Vertical Weighted Strips in R

Andrew M. Raim*, **James A. Livsey, and Kyle M. Irimata**
Center for Statistical Research and Methodology, U.S. Census Bureau
andrew.raim@census.gov andrewraim.github.io

United States Census Bureau

Summary

- Vertical Weight Strips (VWS) is a method of constructing proposals for rejection sampling (Raim, Livsey, and Irimata, 2024+).
- The target density is decomposed into a weight function and base density. The weight function is then majorized and combined with the base density to obtain a distribution which is more convenient to sample from.
- This work is developing an R / C++ package to facilitate the development of VWS proposals, currently focusing on univariate targets.
- This poster contains four pages: **Page 1** reviews the VWS method and the concept for package design, **Page 2** introduces an illustrative example, **Page 3** describes the R interface, and **Page 4** describes the C++ interface.

Target

Generate draws from the **weighted** target density

$$f(x) = w(x)g(x)/\psi, \quad \psi = \int_\Omega w(x)g(x)d\nu(x), \quad \text{where}$$

1. Ω is the support of f,
2. $w(x)$ is a nonnegative weight function,
3. $g(x)$ is a density function ("base distribution") with $\Omega \subseteq \operatorname{supp} g$,
4. ψ is a normalizing constant which may not have a convenient form.

Proposal Distribution

- Partition Ω into N disjoint regions $\mathcal{D}_1, \ldots, \mathcal{D}_N$.
- Suppose $\overline{w}_j$ is a function which majorizes w on $\mathcal{D}_j$ so that $w(x) \le \overline{w}_j(x)$.
- This suggests an envelope for rejection sampling:

$$h_0(x) = \sum_{j=1}^N \overline{w}_j(x)g(x)\,\mathrm{I}\{x \in \mathcal{D}_j\} \quad \Longrightarrow \quad w(x)g(x) \le h_0(x).$$

- Normalizing yields a finite mixture of truncated and reweighted densities

$$h(x) = h_0(x)/\psi_N = \sum_{j=1}^N \pi_j g_j(x),$$

with $\psi_N = \sum_{j=1}^N \overline{\xi}_j$ where $\overline{\xi}_j = \int_{\mathcal{D}_j} \overline{w}_j(x)g(x)d\nu(x)$, and

$$\pi_j = \overline{\xi}_j \Big/ \sum_{\ell=1}^N \overline{\xi}_\ell, \quad g_j(x) = \overline{w}_j(x)g(x)\,\mathrm{I}(x \in \mathcal{D}_j) \Big/ \overline{\xi}_j.$$

Rejection Sampling

Rejection sampling algorithm with h as the proposal.

1. Draw u from Uniform$(0,1)$ and x from h.
2. If $u \le f_0(x)/h_0(x)$, accept x as a draw from f; else repeat Step 1.

Some properties of the sampler.

- The probability of rejecting each proposed draw is $1 - \psi/\psi_N$.
- An upper bound for the probability of rejection is

$$1 - \frac{\sum_{j=1}^N \underline{\xi}_j}{\sum_{j=1}^N \overline{\xi}_j}. \tag{1}$$

Here, $\underline{\xi}_j = \int_{\mathcal{D}_j} \underline{w}_j(x)g(x)d\nu(x)$ and $\underline{w}_j$ a minorizer for w on $\mathcal{D}_j$. This is useful when ψ is difficult to compute.

Refining / Adapting the Proposal

- We decompose Ω into $\mathcal{D}_1, \ldots, \mathcal{D}_N$ before sampling, with N prespecified.
- Sequentially partition the support by selecting a region and bifurcating at midpoint. Regions are selected with probability proportional to bound (1).

