Academic Writing Sample

Estimation of the Eucalyptus tree heights

Charlotte Ayrault

November 12, 2022

1 Introduction

We are looking at a way to estimate the height of eucalyptus trees. One way to solve this problem is to calculate the height from another physical characteristic of the tree. We have a database of 1,400 samples from the circumference of the tree's trunk measured at 1.3 meter from the ground and the actual height of the tree.

I have been asked to verify that the measure of the circumference is a good indicator of the height of the eucalyptus trees and to provide the equation based on 2 statistical models; the simple linear regression (SLR) and the multiple linear regression (MLR).

This problem illustrates how a simple measure can help eucalyptus farmer to identify which tree should be cut depending on the final product issued from the tree.

This academic writing is based on a work performed by myself for the Statistic course at Saclay University in May 2022.

2 Simple Linear Regression

Simple linear regression provides a means to model with a straight line relationship between two variables. In order to use this model, some conditions should be verified on the data set:

1. Homogeneity of the variance: the value of the error in our prediction doesn't change too much across the values of the independent variable.

2. Independence of observations: the data set were collected using statistically valid sampling methods. There is no hidden relationships among observations like choosing specific trees.

3. Normality: The collected data follows a normal distribution.

4. The relationship between the independent and dependent variable is linear: the line of best fit through the data points is a straight line (rather than a curve or some sort of grouping factor).

Figure 1 is a scatter of dataset.

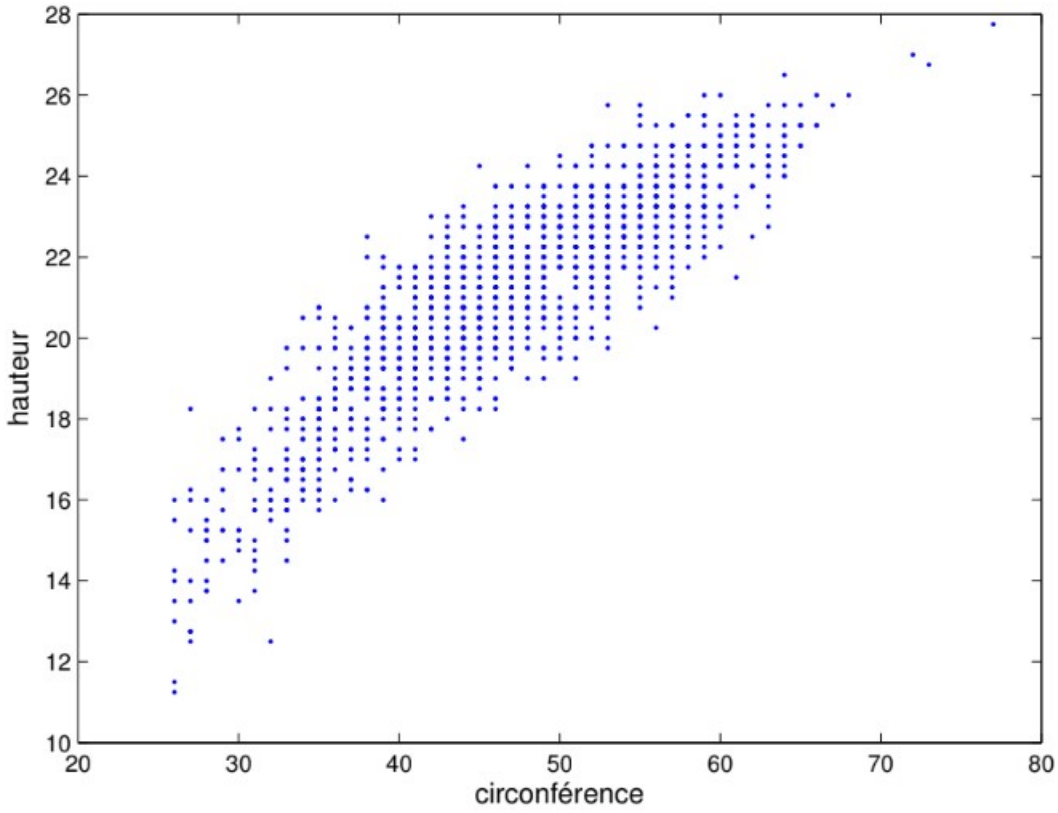

Figure 1: Data set

We can clearly see that the scatter plot is compact (no point completely outside the others), this satisfies the condition 1. It is also arranged along a straight line, this satisfies the condition 4. We can only assume that the collected data are unbias to satisfies the condition 2. Condition 3 is verified on the following figures 2 and 3. The figures trace the distribution of the circumferences and the heights and their average. It shows that the data follows a normal distribution.

