HW1

2023-10-02

Quesiton 3

```
# x ~ iid N(0,2)

# Set n's
n = c(10, 100, 1000)

# Set number of simulations
sims = 10000

# Create a bunch of sample means
xbars = sapply(1:sims, function(sims){
  # Simulate x's for each value of n
  x = sapply(1:length(n), function(i){
    rnorm(n[i], 0, 2)
  })

  # Calculate sample means for each n
  xbar = sapply(1:length(n), function(i){
    1/n[i]*sum(x[[i]])
  })
})

# flip dimensions of xbars to make it more intuitive
xbars = t(xbars)

# Name columns for pretty output
colnames(xbars) = c("10", "100", "1000")
```

```
# Calculate means for each n
x_means = colMeans(xbars)

# Calculate variances for each n
x_vars = sapply(1:length(n), function(i){var(xbars[,i])})
names(x_vars) = c("10", "100", "1000")

# Make the output nice
(samp_mean_normal_df = data.frame(x_means, x_vars))
```

Part a)

```
##            x_means      x_vars
## 10   -0.0098341845 0.399823582
## 100   0.0019700092 0.039568019
## 1000 -0.0002792192 0.003943637
```

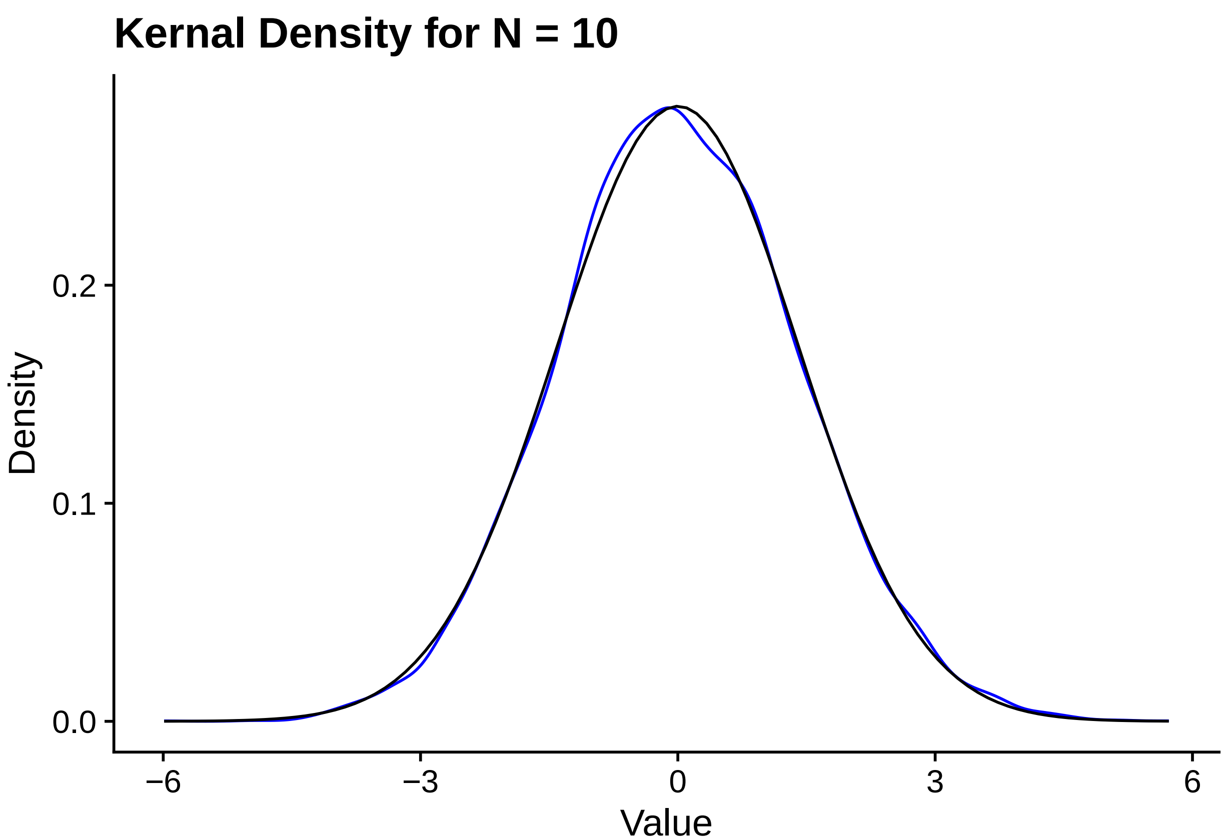
```
# Calculate which simulations have absolute value greater than 0.1
xbars_large = abs(xbars) > 0.1

# Calculate the percentage for each n
large_percentage = sapply(1:length(n), function(i){
  percentage = sum(xbars_large[,i]) / nrow(xbars_large)
})

names(large_percentage) = c("10", "100", "1000")
large_percentage
```

