Sampling from a Normal distribution

We will sample from the standard normal distribution by hand, N(0, 1). After that we can sample from any normal distribution, N(mu, sigma). Finally, we will sample from a multi-variate normal ditribution.

Once we build a sampling method for N(0, 1) we can convert samles to N(mu, sigma) by scaling as follows:

`x = x_starndard * sigma + mu`

Assume we have a access to a uniform random numbers generatorl between 0 and 1. We need to translate the uniform numbers to N(0, 1).

Generate Normal Smaple with Central Limit Theorem

The mean of a sample from any distribution approaches Normal as the size of the sample increases.

As little as 10 elements in a sample is sufficient to approximate the normal distrubution quite closely.

```
library(rethinking)
```

```
## Loading required package: rstan
```

```
## Warning: package 'rstan' was built under R version 3.3.2
```

```
## Loading required package: ggplot2
```

```
## Warning: package 'ggplot2' was built under R version 3.3.2
```

```
## Loading required package: StanHeaders
```

```
## Warning: package 'StanHeaders' was built under R version 3.3.2
```

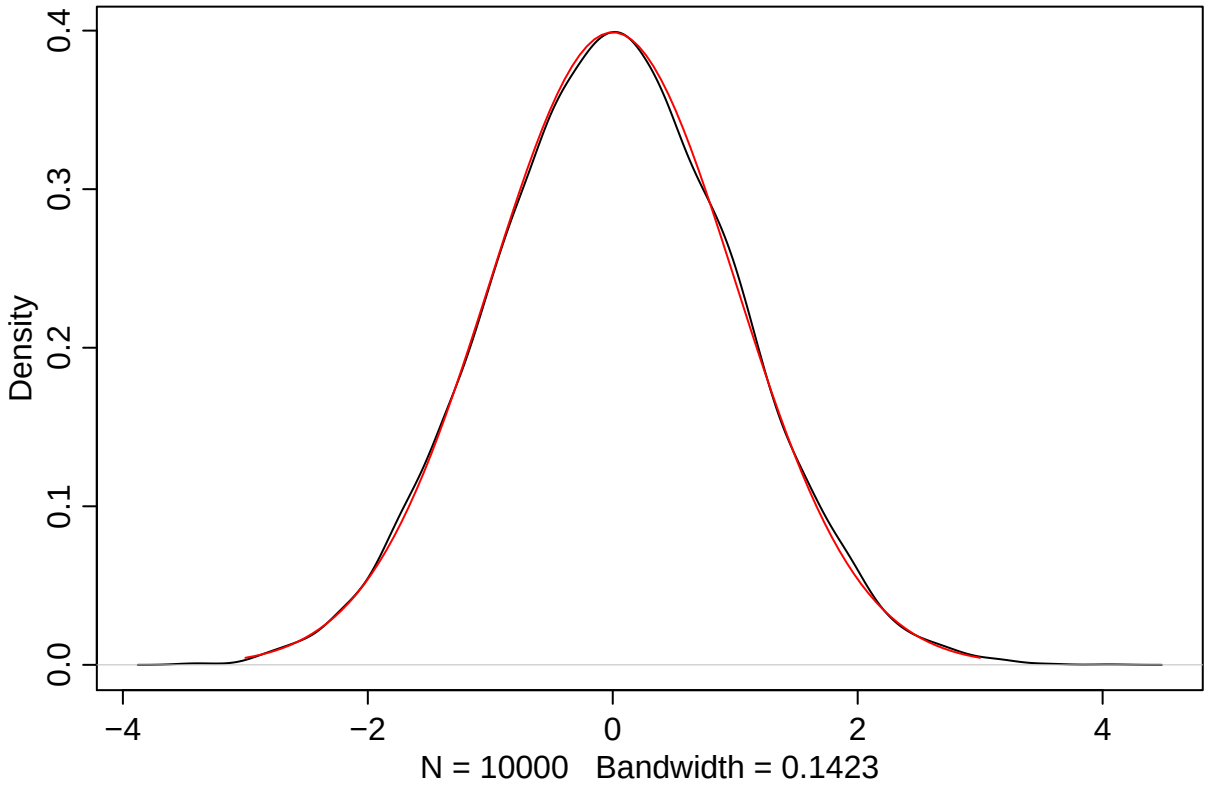
```
## rstan (Version 2.17.2, GitRev: 2e1f913d3ca3)
```

```
## For execution on a local, multicore CPU with excess RAM we recommend calling
## options(mc.cores = parallel::detectCores()).
## To avoid recompilation of unchanged Stan programs, we recommend calling
## rstan_options(auto_write = TRUE)
```

```
## Loading required package: parallel
```

```
## rethinking (Version 1.59)
```

