# Numerical integration

Julia Wrobel

# Overview

Today, we cover:

- Numerical integration
  - Newton-Cotes: Riemann, midpoint, trapezoidal, Simpson's rule
  - Quadrature
  - Laplace approximation
- Rejection sampling
- Monte Carlo integration

Resources:

- Peng Advanced Stat Computing Chapters 5-6

# Numerical integration

In statistical applications we often need to compute quantities of the form

$$E_f[g(X)] = \int g(x)f(x)dx$$

where $X$ is a random variable drawn from a distribution with PDF $f$.

- Also often need to compute normalizing constant for a PDF: if $X$ has a density that is proportional to $p(x|\theta)$, then its normalizing constant is $\int p(x|\theta)dx$.
- In such problems, an integration must be evaluated. When this can't be solved analytically, we use numerical integration

# Numerical integration

**Exact techniques**: involve identifying a sequence of estimates to the integral that eventually converge to the true values as some index (usually involving time or resources) goes to infinity.

- Adaptive quadrature
- independent Monte Carlo
- MCMC

**Approximate techniques**: involves identifying a class of alternative functions that are easier to work with, finding the member of the class that best matches the true function, then working with that alternative function instead to compute the integral

- Laplace approximation
- Variational inference
- Approximate Bayes computation

# Quadrature rules

**Quadrature rules** approximate the value of a definite integral by taking a weighted sum of function values at specific points:

$$\int_a^b f(x)dx \approx \sum_{i=1}^{n} w_i f(x)$$

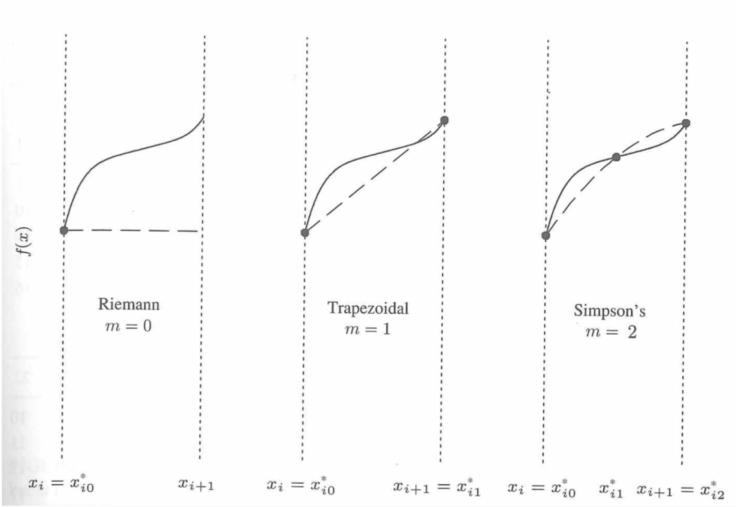
# Quadrature rules

**Quadrature rules** approximate the value of a definite integral by taking a weighted sum of function values at specific points:

$$\int_a^b f(x)dx \approx \sum_{i=1}^{n} w_i f(x)$$

- Newton-Cotes rules
  - Nodes are equally spaced
  - Trapezoidal rule, Simpson's rule
  - Simple but not great for high precision or many points
- Gaussian quadrature
  - Nodes not equally spaced
  - Use more optimal spacing and weights to improve accuracy
- **Best for lower-dimensional problems**

# Newton-Cotes quadrature



# Riemann rule

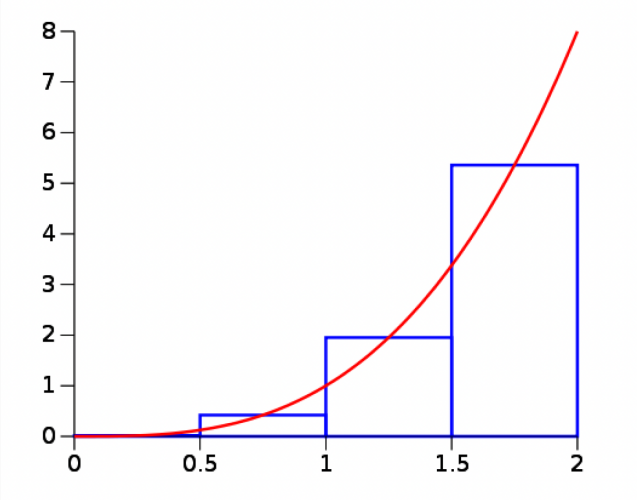
- Given function $f(x)$ on an interval $[a, b]$, divide into $n$ equal subintervals
  - Width of each subinterval is $\Delta x = \frac{b-a}{n}$

$$\int_a^b f(x)dx \approx \sum_{i=1}^{n} \Delta x \times f(x_i^*)$$

where $x_i^*$ is a chosen point in the $i$th subinterval $[x_{i-1}, x_i]$

- **Left, right**: $x_i^* = x_{i-1}$, $x_i^* = x_i$
- **Midpoint**: $x_i^* = \frac{x_{i-1}+x_i}{2}$

