UNIVERSITY OF
COPENHAGEN

Random Number Generation and Rejection Sampling

Computational Statistics

Johan Larsson

Department of Mathematical Sciences, University of Copenhagen

March 31, 2026

Today

Leftovers From Last Lecture
Efficient computations of smoothing splines

Pseudo-Random Numbers
How do you simulate random numbers in R?

Rejection Sampling
General and useful method for sampling from a target distribution

Recall, that our smoothing spline estimate is

$$\boldsymbol{f} = \boldsymbol{\Phi}\beta$$

with $\boldsymbol{\Phi}_{ij} = \varphi_j(x_i)$, and

$$\begin{aligned} L(\boldsymbol{f}) &= (\boldsymbol{y} - \boldsymbol{f})^T(\boldsymbol{y} - \boldsymbol{f}) + \lambda \|f''\|_2^2 \\ &= (\boldsymbol{y} - \boldsymbol{\Phi}\beta)^T(\boldsymbol{y} - \boldsymbol{\Phi}\beta) + \lambda \beta^T \boldsymbol{\Omega}\beta \end{aligned}$$

with

$$\boldsymbol{\Omega}_{jk} = \int \varphi_j''(z)\varphi_k''(z)\mathrm{d}z.$$

Solution
The solution is

$$\hat{\beta} = (\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \lambda\boldsymbol{\Omega})^{-1}\boldsymbol{\Phi}^T\boldsymbol{y}.$$

When n is large, computing $\boldsymbol{S}_\lambda \boldsymbol{y}$ directly is expensive.

In practice, we therefore often use fewer basis functions than data points, $p < n$.

Singular Value Decomposition

Using the singular value decomposition

$$\boldsymbol{\Phi} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^T$$

we can rewrite the smoother matrix as

$$\boldsymbol{S}_\lambda = \widetilde{\boldsymbol{U}}(I + \lambda \operatorname{diag}(\boldsymbol{\gamma}))^{-1}\widetilde{\boldsymbol{U}}^T$$

where $\widetilde{\boldsymbol{U}} = \boldsymbol{U}\boldsymbol{W}$ and

$$\boldsymbol{D}^{-1}\boldsymbol{V}^T\boldsymbol{\Omega}\boldsymbol{V}\boldsymbol{D}^{-1} = \boldsymbol{W}\operatorname{diag}(\boldsymbol{\gamma})\boldsymbol{W}^T.$$

The columns of $\widetilde{\boldsymbol{U}}$ form the **Demmler-Reinsch basis**.

Interpretation

Project the data y onto the Demmler-Reinsch basis (columns of $\widetilde{\boldsymbol{U}}$) to get coefficients $\hat{\boldsymbol{\beta}} = \widetilde{\boldsymbol{U}}^T \boldsymbol{y}$.

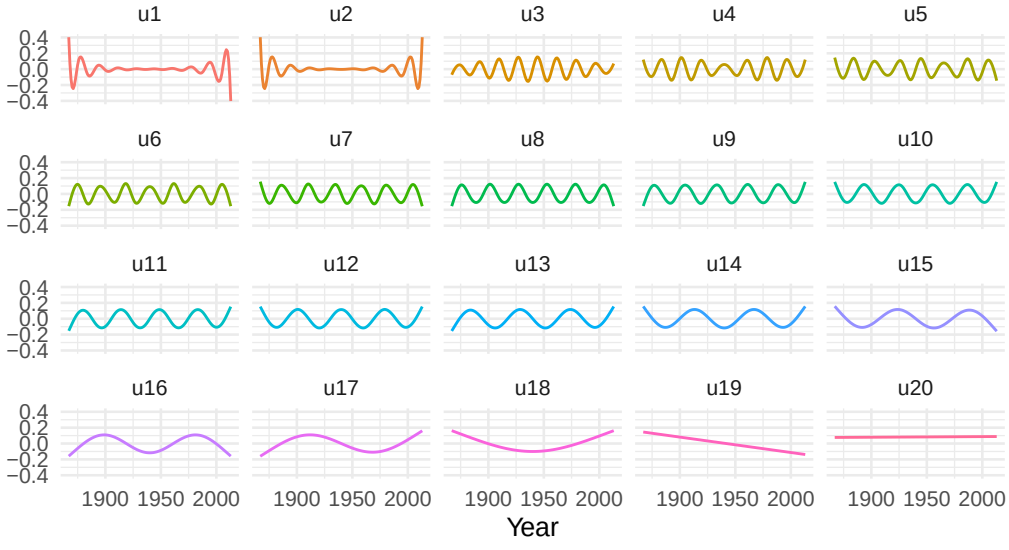
Each coefficient is shrunk according to the corresponding eigenvalue γ_i and smoothing parameter λ:

$$\hat{\boldsymbol{\beta}}_i(\lambda) = \frac{\hat{\beta}_i}{1 + \lambda\gamma_i}.$$

