Exercise 12

Marc Dotson

Multiple regression is often referred to as “key drivers analysis” in business applications. In the Nielsen soup\_data, now available on the AWS database, the outcome of interest is Sales.

1. Using soup\_data, build and compare four *multiple* regressions for CAMPBELL’S Sales for the WEST CENSUS TA using the following promotional explanatory variables: Any\_Disp\_Spend (spend on in-store displays), Any\_Feat\_Spend (spend on coupons and other printed promotions, collectively called “features”), and Any\_Price\_Decr\_Spend (spend on price decreases).
2. Use the best-fitting model to predict counterfactual Sales for Campbell’s. Assume that Campbell’s can spend up to $10,000 for this trading area. Produce predictions for four counterfactual scenarios and use them to detail a proposal for Campbell’s for the WEST CENSUS TA.
3. Render the Quarto document into Word and upload to Canvas and complete the mid-course evaluation.

**Five points total, one point each for:**

* **Importing soup\_data from the database.**
* **Filtering the data for CAMPBELL'S Sales and ‘WEST CENSUS TA’.**
* **Fitting four multiple regressions using a combination of the three explanatory variables and comparing overall model fit.**
* **Produce predictions for four counterfactual scenarios of sales using the best-fitting model and detail a policy proposal for Campbell’s based on the predictions.**
* **One point for submitting a rendered Word document.**

## Model Building

Let’s load all the packages we’ll need.

# Load packages.  
library(tidyverse)

── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──  
✔ dplyr 1.1.4 ✔ readr 2.1.4  
✔ forcats 1.0.0 ✔ stringr 1.5.1  
✔ ggplot2 3.4.4 ✔ tibble 3.2.1  
✔ lubridate 1.9.3 ✔ tidyr 1.3.0  
✔ purrr 1.0.2   
── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
✖ dplyr::filter() masks stats::filter()  
✖ dplyr::lag() masks stats::lag()  
ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors

library(tidymodels)

── Attaching packages ────────────────────────────────────── tidymodels 1.1.1 ──  
✔ broom 1.0.5 ✔ rsample 1.2.0  
✔ dials 1.2.0 ✔ tune 1.1.2  
✔ infer 1.0.5 ✔ workflows 1.1.3  
✔ modeldata 1.2.0 ✔ workflowsets 1.0.1  
✔ parsnip 1.1.1 ✔ yardstick 1.2.0  
✔ recipes 1.0.9   
── Conflicts ───────────────────────────────────────── tidymodels\_conflicts() ──  
✖ scales::discard() masks purrr::discard()  
✖ dplyr::filter() masks stats::filter()  
✖ recipes::fixed() masks stringr::fixed()  
✖ dplyr::lag() masks stats::lag()  
✖ yardstick::spec() masks readr::spec()  
✖ recipes::step() masks stats::step()  
• Search for functions across packages at https://www.tidymodels.org/find/

We first need to import soup\_data from the database.

# Connect to the database.  
con <- DBI::dbConnect(  
 RPostgreSQL::PostgreSQL(),  
 dbname = "analyticsdb",  
 host = "analyticsdb.ccutuqssh92k.us-west-2.rds.amazonaws.com",  
 port = 55432,  
 user = "quantmktg",  
 password = rstudioapi::askForPassword("Database password")  
)  
  
# Import the data.  
soup\_data <- dplyr::tbl(con, "soup\_data") |>  
 collect()  
  
# Disconnect from the database.  
DBI::dbDisconnect(con)  
  
# Write the data locally.  
write\_csv(here::here("Data", "soup\_data.csv"))

Now that we have imported data from the database once, we don’t need to reconnect so I’ve set the above to #| eval: false. Let’s import the data, filter to the specific brand and trade area, and start fitting models.

# Import and filter data.  
soup\_data <- read\_csv(here::here("Data", "soup\_data.csv")) |>   
 filter(Retailer\_Trade\_Areas == "WEST CENSUS TA", Brand\_High == "CAMPBELL'S")

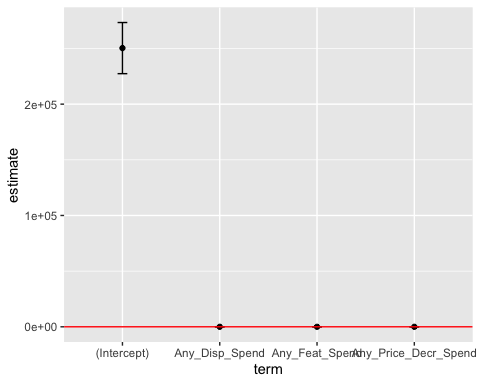
Rows: 3192 Columns: 66  
── Column specification ────────────────────────────────────────────────────────  
Delimiter: ","  
chr (5): Four\_Weeks\_Periods, Retailer\_Trade\_Areas, Category, Sub\_Category, ...  
dbl (61): Sales, Base\_Sales, Incr\_Sales, Units, Base\_Units, Incr\_Units, Perc...  
  
ℹ Use `spec()` to retrieve the full column specification for this data.  
ℹ Specify the column types or set `show\_col\_types = FALSE` to quiet this message.

