Projet TP : BATAILLE !

DUT - INFO
2A – G5

PERALDE François

Année : 2020 / 2021

Description du projet

La bataille est l'un des jeux de carte les plus connu. En préambule, nous mélangeons les cartes et distribuons le paquet de 52 cartes, équitablement et face cachée, entre deux joueurs. A chaque tour, les joueurs retournent la carte située au-dessus de leur paquet. Si l'une des cartes est plus forte que l'autre, le joueur gagnant empoche les deux cartes, qu'il place à la fin de son paquet. Si les deux cartes sont de la même force, il y a bataille. Ainsi, chaque joueur met en jeu la carte suivante de son paquet, sans la retourner, puis retourne la carte encore suivante. Le gagnant empoche dont les 6 cartes mises en jeu. La partie s'arrête lorsqu'un joueur a récupéré la totalité des cartes.

Dans ce projet, nous devrons effectuer diverses probabilités et statistiques. Pour cette raison, nous allons créer 2 modes de jeu :

- Un mode simple, suivant les explications ci-dessus.
- Un mode test, avec des variantes du jeu comme le fait de ne pas distribuer toutes les cartes ou bien de ne pas distribuer le même nombre de cartes entre les joueurs.

Notons que nous attribuons un numéro à chaque carte, suivant cette idée

Dans un jeu normal	2	3	4	5	6	7	8	9	10	Vallet	Dame	Roi	As
Dans ce projet	1	2	3	4	5	6	7	8	9	10	11	12	13

Et que nous ne portons pas d'importance aux couleurs.

/ !\ Notez, que les fonctions entièrement commentées sont disponibles en annexe ou dans le code directement. Dans le reste de ce document, il est simplement question de descriptions / explications.

1 - Fonctionnalités à réaliser

▪ Fonction de distribution

Explication :

Ma fonction de distribution ne prend pas de paramètres, mais retourne 2 listes de cartes. Une pour le premier joueur et l'autre pour le deuxième joueur. Le déroulement est simple. Je pars d'une liste de 56 valeurs représentant les 56 cartes. Par la suite, pour chaque jeu, je crée une boucle qui tant que je n'ai pas tiré 26 cartes, je prends un nombre aléatoire entre 1 et 52, et s'il est encore possible de le récupérer dans le paquet de base, je le mets dans le jeu du joueur sinon je retire un nombre au hasard et répète la même opération. Ensuite, je mets à jour le fait de ne plus pouvoir récupérer la carte dans le paquet de base (remplace la valeur de la carte par 0) et je finis ensuite par incrémenter le nombre de cartes dans le paquet du joueur à + 1.

Ici, nous pouvons parler de variable aléatoire, étant donné que dans la boucle while, chaque carte prend une valeur différente à chaque répétition. Ω = {1,2,3,4,5,6,7,8,9,10,11,12,13}. Où chaque carte a la probabilité de 1/13 d'être tirée.

Comme vous pouvez le voir ci-dessous, le code permettant de remplir les deux paquets de cartes.

```
function [jeu1, jeu2]=distribue()
   paquetdecarte = [1 1 1 1 2 2 2 2 3 3 3 3 4 4 4 4 5 5 5 5 6 6 6 6 7 7 7 7 8 8 8 8 9 9 9 9 10 10 10 10 11 11 11 11
12 12 12 12 13 13 13 13];
   jeu1 = 1:26;
   jeu2 = 1:26;
   k=1
   i=1
   while k ~=27
      res = grand(1,1,"uin",1,52);
      if paquetdecarte(1, res) ~= 0 then
         jeu1(1,k) = paquetdecarte(1, res);
         paquetdecarte(1, res) = 0;
         k= k+1;
      end
   end
   j = 1;
   while j ~=27
      res = grand(1,1,"uin",1,52);
      if paquetdecarte(1, res) ~= 0 then
         jeu2(1,j) = paquetdecarte(1, res);
         paquetdecarte(1, res) = 0;
         j = j + 1;
      end
   end
endfunction
```

▪ Fonction de distribution alternative – pas le même nombre de carte

Explication :

Cette fonction de distribution alternative retourne également les deux jeux de cartes, simplement cette fonction appelle la fonction distribution de bases qui retourne les 2 jeux de 26 cartes. Je passe en paramètre le nombre de cartes pour les deux joueurs. Ainsi, si je décide de donner 30 cartes au joueur 1, il lui manque alors 4 cartes (pour passer de 36 à 30). Pour ce faire, je prends les 4 premières cartes du joueur 2 pour les passer ensuite au joueur 1. Pour finir, j'obtiens mes 2 jeux, que je retourne.

```
function [jeu1, jeu2]=distribuepaslesmemes(nbcartes1, nbcartes2)
    jeu1 = [];
    jeu2 = [];
    recup = [];
    nbcartesenplus = 0;
    [jeu1,jeu2]=distribue();
    //cas ou le joueur 1 a plus de cartes
    if nbcartes1 > nbcartes2 then
        nbcartesenplus = 26 - nbcartes2;
        nbcartesenplus = int(nbcartesenplus);
        for i = 1:nbcartesenplus
            recup(1) = int(jeu2(1));
            jeu1(1,$+1) = recup(1);
            jeu2(1) = [];
        end
    end
    //cas ou le joueur 2 a plus de cartes
    if nbcartes1 < nbcartes2 then
        nbcartesenplus = 26 - nbcartes1;
        nbcartesenplus = int(nbcartesenplus);
        for i = 1:nbcartesenplus
            recup(1) = int(jeu1(1));
            jeu2(1,$+1) = recup(1);
            jeu1(1) = [];
        end
    end
endfunction
```

▪ Fonction de distribution alternative – pas toute les cartes

Explication :

Avant toute chose, cette fonction est assez différente des précédentes. Je lui passe en paramètre un paquet de carte et le nombre de carte (ici, les deux joueurs ont le même nombre de cartes). Pour définir les cartes à passer en paramètre, je demande pour chaque carte, si l'utilisateur souhaite ou non ajouter ces cartes dans le jeu. Ainsi, à chaque demande si le joueur dit oui, ou non, j'ajoute ou non les cartes dans le paquet. Pour avoir le nombre de carte pour les joueurs, je prends la taille de la matrice et la divise par 2.

