Lab 4

Maria Barouti

2/3/2020

Based on `Applied Statistics with R (appliedstats)` by David Dalpiaz (https://github.com/daviddalpiaz/appliedstats)

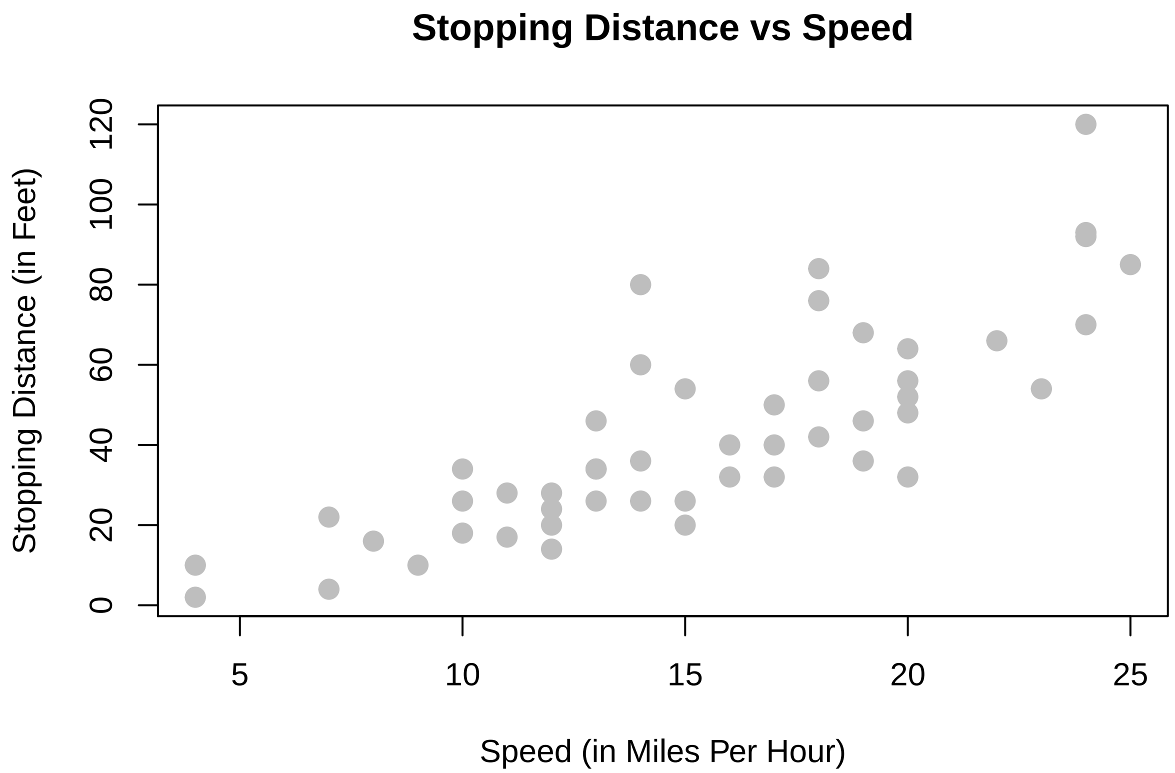
`cars` Example

We use a simple example of how the speed of a car affects its stopping distance, that is, how far it travels before it comes to a stop. To examine this relationship, we will use the `cars` dataset which, is a default `R` dataset. Thus, we don't need to load a package first; it is immediately available.

```
View(cars)
```

