Practicing R: A Review of STA120

One Nominal Variable (Chapter 1)

Remember Buzz and Doris? The two dolphins trained to hit a left button or right button depending on which button was flashing? The experiment involved 16 trials and was designed to test whether Buzz and Doris could communicate. Of these 16 trials 15 were successes (i.e., Buzz or Doris hit the flashing button). That's a 94% success rate.

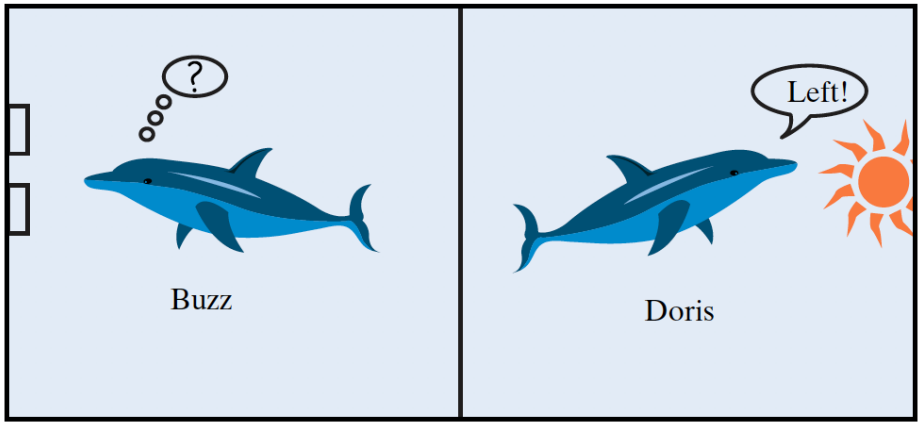


FIGURE 1.1 Depending whether or not the light was blinking or shown steadily, Doris had to communicate to Buzz as to which button to push.

First question: How are these data being measured by the experimenter? That is, are we dealing with a categorical or continuous variable? If categorical, is the variable nominal or ordinal?

The experiment records success and failures. It's analogous to flipping a coin 16 times and recording either heads or tails. Success/failure and heads/tails are just names, so the experiment involves one categorical variable that we'll call "success."

Since this dataset is so simple, we'll construct it ourselves in R. First off, we need to quantify this qualitative variable, which we'll do by the following assignment:

$$success_i = \begin{cases} 0, & \text{Trial } i \text{ was a failure} \\ 1, & \text{Trial } i \text{ was a success.} \end{cases}$$

We've converted names (success or failure) to numbers (1 or 0).

Creating the Buzz-Doris Dataset

Let's create the dataset, which requires us to create a vector in R of 15 ones and 1 zero since there were 15 successful trials and 1 unsuccessful trial. To do this, we'll use the combine function, `c()`, putting 15 ones and 1 zero inside. Typing those 15 ones inside `c()`, each separated by a comma, is time consuming and could easily result in an error. Since we just need the same number 15 times, use the replicate-elements function,

`rep()`, using the arguments 1, `times = 15`. Then we'll just include a single 0 to indicate one time Buzz failed to hit the flashing button. So, to create this dataset run:

```
# Create a vector of 15 repeated ones and 1 zero
buzz_data <- c(rep(1, times = 15), 0)
```

To test the hypothesis of whether Buzz and Doris are in fact communicating or, alternatively, the results were due to chance, we need to derive a sampling distribution. A **sampling distribution** is a special type of probability distribution in which the variable distributed is a statistic.[1] There's only one piece of information from the experiment, the recorded data consisting of 15 successes and 1 failure. But in order to obtain evidence that Buzz and Doris were actually communicating we need to compare our sample statistic, $\hat{\pi}, = 0.9345$ against *all possible* statistics that could have resulted, provided the results were due to chance. Obtaining all possible statistics, here all possible rates of success in 16 trials, assuming the null hypothesis is true, yields a distribution — The sampling distribution to test our hypothesis. So, to summarize, a sampling distribution shows every possible statistic that could result in every possible sample from a population and how frequently each statistic results.

Simulating a Probability Distribution

There's a theoretical sampling distribution for experiments of this type called the *binomial distribution*, which provides exact test results. But let's first use R to create a simulated sampling distribution to test a simulated version of the hypothesis. We want to obtain randomly generated statistics for this type of experiment with 16 trials and under the assumption that the results were due to chance.

In an experiment such as this one, or analogously if the experiment were to count the number of heads after 16 tosses of a fair coin, what would be the (long-run) outcome if the result were due to chance? A flipped coin should land heads 50% of the time and similarly Buzz and Doris would guess the right button 50% of the time if the results were simply due to chance. Thus, the null value of the hypothesis test is 0.5 (50%).

Prior to simulating this sampling distribution, it's worth introducing a crucial function when simulating data. Because we're generating (pseudo) random numbers, each time you run the simulation you're bound to obtain different but similar data. But it's often helpful to reproduce results, for example, when in class when we want to work with the same simulations. To obtain reproducible results, use the `set.seed()` function prior to running the simulation. The only argument is an integer (any integer), which, provided everyone uses the same integer, generates the same simulated data. So let's run the following:

```
# Set random number seed for reproducible results
set.seed(6)
```

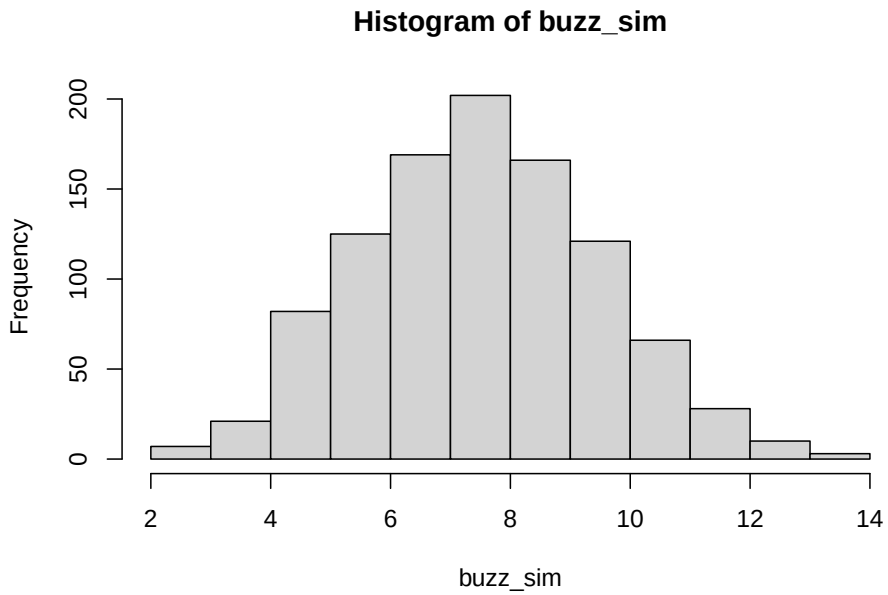
To obtain a simulated sampling distribution under these conditions, we use the `rbinom()` function, the "r" standing for random and the "binom" referring to the binomial distribution that we want to simulate. The arguments of `rbinom()` are the number of samples to take, n, the number of trials in the experiment, size, and the null value, prob. The following code draws 1000 statistics from an experiment with 16 trials and a null value of $\pi_0 = 0.5$:

```
# Binomial simulation; samples = 1000; sample size = 16; pi_0 = 0.5
buzz_sim <- rbinom(n = 1000, size= 16, prob =  0.5)
```

Notice that the simulation data that are added to the Environment pane (upper-right pane) consists of integers between 0 and 16, most of which are between 6 and 10. A quick way to get an impression of the simulated sampling distribution is to generate a histogram:

```
# Histogram of binomial simulation
hist(buzz_sim)
```

[1] Recall that a statistic is some property of a sample; for example, the mean of a sample. In this experiment there's a *sample* statistic, which is the proportion of times Buzz and Doris successfully hit the flashing button. Denote this sample proportion by $\hat{\pi}$ so that $\hat{\pi} = 15/16 = 0.9345$.



From the looks of the histogram the probability that Buzz and Doris would successfully hit the flashing button 15 is quite unlikely (actually 0), which suggests they were not guessing and in fact were communicating. In a hypothesis test we would reject the null hypothesis that $\pi_0 = 0.5$ in favor of the alternative hypothesis.

Computing Simulated and Theoretical *p*-Values

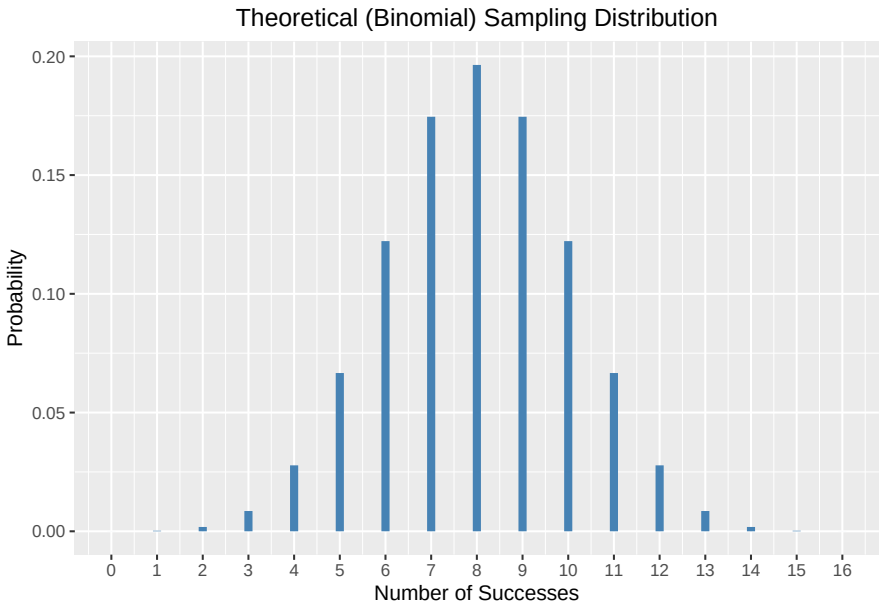
Let's calculate the simulated p-value associated with this histogram for a one-side hypothesis test. A p-value is short for a *probability* value, and probabilities are determined either by areas under a continuous curve, or in this non-continuous case, by summing up all values over a particular range. A p-value is probability of obtaining a statistic as extreme as the sample statistic, here $\hat{\pi} = 15$. Our one-sided alternative hypothesis is $\pi > 0.5$, that is, we conjecture that Buzz and Doris would be successful at hitting the flashing button more than half the time. Given the direction of the alternative hypothesis, we need to sum all simulated values greater than or equal to 15, and then divide by the number of samples (1000) to obtain a probability. In R we'll run the following code:

```
# Computing the simulated p-value
sum(buzz_sim >= 15)/1000
```

Finally, let's obtain the actual (theoretical/non-simulated) results of the hypothesis test. To do that, we'll use the `binom.test()` function by inputting as arguments the number of successes, `x = 15`, the number of trials, `n = 16`, the null-value from the null hypothesis, `p = 0.5` and the direction of the alternative hypothesis, `alternative = "greater"`.

```
# Theoretical p-value for Buzz and Doris experiment
binom.test(x = 15, n = 16, p = 0.5, alternative = "greater")
```

These theoretical results confirm our simulated ones. We can reject the null hypothesis that $\pi_0 = 0.5$ in favor of the alternative. The theoretical sampling distribution yields further evidence in favor of the alternative with a nearly invisible amount of probability massed above 15 and an invisible (but non-zero) amount massed at 16.



