

Consignes pour les séances de TP

1. Au début de la première séance vous vous associerez en binômes et travaillerez **avec le même binôme** durant tout le semestre.
2. Pour chaque exercice vous écrirez un script R. Les scripts des exercices notés ♣ et ♠ seront à rendre à vos chargés de TP via la plateforme MyCourse. Il faudra donc vous inscrire à l'espace MyCourse correspondant à votre groupe. Le binôme constitué de M. Markov et M^{elle} Kovalevskaïa nommera `kovalevskaia_markov_exercice_n.R` le fichier correspondant à l'exercice n . Si il est validé chaque script rapporte un point (exercice ♣) ou deux points (exercice ♠) pour la note de TP.
 - Les exercices 5 et 6 sont à rendre avant la fin de la 4^{ème} séance de TP.
 - Les exercices numérotés de 10 à 14 sont à rendre avant la fin de la 8^{ème} séance de TP.
 - Les exercices numérotés de 15 à 18 sont à rendre avant la fin de la 10^{ème} séance de TP.
 - Les exercices numérotés de 19 à 21 sont à rendre avant la fin de la 12^{ème} séance de TP.
 - Les exercices 22 et 23 sont à rendre avant la fin de la dernière séance de TP.Vous rendrez vos exercices en les regroupant dans un fichier `kovalevskaia_markov_devoir_m.zip`, étant entendu que le $m^{\text{ième}}$ devoir correspond à la série d'exercices à rendre avant la $m^{\text{ième}}$ échéance. Vous pouvez rendre vos exercices avant la date limite mais toute remise est définitive. Les chargés de TP ont toute latitude pour modifier la note de TP finale afin de tenir compte de votre assiduité et de votre comportement.
3. Un script doit être clair et structuré :
 - les quantités numériques données par l'énoncé sont affectées à des variables pour pouvoir être facilement changées ;
 - les lignes de code sont commentées chaque fois que cela est nécessaire ;
 - les résultats théoriques doivent être donnés sous forme de commentaire ;
 - les scripts qui ne seront pas écrits en suivant ces consignes seront rejetés par les chargés de TP et ne rapporteront donc pas de point pour la note de TP.
4. Cette feuille doit vous permettre de travailler **en autonomie**. Il est important que vous exécutiez toutes les commandes données en exemple et que vous observiez ce qui en résulte. Il est aussi important que vous fassiez tous les exercices proposés, même ceux qui ne sont pas à rendre.

Premier contact avec R et Rstudio

On utilisera le logiciel R¹ sur les machines du CRIO UNIX à travers l'interface Rstudio². Vous devez être en possession de votre identifiant ENT et du mot de passe correspondant pour pouvoir utiliser ces machines.

L'application Rstudio se lance en cliquant sur l'icône bleue contenant un R en blanc au milieu. À son ouverture, la fenêtre de Rstudio se décompose en quatre (sous-)fenêtres :

1. en bas à gauche la "console" R, là où sont tapées et exécutées les commandes. À l'invite `>` tapez `> 3+4` puis appuyez sur la touche Entrée du clavier et observez le résultat.
2. en haut à gauche vous trouverez un éditeur de texte ; on peut y taper des commandes qui seront exécutées dans la fenêtre en dessous après avoir cliqué sur Run. Par exemple, tapez sur la première ligne `3+4` puis cliquez sur Run et observez le résultat dans la fenêtre en-dessous. Ensuite, tapez `1+4`, passez à la ligne dans ce fichier/fenêtre en appuyant sur la touche Entrée du clavier, tapez `5+6`, sélectionnez à l'aide de la souris les deux lignes que vous venez de taper puis cliquez sur Run et observez le résultat dans la fenêtre en-dessous. On peut ouvrir simultanément plusieurs fichiers dans cet éditeur, par exemple des fichiers avec l'extension `.R` qui contiennent chacun le code d'une fonction R que vous aurez créée.

1. R est disponible gratuitement, pour Windows, Mac ou Linux, à l'adresse <http://cran.r-project.org>.

2. On n'utilisera que les fonctionnalités de Rstudio disponibles sur la version gratuite téléchargeable à l'adresse <http://www.rstudio.org> (pour Windows, Mac ou Linux).

3. en haut à droite une fenêtre où l'on peut passer de la sous-fenêtre **Workspace** à la sous-fenêtre **History**. La sous-fenêtre **Workspace** contient toutes les variables et fonctions que vous avez définies actuellement présentes dans l'espace de travail, c'est à dire que **R** connaît et que vous pouvez appeler. La sous-fenêtre **History** garde la liste des commandes qui ont été exécutées par **R**.
4. en bas à droite une fenêtre où l'on peut entre autres consulter les différentes sorties graphiques (**Plots**) ainsi que le menu d'aide (**Help**).

