Chapitre # (NUM) 14 Statistiques

Résumé & Plan

L'objectif de ce TP est d'appliquer les scripts Python vus en cours dans des exercices, il servira également de TD associé au cours de Statistiques.

Cadre

On suppose dans tout le TP qu'une série statistique est représentée par une liste comportant l'ensemble des observations.

1. STATISTIQUES UNIVARIÉES

1.1. Principales fonctions

Pour avoir la liste des modalités, il suffit de créer une nouvelle liste sans doublon.

(Modalités)

```
def sans_doublon(L):
    """
    renvoie la liste des éléments de L, chaque élément \
    ↪ apparaissant une unique fois
    """
    M = []
    for x in L:
        if x not in M:
            M.append(x)
    return M
```

On peut également transformer la série statistique de départ en dictionnaire de clefs les modalités, et valeurs l'effectif associé à chaque modalité. C'est la fonction `dico_occur` déjà rencontrée.

(Dictionnaire des effectifs)

```
def dico_occur(L):
    D = {}
    for x in L:
        if x not in D:
            D[x] = 1
        else :
            D[x] += 1
    return D
```

On peut également revenir à une liste d'observations si on le souhaite.

(Dictionnaire des effectifs vers liste)

```
def dico_occur_vers_liste(D):
    L = []
    for x in D:
        # x est une modalité, que l'on duplique autant de \
        ↪ fois que nécessaire
        eff_x = D[x]
        for _ in range(eff_x):
            L.append(x)
    return L
```

La fonction de calcul de moyenne s'appuie notamment sur celle qui calcule la somme.

Lycée Camille GUÉRIN – Poitiers

(Moyenne)

```python
def moyenne(L):
    """
    Renvoie la moyenne des éléments d'une liste
    """
    S = 0
    for x in L:
        S += x
    return S/len(L)
```

Si l'on préfère, on peut aussi calculer directement la moyenne à l'aide du dictionnaire des effectifs : dans ce cas, on pondère par l'effectif associé.

(Moyenne avec effectifs)

```python
def moyenne_avec_eff(D):
    """
    Renvoie la moyenne d'une série associée au dictionnaire \
    ↪ des effectifs
    D
    """
    S = 0
    N = 0 # nombre d'éléments de la série
    for x in D:
        eff_x = D[x]
        S += x*eff_x
        N += eff_x
    return S/N
```

Pour la variance, on utilise généralement la version König-Huygens de la formule : $\mathbb{V}_x = \overline{x^2} - \overline{x}^2$ si x désigne une série statistique.

(Variance)

```python
def variance(L):
    """
    Renvoie la variance, version KH
    """
    S2 = 0
    for x in L:
        S2 += x**2
    return S2/len(L) - moyenne(L)**2
```

Voyons quelques exemples d'exécutions.

Exemple 1 On code la série

19	26	23	20	22	24	20	24
22	20	21	19	21	22	19	20
21	21	22	21	23	22	21	24
25	23	22	19	20	26	24	25
23	26	25	25	21	22	25	24
23	22	24	24	25	23	25	22

avec la liste :

```
>>> L = [19, 26, 23, 20, 22, 24, 20, 24, 22, 20, 21, 19, 21, \
↪ 22, 19, 20, 21, 21, 22, 21, 23, 22, 21, 24, 25, 23, 22, 19, \
↪ 20, 26, 24, 25, 23, 26, 25, 25, 21, 22, 25, 24, 23, 22, 24, \
↪ 24, 25, 23, 25, 22]
>>> modalites(L)
[19, 26, 23, 20, 22, 24, 21, 25]
>>> D = dico_occur(L)
>>> D
{19: 4, 26: 3, 23: 6, 20: 5, 22: 9, 24: 7, 21: 7, 25: 7}
>>> dico_occur_vers_liste(D)
[19, 19, 19, 19, 26, 26, 26, 23, 23, 23, 23, 23, 23, 20, 20, \
↪ 20, 20, 20, 22, 22, 22, 22, 22, 22, 22, 22, 22, 24, 24, 24, \
↪ 24, 24, 24, 24, 21, 21, 21, 21, 21, 21, 21, 25, 25, 25, 25, \
↪ 25, 25, 25]
>>> moyenne(L)
22.5
>>> variance(L)
4.083333333333314
>>> moyenne_avec_eff(D) # on retrouve bien le même résultat
22.5
```

On peut également, après recherche du minimum et du maximum, renvoyer l'étendue de la série.

