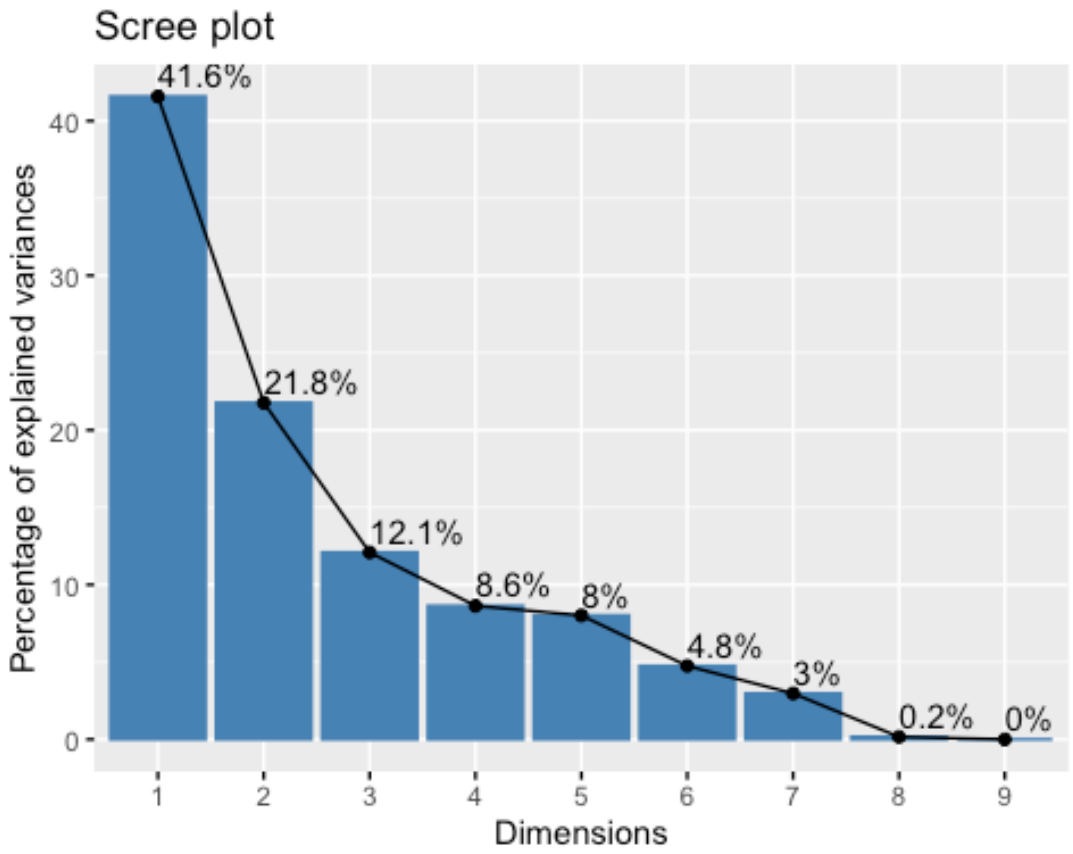
Plotting the variables and coloring them in order to see how much they influence the conponents - the longer the vector the more influence. At the same time it is possible to see how much the difference variables are correlated. The closer the vectors are, the more correlated – which here makes good sense since the different attacks and defense is more correlated with each other than with the number and generation. If the variables is orhtogonal they er uncorrelated. Note that in this data there is no negative correlation.

```
res_pca %>%
  fviz_pca_var(alpha.var = "cos2",
               col.var = "contrib",
               gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
               repel = TRUE,
               ggtheme = theme_gray())
```



Furthermore, looking at a scree plot it is possible to see how much each principal component captures of the data - here in percent. It can be concluded that the first 1-3 components captures just around 75% of the data.

```
res_pca %>%
  fviz_screeplot(addlabels = TRUE,
                 ncp = 10,
                 ggtheme = theme_gray())
```



