MAT 488 Lab 4

Fred Kaesmann

April 2020

In order to better compete with Two Dogs Brewing down the road, Copy Cat Brewers has contacted Fred's Exectuive Research and Statistical Analysis: t to z to model yeast fermentation. A more efficient fermentation profile can reduce variablility in final product, can better resource management, and possibly increase the number of yearly batches per product.

To create this fermentation profile we model the first 8 days of IPA fermentation, until our anticipated new crash point. This fermentation profile, or fermentation curve, serves as a guide to IPA fermentation. A curve modeled from historical data would eliminate the need for consistent testing, freeing up staff for more pressing issues.
[1] Brewers and labratory staff can compare new fermentations against this curve, and would be quickly able to see any alarming deviations in the trend.

The Data

The Copy Cat researcher responsible for the data, though difficult to work with, [2] has generously provided access to her extensive project.

Collection

This data was collected by labratory staff at Copy Cat, compiled into an excel file, and was subject to significant cleaning by the current author.

Staff collected sterile samples from fermentation tanks, run the liquid samples through a centrifuge, and ultimately process them in Gas Chromatigraph Multi Spectrometer (or, lovingly, the GCMS). The GCMS measures the presence of Viscenal Dikeytone, or VDK, in each sample. Lab staff are notified by brewers when fermentation begins, then samples are taken regularly by lab staff on a [3] daily basis.

The Set

We have four numeric responses in our data set, the VDK, the pH, ABV and the Gravity (amount of consumable sugar remaining). We provide a short description of all four, though our main concern is VDK, followed by Gravity.

- VDK: Viscinal Diketone, a chemical which smells like butter, and is considered an off flavor for beer. When brewing, VDK levels should be kept below the human taste threshold, which is either 30, 50, or 80 depending on style. Lighter beers (i.e. pilsners) require a lower level (30), the hops in IPAs can

[1] As we write this in May 2020, the need may not be to free up staff but but to reduce the workload on already reduced staff.
[2] Which pales in comparison to the difficulty of living with this individual for approximately 20 years.
[3] Mostly

mask the flavor up to the 50 ppb threshold, while stouts and heavily fruited beers can sometimes mask levels of up to 80 ppb.

- GRAV: The remaining sugar available for yeast consumption. As the yeast consume sugar they emit alcohol, raising the ABV.
- ABV: The Alcohol by Volume, byproduct of fermentation.
- pH: pH is monitored as an indicator for fermentation integrity. Additionally when brewing sours, it indicates how sour the beer is.

