Predicting the Winning Pokemon

Dale Chen-Song

30/06/2020

Figure 1: Pokemon

1. Introduction

Pokemon, developed by GameFreak and published by Nintendo, is a global phenomenon that has spanned for almost 25 years starting in 1996, starting with video games and expanding to TV series, card games, toys and so on. There are more than 30 core games in the franchise with even more spin-off games. Pokemon boasts the second best-selling video game franchise and the largest media franchise. With a new Pokemon game released in November 15, 2019, there are now almost 900 different Pokemon, not including different forms.

The main goal of Pokemon is to capture these monsters and use them in battle to fight against other people and their Pokemon. However, in core game Pokemon, there is a surprisingly complex combat system. There are many ** combat mechanics** involved, such as team composition, moves, items, abilities, type weakness, and so on, that all this can confuse those who want to start competitive Pokemon battling. Where do we start? In this report, we'll focus simply on the Pokemon and not other combat mechanics. Which Pokemon wins the most fights? Since there are so many Pokemon, we want to narrow down which Pokemon we want to use, since we can only have 6 Pokemon on a team. Can we determine the winner of a battle, just by looking at a Pokemon stats? We'll employ machine learning to help us predict which Pokemon in the battle will ultimately become the victor.

(Note: anything that is bold is a link for more information)

2. Dataset Summary

We obtain our datasets from **Kaggle**; Pokemon- Weedle's Cave.

```
# Obtain Dataset
# Link: https://www.kaggle.com/terminus7/pokemon-challenge

# Download and create database
pokemon <- fread("pokemon.csv")
combat <-fread("combat.csv")
test <-fread("tests.csv")

# Tidy data
names(pokemon)[1] <- "Number"
```

There are three datasets. First one contains all the information on each Pokemon. The second contains simulated Pokemon combat with ID of pokemon and winner. The third contains test combats to predict on.

2.1 Pokemon Dataset

The first dataset, pokemon, contains 800 Pokemon (which include different forms) with all their characteristics.
This dataset has 12 categories:

Number: Pokemon ID numbered by order by National Pokedex.
Name: Name of the Pokemon.
Type 1: Elemental Property of a Pokemon.
Type 2: Second Elemental Property of a Pokemon.

The next 6 categories define the stats of the Pokemon.
HP(Hit Point): How much damage a Pokemon can receive before fainting.
Attack: How much damage a Pokemon deals with a physical move.
Defense: How much damage a Pokemon receives when it is hit with a physical move.
Sp. Atk (Special Attack): How much damage a Pokemon deals with a special move.
Sp. Def (Special Defense): How much damage a Pokemon receives when it is hit with a special move.
Speed: Determines order of Pokemon that can act in battle.

The next 2 categories define a grouping for the Pokemon.
Generation: Grouping for pokemon based on when they are released. Normally when a new generation occurs when a new set of core games are introduced.
Legendary: Grouping for incredibly rare and often very powerful Pokemon.

```
# Data Observation
head(pokemon)
```

```
##    Number           Name Type 1 Type 2 HP Attack Defense Sp. Atk Sp. Def Speed
## 1:      1      Bulbasaur  Grass Poison 45     49      49      65      65    45
## 2:      2        Ivysaur  Grass Poison 60     62      63      80      80    60
## 3:      3       Venusaur  Grass Poison 80     82      83     100     100    80
## 4:      4 Mega Venusaur  Grass Poison 80    100     123     122     120    80
## 5:      5     Charmander   Fire        39     52      43      60      50    65
## 6:      6     Charmeleon   Fire        58     64      58      80      65    80
##    Generation Legendary
## 1:           1     FALSE
## 2:           1     FALSE
## 3:           1     FALSE
## 4:           1     FALSE
## 5:           1     FALSE
## 6:           1     FALSE
```

```
dim(pokemon)
```

```
## [1] 800  12
```

Pokemon Groupings

Mega Evolution: Wait, what's with the **Mega** in front of Mega Venasaur? Is this a different Pokemon compared to Venasaur? Actually, it turns out Mega in front of a Pokemon name means that it is a Mega Evolution of a Pokemon, which, in general, is a more powerful form of the base Pokemon. What makes them special? You need to have a certain item to transform these Pokemon into their Mega Evolution and you can only have one Mega Pokemon in a team. I have added **Primal Pokemon** within this list, because they behave almost like Mega Pokemon, however there is no limitations on how many Primal Pokemon you can have.

```
# Mega
mega <- pokemon %>% filter(str_detect(Name, "Mega ")==TRUE | str_detect(Name, "Primal")==TRUE)
head(mega)
```

```
##   Number             Name Type 1 Type 2 HP Attack Defense Sp. Atk Sp. Def Speed
## 1      4    Mega Venusaur  Grass Poison 80    100     123     122     120    80
## 2      8 Mega Charizard X   Fire Dragon 78    130     111     130      85   100
## 3      9 Mega Charizard Y   Fire Flying 78    104      78     159     115   100
## 4     13   Mega Blastoise  Water        79    103     120     135     115    78
## 5     20    Mega Beedrill    Bug Poison 65    150      40      15      80   145
## 6     24     Mega Pidgeot Normal Flying 83     80      80     135      80   121
##   Generation Legendary
## 1          1     FALSE
## 2          1     FALSE
## 3          1     FALSE
## 4          1     FALSE
## 5          1     FALSE
## 6          1     FALSE
```

```
mega %>% count() %>% pull()
```

```
## [1] 50
```

A closer analysis of this data shows that there are 50 different Mega Pokemon.

Legendary: Let's look at another grouping of Pokemon, **Legendary**. These are considered to be a more powerful and rare Pokemon in the Pokemon world.

```r
# Legendary
legendary <- pokemon %>% filter(Legendary == TRUE)
head(legendary)
```

```
##   Number           Name   Type 1   Type 2  HP Attack Defense Sp. Atk Sp. Def
## 1    157       Articuno      Ice   Flying  90     85     100      95     125
## 2    158         Zapdos Electric   Flying  90     90      85     125      90
## 3    159        Moltres     Fire   Flying  90    100      90     125      85
## 4    163         Mewtwo  Psychic          106    110      90     154      90
## 5    164 Mega Mewtwo X  Psychic Fighting 106    190     100     154     100
## 6    165 Mega Mewtwo Y  Psychic          106    150      70     194     120
##   Speed Generation Legendary
## 1    85          1      TRUE
## 2   100          1      TRUE
## 3    90          1      TRUE
## 4   130          1      TRUE
## 5   130          1      TRUE
## 6   140          1      TRUE
```

```r
legendary %>% count() %>% pull()
```

```
## [1] 65
```

We see that there are 65 Legendary Pokemon.

Mega Legendary: If we take a closer look at Legendary group, we can see that there are Mega (and Primal) in front of some of these names!

```r
# Mega Legendary
mega_legendary <- mega %>% filter(Legendary==TRUE)
head(mega_legendary)
```

```
##   Number            Name  Type 1   Type 2  HP Attack Defense Sp. Atk Sp. Def
## 1    164   Mega Mewtwo X Psychic Fighting 106    190     100     154     100
## 2    165   Mega Mewtwo Y Psychic          106    150      70     194     120
## 3    419     Mega Latias  Dragon  Psychic  80    100     120     140     150
## 4    421     Mega Latios  Dragon  Psychic  80    130     100     160     120
## 5    423   Primal Kyogre   Water          100    150      90     180     160
## 6    425 Primal Groudon  Ground     Fire 100    180     160     150      90
##   Speed Generation Legendary
## 1   130          1      TRUE
## 2   140          1      TRUE
## 3   110          3      TRUE
## 4   110          3      TRUE
## 5    90          3      TRUE
## 6    90          3      TRUE
```

```r
mega_legendary %>% count() %>% pull()
```

```
## [1] 8
```

If Mega versions of more powerful version of the base Pokemon, then this would imply that these 8 Mega Legendary Pokemon are stronger versions than their respective Legendary Pokemon, which are already powerful and rare Pokemon. Maybe we should take note of these Pokemon.

Others: There are Pokemon that appear more than once in this list, such as Rotom and Mega Pokemon. Such Pokemon are considered to have **different forms**. They are distinct enough that they could be considered different Pokemon in combat. However, the only issue is that some competitive Pokemon battles only allow one Pokemon of the same name, to prevent people from having the same Pokemon dominate in a battle. Without these different forms, we have a total of 721 unique Pokemon. We'll take a note of this, however this shouldn't affect us.

```
# Rotom
pokemon[grep("Rotom", Name), ]
```

```
##    Number        Name   Type 1 Type 2 HP Attack Defense Sp. Atk Sp. Def Speed
## 1:    532       Rotom Electric  Ghost 50     50      77      95      77    91
## 2:    533  Heat Rotom Electric   Fire 50     65     107     105     107    86
## 3:    534  Wash Rotom Electric  Water 50     65     107     105     107    86
## 4:    535 Frost Rotom Electric    Ice 50     65     107     105     107    86
## 5:    536   Fan Rotom Electric Flying 50     65     107     105     107    86
## 6:    537   Mow Rotom Electric  Grass 50     65     107     105     107    86
##    Generation Legendary
## 1:          4     FALSE
## 2:          4     FALSE
## 3:          4     FALSE
## 4:          4     FALSE
## 5:          4     FALSE
## 6:          4     FALSE
```

There is another group of Pokemon called **Mythical**, which are considered even more rare than Legendary Pokemon (though not necessarily more powerful). Prior to Generation 5, there was no distinction between Legendary and Mythical Pokemon. Some Legendary Pokemon in this dataset are incorrectly labelled as Legendary when they should be Mythical, and some Mythical aren't labelled at all. I'll leave the pokemon database like this for now, but will consider about this point later on, if necessary.

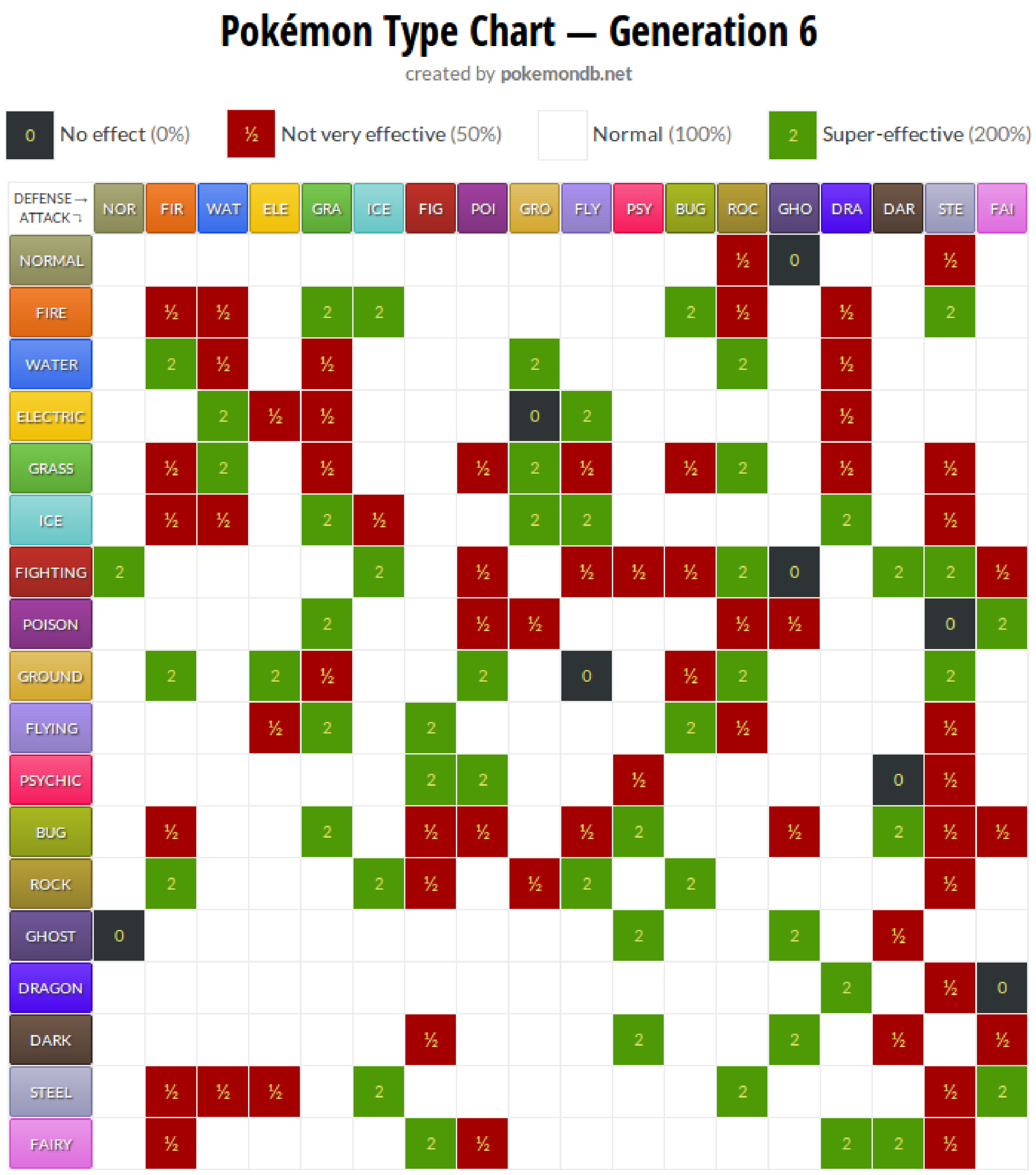
Pokemon Types

```
# Number of Types
n_distinct(pokemon$`Type 1`)
```

```
## [1] 18
```

There are 18 different Pokemon **types**. Each type has its own strength and weakness (even immunity) against other types. An example is that a Fire type is super-effective against a Grass type, however a Fire type is not very effective against a Water type. I have included a Type Chart reference down below.

We can also see that certain Pokemon have one type, like Squirtle, simply being a Water type Pokemon, while others have two types, like Bulbasaur, a Grass/Poison Pokemon (the order of two types do not matter). Within the 800 Pokemon, there are 386 having a single type.



