Data Cleaning and Description

2022-10-07

The purpose of this document is to show how the data will be cleaned for the actual paper and provide a longer description that will be condensed for the submitted paper.

Since the class mostly focused on `tidyverse`, I used those functions for all data cleaning and subsetting.

```
library(tidyverse)
```

Source

The dataset used was "The Complete Pokemon Dataset" from Kaggle user Rounak Banik. The data contains information on pokemon from the game *Pokemon Go*. The data technically isn't complete Pokemon dataset as it is limited to the Pokemon that were already introduced in and before 2017.

Cleaning

I decided to clean the data based on the following:

- combat type modifiers will be removed
- the resulting tibble should be in a tidy format

  This means that entries with multiple values are removed or simplified.
- only one unique identifier column is needed
- columns that are linear combinations of others will be removed
- categorical variables should be stored as factors with up to 15 different levels

  This is due to how some packages have trouble working with more than 15 levels.

```
pokemon <- read_csv("pokemon.csv") %>%
  select(
    !contains("against"),
    -c(abilities, japanese_name, pokedex_number, base_total)
  ) %>%
  filter(name != "Minior") %>%
  mutate(
    capture_rate = as.numeric(capture_rate),
    classification = classfication %>% str_extract(r"([:alpha:]+(?= Pok))") %>% fct_lump_n(8),
    type1 = type1 %>% fct_lump_n(14),
    type2 = type2 %>% fct_explicit_na("None") %>% fct_lump_n(13),
    generation = as.factor(generation),
    is_legendary = as.logical(is_legendary),
    .keep = "unused"
  )
```

The cleaned dataset will be stored in the file `cleaned_pokemon.csv`.

```r
pokemon %>% write_csv("cleaned_pokemon.csv")
```

Description

The resulting tibble has the following columns

- `attack`, `defense`, `speed`, `hp`, `sp_attack`, `sp_defense`

  Numeric columns representing attributes used in combat.

- `base_egg_steps`, `base_happiness`

  Numeric columns representing base values for Pokemon attributes.

- `capture_rate`

  8-bit integer column that is used to calculate the probability that a Pokemon is caught.

- `height_m`

  Numeric column representing the height of the Pokemon in meters.

- `name`

  Character column containing the Pokemon's official English name. This is also a unique identifier.

- `percentage_male`

  Numeric column representing the percent of the pokemon of a species that are male. Pokemon species without sex have `NA` values in this column.

- `type1`, `type2`

  Character columns representing the primary and secondary types of the pokemon respectively.

- `weight_kg`

  Numeric column representing the weight of the Pokemon in kg.

- `generation`

  Factor representing the generation in which the Pokemon was introduced. This dataset only contains up to Generation 7 (i.e. Pokemon from *Ultra Sun*, *Ultra Moon*, and previous titles).

- `is_legendary`

  Logical column that is true when the Pokemon is legendary and false otherwise.

- `classification`

  Character column that describes biological characteristics of the Pokemon.

Basic Additive Step Model

In order to find which predictors may be important, I decided to use a basic additive step model with AIC as the metric. In order to avoid issues for this, incomplete rows were removed.

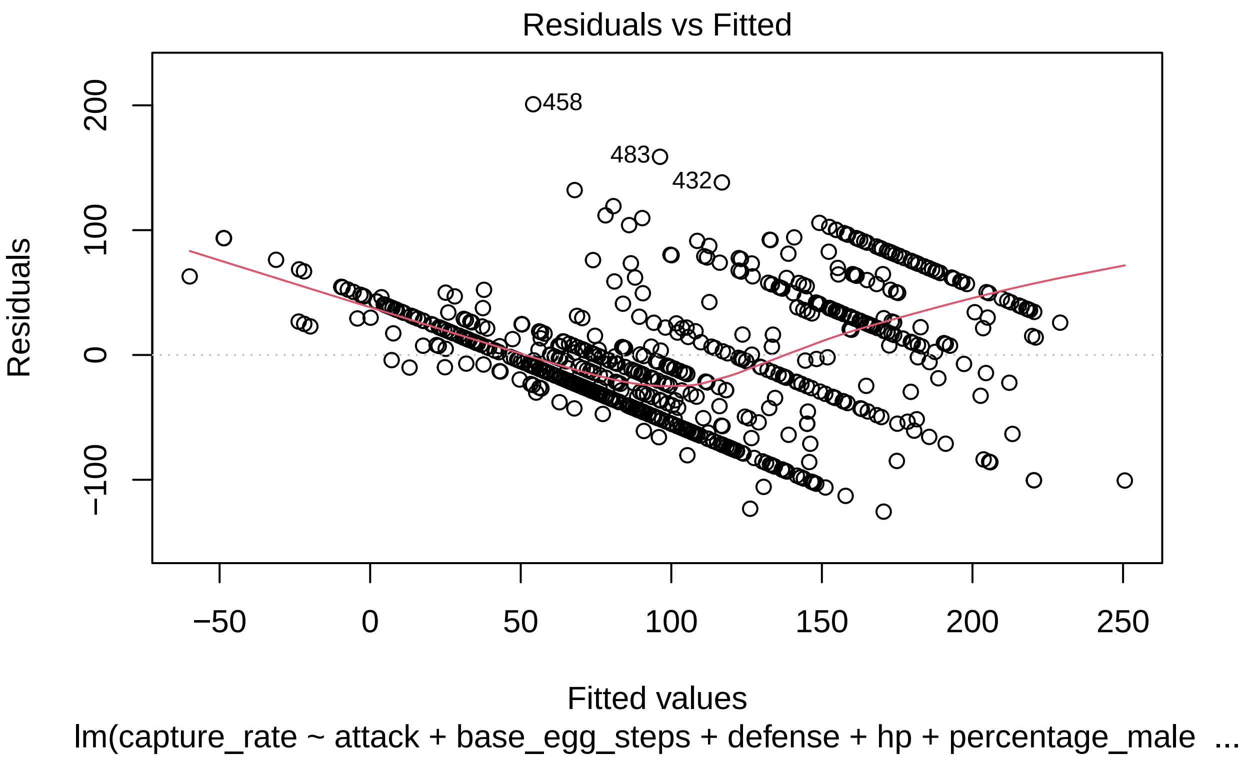
```r
lm1 <- lm(
  capture_rate ~ . - name,
  data = pokemon %>% drop_na()
) %>%
  step(trace = 0)
```

