Math 341 - 12:30PM T/R

Pokemon Competitive Tier Prediction

Nicholas Warfield

16 May, 2019

Abstract

The goal of this study was to predict the competitive viability of pokemon. A secondary goal was to use these predictions to find competitive pokemon that have been overlooked by the community. To do so involved merging two different data sets, as well as some extensive error correction. Proportional odds logistic regression was the modeling technique chosen because the response variable is ordinal. After eliminating some redundant variables, a stepwise algorithm was used to create an initial model. The assumptions of the initial model were confirmed. Then a final model was created that eliminated uninfluential variables. Again, the assumptions of the model were checked, and found to hold true. Ultimately, the study is inconclusive because of severe limitations and lacking data.

Introduction

I like Pokemon and I especially like making competitive teams that use unusual pokemon. With this study, I want to create a model that can predict which tiers a pokemon would be a viable choice for. Of particular interest to me, is if this model will be capable of finding hidden gems, pokemon that are viable but overlooked by the community. Non combat predictors, like the probability of a pokemon being male, may be valid predictors that could lead to discovering hidden gems. But combat related predictors, like the typing of a pokemon, will almost certainly be the largest predictor of viability. In fact, my hypothesis is that the base stats and typing of the pokemon will be the only predictors that matter.

Some important notes I should mention at the beginning is that this study is inherently limited. Two of the most important predictors, move pool and ability, are not included in the data. Additionally, the response variable is heavily influenced by human factors not considered in this study. For example, whatever pokemon is popular will lead to more people choosing it, further increasing the pokemon's ranking. And the last thing to note is that the findings of this study can only be used to analyze Pokemon Sun and Moon, since that is the game the data is based on. These will get expanded on later, but I feel that it is important to note these things at the beginning.

Data Set

Both data sets were extracted from fan websites, serebii and smogon. Serebii has an exhaustive database with dozens of variables, which were extracted from the pokemon games. Smogon is a fan organization that determines which tier a pokemon is placed into, and this is where the response variable of this study comes from. Of particular note is that this response variable is heavily subjected to human interpretation and meddling. I used google sheets instead of R to merge the two sets because it was quite intensive and there were several errors that needed to be corrected by hand. Any pokemon that has multiple forms was only represented once by first data set, but smogon ranks alternate forms separately. For this study I made sure add entries for the mega evolutions of pokemon because it almost always impacts ranking. Other alternate forms do not have a consistent or significant impact on ranking so I did not include them. They did create errors in some of the variables like type2 and weight, so I corrected them.

Variables:

dex - the pokedex number of the pokemon, due to alternate forms it is not unique
gen - the generation the pokemon was introduced in
tier - the competitive tier the pokemon is placed into, with ag being the highest, pu being the lowest, and ou being the most popular
is_legendary - true if the pokemon is a legendary pokemon
is_mega - true if the pokemon is a mega evolution
name -the name of the pokemon
stat_total - the sum of a the pokemon's base stats
hp - the base hp of a pokemon, determines how much damage a pokemon can take
atk - the base attack of a pokemon, determines how much physical damage a pokemon can do
def - the base defense of a pokemon, determines how much physical damage a pokemon can reduce
sp_atk - the base special attack of a pokemon, determines how much special damage a pokemon can do
sp_def - the base special defense of a pokemon, determines how much special damage a pokemon can reduce
spd - the base speed of a pokemon, determines which pokemon attacks first.
type1 - the first type of a pokemon, it is a categorical variable with factors: normal, fighting, flying, poison, ground, rock, bug, ghost, steel, fire, water, grass, electric, psychic, ice, dragon, dark, fairy. Types determine which types of moves a pokemon is resistant and weak to/
type2 - same as type1 but may also be unassigned
weakness - the sum of the damage multipliers of an opponent's move (variables that begin with against)
height_m - the height of a pokemon in meters

