Université de Liège
Faculté des Sciences Appliquées

Rapport sur la résolution probabiliste du problème des anniversaires

Dans le cadre du cours d'éléments du calcul des probabilités

Année Académique 2016-2017

Moitroux Olivier
s152122

Étudiant en 2e Bac Ingénieur Civil
Le 8 octobre 2017

Introduction

Ce projet vise à familiariser l'étudiant avec la notion d'arbre de probabilités et avec la méthode de Monte Carlo au moyen de la résolution pratique du *paradoxe des anniversaires*.

Questions

1 Méthode de Lehman et Leighton

(1.a) Tout d'abord, nous considérerons qu'il n'y a pas plus de chances de naître un certain jour de l'année plutôt qu'un autre, c'est-à-dire que la loi de probabilité est uniforme. Malgré le temps d'exécution assez catastrophique de 5,89 secondes, la réponse fournie (0,27%) par la fonction `naiveBirthdaySol.m` est bien correcte.

En effet, la probabilité que la première personne soit née le même jour que la seconde est le contraire de la probabilité que la seconde personne ait son anniversaire un jour différent. Soit,

$$1 - 364/365 = 1 - 0.9973 = 0,0027 = 0,27\%$$

Après avoir attribué une date d'anniversaire à la première personne puis à la deuxième, cette fonction va, grâce à `createBirthdayTree.m`, construire un arbre peu performant. En effet, de la racine partent 365 branches ayant chacune une probabilité d'occurrence de 1/365 vu l'uniformité de la loi de probabilité. De chacun de ces nouveaux nœuds vont encore être recréés 365 autres branches et ainsi de suite... On constate ainsi que pour N=2, notre arbre cumule un total de $365^2 = 133225$ feuilles d'où la mauvaise performance.

(1.b) Étant donné la procédure utilisée pour créer l'arbre, le pire est à présager pour $N = 3$. Comme expliqué ci-dessus, l'arbre possédera désormais un total de 365^3 feuilles et le temps d'exécution attendu devrait tourner aux alentours de $5,89.365 = 2149,85$ secondes soit un peu plus d'une demi-heure. De la même manière pour $N = 4$ et $N = 5$, le code nécessiterait respectivement 220 heures et 9 années (et un mois).

De façon plus générale, par hypothèse que le temps d'exécution est proportionnel au nombre de branches de l'arbre, on peut calculer le temps d'exécution nécessaire pour pour n'importe quel N :

$$t_N = \frac{5,896463}{365^2}.365^N = 5,896463.365^{N-2}s$$

Cette méthode naïve de résolution est donc absolument à proscrire pour un nombre de personnes plus grand que deux.

(1.c) La méthode naïve consistait, à l'aide d'une boucle, à additionner les probabilités des 365^N feuilles correspondant à la configuration de l'arbre. Seulement, il est inutile de construire un arbre avec toutes les branches possibles quand seules quelques unes sont intéressantes.

Étant donné que cette approche n'est pas acceptable, considérons plutôt un arbre binaire. Chaque nœud de cette arbre se divisera en deux branches, l'une correspondant au cas où la condition est respectée, c'est-à-dire si deux personnes parmi N sont nées le même jour et l'autre correspondant au cas inverse. Il est ainsi possible de créer l'arbre au fur et à mesure des besoins.

Prenons le cas N = 3 comme illustration et désignons par j_i avec $i = \{1, 2, 3\}$ les dates d'anniversaires des personnes 1,2,3 :

Comme nous pouvons le constater sur la figure 1, la racine se divise en deux branches. Si $j_1 = j_2$, alors $P(j_1 = j_2) = 1/365$ et l'on s'arrête vu que la condition est remplie. En d'autres termes, cela nous est égal de savoir si la troisième personne est née le même jour que les deux premières ou non. Si par contre l'évènement n'a pas eu lieu($j_1 \neq j_2$), il nous faut continuer la construction de l'arbre en divisant la branche en deux autres.

Ainsi, deux cas se présentent à nouveau. Soit la troisième personne est née le même jour que la première ou la deuxième personne ($P(j_3 = j_1$ ou $j_3 = j2) = 2/365$), soit la troisième personne est née un autre jour que les deux autres avec $P(j_3 \neq j_2 \neq j_1) = 363/365$.

