In this document, we'll focus on using `R` to generate simple linear regression models, using the `lm` function. We will see how `lm` simplifies the process of model fitting and parameter estimation. Finally, we will assess the quality of the regression and use residual diagnostic plots to assess the validity of the regression assumptions.

Part 1: The `faithful` Data Set

In this section, we will perform regression-related EDA, fit a simple linear regression (SLR) model, and test the assumptions of the model. We will do all this using the `faithful` data set we last looked at in Workshop 4.

1.1 Elementary Data Analysis/Correlation

It is good practice to take a quick visual look at the data you will be using in your models.

1. Are there outliers in either column of the data? Generate boxplots to check.

2. To save typing later, let's extract the columns into two variables. Create a vector `w` that contains the vector of waiting times, and a second vector `d` that contains the durations of eruptions.

3. Plot the waiting times (y-axis) against the durations (x-axis). Is there evidence of a linear relationship? Use the `cor` function to assess the strength of that linear relationship.

4. Is the `cor` function what we want to use here? Check if the values in each of the two variables are normally distributed.

5. Use the `method=''spearman''` argument inside the `cor` function to generate the Spearman's rank correlation coefficient.

6. Find the Kendall's tau correlation coefficient.

   (Note that here we are simply checking there is any point to applying simple linear regression at all. Next week, when we consider multiple linear regressions, we tend to look at pairs of predictor variables using a scatter diagram, this time hoping we **don't** see much in the way of a linear relationship.)

1.2 The Linear Regression Model

The model we're interested in fitting to these data is a simple linear relationship between the waiting times, `w`, and the eruption durations, `d`. The model equation is

$$w = \beta_0 + \beta_1 d + \epsilon$$

The `lm` function in `R` computes the least-squares linear regression for us, and returns an object which summarises all aspects of the regression fit.

Suppose our response variable is Y, and we have a predictor variable X, with observations stored in `R` as the vectors `y` and `x` respectively. To fit `y` as a linear function of `x`, we use the following command:

```
model <- lm(y ~ x)
```

Alternatively, if we have a data frame called `dataset` with variables in columns named `''a''` and `''b''` then we could fit the linear regression of a on b[1] without having to extract the columns into vectors. We do this by specifying a data argument:

```
model <- lm(a ~ b, data=dataset)
```

The tilde symbol (the "∼" symbol) in this expression should be read as "is modelled as". Note that `R` always automatically include a y-intercept[2], so we only need to specify the X and Y variables.

We can inspect the fitted regression object returned from `lm` to get a simple summary of the estimated model parameters. To do this, simply enter the name of your model into `R`.

7 Use `lm` to fit waiting times `w` as a linear function of eruption durations d. Save the result of the regression function to `model`.

8 Note the values of the coefficients (the y-intercept and gradient given). What meaning can be taken from these values, given the meaning of the predictor and response variables?

`R` also has a number of functions that, when applied to the results of a linear regression, will return key quantities such as residuals and fitted values:

`coef(model)` and `coefficicents(model)` will return the estimated model coefficients as a vector (b_0, b_1).

`fitted(model)` and `fitted.values(model)` will return the vector of fitted values, $\hat{y}_i = b_0 + b_1 x_i$.

`resid(model)` and `residuals(model)` will return the vector of residual values, $e_i = y_i - \hat{y}_i$.

[1] Note the terminology here: "a on b". This means using a as the response, and b as the predictor. If a was being used as the predictor, we would say we were "fitting b on a".

[2] Though you can tell `R` not to do so, if for some reason that's important.

9 Use the `coef` function to extract the coefficients of the fitted linear regression model as a vector.

10 Extract the vector of residuals from model, and use this to compute the sum of squares of the residuals as `lsq.Q`.

We can easily extract and inspect the coefficients and residuals of the linear model, but to obtain a more complete statistical summary of the model we use the `summary` function: `summary(model)`
There is a lot of information in this output, but the key quantities to focus on today are:

- Residuals: simple summary statistics of the residuals from the regression.
- Coefficients: a table of information about the fitted coefficients. Its columns are: The label of the fitted coefficient: The first will usually be (Intercept), and subsequent rows will be named after the other predictor variables in the model.
- The Estimate column gives the least squares estimates of the coefficients.
- Residual standard error: This gives s_e, the square root of the unbiased estimate of the residual variance.
- Multiple R-Squared: This is the R^2 value defined in lectures as the squared correlation coefficient and is a measure of goodness of fit.

We will consider the remaining values in next week's workshop.
We can also use the summary to access the individual elements of the regression summary output. If we save the results of a call to the summary function of a `lm` object as summ:

`summ$residuals` extracts the regression residuals as `resid` above.
`summ$coefficients` returns the $p \times 4$ coefficient summary table with columns for the estimated coefficient, its standard error, t-statistic and corresponding (two-sided) p-value (again, we'll take a closer look at this next week).
`summ$sigma` extracts the regression standard error.
`R`'s estimate for the standard deviation of the errors in the model.
`summ$r.squared` returns the coefficient of determination R^2.

11 Apply the summary function to model to produce the regression summary for our example.

12 Use the summary to find the coefficient estimates, the residual standard error, and the coefficient of determination R^2.

13 Extract the value of R^2 from the `summary` output as a new variable. What does the value of R^2 here tell us about the fitted linear model?

Part 1.3 Checking Assumptions

In this final section, we will use the `plot` function to check whether the four assumptions for a simple linear regression model hold for the model we have generated. We will also consider whether there are any points in the model which have high influence.

14 Enter your model into the `plot` function. Remember that the function will cycle through four plots, and that you have to click on the plot window (or press enter) each time to get the next one.

15 Evaluate assumptions one, two and four using the first three plots.

16 Identify which points have high leverage using the fourth plot. Try removing these points from the data and recalculating the model. Does this make any difference to a) the performance of the model, b) the extent to which you believe the model assumptions hold?

17 Evaluate assumption three by calculating the autocorrelation of the residuals. Does our assumption of independence hold?

Once you have finished Part 1.3, you should feel free to spend the remainder of Workshop 5 working on your group mini-project.