Mini-Project-1

Tanner Bessette

2025-01-29

I have followed all rules for collaboration for this project, and I have not used generative AI on this project.

Normal Distribution

Sampling Distribution of the Sample Minimum:

```
n <- 5         # sample size
mu <- 10       # population mean
sigma <- 2     # population standard deviation

# generate a random sample of n observations from a normal population
single_sample <- rnorm(n, mu, sigma) |> round(2)
# look at the sample
single_sample
```

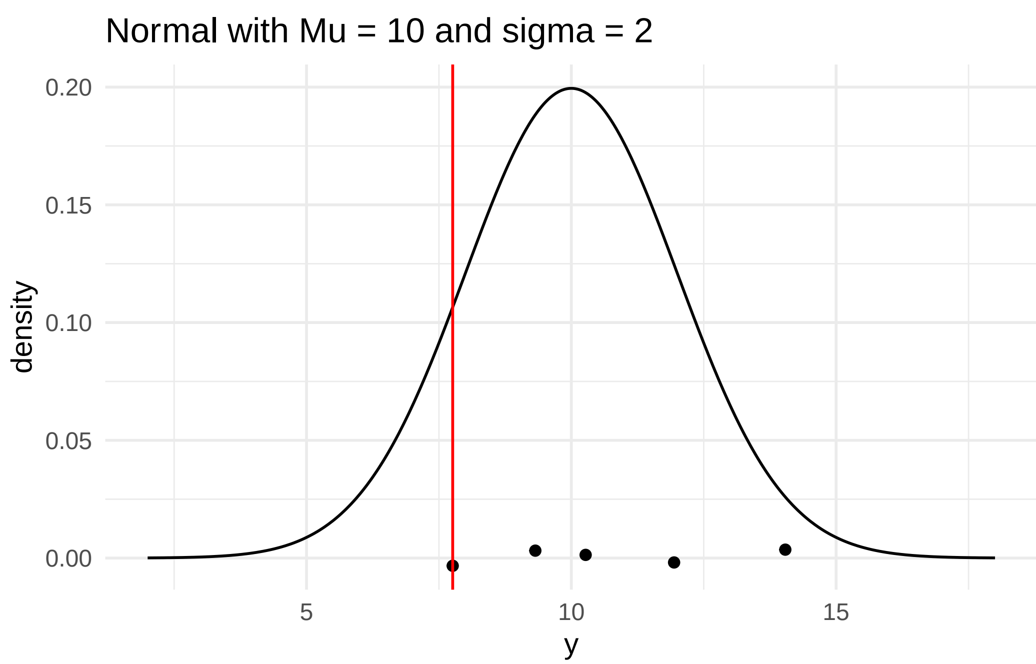
```
[1] 14.04 11.94 10.27  7.76  9.32
```

```
# compute the sample mean
sample_min <- min(single_sample)
# look at the sample mean
sample_min
```

```
[1] 7.76
```

```
# generate a range of values that span the population
plot_df <- tibble(xvals = seq(mu - 4 * sigma, mu + 4 * sigma, length.out = 500)) |>
```

```r
  mutate(xvals_density = dnorm(xvals, mu, sigma))

## plot the population model density curve
ggplot(data = plot_df, aes(x = xvals, y = xvals_density)) +
  geom_line() +
  theme_minimal() +
  ## add the sample points from your sample
  geom_jitter(data = tibble(single_sample), aes(x = single_sample, y = 0),
              width = 0, height = 0.005) +
  ## add a line for the sample mean
  geom_vline(xintercept = sample_min, colour = "red") +
  labs(x = "y", y = "density",
       title = "Normal with Mu = 10 and sigma = 2")
```




```r
n <- 5            # sample size
mu <- 10          # population mean
sigma <- 2        # population standard deviation

generate_samp_min <- function(mu, sigma, n) {

  single_sample <- rnorm(n, mu, sigma)
```

```r
  sample_min <- min(single_sample)

  return(sample_min)
}

## test function once:
generate_samp_min(mu = mu, sigma = sigma, n = n)
```

```
[1] 7.06199
```

```r
nsim <- 5000      # number of simulations

## map through th function -- the \(i) syntax says to just
## repeat the generate_samp_mean function nsim times
mins <- map_dbl(1:nsim, \(i) generate_samp_min(mu = mu, sigma = sigma, n = n))

# Calculate E(Ymin) and SE(Ymin)
E_ymin <- mean(mins)
SE_ymin <- sd(mins)
E_ymin
```

```
[1] 7.66156
```

```r
SE_ymin
```

```
[1] 1.35776
```

```r
## print some of the 5000 means
## each number represents the sample mean from __one__ sample.
norm_mins_df <- tibble(mins, pop = "Normal(10, 4)")
norm_mins_df
```

```
# A tibble: 5,000 x 2
   mins pop
  <dbl> <chr>
1  7.73 Normal(10, 4)
2  7.31 Normal(10, 4)
```

```
 3  7.13 Normal(10, 4)
 4  8.41 Normal(10, 4)
 5  9.24 Normal(10, 4)
 6  9.64 Normal(10, 4)
 7  8.61 Normal(10, 4)
 8  4.02 Normal(10, 4)
 9  9.07 Normal(10, 4)
10  7.51 Normal(10, 4)
# i 4,990 more rows
```

```r
Normal_Min_Hisotgram <-
  ggplot(data = norm_mins_df, aes(x = mins)) +
  geom_histogram(colour = "deeppink4", fill = "deeppink1",
                 bins = 20) +
  theme_minimal() +
  labs(x = "Observed Sample Means",
       title = paste("Sampling Distribution of the
                     Sample Min"))
```

