TP de statistique inférentielle

> ### A propos de jupyter notebook.
>
> C'est un espace qui permet de mélanger du texte (même latex), code etc. Pour executer une cellule tapez Enter. Pour changer le type de cellules vous pouvez utilisez le menu en haut. Par exemple ceci est une cellule dite "Markdown" pour du texte. En bas vous allez voir des celules type "code" ou on peut executer du code (Python3 ici). On encourage à se documenter sur internet (https://jupyter-notebook.readthedocs.io/en/stable/ site officiel par exemple) ou un grand nombre de tutoriels et réponses aux questions sont présents.

A propos des différents modules utilisés.

On encourage fortement à se documenter sur internet pour des questions plus poussées (ou pas)

> **pandas**: est un module qui permet d'analyser et de manipuler les données. On verra en particulier l'objet dataframe qui permet de manipuler des tables de données.
>
> **matplotlib.pyplot**: est un module permettant d'afficher des graphiques.
> seaborn: un module qui permet de visualiser les données. Souvent, quand on sait le manipuler, ça donne des résultats "plus beaux" que matplotlib.pyplot
>
> **numpy**: Un package de calcul scientifique
>
> **scipy.stats**: Un module qui contient un grand nombre d'outils statistiques: distribution, fonction de répartition, tests etc.

Consignes.

> Répondre à chaque question dans les cellules que vous créerez après la question correspondante. La note prendra compte de la présentation et de la clareté des réponses.
>
> Envoyez le tp sur le mail sholom.schechtman@telecom-sudparis.eu. Avec comme sujet: **TP1_NOM_PRENOM**. C'est **important** pour que votre tp ne se perde pas et soit **noté**.

On importe les packages nécessaires

In [ ]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import scipy.stats as stat
import seaborn as sns
%matplotlib inline
```

Exercice 1. pandas et manipulations de données

> 1.
>
> Créez un dictionnaire, representant nos données: `data = {"nom": ["tutu", "toto", "titi"], "sexe": ["M", "M", "F"], "taille": [1.56, 1.82, 1.70], 'naissance": ["11/12/82", "21/12/82", "25/12/83"]}`
>
> Utilisez la commande `df = pd.DataFrame(data)` pour importer les données dans un Dataframe de nom df

> 2.
>
> Testez et décrivez le fonctionnement des commandes `df.head()` , `df.head(0)` , `df.head(1)` , `df.head(2)` .

> 3.
>
> Qu'est ce qui se passe si on remplace head par tail dans la question précédente?

> 4.
>
> Décrivez le fonctionnement de la commande `df.sort_values` : prenez un argument "taille" et testez deux valeurs `ascending= True` et `ascending = False`

> 5.
>
> Testez et décrivez la commande `df["var"] = [1 if x > 1.6 else 0 for x in df["taille"]` ?

> 6.
>
> Supprimez la colonne "var" avec la commande `df.drop` (voir documentation et mettez axis=1 en argument)

> 7.
>
> Testez la commande `df.loc[df["sexe"] == "M"]` . Testez la commande `df.loc[df["sexe"] == "M", "taille"]` .

> 7.

On suppose que le poids d'un homme est poids = (tailleencm)/2 - 10 et le poids d'une femme est poids = (tailleencm)/2 - 10

> A l'aide des questions précédentes completez la colonne poids.

> 6.
>
> Que fait la commande `df["taille"].mean()` ? Faites apparaître le poids moyen des données

> 7.
>
> Que fait la commande `df.loc[df["sexe"]=="M"]` ? Faites apparaître le poids moyen des hommes

> 8.
>
> Faites apparaître le nom de la personne la plus légère.

Exercice 2: Analyse d'un jeude données bancaires

Le jeu de données GermanCredit.data, disponible sur le site de l’ UCI repository, comporte desrenseignements sur 1000 clients d’une banque allemande, chaque client étant décrit par 20 variables.

1.

Téléchargez le et importez le dans une DataFrame en utilisant la commande `german = pd.read_csv("nom.csv, sep = ' ', header=None)`

2.

Creez le dictionnaire `{1: "durée", 4:"montant"}` et utilisez le dans la commande `german.rename()` pour renommer les colonnes correspondantes.

3.

Utilisez la commade `describe` de pandas pour obtenir des premières informations sur la durée et les montants du crédit

4.

Utilisez la commande `german.boxplot()` pour créez une boite à moustache de la variable durée. Faire de même pour la variable montant.

5.

Faire la question précedente en utilisant le module seaborn: `sns.boxplot`

7.

Utilisez la commande `german.hist` pour tracer un histograme des variables durées et montant

8.

Utilisez la commande `german.plot.scatter` pour tracer le scatter plot des variable "durée" et "montant"

> 9.
>
> Tracez la table de correlations entre "durée" et "montant" à l'aide de la commande `german[].corr()` (pensez à selectionner les bonnes colonnes).

> 10.
>
> Utilisez `stat.pearsonr` pour tester la correlation entre les deux variables. Que peut-on en conclure?

Exercice 3: Simulation d'un jeu de données

> 1.
>
> Quel est l'effet de la commande suivante?

> 2.
>
> Commentez la différence avec la question précédente. (essayez d'executer la cellule plusieurs fois de suite)

> 3.
>
> Sauvegardez dans une variable "simul" un échantillon de 100 valeurs aléatoires, chacune tirée uniformément dans [0,1].
>
> Tracez son histogramme avec la commande plt.hist

> 4.
>
> Dans les commandes précédentes remplacez `rand` par `randn` et `exponential` . Qu'est ce qui change?

> 5.
>
> Simulez dans une variable `simul12` , 12 simulations (1 par ligne) de 1000 tirages d'une loi uniforme Utilisez la commande `np.sum` pour avoir une array de 1000 éléments qui somme les 12 tirages. Sauvegarder ce resultat dans une variable `simul_sum` . Enfin tracez l'histogramme de `simul_sum` . De quoi s'aperçoit-on. Commentez.

> 6.
>
> Utilisez la commande `sns.kdeplot` , pour tracer la densité estimée de simul_sum

> 7.
>
> Utilisez la commande `stat.ttest_1samp` du module scipy.stats pour faire un test de student de moyenne 6. Faire de même un test de student de correspondance à une moyenne de 7. Commentez.

Exercice 4: Temps d'attente moyen au guichet

On cherche à estimer le temps d'attente moyen au guichet d'une grande banque aux heures de forte affluence. On a observé 26 clients choisis au hasard et on a obtenu les temps d'attente sauvegardés dans une liste `t_a` .

On suppose que ces temps d'attentes sont distribués normalement et on se demande:

Est-ce qu'on peut rejetter, au risque de 5%, le fait que le temps d'attente au guichet est égal à 4 minutes?

In [ ]:
```
t_a = [6.1, 4.7, 5.6, 4.5, 5.5, 6.8, 2.1, 2.1, 3.5, 2.5, 6.7, 4.4, 4.5, 6
```

> 1.
>
> `On pense effectuer un test de Student. Pourquoi est-il approprié ici? Dans quel cas ne serait-il pas?`

> 2.
>
> Effectuez ce test à l'aide de la fonction `stat.ttest_1samp` . Interpretez les résultats.

> 3.
>
> On voudrait maintenant tester l'hypothèse que le temps d'attente est supérieur à 4. En utilisant l'argument optionnel "alternative" dans la fonction `stat.ttest_1samp` , testez cette hypothèse. (lisez la documntation si besoin). Commentez.

Exercice 5: Test de performance.

Un directeur d'entreprise a fait passer le même test d'aptitude à deux groupes de candidats. Le temps (en minutes) nécessaire à chacun des candidats pour répondre au test sont stocké ci dessous.

Nous supposons que les temps de réponse de chaque groupe sont distribués normalement. Et que les performances des deux groupes sont indépendantes.

In [ ]:
```
time = {"gr1": [8.6, 10.9, 7.3, 9.2, 8.5, 9.2, 9.1, 8.9, 10.7, 8.2, 7.1,
       "gr2": [8.3, 7.2, 8.7, 6.7, 10.3, 6.8, 9.8, 8.9, 9.6, 8.6, 6.7, 7.
```

> 1.
>
> On voudrait tester l'hypothèse que les variances des temps de réponse du groupe 1 et 2 sont identiques. Pourquoi le test de Fisher est adapté à cette tache. A combien de degrés de libertés?

2.

Ce test n'est pas fourni par la bibliothèque scipy. On va donc le coder à la main.
Completez le code ci-dessous.

In [ ]:
```
from scipy.stats import f