Pokémon Type Chart — Generation 6

created by **pokemondb.net**

0 No effect (0%) ½ Not very effective (50%) Normal (100%) 2 Super-effective (200%)

DEFENSE → ATTACK ↴	NOR	FIR	WAT	ELE	GRA	ICE	FIG	POI	GRO	FLY	PSY	BUG	ROC	GHO	DRA	DAR	STE	FAI
NORMAL													½	0			½	
FIRE		½	½		2	2						2	½		½		2	
WATER		2	½		½				2				2		½			
ELECTRIC			2	½	½				0	2					½			
GRASS		½	2		½			½	2	½		½	2		½		½	
ICE		½	½		2	½			2	2					2		½	
FIGHTING	2					2		½		½	½	½	2	0		2	2	½
POISON					2			½	½				½	½			0	2
GROUND		2		2	½			2		0		½	2				2	
FLYING				½	2		2					2	½				½	
PSYCHIC							2	2			½					0	½	
BUG		½			2		½	½		½	2			½		2	½	½
ROCK		2				2	½		½	2		2					½	
GHOST	0										2			2		½		
DRAGON															2		½	0
DARK							½				2			2		½		½
STEEL		½	½	½		2							2				½	2
FAIRY		½					2	½							2	2	½	

Figure 2: Pokemon Type Chart

```r
# Amount of Pokemon with only one Types
pokemon %>% group_by(`Type 2`) %>% filter(str_detect(`Type 2`, "") == FALSE) %>%
  summarize(tot=n())%>% pull()
```

```
## [1] 386
```

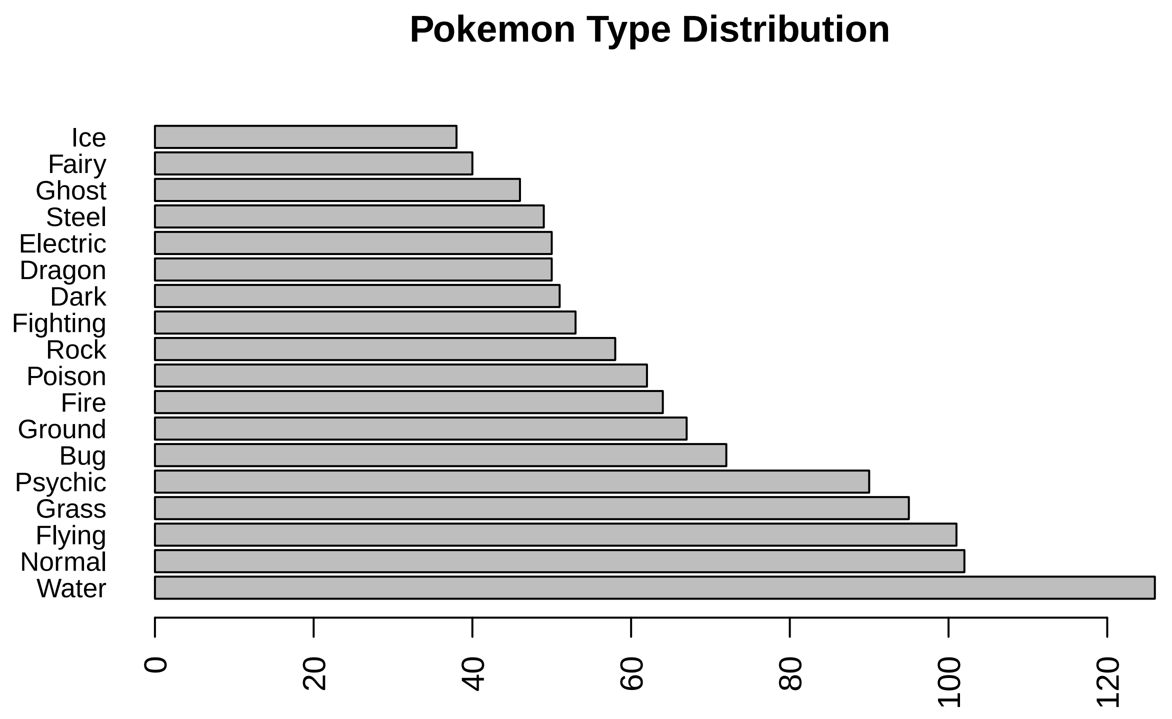
When Pokemon have two types, they combine the weaknesses and strengths of those types together. For example, if the opposing Pokemon is Water/Grass, the Fire type does neutral damage, since the super-effective against cancels out with the not very effective. This is pretty complicated and may not make immediate sense to those unfamiliar to Pokemon, however this creates a more diverse battle system where a user will try to match up their Pokemon against an opponent in which the user has the type advantage. I have provided a link to a **dual-type chart** for those who are interested, however we won't delve too deep into dual types here.

Type Distribution: So let's see how many Pokemon there are for each type, maybe we can figure out a strategy to use to counter the more dominant types. I decided to merge the two Type lists together, since it doesn't quite matter whether they are the primary or secondary types of a Pokemon and the weaknesses and strengths are combined when they are dual type Pokemon.

```r
# Type Distribution
# Type 1 total
type_1<-pokemon %>% group_by(`Type 1`) %>% summarize(Total_1=n())
names(type_1)[1]<- "Type"
# Type 2 total
type_2<-pokemon %>% group_by(`Type 2`) %>%
  filter(str_detect(`Type 2`, "") == TRUE) %>%summarize(Total_2=n())
names(type_2)[1]<- "Type"
# Join type together
type<-inner_join(type_1,type_2)
```

```
## Joining, by = "Type"
```

```r
type$Total<- type$Total_1+type$Total_2
type <- type %>% arrange(desc(Total))

# Graph for Pokemon Type Distribution
par(las =2)
barplot(type$Total, names.arg = type$Type, main="Pokemon Type Distribution", cex.names= 0.8, horiz=TRUE
```



We can see that the 3 most abundant types are Water, Normal and Flying. So, from this observation, we should consider Pokemon that are strong against these abundant types. From the chart above, we can see that the weakness for Water is Electric and Grass. For Normal, it's only Fighting. Finally, for Flying, the weaknesses are Ice, Rock and Electric. We can see that maybe being an Electric type is an advantage.

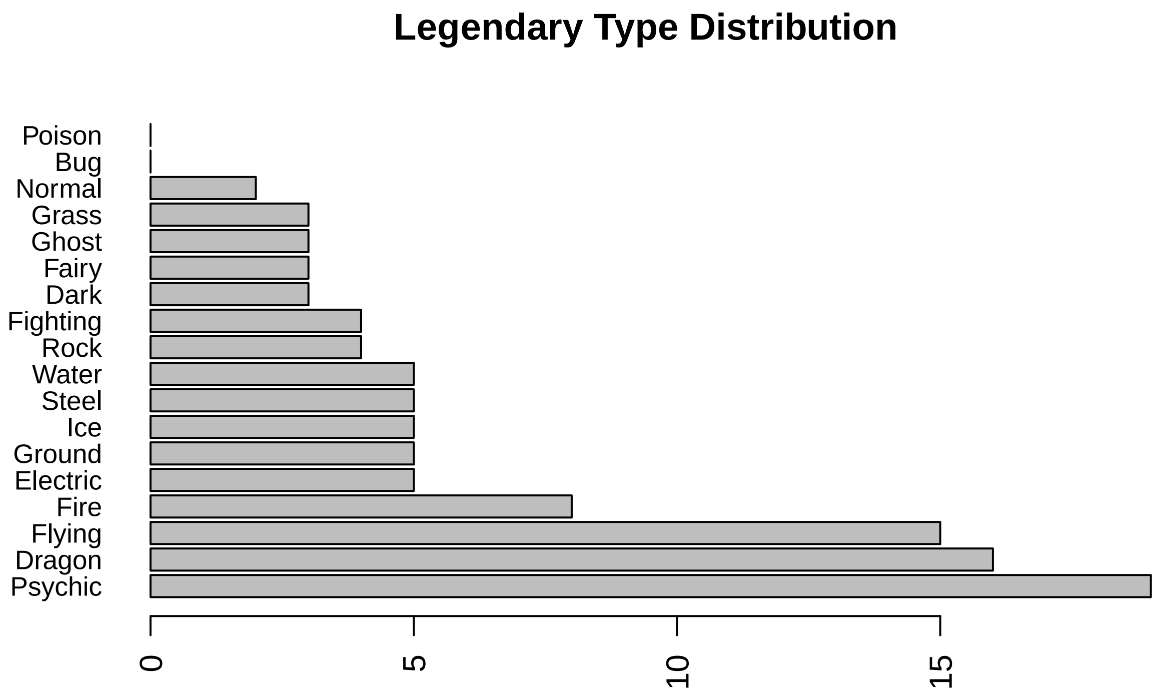
The 3 least abundant types are Ice, Fairy, and Ghosts. This actually makes sense for Fairy types, since they were newly introduced much later during Generation 6 (the newest Generation), as GameFreak wanted to try and counter Dragon types for being too powerful. Therefore, as a newly introduced type, there would be less Pokemon being Fairy type.

Legendary Type Distribution We can observe the distribution of Types within the group for Legendary.

```
# Legendary Type Distribution
# Type 1 total
leg_type_1<-legendary %>% group_by(`Type 1`) %>% summarize(Total_1=n())
names(leg_type_1)[1]<- "Type"

# Type 2 total
leg_type_2<-legendary %>% group_by(`Type 2`) %>%
  filter(str_detect(`Type 2`, "") == TRUE) %>%summarize(Total_2=n())
names(leg_type_2)[1]<- "Type"

# Join type together
leg_type<-full_join(leg_type_1, leg_type_2)
```

```
## Joining, by = "Type"
```

```
leg_type<- leg_type %>% add_row(Type = "Bug")
leg_type<- leg_type %>% add_row(Type = "Poison")
leg_type[is.na(leg_type)]<-0
leg_type$Total<- leg_type$Total_1+leg_type$Total_2
leg_type <- leg_type %>% arrange(desc(Total))

# Graph for Legendary Type Distribution
par(las =2)
barplot(leg_type$Total, names.arg = leg_type$Type, main="Legendary Type Distribution", cex.names= 0.8, 
```



Within the Legendary group, there 3 most abundant types are Psychic, Dragon, and Flying. Thus to counter these powerful Pokemon, we can consider Pokemon strong against these types. Bug, Dark, Ghost against Psychic. Dragon, Ice, and Fairy against Dragon. Ice, Rock, and Electric for Flying.

Turns out that there are no Poison nor Bug type Legendary Pokemon!

Pokemon Stats

Let's take a look at the Pokemon Stats. We should probably take note of the Pokemon with the highest Stats, since this seems to be how a Pokemon can win in a battle.

BST: One common method to immediately know the power of a Pokemon is to look at their Base Stat Total (BST). This is just all of the Pokemon Stats added up together. I have added a BST column in the Pokemon dataset, so we can more easily work with the BST.

```r
# Base Stat Total (BST)
pokemon$BST<- pokemon$HP + pokemon$Attack + pokemon$Defense +
  pokemon$`Sp. Atk` + pokemon$`Sp. Def` + pokemon$Speed
head(pokemon)
```

```
##    Number          Name Type 1 Type 2 HP Attack Defense Sp. Atk Sp. Def Speed
## 1:      1     Bulbasaur  Grass Poison 45     49      49      65      65    45
## 2:      2       Ivysaur  Grass Poison 60     62      63      80      80    60
## 3:      3      Venusaur  Grass Poison 80     82      83     100     100    80
## 4:      4 Mega Venusaur  Grass Poison 80    100     123     122     120    80
## 5:      5    Charmander   Fire        39     52      43      60      50    65
## 6:      6    Charmeleon   Fire        58     64      58      80      65    80
##    Generation Legendary BST
## 1:          1     FALSE 318
## 2:          1     FALSE 405
## 3:          1     FALSE 525
## 4:          1     FALSE 625
## 5:          1     FALSE 309
## 6:          1     FALSE 405
```

We can see immediately that Bulbasaur has lower BST than Ivysaur. Ivysaur has lower BST than Venasaur. Finally, Venasaur has lower BST than Mega Evolution. This makes sense as these Bulbasaur evolves into Ivysaur, which evolves into Venasaur, which then Mega evolves into Mega Venasaur. Evolution in Pokemon is the process for when a Pokemon changes into a different and more powerful species of Pokemon with increased Stats and some even change types. So, in general, fully evolved Pokemon have an edge against those that aren't fully evolved.

