Exercice 1

Implémenter en R une simulation de l'aiguille de Buffon.
Étudier la qualité de l'estimateur de π obtenu en fonction du rapport entre la longueur L de l'aiguille et la largeur D des lames de parquet. On pourra par exemple estimer l'erreur quadratique moyenne*.
On pourra dans un premier temps implémenter une fonction `estim_pi` prenant comme paramètres le nombre n de lancers d'aiguille et le rapport $r = \frac{L}{D}$, et renvoyant une estimation de π.
Dans un deuxième temps, on pourra implémenter une fonction `mse_pi` prenant comme paramètres le nombre n de lancers d'aiguille, le rapport $r = \frac{L}{D}$ et le nombre d'estimations indépendantes r_{rep}, et renvoyant une estimation du Mean Squared Error de l'estimateur de paramètres (n, r).
On pourra enfin tracer le Mean Squared Error de l'estimateur de paramètres (n, r) en fonction du paramètre r (pour une même valeur de n).

Exercice 2

On désire comparer l'efficacité des méthodes d'intégration MC :

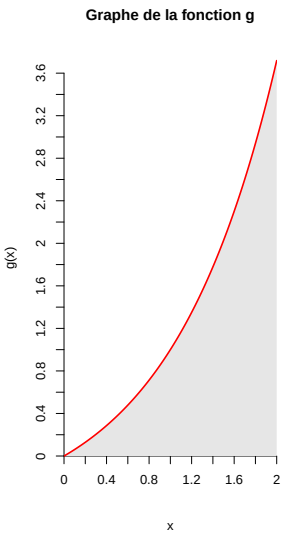
- MC par "tirage noir ou blanc"
- MC simple
- MC suivant l'importance

La fonction que l'on doit intégrer est :

$$g(x) = \frac{\exp(x) - 1}{\exp(1) - 1}$$

On doit calculer :

$$I = \int_0^2 g(u)\, du$$



1. Par intégration analytique, montrer que la valeur de l'intégrale est (facultatif) :

$$I = \frac{\exp(2) - 3}{\exp(1) - 1}$$

   Vous pouvez également vérifier numériquement cette formule en utilisant la fonction `integrate` de R.

2. Implémenter dans R la méthode MC par "tirage blanc ou noir".
3. Implémenter dans R la méthode MC simple (cas particulier de la méthode MC suivant l'importance).
4. Implémenter dans R la méthode MC suivant l'importance en utilisant une loi Bêta de paramètre $(\alpha = 2, \beta = 1)$ comme loi d'échantillonnage. Les lois Bêta ayant comme support $[0,\ 1]$, il faudra considérer une mise à l'échelle pour s'adapter au support d'intégration $[0,\ 2]$. Vous pourrez notamment vous appuyer sur le résultat suivant : Si une variable aléatoire X a comme support $X(\Omega) = [0,\ 1]$, alors la variable $Y = aX$ (avec $a > 0$) a comme support $X(\Omega) = [0,\ a]$ et la densité f_Y de Y peut s'exprimer en fonction de la densité f_X de X :

$$f_Y(y) = \frac{1}{a} f_X\left(\frac{y}{a}\right)$$

*. Mean Squared Error :

$$MSE = \mathbb{E}\left[(\hat{\theta} - \theta)^2\right] \qquad \text{et} \qquad MSE \simeq \frac{1}{n_{\text{rep}}} \sum_{i=1}^{n_{\text{rep}}} (\hat{\theta}_i - \theta)^2$$

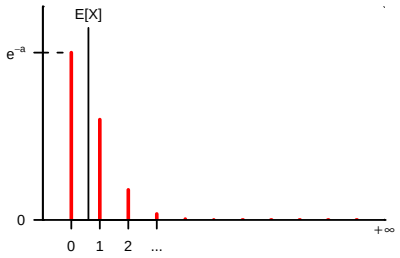
5. Proposer une autre fonction de densité pour la méthode MC suivant l'importance. Vous pouvez rester dans la famille des lois Bêta et tenter soit d'améliorer soit de dégrader la loi d'échantillonnage.

6. Comparer l'efficacité des quatre méthodes. On pourra utiliser le MSE comme critère de qualité. On utilisera la même taille (petite) n d'échantillon pour toutes les méthodes. On effectuera un grand nombre n_{rep} de répétitions de chaque méthode. Par exemple, on pourra prendre $n = 100$ et $n_{\text{rep}} = 1000000$.

