


TD avec le logiciel 

## Théorie des Jeux

### TP : Le musth chez les éléphants

Christelle LOPES, *christelle.lopes@univ-lyon1.fr*

Printemps 2020

## 1 Introduction

Le musth est un état qui revient périodiquement chez les éléphant mâles. Il est caractérisé par des comportements extrêmement agressifs et une épaisse sécrétion ressemblant à du goudron : la frontaline, qui sort des orifices temporaux. Cette sécrétion peut être considérée comme un signal d'une stratégie agressive (telle la stratégie "Faucon" dans le jeu "Faucon-Colombe"). Comment expliquer que les mâles en état de musth "perdent" du temps et de l'énergie à annoncer leur stratégie ? Une telle annonce pourrait éviter les affrontements dans la mesure où les autres ont confiance en ce signal. Mais alors pourquoi les tricheurs (qui utiliseraient le signal sans être particulièrement agressifs) n'envahissent pas la population ? Pour répondre à cette question, on considère le jeu suivant à trois stratégies:

- **F** : Faucon annonceur. L'individu choisit toujours la stratégie Faucon et l'annonce.
- **C** : Colombe. L'individu choisit toujours la stratégie Colombe.
- **T** : Tricheur. L'individu annonce qu'il choisit la stratégie Faucon, mais cherche à s'enfuir si l'autre joueur montre également un comportement agressif.

On suppose qu'un Tricheur est capable d'effrayer une Colombe, et que deux Tricheurs ont la même chance de gagner le combat. On peut émettre deux hypothèses quant à l'issue du combat entre un Tricheur et un Faucon :

- Hypothèse 1 : les Faucons ne sont pas capables de rattraper les Tricheurs lors de leur fuite et gagnent donc sans combattre.
- Hypothèse 2 : les Faucons sont capables de rattraper les Tricheurs lors de leur fuite, et les Faucons gagnent toujours facilement, les Tricheurs sortant blessés du combat.

On appelle  $G$  le gain correspondant à ce qui est gagné lors d'un combat et  $C$  le coût correspondant aux pertes dues aux blessures lors des combats.

Nous allons, grâce aux simulations numériques du système, étudier l'issue de la compétition pour les deux hypothèses et différentes valeurs de  $G$  et  $C$ .

## 2 Le modèle

On appelle  $x$  la proportion d'individus Faucon dans la population,  $y$  la proportion d'individus Colombe et  $z$  la proportion d'individus Tricheur.

Soit  $A_1$  la matrice des gains relative à l'hypothèse 1 du modèle et  $A_2$  la matrice des gains relative à l'hypothèse 2 :

$$A_1 = \begin{pmatrix} \frac{G-C}{2} & G & G \\ 0 & \frac{G}{2} & 0 \\ 0 & G & \frac{G}{2} \end{pmatrix} \quad A_2 = \begin{pmatrix} \frac{G-C}{2} & G & G \\ 0 & \frac{G}{2} & 0 \\ -C & G & \frac{G}{2} \end{pmatrix}$$

Les équations du réplicateur sont alors :

Cas 1 :

$$\begin{cases} \dot{x} = x(\frac{G-C}{2}x + Gy + Gz - \frac{G}{2} + \frac{C}{2}x^2) \\ \dot{y} = y(\frac{G}{2}y - \frac{G}{2} + \frac{C}{2}x^2) \\ \dot{z} = z(Gy + \frac{G}{2}z - \frac{G}{2} + \frac{C}{2}x^2) \end{cases} \quad (1)$$

Cas 2 :

$$\begin{cases} \dot{x} = x(\frac{G-C}{2}x + Gy + Gz - \frac{G}{2} + \frac{C}{2}x^2 + Cxz) \\ \dot{y} = y(\frac{G}{2}y - \frac{G}{2} + \frac{C}{2}x^2 + Cxz) \\ \dot{z} = z(-Cx + Gy + \frac{G}{2}z - \frac{G}{2} + \frac{C}{2}x^2 + Cxz) \end{cases} \quad (2)$$

Ce qui donne, en dimension 2 :

