CSC100/CSC200 Lab #12: Regression

Dominick Bello

Fall 2022 | Data Science for the World

Please complete this notebook by filling in the cells provided. When you're done:

1. Remember to put your name in the header at the top of this notebook where it says `author`.
2. Select `Knit (Knit to Word)` from the toolbar menu.
3. Read that file! If any of your lines are too long and get cut off, we won't be able to see them, so break them up into multiple lines and knit again.
4. Save that Word document as a PDF file.
5. Submit BOTH this `.Rmd` file and the **PDF** file you generated to Gradescope. Some questions are autograded and you may improve your score on the tests given by resubmitting your work as many times as you like up to the deadline.
6. **Passing the automatic tests given does not guarantee full credit on any question.** The tests are provided to help catch some common mistakes, but it is *your* responsibility to answer the questions correctly.

If you are having trouble submitting, ask your lab instructor for help.

This lab assignment is due **November 30 at 11:59PM**.

Reading:

- Chapter 11 textbook

Run the cell below to prepare the notebook.

Part I: How Faithful is Old Faithful? (Note: clever title comes from here.) Old Faithful is a geyser in Yellowstone National Park in the central United States. It's famous for erupting on a fairly regular schedule. You can see a video of it in action here.

Some of Old Faithful's eruptions last longer than others. When it has a long eruption, there's generally a longer wait until the next eruption.

If you visit Yellowstone, you might want to predict when the next eruption will happen, so you can see the rest of the park and later see the geyser when it happens. In lab today we will use a dataset on eruption durations and waiting times to see if we can make such predictions accurately with linear regression.

The dataset has one row for each observed eruption. It includes the following variables:

- **duration**: Eruption duration, in minutes
- **wait**: Time between this eruption and the next, also in minutes

Let's have a look at the data:

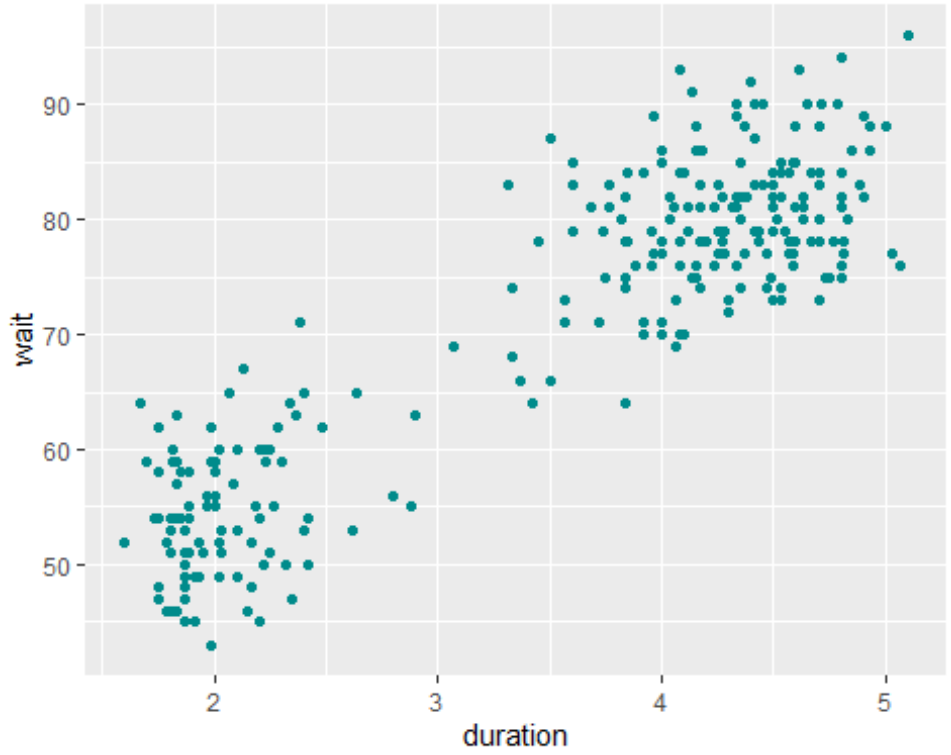
```
faithful
```

```
## # A tibble: 272 × 2
##    duration  wait
##       <dbl> <dbl>
##  1     3.6     79
##  2     1.8     54
##  3     3.33    74
##  4     2.28    62
##  5     4.53    85
##  6     2.88    55
##  7     4.7     88
##  8     3.6     85
##  9     1.95    51
## 10     4.35    85
## # … with 262 more rows
```

We would like to use linear regression to make predictions, but that won't work well if the data aren't roughly linearly associated. To check that, we should visualize the data.