Normally the number of components used to describe the data and therefore selected for further analysis is selected where the scree plot has an “elbow” which here properably would be around 3 components, but it is not really clear and therefore the eigenvalues is often used. The components with an eigenvalue over 1 are typically used. But in the assignment, we are specific asked to calculate the cumulative percentage of variance for the PCA with 4 principal components which is done below by calling the eigenvalues of the PCA.

```
res_pca$eig
```

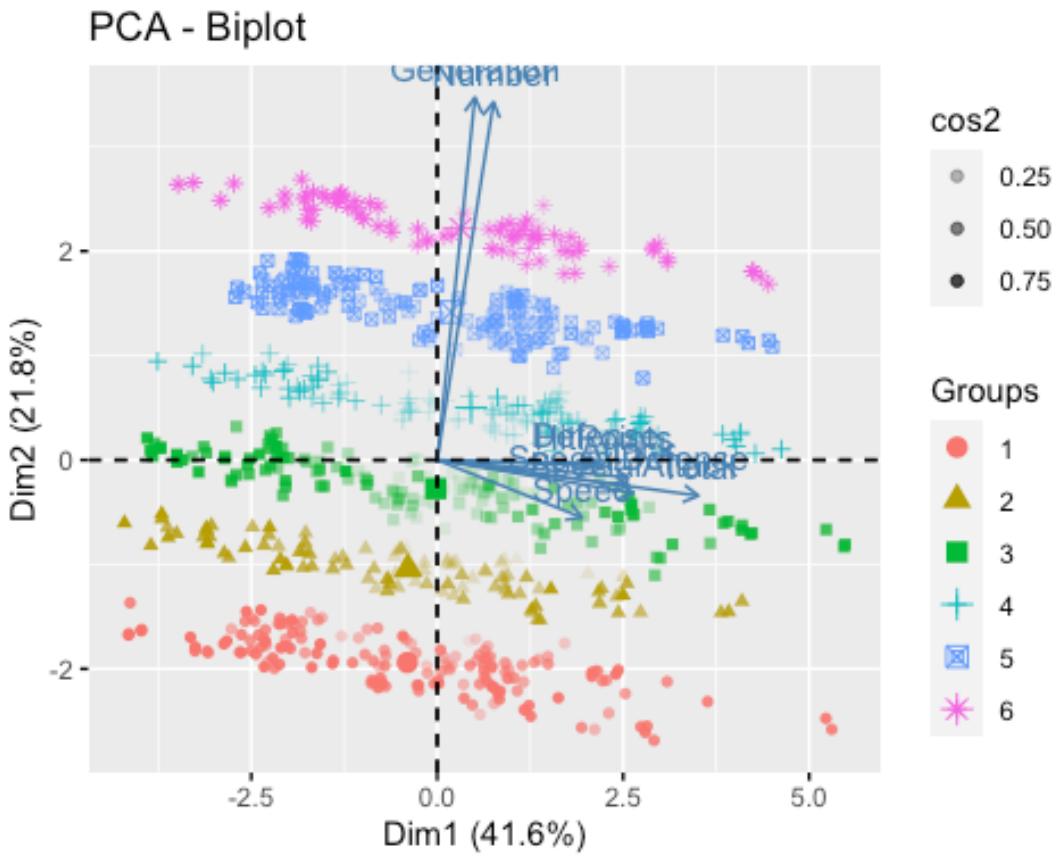
```
##          eigenvalue percentage of variance cumulative percentage of variance
## comp 1 3.742717e+00          4.158575e+01                          41.58575
## comp 2 1.959726e+00          2.177474e+01                          63.36048
```

```
## comp 3 1.088460e+00          1.209400e+01          75.45448
## comp 4 7.776318e-01          8.640354e+00          84.09483
## comp 5 7.215233e-01          8.016925e+00          92.11176
## comp 6 4.277021e-01          4.752246e+00          96.86400
## comp 7 2.674432e-01          2.971591e+00          99.83560
## comp 8 1.479640e-02          1.644045e-01         100.00000
## comp 9 6.628798e-31          7.365331e-30         100.00000
```

The cumulative explained variance ratio with 4 components is 84,1%. This can also be plotted but since it is well explained above, I see no reason.

