Linear Regression Demystified

An introduction to the world of predictive models

Jay Mangat

Table of Contents

Statistics was never my favourite subject in school. The complex ideas, overlapping concepts, and lack of actual numbers (Latin everywhere!) were some of the contributing factors to this mistaken belief. As I began relearning statistics I realized it provides some useful topics which can be used in all walks of life and not just problems within statistics, especially when creating prediction models. The two broad types of regression models are classification and linear models of which the latter will be the primary focus of this article.

Regression models are a broad subset of statistical models that can be used to create predictive models. Regression models have lots of different sub-types, as shown below, but the one we'll be focusing on is linear regression. These models take in multiple different types of data and create fairly accurate predictors which can be used to predict the outcome of data the model hasn't even seen yet. This ability for the model to make accurate predictions can be useful in all sorts of scenarios like predicting whether a team will win a game, whether a cell will become cancerous, and even a movie's rating based on its budget, gross box office earnings, or director. I'll use the final example in this tutorial to guide you through the main steps in creating a linear regression model.

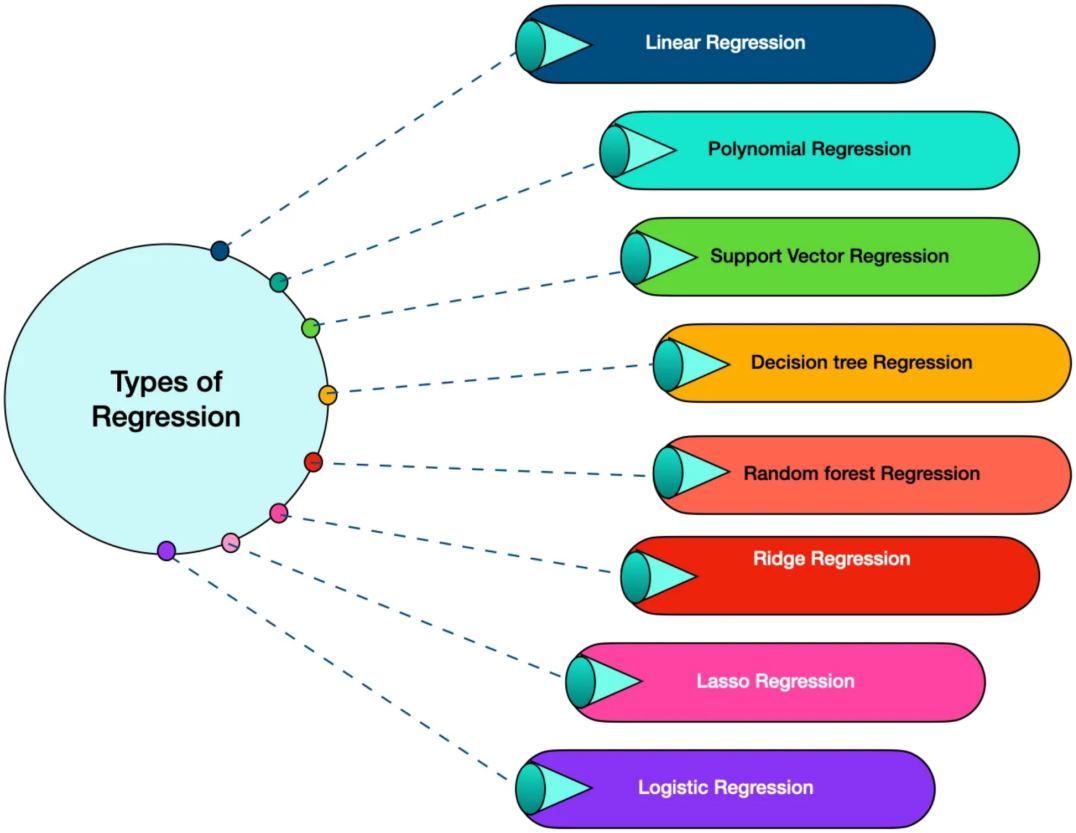


Figure 1: 1. Types of regression models

1 Creating the model

Possibly the most important step in creating a model is determining our question of interest. Given this data set, I want to know if we can create a model that will predict the rating (`imdb_rating`)of a movie based on the movie's total earnings `gross`, the movie's budget `budget`, and the movie's director `director`. By the end of this article, we'll know whether any of those variables are statistically significantly associated with the variable we're trying to prove i.e. `imdb rating`.

