Romero_Lab2

Guillermo Romero

2023-01-18

```r
# Source from lab 1
sourceDir <- "/Users/guillermoromero/Documents/MEDS/eds232_ml/Romero_Lab1.Rmd"
library(knitr)
source(knitr::purl(sourceDir, quiet=TRUE))
```

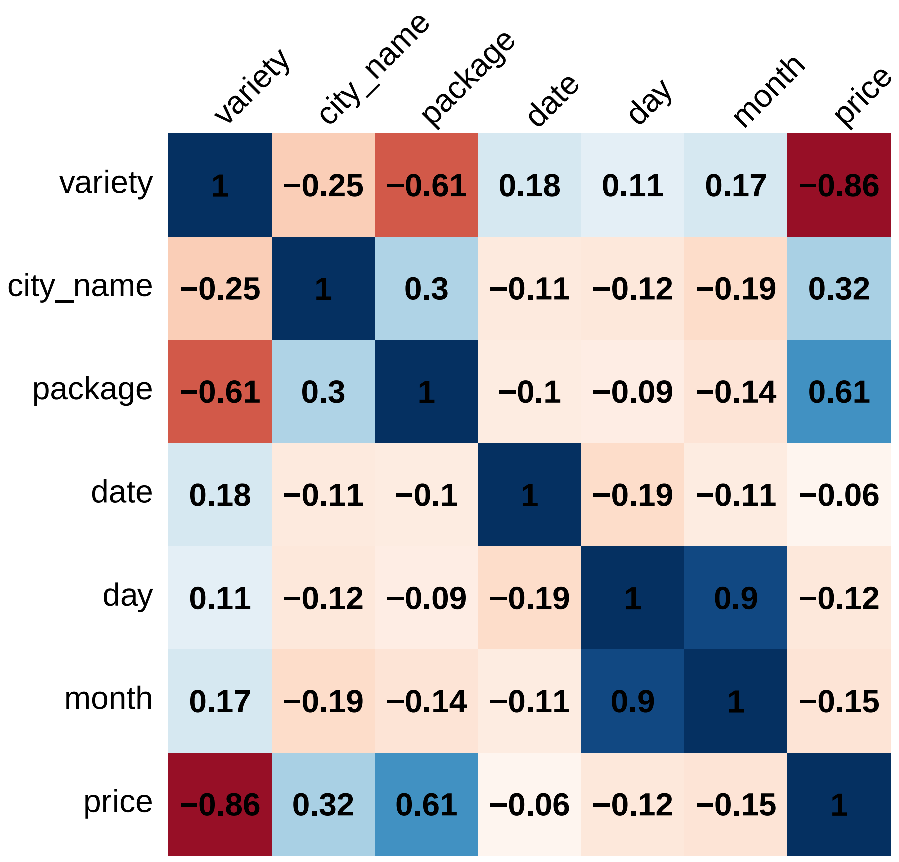
```
## -- Attaching packages -------------------------------------- tidymodels 1.0.0 --

## v broom        1.0.2      v recipes      1.0.4
## v dials        1.1.0      v rsample      1.1.1
## v dplyr        1.0.10     v tibble       3.1.8
## v ggplot2      3.4.0      v tidyr        1.2.1
## v infer        1.0.4      v tune         1.0.1
## v modeldata    1.0.1      v workflows    1.1.2
## v parsnip      1.0.3      v workflowsets 1.0.0
## v purrr        1.0.1      v yardstick    1.1.0

## -- Conflicts ----------------------------------------- tidymodels_conflicts() --
## x purrr::discard() masks scales::discard()
## x dplyr::filter()  masks stats::filter()
## x dplyr::lag()     masks stats::lag()
## x recipes::step()  masks stats::step()
## * Use tidymodels_prefer() to resolve common conflicts.

## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v readr   2.1.3     v forcats 0.5.2
## v stringr 1.5.0
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x readr::col_factor() masks scales::col_factor()
## x purrr::discard()    masks scales::discard()
## x dplyr::filter()     masks stats::filter()
## x stringr::fixed()    masks recipes::fixed()
## x dplyr::lag()        masks stats::lag()
## x readr::spec()       masks yardstick::spec()
##
## Attaching package: 'janitor'
##
##
## The following objects are masked from 'package:stats':
##
##     chisq.test, fisher.test
##
```

```
##
## corrplot 0.92 loaded
##
## New names:
## Rows: 1757 Columns: 27
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (13): City Name, Type, Package, Variety, Sub Variety, Date, Origin, Orig...
## dbl  (5): ...1, Low Price, High Price, Mostly Low, Mostly High
## lgl  (9): Grade, Environment, Quality, Condition, Appearance, Storage, Crop,...
##
## 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.
```

```
## Rows: 1,757
## Columns: 27
## $ ...1              <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 1~
## $ `City Name`       <chr> "BALTIMORE", "BALTIMORE", "BALTIMORE", "BALTIMORE", ~
## $ Type              <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ Package           <chr> "24 inch bins", "24 inch bins", "24 inch bins", "24 ~
## $ Variety           <chr> NA, NA, "HOWDEN TYPE", "HOWDEN TYPE", "HOWDEN TYPE",~
## $ `Sub Variety`     <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ Grade             <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ Date              <chr> "4/29/17", "5/6/17", "9/24/16", "9/24/16", "11/5/16"~
## $ `Low Price`       <dbl> 270, 270, 160, 160, 90, 90, 160, 160, 160, 160, 160,~
## $ `High Price`      <dbl> 280, 280, 160, 160, 100, 100, 170, 160, 170, 160, 17~
## $ `Mostly Low`      <dbl> 270, 270, 160, 160, 90, 90, 160, 160, 160, 160, 160,~
## $ `Mostly High`     <dbl> 280, 280, 160, 160, 100, 100, 170, 160, 170, 160, 17~
## $ Origin            <chr> "MARYLAND", "MARYLAND", "DELAWARE", "VIRGINIA", "MAR~
## $ `Origin District` <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ `Item Size`       <chr> "lge", "lge", "med", "med", "lge", "lge", "med", "lg~
## $ Color             <chr> NA, NA, "ORANGE", "ORANGE", "ORANGE", "ORANGE", "ORA~
## $ Environment       <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ `Unit of Sale`    <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ Quality           <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ Condition         <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ Appearance        <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ Storage           <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ Crop              <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ Repack            <chr> "E", "E", "N", "N", "N", "N", "N", "N", "N", "N", "N~
## $ `Trans Mode`      <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ ...26             <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ ...27             <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
```

```
## Loading required package: timechange
##
## Attaching package: 'lubridate'
##
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` instead.
```