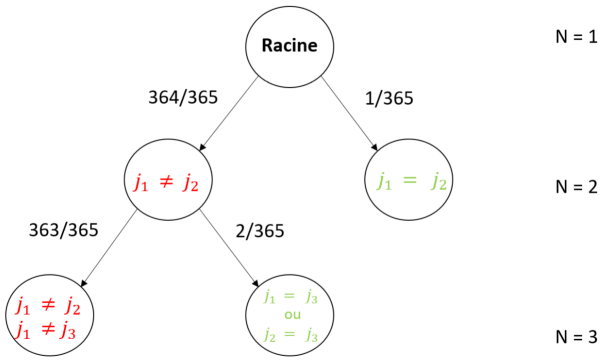

FIGURE 1 – Arbre binaire pour N = 3

On peut, bien entendu, procéder de la sorte pour n'importe quel N jusqu'à la Nième personne et en vérifiant à chaque nœud si la Nième personne est née le même jour qu'une des $N - 1$ autres. De cette manière, la fonction `Q1d`, disponible en Annexe A, permet de résoudre le problème pour n'importe quel N et utilise une formule déduite du fonctionnement de cet arbre.

(1.d) Le tableau ci-dessous reprend les résultats obtenus à l'aide de la fonction Matlab `Q1d` du fichier `Q1d.m` pour différentes valeurs de N.

N	Probabilité	Temps d'exécution (s)
2	0,00274	3,2464.e-06
3	0,008204	4,6377.e-06
4	0,016356	6,029.e-06
5	0,027136	4,1739.e-06
20	0,411438	4,6377.e-06
30	0,706316	9,2754.e-07
40	0,891232	4,6377.e-07
50	0,9703736	9,2754.e-07
60	0,994123	9,2754.e-07
80	0,999914	1,3913.e-06

FIGURE 2 – Probabilités pour qu'au moins 2 personnes parmi N aient leur anniversaire le même jour

Le principe est simple : au lieu de construire un arbre à 365 divisions par nœud, on trouve la solution directement en prenant l'inverse de la probabilité que toutes les autres personnes soient nées un autre jour, soit en faisant "1−parcours à gauche de l'arbre binaire".

$$P(N) = 1 - \prod_{i=1}^{N-1} \frac{365 - i}{365}$$

La première chose qui saute aux yeux est bien entendu le temps d'exécution dérisoire par rapport à la méthode précédente. Intuitivement les réponses sont assez logiques étant donné que la probabilité augmentera

avec le nombre de personnes. Elle augmente aussi d'autant plus vite qu'elle croît avec le nombre de jours ayant un anniversaire associé. En d'autres mots, ajouter une personne implique que l'on doit ajouter un terme supplémentaire (d'autant plus grand que le nombre de personnes est élevé) à la somme des probabilités anciennement calculées sans la nouvelle personne.

2 Méthode de Monte Carlo

(2.a) Il s'agit d'une distribution discrète de probabilités qui prend la valeur 1 avec la probabilité p et 0 avec la probabilité q = 1 – p donc d'une distribution de Bernoulli. En utilisant la fonction $Q1d$, on peut calculer :

$$\begin{cases} P(\chi = 1) = 0,891232 \\ P(\chi = 0) = 0,108768 \end{cases}$$

L'espérance mathématique, définie comme

$$E(\chi) = \sum \chi(\omega) P(\omega)$$

vaut dans notre cas

$$E(\chi) = 1.P(\chi = 1) + 0.P(\chi = 0) = 0,8782$$

La variance est quant à elle définie par

$$V(\chi) = P(\chi = 1)(1 - P(\chi = 1))$$

et vaut

$$V(\chi) = 0,8912 - 0,8912^2 = 0,096963$$

(2.b) La résolution de cette question peut se faire grâce à la fonction `Q2b` dans le fichier `Q2b.m`. Comme demandé, cette fonction crée un vecteur binaire de taille N (spécifiée par l'utilisateur) dont les éléments sont générés par la fonction `birthday40` et en calcule l'espérance ainsi que la variance. Celles-ci sont calculées respectivement grâce aux fonctions `mean` et `var` qui sont déjà implémentées dans MATLAB. Voici les résultats pour différentes valeurs de N :

N	**Espérance**	**Variance**
10	0,8	0.16
10^2	0,89	0,0979
10^3	0,8980	0,0916
10^4	0.8876	0.0998

FIGURE 3 – Espérance et variance pour différentes dimensions (N) de vecteurs

Les valeurs rendues par `birthday40` sont aléatoires et son évaluation est unique pour chaque élément du vecteur,c'est pourquoi d'autres résultats que ceux-ci peuvent être obtenus.
On remarque aussi qu'au fur et à mesure que la taille du vecteur considéré augmente, les valeurs tendent plus ou moins bien vers les résultats théoriques trouvés à la question précédente. Une manière simple d'améliorer la constance et la précision des résultats serait d'augmenter encore la taille du vecteur (au prix du temps d'exécution) ou encore mieux, de prendre une moyenne sur un grand nombre d'expériences (en gardant des vecteurs plus petits).

