Primer on Simple Linear Regression

Table of contents

Introduction to Mathematical Modeling

Many research questions focus on understanding how changes in one variable, an outcome variable, are related to changes in one or more other variables, predictor variables. To answer these questions–even if just approximately–it is important to first specify the kind of relationship believed (or assumed) to exist between the outcome variable and the predictor variables. This is generally referred to as mathematical modeling or modeling, for short—you are modeling the relationship between an outcome variable and one or more predictor variables. As the researcher, you decide which form the relationship takes, but a very common form to select is the linear function:

$$Y = \beta_0 + \beta_1 X_1.$$

The linear function states that changes in the outcome variable, Y, are linearly related to changes in the predictor variable, X_1. The linear relationship between the two variables is described by the slope coefficient, β_1. The slope coefficient tells us the amount of change we expect to see in the outcome variable as we increase the predictor variable by one unit.

A very simple example of a linear relationship is the relationship between the amount of gas you put in your car and the amount you pay. If gas costs \$4 per gallon, then we can model the relationship as:

$$\text{Dollars Owed} = 0 + 4\frac{dollars}{gallon}\text{Gallon}.$$

Using the above function, we know that if we put 10 gallons of gas into our car, our cost will be 40 dollars. This also means that for every gallon of gas we put into our car (a single unit increase), our cost will increase by 4 dollars—at no point will an additional gallon of gas cost us more or less than 4 dollars. Further, the 0 coefficient, or regression intercept, in our model gives us additional information: how much our cost will be when we do not put in any gas, 0 dollars. In general, the regression intercept tells us the value of our outcome variable when the values of all of our predictor variables equal 0.

The cost of gas example is useful when it comes to explaining what a linear relationship is, but it is a bit misleading when it comes to the research questions you will be tackling. Specifically, the example above is a deterministic function—a function where there is no uncertainty in the relationship between the outcome variable and the predictor variables. In your own research, there is going to be a lot of uncertainty in the relationship between the outcome variable and the predictor variables. This is why statistical models, like linear regression, are so useful for researchers—statistical models allows us to model uncertainty, which is what we will discuss next, but first we need to revise our example and generate some data to model.

Introduction to Statistical Modeling

Statistical modeling, like mathematical modeling, involves specifying a relationship between an outcome variable and a set of predictor variables. Unlike mathematical modeling, however, statistical modeling also involves specifying a probability distribution that models the uncertainty in your data. Like choosing the form the relationship between your variables takes, you also have to choose the probability distribution that best describes the uncertainty in your data, and there are many different probability distributions from which to choose. More often than not, however, researchers choose to model the uncertainty in their data with a normal distribution.

To apply a statistical model to our earlier gas example, we need to revise the example by adding in some uncertainty. Pretend you live in a town with only two gas pumps: one that has been around for at least 10 years—the old pump—and one that was just built—the new pump. For both pumps, the cost per gallon of gas is still 4 dollars, but you believe the old pump is malfunctioning and will sometimes charge you a little under 4 dollars per gallon and sometimes it will charge you a little over. Importantly, you believe that these errors tend to be small and are roughly symmetric around a mean of 0 dollars per gallon. This means that small negative errors are as likely to occur as small positive errors and small errors, regardless of being positive or negative, are more likely to occur than large errors. Thus, you decide the errors occurring in the old pump are best modeled by a normal distribution.

Now that we have specified both the functional relationship between our variables and the probability distribution that generates the uncertainty in our data—a linear relationship and normal distribution, respectively—we can write down our statistical model in two equivalent ways. The first way places more emphasis on the functional form between the variables,

whereas the second way places more emphasis on the probability distribution that generates the uncertainty:

$$\text{Dollars Owed} = 0 + 4\frac{Dollars}{Gallon}\text{Gallon} + \epsilon, \epsilon|\text{Gallon} \sim N(0, \sigma)$$

$$\text{Dollars Owed}|\text{Gallon} \sim N(\mu = 0 + 4\frac{Dollars}{Gallon}\text{Gallon}, \sigma).$$

In general, I believe it is better to write down your statistical model using the second way as it more clearly states the assumptions you are making with your model:

1. The amount I pay at the pump is conditional (due to) the amount of gas I pump: Dollars Owed|Gallon.
2. On average, the amount I pay at the pump is equal to: $\mu = 0 + 4\frac{Dollars}{Gallon}\text{Gallon}$.
3. Conditional on how much gas I pump, the amount I actually pay should follow a normal distribution with a mean of $\mu = 0 + 4\frac{Dollars}{Gallon}\text{Gallon}$ and a standard deviation of σ.
4. The variation in cost, σ, is the same regardless of the amount of gas I pumped.

With our model specified, let us now generate some data to go along with our example.

Generating the Gas Pump Data

Building on our example, let us say that in this pretend town you have a friend who hates change and will only use the old pump, whereas you will only use the new pump because you believe the old pump is defective. To determine what is going on with the old pump, for the last 100 visits to the old pump, you have your friend record the gallons they pumped and the cost. You do the same for your last 100 visits to the new pump.