One Continuous Variable (Chapter 2)

Research Question: Can people accurately estimate the length of a short song snippet?

Some researchers asked 48 people to listen to a 10-second snippet of a song, and then record how long each student thought the song played for. Do people accurately estimate, on average, the true length of the song (10 seconds), or do they tend to over- or underestimate its length.

The hypothesis is two-sided because the researches don't have a theory for why people would over- or underestimate the snippet's length; people could be inaccurate in both directions from the song's true length (10 seconds). The hypothesis tests for this experiment therefore take the form:

$$H_0 : \mu = 10 \text{ seconds}$$
$$H_a : \mu \neq 10 \text{ seconds}$$

Observe that the variable being measure is time (measured in seconds), and time is a *continuous* variable: 10 seconds is 3 seconds more than 7 seconds, 13 seconds is 7 seconds less than 20 seconds, and so on.

Importing Data

The dot plot of these data show the distribution of each of the 48 students' estimated length of the 10 second song snippet.

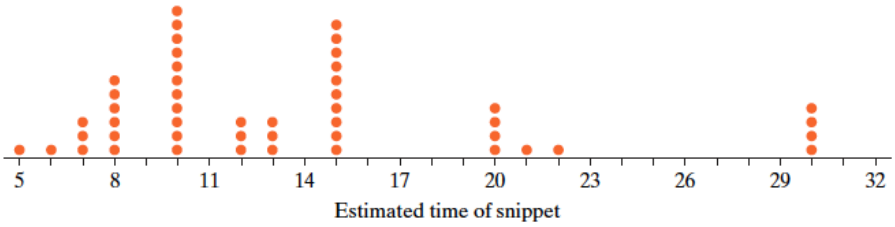


FIGURE 2.5 The results (in seconds) for 48 students trying to estimate the length of a 10-second song snippet.

As displayed, it'd be laborious to recreate these data in R. Fortunately, the textbook provides the dataset on their website and we can directly import it into R. The dataset's url is http://www.isi-stats.com/isi/data/chap2/TimeEstimate.txt, which, you'll notice, means these data were originally uploaded as a text file (.txt). You'll need to be conscious of the file format when uploading data because the functions used to import data depend on the file format. We use the `read_tsv()` function from the **tidyverse** package to import text files into R, inserting the additional `url()` function within `read_tsv()` to let R know these data are located on the internet. The following command imports our desired dataset:

```
# Importing a .txt file from internet
song <- read_tsv(url("http://www.isi-stats.com/isi/data/chap2/TimeEstimate.txt"))
```

Upon running the code, you should verify that the dataset was imported correctly. Make sure, for example, that the column heading is some string of letters (here, "Estimate") and that there are 48 observations (rows).

Computing the Sample Statistic

Now that we have our data we can compute the sample statistic. The (population) parameter we're interested is μ, the mean time length for the everyone in the population. Our goal is to make an inference about the population, that is, we want to say something about μ. But we only have a sample of 48 people from the population. The sample mean, $\bar{x}$, is the natural sample statistic to use in our hypothesis. To find the mean of some variable, use the `mean()` function. The argument will be the time lengths of the 48 people sample, which are listed in the Estimate vector from the song dataframe. To isolate a vector from a dataframe we use the $ symbol between the dataframe and the vector. In the present setup we'll need to input `song$Estimate` inside the `mean()` function; that is:

```
# Take mean of Estimate column from song dataframe
mean(song$Estimate)
```

```
## [1] 13.70833
```

The mean time length that participants of the experiment perceived the 10-second song to be playing is roughly 13.7 seconds. Thus, $\bar{x} = 13.7$.

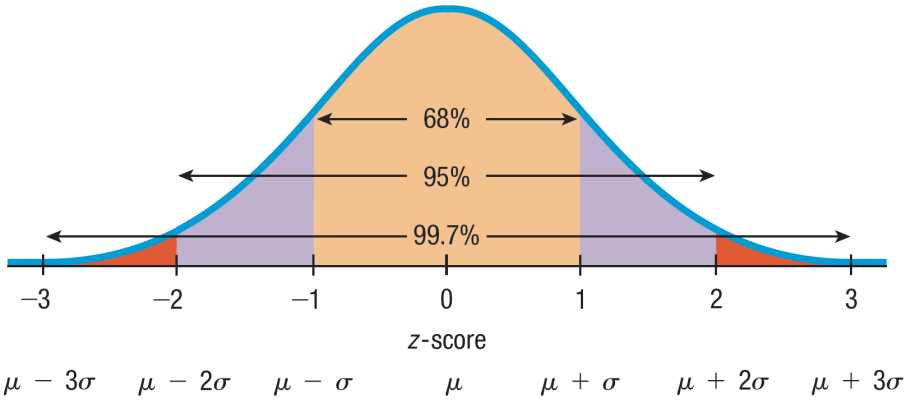
Digression: *Z*-Scores and the "Empirical Rule"

The standard deviation is often used as a ruler in the sense that it indicates how far a value is from the center of the distribution. Knowing some value is 7 units higher than the mean doesn't tell us much without knowing more context. (Are we measuring individual height in centimeters or inches or even feet. Depending on the measurement units, 7 units above the mean would mean wildly different things.) But measuring in standard-deviation units doesn't require us to know context: Knowing something is 7 standard deviations above the mean is absolutely extra-ordinary.

Before continuing, it will prove useful in what follows to keep in mind and to picture in your mind's eye the so-called "Empirical Rule." This rule refers to the fact that for any normal distribution:

- Roughly 66% of its values fall within 1 standard deviation from the mean
- Roughly 95% of its values fall within 2 standard deviations from the mean
- Roughly 99% of its values fall within 3 standard deviations from the mean

The Empirical Rule is most clearly visualized with the standard normal distribution; that is, that normal distribution with $\mu = 0$ and $\sigma = 1$, where μ is its mean and σ the standard deviation.



Note that this means values that are 2 standard deviations above or below the mean are very uncommon in a normal distribution since there's about a 5% probability that such a value would be in the left or right tail of the distribution.

It would clearly be useful if, instead of working with a given data set, we could take advantage of the Empirical Rule's properties. The previous example shows that converting values from their original units into standard-deviation units grants us a convenient way to quickly interpret data. But in certain situations it's absolutely necessary to make this conversion as the following example illustrates.

Example: 2016 Olympic Heptathlon

Nafissatou Thiam won the gold medal in the 2016 heptathlon. In the long jump she recorded a length of 6.58 meters, which was 0.5 meters higher than the mean. Katarina Johnson-Thompson won the 200 meter dash with a time of 23.26 seconds, which was 1.5 seconds faster than the mean time.

The question is: Who's individual performance was more impressive? I don't know enough about the long jump to know if 0.5 meters above the mean small or large; same with a run time 1.5 seconds faster than the mean. And even if I had stronger domain knowledge of heptathlon performance, I'd still want a objective way to compare these performances. The way to do this is by constructing z-scores, also known as *standardized scores*.

The idea is to express the distance from the mean in standard-deviation units, which then *standardizes* the performances. Just like rulers standardize distances, the standard deviation standardizes the distance of values from the mean.

To standardize some value, x_i, subtract the mean, $\bar{x}$, and then divide by the standard deviation, s_x:

$$z_i = \frac{x_i - \bar{x}}{s_x}.$$

The standardized value, z_i, is the distance of x_i from the mean $\bar{x}$ (the numerator), in standard deviations (by dividing by s_x).

Consider this table with the necessary information to calculate Thiam's and Johnson-Thompson's z-scores:

	Long Jump	200-Meter Dash
Mean	6.17 meters	24.58 seconds
Standard Deviation	0.247 meters	0.654 seconds
Individual Performance	6.58 meters (Thiam)	23.26 seconds (Johnson-Thompson)

Before computing these z-scores, let's concentrate on the long jump and see how 1 and 2 standard deviations above the mean compares to Thiam's jump of 6.58 meters. A long jump 1 standard deviation above the mean is: $6.17 + 1(0.247) = 6.417$. Similarly, a jump of 2 standard deviations above the mean is: $6.17 + 2(0.247) = 6.664$. Observe, then, that Thiam's jump of 6.58 meters is just below 2 standard deviations from the mean. Without calculating the z-score, then, we already know Thiam's jump will fall just short of 2 standard deviations.

To calculate Thiam's z-score for the long jump, simply plug in the numbers from the table into the z-score formula above, which yields:

$$\text{Thiam's } z\text{-score for the Long Jump} = \frac{6.58 - 6.17}{0.247} = 1.66.$$

Indeed, Thiam's z-score is shy of the 2 standard deviation mark. What about Johnson-Thompson's z-score for the 200-meter dash? Plugging in the data from the table, we have that:

$$\text{Johnson-Thompson's } z\text{-score for the 200-Meter Dash} = \frac{23.26 - 24.58}{0.654} = -2.02.$$

Now that their respective performances have been leveled to standard deviation units, we can determine which performance was more exceptional. Johnson-Thompson's performance was a little less than 2 standard deviations below the mean time of the 200-meter dash; a more impressive performance than Thiam's long jump that was 1.66 standard deviations above the mean. Even though Thiam came in first in the long jump, Johnson-Thompson's 200-meter dash performance was more impressive.[2]

The t-Statistic and the t-Distribution

You may recall from STA120 the z-statistic, also known as the *standardized statistic*. You should see clear parallels in how we constructed z-scores to the z-statistic. The generic formula for the z-statistic is:

$$\text{Standardized Statistic} = z = \frac{\text{Sample Statistic} - \text{Mean of Null Distribution}}{\text{Standard Deviation of Null Distribution}}$$

The idea is that, if the sampling distribution of a statistic is approximately normal, then

The t-statistic for a one sample t-test is defined as:

$$t_{n-1} = \frac{\bar{x} - \mu_0}{s_{\bar{x}}},$$

where $\bar{x}$ is our sample mean, μ_0 is the hypothesized population mean (here 10 seconds), and $s_{\bar{x}}$ is the *estimated* standard error of the sample mean.

This estimated standard error is defined as:

$$s_{\bar{x}} = \frac{s}{\sqrt{n}}, \text{ where } s = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1},$$

[2] We calculated 2 z-scores above, but they came from different distributions: a distribution of long jump measurements and a distribution of 200-meter dash times). It's more common to convert all values of a distribution into z-scores and therefore transform the distribution from values measured in their original units (e.g., meters) into standard-deviation units. The resulting distribution is called a z-distribution.

where the s's denominator, $n-1$, is the number of degrees of freedom.

Sampling Distributions with a Continuous Variable

Because we're dealing with a continuous variable the sampling distribution is a continuous distribution. Recall that a sampling distribution is a distribution of a statistic, in the present case a sample mean. Time is a continuous variable and if we drew lots of samples of times and took the mean of each, after an infinite number of draws we'd have a continuous distribution of sample means. This is different from the Buzz and Doris scenario where the variable was nominal (success/failure). No matter how many would-be samples we drew, we'd arrive at a number between 0 and 16 with every sample drawn, and hence there'd always be gaps between the possible statistics in the sampling distribution. We saw those gaps between 0, 1, and so on when we plotted the binomial distribution earlier.

The continuous sampling distribution when the experiment involves a single continuous variable is called the t-distribution. You may recall that the t-distribution is similar to the normal distribution, but its tails are thicker than the normal distribution, meaning there's greater uncertainty for values far from the mean than the normal distribution.

