Big data I

Big data, *intelligence artificielle,* deep learning, *réseau de neurones,* machine learning... *plein de mots compliqués ! Le but commun est de faire exécuter à un ordinateur de tâches de plus en plus complexes :* choisir *(par exemple trouver un bon élément parmi des milliards selon plusieurs critères),* décider *(séparer des photos de chats de photos de voitures),* prévoir *(un malade a de la fièvre et le nez qui coule, quelle maladie est la plus probable ?). Dans cette première partie on va utiliser des outils classiques de statistique et de probabilité pour résoudre des problèmes amusants.*

Cours 1 (Des données par milliers)**.**
Lorsque l'on parle de *big data* on parle de fichiers avec des milliards de données. On va plus modestement traiter de fichiers contenant les données (fictives) de 100 000 personnes.
Récupères le fichier `personnes_100000.csv`. Tu disposes aussi des fichiers `personnes_100.csv`, `personnes_1000.csv`, `personnes_10000.csv` qui contiennent moins d'entrées et sont idéaux pour tester les programmes.

Ces fichiers sont disponibles ici :

github.com/exo7math/python2-exo7

Voici un exemple de ligne de ce fichier au format *csv* (*comma separated values*) :

```
femme,Lessard,Capucine,7/31/1978,LA ROCHELLE,56.7,155,O+
```

ou

```
homme,Cadieux,Antoine,11/20/1938,METZ,78.8,166,A-
```

Chaque ligne désigne une personne et ses caractéristiques :

- sexe (homme/femme)
- nom
- prénom
- date de naissance (format jj/mm/aaaa)
- ville
- poids (en kilogrammes)
- taille (en centimètres)
- groupe sanguin

Activité 1 (Sondage)**.**

Objectifs : utiliser un échantillon pour déterminer les caractéristiques d'une population.

1. Programme une fonction `age_moyen(debut,fin,fichier)` qui évalue l'âge moyen d'une liste de personnes (contenue dans le fichier donné) à partir d'un échantillon. Par exemple :

`age_moyen(10,20,"personnes_100.csv")` renvoie la moyenne des âges des personnes de rang 10 (compris) à 20 (exclu) parmi une liste de 100 personnes.

Indication. L'âge d'une personne est l'âge qu'elle aura à la fin de l'année en cours.

Compare l'âge moyen calculé à partir de l'échantillon avec l'âge moyen de toute la liste. Quelle taille de l'échantillon permet d'avoir une estimation à 1 an près ?

2. Programme une fonction `probabilite_initiale(lettre,debut,fin,fichier)` qui estime la probabilité que le nom d'une personne commence par la lettre donnée à partir d'un échantillon. Pour cela on approche la probabilité par la formule :

$$\text{probabilité} \quad \simeq \quad \frac{\text{nombre d'occurences}}{\text{nombre total d'éléments}}$$

3. Programme une fonction `probabilite_groupe_sanguin(debut,fin,fichier)` qui renvoie les probabilités de chaque groupe sanguin. La fonction renvoie un dictionnaire dont le couple « clé/valeur » est « groupe sanguin/probabilité ». Les groupes possibles sont A+, A-, B+, B-, O+, O-, AB+ et AB-. Par exemple (obtenu sur un échantillon de 10 personnes seulement) :

```
{ 'A+': 0.1, 'A-': 0.1, 'B+': 0.1, 'B-': 0.1,
'O+': 0.3, 'O-': 0.3, 'AB+': 0.0, 'AB-': 0.0 }
```

Quels sont les groupes les plus fréquents ?

Cours 2 (Ordre alphabétique)**.**

On rappelle que `Python` connaît l'ordre alphabétique et qu'on peut comparer deux chaînes de caractères :

- « `"A" < "B"` » est « Vrai » (`Python` renvoie `True`),
- « `"BAC" < "ABC"` » est « Faux »,
- « `"A" == "a"` » est « Faux ».

Tu peux consulter la fiche « Le mot le plus long » et aussi la fiche « Tri - Complexité » pour plus d'informations et une activité similaire à l'activité suivante.

Activité 2 (Chercher dans une liste de noms)**.**

Objectifs : chercher dans une liste élément par élément ou bien par dichotomie.

1. **Liste triée.** Programme une fonction `fichier_vers_liste_noms(fichier)` qui renvoie la liste *ordonnée* des noms issus du fichier.

   Indication. `liste.sort()` trie la liste.

2. **Début d'une chaîne.** Programme une fonction `est_debut(debut,chaine)` qui teste si une chaîne débute par les caractères donnés. Par exemple :
   - `est_debut("ABC","ABCDEF")` renvoie « Vrai »,
   - `est_debut("XYZ","ABCDEF")` renvoie « Faux »,
   - `est_debut("ABCD","AB")` renvoie « Faux ».

