Are Pokemon Born Equal? Analisys Using Multidimensional Scaling and Clustering

Michał Szałański

10 marca 2019

Introduction

Goal of the paper

Data preparation

Libraries

```
library(dplyr)
library(tidyr)
library(ggplot2)
library(gridExtra)
library(corrplot)
library(labdsv)
library(smacof)
library(psych)
library(pca3d)
library(NbClust)
library(ClusterR)
library(wesanderson)
library(factoextra)
library(clustertend)
library(knitr)
options(scipen=999)

# Define nice colors
cYellow = '#FADA5E'
cBlue = '#378CC7'
```

Loading the data from a csv.

The source is a fantastic Kaggle dataset containing the statistick for every pokemon released.

```
pokemon <- read.csv('data/Pokemon.csv')
pokemon <- rename(pokemon, 'Special.Attack' = 'Sp..Atk', 'Special.Defense' = 'Sp..Def')
```

Data exploration

First look at the data

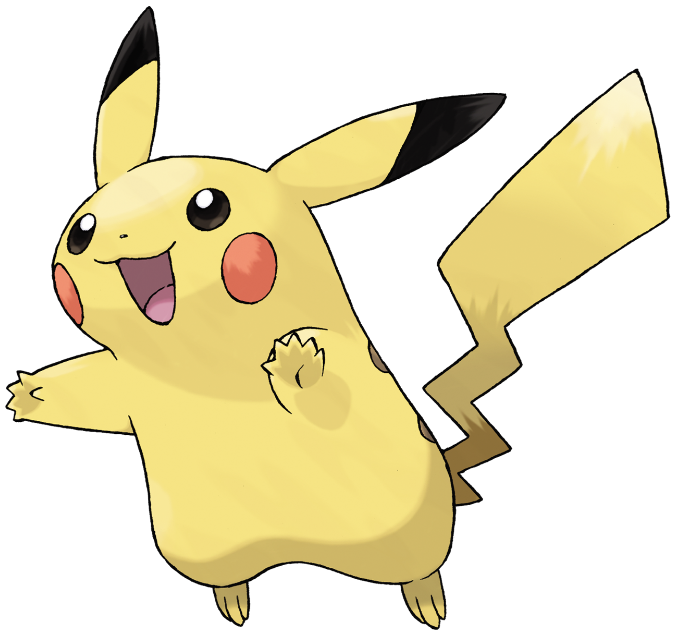
```
a <- pokemon %>% filter(pokemon$Name == 'Pikachu')
kable(a)
```

X.	Name	Type.1	Type.2	Total	HP	Attack	Defense	Special.Attack	Special.Defense	Speed	Generatio
25	Pikachu	Electric		320	35	55	40	50	50	90	1

```r
dim(pokemon)
```

```
[1] 800  13
```

Pikachu is present and has the right statistics. Overall, this dataset has 800 observations and 11 variables.



Pikachu, the most iconic pokemon. Introduced in the 1st generation. Source: bulbapedia.bulbagarden.net

```r
str(pokemon)
```

```
'data.frame':   800 obs. of  13 variables:
 $ X.             : int  1 2 3 3 4 5 6 6 6 7 ...
 $ Name           : Factor w/ 800 levels "Abomasnow","AbomasnowMega Abomasnow",..: 81 330 746 747 103 10
 $ Type.1         : Factor w/ 18 levels "Bug","Dark","Dragon",..: 10 10 10 10 7 7 7 7 7 18 ...
 $ Type.2         : Factor w/ 19 levels "","Bug","Dark",..: 15 15 15 15 1 1 9 4 9 1 ...
 $ Total          : int  318 405 525 625 309 405 534 634 634 314 ...
 $ HP             : int  45 60 80 80 39 58 78 78 78 44 ...
 $ Attack         : int  49 62 82 100 52 64 84 130 104 48 ...
 $ Defense        : int  49 63 83 123 43 58 78 111 78 65 ...
 $ Special.Attack : int  65 80 100 122 60 80 109 130 159 50 ...
 $ Special.Defense: int  65 80 100 120 50 65 85 85 115 64 ...
 $ Speed          : int  45 60 80 80 65 80 100 100 100 43 ...
 $ Generation     : int  1 1 1 1 1 1 1 1 1 1 ...
 $ Legendary      : Factor w/ 2 levels "False","True": 1 1 1 1 1 1 1 1 1 1 ...
```

```r
a <- summary(pokemon)[,c(1:4,12,13)]
b <- summary(pokemon)[,c(5:11)]
kable(a)
```

X.	Name	Type.1	Type.2	Generation	Legendary
Min. : 1.0	Abomasnow : 1	Water :112	:386	Min. :1.000	False:735
1st Qu.:184.8	AbomasnowMega Abomasnow: 1	Normal : 98	Flying : 97	1st Qu.:2.000	True : 65
Median :364.5	Abra : 1	Grass : 70	Ground : 35	Median :3.000	NA
Mean :362.8	Absol : 1	Bug : 69	Poison : 34	Mean :3.324	NA
3rd Qu.:539.2	AbsolMega Absol : 1	Psychic: 57	Psychic : 33	3rd Qu.:5.000	NA
Max. :721.0	Accelgor : 1	Fire : 52	Fighting: 26	Max. :6.000	NA
NA	(Other) :794	(Other):342	(Other) :189	NA	NA

```
kable(b)
```

Total	HP	Attack	Defense	Special.Attack	Special.Defense	Speed
Min. :180.0	Min. : 1.00	Min. : 5	Min. : 5.00	Min. : 10.00	Min. : 20.0	Min. : 5.00
1st Qu.:330.0	1st Qu.: 50.00	1st Qu.: 55	1st Qu.: 50.00	1st Qu.: 49.75	1st Qu.: 50.0	1st Qu.: 45.00
Median :450.0	Median : 65.00	Median : 75	Median : 70.00	Median : 65.00	Median : 70.0	Median : 65.00
Mean :435.1	Mean : 69.26	Mean : 79	Mean : 73.84	Mean : 72.82	Mean : 71.9	Mean : 68.28
3rd Qu.:515.0	3rd Qu.: 80.00	3rd Qu.:100	3rd Qu.: 90.00	3rd Qu.: 95.00	3rd Qu.: 90.0	3rd Qu.: 90.00
Max. :780.0	Max. :255.00	Max. :190	Max. :230.00	Max. :194.00	Max. :230.0	Max. :180.00
NA	NA	NA	NA	NA	NA	NA

The dataset has a nice distribution of variables, 2 categorical, 1 binary and 8 interval. The X variable is a pokemon Id. Note that it is not unique, because later *generations* added new *evolutions* for existing pokemon. **Type 1** designates the primary type of the pokemon - it influences it's strengts and weekneses (e.g. fire pokemon are weak against water pokemon) and it's overall design. Some pokemon also have a second type. **Generation** is the number of generation this pokemon it's from. First generation originated in 1996, and the 6th one in 2013. This dataset is slightly old, as it lack the 7 generation from 2016.
Next are the statisticks for each pokemon.The **Total** vairable is a simple sum of all statistics. **HP** stands for Hit Points, the pokemon *health*. **Attack** signifies how much damage can it do, and it's compared to the **defense** of the enemy's pokemon. **Special attack** is simmilar to normal attack, but it's compared to the **Special Defense**. **Speed** indicates which pokemon attacks first in a given round. Aside from these basic stats, each pokemon has it's *moves*, *abilities* and other variables not included in this dataset. That being said, the basic statline and it's distribution has a great impact on how powerful a given pokemon is.

Subsetting the dataset

```
# All stats
poke <- pokemon[, c(5:11)]

# All stats, legendary
poke <- pokemon %>% filter(Legendary == 'True')
poke <- poke[, c(6:11)]
poke2 <- pokemon %>% filter(Legendary == 'True')

# Stats without the Total
poke <- pokemon[, c(6:11)]
```

For futher analisys, there are 3 coiches of variables. First is the one including all statistics of a given pokemon - Total, HP, Attack, Defense, Special Attack, Special Defense and Speed. Generations is ignored, since it is a nominal variable.

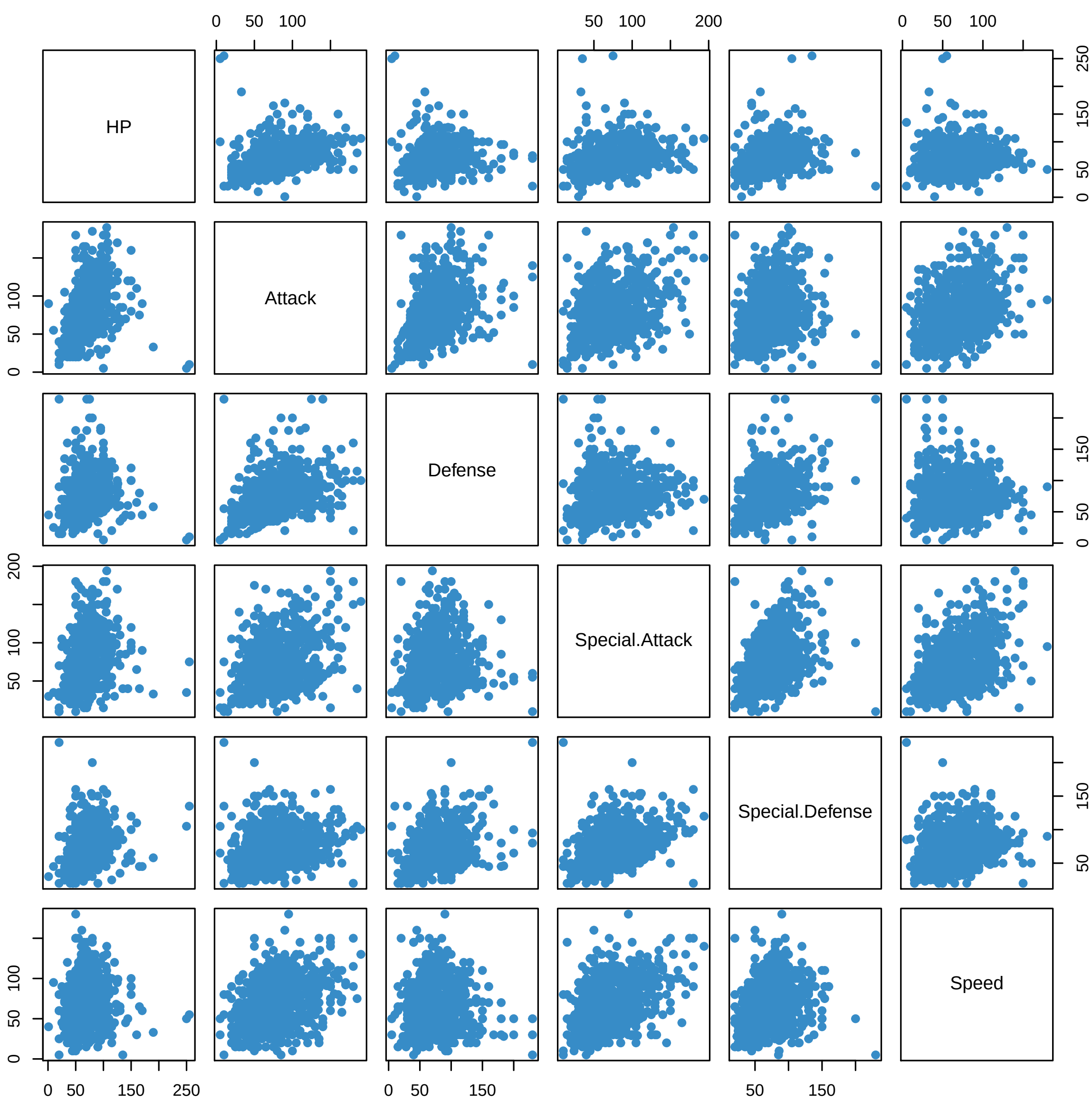
Second choise are only the Legendary Pokemon - a small (65) subset of all pokemon, containign the rarest and most powerful pokemon.

Third choise it again all pokemon, but withous the Total statistic, since it is produced but all other statistics already included.

Visual analisys

We can take a look at the overall distirbutions by plotting the scatter plots for all variables.

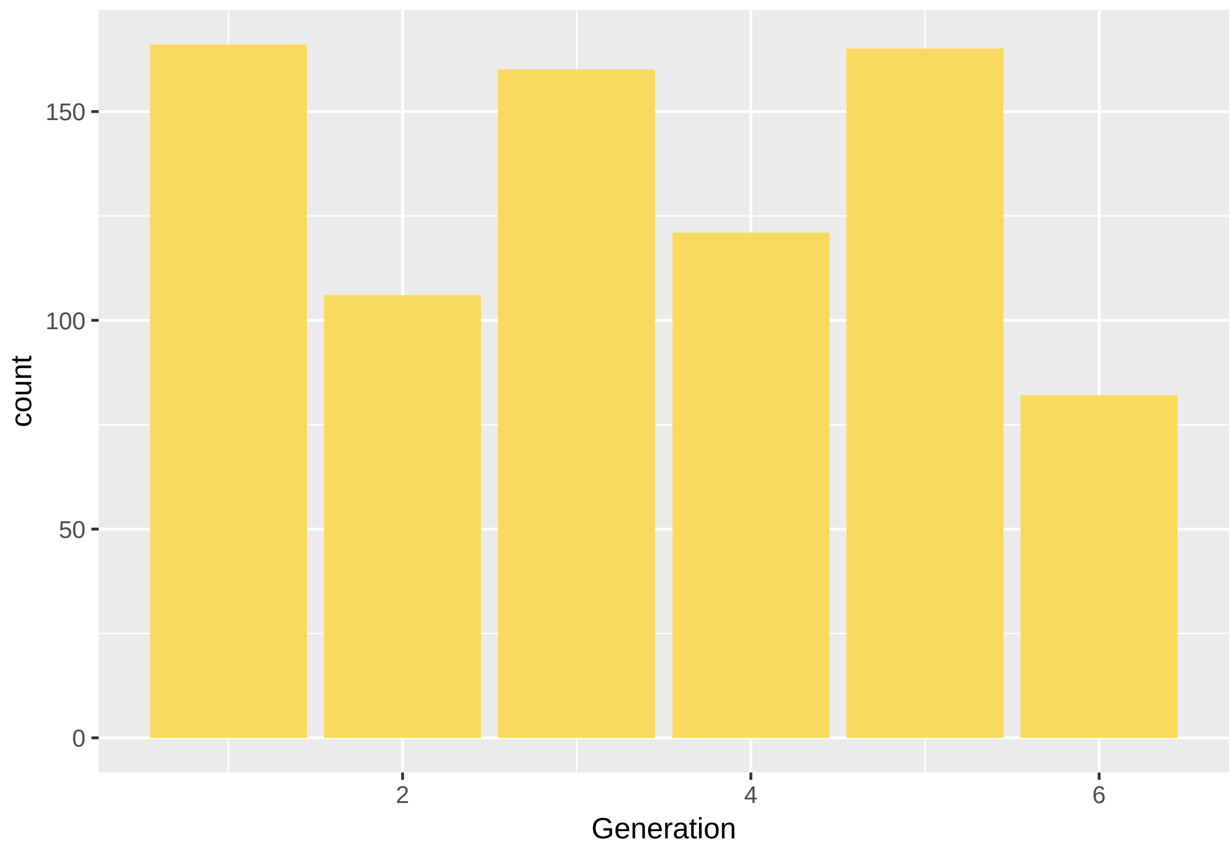
```
plot(poke, col = cBlue, pch = 19)
```



All variables seem to be nicely distributed, the doesn't seem to be any strong correlations. The HP statitick seems to be the most independent. Some outliers can be seen, but they aren't very strong.

