205615894_stats101A_hw2

Takao

1/15/2022

1

```
# install.packages("readxl")
library(readxl)

houses_data <- read_excel("houses.xlsx", sheet = "Sheet1")
```

```
## New names:
## * `Sale Price/1000` -> `Sale Price/1000...1`
## * `Taxes (local, school, county)/1000` -> `Taxes (local, school, county)/1000...2`
## * `Sale Price/1000` -> `Sale Price/1000...3`
## * `Taxes (local, school, county)/1000` -> `Taxes (local, school, county)/1000...4`
```

```
head(houses_data)
```

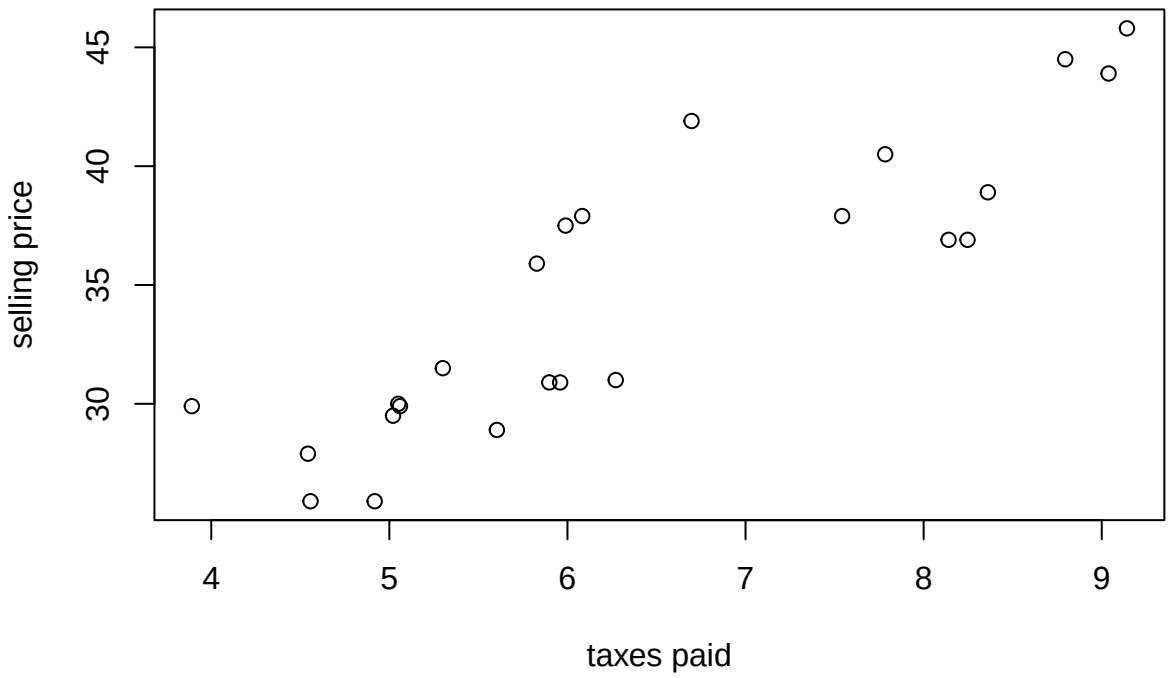
```
## # A tibble: 6 x 4
##   `Sale Price/1000...1` `Taxes (local, sch~ `Sale Price/1000~ `Taxes (local, sc~
##                   <dbl>               <dbl>             <dbl>              <dbl>
## 1                  25.9                4.92              30                 5.05
## 2                  29.5                5.02              36.9               8.25
## 3                  27.9                4.54              41.9               6.70
## 4                  25.9                4.56              40.5               7.78
## 5                  29.9                5.06              43.9               9.04
## 6                  29.9                3.89              37.5               5.99
```

```
dim(houses_data)
```

```
## [1] 12  4
```

(a)

```
houses_data <- data.frame(houses_data)
plot(c(houses_data[, 1], houses_data[, 3]) ~ c(houses_data[, 2], houses_data[, 4]), xlab = "taxes paid"
```




```
slr.model1 <- lm(c(houses_data[, 1], houses_data[, 3]) ~ c(houses_data[, 2], houses_data[, 4]))
slr.model1
```

```
##
## Call:
## lm(formula = c(houses_data[, 1], houses_data[, 3]) ~ c(houses_data[,
##     2], houses_data[, 4]))
##
## Coefficients:
##                             (Intercept)  c(houses_data[, 2], houses_data[, 4])
##                                  13.320                                 3.324
```

```
coef(slr.model1)
```

```
##                             (Intercept) c(houses_data[, 2], houses_data[, 4])
##                               13.320179                              3.324371
```

