# 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, $\beta_1$. 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. This also means that for every additional gallon of gas we put into our car (a single unit increase), our cost will increase by \$4—at no point will an additional gallon of gas cost us more or less than \$4. 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. 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.

## 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, but you believe the old pump is malfunctioning and will sometimes charge you a little under \$4 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 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:

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

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

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 $\sigma$.
4. The variation in cost, $\sigma$, 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, you have your friend record the gallons they pumped at the old pump and the amount they paid. 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.

```r
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)))

# 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`.

```
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`.

```r
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"
  )
```

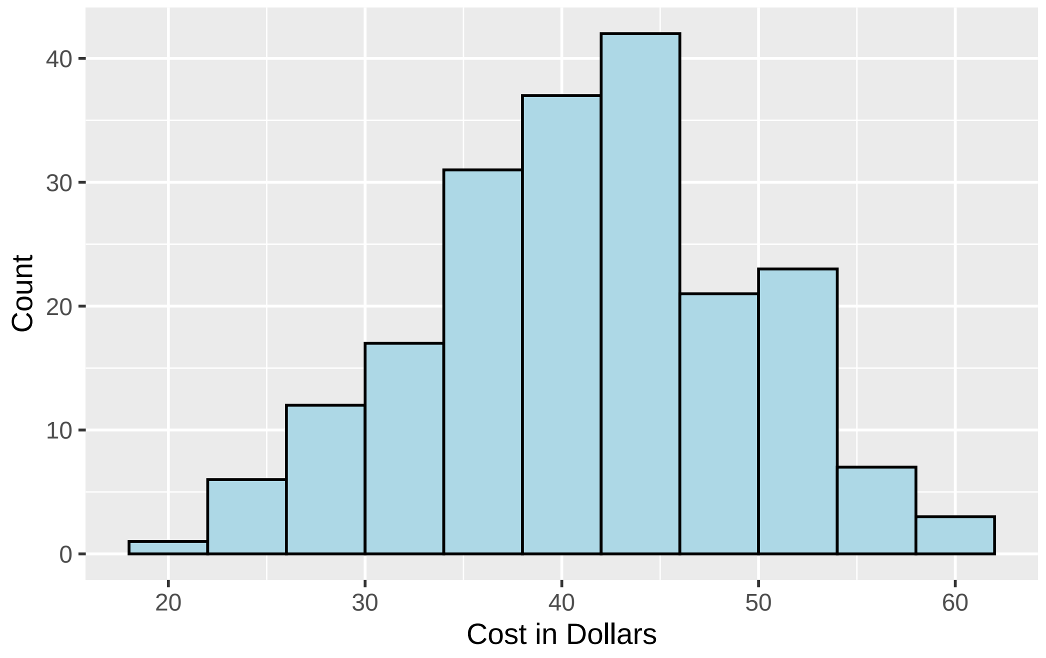


Figure 1: Distribution of Cost in Dollars

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"
  )
```

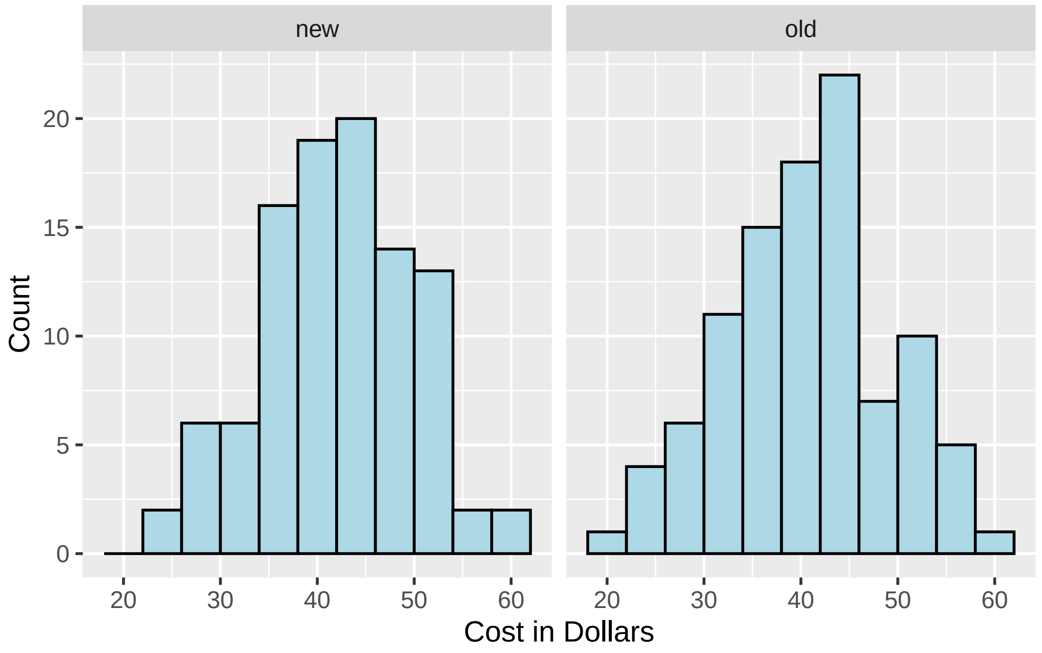


Figure 2: Distribution of Cost in Dollars by Pump

As we will talk about later on, the assumptions statistical models make about the probability distribution that generates your data are about 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"
  )
```

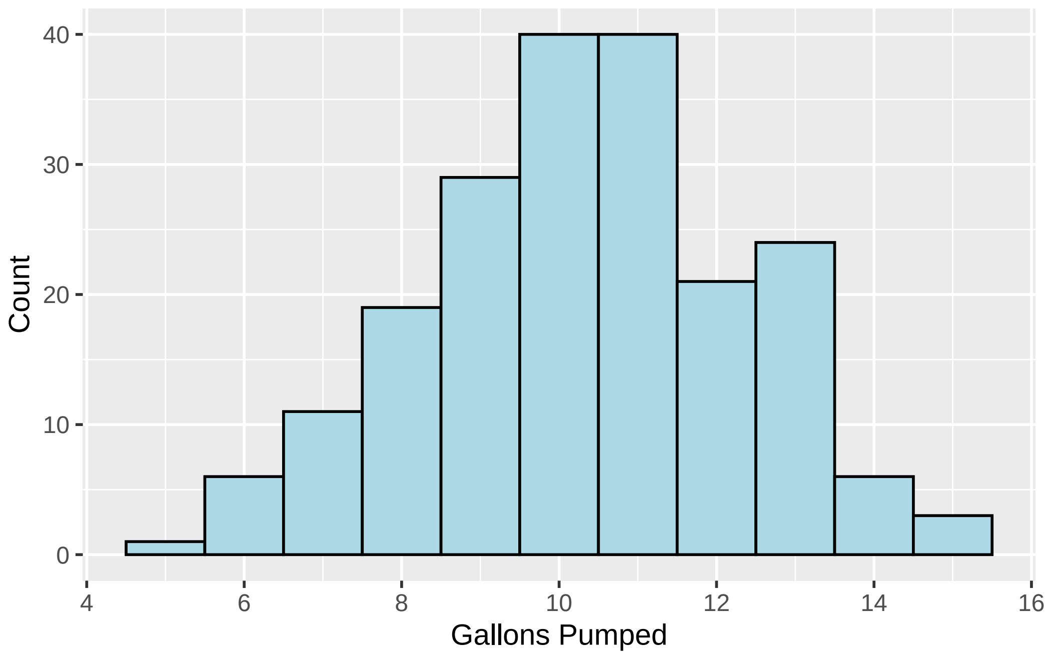


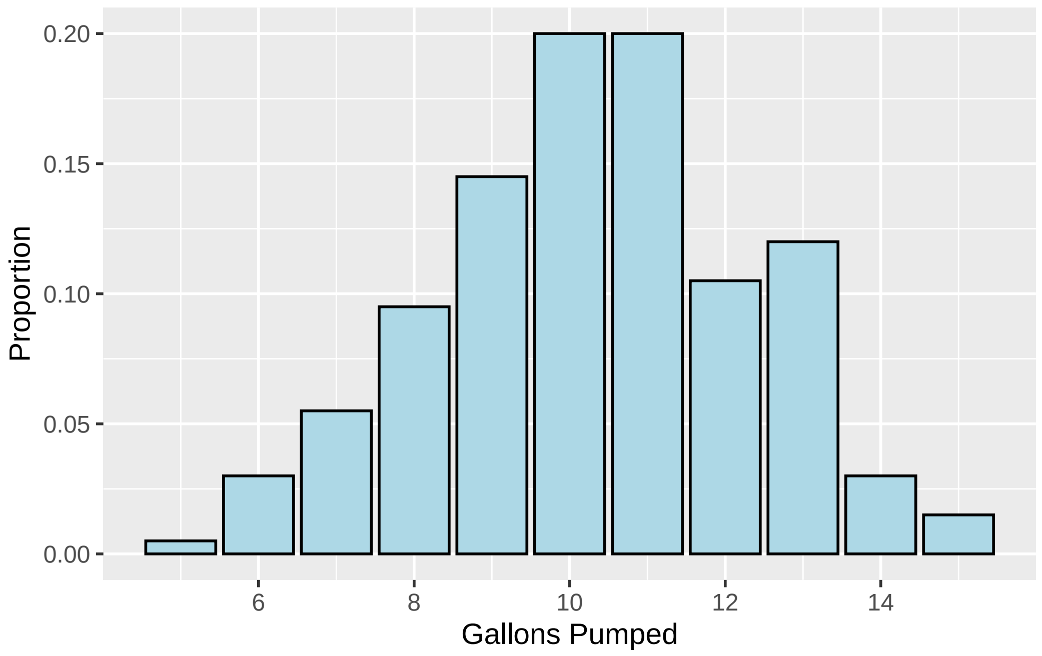
Figure 3: Distribution of Gallons Pumped

Because our `gallons_pumped` variable only takes on discrete values (by design of our simulation), we could also use a bar plot to visualize its distribution (see Figure 3).