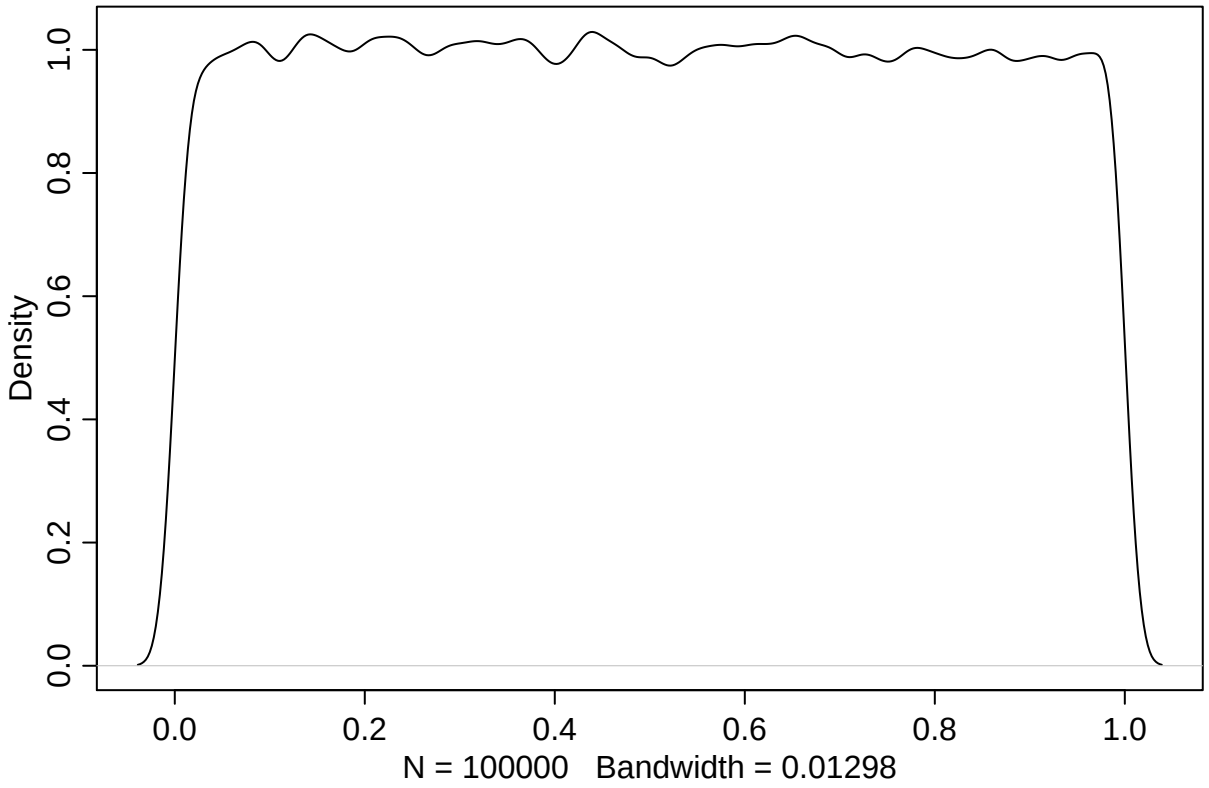
```
n = 10000 # The number of normal samples
sample_n = 10 # We will draw samples form the uniform distribution
samples = runif(n = sample_n * n, min = 0, max = 1)
samples_matrix <- matrix(samples, nrow=n, byrow = T)
x <- rowMeans(samples_matrix)
# dens(samples)
mu = 0.5
sigma = sd(samples) / sqrt(sample_n)
normal_x = (x - mu) / sigma
dens(normal_x, adj = 1)

true_normal.x <- seq(-3, 3, length.out = 100)
true_normal.y <- dnorm(true_normal.x, mean=0, sd=1)
lines(true_normal.x, true_normal.y, col="red")
```



Inverse Transform Sampling

Inverst transform sampling relies on the fact, that if X a random variable, F is its cumulative distrubtion function (i.e. $F(x)$ is the probability of the value $\leq$ x) then $F(x)$ is distributed *uniformly*. Let's test this statement.

```
inverse_transform <- function () {
  samples <- rnorm(n = 100000, mean=0, sd=1)
  cdf <- pnorm(samples)
  dens(cdf)
}
inverse_transform()
```



To sample from any distribution with a known cumulative distribution function F, we need to calculate it's inverse, generate a *uniform* sample u, then we can get the sample from our target distribution as follows:

$$x = F^{-1}(u)$$

This method is not very efficient for continuous cases where the CDF doens't have an analytic integral. Normal distribution is one example. Because of this, other methods are more popular.

Box-Mullter Transform