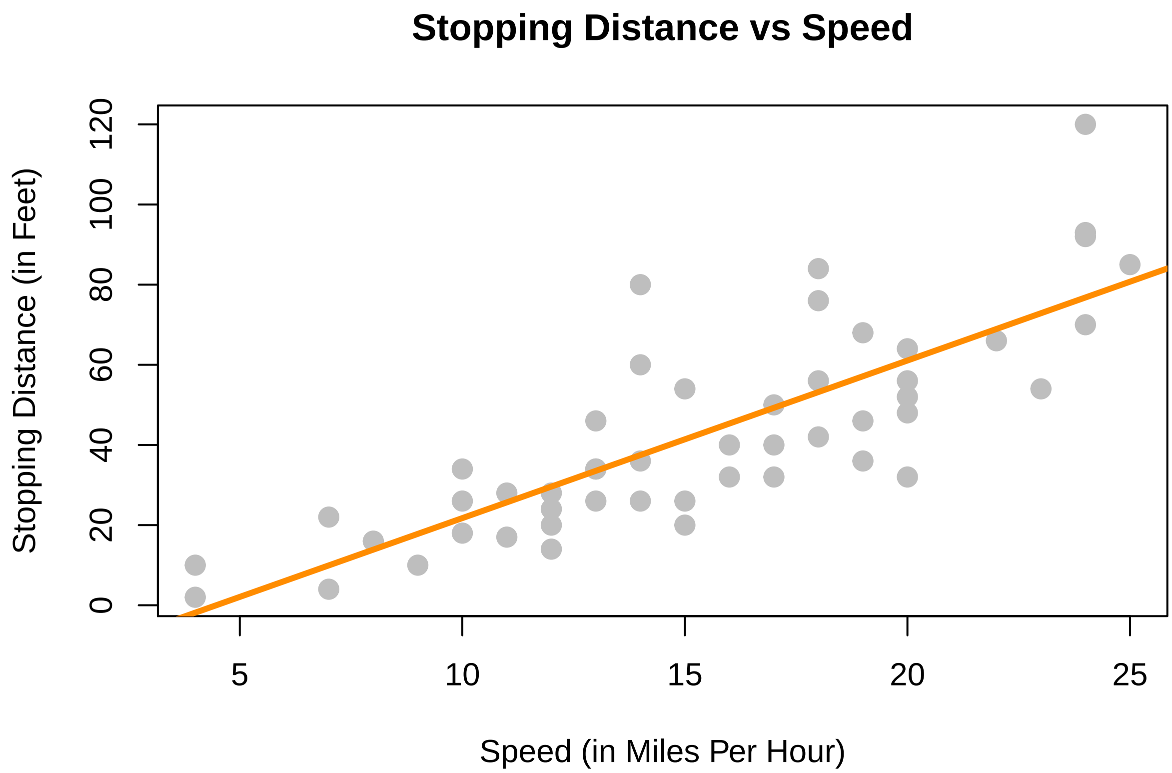
Reading the documentation we learn that this is data gathered during the 1920s about the speed of cars and the resulting distance it takes for the car to come to a stop. The interesting task here is to determine how far a car travels before stopping, when traveling at a certain speed. So, we will first plot the stopping distance against the speed.

```
plot(dist ~ speed, data = cars,
     xlab = "Speed (in Miles Per Hour)",
     ylab = "Stopping Distance (in Feet)",
     main = "Stopping Distance vs Speed",
     pch  = 20,
     cex  = 2,
     col  = "grey")
```



The line on the plot below seems to summarize the relationship between stopping distance and speed quite well. As speed increases, the distance required to come to a stop increases. There is still some variation about this line, but it seems to capture the overall trend.

```
stop_dist_model = lm(dist ~ speed, data = cars)
plot(dist ~ speed, data = cars,
     xlab = "Speed (in Miles Per Hour)",
     ylab = "Stopping Distance (in Feet)",
     main = "Stopping Distance vs Speed",
     pch  = 20,
     cex  = 2,
     col  = "grey")
abline(stop_dist_model, lwd = 3, col = "darkorange")
```



