# Multivariate linear regression analysis
## for robot delivery time prediction

*Muny-Roth Chev, student of Master of Integrated Design, specialized in Computational Design.*

### I. Introduction

Regression analysis is a statistical method used to determine the relationships between variables and make predictions based on those relationships. It is a versatile tool that is used in various fields such as economics, finance, social science and engineering and allows to analyze and model complex relationships between variables. [1]

In the context of the improvement of the functioning of the robot Lifbot, a construction hoist allowing to lift weight up and down a rail attached to scaffolds, we are trying to predict the value of time based on other multiples other parameters. Considering the robot have a constant speed for each run, (one run being the time the robot takes to go up, then down) we decided to focus on the total time of the project. The other parameters that would allow us to predict this time would be given by the user : the total weight lifted, the height of the scaffold and the lenght of the scaffold.

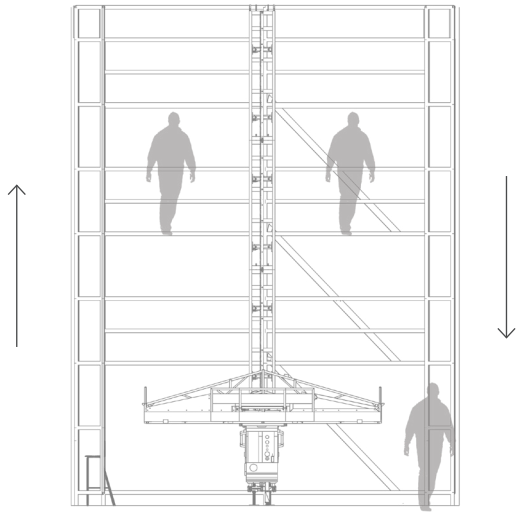


*fig. 1: drawing of Liftbot installation*
*Source: Kewazo, Author*

The main reason why it is interesting for us to use linear regression is because we have a large dataset provided by the continuous usage of the robot, containing both values for the independent and dependent variables. [2] It is recommended to have a minimum of 10 observations or cases for each independent variable included in a linear regression model. For instance, if a model has four predictors, a sample size of at least 40 cases would be needed to ensure sufficient statistical power and reliability of the estimates. This guideline is based on the idea that having an adequate sample size helps to reduce the risk of overfitting and increases the generalizability of the model to new data. [3] The second reason for using linear regression, being the underlying linearity between the independent and dependent variables, as the time taken to transport the material is proportionally bigger as the structure gets heigher and wider, and the material transported gets heavier.

### II. Theory

Univariate regression analysis uses one independent variable to analyze the relationship between that variable and a dependent variable. The result is an equation that represents the linear relationship between the two variables:

$$y= \beta_0 +\beta_1 x_1+ \varepsilon$$

where $y$ is the dependent variable; $\beta_0$ is the value of $y$ when $x_1$ equals to 0, also called an intercept, $x_1$ is one dependent variable, $\beta_1$ is the regression coefficient and $\varepsilon$ is a random error term that represents the difference in the linear model and a particular observed value for $y$.

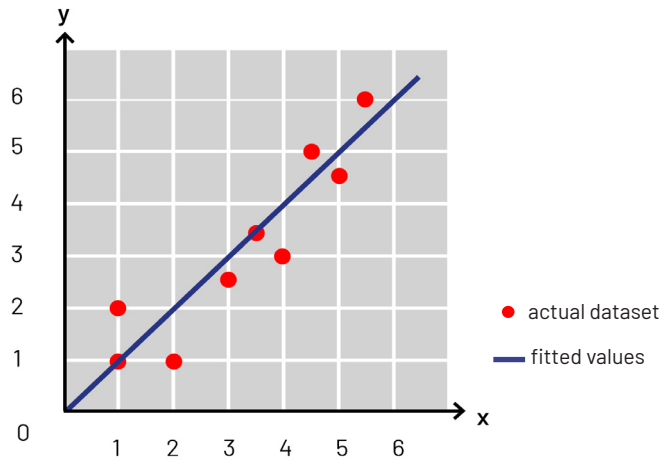
Multivariate regression analysis, on the other hand, uses more than one independent variable to account for variations in the dependent variable. [4] The model for multivariate regression analysis is formulated :

