

Problemset 1

Nikolai German (12712506)

2025-05-06

1.1 Random Variable Generation

a) Implementation of Lehmer RNG

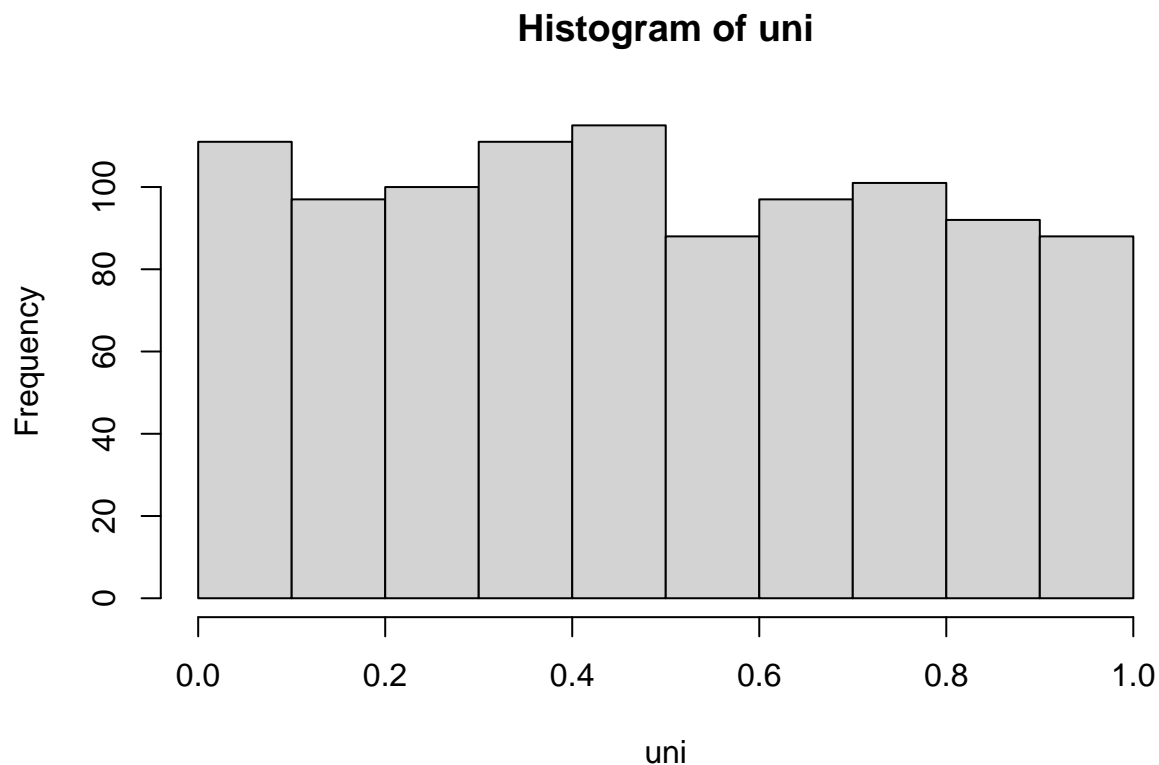
We implement the Lehmer RNG:

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

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:

