HW4

Natalie Brewer

2023-09-20

Problem 4F

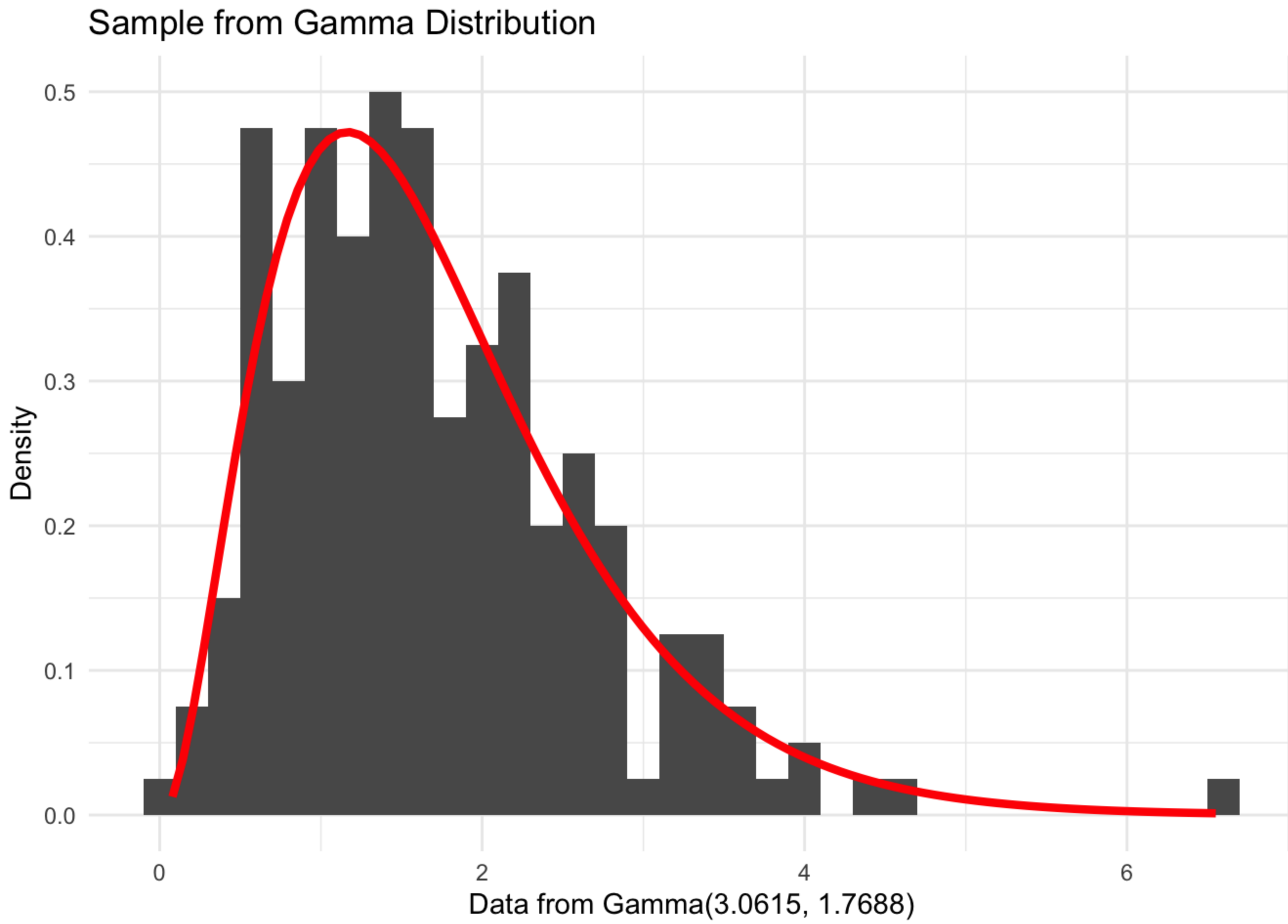
```
#Pick a random r and lambda
set.seed(69)
r <- runif(1,2,4)
r
```

```
## [1] 3.061508
```

```
lambda <- runif(1,1,2)
lambda
```

```
## [1] 1.768808
```

```
#Generate a 200 size sample from this Gamma dist.
sample <- rgamma(200, shape=r, rate=lambda)

sample_df <- data.frame(sample)

ggplot(sample_df, aes(x=sample)) +
  geom_histogram(aes(y=after_stat(density)), binwidth = .2) +
  labs(title="Sample from Gamma Distribution", x="Data from Gamma(3.0615, 1.7688)", y="D
ensity") +
  stat_function(fun=dgamma, args=list(shape=r, rate=lambda), color="red", linewidth=1.5)
+
  theme_minimal()
```