The least squares model for this given data set is predicted sales price = 13.321 + 3.324*(taxes paid).

```
e <- slr.model1$residuals
n <- length(e)
est.variance <- sum(e^2)/(n-2)
print(est.variance)
```

```
## [1] 8.767753
```

The estimate of the variance is 8.767753.

(b)

```
#Plug x = 7.50 into the simple linear regression model.
sellingprice7.5 <- 13.321 + 3.324*7.5
sellingprice7.5
```

```
## [1] 38.251
```

The mean selling price given that the taxes paid are x = 7.50 is 38.251.

(c)

```
#First we must calculate the predicted value of y corresponding to x = 5.898.
sellingprice5.898 <- 13.321 + 3.324*5.898
sellingprice5.898
```

```
## [1] 32.92595
```

The predicted value of y corresponding to x = 5.898 is equal to 32.925952.

```
houses_data[, 2] == 5.898
```

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

```
houses_data[, 4] == 5.898
```

```
##  [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

```
houses_data[7, 1]
```

```
## [1] 30.9
```

```
#30.9 is the observed selling price of where the x is 5.898

#To find the residual of the corresponding point, observed value - predicted value, thus
residual5.898 <- 30.9 - sellingprice5.898
residual5.898
```

```
## [1] -2.025952
```

The residual of the point where x = 5.898 is equal to -2.025952.

(d)

```r
alltaxespaid <- c(houses_data[, 2], houses_data[, 4])
alltaxespaid
```

```
##  [1] 4.9176 5.0208 4.5429 4.5573 5.0597 3.8910 5.8980 5.6039 5.8282 5.3003
## [11] 6.2712 5.9592 5.0500 8.2464 6.6969 7.7841 9.0384 5.9894 7.5422 8.7951
## [21] 6.0831 8.3607 8.1400 9.1416
```

```r
allsellingprice <- c(houses_data[, 1], houses_data[, 3])
allsellingprice
```

```
##  [1] 25.9 29.5 27.9 25.9 29.9 29.9 30.9 28.9 35.9 31.5 31.0 30.9 30.0 36.9 41.9
## [16] 40.5 43.9 37.5 37.9 44.5 37.9 38.9 36.9 45.8
```

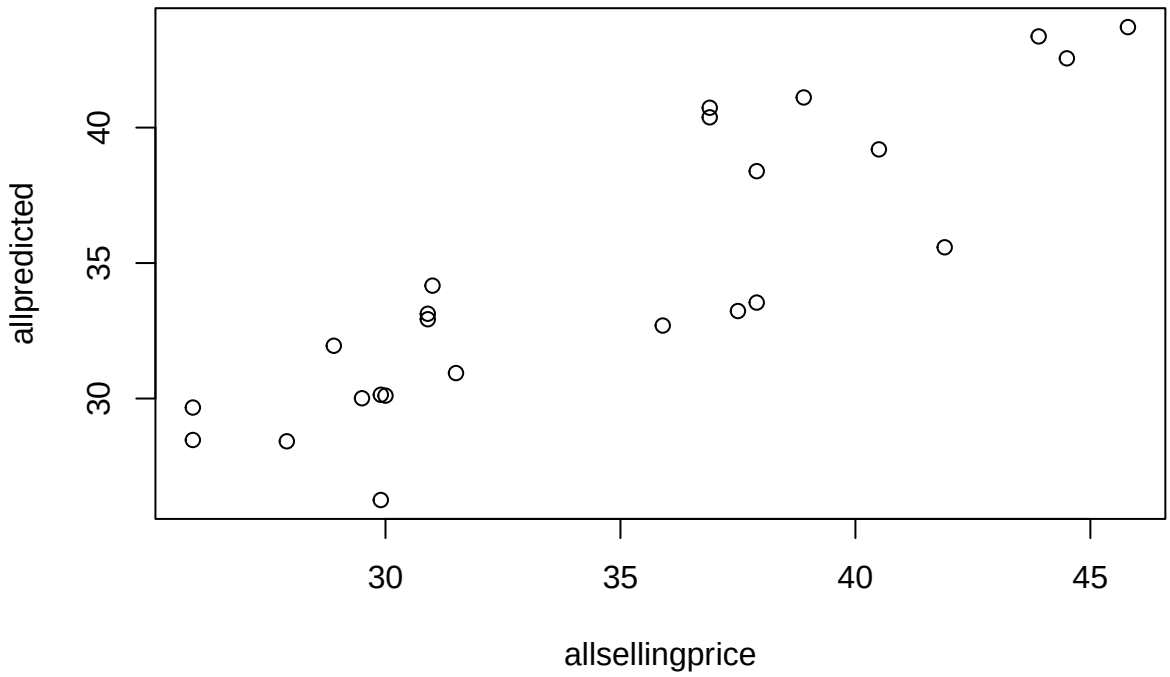
```r
allpredicted <- 13.321 + 3.324*alltaxespaid
allpredicted
```

