MTH210: Lab 11

Bootstrapping

1. **In this first question, we will compare the performance of parameteric and non-parameteric bootstrapping when the true sampling distribution of an estimator is known.**

   Consider $X_1, X_2, \ldots, X_n \overset{iid}{\sim} N(\theta, 1)$. We want to estimate θ using the sample mean $\bar{X}_n = n^{-1}\sum_{i=1}^{n} X_i$. First, let's simulate the data:

```
set.seed(1)
n <- 100
theta <- 5  # choosing a true value for theta
dat <- rnorm(n, mean = theta) # the actual data
```

   Now, we are aware that the exact distribution of $\bar{X}_n$ is:

$$\sqrt{n}(\bar{X}_n - \theta) \sim N\left(0, \frac{1}{n}\right) =: G_n$$

   That is, the sampling distribution of the sample mean is known exactly in this case. However, we will use bootstrap to compare the true G_n with the bootstrap estimates of G_n.

   Set $B = 1000$ and first we will implement non-parametric bootstrap.

```
## non-parametric bootstrap
B <- 1000
np.boot <- numeric(length = B)
for(b in 1:B)
{
  # resample the index with replacement
  foo <- sample(1:n, size = n, replace = TRUE)
  # find bootstrap samples
  boot.samp <- dat[foo]

  # find sample mean of boostrap samples
  np.boot[b] <- ...
```

```
}
```

The elements of `np.boot` are the bootstrap estimates and are all approximately from G_n. Similarly, we can implement parametric bootstrap:

```
p.boot <- numeric(length = B)
for(b in 1:B)
{
  boot.samp <- rnorm(...)
  p.boot[b] <- ...
}
```

Next, notice that if we assume knowlegde of G_n, then we know that

$$\bar{X}_n \sim N\left(\theta, \frac{1}{n}\right).$$

Now, naturally, in real data situations, we don't know θ, so we assume the following:

$$\bar{X}_n \sim N\left(\bar{X}_n, \frac{1}{n}\right).$$

We can compare the above with the estimated sampling distribution obtained from the two bootstrap methods.

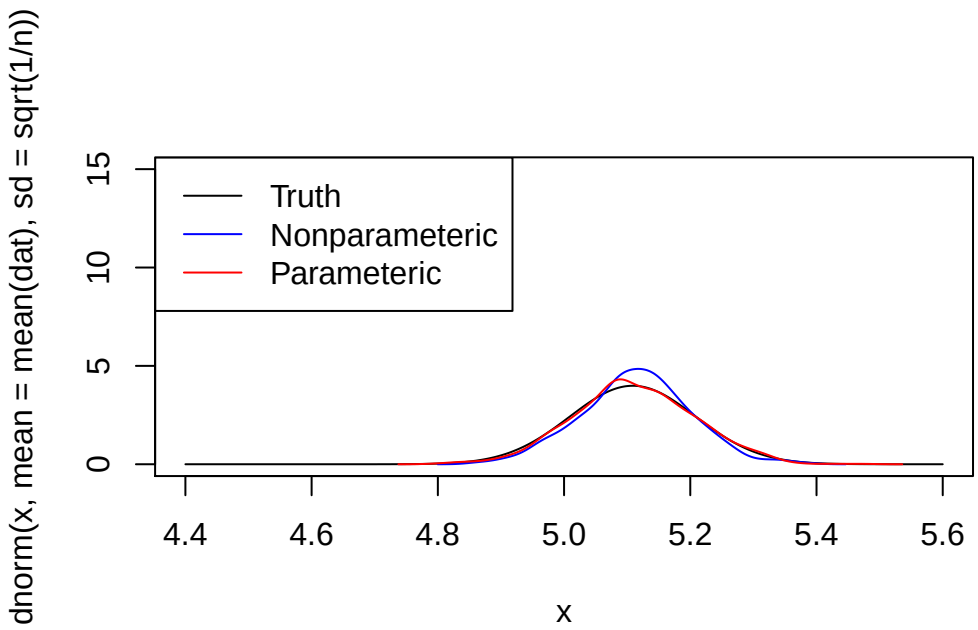
```
x <- seq(theta-6/sqrt(n), theta + 6/sqrt(n), length = 500)
# plotting almost truth
plot(x, dnorm(x, mean = mean(dat), sd = sqrt(1/n)), type = 'l', ylim = c(0,5))
lines(density(np.boot), col = "blue")
lines(density(p.boot), col = "red")
legend("topleft", legend = c("Truth", "Nonparameteric", "Parameteric"),
       col = c("black", "blue", "red"), lty = 1)
```

What do you see when you make the above plot? Do the three densities match each other fairly well? Now repeat the experiment for $n = 10, 50, 500, 1000$. What do you see as n changes?

