# Statistics 1 Unit 2: Random Number Generation


Kurt Hornik

# Outline

- Motivation
- Basics
- Inverse transform method
- Acceptance-rejection method

# Motivation

QFin is very much about financial decision making under uncertainty, with uncertainty modeled probabilistically.

- What are “random numbers”?

# Motivation

QFin is very much about financial decision making under uncertainty, with uncertainty modeled probabilistically.

- What are “random numbers”?
- What are good sources of random numbers?

# Motivation

QFin is very much about financial decision making under uncertainty, with uncertainty modeled probabilistically.

- What are “random numbers”?
- What are good sources of random numbers?
- In particular, where do computers get random numbers from?

# Outline

- Motivation
- Basics
- Inverse transform method
- Acceptance-rejection method

# Uniform random numbers

Interestingly, the “random numbers” obtained from computers are typically generated by deterministic recursions.

# Uniform random numbers

Interestingly, the “random numbers” obtained from computers are typically generated by deterministic recursions.

What is somewhat “random” then is the seed using for starting the recursion: but that can be set for reproducibility.

# Uniform random numbers

Interestingly, the “random numbers” obtained from computers are typically generated by deterministic recursions.

What is somewhat “random” then is the seed using for starting the recursion: but that can be set for reproducibility.

Technically, one uses the term *pseudorandom numbers*.

# Uniform random numbers

Interestingly, the “random numbers” obtained from computers are typically generated by deterministic recursions.

What is somewhat “random” then is the seed using for starting the recursion: but that can be set for reproducibility.

Technically, one uses the term *pseudorandom numbers*.

These are really perfectly deterministic, but work well enough for “real” random numbers.

# Uniform random numbers

Uniform (on the unit interval) pseudorandom numbers can be simulated using multiplicative congruential random number generators which use recursions

$$x_n = bx_{n-1} \pmod{m}, \qquad u_n = x_n/m$$

for suitable initial seed $x_0$.

Clearly,

- $x = 0$ “absorbs”

# Uniform random numbers

Uniform (on the unit interval) pseudorandom numbers can be simulated using multiplicative congruential random number generators which use recursions

$$x_n = bx_{n-1} \pmod{m}, \qquad u_n = x_n/m$$

for suitable initial seed $x_0$.

Clearly,

- $x = 0$ “absorbs”
- The generated sequence will be *periodic*

# Uniform random numbers

Uniform (on the unit interval) pseudorandom numbers can be simulated using multiplicative congruential random number generators which use recursions

$$x_n = bx_{n-1} \pmod{m}, \qquad u_n = x_n/m$$

for suitable initial seed $x_0$.

Clearly,

- $x = 0$ “absorbs”
- The generated sequence will be *periodic*
- The best we can get is period of $m-1$ (giving all remainders from 1 to $m-1$)

# Uniform random numbers

To illustrate:

```
R> myrng <- function(n, m, b, x) {
+     u <- numeric(n)
+     for(i in 1 : n) {
+         x <- (b * x) %% m
+         u[i] <- x / m
+     }
+     u
+ }
```

# Uniform random numbers

BnM Example 5.1:

```
R> myrng(7, 7, 3, 2)

[1] 0.8571429 0.5714286 0.7142857 0.1428571 0.4285714 0.2857143
[7] 0.8571429
```

# Uniform random numbers

BnM Example 5.1:

```
R> myrng(7, 7, 3, 2)

[1] 0.8571429 0.5714286 0.7142857 0.1428571 0.4285714 0.2857143
[7] 0.8571429
```

BnM example for bad:

```
R> myrng(5, 29241, 171, 3)

[1] 0.01754386 0.00000000 0.00000000 0.00000000 0.00000000
```

# Uniform random numbers

BnM Example 5.2:

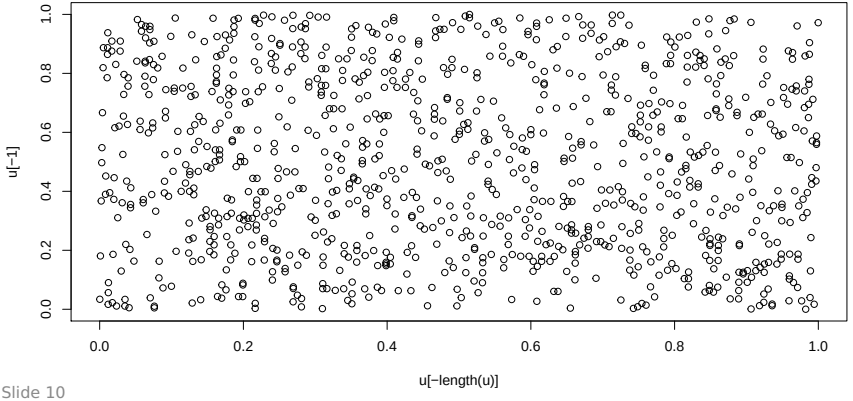
```
R> myrng(50, 30269, 171, 27218)

 [1] 0.76385080 0.61848756 0.76137302 0.19478675 0.30853348 0.75922561
 [7] 0.82757937 0.51607255 0.24840596 0.47741914 0.63867323 0.21312234
[13] 0.44391952 0.91023820 0.65073177 0.27513297 0.04773861 0.16330239
[19] 0.92470845 0.12514454 0.39971588 0.35141564 0.09207440 0.74472232
[25] 0.34751726 0.42545178 0.75225478 0.63556774 0.68208398 0.63636063
[31] 0.81766824 0.82126929 0.43704780 0.73517460 0.71485678 0.24051009
[37] 0.12722587 0.75562457 0.21180085 0.21794575 0.26872378 0.95176583
[43] 0.75195745 0.58472364 0.98774324 0.90409330 0.59995375 0.59209092
[49] 0.24754700 0.33053619
```

# Uniform random numbers

Illustrating the “randomness”:

```
R> u <- myrng(1000, 30269, 171, 27218)
R> plot(u[-length(u)], u[-1])
```



# Uniform random numbers

R has several built-in (pseudo) random number generators. E.g., method “Knuth-TAOCP-2002” uses the recursion

$$x_n = (x_{n-100} - x_{n-37}) \pmod{2^{30}}$$

which has a period around $2^{129}$.

# Uniform random numbers

R has several built-in (pseudo) random number generators. E.g., method “Knuth-TAOCP-2002” uses the recursion

$$x_n = (x_{n-100} - x_{n-37}) \pmod{2^{30}}$$

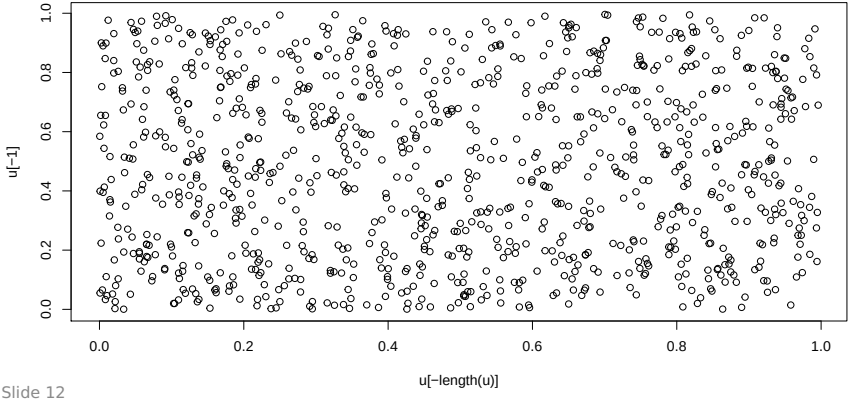
which has a period around $2^{129}$.

To get a sample of uniform (peudo) random numbers: `runif()`.

# Uniform random numbers

Illustrating the “randomness” again:

```
R> u <- runif(1000)
R> plot(u[-length(u)], u[-1])
```



# Uniform distribution

The (continuous) uniform distribution on the interval from $a$ to $b$ has density

$$f_{\text{uniform}(a,b)}(x) = \begin{cases} \frac{1}{b-a}, & a < x < b, \\ 0, & \text{otherwise.} \end{cases}$$

Often denoted by $U(a, b)$ or $U_{a,b}$ or $\mathcal{U}_{a,b}$.

# Uniform distribution

The (continuous) uniform distribution on the interval from $a$ to $b$ has density

$$f_{\text{uniform}(a,b)}(x) = \begin{cases} \frac{1}{b-a}, & a < x < b, \\ 0, & \text{otherwise.} \end{cases}$$

Often denoted by $U(a, b)$ or $U_{a,b}$ or $\mathcal{U}_{a,b}$.

