Rejection Sampling, Accept-Reject Method

Mark M. Fredrickson (mfredric@umich.edu)

Computational Methods in Statistics and Data Science (Stats 406)

Rejection Sampling

Conditioning

When generating **Laplace** RVs we used **conditioning** by noting if

$$f(x) = \frac{1}{2}e^{-|x|}$$

then

$$X \mid X > 0 \sim Exp(1).$$

The reverse could also be useful, if we had a **a source of Laplace RVs** we could use it to **generate Exp(1)**.

Let's illustrate this by first getting the **quantile function** and then using the **inversion method** to make `rlaplace`.

$f(x)$ to $F(x)$

As we did before, let's split up $f(x) = (1/2)e^{-|x|}$ when $x < 0$ and $x \geq 0$:

$$f(x) = \begin{cases} \frac{1}{2}e^{x} & : x < 0 \\ \frac{1}{2}e^{-x} & x \geq 0 \end{cases}$$

Then we have a **piece-wise CDF**. For $x < 0$,

$$F(x) = \frac{1}{2}\left(\int_{-\infty}^{x} e^{t}\, dt\right) = \frac{1}{2}e^{x}$$

For $x \geq 0$ we can decompose $F(x)$ as:

$$F(x) = F(0) + P(0 \leq X \leq x) = \frac{1}{2} + \frac{1}{2}\left(\int_{0}^{x} e^{-t}\, dt\right) = 1 - \frac{1}{2}e^{-x}$$

The quantile function will also be **piece-wise** with the change at:

$$F(0) = 1/2 \Rightarrow Q(1/2) = 0$$

Solving the CDF leads to

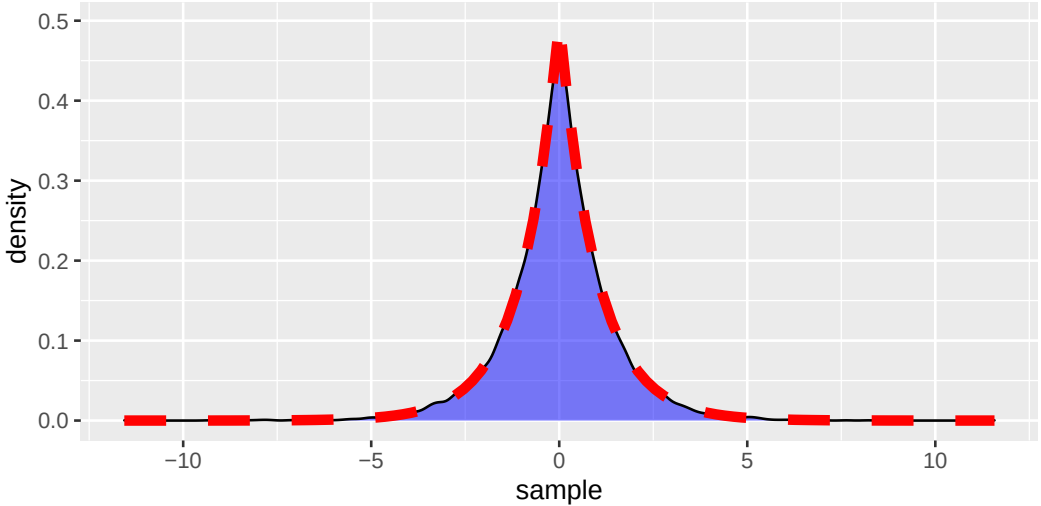
$$F(x) = \begin{cases} \frac{1}{2}e^{x} & : x < 0 \\ 1 - \frac{1}{2}e^{-x} & : x \geq 0 \end{cases} \Rightarrow Q(u) = \begin{cases} \log(2u) : 0 \leq u < \frac{1}{2} \\ \log\left(\frac{1}{2-2u}\right) : \frac{1}{2} \leq u \leq 1 \end{cases}$$

R implementation

$$Q(u) = \begin{cases} \log(2u) : 0 \leq u < \frac{1}{2} \\ \log\left(\frac{1}{2-2u}\right) : \frac{1}{2} \leq u \leq 1 \end{cases}$$

```
> rlaplace <- function(n) {
+     u <- runif(n)
+     ifelse(u < 1/2, log(2 * u), log(1 / (2 - 2 * u)))
+ }
```

Density plot



Rejection Sampling for $Exp(1)$

Suppose we have `rlaplace` but not `rexp`. We saw that if $X \sim$ Laplace(0) , then $X \mid X > 0 \sim$ Exp(1). Let's do that:

```
> x <- rlaplace(1000)
> x_positive <- keep(x, x > 0)
> mean(x_positive) # should be close to 1

[1] 0.9407

> t.test(x_positive, conf.level = 0.999)$conf.int

[1] 0.8021 1.0793
attr(,"conf.level")
[1] 0.999
```

Variance of the Estimator

The method worked fairly well, but **had to throw away about 50% of RVs:**

```
> length(x_positive)

[1] 485
```

Recall the variance of a the sample mean is:

$$\text{Var}\left(\bar{X}\right) = \frac{1}{n}\text{Var}\left(X\right)$$

by throwing away samples, we have **increased the variance** (relative to having a way to keep all the samples).

Rejection Sampling in General

The idea of **rejection sampling** can be used when our **target RV** can be expressed as a subset of another **candidate RV**.

Examples:

- Conditioning on the variable itself, $X \sim \text{Poisson}(\lambda)$, $Y = X \mid X$ is odd
- Truncated distributions, e.g. $X \sim N(0, 1)$, $Y = X \mid a < X < b$
- Uniform points on the unit circle: $U_1, U_2 \sim U(-1, 1)$ (independent), $(V_1, V_2) = (U_1, U_2) \mid U_1^2 + U_2^2 \leq 1$

Procedure: generate from candidate, only keep results that meet criteria.

Example: Truncated Normal Distribution

A **truncated distribution** takes a given distribution and **limits the support**.