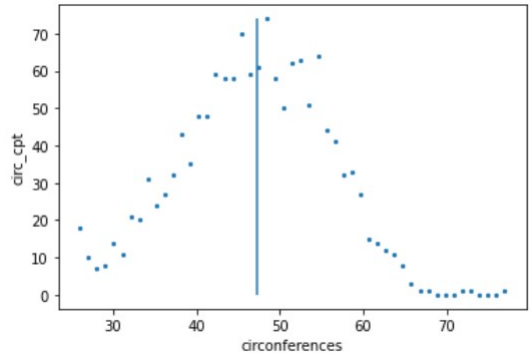

Figure 2: Circumference

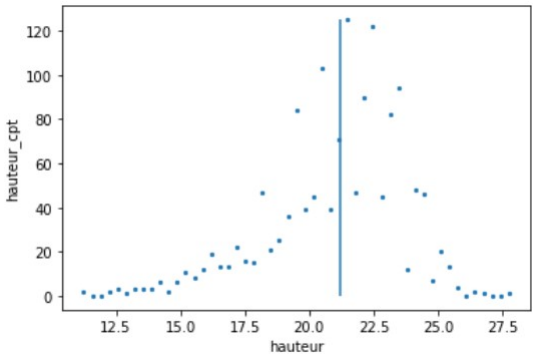

Figure 3: Height

Therefore the Simple Linear Regression can be a good candidate to model this problem as it aims to define the equation of a line.

The Simple Linear Regression model consists to suppose that a variable y can be expressed linearly in function of a variable x with a sample random variation noted ϵ. The variation is called "residual" or "noise".

$$\forall i \in 1, ..., n, y_i = \beta_1 + \beta_2 x_i + \epsilon_i$$

Variables x_i and y_i are observed data, when the value β_1 and β_2 are unknown. We are then looking for a straight line with the equation $y = \beta_1 + \beta_2 x$ that provides the relationship between x and y and minimizes ϵ in a certain way.

One way to solve the problem is to find the value of β_1 et β_2 obtained by minimizing the sum of squared residuals.

$$\varphi(\beta_1, \beta_2) = \sum_{i=1}^{n} \epsilon_i^2 = \sum_{i=1}^{n} (y_i - \beta_1 - \beta_2 x_i)^2$$

3 Calculation of β_1 and β_2

The way to minimum of the function $\varphi(\beta_1, \beta_2)$ is to calculate the derivative of the function $\varphi(\beta_1, \beta_2)$ with respect to β_1 and β_2 and then to find the values where the 2 derivatives are equal to 0.

The first derivative is

$$\frac{\partial \varphi(\beta_1, \beta_2)}{\beta_1} = \frac{\sum_{i=1}^{n} (Y_i - \beta_1 - \beta_2 x_i)^2}{\beta_1}$$

$$= \frac{\sum_{i=1}^{n} Y_i^2 - \beta_1 Y_i - \beta_2 x_i Y_i - \beta_1 Y_i + \beta_1^2 + \beta_1 \beta_2 x_i - \beta_2 x_i Y_i + \beta_2 x_i \beta_1 + \beta_2^2 x_i^2}{\beta_1}$$

$$= \sum_{i=1}^{n} -Y_i - Y_i + 2\beta_1 + \beta_2 x_i + \beta_2 x_i = 2 \sum_{i=1}^{n} -Y_i + \beta_2 x_i + \beta_1$$

The second derivative is

$$\frac{\partial \varphi(\beta_1, \beta_2)}{\beta_2} = \frac{\sum_{i=1}^{n} (Y_i - \beta_1 - \beta_2 x_i)^2}{\beta_2}$$

$$= \frac{\sum_{i=1}^{n} Y_i^2 - \beta_1 Y_i - \beta_2 x_i Y_i - \beta_1 Y_i + \beta_1^2 + \beta_1 \beta_2 x_i - \beta_2 x_i Y_i + \beta_2 x_i \beta_1 + \beta_2^2 x_i^2}{\beta_2}$$

$$= \sum_{i=1}^{n} -x_i Y_i + \beta_1 x_i - x_i Y_i + \beta_1 x_i + 2\beta_2 x_i^2 = 2 \sum_{i=1}^{n} x_i(-Y_i + \beta_2 x_i + \beta_1)$$

We are looking for the values β_1 and β_2 that nullifies the following system.