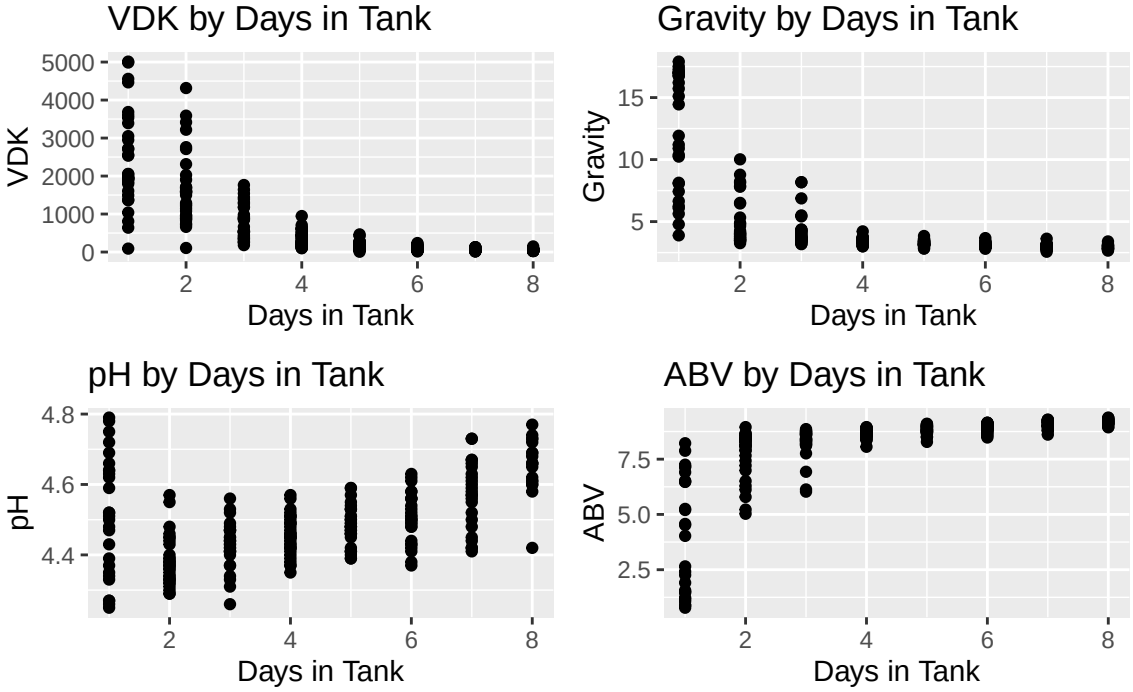
At a Glance

To demonstrate the relationships in the data, we provide a correlation matrix below.

Table 1: Iinterfactor Correlation Matrix

	VDK	pH	ABV	Gravity
VDK	1.0000	-0.2998	-0.5979	0.5886
pH	-0.2998	1.0000	-0.1487	0.1995
ABV	-0.5979	-0.1487	1.0000	-0.9934
Gravity	0.5886	0.1995	-0.9934	1.0000

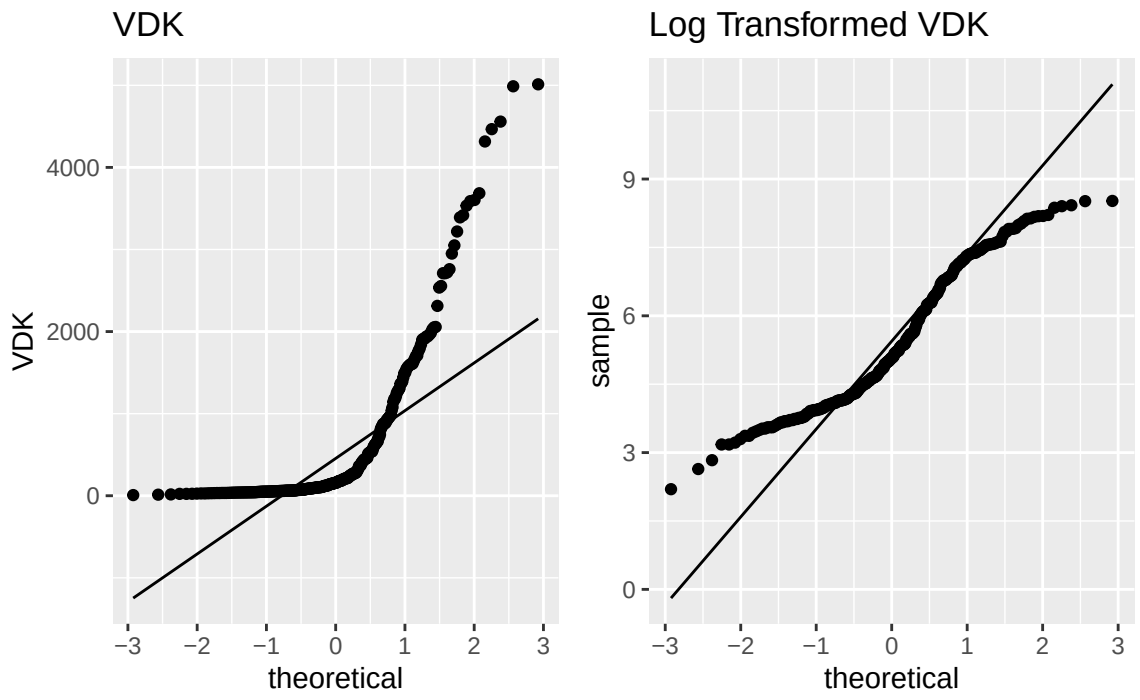
We see our nearly 1 to 1 correlation between ABV and Gravity, as the yeast consume the available sugars (Gravity) they emit alcohol, raising the ABV of the beer.



