Statistics for Data Science - UC3M Master

Multivariate Analysis - First Assignment

Ricardo Hortelano
Javier Martinez
Santiago Raposo

Part 1 - Pre-process the data set for practical analysis.

Missing Values

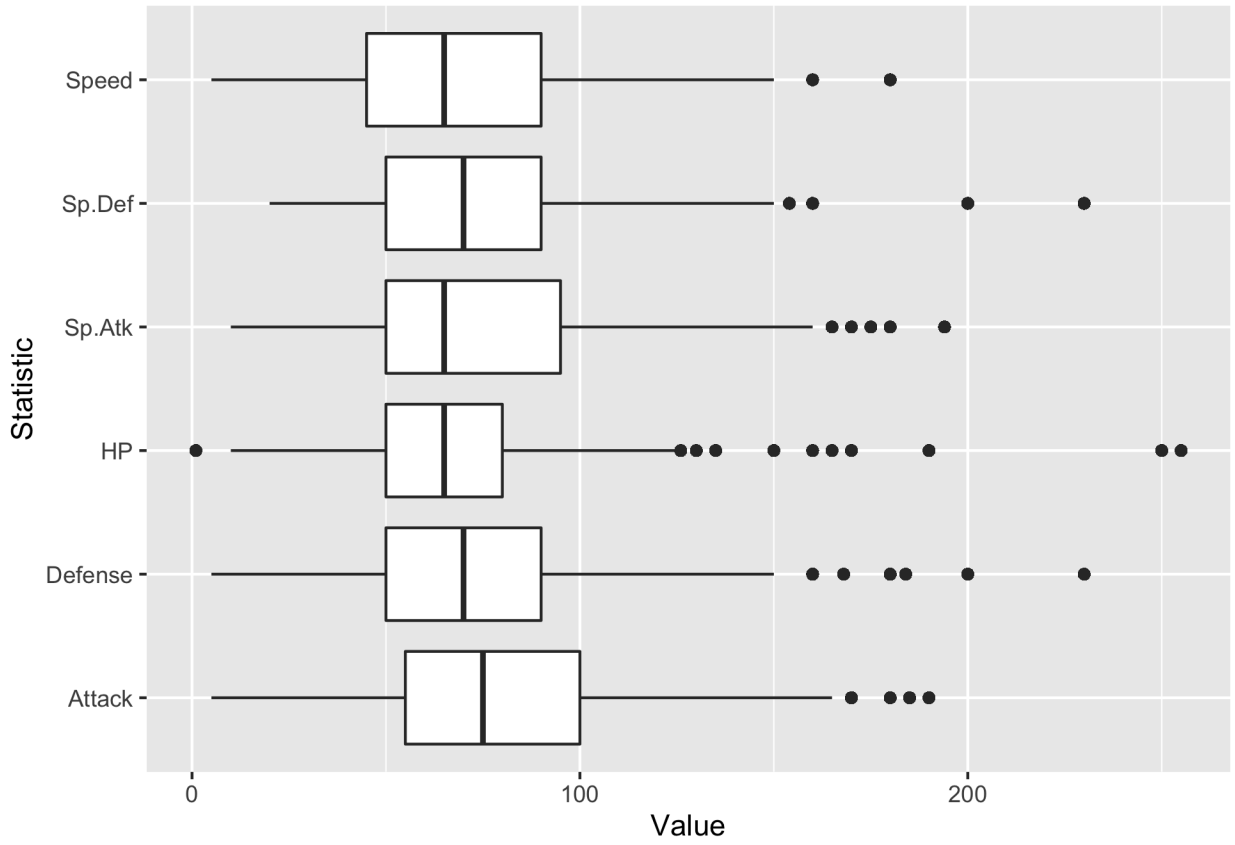
The only columns with missing values is the `poke1_Type.2` and `poke2_Type.2`. These missing values are not real missing values. An empty value in this field means that pokemon has not a second type. Knowing this, we treat that empty value as a new category.

Regarding the rest of the dataset, there are no missing values.

