Pokemon Data Analysis

Section 1: Introduction

A Pokémon (Japanese: ポケモン Pokemon), shortened from Pocket Monster (Japanese: ポケットモンスター Poketto Monsutaa), is any of the 807 documented species of the enormous collectible creatures inhabiting the fictional Pokémon World. Each Pokemon is innately connected to element-based supernatural powers. This project is dedicated to discovering factors that prevail the performance of Pokemon in battles. The dataset used for this project is publicly posted on kaggle.com which captures complete information of 721 Pokemon from 6 generations. To be more specific, the column Total explains the performance of a particular Pokemon, which is the sum of HP, attack, defense, special attack, special defense, and speed. The relationship between the Pokemon's performance and the rest of the variables will be discussed in this project. In this project, the programming language R is used to accommodate the statistical analysis of the performance of Pokemon.

Section 2: Dataset and Preprocess dataset

The raw dataset is provided by Kaggle.com. Each row represents information of each Pokemon. And each column represents a different feature of a Pokemon. In total, there are 721 rows corresponding to 721 different Pokemon and there are 23 columns corresponding to 23 features of a Pokemon. The raw dataset is shown below:

```
library(tidyverse)

## ── Attaching packages ──────────────────────────────────────── tidyver
se 1.2.1 ──

## ✔ ggplot2 3.1.0     ✔ purrr   0.2.5
## ✔ tibble  1.4.2     ✔ dplyr   0.7.8
## ✔ tidyr   0.8.2     ✔ stringr 1.3.1
## ✔ readr   1.1.1     ✔ forcats 0.3.0

## ── Conflicts ───────────────────────────────────────── tidyverse_con
flicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()

library(smmr)
library(ggplot2)
library(broom)
library(MASS)
```

```
##
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':
##
##     select

library(leaps)
pokemon = read_csv("/cloud/project/pokemon2.csv")

## Parsed with column specification:
## cols(
##   .default = col_character(),
##   Number = col_integer(),
##   Total = col_integer(),
##   HP = col_integer(),
##   Attack = col_integer(),
##   Defense = col_integer(),
##   Sp_Atk = col_integer(),
##   Sp_Def = col_integer(),
##   Speed = col_integer(),
##   Generation = col_integer(),
##   Pr_Male = col_double(),
##   Height_m = col_double(),
##   Weight_kg = col_double(),
##   Catch_Rate = col_integer()
## )

## See spec(...) for full column specifications.

pokemon

## # A tibble: 721 x 23
##    Number Name  Type_1 Type_2 Total    HP Attack Defense Sp_Atk Sp_Def
##     <int> <chr> <chr>  <chr>  <int> <int>  <int>   <int>  <int>  <int>
##  1      1 Bulb… Grass  Poison   318    45     49      49     65     65
##  2      2 Ivys… Grass  Poison   405    60     62      63     80     80
##  3      3 Venu… Grass  Poison   525    80     82      83    100    100
##  4      4 Char… Fire   <NA>     309    39     52      43     60     50
##  5      5 Char… Fire   <NA>     405    58     64      58     80     65
##  6      6 Char… Fire   Flying   534    78     84      78    109     85
##  7      7 Squi… Water  <NA>     314    44     48      65     50     64
##  8      8 Wart… Water  <NA>     405    59     63      80     65     80
##  9      9 Blas… Water  <NA>     530    79     83     100     85    105
## 10     10 Cate… Bug    <NA>     195    45     30      35     20     20
## # ... with 711 more rows, and 13 more variables: Speed <int>,
## #   Generation <int>, isLegendary <chr>, Color <chr>, hasGender <chr>,
## #   Pr_Male <dbl>, Egg_Group_1 <chr>, Egg_Group_2 <chr>,
## #   hasMegaEvolution <chr>, Height_m <dbl>, Weight_kg <dbl>,
## #   Catch_Rate <int>, Body_Style <chr>
```

For this study, the missing information is Type_2, Egg_Group_2 and Pr_Male (probability of a Pokemon being a male). All three columns are replaced by a boolean expression. If the Pokemon has a second type, second egg group, or probability of being a male, then it appears true. If they are not specified, then they appear false. Therefore three new columns are included in the updated_pokemon data frame, which are named Type_2_New, Egg_Group_2_New, Pr_Male_New. The updated dataset now have 26 columns with the additional three new columns. The mutate function is used to add a new column into the original dataset pokemon. The updated dataset is called updated_pokemon which is displayed below:

```
updated_pokemon = mutate(pokemon, Type_2_New = ifelse(is.na(Type_2), "False",
 "True"))
updated_pokemon = mutate(updated_pokemon, Egg_Group_2_New = ifelse(is.na(Egg_
Group_2), "False", "True"))
updated_pokemon = mutate(updated_pokemon, Pr_Male_New = ifelse(is.na(Pr_Male)
, "False", "True"))
updated_pokemon
```

```
## # A tibble: 721 x 26
##    Number Name  Type_1 Type_2 Total    HP Attack Defense Sp_Atk Sp_Def
##     <int> <chr> <chr>  <chr>  <int> <int>  <int>   <int>  <int>  <int>
##  1      1 Bulb… Grass  Poison   318    45     49      49     65     65
##  2      2 Ivys… Grass  Poison   405    60     62      63     80     80
##  3      3 Venu… Grass  Poison   525    80     82      83    100    100
##  4      4 Char… Fire   <NA>     309    39     52      43     60     50
##  5      5 Char… Fire   <NA>     405    58     64      58     80     65
##  6      6 Char… Fire   Flying   534    78     84      78    109     85
##  7      7 Squi… Water  <NA>     314    44     48      65     50     64
##  8      8 Wart… Water  <NA>     405    59     63      80     65     80
##  9      9 Blas… Water  <NA>     530    79     83     100     85    105
## 10     10 Cate… Bug    <NA>     195    45     30      35     20     20
## # ... with 711 more rows, and 16 more variables: Speed <int>,
## #   Generation <int>, isLegendary <chr>, Color <chr>, hasGender <chr>,
## #   Pr_Male <dbl>, Egg_Group_1 <chr>, Egg_Group_2 <chr>,
## #   hasMegaEvolution <chr>, Height_m <dbl>, Weight_kg <dbl>,
## #   Catch_Rate <int>, Body_Style <chr>, Type_2_New <chr>,
## #   Egg_Group_2_New <chr>, Pr_Male_New <chr>
```

```
summary(updated_pokemon)
```

```
##      Number          Name              Type_1             Type_2
##  Min.   :  1   Length:721         Length:721         Length:721
##  1st Qu.:181   Class :character   Class :character   Class :character
##  Median :361   Mode  :character   Mode  :character   Mode  :character
##  Mean   :361
##  3rd Qu.:541
##  Max.   :721
##
##      Total             HP              Attack          Defense
##  Min.   :180.0   Min.   :  1.00   Min.   :  5.00   Min.   :  5.00
```

```
##  1st Qu.:320.0   1st Qu.: 50.00   1st Qu.: 53.00   1st Qu.: 50.00
##  Median :424.0   Median : 65.00   Median : 74.00   Median : 65.00
##  Mean   :417.9   Mean   : 68.38   Mean   : 75.01   Mean   : 70.81
##  3rd Qu.:499.0   3rd Qu.: 80.00   3rd Qu.: 95.00   3rd Qu.: 85.00
##  Max.   :720.0   Max.   :255.00   Max.   :165.00   Max.   :230.00
##
##      Sp_Atk           Sp_Def           Speed          Generation
##  Min.   : 10.00   Min.   : 20.00   Min.   :  5.00   Min.   :1.000
##  1st Qu.: 45.00   1st Qu.: 50.00   1st Qu.: 45.00   1st Qu.:2.000
##  Median : 65.00   Median : 65.00   Median : 65.00   Median :3.000
##  Mean   : 68.74   Mean   : 69.29   Mean   : 65.71   Mean   :3.323
##  3rd Qu.: 90.00   3rd Qu.: 85.00   3rd Qu.: 85.00   3rd Qu.:5.000
##  Max.   :154.00   Max.   :230.00   Max.   :160.00   Max.   :6.000
##
##  isLegendary           Color             hasGender            Pr_Male
##  Length:721         Length:721         Length:721         Min.   :0.0000
##  Class :character   Class :character   Class :character   1st Qu.:0.5000
##  Mode  :character   Mode  :character   Mode  :character   Median :0.5000
##                                                           Mean   :0.5534
##                                                           3rd Qu.:0.5000
##                                                           Max.   :1.0000
##                                                           NA's   :77
##  Egg_Group_1        Egg_Group_2        hasMegaEvolution      Height_m
##  Length:721         Length:721         Length:721         Min.   : 0.100
##  Class :character   Class :character   Class :character   1st Qu.: 0.610
##  Mode  :character   Mode  :character   Mode  :character   Median : 0.990
##                                                           Mean   : 1.145
##                                                           3rd Qu.: 1.400
##                                                           Max.   :14.500
##
##    Weight_kg        Catch_Rate     Body_Style         Type_2_New
##  Min.   :  0.10   Min.   :  3.0   Length:721         Length:721
##  1st Qu.:  9.40   1st Qu.: 45.0   Class :character   Class :character
##  Median : 28.00   Median : 65.0   Mode  :character   Mode  :character
##  Mean   : 56.77   Mean   :100.2
##  3rd Qu.: 61.00   3rd Qu.:180.0
##  Max.   :950.00   Max.   :255.0
##
##  Egg_Group_2_New    Pr_Male_New
##  Length:721         Length:721
##  Class :character   Class :character
##  Mode  :character   Mode  :character
##
##
##
##
```

We use “summary” function to achieve a general summary of the data, which includes the minimum, maximum, mean, median, Q1 and Q3 of the quantitative variables. In the next few sessions, statistical methods will be implemented to study how these factors affect the total.

Analysis: Effect of Other factors on Pokemon's total performance

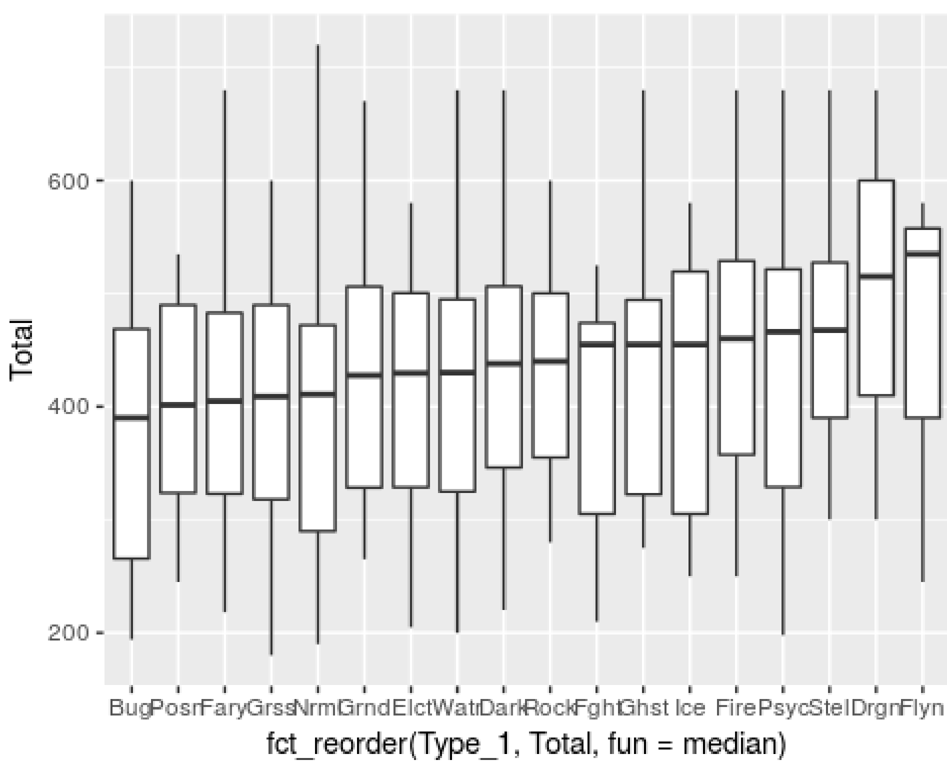
Section 3: Type 1 and 2

In this section, we will discuss the influence on the Pokemon's performance in terms of Pokemon's types.

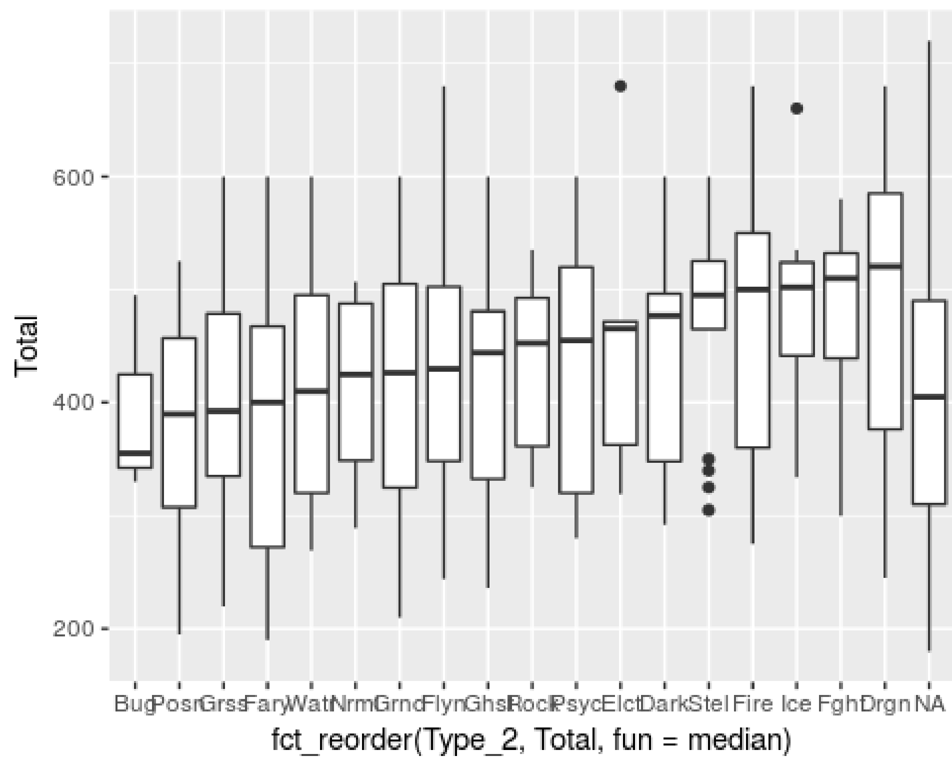
```
updated_pokemon %>% count(Type_2_New)

## # A tibble: 2 x 2
##   Type_2_New     n
##   <chr>      <int>
## 1 False        371
## 2 True         350
```

This table indicates there are 371 Pokemon do not have a second type and 350 Pokemon have a second type.

```
type1_plot = ggplot(updated_pokemon, aes(x=fct_reorder(Type_1, Total, fun=med
ian), y=Total)) + geom_boxplot()
type1_plot + scale_x_discrete(labels = abbreviate)
```