Cas 1 :

$$\begin{cases} \dot{x} = \frac{1}{2}x(Cx^2 - (G+C)x + G) \\ \dot{y} = \frac{1}{2}y(Gy - G + Cx^2) \end{cases} \quad (3)$$

Cas 2 :

$$\begin{cases} \dot{x} = x(-\frac{C}{2}x^2 + \frac{C-G}{2}x - Cxy + \frac{G}{2}) \\ \dot{y} = y(-\frac{C}{2}x^2 + Cx + \frac{G}{2}y - Cxy - \frac{G}{2}) \end{cases} \quad (4)$$

### 3 Implémentation et simulation du modèle

La simulation numérique du modèle nécessitera la librairie **deSolve**. La représentation des trajectoires dans le triangle unitaire dont les sommets sont les stratégies pures nécessitera d'utiliser la librairie **robCompositions**, et plus particulièrement les fonctions **ternaryDiag()** et **ternaryDiagPoints()**. Selon votre version de  $\mathbb{R}$ , l'installation de la librairie **robCompositions** peut s'avérer délicate car elle nécessite d'installer d'autres librairies en amont, et dont une pour laquelle il faut aller "chercher" une ancienne version (pour des questions de compatibilité de version avec  $\mathbb{R}$ ). Pour se faire, les étapes à suivre sont les suivantes:

```
> install.packages("robCompositions")
> #Si vous obtenez un message d'erreur, la procedure est la suivante:
> install.packages("lme4")
> install.packages("devtools")
> library(devtools)
> install_version("pbkrtest",version="0.4-4",repos="http://cran.us.r-project.org")
> install.packages("robCompositions")
```

La représentation triangulaire des données peut également se faire avec la librairie **ade4** et la fonction **triangle.plot**. L'inconvénient de cette fonction est qu'on ne peut pas représenter différentes trajectoires (pour différentes conditions initiales) sur un même portrait de phase. Vous trouverez une fiche utile sur la représentation triangulaire à l'adresse suivante: <https://pbil.univ-lyon1.fr/R/pdf/ter1.pdf>.

L'idée ici est de simuler le modèle à partir de la matrice (dont les coefficients dépendent de  $G$  et  $C$ ) et de faire calculer à  $\mathbb{R}$  les équations du réplicateur en fonction des valeurs de  $G$  et  $C$  données en entrée. Ainsi, vous pourrez non seulement simuler très facilement l'issue du jeu pour différentes valeurs de  $G$  et  $C$ , et ré-utiliser votre code pour simuler n'importe quel jeu à 3 stratégies en ne modifiant que la matrice des gains.

1. Créez une fonction **defmat1()**, respectivement **defmat2()**, ayant comme arguments  $G$  et  $C$  et retournant la matrice de gains  $A_1$  pour le cas 1 et  $A_2$  pour le cas 2, respectivement. Vérifiez que les matrices retournées pour  $G = 2$  et  $C = 4$  sont bien:

```
> A1=defmat1(G=2,C=4)
> A1
```

```
      [,1] [,2] [,3]
[1,]    -1     2     2
[2,]     0     1     0
[3,]     0     2     1
```

```
> A2=defmat2(G=2,C=4)
> A2
```

```
      [,1] [,2] [,3]
[1,]    -1     2     2
[2,]     0     1     0
[3,]    -4     2     1
```

2. Créez une fonction **syst3D1** pour le cas 1, et une fonction **syst3D2** pour le cas 2, prenant comme arguments  $t$  (le temps de simulation),  $y$  le vecteur de variables étudiées et  $parms$  le vecteur de paramètres, et retournant les équations du réplicateurs sous forme de liste. C'est cette fonction qui sera utilisée dans **lsoda** pour faire l'intégration numérique. Notez que le produit matriciel dans  $\mathbb{R}$  se fait avec l'opérateur `%*%`. Vérifiez que pour  $G = 2$  et  $C = 4$ , avec comme condition initiale  $x(0) = 0.1$ ,  $y(0) = 0.6$  et  $z = 0.3$ , vous obtenez bien les prédictions suivantes:

```

> temps=seq(from=0,to=100,by=0.01)
> init1=c(0.1,0.6,0.3)
> #Cas1:
> simul1=lsoda(y=init1,times=temps,func=syst3D1,parms=c(G=2,C=4))
> head(simul1)