Boucle pour demandes les cartes :

```
paquet = [];
        nbtt = 0;
        for i = 1:13
            printf('Voules-vous les %d ? 1 = oui | 0 = non ', i+1);
            resultat = input('');
            while resultat ~= 1 && resultat ~= 0
                printf('Voules-vous les %d ? 1 = oui | 0 = non ', i+1);
                resultat = input('');
            end
            if resultat == 1 then
                for j = 1:4
                    paquet(1,$+1) = i;
                    nbtt=nbtt+1;
                end
            end
        end
```

Fonction :

```
function [jeu1, jeu2]=distribuepastout(paquetdecarte, nbcartes)
    jeu1 = 1:nbcartes;
    jeu2 = 1:nbcartes;
    k=1
    i=1
    z = nbcartes+1;
    u = nbcartes * 2;
    while k ~= z
        res = grand(1,1,"uin",1,u);
        if paquetdecarte(1, res) ~= 0 then
            jeu1(1,k) = paquetdecarte(1, res);
            paquetdecarte(1, res) = 0;
            k= k+1;
        end
    end
    j = 1;
    while j ~= z
        res = grand(1,1,"uin",1,u);
        if paquetdecarte(1, res) ~= 0 then
            jeu2(1,j) = paquetdecarte(1, res);
            paquetdecarte(1, res) = 0;
            j = j + 1;
        end
    end
endfunction
```

▪ Fonction de la bataille

Explication :

Pour la fonction bataille, nous passons en paramètre le jeu du joueur 1 et le jeu du joueur 2. Nous souhaitons retourner 2 valeurs : le gagnant et le nombre de tours qu'a duré la partie. Pour cette fonction, nous avons une grande boucle while qui vérifie que l'un des deux jeux n'est pas vide. Dans cette boucle, nous retrouvons 3 conditions :

- Si la carte du joueur 1 est supérieur à la carte du joueur 2

Alors jeu1 prend les cartes et les mets à la fin de son jeu (ordre : cartejoueur1 puis cartejoueur2).
Nous supprimons ensuite la première carte de chaque paquet.
Nous incrémentons le nombre de tours de 1.

```
if(jeu1(1)>jeu2(1)) then
      //ajout des cartes dans le jeu1
      jeu1=[jeu1,jeu1(1),jeu2(1)];
      TailleM=size(jeu1);
      //suppression des premières cartes dans les 2 paquets
      jeu1=suppJeu(jeu1,TailleM(1,2));
      TailleM=size(jeu2);
      jeu2=suppJeu(jeu2,TailleM(1,2));
      //incrémentation du temps
      temps = temps +1;
   end;
```

Etant donné que nous appelons la fonction suppJeu, je me permets de vous l'expliquer ici directement. Cette fonction prend en paramètre un jeu et une taille et va simplement

```
function L=suppJeu(A, taille)
   L=[];
   t=taille-1
   for i= 1:t
      L=[L,A(i+1)];
   end
endfunction
```

- Si la carte du joueur 2 est supérieur à la carte du joueur 1

Alors jeu2 prend les cartes et les mets à la fin de son jeu (ordre : cartejoueur2 puis cartejoueur1).
Nous supprimons ensuite la première carte de chaque paquet.
Nous incrémentons le nombre de tours de 1.

```
if(jeu2(1)>jeu1(1)) then
      //ajout des cartes dans le jeu1
      jeu2=[jeu2,jeu2(1),jeu1(1)];
      TailleM=size(jeu1);
      //suppression des premières cartes dans les 2 paquets
      jeu1=suppJeu(jeu1,TailleM(1,2));
      TailleM=size(jeu2);
      jeu2=suppJeu(jeu2,TailleM(1,2));
      //incrémentation du temps
      temps = temps +1;
   end;
```

- Si les cartes sont égales

Nous entrons dans une boucle while qui tourne tant que les cartes sont égales.
Nous créons alors une matrice Bataille, qui va prendre les 2 premières cartes de chaque joueur dans l'ordre carte1 joueur1, carte1 joueur2, carte2 joueur1, carte2 joueur2. Nous supprimons également les 2 premières cartes des deux paquets. Et nous vérifions par la suite la 3ème carte de chaque jeu devenu alors première carte suite à la suppression. La carte la plus forte prend alors dans l'ordre : matrice bataille, sa carte puis la carte de l'adversaire.

Exemple :

Jeu1	5,8,6,1			6,1		1,5,5,8,13,6,2
Jeu2	5,13,2,6			2,6		6
Affrontement		5 vs 5			6 vs 2	
Matrice Bataille			5,5,8,13	5,5,8,13		

```
//cas où les cartes sont les mêmes
  if(jeu1(1)==jeu2(1))
    while(jeu1(1)==jeu2(1))
      //virification si un jeu est null
      if(jeu1 == [] || jeu2==[])
        then
        if(jeu1 == [])
        then gagnant = 1;
        return;
        else gagnant = 2;
        return;
        end;
      end;
      //mise à jour de Bataille et des jeux
      Bataille=[Bataille,jeu1(1),jeu2(1)]
      TailleM=size(jeu1);
      jeu1=suppJeu(jeu1,TailleM(1,2));
      TailleM=size(jeu2);
      jeu2=suppJeu(jeu2,TailleM(1,2));
      //virification si un jeu est null
      if(jeu1 == [] || jeu2==[])
        then
        if(jeu1 == [])
        then gagnant = 1;
        return;
        else gagnant = 2;
        return;
        end;
      end;
      ////mise à jour de Bataille et des jeux (2èmes cartes)
      Bataille=[Bataille,jeu1(1),jeu2(1)]
      TailleM=size(jeu1);
      jeu1=suppJeu(jeu1,TailleM(1,2));
      TailleM=size(jeu2);
      jeu2=suppJeu(jeu2,TailleM(1,2));
      //virification si un jeu est null
      if(jeu1 == [] || jeu2==[])
        then
        if(jeu1 == [])
        then gagnant = 1;
        return;
```

```
            else gagnant = 2;
            return;
            end;
        end;
    end;
    //vérification gagnant ??
    if(jeu1(1)>jeu2(1))
    then jeu1=[jeu1,Bataille,jeu1(1),jeu2(1)];
    TailleM=size(jeu1);
    jeu1=suppJeu(jeu1,TailleM(1,2));
    TailleM=size(jeu2);
    jeu2=suppJeu(jeu2,TailleM(1,2));
    temps = temps +1;
    end;
    if(jeu2(1)>jeu1(1))
    then jeu2=[jeu2,Bataille,jeu2(1),jeu1(1)];
    TailleM=size(jeu1);
    jeu1=suppJeu(jeu1,TailleM(1,2));
    TailleM=size(jeu2);
    jeu2=suppJeu(jeu2,TailleM(1,2));
    temps = temps +1;
  end;
  //remet bataille nulle
  Bataille=[];
  end;
end;
```

- ## Menu

Explication :

En plus de devoir faire un compte-rendu sur les statistiques, je voulais que n'importe qui puisse constater par lui-même ce que je vais avancer. Pour ce faire, j'ai voulu faire un menu permettant à n'importe qui de tester les différents modes, mais également d'accéder aux statistiques. Pour cela, un menu simple proposant 4 choix :

Quitter : qui arrête le programme
Mode normal : qui va appeler la fonction de distribution normale et également lancer la fonction de bataille. Sans oublier le fait que j'affiche le gagnant à la fin avec le nombre de tour.
Mode test : ce mode appelle un sous menu proposant le choix entre

- Nombre différent de cartes aux joueurs, qui va demander et vérifier le nombre de carte que l'on souhaite donner au joueur 1. Par la suite, il appelle les fonctions de distributions (alternative) et la bataille.
- Ne pas distribuer toutes les cartes, qui demande pour chaque carte si on souhaite l'avoir dans le paquet. Par la suite, appelle les fonctions de distribution (alternative) et la bataille.

