R Notebook

#Before we start

Running libaries

```
library(tidyverse)

library(lubridate)

library(magrittr)

library(FactoMineR)

library(factoextra)

library(uwot)
```

```
pokemon= read_csv("https://sds-aau.github.io/SDS-master/00_data/pokemon.csv")
```

#Tasks

Give a brief overview of data, what variables are there, how are the variables scaled and variation of the data columns.

We overview the data and see that a pokemons 2. type often is NA so we replace that with “No 2. type” because we dont wanna lose these observations when we remove lines with NAs.

```
pokemon %>% head()
```

```
## # A tibble: 6 x 13
##   Number Name           Type1 Type2 Total HitPoints Attack Defense SpecialAttack
##    <dbl> <chr>          <chr> <chr> <dbl>     <dbl>  <dbl>   <dbl>         <dbl>
## 1      1 Bulbasaur      Grass Pois~   318        45     49      49            65
## 2      2 Ivysaur        Grass Pois~   405        60     62      63            80
## 3      3 Venusaur       Grass Pois~   525        80     82      83           100
## 4      3 VenusaurMega ~ Grass Pois~   625        80    100     123           122
## 5      4 Charmander     Fire  <NA>    309        39     52      43            60
## 6      5 Charmeleon     Fire  <NA>    405        58     64      58            80
## # ... with 4 more variables: SpecialDefense <dbl>, Speed <dbl>,
## #   Generation <dbl>, Legendary <lgl>
```

```
pokemon %>% glimpse()
```

```
## Rows: 800
## Columns: 13
## $ Number         <dbl> 1, 2, 3, 3, 4, 5, 6, 6, 6, 7, 8, 9, 9, 10, 11, 12, 13, ~
```

```
## $ Name           <chr> "Bulbasaur", "Ivysaur", "Venusaur", "VenusaurMega Venus~
## $ Type1          <chr> "Grass", "Grass", "Grass", "Grass", "Fire", "Fire", "Fi~
## $ Type2          <chr> "Poison", "Poison", "Poison", "Poison", NA, NA, "Flying~
## $ Total          <dbl> 318, 405, 525, 625, 309, 405, 534, 634, 634, 314, 405, ~
## $ HitPoints      <dbl> 45, 60, 80, 80, 39, 58, 78, 78, 78, 44, 59, 79, 79, 45,~
## $ Attack         <dbl> 49, 62, 82, 100, 52, 64, 84, 130, 104, 48, 63, 83, 103,~
## $ Defense        <dbl> 49, 63, 83, 123, 43, 58, 78, 111, 78, 65, 80, 100, 120,~
## $ SpecialAttack  <dbl> 65, 80, 100, 122, 60, 80, 109, 130, 159, 50, 65, 85, 13~
## $ SpecialDefense <dbl> 65, 80, 100, 120, 50, 65, 85, 85, 115, 64, 80, 105, 115~
## $ Speed          <dbl> 45, 60, 80, 80, 65, 80, 100, 100, 100, 43, 58, 78, 78, ~
## $ Generation     <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1~
## $ Legendary      <lgl> FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE,~
```

```
pokemon$Type2 %<>%
  replace_na("No 2. type")
```

```
pokemon %>% count(Type1, sort = TRUE)
```

```
## # A tibble: 18 x 2
##    Type1        n
##    <chr>    <int>
##  1 Water      112
##  2 Normal      98
##  3 Grass       70
##  4 Bug         69
##  5 Psychic     57
##  6 Fire        52
##  7 Electric    44
##  8 Rock        44
##  9 Dragon      32
## 10 Ghost       32
## 11 Ground      32
## 12 Dark        31
## 13 Poison      28
## 14 Fighting    27
## 15 Steel       27
## 16 Ice         24
## 17 Fairy       17
## 18 Flying       4
```

```
pokemon %>% count(Type2, sort = TRUE)
```

```
## # A tibble: 19 x 2
##    Type2          n
##    <chr>      <int>
##  1 No 2. type   386
##  2 Flying        97
##  3 Ground        35
##  4 Poison        34
##  5 Psychic       33
##  6 Fighting      26
##  7 Grass         25
```

```
##  8 Fairy         23
##  9 Steel         22
## 10 Dark          20
## 11 Dragon        18
## 12 Ghost         14
## 13 Ice           14
## 14 Rock          14
## 15 Water         14
## 16 Fire          12
## 17 Electric       6
## 18 Normal         4
## 19 Bug            3
```

Character strings: We can see the data has 3 character strings "names" which are the names of the pokemons, "type1" which is the main type of the pokemon, and last "type2" which shows some pokemons has a second type, where NA's means they only have one type.

Logical values: We can see the Legendary column is a logical variable, which means FALSE observation shows pokemons whom are not legendary, and true for those who are legendary. This variable can not be scaled.

Numeric values: We can see the first column just counts the pokemons (ID) so we remove this

```
pokemon %<>% select(!Number)

glimpse(pokemon)
```

```
## Rows: 800
## Columns: 12
## $ Name           <chr> "Bulbasaur", "Ivysaur", "Venusaur", "VenusaurMega Venus~
## $ Type1          <chr> "Grass", "Grass", "Grass", "Grass", "Fire", "Fire", "Fi~
## $ Type2          <chr> "Poison", "Poison", "Poison", "Poison", "No 2. type", "~
## $ Total          <dbl> 318, 405, 525, 625, 309, 405, 534, 634, 634, 314, 405, ~
## $ HitPoints      <dbl> 45, 60, 80, 80, 39, 58, 78, 78, 78, 44, 59, 79, 79, 45,~
## $ Attack         <dbl> 49, 62, 82, 100, 52, 64, 84, 130, 104, 48, 63, 83, 103,~
## $ Defense        <dbl> 49, 63, 83, 123, 43, 58, 78, 111, 78, 65, 80, 100, 120,~
## $ SpecialAttack  <dbl> 65, 80, 100, 122, 60, 80, 109, 130, 159, 50, 65, 85, 13~
## $ SpecialDefense <dbl> 65, 80, 100, 120, 50, 65, 85, 85, 115, 64, 80, 105, 115~
## $ Speed          <dbl> 45, 60, 80, 80, 65, 80, 100, 100, 100, 43, 58, 78, 78, ~
## $ Generation     <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1~
## $ Legendary      <lgl> FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE,~
```

We can see Generation is numeric and show which generation of pokemon game each pokemon is from.

```
pokemon %>%
  count(Generation)
```

```
## # A tibble: 6 x 2
##   Generation     n
##        <dbl> <int>
## 1          1   166
## 2          2   106
## 3          3   160
```

```
## 4           4   121
## 5           5   165
## 6           6    82
```

When we look at the other numerical values we have calculated the standard deviation and the mean, to see if the data is of the same scaling. So we drop NAs select all the numerical variable except Generation and calculate the standard deviationa and mean.

```
pokemon %<>%
  drop_na()

pokemon_sd= pokemon %>%
  select(is.numeric) %>%
  select(!Generation)

s_deviation=apply(pokemon_sd, 2, sd)

mean=colMeans(pokemon_sd)

pokemon_stats= as.data.frame(s_deviation, row.names = c("sd"))%>%
  cbind(as.data.frame(mean, row.names = "mean"))%>%
  print()
```

```
##                s_deviation      mean
## Total            119.96304 435.10250
## HitPoints         25.53467  69.25875
## Attack            32.45737  79.00125
## Defense           31.18350  73.84250
## SpecialAttack     32.72229  72.82000
## SpecialDefense    27.82892  71.90250
## Speed             29.06047  68.27750
```

The above shows that Total has a much larger standard deviation and mean. Else the other variables are almost the same, but because of the Total variable we should scale the data.

We can visually show the variables correlation with each other and if there is any differences between legendary and none legendary pokemons.