Problem 4G

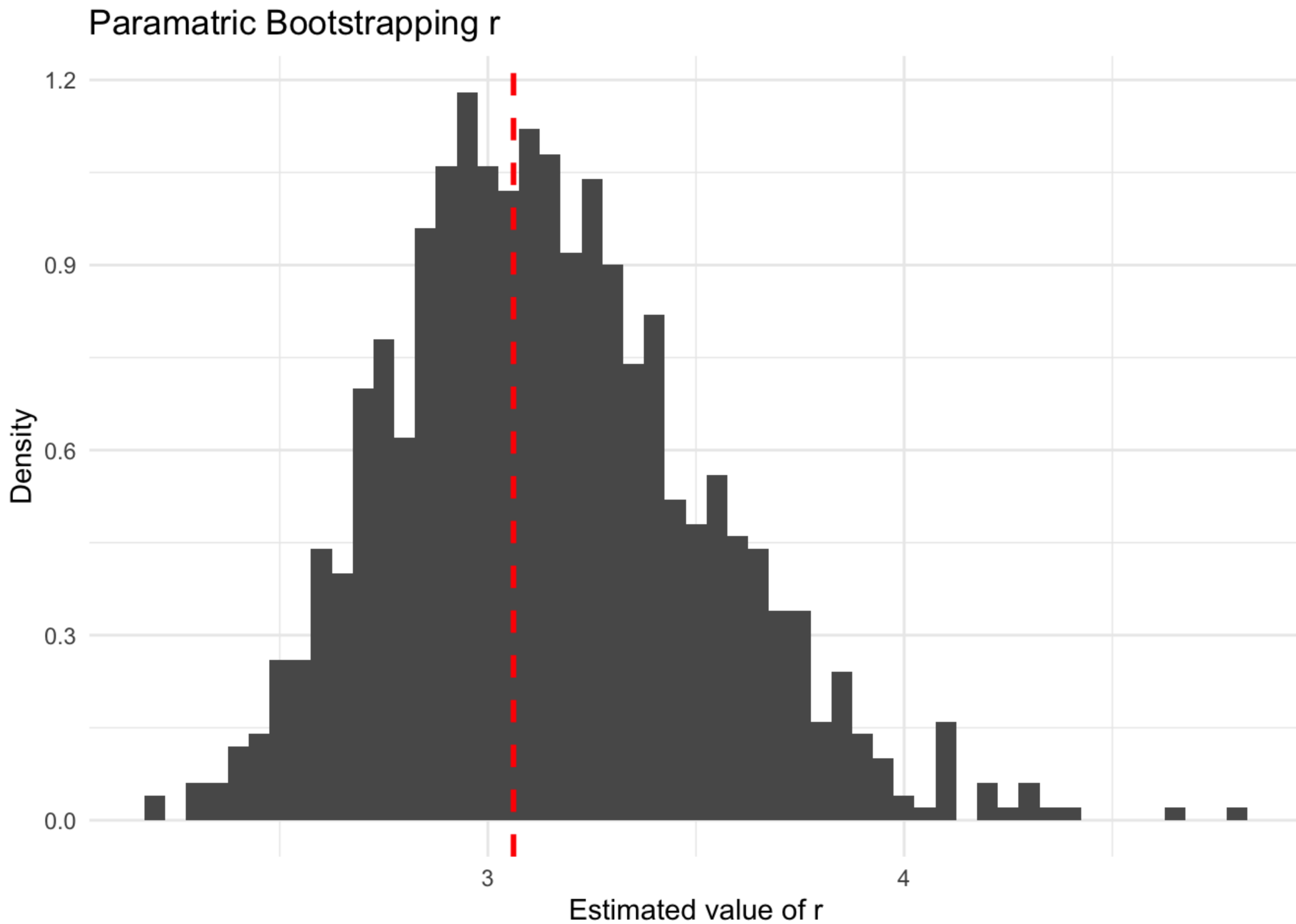
```
#Compute the MOM estimators for r and lambda
xbar <- mean(sample)
sample_var <- var(sample)

r_MOM <- (xbar)^2 / sample_var
lambda_MOM <- xbar / sample_var

r_MOM
```

```
## [1] 3.107342
```