Distribution of Statistics and Outliers

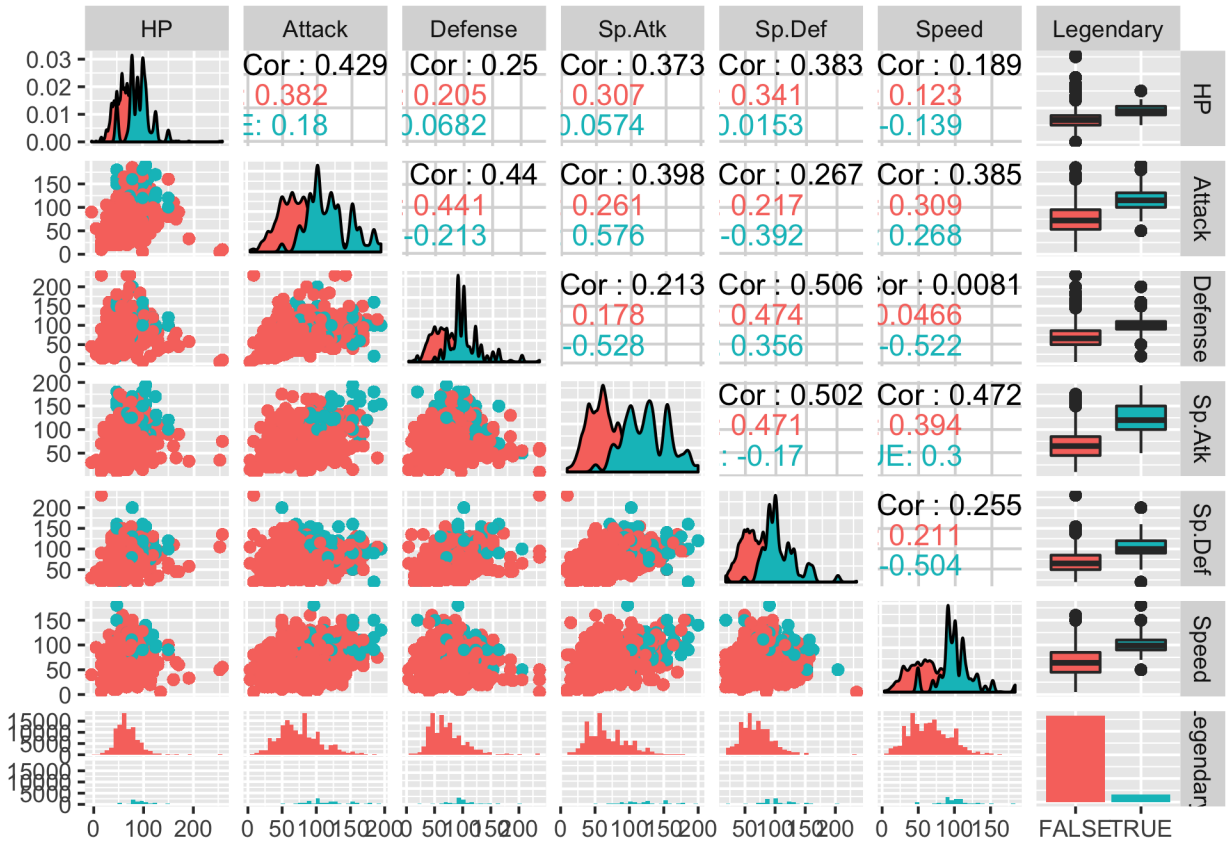
For some of our plots, we are only interested in the distribution of the variables of the pokemons. Since there are two pokemons per row of our dataset, we have joined the distributions of each of the variables in the dataset `poke_long`. After restructuring the dataset in this way, we can see a box plot of the variables so that we can get a rough idea of their distribution.

```
poke_long %>%
  select_if(is.numeric) %>%
  pivot_longer(
    everything(),
    names_to = "Statistic",
    values_to = "Value"
  ) %>%
  ggplot(aes(x = Statistic, y = Value))+
  geom_boxplot()+
  coord_flip()
```