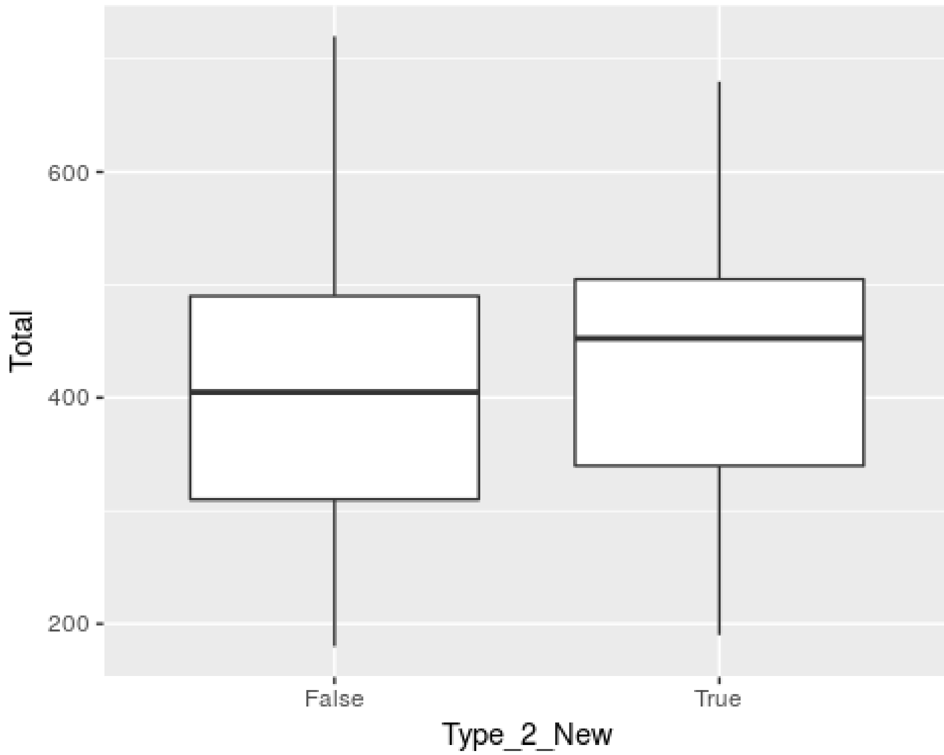



```
type2_plot = ggplot(updated_pokemon, aes(x=fct_reorder(Type_2, Total, fun=med
ian), y=Total)) + geom_boxplot()
type2_plot + scale_x_discrete(labels = abbreviate)
```



Both boxplots above shows the distribution of total performance within each type in type 1 and type 2. The types with low total performance in both type 1 and 2 are bug, poison, grass and fairy. Dragon type is the only type with high total performance in both type 1 and 2. Since the dataset is relatively large, the distribution of type 1 and type 2 with total performance can be approximately normal by the Central Limit Theorem regardless of the skewness and outliers.

```
ggplot(updated_pokemon, aes(x=Type_2_New, y=Total)) + geom_boxplot()
```



The distribution of the Pokemon with and without a second type vs the total performance is approximately normal from the above boxplot. It is appropriate to perform a pooled t-test on this distribution because they share a similar spread.

```
t.test(Total~Type_2_New, data=updated_pokemon, var.equal=T)

##
##  Two Sample t-test
##
## data:  Total by Type_2_New
## t = -3.5138, df = 719, p-value = 0.0004694
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -44.40801 -12.57162
## sample estimates:
## mean in group False  mean in group True
##            404.1159            432.6057
```

By the two sample pooled t-test, we are able to reject the null hypothesis of the mean of total performance between Pokemon with and without type two are the same. Since the p-value of 0.0004694 is less than the alpha of 0.05 and therefore we can conclude that the Pokemon with and without type two are different in mean with each other in terms of total performance.

Section 4: Generation

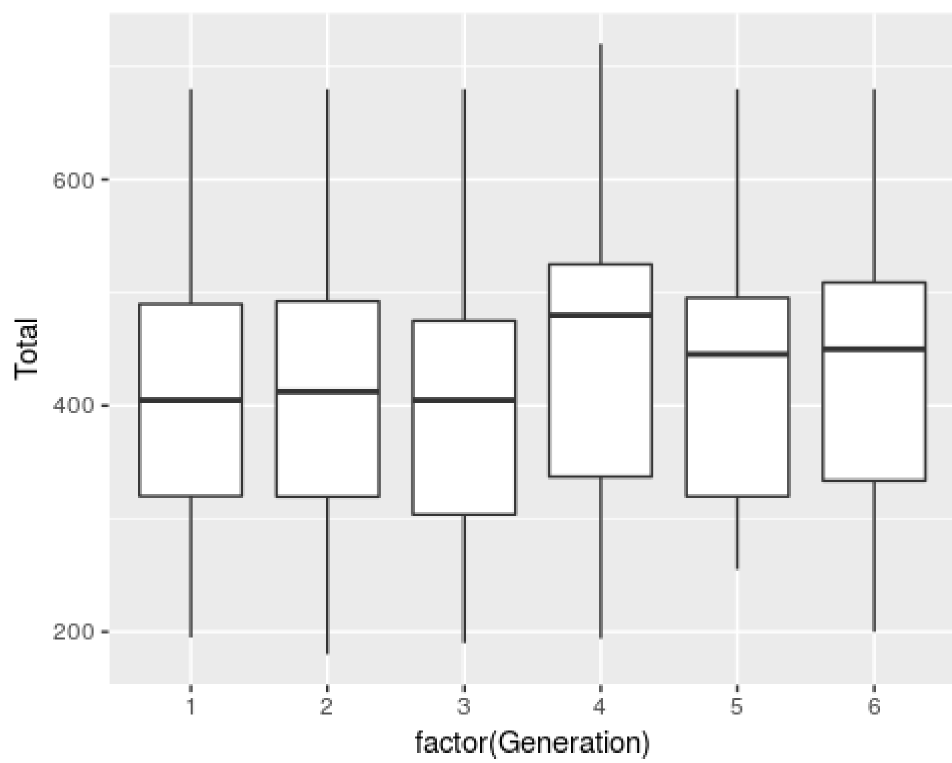
This section will study the relationship between the generation of the Pokemon and Pokemon's performance.

```
updated_pokemon %>% count(Generation)
```

```
## # A tibble: 6 x 2
##   Generation     n
##        <int> <int>
## 1          1   151
## 2          2   100
## 3          3   135
## 4          4   107
## 5          5   156
## 6          6    72
```

In total, there are six generations of Pokemon. The number of pokemon are 151, 100, 135, 107, 156 and 72 respectively for generation one to six.

```
ggplot(updated_pokemon, aes(x=factor(Generation), y=Total)) + geom_boxplot()
```



The boxplots of Pokemon's generation and Total reveal that the dataset is an approximately normal distribution and equal spread within each group. Since the dataset qualifies the assumption of the Analysis of Variance(AVONA), we then examine the dataset with ANOVA test and Tukey's method.

```
generation.1=aov(Total~factor(Generation), data=updated_pokemon)
summary(generation.1)
```

```
##                      Df  Sum Sq Mean Sq F value Pr(>F)
## factor(Generation)    5  166711   33342   2.807 0.0161 *
## Residuals           715 8492096   11877
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We reject the null hypothesis of all generations having the same mean because the p-value of 0.0161 is less than Alpha of 0.05. Therefore, we conclude there are differences on the mean of Total between the generation.

```
TukeyHSD(generation.1)
```

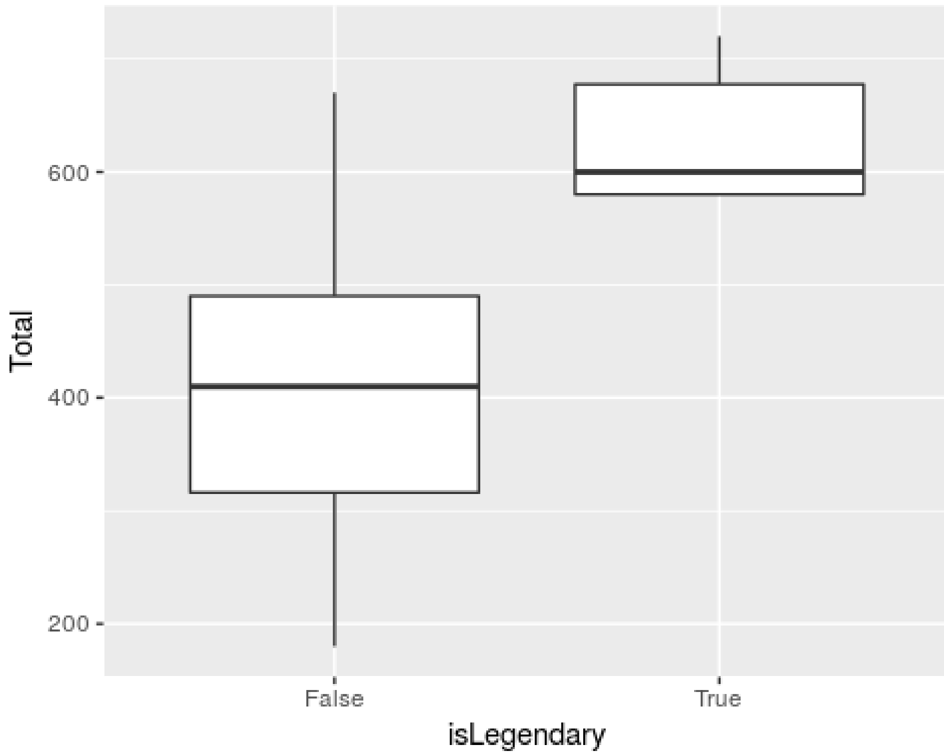
```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = Total ~ factor(Generation), data = updated_pokemon)
##
## $`factor(Generation)`
##            diff        lwr      upr     p adj
## 2-1  -0.8994702 -41.049273 39.25033 0.9999998
## 3-1  -5.0202109 -41.906306 31.86588 0.9988489
## 4-1  38.6775391  -0.674253 78.02933 0.0572432
## 5-1  18.2282221 -17.322910 53.77935 0.6866972
## 6-1  22.5038631 -22.095919 67.10365 0.7013587
## 3-2  -4.1207407 -45.207517 36.96604 0.9997401
## 4-2  39.5770093  -3.736992 82.89101 0.0958988
## 5-2  19.1276923 -20.764941 59.02033 0.7450025
## 6-2  23.4033333 -24.728566 71.53523 0.7334930
## 4-3  43.6977501   3.390429 84.00507 0.0246906
## 5-3  23.2484330 -13.357572 59.85444 0.4567986
## 6-3  27.5240741 -17.921026 72.96917 0.5118616
## 5-4 -20.4493170 -59.538691 18.64006 0.6677134
## 6-4 -16.1736760 -63.641946 31.29459 0.9262290
## 6-5   4.2756410 -40.092773 48.64405 0.9997864
```

From the Tukey test, the adjusted p-value of 0.02469 for generation 4 and 3 is less than the alpha of 0.05. This suggests that the comparison of generation 3 and 4 is significantly different with comparisons of other generations.

Section 5: isLegendary

The focus of this section is whether the legendary form of Pokemon affects Pokemon's performance profoundly.

```
ggplot(updated_pokemon, aes(x=isLegendary, y=Total)) + geom_boxplot()
```



From the boxplot above, it is clearly shown that the distribution of "False" is approximately normal and the distribution of "True" is skewed to the right. Mood's median test would be a good choice to analyze this data because this data does not qualify the assumption of being normally distributed and therefore using the two-sample t-test might be inaccurate.

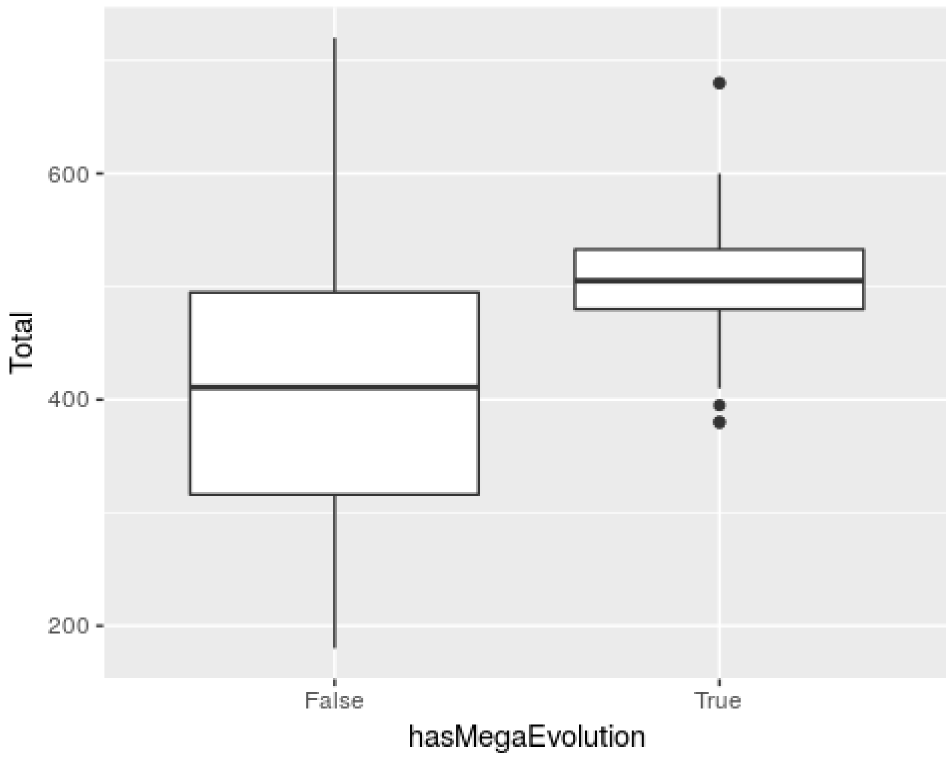
```
median_test(updated_pokemon, Total, isLegendary)

## $table
##        above
## group   above below
##   False   313   360
##   True     46     0
##
## $test
##        what        value
## 1 statistic 4.928102e+01
## 2        df 1.000000e+00
## 3   P-value 2.217979e-12
```

Since p-value of 0.000000000002218 is extremely small compared to Alpha of 0.05, we reject the null hypothesis of legendary pokemon and non-legendary pokemon have the same median of total performance. The box plot indicates that the median of "True" is higher than the median of "False". Based on the above evidence, we conclude that the median of total performance between legendary pokemon and non-legendary pokemon are different and the former one has a higher median.

Section 6: Mega Evolution

In this section, we will examine the relationship between Mega Evolution and pokemon's general performance, the Total.

```
ggplot(updated_pokemon,aes(x=hasMegaEvolution,y=Total))+geom_boxplot()
```



hasMegaEvolution vs Total

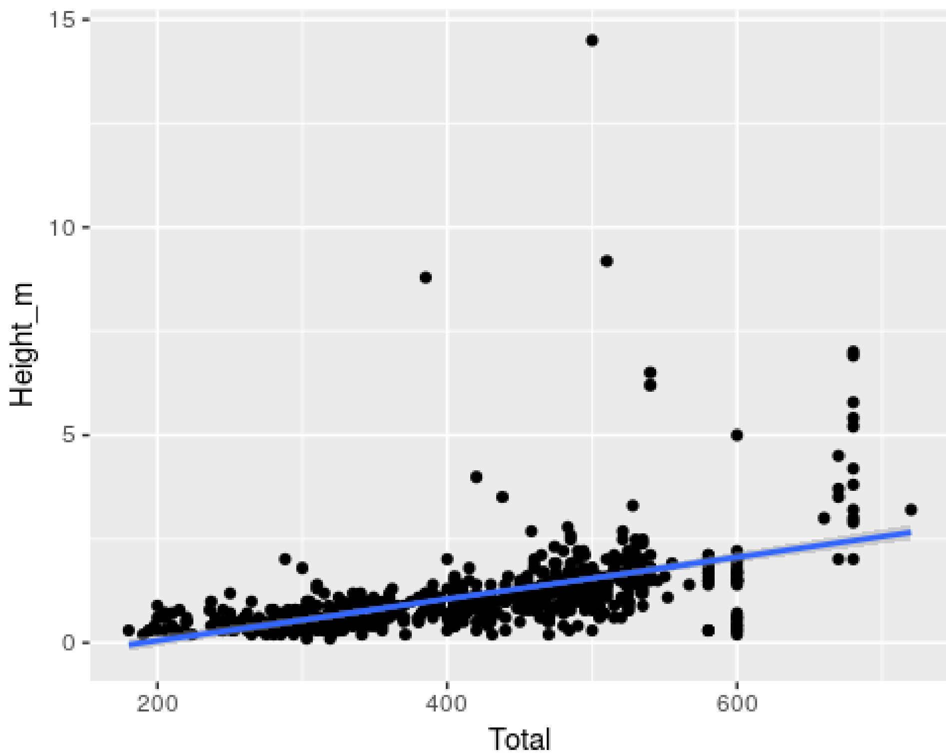
From the boxplot, we observed highly different means and 3 outliers for the pokemon who has Mega Evolution. The means being very different can support our assumption of the medians being very different on the boxplots. We can conclude that if a pokemon has Mega Evolution then it would more likely have resulted in a higher total.

