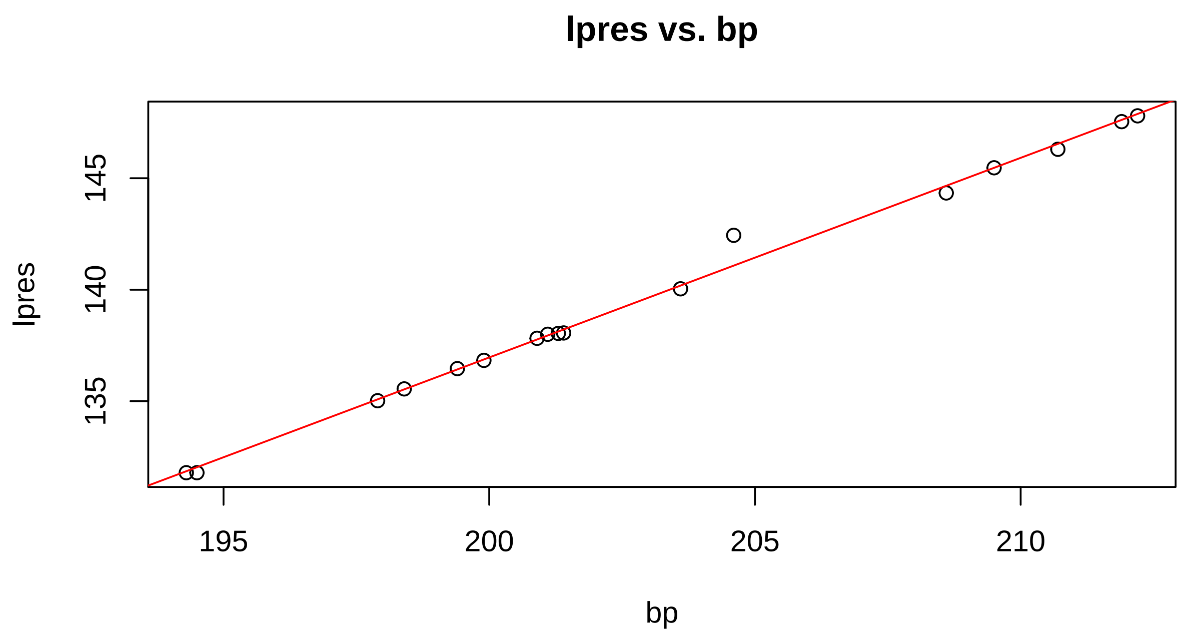
STAT350 Tutorial 2

15/09/2020

This weeks tutorial covers hypothesis tests and interval estimation for the parameters of the simple linear regression (SLR) model, along with prediction of new observations and the coefficient of determination.

We continue with the example from last tutorial where we used the Forbes data to create a SLR with `lpres` as the dependent variable and `bp` as the independent variable. The model call and plot of the data with the fit regression line are below.

```
##
## Call:
## lm(formula = lpres ~ bp, data = forbes_data)
##
## Coefficients:
## (Intercept)           bp
##    -42.1378       0.8955
```



I'm also going to save the model summary object to `my_sum`.

```r
my_sum <- summary(my_mdl)
my_sum
```

```
##
## Call:
## lm(formula = lpres ~ bp, data = forbes_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.32220 -0.14473 -0.06664  0.02184  1.35978
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -42.13778    3.34020  -12.62 2.18e-09 ***
## bp            0.89549    0.01645   54.43  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.379 on 15 degrees of freedom
## Multiple R-squared:  0.995,  Adjusted R-squared:  0.9946
## F-statistic:  2963 on 1 and 15 DF,  p-value: < 2.2e-16
```

The model and summary objects contain many parts. If you want more information about these you can run `help(lm)` or `help(summary.lm)`, respectively. The `help()` function is generally useful if you are confused with any other function you are using in R.

