Carson Slater

6376 Homework 12 Solutions

```
source(here::here("helpers.R"))
```

Preliminary Citations

I got all of the random number generation stuff from *Statistical Computing with R* by Rizzo. Originally I was going to just do a `bench::mark()` for 3(c) but Thomas had the idea of doing `bench::press()`, so I totally stole that idea.

Question 1

I have done the reading for *Statistical Computing with R* by Rizzo.

Question 2

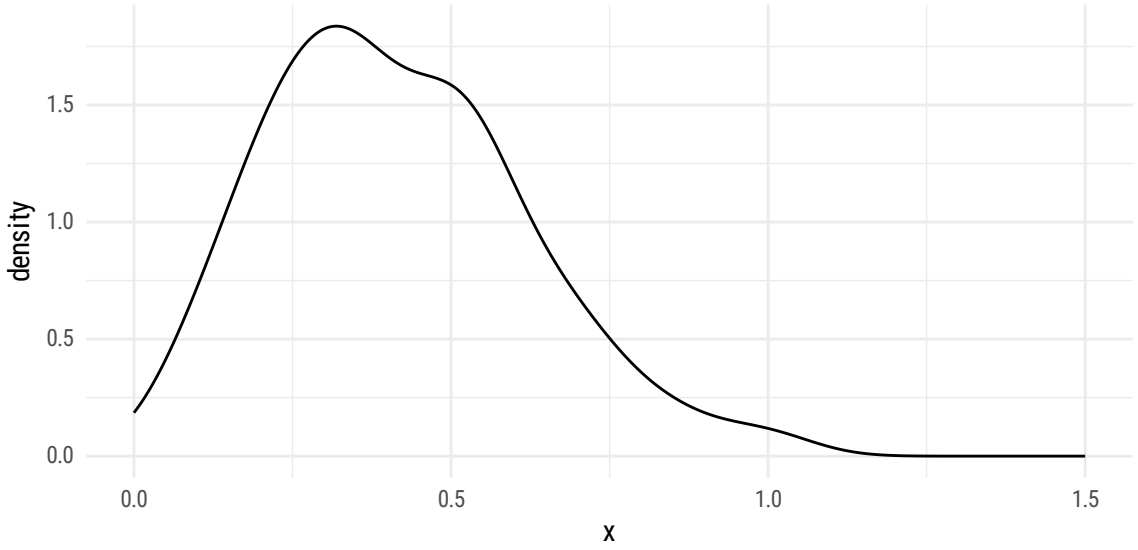
(a)

```
set.seed(1)
x <- rgamma(100, 3, 7)
round(x, 2)
```

```
  [1] 0.23 0.72 0.70 0.46 0.96 0.54 0.50 0.29 0.20 0.23 0.29 0.35 0.18 0.57 0.50
 [16] 0.59 0.56 0.37 0.05 0.48 0.64 0.22 0.26 0.31 0.33 0.23 0.11 0.62 0.27 0.65
 [31] 0.55 0.32 0.26 0.49 0.22 0.09 0.55 0.33 0.42 0.28 0.44 0.15 0.42 0.51 0.32
 [46] 0.50 0.33 0.35 0.53 0.36 0.21 0.11 0.76 0.39 1.02 0.47 0.21 0.33 0.13 0.39
 [61] 0.33 0.37 0.24 0.70 0.55 0.31 0.50 0.44 0.64 0.29 0.52 0.71 0.19 0.67 0.53
 [76] 0.81 0.49 0.13 0.41 0.62 0.56 0.20 0.18 0.15 0.87 0.54 0.59 0.45 0.84 0.23
 [91] 0.36 0.36 0.08 0.27 0.44 0.48 0.39 0.33 0.72 0.31
```

(b)

```
tibble("x" = x) |>
  ggplot(aes(x)) +
    geom_density() +
    xlim(c(0, 1.5))
```



(c)

```
l <- function(th, data = x) {
  sum(dgamma(data, th[1], th[2], log = TRUE))
}
```

```
l(c(1,1))
```

```
[1] -41.65701
```

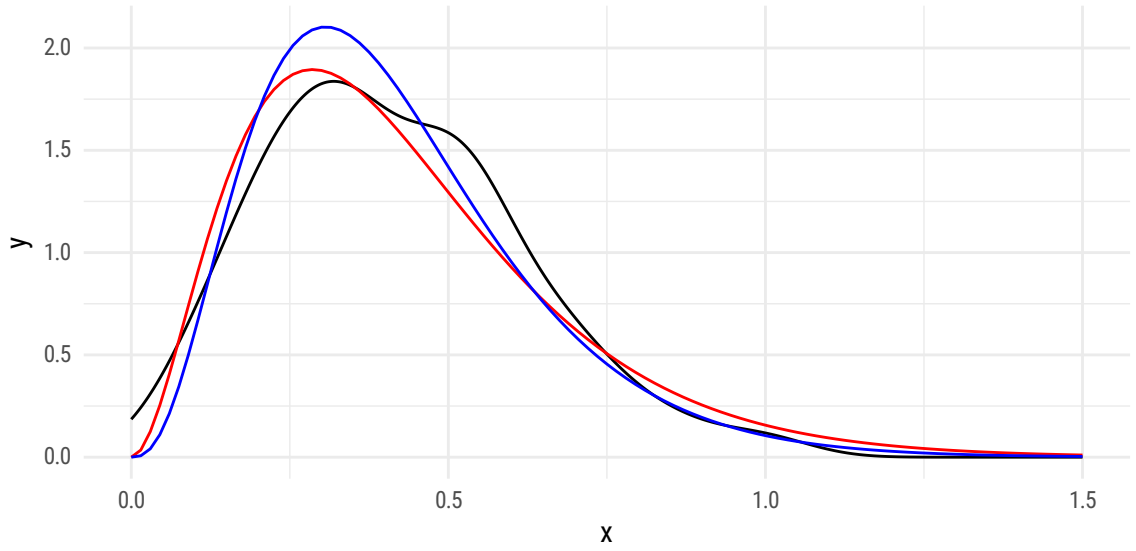
(d)

```
optim(c(2, 6), \(th) -l(th))
```

```
$par
[1] 3.767209 9.043331

$value
[1] -21.44579

$counts
function gradient
      67       NA

$convergence
[1] 0

$message
NULL
```