Mode stats : ce mode amène également à un sous menu proposant différent choix :

- Stats sur la durée moyenne
- Stats sur une forme de force
- Stats sur l'influence de la remise des cartes dans le paquet

2 – Etude de proba et stats

Notons, que chaque étude s'est faite sur 100 parties et que je ne montre par le code de la fonction m'ayant permis de faire l'étude. Elle reste cependant disponible en annexe.

▪ Etude de la durée moyenne

Explication :
Ici, nous pouvons parler de variable aléatoire, étant donné que le gagnant change à chaque répétition. Ω est fini car à chaque nouvelle partie Ω = {Gagné, Perdu}.

La question de la moyenne de tours dans une partie est un peu basique. Mais je voulais commencer par cette étude. Savoir si le fait qu'un joueur ait plus ou moins de cartes, influençait sur le nombre de tours moyens. J'ai alors séparé cette étude en plusieurs parties :

	Mode normal	Joueur 1 : 20 c Joueur 2 : 36 c	Joueur 1 : 10c Joueur 2 : 46c	Avec les cartes 2 à 4	Avec les cartes 2 à 9	Avec les têtes (10 à 13)
Joueur 1 gagne	51	35	28	53	50	51
Joueur 2 gagne	49	65	72	47	50	49
Durée moyenne	265	259	207	6	75	13

En regardant de plus près, nous pouvons ressortir deux hypothèses.
La première étant que, moins il y a de carte, moins il y a de tours en moyenne.
La seconde étant que quand il y a le même nombre de cartes (ici 56 cartes), si un joueur à plus de cartes qu'un autre, la partie durera moins longtemps.

▪ Etude sur une forme de force

Dans un premier temps en refessant un tableau de tours moyens, nous obtenons :

	Mode normal	Joueur 1 : 20 c Joueur 2 : 36 c	Joueur 1 : 10c Joueur 2 : 46c	Avec les cartes 2 à 4	Avec les cartes 2 à 9	Avec les têtes (10 à 13)
Joueur 1 gagne	49	45	19	52	43	49
Joueur 2 gagne	51	55	81	48	57	51
Durée moyenne	264	237	135	7	90	14

En prêtant attention aux cases violettes, nous pouvons déjà remarquer qu'un joueur ayant plus de cartes que son adversaire, a plus de chance de gagner.

En comparant avec la colonne du mode normal, qui est quasiment représentatif (50/50). Dans le cas ou le joueur 2 à 10 cartes de plus, il gagne environ 5 parties de plus. Et dans le cas où il a 20 cartes de plus, il gagne environ 30 parties de plus. Nous pouvons déjà affirmer qu'un joueur ayant plus de cartes que son adversaire a plus de chance de gagner.

Lors d'une partie normale, si nous connaissons les jeux des deux joueurs comme le cas suivant :

Jeu1	1	6	13	11	13	7	5	1	4	6	12	5	12	11	12	1	12	13	5	1	7	9	7	8	3	6
Jeu2	9	10	7	10	9	11	10	3	5	4	11	2	2	4	3	2	4	8	8	3	2	6	13	8	10	9

Nous pouvons imaginer qu'en comparant les premières manches, obtenant :

Jeu1	1	6	13	11	13	7	5	1	4	6	12	5	12	11	12	1	12	13	5	1	7	9	7	8	3	6
Jeu2	9	10	7	10	9	11	10	3	5	4	11	2	2	4	3	2	4	8	8	3	2	6	13	8	10	9
Res	2	2	1	1	1	2	2	2	2	1	1	1	1	1	1	2	1	1	2	2	1	1	2	0	2	2

13 manches gagnées par le joueur 1 et 12 manches gagnées par le joueur 2. Je pourrais supposer que le joueur qui va obtenir le plus de cartes, va par la suite gagner, comme nous avons pu le constater au début. Pour ce premier exemple, la chance m'a souri et c'est bien le joueur 1 qui va gagner la partie. Mais en simulant cette idée sur 100 parties, je ne devine le gagnant que sur 60 parties. Ce qui est certes au-dessus de la moyenne, mais pas assez au-dessus pour s'y fier.

▪ Gagner 10 parties sur 10 parties

Une autre question pourrait être de se demander, sur 10 parties, combien de pourcent de chance de gagner les 10 ? Suivant une loi binomiale de n = 10 :

- Mode normal

Ici, p=0.5
P(X=10) = (10C10)*0.5^(10)*(1-0.5)^(10-10)
P(X=10) = 0.00098

- Mode test où nous avons 46 cartes

Ici, p = 0.8
P(X=10) = (10C10)*0.2^(10)*(1-0.2)^(10-10)
P(X=10) = 0.10737

Pour conclure sur cette étude, il est quasi impossible (<1%) de gagner 10 manches sur 10.

▪ Gagner sachant qu'on perd les premières manches

Ici, nous cherchons l'événement « Gagner sachant qu'on perd les premières manches dans un jeu normal »
En reprenant l'étude de forme de force. Nous avons les événements :
A : « Gagner »
B : « Perdre les premières manches »

En P(A)=0.5 et P(B) = 0.4

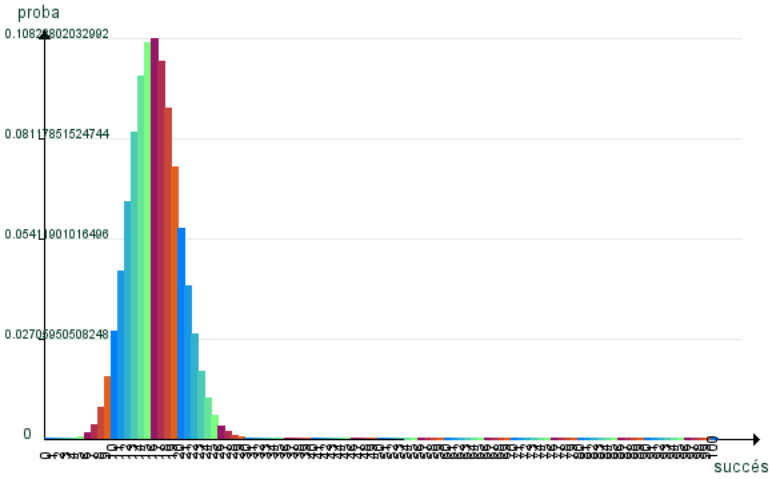
D'après la formule de Bayes
P(A|B) = P(B∩A)/P(B) = 0.4*0.5/0.4 = 0.5

Pour conclure sur cette étude, il y a autant de change de gagner en perdant les premières manches, qu'en les gagnants.

