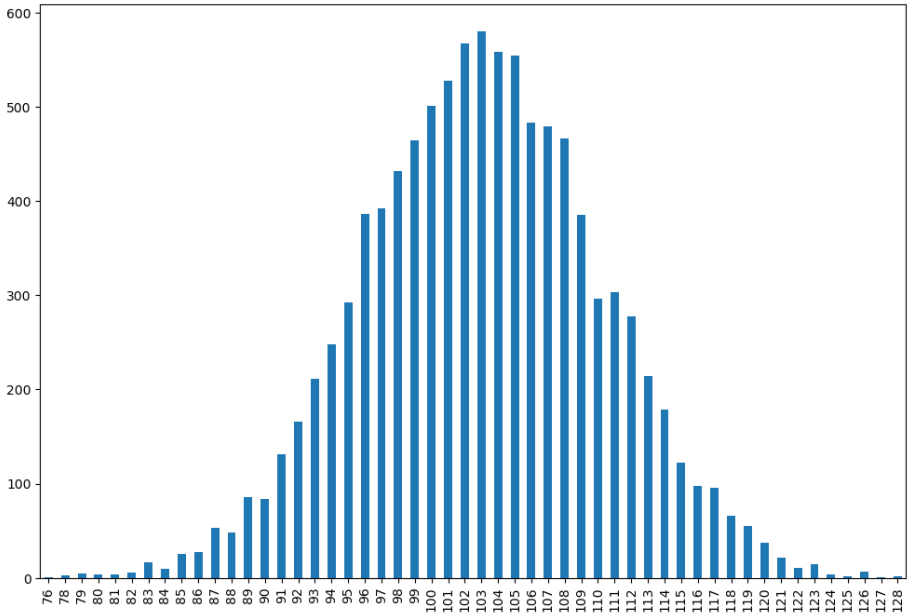
```
import numpy as np
import numpy.random as npr
import scipy.stats as scs
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Cas binaire : représentativité d'un échantillon de sondage

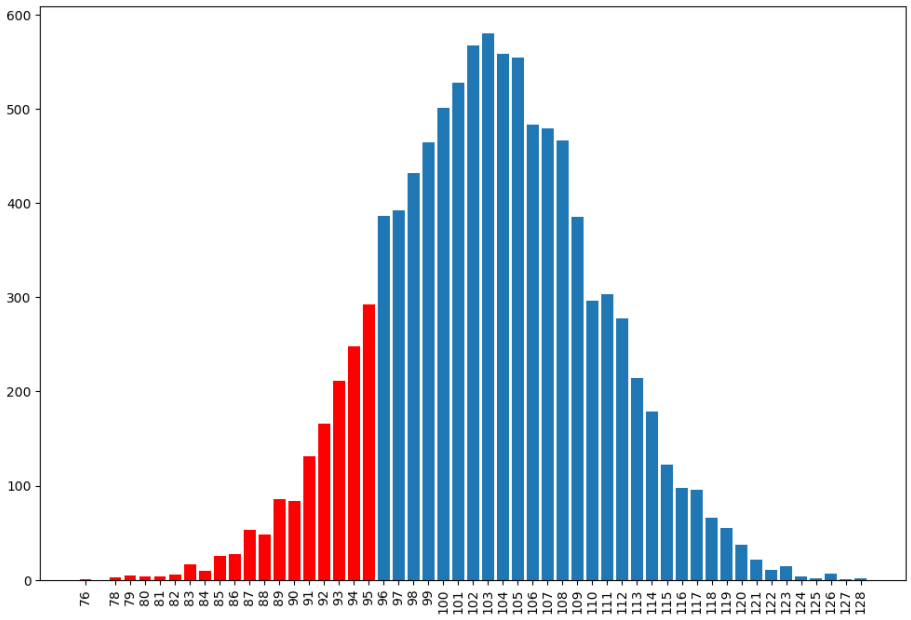
Question : Un échantillon de 200 personnes est constitué pour une étude sur [ce que vous voulez, p.ex., des aménagements locaux dans une ville moyenne en France]. Il contient 95 femmes et 105 hommes. Une association féministe se plaint que la voix des femmes est encore une fois minorée, car elles représentent 51.5% de la population et que l'échantillon aurait donc dû compter 103 femmes. Qu'en pensez-vous ?

