Notes 06 - Simple Linear Regression (SLR)

STS 2300 (Spring 2025)

Updated: 2025-03-10

Table of Contents

[Reading for Notes 06 1](#_Toc192529596)

[Learning Goals for Notes 06 1](#_Toc192529597)

[A “best”-fit line 2](#_Toc192529598)

[Using R for simple linear regression 4](#_Toc192529599)

[Making predictions with our SLR line 6](#_Toc192529600)

[Interpreting and 7](#_Toc192529601)

[The coefficient of determination () 8](#_Toc192529602)

[Revisiting the Learning Goals for Notes 06 9](#_Toc192529603)

# Reading for Notes 06

Read [Chapter 5](https://moderndive.com/5-regression.html) of the Modern Dive textbook. You can skip Section 5.2 since we won’t cover categorical explanatory variables in this class. (Note: You can take STS 2320 to learn more about many different forms of regression including ones using categorical explanatory variables.)

# Learning Goals for Notes 06

* Understand and be able to correctly use important regression terms like explanatory variable, response variable, and residual.
* Be able to use R to generate simple linear regression equations and to make predictions with them.
* Understand how to interpret the intercept and slope of a simple linear regression equation in context.
* Be able to determine and interpret the coefficient of determination () and understand its use and limitations.

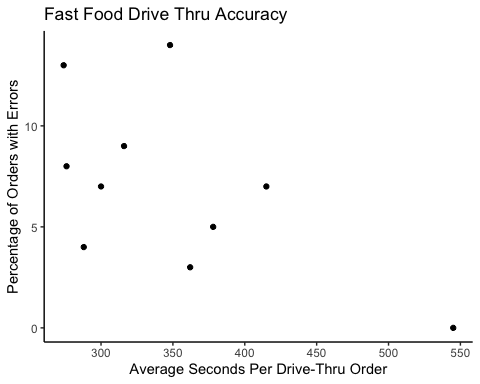
# A “best”-fit line

Simple linear regression (often abbreviated SLR) allows us to use one variable to make predictions about another variable.

* The variable for which we are making predictions is called the **response variable** (or sometimes the dependent variable).
* The variable which is helping us make predictions is called the **explanatory variable** (or sometimes the independent variable).

Let’s start by looking at an example. The scatterplot below is based off of data from [this graph](http://notawfulandboring.blogspot.com/2022/11/chartrs-speed-or-accuracy-its-hard-to.html).

fastfood <- read.csv("https://raw.githubusercontent.com/nbussberg/STS2300-Spring2025/refs/heads/main/Data/fast\_food\_accuracy.csv")  
  
library(ggplot2)  
ggplot(fastfood, aes(x = SecPerOrder,   
 y = PctWithErrors)) +  
 geom\_point() +  
 labs(x = "Average Seconds Per Drive-Thru Order",  
 y = "Percentage of Orders with Errors",  
 title = "Fast Food Drive Thru Accuracy") +  
 theme\_classic()



**Question:** What is our response variable and what is our explanatory variable in this graph?

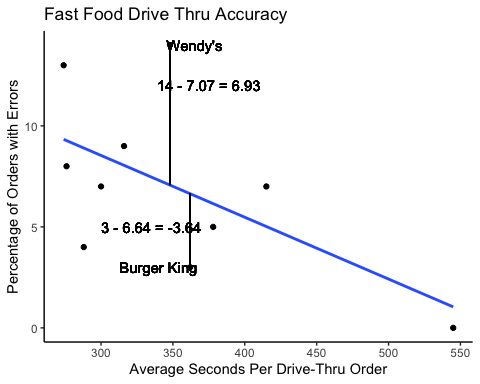
**Answer:**

The main idea behind simple linear regression is that we want to find the “best” line that fits our data. You might be able to eyeball the graph above and draw a line that looks good to you, but someone else’s line might be different. So how do we determine the “best” line? One way that we do this is by thinking about how far each point is away from our line.