(e)

```
tibble("x" = x) |>
  ggplot(aes(x)) +
    geom_density() +
    geom_function(fun = \(x) dgamma(x, shape = 3, rate = 7), color = "red") +
    geom_function(fun = \(x) dgamma(x, shape = 3.77, rate = 9.04), color = "blue") +
    xlim(c(0, 1.5))
```



(f)

```
n <- 100
nrep <- 1000
data_sets <- replicate(nrep, rgamma(n, 3, 7), simplify = FALSE)
str(data_sets, list.len = 3)
```

```
List of 1000
 $ : num [1:100] 0.23 0.721 0.702 0.457 0.962 ...
 $ : num [1:100] 0.15 0.536 0.341 0.133 0.288 ...
 $ : num [1:100] 0.498 0.447 0.268 0.492 0.147 ...
  [list output truncated]
```

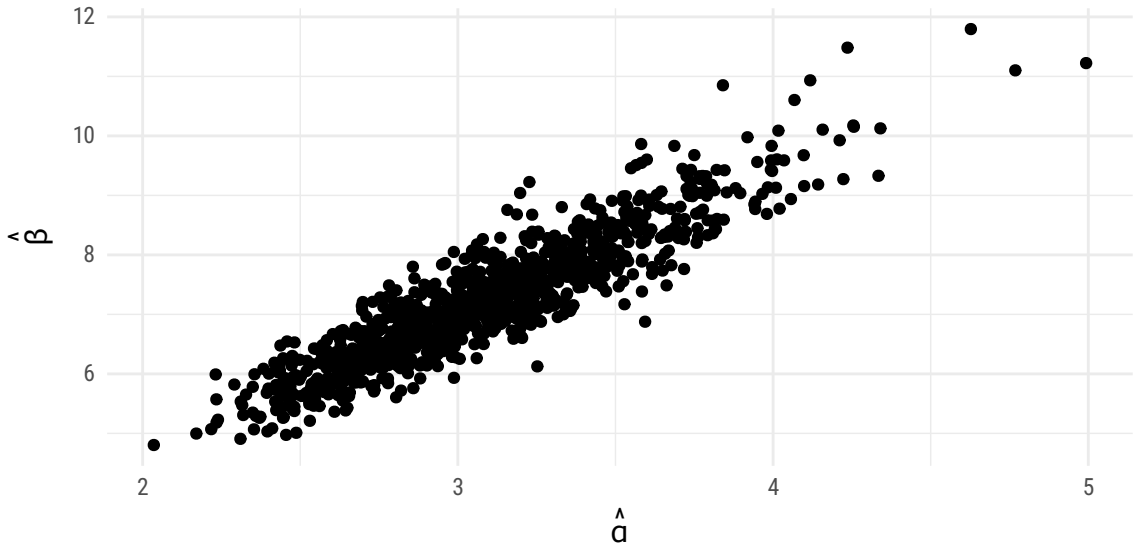
(g)

```
estims <- lapply(data_sets, \(x) {
    optim(c(2, 6), \(th) -l(th, x))$par
  }
)

ests_df <- tibble(
  "alpha" = sapply(1:nrep, \(i) estims[[i]][1]),
  "beta"  = sapply(1:nrep, \(i) estims[[i]][2])
)

str(ests_df)
```

```
tibble [1,000 x 2] (S3: tbl_df/tbl/data.frame)
 $ alpha: num [1:1000] 3.77 2.5 3.1 2.76 2.94 ...
 $ beta : num [1:1000] 9.04 6.24 7.01 6.42 6.95 ...
```

(h)

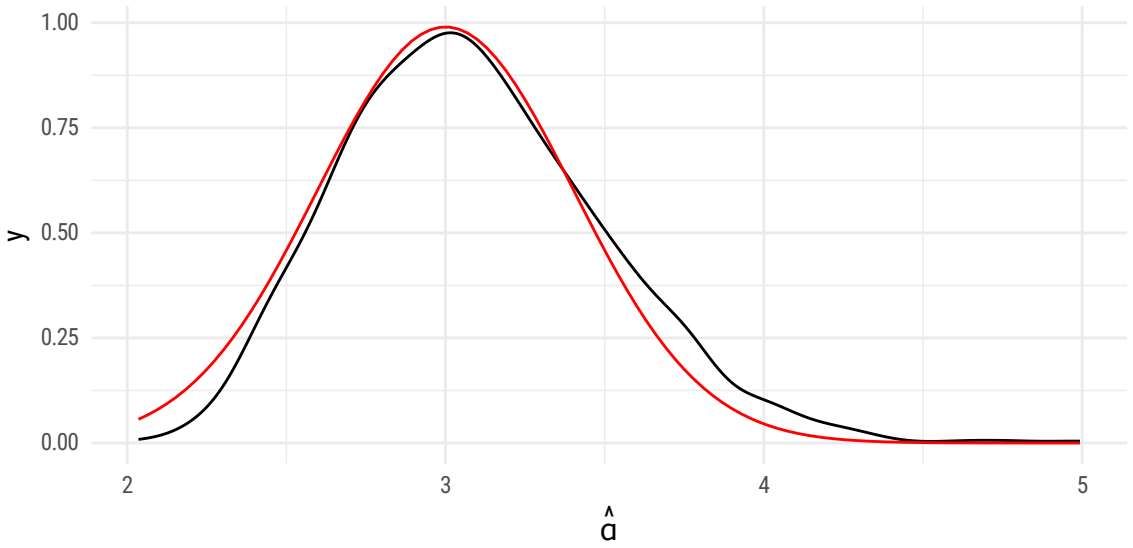
```
ests_df |>
  ggplot(aes(alpha, beta)) +
    geom_point() +
    labs(x = expression(hat(alpha)),
         y = expression(hat(beta)))
```