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

```r
    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"
  )
```



{#fig-dist-gallon-bar plot fig-pos='H'}

Both Figure 3 and Figure 3 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. Because we are interested in the relationship between the outcome variable, `cost`, and the predictor variable, `gallons_pumped`, our initial focus will be on a plot that explores their relationship.

```
ggplot2::ggplot(
  data = data_cost,
  ggplot2::aes(
    x = gallons_pumped,
    y = cost
  )
) +
  ggplot2::geom_point(
    color = "black", fill = "lightblue",
                      shape = 21
  ) +
  ggplot2::labs(
    x = "Gallons Pumped",
    y = "Cost in Dollars"
  )
```

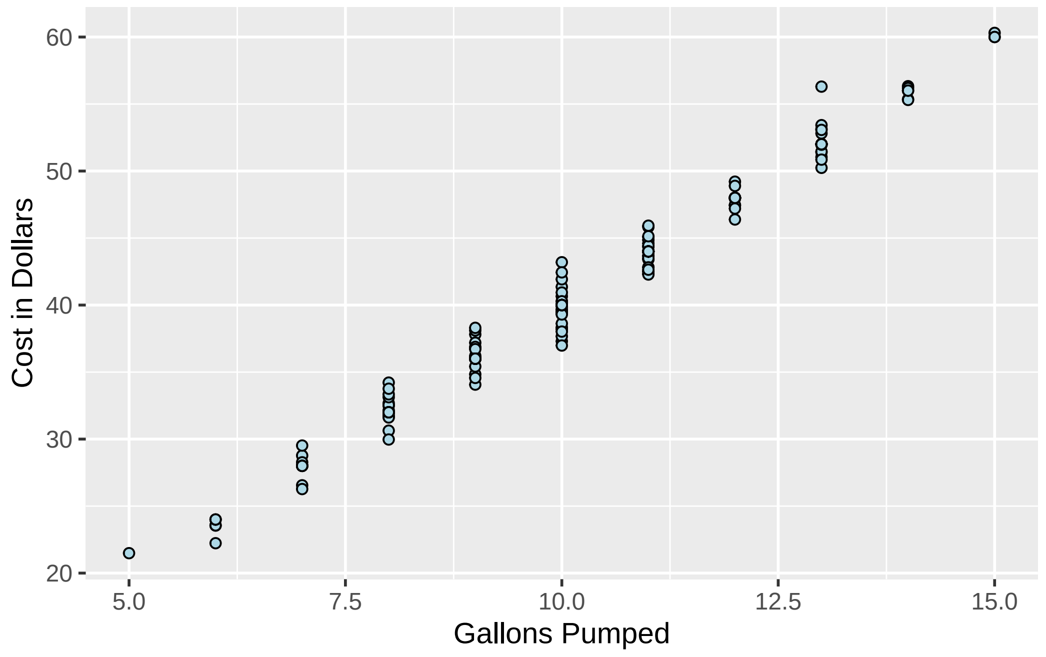


Figure 4: Relationship between Gallons Pumped and Cost in Dollars

The first thing that you may notice in Figure 4 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`.

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 5).

```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 = "black", fill = "lightblue",
                      shape = 21
  ) +
  ggplot2::facet_wrap(
    ~ label
  ) +
  ggplot2::labs(
    x = "Gallons Pumped",
    y = "Cost in Dollars"
  )
```

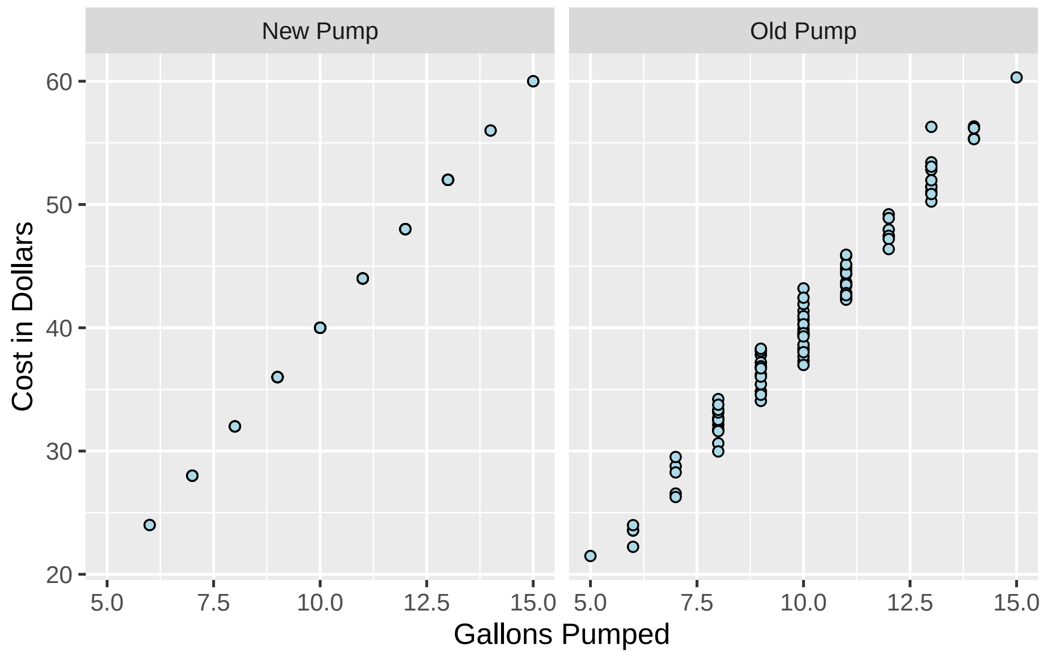


Figure 5: Relationship between Gallons Pumped and Cost in Dollars

Looking at Figure 5, 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 and if you pump 10 gallons you will always owe $40. That is, your outcome variable, `cost`, does not vary when we fix the value of our predictor variable, `gallons_pumped`. This occurs because we generated the new pump data from a deterministic model—a model with no uncertainty/noise.

Things get 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.

## 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 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.

```r
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`.

```r
summary(mod_new)
```

```

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 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`.

```r
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 symmetrically around the median where the first and third 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 Coefficient Estimates

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

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

```r
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`
  )
```

```r
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, $\beta_0$ and the regression slope, $\beta_1$. 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.

Focusing on the Estimate column, 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.

For 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 logically take on a value of 0. If they cannot, then we do not need to worry about interpreting the intercept.

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 use the pump and owe nothing. Thus, we do not need to worry about interpreting the intercept.

Next, 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:

1. One type of point 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).
2. One type of point 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).

```r
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 = "black", fill = "lightblue",
                      shape = 21) +
  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,
      fill = id
    ),
    size = 3,
    color = "black",
    shape = 21
  ) +
  ggplot2::labs(
    x = "Gallons Pumped",
    y = "Cost in Dollars",
    color = "Type"
  )
```

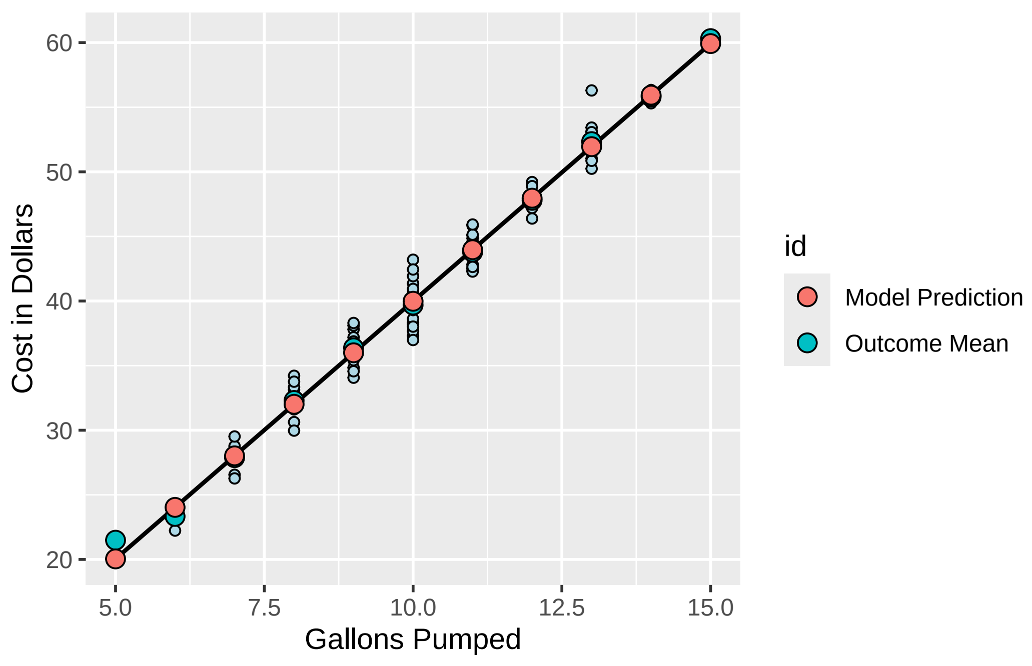


Figure 6: Estimated Relationship between Gallons Pumped and Cost in Dollars

As you can see in Figure 6, you end up paying slightly different amounts of money at any set amount of gas pumped—we saw this in Figure 5. 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 with the least amount of prediction error (i.e. smallest 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 6, 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 average increase of 3.99 dollars. For example, in our dataset, our pretend friend paid 39.69 dollars on average when they pumped 10 gallons of gas and 43.79 when they pumped 11 gallons of gas, which is a 4.1 dollar difference. Whereas they paid 21.49 dollars when they pumped 5 gallons of gas and 23.33 dollars when they pumped 6 gallons of gas, which is a 1.84 dollar difference. Neither difference is exactly equal to the estimated regression slope, 3.99, but they are close.

This assessment of how closely our model predictions are to our observed data leads us to the next topic: assessing the model fit of a linear regression model.

## Assessing Model Fit

Beyond interpreting the individual regression coefficients, it is also important for us to understand how well our model fits our data. That is, how closely do our model predictions match our observed data. To understand this, we first need to define what a model prediction is:

$$\text{Model Prediction} : \hat{Y}_i = \hat{\beta}_0 + \hat{\beta}_1 X_{1i}.$$

If we think about our estimated regression equation as a regular equation, then a model prediction is the output of this equation when we plug in a value for the predictor variable. For instance, the estimated regression coefficients from our gas example are 0.1 for the intercept and 3.99 for the slope. If we plugged 6 in for `gallons_pumped`, then our model would output or predict: $0.1 + 3.99 \times 6$ or 24.04.

Now, rather than plug in a random value for the predictor variable, we can go observation by observation in our dataset and plug in the predictor values that appear for each observation. This will give us a predicted outcome value for each observation. We can look at an example of this using the code below.

```r
# We will only focus on the old pump data, so we filter the overall dataset
# to contain only values from the old pump and save it in a new object:
# data_old
data_old <- data_cost |> dplyr::filter(pump_id == "old")

# Now we use the predict() function to add a column of model predictions to
# our data_old
data_old <-
  data_old |>
  dplyr::mutate(
    model_prediction =
      predict(mod_old, data = data.frame(data_old$gallons_pumped))
  )

# Now we will look at our model predictions compared to our model outcomes
data_old |>
  dplyr::select(
    gallons_pumped,
    cost,
    model_prediction
  )
```

```
# A tibble: 100 x 3
   gallons_pumped  cost model_prediction
            <int> <dbl>            <dbl>
 1             12  48.0             48.0
 2             13  51.0             51.9
 3              9  34.8             36.0
 4             10  39.5             40.0
 5             10  38.3             40.0
 6             12  47.4             48.0
 7             10  43.2             40.0
 8              8  32.7             32.0
 9             10  37.3             40.0
10              9  37.8             36.0
# i 90 more rows
```

Now that we have a prediction for each observation, we can calculate how closely our predicted outcomes are to our observed outcomes by subtracting our predicted outcomes from our observed outcomes. This is the model error, or residual we saw earlier:

$$e_i = Y_i - \hat{Y}_i.$$

Using our model errors, we can calculate two measures of fit: the residual standard error (RSE) and the $R^2$.