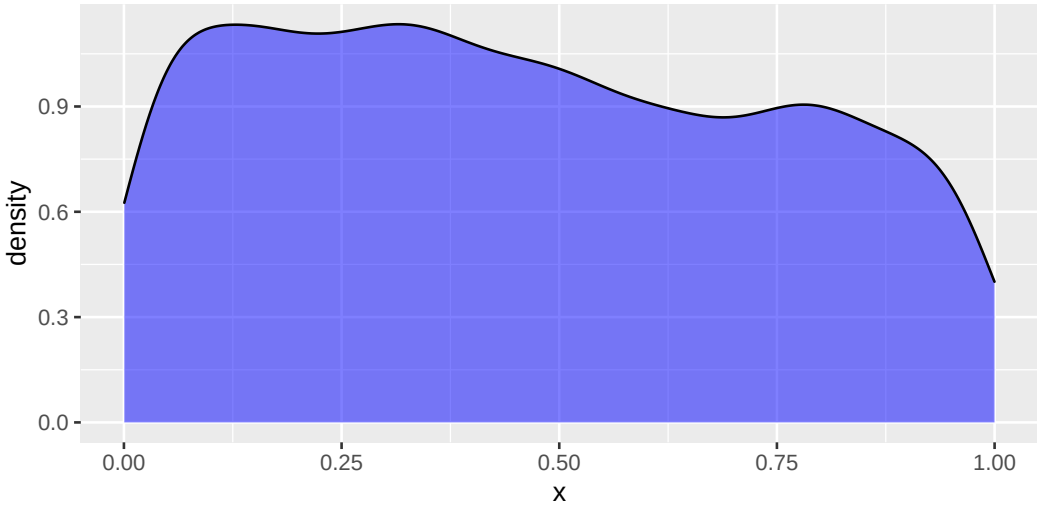
For example, $X \sim N(0,1)$ and $Y = X \mid 0 \leq X \leq 1$. We can sample by drawing from X and only keeping those that fall in $(0,1)$.

```
> x <- rnorm(10000)
> y <- keep(x, 0 < x & x < 1)
> length(y) # number of samples kept

[1] 3390

> mean(y) # use the samples to estimate E(Y)

[1] 0.4621
```


density
0.9
0.6
0.3
0.0
0.00
0.25
0.50
0.75
1.00
x

Computing π

Consider the following method for approximating the value of π:

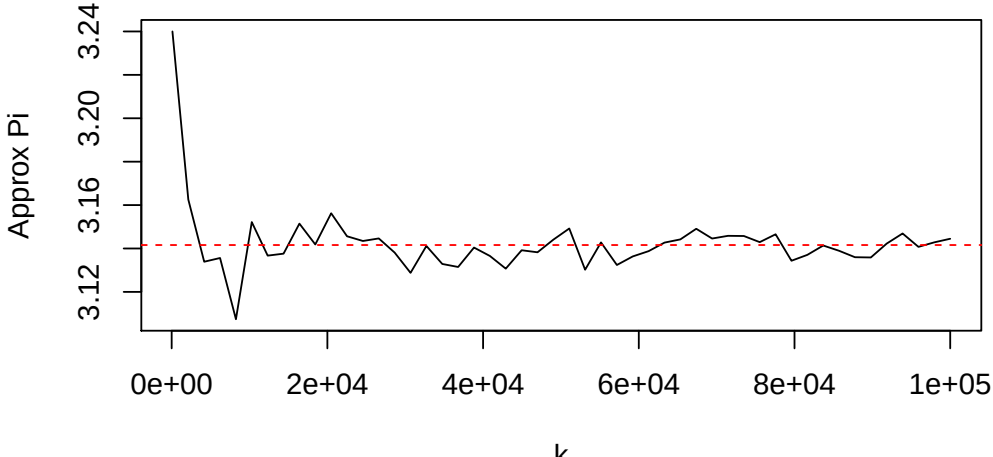
- Initialize $a = 0$
- Generate U_1 and U_2 (independent) from $U(0, 1)$
- If $\sqrt{U_1^2 + U_2^2} \leq 1$ then $a = a + 1$
- Repeat k times.
- Approximate π as

$$\pi = \frac{4a}{k}$$

Calculating in R

```
> approx_pi <- function(k) {
+   u1 <- runif(k)
+   u2 <- runif(k)
+   sqs <- sqrt(u1^2 + u2^2)
+   return(4 * sum(sqs <= 1) / k)
+ }
```

Approximation vs. k

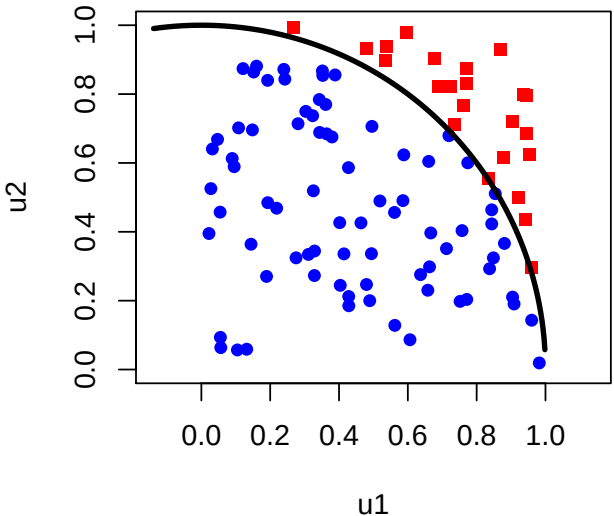


Why does it work?

Recall: the circle centered at $(0, 0)$ and with radius 1, has area π.

We "throw darts" at the upper quadrant of the circle and see how many hit the circle.

This proportion should be about $\pi/4$


u2
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0
u1

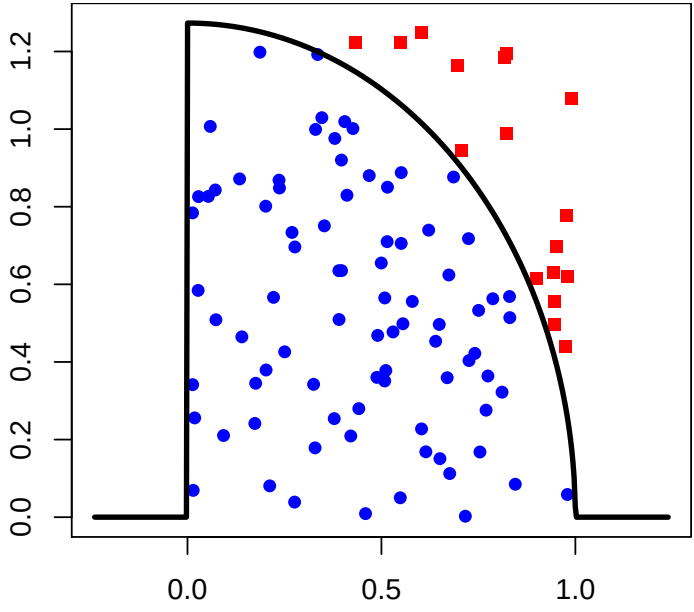
Accept-Reject Method

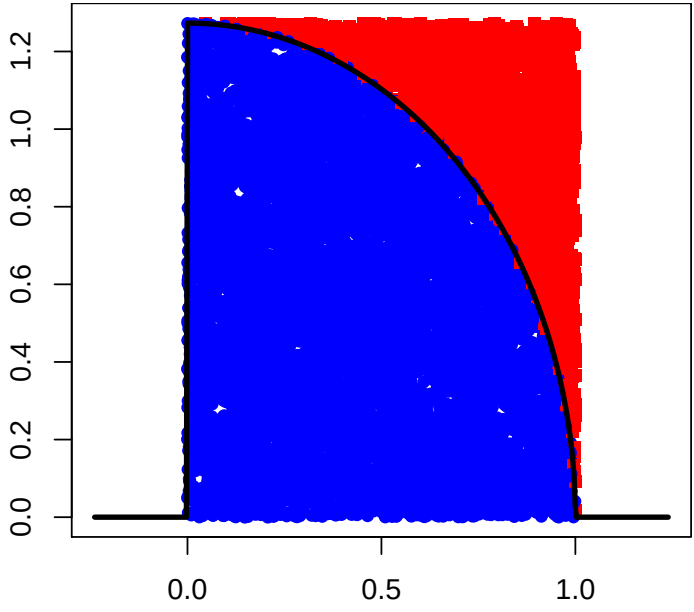
Another interpretation of Approximating π