```
##  [1] 29.66710 30.01014 28.42160 28.46947 30.13944 26.25468 32.92595 31.94836
##  [9] 32.69394 30.93920 34.16647 33.12938 30.10720 40.73203 35.58150 39.19535
## [17] 43.36464 33.22977 38.39127 42.55591 33.54122 41.11197 40.37836 43.70768
```

```r
observe_vs_predict <- cbind(allsellingprice, allpredicted, alltaxespaid)
observe_vs_predict
```

```
##       allsellingprice allpredicted alltaxespaid
##  [1,]            25.9     29.66710       4.9176
##  [2,]            29.5     30.01014       5.0208
##  [3,]            27.9     28.42160       4.5429
##  [4,]            25.9     28.46947       4.5573
##  [5,]            29.9     30.13944       5.0597
##  [6,]            29.9     26.25468       3.8910
##  [7,]            30.9     32.92595       5.8980
##  [8,]            28.9     31.94836       5.6039
##  [9,]            35.9     32.69394       5.8282
## [10,]            31.5     30.93920       5.3003
## [11,]            31.0     34.16647       6.2712
## [12,]            30.9     33.12938       5.9592
## [13,]            30.0     30.10720       5.0500
## [14,]            36.9     40.73203       8.2464
## [15,]            41.9     35.58150       6.6969
## [16,]            40.5     39.19535       7.7841
## [17,]            43.9     43.36464       9.0384
## [18,]            37.5     33.22977       5.9894
## [19,]            37.9     38.39127       7.5422
## [20,]            44.5     42.55591       8.7951
## [21,]            37.9     33.54122       6.0831
## [22,]            38.9     41.11197       8.3607
## [23,]            36.9     40.37836       8.1400
## [24,]            45.8     43.70768       9.1416
```

```
plot(allpredicted ~ allsellingprice)
```



If the relationship between y and x was a deterministic (no random error) straight line, all of the observed y or in this case observed selling price matches with the predicted y or predicted taxes paid. The points in the plot y predicted vs y observed are fairly in a straight line and there is fairly small variation around this arbitrary line. Thus, it can be stated and concluded that the tax paid is an effective regressor variable in predicting the selling price.

Problem 2

```
indicators_data <- read_excel("indicators.xlsx", sheet = "Sheet1")

head(indicators_data)
```

```
## # A tibble: 6 x 3
##   MetroArea PriceChange LoanPaymentsOverdue
##   <chr>           <dbl>               <dbl>
## 1 Atlanta           1.2                4.55
## 2 Boston           -3.4                3.31
## 3 Chicago          -0.9                2.99
## 4 Dallas            0.8                4.26
## 5 Denver           -0.7                3.56
## 6 Detroit          -9.7                4.71
```

```r
dim(indicators_data)
```

```
## [1] 18  3
```

(a)

```r
#  Y = Percentage change in average price from July 2006 to July 2007 (based on
# the S&P/Case-Shiller national housing index); and
#  x = Percentage of mortgage loans 30 days or more overdue in latest quarter
# (based on data from Equifax and Moody's).

slr.model2 <- lm(PriceChange ~ LoanPaymentsOverdue, data = indicators_data)
slr.model2
```

```
##
## Call:
## lm(formula = PriceChange ~ LoanPaymentsOverdue, data = indicators_data)
##
## Coefficients:
##         (Intercept)  LoanPaymentsOverdue
##               4.514               -2.249
```