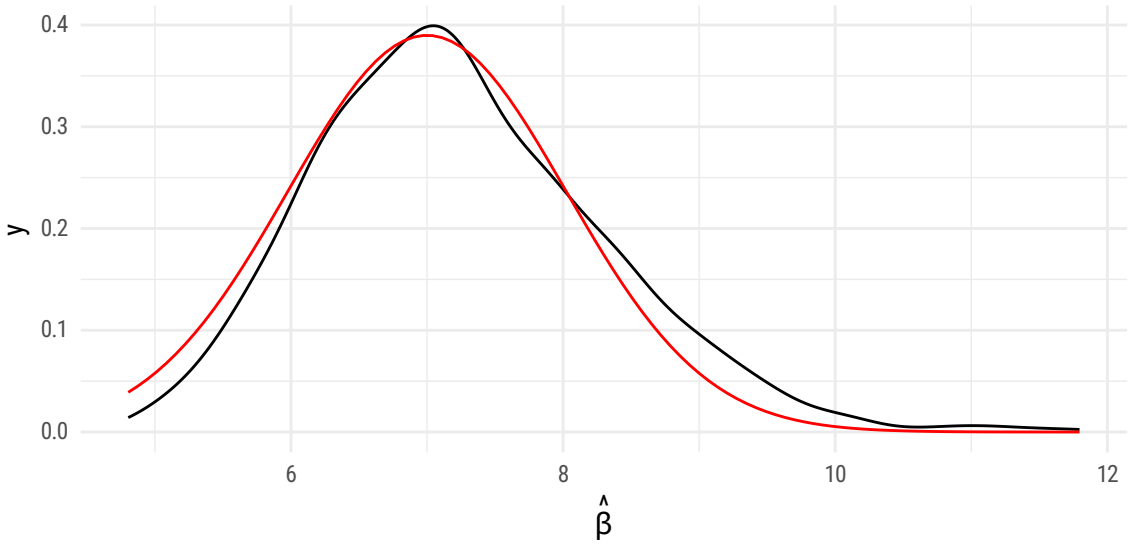
(i)

```
I <- function(th) {
  matrix(c(trigamma((th[1])), 1/th[2], 1/th[2], th[1]/th[2]^2), byrow = TRUE, nrow = 2)
}

Si <- solve(I(c(3,7)))
```

```
ests_df |>
  ggplot(aes(alpha)) +
    geom_density() +
    geom_function(fun = \(x) dnorm(x, 3, sqrt(Si[1, 1]/n)), color = "red") +
    labs(x = expression(hat(alpha)))
```




```
ests_df |>
  ggplot(aes(beta)) +
    geom_density() +
    geom_function(fun = \(x) dnorm(x, 7, sqrt(Si[2, 2]/n)), color = "red") +
    labs(x = expression(hat(beta)))
```



(k)

Overall the theoretical asymptotic distributions of α and β are very consistent with the simulated large samples of their estimates.

Question 3

(a)

```
rchisq2 <- function(n, df) {
  z <- runif(df) |> qnorm() |> replicate(n, expr = _)
  apply(z, MARGIN = 2, \(z) sum(z^2))
}
```

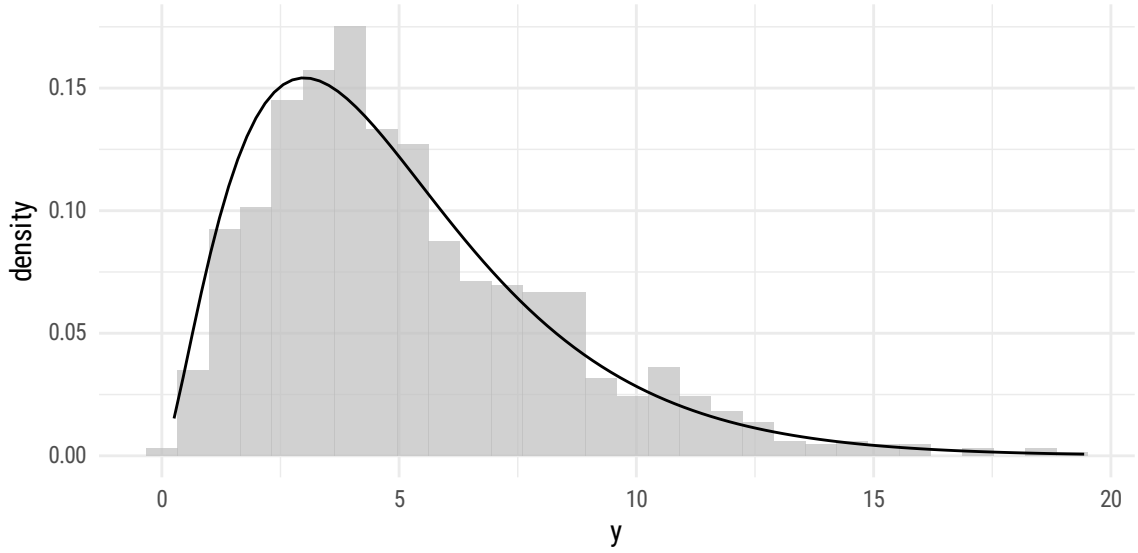
```
set.seed(1)
rchisq2(10, 5) |> round(2)
```

```
 [1] 3.00 6.82 2.40 6.79 4.78 6.51 1.75 2.80 1.62 5.25
```

(b)

```
rchisq2(1000, 5) |>
  tibble("y" = _) |>
  ggplot() +
  geom_histogram(aes(x = y, y = after_stat(density)), fill = "grey", alpha = 0.7) +
  geom_function(fun = \(x) dchisq(x, df = 5), color = "black")
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



(c)

```
# Run the benchmarks using bench::press
results <- bench::press(
  n =c(100, 1000, 10000),
  df = c(3, 5, 10),
  {
    bench::mark(
      rchisq  = rchisq(n, df),
      rchisq2 = rchisq2(n, df),
      check   = FALSE
    )
  }
)

