FIT2086 Studio 6
Linear Regression

Daniel F. Schmidt

September 1, 2017

Contents

1 Introduction

Studio 6 introduces you to simple and multiple linear regression, and asks you to use linear regression to build prediction models. During your Studio session, your demonstrator will go through the answers with you, both on the board and on the projector as appropriate. Any questions you do not complete during the session should be completed out of class before the next Studio. Complete solutions will be released on the Friday after your Studio.

2 Least Squares and Simple Linear Regression

In simple linear regression we are interested in predicting the (average) value of a target, say Y, using a predictor/explanatory variable x. Simple linear regression models the conditional mean of our target Y as a linear function of the predictor, i.e.,

$$\mathbb{E}\left[Y \mid x\right] = \beta_0 + \beta_1 x.$$

This says that the mean of y varies linearly as a function of x, with the amount of variation determined by the coefficient β_1. For every unit increase in x, the mean of Y increases by the amount β_1. The β_0 parameter is called the intercept – this is predicted mean value of Y when the predictor $x = 0$. We can also view the linear model in a more explicit probabilistic formulation; we can write

$$y = \beta_0 + \beta_1 x + \varepsilon \tag{1}$$

where ε is an unobserved, random variable. This says that the target is a linear function of the predictor x plus a random disturbance; this random disturbance could be due to measurement error, or it could be due to other unmeasured random fluctuations. By selecting a probability distribution for ε we can get an explicit probability model; the most common choice is to take $\varepsilon \sim N(0, \sigma^2)$, so that we model the disturbance as a zero-meaned normally distributed random variable. Then we can write the simple linear model as

$$Y \mid x \sim N(\beta_0 + \beta_1 x,\, \sigma^2).$$

Given observed targets $\mathbf{y} = (y_1, \ldots, y_n)$ and associated predictor values $\mathbf{x} = (x_1, \ldots, x_n)$, a standard way of estimating the coefficient β_1 and intercept β_0 is by minimising the residual sum-of-squares (RSS); for any β_0, β_1 we can define the residuals as

$$e_i = y_i - \beta_0 - \beta_1 x_i$$

which are essentially the errors in predicting y_i when using β_0 and β_1. The least-squares estimation method says find the β_0, β_1 that minimise

$$\text{RSS}(\beta_0, \beta_1) = \sum_{i=1}^{n} e_i^2$$

which is a measure of goodness-of-fit. For the simple linear regression, the values of the least-squares estimates are:

$$\hat{\beta}_0 = \frac{\left(\sum_{i=1}^{n} y_i\right)\left(\sum_{i=1}^{n} x_i^2\right) - \left(\sum_{i=1}^{n} y_i x_i\right)\left(\sum_{i=1}^{n} x_i\right)}{n\sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2} \tag{2}$$

$$\hat{\beta}_1 = \frac{\left(\sum_{i=1}^{n} y_i x_i\right) - \hat{\beta}_0\left(\sum_{i=1}^{n} x_i\right)}{\sum_{i=1}^{n} x_i^2} \tag{3}$$

We will begin by examining the equations for the least-squares estimates (2) and (3). Imagine we have some data $\mathbf{y}$, of height measured in meters, and also have a predictor $\mathbf{x}$, which is weight measured in kilograms. Let us call the estimates we obtain for this data $\hat{\beta}_0$ and $\hat{\beta}_1$.

1. Imagine we change our unit of measurement for our target $\mathbf{y}$ from meters to centimeters, i.e., we have new targets $y_i' = 100\, y_i$. What happens to the least-squares estimate of the intercept and coefficient? That is, how are the new estimates $\hat{\beta}_0'$ and $\hat{\beta}_1'$ related to the previous estimates $\hat{\beta}_0$ and $\hat{\beta}_1$ we obtained when y_i was measured in meters?