Just a plot with the visualization of the biplot together with the observation colored by generation and it is seen that the first 2 principal components capture the tendens in the data fine.

```
res_pca %>%
  fviz_pca_biplot(alpha.ind = "cos2",
                  geom = "point",
                  habillage = pokemon %>% pull(Generation) %>% factor(),
                  addEllipses = FALSE,
                  ggtheme = theme_gray())
```



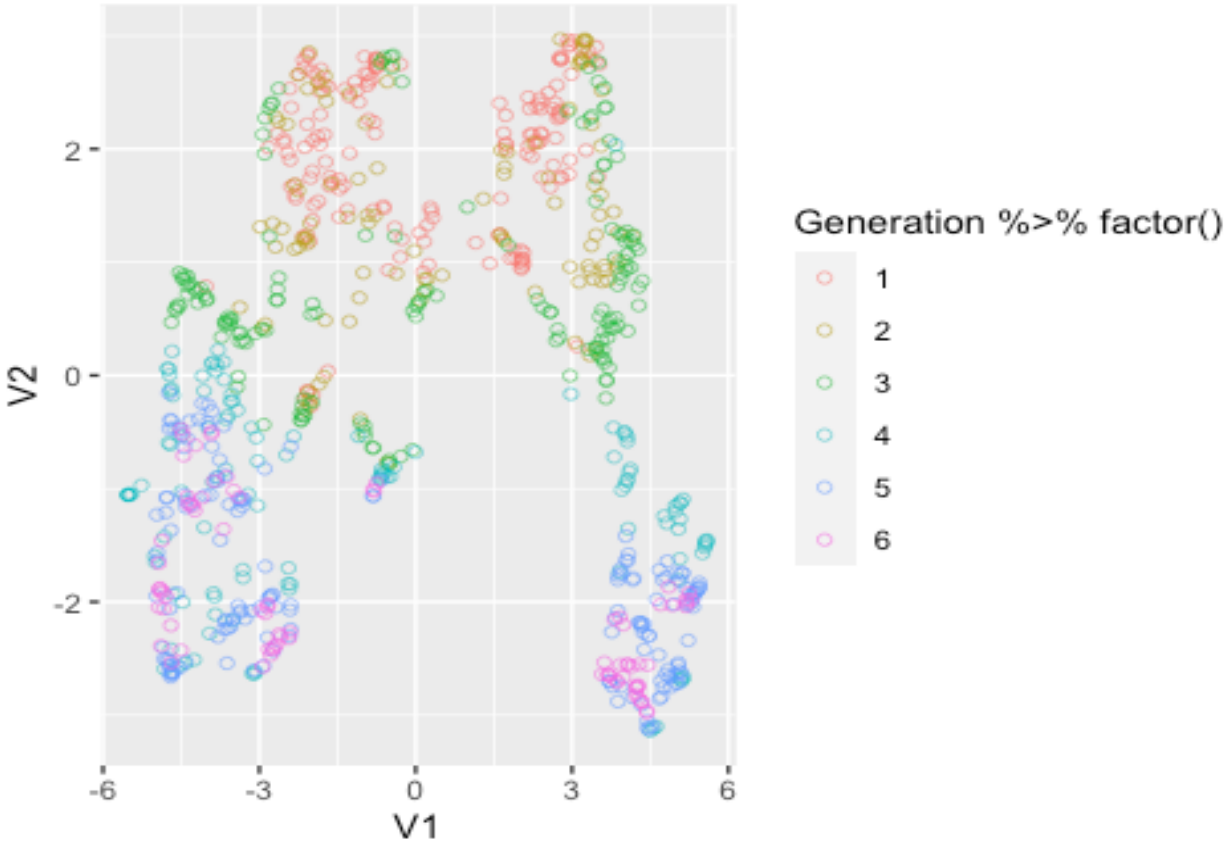
3: Using the UMAP method instead, where the distance is used to make a dimensionality reduction to 2 dimensions (default in the umap function i R).

```
res_umap <- pokemon %>%
  select_if(is_numeric) %>%
  umap(n_neighbors = 15,
       metric = "cosine",
       min_dist = 0.01,
       scale = TRUE)
```

