Traitement des données

Projet informatique

Sébastien Bindel

Semestre 1

Université de Haute Alsace

Contexte

Programmation

- acquisition de connaissances basiques en algorithmie,
- premiers programmes en Python,
- maîtrise des structures conditionnelles, boucles et des structures de données simples.

Contexte

Programmation

- acquisition de connaissances basiques en algorithmie,
- premiers programmes en Python,
- maîtrise des structures conditionnelles, boucles et des structures de données simples.

Mathématique

- acquisition de connaissances basiques en théorie des probabilités.
- étude des variables continues et discrètes.

Contexte

Programmation

- acquisition de connaissances basiques en algorithmie,
- premiers programmes en Python,
- maîtrise des structures conditionnelles, boucles et des structures de données simples.

Mathématique

- acquisition de connaissances basiques en théorie des probabilités.
- étude des variables continues et discrètes.

Objectif

Savoir réaliser à travers un projet, une étude théorique simple sur une loi de probabilité et vérifier les résultats de manière expérimentale.

Projet

Description du projet

Thématique du projet

- Programmation d'un jeux de loto en utilisant une générateur aléatoire uniforme.
- étude théorique,
- tri d'une séquence tirée,
- recherche d'un nombre dans une séquence,
- sauvegarde et chargement des séquences précédentes,
- calcul et visualisation de la distribution.

Déroulement du projet

CM	consolidation des notions de la théorie des probabilités, analyse succinte des algorithmes.
TD	exercice sur les outils probabilites (1h30), algorithmie et programmation (1h30).
TP	présence d'enseignants pour une aide sur le projet.
Autonomie	scéances dédiées et travail personnel.

Déroulement du projet

CM consolidation des notions de la théorie des probabilités, analyse succinte des algorithmes.

TD exercice sur les outils probabilites (1h30), algorithmie et programmation (1h30).

TP présence d'enseignants pour une aide sur le projet.

Autonomie scéances dédiées et travail personnel.

Le projet est individuel se déroule sur trois semaines

Objectifs demandés

Probabilité

- calculer l'espérance d'une variable aléatoire continue,
- calculer l'espérance d'une variable aléatoire discrète,
- calculer les probabilités des combinaisons et des arrangements.

Programmation

- trier une séquence,
- recherche d'un nombre dans une séquence,
- sauvegarder et charger des données,
- calculer un histogramme et l'afficher.

Objectifs sur la partie programmation - Codage du loto

1– Écrire le programme du tirage du loto (5 nombres entre 1 et 45).

2– Utiliser un générateur aléatoire uniforme (voir la bibliothèque numpy).

3– Possibilité d'effectuer plusieurs tirages successifs en utilisant la même graine aléatoire.

Objectifs sur la partie théorique

1– Etudier l'espérance de la variable alétoire

2– Cacluler la probabilité d'avoir une combinaison spécifique.

3– Comparer l'espérance de la variable aléatoire avec celle obtenue par le programme.

Objectifs sur la partie programmation - tri des données

1– Présenter l'algorithme du tri cocktail (itératif), écrire l'algorithme et l'implémenter en Python. Cette fonction sera utilisable sur une séquence donnée par un tirage.

2– Présenter l'algorithme du tri par insertion (itératif), écrire l'algorithme et l'implémenter en Python. Cette fonction sera utilisable sur une séquence donnée par un tirage.

3– Présenter l'algorithme du tri fusion (récursif), écrire l'algorithme et l'implémenter en Python. Cette fonction sera utilisable sur une séquence donnée par un tirage.

Objectifs sur la partie programmation - recherche d'un nombre

1– Présenter l'algorithme de recherche dichotomique, écrire l'algorithme itératif et l'implémenter en Python. Cette fonction sera utilisable sur une séquence donnée par un tirage préalablement triée.

2– Écrire l'algorithme récursif de recherche dichotomique et l'implémenter en Python.

Objectifs sur la partie programmation - sauvegarde des données

1– Effectuer une recherche bibliographique sur la différence entre un format binaire et lisible humainement et présenter deux formats pour ce dernier.

2– Implémenter une méthode relative à la sauvegarde de données binaires et une autre relative au chargement de données binaires (utilisation d'une bibliothèque Python). Ces méthodes devront s'appliquer à la liste des numéros sortis.

3– implémenter deux méthodes de sauvegarde et de chargement de données lisibles humainement, chacune pour un format différent (utilisation d'une bibliothèque Python). Ces méthodes devront s'appliquer à la liste des numéros sortis.