Simulating a t-Distribution

As before, let's simulate the sampling distribution. First, let's a set a seed number so that we all obtain the same statistics:

```
# Set random number seed for reproducible results
set.seed(18934)
```

Since we're simulating a t-distribution use the `rt() function`, where, similar to `rbinom()`, the "r" refers to random and the "t" stands for t-distribution. The `rt()` function requires 2 arguments: the number of samples to be drawn, `n =`, and the number of degrees of freedom, `df =`. The concept of degrees of freedom is a complicated one and not worthy of our attention here. In the present case, the number of degrees of freedom is the number of independent pieces of information in our dataset after accounting for

```
# Simulate t-distribtuion; draw samples 1000; df = 16
t_sim <- rt(n = 1000, df = 47)
```

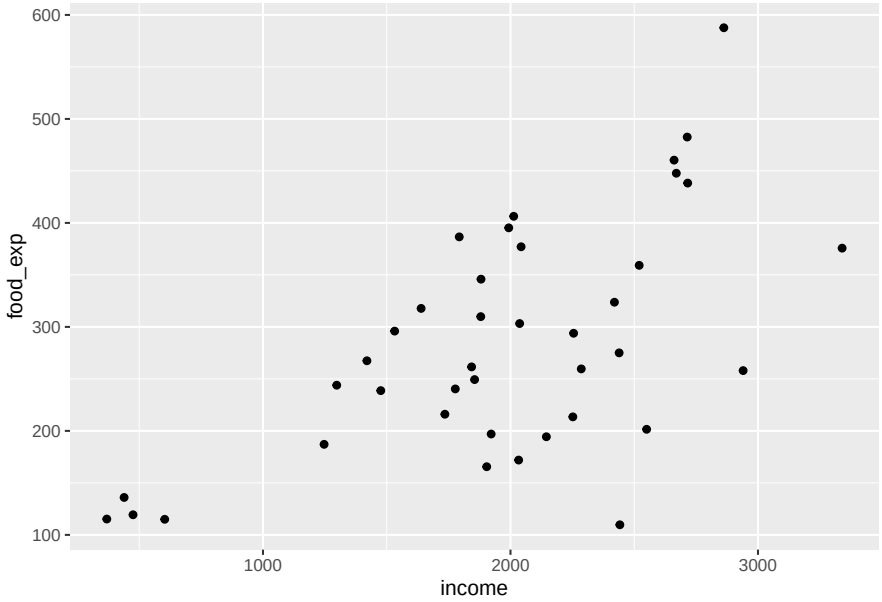
Simple Linear Regression (Chapter 10)

The first dataset we'll use to review linear regression consists of a sample of 40 households that provides each households monthly income and monthly expenditure on food. It's a .csv file located on my github page and can be imported into R by running:

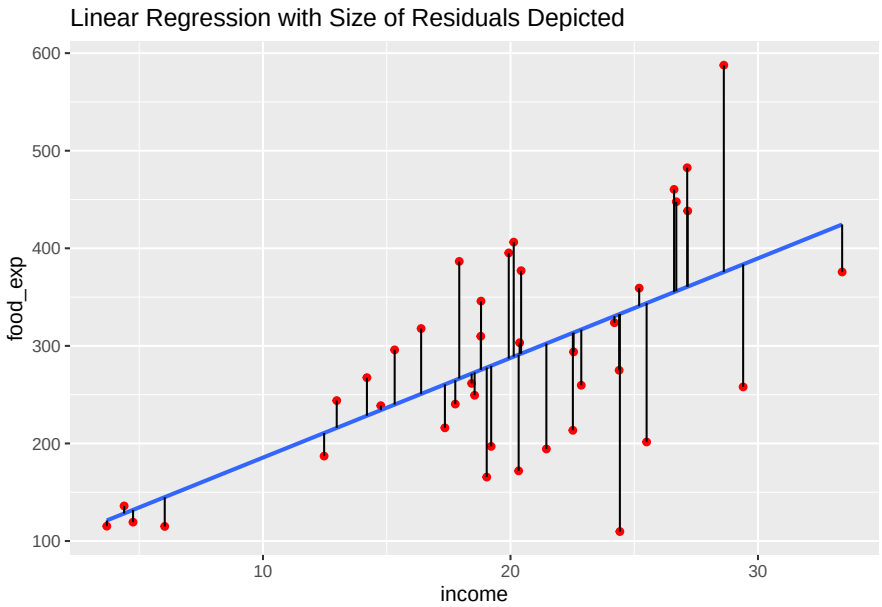
```
# Import data from Github; call dataframe food_df
food_df <- read_csv("https://bit.ly/2UVB1Eg")
```

The scatterplot below depicts the incomes of a sample of households and each households expenditure on food.

```
ggplot(data = food_df,
       mapping = aes(income, food_exp)) +
  geom_point()
```



Suppose we wanted to draw a line through the points. Which line would best represent the average relationship between household income and food expenditure? Many lines could be drawn through the data, but the best one (at least at this introductory level) is based on the **least squares principle**. The least squares principle states that the best fitting line is the line that minimizes the sum of squares of the vertical distances from each point to the line.[3]



The vertical distances shown in the plot are called *residuals*. Squaring each residual and then summing them yields a positive number. That positive number is the *smallest* number that you could find if you repeated the same calculation for any other line.

But why is a line with the smallest sum of squared residuals the *best* line to fit the data? Think of an (extreme) example: All data points are situated on the regression line. That line would then be a perfect fit of the data, and note that, since all points are on the line, each residual equals 0. The sum of the squared residuals is therefore zero, which is the *lowest* number a square can be.

[3]Why take the vertical distances instead of the horizontal distances? The simple linear regression model assumes the values of independent variable are constant (non-stochastic) and thus examines the variation in the dependent variable.

No real-world dataset would appear linear, although it makes sense that the closer the points are to the line the better the fit of the line. In other words, the smaller the residuals, the better the fit. But then, why square the residuals after summing them up? Residuals are squared to prevent large positive residuals from canceling out large negative residuals. If we didn't square the residuals, especially the large positive and negative ones, then they could cancel one another out, and the sum of residuals would be a small number that would make us (misleadingly) believe the line was a good fit.[4] By squaring we also penalize the large residuals, positive and negative, more than the small residuals.[5]

Part I: Running a Linear Regression

Now that we have some intuition about the least squares principle and why R plots a regression line where it does, let's run a regression of these data.

We've plotted many regression lines and thus obtained some general information about the data and an underlying statistical model. But we need more specific information. For example, a plot reveals whether the slope is positive or negative, but what's the magnitude of the slope? To answer such a question you need to run a linear regression. Running a linear regression requires the linear model function, `lm()`, which uses the following general syntax:

```
generic_fit <- lm(y_variable ~ x_variable, data = name_of_dataframe)
```

The first argument of the `lm()` function is the model's *formula*, which specifies the dependent variable and the independent variable(s). That `~` symbol separating the two variables can be interpreted "is a function of."

In our case we need to execute the following code:

```
# Run regression of food_exp on income; call regression food_fit
food_fit <- lm(food_exp ~ income, data = food_df)
```

You'll always want to assign a name to every model because your investigative work is just beginning after you run the regression. For example, we want at the very least to know the values of the estimated parameters and look at summary measures of how well the model fits the data. The `summary()` function returns basic results in the console:

```
# View summary of food_fit
summary(food_fit)
```

From the output returned in the Console, you see various numbers. We'll concentrate on three: The intercept and slope estimates and R^2.

Now consider the estimates for the intercept and the slope. These two estimates completely determine the regression line. That is, the regression equation is written as:

$$\widehat{food} = 83.42 + 0.1 \times income$$

where $\widehat{food}$ is the predicted value of food expenditure.

When viewing regression results the first thing (and most important thing to learn to do well) is to interpret the meaning of your estimated parameters. So let's be clear on how you need to interpret this equation:

1. The intercept: If household income is \$0, then households, *on average*, spend \$83.42 per month on food.
2. The slope: If household income increases by \$1, households, *on average*, spend 10 cents on food.

[4] The statistics we're using, the so-called frequentist statistics, make a critical distinction between estimators and estimates. Simply put, estimators generate estimates from the data. An *estimator* is a formula used to calculate a statistic. When we use an estimator's formula to arrive at a specific value we obtain an *estimate*. In general, estimators are forumulas whereas as estimates are numbers.

[5] Think of the graph of $f(x) = x^2$ compared to $g(x) = x$. Letting x measure the residual distance, then for $x < 1$ and $x > 1$ the vertical distance between the two graphs gets larger and larger as the size of residuals (in absolute value) grows.

The phrase "on average" has been emphasized because the regression line represents an average relationship between household income and food expenditure. The relationship is not a deterministic one, meaning we can't say with absolute confidence that if household income \$1,500, then food expenditure must be $83.42 + 0.1 \times 1,500$, or \$233.42. The latter value is the *predicted* value of food expenditure when income is \$1,500; it is *not* the actual value of expenditure when income is \$1,500.

Let's also examine the 95% confidence intervals for the intercept and slope estimates:

```
# Generate 95% confidence intervals for coefficient estimates
confint(food_fit)
```

```
##                   2.5 %      97.5 %
## (Intercept) -4.46327888 171.2952829
## income       0.05972052   0.1444723
```

A regression yields *predictions* and *residuals*. The predictions tell you the pattern the model captured; the residuals tell you what the model missed. Let's add the predictions and residuals from `food_fit` to the `food_df` by using a most-handy tool called "the pipe," denoted by `%>%`. We'll cover the pipe in much greater detail later, but, to simplify, the pipe allows you to link a sequence of ordered commands. The start of the sequence starts with a data frame. In our case, we'll want to save this data frame, `food_df`, with the residuals and predictions from the model `food_fit`. This means, the sequence will start `food_df <- food_df`. Then we'll insert the `%>%` in front of two new functions, `add_residuals()` and `add_predictions()`. These functions come from the **modelr** package. The only argument you pass to these functions is the model, `food_fit`. In this one block of code you'll add two new columns to `food_df`, called `resid` and `pred`:

```
# Load modelr package
library(modelr)

# Add predictions and residuals to food_df using the pipe (%>%)
food_df <- food_df %>%
  add_residuals(food_fit) %>%
  add_predictions(food_fit)
```

Confidence intervals offer a quick and simple idea of the range of possible values given your dataset. But you probably shouldn't put too much weight into them. I treat them as just the beginning of the investigative work. What confidence intervals can't do for us is give us an idea about how robust these estimates are. To gain a better understanding of the accuracy of these coefficient estimates, let's simulate some food expenditure data using the estimates and random noise. First, to simplify matters, let's rename the coefficients and the standard error of the regression.

```
# Extract and rename intercept estimate
a <- coef(food_fit)[[1]]

# Extract and rename slope estimate
b <- coef(food_fit)[[2]]

# Rename income
x <- food_df$income

# Extract and rename standard error of regression
sigma <- sigma(food_fit)
```

```
# Set seed for reproducible results
set.seed(1111)

# Simulate food_exp data with estimates and some random error
food_exp_sim <- a + b*x + rnorm(n = length(x), mean = 0, sd = sigma)
```

To run the regression we'll need to form a data frame containing both the y-variable (`food_exp_sim`) and x-variable (x):

```
# Create df food_sim with simulated food_exp and income (x)
food_sim <- data.frame(food_exp_sim, x)
```

Okay. Now we're ready to run the regression of the simulated expenditure data on income (i.e., `x`):

```
# Run regression with simulated data
food_fit_sim <- lm(food_exp_sim ~ x, data = food_sim)
```

You could causally compare the two sets of estimates to see how far they deviate from one another. But, let's take a more analytical approach.

```
# Extract and rename simulated slope estimate
b_sim <- coef(food_fit_sim)[[2]]

# Extract and rename simulated slope standard error
b_se_sim <- summary(food_fit_sim)$coefficients[2, 2]
```

I just calculated a particular predicated value of food expenditure, namely the predicated value if income is \$1,500. The function `fitted()` returns all predicted values corresponding to the given income levels in our data frame. We'll add a new column to the `food_df` data frame with the predicted values:

```
# Add predicted values to food df
food_df$predicted <- fitted(food_fit)
```

Finally, the *coefficient of determination*, or R^2, is the percentage of total sample variation in the dependent variable (here, `food_exp`) due to sample variation in the independent variable (here, income). See the Appendix for a detailed discussion and derivation of R^2.