   A: More generally, let us imagine we scale our targets y_i by some scalar c to make $y_i' = cy_i$; then we see that

   $$\sum_{i=1}^{n} y_i' = c\left(\sum_{i=1}^{n} y_i\right) \text{ and } \sum_{i=1}^{n} y_i' x_i = c\left(\sum_{i=1}^{n} y_i x_i\right).$$

   Using these results in (2) we can see that

   $$\hat{\beta}_0' = \frac{c\left(\sum_{i=1}^{n} y_i\right)\left(\sum_{i=1}^{n} x_i^2\right) - c\left(\sum_{i=1}^{n} y_i x_i\right)\left(\sum_{i=1}^{n} x_i\right)}{n\sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2} = c\,\hat{\beta}_0$$

   and using this result in (3) we see that

   $$\hat{\beta}_1' = \frac{c\left(\sum_{i=1}^{n} y_i x_i\right) - c\hat{\beta}_0\left(\sum_{i=1}^{n} x_i\right)}{\sum_{i=1}^{n} x_i^2} = c\,\hat{\beta}_1$$

   So scaling our targets by c also scales our estimate of the intercept and coefficient by c. Therefore, in our example, converting our target from meters into centimeters scales our targets by $c = 100$, and corresponding scales our estimates by $c = 100$.

2. Imagine instead we change the unit of measurement of our predictor $\mathbf{x}$ from kilograms to grams, i.e., we have new predictors $x_i' = 1000\, x_i$. What happens to the least-squares estimate of the intercept and coefficient? That is, how are the new estimates $\hat{\beta}_0'$ and $\hat{\beta}_1'$ related to the previous estimates $\hat{\beta}_0$ and $\hat{\beta}_1$ we obtained when x_i was measured in meters?

   A: More generally, let us imagine we scale our preditor x_i by some scalar c to make $x_i' = cx_i$; then we see that

$$\sum_{i=1}^{n} x_i' = c\left(\sum_{i=1}^{n} x_i\right) \text{ and } \sum_{i=1}^{n} y_i x_i' = c\left(\sum_{i=1}^{n} y_i x_i\right) \text{ and } \sum_{i=1}^{n} (x_i')^2 = c^2\left(\sum_{i=1}^{n} x_i^2\right).$$

   Using these results in (2) we can see that

$$\hat{\beta}_0' = \frac{c^2\left(\sum_{i=1}^{n} y_i\right)\left(\sum_{i=1}^{n} x_i^2\right) - c^2\left(\sum_{i=1}^{n} y_i x_i\right)\left(\sum_{i=1}^{n} x_i\right)}{c^2 n \sum_{i=1}^{n} x_i^2 - c^2\left(\sum_{i=1}^{n} x_i\right)^2} = \hat{\beta}_0$$

   and using this result in (3) we see that

$$\hat{\beta}_1' = \frac{c\left(\sum_{i=1}^{n} y_i x_i\right) - c\hat{\beta}_0\left(\sum_{i=1}^{n} x_i\right)}{c^2 \sum_{i=1}^{n} x_i^2} = \frac{\hat{\beta}_1}{c}$$

   So scaling our predictors by c leaves the estimate of the intercept unchanged, and scale the coefficient estimate by $1/c$. Therefore, in our example, converting our predictor from kilograms to grams scales our preditors by $c = 1000$, and corresponding scales our estimates by $1/c = 1/1000$.

3. What does the behaviour of the least-squares estimates $\hat{\beta}_0$ and $\hat{\beta}_1$ when you change the units of measurement for either the targets y or predictors x imply about the resulting linear model?

   A: These results imply to us that the estimated model is invariant to changes in scales of our targets and predictors. If the target is re-scaled, then the coefficients re-scale by the same amount to compensate, so the resulting model makes the equivalent predictions in the new scale. For example, imagine our targets are measured in meters and our predictors are measured in kilograms, and for the data we have, we obtain least-squares estimates of $\hat{\beta}_0 = 1$ and $\hat{\beta}_1 = 0.2$. Then for if $x = 1kg$, the predicted value of our target would be $\hat{\beta}_0 + \hat{\beta}_1 \times 1 = 1 + 0.2 = 1.2m$.

   (1) If we convert our targets from meters to centimeters, we re-scale our targets by $c = 100$, and from the answers above we know that we rescale our LS estimates by $c = 100$, i.e., $\hat{\beta}_0' = c\hat{\beta}_0 = 100$ and $c\hat{\beta}_1 = \hat{\beta}_1 = 20$. Using this new model, for $x = 1kg$, we predict our target to be $\hat{\beta}_0' + \hat{\beta}_1' \times 1 = 100 + 20 = 120cm$, which is equivalent to $1.2m$.

