Probabilités et Statistiques
Projet noté

MADANI Abdenour
TRIOLET Hugo

Licence 3
2021 - 2022

Table des matières

1 Introduction

1.1 Objectifs

Les objectifs de ce TPs sont :

- implémenter nous-mêmes plusieurs algorithmes de régression linéaire et les comparer à des fonctions issues de librairies scientifiques
- manipuler différentes lois vues en cours via leur implémentation issues de librairies scientifiques
- déterminer des intervalles de confiance et effectuer des applications sur quelques exemples

On utilisera pour ceci **Python** et les bibliothèques de fonctions : Numpy, Scipy, Matplotlib, et Statsmodels, entre autres.

1.2 Définitions

Hugo : tu peux virer ça si t'as aucune définition à mettre (tu peux la réutiliser plus bas et virer cette partie aussi)

> 🎓 DÉFINITION
>
> **Mot défini**
>
> Définition ici

1.3 Résumé de notre approche

Nous avons 3 fichiers, 1 pour chaque TP.

Vis-à-vis du code, nous l'avons documenté à l'aide de la docstring de Python, ainsi que des commentaires normaux : les fonctions se comprennent donc naturellement grâce à ceux-ci.

2 Régression linéaire

2.1 Régression Linéaire simple

La fonction calculant la régression linéaire simple est "regression_lineaire".

Étant donné deux listes x et y de même taille, elle calcule la régression linéaire

$$y = \beta_1 \cdot x + \beta_0$$

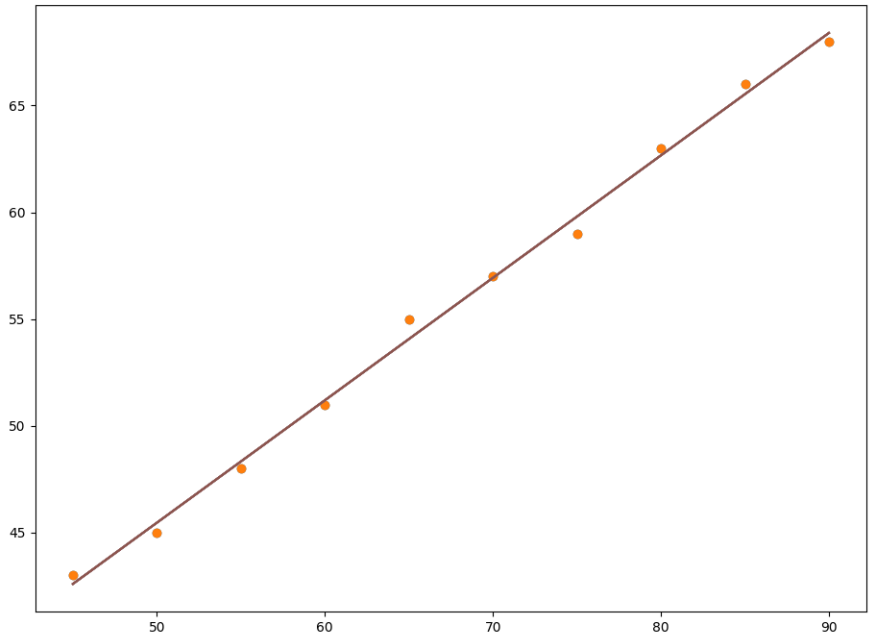
2.1.1 Modèle vectoriel

On applique simplement la formule donnée dans le TP.
La fonction calculant la régression linéaire simple est "regression_lineaire_vec".

Étant donné deux listes x et y de même taille, elle calcule la régression linéaire en utilisant la méthode vectorielle :

$$y = \beta_1 \cdot x + \beta_0$$

2.1.2 Résultats obtenus



Représentation graphique obtenue avec Matplotlib

En orange sont affichés les points de (x_i, y_i), et on voit plusieurs droites superposées de couleurs différentes, quasiment indiscernables : ce sont nos deux régressions linéaires ainsi que celle de Numpy (polyfit).
Les résultats sont donc concordants : visuellement, toutes les régressions linéaires donnent le même résultat sur ce jeu de donnée.
Les coeffcients sont de mêmes très proches voire égaux.

