Université de Montpellier **HLMA408: Traitement de données**
2019/2020 **Joseph Salmon**

TP n° 3 : Tests d'adéquation, modèle quadratique

Objectifs du TP : Test d'égalité de moyennes, permutations aléatoires et hypothèse nulle, modèles linéaires.

Quelques conseils de base : Pour chaque fiche de TP, nous utiliserons un script `jupyter notebook`. Il est conseillé de créer un répertoire HLMA408, puis un sous-répertoire pour chaque TP (*e.g.,* TP1, TP2, TP3). Dans un répertoire (TP3 par exemple), vous stockerez donc :

— le sujet de TP au format PDF,

— les fichiers de données (qui seront téléchargés automatiquement),

— le fichier `TP-note-3-squelette.ipynb` qui contient quelques commandes pré-remplies correspondant au TP, et que vous pouvez télécharger sur le site du cours.

Consignes :

Pour ce travail vous devez déposer un **unique** fichier sur le moodle du cours HLMA408. Vous rendrez un unique fichier d'extension `ext=ipynb`. Attention, vous devrez veiller à ce que le format soit correct : si le fichier ne peut pas être ouvert par le correcteur avec `jupyter-notebook`, la note sera de zéro : **cette vérification vous incombe**!

Pour faciliter la correction, le nom du fichier devra respecter le format suivant :

`nom_fichier = tp_note3_hlma408_gr_nbgr_prenom_nom.ipynb`, en minuscule, sans accent ni espace. Vous remplirez votre nom, prénom, le numéro de groupe qui vous concerne (remplacer `nbgr` par C, D ou E) de manière adéquate[1].

Un point de malus sera appliqué pour les fichiers dont le format est erroné.

Vous devez charger votre fichier sur Moodle, avant le mercredi 9/04/2020, 23h59. La note totale est sur **20** points, répartis comme suit :

- qualité des réponses aux questions : **14** pts,
- qualité de rédaction et d'orthographe : **1** pt,
- qualité des graphiques (légendes, couleurs) : **2** pts
- qualité d'écriture du code (noms de variable clairs, commentaires, code synthétique, etc.) : **1** pt
- Rendu reproductible et absence de bug : le code doit s'exécuter sur la machine du correcteur sans manipulation de sa part (par exemple le correcteur n'est pas supposé aller chercher les fichiers sur internet, les enregistrer, etc.). On veillera donc à ce que les traitements de donnée soient automatisés et ne requièrent pas d'interversion supplémentaire **2** pts.

Les personnes qui n'auront pas soumis leur devoir sur Moodle avant la limite obtiendront **zéro**.

1 Brochet et concentration de polluants

Dans une étude sur les mécanismes de détoxication, on dispose de la concentration du DDT et de ses dérivés, DDD et DDE, (en mg/g) contenus dans des brochets du Nord (*Esox lucius*), capturés dans la rivière Richelieu (province de Québec). Les données sont relatives aux brochets de 2 et 3 ans.

1.1 Comparaison de deux échantillons indépendants

On se place dans le cadre où le statisticien dispose de deux échantillons indépendants à analyser. On note μ_1 l'espérance de la variable dans le modèle de la première population et μ_2 l'espérance de la variable dans le modèle de la seconde. Bien sûr, ces deux paramètres sont inconnus. On souhaite tester

$$\mathcal{H}_0 \ : \ \mu_1 = \mu_2 \quad \text{contre} \quad \mathcal{H}_1 \ : \ \mu_1 \neq \mu_2$$

avec un test de Student. On pose $\mu_{\text{diff}} = \mu_1 - \mu_2$.

1. Exemple : Joseph Salmon est dans le groupe C, son TP s'appelle `tp_note3_hlma408_gr_C_joseph_salmon.ipynb`

1) Télécharger (automatiquement) et importer le jeu de données en `Python` avec la commande `read_csv` de `pandas`. Le lien pour les données :
http://josephsalmon.eu/enseignement/datasets/brochet2.dat

2) [**0.25** pt(s)] Recoder la variable `age` à l'aide de la fonction `replace` de `pandas` pour avoir des valeurs sous format `int` (nombre entier).

3) [**0.25** pt(s)] Calculer et commenter les statistiques résumées obtenues avec les commandes ci-dessous :

```
df_brochets.query('age==2').describe()
df_brochets.query('age==3').describe()
```

On pourra aussi garder les échantillons par âge pour la suite avec la commande :

```
X2 = df_brochets.query('age==2')['conc']
X3 = df_brochets.query('age==3')['conc']
```

4) [**0.5** pt(s)] Afficher un graphique en "violon" de la concentration sur la population entière, puis en distinguant suivant l'âge des brochets.