   (2) Now instead imagine that we rescale our predictor from kilograms to grams. This is equivalent to scaling the predictor by $c = 1000$. From our answer above we know that this leaves our intercept untouched and rescales our coefficient $\hat{\beta}_1$ by $1/c$ to compensate for the increase in scale of our predictor; in this case this means that our LS estimate for the coefficient becomes $\hat{\beta}_1' = 0.2/1000 = 2 \times 10^{-4}$. Now, for $x = 1000g$ (equivalent to $1kg$) we predict our target to be

$\hat{\beta}_0 + \hat{\beta}_1' \times 1000 = 1 + 2 \times 10^{-4} \times 1000 = 1.2m$. So when we rescale our preditors, the LS estimates scale in an inverse proportional fashion to compensate for the increase in scale, which ensures predictions (which are the product of the coefficient and predictor) of the model are unchanged.

It is common in regression analysis to centre the predictors. This means that we replace our original predictors $(x_1, \ldots, x_n)$ with new predictors $(x_1', \ldots, x_n')$ where

$$x_i' = x_i - \bar{x}$$

and $\bar{x}$ is the sample mean of $\mathbf{x}$. The effect of this centering is that the predictor now has a mean of zero.

4. What happens to the formula for the LS estimate of the intercept (2) if we centre our predictor $\mathbf{x}$? What is the new formula equivalent to?

   A: Given that $\mathbf{x}'$ has a mean of zero we also know that the sum

   $$\sum_{i=1}^{n} x_i' = 0. \tag{4}$$

   Using this fact in (2) we see the LS formula reduces to

   $$\hat{\beta}_0' = \frac{\left(\sum_{i=1}^{n} y_i\right)\left(\sum_{i=1}^{n} (x_i')^2\right)}{n\sum_{i=1}^{n} (x_i')^2} = \frac{\sum_{i=1}^{n} y_i}{n}.$$

   This is equivalent to the sample mean of the targets $\mathbf{y}$. So, when we centre our predictor $\mathbf{x}$ our LS estimate for the intercept reduces to the sample mean of the targets.

5. What happens to the formula for the LS estimate of the regression coefficient (3) if we centre our predictor $\mathbf{x}$?

   A: Again, using the property (4) in the LS estimate of the regression coefficient (3) we see that the formula reduces to

   $$\hat{\beta}_1' = \frac{\left(\sum_{i=1}^{n} y_i x_i'\right)}{\sum_{i=1}^{n} (x_i')^2}.$$

   This says that when we centre our predictors, the LS estimate for β_1: (i) no longer depends on the estimate of the intercept $\hat{\beta}_0$ as in (3), and (ii) is equal to the inner product of the predictor and the target, divided by the sum-of-squares of the predictor.

3 Simple Linear Regression

In this question we will use R to perform a simple linear regression on a toy dataset. This will teach you how to use the R regression commands, and also give you a little insight into how certain data values can cause problems for least-squares. This question will also demonstrate the basics of using the `lm()` function and the predict function to perform least-squares regression.

A: See `studio6_solns.R` for answers to these questions.

1. Load the data file `toydata.csv` into R (store it in the dataframe `df`), and plot the variable `df$y` against the variable `df$X`.

2. We would like to fit a linear model to this data, using `X` as our predictor and `y` as our target, i.e., we would like predict `y` using `X`. To do this we can use the `lm()` function, which fits a linear model using least squares. Use the command

   $$\texttt{lm}(\texttt{y} \sim \texttt{X}, \texttt{data} = \texttt{df})$$

   to perform the fit. The "`y` $\sim$ `X`" says to model `y` using `X` as a predictor. What do the estimated values of the intercept and regression coefficient tell you about the relationship between `y` and `X`?

3. We can actually store the results of the fitting process into an object – this lets us get access to a lot more information. To do this, use

   $$\texttt{fit} = \texttt{lm}(\texttt{y} \sim \texttt{X}, \texttt{data} = \texttt{df})$$

   which stores the fitted model in `fit`. To view the results of the fitting procedure, and a number of statistics associated with the fitted model, use the `summary()` function on the object `fit`. Does the p-value for the regression coefficient suggest that it is a significantly associated with our target `y`?