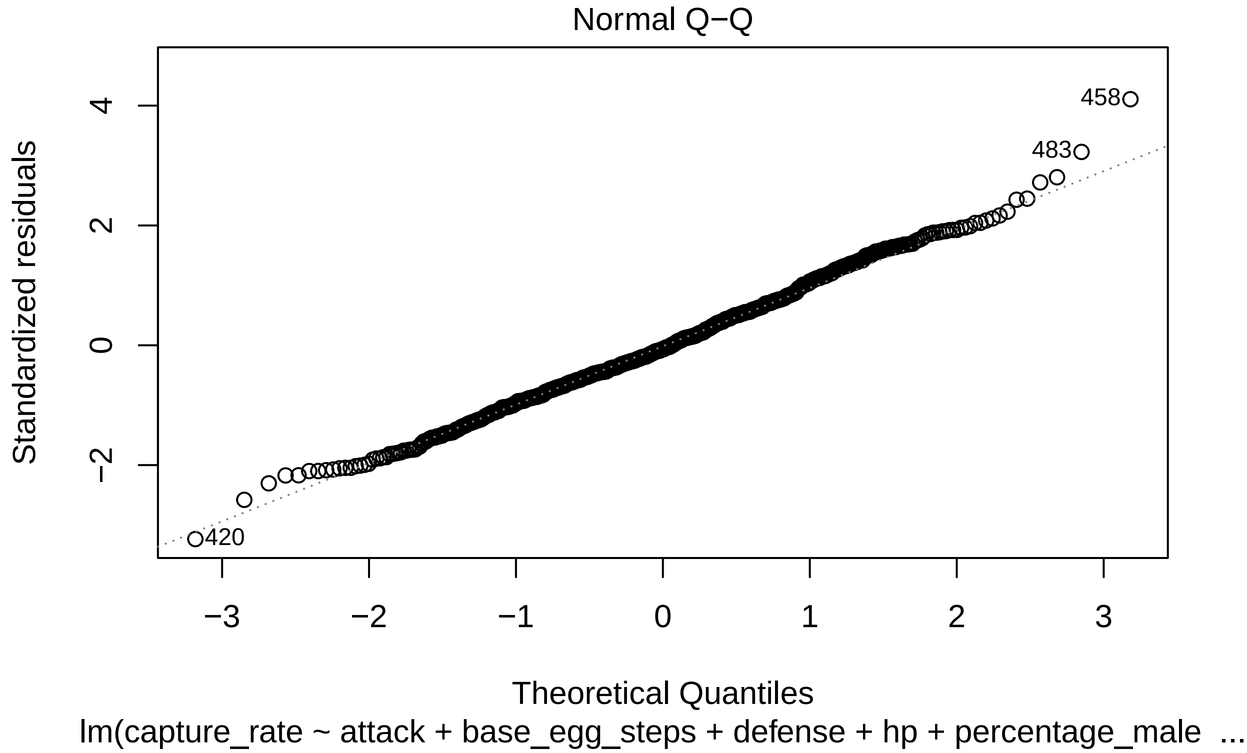
```r
summary(lm1)
```

```
##
## Call:
## lm(formula = capture_rate ~ attack + base_egg_steps + defense +
##     hp + percentage_male + sp_attack + sp_defense + speed + generation +
##     is_legendary, data = pokemon %>% drop_na())
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -125.518  -33.035   -3.194   30.937  200.862
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)      360.676593  10.996741  32.798  < 2e-16 ***
## attack            -0.267014   0.081705  -3.268 0.001138 **
## base_egg_steps    -0.004924   0.001045  -4.710 3.01e-06 ***
## defense           -0.494662   0.084932  -5.824 8.92e-09 ***
## hp                -0.708097   0.088619  -7.990 5.91e-15 ***
## percentage_male   -0.687700   0.096855  -7.100 3.19e-12 ***
## sp_attack         -0.453552   0.080081  -5.664 2.20e-08 ***
## sp_defense        -0.296576   0.098127  -3.022 0.002604 **
## speed             -0.565964   0.080633  -7.019 5.50e-12 ***
## generation2      -16.733081   6.927732  -2.415 0.015986 *
## generation3       12.219837   6.448210   1.895 0.058514 .
## generation4      -11.072779   6.937180  -1.596 0.110928
## generation5        6.290778   6.199610   1.015 0.310614
## generation6        3.538486   7.637734   0.463 0.643307
## generation7      -16.110611   8.044496  -2.003 0.045614 *
## is_legendaryTRUE  98.455237  28.984098   3.397 0.000722 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 49.48 on 668 degrees of freedom
## Multiple R-squared:  0.5608, Adjusted R-squared:  0.551
## F-statistic: 56.87 on 15 and 668 DF,  p-value: < 2.2e-16
```

This does not have a particularly good R^2. It may also help to look at diagnostic plots.

```r
plot(lm1, which = 1:2)
```