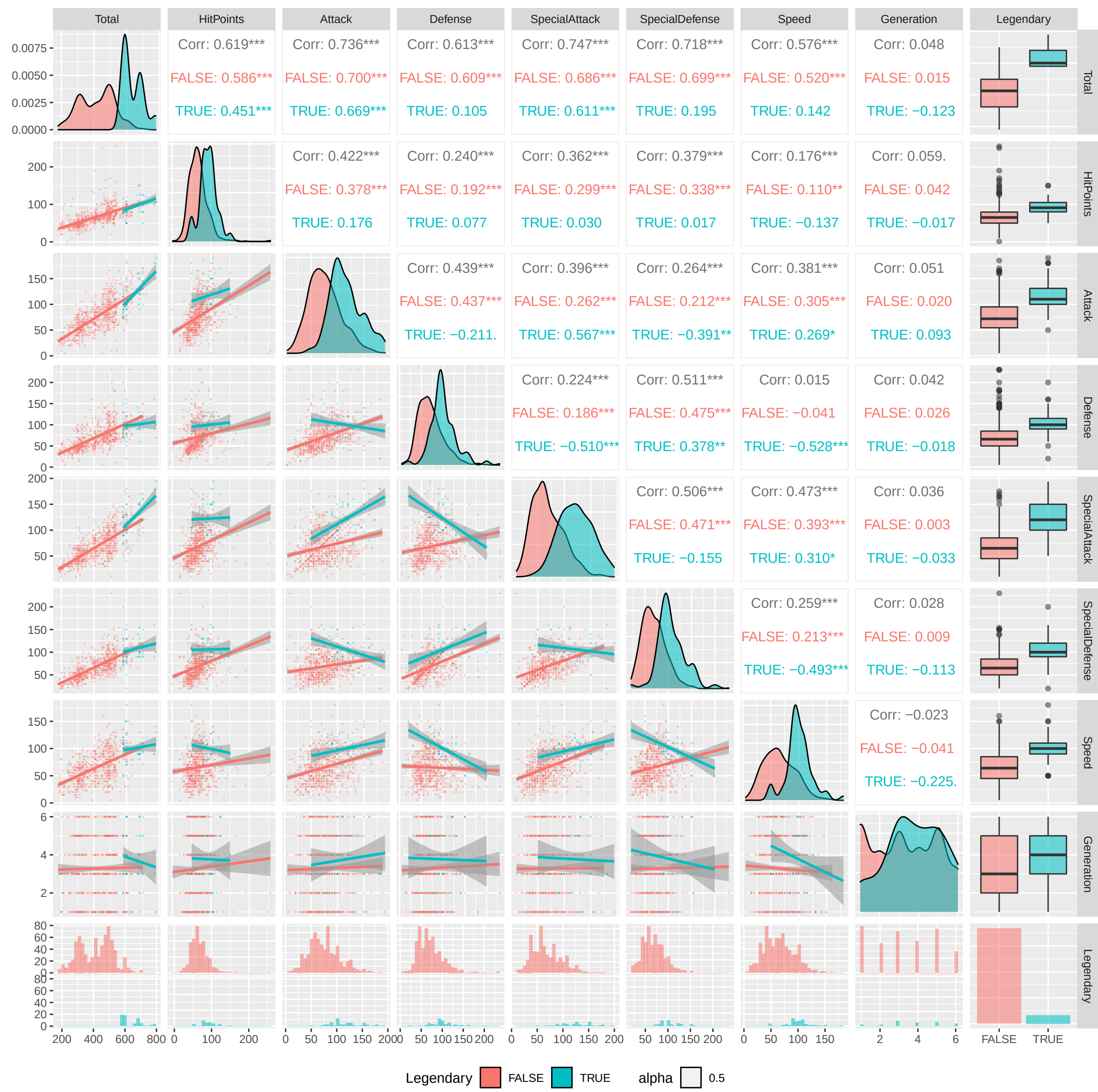
```
library(GGally)
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```
pokemon %>%
  select(-Name, -Type1, -Type2) %>%
  ggpairs(legend = 1,
          mapping = ggplot2::aes(colour=Legendary, alpha = 0.5),
          lower = list(continuous = wrap("smooth", alpha = 0.3, size=0.1))) +
  theme(legend.position = "bottom")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Generation looks uncorrelated with the remaining variables and Legendaries seems stronger in pretty much every stat category.

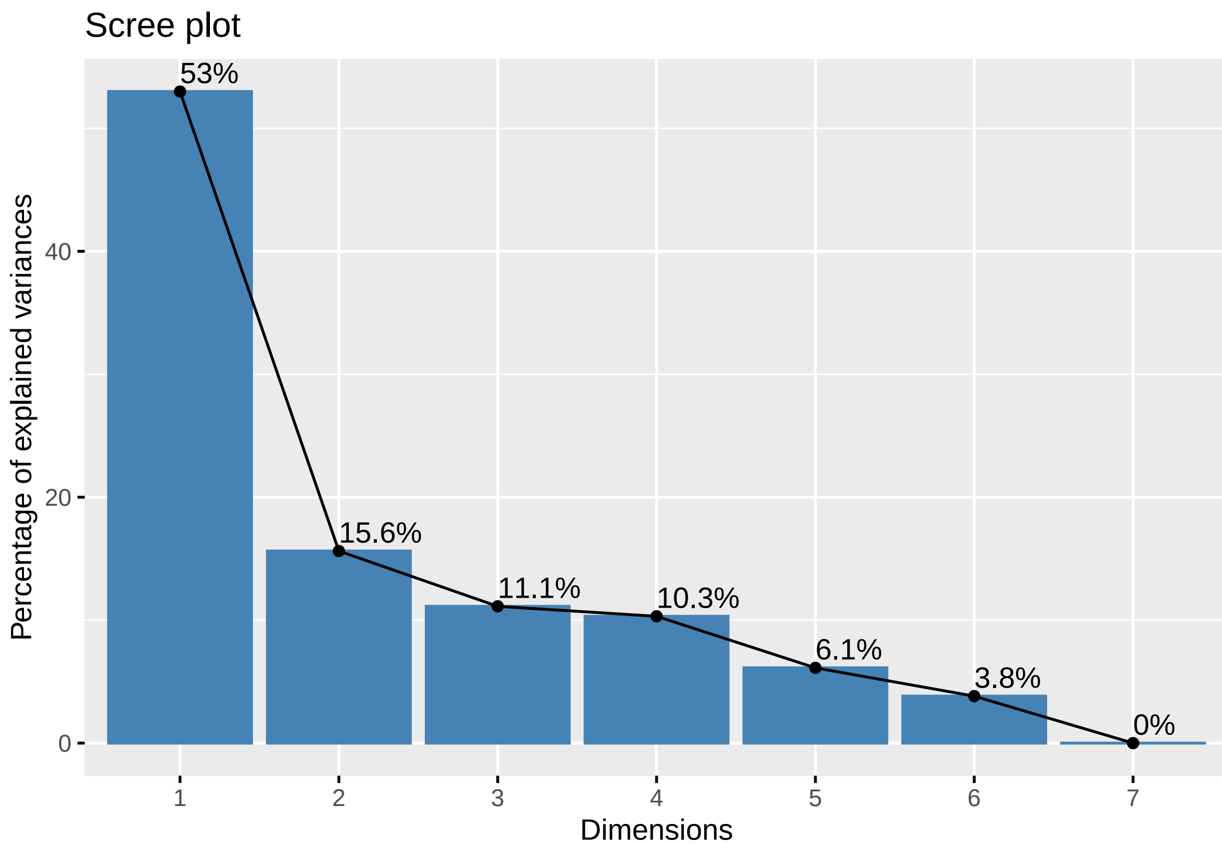
Execute a PCA analysis on all numerical variables in the dataset. Hint: Don't forget to scale them first. Use 4 components. What is the cumulative explained variance ratio? Hint: I am not sure this terminology and code was introduced during class, but try and look into

cumulative explained variance and sklearn(package) and see if you can figure out the code needed.

We run a PCA on the pokemon dataset and scale it. Then we make a scree plot to pick number of dimensions to use.

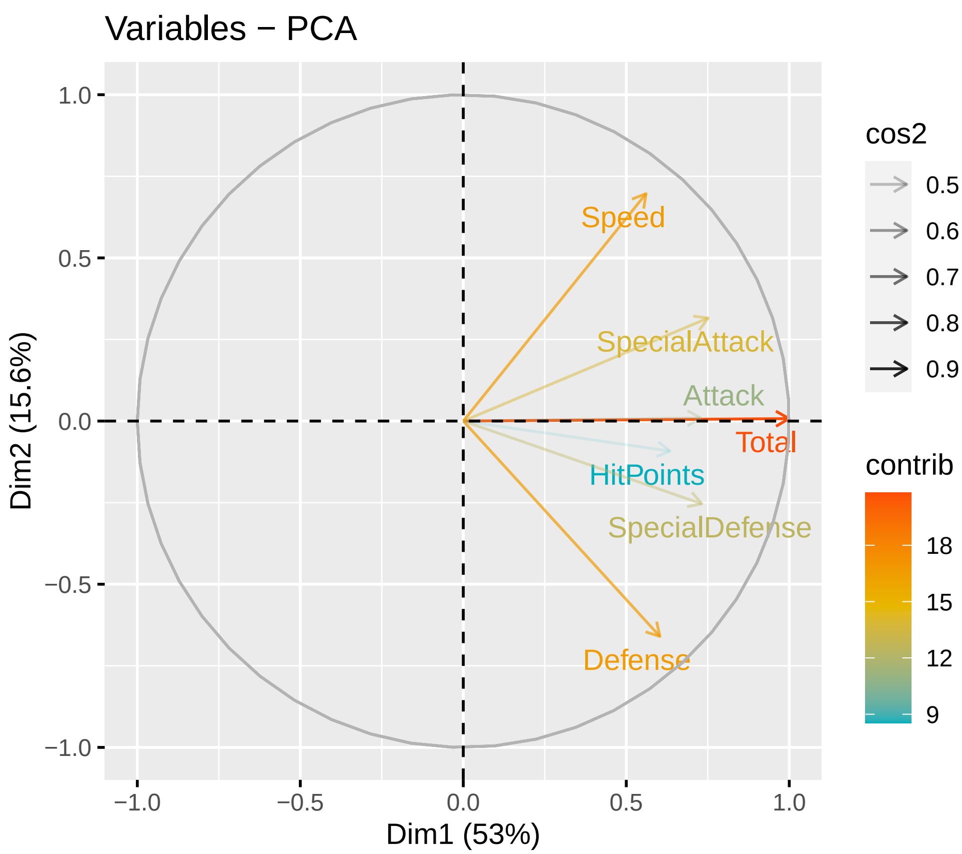
```
res_pca <- pokemon %>%
  select(!Generation)%>%
  select_if(is_numeric) %>%