It has two parameters:

- $a$ is the minimum (inf) of its support,
- $b$ is the maximum (sup) of its support.

If $a = 0$ and $b = 1$, we have the standard uniform distribution.

# Uniform distribution

In R, the parameters are called `min` and `max`.

There are four (“dpqr”) functions for the uniform distribution:

- `dunif()` gives the **d**ensity function

# Uniform distribution

In R, the parameters are called `min` and `max`.

There are four (“dpqr”) functions for the uniform distribution:

- `dunif()` gives the **d**ensity function
- `punif()` gives the **p**robability function (cumulative distribution function)

# Uniform distribution

In R, the parameters are called `min` and `max`.

There are four (“dpqr”) functions for the uniform distribution:

- `dunif()` gives the **d**ensity function
- `punif()` gives the **p**robability function (cumulative distribution function)
- `qunif()` gives the **q**uantile function (the “inverse” of the CDF, more later)

# Uniform distribution

In R, the parameters are called `min` and `max`.

There are four (“dpqr”) functions for the uniform distribution:

- `dunif()` gives the **d**ensity function
- `punif()` gives the **p**robability function (cumulative distribution function)
- `qunif()` gives the **q**uantile function (the “inverse” of the CDF, more later)
- `runif()` generates **r**andom deviates

# Uniform distribution

In R, the parameters are called `min` and `max`.

There are four (“dpqr”) functions for the uniform distribution:

- `dunif()` gives the **d**ensity function
- `punif()` gives the **p**robability function (cumulative distribution function)
- `qunif()` gives the **q**uantile function (the “inverse” of the CDF, more later)
- `runif()` generates **r**andom deviates

For many common probability functions, R provides the above 4 functions, using the d-p-q-r naming scheme.

# Uniform distribution

The 4 functions for the uniform distribution have arguments

```
dunif(x, min = 0, max = 1, log = FALSE)
punif(q, min = 0, max = 1, lower.tail = TRUE, log.p = FALSE)
qunif(p, min = 0, max = 1, lower.tail = TRUE, log.p = FALSE)
runif(n, min = 0, max = 1)
```

These have “surprising” additional arguments:

- `log = TRUE` says give log-densities (can be better numerically)

# Uniform distribution

The 4 functions for the uniform distribution have arguments

```
dunif(x, min = 0, max = 1, log = FALSE)
punif(q, min = 0, max = 1, lower.tail = TRUE, log.p = FALSE)
qunif(p, min = 0, max = 1, lower.tail = TRUE, log.p = FALSE)
runif(n, min = 0, max = 1)
```

These have “surprising” additional arguments:

- `log = TRUE` says give log-densities (can be better numerically)
- `lower.tail = TRUE` takes probabilities as $p = \mathbb{P}(X \leq x)$; otherwise, $\mathbb{P}(X > x)$ (i.e., the complementary probabilities $1 - p$).

# Uniform distribution

The 4 functions for the uniform distribution have arguments

```
dunif(x, min = 0, max = 1, log = FALSE)
punif(q, min = 0, max = 1, lower.tail = TRUE, log.p = FALSE)
qunif(p, min = 0, max = 1, lower.tail = TRUE, log.p = FALSE)
runif(n, min = 0, max = 1)
```

These have “surprising” additional arguments:

- `log = TRUE` says give log-densities (can be better numerically)
- `lower.tail = TRUE` takes probabilities as $p = \mathbb{P}(X \leq x)$; otherwise, $\mathbb{P}(X > x)$ (i.e., the complementary probabilities $1 - p$).
- `log.p = TRUE` says use/give log-probabilities (can be better numerically)

# Normal distribution

The normal distribution has density

$$\frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

where $\mu$ is the mean and $\sigma^2$ the variance, and hence $\sigma$ the sd (standard deviation).

For $\mu = 0$ and $\sigma = 1$ we get the standard normal distribution.

# Normal distribution

In R,

```
dnorm(x, mean = 0, sd = 1, log = FALSE)
pnorm(q, mean = 0, sd = 1, lower.tail = TRUE, log.p = FALSE)
qnorm(p, mean = 0, sd = 1, lower.tail = TRUE, log.p = FALSE)
rnorm(n, mean = 0, sd = 1)
```

# Normal distribution

In R,

```
dnorm(x, mean = 0, sd = 1, log = FALSE)
pnorm(q, mean = 0, sd = 1, lower.tail = TRUE, log.p = FALSE)
qnorm(p, mean = 0, sd = 1, lower.tail = TRUE, log.p = FALSE)
rnorm(n, mean = 0, sd = 1)
```

Note that this uses parameters `mean` ($\mu$) and `sd` ($\sigma$) (but not the variance $\sigma^2$ as we commonly do in probability)!

# Normal distribution

In R,

```
dnorm(x, mean = 0, sd = 1, log = FALSE)
pnorm(q, mean = 0, sd = 1, lower.tail = TRUE, log.p = FALSE)
qnorm(p, mean = 0, sd = 1, lower.tail = TRUE, log.p = FALSE)
rnorm(n, mean = 0, sd = 1)
```

Note that this uses parameters `mean` ($\mu$) and `sd` ($\sigma$) (but not the variance $\sigma^2$ as we commonly do in probability)!

Why?

# Normal distribution

R parametrizes the normal family as the *location-scale* family generated by the standard normal.

# Normal distribution

R parametrizes the normal family as the *location-scale* family generated by the standard normal.

We all know:

$X$ is normal with mean $\mu$ and sd $\sigma \Leftrightarrow Z = \dfrac{X-\mu}{\sigma}$ is standard normal.

I.e.,

$$\mathbb{P}(X \leq x) = \mathbb{P}\left(Z \leq \frac{x-\mu}{\sigma}\right) = \Phi\left(\frac{x-\mu}{\sigma}\right)$$

where $\Phi$ is the CDF of the standard normal distribution.

# Normal distribution

For the densities:

$$\frac{d}{dx}\mathbb{P}(X \leq x) = \Phi'\left(\frac{x-\mu}{\sigma}\right)\frac{1}{\sigma} = \frac{1}{\sigma}\phi\left(\frac{x-\mu}{\sigma}\right)$$

where $\phi$ is the density of the standard normal distribution.

# Normal distribution

For the densities:

$$\frac{d}{dx}\mathbb{P}(X \leq x) = \Phi'\left(\frac{x-\mu}{\sigma}\right)\frac{1}{\sigma} = \frac{1}{\sigma}\phi\left(\frac{x-\mu}{\sigma}\right)$$

where $\phi$ is the density of the standard normal distribution.

There is nothing special about the standard normal here: whenever the numeric random variable $Z$ has CDF $F$ and density $f$, the location-scale family generated by this distribution has CDFs and densities

$$F\left(\frac{x-\mu}{\sigma}\right), \qquad \frac{1}{\sigma}f\left(\frac{x-\mu}{\sigma}\right).$$

# Rate and scale

The reciprocal value of scale is *rate*.

# Rate and scale

The reciprocal value of scale is *rate*.

Somewhat confusingly, for the exponential distribution R takes the "usual" parametrization with rate parameter $\lambda$, i.e., densities

$$f_{\text{exponential}(\lambda)}(x) = \begin{cases} \lambda e^{-\lambda x}, & x > 0, \\ 0, & \text{otherwise.} \end{cases}$$

Clearly, $\sigma = 1/\lambda$ would be the corresponding scale parameter.

# Rate and scale

The reciprocal value of scale is *rate*.

Somewhat confusingly, for the exponential distribution R takes the "usual" parametrization with rate parameter $\lambda$, i.e., densities

$$f_{\text{exponential}(\lambda)}(x) = \begin{cases} \lambda e^{-\lambda x}, & x > 0, \\ 0, & \text{otherwise.} \end{cases}$$

Clearly, $\sigma = 1/\lambda$ would be the corresponding scale parameter.

As a "compromise", for the gamma distribution (which includes the exponential distribution as a special case), one can use both `rate` or `scale`!

# Discrete distributions

The most common discrete distributions are

- The binomial distribution with parameters $n$ (`size`) and $p$ (`prob`), with density (probability mass function)

$$f_{\text{binomial}(n,p)}(x) = \begin{cases} \binom{n}{p} p^x (1-p)^{n-x}, & x \in \{0, \ldots, n\}, \\ 0, & \text{otherwise} \end{cases}$$

  In R, `dbinom()` etc.