NB: R^2 is a *descriptive statistic*. It's an artifact of the data and is not used (directly) for statistical inference. The point: Don't let its value guide your modeling decisions.

What's important for us is that R^2 provides us with a numerical description about the fit of the data. Exercising due caution, you could say the closer R^2 is to 1, the better the regression line fits the data. In fact, if $R^2 = 1$ all the data would be on the regression line. On the other hand, the closer R^2 is to 0, the worse the regression line fits the data. In this example $R^2 = 0.385$, which isn't too surprising after seeing how large many of the residuals were in the plot above.

Tidy Results

A lot of this course has to do with "tidiness": Tidy data, tidy code, and also tidy results. To extract results from a model, as you've just experienced, is pretty painful. This is essentially because the functions you used, e.g., `lm()` and `summary()`, are old functions and they return unwieldy objects that don't play well together. Fortunately, there's a better, more tidy, way that returns model results as a dataframe. The functions we'll need, `glance()` and `tidy()`, are part of the **broom** package, which you should install and load in to R.

Now that you've been introduced to the pipe operator, `%>%`, see what the following block of code returns:

```
# Load broom package into R
library(broom)

# Use glance() to return summary of model
food_fit %>% glance()

# Use tidy() to return a coefficient table
food_fit %>% tidy()
```

Though the benefits of `glance()` and `tidy()` are not so apparent when dealing with a simple linear regression, they are absolutely necessary when we start analyzing multiple models later in the course.

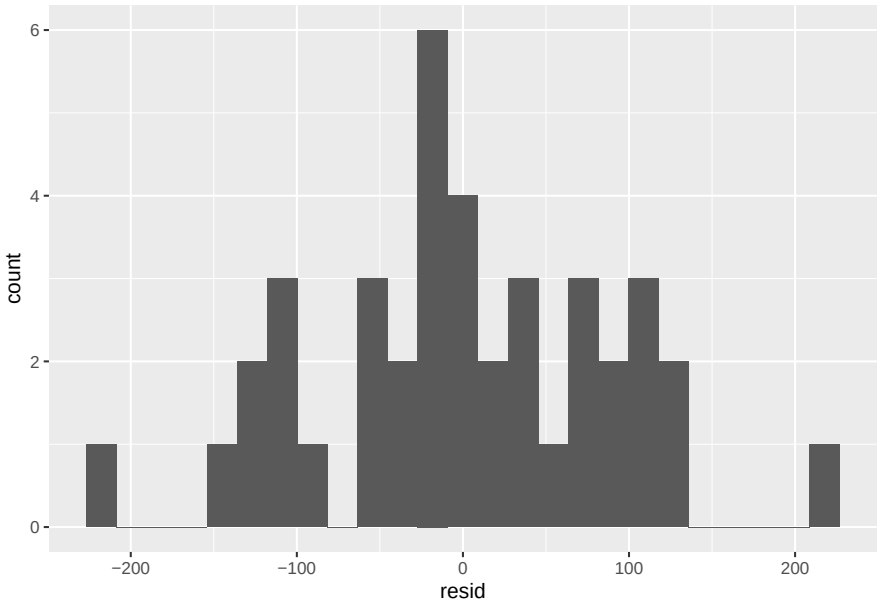
Investigating Residuals

You'll often want to analyze the residuals from your regression. To do so, run the `residuals()` function using the model name as its argument:

```
# Save residuals to food df
food_df$residuals <- residuals(food_fit)
```

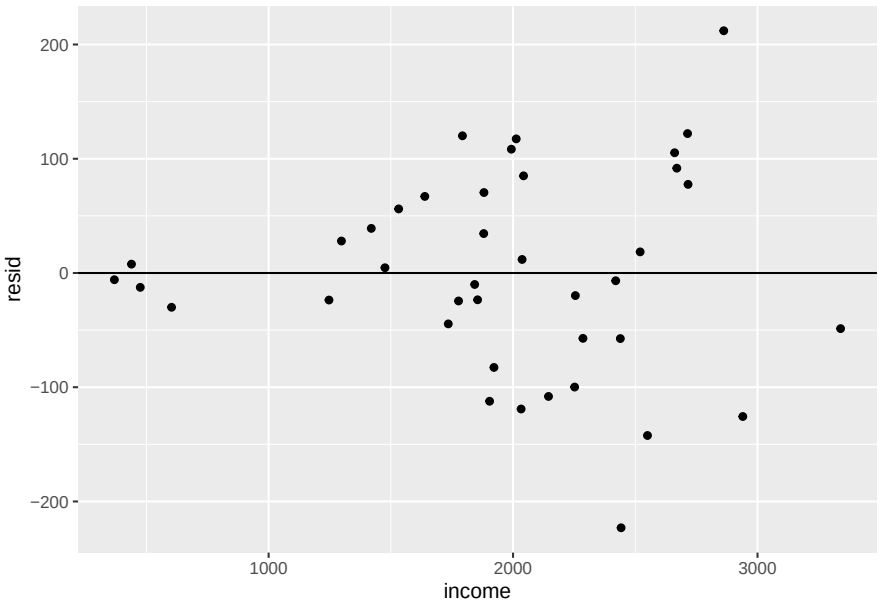
One way to investigate residuals is to plot a histogram:

```
# Plot histogram of residuals
ggplot(data = food_df,
       mapping = aes(x = resid)) +
  geom_histogram(bins = 25)
```



The histogram shows how residuals are distributed and it appears to be relatively symmetrical. But, the histogram doesn't tell us which values of income are paired with each residual. For example, we'd like to know how the residuals vary as income increases. To that end, create a **residual plot**, putting income on the x-axis and the residuals on the y-axis. Since you've already added the residuals column to the `food` data frame, you're ready to generate the residual plot.

```
# Plot histogram of residuals
ggplot(data = food_df,
       mapping = aes(x = income, y = resid)) +
  geom_point() +
  geom_hline(yintercept=0)
```



Notice that as income increases the (absolute) value of the residuals tend to increase.[6] This suggests that error term of the regression has a non-constant variance, a phenomenon know as *heteroskedasticity*. This is a problem that needs to be corrected, but in this introduction we just call attention to it.

Part II: Estimating the Marginal Propensity to Consume

```
# Import consumption-income data from github; call it cf_df
cf_df <- read_csv("https://bit.ly/3dwl5P9")
```

Creating New Variables

The marginal propensity to consume (*mpc*) is defined as the change in consumption spending given a $1 change in disposable income, or:

$$mpc = \frac{\Delta C}{\Delta Y}$$

The `cf_df` data frame consists of (real) consumption expenditures and (real) disposable income per year from 1959 to 2018. You need to first create new columns in `cf_df` with the differences of each variable to correspond with the definition of the *mpc*. Next, to aid in interpreting the data, you'll subtract the mean of the difference from each difference. This is called *mean-centering* a variable, which is explained in detail in the Appendix below. You'll use `mutate()` again to add new variables to `cf_df`. Follow these steps"

Step 1:

To difference a variable, use the `lag()` function, which requires you to specify which variable to lag and by how many periods. Then subtract it from the non-lagged vector. For each new variable add `_delta` to the end of its name to indicate that these new variables are differences.

```
# Create new differenced variables using mutate()
cf_df <- cf_df %>%
  mutate(c_delta = cons - lag(cons, 1),
         yd_delta = yd - lag(yd, 1))
```

You've now created ΔC_t and ΔY_t. The `cf_df` data frame should now have 5 variables after creating `c_delta` and `y_delta`. You should verify that `cf_df` has 5 columns and that the values in the new columns are the

[6] If you added a regression line to this plot it would be horizontal with a constant value of 0. Why? Because the magnitudes of the residuals cancel each other out rendering the average to be 0.

differences between a cell and the cell above it.

Finally, it's critical to notice that one observation is lost after differencing. That's why the values in the first row of `c_delta` and `y_delta` are both `NA`.

Step 2:

Now, mean center each variable by subtracting the mean of the differences from each value of the difference.

```
# Mean center the differences; ignore the observation lost by differencing
cf_df <- cf_df %>%
  mutate(c_delta_mean = c_delta - mean(c_delta, na.rm = TRUE),
         yd_delta_mean = yd_delta - mean(yd_delta, na.rm = TRUE))
```

Because we lost one observation by differencing each time series we had to add the argument `na.rm = TRUE` inside `mean()`. The extra argument, `na.rm = TRUE`, instructs R to remove any cells with `NA` values when taking the mean of a column. If we didn't include `na.rm = TRUE`, then R would yell at us because it can't take the mean of a vector containing a non-numerical values like `NA`.

Verify that data frame `cf` now has 7 variables before moving on.

Digression: Mean Centering Variables

The last command mean-centered both the dependent and independent variable. It's often a good idea to mean center the independent variable in order to make the intercept estimate meaningful. Consider this example from the `wages_df` dataset.

```
# Import wage data from github; call it wages_df
wages_df <- read_csv("https://bit.ly/3AgeBO4")
```

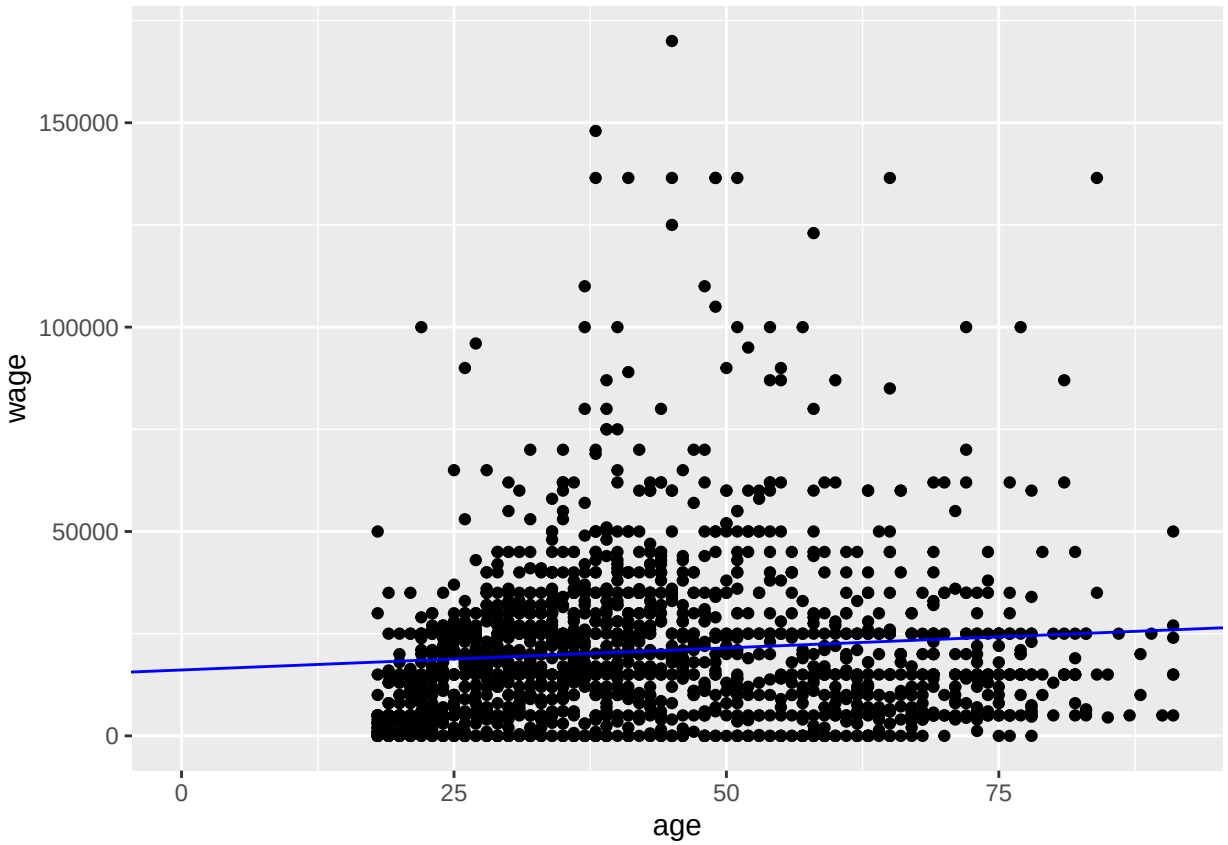
You'll use this dataset in the final section of this module to examine differences in pay between genders. Here, though, let's examine the relationship between age and wages by running a regression and viewing the results:

```
summary(lm(wage ~ age, data = wages_df))
```