3. **Recherche séquentielle.** Programme une fonction `chercher_1(liste,debut)` qui renvoie un nom de la liste commençant par `debut` (ou `None` si un tel nom n'existe pas).

   Méthode. Parcours un par un les noms de la liste (issue de la première question) et teste s'il commence par `debut`.

   Par exemple `chercher_1(liste,"Bri")` peut renvoyer `'Brian'`.

4. **Recherche dichotomique.** Programme une fonction `chercher_2(liste,debut)` qui fait le même travail mais avec un algorithme plus efficace : la dichotomie.

Algorithme.
- — Entrée : un début de mot à trouver et une liste ordonnée de noms.
  — Sortie : un mot trouvé dans la liste commençant par le début souhaité ou `None` en cas d'échec.
- $a \leftarrow 0$ (le rang d'une liste commence à 0).
- $b \leftarrow n-1$ où n est la longueur de la liste.
- Tant que $b \geqslant a$, faire :
  — $k \leftarrow (a+b)//2$
  — Si `debut` est le début du mot `liste[k]` alors renvoyer `liste[k]`.
  — Si `debut` vient après `liste[k]` dans l'ordre alphabétique alors faire $a \leftarrow k+1$,
  — sinon faire $b \leftarrow k-1$.
- Renvoyer `None` (c'est le cas uniquement si aucun nom n'a été trouvé dans la boucle précédente).

Par exemple `chercher_2(liste,"Bri")` peut renvoyer `'Brisebois'`. Le nom trouvé par cet algorithme n'est pas nécessairement le premier qui viendrait dans l'ordre alphabétique.

5. **Complexité.**
   (a) Modifie tes deux fonctions de recherche pour qu'elles renvoient en plus du nom, le nombre d'itérations nécessaires (par exemple le nombre de fois où tu effectues un appel à la fonction `est_debut()`).
   (b) Vérifie expérimentalement que si n est la longueur de la liste ordonnée, alors :
   - pour la recherche séquentielle, il peut y avoir jusqu'à n itérations,
   - pour la recherche par dichotomie, il peut y avoir jusqu'à $E(\log_2(n)+1)$ itérations (où $E(x)$ désigne la partie entière, comme la commande `floor(x)`).

Activité 3 (La formule des tanks)**.**

Objectifs : déterminer la taille N d'une série $1,\dots,N$ en ne connaissant que quelques numéros tirés au hasard.

En plein milieu de la seconde guerre mondiale les Allemands produisent un nouveau tank plus performant. Les Alliés s'inquiètent car ils ne savent pas combien de ces nouveaux tanks sont produits. Les services de renseignements estiment la production à 1500 tanks par mois. Que disent les mathématiques ? Les Alliés ont intercepté 4 tanks produits le même mois et qui portent les numéros :

143 77 198 32

Combien de tanks ont été produit ce mois ?
Modélisation. Sachant que les tanks sont numérotés de 1 à N chaque mois, à quelle valeur peut être estimée la production mensuelle N, connaissant un échantillon de k numéros $[n_1, n_2, \dots, n_k]$?

1. **La formule des tanks.** On note m le maximum des éléments de l'échantillon. On note k la taille de l'échantillon. Alors la formule des tanks estime :
$$N \simeq m + \frac{m}{k} - 1$$
Programme cette formule en une fonction `formule_tanks(echantillon)` qui renvoie cette estimation de N. Quelle est ton estimation pour le nombre de tanks ?
2. **Le double de la moyenne.** On peut essayer d'autres estimations. Par exemple on peut estimer N comme le double de la moyenne de l'échantillon. Programme une fonction

`double_moyenne(echantillon)` qui renvoie cette nouvelle estimation. Compare avec la formule des tanks.

Pour tester l'efficacité de la formule des tanks on va faire le cheminement inverse : on fixe un entier N, on choisit au hasard un échantillon de k éléments et on regarde si nos formules permettent de bien approcher N.

3. **Tirage sans remise.** Programme une fonction `tirage_sans_remise(N,k)` qui renvoie une liste de k entiers différents compris entre 1 et N (inclus).
4. **Erreurs.** Programme une fonction `erreurs(N,k)` (ou mieux `erreurs(N,k,nb_tirages=1000)`) qui calcule l'erreur moyenne commise par nos formules. Pour cela :
   - Effectue un tirage sans remise de k entiers plus petits que N.
   - Calcule la valeur N_1 obtenue à partir de cet échantillon par la formule des tanks.
   - Calcule l'erreur commise $e = |N - N_1|$.

   En faisant ceci pour un grand nombre de tirages, calcule l'erreur moyenne commise. Fais le même travail avec l'autre formule et renvoie les deux erreurs moyennes.

   Pour 20 entiers plus petits que 1000 ($k = 20$ et $N = 1000$) quelle est la meilleure formule et à quelle erreur peut-on s'attendre ?