Part b)

```
##     10    100   1000
## 0.8752 0.6203 0.1122
```

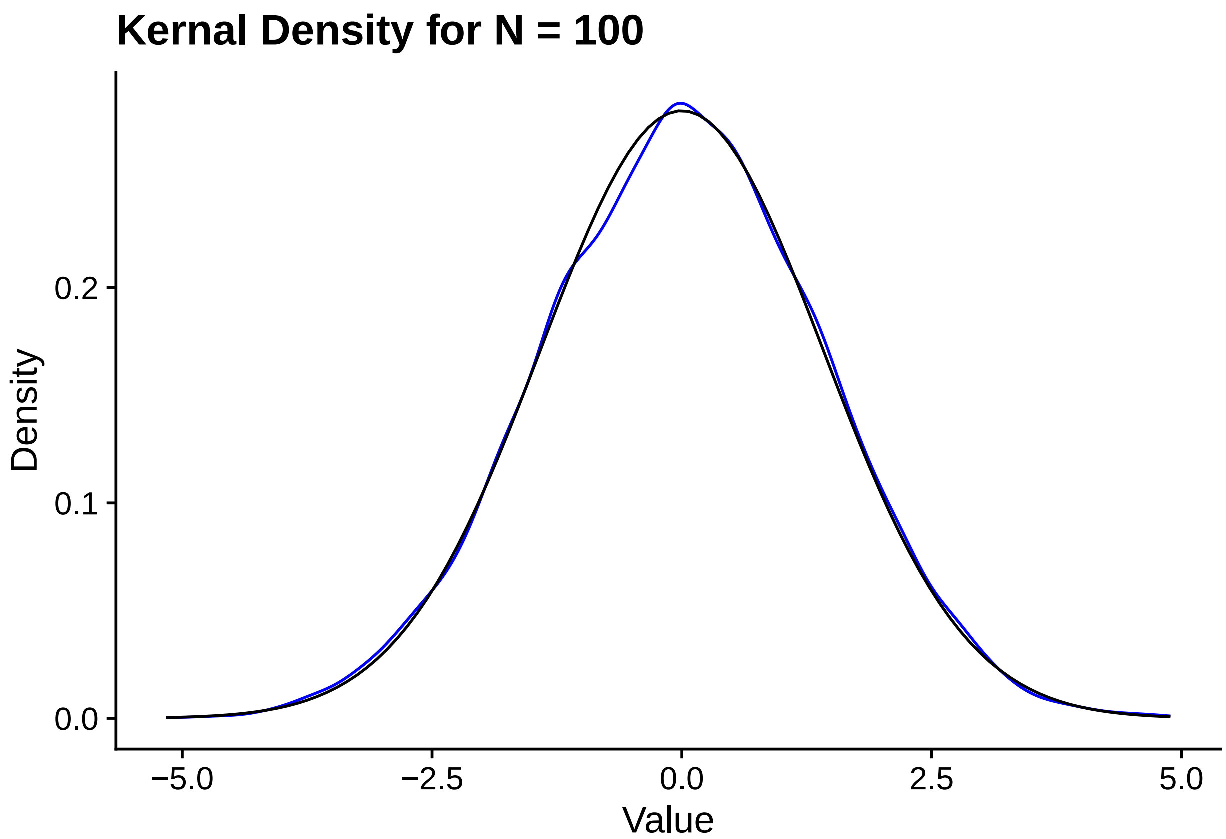
```
# Same as earlier part but instead of sample mean its dividing by sqrt(n)
xbars_sqrt = sapply(1:sims, function(sims){
  # Simulate x's for each value of n
  x = sapply(1:length(n), function(i){
    rnorm(n[i], 0, sqrt(2))
  })

  # Calculate sample means for each n
  xbar = sapply(1:length(n), function(i){
    1/sqrt(n[i])*sum(x[[i]])
  })
})

# flip dimensions of xbars to make it more intuitive
xbars_sqrt = t(xbars_sqrt)

# Name columns for pretty output
colnames(xbars_sqrt) = c("10", "100", "1000")
colnames(xbars) = c("10", "100", "1000")

# Rearrange data frame to make it work for graphing
xbars_graph = pivot_longer(as.data.frame(xbars_sqrt), cols = c("10", "100", "1000"))

# Graph kernal density
xbars_graph |> filter(name == "10") |>
  ggplot(aes(x = value)) +
  geom_density(color = "blue") +
```

```
  stat_function(fun = dnorm, args = list(sd = sqrt(2))) +
  xlab("Value") +
  ylab("Density") +
  labs(title = "Kernal Density for N = 10") +
  cowplot::theme_cowplot()
```

