Lab 2.2 - Working with regressions

Professor MacDonald

2025-04-02

Regression and `dplyr`

Running regressions

The non-`tidyverse` method of doing a regression (aka the base R way) is the example seen online in the following code:

```
reg=lm(Y~X1+X2+X3,data=dat)
summary(reg)
```

In this example, `Y` is the response variable and the `Xn` variables are the predictor variables.

The `summary()` command then provides the regression results, but it can be hard to work with to print or manipulate your results.

In `dplyr`, there are ways to more helpfully produce your regression results. For this, we will need the `broom` library. You first need to create your regression the standard way as above.

We can then tidy the results with the following command from the `broom` library:

```
mtReg <- lm(mpg ~ hp, data=mtcars)

tidy(mtReg)
```

This returns the regression results as a data frame.

We can also view some of the regression diagnostics (don't worry about any of the terms yet besides the r^2)

```
glance(mtReg)
```

Create a regression using the `mtcars` dataset and a different predictor variable. Tidy the results and produce the model fit statistics, then interpret your results.

1. Based on simply the results, how good of a fit do you think your model is?
2. For a one unit change in x, how much change is there in y? Do you estimate that to be a lot or a little?
3. How can you interpret the intercept? Is the intercept meaningful?

Getting the residuals

To get the residuals from a dataset, we need to use the `augment()` command from the `broom` library.

```
mt.cars.resids <- augment(mtReg, mtcars)
```

Notice that when you inspect this new, augmented dataset, it has a number of new data columns. The ones we are most interested in is the `.fitted` and the `.resid`.

We can use the `.resid` column to make a histogram of the residuals.

Make a histogram of your residuals

1. What are we hoping to see in a histogram of the residuals?
2. What do you observe in your residual histogram? Are there any problems?

We can also make a plot of the residuals vs. fitted values. This tells that for any given predicted value of y ($\hat{y}$) how much we 'missed' the prediction by. If there are systematic patterns in this graph, that indicates some possible non-linearity, outlier, or plot thickens condition in the relationship between your predictor variable(s) and response variable.

Make a scatterplot of your residuals vs. predicted (predicted on x axis). Draw a horizontal line at the 0 value for the y axis.

1. How does your plot look? Do you estimate that there are any problems?
2. Interpret your overall model fit based on the residual plots

Nicely printing your regression results

You can use plain old `kable()` to make your regression table from the results of the `tidy()` command but we can do better than this simple table. There are many libraries that print out pretty version of a standard regression table. I particularly like the one called `modelsummary()` (details can be found here).

You can simply use it as follows:

```
library(modelsummary)
library(kableExtra)

modelsummary(mtReg)
```

You can add several models to one table as follows:

```
mtReg.cyl <- lm(mpg~cyl, data=mtcars)

models <- list(
  "Horsepower" = mtReg,
  "Cylinders" = mtReg.cyl
)

modelsummary(models)
```

> Create a nice table of your results. Rename the predictor variable in the table to something human readable (see the `coef_rename` section)

Going deeper

Using the provided `china-aqi.csv` file,

1. Choose a predictor variable and a response variable
2. Make a scatterplot of the relationship and add a smoother - do you feel the relationship is a good fit?
3. Generate the regression results. Interpret your regression coefficients.
4. Check the residuals - do they confirm your visual inspection from the scatterplot? Why or why not?
5. Create a high-quality table based on your regression results using the `modelsummary()` command

If you have time, do the same for another predictor variable and the same response variable and add the results to your `modelsummary()` table.