$E(Y_{min}) = 7.6848$

$SE(Y_{min}) = 1.3464$

Sampling distribution of the sample maximum:

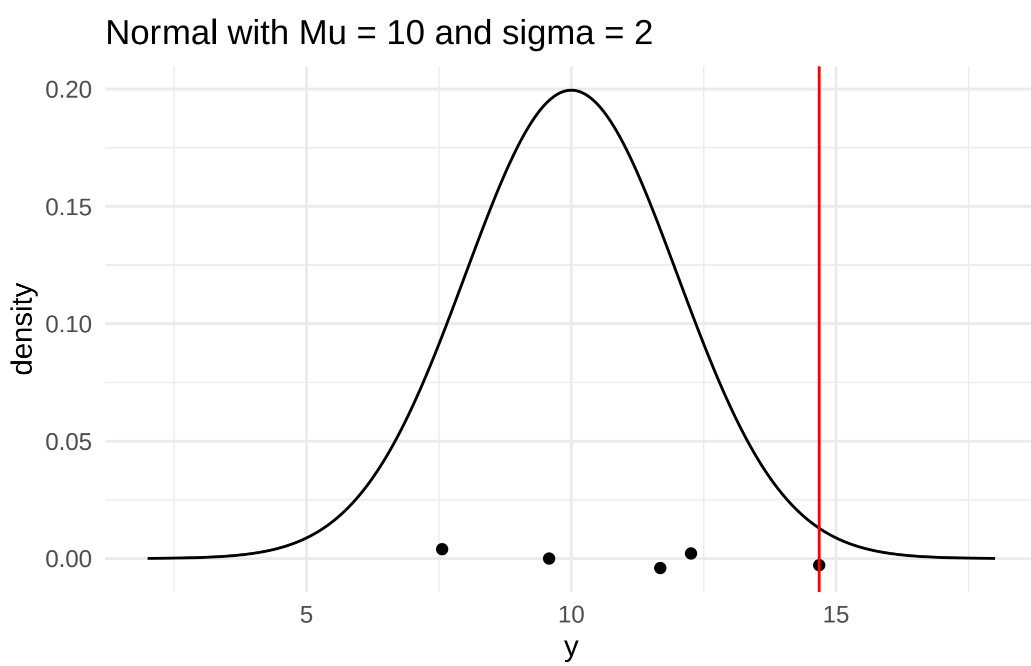
```r
n <- 5        # sample size
mu <- 10      # population mean
sigma <- 2    # population standard deviation

# generate a random sample of n observations from a normal population
single_sample <- rnorm(n, mu, sigma) |> round(2)
# look at the sample
single_sample
```

```
[1] 14.68  7.56  9.58 12.26 11.68
```

```r
# compute the sample mean
sample_max <- max(single_sample)
# look at the sample mean
sample_max
```

```
[1] 14.68
```

```
# generate a range of values that span the population
plot_df <- tibble(xvals = seq(mu - 4 * sigma, mu + 4 * sigma, length.out = 500)) |>
  mutate(xvals_density = dnorm(xvals, mu, sigma))

## plot the population model density curve
ggplot(data = plot_df, aes(x = xvals, y = xvals_density)) +
  geom_line() +
  theme_minimal() +
  ## add the sample points from your sample
  geom_jitter(data = tibble(single_sample), aes(x = single_sample, y = 0),
              width = 0, height = 0.005) +
  ## add a line for the sample mean
  geom_vline(xintercept = sample_max, colour = "red") +
  labs(x = "y", y = "density",
       title = "Normal with Mu = 10 and sigma = 2")
```




```
n <- 5             # sample size
mu <- 10           # population mean
sigma <- 2         # population standard deviation
```

```r
generate_samp_max <- function(mu, sigma, n) {

  single_sample <- rnorm(n, mu, sigma)
  sample_max <- max(single_sample)

  return(sample_max)
}

## test function once:
generate_samp_max(mu = mu, sigma = sigma, n = n)
```

```
[1] 12.23739
```

```r
nsim <- 5000      # number of simulations

## map through th function -- the \(i) syntax says to just
## repeat the generate_samp_mean function nsim times
maxes <- map_dbl(1:nsim, \(i) generate_samp_max(mu = mu, sigma = sigma, n = n))

# Calculate E(Ymax) and SE(Ymax)
E_ymax <- mean(maxes)
SE_ymax <- sd(maxes)
E_ymax
```

```
[1] 12.33514
```

```r
SE_ymax
```

```
[1] 1.356834
```

```r
## print some of the 5000 means
## each number represents the sample mean from __one__ sample.
norm_maxes_df <- tibble(maxes, pop = "Normal(10, 4)")
norm_maxes_df
```

```
# A tibble: 5,000 x 2
   maxes pop
```

```
   <dbl> <chr>
 1  13.0 Normal(10, 4)
 2  11.7 Normal(10, 4)
 3  12.1 Normal(10, 4)
 4  10.9 Normal(10, 4)
 5  10.3 Normal(10, 4)
 6  13.0 Normal(10, 4)
 7  13.1 Normal(10, 4)
 8  13.1 Normal(10, 4)
 9  11.2 Normal(10, 4)
10  11.6 Normal(10, 4)
# i 4,990 more rows
```

```r
Normal_Max_Histogram <-
  ggplot(data = norm_maxes_df, aes(x = maxes)) +
  geom_histogram(colour = "deeppink4", fill = "deeppink1",
                 bins = 20) +
  theme_minimal() +
  labs(x = "Observed Sample Means",
       title = "Sampling Distribution of
                the Sample Max")
```

