-

TD1 - Le modèle de régression linéaire

In [1]:
```
import numpy as np
from scipy import stats
import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sns
sns.set_style(style='ticks')

# Machine learning
from sklearn import linear_model
```

Rappels de cours

Nous ne rappelons pas ici le cas du modèle de régression simple. Celui-ci étant simplement un cas particulier du modèle de régression mult

Cas du modèle de régression multiple

- Le modèle sous la forme matricielle s'écrit : $Y = Xa + \epsilon$.
- L'estimateur des MCO est donné par $\hat{a} = (X'X)^{-1}X'Y$.
- La matricce de variance-covariance des coefficients est sdonnées par : $\hat{\Omega}_{\hat{a}} = \hat{\sigma}^2_{\epsilon}(X'X)^{-1}$

Exercice 1

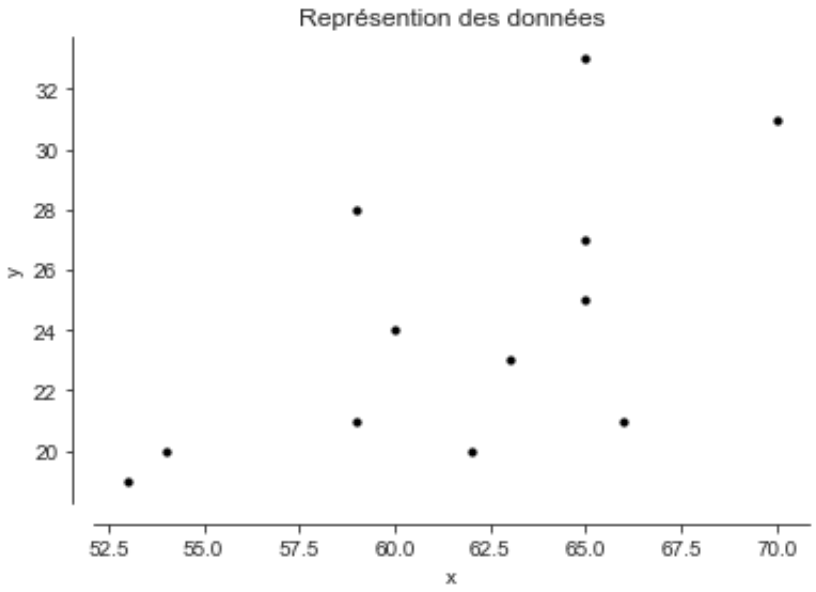
In [26]:
```
x = np.array([54, 53, 59, 66, 63, 62, 65, 60, 59, 65, 70, 65]).reshap
e(12, 1)
y = np.array([20, 19, 21, 21, 23, 20, 25, 24, 28, 27, 31, 33]).reshap
e(12, 1)
```

Question 1

Tracer le graphique y_t en fonction de x_t

In [27]:
```
plt.figure()
plt.scatter(x, y, color='black', s=10)
sns.despine(offset=10)
plt.xlabel('x')
plt.ylabel('y')
plt.title('Représention des données')


plt.show()
```



À priori, on peut utiliser une régression linéaire.

Question 2

Énoncé : Qu'est-ce que représente le terme d'erreur dans le modèle ? Quelles sont les hypothèses faites dans le cadre des Moindres Carrés Ordinaires ?

Hypothèses sur les erreurs

- espérance nul
- variance constante (homoscédasticité)
- pas d'autocorrélation (erreurs idpt entre elles)
- exogénéité (erreur idpt des variables)
- erreurs suivent une loi normale

Question 3

$$\hat{\beta} = (X'X)^{-1}X'y$$

In [28]:
```
X = np.concatenate((np.repeat(1, 12).reshape(12, 1), x), axis=1)
```

In [29]:
```
np.dot(np.dot(np.linalg.inv(np.dot(X.transpose(), X)), X.transpose())
,y)
```

Out[29]:
```
array([[-11.01671225],
       [  0.57247037]])
```

Question 4

Énoncé : Estimé la variance résiduelle et les écarts types des estimateurs a_0 et a_1.