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

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

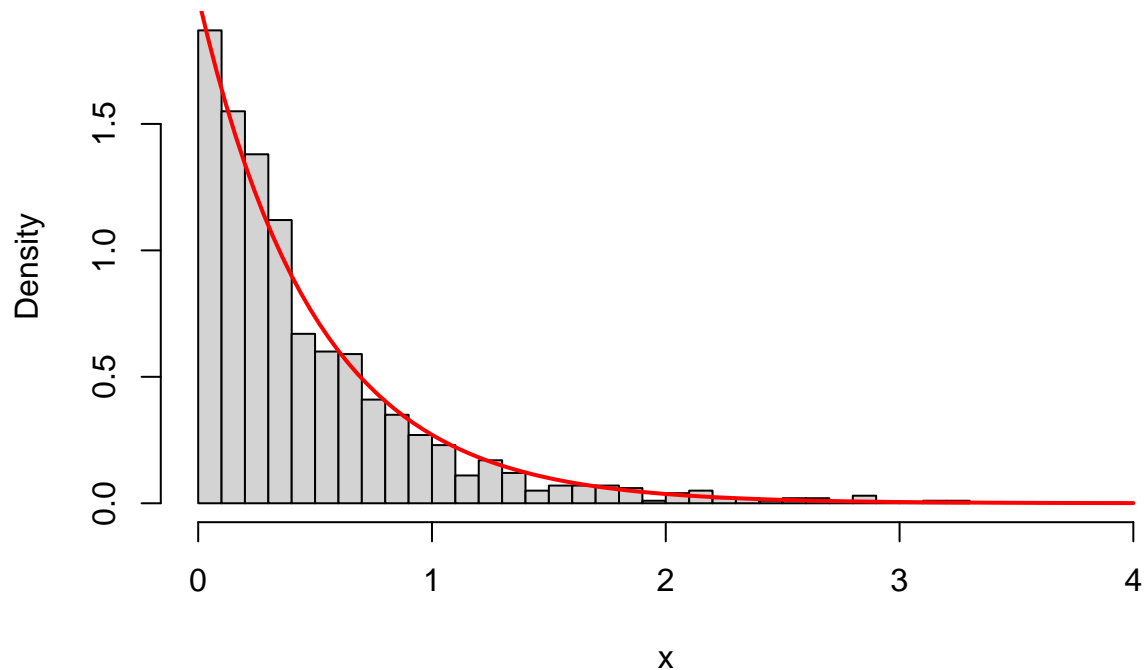
Subsequently we call ITM on the inverse. Additionally we implement the true density $f_X(x) = 2e^{-2x}$.

```
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:

sample vs true density



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
```

χ^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)
x2$p.value
```

```
## [1] 0.2243432
```

```
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
```

Since the χ^2 -Test uses binning, it may be able to smooth out little deviations from the expectation. We would expect a higher p-value for the Kolmogorov-Smirnov-Test, if we would have simulated more samples.

```
ks.test(ITM(Fq, 100000), "pexp", rate = 2)
```

```
##
## Asymptotic one-sample Kolmogorov-Smirnov test
##
## data: ITM(Fq, 1e+05)
## D = 0.0033039, p-value = 0.225
## alternative hypothesis: two-sided
```

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-distribution to be ≥ 1 .

Since the density of the Uniform-distribution on $[-a, a]$ 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 743 of 1000 remaining samples...
## step 2: approved 189 of 257 remaining samples...
## step 3: approved 51 of 68 remaining samples...
## step 4: approved 15 of 17 remaining samples...
## step 5: approved 2 of 2 remaining samples...

## [1] -0.53438862  0.06566327  0.36551234  0.32704624 -0.91516449 -0.48822289

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

## [1] 6

