Simulation et étude probabiliste du Black Jack

Anas El Benna et Luana Timofte

19 février 2019

Introduction

Bienvenue dans l'univers des jeux de cartes ! Tentez-vous votre chance ? Croyez-vous pouvoir gagner ? Nous avons préparé un simple jeu ! Voici les règles : Vous allez jouer contre un croupier et le but du jeu c'est d'avoir la somme des cartes entre 17 et 21, MAIS aussi qu'elle soit plus grande que la somme des cartes du croupier ! Sinon, vous allez perdre, bien entendu !

Notre petite fonction fonctionne bien, mais il faut, avant tout, introduire les paramètres ! Vous êtes un peu comme « the game master ». On vous permet de choisir comment jouer, quels paramètres mettre (comme la limite du croupier, des joueurs, le nombre de joueurs), ainsi que choisir le jeu en mode automatique ou manuel.

Si vous voulez le mode « **AUTOMATIQUE** », c'est-à-dire que l'ordinateur joue à votre place (eh oui, c'est possible, whoauw !), vous n'avez qu'à remplir tous les paramètres avec la *limite du croupier*, *du joueur*, et le *nombre de joueurs* (exemple : playBlackJack [17,17, 5]).

Si vous voulez le mode « **MANUEL** », vous n'allez pas mettre une limite au joueur, c'est à vous de choisir, sur place (et sans pression, bien sûr), quand vous voulez vous arrêter. Donc vous n'avez qu'à compléter avec la limite du croupier (qui est 17, bien entendu, dans tous les casinos du monde et de l'univers entier) et le nombre de joueurs (exemple : playBlackJack [17, ,5]). Mais si vous avez envie de défier les lois de la nature et changer cette limite du croupier, alors à vous de voir!

À vous de trouver la stratégie optimale et gagner (ou pas)!

Black Jack

Le fameux code

```
playBlackJack <- function(limite.croupier,limite.joueur,n, affichage = TRUE){

  #vecteur avec tous les points totaux du joueurs
  point.joueur <- rep(NA,n)

  #Quelques fonctions bien utiles

  #Calcule les points pour le(s) joueur(s) et pour le croupier
  calculPoint <- function(mains, joueur) {

    main <- mains[,joueur]
    main[main %in% c("Valet", "Dame", "Roi")] <- 10
    sub.main <- as.numeric(main[main != "As"])
    sub.tot <- sum(sub.main, na.rm = TRUE)

    main[main == "As"] <- ifelse(sub.tot > 10, 1, 11)

    return(sum(as.numeric(main), na.rm = TRUE))

  }

  #Affiche les points du croupier et du/des joueur(s), dans un tableau
  affichagePoints <- function(){

    matrice.pts <- matrix(nrow=2, ncol=(n + 1))
    colnames(matrice.pts) <- c("Croupier", paste("Joueur", 1:n))
    rownames(matrice.pts) <- c("Somme totale des points","Status")
```

```
  matrice.pts[1,1] <- nb.point  #on indexe les points du croupier
  matrice.pts[1,(2:(n+1))] <- point.joueur #on indexe les points du joueur
  matrice.pts[2,(2:(n+1))] <- Status() #on indexe le status
  matrice.pts[2,1] <- "Banque"
  print(matrice.pts)
}

#Donne un status a chaque joueur
Status <- function(){

 #prend le status des joueurs,
  # par rapport a la banque (gagnant, perdant, egalite)
 status <- rep(NA,n)

  for (i in 1:n){
    if((point.joueur[i] > nb.point & point.joueur[i] <= 21)
       | (nb.point > 21 & point.joueur[i] <= 21)){
      status[i] <- "Gagnant"
    } else if(point.joueur[i] == nb.point & point.joueur[i] <= 21){
      status[i] <- "Egalite"
    } else{
      status[i] <- "Perdant" }
  }
 return(status)
}


#Une petite fonction qui affiche le jeu
afficheJeu <- function()
  print(mains[1:tour,])


# Fini avec les fonctions, voici d'autres elements :

jeu <- rep(c("As", 2:10, "Valet", "Dame", "Roi"), 4)
jeu <- sample(jeu, 52)

mains <- matrix(NA, 52, n+1)
colnames(mains) <- c("Croupier", paste("Joueur", 1:n))

#la premiere distribution des 2 cartes au croupier et n joueurs
mains[1,] <- jeu[1:(n+1)] #distribution 1ere carte
jeu <- jeu[-(1:(n+1))]
mains[2,] <- jeu[1:(n+1)]  #distribution 2eme carte
jeu <- jeu[-(1:(n+1))]

tour <- 2

if (affichage)
  afficheJeu()

# Commencons a jouer - LET THE GAME BEGIN
```

```
# Ici c'est le mode MANUEL
if (missing(limite.joueur)) {

  cat("Vous etes passés en mode MANUEL.
      C'est a vous de choisir les strategies de chaque joueur!")

  for (i in 1:n) #i c'est le joueur
  {
    tour <- 3

    while (TRUE){
      veuxCarte <- readline("Prenez-vous une carte ? (Carte / Non) : ")

      point.joueur[i] <- calculPoint(mains, (i+1))

      if (veuxCarte == "Non") {
        break
      } else if (veuxCarte == "Carte") {

        point.joueur[i] <- calculPoint(mains, (i+1))

        #distribution de carte au joueur i
        if (point.joueur[i] <= 21) {
          mains[tour, (i+1)] <- jeu[1]
          jeu <- jeu[-1]
          tour <- tour + 1
        } else {
          cat("Tu as perdu. On passe au prochain joueur.")
          break
        }

        afficheJeu()
      } else {
        cat("Mauvaise reponse. Re-essayez : Carte/Non")
      }
    }
  }
  tour.save <- tour

  tour <- 3

  # Le croupier joue
  while (TRUE){
    nb.point <- calculPoint(mains, 1)

    if (nb.point < limite.croupier){
      mains[tour, 1] <- jeu[1]
      jeu <- jeu[-1]
      tour <- tour + 1

      afficheJeu()
    } else
      break
```

```
    }

    mains <- mains[1:max(tour.save, tour),]

    Status()
    affichagePoints()

# Ici c'est le mode AUTOMATIQUE

  } else {
    for (i in 1:n)
    {

      tour <- 3
      while (TRUE) {

        point.joueur[i] <- calculPoint(mains, (i+1))

        if (point.joueur[i] < limite.joueur) {
          mains[tour, (i + 1)] <- jeu[1]
          jeu <- jeu[-1]
          tour <- tour + 1
        } else if (point.joueur[i] > 21) {
          break
        } else {
          break
        }


        if (affichage)
          afficheJeu()
      }
    }

    tour.save <- tour

    # Le croupier joue
    tour <- 3

    while (TRUE){
      nb.point <- calculPoint(mains, 1)

      if (nb.point < limite.croupier){
        mains[tour, 1] <- jeu[1]
        jeu <- jeu[-1]
        tour <- tour + 1



        if (affichage)
          afficheJeu()

      } else
```

```
        break
    }

    mains <- mains[1:max(tour.save, tour),]


    if (affichage){
      Status()
       affichagePoints()}

  }

  invisible(Status())
  # ca tourne le Status en mode invisible
   # pour pouvoir l'utiliser dans le Monte Carlo
}
```