Two Majorizers

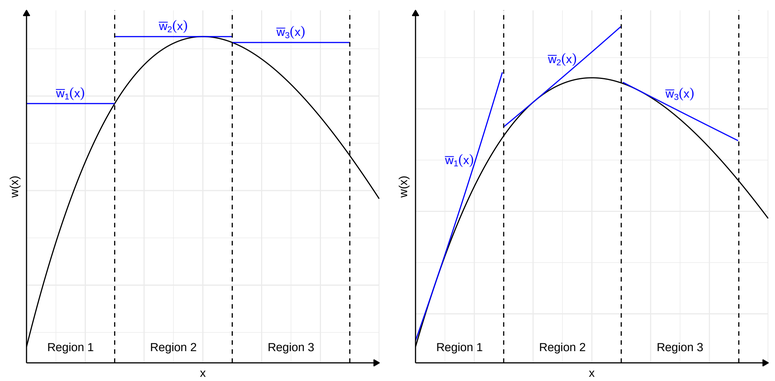


Figure. Two majorizers (left) based on constants $\overline{w}_j$ and (right) linear functions $\log \overline{w}_j(x) = \overline{\beta}_{0j} + \overline{\beta}_{1j}x$.

We focus on univariate targets with regions of the form $\mathcal{D}_j = (\alpha_{j-1}, \alpha_j]$, where $\alpha_0 < \alpha_1 < \cdots < \alpha_N$ and $\Omega \equiv (\alpha_0, \alpha_N]$.

Majorizers

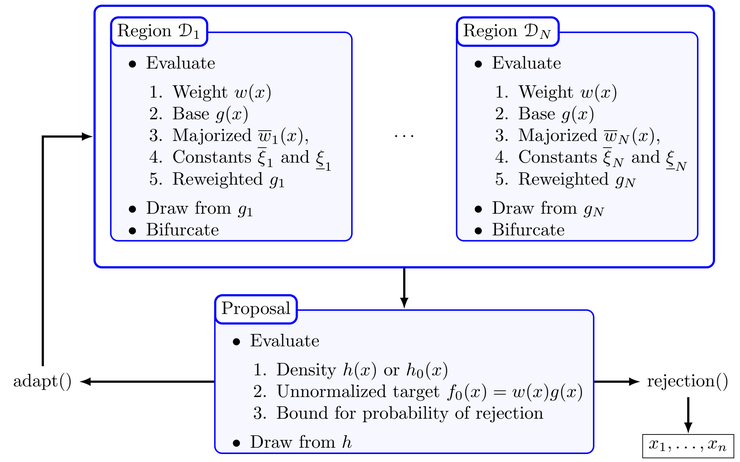
- The constant majorizer can be automated given the CDF and quantile function of g. Numerical optimization can be used to obtain $\overline{w}_j$ terms.
- The vertical strips method (Martino et al., 2018, §3.6) is a special case with f decomposed into uniform g and w includes everything else.
- The linear majorizer can be used when w is either log-concave or log-convex on each $\mathcal{D}_j$. Here we get

$$\overline{\xi}_j = \int_{\mathcal{D}_j} \exp\{\overline{\beta}_{j0} + x\overline{\beta}_{j1}\}g(x)d\nu(x).$$

When the above is tractable and g_j is easy to sample, the resulting proposal can be much more efficient than under the constant majorizer.

- Examples with g as a exponential family density are often tractable.

Concept for Package



- The user specifies a decomposition, a majorizer, and other VWS logic in a `Region` class.
- A `Proposal` is a collection of `Region` objects that represents mixture h.
- `Proposal` can be refined and used in rejection sampling.
- The `vws` R package which implements this design is under development.