$E(Y_{max}) = 12.3333$

$SE(Y_{max}) = 1.35581$

Uniform Distribution

Sampling distribution of the sample minimum:

```
n <- 5              # sample size
theta_1 <- 7
theta_2 <- 13

generate_samp_min <- function(theta_1, theta_2, n) {

  single_sample <- runif(n, theta_1, theta_2)
  sample_min <- min(single_sample)

  return(sample_min)
}

## test function once:
generate_samp_min(theta_1 = theta_1, theta_2 = theta_2, n = n)
```

```
[1] 8.205544
```

```
nsim <- 5000        # number of simulations

## map through th function -- the \(i) syntax says to just
## repeat the generate_samp_mean function nsim times
mins <- map_dbl(1:nsim, \(i) generate_samp_min(theta_1 = theta_1, theta_2 = theta_2, n = n

# Calculate E(Ymin) and SE(Ymin)
E_ymin_unif <- mean(mins)
SE_ymin_unif <- sd(mins)
E_ymin_unif
```

```
[1] 7.996323
```

```
SE_ymin_unif
```

```
[1] 0.8478432
```

```r
## print some of the 5000 means
## each number represents the sample mean from __one__ sample.
unif_mins_df <- tibble(mins, pop = "Uniform(7, 13)")
unif_mins_df
```

```
# A tibble: 5,000 x 2
    mins pop
   <dbl> <chr>
 1  8.86 Uniform(7, 13)
 2  8.35 Uniform(7, 13)
 3  7.38 Uniform(7, 13)
 4  8.00 Uniform(7, 13)
 5  8.15 Uniform(7, 13)
 6  7.99 Uniform(7, 13)
 7  9.42 Uniform(7, 13)
 8  8.29 Uniform(7, 13)
 9  9.78 Uniform(7, 13)
10  7.05 Uniform(7, 13)
# i 4,990 more rows
```

```r
Unif_Min_Histogram <- ggplot(data = unif_mins_df, aes(x = mins)) +
  geom_histogram(colour = "deeppink4", fill = "deeppink1",
                 bins = 20) +
  theme_minimal() +
  labs(x = "Observed Sample Means",
       title = paste("Sampling Distribution of the
                      Sample Min"))
```

$E(Y_{min}) = 8.0031$

$SE(Y_{min}) = 0.8446$

Sampling distribution of the sample maximum:

```r
n <- 5            # sample size
theta_1 <- 7
theta_2 <- 13

generate_samp_max <- function(theta_1, theta_2, n) {

  single_sample <- runif(n, theta_1, theta_2)
  sample_max <- max(single_sample)
```

```
  return(sample_max)
}

## test function once:
generate_samp_max(theta_1 = theta_1, theta_2 = theta_2, n = n)
```

```
[1] 12.41057
```

```
nsim <- 5000      # number of simulations

## map through th function -- the \(i) syntax says to just
## repeat the generate_samp_mean function nsim times
maxes <- map_dbl(1:nsim, \(i) generate_samp_max(theta_1 = theta_1, theta_2 = theta_2, n =

# Calculate E(Ymax) and SE(Ymax)
E_ymax_unif <- mean(maxes)
SE_ymax_unif <- sd(maxes)
E_ymax_unif
```

```
[1] 11.99878
```

```
SE_ymax_unif
```

```
[1] 0.8418832
```

```
## print some of the 5000 means
## each number represents the sample mean from __one__ sample.
unif_maxes_df <- tibble(maxes, pop = "Uniform(7, 13)")
unif_maxes_df
```

```
# A tibble: 5,000 x 2
   maxes pop
   <dbl> <chr>
 1 11.8  Uniform(7, 13)
 2 12.0  Uniform(7, 13)
 3  9.69 Uniform(7, 13)
 4 12.9  Uniform(7, 13)
```

```
 5 12.9  Uniform(7, 13)
 6 10.3  Uniform(7, 13)
 7 12.6  Uniform(7, 13)
 8 12.3  Uniform(7, 13)
 9 12.8  Uniform(7, 13)
10 12.1  Uniform(7, 13)
# i 4,990 more rows
```

```r
Unif_Max_Histogram <- ggplot(data = unif_maxes_df, aes(x = maxes)) +
  geom_histogram(colour = "deeppink4", fill = "deeppink1",
                 bins = 20) +
  theme_minimal() +
  labs(x = "Observed Sample Means",
       title = paste("Samp. Dist. of Samp. Max.
                     Uniform(7, 13)"))
```

