Devoir 2: Méthode de la transformée inverse

La méthode de la transformée inverse permet de simuler un échantillon d'une variable aléatoire X de fonction de répartition F à la condition d'être capable d'inverser cette dernière. Par exemple, soit

$$X \sim Exp(1).$$

On a alors

$$F^{-}(u) = \log(1-u).$$

Pour simuler une réalisation de X, il suffit alors de simuler $u \sim \mathcal{U}(0,1)$ et d'évaluer $F^{-}(u)$.

Autre exemple: Soit $Y \sim Ber(p)$ qui a donc

$$G(x) = (1-p)\mathbb{1}(0 \leq x < 1) + \mathbb{1}(x \geq 1)$$

pour fonction de répartition et

$$G^{-}(u) = \mathbb{1}(u > 1-p)$$

pour inverse (généralisé). On simule une réalisation de Y de la même manière que pour X.

Exercice 1

(a)

Programmez une fonction qui retourne une fonction permettant de simuler des réalisations d'une variable aléatoire discrète en utilisant la méthode de la transformée inverse. Votre fonction doit respecter les contraintes suivantes:

- Accepter un seul argument `x`. Celui-ci est un vecteur réel de taille arbitraire.
- Retourner une fonction (dite «simulateur») acceptant un seul argument `n`. Celui-ci est un entier positif.
- Tenir le compte de chacune des valeurs simulées.

Si `x` est un vecteur de longueur N, la fonction retournée devra simuler les entiers $0, \ldots, N-1$. La valeur k sera simulée avec probabilité proportionnelle à `x[k]`. Par exemple, les vecteurs `x1 = [0.2, 0.8]` et `x2 = [4, 16]` sont tous les deux valides et décrivent la même distribution de probabilité supportée sur $\{0,1\}$. Par ailleurs, un appel au simulateur doit retourner un vecteur de `n` réalisations.

(b)

Programmez une fonction permettant d'appliquer une transformation à un simulateur produit par la fonction que vous avez programmé en (a). Votre fonction doit respecter les contraintes suivantes:

- Accepter deux arguments `sim` et `trans`. `sim` est le simulateur, `trans` la transformation. `trans` est une fonction bijective à image dans les entiers positifs.
- Retourner une fonction (simulateur) permettant de simuler de la distribution originale de `sim` transformée par `trans`.

Mathématiquement, si la fonction originale `sim` permet de tirer des échantillons de la variable aléatoire X et que `trans` correspond à la fonction h, le simulateur retourné par la fonction devra simuler de la variable aléatoire $h(X)$. Par ailleurs, notez que le compte des valeurs simulées pour le nouveau simulateur doit être initialisé à 0.

(c)

Programmez une fonction permettant de faire la somme de deux simulateurs au sens suivant: si `sim1` et `sim2` simulent des données proportionnellement aux vecteurs `x1` et `x2` respectivement, le simulateur résultant simulera proportionnellement à `x1 + x2`. Si `x1` et `x2` ne sont pas de la même longueur, supposez que les valeurs absentes sont 0.

(d)

Programmez une fonction retournant un vecteur contenant la proportion de chacun des nombres générés jusqu'à maintenant. Votre fonction doit prendre un seul argument `sim` qui est le simulateur d'intérêt.

(e)

Programmez une fonction qui affiche l'histogramme des valeurs générées jusqu'à maintenant par un simulateur.

Exercice 2

Vous disposez des chiffres $2, 2, 5, 6, 6, 6, 7, 9$ et des opérateurs binaires +, -, *, /, $\hat{}$. On ne considère que la suite des nombres entiers que l'on peut former en respectant ces règles:

- les chiffres ne peuvent être combinés qu'à l'aide d'une opération binaire,
- on ne peut utiliser chaque chiffre qu'une seule fois,
- il est permis de réutiliser les opérations aussi souvent que voulu et
- les parenthèses sont interdites.

Par exemple, $1062889 = 2 + 2 \times 9^6 + 5$ fait partie de cette suite. Calculez le plus de termes possibles compris entre -1×10^{10} et 1×10^{10}. Pour chacun des termes calculés, il doit être possible de retrouver sa «décomposition» facilement et rapidement.

*Indice: pour utiliser un opérateur infixe comme un opérateur préfixe dans R, il suffit de l'entourer de backticks "`". Par exemple,* `2 + 2 == `+`(2, 2)`