A **residual** is the vertical distance between an observation and our line. In particular, we use the equation

Dots above the line will have positive residuals, and dots below the line will have negative residuals. The smaller the value, the closer the point is to the line. Below is our graph again with a line drawn through the points, and residuals added for two of the states.

* Wendy’s has values of and . The predicted response at is . Thus the residual for Wendy’s is .
* Burger King has and . The predicted response at is . Thus the residual for Burger King is .



One quality that we’d like our “best” line to have is that the residuals sum up to zero. In other words, we want the points to be balanced above and below the line. However, there are actually *lots* of lines that will have this quality, so this can’t be our only criterion. To get around this issue, we will square all of the residuals and look for the line that has the *smallest sum of the squared residuals*. There is only one line with the smallest sum of the squared residuals.

(Note: This is why a regression line is sometimes called the **least-squares regression line**). The “best” line for a data set is the one that minimizes the sum of the squared residuals (sometimes called SSR or SSE) for that data set.

# Using R for simple linear regression

If you’ve taken STS 2120, you may recall that we need to distinguish between the “true” line and our estimate of that line.

The **true model** will describe a linear relationship between our two variables for the entire population. We can write it as:

In this formula:

* is the population mean response for a given value of x.
* is the “true” or population y-intercept (a parameter)
* is the “true” or population slope (a parameter)

We will *estimate* this line based on our data. Our estimate will have the form:

In this formula:

* is the predicted response for a given value of x
* is the estimated y-intercept (a statistic)
* is the estimated slope (a statistic)

We can use the function lm() to come up with the estimates and that give us the best line for this data. The general format is:

lm(response ~ explanatory,   
 data = dataset)

For the fast food data, we want to use the SecPerOrder variable to help explain our response, PctWithErrors.

lm(PctWithErrors ~ SecPerOrder,   
 data = fastfood)

##   
## Call:  
## lm(formula = PctWithErrors ~ SecPerOrder, data = fastfood)  
##   
## Coefficients:  
## (Intercept) SecPerOrder   
## 17.7152 -0.0306

The output above tells us that our estimated y-intercept (or ) is 17.7152, and our estimated slope (or ) is -0.0306. Putting this together, our line to understand the relationship between these two variables is:

where is our prediction for the percentage of drive-thru orders with errors for a given value of x, which is average seconds per order.

**An aside about piping**

Notice how the data argument is not the first argument in the lm() function. This means that you would get an error if you were to type:

fastfood |>   
 lm(PctWithErrors ~ SecPerOrder)

If we use the dplyr pipe (%>%), we can also tell R to pipe our data into a different place in a function using a period. The code below produces the same best-fit line as we saw in our example without piping. Note: this functionality does not (yet) work with the native pipe (|>).

library(dplyr) # to use the %>% pipe  
  
fastfood %>%  
 lm(PctWithErrors ~ SecPerOrder, .)

##   
## Call:  
## lm(formula = PctWithErrors ~ SecPerOrder, data = .)  
##   
## Coefficients:  
## (Intercept) SecPerOrder   
## 17.7152 -0.0306

# Making predictions with our SLR line

If we want to use our line to make predictions, we plug in a value for and use it to calculate our prediction, .

For example, let’s say I want to predict the percentage of orders with errors for a fast food restaurant with an average drive-thru time of 500 seconds.

My best prediction is that if it takes an average of 500 seconds per drive-thru order, the fast food restaurant will make errors around 2.4% of the time.

**Practice**: Find the predicted percentage of orders with errors at a fast food restaurant with an average drive-thru order time of 300 seconds. Hint: you can use R to do the calculation!

**Answer:**

Our line will work best for making predictions that are within the range of what we’ve previously seen (in terms of the x values). These are called **interpolations**. A prediction based on an x value outside the range of what we’ve seen is called an **extrapolation**. While it’s sometimes necessary to extrapolate, these predictions are less likely to be accurate because we don’t know if the trend we’ve seen will continue outside of this range of values.

**Question**: What range of x values could we use for interpolations with this example? Would you trust a prediction more for x = 200 or for x = 100?

