ex1

Nikolai German

2025-05-03

1.1 Random Variable Generation

a) Implementation of Lehmer RNG

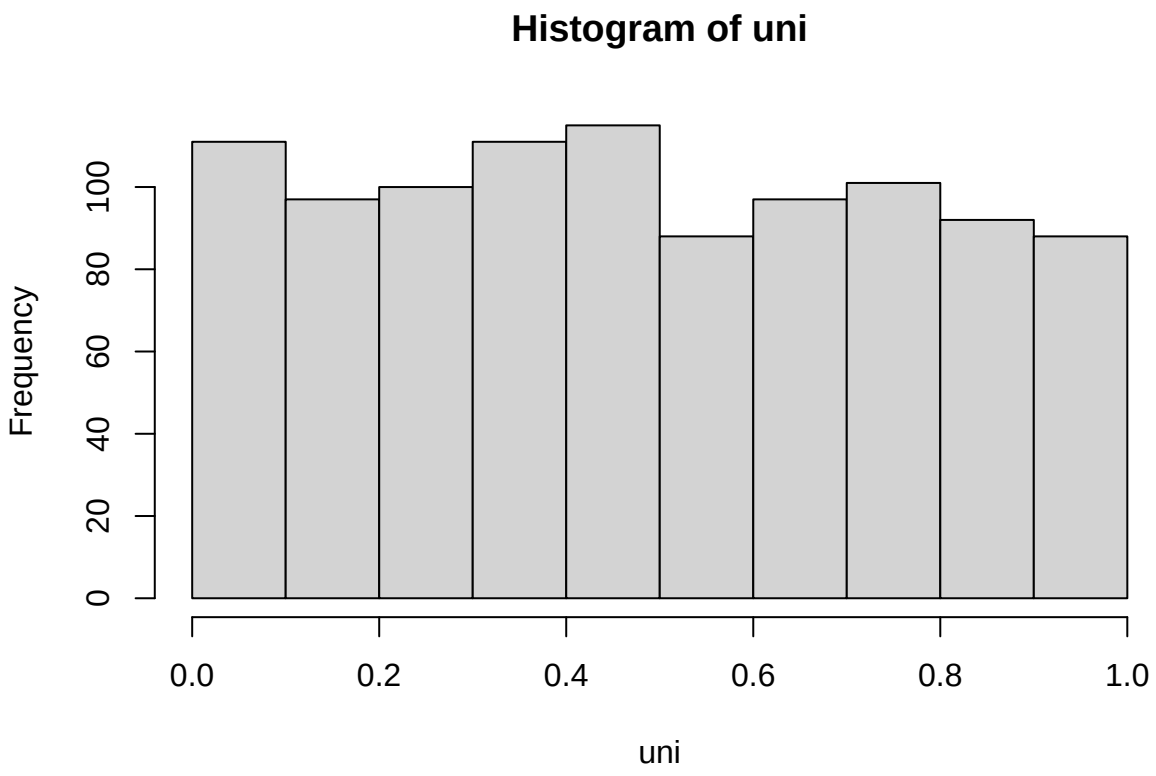
We implement the Lehmer RNG:

$$x_n = (ax_{n-1}) \bmod m$$

The seed x_0 has to be chosen beforehand, $m = 2^31 - 1$, $a = 7^5$

```
LRNG <- function(n, seed = 42) {
  m <- 2^31 -1 # .Machine$integer.max
  a <- 7^5
  samp <- numeric(n)
  samp[[1]] <- (a * seed) %% m
  for (i in seq_len(n - 1)) {
    samp[[i + 1]] <- (a * samp[[i]]) %% m
  }
  return(samp / m)
}
```

We simulate $n = 1000$ samples and plot as a histogram:



b) Implement inverse transformation method

```
ITM <- function(inverse, n, seed = 42) {
  uni <- LRNG(n = n, seed = seed)
  inverse(uni)
}
```

We obtain the inverse:

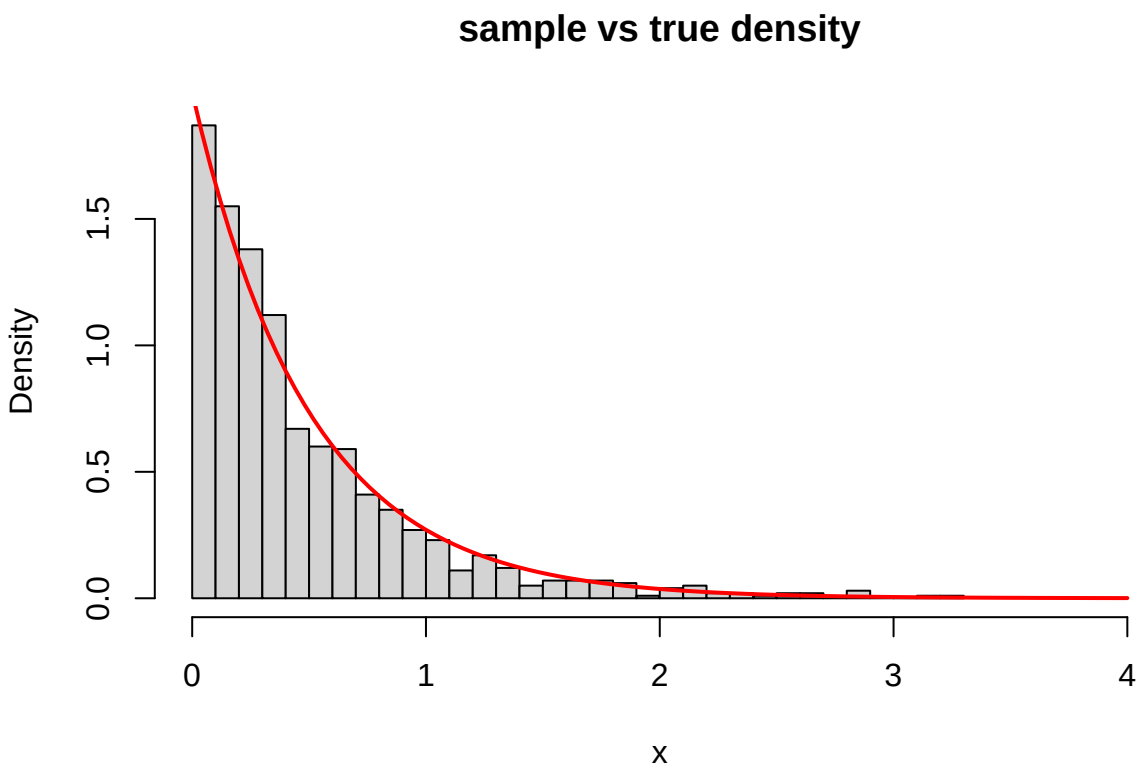
$$F(x) = 1 - e^{-2x} \tag{1}$$

$$\implies F^{-1}(q) = -0.5 * log(1 - q) \tag{2}$$

Subsequently we call ITM on the inverse. Additionally we implement the true density $f_X(x) = 2e^{-2x}$.

```
Fq <- function(q) {
  -.5 * log(1 - q)
}

fx_samp <- ITM(Fq, 1000)
fx_true <- function(x) {2*exp(-2*x)}
```

We plot the true density vs. the simulated samples:



c) Goodness of fit testing

Kolmogorov-Smirnov-Test

```
# ?ks.test
ks.test(fx_samp, "pexp", rate = 2)
```

```
##
##  Asymptotic one-sample Kolmogorov-Smirnov test
##
## data:  fx_samp
## D = 0.044486, p-value = 0.0382
## alternative hypothesis: two-sided
```

