AE 12: Multiple linear regression review

LEGO sets

Oct 31, 2022

❗ Important

The AE is due on GitHub by Thursday, November 03, 11:59pm.

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

The data for this analysis includes information about LEGO sets from themes produced January 1, 2018 and September 11, 2020. The data were originally scraped from Brickset.com, an online LEGO set guide.

You will work with data on about 400 randomly selected LEGO sets produced during this time period. The primary variables are interest in this analysis are

- `Pieces`: Number of pieces in the set from brickset.com.
- `Amazon_Price`: Amazon price of the set scraped from brickset.com (in U.S. dollars)
- `Size`: General size of the interlocking bricks (Large = LEGO Duplo sets - which include large brick pieces safe for children ages 1 to 5, Small = LEGO sets which- include the traditional smaller brick pieces created for age groups 5 and - older, e.g., City, Friends)
- `Theme`: Theme of the LEGO set

The goal of this analysis is to predict the Amazon price based on the number of pieces, theme, and size of pieces. We will only include observations that have recorded values for all relevant variables.

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

- What is a disadvantage of dropping observations that have missing values, instead of using a method to impute, i.e., fill in, the missing data? How might dropping these observations impact the generalizability of conclusions?

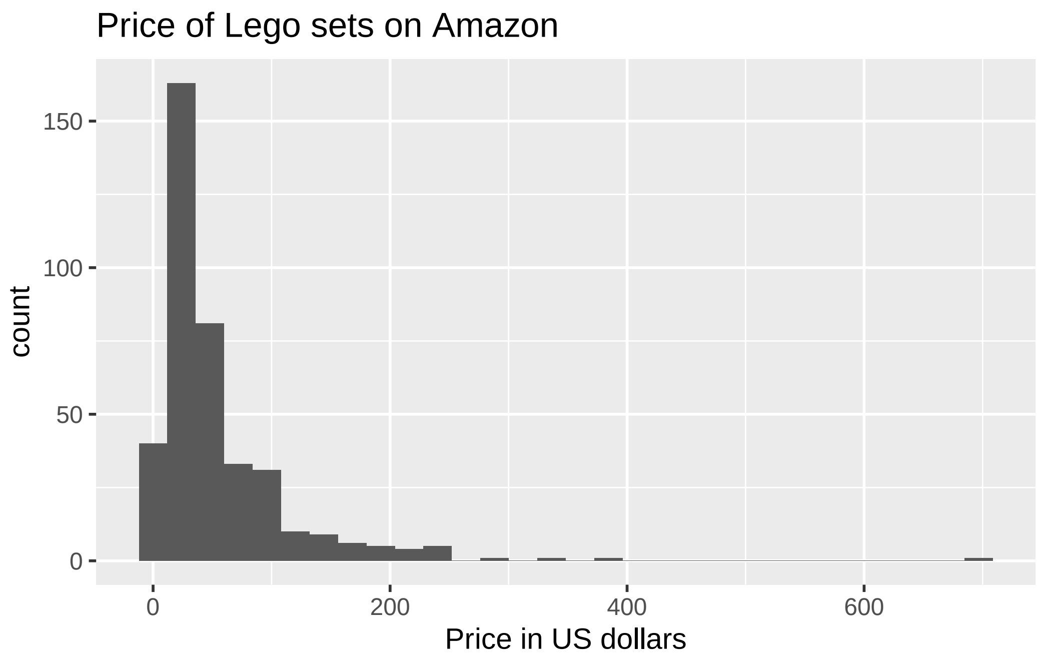
The main disadvantage is that we may be dropping observations that are not independent of one another, and thus possibly removing an important piece of information from the model. This could prevent us from generalizing our conclusions to ALL lego sets - we might instead only be able to generalize to lego sets where the size, number of pieces, and theme are all known.

Can impute based on median or mean of existing data for numeric predictors, mode for categorical predictors (or create a new level called "Missing"). Can create indicator variables to specify observations where you imputed missing data.