The code below generates gas pump data that would follow from our statistical model.

```
library(tibble)
library(dplyr)

set.seed(906)
n_visits <- 100 # Pump vista
pump_id <- c("old", "new") # Identify the pump used

# Simulate the gallons pumped to be only integer values between 4 and 15
# but we adjust the probabilities so that they mirror a normal dist.
gallons_pumped <- sample(
  4:15, n_visits*2, replace = TRUE,
  prob = dnorm(4:15, mean = 10, sd = 2)/sum(dnorm(4:15, mean = 10, sd = 2)))
```

```r
# Simulate the noise/uncertainty for the old pump so that it comes from
# a normal distribution with a sd of 1.50, which means that errors should
# mostly fall between -3*1.50 (reduce the cost by 4.50) and 3*1.50
# (increase cost by 4.50)
noise <- rnorm(n_visits, mean = 0, sd = 1.50)

# Use the first 100 observations in gallons_pumped to simulate the cost from
# the old pump. There are several things to notice:
# Cost/Gallon = $4 and we added the random noise
cost_old <- 4 * gallons_pumped[1:n_visits] + noise

# Use the last 100 observations in gallons_pumped to generate a deterministic
# function --- i.e. no noise is added
cost_new <- 4 * gallons_pumped[(n_visits + 1):(2 * n_visits)]

# Create a dataset that contains all of our simulated data
data_cost <-
  tibble::tibble(
    pump_id = rep(pump_id, each = n_visits),
    gallons_pumped = gallons_pumped,
    noise = c(noise, rep(0, n_visits)),
    cost = c(cost_old, cost_new)
  )
```

Exploring Your Data

Let us look at the data we generated and collected into a data frame object named: `data_cost`.

```r
data_cost
```

```
# A tibble: 200 x 4
   pump_id gallons_pumped   noise  cost
   <chr>            <int>   <dbl> <dbl>
 1 old                 12 -0.0393  48.0
 2 old                 13 -0.962   51.0
 3 old                  9 -1.16    34.8
 4 old                 10 -0.531   39.5
 5 old                 10 -1.66    38.3
 6 old                 12 -0.560   47.4
 7 old                 10  3.20    43.2
```

```
 8 old                  8  0.679   32.7
 9 old                 10 -2.70    37.3
10 old                  9  1.81    37.8
# i 190 more rows
```

We see four columns:

1. `pump_id`: Identifies the pump from which the gas was pumped.
2. `gallons_pumped`: The amount of gas pumped
3. `noise`: The size and direction of the error from the old pump.
4. `cost`: The dollar amount owed from the pump.

To get a better sense of our data, it is always useful to explore our data with plots. We will start by plotting the variables in our data frame separately and then, where it makes sense, together. To build our plots we will rely heavily on `ggplot2`.

First, we will look at the distribution of our cost in dollars variable, `cost`.

```
library(ggplot2)

ggplot2::ggplot(
  data = data_cost,
  ggplot2::aes(
    x = cost
  )
) +
  ggplot2::geom_histogram(
    fill = "lightblue",
    color = "black",
    binwidth = 4
  ) +
  ggplot2::labs(
    x = "Cost in Dollars",
    y = "Count"
  )
```

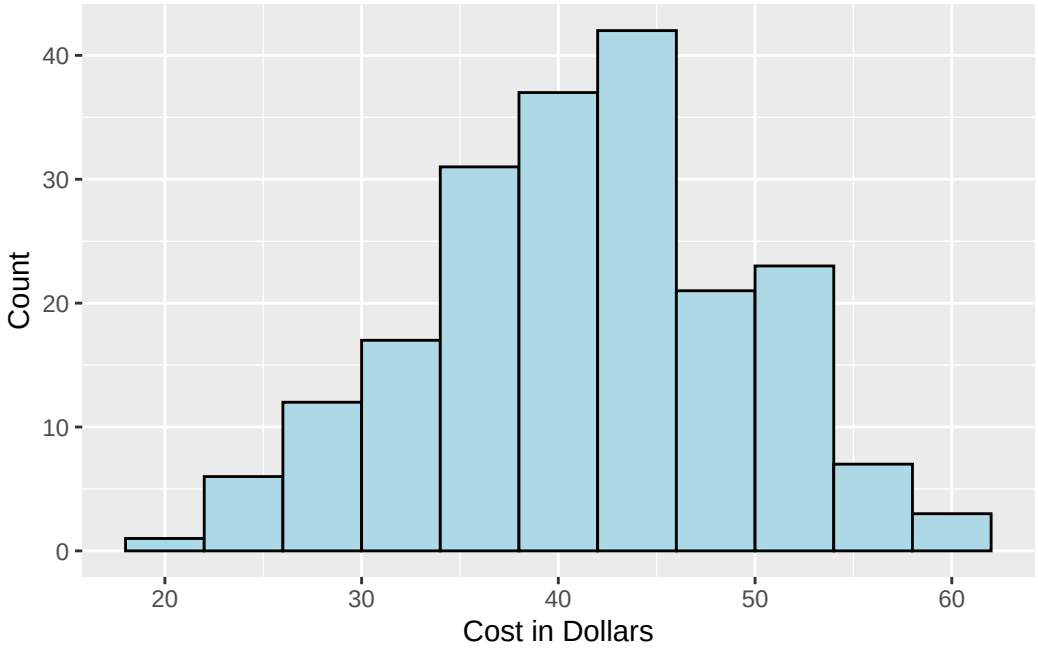

Figure 1: Distribution of Cost in Dollars

In Figure 1, you can see the distribution of cost in dollars across both pumps. The distribution looks roughly normal and symmetric around a mean of 41.27. Further, the lowest amount paid was 21.49 and the highest amount paid was 60.31.