The final smoothed values are reconstructed by combining the shrunk coefficients with the basis functions.

$$\hat{\boldsymbol{f}} = \widetilde{\boldsymbol{U}}\hat{\boldsymbol{\beta}}(\lambda)$$

The Demmler-Reinsch Basis (Columns of $\widetilde{U}$)



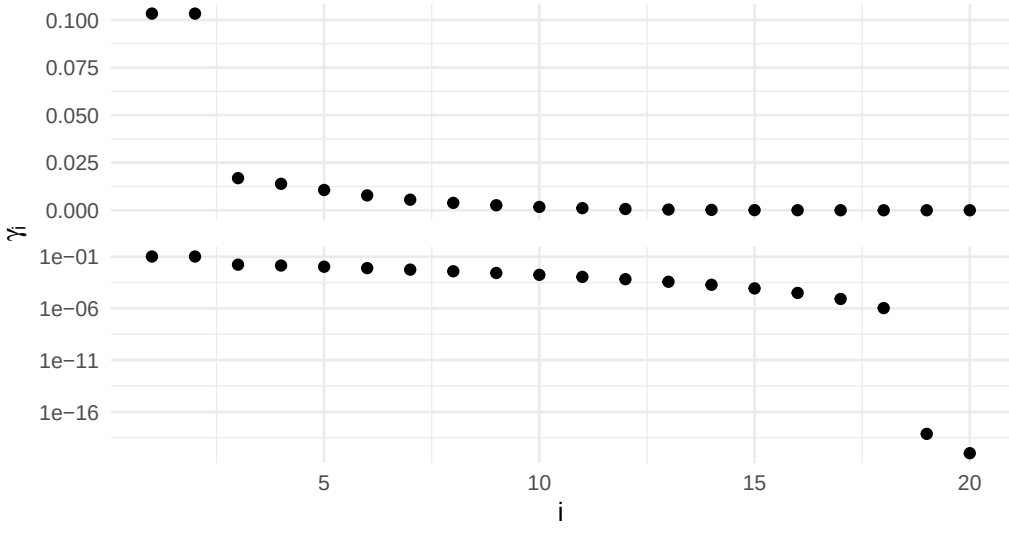


Figure 1: The eigenvalues γ_i of the penalty matrix in the Demmler-Reinsch basis.

Pseudo-Random Numbers

Computers have no concept of "true" randomness and instead generate **pseudo-random** numbers.

Not really random, but appear to be so.

A research field in itself, see `?RNG` in R for available algorithms.

Mersenne Twister
The default in R, which generates integers in the range

$$\{0, 1, \dots, 2^{32} - 1\}.$$

Long period: all combinations of consecutive integers up to dimension 623 occur equally often in a period.

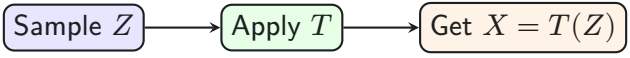
Uniform Random Variables i n R
`runif()` samples by just scaling these pseduo-random integers to the unit interval (with some tweaks to avoid division by 0).

Transformation Methods

What if we want to sample from a different distribution?

We could use **the transformation method**, which relies on the following fact:

If $T : \mathcal{Z} \to \mathbb{R}$ is a map and $Z \in \mathcal{Z}$ is a random variable we can sample, then we can sample $X = T(Z)$.



Transformation methods let us generate samples from complex distributions by applying a function to samples from a simpler distribution.

Example

To sample $X \sim \mathrm{Exp}(\lambda)$, use $X = -\log(U)/\lambda$ with $U \sim \mathrm{Uniform}(0, 1)$.

Inversion Sampling

A type of **transformation** sampling.

If $F^{-1} : (0, 1) \mapsto \mathbb{R}$ is the generalized inverse of a distribution function and U is uniformly distributed on $(0, 1)$ then

$$F^{-1}(U)$$

has distribution function F.

Computing F^{-1}: easy for discrete distributions, but hard for continuous ones.

Gaussian Random Variables

Box–Muller Method
A transformation of two independent uniforms into two independent Gaussian random variables (polar coordinates).