In [30]:
```
def f(x):
    return (-11.01671225 + x*0.57247037)
```

Variance résiduelle

In [31]:
```
residu = y - f(x)
var_res = sum(residu**2)/(len(y)-2)
var_res
```

Out[31]:
```
array([14.0788818])
```

In [32]:
```
a_0 = var_res * (1/len(y)+ np.mean(x)**2/sum((x-np.mean(x))**2))
a_0
```

Out[32]:
```
array([196.9203914])
```

In [33]:
```
x.transpose()
```

Out[33]:
```
array([[54, 53, 59, 66, 63, 62, 65, 60, 59, 65, 70, 65]])
```

In [34]:
```
x - np.mean(x)
```

Out[34]:
```
array([[-7.75],
       [-8.75],
       [-2.75],
       [ 4.25],
       [ 1.25],
       [ 0.25],
       [ 3.25],
       [-1.75],
       [-2.75],
       [ 3.25],
       [ 8.25],
       [ 3.25]])
```

In [35]:
```
a_1 = var_res/(sum((x-np.mean(x))**2))
a_1
```

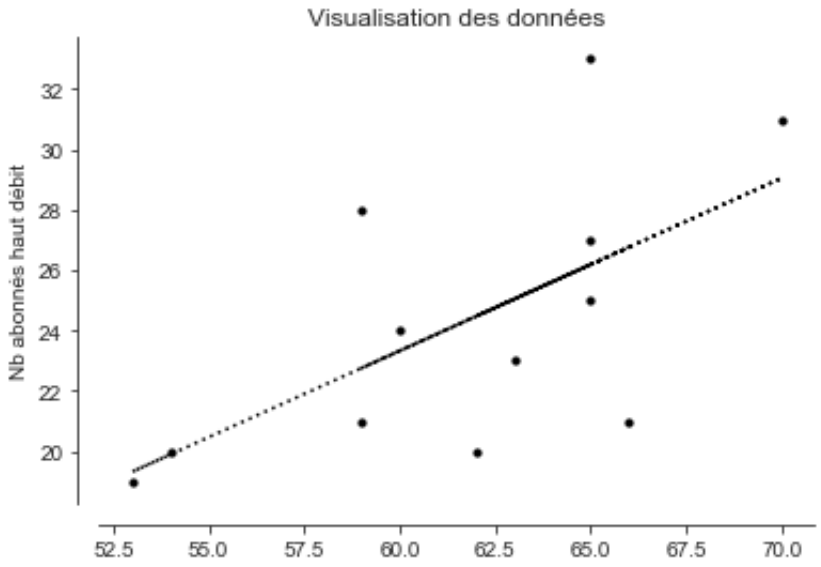
Out[35]:
```
array([0.05133594])
```

Annexe

In [36]:
```
reg = linear_model.LinearRegression()
reg.fit(x, y)
reg.score(x, y)
print('Coef =',reg.coef_)
print('Intercept =',reg.intercept_)
```

```
Coef = [[0.57247037]]
Intercept = [-11.01671225]
```

In [13]:
```
plt.figure()
plt.scatter(x,y, color='black', s=10)
plt.plot(x, reg.predict(x), ls=':', c='black')
sns.despine(offset=10)
# plt.xticks(np.arange(1,13))
plt.title('Visualisation des données')
plt.ylabel('Nb abonnés haut débit')
# plt.savefig('test.svg')
plt.show()
```



In [45]:
```
model = sm.OLS(y, sm.add_constant(x))
model = model.fit()
print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.390
Model:                            OLS   Adj. R-squared:                  0.329
Method:                 Least Squares   F-statistic:                     6.384
Date:                Thu, 01 Oct 2020   Prob (F-statistic):             0.0300
Time:                        14:20:32   Log-Likelihood:                -31.801
No. Observations:                  12   AIC:                             67.60
Df Residuals:                      10   BIC:                             68.57
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const        -11.0167     14.033     -0.785      0.451     -42.284      20.250
x1             0.5725      0.227      2.527      0.030       0.068       1.077
==============================================================================
Omnibus:                        0.617   Durbin-Watson:                   0.836
Prob(Omnibus):                  0.735   Jarque-Bera (JB):                0.296
Skew:                           0.353   Prob(JB):                        0.863
Kurtosis:                       2.695   Cond. No.                         803.
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors
is correctly specified.
```