Exercice 3

1. Implémenter en R une fonction `my_rpois` de simulation d'une variable de Poisson de paramètre (λ) en utilisant une méthode d'inversion de la fonction de répartition. On rappelle que la loi de probabilité d'une variable aléatoire X suivant une distribution de Poisson de paramètre (λ) est :

$$\mathbb{P}[X = x] = \exp(-\lambda)\,\frac{\lambda^x}{x!} \qquad \text{si} \qquad x \in \mathbb{N}$$



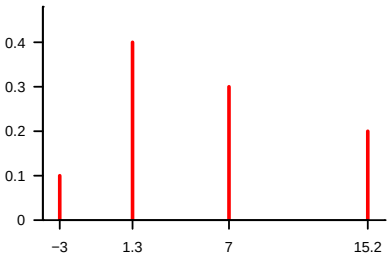
Pour les simulations, utilisez des valeurs non entières de λ.

Vous pouvez utiliser les fonctions `dpois` et `ppois`, mais pas les fonctions `rpois` et `qpois`.

Dans un premier temps, vous pouvez implémenter une fonction `my_rpois_one(lambda)` qui renvoie une seule valeur tirée aléatoirement dans une loi de Poisson de paramètre `lambda`. La fonction `my_rpois(n, lambda)` pourra appeler `n` fois la fonction `my_rpois_one` pour renvoyer un vecteur de `n` valeurs tirées aléatoirement et indépendamment dans une loi de Poisson de paramètre `lambda`.

2. Implémenter en R une fonction `my_rdiscret` de simulation de la variable aléatoire réelle discrète X définie par la loi de probabilité suivante :

x	-3	1.3	7	15.2
$\mathbb{P}[X = x]$	0.1	0.4	0.3	0.2

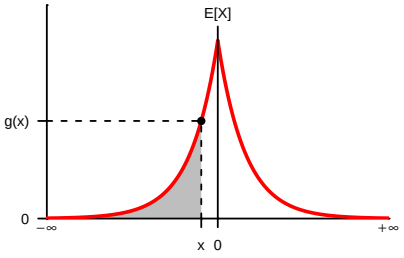


Vous pouvez utiliser la fonction `cumsum`, mais pas la fonction `sample`.

Dans un premier temps, vous pouvez implémenter une fonction `my_rdiscret_one()` qui renvoie une seule valeur tirée aléatoirement dans la loi de X. La fonction `my_rdiscret(n)` pourra appeler `n` fois la fonction `my_rdiscret_one` pour renvoyer un vecteur de `n` valeurs tirées aléatoirement et indépendamment dans la loi de X.

3. **Méthode de transformation :** Implémenter en R une fonction `my_rlaplace` de simulation d'une variable de Laplace en utilisant une méthode d'inversion de la fonction de répartition. La fonction de densité g d'une loi de Laplace est :

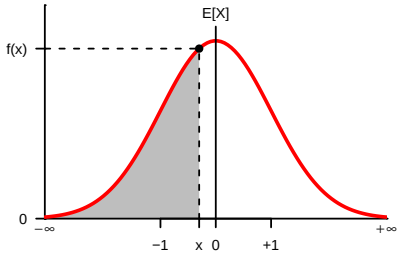
$$g(x) = \frac{1}{2}\,\exp(-|x|) \qquad \text{si} \qquad x \in \mathbb{R}$$



Il faudra dans un premier temps trouver l'expression analytique de la fonction de répartition de la loi de Laplace, puis celle de la fonction quantile (fonction réciproque de la fonction de répartition). Implémenter explicitement une fonction de densité `my_dlaplace`, une fonction de répartition `my_plaplace` et une fonction quantile `my_qlaplace`.

