Regression

Learning Outcomes

At the end of this session, the students should be able to:

1. discuss the concept of regression;
2. differentiate linear and polynomial regression;
3. implement linear and polynomial regression in an R script using built-in R functions and from scratch; and
4. interpret the results of the said methods.

Content

Linear Regression

Linear regression is the process of modelling the relationship of a dependent and independent variable by means of a straight line. It can also be recalled that an equation of a line is defined by:

$$f(x) = a_1 x + a_0,$$

where a is considered to be the slope of the function, and b is the intercept. It can also be observed that linear equations have a degree of one – the highest exponent that is present in the function.

In linear regression, the equation of the line is defined by the following:

$$f(x) = a_1 x + a_0 + \varepsilon,$$

where the ε is defined as the error term – the part of the regression model that is not explained by the regression model.

Plotting Values in R

To better see the data set that we are operating on, we need to plot certain values in R.

```
> cars #A built in data frame in R
   speed   dist
1      4      2
2      4     10
3      7      4
4      7     22
...
> plot(cars$speed, cars$dist, pch = 20, col = "red", main = "Distance vs. Speed", xlab =
"Speed", ylab = "Distance")
```

To perform linear regression in R, one can use the **`lm()`** function. In the case of the built-in `cars` data set, the car speed is the independent variable, and the distance (referred to as `dist`) is the dependent variable.

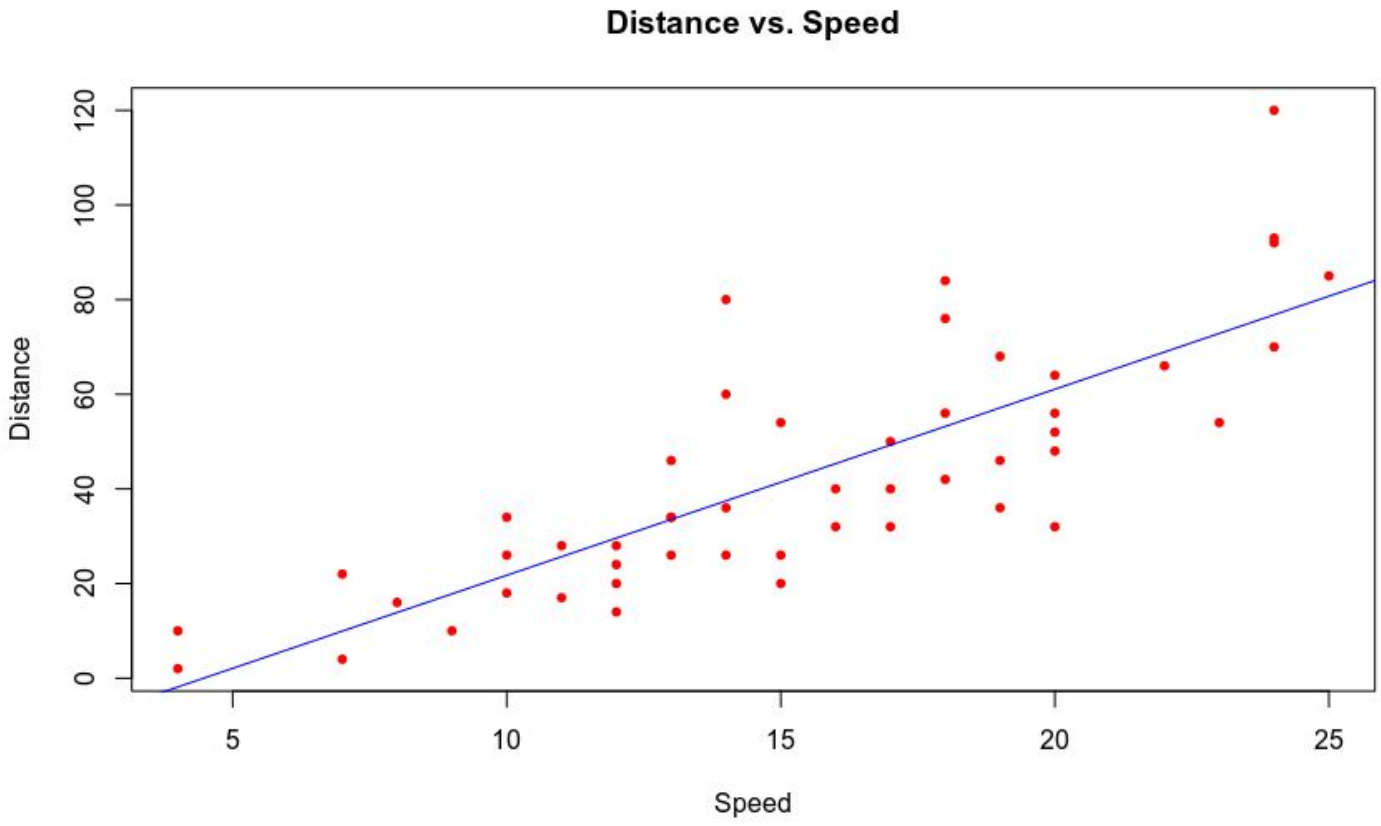
```
> linearModel = lm(dist ~ speed, data = cars)
```