(Only one Pokemon's evolution line doesn't have increased Stats, and that is Scyther to Scizor, where the BST for both is 500, but the Stats are rearranged)

```r
# Scyther and Scizor
pokemon[c(133,229),]
```

```
##    Number    Name Type 1 Type 2 HP Attack Defense Sp. Atk Sp. Def Speed
## 1:    133 Scyther    Bug Flying 70    110      80      55      80   105
## 2:    229  Scizor    Bug  Steel 70    130     100      55      80    65
##    Generation Legendary BST
## 1:          1     FALSE 500
## 2:          2     FALSE 500
```

From Mega Venasaur BST, we can observe that Mega pokemon have higher BST than their base counterpart. This is the case with all Mega Pokemon (except for Alakazam) as they have 100 more BST than their base counterpart, showing that they should be more powerful.

```r
# Mega BST
mega$BST<- mega$HP + mega$Attack + mega$Defense +
  mega$`Sp. Atk` + mega$`Sp. Def` + mega$Speed
# Getting rid of duplicate Mega Evolution (Charizard and Mewtwo have 2 Mega Evolution)
nonmega<-mega[-c(3,15),]
# Obtain base form
nonmega<-nonmega$Number-1
nonmega<-pokemon %>% filter(pokemon$Number %in% nonmega)
# Join both
bothmega<-full_join(nonmega,mega) %>% arrange(Number) %>% select(Number, Name, BST)
```

```
## Joining, by = c("Number", "Name", "Type 1", "Type 2", "HP", "Attack", "Defense", "Sp. Atk", "Sp. Def
```

```
head(bothmega)
```

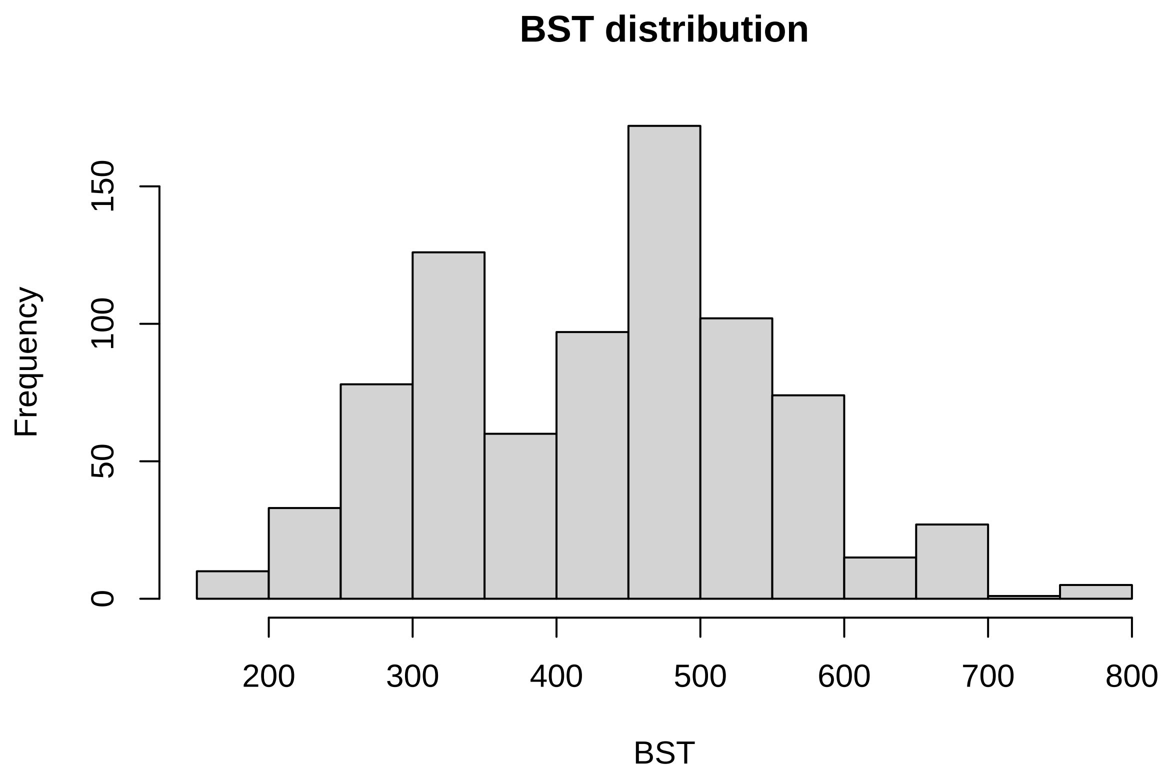
```
##   Number              Name BST
## 1      3          Venusaur 525
## 2      4     Mega Venusaur 625
## 3      7         Charizard 534
## 4      8 Mega Charizard X 634
## 5      9 Mega Charizard Y 634
## 6     12         Blastoise 530
```

The top BST of all Pokemon are shown below.

```
# Highest BST
BST<-pokemon %>% select(Number, Name, BST, Legendary) %>% arrange(desc(BST))
head(BST)
```

```
##   Number           Name BST Legendary
## 1    164  Mega Mewtwo X 780      TRUE
## 2    165  Mega Mewtwo Y 780      TRUE
## 3    427  Mega Rayquaza 780      TRUE
## 4    423  Primal Kyogre 770      TRUE
## 5    425 Primal Groudon 770      TRUE
## 6    553         Arceus 720      TRUE
```

```
hist(pokemon$BST, main = "BST distribution", xlab = "BST")
```



The top 5 are all Pokemon that are Mega Legendary Pokemon, a group that we identified earlier above. This makes sense as Legendary Pokemon are already powerful, and Mega evolution increases their BST by 100, making them even more powerful. The 6th Pokemon is Arceus, which is considered to be a god in the Pokemon World and was considered the most powerful Pokemon before Mega Evolution became a thing.

Surprisingly, the BST distribution is Bimodal. There are a bunch of Pokemon with low BST total.

Alright, let's take a closer look at the BST summary of several groups of Pokemon.

```
# BST summary of all Pokemon
pokemon %>% select(BST) %>% summary(pokemon)
```

```
##       BST
##  Min.   :180.0
##  1st Qu.:330.0
##  Median :450.0
##  Mean   :435.1
##  3rd Qu.:515.0
##  Max.   :780.0
```

```
# BST summary of non Legendary and Mega
pokemon %>% filter(str_detect(Name, "Mega ")==FALSE& str_detect(Name, "Primal")==FALSE & Legendary == F
  select(BST) %>% summary(BST)
```

```
##       BST
##  Min.   :180.0
##  1st Qu.:318.0
##  Median :413.0
##  Mean   :406.1
##  3rd Qu.:490.0
##  Max.   :670.0
```

```
# BST summary of Legendary
pokemon %>% filter(Legendary== TRUE) %>% select(BST) %>% summary(BST)
```

```
##       BST
##  Min.   :580.0
##  1st Qu.:580.0
##  Median :600.0
##  Mean   :637.4
##  3rd Qu.:680.0
##  Max.   :780.0
```

```
# BST summary of Mega
mega %>% select(BST) %>% summary(BST)
```

```
##       BST
##  Min.   :480.0
##  1st Qu.:582.5
##  Median :612.5
##  Mean   :623.6
##  3rd Qu.:638.8
##  Max.   :780.0
```

We can see that Legendary Pokemon have a higher average BST, meaning they are, on average, more powerful than a normal Pokemon. Mega Pokemon have a slightly lower average BST than Legendary, but Mega are still significantly more powerful than a normal Pokemon.

(Note: I have decided to leave out Mega Legendary out when looking at the summaries, because they are a very small group that are skewed to be powerful, and won't give us more insight that can be provided by Mega and Legendary groups.)

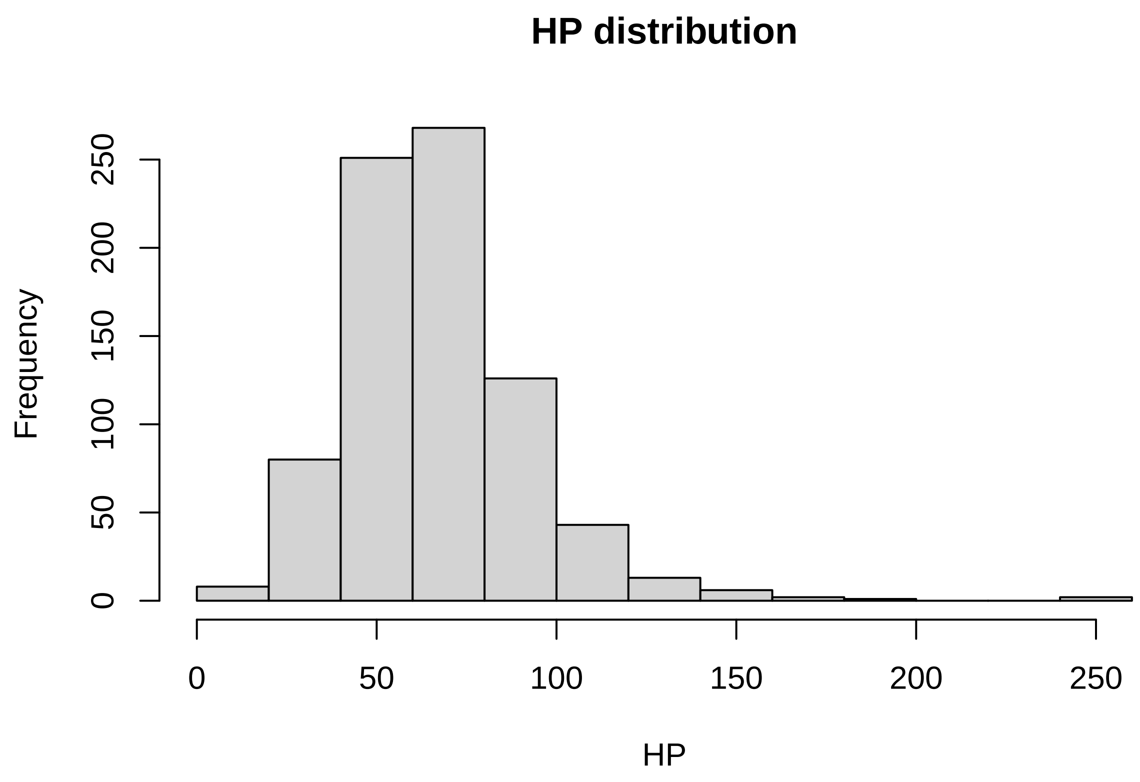
Next, we'll take a look at the distribution of each stats individually.

HP: The HP stat is used to determine how much damage a Pokemon can receive before fainting.

```
# HP
hp<-pokemon %>% select(Number, Name, HP, Legendary) %>% arrange(desc(HP))
head(hp)
```

```
##   Number      Name  HP Legendary
## 1    262   Blissey 255     FALSE
## 2    122   Chansey 250     FALSE
## 3    218 Wobbuffet 190     FALSE
## 4    352   Wailord 170     FALSE
## 5    656 Alomomola 165     FALSE
## 6    156   Snorlax 160     FALSE
```

```r
hist(pokemon$HP, main = "HP distribution", xlab = "HP")
```



Blissey and Chansey has such a high HP stat with 255 and 250 respectively, that it blows away the next Pokemon in Competition, Wobbuffet! However, surprisingly none of the top 6 for HP are Legendary Pokemon!

The distribution for the HP is skewed to the right, with a lot of outliers on the right.

```r
# HP summary of all Pokemon
pokemon %>% select(HP) %>% summary(pokemon)
```

```
##        HP
##  Min.   :  1.00
##  1st Qu.: 50.00
##  Median : 65.00
##  Mean   : 69.26
##  3rd Qu.: 80.00
##  Max.   :255.00
```

```r
# HP summary of non Legendary and Mega
pokemon %>% filter(str_detect(Name, "Mega ")==FALSE& str_detect(Name, "Primal")==FALSE & Legendary == F
  select(HP) %>% summary(HP)
```

```
##        HP
##  Min.   :  1.00
##  1st Qu.: 50.00
##  Median : 64.00
##  Mean   : 66.59
##  3rd Qu.: 78.00
##  Max.   :255.00
```

```
# HP summary of Legendary
pokemon %>% filter(Legendary== TRUE) %>% select(HP) %>% summary(HP)
```

```
##        HP
##  Min.   : 50.00
##  1st Qu.: 80.00
##  Median : 91.00
##  Mean   : 92.74
##  3rd Qu.:105.00
##  Max.   :150.00
```

```
# HP summary of Mega
mega %>% select(HP) %>% summary(HP)
```

```
##        HP
##  Min.   : 50.00
##  1st Qu.: 68.50
##  Median : 78.50
##  Mean   : 79.12
##  3rd Qu.: 93.75
##  Max.   :108.00
```

Legendary (and Mega) Pokemon still have a higher average HP than normal Pokemon, however the Max HP of a Legendary and Mega are significantly lower than the HP of Blissey.

```
# Highest HP for Legendary and Mega
legendary %>% filter(HP == max(HP)) %>% select(Number, Name, HP, Legendary)
```

```
##   Number                   Name  HP Legendary
## 1    545 Giratina Altered Forme 150      TRUE
## 2    546  Giratina Origin Forme 150      TRUE
```

```
mega %>% filter(HP == max(HP)) %>% select(Number, Name, HP, Legendary)
```

```
##   Number          Name  HP Legendary
## 1    495 Mega Garchomp 108     FALSE
```

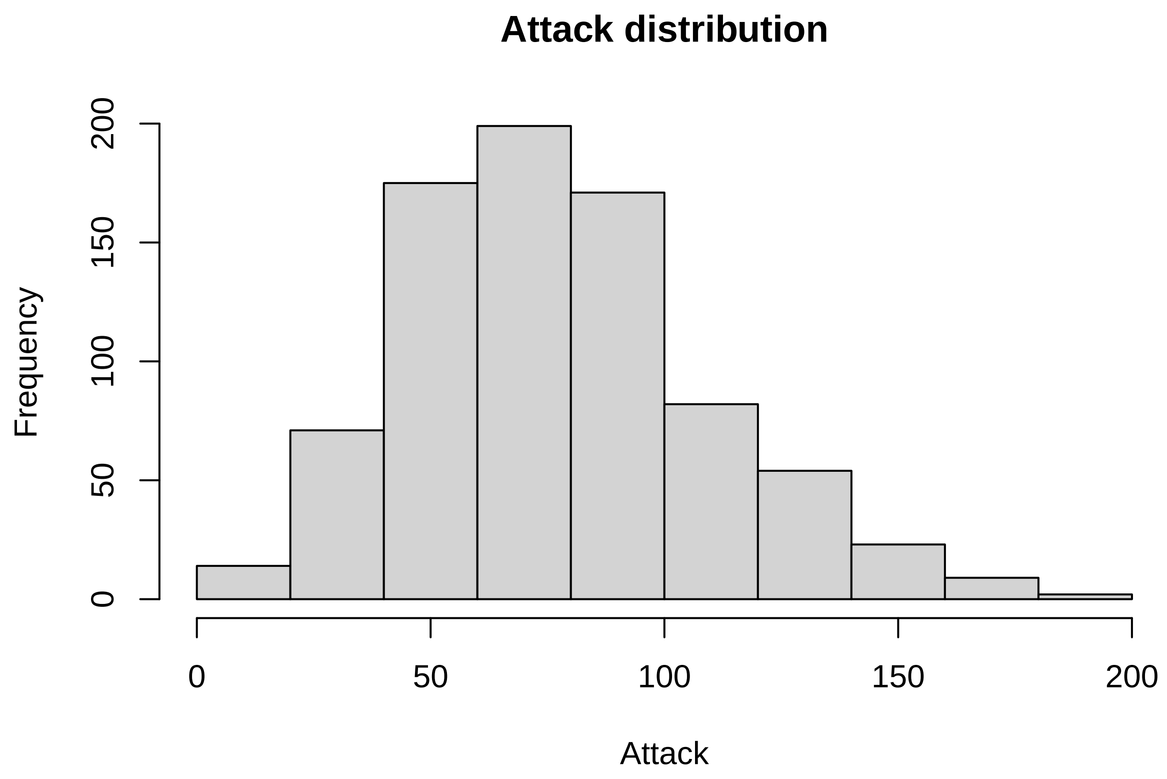
The highest HP stat for a Legendary goes towards both forms of Giratina with 150, while the highest HP stat for a Mega goes towards Mega Garchomp wiht 108.