weight_kg - the weight of a pokemon in kilograms
p_male - the probability of a pokemon being male when generated, may be
unassigned for pokemon that are genderless
capture_rate - determines how easy a pokemon is to catch, with higher values making a
pokemon easier to catch
experience - the experience growth curve a pokemon, there are 6 curves differentiated by the total experience it takes to reach level 100
egg_steps - the number of steps it takes to hatch an egg of a pokemon, there are 10 possible values for this variable
happiness - the amount of happiness a pokemon has when it is first caught
against_bug - how much damage bug moves do against a pokemon
against_dark - same as against_bug, but for dark type moves
against_dragon - same as against_bug, but for dark type moves
against_fairy - same as against_bug, but for dark type moves
against_fight - same as against_bug, but for dark type moves
against_fire - same as against_bug, but for dark type moves
against_flying - same as against_bug, but for dark type moves
against_ghost - same as against_bug, but for dark type moves
against_grass - same as against_bug, but for dark type moves
against_ground - same as against_bug, but for dark type moves
against_ice - same as against_bug, but for dark type moves
against_normal - same as against_bug, but for dark type moves
against_poison - same as against_bug, but for dark type moves
against_psychic - same as against_bug, but for dark type moves
against_rock - same as against_bug, but for dark type moves
against_steel - same as against_bug, but for dark type moves
against_water - same as against_bug, but for dark type moves

Sources:

serebii data set
smogon rankings data set
data used to make corrections
final data set

Analysis

I extracted all of the numerical variables into a new dataframe to create a correlation matrix. This showed that stat_total and all of the other base stats were highly correlated with each other, and that weakness and all of the "against" variables were weakly to moderately correlated with one another. Additionally, height and weight are moderately correlated, but I did not notice that until after I made my first models. I kept base_stats and weakness to eliminate potential collinearity issues. I also removed names because it is an identifier. Dex and gen were left in because I thought that the game a pokemon was introduced in may have some impact on viability. I organized all the potential predictor variables, along with the response variable, into a new table to make it easier to create models.

On top of making a correlation matrix, I plotted most of the predictors against tier to verify that there was some relation between the two. Since I will be using logistic regression, I am not looking for a linear trend, but a good predictor should have some impact. This did not lead to throwing out any additional variables, but it did show me that the "against" variables were not strong predictors. Something interesting is that the different base stats have different relations with tier, but not nearly as tight of a relationship as stat_total. I decided to keep stat_total for it's strong relation, but a more conclusive study would look at how different base stats affect ranking.

Next up was creating models. Proportional Odds Logistic Regression is the most appropriate regression model since the response variable is ordinal, and it is the kind of model I will be creating. I'll be getting into this later, but because of how limited the data sets are I used a stepwise algorithm to create an initial model, and then refined it by removing variables that do not have a large impact on viability. I created two initial models by doing both forwards and backwards steps, and the resulting models were identical. p_male, happiness, and capture rate were removed by both algorithms, which makes sense because none of those variables are combat related.

Before a final model can be derived from the initial one, I need to verify that the there is a linear relationship between the logit and the predictors. To do so, I plotted all of the predictors in the initial model against the predicted value with some jitter. As expected, stat_total, type1, and type2 were very strong indicators of viability. Dex and gen had no discernible trend with the logits, while weakness and egg_steps had very weak trends. Height, weight, is_mega, and is_legendary all had a moderately linear trend with the logit, but not as strong as stat_total. Experience had some trend, but I believe that is because of a potential third variable. When pokemon evolve they move into higher experience curves, but there base stats also increase.

So, for the final model I selected stat_total, type1, type2, is_legendary, is_mega, and height as my predictors. I threw out experience because it is very likely collinear with stat_total, and I threw out weight because it is moderately correlated with height and height has a stronger trend. Dex, gen, weakness, and egg_steps were thrown out because they had little impact. I then plotted the logits of this new model against its predictors to reaffirm that they had linear trends, which they did. I also compared these plots to the ones of the initial model to verify that the model was about as accurate, which it was.

Results

The final (simplified) model produced is as follows:

$$tier^* = (0.034)(stat\_total) + (t1Coeff)(type1) + (t2Coeff)(type2) + (-0.632)(is\_mega) + (0.0655)(is\_legendary) + (-0.024)(height\_m)$$

where t1Coeff and t2Coeff are the coefficients corresponding to the type of the pokemon. These are dummy programmed in R because they are categorical variables, and I don't want to add 35 more terms in my equation. The output is then compared against the intercepts of each tier to determine which tier will be the final output. Let's take Mega Venusaur for example:

$$(0.034)(625) + (-1.09_{\text{grass 1}}) + (0.72_{\text{poison 2}}) + (-0.632)(1) + (0.0655)(0) + (-0.024)(2.4)=20.22$$
$$18.9\ _{\text{bl | ou}} < 20.22 < 21.2\ _{\text{ou | uber}}$$
$$\rightarrow \text{OU}$$

Mega Venusaur is a mega, grass, poison pokemon with a stat total of 625, and height of 2.4 meters. Plugging those values into the models outputs 20.22 which doesn't mean anything. But the bounds for the OU tier are 18.9 and 21.2. Since 20.22 fits between those bounds, Mega Venusaur is predicted to be of OU viability. The full list of the coefficients and bounds are as follows:

Coefficients:	
stat_total	0.03397
is_legendary	0.06555
is_mega	-0.63223
type1dark	0.79947
type1dragon	-0.54994
type1electric	-0.44284
type1fairy	1.42230
type1fighting	1.30834
type1fire	-1.16963
type1flying	0.67628
type1ghost	-0.07508
type1grass	-1.08768
type1ground	0.03499
type1ice	-1.90513
type1normal	-0.52184
type1poison	0.08046
type1psychic	-0.45128
type1rock	-2.22259
type1steel	-0.40424
type1water	-0.52169

Coefficients:	
height_m	-0.02391
type2dark	0.01261
type2dragon	-1.06748
type2electric	-0.70840
type2fairy	0.50141
type2fighting	0.20750
type2fire	-0.80088
type2flying	-0.23664
type2ghost	1.80079
type2grass	-1.35246
type2ground	-0.76531
type2ice	-0.62535
type2normal	0.19398
type2poison	0.71861
type2psychic	-0.49355
type2rock	-1.44476
type2steel	1.29715
type2water	0.05175
type2NA	-1.23434

Bounds:	
unranked|pu	12.3673
pu|bl4	14.9401
bl4|nu	15.0407
nu|bl3	15.9701
bl3|ru	16.0292
ru|bl2	16.9324
bl2|uu	17.2540
uu|bl	18.5097
bl|ou	18.9274
ou|uber	21.2445
uber|ag	26.9458

Limitations:

This study is extremely limited and I want to use Blaziken as a case study to demonstrate this. Blaziken is an Uber tiered pokemon but my model only predicts blaziken to be of BL2 viability. That's because blaziken's stat_total is only 530 and his typing is poor. So why is there such a large difference between Blaziken's actual tier and predicted tier? Because special abilities and individual stats are not considered in my mode. Blaziken has access to the Speed Boost special ability, which is a decent ability. But in context of having a very high base attack and decent base speed, means that Blaziken can do a lot of damage without any set up. Most other pokemon would need to spend a turn or two pumping up before they could do what Blaziken can on turn one. Being able to do that was so powerful that Blaziken was the first non legendary pokemon to get in the Uber tier. That is how much abilities matter in competitive pokemon, and how much these other limitations matter too.

Another major limitation would be the exclusion of a pokemon's move pool. A pokemon with great stats and typing can be crippled by not having access to strong moves. There are some other mechanics like held items and natures that are not included. But since all pokemon have access to all items and natures, this is not impactful in determining rank.

The third limitation is the response variable. There is no objective way to measure how good a pokemon is, that's why the tiers are more or less based around the popularity of pokemon in competitive play. And there is also the meta game to consider. If everyone playing knows that Blaziken is super strong, they will start to pack their teams with pokemon that are good against Blaziken. The stronger people perceive a pokemon to be, the more it influences the decisions people make when constructing their teams. Suddenly, Tentacruel gets popular not because he is inherently strong, but because he is strong against Blaziken. A better model would probably try to find some alternate response variable to measure how good a pokemon is.

The final limitation is the nature of the subject. Pokemon is a game made by humans, and this leads to some weird things. Like having access to 100% accurate data about the entire population. But then a new game gets pushed out and suddenly all of the old data is no longer reliable. It's really weird because it doesn't behave like real world data. On the one hand, we can make really accurate models if we properly account for all of the relevant game mechanics. On the other hand, that model can't be used to accurately predict viability of pokemon not included in the model because new mechanics and changes have been introduced.