# Discrete distributions

The most common discrete distributions are

- The binomial distribution with parameters $n$ (`size`) and $p$ (`prob`), with density (probability mass function)

$$f_{\text{binomial}(n,p)}(x) = \begin{cases} \binom{n}{p} p^x (1-p)^{n-x}, & x \in \{0, \ldots, n\}, \\ 0, & \text{otherwise} \end{cases}$$

  In R, `dbinom()` etc.
- The Poisson distribution with parameter $\lambda$, with density

$$f_{\text{Poisson}(\lambda)}(x) = \begin{cases} \frac{\lambda^x}{x!} e^{-\lambda}, & x \in 0, 1, 2, \ldots, \\ 0, & \text{otherwise} \end{cases}$$

  In R, `dpois()` etc.

# Discrete distributions

These densities are not with respect to Lebesgue measure (hence giving integrals) but with respect to counting measure on the integers (hence giving sums).

More in probability theory!

# Sampling from values

Using `sample()`, we can sample given values with or without replacement:

```
sample(x, size, replace = FALSE, prob = NULL)
```

E.g., to generate a random permutation of the numbers from 1 to 10:

```
R> sample(1 : 10, 10)
 [1]  4  1  7  8  9 10  5  3  6  2
```

# Sampling from values

Using `sample()`, we can sample given values with or without replacement:

```
sample(x, size, replace = FALSE, prob = NULL)
```

E.g., to generate a random permutation of the numbers from 1 to 10:

```
R> sample(1 : 10, 10)
 [1]  4  1  7  8  9 10  5  3  6  2
```

E.g., to randomly draw 13 numbers from 1 : 7 with replacement:

```
R> sample(1 : 7, 13, replace = TRUE)
 [1] 7 5 5 5 4 2 3 7 3 1 5 6 6
```

# Simulation experiments

Once we can draw (pseudo) random numbers, we can perform simulation experiments.

E.g., we know from probability theory that if $X_1, X_2, \ldots$ are i.i.d. (independent identically distributed) $\sim F$ (with distribution function $F$), then by the law of large numbers,

$$\frac{1}{n}\sum_{i=1}^{n} g(X_i) \to \mathbb{E}(g(X)) = \int g(x)\, dF(x).$$

# Simulation experiments

Once we can draw (pseudo) random numbers, we can perform simulation experiments.

E.g., we know from probability theory that if $X_1, X_2, \ldots$ are i.i.d. (independent identically distributed) $\sim F$ (with distribution function $F$), then by the law of large numbers,

$$\frac{1}{n}\sum_{i=1}^{n} g(X_i) \to \mathbb{E}(g(X)) = \int g(x)\, dF(x).$$

We can use this to determine $\mathbb{E}(g(X))$ by *Monte Carlo simulation*: more on this next week.

# Simulation experiments

For now, let us illustrate the central limit theorem.

# Simulation experiments

For now, let us illustrate the central limit theorem.

E.g., if $X$ is binomial with parameters $m$ and $p$ and

$$Z = \frac{X - mp}{\sqrt{mp(1-p)}},$$

then $Z$ is approximately standard normal when $m$ gets large.

# Simulation experiments

For now, let us illustrate the central limit theorem.

E.g., if $X$ is binomial with parameters $m$ and $p$ and

$$Z = \frac{X - mp}{\sqrt{mp(1-p)}},$$

then $Z$ is approximately standard normal when $m$ gets large.

How can we illustrate via a simulation experiment?

# Simulation experiments

For now, let us illustrate the central limit theorem.

E.g., if $X$ is binomial with parameters $m$ and $p$ and

$$Z = \frac{X - mp}{\sqrt{mp(1-p)}},$$

then $Z$ is approximately standard normal when $m$ gets large.

How can we illustrate via a simulation experiment?

- Generate a sample $Z_1, \ldots, Z_n$ from the distribution of $Z$.

# Simulation experiments

For now, let us illustrate the central limit theorem.

E.g., if $X$ is binomial with parameters $m$ and $p$ and

$$Z = \frac{X - mp}{\sqrt{mp(1-p)}},$$

then $Z$ is approximately standard normal when $m$ gets large.

How can we illustrate via a simulation experiment?

- Generate a sample $Z_1, \ldots, Z_n$ from the distribution of $Z$.
- Investigate the *empirical distribution* of the sample, and compare it to reference distribution.

# Simulation experiments

For now, let us illustrate the central limit theorem.

E.g., if $X$ is binomial with parameters $m$ and $p$ and

$$Z = \frac{X - mp}{\sqrt{mp(1-p)}},$$

then $Z$ is approximately standard normal when $m$ gets large.

How can we illustrate via a simulation experiment?

- Generate a sample $Z_1, \ldots, Z_n$ from the distribution of $Z$.
- Investigate the *empirical distribution* of the sample, and compare it to reference distribution.

This needs $n$ large for the approximating the underlying distribution by the empirical distribution.

# Simulation experiments

To generate a sample of size $n$ from the standardized binomial with parameters $m$ and $p$, i.e.,

$$Z_i = \frac{X_i - mp}{\sqrt{mp(1-p)}}, \qquad X_i \sim \text{binomial}(m, p),$$

we can do

```
R> simbin <- function(n, m, p) {
+     (rbinom(n, size = m, prob = p) - m * p) / sqrt(m * p * (1 - p))
+ }
```

# Simulation experiments

To generate a sample of size $n$ from the standardized binomial with parameters $m$ and $p$, i.e.,

$$Z_i = \frac{X_i - mp}{\sqrt{mp(1-p)}}, \qquad X_i \sim \text{binomial}(m, p),$$

we can do

```
R> simbin <- function(n, m, p) {
+     (rbinom(n, size = m, prob = p) - m * p) / sqrt(m * p * (1 - p))
+ }
```

E.g.,

```
R> z <- simbin(1000, 200, 0.4)
```

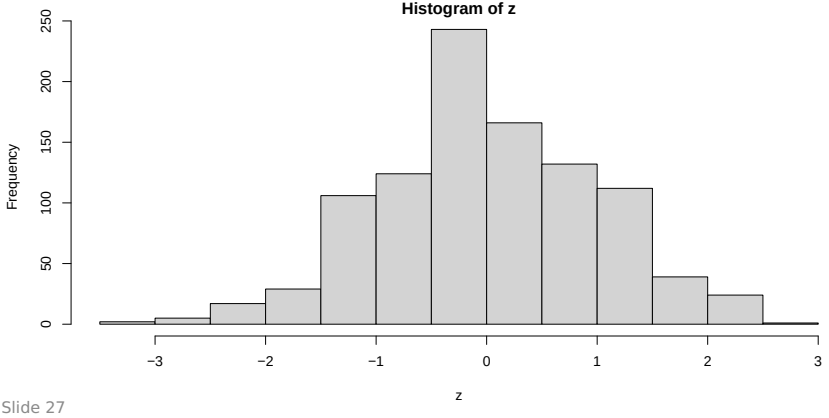
# Simulation experiments

How can we now investigate the empirical distribution?

# Simulation experiments

How can we now investigate the empirical distribution? Usual answer: histogram.

```
R> hist(z)
```



# Simulation experiments

But does this look “normal”?

# Simulation experiments

But does this look “normal”?

One would need to compare against the reference density ($\phi$).

# Simulation experiments

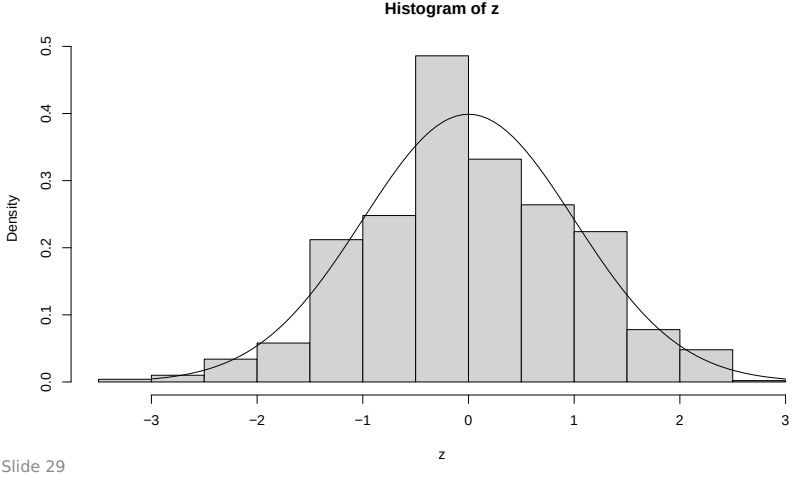
But does this look “normal”?

One would need to compare against the reference density ($\phi$).

One could do so by doing the histogram on the probability scale and superimposing the density of the standard normal.

# Simulation experiments

```
R> hist(z, probability = TRUE); curve(dnorm, -3, 3, add = TRUE)
```



# Simulation experiments

But for comparing distributions it is much better to do a quantile-quantile (QQ plot)!