**Answer:**

We can use R to calculate these predicted values for us by using the predict() function. There are two main arguments for this function. The first is our model from the lm() function, and the second is a data frame of new data. Below is an example of how I can make my prediction for x = 500.

fastfood.lm <- lm(PctWithErrors ~ SecPerOrder,   
 data = fastfood)  
predict(fastfood.lm,   
 newdata = data.frame(SecPerOrder = 500))

## 1   
## 2.416527

At a restaurant with an average drive-thru time of 500 seconds, I predict that about 2.4% of orders will have errors.

(Note: This value may differ slightly from your calculation earlier because it was calculated without rounding or at all.)

# Interpreting and

We can understand and by thinking about what they mean in terms of predictions we might make. You may recall that the y-intercept is the point on our line where . In other words:

General interpretation of : The *predicted* response is when the explanatory variable is 0.

**Question:** What would this interpretation look like for this example?

**Interpretation of :**

Now clearly, this doesn’t make sense because it’s impossible for a restaurant to have an average drive-thru time of 0 seconds (and if they did, they would make errors on A LOT more than 17.7% of orders).

It is often (but not always) true that the y-intercept is an extrapolation, and in many cases our interpretation of the y-intercept won’t make sense. The smallest x value in our dataset was only 274. We have no idea what things would look like for a restaurant with an average drive-thru time so much smaller than 274.

What about interpreting , our estimated slope? For this, let’s try another prediction, this time for x = 501.

Recall that when I made a prediction for x = 500, I got . My prediction for x = 501 is 0.031 *less* than my prediction. This difference is our slope!

General interpretation of : The *predicted* response increases/decreases (based on + or - sign) by the absolute value of for every one unit increase in our explanatory variable.

**Question:** What would this interpretation look like for this example?

**Interpretation of :**

Notice that I’ve italicized the word *predicted* in both interpretations (for and ). This is because these are just my best guesses based on the line. As we can see from our scatterplot, some restaurants will fall above the line, and other restaurants will fall below the line.

# The coefficient of determination ()

We can use the summary() function on our output from the lm() function to get more information about our simple linear regression model. Let’s try that for the fast food example.

summary(fastfood.lm)

##   
## Call:  
## lm(formula = PctWithErrors ~ SecPerOrder, data = fastfood)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -4.903 -1.470 -1.095 1.725 6.933   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 17.7152 5.3486 3.312 0.0107 \*  
## SecPerOrder -0.0306 0.0149 -2.053 0.0741 .  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 3.708 on 8 degrees of freedom  
## Multiple R-squared: 0.3451, Adjusted R-squared: 0.2633   
## F-statistic: 4.216 on 1 and 8 DF, p-value: 0.07412

Notice that our estimates of and show up in an Estimate column next to (Intercept) and SecPerOrder (our explanatory variable) respectively. Next to that column is a standard error for our estimate, a test statistic, and a p-value (more on those in Notes 07).

On the second line from the bottom you’ll see “Multiple R-squared: 0.3451”. This number is called the  **value** or the **coefficient of determination**. It tells us the proportion of the variability in our response variable that can be explained by our explanatory variable.

In other words, if we just looked at PctWithErrors by itself, we could see that the variance is var(fastfood$PctWithErrors) = 18.667. However, if I have my simple linear regression line, I have explained some of the variation between the restaurants. The part I don’t understand is my residuals. The variance in my residuals is var(fastfood.lm$residuals) = 12.224.

In a simple linear regression model, . (Note: This is not true for multiple linear regression.)

**Example:** Verify for the fast food data that is actually calculated as above. Then write an interpretation of the value in context of this example.

**Calculation:**

**Interpretation:**

# Revisiting the Learning Goals for Notes 06

* Be able to use R to generate simple linear regression equations and to make predictions with them.
  + Use the mtcars data to predict a car’s mpg using it’s wt
* Understand how to interpret the intercept and slope of a simple linear regression equation in context.
  + Interpret the y-intercept and the slope of your model from the mtcars data
* Be able to determine and interpret the coefficient of determination () and understand its use and limitations.
  + Find the value and interpret it for the mtcars data