Université Paris Descartes
UFR de Mathématiques et Informatique

Probabilités et Statistiques pour l'informatique- Licence MIA 2e année

Travaux Pratiques - 1ère séance

Le but de cette première séance est de découvrir le logiciel "R" et de l'utiliser pour simuler des variables aléatoires discrètes. Ce logiciel, principalement dévolu aux applications statistiques, est libre et gratuit, disponible pour Linux, Windows et MacOS sur le site de référence `http//cran.r-project.org`.

1 Premiers pas avec Rstudio

"Rstudio" est un environnement de développement adapté à "R". Il est également libre et gratuit, et disponible sur les systèmes d'exploitation les plus courants (cf `http//rstudio.org`).

Lancer "Rstudio" depuis le menu des "Applications", section "Programmation"

L'interface présente quatre fenêtres, deux par colonne, chacune pouvant être déployée sur l'ensemble de la colonne, n'en occuper qu'une partie ou être repliée. La largeur de chaque colonne est modifiable, et peut occuper tout l'espace. À la première ouverture, les quatre fenêtres affichées sont :

- en bas à gauche : la fenêtre "Console" ; elle permet d'utiliser "R" en mode interactif ;
- en haut à gauche : la fenêtre "Source" ; c'est un éditeur intégré (pour la faire apparaître, replier la fenêtre "console") ; il est possible d'éditer un ou plusieurs programmes simultanément et de les faire exécuter par "R" ;
- en haut, à droite : la fenêtre "Workspace/History" ; dans l'onglet "Workspace" apparaîtront les variables et les fonctions qui ont été définies ; dans l'onglet "History" apparaîtront les commandes qui ont été exécutées ;
- en bas à droite : la fenêtre de navigation ; elle permet d'accéder aux fichiers, aux graphiques produits par "R", aux bibliothèques qui permettent d'enrichir "R" en nouvelles fonctions, et à l'aide intégrée.

2 Découvrir le mode interactif de "R"

Cliquer sur la fenêtre "Console" pour la rendre active

Un texte d'introduction s'affiche, suivi du symbole > puis du curseur clignotant. C'est le prompt du logiciel ; il indique que vous pouvez entrer les commandes ici.

2.1 Vecteurs

Depuis le prompt de "R", vous pouvez exécuter toutes les commandes déjà disponibles et même définir vos propres fonctions, mais avant d'en arriver là nous allons d'abord nous familiariser avec les commandes de base de "R".

La structure de données la plus simple sous "R" est le vecteur. Même un simple nombre est vu comme un vecteur de taille 1. Ainsi, que l'on tape `5` ou `pi` après l'invite, puis *[Entrée]*, le résultat affiché précise qu'il s'agit de la première coordonnée (`[1]`) du vecteur unidimensionnel 5 ou pi

```
> 5
[1] 5
> pi
[1] 3.141593
```

Un vecteur peut être à valeurs numériques ou entières, comme dans les exemples précédents. Il peut aussi être à valeurs alphanumériques

```
> "a"
[1] "a"
> "ab"
[1] "ab"
```

ou encore à valeurs booléennes

```
> 0<1
[1] TRUE
> 0==1
[1] FALSE
```

Ouvrez la fenêtre "History" : vous y verrez toutes les commandes que vous avez exécutées. En double-cliquant sur l'une d'entre elles, elle réapparaîtra dans la console.

Pour rappeler une commande précédente dans la console, vous pouvez aussi utiliser les flèches verticales qui dirigent le curseur.