# Simulation experiments

But for comparing distributions it is much better to do a quantile-quantile (QQ plot)!

If the distributions are the same, the QQ plot will be close to the first median.

# Simulation experiments

But for comparing distributions it is much better to do a quantile-quantile (QQ plot)!

If the distributions are the same, the QQ plot will be close to the first median.

If the distributions are the same modulo a location-scale transformation, the QQ plot will be close to a straight line.

# Simulation experiments

But for comparing distributions it is much better to do a quantile-quantile (QQ plot)!

If the distributions are the same, the QQ plot will be close to the first median.

If the distributions are the same modulo a location-scale transformation, the QQ plot will be close to a straight line.

Why?

# Simulation experiments

Suppose

$$G(x) = F\left(\frac{x-\mu}{\sigma}\right), \qquad F(z_\alpha) = \alpha.$$

Then

$$G(x) = \alpha \Leftrightarrow F\left(\frac{x-\mu}{\sigma}\right) = \alpha \Leftrightarrow \frac{x-\mu}{\sigma} = z_\alpha \Leftrightarrow x = \mu + \sigma z_\alpha.$$

(As we know for the normal distribution.)

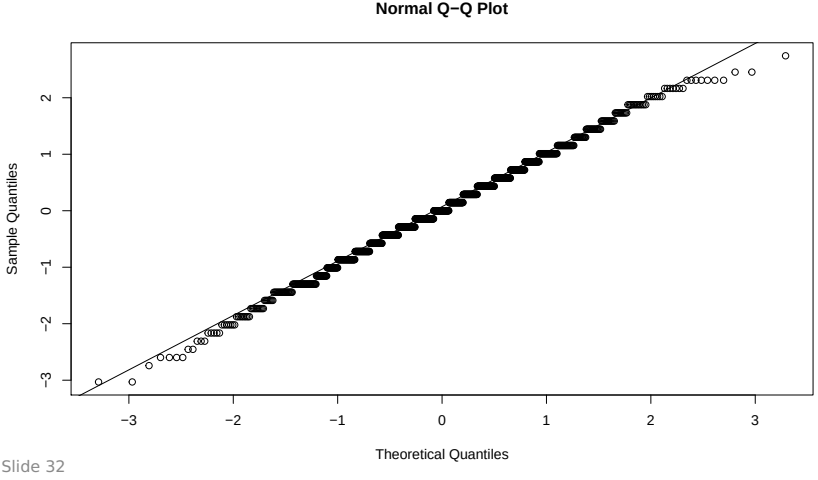
So the quantile functions $Q_F$ and $Q_G$ of $F$ and $G$ satisfy

$$Q_G(\alpha) = \mu + \sigma Q_F(\alpha)$$

and plotting $(Q_F(\alpha), Q_G(\alpha))$ gives a straight line!

# Simulation experiments

```
R> qqnorm(z); qqline(z)
```



# Simulation experiments

If we integrate the QQ plot into the simulation, we can create nice movies illustrating the CLT, i.e., how the approximation becomes better when increasing *m*:

```
R> simbin <- function(n, m, p) {
+     z <- ((rbinom(n, size = m, prob = p) - m * p) /
+           sqrt(m * p * (1 - p)))
+     qqnorm(z, ylim = c(-4, 4), main = paste("QQ-plot, m =", m))
+     qqline(z)
+ }
```

# Simulation experiments

A simple movie:

```
R> for(m in seq(1, 100, 3)) {
+     simbin(1000, m, 0.4)
+     Sys.sleep(1)
+ }
```

# Simulation experiments

Everyone knows that for large $m$, the binomial distribution can be approximated by the normal distribution.

What is much less known is that for large $\lambda$, the Poisson distribution can also be approximated by the normal distribution:

If If $X$ is Poisson with parameter $\lambda$ and

$$Z = \frac{X - \lambda}{\sqrt{\lambda}},$$

then $Z$ is approximately standard normal as $\lambda$ gets large.

# Simulation experiments

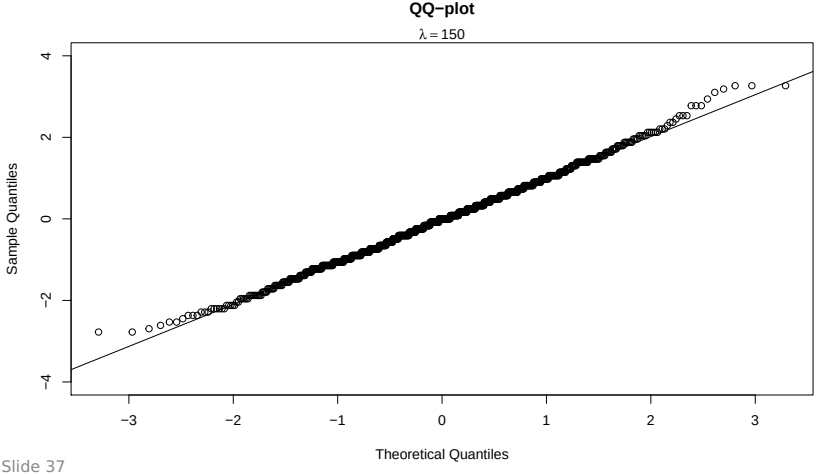
To illustrate, we can use

```
R> simpois <- function(n, lambda) {
+     z <- (rpois(n, lambda = lambda) - lambda) / sqrt(lambda)
+     qqnorm(z, ylim = c(-4, 4), main = "QQ-plot")
+     qqline(z)
+     mtext(bquote(lambda == .(lambda),), 3)
+ }
```

E.g., for $\lambda = 150$:

# Simulation experiments

```
R> simpois(1000, 150)
```



# Simulation experiments

A simple movie:

```
R> for(lambda in seq(1, 100, 3)) {
+     simpois(1000, lambda)
+     Sys.sleep(1)
+ }
```

# Outline

- Motivation
- Basics
- Inverse transform method
- Acceptance-rejection method

# Introduction

How can we simulate from distributions which R does not already implement?

(And how does R actually implement these simulations?)

# Introduction

How can we simulate from distributions which R does not already implement?

(And how does R actually implement these simulations?)

The two most important methods for sampling from a distribution are

- the inverse transform method
- the acceptance-rejection method

# Quantiles

What precisely is a quantile?

# Quantiles

What precisely is a quantile?

Ideally, if $F$ is a distribution function, then the $\alpha$-quantile $x_\alpha = F^{-1}(\alpha)$ solves

$$F(x) = \alpha.$$

However,

- The solution may not be unique ($F$ could be flat in some interval)

# Quantiles

What precisely is a quantile?

Ideally, if $F$ is a distribution function, then the $\alpha$-quantile $x_\alpha = F^{-1}(\alpha)$ solves

$$F(x) = \alpha.$$

However,

- The solution may not be unique ($F$ could be flat in some interval)
- The solution may not even exist!

Remember: if $F$ is a CDF, it is right-continuous with left limits!

# Quantiles

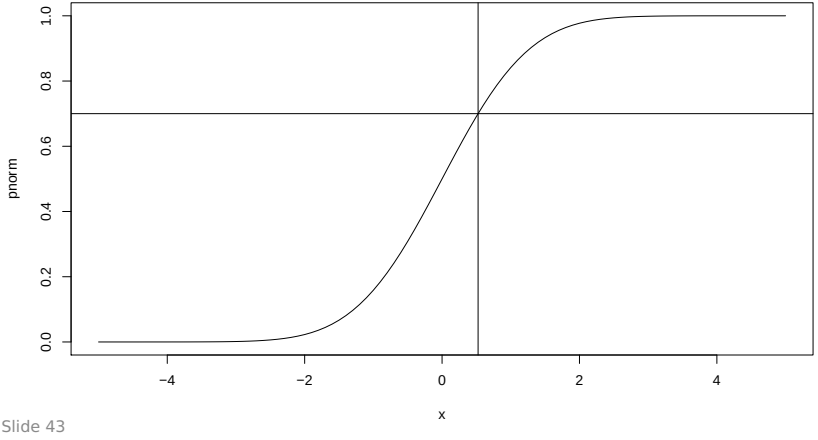
The CDF of the standard normal is continuous and increasing, so

$$\Phi(x) = \alpha$$

has a unique solution for all $0 < \alpha < 1$:

# Quantiles

```
R> p <- 0.7; plot(pnorm, -5, 5); abline(h = p); abline(v = qnorm(p))
```



# Quantiles

The binomial with parameters $m$ and $p$ has its support in $\{0, \ldots, m\}$. So the CDF has jumps at these points, and is flat otherwise.