2.2 Régression linéaire et descente de gradient

Hugo : todo

3 Étude et manipulation de lois de probabilités

3.1 Loi Binomiale

texte

Si tu veux mettre une image Hugo

3.2 Loi Normale univariée

texte

3.3 Simulation de données à partir d'une loi

texte

3.3.1 Cas de la loi normale

texte

3.4 Estimation de densité

texte

3.4.1 Cas de la loi normale

texte

3.4.2 Cas de la loi exponentielle

texte

4 Intervalles de confiance

Le but de cette partie (correspondant au fichier *tp3.py*) est de déterminer les intervalles de confiances de différents échantillons statistiques afin de situer, selon une certaine précision, dans quelle plage de valeurs se situe l'espérance de l'échantillon.
Dans les 2 premiers problèmes, nous ne connaissons pas la variance de l'échantillon, ainsi la fonction calculant l'intervalle de confiance des échantillons utilisera la méthode de calcul via l'écart-type empirique et le fractile t d'ordre $1 - \frac{\alpha}{2}$ de la

loi de student $St(n-1)$. Le problème 3, quant à lui, modélise un échantillon de taille n de $B(p=\frac{1}{2})$. Ainsi la variance de la loi de Bernoulli est connue et vaut $p(1-p)$, donc dans ce cas là, le calcul de l'intervalle de confiance se fera via la formule utilisant l'écart-type véritable et u le fractile d'ordre $1-\frac{\alpha}{2}$ de la loi $N(0,1)$.

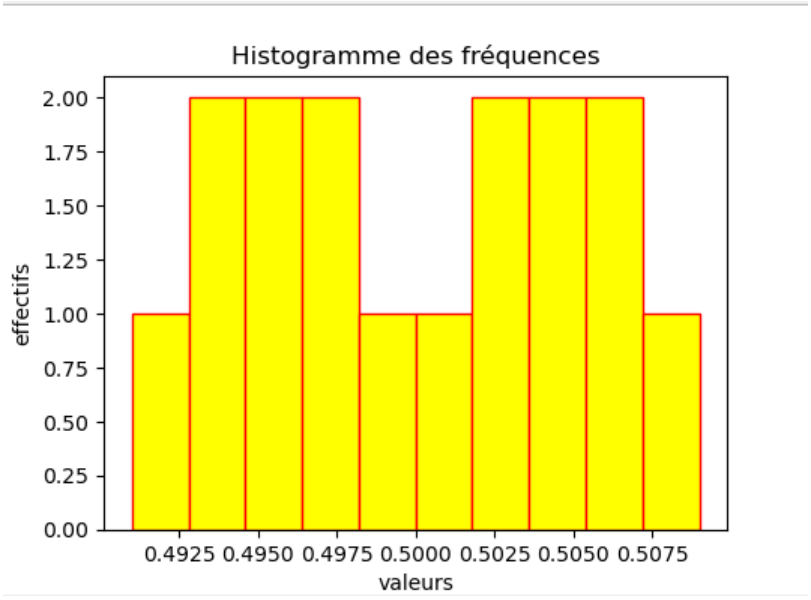
4.1 Fonctions

4.2 Problème 1

On possède deux échantillons de taille 16 à notre disposition : un échantillon de masses de 16 pots de confitures mesurée en kilogramme (kg), et l'autre de masses d'avocats provenant du Mexique mesurée en grammes (g).

→On traite l'échantillon de masses en kg en premier.

Question 1 :
La moyenne empirique obtenue est de 0.5 kg.
Question 2 :
L'histogramme obtenue sur l'échantillon des pots de confitures est le suivant



Histogrammes des fréquences des masses des pots de confitures de l'échantillon

Question 3 :
On calcule les intervalles de confiances à 95% et à 99% pour l'échantillon des

masses des pots. On obtient les intervalles suivantes :

- $IC_{95\%}(\mu) = [0.497, 0.503]$
- $IC_{99\%}(\mu) = [0.496, 0.504]$

Il y a peu de différences dans notre cas entre l'intervalle à 95% et l'intervalle à 99%. on remarque par ailleurs que pour une petite variation de l'ordre de 0.001 sur chacune des bornes de l'intervalle de confiance à 95% pour passer à celle à 99%, on obtient une variation de +0.04 sur la probabilité de trouver une masse dans celle-ci, (i.e $P(0.496 \leq masse\_kg \leq 0.504) = 0.99$) ce qui veut dire que la précision est affiné pour une très petite variation à cet ordre de sûreté.