Ce qu'on obtient, une fois le code compilé et executé :

Mode manuel :

Voici un tout petit exemple du mode **"Manuel"**, où l'utilisateur choisit lui-même de prendre une carte ou pas, à chaque tour. Nous observons bien les étapes en fonction de chaque choix du joueur, ainsi qu'une sortie montrant leurs points totaux et leur status **("Gagnant", "Perdant", "Egalité")**, par rapport au croupier. Pas mal, hein ! Mais malheureusement, aucun gagnant **:'(**.

```
> playBlackJack(17,,3)
     Croupier Joueur 1 Joueur 2 Joueur 3
[1,] "10"     "Valet"  "Dame"   "2"
[2,] "3"      "7"      "Roi"    "8"
Vous etes passés en mode MANUEL.
        C'est a vous de choisir les strategies de chaque joueur!
Prenez-vous une carte ? (Carte / Non) : Non
Prenez-vous une carte ? (Carte / Non) : Non
Prenez-vous une carte ? (Carte / Non) : Carte
     Croupier Joueur 1 Joueur 2 Joueur 3
[1,] "10"     "Valet"  "Dame"   "2"
[2,] "3"      "7"      "Roi"    "8"
[3,] NA       NA       NA       "Roi"
[4,] NA       NA       NA       NA
Prenez-vous une carte ? (Carte / Non) : Non
     Croupier Joueur 1 Joueur 2 Joueur 3
[1,] "10"     "Valet"  "Dame"   "2"
[2,] "3"      "7"      "Roi"    "8"
[3,] "7"      NA       NA       "Roi"
[4,] NA       NA       NA       NA
                      Croupier Joueur 1  Joueur 2  Joueur 3
Somme totale des points "20"     "17"      "20"      "20"
Status                  "Banque" "Perdant" "Egalite" "Egalite"
```

[h]

Mode automatique :