The RSE is an estimate of the outcome standard deviation after we have adjusted for the effects of our predictor variables—the standard deviation of the model errors. It is a measure of a model's absolute predictive ability, where an RSE of 0 tells us that our model perfectly predicts our data and any positive, non-zero number tells us how off our predictions are, on average. To get some more insight into RSE, we can investigate its mathematical definition:

$$\text{RSE} = \sqrt{\frac{\sum (Y_i - \hat{Y}_i)^2}{N - K - 1}} = \sqrt{\frac{RSS}{N - K - 1}}.$$

The equation above tells us a few things. First, the RSE uses the squared difference between our observed and predicted outcome to measure how closely our predicted outcome matches our observed outcome—this is called the Residual Sum of Squares or RSS. The RSS ensures that over-predictions (negative deviations) and under-predictions (positive deviations) do not cancel each other out. Second, the RSE creates an average squared deviation by dividing the sum of squared deviations by the sample size ($N$) minus both the number of predictors in the model ($K$) and the intercept (1). This provides a more accurate estimate of the average deviation than dividing by $N$ alone as it recognizes that we have already used some of our data to estimate the $K$ regression slopes (one for each predictor) and one regression intercept. Finally, the RSE takes the square-root of the averaged squared difference, so that the metric can be interpreted on the scale of the outcome, which is dollars in our example.

We can get the RSE from our model output under the label, `Residual standard error`. The RSE of our model is 1.35, which we can loosely interpret as "on average, our model prediction will be off by about $1.35." A logical next question is: "Is our RSE good—is it small enough?" A possible response is: "Compared to what?" This is where the $R^2$ helps.

The $R^2$ tells us the percent (or proportion) that our baseline RSS (the numerator of the RSE) decreased because of the addition of our predictor variables—a percent change. That is, the $R^2$ is a measure of a model's predictive ability relative to some baseline model. The baseline model typically used is a regression model without any predictors; a model that only estimates the regression intercept. In this baseline model, the regression intercept is equal to the mean of the outcome variable and the estimated RSE is equal to the standard deviation of the outcome variable and the RSS is equal to squared difference between the observed outcome and its mean. We can see this using the code below.

```r
# Estimate a baseline model. The 1 represents the regression intercept.
mod_baseline <- lm(cost ~ 1, data = data_old)

# Estimate the RSS, SD, and mean of the outcome
(sum((data_old$cost - mean(data_old$cost))^2)) |> round(3) # RSS
```

```
[1] 7069.175
```

```r
mean(data_old$cost) |> round(3)
```

```
[1] 40.5
```

```r
sd(data_old$cost) |> round(3)
```

```
[1] 8.45
```

```r
# Look at the model summary and compare the intercept and
# residual standard error to the outcome mean and SD. We can use the
# model RSE to calculate the RSS
summary(mod_baseline)
```

```

Call:
lm(formula = cost ~ 1, data = data_old)

Residuals:
     Min       1Q   Median       3Q      Max 
-19.0125  -5.7301  -0.2221   4.8084  19.8082 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)   40.500      0.845   47.93   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 8.45 on 99 degrees of freedom
```

```r
baseline_rse <- summary(mod_baseline)$sigma^2
n <- nrow(data_old) # Get the sample size using nrow()
baseline_rss <- baseline_rse * (n - 1) # RSS calculated from the baseline model
round(baseline_rss, 3)
```

```
[1] 7069.175
```

Now that we have our baseline RSS ($RSS_{\text{Baseline}}$) and our model RSS ($RSS_{\text{Model}}$), we can calculate $R^2$ as:

$$R^2 = \frac{RSS_{\text{Baseline}} - RSS_{\text{Model}}}{RSS_{\text{Baseline}}}.$$

Looking at the equation for $R^2$, we can loosely interpret it as the proportion that our baseline error is reduced by adding our predictor variables. Or, alternatively, the extent to which our model's predictive power is improved above its baseline by adding predictor variables—the proportion of variance we can explain in our outcome variable due to our predictor variables.

Thankfully, when we use the `summary` function on our regression model in R, it provides both the model RSE and the $R^2$. In our gallons pumped model, the $R^2$ is 0.975. This means that our baseline prediction error—the error when we predict gas cost using the average cost of gas in our dataset (average cost = 40.5)—is reduced by 97.5% due to the addition of the `gallons_pumped` predictor.

Imagine your pretend friend is building a budget for next week and needs to know how much to budget for gas. Using the baseline model, they could decide to budget \$40.5 for gas—the average amount they have spent on gas to date (taking the average of all the `cost` observations in `data_old`). But, let us say that our friend knows that next week they will only need to pump five gallons of gas. Now they can revise their budget and predict that they will only need to budget \$20.04 for gas. Knowing the amount of gas they will need to pump, the value for the predictor variable, helped them make a more accurate prediction for how much they were going to pay, their outcome variable.

Linear regression, and, in general, any statistical model, will never fit your data perfectly—you will never see an RSE value of 0 or an $R^2$ of 1. What these models will hopefully do well is reduce the noise inherent in your data by enough so that you can find the signal—the true set of model parameters that generated your data. This move, however, from using your statistical model to describe your observed data to using it to make general inferences about the population model that generated your data, and all other hypothetical datasets you may have been able to collect, is called statistical inference.

## Statistical Inference and the Linear Regression Model

We need to make one final adjustment to our example. In addition to adding a random amount of error to the cost of gas, the screen on the old pump that displays the cost of gas by the gallon (\$4 per gallon) is broken. Your pretend friend does not know how much the gas costs. They do have a hypothesis about the cost, though. Your friend knows that gasoline is not free and that the more gasoline they pump, the more they will have to pay. They would like to test their hypothesis using the data we collected from the old gas pump.

This type of setup where the researcher (our pretend friend) tries to generalize from their data to the population model—the model that generated their data—by setting up and testing a researcher (or alternative) hypothesis against a null hypothesis is called Null Hypothesis Significance Testing (NHST). It is the most popular way to make inferences from one's model, and it is important for you to know how it works starting with the concept of a sampling distribution.

## The Engine Driving Statistical Inference: The Sampling Distribution

Most, if not all, of the variables you will work with are going to be random variables. Random variables are variables that can take on a set of values, but the actual value that is observed is determined by a probability distribution. For example, the outcome of a coin flip is a random variable. The outcome of a coin flip can take on two values, Heads or Tails, but, prior to observing the coin after the flip, we cannot say what the actual outcome is going to be. We can only talk about the probability of observing a heads or a tails, which for a fair coin follows a probability distribution that states heads has a 50% probability of occurring and tails has a 50% probability of occurring.

Similarly, we can construct a hypothetical scenario where we repeatedly resample our data under identical conditions. The datasets we collect in this scenario are the outcomes of a random process—our population model. This random process is another way of thinking about the uncertainty that is present in the random variables we are measuring. Consider our gas pump example, rather than obtaining one dataset containing 100 observations from the old pump, imagine that we collected 1,000 datasets each containing 100 observations from the old pump under identical conditions. Each one of these datasets contains our two variables: `gallons_pumped` and `cost`, but the values of these variables, especially `cost`, will vary from dataset to dataset because of the uncertainty inherent in the data (i.e. the old pump making random errors).

Because our estimated regression coefficients are based on these random variables whose observed values will vary dataset by dataset, we can also think of our estimated regression coefficients as random variables. That is, the values our regression coefficient estimates can take on are determined by a probability distribution. We call this probability distribution a sampling distribution. Each regression coefficient has its own sampling distribution.

Before we can actually do anything with the sampling distribution, we first have to determine which of the many probability distributions that exist best approximates the sampling distribution. Thankfully, it turns out that with enough observations (usually 30 or more), we can make a strong argument that the sampling distribution of a regression coefficient closely follows a normal distribution. That is, we can assume that our estimated regression coefficient was generated from a normal distribution with a mean and standard deviation. The mean of this distribution is usually of interest to us as it represents the "true" regression coefficient—the thing we are trying to estimate. The standard deviation of a sampling distribution is called the standard error and it quantifies how much uncertainty is present in our regression coefficient estimates. The larger the standard error, the more likely it is that our regression coefficients are far away from their means, their "true" values.

## Simulating a Sampling Distribution

To solidify the concept of a sampling distribution, we are going to use a simulation to construct a sampling distribution for the regression slope that quantifies the relationship between gallons pumped and cost. We will start by turning the simulation code for our initial gas pump example into an R function.

```r
set.seed(2)

reg_slope_est <- numeric()

# Start the for loop, which just tells R to repeat everything contained in "{}"
# 1,000 times
for(i in 1:1000) {

  # Collect our new dataset, call it data_sim
  data_sim <- simulate_gas_example(n_visit = 100,
                                   gallons_pumped_beta = 4)

  # Estimate our linear regression model
  mod_sim <- lm(cost ~ gallons_pumped, data = data_sim)

  # Save the regression slope in an object named reg_slope_est
  reg_slope_est <- c(reg_slope_est, mod_sim$coef[2])


}
```

There are four main things to focus on in the code above. First, using the `simulate_gas_example` function, we simulate a gas dataset, `data_sim`, using the same parameters we used to simulate our initial gas example—this is the idea of collecting a new dataset under identical conditions. Second, using this newly simulated dataset, we estimate a regression model, `mod_sim`, where the outcome is `cost` and the predictor is `gallons_pumped`. Third, we save the regression slope estimate for `cost` in a the `reg_slope_est` object. Fourth, and finally, using a `for` loop, we repeat the first three steps 1,000 times, which is the analogous to us going out and resampling our data under identical conditions 1,000 times.

The object we named `reg_slope_est` contains 1,000 estimates of the `gallons_pumped` regression slope, which estimates the effect `gallons_pumped` has on `cost`. Remember, `reg_slope_est` is a random variable and we refer to its distribution as a sampling distribution. The mean and standard deviation of `reg_slope_est` tell us the mean and standard error of its sampling distribution, respectively.

The mean of the `reg_slope_est` variable is 4.003, which is very close to the true value of the effect `gallons_pumped` has on `cost`, 4. This is expected as the mean of a regression coefficient's sampling distribution should be equal to its population parameter—the true value of the coefficient. Next, the standard error (standard deviation) of `reg_slope_est` is 0.08. The standard error tells us how far away our estimates are away from their mean, which is equal to the true regression slope value. So the larger the standard error, the farther away, on average, the regression estimates are from their true value. This is why we want a small standard error as it means that are estimates should be close to the parameter they are estimating.

