Un intervalle de fluctuation d'une variable binomiale, en première S

B. Intervalle de fluctuation d'une variable aléatoire de distribution binomiale

Il existe plusieurs méthodes pour calculer un intervalle de confiance d'une variable aléatoire de distribution binomiale. Elles peuvent donner des résultats sensiblement différents. La méthode proposée dans le document ressource de première S peut être mise en œuvre par des algorithmes. Nous analyserons et mettrons en œuvre successivement cinq algorithmes et étudierons les différences qui peuvent les caractériser.

B.1. Détermination de l'intervalle de fluctuation de seuil = (1 – α), symétrique en probabilité, d'après la définition du document ressource de première S. (fichier professeur : BinoDistribEtIF0_0.py)

Les cinq algorithmes présentés correspondent à cinq stratégies différentes dont on étudiera les instructions de mise en œuvre avec Python. Un premier prolongement peut constituer à s'interroger sur l'efficacité comparée de ces cinq algorithmes, le critère le plus simple à considérer étant le temps d'exécution estimé, sur une machine donnée. Un deuxième prolongement peut-être la mise en œuvre d'une mesure de ce temps d'exécution.

B.1.a. Algorithme 1 (Lignes 62 à 94 (ALGO 1) du fichier BinoDistribEtIF0_0.py)

Cet algorithme est la mise en œuvre directe et explicite de la définition du document ressource. On recherche, dans la liste contenant les valeurs des probabilités cumulées croissantes de la variable aléatoire X (appelée aussi répartition de X) de distribution binomiale de paramètre n et p, la plus petite valeur strictement supérieure à (1 – seuil) / 2, puis la plus petite valeur supérieure ou égale à 1 – (1 – seuil) / 2 = (1 + seuil) / 2. Les deux valeurs de la variable X correspondant à ces probabilités cumulées sont les bornes **a** et **b** de l'intervalle de fluctuation.

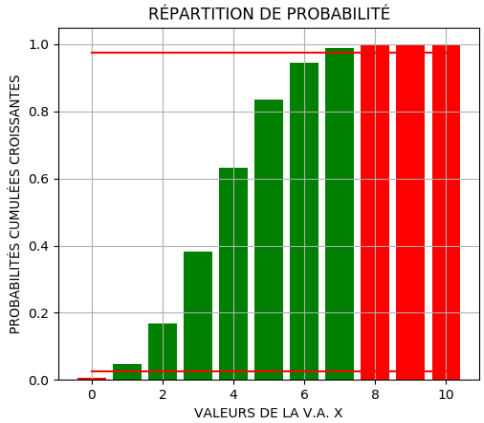
B.1.b. Illustration graphique de cet intervalle de fluctuation. (Lignes 97 à 113 du fichier BinoDistribEtIF0_0.py)

L'objectif est de représenter la répartition de la v.a. binomiale par un diagramme en barres avec des barres de couleur verte pour les valeurs de la v.a. appartenant à l'IF et de couleur rouge pour les autres. On utilisera le même principe que celui vu au **A** : création d'une liste de couleurs avec les bonnes séquences ordonnées.

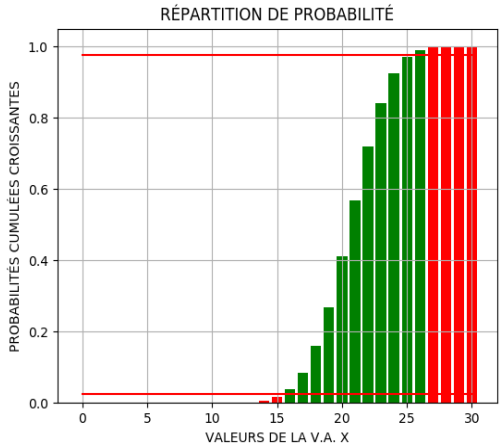
Les valeurs des probabilités (1 – seuil) / 2 et (1 + seuil) / 2 sont illustrées par deux droites rouges parallèles à l'axe des abscisses.

► Quelques exemples des résultats obtenus :

```
>>> IFBino1(n = 10, p = .4, seuil = .95)
l'IF au seuil de 0.95 de la va X est :
[ 1 ; 7 ],
 sa probabilité est 0.9816588288
```




```
>>> IFBino1(n = 30, p = .7, seuil = .95)
l'IF au seuil de 0.95 de la va X est :
[ 16 ; 26 ],
 sa probabilité est 0.973746121196
```



B.1.c. Une fonction Python mettant en œuvre l'algorithme ALGO1 (Lignes 117 à 139 du fichier BinoDistribEtIF0_0.py)

Il s'agit simplement de réunir toutes les lignes de commandes vues précédemment, dans une fonction Python que l'on appellera IFBino1(…) et qui comportera 3 paramètres : n le nombre de tirages avec remise du , p la probabilité de "succès", et seuil le seuil de probabilité pour la détermination de l'intervalle de fluctuation.