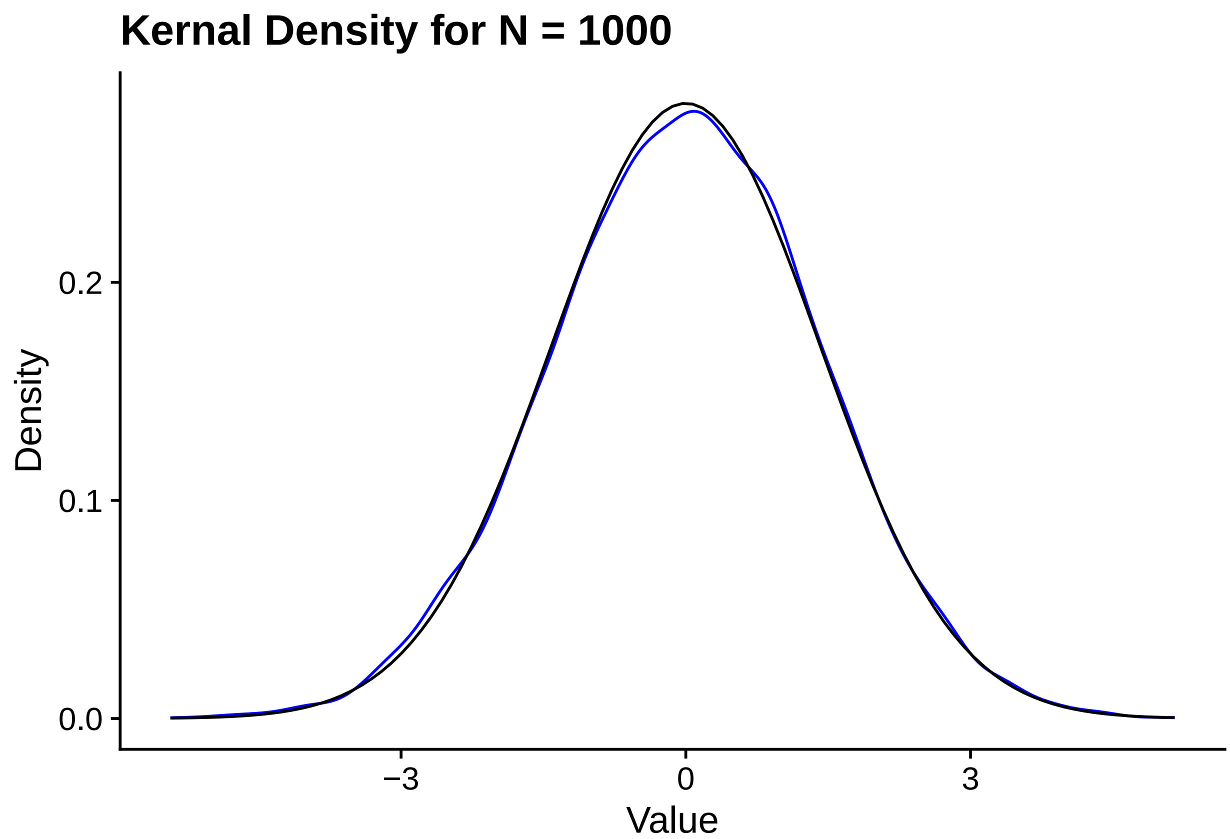


Part c)

```
xbars_graph |> filter(name == "100") |>
  ggplot(aes(x = value)) +
  geom_density(color = "blue") +
  stat_function(fun = dnorm, args = list(sd = sqrt(2))) +
  xlab("Value") +
  ylab("Density") +
  labs(title = "Kernal Density for N = 100") +
  cowplot::theme_cowplot()
```




```r
xbars_graph |> filter(name == "1000") |>
  ggplot(aes(x = value)) +
  geom_density(color = "blue") +
  stat_function(fun = dnorm, args = list(sd = sqrt(2))) +
  xlab("Value") +
  ylab("Density") +
  labs(title = "Kernal Density for N = 1000") +
  cowplot::theme_cowplot()
```




```
# Calculate ks.test
sapply(1:length(n), function(i){
  ks.test(xbars_sqrt[,i], "pnorm", 0, sqrt(2))
})
```

```
##             [,1]
## statistic   0.008916184
## p.value     0.4044093
## alternative "two-sided"
## method      "Asymptotic one-sample Kolmogorov-Smirnov test"
## data.name   "xbars_sqrt[, i]"
## data        list,2
## exact       FALSE
##             [,2]
## statistic   0.008979004
## p.value     0.3956392
## alternative "two-sided"
## method      "Asymptotic one-sample Kolmogorov-Smirnov test"
## data.name   "xbars_sqrt[, i]"
## data        list,2
## exact       FALSE
##             [,3]
## statistic   0.006980379
## p.value     0.714499
## alternative "two-sided"
## method      "Asymptotic one-sample Kolmogorov-Smirnov test"
```

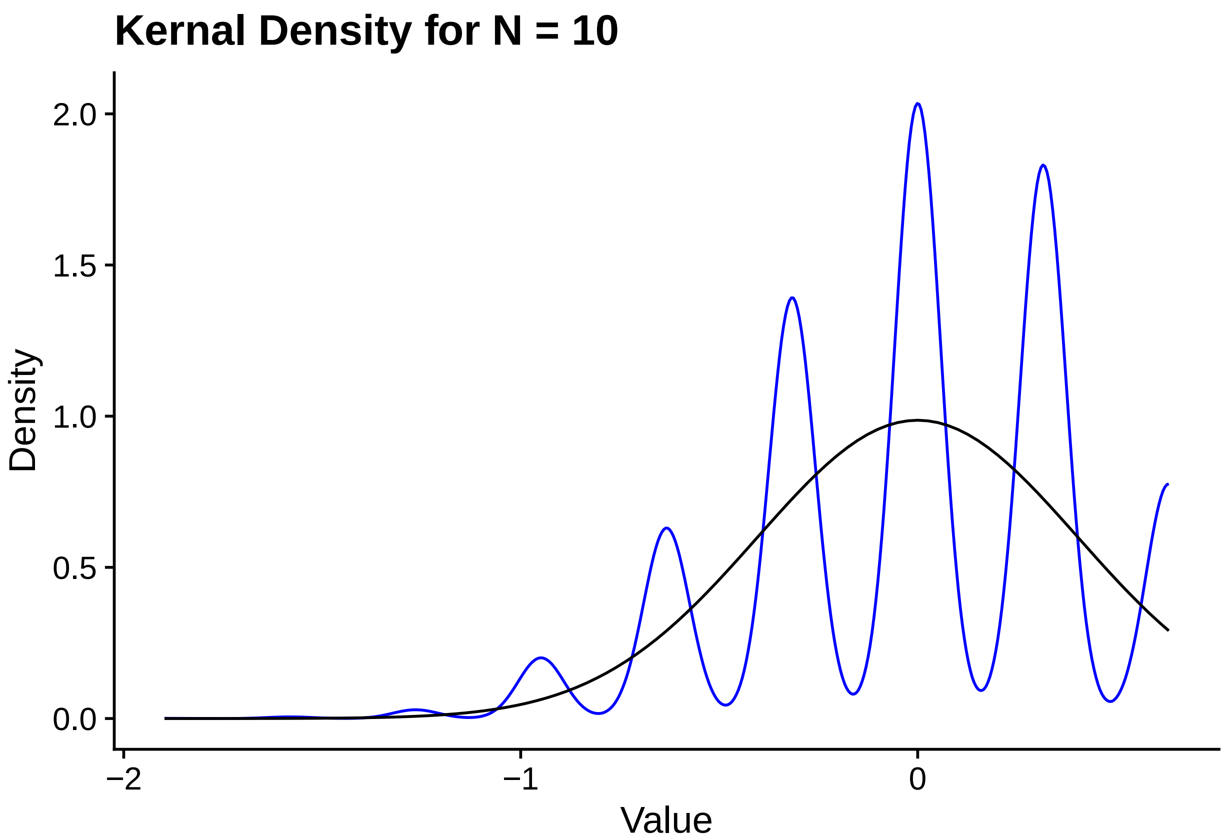
```
## data.name   "xbars_sqrt[, i]"
## data        list,2
## exact       FALSE
```

```r
# x ~ Bernoulli(0,0.8)

# Set n's
n = c(10, 100, 1000)

# Set number of simulations
sims = 10000

# Create a bunch of sample means
xbars_binom = sapply(1:sims, function(sims){
  # Simulate x's for each value of n
  x = sapply(1:length(n), function(i){
    rbinom(n[i], 1, 0.8)
  })

  # Calculate sample means for each n
  xbar = sapply(1:length(n), function(i){
    1/n[i]*sum(x[[i]])
  })
})

# flip dimensions of xbars to make it more intuitive
xbars_binom = t(xbars_binom)

# Name columns for pretty output
colnames(xbars_binom) = c("10", "100", "1000")

# Calculate means for each n
x_means_binom = colMeans(xbars_binom)

# Calculate variances for each n
x_vars_binom = sapply(1:length(n), function(i){var(xbars_binom[,i])})
names(x_vars_binom) = c("10", "100", "1000")

# Make the output nice
(samp_mean_binom_df = data.frame(x_means_binom, x_vars_binom))
```

