Harvardx Data Science Choose-Your-Own-Project

Ng Da Xuan

6/8/2020

INTRODUCTION

The dataset is downloaded from Kaggle (https://www.kaggle.com/abcsds/pokemon). It includes 800 observations with data including: 1) Pokemon name (Name), 2) Type 1 Element (`Type 1`), 3) Type 2 Element (`Type 2`), 4) Total Stats (Total), 5) Health Point (HP), 6) Attack point (Attack), 7) Defense point (Defense), 8) Special Attack point (Sp. Atk), 9) Special Defense point (Sp. Def), 10) Speed point (Speed), 11) Generation (Generation), and 12) Legendary class (Legendary).

It is to note that the data consists of ALL the pokemons and their stats. That is, the data is not a 'sample' of a population; the data includes all observation in the population. The task is 1) to create a linear combination of variables to predict Pokemon Types, and 2) determine the structure and latent variables of the dataset by creating linear combinations of variables (ie., reduce the dimensions of the dataset).

The key steps taken to complete the tasks were: 1) exploring the dataset and understanding the data structure, 2) exploring the relationship between the variables, 3) selecting the best model for predicting Pokemon's types (ie., 'Type 1'), and 4) obtaining a principal components analysis of the data.

The following are the libraries used in exploring the data.

```r
if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
if(!require(matrixStats)) install.packages("matrixStats", repos = "http://cran.us.r-project.org")
if(!require(Rborist)) install.packages("Rborist", repos = "http://cran.us.r-project.org")
if(!require(rpart)) install.packages("rpart", repos = "http://cran.us.r-project.org")
if(!require(devtools)) install.packages("devtools", repos = "http://cran.us.r-project.org")
if(!require(ggfortify)) install.packages("ggfortify", repos = "http://cran.us.r-project.org")
if(!require(ggplot2)) install.packages("ggplot2", repos = "http://cran.us.r-project.org")
if(!require(gridExtra)) install.packages("gridExtra", repos = "http://cran.us.r-project.org")
```

DATA PREPARATION

- Download data

```r
url <- "https://raw.githubusercontent.com/dxng-sg/HarvardxPokemon/master/datasets_121_280_Pokemon.csv"
dl <- tempfile()
download.file(url, dl)
dat <- read_csv(dl)
file.remove(dl)
```

DATA DESCRIPTION

- Exploring the dataset

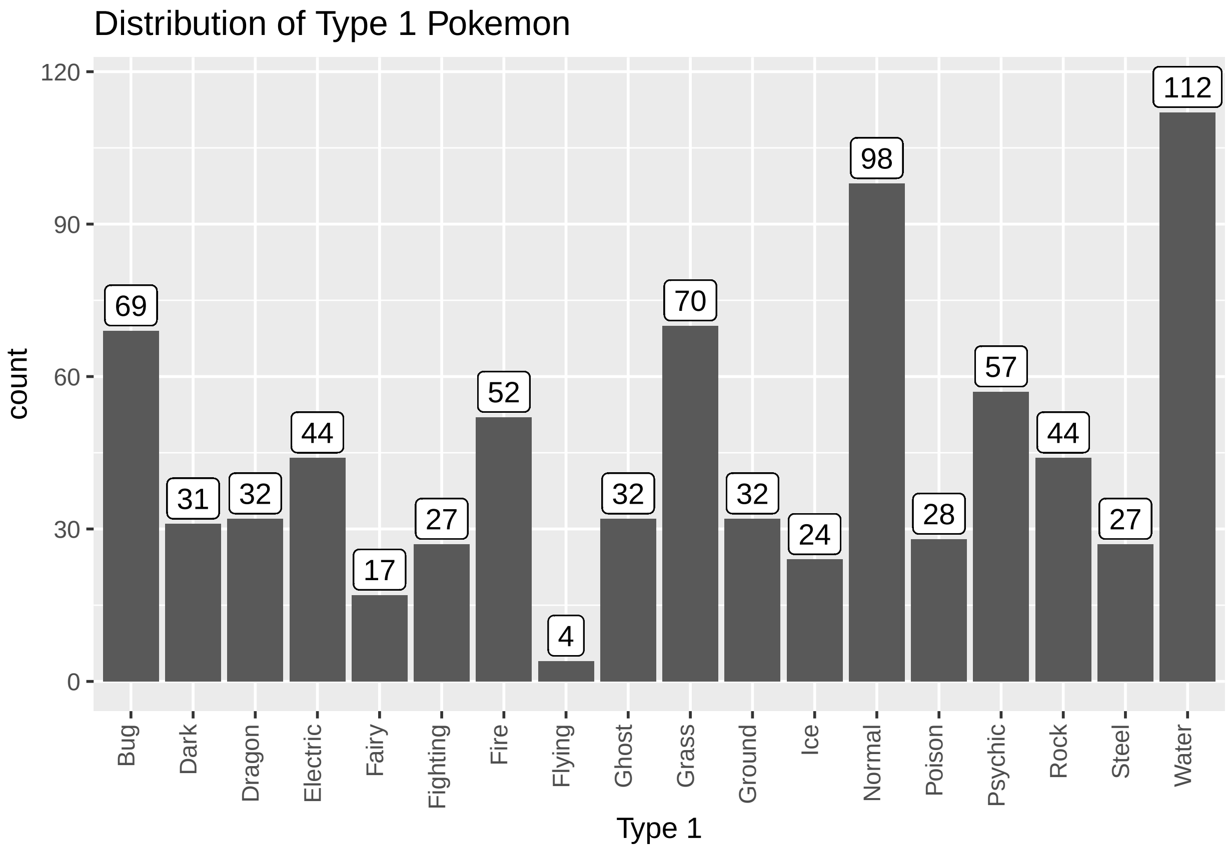
```
## To identify the data type of each variables (ie., factor, numeric, or character)
head(dat)
## # A tibble: 6 x 13
##     `#` Name  `Type 1` `Type 2` Total    HP Attack Defense `Sp. Atk` `Sp. Def`
##   <dbl> <chr> <chr>    <chr>    <dbl> <dbl>  <dbl>   <dbl>     <dbl>     <dbl>
## 1     1 Bulb~ 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     3 Venu~ Grass    Poison     625    80    100     123       122       120
## 5     4 Char~ Fire     <NA>       309    39     52      43        60        50
## 6     5 Char~ Fire     <NA>       405    58     64      58        80        65
## # ... with 3 more variables: Speed <dbl>, Generation <dbl>, Legendary <lgl>
```

There are a total of 13 variables: 1) the observation number, `#`, is of *dbl* class; 2) the name of the Pokemon, `Name`, is of *character* class; 3) the first element of Pokemon, `Type 1`, is of *character* class; 4) the second element of Pokemon, `Type 2`, is of *character* class; 5) the total stats, `Total`, is of *dbl* class; 6) the health point, `HP`, is of *dbl* class; 7) the attack stats, `Attack`, is of *dbl* class; 8) the defense stats, `Defense`, is of *dbl* class; 9) the special attack stats, `Sp. Atk`, is of *dbl* class; 10) the special defense stats, `Sp. Def`, is of *dbl* class; 11) the speed stats, `Speed`, is of *dbl* class; 12) the Generation levels, `Generation`, is of *dbl* class; & 13) the Legendary level, `Legendary`, is of *logical* class.

```
## To identify the total number of observations and variables
dim(dat)
## [1] 800  13

## The number of levels within Pokemon Types (our classification)
levels(as.factor(dat$`Type 1`))
##  [1] "Bug"      "Dark"     "Dragon"   "Electric" "Fairy"    "Fighting"
##  [7] "Fire"     "Flying"   "Ghost"    "Grass"    "Ground"   "Ice"
## [13] "Normal"   "Poison"   "Psychic"  "Rock"     "Steel"    "Water"
```

There are a total of 800 Pokemons, and they are categorised into 18 different types.

```
# To find the distribution of Type 1 pokemon
dat%>%
  group_by(`Type 1`)%>%
  mutate(count=n())%>%
  ungroup()%>%
  ggplot(aes(`Type 1`))+
  geom_bar()+
  geom_label(aes(x=`Type 1`,
               y = count+5,
               label = count))+
  theme(axis.text.x = element_text(angle = 90, hjust = 1, vjust = 0.2))+
  ggtitle("Distribution of Type 1 Pokemon")
```

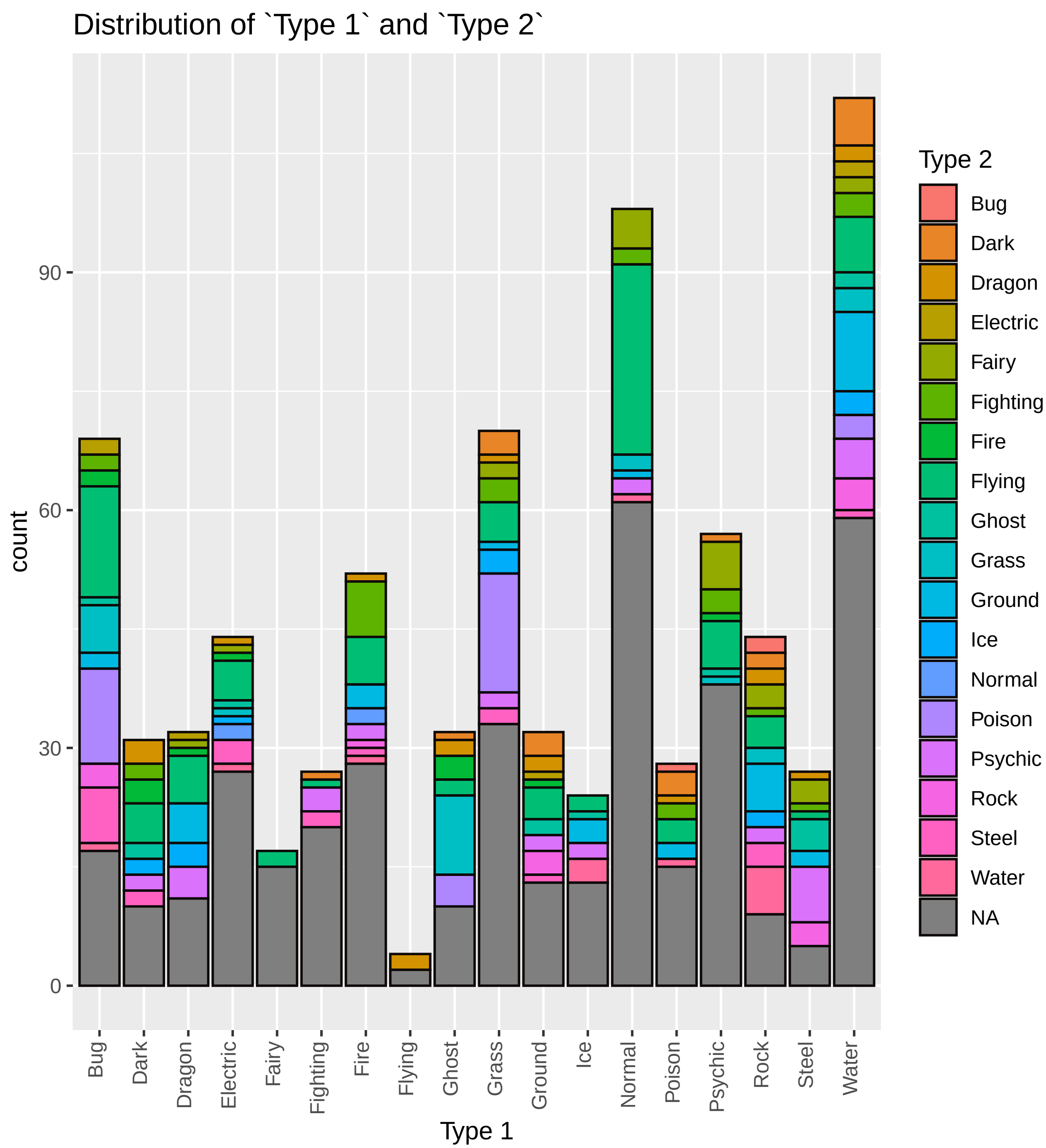


The above plot shows that there are only a few Pokemons of Type 1 'Flying' & 'Fairy'; while, there are many Pokemons of Type 1 'Water', ' Normal', 'Grass', & 'Bug'.

```
# To find the proportion of pokemon with Type 2 characteristics
mean(is.na(dat$`Type 2`) == FALSE)
## [1] 0.5175
```

Fifty-two percent of the Pokemons have a Type 2 characteristic.

