Goffinet Daniel pour CultureMaths

Reports de voix : problèmes et solutions

Dans le système électoral Français il y a souvent deux tours :

- Dans les législatives ceux ayant eu 10% (c'est peut-être 15 %) au premier peuvent se maintenir au second (et j'ignore ce qui se passerait si 20 candidats avaient chacun 5%)
- Dans les municipales .... je ne sais pas
- Pour les présidentielles seuls les deux premiers passent le premier tour

Ce qui suit est raconté avec un vocabulaire "Elections Législatives Françaises" : il y a deux tours, des abstentions, les candidats du second tour sont parmi ceux qui avaient été présents au premier tour

Après le 2-ième tour, c'est alors un jeu traditionnel pour tous les instituts de sondages : le calcul des reports, ce sont les pourcentages des électeurs des candidats du premier tour (éliminés ou non) qui se sont reportés sur chacun des candidats qualifiés

Vous avez tous lu des affirmations comme "53 % des électeurs de Bernard au premier tour ont reporté leurs votes sur Dupont". Les votes étant à bulletins secrets, comment peut-on savoir cela ?

Une première remarque est que l'on ne peut évidemment pas dire : Un exemple est :

4 candidats (A,B,C,D), 10 votants, au premier tour a1=4, b1=4, c1=1,d1=1

dans la configuration "présidentielles" seuls A et B vont au second tour et ils obtiennent a2=4 et b2=4

Une configuration vraisemblable des reports est : les électeurs de A et B du premier tour ont voté pareil, ceux de C et D se sont abstenus En fait il peut s'être passé tout autre chose ... jusqu'à l'échange des électorats de A et B

- Les résultats du Premier Tour constituent un tableau PT de données : on va mettre dans chaque ligne un des bureaux de vote, dans chaque colonne un des candidats du premier tour (la dernière colonne étant pour le candidat "abstention+votes blancs et nuls", noté A ci-dessous)
- Les résultats du Second Tour constituent un tableau ST de données : on va mettre dans chaque ligne un des bureaux de vote, dans chaque colonne un des candidats du second tour (la dernière colonne étant pour le candidat "abstention+votes blancs et nuls", noté A ci-dessous)
- Les Reports que l'on veut calculer sont des pourcentages : ce sont les reports des électeurs d'un des candidats du premier tour sur un des candidats du second tour, ces reports constituent eux aussi un tabeau R : c'est cela que l'on cherche

De premiers exemples se trouvaient dans la partie 'préambulaire'

A Deux résultats de maths

Tous les coefficients sont ici réels, la norme utilisée ici est la $||\ \ ||_2$ usuelle, juste notée $||\ \ ||$

A est une matrice de $M_{p,q}$ et B de $M_{p,r}$ on sait que $p \geq q$ et que le rang de A est q. (on y reviendra en dernière partie)

U_n est la colonne de 1 de hauteur n : multiplier une matrice M par U donne la somme de ses colonnes, si donc M est une matrice dont les lignes sont des pourcentages (ou des probabilités) on obtient 100U (ou U)

Une pseudo solution de A X =? B est une S de $M_{q,r}$ telle que $||AS - B||$ soit minimum

1 S existe et est unique, c'est la solution de ${}^tAAX = {}^tAB$

2 si de plus il existe un s tel que $AU_q = sU_p$ et $BU_r = sU_p$ alors $SU_r = U_q$

Preuve1 : si V est dans $\ker({}^tAA)$ alors ${}^tV{}^tAAV = 0$, soit $||AV = 0||$ or $\text{rang}(A) = q = \dim(\text{Source}(A))$ donc $\dim(\ker(A)) = 0$ donc $V = 0$. La matrice carrée ${}^tAA$ étant inversible l'unique S est $({}^tAA)^{-1}({}^tAB)$