Inverse Transform
$X = \Phi^{-1}(U)$ where Φ is the distribution function for the Gaussian distribution.

Rejection Sampling
See Exercise 5.1 in CSwR or the Ziggurat algorithm.

Computing Φ^{-1}

Recall that

$$\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-z^2/2} \mathrm{d}z.$$

But Φ^{-1} (the quantile function) does not have a closed-form expression.

We need to approximate Φ^{-1} numerically.

This, together with this technical approximation of Φ^{-1} is how R generates samples from $\mathcal{N}(0, 1)$.

Sampling in R

R has three basic options for customizing the behavior, set through `RNGkind()`.

```
RNGkind()
```

```
[1] "Mersenne-Twister" "Inversion"         "Rejection"
```

Options

- `kind`: type of pseudo-random number generator
- `normal.kind`: type of sampler for Normal samples
- `sample.kind`: type of sampler for `sample()`

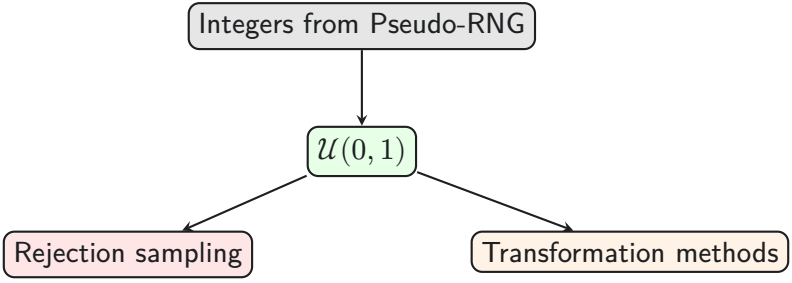


Figure 2: Main categories of sampling methods (in this course).

Example: Sampling from a t-Distribution

If we let

$$T(z, w) = \frac{z}{\sqrt{w/k}},$$

and take $Z \sim \mathcal{N}(0, 1)$ and $W \sim \chi^2_k$, then

$$X = T(Z, W) = \frac{Z}{\sqrt{W/k}} \sim t_k.$$

This transformation is used in R and other software to simulate t-distributed random variables.

Algorithm 1: Transformation sampling for the t-distribution

$z \leftarrow$ sample from $\mathcal{N}(0, 1)$;
$w \leftarrow$ sample from χ^2_k;
return $z/\sqrt{w/k}$;

Von Mises Distribution

The density on $(-\pi, \pi]$ is

$$f(x) = \frac{1}{2\pi I_0(\kappa)} \, e^{\kappa \cos(x-\mu)}$$

for $\kappa > 0$ and $\mu \in (-\pi, \pi]$ parameters and I_0 is the modified Bessel function.

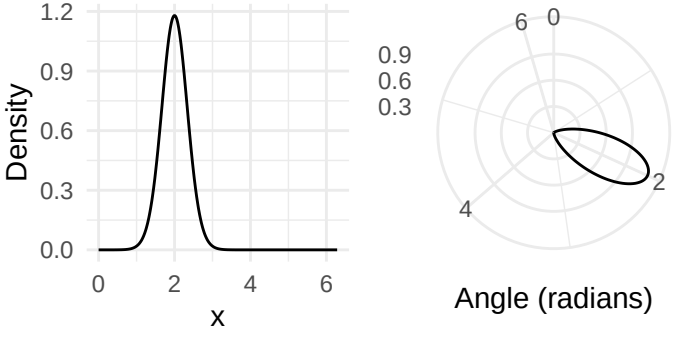


Figure 3: The von Mises density for $\mu = 2$ and $\kappa = 9$.

Using movMF

```r
library(movMF)
xy <- rmovMF(500, 0.5 * c(cos(-1.5), sin(-1.5)))

# rmovMF represents samples as elements on the unit circle
x <- acos(xy[, 1]) * sign(xy[, 2])
```

Problem

Can we sample via the transformation method? Yes, but not easily. There is no closed-form expression for the quantile function.

