Pokemon with stats

Raul Marinău

31 January 2021

Introduction

For the purpose of this project the dataset *Pokemon with stats* has been chosen from Kaggle. It features 721 pokemon from 6 generations (a pokemon generation is a grouping based on the Pokemon game where they first appeared) with their names, ids, types (1/2) and specific numeric stats: Total, HP, Attack, Defense, Sp.Atk (special attack), Sp.Def (special defense) and Speed. Each pokemon must have 1 or 2 types (Type1 and/or Type2).

The id of a pokemon can be determined like this: each evolution has an incremental id and each mega evolution has the same id as the original form. We can see this in table 1, where we have 2 starter pokemons (Bulbasaur and Charmander) and their evolutions.

Table 1: Peaking at the data

	id	Name	Type1	Type2	Total	HP	Attack	Defense	Sp.Atk	Sp.Def	Speed	Generation	Legendary
1	1	Bulbasaur	Grass	Poison	318	45	49	49	65	65	45	1	False
2	2	Ivysaur	Grass	Poison	405	60	62	63	80	80	60	1	False
3	3	Venusaur	Grass	Poison	525	80	82	83	100	100	80	1	False
4	3	Mega Venusaur	Grass	Poison	625	80	100	123	122	120	80	1	False
5	4	Charmander	Fire		309	39	52	43	60	50	65	1	False
6	5	Charmeleon	Fire		405	58	64	58	80	65	80	1	False
7	6	Charizard	Fire	Flying	534	78	84	78	109	85	100	1	False
8	6	Mega Charizard X	Fire	Dragon	634	78	130	111	130	85	100	1	False
9	6	Mega Charizard Y	Fire	Flying	634	78	104	78	159	115	100	1	False

Descriptive statistics

We start by reading the csv file that contains our data. We also trim out the name duplications (ex: *CharizardMega Charizard X* → *Mega Charizard X*).

```
data = read.csv("Pokemon.csv")
data$Name = sub(".*(Mega)", "Mega", data$Name)
```

We can filter through data to find all legendary dragon pokemon:

```
data[data$Legendary=="True" & (data$Type1=="Dragon" | data$Type2=="Dragon"),]
```

Table 2: Finding all the legendary dragon pokemon

	id	Name	Type1	Type2	Total	HP	Attack	Defense	Sp.Atk	Sp.Def	Speed	Generation	Legendary
418	380	Latias	Dragon	Psychic	600	80	80	90	110	130	110	3	True
419	380	Mega Latias	Dragon	Psychic	700	80	100	120	140	150	110	3	True
420	381	Latios	Dragon	Psychic	600	80	90	80	130	110	110	3	True
421	381	Mega Latios	Dragon	Psychic	700	80	130	100	160	120	110	3	True
426	384	Rayquaza	Dragon	Flying	680	105	150	90	150	90	95	3	True
427	384	Mega Rayquaza	Dragon	Flying	780	105	180	100	180	100	115	3	True
541	483	Dialga	Steel	Dragon	680	100	120	120	150	100	90	4	True
542	484	Palkia	Water	Dragon	680	90	120	100	150	120	100	4	True
545	487	GiratinaAltered Forme	Ghost	Dragon	680	150	100	120	100	120	90	4	True
546	487	GiratinaOrigin Forme	Ghost	Dragon	680	150	120	100	120	100	90	4	True
707	643	Reshiram	Dragon	Fire	680	100	120	100	150	120	90	5	True
708	644	Zekrom	Dragon	Electric	680	100	150	120	120	100	90	5	True
711	646	Kyurem	Dragon	Ice	660	125	130	90	130	90	95	5	True
712	646	KyuremBlack Kyurem	Dragon	Ice	700	125	170	100	120	90	95	5	True
713	646	KyuremWhite Kyurem	Dragon	Ice	700	125	120	90	170	100	95	5	True
795	718	Zygarde50% Forme	Dragon	Ground	600	108	100	121	81	95	95	6	True

We can also try to find out what the maximum stat values are for all pokemon:

```
sapply(data[5:11], max, na.rm = TRUE)

data[which.max(data$Total),]
data[which.max(data$HP),]
data[which.max(data$Attack),]
data[which.max(data$Defense),]
data[which.max(data$Sp.Atk),]
data[which.max(data$Sp.Def),]
data[which.max(data$Speed),]
```