À la fin de la guerre les registres Allemands ont été récupérés et indiquaient une production de 245 chars mensuels ! Cette formule est aussi utilisée pour estimer la production d'un produit (par exemple d'un téléphone) à partir des numéros de série.

Cours 3 (Le problème du secrétaire)**.**
La directrice d'une entreprise doit choisir son nouveau secrétaire : elle reçoit $k = 100$ secrétaires un par un et attribue à chacun une note (par exemple un entier entre 0 et $N = 100$). Elle veut choisir le meilleur secrétaire mais il y a une contrainte : elle décide immédiatement après chaque entretien si elle embauche ou pas ce secrétaire. Elle n'a pas la possibilité de revenir en arrière.
Voici la stratégie qu'elle adopte : elle commence par recevoir un certain nombre de candidats (par exemple 25), elle mémorise juste la meilleure note obtenue jusqu'ici sans retenir aucun de ces candidats. Ensuite elle reçoit les candidats suivants et elle sélectionne le premier qui a une note supérieure ou égale à la meilleure note de l'échantillon. Avec cette stratégie elle n'est pas sûre de trouver le meilleur secrétaire, et elle peut même ne sélectionner aucun secrétaire.

Exemple. Voici une liste de scores des candidats (ici avec seulement 10 candidats) :

```
liste = [2,5,3,4,1,6,4,5,8,3]
```

Prenons le pourcentage $p = 25$. La taille de la liste est $k = 10$ donc l'échantillon de 25% est formé des 2 premiers éléments $[2, 5]$. Le score maximum de cet échantillon est $M = 5$. La directrice ne retient pas de candidat dans l'échantillon, par contre elle va choisir le premier candidat suivant dont la note sera supérieure ou égal à M et arrête le processus. Ici c'est donc le candidat avec un score de 6 qui est choisi. Note qu'elle n'a pas sélectionné le meilleur candidat avec un score de 8 mais qui était en fin de liste. Le but est de choisir la bonne taille pour l'échantillon : avec un échantillon trop petit elle va choisir un secrétaire moyen, avec un échantillon trop grand elle ne va pas trouver de secrétaire du tout.

Notations.
- Le nombre de candidats secrétaires est k. Par défaut $k = 100$.
- Les notes vont de 0 à N. Par défaut $N = 100$.

- La taille de l'échantillon s'exprime comme un pourcentage p du nombre total de candidats k total. Par exemple $p = 25$ signifie que l'on teste d'abord 25% de candidats.

Activité 4 (Le problème du secrétaire).

Objectifs : programmer la sélection d'un bon secrétaire et optimiser la taille de l'échantillon.

1. Programme une fonction `genere_liste(k,N)` qui génère une liste de k entiers tirés au hasard entre 0 et N (deux nombres peuvent être identiques). Cela correspond aux notes des k secrétaires.
2. Programme une fonction `choix_secretaire(liste,p)` qui à partir d'une liste de notes et un pourcentage renvoie le score du secrétaire choisi (ou `None` si aucun ne convient). La méthode adoptée est celle décrite dans le cours au-dessus :
   - à partir de l'échantillon formé par les premiers candidats (la taille de l'échantillon est donnée sous la forme d'un pourcentage `p`) on retient le meilleur score M de cet échantillon, mais aucun n'est sélectionné,
   - en recevant un par un les candidats suivants, on prend le premier ayant un score supérieur ou égal à M,
   - si aucun ne convient on renvoie `None`.
3. On souhaite savoir si notre stratégie est efficace ou pas : combien de fois sélectionne-t-on le meilleur secrétaire possible ? Pour cela tu vas tester la méthode sur un grand nombre de tirages.

   Programme une fonction `meilleurs_secretaires(k,N,p,nb_tirages)` qui renvoie le nombre de fois où l'algorithme de la directrice sélectionne le meilleur candidat.
   - k est la longueur des listes,
   - chaque note est un entier entre 0 et N (inclus),
   - p est le pourcentage qui détermine la taille de l'échantillon,
   - `nb_tirages` est le (grand) nombre de listes aléatoires à tester.

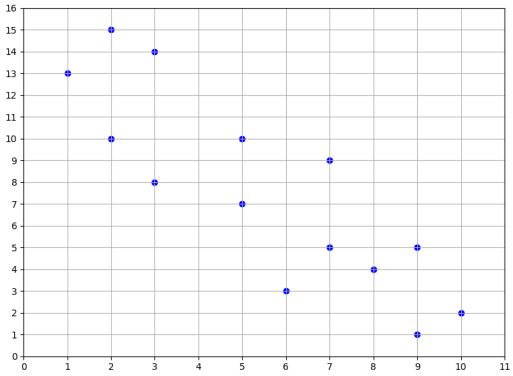
   Par exemple avec $k = 100$, $N = 100$ et le pourcentage $p = 25$ en effectuant de nombreux tirages (au moins 1000) le candidat sélectionné est dans environ 47% des cas le meilleur des candidats de toute la liste.
4. Le pourcentage de 25% pour l'échantillon n'est pas celui qui conduit aux meilleurs résultats. Écris un programme ou bien tâtonne pour trouver la meilleure valeur de p possible.