# Full model (all three explanatory variables).  
fit\_01 <- linear\_reg() |>   
 set\_engine(engine = "lm") |>   
 fit(  
 Sales ~ Any\_Disp\_Spend + Any\_Feat\_Spend + Any\_Price\_Decr\_Spend,   
 data = soup\_data  
 )  
  
# Print parameter estimates.  
tidy(fit\_01, conf.int = TRUE)

# A tibble: 4 × 7  
 term estimate std.error statistic p.value conf.low conf.high  
 <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
1 (Intercept) 2.50e+5 1.14e+4 21.9 1.21e-26 2.27e+5 2.73e+5  
2 Any\_Disp\_Spend 1.27e+0 3.02e-1 4.19 1.19e- 4 6.59e-1 1.88e+0  
3 Any\_Feat\_Spend 4.42e-1 1.61e-1 2.74 8.62e- 3 1.18e-1 7.67e-1  
4 Any\_Price\_Decr\_Spend 3.25e-1 9.33e-2 3.48 1.09e- 3 1.37e-1 5.12e-1

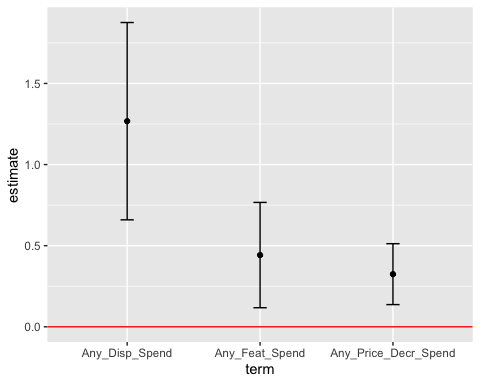
All of the parameter estimates are significant, but the effect of each of the explanatory variables on the outcome are very small, especially compared to the intercept.

# Visualize parameter estimates.  
tidy(fit\_01, conf.int = TRUE) |>   
 ggplot(aes(x = term)) +   
 geom\_point(aes(y = estimate)) +   
 geom\_errorbar(aes(ymin = conf.low, ymax = conf.high), width = .1) +  
 geom\_hline(yintercept = 0, color = "red")



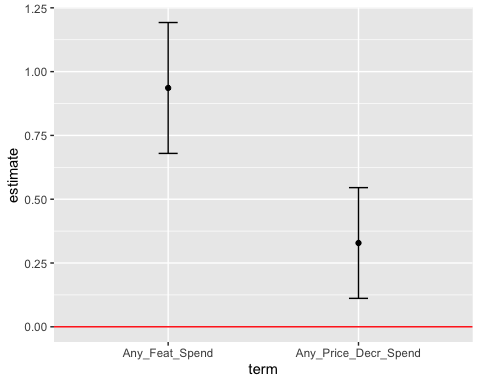
The scale is so different between the intercept and the slope parameters that we can’t even see the difference between the slope parameters. Since the slope parameters are really what we’re interested in here, let’s just plot them.

# Visualize slope parameter estimates.  
tidy(fit\_01, conf.int = TRUE) |>   
 filter(term != "(Intercept)") |>   
 ggplot(aes(x = term)) +   
 geom\_point(aes(y = estimate)) +   
 geom\_errorbar(aes(ymin = conf.low, ymax = conf.high), width = .1) +  
 geom\_hline(yintercept = 0, color = "red")



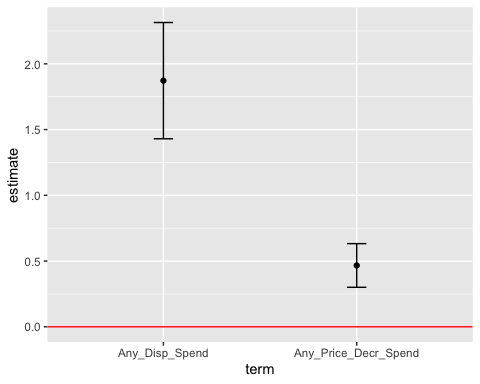
Display spend clearly matters the most. In fact, it’s the only one of the parameters with a confidence interval that suggests it might have an effect on Sales greater than 1 (i.e., would yield a positive return on investment).

# Model without display spend.  
fit\_02 <- linear\_reg() |>   
 set\_engine(engine = "lm") |>   
 fit(  
 Sales ~ Any\_Feat\_Spend + Any\_Price\_Decr\_Spend,   
 data = soup\_data  
 )  
  
# Visualize slope parameter estimates.  
tidy(fit\_02, conf.int = TRUE) |>   
 filter(term != "(Intercept)") |>   
 ggplot(aes(x = term)) +   
 geom\_point(aes(y = estimate)) +   
 geom\_errorbar(aes(ymin = conf.low, ymax = conf.high), width = .1) +  
 geom\_hline(yintercept = 0, color = "red")



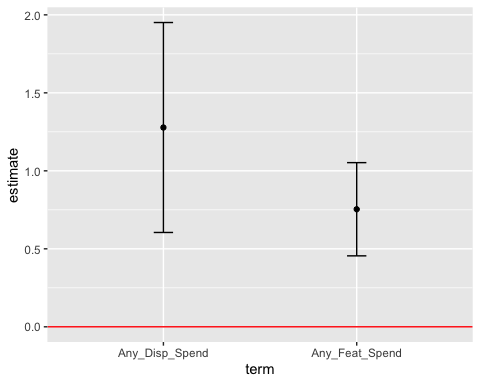
If we leave out display spend, we can see that feature spend is suddenly much more important, even greater than 1. Price decrease spend stays about the same.