$\mathcal{X}^2$-Test

```
chisq <- function(samp, bins = 10, pdf, ...) {
  tab <- table(cut(c(0, samp), breaks = bins)[2:(length(fx_samp)+1)])
  step <- max(samp) / 10
  p <- vapply(seq_len(bins),
              function(x) pdf(x*step, ...) - pdf((x - 1)*step, ...),
              numeric(1))
  chisq.test(tab, p = p, rescale.p = TRUE)
}
```

```
x2 <- chisq(fx_samp, pdf = pexp, rate = 2)
```

```
## Warning in chisq.test(tab, p = p, rescale.p = TRUE): Chi-squared approximation
## may be incorrect
```

```
x2$observed
```

```
##
## (-0.00322,0.322]    (0.322,0.645]    (0.645,0.967]     (0.967,1.29]
##              512              230              133               56
##      (1.29,1.61]      (1.61,1.93]      (1.93,2.26]      (2.26,2.58]
##               26               20               10                6
##       (2.58,2.9]       (2.9,3.23]
##                5                2
```

```
x2$expected
```

```
## (-0.00322,0.322]    (0.322,0.645]    (0.645,0.967]     (0.967,1.29]
##       475.881025       249.777651       131.101833        68.811964
##      (1.29,1.61]      (1.61,1.93]      (1.93,2.26]      (2.26,2.58]
##        36.117621        18.957206         9.950147         5.222575
##       (2.58,2.9]       (2.9,3.23]
##         2.741195         1.438782
```

here something smart about what we have simulated...

1.2 Rejection Sampling, Importance Sampling

a) Rejection Sampler Implementation

The maximum of $exp(-x^2)$ is located at $x_0 = 0$, with a value of $exp(0) = 1$, therefore we need to scale our umbrella-distribuation to be ≥ 1.

Since the density of the Uniform-distribution an $[-a, a]$ is $\frac{1}{2a}$, we search for α such that $\alpha \cdot u \geq 1 \implies \alpha \geq 2a$, where $u \sim U(-a, a)$.

```
rejection_sampler <- function(n_samples, a, verbose = FALSE, print.steps = FALSE) {
  checkmate::assertIntegerish(n_samples, lower = 1, len = 1, any.missing = FALSE)
  checkmate::assertNumeric(a, lower = .Machine$double.eps, len = 1, any.missing = FALSE)
  checkmate::assertFlag(verbose)
  checkmate::assertFlag(print.steps)
  ###
  alpha <- 2*a + .Machine$double.eps
  f <- function(x) exp(-x^2) * (x <= a) * (x >= -a)
  g <- function(x) (1 / 2*a) * (x <= a) * (x >= -a)
  G_inv <- function(q) 2*a*q - a
  ###
  samp <- numeric(n_samples)
  start <- 1
  step <- 1
  while(n_samples > 0) {
    Y <- G_inv(runif(n_samples))
    U <- runif(n_samples)
    test <- U <= f(Y) / (alpha * g(Y))
    approved <- sum(test)
    if(verbose) {
      cat(sprintf("step %d: approved %d of %d remaining samples...\n",
                  step, approved, n_samples))
    }
    step <- step + 1
    if (approved > 0) {
```

```
      samp[start:(start + approved - 1)] <- Y[test]
      start <- start + approved
      n_samples <- n_samples - approved
    }
  }
  if(print.steps) {
    cat(sprintf("##############\n number of necessary steps: %d \n##############\n",
                step))
  }
  return(structure(samp, steps = step)
  )
}

head(rejection_sampler(n = 1000, a = 1, verbose = TRUE))
```

```
## step 1: approved 732 of 1000 remaining samples...
## step 2: approved 199 of 268 remaining samples...
## step 3: approved 52 of 69 remaining samples...
## step 4: approved 11 of 17 remaining samples...
## step 5: approved 4 of 6 remaining samples...
## step 6: approved 2 of 2 remaining samples...
```

```
## [1]  0.8798760  0.2850535 -0.2363907 -0.1297534 -0.3387811 -0.1763324
```

