Comparaison de systèmes électoraux

Essifi Yassine,14658

2021

T.I.P.E

Table des matières

Introduction

Contextualisation :

Dans plusieurs pays autour du monde(ex : États Unis) on remarque la présence de deux parties dominants.
Pourquoi ?
$\rightarrow$ Le scrutin uninominal majoritaire à un tour.

- Existe t-il d'autres systèmes de votes ?
- Comment peut-on les comparer ?

Systèmes comparées :

Les systèmes considérées seront des modes de scrutins uninominaux.

Le scrutin uninominal majoritaire à un tour :

- Chaque électeur vote pour ***le*** candidat qu'il préfère.
- Le candidat ayant rassemblé le plus de voix est élu en toutes circonstances.

Le vote par approbation :

- Chaque électeur vote pour ***les*** candidats qu'il préfère.
- Le candidat ayant rassemblé le plus de voix est élu en toutes circonstances.

Le vote à second tour instantané :

1. Chaque électeur classe les candidats par ordre de préférence.
2. On décompte les premières préférences.
3. Si un candidats a la majorité absolue, il l'emporte.
4. Sinon le candidats à la dernière position, les préférences seront réordonnés, puis on retourne a l'étape 2.

Procédure de comparaison

Théorème d'impossibilité d'Arrow :

Pour au moins trois candidats et deux électeurs, il n'existe pas de système électorale satisfaisant les propriétés suivantes :

1. L'Universalité.
2. La Non-dictature.
3. La Monotonie.
4. La Souveraineté.
5. L'Indépendance des options non pertinentes.

$\Rightarrow$ Comparaison des cas d'échecs et de leurs fréquences.

Méthode de simulation

Modélisation et Hypothèses

Espace Politique :

- L'opinion politique $=$ Point dans un plan.
- $x, y \in [-0.25, 1.25]^2$
- Subdivision de l'espace en 62500 points
- Répartition des électeurs $\sim \mathcal{N}(., 0.5^2)$ suivant chaque axe autour d'un centre d'intérêt.
- Nombre d'électeurs=4000

Modélisation et Hypothèses

Le scrutin majoritaire à un tour :

Les électeurs choisissent le candidat le plus proche idéologiquement.

Vote par approbation :

- Chaque électeur choisit les candidats a une distance acceptable.
- Les distances acceptables $\sim Log\mathcal{N}(-0.5, 0.5^2)$

Le vote à second tour instantané :

Les électeurs classent les candidats par distance.

Explication du code :

Fonctions de base :

```
import numpy as np
import math as mt
def dis(A,B):
    return mt.sqrt((A[0]-B[0])**(2)+(A[1]-B[1])**(2
def pref1(voteur,candidats):
    T=np.zeros((len(candidats),2))
    for i in range(len(candidats)):
        T[i]=(dis(voteur,candidats[i]),i)
    T=T[T[:,0].argsort()]
    return T
```

On définit la notion de distance et de préférence pour un seul électeur.

Explication du code :

```
def preftt(voteurs,candidats):
    T=np.zeros((len(voteurs),len(candidats)))
    for i in range(len(voteurs)):
        T[i]=pref1(voteurs[i],candidats)[:,1]
    return T
def nbv1(voteurs,candidats):
    V=[]
    P=preftt(voteurs,candidats)
    for i in range(len(candidats)):
        c=0
        for j in range(len(voteurs)):
            if int(P[j][0])==i:
                c=c+1
        V.append(c)
    return V
```

On crée une matrice numpy contenant les préférences des électeurs suivant leur ordres, puis on calcul le nombre de première préférence.

Explication du code :

Scrutin Majoritaire à un tour :

```
def plur(voteurs,candidats):
    return nbv1(voteurs,candidats).index(max(nbv1(voteurs,cand
```

Cette fonction retourne l'indice du candidat qui est le plus classé comme première préférence. C'est à dire le vainqueur en cas de scrutin majoritaire à un tour.

Explication du code :

Vote à second tour instantanée :

```
def irv(voteurs,candidats):
    cand=np.copy(candidats)
    candn=[]
    for i in range(len(cand)):
        candn.append((cand[i],i))
    V=nbv1(voteurs,cand)
    vm=max(V)
    p=vm/(len(voteurs))
    while p<0.5 :
        i=V.index(min(V))
        print(i)
        cand=np.delete(cand,i,0)
        candn.pop(i)
```

```
^^IV=nbv1(voteurs,cand)
^^Ivm=max(V)
^^Ip=vm/(len(voteurs))
 return candn[plur(voteurs,cand)][1]
```

On calcul si un des candidats a une majorité absolue et tant que c'est n'est pas vrai on élimine le dernier candidat et on redistribue les votes,cette fonction retourne l'indice du vainqueur par vote à second tour instantanée.