results[,1:7] |> knitr::kable()
```

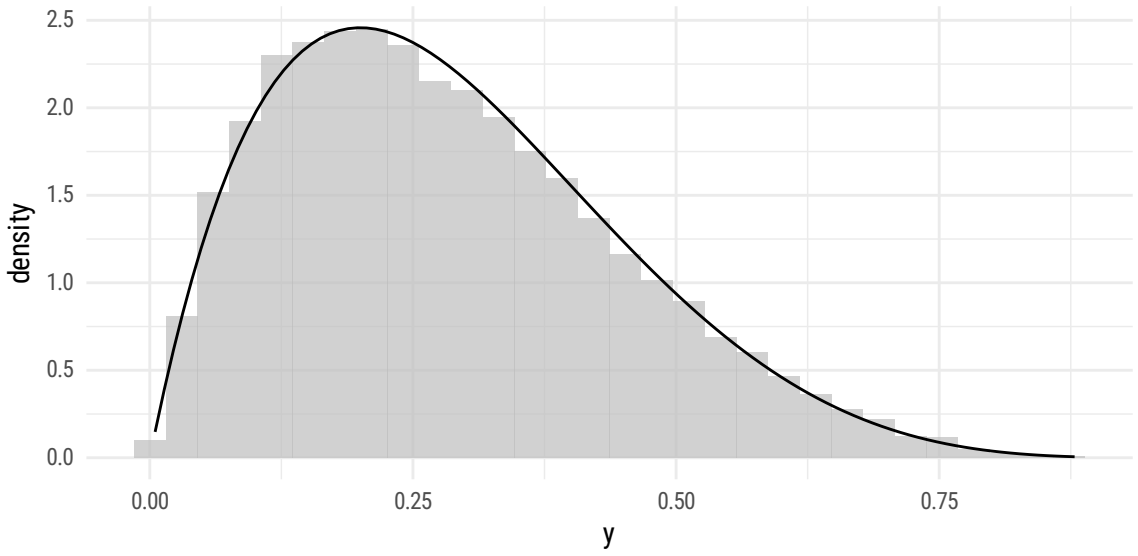
expression	n	df	min	median	itr/sec	mem_alloc
rchisq	100	3	7.38us	10.62us	92519.50365	6.91KB
rchisq2	100	3	311.31us	332.9us	2974.61045	264.42KB
rchisq	1000	3	81.51us	94.75us	10533.73016	10.35KB
rchisq2	1000	3	2.93ms	3.04ms	327.78283	2.57MB
rchisq	10000	3	902.49us	942.69us	1058.83820	80.66KB
rchisq2	10000	3	30.93ms	30.96ms	32.29998	25.81MB
rchisq	100	5	7.05us	10.17us	95904.97944	3.32KB
rchisq2	100	5	317.38us	331.38us	2990.57588	269.11KB
rchisq	1000	5	76.71us	91.68us	10866.27653	10.35KB
rchisq2	1000	5	2.97ms	3.1ms	322.25634	2.62MB
rchisq	10000	5	877.4us	911.66us	1095.93698	80.66KB
rchisq2	10000	5	30.66ms	30.72ms	32.25078	26.27MB
rchisq	100	10	6.56us	8.36us	117684.19525	3.32KB
rchisq2	100	10	336.65us	346.9us	2843.44793	280.83KB
rchisq	1000	10	66.62us	72.65us	13702.84749	10.35KB
rchisq2	1000	10	3.19ms	3.35ms	300.27216	2.73MB
rchisq	10000	10	696.14us	720.08us	1378.44654	80.66KB
rchisq2	10000	10	36.16ms	40.37ms	24.56343	27.41MB

On average, my `rchisq2()` was approximately 30 times slower than `stats::rchisq()`.

Question 4

```
rbeta2 <- function(n, shape1, shape2) {
  accept <- 0
  sample <- numeric(n)
  mode <- dbeta((shape1 - 1)/(shape1 + shape2 - 2), shape1, shape2)

  while (accept <= n) {
    y <- runif(1)
    u <- runif(1)

    if (u <= dbeta(y, shape1, shape2)/mode) {
      accept <- accept + 1
      sample[accept] = y
    }
  }
  sample
}
```

```
# Parameters
n <- 10000
shape1 <- 2 # a = 2
shape2 <- 5 # b = 5
x <- seq(0, 1, length.out = n)

rbeta2(n, shape1, shape2) |>
  tibble("y" = _) |>
  ggplot() +
  geom_histogram(aes(x = y, y = after_stat(density)), fill = "grey", alpha = 0.7) +
  geom_function(fun = \(x) dbeta(x, shape1, shape2), color = "black")
```



Here we see that the `rbeta2(10000, 2, 5)` generates a sample that resembles the shape of a Beta(2, 5) distribution when plotted against a histogram.

Question 5

(a)

```
sqrt_mat <- function(mat) {
  decomp <- eigen(mat)
  decomp$vectors %*% sqrt(diag(decomp$values)) %*% t(decomp$vectors)
}
```

```
mat <- matrix(c(15, 7.5, 7.5, 5), 2, 2)
sqrt_mat(mat)
```

```
         [,1]     [,2]
[1,] 3.610727 1.400946
[2,] 1.400946 1.742800
```

```
sqrt_mat(mat) %*% sqrt_mat(mat)
```

```
     [,1] [,2]
[1,] 15.0  7.5
[2,]  7.5  5.0
```

(b)

```
rmvnorm <- function(n, mean, cov) {
  stopifnot(ncol(mat) == length(mean))

  p <- length(mean)
  Z <- replicate(n, rnorm(p)) |> unlist()
  mu_mat <- rep(mean, times = n) |> matrix(nrow = n)

  t(sqrt_mat(mat) %*% Z) + mu_mat
}
```

```
set.seed(1)
rmvnorm(3, c(0,0), mat)
```

```
           [,1]       [,2]
[1,] -2.0046796 -0.5575741
[2,] -0.7823256  1.6095851
[3,]  0.0403312 -0.9682898
```

```
rmvnorm(1e5, c(0,0), mat) |> cor()
```

```
          [,1]      [,2]
[1,] 1.0000000 0.8663472
[2,] 0.8663472 1.0000000
```

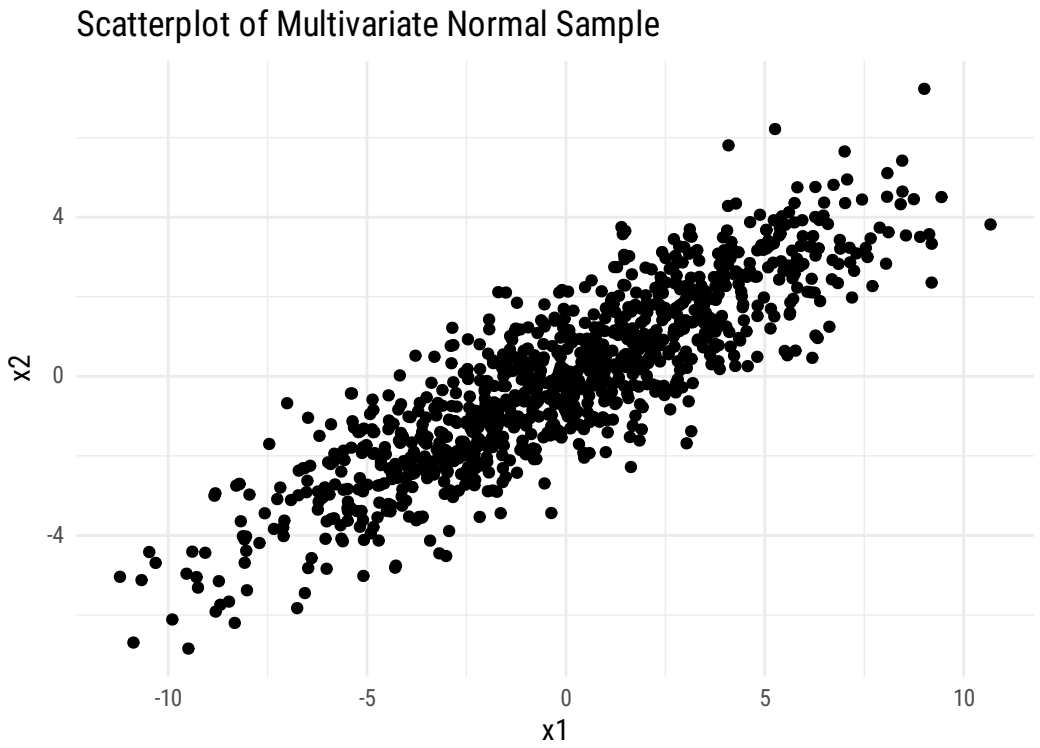
```
mat[1,2] / sqrt(mat[1,1] * mat[2,2]) # = exact correlation
```