Table 3: Pokemon with the highest stats

	id	Name	Type1	Type2	Total	HP	Attack	Defense	Sp.Atk	Sp.Def	Speed	Generation	Legendary
164	150	Mega Mewtwo X	Psychic	Fighting	780	106	190	100	154	100	130	1	True
262	242	Blissey	Normal		540	255	10	10	75	135	55	2	False
164	150	Mega Mewtwo X	Psychic	Fighting	780	106	190	100	154	100	130	1	True
225	208	Mega Steelix	Steel	Ground	610	75	125	230	55	95	30	2	False
165	150	Mega Mewtwo Y	Psychic		780	106	150	70	194	120	140	1	True
231	213	Shuckle	Bug	Rock	505	20	10	230	10	230	5	2	False
432	386	DeoxysSpeed Forme	Psychic		600	50	95	90	95	90	180	3	True

Finding out all the unique pokemon types:

```
unique(data$Type1)
# "Grass"     "Fire"      "Water"     "Bug"       "Normal"    "Poison"
# "Electric"  "Ground"    "Fairy"     "Fighting"  "Psychic"   "Rock"
# "Ghost" "Ice" "Dragon"    "Dark"      "Steel"     "Flying"

length(unique(data$Type1))   # 18

summary(data[5:11])
```

Table 4: Summary of the stats

	Total	HP	Attack	Defense	Sp.Atk	Sp.Def	Speed
X	Min. :180.0	Min. : 1.00	Min. : 5	Min. : 5.00	Min. : 10.00	Min. : 20.0	Min. : 5.00
X.1	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
X.2	Median :450.0	Median : 65.00	Median : 75	Median : 70.00	Median : 65.00	Median : 70.0	Median : 65.00
X.3	Mean :435.1	Mean : 69.26	Mean : 79	Mean : 73.84	Mean : 72.82	Mean : 71.9	Mean : 68.28
X.4	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
X.5	Max. :780.0	Max. :255.00	Max. :190	Max. :230.00	Max. :194.00	Max. :230.0	Max. :180.00

Plotting the distribution of the *Attack* attribute would result in 1:

```
ggplot(data, aes(x=Attack)) +
    geom_histogram(color="black", fill="white", binwidth = 10) +
    geom_vline(aes(xintercept=mean(Attack)), color="red",
        linetype="dashed", size=1)
```

By observing what's happening in the histogram 1 we can see that our distribution is right skewed but is pretty close to being a symmetric one. We can see that the count is very low for high attack values which we can attribute to rare evolutions or legendaries.

Let's look at how the same attribute ranges for different pokemon types 2:

It's very clear to see in 2 that *Dragon* type pokemon have a clear edge over all other types. Also besides *Fighting* type which obviously would score high in this stat, *Fire* type pokemons have a pretty low range of values, so they might prove superior when attacking compared to other elemental types.

Let's try to take a close look over how *Fire* pokemon would fare in a fight with *Water* type:

```
# we start of by picking 50 out of each group, type1 or type2
fire_pokemon = data[data$Type1=='Fire' | data$Type2=='Fire',]
fire_pokemon = head(fire_pokemon, n=50)
water_pokemon = data[data$Type1=='Water' | data$Type2=='Water',]
water_pokemon = head(water_pokemon, n=50)

# we built dataframes pitting their respective attack and defense
fire_vs_water = data.frame(x=fire_pokemon$Attack, y=water_pokemon$Defense)
water_vs_fire = data.frame(x=water_pokemon$Attack, y=fire_pokemon$Defense)
```