Histograms like Figure 1 are a great plot to use when you want to visualize the distribution of your variables. Importantly, if you believe the distribution of your variable is better displayed for subsets of your data like by the pump being used, then you should view each distribution separately like in Figure 2.

```
ggplot2::ggplot(
  data = data_cost,
  ggplot2::aes(
    x = cost
  )
) +
  ggplot2::geom_histogram(
    fill = "lightblue",
    color = "black",
    binwidth = 4
  ) +
  ggplot2::facet_wrap(
```

```
    ~ pump_id
  ) +
  ggplot2::labs(
    x = "Cost in Dollars",
    y = "Count"
  )
```

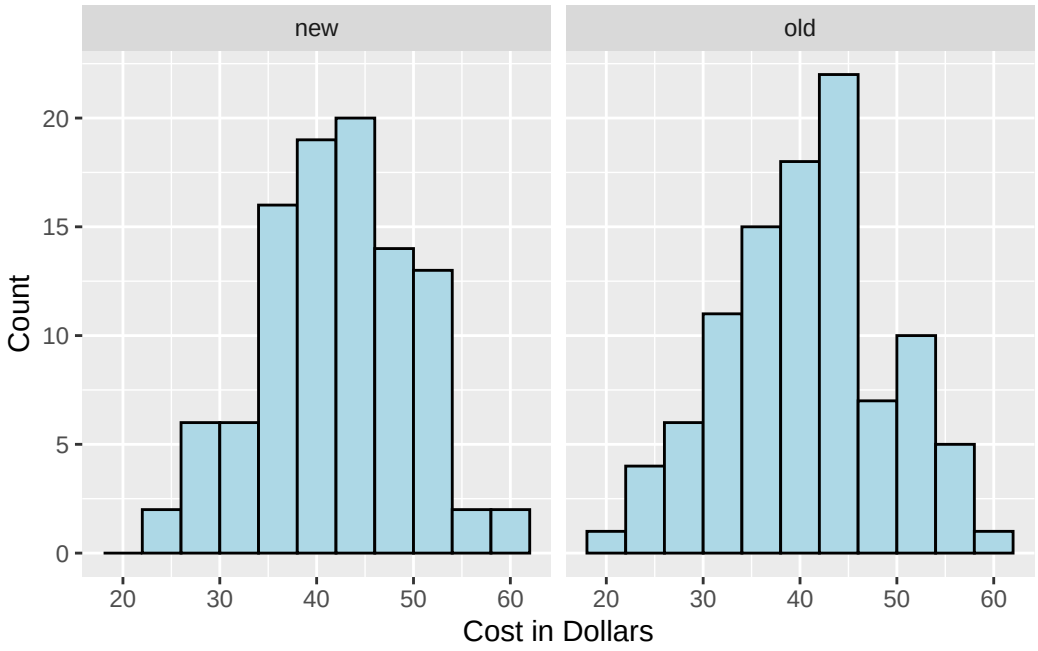

Figure 2: Distribution of Cost in Dollars by Pump

As we will talk about later on, the assumptions most statistical models make about the probability distribution that generates your data are focused on the outcome variable, cost in dollars in our example. So it is important that you plot the distribution of your outcome variable using an appropriate plot like a histogram. Although not as important, it is still useful to visualize the distributions of your predictor variables, which we do for gallons pumped in Figure 3.

```
ggplot2::ggplot(
  data = data_cost,
  ggplot2::aes(
    x = gallons_pumped
  )
) +
  ggplot2::geom_histogram(
```

```
    fill = "lightblue",
    color = "black",
    binwidth = 1
  ) +
  ggplot2::labs(
    x = "Gallons Pumped",
    y = "Count"
  )
```