Part d)

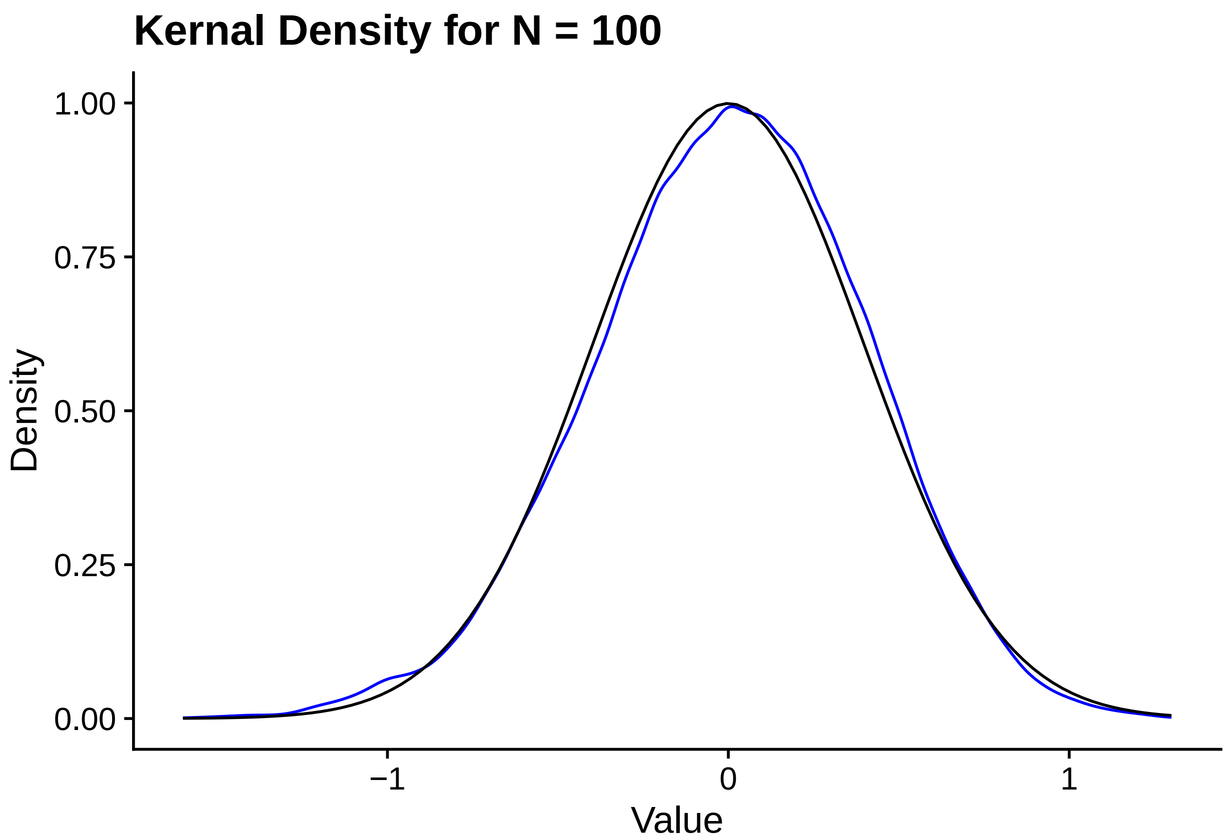
```
##      x_means_binom x_vars_binom
## 10        0.799660 0.0161474991
## 100       0.799436 0.0016083227
## 1000      0.799884 0.0001626534
```

```r
# Same as earlier part but instead of sample mean its dividing by sqrt(n)
xbars_sqrt_binom = sapply(1:sims, function(sims){
  # Simulate x's for each value of n
  x = sapply(1:length(n), function(i){
    rbinom(n[i], 1, 0.8)
  })

  # Calculate sample means for each n
  xbar = sapply(1:length(n), function(i){
    sqrt(n[i])*(1/n[i]*sum(x[[i]]) - 0.8)
  })
})

# flip dimensions of xbars to make it more intuitive
xbars_sqrt_binom = t(xbars_sqrt_binom)

# Name columns for pretty output
colnames(xbars_sqrt_binom) = c("10", "100", "1000")

# Get empirical variance for comparison distribution
x_vars_sqrt_binom = sapply(1:length(n), function(i){var(xbars_sqrt_binom[,i])})

# Rearrange data frame to make it work for graphing
xbars_sqrt_binom_graph = pivot_longer(as.data.frame(xbars_sqrt_binom), cols = c("10", "100", "1000"))

# Graph kernal density
xbars_sqrt_binom_graph |> filter(name == "10") |>
  ggplot(aes(x = value)) +
  geom_density(color = 'blue') +
  stat_function(fun = dnorm, args = list(sd = sqrt(x_vars_sqrt_binom[1]))) +
  xlab("Value") +
  ylab("Density") +
  labs(title = "Kernal Density for N = 10") +
  cowplot::theme_cowplot()
```