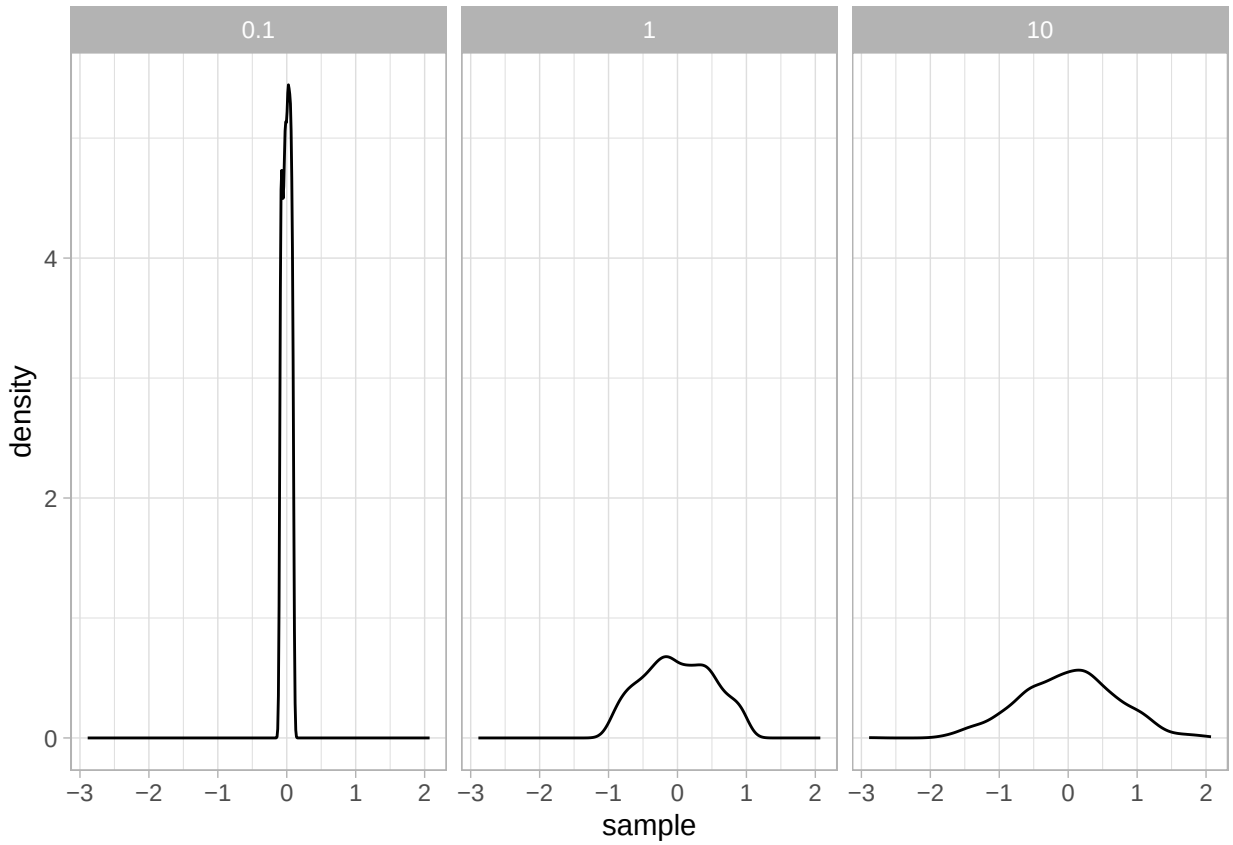
```
attr(rejection_sampler(n = 1000, a = 1), "steps")
```

```
## [1] 8
```

b) Sampling for different values of a

We sample $n = 1000$ with $a \in \{0.1, 1, 10\}$

```
tibble("0.1" = rejection_sampler(1000, .1),
       "1" = rejection_sampler(1000, 1),
       "10" = rejection_sampler(1000, 10)) %>%
  pivot_longer(everything(), names_to = "a", values_to = "sample") %>%
  mutate(a = as.factor(as.numeric(a))) %>%
  ggplot(aes(sample)) +
  geom_density() +
  facet_wrap(~a, ncol = 3) +
  theme_light()
```