Voici un autre exemple, mais cette fois de sortie **"Automatique"**. Pour cela, nous avons choisi 17 comme limite du croupier et du joueur, ainsi que 3 joueurs. La fonction $affficheJeu$ nous montre à chaque fois le jeu, en fonction des choix des joueurs, c'est à dire : *CARTE* ou *NON*. Comme nous pouvons le voir, ici nous n'avons qu'un seul gagnant (quel chanceux !).

```
> playBlackJack(17,17,3)
     Croupier Joueur 1 Joueur 2 Joueur 3
[1,] "Valet"  "Valet"  "Roi"    "2"
[2,] "Dame"   "7"      "3"      "3"
     Croupier Joueur 1 Joueur 2 Joueur 3
[1,] "Valet"  "Valet"  "Roi"    "2"
[2,] "Dame"   "7"      "3"      "3"
[3,] NA       NA       "Dame"   NA
[4,] NA       NA       NA       NA
     Croupier Joueur 1 Joueur 2 Joueur 3
[1,] "Valet"  "Valet"  "Roi"    "2"
[2,] "Dame"   "7"      "3"      "3"
[3,] NA       NA       "Dame"   "5"
[4,] NA       NA       NA       NA
     Croupier Joueur 1 Joueur 2 Joueur 3
[1,] "Valet"  "Valet"  "Roi"    "2"
[2,] "Dame"   "7"      "3"      "3"
[3,] NA       NA       "Dame"   "5"
[4,] NA       NA       NA       "As"
[5,] NA       NA       NA       NA
                      Croupier Joueur 1  Joueur 2  Joueur 3
Somme totale des points "20"     "17"      "23"      "21"
Status                  "Banque" "Perdant" "Perdant" "Gagnant"
>
```

[h]

Monte Carlo

Pour voir si nos strategies sont optimales ou pas et si nos gains sont significatifs, nous avons crée un code pour comparer tout cela. C'est-à-dire, nous avons compilé cet algorithme, automatiquement, un grand nombre de fois. Ensuite, nous avons crée des histogrammes, pour montrer les probabilités des gains.

Un peu de théorie: la méthode Monte Carlo consiste à estimer $E(h(X))$, à partir d'un échantillon $X = (X_1, ..., X_n)$. Pour cela, la loi des grands nombres utilise la moyenne empirique pour construire :

$$E(h(X)) = \frac{\sum_i^n h(X_i)}{n}$$

Dans notre étude, la fonction $h(X)$ que l'on souhaite estimer est l'indicatrice de notre jeu de données. La formule de Monte Carlo n'est alors que la probabilité de X.

Le (un peu moins) fameux code

```
MonteCarlo <- function(nb.strategies,essais){

# nb.strategies = le nb de strategies que vous allez tester
# essais = le nombre de fois que vous allez tourner le code, par strategie
# matrices qui donnent le nombre total de gains et les probabilites, par strategie

  Comptage <- matrix (0, nrow = nb.strategies, ncol = 4)
  colnames(Comptage) <- c(" Strategie ","Egalite","Gagnant","Perdant")

  matriceProba <- matrix (0, nrow = nb.strategies, ncol = 4)
  colnames(matriceProba) <- c("Strategie","Egalite","Gagnant","Perdant")


  for (i in 1:nb.strategies){
    lim.jou <- as.numeric(readline("Quelle limite pour le joueur? (exemple: 17):"))

    #la premiere colonne comprend les stratgies
    Comptage[i,1] <- lim.jou

    VecteurGains <- rep(NA,essais)

    for (j in 1:essais){

      VecteurGains[j] <- playBlackJack(17,lim.jou,1, affichage = FALSE)
    }

    # Vecteur qui ordonne les gains
    sorted.vectgains <- as.vector(sort(t(table(VecteurGains)) ,
                                       ord = 0 , decreasing = FALSE ))
    if ( length(sorted.vectgains) == 2) {
    Comptage[i,3:4] <- sorted.vectgains
    } else {
    Comptage[i,2:4] <- sorted.vectgains

    }
  }

  rownames(Comptage) <- paste(Comptage[,1])
```

```
  rownames(matriceProba) <- paste(Comptage[,1])


   matriceProba[1:nb.strategies, 2:4] <- Comptage[,2:4]/essais
   matriceProba[,1] <- Comptage[1:nb.strategies, 1]

   print("Voici le nombre de gains en fonction de chaque strategie:")
   prmatrix(Comptage , rowlab=rep("", nb.strategies))
   print("Voici la probabilité de gain en fonction de chaque strategie:")
   prmatrix(matriceProba , rowlab=rep("", nb.strategies))


   barplot(t(matriceProba[,2:4]), ylim = c(0, 0.1 + max(matriceProba[,2:4]))
           ,main = "Probabilite de gain, par strategie (Monte Carlo)",
           xlab="Strategies choisies", col=c("white","green","red"),
           beside=TRUE)
   legend ("topleft",c("Egalite", "Gagnant" , "Perdant"), fill =c("white","green","red"))

}
```