  PCA(scale.unit = TRUE, graph =FALSE)
```

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

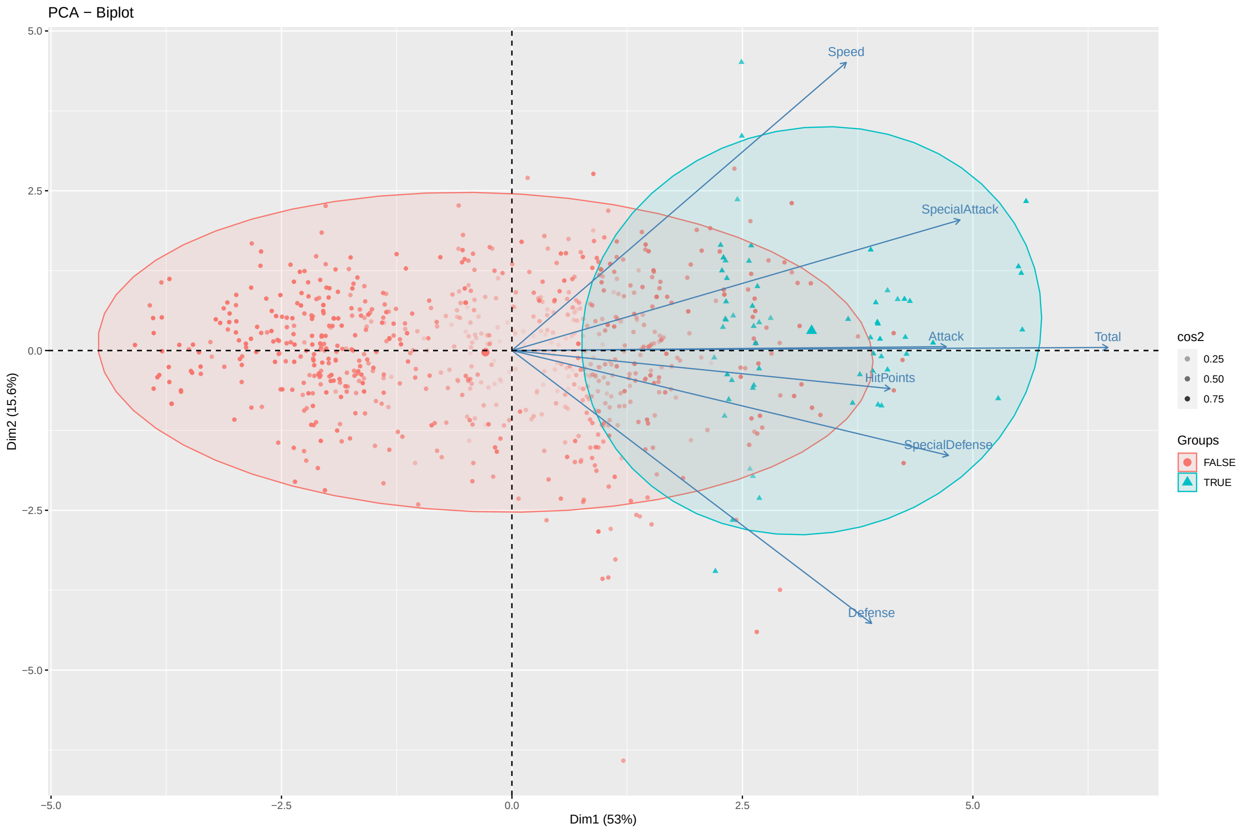


We can see the elbow shows the optimal dimensions are two dimensions. Then we visualize our reduced data in two dimensions.

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




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



From the two plots we can see that some of the variables are correlated eg. Attack and Total seems to be very much correlated the distance between the lines shows the correlation, We can see that speed and Defense has a very low correlation as the angle is almost 90 degrees. We can also see there are no negative correlation between the variables, as there are no angels above 90 degrees. The x-axis seems to divide the attributes into offensive above the axis and defensive below the axis. By separating the data into groups according to their legendary status is seems like the legendary pokemons are located more to the right of the plot which indicates higher levels of attributes.

To answer the latter of the question we simply extract the cumulative variance calculated in the pca analysis.