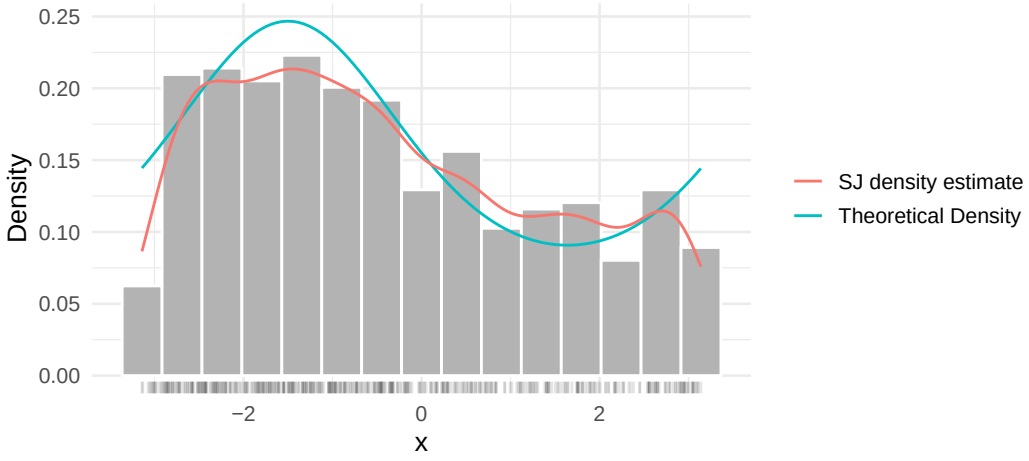


Figure 4: Histogram of samples from rmovMF()

Rejection Sampling

We want to sample from a target distribution $f(y)$ but have no readily available transform T for our proposal $g(y)$.

Let $Y_1, Y_2, ...$ be i.i.d. with density g on $\mathbb{R}$ and $U_1, U_2, ...$ be i.i.d. uniform and independent of Y_i for all i.

Define

$$k = \inf\left\{ n \geq 1 \mid U_n \leq \alpha \frac{f(Y_n)}{g(Y_n)} \right\},$$

for $\alpha \in (0, 1]$, where f is a density.

Theorem
If $f(y) \leq g(y)/\alpha$ for all $y \in \mathbb{R}$, then the distribution of Y_k has density f.

α is the **acceptance probability** and $g(y)/\alpha$ the **envelope** of f.

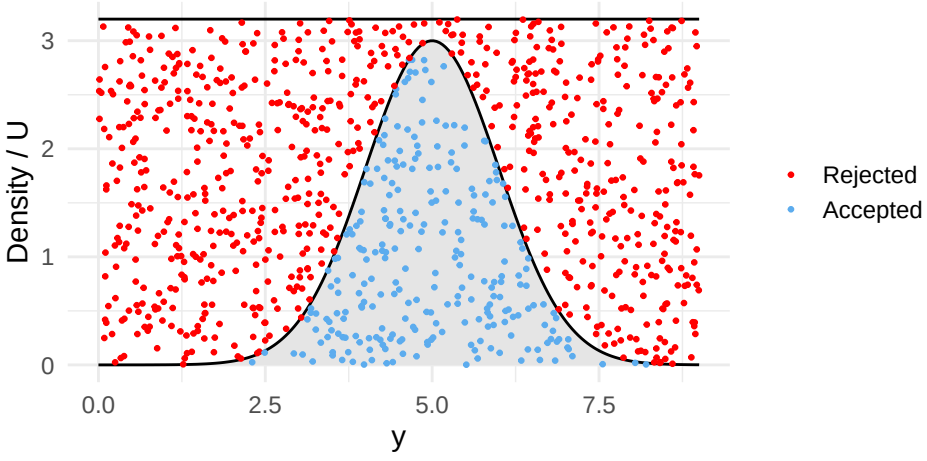


Figure 5: Illustration of rejection sampling for a normal target density, using a uniform proposal.

Normalizing Constants

Rejection sampling works with unnormalized densities!

Suppose:

- Target: $f(y) = cq(y)$ (unknown c)
- Proposal: $g(y) = dp(y)$ (unknown d)

Acceptance test:

$$U \leq \frac{\alpha f(y)}{g(y)} = \frac{\alpha c q(y)}{d p(y)}$$

If $\alpha = \frac{\alpha' d}{c}$, then

$$\frac{\alpha f(y)}{g(y)} = \frac{\alpha' q(y)}{p(y)}$$

Takeaway

You do not need to know c or d!

How to Find α'?

Given functions q and p, we need to find α' such that

$$\alpha' q(y) \leq p(y) \quad \text{for all } y.$$

How?
We can try to solve the problem

$$\alpha' = \inf_{y:q(y)>0} \frac{p(y)}{q(y)} > 0.$$

Optimally, find the minimum analytically.

Otherwise, numerically via optimization (e.g. `optimize()` in R).