Exploratory data analysis

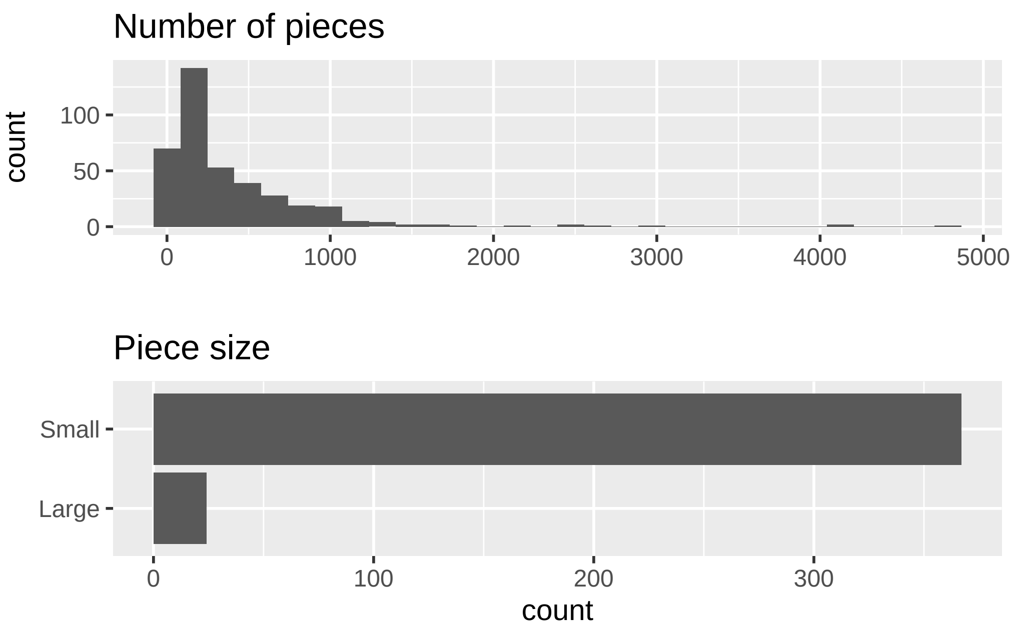
Response variable

```r
ggplot(data = legos, aes(x = Amazon_Price)) +
  geom_histogram() +
    labs(title = "Price of Lego sets on Amazon",
         x = "Price in US dollars")
```



Predictor variables

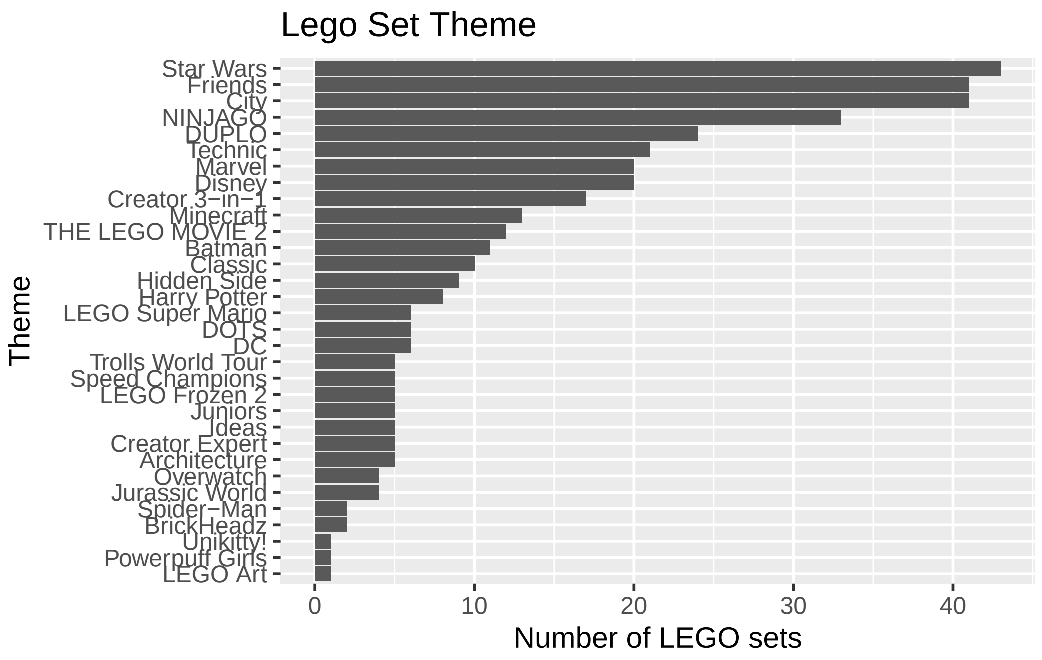
```
p_pieces <- ggplot(data = legos, aes(x = Pieces)) +
  geom_histogram() +
    labs(title = "Number of pieces",
         x = "")

p_size <- ggplot(data = legos, aes(x = Size)) +
  geom_bar() +
    labs(title = "Piece size",
         x = "") +
  coord_flip()

p_pieces / p_size
```



- What (if any) feature engineering might we want to include in a recipe for a model for `Pieces` and `Size`?