```
median_test(updated_pokemon,Total,hasMegaEvolution)

## $table
##        above
## group   above below
##   False   317   356
##   True     42     4
##
## $test
##        what        value
## 1 statistic 3.365001e+01
## 2        df 1.000000e+00
## 3   P-value 6.597460e-09
```

The p-value is very small, as we expected. The medians of hasMegaEvolution and Total will be different.

Section 7 Height_m & Weight_kg

```
ggplot(updated_pokemon,aes(x=Total,y=Height_m))+geom_point()+geom_smooth(meth
od = "lm")
```



Height_m vs Total

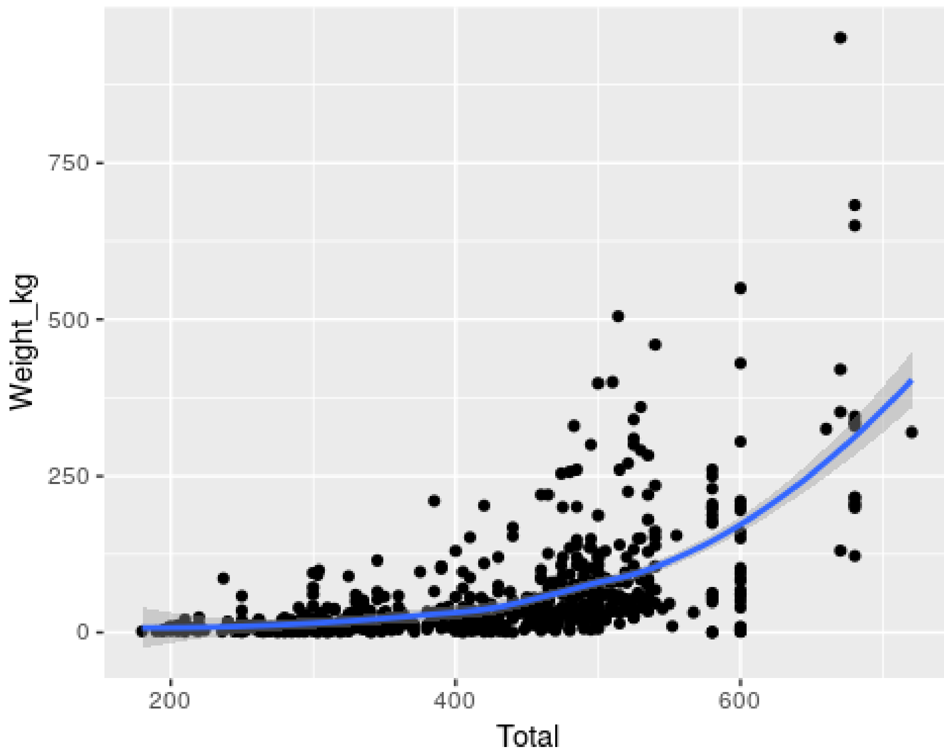
The points scatter around but the linear regression line shows a relationship between the Height and the Total of the pokemon. As the height of pokemon increase, the Total increases as well. Overall Height has very little effect on Total. Since the data is not a normal distribution, we'll need to use Mood's median test for further analyzation.

```
median_test(updated_pokemon,Total,Height_m)

## $table
##        above
## group  above below
##   0.1      0     2
##   0.2      3    10
##   0.3      8    37
##   0.41     5    51
##   0.51     5    50
##   0.61     9    57
##   0.71     6    28
```

```
##   0.79    12    30
##   0.84     1     0
##   0.89    13    17
##   0.99    28    30
##   1.09    21    15
##   1.19    30    15
##   1.3     22     5
##   1.4     25     4
##   1.5     39     2
##   1.6     22     1
##   1.7     22     0
##   1.8     14     2
##   1.91    11     0
##   2.01    17     2
##   2.11     7     0
##   2.21     6     0
##   2.31     2     0
##   2.39     1     0
##   2.49     4     0
##   2.59     1     0
##   2.69     2     0
##   2.79     1     0
##   2.9      1     0
##   3        2     0
##   3.2      2     0
##   3.3      1     0
##   3.51     2     0
##   3.71     1     0
##   3.81     1     0
##   3.99     0     1
##   4.19     1     0
##   4.5      1     0
##   5        1     0
##   5.21     1     0
##   5.41     1     0
##   5.79     1     0
##   6.2      1     0
##   6.5      1     0
##   6.91     1     0
##   7.01     1     0
##   8.79     0     1
##   9.19     1     0
##   14.5     1     0
##
## $test
##        what        value
## 1 statistic 3.438391e+02
## 2        df 4.900000e+01
## 3   P-value 6.752546e-46
```

From Mood's median test, we reject H0, Since the P-value is 6.752546e-46 which is too small, which support our statement earlier. We conclude that there is a difference in the median between Height and Total.

```
ggplot(updated_pokemon,aes(x=Total,y=Weight_kg))+geom_point()+geom_smooth()

## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



Weight_kg vs Total

From the graph above we derive a similar graph as Hight_m vs Total. The line of best fit shows an upward trend, but the effect of Weight on Total seems to be very small, and clearly not a normal distribution, we will need help from Mood's Median test.

```
median_test(updated_pokemon,Total,Weight_kg)

## $table
##          above
## group    above  below
##   0.1        0      3
##   0.3        4      1
##   0.5        0      2
##   0.6        0      3
##   0.8        0      2
##   0.9        0      1
##   1          2      5
##   1.1        1      0
```

```
##   1.2        0     4
##   1.4        1     0
##   1.5        0     2
##   1.7        0     2
##   1.8        0     2
##   1.9        0     2
##   2          0     7
##   2.1        1     3
##   2.2        1     1
##   2.3        0     2
##   2.5        0     4
##   2.6        0     1
##   2.8        0     1
##   2.9        0     1
##   3          2     1
##   3.1        1     1
##   3.2        0     2
##   3.3        0     2
##   3.4        0     1
##   3.5        0     4
##   3.6        0     2
##   3.9        1     2
##   4          2     4
##   4.1        0     1
##   4.2        0     3
##   4.4        1     0
##   4.5        1     2
##   5          3     6
##   5.2        0     1
##   5.3        0     1
##   5.4        0     2
##   5.5        0     5
##   5.7        1     1
##   5.8        1     2
##   5.9        0     2
##   6          0     5
##   6.2        0     1
##   6.3        0     1
##   6.4        0     2
##   6.5        1     4
##   6.6        1     2
##   6.9        0     2
##   7          0     4
##   7.2        0     1
##   7.3        0     2
##   7.4        0     1
##   7.5        1     3
##   7.6        0     1
##   7.7        0     1
##   7.8        0     2
```

```
##   7.9       0    1
##   8         0    4
##   8.1       0    1
##   8.3       0    1
##   8.4       0    1
##   8.5       2    6
##   8.6       0    1
##   8.7       0    1
##   8.8       1    1
##   9         1    5
##   9.3       1    0
##   9.4       0    1
##   9.5       1    4
##   9.9       0    3
##   10        1    3
##   10.1      0    1
##   10.2      0    2
##   10.3      0    1
##   10.4      0    1
##   10.5      2    2
##   10.8      0    2
##   10.9      0    1
##   11        1    3
##   11.2      0    1
##   11.5      0    5
##   11.6      0    1
##   11.8      0    1
##   12        2    4
##   12.4      0    1
##   12.5      2    3
##   13        0    3
##   13.3      0    1
##   13.5      0    2
##   13.6      0    1
##   14        1    2
##   14.3      1    0
##   14.5      1    2
##   14.7      0    1
##   15        2    6
##   15.2      0    2
##   15.3      0    1
##   15.5      2    1
##   15.8      0    1
##   16        0    3
##   16.3      1    1
##   16.5      0    2
##   16.8      0    1
##   17        0    2
##   17.3      0    1
##   17.5      1    1
```

```
##   17.7      0     2
##   18        1     3
##   18.5      0     1
##   18.6      1     0
##   18.8      0     1
##   19        0     3
##   19.2      0     1
##   19.5      0     5
##   19.6      0     1
##   19.8      1     0
##   19.9      1     0
##   20        0     5
##   20.1      1     0
##   20.2      0     2
##   20.3      1     0
##   20.5      2     1
##   20.6      1     0
##   20.8      0     1
##   21        1     2
##   21.2      0     1
##   21.4      0     1
##   21.5      1     1
##   21.6      0     1
##   22        1     2
##   22.5      1     1
##   22.6      1     0
##   23        0     2
##   23.3      0     1
##   23.4      1     0
##   23.5      1     1
##   23.6      1     0
##   23.8      0     1
##   24        1     2
##   24.2      1     0
##   24.4      0     1
##   24.5      2     1
##   24.9      1     0
##   25        2     1
##   25.2      0     1
##   25.3      1     0
##   25.5      1     1
##   25.9      1     0
##   26        0     1
##   26.5      1     0
##   26.6      1     0
##   27        3     1
##   27.3      1     0
##   28        5     3
##   28.4      0     1
##   28.5      1     2
```

```
##   28.8      0     1
##   29        3     1
##   29.5      1     3
##   29.8      0     1
##   29.9      1     0
##   30        4     4
##   30.5      1     1
##   30.6      1     1
##   31        1     3
##   31.5      0     3
##   31.6      1     1
##   32        3     1
##   32.4      0     1
##   32.5      1     3
##   32.6      0     1
##   32.8      1     0
##   33        2     1
##   33.3      1     1
##   33.4      0     1
##   33.5      1     2
##   33.9      1     0
##   34        2     0
##   34.3      1     0
##   34.5      1     1
##   34.6      1     0
##   35        3     1
##   35.3      1     0
##   35.5      1     0
##   35.6      0     1
##   36        0     2
##   36.5      0     1
##   37        0     1
##   37.5      1     1
##   38        3     2
##   38.5      1     0
##   39        3     0
##   39.2      1     1
##   39.5      2     1
##   40        3     1
##   40.2      1     0
##   40.3      1     0
##   40.4      1     0
##   40.5      2     1
##   40.6      1     0
##   40.8      1     0
##   41        1     1
##   41.5      1     0
##   42        1     0
##   42.1      0     1
##   42.4      0     1
```

```
##   42.5      1     0
##   43.8      1     0
##   44        1     0
##   44.4      1     0
##   44.5      1     0
##   45        1     0
##   45.5      0     1
##   46        1     0
##   46.5      1     0
##   46.8      1     0
##   47        2     0
##   48        2     0
##   48.4      1     0
##   48.5      1     0
##   48.7      1     0
##   49.5      0     1
##   49.8      1     0
##   50        0     1
##   50.2      1     0
##   50.5      2     1
##   51        2     0
##   51.3      0     1
##   51.5      1     0
##   52        2     0
##   52.2      1     0
##   52.5      1     1
##   52.6      1     0
##   53        1     0
##   54        3     0
##   54.5      1     0
##   55        5     1
##   55.4      1     0
##   55.5      1     1
##   55.8      1     0
##   56        1     1
##   56.5      0     1
##   57        0     1
##   57.5      1     0
##   58        1     1
##   58.5      0     1
##   59        1     0
##   59.6      1     0
##   60        5     1
##   60.4      1     0
##   60.5      1     1
##   60.8      1     0
##   61        2     0
##   61.5      2     0
##   62        2     0
##   63        2     0
```

```
##   64.8      1     0
##   65        1     1
##   65.5      0     1
##   66.6      1     0
##   68        2     0
##   68.2      1     0
##   70        1     0
##   70.5      0     1
##   71        1     0
##   71.2      1     0
##   71.5      1     0
##   72        0     1
##   75        2     0
##   75.5      1     0
##   75.6      1     0
##   76.5      1     0
##   76.6      1     0
##   77.4      1     0
##   78.5      1     0
##   79.5      3     0
##   80        3     0
##   80.4      1     0
##   80.5      1     0
##   81        2     0
##   81.1      1     0
##   81.5      2     0
##   81.9      1     0
##   82        1     0
##   82.5      1     0
##   84        1     0
##   84.5      1     0
##   85        1     0
##   85.2      1     0
##   85.5      1     0
##   85.6      1     0
##   86.4      0     1
##   87        1     0
##   87.6      0     1
##   88.4      1     0
##   88.8      2     0
##   90        1     1
##   90.5      1     0
##   91        1     0
##   92        0     1
##   92.5      1     0
##   92.9      1     0
##   94.6      2     0
##   95        2     0
##   95.2      0     1
##   96        1     0
```

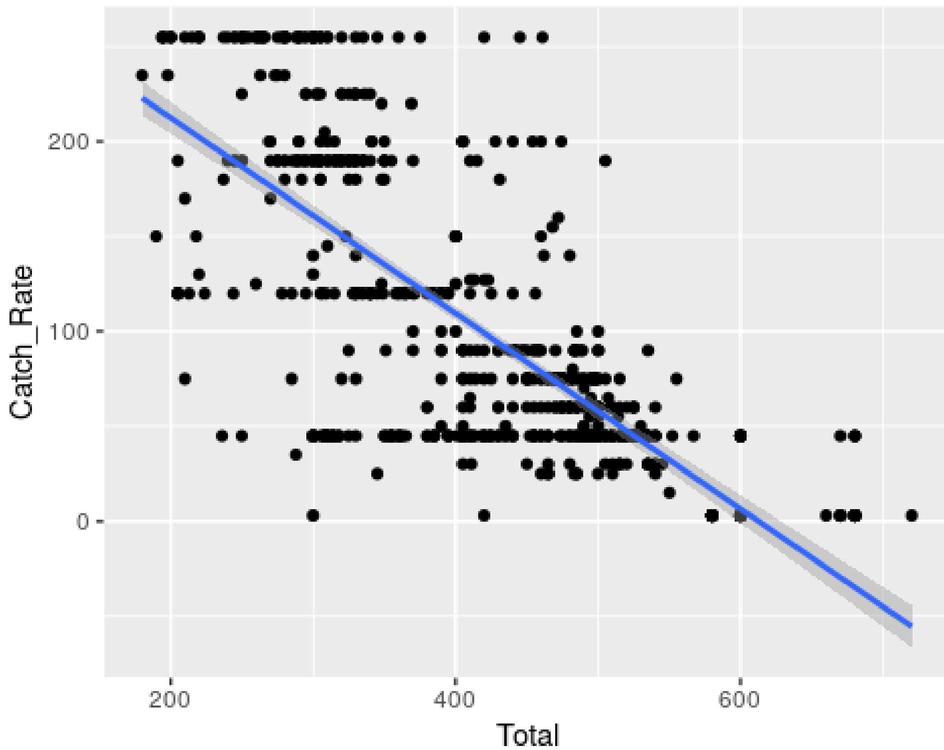
```
##   96.3      1     0
##   97        0     2
##   99.5      0     1
##   100       2     0
##   100.5     1     0
##   102       0     1
##   102.5     1     0
##   102.6     1     0
##   105       0     2
##   105.5     1     0
##   106.6     1     0
##   107.3     1     0
##   108       2     0
##   110       1     0
##   110.5     0     1
##   115       0     1
##   118       1     0
##   120       5     0
##   122       1     0
##   125.8     2     0
##   128.6     1     0
##   130       1     1
##   130.5     1     0
##   132.5     1     0
##   135       1     0
##   135.5     1     0
##   136       1     0
##   138.6     1     0
##   139       1     0
##   140       1     0
##   148       1     0
##   149.5     1     0
##   150       1     0
##   150.5     1     0
##   150.6     1     0
##   152       1     1
##   154       1     0
##   155       1     0
##   160       1     0
##   162       1     0
##   168       1     0
##   175       1     0
##   178       1     0
##   180       1     0
##   187       2     0
##   195       1     0
##   198       1     0
##   199       1     0
##   200       2     0
##   200.5     1     0
```

```
##   202       1     0
##   202.5     0     1
##   203       1     0
##   205       1     0
##   206.5     1     0
##   210       1     1
##   215       1     0
##   216       1     0
##   220       3     0
##   225       1     0
##   230       1     0
##   235       1     0
##   250       1     0
##   253.8     1     0
##   256.5     1     0
##   260       3     0
##   270       1     0
##   282.8     1     0
##   291       1     0
##   300       2     0
##   305       1     0
##   310       1     0
##   320       1     0
##   325       1     0
##   330       2     0
##   336       1     0
##   340       1     0
##   345       1     0
##   352       1     0
##   360       1     0
##   398       1     0
##   400       1     0
##   420       1     0
##   430       1     0
##   460       1     0
##   505       1     0
##   550       1     0
##   650       1     0
##   683       1     0
##   950       1     0
##
## $test
##        what        value
## 1 statistic 4.855853e+02
## 2        df 3.970000e+02
## 3   P-value 1.540259e-03
```

The p-value is larger compared to the previous one, but still very small. So we can reject the null hypothesis, and we can say that there is a difference between the median of Weight and Total.