Question 1. Make a scatter plot of the data. It is convention to put the variable we will try to predict on the vertical axis and the other variable – the *predictor* – on the horizontal axis.

```
faithful %>%
  ggplot() +
  geom_point(aes(x = duration, y = wait), color = "darkcyan")
```



Question 2. Are eruption duration and waiting time roughly linearly associated? Is the relationship positive, negative, or neither?

The eruption duration is rougly linear and is a positive relationship.

We're going to continue with the assumption that they are linearly associated, so it is reasonable to use linear regression to analyze this data. We'd next like to plot the data in standard units.

Question 3. Create a tibble called `faithful_standard` containing the eruption durations and waiting times *in standard units*. There should be two variables in this tibble: `duration_su` and `wait_su`.

```
faithful_standard <- faithful %>%
 transmute(x = duration,
           y = wait,
           duration_su = scale(x),
           wait_su = scale(y)) %>%
  select(duration_su,wait_su)

faithful_standard
```

```
## # A tibble: 272 × 2
##    duration_su[,1] wait_su[,1]
##              <dbl>       <dbl>
##  1          0.0983       0.596
##  2         -1.48        -1.24
```

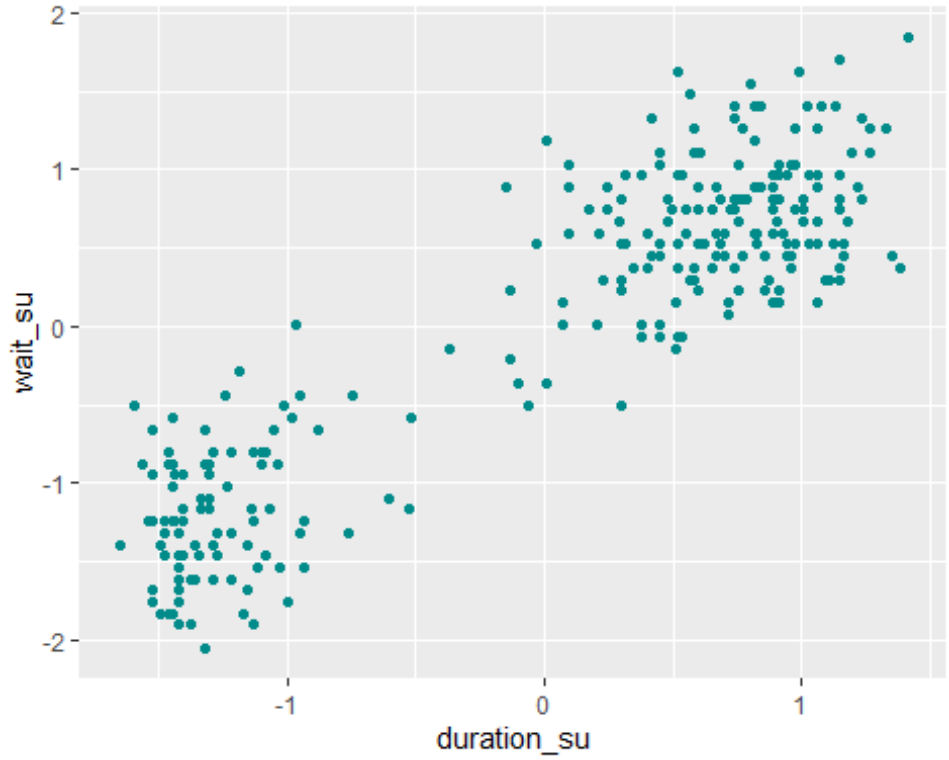
```
##  3          -0.136         0.228
##  4          -1.06         -0.654
##  5           0.916         1.04
##  6          -0.530        -1.17
##  7           1.06          1.26
##  8           0.0983        1.04
##  9          -1.35         -1.46
## 10           0.755         1.04
## # … with 262 more rows

. = ottr::check("tests/part1_q3.R")

## All tests passed!
```

Question 4. Plot the data again, but this time in standard units.

```
faithful_standard %>%
  ggplot() +
  geom_point(aes(x = duration_su, y = wait_su), color = "darkcyan")
```



Notice that this plot looks exactly the same as the last one! The data really are different, but the axes are scaled differently. It is important to read the ticks on the axes.