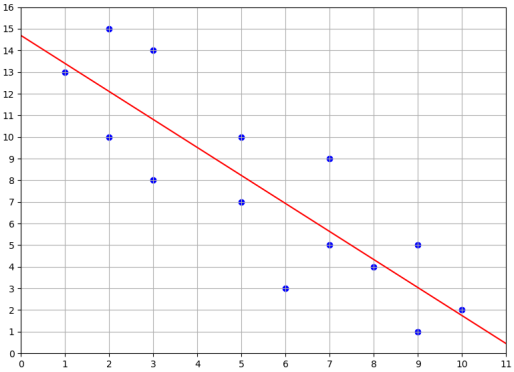
   Indice. La réponse s'appelle la loi en $1/e$!

Cours 4 (Régression linéaire).
On se donne des points $(x_i, y_i)_{1 \leqslant i \leqslant n}$. On cherche si on peut modéliser la situation par une relation linéaire entre l'abscisse et l'ordonnée du type :

$$y = ax + b$$

C'est-à-dire que l'on cherche la droite qui « approche » au mieux tous les points. Cette opération s'appelle la ***régression linéaire*** et la droite est la ***droite des moindres carrés***.
Sur la figure ci-dessous à gauche une série de points (x_i, y_i), à droite la droite des moindres carrés d'équation $y = ax + b$.





Voici comment calculer les coefficients a et b de la droite $y = ax + b$.

- On note m_x la moyenne des $(x_i)_{1\leqslant i\leqslant n}$.
- On note m_y la moyenne des $(y_i)_{1\leqslant i\leqslant n}$.
- On note $\mathrm{Var}(x)$ la ***variance*** des $(x_i)_{1\leqslant i\leqslant n}$:

$$\mathrm{Var}(x) = \frac{1}{n}\sum_{i=1}^{n}(x_i - m_x)^2$$

c'est-à-dire :

$$\mathrm{Var}(x) = \frac{1}{n}\left((x_1 - m_x)^2 + (x_2 - m_x)^2 + \cdots + (x_n - m_x)^2\right)$$

La variance mesure l'écart des valeurs avec la moyenne. La variance est aussi le carré de l'écart-type.

- On note $\mathrm{Cov}(x, y)$ la ***covariance*** des $(x_i)_{1\leqslant i\leqslant n}$ avec $(y_i)_{1\leqslant i\leqslant n}$:

$$\mathrm{Cov}(x, y) = \frac{1}{n}\sum_{i=1}^{n}(x_i - m_x)(y_i - m_y)$$

c'est-à-dire :

$$\mathrm{Cov}(x, y) = \frac{1}{n}\left((x_1 - m_x)(y_1 - m_y) + (x_2 - m_x)(y_2 - m_y) + \cdots + (x_n - m_x)(y_n - m_y)\right)$$

La covariance mesure la corrélation (c'est-à-dire la dépendance) entre les valeurs x_i et y_i.

- Alors les coefficients de la droite $y = ax + b$ sont :

$$a = \frac{\mathrm{Cov}(x, y)}{\mathrm{Var}(x)} \qquad \text{et} \qquad b = m_y - am_x$$

Activité 5 (Régression linéaire)**.**

Objectifs : tracer la droite des moindres carrés.

1. **Moyenne, variance, covariance.**

   Programme :
   - une fonction `moyenne(liste)` qui renvoie la moyenne des éléments (x_i) de la liste,
   - une fonction `variance(liste)` qui renvoie la variance des éléments (x_i) de la liste,
   - une fonction `covariance(listex,listey)` qui renvoie la covariance des éléments (x_i) et (y_i).

   Exemples.
   - Vérifie que la moyenne de $(1, 2, 3, 4, 5)$ est $m_x = 3$ et la variance $\mathrm{Var}(x) = 2$.
   - Vérifie que la covariance entre $(1, 2, 3, 4, 5)$ et $(4, 5, 4, 7, 6)$ vaut $\mathrm{Cov}(x, y) = 1.2$.
   - Vérifie sur un exemple que $\mathrm{Cov}(x, x) = \mathrm{Var}(x)$.

2. **Régression linéaire.**

Programme une fonction `regression_lineaire(points)` qui à partir d'une liste de points $(x_i, y_i)_{1 \leqslant i \leqslant n}$ renvoie les coefficients a, b de la droite d'équation $y = ax + b$.