4. Another advantage of storing the fitted model in an object is that we can use it to make predictions, either on new data, or on the data we have fitted on. We can use this to see how well our model fitted the data. First, write down the fitted linear equation for predicting `y` in terms of `X` as estimated above. You could use the coefficients from the fitted linear model `fit` to make predictions, i.e.,

   $$\texttt{yhat} = \texttt{fit\$coefficients}[[1]] + \texttt{fit\$coefficients}[[2]] * \texttt{df\$X}$$

   The `coefficients` variable in the `fit` object contains the coefficients, and by convention in R, the intercept is always the first element of the list. R also provides a function make predictions using our model; this is called the `predict()` function, and we can call it using

   $$\texttt{yhat} = \texttt{predict}(\texttt{fit}, \texttt{df})$$

   The first argument is a linear model to use to predict; the second argument is a dataframe from which to get the values of the predictors. This can be a dataframe containing new data (different from the data we used to predict) but it must have the same predictors (i.e., same column names) as the data we used to fit the model in the first place. Compare the predictions produced by the two methods – they should be the same.

   To see how well our model fitted the data, plot the predictions against `df$X` using the `lines()` function. How good does the line fit the data?

The data point when $x = 10$ is quite far away from the rest of the datapoints, and it seems to have "dragged" our fitted line away from passing through the bulk of the other 9 data points. A data point that is quite far away from the other data points is called an outlier. One weakness of the least-squares procedure is that it is based on minimising the sum of squared errors, which is very sensitive to large deviations; so outliers can have a very large effect on the fitted line, and the LS procedure will "overadjust" the line to fit the outliers at the expense of the rest of the data which is more closely clustered together.

5. Let us see how our fitted line changes if we do not use this potential outlying point. To do this we can fit our linear model using all but the last datapoint; the `lm()` function allows for this through the use of the `subset` option; e.g.,

$$\texttt{fit2} = \texttt{lm}(\texttt{y} \sim \texttt{X}, \texttt{data} = \texttt{df}, \texttt{subset} = 1:9)$$

   Use `summary()` to view the statistics for this new fit. How much has removing this data point changed the estimates, the p-value for the regression coefficient and the R^2 value (the goodness of fit, see Lecture 6, Slides 38–39)? Does this support the belief that this data point may be an "outlier"?

6. Compute predictions for all 10 datapoints in your dataframe `df` using the model fitted without the outlying datapoint, and plot them using the `lines()` function. How do the two fitted lines differ?

When making predictions using a linear model, we should bear in mind that our estimates are just that – estimates from data. They are not equal to the population parameters, and as we know, if we saw a new sample of data from our population, the resulting estimates would vary by some amount. If the estimates vary with a new sample, then so do the predictions, as they are based on the coefficients. The R `predict()` function provides options to produce the intervals of predictions to help us get an idea of how accurate our predictions are. These are like confidence intervals for parameters, but are now instead intervals on the predicted values of the target `y` for different values of the predictor `X`.

7. Produce an interval for the predicted mean

$$\mathbb{E}\left[y \,|\, x\right] = \beta_0 + \beta_1 x$$

   of the target `y` using `predict()` with the `interval="confidence"` option. This gives you a plausible range of estimates of the mean value of our target `y`, given `x`. Using this option `predict()` returns a dataframe with the columns `fit` (the "best guess" at the mean of our target), `lwr` and `upr` (the interval of plausible guesses for the mean of the target).

   Scatterplot the datapoints `y` against `X`, and then plot the best guess of the mean of `y` given `x` (`fit`) as well as the upper and lower ends of the confidence interval.

8. The `predict()` function also lets you get a so-called prediction interval. Go back and look at the probabilistic interpretation of the linear model given by equation (1). The above confidence interval gives us a range of plausible values for the mean value of `y` if we resampled from the population. The prediction interval gives you an interval of plausible values that our target `y` could take if we drew a new sample from the population. To find a prediction interval, use `predict()` with the `interval="prediction"` option. Once again the `fit` column of our predictions is the best guess of our average value of `y` in the population, given `x`.

   Scatterplot the datapoints `y` against `X`, and then plot the best guess of the mean of `y` given `x` (`fit`) as well as the upper and lower ends of this prediction interval.