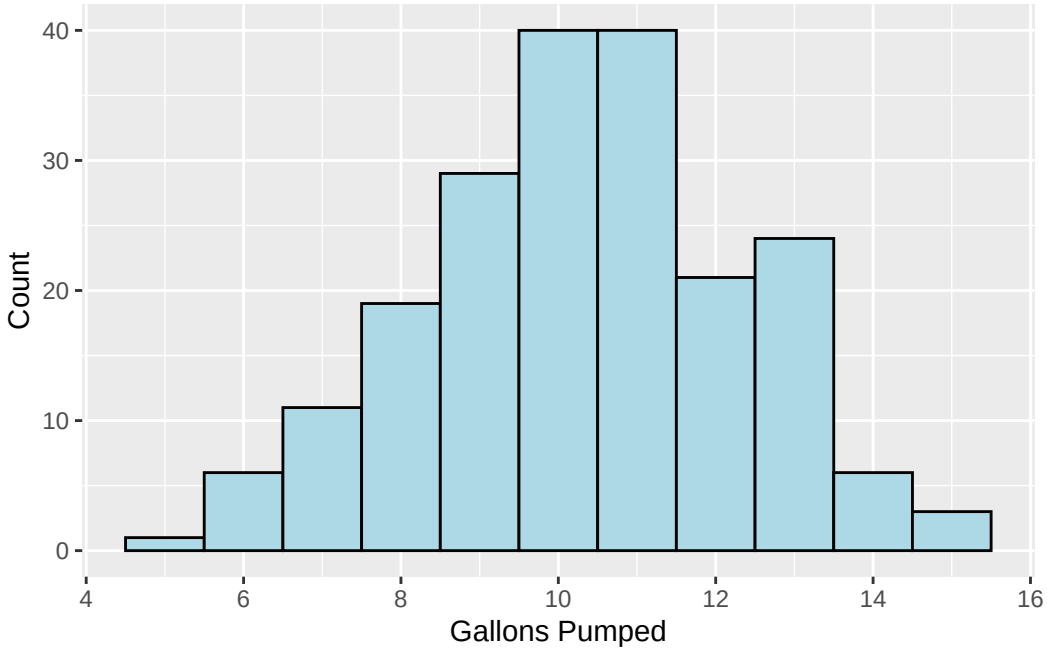

Figure 3: Distribution of Gallons Pumped

Because our `gallons_pumped` variable only takes on discrete values (by design of our simulation), we could also use a barplot to visualize its distribution (see Figure 4).

```
# First we need to restructure our data to work with
# a barplot.
data_barplot <-
  data_cost |>
  dplyr::select(
    gallons_pumped
  ) |>
  dplyr::summarize(
```

```
    count = dplyr::n(),
    .by = gallons_pumped
  ) |>
  dplyr::mutate(
    total = sum(count),
    prop = count / total
  )

# Notice that we are specifying a new data frame for the "data =" argument.
ggplot2::ggplot(
  data = data_barplot,
  ggplot2::aes(
    x = gallons_pumped,
    y = prop
  )
) +
  ggplot2::geom_bar(
    fill = "lightblue",
    color = "black",
    stat = "identity"
  ) +
  ggplot2::labs(
    x = "Gallons Pumped",
    y = "Proportion"
  )
```

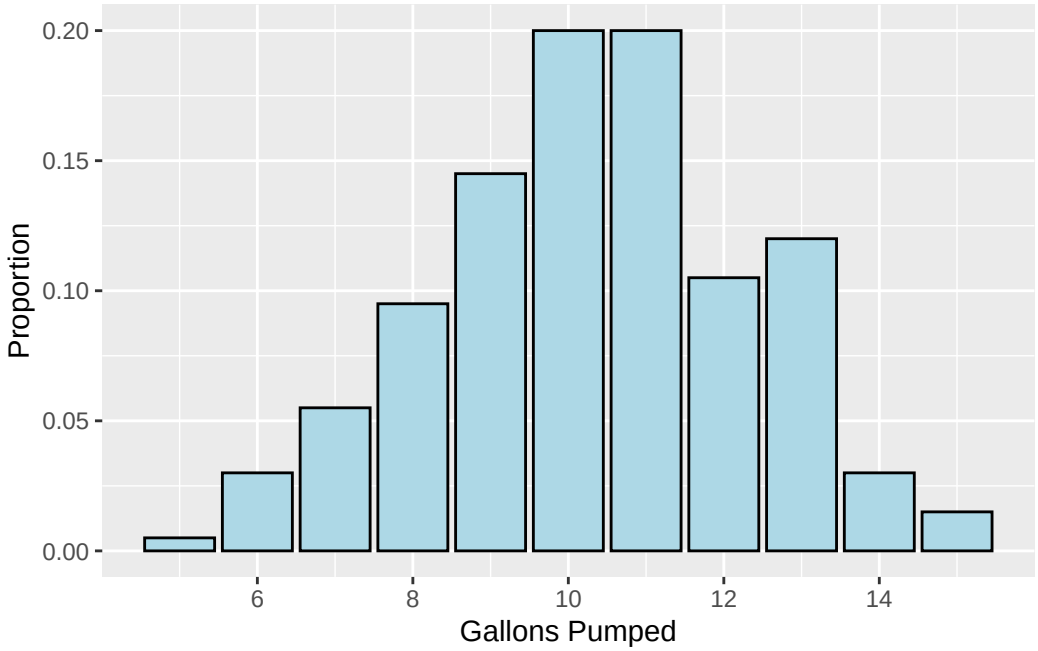

Figure 4: Distribution of Gallons Pumped

Both Figure 3 and Figure 4 are providing the same information—the distribution of the `gallons_pumped` variable. From these figures, we can see that `gallons_pumped` closely follows a normal distribution with observations falling around the mean of 10.32 and a standard deviation of 2.02. That is, on average, you and your friend pumped around 10.32 gallons of gas per visit and across all of your visits you were likely to pump somewhere between 6.28 and 14.36 gallons of gas (two standard deviations above and below the mean).

Now that we have looked at univariate (single variable) plots for each of our variables, it is time to examine some bivariate (two variable) plots using a scatter plot—a plot that allows us to visualize the relationship between two variables. Given that we are interested in the relationship between the outcome variable, `cost`, and the predictor variable, `gallons_pumped`, that is the first bivariate plot on which we will focus.

```
ggplot2::ggplot(
  data = data_cost,
  ggplot2::aes(
    x = gallons_pumped,
    y = cost
  )
) +
  ggplot2::geom_point(
```

```
    color = "lightblue"
  ) +
  ggplot2::labs(
    x = "Gallons Pumped",
    y = "Cost in Dollars"
  )
```

