Simple Linear Regression using R

Pramudita Satria Palar, Vani Virdyawan, Ferryanto

4/24/2022

Basic Theory

Model regresi linear

Linear regression is one of the most used regression models with many applications in engineering. A regression model is used to predict the value of Y as a function of one or more variables X. A predictor with only one variable is called *Simple linear regression.*

In simple linear regression, we define Y as a summation of a linear function with a random error ε:

$$Y = \beta_0 + \beta_1 x + \varepsilon$$

where β_0 is the *intercept* and β_1 is the *slope.* As the name suggest, the slope β_1 is the gradient of the linear regression line that will be found, whereas β_0 is the intersection between the line and the y axis. We call β_0 and β_1 the regression coefficients. The term regression coefficients are important when discussing *multiple linear regression* or nonlinear regression such as *polynomial regression.*

Since ε is a random number, by assuming that $\mathbb{E}(\varepsilon) = 0$, we can then derive

$$\mathbb{E}(Y|x) = \mu_{Y|x} = \beta_0 + \beta_1 x, \tag{1}$$

The equation above is the linear regression model that we usually use in practice. This model can be interpreted as the expected value of Y given x. In the linear regression, we also assume that the error is normally distributed with the mean equals to 0 and the variance equals to σ^2; In other words, $\varepsilon \sim \mathcal{N}(0, \sigma^2)$, or $\mathbb{E}(\varepsilon) = 0$, and $\mathbb{V}(\varepsilon) = \sigma^2$. This assumption enables us to calculate the standard error, perform hypothesis testing, and obtain other important parameters, such as *prediction interval.* By observing that

$$\mathbb{E}(Y|x) = \mathbb{E}(\beta_0 + \beta_1 x + \varepsilon) = \beta_0 + \beta_1 x + \mathbb{E}(\varepsilon) = \beta_0 + \beta_1 x, \tag{2}$$

and

$$\mathbb{V}(Y|x) = \mathbb{V}(\beta_0 + \beta_1 x + \varepsilon) = \mathbb{V}(\beta_0 + \beta_1 x) + \mathbb{V}(\varepsilon) = 0 + \sigma^2 = \sigma^2. \tag{3}$$

It is clear that the regression model is the line of the mean values with variability defined by the *error variance* σ^2.

Calculating Regression Coefficients

To build a regression model, we need data consisting of n pairs of observations: $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$. We can then write the relation between x and y in each observation as follows:

$$y_i = \beta_0 + \beta_1 x_i + \varepsilon_i, \tag{4}$$

Where i is the i-th data. We can then define the residual (e) as the difference between the observations and the linear model predictions. For the i-th data, we can then write

$$e_i = y_i - \hat{y}_i, \tag{5}$$

where $\hat{y}_i$ is the prediction of the linear regression model for the i-th data. The residual shows how close the prediction with the actual data. Therefore, the best regression model is the model that has the smallest difference between the prediction and the actual observation data. Since we have n observation data, our regression model should be as close as possible to these n data, which can be defined as

$$\varepsilon_{RSS} = \sum_{i=1}^{n} e_i^2, \tag{6}$$

where RSS is the *residual sum of squares* (RSS). The error term in the equation above is squared due to the fact that the error value can be either negative or positive. The residual should be minimised with respect to the error regardless of the sign of the error.

To obtain the best regression model, we need to choose coefficients β_0 and β_1 that give the smallest RSS value. By remembering that $e_i = y_i - \beta_0 - \beta_1 x_1$, we then have

$$\varepsilon_{RSS} = \sum_{i=1}^{n} (y_i - \beta_0 - \beta_1 x_i)^2, \tag{7}$$

In the language of calculus, the best β_0 dan β_1 are found so that the following partial differentiations equal to zero:

$$\left.\frac{\partial \varepsilon_{RSS}}{\partial \beta_0}\right|_{\hat{\beta}_0, \hat{\beta}_1} = 0$$

$$\left.\frac{\partial \varepsilon_{RSS}}{\partial \beta_1}\right|_{\hat{\beta}_0, \hat{\beta}_1} = 0$$