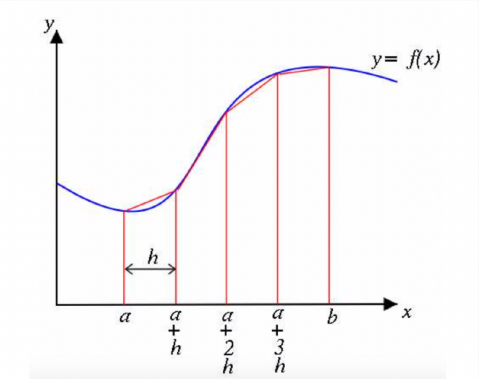
# Midpoint rule



# Trapezoidal rule

**Trapezoidal rule** approximates the function in each subinterval by a line

$$\int_a^b f(x)dx \approx \sum_{i=0}^{n-1} \frac{x_{i+1} - x_i}{2} \times [f(x_i) + f(x_{i+1})] \triangleq T_n(f)$$



# Trapezoidal rule

How good is the trapezoidal rule approximation?

- For some (unknown) $x^* \in [a, b]$, the error in approximation is

$$\left| \int_a^b f(x)dx - T_n(f) \right| = \left| \frac{(b-a)^3 f''(x^*)}{12n^2} \right|$$

- This means the trapezoidal rule provides **quadratic convergence**
  - Doubling the number of subintervals reduces the error in approximation by a factor of 4

# Simpson's rule

Simpson's rule uses a **quadratic polynomial** to approximate each subinterval

- With $n$ equally spaced intervals of length $2h = (b-a)/n$:

$$\int_a^b f(x)dx \approx \frac{h}{3}\sum_{i=0}^{n-1}[f(a+2ih) + 4f(a+(2i+1)h) + f(a+2(i+1)h)]$$

$$= \frac{h}{3}\left[f(a) + \sum_{i=1}^{n-1} 2f(a+2ih) + \sum_{i=0}^{n-1} 4f(a+(2i+1)h) + f(b)\right]$$

# Simpson's rule

- For some (unknown) $x^* \in [a, b]$, the error in approximation $S_n(f)$ is

$$\left| \int_a^b f(x)dx - S_n(f) \right| = \left| \frac{(b-a)^5 f''''(x^*)}{180n^4} \right|$$

- Doubling the number of subintervals reduces the error in approximation by a factor of 16

# Comparison of rules

- For convex functions, the midpoint rule under-estimates and the trapezoid rule over-estimates.
- For concave functions, the midpoint rule over-estimates and the trapezoid rule under-estimates.
- It can be shown that Simpson's rule is a weighted average of the midpoint (M) and trapezoid (T) rules:

$$S = \frac{2M + T}{3}$$

# Newton-Cotes in R

Example: integrate $f(x) = \frac{1}{(x+1)x^{1/2}}$

- See lab exercise to compare Trapezoidal and Simpson's rule.

# Gaussian quadrature

Gaussian quadrature allows subintervals to be more freely chosen.

- Weights and subintervals are node-specific
- Provides the **exact solution** if $f$ is a polynomial with degree at most $2n-1$

$$\int_a^b \omega(x)f(x)dx \approx \sum_{j=1}^{n} w_j f(x_j)$$

- $\omega(x)$ is some weight function
- $w_j, j = 1, \dots, n$ : are specific weights; $x_j$ are nodes
- If $f$ can be approximated well by a degree $2n-1$ polynomial on $[a, b]$, then Gaussian quadrature provides a good approximation to the integral of interest.

# Gaussian quadrature

**Question**: How to choose the weights and placement of the nodes?

- Gauss-Legendre quadrature: $\int_{-1}^{1} f(x)dx$
- Gauss-Chebyshev: $\int_{-1}^{1} \frac{1}{\sqrt{1-x^2}} f(x)dx$
- Gauss-Laguerre: $\int_{0}^{\infty} e^{-x} f(x)dx$
- Gauss-Hermite: $\int_{-\infty}^{\infty} e^{-x^2} f(x)dx$

# Gaussian quadrature

It turns out the suitable placement of the nodes corresponds to the **roots** of a polynomial belonging to a class of **orthogonal polynomials** with respect to some weight function $\omega(x)$.

- A set of orthogonal polynomials w.r.t. $\omega(x)$ defined on $[a, b]$ satisfies the relationship:

$$\int_a^b \omega(x) p_i(x) p_j(x) = \delta_{ij} c_i$$

- $\delta_{ij} = 1$ if $i = j$ and 0 otherwise
- $c_i$ is a constant
- If $c_i = 1$, then the polynomials are orthonormal

# Gauss-Legendre quadrature

If your limits of integration are $[-1, 1]$, you can use the set of **Legendre polynomials** with $\omega(x) = 1$.