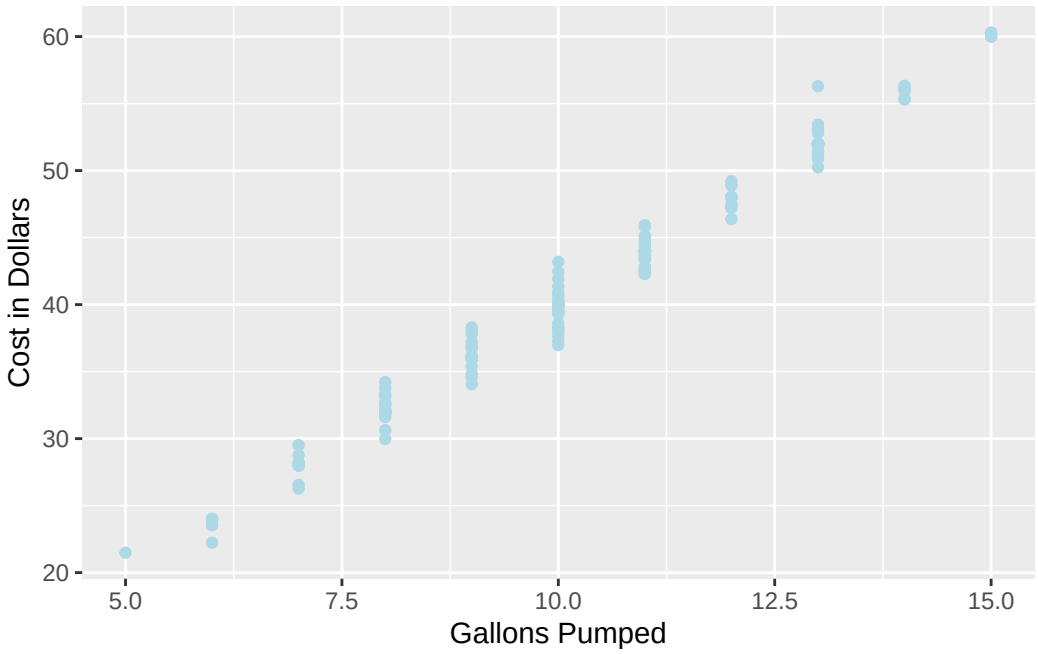

Figure 5: Relationship between Gallons Pumped and Cost in Dollars

The first thing that you may notice in Figure 5 is that the points do not seem to "scatter" across the x-axis rather they only scatter across the y-axis. This is because the variable plotted on our x-axis, `gallons_pumped`, only takes on integer values (ranging from 5 to 15). If our x-axis variable was continuous—meaning it took on fractional values like 8.543—, then we would see the points scatter across the x-axis.

Another thing you may notice is that as you pump more gas (x-axis) you pay more in cost (y-axis). That is, there appears to be a positive relationship between `gallons_pumped` and `cost`.

Now the final thing you may notice is that at any value of `gallons_pumped` the amount you pay (`cost`) takes on a range of values. To better understand this, let us create two new scatter plots: one for the data collected from the old pump and one for the data collected from the new pump (see Figure 6).

```r
ggplot2::ggplot(
  data = data_cost |>
    dplyr::mutate(
      label = dplyr::if_else(pump_id == "new", "New Pump", "Old Pump")
      ),
  ggplot2::aes(
    x = gallons_pumped,
    y = cost
  )
) +
  ggplot2::geom_point(
    color = "lightblue"
  ) +
  ggplot2::facet_wrap(
    ~ label
  ) +
  ggplot2::labs(
    x = "Gallons Pumped",
    y = "Cost in Dollars"
  )
```

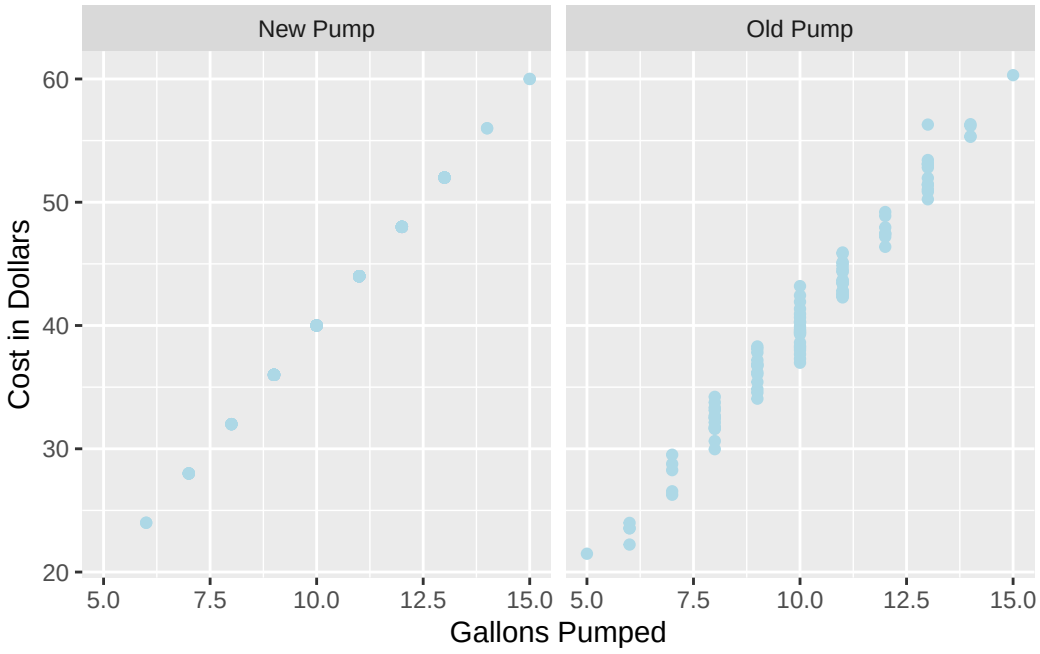

Figure 6: Relationship between Gallons Pumped and Cost in Dollars

Looking at Figure 6, you likely see an immediate difference between the two plots. At the new pump, you always owe the same amount of money when you pump the same amount of gas—if you pump five gallons you will always owe 20 dollars and if you pump 10 gallons you will always owe 40 gallons. That is, your outcome variable, `cost`, does not vary when we fix the value of our predictor variable, `gallons_pumped`. This is because in our simulation we generated the new pump data from a deterministic model—a model with no uncertainty/noise.