(Étendue d'une série)

```
def etendue(L):
    """
    Renvoie l'étendue de la série statistique des éléments de L
    """
    mini = L[0]
    maxi = L[0]
    for x in L[1:]:
        if x < mini:
            mini = x
        elif x > maxi:
            maxi = x
    return maxi - mini
```

Pour calculer la médiane (ou les 3 quartiles), il faut au préalable trier la liste avec par exemple la méthode du tri rapide.

(Tri rapide)

```
def tri_rapide_rec(L):
    """
    Trie la liste L selon le tri rapide (Non en place)
    """
    if len(L) <= 1:
        return L
    else:
        pivot = L[0]
        inf_pivot = []
        sup_pivot = []
        for x in L[1:]:
            if x < pivot:
                inf_pivot.append(x)
            else:
                sup_pivot.append(x)
        return tri_rapide_rec(inf_pivot) + [pivot] + \
        ↪ tri_rapide_rec(sup_pivot)
```

(Médiane)

```
def mediane(L):
    """
    Cherche la médiane d'une liste, après tri rapide des \
    ↪ observations
    """
    L_tri = tri_rapide_rec(L)
    n = len(L)
    if n % 2 == 1:
        # Nombre impair d'observations
        return L_tri[n//2]
    else:
        # Nombre pair d'observations
        return (L_tri[n//2-1] + L_tri[n//2])/2
```

(Quartiles)

```
def quartiles(L):
    """
    renvoie Q1, Q2, Q3, après tri rapide des observations
    """
    L_tri = tri_rapide_rec(L)
    n = len(L)
    if n % 2 != 0:
        # Nombre impair d'observations
        Q2 = L_tri[n//2]
    else:
        # Nombre pair d'observations
        Q2 = (L_tri[n//2-1] + L_tri[n//2])/2
    if n % 4 != 0:
        # Nombre non multiple de 4 d'observations
        Q1 = L_tri[n//4]
        Q3 = L_tri[(3*n)//4]
    else:
        # Nombre multiple de 4 d'observations
        Q1 = L_tri[n//4-1]
        Q3 = L_tri[(3*n)//4-1]
    return Q1, Q2, Q3
```

(Histogramme et diagramme en bâtons) Les deux commandes principales (rappelées en cas de besoin) sont les suivantes :

- `plt.bar(X, H)` pour les diagrammes en bâtons : `X` désigne la liste des modalités, et `H` la liste des hauteurs associée (effectifs ou modalités).
- `plt.hist(L, n)` pour les histogrammes : `L` désigne la liste des observations, et `n` le nombre de classes souhaitées. Le regroupement en classes des données est alors fait tout seul par la commande.

Exercice 1 | Effet d'un médicament Solution Un médecin effectue des recherches sur l'efficacité d'un nouveau béta-bloquant. Cette famille de médicaments est destinée à diminuer le rythme cardiaque des malades atteints de tachycardie (pouls supérieur à 60 battements par minute). Il a donc séparé les malades en 2 groupes : le groupe A reçoit le traitement d'un nouveau médicament, et le groupe B reçoit un placebo. Voici les résultats :

- A : 74 – 91 – 91 – 84 – 95 – 93 – 95 – 102 – 81 – 116 – 88 – 95
- B : 94 – 95 – 113 – 95 – 104 – 113 – 94 – 144 – 105 – 153

1. Calculer à la main la médiane des deux séries.

2. Coder ces deux séries à l'aide de deux listes. En déduire pour chaque série : l'étendue, la médiane, les valeurs extrêmes, le premier quartile et le troisième quartile. Complétez le tableau ci-dessous.

Étendue	Médiane	Min	Max	Q_1	Q_3

3. Construire le diagramme de Tukey de ces deux séries. L'effet du médicament semble-t-il satisfaisant ?

4. Construire l'histogramme de chaque médicament, par exemple pour `n = 10` classes.

Exercice 2 | Solution Une entreprise souhaite étudier la consommation de cigarettes chez ses salariés qui fument.