```
p = 0.515
n = 200
N = 10000
# Version automatique
NbFemmes = npr.binomial(n, p, N)

# Version piétonne
# NbFemmes = []
# for j in range(N):
#     NbFemmes.append(sum(npr.random(n) <= p))

SNbFemmes = pd.Series(NbFemmes)

tab = SNbFemmes.value_counts().sort_index()

plt.figure(figsize=(12,8))
tab.plot(kind='bar')
plt.show()
```



Est-ce que 95 est une valeur atypique ?

```
plt.figure(figsize=(12,8))
mask = (tab.index>95)
plt.bar(tab.index[mask],tab[mask])
plt.bar(tab.index[~mask],tab[~mask],color='r')
plt.xticks(tab.index, rotation=90)
plt.show()
```



Ca n'en a pas l'air : valeur estimée de la probabilité d'avoir 95 femmes ou moins =

```
(SNbFemmes <= 95).mean()
```

```
0.142
```

```
# Autres estimations
for k in range(10):
    print(np.mean(npr.binomial(n, p, N) <= 95))
```

```
0.145
0.1434
0.1545
0.1369
0.1432
0.1444
0.1426
0.1421
0.1479
0.1457
```

Valeur exacte :

```
scs.binom.cdf(95, n, p)
```

```
0.14432156187089648
```

Méthodologie des tests d'hypothèses

Hypothèse de départ H_0, qu'on met sur le grill --- ici, H_0 : échantillon représentatif, soit $H_0 : p_0 = 51.5\%$

Hypothèse alternative H_1 --- devrait être H_1 : échantillon non-représentatif (sous-représentation ou sur-représentation des femmes), même si pour la simplicité du propos je vais prendre plutôt H_1 : sous-représentation des femmes, soit $H_1 : p_0 < 51.5\%$

Forte dissymétrie entre les hypothèses : on est attaché à H_0 et on ne la rejettre que si les données la contredisent **gravement**

A défaut, on la conserve (ce qui ne veut pas dire qu'on l'accepte ou qu'on la valide)

Sur l'exemple considéré, la P-valeur est de 14.4%, qui est supérieure au seuil conventionnel de 5%, il n'y a pas de contradiction grave : on conserve H_0

I.e.: On ne peut pas exclure que cet échantillon soit représentatif, aucun biais significatif n'a été démontré par l'analyse statistique

---

En revanche, avec seulement 90 femmes :

```
scs.binom.cdf(90, n, p)
```

```
0.038478374770585924
```

On aurait rejeté H_0 et conclu à un biais significatif entachant la représentativité de l'échantillon

Avec plusieurs catégories

La situation est la suivante.

- On fait face à un problème avec 10 modalités possibles cette fois-ci
- On veut tester H_0 : probabilités uniformes (soit 10% pour chaque catégorie)
- Contre H_1 : probabilités autres qu'uniformes

Il faut d'abord construire une statistique de test : une valeur observée qui ait un comportement connu et bien déterminé sous H_0, et un comportement différent sous H_1.

Etude d'une statistique de test

```
n = 116
# Génération de n variables aléatoires iid selon une loi uniforme sur {0, 1, 2, ..., 9} :
U = pd.Series(npr.random(n)*10).apply(int)
```

```
U.value_counts().sort_index()
```

```
0    11
1    11
2     9
3     9
4    14
5     8
6    16
7    13
8    15
9    10
dtype: int64
```