Today we will be continuing the pumpkin case study from last week. We will be using the data that you cleaned and split last time (pumpkins_train) and will be comparing our results today to those you have already obtained, so open and run your Lab 1 .Rmd as a first step so those objects are available in your Environment (unless you created an R Project last time, in which case, kudos to you!).

Once you have done that, we'll start today's lab by specifying a recipe for a polynomial model. First we specify a recipe that identifies our variables and data, converts package to a numerical form, and then add a polynomial effect with step_poly()

```
# Specify a recipe
poly_pumpkins_recipe <-
  recipe(price ~ package, data = pumpkins_train) %>%
  step_integer(all_predictors(), zero_based = TRUE) %>%
  step_poly(all_predictors(), degree = 4)
```

How did that work? Choose another value for degree if you need to. Later we will learn about model tuning that will let us do things like find the optimal value for degree. For now, we'd like to have a flexible model, so find the highest value for degree that is consistent with our data.

Polynomial regression is still linear regression, so our model specification looks similar to before.

```
# Create a model specification called poly_spec
poly_spec <- linear_reg() %>%
  set_engine("lm") %>%
  set_mode("regression")
```

Question 1: Now take the recipe and model specification that just created and bundle them into a workflow called poly_df.

```r
# Bundle recipe and model spec into a workflow
poly_wf <- workflow() %>%
  add_recipe(poly_pumpkins_recipe) %>%
  add_model(poly_spec)
```