We did this in class, so I am just presenting the code here:

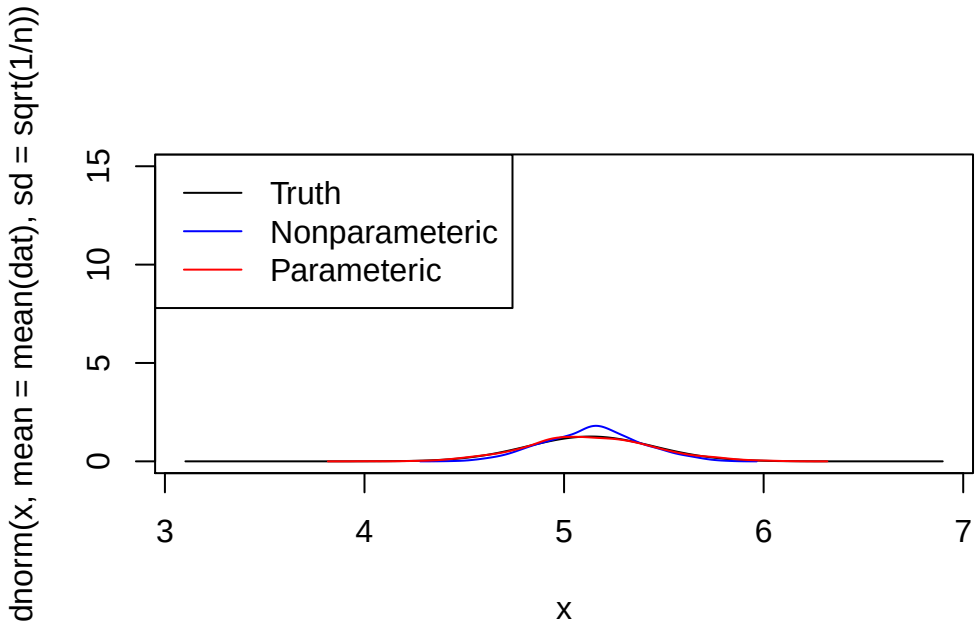
```
set.seed(1)
n <- 100
theta <- 5  # choosing a true value for theta
dat <- rnorm(n, mean = theta)
```

```r
## non-parametric bootstrap
B <- 1000
np.boot <- numeric(length = B)
for(b in 1:B)
{
  foo <- sample(1:n, size = n, replace = TRUE)
  boot.samp <- dat[foo]

  np.boot[b] <- mean(boot.samp)
}

p.boot <- numeric(length = B)
for(b in 1:B)
{
  boot.samp <- rnorm(n, mean = mean(dat))
  p.boot[b] <- mean(boot.samp)
}

x <- seq(theta-6/sqrt(n), theta + 6/sqrt(n), length = 500)
plot(x, dnorm(x, mean = mean(dat), sd = sqrt(1/n)), type = 'l', ylim = c(0,15))
lines(density(np.boot), col = "blue")
lines(density(p.boot), col = "red")
legend("topleft", legend = c("Truth", "Nonparameteric", "Parameteric"),
       col = c("black", "blue", "red"), lty = 1)
```



