Probabilités et statistiques
Université Clermont Auvergne
IUT Info 2A, 2021-2022

A. Khaldi
C. Samir
A. Wohrer

Projet TP : BATAILLE !

Objectifs

- Découvrir la simulation numérique de probabilités à travers un exemple concret.
- Appliquer les méthodes du cours pour calculer une moyenne, une médiane, une fonction de répartition...
- Faire preuve de créativité.
- Rédiger un compte-rendu expérimental.

Consignes

- Même face à votre ordinateur, travaillez toujours avec une feuille de papier et un crayon !
- Ce Projet TP est prévu pour se dérouler sur 4 séances.
- Vous devez rendre un rapport écrit, ainsi que votre code bien commenté.
- Ce projet compte pour **7 points** dans votre note finale.

1 Règle du jeu

La bataille est le plus connu, le plus simple, et le plus ennuyeux, de tous les jeux de cartes.

- Chaque joueur dispose de la moitié d'un paquet de 52 cartes, distribuées au hasard.
- À chaque tour, les joueurs retournent la carte du haut de leur paquet.
- Si l'une des cartes est plus forte que l'autre, le gagnant empoche les deux cartes mises en jeu, et les place à la fin de son paquet.
- Si les deux cartes sont de la même force, il y a **bataille**. Chaque joueur met en jeu la carte suivante de son paquet, sans la retourner, puis retourne la carte encore suivante. Le gagnant empoche les 6 cartes ainsi mises en jeu, sauf en cas de nouvelle égalité, qui entraîne une 'surbataille' (et donc, 10 cartes en jeu), etc.
- La partie s'arrête lorsqu'un joueur a récupéré la totalité des cartes.

Dans ce projet TP, vous devrez

1. Implémenter informatiquement une **simulation** d'une partie de bataille.
2. Une fois le simulateur mis en place, faire des **statistiques** sur un grand nombre de parties, afin de déterminer la longueur typique d'une partie, les probabilités de victoire, etc.

2 Implémentation du simulateur

Représentation des cartes. Afin de simplifier, la valeur d'une carte est représentée par un nombre de 1 à 13, le nombre 13 étant la carte la plus forte (c'est-à-dire l'As dans un jeu classique), et le 1 la carte la plus faible (c'est-à-dire le 2 dans un jeu classique).

Un jeu total possède 4 répétitions de chaque carte, pour un total de 52 cartes. La couleur n'étant pas importante à la bataille, il est inutile de la coder informatiquement.

2.1 Distribution des cartes

D'abord, implémentez une fonction Scilab permettant de générer une distribution au hasard des cartes entre les 2 joueurs :

```
[jeu1, jeu2] = distribue()
```

Cette fonction doit renvoyer deux listes, représentant respectivement le jeu du joueur 1 et du joueur 2. Par exemple, une sortie possible serait

```
jeu1 = [12 2 5 5 7 ...]
jeu2 = [9 3 2 13 1 ...]
```

Ainsi, dans le paquet du joueur 1, la carte du haut est un 12, la suivante est un 2, etc.

Fonctionnement normal. En mode normal, `jeu1` et `jeu2` doivent chacun contenir 26 nombres, et l'union de ces deux listes doit former un jeu complet, dans lequel chaque nombre de 1 à 13 est présent 4 fois.

Fonctionnement "de test". Dans un second temps, vous pouvez éventuellement rajouter des options "de test" à votre fonction `distribue`. Par exemple, si on le souhaite, la fonction pourrait attribuer un nombre différent de cartes au joueur 1 et au joueur 2. Ou encore ne pas distribuer toutes les cartes... Tout dépendra des tests que vous souhaiterez effectuer au moment de votre étude statistique (partie 3).

2.2 Simulateur de bataille

Ensuite, implémentez la bataille proprement dite, dans une fonction Scilab avec la signature suivante :

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

Dans cette signature, `jeu1` et `jeu2` sont les listes ordonnées représentant le jeu initial de chaque joueur, au moment de commencer la partie.

À partir de ces jeux initiaux, votre fonction doit simuler l'intégralité d'une partie de bataille, jusqu'à la victoire de l'un des deux joueurs.

La sortie `gagnant` indique l'identité du joueur gagnant (1 ou 2).

La sortie `temps` indique combien de tours a duré la partie.

Voici quelques détails qui pourront avoir leur importance au moment d'implémenter la fonction `bataille` :

Bataille simplifiée. Vous allez vite vous apercevoir que ce sont les *batailles* qui nécessitent le plus de réflexion afin d'être codées élégamment.

Si vous le souhaitez, vous pouvez commencer par coder une bataille simplifiée, avec la règle suivante : *en cas de bataille, on tire simplement au hasard lequel des joueurs empoche les deux cartes.*

Vous préciserez bien, dans votre compte rendu, si vous avez implémenté une bataille exacte ou une bataille simplifiée. (Si vous le souhaitez, vous pouvez également implémenter les deux règles, et les comparer entre elles...)

