HW 03: Multiple linear regression, Part 2

Dav King

2022-11-05

Set up

```
library(tidyverse)
library(tidymodels)
library(knitr)
library(rms)
library(patchwork)

legos <- read_csv("data/lego-sample.csv") |>
  select(Size, Pieces, Theme, Amazon_Price, Year, Pages) |>
  drop_na()
```

> **Note**
>
> Select this page for Workflow & formatting".

Exercises

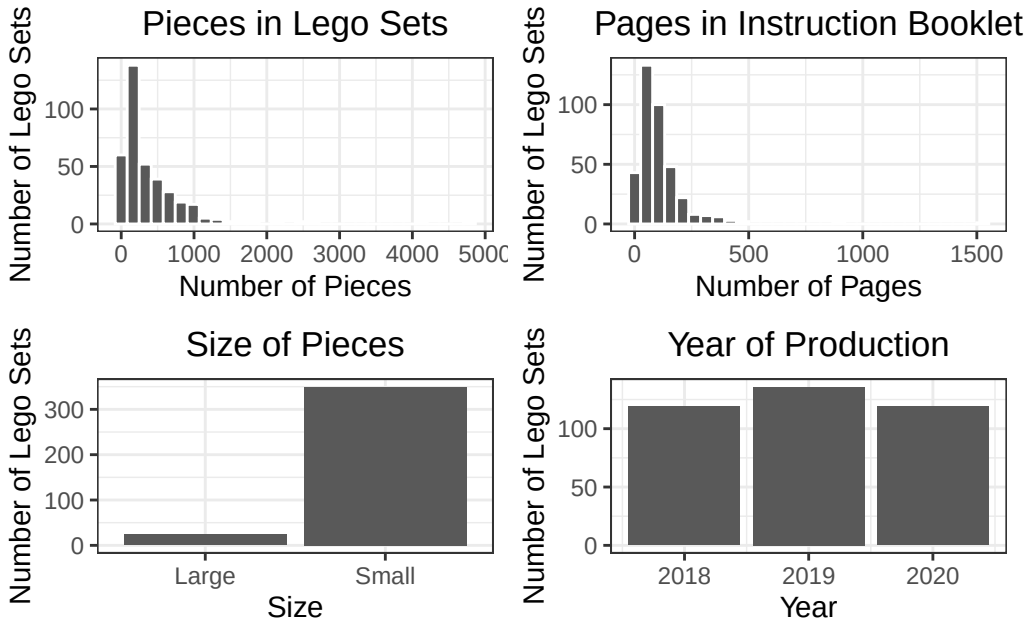
Exercise 1

There are two main problems with dropping observations with missing values from our dataset. First, we are throwing away data that could possibly be used to make our analysis more robust, which is never ideal. Second, we don't know whether there was any trend in the data that caused them to have missing values. In other words, the observations that we're dropping are not necessarily independent of one another, and so we may be losing valuable pieces of information. This may reduce the generalizability of our conclusions - since our sample is technically only complete cases, our findings can only be generalized to complete cases as well. This severely limits what our findings actually apply to in the population.