Now that we have a descriptive understanding of our sampling distribution, we need to plot its overall distribution. This will give us the shape of the distribution.

```r
ggplot2::ggplot(
  data = data.frame(reg_slope_est),
  ggplot2::aes(x = reg_slope_est)
) +
  ggplot2::geom_histogram(
    color = "black",
    fill = "lightblue",
    binwidth = .025
  ) +
  ggplot2::labs(
    x = "Estimate",
    y = "Count"
  )
```

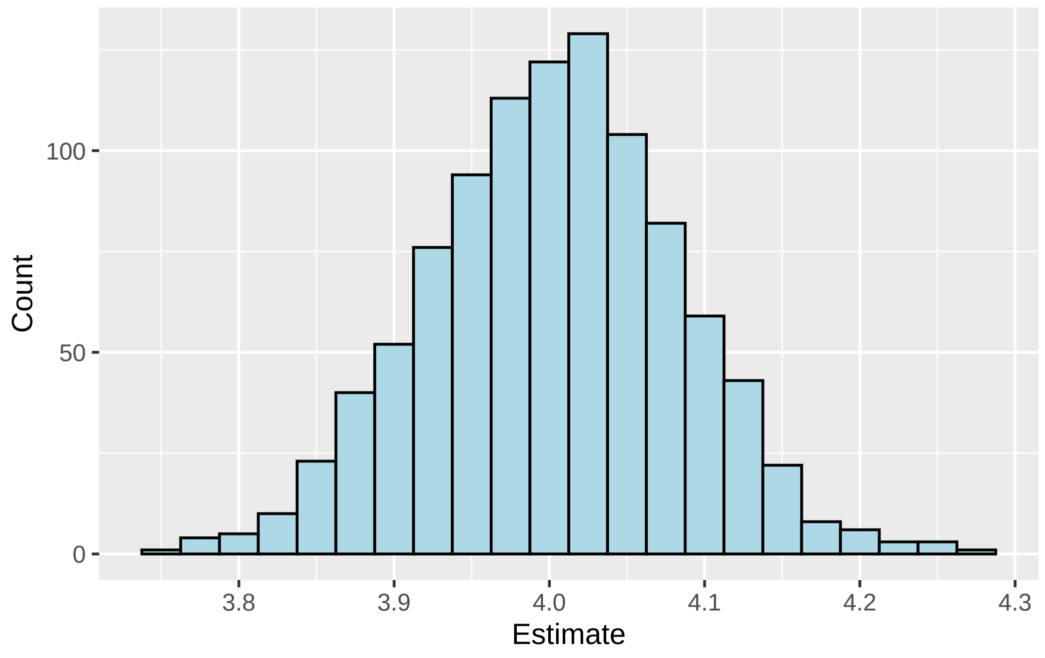


Figure 7: Sampling Distribution of a Regression Slope

Figure 7 shows us that our sampling distribution seems to closely follow a normal distribution. This means that approximately 95% of the estimate values should fall within plus/minus two standard errors of the mean: 3.84 and 4.16. Further, values greater than three standard errors of the mean are highly unlikely.

Now that we have a better understanding of what a sampling distribution is and what it tells us, it is time to see its role in statistical hypothesis testing.

## Understanding Standard Errors, Test Statistics, and P-Values

Earlier, our pretend friend setup an informal alternative hypothesis that stated gas is not free. Let us now formalize their hypothesis into a set of hypotheses, the null hypothesis and the alternative hypothesis:

- $H_0$ or Null Hypothesis: The relationship between gallons pumped and cost of gas equals 0 or $\beta_1 = 0$.
- $H_a$ or Alternative Hypothesis: The relationship between gallons pumped and cost of gas is greater than 0 or $\beta_1 > 0$.

The null hypothesis is a statement that the effect (statistical parameter) takes on a single value, usually a value of 0 which means that there is no effect. In contrast, the alternative hypothesis is a statement that the effect takes on a range of values. The alternative hypothesis can be non-directional or directional. A non-directional alternative hypothesis states that the effect can take on any non-zero value regardless if that value is positive or negative, whereas a directional hypothesis is a statement that the effect takes on a particular sign: positive or negative, but not both—the direction of the effect. We will see later that this has implications for the effect's p-value.

Under the NHST framework, we are only ever testing the null hypothesis, and we can only ever reject the null hypothesis or fail to reject the null hypothesis. We can never accept the null hypothesis. At no point are we ever testing the alternative hypothesis, as the researcher it is our job to provide evidence that supports our alternative hypothesis, so if we reject the null hypothesis, then the only plausible explanation, given the evidence, is our alternative hypothesis. But first, how do we go about testing the null hypothesis?

To understand how we test our null hypothesis, it will be helpful to have the coefficient table from our regression model: `mod_old`. You can find this table below (Table 2).

Table 2: Model Coefficients using Old Gas Pump Data

| | Model ID | Coef. Name | Estimate | Std. Error | Test Stat. | P-Value |
|---|---|---|---|---|---|---|
| 3 | Old | (Intercept) | 0.10 | 0.67 | 0.15 | 0.88 |
| 4 | Old | gallons_pumped | 3.99 | 0.06 | 61.62 | 0.00 |

To begin testing the null hypothesis, we have to make a few assumptions to setup the sampling distribution according to the null hypothesis:

1. Our sampling distribution follows a normal distribution.
2. Our null hypothesis is true (weird, right?).

The first assumption is not new and, thankfully, as long as we have thirty or more observations, is pretty safe. As for the second assumption, this might seem weird to you; why on earth would we assume the null hypothesis is true? We do this because the null hypothesis specifies a specific value for the true effect, 0 in our case—the mean of the sampling distribution under the null hypothesis. The first two assumptions setup the shape of the sampling distribution, normal, and its mean, the value specified by the null hypothesis. To finish setting up the sampling distribution, we just need the standard error, which we get from our data (see the Std. Error column in Table 2). With our sampling distribution fully setup, we can now turn to testing our null hypothesis.

To test our null hypothesis, we first need to transform our regression estimate into a test statistic. A test statistic is a value that summarizes how many standard errors away a parameter estimate falls from the value specified by the null distribution. The following formula is used to calculate a test statistic:

$$\text{Test Statistic} = \frac{\text{Estimate} - H_0}{\text{Standard Error}}$$

To create a test statistic for our regression estimate, the estimate can be found in the third column of Table 2, 3.99, the null value is 0, and the standard error can be found in the fourth column of Table 2, 0.06. Then we subtract 0 from our estimate and divide by its standard error to get the value found in the Test Stat. column in Table 2 (Note that due to rounding error if you divide 3.99 by 0.06 you will get a test statistic of 66.5). The test statistic tells us that the estimate, 3.99 is 61.62 standard errors away from the null value, 0.

With our test statistic in hand, we can calculate the information we need to determine if we can safely reject the null hypothesis—the p-value. For a non-directional hypothesis, the p-value is the probability (probability value) of observing a test statistic, in either direction (positive value or negative value), that is as large or larger than the absolute value of our observed test statistic. That is, we need to know the probability of observing a value as extreme or more, than the negative value of our test statistic and the probability of observing a value as extreme or more, than the positive value of our test statistic. For a directional hypothesis, the p-value is the probability of observing a test statistic, in the hypothesized direction, that is as large or larger than the value of our observed test statistic.

Let us first calculate the p-value for a non-directional hypothesis test. To calculate the probability value, we need to use the cumulative distribution function for a normal distribution (CDF), which is a function in R: `pnorm`. For a random normal variable (like the regression coefficient estimate), the normal CDF tells us the probability of observing a value of the random variable at or below a specific value in the normal distribution. For example, the probability of observing a value of less than or equal to 0 in a normal distribution with a mean of 0 and standard deviation of 1 is .50 and the probability of observing a value less than or equal to two in a normal distribution with a mean of 0 an standard deviation of 1 is 0.98.

```r
# What probability (proportion) of observations fall below 0
pnorm(0, mean = 0, sd = 1)
```

```
[1] 0.5
```

```r
# What probability (proportion) of observations fall below 2
pnorm(2, mean = 0, sd = 1)
```

```
[1] 0.9772499
```

So, if we want to know the probability that our test statistic is not equal to 0, then we need to use the normal CDF with a mean and standard deviation equal to 0 and 1, respectively, to calculate the probability of seeing a value as extreme or more extreme than the positive value of our test statistic (61.62) and a value as extreme or more extreme than the negative value of our test statistic (-61.6244628).

```r
# Get our test statistic from the model output
test_stat <- summary(mod_old)$coefficients[2, 3]

# Calculate probability of a value equal to or smaller than the negative value
# of our test statistic: -test_stat
prob_negative_direction <- pnorm(-test_stat, mean = 0, sd = 1)

# Calculate the probability of a value equal to greater than the positive value
# of our test statistic: test_stat
prob_positive_direction <- 1 - pnorm(test_stat, mean = 0, sd = 1)

# Add the two probabilities together to get our p-value
prob_negative_direction + prob_positive_direction
```

```
[1] 0
```

The reason we have to calculate and add two probabilities together for the non-directional p-value is because we have chosen to hedge our bet on our alternative hypothesis a little—we can reject our null hypothesis with either a positive or negative test statistic. Because of this hedge, though, our p-value will now be larger than if we specified a directional hypothesis.

For a directional hypothesis, if we hypothesize the test statistic will be positive (i.e. greater than 0), then we can only reject the null hypothesis with a large, positive test statistic (and vice versa). It is important to note that R only provides p-values for non-directional hypotheses, but as we will see shortly, it is easy to transform the non-directional p-value into a directional p-value.

To calculate a directional p-value, we take a similar approach to the one we took to calculate the non-directional p-value, but this time we just need one of the probabilities: the probability that follows our hypothesized direction. For our original hypothesis, we specified a positive direction so we only need to probability of observing a value as extreme or more than the positive value of our test statistic.

```r
# Get our test statistic from the model output
test_stat <- summary(mod_old)$coefficients[2, 3]

# Calculate the probability of a value equal to greater than the positive value
# of our test statistic: test_stat
prob_positive_direction <- 1 - pnorm(test_stat, mean = 0, sd = 1)

# The probability in the positive direction is our p-value
prob_positive_direction
```

```
[1] 0
```

Thanks to the symmetry of the normal distribution, the probability values in the positive and negative directions will always be equal to one another, so to calculate the directional p-value from a non-directional p-value we just need to divide the p-value by 2. Similarly, to calculate the non-directional p-value from a directional p-value we just need to multiply the p-value by 2.

```r
# Get the non-directional p-value from our model
non_directional_model_p_value <- summary(mod_old)$coefficients[2, 4]

# Calculate the directional p-value by dividing by 2
directional_model_p_value <- non_directional_model_p_value / 2

directional_model_p_value
```

```
[1] 1.744551e-80
```

Finally, with our directional p-value, we can decide whether or not we are going to reject the null hypothesis.

The decision to reject the null hypothesis is similar to a logical argument known as proof by contradiction. That is, we assume the null hypothesis is correct and then see if our test statistic is "far enough away" from the null hypothesis that it would contradict the null hypothesis. The issue is defining "far enough away". Statisticians refer to "far enough away" as the alpha ($\alpha$) value, it is the value that the p-value needs to be smaller than in order for us to conclude that our test statistic contradicts the null hypothesis as it is "far enough away" from the value specified by the null hypothesis. The alpha value is almost always set at .05, which means we need a p-value less than .05 in order to reject our null hypothesis.

So, because our directional p-value is less than .05 (p-value = 0) and our test statistic is positive, we can reject our null hypothesis at the .05 level. Remember, though, this does not mean we can accept our alternative hypothesis, it is still incumbent upon the researcher to provide a strong, logical argument as to why their alternative hypothesis is correct and not another—statistics can only get us so far.

Now, for the sake of example, let us conduct a hypothesis test for the intercept value:

- $H_0 : \beta_0 = 0$
- $H_a : \beta_0 > 0$.