4. **Méthode de rejet :** Utiliser la méthode de simulation de la distribution de Laplace développée dans la question précédente pour implémenter en R une fonction `my_rnorm` de simulation d'une variable normale $\mathcal{N}(0,1)$ en utilisant une méthode de rejet. On rappelle que la loi de probabilité d'une variable aléatoire X suivant une distribution Normale centrée-réduite $\mathcal{N}(0,1)$ est :

$$f(x) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right) \qquad \text{si} \qquad x \in \mathbb{R}$$



Il faudra dans un premier temps trouver la plus petite valeur m telle que :

$$\forall x \in \mathbb{R}, f(x) \leq m\, g(x)$$

où f est la densité d'une variable normale centrée-réduite et g est la densité d'une variable de Laplace. Pour cela, on pourra rechercher le maximum de la fonction h définie par :

$$\forall x \in \mathbb{R}, h(x) = \frac{f(x)}{g(x)}$$

Montrer que la plus petite valeur est (facultatif) :

$$m = \sqrt{\frac{2\exp(1)}{\pi}}$$

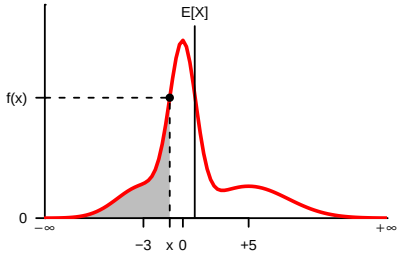
Vous vérifierez que votre implémentation de la méthode a bien le taux de rejet attendu.

5. **Algorithme MCMC :** Implémenter en R une fonction `my_rtarget` de simulation par méthode MCMC de la variable aléatoire réelle continue X définie par la densité f suivante :

$$f(x) = 0.2\; g_1(x) + 0.5\; g_2(x) + 0.3\; g_3(x) \qquad \text{si} \qquad x \in \mathbb{R}$$

avec :

- g_1 densité d'une loi normale $\mathcal{N}(\mu = -3, \sigma = 2)$
- g_2 densité d'une loi normale $\mathcal{N}(\mu = 0, \sigma = 1)$
- g_3 densité d'une loi normale $\mathcal{N}(\mu = 5, \sigma = 3)$



Le choix de la loi de proposition est libre. Vous implémenterez explicitement une fonction de densité `dprop` et une fonction "random" `rprop` pour la loi de proposition. On pourra faire varier les éventuels paramètres de cette loi ou tester d'autres lois de proposition.

Exercice 4

On désire faire une estimation de la fréquence θ de mutants dans une grande population de bactéries. On réalise un échantillonnage de n bactéries, parmi lesquelles on compte x mutants. Pour les applications numériques, on prendra $n = 100$ et $x = 70$.

1. Implémenter une méthode MCMC - Metropolis-Hastings pour échantillonner dans la loi a posteriori $\pi(\theta|x)$ en utilisant une loi a priori $\pi(\theta)$ uniforme sur $[0,\ 1]$ [†]. Au sein de l'algorithme MCMC vous pouvez utiliser la loi de proposition de votre choix, pourvu que son support soit $[0,\ 1]$ [‡]. N.B. : Il sera éventuellement nécessaire de sous-échantillonner la chaine de Markov générée afin d'améliorer votre méthode. Il peut notamment être intéressant de regarder si la chaine que vous avez construite a tendance à rester assez longtemps à une même valeur.

2. Implémenter une méthode de simulation de la loi conjointe (données, paramètres) en utilisant une loi a priori $\pi(\theta)$ uniforme sur $[0,\ 1]$. En pratique, on tirera une valeur de θ dans son prior puis on simulera le nombre de mutants en utilisant ce θ. On réalisera un grand nombre de simulations de ce modèle. On en déduira la loi a posteriori de θ (sachant qu'on a observé $x = 70$ mutants) [§].

3. Donner dans les deux cas un intervalle de crédibilité à 95% de θ.

†. Ce problème peut en réalité être résolu de manière analytique. Lorsque la vraisemblance $f(x \mid \theta)$ est une loi binomiale (n, θ) et que le prior $\pi(\theta)$ est une loi uniforme sur $[0,\ 1]$, alors le posterior $\pi(\theta \mid x)$ est une loi bêta $(x+1, n-x+1)$. N.B. : On peut même généraliser ce résultat lorsque le prior est une loi bêta de paramètres (α, β) quelconques. Dans ce cas, le posterior est une loi bêta $(x + \alpha, n - x + \beta)$.

‡. Les lois bêta répondent à ce critère, en particulier, la loi bêta de paramètres $(1, 1)$, c'est-à-dire la loi uniforme. On peut aussi choisir une loi bêta unimodale de mode θ, par exemple la loi bêta de paramètres $\left(\frac{1}{1-\theta}, 2\right)$.

§. On remarquera qu'avec cette méthode, on n'a pas besoin d'évaluer la vraisemblance $f(x \mid \theta)$.