$E(Y_{max}) = 11.9965$

$SE(Y_{max}) = 0.8471$

Exponential Distribution

Sampling distribution of the sample minimum:

```r
n <- 5            # sample size
lambda <- 0.5

generate_samp_min <- function(lambda, n) {

  single_sample <- rexp(n, lambda)
  sample_min <- min(single_sample)

  return(sample_min)
}

## test function once:
generate_samp_min(lambda = lambda, n = n)
```

```
[1] 0.0800897
```

```r
nsim <- 5000      # number of simulations

## map through th function -- the \(i) syntax says to just
## repeat the generate_samp_mean function nsim times
mins <- map_dbl(1:nsim, \(i) generate_samp_min(
  lambda = lambda, n = n))

# Calculate E(Ymin) and SE(Ymin)
E_ymin_exp <- mean(mins)
SE_ymin_exp <- sd(mins)
E_ymin_exp
```

```
[1] 0.4015463
```

```r
SE_ymin_exp
```

```
[1] 0.4001794
```

```r
## print some of the 5000 means
## each number represents the sample mean from __one__ sample.
exp_mins_df <- tibble(mins, pop = "Exp(0.5)")
exp_mins_df
```

```
# A tibble: 5,000 x 2
     mins pop
    <dbl> <chr>
 1 0.364  Exp(0.5)
 2 0.179  Exp(0.5)
 3 2.32   Exp(0.5)
 4 0.0792 Exp(0.5)
 5 0.365  Exp(0.5)
 6 0.0919 Exp(0.5)
 7 0.0650 Exp(0.5)
 8 0.225  Exp(0.5)
 9 0.217  Exp(0.5)
10 0.256  Exp(0.5)
# i 4,990 more rows
```

```r
Exp_Min_Histogram <- ggplot(data = exp_mins_df,
                            aes(x = mins)) +
  geom_histogram(colour = "deeppink4", fill = "deeppink1",
                 bins = 20) +
  theme_minimal() +
  labs(x = "Observed Sample Means",
       title = "Samp. Dist. of Samp. Min. Exp(0.5)")
```

$E(Y_{min}) = 0.3955$

$SE(Y_{min}) = 0.3922$

Sampling distribution of the sample maximum:

```r
n <- 5            # sample size
lambda <- 0.5

generate_samp_max <- function(lambda, n) {

  single_sample <- rexp(n, lambda)
  sample_max <- max(single_sample)

```

```r
  return(sample_max)
}

## test function once:
generate_samp_max(lambda = lambda, n = n)
```

```
[1] 4.098288
```

```r
nsim <- 5000      # number of simulations

## map through th function -- the \(i) syntax says to just
## repeat the generate_samp_mean function nsim times
maxes <- map_dbl(1:nsim, \(i) generate_samp_max(
  lambda = lambda, n = n))

# Calculate E(Ymax) and SE(Ymax)
E_ymax_exp <- mean(maxes)
SE_ymax_exp <- sd(maxes)
E_ymax_exp
```

```
[1] 4.599737
```

```r
SE_ymax_exp
```

```
[1] 2.471394
```

```r
## print some of the 5000 means
## each number represents the sample mean from __one__ sample.
exp_maxes_df <- tibble(maxes, pop = "Exp(0.5)")
exp_maxes_df
```

```
# A tibble: 5,000 x 2
   maxes pop
   <dbl> <chr>
 1  4.50 Exp(0.5)
 2  2.86 Exp(0.5)
 3  1.89 Exp(0.5)
```

```
 4  7.00 Exp(0.5)
 5  3.26 Exp(0.5)
 6  2.99 Exp(0.5)
 7  5.17 Exp(0.5)
 8  5.84 Exp(0.5)
 9  2.58 Exp(0.5)
10  6.33 Exp(0.5)
# i 4,990 more rows
```

```r
Exp_Max_Histogram <- ggplot(data = exp_maxes_df,
                            aes(x = maxes)) +
  geom_histogram(colour = "deeppink4", fill = "deeppink1",
                 bins = 20) +
  theme_minimal() +
  labs(x = "Observed Sample Means",
       title = "Samp. Dist. of Samp. Max. Exp(0.5)")
```