In the code below, we are going to calculate the test statistic and its directional p-value.

```
# Go into the model coefficient table and retrieve the estimated intercept
estimated_intercept <- summary(mod_old)$coefficients[1, 1]

# Go into the model coefficient table and retrieve the intercept SE
intercept_se <- summary(mod_old)$coefficients[1, 2]

# Calculate the test statistic
test_stat_int <- (estimated_intercept - 0) / intercept_se

# Calculate directional p-value
directional_p_value <- 1 - pnorm(test_stat_int, mean = 0, sd = 1)

# Calculate non-directional p-value for learning
non_directional_p_value <- 2 * directional_p_value
```

```
# Print directional and non-directional p-value
tibble::tibble(`Directional p-value` = round(directional_p_value, 2),
           `Non-directional p-value` = round(non_directional_p_value, 2),
           `Model Calculated p-value` = round(
             summary(mod_old)$coefficients[1, 4], 2)
           ) |>
  knitr::kable()
```

Table 3: Directional and Non-Directional P-Values

| Directional p-value | Non-directional p-value | Model Calculated p-value |
|---:|---:|---:|
| 0.44 | 0.88 | 0.88 |

From the output of the code (Table 3), we can see the directional p-value is 0.44, which is greater than .05. Thus, we cannot reject our null hypothesis, which is not the same as accepting the null hypothesis—future studies with more observations may reject the null. In a research paper, we would have to conclude that more research is needed, but in this example we can feel safe saying that the true value of the intercept is 0—if you do not pump gas, you do not owe money.

With an understanding of NHST, we can now move to the last statistical inference topic: confidence intervals.

### Understanding Confidence Intervals in Linear Regression

NHST is good (depending on who you ask) for providing us with a simple reject/do not reject decision, but sometimes the more interesting question is what range of parameters would lead me to not reject the null hypothesis. What range of parameters are compatible with my data? This is where the confidence interval comes into play.

The confidence interval is an odd concept. It is the interval in which if we were to resample our data 100 times (similar to how we built the sampling distribution) and create an interval around our slope estimate using the confidence interval formula, then roughly 95 of the intervals would contain the the true slope parameter, 4, and 5 would not. Thus, you will hear researchers say "We are 95% confident that the parameter value is contained within the confidence interval." This, unfortunately, does not mean that there is a 95% probability that the parameter falls within the interval.

The other way to think of the confidence interval is as a compatibility interval. The 95% confidence interval tells us the range of null values that would result in a non-directional p-value greater than .05. That is, it is a range of null values for which we would not be able to reject the null hypothesis; the range of parameter values that are compatible with our data.

Let us work through our example to see how this works. First, we need to calculate the confidence interval according to this formula:

$$\text{Confidence Interval}_{.95} = \text{Estimate} \pm \Phi_{.975} \times \text{Standard Error}$$

where $\Phi_{.975}$ is the inverse normal CDF.

Think for a moment what the normal CDF does: it takes a real number as input and outputs the percentile at which that number falls on the normal curve. For example, the number 1 corresponds to the 84th percentile on a normal distribution with a mean equal to 0 and standard deviation equal to 1—roughly 84% of values fall below 1 on this curve.

The inverse normal CDF, $\Phi_{.975}$, does the opposite of the normal CDF. It takes a percentile as input, .975 in this case, and it outputs the number that corresponds to this percentile on a normal distribution. So, if as input we used 0.84, then as output we would get 1.

Now we can move on to some code to see how the confidence intervals works as a compatibility interval.

```r
# Retrieve the slope and standard error estimates from our model
slope_est <- mod_old$coefficients[2]
slope_se <- summary(mod_old)$coefficients[2, 2]

# Calculate the value that lies at the 97.5th percentile on a normal curve
inv_norm_cdf <- qnorm(.975, mean = 0, sd = 1)

# Calculate the lower end point of our confidence interval
conf_int_lower <- slope_est - inv_norm_cdf * slope_se

# Calculate the upper end point of our confidence interval
conf_int_upper <- slope_est + inv_norm_cdf * slope_se
```

Table 4: 95% Confidence Intervals for Regression Slope

| Slope Est. | SE | Inverse Norm. CDF | Lower CI Value | Upper CI Value |
|---:|---:|---:|---:|---:|
| 3.99 | 0.06 | 1.96 | 3.86 | 4.11 |

Table 4 contains the confidence interval values obtained from the code above.

The columns labeled "Lower CI Value" and "Upper CI Value" tell us that if we specified a null hypothesis using any null value between 3.86 and 4.11, the non-directional p-value would be greater than .05. That is, these are the parameter values that are most compatible with our data. Let us calculate a few of these p-values to test this out.

```r
# Create a vector that contains values between the lower and upper CIs
null_range <- seq(conf_int_lower, conf_int_upper, by = .001)

# Retrieve the slope and standard error estimates from our model
slope_est <- mod_old$coefficients[2]
slope_se <- summary(mod_old)$coefficients[2, 2]

# Calculate test statistics for each of the null values in null_range
test_stat_vector <- (slope_est - null_range) / slope_se

# Calculate non-directional p-values for each test stat
p_value_vector <- (1 - pnorm(abs(test_stat_vector), mean = 0, sd = 1)) * 2
```

```r
ggplot2::ggplot(
  data = data.frame(p_value = p_value_vector, par_range = null_range),
  ggplot2::aes(x = par_range, y = p_value)
) +
  ggplot2::geom_point(color = "lightblue") +
  ggplot2::labs(x = "Slope Parameter", y = "P-Value") +
  ggplot2::scale_y_continuous(breaks =
                                round(seq(
                                  min(p_value_vector), max(p_value_vector),
                                  by = 0.1
                                ), 3))
```

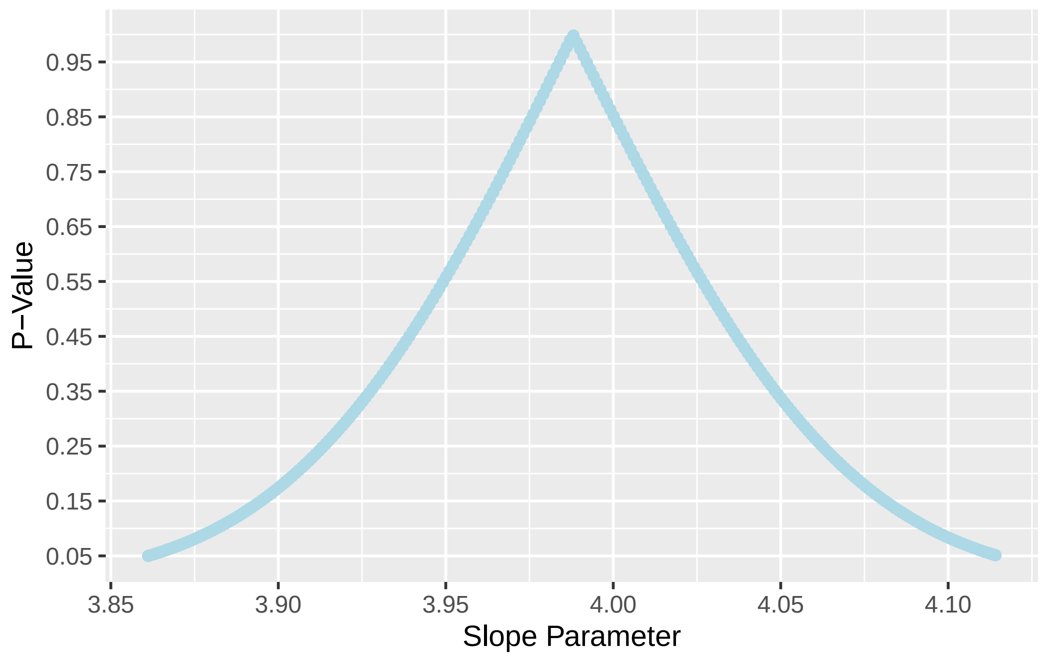


Figure 8: Scatter Plot of Slope Parameters and P-Values within the CI

Figure 8 shows the relationship between the parameter values that fall within the confidence interval and their non-directional p-values. The p-values range from 0.05 to 0.998 and are increasing from about .05 to 1 when moving from the parameter value equal to the lower value of the confidence interval (3.86) to the parameter value equal to the estimated slope (3.99), which is the parameter most compatible with our data, and decreasing from 1 to .05 when moving from the parameter value equal to the estimated slope to the parameter value equal to the upper value of the confidence interval (4.11).

In the context of our gas pump example, we can interpret the confidence intervals as we are 95% confident that the true cost of rate of dollars per gallon is between 3.86 dollars per gallon and 4.11 dollars per gallon. Or, that rates between 3.86 dollars per gallon and 4.11 dollars per gallon are all compatible with our data.

A powerful thing to recognize about the 95% confidence interval is that if 0 is not included in the confidence interval, then we know that we can reject the null hypothesis at an alpha level of .05 because any value outside of the 95% confidence interval will have a non-directional p-value less than .05. Thus, confidence intervals give us everything that hypothesis tests do and more!

## Linear Regression Assumptions & Diagnostics

With interpretation and inference behind us, it is time to discuss the assumptions underlying simple linear regression and the tools available to assess if our model seriously violates those assumptions.

### Linear Regression Assumptions

When we use the linear regression model we have to make four assumptions:

1. Predictors have an additive, linear effect on the outcome
2. Errors are independent of one another
3. Error variance is equal across levels of predictor
4. Errors are normally distributed

The first assumption, which we will refer to as additivity and linearity, is about the functional form of the relationship between the predictors and the outcome: each independent variable has an additive and constant effect on the outcome variable. We saw this assumption earlier when we specified a linear relationship between gallons pumped and cost of gas. The linear assumption states that the effect of the predictor variable, `gallons_pumped`, is constant across all values of the predictor and outcome variables. This is violated when our predictor and outcome variable exhibit nonlinear relationships with one another. An example of such a relationship is if gas became cheaper the more you pumped—a discount—in this example the relationship between `gallons_pumped` and `cost` is changing, it may start off steep, say $4 per gallon, and then become more level (cheaper) as you pump more gas, say $2 per gallon. Figure 9 contains a plot of this example. You can see the two lines are not continuous, there is a sharp break at 18 gallons where the discount takes effect (in this example it would be a discount for semi-trucks that require a lot more gas than a family Toyota). A linear regression model would not be able to adequately capture this relationship.

```r
gallon_pumped <- seq(1, 50, by = .05)

data_nonlinear <-
  tibble::tibble(
    gallon_pumped,
    cost = dplyr::if_else(gallon_pumped < 18,
                          4 * gallon_pumped,
                          2 * gallon_pumped)
  )

ggplot2::ggplot(
  data = data_nonlinear,
  ggplot2::aes(
    x = gallon_pumped,
    y = cost
  )
) +
  ggplot2::geom_point(
    color = "lightblue"
  ) +
  ggplot2::labs(
    x = "Gallons Pumped",
    y = "Cost in Dollars"
  )
```