First we'll import the required libraries (you'll need to install them individually if you don't already have them via `install.packages(package_name)`).After this, we'll read in the data from a local folder. I've included the link to the Tidy Tuesday project from where I obtained this data on github. If you're comfortable, you can download the data directly from github and place it in an easily accessible folder.

```
# https://github.com/rfordatascience/tidytuesday/blob/main/data/2021/2021-03-09/movies.csv
movies <- read.csv('movies.csv')
colnames(movies)
```

```
 [1] "year"          "imdb"          "title"         "test"
 [5] "clean_test"    "binary"        "budget"        "domgross"
 [9] "intgross"      "code"          "budget_2013"   "domgross_2013"
[13] "intgross_2013" "period_code"   "decade_code"   "imdb_id"
[17] "plot"          "rated"         "response"      "language"
[21] "country"       "writer"        "metascore"     "imdb_rating"
[25] "director"      "released"      "actors"        "genre"
[29] "awards"        "runtime"       "type"          "poster"
[33] "imdb_votes"    "error"
```

Based on our initial idea we only need 3 of the columns as our predictors and the `imdb_rating` column as our prediction but this data set comes with 34 columns. Since we don't need all of them we'll filter the data to fit our analysis. In the code chunk below we perform the following steps (listed in the same order so its easy to follow):

1. Retrieve only movies from the data set since it also contains tv shows
2. Convert `intgross` and `domgross` to integers so we can add them
3. Converts director to a factor so it can be used in our regression model
4. Create a `gross` column using the newly converted `intgross` and `domgross`
5. Select the 4 columns we're interested in
6. Drop all rows which contain NAs in any slot. This is a common step of cleaning data and will ensure nothing breaks as we do our analysis

```
movies_eda <- movies |>
  filter(type == 'movie') |>
  mutate(intgross = as.numeric(intgross),
         domgross = as.numeric(domgross),
         director = as.factor(director),
         gross = domgross + intgross) |>
  select(c(gross, budget, imdb_rating, director)) |>
  drop_na()

head(movies_eda)
```

```
      gross    budget imdb_rating          director
1 211714070  20000000         8.3     Steve McQueen
2 208105475  61000000         6.8 Baltasar Kormákur
```

```
3 190040426  40000000         7.6   Brian Helgeland
4 184166317 225000000         6.6       Carl Rinsch
5 371598396  92000000         5.4        John Moore
6 102648667  12000000         7.8    Richard Curtis
```

The next step in creating our linear regression model is to create the model and specify what we're looking for. This part's simple as we'll be using the built-in R functions to accomplish this. The general layout of the `lm()` function as follows:

- `lm(response ~ explanatory1 + explanatory2 + explanatory3, data)`
  - response - the data we're trying to predict. In this example we're interested in a movie's total budget but you can just as easily build a model that tries to predict budget, country, or director (albeit the second two are more in-depth and require some complicated workarounds to make the data usable)
  - explanatory - these are the predictors or data that we think is connected to the response and want to calculate whether it is or isn't. For us, these are gross, budget, and director
  - data - this is the data set where we're drawing both our response and explanatory data from

```
model <- lm(imdb_rating ~ gross + budget + director, movies_eda) |>
  tidy() |>
  filter(p.value < 0.05)
model
```

```
# A tibble: 69 x 5
   term                                estimate std.error statistic  p.value
   <chr>                                  <dbl>     <dbl>     <dbl>    <dbl>
 1 (Intercept)                          7.14e+0  6.47e- 1     11.0  3.65e-26
 2 gross                                1.06e-9  1.14e-10      9.31 1.68e-19
 3 budget                              -8.22e-9  7.11e-10    -11.6  2.16e-28
 4 directorAlan Metter                 -4.06e+0  9.16e- 1     -4.43 1.10e- 5
 5 directorAlan Poul                   -1.87e+0  9.16e- 1     -2.04 4.16e- 2
 6 directorAndrew Bergman              -2.68e+0  9.16e- 1     -2.93 3.52e- 3
 7 directorAngela Robinson             -2.08e+0  7.93e- 1     -2.62 8.94e- 3
 8 directorAnnabel Jankel, Rocky Morton -2.84e+0  9.16e- 1     -3.10 2.03e- 3
 9 directorBradley Parker              -2.19e+0  9.16e- 1     -2.39 1.71e- 2
10 directorBrian Levant                -1.71e+0  7.48e- 1     -2.28 2.28e- 2
# i 59 more rows
```