→Quant à l'échantillon de masses des avocats du Mexique.
Après calcul de son intervalle de confiance, avec toujours le fait que l'on ne connaisse pas sa variance, on obtient $IC_{95\%}(\mu) = [85.941, 88.558]$. Autrement dit, pour une masse en g d'un avocat de cet échantillon, on est sûr à 95% qu'il se situera dans cette intervalle, i.e $P(85.941 \leq masse\_g \leq 88.558) = 0.95$

4.3 Problème 2

Voici l'énoncé" du problème 2 : *Une compagnie aérienne souhaite étudier le pourcentage de voyageurs satisfaits par ses services, on en a interrogé 500 choisis au hasard. Parmi eux, 95 se disent satisfaits. Déterminer un intervalle de confiance pour la proportion inconnue de voyageurs satisfaits au niveau 99%.*

On ne dispose pas d'échantillon mais on sait que $\frac{95}{500}$ personnes se disent satisfaites. On choisit ce même échantillon qui a été pris dans le cadre de l'énoncé (qu'on représente par une liste de 0 et de 1, 0 signifiant que la personne i n'est pas satisfaite et 1 qu'elle est satisfaite) et on pose donc que la moyenne empirique vaut $\frac{95}{500}$. On en connait pas la variance de cet échantillon (étant donné qu'il ne suit pas une loi particulière, à priori). Ainsi, puisque la variance empirique est calculée lors de la détermination de l'intervalle de confiance, on exécute la fonction de calcul d'intervalle lors d'une non-connaissance de la variance.
On obtient l'intervalle suivant : $IC_{99\%}(\mu) = [0.145, 0.235]$
→ i.e $P(0.145 \leq proportion\_satisfaits \leq 0.235) = 0.99$.

Remarque : on peut remarquer que la façon donc l'échantillon a été modéliser et les calculs faits que l'échantillon semble être un ensemble de 500 variables aléatoires suivant une loi de Bernoulli de paramètre θ compris dans notre intervalle de confiance.

4.4 Problème 3

Dans le problème 3, on utilise la fonction *stats.bernoulli.rvs()* (fonctionnement détaillé dans **Fonctions**) qui nous permet de générer un tableau (i.e un échantillon) de n expériences de Bernoulli de paramètre $p = \frac{1}{2}$. On convertit ce tableau en liste pour travailler plus facilement dessus (du fait que le tableau n'est qu'une dimension). Dans le cas présent, nous connaissons la variance, qui vaut très exactement $\sigma^2 = p(1-p)$.
On calcule donc l'intervalle de confiance en prenant en compte ce fait (et donc en utilisant la fonction appropriée).
On obtient comme intervalles de confiance à 95% les intervalles suivants selon n :

n	$IC_{95\%}(\mu)$
10	$[0.19, 0.81]$
20	$[0.331, 0.769]$
50	$[0.381, 0.659]$
100	$[0.422, 0.618]$
200	$[0.501, 0.639]$
500	$[0.428, 0.516]$
1000	$[0.485, 0.547]$
10000	$[0.49, 0.51]$
1000000	$[0.498, 0.5]$

On remarque que plus l'effectif est grand, plus l'intervalle de confiance se réduit et devient précis quant à la valeur du paramètre trouvable à 95% dans celui-ci.

5 Exemples d'utilisation du code

5.1 Comment utiliser le code

Concernant le code, il est séparé en trois fichier (comme dit plus haut en partie ***Résumé de notre approche***), chacun correspondant par son indice à la partie du TP correspondante (*tp_3.py* avec la partie 3, *TP2.py* avec la partie 2, ...).
Si le code est exécuté sur l'IDE `Pyzo`, alors il est possible (après avoir exécuté "l'en-tête" du code, c'est à dire de la première ligne au premier ##) d'exécuter séparément chaque sous partie du code, délimitées par ## *Problème x*.
Autrement, le code s'exécute normalement et en entier, si `Pyzo` n'est pas le logiciel où celui-ci est exécuté. Et bien penser à rentrer une valeur lorsque le programme le demandera (pour le problème 3 notamment).