Running a linear regression in R

September 2, 2021

Today, we will learn how to run a linear regression in R, examine the output, and add the regression line to a scatter plot. As an example, we will use the advertising data from the lecture video.

```
# load the data
library(tidyverse)
advertising_data = read_csv("../../data/Advertising.csv", col_types = "-dddd")
advertising_data
```

```
## # A tibble: 200 x 4
##       TV radio newspaper sales
##    <dbl> <dbl>     <dbl> <dbl>
##  1 230.   37.8      69.2  22.1
##  2  44.5  39.3      45.1  10.4
##  3  17.2  45.9      69.3   9.3
##  4 152.   41.3      58.5  18.5
##  5 181.   10.8      58.4  12.9
##  6   8.7  48.9      75     7.2
##  7  57.5  32.8      23.5  11.8
##  8 120.   19.6      11.6  13.2
##  9   8.6   2.1       1     4.8
## 10 200.    2.6      21.2  10.6
## # ... with 190 more rows
```

Running a linear regression

The function in R to run a linear regression is `lm()`, which stands for "linear model." Below is an example of a simple linear regression:

```
lm_fit = lm(formula = sales ~ TV, data = advertising_data)
```

The `formula` argument is used to specify the response variable and the features to use in the regression. In general, the syntax is

`response ~ feature_1 + feature_2 + ... + feature_p`.

An intercept term is included by default, unless it is suppressed using the `-1` syntax. Here are some examples of formulas:

- **Simple linear regression.**
  - Formula: `sales ~ TV`
  - Meaning: sales $\approx \beta_0 + \beta_1 \times$ TV
- **Removing the intercept.**
  - Formula: `sales ~ TV - 1`
  - sales $\approx \beta_1 \times$ TV
- **Multiple linear regression.**
  - Formula: `sales ~ TV + newspaper + radio`
  - Meaning: sales $\approx \beta_0 + \beta_1 \times$ TV $+ \beta_2 \times$ newspaper $+ \beta_3 \times$ radio

- **Using all other variables in data frame as features.**
  - Formula: `sales ~ .`
  - Meaning: sales $\approx \beta_0 + \beta_1 \times \text{TV} + \beta_2 \times \text{newspaper} + \beta_3 \times \text{radio}$

Inspecting the output

Let's run a regression of `sales` on each of the three other variables:

```
lm_fit = lm(formula = sales ~ ., data = advertising_data)
```

The object `lm_fit` now contains all the information about the linear regression. We can get a preview as follows:

```
lm_fit
```

```
##
## Call:
## lm(formula = sales ~ ., data = advertising_data)
##
## Coefficients:
## (Intercept)           TV        radio    newspaper
##    2.938889     0.045765     0.188530    -0.001037
```

This prints out the fitted coefficients. We can extract the coefficients into a vector as follows:

```
coefs = lm_fit$coefficients
coefs
```

```
##  (Intercept)           TV        radio    newspaper
##  2.938889369  0.045764645  0.188530017 -0.001037493
```

`coefs` is a vector of length 4, and we can operate on it as usual, e.g. subset it:

```
coefs[2:4]
```

```
##           TV        radio    newspaper
##  0.045764645  0.188530017 -0.001037493
```

The entries of the vector also have names, so we can subset the vector based on these names as well:

```
coefs[c("TV", "radio", "newspaper")]
```

```
##           TV        radio    newspaper
##  0.045764645  0.188530017 -0.001037493
```

You can extract lots of other information from the `lm_fit` object, such as the residuals and the fitted values. To get even more information, type `summary(lm_fit)`:

```
summary(lm_fit)
```

```
##
## Call:
## lm(formula = sales ~ ., data = advertising_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.8277 -0.8908  0.2418  1.1893  2.8292
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
```

```
## (Intercept)  2.938889   0.311908   9.422   <2e-16 ***
## TV           0.045765   0.001395  32.809   <2e-16 ***
## radio        0.188530   0.008611  21.893   <2e-16 ***
## newspaper   -0.001037   0.005871  -0.177     0.86
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.686 on 196 degrees of freedom
## Multiple R-squared:  0.8972, Adjusted R-squared:  0.8956
## F-statistic: 570.3 on 3 and 196 DF,  p-value: < 2.2e-16
```

We'll talk in more depth about interpreting this output in the next lecture, but for now observe that the R^2 can be extracted from the summary as follows:

```
summary(lm_fit)$r.squared
```

```
## [1] 0.8972106
```

Adding the regression line to a plot

The easiest way to add a (simple) regression line to a plot is by calling `geom_smooth()`:

```
advertising_data %>%
  ggplot(aes(x = TV, y = sales)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE) +
  theme_bw()
```

```
## `geom_smooth()` using formula 'y ~ x'
```