Explication du code :

Fonctions de base :

```
def votecircle1(voteur,candidats,r):
^^IL=np.zeros(len(candidats))
^^Ifor i in range(len(candidats)):
^^I^^Iif dis(voteur,candidats[i])<=r:
^^I^^I^^IL[i]=1
^^Ireturn L
def votecirclett(voteurs,candidats,r):
^^IT=np.zeros((len(voteurs),len(candidats)))
^^Ifor i in range(len(voteurs)):
^^I^^IT[i]=votecircle1(voteurs[i],candidats,r[i])
^^Ireturn T
```

On détermine les candidats inclus dans le cercle de rayon r autour d'un seul électeur. Puis on crée une matrice numpy contenant le choix de chaque électeur.

Explication du code :

Vote par approbation :

```
def nbvc(voteurs,candidats,r):
V=[]
P=votecirclett(voteurs,candidats,r)
^^Ifor i in range(len(candidats)):
^^I^^Ic=0
^^I^^Ifor j in range(len(P)):
^^I^^I^^Ic=c+(int(P[j][i]))
^^I^^I^^IV.append(c)
^^Ireturn V
def appr(voteurs,candidats,r):
^^IA=nbvc(voteurs,candidats,r)
^^Ireturn A.index(max(A))
```

On compte le nombre de votes pour chaque candidat puis on retourne l'indice du candidat avec le plus grand nombre de votes.

Exploitation de la simulation et résultats

1er Cas d'étude : Candidats a distances égales

Figure 1 – Vote de pluralité

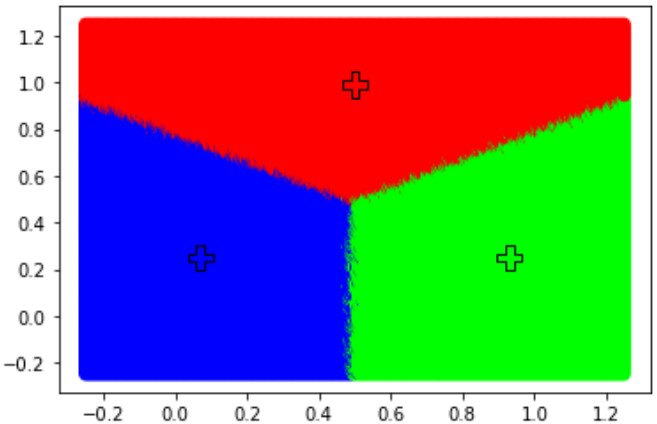


Figure 2 – Vote par approbation

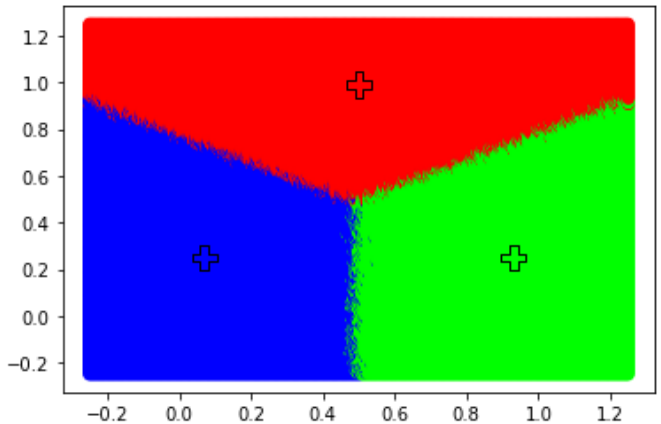
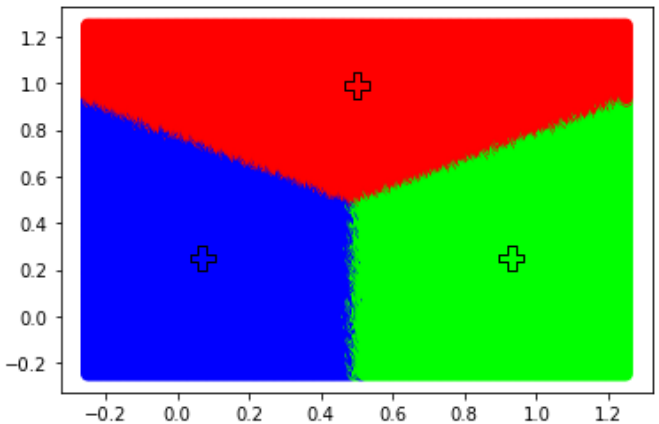