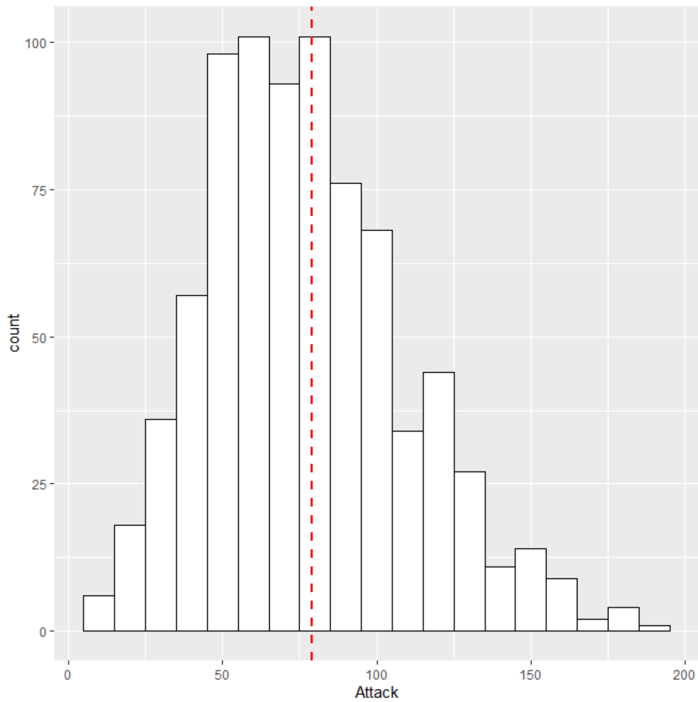

Figure 1: Attack distribution over all generations

```
# plot the points obtained in our dataframes
ggplot(data = fire_vs_water, aes(x=x, y=y)) +
    geom_point(size=2, col="red", shape=15) +
    geom_point(data=water_vs_fire, size=2, col="blue", shape=16) +
    labs(x="Attack", y="Defense")
```

Again we can observe in the scatterplot from figure 3 that *Fire* type pokemon have a preference towards higher attack but slightly lower defence then that of *Water* pokemon.

Hypothesis testing

We now want to see whether there are any significant changes between the *Attack* attribute of *Generation 1* and that of *Generation 2* for which we propose a paired sample test.

```
# pick top 50 Atk values of gen1
gen1 = data[data$Generation==1,]
gen1_atk = gen1[order(-gen1$Attack),]
gen1_atk_top50 = head(gen1_atk, n=50)$Attack
```