Putting the output of `lm()` into `tidy()` gives us a visually appealing output which is easier to understand. Additionally, I've also filtered p.value to less than 0.05 to cut down the data to its most useful parts. The p-value is a way to determine whether a variable is statistically associated with our response (this is true for all the variables above since we've filtered for variables which have a p-value less than 0.05 i.e. statistically significant p-values).

2 Interpreting Model Output

After all this work we finally get to savor the fruits of our labor. Now that we have the output of our model we can interpret the various variables and see what they mean. As a reminder of our initial questions, we want to see if any of `gross`, `budget`, or `director` are associated with `imdb_rating`. We'll interpret gross and budget together since they're simpler to understand than director and cover director at the end.

The estimate for `gross` is 1.06e-09 or 0.00000000106. This means that for every $1 increase in a film's total gross, its `imdb_rating` will be 0.00000000106 higher. On its own this seems useless but we know that the earnings for movies is almost always in the millions of dollars if not hundreds of millions of dollars so we can try interpreting this estimate taking this knowledge into account. For every $100,000,000 increase in a film's `gross`, its `imdb_rating` will be 0.106 higher. Even this interpretation seems like a bad return on investment for any film studio as in order to increase their film's rating by 1 point, they'll have to spend $1,000,000,000. Let's try budget and see if it has more of an effect on rating.

The estimate for `budget` is -8.22e-09 or -0.00000000822. We know from gross that it makes more sense to interpret this estimate in millions of dollars so for every $100,000,000 increase in a film's `budget` its `imdb_rating` decreases by 0.822. There's multiple pieces of information we can get from this estimate. First, the estimate is negative instead of positive like for `gross` so increase a movie's `budget` will lead to a decrease in `imdb_rating`. Second, the amount of the decrease (0.8) is much higher than the amount of the increase from a movie's `gross` (0.1) so a movie's `budget` is a stronger predictor of a movie's `imdb_rating` than its `gross`.

`gross` and `budget` were easier to interpret because they were continuous explanatory variables but director contains different types of names so `lm()` treats it differently. `lm()` chooses a baseline director with whom it compares all other directors. The chosen baseline is the first level for the director column which can be determined using the code below:

```
head(levels(movies_eda$director))
```

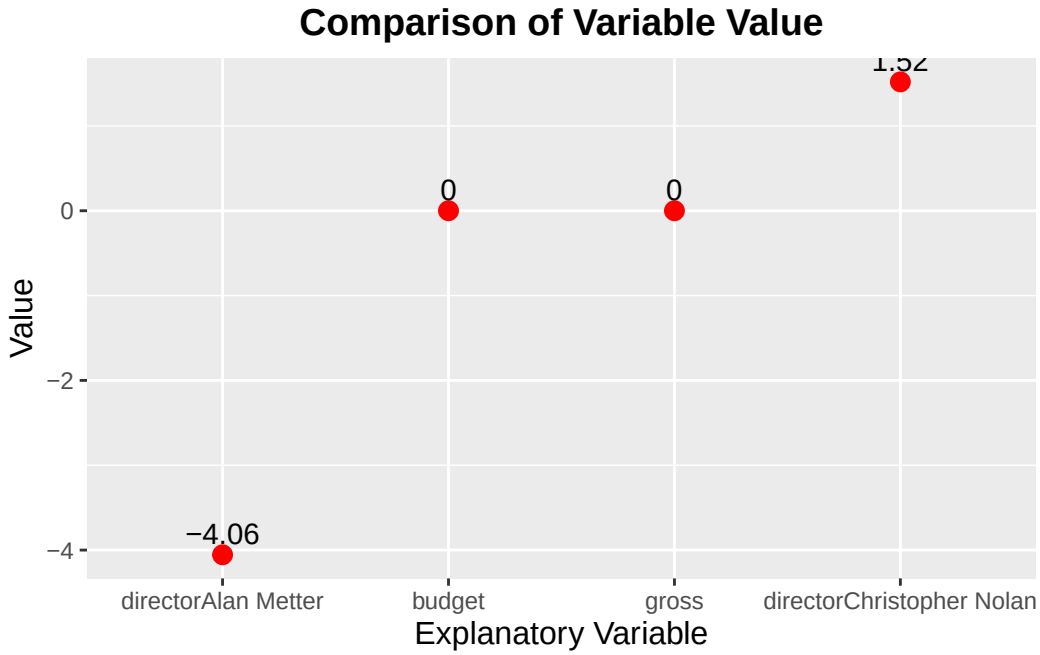
```
[1] "Aaron Schneider" "Adam Brooks"     "Adam Green"      "Adam McKay"
[5] "Adam Shankman"   "Adrian Lyne"
```