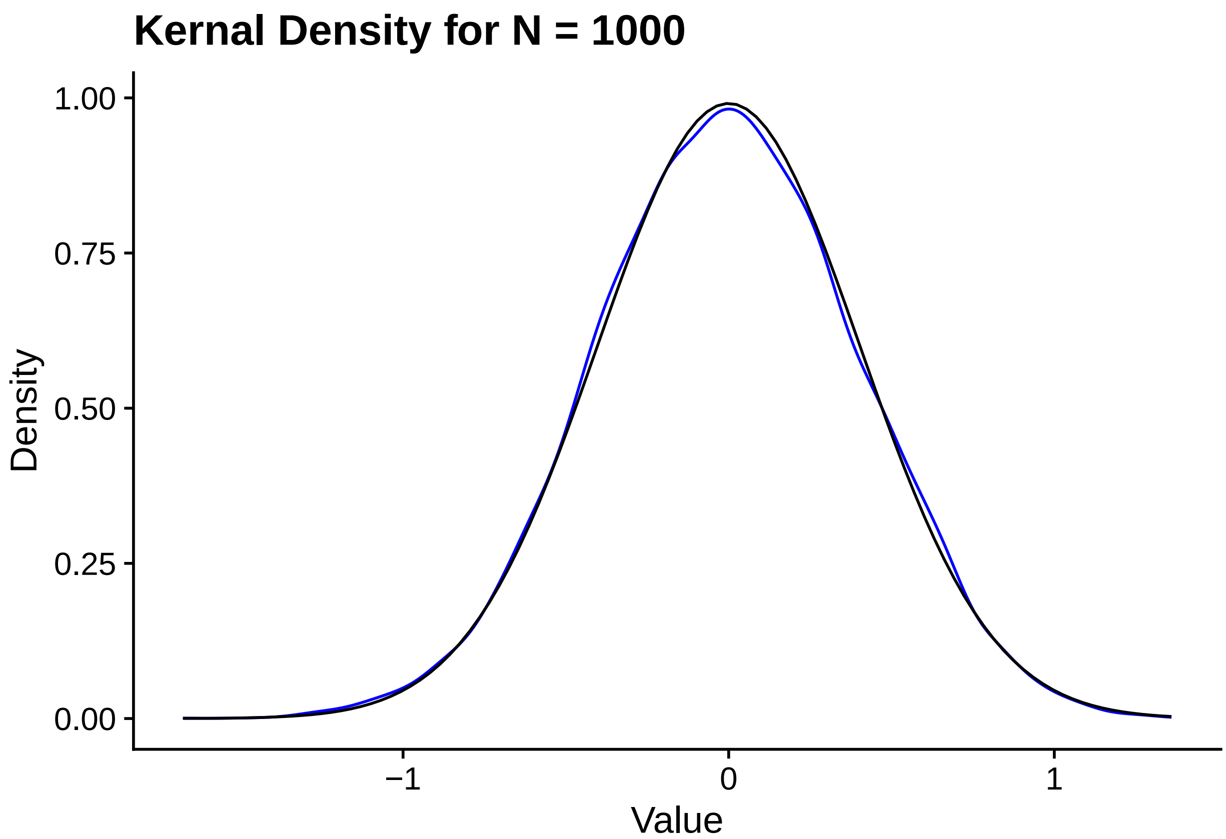


Part e)

```
xbars_sqrt_binom_graph |> filter(name == "100") |>
  ggplot(aes(x = value)) +
  geom_density(color = 'blue') +
  stat_function(fun = dnorm, args = list(sd = sqrt(x_vars_sqrt_binom[2]))) +
  xlab("Value") +
  ylab("Density") +
  labs(title = "Kernal Density for N = 100") +
  cowplot::theme_cowplot()
```




```r
xbars_sqrt_binom_graph |> filter(name == "1000") |>
  ggplot(aes(x = value)) +
  geom_density(color = 'blue') +
  stat_function(fun = dnorm, args = list(sd = sqrt(x_vars_sqrt_binom[3]))) +
  xlab("Value") +
  ylab("Density") +
  labs(title = "Kernal Density for N = 1000") +
  cowplot::theme_cowplot()
```




```
# Calculate ks.test
sapply(1:length(n), function(i){
  ks.test(xbars_sqrt_binom[,i], "pnorm", 0, sqrt(0.16))
})
```

```
## Warning in ks.test.default(xbars_sqrt_binom[, i], "pnorm", 0, sqrt(0.16)): ties
## should not be present for the Kolmogorov-Smirnov test

## Warning in ks.test.default(xbars_sqrt_binom[, i], "pnorm", 0, sqrt(0.16)): ties
## should not be present for the Kolmogorov-Smirnov test

## Warning in ks.test.default(xbars_sqrt_binom[, i], "pnorm", 0, sqrt(0.16)): ties
## should not be present for the Kolmogorov-Smirnov test
```

```