Linear Model

Log Transformed

We chart our data against a normal distribution. We see that our original data does not hold to our QQ line at all, and while we see deviations at the tails of our trasformed data, the middle of our distribution appears much more normal.



We note that our Log VDK's fail both the Shapiro-Wilk Normality Test, as well as the One-Sample Kolmogorov-Smirnov Test. [4] [5]

Model

We model the log transformed data by the following

$$\hat{y} = \beta_0 + \beta_1 x$$

then we reverse our transformation and have

$$\hat{y} = e^{\beta_0 + \beta_1 x}$$
$$\text{VDK} = e^{\beta_0 + \beta_1 \cdot \text{Days in Tank}}$$

Our linear model returned an intercept of 8.158, and a coefficient of -0.586 for days in tank. We have significant p-values for both the intercept as well as the Days in Tank. The model had an F-statistic of 1256,

[4] Yet we carry on.

[5] We would also like to note at this point that were we more comfortable with logistic regression at the time this paper was submitted the first time we would have opted for that approach.

as well as an adjusted R^2 of 0.8314. We feel comfortable claiming we have an accurate model, given that by our adjusted R^2 we can model about 83% of the variation just by doing a simple modeling by day.

From our intercept and slope estimates we have the equation

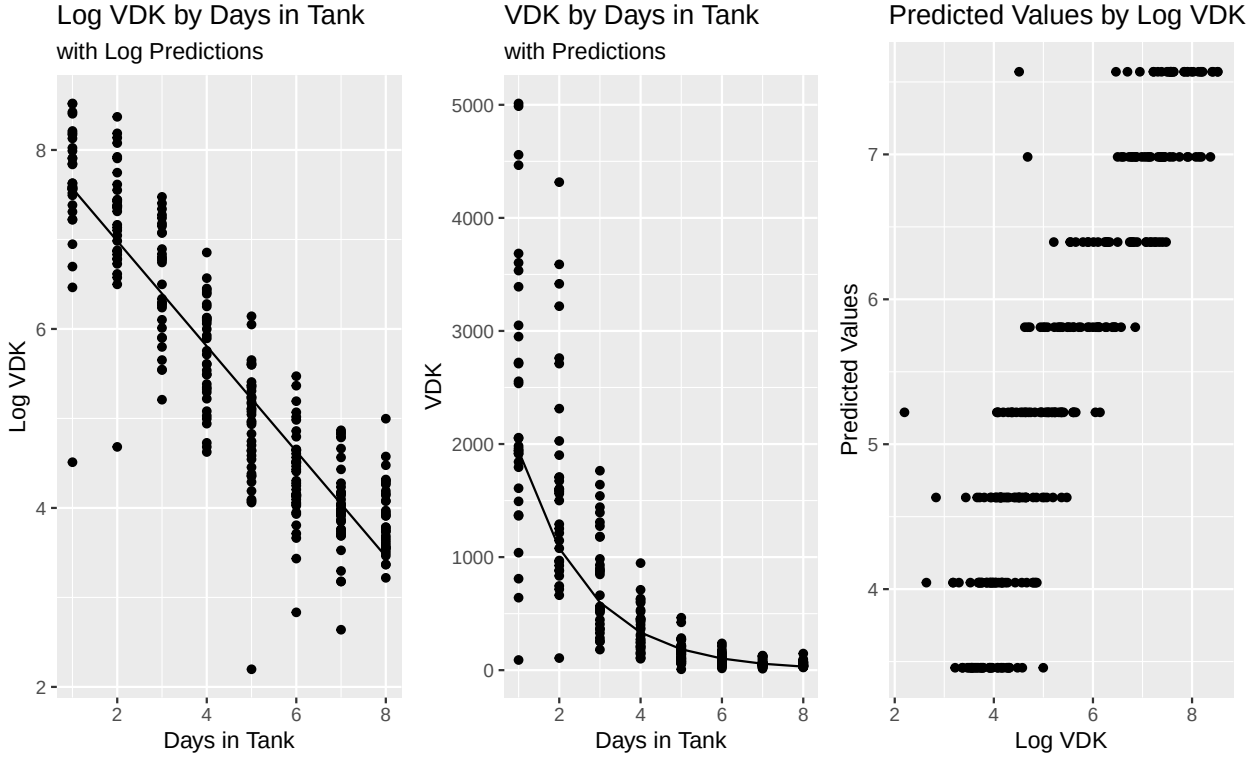
$$\hat{y} = 8.158 - 0.5876x$$

which when exponentiated to fit the original data becomes

$$\hat{y} = e^{8.158-0.5876x}$$
$$\text{VDK} = e^{8.158-0.5876\cdot(\text{Days in Tank})}$$

Predictions

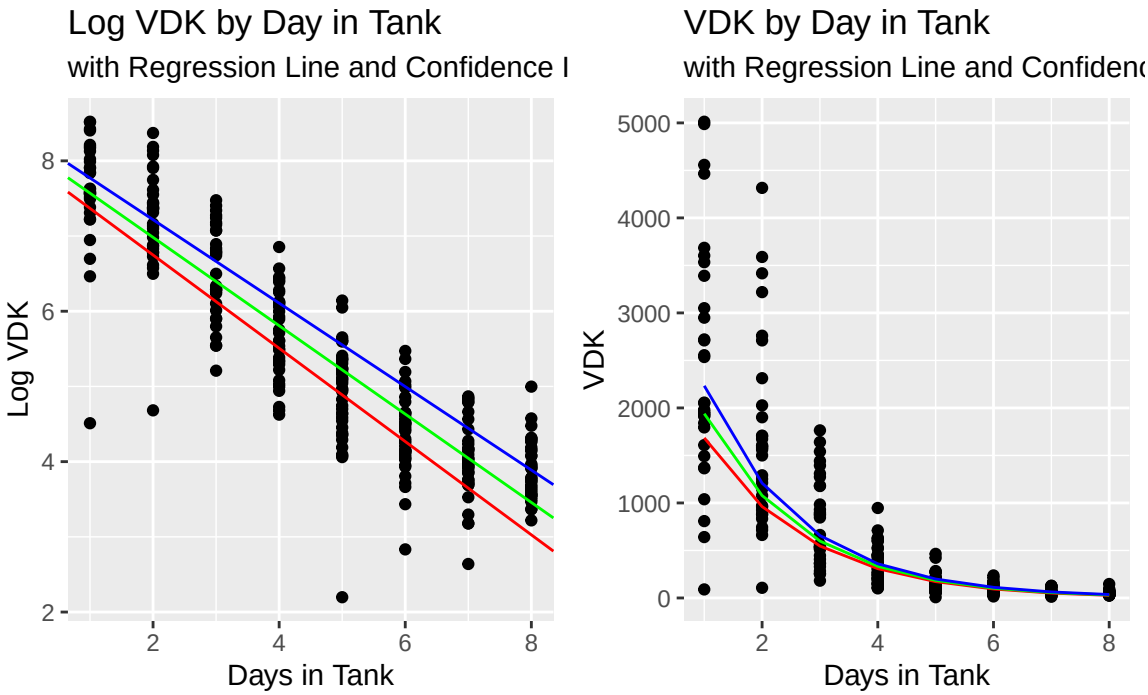
We plot the predictions in three ways. The first plot is the log transformed data with the regression line. The second is the original data with the exponentiated regression line. The third is the predicted values by the log transformed values.



