15.7.2

Return to Linear Regression

Jeremy is really starting to hit his stride. So, when his team wants to figure out how to predict one variable given another, he's pretty excited that he knows the answer: use linear regression!

Of course, before he actually uses the technique, he wants to dig into just exactly how it works. After all, he's leading a team into new territory, and it's up to him to make sure they are headed in the right direction.

Earlier we learned about linear regression and how it can be used to determine our dependent y variable from an independent x variable. In a more formal definition, **linear regression** is a statistical model that is used to predict a continuous dependent variable based on one or more independent variables fitted to the equation of a line.

In school, most students learned that the equation of a line is written as $y = mx + b$:

$$y = mx + b$$

Dependent variable — Slope — Independent variable — y-intercept

The job of a linear regression analysis is to calculate the slope and y intercept values (also known as coefficients) that minimize the overall distance between each data point from the linear model. There are two basic types of linear regression:

- **Simple linear regression** builds a linear regression model with one independent variable.
- **Multiple linear regression** builds a linear regression model with two or more independent variables.

Linear regression is popular in data science because it has multiple applications. First and foremost, linear regression can be used as a predictive modeling tool where future observations and measurements can be predicted and extrapolated from a linear model. Linear regression can also be used as an exploratory tool to quantify and measure the variability of two correlated variables.

A good linear regression model should approximate most data points accurately if two variables are strongly correlated. In other words, linear regression can be used as an extension of correlation analysis. In contrast to correlation analysis, which asks whether a relationship exists between variables A and B, linear regression asks if we can predict values for variable A using a linear model and values from variable B.

To answer this question, linear regression tests the following hypotheses:

H_0 : The slope of the linear model is zero, or m = 0

H_a : The slope of the linear model is not zero, or m ≠ 0

If there is no significant linear relationship, each dependent value would be determined by random chance and error. Therefore, our linear model would be a flat line with a slope of 0.
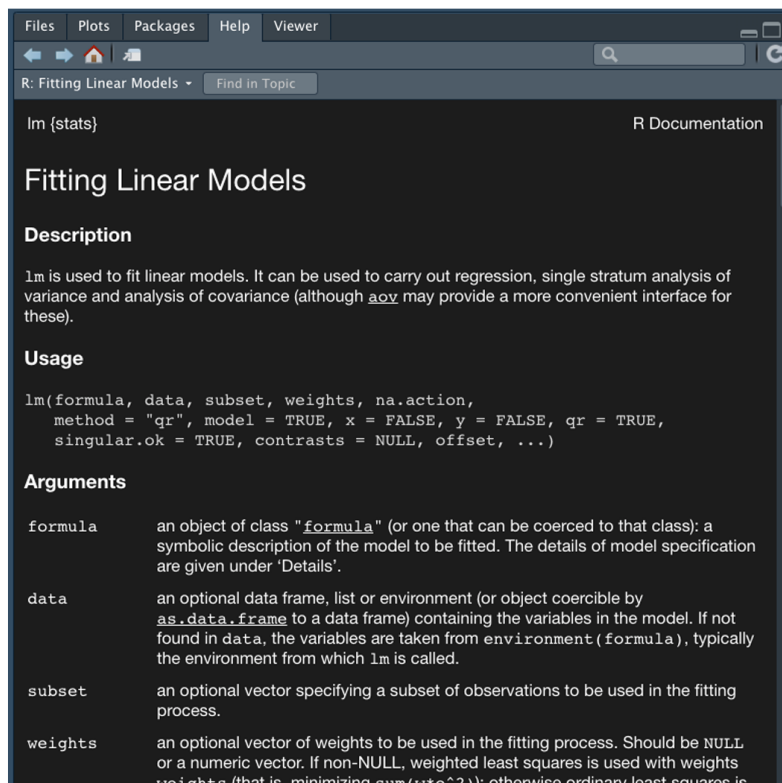
To quantify how well our linear model can be used to predict future observations, our linear regression functions will calculate an r-squared value. The **r-squared (r^2) value** is also known as the coefficient of determination and represents how well the regression model approximates real-world data points. In most cases, the r-squared value will range between 0 and 1 and can be used as a probability metric to determine the likelihood that future data points will fit the linear model.