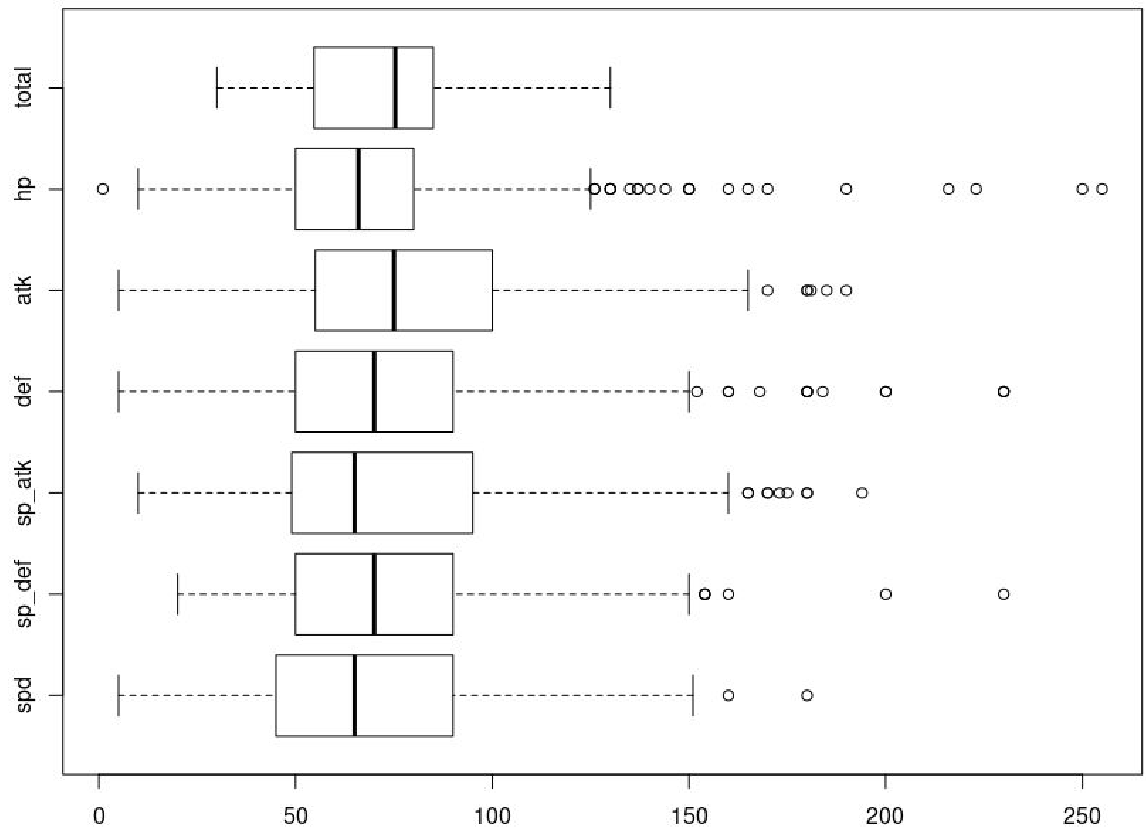
Conclusion

So conclusion time, the model I ended up with is not good. This is largely because it does not account for the special abilities and move pool a pokemon has. If it did, I would consider it to be pretty mediocre because move pool, ability, stats, and typing have a synergistic effect. Having access to a good special move is even better for pokemon with high special attack. Any model that does not account for these and other synergies would be incomplete.