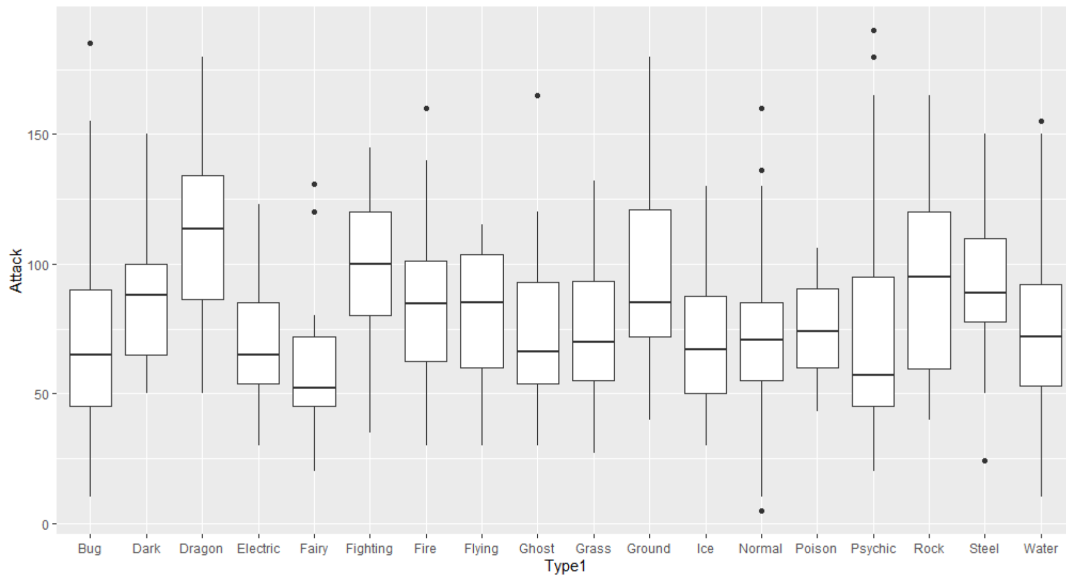

Figure 2: Attack by type

```
# pick top 50 Atk values of gen2
gen2 = data[data$Generation==2,]
gen2_atk = gen2[order(-gen2$Attack),]
gen2_atk_top50 = head(gen2_atk, n=50)$Attack

# plot both sets of points and the connecting line
df_gen1 = data.frame(y=gen1_atk_top50)
ggplot(df_gen1, aes(sample=y)) + stat_qq() + stat_qq_line() -> p1
df_gen2 = data.frame(y=gen2_atk_top50)
ggplot(df_gen2, aes(sample=y)) + stat_qq() + stat_qq_line() -> p2
require(gridExtra)
grid.arrange(p1, p2, ncol=2)
```

Although we can already observe in figure 4 that we don't really have a normal distribution, we can make sure by computing the Shapiro-Wilk normality test:

```
shapiro.test(df_gen1$y) # p-value = 2.246e-05
shapiro.test(df_gen2$y) # p-value = 3.198e-06
```

Both of our p-values are much less than 0.05 significance level so we will use the Wilcox test for comparing our paired data that has no normal distribution.

```
wilcox.test(gen1_atk_top50, gen2_atk_top50, paired = TRUE) # p-value 8.444e-10
```

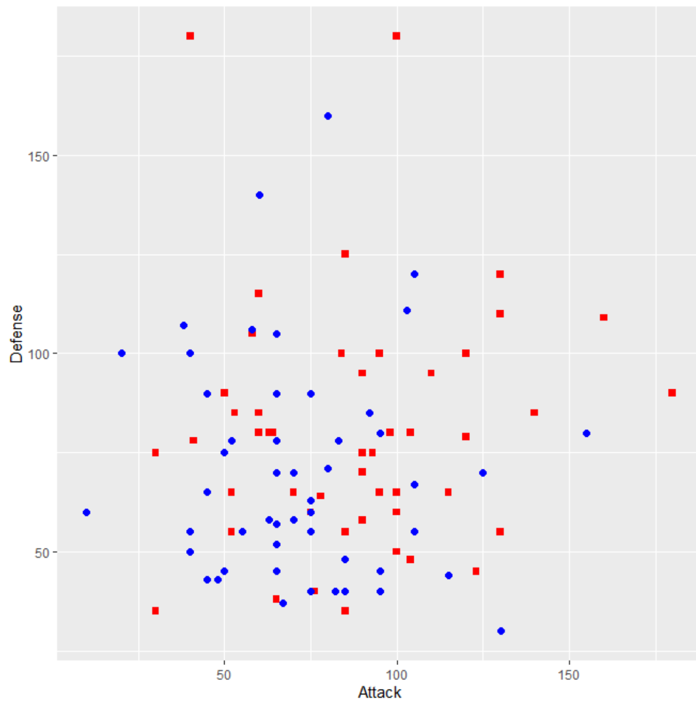

Figure 3: Fire vs Water

Again we obtain a p-value that's a lot lower than 0.05, but this time we can conclude that median of the *Attack* attribute is significantly different between generation 1 and 2.

Going back to our previous observations about *Fire* type pokemon, let's test if they make a better choice for a starter pokemon. A starter pokemon is the first one you get to pick in all iterations of the game from a set of 3: a *Water* type, a *Grass* type and a *Fire* type.

First we check what their population is out of the 800 total, we then make $n = 150$ observations from the top pokemon (those with higher *Total* stat). We can thus formulate the problem as a one-proportion Z-Test.

```
data[data$Type1=="Water" | data$Type2=="Water",] -> data_water
# nrow 126
data[data$Type1=="Fire" | data$Type2=="Fire",]   -> data_fire
# nrow 64
data[data$Type1=="Grass" | data$Type2=="Grass",] -> data_grass
# nrow 95

# n=800 total pokemon
# p_w = 126*100/800 = 15.75%
```