Exécutez l'une après l'autre les commandes suivantes, soit depuis la fenêtre en haut à gauche, soit directement dans la fenêtre en bas à gauche (ce tableau se lit ligne par ligne). Observez les résultats ainsi que les changements dans les fenêtres **History** et **Workspace**

```
> 2/7                > 2^2                > 1:10
> 1.5:10             > 10:1                > seq(from = -5, to = 10, by = 3)
> seq(-5,10,3)       > seq(from = -5, length = 10) > seq(-5,10)
> rep(x=1, times=5)  > rep(1,5)           > w <- 1:5
> (x <- 1:5)         > rep(w,5)           > rep(1:5,5)
```

La flèche `<-`, obtenue en tapant successivement le signe "strictement inférieur" `<` puis le signe "moins" `-`, permet d'affecter un contenu à une variable.

Cherchez en exécutant

```
> help(seq)
```

ou encore

```
> ?rep
```

puis à l'aide de la fenêtre **Help** à quoi correspondent **seq** et **rep**.

Exécutez les commandes suivantes

```
> z <- 3:7          > w+z          > w/z          > w^z
```

Vous observerez que les opérateurs utilisés ci-dessus agissent "terme à terme". Il faut être extrêmement précautionneux avec le format des objets dans **R** car celui-ci accepte d'effectuer certains calculs qui n'ont pas forcément de sens

```
> z <- 1:5          > w <- 1:10          > (w+z)
```

Le langage **R** dispose de fonctions et constantes pré-définies

```
> cos(pi)          > exp(0+1i)          > abs(-3)          > sqrt(4)          > sqrt(-4)          > sqrt(1:5)
```

On peut définir de nouvelles fonctions, par exemple la fonction *translation* comme suit

```
> translation <- function(x) {
+   return(x+1)
+ }
```

L'apparition de `+` à la place de `>` est due au fait que **R** attend que vous complétiez le code commencé avec l'accolade ouvrante de la première ligne. Notez après avoir tapé ce code le changement dans la fenêtre **Workspace**. La fonction *translation* est présente dans l'espace de travail : on peut l'utiliser. Exécutez les commandes

```
> translation(2)          > translation(1:5)
```

Exécutez les commandes suivantes en observant la fenêtre **Workspace**

```
> a <- 2+3          > a          > b <- 1+a          > b          > a <- 2          > a          > b
```

Pour obtenir en ligne de commande le contenu de l'espace de travail tapez

```
> ls()
```

Il faut régulièrement purger l'espace de travail afin de ne pas encombrer inutilement la mémoire de l'ordinateur. Pour cela tapez

```
> rm(list=ls())
```

ou procédez directement depuis la fenêtre **Workspace**.

Exécutez les commandes suivantes en observant le résultat dans la fenêtre **Plots**

```
> x <- rnorm(100) > y <- rnorm(100) > plot(x,y)
```

Cherchez à partir de la fenêtre **Help** à quoi correspondent **rnorm** et **plot**.

Les objets de base

Un objet est désigné par son nom. Les objets de base sont :

1. les vecteurs et matrices;
2. les data-frames;

3. les listes.

Le contenu des objets peut être de différents types :

1. numérique (entier, réel, complexe) ;
2. chaîne de caractères ;
3. booléens (TRUE, FALSE, NA).

Nous ne nous intéresserons presque exclusivement qu'aux vecteurs et matrices de numériques ainsi qu'aux objets booléens.

Conseil : il faut garder en tête que, comme dans le langage Matlab/Octave, une utilisation judicieuse des matrices et vecteurs permet souvent d'éviter le recours à des structures d'itération, par exemple des boucles "for".

La fonction `c()` concatène des nombres ou des vecteurs pour obtenir des vecteurs

```
> (x <- c(-2,4,9))      > (y <- c(x,1,3))      > (z <- c(1:3,4,5:2))
```

Des commandes sont disponibles pour créer des matrices particulières

```
> (a <- diag(1:4))      > (b <- diag(rep(1,4)))      > (c <- toeplitz(1:4))
```

En général, les matrices sont créées à partir de vecteurs à l'aide de la commande `matrix()` en fixant le nombre de colonnes et/ou le nombre de lignes. Par défaut la matrice ainsi créée est remplie colonne par colonne. Pour la remplir ligne par ligne il faut ajouter la commande `byrow=TRUE`

```
> (x <- matrix(1:8, ncol=4))      > (y <- matrix(1:8, nrow=4))
> (z <- matrix(1:8, nrow=4, byrow=TRUE))
```