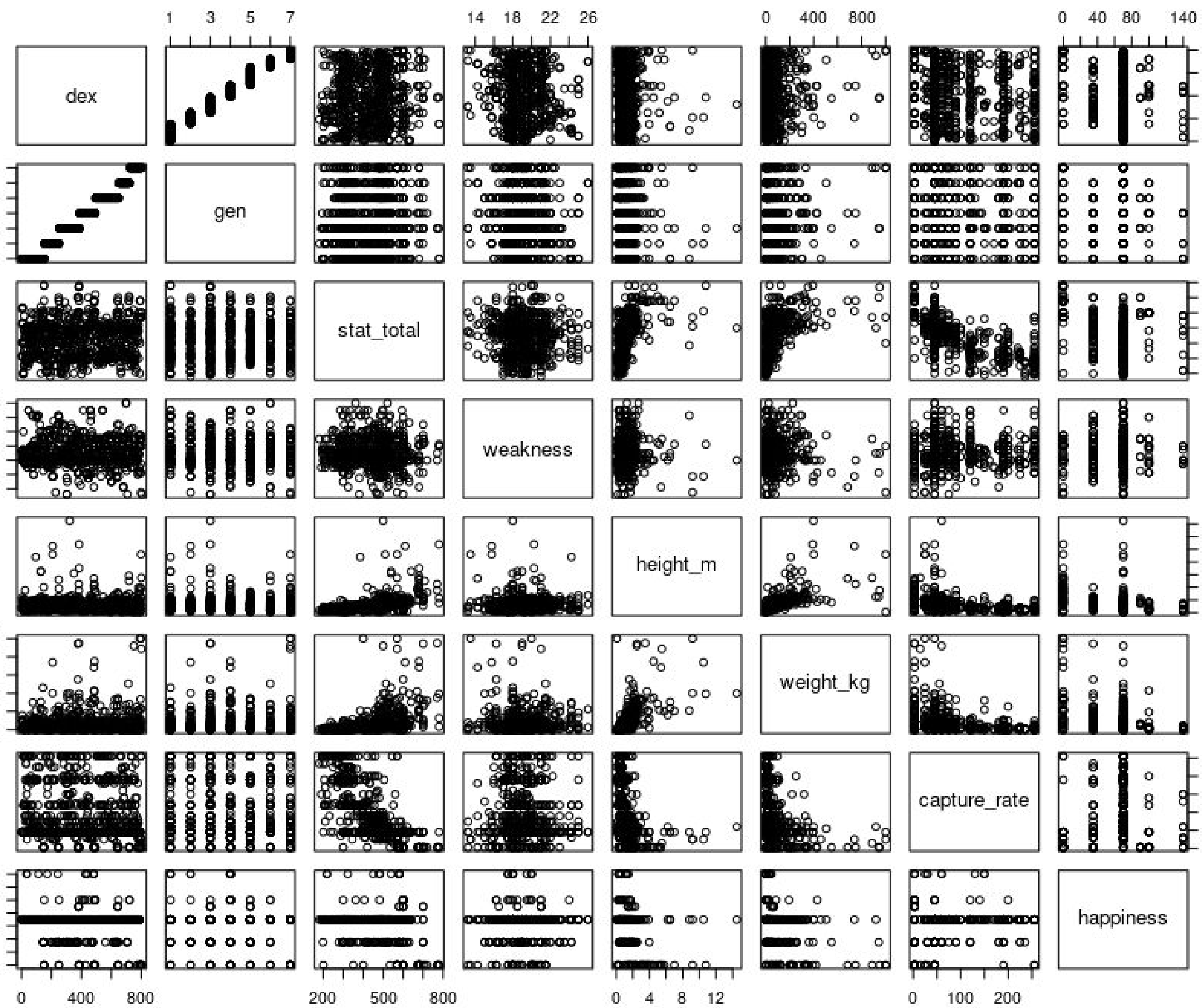
Additionally, I wanted to use this model to find hidden gems, but my model is mostly influenced by total_stats and typing. Those are the things that most people consider when constructing their teams. So the model I made is more of a shortcut for finding pokemon that people would already consider to be good. And if it does point to another pokemon as being good, it's probably inaccurate because that pokemon may have weak moves or poor abilities. My model does not support my hypothesis either. Even though stat_total and typing are the main drivers of viability, other parameters, like height_m, can be used to fine tune the prediction.

What I would want to see in a continuation would be a few things. Number one is factoring in the effects of move pool and abilities. Number two is accounting for synergies between these predictors, which may involve using some sort of non linear modeling. Number three is an alternate response, relying on fan made tiers to generate models will make the model less likely to find hidden gems. The final thing would be including some more parameters. For example, I think that whether or not a pokemon has an evolution would be a significant predictor of viability. Competitive pokemon tend to get categorized as well, blaziken is called a sweeper because he is fast and does a lot of damage. These additional parameters would make the model better account for the human elements at play.