As you would expect, as workers get older the more they are paid. But, what does an estimated intercept of about the intercept of 16117.50 mean? Is it meaningful?

Plotting the data and the estimated regression line clearly shows we're extrapolating well beyond the data:

```
ggplot(data = wages_df, aes(x = age, y = wage)) +
  geom_point() +
  geom_abline(intercept = 16117.5, slope = 108, color =
                'blue') +
  expand_limits(x = 0)
```



In order to obtain a meaningful intercept estimate, you should mean-center the independent variable, `age`. When you mean center the independent variable the intercept is no longer the estimated value of the dependent variable when the independent variable is 0; instead, the intercept is the estimated value of the dependent variable when the independent variable is at its mean. In this example, after mean centering `age`, the intercept estimate is the expected wage for average-aged workers.

Plotting the Data

Using **ggplot2** make one scatterplot with the following attributes:

1. Put `c_delta_mean` on the vertical axis and `yd_delta_mean` on the horizontal axis.
2. Add a regression line to the plot without confidence intervals.
3. Make an approriate title

Running a Regression (Again)

Plotting the regression line offers general information about the model. But we need more specific information. For example, a plot reveals whether the slope is positive or negative, but what's the actual slope of the regression line? To answer such a question you need to run the linear regression model. Running a linear regression requires the linear model function, `lm()`, which uses the following general syntax:

```
generic_fit <- lm(y_variable ~ x_variable, data = name_of_dataframe)
```

The first argument of the `lm()` function is the model's *formula*, which specifies the dependent variable and the independent variable(s). The `~` symbol separating the two can be interpreted here as "is a function of".

In our case, we need to execute the following code:

```
cf_fit <- lm(c_delta_mean ~ yd_delta_mean, data = cf_df)
```

You'll always want to assign a name to every model because your investigative work begins after running. For example, we want at the very least to know the values of the estimated parameters and look at summary measures of how well the model fits the data. Use `tidy()` and `glance()` from the **broom** package as before:

```
# Load broom package into R
library(broom)

# Use glance() to return summary of model
cf_fit %>% glance()

# Use tidy() to return a coefficient table
food_fit %>% tidy()
```

First, note that the estimate for `yd_delta_mean` is 0.68 after rounding up. This means the regression line we already plotted takes the following form:

$$(\Delta C_t - \overline{\Delta C}) = 0.68(\Delta Y_t - \overline{\Delta Y})$$

where the bar over a variable designates the mean of the variable.

How are we supposed to interpret the estimated *mpc*? An estimate of 0.68 means that, on average, an increase in disposable income of $1 billion above normal is associated with an increase in consumption of $0.68 billion above normal.

Least Squares Estimation

```
# Add residuals of model to cf_df
cf_df <- cf_df %>% add_residuals(cf_fit)
```

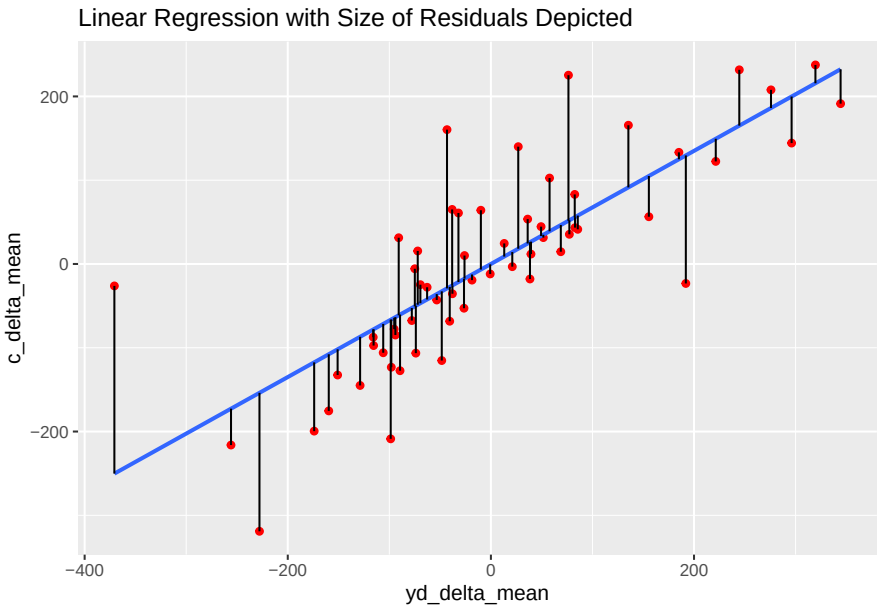
```
# Add predicted values to cf_df
cf_df <- cf_df %>% add_predictions(cf_fit)
```

```
ggplot(cf_df,
       mapping = aes(yd_delta_mean, c_delta_mean)) +
  geom_point(color = "red") +
  geom_smooth(method = "lm", se = FALSE) +
  geom_segment(aes(x = yd_delta_mean, y = c_delta_mean,
                   xend = yd_delta_mean, yend = pred)) +
  labs(title = "Linear Regression with Size of Residuals Depicted")
```

```
## Warning: Removed 1 rows containing non-finite values (stat_smooth).

## Warning: Removed 1 rows containing missing values (geom_point).

## Warning: Removed 1 rows containing missing values (geom_segment).
```



Goodness of Fit Measures

How well did the linear regression model fit the data? The fit of the model can be summarized by two measures: the estimated variance of the error, $\hat{\sigma}^2$ and by an estimator called *coefficient of determination*, R^2.[7]

Why does $\hat{\sigma}^2$ provide a measure of model fit? Think about the regression line in the scatterplot and specifically the vertical distances of the data from the line that represent the error. The smaller the variability of the the errors (the vertical distances), the less spread out the deviations of the data from the regression line. The observed data are, on average, closer to the regression line when the variance of the disturbance is smaller.

The coefficient of determination, R^2, is a more widely used measure to evaluate how well your model fits the data, and appears when you run the summary() function of your model. R^2 is an estimator of the percentage of the total variation in the dependent variable due to variation in the independent variable(s).

Appendix

Mean-Centering Variables

In estimating the *mpc* you centered both variables around their respective means. That is, you subtracted the mean of the variable from each of its values. This was unnecessary insofar as the estimation of the *mpc* is concerned; not centering the variables generates the same estimate for the *mpc* (and its standard error). To see why consider this general regression equation with the independent variable mean-centered:

$$\begin{aligned} y_i &= \beta_0 + \beta_1(x_i - \bar{x}) \\ &= \beta_0 + \beta_1 x_i - \beta_1 \bar{x} \\ &= (\beta_0 - \beta_1 \bar{x}) + \beta_1 x_i. \end{aligned}$$

Note that the coefficient for x_i remains unchanged; it's still β_1. But what does $(\beta_0 - \beta_1 \bar{x})$ mean? The interpretation of the intercept is now the mean of y_i when the x_i equals its mean value.

[7] Squaring isn't the only way to limit the degree to which residuals cancel each other out. We could also take absolute of the residuals. But in more advanced applications, working with sqaures is more mathematically convenient than working with absolute values.

Recall that a regression line is characterized by the mean value of y conditional on the independent variable(s). As a result, the intercept, as one point on the regression line, is to be interpreted as a mean value of y. If the dependent variable is centered around its mean the intercept term must be zero. Why? Because the intercept term then becomes the mean of a variable whose centered around its mean. How so? First, it's useful to know that the sum of deviations of any variable from its mean is 0. That is:

$$\sum_{i=1}^{n}(a_i - \bar{a}) = 0,$$

for any vector $(a_1, \ldots, a_n)$. See the following footnote for the proof.[8] So we want to know the mean of $(y_i - \bar{y})$, which is:

$$\frac{1}{n}\sum_{i=1}^{n}(y_i - \bar{y}) = 0.$$

since $\sum_{i=1}^{n}(y_i - \bar{y}) = 0$ by the above result.

It's common to mean-center the independent variable(s), while leaving y_i as is in order to get an interpretable intercept term.[9] But, as just shown, mean-centering the dependent variable, irrespective of how x_i is transformed, forces the intercept term to be zero.

The reason behind mean-centering both variables when estimating the *mpc* was to enhance the interpretability of the data and regression results. By centering variables around a number we move their relative location on the axis without changing the distances between any two values. In our case, we centered variables around their respective means, so that if any value equals the mean the centered value is zero. By locating values around zero we can interpret their magnitudes as above or below "normal," where normal is the mean of the variable.

R^2 Explained Graphically and Analytically

Graphical Explanation

The coefficient of determination, R^2, can be thought of as a comparison of the total variation of the data around the mean value of the outcome variable, y, against the total variation of the data around the regression line. It can be written informally as:

$$R^2 = \frac{\text{Var}(Mean) - \text{Var}(Line)}{\text{Var}(Mean)}.$$

It's worth pointing out that the denominator only serves to normalize the value of R^2 so that it ranges between 0 and 1; the real action is in the numerator, the comparision between variability around the mean and variability around the regression line.[10]

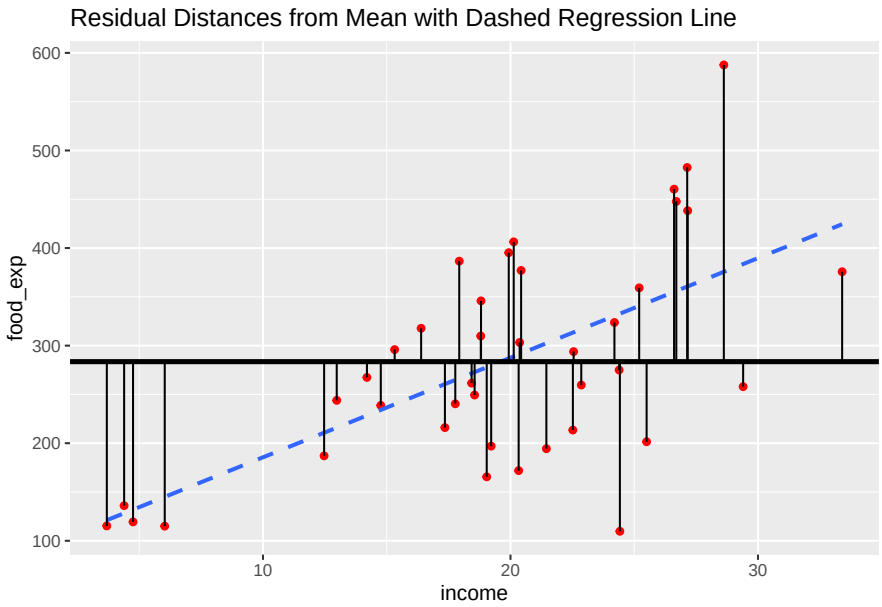
[8] Specifically:

$$\sum_{i=1}^{n}(a_i - \bar{a}) = \sum_{i=1}^{n} a_i - n\bar{a} = \sum_{i=1}^{n} a_i - \sum_{i=1}^{n} a_i = 0,$$

since a sum of a n constants (e.g., $\bar{a}$) is n times the constant.

