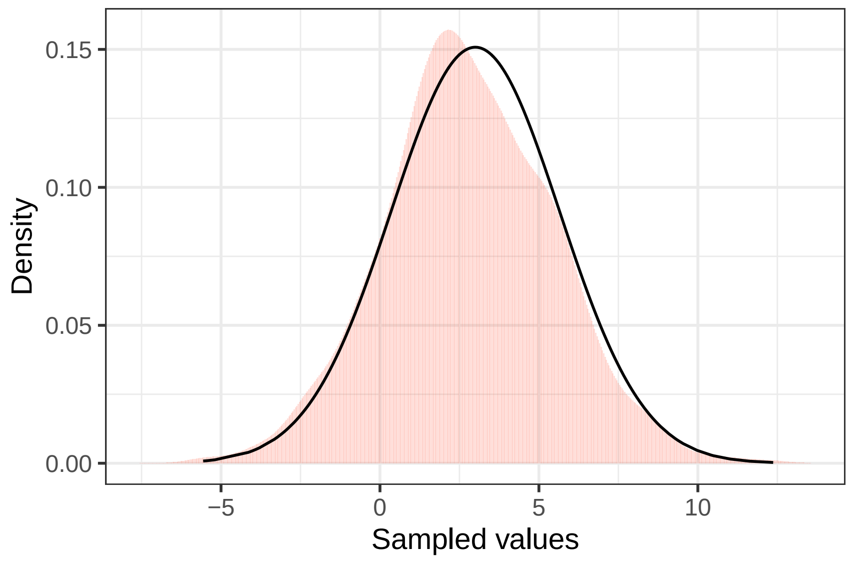
Exercises

Bruna Wundervald

Exercise 2.1 (Preliminary Simulation). Familiarise yourself with the support provided by `R` for simple simulation tasks. In particular:

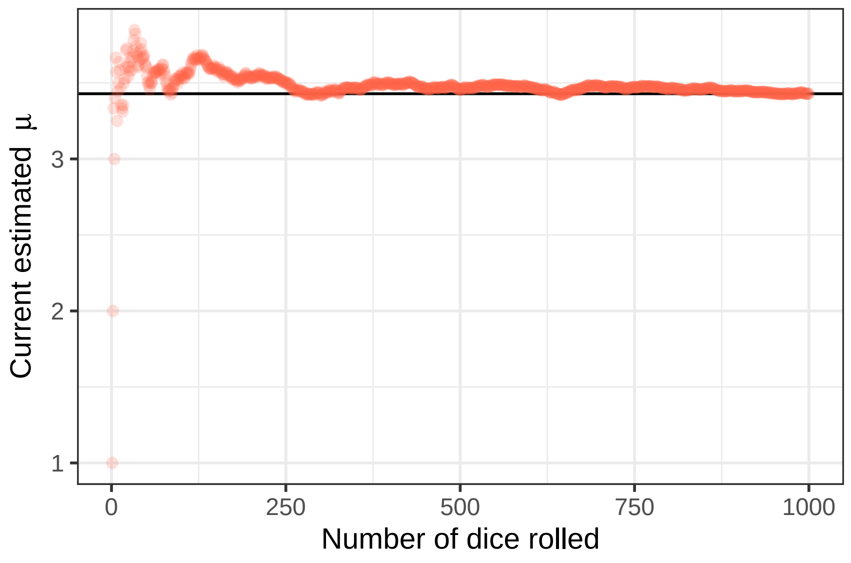
1. Generate a large sample of $N(3, 7)$ random variables (see `rnorm`); plot a histogram (hist with sensibly-chosen bins) of your sample and overlay a normal density upon it (see `dnorm`, `lines`).
2. Use sample to simulate the rolling of 1,000 dice; use your sample to estimate the average value obtained when rolling a standard die. Show how the estimate obtained using n dice behaves for $n \in [0, 1000]$ using a simple plot.

```r
set.seed(2019)

# Normal exercise ------------------------
samp <- rnorm(1000, mean = 3, sd = sqrt(7))

data.frame(samp,
           dens = dnorm(samp, mean = 3, sd = sqrt(7))) %>%
  ggplot(aes(x = samp)) +
  geom_histogram(alpha = 0.2,
                 fill = "tomato", stat = "density") +
  geom_line(aes(x = samp, y = dens)) +
  xlim(3 - 4*sqrt(7), 3 + 4*sqrt(7)) +
  labs(x = "Sampled values", y = "Density") +
  theme_bw()
```



```r
# Rolling dices exercise -----------------
dice <- 1000
vals <- 1:6
reps <- replicate(n = dice, sample(vals, size = 1))
```

```r
data.frame(reps,
           n = 1:dice,
           sum = cumsum(reps)) %>%
  mutate(mu_hat = sum/n) %>%
  ggplot(aes(x = n, y = mu_hat)) +
  geom_hline(yintercept = mean(reps)) +
  geom_point(alpha = 0.2,
             colour = "tomato") +
  labs(x = "Number of dice rolled",
       y = expression("Current estimated "~mu)) +
  theme_bw()
```