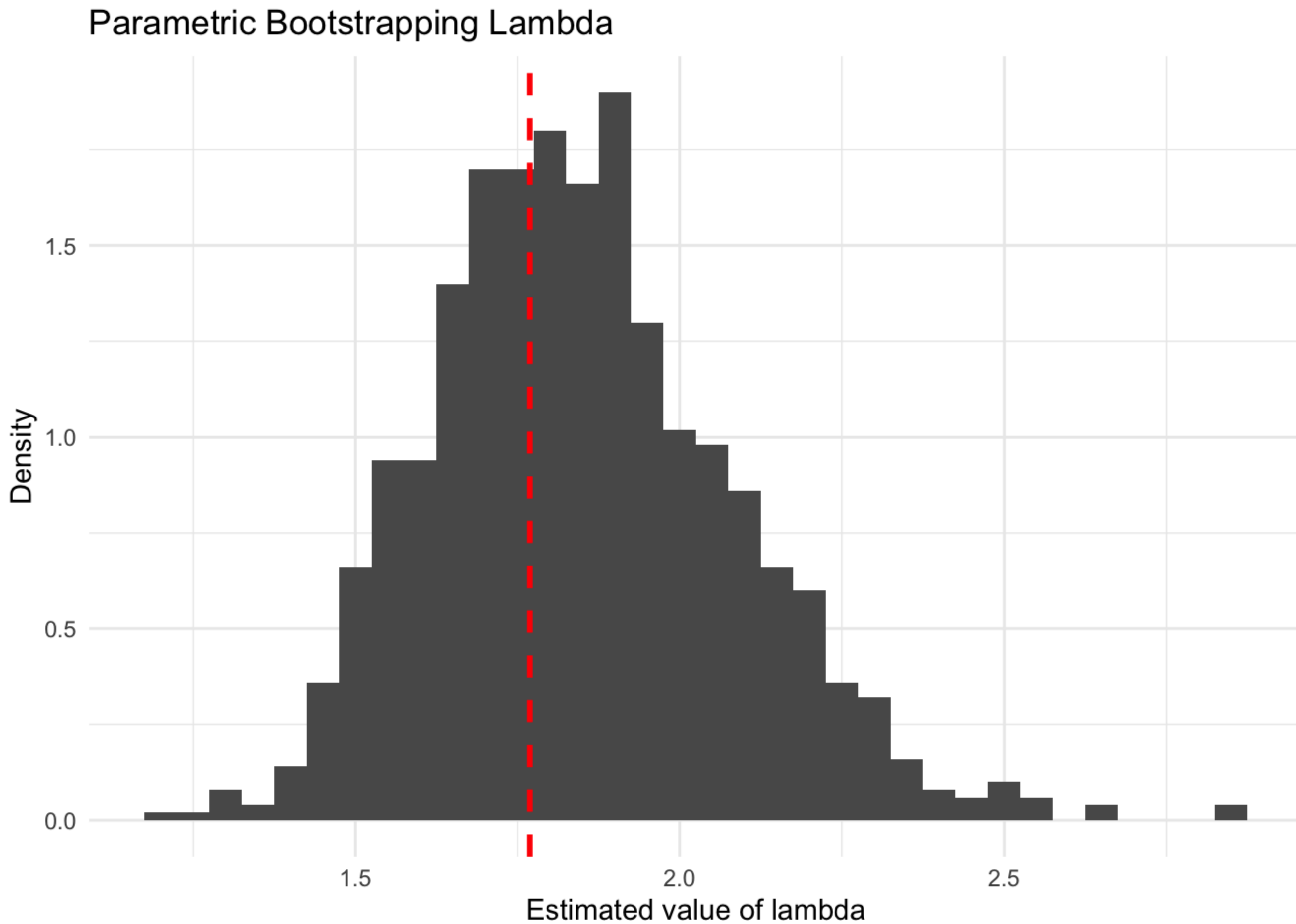
```
lambda_MOM
```

```
## [1] 1.824389
```

```
find_estimators <- function(){
  new_sample <- rgamma(200, shape=r_MOM, rate=lambda_MOM)

  new_xbar <- mean(new_sample)
  new_sample_var <- var(new_sample)

  new_r_MOM <- (new_xbar)^2 / new_sample_var
  new_lamda_MOM <- new_xbar / new_sample_var

  return(c(new_r_MOM,new_lamda_MOM))
}

parametric_df <- data.frame(
  r_estimate = c(),
  lambda_estimate = c()
)

for (i in 1:1000) {
  estimators <- find_estimators()
  parametric_df <- rbind(parametric_df, data.frame(r_estimate=estimators[1], lambda_esti
mate=estimators[2]))
}

#Draw histogram for r estimates
ggplot(parametric_df, aes(x=r_estimate)) +
  geom_histogram(aes(y=after_stat(density)), binwidth = .05) +
  labs(title="Paramatric Bootstrapping r", x="Estimated value of r", y="Density") +
  geom_vline(aes(xintercept = r), color = "red", linetype = "dashed", linewidth = 1) +
  theme_minimal()
```




```
#Draw histogram for lambda estimates
ggplot(parametric_df, aes(x=lambda_estimate)) +
  geom_histogram(aes(y=after_stat(density)), binwidth = .05) +
  labs(title="Parametric Bootstrapping Lambda", x="Estimated value of lambda", y="Densit
y") +
  geom_vline(aes(xintercept = lambda), color = "red", linetype = "dashed", linewidth =
1) +
  theme_minimal()
```