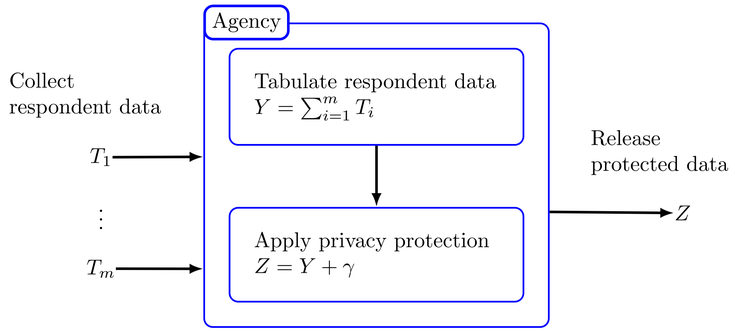
Vertical Weighted Strips in R

Andrew M. Raim*, **James A. Livsey, and Kyle M. Irimata**
Center for Statistical Research and Methodology, U.S. Census Bureau
andrew.raim@census.gov andrewraim.github.io

United States Census Bureau

An Example

- Suppose an agency collects sensitive data from respondents and releases a privacy-protected version to the public.
- Let Y represent a tabulation of respondents' sensitive data and $Z = Y + \gamma$ be a protected version of Y with random noise γ added by the agency which is suitable for release.
- Differential privacy (DP) offers frameworks to guarantee privacy under particular noise mechanisms (Dwork and Roth, 2014). This is an active area of interest at the U.S. Census Bureau (e.g., Abowd et al., 2022).
- Our present example focuses on the data user side: to perform inference on Y given an observed $Z = z$.



Setup

- Suppose Y and γ are independently distributed with

$$Y \sim \text{Lognormal}(\mu, \sigma^2) \quad \text{and} \quad \gamma \sim \text{N}(0, \lambda^2).$$

  This setting was considered by Raim (2021) and Irimata et al. (2022).
- The variance λ^2 of the noise mechanism is known and supplied with the noisy data under DP. Also assume that μ and σ^2 are known for now; in practice these need to be learned from the observed data.
- Fix parameters $\mu = 5$, $\sigma^2 = 0.5$, and $\lambda^2 = 100$ and take a single draw from the distribution $[Z, Y]$ yielding $y = 63.21$ and $z = 65.99$.

Target and Factorization

- Target distribution is the conditional of $[Y \mid Z = z]$ given by

$$f(y \mid z) \propto \frac{1}{\lambda\sqrt{2\pi}} \exp\left\{-\frac{1}{2\lambda^2}(z-y)^2\right\} \qquad \leftarrow g(y)$$
$$\times \frac{1}{y} \exp\left\{-\frac{1}{2\sigma^2}(\log y - \mu)^2\right\} \mathrm{I}(y > 0). \qquad \leftarrow w(y)$$

- Here we have decomposed f into weight function $w(y)$ from the Lognormal component and base distribution $g(y)$ as $\text{N}(z, \lambda^2)$.

Weight Function

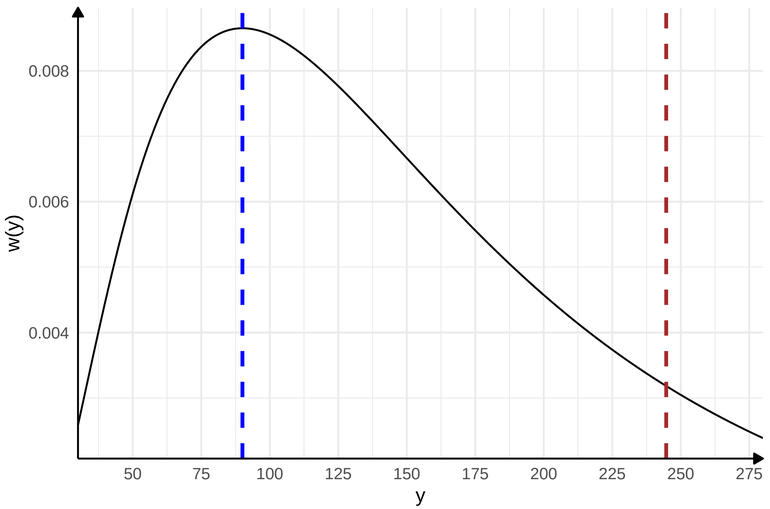


Figure. The weight function $w(y)$.