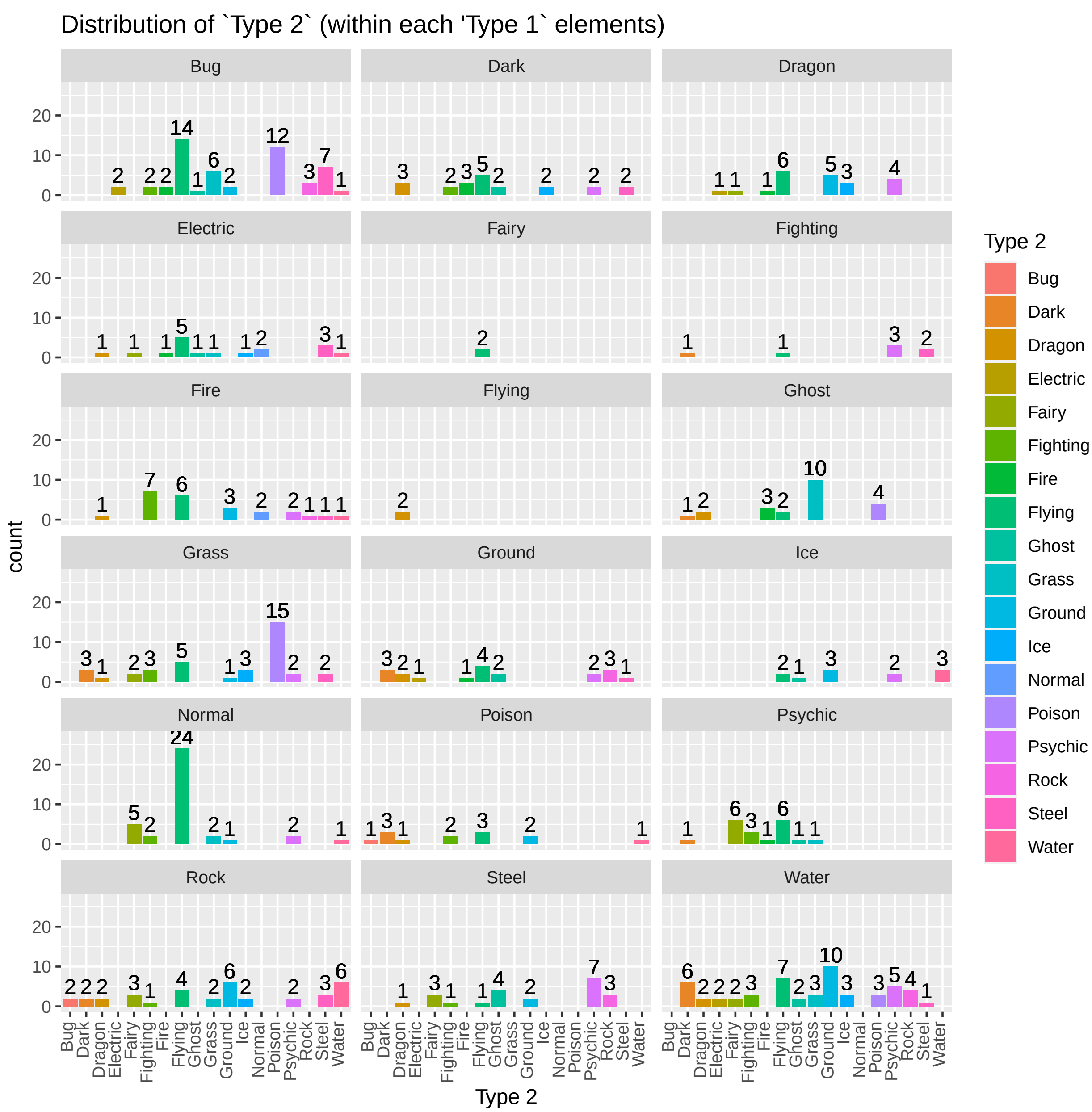
```
# To find the distribution of proportion of `Type 1` Pokemons with `Type 2`
dat%>%
  group_by(`Type 1`)%>%
  mutate(count=n())%>%
  ungroup()%>%
  ggplot(aes(`Type 1`))+
  geom_bar(aes(fill = `Type 2`),colour = "#0E0A09")+
  theme(axis.text.x=element_text(angle = 90, hjust = 1, vjust = 0.2))+
  ggtitle("Distribution of `Type 1` and `Type 2`")
```



The above plot shows that Bug, Dark, Dragon, Ghost, Rock, Steel are likely to have a `Type 2` characteristic (indicated by a larger proportion of colors within each bar), while Fairy, Fighting, Psychic are less likely to have a `Type 2` characteristic (indicated by a larger proportion of greys within each bar).

```
# To find the distribution of Type 2 within each `Type 1`
dat%>%
  filter(is.na(`Type 2`) == FALSE)%>%
  group_by(`Type 1`, `Type 2`)%>%
  mutate(count=n())%>%
  ungroup()%>%
  ggplot(aes(`Type 2`))+
```

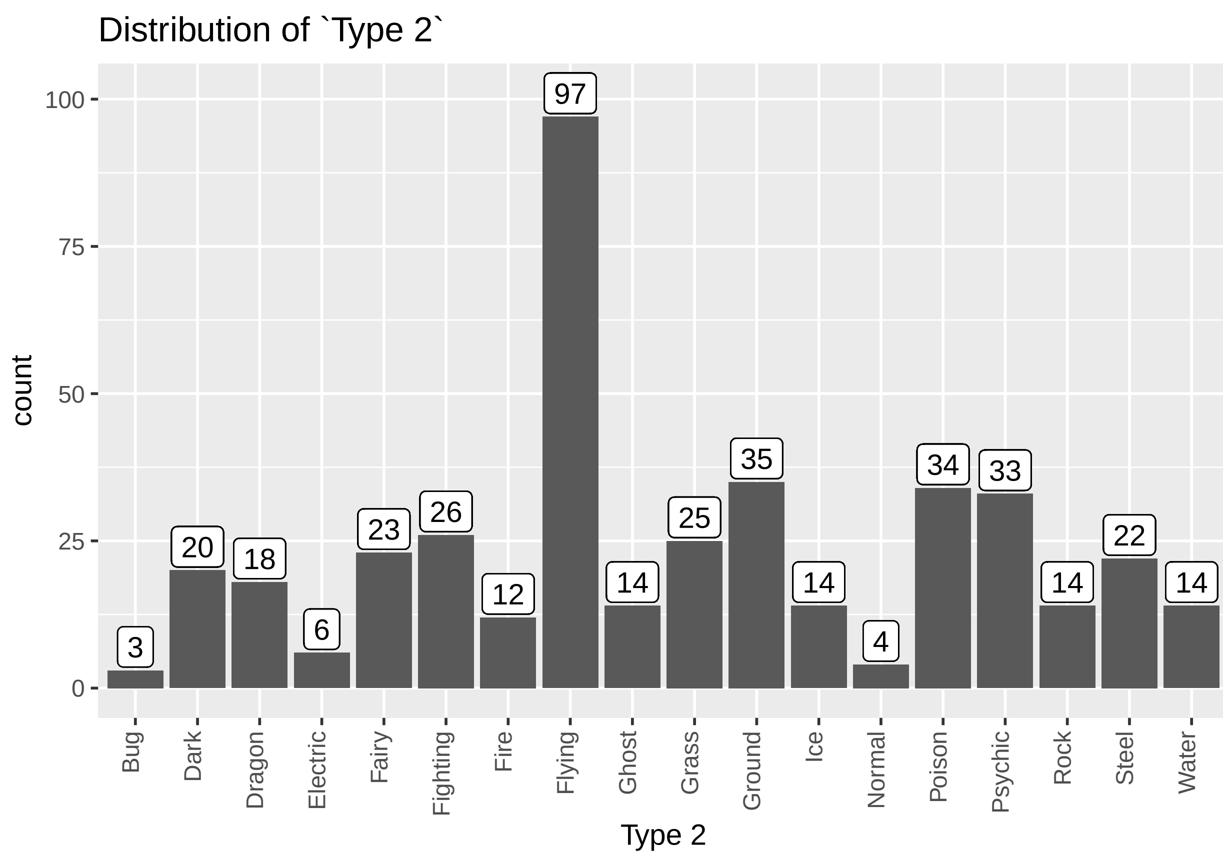
```r
geom_bar(aes(fill = `Type 2`))+
facet_wrap(~`Type 1`, ncol = 3)+
geom_text(aes(x=`Type 2`,
              y = count+3,
              label = count))+
theme(axis.text.x=element_text(angle = 90, hjust = 1, vjust = 0.2))+
ggtitle("Distribution of `Type 2` (within each 'Type 1` elements)")
```



The above plots show that within some `Type 1` class, there is an even spread of `Type 2` class. For example, within "Water" & "Rock" `Type 1` Pokemons, there is a even spread of `Type 2` class.

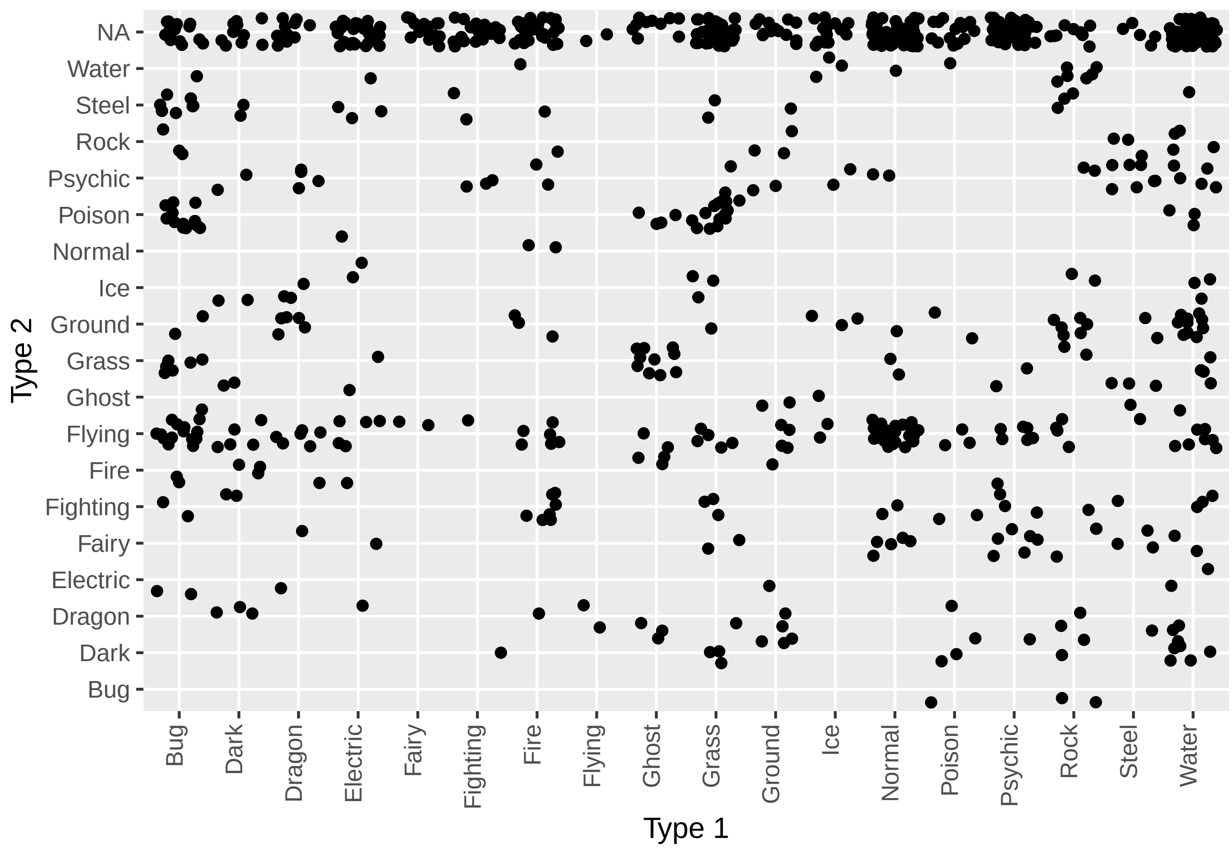
However, for some `Type 1` class, only specific `Type 2` are associated with them. For example, "Ghost" `Type 1` are closely associated with "Grass;"Bug" `Type 1` are associated with "Flying" & "Poison"; "Grass" `Type 1` are associated with "Poison"; & "Normal" `Type 1` are associated with "Flying".

```r
# To find the overall distribution of Type 2 pokemon
dat%>%
  filter(is.na(`Type 2`) == FALSE) %>%
  group_by(`Type 2`)%>%
  mutate(count = n())%>%
  ungroup%>%
  ggplot(aes(`Type 2`))+
  geom_bar()+
  geom_label(aes(x=`Type 2`,
                 y = count + 4,
                 label = count))+
  theme(axis.text.x=element_text(angle = 90, hjust = 1, vjust = 0.2))+
  ggtitle("Distribution of `Type 2`")
```



