Some linear models - part 2

Søren Højsgaard

2024-10-01

Contents

Packages used:

```
library(ggplot2)
library(broom)
library(dplyr)
```

1 Introduction

This note describes additional aspects of linear models. In particular we focus on the output from fitting a linear model with `lm` and summarizing the output with `tidy`.

2 The output from fitting a linear model

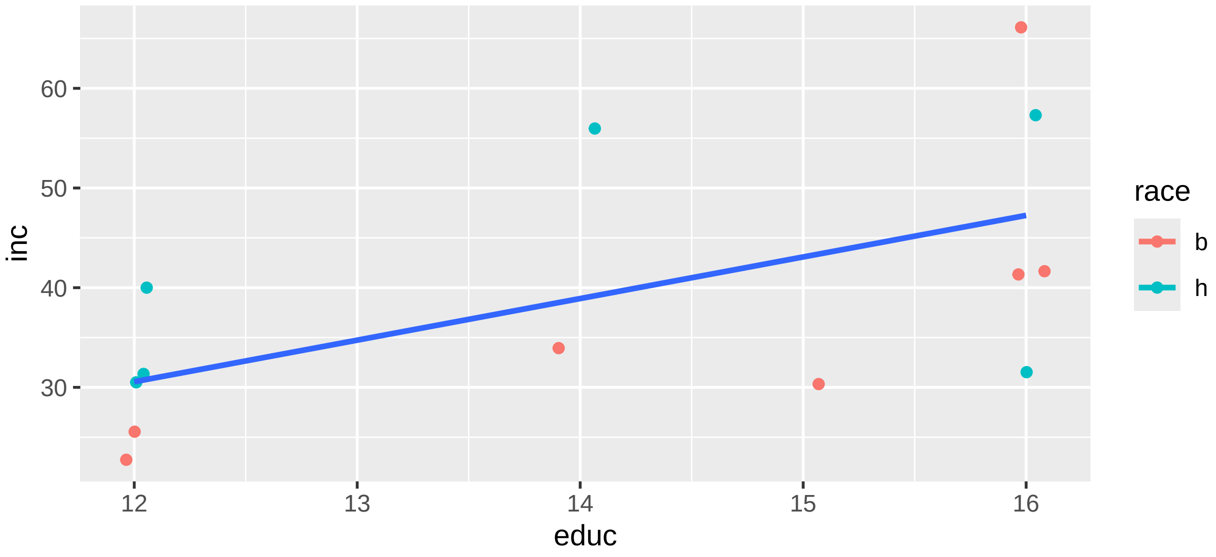
We consider the income data from the `doBy` package.

For didactical purposes we focus on people of hispanic and black origin and with 12 to 16 years of education. We will look at the relationship between income and education.

```
dat <- doBy::income |>
  filter(race != "w" & educ <= 16 & educ >=12)
dat
```

```
##    inc educ race
## 1   34   14    b
## 2   22   12    b
## 3   42   16    b
## 4   42   16    b
## 5   66   16    b
## 6   26   12    b
## 7   30   15    b
## 8   32   16    h
## 9   58   16    h
## 10  30   12    h
## 11  40   12    h
## 12  56   14    h
## 13  32   12    h
```

```
pl0 <- dat |> ggplot(aes(x=educ, y=inc, color=race)) + geom_jitter(width=0.1)
pl0 + geom_smooth(method="lm", aes(group=1), se=FALSE)
```



We shall look at the output from fitting a linear regression model to the data. The model is

```
lm_fit1 <- lm(inc ~ educ, data=dat)
tidy(lm_fit1)
```

```
## # A tibble: 2 x 5
##   term        estimate std.error statistic p.value
##   <chr>          <dbl>     <dbl>     <dbl>   <dbl>
## 1 (Intercept)   -19.5      25.3     -0.772  0.456
## 2 educ            4.17      1.78     2.34   0.0390
```

We shall go through the output from the `tidy` function.

2.1 The estimate column

```
tidy(lm_fit1)
```

```
## # A tibble: 2 x 5
##   term        estimate std.error statistic p.value
##   <chr>          <dbl>     <dbl>     <dbl>   <dbl>
## 1 (Intercept)   -19.5      25.3     -0.772  0.456
## 2 educ            4.17      1.78     2.34   0.0390
```

The estimate column are the least squares estimates of the coefficients. That is, if the model is $y_i = b_1 + b_2 x_i + e_i$, then the estimate of b_1 is `Intercept` and the estimate of b_2 is `educ`.