```
res_pca$eig[,3][1:4]
```

```
##    comp 1    comp 2    comp 3    comp 4
## 52.99246 68.61547 79.74109 90.05686
```

We can see that the cumulative variance at component 4 is 90.05% This means the 4 dimensions explain 90.05% of the variance in the data.

Use a different dimensionality reduction method (eg. UMAP/NMF) – do the findings differ?

We use UMAP which is a different dimensionality reduction method. UMAP is more optimal when dealing with two dimensions.

```
res_umap <- pokemon %>%
  select(!Generation)%>%
  select_if(is_numeric) %>%
  umap(n_neighbors = 15,
```

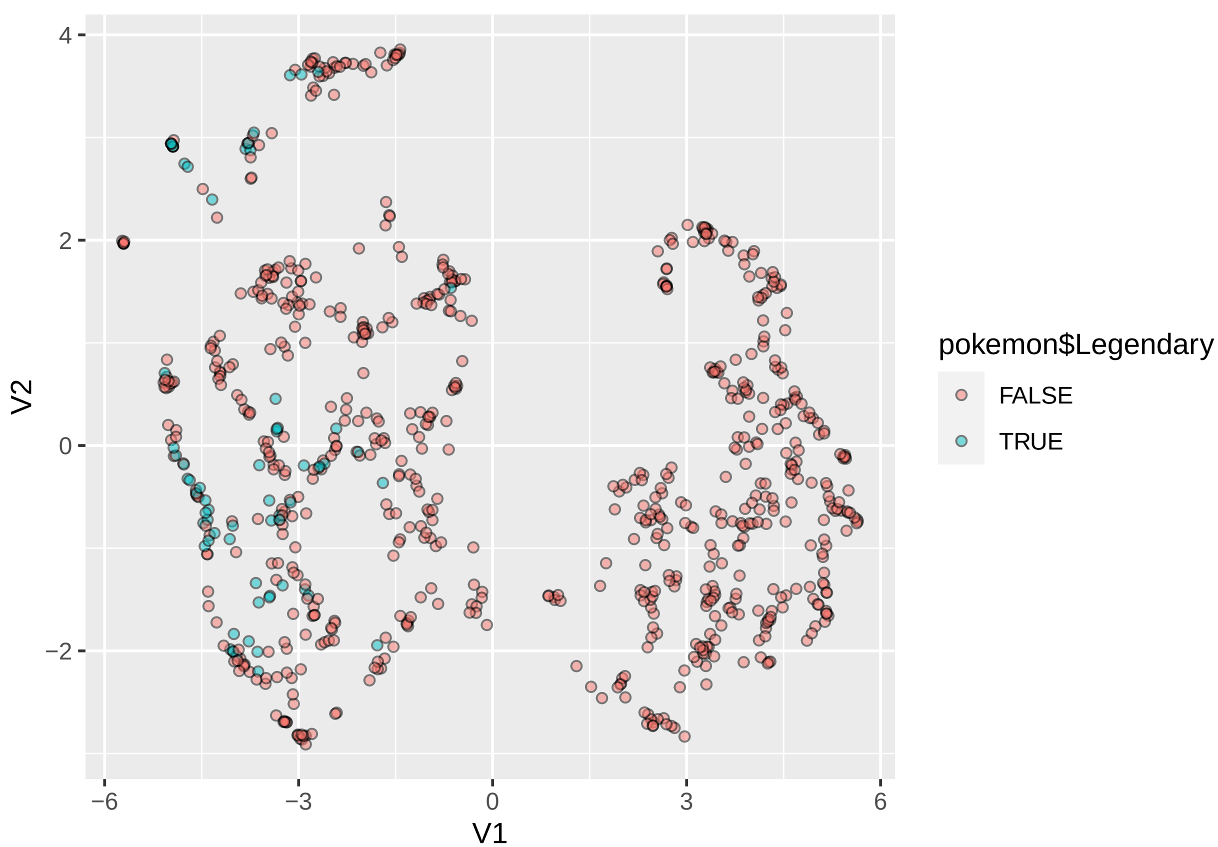
```r
       metric = "cosine",
       min_dist = 0.01,
       scale = TRUE)
```

```r
res_umap %>% as_tibble() %>%
  glimpse()
```

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

```
## Rows: 800
## Columns: 2
## $ V1 <dbl> 5.1593713, 5.1571059, -4.7837701, -4.5692702, 3.3412218, 2.4063252,~
## $ V2 <dbl> -0.97723883, -1.61101097, -0.17420584, -0.49225313, -1.96048445, -2~
```

```r
res_umap %>%
  as_tibble() %>%
  ggplot(aes(x = V1, y = V2, fill = pokemon$Legendary)) +
  geom_point(shape = 21, alpha = 0.5)
```

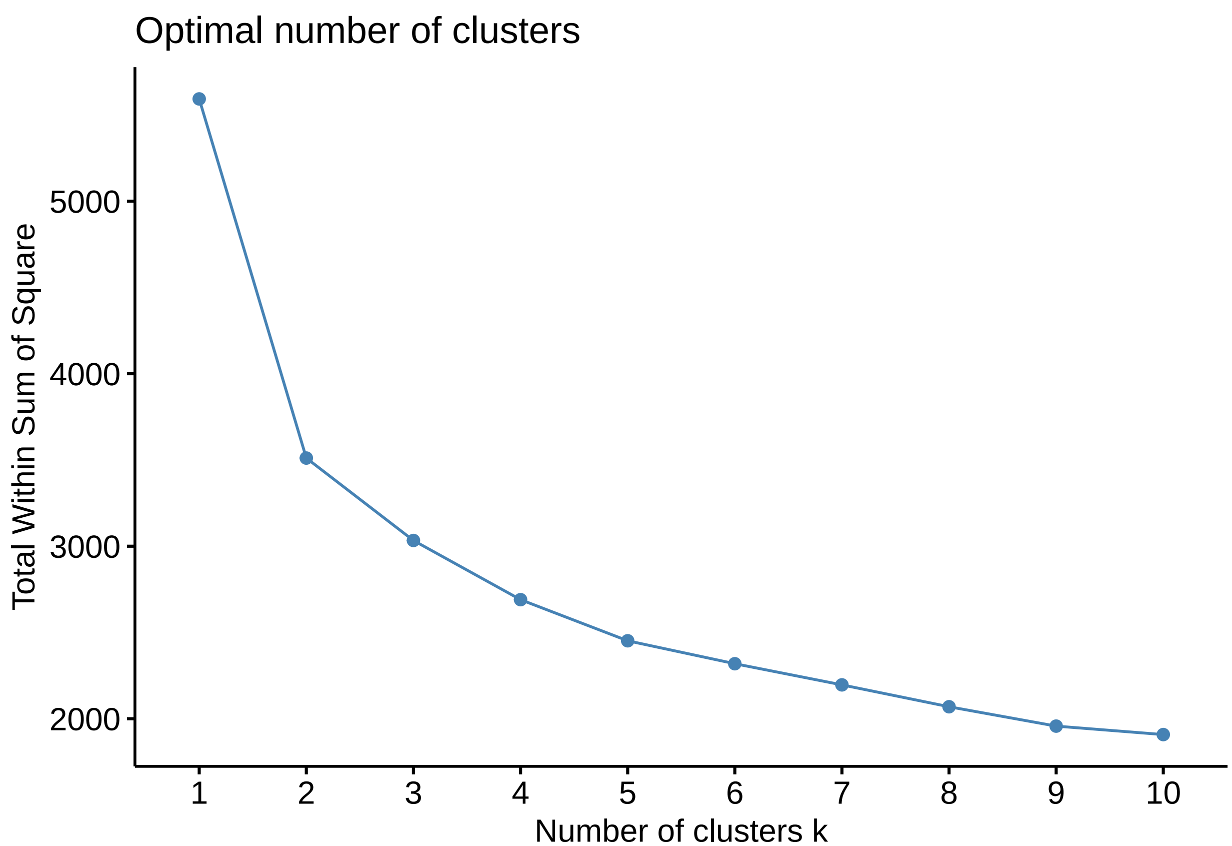


It looks like UMAP are separating the legendary pokemon from the normal pokemon a bit better then the PCA analysis by clustering them closer together.

Perform a cluster analysis (KMeans) on all numerical variables (scaled & before PCA). Pick a realistic number of clusters (up to you where the large clusters remain mostly stable).

We start by selecting all our numerical variables except Generations and run a Kmean cluster analysis.

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



We can see from the plot that it looks like we should use two clusters, because this is where we see the elbow.