$$y= \beta_0 +\beta_1 x_1+ ...+ \beta_n x_n+ \varepsilon$$

Here we can see we have multiple dependent variables and their regression coefficients. In linear regression, the dependent variable is the variable we want to predict, while the independent variable(s) is the variable(s) we use to make that prediction. Because the independent variable(s) can help us predict the value of the dependent variable, it is often referred to as the predictor variable. [1] In our previous example, our predictor variable is the time we are trying to predict, while the parameters of height, lenght and weight transported are the one helping us reaching that goal.

The purpose is to identify the linear relationship between the independent variables and the dependent variable, and to estimate the coefficients of the linear model. This can be done using the least squares method, which minimizes the sum of the squared errors between the predicted values and the actual values in the dataset. Once the coefficients of the linear model have been estimated, they can be used to make predictions on new data points.

### a) Cost function




| y | x | $\overline{y}$ | $\overline{y}$ - y |
|---|---|---|---|
| 1 | 1 | 1 | 1-1 = 0 |
| 1 | 2 | 2 | 2-1 =0 |
| 2 | 1 | 1 | -1 |
| 2.5 | 3 | 3 | 0.5 |
| 3 | 4 | 4 | 1 |
| 4 | 3.5 | 3.5 | -0.5 |
| 4.5 | 5 | 5 | 0.5 |
| 5 | 4.5 | 4.5 | -0.5 |
| 6 | 5.5 | 5.5 | -0.5 |

*fig. 2: Graphical representation of linear regression data.*
*Souce: Udemy. "Deep Learning Foundation : Linear Regression and Statistics."*

The red points in this graphic example represents data points from the dataset, and the blue curve represents the prediction or the best fitted line, which can be written, as previously mentioned:

$$\overline{y} = \beta x + \beta_0$$

To check how good of a fit is the blue curve, it is necessary to measure the error which can be defined by $\overline{y}$ - y with is the difference between the real value of y provided by the dataset and the $\overline{y}$ which is predicted value given by our fitted curve. As it is necessary to sum out all the values without negatives, the sum of the total error is then squared. The result is the cost function which can be written as:

$$\text{Cost function} = (\overline{y} - y)^2$$

We can then replace the formula of $\overline{y}$ in the cost / loss function:

$$L = ((\beta x + \beta_0) - y)^2$$
$$L = f(\beta, \beta_0)$$

Considering the size of the dataset and the number of points have also an influence, we can rewrite the cost function:

$$L_{total} = \frac{1}{N}\sum_{i=1}^{N}((\beta x + \beta_0) - y_i)^2$$

The goodness of fit of the curve is measured by this loss function. The lower this value is, the better is the curve fitted. [5] The process of gradient descend will then allow to minimize the loss function, progressively update the each regression coefficients and get the best fitted curve.

### b) Gradient descent

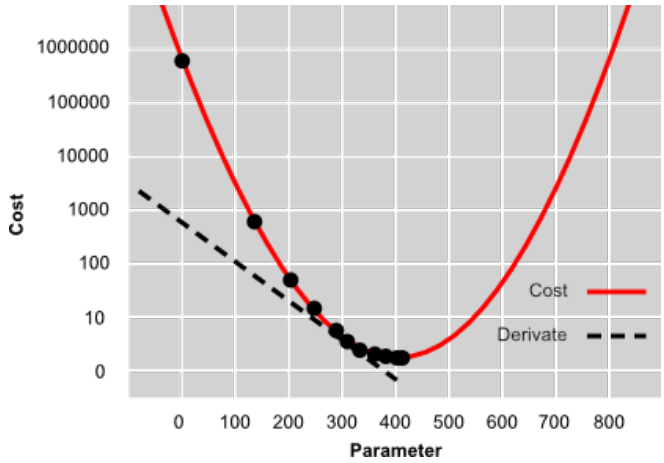



*fig. 3: Graphical representation of the processus of gradient descend*
*Source: quicktomaster.com*