# Two-point Gauss-Legendre example

# Two-point Gauss-Legendre example

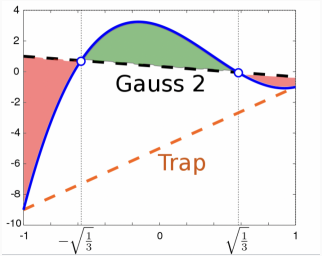
# Gauss-Legendre nodes and weights

| Number of points, $n$ | Points, $x_i$ | | Weights, $w_i$ | |
|---|---|---|---|---|
| 1 | 0 | | 2 | |
| 2 | $\pm\frac{1}{\sqrt{3}}$ | ±0.57735... | 1 | |
| 3 | 0 | | $\frac{8}{9}$ | 0.888889... |
| | $\pm\sqrt{\frac{3}{5}}$ | ±0.774597... | $\frac{5}{9}$ | 0.555556... |
| 4 | $\pm\sqrt{\frac{3}{7}-\frac{2}{7}\sqrt{\frac{6}{5}}}$ | ±0.339981... | $\frac{18+\sqrt{30}}{36}$ | 0.652145... |
| | $\pm\sqrt{\frac{3}{7}+\frac{2}{7}\sqrt{\frac{6}{5}}}$ | ±0.861136... | $\frac{18-\sqrt{30}}{36}$ | 0.347855... |
| 5 | 0 | | $\frac{128}{225}$ | 0.568889... |
| | $\pm\frac{1}{3}\sqrt{5-2\sqrt{\frac{10}{7}}}$ | ±0.538469... | $\frac{322+13\sqrt{70}}{900}$ | 0.478629... |
| | $\pm\frac{1}{3}\sqrt{5+2\sqrt{\frac{10}{7}}}$ | ±0.90618... | $\frac{322-13\sqrt{70}}{900}$ | 0.236927... |

# Gauss-Legendre example

Let $f(x) = 7x^3 - 8x^2 - 3x + 3$

- 2-point Gauss-Legendre quadrature rule provides exact answer on [-1,1] for a polynomial of degree $2 * 2 - 1 = 3$ or less
  - Use $P_2'$ for 2-point rule



# Changing limits of integration

Now that we know how to integrate over $[-1, 1]$ it would be convenient if we could transform any problem with general definite limits of integration $[a, b]$ to $[-1, 1]$

It is possible to do so!

$$\int_a^b f(x)dx = \int_{-1}^{1} f\left(\frac{b-a}{2}y + \frac{a+b}{2}\right)\frac{dx}{dy}dy$$

where $\frac{dx}{dy} = \frac{b-a}{2}$

# Changing limits of integration

Thus, using Gauss-Legendre weights $w_i$ and nodes $y_i$:

$$\int_a^b f(x)dx \approx \frac{b-a}{2} \sum_{i-1}^{n} w_i f\left(\frac{b-a}{2} y_i + \frac{a+b}{2}\right)$$

# Laplace approximation

This technique can be used for reasonably well behaved functions that have most of their mass concentrated in a small area of their domain. Technically, it works for functions that are in the class square integrable functions meaning:

$$\int f(x)^2 dx < \infty$$

Such a function generally has very rapidly decreasing tails so that in the far reaches of the domain we would not expect to see large spikes.

The general idea is to approximate a well-behaved, **unimodal function** with a **Normal density** function, which is a very well understood quantity.

# Laplace approximation

# Laplace approximation

Suppose we have a square integrable function $f(x)$ that that achieves its maximum at $x_0$.

- Let $h(x) = \log f(x)$ so that

$$\int_a^b f(x)dx = \int_a^b \exp(h(x)dx$$

- Note that $x_0 = \arg\max f(x) \implies x_0 = \arg\max h(x)$

Next, apply a Taylor series approximation of $h(x)$ around $x_0$:

$$\int_a^b \exp(h(x))\,dx \approx \int_a^b \exp\left(h(x_0) + h'(x_0)(x - x_0) + \frac{1}{2}h''(x_0)(x - x_0)^2\right)\,dx$$

# Laplace approximation

Since $h(x)$ achieves it max at $x_0$, $h'(x_0) = 0$ and our Taylor series approximation reduces to:

$$\int_a^b \exp(h(x))\,dx \approx \int_a^b \exp\left(h(x_0) + \frac{1}{2}h''(x_0)(x - x_0)^2\right)\,dx$$

Using the fact that $h(x_0)$ is a constant and rearranging terms, we simplify to:

$$\int_a^b \exp(h(x))\,dx \approx \exp(h(x_0)) \int_a^b \exp\left(-\frac{(x - x_0)^2}{2\left(-h''(x_0)\right)^{-1}}\right)\,dx$$

The integrand on the right side has a quantity that is proportional to a Normal density with mean $x_0$ and variance $-h''(x_0)^{-1}$.

# Laplace approximation

At this point we are just one call to the `pnorm()` function away from approximating our integral. All we need is to compute our normalizing constants.

- Let $\Phi(x|\mu, \sigma^2)$ denote the CDF of the Normal distribution with mean $\mu$ and variance $\sigma^2$, and $\phi(x|\mu, \sigma^2)$ its density.
- Then, the approximation can be written as:

$$= \exp(h(x_0))\sqrt{\frac{2\pi}{-h''(x_0)}} \int_a^b \varphi\left(x \mid x_0, (-h''(x_0))^{-1}\right) dx \quad (1)$$

$$= \exp(h(x_0))\sqrt{\frac{2\pi}{-h''(x_0)}} \left[\Phi\left(b \mid x_0, (-h''(x_0))^{-1}\right) - \Phi\left(a \mid x_0, (-h''(x_0))^{-1}\right)\right] \quad (2)$$

# Laplace approximation

Note that Laplace approximation replaces the problem of integrating a function with the problem of maximizing it.