##             [,1]
## statistic   0.1728
## p.value     0
## alternative "two-sided"
## method      "Asymptotic one-sample Kolmogorov-Smirnov test"
## data.name   "xbars_sqrt_binom[, i]"
## data        list,2
## exact       FALSE
##             [,2]
## statistic   0.064
## p.value     0
```

```
## alternative "two-sided"
## method      "Asymptotic one-sample Kolmogorov-Smirnov test"
## data.name   "xbars_sqrt_binom[, i]"
## data        list,2
## exact       FALSE
##             [,3]
## statistic   0.02337185
## p.value     3.60097e-05
## alternative "two-sided"
## method      "Asymptotic one-sample Kolmogorov-Smirnov test"
## data.name   "xbars_sqrt_binom[, i]"
## data        list,2
## exact       FALSE
```

Question 5)

```r
# Set n and sims
N = 100
sims = 10000

# Simulate data a bunch of times
beta = sapply(1:sims, function(i){

  # Simulate data
  x = matrix(c(rep(1,N), rnorm(N, 0, 2)), ncol = 2)
  e = rnorm(N, 0, 1)

  # Calculate y
  y = 2*x[,2] + e

  # Calculate OLS estimator
  beta = solve(t(x)%*%x)%*%t(x)%*%y

})

# Calculate mean and variance
(means = sapply(1:2, function(i){mean(beta[i,])}))
```

a)

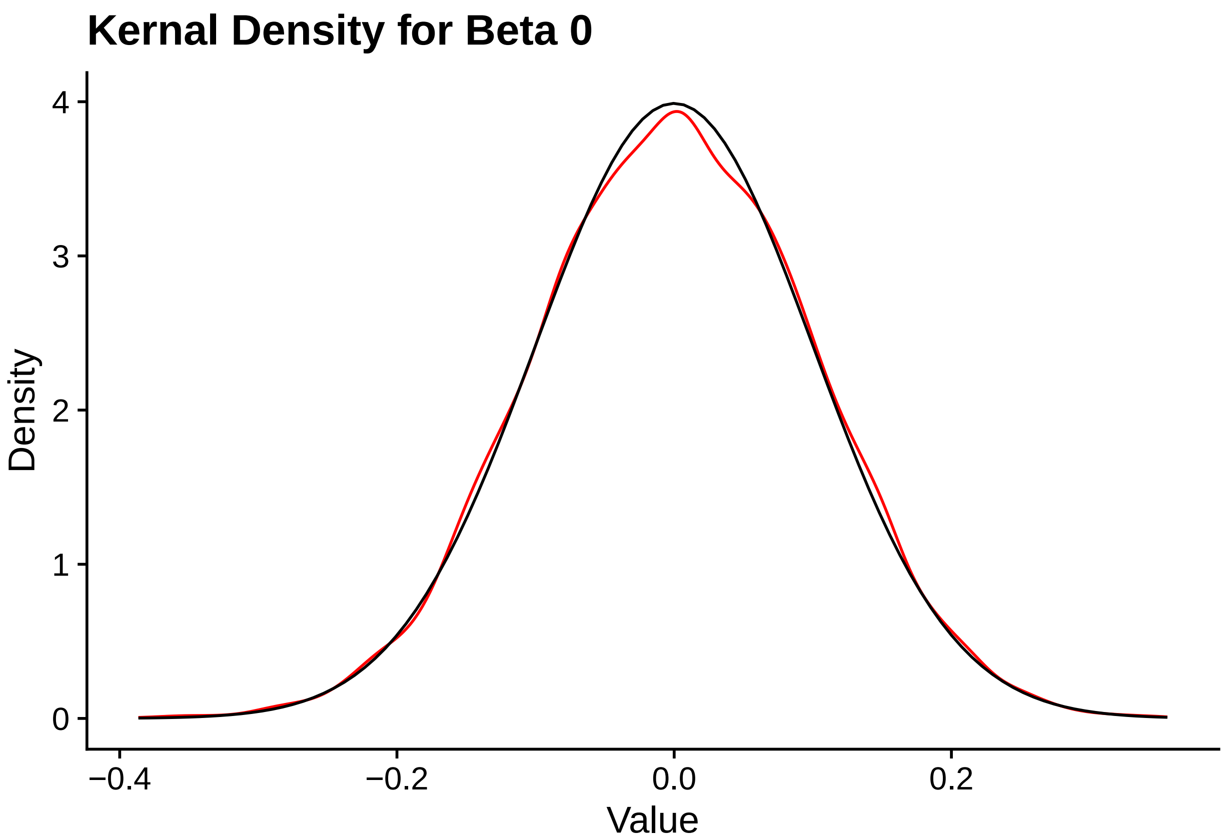
```
## [1] 0.0002779961 1.9996014094
```

```r
(vars = sapply(1:2, function(i){var(beta[i,])}))
```

```
## [1] 0.010137806 0.002628666
```