▪ Gagner avec une seule carte

Comme nous avons pu le voir, un joueur ayant plus de carte a plus de chance de gagner ! Mais il n'est pas impossible de gagner avec une seule carte. En simulant 100 parties où le joueur 1 n'avait qu'une carte, à ma grande surprise, le joueur 1 a gagné 16 parties sur 100. En gardant la probabilité de 16/100 ➔ 0.16. Suivant la loi binomiale avec n=100.
P(X<= 30) = 0.998599

E(X) = n*p = 0.16 * 100 = 16
V(x) = n*p*(1-p) = 16*(1-0.16) = 13.44
σ (X) = sqrt(V(x)) = 3.67



- **Probabilité que notre carte gagne – études des AS**

Chaque carte à une force sur les autres. Les 8 par exemples battent les 7,6,5,4,3,2. Autrement dit, les 8 battent 24 cartes. Ici, nous cherchons à déterminer la probabilité qu'une carte gagne sur les autres. Partons du principe que les 2 ne battent pas les 2 et ainsi les as ne battent pas les as. En faisant des produits en croix, en définissant que le paquet de 52 cartes correspondait à 100%, nous obtenons le tableau suivant :

Carte	Combien de carte gagne	Proba gagner en %	Carte	Combien de carte gagne	Proba gagner en %
2	0	0 %	9	28	53.85 %
3	4	7.69 %	10	32	61,54 %
4	8	15.38 %	V	36	69.23 %
5	12	23.08 %	D	40	76.92 %
6	16	30.77 %	R	44	84.62 %
7	20	38.46 %	A	48	92.31 %
8	24	46.15 %			

Ainsi, nous constatons que la carte la plus forte est l'as. Si nous considérons X, une variable uniforme sur les 13 cartes possibles, P(X=AS) = 4/52 = 1/13. Maintenant, cherchons la probabilité d'avoir les 4 as à la suite dans le paquet. P{X1,X1,X1,X1} = {As,As,As,As} = 4 * 4/52 * 3/51 * 2/50 * 1/49 = 0.00001477514

Bon, cela est relativement faible. Alors la probabilité de gagner une bataille avec un as :
P{X,Y,AS} = {carte 1, carte inconnue, AS}
Pour cette étude, nous supposerons que les cartes auront des valeurs toutes différentes.
Ainsi, P{X,Y,AS} = 3 * 4/52*4/51*4/50 = 2,4%

▪ Partie nulle

Il existe une permutation donnant un jeu périodique avec un nombre fini de distributions possible. Ainsi, une partie peut être nulle :

- Si les joueurs ont les mêmes cartes dans le même ordre.

Pour que les joueurs aient les mêmes cartes dans le même ordre, il faut qu'en mélangeant le paquet, nous obtenons quelque quelque-chose suivant cette idée :
Paquet {1,1,8,8,13,13,5,5,3,3,…}
Chaque carte doit être suivie d'une de ses jumelles. Ainsi, nous avons une probabilité détaillée de :

(4/52)*(3/51)*(2/50)*(1/49)	2
(4/48)*(3/47)*(2/46)*(1/45)	3
(4/44)*(3/43)*(2/42)*(1/41)	4
(4/40)*(3/39)*(2/38)*(1/37)	5
(4/36)*(3/35)*(2/34)*(1/33)	6
(4/32)*(3/31)*(2/30)*(1/29)	7
(4/28)*(3/27)*(2/26)*(1/25)	8
(4/24)*(3/23)*(2/22)*(1/21)	9
(4/20)*(3/19)*(2/18)*(1/17)	10
(4/16)*(3/15)*(2/14)*(1/13)	V
(4/12)*(3/11)*(2/10)*(1/9)	D
(4/8)*(3/7)*(2/6)*(1/5)	R
(4/4)*(3/3)*(2/2)*(1/1)	A
52*reste = 5.43e-49	Total

Ainsi, la probabilité reste faible, mais une partie peut quand même être nulle.

Bien sur, moins il y a de cartes, plus la partie à plus de chance d'être nulle ! Exemple avec les 2,3 et 4 :
12*(4/12)*(3/11)*(2/10)*(1/9)*(4/8)*(3/7)*(2/6)*(1/5)*(4/4)*(3/3)*(2/2)*(1/1) = 0.00035 soit 0.035% de chance de faire partie nulle.

- **La remise dans le paquet influence-t-elle le jeu ?**

En reprenant la fonction bataille, rappelez-vous que nous ajoutions d'abord notre carte avant celle de l'adversaire. Pour vérifier ou non la théorie sur l'importance ou non de la remise dans le paquet voici les correctifs apportés :

Première fonction bataille	Nouvelle fonction bataille	Explication
jeu1=[jeu1,jeu1(1),jeu2(1)];	jeu1=[jeu1,jeu2(1),jeu1(1)];	Quand le joueur1 gagne, il remettra d'abord la carte de l'adversaire avant la sienne
jeu2=[jeu2,jeu2(1),jeu1(1)];	jeu2=[jeu2,jeu1(1),jeu2(1)];	Quand le joueur2 gagne, il remettra d'abord la carte de l'adversaire avant la sienne
Bataille=[Bataille,jeu1(1),jeu2(1)]	Bataille=[Bataille,jeu2(1),jeu1(1)]	Dans la matrice bataille, nous allons d'abord mettre les cartes du joueur 2 avant celle du joueur 1.

Notons que j'inverse simplement l'ordre de remise dans le paquet. Et non pas, je le change. On pourrait également faire des tests où la carte du joueur 2 est toujours remise en première par rapport au joueur1.

Dans un premier temps, j'ai voulu comparer sur 100 parties avec chaque fonction bataille, sur combien de parties le changement de remise allait influencer le cours de la partie.

Sur 100 parties, le changement de remise a influencé 48 parties. Soit environ 50% de chance que le vainqueur change en fonction de l'ordre de remise.

Par la suite j'ai voulu voir si cela changeait également le nombre de tours moyens ! Sur 100 parties, avec la première méthode de remise, la moyenne était de 285 tours, alors qu'avec l'autre méthode de remise, nous avons 269 coups.