9. How do the two intervals compare? Why do you think the prediction interval is wider?

4 Multiple linear regression

In this question we will be using the R function `lm()` to perform multiple linear regression and build a prediction model for a real dataset. The dataset we will be looking at is related to red and white variants of the Portuguese "Vinho Verde" wine. It consists of 12 variables in total; $p = 11$ explanatory variables/predictors:

1. Fixed acidity (`fixed.acidity`)
2. Volatile acidity (`volatile.acidity`)
3. Citric acid (`citric.acid`)
4. Residual sugar (`residual.sugar`)
5. Chlorides (`chlorides`)
6. Free sulfur dioxide (`free.sulfur.dioxide`)
7. Total sulfur dioxide (`total.sulfur.dioxide`)
8. Density (`density`)
9. pH level(`pH`)
10. Sulphates (`sulphates`)
11. Alcohol (`alcohol`)

and one target, a numerical quality score (`quality`) ranging from 0 (poor) to 10 (excellent), that was assessed by wine tasters. It is clearly of economic interest to wine makers to be able to identify chemical characteristics of wines that predict poor quality wine.

1. Load the `wine_train.csv` file into the dataframe `wine`. Use `summary()` to examine the dataset and get an idea of what the variables look like. You can also boxplot/histogram the variables to see how they look.

   A: See `studio6_solns.R`. All the predictors are continuous variables and seem to be (normal-ish) distributed, though some exhibit some non-symmetric distributions. The target `quality` is the most interesting case. It is a discrete variable that takes on a finite number (11, 0 through to 10) of values. These are clearly not exactly normally distributed; however, if the variable is discrete and the number of values is larger than 5 or 6 we can often still use least-squares and normal linear models to good effect.

2. Fit a linear model to the data using all the predictors to predict `quality` using

   $$\texttt{fit} = \texttt{lm}(\texttt{quality} \sim ., \texttt{wine})$$

   The "`quality ~ .`" is shorthand for using all variables other than `quality` to predict `quality`. Use `summary()` to examine the fitted model. Which variables do you think are potentially associated with wine `quality` based on their p-values?

   A: See `studio6_solns.R`. Looking at the p-values for each of the variables, we see that the three predictors `volatile.acidity`, `residual.sugar` and `free.sulfur.dioxide` seem to be

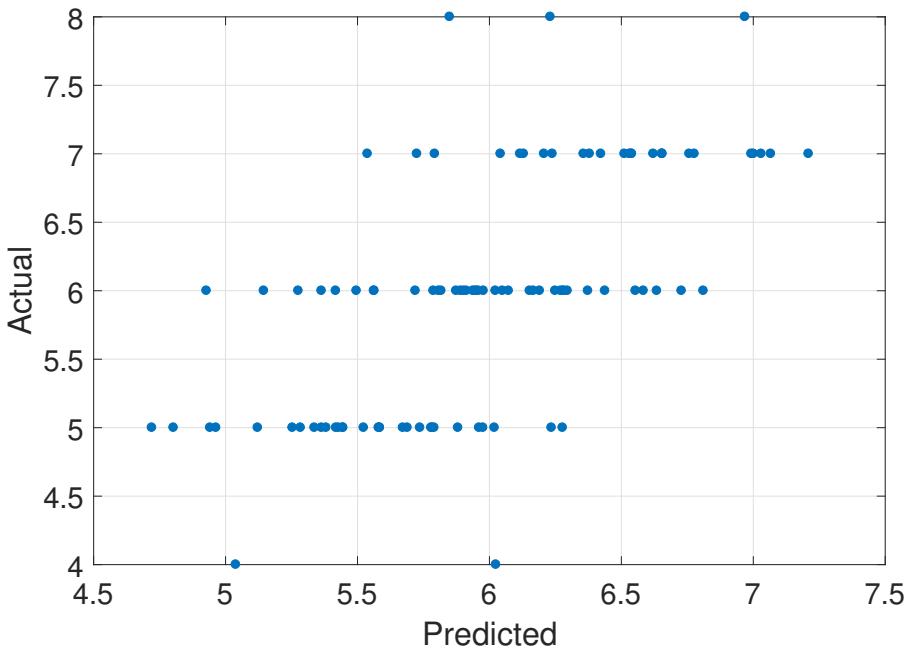

Figure 1: Plot of predicted quality of wine (x-axis) against the actual quality of wine (y-axis), for each of the observations in our dataset.

borderline ($p < 0.1$) associated and are probably our best guesses, based on p-values, of which variables might be associated with `quality`. Remember, in this setting, the p-value is the evidence against the null hypothesis that the coefficient $\beta_j = 0$, i.e., the coefficient has a value of zero at the population level (that is, it is unassociated with the target). Remember, a $\beta_j = 0$ means it is unassociated because 0 times any number will still be zero – so now matter how big the value of the predictor x_j is, its contribution to our predicted value of y will still be zero. A small p-value will therefore be suggestive that the data is at odds with the null hypothesis of no assocation, and we should potentially consider the predictor as being associated with the target.