The above plot shows that the most common `Type 2` class is “Flying”.

```r
# To identify clusters of `Type 1` and `Type 2`
dat%>%
  ggplot(aes(x = `Type 1`, y = `Type 2`))+
  geom_jitter()+
  theme(axis.text.x=element_text(angle = 90, hjust = 1, vjust = 0.2))
```

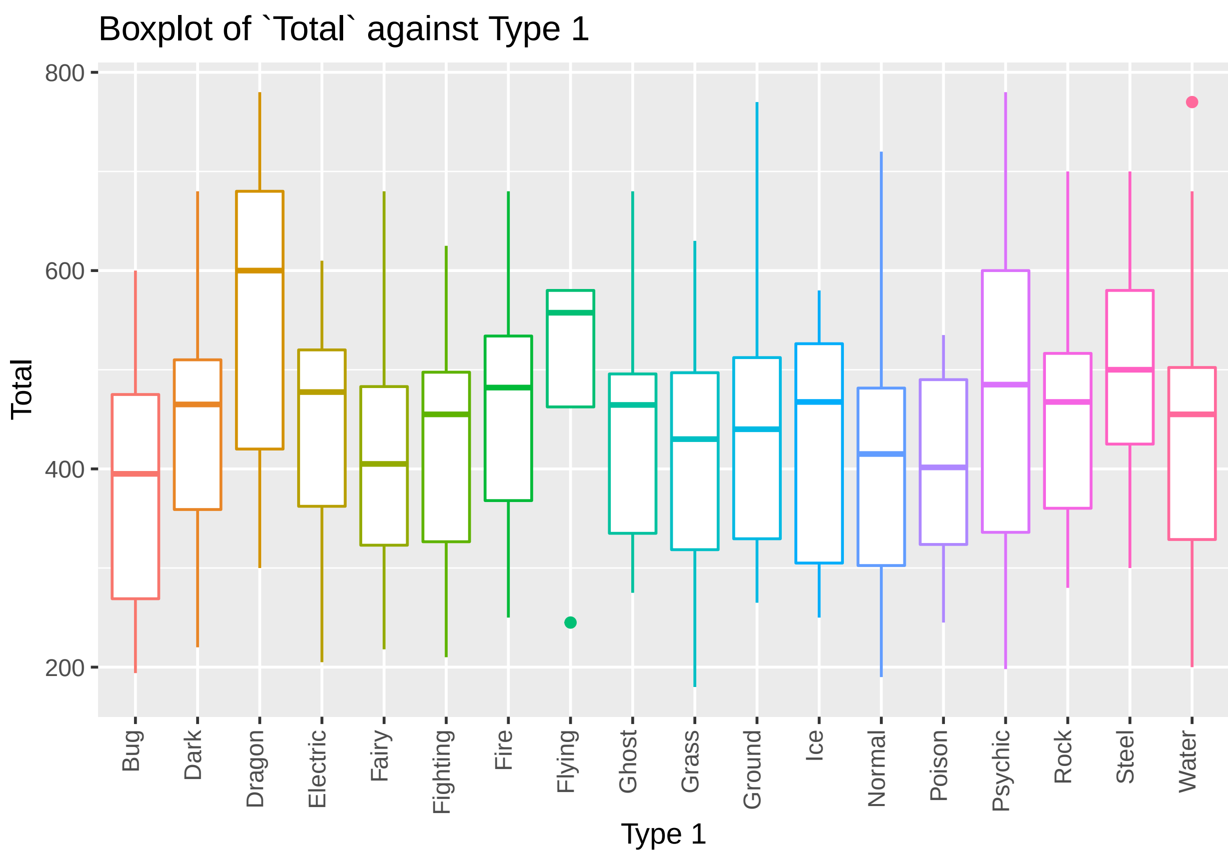


The above plot confirms several clusters. For example `Type 1` "Normal" with `Type 2` "Flying"; `Type 1` "Grass" with `Type 2` "Poison"; `Type 1` "Bug" with `Type 2` "Poison".

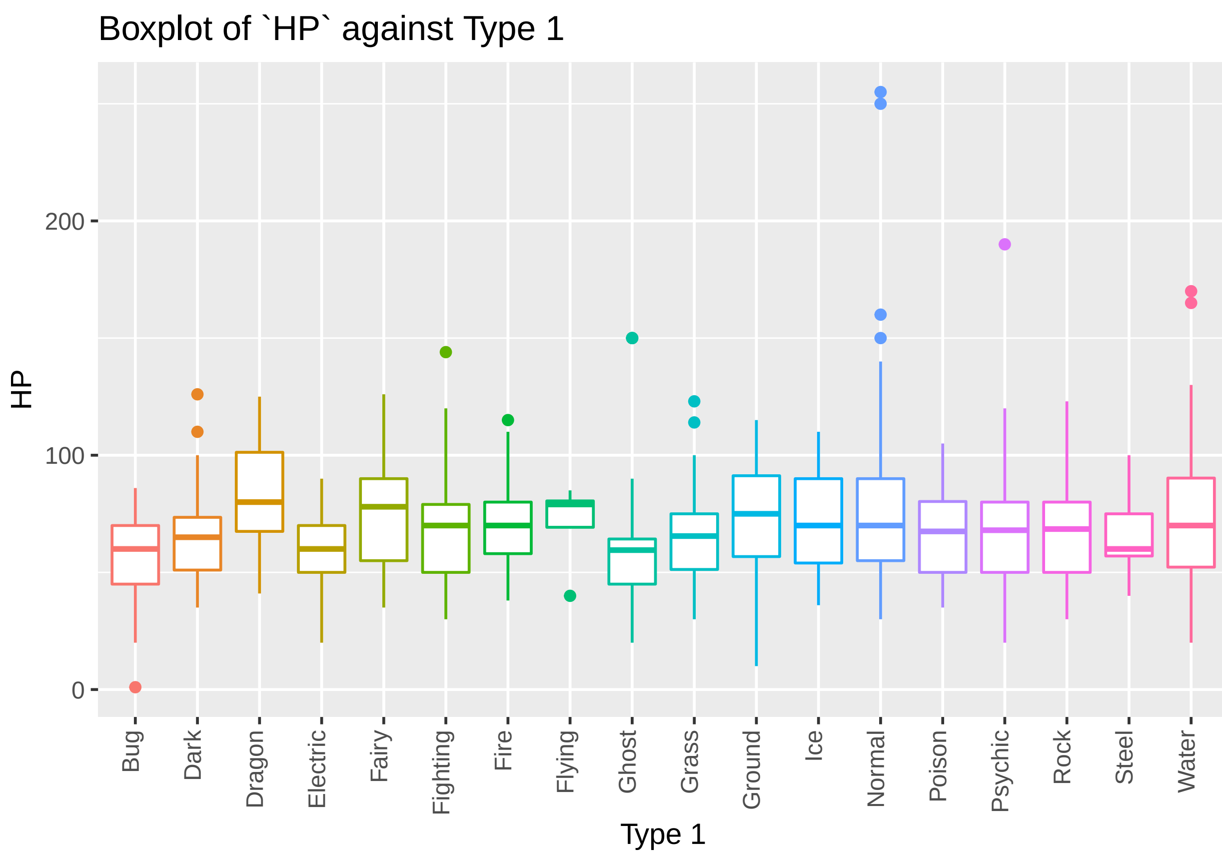
```
# The summary statistics for the variables
summary(dat[5:13])
##      Total             HP             Attack       Defense
##  Min.   :180.0   Min.   :  1.00   Min.   :  5   Min.   :  5.00
##  1st Qu.:330.0   1st Qu.: 50.00   1st Qu.: 55   1st Qu.: 50.00
##  Median :450.0   Median : 65.00   Median : 75   Median : 70.00
##  Mean   :435.1   Mean   : 69.26   Mean   : 79   Mean   : 73.84
##  3rd Qu.:515.0   3rd Qu.: 80.00   3rd Qu.:100   3rd Qu.: 90.00
##  Max.   :780.0   Max.   :255.00   Max.   :190   Max.   :230.00
##     Sp. Atk          Sp. Def          Speed          Generation
##  Min.   : 10.00   Min.   : 20.0   Min.   :  5.00   Min.   :1.000
##  1st Qu.: 49.75   1st Qu.: 50.0   1st Qu.: 45.00   1st Qu.:2.000
##  Median : 65.00   Median : 70.0   Median : 65.00   Median :3.000
##  Mean   : 72.82   Mean   : 71.9   Mean   : 68.28   Mean   :3.324
##  3rd Qu.: 95.00   3rd Qu.: 90.0   3rd Qu.: 90.00   3rd Qu.:5.000
##  Max.   :194.00   Max.   :230.0   Max.   :180.00   Max.   :6.000
##  Legendary
##  Mode :logical
##  FALSE:735
##  TRUE :65
##
##
##
```

For visualisation of the summary statistics, please refer to the boxplots attached below this paragraph.

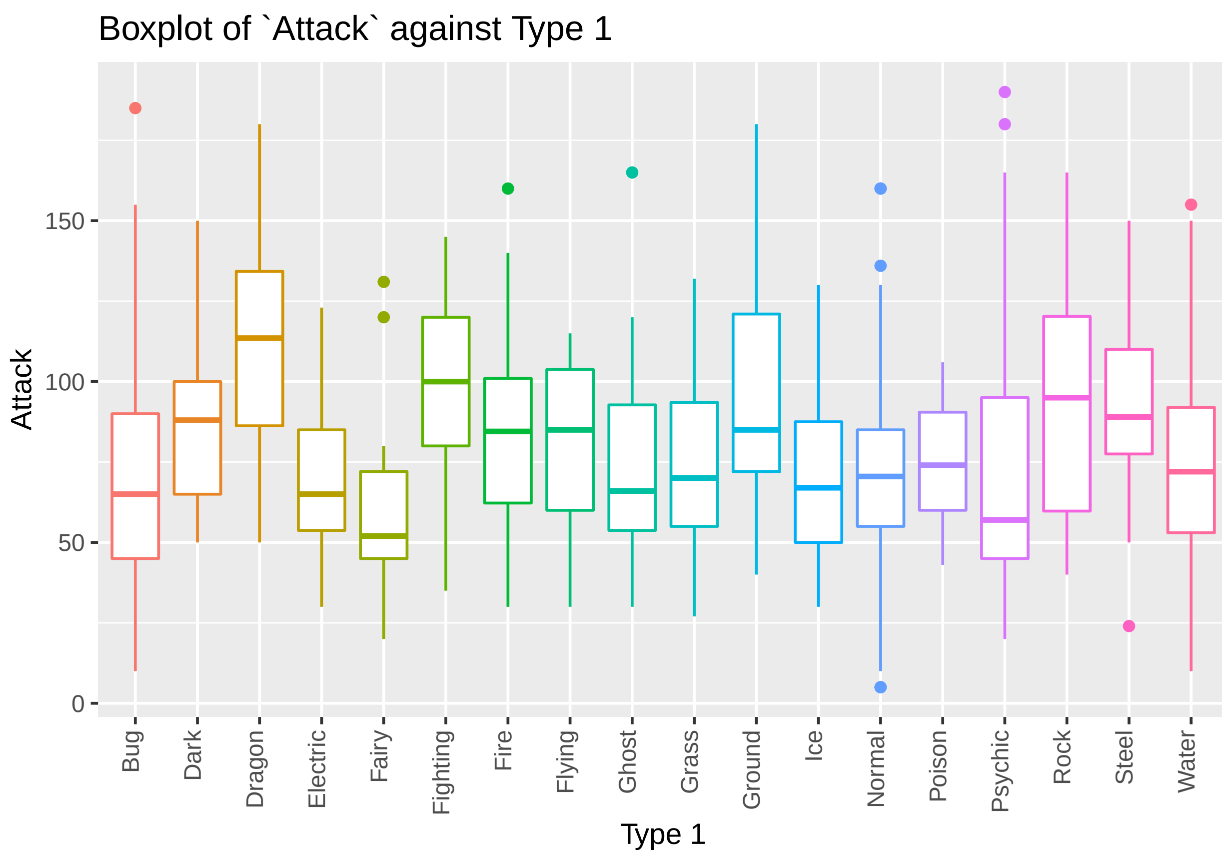
```
## Explore how the various variables are distributed across Type 1 characteristics
# Total against `Type 1`
dat%>%
  ggplot(aes(y = Total,
             x = `Type 1`))+
  geom_boxplot(aes(color=`Type 1`),
               show.legend = FALSE)+
  theme(axis.text.x=element_text(angle = 90,
                                 hjust = 1,
                                 vjust = 0.2))+
  ggtitle("Boxplot of `Total` against Type 1")
```