Making the same plot as above with the components on the axes and coloring by generation.

```
res_umap %>%
  as_tibble() %>%
  bind_cols(pokemon %>% select(Generation)) %>%
  ggplot(aes(x = V1, y = V2, col = Generation %>% factor())) +
  geom_point(shape = 21, alpha = 0.5)
```

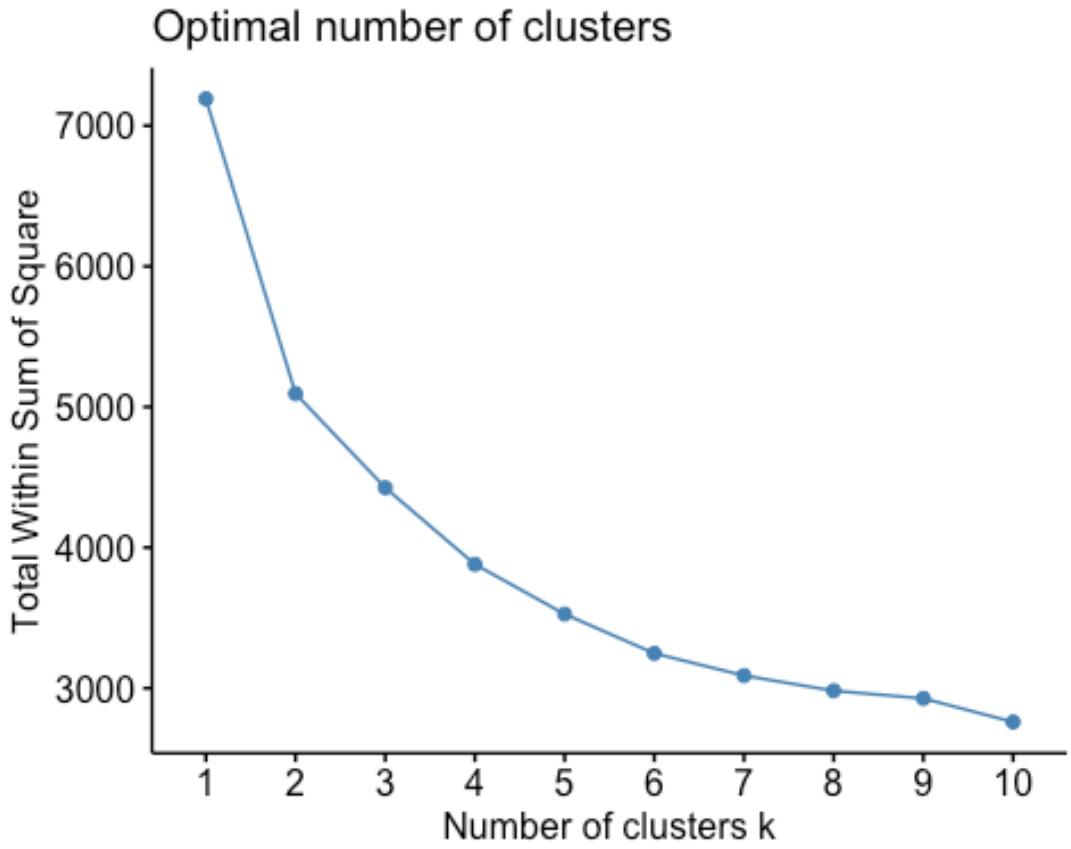
```
## Warning: The `x` argument of `as_tibble.matrix()` must have unique column names
if `.name_repair` is omitted as of tibble 2.0.0.
## Using compatibility `.name_repair`.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_warnings()` to see where this warning was generated.
```



It does not seem as if the UMAP function is able to distinguish as well as the PCA.

