Programmer en Python
Licence 2, Mathématiques
V. Monbet
TD 4. Un peu de statistique

1 Changement climatique

On considèreune série temporelle qui décrit les moyennes mensuelles de la température en France entre 1901 et 2015.

1. **Importation de données**
   Importer les données à partir du fichier `tas_2001_2015.csv` disponible sur la page `https://perso.univ-rennes1.fr/valerie.monbet/enseignement.html`.
2. **Visualisation**
   En utilisant la bibliothèque `matplotlib.pyplot`, tracer la série temporelle des températures. Indiquer les années en abscisse. Mettre des légendes d'axes.
   On observe une saisonnalité attendue (alternance d'étés où il fait chaud et d'hivers où il fait froid ...).
   Perçoit-on une augmentation des températures qui pourrait être due au changement climatique ?
3. **Moyennes annuelles**
   Écrire une fonction pour calculer les moyennes annuelles des températures.
   Tracer la série temporelle des moyennes annuelles. Indiquer les années en abscisse. Mettre des légendes d'axes.
   Perçoit-on une augmentation des températures qui pourrait être due au changement climatique ?
4. **Lissage (moyennes mobiles)**
   On a vu sur le graphe de la question précédente que quand on calcule des moyennes annuelles, on a tendance à déstructurer les données.
   Une méthode alternative usuelle consiste à lisser la série des températures en calculant des *moyennes mobiles*. La moyenne mobile m_X de largeur de fenêtre bw (pour bandwidth) est définie, pour tout $bw/2 < k < n - bw/2$ par

$$m_X[k] = \frac{1}{bw} \sum_{i=k-bw/2}^{k+bw/2} X[i]$$

   pour une série temporelle X de longueur n.
   Écrire une fonction `moyenne_mobile` qui calcule la moyenne mobile d'une série temporelle pour une largeur de fenêtre `bw`.

Tester votre fonction sur la série des moyennes mensuelles de température. Tracer successivement la série moyenne mobile obtenue pour les largeurs de fenêtre `bw = 12, 24, 48`.
Perçoit-on une augmentation des températures qui pourrait être due au changement climatique ? A partir de quelle année observe t'on une rupture ?

5. **Modélisation de la tendance**
   On va maintenant essayer de modéliser la tendance au réchauffement entre 1910 et 2005. Nous allons tout d'abord supposer que la tendance est linéaire.
   Utiliser la fonction de régression logistique que vous avez implémentée dans le TP3, pour ajuster un modèle linéaire décrivant le réchauffement en fonction du temps (Y = moyenne mobile de la série de température, X = temps).
   On a observé une rupture au début des années 1980. Ajuster deux modèles linéaires : le premier pour la période 1910-1980 et le second pour la période 1980-2005.

6. **Prédiction**
   Extrapoler les 2 modèles que vous avez construit pour obtenir une prévision de la température moyenne en 2020 puis en 2050.

7. **Significativité de la tendance**
   Nous considérons ici le premier modèle linéaire (celui qui est calibré sur la période 1901-2015). On se demande si la valeur de la pente qu'on a estimée est significativement différente de 0 (ie si le réchauffement est significatif).
   Ceci peut être testé par un *test par permutation*. L'idée est la suivante : si la valeur de la pente est nulle, alors si on permute aléatoirement les années de température, on doit obtenir une estimation de la pente proche de 0 et ceci quelque soit la permutation. Pour simplifier le programme, peut aussi permuter les valeurs sans tenir compte des blocs "années".
   Estimer les valeurs des pentes pour 50 permutations aléatoires des années et comparer l'estimation obtenue à la question 5. à ces valeurs.
   Pour la permutation aléatoire : `np.random.permutation`.

2 Clustering

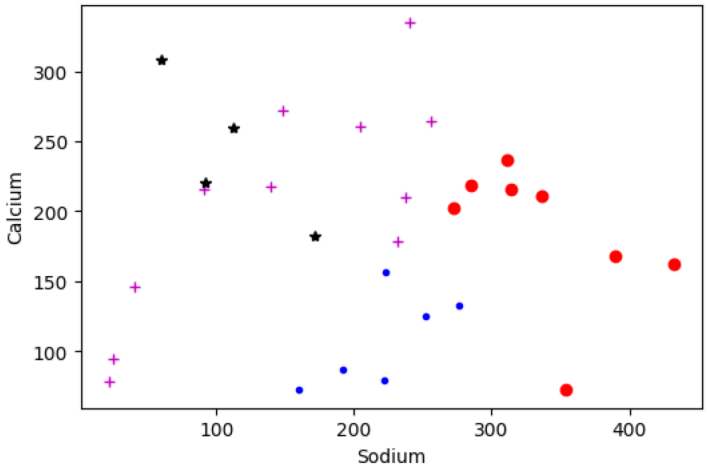
Le partitionnement en k-moyennes (ou k-means en anglais) est une méthode de partitionnement de données. Étant donnés des points et un entier K, le problème est de diviser les points en K groupes, souvent appelés clusters, de façon à minimiser la distance d'un point à la moyenne des points de son cluster.
Dans la suite, nous allons utiliser l'algorithme des k-moyennes (voir ci-dessous) pour obtenir une partition de fromages ie on veut construire, automatiquement, un petit nombre de groupes (ou clusters) tels que les fromages qui appartiennent au même groupe se ressemblent et les fromages qui appartienne à des groupes différents sont différents.