$E(Y_{max}) = 4.5812$

$SE(Y_{max}) = 2.4562$

Beta Distribution

Sampling distribution of the sample minimum:

```
n <- 5              # sample size
alpha <- 8
beta <- 2

generate_samp_min <- function(alpha, beta, n) {

  single_sample <- rbeta(n, alpha, beta)
  sample_min <- min(single_sample)

  return(sample_min)
}

## test function once:
generate_samp_min(alpha = alpha, beta = beta, n = n)
```

```
[1] 0.650141
```

```
nsim <- 5000       # number of simulations

## map through th function -- the \(i) syntax says to just
## repeat the generate_samp_mean function nsim times
mins <- map_dbl(1:nsim, \(i) generate_samp_min(
  alpha = alpha, beta = beta, n = n))

# Calculate E(Ymin) and SE(Ymin)
E_ymin_beta <- mean(mins)
SE_ymin_beta <- sd(mins)
E_ymin_beta
```

```
[1] 0.6469552
```

```
SE_ymin_beta
```

```
[1] 0.1064372
```

```r
## print some of the 5000 means
## each number represents the sample mean from __one__ sample.
beta_mins_df <- tibble(mins, pop = "Beta(8, 2)")
beta_mins_df
```

```
# A tibble: 5,000 x 2
    mins pop
   <dbl> <chr>
 1 0.860 Beta(8, 2)
 2 0.731 Beta(8, 2)
 3 0.446 Beta(8, 2)
 4 0.546 Beta(8, 2)
 5 0.778 Beta(8, 2)
 6 0.731 Beta(8, 2)
 7 0.576 Beta(8, 2)
 8 0.462 Beta(8, 2)
 9 0.749 Beta(8, 2)
10 0.689 Beta(8, 2)
# i 4,990 more rows
```

```r
Beta_Min_Histogram <- ggplot(data = beta_mins_df,
                             aes(x = mins)) +
  geom_histogram(colour = "deeppink4", fill = "deeppink1",
                 bins = 20) +
  theme_minimal() +
  labs(x = "Observed Sample Means",
       title = "Samp. Dist. of Samp. Min. Beta(8, 2)")
```

$E(Y_{min}) = 0.6471$

$SE(Y_{min}) = 0.1079$

Sampling distribution of the sample maximum:

```r
n <- 5           # sample size
alpha <- 8
beta <- 2

generate_samp_max <- function(alpha, beta, n) {

  single_sample <- rbeta(n, alpha, beta)
  sample_max <- max(single_sample)
```

```r
  return(sample_max)
}

## test function once:
generate_samp_max(alpha = alpha, beta = beta, n = n)
```

```
[1] 0.9628768
```

```r
nsim <- 5000      # number of simulations

## map through th function -- the \(i) syntax says to just
## repeat the generate_samp_mean function nsim times
maxes <- map_dbl(1:nsim, \(i) generate_samp_max(
  alpha = alpha, beta = beta, n = n))

# Calculate E(Ymax) and SE(Ymax)
E_ymax_beta <- mean(maxes)
SE_ymax_beta <- sd(maxes)
E_ymax_beta
```

```
[1] 0.9220068
```

```r
SE_ymax_beta
```

```
[1] 0.04533354
```

```r
## print some of the 5000 means
## each number represents the sample mean from __one__ sample.
beta_maxes_df <- tibble(maxes, pop = "Beta(8, 2)")
beta_maxes_df
```

```
# A tibble: 5,000 x 2
   maxes pop
   <dbl> <chr>
 1 0.944 Beta(8, 2)
 2 0.887 Beta(8, 2)
 3 0.972 Beta(8, 2)
```

```
 4 0.859 Beta(8, 2)
 5 0.854 Beta(8, 2)
 6 0.965 Beta(8, 2)
 7 0.882 Beta(8, 2)
 8 0.909 Beta(8, 2)
 9 0.953 Beta(8, 2)
10 0.968 Beta(8, 2)
# i 4,990 more rows
```

```r
Beta_Max_Histogram <- ggplot(data = beta_maxes_df,
                             aes(x = maxes)) +
  geom_histogram(colour = "deeppink4", fill = "deeppink1",
                 bins = 20) +
  theme_minimal() +
  labs(x = "Observed Sample Means",
       title = "Samp. Dist. of Samp. Max. Beta(8, 2)")
```