The first two charts above show that we do pretty well predicting days 1 through 7, though we seem to underpredict the 8th day. We see this in the shift of the lowest set of points on the y axis in the third chart. Additionally it's worth noting here that the possible outliers are all present as significantly lower VDK readings. The consistency of the outliers leads us to question whether or not they are actually outliers, or whether they represent misattributed data. These could have been a result of late sampling or mislabeled data. For the purposes of this analysis they have been kept in the set.

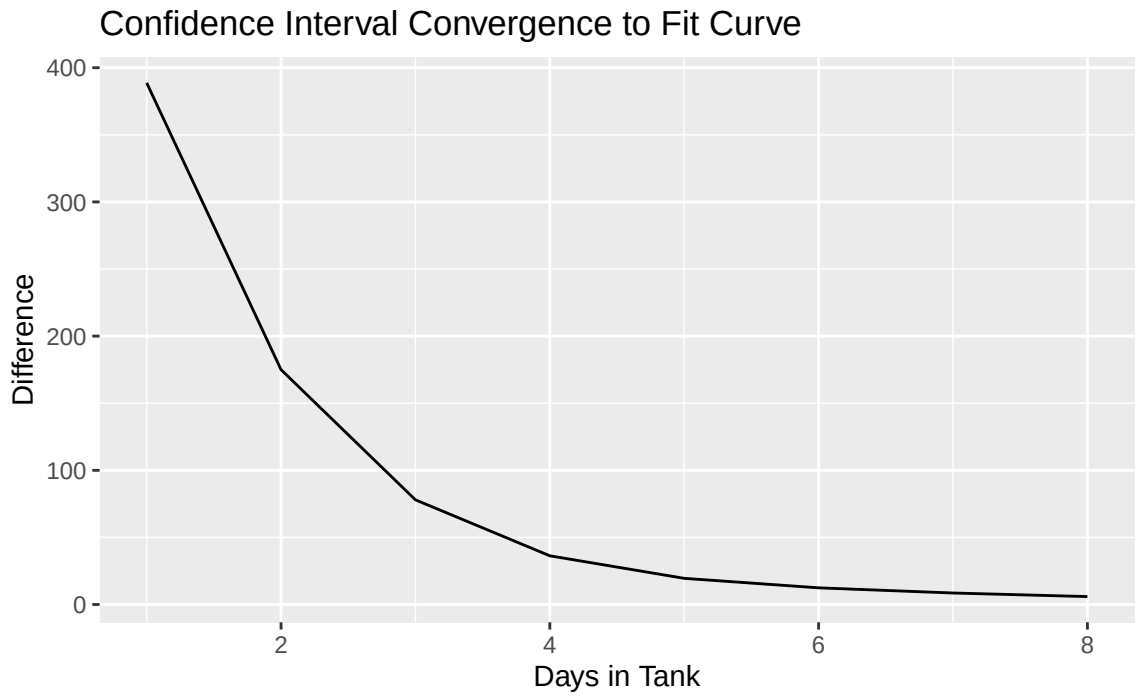
Confidence Intervals We evaluate out model at the 95% confidence interval.

	2.5 %	97.5 %
Intercept	7.9881721	8.3273481
Days in Tank	-0.6201782	-0.5549205



The following was used in evaluating spread of the confidence intervals

$$\text{Difference} = \sqrt{(e^{\text{upr}} - e^{\text{fit}})^2 + (e^{\text{lwr}} - e^{\text{fit}})^2}$$