Question 5. Among the following numbers, which would you guess is closest to the correlation between eruption duration and waiting time in this dataset? Assign your guess to the name `my_corr_guess`.

- -1

- 0
- 1

```
my_corr_guess <- 0

. = ottr::check("tests/part1_q5.R")

## All tests passed!
```

Question 6. Compute the correlation `r` using `faithful_standard`. Do *NOT* try to shortcut this step by using `cor()`! You should use the same approach as shown in lecture and Section 11.1. Assign your answer to the name `r`.

```
r <- faithful_standard %>%
  mutate(prod= duration_su*wait_su) %>%
  pull(prod) %>%
  mean()

r

## [1] 0.8974994

. = ottr::check("tests/part1_q6.R")

## All tests passed!
```

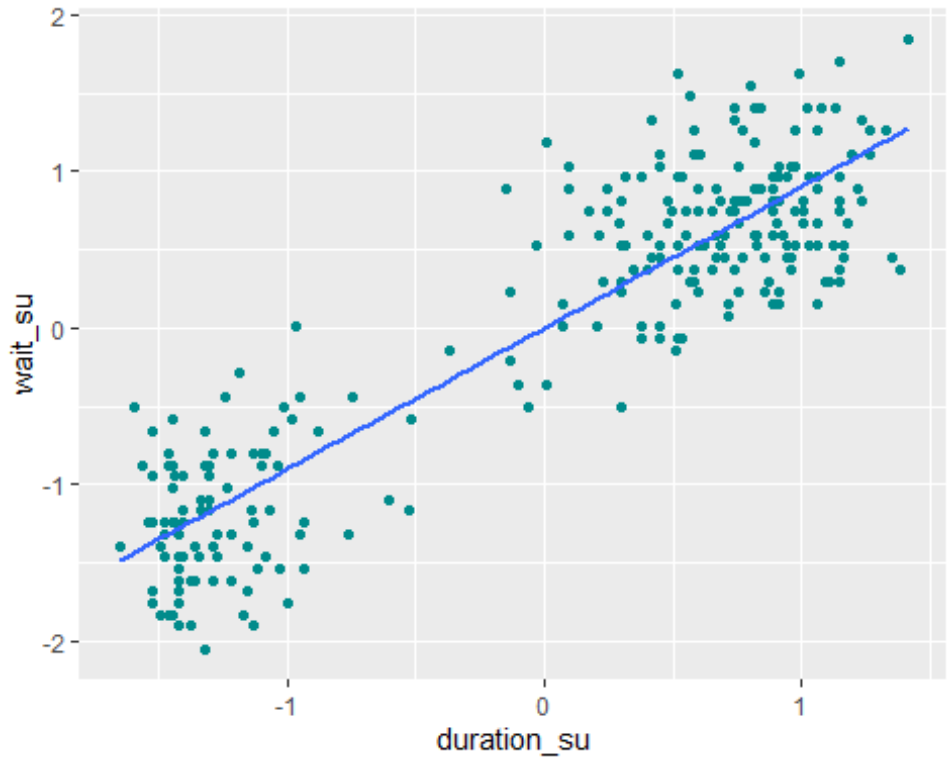
Part II: The regression line. Recall that the correlation is the *slope* of the regression line when the data are put in standard units.

The next cell plots the regression line in standard units:

$$\text{waiting time (standard units)} = r \times \text{eruption duration (standard units)}.$$

The regression line is overlayed atop the original data, for comparison.

```
ggplot(faithful_standard,
       aes(x = duration_su, y = wait_su)) +
  geom_point(color = "darkcyan") +
  geom_smooth(method = "lm", se = FALSE)
```



How would you take a point in standard units and convert it back to original units? We'd have to "stretch" its horizontal position by the SD of `duration` and its vertical position by the SD of `wait`. That means the same thing would happen to the slope of the line.

