

# An R Package for Fast Sampling from the von Mises–Fisher Distribution

Aristide Houndetouungan

2025-11-09

## Introduction

The package **vMF** simulates von Mises–Fisher distribution ( $\mathcal{M}$ ). Unlike the package **mvMF** (Hornik and Grün, 2014), which simulates and estimates mixtures of  $\mathcal{M}$ , **vFM** performs fast sampling as its source code is written in C++. **vFM** also computes the density and the normalization constant of  $\mathcal{M}$ .

The von Mises–Fisher distribution is used to model coordinates on a hypersphere of dimension  $p \geq 2$ . Roughly speaking, it is the equivalent of the normal distribution on a hypersphere. As the normal distribution,  $\mathcal{M}$  is characterized by two parameters. The location (or mean directional) parameter  $\mu$  around which draws will be concentrated and the intensity parameter  $\eta$  which measures the intensity of concentration of the draws around  $\mu$ . The higher  $\eta$ , the more the draws are concentrated around  $\mu$ . Compared to the normal distribution,  $\mu$  is similar to the mean parameter of the normal distribution and  $1/\eta$  is similar to the standard deviation.

There are several definitions of the density function of  $\mathcal{M}$ . In this package, the density is normalized by the uniform distribution without loss of generality. This is also the case in Mardia and Jupp (2009) and Hornik and Grün (2013).

Let  $\mathbf{z} \sim \mathcal{M}(\eta, \mu)$ . The density of  $\mathbf{z}$  is given by

$$f_p(\mathbf{z}|\eta, \mu) = C_p(\eta) e^{\eta \mathbf{z}' \mu},$$

where  $C_p(x) = \left(\frac{x}{2}\right)^{\frac{p}{2}-1} \frac{1}{\Gamma\left(\frac{p}{2}\right) I_{\frac{p}{2}-1}(x)}$  is the normalization constant and  $I_*(.)$  the Bessel function of the first kind defined by:

$$I_\alpha(x) = \sum_{m=0}^{\infty} \frac{\left(\frac{x}{2}\right)^{2m+\alpha}}{m! \Gamma(m+\alpha+1)}.$$

The normalization with respect to the uniform distribution implies  $C_p(0) = 1$ .

## Simulation from von Mises Fisher distribution

The following algorithm provides a rejection sampling scheme for drawing a sample from  $\mathcal{M}$  with mean directional parameter  $\mu = (0, \dots, 0, 1)$  and concentration (intensity) parameter  $\eta \geq 0$  (see Section 2.1 in Hornik and Grün, 2014).

- Step 1. Calculate  $b$  using \* Step 1. Calculate  $b$  using

$$b = \frac{p-1}{2\eta + \sqrt{2\eta^2 + (p-1)^2}}.$$

Let  $x_0 = (1-b)/(1+b)$  and  $c = \eta x_0 + (p-1) \log(1-x_0^2)$ .

- Step 2. Generate  $Z \sim Beta((p-1)/2, (p-1)/2)$  and  $U \sim Unif([0, 1])$  and calculate

$$W = \frac{1 - (1+b)Z}{1 - (1-b)Z}.$$

- Step 3. If

$$\eta W + (p-1) \log(1 - x_0 W) - c < \log(U),$$

go to step 2.

- Step 4. Generate a uniform  $(d-1)$ -dimensional unit vector  $\mathbf{V}$  and return

$$\mathbf{X} = \left( \sqrt{1-W^2} \mathbf{V}', W \right)'$$

The uniform  $(d-1)$ -dimensional unit vector  $\mathbf{V}$  can be generated by simulating  $d-1$  independent standard normal random variables and normalizing them so as  $\|\mathbf{V}\|_2 = 1$ . To get sampling from  $\mathcal{M}$  with arbitrary mean direction parameter  $\mu$ ,  $\mathbf{X}$  is multiplied from the left with a matrix where the first  $d-1$  columns consist of unitary basis vectors of the subspace orthogonal to  $\mu$  and the last column is equal to  $\mu$ .