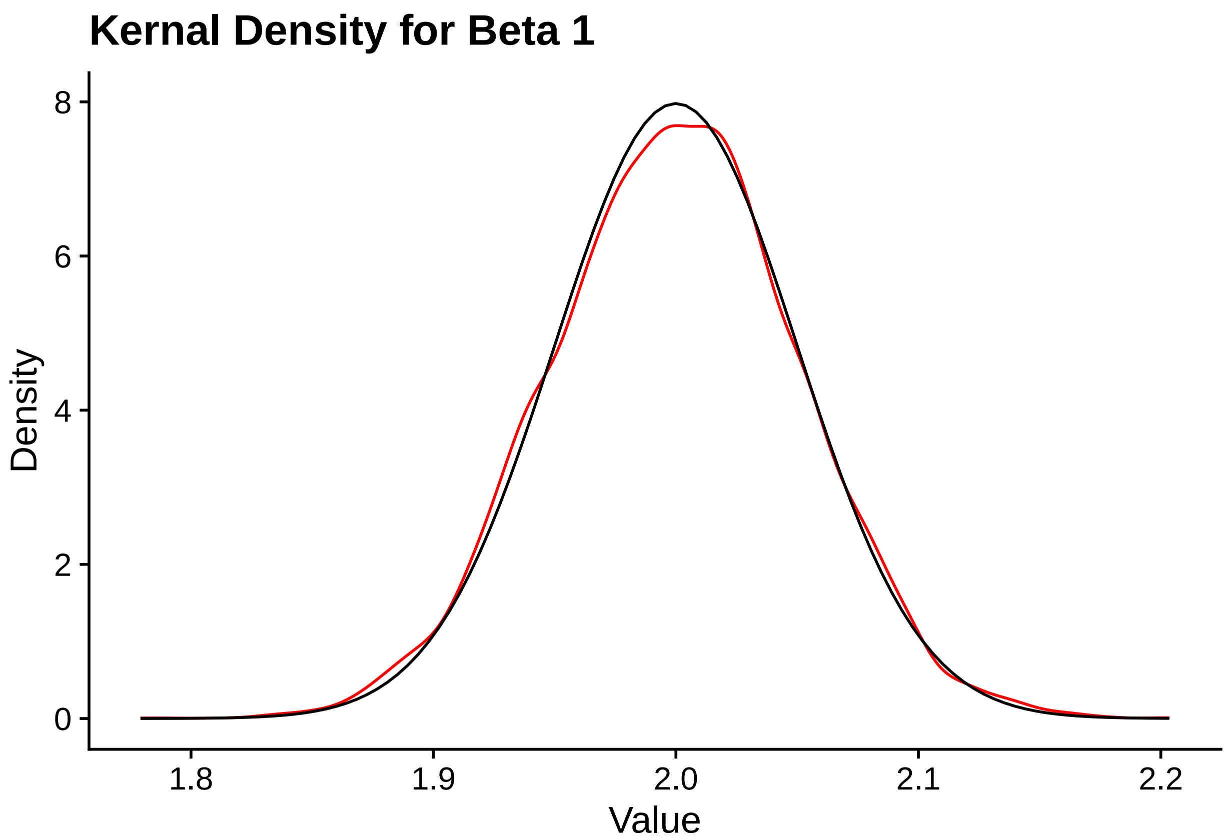
```r
beta_graph = as.data.frame(t(beta))
```

```r
# Graph densities
beta_graph |> ggplot(aes(x = V1)) +
  geom_density(color = 'red') +
  stat_function(fun = dnorm, args = list(sd = 0.1)) +
  xlab("Value") +
  ylab("Density") +
  labs(title = "Kernal Density for Beta 0") +
  cowplot::theme_cowplot()
```



b)

```r
beta_graph |> ggplot(aes(x = V2)) +
  geom_density(color = 'red') +
  stat_function(fun = dnorm, args = list(mean = 2, sd = 0.05)) +
  xlab("Value") +
  ylab("Density") +
  labs(title = "Kernal Density for Beta 1") +
  cowplot::theme_cowplot()
```




```r
# Run ks test for both betas
ks.test(beta_graph$V1, 'pnorm', 0, 0.1)
```

```
##
##  Asymptotic one-sample Kolmogorov-Smirnov test
##
## data:  beta_graph$V1
## D = 0.0084663, p-value = 0.4704
## alternative hypothesis: two-sided
```

```r
ks.test(beta_graph$V2, 'pnorm', 2, 0.05)
```

```
##
##  Asymptotic one-sample Kolmogorov-Smirnov test
##
## data:  beta_graph$V2
## D = 0.01326, p-value = 0.05939
## alternative hypothesis: two-sided
```

```r
# set vector of n's
n = c(10, 100, 1000)
```

```r
# Calculate betas for each n
beta_ns = sapply(1:sims, function(i){

  sapply(1:3, function(i){
    # Simulate data
    x = matrix(c(rep(1,n[i]), rnorm(n[i], 0, 2)), ncol = 2)
    e = rnorm(n[i], 0, 1)

    # Calculate y
    y = 2*x[,2] + e

    # Calculate OLS estimator
    beta = solve(t(x)%*%x)%*%t(x)%*%y

    # Centered distribution of betas
    sqrt(n[i])*(beta - c(0,2))
  })
})

# KS test for n = 10 for beta 0 then beta1
ks.test(beta_ns[1,], 'pnorm', 0, 1)
```