- In other words, to compute the Laplace approximation we have to compute the location of the mode, which is an optimization problem.
- Often, this optimization problem is faster to solve using well understood function optimizers than integrating the function of interest using a different method

# Laplace approximation and GLMMs

Generalized linear mixed models (GLMMs) take the form:

The GLM likelihood involves an integral over random effects:

$$L(\beta) = \int \prod_{i=1}^{n} p(y_i|\eta_i, b_i)p(b|\beta)db_i$$

- Integral (usually) not tractable analytically

# Laplace approximation and GLMMs

In practice, a Laplace approximation is used to approximate this integral by:

- Finding the mode $\widehat{b}$
- Approximating the integrand around $\widehat{b}$ using a second-order Taylor expansion
- Results in a Gaussian integral that can be evaluated in closed form

This gives an approximation to the marginal likelihood, which can then be used for inference or optimization.

- `lme4::glmer()` function in `R` uses this approach

# Numerical integration and GLMMs

glmer {lme4} R Documentation

Fitting Generalized Linear Mixed-Effects Models

**Description**

penalized iteratively reweighted least squares (PIRLS) steps.

| | |
|---|---|
| `nAGQ` | integer scalar - the number of points per axis for evaluating the adaptive Gauss-Hermite approximation to the log-likelihood. Defaults to 1, corresponding to the Laplace approximation. Values greater than 1 produce greater accuracy in the evaluation of the log-likelihood at the expense of speed. A value of zero uses a faster but less exact form of parameter estimation for GLMMs by optimizing the random effects and the fixed-effects coefficients in the penalized iteratively reweighted least squares step. (See Details.) |

**Details**

Fit a generalized linear mixed model, which incorporates both fixed-effects parameters and random effects in a linear predictor, via maximum likelihood. The linear predictor is related to the conditional mean of the response through the inverse link function defined in the GLM `family`.

The expression for the likelihood of a mixed-effects model is an integral over the random effects space. For a linear mixed-effects model (LMM), as fit by `lmer`, this integral can be evaluated exactly. For a GLMM the integral must be approximated. The most reliable approximation for GLMMs is adaptive Gauss-Hermite quadrature, at present implemented only for models with a single scalar random effect. The `nAGQ` argument controls the number of nodes in the quadrature formula. A model with a single, scalar random-effects term could reasonably use up to 25 quadrature points per scalar integral.

# INLA

The Laplace approximation is a core building block of INLA (Integrated Nested Laplace Approximation)

- INLA is a deterministic alternative to MCMC for approximate Bayesian inference
- Very popular `R-INLA` package

# Posterior mean

In Bayesian computations we often want to compute the posterior mean of a parameter given the observed data.

- If $y$ represents data we observe, and $y$ comes from the density $f(y|\theta)$ with $\theta \sim \pi(\theta)$, then we usually want to compute the posterior distribution $p(\theta|y)$ and its mean:

$$E_p[\theta] = \int \theta p(\theta|y) d\theta$$

# Posterior mean

Note that

$$\int \theta \, p(\theta \mid y) \, d\theta = \int \theta \, \frac{f(y \mid \theta) \, \pi(\theta)}{\int f(y \mid \theta) \, \pi(\theta) \, d\theta} \, d\theta \tag{3}$$

$$= \int \theta \, \frac{\exp\{\log[f(y \mid \theta) \, \pi(\theta)]\}}{\int \exp\{\log[f(y \mid \theta) \, \pi(\theta)]\} \, d\theta} \, d\theta \tag{4}$$

- Let $h(\theta) = \log f(y|\theta)\pi(\theta)$
- Now we can use Laplace approximation for this interval, but need to know where $h(\theta)$ achieves its maximum

# Posterior mean

- Let $h(\theta) = \log f(y|\theta)\pi(\theta)$
- Now we can use Laplace approximation for this interval, but need to know where $h(\theta)$ achieves its maximum
  - Since $f(y|\theta)\pi(\theta)$ is proportional to the posterior density, it achieves its maximum at the **posterior mode**
  - Since $h(\cdot)$ is a monotonic transformation of $f(y|\theta)\pi(\theta)$, $h(\theta)$ also achieves its maximum at the posterior mode

# Posterior mean

- Let $\hat{\theta}$ denote the posterior mode. Then:

$$\int \theta\, p(\theta \mid y)\, d\theta \approx \frac{\int \theta \exp\left(h(\hat{\theta}) + \frac{1}{2}h''(\hat{\theta})(\theta - \hat{\theta})^2\right) d\theta}{\int \exp\left(h(\hat{\theta}) + \frac{1}{2}h''(\hat{\theta})(\theta - \hat{\theta})^2\right) d\theta} \tag{5}$$

$$= \frac{\int \theta \exp\left(\frac{1}{2}h''(\hat{\theta})(\theta - \hat{\theta})^2\right) d\theta}{\int \exp\left(\frac{1}{2}h''(\hat{\theta})(\theta - \hat{\theta})^2\right) d\theta} \tag{6}$$

$$= \frac{\sqrt{\frac{2\pi}{-h''(\hat{\theta})}} \int \theta\, \phi(\theta \mid \hat{\theta}, -h''(\hat{\theta})^{-1})\, d\theta}{\sqrt{\frac{2\pi}{-h''(\hat{\theta})}} \int \phi(\theta \mid \hat{\theta}, -h''(\hat{\theta})^{-1})\, d\theta} \tag{7}$$

$$= \hat{\theta} \tag{8}$$

# Posterior mean

We've just shown that the Laplace approximation to the posterior mean is equal to the posterior mode.