On peut concaténer horizontalement ou verticalement des vecteurs et des matrices à l'aide des commandes `cbind()` et `rbind()`

```
> (u <- cbind(y,z))      > (v <- rbind(y,z))
```

Pour extraire des éléments d'une matrice on procède comme suit

```
> u[1,3]      > u[,3]      > u[3,]      > u[1:3,1:3]
```

Le logiciel R dispose de certaines fonctions pour manipuler matrices et vecteurs

```
> u %*% u      > det(u)      > eigen(u)      > solve(u)      > t(u)
```

On utilisera par la suite la fonction `chol()` qui à toute matrice symétrique définie positive A associe la matrice triangulaire supérieure R telle que $A = R^t R$ où R^t est la transposée de R .

Exercice 1 Créer un vecteur contenant 5 valeurs régulièrement espacées entre 4 et 9 et utilisez-le comme argument des fonctions `sum()`, `prod()`, `mean()`, `median()`, `var()`, `sd()`, `max()`, `min()`, `length()`, `which.max()`, `which.min()`, `cumsum()`, `cumprod()`.

Exercice 2 Calculer les sommes des carrés des inverses des 10, 100 et 1000 premiers entiers non nuls et comparer les valeurs obtenues à $\sum_{k=1}^{\infty} \frac{1}{k^2} = \frac{\pi^2}{6}$.

Exercice 3 Utiliser R pour calculer les produits

$$\begin{pmatrix} 2 & 1 \end{pmatrix} \begin{pmatrix} 4 \\ 2 \end{pmatrix}, \quad \begin{pmatrix} 2 & 1 \end{pmatrix} \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}, \quad \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix} \begin{pmatrix} 4 \\ 2 \end{pmatrix}, \quad \begin{pmatrix} 2 & 1 \end{pmatrix} \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix} \begin{pmatrix} 4 \\ 2 \end{pmatrix}.$$

Exercice 4 Utiliser R pour résoudre le système

$$\begin{cases} 3x + 2y + z = 5 \\ 2x + 3y + z = 1 \\ x + 2y + 3z = 7 \end{cases}$$

Exercice 5 ♣ Soit X une variable aléatoire de loi uniforme sur les entiers de la forme k^2 où k est un entier compris entre 1 et 10. Calculer l'espérance et la variance de X .

Les objets de type booléens permettent souvent de raccourcir des calculs. Exécutez

```
> a <- 1      > b <- 1+1i      > a == 2
> is.complex(a)      > is.complex(b)      > is.complex(a) & is.complex(b)
> is.complex(a) | is.complex(b)      > a <- 3:7      > b <- 5
> a < b      > a <= b      > a[a<b]
```

Exercice 6 ♪

1. À l'aide des commandes vues jusqu'ici et de l'approximation par une somme de Riemann à 10, 100 puis 1000 termes, calculer une approximation de $\int_{-\pi}^{\pi} (\sin(10x))_+ dx$ où $(\sin(u))_+ = \max\{\sin(u), 0\}$;
2. Faire le calcul exact de $\int_{-\pi}^{\pi} (\sin(10x))_+ dx$. Donner le résultat de ce calcul en commentaire dans le script et comparer cette quantité aux approximations obtenues à la question précédente.

Les fonctions

On a déjà vu un exemple simple de fonction. La structure générale d'une fonction est

```
> nom_de_la_fonction <- function(liste_des_paramètres) {  
+ commandes  
+ return(objet_retourné_par_la_fonction)  
+ }
```

On a vu plus haut un exemple de fonction dont le paramètre d'entrée est un scalaire et qui renvoie un scalaire. Généralement quand on entre un vecteur (x_1, \dots, x_N) comme paramètre à une telle fonction, elle renvoie $(f(x_1), \dots, f(x_N))$.