```r
confint(slr.model2, level = 0.95)
```

```
##                           2.5 %     97.5 %
## (Intercept)         -2.532112 11.5611000
## LoanPaymentsOverdue -4.163454 -0.3335853
```

If, in repeated random sampling, we construct a large number of confident intervals, 95% of those intervals will contain the true parameter value β_1. This is because confidence interval are random variables so it may be that this confidence interval that was generated does not include the true parameter value, but this "idea" of repeating the same process over and over again works.

In practice, we can say that we are 95% confident that the interval (-4.163, -0.334) contains the true value of the slope β_1.

The constructed 95% confidence interval does not contain the value 0 and the entire interval is negative. Thus, there is sufficient evidence that there is a significant negative linear association of the percentage of mortgage loans 30 days or more overdue in latest quarter and the percentage change in average price from July 2006 to July 2007.

(b)

```r
data.for.prediction <- data.frame(LoanPaymentsOverdue = 4)
predict(slr.model2, data.for.prediction, interval = "confidence")
```

```
##         fit       lwr       upr
## 1 -4.479585 -6.648849 -2.310322
```

We are 95% confident that the interval between (-6.649, -2.310) contains the true expected value of the percentage change in average price from July 2006 to July 2007 given that the percentage of mortgage loans 30 days or more in latest quarter is 4 percent.

0% is not a feasible value for E(Y|X = 4) because 0% is outside of the 95 percent confidence interval that we constructed.

Problem 3

(a)

```
#degree of freedom = n - 2 = 30 - 2 = 28
# critical value from the table using df = 28 and confidence interval = 95%, critical value = 2.0481
# since we are interested in finding the 95 percent confidence interval for the start-up time, beta0, w
# intercept estimate = 0.6417099
# intercept std. error = 0.1222707

# beta0 +- std. error*critical value
# intercept estimate +- intercept std. error*critical value

higher <- 0.6417099 + 0.1222707*2.0481
higher
```

```
## [1] 0.8921325
```

```
lower <- 0.6417099 - 0.1222707*2.0481
lower
```

```
## [1] 0.3912873
```

```
c(lower, higher)
```

```
## [1] 0.3912873 0.8921325
```

We are 95% confident that the interval between (0.391, 0.892) contains the true start-up time.

(b)

Null hypothesis: beta1 = 0.01 (The true processing time for an additional invoice is 0.01 hours) Alternative hypothesis: beta1 != 0.01 (The true processing time for an additional invoice is not 0.01 hours)

Since we are interesting in performing a test for beta1, we will be looking at the invoice estimate and invoice std. error.

Invoice estimate = 0.0112916 Invoice std. error = 0.0008184

We must obtain the test statistic

t = (predicted beta1 - observed beta1) / standard error of predicted beta1

observed beta1 = 0.01

t statistic = (0.0112916 - 0.01) / 0.0008184

```
(0.0112916 - 0.01) / 0.0008184
```

```
## [1] 1.578201
```

t statistic = 1.578201

To reject null hypothesis, the test statistic must be greater than critical value, thus to find critical value,

```
qt(1 - 0.025, 28)
```

```
## [1] 2.048407
```

Since the test statistic is not greater than the critical value, we fail to reject the null hypothesis that the average processing time for an aditional invoice is 0.01 hours.

(c)

n or the sample size is 28 + 2 = 30 Using the table of critical values of t distribution using degree of freedom 28 and a CI of 95%, we get a critical value of 2.0481. Residual standard error is 0.3298.

First, we get the expected time plugging in 130 into x

```
expected130 <- 0.6417099 + 0.0112916*130
expected130
```

```
## [1] 2.109618
```

Now to find the confidence interval, we must add and subtract the margin of error to this expected time to process 130 invoices Plug numbers into margin of error formula ((x-xbar)^2 = 0 so this component is all 0).

```
margin_of_error <- qt(1 - 0.025, 28) * sqrt(1 + (1/30))*0.3298
margin_of_error
```

```
## [1] 0.6867318
```

```
high <- expected130 + margin_of_error
low <- expected130 - margin_of_error
c(low, high)
```

```
## [1] 1.422886 2.796350
```

We are 95% confident that the interval between (1.423, 2.796) contains the true time taken to process 130 invoices.