Algorithme des k-moyennes

Initialisation : Choisir aléatoirement K points qui représentent les K centres de classes (représentés par la moyenne dans la classe) : $G_1, \cdots, G_K$
Répéter jusqu'à ce qu'il y ait convergence :
1. assigner chaque observation à la partition la plus proche (ceci nécessite, pour chaque individu et chaque classe, de calcul la distance entre l'individu et le centre de la classe) ; plus précisément, pour chaque cluster k et chaque individu i, calculer $dist(X[i,:], G[k,:])$ puis attribuer la classe ℓ à i ($C[i] = \ell$) si $dist(X[i,:], G[\ell,:]) \leq dist(X[i,:], G[k,:])$ pour tout k.
2. mettre à jour les centres de classes $G_1, \cdots, G_K$ ie pour chaque classe estimer la moyenne des observations de la classe.

1. **Importation des données**
   Importer les données à partir du fichier `fromages.dat` disponible sur la page `https://perso.univ-rennes1.fr/valerie.monbet/enseignement.html`.
   Stocker les données dans un tableau (`np.array`) avec en ligne les différents fromages (ie les individus) et en colonnes leurs caractéristiques (ie les variables).
   On note `n` le nombre de fromages.
2. **Critère d'arrêt**
   Proposer un critère d'arrêt pour la boucle "répéter".
3. **Tirage aléatoire de nombres entiers**
   La fonction `np.random.randint` qui prend comme argument `low` (plus petit entier), `high` plus grand entier, `K` nombre d'entier à tirer aléatoirement entre `low` et `high`. Tester cette fonction pour les valeurs `low` $= 0$, `high` $=$ `n` et `K` $= 4$.
4. **Lister les indices des points appartenant au cluster** k
   A quoi sert la fonction `np.where` ? Que renvoie t'elle ? Comment l'utiliser pour lister les indice des points appartenant à une classe C ?
5. **k-means**
   Implémenter l'algorithme ci-dessus dans une fonction `kmeans`. La fonction doit renvoyer le numéro de cluster de chaque individu et les centres de classe.
   Rq : la fonction `np.mean` permet de calculer la moyenne de chaque colonne d'une matrice `X` : `mX = np.mean(X,axis=0)`.
6. **Partition des fromages**
   Utiliser la fonction `kmeans` pour regrouper les fromages en 4 groupes.

Pour visualiser les résultats, on peut tracer un nuage de points de 2 variables (par exemple les sodium et le calcium), avec une couleur différente par cluster.



Il est aussi intéressant de lister les fromages de chaque groupe.

7. **Centrer et réduire**
   Si les variables prennent des valeurs très différentes (différence d'ordre de grandeur), les variables à valeurs fortes prennent un poids trop fort dans le calcul de la distance euclidienne. Il est alors d'usage de centrer et réduire les variables.
   - centrer = retirer la moyenne (pour se ramener à une moyenne nulle)
   - réduire = diviser par l'écart-type (pour se ramener à une écart-type 1)

```
X_cr = X.copy()
for k in range(X.shape[1]):
    mk = np.mean(X[:,k])
    sk = np.std(X[:,k])
    X_cr[:,k] = (X[:,k]-mk)/sk
```

   Tester l'algortihme des k-moyennes sur la matrice des données centrées et réduites.

8. **Classification hiérarchique ascendante**
   En Python, l'algorithme des kmeans est disponible dans les modules `scipy` et `sklearn`. Il existe aussi d'autres algorithmes de classification comme la classification hiérarchique ascendante par exemple. Cet algorithme permet des représentations graphiques intéressantes (mais est trop coûteux pour les gros jeux de données).

```
from scipy.cluster.hierarchy import dendrogram, linkage
# Générer la matrice des liens
Z = linkage(X_cr,method='ward',metric='euclidean')
# affichage du dendrogramme
plt.title("CAH")
dendrogram(Z,labels=fromages,orientation='left',color_threshold=0)

# Pour matérialiser le découpage en 4 classes
# (découpage à la hauteur t = 6)
```

```
plt.title('CAH avec matérialisation des 4 classes')
dendrogram(Z,labels=fromages,orientation='left',color_threshold=6)
```