Part 2: Simple Linear Regression

Alex Gui and Lathan Liou

Date Submitted: February 12, 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*[1], 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

# plot
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))

# residual
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")
```

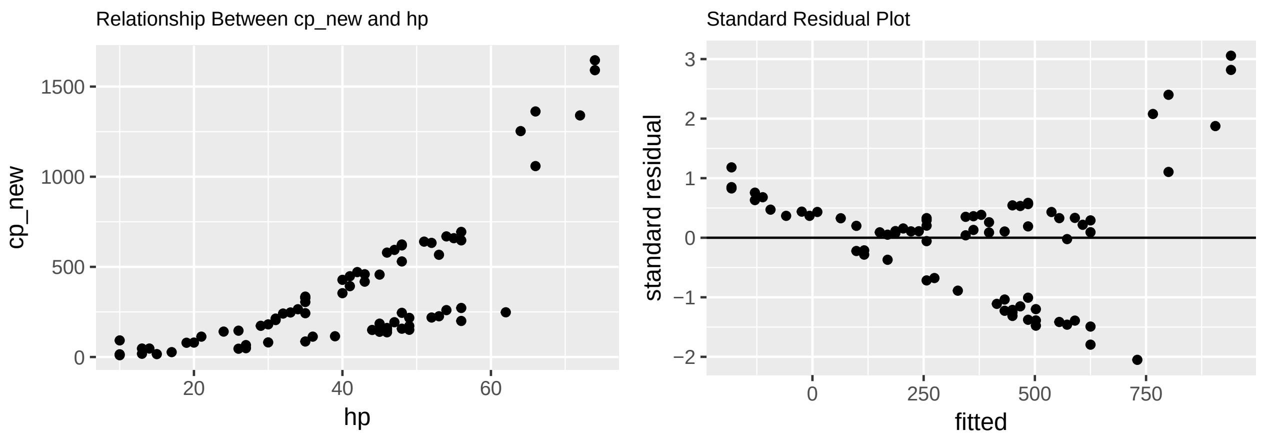

Figure 1: Plot of cp_new and hp

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

# plot
p3 <- 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))

# residual
p4 <- 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(p3, p4, nrow = 1, ncol = 2, bottom = "Figure 2: Plot of log transformation
             on both cp_new and hp")
```

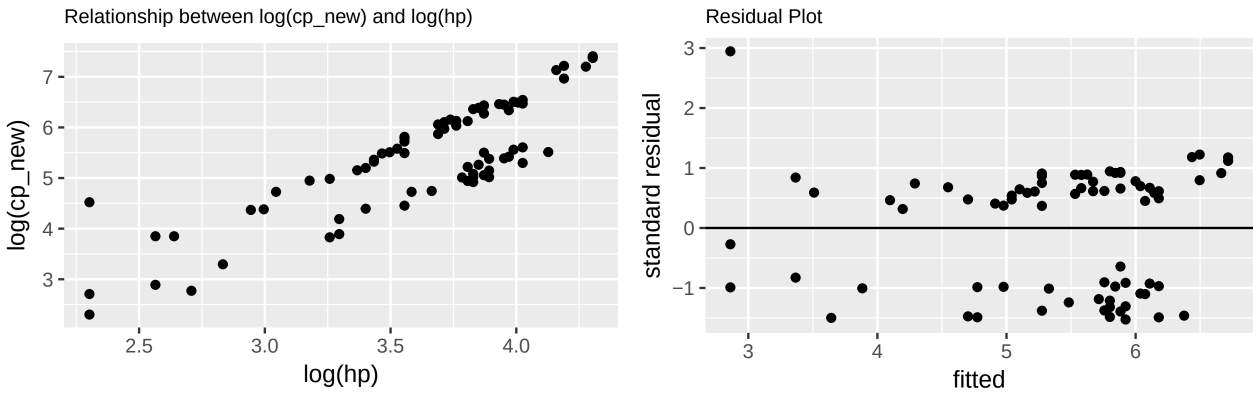

Figure 2: Plot of log transformation
on both cp_new and hp

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
```

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).

Mean Intervals and Prediction Intervals

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

```
# Unadjusted mean interval and prediction interval
crit_val <- qt(0.975, 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)

# Calculate Bonferonni, Working-Hotelling stats
num_int <- 3
W <- sqrt(2 * qf(p = 0.95, df1 = num_int, df2 = glance(pokemon_lm_trans)$df.resid))
B <- 1 - qt(0.975/num_int, glance(pokemon_lm_trans)$df.resid)

pred <- pred %>% mutate(lower_PI = .fitted - crit_val * .se.pred,
    upper_PI = .fitted + crit_val * .se.pred, lower_CI = .fitted -
        crit_val * .se.fit, upper_CI = .fitted + crit_val * .se.fit,
    lower_CI_bonf = .fitted - B * .se.fit, upper_CI_bonf = .fitted +
        B * .se.fit, lower_CI_WH = .fitted - W * .se.fit, upper_CI_WH = .fitted +
        W * .se.fit)
```