These estimates are found by minimizing the sum of squared residuals:

$$RSS = \sum_{i=1}^{n} (y_i - b_1 - b_2 x_i)^2$$

There exists a simple mathematical formula for the estimates, but it is not important to know it.

2.2 The `std.error` column

```r
tidy(lm_fit1)
```

```
## # A tibble: 2 x 5
##   term        estimate std.error statistic p.value
##   <chr>          <dbl>     <dbl>     <dbl>   <dbl>
## 1 (Intercept)   -19.5      25.3     -0.772  0.456
## 2 educ            4.17      1.78     2.34   0.0390
```

The `std.error` column is the standard error of the estimate. It is a measure of the uncertainty of the estimate (the smaller the standard error, the more certain we are about the estimate). Again, there is a simple mathematical formula for the standard error, but it is not important to know it.

It is more important to get the intuition that the standard error is a measure of the uncertainty of the estimate:

1. Conceptually, suppose we repeat the experiment many times, each time collecting a new sample of data and fitting the same model to the new data. This gives us a new estimate of the coefficients each time. The standard error is the standard deviation of the estimates.
2. We can not repeat the experiment many times, but we can simulate it by resampling data with replacement (so that some observations may not appear in the new sample and some may appear more than once).

```r
set.seed(2024) # for reproducibility
i <- sample(nrow(dat), replace = T)
dat_new <- dat[i,]
dat |> head()
```

```
##   inc educ race
## 1  34   14    b
## 2  22   12    b
## 3  42   16    b
## 4  42   16    b
## 5  66   16    b
## 6  26   12    b
```

```r
dat_new |> head()
```

```
##      inc educ race
## 2     22   12    b
## 5     66   16    b
## 13    32   12    h
## 12    56   14    h
## 1     34   14    b
## 13.1  32   12    h
```

```r
## Re-use model formula but fit model to new dataset.
mm_new <- update(lm_fit1, data=dat_new)
## Same as writing
## mm_new <- lm(inc ~ educ, data=dat_new)
coef(mm_new)
```

```
## (Intercept)        educ
##      -67.60        8.18
```

```r
coef(lm_fit1)
```

```
## (Intercept)        educ
##      -19.51        4.17
```

We do so many times and fit the model to the new data each time. This gives us a new estimate of the coefficients each time. The standard error is the standard deviation of the estimates.

```
set.seed(2024) # for reproducibility
parm <- replicate(999, {
    dat_new <- dat[sample(nrow(dat), replace = T),]
    mm_new <- update(lm_fit1, data=dat_new)
    coef(mm_new)
})

parm <- parm |> t()
parm |> head()
```

```
##      (Intercept) educ
## [1,]       -67.6 8.18
## [2,]       -32.9 5.28
## [3,]        16.5 1.80
## [4,]       -64.5 7.19
## [5,]       -40.0 5.75
## [6,]       -27.3 4.96
```

Now compare the std.error column with the standard deviation of the estimates. These numbers are not identical, but they are close. The standard error is a measure of the uncertainty of the estimate if the study is repeated many times under identical conditions.

```
tidy(lm_fit1)
```

```
## # A tibble: 2 x 5
##   term        estimate std.error statistic p.value
##   <chr>          <dbl>     <dbl>     <dbl>   <dbl>
## 1 (Intercept)   -19.5      25.3     -0.772  0.456
## 2 educ            4.17      1.78     2.34   0.0390
```

```
sd(parm[,1])
```

```
## [1] 21.7
```

```
sd(parm[,2])
```

```
## [1] 1.6
```

2.2.1 Histograms of the estimates

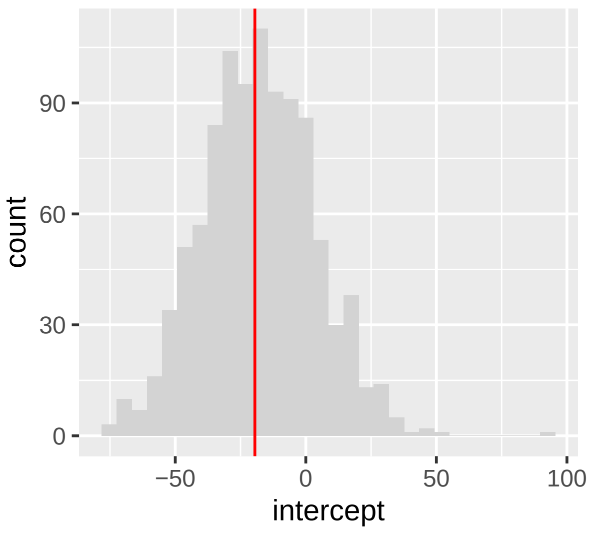
There is more to be said: Histograms of the estimates show how they vary when an experiment is repeated under and that they are close to being normally distributed.