- Stretching a line horizontally makes it less steep, so we divide the slope by the stretching factor.
- Stretching a line vertically makes it more steep, so we multiply the slope by the stretching factor.

Question 1. What is the slope of the regression line in *original units*? Use `dplyr` code and the tibble `faithful` to answer this. Assign your answer (a double value) to the name `faithful_slope`.

```
slope <- function(tibble, x, y) {
  tibble %>%
    summarize(sd_x = sd({{ x }}),
              sd_y = sd({{ y }}),
              r = cor({{ x }}, {{ y }}),
              slope = r * sd_y / sd_x) %>%
    pull(slope)
}

faithful_slope <- slope(faithful, duration, wait)
faithful_slope
```

```
## [1] 10.72964
```

```
faithful_slope
```

```
## [1] 10.72964
```

```
. = ottr::check("tests/part2_q1.R")
```

```
## All tests passed!
```

We know that the regression line passes through the point `(duration_mean, wait_mean)`. You might recall from high school algebra that the equation for the line is therefore:

$$\text{waiting time} - \verb|wait_mean| = \texttt{slope} \times (\text{eruption duration} - \verb|duration_mean|)$$

Question 2. After rearranging that equation slightly, what is the intercept *in original units*? Assign your answer (a double value) to the name `faithful_intercept`.

```
intercept <- function(tibble, x, y) {
  tibble %>%
    summarize(y_avg = mean({{ y }}),
              x_avg = mean({{ x }}),
              intercept = y_avg - slope(tibble, {{ x }}, {{ y }}) * x_avg)
%>%
    pull(intercept)
}

faithful_intercept <- intercept(faithful, duration, wait)


tibble(faithful_intercept)
```

```
## # A tibble: 1 × 1
##   faithful_intercept
##                <dbl>
## 1               33.5
```

```
. = ottr::check("tests/part2_q2.R")
```

```
## All tests passed!
```

Part III: Investigating the regression line. The slope and intercept tell you exactly what the regression line looks like. To predict the waiting time for an eruption is easy! Just multiply the eruption's duration by `faithful_slope` and then add `faithful_intercept`.

Question 1. Compute the predicted waiting time for an eruption that lasts 2 minutes, and for an eruption that lasts 5 minutes.

```
two_minute_predicted_waiting_time <- (2*faithful_slope)+faithful_intercept
five_minute_predicted_waiting_time <- (5*faithful_slope)+faithful_intercept

print(two_minute_predicted_waiting_time)
```

```
## [1] 54.93368
```

```
print(five_minute_predicted_waiting_time)
```

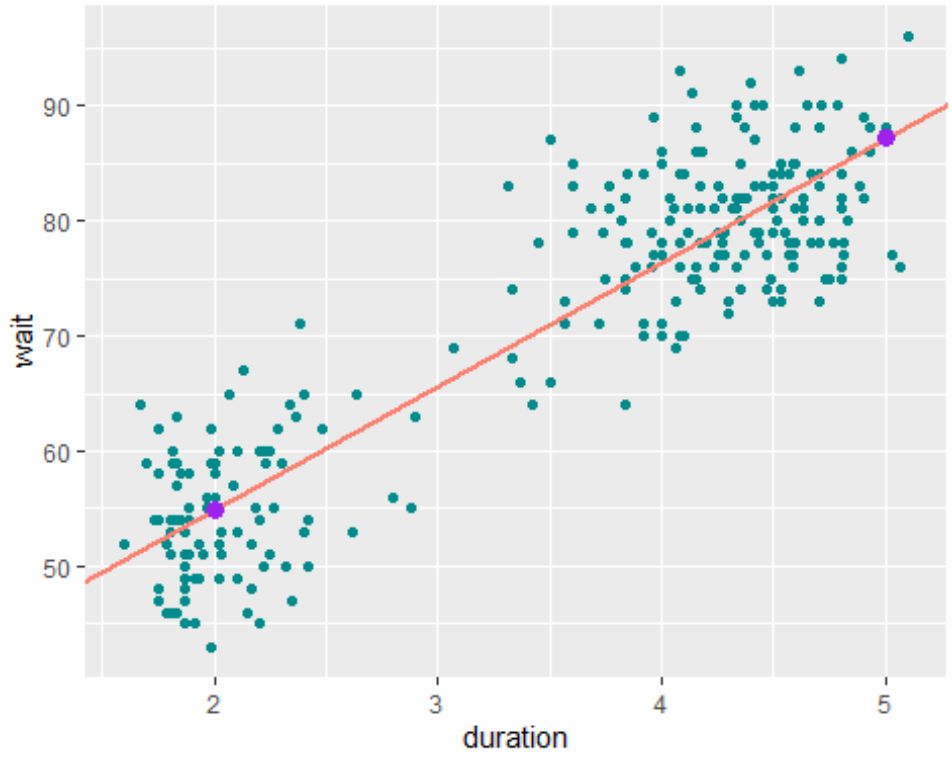
```
## [1] 87.1226
```

```
. = ottr::check("tests/part3_q1.R")
```

```
## All tests passed!
```