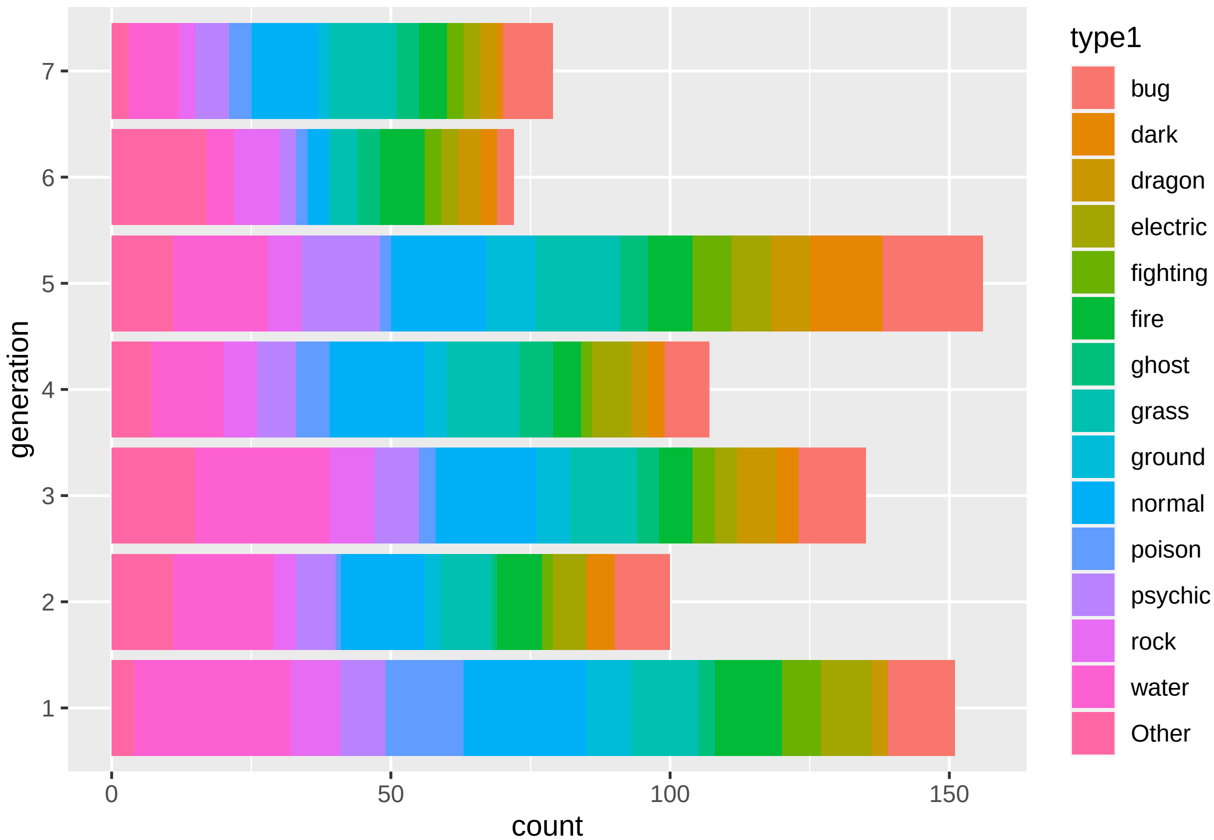


There seems to be some clear lines in the residuals. Fortunately, the residuals do appear to be normally distributed.

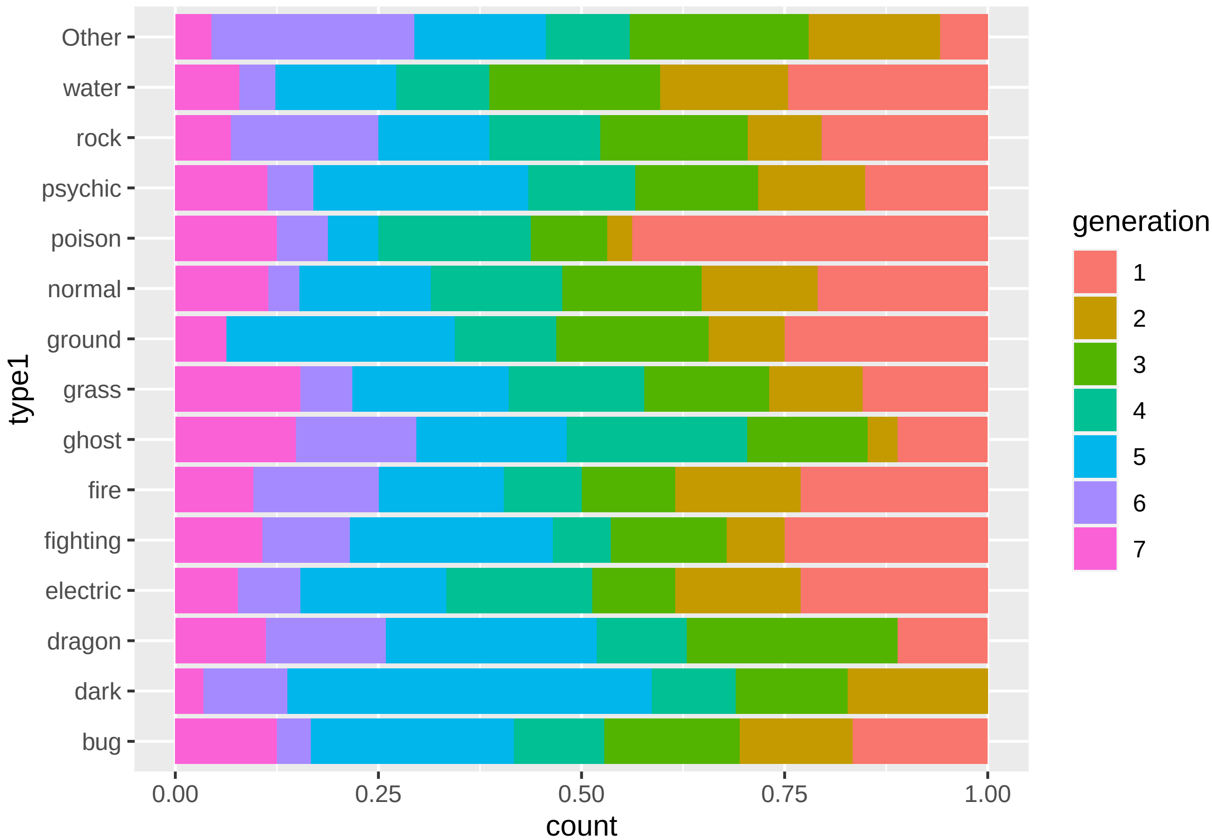
Visualization

The following plot tries to show how many of each type of pokemon were introduced in each generation.

```
ggplot(
  pokemon,
  aes(y = generation, fill = type1)
) +
  geom_bar()
```




```
ggplot(
  pokemon,
  aes(y = type1, fill = generation)
) +
  geom_bar(position = "fill")
```



Anova Models

It may also help to test out some ANOVA models to see the if there are significant differences in capture rate between different categorical variables.

To do this, I decided to subset the dataset to only include categorical variables as well as the response.

```
anova_data <- pokemon %>%
  select(
    capture_rate,
    !where(is.numeric)
  )
```

In order to figure out which categorical variables had an effect on `capture_rate`, we tested a model that used types, generation, legendary status, and classification. Since primary and secondary types are both types, their interaction was considered.