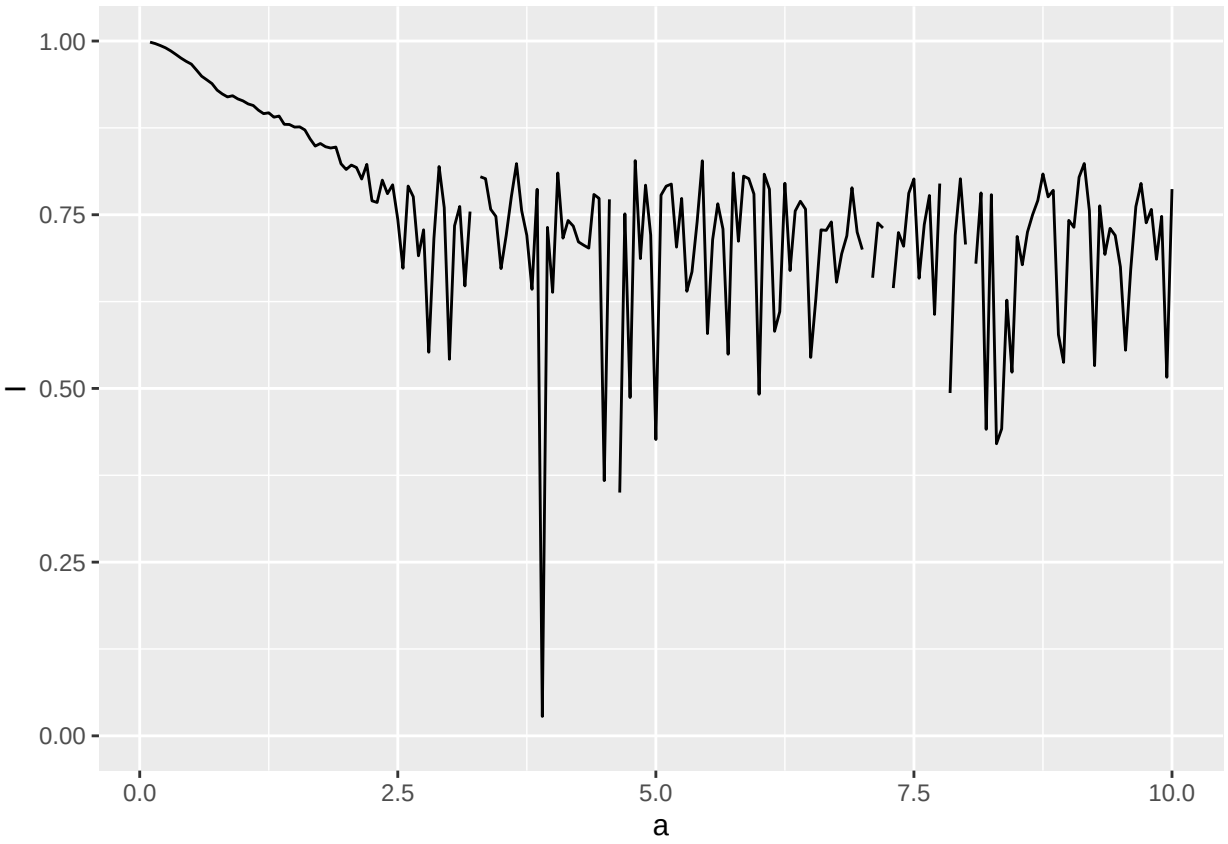


something unsefull here!