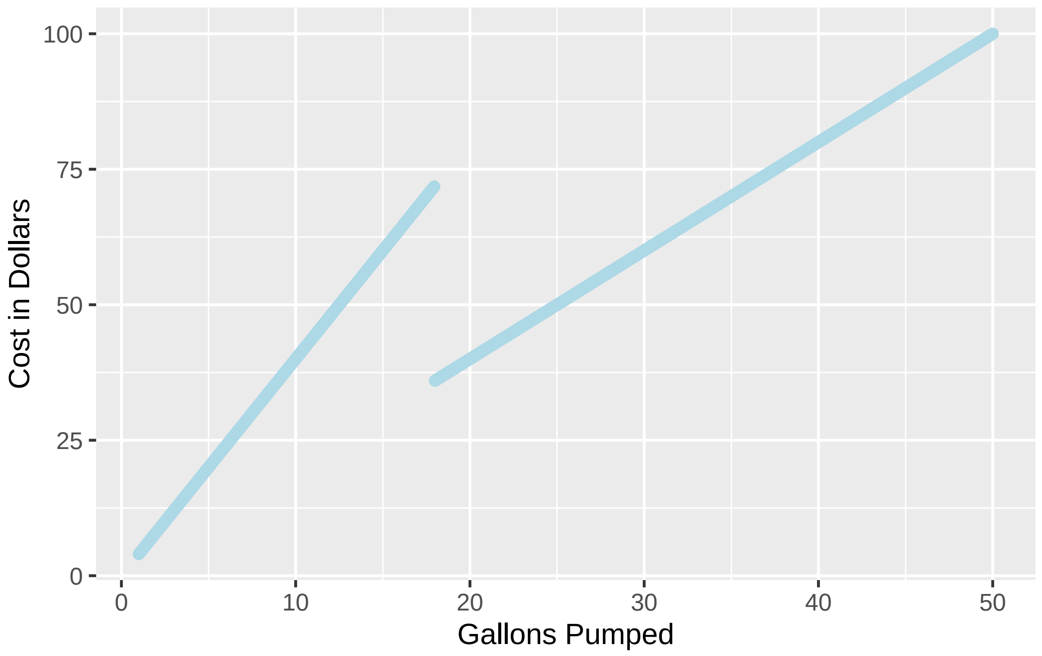


Figure 9: Example of a Nonlinear Relationship

The remaining assumptions are all about the model errors or residuals, so before we talk about the assumptions we need to make sure we have a good understanding of what a model error is. The model error is really the noise, the uncertainty that is present in our outcome variable after we adjust for the effects of the various predictors. This is easier to see when we look at the functional form representation of our gas example:

$$\text{Cost}_i = 0 + \beta_1 \times \text{Gallon}_i + \epsilon_i, \epsilon_i \sim N(0, \sigma)$$

$$\text{Cost}_i - \beta_1 \times \text{Gallon}_i = \epsilon_i.$$

The second equation above is the way to mathematically represent adjusting for a predictor variable—we are removing the effect of the predictor variable from the outcome variable. All that remains after this adjustment is the uncertainty, $\epsilon$, which we can refer to as the model error, residual, or noise. It is this error on which our remaining assumptions focus.

The second assumption is about the independence of our errors. In our gas example, we assume that the error or noise in our first observation is independent of the errors made in all of our subsequent observations. Practically, this means that our errors do not add up and we can safely average over them. This assumption quickly unravels when we believe that our observations are correlated across time like a stock price for a single stock over the course of days.

The third assumption is about the error variance or $\sigma^2$. This assumption states that the error variance is constant across all values of our predictor variable. Statisticians refer to this as homoscedasticity. This assumption means that the uncertainty in our outcome variable neither increases nor decreases as we go up or down on our predictor variable. In the context of our example this means that at a given amount of gas pumped, we expect the observed cost to vary around its average of $4 \times$ Gallons Pumped by about \$1.50, on average—the standard deviation of the error we specified in our simulation. When we pump 4 gallons of gas, we expect about 68% of our cost observations to be between \$14.50 and \$17.50 (1.50 +/- a mean of 4 * 4 or 16) and when we pump 10 gallons of gas, we expect about 68% of our cost observations to be between \$38.50 and \$41.50 (1.50 +/- a mean of 4 * 10 or 40)—the mean has changed as we have pumped more gas, but the variance (and its standard deviation) has remained constant.

The fourth assumption is about the distribution of the model errors—the probability distribution that generates our uncertainty. To use the linear regression model, we have to assume that uncertainty in our outcome variable (the model error) was generated by a normal distribution—even if just approximately. This means that at any value of the predictor variable our observations should fall along a bell curve around their mean. Using our gas example, this means when we pump 10 gallons of gas, we should see our observed costs fall symmetrically around a mean of \$40 with most of the observed costs falling close to \$40 and very few costs falling less than \$37 or more than \$43 (two standard deviations away from the mean of \$40).

With all of our assumptions laid bare, let us now look at our statistical model written as a probability model and highlight the assumptions it is making:

$$Y_i | X_{1i} \sim N(\mu_i = \beta_0 + \beta_1 X_{1i}, \sigma).$$

First, lets talk about the subscript, $i$. This denotes the specific observation on which we are focused and it can take on integer values ranging from 1 to the number of observations in our dataset. In our gas example, $i$ can take on integer values 1 to 100, where $i = 1$ means we are focused on the first observation in our dataset, $i = 27$ means we are focused on the 27th observation in our dataset, and so on. Thus, $X_{1i}$ means the *ith* observation the predictor variable, $X_1$, and $Y_i$ means the *ith* observation on the outcome variable.

Looking at the probability model, we see that the first assumption, linearity and additivity, comes into play through the mean of the probability model: $\mu_i = \beta_0 + \beta_1 X_{1i}$. This tells us that the mean of the *ith* outcome is linearly related to the value of the *ith* predictor variable, $X_1$. A one unit increase in the predictor variable will lead to a $\beta_1$ increase in the outcome variable's mean, always. Using our gas example, if we took all of the observed values of cost where we pumped exactly 10 gallons of gas and averaged those cost values, the average would be right around \$40 or $4 \times$ Gallons Pumped. If we then took all of the observed values of cost where we pumped exactly 11 gallons of gas (a one unit increase from 10 gallons) and averaged those cost values, the average would be right around \$44—a \$4 increase in the average cost.

Moving onto the assumption of independent errors, this assumption is more subtle and harder to see in the probability model. This assumption is subtle in that it is the absence of something that signals this assumption: the absence of a more complicated probability model. If we believe our errors are not independent of one another, then we need to specify something more complicated than a normal distribution with a single error variance component, $\sigma$. Without getting overly complicated, we would have to specify something called an error structure—enough on that.

Similar to the assumption of independent errors, it is the absence of something that shows us the assumption of constant error variance: a lack of an $i$ subscript on the error standard deviation, $\sigma$. This means that after adjusting for our predictor variables, the variance of $Y_i$ will be $\sigma$ for all observations.

Finally, the assumption that our errors follow a normal probability distribution is signaled by the presence of the normal distribution function symbol: $\sim N()$ where $\sim$ is a symbol for "distributed" and $N$ is a symbol for normal distribution meaning "distributed normally". Together, these last three assumptions about the error serve to simplify our probability model. We need only specify a normal probability distribution function with a single, constant error variance component, $\sigma$, to describe the uncertainty in our model.

## Linear Regression Diagnostics

At this point, we have gone from an initial exploration of our data to estimating and interpreting a regression model to making inferences about a population model from our estimated regression model. But for us to have confidence in our model predictions and inferences, we need to determine how badly our model assumptions are being violated by our data. To do this, we will rely on a set of graphical and statistical tests. We start by discussing data points that are not well predicted by our model. Following this, we then explore methods to determine if our model assumptions are badly violated. Often times unusual data points will lead us to a better understanding of the ways in which our model assumptions are violated, which will hopefully help us determine the adjustments we need to make to improve our model such as by modifying our linear function to capture a nonlinear effect.

### Leverage, Outliers, and Influence

One of the first things we look for to determine if our model is correctly specified is to plot our model residuals (errors) against the values of our predictor variables. We want to look out for unusual observations, which are observations that have an unusually large residual, an unusually large predictor value, or both.

An observation with an unusually large residual is called an outlier—an observed outcome that our model is unable to predict well. An observation with an unusual predictor value (either much larger or smaller than the average value) is called a leverage point. By themselves, outliers and leverage points may not harm the inferences we are making from our model, but together they can drastically harm those inferences. An influential data point is a data point that is both an outlier and a leverage point. Influential points should alert us that something is not right with our data, our model, or both. It can signal that the observation was incorrectly coded—the outcome value, predictor value, or both could have been entered incorrectly. Or it can signal that our model is misspecified and is unable to describe an important pattern in our data.

To get a better understanding of the difference between outliers, leverage points, and influential points, let us conduct another simulation. This time we are going to simulate three separate datasets: one that has an outlier, one that has a leverage point, and one that has an influential point. This will let us explore the effects that each point has on our model.

```r
set.seed(9999)

n_visits <- 100 # Pump vista

gallons_pumped <- sample(
  4:15,
  n_visits,
  replace = TRUE,
  prob = dnorm(4:15, mean = 10, sd = 2) / sum(dnorm(4:15, mean = 10, sd = 2))
)

noise <- rnorm(n_visits, mean = 0, sd = 1.50)

cost_old <- 4 * gallons_pumped + noise

# Create a dataset that contains all of our simulated data
data_unusual_points <-
```

```r
  tibble::tibble(point_id = "regular",
                 gallons_pumped = gallons_pumped,
                 cost = cost_old)

# Add unusual points
data_unusual_points <-
  data_unusual_points |>
  dplyr::mutate(point_id = "Outlier") |>
  dplyr::bind_rows(tibble::tibble(
    point_id = "Outlier",
    gallons_pumped = 10,
    cost = 500
  )) |>
  dplyr::bind_rows(
    data_unusual_points |>
      dplyr::mutate(point_id = "Leverage Point") |>
      dplyr::bind_rows(
        tibble::tibble(
          point_id = "Leverage Point",
          gallons_pumped = 25,
          cost = 100
        )
      )
  ) |>
  dplyr::bind_rows(
    data_unusual_points |>
      dplyr::mutate(point_id = "Influential Point") |>
      dplyr::bind_rows(
        tibble::tibble(
          point_id = "Influential Point",
          gallons_pumped = 25,
          cost = 500
        )
      )
  ) |>
  dplyr::mutate(point_id = factor(
    point_id,
    levels = c("Outlier", "Leverage Point", "Influential Point")
  ))

plot_unusual_point <-
  ggplot2::ggplot(data = data_unusual_points,
                ggplot2::aes(x = gallons_pumped, y = cost)) +
  ggplot2::geom_point(
    fill = "lightblue",
    color = "black",
    size = 3,
    shape = 21
  ) +
  ggplot2::geom_smooth(
    method = lm,
    se = FALSE,
    formula = y ~ x,
    color = "black",
    alpha = .50
  ) +
  ggplot2::facet_wrap( ~ point_id, scales = "free") +
  ggplot2::labs(x = "Gallons Pumped", y = "Cost in Dollars")
```