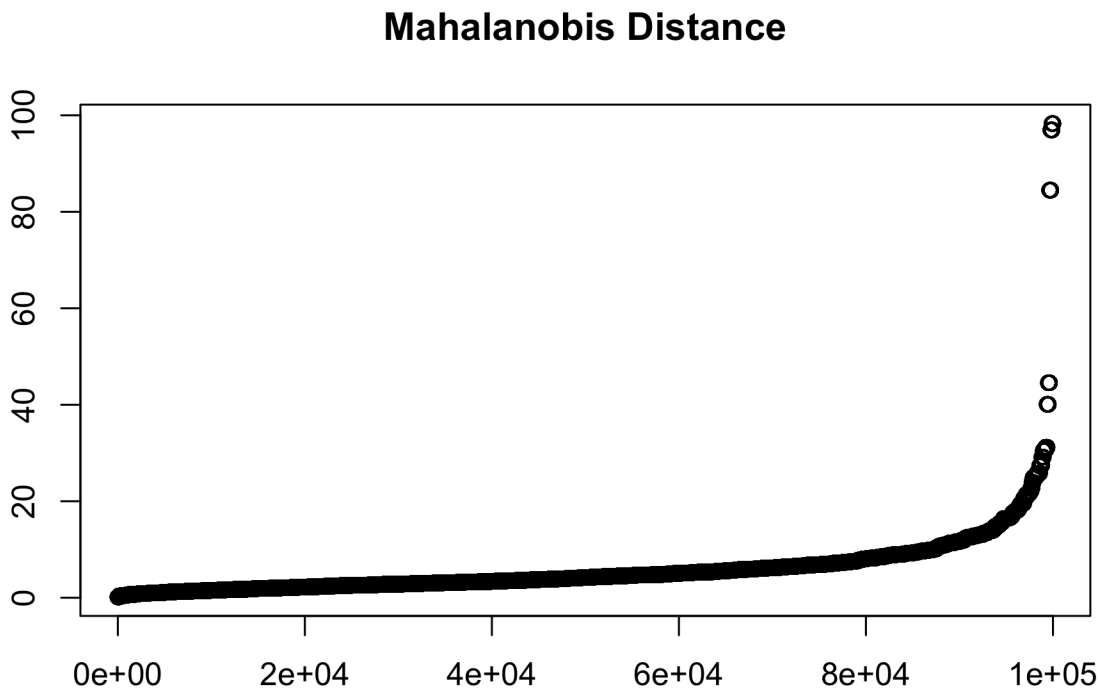
As we have some knowledge of the structure of our dataset, we might wonder if pokemons which are labeled as `Legendary` are outliers in the distribution of their statistics. We can check this using a parallel coordinates plot.

```
numeric_vars <- poke_long %>%
  select_if(~is.numeric(.) || is.logical(.))
# Take a sample of size = 1000 for plots
numeric_sample <- numeric_vars[sample(nrow(numeric_vars)), ][1:1e3,]
ggpairs(numeric_vars, aes(color = Legendary))
```



In order to check if there are any outliers in our variables, we employ the False Discovery Rate method. First, we calculate the Mahalanobis distance for the numeric variables in our dataset.

```
n <- nrow(poke)
p <- ncol(poke)
poke_x <- poke_long %>%
  select_if(is.numeric)
mah_poke <- mahalanobis(poke_x, colMeans(poke_x), cov(poke_x))
plot(sort(mah_poke), main = "Mahalanobis Distance", xlab = "", ylab = "")
```



