

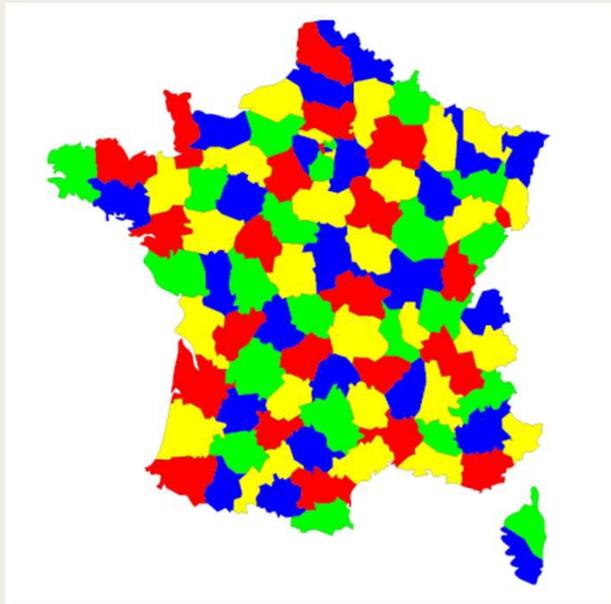


JOURNÉE MATHÉMATIQUES ET NUMÉRIQUE

Démontrer avec les outils numériques



Introduction : le théorème des 4 couleurs



Source : www.math93.com

Le théorème des quatre couleurs énonce la possibilité de colorier (on dit aussi colorer en théorie des graphes) avec quatre couleurs seulement une carte géographique sans que deux pays voisins aient la même couleur.

En 1976, le mathématicien américain Kenneth Appel et le mathématicien allemand Wolfgang Haken réalisent le programme de Heesch.

Ils montrent en utilisant des dizaines de milliers de figures, que toute carte non 4-coloriable doit contenir l'une des 1478 configurations, et, avec 1200 heures de calcul, que ces configurations sont réductibles.

C'est en fait la première preuve informatisée d'un théorème mathématique !

Situation 1 : un classique revisité

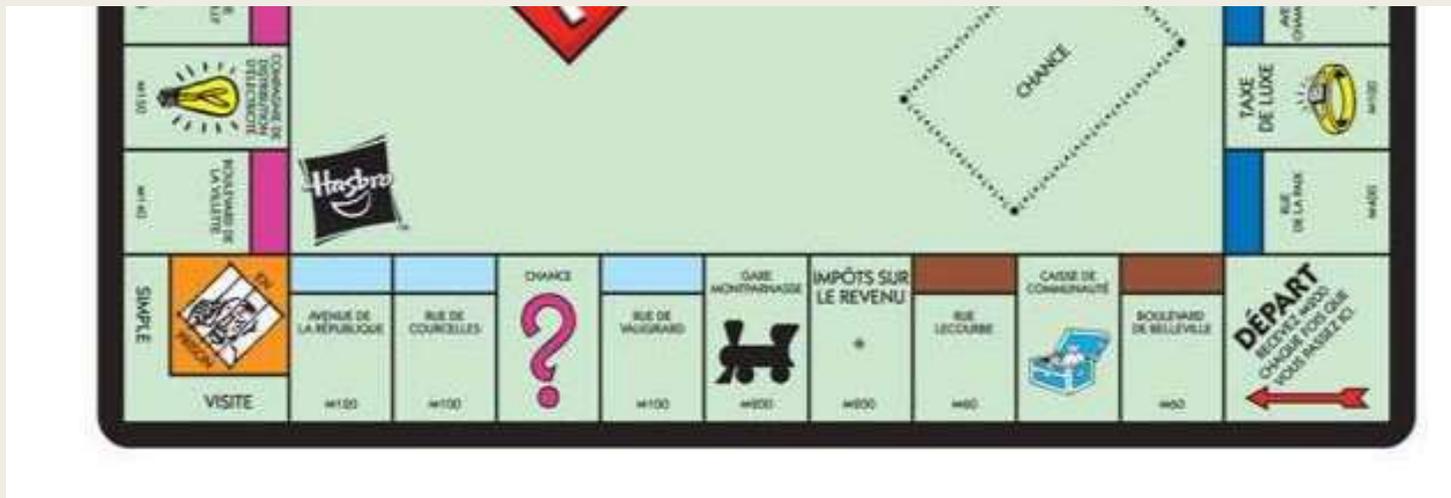
Problème du Grand Duc de Toscane : Le Grand Duc de Toscane, qui avait une grande expérience des jeux de hasard, avait remarqué que, lorsqu'on lance 3 dés à six faces, la somme des points sur les faces apparentes est plus souvent 10 que 9.