In other words, the values of β_0 dan β_1 which make the above partial differentiations equal to zero, are the solution of the optimisation problem to find the smallest value of the RSS. Based on a simple calculus, it is easily shown that

$$\left.\frac{\partial \varepsilon_{RSS}}{\partial \beta_0}\right|_{\hat{\beta}_0, \hat{\beta}_1} = -2 \sum_{i=1}^{n} (y_i - \beta_0 - \beta_1 x_i) = 0 \tag{8}$$

and

$$\left.\frac{\partial \varepsilon_{RSS}}{\partial \beta_1}\right|_{\hat{\beta}_0, \hat{\beta}_1} = -2 \sum_{i=1}^{n} (y_i - \beta_0 - \beta_1 x_i) x_i = 0. \tag{9}$$

The above equations can be simplified to

$$n\hat{\beta}_0 + \hat{\beta}_1 \sum_{i=1}^{n} x_i = \sum_{i=1}^{n} y_i \tag{10}$$

and

$$\hat{\beta}_0 \sum_{i=1}^{n} x_i + \hat{\beta}_1 \sum_{i=1}^{n} x_i^2 = \sum_{i=1}^{n} y_i x_i, \tag{11}$$

We call the equations above **least squares normal equations**. The solutions of the above equations are

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x} \tag{12}$$

$$\hat{\beta}_1 = \frac{S_{xx}}{S_{xy}} \frac{\sum_{i=1}^{n} y_i x_i - \frac{(\sum_{i=1}^{n} y_i)(\sum_{i=1}^{n} x_i)}{n}}{\sum_{i=1}^{n} x_i^2 - \frac{(\sum_{i=1}^{n} x_i)^2}{n}}, \tag{13}$$

where $\bar{y} = (1/n)\sum_{i=1}^{n} y_i$ dan $\bar{x} = (1/n)\sum_{i=1}^{n} x_i$,

After obtaining the coefficients, we can then determine other important parameters, such as R^2, *prediction interval*, *confidence interval*, and *hypothesis testing* of the coefficients. It is easy to get these parameters using the R library.

Application to Simple Data set

The figure below shows the data that we will use in this tutorial. As can be seen in the figure, there is a linear trend between the level of hydrocarbon and oxygen purity. Our task is to define the best linear regression line to approximate the data. After getting the model, we need to calculate the uncertainty from the linear regression model we obtained.

