Tutorial: Rejection Sampling

Matteo Fasiolo

Sampling

Sampling is one of the most common tasks in statistical data analysis and in particular Monte Carlo simulations.

The task of sampling is to draw samples of a random variable (r.v.), given its probability density/mass function.

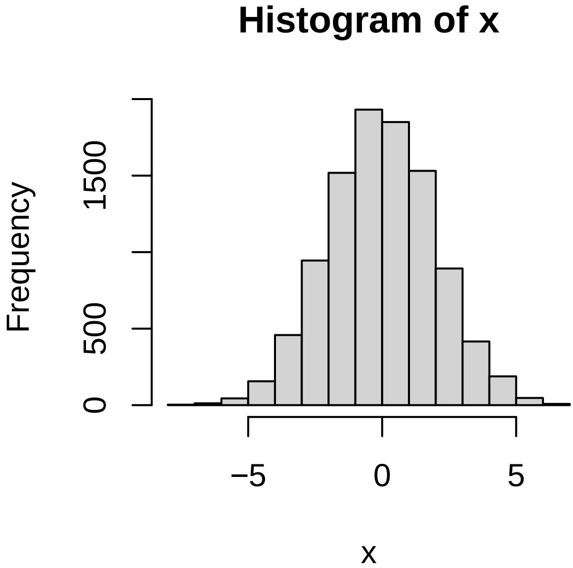
R provides lots of built-in samplers following the pattern:

```
rdistr(n, par1, par2, par3, ...)
```

which outputs a vector of n random variable from a distribution with parameters `par1`, `par2`, `par3`, ...

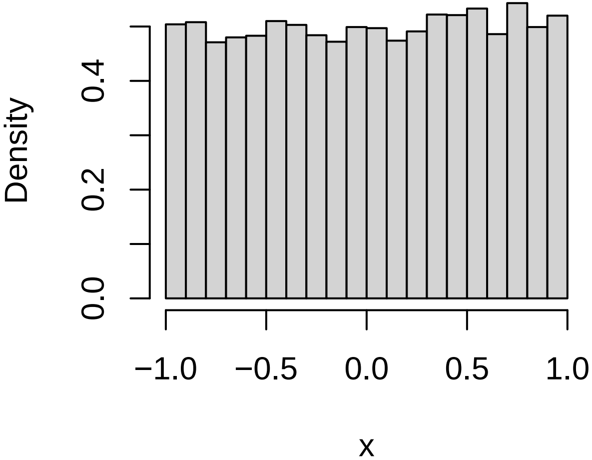
Example: sample from $N(\mu, \sigma^2)$ with $\mu = 0$ and $\sigma = 2$:

```r
x <- rnorm(n = 1e4, mean = 0, sd = 2)
hist(x)
```



Sample from $Unif(a, b)$ with $a = -1$ and $b = 1$:

```
x <- runif(n = 1e4, min = -1, max = 1)
hist(x, prob = TRUE, main = "")
```

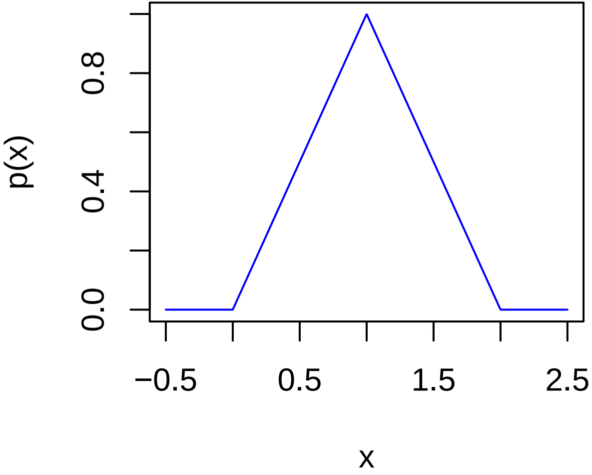


We use `prob = TRUE` to have probability densities on y-axis.

Sampling for non-standard distributions

But, how do you sample from (e.g.) a triangular distribution?