Nb cigarettes par jour	5	10	15	20	25	30	35	40
Effectifs	8	17	12	8	5	3	2	1

1. Coder ce tableau à l'aide d'un dictionnaire des effectifs.

2. En déduire la médiane, la moyenne, l'étendue. *On pourra donc, si nécessaire, revenir à une liste complète d'observations plutôt qu'un dictionnaire au moyen de* `dico_occur_vers_liste`.

Médiane	Moyenne	Étendue

3. On souhaite ici calculer le mode de la série, *i.e.* la modalité d'effectif maximal.

3.1) Écrire une fonction `mode(L)` qui contient une série statistique sous forme de liste et qui retourne son mode. *On pourra utiliser le dictionnaire des effectifs.*

3.2) En déduire le mode de la série étudiée.

2. STATISTIQUES BIVARIÉES

2.1. Principales fonctions

En utilisant directement les définitions de chaque quantité, on en déduit les fonctions associées ci-dessous.

(Covariance)

```
def covariance(L, M):
    """
    Renvoie la covariance des deux séries
    """
    Prod = [L[i]*M[i] for i in range(len(M))]
    return moyenne(Prod) - moyenne(L)*moyenne(M)
```

(Coefficient de corrélation)

```
def coeff_cor(X, Y):
    """
    Renvoie le coefficient de corrélation des deux séries
    """
    return covariance(X, \
    ↪ Y)/(ma.sqrt(variance(X))*ma.sqrt(variance(Y)))
```

(Coefficients de régression)

```
def regression_lin(X, Y):
    """
    renvoie les coefficients a, b de régression  linéaire \
    ↪ associée au
    nuage de points (X, Y)
    """
    a = covariance(X, Y)/variance(X)
    b = moyenne(Y)-a*moyenne(X)
    return a, b
```

2.2. Exercices

Dans la suite, nous aurons parfois besoin de tracer un nuage de points et la droite de régression associée, on commence donc avec un exercice permettant d'effectuer ce tracé.

Exercice 3 | Tracer un nuage et la droite de régression Solution Écrire une fonction `nuage` prenant en argument deux séries statistiques `L1` et `L2` de même taille et qui affiche le nuage de points pour lequel les abscisses sont les valeurs de la liste `L1` et les ordonnées correspondantes les valeurs de la liste `L2`. Le graphique devra également faire figurer le point moyen (en une autre couleur). *Pour le nuage de points, on utilisera l'option* `"bo", markersize=1`, *et pour le point moyen* `"ro", markersize=10`.

Dans la suite, on pourra utiliser la fonction ci-après, qui en plus du nuage de points trace la droite de régression associée, et renvoie les coefficients de la droite, ainsi que le coefficient de détermination.

```
def nuage_reg(L1, L2):
    nuage(L1, L2)
    # droite de régression :
    a, b = regression_lin(L1, L2)
    r = coeff_cor(L1, L2)**2
    plt.plot(L1, [a*x+b for x in L1], "g", label="r =
"+str(round(r,2)))
    plt.legend()
    plt.show()
    return a, b, coeff_cor(L1, L2)**2
```

Exercice 4 | Distance et freinage Solution Le tableau suivant donne la distance de freinage d'un véhicule roulant sur route sèche en fonction de sa vitesse.

Vitesse en km/h	40	50	60	70	80	90	100	110
Distance en m	8	12	18	24	32	40	48	58

1. Calculer le coefficient de corrélation de cette série double.

2. >_ Représenter le nuage de points de cette série double et la droite de régression associée.
3. >_ Déterminer l'équation de la droite d'ajustement affine de la distance en fonction de la vitesse.

4. >_ Estimer la distance de freinage d'un véhicule roulant à 120 km/h.

Exercice 5 | Pression et altitude Solution Le tableau ci-dessous indique la pression (en hPa) en fonction de l'altitude (en km) :

Altitude x	0	1	2	3	4	5	6	7	8
Pression y	1010	896	792	699	614	538	470	409	355
Altitude x	9	10	12	15	20	25	30	35	40
Pression y	306	264	194	121	55	26	12	6	3

1. >_ Le nuage de points associé à (x, y) suggère-t-il qu'un ajustement affine est adapté ?
2. >_ On pose $z = \ln(y)$. Visualiser le nuage de points associé à la série double (x, z). Un ajustement affine semble-t-il adapté ?
3. >_ Obtenir la droite de régression de z en x. En déduire une approximation de la pression à une altitude de 50 km.