Nous pouvons conclure que l'ordre de remise des cartes influence énormément le vainqueur de la partie.

3 – Annexe

Possible améliorations

La principale amélioration de mon programme est le problème d'ergonomie. Par exemple, les 3 fonctions de distributions pourraient être résumées en une seule tout comme les deux fonctions de bataille pourraient être résumés en une seule. Certes, ce choix est fait pour mieux visualiser les fonctions et le fonctionnement, mais il aurait été préférable de faire moins de ligne de code.

Code menu

```
// Deux lignes utiles à mettre au début de chaque script :
xdel(winsid());              // ferme toutes les fenetres a chaque nouvel appel du script
clear;                       // nettoie toutes les variables a chaque nouvel appel du script
mode(0);                     // pour que le script se comporte comme une console (pas de point virgule => affichage du
résultat)
// Charge les fonctions du TP
exec("proj_fonctions.sci", -1);
// Déclaration des variables
x = [];
y = [];
gagnant = 0;
temps = 0;
nbparties = 0;

// Menu
printf('=== CHOIX DU MODE ===\n');
printf("0 - quitter \n")
printf("1 - mode normal\n")
printf("2 - mode test\n")
printf("3 - mode stats\n")
choixmode = input('Choisir avec le numéro associé ');
if choixmode == 0 then
   halt()
end
while choixmode ~= 1 && choixmode ~= 2 && choixmode ~= 3
   printf('=== CHOIX DU MODE ===\n');
   printf("0 - quitter\n")
   printf("1 - mode normal\n")
   printf("2 - mode test\n")
   printf("3 - mode stats\n")
   choixmode = input('Choisir avec le numéro associé ');
end

////////////////////////////////////////////////////////////////////////
//////////////////////////// MODE NORMAL ///////////////////////////////
////////////////////////////////////////////////////////////////////////

printf("Vous avez choisi le mode numéro %d \n",choixmode);
if choixmode == 1 then
   printf('=== MODE NORMAL ===\n');
   printf('====================\n');
   [x,y]=distribue();
   printf("Jeu1 \n");
   afficherjeu(x);
   printf("Jeu2 \n");
   afficherjeu(y);
   [gagnant,temps]=bataille(x,y);
   printf("Le joueur %d gagne en %d tours !!! ", gagnant, temps);
end
```

```
////////////////////////////////////////////////////////////////
//////////////////////////// MODE TEST ////////////////////////////
////////////////////////////////////////////////////////////////

if choixmode == 2 then
    printf('=== MODE TEST ===\n');
    printf('==================\n');
    printf('1 - Nombre différent de cartes aux joueurs\n');
    printf('2 - Ne pas distribuer toutes les cartes\n');
    choixsousmode = input('Choisir avec le numéro associé ');
    while choixsousmode ~= 1 && choixsousmode ~= 2
        printf('=== MODE TEST ===\n');
        printf('==================\n');
        printf('1 - Nombre différent de cartes aux joueurs\n');
        printf('2 - Ne pas distribuer toutes les cartes\n');
        choixsousmode = input('Choisir avec le numéro associé ');
    end
    if choixsousmode == 1 then
        choixnbcartes = input('Combien de cartes pour joueur 1 ? ');
        while int(choixnbcartes) > 55 || int(choixnbcartes) < 1
            printf('Choisir un nombre entre 1 et 55 ');
            choixnbcartes = input('Combien de cartes pour joueur 1 ? ');
        end
        nb1 = int(choixnbcartes);
        nb2 = 56 - nb1;
        //printf("%d %d",nb1,nb2);
        [x,y]=distribuepaslesmemes(nb1,nb2);
        printf("Jeu1 \n");
        afficherjeu(x);
        printf("Jeu2 \n");
        afficherjeu(y);
        [gagnant,temps]=bataille(x,y);
        printf("Le joueur %d gagne en %d tours !!! ", gagnant, temps);
        //printf("Le joueur %d gagne en %d tours !!! ", gagnant, temps);
    end
//////////////////////////// PAS TTE LES CARTES ////////////////////////////
    if choixsousmode == 2 then
        paquet = [];
        nbtt = 0;
        for i = 1:13
            printf('Voules-vous les %d ? 1 = oui | 0 = non ', i+1);
            resultat = input('');
            while resultat ~= 1 && resultat ~= 0
                printf('Voules-vous les %d ? 1 = oui | 0 = non ', i+1);
                resultat = input('');
            end
            if resultat == 1 then
                for j = 1:4
                    paquet(1,$+1) = i;
                    nbtt=nbtt+1;
                end
            end
        end
        [x,y]=distribuepastout(paquet,nbtt/2);
        printf("Jeu1 \n");
        afficherjeu(x);
        printf("Jeu2 \n");
        afficherjeu(y);
        [gagnant,temps]=bataille(x,y);
        printf("Le joueur %d gagne en %d tours !!! ", gagnant, temps);
    end
```

```
end

////////////////////////////////////////////////////////////////////
//////////////////////////// MODE STATS ////////////////////////////
////////////////////////////////////////////////////////////////////

if choixmode == 3 then
    printf('=== MODE STATS ===\n');
    printf('==================\n');
    printf('1 - Stats sur la durée moyenne\n');
    printf('2 - Stats sur une forme de force\n');
    printf('3 - Stats sur linfluence de la remise\n');

    choixsousmode = input('Choisir avec le numéro associé ');
    while choixsousmode ~= 1 && choixsousmode ~= 2 && choixsousmode ~= 3
        printf('1 - Stats sur la durée moyenne\n');
        printf('2 - Stats sur une forme de force\n');
        printf('3 - Stats sur linfluence de la remise\n');
        choixsousmode = input('Choisir avec le numéro associé ');
    end

    printf('==================\n');

    Etude_Stats(choixsousmode);


end
```