(2.c) On se propose ainsi de suivre la deuxième méthode. La fonction `Q2c`, disponible en annexe, permet de dresser une série de résultats qui sont repris dans la figure 4. Cette fonction prend en argument non seulement les dimensions des vecteurs souhaités mais également, si désiré, une valeur booléenne *Display* permettant d'afficher les résultats. On utilisera l'appel $Q2c([10 10^2 10^3 10^4])$; pour générer les éléments de la figure 4.

Notons tout de même que le temps d'exécution nécessaire à la génération de ce tableau est d'environ cinq minutes (300,220886 s) pour 1000 expériences. Ainsi, il n'est pas recommandé d'utiliser une taille de vecteur de 10^5, sachant qu'il faut déjà approximativement 270 secondes pour générer la dernière ligne du tableau. De cette manière, vu que le temps d'exécution semble augmenter d'un facteur 10 d'une ligne à l'autre, il faudrait approximativement 3/4 heures pour aller jusque $N = 10^5$.

Dimension (N)	**Espérance**		**Variance**		**Temps d'exécution(s)**
	Espérance	Variance	Espérance	Variance	
10	0,8902	0,010364	0,08738	0,0052589	0,34888
10^2	0,89212	0,00093271	0,095309	0,00056669	2,7626
10^3	0,89174	9,9124e-05	0,09644	6,0603e-05	27,056
10^4	0,89123	9,6189e-06	0,096927	5,8855e-06	271,7518

FIGURE 4 – Moyenne des espérances et variances pour différentes dimensions(N) de vecteurs

Comme précisé au point précédent, augmenter le nombre d'expériences sur des vecteurs plus petits est un choix plus judicieux au vu du temps d'exécution et de la précision.

Qu'en est-il des résultats ? On voit clairement qu'il y a une évolution par rapport au point précédent. Les valeurs tendent bien à se rapprocher des valeurs théoriques et les écarts entre les lignes sont moindres. Bien évidemment, la dernière ligne correspond à la meilleure approximation.

(2.d) La fonction `Q2d` disponible en annexe répond à la question posée. Elle prend comme argument une variable N représentant la taille d'un vecteur dont les éléments suivent la loi de probabilité étudiée.

Les éléments de ce vecteur sont générés à l'aide de la fonction *normrnd* qui renvoit des valeurs répondant à une loi normale N(μ ; σ) dont l'espérance (μ) vaut 10 et l'écart-type(σ) 5^2 dans notre cas.

Dimension (N)	**Espérance**	**Variance**	**Temps d'exécution (s)**
10	-2,2847	3466,9431	0,2421 s
10^2	1,1683	377,2848	1,6587 s
10^3	0,6373	299,8637	16,4848 s
10^4	1,2534	1181,7798	2 min 48 s
10^5	0,1650	4,5018	$\approx$ 27 min

FIGURE 5 – Espérance et variance en fonction de N

On constate rapidement que l'on ne peut tirer de conclusions intéressantes de ce tableau si ce n'est le fait que la méthode de Monte Carlo n'est pas adaptée à la résolution du problème. En effet, aucun résultat ne semble converger.

Pourquoi cela ? Notamment à cause de la fonction *normrdn* qui affecte aux éléments des vecteurs des valeurs aléatoires. La moyenne et la variance s'en trouvent donc également affectées d'où en plus un résultat différent à chaque expérience... On comprend alors rapidement pourquoi la variance ne tend pas à diminuer vers 0.

En conclusion, on préfèrera plutôt la loi de Cauchy pour la résolution de problèmes faisant intervenir le quotient de deux variables aléatoires indépendantes.

3 Annexes

Par soucis de lisibilité, la majorité des commentaires ont été retirés. Ils sont bien entendu toujours disponibles dans les fichiers MATLAB contenant les fonctions.