When using a simple linear regression model, the r-squared metric can be approximated by calculating the square of the Pearson correlation coefficient between the two variables of interest.

By combining the p-value of our hypothesis test with the r-squared value, the linear regression model becomes a powerful statistics tool that both quantifies a relationship between variables and provides a meaningful model to be used in any decision-making process.

Although the interpretation of a simple linear regression is different from a multiple linear regression, their model implementation is the same. In R, we'll build our linear models using the built-in `lm()` function. Type the following code into the R console to look at the `lm()` documentation in the Help pane:

```
>?lm()
```

Even though there are many optional arguments for the `lm()` function, the `lm()` function only requires us to provide two arguments:

- **formula**

- **data**

↻ Retake

Similar to our t-test analysis, there are a few assumptions about our input data that must be met before we perform our statistical analysis:

1. The input data is numerical and continuous.

2. The input data should follow a linear pattern.

3. There is variability in the independent *x* variable. This means that there must be more than one observation in the x-axis and they must be different values.

4. The residual error (the distance from each data point to the line) should be normally distributed.

| IMPORTANT |

> Validating the fourth assumption is outside the scope of this course as it involves more robust statistical methods. However, in most real-world cases, we can expect our data to meet the fourth assumption.
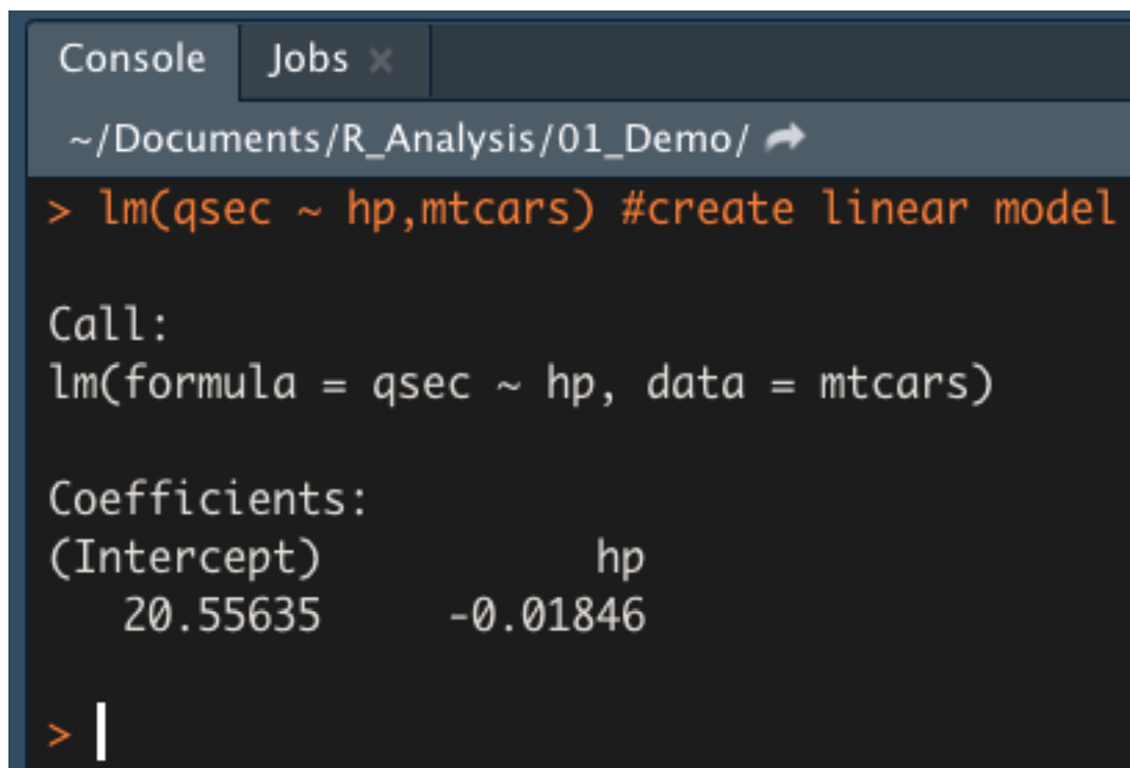
Once we have our data in a single data frame that meets the assumptions of our linear regression analysis, we're ready to implement the `lm()` function.