```
# Statistique de test : une certaine mesure de l'écart
# Dite Chi^2 de Pearson
T = ((U.value_counts().sort_index() - n/10)**2/(n/10)).sum()
T
```

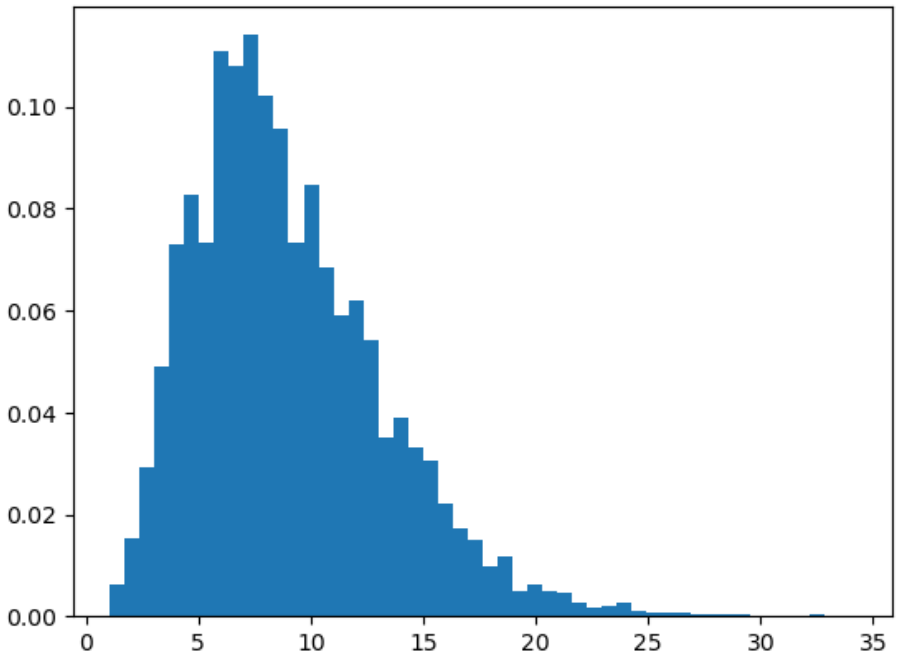
```
5.896551724137931
```

A quoi ressemble la loi de T ?

```
def varUnif(k):
    U = pd.Series(npr.random(k)*10).apply(int).value_counts().sort_index()
    T = ((U - k/10)**2/(k/10)).sum()
    return T
```

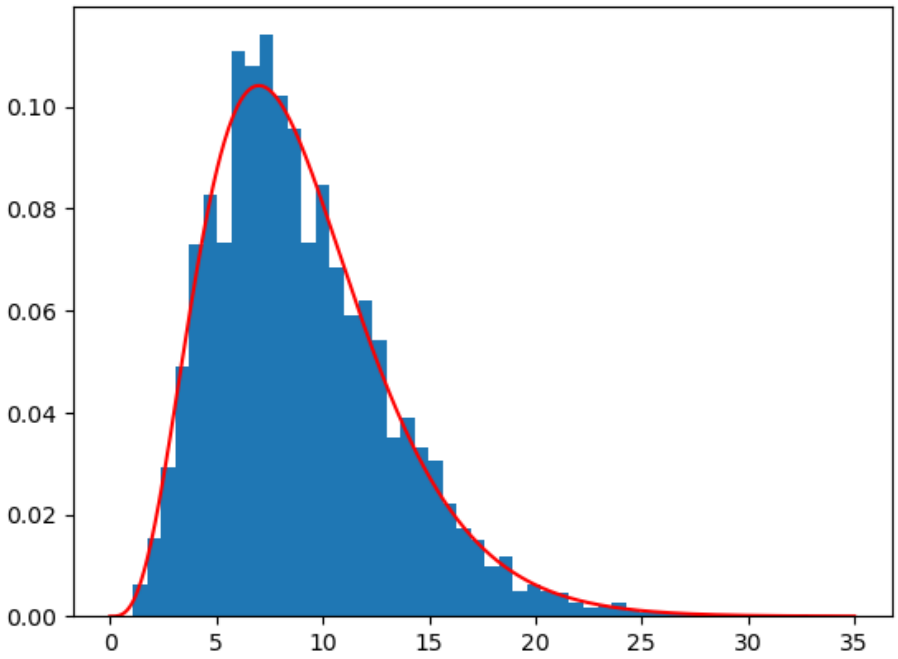
```
n = 116
N = 10000
Res = pd.Series(np.zeros(N))
for k in range(N):
    Res[k] = varUnif(n)
```

```
plt.hist(Res, density=True, bins=50)
plt.show()
```



En fait, la loi de T converge, quand n tend vers l'infini, vers une loi du χ^2 à 9 degrés de liberté