Attack: The Attack stat determines how much damage a Pokemon deals with a physical move.

```r
# Attack
atk<-pokemon %>% select(Number, Name, Attack, Legendary) %>% arrange(desc(Attack))
head(atk)
```

```
##   Number               Name Attack Legendary
## 1    164      Mega Mewtwo X    190      TRUE
## 2    233     Mega Heracross    185     FALSE
## 3    425     Primal Groudon    180      TRUE
## 4    427      Mega Rayquaza    180      TRUE
## 5    430 DeoxysAttack Forme    180      TRUE
## 6    495      Mega Garchomp    170     FALSE
```

```r
hist(pokemon$Attack, main = "Attack distribution", xlab = "Attack")
```



The top 6 Pokemon for Attack are all either Mega Pokemon and/or Legendary Pokemon, with the highest being a Mega Legendary, Mega Mewtwo X with 190.

The distribution for Attack is skewed to the right.

```r
# Highest ATtack for a normal Pokemon
pokemon %>% filter(str_detect(Name, "Mega ")==FALSE& str_detect(Name, "Primal")==FALSE & Legendary == F
  filter(Attack == max(Attack)) %>% select(Number, Name, Attack, Legendary)
```

```
##   Number      Name Attack Legendary
## 1    455 Rampardos    165     FALSE
```

The highest Attack stat for a normal Pokemon is Rampardos with 165.

```
# Attack summary of all Pokemon
pokemon %>% select(Attack) %>% summary(pokemon)
```

```
##      Attack
##  Min.   :  5
##  1st Qu.: 55
##  Median : 75
##  Mean   : 79
##  3rd Qu.:100
##  Max.   :190
```

```
# Attack summary of non Legendary and Mega
pokemon %>% filter(str_detect(Name, "Mega ")==FALSE& str_detect(Name, "Primal")==FALSE & Legendary == F
  select(Attack) %>% summary(Attack)
```

```
##      Attack
##  Min.   :  5.00
##  1st Qu.: 52.00
##  Median : 70.00
##  Mean   : 72.83
##  3rd Qu.: 90.00
##  Max.   :165.00
```

```
# Attack summary of Legendary
pokemon %>% filter(Legendary== TRUE) %>% select(Attack) %>% summary(Attack)
```

```
##      Attack
##  Min.   : 50.0
##  1st Qu.:100.0
##  Median :110.0
##  Mean   :116.7
##  3rd Qu.:131.0
##  Max.   :190.0
```

```
# Attack summary of Mega
mega %>% select(Attack) %>% summary(Attack)
```

```
##      Attack
##  Min.   : 50.0
##  1st Qu.:100.8
##  Median :133.5
##  Mean   :127.8
##  3rd Qu.:150.0
##  Max.   :190.0
```

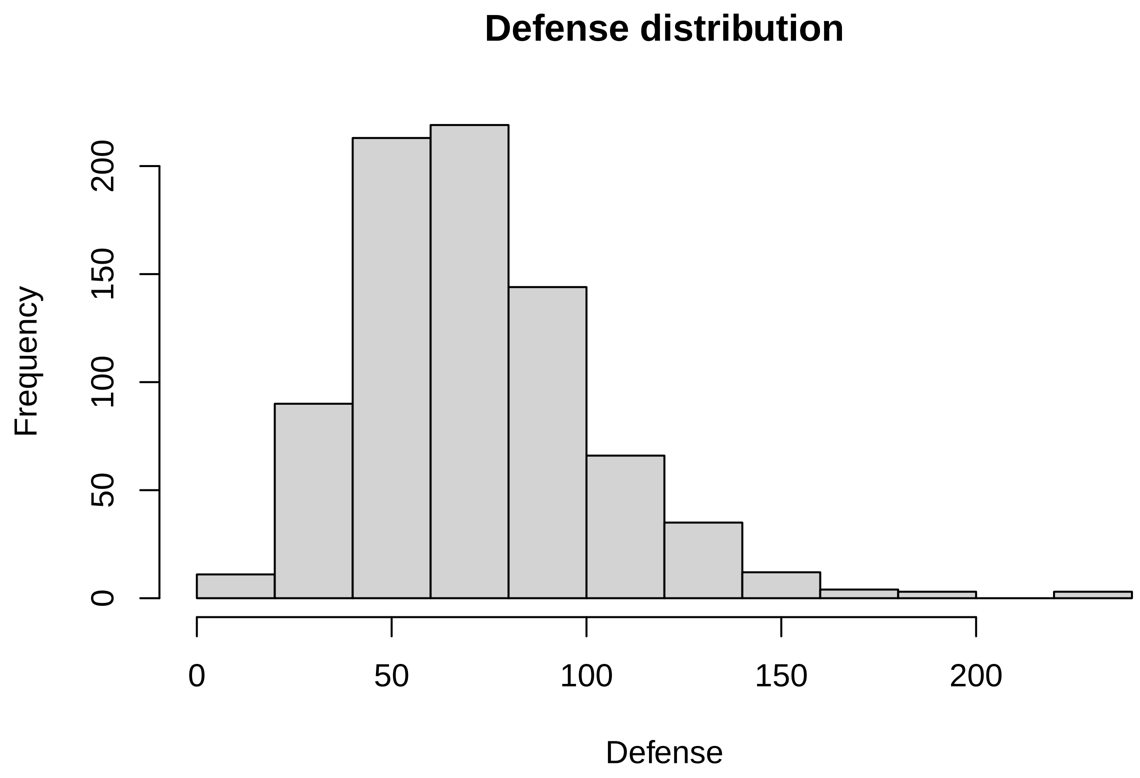
Both Mega and Legendary have a higher average Attack than normal Pokemon, while Mega Pokemon have a higher average Attack than Legendary.

Defense: The Defense stat determines how much damage a Pokemon receives when it is hit with a physical move (based off of the opponent's Attack stat).

```
# Defense
defense<-pokemon %>% select(Number, Name, Defense, Legendary) %>% arrange(desc(Defense))
head(defense)
```

```
##   Number         Name Defense Legendary
## 1    225 Mega Steelix     230     FALSE
## 2    231      Shuckle     230     FALSE
## 3    334  Mega Aggron     230     FALSE
## 4    224      Steelix     200     FALSE
## 5    415     Regirock     200      TRUE
## 6    790      Avalugg     184     FALSE
```

```
hist(pokemon$Defense, main = "Defense distribution", xlab = "Defense")
```



The Pokemon with the top Defense are Mega Steelix, Mega Aggron and Shuckle, with a whopping 230, while the Legendary with the highest Defense Stat is Regirock with 200.

The distribution for Defense is skewed to the right, with many outliers.

```
# Defense summary of all Pokemon
pokemon %>% select(Defense) %>% summary(pokemon)
```

```
##     Defense
##  Min.   :  5.00
##  1st Qu.: 50.00
##  Median : 70.00
##  Mean   : 73.84
##  3rd Qu.: 90.00
##  Max.   :230.00
```

```r
# Defense summary of non Legendary and Mega
pokemon %>% filter(str_detect(Name, "Mega ")==FALSE& str_detect(Name, "Primal")==FALSE & Legendary == F
  select(Defense) %>% summary(Defense)
```

```
##     Defense
##  Min.   :  5.00
##  1st Qu.: 50.00
##  Median : 65.00
##  Mean   : 69.41
##  3rd Qu.: 85.00
##  Max.   :230.00
```

```r
# Defense summary of Legendary
pokemon %>% filter(Legendary== TRUE) %>% select(Defense) %>% summary(Defense)
```

```
##     Defense
##  Min.   : 20.00
##  1st Qu.: 90.00
##  Median :100.00
##  Mean   : 99.66
##  3rd Qu.:115.00
##  Max.   :200.00
```

```r
# Defense summary of Mega
mega %>% select(Defense) %>% summary(Defense)
```

```
##     Defense
##  Min.   : 40.0
##  1st Qu.: 80.0
##  Median :100.0
##  Mean   :106.9
##  3rd Qu.:120.0
##  Max.   :230.0
```

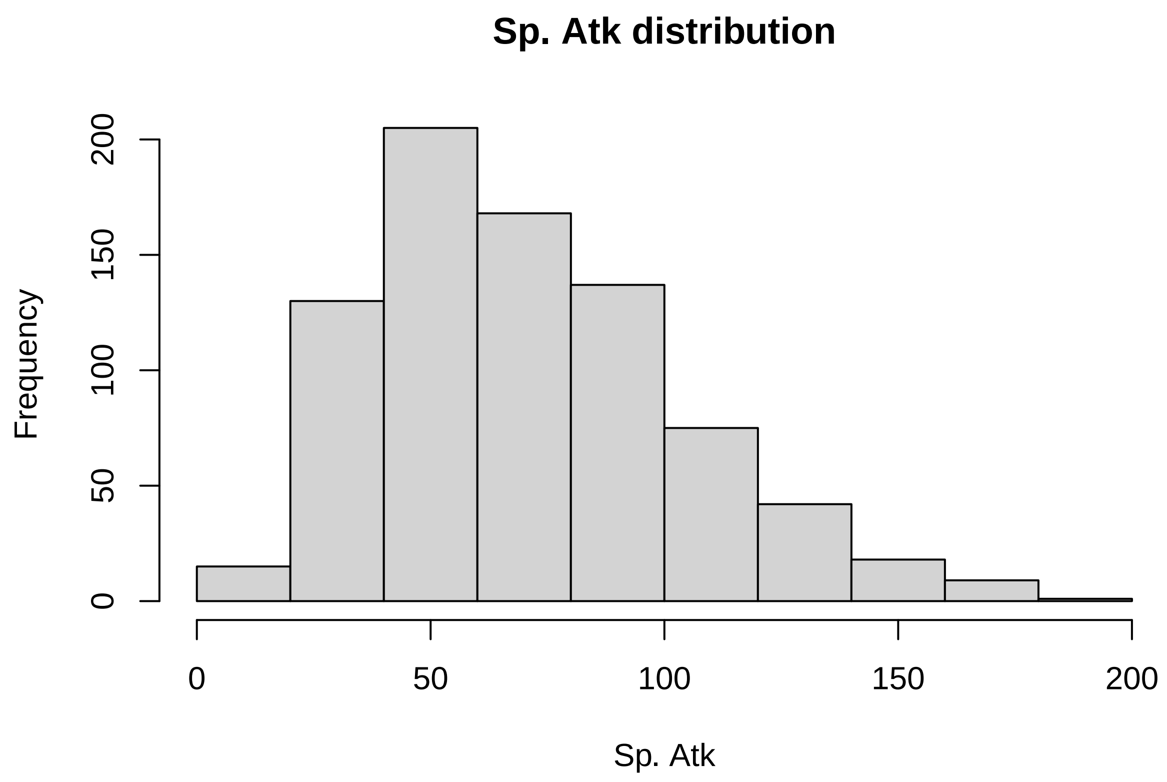
Both Mega and Legendary have a higher average Defense than normal Pokemon, while Mega Pokemon have a higher average Defense than Legendary, similar to the distribution of Attack stat.

Sp. Atk: The Special Attack stat determines how much damage a Pokemon deals with a special move.

```r
# Special Attack
satk<-pokemon %>% select(Number, Name, `Sp. Atk`, Legendary) %>% arrange(desc(`Sp. Atk`))
head(satk)
```

```
##   Number                Name Sp. Atk Legendary
## 1    165       Mega Mewtwo Y     194      TRUE
## 2    423       Primal Kyogre     180      TRUE
## 3    427       Mega Rayquaza     180      TRUE
## 4    430 DeoxysAttack Forme     180      TRUE
## 5     72       Mega Alakazam     175     FALSE
## 6    103         Mega Gengar     170     FALSE
```

```
hist(pokemon$`Sp. Atk`, main = "Sp. Atk distribution", xlab = "Sp. Atk")
```



The Pokemon with the top Sp. Atk is Mega Mewtwo Y with a stat of 194.

The distribution for Sp. atk is skewed to the right.

```
# Sp. Atk summary of all Pokemon
pokemon %>% select(`Sp. Atk`) %>% summary(pokemon)
```

```
##     Sp. Atk
##  Min.   : 10.00
##  1st Qu.: 49.75
##  Median : 65.00
##  Mean   : 72.82
##  3rd Qu.: 95.00
##  Max.   :194.00
```

```
# Sp. Atk summary of non Legendary and Mega
pokemon %>% filter(str_detect(Name, "Mega ")==FALSE& str_detect(Name, "Primal")==FALSE & Legendary == F
  select(`Sp. Atk`) %>% summary(`Sp. Atk`)
```

```
##     Sp. Atk
##  Min.   : 10.00
##  1st Qu.: 45.00
##  Median : 61.00
##  Mean   : 66.17
##  3rd Qu.: 85.00
##  Max.   :150.00
```

```
# Sp. Atk summary of Legendary
pokemon %>% filter(Legendary== TRUE) %>% select(`Sp. Atk`) %>% summary(`Sp. Atk`)
```

```
##     Sp. Atk
##  Min.   : 50.0
##  1st Qu.:100.0
##  Median :120.0
##  Mean   :122.2
##  3rd Qu.:150.0
##  Max.   :194.0
```

```
# Sp. Atk summary of Mega
mega %>% select(`Sp. Atk`) %>% summary(`Sp. Atk`)
```

```
##     Sp. Atk
##  Min.   : 15.0
##  1st Qu.: 80.0
##  Median :121.0
##  Mean   :115.5
##  3rd Qu.:145.0
##  Max.   :194.0
```