Von Mises Rejection Sampling

Rejection sampling using the uniform proposal, $g(y) \propto 1$.

Since

$$e^{\kappa(\cos(y)-1)} = \alpha' e^{\kappa \cos(y)} \leq 1,$$

where $\alpha' = \exp(-\kappa)$, we reject if

$$U > e^{\kappa(\cos(Y)-1)}.$$

Algorithm 2: Rejection sampling for the von Mises distribution

repeat

  Generate $Y \sim \text{Uniform}(-\pi, \pi)$;

  Generate $U \sim \text{Uniform}(0, 1)$;

  if $U \leq e^{\kappa(\cos(Y)-1)}$ **then**

    Accept Y;

until *sample is accepted*;

Implementation

```
get_one_sample_vmises <- function(kappa) {
  repeat {
    Y <- runif(1, -pi, pi)
    U <- runif(1)
    if (U <= exp(kappa * (cos(Y) - 1))) {
      break
    }
  }
  Y
}
```

```
sample_vonmises_slow <- function(n, kappa) {
  replicate(n, get_one_sample_vmises(kappa))
}
```

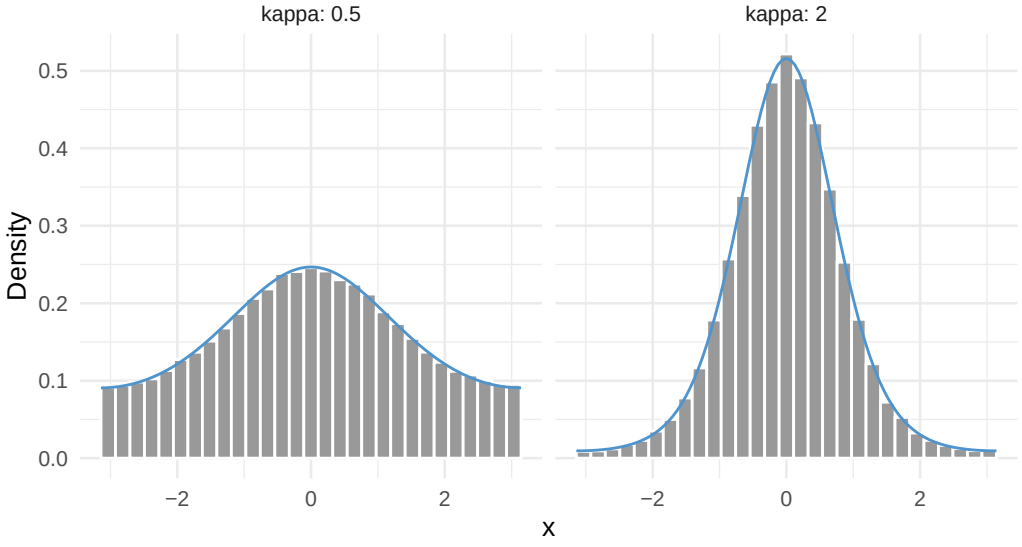


Figure 6: Histogram of samples from `sample_vonmises_slow()`, with theoretical density overlaid.

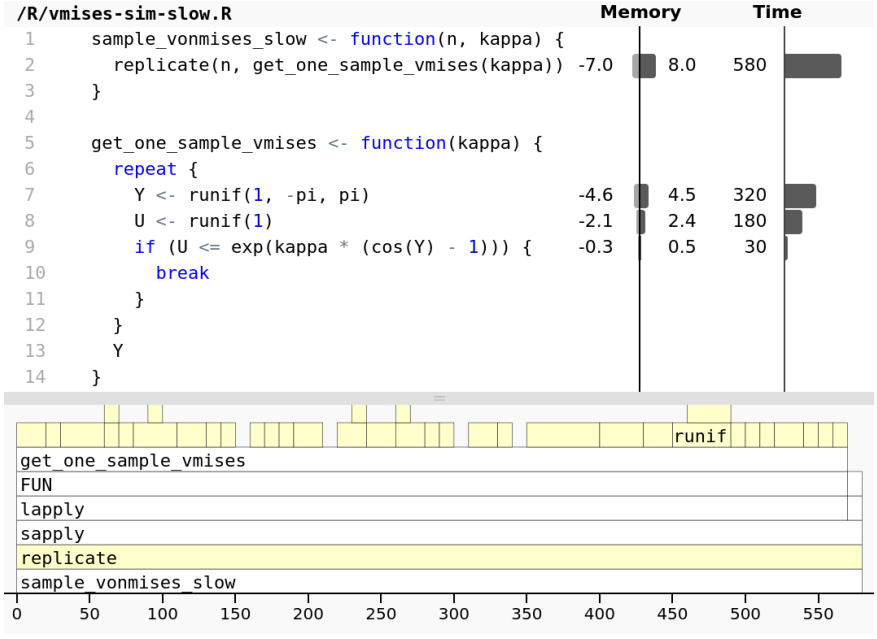


Figure 7: A line profile of the naive von Mises sampling implementation.