- This approximation is likely to work well when the posterior is unimodal and relatively symmetric around the mode.

# Posterior mean

**Example**: Poisson data with a Gamma prior. The model is

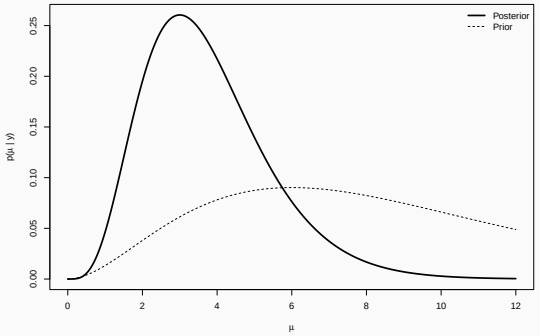
$$Y|\mu \sim Poisson(\mu) \tag{9}$$

$$\mu \sim Gamma(a, b) \tag{10}$$

- Let $a = 3$ and $b = 3$
- Given an observation $y$, the posterior distribution is a Gamma distribution with shape $y + a$ and scale $1 + 1/b$.

# Posterior mean

Suppose we observe $y = 2$. Then the posterior looks like:




Because the posterior is skewed, the mean and the mode should not match.

# Posterior mean

Since this is a Gamma distribution, we can compute the posterior mode and mean in closed form.

```
a = 3; b = 3
pmode = (y + a - 1)*(1/(1 + 1/b))
pmode
```

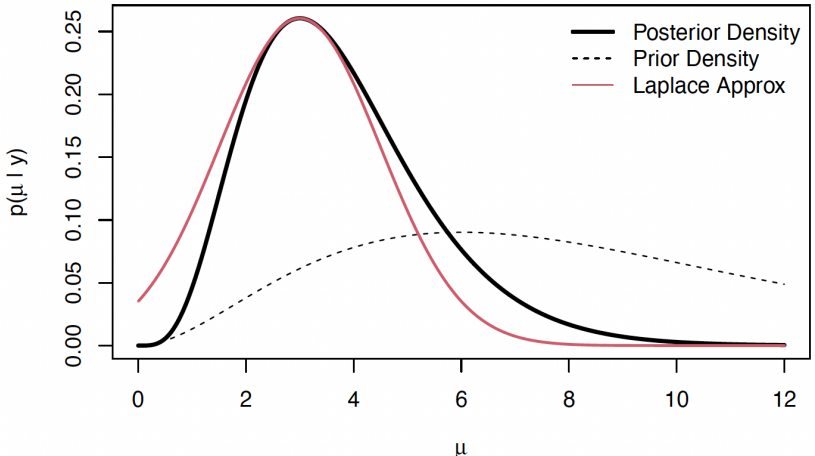
```
[1] 3
```

```
pmean = (y + a)*(1/(1 + 1/b))
pmean
```

```
[1] 3.75
```

# Posterior mean

Let's plot the Laplace approximation of the posterior.



# Posterior mean

In the neighborhood of the mode, the Laplace approximation is reasonable.

- As we move farther away from the mode, the tail of the Gamma is heavier on the right.
- We expect the approximation to improve as the sample size increases.

# Monte Carlo integration

Idea: deterministic quadrature methods work well for one-dimensional integration

- Quadrature methods suffer from curse of dimensionality, where convergence rate becomes much worse with increased dimensions.
- Integration approaches based on Monte Carlo techniques (simulation) extend naturally to multiple dimensions
- Convergence rate is independent of the number of dimensions of the integral.

# Independent Monte Carlo

Suppose we want to compute for some function $h : \mathcal{R}^k \to \mathcal{R}$,

$$E_f[h(X)] = \int h(x)f(x)dx.$$

If we could simulate $x_1, \dots, x_n \sim^{i.i.d.} f$, then by the law of large numbers we would have

$$\frac{1}{n}\sum_{i=1}^{n} h(x_i) \to E_f[h(X)].$$

Furthermore, we have that

$$Var\left(\frac{1}{n}\sum_{i=1}^{n} h(x_i)\right) = \frac{1}{n^2}\sum_{i}^{n} Var(h(x_i)) = \frac{1}{n}Var(h(X_1)),$$

which does not depend on the dimension of the random variable $X_1$.

# Monte Carlo integration

This approach to computing the expectation above is known as Monte Carlo integration.

- Takes advantage of the law of large numbers, saying that averages of numbers converge to their expectation.
- Monte Carlo integration can be quite useful, but it takes the problem of computing the integral directly (or via approximation) and replaces it with a problem of drawing samples from an arbitrary density function $f$.
- In order to be able to do this, we need to be able to generate random numbers from an arbitrary distribution $f$.