- ## Code fonction

```
//fonction de distribution normal
function [jeu1, jeu2]=distribue()
    //déclaration des variables
    paquetdecarte = [1 1 1 1 2 2 2 2 3 3 3 3 4 4 4 4 5 5 5 5 6 6 6 6 7 7 7 7 8 8 8 8 9 9 9 9 10 10 10 10 11 11 11 11
12 12 12 12 13 13 13 13];
    jeu1 = 1:26;
    jeu2 = 1:26;
    k=1
    i=1
    //distribution cartes jeu1
    //Tant que le joueur 1 n'a pas 26 cartes dans son paquet (26 car on part de k=1)
    while k ~=27
        //tirage au sort d'un nombre entre 1 et 52
        res = grand(1,1,"uin",1,52);
        //si la carte numéro res(tiré au sort au dessus) est présente dans paquetdecarte (valeur différente de 0) alors
        if paquetdecarte(1, res) ~= 0 then
            //on ajoute la carte au jeu1
            jeu1(1,k) = paquetdecarte(1, res);
            //on supprime ensuite la carte (on met sa valeur à 0) dans le paquetdecarte
            paquetdecarte(1, res) = 0;
            //on définie qu'on a ajouté une carte au jeu1
            k= k+1;
        end
    end
    j = 1;
    //distribution cartes jeu2, même principe que pour le jeu1
    while j ~=27
        res = grand(1,1,"uin",1,52);
        if paquetdecarte(1, res) ~= 0 then
            jeu2(1,j) = paquetdecarte(1, res);
            paquetdecarte(1, res) = 0;
            j = j + 1;
```

```
        end
    end
endfunction
//fonction de distribution alternative -> pas toutes les cartes
//même principe que la première fonction de distribution (détaillé)
function [jeu1, jeu2]=distribuepastout(paquetdecarte, nbcartes)
    jeu1 = 1:nbcartes;
    jeu2 = 1:nbcartes;
    k=1
    i=1
    z = nbcartes+1;
    u = nbcartes * 2;
    while k ~= z
        res = grand(1,1,"uin",1,u);
        if paquetdecarte(1, res) ~= 0 then
            jeu1(1,k) = paquetdecarte(1, res);
            paquetdecarte(1, res) = 0;
            k= k+1;
        end
    end
    j = 1;
    while j ~= z
        res = grand(1,1,"uin",1,u);
        if paquetdecarte(1, res) ~= 0 then
            jeu2(1,j) = paquetdecarte(1, res);
            paquetdecarte(1, res) = 0;
            j = j + 1;
        end
    end
endfunction
//fonction de distribution alternative -> pas le même nombre de cartes
function [jeu1, jeu2]=distribuepaslesmemes(nbcartes1, nbcartes2)
    jeu1 = [];
    jeu2 = [];
    recup = [];
    nbcartesenplus = 0;
    //distribution des cartes aux 2 joueurs (26 de chaque coté)
    [jeu1,jeu2]=distribue();
    //cas ou le joueur 1 a plus de cartes
    if nbcartes1 > nbcartes2 then
        //on calcul le nombre de carte en plus à donner au joueur1
        nbcartesenplus = 26 - nbcartes2;
        //on transforme ce nombre en entier pour éviter un problème de variable
        nbcartesenplus = int(nbcartesenplus);
        //pour le nombre de carte à donner en plus au joueur 1
        for i = 1:nbcartesenplus
            //on définie une matrice qui prent la 1ère carte du joueur2
            recup(1) = int(jeu2(1));
            //on la donne au joueur 1 (en fin de son paquet)
            jeu1(1,$+1) = recup(1);
            //on supprime la 1ère carte du joueur 2 qui vient d'être donné au joueur1
            jeu2(1) = [];
        end
    end
    //cas ou le joueur 2 a plus de cartes ==> Même idée que pour le cas au dessus
    if nbcartes1 < nbcartes2 then
        nbcartesenplus = 26 - nbcartes1;
        nbcartesenplus = int(nbcartesenplus);
        for i = 1:nbcartesenplus
            recup(1) = int(jeu1(1));
            jeu2(1,$+1) = recup(1);
```

```
        jeu1(1) = [];
      end
    end
endfunction
//fonction de la bataille
function [gagnant, temps]=bataille(jeu1, jeu2)
    temps =0;
    gagnant = 0;
    Bataille=[];
    B=[];
    //tant qu'il n'y a pas de gagnant
    while(jeu1 ~=[] && jeu2 ~= [])
    //si le joueur1 gagne la manche
      if(jeu1(1)>jeu2(1))then
        //ajout des cartes dans le jeu1
        jeu1=[jeu1,jeu1(1),jeu2(1)];
        TailleM=size(jeu1);
        //suppression des premières cartes dans les 2 paquets
        jeu1=suppJeu(jeu1,TailleM(1,2));
        TailleM=size(jeu2);
        jeu2=suppJeu(jeu2,TailleM(1,2));
        //incrémentation du nombre de tours
        temps = temps +1;
      else
        //si le joueur2 gagne la manche
        //ajout des cartes dans le jeu2
        jeu2=[jeu2,jeu2(1),jeu1(1)];
        //suppression des premières cartes dans les 2 paquets
        TailleM=size(jeu1);
        jeu1=suppJeu(jeu1,TailleM(1,2));
        TailleM=size(jeu2);
        jeu2=suppJeu(jeu2,TailleM(1,2));
        //incrémentation du nombre de tours
        temps = temps +1;
      end;
      //cas où les cartes sont les mêmes
      if(jeu1(1)==jeu2(1))
        //vérification si gagnant
        while(jeu1(1)==jeu2(1))
          if(jeu1 == [] || jeu2==[])
            then
            if(jeu1 == [])
            then gagnant = 1;
            return;
            else gagnant = 2;
            return;
            end;
          end;
          //mise à jour de Bataill et des deux jeux
          Bataille=[Bataille,jeu1(1),jeu2(1)]
          TailleM=size(jeu1);
          jeu1=suppJeu(jeu1,TailleM(1,2));
          TailleM=size(jeu2);
          jeu2=suppJeu(jeu2,TailleM(1,2));
          //vérification si gagnant
          if(jeu1 == [] || jeu2==[])
            then
            if(jeu1 == [])
            then gagnant = 1;
            return;
            else gagnant = 2;
```

```
            return;
            end;
        end;
        //mise à jour de Bataill et des deux jeux (2ème carte)
        Bataille=[Bataille,jeu1(1),jeu2(1)]
        TailleM=size(jeu1);
        jeu1=suppJeu(jeu1,TailleM(1,2));