c) Importance Sampling

We want to estimate $I_a = \int_{-a}^{a} \frac{cos(x)}{1+x^2} dx$. Upon multiplying with $\frac{g(x)}{g(x)} = \frac{exp(-x^2)}{exp(-x^2)}$ we obtain $I_a = \mathbb{E}_G(h(X)\frac{f(X)}{g(X)}) = \int_{-a}^{a} \frac{1}{1+x^2} \frac{cos(x)}{exp(-x^2)} exp(-x^2) dx$.

```
importance_sampler <- function(n = 1000, a = 1) {
  f <- function(x) cos(x)
  g <- function(x) exp(-x^2)
  h <- function(x) 1 / (1 + x^2)
  Y <- rejection_sampler(n_samples = n, a = a)
  mean(h(Y) * f(Y) / g(Y))
}
```



d) Computing standard error

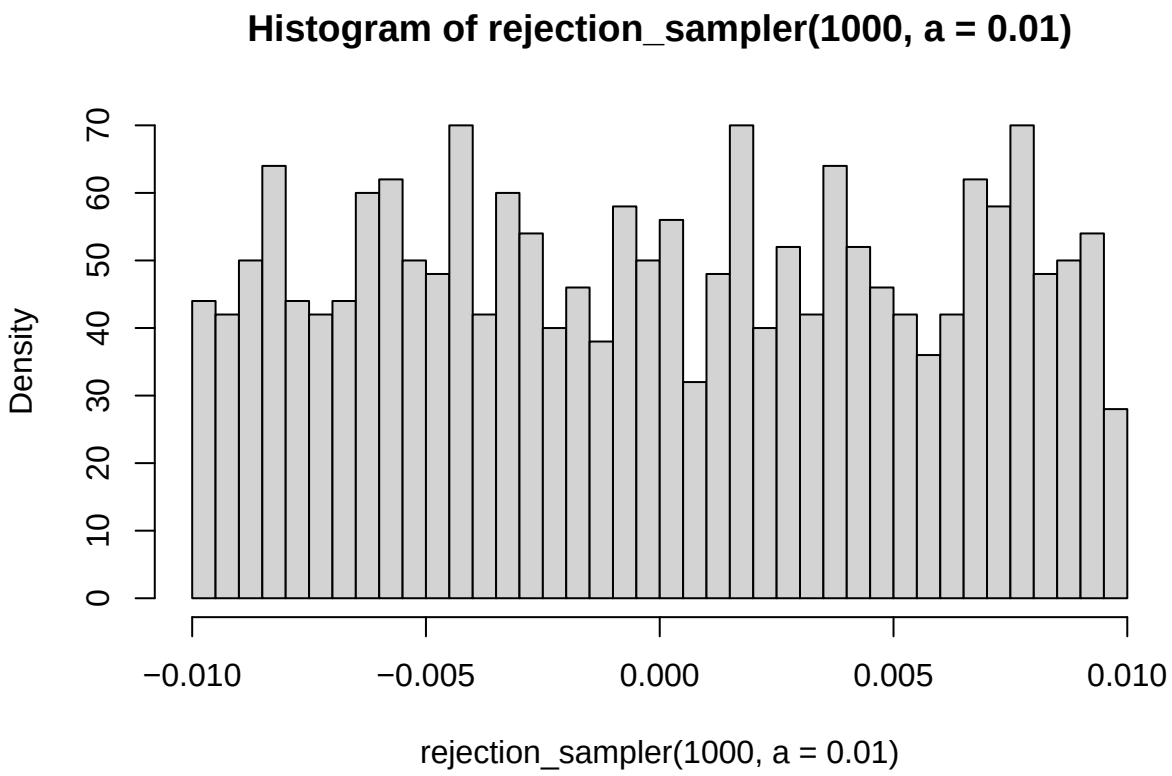
```
integrand <- function(x) cos(x)/(1 + x^2)

se_est <- function(a) {
  num <- integrate(integrand, -a, a)
  est <- importance_sampler(a = a)

}

integrate(integrand, -3, 3)
```

```
## 1.248808 with absolute error < 7.5e-08
```

```
importance_sampler(10000000, a = 3)
```

```
## [1] 0.704772
```