- Maximum value $\exp(\sigma^2/2 - \mu)$ occurs at dashed blue line $y = \exp(\mu - \sigma^2)$.
- Function changes from log-concave to log-convex at dashed brown line $y = \exp(\mu - \sigma^2 + 1)$.

Constant Majorizer

- The maximum $\overline{w}_j$ and minimum $\underline{w}_j$ of w on $(\alpha_{j-1}, \alpha_j]$ can be found analytically.
- Mixture components g_j for proposal h are densities of $\text{N}(z, \lambda^2)$ truncated to $(\alpha_{j-1}, \alpha_j]$.
- Expressions for $\overline{\xi}_j$ and $\underline{\xi}_j$ are

$$\overline{\xi}_j = \overline{w}_j \left\{\Phi(\alpha_j \mid z, \lambda^2) - \Phi(\alpha_{j-1} \mid z, \lambda^2)\right\},$$
$$\underline{\xi}_j = \underline{w}_j \left\{\Phi(\alpha_j \mid z, \lambda^2) - \Phi(\alpha_{j-1} \mid z, \lambda^2)\right\}.$$

Linear Majorizer

- Start with at least two initial regions

$$\mathcal{D}_1 = (0, \exp(\mu - \sigma^2 + 1)] \quad \text{and} \quad \mathcal{D}_2 = (\exp(\mu - \sigma^2 + 1), \infty]$$

  to ensure any refined regions are entirely concave or convex.
- Can find suitable constants $\overline{\beta}_{0j}$, $\overline{\beta}_{1j}$, $\underline{\beta}_{0j}$, and $\underline{\beta}_{1j}$ so that

$$\overline{w}_j(x) = \exp\{\overline{\beta}_{0j} + \overline{\beta}_{1j} x\},$$
$$\underline{w}_j(x) = \exp\{\underline{\beta}_{0j} + \underline{\beta}_{1j} x\},$$

  are a majorizer and minorizer on $(\alpha_{j-1}, \alpha_j]$.
- Mixture components g_j for proposal h are densities of $\text{N}(z + \lambda^2\overline{\beta}_{1j}, \lambda^2)$ truncated to $(\alpha_{j-1}, \alpha_j]$.
- Expressions for $\overline{\xi}_j$ and $\underline{\xi}_j$ are

$$\overline{\xi}_j = \exp\left\{\overline{\beta}_{0j} + z\overline{\beta}_{1j} + \overline{\beta}_{1j}^2 \lambda^2/2\right\}$$
$$\times \left\{\Phi(\alpha_j \mid z + \overline{\beta}_{1j}\lambda^2, \lambda^2) - \Phi(\alpha_{j-1} \mid z + \overline{\beta}_{1j}\lambda^2, \lambda^2)\right\}, \qquad (2)$$
$$\underline{\xi}_j = \exp\left\{\underline{\beta}_{0j} + z\underline{\beta}_{1j} + \underline{\beta}_{1j}^2 \lambda^2/2\right\}$$
$$\times \left\{\Phi(\alpha_j \mid z + \underline{\beta}_{1j}\lambda^2, \lambda^2) - \Phi(\alpha_{j-1} \mid z + \underline{\beta}_{1j}\lambda^2, \lambda^2)\right\}, \qquad (3)$$

Vertical Weighted Strips in R

Andrew M. Raim*, **James A. Livsey, and Kyle M. Irimata**
Center for Statistical Research and Methodology, U.S. Census Bureau
andrew.raim@census.gov andrewraim.github.io

United States Census Bureau

Constant Majorizer with Numerical Optimization

```r
library(vws)

# Define weight function
w = function(y, log = TRUE) {
    out = -log(y) - (log(y) - mu)^2 / (2*sigma2) + log(y > 0)
    out[y == 0] = -Inf
    if (log) { return(out) } else { return(exp(out)) }
}

# A "helper" for normal base distributions
helper = normal_helper(mean = z, sd = sqrt(lambda2))

# Construct proposal
supp = UnivariateConstRegion$new(a = 0, b = Inf, w, helper)
regions = list(supp)
h_init = FMMProposal$new(regions)

# Refine proposal
adapt_out = adapt(h_init, N = 50 - 1, report = 10)
h = adapt_out$h

# Rejection sampling
ctrl = rejection_control(report = 5000, extra_outputs = TRUE)
out = rejection(h, n = 10000, control = ctrl)

# Get results
y = unlist(out$draws)  # The accepted draws
rejects = out$rejects  # Vector with rejection count for each draw
```