4: Performing k-mean with scaling and without PCA. First determining the number of clusters with a scree plot

```
pokemon %>%
  drop_na() %>%
  select_if(is_numeric) %>%
  scale() %>%
  fviz_nbclust(kmeans, method = "wss")
```



From the plot a total of 2-4 clusters would be suitable because of the relative steep fall between 2-3 and the same fall between 3-4. I choose to work with 4 clusters.

```
pokemon_numeric <- pokemon %>%
  drop_na() %>%
  select_if(is_numeric) %>%
  scale()
```

The general idea of the k-mean algorithm is that it starts by assigning each observation randomly to a cluster and then calculating the clustermean. After it reassign the observation to the cluster

with the closest clustermean and this continues until no observation can be assign to a new cluster i.e. it has converged. Setting nstart=20 so the algorthm will run 20 time and pick the best model within this 20 – the model that minimizes the sum of squares. The downside of the k-mean is that the number of cluster has to be predetermined.

```
km.out <- kmeans(pokemon_numeric, centers = 4, nstart = 20, iter.max = 50)
```

5: Visualizing the data by the 2 first principal components and coloring by clusters.

```
km.out %>%
  fviz_cluster(data = pokemon %>% select_if(is_numeric),
               ggtheme = theme_gray())
```

The number of clusters seems to be reasonable.