Figure 3 – Vote à second tour instantané



2ieme Cas d'étude : Fractionnement des voix

Figure 1 – Vote de pluralité

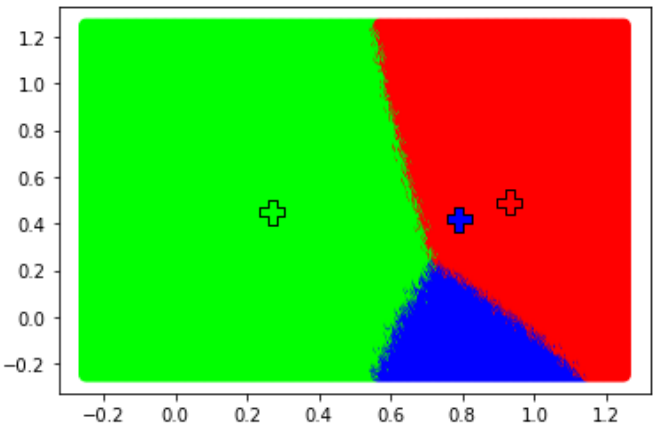


Figure 2 – Vote par approbation

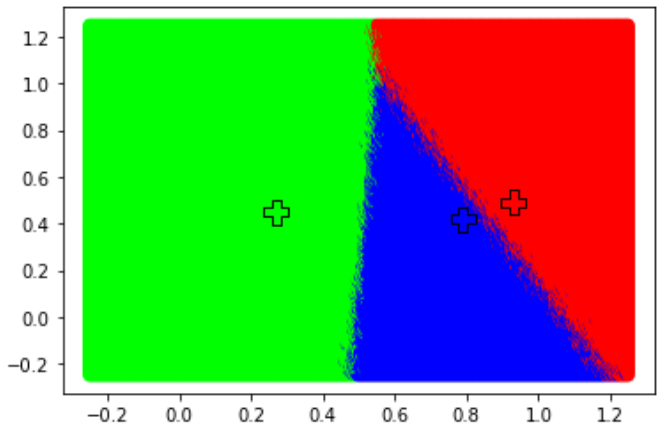
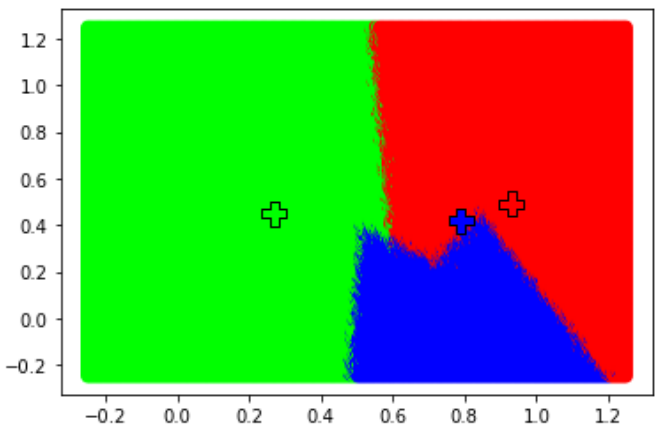