Findings from Profiling

The bottleneck is the repeated calls to `runif(1)`.

Both because of the for loop overhead, but also because calling the RNG for a single number is inefficient.

It is faster to generate all random numbers once and store them in a vector.

But how to do that for rejection sampling with an unknown number of rejections?

Vectorized (but Random Length) Rejection Sampler

We can speed up the rejection sampler by generating m proposal samples at once.

```
fixed_vonmises_sampler <- function(m, kappa) {
  Y <- runif(m, -pi, pi)
  u <- runif(m)
  accept <- u <= exp(kappa * (cos(Y) - 1))
  Y[accept]
}
```

Simple and fast, but returns a **random** number of samples!

We need a function that returns exactly n samples. But we don't know α in advance.

Algorithm 3: Sampling exactly n values via rejection sampling with unknown acceptance probability α.

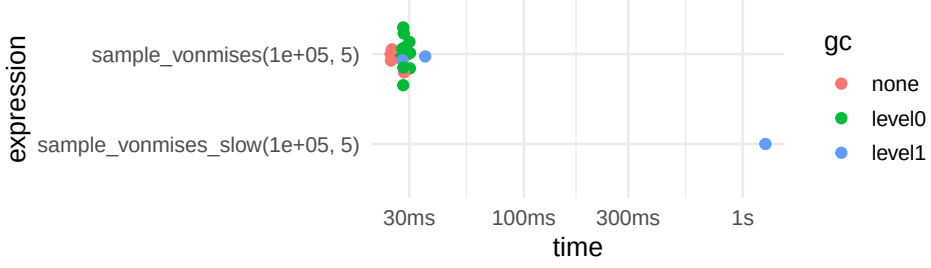
$\alpha \leftarrow 1$;
$\boldsymbol{y} \leftarrow$ empty vector;
$n_{\text{accepted}} \leftarrow 0$;
while $n_{accepted} < n$ **do**
  $m \leftarrow \lceil (n - n_{\text{accepted}})/\alpha \rceil$;
  $\boldsymbol{y}_{\text{new}} \leftarrow$ Generate m candidate samples $x_1, \ldots, x_m$;
  $n_{\text{accepted}} \leftarrow n_{\text{accepted}} +$ length$(\boldsymbol{y}_{\text{new}})$;
  if *first batch* **then**
    $\alpha \leftarrow (n_{\text{accepted}} + 1)/(m + 1)$ `// estimate acceptance probability`
  Append $\boldsymbol{y}_{\text{new}}$ to $\boldsymbol{y}$;
return $\boldsymbol{y}_{1:n}$

Function Factory for Rejection Sampling

```
new_rejection_sampler <- function(generator) {
  function(n, ...) {
    alpha <- 1
    y <- numeric(0)
    n_accepted <- 0
    while (n_accepted < n) {
      m <- ceiling((n - n_accepted) / alpha)
      y_new <- generator(m, ...)
      n_accepted <- n_accepted + length(y_new)
      if (length(y) == 0) {
        alpha <- (n_accepted + 1) / (m + 1)
      }
      y <- c(y, y_new)
    }
    list(x = y[seq_len(n)], alpha = alpha)
  }
}
```

```r
sample_vonmises <- new_rejection_sampler(fixed_vonmises_sampler)

bench::mark(
  sample_vonmises(1e5, 5),
  sample_vonmises_slow(1e5, 5),
  check = FALSE
) |>
  plot()
```



Adaptive Envelopes

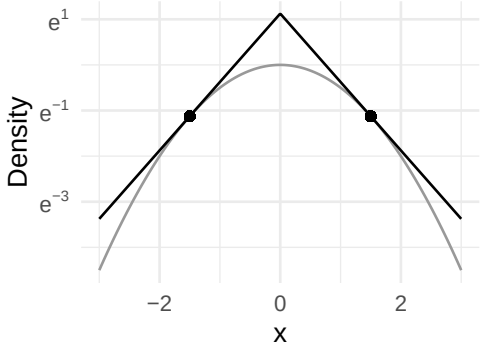
When f is *log-concave* on I we can construct bounds of the form

$$f(x) \leq e^{V(x)}$$

where

$$V(x) = \sum_{i=1}^{m} (a_i x + b_i) \mathbf{1}_{I_i}(x)$$

for intervals I_i forming a partition of I.