```

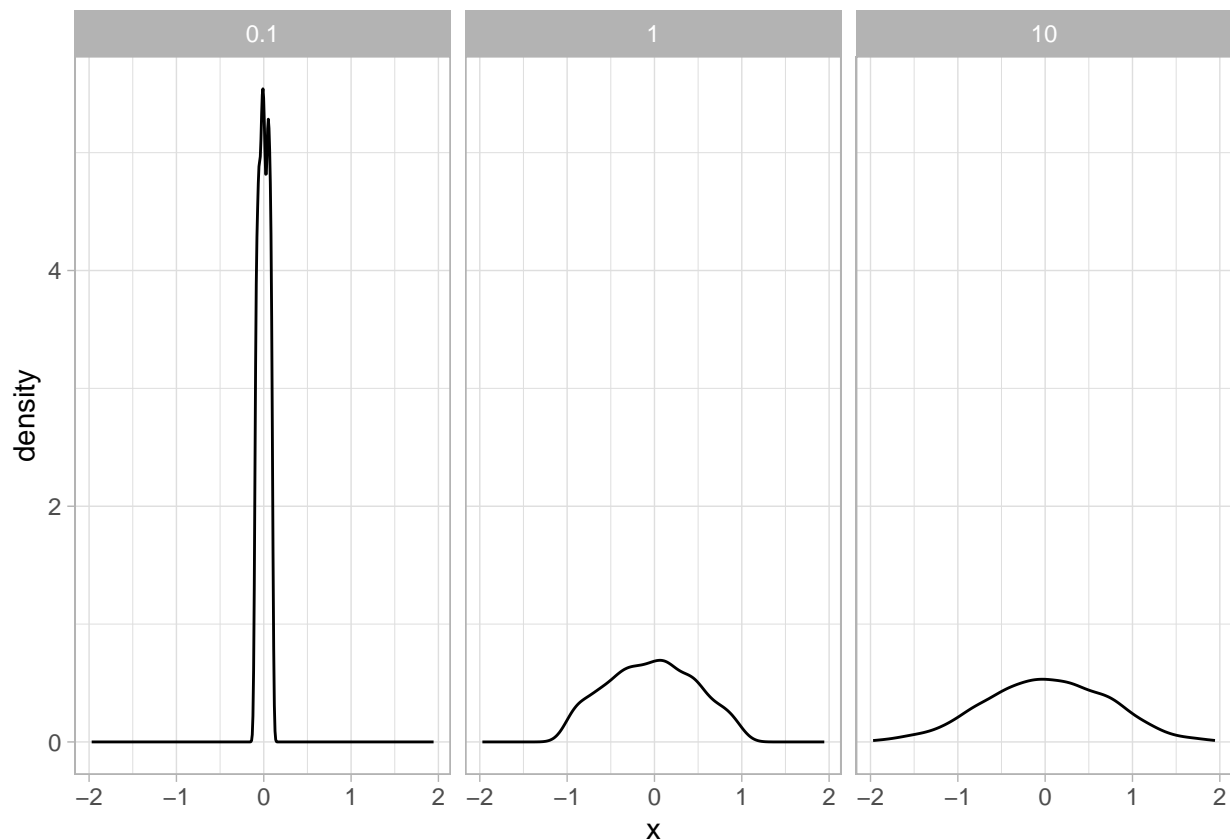
b) Sampling for different values of a

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

```

tibble(a = c(.1, 1, 10)) %>%
  mutate(x = lapply(a, function(a) rejection_sampler(1000, a))) %>%
  unnest_longer(x) %>%
  ggplot(aes(x)) +
  geom_density() +
  facet_wrap(~a, ncol = 3) +
  theme_light()

```



With bigger a , we see that we have a bigger variance in the cdf.

c) Importance Sampling

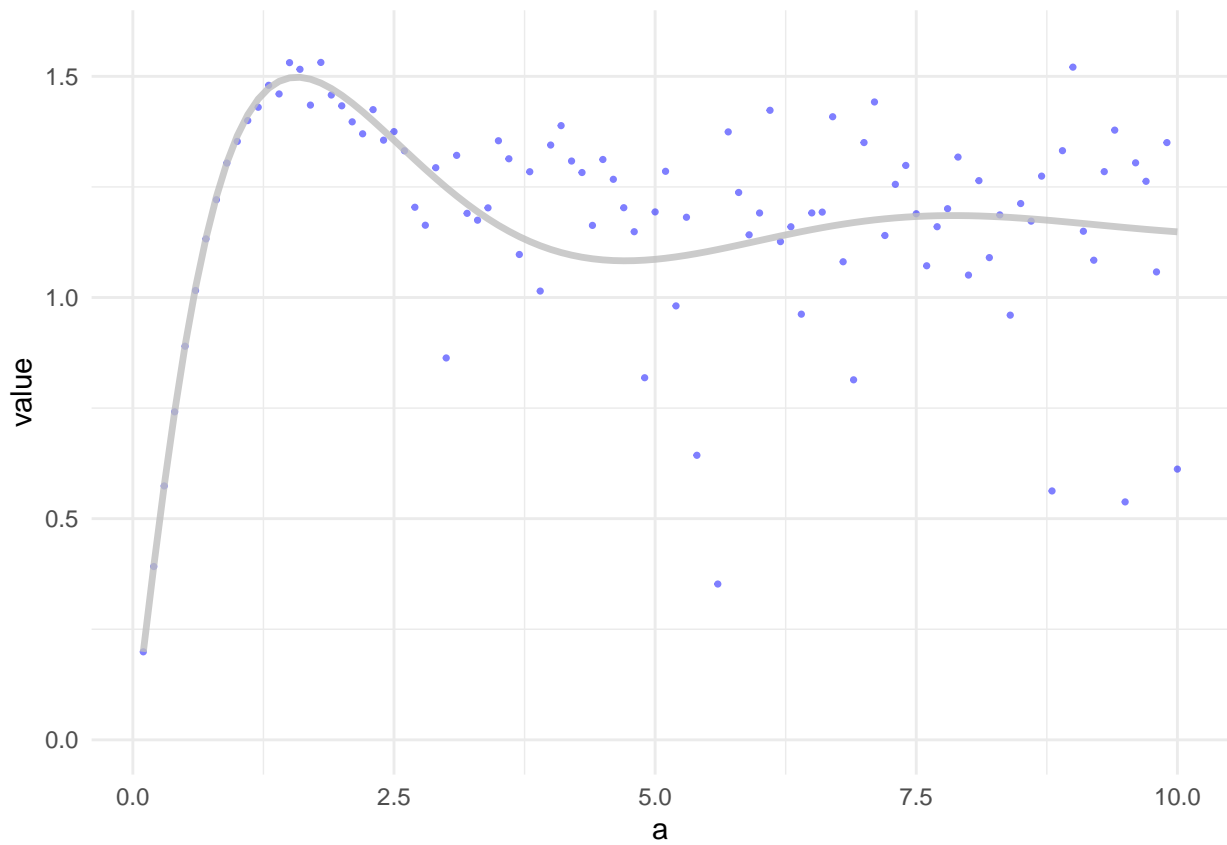
We want to estimate $I_a = \int_{-a}^a \frac{\cos(x)}{1+x^2} dx$. Upon multiplying with $\frac{g(x)}{g(x)} = \frac{\exp(-x^2)}{\exp(-x^2)}$ we obtain $I_a = \mathbb{E}_G(h(X) \frac{f(X)}{g(X)}) = \int_{-a}^a \frac{1}{1+x^2} \frac{\cos(x)}{\exp(-x^2)} \exp(-x^2) dx$.