Pour construire un vecteur, on peut procéder par concaténation en utilisant la fonction `c` :

```
> c(2,3,5)
[1] 2 3 5
> c(2,3,5)[1]
[1] 2
> c(2,3,5)[2]
[1] 3
> c(2,3,5)[c(1,2)]
[1] 2 3
> c(2,3,5)[-1]
[1] 3 5
> c(2,3,5)[-2]
[1] 2 5
```

On peut aussi construire un vecteur numérique en utilisant la commande `m:n`

```
> 1:30
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
[26] 26 27 28 29 30
> 10:-10
 [1]  10   9   8   7   6   5   4   3   2   1   0  -1  -2  -3  -4  -5  -6  -7  -8
[20]  -9 -10
> c(1:2,4)
[1] 1 2 4
```

On peut additionner deux vecteurs numériques de même longueur, additionner un vecteur et un scalaire (vecteur de longueur 1), multiplier deux vecteurs coordonnée par coordonnée, multiplier un vecteur par un scalaire, etc.

```
> 1:3 + 1:3
[1] 2 4 6
> 1:3 + 3
[1] 4 5 6
> 1:3 * 1:3
[1] 1 4 9
> 1:3 * 3
[1] 3 6 9
> 1:3 == 3:1
[1] FALSE  TRUE FALSE
> 1:3 == 3
[1] FALSE FALSE  TRUE
```

On peut affecter un vecteur à une variable grâce à l'instruction <- ou =

```
> a=3
> a
[1] 3
> a=a+2
> a
[1] 5
> alphabet<-c("a","b","c")
> alphabet
[1] "a" "b" "c"
> alphabet[2]
[1] "b"
> alphabet[2]<-"z"
> alphabet
[1] "a" "z" "c"
```

Dans ce dernier exemple, le vecteur doit avoir été défini ou déclaré auparavant

```
> chiffre[1]<-5
Erreur dans chiffre[1] <- 5 : objet 'chiffre'  introuvable
> chiffre<-0
> chiffre[1]<-5
> chiffre
[1] 5
> chiffre[2]
[1] NA
> chiffre[4]<-2
> chiffre
[1]  5 NA NA  2
```

Ouvrez la fenêtre "Workspace" : vous y verrez les variables que vous avez définies dans la section "Values". En cliquant sur l'une d'entre elles, vous pouvez voir son contenu (s'il s'agit d'un vecteur de dimension supérieure à 2) et modifier sa valeur. Faites ainsi de `chiffre` *le vecteur* $(5, 4, 3, 2, 1)$.

2.2 Fonctions

Un grand nombre de fonctions sont disponibles sous "R", présentes dans le module de base ou dans des bibliothèques séparées. Elles peuvent avoir zéro,

un ou plusieurs arguments, et renvoient un vecteur ou un objet de structure plus élaborée. La syntaxe générale est *nom_de_fonction*`(arg1,arg2,...)`. L'une d'entre elles a déjà été vue : `c`. Voici quelques autres fonctions qui pourront être utiles :

— `seq` : cette fonction permet de construire des suites de nombres équidistants

```
> seq(0,1,length=11)
 [1] 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
> seq(0,1,by=.2)
[1] 0.0 0.2 0.4 0.6 0.8 1.0
> seq(0,1,by=.3)
[1] 0.0 0.3 0.6 0.9
```

— `rep` : cette fonction permet de construire des suites (de nombres, de lettres, de booléens,...) par répétition d'un même vecteur

```
> rep(1,5)
[1] 1 1 1 1 1
> rep(1:4,2)
[1] 1 2 3 4 1 2 3 4
> rep(1:4,each=2)
[1] 1 1 2 2 3 3 4 4
> rep(1:4,length=10)
 [1] 1 2 3 4 1 2 3 4 1 2
> rep(1:4, c(2,1,2,1))
[1] 1 1 2 3 3 4
> rep(c("encore","bravo"),3)
 [1] "encore" "bravo"  "encore" "bravo"  "encore" "bravo"
```

— `sort(v)` : cette fonction renvoie un vecteur dont les coordonnées sont celles de `v` après un tri croissant ou décroissant

```
> sort(4:1)
[1] 1 2 3 4
> sort(1:4, dec=TRUE)
[1] 4 3 2 1
> sort(c("c","a","b"))
[1] "a" "b" "c"
```

— `sum(v)` : cette fonction effectue la somme des coordonnées du vecteur `v` ; si ce vecteur est booléen, alors `sum(v)` est égal au nombre de ses coordonnées "vraies"

```
> sum(1:5)
[1] 15
> sum(1:2<2:1)
```

```
[1] 1
```