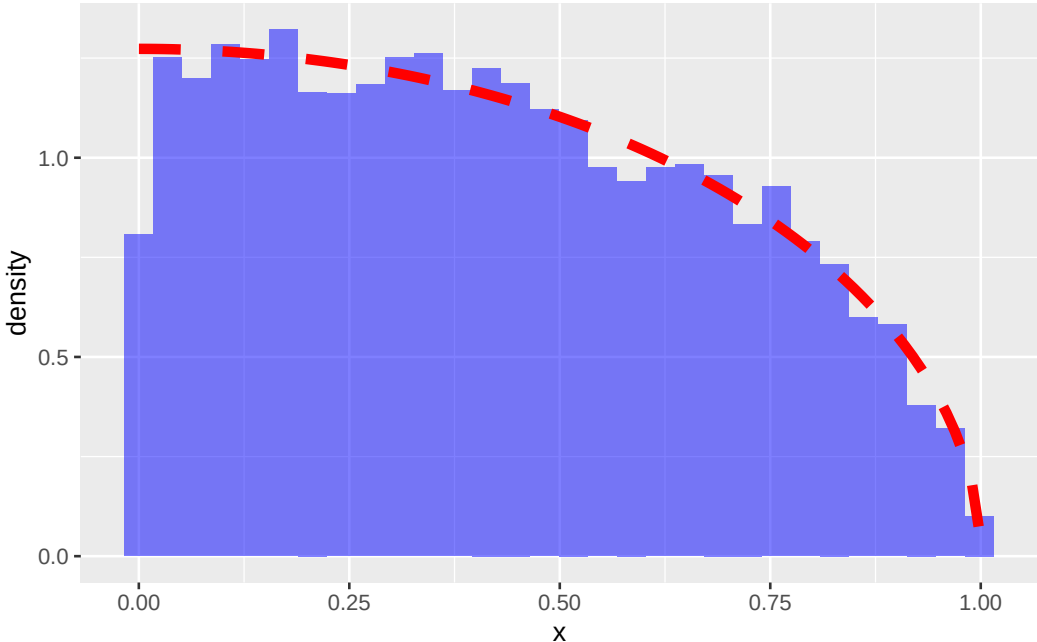
Notice that the following is a valid **probability density function**:

$$f(x) = \frac{4}{\pi}\sqrt{1-x^2}, 0 \leq x \leq 1$$

This curve is just a **scaled version of the unit circle**.


0.0
0.2
0.4
0.6
0.8
1.0
1.2
0.0
0.5
1.0


0.0
0.2
0.4
0.6
0.8
1.0
1.2
0.0
0.5
1.0


density
1.0
0.5
0.0
0.00
0.25
0.50
0.75
1.00
x

Interpretation

- We started by drawing U_1 and U_2, and then keeping U_2 if $U_1^2 + U_2^2 \leq 1$.
- This is equivalent to picking U_2, and keeping U_1 if

$$\frac{4}{\pi} U_2 \leq \frac{4}{\pi} \sqrt{1 - U_1^2} = f(U_1)$$

  or written another way:

$$U_2 \leq \frac{\pi}{4} \frac{f(U_1)}{g(U_1)}, \quad g(x) = 1 \text{ (pdf of } U_1)$$

- This process generated samples from X (the variable with density $f(x)$)!

Accept-Reject in General

Suppose we want to sample from density (or mass) function:

$$f(x)$$

We know $f(x)$, but can't easily sample from it directly (e.g., Normal distribution).

But what if we had **another density** $g(y)$ such that

$$c \times \frac{g(x)}{f(x)} \geq 1$$

for some $c > 0$ and for all x such that $f(x) > 0$.

In other words we need $cg(x)$ to lie above $f(x)$ for any x where $f(x)$ is positive.

Accept-reject for $f(x)$

For the example we just did the **target density** was

$$f(x) = \frac{4}{\pi}\sqrt{1-x^2}, 0 \leq x \leq 1$$

The **candidate density** was the uniform distribution $g(x) = 1, 0 \leq x \leq 1$

The **constant** was

$$c = \frac{4}{\pi}$$

This ensures that:

$$cg(x) = \frac{4}{\pi} \geq \frac{4}{\pi}\sqrt{1-x^2} = f(x)$$

(NB: c is not unique. We could have pick any $c \geq 4/\pi$.)

Example: Uniform and Beta

Suppose we want to sample from the Beta(2,2) distribution which is:

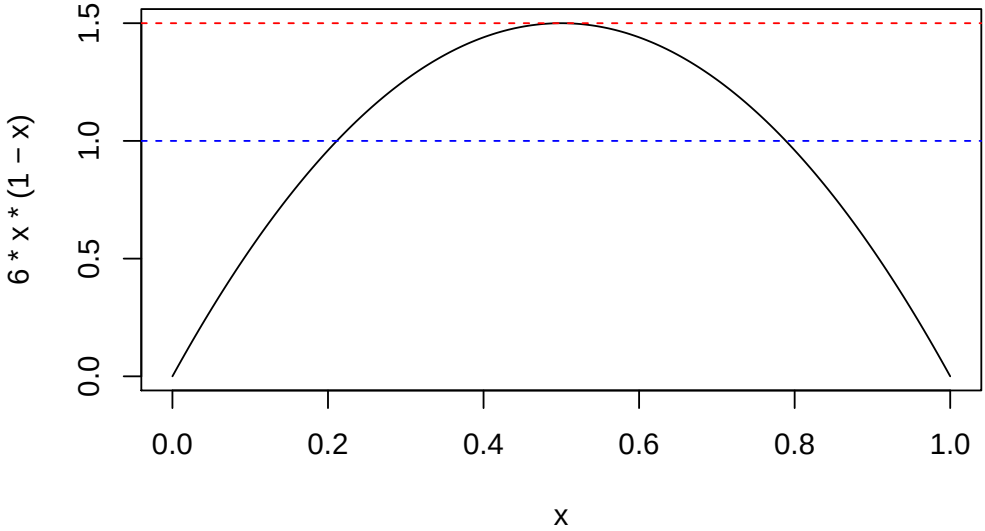
$$f(x) = 6x(1-x), \quad 0 \leq x \leq 1$$

but we only have the standard uniform $g(x) = 1$.