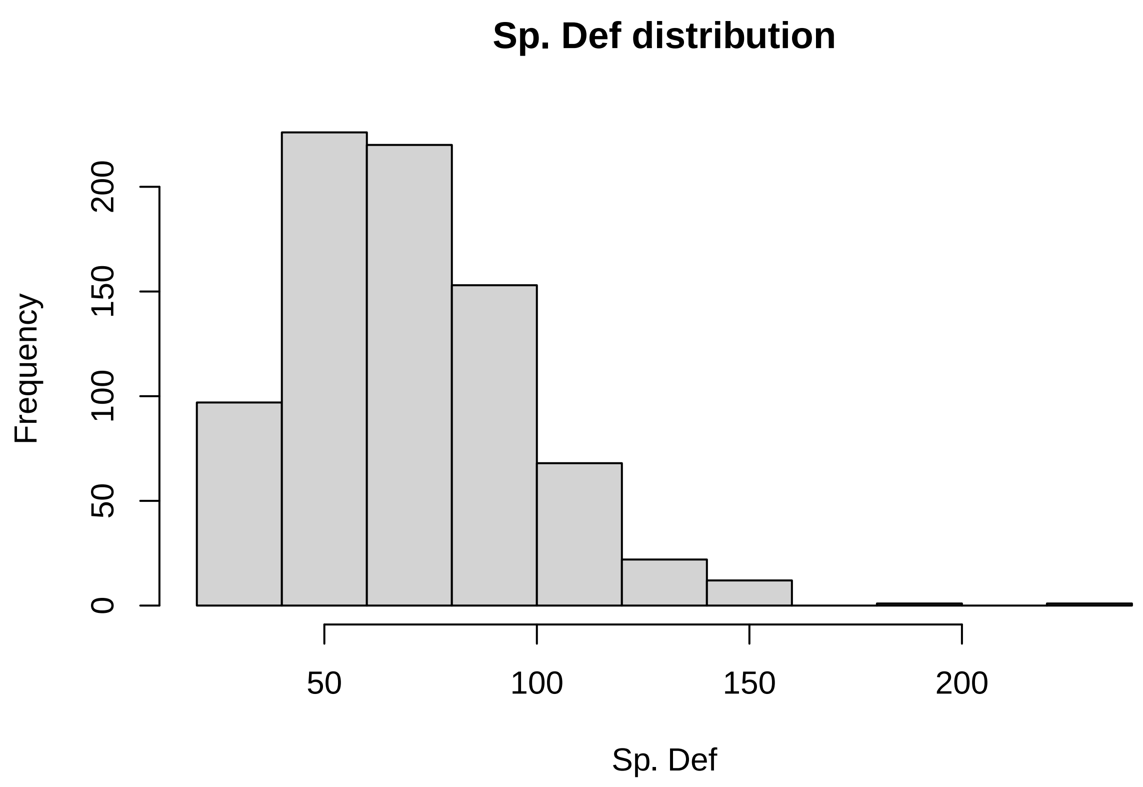
Legendary Pokemon have a slightly higher average Sp. Atk stat than Mega Pokemon, while both of these groups have a much higher average than normal Pokemon.

Sp. Def: The Special Defense stat determines how much damage a Pokemon deals with a special move, based upon the opponent's Sp. Atk stat.

```
# Special Defense
sdef<-pokemon %>% select(Number, Name, `Sp. Def`, Legendary) %>% arrange(desc(`Sp. Def`))
head(sdef)
```

```
##   Number                 Name Sp. Def Legendary
## 1    231              Shuckle     230     FALSE
## 2    416               Regice     200      TRUE
## 3    423        Primal Kyogre     160      TRUE
## 4    431 Deoxys Defense Forme     160      TRUE
## 5    270                Lugia     154      TRUE
## 6    271                Ho-oh     154      TRUE
```

```r
hist(pokemon$`Sp. Def`, main = "Sp. Def distribution", xlab = "Sp. Def")
```



The Pokemon with the highest Sp. Def stat is Shuckle with 230! It's surprising to see a normal Pokemon with two of the highest stats! Let's take note of Shuckle.

The Legendary with the highest Sp. Def stat is Regice with a stat of 200, while the Mega Pokemon with the highest Sp. Def stat is Primal Kyogre with 160. The distribution is skewed to the right, with only a few outliers to the right.

```r
# Sp. Def summary of all Pokemon
pokemon %>% select(`Sp. Def`) %>% summary(pokemon)
```

```
##     Sp. Def
##  Min.   : 20.0
##  1st Qu.: 50.0
##  Median : 70.0
##  Mean   : 71.9
##  3rd Qu.: 90.0
##  Max.   :230.0
```

```r
# Sp. Def summary of non Legendary and Mega
pokemon %>% filter(str_detect(Name, "Mega ")==FALSE& str_detect(Name, "Primal")==FALSE & Legendary == F
  select(`Sp. Def`) %>% summary(`Sp. Def`)
```

```
##     Sp. Def
##  Min.   : 20.00
##  1st Qu.: 50.00
##  Median : 65.00
##  Mean   : 67.19
##  3rd Qu.: 80.00
##  Max.   :230.00
```

```r
# Sp. Def summary of Legendary
pokemon %>% filter(Legendary== TRUE) %>% select(`Sp. Def`) %>% summary(`Sp. Def`)
```

```
##     Sp. Def
##  Min.   : 20.0
##  1st Qu.: 90.0
##  Median :100.0
##  Mean   :105.9
##  3rd Qu.:120.0
##  Max.   :200.0
```

```r
# Sp. Def summary of Mega
mega %>% select(`Sp. Def`) %>% summary(`Sp. Def`)
```

```
##     Sp. Def
##  Min.   : 60.0
##  1st Qu.: 85.0
##  Median : 98.0
##  Mean   :100.4
##  3rd Qu.:113.8
##  Max.   :160.0
```

Legendary Pokemon have a slightly higher average Sp. Def stat than Mega, while both of these groups have a much higher average than normal Pokemon.

Speed: The Speed stat determines the order of Pokemon that can act in battle. In general, the Pokemon with the higher Speed stat will attack first.

`{ r speed, echo = TRUE} # Speed speed<- pokemon %>% select(Number, Name, Speed, Legendary) %>% arrange(desc(Speed)) head(speed) hist(pokemon$Speed, main = "Speed distribution", xlab = "Speed")`

The Pokemon with the highest Speed is a Legendary Pokemon, Deoxys (Speed Forme), with a stat of 180.

Ninjask, a normal Pokemon, comes after with a stat of 160. The Mega Pokemon with the highest speed is Mega Alakazam and Mega Aerodactyl with 150. The distribution for Speed is skewed to the right.

```r
# Speed summary of all Pokemon
pokemon %>% select(Speed) %>% summary(pokemon)
```

```
##      Speed
##  Min.   :  5.00
##  1st Qu.: 45.00
##  Median : 65.00
##  Mean   : 68.28
##  3rd Qu.: 90.00
##  Max.   :180.00
```

```
# Speed summary of non Legendary and Mega
pokemon %>% filter(str_detect(Name, "Mega ")==FALSE& str_detect(Name, "Primal")==FALSE & Legendary == FA
  select(Speed) %>% summary(Speed)
```

```
##      Speed
##  Min.   :  5.00
##  1st Qu.: 45.00
##  Median : 60.00
##  Mean   : 63.93
##  3rd Qu.: 84.00
##  Max.   :160.00
```

```
# Speed summary of Legendary
pokemon %>% filter(Legendary== TRUE) %>% select(Speed) %>% summary(Speed)
```

```
##      Speed
##  Min.   : 50.0
##  1st Qu.: 90.0
##  Median :100.0
##  Mean   :100.2
##  3rd Qu.:110.0
##  Max.   :180.0
```

```
# Speed summary of Mega
mega %>% select(Speed) %>% summary(Speed)
```

```
##      Speed
##  Min.   : 20
##  1st Qu.: 75
##  Median :100
##  Mean   : 94
##  3rd Qu.:115
##  Max.   :150
```

Others: We can see something weird if we just look at the lowest stats of all the Pokemon.

```
# Lowest Stat
min <- pokemon %>%  select(HP, Attack, Defense, `Sp. Atk`, `Sp. Def`, Speed)
apply(min,2,min)
```

```
##      HP  Attack Defense Sp. Atk Sp. Def   Speed
##       1       5       5      10      20       5
```

There is a Pokemon with only 1 HP stat, which is strange, as the minimum stat for all the other stats is 5.

```
# One HP Pokemon
pokemon %>% filter(HP==1)
```

```
##   Number     Name Type 1 Type 2 HP Attack Defense Sp. Atk Sp. Def Speed
## 1    317 Shedinja    Bug  Ghost  1     90      45      30      30    40
##   Generation Legendary BST
## 1          3     FALSE 236
```

Turns out the 1 HP Pokemon is Shedinja. Seems really weak, huh? Turns out this Pokemon has an ability called **Wonder Guard** that makes this Pokemon immune to moves that aren't super effective against it, meaning only certain moves will affect this Pokemon. However, due to the limitations we've set up earlier, this Pokemon probably won't be able to win many battles.

2.2 Combat Dataset

The combat dataset contains 50000 randomly generated battles with 3 columns. This dataset uses the ID of the Pokemon to denote the Pokemon. The first column shows the first Pokemon (who always goes first in battle), the second column shows the second Pokemon in the battle, and finally, the third column shows the winner of the battle.

```
# Combat dataset
head(combat)
```

```
##    First_pokemon Second_pokemon Winner
## 1:           266            298    298
## 2:           702            701    701
## 3:           191            668    668
## 4:           237            683    683
## 5:           151            231    151
## 6:           657            752    657
```

```
dim(combat)
```

```
## [1] 50000     3
```

Win Rate

Let's see which Pokemon have the highest win rate. First, we'll obtain the number of matches each Pokemon has fought.

```
# Number of Wins
numberWins <- combat %>% group_by(Winner) %>% count()
head(numberWins)
```

```
## # A tibble: 6 x 2
## # Groups:   Winner [6]
##   Winner     n
##    <int> <int>
## 1      1    37
## 2      2    46
## 3      3    89
## 4      4    70
## 5      5    55
## 6      6    64
```

```
dim(numberWins)
```

```
## [1] 783   2
```

We can see from the dimensions of the numberWins, that not all 800 Pokemon has fought, as there are only 783 Pokemon accounted for.

```
# Finding which Pokemon hasn't fought
anti_join(pokemon, numberWins, by = c("Number"="Winner"))
```

```
##    Number                Name   Type 1  Type 2  HP Attack Defense Sp. Atk
## 1      12           Blastoise    Water          79     83     100      85
## 2      33           Sandshrew   Ground          50     75      85      20
## 3      46          Wigglytuff   Normal   Fairy 140     70      45      85
## 4      66             Poliwag    Water          40     50      40      40
## 5      78          Victreebel    Grass  Poison  80    105      65     100
## 6      90            Magneton Electric   Steel  50     60      95     120
## 7     144               Ditto   Normal          48     48      48      48
## 8     183             Ariados      Bug  Poison  70     90      70      60
## 9     231             Shuckle      Bug    Rock  20     10     230      10
## 10    236            Ursaring   Normal          90    130      75      75
## 11    322            Hariyama Fighting         144    120      60      40
## 12    419         Mega Latias   Dragon Psychic  80    100     120     140
## 13    479           Honchkrow     Dark  Flying 100    125      52     105
## 14    556             Servine    Grass          60     60      75      60
## 15    618            Maractus    Grass          75     86      67     106
## 16    655           Jellicent    Water   Ghost 100     60      70      85
## 17    782 Pumpkaboo Small Size    Ghost   Grass  44     66      70      44
##    Sp. Def Speed Generation Legendary BST
## 1      105    78          1     FALSE 530
## 2       30    40          1     FALSE 300
## 3       50    45          1     FALSE 435
## 4       40    90          1     FALSE 300
## 5       70    70          1     FALSE 490
## 6       70    70          1     FALSE 465
## 7       48    48          1     FALSE 288
## 8       60    40          2     FALSE 390
## 9      230     5          2     FALSE 505
## 10      75    55          2     FALSE 500
## 11      60    50          3     FALSE 474
## 12     150   110          3      TRUE 700
## 13      52    71          4     FALSE 505
## 14      75    83          5     FALSE 413
## 15      67    60          5     FALSE 461
## 16     105    60          5     FALSE 480
## 17      55    56          6     FALSE 335
```

Even though there are Pokemon that haven't fought, we will be able to predict their win percentages with machine learning.

Going further, not all Pokemon fought the same number of battles, therefore, we need to normalize by the number of battles fought.

```
# normalize first pokemon
firstCount <- combat %>% group_by(First_pokemon) %>% count()
head(firstCount)
```

```
## # A tibble: 6 x 2
## # Groups:   First_pokemon [6]
##   First_pokemon     n
##           <int> <int>
## 1             1    70
## 2             2    55
## 3             3    68
## 4             4    62
## 5             5    50
## 6             6    66
```

```r
dim(firstCount)
```

```
## [1] 784   2
```

```r
# normalize second pokemon
secondCount <- combat %>% group_by(Second_pokemon) %>% count()
head(secondCount)
```

```
## # A tibble: 6 x 2
## # Groups:   Second_pokemon [6]
##   Second_pokemon     n
##            <int> <int>
## 1              1    63
## 2              2    66
## 3              3    64
## 4              4    63
## 5              5    62
## 6              6    52
```

```r
dim(secondCount)
```

```
## [1] 784   2
```

There seems to be something wrong. Looking at the firstCount and secondCount, they all have 784 Pokemon, however numberWins shows that there are 783 Pokemon. This seems to imply that one Pokemon just couldn't win no matter what. Let's find out which Pokemon this is.

```r
# Finding the Pokemon that couldn't win
neverWon <- firstCount$First_pokemon[!firstCount$First_pokemon %in% numberWins$Winner]
pokemon[pokemon$Number == neverWon,]
```

```
##    Number    Name Type 1 Type 2 HP Attack Defense Sp. Atk Sp. Def Speed
## 1:    231 Shuckle    Bug   Rock 20     10     230      10     230     5
##    Generation Legendary BST
## 1:          2     FALSE 505
```

Aw, poor Shuckle. We saw that Shuckle had the highest Defenses of any Pokemon. However, this doesn't seem to translate to Shuckle winning. We can see that Shuckle's other stats are abysmal. Perhaps this has something to do with Shuckle not being able to win. Either way, we need to add Shuckle into the dataset, even though it has never won.

```r
# Adding Shuckle to dataset
numberWins <- rbind(numberWins, c(Winner= 231, n =0))
numberWins <- numberWins[order(numberWins$Winner),]
```

After adding Shuckle, we can now calculate the win percentage of each Pokemon.

```r
# Obtain total battles for each pokemon
numberWins$Total_Battles <- firstCount$n + secondCount$n
# Win percentage
numberWins$Win_Percentage <- numberWins$n/numberWins$Total_Battles
head(numberWins)
```

```
## # A tibble: 6 x 4
## # Groups:   Winner [6]
##   Winner     n Total_Battles Win_Percentage
##    <dbl> <dbl>         <int>          <dbl>
## 1      1    37           133          0.278
## 2      2    46           121          0.380
## 3      3    89           132          0.674
## 4      4    70           125          0.56
## 5      5    55           112          0.491
## 6      6    64           118          0.542
```

Let's see the Pokemon with the highest win rate.

```r
# Finding highest win rate Pokemon
pokemon<-full_join(pokemon, numberWins, by = c("Number" = "Winner"))
pokemon <- pokemon %>% rename(Wins = n)
top_winner<-pokemon %>% group_by(Win_Percentage) %>% select(-`Type 1`, -`Type 2`, -Generation, - Wins,
  arrange(desc(Win_Percentage))
head(top_winner)
```

```
## # A tibble: 6 x 11
## # Groups:   Win_Percentage [6]
##   Number Name     HP Attack Defense `Sp. Atk` `Sp. Def` Speed Legendary   BST
##    <dbl> <chr> <int>  <int>   <int>     <int>     <int> <int> <lgl>     <int>
## 1    155 Mega~    80    135      85        70        95   150 FALSE       615
## 2    513 Weav~    70    120      65        45        85   125 FALSE       510
## 3    704 Torn~    79    100      80       110        90   121 TRUE        580
## 4     20 Mega~    65    150      40        15        80   145 FALSE       495
## 5    154 Aero~    80    105      65        60        75   130 FALSE       515
## 6    477 Mega~    65    136      94        54        96   135 FALSE       580
## # ... with 1 more variable: Win_Percentage <dbl>
```

We can see that there are more Mega Pokemon within highest win rate Pokemon than Legendary Pokemon and surprisingly there is a normal Pokemon that have higher win rate than a Legendary!