The gradient descent is an iterative solution that incrementally steps towards an optimal solution and is used in varieties of situations. This process has to be implemented for each parameter of the loss function. It usually starts with an initial guess and then improves it one at the time, to reach an optimal solution. In order to reach the minima of a function, we use the derivative as its value indicate how far we are to the optimized solution. [6]

The magnitude of the derivative is proportional to how big of a step we should take towards the minimim. A relatively large value for the derivative corresponds to a steep slope and suggest that we are relatively far from the bottom of the curve, while a small value suggests we are relatively close to the bottom of the curve. The sign (+/-) indicates us which direction: a negative derivative tells us, it is necessary to take a step to the right to get to the minima, while a positive derivative indicates we need to take a step to the left. [6]

The derivative in relationship to the intercept $\beta_0$ can be written in the following way:

$$\frac{\partial L}{\partial \beta_0} = \frac{\partial}{\partial \beta_0}\left(\sum_{i=1}^{N}((\beta x + \beta_0) - y_i)^2\right)$$

Using the chain rule, we can then rewrite it in the following way:

$$\frac{\partial}{\partial \beta_0} = 2\sum_{i=1}^{N}( ( \beta x + \beta_0 ) - y_i)$$

The derivative in relationship to the parameter $\beta$ can also be written:

$$\frac{\partial L}{\partial \beta} = \frac{\partial}{\partial \beta}(\sum_{i=1}^{N}( ( \beta x + \beta_0 ) - y_i)^2)$$

$$\frac{\partial L}{\partial \beta} = 2\sum_{i=1}^{N} x_i( ( \beta x + \beta_0 ) - y_i)$$

After getting the the derivative equations, it is now possible to calculate the step size from one point to another on the curve, which is written:

$$\text{Step size} = \text{Derivative} \times \text{Learning rate}$$

The learning rate being a value that prevents us taking steps that are too big and skiping past the lowest point in the curve. Typically for Gradient descent, the learning rate is determined automatically: it starts relatively large and gets smaller with every steps taken. [6]

As we want to get closer to the optimal value for the intercept ($\beta_0$ in our case), the formula will be written:

$$\text{New intercept} = \text{Current intercept - Step size}$$

Then, we repeat those different steps updating the intercept after each iteration until the step size is close to 0, or we take the maximum number of steps, which is often set to 1000 iterations. Those steps are also repeated for $\beta$.

### c) Goodness of fit

R-squared ($R^2$): measures the squared correlation between the actual outcome values and the values predicted by the model. The higher the adjusted $R^2$, the better the model. An $R^2$ value of 1 indicates a perfect fit, while values closer to 0 indicate a poor fit. [3]

$$R^2 = \frac{Model\ sum\ of\ squared}{Total\ sum\ of\ squared} = \frac{\sum_{i=1}^{n}(\hat{y}_i - \bar{y})^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

Where $\hat{y}_i$ is the predicted outcome, $y_i$ is the observed outcome and $\overline{y}$ is the average of observed outcomes.

Other measures of error include the Root Mean Squared Error (RMSE), which measures the average magnitude error made by the model while predicting an observation's outcome. It's the square root of the average of squared residuals. Residuals are the difference between the actual values and the predicted values. The lower the RMSE, the better the model. Mean Absolute Error (MAE) is a measure of average absolute differences between observed and predicted outcomes. The lower the MAE, the better the model. [3]

## III. Method

### a) Sample

The research data presented below, under the csv format is composed of a the columns Height, Lenght which corresponds to the height and lenght of the structure in meters, the column Time, which corresponds to the total time of the project in hours, as well as the column Weight which corresponds to the total weight transported in tons.

```
Project,Height,Lenght,Time,Weight
1, 32, 15, 42, 36.000
2, 55, 22, 137, 40.572
3, 25, 7.5, 32, 16.250
4, 38, 12, 41, 25.240
5, 39, 14, 50, 27.536
6, 20, 6, 15, 12.660
7, 35, 12, 44,32.922
8, 49, 15, 89, 37.750
9, 24, 9, 22, 18.205
10, 42, 14, 65, 33.750
11, 48, 12, 66, 28.750
12, 30, 12.5, 48, 33.428
13, 25, 8.5, 36, 20.510
14, 15, 6, 25, 15.620
15, 52, 18.5, 120, 42.600
16, 45, 14, 74, 35.400
17, 42, 13, 58, 31.250
18, 20, 7, 34, 17.520
19, 24, 11, 33, 18.600
```

