

Rapport sur le TP de probabilité et statistiques

Wassim Saidane, Aurélien Authier

Table des matières

1	Première partie : Régression linéaire	1
1.1	Régression Linéaire simple	1
1.2	Régression linéaire et descente de gradient	5

Introduction

Dans ce TP nous allons modéliser différents problèmes de probabilité et statistique en python

1 Première partie : Régression linéaire

Cette partie correspond au fichier AuthierSaidanePart1.py

1.1 Régression Linéaire simple

Soit le modèle linéaire suivant :

$$Y_1 = \beta_0 + \beta_1 x_i$$

Avant de calculer le coefficient de régression nous avons créé une fonction moyenne.

```
1     def moyenne(tab):
2         return sum(tab)/len(tab)
3
```

Ainsi nous pouvons faire la moyenne de différentes valeurs. Une fois la fonction moyenne codée on peut faire la fonction les coefficients de régressions. Voici la formule du coefficient de régression :

$$a = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})^2}$$

$$b = \bar{y} - a\bar{x}$$

En python cela se traduit par :

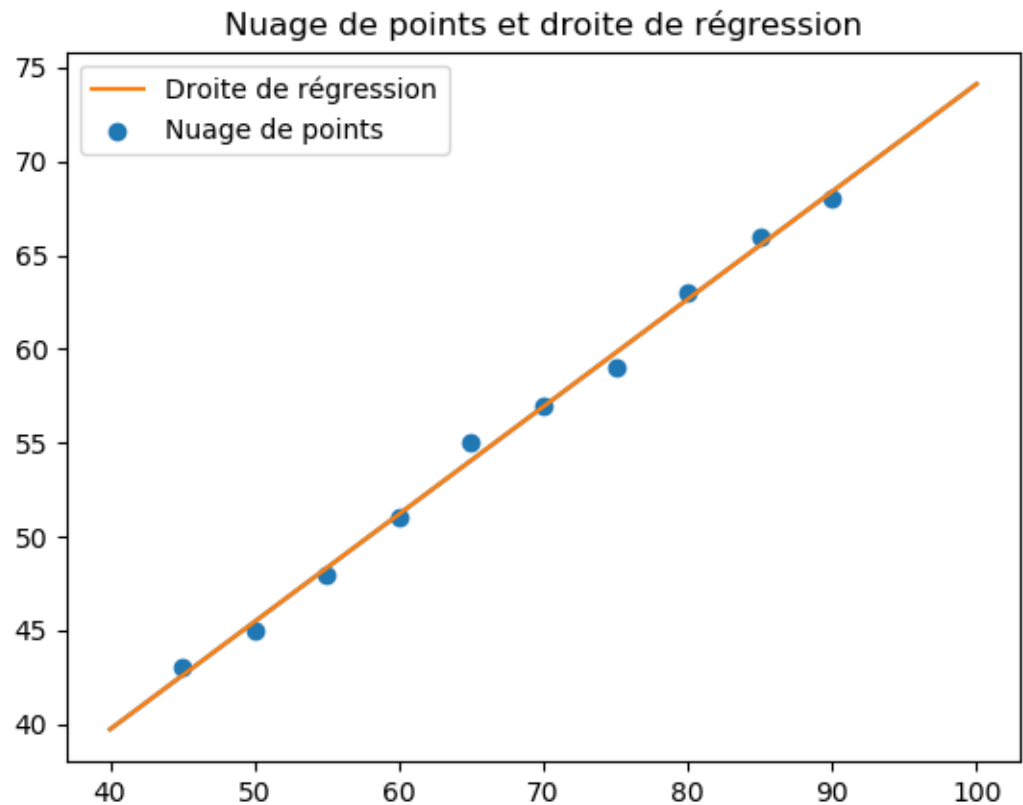
```
1     def coef_regression(tab1, tab2):
2         beta_1 = 0
3         temp1 = 0
4         beta_0 = 0
5         temp2 = 0
6         for i in range(0, len(tab1)):
7             temp1 += (tab1[i]-moyenne(tab1))*(tab2[i]-moyenne(
8                 tab2))
9             temp2 += (tab1[i]-moyenne(tab1))**2
10        beta_1 = temp1/temp2
11        beta_0 = moyenne(tab2)-beta_1*moyenne(tab1)
12        return (beta_0, beta_1)
```

Les deux tableaux en paramètres de notre fonction sont les échantillons du sujet.
beta_0 correspond à b et beta_1 correspond à a . On obtient :
b=0.5733333333333334
a=16.799999999999997