☺ ☺ ☺ ☺ ☺ ☺ ☺ ☺ ☺ ☺ ☺ ☺ ☺ ☺ ☺ **PROLONGEMENTS** ☺ ☺ ☺ ☺ ☺ ☺ ☺ ☺ ☺ ☺ ☺ ☺ ☺ ☺ ☺

B.2. Autres algorithmes de détermination de l'IF du document ressource de première S

B.2.a. Algorithme 2 (Lignes 144 à 176 (ALGO 2) du fichier BinoDistribEtIF0_0.py)

La recherche de **a** se fait en affectant à chaque élément de la liste de la répartition de X un booléen "True" ou "False" selon que sa valeur est inférieure ou égale à (1 – seuil) / 2 ou non. Dans une opération numérique "True" prend la valeur 1 et "False" la valeur 0.

a est la somme de la liste des booléens ainsi obtenus.

La recherche de **b** se fait en remplaçant "inférieure ou égale" par "strictement inférieure" et (1 – seuil) / 2 par (1 + seuil) / 2.

B.2.b. Algorithme 3 (Lignes 179 à 212 (ALGO 3) du fichier BinoDistribEtIF0_0.py)

C'est l'algorithme que l'on rencontre parfois dans les manuels. On parcourt la liste de la répartition de X tant que l'on a une valeur inférieure ou égale à (1 – seuil) / 2.

a est l'indice obtenu à la fin de cette boucle tant que.

b est l'indice obtenu à la fin de la boucle tant que la valeur de l'élément de la liste de la répartition de X est strictement inférieure à (1 + seuil) / 2.

B.2.c. Algorithme 4 (Lignes 215 à 233 (ALGO 4) du fichier BinoDistribEtIF0_0.py)

Comme **B.1.c.** mais pour la recherche de **b**, on évite de parcourir la liste depuis le début, en continuant le parcourt après la fin de la première boucle tant que.

B.2.d. Algorithme 5. (Lignes 236 à 267 (ALGO 5) du fichier BinoDistribEtIF0_0.py)

On parcourt la liste de la répartition de X en comparant l'élément d'indice i – 1 et celui d'indice i, au seuil (1 – seuil) / 2 pour détecter **a**, puis au seuil (1 + seuil) / 2 pour détecter **b**. Il faut traiter le cas de la première valeur de la répartition à part.

B.3. Estimation d'un intervalle de fluctuation d'une variable aléatoire par simulation (Traduction en Python, à venir)

Lorsque l'on n'a pas pu modéliser la distribution de la variable aléatoire par une loi binomiale ou une autre loi, la simulation est une bonne technique pour calculer un intervalle de fluctuation.

Nom Prénom :

TP_ALGO_B.1.a. (Fiche élève 1/3, fichier BinoDistribEtIF_Elev.py, lignes 12 à 32)

1° Compléter le tableau suivant en marquant les résultats obtenus en appliquant "à la main" les commandes indiquées. **2°** Vérifier en les exécutant.

```
"La définition de l'IF de X au seuil de probabilité de (1 - alpha),"
"    donnée par le document ressource de première S est la suivante:"
"IF de X = [a ; b] avec
    "a est la plus petite valeur de X telle que p(X <= a) > (1 - seuil)/2"
    "b est la plus petite valeur de X telle que p(X <= b) >= (1 + seuil)/2)"
"------------------------------ ALGO 1 -------------------------------------"
"Détermination de l'intervalle de fluctuation selon la définition du document ressource de 1ere S"
"C'est un IF bilatéral, de seuil = (1 - alpha), 'symétrique' en probabilité"
"ALGO 1 : mise en œuvre directe et explicite de la définition"
from scipy.stats import binom
n, p, seuil = 5, 0.6, 0.80
"x est la liste des valeurs (de 0 à n) de la variable aléatoire X"
x = range(n + 1)
Résultats :    [0              1              ...             ...             ...             5]

"reparti est la liste des probabilités cumulées croissantes"
"    appelée aussi répartition de probabilité de la variable X"
reparti = binom.cdf(x, n, p)

Résultats :   [0,01          0,087           ...             ...             ...             1]

"Les index des éléments de reparti qui sont > (1 - seuil)/2. Avec seuil = 0,8, (1 - seuil)/2 = 0,1"
"Ces index sont des valeurs de la v.a. X"
idap = [i for i in x if reparti[i] > (1 - seuil) / 2]

Résultats :   [...            ...             ...             ...]

"a est la plus petite valeur de X telle que p(X <= a) > (1 - seuil)/2"
a = min(idap)
Le minimum de la liste idap est :
```