5) [**0.5** pt(s)] Effectuer un test de Student sur la concentration en fonction de l'âge (pour deux échantillons indépendants) avec la commande `ttest_ind` de `scipy.stats.mstats`, en expliquant ce que renvoie cette fonction.

6) [**1** pt(s)] Donner la formule mathématique de la statistique de test utilisée, selon que l'on utilise `ttest_ind` avec l'attribut `equal_var=True` ou `equal_var=False` (on pourra utiliser les cellules markdown avec du LaTex pour écrire des formules de mathématiques, *cf.* le fichier squelette fourni).

7) [**0.25** pt(s)] Interpréter les résultats obtenus soit en faisant le choix `equal_var=True` soit le choix `equal_var=False` ; conclure.

1.2 Test d'égalité des variances

Notons σ_2^2 la variance (théorique, c'est-à-dire celle du modèle) de la première population (les brochets d'âge 2 ans) et σ_3^2 la variance de la seconde population (les brochets d'âge 3 ans). Le test de Fisher permet de tester l'égalité de ces deux quantités. La statistique de ce test sous l'hypothèse nulle :

$$\mathcal{H}_0 \ : \ \sigma_2^2 \ = \ \sigma_3^2 \ ,$$

est la suivante :

$$\frac{\hat{\sigma}_2^2}{\hat{\sigma}_3^2} \sim \mathcal{F}(n_2 - 1, n_3 - 1) \ ,$$

où $\mathcal{F}$ est la loi de Fisher [2]. Pour effectuer un tel test, on va utiliser la fonction `f` de `scipy.stats`.

Pour appliquer correctement ce test, il faut vérifier que les deux échantillons proviennent de distributions normales.

Par défaut, l'hypothèse alternative est $\mathcal{H}_1 \ : \sigma_2^2 \neq \sigma_3^2$ (cas bilatéral) [3].

8) [**0.25** pt(s)] Calculer la statistique : `f_stat = np.var(X2, ddof=1) / np.var(X3, ddof=1)`. Que signifie `ddof` dans le calcul des variances ci-dessus ?

9) [**1** pt(s)] Que vaut la quantité affichée ci-dessous :

```
from scipy.stats import f
f_distrib = f(len(X2) - 1, len(X3) - 1)
print(2 * min(f_distrib.cdf(f_stat), 1 - f_distrib.cdf(f_stat)))
```

On donnera quelques détails mathématiques pour expliquer la conclusion.

10) [**0.5** pt(s)] Proposer un test d'égalité des variances entre les deux classes d'âge différentes et conclure. On précisera les hypothèses utilisées.

2. La loi de Fisher est décrite ici : https://fr.wikipedia.org/wiki/Loi_de_Fisher
3. On peut changer d'hypothèse alternative pour des tests unilatéraux : pour avoir $H_2 \ : \sigma_2^2 > \sigma_3^2$ (resp. $H_1 \ : \sigma_2^2 < \sigma_3^2$)

1.3 Comparaison de la pollution sur Toulouse et sur Montpellier

Nous allons maintenant utiliser des données de pollution recueillies sur diverses communes de l'Occitanie[4]. Pour une description plus précise des données on pourra consulter :
http://data-atmo-occitanie.opendata.arcgis.com/datasets/4a648b54876f485e92f22e2ad5a5da32_0

11) Télécharger et charger les données `Mesure_journaliere_Region_Occitanie_Polluants_Principaux.csv` avec `pandas` dans un dataframe `df_pol_occ`.

12) [**0.25** pt(s)] Observer ce que donnent les commandes suivantes et commenter :

```
df_pol_occ['polluant'].unique()
df_pol_occ['nom_com'].unique()
```

13) Lancer le script suivant pour visualiser les violons des divers polluants sur Montpellier et Toulouse sur la période 2017-10-18 — 2018-10-07 :

```
import seaborn as sns
polluants = ['NO2', 'PM10', 'PM2.5','O3']

fig, axes = plt.subplots(2, 2, figsize=(6, 6))
df_pol_occ = df_pol_occ.rename(columns={'nom_com': 'Ville'})
for i, polluant in enumerate(polluants):
    querry = "(Ville=='TOULOUSE' or Ville=='MONTPELLIER') and " + \
        "polluant=='{}'".format(polluant)
    df_polluant = df_pol_occ.query(querry)[['Ville', 'valeur_originale']]
    plt.title(polluant)
    ax = axes.reshape(-1)[i]
    sns.violinplot(x='Ville', y='valeur_originale',
                   data=df_polluant, orient='v', ax=ax)
    ax.set_title(polluant)
    ax.set_ylabel(r'Concentration ($\mu g/m^3$)')
    plt.tight_layout()
plt.subplots_adjust(top=0.9)
fig.suptitle("Pollution en Occitanie (2017-10-18 / 2018-10-07)", fontsize=16)
plt.show()
```

Dans la suite on ne s'intéresse qu'aux données de ces deux villes.