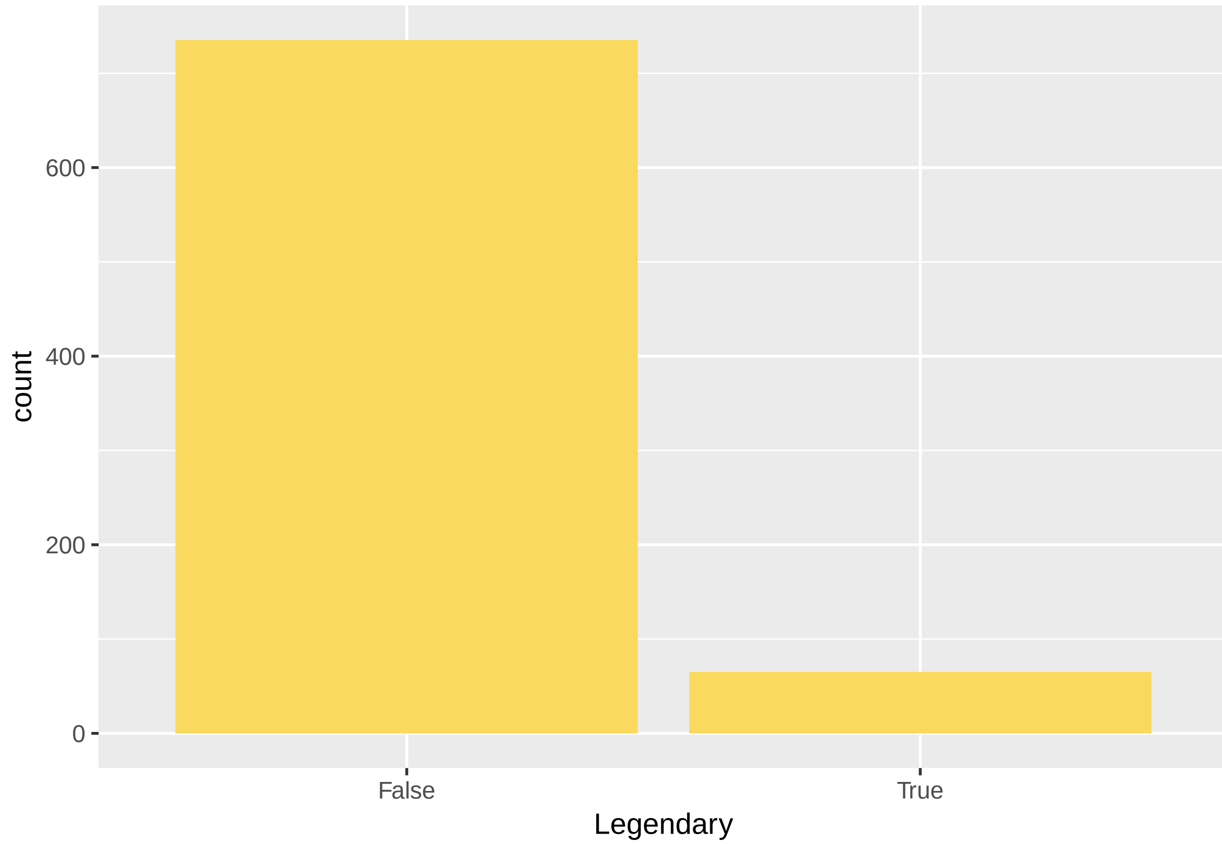
Now, we can take a closer look at the distributions of some of the variables.

```
ggplot(pokemon) + geom_bar(aes(x = Generation), fill = cYellow)
```



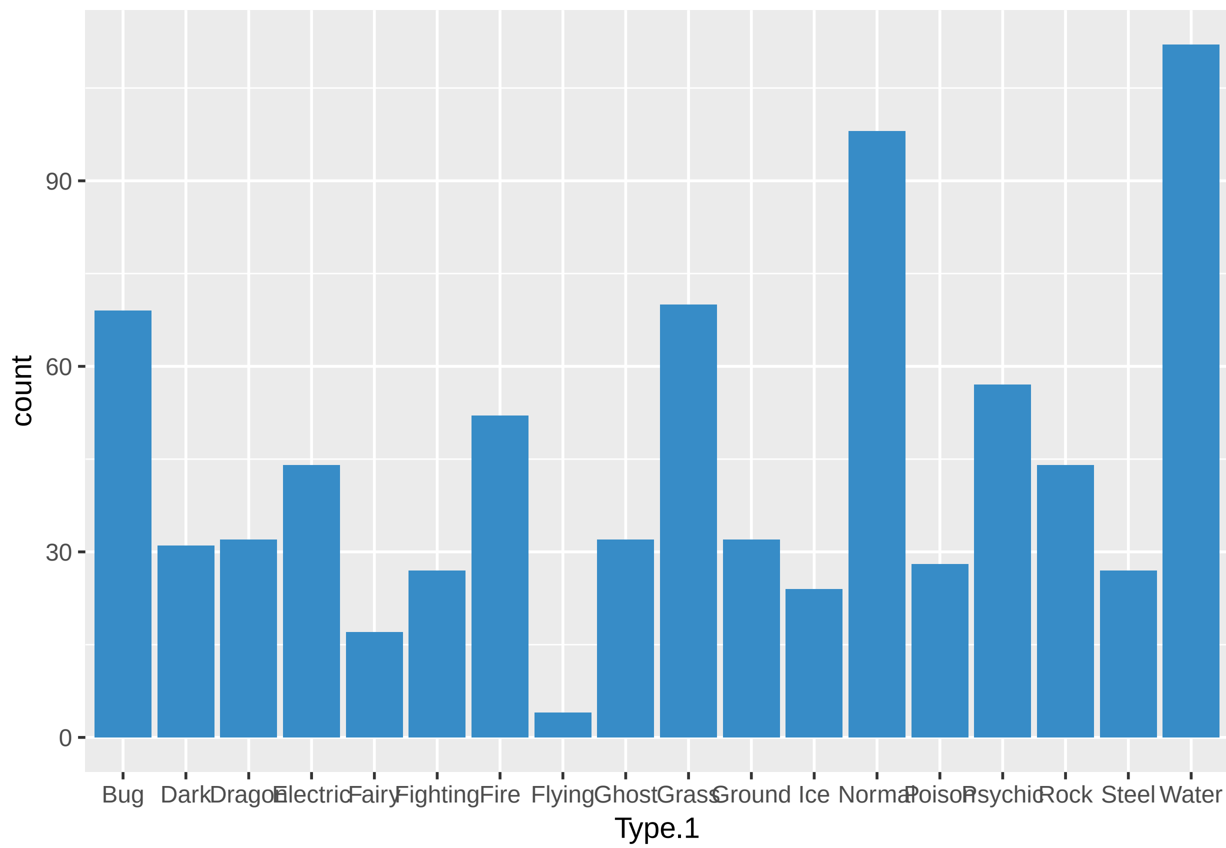
The number of pokemon per generation seems to follow a "tick-tok" distribution - each even generation has a significanly lower number of new pokemon.

```
ggplot(pokemon) + geom_bar(aes(x = Legendary), fill = cYellow)
```



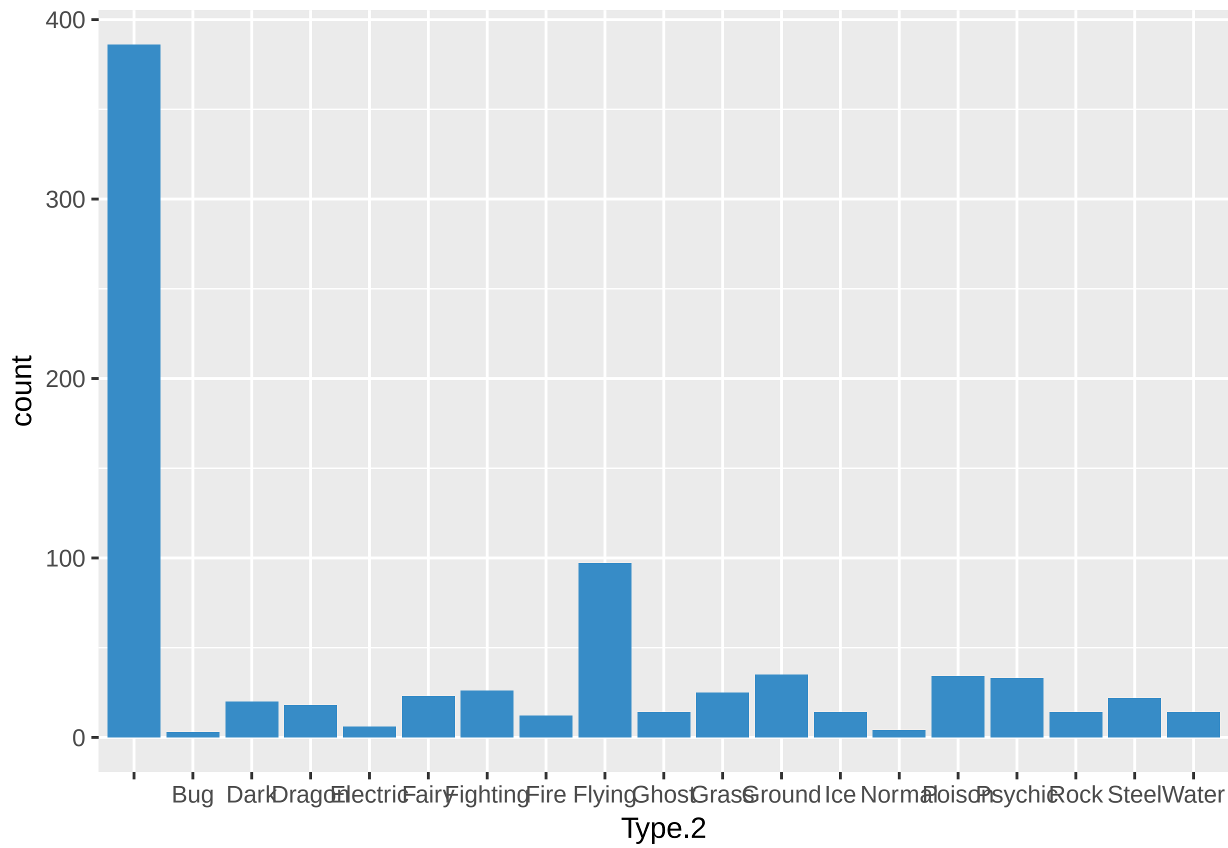
There is a large difference beetwen the number of legendary pokemon (65) and normal ones (735)

```
ggplot(pokemon) + geom_bar(aes(x = Type.1), fill = cBlue)
```



The number of pokemon in each cattegory in not distibutet evenly. There are a lot more of the *water*, *normal* and *bug* types of pokemons. These might be a design decision, since a lot of the time in-game take place in forrests and simmilar locations. Also notable is the *flying* type - almost no pokemon has it as it's first type.

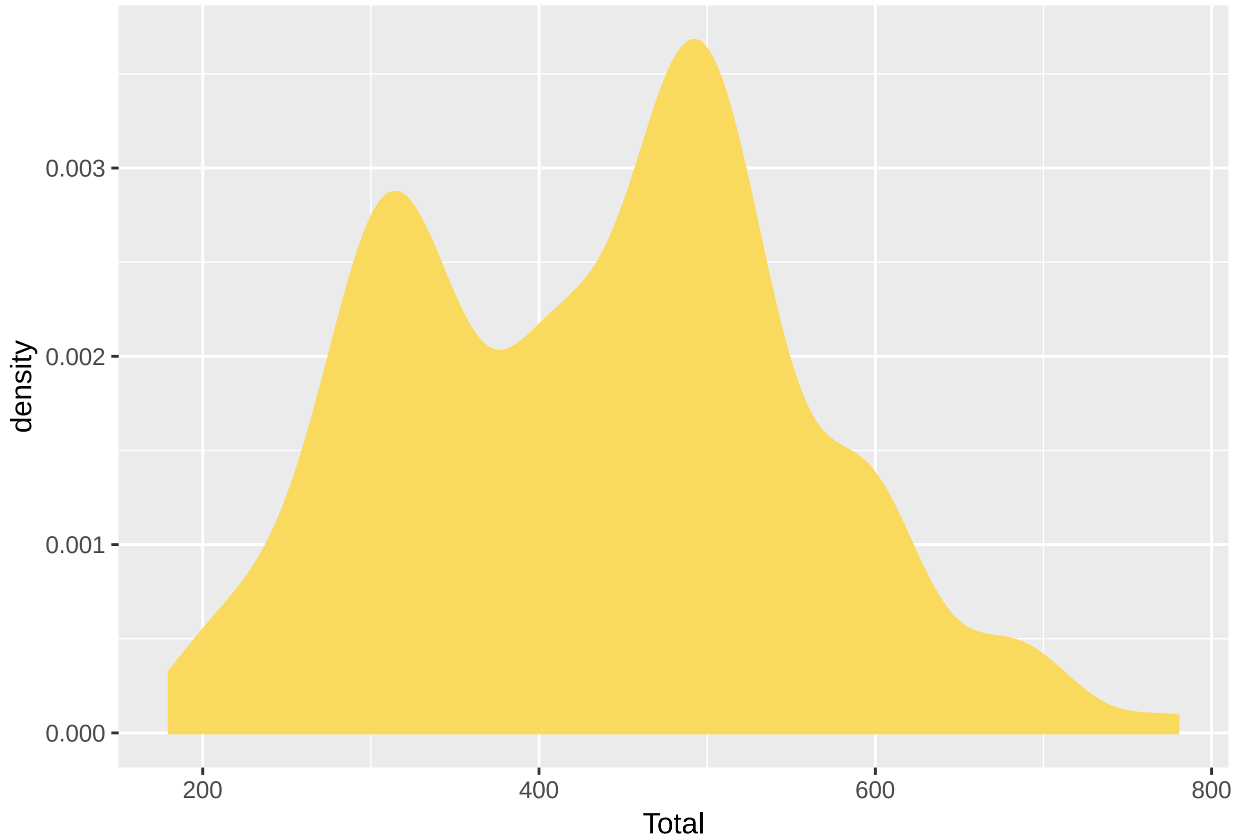
```
ggplot(pokemon) + geom_bar(aes(x = Type.2), fill = cBlue)
```



Most pokemon don't have a second type - but when they do, it's typically *flying*.

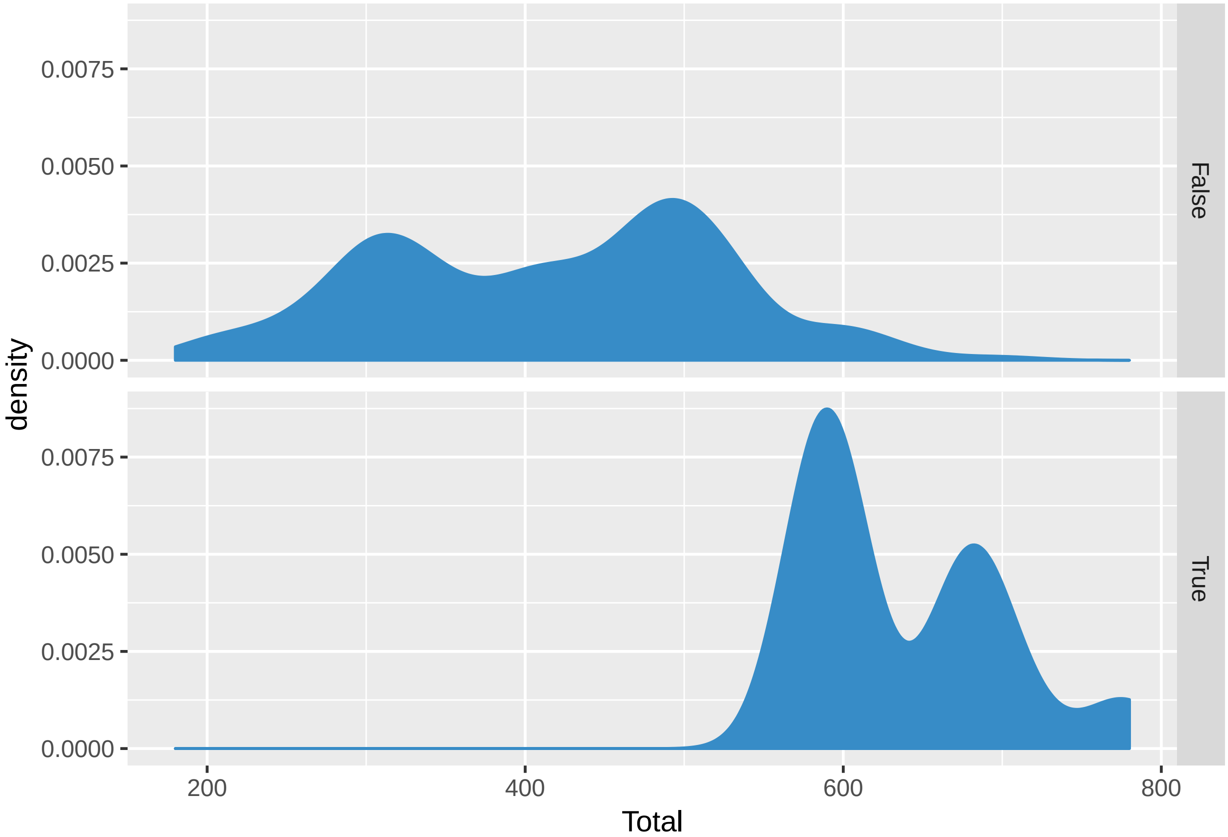
We can take a look how is the Total variable distributed. It gives a good estimation on how powerful a certain pokemon can be.

```
ggplot(pokemon) + geom_density(aes(x = Total), fill = cYellow, colour = cYellow)
```