```r
# mean intervals over entire range of x

pokemon_gl <- glance(pokemon_lm_trans)
pokemon_sig <- dplyr::pull(pokemon_gl, sigma)

pred <- augment(pokemon_lm_trans) %>% mutate(.se.pred = sqrt(pokemon_sig^2 +
    .se.fit^2)) %>% mutate(lower_PI = .fitted - crit_val * .se.pred,
    upper_PI = .fitted + crit_val * .se.pred, lower_CI = .fitted -
        crit_val * .se.fit, upper_CI = .fitted + crit_val * .se.fit,
    lower_CI_bonf = .fitted - B * .se.fit, upper_CI_bonf = .fitted +
        B * .se.fit, lower_CI_WH = .fitted - W * .se.fit, upper_CI_WH = .fitted +
        W * .se.fit)

# plot with unadjusted, bonferroni, and working-hotelling
p5 <- ggplot(pred, aes(x = log.hp., y = log.cp_new.)) + geom_point() +
    stat_smooth(method = "lm", se = TRUE, fill = "darkolivegreen4") +
    geom_ribbon(data = pred, aes(ymin = lower_CI_bonf, ymax = upper_CI_bonf),
        fill = "darkolivegreen1", alpha = 0.2) + geom_ribbon(data = pred,
    aes(ymin = lower_CI_WH, ymax = upper_CI_WH), fill = "gray36",
    alpha = 0.2) + ggtitle("Mean Intervals") + theme(plot.title = element_text(size = 9))

# calculate Scheffe stat
S <- sqrt(num_int * qf(p = 0.95, df1 = num_int, df2 = glance(pokemon_lm_trans)$df.resid))

# prediction intervals for response
pred <- pred %>% mutate(lower_PI = .fitted - crit_val * .se.pred,
    upper_PI = .fitted + crit_val * .se.pred, lower_PI_bonf = .fitted -
        B * .se.pred, upper_PI_bonf = .fitted + B * .se.pred,
    lower_PI_S = .fitted - S * .se.pred, upper_PI_S = .fitted +
        S * .se.pred, lower_CI = .fitted - crit_val * .se.fit,
    upper_CI = .fitted + crit_val * .se.fit, lower_CI_bonf = .fitted -
        B * .se.fit, upper_CI_bonf = .fitted + B * .se.fit, lower_CI_WH = .fitted -
        W * .se.fit, upper_CI_WH = .fitted + W * .se.fit)

# prediction intervals over entire range of x
pred <- augment(pokemon_lm_trans) %>% mutate(.se.pred = sqrt(pokemon_sig^2 +
    .se.fit^2)) %>% mutate(lower_PI = .fitted - crit_val * .se.pred,
    upper_PI = .fitted + crit_val * .se.pred, lower_PI_bonf = .fitted -
        B * .se.pred, upper_PI_bonf = .fitted + B * .se.pred,
    lower_PI_S = .fitted - S * .se.pred, upper_PI_S = .fitted +
        S * .se.pred, lower_CI = .fitted - crit_val * .se.fit,
    upper_CI = .fitted + crit_val * .se.fit, lower_CI_bonf = .fitted -
        B * .se.fit, upper_CI_bonf = .fitted + B * .se.fit, lower_CI_WH = .fitted -
        W * .se.fit, upper_CI_WH = .fitted + W * .se.fit)

# plot with unadjusted, bonferroni, and scheffe
p6 <- ggplot(pred, aes(x = log.hp., y = log.cp_new.)) + geom_point() +
    stat_smooth(method = "lm", se = FALSE) + geom_ribbon(data = pred,
    aes(ymin = lower_PI, ymax = upper_PI), fill = "darkolivegreen4",
    alpha = 0.2) + geom_ribbon(data = pred, aes(ymin = lower_PI_bonf,
    ymax = upper_PI_bonf), fill = "darkolivegreen1", alpha = 0.2) +
    geom_ribbon(data = pred, aes(ymin = lower_PI_S, ymax = upper_PI_S),
        fill = "gray36", alpha = 0.2) + ggtitle("Prediction Intervals") +
    theme(plot.title = element_text(size = 9))