— `sample(v)` : cette fonction permet d'effectuer des tirages aléatoires uniformes dans l'ensemble des coordonnées du vecteur `v`, avec remise ou non

```
> sample(1:4)
[1] 3 1 2 4
> sample(1:4,1)
[1] 3
> sample(1:4,2)
[1] 2 4
> sample(1:4,5)
Erreur dans sample(length(x), size, replace, prob) :
        impossible de prendre un \'echantillon plus grand
que la population lorsque 'replace = FALSE'
> sample(1:4,5,replace=TRUE)
[1] 1 3 2 4 4
```

— `print(v)` : cette fonction permet d'afficher le vecteur `v` ; elle est utile dans une boucle ou lors de l'exécution d'un programme écrit dans un fichier

```
> print(pi)
[1] 3.141593
```

Astuce : la console (ainsi que l'éditeur) permet d'utiliser l'auto-complétion : pour introduire une commande de "R" ou rappeler une variable ou une fonction déjà introduite, il suffit de taper les premières lettres, puis sur la touche de tabulation, et "Rstudio" complète ou vous propose plusieurs choix. Essayez avec `pri`*.*

Exercice 1. *Écrire une commande qui renvoie le résultat*

1. *du lancer d'un dé*
2. *du lancer de 2 dés*
3. *de la somme des résultats du lancer de 2 dés*
4. *du tirage du loto (5 numéros parmi 49)*

Exercice 2. *Quelle est la loi de la variable aléatoire définie par la commande* `sum(sample(rep(0:1,c(5,10)),8,replace=TRUE))` *?*

Définir ses propres fonctions

Il est possible aussi de créer ses propres fonctions. La syntaxe générale est *nom_de_fonction*`<-function(arg1,arg2,...){...}`. Exemple :

```
> Un<-function(n){1:n/1:n}
> Un(3)
[1] 1 1 1
```

La même fonction écrite sur plusieurs lignes :

```
> Un<-function(n){
+ 1:n/1:n
+ }
>
```

Notez la présence de la fonction `Un` *dans la fenêtre "Workspace" section Functions.*

Remarques :

- S'il n'y a qu'une seule commande sur une seule ligne, les accolades sont superflues.
- La valeur retournée par la fonction sera toujours la valeur de la dernière instruction de la fonction.

Exercice 3. *Créer une fonction* `Urne` *à trois arguments* `k,p,q` *qui modélise le tirage sans remise de k boules dans un sac contenant p boules rouges et q boules noires. Indication : voici un exemple de sortie :*

```
> Urne(6,8,5)
[1] "Rouge" "Noire" "Noire" "Rouge" "Rouge" "Rouge"
```

3 Aller un peu plus loin avec "R"

3.1 Fichiers

Pour sauvegarder son travail ou élaborer un programme plus complexe, il est plus pratique d'écrire les lignes de code dans un fichier externe, plutôt que d'utiliser le mode interactif.

Créez un répertoire `TP-Proba` *à l'aide du navigateur de fichiers, puis allez dans le fenêtre "Source" ou ouvrez un nouveau fichier en cliquant sur [File>New>R Script]. Recopiez dans l'éditeur la fonction* `Urne`*, puis sauvegardez le fichier sous le nom* `Seance1.R`*, dans le répertoire précédent.*

Remarque :

- Il est traditionnel de donner aux *scripts* "R" l'extension `.R`, mais ce n'est pas obligatoire.
- Le symbole `#` permet d'écrire des commentaires : tout ce qui est écrit après un `#` ne sera pas interprété par "R".

Pour exécuter du code contenu dans un fichier, vous pouvez cliquer sur *[Run]* ou *[Re-run]* qui exécutera la ligne sur laquelle se trouve le curseur ou la région sélectionnée ; vous pouvez aussi exécuter tout le script contenu dans le fichier en cliquant sur *[Source]*.