### b) Programming

Based on the theory previously presented, we are going to go through the coding of the algorithm, underline the most important points and provide an analysis of the results.

The first step of the coding involves importing the different libraries necessary for the process:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib as mpl
mpl.use('qt5agg')
```

We use Pandas for data analysis and extracting our data from the csv file, NumPy to deal with array-like objects and finally Matplotlib to plot our values.

We are then importing both the time column as our Y, which is our dependent variable and the columns height, lenght and weight, as our X, which are our independent variables.

```
#Multivariate regression analysis

X = df [['Height', 'Lenght', 'Weight']]
Y = df.Time
```

The second step is to initialize the values of regression coefficients with random numbers, but also the values of Y current, "pre-cost" function that is later on going to be updated with the cost function, and learning rate for the Gradient descent.

```
# Y = mX+c
# Initializing m and c with random numbers
k = X.shape[1] # return the total number of parameters in X
m_current = np.zeros(k)
c_current=0
N = X.shape [0] # return the total number of values
Y_current = 0
print (X)

# cost function
pre_cost = sum ((Y-Y_current)**2) /2

# learning_rate
learning_rate = (1 / pre_cost)
```

As an additional step, we ask the user to input values for height, lenght and weight, that are going to be plotted afterwards, in order to find the time values for those specific input. np.array (user_weight) creates a numpy array from the user_weight variable, and reshape(1, -1) changes the shape of the numpy array to have one row and as many columns as necessary (-1). This is done to ensure that the array has the right shape to perform matrix multiplication with the m_current variable, which has a shape of (3,) (3 features in X: Height,Weight and Lenght). The shape of user_weight_np after this line is (1, 1), which is compatible with the shape of m_current.

The next step is the start of the Gradient descent, we use a for loop and give a certain number of iterations that can then be adjusted afterwards.We set the formula for the partial derivatives in regards to m and c ( since Y= mX + c in the script) and through the process of Gradient descent, updating those values with the learning rate we previously defined. With the new values of m and c, Y and the value of the cost function are updated.

```
for i in range (25):

    m_grad = ((2/N) * (np.dot(X.T,(Y_current- Y))))

    c_grad = (2/N) * sum (Y_current-Y)


    # Multiply the gradient with the learning rate
    # Update m and c by substracting the derivative from initial numbers

    m_update = m_current - m_grad * learning_rate
    c_update = c_current - c_grad * learning_rate

    #We need to sum all the X column values
    Y_update = (m_update * X). sum (axis = 1) + c_update



    #Cost function
    cost = sum ((Y_update - Y)**2) / N

    print ('cost', i, cost)
```

We set a new condition, where if the new cost value is lower than the previous cost, we then update the values of m, c, the cost and plot the values for each iterations.

```
#Verify that if the cost value gets lower, then we are on the right track

if pre_cost > cost:
      m_current = m_update
      c_current = c_update
      pre_cost = cost

      # Trying to plot our values for only cost (to see how the cost infl

      fig = plt.figure(figsize=(12,12))

      fig.subplots_adjust(left=0.1, bottom=0.1, right=1, top=0.9, wspace=

      # creating subplots
      ax0 = plt.subplot2grid((3,6), (0,0), rowspan=1, colspan=2)
      ax1 = plt.subplot2grid((3,6), (1,0), rowspan=1, colspan=2)
      ax2 = plt.subplot2grid((3,6), (2,0), rowspan=1, colspan=2)
      ax3 = plt.subplot2grid((3,6), (0,2), rowspan=2, colspan=4, projecti
```

In order to get a linear curve, it is also necessary to sort the values before plotting:

```
    #sorting the values for weight before plotting
    sorted_x0 = np.sort(df.Weight, axis=None)
    sorted_x0 = sorted_x0[:, np.newaxis] # add a new axis
    sorted_y0 = (m_update * sorted_x0).sum(axis=1) + c_update

    #sorting the values for height before plotting
    sorted_x1 = np.sort(df.Height, axis=None)
    sorted_x1 = sorted_x1[:, np.newaxis] # add a new axis
    sorted_y1 = (m_update * sorted_x1).sum(axis=1) + c_update

    #sorting the values for height before plotting
    sorted_x2 = np.sort(df.Lenght, axis=None)
    sorted_x2 = sorted_x2[:, np.newaxis] # add a new axis
    sorted_y2 = (m_update * sorted_x2).sum(axis=1) + c_update
```

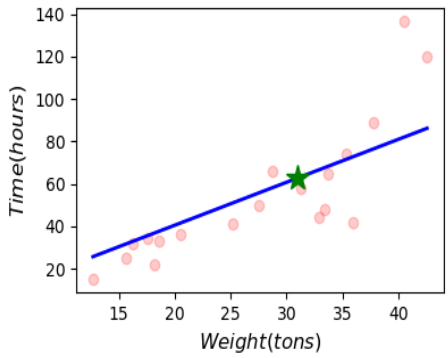
Finally, to show the results we plot each of the independent variables in relationship to the dependent variable of time, to show how each one of them affect the total time globally. We also plotted the results in 3D to see the results in regards to Time, Weight and Height.

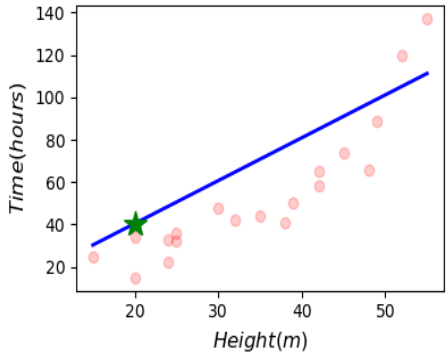
After the end of the iterations and the process of Gradient descent, the coefficients of regression as well as the optimal time for the user inputs are printed.The final $R^2$ is also calculated, as a way for us to analyze how good of a model we have.

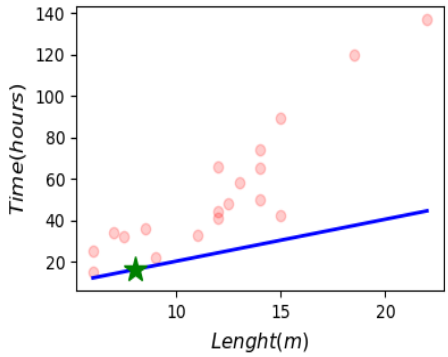
```
print(cost,learning_rate)
print ("m_current =", m_current)
print ("c_current =", c_current)
print ("optimal time (h) =", (m_update * user_np).sum(axis=1) + c_update)


# calculate R2 (coefficient of determination) in order to find out how soli
def r2(y_true, y_pred):
    y_bar = np.mean(y_true)
    ss_tot = np.sum((y_true - y_bar)**2)
    ss_res = np.sum((y_true - y_pred)**2)
    r2 = 1 - (ss_res/ss_tot)
    return r2