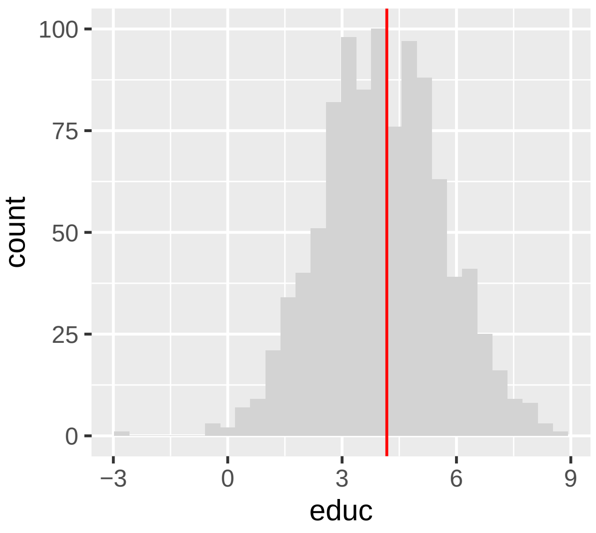
```
parm_df <- as.data.frame(parm) |>
  setNames(c("intercept", "educ"))

h1 <- parm_df |>
    ggplot(aes(x=intercept)) + geom_histogram(bins=30, fill="lightgray") +
    geom_vline(xintercept=coef(lm_fit1)[1], col="red")

h2 <- parm_df |>
    ggplot(aes(x=educ)) + geom_histogram(bins=30, fill="lightgray") +
    geom_vline(xintercept=coef(lm_fit1)[2], col="red")

library(patchwork)
h1 + h2
```





2.3 The `statistic` column

```
tidy(lm_fit1)
```

```
## # A tibble: 2 x 5
##   term        estimate std.error statistic p.value
##   <chr>          <dbl>     <dbl>     <dbl>   <dbl>
## 1 (Intercept)   -19.5      25.3     -0.772  0.456
## 2 educ            4.17      1.78     2.34   0.0390
```

This column is the estimate divided by its standard error. It is also called at t statistic. It is a measure of how many standard errors the estimate is from zero. The larger the statistic (numerically), the more certain we are that the estimate is different from zero.

2.4 The `p.value` column

```
tidy(lm_fit1)
```

```
## # A tibble: 2 x 5
##   term        estimate std.error statistic p.value
##   <chr>          <dbl>     <dbl>     <dbl>   <dbl>
## 1 (Intercept)   -19.5      25.3     -0.772  0.456
## 2 educ            4.17      1.78     2.34   0.0390
```

This column is the p-value of the estimate. It is the probability of observing a test statistic more extreme than the observed test statistic if the true value of the parameter is zero. The test statistic is in the statistic column.

This is a difficult concept, but it can be illustrated.

If there is no effect of the educ variable, we can reshuffle the response (`inc`) in the dataframe, because if the educ variable has no effect, it is just an unimportant label attached to the observations.

```
set.seed(2024)
dat_sim <- mutate(dat, inc=sample(inc))
dat_sim
```

```
##    inc educ race
## 1   22   14    b
## 2   66   12    b
## 3   34   16    b
## 4   58   16    b
## 5   32   16    b
## 6   30   12    b
## 7   56   15    b
## 8   42   16    h
## 9   32   16    h
## 10  30   12    h
## 11  42   12    h
## 12  26   14    h
## 13  40   12    h
```

We fit the same model to the reshuffled data and should (in most cases) get parameter estimates that are similar to those of the first fit.

```
lm_sim <- update(lm_fit1, data=dat_sim)
coef(lm_fit1)
```

```
## (Intercept)        educ
##      -19.51        4.17
```

```
coef(lm_sim)
```

```
## (Intercept)        educ
##     40.3421     -0.0789
```