The equation

$$\text{pbinom}(x, m, p) = \alpha$$

only has a solution for

$$\alpha \in \{\text{pbinom}(0, m, p), \text{pbinom}(1, m, p), \ldots, \text{pbinom}(m, m, p)\}.$$

# Quantiles

The binomial with parameters $m$ and $p$ has its support in $\{0, \ldots, m\}$. So the CDF has jumps at these points, and is flat otherwise.

The equation

$$\text{pbinom}(x, m, p) = \alpha$$

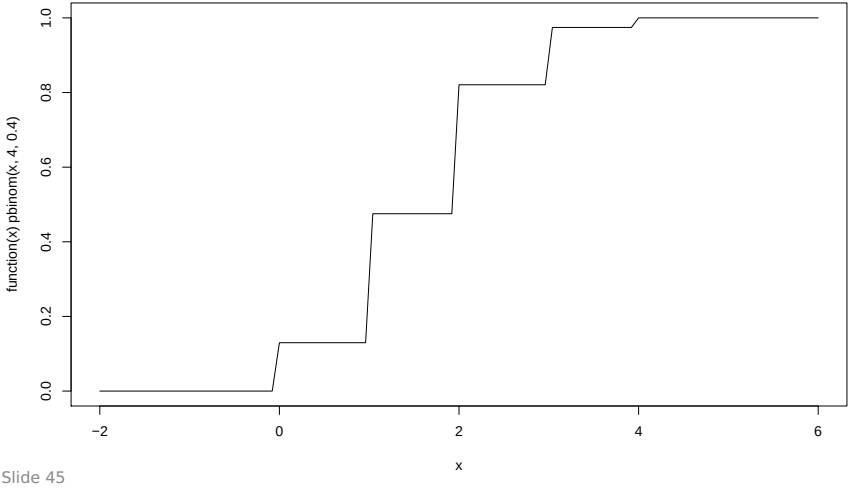
only has a solution for

$$\alpha \in \{\text{pbinom}(0, m, p), \text{pbinom}(1, m, p), \ldots, \text{pbinom}(m, m, p)\}.$$

To illustrate for $m = 4$ and $p = 0.4$:

# Quantiles

```
R> plot(function(x) pbinom(x, 4, 0.4), -2, 6)
```



# Quantiles

This is actually a bit silly in context as we don’t see the jumps. Can we do better?

We can create a step function $s$ which does

$$s(x) = \begin{cases} y_0, & x < x_1, \\ y_i, & x_i < x < x_{i+1}, 1 \leq i < n, \\ y_n, & x > x_n. \end{cases}$$

(and specify whether we want right or left continuous).

# Quantiles

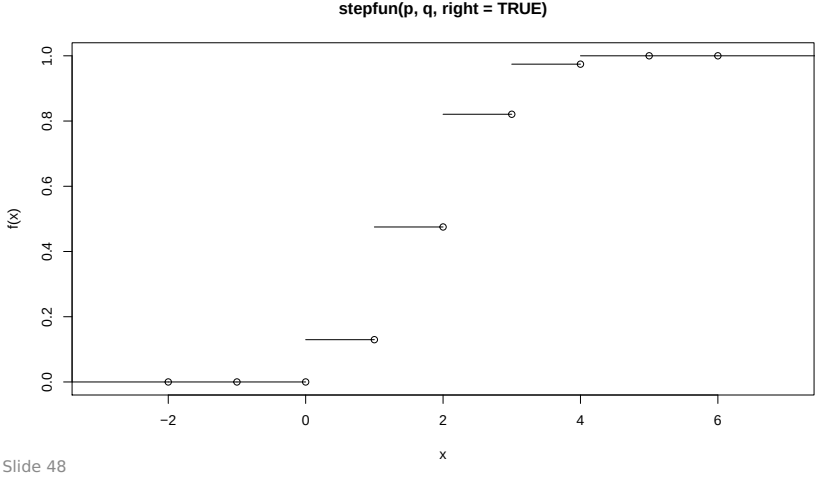
E.g.,

```
R> p <- seq(-2, 6)
R> q <- c(0, pbinom(p, size = 4, p = 0.4))
R> s <- stepfun(p, q, right = TRUE)
```

With this:

# Quantiles

```
R> plot(s, vertical = FALSE)
```



# Quantiles

Voilà! (Well, only plotting at the support would have been even better.)

# Quantiles

Voilà! (Well, only plotting at the support would have been even better.)

Ok, so we can now see the jumps, but what should we do about the quantiles?

# Quantiles

Voilà! (Well, only plotting at the support would have been even better.)

Ok, so we can now see the jumps, but what should we do about the quantiles?

E.g., for $\alpha = 0.7$, we see that

$$\begin{aligned} F(1) &= \text{pbinom}(1, 4, 0.4) = 0.4752 < 0.7, \\ F(2) &= \text{pbinom}(2, 4, 0.4) = 0.8208 > 0.7 \end{aligned}$$

so should we take 1 or 2? (Or perhaps the solution using the linear interpolation of $F(1)$ and $F(2)$?)

# Quantiles

Voilà! (Well, only plotting at the support would have been even better.)

Ok, so we can now see the jumps, but what should we do about the quantiles?

E.g., for $\alpha = 0.7$, we see that

$$\begin{aligned} F(1) &= \text{pbinom}(1, 4, 0.4) = 0.4752 < 0.7, \\ F(2) &= \text{pbinom}(2, 4, 0.4) = 0.8208 > 0.7 \end{aligned}$$

so should we take 1 or 2? (Or perhaps the solution using the linear interpolation of $F(1)$ and $F(2)$?)

The common convention is to use the smallest $x$ such that $F(x) \geq \alpha$.

# Quantiles

Formally: if $F$ is a distribution function, its quantile function is defined as

$$Q_F(u) = F^{-1}(u) = \inf\{x : F(x) \geq u\}.$$

# Quantiles

Formally: if $F$ is a distribution function, its quantile function is defined as

$$Q_F(u) = F^{-1}(u) = \inf\{x : F(x) \geq u\}.$$

Note that this is not necessarily the inverse function commonly denoted by $F^{-1}$! As discussed . . .

Hence, to make the distinction clear, one sometimes writes $F^{\leftarrow}$ or (my preference) $Q_F$.

# Quantiles

Formally: if $F$ is a distribution function, its quantile function is defined as

$$Q_F(u) = F^{-1}(u) = \inf\{x : F(x) \geq u\}.$$

Note that this is not necessarily the inverse function commonly denoted by $F^{-1}$! As discussed . . .

Hence, to make the distinction clear, one sometimes writes $F^{\leftarrow}$ or (my preference) $Q_F$.

Hence in our case, the 0.7 quantile must be 2:

```
R> qbinom(0.7, 4, 0.4)

[1] 2
```

# Quantile transform method

Now let $F$ be a CDF, $Q_F$ its quantile function, and $U$ be a standard uniform random variable: $U \sim U_{0,1}$.

# Quantile transform method

Now let $F$ be a CDF, $Q_F$ its quantile function, and $U$ be a standard uniform random variable: $U \sim U_{0,1}$.

By the definition of the quantile function,

$$Q_F(U) \leq x \Leftrightarrow U \leq F(x).$$

# Quantile transform method

Now let $F$ be a CDF, $Q_F$ its quantile function, and $U$ be a standard uniform random variable: $U \sim U_{0,1}$.

By the definition of the quantile function,

$$Q_F(U) \leq x \Leftrightarrow U \leq F(x).$$

Hence, as $U$ is standard uniform,

$$\mathbb{P}(Q_F(U) \leq x) = \mathbb{P}(U \leq F(x)) = F(x)$$

# Quantile transform method

Now let $F$ be a CDF, $Q_F$ its quantile function, and $U$ be a standard uniform random variable: $U \sim U_{0,1}$.

By the definition of the quantile function,

$$Q_F(U) \leq x \Leftrightarrow U \leq F(x).$$

Hence, as $U$ is standard uniform,

$$\mathbb{P}(Q_F(U) \leq x) = \mathbb{P}(U \leq F(x)) = F(x)$$

I.e., the *quantile transform* $Q_F(U)$ has distribution (function) $F$!

# Quantile transform method

This nice mathematical theorem forms the basis of the *quantile transform method*:

> To draw $X \sim F$, take $X = Q_F(U)$ with $U \sim U_{0,1}$.

# Quantile transform method

This nice mathematical theorem forms the basis of the *quantile transform method*:

> To draw $X \sim F$, take $X = Q_F(U)$ with $U \sim U_{0,1}$.

This looks nice and very general, but is useful only if we can efficiently compute the quantile function $Q_F$!

Of course, we can always use repeated bisection, but that may not be efficient enough.