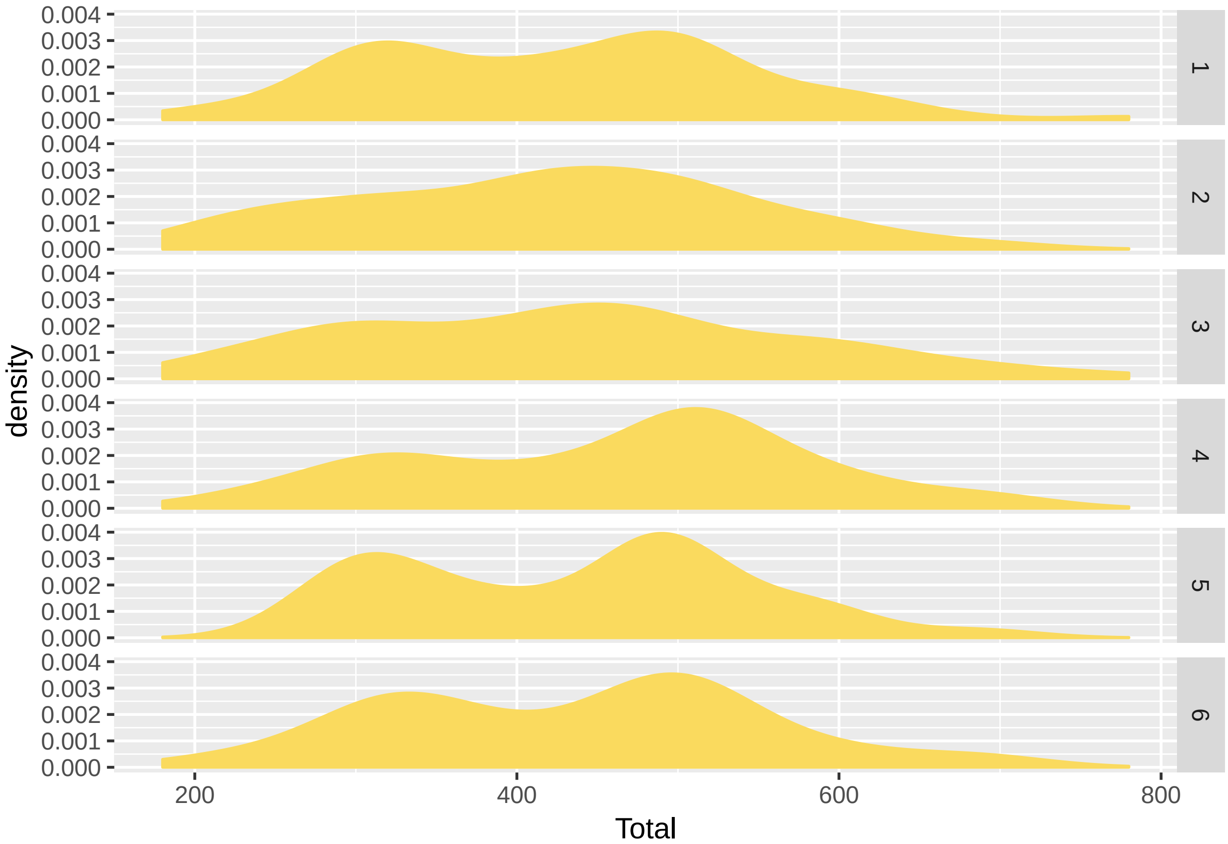
The distribution has two strong peaks - one at around 300, and the other at 500. These is because pokemon has a machanism of *evolution*. Many pokemon (but not all) have at least two forms. One that is weaker (e.g. a bug) and can be encountered earlier in the game, and second (e.g. a butterfly) that is a result of a *evolution* that a pokemon undergoes when it reaches a certain level of expirience. To give a sense of progress, the evolved formes are typically much stronger.

```
ggplot(pokemon) + geom_density(aes(x = Total), fill = cBlue, colour = cBlue) + facet_grid(Legendary ~ .)
```



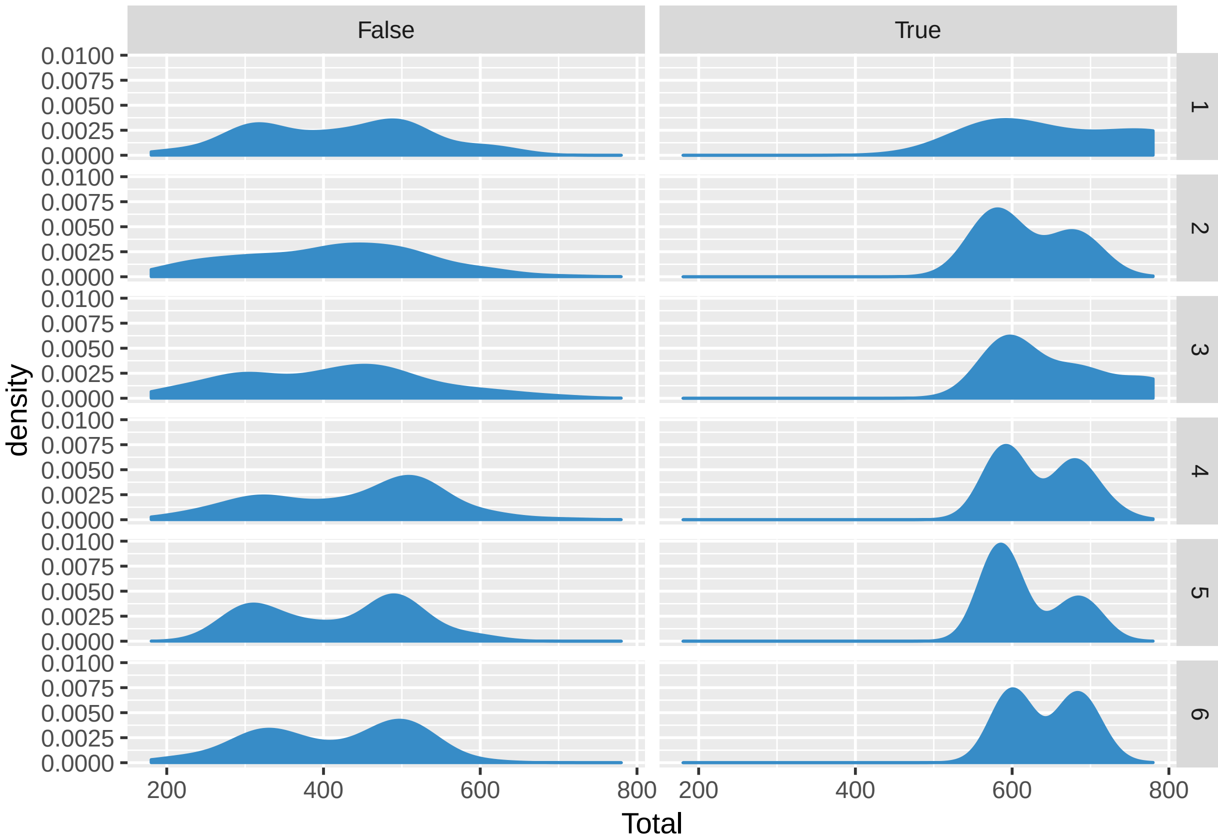
When comparing the Total beetwen Legendary and non-legendary pokemon, it is claer that most high spots are occupied by the legendary pokemon. They do seem to also follow ths interesting two-peak distribution, even though almost none of the evolve.

```
ggplot(pokemon) + geom_density(aes(x = Total), fill = cYellow, colour = cYellow) + facet_grid(Generatio
```



We can now see, if there exist any "Power Creep" in pokemon. "Power creep" is a idea, that each new version on generation is stronger than the previous one, typically to help sell new editions. This is a significant problem in many card games. When comparing the Total beetwen generations the "power creep" seems to be nonexitant. It has to be noted that the tow-peak distribution seems to be getting stronger with each generation.

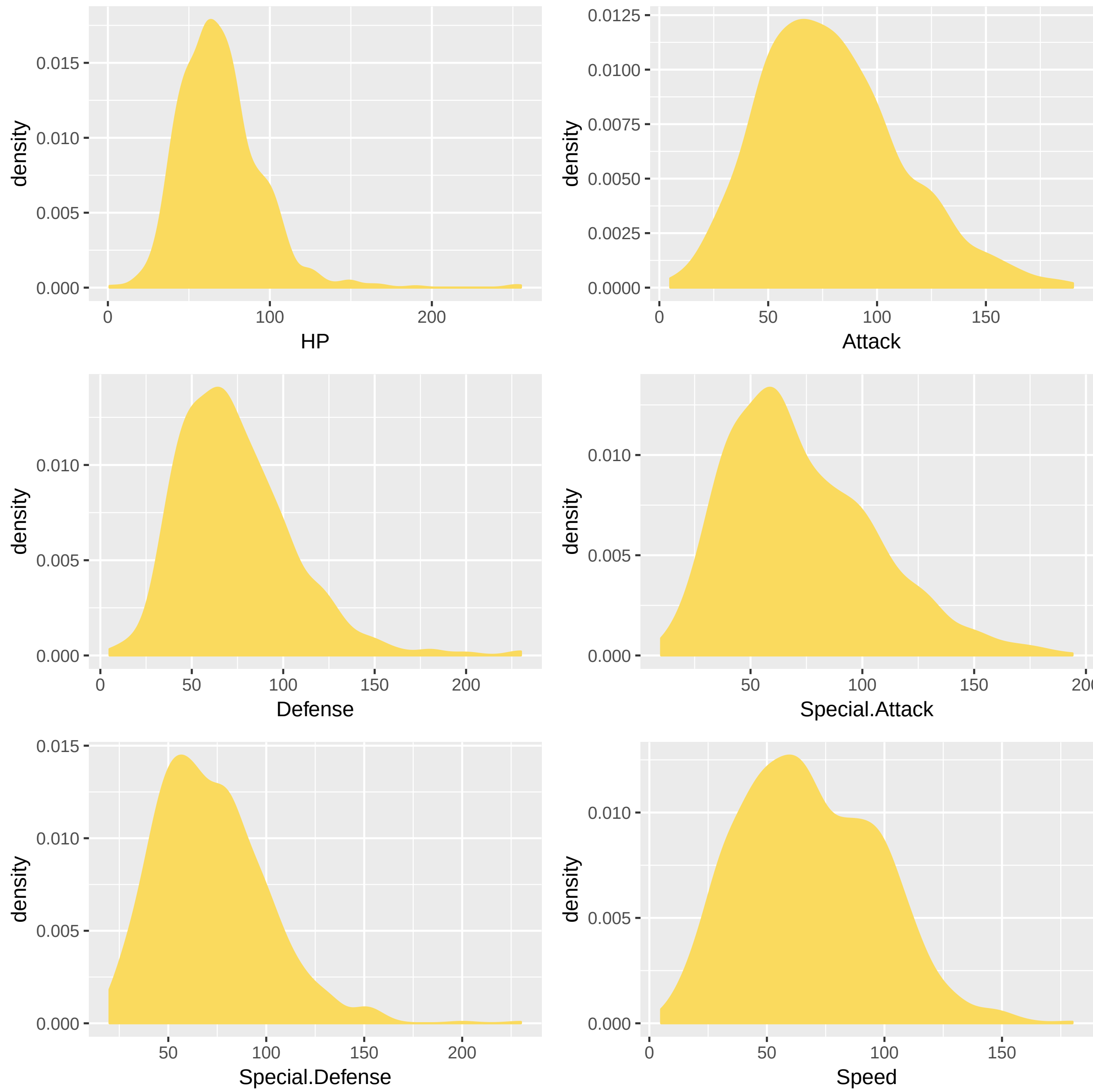
```
ggplot(pokemon) + geom_density(aes(x = Total), fill = cBlue, colour = cBlue) + facet_grid(Generation ~ 
```



Comparing Total over generations and legendary vs non-legendary pokemon doesn't bring new insights - the distributions seems to be quite consistent.

We can take a look how are the other variables distributed.

```
p1 <- ggplot(pokemon) + geom_density(aes(x = HP), fill = cYellow, colour = cYellow)
p2 <- ggplot(pokemon) + geom_density(aes(x = Attack), fill = cYellow, colour = cYellow)
p3 <- ggplot(pokemon) + geom_density(aes(x = Defense), fill = cYellow, colour = cYellow)
p4 <- ggplot(pokemon) + geom_density(aes(x = Special.Attack), fill = cYellow, colour = cYellow)
p5 <- ggplot(pokemon) + geom_density(aes(x = Special.Defense), fill = cYellow, colour = cYellow)
p6 <- ggplot(pokemon) + geom_density(aes(x = Speed), fill = cYellow, colour = cYellow)
grid.arrange(p1, p2, p3, p4, p5, p6, nrow = 3)
```



All of them follow a roughly normal distribution, with means around 55. Some outliers in each cattegory can be seen.

Correlation plot

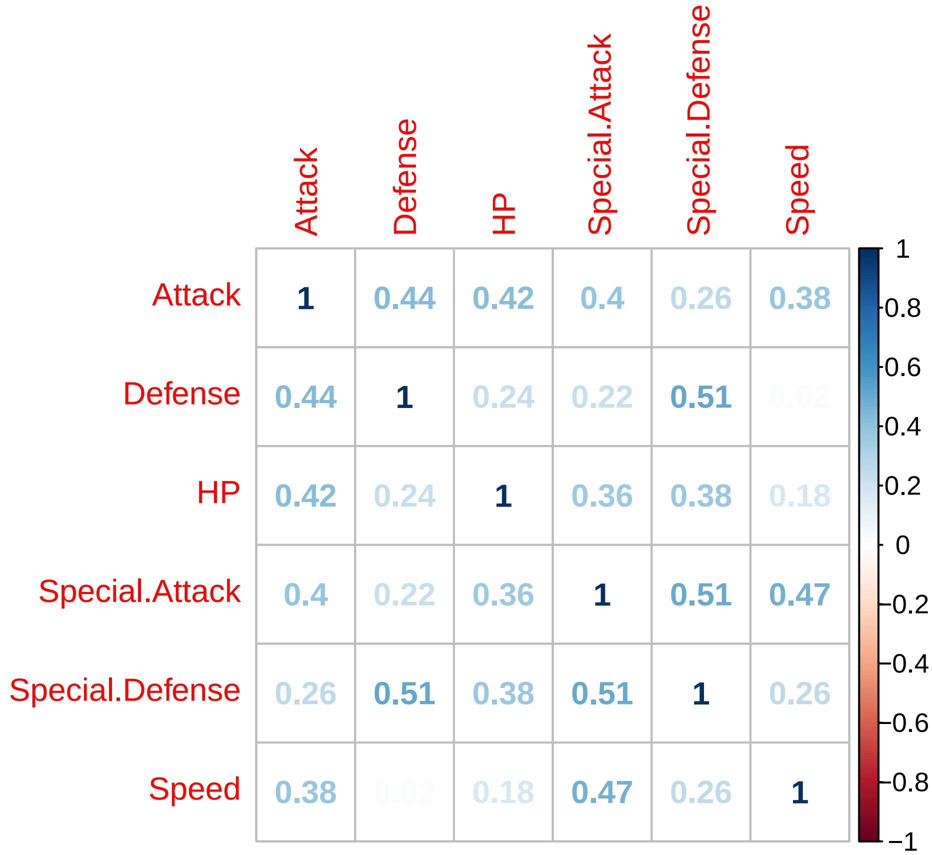
To see how the variables interact with each other, we can compute and display the correlation matrix.

```r
pokemonCorrelation <- cor(pokemon[, c(6:11)], method="pearson")
print(pokemonCorrelation, digits=2)
```

```
                 HP Attack Defense Special.Attack Special.Defense Speed
HP             1.00   0.42   0.240           0.36            0.38 0.176
Attack         0.42   1.00   0.439           0.40            0.26 0.381
Defense        0.24   0.44   1.000           0.22            0.51 0.015
Special.Attack 0.36   0.40   0.224           1.00            0.51 0.473
```

```
Special.Defense 0.38    0.26    0.511            0.51            1.00 0.259
Speed           0.18    0.38    0.015            0.47            0.26 1.000
```

```r
corrplot(pokemonCorrelation, order ="alphabet", method = 'number')
```



There are no strong correlations. Some mild ones can be observed between Defense and Special Defense, and Special Defense and Special Attack. Also note a absolute lack of correlation between Defense and Speed.

Multidimensional Scaling

Classical multidimentional scaling

After exploring the dataset, we can move to multidimensional scaling. The general idea it that aldough we have 6 (or 7 when counting the Total) variables, there are similarities beetwen them, that would allow us to reduce the number of variables to 2, all without loosing too much information.