Session 8: Catch_Rate and Body_Type

```
ggplot(updated_pokemon,aes(x=Total,y=Catch_Rate))+geom_point()+geom_smooth(me
thod = "lm")
```



From all the points scatter around the line, we can say that the relation between is very weak. It appears that the higher the catch rate, higher the Total.

```
median_test(updated_pokemon,Total,Catch_Rate)

## $table
##       above
## group above below
##   3      48     2
##   15      1     0
##   25      9     1
##   30     17     2
##   35      0     1
##   45    141    80
##   50      5     2
##   55      3     0
##   60     40     4
##   65      2     1
##   70      1     0
##   75     47    10
##   80      1     0
##   90     25    11
```

```
##   100     2     3
##   120     3    48
##   125     0     3
##   127     0     4
##   130     0     2
##   140     2     2
##   145     0     1
##   150     1     6
##   155     1     0
##   160     1     0
##   170     0     2
##   180     1     9
##   190     1    68
##   200     5    13
##   205     0     1
##   220     0     2
##   225     0    14
##   235     0     6
##   255     2    62
##
## $test
##        what        value
## 1 statistic 3.566749e+02
## 2        df 3.200000e+01
## 3   P-value 1.734118e-56
```

From Mood's median test, we have P-value is 1.734118e-56, which is too small. Therefore we reject the null hypothesis, hence the median is different between Total and Catch_Rate.

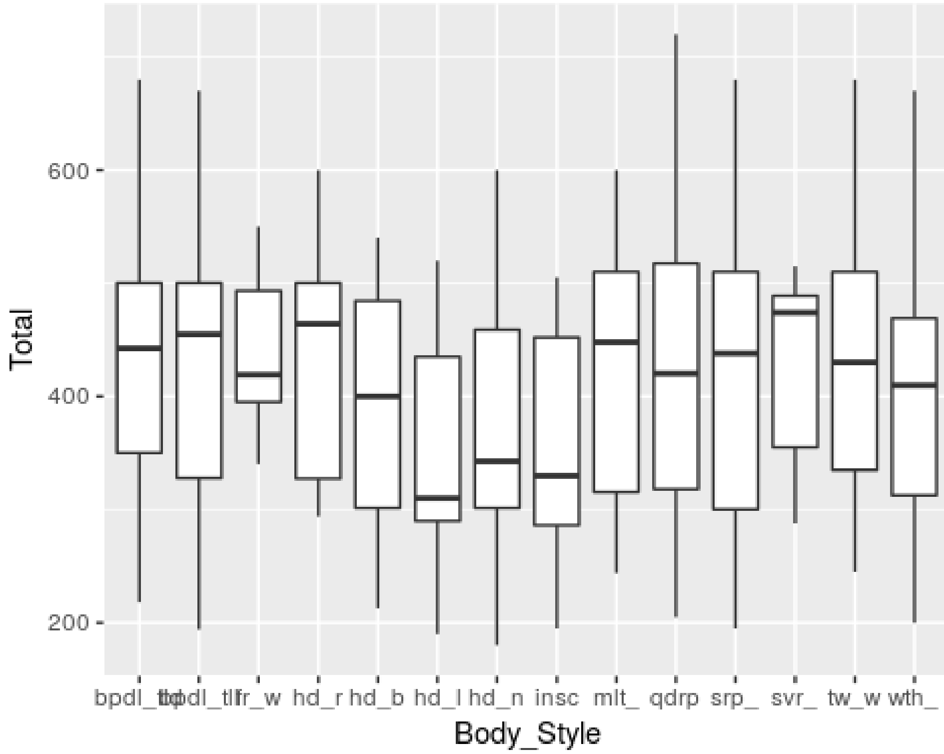
Body_Style vs Total

```
updated_pokemon %>% count(Body_Style)
```

```
## # A tibble: 14 x 2
##    Body_Style           n
##    <chr>            <int>
##  1 bipedal_tailed     158
##  2 bipedal_tailless   109
##  3 four_wings          18
##  4 head_arms           39
##  5 head_base           30
##  6 head_legs           17
##  7 head_only           34
##  8 insectoid           30
##  9 multiple_bodies     15
## 10 quadruped          135
## 11 serpentine_body     29
## 12 several_limbs       13
## 13 two_wings           63
## 14 with_fins           31
```

We have the total of 14 different body styles for pokemon.

```
body = ggplot(updated_pokemon,aes(x=Body_Style,y=Total))+geom_boxplot()
body + scale_x_discrete(labels = abbreviate)
```




```
median_test(updated_pokemon,Total,Body_Style)

## $table
##                     above
## group               above below
##   bipedal_tailed       85    73
##   bipedal_tailless     59    50
##   four_wings            8     9
##   head_arms            21    18
##   head_base            13    16
##   head_legs             5    12
##   head_only            13    21
##   insectoid             8    22
##   multiple_bodies       9     6
##   quadruped            65    70
##   serpentine_body      15    14
##   several_limbs         8     5
##   two_wings            35    28
##   with_fins            15    16
##
## $test
##          what       value
```

```
## 1 statistic 15.8731398
## 2        df 13.0000000
## 3   P-value  0.2560482
```

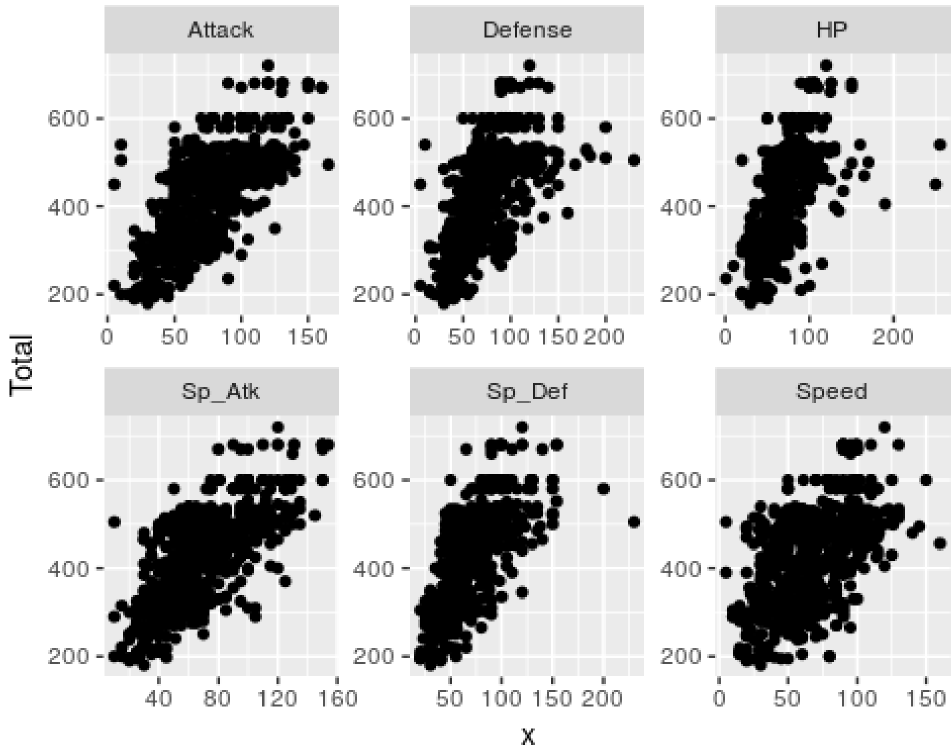
The p-value from the median test is 0.2560482 which fail to reject the null hypothesis, therefore the median difference is zero. There is no evidence that the body style shows any difference on average.

Section 9: Multiple Linear Regression Model and Linear Prediction

The goal for this section is to build a multiple linear regression that can predict the total performance (Total) based on all the other explanatory variables. Total performance(Total) is the sum of six inclusive features of a Pokemon, Total = HP + Attack + Defense + Sp_Atk(special attack) + Sp_Df(special defense) + Speed. And all the other variables are called exclusive features of a Pokemon. First of all, the three key assumptions of multiple linear regression should be ensured before concluding the model. The normality in residuals, no multicollinearity among independent variables, and lastly homoscedasticity of the error terms are required to build a satisfactory predictor.

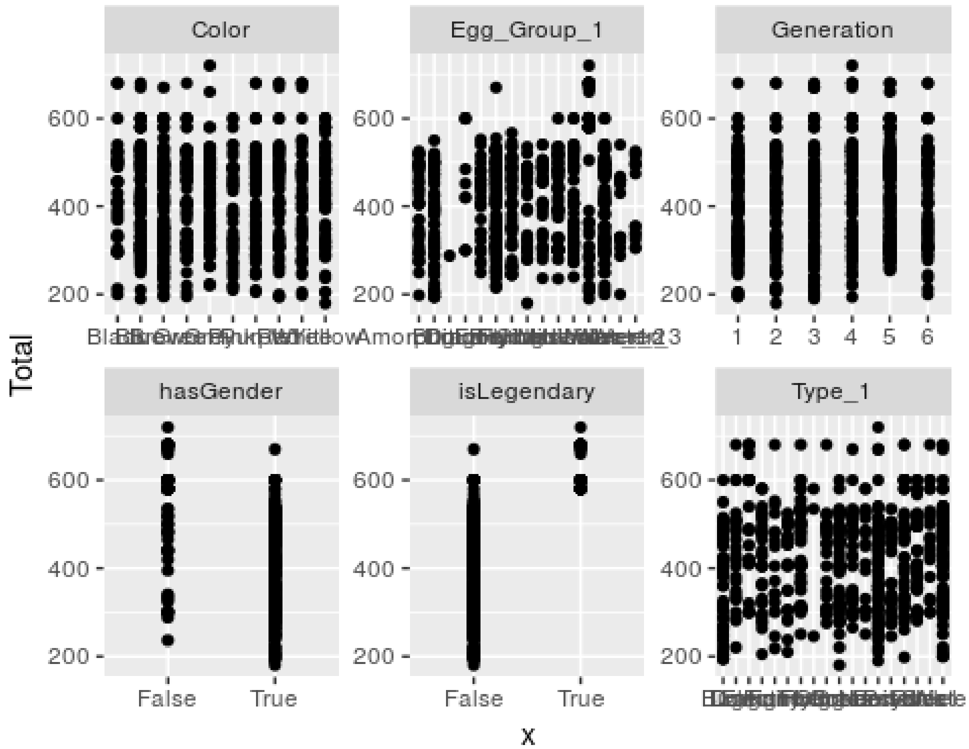
First, multiple linear regression requires the relationship between the independent and dependent variables to be linear. The linearity assumption can best be tested with scatterplots. Now let us first look at the relationship and distribution between the total performance(Total) and its inclusive features.

```
inclusive_var = updated_pokemon %>% gather(xname, x, c(HP:Speed)) %>% ggplot(
aes(x=x, y=Total)) + geom_point() + facet_wrap(~xname, scales="free")
inclusive_var
```

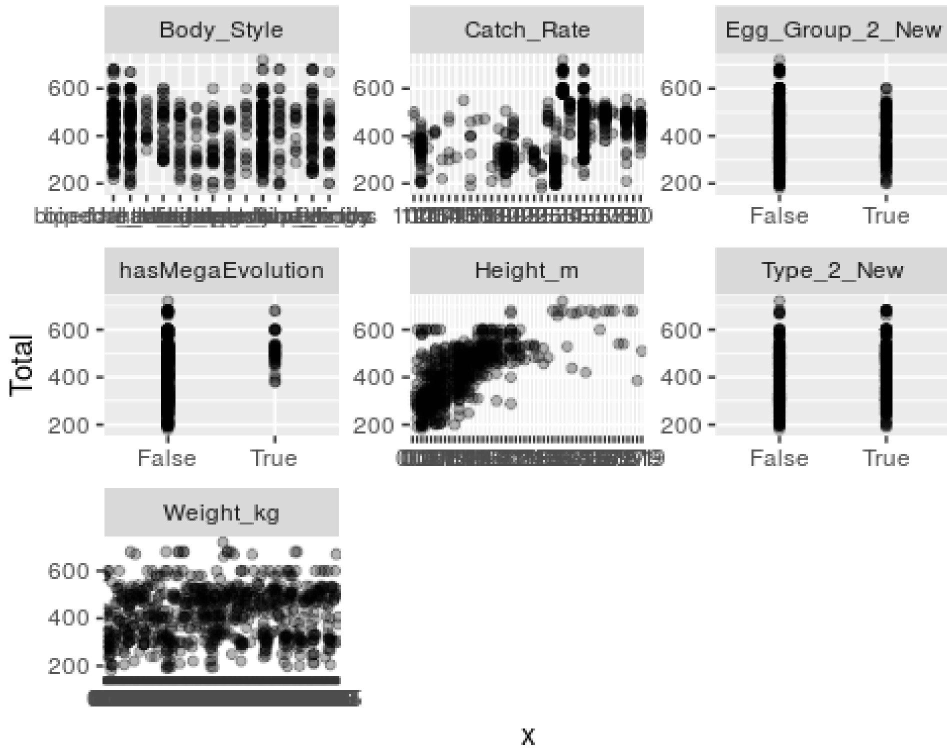


The above distributions all look fairly linear. This is not surprising because total performance(Total) can be predicted perfectly with a multiple linear regression with weight of 1 given to each of the inclusive features of a Pokemon. Hence, the inclusive features should be not be considered in the model. The relationship between the total performance and the rest of the variables are more interesting to look at:

```
exclusive_var = updated_pokemon %>% gather(xname, x, c(Type_1, Generation:has
Gender, Egg_Group_1)) %>% ggplot(aes(x=x, y=Total)) + geom_point() + facet_wr
ap(~xname, scales="free")
exclusive_var
```