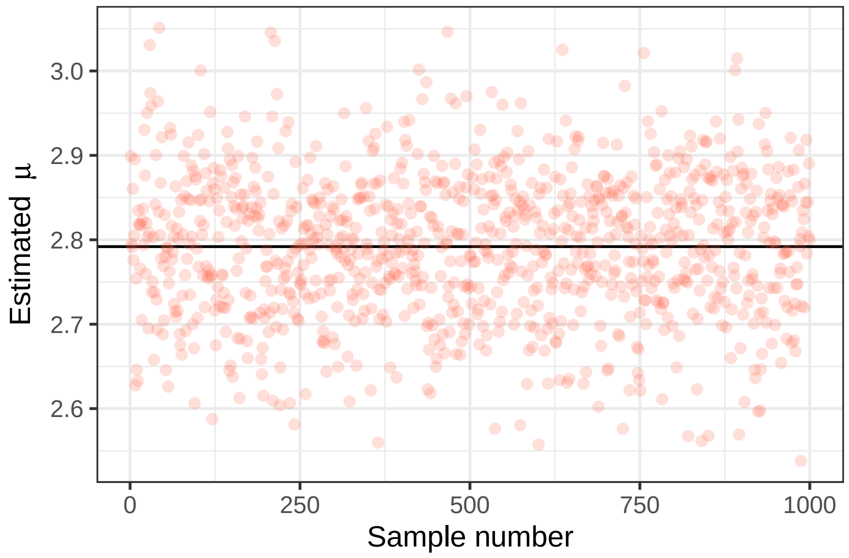


Exercise 2.2 (Towards Bootstrap Methods). Imagine that you have a sample of 1,000 values from a population of unknown distribution (for our purposes, you can obtain such a sample using `rnorm` as in the previous question and pretending that the distribution is unknown).

1. Write code to repeat the following 1,000 times:

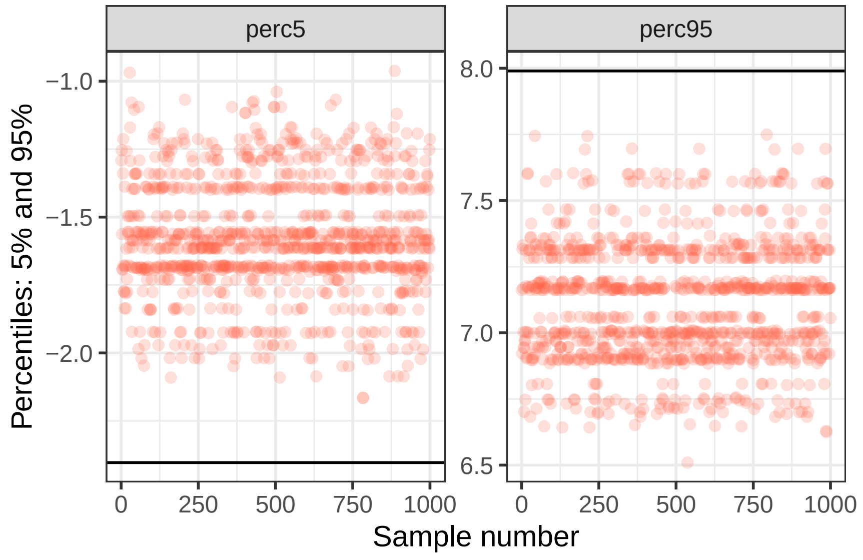
(a) Sample 1,000 times with replacement from the original sample to obtain 1,000 resampled sets of values.
(b) Compute the sample mean of your resampled set.

```r
samps <- map(1:1000,
             ~{sample(samp, size = length(samp),
                      replace = TRUE)}) %>%
  unlist()


df <- data.frame(samps = samps,
           ind = rep(1:1000, each = length(samp)))

df %>%
  group_by(ind) %>%
  summarise(mu_hat = mean(samps)) %>%
  #mutate(dens = dnorm(mu_hat, mean = 3, sd = sqrt(7))) %>%
  ggplot(aes(x = ind, y = mu_hat)) +
  geom_hline(yintercept = mean(samps)) +
  geom_point(alpha = 0.2,
             colour = "tomato") +
  #geom_line(aes(x = dens, y = mu_hat)) +
  labs(x = "Sample number",
       y = expression("Estimated "~mu)) +
  theme_bw()
```



2. You now have 1,000 sample means for resampled subsets of your data. Find the 5th and 95th percentile of this collection of resampled means.
3. How does this compare with a standard 90% confidence interval for the mean of a sample of size 1,000 from your chosen distribution?