Or, en ajoutant 3 nombres entre 1 et 6, il y a 6 façons différentes d'obtenir la somme 10 et 6 façons différentes d'obtenir la somme 9. Le Grand Duc voyait là un paradoxe. Il posa la question à Galilée (qui était mathématicien de l'université de Pise).

Situation 1 : un classique revisité

Revenons à la somme de 2 dés pour simplifier : Le jeu du Monopoly

Quelle est la probabilité de tomber sur une case avec un bandeau de couleur au premier lancer ?



Situation 1 : un classique revisité

Dans ces deux situations, qu'apporte la simulation ?

Comment démontrer que l'hypothèse d'équiprobabilité posée sur les 11 cases atteignables du jeu est fautive, tout comme l'hypothèse d'équiprobabilité posée sur l'univers des combinaisons par le Grand Duc de Toscane ?

Situation 1 : un classique revisité

Dans ces deux situations, qu'apporte la simulation ?

La simulation permet seulement d'approcher les probabilités dans l'univers sur lequel on aura posé une hypothèse d'équiprobabilité.

=ALEA.ENTRE.BORNES(2;12) : spontanément proposé et cohérent avec l'hypothèse d'équiprobabilité des cases.

=ALEA.ENTRE.BORNES(1;6)+ALEA.ENTRE.BORNES(1;6) : suppose d'avoir déjà dépassé cette erreur, on tourne en rond!!

Situation 1 : un classique revisité

Comment démontrer que l'hypothèse d'équiprobabilité posée sur les 11 cases atteignables du jeu est fausse, tout comme l'hypothèse d'équiprobabilité posée par le Grand Duc de Toscane ?

Ce n'est pas la simulation qui permet de contredire une fausse hypothèse d'équiprobabilité mais l'expérimentation, comme le Grand Duc qui a pu observer que son hypothèse n'était pas vérifiée par l'expérience.

Situation 1 : un classique revisité

Une fois l'univers aux issues équiprobables validé par l'expérimentation (différencier les dés et prendre l'ordre en compte), utiliser un algorithme pour obtenir les probabilités de la somme des 2 dés puis de 3 dés en dénombrant.

Situation 1 : un classique revisité

```
1 tirages = [[d1 , d2] for d1 in range(1 , 7) for d2 in range(1 , 7)]
2 effectifs = [0]*12
3
4 for t in tirages:
5     effectifs[sum(t)-1] += 1
6
7 loi = [i/sum(effectifs) for i in effectifs]
```

Sur 2 dés, la démonstration par dénombrement à la main est facilement accessible.

```
*** Console de processus distant Réinitialisée ***
>>>
>>> effectifs
[0, 1, 2, 3, 4, 5, 6, 5, 4, 3, 2, 1]
>>> sum(effectifs)
36
```

Situation 1 : un classique revisité

```
1 tirage = [[d1 , d2 , d3]
2     for d1 in range(1 , 7)
3     for d2 in range(1 , 7)
4     for d3 in range(1 , 7)]
5 effectifs = [0]*18
6
7 for t in tirage:
8     effectifs[sum(t)-1] += 1
9
10 loi = [i/sum(effectifs) for i in effectifs]
```

Sur 3 dés, la construction de l'algorithme pour dénombrer commence à trouver sa justification.

```
*** Console de processus distant Réinitialisée ***
>>>
>>> effectifs
[0, 0, 1, 3, 6, 10, 15, 21, 25, 27, 27, 25, 21, 15, 10, 6, 3, 1]
>>> sum(effectifs)
216
```

Prolongement : le jeu de dés (IREM)

On considère l'expérience aléatoire suivante : dans un premier temps on lance un dé puis, dans un deuxième temps, on relance autant de dés que l'indique le résultat obtenu à l'issue du premier lancer.

On observe la somme des points obtenus au 2^e lancer.

Question : quelle est, parmi toutes les sommes possibles, celle qui est la plus probable?

