The R Companion to STATS2

Will Hopper

08/18/2020

Table of contents

Preface

This is a guide to conducting the analysis demonstrated in Stat2: Modeling with Regression and ANOVA (Cannon et al. 2018) using R, with a focus on using tools and techniques from the Tidyverse.

Part I

Unit A: Linear Regression

1 Simple Linear Regression

Linear regression is introduced in Chapter 1 with the motivating question:

> How much should you expect to pay for a used Honda Accord, if you know how many miles the car has already been driven?

and introduces the `AccordPrice` data set, which contains information about the list price and mileage for a sample of 30 Honda Accords. The `AccordPrice` data set is included with the `Stat2Data` R package, so to access the data for yourself, you'll need to install the package. If you don't already know how to install R packages, here are two good resources to walk you through the process:

- Reading: ModernDive Chapter 1.3.1: Installing Packages
- Watching: How to Install Packages in R Studio on YouTube

Once you have the package installed, load the package into your R session using:

```
library(Stat2Data)
```

To load the `AccordPrice` data set into your R environment, use the command:

```
data("AccordPrice")
```

```
   Age Price Mileage
1    7  12.0    74.9
2    4  17.9    53.0
3    4  15.7    79.1
4    7  12.5    50.1
5    9   9.5    62.0
6    1  21.5     4.8
7   18   3.5    89.4
8    2  22.8    20.8
9    2  26.8     4.8
10   5  13.6    48.3
11   2  19.4    46.5
12   2  19.5     3.0
```

```
13   6   9.0    64.1
14   3  17.4     8.3
15   3  17.8    27.1
16   2  17.5    20.3
17   4  13.5    68.4
18  14   7.0    86.9
19   9  11.6    64.5
20  10   7.9   150.5
21   5  11.7    65.2
22   3  15.6    56.1
23  12   5.0   139.4
24   3  21.0    13.9
25   4  15.6    18.6
26   2  17.0    15.7
27   3  16.0    38.5
28   3  17.6    19.8
29  11   6.9   119.3
30  13   5.5   122.5
```

As a side note: not much information is given in the text about how this sample of 30 Accords was collected, but we can gather a bit more information by looking at the help page for the `AccordPrice` data set. To open the help page for the `AccordPrice` data set, you can run the command

```
?AccordPrice
```

in the R console. By reading the "Details" and "Source" sections, we can learn that these 30 Accords were listed for sale on Cars.com in Lakewood Ohio during February 2017. Whenever you want to to know more about one of the textbook's data sets, the help page for that data set is a good place to look first. Sometimes there's not much more information than given in the textbook, but every little bit helps!

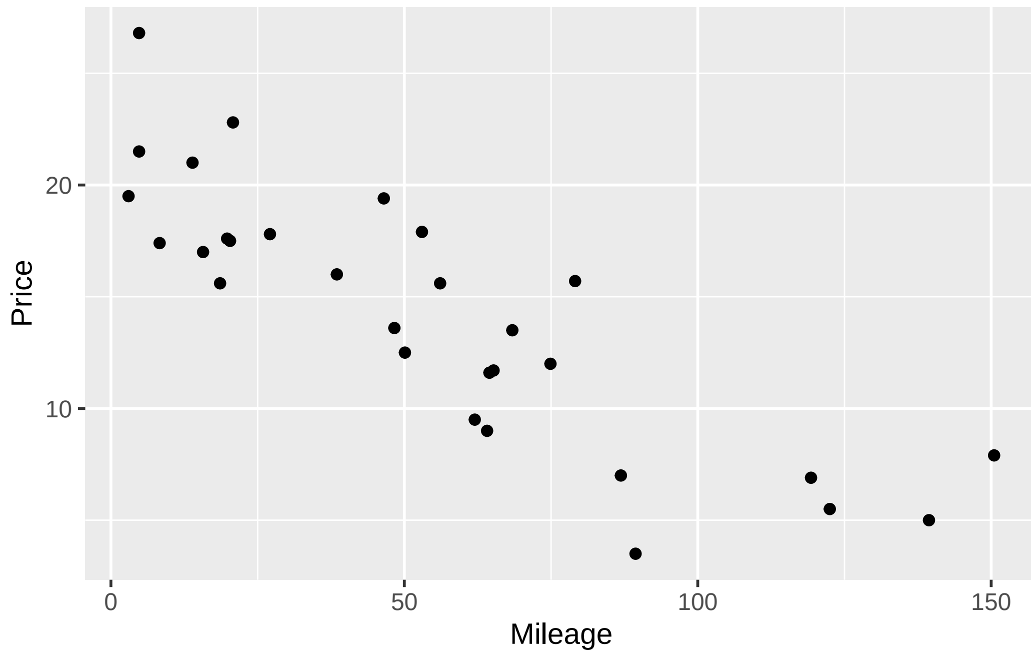
Figure 1.2 displays a scatter plot of the `Mileage` and `Price` variable, showing how those variables relate to one another. To re-produce this scatter plot, we'll use the `ggplot2` R package (Wickham 2016). If you're not already familiar with the `ggplot2` package, here are a few good resources to help you get started:

- Reading: ModernDive Chapter 2: Data Visualization
- Reading: Effective data visualization
- Watching: ggplot for plots and graphs on YouTube

To re-create this scatter plot, we'll map the `Mileage` variable to x-axis aesthetic, and the `Price` variable to the y-axis aesthetic, and draw a layer of points to represent each of the 30 cars using `geom_point()`

```r
library(ggplot2)

ggplot(data = AccordPrice,
       mapping = aes(x=Mileage, y=Price)
       ) +
  geom_point()
```



Aside

If you want to **exactly** reproduce the scatter plots in STAT2, right down to the colors, backgrounds, and fonts, you can use the following ggplot2 theme:

```
theme_stat2 <- function(base_size = 11,
                        base_family = "",
                        base_line_size = base_size/22,
                        base_rect_size = base_size/22) {

  theme_bw() %+replace%
  theme(axis.text.x = element_text(color="black"),
        axis.text.y = element_text(color="black"),
        panel.border = element_blank(),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.background=element_rect(colour="black"),
        complete = TRUE
        )
}
```

And use the hex color code #92278f for your geometric objects. For example, this could **exactly** reproduce Figure 1.2 by adapting the code above to use this new theme:

```
ggplot(data = AccordPrice,
       mapping = aes(x=Mileage, y=Price)
       ) +
  geom_point(color="#92278f") +
  theme_stat2()
```

In the rest of this book, we *won't* use the STAT2 theme for our visualizations, but provide it here for completeness.

1.1 Modeling the `Mileage` vs. `Price` relationsip

Example 1.3 shows a summary of a simple linear regression model fit to the `Mileage` and `Price` variable in the `AccordPrices` data set. This summary is actually a mix of two different summaries, a regression table and an Analysis of Variance (ANOVA) table. Reproducing this summary will be a 3 step process in R:

1. Fitting the model using the `lm()` function
2. Printing the regression table with the `summary()` function
3. Printing the ANOVA table with the `anova()` function

1.1.1 Fitting a simple linear regression model

The `lm()` function (short for **l**inear **m**odel) does the "heavy lifting' of estimating the coefficients of the simple linear model. In other words, the `lm()` function find the optimal values for $\widehat{\beta}_0$ and $\widehat{\beta}_1$ in the model $Price = \widehat{\beta}_0 + \widehat{\beta}_1 \cdot Mileage + \epsilon$.

To fit a linear regression model using `lm`, you need to supply:

1. A formula describing relationship between the outcome and explanatory variable(s)
2. The name of a data set where the outcome and explanatory variables can be found.

In this case, our call to the `lm` function would be:

```
price_mileage_model <- lm(Price ~ Mileage, data = AccordPrice)
```

The first argument inside the `lm()` function is the formula describing the structure of the model. In R, model formulas are always created using the `~` symbol, with the outcome variable named on the left, and the explanatory variables(s) named on the right. As you might notice, R's model formula code is an adaptation of how the model is described in mathematical notation.

Also, take note that we've saved the results from fitting this linear model in a new R object named `price_mileage_model`. We'll need to use this new object to produce the regression table and the ANOVA table in steps 2 and 3 below.

1.1.2 Reporting the regression table

In order to report the regression table, we need to call the `summary()` function on the linear model object we just created:

```
price_mileage_model <- lm(Price ~ Mileage, data = AccordPrice)
summary(price_mileage_model)
```

```

Call:
lm(formula = Price ~ Mileage, data = AccordPrice)

Residuals:
    Min      1Q  Median      3Q     Max 
-6.5984 -1.8169 -0.4148  1.4502  6.5655 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
```

```
(Intercept)  20.8096     0.9529   21.84  < 2e-16 ***
Mileage      -0.1198     0.0141   -8.50 3.06e-09 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.085 on 28 degrees of freedom
Multiple R-squared:  0.7207,    Adjusted R-squared:  0.7107 
F-statistic: 72.25 on 1 and 28 DF,  p-value: 3.055e-09
```

As we can see, the `summary()` function first prints out a few things *not* shown as part of the summary in the textbook: a copy of the code used to fit the model, and a the Five-number summary of the model's residual errors. These are followed by the regression table summarizing the intercept and slope, and a "goodness of fit" summary of the model as whole.