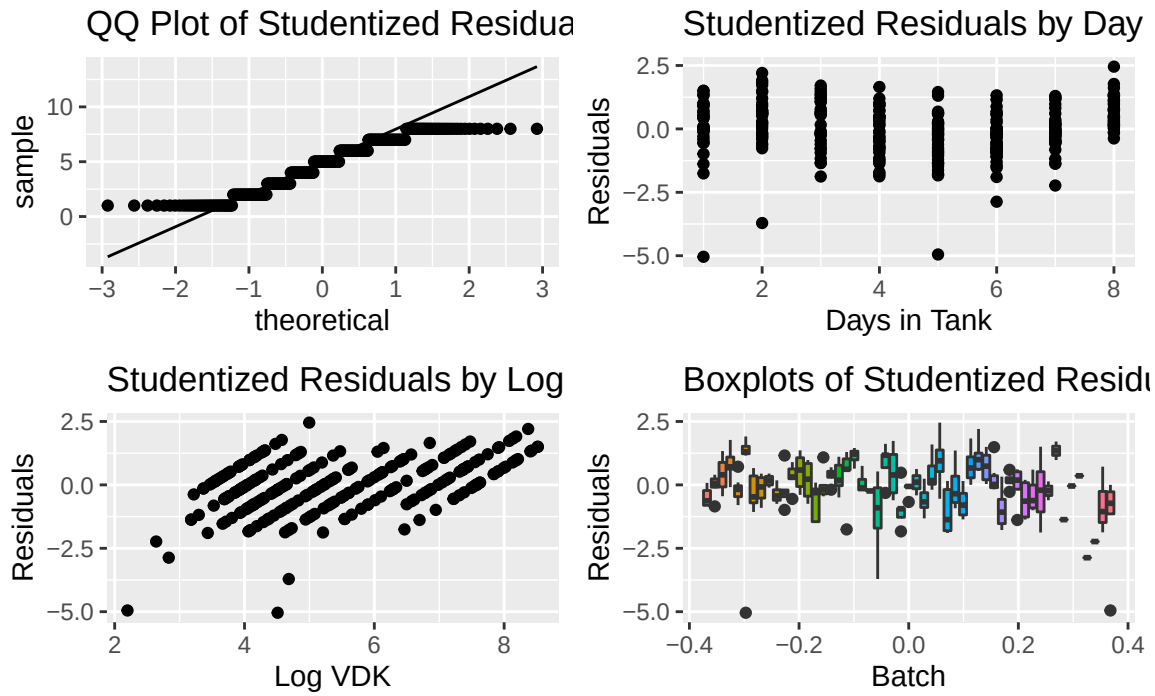
Our curves show our that as our x value increases our the confidence interval for our predictions shows a narrower range. Assuming our data is not overfit we would feel confident using the predictions for the later dates.

Residual Analysis

We present four visualizations for our residuals.

1. The QQ plot for normal distribution of residuals
2. Residuals by Day in Tank
3. Residuals by Log VDK
4. Boxplots of Residuals by Batch

Before visualizing we studentized the residuals. We opted to studentized as opposed to standardized given our concerns about outliers in the data.



We see in the various charts above that most of our studentized residuals are distributed around 0, with most falling within the -2.5 to 2.5 range. We do note however the few -5's, our earlier possible outliers continue to present themselves. Additionally, we see a wave present in our residuals by days in tank, indicating some other trend we may have missed.

Shapiro vs Kolmogorov-Smirnov We test our Studentized residuals with both the Shapiro test as well as the Kolmogorov-Smirnov test.

W-Statistic	p-value
0.949	$1.77e^{-08}$

Table 3: Shapiro-Wilk Normality Test

D-Statistic	p-value
0.060	0.253

Table 4: One-Sample Kolmogorov-Smirnov Test

From our two tables above, we see that our residuals fail the Shapiro test, however they don't fail the Komogorov-Smirnov test, which is less sensitive to sample size. Given the charts above we are inclined to believe the Kolmogorov-Smirnov Test.