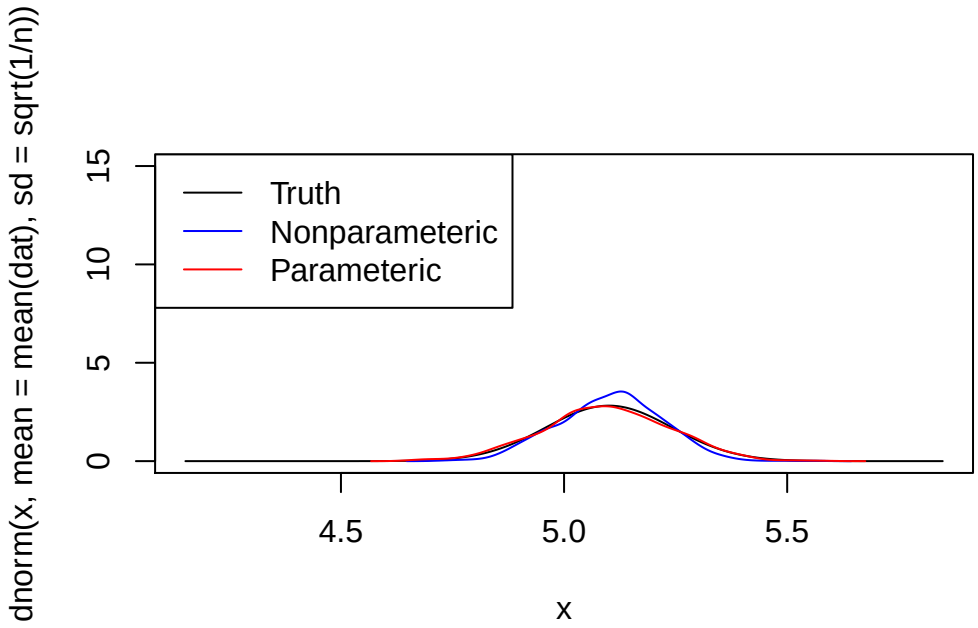
I'll now repeat this for different values of n.

```r
set.seed(1)
n <- 10
theta <- 5  # choosing a true value for theta
dat <- rnorm(n, mean = theta)

## non-parametric bootstrap
B <- 1000
np.boot <- numeric(length = B)
for(b in 1:B)
{
  foo <- sample(1:n, size = n, replace = TRUE)
  boot.samp <- dat[foo]

  np.boot[b] <- mean(boot.samp)
}

p.boot <- numeric(length = B)
for(b in 1:B)
{
  boot.samp <- rnorm(n, mean = mean(dat))
  p.boot[b] <- mean(boot.samp)
}

x <- seq(theta-6/sqrt(n), theta + 6/sqrt(n), length = 500)
plot(x, dnorm(x, mean = mean(dat), sd = sqrt(1/n)), type = 'l', ylim = c(0,15))
lines(density(np.boot), col = "blue")
lines(density(p.boot), col = "red")
legend("topleft", legend = c("Truth", "Nonparameteric", "Parameteric"),
       col = c("black", "blue", "red"), lty = 1)
```