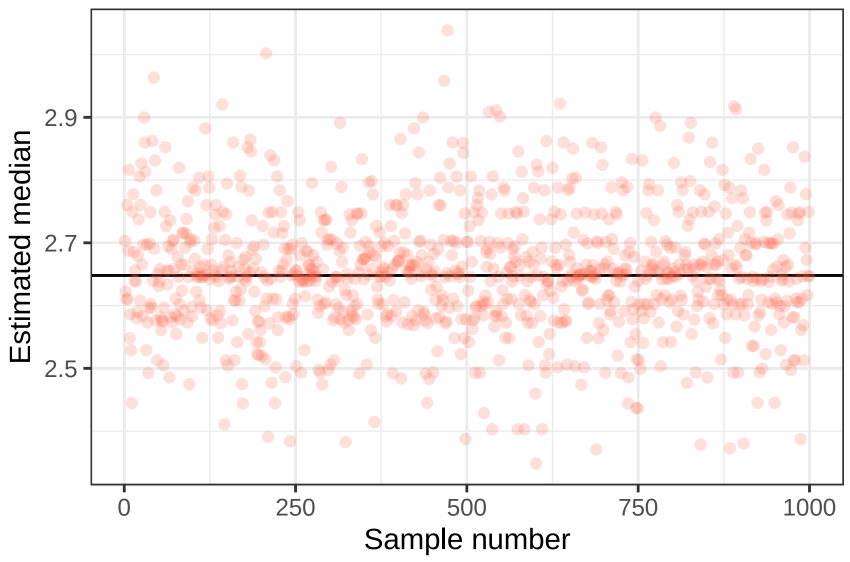
```
df2 <- df %>%
  group_by(ind) %>%
  summarise(perc5 = quantile(samps, 0.05),
            perc95 = quantile(samps, 0.95)) %>%
  gather(type, value, -ind )  %>%
  arrange(type) %>%
  mutate(confs =
           rep(
             c(mean(samp) - 1.96 * sd(samp),
               mean(samp) + 1.96 * sd(samp)),
             each = 1000))

df2 %>%
ggplot(aes(x = ind, y = value)) +
  facet_wrap(~type, scales = "free") +
  geom_hline(data = df2,
             aes(yintercept = confs)) +
  geom_point(alpha = 0.2,
             colour = "tomato") +
  labs(x = "Sample number",
       y = "Percentiles: 5% and 95%") +
  theme_bw()
```



4. Repeat the above using the median rather than mean.

5. Why might this be a useful technique? Note that we haven't done anything to justify the approach yet.

```
df %>%
  group_by(ind) %>%
  summarise(median_hat = median(samps)) %>%
  ggplot(aes(x = ind, y = median_hat)) +
  geom_hline(yintercept = median(samps)) +
  geom_point(alpha = 0.2,
             colour = "tomato") +
  labs(x = "Sample number",
       y = expression("Estimated median")) +
  theme_bw()
```



Exercise 2.3 (Transformation Methods). The Box-Muller method transforms a pair of uniformly-distributed random variables to obtain a pair of independent standard normal random variates. If

$$U_1, U_2 \sim U[0,1]$$

and

$$X_1 = \sqrt{-2log(U_1)}cos(2\pi U_2)$$
$$X_2 = \sqrt{-2log(U_1)}cot(2\pi U_2),$$