Let's take a look at the Pokemon with the lowest win rate now.

```r
# Finding lowest win rate Pokemon
top_loser <-top_winner %>% arrange(Win_Percentage)
head(top_loser)
```

```
## # A tibble: 6 x 11
## # Groups:   Win_Percentage [6]
##   Number Name     HP Attack Defense `Sp. Atk` `Sp. Def` Speed Legendary   BST
##    <dbl> <chr> <int>  <int>   <int>     <int>     <int> <int> <lgl>     <int>
## 1    231 Shuc~    20     10     230        10       230     5 FALSE       505
## 2    290 Silc~    50     35      55        25        25    15 FALSE       205
## 3    190 Toge~    35     20      65        40        65    20 FALSE       245
## 4    639 Solo~    45     30      40       105        50    20 FALSE       290
## 5    237 Slug~    40     40      40        70        40    20 FALSE       250
## 6    577 Munna    76     25      45        67        55    24 FALSE       292
## # ... with 1 more variable: Win_Percentage <dbl>
```

Wow, there is such a disparity between the stats of the highest win rate Pokemon and the lowest win rate Pokemon. At a glance, the most disparing of these Pokemon's stats are the Attack and Speed stats. This may suggest that the Attack and Speed stats are important to determine the winning Pokemon in a battle. We can suggest that maybe those that can attack hard first, will have the advantage since they may be able to win the battle faster than the opposing Pokemon.

Win Rate by Type:

```
# Average Win Percentage by Type 1
type_win_rate_1 <- drop_na(pokemon)
type_win_rate_1 <- aggregate(type_win_rate_1$Win_Percentage, by=list(type_win_rate_1$`Type 1`), FUN = me
type_win_rate_1 <- type_win_rate_1 %>% rename(Type = Group.1, Win_Percentage_1 = x) %>% arrange(desc(Wi
# Average Win Percentage by Type 2
type_win_rate_2 <- drop_na(pokemon)
type_win_rate_2 <- aggregate(type_win_rate_2$Win_Percentage, by=list(type_win_rate_2$`Type 2`), FUN = me
type_win_rate_2 <- type_win_rate_2[-1, ]
type_win_rate_2 <- type_win_rate_2 %>% rename(Type = Group.1, Win_Percentage_2 = x) %>% arrange(desc(Wi
# Average Win Percentage by Type
type_win_rate <- full_join(type_win_rate_1, type_win_rate_2)
```

```
## Joining, by = "Type"
```

```
type_win_rate <- full_join(type, type_win_rate)
```

```
## Joining, by = "Type"
```

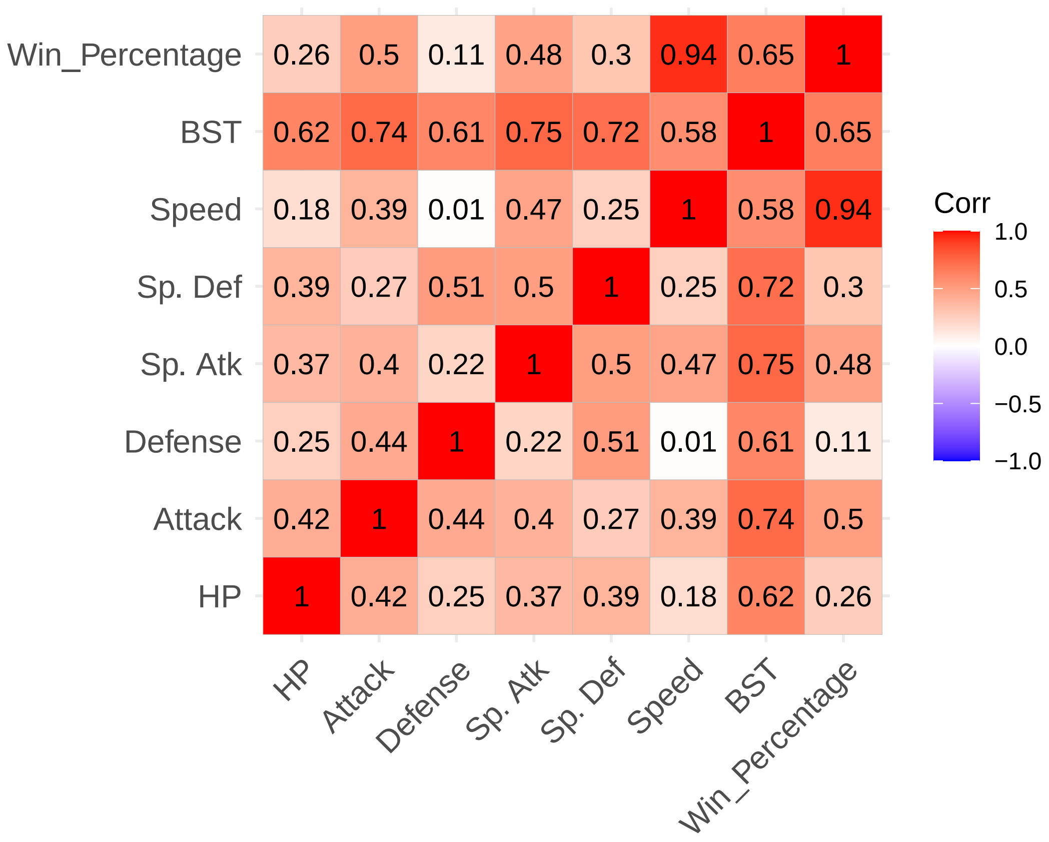
```
type_win_rate <- type_win_rate %>%
  mutate(Total_Win_Percentage = ((Win_Percentage_1 * Total_1)+(Win_Percentage_2 * Total_2))/Total) %>%
  select(Type, Total_Win_Percentage) %>% arrange(desc(Total_Win_Percentage))
type_win_rate
```

```
## # A tibble: 18 x 2
##    Type     Total_Win_Percentage
##    <chr>                   <dbl>
##  1 Flying                  0.668
##  2 Dragon                  0.624
##  3 Electric                0.621
##  4 Dark                    0.616
##   5 Fire                        0.589
##   6 Fighting                    0.574
##   7 Normal                      0.540
##   8 Psychic                     0.533
##   9 Ice                         0.494
##  10 Ghost                       0.472
##  11 Water                       0.463
##  12 Steel                       0.459
##  13 Ground                      0.455
##  14 Poison                      0.444
##  15 Grass                       0.435
##  16 Bug                         0.433
##  17 Fairy                       0.388
##  18 Rock                        0.376
```

We can see that the 3 types with the highest average Win Percentage are Flying, Dragon, and Electric and the three lowest are Rock, Fairy and Bug. Perhaps GameFreak was correct in introducing the Fairy Type as mentioned earlier, to reign in Dragon Types being too overpowered.

Correlation by Stats:

```
# Correlation Table for Stats
stats<-pokemon %>% select(-Number, -Name, -`Type 1`, - `Type 2`, -Generation, -Legendary, -Wins, -Total_
stats<-drop_na(stats)
stats<-signif(cor(stats),2)
ggcorrplot(stats, lab = TRUE)
```

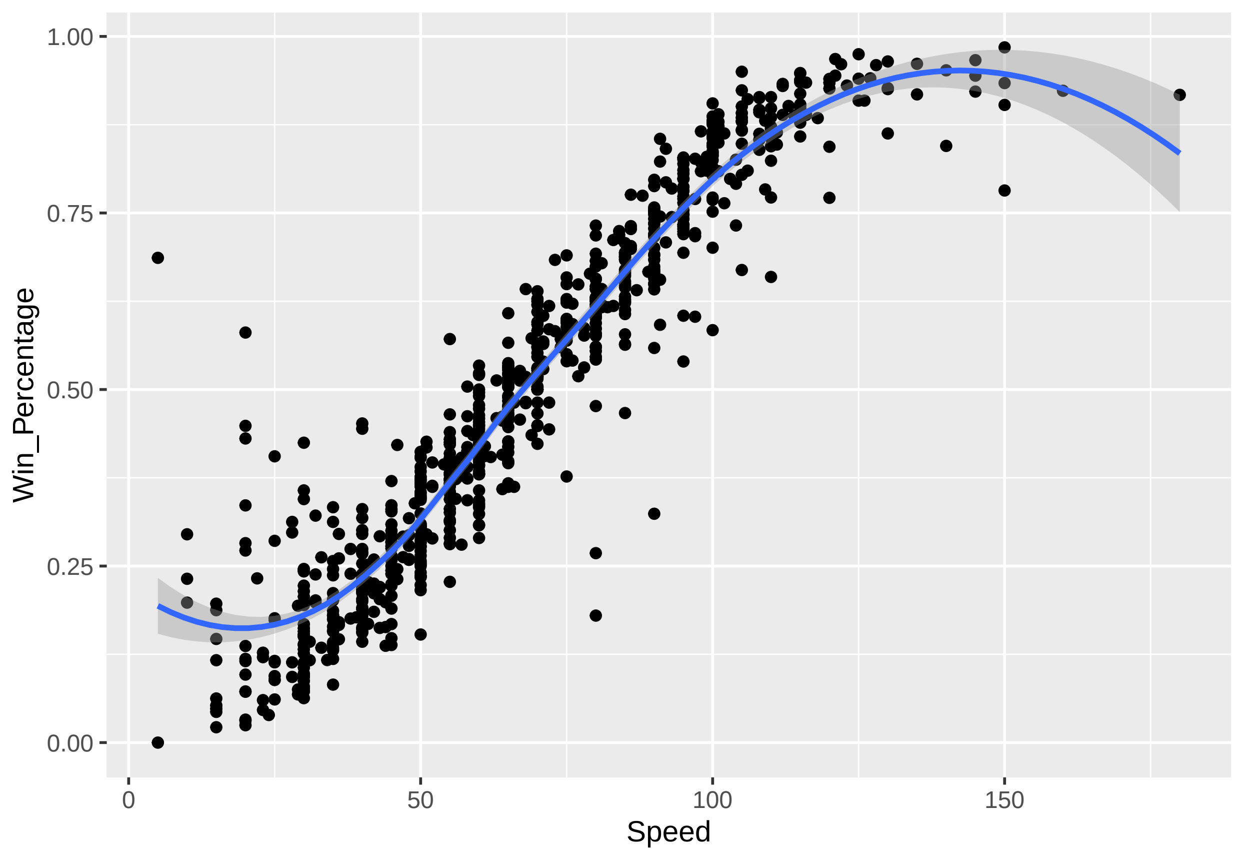


	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	BST	Win_Percentage
Win_Percentage	0.26	0.5	0.11	0.48	0.3	0.94	0.65	1
BST	0.62	0.74	0.61	0.75	0.72	0.58	1	0.65
Speed	0.18	0.39	0.01	0.47	0.25	1	0.58	0.94
Sp. Def	0.39	0.27	0.51	0.5	1	0.25	0.72	0.3
Sp. Atk	0.37	0.4	0.22	1	0.5	0.47	0.75	0.48
Defense	0.25	0.44	1	0.22	0.51	0.01	0.61	0.11
Attack	0.42	1	0.44	0.4	0.27	0.39	0.74	0.5
HP	1	0.42	0.25	0.37	0.39	0.18	0.62	0.26

