Part 2: Simple Linear Regression

Alex Gui and Lathan Liou

2/9/2018

Introduction

Pokemon Go became an overnight sensation with hundreds of millions of people having downloaded the mobile game. The whole point of the game is to try to catch all the Pokemon available and train them (increase their combat power, or cp) so that you can battle other players with your strengthened Pokemon. A quick note about Pokemon is that they can evolve into stronger forms, so an evolved Pokemon will generally always have a higher cp than a non-evolved Pokemon. A number of people have tried to generate models in an attempt to predict the best way to maximize cp for their Pokemon. In other words, people have tried to generate models that most closely match how the game's algorithms and mechanics calculate cp in reality. This is what we will attempt to do ourselves: create a model to predict cp. In this segment of our project, we'll fit a simple regression model, regressing *cp_new*, the combat power of the evolved Pokemon on *hp*, which is hit points, or the amount of health a Pokemon has. For some context, if *hp* goes to 0, your Pokemon will faint, and you cannot battle with it anymore. Ideally, you'd like a Pokemon with a large amount of *hp*, so that it will take more hits to deplete it to 0. We decided to pick *hp* as our predictor variable in our SLR because we think *hp* could be an important predictor for *cp_new* given that having both high *hp* and high *cp* seem to grant Pokemon a distinct advantage over opponent Pokemon!

To refresh your memory, the dataset we are looking at is an original data set collected by OpenIntro, most likely by some individual who was playing the game, Pokemon Go and who decided to record data. The dataset contains 75 observations across 26 variables, with each observation representing a randomly generated Pokemon that the gamer caught. Only 4 species are represented in this data, which is not representative of the entire Pokemon Go world, in which there are over 200 species and counting (depending on game developers' updates).

You can follow our work here: https://github.com/alexaaag/math158-project.

Hypotheses

$$H_0 : \beta_1 = 0$$
$$H_a : \beta_1 > 0$$

Our null hypothesis is that there is no linear relationship between *cp_new* and *hp*. The alternative hypothesis is there is a positive linear relationship between *cp_new* and *hp*. We pick a one-tail hypothesis because there is no negatively correlated variables observed in the exploratory analysis.

Checking Assumptions

```r
pokemon_lm = lm(cp_new ~ hp, data = pokemon)
std = augment(pokemon_lm)$.std.resid
fitted = augment(pokemon_lm)$.fitted
p1 <- ggplot(data = pokemon, aes(x = hp, y = cp_new)) + geom_point() +
    ggtitle("Relationship Between cp_new and hp") + theme(plot.title = element_text(size = 9))
p2 <- ggplot(data = pokemon_lm, aes(x = fitted, y = std)) + geom_point() +
    xlab("fitted") + ylab("standard residual") + geom_hline(yintercept = 0) +
    ggtitle("Standard Residual Plot") + theme(plot.title = element_text(size = 9))
grid.arrange(p1, p2, nrow = 1, ncol = 2, bottom = "Figure 1: Plot of cp_new and hp")
```

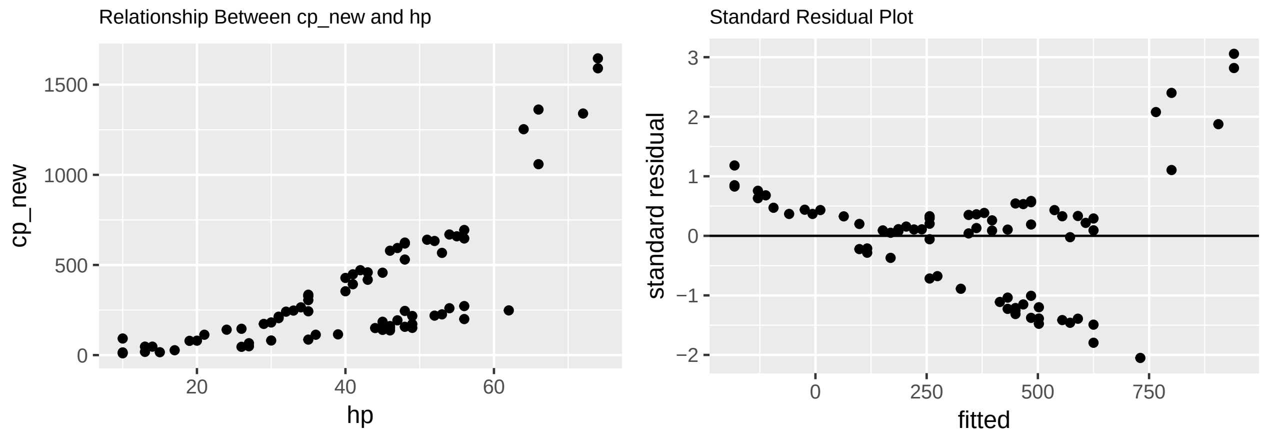

Figure 1: Plot of cp_new and hp

The assumptions we checked for linear regression were linearity, independence, normal errors, and equal variance. We plotted *cp_new* vs. *hp* and generated a residual plot. Currently, the plot of *cp_new* vs. *hp* does not look linear and in the residual plot, the errors are not symmetric nor are they constant. So far, none of the assumptions for linear regression are being met, aside from the independent errors condition. First, we transformed the y variable, *cp_new* using a log transformation. The plot looks a lot more linear since a log transformation squishes large values. The residual plot, while better, still is not the best because there is a distinctive pattern as opposed to the random scattered points that we would like to see. Next, we tried log-transforming the x variable, *hp*, which didn't improve linearity, normality or equal variance. Lastly, we log-transformed both the y and the x variables and that improved linearity as well as the residual plot. The residual plot as shown in Figure 2b, demonstrates normality and constancy of errors.