Annexe A

3.1 Q1d.m

```
function [Prob, t] = Q1d( N, Display )
% Q1d
%   Détermine la probabilité pour qu'au min 2 personnes parmi N  aient leur
%   anniversaire le même jour

if nargin<2
    Display = 1;
end

Length = length(N); % Plus rapide et recommandé pour les boucles
t = zeros(1,Length);
Prob = ones(1,Length);

for i = 1:Length
    tic;
    for j = 1:N(i)-1
        % Prob tous différents
        Prob(i) = Prob(i) * (365 - j)/365;
    end

    % Prob meme jour
    Prob(i) = 1 - Prob(i);
    t(i) = toc;
end
if Display == 1
    disp(['Solution : ' num2str(Prob)]);
    disp(['Temps (s) : ' num2str(t)]);
end
end
```

Annexe B

3.2 Q2b.m

```
function [Solution] = Q2b( N )
%Q2B fonction qui renvoit l'espérance et la variance de vecteurs binaires

Length = length(N);
Solution = zeros(Length,2);
```

```
fprintf('\t\t\t Espérance: \t Variance:\n' );
for i =1:Length
    tab = zeros(1,N(i));
    for j=1:N(i)
        tab(j) = birthday40;
    end
    % Calcul de l'espérance via mean
    Solution(i, 1) = mean(tab); % Espérance pour un vecteur de taille N(i)

    % Calcul de la variance via var
    Solution(i,2) = var(tab,1);% Variance pour un vecteur de taille N(i)

    fprintf('N = %d :\t\t', N(i));
    fprintf('%s \t\t %s \n', num2str(Solution(i,1)), num2str(Solution(i,2)));
end
end
```

3.3 Q2c.m

```
function [ Sol ] = Q2c( N, Display)
%Q2.C Fonction qui calcule la moyenne (variance) de l'espérance et la moyenne
%(variance) de la variance.

if(nargin<2)
    Display = 1;
end

NBRE_EXP = 1000;
MeanVect = zeros(1,NBRE_EXP);
VarVect = zeros(1,NBRE_EXP);
Length = length(N);
Sol = zeros(Length, 4);
tic;

% Pour chaque dimension de vecteur dans N:
for i = 1:Length
    Array = zeros (1,N(i));
    % Pour une moyenne sur NBRE_EXP
    for j = 1 : NBRE_EXP
        % Pour chaque élément du vect de dim N(i)
        for k = 1 : N(i)
            Array(k) = birthday40;
        end

        MeanVect(j) = mean(Array);

        VarVect(j) = var(Array,1);
    end
    % Moyenne de l'espérance
    Sol(i,1) = mean(MeanVect);

    % Variance de l'espérance
```

```
    Sol(i,2) = var(MeanVect,1);

    % Moyenne de la variance
    Sol(i,3) = mean(VarVect);

    % Variance de la variance
    Sol(i,4) = var(VarVect,1);
    t = toc;
    if Display == 1
        disp(['N = ' num2str(N(i)) ' : [' num2str(Sol(i,:)) '] ; t = ' num2str(t) ' s']);
    end
end
end
```

4 Q2d.m

```
function [Sol] = Q2d( N, Display)
%Q2D Fonction qui renvoit la moyenne ainsi que la variance des 1000 estimateurs de l'espérance de Y.

if nargin < 2
    Display = 1;
end

NBRE_EXP = 1000;
MU = 10;
SIGMA = 5^2;
MeanRatio = zeros(1, NBRE_EXP);
Length = length(N);
Sol = zeros(Length, 2);
% Pour chaque vecteur dont la dim est fournie en argument
for i=1:Length
    % Pour un certain NBRE_EXP
    for j=1:NBRE_EXP

        V1 = zeros(1,N(i));
        V2 = V1;
        % Pour chaque élément du vecteur étudié
        for k=1:N(i)

            % Retard ami 1
            V1(k)=normrnd(MU,SIGMA);

            % Retard ami 2
            V2(k)=normrnd(MU,SIGMA);
        end
    % Rapport (ratio) composante par composante des retards
    Ratio=(V1)./(V2);

    % Moyenne des rapports (1 par expérience)
    MeanRatio(j)=mean(Ratio);

    end
```

```
    % Espérance
    Sol(i,1) = mean(MeanRatio);

    % Variance
    Sol(i,2) = var(MeanRatio,1);
    t = toc;
    if Display == 1
        disp(['N = ' num2str(N(i)) ' : Esperance = ' num2str(Sol(i,1)) ' ; Variance = ' num2str(Sol(i,2)
    end
end
end
```