ENG-270

Computational Methods and Tools PROJECT Report

MARTIN Adèle, MAEDA Zoé

1. Introduction

For this programmation project in Computational Methods and Tools, we decided to use our knowledge in statistics to determine a relationship between two variables: pollutant concentrations in fishes and the distance to the pollution emitter. By coding the tools useful for linear and exponential regressions, we were able to determine relationships, which we later used to find where a certain polluter was, based on pollutant concentrations in fishes collected in a river.

This project was divided into three parts. First, we determined exponential and linear relationships between the concentrations of different pollutants and the distance to the polluter; the data used to determine these relations consisted of real data points (concentrations in fishes and distances associated), provided by Professor Jean-Christophe Loubier (HES-SO Valais-Wallis). Then, we applied these relations to an imaginary case, the case of Ennui-sur-Blasé (imaginary town), which faced high pollution levels in its river: by applying the relationships found earlier to the concentrations of fishes collected in the river of Ennui-sur-Blasé, we were able to determine where the polluter was located. Finally, we checked the precision of our relationships and reliability of the distances predicted for the Ennui-sur-Blasé case.

For this project, we mixed our knowledge in programmation and our mathematical knowledge in statistical tools, to find relationships that approximately described what happens in a polluted river and that could be useful for pollution prediction.

2. Methods

The main goal of our project is to define a relationship between pollutant concentrations in fish and the distance to the polluter. To find this relationship and calculate its precision, we used several statistical tools: linear regression, exponential regression, uncertainty and variance of prediction.

2.1. Linear regression

Linear regression is a model that estimates a linear relationship between a dependent variable y and one or more independent variables x. In our case, there is only one independent variable x, so linear regression consists in finding a good estimation of the two parameters a and b in the equation: **y = ax + b**.

Its objective is to minimize the difference (error) between the observed values (y) and the predicted values ($\hat{y}$). To achieve this objective, we use the least squares method, which consists in minimizing the mean squared error: $\frac{1}{2m}\sum(y_i - \hat{y}_i)^2$ ($\hat{y}_i$ is the predicted value $\hat{y}_i = ax_i + b$, and m is the number of data points (x_i, y_i)).

The mean squared error is: $J(a,b) = \frac{1}{2m}\sum(y_i - \hat{y}_i)^2 = \frac{1}{2m}\sum(y_i - ax_i + b)^2$. Since it is a function of a and b, we can calculate the gradient of J(a,b): $(\frac{\partial J}{\partial a}(a,b), \frac{\partial J}{\partial b}(a,b))$. The gradient is used in linear regression to do what we call a gradient descent.

To concretely find the linear relationship "y = ax + b", we have to follow these steps:

(1) Find ourselves a first estimation of the parameters a and b, by analyzing the graph of the observed values y versus x (give an approximation of the slope a and the intercept b). In addition to the initial values for the parameters a and b, choose a learning rate α.

(2) Calculate the mean squared error $\frac{1}{2m}\sum(y_i - \hat{y}_i)^2$.

(3) Calculate new temporary values for a and b: (this is a gradient descent)

$a_{temporary} = a - \alpha * \frac{\partial J}{\partial a}(a,b)$

$b_{temporary} = b - \alpha * \frac{\partial J}{\partial b}(a,b)$.

(4) Calculate the mean squared error $\frac{1}{2m}\sum(y_i - \hat{y}_i)^2$ with the temporary values $a_{temporary}$ and $b_{temporary}$ as the parameters a and b.
If the new mean squared error with the temporary values < the mean squared error calculated in (2): $a \leftarrow a_{temporary}$, $b \leftarrow b_{temporary}$ and $\alpha \leftarrow 1.5\alpha$.
Otherwise: a and b stay the same, and $\alpha \leftarrow 0.5\alpha$.

(5) Go back to (2).

This is an iterative approach and we chose to stop it until a certain minimal value of the mean squared error (which we chose arbitrarily). The values used for the parameters a and b are the final values calculated during the iterative approach.

2.2. Exponential regression

Exponential regression is a model that estimates a relationship between a dependent variable y and an independent variable x, of the general form: $\mathbf{y = a \cdot e^{bx}}$.

To make an exponential regression, we can go through a linear regression. Indeed, by taking the natural logarithm "ln" on each side of the equation, we find another relationship: **ln(y) = ln(a) + bx**, i.e. **y' = a'x + b'**, where y' = ln(y), a' = b, b' = ln(a). So, by taking ln(y) as the dependent variable, we can find the parameters a' and b' using a linear regression, and then deduce the parameters a and b of the exponential relationship "$y = a \cdot e^{bx}$".

2.3. Uncertainty / variance of prediction

In our project, we calculated two relationships: one linear and one exponential. By finding the uncertainty of each model, we can deduce which one fits our data better.