```

```

      time      1      2      3
[1,] 0.00 0.1000000 0.6000000 0.3000000
[2,] 0.01 0.1007217 0.5977184 0.3015600
[3,] 0.02 0.1014466 0.5954336 0.3031198
[4,] 0.03 0.1021749 0.5931457 0.3046794
[5,] 0.04 0.1029064 0.5908548 0.3062388
[6,] 0.05 0.1036412 0.5885611 0.3077977

```

```

> #Cas 2:
> simul2=lsoda(y=init1,times=temps,func=syst3D2,parms=c(G=2,C=4))
> head(simul2)

```

```

      time      1      2      3
[1,] 0.00 0.1000000 0.6000000 0.3000000
[2,] 0.01 0.1008431 0.5984423 0.3007146
[3,] 0.02 0.1016924 0.5968891 0.3014185
[4,] 0.03 0.1025479 0.5953406 0.3021115
[5,] 0.04 0.1034097 0.5937969 0.3027934
[6,] 0.05 0.1042778 0.5922580 0.3034642

```

Le portrait de phase représentant les trajectoires du cas 1 pour  $G = 2$  et  $C = 4$ , avec 3 conditions initiales différentes, est présenté sur la Figure 1 :

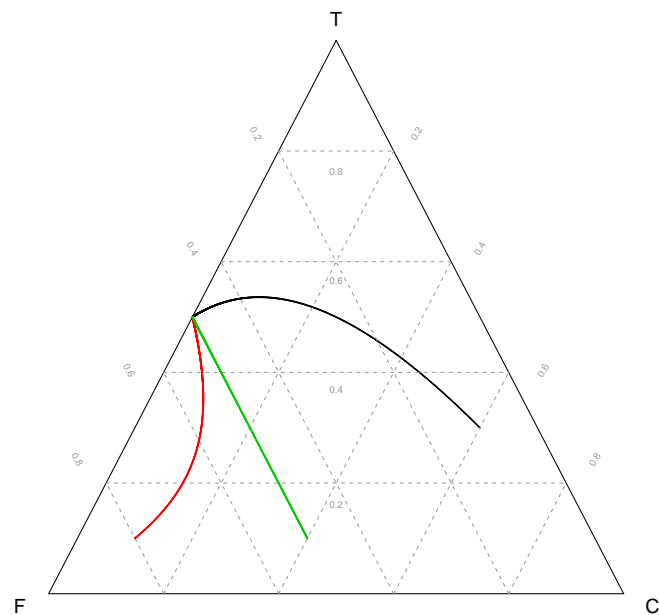


Figure 1: Portraits de phase et chroniques du modèle (3) pour  $G = 2$  et  $C = 4$  et différentes conditions initiales.

3. Retrouvez les conditions initiales correspondant à chaque trajectoire.

4. Reproduisez-le en utilisant la librairie `robCompositions` et les fonctions associées `ternaryDiag()` et `ternaryDiagPoints()` (ou la librairie `ade4` et la fonction `triangle.plot`).
5. Que se passe-t-il si vous partez des conditions initiales suivantes:
  - $x(0) = 1; y(0) = z(0) = 0$ .
  - $x(0) = z(0) = 0; y(0) = 1$ .
  - $x(0) = y(0) = 0; z(0) = 1$ .
  - $x(0) = y(0) = 0.5; z(0) = 0$ .
  - $x(0) = z(0) = 0.5; y(0) = 0$ .
6. Tracez les chroniques associées aux trajectoires représentées sur la Figure 1. Discutez de l'issue du jeu selon l'hypothèse 1.
7. Que se passe-t-il si  $G = 4$  et  $C = 2$  ?
8. Etudiez l'issue du jeu pour l'hypothèse 2. Discutez.