Exercice 6 | Loi SPAR Solution Plus une région est vaste, plus le nombre d'espèces y vivant est grand. Pour modéliser mathématiquement ce phénomène (et mesurer la biodiversité), les scientifiques utilisent régulièrement la loi SPAR (« species-area relationship »). Elle stipule que si S représente la surface de la région étudiée et n le nombre d'espèces présentes dans cette région, alors $n = Cs^Z$, où C et Z sont des constantes à ajuster selon la région étudiée.

On demande à des élèves d'une classe de BCPST de vérifier cette loi pour les plantes présentes dans une prairie. Les données récoltées sont résumées dans le tableau suivant, exprimant le nombre d'espèces différentes présentes n en fonction de la surface s de la prairie étudiée :

Surface s en m^2	1	2	4	8	16	32	64
Nombre d'espèces n	6	7	8	10	10	13	14

Afin de mettre en valeur la relation donnée par la loi SPAR, on décide de procéder à une régression linéaire de $\ln n$ sur $\ln s$.

1. Justifier le choix de cette régression linéaire.

2. >_ Tracer le nuage de points correspondant aux variables $\ln s$ (en abscisses) et $\ln n$, ainsi que la droite de régression linéaire.
3. >_ Combien vaut le coefficient de corrélation linéaire ? Que peut-on en déduire ?

4. >_ Donner alors une approximation de C et Z.

3. SOLUTIONS DES EXERCICES

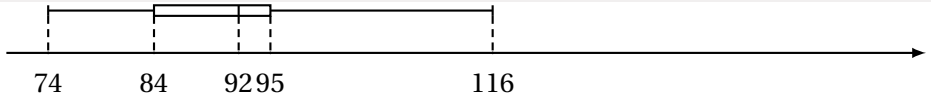
Solution (exercice 1) Énoncé Expliquons tout d'abord l'aspect mathématique, on commence par ordonner les valeurs :

- A : 74 – 81 – 84 – 88 – 91 – 91 – 93 – 95 – 95 – 95 – 102 – 116
- B : 94 – 94 – 95 – 95 – 104 – 105 – 113 – 113 – 144 – 153

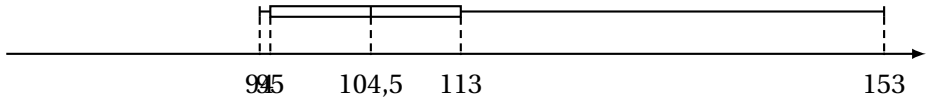
L'étendue de la série A est de 116 – 74, soit $\boxed{42}$. Sa médiane est la moyenne entre les deux valeurs du milieu, 91 et 93, soit $\boxed{92}$. L'étendue de la série B est de 153 – 94, soit $\boxed{59}$. Sa médiane est la moyenne entre les deux valeurs du milieu, 104 et 105, soit $\boxed{104.5}$. Pour la série A, on trouve $Q_1 = 84$ et $Q_3 = 95$.

1.
```
A = [74, 81, 84, 88, 91, 91, 93, 95, 95, 95, 102, 116]
B = [94, 94, 95, 95, 104, 105, 113, 113, 144, 153]
```
```
>>> moyenne(A)
92.08333333333333
>>> moyenne(B)
111.0
>>> quartiles(A)
(84, 92.0, 95)
>>> quartiles(B)
(95, 104.5, 113)
>>> etendue(A)
42
>>> etendue(B)
59
>>> maximum(A)
116
>>> minimum(A)
74
>>> maximum(B)
153
>>> minimum(B)
94
```

2.

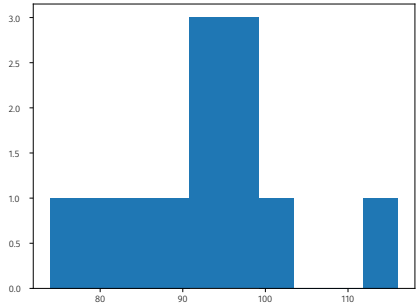


Pour la série B, on trouve $Q_1 = 95$ et $Q_3 = 113$. Il reste à construire les deux diagrammes.