Preuve2 : de $AU_q = sU_p$ et $BU_r = sU_p$ découlent

$$ {}^tAsU_p = {}^tABU_r = {}^tA(AS)U_r = {}^tABU_r = s\,{}^tAU_p = {}^tAAU_q $$

l'inversibilité de ${}^tAA$ fait alors conclure

B On programme cela

```
import numpy as np
import numpy.linalg as alg

"""
Ci-dessous
NCPT = Nombre de Candidats au Premier Tour
(l'un d'entre eux regroupe : abstentions, blancs, nuls)
```

```
NCST = Nombre de Candidats au Second Tour
(l'un d'entre eux regroupe : abstentions, blancs, nuls)
NEB = Nombre d' Electeurs par Bureau
NBV = Nombre de Bureaux de Vote
"""
NCPT = 5
NCST = 3
NEB = [20,30,25,28,31,25,22,29,26]
NBV = len(NEB)

"""
Ci-dessous sont exprimés en pourcentages :
debut_PT =la liste des résultats du premier tour
    concernant tous les candidats sauf le dernier (regroupant : abstentions, blancs, nuls)
debut_ST =la liste des résultats du second tour
    concernant tous les candidats sauf le dernier  (regroupant : abstentions, blancs, nuls)
    Si on avait les résultats en nombre de voix, il faudrait 'normaliser' avec NEB
"""
debut_PT=[[1,5,7,6,3,4,2,8,9],[5,10,6,7,8,12,9,10,7],
                  [10,10,8,10,6,8,9,9,9],[30,35,40,25,16,27,13,40,19]]
debut_ST=[[40,30,45,55,41,53,35,38,40],[43,28,41,40,40,30,30,40,40]]

def completeLa100(t):   % complète une Ligne à 100
    return t+[100-sum(t)]
```

```
def completeLLa100(t):  % complète toutes les Lignes à 100
    return  [completeLa100(x) for x in t]

""" avec np.transpose ... les résultats sont faux (???) :
les array ne sont pas aux bons endroits       j'ai donc refabriqué tr.
 Après coup j'ai compris : mes PT ,et ST étaient des array :
     ils auraient dû  être de np.array, la transposée de M s'obient alors avec M.T     """

def tr(M):
    return [[M[k][m] for k in range(len(M))] for m in range(len(M[0]))]

trdPT = tr(debut_PT)
trdST = tr(debut_ST)

PT = completeLLa100(trdPT)
ST = completeLLa100(trdST)

MPT = np.array(PT)
tMPT = MPT.T
tPTPT = tMPT.dot(MPT)

tPTST = tMPT.dot(ST)
```

```
R = alg.solve(tPTPT,tPTST)
```

La matrice R a des termes < 0 ce qui surpend (au début)

```
""" on fabrique (artificiellement) un exemple qui va marcher :
  je pars de PT, j'en déduis une matrice de ST à  l'aide de R0 (une matrice de reports),
  puis je modifie les résultats du second tour : STmodifié et je re-caccule R """

A = MPT
R0 = np.array([[0.2,0.3,0.5],[0.1,0.7,0.2],[0.2,0.2,0.6],[0.4,0.3,0.2],[0.5,0.1,0.4]])
B = A.dot(R0)

""" pour modifier : mes lignes de Second tour :
le np.random.rand(nblignes,nbcolonnes) fournit une ligne à termes dans [0..1]
      j'en calcule la moyenne m
      np.ones(3) est [1,1,1], la - m*np.ones(3) est donc une liste de somme nulle,
je multiplie par 0.1 pour avoir des termes tous inférieurs à ceux de R0
     le .tolist() retransforme la matrice en liste """

def nprr3():
    la = np.random.rand(1,3)
    m = (la[0,1]+la[0,2]+la[0,0])/3.
    return (.1*(la - m*np.ones(3))[0]).tolist()

alea = np.array([nprr3(),nprr3(),nprr3(),nprr3(),nprr3(),nprr3(),nprr3(),nprr3(),nprr3()])
```

```
STmodifie = B + alea

nouvelR = alg.solve(tPPT,tMPT.dot(STmodifie))

"""   array([[ 0.20024016,  0.29835875,  0.50140109],
[ 0.10057014,  0.69658178,  0.20284807],
[ 0.19657986,  0.19505577,  0.60836437],
[ 0.40009179,  0.30162651,  0.19828169],
[ 0.50030442,  0.10088975,  0.39880583]]) """
```