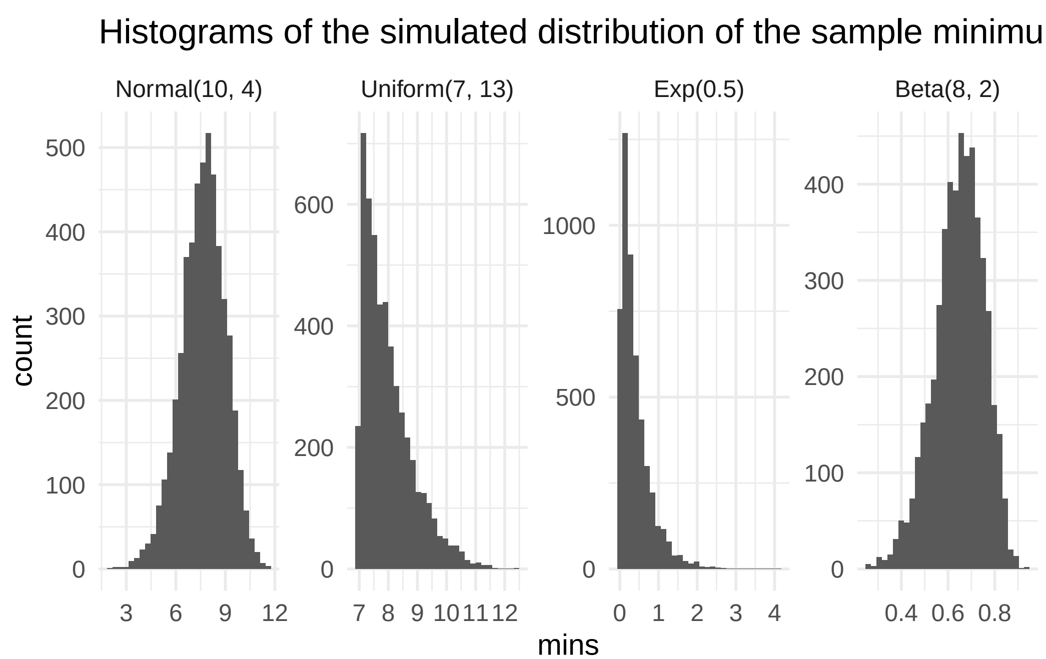
$E(Y_{max}) = 0.9219$

$SE(Y_{max}) = 0.0464$

Histograms of the simulated distribution of the sample minimum

```
pop_plot <- bind_rows(norm_mins_df, unif_mins_df,
                      exp_mins_df, beta_mins_df) |>
  mutate(pop = fct_relevel(pop,
                           c("Normal(10, 4)", "Uniform(7, 13)",
                             "Exp(0.5)", "Beta(8, 2)")))

ggplot(data = pop_plot, aes(x = mins)) +
  geom_histogram() +
  theme_minimal() +
  facet_wrap(~ pop, nrow = 1, scales = "free") +
  labs(title = "Histograms of the simulated distribution of the sample minimum")
```

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



Histograms of the simulated distribution of the sample maximum

```
pop_plot_2 <- bind_rows(norm_maxes_df, unif_maxes_df,
                        exp_maxes_df, beta_maxes_df) |>
  mutate(pop = fct_relevel(pop,
                           c("Normal(10, 4)", "Uniform(7, 13)",
                                    "Exp(0.5)", "Beta(8, 2)")))

ggplot(data = pop_plot_2, aes(x = maxes)) +
  geom_histogram() +
  theme_minimal() +
  facet_wrap(~ pop, nrow = 1, scales = "free") +
  labs(title = "Histograms of the simulated distribution of the sample minimum")
```