On voit sur le code générique ci-dessus que l'on peut passer plusieurs arguments à une fonction. Cependant, elle ne doit retourner qu'un objet, lequel peut être par exemple un vecteur. Ainsi

```
> premiers_carres <- function(x) {  
+ u <- 1:x  
+ return(u^2)  
+ }
```

retourne les carrés des x premiers entiers non-nuls. Pour pouvoir passer un vecteur comme argument à une telle fonction il faut utiliser la commande `Vectorize()`. Exécuter

```
> vpremiers_carres <- Vectorize(premiers_carres, 'x')  
> x <- 1:5  
> vpremiers_carres(x)
```

Considérons maintenant le cas d'une fonction de deux arguments scalaires renvoyant un scalaire. Par exemple la fonction suivante

```
> norme <- function(x,y) {  
+ return(sqrt(x^2+y^2))  
+ }
```

Pour passer deux vecteurs (x_1, \dots, x_N) et (y_1, \dots, y_M) en argument à une telle fonction et récupérer la matrice de terme général $\text{norme}(x_i, y_j)$ il faut utiliser la commande `outer()`

```
> x <- 1:5  
> y <- 1:3  
> M <- outer(x,y,'norme')
```

Finalement, on peut définir des fonctions qui associent des vecteurs ou matrices à plusieurs scalaires, comme dans

```
> premieres_puissances <- function(x,y) {  
+ u <- 1:x  
+ return(u^y)  
+ }
```

qui retourne la liste des puissances y -ièmes des x premiers entiers non-nuls.

Exercice 7 À l'aide de la fonction `factorial()` et des commandes vues jusqu'ici créer une fonction qui à tout entier n associe la probabilité qu'une variable aléatoire de loi de Poisson de paramètre π soit supérieure ou égale à n .

Exercice 8 Créer à l'aide des commandes vues jusqu'ici une fonction qui à tout entier non-nul n associe la probabilité qu'une variable aléatoire de loi géométrique de paramètre $1/\pi$ soit supérieure ou égale à n .

Exercice 9 Calculer à l'aide des commandes vues jusqu'ici la probabilité qu'une variable aléatoire de loi binomiale de paramètres 8 et $1/\pi$ soit supérieure ou égale à 4.

Les graphes

Le langage R dispose de fonctions dédiées aux graphes.

Fonctions graphiques.	
<code>plot(x)</code>	Trace un graphe en prenant $1:n$ en abscisse et $x(1), \dots, x(n)$ en ordonnée.
<code>plot(x,y)</code>	Trace un graphe en prenant x en abscisse et y en ordonnée.
<code>curve(f)</code>	Trace le graphe de la fonction f .
<code>persp(x,y,z)</code>	Trace en “3 dimensions” la nappe correspondant aux points $x[i], y[j], z[i,j]=f(x[i],y[j])$.
Options de fonctions graphiques.	
<code>col=</code>	Fixe la couleur du tracé.
<code>type=</code>	Type de graphe : ‘p’ pour des points (option par défaut), ‘l’ pour des lignes...
<code>main=</code>	Titre de la figure
<code>axes=</code>	Si <code>axes=TRUE</code> fait apparaître les axes sur le graphe.
<code>xlab=, ylab=</code>	Noms des axes, pour ne rien faire figurer <code>xlab=</code> .
<code>xlim=, ylim=</code>	Limites des axes.

Pour superposer plusieurs tracés sur un même graphe il faut, au préalable, introduire la commande `par(new=TRUE)` comme dans l'exemple suivant :

```
> x <- seq(-pi,pi,0.1)
> y <- sin(x^2-x)
> z <- cos(x^2-x)
> plot(x,y,type='l',col='red')
observer le résultat puis continuer en exécutant
> par(new=TRUE)
> plot(x,z,type='l',col='cyan',main='Superposition de deux courbes')
```

Cependant, quand les graphiques sont à des échelles différentes, procéder comme précédemment peut s'avérer problématique. Pour s'en convaincre reprendre la série d'instructions précédentes en remplaçant la définition du vecteur z par

```
> z <- 0.5*cos(x^2-x)
```

Pour superposer plusieurs tracés en gardant comme référence les échelles du premier d'entre eux il faut procéder en utilisant la commande `lines` comme suit

```
> x <- seq(-pi,pi,0.1)
> y <- sin(x^2-x)
> z <- 0.5*cos(x^2-x)
> plot(x,y,type='l',col='red')
> lines(x,z,col='cyan')
```

Les tracés ainsi obtenus peuvent être sauvegardés en utilisant les menus de la fenêtre située en bas à droite. On peut aussi créer des figures “en 3 dimensions”

```
> x <- seq(-10,10,length=30)
> y <- x
> f <- function(x,y){
r <- sqrt(x^2 + y^2)
return(10*sin(r)/r)
}
z <- outer(x,y,f)
persp(x,y,z, theta=30, phi=30, expand=0.5,col='lightblue')
```

On utilisera fréquemment des histogrammes pour rendre compte des résultats de simulations. En effet, il s'agit d'un moyen pratique de comparer visuellement des distributions empiriques et des distributions théoriques. La fonction `hist()` appliquée à un vecteur de données (x_1, \dots, x_N) permet d'obtenir un histogramme. On a plusieurs façons de définir les classes de l'histogramme, ce qui se fait via l'option `breaks`