In Conclusion

We feel comfortable using our model as a guide to better planning and organizing fermentations and their associated resources. We note that our model does a better job at predicting VDk's after a few days in tank. We assert this is a feature and not a bug, as we are more concerned with when our beer crosses the magic 40ppb of VDK. [6]

[6] Again, had we been more familiar with Logistic Regression we would have tried our hand at that.

Appendix I

On Outliers Present in the Data

We would like to mention that the outliers present in the data could have come from factors other than the natural fermentation process itself. There are known issues with some of the technicians who collected and logged the data, and as a result, some samples collected may have been incorrectly marked when sampled. Additionally, we are aware of occasionaly significant time lapses between the actual start of the fermentation and when the information was recorded and initial samples taken.

Appendix II

The following questionable code was used in the production of this document

```
##### RMarkdown Document Options ####
library(knitr)
opts_chunk$set(echo = FALSE,
                      include = FALSE,
                      warning = FALSE,
                      message = FALSE,
               # Figure Dimensions
               fig.width = 6,
               fig.asp = 0.618)

##### Load Libraries ####
library(readxl)
library(magrittr)
library(tidyverse)
library(ggplot2)
library(gridExtra)

##### Import Data ####
imported_data <- read_xlsx("data/VDKs.xlsx")

# We only want a subset of the data
df <- imported_data[substr(imported_data$Batch, 1, 3) == "TRB", ] # Just the Two Roads Brewery R2R
df <- df[df$VDK > 0, ]
df <- df[df$`Recipe Days Post KO` < 9, ]
df <- df[c(1:3, 6:10)]
df <- df[!is.na(df$VDK), ]

##### Append Log Transformed Data ####
df <- df %>%
  mutate(VDK_log = log(VDK+1),
         GRAV_log = log(GRAV+1))


##### QQ plot, Original Data ####
norm1 <- df %>%
  ggplot(aes(sample = VDK)) +
  geom_qq() +
  geom_qq_line() +
  ggtitle("Normality Plot","Original VDK")

##### QQ plot, Log Transformed Data ####
norm2 <- df %>%
  ggplot(aes(sample = VDK_log)) +
  geom_qq() +
  geom_qq_line() +
  ggtitle("Normality Plot",
          "Log Transformed VDK")

##### Arrange Data Nicely ####
grid.arrange(norm1, norm2, nrow = 1)
```

```
##### Log VDK Shapiro Test ####
shapiro.test(df$VDK_log)

##### Log VDK KS Test ####
ks.test(df$VDK_log, "pnorm")

##### VDK by Days in Tank Model ####
model_dit <- lm(VDK_log ~ `Recipe Days Post KO`, data = df)
summ_dit <- summary(model_dit)
summ_dit

##### Log Predictions by Days in Tank ####
pred1 <- df %>%
  mutate(predictions = predict(model_dit)) %>%
  ggplot(aes(x = `Recipe Days Post KO`)) +
  geom_point(aes(y = VDK_log)) +
  geom_line(aes(y = predictions)) +
  xlab("Days in Tank") +
  ylab("Log VDK")  +
  ggtitle("Log VDK by Days in Tank",
          "with Log Predictions")

##### Predictions by Days in Tank ####
pred2 <- df %>%
  mutate(predictions = exp(predict(model_dit))) %>%
  ggplot(aes(x = `Recipe Days Post KO`)) +
  geom_point(aes(y = VDK)) +
  geom_line(aes(y = predictions)) +
  xlab("Days in Tank") +
  ylab("VDK")  +
  ggtitle("VDK by Days in Tank",
          "with Predictions")

##### Predictions by Log VDK ####
pred3 <- df %>%
  mutate(predictions = predict(model_dit)) %>%
  ggplot(aes(x = VDK_log)) +