Fit the model

Fit the model using `lm()` then use `summary()` to view the results in greater detail.

```
stop_dist_model = lm(dist ~ speed, data = cars)
summary(stop_dist_model)
```

```
##
## Call:
## lm(formula = dist ~ speed, data = cars)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -29.069  -9.525  -2.272   9.215  43.201
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -17.5791     6.7584  -2.601   0.0123 *
## speed         3.9324     0.4155   9.464 1.49e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 15.38 on 48 degrees of freedom
## Multiple R-squared:  0.6511, Adjusted R-squared:  0.6438
## F-statistic: 89.57 on 1 and 48 DF,  p-value: 1.49e-12
```

Tests in R

We will now discuss the results displayed called `Coefficients`. First recall that we can extract this information directly.

```r
names(summary(stop_dist_model))
```

```
##  [1] "call"          "terms"         "residuals"     "coefficients"
##  [5] "aliased"       "sigma"         "df"            "r.squared"
##  [9] "adj.r.squared" "fstatistic"    "cov.unscaled"
```

```r
summary(stop_dist_model)$coefficients
```

```
##               Estimate Std. Error   t value     Pr(>|t|)
## (Intercept) -17.579095  6.7584402 -2.601058 1.231882e-02
## speed         3.932409  0.4155128  9.463990 1.489836e-12
```

The `names()` function tells us what information is available, and then we use the `$` operator and `coefficients` to extract the information we are interested in. Two values here should be immediately familiar.

$$b_0 = -17.5790949$$

and

$$b_1 = 3.9324088$$

which are our estimates for the model parameters β_0 and β_1.

Let's now focus on the second row of output, which is relevant to β_1.

```r
summary(stop_dist_model)$coefficients[2,]
```

```
##     Estimate   Std. Error      t value     Pr(>|t|)
## 3.932409e+00 4.155128e-01 9.463990e+00 1.489836e-12
```

Again, the first value, `Estimate` is

$$b_1 = 3.9324088.$$

The second value, `Std. Error`, is the square root of the estimated variance (standard error) of b_1,

$$s\{b_1\} = \frac{\sqrt{\text{MSE}}}{\sqrt{\sum(X_i - \bar{X})}} = 0.4155128.$$

The third value, `t value`, is the value of the test statistic for testing $H_0 : \beta_1 = 0$ vs $H_1 : \beta_1 \neq 0$,

$$t = \frac{b_1 - 0}{s\{b_1\}} = 9.46399.$$

Lastly, `Pr(>|t|)`, gives us the p-value of that test.

$$\text{p-value} = 1.4898365 \times 10^{-12}$$

Note here, we are specifically testing whether or not $\beta_1 = 0$.

The first row of output reports the same values, but for β_0.

```r
summary(stop_dist_model)$coefficients[1,]
```

```
##     Estimate   Std. Error      t value     Pr(>|t|)
## -17.57909489   6.75844017  -2.60105800   0.01231882
```

In summary, the following code stores the information of `summary(stop_dist_model)$coefficients` in a new variable `stop_dist_model_test_info`, then extracts each element into a new variable which describes the information it contains.

```r
stop_dist_model_test_info = summary(stop_dist_model)$coefficients

b_0      = stop_dist_model_test_info[1, 1] # Estimate
b_0_se   = stop_dist_model_test_info[1, 2] # Std. Error
b_0_t    = stop_dist_model_test_info[1, 3] # t value
b_0_pval = stop_dist_model_test_info[1, 4] # Pr(>|t|)

b_1      = stop_dist_model_test_info[2, 1] # Estimate
b_1_se   = stop_dist_model_test_info[2, 2] # Std. Error
b_1_t    = stop_dist_model_test_info[2, 3] # t value
b_1_pval = stop_dist_model_test_info[2, 4] # Pr(>|t|)
```

Task

Verify some equivalent expressions: the t test statistic for b_1 and the two-sided p-value associated with that test statistic.

```r
(b_1 - 0) / b_1_se
```

```
## [1] 9.46399
```

```r
b_1_t
```

```
## [1] 9.46399
```

```r
2 * pt(abs(b_1_t), df = length(resid(stop_dist_model)) - 2, lower.tail = FALSE)
```

```
## [1] 1.489836e-12
```

```r
b_1_pval
```

```
## [1] 1.489836e-12
```

Significance of Regression, t-Test

We pause to discuss the **significance of regression** test. First, note that based on the above distributional results, we could test β_0 and β_1 against any particular value, and perform both one and two-sided tests.