Question 2: fit a model to the pumpkins_train data using your workflow and assign it to poly_wf_fit

```r
# Create a model
poly_wf_fit <- fit(poly_wf, data = pumpkins_train)
```

```r
# Print learned model coefficients
poly_wf_fit
```

```
## == Workflow [trained] ==========================================================
## Preprocessor: Recipe
## Model: linear_reg()
##
## -- Preprocessor ----------------------------------------------------------------
## 2 Recipe Steps
##
## * step_integer()
## * step_poly()
##
## -- Model -----------------------------------------------------------------------
##
## Call:
## stats::lm(formula = ..y ~ ., data = data)
##
## Coefficients:
##    (Intercept)  package_poly_1  package_poly_2  package_poly_3  package_poly_4
##        27.9706        103.8566       -110.9068        -62.6442          0.2677
```

```r
# Make price predictions on test data
poly_results <- poly_wf_fit %>% predict(new_data = pumpkins_test) %>%
  bind_cols(pumpkins_test %>% select(c(package, price))) %>%
  relocate(.pred, .after = last_col())

# Print the results
poly_results %>%
  slice_head(n = 10)
```

```
## # A tibble: 10 x 3
##    package              price .pred
##    <chr>                <dbl> <dbl>
##  1 1 1/9 bushel cartons  13.6  15.9
##  2 1 1/9 bushel cartons  16.4  15.9
##  3 1 1/9 bushel cartons  16.4  15.9
##  4 1 1/9 bushel cartons  13.6  15.9
##  5 1 1/9 bushel cartons  15.5  15.9
##  6 1 1/9 bushel cartons  16.4  15.9
##  7 1/2 bushel cartons    34    34.4
##  8 1/2 bushel cartons    30    34.4
##  9 1/2 bushel cartons    30    34.4
## 10 1/2 bushel cartons    34    34.4
```

Now let's evaluate how the model performed on the test_set using yardstick::metrics().

```
metrics(data = poly_results, truth = price, estimate = .pred)
```

```
## # A tibble: 3 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard       3.27
## 2 rsq     standard       0.892
## 3 mae     standard       2.35
```

Question 3: How do the performance metrics differ between the linear model from last week and the polynomial model we fit today? Which model performs better on predicting the price of different packages of pumpkins?

The performance metrics are all much lower for all the polynomial fit when compared to the linear model from last week. The root mean squared was 7.23, the average price by which the predictions are wrong and lower at 3.27 this week. The polynomial fit model performs better on predicting the price of different packages of pumpkins.

Let's visualize our model results. First prep the results by binding the encoded package variable to them.

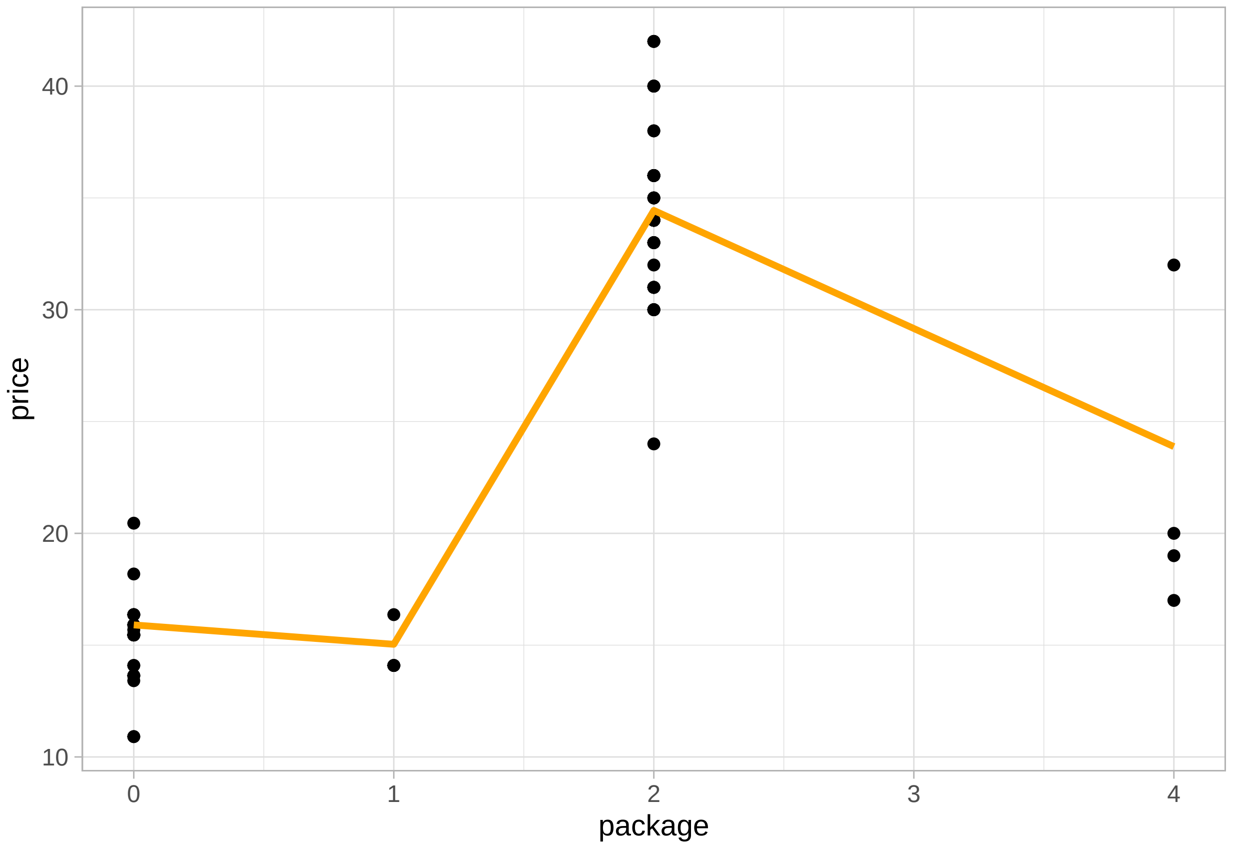
```
# Bind encoded package column to the results
poly_results <- poly_results %>%
  bind_cols(package_encode %>%
              rename(package_integer = package)) %>%
  relocate(package_integer, .after = package)


# Print new results data frame
poly_results %>%
  slice_head(n = 5)
```

```
## # A tibble: 5 x 4
##   package              package_integer price .pred
##   <chr>                          <int> <dbl> <dbl>
## 1 1 1/9 bushel cartons               0  13.6  15.9
## 2 1 1/9 bushel cartons               0  16.4  15.9
## 3 1 1/9 bushel cartons               0  16.4  15.9
## 4 1 1/9 bushel cartons               0  13.6  15.9
## 5 1 1/9 bushel cartons               0  15.5  15.9
```