# Quantile transform method: Example 1

Suppose we want to draw from the distribution with density

$$f(x) = 3x^2, \quad 0 < x < 1.$$

(This is a Beta distribution.)

# Quantile transform method: Example 1

Suppose we want to draw from the distribution with density

$$f(x) = 3x^2, \quad 0 < x < 1.$$

(This is a Beta distribution.)

Then for $0 \leq x \leq 1$, the corresponding CDF $F$ is

$$F(x) = \int_0^x f(t)\,dt = x^3.$$

and the $Q_F$ is determined via

$$F(x) = x^3 = u \Rightarrow x = Q_F(u) = u^{1/3}.$$

# Quantile transform method: Example 1

Suppose we want to draw from the distribution with density

$$f(x) = 3x^2, \quad 0 < x < 1.$$

(This is a Beta distribution.)

Then for $0 \leq x \leq 1$, the corresponding CDF $F$ is

$$F(x) = \int_0^x f(t)\,dt = x^3.$$

and the $Q_F$ is determined via

$$F(x) = x^3 = u \Rightarrow x = Q_F(u) = u^{1/3}.$$

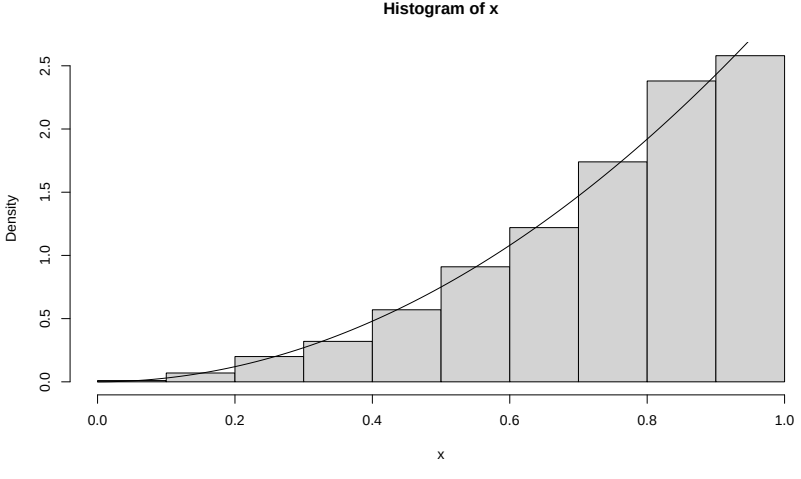
Thus, if $U$ is standard uniform, $U^{1/3} \sim F$.

# Quantile transform method: Example 1

To illustrate, generate a sample, draw its (probability) histogram, and add the true density:

```
R> n <- 1000
R> x <- runif(n) ^ (1/3)
R> hist(x, probability = TRUE)
R> y <- seq(0, 1, .01)
R> lines(y, 3 * y ^ 2)
```

# Quantile transform method: Example 1



# Quantile transform method: Example 2

The exponential distribution with rate parameter $\lambda$ has

$$f(t) = \lambda e^{-\lambda t}, \qquad F(t) = 1 - e^{-\lambda t}, \qquad t \geq 0.$$

# Quantile transform method: Example 2

The exponential distribution with rate parameter $\lambda$ has

$$f(t) = \lambda e^{-\lambda t}, \qquad F(t) = 1 - e^{-\lambda t}, \qquad t \geq 0.$$

For the quantile function, we get

$$1 - e^{-\lambda t} = u \Rightarrow x = Q_F(u) = -\log(1-u)/\lambda.$$

# Quantile transform method: Example 2

The exponential distribution with rate parameter $\lambda$ has

$$f(t) = \lambda e^{-\lambda t}, \qquad F(t) = 1 - e^{-\lambda t}, \qquad t \geq 0.$$

For the quantile function, we get

$$1 - e^{-\lambda t} = u \Rightarrow x = Q_F(u) = -\log(1-u)/\lambda.$$

Thus, if $U$ is standard uniform, $-\log(1-U)/\lambda \sim \text{exponential}(\lambda)$.

# Quantile transform method: Example 2

The exponential distribution with rate parameter $\lambda$ has

$$f(t) = \lambda e^{-\lambda t}, \qquad F(t) = 1 - e^{-\lambda t}, \qquad t \geq 0.$$

For the quantile function, we get

$$1 - e^{-\lambda t} = u \Rightarrow x = Q_F(u) = -\log(1-u)/\lambda.$$

Thus, if $U$ is standard uniform, $-\log(1-U)/\lambda \sim \text{exponential}(\lambda)$.

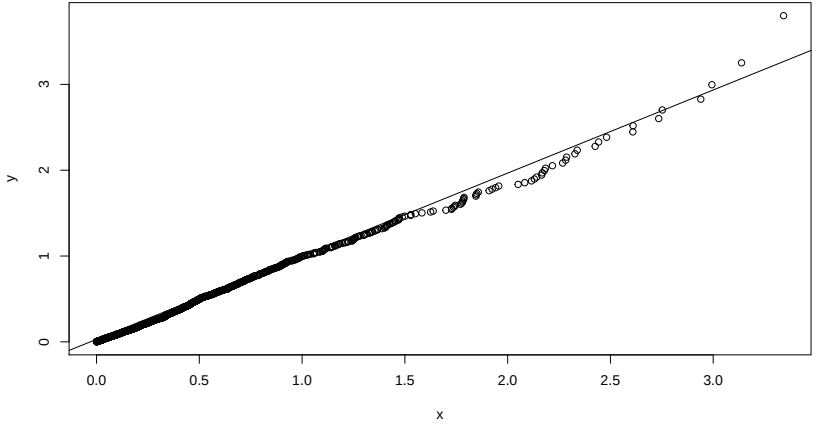
In fact, $1-U$ is standard uniform too. So we can simplify to $-\log(U)/\lambda \sim \text{exponential}(\lambda)$.

# Quantile transform method: Example 2

To illustrate, generate a sample, and do a QQ plot against the exponential distribution.

```
R> rate <- 2
R> n <- 1000
R> x <- -log(runif(n)) / rate
R> y <- qexp(ppoints(n), rate = rate)
R> qqplot(x, y)
R> qqline(x, distribution = function(p) qexp(p, rate = rate))
```

# Quantile transform method: Example 2



# Quantile transform method: Example 3

The simplest (non-trivial) discrete distribution is the Bernoulli distribution:

$$\mathbb{P}(X = 1) = p, \quad \mathbb{P}(X = 0) = q = 1 - p.$$

(Of course, this is a binomial distribution with size 1.)

# Quantile transform method: Example 3

The simplest (non-trivial) discrete distribution is the Bernoulli distribution:

$$\mathbb{P}(X = 1) = p, \quad \mathbb{P}(X = 0) = q = 1 - p.$$

(Of course, this is a binomial distribution with size 1.)

This has CDF

$$F(x) = \begin{cases} 0, & x < 0, \\ 1 - p, & 0 \le x < 1, \\ 1 & x \ge 1. \end{cases}$$

# Quantile transform method: Example 3

The simplest (non-trivial) discrete distribution is the Bernoulli distribution:

$$\mathbb{P}(X = 1) = p, \quad \mathbb{P}(X = 0) = q = 1 - p.$$

(Of course, this is a binomial distribution with size 1.)

This has CDF

$$F(x) = \begin{cases} 0, & x < 0, \\ 1 - p, & 0 \le x < 1, \\ 1 & x \ge 1. \end{cases}$$

Hence the quantile function is

$$Q_F(u) = \begin{cases} 1, & u > 1 - p, \\ 0, & \text{otherwise.} \end{cases}$$

# Quantile transform method: Example 3

Thus, if $U$ is standard uniform, then $I(U > 1-p)$ has a Bernoulli distribution with parameter $p$.

But

$$U > 1-p \Leftrightarrow 1-U < p.$$

So we can simplify to $I(U < p)$.

# Quantile transform method: Example 3

Thus, if $U$ is standard uniform, then $I(U > 1-p)$ has a Bernoulli distribution with parameter $p$.

But

$$U > 1-p \Leftrightarrow 1-U < p.$$

So we can simplify to $I(U < p)$.

Which is what we would presumably have used in the first place, without knowing the quantile transform method.

# Quantile transform method: Example 4

Suppose we run a Bernoulli experiment until the first success. What is the distribution of the number $X$ of failures before the first success?

# Quantile transform method: Example 4

Suppose we run a Bernoulli experiment until the first success. What is the distribution of the number $X$ of failures before the first success?

We have

$$\mathbb{P}(X = n) = \mathbb{P}(n \text{ failures, then success}) = \mathbb{P}(\text{failure})^n \mathbb{P}(\text{success}) = q^n p$$

# Quantile transform method: Example 4

Suppose we run a Bernoulli experiment until the first success. What is the distribution of the number $X$ of failures before the first success?