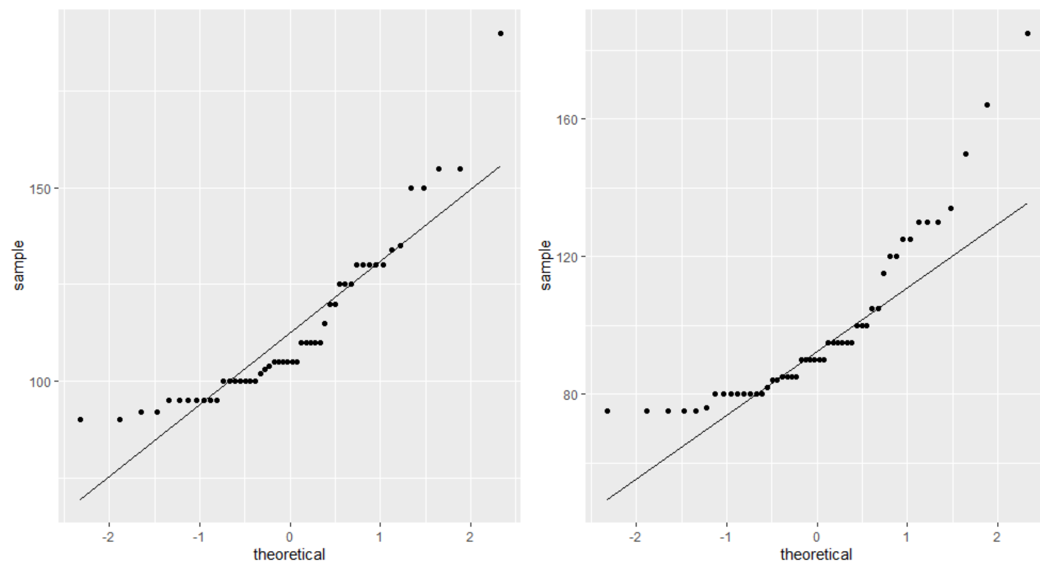

Figure 4: Gen1 vs Gen2

```
# p_f = 64*100/800   = 8%
# p_e = 95*100/800   = 11.875%

top = head(data[order(-data$Total),], n=150)

# observations
top[top$Type1=="Water" | top$Type2=="Water",] -> top_water
# nrow 20
top[top$Type1=="Fire" | top$Type2=="Fire",]    -> top_fire
# nrow 21
top[top$Type1=="Grass" | top$Type2=="Grass",] -> top_grass
# nrow 9

prop.test(x = 20, n = 150, p = 0.1575, correct = FALSE)
prop.test(x = 21, n = 150, p = 0.08,   correct = FALSE)
prop.test(x = 9,  n = 150, p = 0.1187, correct = FALSE)
```

The p-values obtained are as follows:

- water: $p - value = 0.4165$
- fire: $p - value = 0.0067$
- grass: $p - value = 0.0262$

So, on a 95% confidence interval all 3 types confirm our hypothesis of being good picks if we want to end up with a pokemon in the top 150 (assuming they evolve up to that point) but because *Fire* is the lowest of the bunch, we can assume it is the best pick.

Linear Regression

In order to build a linear regression model let's analyze the correlations between the features of our dataset.

```
library(ggcorrplot)
attributes_dtf = data[,5:11]
corr = round(cor(attributes_dtf), 1)
ggcorrplot(corr, hc.order = TRUE, lab = TRUE)
```