6: Inspecting the distribution of the "Type1" over the clusters.

```
table(pokemon$Type1, km.out$cluster)
```

```
##
##              1  2  3  4
##   Bug       21 11 19 18
##   Dark       8 12  8  3
##   Dragon     4 21  5  2
##   Electric  12 14 10  8
##   Fairy      3  6  4  4
##   Fighting   9  8  5  5
##   Fire      19 15 10  8
##   Flying     0  3  1  0
##   Ghost      6 14  9  3
##   Grass     20 21 15 14
##   Ground     8 11  5  8
##   Ice        5 10  4  5
##   Normal    30 21 21 26
##   Poison     9  5  5  9
##   Psychic   14 24 11  8
##   Rock      12 14  9  9
##   Steel      7 12  5  3
##   Water     42 24 17 29
```

```
km.out
```

```
## K-means clustering with 4 clusters of sizes 229, 246, 163, 162
##
## Cluster means:
##       Number      Total  HitPoints     Attack    Defense SpecialAttack
## 1 -0.8679462  0.3866740  0.3264243  0.2124001  0.2554351     0.2082124
## 2  0.7724650  0.9483539  0.5866960  0.7099610  0.5700254     0.7141000
## 3  0.8974477 -0.8321943 -0.5242905 -0.5636849 -0.5066639    -0.6160150
## 4 -0.8490782 -1.1493563 -0.8248088 -0.8111690 -0.7168806    -0.7588816
##   SpecialDefense      Speed Generation
## 1      0.3482488  0.2301258 -0.8888202
## 2      0.6671111  0.5517979  0.7129278
## 3     -0.6122218 -0.5214919  0.9794617
## 4     -0.8892973 -0.6385057 -0.8116831
##
## Clustering vector:
##   [1] 4 1 1 1 4 1 1 1 1 4 1 1 1 4 4 1 4 4 1 1 4 4 1 1 4 1 4 1 4 1 4 1 4 1 4 4 1
##  [38] 4 4 1 4 1 4 1 4 1 4 1 4 1 1 4 1 4 1 4 1 4 1 4 1 4 1 4 4 1 4 1 1 1 4 1
##  [75] 1 4 4 1 4 1 4 4 1 1 1 4 1 1 4 1 4 4 1 4 1 4 1 4 1 4 1 1 1 4 4 1 4 1 4 1 4
## [112] 1 4 1 1 1 1 4 1 4 1 1 1 1 1 4 1 4 1 4 1 1 1 1 1 1 1 1 4 1 1 1 4 4 1 1 1
## [149] 1 4 1 4 1 1 1 1 1 1 1 4 1 1 1 2 2 1 4 1 1 4 1 1 4 1 1 4 1 4 1 4 1 4 1 1 4
## [186] 1 4 4 4 4 1 4 1 4 4 1 1 1 4 1 1 1 4 4 1 4 4 1 4 4 1 1 1 1 1 1 4 1 1 4 1 1
```

```
## [223] 1 1 1 4 1 1 1 1 1 1 1 4 1 4 1 4 1 4 4 1 4 1 1 4 1 1 1 4 1 1 1 4 4 1 4 4
## [260] 4 1 1 1 1 1 4 1 1 2 2 2 1 4 1 1 2 4 1 1 2 4 1 1 2 4 1 4 1 4 4 4 4 1 4 4 1
## [297] 4 4 1 4 1 4 1 4 4 1 2 4 1 4 1 4 1 2 4 1 4 4 4 1 4 1 4 4 4 4 4 1 4 1 4 1 1
## [334] 2 4 1 1 4 1 2 1 1 1 1 1 4 1 4 1 2 1 1 4 1 2 1 4 1 4 4 4 1 4 1 4 1 2 1 1 1
## [371] 1 4 1 4 1 4 1 4 1 4 1 4 2 1 1 4 1 2 4 1 1 1 1 2 4 4 1 2 4 1 2 4 1 1 1 4 4
## [408] 1 2 2 4 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 2 3 3 2 3 3 2 3 3 2
## [445] 3 3 4 3 3 3 2 3 2 3 2 3 2 3 3 3 3 3 3 2 3 3 2 3 2 3 2 2 3 2 3 2 2 2 2 3 2
## [482] 3 3 2 3 2 3 3 3 3 2 3 3 2 2 3 3 2 2 3 2 3 2 3 2 2 3 2 3 3 2 2 2 2 2 2 2 2
## [519] 2 2 2 2 2 2 2 2 2 2 2 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3
## [556] 3 2 3 3 2 3 3 2 3 3 3 3 2 3 2 3 2 3 2 3 2 3 2 3 3 2 3 2 3 3 2 3 3 3 2 2 2
## [593] 3 3 2 3 3 2 2 2 3 3 2 3 3 2 3 2 3 2 2 3 3 2 3 2 2 2 3 2 3 2 2 3 2 3 2 3 2
## [630] 3 2 3 2 3 2 3 3 2 3 3 2 3 2 3 3 2 3 2 3 3 2 3 2 3 2 2 3 2 3 2 3 3 2 3 3 2
## [667] 3 2 3 3 2 3 3 2 3 2 2 3 2 2 3 2 2 3 2 3 2 2 3 2 3 2 2 2 3 3 2 3 2 2 2 2 2
## [704] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 2 3 3 2 3 3 2 3 3 3 3 2 3 3 3 3 2 3 3 2
## [741] 3 2 3 2 2 3 2 2 3 3 2 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 2 2 3 2 3 2 2
## [778] 2 3 2 3 3 3 3 2 2 2 2 3 2 3 2 2 2 2 2 2 2 2 2
##
## Within cluster sum of squares by cluster:
## [1] 1334.6391 1567.1787  466.3004  512.9147
##  (between_SS / total_SS =  46.0 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

Just to sum up, the total amount of different type of pokemons are 163 in cluster 1, 162 in cluster 2, 229 in cluster 3 and 246 in cluster 4. The main part of the pokemons are in cluster 3 and 4, which can also be seen in the table.

Looking at the sum of squares in the different clusters it is clear to see that cluster 3 and 4 have a larger range in the cluster and thereby a larger sum of squares. The main 5 types of pokemons in “Water”, “Psychic”, “Normal”, “Fire” and “Bug”. Inspecting the cluster vector it is clear that it is mainly the first half of the observation that are assigned to cluster 2 and 3 and the last half to cluster 1 and 4.

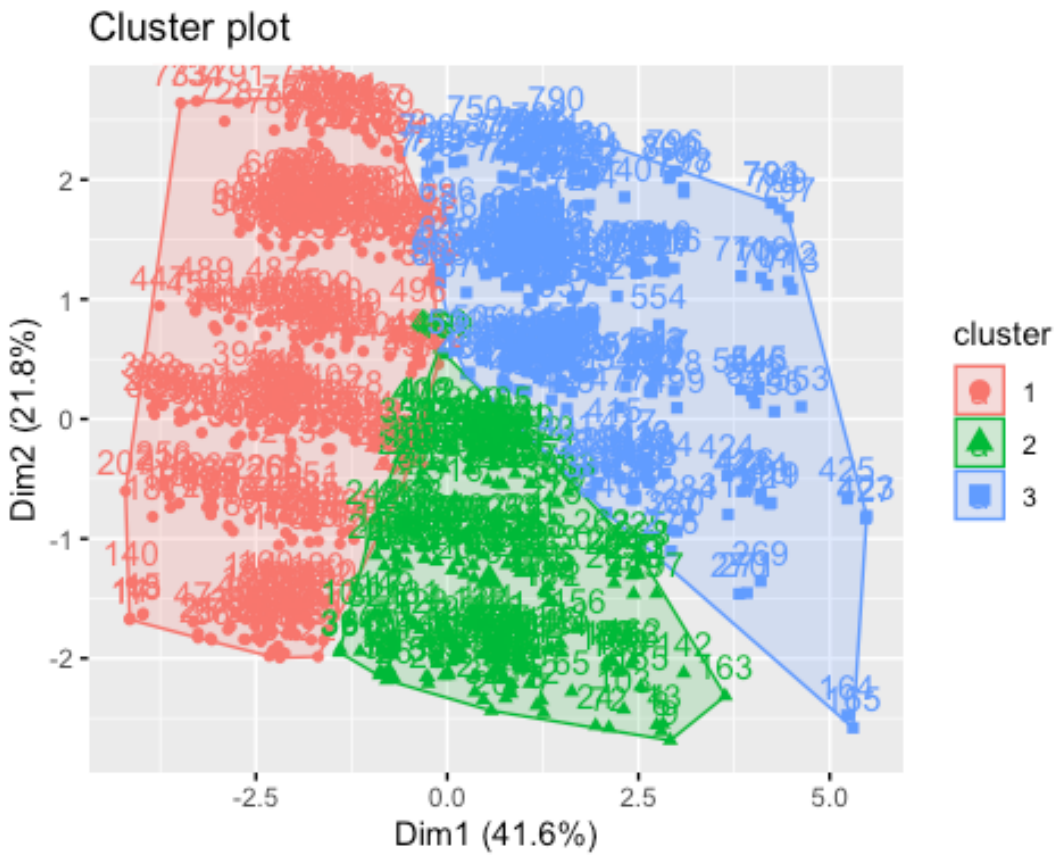
7: Performning a cluster analysis on the scaled and diminsionality reduced data from 2. Since there are no specifications on which cluster algorithm, I choose hierarchical clustering.

The argument “nb.clust=-1” makes sure that the optimal number of cluster is chosen - which is 3 clusters (see plot below).