It's very possible that we should log-transform `Pieces`, in order to get it to resemble a much more normal distribution. We would also probably turn `Size` into a dummy variable. We can also mean-center our variables in order to make for an interpretable intercept.

```
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()
```



- Why should we avoid putting `Theme` in a model as is?

We're going to have wayyyyyy too many predictors in our model if we include all of these different `Theme`s as predictors in our model. It will be very difficult to draw any meaningful conclusions from that model.

- How can we use `step_other()` in the recipe to make `Theme` more usable in the model?

`step_other()` will pool any observations in a categorical predictor that do not occur more frequently than some specified threshold into an overarching `other` category. This means that any of our `Theme`s that are not particularly common can all be grouped together, and we can leave ourselves with a reasonable, interpretable number of predictors in the model.

Model Fitting

Training and test sets

Write code to data into training (75%) and testing (25%) sets.

```r
set.seed(1031)

lego_split <- initial_split(legos)
lego_train <- training(lego_split)
lego_test <- testing(lego_split)
```

Model workflow

Fit the model using `Pieces`, `Size`, and `Theme` to understand variability in `Amazon_Price`. To do so, specify the model and use training data to create a recipe applying the feature engineering steps mentioned above. Then, build the model workflow and fit the model.

```r
lego_model <- linear_reg() %>%
  set_engine("lm")
```

```r
lego_rec <- recipe(Amazon_Price ~ ., lego_train) %>%
  step_other(Theme) %>%
  step_dummy(Size)
```

```r
# This is an optional step to see the outcome of the recipe

# add code
```

```r
lego_workflow <- workflow() %>%
  add_recipe(lego_rec) %>%
  add_model(lego_model)
```

```r
lego_fit <- lego_workflow %>%
  fit(data = lego_train)

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

term	estimate	std.error	statistic	p.value
(Intercept)	18.106	5.938	3.049	0.003
Pieces	0.107	0.003	31.848	0.000
ThemeDUPLO	6.794	8.791	0.773	0.440
ThemeFriends	-12.338	7.850	-1.572	0.117
ThemeMarvel	-7.453	9.588	-0.777	0.438
ThemeNINJAGO	-7.942	9.230	-0.860	0.390
ThemeStar Wars	3.520	7.616	0.462	0.644
ThemeTechnic	-17.078	9.919	-1.722	0.086
Themeother	-9.936	6.393	-1.554	0.121
Size_Small	NA	NA	NA	NA

Check conditions + multicollinearity

When we fit a model using `recipe` and `workflow`, we need to extract the model object before using the `augment` function. Fill in the name of the fitted model in the code below.

Update the option to `eval: true`, so the code chunk evaluates when the document is rendered.

```
lego_fit_model <- extract_fit_parsnip(______)
lego_aug <- augment(lego_fit_model)
```

Residuals vs predicted values

Use the code below to make a plot of residuals vs. predicted values

Update the option to `eval: true`, so the code chunk evaluates when the document is rendered.

```
ggplot(data = lego_aug, aes(x = .fitted, y= .resid)) +
  geom_point() +
  geom_hline(yintercept = 0, color = "red", linetype = "dashed") +
  labs(x = "Predicted",
       y = "Residuals",
       title = "Residuals vs. Predicted")
```

Distribution of residuals

Fill in the code below to make a histogram of the residuals with an overlay of the normal distribution.

> Update the option to `eval: true`, so the code chunk evaluates when the document is rendered.

```r
ggplot(lego_aug, aes(.resid)) +
  geom_histogram(aes(y = after_stat(density))) +
  stat_function(
    fun = dnorm,
    args = list(mean = mean(lego_aug$_____), sd = sd(_____)),
    color = "red"
  )
```

Use the plots and information about the data to comment on whether each condition is satisfied.

- Linearity: [Add response]
- Constant variance: [Add response]
- Normality: [Add response]
- Independence: [Add response]

Multicollinearity

Fun the code to use the `vif()` function from the **rms** package to check multicollinearity.

> Update the option to `eval: true`, so the code chunk evaluates when the document is rendered.

```r
vif(lego_fit_model$fit)
```

- Are there issues with multicollinearity in the model?

Next steps

- Based on the assessment of the model conditions and multicollinearity, what is the next step you might take in the model building process?

❗ Important

To submit the AE:

- Render the document to produce the PDF with all of your work from today's class.
- Push all your work to your `ae-12-` repo on GitHub. (You do not submit AEs on Gradescope).