The first director is Aaron Schneider so all other directors are compared to this director. The estimates given for each `director` is the increase or decrease in `imdb_rating` if you were to switch from Aaron to that director while keeping all other variables (`gross`, `budget`) the same. Lets go over two examples to illustrate the point.

- Alan Metter has an estimate of -4.05. The interpretation of this estimate is that if you switched the `director` of a movie from Aaron Schneider to Alan Metter while keeping the movie's `budget` and `gross` exactly the same, you can expect the movie's `imdb_rating` to drop by 4.05 points
- Christopher Nolan has an estimate of 1.52. The interpretation of this estimate is that if you switched the `director` of a movie from Aaron Schneider to Christopher Nolan while keeping the movie's `budget` and `gross` exactly the same, you can expect the movie's `imdb_rating` to increase by 1.52 points.

If you're a visual learner, you can also compare all 4 of our chosen examples via a plot to see the effect each of these on `imdb_rating`.

```
model |>
  slice(c(2, 3, 4, 15)) |>
  arrange(estimate) |>
  ggplot(aes(x=reorder(term, estimate), y=estimate, label=round(estimate, 2))) +
  geom_point(size=3, color="red") +
  geom_text(vjust=-0.5, hjust=0.5) +
  labs(
    title="Comparison of Variable Value",
    x="Explanatory Variable",
    y="Value"
  ) +
  theme(
    plot.title = element_text(hjust=0.5, face="bold", size=14),
    axis.title = element_text(size=12)
  )
```



3 Final thoughts for future study

This tutorial covered how to create a simple `lr()` model in R and covered the interpretations for the outputs of that model. We also covered some more advanced topics like dealing with non-continuous columns which should give you a good starting point for further study. At the end of a exercise we should also look back and see where we could have improved. I'll list some of my own findings here so you can get an idea of how to look back and review your own work.

Things to keep in mind for future projects:

- Was the exploratory data analysis (EDA) satisfactory? This depends on our goal for the data and differs from project to project. In this case, I think I should have paid more attention to the director column and cleaned it more. Having 66 directors who are statistically associated with `imdb_rating` seems high. After some of my own searching, I found that most of these directors have 1-2 movies under their belt so their estimates are calculated based on only 1-2 data points. This isn't a reliable method for predicting association so for next time we can select only directors who have more than 5 movies since 5+ movies allows for a pattern.
- Even though we only chose the 4 columns we're interested in we could have still used the other columns in filtering our data. A good example is if we're only interested in a

specific date range. The `year` column is useful for this as it will allow us to select movies between certain dates.

- We only used 3 explanatory variables (`gross`,`budget` , and `director`) but there could be others within the dataset which are better predictors of `imdb_rating` than these three. For future iterations we can try adding more of these to our model and check their effect on the response.

Learning statistical models is no easy feat especially if you come from a different background (mine was finance) so don't feel overwhelmed. There's many different websites and videos which explain the ideas in detail and are available for all expertise levels. Once these concepts click, it's easy to see the benefit to using these models in all sorts of disciplines than just statistics or math. It took me a long time to see the power of predictive models but the work was worth it as I now have a powerful new tool in my professional toolbox.

3.1 References

1. Regression Algorithms