17.3: Estimation of linear regression coefficient

Introduction

In discussing correlations we made the distinction between inference, testing the statistical significance of the estimate, and the process of getting the estimate of the parameter itself. **Estimating parameters** is possible for any data set; whether or not the particular model is a good and useful model is another matter. Statisticians speak about the **fit of a model**… that a model with one or more **independent predictor variables** explains a substantial amount of the variation in the **dependent variable**, that it describes the relationship between the predictors and the dependent variable without bias. A number of tools have been developed to assess model fit. For starters, I'll list just two ways you can approach whether a **linear model** fits your data or requires some intervention on your part.

Assess fit of a linear model

Recall our R output from the regression of Number of Matings on Body Mass from the bird data set. We used the linear model function.

```
LinearModel.1 <- lm(Matings ~ Body.Mass, data=bird_matings)
summary(LinearModel.1)
Call: lm(formula = Matings ~ Body.Mass, data = bird_matings)
Residuals:       Min       1Q    Median       3Q      Max
            -2.29237 -1.34322 .-0.03178..1.33792 .2.70763
Coefficients:
             Estimate  Std. Error  t value  Pr(>|t|)
(Intercept)   -8.4746      4.6641   -1.817    0.1026
Body.Mass      0.3623      0.1355    2.673    0.0255 *
---
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
Residual standard error: 1.776 on 9 degrees of freedom
Multiple R-squared: 0.4425, Adjusted R-squared: 0.3806
F-statistic: 7.144 on 1 and 9 DF, p-value: 0.02551
```

Request R to print the ANOVA table.

```
Anova(RegModel.1, type="II")
Anova Table (Type II tests)
Response: Matings
            Sum Sq   Df   F value    Pr(>F)
Body.Mass   22.528    1    7.1438   0.02551 *
Residuals   28.381    9
```

With a little rounding we have the following statistical model:

$$Y_i = -8.5 + 0.36 \cdot X_i$$

and in English, *Number of matings equals* `Body.mass` *multiplied by 0.36 then subtract 8.5;* the intercept was -8.5, the slope was 0.36.

Note:

The results of the regression analyses have been stored in the object called " `LinearModel.1` ". This is a nice feature of Rcmdr — it automatically provides an object name for you. Note that with each successive run of the linear model function via Rcmdr that it will change the object name by adding numbers successively. For example, after `LinearModel.1` the next

run of `lm()` in Rcmdr will automatically be called " `LinearModel.2` " and so on. In your own work you may specify the names of the objects directly or allow Rcmdr to do it for you, but do keep track of the object names!

From the R output we see that the estimate of the slope was +0.36, statistically different from zero ($p = 0.025$). The intercept was -8.5, but not statistically significant ($p = 0.103$), which means the intercept may be zero.

As a general rule, if you make an estimate of a parameter or coefficient, then you should provide a confidence interval in the following form:

$$\text{estimate} \pm \text{critical value} \times \textbf{standard error of the estimate}$$

Reminder:

Approximate 95% CI can be obtained by $\pm$ twice the standard error for the coefficient.

For confidence interval of regression slope we have a couple of options in R

Option 1.

```
confint(LinearModel.1, 'Body.Mass', level=0.95)

               2.5 %    97.5 %
Body.Mass 0.05565899 0.6689173
```

Option 2.

Goal: Extract the coefficients from the output of the linear model and calculate the approximate SE with nine degrees of freedom. This is the big advantage of saving output from functions as objects. Typically, much more information is about the results are available, and, additionally, can be retrieved for additional use. Extracting coefficients from the objects is the best option, but does come with a learning curve. Let's get started.