Pour la représentation graphique nous avons utilisé l'outil pyplot de la bibliothèque matplotlib.

```
1     coefs = coef_regression(x_i, y_i)
2     x = np.linspace(40, 100)
3     plt.plot(x, coefs[0]+coefs[1]*x)
4     plt.scatter(x_i, y_i, label="Nuage de points")
5     p = np.polyfit(x_i, y_i, 1)
6     plt.plot(x, p[1]+p[0]*x, label="Droite de regression")
7     plt.legend()
8     plt.title("Nuage de points et droite de regression")
9     plt.show()
10
```

On obtient le graphique suivant :



Soit le modèle vectoriel suivant :

$$y = A\beta$$

Pour pouvoir faire des opérations sur les matrices nous avons utilisé la bibliothèque numpy.

Nous avons codé la formule $(A^T A)^{-1} A^T y$.

Tout d'abord nous avons implémenté deux petites fonctions permettant de déterminer A , A^T et Y .

```
1 def matA(x):
2     vect = np.ones(len(x))
3     Atranspose = np.vstack((vect, x))
4     A = np.transpose(Atranspose)
5     return (A,Atranspose)
6 def matY(y):
7     return np.matrix(y_i).transpose()
8
```

Ensuite en deux opérations nous avons calculé $A^T A$ et $(A^T A)^{-1}$.

```
1 A_transposeA = np.dot(A_transpose, A)
2 inv_ATA = np.linalg.inv(A_transposeA)
3
```

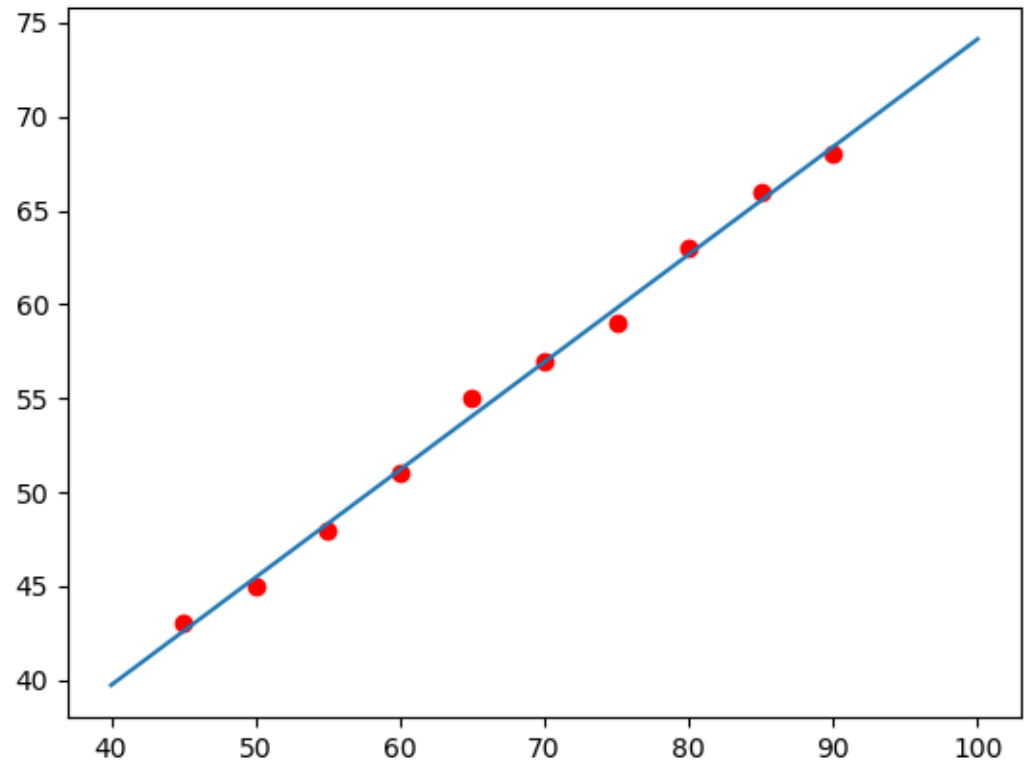
Pour finir nous avons implémenté la formule demandée :

```
1 def matBeta(inv,Atranspose,Y):
2     B1=np.dot(inv,Atranspose)
3     B2=np.dot(B1,Y)
4     return np.asarray(B2).reshape(-1)
5
```

On obtient ainsi Beta : [16.8 0.57333333]

Ainsi que la graphique suivante : (voir page suivante)

Nuage de points et droite de régression



1.2 Régression linéaire et descente de gradient