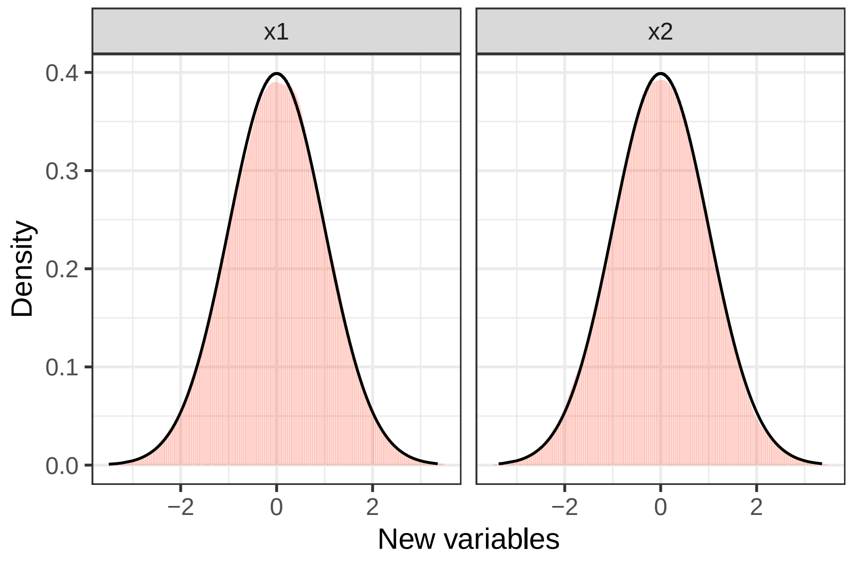
then $X_1, X_2 \sim N(0,1)$.

(a) Write a function which takes as arguments two vectors (U_1, U_2) of uniformly distributed random variables, and returns the two vectors (X_1, X_2) obtained by applying the Box-Muller transform elementwise.

(b) The `R` function runif provides access to a 'pseudo-random number generator' (PRNG), which we'll discuss further during the module. Generate 10,000 $U[0,1]$ random variables using this function, and transform this vector to two vectors, each of 5,000 normal random variates.

(c) Check that the result from (b) is plausibly distributed as pairs of independent, standard normal random variables, by creating a scatter plot of your data.

```
u1 <- runif(5000)
u2 <- runif(5000)

bm <- function(u1, u2) {
  x1 <- sqrt(-2 * log(u1))*cos(2*pi*u2)
  x2 <- sqrt(-2 * log(u2))*cos(2*pi*u1)
  return(data.frame(x1 = x1, x2 = x2))
}

x <- bm(u1, u2)

x %>%
  gather(type, value) %>%
  mutate(dens = dnorm(value)) %>%
  ggplot(aes(x = value)) +
  facet_wrap(~type) +
  geom_histogram(alpha = 0.2,
                 fill = "tomato", stat = "density") +
  geom_line(aes(x = value, y = dens)) +
  xlim(-3.5, 3.5) +
  labs(x = "New variables", y = "Density") +
  theme_bw()
```

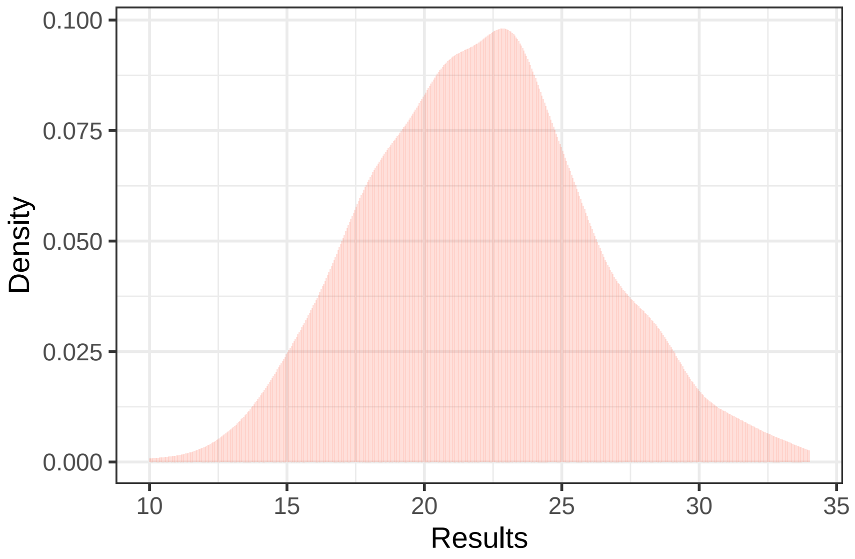


Exercise 2.4 (Simulating Markov chains). Consider a simple board game in which players take it in turns to move around a circular board in which an annulus is divided into 40 segments. Players move by rolling two standard dice and moving their playing piece, clockwise around the annulus, the number of spaces indicated. For simplicity we'll neglect the game's other features.

1. All players begin in a common space. Write R code to simulate a player's position after three moves of the game and repeat this 1,000 or so times. Plot a histogram to show the distribution of player positions after three moves. Is this consistent with your expectations?