For practice, let's revisit our correlation example using the mtcars dataset. Using our simple linear regression model, we'll test whether or not quarter-mile race time (`qsec`) can be predicted using a linear model and horsepower (`hp`).

Remember from our correlation example that our Pearson correlation coefficient's r-value was -0.71, which means there is a strong negative correlation between our variables. Therefore, we anticipate that the linear model will perform well.

To create a linear regression model, our R statement would be as follows:
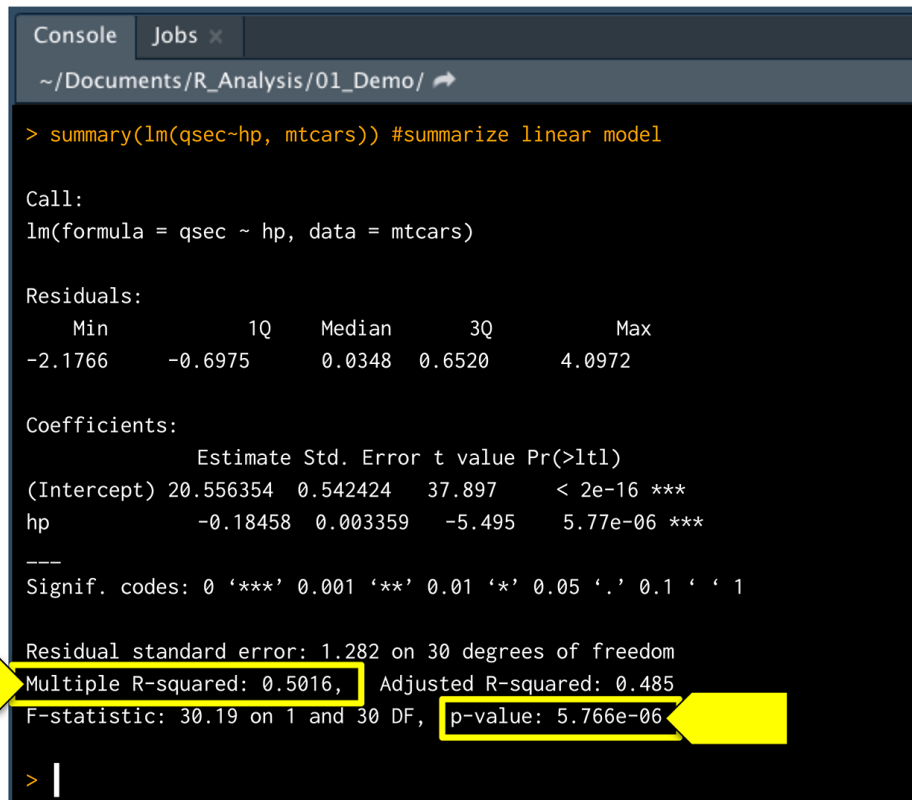
```
> lm(qsec ~ hp,mtcars) #create linear model
```

```
Console    Jobs ×
~/Documents/R_Analysis/01_Demo/
> lm(qsec ~ hp,mtcars) #create linear model

Call:
lm(formula = qsec ~ hp, data = mtcars)

Coefficients:
(Intercept)                hp
   20.55635          -0.01846

>
```

The output of the `lm()` function will be the metrics from our model. Specifically, the `lm()` function returns our y intercept (`Intercept`) and slope (`hp`) coefficients. Therefore, the linear regression model for our dataset would be

`qsec = -0.02hp + 20.56`.

To determine our p-value and our r-squared value for a simple linear regression model, we'll use the `summary()` function:

```
> summary(lm(qsec~hp,mtcars)) #summarize linear model
```

```
Console   Jobs ×
~/Documents/R_Analysis/01_Demo/ ➔
> summary(lm(qsec~hp, mtcars)) #summarize linear model

Call:
lm(formula = qsec ~ hp, data = mtcars)

Residuals:
    Min         1Q    Median       3Q        Max
-2.1766    -0.6975    0.0348    0.6520     4.0972

Coefficients:
               Estimate Std. Error t value Pr(>ltl)
(Intercept) 20.556354   0.542424    37.897    < 2e-16 ***
hp          -0.18458    0.003359    -5.495    5.77e-06 ***
---
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.282 on 30 degrees of freedom
Multiple R-squared: 0.5016,    Adjusted R-squared: 0.485
F-statistic: 30.19 on 1 and 30 DF,  p-value: 5.766e-06

>
```

Although there are a number of quantitative metrics produced by the `summary(lm())` function, we are only concerned with the r-squared and p-value metrics at the bottom of the output.