Constant Majorizer with Analytical Optimization

```r
supp = CustomConstRegion$new(a = 0, b = Inf, w, helper)
regions = list(supp)
```

Linear Majorizer

```r
hi = 1e8; y0 = exp(mu - sigma2 + 1)
r1 = CustomLinearRegion$new(a = 0, b = y0, mu, sigma2, z, lambda2)
r2 = CustomLinearRegion$new(a = y0, b = hi, mu, sigma2, z, lambda2)
regions = list(r1, r2)
```

Abstract Region Class

The user extends this class to implement VWS for a sampling problem.

Method	Description
`w`	Compute weight function $w(y)$.
`w_major`	Compute majorized weight function $\overline{w}_j(y)$.
`d_base`	Compute the density function g.
`r`	Generate draws from g_j.
`d`	Compute the density g_j.
`s`	Is point y is in the support of g_j?
`xi_upper`	Compute $\overline{\xi}_j$.
`xi_lower`	Compute $\underline{\xi}_j$.
`bifurcate`	Bifurcate object into two new Region objects.

UnivariateConstRegion Class

- A subclass of `Region` for univariate targets with constant majorizer.
- Uses numerical optimization to compute $\overline{w}_j$ and $\underline{w}_j$.
- Takes a `univariate_helper` that implements base distribution operations: the density, CDF, quantile function, and support indicator.
- Constant $\overline{\xi}_j$ and $\underline{\xi}_j$ are computed automatically from the above.

CustomConstRegion Class

- A subclass of `UnivariateConstRegion` for "Constant Majorizer" on Page 2.
- Overrides numerical optimization with analytical maximization and minimization of w on $\mathcal{D}_j$.

CustomLinearRegion Class

Subclass of `UnivariateConstRegion` based on "Linear Majorizer" on Page 2.

Method	Description
`w`	$w(y) = \frac{1}{y}\exp\left\{-\frac{1}{2\sigma^2}(\log y - \mu)^2\right\}\mathrm{I}(y > 0)$.
`w_major`	$\overline{w}_j(y) = \exp\{\overline{\beta}_{0j} + \overline{\beta}_{1j} y\}$.
`d_base`	Density of $\mathsf{N}(z, \lambda^2)$.
`r`	Draw from $\mathsf{N}(z + \lambda^2\overline{\beta}_{1j}, \lambda^2)$ truncated to $(\alpha_{j-1}, \alpha_j]$.
`d`	Density of $\mathsf{N}(z + \lambda^2\overline{\beta}_{1j}, \lambda^2)$ truncated to $(\alpha_{j-1}, \alpha_j]$.
`s`	$\mathrm{I}\{\alpha_{j-1} < y \leq \alpha_j\}$.
`xi_upper`	Compute $\overline{\xi}_j$ via (2).
`xi_lower`	Compute $\underline{\xi}_j$ via (3).
`bifurcate`	Bifurcate region at the midpoint.