Graphs

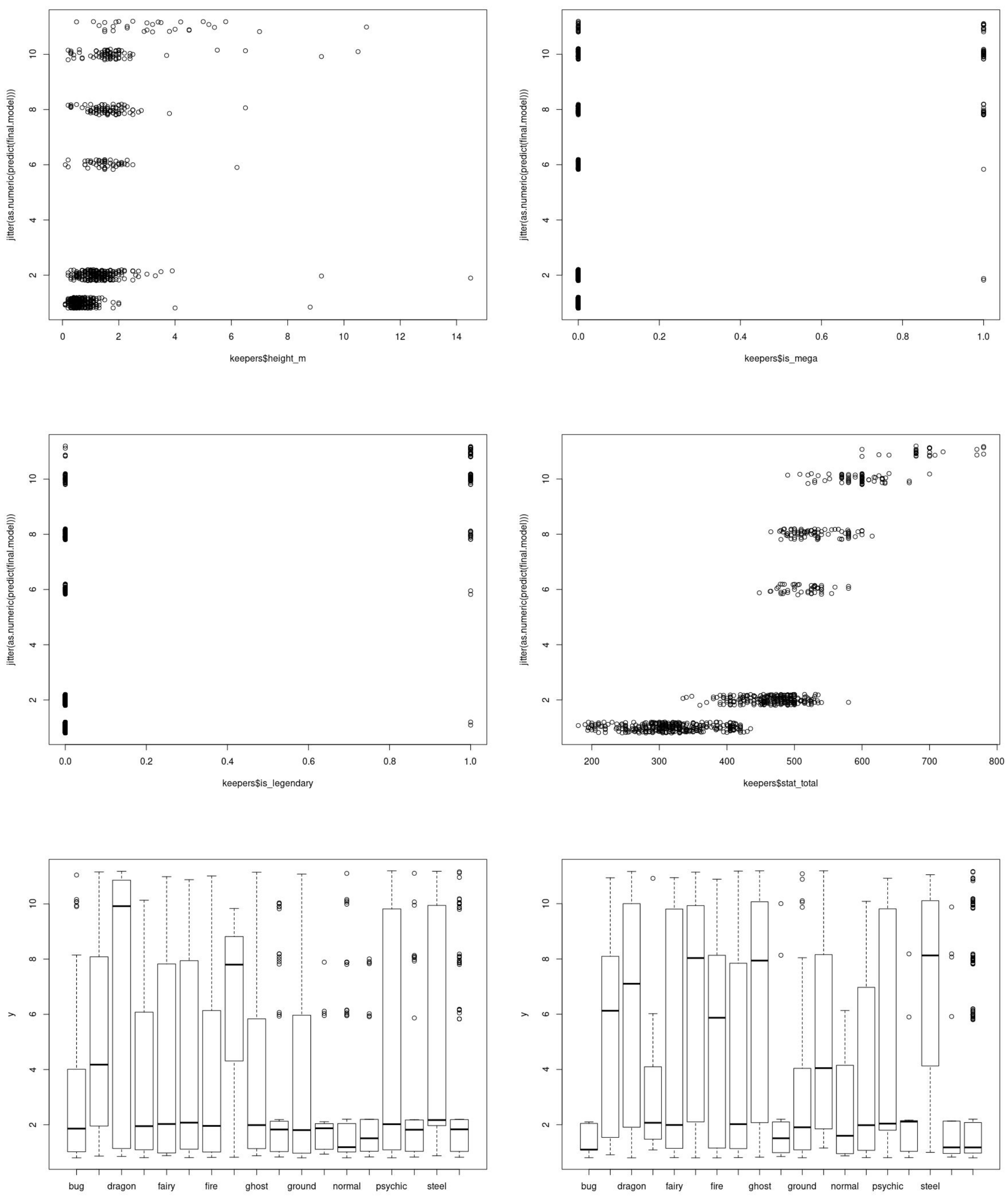


Distribution of base stats and total stats divided by 6



plots of initial numerical data

final model logit plots



Models

Initial Modely Summary:

Call:
polr(formula = tier ~ stat_total + type1 + type2 + weight_kg +
gen + is_mega + weakness + dex + experience + egg_steps +
height_m + is_legendary, data = keepers, method = "logistic")

Coefficients:

	Value	Std. Error	t value
stat_total	0.036002	0.001806	19.9380
type1dark	0.302182	0.371114	0.8143
type1dragon	-1.051151	0.451548	-2.3279
type1electric	-1.336866	0.343225	-3.8950
type1fairy	0.203858	0.515070	0.3958
type1fighting	1.383370	0.379997	3.6405
type1fire	-1.893569	0.311939	-6.0703
type1flying	-0.042491	0.077126	-0.5509
type1ghost	-1.070882	0.441133	-2.4276
type1grass	-0.815692	0.297203	-2.7446
type1ground	-0.114102	0.421500	-0.2707
type1ice	-1.606569	0.469848	-3.4193
type1normal	-1.575844	0.266262	-5.9184
type1poison	-0.912934	0.419194	-2.1778
type1psychic	-0.828223	0.315527	-2.6249
type1rock	-1.937912	0.382750	-5.0631
type1steel	-1.560357	0.408553	-3.8192
type1water	-1.373415	0.228973	-5.9982
type2dark	0.096080	0.420313	0.2286
type2dragon	-0.987252	0.444646	-2.2203
type2electric	-1.136560	0.705412	-1.6112
type2fairy	-0.078226	0.407251	-0.1921
type2fighting	0.844792	0.370812	2.2782
type2fire	-1.080602	0.567304	-1.9048
type2flying	-0.396492	0.243851	-1.6260
type2ghost	1.413075	0.572248	2.4693
type2grass	-0.795971	0.511125	-1.5573
type2ground	-0.607366	0.384019	-1.5816
type2ice	0.105012	0.595697	0.1763
type2normal	0.193904	0.049567	3.9120
type2poison	-0.126532	0.393473	-0.3216
type2psychic	-0.420431	0.385479	-1.0907
type2rock	-0.813826	0.509231	-1.5981
type2steel	0.329632	0.424658	0.7762
type2water	-0.553267	0.585891	-0.9443
type2NA	-1.439092	0.181857	-7.9133
weight_kg	-0.002989	0.001005	-2.9733
gen	-1.271770	0.338392	-3.7583
is_megaTRUE	-0.693159	0.362104	-1.9143
weakness	-0.287256	0.034026	-8.4422
dex	0.009361	0.002852	3.2829
experience.L	-0.556252	0.528352	-1.0528
experience.Q	-0.900326	0.487923	-1.8452
experience.C	0.615779	0.392534	1.5687
experience^4	-0.662683	0.307607	-2.1543
experience^5	0.587246	0.198246	2.9622
egg_steps.L	-1.026874	0.387621	-2.6492
egg_steps.Q	-1.568044	0.266953	-5.8739
egg_steps.C	0.848345	0.361707	2.3454
egg_steps^4	-0.345678	0.406835	-0.8497
egg_steps^5	1.668486	0.406672	4.1028
egg_steps^6	0.933613	0.413063	2.2602
egg_steps^7	0.406958	0.406067	1.0022
egg_steps^8	0.367995	0.369925	0.9948
egg_steps^9	0.934947	0.394198	2.3718
height_m	0.196680	0.096264	2.0431
is_legendaryTRUE	1.518748	0.263725	5.7588

