# Research Methods and Statistics with R 3

# Week 8 – Multiple Linear Regression

## Introduction to the session

In this session, you will practice using multiple regression in R. As you have seen in the lecture, we can run regression with a single predictor (i.e., simple linear regression). However, it is far more common in psychology and neuroscience to test more complicated models. In this session, you can select one of a few different options to practice building and evaluating more complicated models.

## Learning Outcomes

By the end of this session, you will be able:

1. To apply multiple regression using R, including dealing with assumption violations
2. To interpret and summarise multiple regression outputs

## Pre-workshop Activities

1. Review the lecture material and readings prior to the session, particularly the material regarding assumptions of regression (Week 7).

## The Setup

For this session you have a choice:

1. **Create your own model to predict property prices using the London Housing Dataset.** Thelecture model explained 40% of the variance (adjusted R2 = .4). Can you do better?
2. We know from the lecture material that the lecture model violated several of the assumptions of multiple regression. We also discussed several possible ways to address these violations. **Can you find a solution to one or more of the violations?**
3. The Decade of Dance Dataset contains a number of variables about songs across the decade. **Can you generate a model that will predict the popularity of a given song with an adjusted R-squared above 50%?**

Whether you use the London Housing Dataset or the Decade of Dance dataset, the initial steps to generating a model will be largely the same. Below, we go through a suggested workflow, focussing on the London Housing Dataset.

## Procedure (based on the London Housing Dataset)

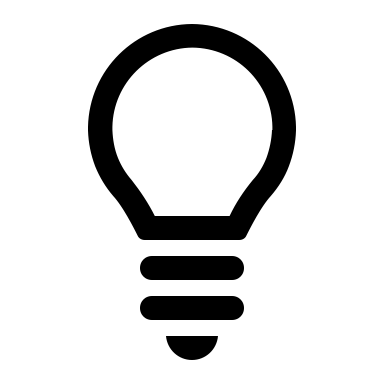
1. Before you proceed to the exploration and analysis of your data, make sure that you are clear about your **research question** and **null/alternative** hypotheses*.* Start a new script applying the heading and commenting practices you have developed in RM1/2 last year and include the research question and hypotheses as comments.
2. Download the appropriate dataset (e.g., **London.csv**) from KEATS and load it into your R environment. The data will be available as a \*.csv, so **read\_csv()** or **rio::import()** are both appropriate functions.

All the variables are self-explanatory, but feel free to rename them as you wish after loading the data. For the London.csv dataset in particular, the variable names currently includes spaces, which means R will need quotation marks around them whenever you call them (which is a pain). To deal with this in an efficient manner, you might want to look at the function **janitor::clean\_names()**.

1. Before going any further, you may want to prepare the dataset as done in the lecture. In the lecture, the dataset was restricted to properties with 2 to 5 bedrooms, priced at no more than £10M, and of the following types only: House, Flat / Apartment, and New development.   
     
   To filter the dataset, you could use base R as in the lecture or build a pipeline with *tidyverse*, particularly with **dplyr::filter()**. In that pipeline, **dplyr::mutate()** will allow you to convert character fields into factors (see below to do it for multiple variables at once) and transform the price to be in thousands of Pounds. You could also use **dplyr::select()** to drop the variables that we are not planning to include in our model.

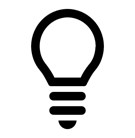
my\_data %>%

dplyr::mutate(across(where(is.character), as.factor))