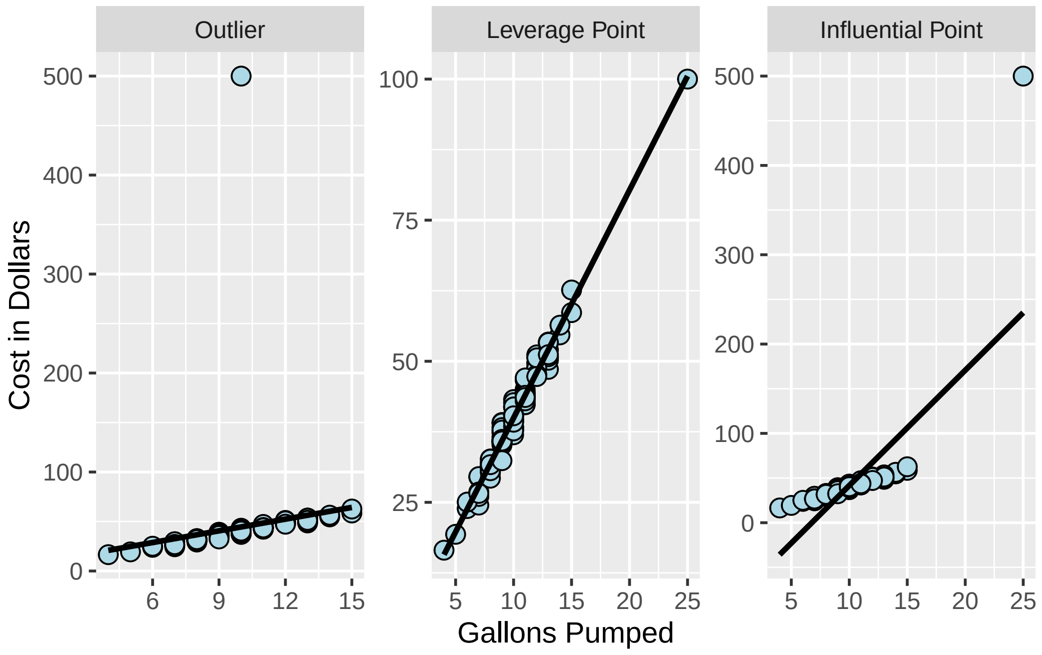


Figure 10: Effects of Outliers, Leverage, and Influence

Moving from left to right on Figure 10, we see the effects an outlier, a leverage point, and an influential point have on the estimated regression line (the line appearing on the plot). The outlier does not impact the estimated regression line too much because it occurs with a predictor value that is close to the mean of the predictor variable and, thus, the regression model can essentially ignore the outlier as there is enough information in that range of the predictor variable (i.e. the bulk of the data are centered around there). Next, the leverage point does not impact the estimated regression line too much because the leverage point follows the overall pattern seen in the remaining data—it falls along the predicted regression line. Finally, the influential point has a noticeable impact on the estimated regression line because the influential point takes on an extreme predictor value (`gallons_pumped` = 25) and the data point does not follow the pattern seen in the remaining data—we would expect a predictor value of 25 gallons to have a cost of \$100, not \$500.

All three types of unusual points: outliers, leverage points, and influential points should prompt you to investigate your data further as they could lead you to discover that some of the data was entered incorrectly or that you need a new or revised theory to account for the extreme observations. But only influential points, a data point that is both an outlier and a leverage point, will really hurt the inferences you are making from your estimated model. As their name implies, influential points will influence your model estimates and their standard errors.

Beyond plotting your data, there are several statistics we can calculate to identify outliers, leverage points, and influential points:

- Studentized Residual to identify an outlier
- Hat values to identify a leverage point
- Cook's Distance to identify an influential point.

To identify an outlier, we can calculate a studentized residual from our model, which is a special, standardized version of your model residuals (scaling your residuals so that they have a standard deviation of 1). An outlier is an observation that has a studentized residual greater than 2 in absolute value.

To identify a leverage point, we can calculate a metric called a hat-value, $h_i$, for each observation. A hat-value essentially tells us how far away a predictor value is from its average. A leverage point is an observation with a hat-value greater than double the average hat-value.

To identify an influential point, we can calculate a metric called Cook's Distance, $D_i$, for each observation. Cook's Distance is the calculated, roughly, by multiplying an observation's standardized residual by its hat-value—it is a function of Outlier × Leverage. An influential point is an observation with a Cook's Distance value greater than about $\frac{4}{n-2}$.

Thankfully, R has functions that let us easily compute an observations studentized residual, hat-value, and Cook's Distance.

```r
# Each of the functions below require an estimated regression model as
# their argument:

# Calculate studentized residual
MASS::studres(mod_old)
```

```r
# Calculate Hat-values
hat(mod_old)

# Calculate Cook's Distance
cooks.distance(mod_old)
```

#### Testing Regression Assumptions

Even if you do not find any unusual observations in your data, it is still important to investigate if and how your model assumptions are being violated. The best way to investigate these violations is through the use of plots and statistical tests.

The first assumption we will look at is the assumption of linearity and additivity—does a line adequately capture the relationship between our outcome variable and predictor variables. To see how this violation manifests itself in data, we are going to simulate two different types of nonlinear relationships between and outcome and predictor variable. The first nonlinear relationship will be a quadratic relationship between the outcome and predictor: $Y = \beta_0 + \beta_1 X_1 + \beta_2 {X_1}^2$. The second nonlinear relationship will be an exponential relationship between the outcome and predictor: $Y = \beta_1 e^{\beta_2 X_1}$.

```r
# Simulating nonlinear relationships
set.seed(16)
n <- 500
x <- rnorm(n)

y_quad <- 9 + x + - .75 * x^2 + rnorm(n, sd = 2.5)
y_exp <- .75 * exp(-.50 * x + rnorm(n, sd = .75))

data_nonlinear <-
  tibble::tibble(
    x,
    y_quad,
    y_exp
  )

# Fitting linear models to nonlinear data
mod_quad <- lm(y_quad ~ x, data = data_nonlinear)
mod_exp <- lm(y_exp ~ x, data = data_nonlinear)

# Getting Studentized Residuals & creating a data frame for plotting
data_nonlinear_plot <-
  tibble::tibble(
    id = rep(c("Quadratic", "Exponential"), each = n),
    x = c(x, x),
    stud_res = c(MASS::studres(mod_quad), MASS::studres(mod_exp))
  )

# Plotting residuals against predictors
plot_linear_check <-
  ggplot2::ggplot(
    data = data_nonlinear_plot,
    ggplot2::aes(
      x = x,
      y = stud_res
    )
  ) +
  ggplot2::geom_point(color = "black", fill = "lightblue",
                      shape = 21) +
  ggplot2::facet_wrap(~ id, scales = "free") +
  ggplot2::geom_smooth(se = FALSE) +
  ggplot2::labs(
    x = "Predictor Variable",
    y = "Studentized Residuals"
  )
```

Figure 11 contains the predictor values plotted against their studentized residuals. Remember, a residual is an outcome variable adjusted for the linear effects of the predictor variables, so when you plot a residual against its predictor you should not see any relationship between the two if your model adequately captures the relationship between the outcome and predictor variables. Looking Figure 11, thanks to the smoothed line added by `geom_smooth`, we can see that their appears to be a nonlinear relationship between the residuals and their predictors.

It is fairly straightforward to model nonlinear effects using a linear regression model. If you know the type of nonlinear relationship you may be able to transform either your outcome or predictor variables to model or remove the nonlinear effect. For example, if we knew the true relationship between our outcome and predictor variables was an exponential relationship, then we could transform our outcome variable by taking the natural log of the outcome variable ($\ln(Y)$) and use the transformed outcome in the regression model (note this makes it harder to interpret the model estimates). If, however, you do no know the type of nonlinear relationship, then you can add polynomial transformations of your predictor variable into your model (e.g. $X^2, X^3, ...$) in order to approximate the relationship. I would only recommend going as high as a second ($X^2$) or third ($X^3$) degree polynomial. We will look at an example of this now.

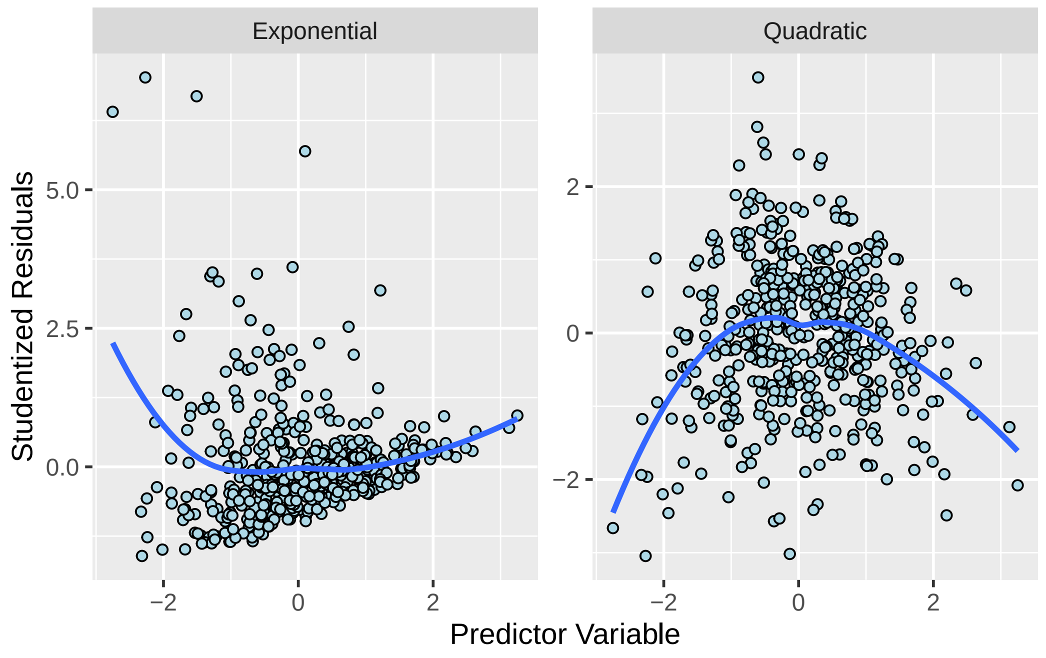


Figure 11: Looking for Nonlinear Relationships with Residual Plots

```r
# Let's create some transformed variables in our data frame
data_nonlinear <-
  data_nonlinear |>
  dplyr::mutate(
    log_y_exp = log(y_exp),
    x_sq = x^2,
    x_cube = x^3
  )

# Let's fit models with these transformed variables
mod_log_transform <- lm(log_y_exp ~ x, data = data_nonlinear)
mod_exp_approx <- lm(y_exp ~ x + x_sq + x_cube, data = data_nonlinear)

# Now let's examine the residual plots
# Getting Studentized Residuals & creating a data frame for plotting
data_transform_plot <-
  tibble::tibble(
    id = rep(c("Log Transform", "Polynomial Approximation"), each = n),
    x = c(x, x),
    stud_res = c(MASS::studres(mod_log_transform), MASS::studres(mod_exp_approx))
  )

# Plotting residuals against predictors
plot_linear_transform_check <-
  ggplot2::ggplot(
    data = data_transform_plot,
    ggplot2::aes(
      x = x,
      y = stud_res
    )
  ) +
  ggplot2::geom_point(color = "black", fill = "lightblue",
                      shape = 21) +
  ggplot2::facet_wrap(~ id, scales = "free") +
  ggplot2::geom_smooth(se = FALSE) +
  ggplot2::labs(
    x = "Predictor Variable",
    y = "Studentized Residuals"
  )
```

As you can see in Figure 12, we were able to use a linear regression model with transformed variables to model nonlinear relationships. The smoothed lines in Figure 12 are not nearly as curved as they were before which indicates that we have approximately (or completely with the log transform model) modeled the nonlinear relationship between our outcome and predictor variable.

The next assumption we will check is the independence of our model errors. Unfortunately, unless the violation is particularly bad, plots do not help us identify this violation. Although, for the sake of example, we will look at several plots of studentized residuals to see how this violation appears graphically. As Figure 13 demonstrates, we can look for violations of independence by looking at a scatter plot the model residuals. If we see noticeable grouping, as we see in two of the plots, then we may begin to suspect that our model's assumption about independent errors may be violated. This violation can severely impact our standard errors, which hurts our ability to draw correct inferences from our model.