We have

$$\mathbb{P}(X = n) = \mathbb{P}(n \text{ failures, then success}) = \mathbb{P}(\text{failure})^n \mathbb{P}(\text{success}) = q^n p$$

This is the *geometric distribution* with parameter $p$.

# Quantile transform method: Example 4

Suppose we run a Bernoulli experiment until the first success. What is the distribution of the number $X$ of failures before the first success?

We have

$$\mathbb{P}(X = n) = \mathbb{P}(n \text{ failures, then success}) = \mathbb{P}(\text{failure})^n \mathbb{P}(\text{success}) = q^n p$$

This is the *geometric distribution* with parameter $p$.

(This is the parametrization used by R as well: one can also count the number of experiments needed for the first success.)

# Quantile transform method: Example 4

Suppose we run a Bernoulli experiment until the first success. What is the distribution of the number $X$ of failures before the first success?

We have

$$\mathbb{P}(X = n) = \mathbb{P}(n \text{ failures, then success}) = \mathbb{P}(\text{failure})^n \mathbb{P}(\text{success}) = q^n p$$

This is the *geometric distribution* with parameter $p$.

(This is the parametrization used by R as well: one can also count the number of experiments needed for the first success.)

How can we use the quantile transform method to draw from the geometric distribution?

# Quantile transform method: Example 4

In general, how can we find the quantile function for a discrete distribution?

# Quantile transform method: Example 4

In general, how can we find the quantile function for a discrete distribution?

Suppose that the support of the distribution is $x_1 < x_2 < \cdots$ (possibly countably infinite, as for the geometric of Poission).

Then

$$F(x) = F(x_i), \quad x_i \leq x < x_{i+1}.$$

Clearly,

$$F(x_{i-1}) < u \leq F(x_i) \Leftrightarrow Q_F(u) = x_i.$$

Why? $F(x_{i-1}) < u$ and $F(x_i) \geq u$, and we're looking for smallest $x$ such that $F(x) \geq u$!

# Quantile transform method: Example 4

For the geometric distribution (and $x$ a non-negative integer),

$$F(x) = \sum_{i=0}^{x} pq^i = p\frac{1-q^{x+1}}{1-q} = 1-q^{x+1}.$$

# Quantile transform method: Example 4

For the geometric distribution (and $x$ a non-negative integer),

$$F(x) = \sum_{i=0}^{x} pq^i = p\frac{1-q^{x+1}}{1-q} = 1-q^{x+1}.$$

For the quantile function, we thus get

$$\begin{aligned} Q_F(u) = x &\Leftrightarrow 1-q^x < u \leq 1-q^{x+1} \\ &\Leftrightarrow q^x > 1-u \geq q^{x+1} \\ &\Leftrightarrow x < \frac{\log(1-u)}{\log(q)} \leq x+1 \end{aligned}$$

# Quantile transform method: Example 4

Now

$$\begin{aligned} x < \xi \leq x+1 &\iff \xi - 1 \leq x < \xi \\ &\iff x = \text{ceiling}(\xi - 1) = \text{ceiling}(\xi) - 1. \end{aligned}$$

# Quantile transform method: Example 4

Now

$$\begin{aligned} x < \xi \leq x+1 &\iff \xi - 1 \leq x < \xi \\ &\iff x = \text{ceiling}(\xi - 1) = \text{ceiling}(\xi) - 1. \end{aligned}$$

Hence, we found:

$$Q_F(u) = \text{ceiling}\left(\frac{\log(1-u)}{\log(q)}\right) - 1.$$

And thus: if $U$ is standard uniform,

$$\text{ceiling}\left(\frac{\log(1-U)}{\log(1-p)}\right) - 1 \sim \text{geometric}(p).$$

# Outline

- Motivation
- Basics
- Inverse transform method
- Acceptance-rejection method

# Motivation

Suppose we want to sample from the distribution with density

$$f(x) = 6x(1-x), \qquad 0 < x < 1.$$

We can still easily compute

$$F(x) = \int_0^x f(t)\,dt = 6\left(\frac{t^2}{2} - \frac{t^3}{3}\right)\Bigg|_{t=0}^{x} = 3x^2 - 2x^3, \qquad 0 \le x \le 1$$

but solving $F(x) = u$ gives a cubic equation, which is not so "straightforward" to solve.

# Motivation

Suppose we want to sample from the distribution with density

$$f(x) = 6x(1-x), \qquad 0 < x < 1.$$

We can still easily compute

$$F(x) = \int_0^x f(t)\,dt = 6\left(\frac{t^2}{2} - \frac{t^3}{3}\right)\Bigg|_{t=0}^{x} = 3x^2 - 2x^3, \qquad 0 \le x \le 1$$

but solving $F(x) = u$ gives a cubic equation, which is not so "straightforward" to solve.

Can we draw from $F$ without determining $Q_F$?

# Acceptance-rejection method

The acceptance-rejection method allows the following:

We want to draw $X \sim f$ and know to draw $Y \sim g$, where $f(t) \leq cg(t)$ for all $t$.

We can do this as follows: for each random variate required,

1. Draw $y$ from $g$
2. Draw $u$ from $U_{0,1}$
3. If $u < f(y)/(cg(y))$ accept and deliver $x = y$; otherwise, reject $y$ and restart.

# Acceptance-rejection method

The acceptance-rejection method allows the following:

We want to draw $X \sim f$ and know to draw $Y \sim g$, where $f(t) \leq cg(t)$ for all $t$.

We can do this as follows: for each random variate required,

1. Draw $y$ from $g$
2. Draw $u$ from $U_{0,1}$
3. If $u < f(y)/(cg(y))$ accept and deliver $x = y$; otherwise, reject $y$ and restart.

Why does this work?

# Acceptance-rejection method

The acceptance-rejection method allows the following:

We want to draw $X \sim f$ and know to draw $Y \sim g$, where $f(t) \leq cg(t)$ for all $t$.

We can do this as follows: for each random variate required,

1. Draw $y$ from $g$
2. Draw $u$ from $U_{0,1}$
3. If $u < f(y)/(cg(y))$ accept and deliver $x = y$; otherwise, reject $y$ and restart.

Why does this work? Sorry in advanced for the awful notation.

# Acceptance-rejection method

First,

$$\mathbb{P}(\text{accept}|y) = \mathbb{P}\left(U < \frac{f(y)}{cg(y)}\right) = \frac{f(y)}{cg(y)}.$$

Don't worry if $g(y) = 0$: we cannot really draw such $y$.

# Acceptance-rejection method

First,

$$\mathbb{P}(\text{accept}|y) = \mathbb{P}\left(U < \frac{f(y)}{cg(y)}\right) = \frac{f(y)}{cg(y)}.$$

Don't worry if $g(y) = 0$: we cannot really draw such $y$.

Hence, using a variant of the theorem of total probability,

$$\mathbb{P}(\text{accept}) = \int \mathbb{P}(\text{accept}|y)dG(y) = \int \frac{f(y)}{cg(y)}g(y)dy = \frac{1}{c}$$

# Acceptance-rejection method

To see that $X$ has the right distribution:

$$\begin{aligned}
\mathbb{P}(X = x|\text{accept}) &= \frac{\mathbb{P}(Y = x, \text{accept})}{\mathbb{P}(\text{accept})} \\
&= \frac{\mathbb{P}(\text{accept}|x)\mathbb{P}(Y = x)}{\mathbb{P}(\text{accept})} \\
&= \frac{\frac{f(x)}{cg(x)}g(x)}{\frac{1}{c}} \\
&= f(x)
\end{aligned}$$

# Acceptance-rejection method

To see that $X$ has the right distribution:

$$\begin{aligned}
\mathbb{P}(X = x | \text{accept}) &= \frac{\mathbb{P}(Y = x, \text{accept})}{\mathbb{P}(\text{accept})} \\
&= \frac{\mathbb{P}(\text{accept}|x)\mathbb{P}(Y = x)}{\mathbb{P}(\text{accept})} \\
&= \frac{\frac{f(x)}{cg(x)} g(x)}{\frac{1}{c}} \\
&= f(x)
\end{aligned}$$

WOW!

# Acceptance-rejection method

So we see that this method really works!

# Acceptance-rejection method

So we see that this method really works!

We also see that the probability of accepting is $1/c$. Clearly, we want this to be as large as possible, and hence $c$ should be as small as possible.

# Acceptance-rejection method

So we see that this method really works!

We also see that the probability of accepting is $1/c$. Clearly, we want this to be as large as possible, and hence $c$ should be as small as possible.