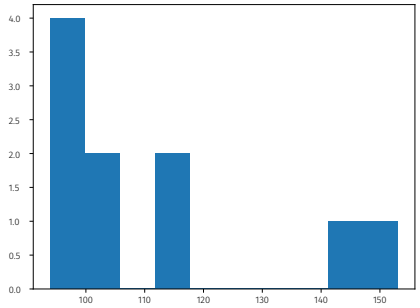


Les valeurs de la série A sont sensiblement plus faibles que celles de la série B : le médicament semble efficace. Pour conclure de manière plus qualitative, il faudrait cependant faire plutôt un test d'adéquation (voir programme de 2ème année).

3.
```
plt.hist(A, 10)
```



```
plt.hist(B, 10)
```



Solution (exercice 2) Énoncé

1.
```
D = {5:8, 10:17, 15:12, 20:8, 25:5, 30:3, 35:2, 40:1}
L = dico_occur_vers_liste(D)
```
```
>>> mediane(L)
15.0
>>> moyenne(L)
15.625
>>> etendue(L)
35
```

2. 2.1) On fait une recherche de valeur maximale du dictionnaire, on retourne ensuite la clef associé. Cette fonction est analogue à la fonction

`maximum` déjà rencontrée pour les listes. On initialise par exemple la valeur à zéro, puisque les valeurs du dictionnaire sont positives (effectifs). La petite difficulté ici est d'avoir l'une des modalités, c'est pour cette unique raison qu'on indique `L` en argument de fonction plutôt que le dictionnaire des effectifs.

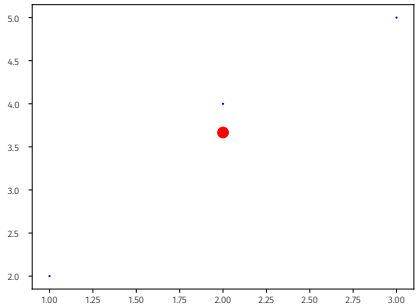
```
def mode(L):
    eff_max = 0
    mode = L[0]
    D = dico_occur(L)
    for modalite in D:
        if D[modalite] > eff_max:
            eff_max = D[modalite]
            mode = modalite
    return mode
```

2.2) On l'exécute sur la série précédente.

```
>>> mode(L)
10
```

Solution (exercice 3) Énoncé

```
def nuage(L1, L2):
    plt.plot(L1, L2, "bo", markersize=1) # o pour des points \
    ↪ non reliés
    # point moyen
    x, y = moyenne(L1), moyenne(L2)
    plt.plot(x, y, "ro",  markersize=10)
nuage([1, 2, 3], [2, 4, 5])
plt.show()
```



Solution (exercice 4) Énoncé

1. Les camculs peuvent s'effectuer à l'aide de Python ou à la main (à l'aide de la calculatrice). On note x la série des vitesses, et y la série des distances. On calcule la moyenne des deux séries : on obtient $\overline{x} = 75$ et $\overline{y} = 30$. On calcule également la moyenne de la série produit, et on obtient $\overline{xy} = 2627.5$. Grâce à la formule de König-Huygens, on a alors :

$$\mathbb{C}_{x,y} = \overline{xy} - \overline{x} \times \overline{y} = 377.5.$$

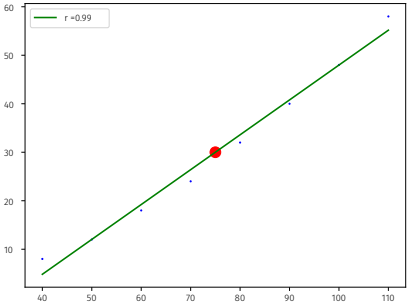
Pour en déduire le coefficient de corrélation, on doit calculer les écart-types de x et y. Grâce à la formule de König-Huygens on a :

$$\mathbb{V}_x = \frac{1}{N}\sum_{i=1}^{N} x_i^2 - \overline{x}^2 = 525, \quad \mathbb{V}_y = \frac{1}{N}\sum_{i=1}^{N} y_i^2 - \overline{y}^2 = 275.$$

On obtient ainsi $\rho_{x,y} = \dfrac{\mathbb{C}_{x,y}}{\sigma_x \sigma_y} = \dfrac{\mathbb{C}_{x,y}}{\sqrt{\mathbb{V}_x \mathbb{V}_y}}$, soit $\boxed{\rho_{x,y} \simeq 0.99}$. On remarque que le coefficient de corrélation est très proche de 1 : les données sont quasiment alignées sur une droite.