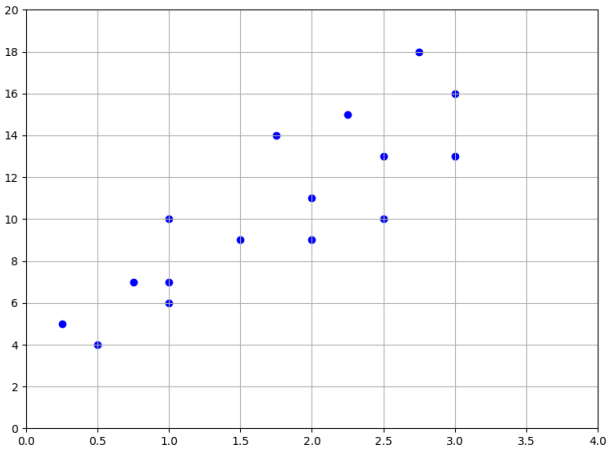
Exemple. Pour le bac blanc on a demandé à chaque élève le temps qu'il a passé pour ses révisions (valeur x_i) et la note qu'il a obtenue (valeur y_i).

Voici la liste (x_i, y_i) des données, par exemple le premier élève à réviser 0.25 heures et a obtenu 5, le dernier à réviser 3 heures et a obtenu 16.

```
eleves = [ (0.25,5), (0.5,4), (0.75,7), (1,6), (1,7),
           (1,10), (1.5,9), (1.75,14), (2,9), (2,11), (2.25,15),
           (2.5,10), (2.5,13), (2.75,18), (3,13), (3,16) ]
```

Calcule les coefficients a et b associés à la régression linéaire.

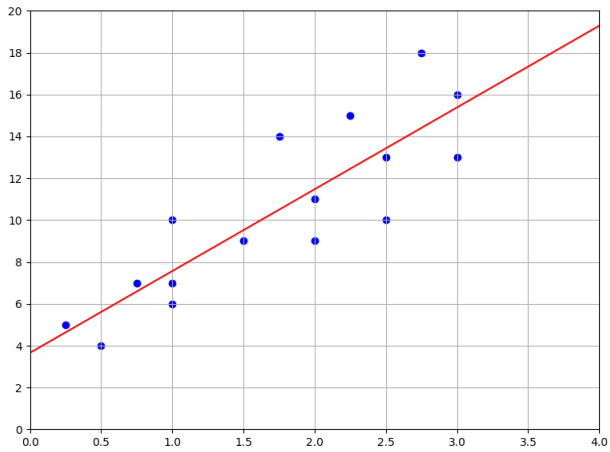
Voici ces points, en abscisse le temps de travail (en heures) et en ordonnée la note obtenue (sur 20) :



J'ai révisé pendant 2 heures, quelle note puis-je espérer ?

3. **Affichage.**

   Programme une fonction `afficher(points)` qui réalise l'affichage des points et de la droite des moindres carrés d'équation $y = ax + b$.



4. **Travailler plus pour gagner plus.**

   Écris un petit programme qui demande à l'utilisateur « Quelle note aimerais-tu avoir ? » et à partir de la réponse donnée affiche une phrase du type « Tu dois travailler au moins 2 heures et 30 minutes. »

   Indications.

- `input()` attend de l'utilisateur une réponse qui est renvoyée sous la forme d'une chaîne de caractères.
- `float(chaine)` transforme une chaîne en un nombre flottant, par exemple `float("12.5")` renvoie `12.5`.

Cours 5 (Distribution de Gauss)**.**

Distribution de Gauss.

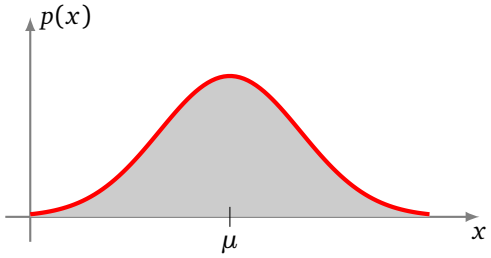
Certaines données se répartissent suivant une ***distribution de Gauss*** (appelée aussi ***loi normale***). Une distribution de Gauss est déterminée par deux paramètres :

- l'espérance (ou la moyenne) μ,
- la variance (ou l'écart-type au carré) σ^2,

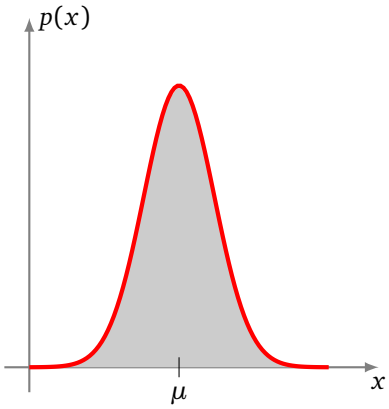
et se calcule par la formule :