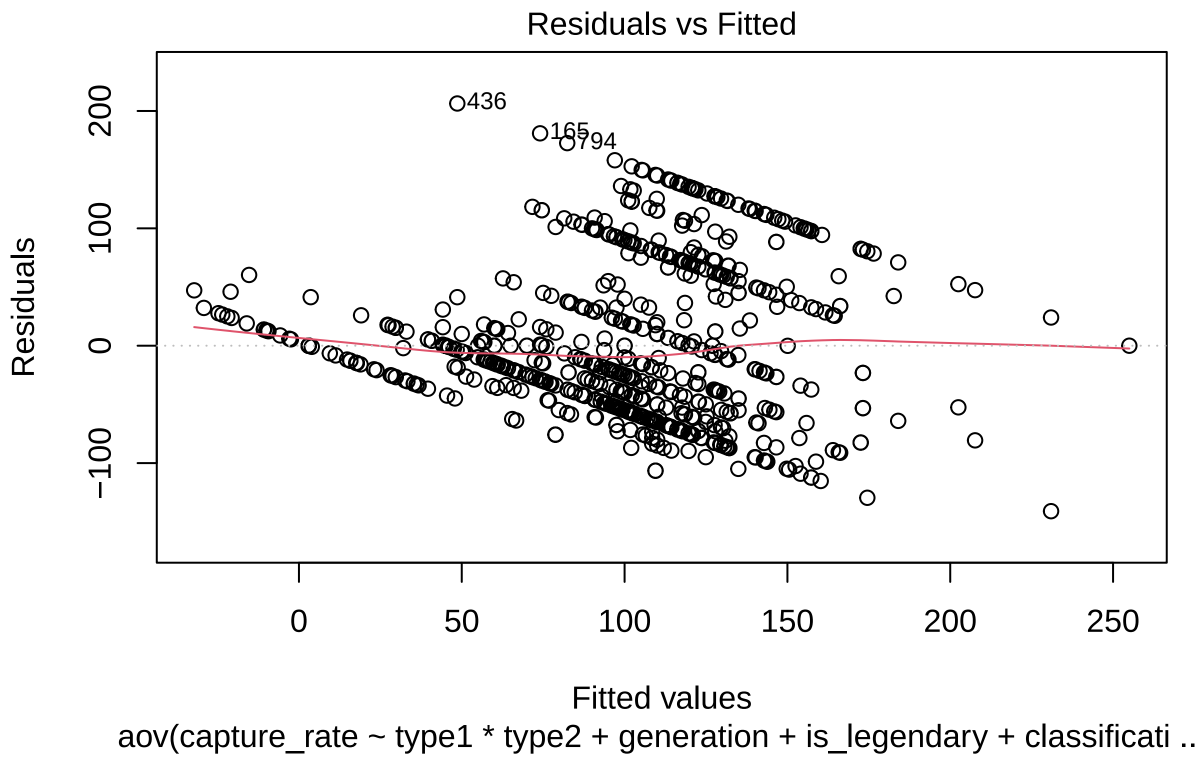
```
aov1 <- aov(
  capture_rate ~ type1 * type2 + generation + is_legendary + classification,
  anova_data
)

summary(aov1)
```

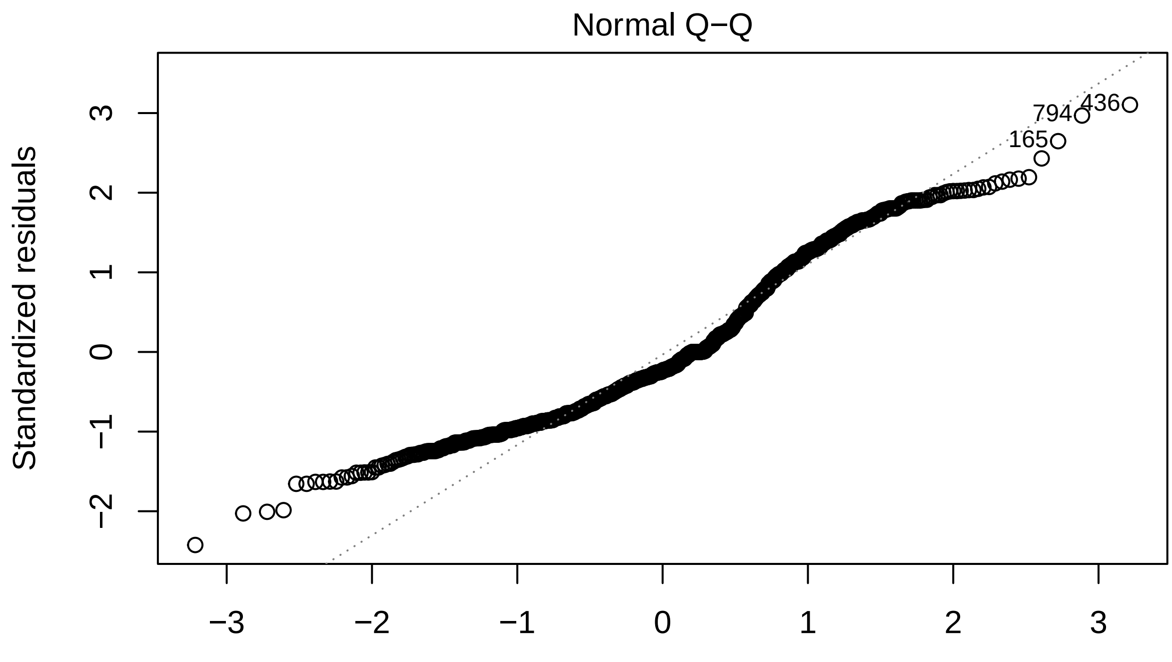
```
##                   Df  Sum Sq Mean Sq F value   Pr(>F)
## type1             14  313583   22399   4.381 1.67e-07 ***
## type2             13  121888    9376   1.834 0.034939 *
## generation         6  122243   20374   3.985 0.000632 ***
```

```
## is_legendary     1  314290  314290  61.468 1.86e-14 ***
## classification   8   63035    7879   1.541 0.139617
## type1:type2    111  408309    3678   0.719 0.983950
## Residuals      646 3303043    5113
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
plot(aov1, which = 1:2)
```

```
## Warning: not plotting observations with leverage one:
##   248, 251, 290, 389, 395, 400, 442, 448, 475, 485, 487, 492, 530, 638, 639, 646, 655, 660, 675, 691
```





aov(capture_rate ~ type1 * type2 + generation + is_legendary + classificati ...