Figure 3 – Vote à second tour instantané



3ieme Cas d'étude : Évincement d'un candidat

Figure 1 – Vote de pluralité

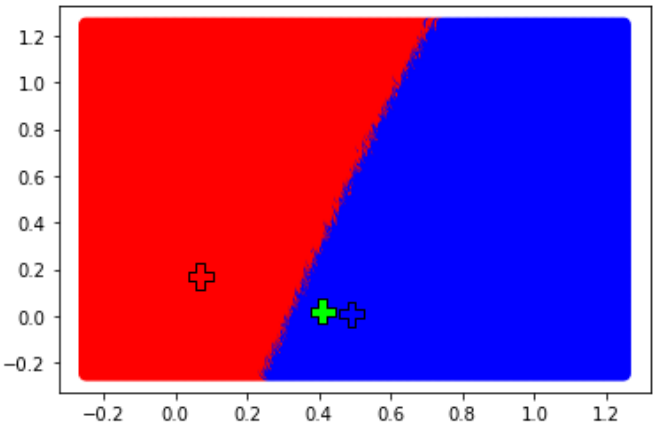


Figure 2 – Vote par approbation

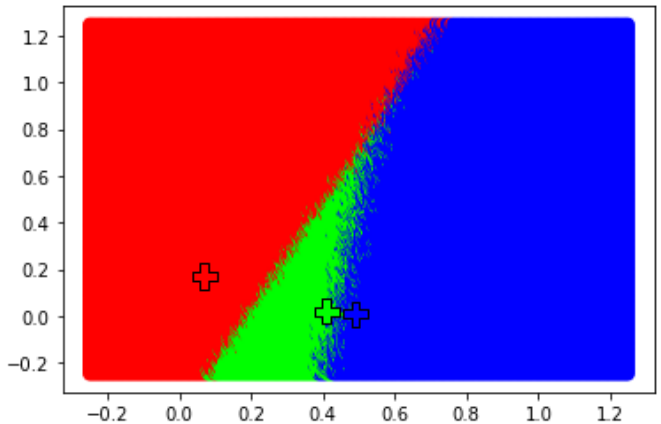
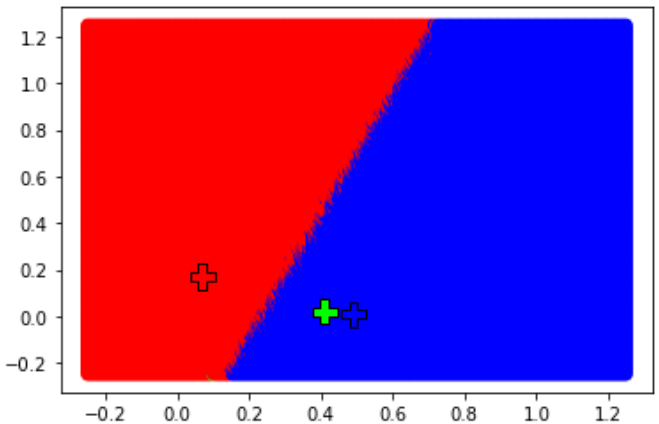


Figure 3 – Vote à second tour instantané



4ieme Cas d'étude : Critère de monotonicité

Figure 1 – Vote de pluralité

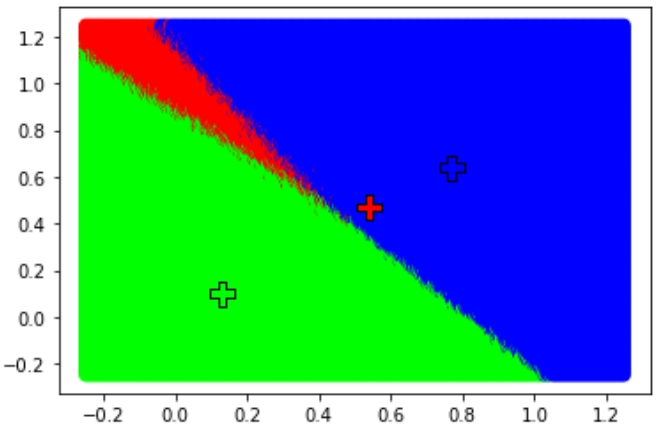


Figure 2 – Vote par approbation

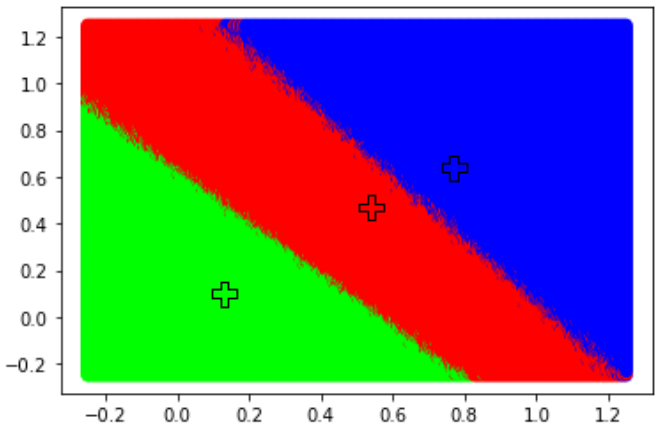
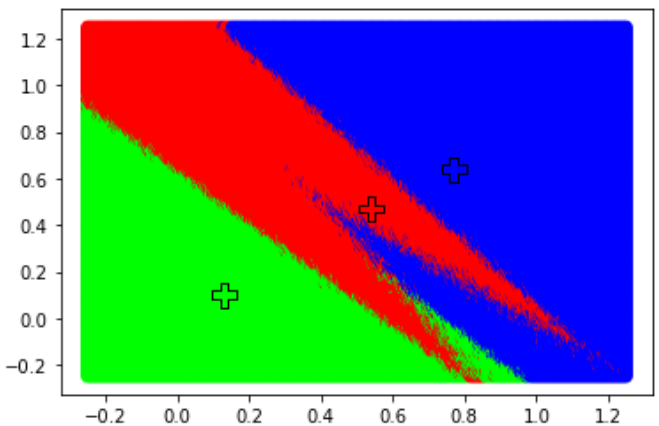


Figure 3 – Vote à second tour instantané



Les failles et les stratégies exploitables :

Vote de pluralité

- Faille :Évincement d'un candidat
- Stratégie : Choisir seulement un des deux candidats les plus populaires

Vote à second tour instantané :

- Faille :Non respect du critère de monotonie (conséquence : effet "center-squeeze").
- Stratégie : Allez parfois contre le favori.