In our model, what was predicted finally is the distance from the polluter. Indeed, initially, we calculated a relation between y = the concentration (or *ln(concentration)*)

in the case of the exponential model) and x = the distance from the polluter: y = ax + b. This relationship was determined from real data points (x_i, y_i) (we knew both the concentration and the distance from the polluter for each fish).

But then we "inverted" it: x = (1/a)y - (b/a) (still a linear relationship !). So, x becomes the new y. Indeed, for the Ennui-sur-Blasé case, we only knew the concentration values and wanted to know what the distance from the polluter was. We made an assumption that all the fishes had been collected in the same place (so at the same distance from the polluter), so to estimate the real distance, we chose to calculate the average value of all the distances predicted from the relationship "x = (1/a)y - (b/a)" (to only have one estimated value of the distance, because there is only "one" distance). So, we have 2 linear relationships:

- y = concentration (or *ln(concentration)*) and x = distance: y = ax + b **(1)**
- x = distance and y = concentration (or *ln(concentration)*): x = (1/a)y - (b/a) **(2)**
  a' = 1/y ; **b' = -(b/a)** -> **x = a'y + b'**.

Very important: By taking a look at the graph of iron concentrations versus distance with the line of the linear and exponential relationships found earlier, and comparing it with the cadmium case, we saw that the iron relationship seemed to better describe the reality than the cadmium relationship with cadmium concentrations, so we made the assumption that the Ennui-sur-Blasé concentrations were iron concentrations, so that we could use the iron relationships which seemed more precise and so that the distance predicted would be more reliable. We also made the assumption that the iron relationships were valid in any river (the initial river and the river of the Ennui-sur-Blasé).

To calculate the standard error of prediction for each model, we need:

(1) Variance of residuals: $\frac{\Sigma\,((distance\ \ i) - (distance\ i\ predicted\ by\ the\ model))2}{m-2}$

m = number of data points.
This is a measure of how well the model fits the calibration data.
Here, we use the real data points thanks to which we could define the linear relationship **(1)**. The distance i is predicted by the model **(2)**.

(2) Variance of **a'**: $\frac{Variance\ of\ residuals}{\sum\,((distance\ i) - (mean\ distance))2}$

Here, we also use the real data points thanks to which we could define the linear relationship **(1)**, and it makes sense because a was determined thanks to those data points. The distance i is predicted by the model **(2)**.
The mean distance is the average of the distances of those observed data points (real data).

(3) Variance of **b'**: (Variance of residuals)^2 · $(\frac{1}{m} + \frac{(mean\ distance)2}{\sum\,((distance\ i) - (mean\ distance))2})$

m = number of data points.

(4) Variance of prediction:

(Variance of residuals)^2 · $(1 + \frac{1}{m} + \frac{\Sigma\,((concentration\ ESB\ \ j) - (mean\ concentration))2}{\sum\,((distance\ i) - (mean\ distance))2})$

Here, “concentration ESB j” is part of the concentration values of the Ennui-sur-Blasé data (for which we want to determine the distance, and which verifies the iron relationship determined thanks to our data points). Then, “mean concentration” means the average value of the distances of the data points thanks to which we determined **(1)** (so NOT the Ennui-sur-Blasé data). “Distance i” and “mean distance” refer also to those data points.

We can compare the variance of prediction for both the linear and the exponential models. The one with the lower variance of prediction fits our data better, so the distance predicted by that model seems more reliable.

3. Approach used / programmation tools

3.1 General structure of the code

Our project is separated between 3 programs :
The first one is a pre-analysis of the data (fish concentrations and distances associated for several pollutants) : premiersgraphs.py
We plotted the data on Python which enabled us to:

1. select what data to use : iron and cadmium
2. eliminate the possible outliers in this data set
3. have a first idea of the relationship between the distance to the pollution emitter and the pollution concentration in the fishes.

After that, in a second program, we used linear and exponential regressions to find the parameters of the two models, this time on C : regression_lineaire.c

Finally, we used these newly found parameters to plot the final graphs of the data, and the models in Python. In addition, we used the relationship we found to predict the distance to the pollution emitter for the Ennui-sur-Blasé case (for which we only had concentrations in fishes collected at the same place, and for which we wanted to predict the distance from the pollution emitter). Then we found the uncertainty of this distance based on the data and regression : final_graph_prediction.py

3.2 Programming languages used

In this project, we chose to use two programming languages, a high-level language: Python, and a low-level one: C.

We chose to use Python because we have more knowledge in Python than in Matlab. Python is also open-source and free, making it easier to find information on how to resolve certain bugs and other issues, thus more widespread than Matlab.

In addition to this high-level language, we also used a low level-language that is C. C can handle a lot of data and a lot of calculations. Indeed it is very fast in comparison

to Python or Matlab as it is a low-level programming language (no automatic memory management).