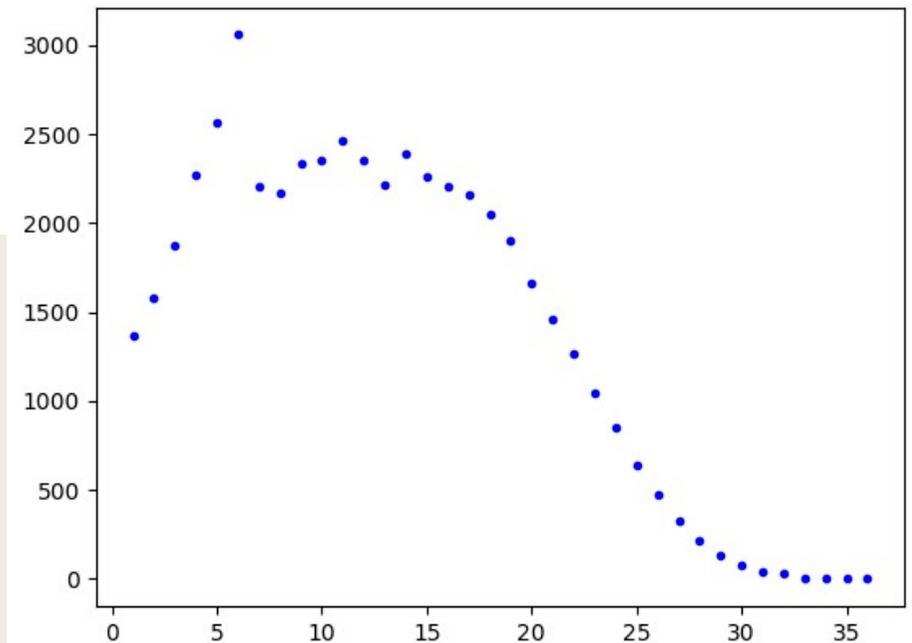
Jeu de dés : simulation

```
11 def lancer():
12     n = random.randint(1, 6)
13     tot = 0
14     for i in range(n):
15         tot = tot + random.randint(1, 6)
16     return tot
```

```
20 def effectifs(N):
21     result = [ 0]*36
22     for j in range (N):
23         val = lancer()
24         result[val - 1] += 1
25     return(result)
```

```
29 pylab.plot(range(1,37),effectifs(50000),'b.')
```

```
30 pylab.show()
```



Jeu de dés : dénombrement

On stocke dans une liste à la i-ème position le nombre de combinaisons donnant la somme i pour tous les cas : 1 dé, 2 dés, etc.

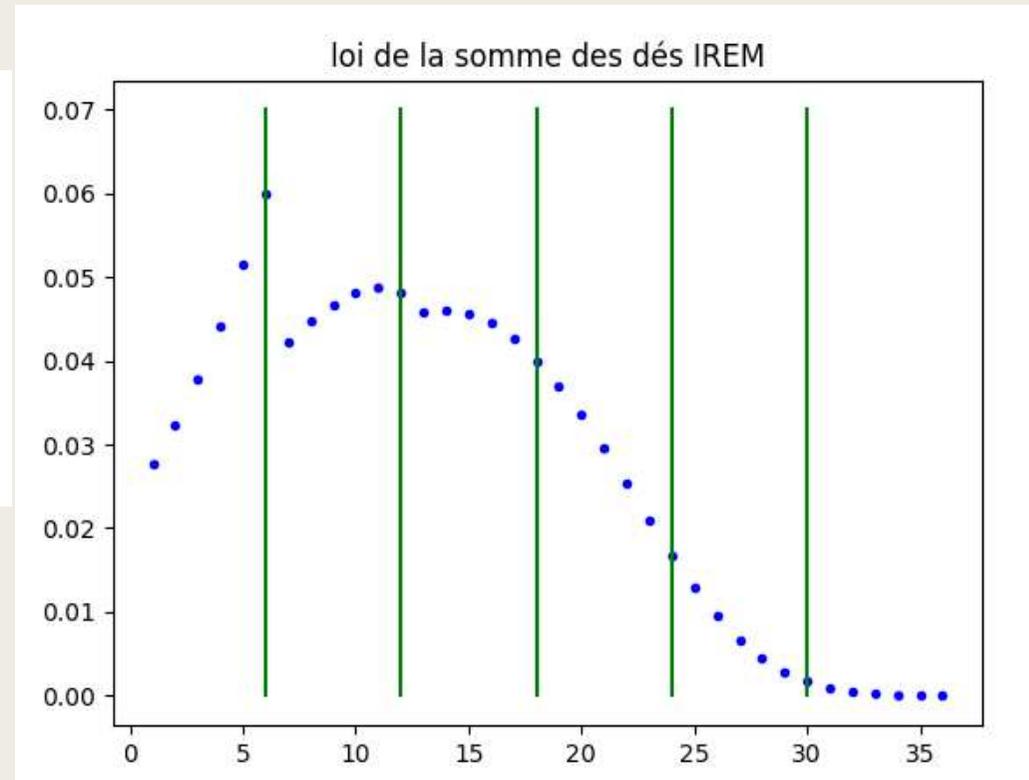
Toutes les listes sont de longueur 36 pour faciliter la mise en commun finale. On réunit toutes les possibilités comme sur un arbre avec leur probabilités.

```
def dés4():
    tirage = [[d1 , d2 , d3 , d4]
              for d1 in range(1 , 7)
              for d2 in range(1 , 7)
              for d3 in range(1 , 7)
              for d4 in range(1 , 7)]
    eff4 = [0]*36
    for t in tirage:
        eff4[sum(t)-1] += 1
    return eff4
```

```
def loi():
    probas = [0]*36
    for i in range(36):
        probas [i] = dés1()[i]/6**2 + dés2()[i]/6**3 + dés3()[i]/6**4 + dés4()[i]/6**5 + dés5()[i]/6**6 + dés6()[i]/6**7
    return probas
```