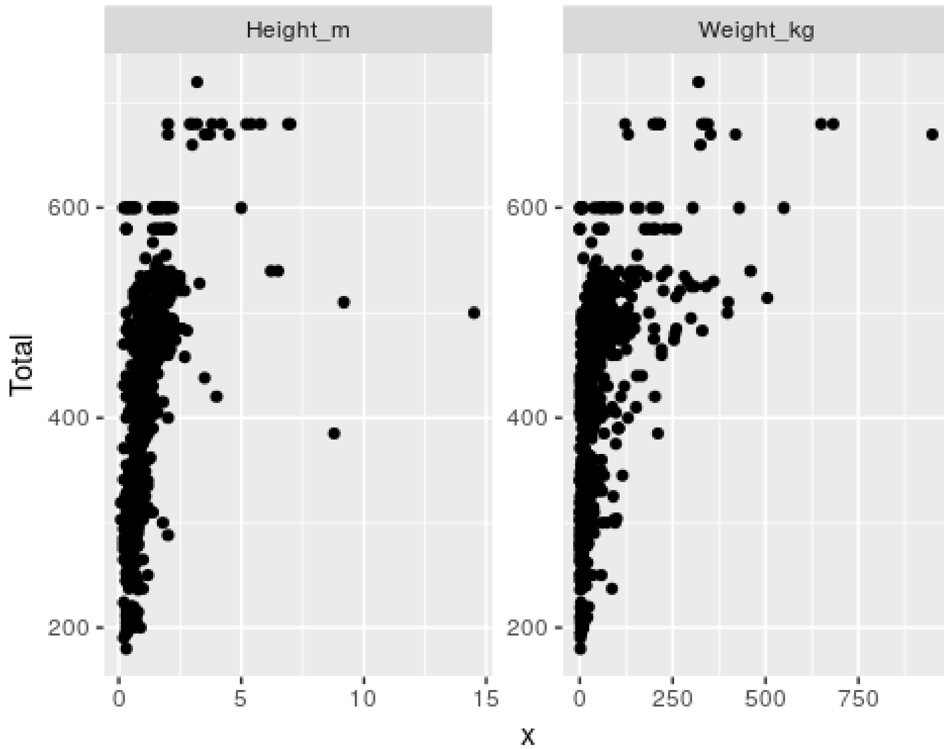



```
exclusive_var = updated_pokemon %>% gather(xname, x, c(hasMegaEvolution:Egg_G
roup_2_New)) %>% ggplot(aes(x=x, y=Total)) + geom_point(alpha=0.3) + facet_wr
ap(~xname, scales="free", ncol=3)
exclusive_var
```



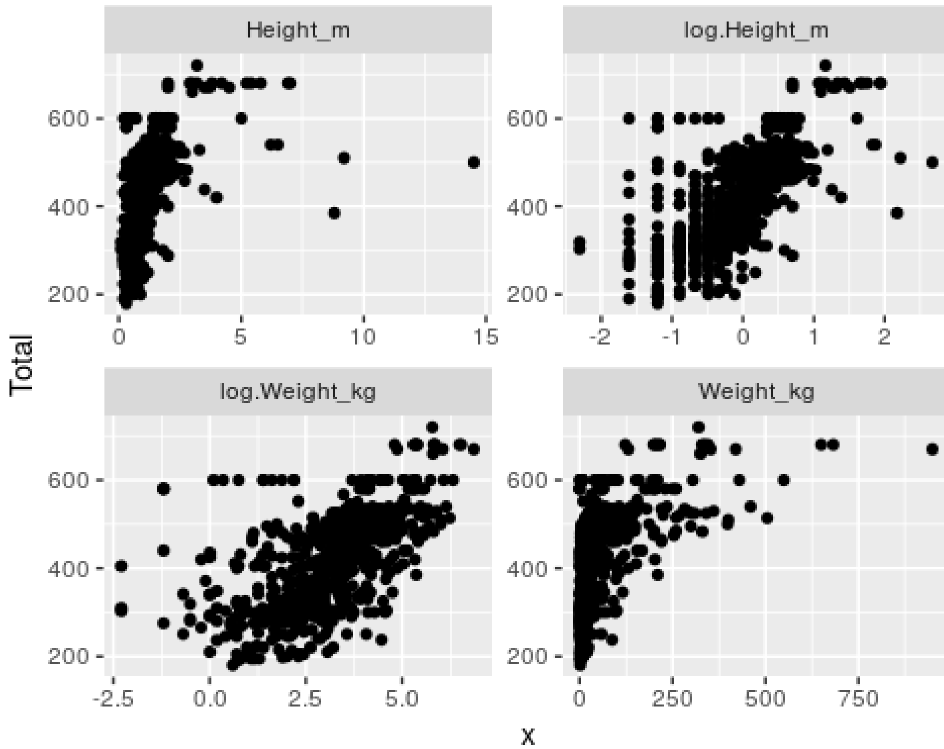
From above, there are a few variables not behaving linearly. They should be examined individually. The next diagrams show clearer relationships of height-total and weight-total.

```
exclusive_var = updated_pokemon %>% gather(xname, x, c(Height_m,Weight_kg)) %
>% ggplot(aes(x=x, y=Total)) + geom_point() + facet_wrap(~xname, scales="free
")
exclusive_var
```



It is concerning that a moderate non-linearity occurs in height-total and weight-total relationships. A data transformation of weight and height is desired in this situation in order to meet the assumption of linearity. After a few trials and errors, the remedy to this problem is to interpret height and weight as log of the height and log of the weight in the model. The dataset updated_pokemon has two additional columns log.Weight_kg and log.Height_m.

```
updated_pokemon = updated_pokemon %>% mutate(log.Weight_kg=log(Weight_kg))
updated_pokemon = updated_pokemon %>% mutate(log.Height_m=log(Height_m))
g = updated_pokemon %>% gather(xname, x, c(log.Height_m, log.Weight_kg, Heigh
t_m, Weight_kg)) %>% ggplot(aes(x=x, y=Total)) + geom_point() + facet_wrap(~x
name, scales="free")
g
```

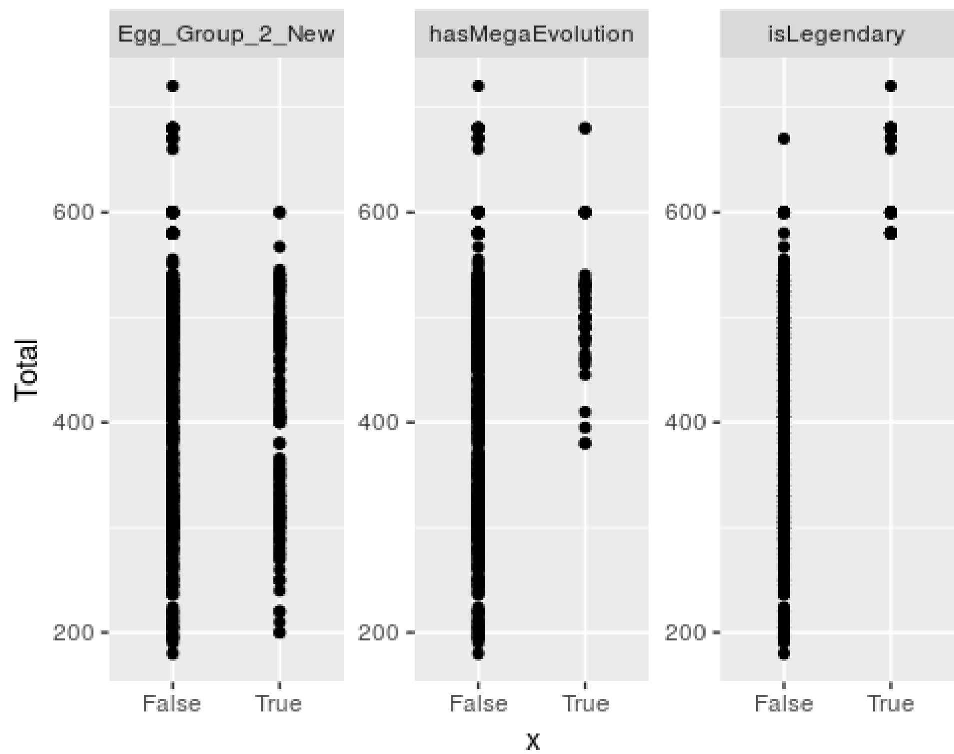



```
updated_pokemon

## # A tibble: 721 x 28
##    Number Name  Type_1 Type_2 Total    HP Attack Defense Sp_Atk Sp_Def
##     <int> <chr> <chr>  <chr>  <int> <int>  <int>   <int>  <int>  <int>
##  1      1 Bulb… Grass  Poison   318    45     49      49     65     65
##  2      2 Ivys… Grass  Poison   405    60     62      63     80     80
##  3      3 Venu… Grass  Poison   525    80     82      83    100    100
##  4      4 Char… Fire   <NA>     309    39     52      43     60     50
##  5      5 Char… Fire   <NA>     405    58     64      58     80     65
##  6      6 Char… Fire   Flying   534    78     84      78    109     85
##  7      7 Squi… Water  <NA>     314    44     48      65     50     64
##  8      8 Wart… Water  <NA>     405    59     63      80     65     80
##  9      9 Blas… Water  <NA>     530    79     83     100     85    105
## 10     10 Cate… Bug    <NA>     195    45     30      35     20     20
## # ... with 711 more rows, and 18 more variables: Speed <int>,
## #   Generation <int>, isLegendary <chr>, Color <chr>, hasGender <chr>,
## #   Pr_Male <dbl>, Egg_Group_1 <chr>, Egg_Group_2 <chr>,
## #   hasMegaEvolution <chr>, Height_m <dbl>, Weight_kg <dbl>,
## #   Catch_Rate <int>, Body_Style <chr>, Type_2_New <chr>,
## #   Egg_Group_2_New <chr>, Pr_Male_New <chr>, log.Weight_kg <dbl>,
## #   log.Height_m <dbl>
```

Once the data transformation is performed on the height and weight variables, the linearity has been notably improved.

Now let us get a better picture of total performance against isLegendary, hasMegaEvolution and Egg_Group_2_New.

```
exclusive_var = updated_pokemon %>% gather(xname, x, c(isLegendary, hasMegaEv
olution, Egg_Group_2_New)) %>% ggplot(aes(x=x, y=Total)) + geom_point() + fac
et_wrap(~xname, scales="free")
exclusive_var
```



The unequal spread in hasMegaEvolution and isLegendary are observed between True and False. However, they still remain in the linear model before checking residuals against the fitted values. With prior knowledge, it is expected and natural for a Pokemon to have a strong total performance if it has mega evolution or is legendary.

Moreover, multicollinearity among independent variables is not allowed in the multiple linear models. So, exploring the correlation among the explanatory variables is required.

Let us first look at the correlation among the quantitative variables such as height, weight, and catch rate.

```
cor(updated_pokemon[,c ("Height_m", "Weight_kg", "Catch_Rate")])

##              Height_m  Weight_kg Catch_Rate
## Height_m    1.0000000  0.6613415 -0.3828625
## Weight_kg   0.6613415  1.0000000 -0.3677981
## Catch_Rate -0.3828625 -0.3677981  1.0000000
```

A strong uphill linear relationship between height and weight is captured in the dataset due to a moderately high correlation of 0.66. Since they are highly correlated, weight is dropped in the model. Correlation among categorical variables is studied next. The top two most correlated pairs of categorical variables are hasGender-Pr_Male_New and hasMegaEvolution-isLegendary. The function count is used below to count the number of same or different Trues or Falses in the dataset.

```
updated_pokemon%>%count(hasGender, Pr_Male_New)

## # A tibble: 2 x 3
##   hasGender Pr_Male_New     n
##   <chr>     <chr>       <int>
## 1 False     False          77
## 2 True      True          644

updated_pokemon%>%count(hasMegaEvolution, isLegendary)

## # A tibble: 4 x 3
##   hasMegaEvolution isLegendary     n
##   <chr>            <chr>       <int>
## 1 False            False         634
## 2 False            True           41
## 3 True             False          41
## 4 True             True            5
```

In conclusion, either hasGender or Pr_Male_New should be dropped in the linear model because they have a perfect correlation. Height, hasMegaEvolution, and isLengendary stay in the model because there is still information that might be helpful and they will be dealt with later if more problems appear.

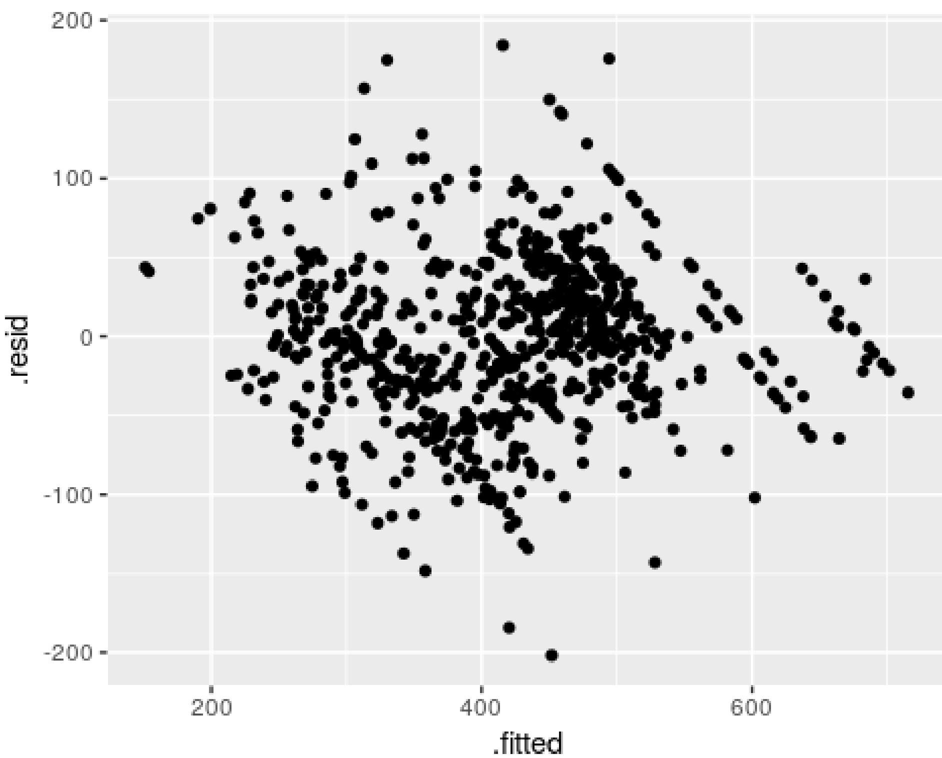
Now the data is ready to be analyzed. In multiple regression, since it is tough to guess which variables affect response, so it is wise to start by looking at everything. Since hasGender and Pr_Male_New have a perfect correlation, to avoid singularity problem in the regression, Pr_Male_New column is dropped to compute the first linear model pika.1.

```
pika.1=lm(Total~factor(Type_1)+factor(Type_2_New)+factor(Generation)+factor(i
sLegendary)+
           factor(Color)+factor(hasGender)+factor(Egg_Group_1)+factor(Egg_Gro
up_2_New)+
            factor(hasMegaEvolution)+ log.Height_m+Catch_Rate+factor(Body_Sty
le), data=updated_pokemon)
summary(pika.1)$adj.r.squared

## [1] 0.7579039

summary(pika.1)$r.squared

## [1] 0.7797598

tidy(pika.1) %>% arrange(p.value)
```

```
## # A tibble: 66 x 5
##    term                              estimate std.error statistic  p.value
##    <chr>                                <dbl>     <dbl>     <dbl>    <dbl>
##  1 log.Height_m                        71.9      4.23      17.0   5.82e-54
##  2 Catch_Rate                          -0.613    0.0362   -16.9   1.51e-53
##  3 (Intercept)                        472.      28.2       16.7   1.16e-52
##  4 factor(isLegendary)True             78.9     16.3        4.84 1.60e- 6
##  5 factor(hasGender)True              -52.5     11.3       -4.66 3.76e- 6
##  6 factor(hasMegaEvolution)True        41.9      9.09       4.61 4.81e- 6
##  7 factor(Body_Style)serpentine_body  -50.6     12.2       -4.15 3.72e- 5
##  8 factor(Egg_Group_1)Fairy            65.7     17.6        3.74 1.97e- 4
##  9 factor(Generation)4                 25.3      7.34       3.45 5.89e- 4
## 10 factor(Generation)5                 23.4      7.01       3.33 9.03e- 4
## # ... with 56 more rows
```