OK, now let's take a look!

Question 4: Create a scatter plot that takes the poly_results and plots package vs. price. Then draw a line showing our model's predicted values (.pred). Hint: you'll need separate geoms for the data points and the prediction line.

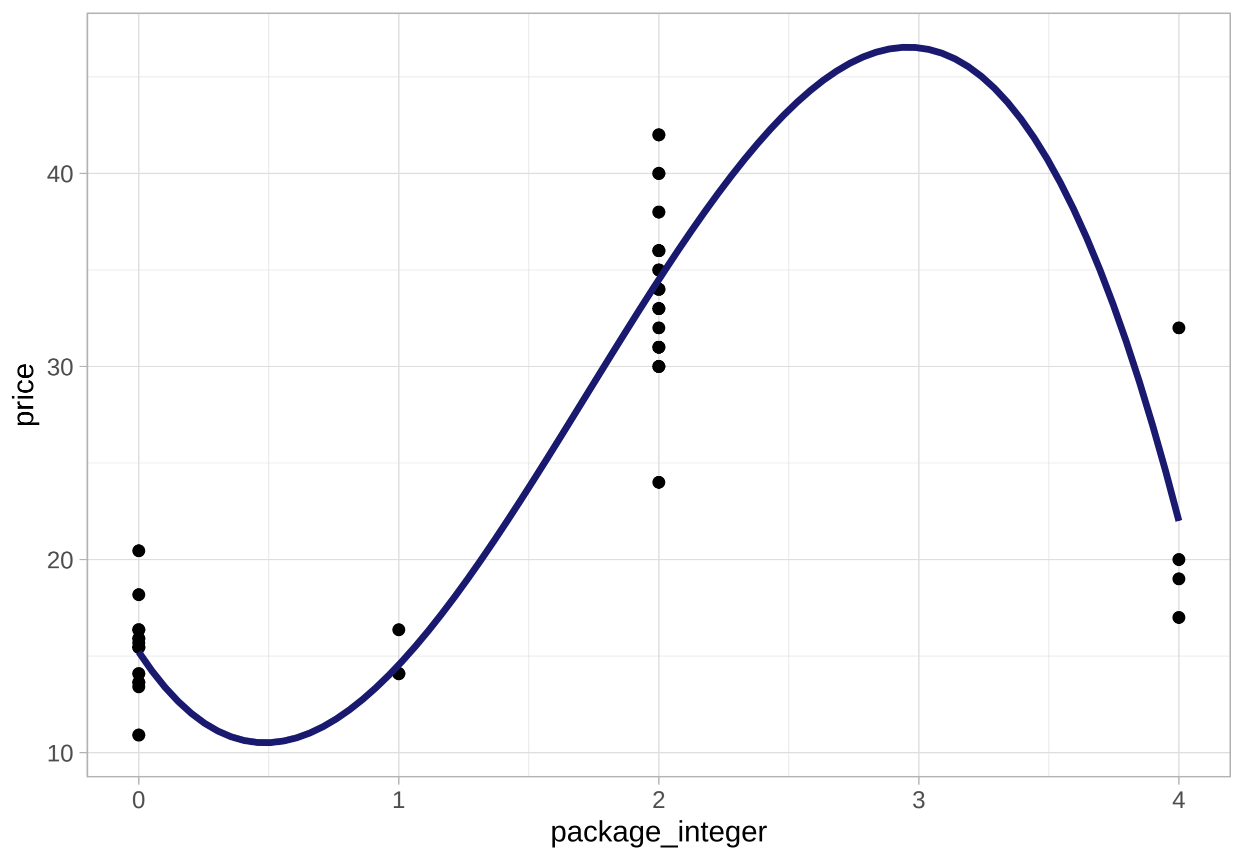
```
# Make a scatter plot
poly_results %>%
  ggplot(mapping = aes(x = package_integer, y = price)) +
   geom_point(size = 1.6) +
   # Overlay a line of best fit
   geom_line(aes(y = .pred), color = "orange", size = 1.2) +
   xlab("package")
```