C explication des reports négatifs

La théorie des pseudo solutions est de l'algèbre linéaire : le corps de base est $\mathbb{R}$ ... pas [0..1], la matrice R obtenue a ses termes de somme 1 (ça tombe bien pour l'interprétation pourcentages !) mais ils peuvent bien valoir 1.6 ou -0.5

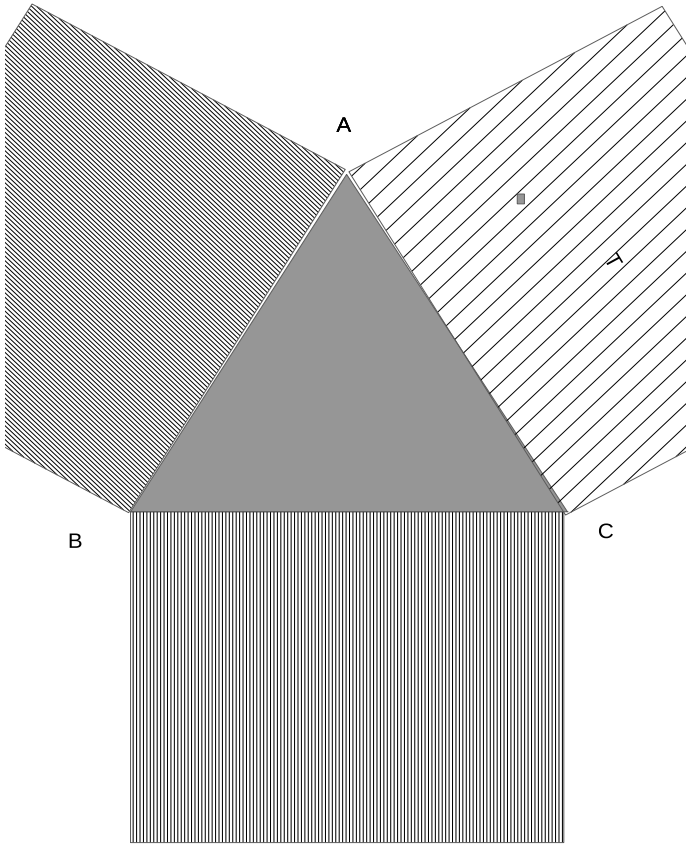
Si on se contente "des données raisonnables" R va être toujours bonne à voir, mais si on fait 'seulement des maths' il faut traiter tous les cas : quand des termes de R sont hors de [0..1] on va chercher la matrice à termes dans [0..1] la plus proche de R

D les dimensions 2 et 3

1 la dimension 2

Le problème est de dimension 2 quand on a 3 candidats : ST1, ST2 et A, chacune des lignes de R

est de somme 1, on aimerait que ses termes soient positifs. Les lignes positives de longueur 3 constituent un triangle équilatéral (ça se voit bien si on y pense en dimension 3 : les x,y,z >= 0 de somme 1 forment le triangle de coins [1,0,0], [0,1,0], [0,0,1]) c'est T_3 ci dessous. Pour chacune des lignes de R (c'est un point P du plan x+y+z=1 qui contient T_3) on veut le point de T_3 le plus proche de P



on voit bien qu'il y a 7 zones :

1. Le triangle ABC lui même : chaque point se projette en lui même (reports tous > 0)
2. en dessous de BC : chaque point se projette sur le côté BC
3. à droite de AC : de même

4. à gauche de AB : de même
5. au dessus de A : c'est le point A qui est le plus proche
6. du côté de B : de même
7. du côté de C : de même