With significant p-value and F-statistic, the adjusted R-square is 0.7579 which is great. However there are many non-significant variables, before consider removing non-significant variables, it is natural to check residual plots. This might help us to get a fairer judgement of what is happening. The residual plot is computed and as shown below:

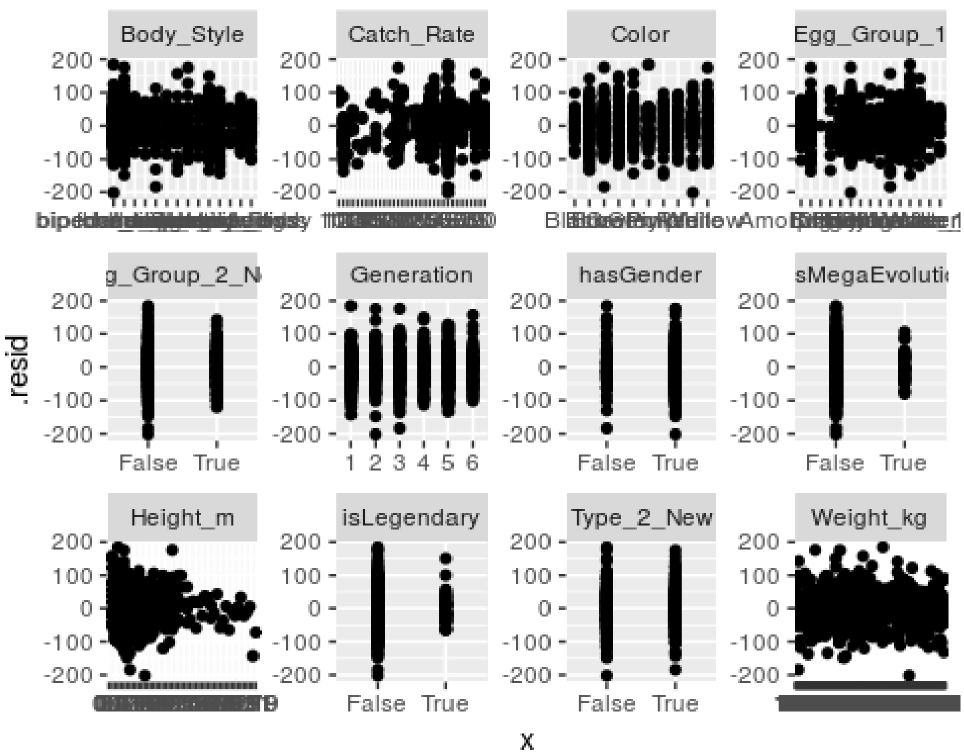
```
ggplot(pika.1,aes(x=.fitted,y=.resid))+geom_point()
```



In general, the residual plot is symmetrically distributed, tending to cluster towards the middle of the plot with a relatively random pattern. The residuals cannot be predicted by the fitted values so the assumption of homoscedasticity in multiple linear regression is ensured.

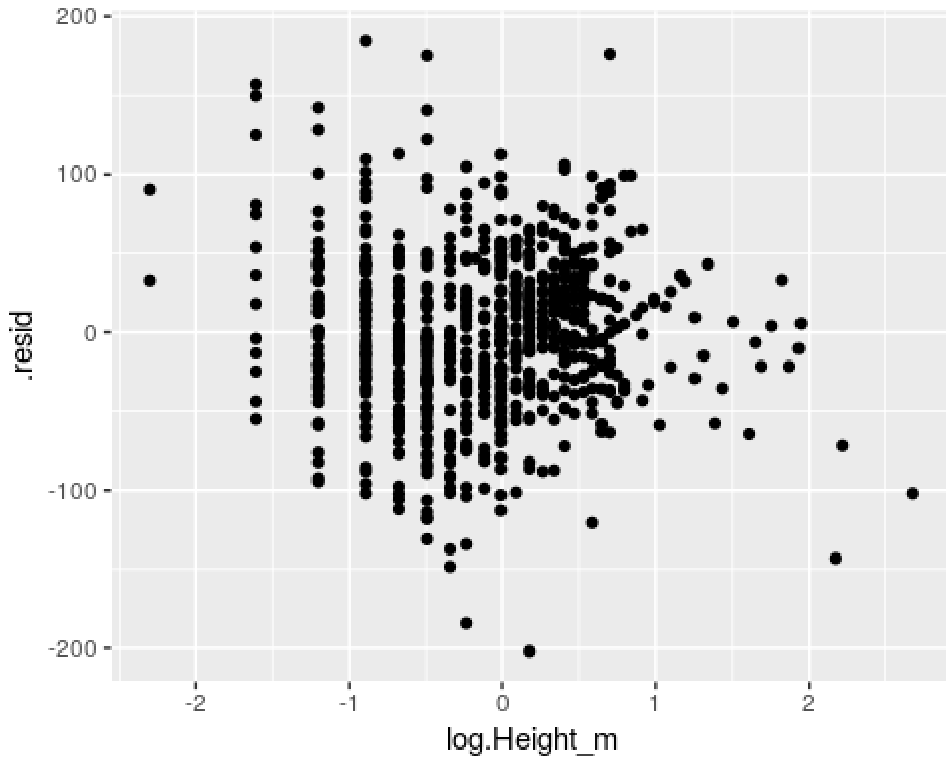
A residual plot against each explanatory variable would help get a more precise view of any patterns.

```
pika.1a=augment(pika.1, updated_pokemon)
g = pika.1a%>%gather(xname, x, c(Generation:hasGender, Egg_Group_1, hasMegaEv
olution:Egg_Group_2_New)) %>% ggplot(aes(x=x, y=.resid)) + geom_point() + fac
et_wrap(~xname, scales="free")
g
```



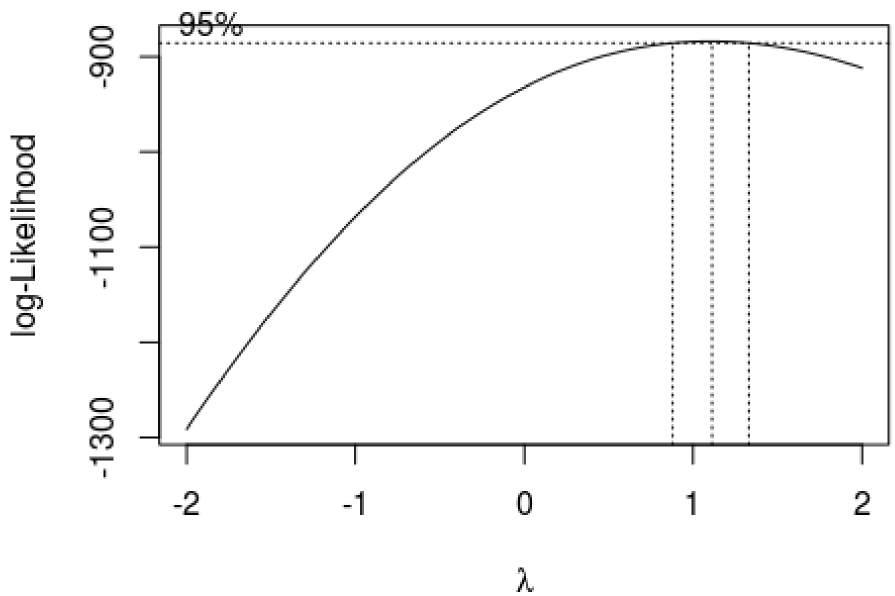
To zoom in, the residual plot for log height seems unbalanced to the right side of it.

```
pika.1a=augment(pika.1, updated_pokemon)
g = ggplot(pika.1a, aes(x=log.Height_m, y=.resid)) + geom_point()
g
```



Box-Cox is applied in this situation to find out the best parameter if the data transformation for the response variable is needed.

```
boxcox(Total~factor(Type_1)+factor(Type_2_New)+factor(Generation)+factor(isLe
gendary)+
           factor(Color)+factor(hasGender)+factor(Egg_Group_1)+factor(Egg_Gro
up_2_New)+
            factor(hasMegaEvolution)+ log.Height_m+log.Weight_kg+Catch_Rate+f
actor(Body_Style), data=updated_pokemon)
```



Since lambda is near 1, box-cox suggests no transformation is needed in this situation. So a linear model is the best fit for the dataset and using the function leaps here can perform a detailed search for the best subsets of the variables in x for predicting y in linear regression.

Now the procedure of removing non-significant variables start.

```
leaps=regsubsets(Total~factor(Type_1)+factor(Type_2_New)+factor(Generation)+f
actor(isLegendary)+
           factor(Color)+factor(hasGender)+factor(Egg_Group_1)+factor(Egg_Gro
up_2_New)+
            factor(hasMegaEvolution)+log.Height_m+log.Weight_kg+
             Catch_Rate+factor(Body_Style),data=updated_pokemon,really.big=TR
UE, nbest=1)
s=summary(leaps)
with(s,data.frame(adjr2,outmat))
```

```
##                adjr2 factor.Type_1.Dark factor.Type_1.Dragon
## 1  ( 1 ) 0.5444241
## 2  ( 1 ) 0.6583685
## 3  ( 1 ) 0.7066196
## 4  ( 1 ) 0.7136802
## 5  ( 1 ) 0.7197992
## 6  ( 1 ) 0.7257060
## 7  ( 1 ) 0.7291455
## 8  ( 1 ) 0.7327799
##          factor.Type_1.Electric factor.Type_1.Fairy factor.Type_1.Fighting
```

```
## 1  ( 1 )
## 2  ( 1 )
## 3  ( 1 )
## 4  ( 1 )
## 5  ( 1 )
## 6  ( 1 )
## 7  ( 1 )
## 8  ( 1 )
##          factor.Type_1.Fire factor.Type_1.Flying factor.Type_1.Ghost
## 1  ( 1 )
## 2  ( 1 )
## 3  ( 1 )
## 4  ( 1 )
## 5  ( 1 )
## 6  ( 1 )
## 7  ( 1 )
## 8  ( 1 )
##          factor.Type_1.Grass factor.Type_1.Ground factor.Type_1.Ice
## 1  ( 1 )
## 2  ( 1 )
## 3  ( 1 )
## 4  ( 1 )
## 5  ( 1 )
## 6  ( 1 )
## 7  ( 1 )
## 8  ( 1 )
##          factor.Type_1.Normal factor.Type_1.Poison factor.Type_1.Psychic
## 1  ( 1 )
## 2  ( 1 )
## 3  ( 1 )
## 4  ( 1 )
## 5  ( 1 )
## 6  ( 1 )
## 7  ( 1 )
## 8  ( 1 )
##          factor.Type_1.Rock factor.Type_1.Steel factor.Type_1.Water
## 1  ( 1 )
## 2  ( 1 )
## 3  ( 1 )
## 4  ( 1 )
## 5  ( 1 )
## 6  ( 1 )
## 7  ( 1 )
## 8  ( 1 )
##          factor.Type_2_New.True factor.Generation.2 factor.Generation.3
## 1  ( 1 )
## 2  ( 1 )
## 3  ( 1 )
## 4  ( 1 )
## 5  ( 1 )
```

```
## 6  ( 1 )
## 7  ( 1 )
## 8  ( 1 )
##          factor.Generation.4 factor.Generation.5 factor.Generation.6
## 1  ( 1 )
## 2  ( 1 )
## 3  ( 1 )
## 4  ( 1 )
## 5  ( 1 )
## 6  ( 1 )
## 7  ( 1 )
## 8  ( 1 )                                       *                   *
##          factor.isLegendary.True factor.Color.Blue factor.Color.Brown
## 1  ( 1 )
## 2  ( 1 )
## 3  ( 1 )                       *
## 4  ( 1 )                       *
## 5  ( 1 )                       *
## 6  ( 1 )                       *
## 7  ( 1 )                       *
## 8  ( 1 )                       *
##          factor.Color.Green factor.Color.Grey factor.Color.Pink
## 1  ( 1 )
## 2  ( 1 )
## 3  ( 1 )
## 4  ( 1 )
## 5  ( 1 )
## 6  ( 1 )
## 7  ( 1 )
## 8  ( 1 )
##          factor.Color.Purple factor.Color.Red factor.Color.White
## 1  ( 1 )
## 2  ( 1 )
## 3  ( 1 )
## 4  ( 1 )
## 5  ( 1 )
## 6  ( 1 )
## 7  ( 1 )
## 8  ( 1 )
##          factor.Color.Yellow factor.hasGender.True factor.Egg_Group_1.Bug
## 1  ( 1 )
## 2  ( 1 )
## 3  ( 1 )
## 4  ( 1 )
## 5  ( 1 )
## 6  ( 1 )                                         *
## 7  ( 1 )                                         *
## 8  ( 1 )                                         *
##          factor.Egg_Group_1.Ditto factor.Egg_Group_1.Dragon
## 1  ( 1 )
```

```
## 2  ( 1 )
## 3  ( 1 )
## 4  ( 1 )
## 5  ( 1 )
## 6  ( 1 )
## 7  ( 1 )
## 8  ( 1 )
##          factor.Egg_Group_1.Fairy factor.Egg_Group_1.Field
## 1  ( 1 )
## 2  ( 1 )
## 3  ( 1 )
## 4  ( 1 )
## 5  ( 1 )                        *
## 6  ( 1 )                        *
## 7  ( 1 )                        *
## 8  ( 1 )                        *
##          factor.Egg_Group_1.Flying factor.Egg_Group_1.Grass
## 1  ( 1 )
## 2  ( 1 )
## 3  ( 1 )
## 4  ( 1 )
## 5  ( 1 )
## 6  ( 1 )
## 7  ( 1 )
## 8  ( 1 )
##          factor.Egg_Group_1.Human.Like factor.Egg_Group_1.Mineral
## 1  ( 1 )
## 2  ( 1 )
## 3  ( 1 )
## 4  ( 1 )
## 5  ( 1 )
## 6  ( 1 )
## 7  ( 1 )
## 8  ( 1 )
##          factor.Egg_Group_1.Monster factor.Egg_Group_1.Undiscovered
## 1  ( 1 )
## 2  ( 1 )
## 3  ( 1 )
## 4  ( 1 )
## 5  ( 1 )
## 6  ( 1 )
## 7  ( 1 )                          *
## 8  ( 1 )
##          factor.Egg_Group_1.Water_1 factor.Egg_Group_1.Water_2
## 1  ( 1 )
## 2  ( 1 )
## 3  ( 1 )
## 4  ( 1 )
## 5  ( 1 )
## 6  ( 1 )
```

```
## 7  ( 1 )
## 8  ( 1 )
##          factor.Egg_Group_1.Water_3 factor.Egg_Group_2_New.True
## 1  ( 1 )
## 2  ( 1 )
## 3  ( 1 )
## 4  ( 1 )
## 5  ( 1 )
## 6  ( 1 )
## 7  ( 1 )
## 8  ( 1 )
##          factor.hasMegaEvolution.True log.Height_m log.Weight_kg
## 1  ( 1 )
## 2  ( 1 )                                         *
## 3  ( 1 )                                         *
## 4  ( 1 )                                         *
## 5  ( 1 )                                         *
## 6  ( 1 )                                         *
## 7  ( 1 )                                         *
## 8  ( 1 )                                         *
##          Catch_Rate factor.Body_Style.bipedal_tailless
## 1  ( 1 )          *
## 2  ( 1 )          *
## 3  ( 1 )          *
## 4  ( 1 )          *
## 5  ( 1 )          *
## 6  ( 1 )          *
## 7  ( 1 )          *
## 8  ( 1 )          *
##          factor.Body_Style.four_wings factor.Body_Style.head_arms
## 1  ( 1 )
## 2  ( 1 )
## 3  ( 1 )
## 4  ( 1 )
## 5  ( 1 )
## 6  ( 1 )
## 7  ( 1 )
## 8  ( 1 )
##          factor.Body_Style.head_base factor.Body_Style.head_legs
## 1  ( 1 )
## 2  ( 1 )
## 3  ( 1 )
## 4  ( 1 )
## 5  ( 1 )
## 6  ( 1 )
## 7  ( 1 )
## 8  ( 1 )
##          factor.Body_Style.head_only factor.Body_Style.insectoid
## 1  ( 1 )
## 2  ( 1 )
```

```
## 3  ( 1 )
## 4  ( 1 )
## 5  ( 1 )
## 6  ( 1 )
## 7  ( 1 )
## 8  ( 1 )
##          factor.Body_Style.multiple_bodies factor.Body_Style.quadruped
## 1  ( 1 )
## 2  ( 1 )
## 3  ( 1 )
## 4  ( 1 )
## 5  ( 1 )
## 6  ( 1 )
## 7  ( 1 )
## 8  ( 1 )
##          factor.Body_Style.serpentine_body factor.Body_Style.several_limbs
## 1  ( 1 )
## 2  ( 1 )
## 3  ( 1 )
## 4  ( 1 )                                 *
## 5  ( 1 )                                 *
## 6  ( 1 )                                 *
## 7  ( 1 )                                 *
## 8  ( 1 )                                 *
##          factor.Body_Style.two_wings factor.Body_Style.with_fins
## 1  ( 1 )
## 2  ( 1 )
## 3  ( 1 )
## 4  ( 1 )
## 5  ( 1 )
## 6  ( 1 )
## 7  ( 1 )
## 8  ( 1 )
```

The best model from the backward model selection includes 6 variables generation, isLegendary, hasGender, Egg_Group_1, Catch_Rate and log.Height_m.