Intercepts:

	Value	Std. Error	t value
unranked|pu	6.9174	0.2298	30.1016
pu|bl4	9.6218	0.3204	30.0345
bl4|nu	9.7243	0.3220	30.2025
nu|bl3	10.6877	0.3371	31.7056
bl3|ru	10.7505	0.3380	31.8102
ru|bl2	11.7071	0.3537	33.0990
bl2|uu	12.0423	0.3590	33.5394
uu|bl	13.3530	0.3856	34.6263
bl|ou	13.7972	0.3962	34.8256
ou|uber	16.3175	0.4963	32.8788
uber|ag	22.8773	1.3627	16.7878

Residual Deviance: 1791.464
AIC: 1927.464

Final Model Summary:

Call:
MASS::polr(formula = tier ~ ., data = final, method = "logistic")

Coefficients:

	Value	Std. Error	t value
is_legendaryTRUE	0.06555	0.329596	0.19888
is_megaTRUE	-0.63223	0.342909	-1.84373
stat_total	0.03397	0.001753	19.37516
type1dark	0.79947	0.384621	2.07860
type1dragon	-0.54994	0.420410	-1.30809
type1electric	-0.44284	0.368119	-1.20299
type1fairy	1.42230	0.513201	2.77144
type1fighting	1.30834	0.399408	3.27570
type1fire	-1.16963	0.341797	-3.42200
type1flying	0.67628	0.723925	0.93418
type1ghost	-0.07508	0.423559	-0.17725
type1grass	-1.08768	0.320894	-3.38953
type1ground	0.03499	0.446466	0.07837
type1ice	-1.90513	0.531185	-3.58656
type1normal	-0.52184	0.295725	-1.76460
type1poison	0.08046	0.415771	0.19352
type1psychic	-0.45128	0.344951	-1.30824
type1rock	-2.22259	0.379030	-5.86389
type1steel	-0.40424	0.436135	-0.92687
type1water	-0.52169	0.269261	-1.93750
type2dark	0.01261	0.421037	0.02995
type2dragon	-1.06748	0.456817	-2.33677
type2electric	-0.70840	0.708172	-1.00033
type2fairy	0.50141	0.381055	1.31584
type2fighting	0.20750	0.352053	0.58940
type2fire	-0.80088	0.537245	-1.49072
type2flying	-0.23664	0.236069	-1.00241
type2ghost	1.80079	0.572821	3.14372
type2grass	-1.35246	0.517823	-2.61183
type2ground	-0.76531	0.377311	-2.02832
type2ice	-0.62535	0.589405	-1.06099
type2normal	0.19398	0.995644	0.19483
type2poison	0.71861	0.365401	1.96662
type2psychic	-0.49355	0.382918	-1.28891
type2rock	-1.44476	0.514717	-2.80691
type2steel	1.29715	0.398661	3.25376
type2water	0.05175	0.557888	0.09276
type2NA	-1.23434	0.180336	-6.84469
height_m	-0.02391	0.073716	-0.32436

Intercepts:

	Value	Std. Error	t value
unranked|pu	12.3673	0.6701	18.4550
pu|bl4	14.9401	0.7592	19.6794
bl4|nu	15.0407	0.7616	19.7498
nu|bl3	15.9701	0.7827	20.4027
bl3|ru	16.0292	0.7840	20.4467
ru|bl2	16.9324	0.8030	21.0858
bl2|uu	17.2540	0.8099	21.3031
uu|bl	18.5097	0.8362	22.1354
bl|ou	18.9274	0.8437	22.4339
ou|uber	21.2445	0.9135	23.2570
uber|ag	26.9458	1.5020	17.9403

Residual Deviance: 1857.984
AIC: 1957.984

R Code

```
library(dplyr)
library(MASS)

pkt <- read_csv("pk_tier.csv", col_types = cols(X42 = col_skip(),
        atk = col_integer(), def = col_integer(),
        dex = col_integer(), gen = col_integer(),
        hp = col_integer(), is_legendary = col_logical(),
        is_mega = col_logical(), sp_atk = col_integer(),
        sp_def = col_integer(), stat_total = col_integer()))
View(pkt)

pkt$tier <- factor(pkt$tier, c("unranked", "pu", "bl4", "nu", "bl3", "ru", "bl2", "uu", "bl", "ou", "uber",
"ag"), ordered = TRUE)

pkt$type1 <- factor(pkt$type1)
pkt$type2 <- factor(pkt$type2)
pkt$type2 <- addNA(pkt$type2)
pkt$p_male <- factor(pkt$p_male, ordered = TRUE)
pkt$p_male <- addNA(pkt$p_male)
pkt$experience <- factor(pkt$experience, ordered = TRUE)
pkt$egg_steps <- factor(pkt$egg_steps, ordered = TRUE)

nums <- select_if(pkt, is.numeric)
View(nums)
c_nums <- cor(nums)
View(c_nums)

select(pkt, spd, sp_def, sp_atk, def, atk, hp, stat_total) -> p_stats
p_stats <- transform(p_stats, total = stat_total / 6)
p_stats$stat_total <- NULL
boxplot(p_stats, horizontal = TRUE)
plot(p_stats)
p_stats$total <- pkt$stat_total
plot(p_stats)
cor.test(pkt$atk, pkt$def)

keepers <- select(pkt, tier, dex, gen, is_legendary, is_mega, stat_total, type1, type2, weakness,
height_m, weight_kg, p_male, capture_rate, experience, egg_steps, happiness)
```

```r
min.model <- polr(tier~1, keepers, method = "logistic")
max.model <- polr(tier~., keepers, method = "logistic")
fwd.model <- stepAIC(min.model, formula(max.model), direction = "forward")
bwd.model <- stepAIC(max.model, formula(min.model), direction = "backward")
summary(fwd.model)
summary(bwd.model)

plot(keepers$weakness, jitter(as.numeric(predict(fwd.model))))
plot(keepers$stat_total, jitter(as.numeric(predict(fwd.model))))
plot(keepers$dex, jitter(as.numeric(predict(fwd.model))))
plot(keepers$gen, jitter(as.numeric(predict(fwd.model))))
plot(keepers$is_legendary, jitter(as.numeric(predict(fwd.model))))
plot(keepers$is_mega, jitter(as.numeric(predict(fwd.model))))
plot(keepers$type1, jitter(as.numeric(predict(fwd.model))))
plot(keepers$type2, jitter(as.numeric(predict(fwd.model))))
plot(keepers$weakness, jitter(as.numeric(predict(fwd.model))))
plot(keepers$height_m, jitter(as.numeric(predict(fwd.model))))
plot(keepers$weight_kg, jitter(as.numeric(predict(fwd.model))))
plot(keepers$experience, jitter(as.numeric(predict(fwd.model))))
plot(keepers$egg_steps, jitter(as.numeric(predict(fwd.model))))

cor.test(keepers$weight_kg, keepers$height_m)

final <- dplyr::select(keepers, tier, is_legendary, is_mega, stat_total, type1, type2, height_m)
final.model <- polr(tier~., final, method = "logistic")
summary(final.model)

plot(keepers$stat_total, jitter(as.numeric(predict(final.model))))
plot(keepers$is_legendary, jitter(as.numeric(predict(final.model))))
plot(keepers$is_mega, jitter(as.numeric(predict(final.model))))
plot(keepers$type1, jitter(as.numeric(predict(final.model))))
plot(keepers$type2, jitter(as.numeric(predict(final.model))))
plot(keepers$height_m, jitter(as.numeric(predict(final.model))))
```