# Random number generation

Most of the time in the software we are using, there is a function that will do this for us. For example, `runif()`, `rnorm()`, `rbinom()`…

Nevertheless, there are two issues that are worth considering here:

1. How exactly is the sequence of numbers created? Can be good to understand what's going on under the hood.
2. Built-in functions in `R` are only useful for well-known or well-characterized distributions. However, for many simulation-based techniques, we will want to generate random numbers from distributions that we have likely never seen before and for which there will not be any built-in function.

We will assume we can randomly generate from a uniform distribution, and talk about techniques to generate non-uniform numbers.

# Inverse CDF transformation

The most generic method uses the inverse of the cumulative distribution function of the distribution. Suppose we want to draw samples from a distribution with density $f$ and CDF $F(x) = \int_{-\infty}^{x} f(t)dt$. Then we can do the following:

1. Draw $U \sim Unif(0, 1)$ using `runif()`
2. Let $X = F^{-1}(U)$. Then $X$ is distributed according to $f$.

**Idea**: $P(F^{-1}(U) \leq x) = P(U \leq F(x)) = F(x)$ - Requires inversion of $F$, which is not always possible

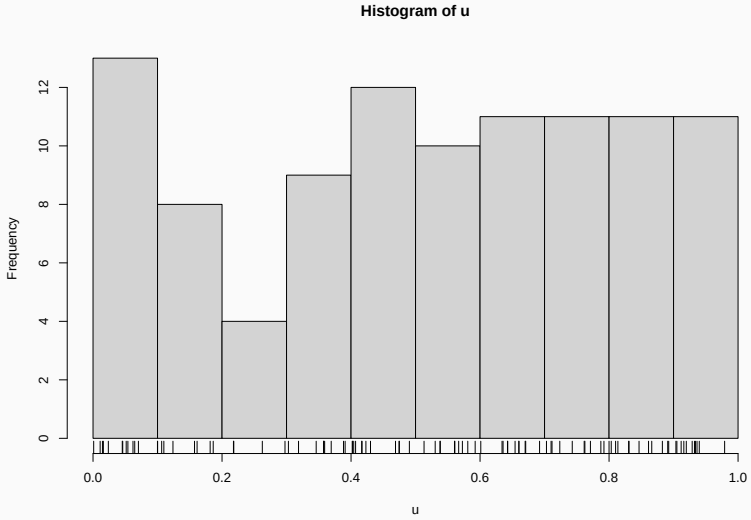
# Exponential distribution example

# Exponential distribution example

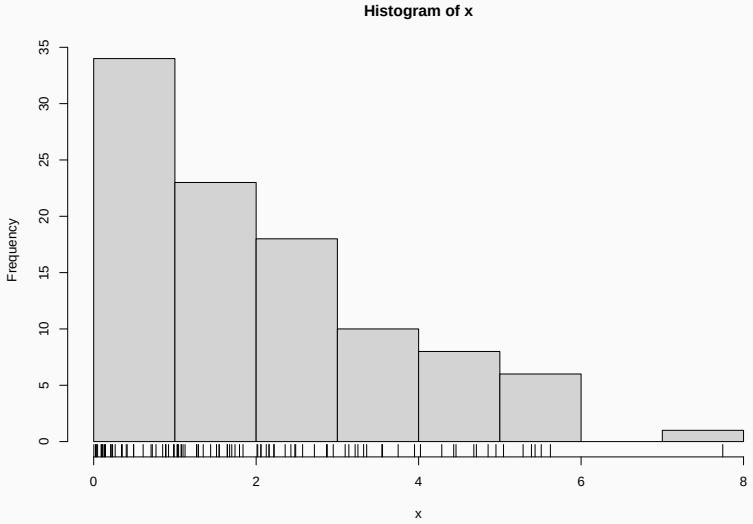
```r
set.seed(235234)
u <- runif(100)
hist(u)
rug(u)

## apply inverse CDF
lambda <- 2 ## Exponential with mean 2
x <- -lambda * log(1 - u)
hist(x)
rug(x)
```

# Exponential distribution example



# Exponential distribution example



# Rejection sampling

**Idea**: sample from a **target distribution** $f(x)$ (up to a constant) using an easy-to-sample **proposal distribution** $g(x)$.

- Though we cannot easily sample from $f$, there exists another density $g$, like a Normal distribution, from which it is easy for us to sample.
- Then we can sample from $g$ directly then "reject" the samples in a strategic way to make the resulting "non-rejected" samples look like they came from $f$.

# Rejection sampling

- First ensure the support of $f$ is a subset of the support of $g$
  - Intuition: if there is a region of the support of $f$ that $g$ can never touch, then that area will never get sampled.
- Also need to assume that for some constant $M$

$$M = \sup_{x \in \mathcal{X}_f} \frac{f(x)}{g(x)} < \infty$$

Easiest way to satisfy this second assumption is to make sure that $g$ has heavier tails than $f$.