Outputs

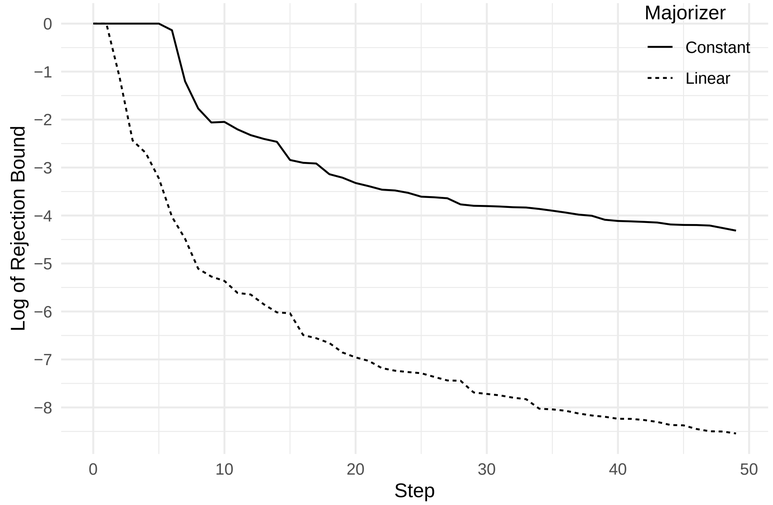


Figure. Bound (1) for probability of rejection (log-scale) for VWS samplers based on constant majorizer and linear majorizer.

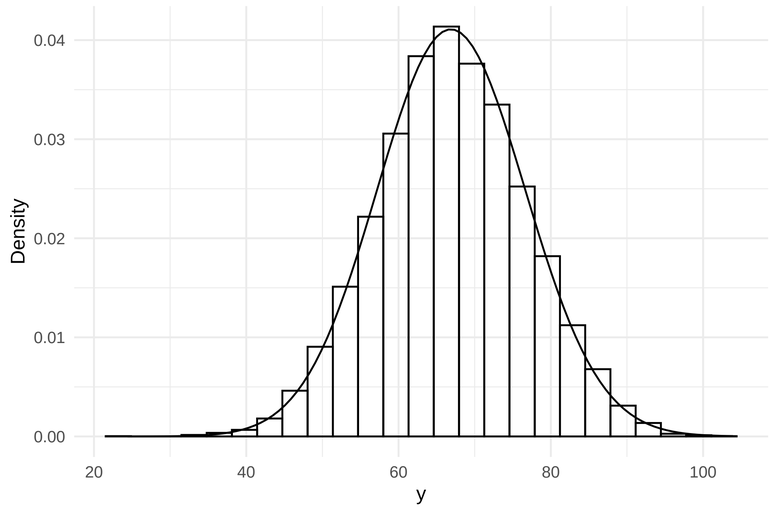


Figure. Empirical distribution of $n = 10{,}000$ draws (histogram) versus target density (solid line).

~~Vertical Weighted Strips in R~~
Vertical Weighted Strips in C++

Andrew M. Raim*, **James A. Livsey, and Kyle M. Irimata**
Center for Statistical Research and Methodology, U.S. Census Bureau
andrew.raim@census.gov andrewraim.github.io

United States Census Bureau

Comments

- The current R interface is based on S3 and R6 (Chang, 2021) for object-oriented programming.
- Sampling problems are programmed by either: (1) inheriting from the abstract `Region` class, or (2) using the `UnivariateConstRegion` and specifying the base distribution function through a "helper".
- The `vws` package also provides an object-oriented C++ interface which is similar to the R interface.
- Samplers may be developed in C++ and used from R via Rcpp (Eddelbuettel, 2013).
- Why (or why not) C++?
  - (+) Performance can be much faster than similar R code.
  - (+) More formal (safer) type handling and object-orientation, including support for template classes & functions.
  - (+) Standard libraries for data structures with efficient memory usage.
  - (−) Has a steeper learning curve than R; generally requires more effort.

Design Principles

- The `vws` package exports C++ interface and code through its `inst/include` folder. User's C++ code includes headers and compiles along with the exported code.
- Use templates to support sampling of doubles, integers, and potentially other composite data types.
- Use STL `function` class to define and pass functions as variables. Functions can use ("capture") variables which are in scope, similarly to functions in R which can use variables in the environment.
- Another R package—currently named `fntl`—is in development to facilitate root-finding, integration, and other function-based operations in C++ with STL `function` objects. It is used for numerical optimization here.