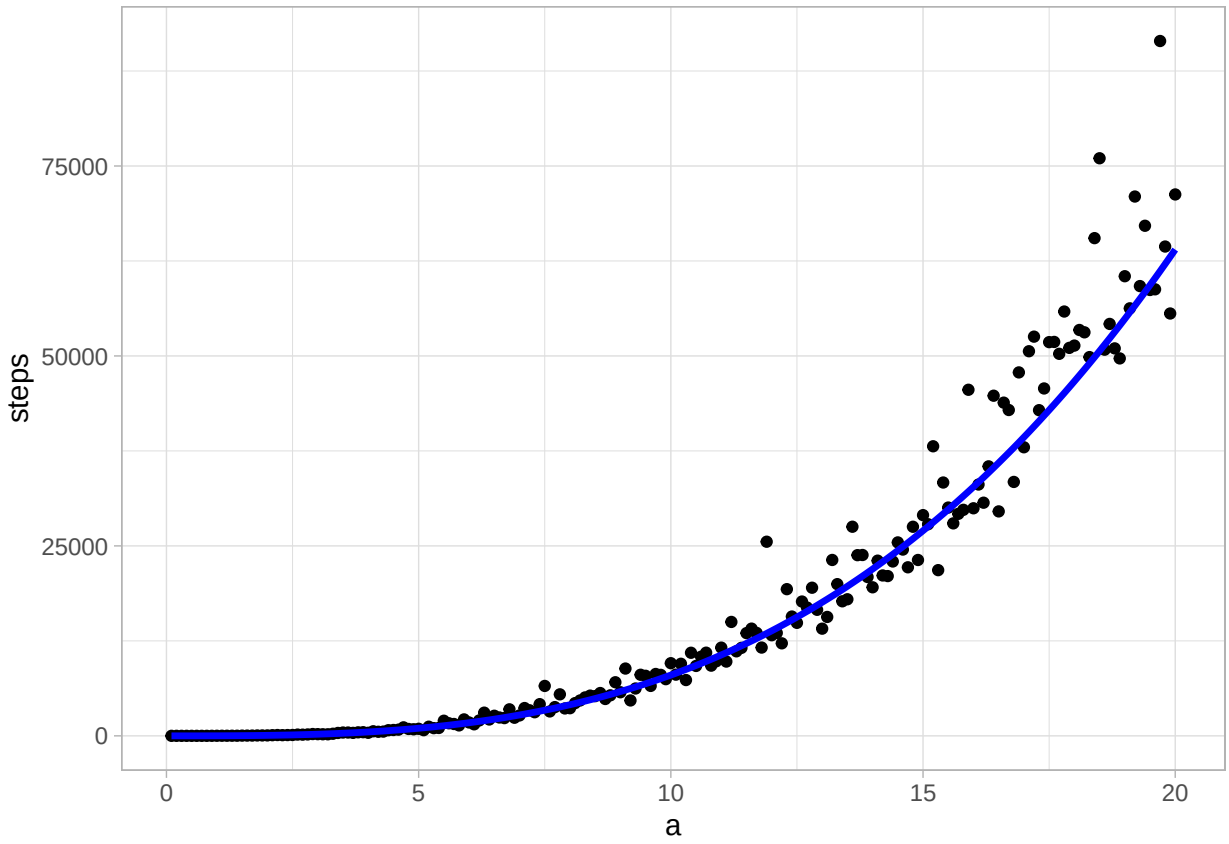
```
hist(rejection_sampler(1000, a = 0.01), probability = TRUE, breaks = 50)
```



e) Can this method be used for $a \to \infty$?

By using very large values for a, the amount of rejected samples is increasing sharply.

The amount of steps needed to compute the rejection sample for incresing values of a are shown below. The number of steps can be interpreted as complexity of the operation. It can be visually approximated that $steps = \mathcal{O}(a^3), a \in [0, 20]$


steps
75000
50000
25000
0
0
5
10
15
20
a