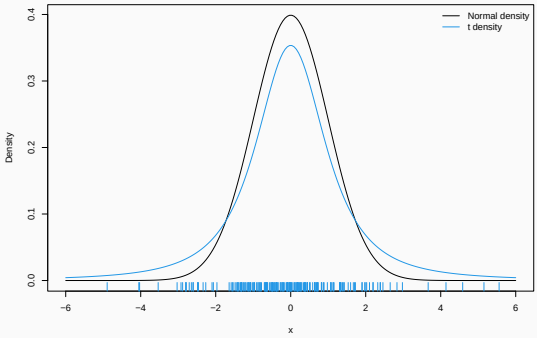
# Rejection sampling algorithm

1. Simulate $U \sim Unif(0, 1)$
2. Simulate a candidate $X \sim g$ from the proposal distribution
3. If $U \leq \frac{f(X)}{Mg(x)}$ then "accept" the candidate $X$. Otherwise, "reject" X and go back to (1).

Repeat algorithm until the desired number of samples from the target density $f$ has been accepted.

# Rejection sampling example

As a simple example, suppose we wanted to generate samples from a $N(0, 1)$. We can use the $t_2$ distribution as our proposal as it has heavier tails:



# Properties of rejection sampling

The number of draws we need to take from the $g$ before we accept a candidate is a geometric random variable with success probability $1/M$. We can think of the decision to accept or reject a candidate as a sequence of iid coin flips that has a specific probability of coming up "heads" (i.e. being accepted).

$$\mathbb{P}(X \text{ accepted}) = \mathbb{P}\left(U \leq \frac{f(X)}{M\,g(X)}\right) \tag{11}$$

$$= \int \mathbb{P}\left(U \leq \frac{f(x)}{M\,g(x)} \,\middle|\, X = x\right) g(x)\,dx \tag{12}$$

$$= \int \frac{f(x)}{M\,g(x)} g(x)\,dx \tag{13}$$

$$= \frac{1}{M} \tag{14}$$

# Back to Monte Carlo integration

Consider evaluating the integral $\theta = \int_a^b f(x)dx$.

- Multiplying and dividing by $(b-a)$, we can write $\theta = (b-a)E(f(X))$ where $X \sim Unif(a,b)$, since

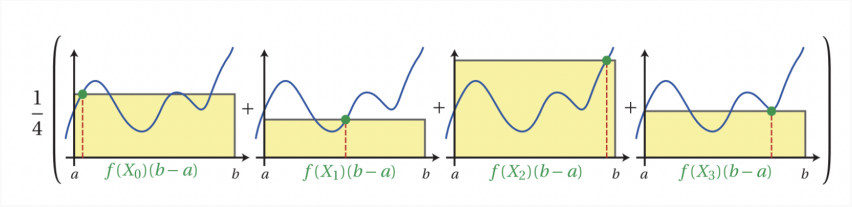
$$\theta = (b-a)\int_a^b f(x)\frac{1}{b-a}dx$$

Thus, we've transformed the integration problem into the problem of estimating the expected value of a function of a random variable.

# Monte Carlo integration

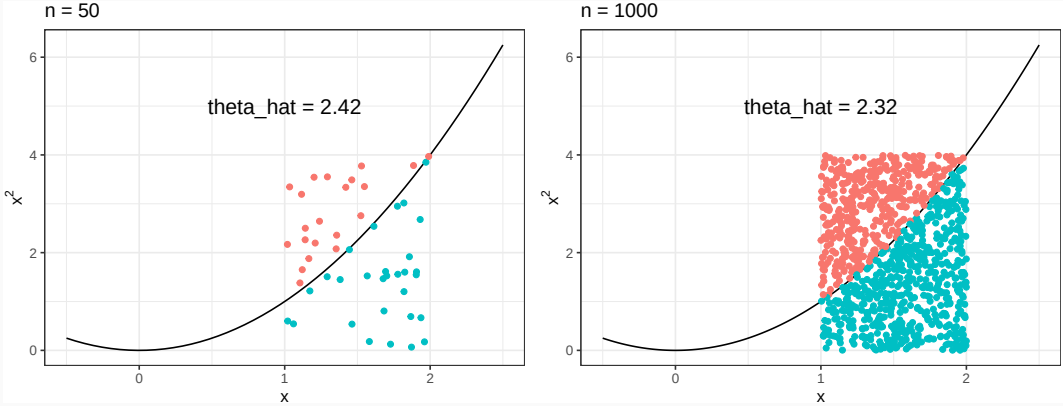
The Monte Carlo method works by generating an iid sample $X_i \sim Unif(a, b), i = 1, \ldots, n$, and estimating $\theta$ using:

$$\hat{\theta} = (b - a)\frac{\sum_{i=1}^{n} f(X_i)}{n}$$



# Visual intuition

We're going to find $\theta = \int_1^2 x^2 dx = 7/3$



# Visual intuition

- Points are uniformly sampled from a rectangle enclosing the curve.
- The fraction of points under the curve gives an estimate of the integral
- More points = better approximation

# Monte Carlo integration

Monte Carlo integration can also be used more generally for $d-$dimensional integrals of the form

$$\int_D f(\mathbf{x}) dP\mathbf{x},$$

where $dP\mathbf{x}$ is the probability density/mass function of a random vector $\mathbf{X}$, and it is possible to sample from $dP\mathbf{x}$.

# Monte Carlo integration

The primary justification for Monte Carlo integration is through the **laws of large numbers**.