  geom_point(aes(y = predictions)) +
  xlab("Log VDK") +
  ylab("Predicted Values")  +
  ggtitle("Predicted Values by Log VDK")

##### Arrange Predictions Nicely ####
grid.arrange(pred1, pred2, pred3, nrow=1)

##### Confidence Intervals ####
# As object to change dimnames
interval <- confint(model_dit)
dimnames(interval)[[1]] <- c("Intercept", "Days in Tank")

##### Include Nice Table of Intervals ####
kable(interval)
```

```
##### Log Confidence Intervals Chart ####
cf1 <- df %>%
  ggplot(aes(x = `Recipe Days Post KO`,
             y = VDK_log)) +
  geom_point() +
  geom_abline(intercept = interval[1],
              slope = interval[2],
              color = "red") +
  geom_abline(intercept = interval[3],
              slope = interval[4],
              color = "blue") +
  geom_abline(intercept = model_dit$coefficients[1],
              slope = model_dit$coefficients[2],
              color = "green") +
  xlab("Days in Tank") +
  ylab("Log VDK") +
  ggtitle("Log VDK by Day in Tank",
          "with Regression Line and Confidence Intervals")

##### Bind Confidence Interval Predictions to Our Frame ####
predictions <- predict(model_dit, interval = "confidence") %>%
  as.data.frame()
df <- cbind(df, predictions)

##### Confidence Interval Lines on Original Data ####
cf2 <- df %>%
  ggplot(aes(x = `Recipe Days Post KO`,
             y = VDK)) +
  geom_point() +
  geom_line(aes(y = exp(lwr)), color = "red") +
  geom_line(aes(y = exp(fit)), color = "green") +
  geom_line(aes(y = exp(upr)), color = "blue") +
  xlab("Days in Tank") +
  ylab("VDK") +
  ggtitle("VDK by Day in Tank",
          "with Regression Line and Confidence Intervals")

##### Arrange Interval Plots Nicely ####
grid.arrange(cf1, cf2, nrow = 1)

##### Convergence Plot ####
# Inclusing TBD
df %>%
  mutate(convergence = ((exp(upr) - exp(fit))^2 + (exp(lwr) - exp(fit))^2)^.5) %>%
  ggplot(aes(x = `Recipe Days Post KO`,
             y = convergence)) +
  geom_line() +
  xlab("Days in Tank") +
  ylab("Difference") +
  ggtitle("Confidence Interval Convergence to Fit Curve")

##### QQ Plot of Residuals ####
res1 <- df %>%
```

```
  mutate(residuals = rstudent(model_dit)) %>%
  ggplot(aes(sample = `Recipe Days Post KO`)) +
  geom_qq() +
  geom_qq_line() +
  ggtitle("Normality Plot of Studentized Residuals")

##### Residuals by Day in Tank ####
res2 <- df %>%
  mutate(residuals = rstudent(model_dit)) %>%
  ggplot(aes(x = `Recipe Days Post KO`,
             y = residuals)) +
  geom_point() +
  xlab("Days in Tank") +
  ylab("Residuals") +
  ggtitle("Studentized Residuals by Day in Tank")

##### Residuals by Actual ####
res3 <- df %>%
  mutate(residuals = rstudent(model_dit)) %>%
  ggplot(aes(x = VDK_log,
             y = residuals)) +
  geom_point() +
  xlab("Log VDK") +
  ylab("Residuals") +
  ggtitle("Studentized Residuals by Log VDK")

##### Boxplots of Residuals by Batch ####
res4 <- df %>%
  mutate(residuals = rstudent(model_dit)) %>%
  ggplot(aes(y = residuals,
             fill = Batch)) +
  geom_boxplot(show.legend = F) +
  xlab("Batch") +
  ylab("Residuals") +
  ggtitle("Boxplots of Studentized Residuals")

##### Arrange the Residual Plots Nicely ####
grid.arrange(res1, res2, res3, res4, nrow=2)

##### Residuals Shapiro Test ####
model_dit %>%
  rstudent() %>%
  shapiro.test()

##### Residuals KS Test ####
model_dit %>%
  rstudent() %>%
  ks.test("pnorm")
```

‘‘‘