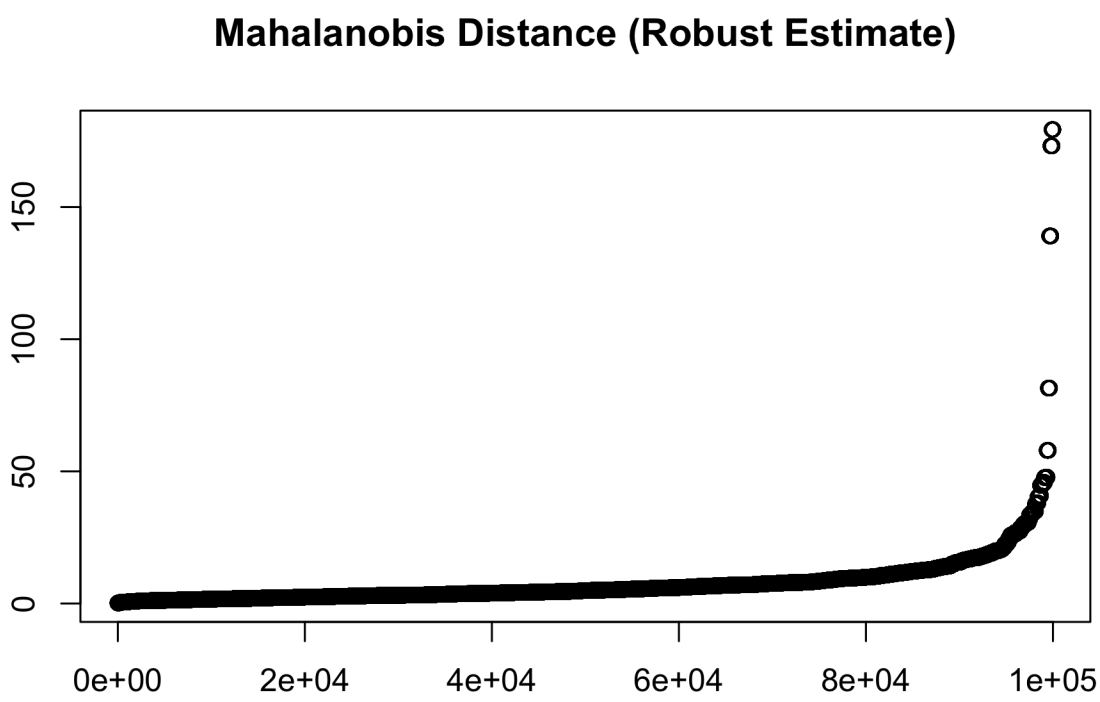
Now we can compute the outliers using the χ^2 distribution:

```
p_values <- 1 - pchisq(mah_poke, p)
sort_index <- order(p_values)
outliers_index <- sort_index[which(p_values[sort_index] < ((1:n)/n*0.01))]

head(poke_long[outliers_index, ])
```

```
# A tibble: 6 x 13
  fight Name  Type1 Type2    HP Attack Defense Sp.Atk Sp.Def Speed
  <chr> <chr> <chr> <chr> <dbl>  <dbl>   <dbl>  <dbl>  <dbl> <dbl>
1 poke1 Chan~ Norm~ <NA>    250      5       5     35    105    50
2 poke1 Chan~ Norm~ <NA>    250      5       5     35    105    50
3 poke2 Chan~ Norm~ <NA>    250      5       5     35    105    50
4 poke2 Chan~ Norm~ <NA>    250      5       5     35    105    50
5 poke1 Chan~ Norm~ <NA>    250      5       5     35    105    50
6 poke2 Chan~ Norm~ <NA>    250      5       5     35    105    50
# ... with 3 more variables: Generation <fct>, Legendary <lgl>,
#   number <fct>
```

As there are indeed some outliers, we can try to compute the Mahalanobis distance again using the robust estimators for the sample covariance matrix.

```
mcd_poke <- covMcd(poke_x, alpha = .9)
mah_robust <- mahalanobis(poke_x, mcd_poke$center, mcd_poke$cov)

plot(sort(mah_robust), main = "Mahalanobis Distance (Robust Estimate)", xlab = "", ylab= "")
```



And for the calculation of p-values and detection of outliers:

```
p_values_robust <- 1 - pchisq(mah_robust, p)
sort_index_robust <- order(p_values_robust)
outliers_index_robust <-
  sort_index_robust[which(p_values_robust[sort_index_robust] < ((1:n)/n*0.01))]

head(poke_long[outliers_index, ])
```

```
# A tibble: 6 x 13
  fight Name  Type1 Type2    HP Attack Defense Sp.Atk Sp.Def Speed
  <chr> <chr> <chr> <chr> <dbl>  <dbl>   <dbl>  <dbl>  <dbl> <dbl>
1 poke1 Chan~ Norm~ <NA>    250      5       5     35    105    50
2 poke1 Chan~ Norm~ <NA>    250      5       5     35    105    50
3 poke2 Chan~ Norm~ <NA>    250      5       5     35    105    50
4 poke2 Chan~ Norm~ <NA>    250      5       5     35    105    50
5 poke1 Chan~ Norm~ <NA>    250      5       5     35    105    50
6 poke2 Chan~ Norm~ <NA>    250      5       5     35    105    50
# ... with 3 more variables: Generation <fct>, Legendary <lgl>,
#   number <fct>
```

```
# Outlier Pokemon
unique(poke_long[outliers_index, ]$Name)
```

```
[1] "Chansey" "Blissey" "Shuckle"
```

But indeed we get the same 3 pokemon showing up as outliers: `Chansey`, `Blissey` and `Shuckle`. The reason these 3 show up as outliers is their extreme values of `HP` (for `Chansey` and `Blissey`) and `Defense` and `Sp.Def`

(for `Shuckle`).

We will not be removing these data points, as they are completely fine pokemon, but we will have to make sure that we are using robust estimators in our analyses. This also does reinforce the fact that legendary pokemon are not outliers, as neither `Chansey`, `Blissey` nor `Shuckle` is legendary.

Part 2 - Perform a descriptive analysis of the pre-processed data set and obtain interesting conclusions from the analysis.

[Insert conclusions here]