```
res_km <- pokemon %>%
  select(!Generation) %>%
  select_if(is_numeric) %>%
  scale() %>%
  kmeans(centers = 2, nstart = 20)

pokemon_nr = pokemon %>%
  select(!Generation)%>%
  select_if(is_numeric)
```

```
res_km
```

```
## K-means clustering with 2 clusters of sizes 422, 378
```

```
##
## Cluster means:
##        Total  HitPoints     Attack    Defense SpecialAttack SpecialDefense
## 1  0.7917080  0.5266335  0.5709621  0.4978530     0.5526878      0.5911326
## 2 -0.8838645 -0.5879347 -0.6374233 -0.5558041    -0.6170218     -0.6599417
##        Speed
## 1  0.4451328
## 2 -0.4969472
##
## Clustering vector:
##   [1] 2 2 1 1 2 2 1 1 1 2 2 1 1 2 2 2 2 2 2 1 2 2 1 1 2 2 2 1 2 1 2 1 2 1 2 2 1
##  [38] 2 2 1 2 1 2 1 2 1 2 1 2 2 1 2 2 2 1 2 2 2 1 2 1 2 1 2 1 2 2 1 2 2 1 1 2 2
##  [75] 1 2 2 1 2 1 2 2 1 2 1 2 1 1 2 1 2 2 1 2 1 2 1 2 1 2 2 1 1 2 2 1 2 1 2 1 2
## [112] 1 2 2 1 1 2 2 1 2 1 1 2 1 1 2 1 2 1 2 1 1 1 1 1 1 1 1 1 2 1 1 1 2 2 1 1 1
## [149] 2 2 1 2 1 1 1 1 1 1 1 1 2 2 1 1 1 1 1 2 2 1 2 2 1 2 2 1 2 2 2 1 2 2 2 2 1 2
## [186] 1 2 2 2 2 2 2 1 2 2 1 1 1 2 2 2 1 2 2 1 2 2 1 2 2 1 1 1 2 1 1 2 2 1 2 1 2
## [223] 2 1 1 2 1 2 1 1 1 1 1 2 2 1 2 2 2 1 2 2 1 2 1 1 2 1 1 1 2 1 1 1 2 2 1 2 2
## [260] 2 1 1 1 1 1 2 2 1 1 1 1 1 2 2 1 1 2 2 1 1 2 2 1 1 2 2 2 2 2 2 2 2 2 2 2 1
## [297] 2 2 1 2 2 2 2 2 2 1 1 2 2 2 1 2 1 1 2 1 2 2 2 1 2 1 2 2 2 2 2 1 2 1 2 2 1
## [334] 1 2 2 1 2 1 1 2 2 2 2 2 2 1 2 1 1 2 1 2 1 1 1 2 1 2 2 2 1 2 1 2 1 1 1 1 1
## [371] 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 1 2 1 1 2 1 1 2 2 1 1 2 2 1 2 1 1 1 2 2
## [408] 2 1 1 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 1 2 2 1 2 2 1 2 2 1
## [445] 2 2 2 2 2 2 1 2 1 2 1 2 1 2 1 2 2 2 2 1 2 2 1 2 1 2 1 1 2 1 2 1 1 1 1 2 1
## [482] 2 2 1 2 1 2 2 2 2 1 2 2 1 1 2 2 1 1 2 1 2 1 2 1 1 2 1 2 2 1 1 1 1 1 1 1 1
## [519] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2
## [556] 2 1 2 2 1 2 2 1 2 2 2 2 1 2 1 2 1 2 1 2 1 2 1 2 2 1 2 1 2 2 1 2 2 2 1 1 1
## [593] 2 2 1 2 2 1 1 1 2 2 1 2 2 1 2 1 2 1 1 2 2 1 2 1 1 1 2 1 2 1 1 2 1 2 1 2 1
## [630] 2 1 2 1 2 1 2 2 1 2 2 1 2 1 2 2 1 2 1 2 2 1 2 1 2 1 1 2 1 2 1 2 1 1 2 2 1
## [667] 2 1 2 2 1 2 2 1 2 1 1 2 1 1 2 1 1 2 1 2 1 1 2 1 2 1 1 1 2 2 1 2 1 1 1 1 1
## [704] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 1 2 2 1 2 2 1 2 2 2 2 1 2 2 2 2 1 2 2 1
## [741] 2 1 2 1 1 2 1 1 2 1 1 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 1 1 2 1 2 1 1
## [778] 1 2 1 2 2 2 2 1 1 1 1 2 1 2 1 1 1 1 1 1 1 1 1
##
## Within cluster sum of squares by cluster:
## [1] 2412.234 1098.859
##  (between_SS / total_SS =  37.2 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

```r
res_km %>%
  fviz_cluster(data = pokemon_nr %>% select_if(is_numeric) ,
               ggtheme = theme_gray())
```

```
## Warning: Deprecated
```

```
## Warning: Deprecated
```

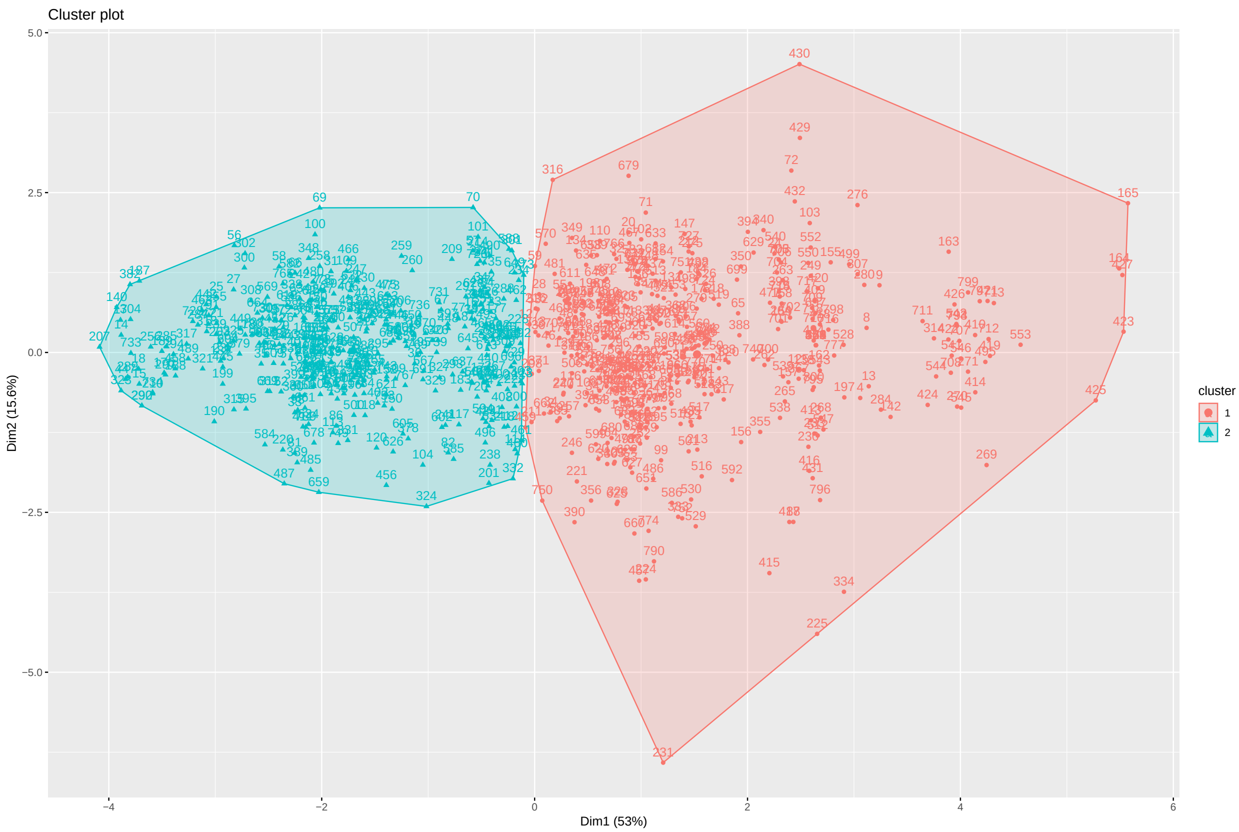
```
## Warning: Deprecated
```

```
## Warning: Deprecated
```

```
## Warning: Deprecated

## Warning: Deprecated

## Warning: Deprecated
```



After using KMEANS on the scaled and not dimentionnaly reduced data, we get two clusters.