$$\begin{cases} \sum_{i=1}^{n} x_i(-Y_i + \beta_2 x_i + \beta_1) = 0 \\ \sum_{i=1}^{n} -Y_i + \beta_2 x_i + \beta_1 = 0 \end{cases}$$

$$\begin{cases} \sum_{i=1}^{n} -Y_i x_i + \sum_{i=1}^{n} \beta_2 x_i^2 + \sum_{i=1}^{n} \beta_1 x_i = 0 & (1) \\ \sum_{i=1}^{n} -Y_i + \sum_{i=1}^{n} \beta_2 x_i + \sum_{i=1}^{n} \beta_1 = 0 & (2) \end{cases}$$

$$\begin{cases} \sum_{i=1}^{n} -Y_i x_i + \beta_2 \sum_{i=1}^{n} x_i^2 + \beta_1 \sum_{i=1}^{n} x_i = 0 & (1) \\ \sum_{i=1}^{n} -Y_i + \beta_2 \sum_{i=1}^{n} x_i + n\beta_1 = 0 & (2) \end{cases}$$

We calculate $(3) = n(1) - (2)\sum_{i=1}^{n} x_i$

$$\begin{cases} n\sum_{i=1}^{n} -Y_i x_i + n\beta_2 \sum_{i=1}^{n} x_i^2 + \sum_{i=1}^{n} Y_i \sum_{i=1}^{n} x_i - \beta_2 \left(\sum_{i=1}^{n} x_i\right)^2 = 0 & (3) \\ \sum_{i=1}^{n} -Y_i + n\beta_2 \sum_{i=1}^{n} x_i + n\beta_1 = 0 & (2) \end{cases}$$

$$\begin{cases} -n\sum_{i=1}^{n} Y_i x_i + \sum_{i=1}^{n} Y_i \sum_{i=1}^{n} x_i = \beta_2 \left(\sum_{i=1}^{n} x_i\right)^2 - n\beta_2 \sum_{i=1}^{n} x_i^2 & (3) \\ \sum_{i=1}^{n} -Y_i + n\beta_2 \sum_{i=1}^{n} x_i + n\beta_1 = 0 & (2) \end{cases}$$

$$\begin{cases} \beta_2 = \frac{\sum_{i=1}^{n} Y_i \sum_{i=1}^{n} x_i - n\sum_{i=1}^{n} Y_i x_i}{\left(\sum_{i=1}^{n} x_i\right)^2 - n\sum_{i=1}^{n} x_i^2} & (3) \\ \beta_1 = \frac{1}{n}\left(\sum_{i=1}^{n} Y_i - n\beta_2 \sum_{i=1}^{n} x_i\right) & (2) \end{cases}$$

To obtain the following values.

$$\begin{cases} \beta_2 = \frac{\sum_{i=1}^{n} Y_i \sum_{i=1}^{n} x_i - n\sum_{i=1}^{n} Y_i x_i}{\left(\sum_{i=1}^{n} x_i\right)^2 - n\sum_{i=1}^{n} x_i^2} & (3) \\ \beta_1 = \frac{\sum_{i=1}^{n} x_i \sum_{i=1}^{n} x_i Y_i - \sum_{i=1}^{n} x_i^2 \sum_{i=1}^{n} Y_i}{\left(\sum_{i=1}^{n} x_i\right)^2 - n\sum_{i=1}^{n} x_i^2} & (2) \end{cases}$$

The values can be calculated using a Python program (see appendix A). I obtain $\beta_1 = 9.037475668452768$ et $\beta_2 = 0.257137855007109$ and

- Average of the ϵ_i: -3.603610217941441e-11
- Sum of squared ϵ_i : 19.492804231375466