Nom Prénom :

TP_ALGO_B.1.a. et B.1.b. (Fiche élève 2/3, fichier BinoDistribEtIF_Elev.py, lignes 34 à 63)

1° Compléter le tableau suivant en marquant les résultats obtenus en appliquant "à la main" les commandes indiquées. **2°** Vérifier en les exécutant.

```
"Les index des éléments de reparti qui sont >= (1 + seuil)/2. Ces index sont des valeurs de la v.a. X"
idbp = [i for i in x if reparti[i] >= (1 + seuil) / 2)]

Résultats :     [...              ...]

"b est la plus petite valeur de X telle que p(X <= b) >= (1 + seuil)/2"
b = min(idbp)
Le minimum de la liste idbp est :

"Je calcule et j'affiche p(a <= X <= b)"
pIF = binom.cdf(b, n, p) - binom.cdf(a - 1, n, p)
print("l'IF au seuil de", seuil, "de la va X est : [", a, ";", b, "],\n sa probabilité est ", pIF)

L'intervalle de fluctuation est :

"Construction du graphique de la répartition de X avec l'IF en couleur :"
"Il y a a barres de couleur rouge puis (b - a + 1) barres de couleur verte puis"
"    à nouveau (n - b) barres de couleur rouge. Avec n = 5 ; p = 0,6 ; seuil = 0,80, on trouve a = 2 et b = 4"
listCoul = ['red'] * a + ['green'] * (b - a + 1) + ['red'] * (n - b)

Résultats :    ['red'            ...              ...              ...              ...             'red']

"Je superpose tous les éléments du graphique, barres de couleur, droites rouges, titres divers, grille.
from matplotlib import pyplot as plt
plt.figure()
plt.bar(x, reparti, color = listCoul)
plt.plot((0, n), ((1 - seuil) / 2, (1 - seuil) / 2), color = 'red')
plt.plot((0, n), ((1 + seuil) / 2, (1 + seuil) / 2), color = 'red')
plt.title("RÉPARTITION DE PROBABILITÉ")
plt.xlabel("VALEURS DE LA V.A. X")
plt.ylabel("PROBABILITÉS CUMULÉES CROISSANTES")
plt.grid()
plt.show()
```

Nom Prénom :

TP_ALGO_B.1.c. (Fiche élève 3/3, fichier BinoDistribEtIF_Elev.py, lignes 65 à 88)

1° Le tableau qui suit présente le début et la fin de la fonction Python que vous devez obtenir. Compléter par les lignes qui manquent en recopiant les lignes de commande) (et uniquement les lignes de commande) utilisées dans les deux tableaux précédents.

```
"Une fonction correspondant à ALGO 1. Nécessite pyplot de matplotlib et binom de scipy.stats"
def IFBino1(n = 10, p = .5, seuil = .95):
    from scipy.stats import binom
    from matplotlib import pyplot as plt
    ..............................................................................
    ..............................................................................
    ..............................................................................
    ..............................................................................
    ..............................................................................
    ..............................................................................
        ..........................................................................
    ..............................................................................
    ..............................................................................
    ..............................................................................
    ..............................................................................
    ..............................................................................
    ..............................................................................
    ..............................................................................
    ..............................................................................
    ..............................................................................
    plt.grid() ; plt.show()
```

2° Dans Spyder ou Pisa, ouvrir le fichier **BinoDistribEtIF_Elev.py**. **Rassembler** (**copier-coller** dans la fonction IFBino1(…), **lignes 71 à 86**) toutes les lignes de commande et uniquement les lignes de commande (sans les commentaires) pour compléter la fonction Python ayant pour arguments les paramètres n et p de la loi binomiale, la valeur du seuil de probabilité et qui affichera l'intervalle de fluctuation et le graphique en barres de la répartition de la loi binomiale dont les barres correspondant à l'intervalle de fluctuation seront de couleur différente. Les deux valeurs seuil de la probabilité seront représentées par des droites rouges parallèles à l'axe des abscisses. Vous pouvez voir, page 1 un exemple des affichages demandés. Le fichier devra être enregistré avec le nom ****IFBino.py, en remplaçant **** par votre nom d'utilisateur informatique.