Observe that $f(x)$ achieves its max at $x = 0.5$, the maximum value is $f(0.5) = 6/4$. Therefore we have

$$\frac{6}{4}\frac{g(x)}{f(x)} \geq 1$$

for any point in $[0, 1]$.


6 * x * (1 – x)
0.0
0.5
1.0
1.5
0.0
0.2
0.4
0.6
0.8
1.0
x

Accept-Reject Algorithm

- Draw a Y from $g(y)$
- Draw $U \sim U(0,1)$
- If

$$U < \frac{f(Y)}{cg(Y)}$$

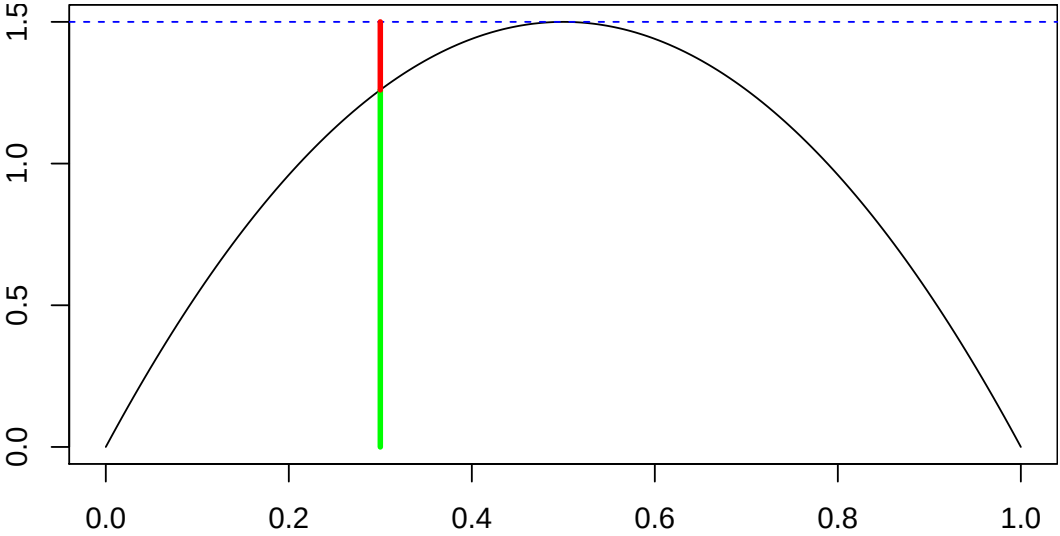
  then **accept** $X = Y$
- Otherwise, **reject** Y as a candidate and repeat.

Example: Beta/Uniform

Say we generate $Y = 0.3$.

- $f(0.3) = 1.26$
- $g(0.3) = 1$
- We will accept if

$$U < \frac{1.26}{(6/4) \times 1} = 0.84$$

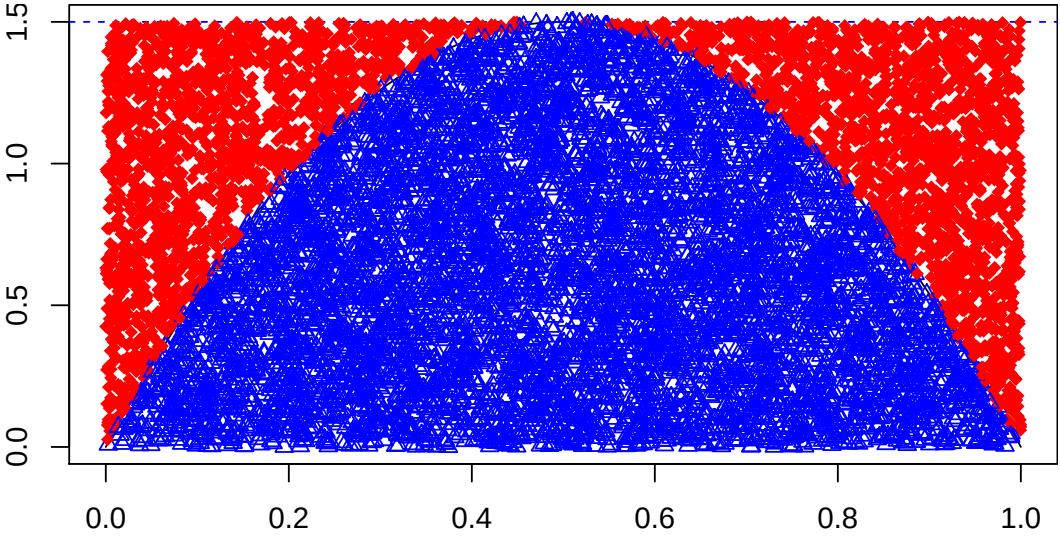

1.5
1.0
0.5
0.0
0.0
0.2
0.4
0.6
0.8
1.0

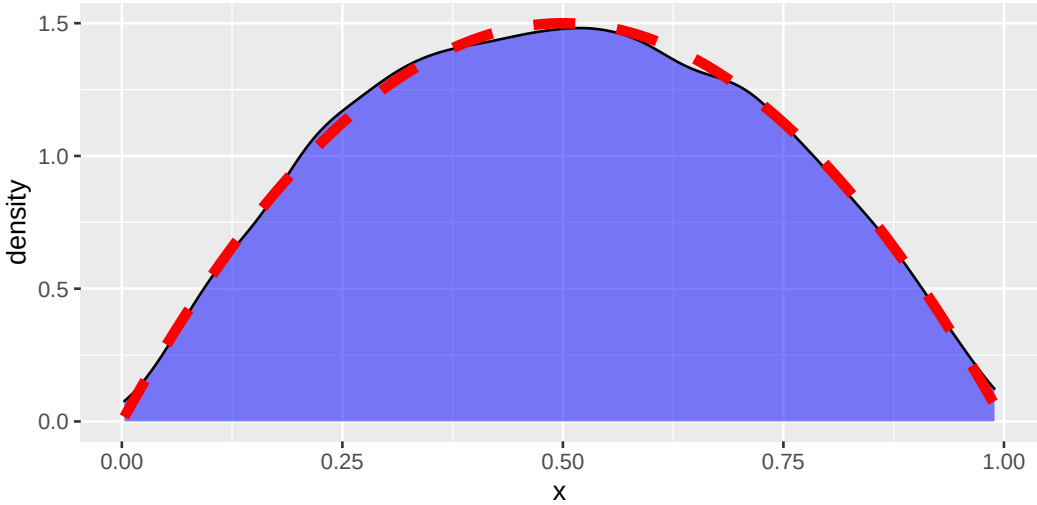
R implementation

```
> k <- 10000
> ys <- runif(k)
> fys <- 6 * ys * (1 - ys)
> gys <- 1
> ratios <- fys / (gys * (6/4))

> us <- runif(k)
> accept <- us < ratios
> accepted <- ys[accept]
> rejected <- ys[!accept] ; mean(!accept)

[1] 0.3377
```


1.5
1.0
0.5
0.0
0.0
0.2
0.4
0.6
0.8
1.0


1.5
1.0
0.5
0.0
density
0.00
0.25
0.50
0.75
1.00
x

Proving the general case

When proving an algorithm generates X we need to show same cumulative distribution function (recall, we did this for inversion method, transformations, etc.).