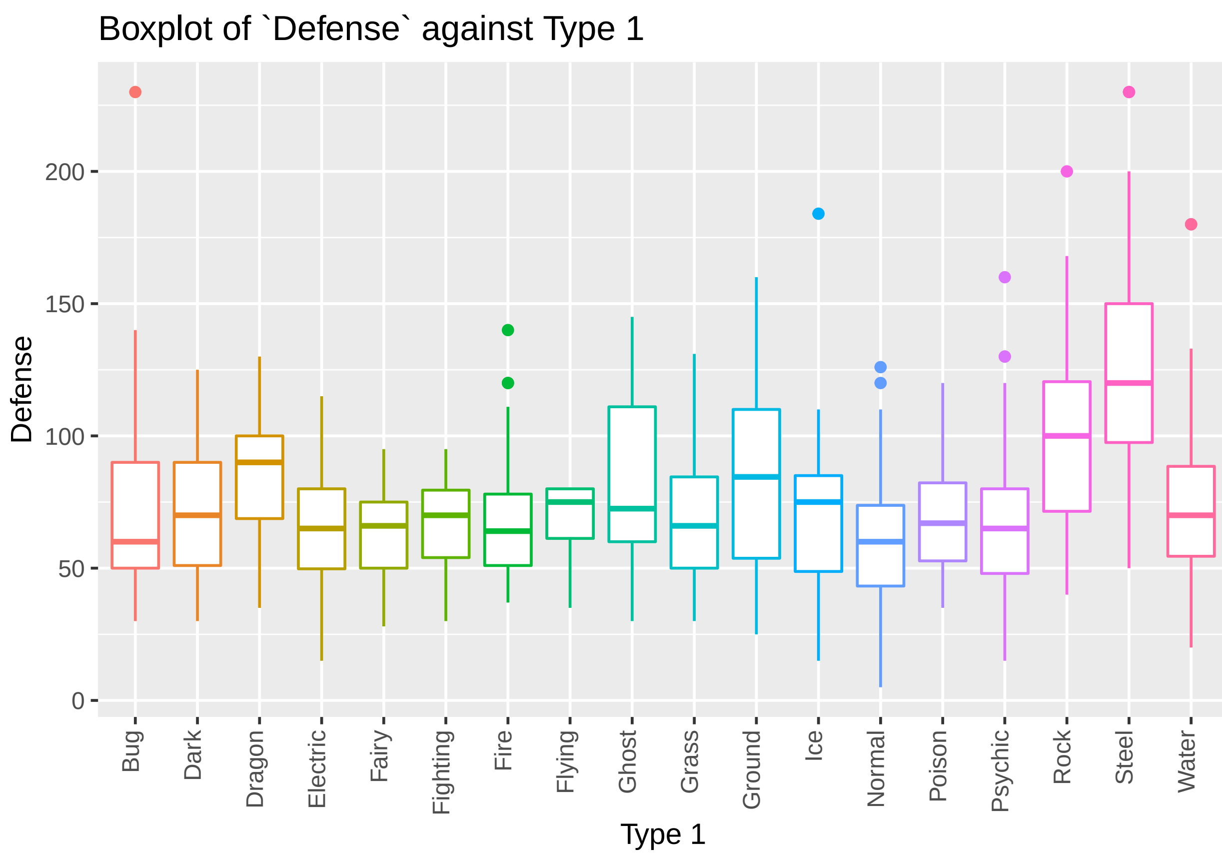

```
# HP against `Type 1`
dat%>%
  ggplot(aes(y = HP,
             x = `Type 1`))+
  geom_boxplot(aes(color=`Type 1`),
               show.legend = FALSE)+
  theme(axis.text.x=element_text(angle = 90,
                                 hjust = 1,
                                 vjust = 0.2))+
  ggtitle("Boxplot of `HP` against Type 1")
```



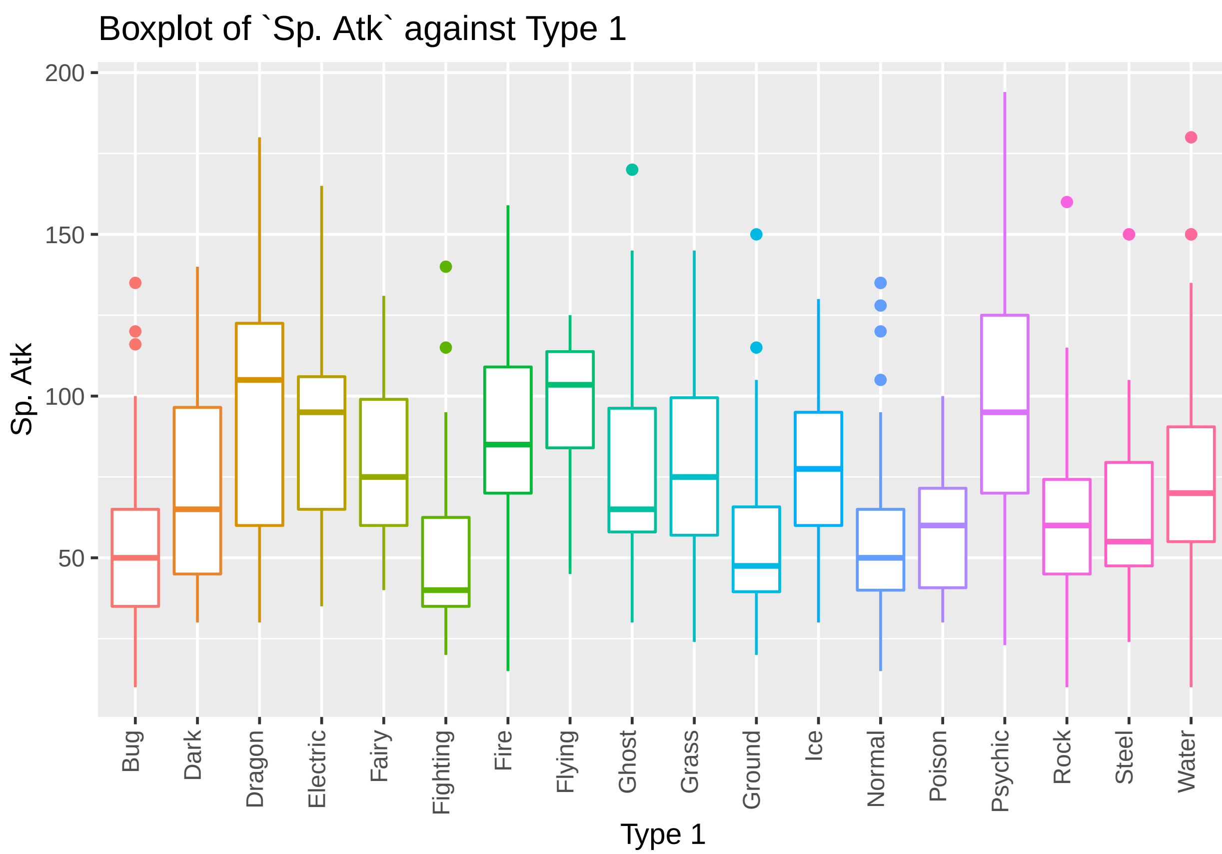

```
# Attack against `Type 1`
dat%>%
  ggplot(aes(y = Attack,
             x = `Type 1`))+
  geom_boxplot(aes(color=`Type 1`),
               show.legend = FALSE)+
  theme(axis.text.x=element_text(angle = 90, hjust = 1, vjust = 0.2))+
  ggtitle("Boxplot of `Attack` against Type 1")
```



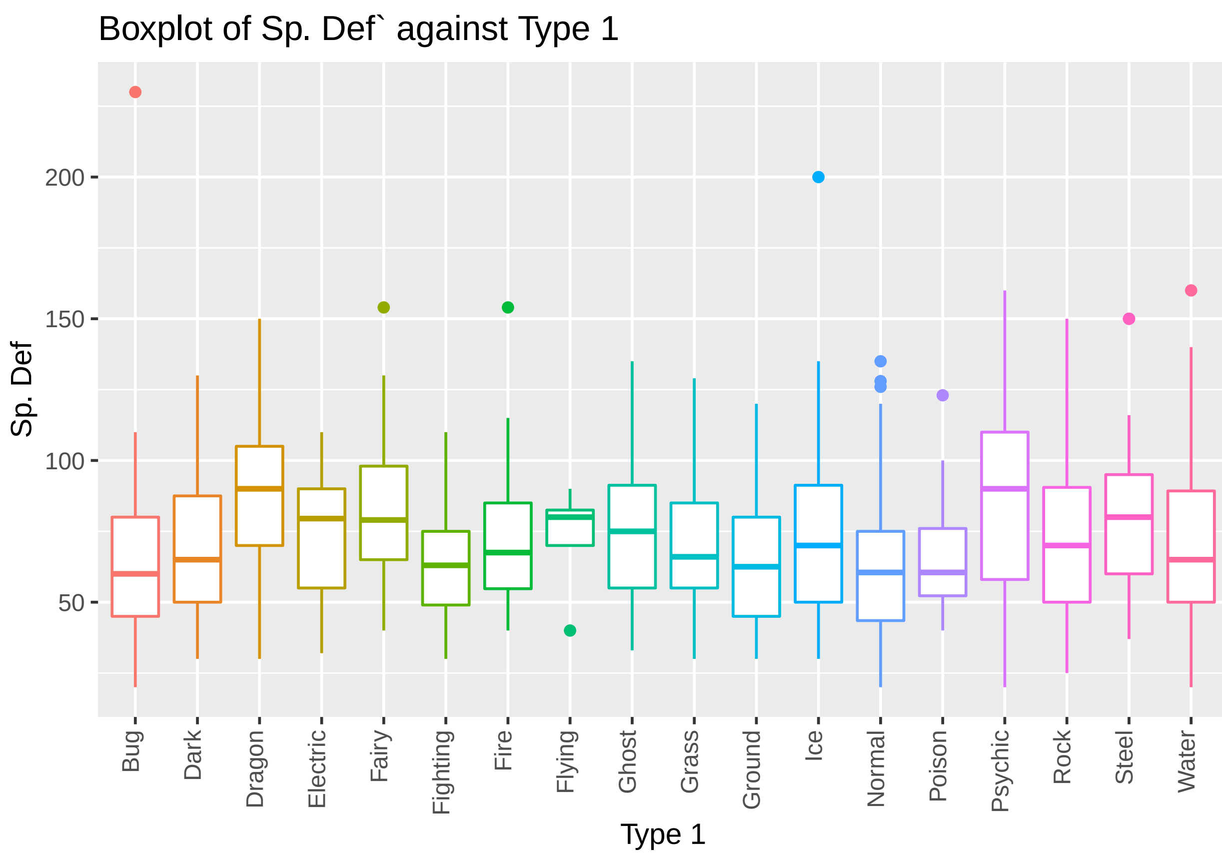

```
# Defense against `Type 1`
dat%>%
  ggplot(aes(y = Defense,
             x = `Type 1`))+
  geom_boxplot(aes(color=`Type 1`),
               show.legend = FALSE)+
  theme(axis.text.x=element_text(angle = 90, hjust = 1, vjust = 0.2))+
  ggtitle("Boxplot of `Defense` against Type 1")
```