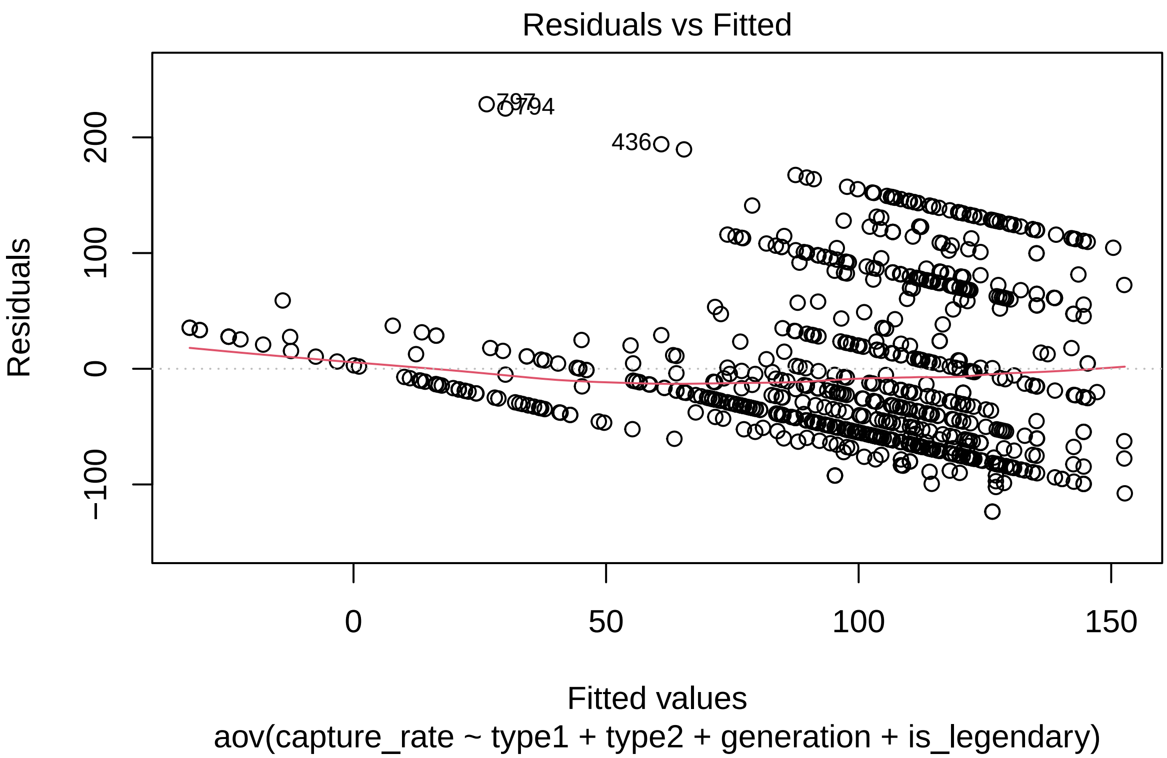
The summary from this model seemed to indicate that the interaction between primary and secondary types was not significant. It also seemed to indicate that classification was also not a significant factor.

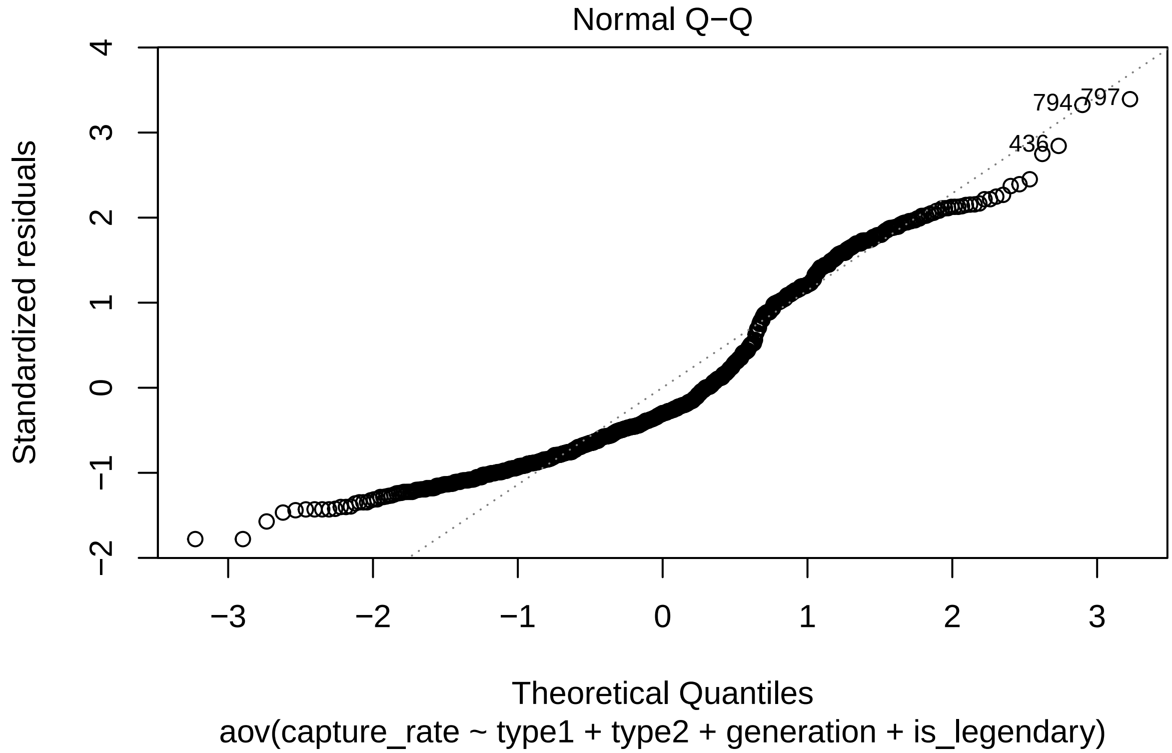
From here, a reduced model was used.

```
aov2 <- aov(
  capture_rate ~ type1 + type2 + generation + is_legendary,
  anova_data
)

summary(aov2)
```

```
##                 Df  Sum Sq Mean Sq F value   Pr(>F)
## type1           14  313583   22399   4.540 6.31e-08 ***
## type2           13  121888    9376   1.900 0.026927 *
## generation       6  122243   20374   4.129 0.000432 ***
## is_legendary     1  314290  314290  63.701 5.30e-15 ***
## Residuals      765 3774387    4934
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
plot(aov2, which = 1:2)
```