The straight line is shown on figure 4

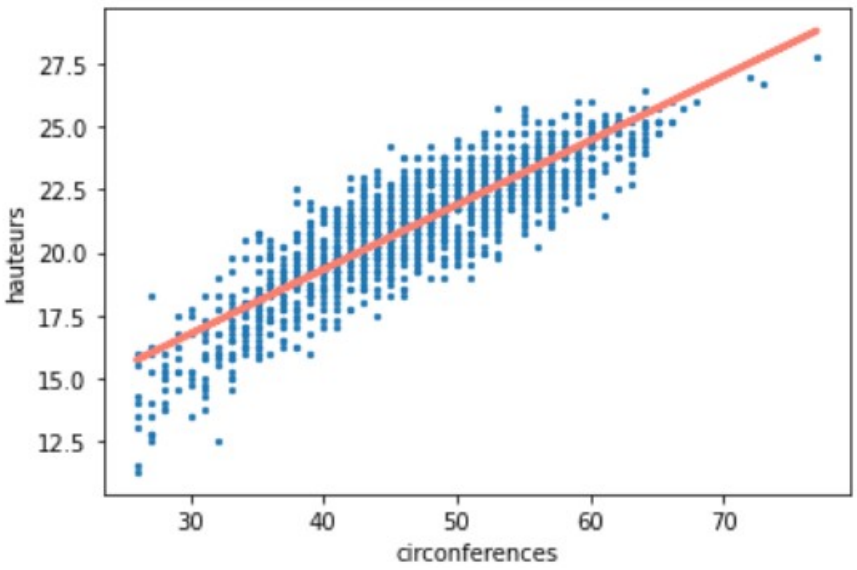

Figure 4: Simple Linear Regression

We can see that for the medium values of the circumferences the Linear model is accurate, for the low and high values of the circumferences the line over-estimates the height. Adding a

4 Multiple Linear Regression

The Multiple Linear Regression Model is used when the relation is between two or more explanatory variables and a response variable.

$$\forall i \in \{1, \ldots, n\}, Y_i = \beta_1 + \beta_2 x_2 + \beta_3 x_3 + \ldots + \beta_n x_n + \epsilon_i$$

Where the $b_i x_i$ are called the regression coefficients of the regression.

On our problem, with only have one explanatory variable; the circumference of the eucalyptus tree. So a way to apply the Multiple Linear Model is to use polynomial values of the variable to generate "new" explanatory variables.

We try to correct the model by lightly lowered both side of the straight line. Based on the Simple Linear Model, we need to find which polynomial could provide the best correction. Polynomials greater than 1 may provide a too aggressive correction. Therefore, we need to propose a polynomial less than 1. Let's try $\sqrt{x}$, as lesser value could be too aggressive too. Let try the following model.

$$\forall i \in \{1, \ldots, n\}, Y_i = \beta_1 + \beta_2 x_i + \beta_3 \sqrt{x_i} + \epsilon_i$$

The method used to calculate the β_i for the Simple Linear Model cannot be used. A more generic resolution method should be used. The Multiple Linear Regression can be rewritten using matrices.

$$Y = X\beta + \epsilon$$

With

- Y a column vector of size $n \times 1$ build as $\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}$,
- X a matrix of size $n \times 3$ build as $\begin{pmatrix} 1 & x_1 & \sqrt{x_1} \\ 1 & x_2 & \sqrt{x_2} \\ \vdots & \vdots & \vdots \\ 1 & x_n & \sqrt{x_n} \end{pmatrix}$
- $\beta = \begin{pmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \end{pmatrix}$
- ϵ column vectors of size $n \times 1$, $\begin{pmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{pmatrix}$

As for the Simple Linear Regression we try to minimize

$$\varphi(\beta) = \sum_{i=1}^{n} \epsilon_i^2 = \sum_{i=1}^{n} (y_i - \beta_1 - \beta_2 x_i - \sqrt{\beta_3})^2 = \|Y - X\beta\|^2$$

We note F the vector space based on the column of X. $F = Vect(1, x, \sqrt{x})$. A vector z belongs to F if and only if there exists β such that $z = X\beta'$. The problem to solve is to find the value β that minimize the function $\varphi(\beta)$ for the given values Y and X.

Looking for minimizing $\|Y - X\beta\|^2$, is identical than finding the point from the vector space F which is at the closest euclidian distance from Y. This is the definition of the orthogonal projection of Y on F. As $z \in F$, if and only if $z = X\beta'$, we are looking for β such that $X\beta = P_F(Y)$ where P_F is the orthogonal projection of Y on z.

To find the minima regarding β, we can derive the expression with respect to β and nullifies the result. This is done using the fact that $\sum_{i=1}^{n} (Y - X\beta)^2 = (Y - X\beta)^t(Y - X\beta)$ where X^t is the transpose on X.

$$(Y - X\beta)^t(Y - X\beta) = (Y^t - \beta^t X^t)(Y - X\beta) = Y^tY - Y^tX\beta - \beta^tX^tY + \beta^tX^tX\beta$$

et

$$\frac{\partial (Y - X\beta)^t(Y - X\beta)}{\partial \beta} = -Y^tX + \beta^tX^tX$$