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}\right)$$

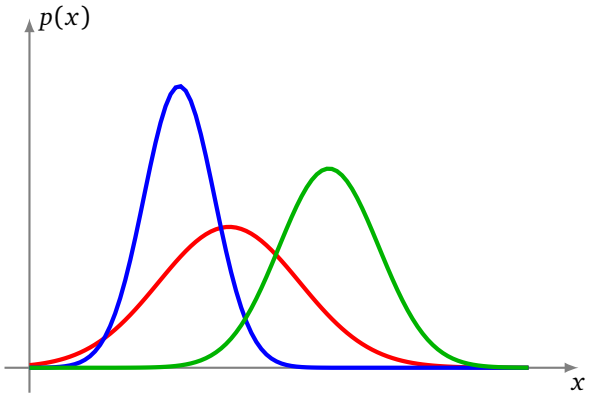
Le graphe de cette fonction est une courbe en cloche, centrée sur l'espérance μ, et qui s'étale plus ou moins en fonction de la variance σ^2.



Cas $\mu = 4$, $\sigma^2 = 2$



Cas $\mu = 3$, $\sigma^2 = \frac{1}{2}$



La fonction $x \mapsto p(x)$ s'appelle une ***densité de probabilité***. $p(x)$ n'est pas une probabilité et on peut avoir $p(x) > 1$ pour certains x. Par contre l'aire sous la courbe vaut toujours 1.

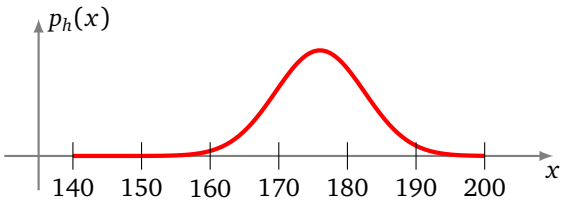
Détermination de la distribution à partir d'un échantillon.

Si on nous donne un échantillon de données $x_1, \ldots, x_n$ alors on peut lui associer une distribution de Gauss en prenant :

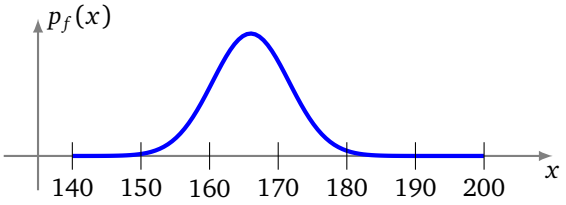
- μ la moyenne de x_i,

- σ^2 la variance des x_i.

Exemple de la taille. Voici une liste de tailles de 5 hommes : $[181, 170, 186, 175, 169]$. À partir de cet échantillon on calcule la moyenne $\mu_h = 176$ (arrondie à 1 cm près) et $\sigma_h^2 = 42$. (Tu calculeras les vraies valeurs dans l'activité suivante.) On peut calculer la distribution de Gauss de la taille des hommes (voir le graphique ci-dessous).
On peut faire la même chose à partir de l'échantillon suivant de tailles de femmes : $[162, 174, 160, 171, 162]$. On trouve $\mu_f = 166$ et $\sigma_f^2 = 32$.
On obtient donc deux densités de probabilité : $p_f(x)$ pour les hommes et $p_f(x)$ pour les femmes.

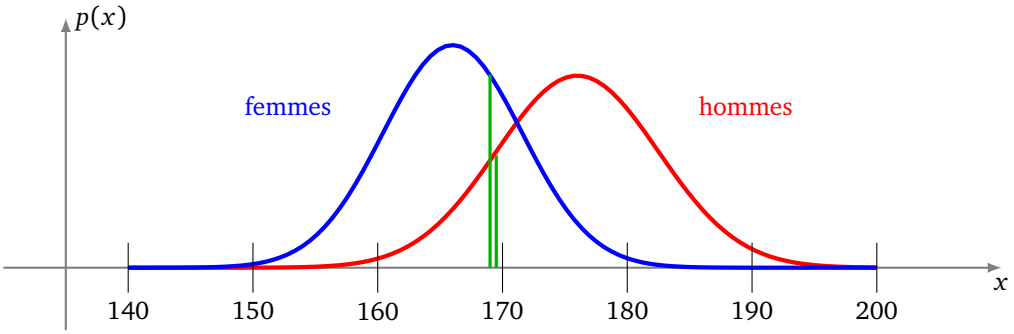


Distribution de la taille des hommes



Distribution de la taille des femmes

Classification.
On veut savoir si quelqu'un de taille $x = 169$ cm est plutôt un homme ou une femme ? On a donc deux densités de probabilité : on calcule $p_h(x)$ (comme si c'était un homme) et $p_f(x)$ (comme si c'était une femme) et on compare c'est deux valeurs. Si $p_h(x) > p_f(x)$ alors c'est plus probablement un homme, sinon c'est plutôt une femme.
Ici on calcule $p_h(x) \simeq 0.035$ et $p_f(x) \simeq 0.061$, donc avec nos données, quelqu'un de 169 cm est plus probablement une femme.
On peut aussi le voir graphiquement ci-dessous : pour $x = 169$ la courbe des femmes est au-dessus de la courbe des hommes.