We can repeat this many times and get a distribution of the estimates.

```
nsim <- 999
set.seed(101)

parm <- replicate(nsim, {
    ## Simulate data assuming no effect of educ:
    dat_sim <- mutate(dat, inc=sample(inc))
    lm_sim <- update(lm_fit1, data=dat_sim)
    coef(lm_sim)
})

parm <- t(parm)
parm |> head()
```

```
##      (Intercept)   educ
## [1,]        81.6 -3.011
## [2,]       102.9 -4.526
## [3,]        49.3 -0.714
## [4,]        29.3  0.703
## [5,]        71.3 -2.278
## [6,]        30.0  0.654
```

Now compute how often the estimates are larger in absolute value than the estimates from the original data.

```
(abs(parm[,1]) > abs(coef(lm_fit1)[1])) |> head()
```

```
## [1] TRUE TRUE TRUE TRUE TRUE TRUE
```

We can compute the proportion of times the estimates are larger in absolute value than the estimates from the original data.

```
tidy(lm_fit1)
```

```
## # A tibble: 2 x 5
##   term        estimate std.error statistic p.value
##   <chr>          <dbl>     <dbl>     <dbl>   <dbl>
## 1 (Intercept)   -19.5      25.3     -0.772  0.456
## 2 educ            4.17      1.78     2.34   0.0390
```

```r
mean(abs(parm[,1]) > abs(coef(lm_fit1)[1]))
```

```
## [1] 0.761
```

```r
mean(abs(parm[,2]) > abs(coef(lm_fit1)[2]))
```

```
## [1] 0.038
```

1. If the educ variable has an effect, the (absolute value) of the estimate from the original data should be larger than the (absolute value) of the estimates from the reshuffled data.
2. If the educ variable has no effect, the estimates should be similar.
3. The p-value is the proportion of times the estimates from the reshuffled data are more extreme than the estimates from the original data.

2.5 Relating the `statistic` and the `p.value` columns

There is a long tradition in statistics to use the p-value to decide if a parameter is important or not. The most common rule is to use a 5% level of significance. If the p-value is less than 0.05, then the parameter is considered important. If the p-value is larger than 0.05, then the parameter is considered unimportant.

The p-value is related to the statistic column, and the connection is as follows: If the statistic is between -1.96 and 1.96 (in practice between -2 and 2), then the p-value is larger than 0.05. If the statistic is outside this interval, then the p-value is less than 0.05.

2.6 Confidence intervals

```r
confint(lm_fit1)
```

```
##                2.5 % 97.5 %
## (Intercept) -75.125  36.10
## educ          0.253   8.09
```

```r
coef(lm_fit1)[1] + c(-1,1) * 1.96*sd(parm_df[,1])
```

```
## [1] -62  23
```

```r
coef(lm_fit1)[2] + c(-1,1) * 1.96*sd(parm_df[,2])
```

```
## [1] 1.04 7.31
```

The confidence intervals are constructed from the estimates and the standard errors. The 95% confidence interval is the interval that contains the true value of the parameter with probability 0.95 if the study is repeated many times under identical conditions.

For practical purposes, the confidence interval is a range of values that is likely to contain the true value of the parameter. If the confidence interval does not contain zero, then the parameter is considered important. If the confidence interval contains zero, then the parameter is considered unimportant.

2.7 Parametric bootstrap and `pbkrtest` - optional*

Notice here: Reshuffling the response mimics simulating data from the model with no effect of the `educ` variable. Such a simulation leads to an idea called parametric bootstrap which is implemented on the `pbkrtest` package.

```r
library(pbkrtest)
tidy(lm_fit1)
```

```
## # A tibble: 2 x 5
##   term        estimate std.error statistic p.value
##   <chr>          <dbl>     <dbl>     <dbl>   <dbl>
## 1 (Intercept)   -19.5      25.3     -0.772  0.456
## 2 educ            4.17      1.78     2.34   0.0390
```

```r
PBmodcomp(lm_fit1, .~. - educ)
```

```
## Bootstrap test; time: 0.40 sec; samples: 1000; extremes: 43;
## large : inc ~ educ
## inc ~ 1
##        stat df p.value
## LRT    5.26  1   0.022 *
## PBtest 5.26      0.044 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```