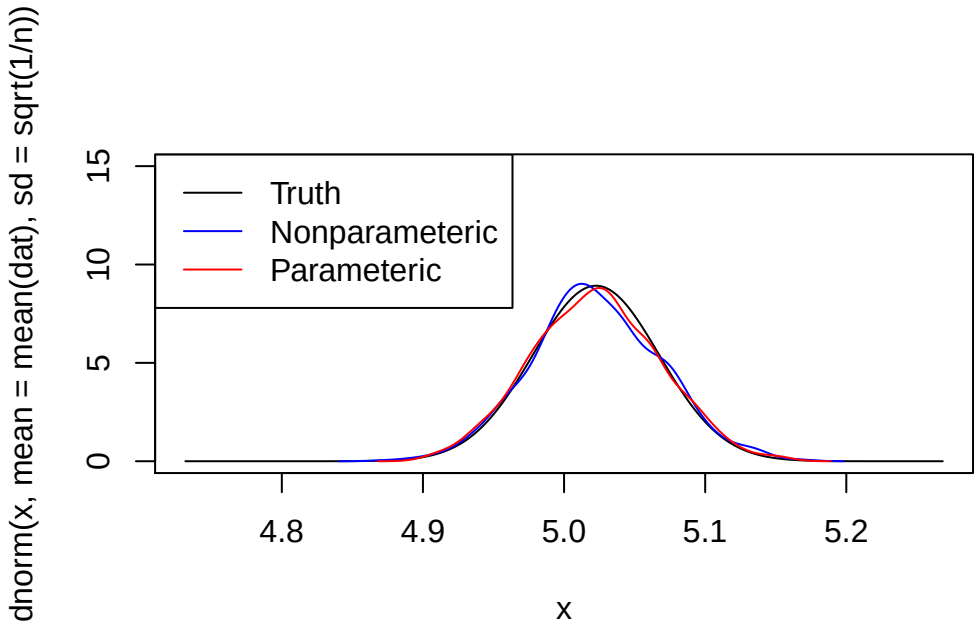

```r
# n = 50
set.seed(1)
n <- 50
theta <- 5  # choosing a true value for theta
dat <- rnorm(n, mean = theta)

## non-parametric bootstrap
B <- 1000
np.boot <- numeric(length = B)
for(b in 1:B)
{
  foo <- sample(1:n, size = n, replace = TRUE)
  boot.samp <- dat[foo]

  np.boot[b] <- mean(boot.samp)
}

p.boot <- numeric(length = B)
for(b in 1:B)
{
  boot.samp <- rnorm(n, mean = mean(dat))
  p.boot[b] <- mean(boot.samp)
}

x <- seq(theta-6/sqrt(n), theta + 6/sqrt(n), length = 500)
```

```r
plot(x, dnorm(x, mean = mean(dat), sd = sqrt(1/n)), type = 'l', ylim = c(0,15))
lines(density(np.boot), col = "blue")
lines(density(p.boot), col = "red")
legend("topleft", legend = c("Truth", "Nonparameteric", "Parameteric"),
       col = c("black", "blue", "red"), lty = 1)
```



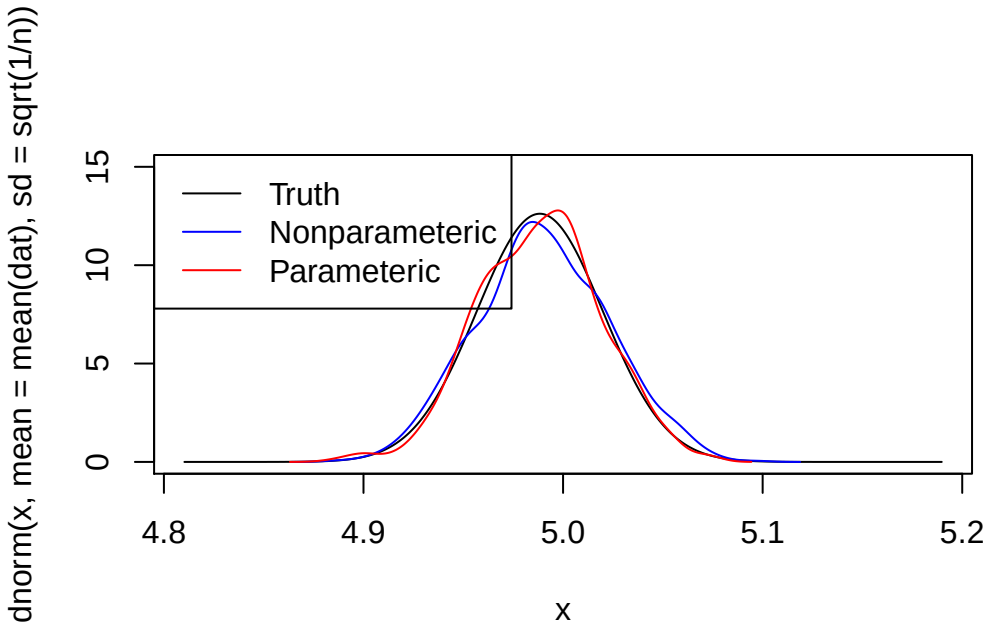

```r
# n = 500
set.seed(1)
n <- 500
theta <- 5  # choosing a true value for theta
dat <- rnorm(n, mean = theta)

## non-parametric bootstrap
B <- 1000
np.boot <- numeric(length = B)
for(b in 1:B)
{
  foo <- sample(1:n, size = n, replace = TRUE)
  boot.samp <- dat[foo]

  np.boot[b] <- mean(boot.samp)
}

p.boot <- numeric(length = B)
for(b in 1:B)
```

```r
{
  boot.samp <- rnorm(n, mean = mean(dat))
  p.boot[b] <- mean(boot.samp)
}

x <- seq(theta-6/sqrt(n), theta + 6/sqrt(n), length = 500)
plot(x, dnorm(x, mean = mean(dat), sd = sqrt(1/n)), type = 'l', ylim = c(0,15))
lines(density(np.boot), col = "blue")
lines(density(p.boot), col = "red")
legend("topleft", legend = c("Truth", "Nonparameteric", "Parameteric"),
       col = c("black", "blue", "red"), lty = 1)
```




```r
#n = 1000
set.seed(1)
n <- 1000
theta <- 5  # choosing a true value for theta
dat <- rnorm(n, mean = theta)

## non-parametric bootstrap
B <- 1000
np.boot <- numeric(length = B)
for(b in 1:B)
{
  foo <- sample(1:n, size = n, replace = TRUE)
  boot.samp <- dat[foo]
```

```r
  np.boot[b] <- mean(boot.samp)
}

p.boot <- numeric(length = B)
for(b in 1:B)
{
  boot.samp <- rnorm(n, mean = mean(dat))
  p.boot[b] <- mean(boot.samp)
}

x <- seq(theta-6/sqrt(n), theta + 6/sqrt(n), length = 500)
plot(x, dnorm(x, mean = mean(dat), sd = sqrt(1/n)), type = 'l', ylim = c(0,15))
lines(density(np.boot), col = "blue")
lines(density(p.boot), col = "red")
legend("topleft", legend = c("Truth", "Nonparameteric", "Parameteric"),
       col = c("black", "blue", "red"), lty = 1)
```