2.

```
nuage_reg([40 , 50 , 60 , 70, 80, 90, 100, 110], [8, 12, \
↪ 18, 24, 32, 40, 48, 58] )
```



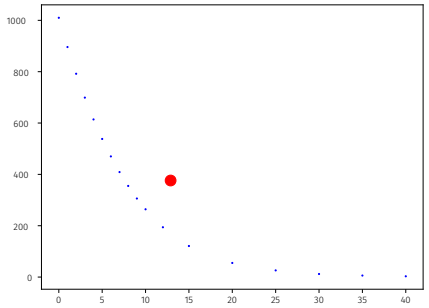
3. D'après le cours, on sait que la droite d'ajustement affine a pour équation $y = ax + b$, avec $a = \dfrac{\mathbb{C}_{x,y}}{\mathbb{V}_x}$ et $b = \overline{y} - a\overline{x}$. On en déduit alors $a \simeq 0.72$ et $b \simeq -23.9$. L'équation de la droite d'ajustement affine est donc de $\boxed{y = 0.72x - 23.9}$.

4. D'après la régression linéaire, on a, pour une vitesse de 120 km/h, une distance de $0.72 \times 120 - 23.9$, soit environ $\boxed{62.4\text{m}}$.

Solution (exercice 5) Énoncé

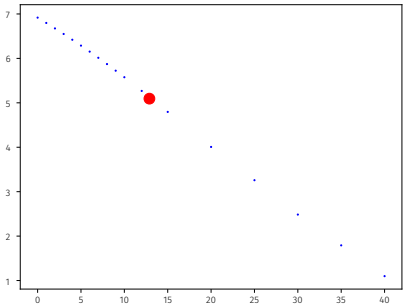
1.

```
L_alt = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 15, 20, 25, \
↪ 30, 35, 40]
L_press = [1010, 896, 792, 699, 614, 538, 470, 409, 355, \
↪ 306, 264, 194, 121, 55, 26, 12, 6, 3]
nuage(L_alt, L_press)
```



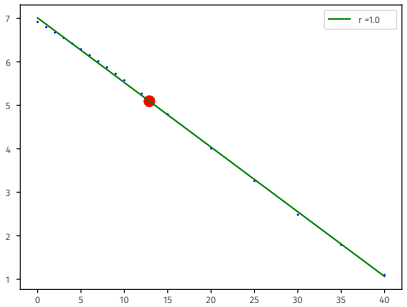
D'après la figure, le modèle linéaire ne semble donc pas très pertinent.

2.
```
L_lnpress = [ma.log(p) for p in L_press]
nuage(L_alt, L_lnpress)
```



Un ajustement affine semble donc adapté à cette nouvelle série bivariée (x, z).

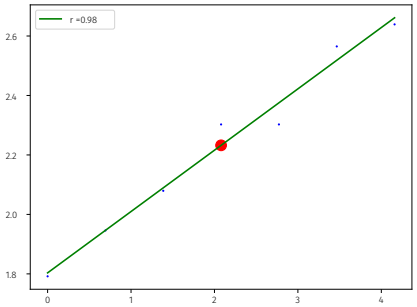
3.
```
>>> a, b, _ = nuage_reg(L_alt, L_lnpress)
>>> ma.exp(a*50+b) # pression estimée à 50 km, sans \
↪ l'exponentielle c'est son log
0.6554755510356131
```



Solution (exercice 6) Énoncé

1. Si la relation est vérifiée, alors en passant au logarithme on obtient : $\ln n = \ln C + Z \ln s$. Ainsi, la série $(\ln s, \ln n)$ suit un modèle affine.

2.
```
L_s = [1, 2, 4, 8, 16, 32, 64]
L_n = [6, 7, 8, 10, 10, 13, 14]
L_lns = [ma.log(x) for x in L_s]
L_lnn = [ma.log(x) for x in L_n]
a, b, rho = nuage_reg(L_lns, L_lnn)
```



3. Le coefficient de corrélation vaut 0.98 d'après le question précédente. On suspecte donc une relation linéaire entre $\ln N$ et $\ln S$.

4. En reprenant la première question, on observe que $\ln C$ est l'ordonnée à l'origine de la droite de régression, et z son coefficient directeur.

```
>>> a # valeur approchée de z
0.20625981968084586
>>> ma.exp(b) # valeur approchée de C
6.070382539605234
```