Exercice 4. *Lorsqu'on effectue n tirages indépendants d'une même expérience aléatoire, on appelle fréquence empirique du résultat k le rapport entre le nombre de fois où k est tiré, et n. Exemple : on jette à 7 reprises un dé, avec pour résultats $1, 1, 5, 2, 6, 5, 3$; la fréquence empirique de 5 est $2/7$, celle de 4 est 0.*

Écrire une fonction `FreqEmp` *à un paramètre* `n` *qui renvoie la fréquence empirique de 5 lors de n tirages indépendants d'un dé à six faces. Comparer les fréquences empiriques pour $n = 10$, $n = 100$ puis $n = 1000$, avec la probabilité (théorique) de tirer un 5 lorsqu'on lance un dé.*

3.2 Aide et compléments

"R" est un logiciel très riche, et il n'est pas possible ni utile de le connaître dans sa totalité. Un très grand nombre d'extensions (dites "Packages") sont disponibles pour différents besoins : vous pouvez en parcourir une sélection grâce au navigateur intégré.

Ce navigateur vous permet aussi d'accéder à l'aide fournie avec "R" : soit en parcourant la documentation disponible depuis la page d'accueil ; soit en tapant directement le nom d'une commande dans l'invite de recherche. Dans ce dernier cas, le navigateur permet d'accéder à la description et à la syntaxe de la commande.

Exercice 5. *Trouvez une fonction qui permette de définir une matrice et utilisez-la pour construire la matrice* $\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$.

4 Un jeu de tirage au sort

Dans cette partie nous allons simuler un jeu de tirage au sort ; d'abord dans une version simple puis dans une version "cachée". Ces deux versions

constituent en fait des exemples basiques de chaînes de Markov et de chaînes de Markov à états cachées.

4.1 Le jeu des urnes - première version

On considère deux urnes A et B contenant des boules, elles-mêmes marquées a et b :

— l'urne A contient 10 boules marquées a et une boule b,

— l'urne B contient 15 boules b et une boule a.

Le jeu consiste à tirer successivement des boules dans les urnes, suivant le procédé suivant : on commence par tirer au sort une boule de l'urne A. Si cette boule est marquée a, on retirera dans l'urne A à l'étape suivante ; si elle est marquée b, on tirera dans l'urne B à l'étape suivante ; et ainsi de suite. Ainsi, à chaque étape, la lettre marquée sur la boule tirée indique ainsi quelle urne sera choisie à l'étape suivante. Les tirages se font avec remise : à chaque fois les boules sont replacées dans leurs urnes respectives, de sorte que les urnes A et B contiennent toujours les mêmes boules à chaque tirage.

A chaque étape du jeu on peut noter les événements suivants :

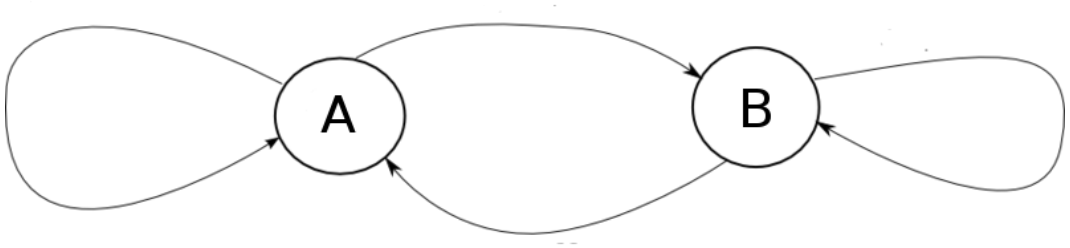
C_A ="On tire dans A à l'étape courante",

C_B ="On tire dans B à l'étape courante",

S_A ="On tire une boule a à l'étape courante"="On tire dans A à l'étape suivante",

S_B ="On tire une boule b à l'étape courante"="On tire dans B à l'étape suivante".

Ce jeu modélise une chaîne de Markov : chaque urne représente un état ; la marque de la boule donne la transition. On peut représenter ce jeu par le diagramme suivant :



Exercice 6. *Remplissez le tableau suivant avec les bonnes probabilités :*

	S_A	S_B
C_A	$P(S_A\|C_A) =$	$P(S_B\|C_A) =$
C_B	$P(S_A\|C_B) =$	$P(S_B\|C_B) =$

et compléter le diagramme en indiquant ces probabilités sur les flèches correspondantes. Exprimer $P(S_A)$ et $P(S_B)$ en fonction de $P(C_A)$ et $P(C_B)$, puis montrer que le vecteur colonne formé de $P(S_A)$ et $P(S_B)$ (vecteur d'état à l'étape suivante) se déduit du vecteur d'état à l'étape courante (donc formé de $P(C_A)$ et $P(C_B)$) par la multiplication d'une matrice.