Objectifs sur la partie programmation - Affichage de la distribution

1– Écrire l'algorithme du calcul de l'histogramme des numéros sortis et implémenter le programme en Python.

2– Afficher le résultat sous la forme d'un diagramme en bâtons en utilisant la bibliothèque matplotlib.

Théorie des probabilités

Expérience

Définition

Dérivé du mot latin *experientia* qui signifie épreuve, essai, tentative, une expérience est une opération conduite sous des conditions contrôlées en vue de découvrir un effet ou une loi inconnue, de tester ou d'établir une hypothèse, ou encore d'illustrer une loi connue.

Exemple

- Vérifier l'efficacité d'un traitement pharmaceutique.
- Est ce que le débit réseau peut être décrit par une loi normale ?
- Y a t'il une corrélation entre la puissance du signal Wi-Fi et la distance entre la station de base et le client ?

Expérience aléatoire

Propriétés

- on ne peut pas prédire la certitude du résultat,
- On peut décrire l'ensemble des résultats avant le déroulement de l'expérience.

Définition

L'ensemble fondamental d'une expérience aléatoire est l'ensemble de tous les résultats possibles de l'expérience. Il est dénoté par Ω.

Pour le jet d'un dé à six faces, $\Omega = \{1, 2, 3, 4, 5, 6\}$.

Pour un lancement d'une pièce composée d'une partie face (F) et d'une partie pile (P), $\Omega = \{P, F\}$.

D'une manière générale un ensemble fondamental peut être **fini**, **infini dénombrable** ou **infini non dénombrable**.

Ensembles fini, infini, dénombrable et non dénombrable

Ensemble fini

Tous les résultats possibles de l'expérience sont dénombrables sur un domaine fini (ensemble utilisé dans ce projet), e.g. lancé d'un dé, jeux du loto, jeux du bingo, ...

Infini dénombrable

Le nombre de résultats est infini mais dénombrable. On peut alors associer chaque résultat à un nombre différent par exemple $\mathbb{N}$. De manière plus formelle il existe une bijection entre l'ensemble des résultats et $\mathbb{N}$ (comme $\mathbb{N}$ avec $\mathbb{Z}$). On retrouve par exemple un nombre infini de lancers d'une pièce.

Infini non dénombrable

Le nombre de résultats est infini mais non associable à l'ensemble des entiers naturels. Par exemple, un relevé continu de la vitesse du vent représenté par un réel.

L'évènement d'une expérience aléatoire

Définition

Un évènement est le résultat d'une expérience aléatoire, c'est à dire une combinaison de résultats possibles. Plus formellement, un évènement est un sous-ensemble de Ω.

Exemple

Considérons deux lancés de dés, alors la somme des points égal à six :
$A = \{(1,5),(2,4),(3,3),(4,2),(5,1)\}$

Évènements particuliers

Évènement particulier

Il représente le sous-ensemble qui contient un seul évènement.

Évènement certain

Il représente le sous-ensemble qui contient tous les éléments, autrement dit l'ensemble fondamental lui même.

Évènement impossible

Il représente le sous-ensemble vide et est représenté par $\varnothing$.

Opérations sur les évènements

Supposons une expérience aléatoire d'un lancé d'un dé et l'évènement B qui est l'évènement d'obtenir un chiffre inférieur à six et l'évènement C qui est l'évènement d'avoir un chiffre impair.

Le complémentaire : soit $\overline{A}$ l'évènement qui se réalise lorsque l'évènement A ne se réalise pas, alors la probabilité associée est donnée par :

$$p(\overline{A}) = 1 - p(A)$$

L'opération de conjonction : l'évènement combiné D = "obtenir un chiffre impair et inférieur à six" est représenté par l'intersection entre B et C tel que :

$$D = B \cap C$$

L'opération de disjonction : L'évènement combiné E = "obtenir un chiffre impair ou inférieur à six" est représenté par l'union entre B et C tel que :

$$E = B \cup C$$

Permutations

Définition

On appelle permutation, un rangement ou classement ordonné de n objets.

Exemple

Si l'on dispose de trois objets a, b et c alors les permutations possibles sont : abc, acb, bac, bca, cab, cba.

le nombre de permutations de n objets est n !