## Comparison of vMF and movMF

In this section, I compare **vMF** and **movMF**.

```
library(rbenchmark)

fcompare <- function(n) {
  benchmark("vMF" = rvMF(n, c(1,0,0)), "movMF" = rmovMF(1,c(1,0,0)))
}

fcompare(1)
#>   test replications elapsed relative user.self sys.self user.child sys.child
#> 2 movMF       100  0.024      24    0.024      0       0       0
#> 1  vMF        100  0.001       1    0.001      0       0       0

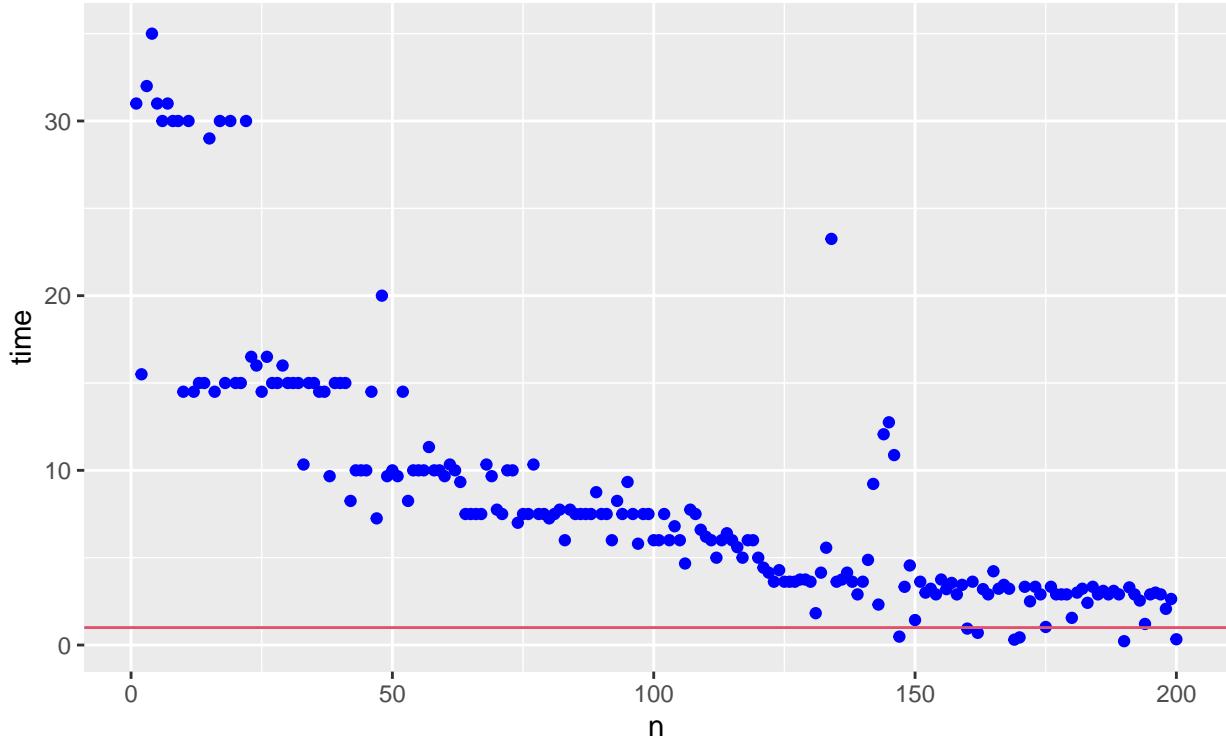
fcompare(10)
#>   test replications elapsed relative user.self sys.self user.child sys.child
#> 2 movMF       100  0.025     12.5    0.024      0       0       0
#> 1  vMF        100  0.002      1.0    0.002      0       0       0

fcompare(100)
#>   test replications elapsed relative user.self sys.self user.child sys.child
#> 2 movMF      100  0.025     3.571    0.025      0       0       0
#> 1  vMF       100  0.007     1.000    0.008      0       0       0
```

**vMF** performs over **movMF**. The performance of **vMF** is much better when only few simulations are performed. When the sample is too large, the two package require approximately the same running time.

```
out <- unlist(lapply(1:200, function(x) fcompare(x)$elapsed[1]/fcompare(x)$elapsed[2]))

library(ggplot2)
ggplot(data = data.frame(n = 1:200, time = out), aes(x = n, y = time)) +
  geom_point(col = "blue") + geom_hline(yintercept = 1, col = 2)
```



Many papers use simulations from the von-Mises Fisher distribution in a Markov Chain Monte Carlo (MCMC) process. A single draw is performed at each iteration of the MCMC. This is for example the case in [Boucher and Houndetoungan \(2022\)](#), [Breza et al. \(2020\)](#), [McCormick and Zheng \(2015\)](#). In such a simulation context, using **vMF** would take much less time than **movMF**. For example, I consider the process  $(\mathbf{z}_t)_{t \in \mathbb{N}}$  which follows a random walk of the von-Mises Fisher distribution. The first variable,  $\mathbf{z}_0$ , is randomly set on a 4-dimensional hypersphere and  $\mathbf{z}_t \sim \mathcal{M}(1, \mathbf{z}_{t-1}) \forall t > 0$ . Simulating this process has about the same complexity as using von-Mises Fisher drawings in an MCMC.

```
set.seed(123)
P           <- 4
initial      <- rmovMF(1, rep(0, P))
# Fonction based on vMF to simulate theta
SamplevMF    <- function(n) {
  output       <- matrix(0, n + 1, P)
  output[1, ]   <- initial
  for (i in 1:n) {
    output[i + 1,] <- rvMF(1, output[i,])
  }
  return(output)
}

# Fonction based on movMF to simulate theta
SamplemovMF  <-function(n){
  output       <- matrix(0, n + 1, P)
  output[1, ]   <- initial
  for (i in 1:n) {
    output[i + 1,] <- rmovMF(1, output[i,])
  }
  return(output)
}
benchmark("vMF" = SamplevMF(1000), "movMF" = SamplemovMF(1000))
```

```
#>      test replications elapsed relative user.self sys.self user.child sys.child
#> 2  movMF        100   35.605    51.902     35.260     0.032         0         0
#> 1  vMF         100    0.686     1.000     0.643     0.044         0         0
```

The comparison of the running times **vMF** is less time-consuming

## References

- Boucher, V. and Houndetoungan, E. A. (2022). Estimating peer effects using partial network data.
- Breza, E., Chandrasekhar, A. G., McCormick, T. H., and Pan, M. (2020). Using aggregated relational data to feasibly identify network structure without network data. *American Economic Review*, 110(8):2454–84.
- Hornik, K. and Grün, B. (2013). On conjugate families and jeffreys priors for von mises–fisher distributions. *Journal of statistical planning and inference*, 143(5):992–999.
- Hornik, K. and Grün, B. (2014). movmf: an r package for fitting mixtures of von mises–fisher distributions. *Journal of Statistical Software*, 58(10):1–31.
- Mardia, K. V. and Jupp, P. E. (2009). *Directional statistics*, volume 494. John Wiley & Sons.
- McCormick, T. H. and Zheng, T. (2015). Latent surface models for networks using aggregated relational data. *Journal of the American Statistical Association*, 110(512):1684–1695.