- Si on prend comme classes des intervalles $[a_1, a_2[, [a_2, a_3[, \dots, [a_{k-1}, a_k[$ alors il faut passer comme paramètre de l'option `breaks` le vecteur de coordonnées a_1, \dots, a_k ;
- Si on veut un nombre de classes m fixé à l'avance on passe cet entier m comme paramètre de l'option `breaks`

Par exemple, le code suivant permet d'obtenir le résultat de 100 lancers d'un dé à six faces équilibré (les détails seront vu au paragraphe suivant)

```
> x <- sample(x=1:6, prob=rep(1/6,6), size=100, replace=TRUE)
résumé par l'histogramme obtenu avec
> bords <- seq(0.5,6.5,1)
> h <- hist(x,breaks=bords,plot=TRUE,freq=FALSE)
> print(h)
```

Probabilités

Le logiciel R permet de manipuler un certain nombre de lois en combinant leurs noms avec les préfixes **d**, **p**, **q** ou **r**. Par exemple

- **dnorm(x)** donne la valeur en x de la densité de la loi $\mathcal{N}(0, 1)$;
- **pnorm(x)** donne la valeur en x de la fonction de répartition de la loi de densité $\mathcal{N}(0, 1)$;
- **qnorm(x)** donne le quantile d'ordre $x \in]0, 1[$ de la loi de densité $\mathcal{N}(0, 1)$;
- **rnorm(n)** retourne le résultat de n tirages au sort indépendants effectués suivant la loi $\mathcal{N}(0, 1)$.

Loi	Nom dans R	Nom des paramètres	Valeur par défaut des paramètres
Beta	beta	shape1, shape2	
Binomiale	binom	size, prob	
Cauchy	cauchy	location, scale	0,1
χ^2	chisq	df	
Exponentielle	exp	1/mean	1
Fisher	f	df1, df2	
Gamma	gamma	shape, 1/scale	
Géométrique	geom	prob	
Gaussienne	norm	mean, sd	0,1
Hypergéométrique	hyper	m, n, k	
Poisson	pois	lambda	
Student	t	df	
Uniforme continue	unif	min, max	0,1
Weibull	weibull	shape	

Exercice 10 ♡

1. Simuler le tirage de 1000 variables aléatoires indépendantes de loi $\mathcal{N}(0, 1)$.
2. Tracer un histogramme à 50 classes correspondant aux résultats de ces tirages.
3. Sur le même graphique, tracer la densité de la loi $\mathcal{N}(0, 1)$ et comparer les deux figures.
4. On note x_1, \dots, x_{1000} les résultats de ces tirages. Tracer l'extrapolation linéaire de la famille de points $(n, \frac{1}{n} \sum_{k=1}^n x_i)_{1 \leq n \leq 1000}$.

Le logiciel R permet également de traiter les tirages avec ou sans remise dans une population finie à l'aide de la commande **sample()**. Sa syntaxe est **sample(x, n, replace= , prob=)** où

- **x** est un vecteur listant la population sur laquelle est effectuée le tirage;
- **n** est le nombre de tirages effectués;
- **replace=TRUE** signifie que le tirage est effectué avec remise tandis que **replace=FALSE** signifie que le tirage est effectué sans remise. Par défaut **replace=FALSE**.
- Si **replace=TRUE**, dans **prob=p** l'argument **p** est un vecteur de même taille que **x** et **p[i]** est la probabilité de tirer **x[i]**. Par défaut, le tirage avec remise s'effectue suivant la loi uniforme. Si **replace=FALSE**, dans **prob=p** l'argument **p** est un vecteur de même taille que **x** et **p[i]** est la probabilité de tirer **x[i]** en premier. Ensuite, le vecteur de probabilités de référence est adapté pour que le résultat corresponde bien à un tirage sans remise.

Exercice 11 ♡

1. Simuler 1000 tirages indépendants d'un dé pipé de sorte que $p_4 = 1/3$ et $p_1 = p_2 = p_3 = p_5 = p_6$.
2. Tracer un histogramme à 6 classes correspondant aux résultats de ces tirages.
3. On note x_1, \dots, x_{1000} les résultats de ces tirages. Tracer l'extrapolation linéaire de la famille de points $(n, \frac{1}{n} \sum_{k=1}^n x_i)_{1 \leq n \leq 1000}$.