You can see that a curved line fits your data much better.

Question 5: Now make a smoother line by using geom_smooth instead of geom_line and passing it a polynomial formula like this: geom_smooth(method = lm, formula = y ~ poly(x, degree = 3), color = "midnightblue", size = 1.2, se = FALSE)

```
# Make a smoother scatter plot
poly_results %>%
  ggplot(mapping = aes(x = package_integer, y = price)) +
   geom_point(size = 1.6) +
  geom_smooth(method = lm, formula = y ~ poly(x, degree = 3), color = "midnightblue", size = 1.2, se = F
```



OK, now it's your turn to go through the process one more time.

Additional assignment components :

6. Choose a new predictor variable (anything not involving package type) in this dataset. 7. Determine its correlation with the outcome variable (price). (Remember we calculated a correlation matrix last week)

```
cor(baked_pumpkins$day, baked_pumpkins$price)
```

```
## [1] -0.1245279
```

8. Create and test a model for your new predictor: - Create a recipe - Build a model specification (linear or polynomial) - Bundle the recipe and model specification into a workflow - Create a model by fitting the workflow - Evaluate model performance on the test data - Create a visualization of model performance

```
#Create a recipe
poly_recipe_day <-
  recipe(price ~ day, data = pumpkins_train) %>%
  step_integer(all_predictors(), zero_based = TRUE) %>%
  step_poly(all_predictors(), degree = 4)
```

```
# Create a model specification
poly_spec_day <- linear_reg() %>%
  set_engine("lm") %>%
  set_mode("regression")
```

```r
# Bundle recipe and model spec into a workflow
poly_wf_day <- workflow() %>%
  add_recipe(poly_recipe_day) %>%
  add_model(poly_spec_day)
poly_wf_day
```

```
## == Workflow ====================================================================
## Preprocessor: Recipe
## Model: linear_reg()
##
## -- Preprocessor ----------------------------------------------------------------
## 2 Recipe Steps
##
## * step_integer()
## * step_poly()
##
## -- Model -----------------------------------------------------------------------
## Linear Regression Model Specification (regression)
##
## Computational engine: lm
```

```r
# Create a model by fitting the workflow
poly_fit_day <- fit(poly_wf_day, data = pumpkins_train)
```

```r
# Make price predictions on test data
poly_results_day <- poly_fit_day %>% predict(new_data = pumpkins_test) %>%
  bind_cols(pumpkins_test %>% select(c(day, price))) %>%
  relocate(.pred, .after = last_col())

# Print new results data frame
poly_results_day %>%
  slice_head(n = 5)
```