Hypothesis Test for the βs

Typically we are interested in testing whether or not a parameter should be included in the model. Here we'll consider the slope. The hypothesis for this is:

$$\mathrm{H}_0: \ \beta_1 = 0 \ \text{ vs. } \ \mathrm{H}_A: \beta_1 \neq 0.$$

To test this we'll go through same procedure as other hypothesis test you've come across; compute the test statistic based on the data, find its corresponding p-value, then compare to α.

Here we'll be doing a t-test where the test statistic is given by:

$$t = \frac{\hat{\beta}_1}{se(\hat{\beta}_1)}.$$

After calculating this value, we could then compute the corresponding p-value with the `pt()`, specifying the correct degrees of freedom.

Luckily for us, all values we need can be found in the summary:

```
my_sum$coefficients
```

```
##                Estimate Std. Error   t value     Pr(>|t|)
## (Intercept) -42.1377793 3.34019890 -12.61535 2.175977e-09
## bp            0.8954937 0.01645176  54.43147 1.186078e-18
```

For the slope (the coefficient for `bp`) we look at the second row. Here we see the estimated slope is 0.895 and has an estimated standard error of 0.016. The corresponding t-value is 54.431 and p-value 0. The p-value is extremely small, so for any reasonable value of α we'll reject the null hypothesis.

Other hypothesis tests for the coefficients such as a two-sided test if β equals a specific value or one-sided tests for checking if β is less than or greater than some value can also be performed. For these we perform a t-test as well, but the test statistic is slightly different:

$$t = \frac{\hat{\beta}_1 - \beta_1}{se(\hat{\beta}_1)}.$$

So we can't just pull the t- and p-values from the summary object, though their computation is relatively straight forward using the estimate and its standard error.

A side note: in the context of SLR the F-statistic seen in the summary

```
my_sum$fstatistic
```

```
##    value    numdf    dendf
## 2962.785    1.000   15.000
```

is just the squared t-value for the estimate of the slope.

```
my_sum$coefficients[2,3]^2
```

```
## [1] 2962.785
```

We can therefore test the same two-sided hypothesis as above, but now with an analysis-of-variance F-test. This test will be more useful when we cover multiple linear regression.

Intervals for the βs

A $100(1-\alpha)$ percent confidence interval (CI) on a given coefficient is given by

$$\hat{\beta} - t_{\alpha/2,n-2}se(\hat{\beta}) \leq \beta \leq \hat{\beta} + t_{\alpha/2,n-2}se(\hat{\beta}).$$

First, we assign the estimated coefficients and their standard errors to their own objects.

```
beta_hat <- my_sum$coefficients[,1]
beta_ses <- my_sum$coefficients[,2]
```

Next, we can compute the critical t-value. Here we'll be computing a 95% CI, so the t-value we are looking corresponds to the probability $1-(0.05/2)$. For the degrees of freedom, we can get it from the model object.

```
(tval <- qt(p = 0.975, df = my_mdl$df))
```

```
## [1] 2.13145
```

Now we have all the values we need and can compute the CI by applying the formula above.

```
(CIs <- data.frame(estimate = beta_hat,
                   lower = beta_hat - tval*beta_ses,
                   upper = beta_hat + tval*beta_ses))
```

```
##               estimate       lower       upper
## (Intercept) -42.1377793 -49.2572447 -35.0183139
## bp            0.8954937   0.8604276   0.9305598
```

Interpretation of confidence intervals is important and regularly confused by students. For a 95% confidence interval on β, we say that for each sample of data and CI computed in this way, 95% of the CIs will contain the true value of β.