Now things get a bit more interesting when we look at the relationship between `gallons_pumped` and `cost` from the data collected from the old pump. Here the amount you owe tends to vary when you pump the same amount of gas—if you pump 10 gallons you could end up paying anywhere between 36.98 dollars and 43.2 dollars. This is what we mean by uncertainty and it is the reason why we need statistical models like linear regression, which is what we turn to next.

Understanding Linear Regression

At this point, we have a good descriptive understanding of our variables (e.g. their means, standard deviations, and general shape of their distributions) and their relationships with one another, so now it is time to start modeling the relationships among the variables with a statistical model like linear regression.

Linear regression is a statistical model that makes two defining assumptions:

1. A linear function can be used to describe the relationship between the predictor variables and the outcome variable.
2. The uncertainty in our outcome variable that remains once we adjust for our predictor variables can be modeled with a normal distribution.

It is unlikely that your data will perfectly satisfy those two assumptions, but linear regression can still be effective even when those assumptions are only approximately satisfied. You can think of this like using a straight line to approximate the distance between your house and a family member's house in a different state—the distance will not be exactly right, but it will be close enough to be useful.

Given that we believe the relationship between cost and gallons pumped should be linear and that the uncertainty in our data is likely to have been generated by a normal distribution (an assumption we can assess), we will proceed with analyzing our pump data using a linear regression model.

Estimating & Interpreting a Linear Regression Model with R

The first thing we will do is use the `lm()` function in R to estimate our linear regression models. For comparison, we will estimate one model using only the data from the new pump (`mod_new`) and one model using only data from the old pump (`mod_old`).

As you can see in the code below, to estimate a linear regression model using the `lm()` function, you have to provide it a model formula, which defines the outcome variable and the predictor variables. For two predictor variables, the model formula is written as: `outcome variable ~ predictor 1 + predictor 2`. Everything to the left of the `~` will be used as an outcome variable and everything to the right of the `~` will be used as a predictor variable. The `~` sign itself can be interpreted as "regress the outcome variable onto the predictor variables." After the model formula, the next argument you should provide the `lm()` function is the name of the data frame that contains your data. In `mod_new`, you will see we specify `data = data_cost |> dplyr::filter(pump_id == "new")`. This tells the `lm()` function that our data is contained in the data frame `data_cost`, which has been filtered to only contain observations from the new pump (`data_cost |> dplyr::filter(pump_id == "new")`).

```
mod_new <- lm(cost ~ gallons_pumped,
              data = data_cost |> dplyr::filter(pump_id == "new"))

mod_old <- lm(cost ~ gallons_pumped,
              data = data_cost |> dplyr::filter(pump_id == "old"))
```

Now that the we have created our models, we can use the `summary()` function to display the model results. We will first interpret the output from `mod_new`.

```
summary(mod_new)
```

```
Warning in summary.lm(mod_new): essentially perfect fit: summary may be
unreliable


Call:
lm(formula = cost ~ gallons_pumped, data = dplyr::filter(data_cost,
    pump_id == "new"))

Residuals:
       Min         1Q     Median         3Q        Max
-2.731e-14 -1.492e-16  1.839e-16  7.876e-16  1.842e-15

Coefficients:
                 Estimate Std. Error    t value Pr(>|t|)
(Intercept)    -1.705e-14  1.567e-15 -1.088e+01   <2e-16 ***
gallons_pumped  4.000e+00  1.467e-16  2.727e+16   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Residual standard error: 2.84e-15 on 98 degrees of freedom
Multiple R-squared:      1, Adjusted R-squared:      1
F-statistic: 7.438e+32 on 1 and 98 DF,  p-value: < 2.2e-16
```

The first piece of information the `summary()` function provides you with is the minimum, first quartile (25th percentile), median (50th percentile), third quartile (75th percentile), and maximum of the model residuals or errors. We will have more to say about this in the section on linear regression diagnostics, but model residuals, or residuals, can be thought of as estimates of the noise inherent your data. Residuals are calculated as:

$$e_i = \text{Actual Cost}_i - \text{Model Predicted Cost}_i.$$

When we say a particular statistical model fits the data well what we mean is that the predictions the model makes about the observed outcome are not far away from what the outcome actually is. Because the data from the new pump was generated from a deterministic model, we see that each residual value is equal to 0 meaning that we are able to exactly predict the observed outcome from our model—there is no uncertainty!