[9] In many instances, particularly in economics, the mean value of y_i when $x_i = 0$ is meaningless because many economic variables are non-negative and typically positive.

[10] The reason why division by $Var(Mean)$ bounds R^2 within the interval $[0, 1]$ is because $0 \leq Var(Line) \leq Var(Mean)$. In words, variances are always non-negative and there will (almost) always be more variability around the mean than a regression line.



Note that if the mean of the outcome variable were as good or better at predicting the outcome than the dependent variable, then the dependent variable would contribute nothing to explaining the variation of y. That is, the slope of the mean would be 0, implying there's no relationship between the two variables. Under these circumstances, R^2 would also be zero (or lower).

But the regression line will (almost) always be a better predictor than $\bar{y}$ in the sense that the distance between the data the regression line is smaller than the distance between the data and the mean of y. The vertical distances between the data and the mean of y is shown in the plot along with a dashed regression line. By inspection, it's clear that, although some points are closer to the mean line than the regression line, overall the data is located closer to the regression line.

As an example, suppose you run a regression and, after running the `summary()` command, you see R^2 is 0.65. How do you interpret $R^2 = 0.65$? What this means is that there is 65% *less* variation around the regression line than around the mean. (Remember less variation the closer the fit.) Alternatively, the relationship between income and food expenditure accounts for 65% of the total variation.

The following subsection provides a mathematically intensive explanation of R^2. It's important to link this graphical explanation to the one below. Specifically, note that we can offer a more precise specification of R^2 than the one given above, namely:

$$R^2 = \frac{Var(Mean) - Var(Line)}{Var(Mean)} = \frac{\sum_{i=1}^{n}(y_i - \bar{y})^2 - \sum_{i=1}^{n}(y_i - \hat{y})^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

The sum $\sum_{i=1}^{n}(y_i - \bar{y})^2$, what I've called the "variance of the mean," is called the *total sum of squares*, typically abbreviated as SST. On the other hand, the "variance of the line," is given by $\sum_{i=1}^{n}(y_i - \hat{y})^2$ and is called the *sum of squared residuals*, denoted by SSR. Simple algebra shows that R^2, as explained here, is written as $(SST - SSR)/SST$ and is equivalent to the alternative expression given below.

Analytical Explanation

The total variation of Y from its (sample) mean is given by taking its *squared* variation; or $\sum(y_i - \bar{y})^2$. Call this sum SST for total sum of squares. A question naturally arises about this sum of squares: Why do we square the sum instead of just taking the sum? Suppose that we did just sum up the difference of each y_i and its mean, $\bar{y}$. Then the positive differences and negative differences cancel out each other out, leaving us with an answer of 0. That is, $\sum_{i=1}^{n}(y_i - \bar{y}) = 0$ for *any* set of numbers $\{y_1, \ldots, y_n\}$. You can verify this in R with the following code:

```r
# Create a vector of numbers called x
x <- c(1, 4, -5, 3, 5, -9, -2, 0)

# Subtract each number from the mean of x and then sum
sum(x - mean(x))
```

Put any numbers you want into the vector x and R will return 0. Further, if you're told that the divergence between some set of numbers and its mean is 0, then you'd rightly conclude that there is no divergence and that the numbers must all be equal. That's obviously not helpful in our case. So, by squaring each difference, the positive differences stay positive and negative differences become positive, yielding a sum of squares that is positive. That's helpful.

A convenient and, perhaps, intuitive fact about the SST is that it can be partitioned into two additive terms. The first term consists of variation in y from its mean due to variation in the independent variable(s) from its mean(s). This element is the *explained* sum of square variation from the mean, which we denote by SSE. The second term consists of the variation in y from its mean not attributable to variation in the independent variables. This term is the *unexplained* sum of squared variation from the mean, denoted SSR. So we can therefore write:

$$SST = SSE + SSR.$$

Our original goal was to gain an understanding of how much the variation of the independent variable explained the dependent variable, that is, R^2. The decomposition of SST now gives us the answer:

$$R^2 = \frac{SSE}{SST} = \frac{\sum(\hat{y}_i - \bar{y})^2}{\sum(y_i - \bar{y})^2},$$

where $\hat{y}_i$ is the predicated value of y as given by the regression. (Note that $SSE = SST - SSR$; the numerator from the graphical explanation of R^2.)

A Detailed and Advanced Explanation of the Coefficient of Determination

In a simple linear regression the sample variation of the dependent variable y, given by its total sum of squares (SST), can be decomposed into the estimated effect of the independent variable x on the variation of y (SSR) and the estimated effect of the disturbance on the variation of y, given by the sum of squared errors (SSE); that is,

$$\sum_{i=1}^{n}(y_i - \bar{y})^2 = \sum_{i=1}^{n}(\hat{y}_i - \bar{y})^2 + \sum_{i=1}^{n} e_i^2.$$

The SSR can be further developed as follows:

$$\begin{aligned}
SSR = \sum_{i=1}^{n}(\hat{y}_i - \bar{y})^2 &= \sum_{i=1}^{n}(\hat{\beta}_0 - \hat{\beta}_1 x_i - \hat{y})^2 \\
&= \sum_{i=1}^{n}((\hat{\beta}_0 - \hat{\beta}_1 x_i) - \hat{y})^2 \\
&= \sum_{i=1}^{n}((\hat{y} - \hat{\beta}_1 \bar{x}) - \hat{\beta}_1 \bar{x} - \hat{y})^2 \\
&= \sum_{i=1}^{n}(\hat{\beta}_1 (x_i - \bar{x})^2 \\
&= \hat{\beta}^2 \sum_{i=1}^{n}((x_i - \bar{x})^2.
\end{aligned}$$

Also, the SSE can be written as:

$$\sum_{i=1}^{N} e_i^2 = \sum_{i=1}^{N} (y_i - \hat{y}_i)^2.$$

Starting with the definition of the total sum of squares, we can derive the coefficient of determination. R^2 is the ratio of the explained variation compared to the total variation; thus, it is interpreted as the fraction of the sample variation in y that is explained by x. Dividing the definition SST by itself we get:

$$1 = \frac{SSE}{SST} + \frac{SSR}{SST}.$$

And R^2 is given by the first term on the right-hand side, or:

$$R^2 = \frac{SSE}{SST} = 1 - \frac{SSR}{SST}.$$

Question: How much of the total variation in y is explained by the variation in x?

Answer: First, determine the total variation in y. Variation is understood to be a numerical measure of the distance of y from its central tendency, here, measured by the mean $\bar{y}$. That is:

$$\sum_{i=1}^{n} (y_i - \bar{y})^2 = SST$$

Now, let's reverse the original question: How much of the total variation in y is *not* explained by x? The variation in y not explained by x must result from all other factors influencing y excluding x's effect. In other words, the variation due to random disturbance. This variation is written as:

$$\sum_{i=1}^{n} (\hat{y}_i - \bar{y})^2 = SSR.$$

Now, consider their ratio:

$$\frac{SSR}{SST}$$

This fraction tells us the percentage of the total variation that is *not* described by variation in x. Subtracting this from 1 yields the answer to our original question. The total variation in y explained by variation in x is:

$$1 - \frac{SSR}{SST}$$

Assumptions of the Simple Linear Model

The following set of assumptions constitute the simple linear regression model.

1. **Linear in Parameters:** The model in the population can be written as:

   $$y_i = \beta_0 + \beta_1 x_i + \epsilon_i$$

   where β_0 and β_1 are the unknown parameters (constants) and ϵ is an unobserved random error.

2. **Zero Mean:** The error ϵ has an expected value of zero, that is:

   $$\mathbb{E}(\epsilon_i) = 0 \iff \mathbb{E}(y_i) = \beta_0 + \beta_1 x_i$$

3. **Homoskedasticity:** The error ϵ has the same variance given any value of the independent variable x_1. That is:

   $$\text{var}(\epsilon_i) = \sigma^2.$$

4. **Zero Covariance:** The covariance of ϵ_i and ϵ_j is zero for all i and j, that is:

$$\text{cov}(\epsilon_i \epsilon_j) = 0.$$

5. **Non-Stochastic Independent Variable:** The independent variable x_1 is not random and must take at least two different values.

6. **Normality of the Error Term:** The error term ϵ is distributed normally, which, by Assumptions 2 and 3 implies:

Comparing Two Means (Chapter 6)

Chapter 6 from the STA120 textbook concerns the general question of whether there's a significant difference between 2 groups, the difference being measured by the means of the 2 groups. In this review, we'll examine the same question (and extensions to it), but from a regression perspective. So this material builds upon the previous section on simple linear regression. Import these data with the following command:

```
# Import wage data from github; call it wages_df
wages_df <- read_csv("https://bit.ly/3AgeBO4")
```

This data frame, `wages_df`, contains data on workers, including information on salaries, race and gender. To get an idea of the data frame, use the **tidyverse** function `glimpse()` to see its variables and the corresponding data types.

```
# Glimpse wages_df data frame
glimpse(wages_df)
```

```
## Rows: 1,813
## Columns: 13
## $ wage             <dbl> 50000, 60000, 30000, 25000, 50000, 62000, 51000, 9000~
## $ wagek            <dbl> 50.00, 60.00, 30.00, 25.00, 50.00, 62.00, 51.00, 9.00~
## $ ethnicity        <chr> "White", "White", "White", "White", "Other", "Black",~
## $ educ             <dbl> 16, 16, 16, 17, 16, 18, 17, 15, 12, 17, 15, NA, 12, 1~
## $ mother_education <dbl> 16, 16, 16, 17, 16, 18, 17, 15, 12, 17, 15, 99, 12, 1~
## $ father_education <dbl> 16, 16, 16, NA, 16, 18, 17, 15, 12, 17, 15, 99, 12, N~
## $ walk             <dbl> 3, 6, 8, 8, 5, 1, 3, 7, 2, 7, 8, 1, 1, 4, 7, 7, 6, 4,~
## $ exercise         <dbl> 3, 5, 1, 1, 6, 1, 1, 4, 2, 1, 1, 1, 2, 1, 4, 4, 5, 6,~
## $ smokenow         <dbl> 2, 1, 2, 2, 2, 2, 2, 1, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2,~
## $ tense            <dbl> 0, 0, 1, 0, 0, 2, 4, 4, 0, 0, 0, 0, 1, 0, 2, 0, 2, 0,~
## $ angry            <dbl> 0, 0, 1, 0, 0, 2, 4, 4, 0, 0, 0, 0, 1, 0, 2, 0, 2, 0,~
## $ age              <dbl> 45, 58, 29, 57, 91, 54, 39, 26, 49, 46, 21, 53, 26, 6~
## $ gender           <chr> "Male", "Female", "Female", "Female", "Female", "Fema~
```

Notice that the gender vector indicates whether an employee is male or female. It's what's called a "character" vector as you'll learn in more depth later. (That's what the `<chr>` before the values refers to.) But, for present purposes, what's important is that it's a categorical variable and not a continuous variable. The vector wage, on the other hand, is a continuous variable; the `<dbl>` you see before the numbers stands for "double," which means the entries in the vectors are real numbers. If our research question is to examine whether male employees make more than their female counterparts, then the analysis will therefore involve a categorical variable, gender, and a continuous variable, wage.