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

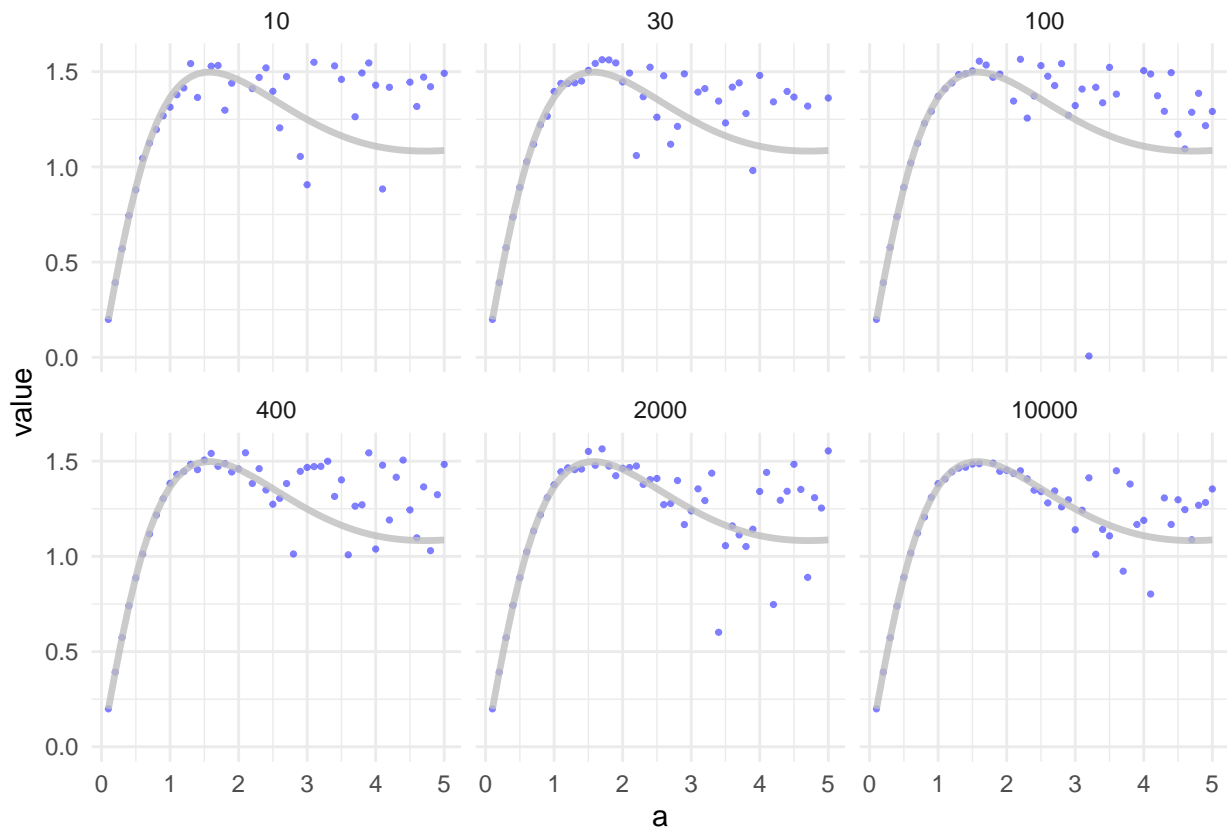
importance_sampler <- function(n = 1000, a = 1) {
  g <- function(x) exp(-x^2)
  ### calculate normalization constant
  nu <- mean(g(runif(1000, -a, a))) * 2 * a
  Y <- rejection_sampler(n_samples = n, a = a)
  mean(integrand(Y) / g(Y) * nu)
}