3.3 Programming tools and paradigms used

During this project, we had to use other libraries than what is initially in Python or C. First of all, we had to use the CSV library of Python to read CSV files. We also used “matplotlib” to plot data. Finally, we used “numpy” to use a vectorization approach, which enabled us to do a lot of calculations in a few lines of code. Indeed, using vectorisation, we can do our calculations on full matrices instead of having to use loops to calculate each element individually. We can also easily calculate matrix multiplication and other built in functions such as np.zeros() or np.colomn_stack().

In the regression_lineaire.c program, we used the <stdio.h>, <string.h> and <stdlib.h>, they enabled us to use functions that are not yet in C such as printf(), strtok() or atof(). In this second program, we used a functional approach as functions are more easily implementable and can be tested individually. To some extent, they enable us to do calculations without worrying about changing parameters or global variables (in our case we often use arrays that are mutable within function).

3.4 Why use certain languages for each part

Python, especially the “numpy” library, was also useful to implement vectorisation principles that are very useful, easier and more intuitive when doing calculations (than using loops and lists).

We used Python to print out graphs and do basic calculations. That doesn’t require a lot of calculation and is quite easy to implement in Python using widespread libraries.

On the other hand, linear regression requires quite heavy calculations which would take Python much more time than C. While it would be easier to code in Python with a vectorization approach, the final calculation would take too long thus we chose to code this part in C.

3.5 What automatization we chose

In this project, we chose to link the different programs using files. Indeed, between the first analysis and the linear regression, it makes sense to use a file in which to write in, as the new data could be useful to have later on (in the final analysis for instance). As for the link between the second and third program, we also used a file instead of calling the regression_lineaire.c program. By writing our results in a file, we don’t have to rerun the C program (that takes some time) every time we run the final analysis program (without using the REPL). Moreover, it is easier to work with and could enable us to have additional programs using the parameters found in regression_lineaire.c by just reading a file.

4. Results

4.1. First analysis of the initial data

Initially, we had a collection of fish concentrations and associated distances for several pollutants: cadmium, copper, iron, and zinc. We plotted concentration versus distance on a graph on Python for each of these pollutants, and then analyzed it to see if the two variables seemed correlated (and if it seemed to verify a relation that looked approximately linear or exponential). The two pollutants that fulfilled the best these criteria were: iron and cadmium. Then, for each of these pollutants, we eliminated outliers in the data, i.e. data points that significantly differed from the other observations in the data set and did not seem to verify the linear or exponential relationships; so, we got an "adapted / fixed" data for iron and cadmium, which we later used to make linear and exponential regressions.

Iron LINEAR $y = ax + b$	Iron EXP $y = a \cdot e^{bx}$ $y' = a'x + b'$, with: - $y' = ln(y)$, - $a' = b$, - $b' = ln(a)$	Cadmium LINEAR $y = ax + b$	Cadmium EXP $y = a \cdot e^{bx}$ $y' = a'x + b'$, with: - $y' = ln(y)$, - $a' = b$, - $b' = ln(a)$
a = - 0.139511	a' = -0.000452	a = -0.032956	a' = - 0.001099
b = 399.992892	b' = 5.900002	b = 64.999997	b' = 4.062654

4.2. Efficiency / precision of the linear and exponential models, comparison

Once we determined the linear and exponential relationships for both iron and cadmium, we plotted it to check if those relationships seemed correct.

Then, for the Ennui-sur-Blasé case, we calculated the variance of prediction for both the linear and exponential models (which were determined thanks to our initial data points for iron). The variance of prediction for the linear model was lower than the variance of prediction of the exponential model, we could then deduce that the distance predicted by the linear model was more reliable (the linear model is a better model to describe our situation). Indeed, the square root of the variance for respectively linear and exponential models were:

- $\sigma(linear\ model)$ = 296.9896105800691
- $\sigma(exponential\ model)$ = 338.65782507962047
  -> $\sigma(linear\ model) < \sigma(exponential\ model)$.

5. Discussion

We could have tried to determine other types of relationships between concentration and distance, for example polynomials of degree 2+, but the approach would have

been more difficult and not within our mathematical knowledge. But still, it would have been interesting to compare other models.

In addition, we didn't have a lot of data to train the model with but the programs can run with more data.

Finally, we assumed that the two rivers (the one to train the model and the one where we want to find the pollution emitter) and two pollution emitters were similar enough to use the same model in both cases, which is not necessarily the case. Nevertheless, we could still expect a linear model in both cases.

6. Conclusion

In conclusion, during this programmation project, we used statistical tools to determine relationships which approximately described the evolution of pollutant concentrations in a river. Those statistical tools also allowed us to compare two different models (exponential and linear), to see which one seemed to describe reality better. Finally, we applied these relationships to a concrete case to predict where pollution was coming from.

This project allowed us to improve our coding skills and to find useful relationships for pollution production, in addition to learning how to do computer math.

7. Bibliography

"What Is Gradient Descent?" Built In,
https://builtin.com/data-science/gradient-descent. Accessed 17 Dec. 2024.