2. **Now we will create 95% confidence intervals for θ in the previous example, and compare their coverage of these intervals is. The coverage of a confidence is the proportion of intervals that contain the true parameter, in repeated experiments.**

   Thus, we will repeat Problem 1 repeatedly and check whether the true θ is in the intervals or not.

```r
set.seed(1)
reps <- 100
n <- 100 # this n right now
B <- 1000

theta <- 5  # choosing a true value for theta

alpha <- .05
ends <- c(alpha/2, 1 - alpha/2)

# will contain 1 or 0 based on whether the
# truth parameter was in the interval in each rep
truth <- numeric(length = reps)
npara <- numeric(length = reps)
para <- numeric(length = reps)

for(r in 1:reps)
{
  # draw new sample in every repetition
  dat <- rnorm(n, mean = theta)

  ## parametric bootstrap
  np.boot <- numeric(length = B)
  for(b in 1:B)
  {
    foo <- sample(1:n, size = n, replace = TRUE)
    boot.samp <- dat[foo]
    np.boot[b] <- mean(boot.samp)
  }
  np.int <- quantile(np.boot, ends)
  npara[r] <- (theta >= np.int[1] && theta <= np.int[2])

  p.boot <- numeric(length = B)
  for(b in 1:B)
  {
    boot.samp <- rnorm(n, mean = mean(dat))
    p.boot[b] <- mean(boot.samp)
  }
  p.int <- quantile(p.boot, ends)
  para[r] <- (theta >= p.int[1] && theta <= p.int[2])
}
```

```
mean(npara)
```

```
[1] 0.92
```

```
mean(para)
```

```
[1] 0.95
```

How does the coverage probabilities compare for the two methods? What happens when n is large and when n is small?

In the above code non-parameteric estimator has lower than 95% coverage, and the parametric estimator has good coverage – the desired 95%.