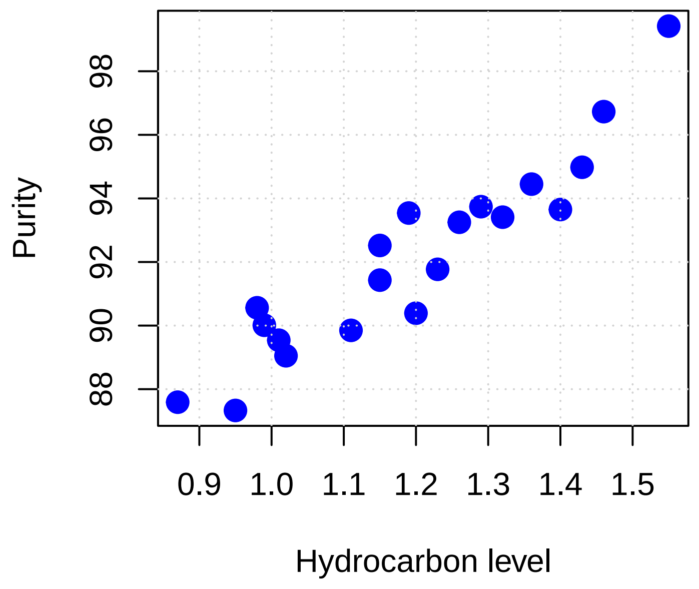

Figure 1: Visualization of the Hydrocarbon level-purity data set

Defining the data set

We start with defining the data. We define X as a variable `X` and Y as a variable `Y` in R. In this tutorial, X and Y are defined as the vector of numerical data. Later on, we will use the data frame format to process the data, which is easier to use. We can then start with executing the following line of code:

```
X <- c(0.99, 1.02, 1.15, 1.29, 1.46, 1.36, 0.87, 1.23, 1.55, 1.4, 1.19, 1.15, 0.98,
       1.01, 1.11, 1.2, 1.26, 1.32, 1.43, 0.95)
Y <- c(90.01, 89.05, 91.43, 93.74, 96.73, 94.45, 87.59, 91.77, 99.42, 93.65,
       93.54, 92.52, 90.56, 89.54, 89.85, 90.39, 93.25, 93.41, 94.98, 87.33)
```

It should be noted that `c()` is how we define a vector or a list containing numerical data (notice that `c()` can also be used to create a list containing non-numerical data).

Based on the above data and by using analytical equations to calculate the regression coefficients, we obtained the following linear regression model:

$$Y = 74.28 + 14.95X.$$

We can obtain the model either by using the built-in function from R or by writing analytical equations manually. We start by calculating the coefficients with the equations that we write manually in R:

```
n <- length(Y) # Size of data

S_xy <- (sum(X*Y)-((sum(Y)*sum(X))/n)) # Calculating S_xy
S_xx <- (sum(X^2)-(sum(X)^2)/n) # Calculating S_xx

# Calculate beta_1 and beta_0
beta_1 <- S_xy/S_xx # Beta (slope)
beta_0 <- mean(Y)-beta_1*mean(X) # Beta 0 (intercept)

cat("The value of beta_0 is ", beta_0, "\n")
```

```
## The value of beta_0 is  74.28331
```

```
cat("The value of beta_1 is ", beta_1, "\n")
```

```
## The value of beta_1 is  14.94748
```

The calculation is easy, and we can also get the correct coefficient values. However, to get other parameters (e.g. confidence and prediction interval), you have to write the code manually as well. Fortunately, R already has special modules to calculate a linear regression model called `lm()`.

Syntax for building a linear regression model

To obtain the linear regression model, we will use the built-in function from R. The main function that we use for one dimension linear regression model is `lm()` with the following syntax:

```
linregmod <- lm(Y~X) # Create a regression model named `linregmod`
```

Using the above line of codes, we create a linear regression model called ‘`linregmod` using `lm()`. Using the syntax, we will generate a model $Y = \beta_0 + \beta_1 x$ (`Y~X`) (the intercept will be automatically obtained without having to be manually programmed).

We can then check the generated linear regression model. We can get the coefficient's values by using the following syntax:

```
print(linregmod) # Cetak koefisien dan juga formula dari model regresi linear
```

```
##
## Call:
## lm(formula = Y ~ X)
##
## Coefficients:
## (Intercept)            X
##       74.28        14.95
```

Hypothesis Testing, Standard Error, and Coefficient of Determination

After obtaining the linear regression model, we want to know the values of the other parameters, such as *standard error* and hypothesis testing. Another parameter that we may be interested in are the *coefficient of determination* (R^2) and also the Adjusted-R^2. We can use a built-in function in R `summary()` to your model to obtain this information. You can try executing the following code:

```
summary(linregmod) # Cetak informasi lain dari model regresi linear
```

```
##
## Call:
## lm(formula = Y ~ X)
##
```

```
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.83029 -0.73334  0.04497  0.69969  1.96809
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   74.283      1.593   46.62  < 2e-16 ***
## X             14.947      1.317   11.35 1.23e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.087 on 18 degrees of freedom
## Multiple R-squared:  0.8774, Adjusted R-squared:  0.8706
## F-statistic: 128.9 on 1 and 18 DF,  p-value: 1.227e-09
```

With `summary()`, we can get the following information: (1) Linear regression formula, (2) residual, (3) coefficients and their standard error, *t-value*, and *p-value* from hypothesis testing (4) coefficient of determination, and (5) the results of ANOVA. There is also information about the level of significance of the hypothesis testing that is pointed with `Signif. codes`. As an example, if the code shows `*` in one of the coefficients, it means that the p-value is lower than 0.05, which means that the null hypothesis is rejected with a 5 % level of significance.