Itérations et structures de contrôle

Les itérations se font suivant un nombre d'exécutions fixé à l'avance, par exemple

```
> for(i in 1:10){ print(i) }
```

ou en subordonnant les exécutions à la vérification d'une condition (ce qui peut conduire à une boucle infinie)

```
> i <- 1
```

```
> while(i<11){ print(i); i <- i+1 }
```

On dispose aussi de la construction `if (condition) {A} else {B}` que l'on peut utiliser par exemple comme dans

```
xlogx <- function(x) {  
+ if(x>0){return(x*log(x))}  
+ else{return(0)}  
}
```

Exercice 12 *Le paradoxe des anniversaires* ♣

1. Créer une fonction qui à tout entier N tel que $2 \leq N \leq 365$ associe la probabilité que dans une classe de N élèves au moins 2 élèves fêtent leurs anniversaires le même jour.
2. Trouver la plus petite valeur de N pour laquelle cette probabilité dépasse 0.5.
3. Tracer le graphe de la fonction trouvée à la première question.

Exercice 13 ♣ Soit X une variable aléatoire de densité $f(x) = \frac{C}{(1+x)^2} \mathbb{1}_{[0,2]}(x)$.

1. Calculer C pour que f soit effectivement une densité de probabilité.
2. Simuler à l'aide de la méthode d'inversion un échantillon de taille 1000 de la loi de densité f .
3. Tracer un histogramme à 50 classes correspondant aux résultats de ces tirages.
4. Sur le même graphique, tracer le graphe de f et comparer les deux figures.

Exercice 14 ♣ Soit X une variable aléatoire de densité $f(x) = Ce^x \mathbb{1}_{[0,1]}(x)$.

1. Calculer C pour que f soit effectivement une densité de probabilité. Calculer $\mathbb{E}[X]$.
2. Simuler à l'aide de la méthode de rejet un échantillon de taille 1000 de la loi de densité f .
3. Tracer un histogramme à 50 classes correspondant aux résultats de ces tirages.
4. Sur le même graphique, tracer le graphe de f et comparer les deux figures.

Exercice 15 ♣ Soit X une variable aléatoire de loi de Cauchy, c'est à dire une variable aléatoire réelle de densité $f(x) = \frac{1}{\pi(1+x^2)}$.

1. Montrer que X n'est pas intégrable.
2. Simuler un échantillon de taille 1000 de la loi de Cauchy.
3. On note x_1, \dots, x_{1000} les résultats de ces tirages. Tracer l'extrapolation linéaire de la famille de points $(n, \frac{1}{n} \sum_{k=1}^n x_k)$.

Exercice 16 ♣ Soit $(X_i^j)_{i,j \in \mathbb{N}^*}$ une suite de variables aléatoires indépendantes de même loi de Poisson de paramètre 1 et $(Y_i^j)_{i,j \in \mathbb{N}^*}$ une suite de variables aléatoires indépendantes de même loi uniforme sur $[0, 1]$. On introduit pour tous $j, n \in \mathbb{N}^*$ les variables

$$M^X(j, n) = \frac{1}{n} \sum_{i=1}^n X_i^j, \quad M^Y(j, n) = \frac{1}{n} \sum_{i=1}^n Y_i^j, \quad T^X(j, n) = \sqrt{n} (M^X(j, n) - 1),$$

et

$$T^Y(j, n) = \sqrt{n} \left(\frac{M^Y(j, n) - \frac{1}{2}}{\sqrt{\frac{1}{12}}} \right).$$

1. Faire un tirage au sort des valeurs de X_i^j et Y_i^j pour $1 \leq i, j \leq 500$.
2. Tracer les histogrammes correspondant aux vecteurs $(T^X(j, 500))_{1 \leq j \leq 500}$ et $(T^Y(j, 500))_{1 \leq j \leq 500}$ pour les résultats des tirages effectués ci-dessus. Les comparer à la courbe de la densité de la loi $\mathcal{N}(0, 1)$.

3. Tracer les histogrammes correspondant aux vecteurs $(T^X(j, 5))_{1 \leq j \leq 500}$ et $(T^Y(j, 5))_{1 \leq j \leq 500}$ pour les résultats des tirages effectués ci-dessus. Les comparer à la courbe de la densité de la loi $\mathcal{N}(0, 1)$.