We repeat this process now for $n = 10$ and $n = 1000$

```
set.seed(1)
reps <- 100
n <- 10  # n = 10
B <- 1000

theta <- 5  # choosing a true value for theta

alpha <- .05
ends <- c(alpha/2, 1 - alpha/2)

# will contain 1 or 0 based on whether the
# truth parameter was in the interval in each rep
truth <- numeric(length = reps)
npara <- numeric(length = reps)
para <- numeric(length = reps)

for(r in 1:reps)
{
  # draw new sample in every repetition
  dat <- rnorm(n, mean = theta)

  ## parametric bootstrap
  np.boot <- numeric(length = B)
  for(b in 1:B)
  {
    foo <- sample(1:n, size = n, replace = TRUE)
    boot.samp <- dat[foo]
```

```
    np.boot[b] <- mean(boot.samp)
  }
  np.int <- quantile(np.boot, ends)
  npara[r] <- (theta >= np.int[1] && theta <= np.int[2])

  p.boot <- numeric(length = B)
  for(b in 1:B)
  {
    boot.samp <- rnorm(n, mean = mean(dat))
    p.boot[b] <- mean(boot.samp)
  }
  p.int <- quantile(p.boot, ends)
  para[r] <- (theta >= p.int[1] && theta <= p.int[2])
}

mean(npara)
```

```
[1] 0.92
```

```
mean(para)
```

```
[1] 0.94
```

```
set.seed(1)
reps <- 100
n <- 1000  # n = 1000
B <- 1000

theta <- 5  # choosing a true value for theta

alpha <- .05
ends <- c(alpha/2, 1 - alpha/2)

# will contain 1 or 0 based on whether the
# truth parameter was in the interval in each rep
truth <- numeric(length = reps)
npara <- numeric(length = reps)
para <- numeric(length = reps)

for(r in 1:reps)
{
  # draw new sample in every repetition
  dat <- rnorm(n, mean = theta)
```

```
  ## parametric bootstrap
  np.boot <- numeric(length = B)
  for(b in 1:B)
  {
    foo <- sample(1:n, size = n, replace = TRUE)
    boot.samp <- dat[foo]
    np.boot[b] <- mean(boot.samp)
  }
  np.int <- quantile(np.boot, ends)
  npara[r] <- (theta >= np.int[1] && theta <= np.int[2])

  p.boot <- numeric(length = B)
  for(b in 1:B)
  {
    boot.samp <- rnorm(n, mean = mean(dat))
    p.boot[b] <- mean(boot.samp)
  }
  p.int <- quantile(p.boot, ends)
  para[r] <- (theta >= p.int[1] && theta <= p.int[2])
}

mean(npara)
```

```
[1] 0.94
```

```
mean(para)
```

```
[1] 0.94
```

When sample size is larger, the coverage probability is better. You may repeat this with larger value of `reps` to get a better estimate of the coverage probability.

3. **Consider F = Gamma$(10, 5)$ where 5 is the rate parameter. What is the mean of this distribution and what is the variance?**

   Recall that the mean and variance of Gamma(α, β) is:

$$\text{Mean} = \frac{\alpha}{\beta}, \qquad \text{Variance} = \frac{\alpha}{\beta^2}$$

   Suppose **I simulate the following data for you:**

```r
set.seed(1)
alpha <- 10
beta <- 5
n <- 100
dat <- rgamma(n, shape = alpha, rate = beta)
```

Use the data to estimate the coefficient of variation and obtain nonparameteric and parameteric confidence intervals for the coefficient of variation.

The coefficient of variation in this case the mean divided by the standard deviation, which in this case is

$$\frac{\frac{\alpha}{\beta}}{\sqrt{\frac{\alpha}{\beta^2}}} = \sqrt{\alpha}\,.$$

Further, note that we can estimate α and β from the sample mean and sample variance. The mean divided by variance is β so we can estimate β with