Jeu de dés : dénombrement

```
# affichage graphique de la loi de probabilité  
pylab.plot(range(1, 37), loi(), 'b.')  
pylab.plot([6, 6], [0, 0.07], 'g-')  
pylab.plot([12, 12], [0, 0.07], 'g-')  
pylab.plot([18, 18], [0, 0.07], 'g-')  
pylab.plot([24, 24], [0, 0.07], 'g-')  
pylab.plot([30, 30], [0, 0.07], 'g-')  
pylab.title("loi de la somme des dés IREM")  
pylab.show()
```



Situation 2 : le capitaine de troupe

Un capitaine de troupe veut ranger ses « hommes » en rang.
Mais il reste embarrassé car :

- S'il les range par 5, il en reste 4 non rangés
- S'il les range par 6, il en reste 5 non rangés
- S'il les range par 7, il en reste 6 non rangés

Combien y a-t-il d'hommes à ranger sachant qu'il ne peut pas y en avoir plus de 400 ?

D'après la brochure n° 154 APMEP « Pour un enseignement problématisé des mathématiques au lycée »

Situation 2 : le capitaine de troupe

La solution peut être trouvée à l'aide du tableur en envisageant tous les cas possibles.

	A	B	C	D	E	F	G	H	I
1	Nb d'hommes	rang par 5	test 1	rang par 6	test 2	rang par 7	test 3	test final	nb de solutions
2	1	1	0	1	0	1	0	0	1
3	2	2	0	2	0	2	0	0	
4	3	3	0	3	0	3	0	0	
5	4	4	1	4	0	4	0	1	
6	5	0	0	5	1	5	0	1	

Il s'agit bien d'une démonstration.

	A	B	C	D	E	F	G	H	I
1	Nb d'hommes	rang par 5	test 1	rang par 6	test 2	rang par 7	test 3	test final	nb de solutions
2	1	=MOD(A2;5)	=SI(B2=4;1;0)	=MOD(A2;6)	=SI(D2=5;1;0)	=MOD(A2;7)	=SI(F2=6;1;0)	=C2+E2+G2	=NB.SI(H2:H402;3)
3	2	=MOD(A3;5)	=SI(B3=4;1;0)	=MOD(A3;6)	=SI(D3=5;1;0)	=MOD(A3;7)	=SI(F3=6;1;0)	=C3+E3+G3	
4	3	=MOD(A4;5)	=SI(B4=4;1;0)	=MOD(A4;6)	=SI(D4=5;1;0)	=MOD(A4;7)	=SI(F4=6;1;0)	=C4+E4+G4	