Visualize the first 2 principal components and color the datapoints by cluster.

```
pokemon_nr = pokemon %>%
  select(!Generation) %>%
  select(is.numeric)
```

```
pokemon_nr[,"pca1"] <- res_pca$ind$coord[,1]
pokemon_nr[,"pca2"] <- res_pca$ind$coord[,2]

glimpse(pokemon_nr)
```

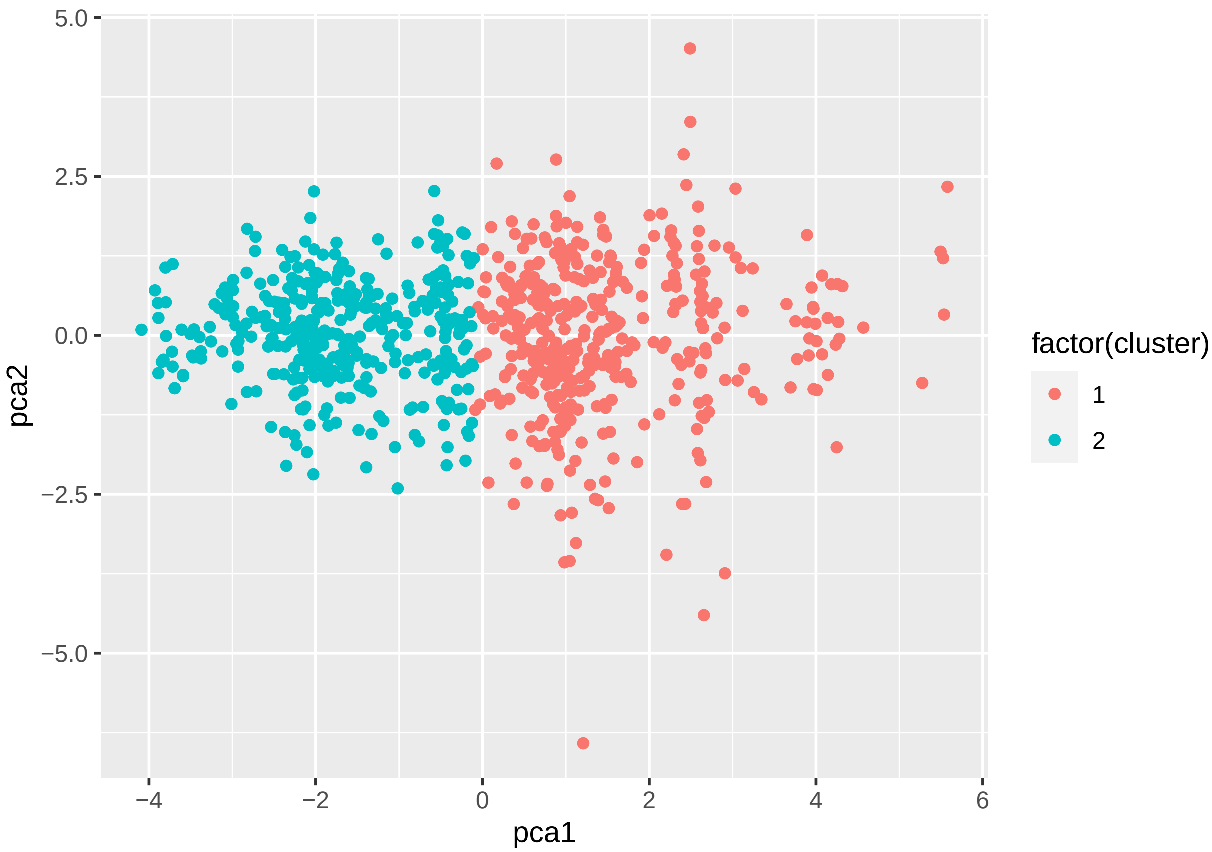
```
## Rows: 800
## Columns: 9
## $ Total          <dbl> 318, 405, 525, 625, 309, 405, 534, 634, 634, 314, 405, ~
## $ HitPoints      <dbl> 45, 60, 80, 80, 39, 58, 78, 78, 78, 44, 59, 79, 79, 45,~
## $ Attack         <dbl> 49, 62, 82, 100, 52, 64, 84, 130, 104, 48, 63, 83, 103,~
## $ Defense        <dbl> 49, 63, 83, 123, 43, 58, 78, 111, 78, 65, 80, 100, 120,~
## $ SpecialAttack  <dbl> 65, 80, 100, 122, 60, 80, 109, 130, 159, 50, 65, 85, 13~
```

```
## $ SpecialDefense <dbl> 65, 80, 100, 120, 50, 65, 85, 85, 115, 64, 80, 105, 115~
## $ Speed          <dbl> 45, 60, 80, 80, 65, 80, 100, 100, 100, 43, 58, 78, 78, ~
## $ pca1           <dbl> -1.8400876, -0.4435426, 1.4803086, 3.0605575, -2.045946~
## $ pca2           <dbl> 0.02549350, 0.05076025, 0.05842799, -0.71193930, 0.7114~
```

```
pokemon_cluster=pokemon
pokemon_cluster_nr=pokemon_nr

pokemon_cluster[,"cluster"] <- res_km$cluster
pokemon_cluster_nr[,"cluster"] <- res_km$cluster

pokemon_cluster_nr %>%
  ggplot(aes(x=pca1, y=pca2, color=factor(cluster)))+
  geom_point()
```



Inspect the distribution of the variable Type1 across clusters. Does the algorithm separate the different types of pokemon?

```
type1_table=table(pokemon_cluster$cluster, pokemon_cluster$Type1)

type1_table
```

```
##
##     Bug Dark Dragon Electric Fairy Fighting Fire Flying Ghost Grass Ground Ice
##   1  24   17     23       25     8       15   29      3    19    35     16  14
```

```
##   2  45   14      9       19      9       12    23      1     13     35      16   10
##
##     Normal Poison Psychic Rock Steel Water
##   1     43     14      35   24    19    59
##   2     55     14      22   20     8    53
```

We can see that the algorithm does not fully separate the different types of pokemons into the 2 clusters. As many of the types are equaly split between the two clusters.

```
pokemon_cluster_nr %>%
  select_if(is_numeric) %>%
  group_by(cluster) %>%
  mutate(n = n()) %>%
  summarise_all(funs(mean)) %>%
  pivot_longer(-cluster) %>%
  pivot_wider(names_from = cluster, values_from = value)
```

```
## Warning: Deprecated

## Warning: Deprecated

## Warning: Deprecated

## Warning: Deprecated

## Warning: Deprecated

## Warning: Deprecated

## Warning: Deprecated

## Warning: Deprecated

## Warning: Deprecated

## Warning: Deprecated

## Warning: `funs()` was deprecated in dplyr 0.8.0.
## Please use a list of either functions or lambdas:
##
##   # Simple named list:
##   list(mean = mean, median = median)
##
##   # Auto named with `tibble::lst()`:
##   tibble::lst(mean, median)
##
##   # Using lambdas
##   list(~ mean(., trim = .2), ~ median(., na.rm = TRUE))
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_warnings()` to see where this warning was generated.

## # A tibble: 10 x 3
##    name                `1`      `2`
```

```
##     <chr>              <dbl>    <dbl>
##   1 Total            530.     329.
##   2 HitPoints         82.7     54.2
##   3 Attack            97.5     58.3
##   4 Defense           89.4     56.5
##   5 SpecialAttack     90.9     52.6
##   6 SpecialDefense    88.4     53.5
##   7 Speed             81.2     53.8
##   8 pca1               1.53    -1.70
##   9 pca2              -0.0298   0.0332
## 10 n                 422      378
```