```
hcpc.out <- res_pca %>%
  HCPC(nb.clust = -1, graph = FALSE)
```

Visualizing the first 2 principal components as earlier:

```
hcpc.out %>%
  fviz_cluster(data = pokemon %>% select_if(is_numeric),
               ggtheme = theme_gray())
```



It is seen that the algorithm has chosen 3 clusters instead of 4 as before the dimensionality reduction. The clusters are comment below.

8: Inspect the distribution of the variable "Type 1" across clusters.

```
table(pokemon$Type1,hcpc.out$data.clust$clust)
```

```
##
##              1  2  3
##    Bug       35 22 12
##    Dark      10  8 13
##    Dragon     7  4 21
##    Electric  17 10 17
##    Fairy      8  3  6
##    Fighting  10  9  8
##    Fire      16 19 17
```

```
##   Flying    1  0  3
##   Ghost    12  6 14
##   Grass    29 19 22
##   Ground   13  8 11
##   Ice       9  5 10
##   Normal   45 30 23
##   Poison   12 11  5
##   Psychic  18 14 25
##   Rock     16 14 14
##   Steel     6  7 14
##   Water    43 44 25
```

```
summary(hcpc.out$data.clust$clust)
```

```
##   1   2   3
## 307 233 260
```

As seen in the summary cluster 1 is now the largest cluster, but also cluster 2 and 3 contains more pokemons. The distribution of the most common types of pokemons as “Water” and “Normal” is more likely to be placed in cluster 1 and 3 while the more special pokemon types as “Ice” and “Flying” are more likely to be placed in 1 and 3. It is clear the distribution of the Type1 will be different since the number of clusters are different. Looking at the plots of the different clusters it seems as if cluster 1 and 2 in the first model was assigned as clsuter 1 in the secound model.