Now let us look at the same output for `mod_old`.

```
summary(mod_old)
```

```
Call:
lm(formula = cost ~ gallons_pumped, data = dplyr::filter(data_cost,
    pump_id == "old"))

Residuals:
    Min      1Q  Median      3Q     Max
-3.0010 -0.9547 -0.0588  0.8986  4.3534

Coefficients:
               Estimate Std. Error t value Pr(>|t|)
(Intercept)     0.10219    0.66925   0.153    0.879
gallons_pumped  3.98797    0.06471  61.624   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.347 on 98 degrees of freedom
Multiple R-squared:  0.9748,    Adjusted R-squared:  0.9746
F-statistic:  3798 on 1 and 98 DF,  p-value: < 2.2e-16
```

The residuals output from the `summary()` function tells us that our worst over prediction is by 3 dollars (the minimum residual) and our worst under prediction is by 4.35 dollars (the maximum residual). From the residual summary, we can also see that in general the residuals seem to fall symetrically around the median where the first and thrid quartiles are similar distances from the median.

The next thing we see from the `summary()` output is a table of numbers labeled: "Coefficients." This table is going to be our main focus as it contains the estimates of our model parameters (`Estimate`), the precision of our estimates called their standard error (`Std. Error`), a test statistic we can use for hypothesis tests (`t value`), and the probability value or p-value we can use to inform our decision on whether to reject our null hypothesis or not (`Pr(>|t|)`).

Interpreting Regression Estimates

Before we begin interpreting the regression estimates, let us save create a more readable coefficient table using the code below.

```
table_new <- summary(mod_new)$coef |> round(2)
```

```
Warning in summary.lm(mod_new): essentially perfect fit: summary may be
unreliable
```

```
colnames(table_new) <- c("Estimate", "Std. Error", "Test Stat.", "P-Value")
table_new[, 3] <- c(Inf, Inf)

table_old <- summary(mod_old)$coef |> round(2)
colnames(table_old) <- c("Estimate", "Std. Error", "Test Stat.", "P-Value")

coef_table <-
  rbind(table_new, table_old) |>
  round(3) |>
  as.data.frame() |>
  tibble::rownames_to_column() |>
  dplyr::mutate(
    `Coef. Name` = c(rownames(table_new), rownames(table_old)),
    `Model ID` = c(rep("New", nrow(table_new)), rep("Old", nrow(table_old)))
  ) |>
  dplyr::select(
    `Model ID`, `Coef. Name`, Estimate,
    `Std. Error`, `Test Stat.`, `P-Value`
  )
```

```
knitr::kable(coef_table)
```

Table 1: Model Coefficients

Model ID	Coef. Name	Estimate	Std. Error	Test Stat.	P-Value
New	(Intercept)	0.00	0.00	Inf	0.00
New	gallons_pumped	4.00	0.00	Inf	0.00
Old	(Intercept)	0.10	0.67	0.15	0.88
Old	gallons_pumped	3.99	0.06	61.62	0.00

Table 1 combines the coefficient tables from `mod_new` and `mod_old` into one single table.

The first column we will focus on in Table 1 is the Estimate column. Before we do that, however, it is important to know the difference between a parameter and an estimate. Think back to the "true" regression model we used to generate our gas pump data. The linear model we used for both the deterministic and statistical models contained two parameters: the regression intercept, β_0 and the regression slope, β_1. We decided to set these parameters equal to 0 and 4, respectively, so we consider 0 and 4 to be the true values of the intercept and slope parameters. If we were not using simulated data, we would not know the "true" values of those parameters. We would need to collect data and estimate those parameters by fitting a statistical model to the data, which is exactly what the Estimate column in Table 1 contains.

The first difference we see between the estimates from the new model and those from the old model is that the estimates for the new model are exactly equal to the parameters we used to generate the data whereas the estimates from the old model are close to the values of the model parameters, but not identical—this is due to the uncertainty in our data from the old pump, which is not present in the new pump data.

Now, focusing on the old pump estimates, we can provide the following interpretation for the regression intercept and slope. The regression intercept is the average amount we would pay if we pumped 0 gallons of gas. So, if we pumped 0 gallons of gas, we would owe .10 cents, on average. This, of course, does not make any sense. If we did not pump any gas, we should not owe any money. To understand this awkward interpretation, we need to take a step back and think about what the intercept is telling us: what is the predicted value of the outcome variable when the values of all the predictor variables in our model are equal to 0. Looking at this prediction only makes sense if all of your predictor variables can take on a value of 0. If they cannot, then we do not need to worry about intepreting the intercept. Moving back to our example, it does not make sense for our predictor variable, `gallons_pumped`, to take on a value of 0 as we know that means we did not interact with the pump and would owe nothing. Thus, we do not need to worry about its awkard intepretation.