1. For regression analyses, we have five assumptions to check, including two that we can check prior to computing the regression model. It is also a good idea at this stage to inspect the distribution of the outcome variable to see if extreme values might have an undue influence.
   1. To inspect the distribution, a simple histogram will do the job. Functions such as **hist()** or **ggplot2::geom\_histogram()** will be of use here. If you want to compare the distribution for each level of the categorical predictor(s) **ggplot2::geom\_density()**, using a **fill** argument to pass the categorical predictor in the aesthetics, is another option. Based on the output, you might decide to limit the price range further than was done in the lecture.
   2. The first assumption that can be checked prior to computing the model is the linearity of the relationship between the predictor variables and the outcome: this can be checked by visualizing these relationships in scatterplots. Remember this is only relevant for continuous predictors. You can do this with **ggplot2::geom\_point()**, or for a shorter option **ggpubr::ggscatter()**. Don’t forget to add a fitted line with **ggplot2::geom\_smooth(method = “lm”)**. Remember that you will need to produce such scatterplots for each continuous variable, so for publication purposes, you might want to present them as panels (refer back to the lecture to see one way to do this, noting that the **ggpubr::ggarrange()** function will only work with plots produced with *ggplot* based methods).
   3. The second assumption that can be checked PARTLY prior to running the model is the absence of multicollinearity. Although you will need to compute the model to obtain the VIF, you can still obtain the correlation matrix and explore where potential issues may arise. If you did not drop the unused variables earlier (step 5), now is the time! Once you have created a dataframe with only the predictors, make sure to transform the categorical predictor(s) into a numeric variable. Then create a correlation matrix with **stats::cor()** and use **corrplot::corrplot\_mixed()** to visualize the correlations between the predictors.
2. Good news! The **stats::lm()** function will cope on its own with missing values, excluding the observations pairwise (i.e., excluding fully each observation with even one missing value on one of the variables of interest). You should still inspect the data to determine the amount of missing values. Here you want to know both how many observations are complete — which you can obtain using **sum(stats::complete.cases(my\_df)) —** and which variables have data missing. For the latter, you can explore this with similar methods as for ANOVAs (i.e., using **summary()** or **skimr::skim()**.
3. While we are not interested in cell means here, it is generally useful to obtain the descriptive statistics for the predictors and outcomes. The good news is that **skimr::skim()**, just above, would have given you these, including frequencies for each level of the categorical predictor(s)!
4. The next step is to compute our model so we can test the remaining assumptions. This is similar to the between-subject ANOVA, but using **stats::lm()** to insert the formula: **outcome ~ predictor1 + predictor2 + … + predictorn**.

**Don’t forget to tell R what dataset to use, with the argument `data = my\_df` and to save your model to `my\_lm\_object` or a similar object name!**

1. As mentioned above, multiple regression has a number of assumptions, some that you can check before computing the model, and some that you need to check after. These include:
   1. Linearity of the relationships, through visual inspection of scatterplots (accomplished above);
   2. Multicollinearity, using for example **stats::cor()** and **corrplot::corrplot\_mixed()**, as well as **car::vif(my\_lm\_object)**
   3. Normality of the residuals, using **plot(my\_lm\_object, which = 2)** and **stats::shapiro.test(my\_lm\_object$residuals)**;
   4. Homogeneity of variance with **plot(my\_lm\_object, which = 1)** and **car::ncvTest(my\_lm\_object)**;
   5. Alternatively, **performance::check\_model()** will allow you to check all assumptions in one go.

**Reminder: the lm model object used to test assumptions should be the final model.**

1. You watched the lecture, so no spoiler here, but the assumptions are not met. You might have already tried to reduce further the dataset above after looking at the distribution(s). If you have done so, you will have observed that even reducing to properties up to £2.5M is not effective. *Maybe you can try a further reduction*?

Consult the lecture material for more options to deal with that kind of problem: we will explore many of these only later in the module, but in the meantime, you could try a log transformation with **log()**. You could also combine approaches (reduce on one or more feature(s) and/or log transform).

1. For reporting purposes, you will need to also obtain the effect size(s) of your predictor(s) of interest. This can be done with **rsq::rsq.partial()**.
2. Although it is unlikely that the data now meet all the assumptions, write up the results as you would for a research paper. Don’t forget to call **stats::summary.lm(my\_lm\_object)** to obtain the information you need for this

## Coding Challenge #1

Compare the outputs of a regression model including only the categorical predictor (property type) with an ANOVA output using the same predictor as variable (between-subject).

You could also run such a model with only two levels of the categorical predictor (you could drop New development for example), and compare the output to a *t*-test.

## Coding Challenge #2 (Matrix Algebra)

Create a 2x2 matrix and calculate its inverse of a 2x2 matrix. Then use R or MATLAB to calculate the inverse of large matrices.