Typically, $a_i x + b_i$ is tangent to the graph of $\log(f)$ at $x_i \in I_i = (z_{i-1}, z_i]$ for

$$z_0 < x_1 < z_1 < x_2 < ... < z_{m-1} < x_m < z_m.$$

Beta Distribution

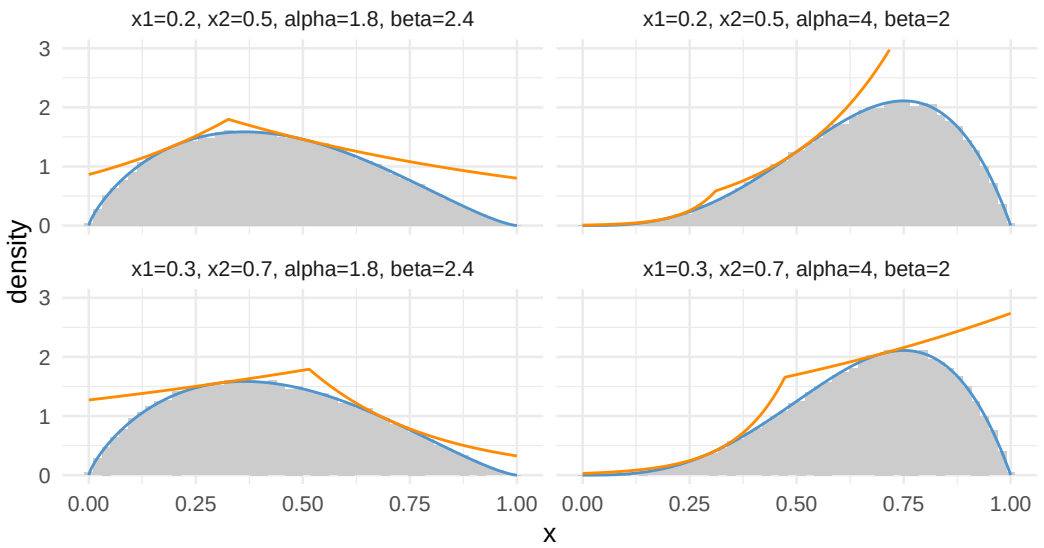


Figure 8: Histograms of samples from an adaptive envelope sampler with theoretical densities (blue) and envelopes (orange) overlaid.

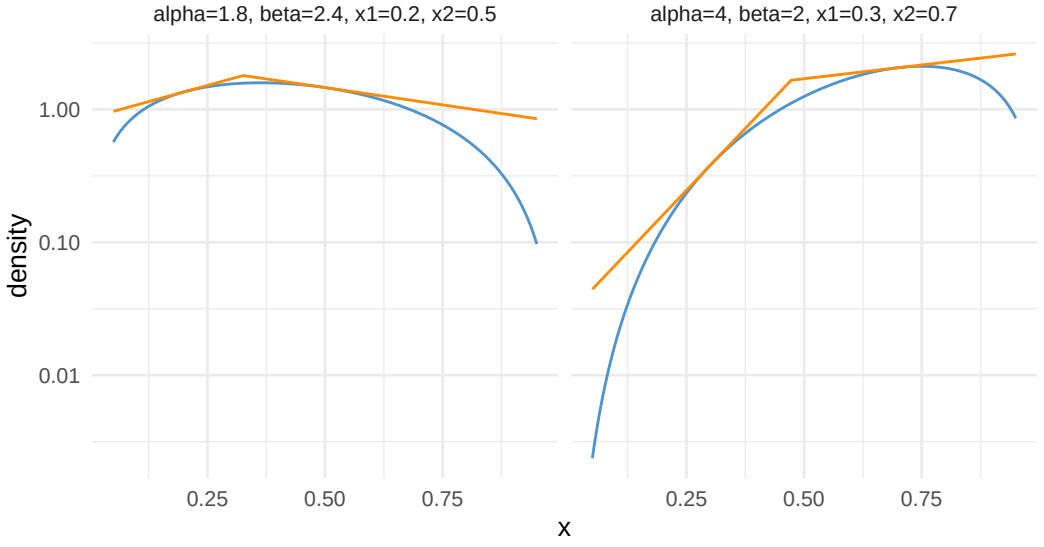


Figure 9: Theoretical densities (blue) and envelopes (orange) for different parameter values and choices of x_1 and x_2.