The residual is the difference between the data and the prediction from the linear regression model. The model rejects the null hypothesis of $\beta_0 = 0$ and $\beta_1 = 0$. It means that there is indeed a correlation between the level of hydrocarbon with oxygen purity. In this case, null hypothesis $\beta_0 = 0$ does not have a significant meaning. The standard error can be interpreted as the uncertainty of the β_0 and β_1 values. It means that the slope can be steeper or less steep, and the point of interception can be either higher or lower. The R^2 value is high, meaning that the linear model can be used for the data.

Using Data Frame format

Data Frame Definition

Sometimes we want to manage our data in a table or matrix. For this purpose, we use data frame format by using a function called `data.frame()`:

```
DATA = data.frame(
  Hydrocarbon = c(0.99, 1.02, 1.15, 1.29, 1.46, 1.36, 0.87, 1.23, 1.55, 1.4, 1.19, 1.15, 0.98,
       1.01, 1.11, 1.2, 1.26, 1.32, 1.43, 0.95),
  Purity = c(90.01, 89.05, 91.43, 93.74, 96.73, 94.45, 87.59, 91.77, 99.42, 93.65,
       93.54, 92.52, 90.56, 89.54, 89.85, 90.39, 93.25, 93.41, 94.98, 87.33)
)
```

The content of the variable `DATA` can be inspected by calling the variable name (`DATA`) in the R Studio console:

```
DATA
```

```
##    Hydrocarbon Purity
## 1         0.99  90.01
## 2         1.02  89.05
## 3         1.15  91.43
## 4         1.29  93.74
## 5         1.46  96.73
## 6         1.36  94.45
## 7         0.87  87.59
## 8         1.23  91.77
## 9         1.55  99.42
## 10         1.40  93.65
## 11         1.19  93.54
## 12         1.15  92.52
## 13         0.98  90.56
## 14         1.01  89.54
## 15         1.11  89.85
## 16         1.20  90.39
## 17         1.26  93.25
## 18         1.32  93.41
## 19         1.43  94.98
## 20         0.95  87.33
```

To get the data for a particular column, we can use the symbol `$` followed by the variable's name in the data frame that you want to get. For example, if we want to see the content of the Hydrocarbon variable, you can type `DATA$Hydrocarbon`, as can be seen in the following examples:

```
DATA$Hydrocarbon # Display Hydrocarbon data
```

```
##  [1] 0.99 1.02 1.15 1.29 1.46 1.36 0.87 1.23 1.55 1.40 1.19 1.15 0.98 1.01 1.11
## [16] 1.20 1.26 1.32 1.43 0.95
```

```
DATA$Purity # Display purity data
```

```
##  [1] 90.01 89.05 91.43 93.74 96.73 94.45 87.59 91.77 99.42 93.65 93.54 92.52
## [13] 90.56 89.54 89.85 90.39 93.25 93.41 94.98 87.33
```

Generating a linear regression model using a data frame

After defining the data in the data frame format, you can then specify the variable `data=DATA` as a variable of the function `lm()`. It means that you will use the variable `DATA` to be processed fby the linear regression model. The next step is to define the formula of the linear model that we want to generate:

$$\text{Purity} = \beta_0 + \beta_1 \times \text{ Hydrocarbon}.$$

We define the new model called `linregmod2`, which is similar to the previous model, `linregmod`:

```
linregmod2 = lm(data=DATA, formula=Purity~Hydrocarbon)
```

If you print this model, you will get:

```
print(linregmod2)
```

```
##
## Call:
## lm(formula = Purity ~ Hydrocarbon, data = DATA)
##
## Coefficients:
## (Intercept)  Hydrocarbon
##       74.28        14.95
```