Let V be the random variable produced by the AR algorithm, we need to show that

$$P(V \leq x) = P(X \leq x)$$

for any x in the support of X.

What kind variable is V?

We know the distribution of Y and U, but what can we say about V?

Notice that V is set to Y, when $U \leq f(Y)/(cg(Y))$, a conditional distribution:

$$V = Y \,\Big|\, U \leq \frac{f(Y)}{c\, g(Y)}$$

For events A and B, recall the definition of conditional probability:

$$P(A|B) = \frac{P(A, B)}{P(B)}$$

Proof

$$P(V \leq x) = P\left(Y \leq x \,\middle|\, U \leq \frac{f(Y)}{c\,g(Y)}\right) = \frac{P\left(Y \leq x, U \leq \frac{f(Y)}{c\,g(Y)}\right)}{P\left(U \leq \frac{f(Y)}{c\,g(Y)}\right)}$$

We'll take the numerator and denominator separately.

In both cases, think of writing $P(\ldots) = E(I(\ldots)$, e.g.:

$$P\left(Y \leq x, U \leq \frac{f(Y)}{c\,g(Y)}\right) = \mathrm{E}\left(I\left(Y \leq x, U \leq \frac{f(Y)}{c\,g(Y)}\right)\right)$$

Numerator

Notice (a) by independence, the joint density is $1 \times g(y)$ and (b) $I(A, B) = I(A)I(B)$.

$$
\begin{aligned}
P\left(Y \le x, U \le \frac{f(Y)}{c\, g(Y)}\right) &= \int_{-\infty}^{\infty} \int_0^1 I(y \le x) I(u \le f(y)/(cg(y)) g(y)\, du\, dy \\
&= \int_{-\infty}^{\infty} I(y \le x) g(y) \left[\int_0^1 I(u \le f(y)/(cg(y))\, du\right] dy \\
&= \int_{-\infty}^{x} g(y) \left[\int_0^{f(y)/(cg(y))} 1\, du\right] dy \\
&= \int_{-\infty}^{x} g(y) \frac{f(y)}{cg(y)}\, dy \\
&= \frac{1}{c} \int_{-\infty}^{x} f(y)\, dy = \frac{1}{c} P(X \le x)
\end{aligned}
$$

Denominator

$$P\left(U \leq \frac{f(Y)}{c\,g(Y)}\right) = \mathsf{E}\left(I\left(U \leq \frac{f(Y)}{c\,g(Y)}\right)\right)$$

By similar logic,

$$\begin{aligned} P\left(U \leq \frac{f(Y)}{c\,g(Y)}\right) &= \int_{-\infty}^{\infty} g(y)\left[\int_0^{f(y)/(cg(y))} 1\,du\right] dy \\ &= \int_{-\infty}^{\infty} g(y)\frac{f(y)}{cg(y)}\,dy \\ &= \int_{-\infty}^{\infty} \frac{1}{c} f(y)\,dy = \frac{1}{c} \end{aligned}$$

Taking the ratio, the $(1/c)$ cancels, so $P(V \leq x) = P(X \leq x)$.

Example: Truncated Normal

We've used **rejection sampling algorithm** for the Truncated Normal before:

$$Y = Z \mid 0 \leq Z \leq 1, Z \sim N(0,1)$$

To use AR, we need the **density function**. To get this, start with the CDF:

$$F_Y(y) = P(Y \leq y) = P(Z \leq y \mid 0 \leq Z \leq 1) = \frac{P(Z \leq y, 0 \leq Z \leq 1)}{P(0 \leq Z \leq 1)}$$

$$= \frac{1}{P(Z \leq 1) - P(Z \leq 0)} \int_0^y \phi(x)\, dx$$

where ϕ is the PDF of the standard Normal distribution.

By definition, the PDF is the derivative of the CDF, so for $0 \leq y \leq 1$,

$$f(y) = \frac{\phi(y)}{P(Z \leq 1) - P(Z \leq 0)}$$

Implementing in R

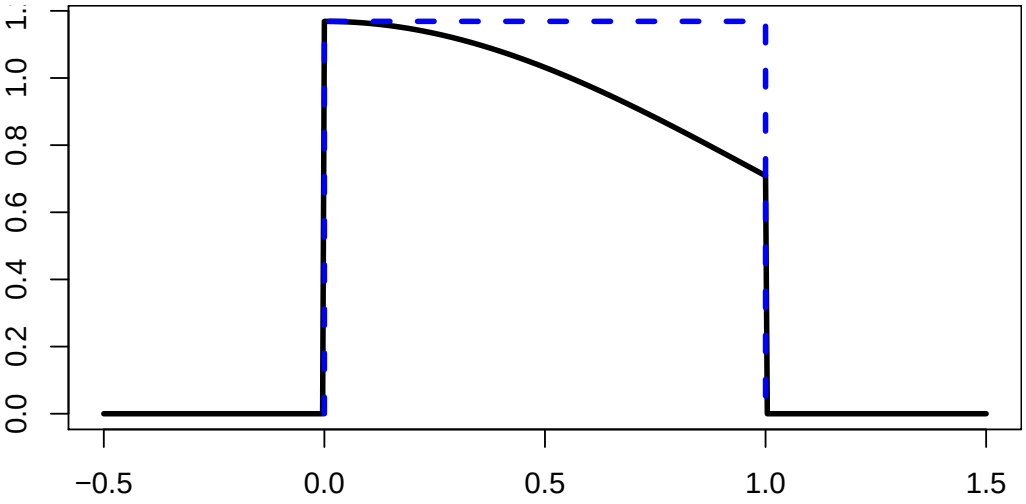
```
> (scaling_const <- pnorm(1) - pnorm(0))
```

```
[1] 0.3413
```

Then we compute the density function as:

```
> truncated <- function(x) {
+   ifelse(x >= 0 & x <= 1,
+          dnorm(x) / scaling_const,
+          0)
+ }
```