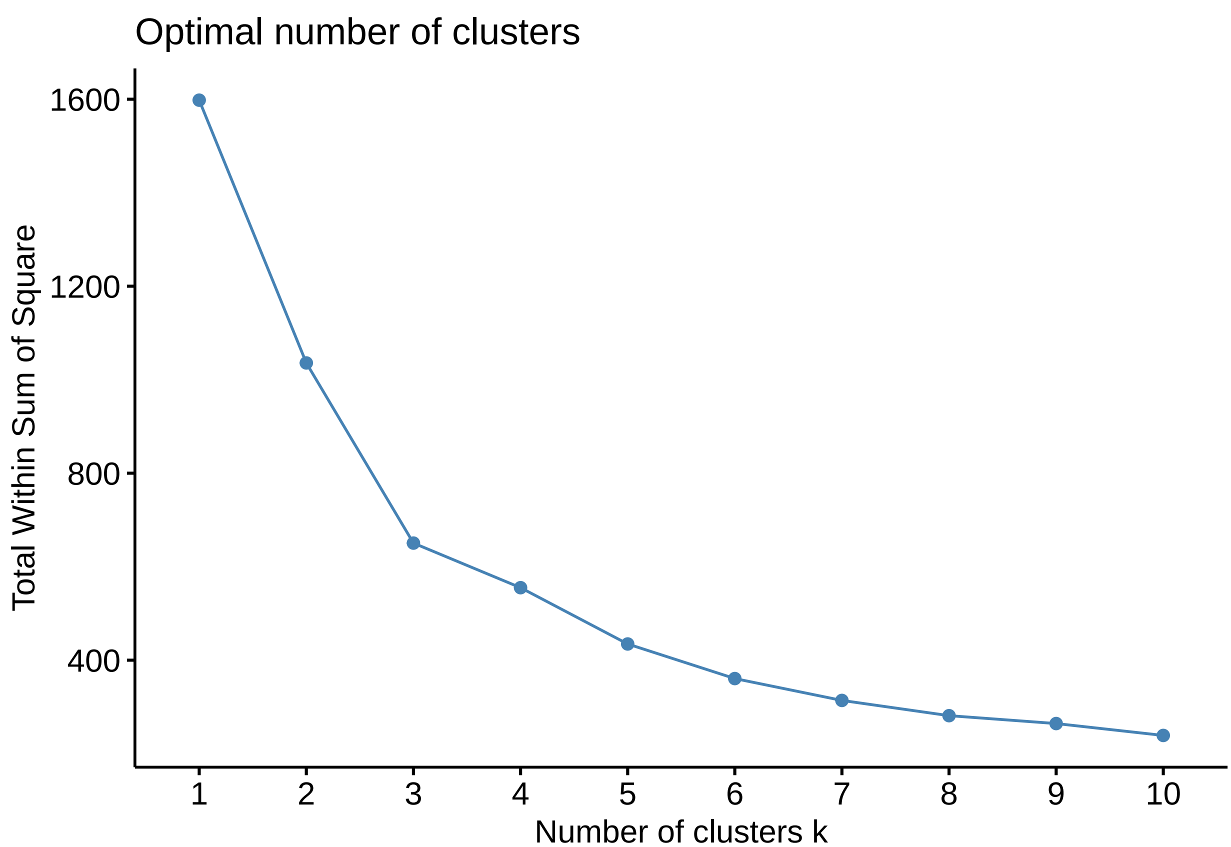
We can see the two clusters are mainly seperatet by overall attribute and not the type of the pokemon. (by looking at the mean of each attribute in the 2 clusters)

Perform a cluster analysis on all numerical variables scaled and AFTER dimensionality reduction and visualize the first 2 principal components.

We do the same steps as above now only using the two columns showing the 2 dimensions from the PCA analysis.

```
pokemon_pca= pokemon_nr%>%
  select(pca1, pca2)

pokemon_pca %>%
  scale() %>%
  fviz_nbclust(kmeans, method = "wss")
```



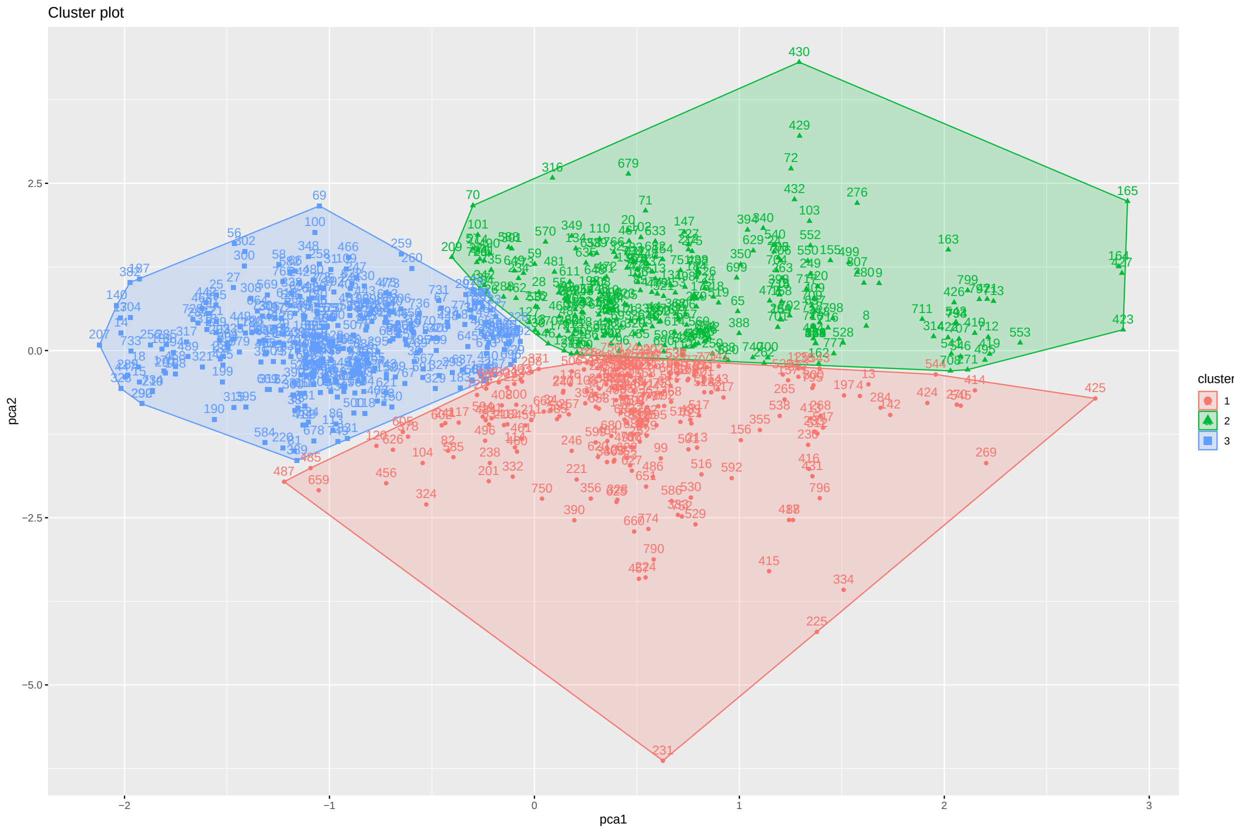
We can now see the elbow is formed at 3 clusters.

```r
res_km_pca <- pokemon_pca %>%
  scale() %>%
  kmeans(centers = 3, nstart = 20)
```

```r
res_km_pca
```

```
## K-means clustering with 3 clusters of sizes 224, 257, 319
##
## Cluster means:
##         pca1       pca2
## 1  0.5171839 -1.0593594
## 2  0.7794596  0.7979146
## 3 -0.9911295  0.1010422
##
## Clustering vector:
##   [1] 3 3 2 1 3 3 2 2 2 3 3 1 1 3 3 3 3 3 2 3 3 2 2 3 2 3 2 3 2 3 2 3 1 3 3 1
##  [38] 3 3 2 3 1 3 2 3 2 3 2 3 3 1 3 1 3 2 3 2 3 2 3 2 3 2 3 2 3 3 1 3 2 2 2 3 1
##  [75] 1 3 3 2 3 2 3 1 1 2 2 3 1 1 3 2 3 3 2 3 1 3 1 3 1 3 2 2 2 1 3 1 3 1 3 2 3
## [112] 2 3 1 2 1 1 3 1 1 1 1 1 1 1 3 2 3 2 3 2 2 2 2 2 2 1 1 2 3 1 1 1 3 3 2 2 2
## [149] 3 3 1 3 1 2 2 1 1 2 2 3 3 2 2 2 2 2 3 3 1 3 3 2 3 3 1 3 3 3 2 3 3 3 3 2 3
## [186] 2 3 3 3 3 1 3 2 3 3 1 1 1 3 1 1 1 3 3 2 3 3 1 2 3 1 2 1 2 1 2 3 1 2 3 1 1
## [223] 1 1 1 3 1 3 1 1 1 1 1 2 3 1 3 1 3 1 1 3 1 3 1 1 3 2 2 2 3 1 1 2 3 3 1 3 3
## [260] 3 1 2 2 2 1 3 3 1 1 1 2 2 3 2 2 2 3 3 2 2 3 3 1 1 3 3 3 2 3 3 3 3 3 3 3 2
## [297] 3 3 2 3 2 3 1 3 3 2 2 3 3 3 2 3 2 2 3 2 3 3 3 2 3 1 3 1 3 3 3 1 3 1 3 1 1
## [334] 1 3 3 2 3 2 2 2 2 3 3 3 3 1 3 2 2 3 2 3 1 1 1 3 2 3 3 3 2 3 2 3 1 1 2 2 2
## [371] 1 3 1 3 1 3 1 1 1 3 1 3 2 3 1 3 2 2 3 1 1 3 2 2 3 3 2 2 3 1 1 3 1 1 1 3 3
## [408] 1 2 2 3 1 1 1 1 1 1 2 2 2 2 2 2 1 1 2 2 2 2 2 1 2 3 1 1 3 2 2 3 3 1 3 3 2
## [445] 3 3 3 3 3 3 2 3 2 3 2 1 1 3 1 1 1 2 3 1 3 3 2 3 2 3 1 2 3 2 3 2 2 2 2 3 2
## [482] 3 3 2 1 1 1 3 3 2 1 3 3 2 2 1 3 2 2 3 1 3 1 3 2 1 3 2 3 3 1 1 2 1 1 1 1 2
## [519] 2 2 2 1 1 1 2 2 2 2 1 1 2 2 1 1 1 1 1 1 1 2 2 2 1 1 1 2 1 2 2 2 2 2 2 2 3
## [556] 3 2 3 3 2 3 3 2 3 3 3 3 1 3 2 3 2 3 2 3 2 3 1 3 3 2 3 2 3 1 1 3 2 3 2 1 1
## [593] 3 1 1 3 3 2 1 2 3 1 2 3 1 2 3 2 3 2 2 3 3 2 3 2 1 2 3 1 3 1 2 3 1 1 1 3 2
## [630] 3 1 3 2 3 2 3 3 1 3 3 1 3 2 3 3 2 3 2 2 3 1 3 1 3 1 1 3 2 1 1 3 1 1 3 1 1
## [667] 3 1 3 3 2 3 3 2 3 1 2 3 2 1 3 2 1 3 1 3 1 1 3 2 3 1 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 1 1 3 3 2 3 2 2 3 3 3 3 2 3 3 2 3 2 3 3 2
## [741] 3 2 3 1 2 3 2 2 3 1 2 1 3 1 3 1 3 1 3 1 3 1 3 1 3 2 3 1 3 1 1 2 2 1 3 2 2
## [778] 1 3 1 3 3 3 3 1 1 1 1 3 1 3 2 2 2 1 1 2 2 2 1
##
## Within cluster sum of squares by cluster:
## [1] 233.6856 228.6202 188.0082
##  (between_SS / total_SS =  59.3 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