As $f$ and $g$ are densities,

$$f(t) \leq cg(t) \text{ for all } t \Rightarrow \int f(t)\,dt \leq \int cg(t)\,dt \Rightarrow 1 \leq c.$$

# Acceptance-rejection method

So we see that this method really works!

We also see that the probability of accepting is $1/c$. Clearly, we want this to be as large as possible, and hence $c$ should be as small as possible.

As $f$ and $g$ are densities,

$$f(t) \leq cg(t) \text{ for all } t \Rightarrow \int f(t)\,dt \leq \int cg(t)\,dt \Rightarrow 1 \leq c.$$

Every try accepts with probability $p = 1/c$, so the number of rejects has a geometric distribution with parameter $p$.

On average, the number of tries needed is thus

$$1 + \mathbb{E}(\text{geometric}(p)) = 1 + \sum_{x=0}^{\infty} xpq^x = 1 + \frac{q}{p} = \frac{p+q}{p} = \frac{1}{p} = c.$$

# Acceptance-rejection method

In case you really want to know: for $|q| < 1$,

$$\sum_{x=0}^{\infty} q^x = \frac{1}{1-q}.$$

# Acceptance-rejection method

In case you really want to know: for $|q| < 1$,

$$\sum_{x=0}^{\infty} q^x = \frac{1}{1-q}.$$

Differentiate, interchanging summation and differentiation on the LHS:

$$\sum_{x=0}^{\infty} x q^{x-1} = \frac{1}{(1-q)^2}$$

# Acceptance-rejection method

In case you really want to know: for $|q| < 1$,

$$\sum_{x=0}^{\infty} q^x = \frac{1}{1-q}.$$

Differentiate, interchanging summation and differentiation on the LHS:

$$\sum_{x=0}^{\infty} x q^{x-1} = \frac{1}{(1-q)^2}$$

Now multiply by $pq$:

$$\sum_{x=0}^{\infty} x p q^x = pq \frac{1}{(1-q)^2} = \frac{q}{p}.$$

# Acceptance-rejection method: Example

Suppose we want to sample from the distribution with density

$$f(x) = 6x(1-x), \qquad 0 < x < 1.$$

# Acceptance-rejection method: Example

Suppose we want to sample from the distribution with density

$$f(x) = 6x(1-x), \qquad 0 < x < 1.$$

This is actually a Beta distribution, which generally has density

$$f(x) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{B(\alpha,\beta)}, \qquad 0 < x < 1,$$

where

$$B(\alpha,\beta) = \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)}$$

is the Beta function.

# Acceptance-rejection method: Example

Suppose we want to sample from the distribution with density

$$f(x) = 6x(1-x), \qquad 0 < x < 1.$$

This is actually a Beta distribution, which generally has density

$$f(x) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{B(\alpha,\beta)}, \qquad 0 < x < 1,$$

where

$$B(\alpha,\beta) = \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)}$$

is the Beta function. We see that in our case, $\alpha = \beta = 2$.

# Acceptance-rejection method: Example

How can we use use acceptance-rejection to draw from $f$?

# Acceptance-rejection method: Example

How can we use use acceptance-rejection to draw from $f$?

We need a density $g$ on $(0, 1)$ we know to draw from.

# Acceptance-rejection method: Example

How can we use use acceptance-rejection to draw from $f$?

We need a density $g$ on $(0, 1)$ we know to draw from.

You get three guesses . . .

# Acceptance-rejection method: Example

How can we use use acceptance-rejection to draw from $f$?

We need a density $g$ on $(0, 1)$ we know to draw from.

You get three guesses . . .

Ok, that was too easy. Clearly, we can take the standard uniform.

# Acceptance-rejection method: Example

How can we use use acceptance-rejection to draw from $f$?

We need a density $g$ on $(0, 1)$ we know to draw from.

You get three guesses . . .

Ok, that was too easy. Clearly, we can take the standard uniform.

What about $c$? Well, for $0 < x < 1$

$$\frac{f(x)}{g(x)} = \frac{6x(1-x)}{1} = 6x(1-x) \leq 6$$

so we could take $c = 6$, and accept if

$$\frac{f(y)}{cg(y)} = \frac{6y(1-y)}{6} = y(1-y) > u.$$

# Acceptance-rejection method: Example

Let us write an acceptance-rejection sampler for $f$ which hard-wires $c = 6$ (not so good) but counts the number of tries (good).

# Acceptance-rejection method: Example

```
R> myrbeta22 <- function(n) {
+     k <- 0
+     j <- 0
+     x <- numeric(n)
+     while(k < n) {
+         u <- runif(1)
+         j <- j + 1
+         y <- runif(1)    # random variate from g
+         if(y * (1 - y) > u) {
+             k <- k + 1
+             x[k] <- y
+         }
+     }
+     list(x = x, num_of_iterations = j)
+ }
```

# Acceptance-rejection method: Example

Then to generate a sample of size $n = 1000$:

```
R> res <- myrbeta22(1000)
R> res$num_of_iterations

[1] 6063
```

# Acceptance-rejection method: Example

Then to generate a sample of size $n = 1000$:

```
R> res <- myrbeta22(1000)
R> res$num_of_iterations
```

```
[1] 6063
```

We see that the number of iterations/tries is close to the expected value

$$n \times c = 1000 \times 6 = 6000.$$

# Acceptance-rejection method: Example

Then to generate a sample of size $n = 1000$:

```
R> res <- myrbeta22(1000)
R> res$num_of_iterations
```

```
[1] 6063
```

We see that the number of iterations/tries is close to the expected value

$$n \times c = 1000 \times 6 = 6000.$$

Of course, using $c = 6$ is really silly!

# Acceptance-rejection method: Example

We need that for $0 < x < 1$,

$$\frac{f(x)}{g(x)} = 6x(1-x) \leq c$$

and

$$\max_{0 \leq x \leq 1} 6x(1-x) = 6x(1-x)|_{x=1/2} = \frac{6}{4} = \frac{3}{2}$$

# Acceptance-rejection method: Example

We need that for $0 < x < 1$,

$$\frac{f(x)}{g(x)} = 6x(1-x) \leq c$$

and

$$\max_{0 \leq x \leq 1} 6x(1-x) = 6x(1-x)|_{x=1/2} = \frac{6}{4} = \frac{3}{2}$$

So any $c \geq 3/2$ will work, and $c = 3/2$ is “best possible”.

# Acceptance-rejection method: Example

We need that for $0 < x < 1$,

$$\frac{f(x)}{g(x)} = 6x(1-x) \leq c$$

and

$$\max_{0 \leq x \leq 1} 6x(1-x) = 6x(1-x)|_{x=1/2} = \frac{6}{4} = \frac{3}{2}$$

So any $c \geq 3/2$ will work, and $c = 3/2$ is “best possible”.

Using the best possible $c$, the average number of iterations/tries would go down to

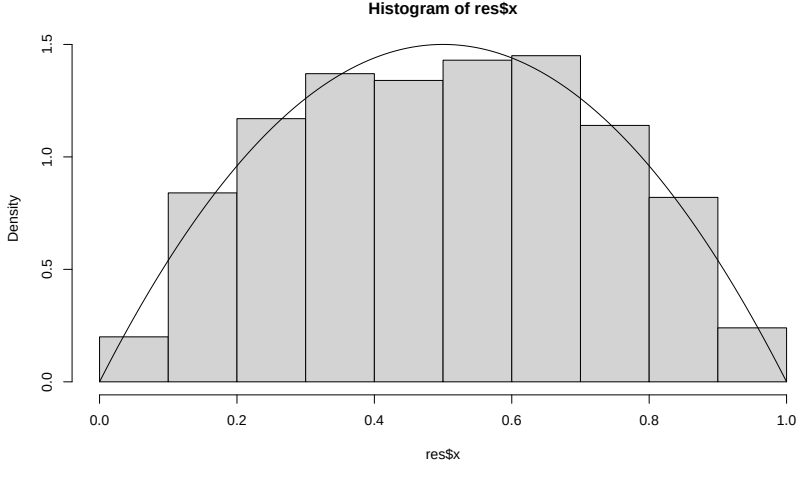
$$n \times \frac{3}{2} = 1000 \times \frac{3}{2} = 1500!$$

# Acceptance-rejection method: Example

To illustrate that our sampler “works”, first using histograms:

```
R> hist(res$x, probability = TRUE)
R> x <- seq(0, 1, by = 0.001)
R> lines(x, dbeta(x, 2, 2))
```

# Acceptance-rejection method: Example



# Acceptance-rejection method: Example

Or (better), using QQ plots:

```
R> qqplot(res$x, qbeta(x, 2, 2))
```