1.1.3 Reporting the ANOVA table

The ANOVA table is found by calling the aptly named `anova()` function on the linear model, the same way we just did with the summary() function a moment ago:

```r
price_mileage_model <- lm(Price ~ Mileage, data = AccordPrice)
anova(price_mileage_model)
```

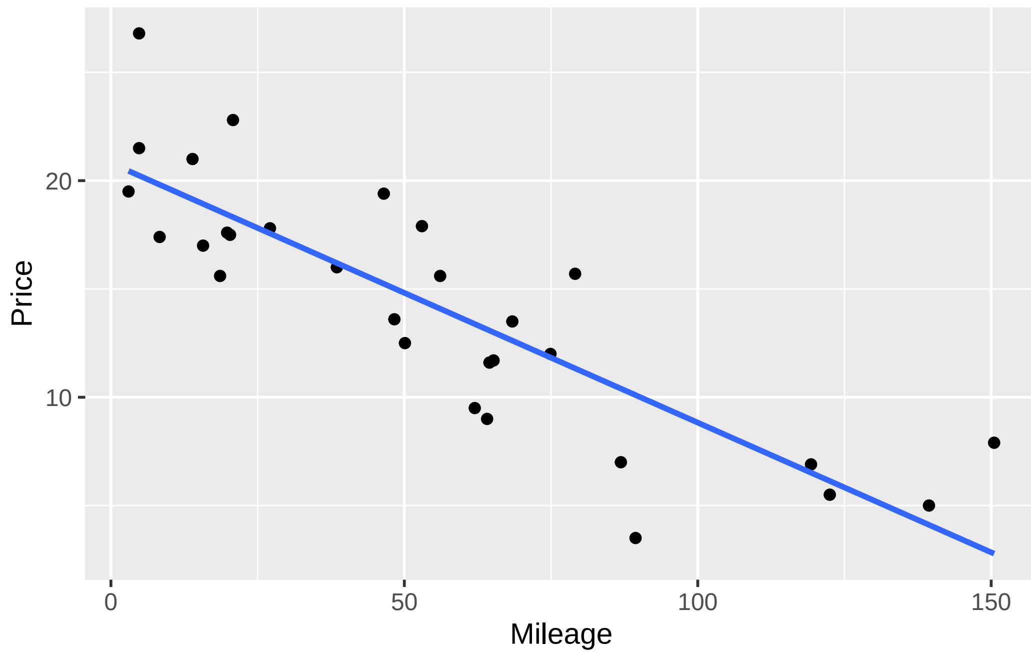
```
Analysis of Variance Table

Response: Price
          Df Sum Sq Mean Sq F value    Pr(>F)    
Mileage    1 687.66  687.66  72.253 3.055e-09 ***
Residuals 28 266.49    9.52                      
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

1.2 Adding the regression line to a scatterplot

Figure 1.3 shows the Price vs. Mileage scatter plot again, but this time with a line representing the regression model's predictions drawn on top of the raw data. Surprisingly, the easiest method for visualizing the predictions of a regression model **doesn't** involve the fitted model object. Instead, we will begin with the same ggplot code we used to draw the Mileage vs. Price scatter plot earlier, and add to it. The `geom_smooth()` function is used to draw the regression line on top of the raw data:

```
ggplot(data = AccordPrice,
       mapping = aes(x = Mileage, y = Price)
       ) +
  geom_point() +
  geom_smooth(method = lm, se = FALSE, formula = y~x)
```



`geom_smooth()` is a generic smoothing function: the key argument that tells it to fit and display a linear regression model is the `method = lm` argument. Without the `method=lm` argument, `geom_smooth()` will not display a linear model.

The `se = FALSE` argument is included to stop ggplot from drawing confidence interval bands around the regression line. And, the `formula = y~x` argument is included simply to prevent ggplot from printing an annoying message that says `geom_smooth()` using formula 'y ~ x' when creating the plot.

1.3 Centering the Mileage Variable

Example 1.4 demonstrates how centering a variable (i.e., shifting all the values left or right by a single chosen number) changes the interpretation of the intercept coefficient, but not the slope coefficient. In this example, the Mileage variable is shifted to the left by 50; in other words, 50 is subtracted from all the Mileage values *before* fitting the model.

The easiest way to replicate this model is create a new variable in the `AccordPrices` data set which holds the centered Mileage values. To make this new column, we'll use the `mutate` function from the `dplyr` package (Wickham et al. 2022). If you aren't familiar with the `mutate()` function or the `dplyr` package, here are a few good resources to investigate:

- Reading: ModernDive Chapter 3: Data Wrangling
- Reading: Cleaning and Wrangling Data
- Watching: Dplyr Essentials on YouTube

In this case, the 'mutation' we apply is quite simple: we just use the subtraction operator to subtract 50, and R automatically applies this subtraction to all 30 values in the Mileage column.