c)

```
##
##  Asymptotic one-sample Kolmogorov-Smirnov test
##
## data:  beta_ns[1, ]
## D = 0.020516, p-value = 0.0004417
## alternative hypothesis: two-sided
```

```r
ks.test(beta_ns[2,], 'pnorm', 0, 0.5)
```

```
##
##  Asymptotic one-sample Kolmogorov-Smirnov test
##
## data:  beta_ns[2, ]
## D = 0.032813, p-value = 8.895e-10
## alternative hypothesis: two-sided
```

```r
# n = 100
ks.test(beta_ns[3,], 'pnorm', 0, 1)
```

```
##
##  Asymptotic one-sample Kolmogorov-Smirnov test
##
## data:  beta_ns[3, ]
## D = 0.0060459, p-value = 0.8581
## alternative hypothesis: two-sided
```

```r
ks.test(beta_ns[4,], 'pnorm', 0, 0.5)
```

```
##
##  Asymptotic one-sample Kolmogorov-Smirnov test
##
## data:  beta_ns[4, ]
## D = 0.0077811, p-value = 0.5801
## alternative hypothesis: two-sided
```

```r
# n = 1000
ks.test(beta_ns[5,], 'pnorm', 0, 1)
```

```
##
##  Asymptotic one-sample Kolmogorov-Smirnov test
##
## data:  beta_ns[5, ]
## D = 0.010874, p-value = 0.1878
## alternative hypothesis: two-sided
```

```r
ks.test(beta_ns[6,], 'pnorm', 0, 0.5)
```

```
##
##  Asymptotic one-sample Kolmogorov-Smirnov test
##
## data:  beta_ns[6, ]
## D = 0.0093916, p-value = 0.341
## alternative hypothesis: two-sided
```

```r


# same as part a, but we're now making the errors ~ U(0,2)
# Set n and sims
N = 100
sims = 10000

# Simulate data a bunch of times
beta = sapply(1:sims, function(i){

  # Simulate data
  x = matrix(c(rep(1,N), rnorm(N, 0, 2)), ncol = 2)
  e = runif(N, 0, 2)

  # Calculate y
  y = 2*x[,2] + e

  # Calculate OLS estimator
  beta = solve(t(x)%*%x)%*%t(x)%*%y

})

# Calculate mean and variance
(means = sapply(1:2, function(i){mean(beta[i,])}))
```

d)

```
## [1] 1.000933 1.999613
```

```r
(vars = sapply(1:2, function(i){var(beta[i,])}))
```

```
## [1] 0.0034218555 0.0008504284
```

```r
# Same as part c, but we're changing the erros again. We could have written a function which takes the
# set vector of n's
n = c(10, 100, 1000)

# Calculate betas for each n
beta_ns = sapply(1:1, function(q){

  sapply(1:3, function(i){
    # Simulate data
    x = matrix(c(rep(1,n[i]), rnorm(n[i], 0, 2)), ncol = 2)
    e = runif(n[i], 0, 2)

    # Calculate y
    y = 2*x[,2] + e

    # Calculate OLS estimator
    beta = solve(t(x)%*%x)%*%t(x)%*%y

    # Centered distribution of betas
    sqrt(n[i])*(beta - c(1,2))
  })
})

# KS test for n = 10 for beta 0 then beta1
ks.test(beta_ns[1,], 'pnorm', 0, 1/3)
```

e)

```
##
##  Exact one-sample Kolmogorov-Smirnov test
##
## data:  beta_ns[1, ]
## D = 0.98104, p-value = 0.03793
## alternative hypothesis: two-sided
```

```r
ks.test(beta_ns[2,], 'pnorm', 0, I(1/3+1/4))
```

```
##
##  Exact one-sample Kolmogorov-Smirnov test
##
## data:  beta_ns[2, ]
## D = 0.69851, p-value = 0.603
## alternative hypothesis: two-sided
```

```r
# n = 100
ks.test(beta_ns[3,], 'pnorm', 0, 1/3)
```

```
##
##  Exact one-sample Kolmogorov-Smirnov test
##
## data:  beta_ns[3, ]
## D = 0.91858, p-value = 0.1628
## alternative hypothesis: two-sided
```

```r
ks.test(beta_ns[4,], 'pnorm', 0, I(1/3+1/4))
```

```
##
##  Exact one-sample Kolmogorov-Smirnov test
##
## data:  beta_ns[4, ]
## D = 0.57283, p-value = 0.8543
## alternative hypothesis: two-sided
```

```r
# n = 1000
ks.test(beta_ns[5,], 'pnorm', 0, 1/3)
```

```
##
##  Exact one-sample Kolmogorov-Smirnov test
##
## data:  beta_ns[5, ]
## D = 0.83894, p-value = 0.3221
## alternative hypothesis: two-sided
```

```r
ks.test(beta_ns[6,], 'pnorm', 0, I(1/3+1/4))
```

```
##
##  Exact one-sample Kolmogorov-Smirnov test
##
## data:  beta_ns[6, ]
## D = 0.5194, p-value = 0.9612
## alternative hypothesis: two-sided
```

```r


# Now add correlation between x and epsilon

# Set n and sims
N = 100
sims = 10000

# Simulate data a bunch of times
beta = sapply(1:sims, function(i){

  # Simulate data
```

```r
  x_e = MASS::mvrnorm(n = 100, c(0,0), matrix(c(4,.4,.4,1), nrow = 2))

  # Calculate y
  y = 2*x_e[,1] + x_e[,2]

  # Calculate OLS estimator
  x = matrix(c(rep(1,N), x_e[,1]), ncol = 2)
  beta = solve(t(x)%*%x)%*%t(x)%*%y

})

colMeans(t(beta))
```

e)

```
## [1] -0.001494128  2.099817750
```