# Model without feature spend.  
fit\_03 <- linear\_reg() |>   
 set\_engine(engine = "lm") |>   
 fit(  
 Sales ~ Any\_Disp\_Spend + Any\_Price\_Decr\_Spend,   
 data = soup\_data  
 )  
  
# Visualize slope parameter estimates.  
tidy(fit\_03, conf.int = TRUE) |>   
 filter(term != "(Intercept)") |>   
 ggplot(aes(x = term)) +   
 geom\_point(aes(y = estimate)) +   
 geom\_errorbar(aes(ymin = conf.low, ymax = conf.high), width = .1) +  
 geom\_hline(yintercept = 0, color = "red")



If we leave out feature spend, we can see that display spend is suddenly even *more* important, clearly greater than 1 and maybe even greater than 2. Price decrease spend again stays about the same.

# Model without price decrease spend.  
fit\_04 <- linear\_reg() |>   
 set\_engine(engine = "lm") |>   
 fit(  
 Sales ~ Any\_Disp\_Spend + Any\_Feat\_Spend,   
 data = soup\_data  
 )  
  
# Visualize slope parameter estimates.  
tidy(fit\_04, conf.int = TRUE) |>   
 filter(term != "(Intercept)") |>   
 ggplot(aes(x = term)) +   
 geom\_point(aes(y = estimate)) +   
 geom\_errorbar(aes(ymin = conf.low, ymax = conf.high), width = .1) +  
 geom\_hline(yintercept = 0, color = "red")



Without price decrease spend, feature spend shifts the most, with the confidence interval now including 1.

All of these comparisons demonstrate how sensitive the parameter estimates are to what is or isn’t included in the model. My understanding of sales says that all three of the explanatory variables should be included in the model. We can also compare the overall model fit to see which one is indeed the “best-fitting model.”

# Model comparison.  
tibble(  
 model = c(  
 "Full model",   
 "Model without display spend",   
 "Model without feature spend",   
 "Model without price decrease spend"  
 )  
) |>   
 bind\_cols(  
 bind\_rows(  
 glance(fit\_01),   
 glance(fit\_02),   
 glance(fit\_03),   
 glance(fit\_04)  
 )  
 ) |>   
 arrange(desc(r.squared))

# A tibble: 4 × 13  
 model r.squared adj.r.squared sigma statistic p.value df logLik AIC  
 <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
1 Full mod… 0.900 0.893 59734. 143. 5.77e-24 3 -644. 1297.  
2 Model wi… 0.884 0.879 63574. 187. 1.20e-23 2 -647. 1303.  
3 Model wi… 0.874 0.869 66149. 171. 8.41e-23 2 -649. 1307.  
4 Model wi… 0.863 0.857 69095. 154. 7.12e-22 2 -652. 1311.  
# ℹ 4 more variables: BIC <dbl>, deviance <dbl>, df.residual <int>, nobs <int>

The “full model” that includes all three of the explanatory variables fits best according to the R-squared overall fit statistic.

## Counterfactual Predictions

Let’s use the full model to predict Sales. Note that we are assuming we have $10,000 to spend across all three of the explanatory variables.

# Create new data representing various counterfactual scenarios for spending our budget.  
scenarios <- tibble(  
 Any\_Disp\_Spend = c(10000, 0, 0, 10000/3),  
 Any\_Feat\_Spend = c(0, 10000, 0, 10000/3),  
 Any\_Price\_Decr\_Spend = c(0, 0, 10000, 10000/3)  
)  
  
# Predict sales using fit\_01.  
predict(fit\_01, new\_data = scenarios) |>   
 bind\_cols(  
 predict(fit\_01, new\_data = scenarios, type = "pred\_int"),  
 scenarios  
 ) |>   
 arrange(desc(.pred))

# A tibble: 4 × 6  
 .pred .pred\_lower .pred\_upper Any\_Disp\_Spend Any\_Feat\_Spend  
 <dbl> <dbl> <dbl> <dbl> <dbl>  
1 263026. 140634. 385418. 10000 0   
2 257132. 134904. 379361. 3333. 3333.  
3 254773. 132494. 377052. 0 10000   
4 253597. 131395. 375800. 0 0   
# ℹ 1 more variable: Any\_Price\_Decr\_Spend <dbl>

Based on our model, the best thing Campbell’s can do in this trade area is put the entire promotional spend budget in displays. We inferred this from it being the most important of the explanatory variables, but now we can directly see that spending $10,000 on display should produce sales of about $263,026 dollars. Or, even better, the prediction says such a display spend should produce sales anywhere from $140,634 to $385,418, accounting for the uncertainty in our parameter estimates.