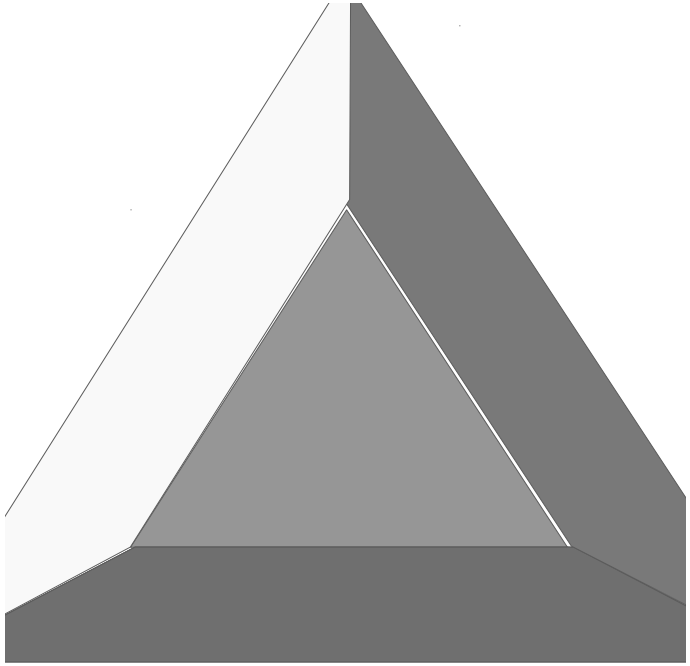
la traduction numérique est facile dans les cas 1,5,6,7 :
[0.2,0.3,0.5] va sur lui même
[1.3,-0.1,-0.2] va sur [1,0,0] (et deux cas analogues)
mais où va aller P_2 = [-0.2,0.5,0.7]? ... on a dit qu'on allait le projeter ... c'est bien cela qu'il faut faire, on va se déplacer perpendiculairement à AC = [0,1,-1] et à [1,1,1] la normale au plan, donc en direction [2,-1,-1], on va donc en [-0.2+2m,0.5-m,0.7-m] (pour être sur AC m=0.1) et on est arrivé en [0,0.4,0.6]

2 la dimension 3

Le problème est de dimension 3 quand on a 4 candidats : ST1, ST2 , ST3 et A, chacune des lignes de R est de somme 1, on aimerait que ses termes soient positifs. Les lignes positives de longueur 4 constituent un tétraèdre régulier (ça se voit bien si on y pense en dimension 4 : les x,y,z,t >= 0 de somme 1 forment le tétraèdre de coins [1,0,0,0], [0,1,0,0], [0,0,1,0] et [0,0,0,1]) c'est T_4 ci dessous. Pour chacune des lignes de R (c'est un point P de l'espace x+y+z+t=1 qui contient T_4) on veut le point de T_4 le plus proche de P

... et le choses NE se passent PAS comme en dimension inférieure (du moins quand j'ai essayé ça n'a pas marché)

En fait le dessin ci-dessus avec les hachures est ... vrai mais psychologiquement faux !!!



avec cette seconde figure l'extérieur du triangle est coupé en trois morceaux qui vont aboutir sur les trois côtés

a premier exemple

on prouvera dans le cadre de la dimension n ce qui se devine bien sur le dernier dessin : tout point P de l'espace x+y+z+t=1 est à coordonnées positives sur 3 (seulement 3) des vecteurs allant du centre G de T_4 vers ses sommets. Quand la somme des composantes est inférieure à 1 on est dans T_4, quand elle est strictement supérieure à 1 ... on va chercher le point le plus proche

Dans ce premier exemple je prend le tétraèdre A,B,C, D de centre G, vu comme :

$$A = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \; B = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \; C = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} \; D = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} \; G = \frac{1}{4} \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