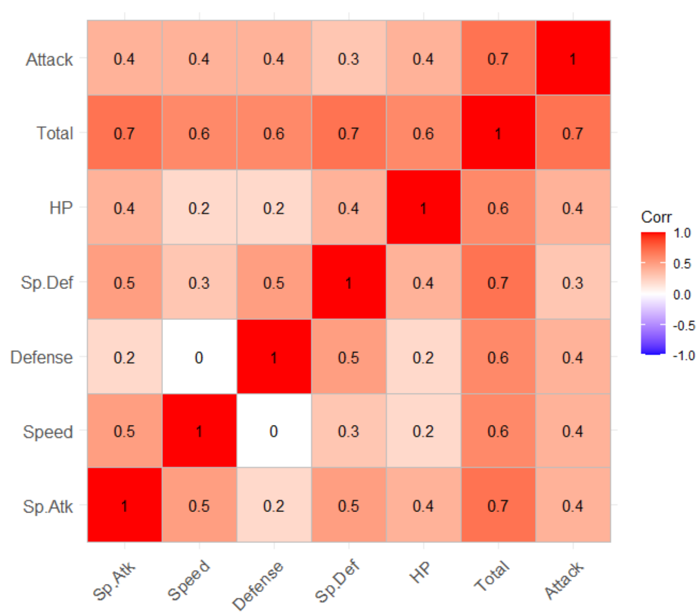

Figure 5: Heatmap of the correlation matrix

We will start by building a simple linear regression model based on the *Attack* feature as it proves to be one of the variables with the highest correlation towards the *Total*.

```
# we first split our data in 75% train and 25% test
sample_size = floor(0.75 * nrow(attributes_dtf))

set.seed(123)
```

```r
train_ind = sample(seq_len(nrow(attributes_dtf)), size=sample_size)

train = attributes_dtf[train_ind, ]
test = attributes_dtf[-train_ind, ]

# we build the simple linear regression model based on the train data
model = lm(Total ~ Attack, data=train)

# plot the points and the prediction function of our model
plot(attributes_dtf$Attack, attributes_dtf$Total, main = "Simple Linear Regressi
     xlab = "Attack", ylab = "Total",
     pch = 19, frame = FALSE)
abline(model, col = "blue")
```

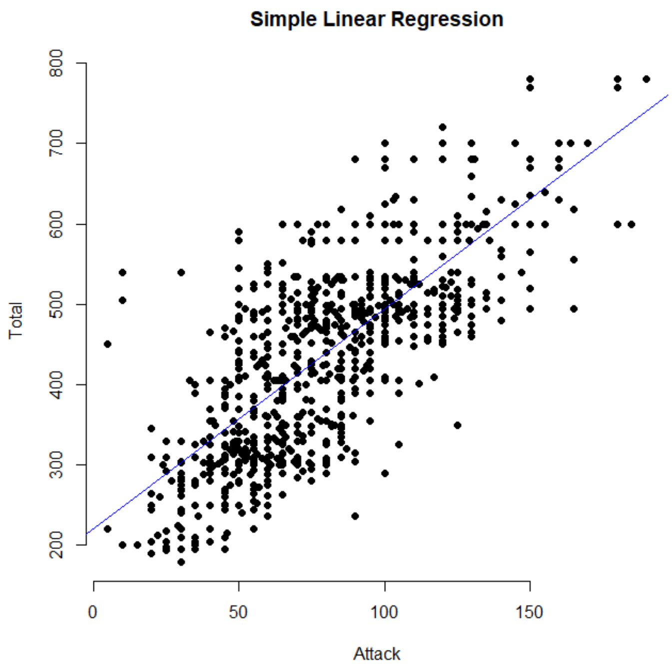

Figure 6: Simple Linear Regression

Our model obtained $R^2 = 0.558$ and also we can observe the correlation of our predictions:

```r
prediction = predict(model, test)

actual_predict_dtf = data.frame(cbind(actuals=test$Total, predicteds=prediction)
cor(actual_predict_dtf) # 0.70
```

```
head(actual_predict_dtf)
#    actuals predicteds
# 1      318   354.7236
# 3      525   444.8306
# 7      534   450.2916
# 9      634   504.9019
# 15     205   275.5386
# 17     195   316.4963
```

So, as expected our single feature isn't very accurate. Let's see what we can obtain using multiple features.

```
model = lm(Total ~ Sp.Atk + Sp.Def + Attack + Defense, data=train)
summary(model)
# this already proves to be a very good model with R^2=0.91

prediction = predict(model, test)

actual_predict_dtf = data.frame(cbind(actuals=test$Total, predicteds=prediction))
cor(actual_predict_dtf)

head(actual_predict_dtf, n=10)
```

Table 5: Results of multiple linear regression

	actuals	predicteds
1	318.00	352.98
3	525.00	525.21
7	534.00	515.06
9	634.00	657.34
15	205.00	199.19
17	195.00	192.32
22	349.00	332.07
25	253.00	257.25
27	262.00	262.39
28	442.00	416.54

This model proves to be much more accurate, so it would be easy to use in order to predict new *Total* values based only on *Attack* and *Defense* (+ specials) attributes of our pokemon.