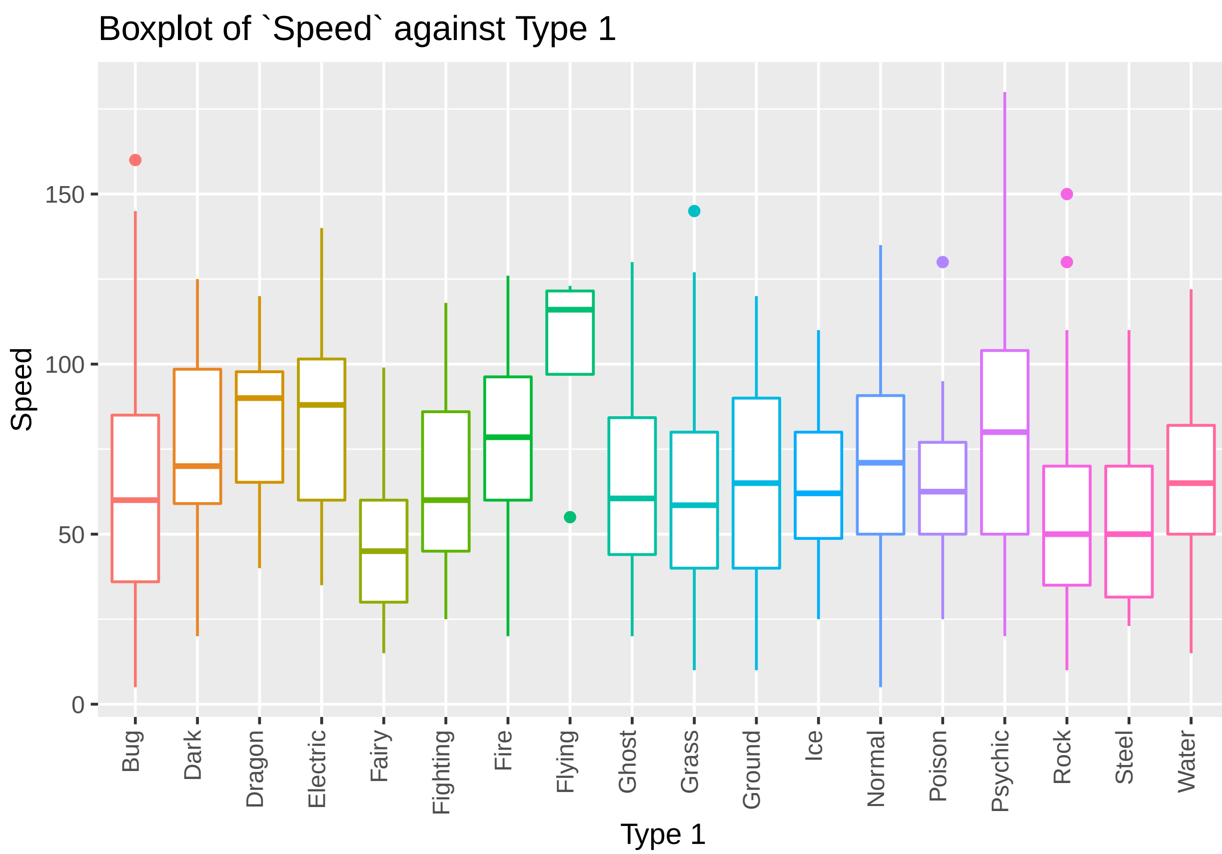

```r
# Sp. Atk against `Type 1`
dat%>%
  ggplot(aes(y = `Sp. Atk`,
             x = `Type 1`))+
  geom_boxplot(aes(color=`Type 1`),
               show.legend = FALSE)+
  theme(axis.text.x=element_text(angle = 90, hjust = 1, vjust = 0.2))+
  ggtitle("Boxplot of `Sp. Atk` against Type 1")
```




```r
# Sp. Def against `Type 1`
dat%>%
  ggplot(aes(y = `Sp. Def`,
             x = `Type 1`))+
  geom_boxplot(aes(color=`Type 1`),
               show.legend = FALSE)+
  theme(axis.text.x=element_text(angle = 90, hjust = 1, vjust = 0.2))+
  ggtitle("Boxplot of Sp. Def` against Type 1")
```

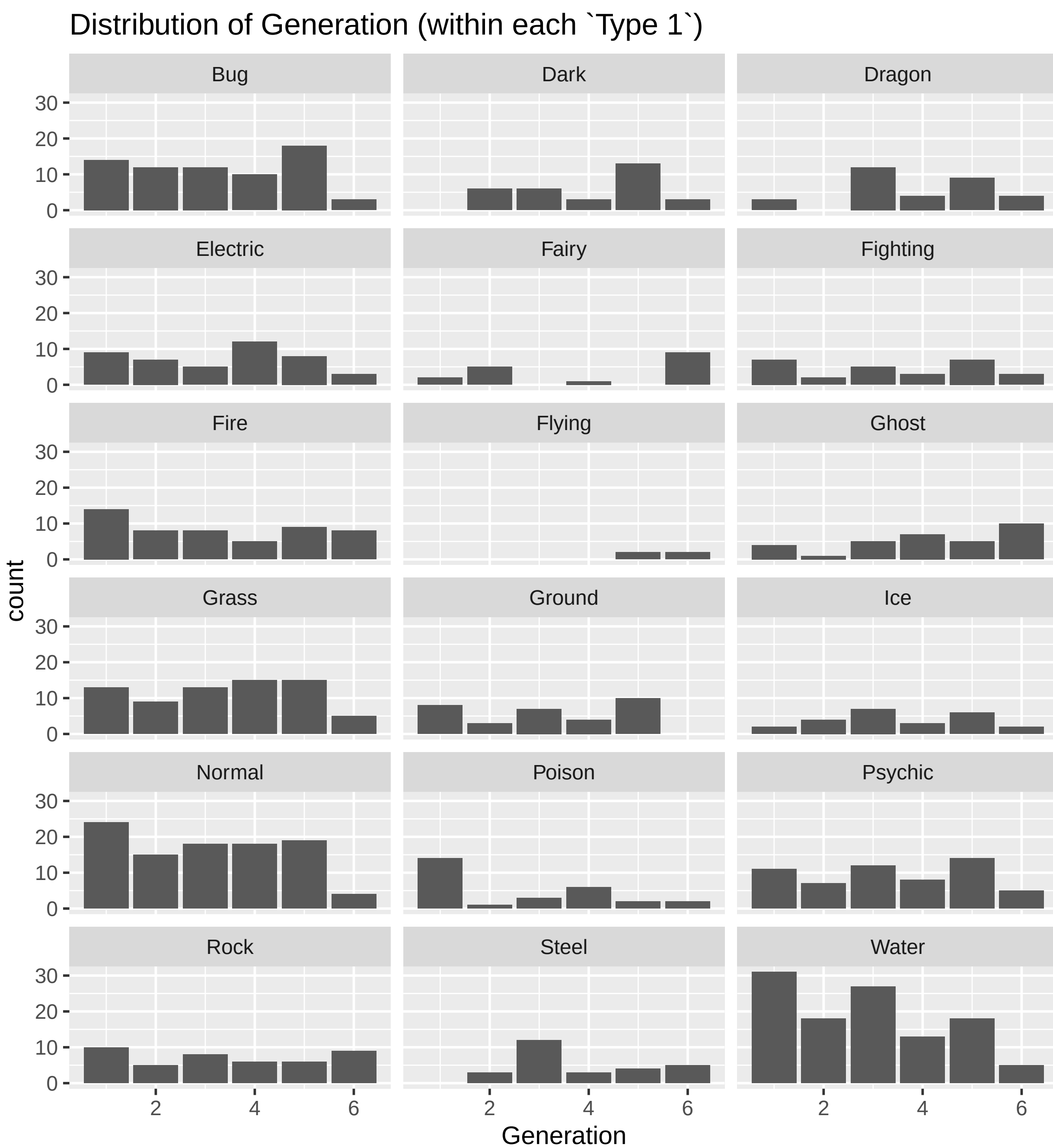



```r
# Speed against `Type 1`
dat%>%
  ggplot(aes(y = Speed,
             x = `Type 1`))+
  geom_boxplot(aes(color=`Type 1`),
               show.legend = FALSE)+
  theme(axis.text.x=element_text(angle = 90, hjust = 1, vjust = 0.2))+
  ggtitle("Boxplot of `Speed` against Type 1")
```



Just from these graphs, there are no visible difference between Pokemon Types in each of these variables.

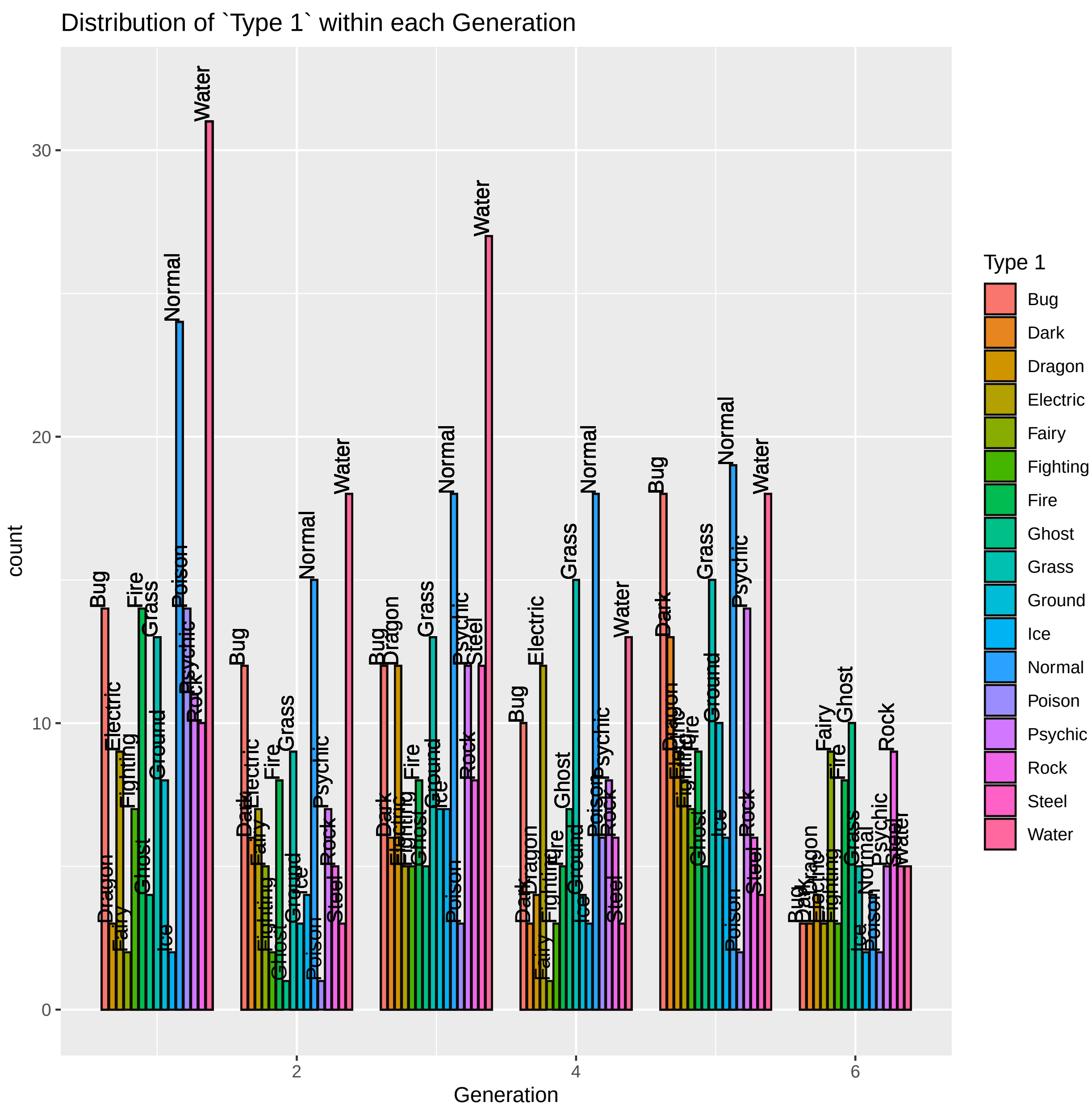
```
## Distribution of Generation within each ‘Type 1‘ class
dat%>%
  ggplot(aes(x = Generation))+
  geom_bar()+
  facet_wrap(~‘Type 1‘, ncol = 3)+
  ggtitle("Distribution of Generation (within each ‘Type 1‘)")
```



The Generation within each Pokemon Types are pretty evenly distributed.