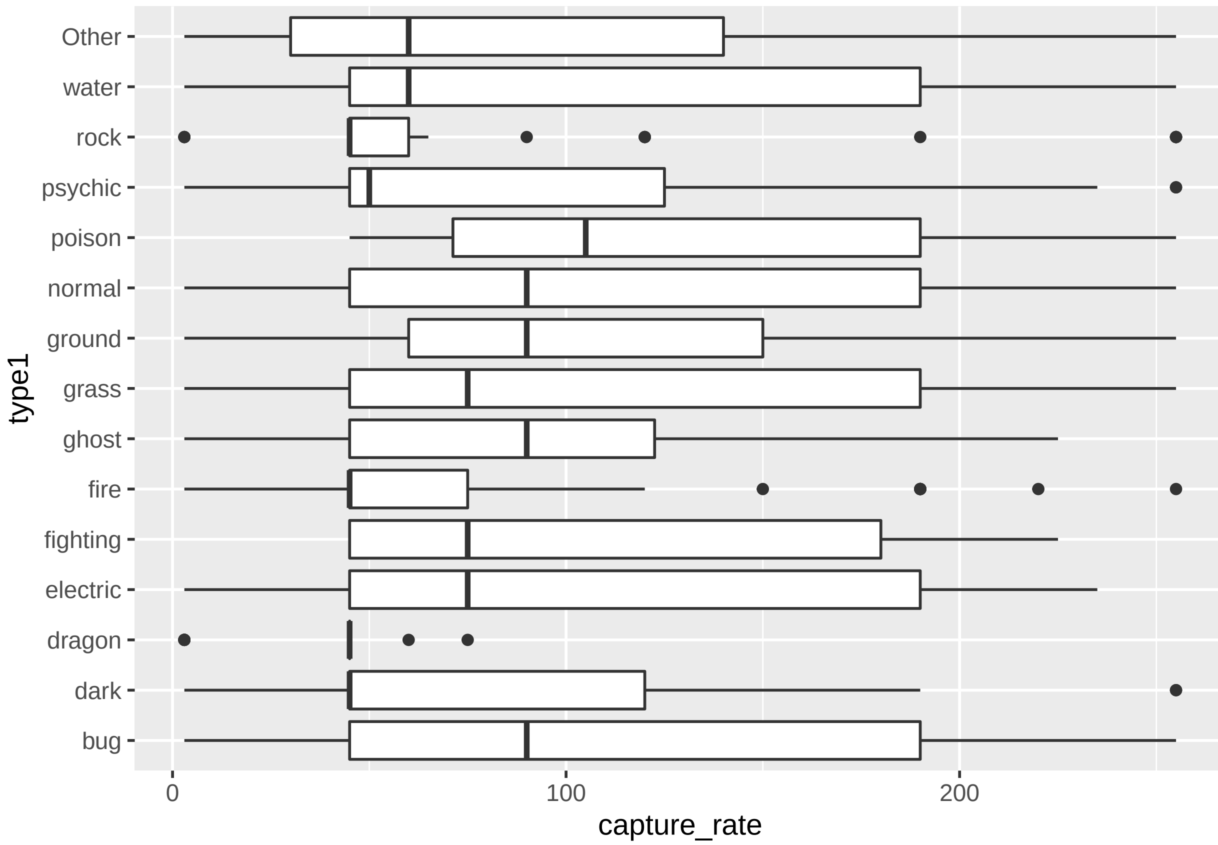




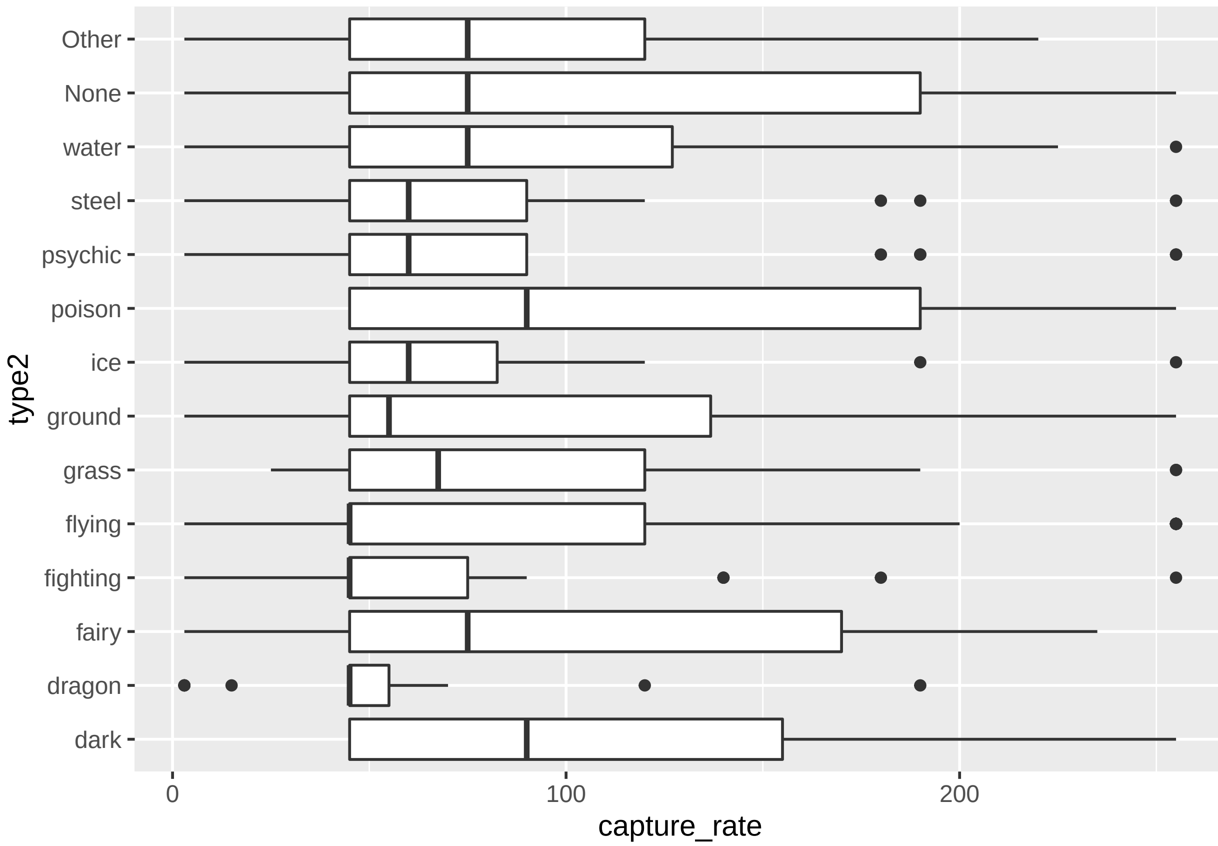
In the reduced model, all factors were significant. In the diagnostic plots from both, the QQPlots seemed close to normal. When box-cox tests were used, small powers in the range of 0.02 to 0.12 were recommended for transformation. Both seemed to violate the constant variance assumption for ANOVA. As such, the results may not be valid.