From our linear regression model, the r-squared value is 0.50, which means that roughly 50% of the variablilty of our dependent variable (quarter-mile time predictions) is explained using this linear model. Compared to the Pearson correlation coefficient between quarter-mile race time and horsepower of -0.71, we can confirm that our r-squared value is approximately the square of our r-value. In a simple linear regression model, the higher the correlation is between two variables, the more that one variable can explain/predict the value of the other.

In addition, the p-value of our linear regression analysis is 5.77×10^{-6}, which is much smaller than our assumed significance level of 0.05%. Therefore, we can state that there is sufficient evidence to reject our null hypothesis, which means that the slope of our linear model is not zero.
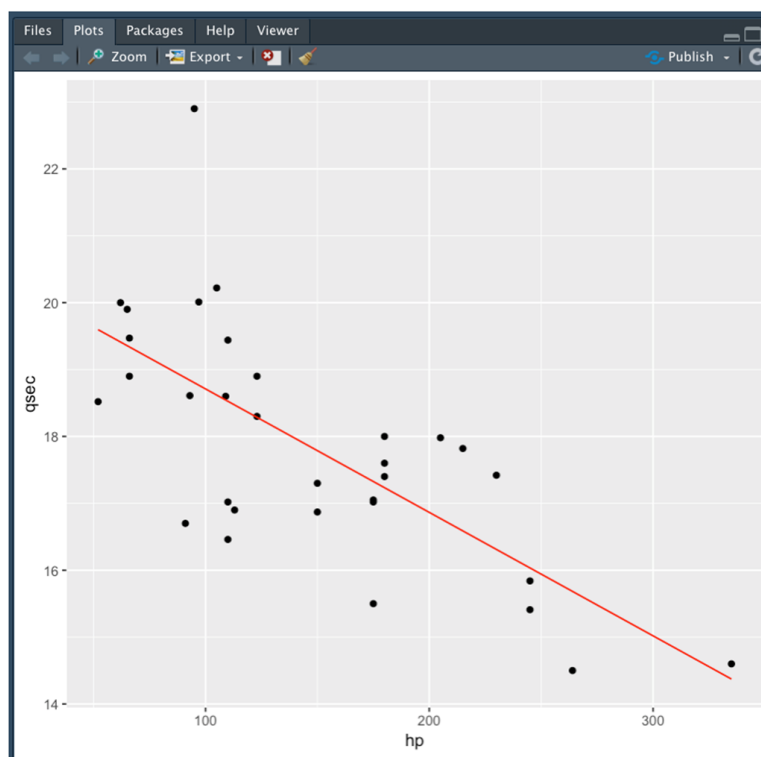
Once we have calculated our linear regression model, we can visualize the fitted line against our dataset using ggplot2.

First, we need to calculate the data points to use for our line plot using our `lm(qsec ~ hp,mtcars)` coefficients as follows:

```
> model <- lm(qsec ~ hp,mtcars) #create linear model
> yvals <- model$coefficients['hp']*mtcars$hp +
model$coefficients['(Intercept)'] #determine y-axis values from linear model
```

Once we have calculated our line plot data points, we can plot the linear model over our scatter plot:

```
> plt <- ggplot(mtcars,aes(x=hp,y=qsec)) #import dataset into ggplot2
> plt + geom_point() + geom_line(aes(y=yvals), color = "red") #plot scatter and linear model
```


Using our visualization in combination with our calculated p-value and r-squared value, we have determined that there is a significant relationship between horsepower and quarter-mile time.

Although the relationship between both variables is statistically significant, this linear model is not ideal. According to the calculated r-squared value, using only quarter-mile time to predict horsepower is roughly as accurate as guessing

using a coin toss. In other words, the variability we observed within our horsepower data must come from multiple sources of variance. To accurately predict future horsepower observations, we need to use a more robust model.