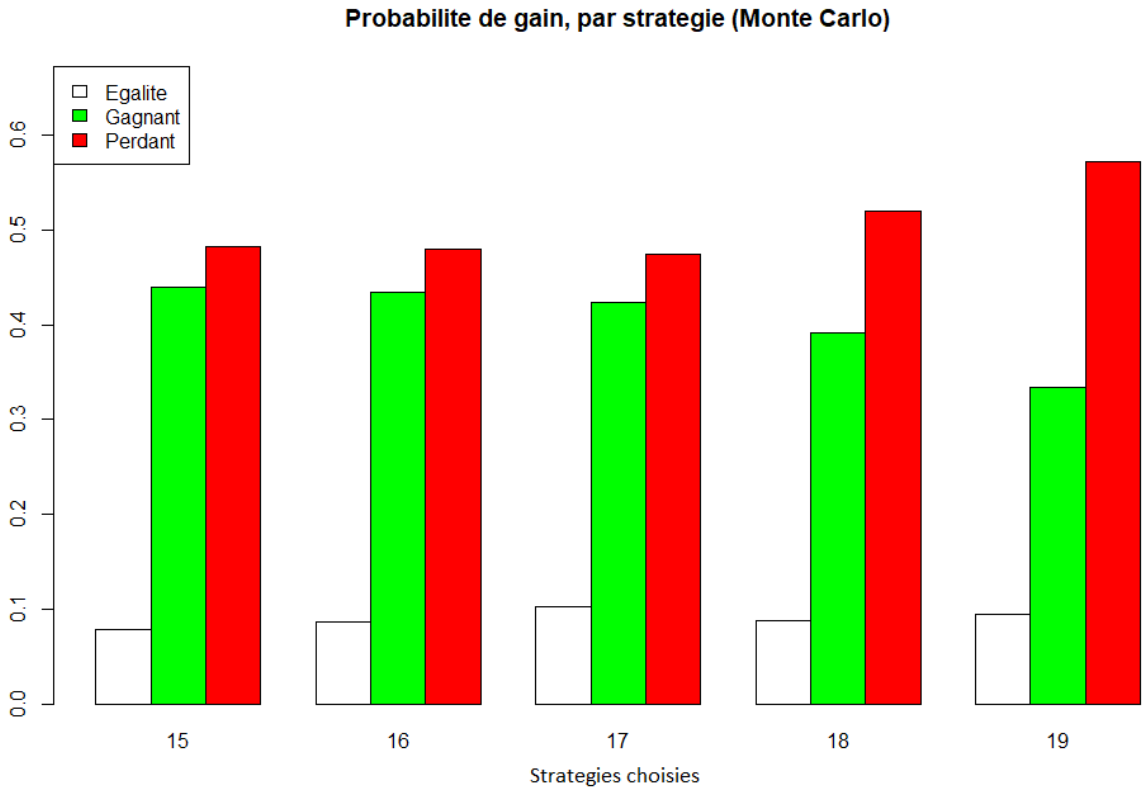
Interprétation

Les résultats obtenus sont les suivants, classés dans 2 matrices, une qui montre le nombre de gains et l'autre qui montre la probabilité des gains, pour 1000 essais et 5 strategies:

```
[1] "Voici le nombre de gains en fonction de chaque strategie:"
  Strategie  Egalite Gagnant Perdant
          15      78     439     483
          16      86     434     480
          17     103     423     474
          18      88     392     520
          19      94     334     572
[1] "Voici la probabilité de gain en fonction de chaque strategie:"
 Strategie Egalite Gagnant Perdant
        15   0.078   0.439   0.483
        16   0.086   0.434   0.480
        17   0.103   0.423   0.474
        18   0.088   0.392   0.520
        19   0.094   0.334   0.572
```

[h]

Voici un histogramme.



[h]

Mais qu'en est-il du gain des joueurs, si le croupier change de stratégie ?? Si nous changeons la limite du croupier et nous faisons encore les mêmes essais avec différentes stratégies pour les joueurs, nous observons encore que les chances de perdre, des joueurs, sont plus grandes que les chances de gagner...

Et oui, malheureusement, on ne gagne pas autant qu'on l'aurait cru... Peu importe la stratégie des joueurs et la mauvaise stratégie du croupier (quoique, nous savons tous que le croupier n'a jamais de "mauvaise stratégie", *the house always wins*), les joueurs sont bel et bien perdants... Stay safe, kids, don't gamble !

Conclusion

Comme nous l'avons vu, la probabilité de perdre est plus grande que celle de gagner, malgré une toute petite différence, si nous choisissons comme strategie de jeu de s'arreter à un certain point. Reste à vérifier si la stratégie de comptage de cartes (certe interdite dans les casinos) marche mieux. La meilleure conclusion à tirer est qu'il ne faut pas mettre son argent dans les jeux de cartes. **"Don't try this at home"** !