Interval for σ^2

The estimate for σ^2 is not readily available from either the model or summary objects, but we can quickly compute it. First we can compute the residual sum of squares:

```
(SS_res <- sum(my_mdl$residuals^2))
```

```
## [1] 2.154927
```

Then divide that by the degrees of freedom:

```
(sigma2_hat <- SS_res/my_mdl$df.residual)
```

```
## [1] 0.1436618
```

So we have an estimated model variance of 0.144. To compute a CI for the parameter we use:

$$\frac{(n-2)\hat{\sigma}^2}{\chi^2_{\alpha/2,n-2}} \leq \sigma^2 \leq \frac{(n-2)\hat{\sigma}^2}{\chi^2_{1-\alpha/2,n-2}}$$

For a 95% CI we set α to 0.05 and our two χ^2-values are:

```
(chi_up <- qchisq(0.975, my_mdl$df.residual))
```

```
## [1] 27.48839
```

```
(chi_low <- qchisq(0.025, my_mdl$df.residual))
```

```
## [1] 6.262138
```

Finally, we have the confidence inter val for σ^2:

```
(CI_sigma <- data.frame(estimate = sigma2_hat,
                        lower = my_mdl$df.residual*sigma2_hat/chi_low,
                        upper = my_mdl$df.residual*sigma2_hat/chi_up))
```

```
##    estimate     lower      upper
## 1 0.1436618 0.3441201 0.07839408
```

Prediction

Given our fitted SLR model we can make predictions for new values of `bp`. When making predictions we want to avoid extrapolation, so I'll first check the range of `bp` values.

```r
range(forbes_data$bp)
```

```
## [1] 194.3 212.2
```

We'll make a prediction at `bp = 207`.

We make predictions with the `predict()` function. The first input is the model object. We the specify the new predictor value in `newdata`; the format here is important as the function needs a data frame with the same variables that defined the initial model.

```r
predict(my_mdl, newdata = data.frame(bp = 207))
```

```
##        1
## 143.2294
```

If we were to omit the `newdata` argument the function returns the fitted values.

```r
predict(my_mdl)
```

```
##        1        2        3        4        5        6        7        8
## 132.0357 131.8566 135.0804 135.5282 136.4237 136.8714 137.7669 137.9460
##        9       10       11       12       13       14       15       16
## 138.2146 138.1251 140.1847 141.0802 145.4681 144.6622 146.5427 147.6173
##       17
## 147.8860
```

The predict function can also give us intervals for our predictions through the `interval` argument. We have three choices: `'none'` (the default), `'confidence'`, or `'prediction'`. The default level for the intervals is 95%, but this can be changed with the `level` argument

Below We use `'confidence'` to obtain a 95% confidence interval for the mean response.

```r
(conf_int <- predict(my_mdl,
                     newdata = data.frame(bp = 207),
                     interval = 'confidence'))
```

```
##        fit      lwr      upr
## 1 143.2294 142.9875 143.4713
```

We use `'prediction'` if we want a prediction interval.

```
(pred_int <- predict(my_mdl,
                     newdata = data.frame(bp = 207),
                     interval = 'prediction'))
```

```
##        fit      lwr      upr
## 1 143.2294 142.3861 144.0727
```

As we can see the prediction interval is wider than the confidence interval, why? The confidence interval is for a parameter, in this case we are interested in the mean of the response values for a particular value of the independent variable. The prediction interval gives us the range for the observed value of the response, so there is more uncertainty here as we need to consider ϵ in it's calculation.

Coeffecient of Determination, R^2

The coefficient of determination is also part of the summary object.

```
my_sum$r.squared
```

```
## [1] 0.9949627
```

Also called the proportion of variance explained, it is a measure of how well our model performs. As you will see it is the ratio of the variation in the residuals to the variation in the response. Therefore $0 \leq R^2 \leq$. Values close to 1 mean we've explained most of the variability - our model performs well.

For SLR, it turns out that R^2 is just the squared correlation between the two variables.

```
cor(forbes_data$bp, forbes_data$lpres)^2
```

```
## [1] 0.9949627
```

The summary also contains something called the adjusted R^2.

```
my_sum$adj.r.squared
```

```
## [1] 0.9946269
```

This is a function of the R^2 and the sample size. Just ignore it for now.