In [14]:
```
def f(x):
    return (-11.01671225 + x*0.57247037)
```

Variance résiduelle

In [15]:
```
residu = y - f(x)
var_res = sum(residu**2)/(len(y)-2)
var_res
```

Out[15]:
```
array([14.0788818])
```

In [103]:
```
a_0 = var_res * (1/len(y)+ np.mean(x)**2/sum((x-np.mean(x))**2))
a_0
```

Out[103]:
```
array([196.9203914])
```

In [147]:
```
a_1 = var_res**2/(sum((x-np.mean(x))**2))
a_1
```

Out[147]:
```
array([0.07369058])
```

Exercice 2

Soit la variable à expliquer y_t et deux variables explicatives $x_{1,t}$ et $x_{2,t}$ observées sur 10 périodes.\ Les valeurs des variables sont reportées dans le tableau ci-dessous.

On cherche

$$y_t = a_0 + a_1 x_{1,t} + a_2 x_{2,t} + \epsilon_t$$

In [117]:
```
y = np.array([12, 21, 24, 24, 13, 17, 21, 26, 31, 30]).reshape(10, 1)
X = np.array([np.repeat(1, 10),[7, 9, 11, 12, 7, 9, 12, 14, 19, 21],
[48, 40, 18, 28, 40, 32, 31, 24, 22, 25]]).transpose()
```

In [118]:
```
X
```

Out[118]:
```
array([[ 1,  7, 48],
       [ 1,  9, 40],
       [ 1, 11, 18],
       [ 1, 12, 28],
       [ 1,  7, 40],
       [ 1,  9, 32],
       [ 1, 12, 31],
       [ 1, 14, 24],
       [ 1, 19, 22],
       [ 1, 21, 25]])
```

Connaître la formule de :

$$\hat{\beta} = (X'X)^{-1}X'y$$

In [119]:
```
np.dot(np.dot(np.linalg.inv(np.dot(X.transpose(), X)), X.transpose())
,y)
```

Out[119]:
```
array([[18.8720233 ],
       [ 0.90185438],
       [-0.255989  ]])
```

In [123]:
```
def f(x1, x2):
    return (18.8720233 + x1*0.90185438 -0.255989*x2)
```

Variance résiduelle

In [149]:
```
residu = y.transpose()[0] - f(X[:,1] ,X[:, 2])
var_res = sum(residu**2)/(len(y)-3)
print('var_res',var_res)
```

```
var_res 4.495513587420975
```

$$\hat{\Omega}_{\hat{a}} = var_{residuelle} * (X'X)^{-1}$$

In [154]:
```
np.linalg.inv(np.dot(X.transpose(), X))
```

Out[154]:
```
array([[ 6.22459242e+00, -2.15810688e-01, -1.14067633e-01],
       [-2.15810688e-01,  9.44283029e-03,  3.29715718e-03],
       [-1.14067633e-01,  3.29715718e-03,  2.40818284e-03]])
```

Exercice 3

On utilise le modèle suivant : On cherche

$$Conso_t = a_0 + a_1 * Essai_t + a_2 * Temp_t + \epsilon_t$$

In [155]:
```
y = np.array([260, 300, 350, 310, 190, 380, 240, 210, 370, 400]).resh
ape(10, 1)
X = np.array([np.repeat(1, 10),[4, 5, 11, 4, 2, 9, 0, 2, 10, 7], [17,
20, 17, 18, 14, 16, 13, 13, 15, 16]]).transpose()
```

In [156]:
```
X
```

Out[156]:
```
array([[ 1,  4, 17],
       [ 1,  5, 20],
       [ 1, 11, 17],
       [ 1,  4, 18],
       [ 1,  2, 14],
       [ 1,  9, 16],
       [ 1,  0, 13],
       [ 1,  2, 13],
       [ 1, 10, 15],
       [ 1,  7, 16]])
```