$$\hat{\beta} = \frac{\text{sample mean}}{\text{sample variance}}\,.$$

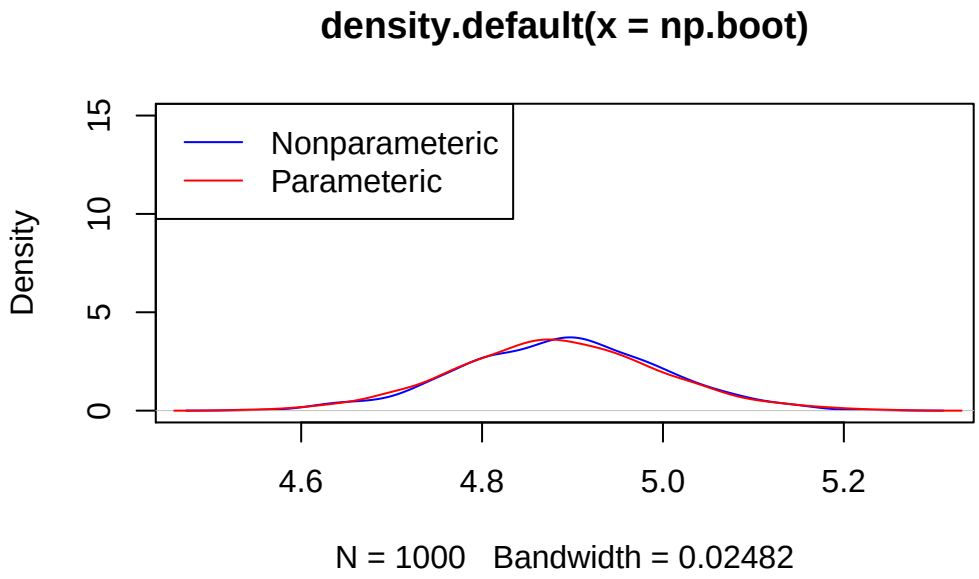
Further, we know the sample mean is α/β so

$$\hat{\alpha} = \hat{\beta} \cdot \text{Sample Mean}\,.$$

This will be used in parameteric bootstrap

```r
# truth
alp <- .05
ends <- c(alp/2, 1 - alp/2)

# will contain 1 or 0 based on whether the
# truth parameter was in the interval in each rep


## non-parametric bootstrap
B <- 1000
np.boot <- numeric(length = B)
for(b in 1:B)
{
  foo <- sample(1:n, size = n, replace = TRUE)
  boot.samp <- dat[foo]
  np.boot[b] <- mean(boot.samp)/sd(boot.samp)
```

```r
}

beta.hat <- mean(dat)/var(dat)
alpha.hat <- beta.hat * mean(dat)
p.boot <- numeric(length = B)
for(b in 1:B)
{
  boot.samp <- rgamma(n, shape = alpha.hat, rate = beta.hat)
  p.boot[b] <- mean(boot.samp)/sd(boot.samp)
}

plot(density(np.boot), type = 'l', ylim = c(0,15), col = "blue")
lines(density(p.boot), col = "red")
legend("topleft", legend = c( "Nonparameteric", "Parameteric"),
       col = c("blue", "red"), lty = 1)
```



Further, repeat the experiment multiple times to obtain the coverage probabilities of the nonparametric and parametric confidence intervals.

```r
alpha <- 10
beta <- 5
n <- 100
truth <- sqrt(alpha)

reps <- 100
npara <- numeric(length = reps)
```

```
para <- numeric(length = reps)

for(r in 1:reps)
{
  dat <- rgamma(n, shape = alpha, rate = beta)

  np.boot <- numeric(length = B)
  for(b in 1:B)
  {
    foo <- sample(1:n, size = n, replace = TRUE)
    boot.samp <- dat[foo]
    np.boot[b] <- mean(boot.samp)/sd(boot.samp)
  }

  # confidence intervals
  np.int <- quantile(np.boot, ends)
  npara[r] <- (truth >= np.int[1] && truth <= np.int[2])

  beta.hat <- mean(dat)/var(dat)
  alpha.hat <- beta.hat * mean(dat)
  p.boot <- numeric(length = B)
  for(b in 1:B)
  {
    boot.samp <- rgamma(n, shape = alpha.hat, rate = beta.hat)
    p.boot[b] <- mean(boot.samp)/sd(boot.samp)
  }
 p.int <- quantile(p.boot, ends)
  para[r] <- (truth >= p.int[1] && truth <= p.int[2])
}

# fairly good coverage.
mean(npara)
```

```
[1] 0.93
```

```
mean(para)
```

```
[1] 0.95
```

4. **Consider the `cars` dataset in car where the response variable is `dist` and the predictor is `speed`. We will fit a linear regression model for just this covariate (without intercept).**

   Fit a ridge regression model with $\lambda = 20$ and obtain $\hat{\beta}_{ridge}$. Assuming errors are normally

distributed, what is the distribution of $\hat{\beta}_{ridge}$? Obtain a parametric bootstrap estimate of the variance of $\hat{\beta}_{ridge}$ and compare with the theoretical variance.