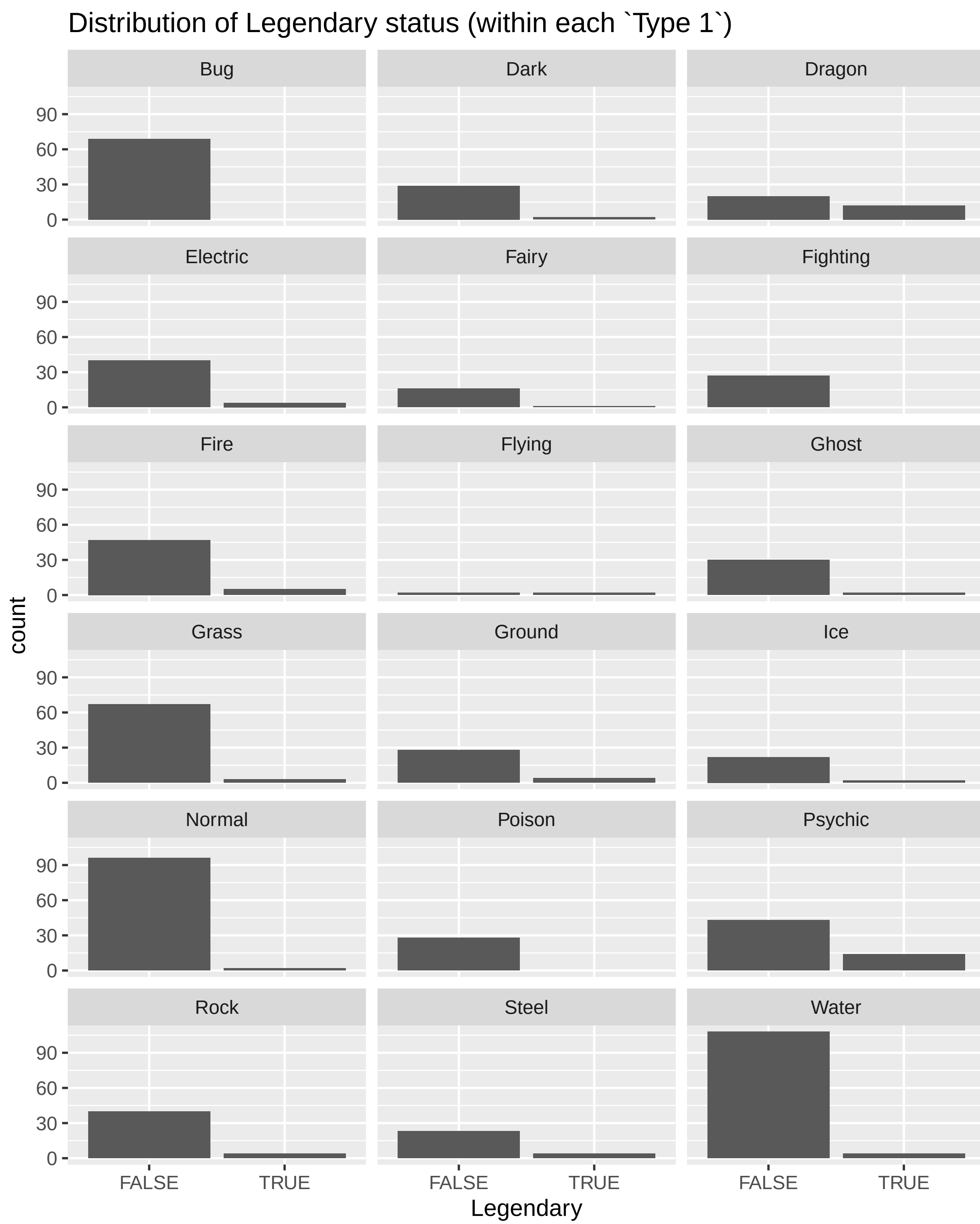
```
## Distribution of 'Type 1' class within each generation levels
dat%>%
  filter(`Type 1` != "Flying")%>%
  group_by(Generation, `Type 1`)%>%
  mutate(count=n())%>%
  ungroup()%>%
  ggplot(aes(x = Generation,
             y = count,
             fill = `Type 1`))+
```

```
geom_bar(stat="identity",
         colour = "#0E0A09",
         width = 0.7,
         position = position_dodge(width = 0.8))+
geom_text(position=position_dodge(width=0.8),
          aes(label=‘Type 1‘),
          hjust=0,
          vjust=0,
          angle=90)+
ylim(0, 32)+
ggtitle("Distribution of ‘Type 1‘ within each Generation")
```



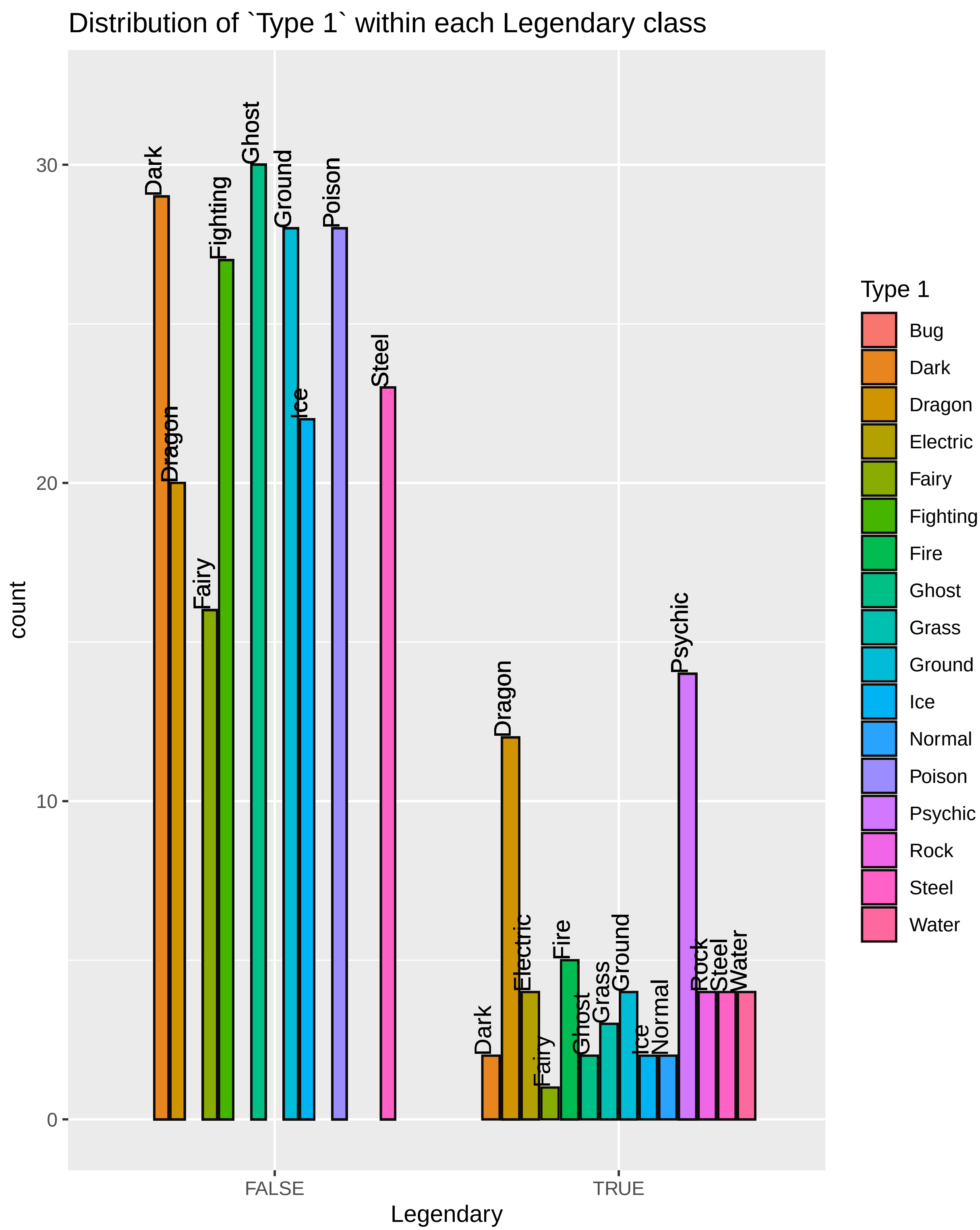
There are more “Water” and “Normal” types Pokemons in Generation levels 1 - 5, while there are more “Rock”, “Ghost”, and “Fairy” types Pokemons in Generation level 6.

```r
## Distribution of Legendary within each `Type 1` class
dat%>%
  ggplot(aes(x = Legendary))+
  geom_bar()+
  facet_wrap(~`Type 1`, ncol = 3)+
  ggtitle("Distribution of Legendary status (within each `Type 1`)")
```



It is more likely to capture a non-legendary pokemon than a legendary pokemon among all `Type 1`. It also appeared that you will have higher chance of capturing a legendary pokemon when you capture a "Dragon" than the other types.

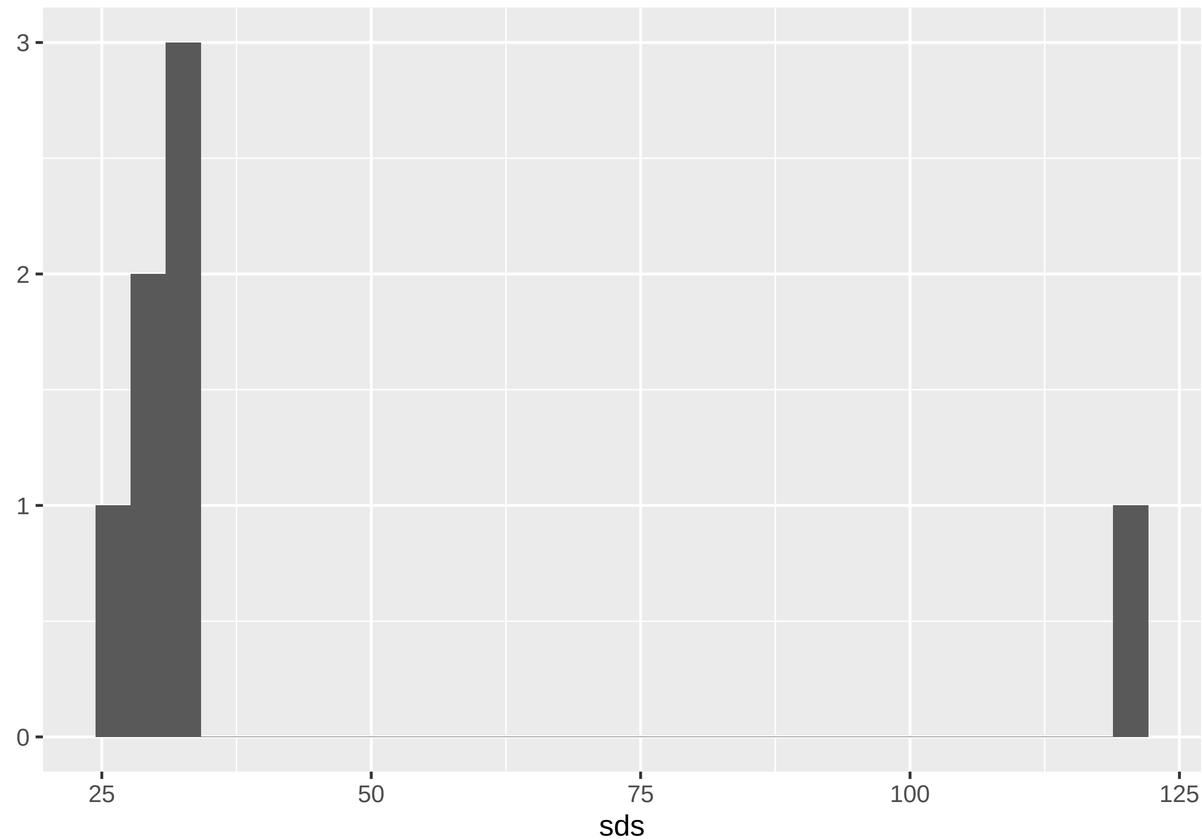
```
## Distribution of `Type 1` class within each Legendary level
dat%>%
  filter(`Type 1` != "Flying")%>%
  group_by(Legendary, `Type 1`)%>%
  mutate(count=n())%>%
  ungroup()%>%
  ggplot(aes(x = Legendary,
             y = count,
             fill = `Type 1`))+
  geom_bar(stat="identity",
           colour = "#0E0A09",
           width = 0.7,
           position = position_dodge(width = 0.8))+
  geom_text(position=position_dodge(width=0.8),
            aes(label=`Type 1`),
            hjust=0,
            vjust=0,
            angle=90)+
  ylim(0, 32)+
  ggtitle("Distribution of `Type 1` within each Legendary class")
```



Among Legendary pokemon, there are more “Psychic” and “Dragon” than the other types.

```
## Identifying the variables that has the least variation
x <- as.matrix(dat[5:11])
y <- factor(dat$`Type 1`)
sds<-colSds(x)
```

```r
qplot(sds)
```



Continuous variables(ie., Total, HP, Attack, Defense, 'Sp. Atk', 'Sp. Def', Speed) are chosen to build the prediction model as they explained for the most variance within the dataset. Within the 7 chosen variables, 'Total' carries the highest sds.