Von Mises Adaptive Envelopes

The von Mises density is unfortunately not globally log-concave.

But, it *is*

- log-concave on $(-\pi, \pi/2)$ and $(\pi/2, \pi)$ and
- log-convex on $(-\pi, -\pi/2)$ and $(-\pi/2, \pi/2)$.

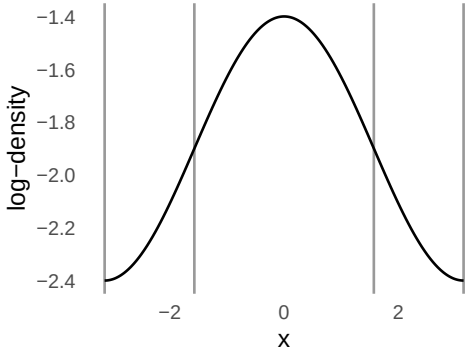


Figure 10: The von Mises density with $\kappa = 0.5$.

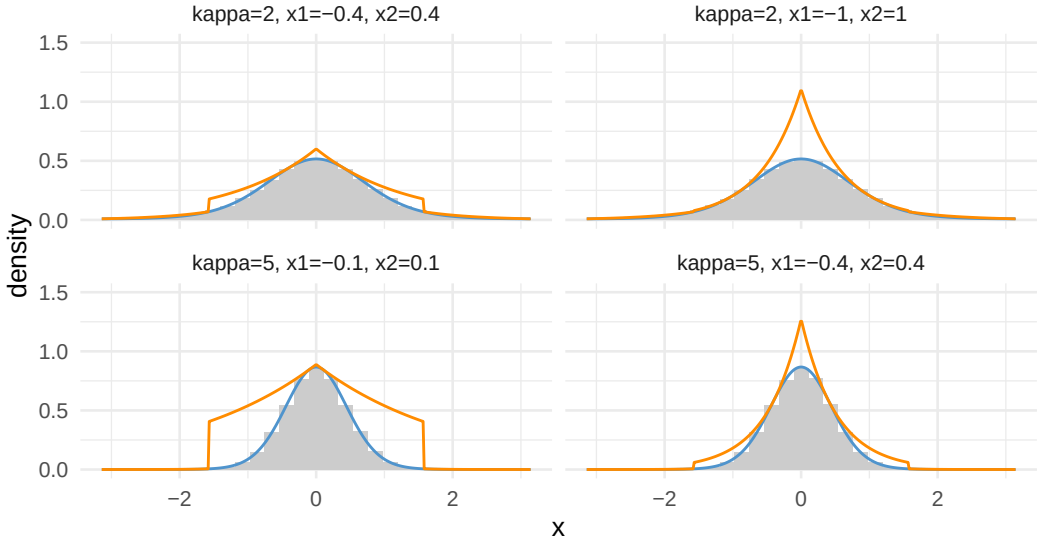


Figure 11: Histograms of samples from an adaptive envelope sampler for the von Mises distribution, with theoretical densities (blue) and envelopes (orange) overlaid.

Caveats about Adaptive Envelopes

Adaptive envelopes is a general and powerful technique, but tricky to implement!

Efficiency depends on number of rejections and also quality of implementation.

A good implementation should make use of new points to update envelope.

Transformation Methods vs. Rejection Sampling

Transformation Methods

Transform samples from a simple distribution to the target distribution.

Efficient if the transformation is available and cheap to compute.

But may not work for complex or unnormalized distributions!

Rejection Sampling

Uses a proposal distribution and accepts/rejects samples.

Works even if quantile function is unavailable or expensive.

Often less efficient, but very general.

Useful for complex, high-dimensional, or unnormalized distributions.

Takeaway

Use transformation methods when possible for efficiency; use rejection sampling when transformation is impractical or impossible.

Exercises

Step 1

Use the fact that

$$W = X_1^2 + ... + X_k^2 \sim \chi_k^2 \quad \text{for} \quad X_1, ..., X_k \sim_{\text{i.i.d.}} \mathcal{N}(0, 1)$$

to implement a function, `my_rchisq()`, such that `my_rchisq(n, k)` returns a sample of n i.i.d. observations from χ_k^2.

Step 2

Make another function `my_other_rchisq()` that uses inverse sampling.

Step 3

Benchmark your implementations against one another and `rchisq()`.