R2.15 – Initiation à l'analyse de données

BUT Informatique

UPEC UNIVERSITÉ PARIS-EST CRÉTEIL | iut CRÉTEIL-VITRY

ACTIVITE #2 – LOI DE BENFORD

Cette activité propose une initiation à l'analyse de données centrée sur la loi de Benford. Cette loi stipule que dans une liste de nombres, les chiffres initiaux suivent une distribution logarithmique, avec le chiffre 1 apparaissant plus fréquemment que les autres.

Les applications pratiques de la loi de Benford sont nombreuses et diverses. Elles touchent, par exemple, le domaine de la détection des fraudes (comptables, électorales, scientifiques, ...), la génomique (prédiction de la longueur maximale du génome), la cybersécurité (identification d'activités anormales), ou encore dans le cadre de l'étude de données environnementales.

L'activité est divisée en trois étapes :

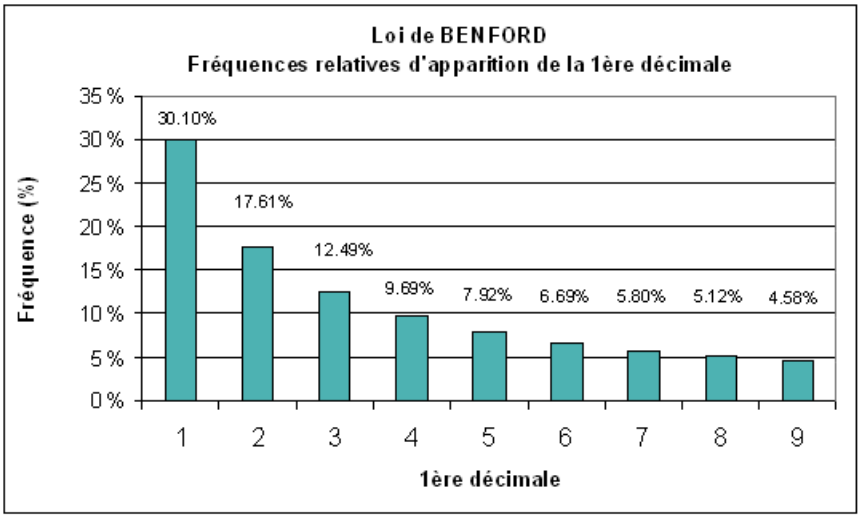
1. Code source utile : Comprendre et utiliser des fonctions Python pour calculer les probabilités de la loi de Benford, extraire le premier chiffre significatif d'un nombre, compter les occurrences de ces chiffres et calculer leurs fréquences.
2. Vérifications sur des données simulées : Adapter un code source pour vérifier la loi de Benford sur des suites de nombres comme les puissances de 2, la suite de Fibonacci, et des nombres aléatoires.
3. Application sur des données réelles : Télécharger des données de population de l'INSEE, les analyser pour vérifier si elles suivent la loi de Benford, et séparer les données par sexe et par commune pour des analyses plus détaillées.

En supplément, cette activité inclut également des instructions pour télécharger et analyser d'autres jeux de données disponibles en ligne pour vérifier leur conformité à la loi de Benford.

Loi de Benford

La loi de Benford prédit que statistiquement dans une liste de nombres donnés, la probabilité qu'un de ces nombres commence par le chiffre 1 est plus importante que celle qu'un nombre commence par le chiffre 9. Plus précisément la loi de Benford prédit que la probabilité p_d qu'un nombre commence par le chiffre $d \in \{1,2,3,\cdots,9\}$ est :

$$p_d = \log_{10}\left(1 + \frac{1}{d}\right) = \log_{10}(d+1) - \log_{10}(d)$$



Mise en œuvre

Etape #1 – Code source utile

Les fonctions suivantes sont fournies. Vous devez les comprendre avant de les utiliser :

- Une fonction `benford()` qui retourne les valeurs de p_d données par la loi de Benford pour $d \in \{1,2,3,\cdots,9\}$.
- Une fonction `firstdigit(n)` qui pour un nombre `n` donné retourne son premier chiffre significatif.
- Une fonction `occurrences(liste)` qui retourne le nombre d'occurrences des premiers chiffres significatifs de `liste`.
- Une fonction `frequences(count)` qui calcule la fréquence de ces chiffres à partir des occurrences `count`.

Etape #2 – Vérifications sur des données simulées

Le code source de la première vérification est fourni (`benford-pow2.py`). A vous de l'adapter pour les deux jeux de données simulées suivants.

- Vérifier si la loi de Benford semble satisfaite pour la suite des nombres $(2^n)_{n\in\mathbb{N}}$ en comparant l'histogramme empirique avec la loi de Benford.
- Vérifier si la loi de Benford semble satisfaite pour la suite de Fibonacci $F_0 = 0,\ F_1 = 1,\ F_n = F_{n-1} + F_{n-2}$ pour $n \geq 2$.
- Vérifier si la loi de Benford semble satisfaite pour les chiffres significatifs de n entiers aléatoires entre 1 et $10000 \times n$.

Que pouvez-vous conclure d'après les résultats obtenus ?

Etape #3 – Application sur des données réelles

- En allant sur le site de l'INSEE à l'adresse : https://www.insee.fr/fr/statistiques/5395878 télécharger le fichier au format CSV contenant les données de la population par sexe et âge regroupé (POP1A) en 2018 (fichier également disponible sur Eprel).
- Importer ces données pour avoir la population par code postal, sexe et tranche d'âge.