However, one very specific test,

$$H_0 : \beta_1 = 0 \quad \text{vs} \quad H_1 : \beta_1 \neq 0$$

is used most often. Let's think about this test in terms of the simple linear regression model,

$$Y_i = \beta_0 + \beta_1 X_i + \epsilon_i.$$

If we assume the null hypothesis is true, then $\beta_1 = 0$ and we have the model,

$$Y_i = \beta_0 + \epsilon_i.$$

In this model, the response does **not** depend on the predictor. So then we could think of this test in the following way,

- Under H_0 there is not a significant linear relationship between X and Y.
- Under H_1 there is a significance **linear** relationship between Y and Y.

For the `cars` example,

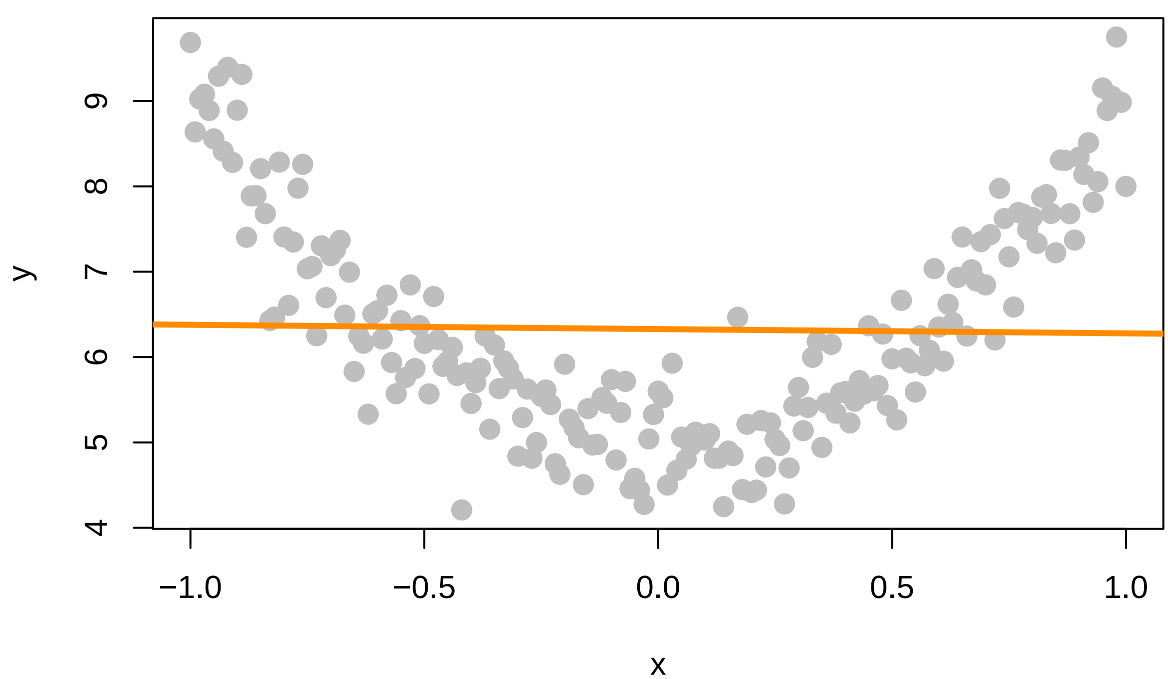
- Under H_0 there is not a significant linear relationship between speed and stopping distance.
- Under H_1 there is a significant **linear** relationship between speed and stopping distance.