First, we have to compute the distance matrix for our data subset, then feed it into the cmdscale function from the stats package. As a result we'll be able to reduce the number of variables to 2.

```r
poke.dist<-dist(poke)
a <- as.matrix(poke.dist)[1:10, 1:10]
kable(a)
```

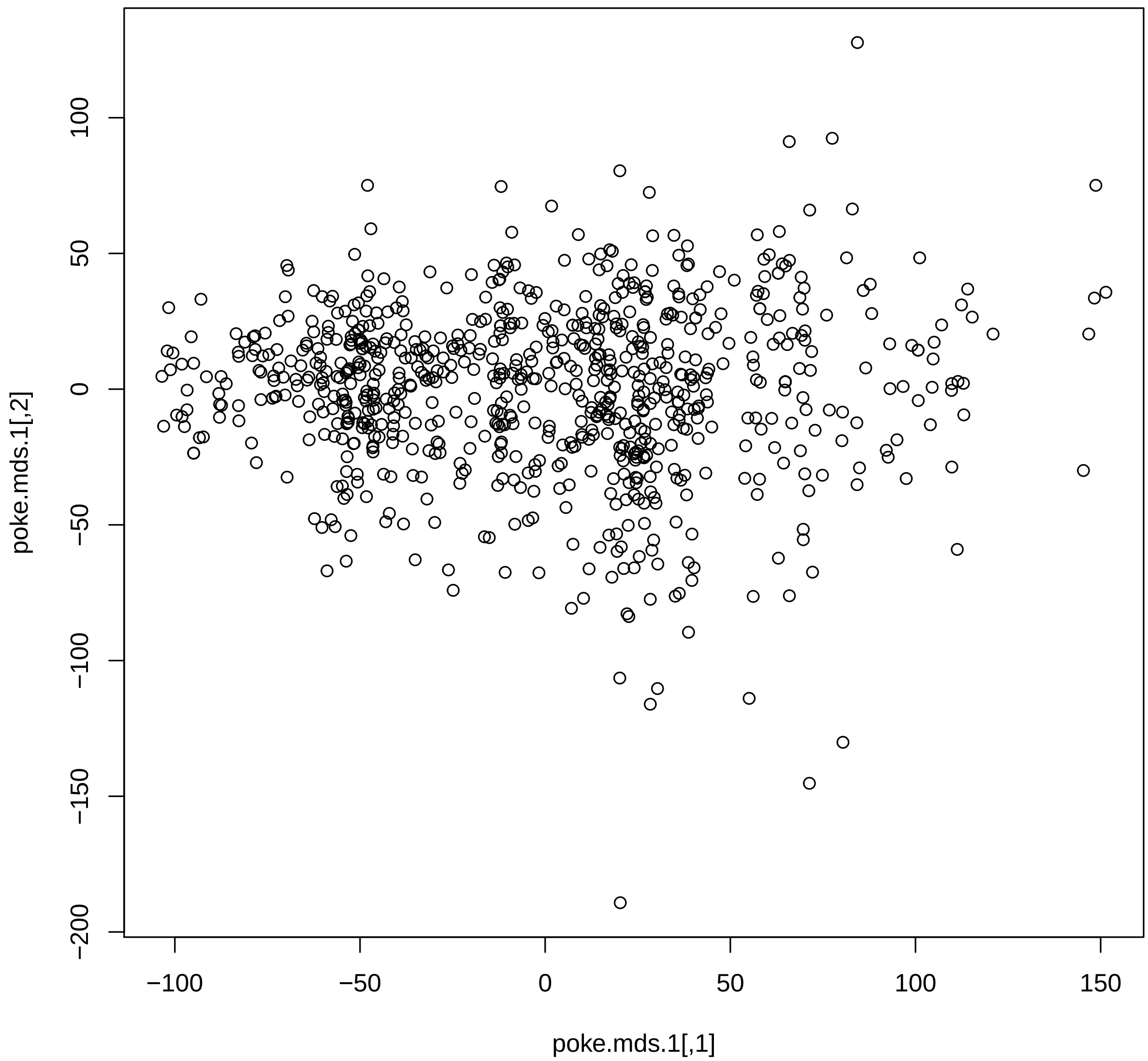
1	2	3	4	5	6	7	8	9	10
0.00000	35.56684	84.52810	129.61867	27.03701	43.87482	92.28218	138.36184	138.98201	22.09072
35.56684	0.00000	48.98979	95.95832	47.60252	25.65151	59.15235	106.66302	106.67240	43.60046
84.52810	48.98979	0.00000	52.99057	92.22798	55.29919	27.18455	67.94851	67.96323	89.11229

1	2	3	4	5	6	7	8	9	10
129.61867	95.95832	52.99057	0.00000	139.11865	103.89418	63.86705	52.31635	61.95966	130.58714
27.03701	47.60252	92.22798	139.11865	0.00000	39.74921	92.84934	139.92498	143.87842	36.12478
43.87482	25.65151	55.29919	103.89418	39.74921	0.00000	53.30103	104.23531	107.42905	52.64029
92.28218	59.15235	27.18455	63.86705	92.84934	53.30103	0.00000	60.38212	61.64414	98.95454
138.36184	106.66302	67.94851	52.31635	139.92498	104.23531	60.38212	0.00000	59.21149	141.72509
138.98201	106.67240	67.96323	61.95966	143.87842	107.42905	61.64414	59.21149	0.00000	148.96980
22.09072	43.60046	89.11229	130.58714	36.12478	52.64029	98.95454	141.72509	148.96980	0.00000

```r
poke.mds.1 <- cmdscale(poke.dist, k=2)
b <- summary(poke.mds.1)
kable(b)
```

V1	V2
Min. :-103.496	Min. :-189.23
1st Qu.: -44.222	1st Qu.: -16.69
Median : 3.725	Median : 3.31
Mean : 0.000	Mean : 0.00
3rd Qu.: 31.978	3rd Qu.: 19.68
Max. : 151.413	Max. : 127.70

```r
plot(poke.mds.1)
```

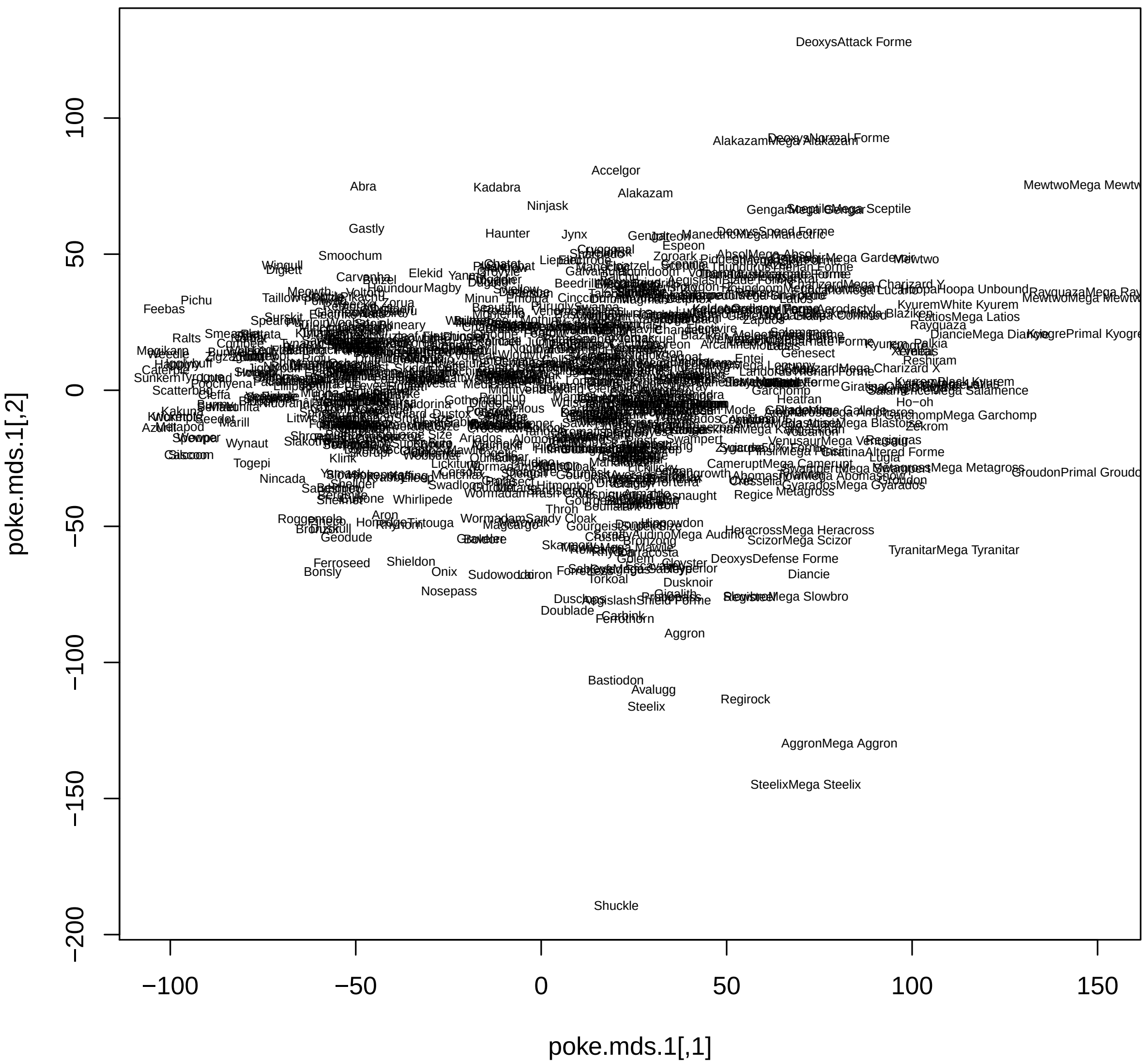


The results look promising - most of the data is located along the x axis, with some outliers on the top, bottom, and the left side. There are two clear major clusters in the data - maybe they have something to do with the dual-peak distribution discovered earilier? The data also follows a kind of "cone" distribution, having a low variance for x < 0 and higher for x > 0.

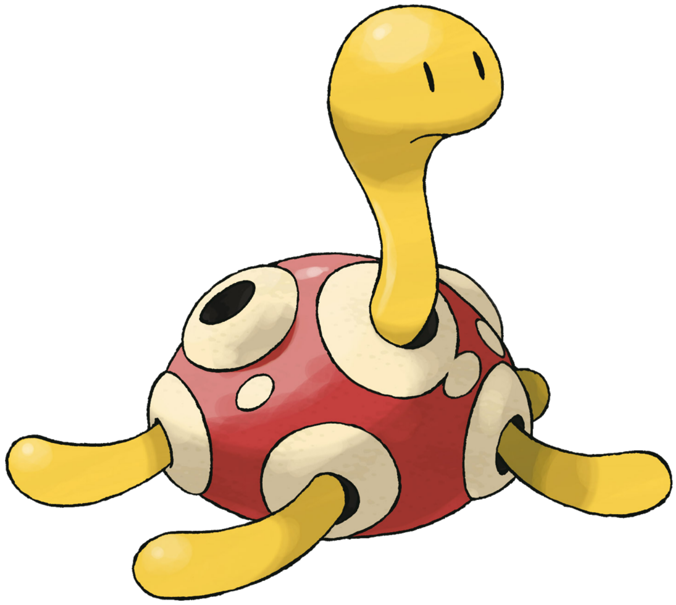
Closer look at the clusters and outliers

We can try to gain more insights by displaying the names of the pokemon, and comparing the results to a outside source, the pokemon wikipedia.

```
plot(poke.mds.1, type = 'n')
text(poke.mds.1, labels = pokemon$Name, cex=0.5, adj = 0.5)
```



The plot looks crowded, but we can gain much insight from it. First, we can take a look at the outlier at the bottom, in the center **Shuckle**. According to Bulbapedia *“Shuckle has the most extreme stat distribution of any Pokémon, being either the best or in the bottom three of every base stat”*. This is a great information, because it means the MDS algorithm was able to idenfity the this outlier.



Shuckle, looslely based on the real-life endoliths. Introduced in the 2nd generation. Source: bulbapedia.bulbagarden.net

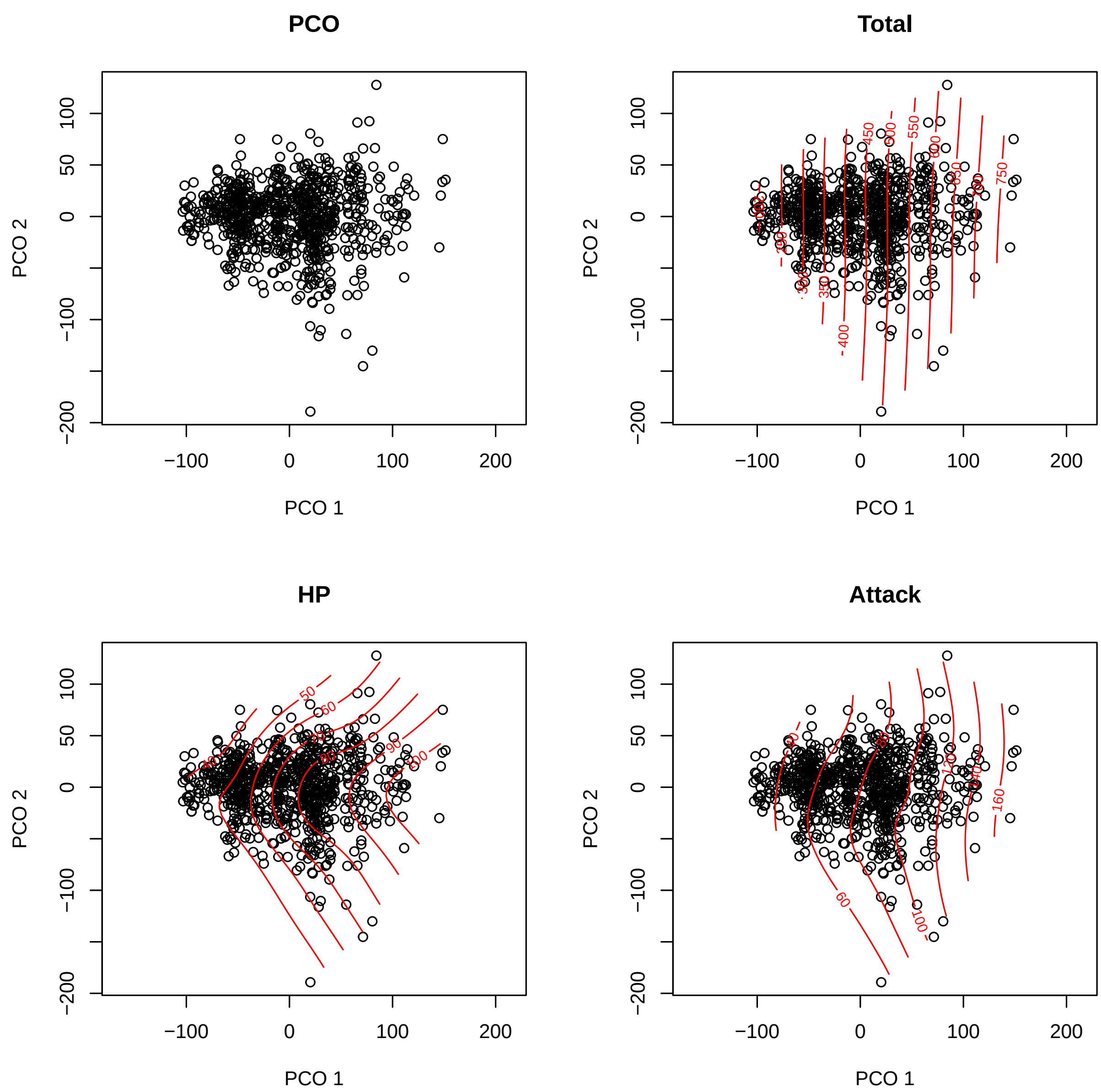
Other interesting pokemon are the Pichu, that can be found on the left-hand side in the center, that is one of the weakes pokemons, and Mewtwo Mega, that can be found on the right-hand side, also in the center, that is one of the most powerful pokemon.