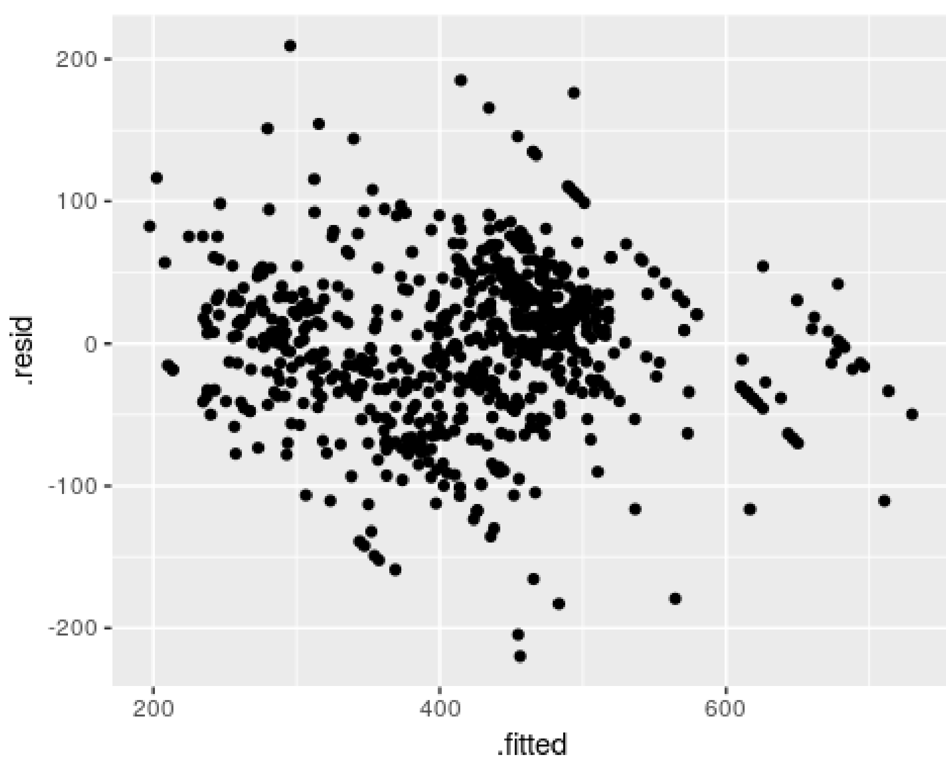
```
pika.f=lm(Total~factor(Generation)+factor(isLegendary)+
           factor(hasGender)+factor(Egg_Group_1)+
            Catch_Rate+log.Height_m, data=updated_pokemon)
summary(pika.f)$adj.r.squared

## [1] 0.7390359

summary(pika.f)$r.squared

## [1] 0.7473722

tidy(pika.f) %>% arrange(p.value)
```

```
## # A tibble: 24 x 5
##    term                        estimate std.error statistic   p.value
##    <chr>                          <dbl>     <dbl>     <dbl>     <dbl>
##  1 (Intercept)                  526.       14.5        36.2  3.66e-162
##  2 Catch_Rate                    -0.639     0.0350    -18.2  5.28e- 61
##  3 log.Height_m                  67.1       3.99       16.8  1.55e- 53
##  4 factor(hasGender)True        -56.4      10.9        -5.16 3.25e-  7
##  5 factor(isLegendary)True       77.4      16.1         4.81 1.82e-  6
##  6 factor(Egg_Group_1)Fairy      47.0      13.7         3.44 6.11e-  4
##  7 factor(Generation)5           21.1       6.66        3.17 1.59e-  3
##  8 factor(Generation)4           21.4       7.33        2.91 3.67e-  3
##  9 factor(Generation)6           23.9       8.28        2.88 4.05e-  3
## 10 factor(Egg_Group_1)Ditto    -135.       58.0        -2.32 2.05e-  2
## # ... with 14 more rows
```

The adjusted r-squared goes down by 0.02 from 0.7606 to 0.7390 by reducing more than half of the explanatory variables. Therefore the pika.f model is accepted and the multiple linear model assumptions should be checked again.

Let us first look at the residual plot for the final model pika.f.

```
g=ggplot(pika.f,aes(y=.resid,x=.fitted))+geom_point()
g
```



The residual plot still looks similar to the first model’s. No noticeable pattern is observed.

In order to make a fair judgement on how well the final model is good at prediction, it is beneficial to compare the observed response variable with the predicted.

A random subset takes 10 observations from the updated_pokemon as seen below. The fitted.results is a vector contains the predicted or expected response variables.

```
subset= updated_pokemon[325:335,]
fitted.results = predict(pika.f, newdata = subset(subset, select =c(12, 13, 1
5, 17, 22, 28)),type='response')
as.vector(round(fitted.results))

##  [1] 290 429 318 268 383 472 311 471 255 455 436

subset$Total

##  [1] 330 470 360 290 340 520 335 475 310 490 458
```

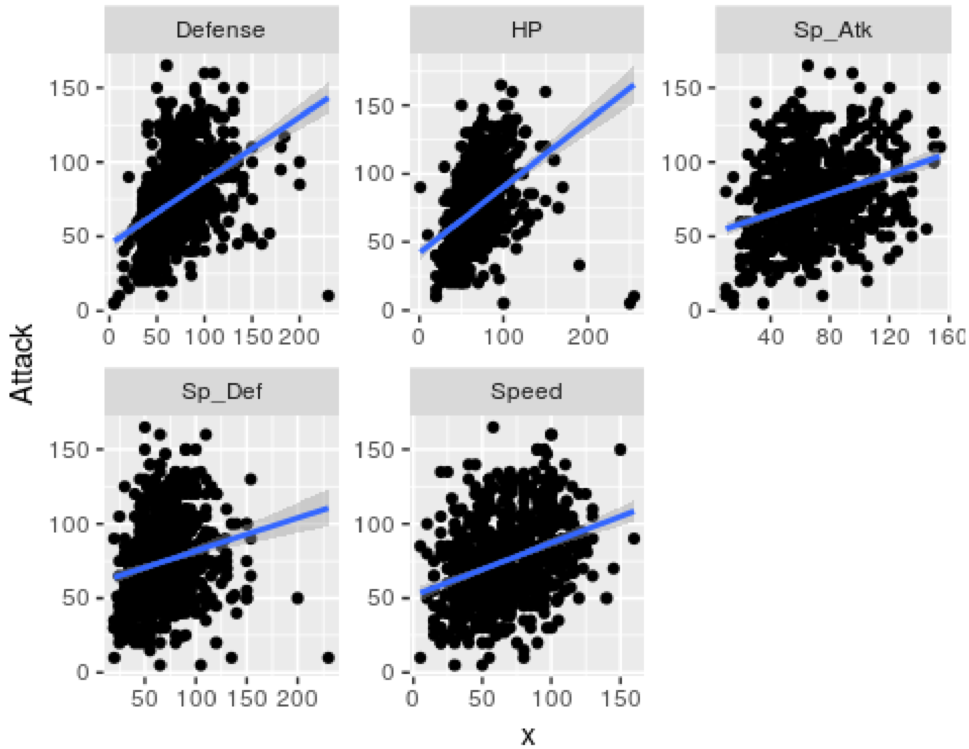
From the above chart, it is evident that the final model pika.f is a good fit. Half of the data points are overfitted and the other half are underfitted within the arranging of -50 to 50. By considering the fact that the total performance ranges from 180 to 720, the model is generally good at predicting.

Section 10 Response Variables

The variable Total is the ultimate response variable we are analyzing on. It consists of the sums of each Pokemon's values of Attack, Defense, HP, Special Attack, Special Defense, and Speed. In this Section, we are going to focus on the relationship between the variables that build up the final Total value.

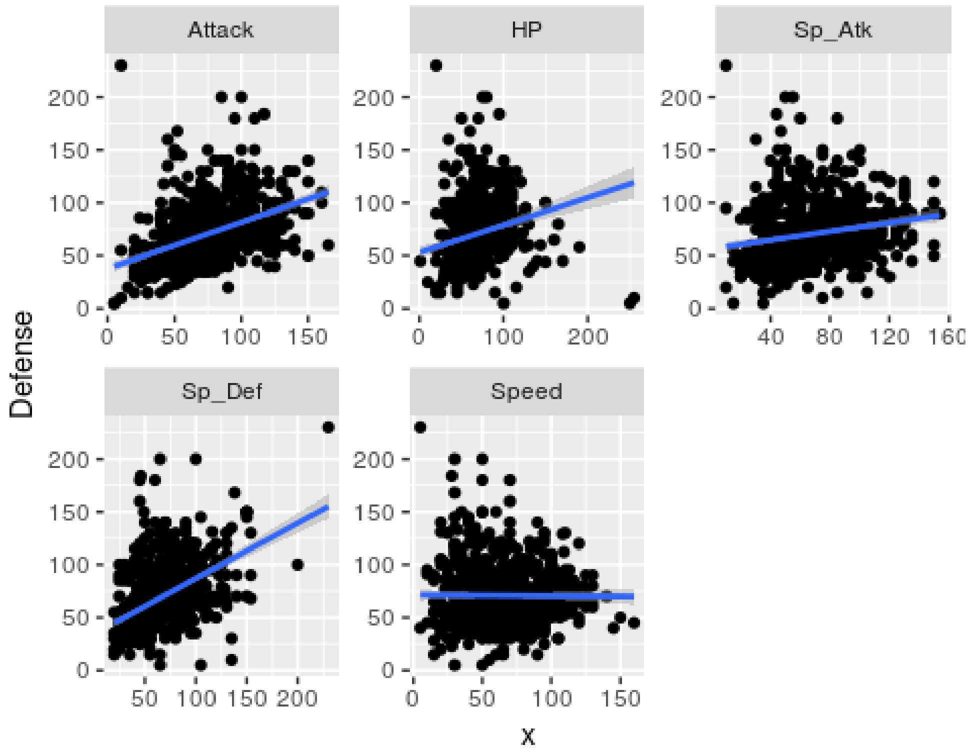
Let's go through each of the inclusive variables and its relationship to the other ones.

```
## ATTACK
response_vars_attack = pokemon %>% gather(xname, x, c(HP, Defense: Speed)) %>
% ggplot(aes(x=x, y=Attack)) + geom_point() + geom_smooth(method='lm') + face
t_wrap(~xname, scales="free", ncol=3)
response_vars_attack
```



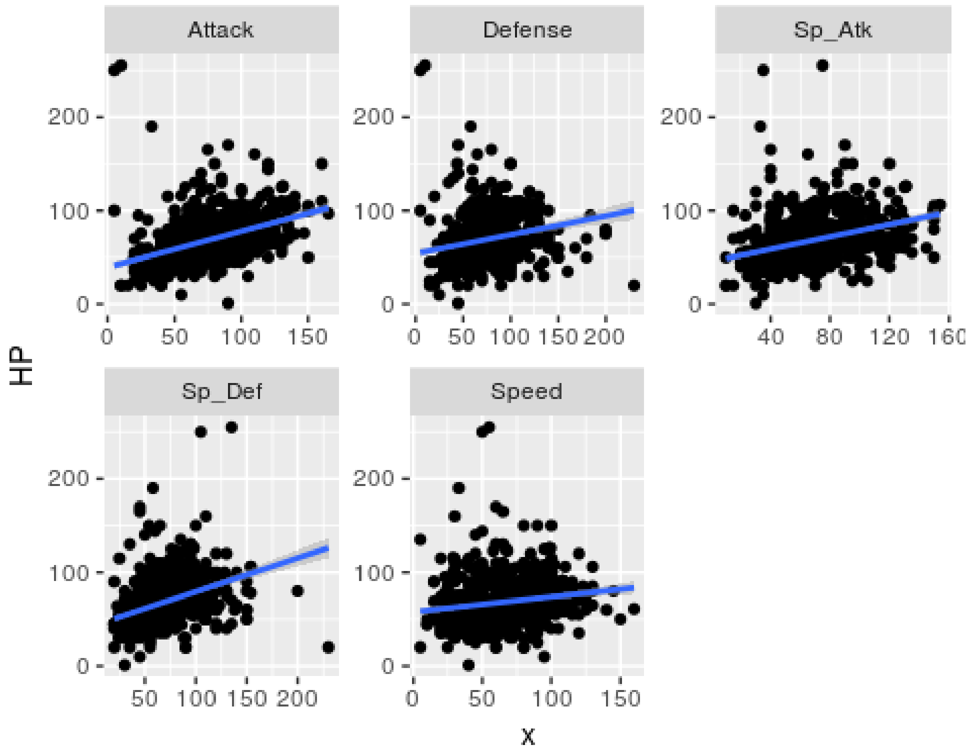
These are the scatter plots of each section compared to Attack. We can conclude that the variables, including Defense, HP, Special Attack, Special Defense, and Speed, are positively correlated to Attack. Among all the variables, Defense and HP have a stronger correlation to Attack, while Special Attack, Special Defense, and Speed showing relatively weak correlation. It is worth to mention that the Attack and Special Attack of a pokemon is not really related to each other which is a bit surprising.

```
## DEFENSE
response_vars_defense = pokemon %>% gather(xname, x, c(HP, Attack, Sp_Atk: Sp
eed)) %>% ggplot(aes(x=x, y=Defense)) + geom_point() + geom_smooth(method='lm
') + facet_wrap(~xname, scales="free", ncol=3)
response_vars_defense
```



These are the scatter plots of the variables to Defense. Unsurprisingly, Attack and Defense show a relatively strong relationship while all the others hold weak correlation. We can hardly tell that Speed to Defense has a positive correlation and it is doubtful until the test statistics applied.