The regression model for this univariate problem is

$$y_i = \beta x_i + \epsilon_i$$

where $\epsilon_i \sim N(0, \sigma^2)$. Now recall that for a general p dimensional problem the ridge solution was:

$$\hat{\beta}_{Ridge} = (X^T X + \lambda I_p)^{-1} X^T y$$

For the univariate situation, this will be:

$$\hat{\beta}_{Ridge} = \frac{\sum_{i=1}^n x_i y_i}{\sum_{i=1}^n x_i^2 + \lambda}$$

Now, since $y_i \sim N(x_i\beta, \sigma^2)$, we can find the distribution of $\hat{\beta}_{ridge}$

$$\hat{\beta}_{Ridge} \sim N\left(\frac{\sum_{i=1}^n x_i E(y_i)}{\sum_{i=1}^n x_i^2 + \lambda}, \frac{\sum_{i=1}^n x_i^2 \text{Var}(y_i)}{(\sum_{i=1}^n x_i^2 + \lambda)^2}\right) = N\left(\frac{\sum_{i=1}^n x_i^2 \beta}{\sum_{i=1}^n x_i^2 + \lambda}, \frac{\sum_{i=1}^n x_i^2 \sigma^2}{(\sum_{i=1}^n x_i^2 + \lambda)^2}\right)$$

Now we will compare the above variance with the parameteric bootstrap estimate of the variance. Unfortunately, there was a typo in the question, it should've been **non-parametric** estimator. First, σ^2 will need to be estimated, and we can estimate than from the usual way we estimate when we use MLE:

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

```
y <- cars$dist
X <- cars$speed
B <- 1000
lambda <- 20
n <- length(y)

#ridge
ridge <- sum(X*y)/(sum(X^2) + lambda)

np.boot <- numeric(length = B)
for(b in 1:B)
{
    foo <- sample(1:n, size = n, replace = TRUE)
```

```
    y.samp <- y[foo]
    X.samp <- X[foo]
    np.boot[b] <- sum(X.samp*y.samp)/(sum(X.samp^2) + lambda)
}

# estimating sigma^2
sig2 <- mean( (y - X*ridge)^2)

# true variance
(true.var <- sum(X^2*sig2)/(sum(X^2) + lambda)^2)
```

```
[1] 0.01952668
```

```
# variance of np.boot
# fairly close to the truth
var(np.boot)
```

```
[1] 0.02376819
```

The theoretical variance is fairly close to the bootstrap variance.

Repeat the above process for a lasso regression estimate for $\lambda = 20$. Do we know the distribution of the estimator here?

For fitting the lasso we will require the Bridge regression. However, for lasso, it is not clear what the distribution of this will be! Thus, the true variance is not known here. So we will just estimate the non-parameteric variance.

```
# we need to change the code so that
# this works in the 1 dimension
bridgeReg <- function(y.this, X.this, alpha = 1, lambda, max.iter, tol = 1e-3)
{
  # a value larger than tol
  distance <- tol + 1
  iter <- 0

  # starting from the ridge solution
  current <- sum(X.this*y.this)/(sum(X.this^2) + lambda)

  while(distance > tol)
  {
    iter <- iter + 1
    if(iter > max.iter)
    {
```

```
      print("Maximum iterations reached")
      break
    }

    # MM steps
    previous <- current
    mjs <- alpha* abs(current)^(alpha - 2)

    ## using qr.solve since that is more stable than solve
    current <- sum(X.this*y.this)/(sum(X.this^2) + lambda/alpha * mjs)
    distance <- (previous - current)^2
  }
  #returning the last iterate of the
  return(current)
}

y <- cars$dist
X <- cars$speed
B <- 1000
lambda <- 20
n <- length(y)


#lasso
lasso <- bridgeReg(y, X, alpha = 1, lambda = 20, max.iter = 1e3)

np.boot <- numeric(length = B)
for(b in 1:B)
{
    foo <- sample(1:n, size = n, replace = TRUE)
    y.samp <- y[foo]
    X.samp <- X[foo]
    np.boot[b] <- bridgeReg(y.samp, X.samp, alpha = 1, lambda = 20, max.iter = 1e3)
}

# true variance is not known!
# variance of np.boot
# fairly close to the truth
var(np.boot)
```

```
[1] 0.02424121
```