```r
library(dplyr)

AccordPrice <- AccordPrice |>
  mutate(Mileage_c50 = Mileage - 50)
AccordPrice
```

```
   Age Price Mileage Mileage_c50
1    7  12.0    74.9        24.9
2    4  17.9    53.0         3.0
3    4  15.7    79.1        29.1
4    7  12.5    50.1         0.1
5    9   9.5    62.0        12.0
6    1  21.5     4.8       -45.2
7   18   3.5    89.4        39.4
8    2  22.8    20.8       -29.2
9    2  26.8     4.8       -45.2
10   5  13.6    48.3        -1.7
11   2  19.4    46.5        -3.5
12   2  19.5     3.0       -47.0
13   6   9.0    64.1        14.1
14   3  17.4     8.3       -41.7
15   3  17.8    27.1       -22.9
16   2  17.5    20.3       -29.7
17   4  13.5    68.4        18.4
18  14   7.0    86.9        36.9
19   9  11.6    64.5        14.5
20  10   7.9   150.5       100.5
21   5  11.7    65.2        15.2
22   3  15.6    56.1         6.1
```

```
23  12   5.0   139.4          89.4
24   3  21.0    13.9         -36.1
25   4  15.6    18.6         -31.4
26   2  17.0    15.7         -34.3
27   3  16.0    38.5         -11.5
28   3  17.6    19.8         -30.2
29  11   6.9   119.3          69.3
30  13   5.5   122.5          72.5
```

Note that we saved our centered mileage scores in a variable named `Mileage_c50`, to help us keep track of what these values mean: they are mileage values that have been centered by 50.

From here, we just need to fit another linear model with `lm()`, using our new `Mileage_c50` variable as the explanatory variable in our model formula:

```r
centered_mileage_model <- lm(Price ~ Mileage_c50, data = AccordPrice)
```

The textbook only presents the fitted model equation (not the full regression table) in order to show the intercept and slope coefficients. If you ever need **just** the coefficient values, without the rest of the summaries in the regression table, you can use the `coef()` function on your model object to print them out:

```r
centered_mileage_model <- lm(Price ~ Mileage_c50, data = AccordPrice)
coef(centered_mileage_model)
```

```
(Intercept) Mileage_c50 
 14.8190154  -0.1198119 
```

1.4 Displaying the fitted model equation

If you are using a literate programming environment (like an RMarkdown or Quarto document), you might find yourself wanting to display the fitted model equation in your document, formatted like a "fancy" mathematical equation. You could always write the LaTeX markup you need yourself, but the equatiomatic package (Anderson, Heiss, and Sumners 2022) can automatically generate what you need, straight from the model object itself!

To demonstrate, let's display a formatted equation representing the fitted regression model based on the centered mileage scores by using the `extract_eq()` function on the model object.

$$\hat{\text{Price}} = 14.82 - 0.12(\text{Mileage\_c50})$$

```r
library(equatiomatic)

centered_mileage_model <- lm(Price ~ Mileage_c50, data = AccordPrice)
```

Warning

As the time of writing, there are problems with using the equatiomatic package to display equations when rendering Quarto documents to PDF. Thankfully, there is a workaround that is not too difficult, which involves saving the equation as a variable, and `cat()`-ing the equation yourself:

````
```{r}
#| results: asis
eq <- extract_eq(centered_mileage_model, use_coefs = TRUE)
cat("$$", eq, "$$", sep = "\n")
````

Just be sure to set the `results: asis` chunk option!
````

References

Anderson, Daniel, Andrew Heiss, and Jay Sumners. 2022. *Equatiomatic: Transform Models into 'LaTeX' Equations*. https://CRAN.R-project.org/package=equatiomatic.

Cannon, A. R., G. W. Cobb, B. A. Hartlaub, J. M. Legler, R. H. Lock, T. L. Moore, A. J. Rossman, and J. A. Witmer. 2018. *Stat2: Modeling with Regression and ANOVA*. Macmillan Learning. https://www.macmillanlearning.com/college/us/product/STAT2/p/1319054072.

Wickham, Hadley. 2016. *Ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York. https://ggplot2.tidyverse.org.

Wickham, Hadley, Romain François, Lionel Henry, and Kirill Müller. 2022. *Dplyr: A Grammar of Data Manipulation*. https://CRAN.R-project.org/package=dplyr.