Log transformation on both *cp_new* and *hp*:

```
pokemon_lm_trans = lm(log(cp_new) ~ log(hp), data = pokemon)
std = augment(pokemon_lm_trans)$.std.resid
fitted = augment(pokemon_lm_trans)$.fitted
p1 <- ggplot(pokemon, aes(x = log(hp), y = log(cp_new))) + geom_point() +
    ggtitle("Relationship between log(cp_new) and log(hp) ") +
    theme(plot.title = element_text(size = 9))
p2 <- ggplot(pokemon_lm_trans, aes(x = fitted, y = std)) + geom_point() +
    xlab("fitted") + ylab("standard residual") + geom_hline(yintercept = 0) +
    ggtitle("Residual Plot") + theme(plot.title = element_text(size = 9))
grid.arrange(p1, p2, nrow = 1, ncol = 2, bottom = "Figure 2: Plot of log transformation on both cp_new
```

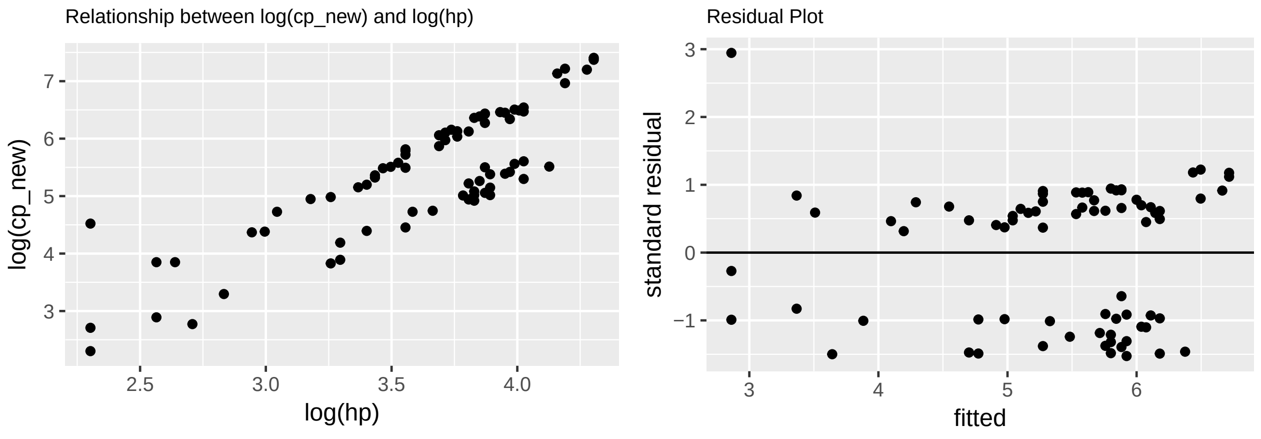

Figure 2: Plot of log transformation on both cp_new and hp

Inference

Confidence Interval for β_1

```
broom::tidy(pokemon_lm_trans, conf.int = TRUE, conf.level = 0.95)
```

```
##          term estimate std.error statistic  p.value conf.low conf.high
## 1 (Intercept)    -1.57     0.525      -3.0 3.73e-03    -2.62    -0.527
## 2     log(hp)     1.93     0.144      13.4 2.78e-21     1.64     2.214
```

Mean and Prediction Internval

Mean interval and prediction interval of $\log(cp\_new)$ when hp is 40.

```
crit_val <- qt(0.95, glance(pokemon_lm_trans)$df.resid)
val <- data.frame(hp = c(40))
pred <- augment(pokemon_lm_trans, newdata = val, type.predict = "response")
.se.pred <- sqrt(glance(pokemon_lm_trans)$sigma^2 + pred$.se.fit)

pred <- pred %>% mutate(lower_PI = .fitted - crit_val * .se.pred,
    upper_PI = .fitted + crit_val * .se.pred)

pred
```

```
##   hp .fitted .se.fit lower_PI upper_PI
## 1 40    5.53    0.07     4.44     6.62
```

```
summary(pokemon_lm_trans)$r.squared
```

```
## [1] 0.71
```

We ran a confidence interval for β_1 of ln(cp_new)~ln(hp), and we are 95% confident that the true slope falls within the interval of (1.64, 2.214).

We also made a confidence interval for the mean $log(cp\_new)$ at $hp = 40$, and we are 95% confident that the mean $log(cp\_new)$ value at $hp = 40$ falls within (5.41, 5.65) which corresponds to (223.6, 284.3) when back-transformed to regular cp_new units. We chose $hp = 40$ because it seemed to be the "central" value in our data.

Lastly, we made a prediction interval for an individual $log(cp\_new$ at $hp = 40$, and this means that 95% of $log(cp\_new$'s for Pokemon with $hp = 40$ falls within (4.44, 6.62).

We had an R^2 of 0.71 which means that 71% of the variability in $log(cp\_new)$ is explained by $log(hp)$ in our log-transformed model, $log(cp\_new)$ $log(hp)$. Based on the residual plot, this current model of cp_new vs hp does not adequately describe the Eevee data points, which right now would be considered outliers in the current model. This linear model is not appropriate to be able to describe the Eevee population, and currently seems to only represent the subset of the 4 species of Pidgey, Caterpie, and Weedle.