# FIT2086 Studio 6
# Linear Regression

Daniel F. Schmidt

August 28, 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 $\beta_1$. For every unit increase in $x$, the mean of $Y$ increases by the amount $\beta_1$. The $\beta_0$ 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 $\varepsilon$ 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 $\varepsilon$ 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 $\beta_1$ and intercept $\beta_0$ is by minimising the residual sum-of-squares (RSS); for any $\beta_0$, $\beta_1$ 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 $\beta_0$ and $\beta_1$. The least-squares estimation method says find the $\beta_0$, $\beta_1$ 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?

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?

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?

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?

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

# 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.

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.

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?

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?

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?

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?

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

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?

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?

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?

10. The last question is more freeform. For example, you could (i) 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); or (ii) check some of the statements in the lectures; for example, for your fitted model calculate the residuals and confirm that they have a mean of zero and are uncorrelated with all the predictors, etc.