        TailleM=size(jeu2);
        jeu2=suppJeu(jeu2,TailleM(1,2));
        //vérification si gagnant
        if(jeu1 == [] || jeu2==[])
            then
            if(jeu1 == [])
            then gagnant = 1;
            return;
            else gagnant = 2;
            return;
            end;
        end;
    end;
    //vérification de qui gagne
    if(jeu1(1)>jeu2(1))then
        jeu1=[jeu1,Bataille,jeu1(1),jeu2(1)];
        TailleM=size(jeu1);
        jeu1=suppJeu(jeu1,TailleM(1,2));
        TailleM=size(jeu2);
        jeu2=suppJeu(jeu2,TailleM(1,2));
        temps = temps +1;
    else
        jeu2=[jeu2,Bataille,jeu2(1),jeu1(1)];
        TailleM=size(jeu1);
        jeu1=suppJeu(jeu1,TailleM(1,2));
        TailleM=size(jeu2);
        jeu2=suppJeu(jeu2,TailleM(1,2));
        temps = temps +1;
    end;
    Bataille=[];
  end;
end;
//vérification si gagnant
if(jeu1 == [])
  then gagnant = 2;
  else gagnant = 1;
end;

endfunction
//fonction de bataille alternative -> inverse ordre de remise
//même principe que la première fonction de bataille (détaillé)
function [gagnant, temps]=batailleGalanterie(jeu1, jeu2)
  temps =0;
  gagnant = 0;
  Bataille=[];
  B=[];
  while(jeu1 ~=[] && jeu2 ~= [])
  if(jeu1(1)>jeu2(1))
    then jeu1=[jeu1,jeu2(1),jeu1(1)];
    TailleM=size(jeu1);
    jeu1=suppJeu(jeu1,TailleM(1,2));
    TailleM=size(jeu2);
    jeu2=suppJeu(jeu2,TailleM(1,2));
    temps = temps +1;
```

```
else
    jeu2=[jeu2,jeu1(1),jeu2(1)];
    TailleM=size(jeu1);
    jeu1=suppJeu(jeu1,TailleM(1,2));
    TailleM=size(jeu2);
    jeu2=suppJeu(jeu2,TailleM(1,2));
    temps = temps +1;
end;
if(jeu1(1)==jeu2(1))
    while(jeu1(1)==jeu2(1))
        if(jeu1 == [] || jeu2==[])
            then
            if(jeu1 == [])
            then gagnant = 1;
            return;
            else gagnant = 2;
            return;
            end;
        end;
        Bataille=[Bataille,jeu2(1),jeu1(1)]
        TailleM=size(jeu1);
        jeu1=suppJeu(jeu1,TailleM(1,2));
        TailleM=size(jeu2);
        jeu2=suppJeu(jeu2,TailleM(1,2));
        if(jeu1 == [] || jeu2==[])
            then
            if(jeu1 == [])
            then gagnant = 1;
            return;
            else gagnant = 2;
            return;
            end;
        end;
        Bataille=[Bataille,jeu2(1),jeu1(1)]
        TailleM=size(jeu1);
        jeu1=suppJeu(jeu1,TailleM(1,2));
        TailleM=size(jeu2);
        jeu2=suppJeu(jeu2,TailleM(1,2));
        if(jeu1 == [] || jeu2==[])
            then
            if(jeu1 == [])
            then gagnant = 1;
            return;
            else gagnant = 2;
            return;
            end;
        end;
    end;
    if(jeu1(1)>jeu2(1))
    then jeu1=[jeu1,Bataille,jeu2(1),jeu1(1)];
    TailleM=size(jeu1);
    jeu1=suppJeu(jeu1,TailleM(1,2));
    TailleM=size(jeu2);
    jeu2=suppJeu(jeu2,TailleM(1,2));
    temps = temps +1;
else
    jeu2=[jeu2,Bataille,jeu1(1),jeu2(1)];
    TailleM=size(jeu1);
    jeu1=suppJeu(jeu1,TailleM(1,2));
    TailleM=size(jeu2);
    jeu2=suppJeu(jeu2,TailleM(1,2));
```

```
        temps = temps +1;
    end;
    Bataille=[];
    end;
 end;

 if(jeu1 == [])
     then gagnant = 2;
    else gagnant = 1;
 end;

endfunction

function L=suppJeu(A, taille)
    L=[];
    t=taille-1
    for i= 1:t
        L=[L,A(i+1)];
    end
endfunction

function afficherjeu(jeucartes)
    for i = 1:length(jeucartes)
        printf("%d ", jeucartes(i));
    end
    printf("\n");
endfunction
//Fonction d'étude de proba et stats
function Etude_Stats(choixEtude)
    nbparties = 100;
    jeu1 = [];
    jeu2 = [];
    j1gg = 0;
    j2gg = 0;
    moyenne = 0;
    //étude de la durée moyenne d'une partie + gagnant, etc
    if choixEtude == 1 then
        //simulation de 100 parties normales
        printf("Simulation sur 100 parties\n");
        printf("===== Partie normales =====\n");
        for i = 1:100
            //distribution
            [jeu1,jeu2]=distribue();
            //bataille
            [gagnant,temps]=bataille(jeu1,jeu2);
            //compter combien de victoirede chaque cotés
            if gagnant == 1 then
                j1gg = j1gg+1;
            else
                j2gg = j2gg+1;
            end
            //mise à jours de la moyenne
            moyenne = moyenne + temps;
        end
        //calcul de la moyenne
        moyenne = moyenne/100;
        //affichages caractèristiques
        printf("Le joueur 1 a gagné : %d parties\n",j1gg);
        printf("Le joueur 2 a gagné : %d parties\n",j2gg);
        printf("Une partie dure en moyenne %d tours\n",moyenne);
```

```
printf("\n===== Partie test =====\n");
printf("=> Pas le même nombre de carte \n Joueur 1 : 20 cartes \n Joueur 2 : 36 cartes\n");
jeu1 = [];
jeu2 = [];
j1gg = 0;
j2gg = 0;
moyenne = 0;
 //simulation de 100 parties où le joueur 2 à plus de carte
for i = 1:100
    //distribution
    [jeu1,jeu2]=distribuepaslesmemes(20,36);
    //bataille
    [gagnant,temps]=bataille(jeu1,jeu2);
    //maj gagnant
    if gagnant == 1 then
        j1gg = j1gg+1;
    else
        j2gg = j2gg+1;
    end
    //maj moyenne
    moyenne = moyenne + temps;
end
//calcul de la moyenne
moyenne = moyenne / 100;
//affichages caractèristiques
printf("Le joueur 1 a gagné : %d parties\n",j1gg);
printf("Le joueur 2 a gagné : %d parties\n",j2gg);
printf("Une partie dure en moyenne %d tours\n",moyenne);

//simulation de 100 parties où le joueur 2 à beaucoup plus de carte
printf("\n=> Pas le même nombre de carte \n Joueur 1 : 10 cartes \n Joueur 2 : 46 cartes\n");
jeu1 = [];
jeu2 = [];
j1gg = 0;
j2gg = 0;