```r
res_km_pca %>%
  fviz_cluster(data = pokemon_pca,
               ggtheme = theme_gray())
```

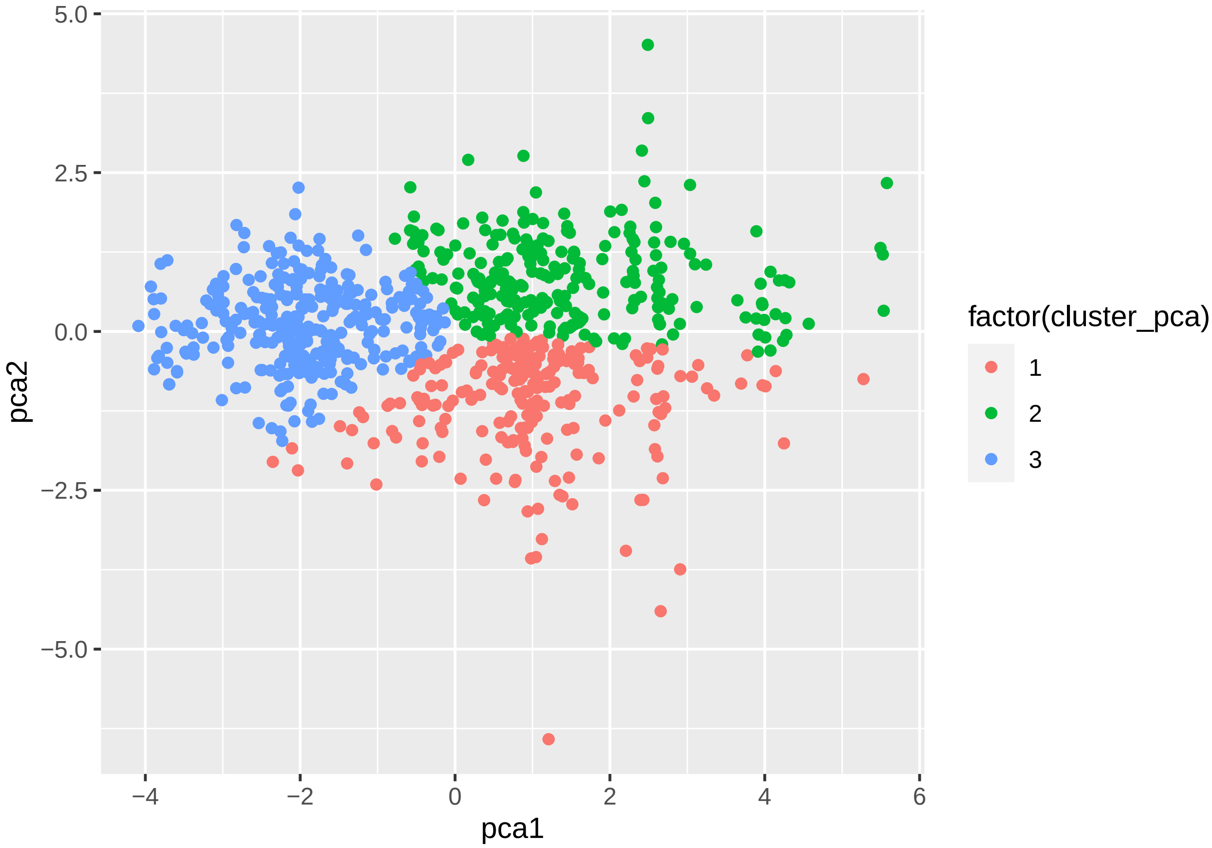


We can see it has split the observations into 3 clusters very nicely

```
pokemon_cluster_pca= pokemon
pokemon_cluster_nr_pca= pokemon_nr

pokemon_cluster_pca[,"cluster_pca"] <- res_km_pca$cluster
pokemon_cluster_nr_pca[,"cluster_pca"] <- res_km_pca$cluster

pokemon_cluster_nr_pca %>%
  ggplot(aes(x=pca1, y=pca2, color=factor(cluster_pca)))+
  geom_point()
```



Again, inspect the distribution of the variable "Type 1" across clusters, does it differ from the distribution before dimensionality reduction?

```
table(pokemon_cluster_pca$cluster_pca, pokemon_cluster_pca$Type1)
```

```
##
##     Bug Dark Dragon Electric Fairy Fighting Fire Flying Ghost Grass Ground Ice
##   1  17    6      5       10     6        9    7      0    10    19     14   8
##   2  15   13     19       20     3        8   25      3    10    21      6   7
##   3  37   12      8       14     8       10   20      1    12    30     12   9
##
##     Normal Poison Psychic Rock Steel Water
##   1     17      7      11   25    18    35
##   2     33      7      27    7     4    29
##   3     48     14      19   12     5    48
```

```
type1_table
```

```
##
##     Bug Dark Dragon Electric Fairy Fighting Fire Flying Ghost Grass Ground Ice
##   1  24   17     23       25     8       15   29      3    19    35     16  14
##   2  45   14      9       19     9       12   23      1    13    35     16  10
##
##     Normal Poison Psychic Rock Steel Water
##   1     43     14      35   24    19    59
##   2     55     14      22   20     8    53
```

It seems like the clusters are categorized in relation to attributes or abilities, and not so much the type of the pokemon. As was also the conclusion using the not dimensionalised data.