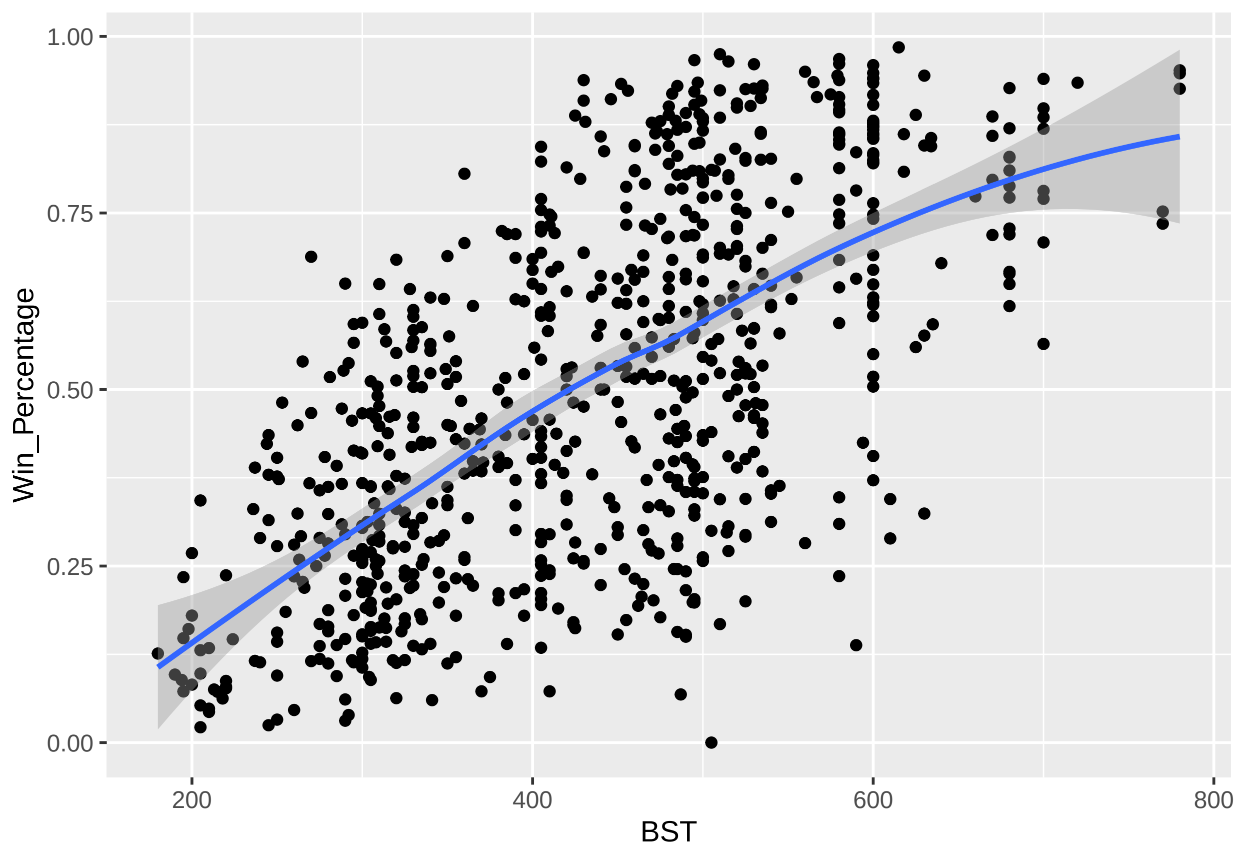
From the correlation table, we can see that the Win Percentage is more correlated to the Speed stat, rather than anything else. Following that is BST. A higher BST means a higher Attack stats, so we can see how that is correlated to a higher Win Percentage. Next are the two Attack stats. Lastly, the other stats aren't as correlated. Defenses doesn't matter, if you can't beat the opposing Pokemon.

We can plot the Speed, BST, Attack and Sp. Atk against Win Percentage.

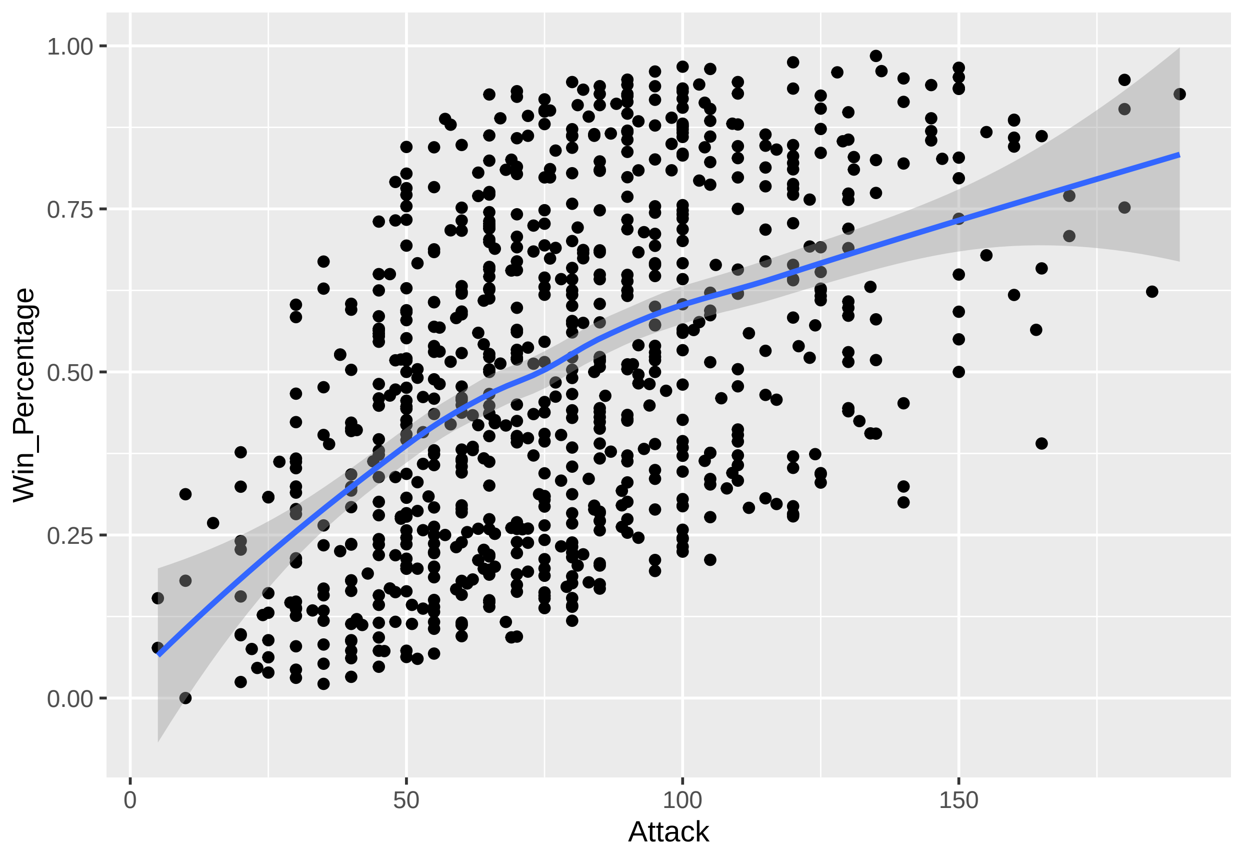
```r
# plotting win percentage against Stat
ggplot(pokemon, aes(x = Speed, y =  Win_Percentage)) + geom_point()+ geom_smooth()
```



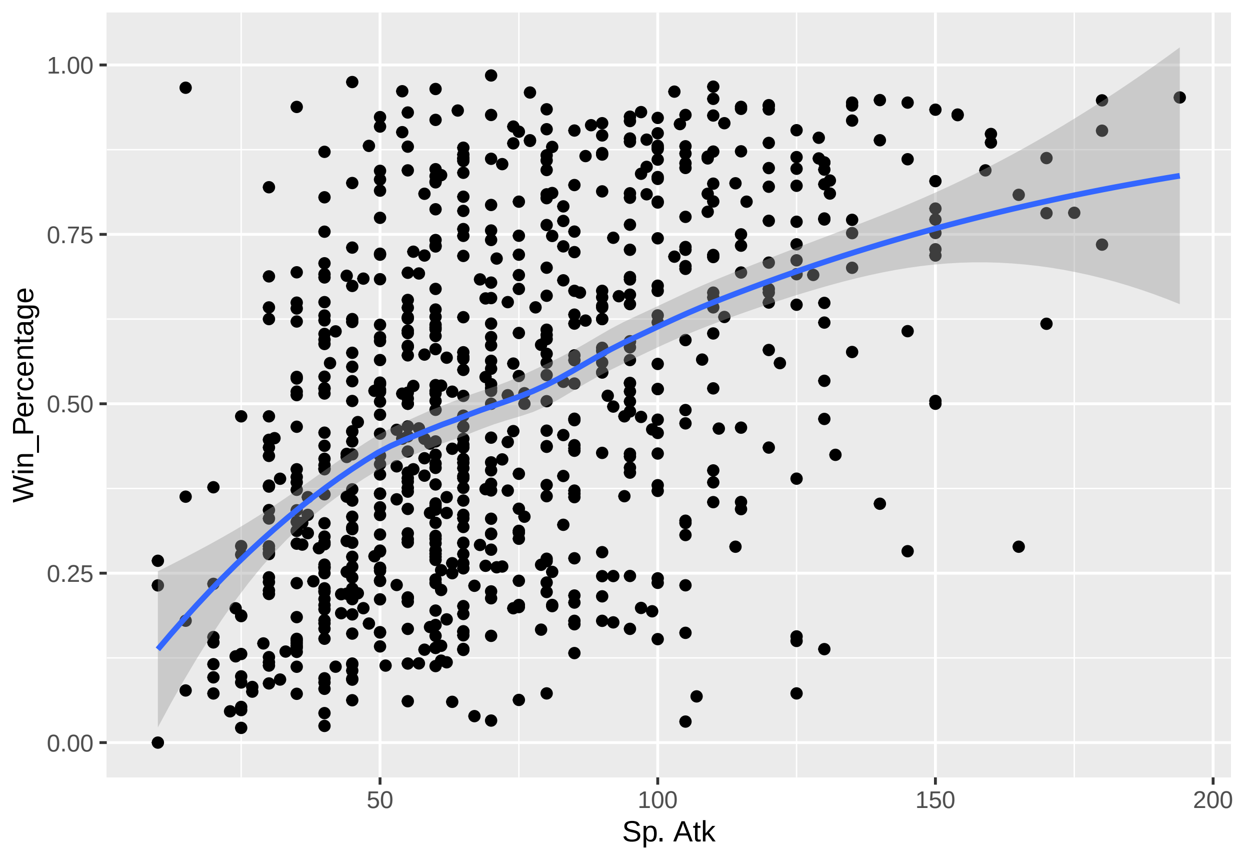
```r
ggplot(pokemon, aes(x = BST, y =  Win_Percentage)) + geom_point()+ geom_smooth()
```



```
ggplot(pokemon, aes(x = Attack, y =  Win_Percentage)) + geom_point()+ geom_smooth()
```



```r
ggplot(pokemon, aes(x = `Sp. Atk`, y =  Win_Percentage)) + geom_point()+ geom_smooth()
```



We can clearly see that Speed is much more correlated to Win Percentage.

Win Rate by Groups:

```
# Legendary Win Rate
leg_win_rate <- drop_na(pokemon)
leg_win_rate <- aggregate(leg_win_rate$Win_Percentage,
        by=list(leg_win_rate$Legendary), FUN = mean)
leg_win_rate
```

Legendary

```
##   Group.1         x
## 1   FALSE 0.4761663
## 2    TRUE 0.7791366
```

We can see that Legendary Pokemon have a high Win Percentage with 0.779.

```
# Mega Win Rate
mega_win_rate<- left_join(mega, pokemon)
```

Mega

```
## Joining, by = c("Number", "Name", "Type 1", "Type 2", "HP", "Attack", "Defense", "Sp. Atk", "Sp. Def
```

```
mega_win_rate<- drop_na(mega_win_rate)
mega_win_rate<- sum(mega_win_rate$Wins)/sum(mega_win_rate$Total_Battles)
mega_win_rate
```

```
## [1] 0.7211035
```

And that Mega Pokemon have a lower but still pretty high Win Percentage of 0.721.

```


# Mega Legendary Win RAte
mega_leg_win_rate<- left_join(mega_legendary, pokemon)
```

Mega Legendary

```
## Joining, by = c("Number", "Name", "Type 1", "Type 2", "HP", "Attack", "Defense", "Sp. Atk", "Sp. Def
```

```
mega_leg_win_rate<- drop_na(mega_leg_win_rate)
mega_leg_win_rate<- sum(mega_leg_win_rate$Wins)/sum(mega_leg_win_rate$Total_Battles)
mega_leg_win_rate
```

```
## [1] 0.8713318
```

While Mega Legendary Pokemon have the highest overall Win Percentage with 0.871.

3. Analysis/Methods

To figure out winner between combats, we need to combine the two dataset to form a new one, so that the combat dataset has the information from the Pokemon dataset. Winner will be binary, so that if the First Pokemon won, it would be a 1, otherwise it'll be a 0.

```
# Combining datasets
fight<-combat
# Join two dataset together by Number
fight<- fight %>% left_join(pokemon, by= c("First_pokemon" = "Number")) %>%
  left_join(pokemon, by = c("Second_pokemon" = "Number")) %>% select(-Name.x, -Generation.x,
      -Wins.x, -Total_Battles.x, -Name.y, -Generation.y, -Wins.y, -Total_Battles.y) %>%
  mutate(Winner_First = ifelse(Winner == First_pokemon, 1, 0))
```

Model 1: First Pokemon

Since we have concluded that Speed is an important factor for deciding the winner and as the first column dictates which Pokemon goes first, maybe we can guess that the first Pokemon normally wins. This is a simple method to obtain our predictions.

```r
# First Pokemon Model
mean(fight$First_pokemon == fight$Winner)
```

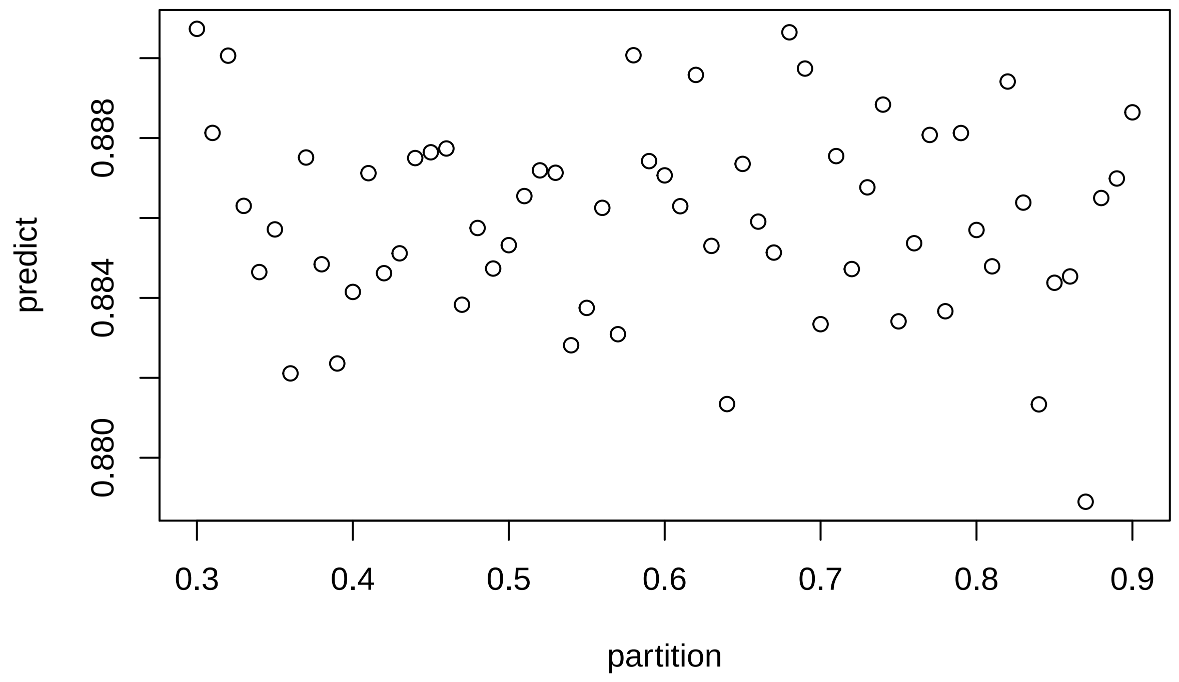
```
## [1] 0.47202
```

Huh, only an accuracy of 0.472. That's worse than just simply guessing the winning Pokemon from the two battling, which would be 50-50. If we look at the dataset, from the first row, the first Pokemon has a lower Speed than the second Pokemon. I made an assumption that the first Pokemon to attack was equal to it having a higher speed. This isn't the case, and it seems to be random.