14) [**0.25** pt(s)] Noter que pour les violons pour les particules fines PM2.5 peuvent prendre des valeurs négatives. En utilisant une option de `sns.violon`, proposer un remède contre cette étrangeté.

15) [**0.25** pt(s)] Le site https://www.respire-asso.org/ozone-o3/ recommande de ne pas dépasser le seuil des 100 $\mu g/m^3$ pour l'ozone. Calculer le nombre de fois que ce seuil est dépassé pour Toulouse et pour Montpellier sur la période d'étude.

16) [**1.5** pt(s)] En utilisant une boucle `for`, tester l'égalité des niveaux des pollutions entre les deux villes, et ce, pour les quatre polluants vus précédemment. On commencera par décrire, avec soin, dans une boîte `markdown`, la méthodologie envisagée et les hypothèses jugées utiles pour l'analyse.

1.4 Permutations aléatoires

On considère de nouveau les données concernant Montpellier et Toulouse, mais uniquement pour l'ozone (O3). Notons μ_1 la concentration en ozone pour Montpellier et μ_2 celle de Toulouse. On note alors $\delta = \mu_1 - \mu_2$ l'écart entre les moyennes des concentrations pour ce gaz entre les deux villes.

17) [**0.25** pt(s)] Créer un dataframe `df_ozone` ayant deux colonnes nommées `Ville` et `O3` ; calculer δ.

On va se fixer un nombre de répétitions $n_{\text{repet}} = 1000$, et on va appliquer les étapes suivantes, en les répétant n_{repet} fois :

4. Données disponibles ici :
http://josephsalmon.eu/enseignement/datasets/Mesure_journaliere_Region_Occitanie_Polluants_Principaux.csv

- effectuer une permutation aléatoire de la colonne `O3`, en gardant figée la colonne `Ville` (remarque : ce faisant il est possible que des données de Toulouse soient maintenant affectées sur Montpellier et réciproquement).
- calculer alors le nouvel écart entre les moyennes empiriques des concentrations d'O3 entre Montpellier et Toulouse.

On stockera dans un vecteur `deltas` (de taille n_{repet}) les n_{repet} écarts ainsi calculés.

18) [**1** pt(s)] Implémenter la procédure précédente.

19) [**0.5** pt(s)] Pour les écarts ainsi obtenus, afficher avec `subplots` un histogramme (à gauche) et un estimateur à noyaux de la densité (à droite) de cette grandeur. On affichera deux barres verticales (en noir) d'équation $x = -|\delta|$ et $x = |\delta|$.

20) [**1.5** pt(s)] Donner

   a) la proportion de ces écarts qui sont plus grands que $|\delta|$,

   b) la proportion de ces écarts qui sont plus petits que $-|\delta|$.

   Interpréter alors la somme de ces deux proportions.

2 Modèle quadratique

Dans cette partie on reprend les données `cars` vues en cours et disponibles ici : https://forge.scilab.org/index.php/p/rdataset/source/file/master/csv/datasets/cars.csv [5]. On rappelle que les x_i ($i = 1, \dots, 50$) sont les vitesses de voitures (en mph), et y_i ($i = 1, \dots, 50$), sont les distances de freinage (en ft) :

21) Charger les données dans un dataframe `df_cars`.

22) [**0.5** pt(s)] Après avoir chargé le package `statsmodels` avec

```
import statsmodels.formula.api as smf
```

   utiliser `smf.ols` pour ajuster le modèle suivant :

$$y_i = \beta_1^\star x_i^2 + \varepsilon_i, \quad \text{pour } i = 1, \dots, 50 \ .$$

23) [**1** pt(s)] Afficher sur un même graphique les données, les prédictions obtenues pour des vitesses allant de 0 à 30 mph, et les intervalles de confiance (à 95%) en prédiction associés.

24) [**0.5** pt(s)] Reprendre le travail ci-dessus en ajustant cette fois un modèle :

$$y_i = \beta_0^\star + \beta_1^\star x_i^2 + \varepsilon_i, \quad \text{pour } i = 1, \dots, 50 \ .$$

25) [**2** pt(s)] Vérifier mathématiquement que la forme des intervalles de confiance (obtenus dans le premier cas) est conforme à la théorie, sous des hypothèses à préciser.

5. http://josephsalmon.eu/enseignement/TELECOM/MDI720/datasets/cars.csv