Week 4 Lab

Regression using OLS

Micaela Wood
01/27/2022

Today

- Learn to compute OLS Manually
- Learn to use `lm`
- Learn how to read output from `lm`

Manual OLS

First we will need some data

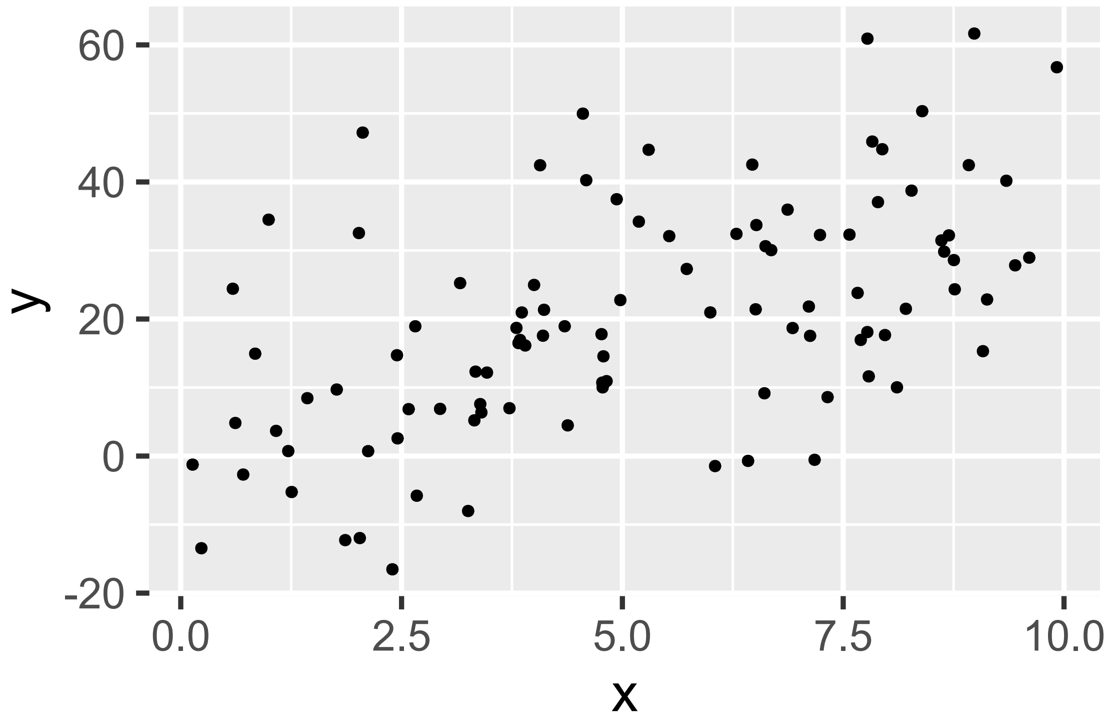
```
set.seed(1)

ols.data <- tibble(
  x = runif(100, 0, 10),
  e = rnorm(100, 0, 15),
  y = 5 + 3*x + e
)

head(ols.data)
```

```
#> # A tibble: 6 x 3
#>       x     e     y
#>   <dbl> <dbl> <dbl>
#> 1  2.66  5.97 18.9
#> 2  3.72 -9.18  6.98
#> 3  5.73  5.12 27.3
#> 4  9.08 -16.9  15.3
#> 5  2.02  21.5  32.5
#> 6  8.98  29.7  61.7
```

Manual OLS

A good place to start is with visualization.

```
ggplot(ols.data, aes(x = x, y = y))+
  geom_point()
```



Manual OLS

- Now suppose we did not create this data on our own.
- Typically we will not know how our data is generated so we use OLS to estimate the relationship

The equations we need to calculate our estimates are

$$\hat{\beta}_2 = \frac{\sum_{i=1}^{n}(Y_i - \bar{Y})(X_i - \bar{X})}{\sum_{i=1}^{n}(X_i - \bar{X})^2}$$

$$\hat{\beta}_1 = \bar{Y} - \hat{\beta}_2 * \bar{X}$$

Manual OLS

First we should solve for $\hat{\beta}_2$

```
#firt find the differences between the variable and its mean
ols.data <- ols.data %>% mutate(
  y.err = y - mean(y),
  x.err = x - mean(x)
)

#second you need to square the differences in the x's
ols.data <- ols.data %>% mutate(
  x.err.sq = x.err^2
)

#third we need the covariance of y and x
ols.data <- ols.data %>% mutate(
  cov.xy = y.err * x.err
)
```

Manual OLS

```
#now sum over the x squared error and x, y covariance
cov.sum <- sum(ols.data$cov.xy)
x.err.sq.sum <- sum(ols.data$x.err.sq)

#Finally divide to get b2
b2 <- cov.sum/x.err.sq.sum
b2
```

```
#> [1] 3.468515
```

Manual OLS

```
b2
```

```
#> [1] 3.468515
```

Notice that this isn't quite 3 but pretty close.