Ordre de remise des cartes. Lorsqu'un joueur gagne un tour, il repose les deux cartes qu'il a gagnées à la fin de son paquet. Mais dans quel ordre les repose-t'il ? D'abord sa carte, puis celle de l'adversaire ? L'inverse ? Au hasard ?

Étonnamment, cette précision, qui semble être un point de détail, a une influence réelle sur le déroulement des parties. Lorsque les deux joueurs appliquent systématiquement des règles de remise "opposées", on peut même voir parfois apparaître des parties qui ne terminent jamais ! Si jamais vous observez ce phénomène, vous pouvez l'étudier et essayer de le comprendre... ou bien juste trouver la solution pour l'éviter. :)

3 Étude probabiliste et statistique

Une fois votre code de simulation implémenté, commence la partie réellement liée à ce cours. Vous allez devoir **établir les propriétés (probabilistes) du jeu de bataille**. Ceci, de manière purement empirique, en simulant un grand nombre de parties.

Votre étude est libre. Vous devez vous poser des questions concernant le jeu de bataille, et simuler un nombre suffisant de parties pour établir une réponse quantitative à chaque question.

Voici quelques exemples –non exhaustifs– de questions que l'on peut se poser. Elles sont volontairement vagues, afin de stimuler votre créativité.

- Quelle est la durée moyenne d'une partie ? La partie peut-elle être beaucoup plus courte, ou beaucoup plus longue, suivant les cas ?
- En inspectant rapidement le jeu initial des deux joueurs, peut-on prédire une forme de "force", c'est-à-dire, lequel des deux a le plus de chances de gagner ?
- Lorsqu'un joueur est en situation défavorable (si tant est qu'on sache quantifier ce que cela signifie), quelle est sa probabilité de gagner malgré tout ?
- La règle de 'bataille simplifiée', exposée plus haut, change-t'elle beaucoup les propriétés probabilistes du jeu ?
- Pouvez-vous vérifier, avec des méthodes statistiques, que votre fonction `distribution` est bien implémentée, c'est-à-dire qu'elle distribue bien les cartes des deux joueurs totalement au hasard ?
- etc.

Une fois posée une question, vous devrez y répondre de manière aussi quantitative que possible, à l'aide de graphes et de calculs dont vous expliquerez la signification précise (si possible dans un langage probabiliste rigoureux).

Vous êtes encouragés à appliquer autant d'éléments de cours que possible. À vous de faire un effort d'imagination pour "caser" dans votre étude le plus grand nombre possible de techniques et de théories vues en cours !

4 Rendu

Vous devrez déposer 3 fichiers sur l'ENT, à la page du cours 'Probas et Stats' correspondant au projet. La date limite de soumission est le *dimanche 8 novembre*.

Le premier fichier, avec un nom du genre

`bataille_fonctions.sci`

contiendra vos fonctions Scilab permettant d'implémenter UNE partie de bataille. Le code devra être **commenté avec soin**.

Le second fichier, avec un nom du genre

`etude_bataille.sce`

contiendra les commandes Scilab que vous avez appliquées afin de mener à bien l'étude statistique et probabiliste du jeu de bataille. Ce fichier commencera par charger les fonctions contenues dans le fichier `bataille_fonctions.sci`, avant de les appeler un grand nombre de fois pour réaliser votre étude.

Remarque : Il est possible que, afin d'établir des probabilités précises, vous deviez effectuer des calculs relativement longs (de l'ordre de plusieurs minutes). Dans ce cas, merci d'utiliser un plus petit nombre d'essais dans le code que vous nous rendrez, afin que nous puissions tester votre code rapidement. Par contre, votre rapport peut utiliser les résultats de calculs plus longs et plus précis.

Le troisième fichier, avec un nom du genre

`rapport_nom_prenom.pdf`

sera votre rapport proprement dit, au format pdf. Vous y exposerez, aussi clairement que possible, les conclusions de votre étude statistique :

- Quelles questions vous vous êtes posées.
- Pour chaque question posée, quelles grandeurs probabilistes permettent de répondre à cette question, et quel a été le résultat.
- L'usage de graphiques est bienvenu, lorsqu'il semble utile.

5 Évaluation

Ce projet étant noté, votre encadrant ne peut pas le faire à votre place. Vous pouvez bien sûr intéragir avec lui, lui poser des questions, y compris techniques... Mais les idées créatrices devront émaner de vous, et non de lui.

L'ensemble du projet est noté sur 7 points. Votre notation sera basée sur les élements suivants :

- Qualité du rapport (clarté, précision du vocabulaire probabiliste et statistique, rédaction...)
- Qualité de votre implémentation (efficacité, commentaires). À ce propos, il est évident que les fichiers que vous soumettez doivent s'exécuter sans erreur !
- Votre implication générale sur l'ensemble des séances de TP.