Ainsi le plan ABC (hors de la somme 1) est z=0, l'arête AC a de plus x=0, les vecteurs

$$\overrightarrow{GA} = \frac{1}{4} \begin{pmatrix} -1 \\ -1 \\ -1 \\ 3 \end{pmatrix} \; \overrightarrow{GB} = \frac{1}{4} \begin{pmatrix} 3 \\ -1 \\ -1 \\ -1 \end{pmatrix} \; \overrightarrow{GC} = \frac{1}{4} \begin{pmatrix} -1 \\ 3 \\ -1 \\ -1 \end{pmatrix}$$

et comme exemple de point hors de T_4 je prend $M=G+0.2\overrightarrow{GA}+0.2\overrightarrow{GB}+0.8\overrightarrow{GC}$ (on a bien des coordonnées positives de somme > 1). Pour placer M par rapport à T_4 je cherche M_0 intersection du segment [GM] et de la face ABC :

$$M = \frac{1}{4} \begin{pmatrix} 0.6 \\ 3 \\ -0.2 \\ 0.6 \end{pmatrix} \; M_0 = M + \lambda \overrightarrow{GM} = \frac{1}{4} \begin{pmatrix} 0.6 \\ 3 \\ -0.2 \\ 0.6 \end{pmatrix} + \lambda \begin{pmatrix} -0.4 \\ 2 \\ -1.2 \\ -0.4 \end{pmatrix}$$

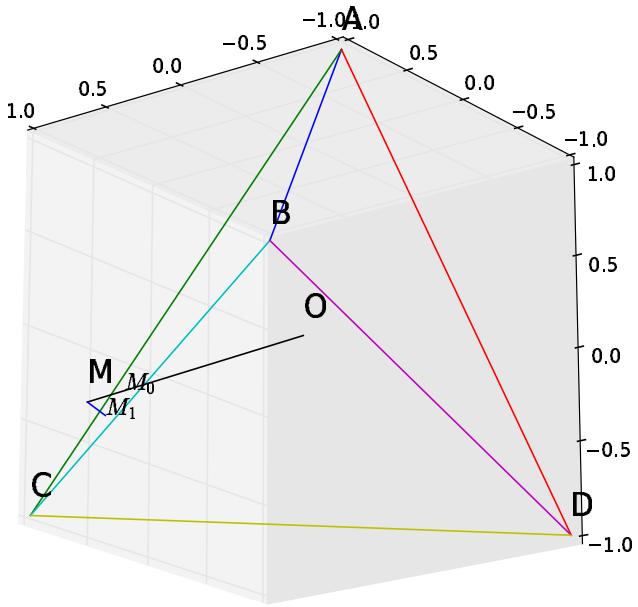
la face ABC étant z=0, je veux donc ... $\lambda = -\frac{1}{6}$ et finalement $M_0 = \frac{1}{6} \begin{pmatrix} 1 \\ 4 \\ 0 \\ 1 \end{pmatrix}$

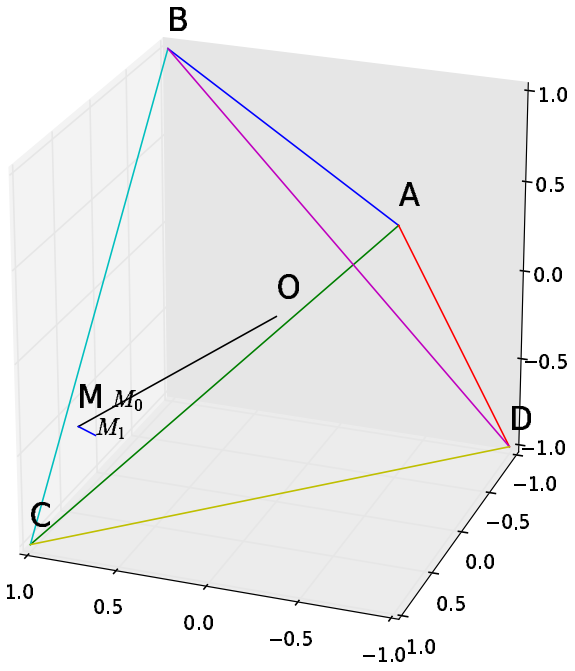
on veut maintenant projeter M sur [ABC] : M c'est M_0+$\overrightarrow{M_0M}$ qui va se projeter en M_0+$\text{proj}_{\text{face}}(\overrightarrow{M_0M})$,

on connait $\overrightarrow{F}$ normal à la face : $\begin{pmatrix} -1 \\ -1 \\ 3 \\ -1 \end{pmatrix}$ et on veut projeter $\overrightarrow{\nu} = -\frac{1}{6}\begin{pmatrix} -0.1 \\ 0.5 \\ -0.3 \\ -0.1 \end{pmatrix}$