3° Tester cette fonction en utilisant les exemples des graphiques de la **page 1**.

NOM Prénom : **PROLONGEMENTS**

TP_ALGO_B.2.a. (Fiche élève)

1° Parmi les lignes **97 à 124** du fichier **BinoDistribEtIF_Elev.py** exécuter une à une les lignes de commandes de **ALGO 2** et noter leurs résultats.

```
"----------------------------- ALGO 2 -----------------------------------"
from scipy.stats import binom
n, p, seuil = 5, .6, 0.80
"x est la liste des valeurs (de 0 à n) de la variable aléatoire X"
x = range(n + 1)
Résultats :    [0              1              ...              ...              ...              5]
"reparti est la liste des probabilités cumulées croissantes"
reparti = binom.cdf(x, n, p)
Résultats :   [0,01           0,087           ...              ...              ...              1]
"Je repère par 'True' toutes les valeurs de reparti qui sont <= (1 - seuil) / 2"
boolam = reparti <= (1 - seuil) / 2
Résultats :   [True            ...              ...              ...              ...            False]
               [1              ...              ...              ...              ...              0]
"La somme des 'True' transformées en 1 donne la valeur de a"
a = sum(boolam)
Résultat :
"Je repère par 'True' toutes les valeurs de reparti qui sont < (1 + seuil) / 2"
boolbm = reparti < (1 + seuil) / 2
Résultats :   [True            ...              ...              ...              ...            False]
"La somme des 'True' transformées en 1 donne la valeur de b"
b = sum(boolbm)
Résultat :
"Je calcule p(a <= X <= b)"
pIF = binom.cdf(b, n, p) - binom.cdf(a - 1, n, p)
Résultat :
print("l'IF au seuil de", seuil, "de la va X est :", "[", a, ";", b, "],\n sa probabilité est", pIF)
Résultat :
```

2° Dans le fichier **BinoDistribEtIF_Elev.py,** compléter les **lignes 257 à 264** de la fonction `IFBino2(...)` avec des lignes de commande du tableau ci-dessus.
L'exécuter et l'utiliser pour tester son bon fonctionnement : ne "plante" pas ET donne les bons résultats.

NOM Prénom : **PROLONGEMENTS**

TP_ALGO_B.2.b. (Fiche élève)

1° Au DESSOUS de chacune des lignes ou groupes de lignes du tableau, rédiger une explication détaillant ce qu'elles font et les résultats attendus.

2° Parmi les lignes **127 à 159** du fichier **BinoDistribEtIF_Elev.py** exécuter une à une les lignes de commandes de **ALGO 3** et noter leurs résultats.

```
"------------------------------ ALGO 3 -----------------------------------"
from scipy.stats import binom
n, p, seuil = 5, .6, 0.80


x = range(n + 1)


reparti = binom.cdf(x, n, p)


i = 0
while reparti[i] <= (1 - seuil) / 2:
    i += 1
a = i



j = 0
while reparti[j] < (1 + seuil) / 2:
    j += 1
b = j



pIF = binom.cdf(b, n, p) - binom.cdf(a - 1, n, p)
print("l'IF au seuil de", seuil, "de la va X est :", "[", a, ";", b, "],\n sa probabilité est", pIF)
```

3° Dans le fichier **BinoDistribEtIF_Elev.py,** compléter les **lignes 281 à 294** de la fonction `IFBino3(…)` avec des lignes de commande du tableau ci-dessus. L'exécuter et l'utiliser pour contrôler son bon fonctionnement.

NOM Prénom : **PROLONGEMENTS**

TP_ALGO_B.2.c., B.2.d. (Fiche élève 1/2)

A – Quelle amélioration apporte la fonction `IFBino4(…)` par rapport à `IFBino3(…)` ?

B – Décrire et schématiser les 5 algorithmes de calcul de l'IF, utilisés dans les fonctions `IFBino1(…)`, `IFBino2(…)`, `IFBino3(…)`, `IFBino4(…)`, `IFBino5(…)`.

C – On désire comparer les temps d'exécution des différentes fonctions IFBinoX(…). Donnez votre avis sur les questions suivantes, en les justifiant :

1° Dans quelles conditions peut-on comparer les temps d'exécution ?

2° Classer les fonctions de la plus lente à la plus rapide.

3° Rechercher comment on peut comparer le temps d'exécution d'une fonction en Python ?

4° Proposer une mise en œuvre codée.