grid.arrange(p5, p6, nrow = 1, ncol = 2, bottom = "Figure 3: Plots of Mean Intervals and Prediction Inte
```

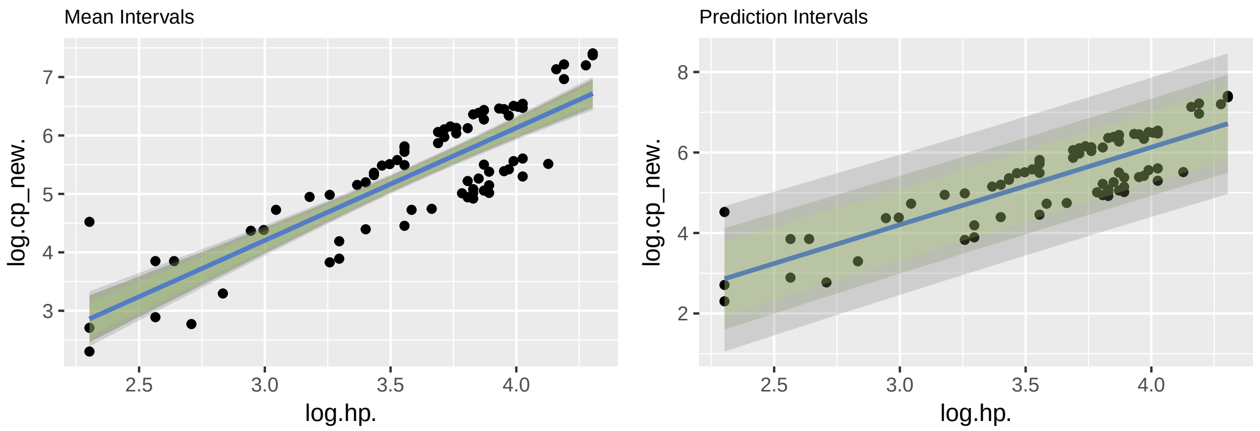

Figure 3: Plots of Mean Intervals and Prediction Intervals

For our mean interval, we used three different methods to generate different bounds for our mean 95% confidence intervals: no adjustment, Bonferroni-adjusted, and Working-Hotelling-adjusted. The 95% confidence intervals are as follows for the mean $log(cp\_new)$ at $hp = 40$:

$Unadjusted : (5.39, 5.67)$

$Bonferroni : (5.43, 5.63)$

$Working - Hotelling : (5.37, 5.71)$

We are 95% confident that the mean $log(cp\_new)$ value at $hp = 40$ falls within (5.39, 5.67) which corresponds to (223.6, 284.3) when back-transformed to cp_new units, or (5.43, 5.63) which corresponds to (228.1, 278.7) if we perform a Bonferroni adjustment, or (5.37, 5.71) which corresponds to (214.8, 301.9) if we perform a Working-Hotelling adjustment. We chose $hp = 40$ because it seemed to be the "central" value in our data.

For our prediction intervals, we also used three different methods to generate different bounds for our 95% prediction intervals: no adjustment, Bonferroni-adjusted, and Scheffé-adjusted. The 95% prediction intervals are as follows for the mean $log(cp\_new)$ at $hp = 40$:

$Unadjusted : (4.23, 6.84)$

$Bonferroni : (4.58, 6.48)$

$Scheffé : (3.66, 7.4)$

95% of $log(cp\_new)$'s for Pokemon with $hp = 40$ falls within (4.44, 6.62) which corresponds to (84.7, 750) when back-transformed, or (97.5, 652) if we perform a Bonferroni adjustment, or (38.8, 1636) if we perform a Scheffé adjustment. Generally speaking, it is important to adjust for multiple comparisons because the more comparisons that are made, the more likely erroneous inferences are to occur. The way I think about it is if someone was testing whether new drug X was better than all other existing drugs and kept on testing the effect of the drug on as many symptoms as possible, the scientist is bound to find at least one symptom in which the drug improves due to random sampling alone. Adjustments have been developed to counteract this false discovery rate.

The Bonferroni adjustment seems to have tightened both the confidence and prediction intervals so if we were mainly concerned with reducing our Type I error rate at the cost of power, than we would go with the Bonferroni. On the other hand, if we wanted to increase power, we could go with the Working-Hotelling and Scheffé adjustments for the CI and PI respectively.

Determination of R^2

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

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

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.

Conclusion

Overall, log-transforming our original SLR, cp_new hp to $log(cp\_new)$ $log(hp)$ improved the residual plot and better addressed the conditions necessary for inference of a linear regression model. We made 95% confidence intervals on the slope as well as the mean cp_new at $hp = 40$ of our log-transformed model and a 95% prediction interval of individual cp_new at $hp = 40$. What was interesting was that there seemed to be a really high correlation between hp and cp_new, whereas in Part 1, when we did an exploratory analysis, for one of our plots involving another variable called $power\_up\_stardust$, at first glance, it looked to be almost a perfect correlation, with the exception of the Eevee data points. We weren't too surprised by the results of the data because after our explanatory analysis, we were expecting a good correlation between hp and cp_new.

Moving forward, we wonder how adding other predictor variables will affect the fit of the model. In addition, is hp correlated with any other predictor variables within the data set? Lastly, what changes do we need to make to the model in order to better account for the Eevee data points?

Sources

- OpenIntro (https://www.openintro.org/stat/data/?data=pokemon). This is where downloaded the csv for our data.
- baptiste (https://stackoverflow.com/questions/30299529/ggplot2-define-plot-layout-with-grid-arrange-as-argument-of-do-call). How we make nicely arranged graphs.
- sape research group (http://sape.inf.usi.ch/quick-reference/ggplot2/colour) Colors for ggplot are great