Exercise 2

```
pieces <- ggplot(legos, aes(x = Pieces)) +
  geom_histogram(color = "white") +
  theme_bw() +
  labs(x = "Number of Pieces", y = "Number of Lego Sets",
       title = "Pieces in Lego Sets") +
  theme(plot.title = element_text(hjust = 0.5))

size <- ggplot(legos, aes(x = Size)) +
  geom_bar() +
  theme_bw() +
  labs(x = "Size", y = "Number of Lego Sets",
       title = "Size of Pieces") +
  theme(plot.title = element_text(hjust = 0.5))

year <- ggplot(legos, aes(x = Year)) +
  geom_bar() +
  theme_bw() +
  labs(y = "Number of Lego Sets", title = "Year of Production") +
  theme(plot.title = element_text(hjust = 0.5))

pages <- ggplot(legos, aes(x = Pages)) +
  geom_histogram(color = "white") +
  theme_bw() +
  labs(x = "Number of Pages", y = "Number of Lego Sets",
       title = "Pages in Instruction Booklet") +
  theme(plot.title = element_text(hjust = 0.5))

(pieces + pages) / (size + year)
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

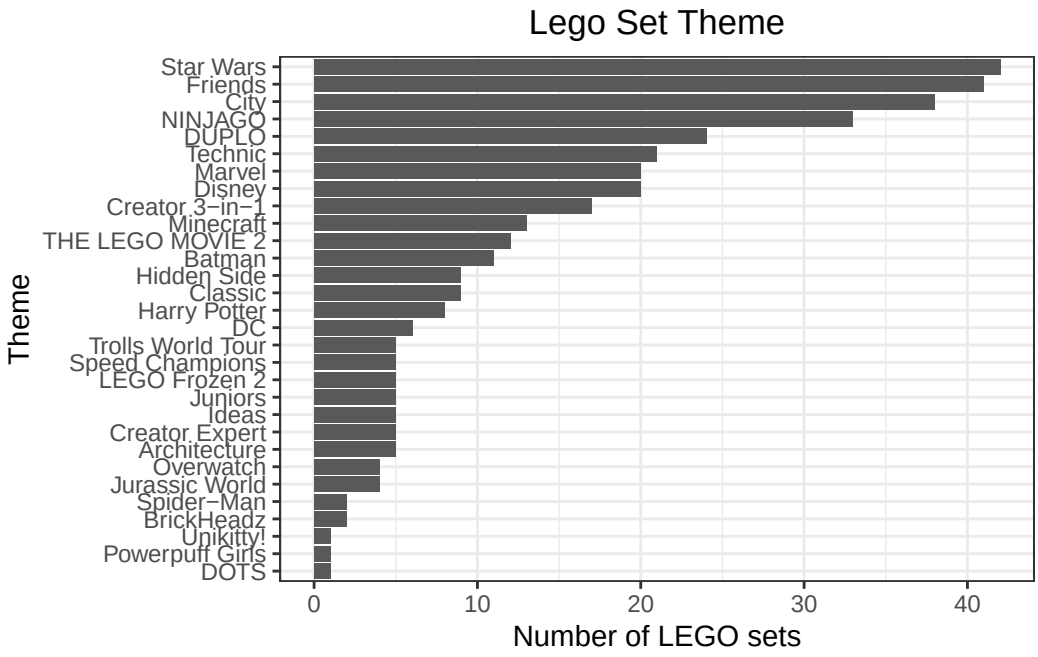

Pieces in Lego Sets
Number of Lego Sets
100
50
0
0
1000
2000
3000
4000
5000
Number of Pieces
Pages in Instruction Booklet
Number of Lego Sets
100
50
0
0
500
1000
1500
Number of Pages
Size of Pieces
Number of Lego Sets
300
200
100
0
Large
Small
Size
Year of Production
Number of Lego Sets
100
50
0
2018
2019
2020
Year

Exercise 3

Because `Year` only encompasses 3 different years, it should really be treated as a factor in this model. However, it is a numerical variable in the `legos` dataframe. Thus, my first function will be using `step_dummy()` to convert `Year` into a factor variable. Next, I would want to ensure I don't have any meaningless predictors, so I would remove any predictors that have zero variance using `step_zv()`. Third, I would like to have an interpretable intercept. Thus, I will use `step_center()` on all of my numerical variables to give me a meaningful interpretation of the intercept - the price we would expect a lego set that is average in all other variables to cost.

Exercise 4

```
legos %>%
  count(Theme) %>%
  ggplot(aes(x = fct_reorder(Theme, n), y = n)) +
  geom_col() +
    labs(title = "Lego Set Theme",
         x = "Theme",
         y = "Number of LEGO sets") +
  coord_flip() +
  theme_bw() +
  theme(plot.title = element_text(hjust = 0.5))
```



There are simply too many levels to `Theme`. If we were to put `Theme` into our model as is, we would have some 30 different slopes associated with these different levels, and this would be entirely meaningless for some of the lego sets that only have 1 or 2 observations. Instead, we should collapse it into a meaningful number of levels using `step_other()`.

Exercise 5

```
set.seed(5)
lego_split <- initial_split(legos)
lego_train <- training(lego_split)
lego_test <- testing(lego_split)

set.seed(5)
lego_folds <- vfold_cv(lego_train, 10)

spec <- linear_reg() %>%
  set_engine("lm")
```

Exercise 6

```
lego_rec <- recipe(Amazon_Price ~ Size + Theme + Pages, lego_train) %>%
  step_other(Theme, threshold = 20) %>%
  step_center(Pages) %>%
  step_dummy(all_nominal_predictors()) %>%
  step_zv(all_predictors())

lego_wflow <- workflow() %>%
  add_model(spec) %>%
  add_recipe(lego_rec)
```

Exercise 7

```
lego_fit_rs <- lego_wflow %>%
  fit_resamples(resamples = lego_folds,
                control = control_resamples())
```