```
plt.hist(Res, density=True, bins=50)

ddl = 9
x = np.linspace(0, 35, 1000)
plt.plot(x, scs.chi2.pdf(x,ddl), 'r')

plt.show()
```



Jeu de données d'élections présidentielles 2009 en Iran

```
iran = pd.read_csv('Iran-Elections-2009.csv', sep=';')
iran
```

	Province	Candidate	Score
0	Ardabil	Ahmadinejad	325911
1	Azarbaijan, East	Ahmadinejad	1131111
2	Azarbaijan, West	Ahmadinejad	623946
3	Bushehr	Ahmadinejad	299357
4	Chahar Mahaal and Bakhtiari	Ahmadinejad	359578
..	...	...	...
111	Semnan	Rezaee	4440
112	Sistan and Baluchistan	Rezaee	6616
113	Tehran	Rezaee	147487
114	Yazd	Rezaee	8406
115	Zanjan	Rezaee	7276

[116 rows x 3 columns]

```
iran['Candidate'].value_counts()
```

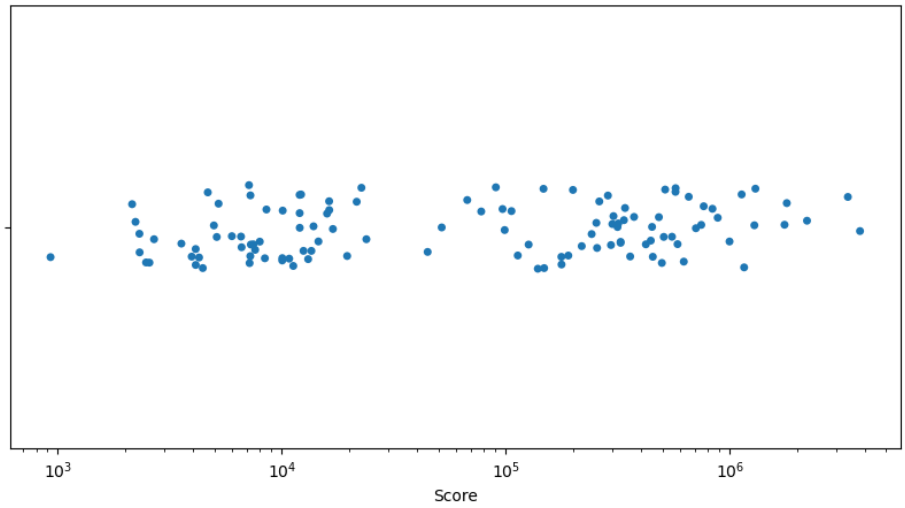
```
Ahmadinejad    29
Karroubi       29
Mousavi        29
```

```
Rezaee         29
Name: Candidate, dtype: int64
```

```
iran['Province'].value_counts()
```

```
Ardabil                         4
Khorasan, Razavi                4
Yazd                            4
Tehran                          4
Sistan and Baluchistan          4
Semnan                          4
Qom                             4
Qazvin                          4
Mazandaran                      4
Markazi                         4
Kurdistan                       4
Kohgiluyeh and Boyer-Ahmad      4
Khuzestan                       4
Khorasan, South                 4
Khorasan, North                 4
Azarbaijan, East                4
Kermanshah                      4
Kerman                          4
Isfahan                         4
Ilam                            4
Hormozgan                       4
Hamadan                         4
Golestan                        4
Gilan                           4
Fars                            4
Chahar Mahaal and Bakhtiari     4
Bushehr                         4
Azarbaijan, West                4
Zanjan                          4
Name: Province, dtype: int64
```

```
# Distribution des 4 x 29 = 116 scores
plt.figure(figsize=(10,5))
sns.stripplot(data=iran, x='Score', size=5).set_xscale('log')
plt.show()
```



Principe : si résultats authentiquement reportés, alors chiffres des unités uniformément distribués. Testons cela !