Again, that test is seen in the output from `summary()`,

$$\text{p-value} = 1.4898365 \times 10^{-12}.$$

With this extremely low p-value, we would reject the null hypothesis at any reasonable α level, say for example $\alpha = 0.01$. So we say there is a significant **linear** relationship between speed and stopping distance. Notice that we emphasize **linear**.

```r
set.seed(42)
x = seq(-1, 1, 0.01)
y = 5 + 4 * x ^ 2 + rnorm(length(x), 0, 0.5)
plot(x, y, pch  = 20, cex  = 2, col  = "grey")
abline(lm(y ~ x), lwd = 3, col = "darkorange")
```



In this plot of simulated data, we see a clear relationship between x and y, however it is not a linear relationship. If we fit a line to this data, it is very flat. The resulting test for $H_0 : \beta_1 = 0$ vs $H_1 : \beta_1 \neq 0$ gives a large p-value, in this case 0.7564548, so we would fail to reject and say that there is no significant linear relationship between x and y. We will see later how to fit a curve to this data using a "linear" model, but for now, realize that testing $H_0 : \beta_1 = 0$ vs $H_1 : \beta_1 \neq 0$ can only detect straight line relationships.

Confidence Intervals in R

Using R we can very easily obtain the confidence intervals for β_0 and β_1.

```
confint(stop_dist_model, level = 0.99)
```

```
##                  0.5 %    99.5 %
## (Intercept) -35.706610 0.5484205
## speed         2.817919 5.0468988
```

This automatically calculates 99% confidence intervals for both β_0 and β_1, the first row for β_0, the second row for β_1.

For the `cars` example when interpreting these intervals, we say, we are 99% confident that for an increase in speed of 1 mile per hour, the average increase in stopping distance is between 2.8179187 and 5.0468988 feet, which is the interval for β_1.

Note that this 99% confidence interval does **not** contain the hypothesized value of 0. Since it does not contain 0, it is equivalent to rejecting the test of $H_0 : \beta_1 = 0$ vs $H_1 : \beta_1 \neq 0$ at $\alpha = 0.01$, which we had seen previously.

You should be somewhat suspicious of the confidence interval for β_0, as it covers negative values, which correspond to negative stopping distances. Technically the interpretation would be that we are 99% confident that the average stopping distance of a car traveling 0 miles per hour is between -35.7066103 and 0.5484205 feet, but we don't really believe that, since we are actually certain that it would be non-negative.

Note, we can extract specific values from this output a number of ways.

```
confint(stop_dist_model, level = 0.99)[1,]
```

```
##       0.5 %      99.5 %
## -35.7066103   0.5484205
```

```
confint(stop_dist_model, level = 0.99)[1, 1]
```

```
## [1] -35.70661
```

```
confint(stop_dist_model, level = 0.99)[1, 2]
```

```
## [1] 0.5484205
```

```
confint(stop_dist_model, parm = "(Intercept)", level = 0.99)
```

```
##                 0.5 %    99.5 %
## (Intercept) -35.70661 0.5484205
```

```
confint(stop_dist_model, level = 0.99)[2,]
```

```
##    0.5 %   99.5 %
## 2.817919 5.046899
```

```
confint(stop_dist_model, level = 0.99)[2, 1]
```

```
## [1] 2.817919
```

```r
confint(stop_dist_model, level = 0.99)[2, 2]
```

```
## [1] 5.046899
```

```r
confint(stop_dist_model, parm = "speed", level = 0.99)
```

```
##          0.5 %   99.5 %
## speed 2.817919 5.046899
```

Task

Verify that calculations that `R` is performing for the β_1 interval.

```r
# store estimate
b_1 = coef(stop_dist_model)[2]

# store standard error
b_1_se = summary(stop_dist_model)$coefficients[2, 2]

# calculate critical value for two-sided 99% CI
crit = qt(0.995, df = length(resid(stop_dist_model)) - 2)

# est - margin, est + margin
c(b_1 - crit * b_1_se, b_1 + crit * b_1_se)
```

```
##    speed    speed
## 2.817919 5.046899
```