## sampler2: use density of continuous uniform function on [-a,a]
importance_sampler2 <- function(n = 1000, a = 1) {
  U <- runif(1000, -a, a)
  ### calculate normalization constant
  nu <- 2*a
  mean(integrand(U) * 2 * a)
}
```

Plot the samples (blue) for different values of a . The grey line is the numerical solution to the integral.



Show behaviour for different values of n :

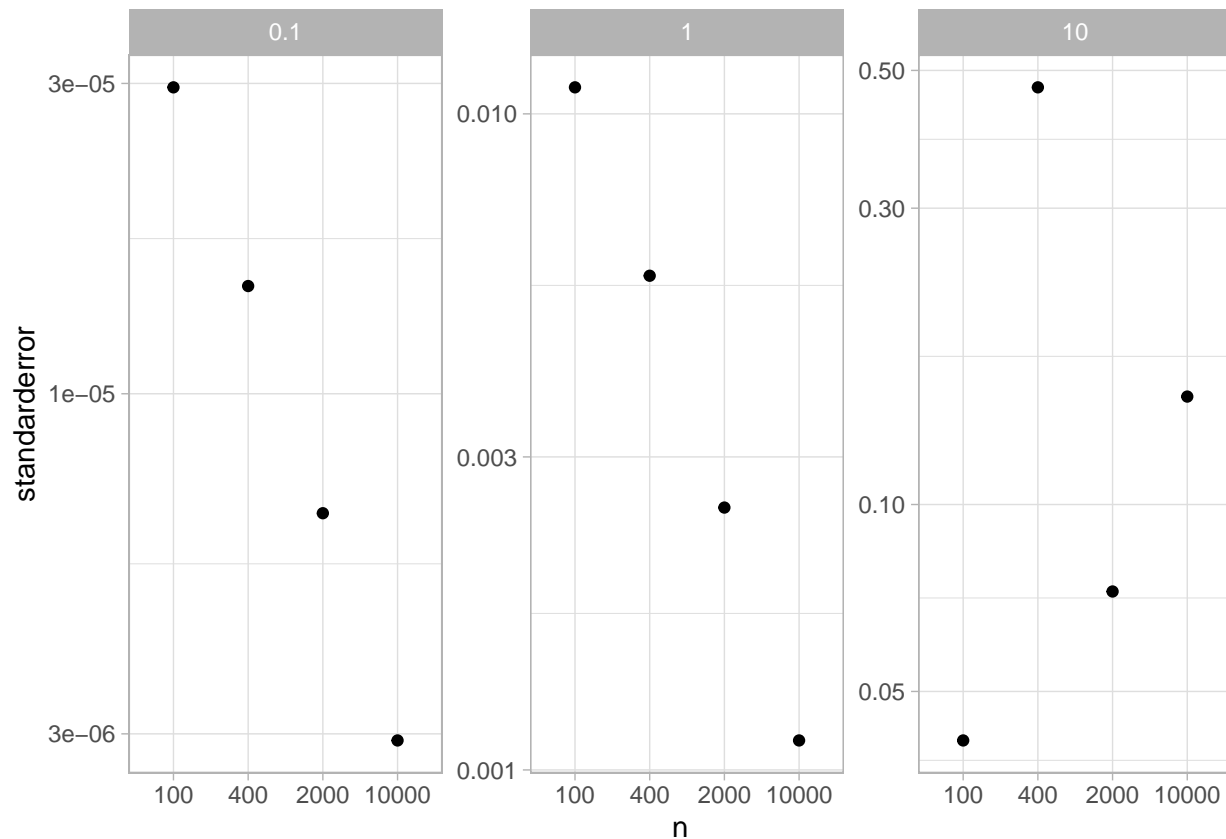


d) Computing standard error

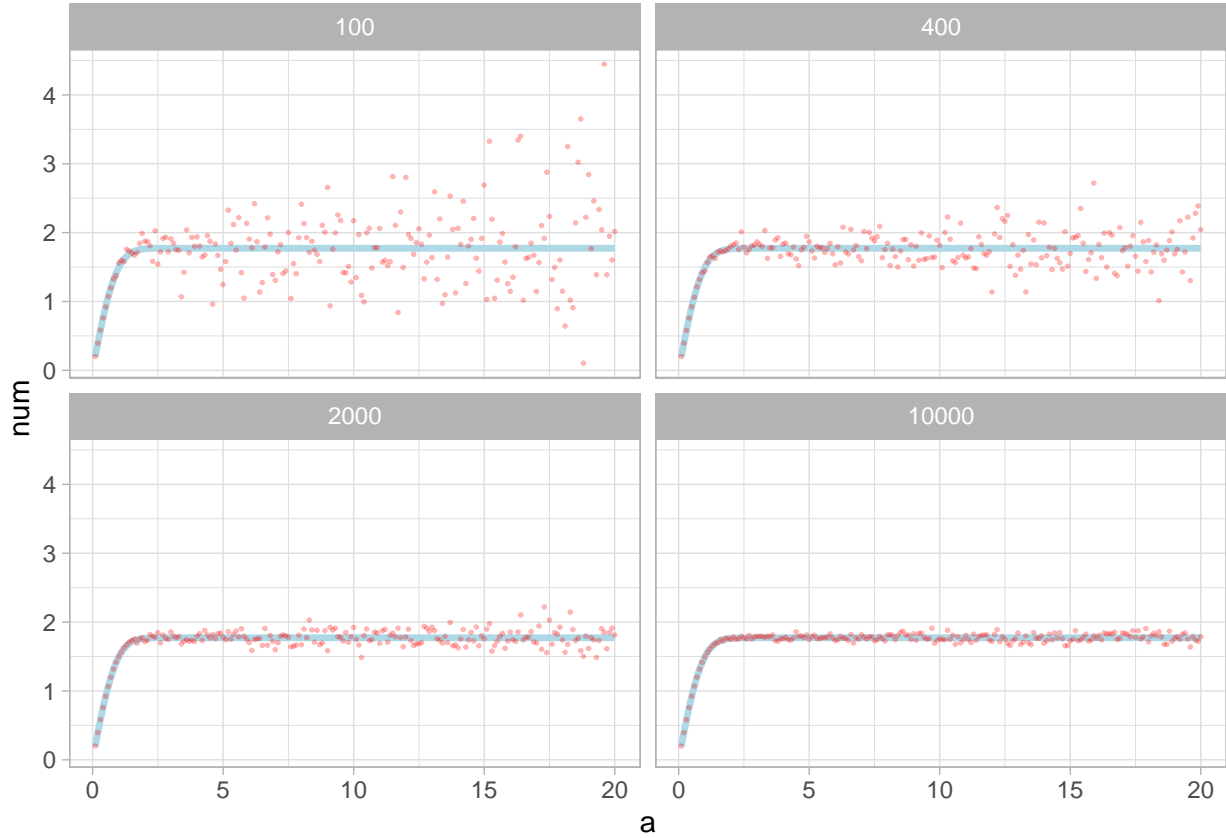
We compute the standard error as $var[I_a] = var[\mathbb{E}_G(h(X) \frac{f(X)}{g(X)})] = \frac{var[h(X) \frac{f(X)}{g(X)}]}{n}$

```
se_est <- function(n, a) {
  g <- function(x) exp(-x^2)
  integrand <- function(x) cos(x)/(1 + x^2)
  ### calculate normalization constant
  nu <- mean(g(runif(1000, -a, a))) * 2 * a
  Y <- rejection_sampler(n_samples = n, a = a)
  sigma <- sqrt(var(integrand(Y) / g(Y) * nu) / n)
  return(sigma)
}
```

Plotting se over a, for different n yields:



Plotting the samples for different n over a vs. the expected value:



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

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