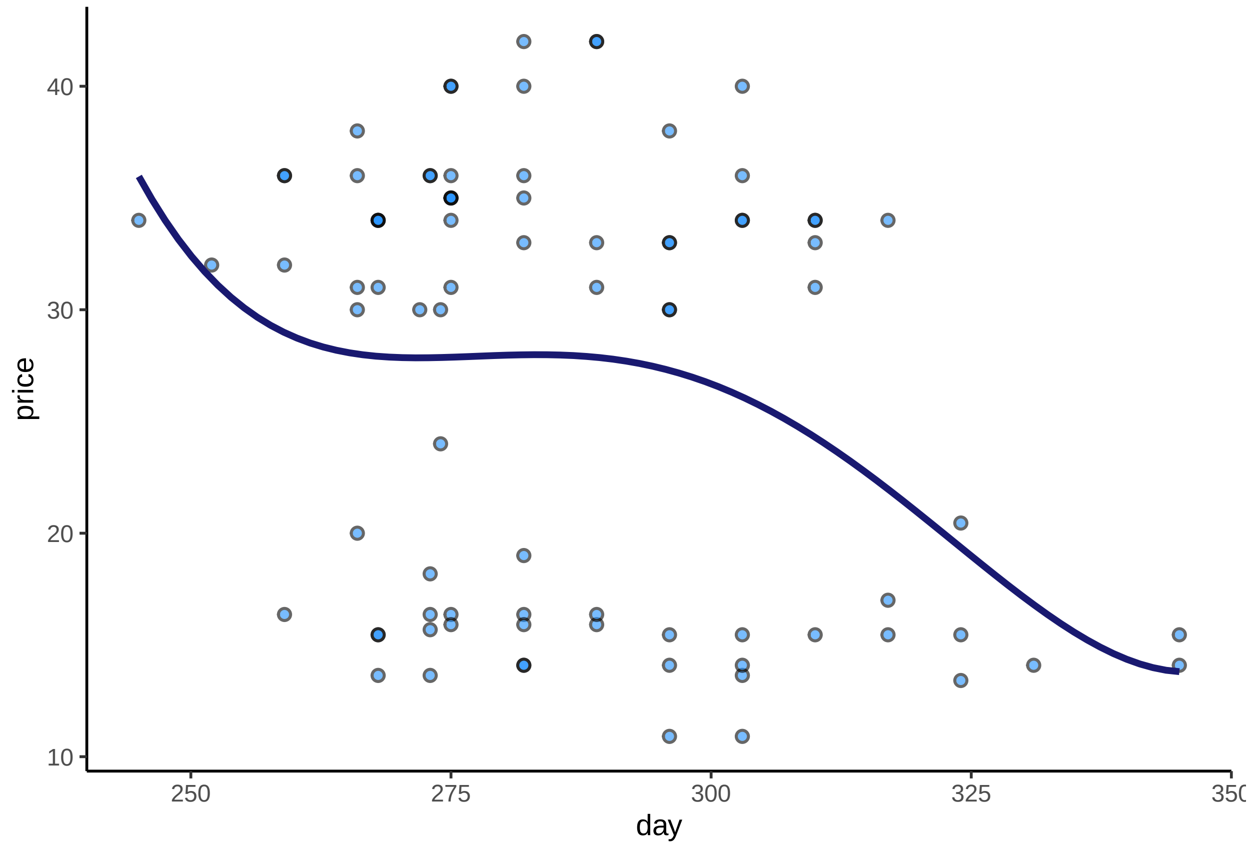
```
## # A tibble: 5 x 3
##     day price .pred
##   <dbl> <dbl> <dbl>
## 1   268  13.6  29.1
## 2   275  16.4  28.9
## 3   282  16.4  29.0
## 4   303  13.6  28.2
## 5   303  15.5  28.2
```

```r
#Evaluate model performance on the test data
metrics(data = poly_results_day, truth = price, estimate = .pred)
```

```
## # A tibble: 3 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard      9.53
## 2 rsq     standard      0.0779
## 3 mae     standard      8.47
```

The model performance metrics are all worse, the rmse using the predictor 'variety' with a polynomial fit is higher than that of package type 9.53 vs. 3.27 and is worse than the linear regression from last week.

```
# Create a visualization of model performance

poly_results_day %>%
  ggplot(mapping = aes(x = day, y = price)) +
   geom_point(size = 1.6, colour = 'black', alpha = 0.6,
              stroke = .8,fill = 'dodgerblue',
              shape = 21) +
  geom_smooth(method = lm, formula = y ~ poly(x, degree = 4), color = "midnightblue", size = 1.2, se = F
  theme_classic()
```



Lab 2 due 1/24 at 11:59 PM