On the environment of RStudio, you should see the variable named `linearModel`, being a list of 12 elements. The coefficients of the linear model can be accessed by: `linearModel$coefficients`.

To plot the linear regression line, together with the scatterplot, you can do the following commands:

```
> plot(cars$speed, cars$dist, pch = 20, col = "red", main = "Distance vs. Speed", xlab =
"Speed", ylab = "Distance")
> lines(cars$speed, predict(linearModel), col = "blue")
```

Upon executing the commands above, you should see a plot similar to this:



Further tests are needed to determine whether the gathered line is indeed statistically significant.

Polynomial Regression

Polynomial regression is the process of modelling the relationship of a dependent and independent variable by means of a polynomial. It can be recognized that a linear function is also a polynomial, but the treatment of this discussion is the generalized form of regression.

A polynomial has a degree n, which is the highest exponent in a term of the polynomial. Given the said information, a polynomial is defined with the form:

$$f(x) = a_n x^n + a_{n-1} x^{n-1} + a_{n-2} x^{n-2} + \ldots + a_2 x^2 + a_1 x^1 + a_0 x^0,$$

where a_i, $i \in [0, n]$ are the coefficients of the polynomial. Since this is a regression line, an error term ε is needed to be added. Such error term is used to represent the unexplained part of the regression line, based from the gathered data.

To perform polynomial regression in R, we can still use the **`lm()`** function, but we shall add the **`poly()`** function, to create a polynomial.

```
> quadraticModel = lm(dist ~ poly(speed, 2, raw=TRUE), data = cars)
> cubicModel = lm(dist ~ poly(speed, 3, raw=TRUE), data = cars)
> plot(cars$speed, cars$dist, pch = 20, col = "red", main = "Distance vs. Speed", xlab =
"Speed", ylab = "Distance")
> lines(cars$speed, predict(quadraticModel), col = "green")
> lines(cars$speed, predict(cubicModel), col = "orange")
```

In this exercise, we shall perform polynomial regression from scratch. To obtain a polynomial regression line of degree n, one must construct an augmented coefficient matrix for a system of $n+1$ equations of $n + 1$ unknowns. The $n + 1$ can be attributed to the number of coefficients that are present in a polynomial of degree n.

In this example, d corresponds to the number of data points. x_i and y_i correspond to the independent (x) and dependent (y) variables of the i-th element of the data set, respectively. We would also assume that x_i and y_i are all real numbers.

This would also mean, that if we have d data points, we can create a polynomial with a maximum degree of $d - 1$.

The general formula for the augmented coefficient matrix is represented in the figure below.

1	$\sum_{i=1}^{d} x_i^0$	$\sum_{i=1}^{d} x_i^1$	$\sum_{i=1}^{d} x_i^2$	$\sum_{i=1}^{d} x_i^3$	...	$\sum_{i=1}^{d} x_i^{n-1}$	$\sum_{i=1}^{d} x_i^n$	$\sum_{i=1}^{d} x_i^0 y_i$
2	$\sum_{i=1}^{d} x_i^1$	$\sum_{i=1}^{d} x_i^2$	$\sum_{i=1}^{d} x_i^3$	$\sum_{i=1}^{d} x_i^4$	...	$\sum_{i=1}^{d} x_i^n$	$\sum_{i=1}^{d} x_i^{n+1}$	$\sum_{i=1}^{d} x_i^1 y_i$
3	$\sum_{i=1}^{d} x_i^2$	$\sum_{i=1}^{d} x_i^3$	$\sum_{i=1}^{d} x_i^4$	$\sum_{i=1}^{d} x_i^5$	...	$\sum_{i=1}^{d} x_i^{n+1}$	$\sum_{i=1}^{d} x_i^{n+2}$	$\sum_{i=1}^{d} x_i^2 y_i$
...	...	...	...	...	...	...	...	...
n	$\sum_{i=1}^{d} x_i^{n-1}$	$\sum_{i=1}^{d} x_i^n$	$\sum_{i=1}^{d} x_i^{n+1}$	$\sum_{i=1}^{d} x_i^{n+2}$	...	$\sum_{i=1}^{d} x_i^{2n-2}$	$\sum_{i=1}^{d} x_i^{2n-1}$	$\sum_{i=1}^{d} x_i^{n-1} y_i$
n+1	$\sum_{i=1}^{d} x_i^n$	$\sum_{i=1}^{d} x_i^{n+1}$	$\sum_{i=1}^{d} x_i^{n+2}$	$\sum_{i=1}^{d} x_i^{n+3}$	...	$\sum_{i=1}^{d} x_i^{2n}$	$\sum_{i=1}^{d} x_i^{2n}$	$\sum_{i=1}^{d} x_i^n y_i$
	x^0	x^1	x^2	x^3	...	x^{n-1}	x^n	RHS