In order to visualize some differences, it may help to use plots.

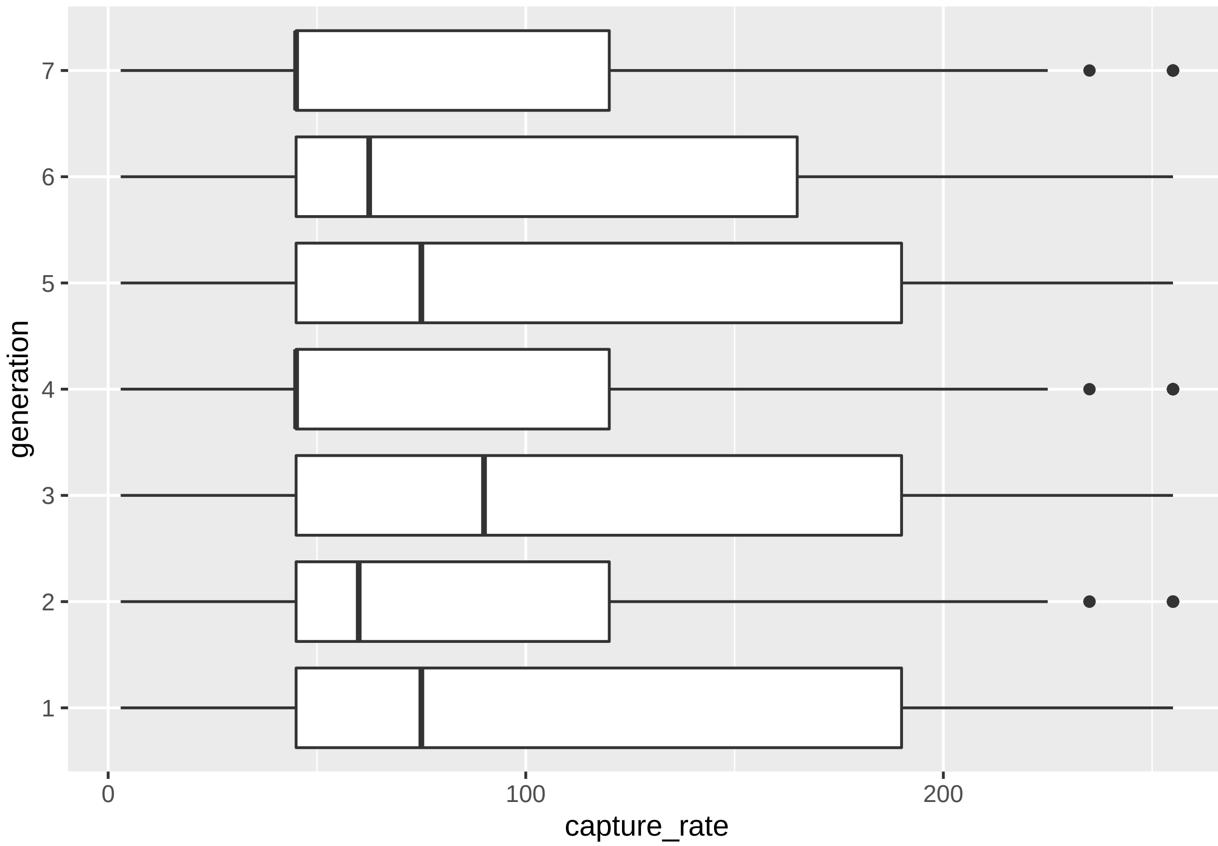
```
ggplot(pokemon, aes(capture_rate, type1)) + geom_boxplot()
```