Exercice 17 ♣ On rappelle l'expérience des urnes de Polya : à l'instant 0, une urne contient une boule rouge et une boule verte et on effectue une succession de tirages définis par la règle suivante : on tire une boule de l'urne au hasard et on la remet dans l'urne en ajoutant une boule de la même couleur (on dispose en réserve d'une infinité de boules rouges et vertes). On note S_n le nombre de boules rouges au temps $n \geq 0$. Il a été vu en TD que pour tout $n \geq 0$ et tout $k \in \{1, \dots, n+1\}$ on a

$$\mathbb{P}(S_n = k) = \frac{1}{n+1}.$$

1. Effectuer une simulation de $(S_n)_{0 \leq n \leq 20}$. Il s'agit de simuler des trajectoires.
2. Soient $(S_{20}^j)_{1 \leq j \leq 1000}$ des variables aléatoires indépendantes de même loi que S_{20} . Effectuer un tirage de ces variables aléatoires et donner, à l'aide du moyen graphique que vous jugerez approprié, une illustration du résultat qui a été prouvé en TD.

Exercice 18 *Un modèle simple de sondage.* ♣ Une élection se prépare dans une population composée de N votants. Deux candidats sont en lice : n_A votants ont l'intention de voter pour A , les $n_B = N - n_A$ autres votants ont l'intention de voter pour B (il n'y a pas d'indécis). Un institut de sondage est mandaté pour interroger n personnes prises au hasard uniformément parmi les N votants.

1. Créer une fonction qui aux arguments n, n_A, n_B associe l'intention de vote dont est crédité A à l'issue de ce sondage.
2. Répéter 1000 fois ce sondage indépendamment les uns des autres en fixant $n = 20, n_A = 300, n_B = 400$.
 - (a) Combien de sondages donnent A vainqueur ?
 - (b) Quelle est la variance observée sur les prédictions de ces 1000 sondages.
 - (c) Répondre aux mêmes questions en prenant $n = 50$.

On suppose à présent que la population se partage en trois groupes : $n_A = 240$ votants ont l'intention de voter pour A , $n_B = 320$ votants ont l'intention de voter pour B , $n_c = 140$ sont indécis et ne se prononcent donc pas. Répondez aux mêmes questions que précédemment.

Exercice 19 ♣ Simuler le tirage de 50 couples aléatoires gaussiens indépendants

1. de moyenne le vecteur nul et de matrice de dispersion $\Sigma_1 = I_2$;
2. de moyenne le vecteur nul et de matrice de dispersion

$$\Sigma_2 = \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix};$$

3. de moyenne le vecteur nul et de matrice de dispersion

$$\Sigma_3 = \begin{pmatrix} 4 + \varepsilon & 2 \\ 2 & 1 + \varepsilon \end{pmatrix}$$


où ε prend successivement les valeurs 1, 0.1, 0.01.

Dans chacun de ces cas, tracer le nuage de points $(X_i, Y_i)_{1 \leq i \leq 50}$.

Exercice 20 ♣ On rappelle une expérience aléatoire considérée dans la feuille de TD : Un joueur peut lancer deux fois de suite un dé à six faces équilibré pour obtenir le meilleur score possible. Si le score obtenu au premier lancer le satisfait il peut en rester là. Si ce score ne le satisfait pas il peut lancer une deuxième fois le dé mais le score obtenu au premier lancer est alors perdu. Le joueur décide de la stratégie suivante : il s'arrêtera si le résultat du premier lancer est supérieur ou égal à un seuil k et il lancera une deuxième fois sinon.

1. Pour chaque valeur de $k \in \{2, 6\}$ simuler le tirage de 100 variables indépendantes de même loi que X^k . Tracer les histogrammes correspondants et calculer pour chaque valeur de k la moyenne empirique de la valeur de X^k ainsi obtenue. Comparer cette valeur à la valeur exacte obtenue en TD.

- À présent le joueur a la possibilité de lancer trois fois de suite le dé suivant le même principe : si le score obtenu au premier lancer le satisfait il peut en rester là. Si ce score ne le satisfait pas il peut lancer une deuxième fois le dé mais le score obtenu au premier lancer est alors perdu. Si le score obtenu au deuxième lancer le satisfait il peut en rester là. Si ce score ne le satisfait pas il peut lancer une troisième fois le dé mais le score obtenu au lancer précédent est alors perdu. Calculer les moyennes empiriques de la valeur de X^{k_1, k_2} sur 100 tirages où X^{k_1, k_2} est le résultat de la procédure avec un seuil k_1 au premier tirage et un seuil k_2 au deuxième tirage et ce pour toutes les valeurs possibles de k_1 et k_2 .