Once the augmented coefficient matrix is constructed, it can then now be solved using an algorithm which gives a solution to a system of linear equations. It should be taken note of that x_i, $i \in [1, n+1]$ is the solution set of the system of linear equations generated by the above augmented coefficient matrix.

The solution set of the system of linear equations, together with the powers of x, given above can be summarized into a function as follows:

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \cdots \\ x_n \\ x_{n+1} \end{bmatrix} \times \begin{bmatrix} x^0 & x^1 & x^2 & \cdots & x^{n-1} & x^n \end{bmatrix} = \begin{bmatrix} x_1 x^0 \\ x_2 x^1 \\ x_3 x^2 \\ \cdots \\ x_n x^{n-1} \\ x_{n+1} x^n \end{bmatrix}$$

The sum of the resulting vector can then be used to construct a polynomial which is a regression line, based on the given data. The said regression line will be of the form:

$$f(x) = x_1 x^0 + x_2 x^1 + x_3 x^2 + \ldots + x_n x^{n-1} + x_{n+1} x^n$$

Overfitting

It should be taken note of that regression only describes the trend or the fitness of the data, given a certain polynomial. It can be used on instances where there are two data points having the same values of the independent variable (i.e. the data set is a relation). This is because, the output of regression is a function which describes the trend, and a function is a one-to-one mapping.

Another is that we would not like to consider the residual noise in graphing the function. Doing so will make our regression line fail to predict future observations, if we are to use such function. This can be avoided by using an n which is sufficiently less than the number of data points observed. If it is intended to create a function which plots the exact location of the data points, interpolation should be considered instead.

Evaluating Strings

To have strings to be converted into an R expression, use the **`parse()`** function. To execute the said expression, use the **`eval()`** function.

```
> s = "function (x) = x + 4"
> parse(text = s)
expression("function (x) = x + 4")
> eval(parse(text = s))
function(x) = x + 4
```

Learning Experiences

1. Students will have a hands-on experience in importing previously constructed codes in R.
2. Students will create their own R script that will create a polynomial regression line from a given data set.
3. Students will use their R script to solve a given problem which demands to be solved by polynomial regression.
4. Students will attempt to accomplish ***sample exercises for self-learning*** provided below.

Sample Exercises for Self-learning

1. Required Competencies:
   Linear Regression, Polynomial Regression

Create an R code which imports the previous exercises on the augmented coefficient matrix, Gaussian and Gauss-Jordan Elimination. On the same code, create a function named `PolynomialRegression` which accepts a vector for the dependent variable, a vector for the independent variable, and an integer which is the degree of the polynomial that you wish to create a polynomial regression of.

The functions should return a labelled list which has the following labels:

[a] `coefficients`
A vector of numeric values which contains the coefficients of the polynomial.

[b] `function`
A function in R, in which when run, will compute for the value of the regression polynomial, with a given value of x.

The function should also return the value NA if the integer input is a value less than one. This is because regression will not be valid on the said domain.

The function will also return NA if the length of the vectors for the dependent and independent variable are different of each other.

Assessment Tool

A ***programming exercise*** on the implementation of polynomial regression, written in an R script.

Exercise

The following data is collected from an experiment which produced yields from a given temperature.

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Temperature	50	50	50	70	70	70	80	80	80	90	90	90	100	100	100
Yield	3.3	2.8	2.9	2.3	2.6	2.1	2.5	2.9	2.4	3.0	3.1	2.8	3.3	3.5	3.0

Find an appropriate polynomial for the data set as presented, for degrees 1-5. Use the code that you have created in finding the solution, and the built in R functions to check your answer.