METHOD AND ANALYSIS

Finding the most accurate class prediction model

- Knn Model

```r
## Knn model
set.seed(2007, sample.kind="Rounding")
fit_knn <- knn3(y ~ x, data = dat, k = 1)
y_hat_knn<-predict(fit_knn, dat, type = "class")
cm <- confusionMatrix(y_hat_knn,y)
cm$overall["Accuracy"]
cm$byClass[,1:2]
## Accuracy
##   0.9875
##                  Sensitivity Specificity
## Class: Bug         1.0000000   1.0000000
## Class: Dark        1.0000000   1.0000000
## Class: Dragon      1.0000000   1.0000000
```

```
## Class: Electric   0.9772727   0.9986772
## Class: Fairy      1.0000000   1.0000000
## Class: Fighting   1.0000000   1.0000000
## Class: Fire       0.9807692   0.9973262
## Class: Flying     0.7500000   0.9987437
## Class: Ghost      1.0000000   1.0000000
## Class: Grass      0.9571429   0.9986301
## Class: Ground     1.0000000   1.0000000
## Class: Ice        1.0000000   1.0000000
## Class: Normal     1.0000000   1.0000000
## Class: Poison     1.0000000   1.0000000
## Class: Psychic    1.0000000   0.9959623
## Class: Rock       1.0000000   1.0000000
## Class: Steel      0.9629630   1.0000000
## Class: Water      0.9732143   0.9970930
```

Using knn model (with a k = 1) predicts a high accuracy (99.4%), indicating that each Pokemon can be uniquely predicted by the variables. The model reveals that “Electric”, “Fairy”, “Fire”, “Flying”, “Water” are harder to detect; while “Dark”, “Electric”, “Fire”, “Flying”, “Grass” are often incorrectly detected.

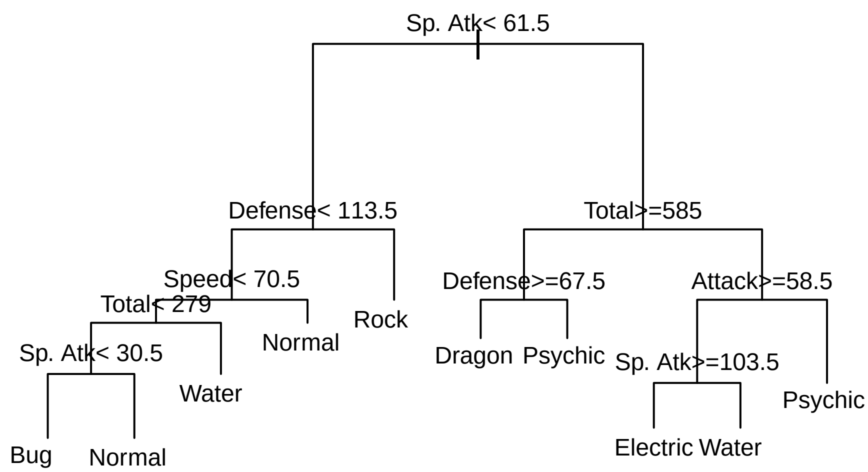
*Qda Model

```
## QDA model
fit_lda <- train(`Type 1` ~ HP + Attack + Defense + `Sp. Atk` + `Sp. Def` + Speed + Generation + Legenda
                 method = "lda",
                 data = dat)
y_hat_lda <- predict(fit_lda, dat)
cm <- confusionMatrix(y_hat_lda,y)
cm$overall["Accuracy"]
cm$byClass[,1:2]
## Accuracy
##  0.27125
##                  Sensitivity Specificity
## Class: Bug        0.27536232   0.9097127
## Class: Dark       0.00000000   0.9973992
## Class: Dragon     0.28125000   0.9570312
## Class: Electric   0.36363636   0.9537037
## Class: Fairy      0.29411765   0.9897829
## Class: Fighting   0.22222222   0.9702458
## Class: Fire       0.09615385   0.9772727
## Class: Flying     0.25000000   0.9937186
## Class: Ghost      0.03125000   0.9960938
## Class: Grass      0.11428571   0.9479452
## Class: Ground     0.15625000   0.9947917
## Class: Ice        0.00000000   1.0000000
## Class: Normal     0.63265306   0.8646724
## Class: Poison     0.00000000   1.0000000
## Class: Psychic    0.14035088   0.9676985
## Class: Rock       0.29545455   0.9563492
## Class: Steel      0.33333333   0.9676585
## Class: Water      0.44642857   0.7500000
```

The accuracy of qda model is low and is not a good model for the dataset.

*Classification and Regression Tree (CART)

```
## Use decision tree
set.seed(1985, sample.kind="Rounding") #set.seed is created to facilitate discussion
fit_rpart<-train(x,y,
                   method = "rpart",
                   tuneGrid=data.frame(cp=seq(0,0.1,
                                               len=25)),
                   control = rpart.control(minsplit = 0))
confusionMatrix(predict(fit_rpart,x),y)$overall["Accuracy"]
plot(fit_rpart$finalModel, margin = 0.1)
text(fit_rpart$finalModel, cex = 0.75)
```



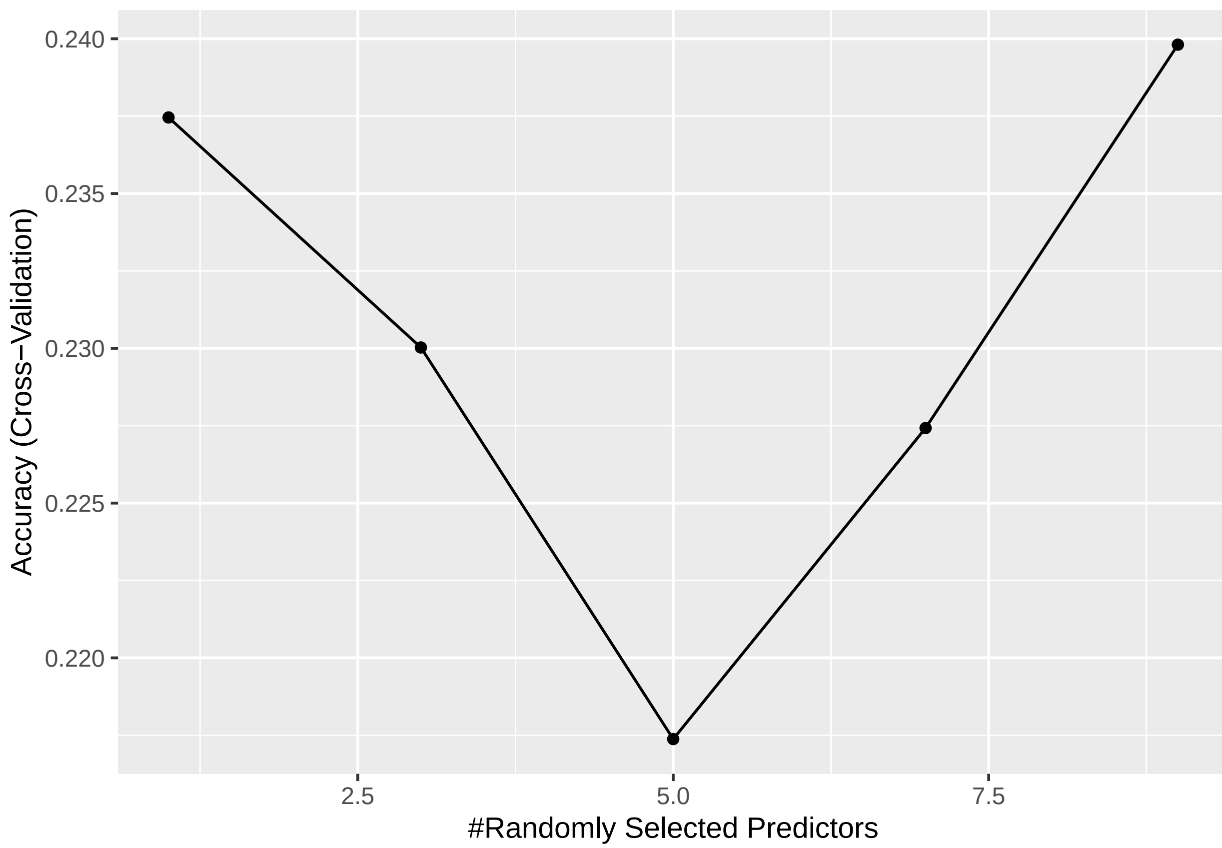

```
## Accuracy
##     0.26
```

Using CART produces a decision tree that gives us some insight into the properties (e.g., defense, attack, speed, etc) of each type of Pokemon, however the model is low in accuracy.

- Random Forest

```
#Use Random Forest
set.seed(2007, sample.kind="Rounding")
grid <- data.frame(mtry = c(1, 3, 5, 7, 9))
control<-trainControl(method = "cv", number = 5)
fit_rf <- train(x, y,
```

```
                  method = "rf",
                  ntree = 50,
                  trControl = control,
                  tuneGrid = grid,
                  nSamp = 700)
ggplot(fit_rf)
```




```
fit_rf$bestTune
y_hat_rf <- predict(fit_rf,dat)
cm <- confusionMatrix(y_hat_rf,y)
cm$overall["Accuracy"]
```

```
##   mtry
## 5    9
## Accuracy
##   0.9875
```

```
##Finding the class with the lowest sensitivity and specificity
cm$byClass[,1:2]
```

```
##                   Sensitivity Specificity
## Class: Bug          1.0000000   1.0000000
## Class: Dark         1.0000000   0.9986996
## Class: Dragon       1.0000000   1.0000000
```

```
## Class: Electric   1.0000000   0.9986772
## Class: Fairy      0.9411765   1.0000000
## Class: Fighting   1.0000000   1.0000000
## Class: Fire       0.9615385   1.0000000
## Class: Flying     0.7500000   1.0000000
## Class: Ghost      1.0000000   1.0000000
## Class: Grass      0.9571429   1.0000000
## Class: Ground     1.0000000   1.0000000
## Class: Ice        0.9583333   1.0000000
## Class: Normal     1.0000000   1.0000000
## Class: Poison     1.0000000   1.0000000
## Class: Psychic    1.0000000   0.9959623
## Class: Rock       1.0000000   1.0000000
## Class: Steel      0.9629630   1.0000000
## Class: Water      0.9910714   0.9927326
```

RandomForest model carries a high accuracy. Similar to knn model, the model finds “Fairy”, “Fire”, “Flying”, “Grass”, “Ice”, “Steel”, “Water” harder to detect; while “Dark”, “Electric”, “Psychic”, “Water” are often incorrectly detected.

```
##Identifying the variables that are the most predictive of 'Type 1' class
varImp(fit_rf)
## rf variable importance
##
##          Overall
## Speed     100.00
## Attack     88.17
## Sp. Atk    85.75
## HP         78.59
## Defense    72.45
## Total      43.74
## Sp. Def     0.00
```

Using RandomForest, We have identified ‘Speed’ as the most important variable for predicting ‘Type 1’.

Results

1) Fitting the best class prediction model

It is to note that this project did not partition the dataset (ie., into ‘Training set’ and ‘Testing set’), as there is no need for us to train a model to predict an unknown population parameter. There is also no need to use a sample to estimate a population parameter as the dataset includes all observation present in the population.