```
Warning: package 'rlang' was built under R version 4.1.3
```

```
collect_metrics(lego_fit_rs) %>%
  filter(.metric == 'rmse') %>%
  select(.metric, mean) %>%
  kable(col.names = c("Summary", "Mean Value"), digits = 3)
```

Summary	Mean Value
rmse	50.753

Exercise 8

```
lego_rec2 <- recipe(Amazon_Price ~ ., lego_train) %>%
  step_other(Theme, threshold = 20) %>%
  step_center(Pages, Pieces) %>%
  step_mutate(since2018 = Year - 2018) %>%
  step_rm(Year) %>%
  step_dummy(all_nominal_predictors()) %>%
  step_zv(all_predictors())

lego_wflow2 <- workflow() %>%
  add_model(spec) %>%
  add_recipe(lego_rec2)

lego_fit_rs2 <- lego_wflow2 %>%
  fit_resamples(resamples = lego_folds,
                control = control_resamples())

collect_metrics(lego_fit_rs2) %>%
  filter(.metric == 'rmse') %>%
  select(.metric, mean) %>%
  kable(col.names = c("Summary", "Mean Value"), digits = 3)
```

Summary	Mean Value
rmse	31.404

Exercise 9

Based on our summarized RMSE scores, we would select the second model. Its mean RMSE score across the ten folds is much lower (31.404 vs 50.753), meaning that its average error is a lot less and it is thus a better fit for the training data.

Exercise 10

```
lego_fit <- lego_wflow2 %>%
  fit(lego_train)

tidy(lego_fit) %>%
  kable(digits = 3)
```

term	estimate	std.error	statistic	p.value
(Intercept)	101.754	9.834	10.348	0.000
Pieces	0.118	0.005	23.402	0.000
Pages	-0.085	0.023	-3.724	0.000
since2018	-13.756	2.199	-6.257	0.000
Size_Small	-17.963	7.779	-2.309	0.022
Theme_Friends	-13.581	7.703	-1.763	0.079
Theme_NINJAGO	-17.166	8.012	-2.142	0.033
Theme_Star.Wars	-3.131	7.833	-0.400	0.690
Theme_other	-15.755	6.174	-2.552	0.011

```
lego_fit_extract <- extract_fit_parsnip(lego_fit)
vif(lego_fit_extract$fit)
```

```
      Pieces        Pages    since2018   Size_Small Theme_Friends
    2.827464     2.898586     1.014699     1.088859     1.951379
Theme_NINJAGO Theme_Star.Wars  Theme_other
    1.868271     2.130649     3.110897
```

Our VIF values do not give us any reason for concern with multicollinearity. None of these values are above 10 or even close to it, so we do not have cause for concern.

Exercise 11

```
trainPred <- predict(lego_fit, lego_train) %>%
  bind_cols(lego_train)
rmse(trainPred, truth = Amazon_Price, estimate = .pred)
```

```
# A tibble: 1 x 3
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 rmse    standard        28.5
```

```
testPred <- predict(lego_fit, lego_test) %>%
  bind_cols(lego_test)
rmse(testPred, truth = Amazon_Price, estimate = .pred)
```

```
# A tibble: 1 x 3
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 rmse    standard        25.2
```

Surprisingly, the RMSE score for the model's performance on the testing data is actually better than its score on the training data, with an RMSE score of 25.2 (compared to 28.5) suggesting the model is wrong by less, on average, on the new data. This completely avoids signs of model overfit - given that the model is actually more effective (by RMSE standards) on the new data, it is clearly versatile and adapted well to its purpose.

Exercise 12

Many different categories within `Theme` have a significant impact on the price of a lego set. Compared to the baseline of a `Theme` City, we would expect a lego set from NINJAGO to cost \$17.166 less, on average, holding all else constant (significant at the $\alpha = .05$ level, p = .033). Compared to the baseline of a City `Theme` lego set, we would expect a legoset from a theme that is not Star Wars, Friends, or NINJAGO to cost \$15.755 less, on average, holding all else constant (significant at the $\alpha = .05$ level, p = .011).

The other two `Theme` categories are not statistically significant. Compared to the baseline of a City `Theme` lego set, we would expect a Friends `Theme` lego set to cost \$13.581 less, on average, holding all else constant. However, this effect is not significant at the $\alpha = .05$ level (p = .079). Compared to the baseline of a City `Theme` lego set, we would expect a Star Wars `Theme` lego set to cost \$3.131 less, on average, holding all else constant. However, this effect is not significant at the $\alpha = .05$ level (p = .690).