Devoir 3 : MCMC & OOP

Patrick Fournier

22 octobre 2021

Répondez aux questions en complétant le fichier `solution.R`. Veuillez respecter la structure du fichier. N'hésitez pas à consulter la documentation de R !
Les exercices suivants visent à vous guider dans la construction d'un échantilloneur Metropolis-Hastings de base. Chaque exercice aboutit avec la création d'une nouvelle classe S3. N'oubliez pas de valider les arguments passés par l'utilisateur aux constructeurs que vous implémenterez.

Exercice 1

Dans un premier temps, nous avons besoin d'une classe pour les distribution de probabilité sur $\mathbb{R}$ possédant une densité. Programmez la classe `distr` en implémentant les fonctions/méthodes suivantes :

- `distr(density, parameters, name)` où
  - `density` $= f(y, |x, \theta)$ est la fonction de densité de la distribution,
  - `parameters` est une liste de paramètres et
  - `name` est une chaîne de caractères décrivant la distribution.

  Comme le suggère le nom de la fonction, il s'agit du contructeur pour la classe.
- `dens(d, y, x, log = FALSE)` où
  - `d` $\in$ `distr`,
  - `x` et `y` $\in \mathbb{R}$.

  Un appel à cette méthode doit retourner la (log-)densité associée à `d` évaluée en `y` conditionnellement à `x`.
- `print(x)` où `x` $\in$ `distr`. Doit afficher à l'écran la description de la distribution et la valeur des différents paramètres.

Exemple (Bernoulli) :

```
> ber <- distr(\(y, ...) dbinom(x = y, size = 1L, ...),
               list(prob = 0.75),
               "Bernoulli 3/4")
> ber
Bernoulli 3/4
prob = 0.75
> dens(ber, 0.0)
[1] 0.25
> dens(ber, 0.0, log = TRUE)
[1] -1.386294
> dens(ber, -42.0, log = TRUE)
[1] -Inf
```

Exercice 2

Nous avons besoin d'une classe regroupant les distributions de probabilités à densité symétrique desquelles nous pouvons facilement simuler. Implémentez une telle classe nommée `kern` en spécialisant `distr`. Pour ce faire, implémentez les méthodes suivantes :

- `kern(density, parameters, name, sampler)` où `sampler` $= f(n|x, \theta)$ est une fonction retournant un échantillon de taille `n` de la distribution, les trois autres paramètres comme pour `distr`. Il s'agit du constructeur pour la classe.
- `rand(d, n, x)` où `d` $\in$ `kern`, `n` $\in \mathbb{N}$ et `x` $\in \mathbb{R}$. Génère un échantillon de taille `n` distribué selon `d` conditionnellement à `x`.

Exemple (gausienne) :

```
> gauss <- kern(\(y, x, ...) dnorm(x = y, mean = x, ...),
                list(sd = 1.0),
                "Normale mean = x, sd = 1",
                \(n, x, ...) rnorm(n, mean = x, ...))
> gauss
Normale mean = x, sd = 1
sd = 1
> dens(gauss, 0.6, 0.0)
[1] 0.3332246
> dens(gauss, 0.6, 3.0)
[1] 0.02239453
> dens(gauss, 0.6, 0.0, log = TRUE)
[1] -1.098939
> rand(gauss, 10, x = 0)
 [1]  0.82161757  0.84298978  0.44683505  1.08986384  0.01572262  1.72111406
 [7]  0.55479635  0.85091031  0.49277531 -0.12014435
> rand(gauss, 10, x = 3)
 [1] 2.451766 2.635137 2.115987 3.785050 4.632933 4.043525 1.961600 3.730628
 [9] 2.349120 4.101349
```

Exercice 3

Finalement, programmez la classe `mhsampler` représentant un échantilloneur Metropolis-Hastings. De manière informelle, l'algorithme est le suivant : Pour une densité cible $\pi(y)$ et un noyeau de transition $q(y|x)$, à chaque itération,

1. Échantillonez $y \sim q(\cdot|x)$ où x est la valeur échantillonnée à la dernière itération
2. Calculez le ratio d'acceptation
$$\alpha(y|x) = \frac{\pi(y)q(x|y)}{\pi(x)q(y|x)}$$
3. Retournez y avec probabilité $\min(\alpha(y|x), 1)$, x sinon

Implémentez les méthodes suivantes :

— `mhsampler(target, proposal)` où `target` ∈ `distr` et `proposal` ∈ `kern`. Il s'agit du constructeur.
— `aratio(sampler, y, x, log = FALSE)` où `sampler` ∈ `mhsampler` et `y`, `x` ∈ $\mathbb{R}$. Calcule le (log-)ratio d'acceptation donné plus haut.
— `rand(sampler, n, init = 0.0)` où `sampler` ∈ `mhsampler`, `n` ∈ $\mathbb{N}$ et `init` ∈ $\mathbb{R}$. Génère un échantillon de taille `n` en utilisant l'algorithme de Metropolis-Hastings initialisé à `init`.

```
expo <- distr(\(y, ...) dexp(x = y, ...),
              list(rate = 1),
              "Exponentielle   = 1")
> mh_rexpo <- mhsampler(expo, gauss)
> rand(mh_rexpo, 25)
 [1] 0.0000000 0.6922293 0.6922293 0.6922293 0.6922293 0.8097684 0.8097684
 [8] 0.8097684 1.5030231 1.5030231 1.5030231 2.1219724 1.5705083 2.2731373
[15] 2.5026679 1.7453287 1.5488444 0.4250888 0.4250888 0.7718751 0.7718751
[22] 0.7718751 0.7718751 0.2289032 0.2289032
```

Indices

— `do.call` : Pour appeller une fonction avec une `list` d'arguments.
— `...` : Faites-en un usage judicieux !