The next cell plots the regression line and these two points (in purple); you can see that the regression line passes through them.

```
ggplot(faithful,
       aes(x = duration, y = wait)) +
  geom_point(color = "darkcyan") +
  geom_abline(aes(slope = faithful_slope,
                  intercept = faithful_intercept),
              size = 1, color = "salmon") +
  geom_point(aes(x = 2,
                 y = two_minute_predicted_waiting_time),
             size = 3, color = "purple") +
  geom_point(aes(x = 5,
                 y = five_minute_predicted_waiting_time),
             size = 3, color = "purple")
```



Question 2. Make predictions for the waiting time after each eruption in the `faithful` tibble. (Of course, we know exactly what the waiting times were! We are doing this so we

can see how accurate our predictions are.) Put these numbers into a variable called `prediction` in a new tibble called `faithful_predictions`. Its first row should look like this:

duration	wait	prediction
3.6	79	72.096

```
faithful_predictions<- faithful%>%
  mutate(prediction=(duration*faithful_slope)+faithful_intercept)

faithful_predictions
```

```
## # A tibble: 272 × 3
##    duration  wait prediction
##       <dbl> <dbl>      <dbl>
##  1     3.6     79       72.1
##  2     1.8     54       52.8
##  3     3.33    74       69.2
##  4     2.28    62       58.0
##  5     4.53    85       82.1
##  6     2.88    55       64.4
##  7     4.7     88       83.9
##  8     3.6     85       72.1
##  9     1.95    51       54.4
## 10     4.35    85       80.1
## # … with 262 more rows
```

```
. = ottr::check("tests/part3_q2.R")

## All tests passed!
```

Question 3. How close were we? Compute the *residual* for each eruption in the dataset. The residual is the difference (**not** the absolute difference!) between the actual waiting time and the predicted waiting time. Add the residuals to `faithful_predictions` as a new variable called `residual`, naming the resulting tibble `faithful_residuals`.

```
faithful_residuals<-faithful_predictions%>%
  mutate(residual= wait-prediction)

faithful_residuals
```

```
## # A tibble: 272 × 4
##    duration  wait prediction residual
##       <dbl> <dbl>      <dbl>    <dbl>
##  1     3.6     79       72.1     6.90
##  2     1.8     54       52.8     1.21
##  3     3.33    74       69.2     4.76
##  4     2.28    62       58.0     4.03
##  5     4.53    85       82.1     2.89
##  6     2.88    55       64.4    -9.41
##  7     4.7     88       83.9     4.10
```