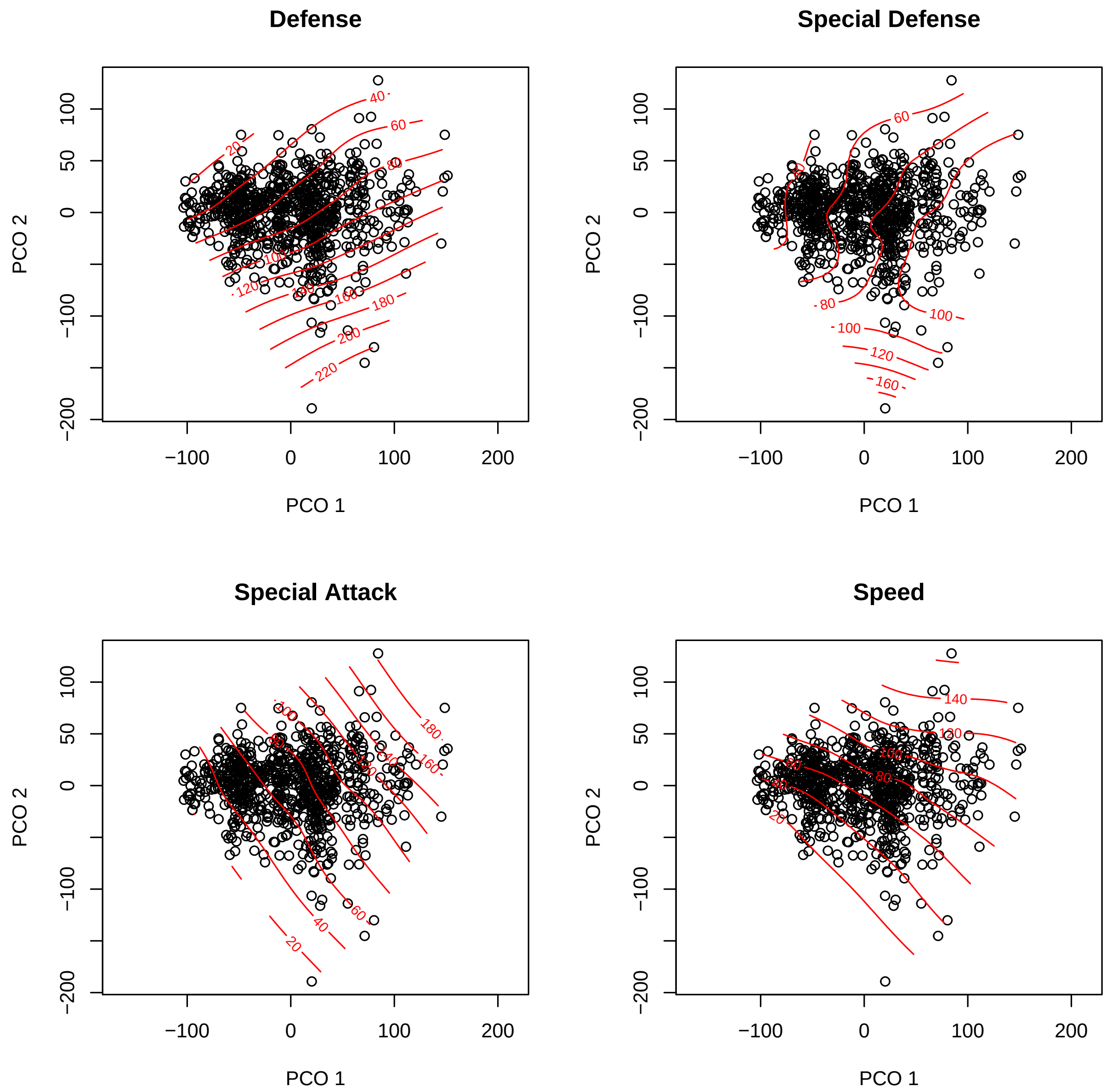
Idenfyting the infuence of different variables on the MDS

To see the effects of different variables, we can plot their surfaces on the scatter plots. We'll be using the pco() function that is a wrapper for the cmdscale() to enable better ploting.

```
poke <- pokemon[, c(5:11)]
poke.mds.2<-pco(poke.dist, k=2)
par(mfrow=c(2,2))
plot(poke.mds.2)
title(main = "PCO")

plot(poke.mds.2)
title(main = "Total")
surf(poke.mds.2, poke$Total)

plot(poke.mds.2)
title(main = "HP")
surf(poke.mds.2, poke$HP)

plot(poke.mds.2)
title(main = "Attack")
surf(poke.mds.2, poke$Attack)

plot(poke.mds.2)
title(main = "Defense")
surf(poke.mds.2, poke$Defense)
```

```r
plot(poke.mds.2)
title(main = "Special Defense")
surf(poke.mds.2, poke$Special.Defense)

plot(poke.mds.2)
title(main = "Special Attack")
surf(poke.mds.2, poke$Special.Attack)

plot(poke.mds.2)
title(main = "Speed")
surf(poke.mds.2, poke$Speed)

par(mfrow=c(1,1))
```





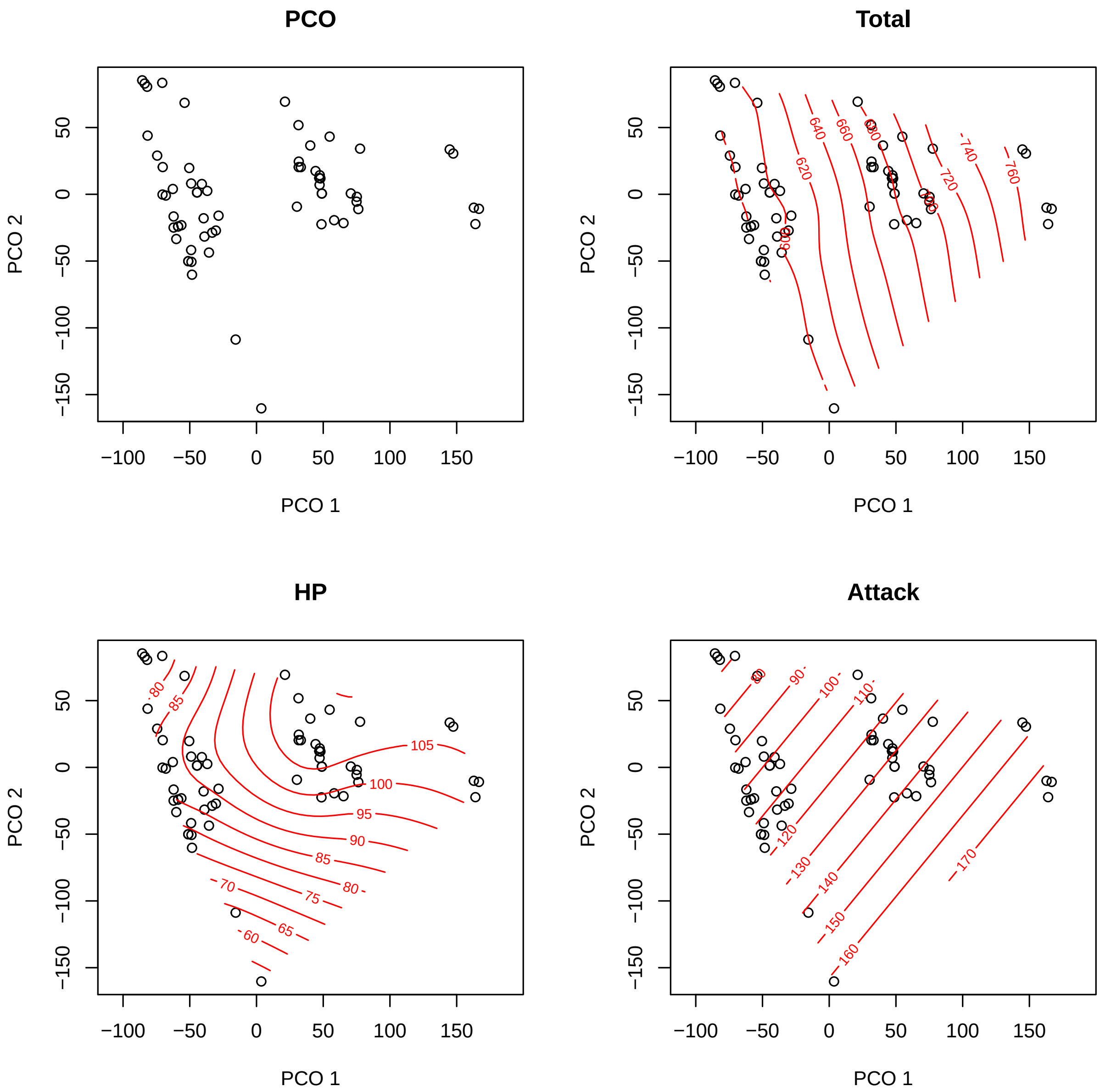
We can see many interesting interactions. It seems that going along the X axis to the rigth directly increases the Total and Attack statistics. The HP also increases, but only if we stay near the 0 on the Y axis. When analising the Y axis, it seems that going down increases the Defense and Special Defense, while going up and to the left increases the Special Attack and Speed.

From this, we can see that there is a kind of a trade-off system. The X axis functions as a "power lever", gennerally the further right the more powerful the pokemon.
All powerful pokemon have higher attack, and Total. As for the other stats, there seems to be 3 paths - the can stay at $Y \sim 0$, and get more HP, they can get below $Y < 0$, and get Defense and Special Defense, or they can go $Y > 0$, and get Special Attack and Speed.

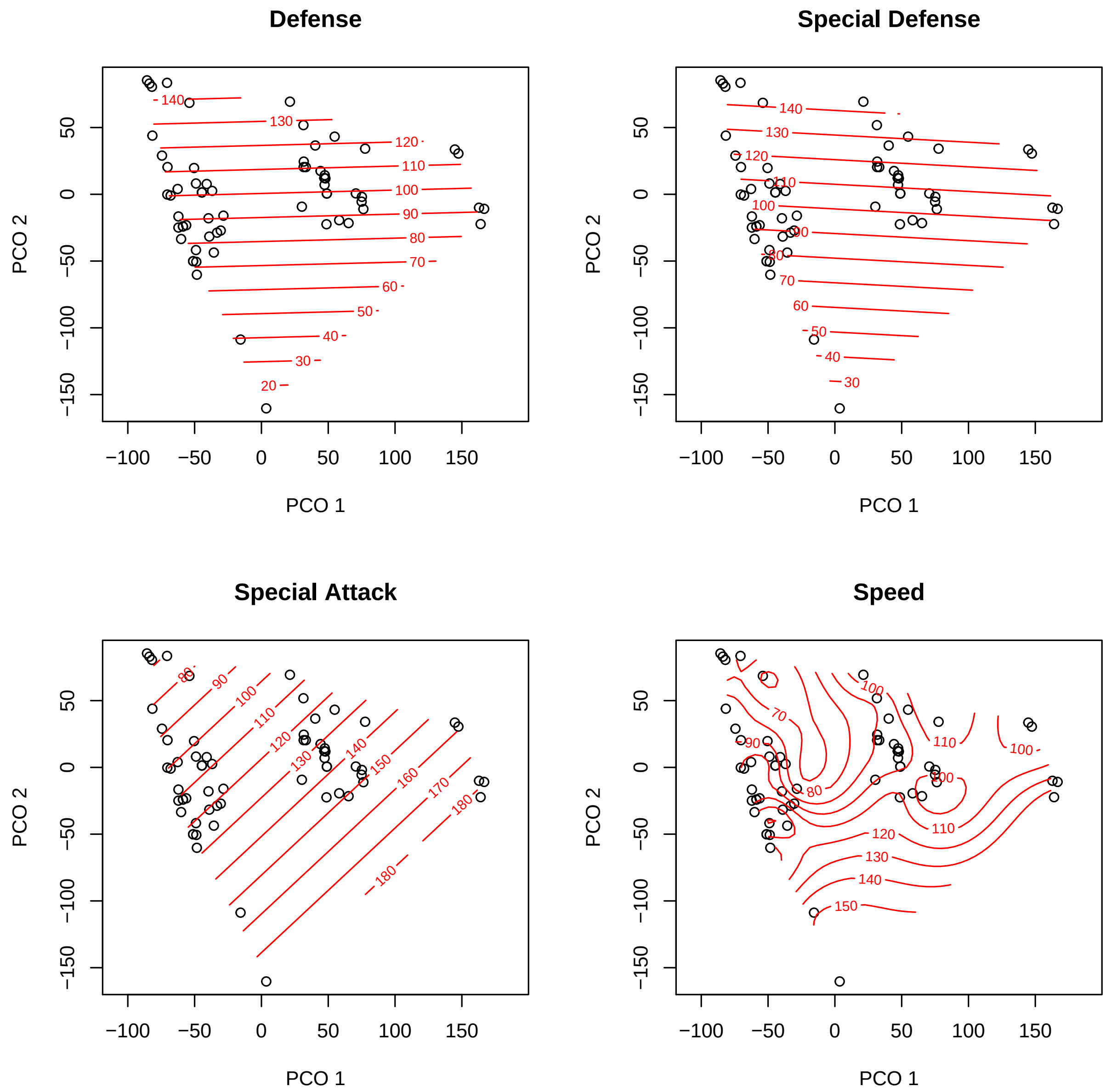
It seems that for the most powerful pokemon, they can be either Agressive ($Y > 0$), Defensive ($Y < 0$), or well-rounded ($Y \sim 0$). It also looks like the differences are far smaller for the weker pokemon.

To investigate this further, we can do the same analisys only for the legendary pokemon.

```r
poke <- pokemon %>% filter(Legendary == 'True')
poke <- poke[, c(5:11)]
poke.dist<-dist(poke)
poke.mds.2<-pco(poke.dist, k=2)
par(mfrow=c(2,2))
plot(poke.mds.2)
title(main = "PCO")

plot(poke.mds.2)
title(main = "Total")
surf(poke.mds.2, poke$Total)

plot(poke.mds.2)
title(main = "HP")
surf(poke.mds.2, poke$HP)

plot(poke.mds.2)
title(main = "Attack")
surf(poke.mds.2, poke$Attack)

plot(poke.mds.2)
title(main = "Defense")
surf(poke.mds.2, poke$Defense)

plot(poke.mds.2)
title(main = "Special Defense")
surf(poke.mds.2, poke$Special.Defense)

plot(poke.mds.2)
title(main = "Special Attack")
surf(poke.mds.2, poke$Special.Attack)

plot(poke.mds.2)
title(main = "Speed")
surf(poke.mds.2, poke$Speed)

par(mfrow=c(1,1))

poke <- pokemon[, c(6:11)]
```



PCO
Total
HP
Attack
PCO 1
PCO 2



The results are simmilar, but there are some differences. As prevously, the X axis serves as a kind of "power level". On the Y axis, it looks like that for Y > 0 HP, Defense and Special Defense increase, and for the Y < 0, the Attack and Special Attack increase. Note that while Attack and Special Attack seem to be also correlated with the power level of the creature, this in not true for Defense and Special Defense. Speed seems to be more individual, aldough favouring the Y < 0 side.

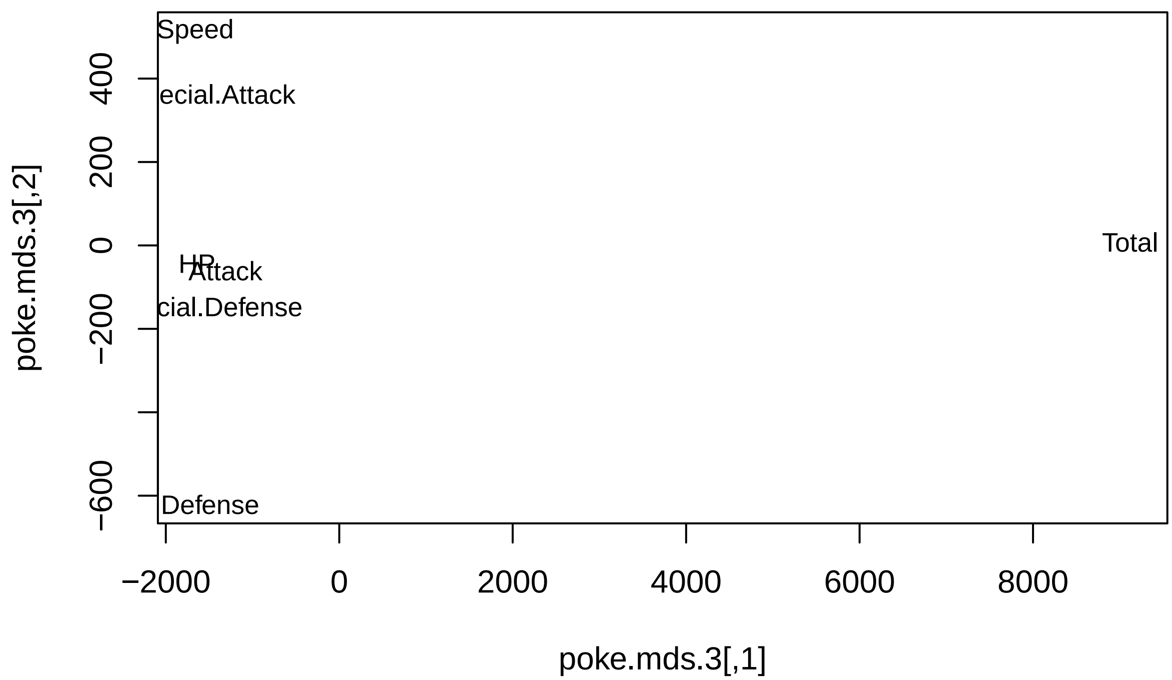
MDS on variables

We can test this interactions beetwen variables, by performing the MDS on a transposed subset of our dataset.

```
poke.dist.t<-dist(t(pokemon[, c(5:11)]))
poke.mds.3<-cmdscale(poke.dist.t, k=2)
a <- summary(poke.mds.3)
kable(a)
```

V1	V2
Min. :-1660	Min. :-621.05
1st Qu.:-1589	1st Qu.:-106.92
Median :-1490	Median : -42.91
Mean : 0	Mean : 0.00
3rd Qu.:-1395	3rd Qu.: 182.13
Max. : 9118	Max. : 513.54

```
plot(poke.mds.3, type = 'n')
text(poke.mds.3, rownames(poke.mds.3), cex=0.8, adj = 0.5)
```



The X axis was taken wholey by the Total variable. On the Y axis, we can see a partial confirmation of the conslusions from above - indeed the Speed and Special Attack occupy the Y > 0, and Defense the Y < 0. Only Special Defense seems to be closer to Attack and HP.