That is because we only have 100 observation but have a lot of variance.

If we increased the number of observations, the estimate would get closer to 3.

Manual OLS

Now lets calculate the intercept also known as $\hat{\beta}_1$.

Because we already have **b2** we just plug it in to the formula along with the means of **Y** and **X**.

```
b1 = mean(ols.data$y) - b2 * mean(ols.data$x)
b1
```

```
#> [1] 2.310118
```

Manual OLS

Now we can redo the scatterplot and add the regression line using `geom_abline`

```
ggplot(ols.data, aes(x = x, y = y))+
  geom_point()+
  geom_abline(slope = b2, intercept = b1)
```

Manual OLS

What if we want to include control variables?

Then this requires matrix algebra.

Another option is to use `lm`

OLS using lm

Let's start with an simple regression

```
lm1 <- lm(data = ols.data, y~x)
```

OLS using lm

- lm saves as a list of 12 different items.

- This allows you to call elements of the list in later code

```
lm1$coefficients
```

```
#> (Intercept)           x
#>    2.310118    3.468515
```

OLS using lm

- You can also view parts of the list all at once using `summary`

```
summary(lm1)
```

```
#>
#> Call:
#> lm(formula = y ~ x, data = ols.data)
#>
#> Residuals:
#>     Min      1Q  Median      3Q     Max
#> -27.747  -8.433  -1.306   7.864  37.749
#>
#> Coefficients:
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept)   2.3101     3.0872   0.748    0.456
#> x             3.4685     0.5302   6.542 2.77e-09 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 14.12 on 98 degrees of freedom
#> Multiple R-squared:  0.304,    Adjusted R-squared:  0.2969
```

OLS using lm

Now lets create a new data where y is explained by more than one variable.

```
n = 100
set.seed(1)

dgp_df <- tibble(
  e = rnorm(n, sd = 30),
  x = runif(n, min = 0, max = 10),
  z = runif(n, min = 5, max = 15),
  y = 5 + 3*x + 4*z + e
)

head(dgp_df)
```

```
#> # A tibble: 6 x 4
#>         e     x     z     y
#>     <dbl> <dbl> <dbl> <dbl>
#> 1 -18.8    2.68 11.7   41.2
#> 2   5.51   2.19  5.95  40.9
#> 3 -25.1    5.17  9.93  35.1
#> 4  47.9    2.69  9.62  99.4
#> 5   9.89   1.81  8.75  55.3
```

OLS using lm

First let's do a regression with just x and y

```
lm2a ← lm(data = dgp_df, y ~ x)
```

OLS using lm

```
summary(lm2a)
```

```
#> 
#> Call:
#> lm(formula = y ~ x, data = dgp_df)
#> 
#> Residuals:
#>     Min      1Q  Median      3Q     Max 
#> -70.106 -20.781   2.893  19.576  63.633 
#> 
#> Coefficients:
#>             Estimate Std. Error t value Pr(>|t|)    
#> (Intercept)   43.006      5.463   7.872 4.75e-12 ***
#> x              4.237      1.059   4.001 0.000123 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#> 
#> Residual standard error: 29.37 on 98 degrees of freedom
#> Multiple R-squared:  0.1404,	Adjusted R-squared:  0.1316 
#> F-statistic: 16.01 on 1 and 98 DF,  p-value: 0.0001225
```

OLS using lm

Now let's add the second variable and see how our results change.

```
lm2b ← lm(data = dgp_df, y ~ x + z)
```

OLS using lm

```
summary(lm2b)
```

```
#> 
#> Call:
#> lm(formula = y ~ x + z, data = dgp_df)
#> 
#> Residuals:
#>    Min     1Q Median     3Q    Max 
#> -73.83 -17.79  -0.14  17.32  68.06 
#> 
#> Coefficients:
#>             Estimate Std. Error t value Pr(>|t|)    
#> (Intercept)   4.7887    10.3545   0.462 0.644776    
#> x             3.8048     0.9835   3.869 0.000198 ***
#> z             3.9977     0.9458   4.227 5.36e-05 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#> 
#> Residual standard error: 27.13 on 97 degrees of freedom
#> Multiple R-squared:  0.2741,	Adjusted R-squared:  0.2591 
#> F-statistic: 18.31 on 2 and 97 DF,  p-value: 1.787e-07
```