On obtient donc $M_0 - [\overrightarrow{\nu} - \frac{<\overrightarrow{\nu}|F>F}{<F|F>}] = \frac{1}{15}\begin{pmatrix} 2 \\ 11 \\ 0 \\ 2 \end{pmatrix}$

Les deux images qui suivent montrent tout cela sous deux angles


A
B
O
M
M_0
M_1
C
D
1.0
0.5
0.0
−0.5
−1.0



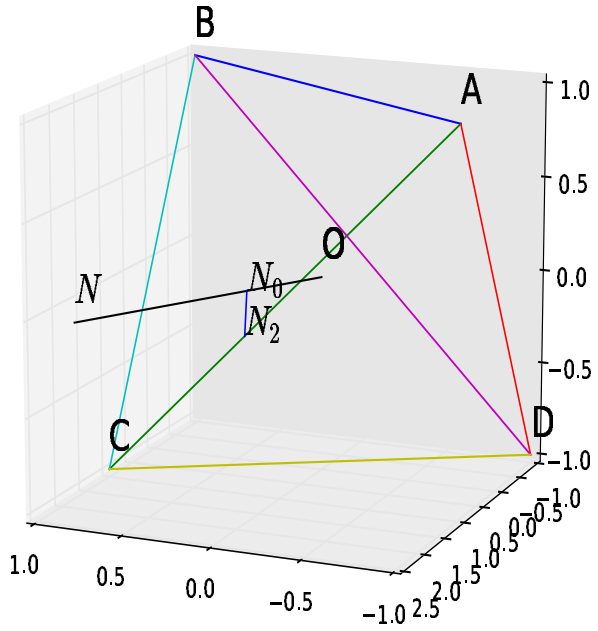
vous imaginez aisément le nombre d'erreurs de calcul à faire dans ce qui précède (+ sa traduction en images de dimension 3)

l'image a un avantage : on y voit que mon exemple était mauvais je suis parti d'un point symétrique en A et B (j'étais parti avec 0.2 et 0.2)

b deuxième exemple

Pour ce second exemple (même cadre général), je vais chercher un N (hors de T_4) qui ne se projette pas dans la triangle [ABC] : je prend N=G+1.2$\overrightarrow{GA}$+0.4$\overrightarrow{GB}$+1.6$\overrightarrow{GC}$ (on a bien des coordonnées positives

de somme > 1). Pour placer N par rapport à T_4 je cherche N_0 intersection du segment [GN] et de la face ABC : $N_0 = N + \theta\overrightarrow{GN}$ doit avoir son z nul : d'où $\theta = -11/16$, on projette alors $\overrightarrow{N_0N}$ sur la face ABC on obtient $N_1 = \frac{1}{24}\begin{pmatrix} -8 \\ 20.8 \\ 0 \\ 11.2 \end{pmatrix}$ il est trop loin (pas dans le triangle) alors je cherche $N_2 = N_0 + \omega\overrightarrow{N_0N_1}$ sur l'arête AC, d'où ω=1/44. Il reste encore à voir si ce $\overrightarrow{NN_2}$ est orthogonal à AC : chic c'est le cas, on n'a donc pas besoin de se déplacer sur AC pour chercher le minimum



c troisième exemple

Pour ce troisième exemple (même cadre général), je vais chercher un P (hors de T_4) qui ne se projette pas dans le triangle [ABC] et qui ne se projette pas non plus dans le côté AC. Cet ensemble de calculs étant vraiment trop pénible, je triche : P=G + $2\overrightarrow{GC}$ ... miracle c'est C le point le plus proche