Exercice 7. *Écrivez une fonction permettant de réaliser un tirage de n boules qui respecte les règles de transition définies ci-dessus et qui commence à partir de l'urne A. Réaliser plusieurs fois la simulation du jeu pour $n = 100$ et afficher la séquence des boules tirées. Qu'observe-t-on ?*

4.2 Le jeu des urnes - version "cachée"

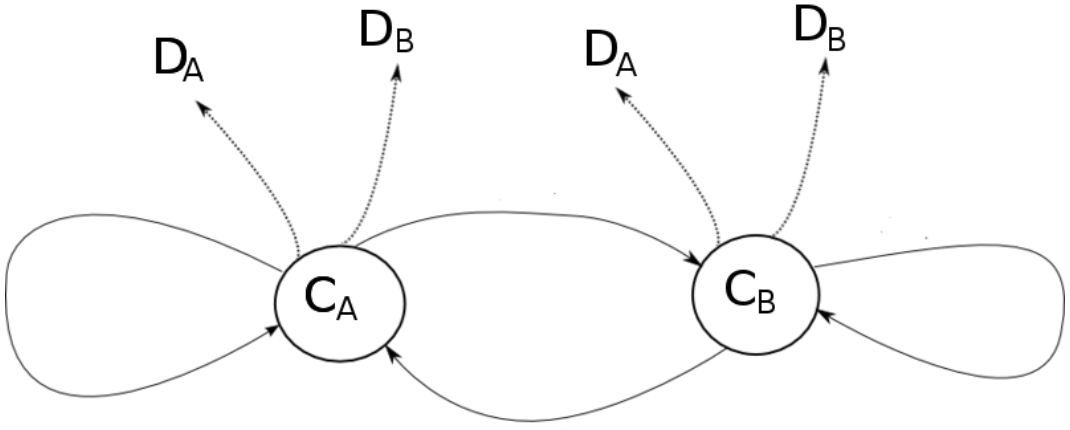
Supposons maintenant que le jeu nous est caché : les urnes et la personne tirant les boules sont derrière un rideau. La personne est censée nous annoncer à chaque fois dans quelle urne elle tire au sort, mais ment dans 10% des cas (elle dit alors le contraire : A au lieu de B, et inversement). On ne peut donc noter que la séquence des boules annoncée par la personne, et non la vraie séquence. On voudrait essayer de savoir quelles informations sur le jeu on peut découvrir à partir de la séquence annoncée : peut-on essayer de déceler lors de quels tirages la personne a probablement menti ? Peut-on retrouver la vraie séquence ? Peut-on connaître la répartition des boules a et b dans les urnes si on ne la connaissait pas au départ ? Ces questions d'*estimation* ne sont pas simples à traiter ; ici nous nous limiterons à simuler ce jeu et à observer les résultats.

Ce jeu constitue un exemple basique de chaîne de Markov à états cachés. Outre les événements C_A, C_B, S_A, S_B, on note à présent :

D_A ="La personne déclare tirer dans l'urne A à l'étape courante",

D_B ="La personne déclare tirer dans l'urne B à l'étape courante".

Le diagramme correspondant à ce jeu est le suivant :



Exercice 8. *Donner les valeurs des probabilités* $P(D_A|C_A)$, $P(D_B|C_A)$, $P(D_A|C_B)$ *et* $P(D_B|C_B)$, *et compléter le diagramme.*

Exercice 9. *Écrivez une fonction qui simule ce jeu et renvoie la séquence annoncée pour* n *tirages. Exécutez cette fonction pour* $n = 100$ *plusieurs fois et affichez les séquences obtenues. Qu'observe-t-on ?*