Recall that a p-value of 0.1 means that the chance of seeing an association as strong as the one we have observed, just by chance, even if there was no association at the population level, is 1 in 10 – which is neither highly likely, nor is it particuarly unlikely – hence these are "borderline" associated as the data is not strongly at odds with our null hypothesis of no association.

In practice, if the sample size is large we might expect much smaller p-values if the variables are associated. For smaller sample sizes the p-values will often be a bit greater, unless the effect is very strong.

3. Because we have more than one predictor we cannot plot our predictions against all the predictors. Instead, produce predictions for wine quality using our data `wine` and scatter plot these predictions against the actual values of `quality` that we estimated our linear model from. What would we expect this plot to look like if our model fitted the data perfectly?

A: See `studio6_solns.R`. Looking at this plot (see Figure 1, created in MATLAB – but a similar plot is easily created in R), we can interpret it in the following way. It plots the predicted quality

of a wine on the x-axis against the actual quality of that particular wine on the y-axis, for all the wines in our training data sample. So for example, from our plot, we can see that when our model predicts `quality` to be around 6, the actual `quality` value are concentrated around 5-6, and when our model predicts the `quality` to be around 7, the actual quality scores are larger on average, and approximately around 7. Of course the predictions are not perfect. If they were, we would expect our predicted value to exactly equal to the actual values of quality in the data, which would result in a diagonal line. The more "diagonal" the line the better the fit. In this example there is clearly some concordance between the predictions and the actual values and the overall trend is somewhat diagonal – so our model is doing a decent job of predicting wine quality in our data.

4. We can see how well our model predicts by using new unseen data from the same population. The file `wine_test.csv` contains $4,798$ new data points that we can test our model on. Produce predictions for this new dataframe using the model we fitted above, and calculate the mean squared-prediction error (MSPE) on this new data using

$$\texttt{mean}((\texttt{wine\_test\$quality} - \texttt{yhat})\texttt{\^{}2})$$

What does this error measure tell you?

A: See `studio6_solns.R`. This error tells us how good our model is at predicting the quality of wines from new, unseen data using the 11 predictors associated with the wines. The smaller the error the better the model fit, with an error of zero being a perfect prediction. The value of error in this case is 0.619, which is the *average squared error* between our predictions and the new data. Square-rooting this quantity gives an error of 0.787, which can be interpreted in the following way: on average, for future data our predicted quality score for a wine, given the 11 chemical measurements our model was trained on, is around 0.787 different (in either direction) from the actual quality of the wine.

5. Our model has included all 11 predictors; perhaps some of them are not associated and we can improve the predictive performance of our model by excluding them, or at least obtain a similar predictive performance with a simpler model. We can use the `step()` function to perform stepwise variable selection (see Lecture 6). To do this, we use

$$\texttt{fit.sw} = \texttt{step}(\texttt{fit})$$

which takes a previous full linear model (`fit`) fitted using `lm()`, and uses the information criteria approach and forward stepwise selection to try and figure out which variables are important. It does this by iteratively enlarging the model until it finds the one with the smallest information criterion score. Do this, and then examine the model `fit.sw` using `summary()`. What variables have been removed? Also calculate the MSPE for this new model on our testing data. How does this compare to our full model?

A: See `studio6_solns.R`. The stepwise procedure (which tries to remove "unimportant" variables by minimising the Akaike information criterion score, which trades off how well our model fits our data against how complex the model is) has removed 6 of our 11 predictors, leaving us with `volatile.acidity`, `residual.sugar`, `free.sulfur.dioxide` (which the p-values we examined above suggested were potentially important) as well as `density` and `alcohol`, which the p-values did not suggest were important (both were > 0.15). We can note, from the `summary()` of the new

model, that the p-values of the remaining variables have all gone down, particularly for alcohol and density.

When we compute the mean squared-prediction error (MSPE) on our test data, we see that the MSPE is 0.611 which is smaller than the error of 0.619 obtained using the full model. So the smaller model actually performs better for prediction than using all the variables. This suggests that at least some of the 6 variables we have removed are not actually associated with quality, and any predictions made using them would just be adding "noise" to our predictions.