Suppose we know its p.d.f. $p(x) = 1 - |x - 1|, x \in [0\ 2]$?



Solution: Rejection Sampling

Basic idea:

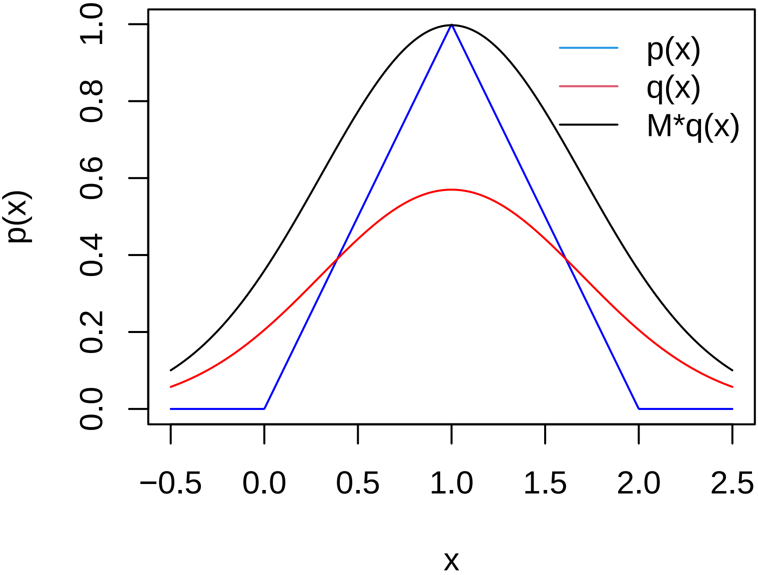
1. sample from a distribution you know how to sample from (the **proposal distribution**);
2. "adjust" that sample so it becomes a sample from the distribution of interest (the **target distribution**).

Let $p(x)$ be the p.d.f. of target and $q(x)$ be the p.d.f. of proposal.

Let $M^* > 1$ be smallest constant such that:

$$p(x) \leq M^* q(x) \text{ for any } x \text{ s.t. } p(x) > 0.$$

Sounds confusing? We just need to $Mq(x)$ to "cover" $p(x)$...



NB: you can choose $M > M^*$, but $p(x) \leq Mq(x)$ is necessary.

Algorithmic Implementation

Objective: get a sample of n_{max} independent random variables (r.v.s) from target.

Algorithm:

1. Draw a r.v. x from $q(x)$ and a uniform sample $u \sim U(0,1)$
2. If $u < \frac{p(x)}{M^* q(x)}$, accept sample x and store it.
3. If number of accepted samples so far equals to n_{max}, quit. Otherwise, go back to step 1.

Here we will use $Unif(a=0, b=2)$ as proposal distribution $q(x)$.

When accepting a sample in step 2, we need to add it to the vector of accepted samples so far.

Two ways of doing it:

```r
a <- c(1,2,3,4)


a[5] <- 5
print(a)
```

```
[1] 1 2 3 4 5
```

```r
# OR
a <- c(a, 6)
print(a)
```

```
[1] 1 2 3 4 5 6
```

Code Skeleton

```
# how many samples do we want?
n_max = 1000

p_target <- function(x){
  # TODO: the PDF of distribution, from which
  # you want to sample.
}

acc <- c() # create an empty vector
n <- 0 # how many samples have we already obtained?
M <- NA # TODO you should be able to work it out!
while(n < n_max){
 # TODO
}
hist(acc) #plot the histogram of accepted samples
```

Questions:

1. Complete the above code skeleton according to the rejection sampling algorithm.
2. What output is expected when the algorithm is implemented correctly?
3. What happens if you increase M above M^*? Is the algorithm still correct (use visualisation to assess this)? Does the code become faster or slower?
4. What happens if you decrease M below minimum required value?

Challenge: try using $N(\mu = 1, \sigma^2 = 0.5^2)$ as proposal:

- Computing optimal M^* might be hard, but you just need M big enough.
- To compute it you need to know the Gaussian p.d.f. (see `?dnorm`).
- Is this more efficient than using a uniform proposal?