Usage in R via Rcpp

```
R> Rcpp::sourceCpp("sampler.cpp")
R> out = r_ln_norm(n = 10000, z, mu, sigma2, lambda2, N = 10)
2024-05-20 11:54:55 - After 5000 candidates, 4748 accepts
2024-05-20 11:54:55 - After 10000 candidates, 9514 accepts
R> names(out)
[1] "draws"   "rejects"
```

Constant Majorizer with Numerical Optimization (sampler.cpp)

```cpp
// [[Rcpp::depends(vws, fntl)]]
#include "vws.h"
#include "NormalHelper.h"
using namespace vws;

// [[Rcpp::export]]
Rcpp::List r_ln_norm(unsigned int n, double z, double mu,
    double sigma2, double lambda2, unsigned int N)
{
    // Set options for rejection sampling
    rejection_args args;
    args.max_rejects = 10000;
    args.report_period = 5000;

    // Define w(x) as a C++ lambda within an STL function
    std::function<double(double, bool)> w =
    [&](double x, bool log = true) {
        double out = R_NegInf;
        if (x > 0) {
            out = -std::log(x) -
                std::pow(std::log(x) - mu, 2) / (2 * sigma2);
        }
        return log ? out : std::exp(out);
    };

    // Define initial partition
    NormalHelper helper(z, lambda2);
    UnivariateConstRegion supp(0, R_PosInf, w, helper);

    // Construct the proposal. Template arguments are:
    // 1. type of object being sampled,
    // 2. type of Region that specifies VWS logic.
    FMMProposal<double, UnivariateConstRegion> h({ supp });

    // Refine proposal
    h.adapt(N - 1);

    // Rejection sampling
    auto out = rejection(h, n, args);

    // Wrap result to Rcpp::List and return
    return Rcpp::wrap(out);
}
```

Optimization with fntl

C++ code to compute the MLE of $X_1, \ldots, X_n \overset{\text{iid}}{\sim} \mathrm{N}(\mu, \sigma^2)$.

```cpp
// [[Rcpp::depends(fntl)]]
#include "fntl.h"

// [[Rcpp::export]]
Rcpp::List neldermead_mle(Rcpp::NumericVector x)
{
    // Define loglikelihood as a C++ lambda within an STL function
    std::function<double(const Rcpp::NumericVector&)> loglik =
    [&](const Rcpp::NumericVector& par) {
        double mu = par(0);
        double sigma2 = std::exp(par(1));  // Transform to nonnegative
        return Rcpp::sum(Rcpp::dnorm(x, mu, std::sqrt(sigma2), true));
    };

    fntl::neldermead_args args;
    args.fnscale = -1;                  // Flip to maximization problem

    // Set initial values and run optimization
    auto init = Rcpp::NumericVector::create(0, 0);
    auto out = fntl::neldermead(init, loglik, args);

    return Rcpp::wrap(out);     // Wrap result to Rcpp::List and return
}
```

References

John Abowd et al. The 2020 Census Disclosure Avoidance System TopDown Algorithm. *Harvard Data Science Review*, (Special Issue 2), 2022.

Winston Chang. *R6: Encapsulated Classes with Reference Semantics*, 2021. R package version 2.5.1.

Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3–4):211–407, 2014.

Dirk Eddelbuettel. *Seamless R and C++ Integration with Rcpp*. Springer, 2013.

Kyle M. Irimata et al. Evaluation of Bayesian hierarchical models of differentially private data based on an approximate data model. Research Report Series: Statistics #2022-05, U.S. Census Bureau, 2022.

Luca Martino, David Luengo, and Joaquín Míguez. *Independent Random Sampling Methods*. Springer, 2018.

Andrew M. Raim. Direct sampling in Bayesian regression models with additive disclosure avoidance noise. Research Report Series: Statistics #2021-01, U.S. Census Bureau, 2021.

Andrew M. Raim, James A. Livsey, and Kyle M. Irimata. Rejection sampling with vertical weighted strips. 2024+. https://arxiv.org/abs/2401.09696.