```
[1] 0.8660254
```

(c)

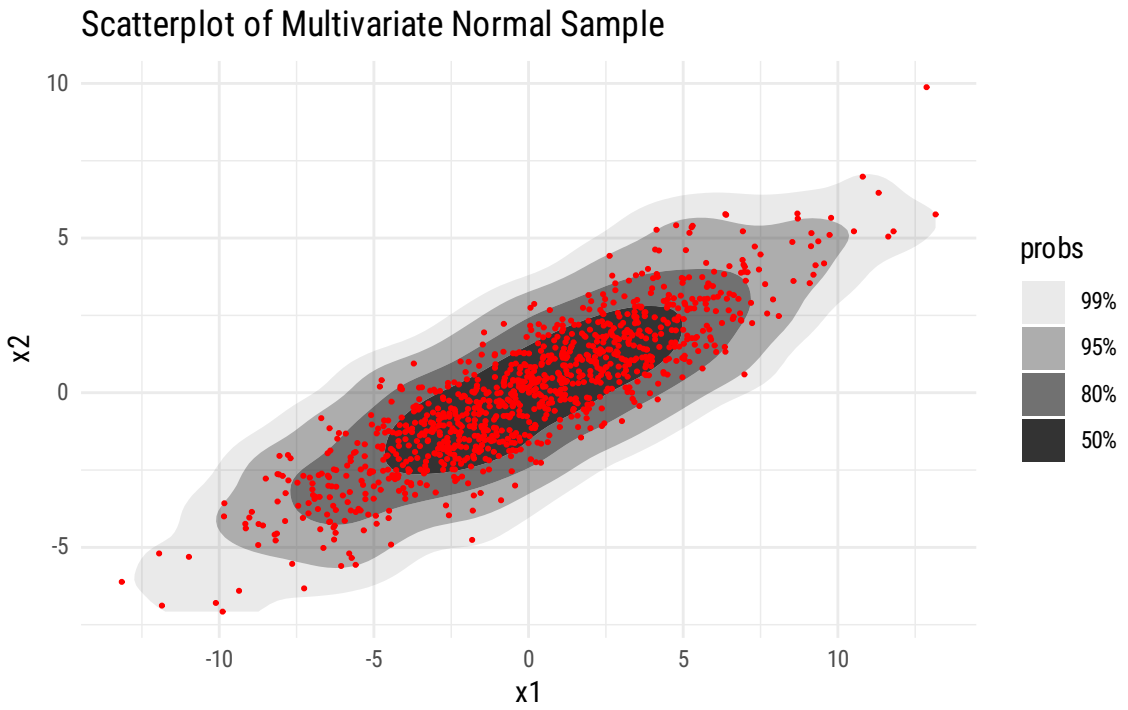
```
rmvnorm(1e3, c(0,0), mat) |>
  as_tibble() |>
  ggplot(aes(V1, V2)) +
    geom_point() +
    labs(title = "Scatterplot of Multivariate Normal Sample",
```

```
        x = "x1",
        y = "x2") +
    coord_equal()
```



(d)

```
data <- rmvnorm(1e3, c(0,0), mat) |> as_tibble()

data |>
  ggplot(aes(V1, V2)) +
    geom_hdr() +
    geom_point(color = "red", size = 0.4) +
    labs(title = "Scatterplot of Multivariate Normal Sample",
         x = "x1",
         y = "x2") +
    coord_equal()
```



(e)

The shapes of the contours generally seem to be elliptical, with wavy edges that are consistent with each other ellipse.

(f)

```
# sample mean
xbar <- apply(data, 2, mean) |> unname()

# sample covariance
cov(data)
```

```
          V1       V2
V1 16.303042 8.076332
V2  8.076332 5.196547
```

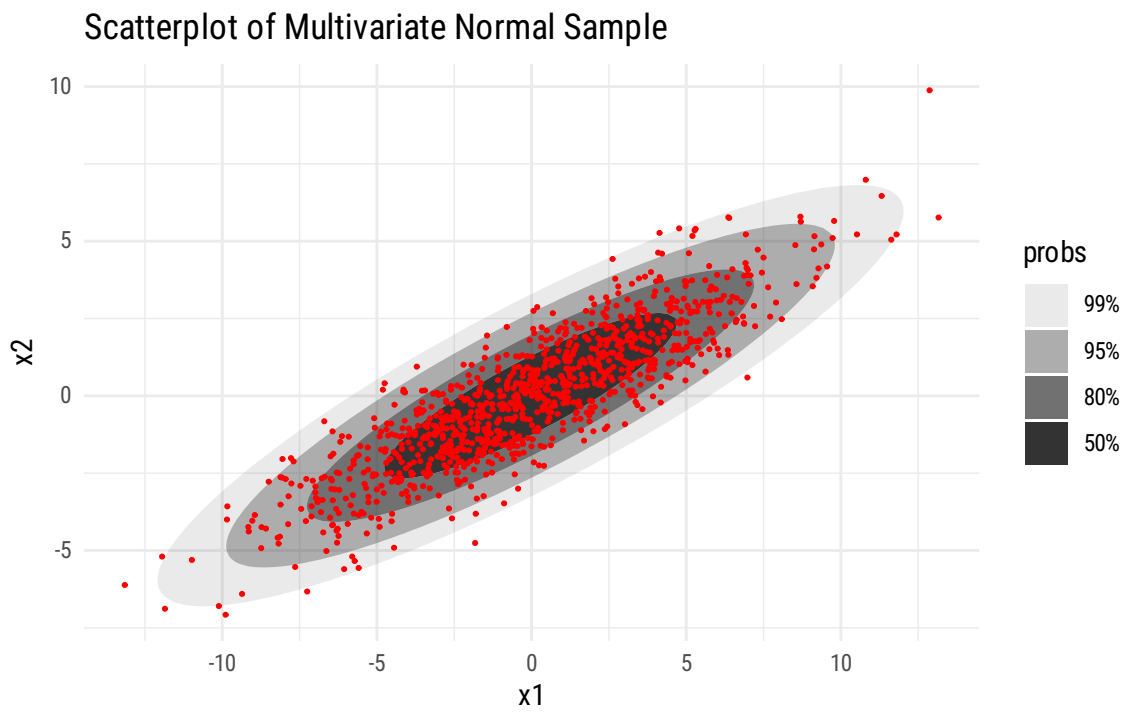
(g)