- Let $\mu \triangleq E(Y)$ for some random variable $Y \sim D$.
- Let $\widehat{\mu} \triangleq \frac{1}{n} \sum_{i=1}^{n} Y_i$

# Monte Carlo integration

The primary justification for Monte Carlo integration is through the **laws of large numbers**.

- Let $\mu \triangleq E(Y)$ for some random variable $Y \sim D$.
- Let $\hat{\mu} \triangleq \frac{1}{n} \sum_{i=1}^{n} Y_i$
- As long as $\mu$ exists and we use iid sampling to obtain $Y_1, \ldots, Y_n$:

$$\lim_{n \to \infty} Pr(|\hat{\mu}_n - \mu| \leq \epsilon) = 1$$
$$Pr(\lim_{n \to \infty} |\hat{\mu}_n - \mu| = 0) = 1$$

- The **weak law** tells us that our chance of missing the true answer by more than $\epsilon$ goes to zero as $n \to \infty$.
- The **strong law** tells us that the absolute error will eventually get below $\epsilon$ and then stay there.

# Monte Carlo integration

Both laws of large numbers tell us we can eventually attain an error as small as we like.

- How large does $n$ need to be for this to happen?
- For a given $n$, what is the error in our approximation?

Suppose that $Var(y) = \sigma^2 < \infty$

- With iid sampling, $\widehat{\mu_n}$ is a **random variable** with its own mean and variance. It's mean is:

$$E[\widehat{\mu}_n] = \frac{1}{n}\sum_{i}^{n} E[Y_i] = E[Y] = \mu$$

$\implies$ Monte Carlo integration is **unbiased**.

# Monte Carlo integration

The variance of $\hat{\mu}_n$ is

$$E[(\hat{\mu}_n - \mu)^2] = \frac{\sigma^2}{n}$$

$\implies$ SE is $\sigma/\sqrt{n}$

- Thus, our Monte Carlo estimate has root $n$ convergence
- For a given $\sigma$, if we want **one more decimal digit of accuracy** (the SE one tenth as large) we need a *100-fold* increase in computation
- The error can also be improved by decreasing $\sigma$

# Quantifying uncertainty

Let's return to our problem of estimating $\theta = \int_a^b f(x)dx$ using Monte Carlo integration.

$$\hat{\theta} = (b-a)\frac{\sum_{i=1}^{n} f(X_i)}{n}$$

The variance of $\hat{\theta}$ is

$$V(\hat{\theta}) = \frac{(b-a)^2}{n^2}\sum_{i=1}^{n} V(f(Y_i)) \tag{15}$$

$$= \frac{(b-a)^2}{n} V(f(Y)) \tag{16}$$

# Quantifying uncertainty

Then, an estimator for $V(f(Y))$ is

$$\frac{1}{n-1}\sum_{i}^{n}\left(f(Y_i)-\frac{1}{n}\sum_{i}^{n}f(Y_i)\right)^2$$

And an estimator for $V(\hat{\theta})$ is

$$\widehat{V}(\hat{\theta}) = \frac{(b-a)^2}{n}\cdot\frac{1}{n-1}\sum_{i=1}^{n}\left(f(Y_i)-\frac{1}{n}\sum_{i=1}^{n}f(Y_i)\right)^2 \tag{17}$$

$$= \frac{1}{n-1}\sum_{i=1}^{n}\left((b-a)f(Y_i)-\hat{\theta}\right)^2\cdot\frac{1}{n} \tag{18}$$

# MC integration example

Want to estimate $\theta = \int_0^5 (x^3 - 5x^2 + 3x + 15)dx$ - Closed form solution is 60.4166667

```
set.seed(2135)
n = 100000; a=0; b=5
r = runif(n, a, b)
f = function(x){x^3-5*x^2 + 3*x + 15}
theta_hat = (b-a)*sum(f(r))/n
theta_hat
```

```
[1] 60.3697
```

```
var_theta_hat = (b-a)^2/n * 1/(n-1) * sum((f(r)-mean(f(r)))^2)

var_theta_hat
```

```
[1] 0.006899472
```

# Multivariate integrals

- For most one-dimensional integrals, Monte Carlo integration is not needed but can still be applied.
- Monte Carlo is often the best option for multivariate integrals.
- However, the sample size required for accuracy grows exponentially with the dimension of the integral.

# Other topics

- Importance sampling
  - An efficient adaptation of rejection sampling for computing expectations
  - Good discussion in Peng Chapter 6
- How random number generators actually work
- We've really only scratched the surface of numerical integration using simple examples
- MC techniques can be combined with quadrature methods in higher-dimensional problems, using one for some variables and the other for the rest

# Resources

- More on quadrature rules
- Nice explanation of Gaussian quadrature
- Monte Carlo integration

# Gauss-Legendre quadrature

The weights for Gauss-Legendre quadrature are given by:

$$w_i = \frac{2}{(1-x_i^2)[P_n'(x_i)]^2}$$

Where the first few Legendre polynomials are:

$$P_0(x) = 1$$

$$P_1(x) = x$$

$$P_2(x) = \frac{1}{2}\left(3x^2 - 1\right)$$

$$P_3(x) = \frac{1}{2}\left(5x^3 - 3x\right)$$

$$P_4(x) = \frac{1}{8}\left(35x^4 - 30x^2 + 3\right)$$