Picking a candidate density: Uniform



Implementing

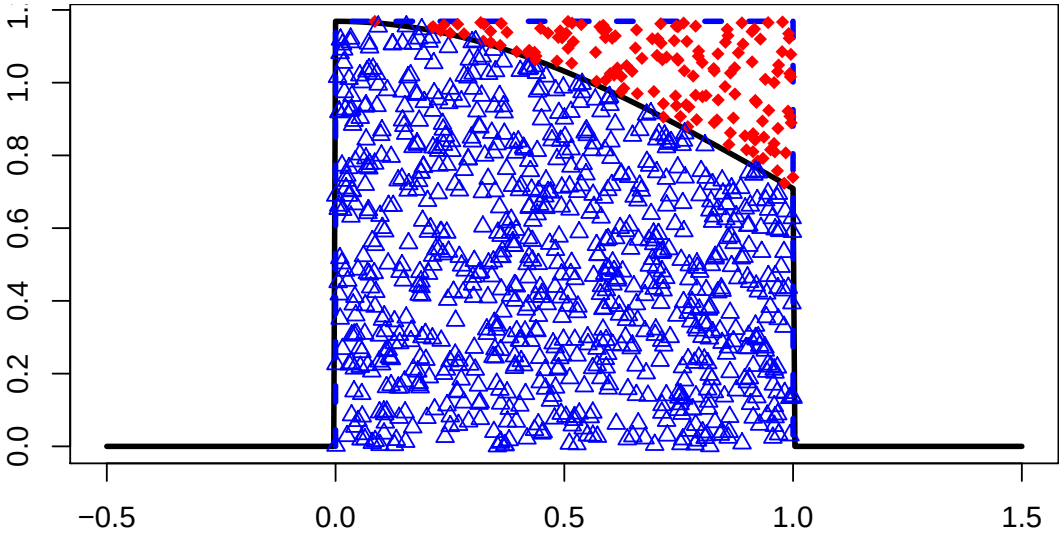
```
> ys <- runif(1000)
> const <- truncated(0)
> ratios <- truncated(ys) / (const * 1) # g(y) = 1

> us <- runif(1000)
> accept_uniform <- us < ratios
> accepted_uniform <- ys[accept_uniform]
> rejected_uniform <- ys[!accept_uniform]
```

Estimating $\mathsf{E}(X)$:

```
> mean(accepted_uniform)

[1] 0.4684
```



We rejected 15.1% of the candidate draws. Can we do better?

Fitting a closer candidate distribution

Consider the density:

$$g(y) = \frac{2}{3}(2 - y), \quad 0 \leq y \leq 1$$

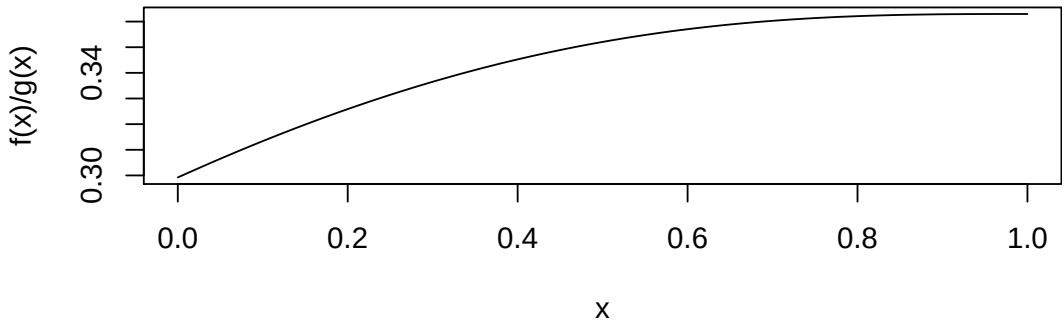
with quantile function

$$Q_y(p) = 2 - \sqrt{4 - 3p}$$

Need to find a c such that

$$c \times \frac{g(x)}{f(x)} \geq 1 \Rightarrow c \geq \frac{f(x)}{g(x)}, 0 \leq x \leq 1$$

Notice that the **ratio $f(x)/g(x)$ is increasing**:

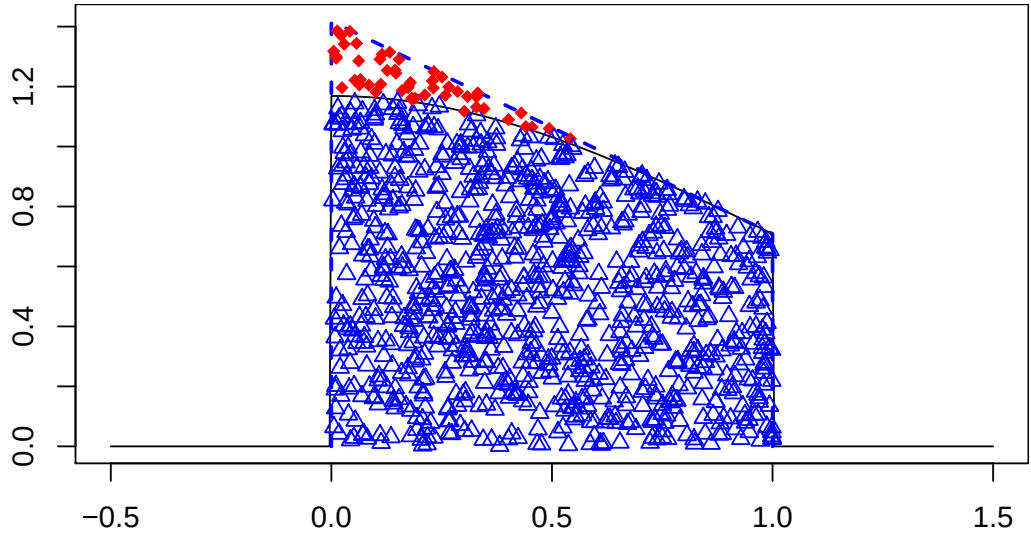


Then the maximum ratio is at $x = 1$.

$$c = f(1)/g(1) = 0.363 \Rightarrow \frac{cg(x)}{f(x)} \geq 1, 0 \leq x \leq 1$$

```
> g <- function(y) { (2/3) * (2 - y)}
> qg <- function(p) { 2 - sqrt(4 - 3 * p)}
> ys <- qg(runif(1000))
> const <- truncated(1) / g(1)
> ratios <- truncated(ys) / (const * g(ys))

> us <- runif(1000)
> accept_g <- us < ratios
> accepted_g <- ys[accept_g]
> rejected_g <- ys[!accept_g]
```



Now we reject only 5.1% of the candidate draws.

Comparing Efficiency

Recall, we define the **efficiency of an estimator** as the **variance of the sampling distribution** for that estimator.

For both the uniform and tuned candidates, the variance is given by:

$$\frac{1}{n}\text{Var}(Y)$$

. The $\text{Var}(Y)$ term is the same in both cases, so when calculating the **relative efficiency** we only consider the resulting (non-rejected) sample size:

```
> sum(accept_g) / sum(accept_uniform)
```

```
[1] 1.118
```

Interpretation: the method using g is 10% more efficient than the uniform method.