Exercice 21  Un pou vivant dans un milieu scolaire peut se trouver dans trois états différents : sur la tête d'un enfant (état E), sur la tête d'un instituteur (état I) ou sur la tête du directeur (état D). Sa situation peut changer chaque jour une fois en suivant la règle :


- quand il est sur la tête d'un enfant il peut y rester ou passer sur la tête d'un autre enfant avec probabilité 0.7, passer sur la tête d'un instituteur avec probabilité 0.2 ou passer sur la tête du directeur avec probabilité 0.1 ;
- quand il est sur la tête d'un instituteur il peut y rester ou passer sur la tête d'un autre instituteur avec probabilité 0.3, passer sur la tête d'un enfant avec probabilité 0.5 ou passer sur la tête du directeur avec probabilité 0.2.
- quand il est sur la tête du directeur il peut y rester avec probabilité 0.1, passer sur la tête d'un instituteur avec probabilité 0.6 ou passer sur la tête d'un enfant avec probabilité 0.3.

On s'intéresse à la trajectoire d'un pou dans l'ensemble $\{E, I, D\}$ pendant cent jours que l'on modélise par des variables aléatoires X_1, \dots, X_{100} . On suppose que $\mathbb{P}(X_1 = E) = 1$.


- Simuler la trajectoire d'un pou et compter la proportion de temps passé par le pou dans chacun des trois états possibles.
- Simuler les trajectoires de cent poux en supposant qu'elles sont indépendantes les unes de autres et suivent toutes la règle donnée ci-dessus. (en particulier ils sont tous sur la tête d'un enfant le premier jour). Pour chacun des 3 états possibles, quelle est la proportion des cent poux finissant dans cet état ?
- Répondre aux mêmes questions quand le pou/les poux sont tous sur la tête du directeur le premier jour.
- On suppose à présent que le premier jour chaque pou se trouve dans un état tiré au hasard uniformément et indépendamment des états des autres poux. Répondre aux mêmes questions que précédemment.
- Résoudre le système

$$\begin{cases} 0.7x + 0.5y + 0.3z = x \\ 0.2x + 0.3y + 0.6z = y \\ 0.1x + 0.2y + 0.1z = z. \end{cases}$$

Que constatez-vous ?

Exercice 22  Soit X une variable aléatoire de densité $f(x) = Ce^x \mathbb{1}_{[0,1]}(x)$. Il a été vu à l'exercice 15 comment obtenir un échantillon de cette loi à l'aide de la méthode du rejet. On se propose à présent de comparer les performances de cette méthode suivant que l'on considère comme loi de référence la loi de densité $g_1(x) = \mathbb{1}_{[0,1]}(x)$ ou la loi de densité $g_2(x) = 2x \mathbb{1}_{[0,1]}(x)$.

- Déterminer les plus petites constantes c_1 et c_2 telles que pour tout $x \in \mathbb{R}$ on a $f(x) \leq c_1 g_1(x)$ et $f(x) \leq c_2 g_2(x)$.
- Simuler à l'aide de la méthode de rejet deux échantillons de taille 1000 de la loi de densité f , l'un à partir de g_1 et l'autre à partir de g_2 . Laquelle des deux simulations produit le plus de rejets ?

Exercice 23  Soit X_1, \dots, X_n un échantillon de la loi $\mathcal{N}(\theta, 1)$ où $\theta \in \mathbb{R}$ est un paramètre inconnu. En exécutant

```
> theta <- rnorm(1)
```

on fixe une valeur pour θ qui demeure inconnue tant que l'on ne consulte pas la fenêtre **Workspace** (ce qui est recommandé). Le paramètre θ nous étant inconnu, on note \mathbb{P}_θ la probabilité sous laquelle X_1, \dots, X_n sont indépendantes et de loi $\mathcal{N}(\theta, 1)$. L'exercice consiste en proposer un intervalle A dont les bornes sont des fonction de X_1, \dots, X_n et tel que *pour toute valeur de θ* on a $\mathbb{P}_\theta(\theta \in A) \geq 95\%$.

1. À l'aide des commandes de **R** trouver deux réels α, β tels que si U est de loi $\mathcal{N}(0, 1)$ sous \mathbb{P} alors

$$\mathbb{P}(U \in [\alpha, \beta]) = 95\%.$$

Existe-t-il un seul couple (α, β) tel que la condition ci-dessus est vérifiée? Existe-t-il un couple (α_0, β_0) meilleur que les autres (en quel sens?) vérifiant la condition ci-dessus?

2. Trouver à l'aide de (α_0, β_0) et des réalisations de X_1, \dots, X_n un intervalle A tel que pour tout θ on a $\mathbb{P}_\theta(\theta \in A) = 95\%$.
3. Répéter de façon indépendante n tirages d'un échantillon de taille 100 de la loi $\mathcal{N}(\theta, 1)$ et compter combien de fois θ est bien dans l'intervalle trouvé à la question précédente.