```
# Pour extraire le chiffre des unités :
def extrunité(k):
    return k - int(k/10)*10

# Exemples :
extrunité(156878), extrunité(351), extrunité(22)
```

```
(8, 1, 2)
```

```
iran['Unités'] = iran['Score'].apply(extrunité)
iran
```

	Province	Candidate	Score	Unités
0	Ardabil	Ahmadinejad	325911	1
1	Azarbaijan, East	Ahmadinejad	1131111	1
2	Azarbaijan, West	Ahmadinejad	623946	6
3	Bushehr	Ahmadinejad	299357	7
4	Chahar Mahaal and Bakhtiari	Ahmadinejad	359578	8
..	...	...	...	...
111	Semnan	Rezaee	4440	0
112	Sistan and Baluchistan	Rezaee	6616	6
113	Tehran	Rezaee	147487	7
114	Yazd	Rezaee	8406	6
115	Zanjan	Rezaee	7276	6

[116 rows x 4 columns]

```
I = iran['Unités'].value_counts().sort_index()
I
```

```
0     9
1    11
2     8
3     9
4    10
5     5
6    14
7    20
8    17
9    13
Name: Unités, dtype: int64
```

```
n = 116
T_iran = ((I - n/10)**2/(n/10)).sum()

# Valeur de la statistique de test sur ce jeu de données :
T_iran
```

```
15.551724137931032
```

```
# Est-ce une valeur typique ?
# Il faut voir que sous H1, T tend à prendre des valeurs plus grandes, de sorte qu'on appré
(Res >= T_iran).mean()
```

```
0.0784
```

C'est limite...

Jeu de données d'échange de magnets

Histoire : collection de magnets; au début, on n'a pas de doublons puis cela devient de plus en plus difficile d'obtenir au hasard des achats les magnets manquants. On a l'impression que certains reviennent plus souvent que d'autres, non ? Pourtant, ceux-ci sont (normalement) uniformément distribués. Testons cela...

Plateforme d'échange : https://www.brossard.fr/echange-magnet-brossard/


```
magnets = pd.read_csv('Magnets-Brossard.csv', sep=';')
magnets
```

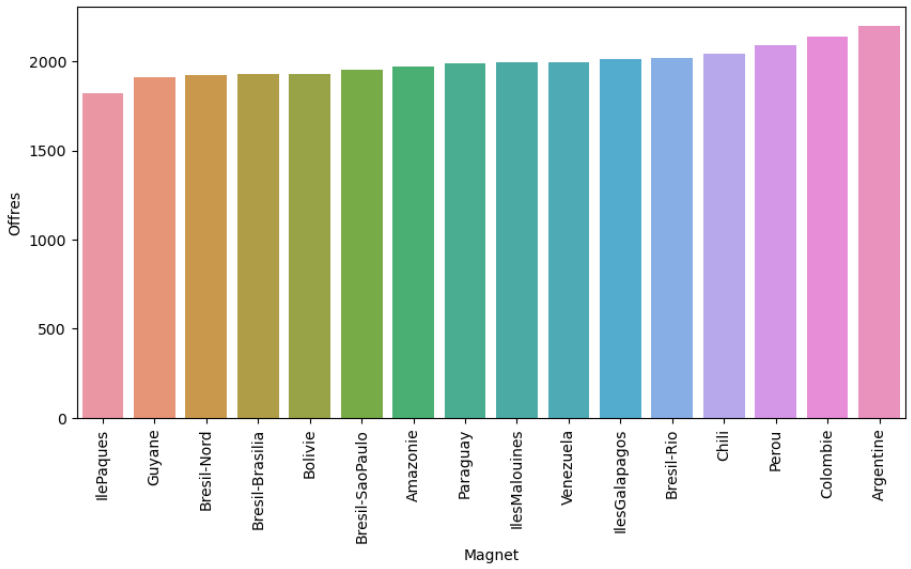
	Magnet	Offres
0	Colombie	2141
1	Venezuela	1998
2	Guyane	1909
3	Perou	2093
4	Amazonie	1971
5	Bresil-Nord	1922
6	Bresil-Rio	2018

```
7   Bresil-Brasilia    1928
8   Bresil-SaoPaulo    1956
9           Bolivie    1931
10         Paraguay    1991
11            Chili    2046
12        Argentine    2198
13    IlesGalapagos    2012
14        IlePaques    1824
15    IlesMalouines    1996
```

```
magnets['Offres'].min(), magnets['Offres'].max()
```

```
(1824, 2198)
```

```
plt.figure(figsize=(10,5))
sns.barplot(data=magnets.sort_values('Offres'), x='Magnet', y='Offres')
plt.xticks(rotation=90)
plt.show()
```