moyenne = 0;
for i = 1:100
    [jeu1,jeu2]=distribuepaslesmemes(10,46);
    [gagnant,temps]=bataille(jeu1,jeu2);
    if gagnant == 1 then
        j1gg = j1gg+1;
    else
        j2gg = j2gg+1;
    end
    moyenne = moyenne + temps;
end
moyenne = moyenne / 100;
printf("Le joueur 1 a gagné : %d parties\n",j1gg);
printf("Le joueur 2 a gagné : %d parties\n",j2gg);
printf("Une partie dure en moyenne %d tours\n",moyenne);

//simulation de 100 parties où l'on joue qu'avec les 2,3 et 4
printf("\n=> Seulement avec les cartes 2 à 4\n");
paquet = [1 1 1 1 2 2 2 2 3 3 3 3];
jeu1 = [];
jeu2 = [];
j1gg = 0;
j2gg = 0;
moyenne = 0;
for i = 1:100
```

```
    [jeu1,jeu2]=distribuepastout(paquet,6);
    [gagnant,temps]=bataille(jeu1,jeu2);
    if gagnant == 1 then
      j1gg = j1gg+1;
    else
      j2gg = j2gg+1;
    end
    moyenne = moyenne + temps;
  end
  printf("Le joueur 1 a gagné : %d parties\n",j1gg);
  printf("Le joueur 2 a gagné : %d parties\n",j2gg);
  moyenne = moyenne / 100;
  printf("Une partie dure en moyenne %d tours\n",moyenne);
  //simulation de 100 parties où l'on joue qu'avec les cartes 2 à 9
  printf("\n=> Seulement avec les cartes 2 à 9\n");
  paquet = [1 1 1 1 2 2 2 2 3 3 3 3 4 4 4 4 5 5 5 5 6 6 6 6 7 7 7 7 8 8 8 8];
  jeu1 = [];
  jeu2 = [];
  j1gg = 0;
  j2gg = 0;
  moyenne = 0;
  for i = 1:100
    [jeu1,jeu2]=distribuepastout(paquet,16);
    [gagnant,temps]=bataille(jeu1,jeu2);
    if gagnant == 1 then
      j1gg = j1gg+1;
    else
      j2gg = j2gg+1;
    end
    moyenne = moyenne + temps;
  end
  moyenne = moyenne / 100;
  printf("Le joueur 1 a gagné : %d parties\n",j1gg);
  printf("Le joueur 2 a gagné : %d parties\n",j2gg);
  printf("Une partie dure en moyenne %d tours\n",moyenne);
  //simulation de 100 parties où l'on joue qu'avec les 10,11,12 et 13
  printf("\n=> Seulement avec les plus fortes valeurs : 10 à 13\n");
  paquet = [10 10 10 10 11 11 11 11 12 12 12 12 13 13 13 13];
  jeu1 = [];
  jeu2 = [];
  j1gg = 0;
  j2gg = 0;
  moyenne = 0;
  for i = 1:100
    [jeu1,jeu2]=distribuepastout(paquet,8);
    [gagnant,temps]=bataille(jeu1,jeu2);
    if gagnant == 1 then
      j1gg = j1gg+1;
    else
      j2gg = j2gg+1;
    end
    moyenne = moyenne + temps;
  end
  moyenne = moyenne / 100;
  printf("Le joueur 1 a gagné : %d parties\n",j1gg);
  printf("Le joueur 2 a gagné : %d parties\n",j2gg);
  printf("Une partie dure en moyenne %d tours\n",moyenne);


end
```

```
//étude d'une potentiel force
if choixEtude == 2 then
    jeu1 = [];
    jeu2 = [];
    win1 = 0;
    win2 = 0;
    quiwin = [];
    suppwin = 0;
    gagnant = 0;
    temps = 0;
    gagne = "";
    //exemple en affichant 2 jeu de carte
    [jeu1,jeu2]=distribue();
    printf("Jeu1 \n");
    afficherjeu(jeu1);
    printf("Jeu2 \n");
    afficherjeu(jeu2);
    //affiche le gagnant de chaque premières manches
    for i = 1:26
        if jeu1(i) > jeu2(i)then
            quiwin(1,i) = 1;
            win1 = win1+1;
        end
        if jeu1(i) < jeu2(i)then
            quiwin(1,i) = 2;
            win2 = win2+1;
        end
        if jeu1(i) == jeu2(i)then
            quiwin(1,i) = 0;
        end
    end
    //affichage caractèristique de qui gagne combien de premières manches
    printf("\nEn comptant les premières manches nous obtenons : \n")
    afficherjeu(quiwin);
    printf("%d manches gagné par le joueur 1\n",win1);
    printf("%d manches gagné par le joueur 2\n",win2);

    if win1 > win2 then
        suppwin = 1;
    else
        suppwin = 2;
    end

    [gagnant,temps]=bataille(jeu1,jeu2);

    if suppwin == gagnant then
        gagne = "Gagné";
    else
        gagne = "Perdu";
    end
    //fin de l'exemple et début de la simulation
    printf("Nous pouvons supposer que le joueur %d va gagner\n",suppwin);
    printf("Dans ce cas, cest le joueur %d qui gagne. Cest donc %s \n",gagnant, gagne);
    printf("\nSimulation sur 100 parties :\n")
    gagne = 0;
    //lancement de 100 parties
    for i = 1:100
        jeu1 = [];
        jeu2 = [];
        win1 = 0;
        win2 = 0;
```

```
        suppwin = 0;
        gagnant = 0;
        temps = 0;
        //distribution
        [jeu1,jeu2]=distribue();
        for i = 1:26
            if jeu1(i) > jeu2(i)then
                win1 = win1+1;
            end
            if jeu1(i) < jeu2(i)then
                win2 = win2+1;
            end
        end
        //suposition gagnant
        if win1 > win2 then
            suppwin = 1;
        else
            suppwin = 2;
        end

        [gagnant,temps]=bataille(jeu1,jeu2);
        //compter le nombre de fois où les suppositions sont bonnes
        if suppwin == gagnant then
            gagne = gagne +1;
        end
    end
    printf("Nous avons déviné le gagnant de %d parties sur 100 parties !", gagne);
end
//etude de la remise dans le paquet
if choixEtude == 3 then
    gagnantp1 = 0;
    gagnantp2 = 0;
    tmps1 = 0;
    tmps2 = 0;
    nbdiff = 0;
    moy1 = 0;
    moy2 = 0;
    for i = 1:100
        jeu1 = [];
        jeu2 = [];
        gagnantp1 = 0;
        gagnantp2 = 0;
        tmps1 = 0;
        tmps2 = 0;
        [jeu1,jeu2]=distribue();
        [gagnantp1,tmps1]=bataille(jeu1,jeu2);
        [gagnantp2,tmps2]=batailleGalanterie(jeu1,jeu2);
        if gagnantp1 ~= gagnantp2 then
            nbdiff = nbdiff + 1;
        end
        moy1 = moy1 + tmps1;
        moy2 = moy2 + tmps2;
    end
    moy1 = moy1/100;
    moy2 = moy2/100;
    printf("Le changement de remise a influencé %d parties\n",nbdiff);
    printf("Moyenne 1 : %d \nMoyenne 2 : %d\n",moy1, moy2);
end
endfunction
```