First, what information is available in the linear model output beyond the default information? To find out, use the `names()` function

```
names(LinearModel.1)
```

R output:

```
 [1] "coefficients"  "residuals"     "effects"       "rank"
 [5] "fitted.values" "assign"        "qr"            "df.residual"
 [9] "xlevels"       "call"          "terms"         "model"
```

Another way is to use the `summary()` function call.

```
summary(LinearModel.1)$coefficients
              Estimate Std. Error   t value   Pr(>|t|)
(Intercept) -8.4745763  4.6640856 -1.816986 0.10259256
Body.Mass    0.3622881  0.1355472  2.672781 0.02550595
```

How can we get just the standard error for the slope? Note that the estimates are reported in a 2×4 matrix like so:

1,1	1,2	1,3	1,4
2,1	2,2	2,3	2,4

Therefore, to get the standard error for the slope we identify that it is stored in cell `2,2` of the matrix and we write

```
summary(LinearModel.1)$coefficients[2,2]
```

which returns

```
[1] 0.1355472
```

Let's use this information to calculate confidence intervals:

```
slp=summary(LinearModel.1)$coefficients[2,1]
slpErrs=summary(LinearModel.1)$coefficients[2,2]
slp + c(-1,1)*slpErrs*qt(0.975, 9)
```

where `qt()` is the quantile function for the t distribution and "9" is the degrees of freedom from the regression. Results follow.

```
coef + c(-1,1)*errs*qt(0.975, 9)
[1] 0.05565899 0.66891728
```

And for the intercept

```
int=summary(LinearModel.1)$coefficients[1,1]
intErrs=summary(LinearModel.1)$coefficients[1,2]
int + c(-1,1)*intErrs*qt(0.975, 9)
```

Results

```
int + c(-1,1)*intErrs*qt(0.975, 9)
[1] -19.025471   2.076318
```

In conclusion, part of fitting a model includes reporting the estimates of the coefficients (model parameters). And, in general, when estimation is performed, reporting of suitable confidence intervals are expected.

Extract additional statistics from R's linear model function

The `summary()` function is used to report the general results from ANOVA and linear model function output in R software, but additional functions can be used to extract the rest of the output, e.g., coefficient of determination. To complete our example of extracting information from the `summary()` function, we next turn to `summary.lm()` function to see what is available.

At the R prompt type and submit

```
summary.lm(LinearModel.1)
```

This returns the following R output:

```
Call:
lm(formula = Matings ~ Body.Mass, data = bird_matings)

Residuals:
     Min       1Q   Median       3Q      Max
-2.29237 -1.34322 -0.03178  1.33792  2.70763
```

```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  -8.4746     4.6641  -1.817   0.1026
Body.Mass     0.3623     0.1355   2.673   0.0255 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.776 on 9 degrees of freedom
Multiple R-squared:  0.4425,    Adjusted R-squared:  0.3806
F-statistic: 7.144 on 1 and 9 DF,  p-value: 0.02551
```

Looks exactly like the output from `summary()` . Let's look at what is available in the `summary.lm()` function

```
names(summary.lm(LinearModel.1))
 [1] "call"          "terms"         "residuals"     "coefficients"
 [5] "aliased"       "sigma"         "df"            "r.squared"
 [9] "adj.r.squared" "fstatistic"    "cov.unscaled"
```

We see some information we got from `summary()` , e.g., "coefficients". If we interrogate the name coefficients like so

```
summary.lm(LinearModel.1)$coefficients
```

we get

```
summary.lm(LinearModel.1)$coefficients
              Estimate Std. Error   t value   Pr(>|t|)
(Intercept) -8.4745763  4.6640856 -1.816986 0.10259256
Body.Mass    0.3622881  0.1355472  2.672781 0.02550595
```

which, again, is a 2×4 matrix (see above)

So to get the standard error for the slope we identify that it is stored in cell `2,2` of the matrix and call it `LinearModel.1$coefficients[2,2]` .

Questions

pending

This page titled 17.3: Estimation of linear regression coefficient is shared under a CC BY-NC-SA 4.0 license and was authored, remixed, and/or curated by Michael R Dohm via source content that was edited to the style and standards of the LibreTexts platform.