```
(t(as.matrix(data)) %*% as.matrix(data) - xbar %*% t(xbar))/1e3
```

```
          V1       V2
V1 16.287854 8.068067
V2  8.068067 5.191382
```

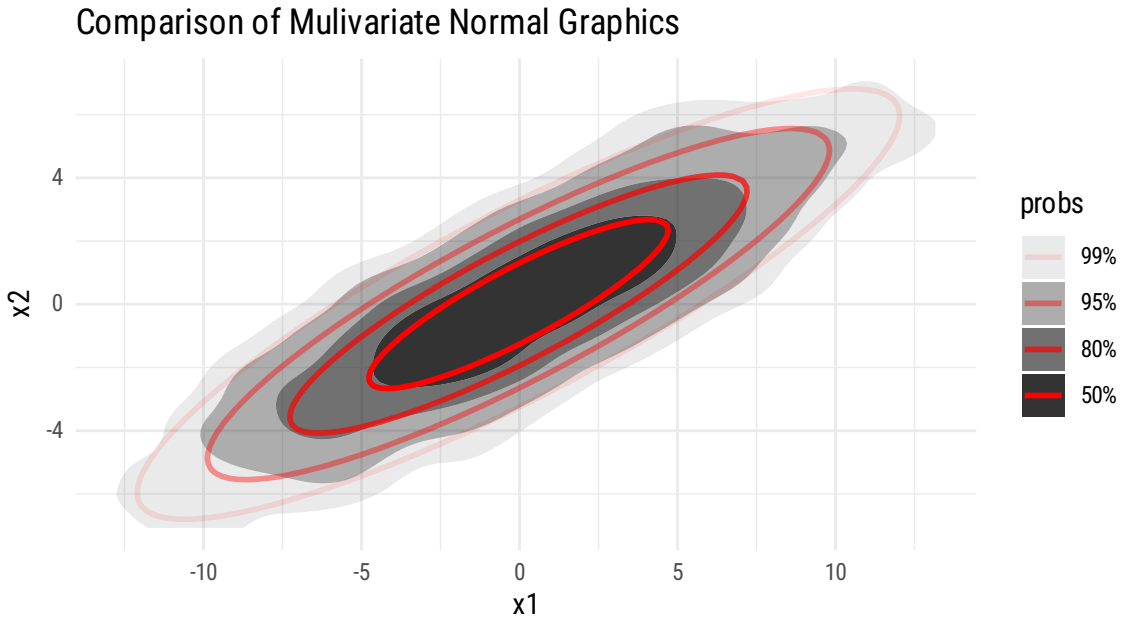
(f)