Distribution hommes et femmes

Activité 6 (Classification bayésienne naïve)**.**

Objectifs : classer une donnée dans une catégorie ou une autre, par exemple décider si une personne est un homme ou une femme connaissant sa taille et son poids.

1. **Moyenne et variance.** Détermine la moyenne μ_h et la variance σ_h^2 de la taille des hommes à partir de cet échantillon de taille (en cm) :

```
taille_hommes = [172,165,187,181,167,184,168,174,180,186]
```

Même chose avec μ_f et σ_f^2 pour les femmes :

```
taille_femmes = [172,156,164,182,171,164,162,170,161,167]
```

2. **Densité de probabilité.** Programme une fonction `densite_gauss(x,mu,sigma2)` qui renvoie

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}\right)$$

où `mu` $= \mu$ est une espérance (notre moyenne) et `sigma2` $= \sigma^2$ est une variance.

3. **Homme ou femme par la taille.**

On donne une taille x, par exemple $x = 170$, on souhaite savoir si c'est plus probablement un homme ou une femme. Pour cela calcule la densité de probabilité $p_h(x)$ associée à μ_h et σ_h^2 et la densité de probabilité $p_f(x)$ associée à μ_f et σ_f^2.

Si $p_h(x) > p_f(x)$ alors il est plus probable que ce soit un homme, sinon c'est plutôt une femme.

Remarques.

- Ce n'est bien sûr pas une certitude ! On va faire mieux dans la question suivante en prenant aussi en compte le poids.
- Les nombres sont très petits, il peut être plus parlant de regarder si $p_h(x)/p_f(x)$ est plus grand ou plus petit que 1.

4. **Par la taille et le poids.**

Voici des échantillons de taille/poids (en cm et kg) pour des hommes puis des femmes.

```
hommes = [(172,68),(165,71),(187,85),(181,73),(167,75),
     (184,93),(168,67),(174,83),(180,70),(186,73)]
femmes = [(172,66),(156,57),(164,48),(182,71),(171,55),
     (164,68),(162,52),(170,68),(161,76),(167,67)]
```

Pour une taille x, on a maintenant une probabilité $p_h^{\text{taille}}(x)$ et $p_f^{\text{taille}}(x)$. En calculant des moyennes et des variances on obtient pour un poids y une probabilité $p_h^{\text{poids}}(y)$ et $p_f^{\text{poids}}(y)$. On multiplie les probabilités pour déterminer si une donnée correspond plutôt à un homme ou une femme. On prend une personne de (taille, poids) $= (x, y)$. Si on a :

$$p_h^{\text{taille}}(x) \cdot p_h^{\text{poids}}(y) \;>\; p_f^{\text{taille}}(x) \cdot p_f^{\text{poids}}(y)$$

alors c'est plus probablement un homme, sinon c'est plutôt une femme.

Une personne de taille 176 cm pesant 64 kg est plutôt un homme ou une femme ?

Activité 7 (Classification bayésienne naïve (suite))**.**

Objectifs : classer des phrases dans une catégorie en fonction des mots qu'elle contient.

Voici une liste de titres de sport :

```
titres_sport = [
"un beau match de championnat",
"victoire de Paris en finale",
"défaite à Marseille",
"le coach viré après la finale",
"Paris change de coach"]
```

et des titres ne concernant pas le sport :

```
titres_passport = [
"un beau printemps à Paris",
"un robot écrase un chien à Marseille",
"célébration de la victoire de la grande guerre",
"grève finale au lycée"]
```

À partir de ces titres déjà classés sport/pas sport tu vas faire déterminer par l'ordinateur si les phrases suivantes parlent de sport ou pas :

`"victoire de Marseille"`
`"un beau chien"`
`"Paris écrase Barcelone en finale"`

1. **Mots.** Programme une fonction `liste_mots(titres)` qui renvoie la liste de tous les mots à partir d'une liste de titres.

   Voici le début de la liste obtenue à partir des titres de sports :

   `['un','beau','match','de','championnat','victoire','de','Paris',...]`

2. **Probabilité d'un mot.** La probabilité qu'un mot m donné soit dans une liste de mots se calcule par la formule :

   $$p(m) = \frac{\text{nombre d'occurences du mot } m}{\text{nombre total de mots}}$$

   Par exemple le mot $m =$ `"Paris"` est présent 2 fois parmi les 23 mots des titres de sport. Ainsi la probabilité vaut $p(m) = \frac{2}{23} = 0.086\ldots$

   Programme une fonction `probabilite_mot(mot,liste_mots)` qui renvoie la probabilité du mot donné par rapport à une liste des mots.