As before, we can also print all of the information of the model using `summary()`:

```
summary(linregmod2)
```

```
##
## Call:
## lm(formula = Purity ~ Hydrocarbon, data = DATA)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.83029 -0.73334  0.04497  0.69969  1.96809
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   74.283      1.593   46.62  < 2e-16 ***
## Hydrocarbon   14.947      1.317   11.35 1.23e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.087 on 18 degrees of freedom
## Multiple R-squared:  0.8774, Adjusted R-squared:  0.8706
## F-statistic: 128.9 on 1 and 18 DF,  p-value: 1.227e-09
```

Plot a linear regression model

Simple plot

To visualise the linear regression model that we created, we can use `plot()`. There are many ways to visualise the data and the model using `plot()` (e.g. you can use a specific library, such as `ggplot`). In this tutorial, we will use the simplest method with a built-in function from R.

To perform the prediction, we use `predict()`. The function `predict()` requires a linear regression model that we created. The prediction results will be stored as a new variable in the data frame format. In the following example, we will plot the linear regression line together with the data in a Hydrocarbon range of 0.7 – 1.9:

```
xnew = data.frame(Hydrocarbon = seq(0.7,1.9,0.05))
ynew = predict(linregmod2,xnew)
plot(DATA$Hydrocarbon,DATA$Purity,pch=19,col="blue",cex=1.5,xlab = "Hydrocarbon level (x)",
     ylab="Purity (y)",xlim=c(0.7, 1.9))
lines(xnew$Hydrocarbon, ynew, col="red",type = "l",lwd = 2)
grid()
```

In the above lines of code, `xnew` is the variable where we store hydrocarbon values at which the oxygen purity will be predicted. The prediction results will then be stored in `ynew`. To create the plot, we use function `plot()`, `lines()` is the function to add the linear regression line, and `grid()` is a function to show the grid on the plot. To generate an incremental sequence from 0.7 to 1.9 with a 0.05 increment, we use `seq()`.

Calculation of the confidence interval for the coefficients

Calculation of the confidence interval for the coefficients can be easily done by using the function "confint()"'. To take one example, for 95% confidence level then:

```
confint(linregmod2,level=0.95)
```

```
##                2.5 %   97.5 %
## (Intercept) 70.93555 77.63108
## Hydrocarbon 12.18107 17.71389
```

Prediction and Confidence Interval Plot

As discussed in the previous subsections, we do not exactly know the true linear line of the data. Therefore, we also want to see the confidence interval of our linear regression model. The confidence interval can be calculated from the standard error of β_0 and β_1. These standard errors can then be used to calculate the confidence interval. We can also plot the linear regression line and the confidence interval in the same plot. To add more information about the confidence and prediction intervals, we can use an additional argument in