Connaître la formule de :

$$\hat{\beta} = (X'X)^{-1}X'y$$

In [157]:
```
np.dot(np.dot(np.linalg.inv(np.dot(X.transpose(), X)), X.transpose())
,y)
```

Out[157]:
```
array([[151.79535468],
       [ 15.94266583],
       [  3.96944968]])
```

In [158]:
```
def f(x1, x2):
    return (18.8720233 + x1*0.90185438 -0.255989*x2)
```

Variance résiduelle

In [159]:
```
residu = y.transpose()[0] - f(X[:,1] ,X[:, 2])
var_res = sum(residu**2)/(len(y)-3)
print('var_res',var_res)
```

```
var_res 119625.56969668434
```

$$\hat{\Omega}_{\hat{a}} = var_{residuelle} * (X'X)^{-1}$$

In [160]:
```
np.linalg.inv(np.dot(X.transpose(), X))
```

Out[160]:
```
array([[ 5.93427495,  0.04375392, -0.38179535],
       [ 0.04375392,  0.00939527, -0.00594267],
       [-0.38179535, -0.00594267,  0.02603055]])
```

In [17]:
```
import pandas as pd
import numpy as np
from sklearn import datasets, linear_model
from sklearn.linear_model import LinearRegression
import statsmodels.api as sm
from scipy import stats

diabetes = datasets.load_diabetes()
X = diabetes.data
y = diabetes.target

X2 = sm.add_constant(X)
est = sm.OLS(y, X2)
est2 = est.fit()
print(est2.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:
0.518
Model:                            OLS   Adj. R-squared:
0.507
Method:                 Least Squares   F-statistic:
46.27
Date:                Thu, 01 Oct 2020   Prob (F-statistic):
3.83e-62
Time:                        14:06:41   Log-Likelihood:
-2386.0
No. Observations:                 442   AIC:
4794.
Df Residuals:                     431   BIC:
4839.
Df Model:                          10
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025
0.975]
------------------------------------------------------------------------------
const        152.1335      2.576     59.061      0.000     147.071
157.196
x1           -10.0122     59.749     -0.168      0.867    -127.448
107.424
x2          -239.8191     61.222     -3.917      0.000    -360.151
-119.488
x3           519.8398     66.534      7.813      0.000     389.069
650.610
x4           324.3904     65.422      4.958      0.000     195.805
452.976
x5          -792.1842    416.684     -1.901      0.058   -1611.169
26.801
x6           476.7458    339.035      1.406      0.160    -189.621
```

```
1143.113
x7            101.0446    212.533      0.475      0.635    -316.685
518.774
x8            177.0642    161.476      1.097      0.273    -140.313
494.442
x9            751.2793    171.902      4.370      0.000     413.409
1089.150
x10            67.6254     65.984      1.025      0.306     -62.065
197.316
==============================================================================
==========
Omnibus:                        1.506   Durbin-Watson:
2.029
Prob(Omnibus):                  0.471   Jarque-Bera (JB):
1.404
Skew:                           0.017   Prob(JB):
0.496
Kurtosis:                       2.726   Cond. No.
227.
==============================================================================
==========

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors
is correctly specified.
```

In [22]:
```
X
```

Out[22]:
```
array([[ 0.03807591,  0.05068012,  0.06169621, ..., -0.00259226,
         0.01990842, -0.01764613],
       [-0.00188202, -0.04464164, -0.05147406, ..., -0.03949338,
        -0.06832974, -0.09220405],
       [ 0.08529891,  0.05068012,  0.04445121, ..., -0.00259226,
         0.00286377, -0.02593034],
       ...,
       [ 0.04170844,  0.05068012, -0.01590626, ..., -0.01107952,
        -0.04687948,  0.01549073],
       [-0.04547248, -0.04464164,  0.03906215, ...,  0.02655962,
         0.04452837, -0.02593034],
       [-0.04547248, -0.04464164, -0.0730303 , ..., -0.03949338,
        -0.00421986,  0.00306441]])
```

In [ ]:
```

```