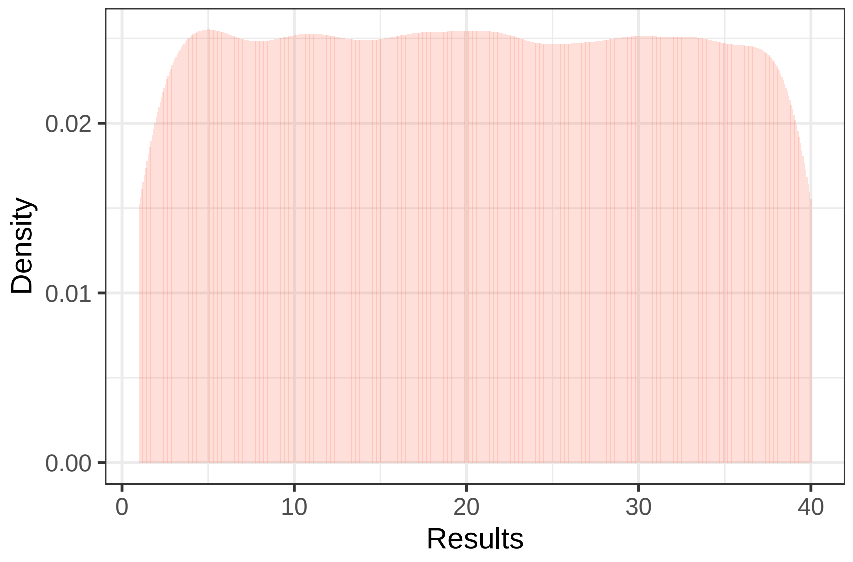
```
numb <- 1:40
dice <-  2

game <- function(samp_player){
  for(i in 1:3){
    add <- replicate(n = dice, sample(vals, size = 1)) %>% sum()
    samp_player <- samp_player + add
    if(samp_player > 40){
      samp_player = samp_player - 40
    }
  }
  return(samp_player)
}
game <- Vectorize(game, "samp_player")
```

```r
#players <- replicate(1000, sample(numb, size = 1))

games <- data.frame(game = game(rep(1, 1000)))

games %>%
  ggplot(aes(x = game)) +
  geom_histogram(alpha = 0.2,
                 fill = "tomato",
                 stat = "density") +
  labs(x = "Results", y = "Density") +
  theme_bw()
```



2. Now modify your code to simulate the sequence of spaces occupied by a player over their first 10,000 moves. Plot a histogram to show the occupancy of each of the forty spaces during these 10,000 moves. Are there any interesting features?

```r
game <- function(samp_player){
  res <- vector(length = 10000)
  res[1] <- samp_player
  for(i in 1:10000){
    add <- replicate(n = dice, sample(vals, size = 1)) %>% sum()
    res[i+1] <- res[i] + add
    if(res[i+1]  > 40){
      res[i+1]  = res[i+1]  - 40
    }
  }
  return(res)
}

games <- data.frame(game = game(1))

games %>%
  ggplot(aes(x = game)) +
  geom_histogram(alpha = 0.2,
                 fill = "tomato",
                 stat = "density") +
  labs(x = "Results", y = "Density") +
  theme_bw()
```



3. If a player's score increases by 1 every time they land on the first space (the starting space), 2 every time they land in the space after that and so on up to 40 for landing in the space immediately before that space then approximately what would be the long-run average number of points per move (use your simulation to obtain an approximate answer).

```r
# ?
2*mean(games$game)
```

```
## [1] 40.94291
```

```r
game <- function(samp_player){
  res <- vector(length = 10000)
  diff <- vector(length = 10000)
  res[1] <- samp_player
  diff[1] <- 0
  for(i in 1:10000){
    add <- replicate(n = dice, sample(vals, size = 1)) %>% sum()
    res[i+1] <- res[i] + add
    diff[i+1] <- (res[i] + add)*2 - res[i]
    if(res[i+1]  > 40){
      res[i+1]  = res[i+1]  - 40
    }
  }
  return(diff)
}

games <- data.frame(game = game(1))
mean(games$game)
```

```
## [1] 34.62694
```