Before investigating this research question, let's do some exploratory data analysis. To begin, look at the 5-point summary of the wage vector and create a histogram to get an idea of its distribution:

```
# Get 5-point summary of wage vector
summary(wages_df$wage)
```

```
# Create a histogram of wage vector
ggplot(data = wages_df, mapping = aes(x = wage)) +
  geom_histogram(fill = "lightblue", color = "grey30")
```

The 5-point (really 6-point) summary indicates the distribution is skewed right, as most distributions of income are, since the mean wage exceeds the median wage. The histogram's long right tail confirms this.

We want to examine the relationship between `wages` and `gender`. The `gender` vector, however, is a character vector, gender being a categorical variable; its entries are the categories "Male" and "Female". To further explore the dataset, but now with our attention fixed on the gender pay gap, let's create separate and side-by-side histograms for wages according to gender.

We'll construct a similar histogram of wages, but we'll introduce a very useful `ggplot()` option called *faceting*, which allows you to generate a single plot consisting of multiple subplots. To create faceted plots you need a categorical variable that tells R how to separate the subplots. In our case, we want two histograms of wages, the basis on which to divide histograms being gender with its female and male categories. The additional function you tag on to the previous code is called `facet_grid()`.[11]

The main argument inside `facet_grid()` is a formula that describes the variable to use for subsetting the data. This formula will be like the main argument inside `lm()` that specifies the relationship between variables and separated a `~`. The more complicated faceted plots will include two variables, but we'll only have use for one categorical variable. One side of the formula will therefore be empty, which you represent by a period, `.`.

Plots are arranged horizontally or vertically depending on how you specify the formula inside `facet_grid()`. If the subsetting variable, `gender` in this example, is to the left of `~`, then the plots are arranged vertically. If gender is on the right, `.` on the left of `~`, then the plots are placed horizontally (side-by-side), as done in the code below. After generating this horizontally-arranged plot, flip the variables in the formula, that is, `gender ~ .`, to view vertically-arranged histograms.

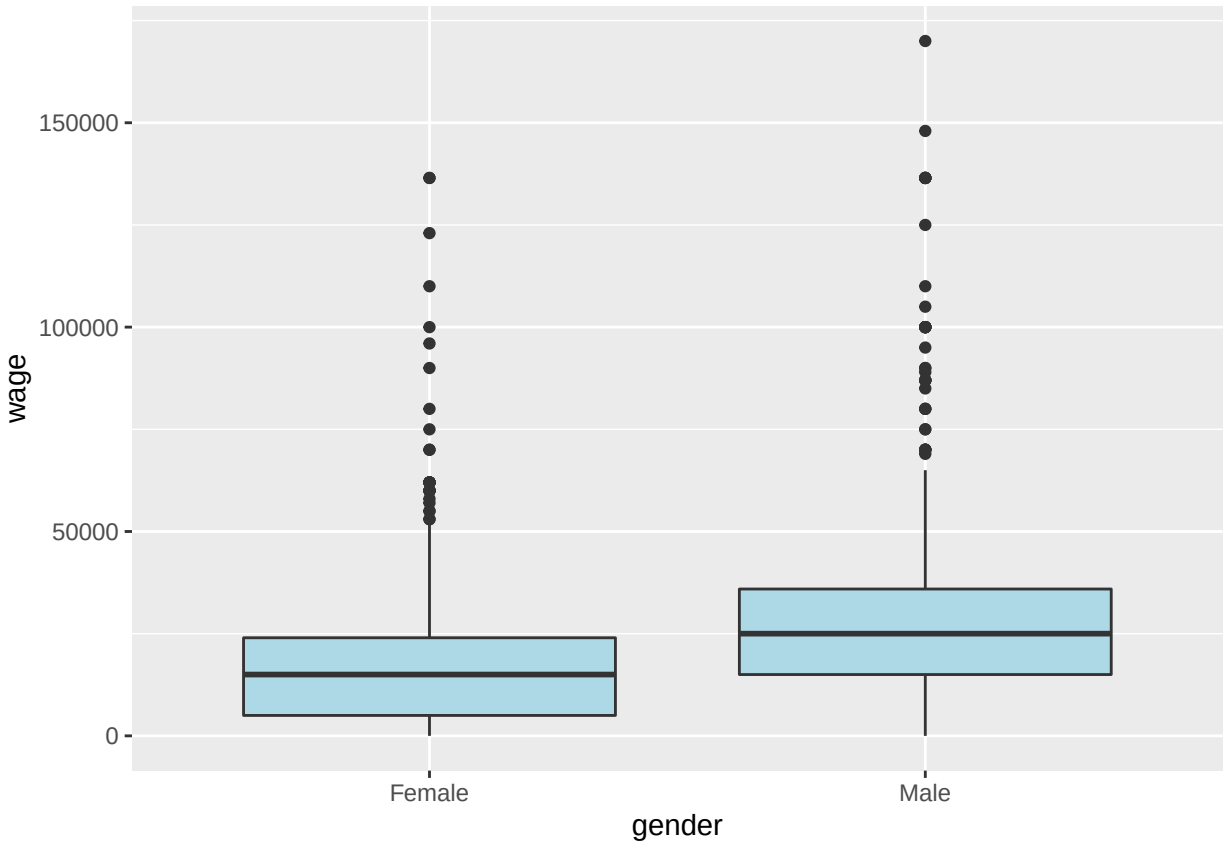
```
# Create a side-by-side histogram
ggplot(data = wages_df,
       mapping = aes(x = wage)) +
  geom_histogram(fill = "lightblue", color = "grey30") +
  facet_grid(. ~ gender)
```

As you can see, both histograms are right-skewed like the aggregate histogram for wages. But the distributional mass of female wages is highly concentrated at low wages, while male wages are more normally distributed.

It's natural when dealing with one categorical and one continuous variable to create a boxplot. The boxplot provides a visualization of the 5-point summary of wages, conditioning on gender.

```
# Create a boxplot of wages by gender
ggplot(data = wages_df,
       mapping = aes(x = gender, y = wage)) +
  geom_boxplot(fill = "lightblue")
```

[11] There is alternative way to arrange subplots instead of `facet_grid()`. What `facet_wrap()` does is to lay the subplots out horizontally and wrap around to a new row of subplots, like words on a page. See Chapter 11 of Chang's *R Graphics Cookbook* for a summary account of faceting with `ggplot()`.



As expected, median wages for males exceed female wages (the thick black lines inside the boxes). What we want to investigate, though, is whether there's a significant difference between *mean* wages of male and female workers. In STA120 this was accomplished by a 2-sample t-test, which is easily handled in R:

```
# Perform a 2 sample t-test
t.test(wage ~ gender, data = wages_df)
```

As the output shows, you can reject the null hypothesis that the mean wages are equal for females and males, which suggests that a gender wage gap does indeed exist. We look further into this question, but within a regression framework, in what follows.

Indicator Variables and the Gender Wage Gap

You are familiar with linear regression models, particularly those with *continuous* variables. These variables take the form of numbers that can be added, multiplied, and so on. Alternatively, basic arithmetic cannot be performed on a particular type of categorical variable called *nominal* variables. Examples of nominal variables include gender, states of the US and marriage status. The data constituting nominal variables are names (not numbers), and hence the name.

Because of the fundamental difference between data as numbers and data as names, how do we include qualitative information into a regression model? The most straightforward way is to create an ***indicator variable***, which allows us to quantify a qualitative difference among named categories. An indicator variable assumes two or more possible values, but we'll consider the most basic case in which the indicator variables takes on only two values. In this case, the variable equals 1 to indicate the presence of some characteristic or equals 0 to indicate the absence of that characteristic.[12] By way of example, let's consider gender. We'll write the indicator variable as $female_i$, which is defined as follows:

[12] An indicator variable that assumes only two values, like gender, is called a ***binaryvariable***.

$$female_i = \begin{cases} 0, & \text{Individual } i \text{ is male} \\ 1, & \text{Individual } i \text{ is female.} \end{cases}$$

In the `wages_df` data frame, gender is a still a character vector. Let's create a new variable, an indicator variable called `female`, in accordance with the above definition. To do so, use the `if_else()` function from the **tidyverse** package. The `if_else()` function has 3 arguments:

1. A test condition that is evaluates to either true or false for some vector.
2. The value given to the new vector if the condition is true.
3. The value given to the new vector if the condition is false.

So the "if-else" in `if_else()` makes sense: If an observation satisfies the test condition, then give it the value stipulated in the 2nd argument; else (i.e, if the observation fails the test condition) give it the value stipulated in the 3rd argument.

In our case, our test criterion will be `gender == "Female"`, which if true receives a value of 1, otherwise it receives a value of 0. That is, the if-else statement should read `if_else(gender == "Female", 1, 0)`. To add this new vector to the `wages_df` data frame, we'll use another **tidyverse** function called `mutate()`. This function's arguments are the data frame you want to add this new vector to and a definition of the new vector. Because you'll want it stored to `wages_df`, you'll need to start the command with `wages_df <-`. So, putting everything together we have:

```
# Add indicator variable called female to wages dataframe
wages_df <- mutate(wages_df, female = if_else(gender == "Female", 1, 0))
```

Verify that a new column called female has been added to wages and consists of 0s and 1s according to the definition of $female_i$ set forth above.

It's important to name an indicator variable in reference to the *base* or *benchmark group*; that is, the group that receives a value of 1, which in this example consists of females. If you had named the indicator variable $gender_i$ instead of $female_i$ you wouldn't know whether females or males were the benchmark group. Naming the variable $female_i$ indicates which group is the benchmark group, which greatly facilitates the interpretation of estimates of indicator variables.

Note also that by assigning all females in the dataset the value 1 and all males the value 0, we've quantified a qualitative distinction. We can now include $female_i$ in a regression model. But, as we're about to find out, the real challenge of using qualitative variables is not defining them, but rather in how to interpret them.

Regressions with an indicator variable don't alter in any formal sense the specification of the model. In the present case we'll write the regression equation as:

$$wage_i = \beta_0 + \delta_0 female_i + \epsilon_i,$$

where the only difference from past regressions concerns how we've denoted the coefficient of the indicator variable. Although not necessary, we denoted the coefficient of female by δ_0 to remind us that we're not dealing with a typical continuous variable (e.g., incomes, price levels), but rather with an indicator variable that can take on only two (arbitrarily-determined) values.