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

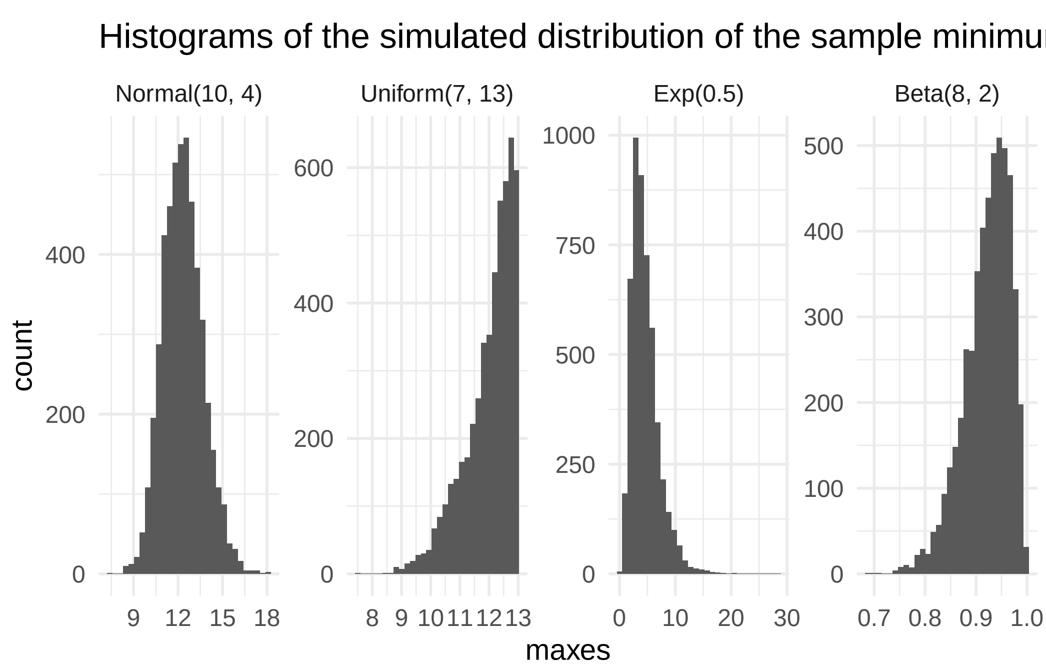


Table of Results

Table 1: Table of Results

	N($\mu = 10, \sigma^2 = 4$)	Unif($\theta_1 = 7, \theta_2 = 13$)	Exp($\lambda = 0.5$)	Beta($\alpha = 8, \beta = 2$)
E(Y_{min})	7.6848	8.0031	0.3955	0.6471
E(Y_{max})	12.3333	11.9965	4.5812	0.9219
SE(Y_{min})	1.3464	0.8446	0.3922	0.1079
SE(Y_{max})	1.3558	0.8471	2.4562	0.0464

Questions

Question 1: For the normal distribution and the uniform distribution, $SE(Y_{min})$ and $SE(Y_{max})$ are nearly identical, and I think that we can make the result that we expect $SE(Y_{min})$ and $SE(Y_{max})$ to be the same for Normal and Uniform distributions. However, for the Exponential and the Beta distributions, a rule appears less clear-cut to make. It makes sense to make a general rule that we expect $SE(Y_{max})$ to be significantly higher than $SE(Y_{min})$ for the Exponential distribution, and we expect $SE(Y_{min})$ to be significantly higher than $SE(Y_{max})$ for the Beta distribution.

Question 2: (Choosing the third option, exponential)

Note: I looked up how to calculate integrals in R so I could keep the work in the same file.

Calculate the pdfs:

Y_{min} pdf: $= n(1 - F(y))^{n-1} * f(y)$

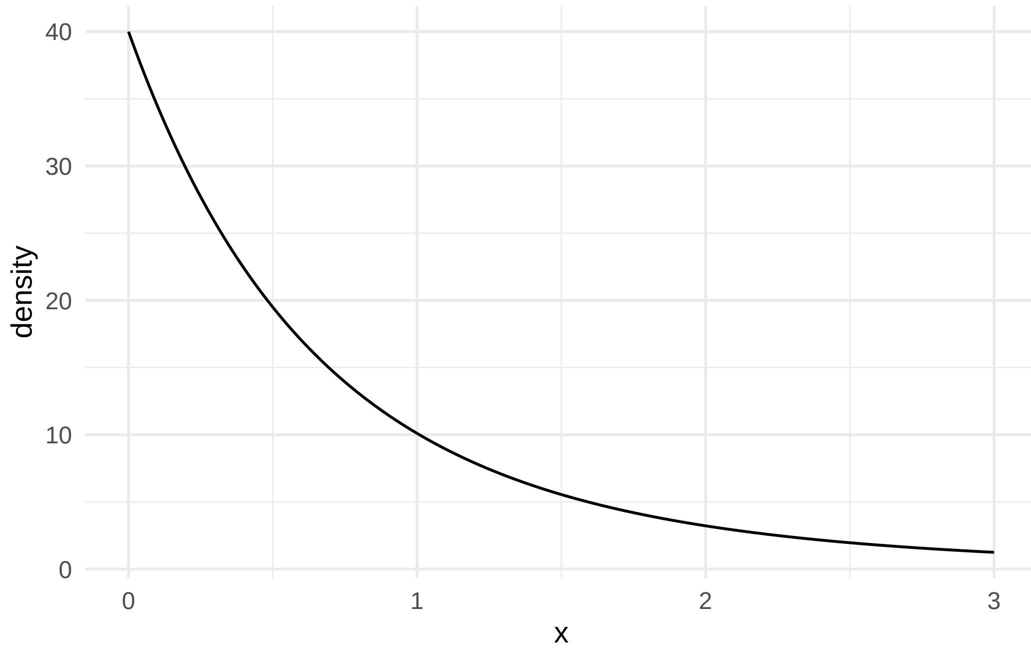
$= n(1 + e^{-\lambda * y})^{n-1} * \lambda * e^{-\lambda * y}$

Y_{max} pdf: $= n(-e^{-\lambda * y})^{n-1} * \lambda * e^{-\lambda * y}$

Plot the pdfs in R:

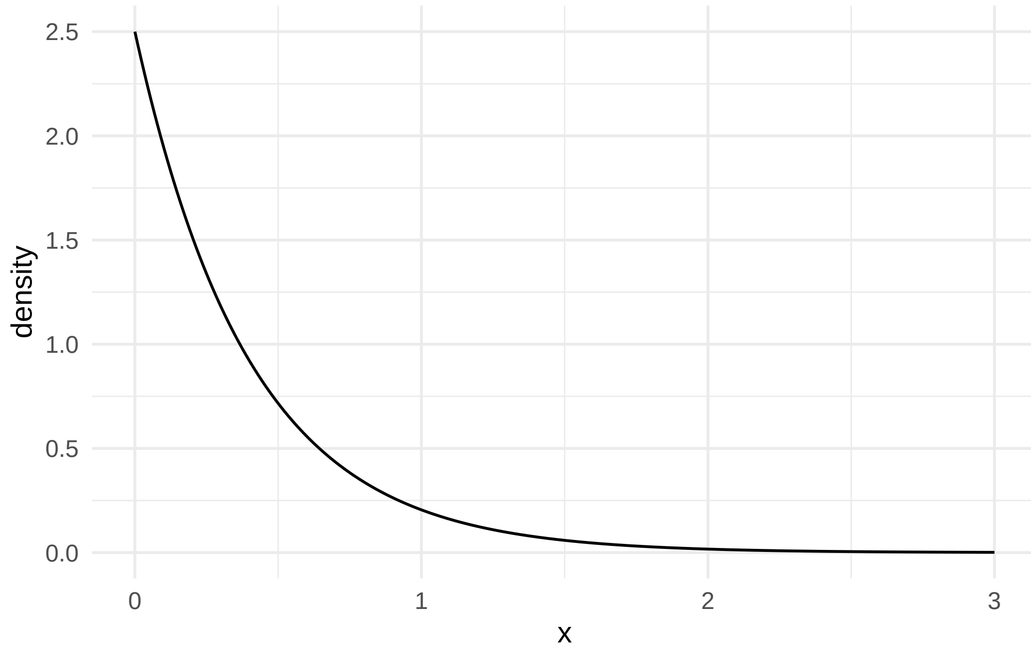
pdf of Y_{min}:

```
n <- 5
## CHANGE 0 and 3 to represent where you want your graph to start and end
## on the x-axis
x <- seq(0, 3, length.out = 1000)
## CHANGE to be the pdf you calculated. Note that, as of now,
## this is not a proper density (it does not integrate to 1).
density <-
  n * (1 + exp(-lambda * x))^(n - 1) * lambda * exp(-lambda * x)

## put into tibble and plot
samp_min_df <- tibble(x, density)
ggplot(data = samp_min_df, aes(x = x, y = density)) +
  geom_line() +
  theme_minimal()
```



pdf of Y_{max}:

```
n <- 5
## CHANGE 0 and 3 to represent where you want your graph to start and end
## on the x-axis
x <- seq(0, 3, length.out = 1000)
## CHANGE to be the pdf you calculated. Note that, as of now,
## this is not a proper density (it does not integrate to 1).
density <- n * (-exp(-lambda * x))^(n-1) * lambda * exp(-lambda * x)

## put into tibble and plot
samp_min_df <- tibble(x, density)
ggplot(data = samp_min_df, aes(x = x, y = density)) +
  geom_line() +
  theme_minimal()
```



Use integrals to calculate expected value and standard error for Y_{min}:

```
function_ymin <- function(x) {
  x * n * (1 - exp(-lambda * x))^(n - 1) * lambda * exp(-lambda * x)
}

E_ymin <- integrate(function_ymin, lower = 0, upper = Inf)
```

Calculate expected value and standard error for Y_{max}:

```
function_ymax <- function(x) {
  x * n * (-exp(-lambda * x))^(n-1) * lambda * exp(-lambda * x)
}

E_ymax = integrate(function_ymax, lower = 0, upper = Inf)
```

My answers are essentially swapped, with my E(Ymin) being what my E(Ymax) was in the table. I am assuming I accidentally swapped something somewhere that I can't find, so I believe that my answers are nearly equal to the simulated answers, but are slightly off. (roughly just 0.01 off from my simulated expected values).

Calculate standard errors using integrals:

Y_{min} se:

```
function_ymin_se <- function(x) {
  x^2 * n * (1 - exp(-lambda * x))^(n - 1) * lambda * exp(-lambda * x)
}

E_ymin2 <- integrate(function_ymin_se, lower = 0, upper = Inf)

# E_ymin = 4.56667
# E(ymin^2) = 26.7089
se_ymin <- sqrt(26.7089 - 4.5667^2)
se_ymin # = 2.4195
```

```
[1] 2.419535
```

Y_{max} se:

```
function_ymax_se <- function(x) {
  x^2 * n * (-exp(-lambda * x))^(n-1) * lambda * exp(-lambda * x)
}

E_ymax2 = integrate(function_ymax_se, lower = 0, upper = Inf)

# E_ymax = 0.4
# E(ymax^2) = 0.32
se_ymax <- sqrt(0.32 - 0.32^2)
se_ymax # = 0.4665
```

```
[1] 0.4664762
```

Both of the SE's are also similar to what we calculated in the simulation, but this time a little farther off from the results from the simulation than it was with the expected values. (About ~0.15 and ~ 0.07 off).