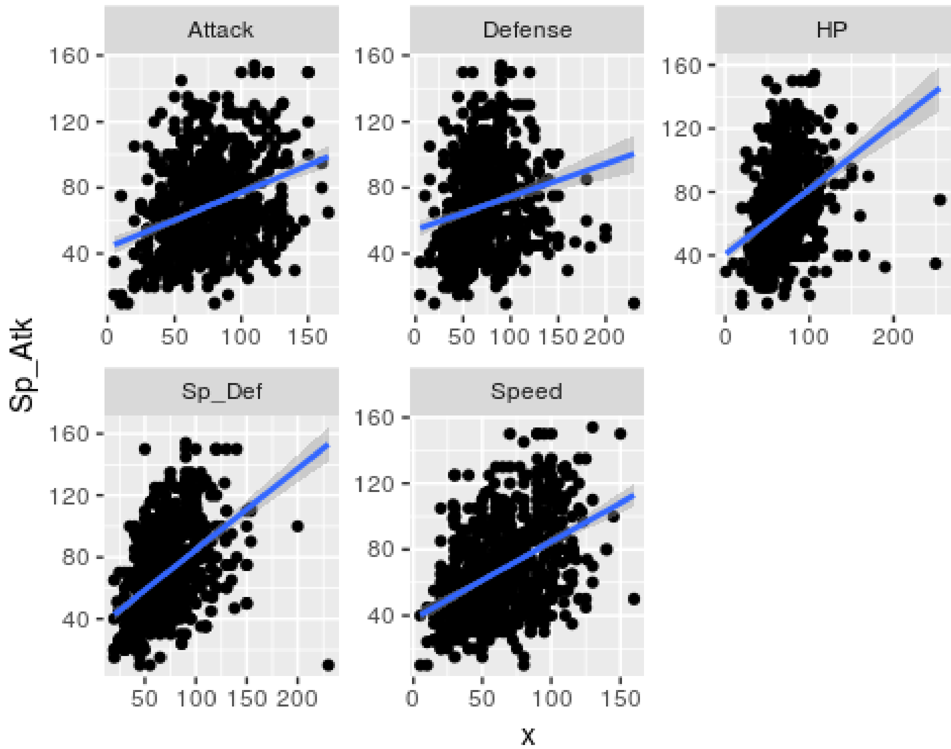
```
## HP
response_vars_hp = pokemon %>% gather(xname, x, c(Attack: Speed)) %>% ggplot(
aes(x=x, y=HP)) + geom_point() + geom_smooth(method='lm') + facet_wrap(~xname
, scales="free", ncol=3)
response_vars_hp
```



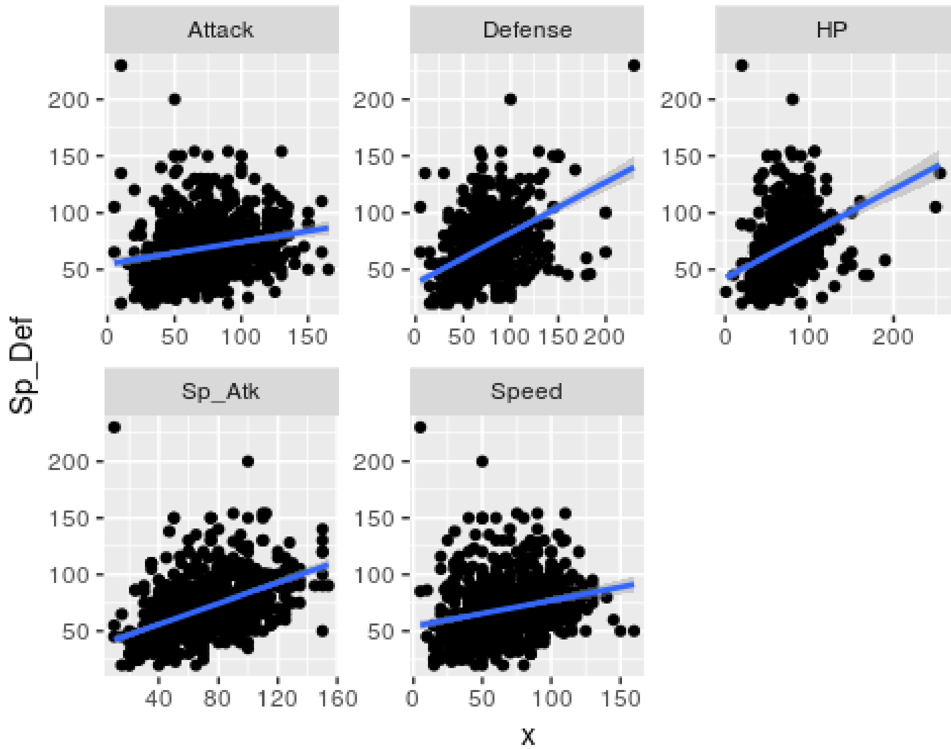
The graphs above are the scatter plot of HP to the other variables. HP has a stronger relationship to all the other fields compared to Attack and Defense we previously looked at, and all of them has a positive correlation even HP-Speed does not have a steep slope when we are drawing the line of best fit.

Since most of the relationships we have already talked about, we will skip the details for the rest of the groups and discuss them together.

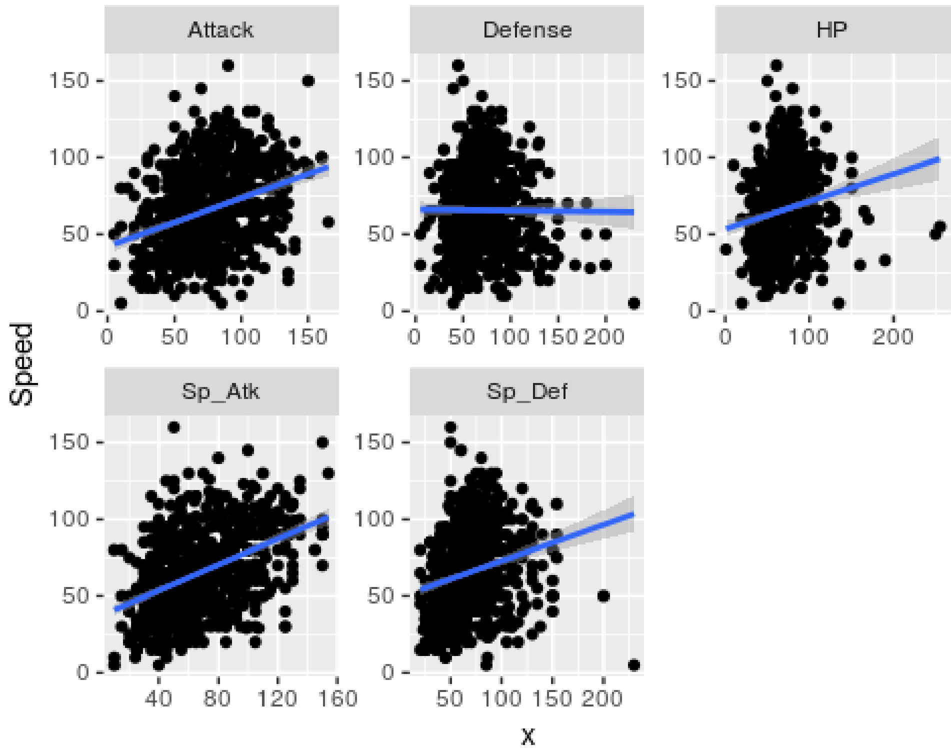
```
## SPECIAL ATTACK
response_vars_special_attack = pokemon %>% gather(xname, x, c(HP: Defense, Sp
_Def, Speed)) %>% ggplot(aes(x=x, y=Sp_Atk)) + geom_point() + geom_smooth(met
hod='lm') + facet_wrap(~xname, scales="free", ncol=3)
response_vars_special_attack
```




```
## SPECIAL DEFENSE
response_vars_special_defense = pokemon %>% gather(xname, x, c(HP: Sp_Atk, Sp
eed)) %>% ggplot(aes(x=x, y=Sp_Def)) + geom_point() + geom_smooth(method='lm'
) + facet_wrap(~xname, scales="free", ncol=3)
response_vars_special_defense
```




```
# SPEED
response_vars_speed = pokemon %>% gather(xname, x, c(HP: Sp_Def)) %>% ggplot(
aes(x=x, y=Speed)) + geom_point() + geom_smooth(method='lm') + facet_wrap(~xn
ame, scales="free", ncol=3)
response_vars_speed
```



The above 3 groups of scatter plots are already analyzed and we just put them here to confirm our thoughts from the previous groups, and there is no surprise we get these results.

We now know that the response variables are positively correlated from the graphs above, but they are not that clear. We will run a correlation matrix to see how they are correlated:

```
cor(updated_pokemon[,c ("HP", "Attack", "Defense", "Sp_Atk", "Sp_Def")])
```

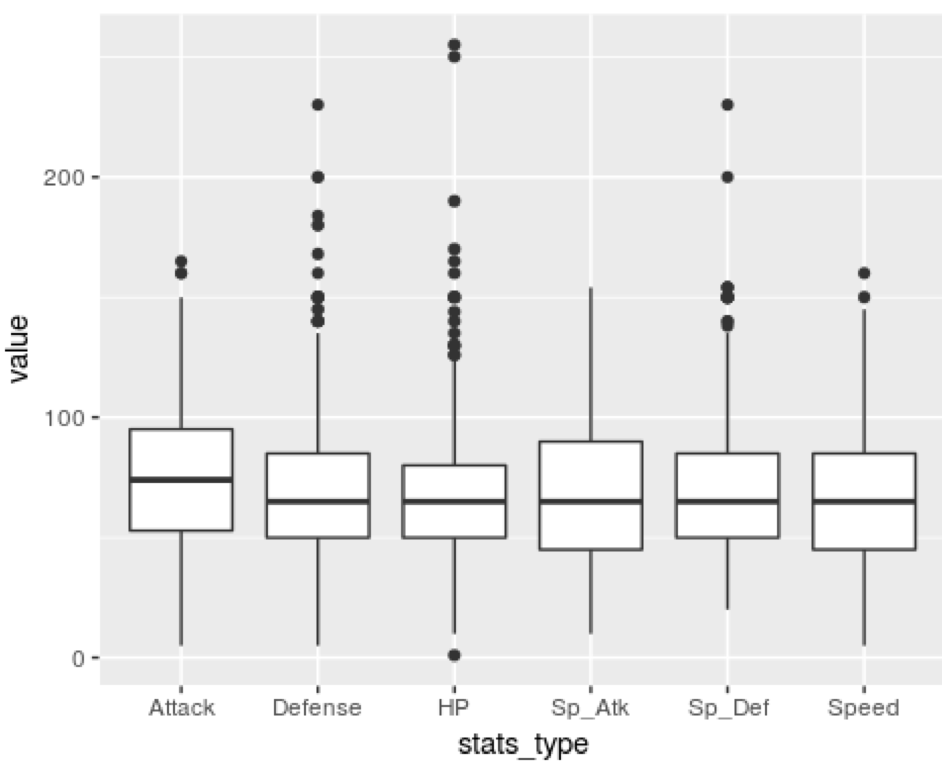
```
##                HP    Attack   Defense    Sp_Atk    Sp_Def
## HP      1.0000000 0.4316802 0.2288341 0.3686397 0.3760056
## Attack  0.4316802 1.0000000 0.4332328 0.3352049 0.2072106
## Defense 0.2288341 0.4332328 1.0000000 0.2025192 0.4839862
## Sp_Atk  0.3686397 0.3352049 0.2025192 1.0000000 0.4928608
## Sp_Def  0.3760056 0.2072106 0.4839862 0.4928608 1.0000000
```

From the matrix above, we can confirm that the response variables are positively correlated. The worse correlation is between Defense and Special Attack, which is 0.2025. This suggests that an increase in any of the particular performance of a pokemon will lead to an increase in another performance. This is normal because a stronger pokemon should be able to perform better in all the criteria, otherwise they will have more weaknesses.

Finally, we want to see if the means of these particular performances of pokemon are different from each other.

We will first gather the data into two columns, the first column with the type of performance of all 721 pokemon, the second column is the data associated with that. Then, a boxplot is run on that.

```
updated_pokemon_stats <- updated_pokemon[c(6:11)]
(updated_pokemon_stats %>% gather(stats_type, value, HP:Speed, na.rm=T) -> po
kemon_stats)

## # A tibble: 4,326 x 2
##    stats_type value
##    <chr>      <dbl>
##  1 HP            45
##  2 HP            60
##  3 HP            80
##  4 HP            39
##  5 HP            58
##  6 HP            78
##  7 HP            44
##  8 HP            59
##  9 HP            79
## 10 HP            45
## # ... with 4,316 more rows

ggplot(pokemon_stats, aes(x=stats_type, y=value)) + geom_boxplot()
```



From the plots above, we can see that there are 2 outliers in Attack and Speed respectively, and even more outliers in Defense, HP, and Special Defense. Most of the outliers spread on the top end to the majority (One data point lies on the bottom end of HP). Special Attack is the only field without any outlier but slightly skewed to the right. The median of each performance looks like they are on the same level. Since the dataset is relatively large, the distribution of these performances can be approximately normal by the Central Limit Theorem regardless of the skewness and outliers. Therefore, we will run Analysis of variances (ANOVA) on these data:

```
pokemon_stats.1=aov(value~stats_type,data=pokemon_stats)
summary(pokemon_stats.1)
```

```
##                Df  Sum Sq Mean Sq F value  Pr(>F)
## stats_type      5   34735    6947   8.927 1.9e-08 ***
## Residuals    4320 3361833     778
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

As p-value is smaller than significance level of 0.05, we can reject the null hypothesis and conclude that the mean of each performance is significantly different with each other, contrary to the observation we had on boxplot. To see the details of how they are different from each other, Tukey's test is run:

```
TukeyHSD(pokemon_stats.1)
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = value ~ stats_type, data = pokemon_stats)
##
## $stats_type
##                      diff        lwr         upr     p adj
## Defense-Attack -4.2052705  -8.394064 -0.01647723 0.0484425
## HP-Attack      -6.6338419 -10.822635 -2.44504866 0.0000949
## Sp_Atk-Attack  -6.2760055 -10.464799 -2.08721232 0.0002860
## Sp_Def-Attack  -5.7226075  -9.911401 -1.53381426 0.0013960
## Speed-Attack   -9.2995839 -13.488377 -5.11079068 0.0000000
## HP-Defense     -2.4285714  -6.617365  1.76022180 0.5632422
## Sp_Atk-Defense -2.0707351  -6.259528  2.11805814 0.7213904
## Sp_Def-Defense -1.5173370  -5.706130  2.67145620 0.9070044
## Speed-Defense  -5.0943135  -9.283107 -0.90552023 0.0070247
## Sp_Atk-HP       0.3578363  -3.830957  4.54662957 0.9998838
## Sp_Def-HP       0.9112344  -3.277559  5.10002762 0.9895866
## Speed-HP       -2.6657420  -6.854535  1.52305120 0.4564660
## Sp_Def-Sp_Atk   0.5533981  -3.635395  4.74219129 0.9990178
## Speed-Sp_Atk   -3.0235784  -7.212372  1.16521486 0.3098587
## Speed-Sp_Def   -3.5769764  -7.765770  0.61181681 0.1446005
```

From the output above, we only look at those with the p-value less than 0.05 significance level. It is discovered that attack is significantly different with all other performances, HP, defense, special attack, special defense and speed. It is not that surprising because every pokemon is trained to battle with other pokemon, so attack is a major thing to determine if a pokemon can win the battle by knocking out the others. Attack should dominate all other performances. If a pokemon has higher defense than other performance, it is rare for it to knock out other pokemon.

Besides, speed is also significantly different with defense, along with the observation that the p-value of the difference between mean of speed and attack is 0, we can say that the attack and defense value of a pokemon is way different with speed, which is also normal because speed is not a major value in a battle. Even for a fast-moving pokemon, the speed value will not necessarily similar to its attack value, instead, it is usually much lower.

Section 11 Conclusion

To conclude, we found out that all variables, except body style, will impact the total performance of pokemon. However, there might be total much information needed to gather, so we build a linear model for the purpose of predicting the value of total performance. We figured out that we only need 6 pieces of: whether it is a legendary pokemon, it has mega evolution and it has gender; its catch rate, first egg group and generation. As there are still a lot of undiscovered mysterious pokemon, we can use this model to estimate the total performance of those pokemon and decide if we should catch and train them.