A bimodal density

Suppose we have a density of the form:

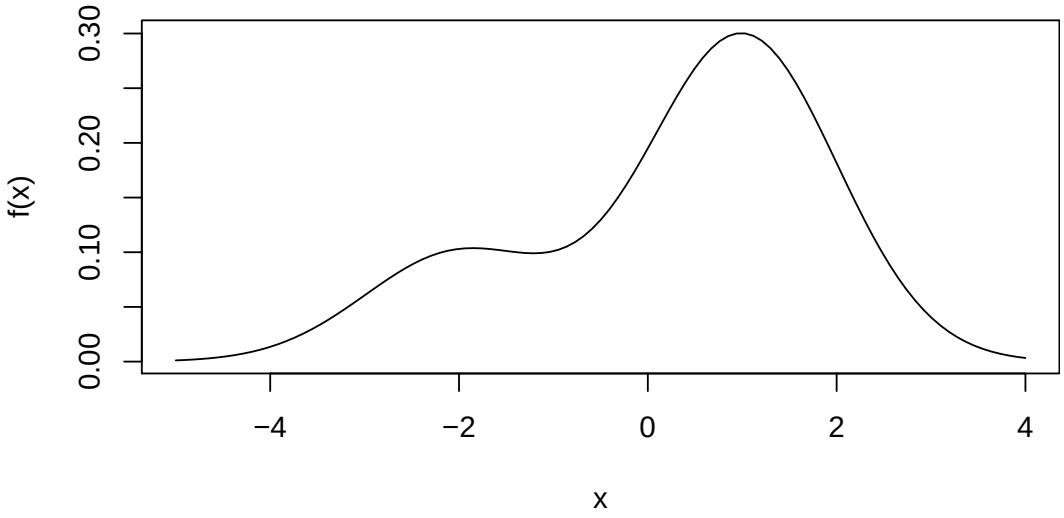
$$f(x) = 0.25\phi(x+2) + 0.75\phi(x-1), -\infty < x < \infty$$

where ϕ is the standard Normal PDF.

```
> f <- function(x) { 0.25 * dnorm(x + 2) + 0.75 * dnorm(x - 1)}
```

A density like is this called **bimodal** because it contains two local maxima.

Density of f



What candidate distribution?

Our initial thought might be to use $\phi(x)$ (standard Normal) as the candidate.

The only problem with that is that

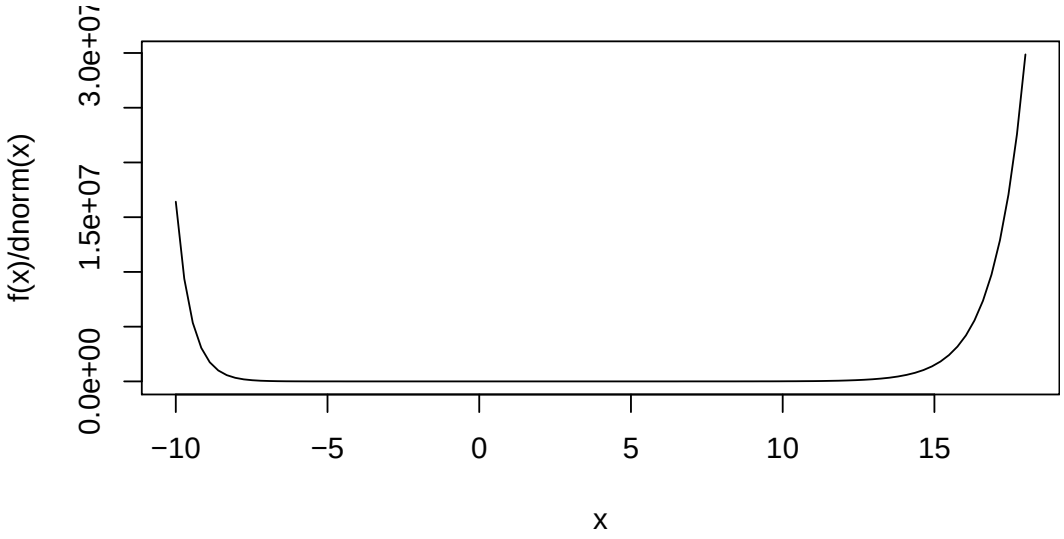
$$\lim_{x \to \pm\infty} f(x)/\phi(x) = \infty$$

(i.e., no c exists that uniformly bounds f).

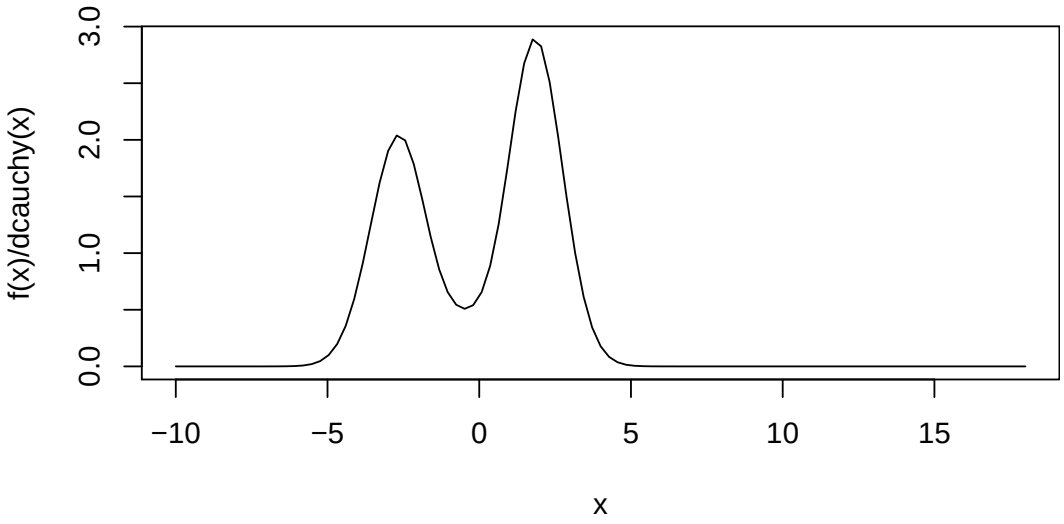
We saw that the Cauchy distribution has **fat tails**, perhaps that would be useful?

$$g(x) = \frac{1}{\pi(1+x)^2}$$

Ratio of $f(x)/\phi(x)$



Ratio of $f(x)/g(x)$

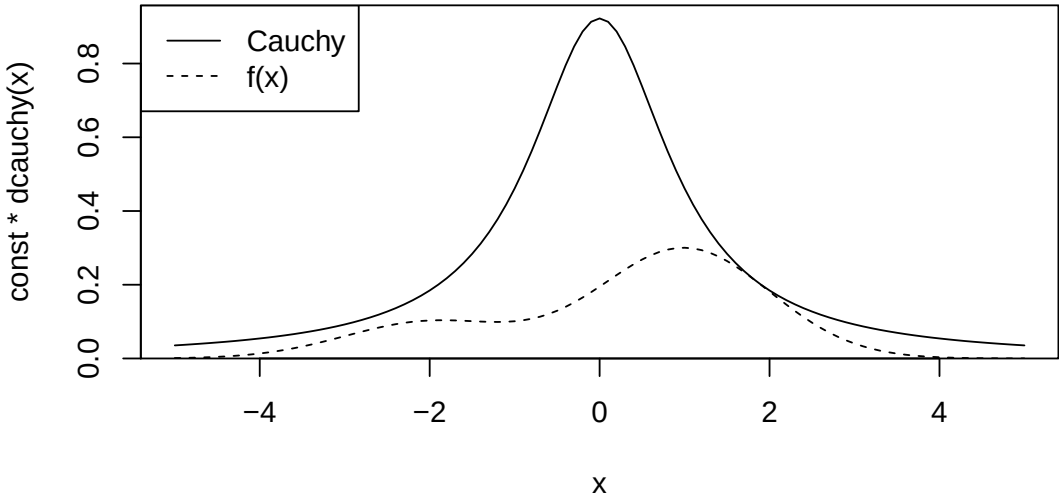


Lazy mode: find c by evaluation

```
> h <- function(x) { f(x) / dcauchy(x)}
> xs <- seq(-4, 4, length.out = 1000)
> (const <- max(h(xs)))

[1] 2.898
```