As shown below, rf model is the best predictive model.

```
cm <- confusionMatrix(y_hat_rf,y)
cm$overall["Accuracy"]
## Accuracy
##   0.9875
```

2) Performing PCA

```
## Correlation between the continuous variables
cor(dat[5:11])
```

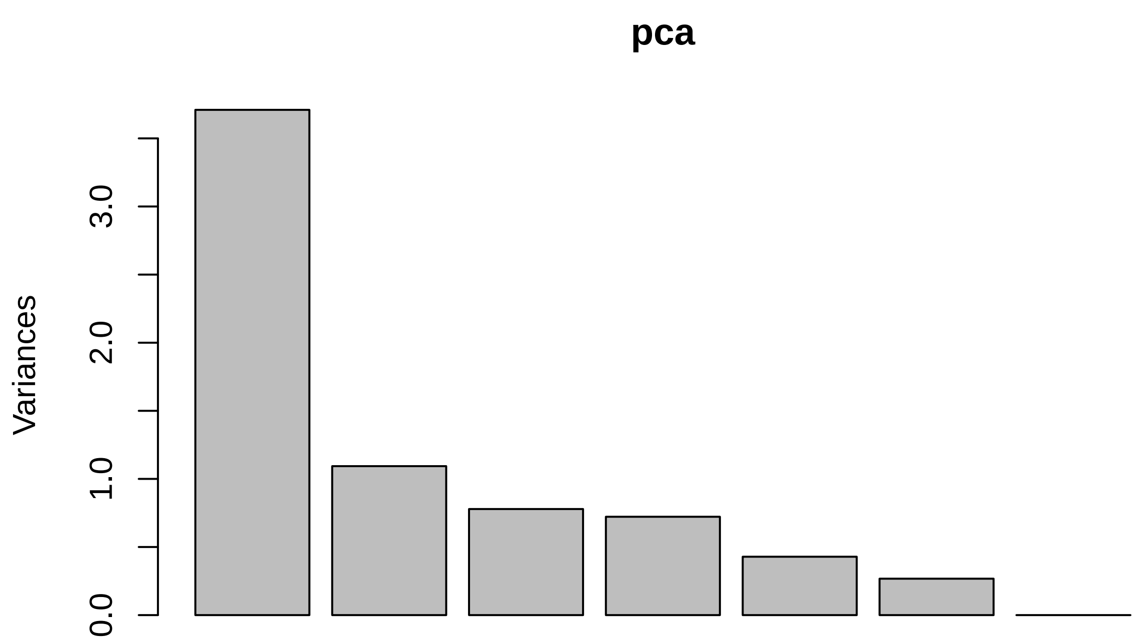
```
##             Total        HP    Attack   Defense   Sp. Atk   Sp. Def     Speed
## Total   1.0000000 0.6187484 0.7362107 0.6127874 0.7472499 0.7176095 0.5759427
## HP      0.6187484 1.0000000 0.4223860 0.2396223 0.3623799 0.3787181 0.1759521
## Attack  0.7362107 0.4223860 1.0000000 0.4386871 0.3963618 0.2639896 0.3812397
## Defense 0.6127874 0.2396223 0.4386871 1.0000000 0.2235486 0.5107466 0.0152266
## Sp. Atk 0.7472499 0.3623799 0.3963618 0.2235486 1.0000000 0.5061214 0.4730179
## Sp. Def 0.7176095 0.3787181 0.2639896 0.5107466 0.5061214 1.0000000 0.2591331
## Speed   0.5759427 0.1759521 0.3812397 0.0152266 0.4730179 0.2591331 1.0000000
```

From the correlation coefficient, most of the variables are moderately correlated with each other ($r > 0.3$).

```
# Perform PCA to isolate the factors, explaining the high correlations among variables
pca <- prcomp(x, scale = TRUE)
summary(pca)
## Importance of components:
##                           PC1    PC2    PC3    PC4     PC5     PC6       PC7
## Standard deviation     1.9260 1.0458 0.8825 0.8498 0.65465 0.51716 1.727e-15
## Proportion of Variance 0.5299 0.1562 0.1113 0.1032 0.06122 0.03821 0.000e+00
## Cumulative Proportion  0.5299 0.6862 0.7974 0.9006 0.96179 1.00000 1.000e+00
pca$rotation
##               PC1          PC2          PC3         PC4          PC5
## Total   0.5188676  0.007123997 -0.006491292  0.03641716  0.005678276
## HP      0.3293192 -0.088082002 -0.466660521 -0.73046178  0.218990199
## Attack  0.3780749  0.008991825 -0.593397272  0.38872762 -0.192640251
## Defense 0.3131456 -0.631028413  0.069250532  0.40945591  0.057520652
## Sp. Atk 0.3900939  0.301919048  0.308868500 -0.16014375 -0.736993712
## Sp. Def 0.3800275 -0.242611805  0.568862827 -0.19611196  0.298367017
## Speed   0.2910850  0.666256332  0.079152993  0.28675801  0.528592045
##                 PC6        PC7
## Total    0.03135149 -0.8534297
## HP       0.22691029  0.1816563
## Attack  -0.51314874  0.2309051
## Defense  0.52798713  0.2218427
## Sp. Atk  0.19564011  0.2327898
## Sp. Def -0.55356636  0.1979778
## Speed    0.24642644  0.2067393
```

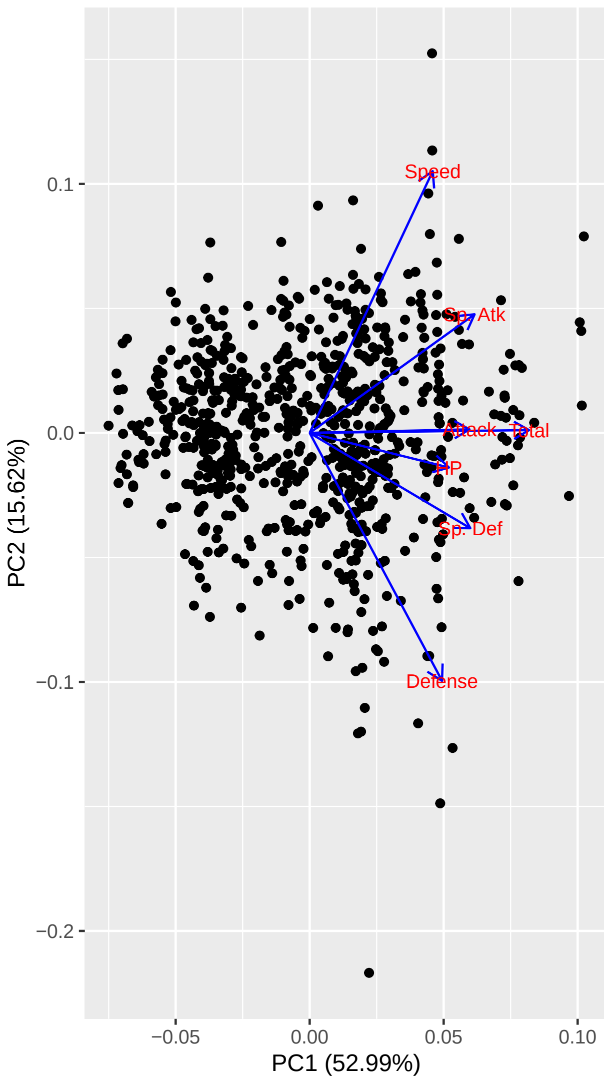
Most of the variance (53%) is explained by 2 components.

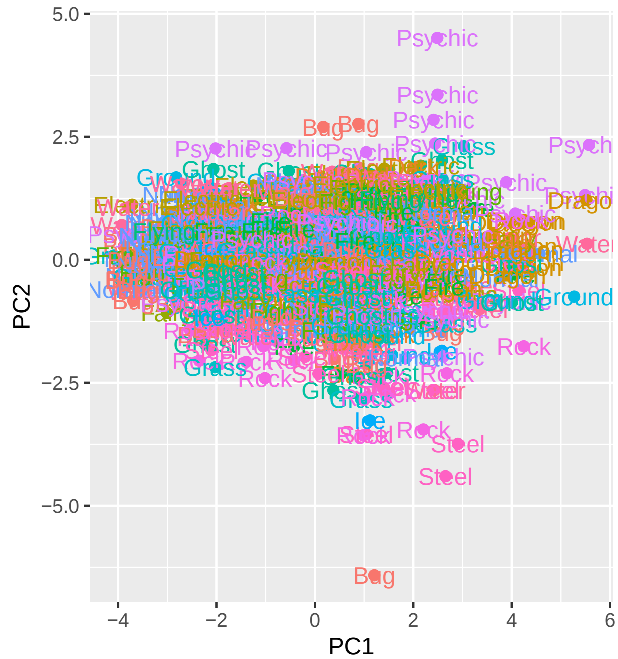
```
screeplot(pca)
```



Scree plot shows that 2 components are needed to explain most of the data variance.

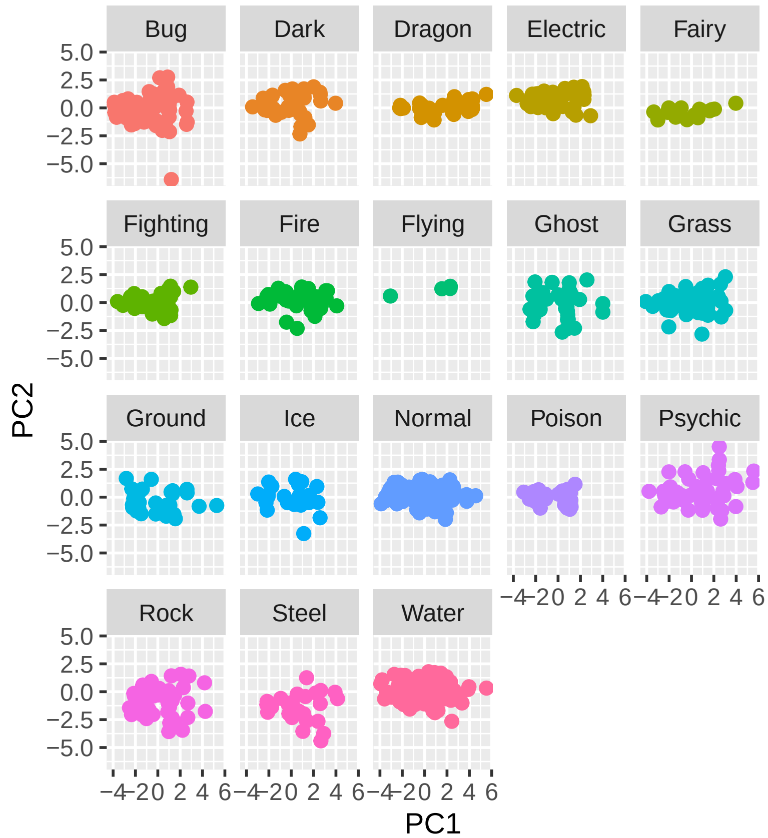
```
plot_1 <- autoplot(pca,
         data=x,
         loadings=TRUE,
         loadings.colour='blue',
         loadings.label = TRUE,
         loadings.label.size=3)


plot_2 <- data.frame(pca$x[,1:2],Type_1 = factor(dat$`Type 1`))%>%
  ggplot(aes(PC1, PC2,
             fill = Type_1,
             color = Type_1))+
  geom_point(size=2,
             shape = 21,show.legend = FALSE)+
  geom_text(aes(label = Type_1), show.legend = FALSE)+
  coord_fixed(ratio=1)

grid.arrange(plot_1,plot_2, ncol=2)
```





The above plot shows that most of the Pokemons can be separated by variables ‘Total’, ‘Attack’ & ‘Sp. Atk’ on the x-axis, and by variables ‘Speed’ & ‘Defense’ on the y-axis. It appears that Component 1 separates between Strong Attacking Types Pokemons from Weak Attacking Types Pokemons, while Component 2 separates between Fast Pokemons from Strong Defense Type Pokemons.

```
data.frame(pca$x[,1:2],Type_1 = factor(dat$`Type 1`))%>%
  ggplot(aes(PC1, PC2,
             fill = Type_1,
             color = Type_1))+
  facet_wrap(~Type_1)+
  geom_point(size=2,
             shape = 21,show.legend = FALSE)+
  coord_fixed(ratio=1)
```



The above plot provides insight into the capabilities (ie., Attack, Speed, Defense) of each Pokemon types by plotting the components for each Pokemon Types.

Conclusion

This topic was chosen out of personal curiosity. This project attempts to predict the type of Pokemon based on its properties (e.g., HP, Attack stats, etc) and to discover the underlying structure/dimensions that best classifies all the Pokemons.

Using randomForest model, i was able to predict class membership close to 98% of the time. Using the model, it was also revealed that `Speed` is the strongest predictor of `Type 1` membership.

Using just 2 components, I was able to predict close to 68% of the variance.

The work summarised in this report will benefit players in determining the class of their opponent's Pokemons or their own, and also helped them in choosing the best Pokemon Types for battle.