3. **Probabilité d'une phrase.**

   On définit la probabilité associée à une phrase comme le produit des probabilités des mots qu'elle contient (une liste de mots de titres étant donnée). Si une phrase est formée des mots $m_1, m_2, \ldots, m_k$ alors la probabilité de la phrase est :

   $$p = p(m_1) \cdot p(m_2) \cdots p(m_k)$$

   Par exemple la phrase `"la finale de Paris"` est formée des mots `"la"`, `"finale"`, `"de"` et `"Paris"` qui apparaissent respectivement 1, 2, 3 et 2 fois, pour un total des 23 mots des titres de sport. La probabilité associée à la phrase est donc

   $$p = \frac{1}{23} \times \frac{2}{23} \times \frac{3}{23} \times \frac{2}{23} = 0.0000428\ldots$$

   Programme une fonction `probabilite_phrase(phrase,liste_mots)` qui renvoie la probabilité de la phrase donnée par rapport à une liste de mots donnée.

   Application : sport ou pas ?

   Pour savoir si la phrase `"la finale de Paris"` parle de sport ou pas :

   - on calcule la probabilité p(phrase|sport) de la phrase donnée par rapport à la liste des mots de sport,
   - on calcule la probabilité p(phrase|pas sport) de la phrase donnée mais cette fois par rapport à la liste des mots des titres ne parlant pas de sport,
   - si p(phrase|sport) $>$ p(phrase|pas sport) alors il probable que la phrase concerne le sport.

   Exemple.

   La phrase est `"la finale de Paris"` :

- on a vu $p(\text{phrase}|\text{sport}) = 0.0000428\ldots$
- on calcule de même :
$$p(\text{phrase}|\text{pas sport}) = \frac{2}{24} \times \frac{1}{24} \times \frac{2}{24} \times \frac{1}{24} = 0.0000120\ldots$$
- Comme $p(\text{phrase}|\text{sport}) > p(\text{phrase}|\text{pas sport})$ alors la phrase concerne probablement du sport (même si les nombres sont petits, l'un est 3 fois plus grand que l'autre).

4. **Probabilité modifiée d'un mot.** On a un problème avec la phrase `"le coach perd la finale"` car le mot `"perd"` n'apparaît pas dans nos titres, donc la probabilité de ce mot est $p(m) = 0$. Aussi lorsque l'on calcule la probabilité de la phrase, on obtient $p(\text{phrase}|\text{sport}) = 0$ et $p(\text{phrase}|\text{pas sport}) = 0$ (car on a à chaque fois un facteur nul).

   On remédie à ce problème avec une probabilité modifiée et jamais nulle. Elle est définie par :
$$\tilde{p}(m) = \frac{\text{nombre d'occurences du mot } m \quad + 1}{\text{nombre total de mots}}$$
   On a fait comme si n'importe quel mot apparaissait au moins une fois (ce que l'on obtient n'est plus vraiment une probabilité, mais résout notre problème).

   Modifie tes fonctions précédentes en :
   - `probabilite_mot_bis(mot,liste_mots)` qui renvoie la probabilité modifiée $\tilde{p}$ d'un mot,
   - `probabilite_phrase_bis(phrase,liste_mots)` qui renvoie la probabilité de la phrase comme le produit des probabilités modifiées de chaque mot.

   Exemple.

   La phrase est `"le coach perd la finale"` :
   - $\tilde{p}(\text{phrase}|\text{sport}) = \frac{1+1}{23} \times \frac{2+1}{23} \times \frac{0+1}{23} \times \frac{1+1}{23} \times \frac{2+1}{23} \simeq 5.59 \cdot 10^{-6}$
   - $\tilde{p}(\text{phrase}|\text{pas sport}) = \frac{0+1}{24} \times \frac{0+1}{24} \times \frac{0+1}{24} \times \frac{2+1}{24} \times \frac{1+1}{24} \simeq 7.53 \cdot 10^{-7}$
   - Comme $\tilde{p}(\text{phrase}|\text{sport}) > \tilde{p}(\text{phrase}|\text{pas sport})$ la phrase concerne très probablement du sport (le premier nombre est 7 fois plus grand que le second).

   Conclusion. Les phrases suivantes parlent-elles de sport ou pas ?

   `"victoire de Marseille"`

   `"un beau chien"`

   `"Paris écrase Barcelone en finale"`

C'est aussi avec cette méthode de classification que les courriers électroniques sont filtrés *spam* ou *pas spam*.