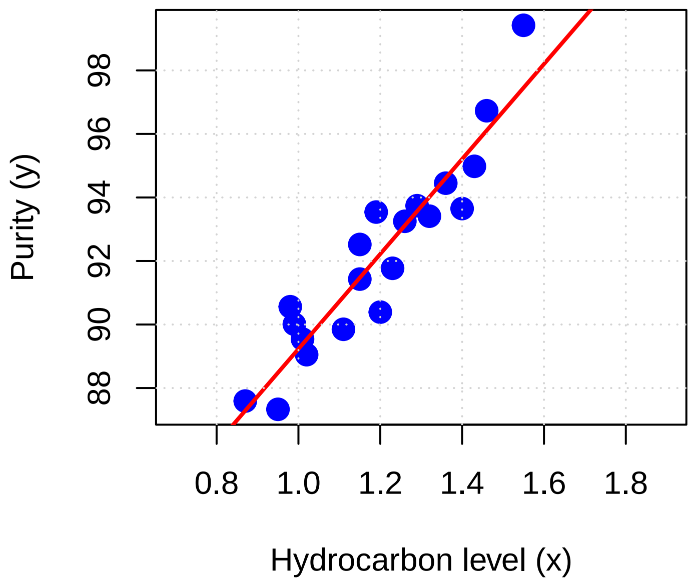

Figure 2: Visualization of the linear regression model that we just created

predict(), which is `interval=`. The value of the `interval=` argument`can be either`confidence`or`prediction'. The following example shows the method to show the confidence interval:

```
ynew = predict(linregmod2,xnew,interval="confidence")
print(ynew)
```

```
##         fit      lwr       upr
## 1  84.74655 83.28255  86.21055
## 2  85.49392 84.15869  86.82916
## 3  86.24130 85.03272  87.44987
## 4  86.98867 85.90390  88.07344
## 5  87.73605 86.77113  88.70096
## 6  88.48342 87.63273  89.33411
## 7  89.23079 88.48612  89.97547
## 8  89.97817 89.32727  90.62907
## 9  90.72554 90.15016  91.30093
## 10 91.47292 90.94686  91.99897
## 11 92.22029 91.70974  92.73084
## 12 92.96766 92.43582  93.49951
## 13 93.71504 93.12911  94.30097
## 14 94.46241 93.79755  95.12727
## 15 95.20979 94.44885  95.97072
## 16 95.95716 95.08867  96.82565
## 17 96.70453 95.72077  97.68830
## 18 97.45191 96.34756  98.55626
## 19 98.19928 96.97061  99.42795
## 20 98.94666 97.59095 100.30236
## 21  99.69403  98.20927 101.17879
## 22 100.44140  98.82605 102.05676
## 23 101.18878  99.44164 102.93591
## 24 101.93615 100.05630 103.81601
## 25 102.68353 100.67020 104.69685
```

As can be seen, in addition to the prediction, we have two additional columns: `lwr` and `upr`, which are the lower and upper limit of the confidence interval, respectively. To access the values of the `lwr` and `upr` you can call it using `ynew[,2]` and `ynew[,3]`.

The following lines of code are used to make a plot of the linear regression line with its confidence interval:

```
xnew = data.frame(Hydrocarbon = seq(0.7,1.9,0.05))
ynew = predict(linregmod2,xnew,interval="confidence")
plot(xnew$Hydrocarbon,ynew[,1],type = "l",col = "red",lwd=2,xlab = "Hydrocarbon level (x)"
     ,ylab="Purity (y)",xlim=c(0.7, 1.9), ylim = c(88,98))
lines(xnew$Hydrocarbon,ynew[,2],col="black",lty = 2)
lines(xnew$Hydrocarbon,ynew[,3],col="black",lty = 2)
par(new=TRUE)
plot(DATA$Hydrocarbon,DATA$Purity,pch=19,col="blue",cex=1.5,xlab = "Hydrocarbon level (x)",
     ylab="Purity (y)",xlim=c(0.7, 1.9), ylim = c(88,98))
legend(0.7,98.2,legend=c("Regression line","Confidence interval"),col=c("red","black")
       ,lty = c(1,2), pt.cex=1, cex=0.75)
grid()
```

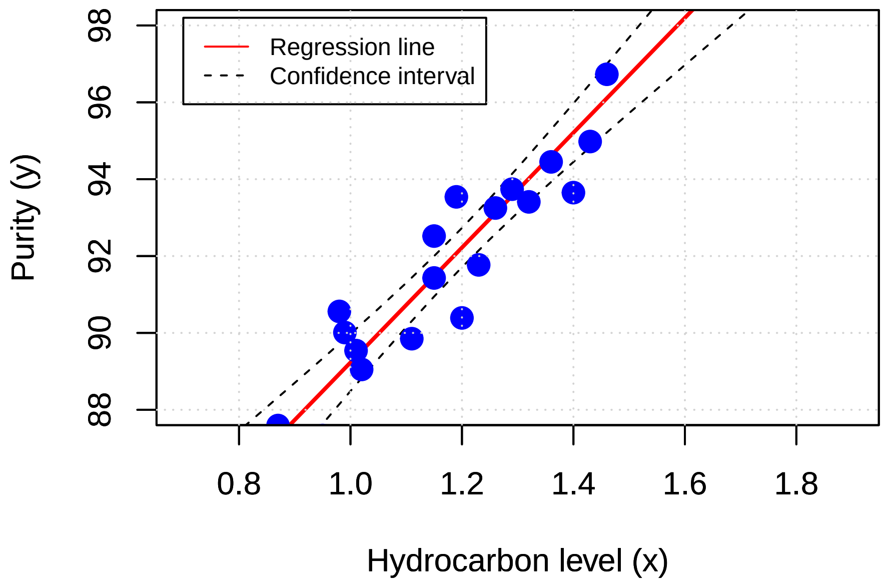

Figure 3: Visualization of the linear regression model with the confidence interval