## 4 Ajout de stochasticité

Selon l'hypothèse 1, les Tricheurs peuvent se maintenir dans la population au détriment des Colombes qui disparaissent. Il peut alors être "tentant", pour les Colombes, de tricher. On va donc considérer un nouveau jeu où une proportion  $p$  des Colombes trichent et adoptent un comportement de Tricheur, et une proportion  $(1 - p)$  restent des Colombes. On appellera  $K$  cette nouvelle stratégie.

9. Ecrivez la matrice de gains pour ce nouveau jeu et implémentez-la dans `R` sur la base de la fonction `defmat1` définie plus haut. Vérifiez que la matrice retournée pour  $G = 2$ ,  $C = 4$  et  $p = 0.4$  est bien:

```
> A=defmat(G=2,C=4,p=0.4)
> A

      [,1] [,2] [,3]
[1,]   -1  2.0  2.0
[2,]    0  1.0  0.4
[3,]    0  1.6  1.0
```

10. Simulez le modèle pour différentes valeurs de  $p$ . Discutez.

On suppose maintenant que le fait de Tricher confère un avantage :  $a(K, K) = \frac{G}{2} \times \alpha$ , avec  $1 \leq \alpha \leq 2$ .

11. On se place d'abord dans le cas où  $\alpha = 1.6$ . Simulez le modèle avec la nouvelle matrice de gains et discutez des prédictions pour  $p < 0.6$ ,  $p > 0.6$  et  $p = 0.6$ , et les conditions initiales suivantes:
  - $CI1$  :  $x(0) = 0.8$ ,  $y(0) = 0.1$  et  $z(0) = 0.1$ .
  - $CI2$  :  $x(0) = 0.1$ ,  $y(0) = 0.8$  et  $z(0) = 0.1$ .
  - $CI3$  :  $x(0) = 0.1$ ,  $y(0) = 0.1$  et  $z(0) = 0.8$ .
12. On se place maintenant dans le cas général où  $0 \leq p \leq 1$  et  $1 \leq \alpha \leq 2$ . Simulez le modèle pour les différentes combinaisons de valeurs de  $p$  et  $\alpha$ , et les différentes conditions initiales précédentes. Pour synthétiser vos résultats, créez, pour chaque condition initiale, une matrice contenant les valeurs de  $x$  quand  $t$  tend vers l'infini pour chaque couple de valeurs  $(p, \alpha)$ , une matrice contenant les valeurs de  $y$  quand  $t$  tend vers l'infini et une matrice contenant les valeurs de  $z$  quand  $t$  tend vers l'infini. Vous pourrez ensuite représenter vos résultats en 3 dimensions en utilisant le package `plot3D` et les fonctions `image2D`, `ribbon3D` ou `persp3D`. La Figure 2 synthétise les issues du jeu possibles pour les différentes combinaisons de valeurs de  $p$  et  $\alpha$ .

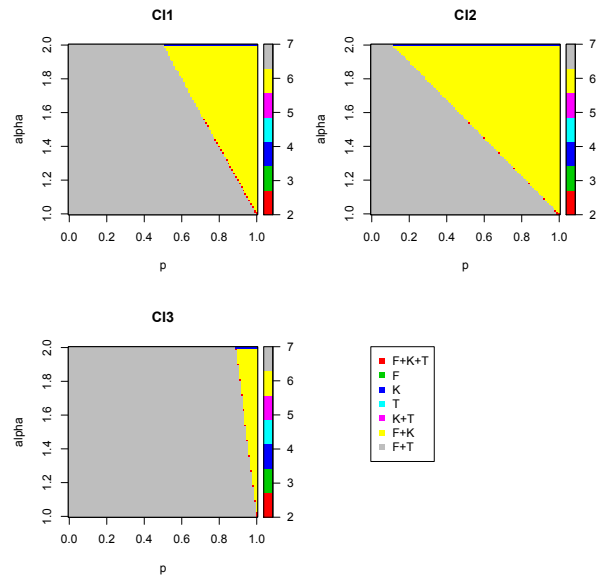


Figure 2: Issues du jeu pour les 3 conditions initiales différentes selon les valeurs de  $p$  et  $\alpha$