# calculate R2
Y_pred= (m_update * X).sum(axis=1) + c_update
r2_score = r2(Y,Y_pred)
print('R2 score: {:.4f}'.format(r2_score))
```

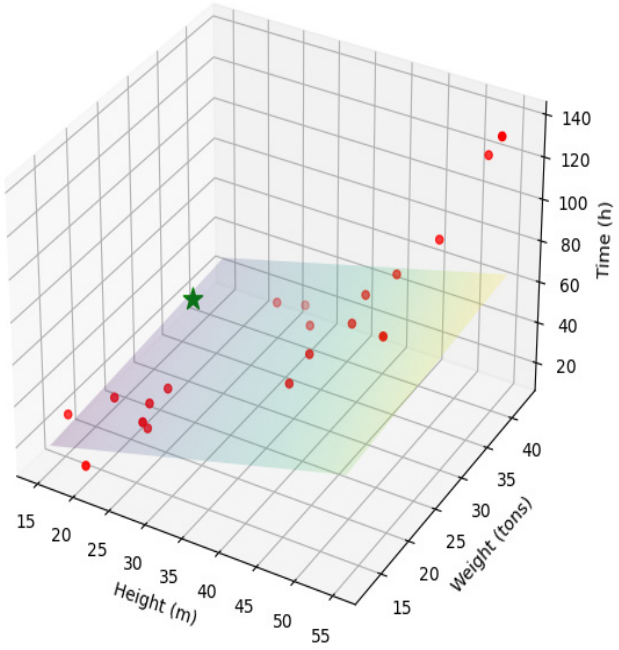
### c) Interpretation of the results

The graph shows us correctly the dataset and the fitted values for each variable. We can get a sense of how each of these variable impact the final Time. We also can see the input value of the user.




The printed values shows us how the progression of the iterations and the progression of the cost error, which allows us to re-adjust the learning rate to reach better final values. The final $R^2$ of 0.705 indicates that the model explains 71% of the variability in the data around its mean. In other words, there is still 29% of the variability that is not explained by the model. $R^2$ can be improved with various techniques such as normalization of the data or inscrease of the dataset. [2]

```
please enter the value of the total load you want to carry (in tons): 31
please enter the height of the structure(m): 20
please enter the lenght of the structure(m): 8
cost 0 3072.150234233784
cost 1 2332.318205601131
cost 2 1706.5565456809873
cost 3 1194.8652544733536
cost 4 797.2443319782294
cost 5 513.6937781956158
cost 6 344.213593125512
cost 7 288.8037767679182
cost 8 347.46432912283416
cost 9 289.5366654113468
cost 10 288.8257339663405
cost 11 288.8054591711011
cost 12 288.8039398750698
cost 13 288.80379302730177
cost 14 288.8037783933431
cost 15 288.80377693045546
cost 16 288.8037767841719
cost 17 288.80377676954356
cost 18 288.8037767680807
cost 19 288.8037767679346
cost 20 288.8037767679196
cost 21 288.8037767679184
cost 22 288.80377676791824
cost 23 288.8037767679181
cost 24 288.80377676791824
288.80377676791824 2.681144848850459e-21
m_current = [0.94843665 0.32954939 0.74460114]
c_current = 0.023277981803493254
optimal time (h) = [44.71104122]
R2 score: 0.7058
```

### IV. References


(1) Ng, Set Foong, Yee Chew, Pei Chng, and Kok Shien Ng. "An Insight of Linear Regression Analysis." Scientific Research Journal 15 (December 31, 2018): 1.

(2) Montgomery, Douglas C., Elizabeth A. Peck, and G. Geoffrey Vining. Introduction to Linear Regression Analysis. 5. edition. Hoboken, NJ: Wiley, 2012.

(3) El Aissaoui, Ouafae, Yasser El Alami El Madani, Lahcen Oughdir, Ahmed Dakkak, and Youssouf El Allioui. "A Multiple Linear Regression-Based Approach to Predict Student Performance." In Advanced Intelligent Systems for Sustainable Development (AI2SD'2019), edited by Mostafa Ezziyyani, 1102:9–23. Advances in Intelligent Systems and Computing. Cham: Springer International Publishing, 2020.

(4) Kaya Uyanık, Gülden, and Neşe Güler. "A Study on Multiple Linear Regression Analysis." Procedia - Social and Behavioral Sciences 106 (December 1, 2013): 234–40.

(5) Udemy. "Deep Learning Foundation : Linear Regression and Statistics." Accessed March 20, 2023. https://www.udemy.com/course/linear-regression-in-python-statistics-and-coding/.

(6) PhD, Josh Starmer. The StatQuest Illustrated Guide To Machine Learning. Independently published, 2022.