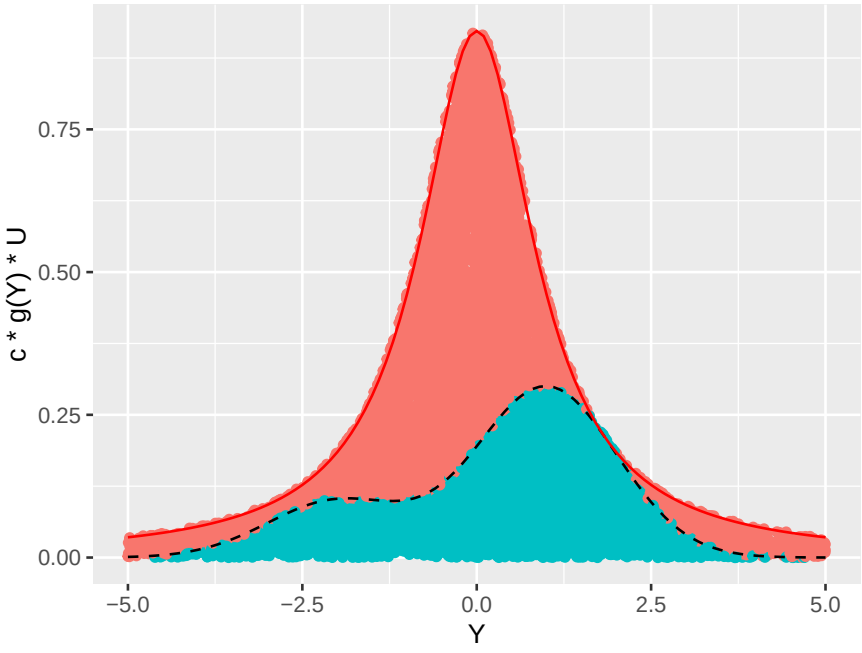
Plotting Distributions



Putting it together

```
> k <- 10000
> ys <- rcauchy(k)
> ratios <- f(ys) / (const * dcauchy(ys))
> us <- runif(k)
> accept <- us < ratios
> x <- ys[accept] ; mean(accept)

[1] 0.3526
```


0.75
0.50
0.25
0.00
c * g(Y) * U
−5.0
−2.5
0.0
2.5
5.0
Y
accept
FALSE
TRUE

What if we got c wrong?

We found $c = 2.898$. What if we were wrong? How would the distribution change? Too large, too small?

```
> wrong_consts <- c(2 *  const,
+                   const,
+                   0.75 * const,
+                   0.25 * const,
+                   0.1 * const,
+                   0)
```

Functions to handle AR

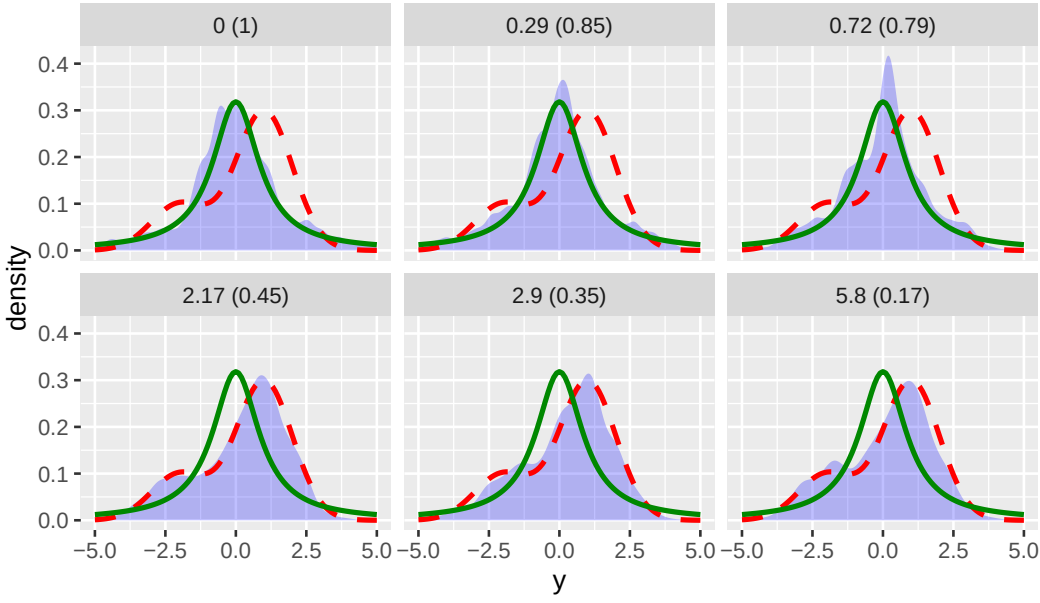
For numerical reasons, we'll rewrite $U \leq f(Y)/(cg(Y))$ as

$$c\, g(Y)\, U \leq f(Y)$$

```
> accept_reject <- function(n, f, g, rg, const) {
+     ys <- rg(n)
+     us <- runif(k)
+     accept <- us * const * g(ys) <= f(ys)
+     data.frame(y = ys, accept = accept)
+ }
```

Using the bad constants

```
> bimodal <- function(const) {
+     accept_reject(1000, f, dcauchy, rcauchy, const)
+ }
> wrong <- map_dfr(wrong_consts, .id = "constant", bimodal)
> wrong$constant <- wrong_consts[as.numeric(wrong$constant)] # get the orig
```


0 (1)
0.29 (0.85)
0.72 (0.79)
2.17 (0.45)
2.9 (0.35)
5.8 (0.17)
density
0.0
0.1
0.2
0.3
0.4
−5.0
−2.5
0.0
2.5
5.0
y

Accept-Reject Summary

- For a density f, use a density g for the sampling.
- Draw uniform values to decided to accept or reject draws for g.
- Need to figure out the scaling constant c to make $cg(y)/f(y) \geq 1$
- Generally, we accept about $100 \times (1/c)\%$ for the candidates, so making c small is useful.
- Picking c too large is ok, too small leads to draws from g.