We are looking for β such that

$$-Y^tX + \hat{\beta}^tX^tX = 0$$

$$\beta^tX^tX)^t = (-Y^tX)^t$$

So

$$\beta = (X^tX)^{-1}X^tY$$

The values can be calculated using a Python program (see appendix B)

$$\beta = \begin{pmatrix} -24.35200327 \\ -0.48294547 \\ 9.98688814 \end{pmatrix}$$

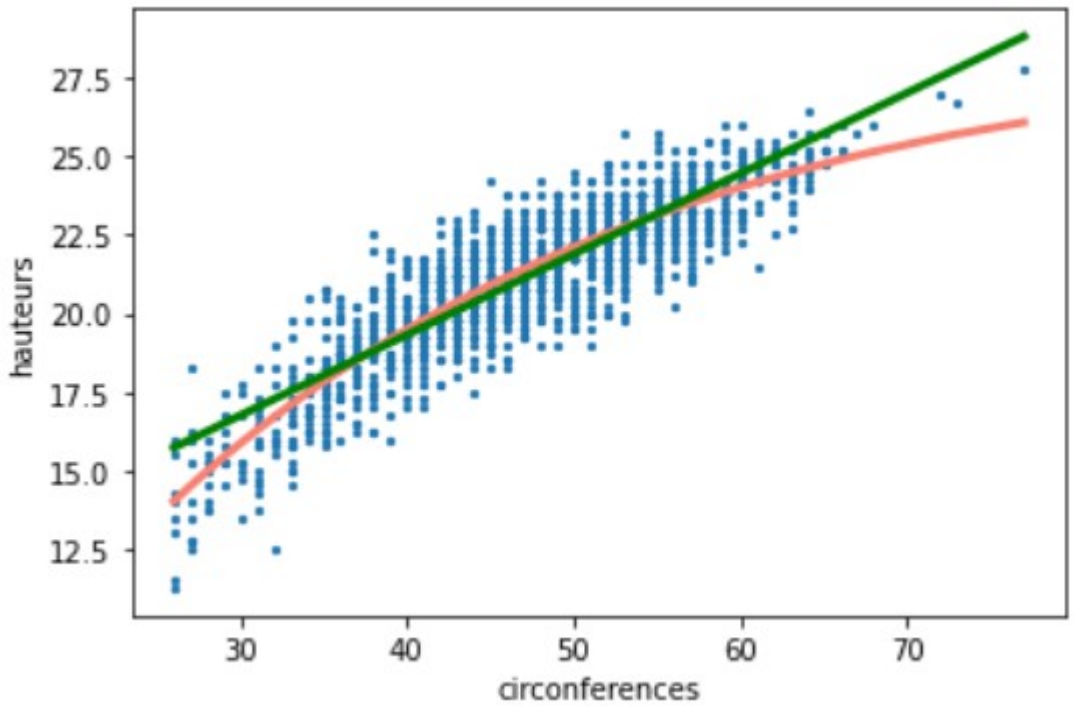

Figure 5: Regression multiple

- Average of the ϵ_i : 1.0692449957862278e-13
- Sum of the squared ϵ_i : 19.32298986873724

The new values are closed but better than the those of the Simple Linear Regression.

The Python program also calculates the values of β for the Simple Linear Regression using $X = \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{pmatrix}$. I found the same values of β than for the Simple Linear Regression.