var1 = np.var(time["gr1"], ddof=1)
var2 = np.var(time["gr2"], ddof=1)

## Degrés de libertés
d1=
d2=

## F est notre statistque de test, formés des variables définies au dessu
F =

##Mettez alpha = 0.05 et calculez les quantiles alpha, et 1-alpha de la l
alpha= 0.05
q1 = f.cdf()
q2 = f.cdf()

### Test d'hypothèse
if F<q1 or F > q2:
    print("H_0 est rejetté au risque < 5%")
else:
    print("On ne peut pas rejetter H_0 au risque < 5%")

print("variance empirique du groupe 1 : ", var1)
print("variance empirique du groupe 2 : ", var2)
print("statistique de test : ", F)
```

> 4.
>
> Pourquoi a-t-on mis ddof=1 dans les calculs de la variance?

> 5.
>
> En utilisant la fonction f.cdf et la statistique F calculée à la question précédente. Calculez la p value du test.

> 6.
>
> Si la performance des candidats des deux sexes lors du test n'est évaluée que par le temps nécessaire pour y répondre, peut-on affirmer qu'il y a une différence réelle entre la performance moyenne des candidats et celle des candidates ? En utilisant `stat.ttest_ind` effectuez le test correspondant et justifiez le choix de ce test.

Exercice 6. Test d'adéquation à une loi

Le fichier des.csv (csv des) contient 1200 réalisation d'un lancé de dès. Importez le avec la fonction `pd.read_csv` .

> 1.
>
> On voudrait tester l'hypothèse que le dè n'est pas truqué. Quel test choisir? Effectuez le test avec la fonction du module `scipy.stats` (importé `stat` précédemment) correspondante. indice: utilisez la fonction `np.unique()` pour calculez les fréquences d'apparitions.

> 2.
>
> Effectuez le test d'adéquation à la loi qui donne 1/6 de probabilité d'apparition au chiffres 1,3,4,6; 1/12 au chiffre 2 et 1/4 au chiffre 5.

Exercice 7. Tests d'indépendance

> 1.
>
> Importez le fichier `hsb2.csv` dans une dataframe. Affichez les première lignes.

dans la suite on choisira un test adapté pour répondre aux questions posées

> 2.
>
> Le sexe des lycéens influence-t-il le type d'établissement fréquenté?

> 3.
>
> Pourquoi il n'est pas très judicieux d'effectuer un test similaire pour tester l'indépendance entre le statut social (varible "ses") et l'ethnie?
> indice: faites afficher le tableau de contingence pour ces variables.

> 4.
>
> Proposez une méthode de modification de données pour tester tout de même l'indépendance. Effectuez le test correspondant.