To interpret the regression slope, it is helpful to recreate our earlier scatter plot. This time, however, we can add a line using our estimated regression coefficients as well as two different colored points: one to indicate what the mean of the outcome variable is at a set level of the predictor variable (e.g. how much are you paying, on average, when you pump 10 gallons of gas) and one to indicate the predicted value of the outcome variable at a set level of the predictor (e.g. what does the model predict you will pay when you pump 10 gallons of gas).

```
data_mean_labels <-
  data_cost |>
  dplyr::filter(
    pump_id == "old"
  ) |>
  dplyr::summarize(
    mean_cost = mean(cost),
    .by = gallons_pumped
  ) |>
  dplyr::mutate(
    label = round(mean_cost, 2)
  ) |>
  dplyr::arrange(gallons_pumped)

data_pred_labels <-
  data_cost |>
  dplyr::filter(
    pump_id == "old"
  ) |>
  dplyr::select(gallons_pumped) |>
  dplyr::distinct() |>
  dplyr::mutate(
    pred = mod_old$coefficients[1] + mod_old$coefficients[2] * gallons_pumped,
    label = as.character(round(pred, 2))
  ) |>
  dplyr::arrange(gallons_pumped)

data_labels <-
  data_mean_labels |>
  dplyr::select(
    gallons_pumped,
    value = mean_cost
  ) |>
  dplyr::mutate(
    id = "Outcome Mean"
  ) |>
```

```
  dplyr::bind_rows(
    data_pred_labels |>
      dplyr::select(
        gallons_pumped,
        value = pred
      ) |>
      dplyr::mutate(
        id = "Model Prediction"
      )
  )

ggplot2::ggplot(
  data = data_cost |> dplyr::filter(pump_id == "old"),
  ggplot2::aes(
    x = gallons_pumped,
    y = cost
  )
) +
  ggplot2::geom_point(color = "lightblue") +
  ggplot2::geom_smooth(method = "lm", se = FALSE, formula = y ~ x,
                       color = "black", linewidth = .75) +
  ggplot2::geom_point(
    data = data_labels,
    ggplot2::aes(
      x = gallons_pumped,
      y = value,
      color = id
    ),
    size = 3
  ) +
  ggplot2::labs(
    x = "Gallons Pumped",
    y = "Cost in Dollars",
    color = "Type"
  )
```

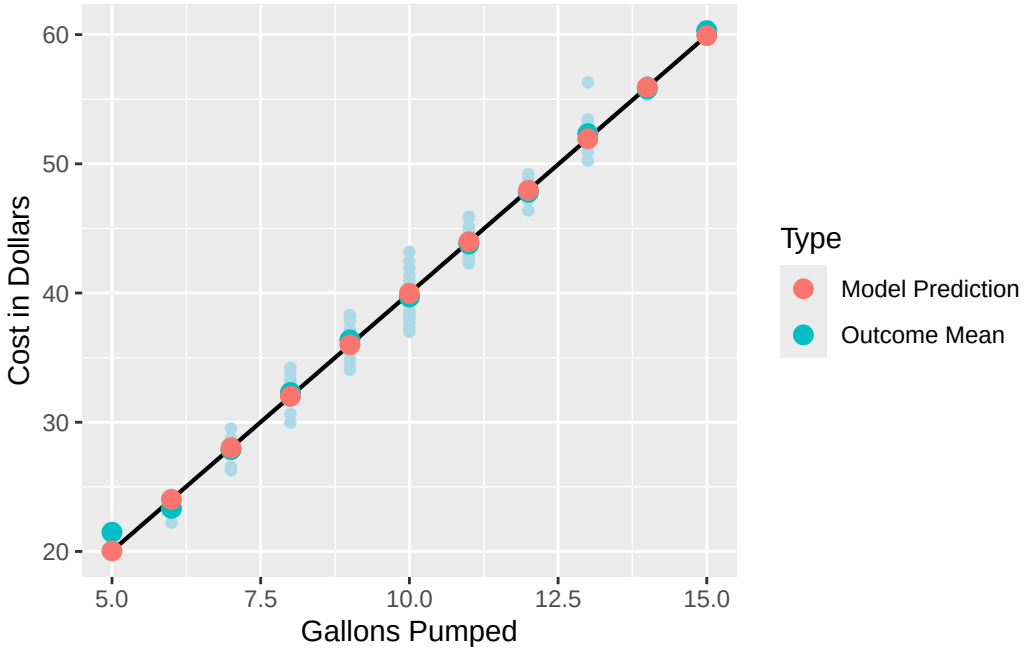

Figure 7: Estimated Relationship between Gallons Pumped and Cost in Dollars

As you can see in Figure 7, you end up paying slightly different amounts of money at any set amount of gas pumped—we saw this in Figure 6. Because we have different values of the outcome variable at fixed levels of the predictor variable, no single line will fit every point perfectly—we have to choose what line will best fit our data. For instance, we could decide that our line should be close to the maximum value of the outcome variable at each level of the predictor variable. Alternatively, we could decide that our line should be close to the minimum value of the outcome variable at each level of the predictor.

It turns out that if we want a line the least amount of prediction error (i.e. small residuals), then we should choose a line that is close to the mean of the outcome variable at each level of the predictor variable. This is exactly what the linear regression does. Looking at Figure 7, we can see that at a fixed value of the predictor variable (e.g. 10 gallons), the regression line goes as close as possible to the mean of the outcome variable. In effect, the regression slope is a single number summary of how the average value of the outcome variable changes as we move up one unit on the predictor variable.

In the context of our gas example, the estimated regression slope tells us that for every additional gallon pumped (a one unit increase on the predictor variable), we will pay 3.99 dollars more, on average. For example, if our pretend friend pumps 10 gallons of gas at the old pump, our model predicts they will pay 39.98 and if they pump 11 gallons, a one unit increase, our model predicts they will pay 43.97, which is a difference of 3.99 dollars—the value of our estimated regression coefficient.

Of course, if we look at the actual values of our data, moving up a single gallon of gas pumped does not always result in an increase of 3.99 dollars. For example,

Linear Regression with an Example from the Social/Behavioral Sciences

Using Categorical Predictors

Modeling Interactions in Linear Regression

Linear Regression Diagnostics

Assumptions Underlying Linear Regression