Think about what this regression implies. For each male in the dataset $female_i = 0$, while for females $female_i = 1$. We therefore obtain two different regression equations; one for males and one for females:

$$wage_i = \begin{cases} \beta_0 + \epsilon, & \text{if } female_i = 0 \\ \beta_0 + \delta_0 + \epsilon, & \text{if } female_i = 1. \end{cases}$$

To simplify notation, we'll write $wage_i^m$ if the wage refers to a male worker and $wage_i^f$ if the wage refers to a female worker.

Now, after estimating the model we obtain estimates for $\hat{\beta}_0$ and $\hat{\delta}_0$, giving us the estimated equations:

$$\hat{wage}^m = \hat{\beta}_0$$
$$\hat{wage}^f = \hat{\beta}_0 + \hat{\delta}_0.$$

Notice that if we were to graph these two equations they would appear as horizontal lines. (Remember that $\hat{\beta}_0$ and $\hat{\delta}_0$ are constants (specific numbers).) The horizontal line for males would be located at $\hat{\beta}_0$ and the horizontal line for females located at $\hat{\beta}_0 + \hat{\delta}_0$. The vertical distance between the two horizontal lines, how much they *differ* by, is determined by $\hat{\delta}_0$. Indeed, $\hat{\delta}_0$ estimates the *difference* in wages between females and males.

More specifically, though, how do we interpret these estimates? Consider the estimated equation for males. The predicted wage for males equals $\hat{\beta}_0$. The estimate $\hat{\beta}_0$ is an intercept, and like all intercepts in linear regression models it represents an average of the dependent variable.

The predicted wage for females is $\hat{\beta}_0 + \hat{\delta}_0$. It's the predicated wage for males, $\hat{beta}_0$, plus $\hat{\delta}_0$. But what's the meaning of $\hat{\delta}_0$? Note that if we subtracted the estimated equation for males from the estimated equation for females we'd be left with $\hat{\delta}_0$. That is:

$$\hat{wage}^f - \hat{wage}^m = (\hat{\beta}_0 + \hat{\delta}_0) - \hat{\beta}_0 = \hat{\delta}_0.$$

As mentioned above, $\hat{\delta}_0$ is the estimated difference in wages between females and males. It will be important in what follows to understand what it means when $\hat{\delta}_0 > 0$ versus when $\hat{\delta}_0 < 0$.

Finally, another way to think about $\hat{\delta}_0$ is as a slope. This interpretation will be the one you'll need to think of when plotting a regression line to a scatterplot of the data. Slopes are rise-over-run relations. The independent variable, $female_i$, is a binary variable and thus assumes either a 1 or 0. This means that the predicted slope is given as:

$$\frac{\hat{wage}^f - \hat{wage}^m}{1 - 0} = \hat{wage}^f - \hat{wage}^m = \hat{\delta}_0.$$

With this background on indicator variables, let's analyze and estimate the gender wage gap.

1. Create a scatterplot of earnings by gender using female as the independent variable and wage as the dependent variable. Also, include a regression line to your plot without confidence intervals.

2. Run a linear regression and obtain results.

3. Answer the following questions:

Question 1: Look at your scatterplot (just the plot; ignore the regression line). What's different about this plot compared to all previous plots you've generated in lab? What explains this difference?

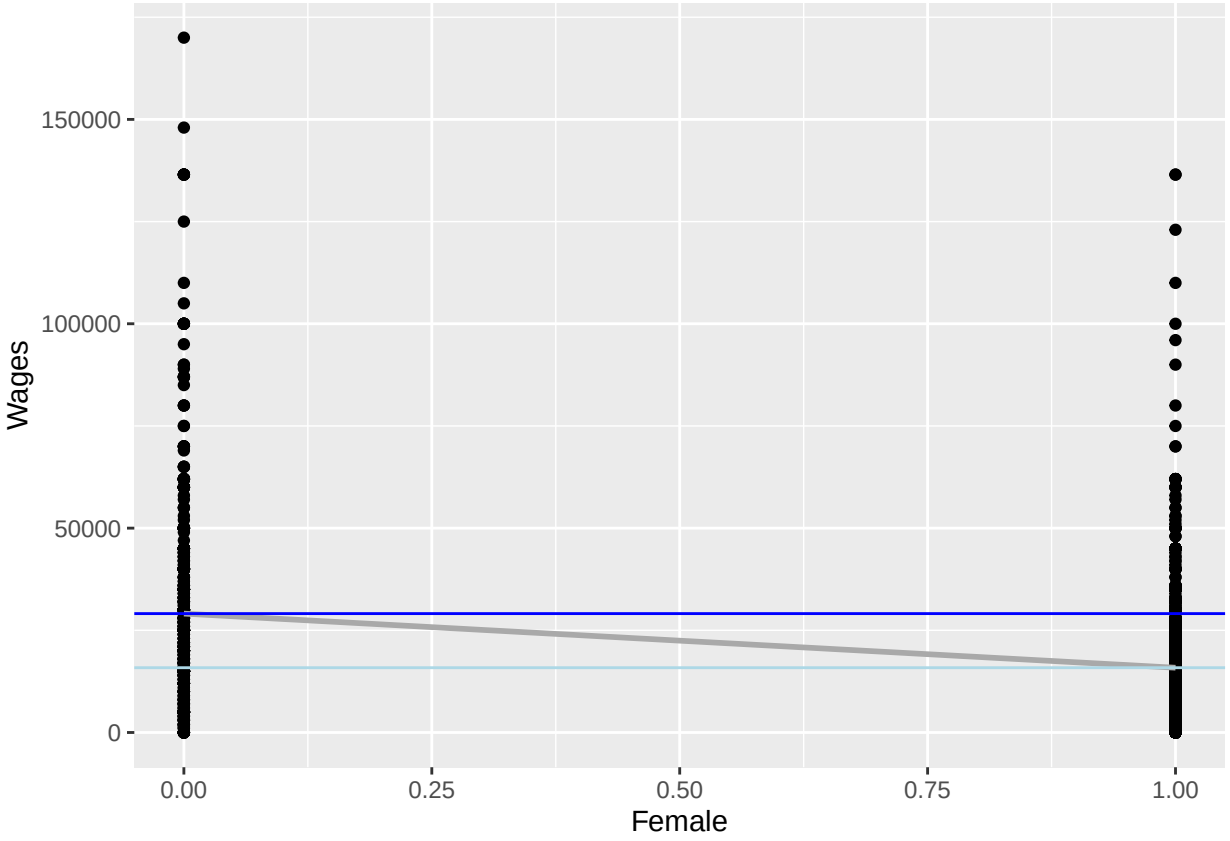
Recall how to subset data frames using square brackets, `[]`, and a condition inside the brackets. To find mean wages for females and males, wrap `mean()` around the subsetting commands:

```
# Compute mean wages for male workers and female works
mean(wages_df$wage[wages_df$female == 0])
mean(wages_df$wage[wages_df$female == 1])
```

Let's include these means as horizontal lines in a scatterplot. To do so, recreate the scatterplot with the regression line, but adding `geom_hline()` twice to produce two horizontal lines. To locate these lines on the plot, inside `geom_hline()` insert `yintercept =` and then mean wages for males and females. In fact, instead of manually typing in 29086.3 and 15847.94 for the means, you can copy and paste `mean(wages_df$wage[wages_df$female == 0])` and `mean(wages_df$wage[wages_df$female == 1])` after `=` inside `geom_hline()`, as shown in the following:

```
ggplot(data = wages_df, mapping = aes(x = female, y = wage)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE, color = "darkgray") +
  geom_hline(yintercept = mean(wages_df$wage[wages_df$female == 0]), color = "blue") +
  geom_hline(yintercept = mean(wages_df$wage[wages_df$female == 1]), color = "lightblue") +
  labs(x = "Female", y = "Wages")
```

```
## `geom_smooth()` using formula 'y ~ x'
```



Question 2: Given the foregoing discussion on the meaning of the coefficient estimates, does it makes sense that mean wages of females and males should or should not be on the regression line. Explain.

Question 3: Now consider your regression results. Interpret your estimate for the intercept and for the slope. Do your estimates make economic sense in relation to the gender pay gap? Explain.

Multiple Regression: Gender Gaps and Returns to Education

A well-established econometric finding is that the higher the level of education individuals attain, the higher their earnings. Let's examine the so-called "return to education," but do so while keeping $female_i$ in the model.

We'll specify this new regression as follows:

$$wage_i = \beta_0 + \delta_0 female_i + \beta_1 educ_i + \epsilon_i,$$

where $educ_i$ is the number of years of education completed by individuals. With the addition of $educ_i$ the coefficient δ_0 now has a new meaning: It's the wage differential between female and male workers *with the same level of education.*

Including a continuous variable like education in a regression with an indicator variable has a nice graphical counterpart as shown Figure 1. Note that the indicator variable introduces an *intercept shift* such that males and females have different regression lines. However, the slopes of the two lines are equal, meaning the effect of educational attainment on wages is the same for males and females.

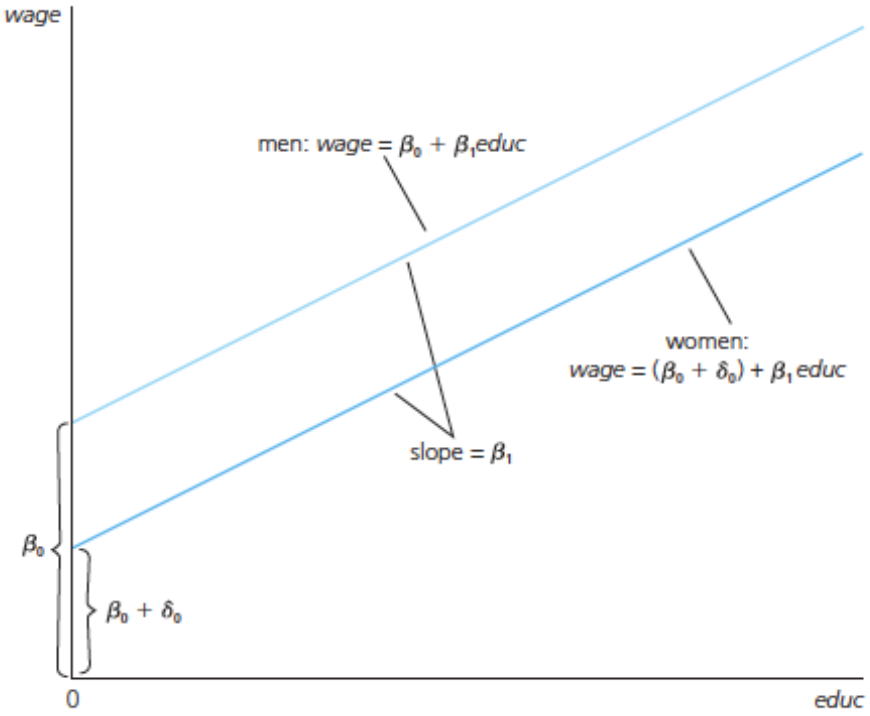


Figure 1: Graph of $wage = \beta_0 + \delta_0 female + \beta_1 educ$ for $\delta_0 < 0$

Adding education as a second independent variable to the model means we're now running a *multiple* regression model. Multiple regression models are easy to implement in R; we just insert a `+` followed by the variable's name after the first independent variable. So your command should look something like:

```
# Run regression of wage on gender and education level
gender_educ_fit <- lm(wage ~ female + educ, data = wages_df)
```

Question 4: Run the multiple regression model. Given the results, provide specific (i.e., numerical) interpretations of the coefficient estimates for the intercept, for female and the slope estimate for `educ`.

Question 5: Are the results consistent with your understanding of the gender wage gap and the return to education? Explain.