6. How do the R^2 values of the full regression model and the one found by the stepwise procedure above compare?

   A: See `studio6_solns.R`. The R^2 for the full model is 0.4, while for the smaller model selected by the stepwise procedure it is 0.389. As has been discussed in Lecture 6, the more variables you include in a model, the better the fit to the data you are training the model on will be – and therefore, the better the R^2. However, if some of those predictors are not really associated with the target, or are so weakly associated that it is difficult to properly estimate their coefficients well, we can improve prediction error on new data by removing them – as has been done by our stepwise procedure.

7. Write down the multiple linear regression equation found by the stepwise procedure above. What happens to `quality` as `alchohol` increases? What happens to `quality` as `volatile.acidity` increases?

   A: See `studio6_solns.R`. The multiple linear regression equation found by the stepwise procedure is:

$$\begin{aligned}\mathbb{E}\left[\texttt{quality}\right] &= 127.56 - 1.347 \times \texttt{volatile.acidity} + 0.0857 \times \texttt{residual.sugar} \\ &\quad +0.0115 \times \texttt{free.sulfur.dioxide} - 126.1 \times \texttt{density} + 0.298 \times \texttt{alcohol}\end{aligned}$$

   From this, we can see that our model says that as `alcohol` increases, the predicted `quality` increases. For every unit increase in `alcohol` content, the `quality` score goes up by approximately 0.3. So increased alcohol content seems to make a wine taste better.

   According to our model, as `volatile.acidity` increases, the predicted `quality` decreases. For every unit increase in `volatile.acidity`, the `quality` score decreases by approximately 1.34. So, increased volatile acidity seems to make a wine less palatable.

   A manufacturer could then make use this model to try and determine which characteristics of the wine to reduce/increase to increase perceived taste, and therefore, increase sales.

8. By default the `step()` function uses the Akaike information criterion (AIC). We can also use the Bayesian information criterion (BIC) (see Lecture 6) by using the following code

$$\texttt{fit.sw.bic} = \texttt{step}(\texttt{fit}, \texttt{k} = \texttt{log}(\texttt{length}(\texttt{wine\$quality})))$$

   (AIC is obtained by setting `k = 2`, which is the default. Note that `lm()` multiplies the information criteria presented in Lecture 6 by 2, which is sometimes done by packages). How does the model fitted by BIC compare to the one fitted using AIC? Compute the MSPE for this new

model. Why do you think the MSPE is so different?

A: See `studio6_solns.R`. The BIC method is more conservative than the AIC method used in 4.5, and has only selected 3 predictors: `residual.sugar`, `free.sulfur.dioxide` and `alcohol`. The prediction error on new data is actually worse in this case (0.652 as compared to 0.619 for the full model and 0.611 for the model selected using AIC in 4.5). It seems the two predictors BIC omitted but AIC retained (`volatile.acidity` and `density`) are actually important for accurately predicting quality, and leaving them out has hurt our predictions.

This does not mean BIC selected models are always worse than AIC selected models – it just means that it can be actually quite tricky to select a good model using just the data. This subject gives you some tools to do so, but a large part of good data modeling is an art.

9. Fit a linear model to the `wine_test.csv` data. This is much larger than our small training dataset of 100 datapoints. How do the estimates compare between the models fitted on the training and testing data? Which variables do not seem to be associated with `quality` in this very large dataset?

   A: See `studio6_solns.R`. From the results of fitting on a LOT of data (4798 observations, as opposed to just 100 we used in our original training dataset `wine_train.csv`) we see that all but three of the predictors are found to have very small p-values (mostly in the order of 10^{-10} or smaller!). The chance of seeing associations this strong just by chance are then absolutely tiny (1 in trillions). Three of the variables have much larger p-values and are probably not associated (`citric.acid`, `chlorides`, `total.sulfur.dioxide`).

   Interestingly, if we look at the model selected by our stepwise procedure in Q4.5, we see that those three variables are among the 6 predictors that were dropped from our model – which potentially explains why our smaller model predicted better onto this new data than the full model using all the predictors did. The associations between these three variables and the target that we estimated in the full model were likely to be incorrect and adversely effect our predictions.

10. The last question is more freeform. Play around with `lm()` and see if you can improve the predictive performance of your model by considering possible different transformations of the data (logarithmic, polynomial, anything else you can think of).