Model 2: GLM

Before working on the next model, a linear regression model, we need to partition our dataset to train and test them. We didn't need to do this in our previous model, since it was a simple formula of guessing the first Pokemon to fight. However, finding the right partition to split the dataset for test and train set is important as we may overfit the data. There are many factors that can dictate what percentage we divide the test and train set, but it may just seem arbitrary. One common method is to use 70% train, 30% test. We'll test out different percentages to see which percentage we'll use for our partitions.

```r
# Testing what percentage to Partition
partition <- seq(from=.30, to =.90, by=.01)
# Calculate accuracy for each value
predict<- sapply(partition, function(p){
  test_index<- createDataPartition(fight$Winner_First, times =1, p=p, list= FALSE)
  test_set <- fight[test_index,]
  train_set <- fight[-test_index,]
  fit <- glm(Winner_First ~ HP.x+Attack.x+Defense.x+`Sp. Atk.x`+`Sp. Def.x`+
    Speed.x+HP.y+Attack.y+Defense.y+`Sp. Atk.y`+`Sp. Def.y`+ Speed.y,
    data = train_set)
  glm_preds<-round(predict(fit, test_set))
  mean(glm_preds==test_set$Winner_First)
})

# Plotting accuracy over percentage
plot(partition, predict)
```



```
partition[which.max(predict)]
```

```
## [1] 0.3
```

```
max(predict)
```

```
## [1] 0.8907333
```

There seems to be some differences between the partition percentage and the accuracy. So, we shall use the more common percentage to partition our dataset. We will use this 70-30 split the dataset into train and test set respectively for further analysis.

For our generalized linear model, we'll use just the Stats of the Pokemon.

```
# Partitioning Train and Test Set
set.seed(1, sample.kind = "Rounding")
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
test_index<-createDataPartition(fight$Winner, times = 1, p = 0.3, list = FALSE)
test_set<- fight[test_index,]
train_set<-fight[-test_index,]
```

```r
# GLM
fit <- glm(Winner_First ~ HP.x+Attack.x+Defense.x+`Sp. Atk.x`+`Sp. Def.x`+
    Speed.x+HP.y+Attack.y+Defense.y+`Sp. Atk.y`+`Sp. Def.y`+ Speed.y,
    data = train_set)
glm_preds<-round(predict(fit, test_set))
mean(glm_preds==test_set$Winner_First)
```

```
## [1] 0.8909479
```

Alright, that's an accuracy of 0.891, not that bad. This certainly is much better than our first Model.

Model 3: Only Speed GLM

From beforehand, we can see that Speed is much more closely correlated to Winning than the other Stats. We should remove the other less correlated Stats. Thus, we'll only use the Speed from both Pokemon to see if we can improve our model.

```r
# GLM speed only
speed_fit <- glm(Winner_First ~ Speed.x+Speed.y,
           data = train_set)
glm_speed_pred<-round(predict(speed_fit, test_set))
mean(glm_speed_pred==test_set$Winner_First)
```

```
## [1] 0.9242768
```

Yay, our new model gave us a higher accuracy of 0.924! We're doing better, but let's improve this further.

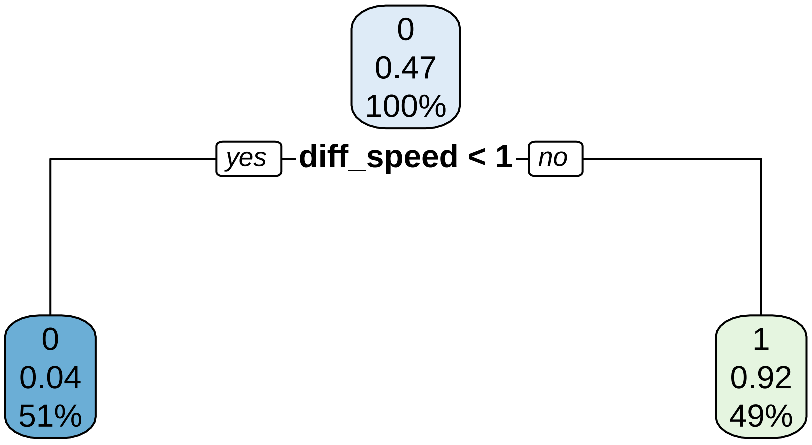
Model 4: Classification Tree

So, we can see that Speed is much more important for our models. Due to this, we can simplify this slightly by getting the difference between the two Pokemon's Speed Stat. So that this diff_speed, is based on the first Pokemon. We can use a Classification Tree to make our new model this time. Classification tree is a decision tree that takes our data to go from observations about an item to conclusion about the item's target value.

```r
# Add difference of speed
fight <-fight %>% mutate(diff_speed = Speed.x-Speed.y)
# Partitioning Train and Test Set
set.seed(1, sample.kind = "Rounding")
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```r
test_index<-createDataPartition(fight$Winner, times = 1, p = 0.3, list = FALSE)
test_set<- fight[test_index,]
train_set<-fight[-test_index,]

# Tree
tree<-rpart(Winner_First ~ diff_speed+Speed.x + Speed.y, data= train_set, method = 'class')
rpart.plot(tree)
```



```
predict_tree<- predict(tree, test_set, type = 'class')
mean(predict_tree==test_set$Winner_First)
```

```
## [1] 0.9430743
```

We have now a further increase in accuracy to 0.943. This tree is really simple, having only one node; we didn't need to consider each Pokemon's Speed individually. This is because the difference of that Speed stat is just that important, making it able to predict the majority of the Winners already. As long as a Pokemon is faster, it will probably win the fight. But can our model be improved even further?

Model 5: Random Forest

Our next model to consider is Random Forest. This is also a tree model, similar to our previous model, however this time, we will use multiple decision trees with more variables to consider and outputting the classs that is mode of the classes of the individual trees. A random decision forests correct for decision tree's habit of overfitting to their training set. The more trees that we can, the more we minimize the errors.

```
# Rename variables so they are easier to work with (probably should've done this earlier)
fight<- fight %>% rename(Type1x = `Type 1.x`, Type2x =`Type 2.x`, Type1.y =`Type 1.y`, Type2.y=`Type 2.
        Sp_Atk.x = `Sp. Atk.x`, Sp_Atk.y = `Sp. Atk.y`,Sp_Def.x = `Sp. Def.x`,Sp_Def.y = `Sp. Def.y`) %
  select(-First_pokemon, -Second_pokemon, - Winner, -Win_Percentage.x, -Win_Percentage.y)
fight<-transform(fight, Winner_First = as.factor(Winner_First))
# Recreate Partition
set.seed(1, sample.kind = "Rounding")
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
test_index<-createDataPartition(fight$Winner, times = 1, p = 0.3, list = FALSE)
test_set<- fight[test_index,]
train_set<-fight[-test_index,]
# Random Forest
rf<- randomForest(Winner_First~., data = train_set, type = "classification", ntree=500)
rf
```

```
##
## Call:
##  randomForest(formula = Winner_First ~ ., data = train_set, type = "classification",      ntree = 50
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 4
##
##         OOB estimate of  error rate: 4.41%
## Confusion matrix:
##       0     1 class.error
## 0 17546   933  0.05048975
## 1   609 15911  0.03686441
```

```
#Accuracy
predict_rf<-predict(rf,test_set)
mean(predict_rf==test_set$Winner_First)
```

```
## [1] 0.954603
```

The accuracy has increased to 0.955 with our new Random Forest. However, there is a way to increase the accuracy even further, as we've just used the default for Random Forest.

Model 6. Optimized Random Forest

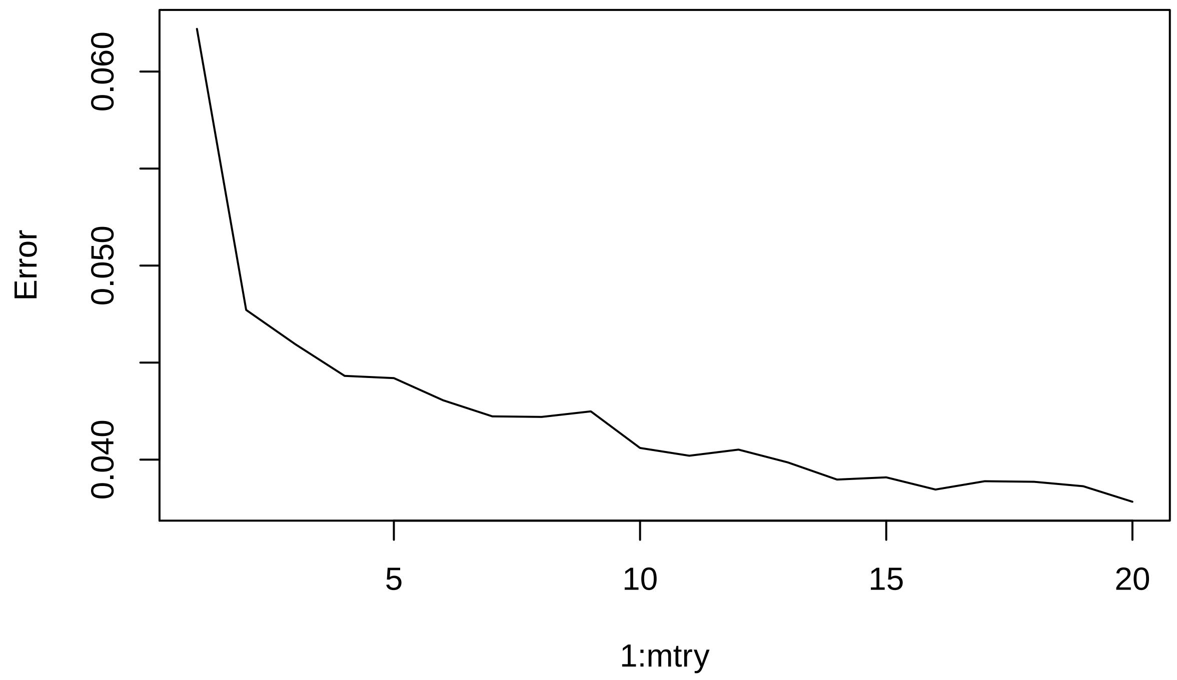
We can improve by finding the optimal mtry, which means the number of features picked randomly to create the tree.

So let's optimize mtry.

```
err = double(20)
for(mtry in 1:20){
   rf_err=randomForest(Winner_First~., data = train_set, mtry = mtry, ntree=50)
   err[mtry] = rf_err$err.rate[50]
   cat(mtry," ")
}
```

```
## 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20
```

```
plot(err, ylab= "Error", xlab= '1:mtry', type = 'l')
```



Alright, from our plot, we can see that 20 mtry is optimal to decrease the error. We'll use mtry = 20 for our optimized Random Forest model, to get the minimized error.

```
optimized_rf<- randomForest(Winner_First~., data = train_set, mtry=20, ntree=500, type = "classificatio
optimized_rf
```

```
##
## Call:
##  randomForest(formula = Winner_First ~ ., data = train_set, mtry = 20,      ntree = 500, type = "cla
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 20
##
##         OOB estimate of  error rate: 3.65%
## Confusion matrix:
##       0     1 class.error
## 0 17753   726  0.03928784
## 1   550 15970  0.03329298
```

```
pred_optimized_rf<-predict(optimized_rf, test_set)
mean(pred_optimized_rf == test_set$Winner_First)
```

```
## [1] 0.9633358
```

Now, our optimized Random Forest model produces an accuracy of 0.963!

4. Results and Conclusion

From our 6 models, we can see that our Optimized Random Forest produces the best accuracy of 0.963. To recount, our models are:
First Pokemon Model: 0.472.
GLM Model: 0.891.
Only Speed GLM Model: 0.924.
Classification Tree Model: 0.943.
Random Forest Model: 0.955.
Optimized Random Forest Model: 0.963.

We have figured out that Speed is really important to determine the winners for each Pokemon battle. However, other than Speed, we haven't really accounted for any other factors with our models, just because Speed was super important. In further works, We could figure out a way to account for Types and Legendary of each Pokemon. Another factor that we could consider is that Attack and Sp. Atk Stat separates the correlation between the both of these Stats to the win rate, as normally one Pokemon focuses on one or the other Attack Stat. If we could perhaps aggregate both Attack Stats together with the highest Attack, maybe we can see a higher correlation between these Stats and the Pokemon's Win rate.

All of this still doesn't tell us the true win rates of these Pokemon in an actual Pokemon battle, due to factors not considered in this report, such as Abilities, Moveset, EVs, Items, and more. However, that would be a lot of data with a lot more details to consider. Ultimately, this is a general way of determining the winners of a Pokemon battle and is still pretty useful to determining a good Pokemon to use to fight in a battle.