  Indication : Utiliser la fonction `read_csv` du module Pandas afin de lire le contenu du fichier CSV. La première ligne indique le nom des variables et chaque ligne suivante contient la valeur des indicateurs retenus pour chaque commune et chaque arrondissement municipal, séparées par le caractère ';'.

```
NIVGEO;CODGEO;LIBGEO;SEXE;AGEPYR10;NB
ARM;13201;Marseille 1er Arrondissement;1;0;825.53120305
ARM;13201;Marseille 1er Arrondissement;1;3;662.46699673
ARM;13201;Marseille 1er Arrondissement;1;6;940.25987932
. . .
COM;97424;Cilaos;2;55;299.68554466
COM;97424;Cilaos;2;65;316.0341761
COM;97424;Cilaos;2;80;114.79675508
```

  où :

  - `NIVGEO` indique le niveau géographique (`ARM`=arrondissement, `COM`=commune) ;
  - `CODGEO` et `LIBGEO` donnent respectivement le code et le nom de l'arrondissement ou de la commune ;
  - `SEXE` indique le genre 1=Homme, 2=Femme ;
  - `AGEPYR10` indique la classe d'âge (ici 10 classes) :
    - 0 : moins de 3 ans
    - 3 : 3 à 5 ans
    - 6 : 6 à 10 ans
    - 11 : 11 à 17 ans
    - 18 : 18 à 24 ans
    - 25 : 25 à 39 ans
    - 40 : 40 à 54 ans
    - 55 : 55 à 64 ans
    - 65 : 65 à 79 ans
    - 80 : 80 ans ou plus
  - `NB` est le nombre[1] d'individus par sexe et par tranche d'âge estimé dans la zone géographique (commune ou arrondissement).

> La loi de Benford doit s'appliquer aux données de population mesurées par commune, sexe et classe d'âge principalement parce que ces données couvrent un large éventail de valeurs, avec des populations de communes allant de petites à très grandes. La croissance démographique et la répartition géographique des populations créent une structure de données où les premiers chiffres suivent une répartition logarithmique naturelle, favorisant l'apparition plus fréquente de petits chiffres, conformément à la loi de Benford.

[1] On pourra remarquer que le nombre d'individus est donné sous forme d'une valeur réelle et non entière. En effet, les chiffres non entiers peuvent résulter d'approximations, de modèles statistiques, et de méthodes de calcul qui visent à fournir des estimations plus précises.

- Déterminer si la liste de toutes les populations par commune, sexe et âge suit la loi de Benford.
- Séparer les données précédentes pour obtenir les listes des populations par sexe et déterminer si elles suivent la loi de Benford.
- Sommer les données précédentes pour obtenir la liste des populations par commune (tout âge et tout sexe confondus) et déterminer si elle suit la loi de Benford.

Que pouvez-vous conclure d'après les résultats obtenus ?

Supplément

En allant sur le site de l'INSEE ou un autre site, télécharger votre jeu de données préféré, et tester s'il suit la loi de Benford.

Par exemple, vous pouvez utiliser les comptes détaillés de l'État disponibles ici. D'autres jeux de données sont disponibles sur le portail de l'INSEE ici.

Code source pour l'étape #1 : fonctions utiles

```python
import math

def benford():
    # Calculer les probabilités de la loi de Benford pour d = 1, ..., 9
    p = {d: math.log10(1 + 1/d) for d in range(1, 10)}
    return p

def firstdigit(n):
    # Prendre la valeur absolue de n et convertir en chaîne
    n_str = str(abs(n))

    # Enlever la partie avant et après le point décimal si présent
    if '.' in n_str:
        n_str = n_str.replace('.', '')

    # Chercher le premier chiffre non nul
    for char in n_str:
        if char != '0':
            return int(char)

    # Si n vaut 0, retourner 0
    return 0

def occurrences(liste):
    # Créer un dictionnaire pour compter les occurrences des premiers chiffres
    count = {i: 0 for i in range(1, 10)}  # Dictionnaire pour les chiffres de 1 à 9

    # Parcourir chaque élément de la liste
    for n in liste:
        # Trouver le premier chiffre non nul du nombre
        premier_chiffre = firstdigit(n)

        # On ignore les occurrences de 0
        if premier_chiffre != 0:
            count[premier_chiffre] += 1

    return count

def frequences(count):
    # Calculer le total des occurrences
    total = sum(count.values())

    # Calculer la fréquence de chaque élément
    f = {k: v / total for k, v in count.items()}

    return f
```

Code source pour l'étape #2 : vérification pour la suite $(2^n)_{n\in\mathbb{N}}$

```
import math
import matplotlib.pyplot as plt

# Fonction pour générer les puissances de 2 et extraire les premiers chiffres non nuls
def generate_powers_of_2(n_max):
    return [2 ** n for n in range(n_max)]

# Nombre de puissances de 2 à analyser
n_max = 1000  # Par exemple, les 1000 premières puissances de 2

# Générer les puissances de 2
powers_of_2 = generate_powers_of_2(n_max)

# Calcul des occurrences et des fréquences empiriques
occurrences_empiriques = occurrences(powers_of_2)
frequences_empiriques = frequences(occurrences_empiriques)

# Calcul des fréquences théoriques selon la loi de Benford
frequences_benford = benford()

# Affichage des résultats
# Histogramme des fréquences empiriques
plt.bar(frequences_empiriques.keys(), frequences_empiriques.values(), alpha=0.6,
label='Fréquences empiriques')

# Histogramme des fréquences théoriques (Loi de Benford)
plt.plot(frequences_benford.keys(), frequences_benford.values(), 'ro-', label='Loi de
Benford')

# Titrage et légendes
plt.title('Comparaison de la loi de Benford avec les puissances de 2')
plt.xlabel('Premier chiffre non nul')
plt.ylabel('Fréquence')
plt.legend()

# Affichage
plt.show()
```