Dependent errors tend to occur when our data is subject to some kind of hierarchy (e.g. students nested in a classroom or multiple observations nested within a single person) or temporal ordering such as time series data. When our data has some inherent hierarchy to it, we can use a more general class of linear models to correct for this, but those models are beyond the scope of this primer.

The third assumption we need to check is that our error variance is constant across the levels of our predictor variables. We can check this assumption by plotting our studentized residuals against our predictor variable—exactly like we did to check the linearity and additivity

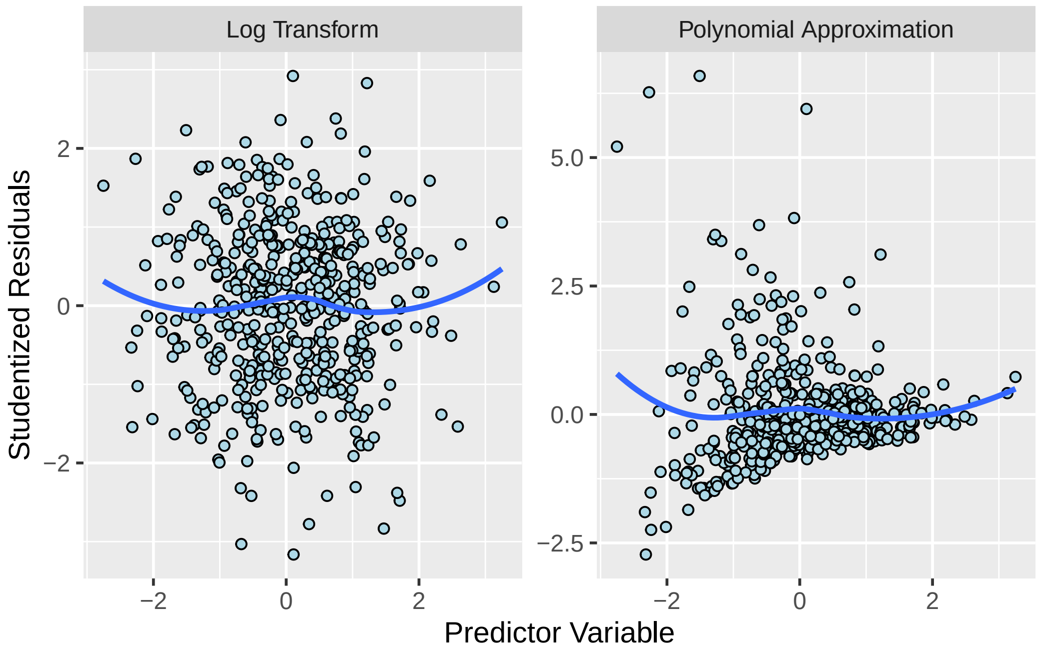


Figure 12: Residual Plots with Nonlinear Effect Removed

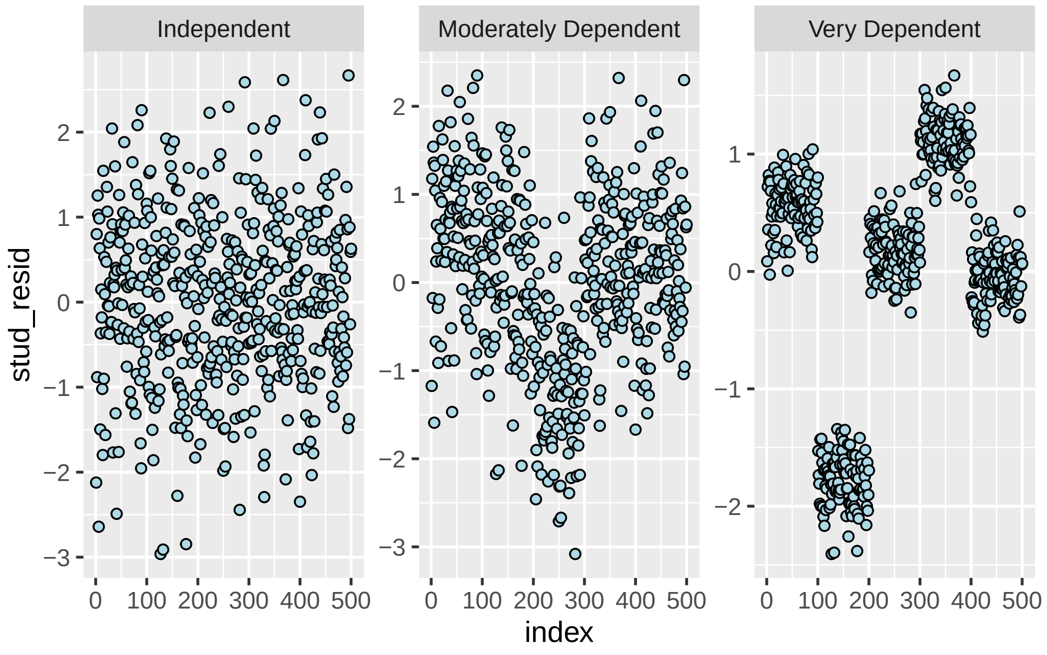


Figure 13: Residual Plots for Violations of Independent Errors

assumption—and visually inspect if our residuals become more or less spread out as we go up on our predictor variable. Figure 14 contains two plots, one where our errors are constant and one where they are not constant.

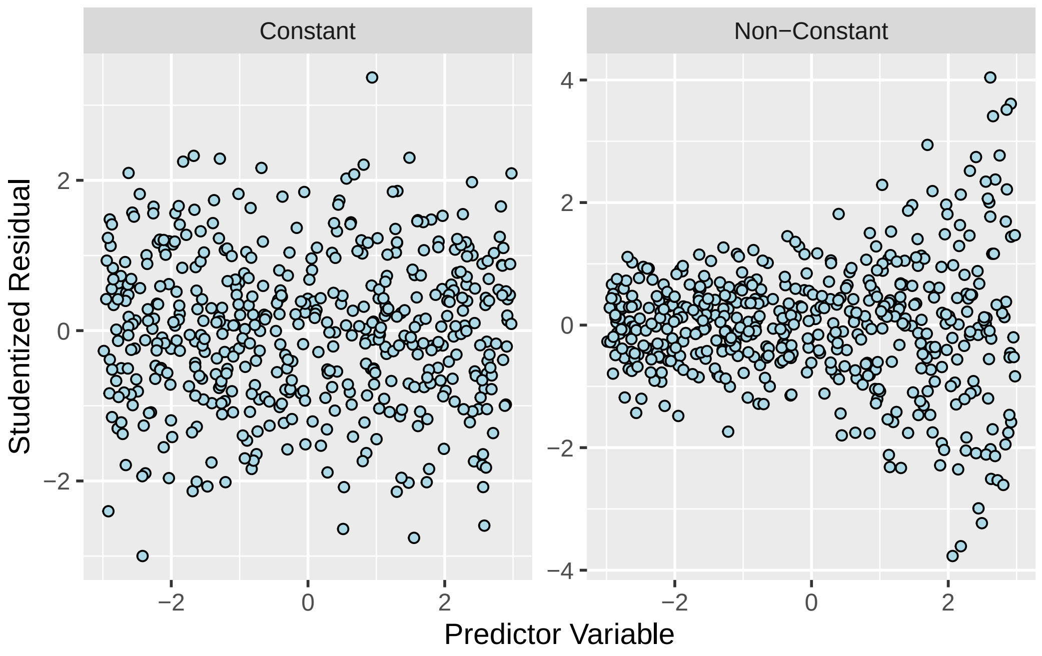


Figure 14: Residual Plots for Non-constant Error Variance

Looking at Figure 14, we see for the plot labelled "Constant" that the spread of of the residuals does not change as we look up or down the predictor values, whereas in the plot labelled "Non-Constant" we see that the residuals are noticeably more spread out as we move up on the predictor variable. This is a clear sign of non-constant error variance. To correct for this violation, we need to use a more advanced regression model known as a weighted least squares model, which is outside the scope of this primer.

Finally, the last assumption we need to check also has the least amount of impact on our inferences, the normality of the residuals. This check is fairly straightforward. We can either use a histogram to plot the distribution of the studentized residuals and examine how closely it follows a normal distribution or we can create a normal quantile-quantile plot which shows us if the the percentiles of our residual distribution match the percentiles of a normal distribution. We can see an example of both plots below in Figure 15 for residuals that follow a normal distribution.

The first thing to notice in Figure 15 is the plot on the left is the distribution of the outcome variable and the plot on the right is the distribution of the residuals—the outcome variable after we have adjusted for the effects of our predictor variables. Notice that the outcome variable distribution has a slight right skew whereas the studentized residuals look more normally

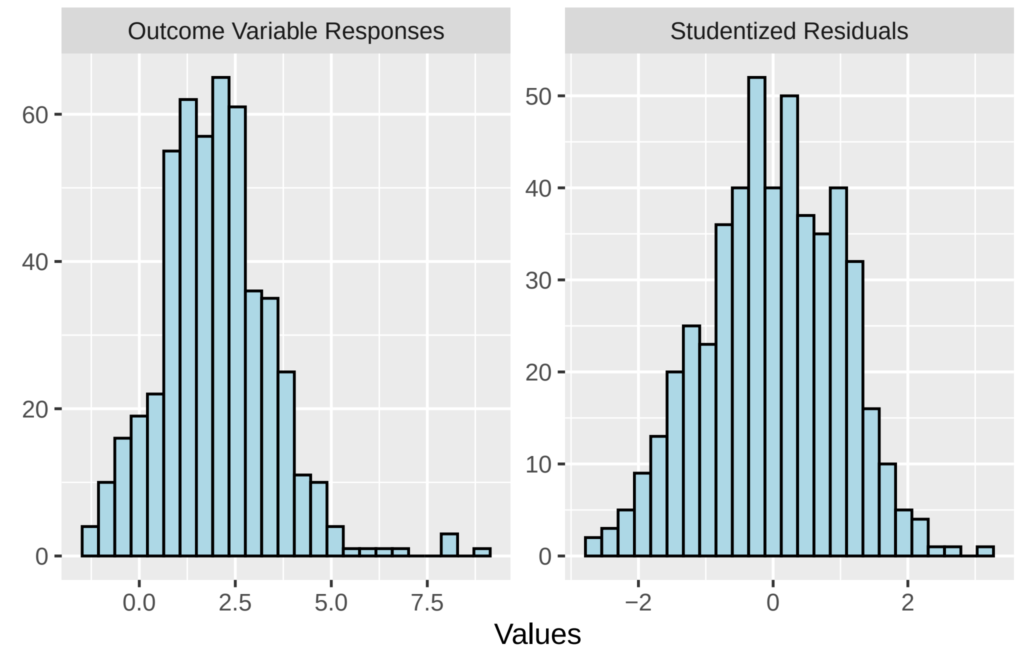


Figure 15: Distribution of Model Residuals

distributed. This is an important distinction: this assumption is not focused on the distribution of the outcome variable, it is focused on the distribution of the errors (residuals). It is possible for our outcome variable to be non-normally distributed even if our errors are normally distributed. If you check this assumption using a distribution of the outcome variable, then you may arrive at the incorrect conclusion that it is violated.