Conclusion

```python
import concurrent.futures
import matplotlib.pyplot as plt
import numpy as np
import math as mt

#def distance:
def dis(A,B):
    return mt.sqrt((A[0]-B[0])**(2)+(A[1]-B[1])**(2))

#Classement pref 1 voteur:
def pref1(voteur,candidats):
    T=np.zeros((len(candidats),2))
    for i in range(len(candidats)):
        T[i]=(dis(voteur,candidats[i]),i)
    T=T[T[:,0].argsort()]
    return T
#Classement pref tt les voteurs:
def preftt(voteurs,candidats):
    T=np.zeros((len(voteurs),len(candidats)))
    for i in range(len(voteurs)):
        T[i]=pref1(voteurs[i],candidats)[:,1]
    return T
```

```python
#nbr de premiere pref
def nbv1(voteurs,candidats):
    V=[]
    P=preftt(voteurs,candidats)
    for i in range(len(candidats)):
        c=0
        for j in range(len(voteurs)):
            if int(P[j][0])==i:
                c=c+1
        V.append(c)
    return V
#Plur
def plur(voteurs,candidats):
    A=nbv1(voteurs,candidats)
    return A.index(max(A))
#VASTI
def irv(voteurs,candidats):
    cand=np.copy(candidats)
    candn=[]
    for i in range(len(cand)):
        candn.append((cand[i],i))
    V=nbv1(voteurs,cand)
```

```
    vm=max(V)
    p=vm/(len(voteurs))
    while p<0.5 :
        i=V.index(min(V))
        print(i)
        cand=np.delete(cand,i,0)
        candn.pop(i)
        V=nbv1(voteurs,cand)
        vm=max(V)
        p=vm/(len(voteurs))
    return candn[plur(voteurs,cand)][1]
#Cercle d'approbation 1 voteur:
def votecircle1(voteur,candidats,r):
    L=np.zeros(len(candidats))
    for i in range(len(candidats)):
        if dis(voteur,candidats[i])<=r:
            L[i]=1
    return L
#Cercle d'approbation 1 voteur:
def votecirclett(voteurs,candidats,r):
    T=np.zeros((len(voteurs),len(candidats)))
    for i in range(len(voteurs)):
```

```
        T[i]=votecircle1(voteurs[i],candidats,r[i])
    return T
#nbr de vote
def nbvc(voteurs,candidats,r):
    V=[]
    P=votecirclett(voteurs,candidats,r)
    for i in range(len(candidats)):
        c=0
        for j in range(len(P)):
            c=c+(int(P[j][i]))
        V.append(c)
    return V
#approbation:
def appr(voteurs,candidats,r):
    A=nbvc(voteurs,candidats,r)
    return A.index(max(A))
#simulation:
d= np.mgrid[-0.25:1.25:250j, -0.25:1.25:250j].reshape(2,-1).T
candidats1=np.array([[0.5,0.99],[0.07,0.25],[0.93,0.25]])
candidats2=np.array([[0.93,0.49],[0.79,0.42],[0.27,0.45]])
candidats3=np.array([[0.07,0.17],[0.49,0.01],[0.41,0.02]])
candidats4=np.array([[0.54,0.47],[0.77,0.64],[0.13,0.10]])
```

```
L=[candidats1,candidats2,candidats3,candidats4]
C=['#ff0000','#0000ff','#00ff00']

def color(c):
    #pour appr : r=np.random.lognormal(-0.5,0.5,10000)
    voters=np.random.multivariate_normal(c,[[0.25,0],[0,0.25]],10000)
    return C[irv(voters,L[3])]
d1=d.tolist()
X,Y=d.T
x,y=L[3].T
#utilisation de plusieurs coeurs par subdivision du calcul des couleur
#chaque point
if __name__ == '__main__':
    with concurrent.futures.ProcessPoolExecutor() as executor:
        r=executor.map(color,d1)
        C1=list(r)
        plt.scatter(X,Y,c=C1)
        plt.scatter(x,y,s=200,c=C,marker="P",edgecolors='black')
        plt.show()
```