Exercise A.1. What are the maximum likelihood estimators for the parameters in the following situations:
(a)$X_1, \dots, X_n \sim N(\mu, \sigma^2)$, where μ and σ^2 are unknown;

$$f(\mathbf{x}|\mu, \sigma^2) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} exp\{\frac{-(x_i-\mu)^2}{2\sigma^2}\}$$

$$log(f(\mathbf{x}|\mu, \sigma^2)) = n\, log\Big(\frac{1}{\sqrt{2\pi\sigma^2}}\Big) + \sum_{i=1}^{n} \frac{-(x_i-\mu)^2}{2\sigma^2}$$

$$\frac{\partial log(f(\mathbf{x}|\mu, \sigma^2))}{\partial \mu} \approx \sum_{i=1}^{n} -(-(x_i-\mu))$$

$$\hat{\mu} = \frac{\sum_{i=1}^{n} x_i}{n}$$

$$\frac{\partial log(f(\mathbf{x}|\mu, \sigma^2))}{\partial \sigma^2} = -\frac{n}{2\sigma^2} + \sum_{i=1}^{n} \frac{(x_i-\mu)^2}{2(\sigma^2)^2}$$

$$\frac{\partial log(f(\mathbf{x}|\mu, \sigma^2))}{\partial \sigma^2} = -n + \sum_{i=1}^{n} \frac{(x_i-\mu)^2}{\sigma^2}$$

$$\hat{\sigma}^2 = \sum_{i=1}^{n} \frac{-(x_i-\mu)^2}{n}$$

(b) $X_1, \dots, X_n \sim U[0, \theta]$ where θ is unknown.

$$f(\mathbf{x}|\theta) = \prod_{i=1}^{n} \frac{1}{\theta}\, \mathbb{I}[\mathbb{X}_\daleth \in [\not{\vdash}, \theta]]$$

$$log(f(\mathbf{x}|\theta)) = nlog\Big(\frac{1}{\theta}\Big)\, \mathbb{I}[\mathbb{X}_\daleth \in [\not{\vdash}, \theta]]$$

$$\frac{\partial log(f(\mathbf{x}|\theta))}{\theta} = \frac{-n}{\theta} < 0$$

Because of that, $\hat{\theta} = max(X_1, \dots, X_n)$ is the MLE, since the maximum will have to be $\leq \theta$, and it's the closest possible we can get to the true value of the parameter.

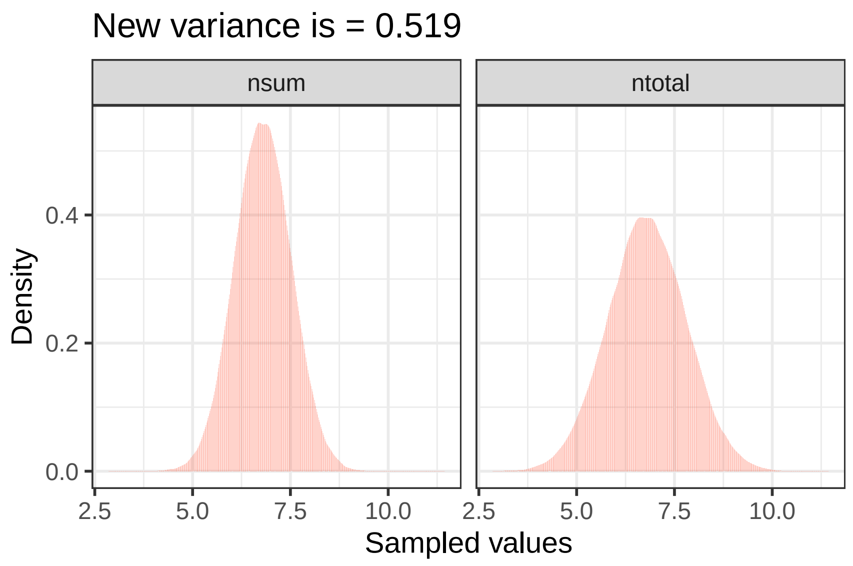
(c) $X_1, \dots, X_n \sim f(x; \mu_1, \mu_2, w)$, where $f(x; \mu_1, \mu_2, w) := wN(x; \mu_1, 1) + (1-w)N(x; \mu_2, 1)$, with $w \in (0, 1)$ and $\mu_1, \mu_2 \in \mathbb{R}$?

We know that

$$wN(x; \mu_1, 1) + (1-w)N(x; \mu_2, 1) \sim N(w\mu_1 + (1-w)\mu_2, w^2\sigma^2 + (1-w)^2\sigma^2),\ \sigma^2 = 1$$

E.g.:

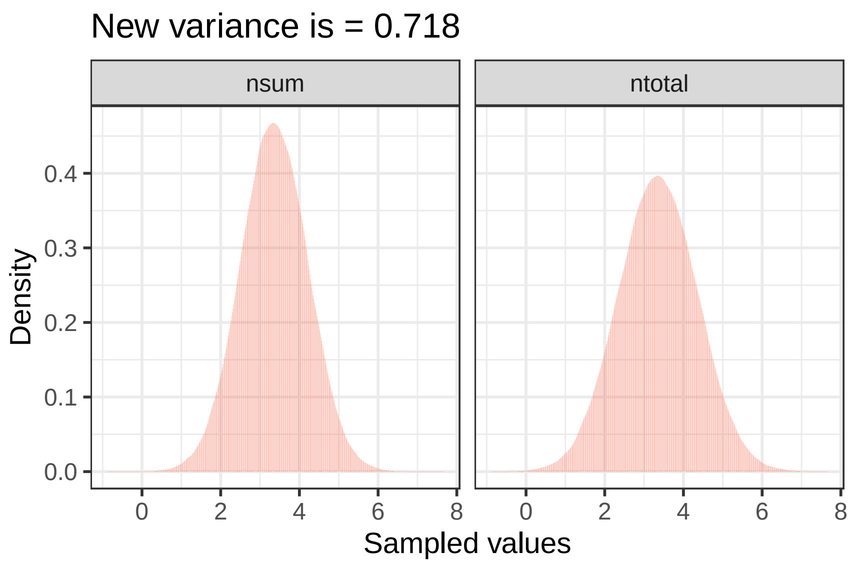
```r
plot_sim <- function(alpha){
  df <- data.frame(
    n1 = rnorm(n = 100000, mean = 2),
    n2 = rnorm(n = 100000, mean = 10),
    ntotal = rnorm(n = 100000, mean = alpha*2 + (1-alpha)*10)) %>%
    mutate(nsum = alpha*n1 + (1-alpha)*n2)

  df %>%
    select(ntotal, nsum) %>%
    gather(type, value) %>%
    ggplot(aes(value)) +
    geom_histogram(alpha = 0.2,
                   fill = "tomato", stat = "density") +
    facet_wrap(~type) +
    labs(title = paste0("New variance is = ",
                        round(var(df$ns), 3))) +
    labs(x = "Sampled values", y = "Density") +
    theme_bw()
}

plot_sim(0.4);
```



```r
(0.6^2)+(0.4^2)
```

```
## [1] 0.52
```

```r
plot_sim(0.83);
```



```r
(0.83^2)+(0.17^2)
```

```
## [1] 0.7178
```

$$log(f(\mathbf{x}|\mu,\sigma^2)) = n\,log\Big(\frac{1}{\sqrt{2\pi(w^2+(1-w)^2)}}\Big) + \sum_{i=1}^{n}\frac{-(x_i - w\mu_1 - (1-w)\mu_2)^2}{2(w^2+(1-w)^2)}$$

$$\frac{\partial log(f(\mathbf{x}|\mu_1,\mu_2,\sigma^2))}{\partial \mu_2} \approx \sum_{i=1}^{n} -(-(\mu_i - \mu_2)(-\mu_1 w + x + (w-1)\mu_2) \approx \sum_{i=1}^{n}(-\mu_1 w + x_+(w-1)\mu_2)$$

$$\hat{\mu}_2 = \sum_{i=1}^{n}\frac{x_i - \mu_1 w}{n(1-w)}$$

$$\hat{\mu}_1 = \sum_{i=1}^{n}\frac{x_i - \mu_2(1-w)}{nw}$$

Exercise A.2. Find the estimators which minimise the posterior expected loss for the following loss functions for continuous parameters:

(a) Squared-error (or quadratic) loss: $C(\theta,\mathbb{V}) = (\theta - \mathbb{V})^2$.

$$\Big[\int_\Theta (\theta-\mathbb{V})^2 f(\theta|x)d\theta\Big]' = -2\int_\Theta(\theta-\mathbb{V})f(\theta|x)d\theta,$$

$$-2\int_\Theta(\theta-\mathbb{V})f(\theta|x)d\theta = 0$$

$$\mathbb{V} = \int_\Theta \theta f(\theta|x)d\theta = \mathbb{E}_{\theta|x}[\theta]$$

(b) Absolute-error loss: $C(\theta, \mathbb{V}) = |\theta - \mathbb{V}|$.

$$\Big[\int_{\Theta} |\theta - \mathbb{V}| f(\theta|x) d\theta\Big]' = \Big[\int_{-\infty}^{\mathbb{V}} (-\theta + \mathbb{V}) f(\theta|x) d\theta + \int_{\mathbb{V}}^{\infty} (\theta - \mathbb{V}) f(\theta|x) d\theta\Big]'$$
$$2\int_{-\infty}^{\mathbb{V}} f(\theta|x) d\theta = \int_{\infty}^{\infty} f(\theta|x) d\theta = 1$$

and $\mathbb{V}$ will be the median of the posterior $f(\theta|x)$.

and for *discrete* parameters:

(c) Squared-error (or quadratic) loss: $C(\theta, \mathbb{V}) = (\theta - \mathbb{V})^2$.

$$\big[\sum^{\Theta} (\theta - \mathbb{V})^2 P(\theta|x) Big]' = -2\sum^{\Theta} (\theta - \mathbb{V}) P(\theta|x) = 0,$$

$$\mathbb{V} = \sum^{\Theta} \theta P(\theta|x) = \mathbb{E}_{P(\theta|x)}[\theta]$$

(d) Zero-one loss:

$$C(\theta, \mathbb{V}) = \begin{cases} 0, & \text{if } \theta = \mathbb{V}, \\ 1, & \text{otherwise.} \end{cases}$$

$$\sum_{\theta \in \Theta \ \ \{\theta^*\}} P(\theta|x) = 1 - P(\theta^*|x)$$

the $\mathbb{V}$ will be the mode θ^*, because that is the most likely value.

Exercise B.1. Given an example of a sequence of random variables which:

(a) converge to a limit with probability one;

If $X_1, \dots, X_n \sim Ber(1/2)$ and $Y_n = 2^n \prod_{i=1}^{n} X_i$, letting $0 < \epsilon < 2^m$,

$$\begin{aligned} P\{|Y_n - 0| < \epsilon \text{ for all } n \geq m\} &= P\{X_n = 0 \text{ for some } n \leq m\} \\ &= 1 - P\{X_n = 1 \text{ for all } n \geq m\} \\ &= 1 - (1/2)^m = 1, \text{ when } n \to \infty \end{aligned}$$

(b) converge to a limit in probability but not almost surely; and

If $X_1, \dots, X_n \sim Exp(n)$, the sequence

$$\begin{aligned} \lim_{n\to\infty} P\{|X_n - 0| \geq \epsilon\} &= \lim_{n\to\infty} P\{X_n \geq \epsilon\} \\ &= \lim_{n\to\infty} e^{-n\epsilon} = 0, \text{ for all } \epsilon. \end{aligned}$$

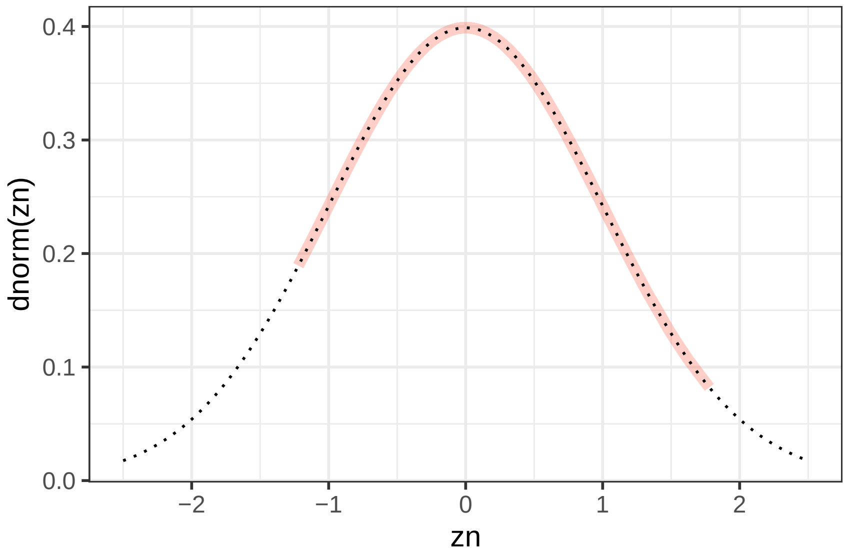
(c) converge to a limit in distribution but not in probability.

CLT - Let $X_1, \dots, X_n \sim U[-1,1]$ be i.i.d. with finite mean $E(X)$ and some variance σ_x^2. The normalized sum

$$Z_n = \frac{\sum_{i=1}^n X_i}{\sqrt{n/3}}$$

will quickly approach a Gaussian distribution.

```r
n <- 10000
unif <- data.frame(unif_val = runif(n, min = -1, max = 1)) %>%
  mutate(
    ind = rep(1, n),
    cs = cumsum(unif_val),
    ns = cumsum(ind),
    zn = cs/(sqrt(ns/3)))

unif %>%
  ggplot(aes(zn, dnorm(zn))) +
  geom_line(colour = "tomato", alpha = 0.3, size = 2) +
  stat_function(fun = dnorm, args = list(mean = 0, 1),
                linetype = 3)+
  xlim(-2.5, 2.5) +
  theme_bw()
```


dnorm(zn)
zn