```
data |>
  ggplot(aes(V1, V2)) +
    geom_hdr(method = "mvnorm") +
    geom_point(color = "red", size = 0.4) +
    labs(title = "Scatterplot of Multivariate Normal Sample",
         x = "x1",
         y = "x2") +
    coord_equal()
```



(h)

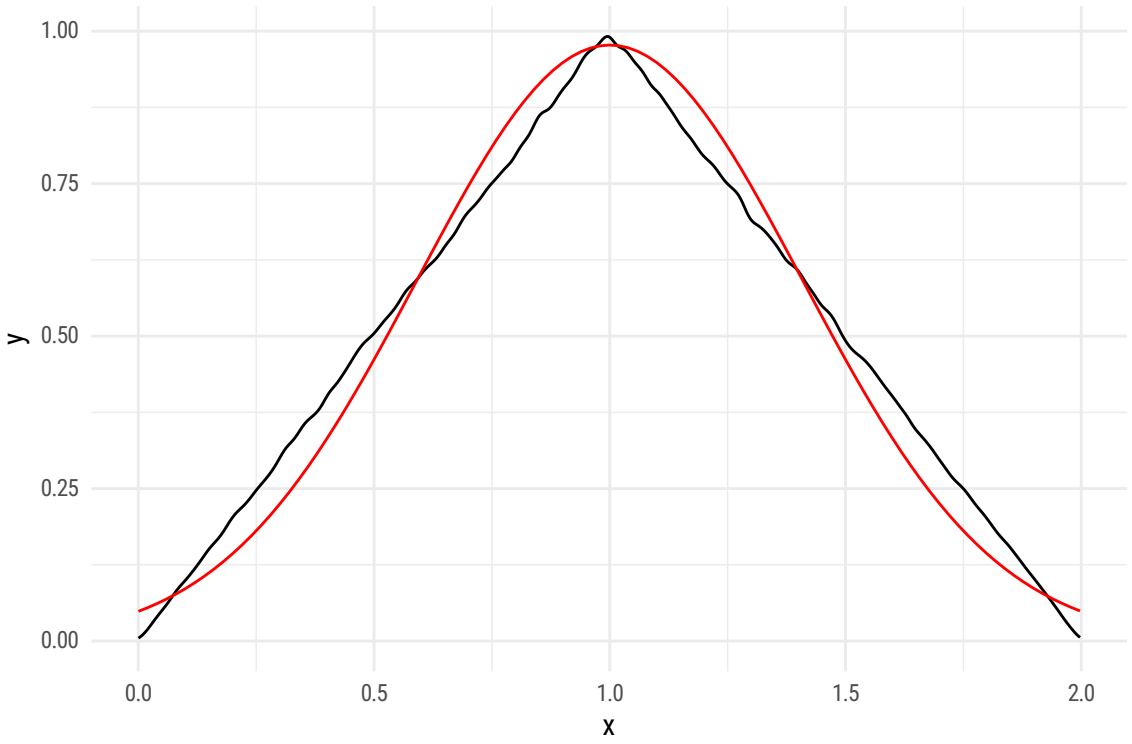
```
data |>
  ggplot(aes(V1, V2)) +
    geom_hdr() +
    geom_hdr_lines(method = "mvnorm", color = "red") +
    labs(title = "Comparison of Mulivariate Normal Graphics",
         x = "x1",
         y = "x2") +
    coord_equal()
```



Question 6

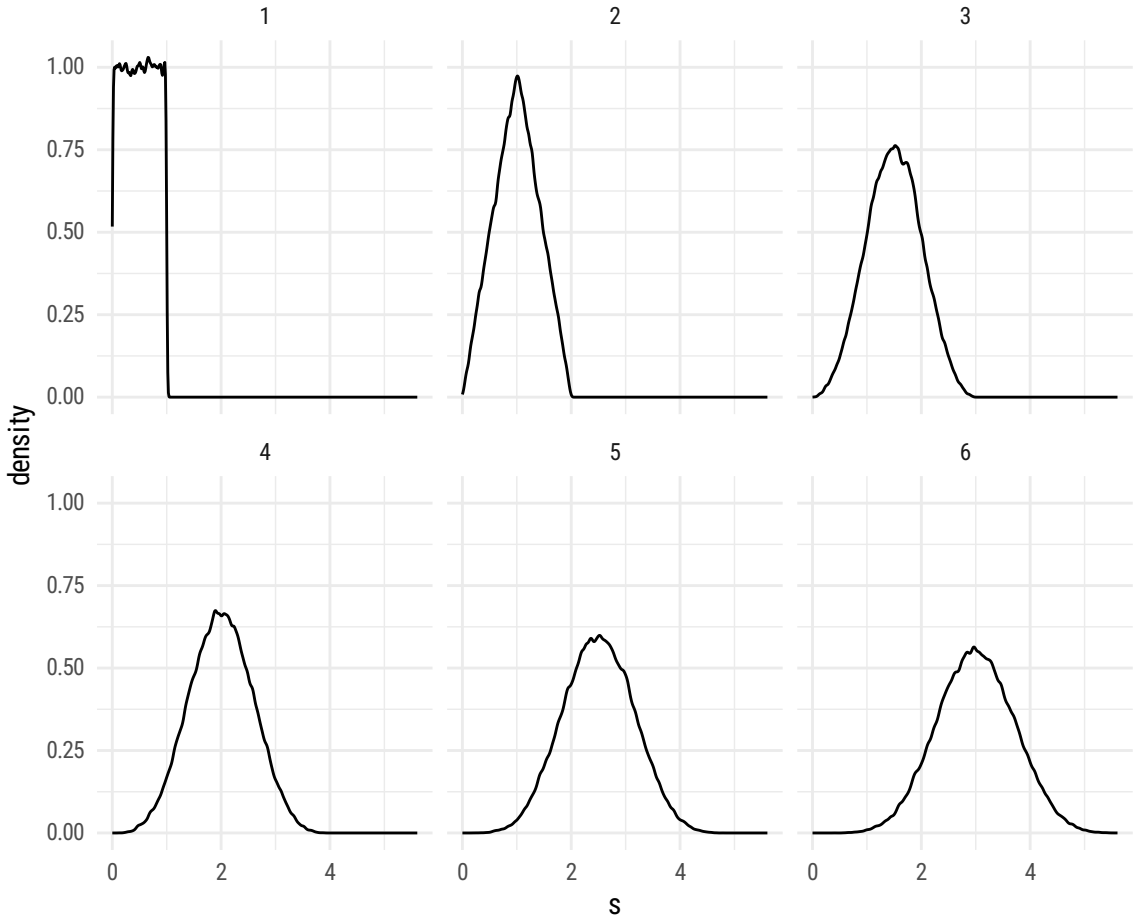
(a)

```
x1 <- runif(1e6)
x2 <- runif(1e6)
s2 <- tibble("x" = x1 + x2)

s2 |> ggplot(aes(x = x)) +
  geom_density(adjust = 1/2) +
  geom_function(fun = \(x) dnorm(x, 1, sqrt(1/6)), color = "red")
```



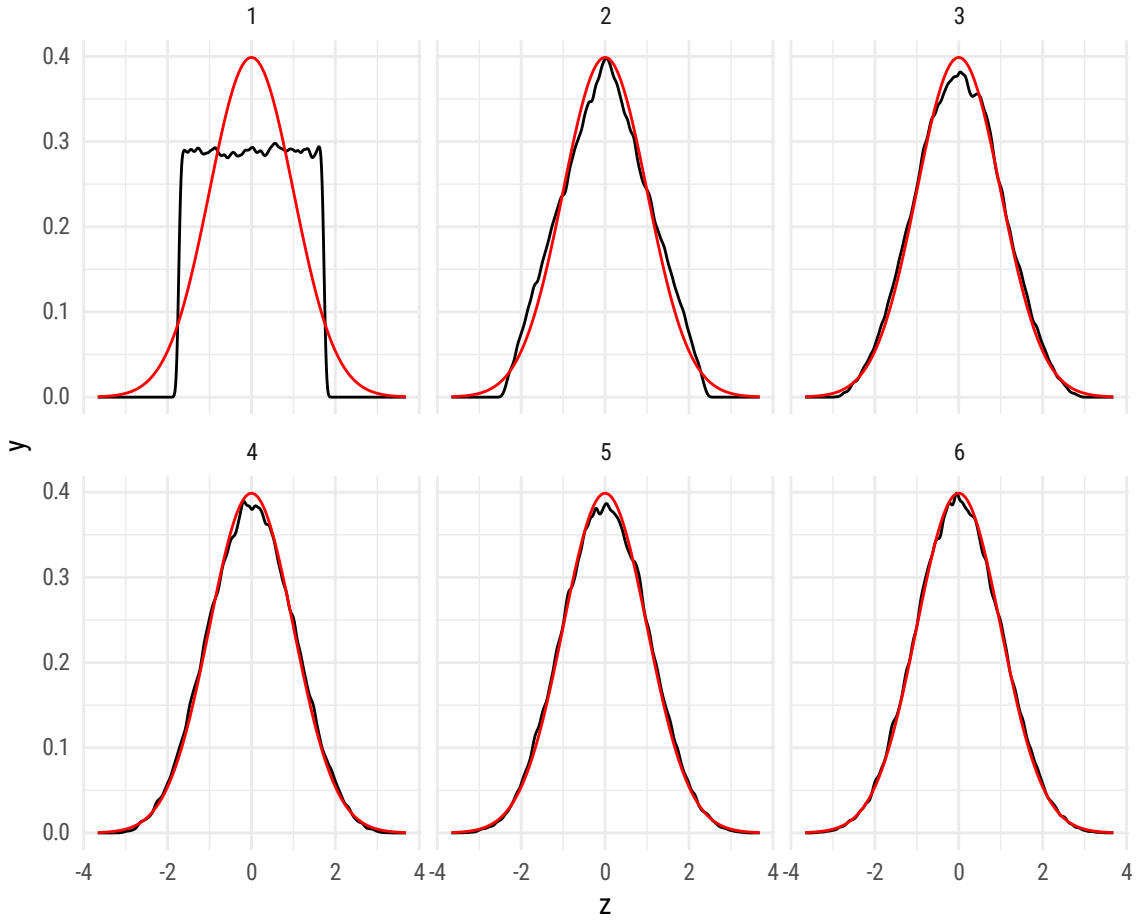
(b)