MDS on variables - excluding the Total

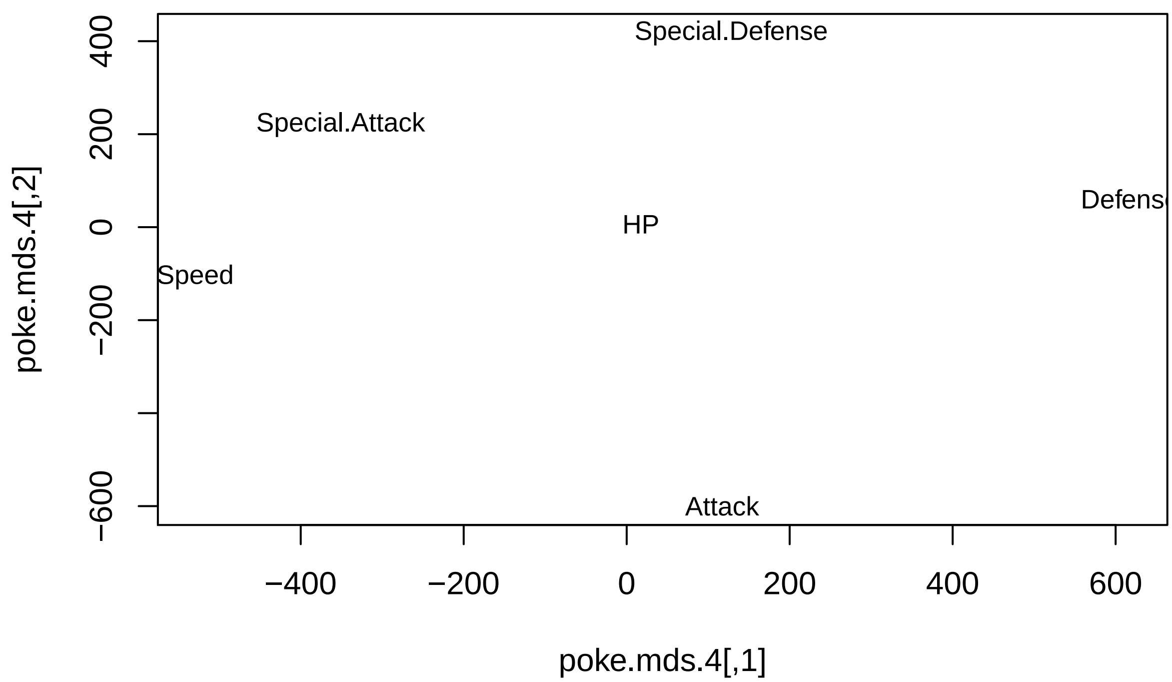
To see the differences better, we can perform the analisys again, this time without the Total varaible.

```
poke.dist.t.2<-dist(t(pokemon[, c(6:11)]))
poke.mds.4<-cmdscale(poke.dist.t.2, k=2)
a <- summary(poke.mds.4)
kable(a)
```

V1	V2
Min. :-529.41	Min. :-599.9

V1	V2
1st Qu.:-258.89	1st Qu.: -78.4
Median : 67.26	Median : 33.9
Mean : 0.00	Mean : 0.0
3rd Qu.: 125.48	3rd Qu.: 181.0
Max. : 617.60	Max. : 418.0

```
plot(poke.mds.4, type = 'n')
text(poke.mds.4, rownames(poke.mds.4), cex=0.8, adj = 0.5)
```



Removing the total variable gives a better picture - on this graph the left-top corner represents the Agresive group, the right-top the Defensive, and the bottom-middle the Well-rounded group.

Testing the Goodness of Fit

To test how well the MDS algorith was able to approximate the original dataset, we can compare it to a stress of a MDS algorith run on a random disssimilarity matrix.

```
poke <- pokemon[, c(6:11)]
poke.dist <- dist(t(poke))
poke.mds.4 <- mds(poke.dist, ndim=2,  type="ordinal")
poke.mds.4
```

```
Call:
mds(delta = poke.dist, ndim = 2, type = "ordinal")
```

```
Model: Symmetric SMACOF
Number of objects: 6
Stress-1 value: 0.065
Number of iterations: 35
```

```
stress.random.matrix <- randomstress(n=800, ndim=2, nrep = 1)
poke.mds.4$stress/ mean(stress.random.matrix)
```

```
[1] 0.1104603
```

The resulting coefficient is 0.11, which is a fair result. We could improve it by adding the 3rd dimension, but that makes plots much less clear.

Principal Value Decomposition

PCA using the singular value decomposition

After the MDS, we can move to a alternative, the PCA. The prcomp function takes care of the centering and scaling the data.

```
poke <- pokemon[, c(6:11)]

poke.pca.1<-prcomp(poke, center=TRUE, scale.=TRUE)
a <- poke.pca.1$rotation
kable(a, digits = 2)
```

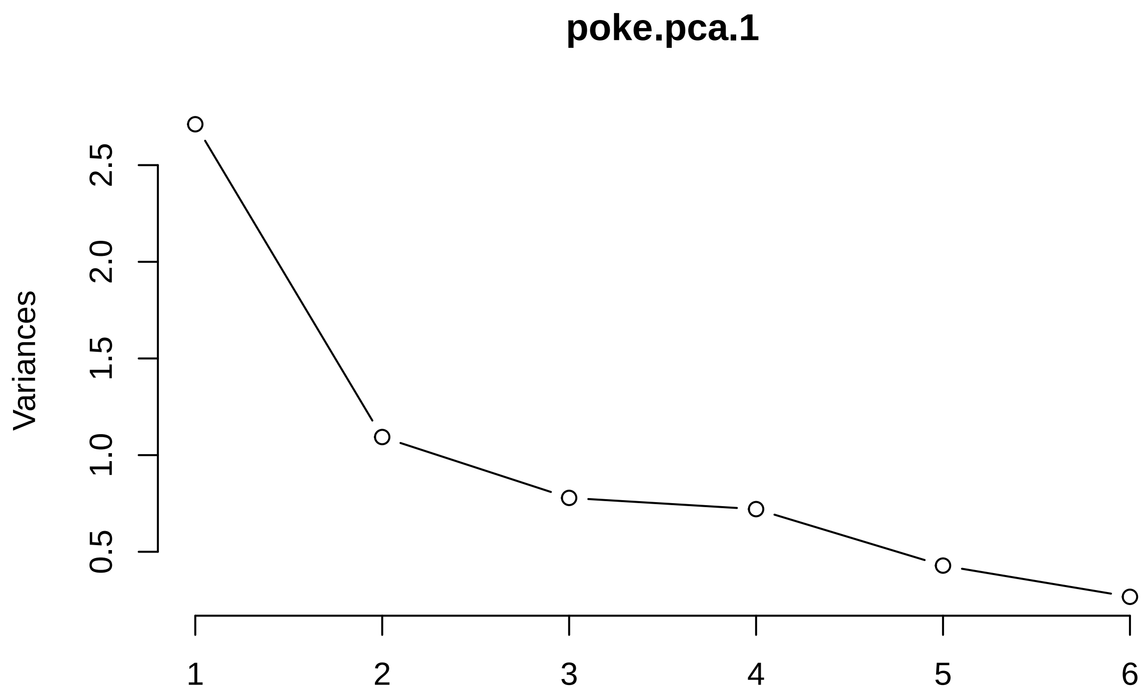
	PC1	PC2	PC3	PC4	PC5	PC6
HP	0.39	0.08	-0.47	0.72	-0.22	0.23
Attack	0.44	-0.01	-0.59	-0.41	0.19	-0.50
Defense	0.36	0.63	0.07	-0.42	-0.06	0.54
Special.Attack	0.46	-0.31	0.31	0.15	0.74	0.20
Special.Defense	0.45	0.24	0.57	0.19	-0.30	-0.55
Speed	0.34	-0.67	0.08	-0.30	-0.53	0.26

The results are simmilar - the PC1 increases with each statistic (the power level). The PC2 is more interesting - It is mostly influenced by the Defense (+), and Speed (-). The Special Attack (-) and Special Defense (-) are also significant, but their effect is about half as strong.

```
summary(poke.pca.1)
```

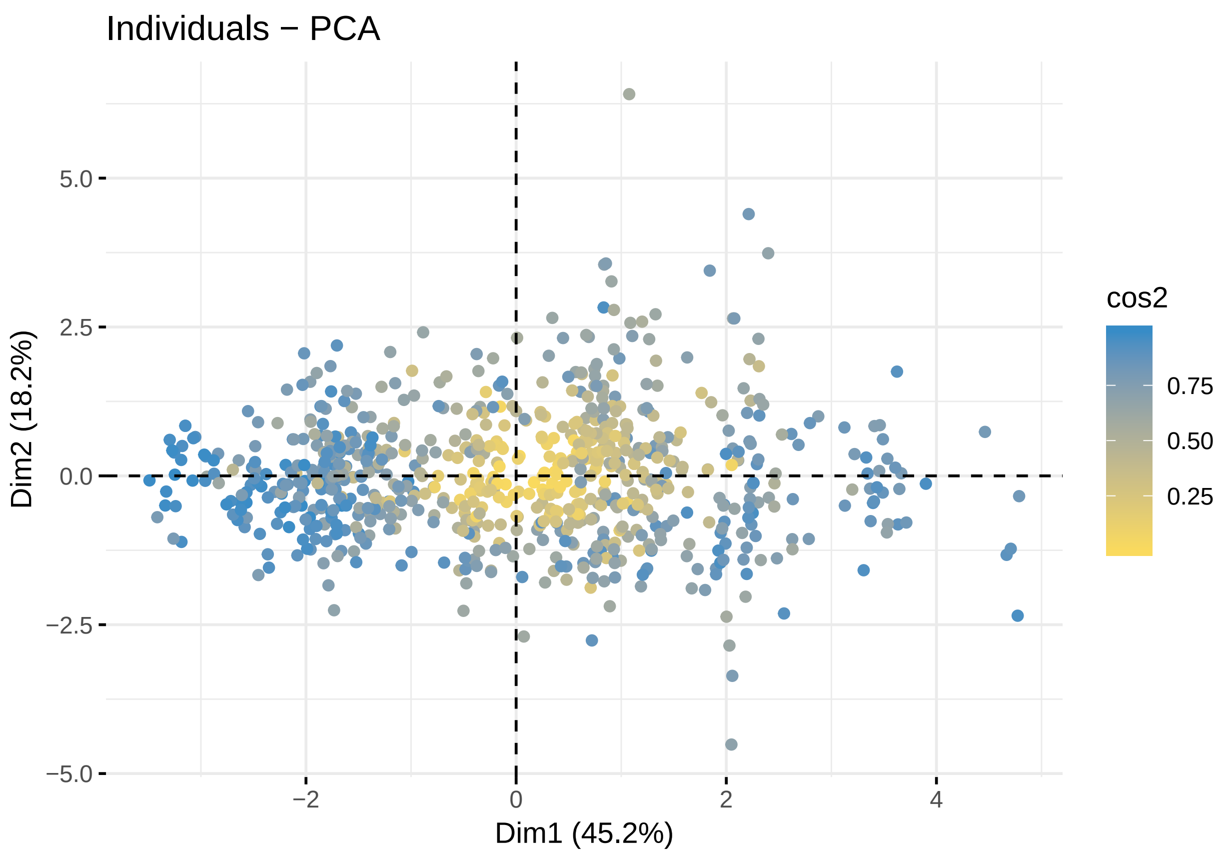
```
Importance of components:
                          PC1    PC2    PC3    PC4     PC5     PC6
Standard deviation     1.6466 1.0457 0.8825 0.8489 0.65463 0.51681
Proportion of Variance 0.4519 0.1822 0.1298 0.1201 0.07142 0.04451
Cumulative Proportion  0.4519 0.6342 0.7640 0.8841 0.95549 1.00000
```

```
plot(poke.pca.1, type = "l")
```



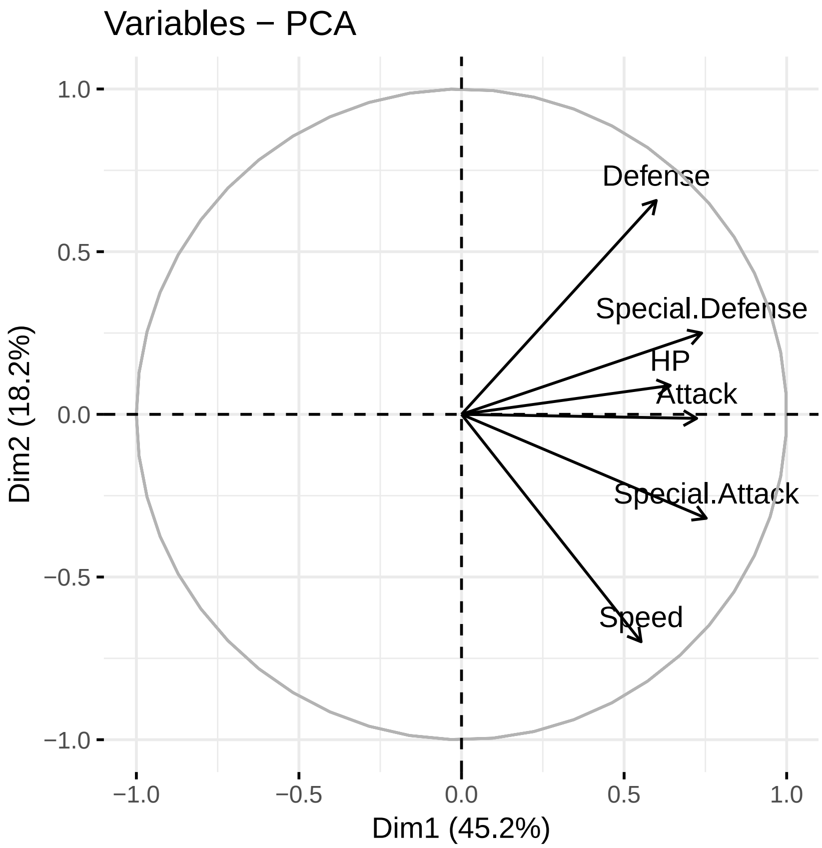
Using 2 variables we can explain 63% of the variance. It is quite good, but could be improved. Adding a third dimension increases the ratio to 77%.

```
fviz_pca_ind(poke.pca.1, col.ind="cos2", geom = "point", gradient.cols = c(cYellow, cBlue))
```



Plotting the obseravtons show us that they are quite similar to MDS.

```
fviz_pca_var(poke.pca.1, col.var="black")
```



We can plot the influence of each variable in a nice graph. As previously discussed, the Defense and Speed have the most significant effects.