```
# Total :
n = magnets['Offres'].sum()
n
```

```
31934
```

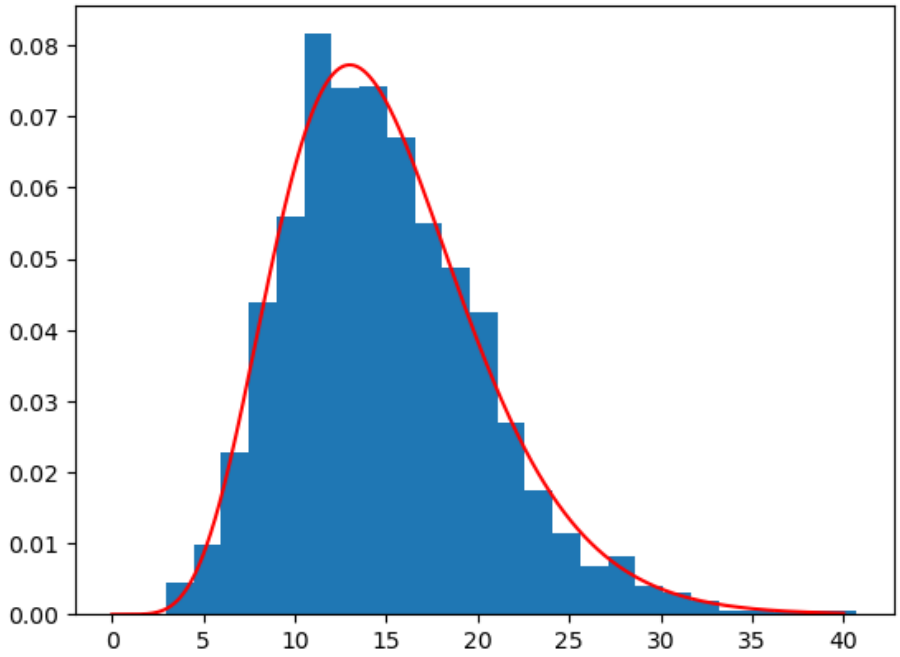
Re-simulons la loi de T avec cette valeur de n :

```
# Attention, il y a 16 modalités désormais
def varUnifMagnet(n):
    U = pd.Series(npr.random(n)*16).apply(int).value_counts().sort_index()
```

```
    T = ((U - n/16)**2/(n/16)).sum()
    return T

N = 2500
ResL = pd.Series(np.zeros(N))
for k in range(N):
    ResL[k] = varUnifMagnet(n)
```

Ici, la loi de T converge, quand n tend vers l'infini, vers une loi du χ^2 à 15 degrés de liberté

```
plt.hist(ResL, density=True, bins=25)

ddl = 15
x = np.linspace(0, 40, 1000)
plt.plot(x, scs.chi2.pdf(x,ddl), 'r')

plt.show()
```



Sur les données, T vaut :

```
T_magnets = ((magnets['Offres'] - n/16)**2/(n/16))
T_magnets
```

```
0     10.552397
1      0.002262
```

```
2      3.781432
3      4.726381
4      0.310022
5      2.734398
6      0.245264
7      2.308269
8      0.796651
9      2.108732
10     0.011907
11     1.258854
12    20.469476
13     0.130277
14    14.801035
15     0.000008
Name: Offres, dtype: float64
```

```
T_magnets.sum()
```

```
64.23736456441411
```

```
max(ResL)
```

```
40.70382664245005
```

On rejette donc très fermement l'hypothèse H_0 que les mangets proposés à l'échange sont uniformément distribués (cela n'implique pas nécessairement qu'en production, ils ne le soient pas : il y a peut-être juste des attachements ou répulsions subjectifs des consommateurs à certains magnets). Les déviations les plus marquantes par rapport à l'effectif attendu $n/16 = 1995.875$ sont pour ceux d'indices 0, 12 et 14 :

```
magnets.iloc[[0,12,14],:]
```

```
       Magnet  Offres
0    Colombie    2141
12  Argentine    2198
14  IlePaques    1824
```