3 résumé de la dimension 3

par la méthode des pseudos inverses on a trouvé une matrice R la plus proche d'une solution au problème PT.R = ST, on sait que les termes de R sont de sommes 1 par ligne

1. si tous les termes de R sont positifs : c'est fini
   sinon : si un ou des termes de R sont < 0 : la structure euclidienne nous dit (Pythagore) que les minimisations de distances des lignes sont des problèmes indépedants, pour chaque telle ligne il s'agit de trouver lr point de T_4 le plus proche

2. on verra dans le § suivant comment ramener tous les cas à des composantes seulement ≥ 0

3. On a vu (exemple 1) comment calculer le point le plus proche quand R se projette sur une face ([ABC] convexe)

4. On a vu (exemple 2) comment calculer le point le plus proche quand R se projette hors d'une face [ABC] mais va sur une arête

5. On a mal vu (exemple 3) comment calculer le point le plus proche quand R se projette hors d'une face [ABC] et hors des arêtes mais sur un coin
   on procède par projections comme dans les exemples 1 et 2, quand on constate est hors du [ABC] on prolonge la projection jusqu'à un des côtés, sur ce côté on se déplace jusqu'à la projection est si on est hors du segment on va aucoin le plus proche

E divers problèmes

1 que signifie “le rang de A est q”

cela signifie que les résultats des différents bureaux de voté sont ... différents! Si on avait deux bureaux donnant exactement les mêmes résultats on les regroupe en un seul. De même si un bureau donne une combinaison linéaire des autres

2 comment se ramener aux termes tous > 0 en nombre au plus n (dimension n)?

Les vecteurs qui joignent le centre G du simplexe régulier aux sommets A_k sont au nombre de n+1, leur somme est nulle.

On part d'un point M de l'espace : comme les $\overrightarrow{GA_k}$ sont une famille génératrice $\overrightarrow{GM}$ est combinaison linéaire des $\overrightarrow{GA_k}$:

$$\overrightarrow{GM} = \sum_k \mu_k \overrightarrow{GA_k}$$

je choisis un μ_m minimum, et je remplace $\mu_m \overrightarrow{GA_{\mu_m}}$ par $\Sigma_{k \neq m} -\mu_m \overrightarrow{GA_k}$: il me reste une somme de n vecteurs à coefficients positifs

3 comment traiter la dimension n ?

comme dit ci-dessus : on se ramène a une combinaison positive de n des $\overrightarrow{GA_k}$, et on veut passer du cône (infini) à la face du simplexe ; pour cela on joint à G puis on projette à répétition “en tronquant quand ça dépasse”, comme la dimension diminue strictement à chaque étape, à la fin cela finit par finir

comment être sûr qu'il suffit de se limiter à la face ? : tout point du simplexe $A_1...A_{n+1}$ joint à un point extérieur dans la direction de la face $A_1...A_n$ va constituer un segment qui va rencontrer $[A_1...A_n]$ le minimum de distance pour cette face est donc bien le minimum général

je suis bien conscient que ce qui précède devrait s'accompagner de preuves sérieuses

4 pourquoi la distance euclidienne?

elle seule permet d'avoir des arguments "géométriques" : pseudo-inverse, projections, Pythagore ...

Il y a une amélioration si on veut mélanger les bureaux de Vote : au lieu de les compter comme égaux, on peut tenir compte du Nombre d'Electeurs par Bureau (d'électeurs ?ou d'inscrits (?) ou d'exprimés ou ...) cela change juste un peu les calculs : la matrice (NEB diagonale) des effectifs des bureaux est insérée entre ${}^{t}M$ et M en ${}^{t}M.NEB.M$

Déjà que l'on ne savait pas s'il fallait les inscrits ou les exprimés il faudrait peut être l'élever à une certaine puissance cette NEB

On peut aussi enviseager la $||\quad||_1$? en ce cas il semble avisé (?) de partie de la R obtenue ci-dessus pour servir de germe au processus de minimisation