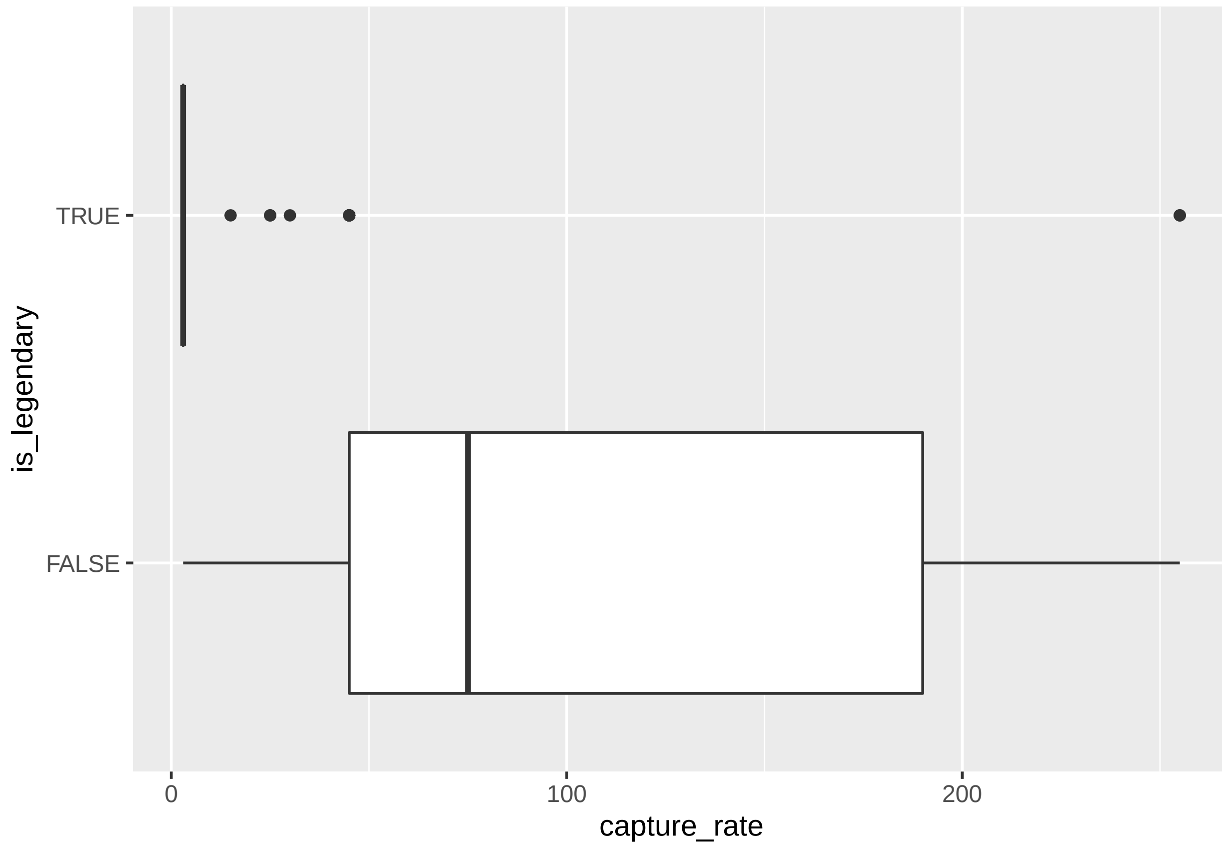

```
ggplot(pokemon, aes(capture_rate, type2)) + geom_boxplot()
```



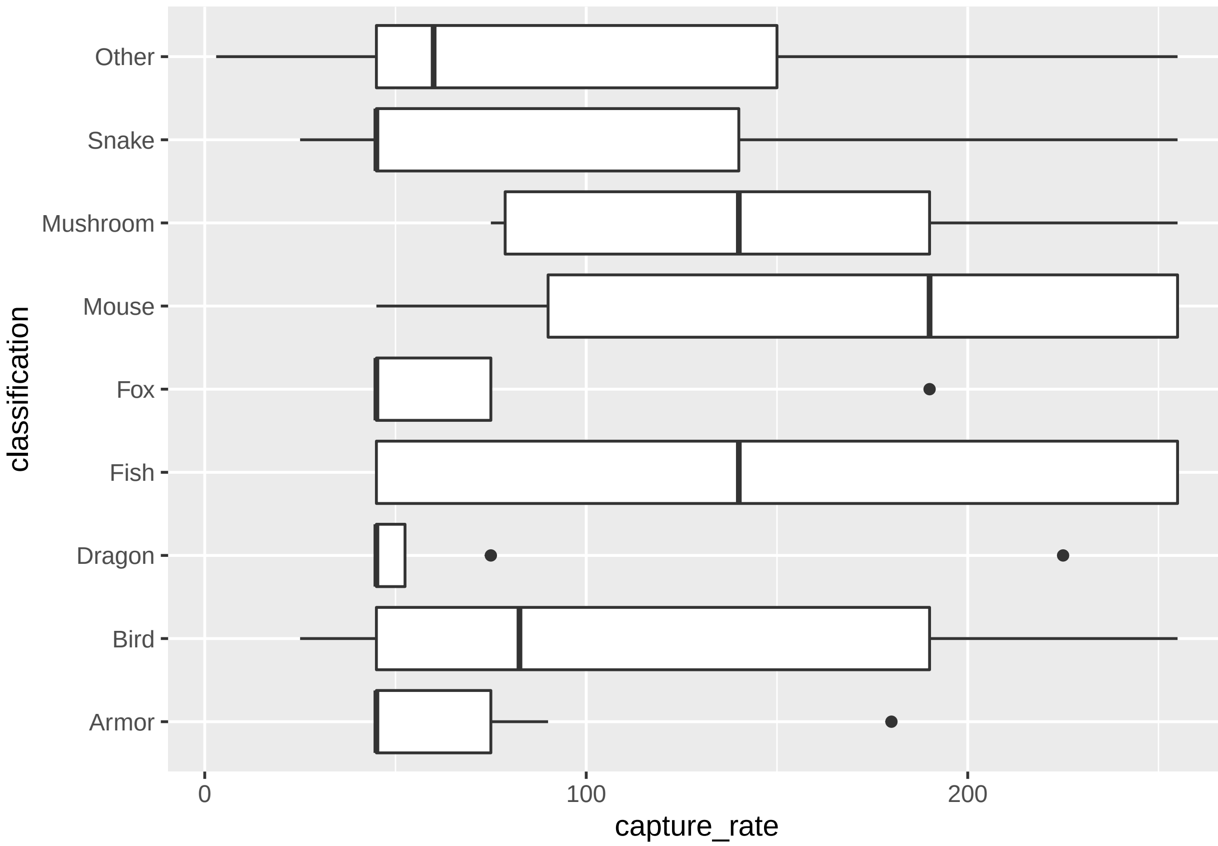

```
ggplot(pokemon, aes(capture_rate, generation)) + geom_boxplot()
```




```
ggplot(pokemon, aes(capture_rate, is_legendary)) + geom_boxplot()
```




```
ggplot(pokemon, aes(capture_rate, classification)) + geom_boxplot()
```



Additional Notes

By default, Hoopa has classification "Mischief Pokémon (Confined)Djinn Pokémonn (Unbound)". When cleaning the classfication column, the regular expression used extracted "Mischief". This can be changed if needed.

The Pokemon Minior has two different values for `capture_rate` depending on its form. Since it has multiple values for the response variable, I decided to omit it.