You can see that the true regression model can be more or less steep, with 95% *confidence interval* you can see the range of possibilities where the true line might be located.

In addition to showing the confidence interval, we can also visualise the prediction interval. The prediction interval gives information about the uncertainty of our prediction based on the model that we have. The prediction can either be higher or lower than the actual data. It means we cannot be fully sure of the prediction that we have. Therefore, it is important for us to know how uncertain our prediction is. The uncertainty of the prediction interval comes from two sources: the uncertainty of the linear regression model and the error variance. You can execute the following lines of code to plot the prediction interval:

```
ynew = predict(linregmod2,xnew,interval="prediction")
plot(xnew$Hydrocarbon,ynew[,1],type = "l",col = "red",lwd=2,xlab = "Hydrocarbon level (x)"
     ,ylab="Purity (y)",xlim=c(0.7, 1.9), ylim = c(88,98))
lines(xnew$Hydrocarbon,ynew[,2],col="black",lty = 2)
lines(xnew$Hydrocarbon,ynew[,3],col="black",lty = 2)
par(new=TRUE)
plot(DATA$Hydrocarbon,DATA$Purity,pch=19,col="blue",cex=1.5,xlab = "Hydrocarbon level (x)",
     ylab="Purity (y)", xlim=c(0.7, 1.9), ylim = c(88,98))
legend(0.7,98,legend=c("Regression line","Prediction interval"),col=c("red","black")
       ,lty = c(1,2), pt.cex=1, cex=0.75)
grid()
```

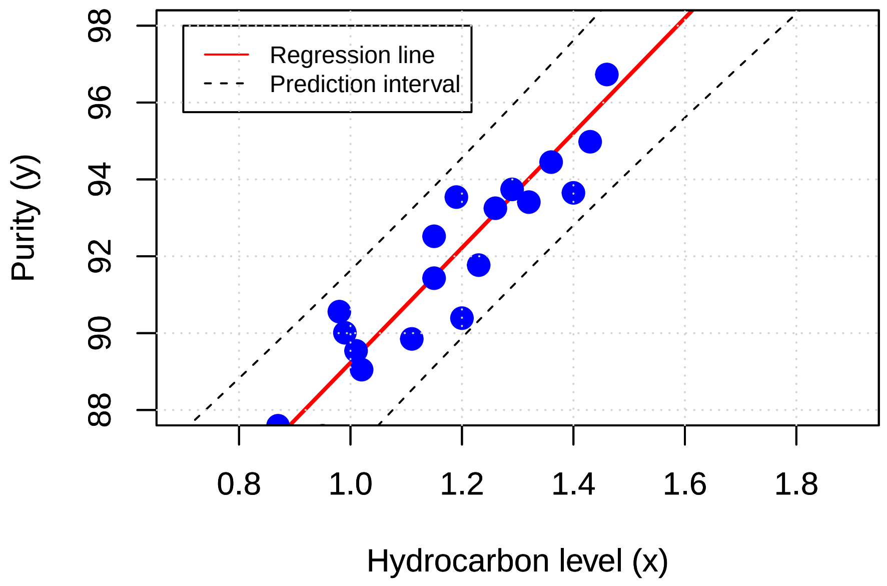

Figure 4: Visualization of the linear regression model with the prediction interval

Conclusion

In this tutorial, we discuss the linear regression model. Hopefully, it can help students to understand the linear regression model and how to use R to create a linear regression model.