5 Appendices. Python code

5.1 Simple Linear Regression

```
import csv
import matplotlib.pyplot as plt
import numpy as np
from scipy.stats import t
import os

# lecture fichier Eucalypcus
file = open(os.getcwd()+"\\Eucaliptus.csv")
csvreader = csv.reader(file)
next(csvreader)

circs = np.empty([0])
hauteurs = np.empty([0])

for row in csvreader:
    c = float(row[1])
    h = float(row[2])
    circs = np.append(circs,c)
    hauteurs = np.append(hauteurs,h)

max_c = np.amax(circs)
min_c = np.amin(circs)
max_h = np.amax(hauteurs)
min_h = np.amin(hauteurs)

pas = 50

# circs
circs_cpt = np.zeros(pas+1)
diff_c = max_c - min_c
axis_x = min_c + np.arange(0,pas+1)*diff_c/pas
for c in circs:
    index = int((c - min_c)*pas/diff_c)
    circs_cpt[index] += 1

plt.scatter(axis_x, circs_cpt,s=5)
plt.vlines(x = np.mean(circs), ymin = 0, ymax = np.amax(circs_cpt))
plt.ylabel("circ_cpt");
plt.xlabel("circonferences")
plt.show()
```

```
# hauteurs
hauteurs_cpt = np.zeros(pas+1)
diff_h = max_h - min_h
axis_h = min_h + np.arange(0,pas+1)*diff_h/pas
for h in hauteurs:
    index = int((h - min_h)*pas/diff_h)
    hauteurs_cpt[index] += 1

plt.scatter(axis_h, hauteurs_cpt,s=5)
plt.vlines(x = np.mean(hauteurs), ymin = 0, ymax = np.amax(hauteurs_cpt))
plt.ylabel("hauteur_cpt");
plt.xlabel("hauteur")
plt.show()

# Calcul des betas
xi = 0.0
yi = 0.0
xi2 = 0.0
xiyi= 0.0
circs = np.empty([0])
hauteurs = np.empty([0])

for row in csvreader:
    c = float(row[1])
    h = float(row[2])
    xi += c
    yi += h
    xiyi += c*h
    xi2 += c**2
    circs = np.append(circs,c)
    hauteurs = np.append(hauteurs,h)

    n = circs.size
    beta1 = (xi2*yi - xi*xiyi)/(n*xi2-xi**2)
    beta2 = (n*xiyi - xi*yi)/(n*xi2-xi**2)
    print(f"beta1: {beta1}, beta2: {beta2}")

    y = beta1+beta2*circs

    plt.scatter(circs, hauteurs,s=5)
    plt.plot(circs, y, color='salmon', linewidth=3)
    plt.xlabel("circonferences");
    plt.ylabel("hauteurs")
    plt.show()

    print(f"Moyenne de espilon: {np.sum(y-hauteurs)/n}")
```

5.2 Multiple Linear Regression

```
    # lecture fichier Eucalypcus
    file = open(os.getcwd()+"\\Eucaliptus.csv")
    csvreader = csv.reader(file)
    next(csvreader)

    circs = np.empty([0,3], float)
    circs1 = np.empty([0,2], float)
    circs2 = np.empty([0,5], float)
    hauteurs = np.empty([0])

    for row in csvreader:
        c = float(row[1])
        h = float(row[2])
        circs  = np.append(circs,[(1.0,c,np.sqrt(c))],axis=0)      # regression
        circs1 = np.append(circs1,[(1.0,c)],axis=0)                # regression
        hauteurs = np.append(hauteurs,h)

    file.close()

def calcule_beta(X, Y):
    XT = X.transpose()
    A = np.matmul(XT,X)
    A_inv = np.linalg.inv(A)
    B = np.matmul(XT,Y)
    beta = np.matmul(A_inv,B)
    return beta

print(f"Beta: {calcule_beta(circs, hauteurs)}")
print(f"Beta: {calcule_beta(circs1, hauteurs)}") # meme resultats que la regres

beta = calcule_beta(circs, hauteurs)
# Tri par hauteurs pour tracer proprement la courbe
circs_sort = circs[circs[:, 1].argsort()]
y = beta[0]+beta[1]*circs_sort[:,[1]]+beta[2]*circs_sort[:,[2]]

beta1 = calcule_beta(circs1, hauteurs)
print(f"Beta multiple: {beta}")

beta1 = calcule_beta(circs1, hauteurs)
# Tri par hauteurs pour tracer proprement la courbe
circs1_sort = circs1[circs1[:, 1].argsort()]
y1 = beta1[0]+beta1[1]*circs1_sort[:,[1]]
print(f"Beta simple: {beta1}") # meme resultats que la regression lineaire simp
```

```
plt.scatter(circs[:,[1]], hauteurs,s=5)
plt.plot(circs_sort[:,[1]], y, color='salmon', linewidth=2)   # regression line
plt.plot(circs1_sort[:,[1]], y1, color='green', linewidth=2)  # regression line
plt.xlabel("circonferences")
plt.ylabel("hauteurs")
plt.show()

res = 0
for i in range(hauteurs.size):
    h = beta[0] + beta[1]*circs[i,1] + beta[2]*circs[i,2]
    res = res + np.power(hauteurs[i] - h,2)
res1 = 0
for i in range(hauteurs.size):
    h1 = beta1[0] + beta1[1]*circs1[i,1]
    res1 = res1 + np.power(hauteurs[i] - h1,2)

print(f"Moyenne de espilon : {(np.sum(y)-np.sum(hauteurs))/n}")
print(f"risque quadratique : {h}")
print(f"Moyenne de espilon : {(np.sum(y1)-np.sum(hauteurs))/n}")
print(f"risque quadratique : {h1}")
```