PCA using the eigen

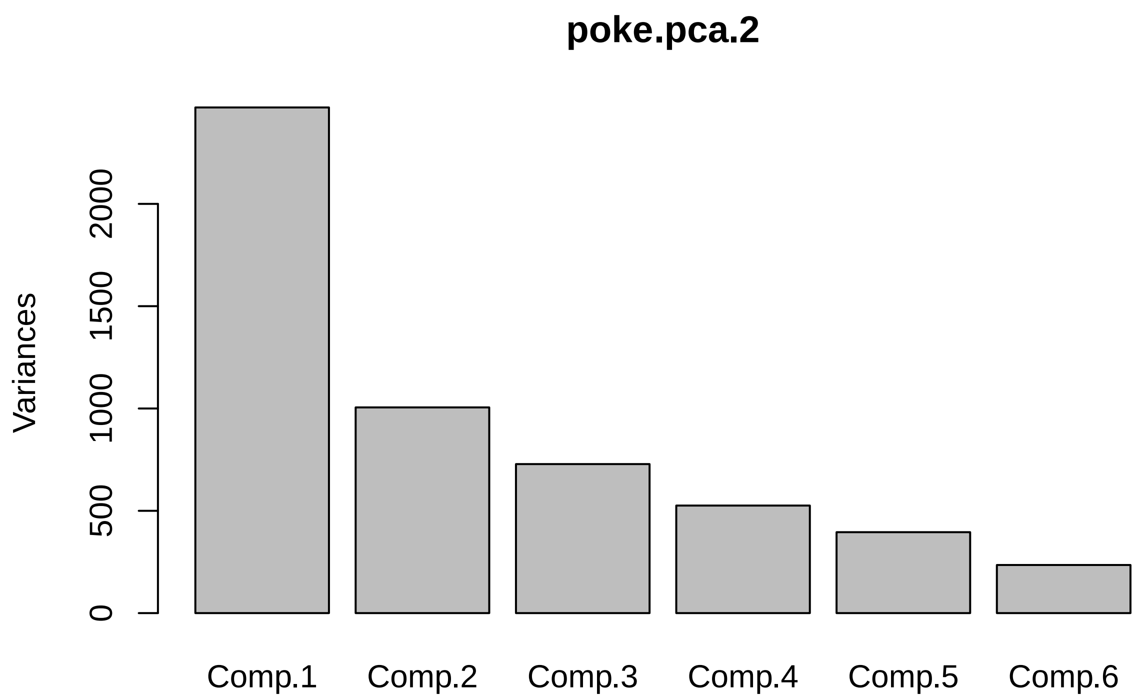
Using the princomp() function we can calculate the PCA using the eigen on the correlation matrix.

```
poke.pca.2<-princomp(poke)
loadings(poke.pca.2)
```

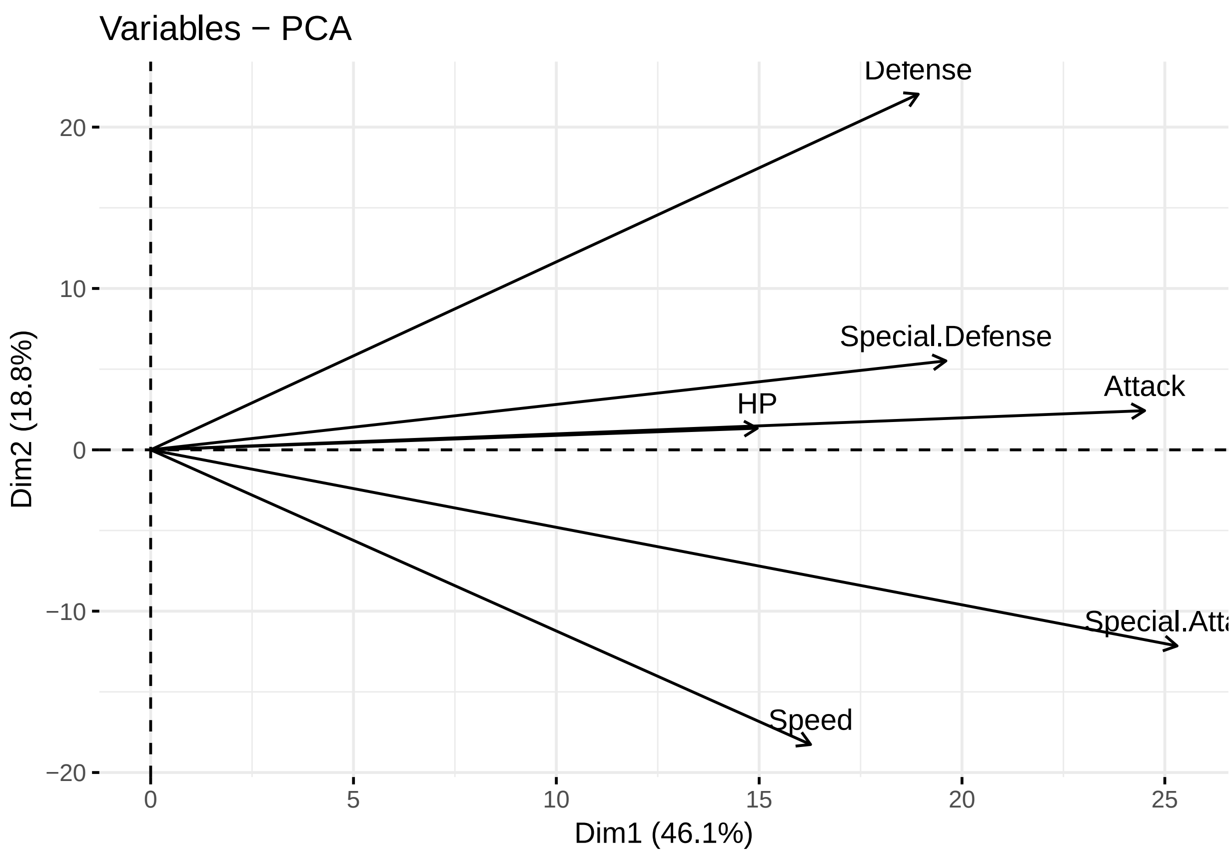
```
Loadings:
                Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6
HP               0.301                0.802  0.387  0.334
Attack           0.493         0.730        -0.193 -0.424
Defense          0.381  0.695        -0.366         0.485
Special.Attack   0.509 -0.383 -0.385  0.101 -0.641  0.158
Special.Defense  0.394  0.174 -0.541         0.375 -0.616
Speed            0.327 -0.576  0.144 -0.459  0.510  0.263

               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
```

As expected, the results are very simmilar.

```r
plot(poke.pca.2)
```




```r
fviz_pca_var(poke.pca.2, col.var="black")
```



Rotated PCA

To have more significant and easier to interpret results, we can use the rotated PCA approach.

```
poke.pca.3 <- principal(poke, nfactors=3, rotate="varimax")
poke.pca.3
```

```
Principal Components Analysis
Call: principal(r = poke, nfactors = 3, rotate = "varimax")
Standardized loadings (pattern matrix) based upon correlation matrix
                  RC1   RC2  RC3   h2   u2 com
HP               0.21  0.15 0.73 0.59 0.41 1.3
Attack           0.14  0.24 0.85 0.80 0.20 1.2
Defense          0.79 -0.16 0.38 0.79 0.21 1.5
Special.Attack   0.38  0.74 0.21 0.74 0.26 1.7
Special.Defense  0.85  0.37 0.08 0.86 0.14 1.4
Speed           -0.08  0.87 0.20 0.80 0.20 1.1

                       RC1  RC2  RC3
SS loadings           1.55 1.55 1.49
Proportion Var        0.26 0.26 0.25
Cumulative Var        0.26 0.52 0.76
Proportion Explained  0.34 0.34 0.32
Cumulative Proportion 0.34 0.68 1.00

Mean item complexity =  1.4
Test of the hypothesis that 3 components are sufficient.
```

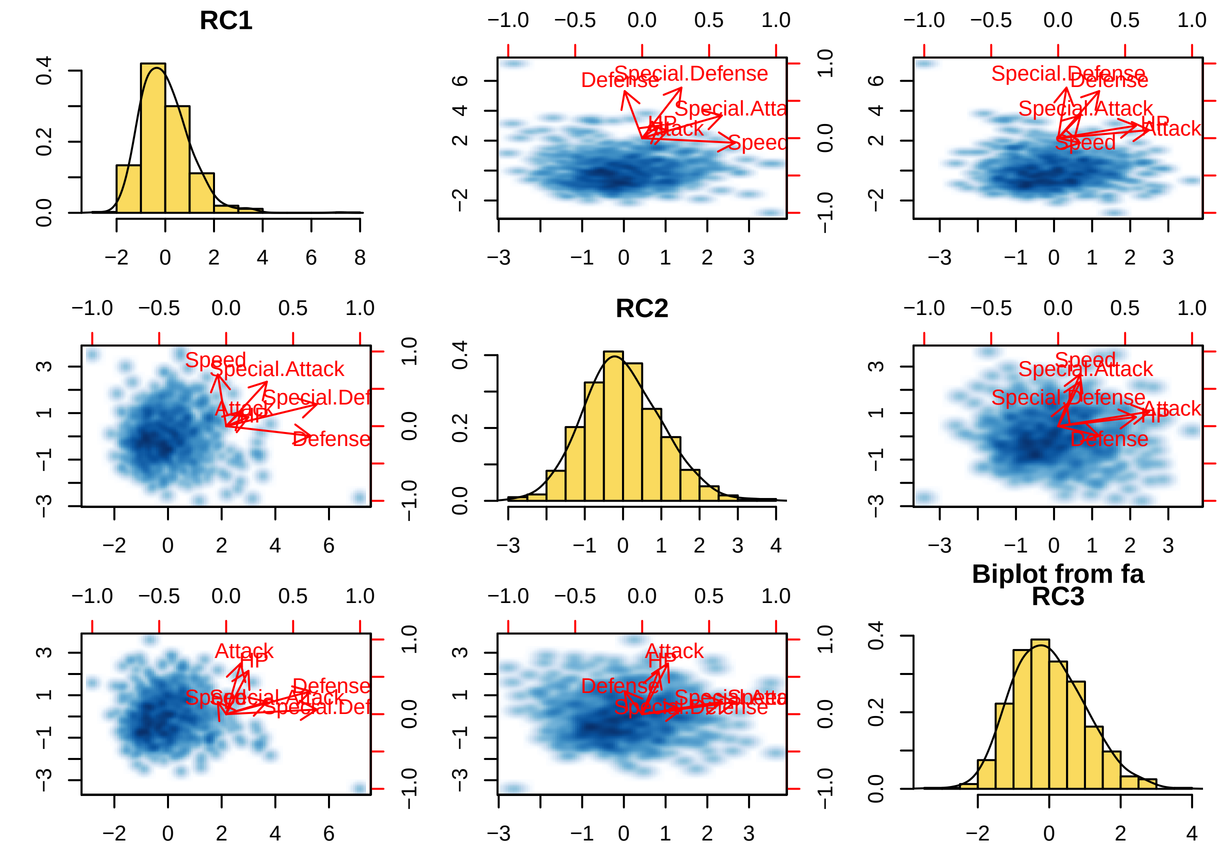
```
The root mean square of the residuals (RMSR) is  0.12
 with the empirical chi square  318.75  with prob <  NA

Fit based upon off diagonal values = 0.9
```

```r
print(loadings(poke.pca.3), digits=2, cutoff=0.4, sort=TRUE)
```

```
Loadings:
                RC1   RC2   RC3  
Defense          0.79            
Special.Defense  0.85            
Special.Attack         0.74      
Speed                  0.87      
HP                           0.73
Attack                       0.85

                RC1  RC2  RC3 
SS loadings    1.55 1.55 1.49
Proportion Var 0.26 0.26 0.25
Cumulative Var 0.26 0.52 0.76
```

These results are much more interesting - without the cut-off we can explain 68% of the variance. Cuting the influence at 0.4 lowers the ratio to 52%, but let's us see the results better. It seems that we have a variable for each of our previously discovered groups - RC1 for Defensive pokemon, RC2 for Agressive, and RC3 for Well-rounded.

```r
biplot(poke.pca.3, hist.col = cYellow, smoother = TRUE)
```



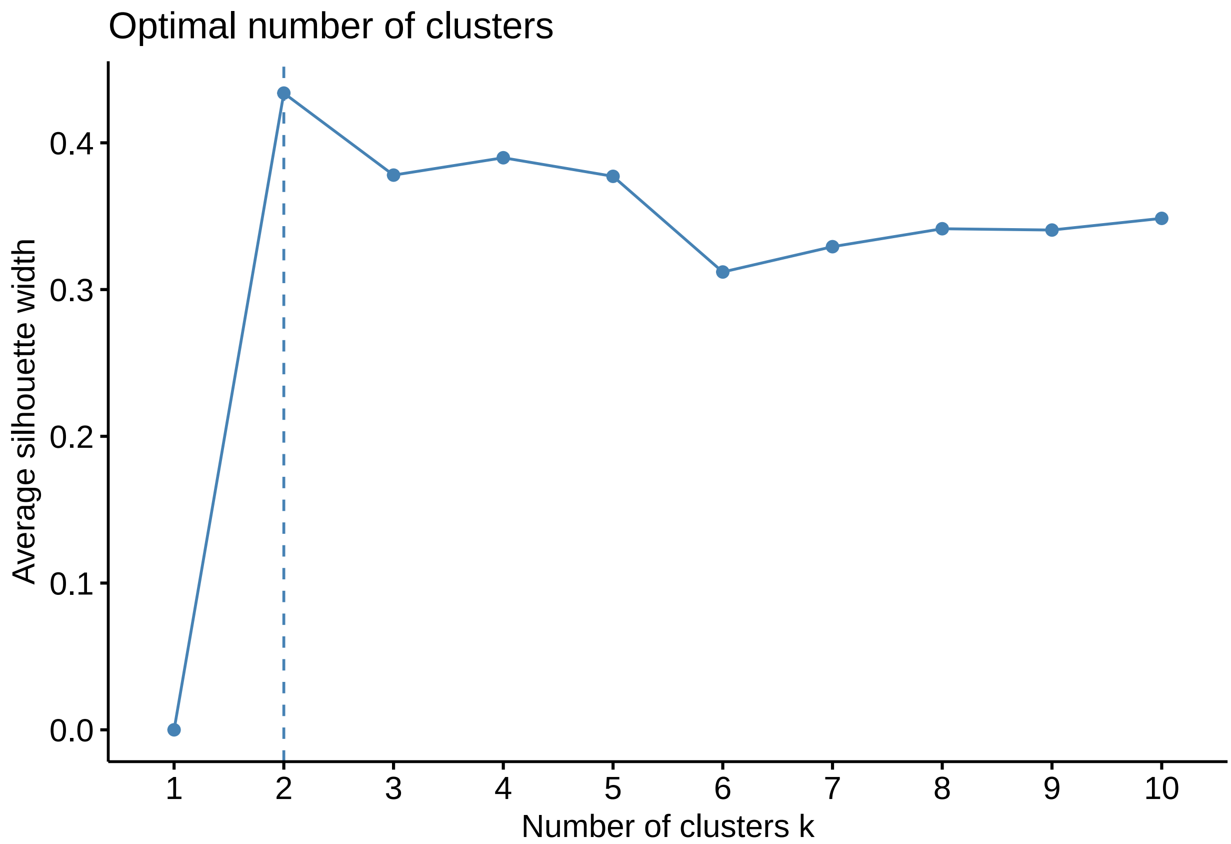
The biplot let's us see the combinations of these 3 variables.

Clustering the results

Optimal number of clusters

After performind the dimensionality reduction in MDS and PCA, we can cluster the results in K-means and PAM.
First step is to find a sutiable number of clusters. For this we can run the fviz_nbclust() function from the factoextra package.

```
# Prepare data
poke.dist<-dist(poke)
poke.mds.1 <- cmdscale(poke.dist, k=2)
poke.mds.1.center <- center_scale(poke.mds.1)
poke.mds.1.center <- poke.mds.1

fviz_nbclust(as.data.frame(poke.mds.1), FUNcluster=pam)
```



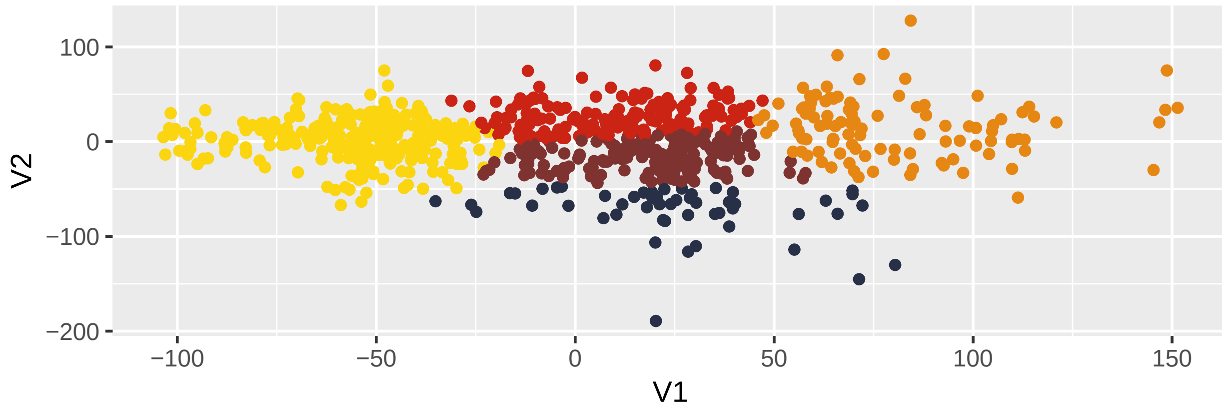
The optimal number of clusters is 2. But, because we want to see the groups identified earlier, we can settle for 5 clusters, which have a bit lower silhouette.

