TRAVAUX PRATIQUES DE GENIE INFORMATIQUE

1 • BUT DE LA SEANCE.

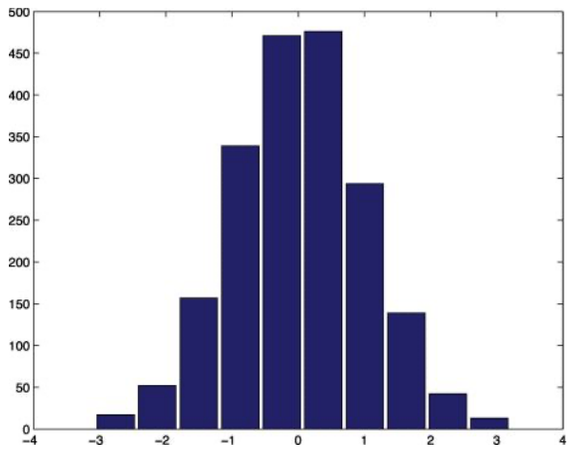
Il s'agit maintenant pour vous d'être capable de créer une fonction avec très peu de directive et de la tester. Vous allez créer une fonction qui calcule un histogramme. Vous pouvez bien sûr comparer ce que vous obtenez avec la fonction qui calcule les histogrammes proposée par Python (`pyplot.hist()` de matplotlib).

2 • QUELQUES RAPPELS.

2.1 • Histogramme.

D'après Wikipedia :

"Un histogramme est un graphique permettant de représenter la répartition d'une variable **continue** de façon discrète en la représentant avec des colonnes verticales".



Exemple d'histogramme.

Il existe plusieurs façon de calculer un histogramme à partir d'un échantillon de la variable continue en question. Appelons X cette variable continue et $x_n(n = 1 \cdots N)$ ses échantillons.

Pour réaliser un histogramme de X à partir de ses N échantillons, il faut déjà définir un intervalle I sur lequel on veut calculer l'histogramme - dans l'exemple ci-dessus c'est l'intervalle $I = [-3{,}3]$. Il faut ensuite définir p le nombre de sous-intervalles que l'on va créer dans cet intervalle - dans l'exemple ci-dessus il y en a $p = 10$.

Calculer un histogramme, c'est compter le nombre de données $x_n(n = 1 \cdots N)$ qui tombent dans chacun des p sous-intervalles $I_1 \cdots I_p$. Ses comptes, notés $h_1 \cdots h_p$, sont les hauteurs qui sont affichées sur le graphique de l'histogramme et ses modes, notés $m \cdots m_p$, sont simplement les milieux de ses sous-intervalle$I_1 \cdots I_p$.

2.2 - Quelques petites aides algorithmiques.

Premièrement, nous allons définir le prototype de votre fonction. Il faudra pouvoir l'appeler ainsi :

```
>> Histo, Mode = Histogramme(Data, Intervalle, NombreDeCellules)
```

`Histo` contiendra les valeurs $h_1 \cdots h_p$, `Mode` contiendra les valeurs $m \cdots m_p$, `Data` est le vecteur des échantillons de la variable dont on veut tracer l'histogramme, `Intervalle` est l'intervalle de définition (donc I) et `NombreDeCellules` est le nombre de sous-intervalles (donc p).
Une fois un histogramme calculé, vous pouvez l'afficher en utilisant non pas la fonction `pyplot.plot()` mais la fonction `pyplot.bar()` - regardez dans l'aide comment elle s'utilise. Si vous faites une fonction d'affichage, ce sera plus simple d'ailleurs.
Quelque soit le calcul, il y a toujours plusieurs façon de le réaliser. Certaines méthodes sont plus longues que d'autre. On appelle cela la "complexité algorithmique".
Je vous donne ci-dessous, en pseudo-code, la méthode la plus simple à programmer, mais aussi ayant la pire complexité algorithmique.

Initialiser toutes les valeurs $h_1 \cdots h_p$.
pour tous les sous intervalles $I_k(k\ de\ 1 \text{ à } p)$
    pour tous les échantillons $x_n(n = 1 \cdots N)$ *de la variable*
        si $x_n \in I_k$ *alors on inerémente* $h_k\ de\ 1$
    fin_pour
fin_pour

On fait donc Np calculs.
Essayez déjà de programmer cela, puis réfléchissez à comment on peut rendre cet algorithme moins complexe.
Vous ferez aussi en sorte qu'on puisse appeler la fonction de façon dégradée, c'est à dire sans préciser certains arguments.

- Si on appelle la fonction sans le nombre de cellules, par défaut il est égal à 10.

```
>> Histo, Mode = Histogramme(Data, Intervalle)
```

- Si on appelle la fonction sans préciser l'intervalle, par défaut cet intervalle est défini par les plus petites et plus grandes valeurs des échantillons.

```
>> Histo, Mode = Histogramme(Data)
```

- Si on appelle la fonction avec un seul argument de retour, on ne renvoit pas les modes.

```
>> Histo = Histogramme(Data, Intervalle)
```

Faites en sorte que votre programmation soit propre, bien organisée et bien commentée.
Pour tester votre algorithme, vous pouvez utiliser les fichiers de données qui vous sont fournis en aide.

Si vous faites cette partie rapidement, passez au TP suivant.