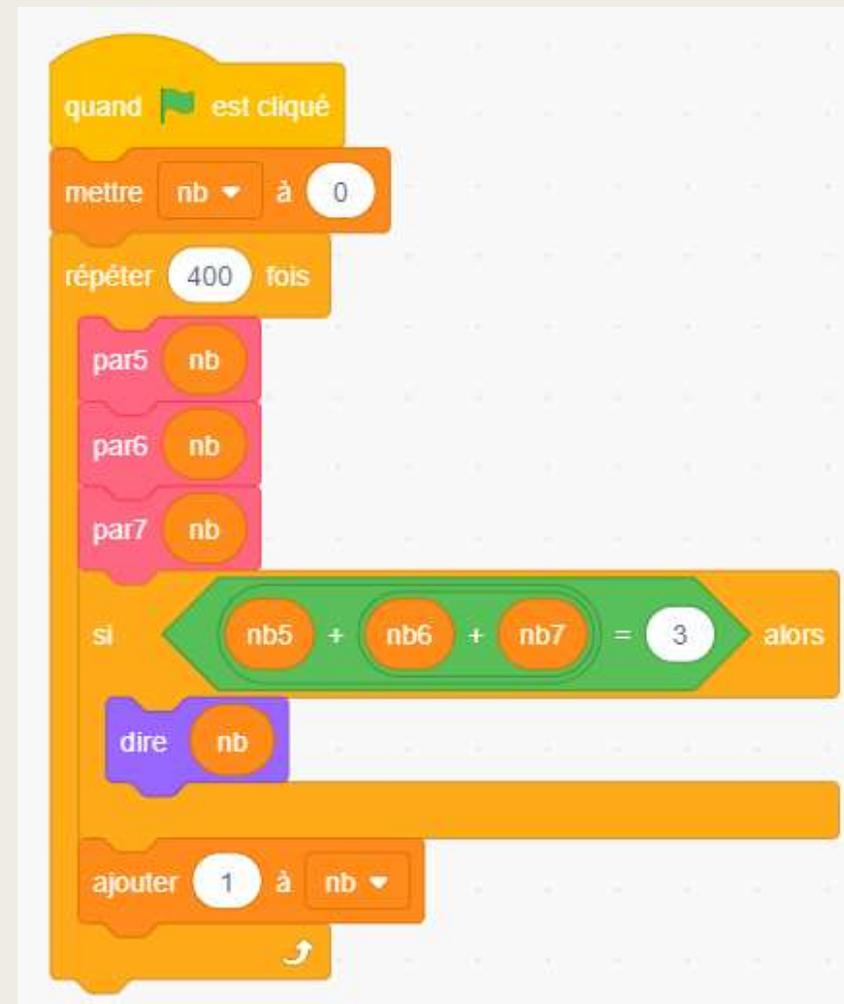
Situation 2 : le capitaine de troupe

La version tableur peut servir de base pour construire ensuite l'algorithme en python.

```
1 def par5(x):
2     if x % 5 == 4: return 1
3     else: return 0
4
5 def par6(x):
6     if x % 6 == 5: return 1
7     else: return 0
8
9 def par7(x):
10    if x % 7 == 6: return 1
11    else: return 0
12
13 def solution():
14    for i in range(401):
15        if par5(i) + par6(i) + par7(i) == 3:
16            return i
17
```

Situation 2 : le capitaine de troupe

La version tableur peut servir de base pour construire ensuite l'algorithme en scratch.



Situation 2 : le capitaine de troupe

La démonstration passant par les critères de divisibilité permet simplement de diminuer le nombre de cas à tester par rapport à la démonstration numérique.

- S'il les range par 5, il en reste 4 non rangés

$a - 4$ se termine par 0 ou 5 donc a se termine par 4 ou 9.

- S'il les range par 6, il en reste 5 non rangés

$a - 5$ se termine par un chiffre pair, donc a se termine par un chiffre impair, donc 9.

Situation 2 : le capitaine de troupe

- S'il les range par 7, il en reste 6 non rangés

$a - 6$ se termine par 3 et est un multiple de 7.

En observant les premiers multiples de 7, on retient que seul 63 vérifie cette condition.

Alors, le quotient entier de $a - 6$ par 7 se termine par 9.

Les quotients possibles sont : 9, 19, 29, 39, 49.

Les valeurs à tester pour a sont donc : 69, 139, 209, 279, 349.

Seul le nombre 209 vérifie les 3 conditions.

- C'est en fait une application du théorème des restes chinois.