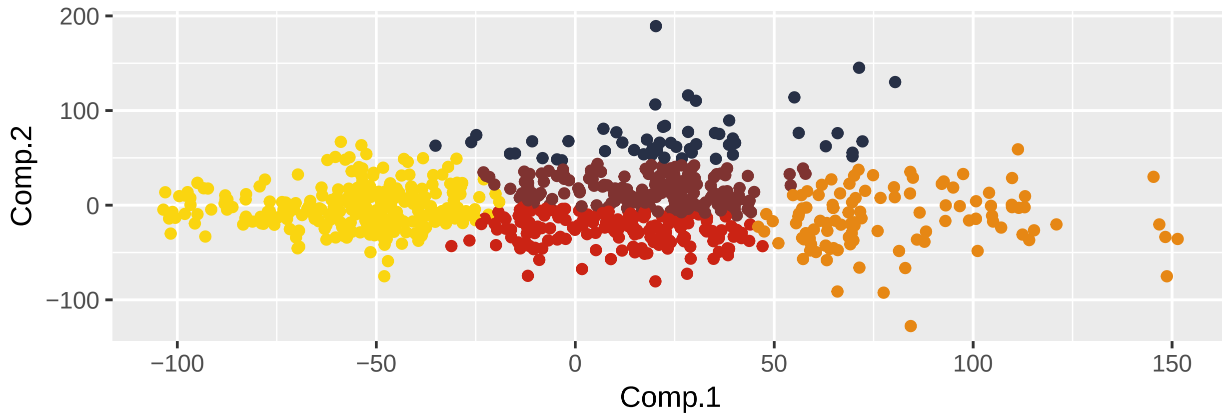
K-means in ClusteR

We can now proceed to clustering using K-means. We'll use the KMenas_rcpp() function from ClusteR library. We'll run this algorithm on the data form MDS and PCA.

```
poke.km <- KMeans_rcpp(poke.mds.1.center, clusters=5, num_init=30, max_iters = 10000)
poke.km.pca <- KMeans_rcpp(poke.pca.2$scores[, 1:2], clusters=5, num_init=30, max_iters = 10000)

x1 <- ggplot(as.data.frame(poke.mds.1.center)) + geom_point(aes(x = V1, y = V2, colour = poke.km$cluste
x2 <- ggplot(as.data.frame(poke.pca.2$scores[, 1:2])) + geom_point(aes(x = Comp.1, y = Comp.2, colour =
```

```
grid.arrange(x1, x2, nrow=2)
```

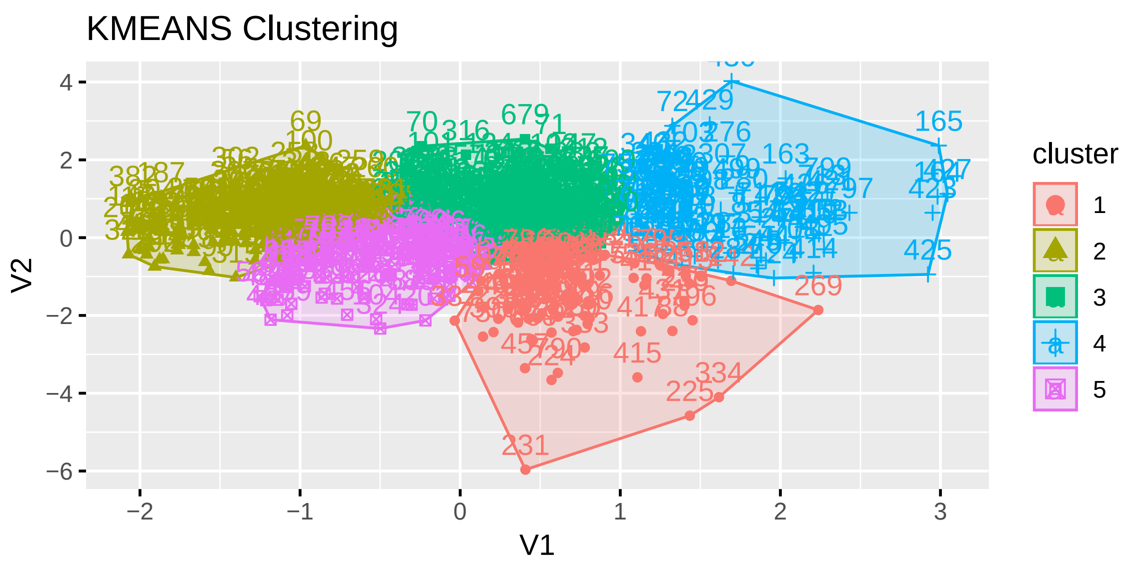




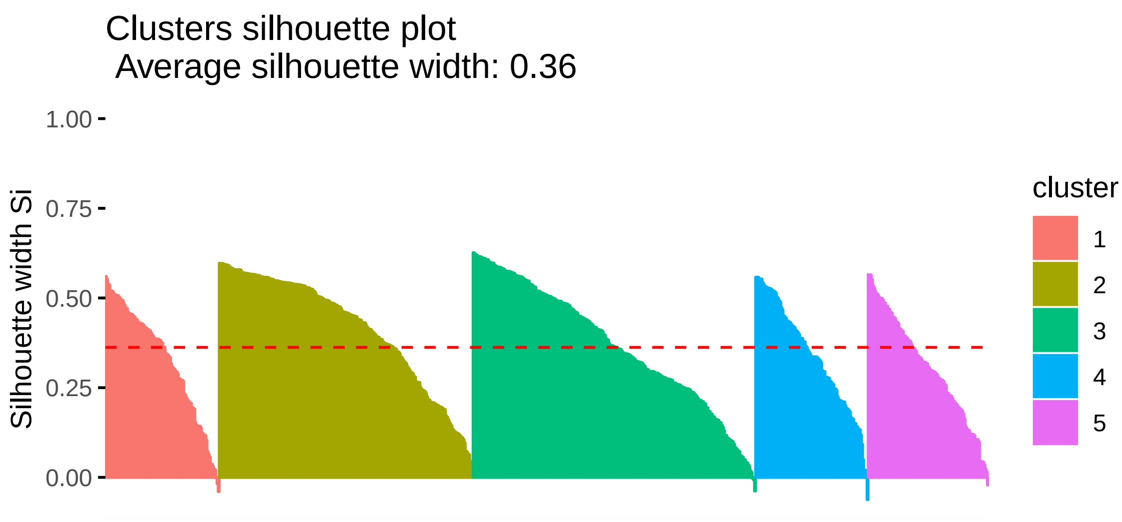
The data was nicely clustered into 5 groups: 'weak', 'average', 'aggresive', 'defensive' and 'legendary' pokemon. The results from MDS and PCA are virtually identical, only the Y axis is reversed.

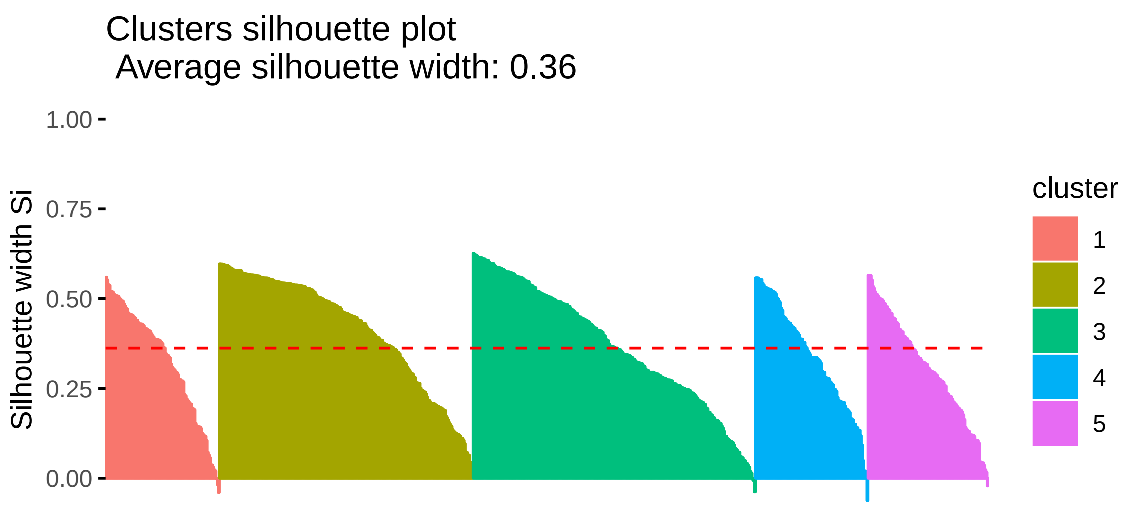
K-means in Factoextra

```
poke.km.2 <- eclust(as.data.frame(poke.mds.1), "kmeans", k = 5)
poke.km.2.pca <- eclust(as.data.frame(poke.pca.2$scores[, 1:2]), "kmeans", k = 5)

fviz_silhouette(poke.km.2)
fviz_silhouette(poke.km.2.pca)
```


KMEANS Clustering
V2
V1
cluster
1
2
3
4
5

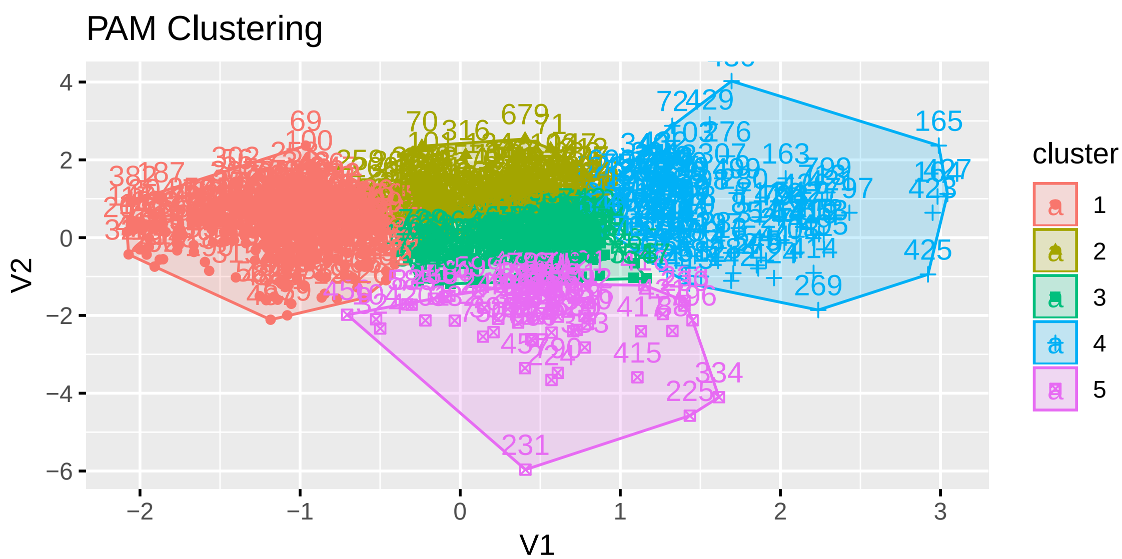

KMEANS Clustering
Comp.2
Comp.1
cluster
1
2
3
4
5

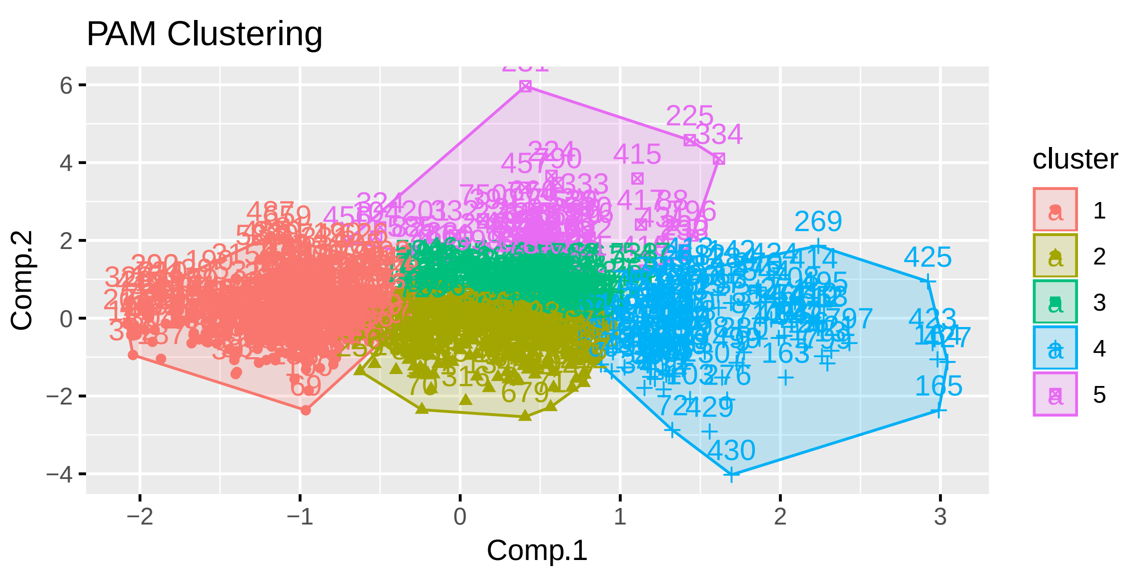




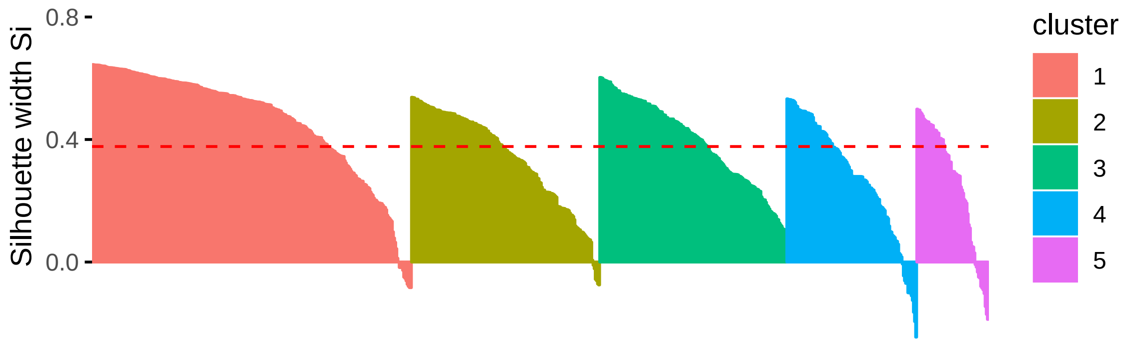
PAM

```
poke.pam <- eclust(as.data.frame(poke.mds.1), "pam", k = 5)
poke.pam.2 <- eclust(as.data.frame(poke.pca.2$scores[, 1:2]), "pam", k = 5)
fviz_silhouette(poke.pam)
fviz_silhouette(poke.pam.2)
```

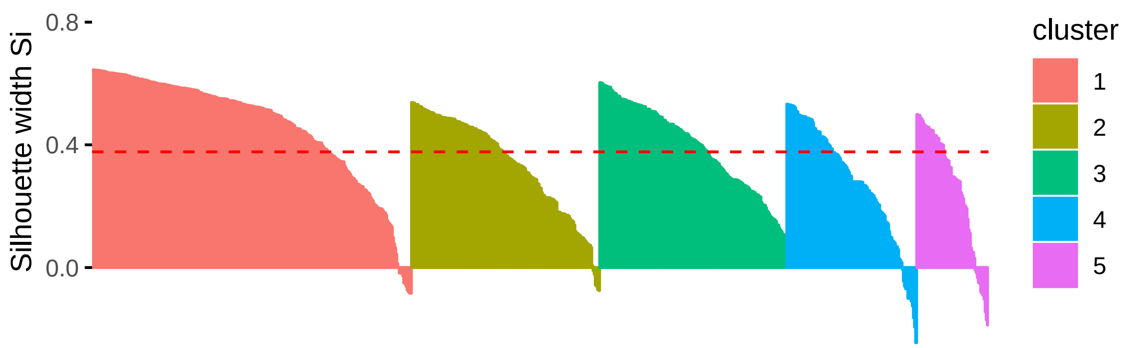

PAM Clustering
V1
V2
cluster
1
2
3
4
5


PAM Clustering
Comp.1
Comp.2
cluster
1
2
3
4
5

Clusters silhouette plot
 Average silhouette width: 0.38



Clusters silhouette plot
 Average silhouette width: 0.38



Conclusions