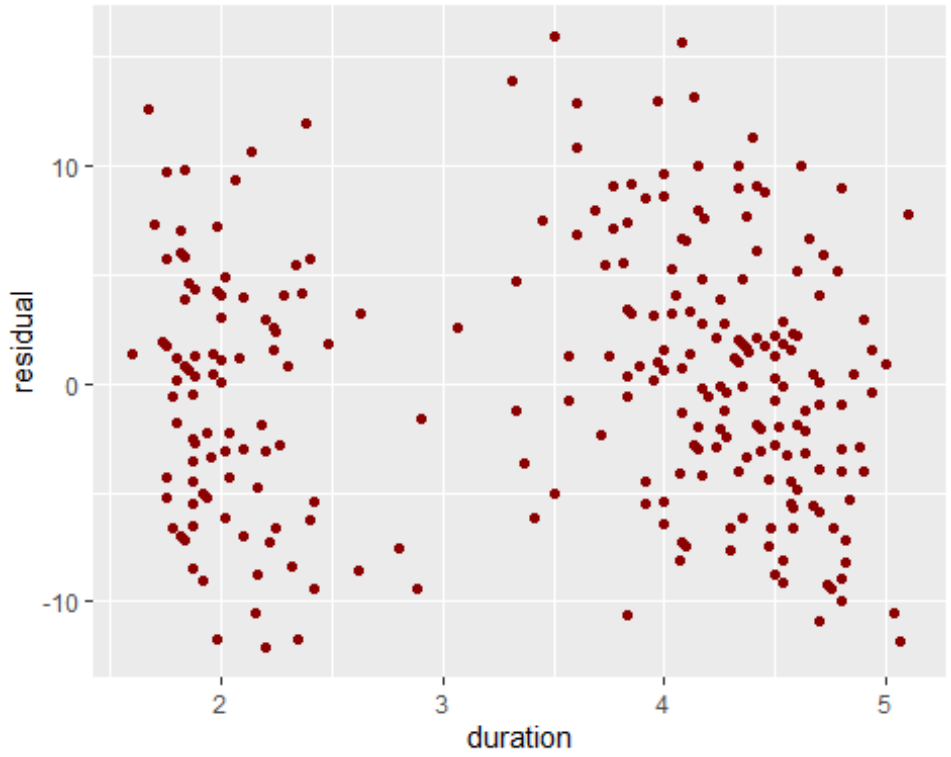
```
##  8     3.6     85       72.1    12.9
##  9     1.95    51       54.4    -3.40
## 10     4.35    85       80.1     4.85
## # … with 262 more rows

. = ottr::check("tests/part3_q3.R")

## All tests passed!
```

Here is a plot of the residuals you computed. Each point corresponds to one eruption. It shows how much our prediction over- or under-estimated the waiting time.

```
ggplot(faithful_residuals,
       aes(x = duration, y = residual)) +
  geom_point(color = "darkred")
```



There isn't really a pattern in the residuals, which confirms that it was reasonable to try linear regression. It's true that there are two separate clouds; the eruption durations seemed to fall into two distinct clusters. But that's just a pattern in the eruption durations, not a pattern in the relationship between eruption durations and waiting times.

Part IV: How accurate are different predictions? Earlier you should have found that the correlation is fairly close to 1, so the line fits fairly well on the data we have available in `faithful` (students taking a machine learning course would rightly label this data as *training data*). That means the residuals are overall small (close to 0; see the residual plot above) in comparison to the waiting times.

However, unless there is a strong reason to believe that the linear regression model is true, you should be wary of applying your prediction model to data that are very different from the training data.

Question 1. In `faithful`, no eruption lasted exactly 0, 2.5, or 60 minutes. Using this line, what is the predicted waiting time for an eruption that lasts 0 minutes? 2.5 minutes? An hour? Assign your answers to the names in the following cell:

```
zero_minute_predicted_waiting_time <- 20
two_point_five_minute_predicted_waiting_time <- 40
hour_predicted_waiting_time <- 800

print(paste("After an eruption lasting", 0, "minutes, we predict you'll wait
", zero_minute_predicted_waiting_time, "minutes until the next eruption."))
```

```
## [1] "After an eruption lasting 0 minutes, we predict you'll wait  20
minutes until the next eruption."
```

```
print(paste("After an eruption lasting", 2.5, "minutes, we predict you'll
wait ", two_point_five_minute_predicted_waiting_time, "minutes until the next
eruption."))
```

```
## [1] "After an eruption lasting 2.5 minutes, we predict you'll wait  40
minutes until the next eruption."
```

```
print(paste("After an eruption lasting", 60, "minutes, we predict you'll wait
", hour_predicted_waiting_time, "minutes until the next eruption."))
```

```
## [1] "After an eruption lasting 60 minutes, we predict you'll wait  800
minutes until the next eruption."
```

```
. = ottr::check("tests/part4_q1.R")
```

```
## All tests passed!
```

Question 2. Do you believe any of these values are reliable predictions? Explain why or why not.

I do not think these numbers are very accurate because the two intervals don't correlated to the data we see.

Yippee – you are done with Lab #12! Time to submit.