Arrangement

Un arrangement est un classement où l'on tient compte de l'ordre. Le nombre d'arrangements de k objets parmi n sans remise est de :

$$A_n^k = \frac{n!}{(n-k)!} = n(n-1)(n-2)\dots(n-k+1),$$

Lorsque l'on considère la possibilité de remise alors le nombre d'arrangements est de :

$$r_n^k = n^k$$

Combinaison

Une combinaison est un classement où l'on ne tient pas compte de l'ordre. Le nombre de combinaisons de k objets parmi n sans remise est de :

$$C_n^k = \frac{n!}{k!(n-k)!},$$

Lorsque l'on considère la possibilité de remise alors le nombre d'arrangements est de :

$$K_n^k = \frac{(n+k-1)!}{k!(n-1!)}$$

Algorithme et programmation

Algorithme vs implémentation

Algorithme

- résolution d'un problème,
- indépendant d'un langage,
- simplification d'opérations.

Implémentation

- dépend du langage utilisé,
- quelques modifications subtiles peuvent être présentes,
- nombre d'opérations élémentaires est à une constante près.

Fonction itérative vs fonction récursive

Itérative fonction qui utilise une boucle,
arrêt lorsque la condition devient fausse,
approche simple.

Récursive fonction qui s'appelle elle-même,
condition d'arrêt nécessaire,
permet souvent de réduire le nombre d'opérations.

Fonction itérative vs fonction récursive

Itérative fonction qui utilise une boucle,
arrêt lorsque la condition devient fausse,
approche simple.

Récursive fonction qui s'appelle elle-même,
condition d'arrêt nécessaire,
permet souvent de réduire le nombre d'opérations.

Il existe toujours une version récursive et itérative d'un algorithme.

Exemple de la factorielle version itérative

Algorithme 1 : Calcul de la factorielle de manière itérative

Données : n, la valeur de la factorielle à calculer

Fonction *Factorielle*(n)

    $resultat \leftarrow 1$

    pour $i \leftarrow 1$ **à** n **faire**

        $resultat \leftarrow resultat \times i$

    fin

    retourner $resultat$

fin

Exemple de la factorielle version récursive

Algorithme 2 : Calcul de la factorielle de manière récursive

Données : n, la valeur de la factorielle à calculer

Fonction *Factorielle*(n)

    si $n = 1$ **alors**
        retourner 1
    fin
    sinon
        retourner $n \times Factorielle(n-1)$
    fin

fin

Algorithmes de tri

- tous les algorithmes n'ont pas la même efficacité,
- celle-ci dépend de la taille des données,
- itératif ou récursif.

- tous les algorithmes n'ont pas la même efficacité,
- celle-ci dépend de la taille des données,
- itératif ou récursif.

Quelques algorithmes

- tri à bulles (étudié dans ce cours),
- tri cocktail (amélioration du précédent),
- tri par insertion (efficace pour un faible nombre d'éléments),
- tri fusion (efficace pour un grand nombre d'éléments).

Algorithme de tri à bulles

Présentation

- permutations répétés d'éléments contigus qui ne sont pas dans le bon ordre,
- algorithme simple mais inefficace (n^2 opérations, n est la taille de la liste).

Algorithme 3 : Algorithme du trie à bulles

Données : tab, une liste d'éléments
Données : n, taille de la liste
pour $i \leftarrow 1$ **à** n **faire**
    pour $j \leftarrow 1$ **à** n **faire**
        si $tab[j] > tab[j+1]$ **alors**
            $temp \leftarrow tab[j]$
            $tab[j] \leftarrow tab[j+1]$
            $tab[j+1] \leftarrow temp$
        fin
    fin
fin

La recherche séquentielle

Principe

L'algorithme parcourt les éléments successifs d'une liste jusqu'à trouver l'élément recherché ou bien atteint la fin de la liste.

Principe

L'algorithme parcourt les éléments successifs d'une liste jusqu'à trouver l'élément recherché ou bien atteint la fin de la liste.

Algorithme 5 : Algorithme itératif de recherche séquentielle

Données : tab, une liste d'éléments
Données : n, taille de la liste
Données : e, l'élément recherché
$trouver \leftarrow faux$
$i \leftarrow 1$
tant que $i \leq n \vee trouver = faux$ **faire**
    si $tab[i] = e$ **alors**
        $trouver \leftarrow vrai$
    fin
    $i \leftarrow i + 1$
fin
retourner $trouver$

Version récursive de la recherche séquentielle

Algorithme 6 : Algorithme récursif de recherche séquentielle

Données : tab, une liste d'éléments
Données : n, taille de la liste
Données : e, l'élément recherché
Fonction *Recherche*(tab, n, $valeur$, idx)
    si $idx = n$ **alors**
        retourner $faux$
    fin
    sinon
        si $tab[idx] = valeur$ **alors**
            retourner $vrai$
        fin
        sinon
            retourner $Recherche(tab, n, valeur, i + 1)$
        fin
    fin
fin