```
sum_unifs <- function(n, sample_size = 1e5) {
  replicate(n, runif(sample_size)) |> apply(1, sum)
}

samples <- lapply(1:6, \(x) tibble(
    "s" = sum_unifs(x),
    "n" = x,
    "z" = (s - (x/2))/sqrt(x/12)
    )
  )

samples_pivoted <- samples |> reduce(rbind)

samples_pivoted |> ggplot(aes(s)) +
  geom_density(adjust = 1/2) +
  facet_wrap(n ~ .)
```



(c)

```
# I created z-transformed variables in 6b already

samples_pivoted |> ggplot(aes(z)) +
  geom_density(adjust = 1/2) +
  geom_function(fun = \(x) dnorm(x), color = "red") +
  facet_wrap(n ~ .)
```



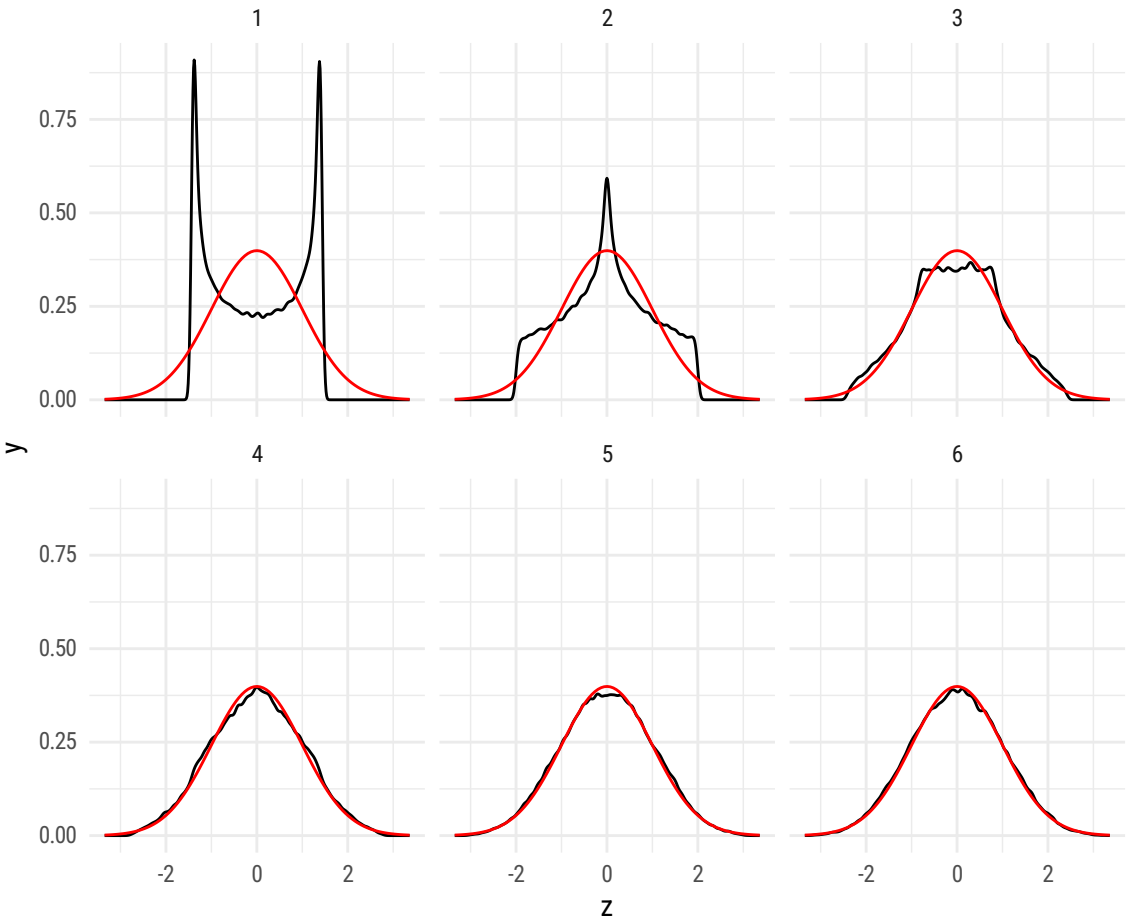
(d)

It appears the convolved uniform distribution resembles a gaussian shape distribution after just three convolutions, which far less than the typical 30 'rule of thumb.' It appears that the 'rule of thumb' might be rather irrelevant for uniform samples, since it appears to converge much faster.

(e)

```
sum_betas <- function(n, sample_size = 1e5) {
  replicate(n, rbeta(sample_size,  1/2, 1/2)) |> apply(1, sum)
}

beta_samples <- lapply(1:6, \(x) tibble(
```

```
    "s" = sum_betas(x),
    "n" = x,
    "z" = (s - (x/2))/sqrt(x*(0.25/2))
    )
  ) |> reduce(rbind)

beta_samples |> ggplot(aes(z)) +
  geom_density(adjust = 1/2) +
  geom_function(fun = \(x) dnorm(x), color = "red") +
  facet_wrap(n ~ .)
```



The approximation goes from a horseshoe-shaped distribution to a gaussian shape rather quickly, and is very close to the standard normal distribution after four convolutions - slightly slower than the uniform convolutions.