It looks like these distributions can be approximated by the normal distribution. Like Figure 8.4, the shape is approximately normal, centered close to the MOM estimate of the parameter that we used to build our gamma distribution to draw our 1000 samples. The locations of the actual values of parameters is not surprising because they are close to the center of the distribution. This is actually because our MOM estimators are quite close to the true values and these MOM estimates are what we used to generate the gamma distribution from which we drew our samples.

```
#Approximate the standard errors
SE_r <-  sd(parametric_df$r_estimate)
SE_lambda <- sd(parametric_df$lambda_estimate)

SE_r
```

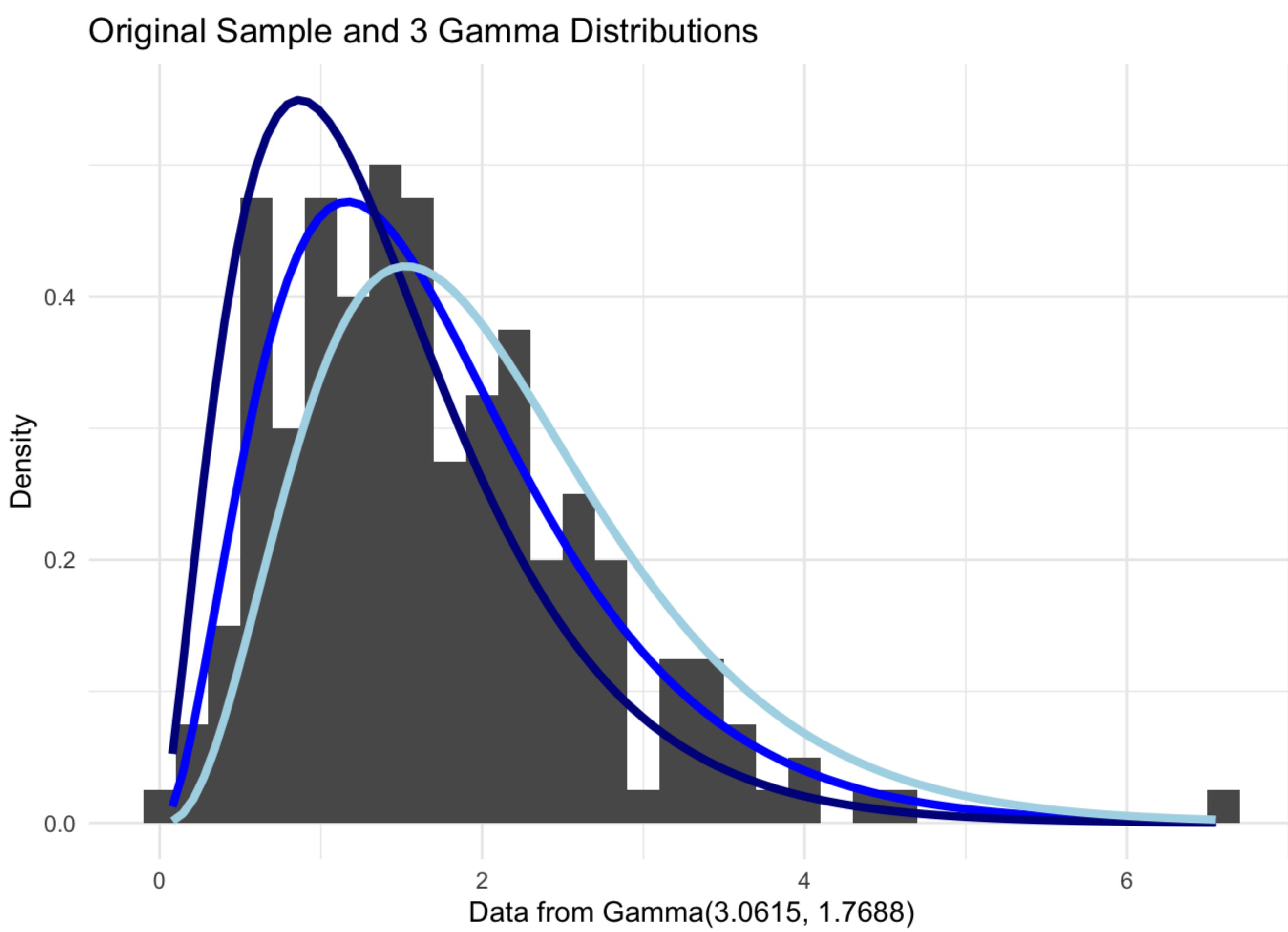
```
## [1] 0.377023
```

```
SE_lambda
```

```
## [1] 0.2328184
```

Problem 4H

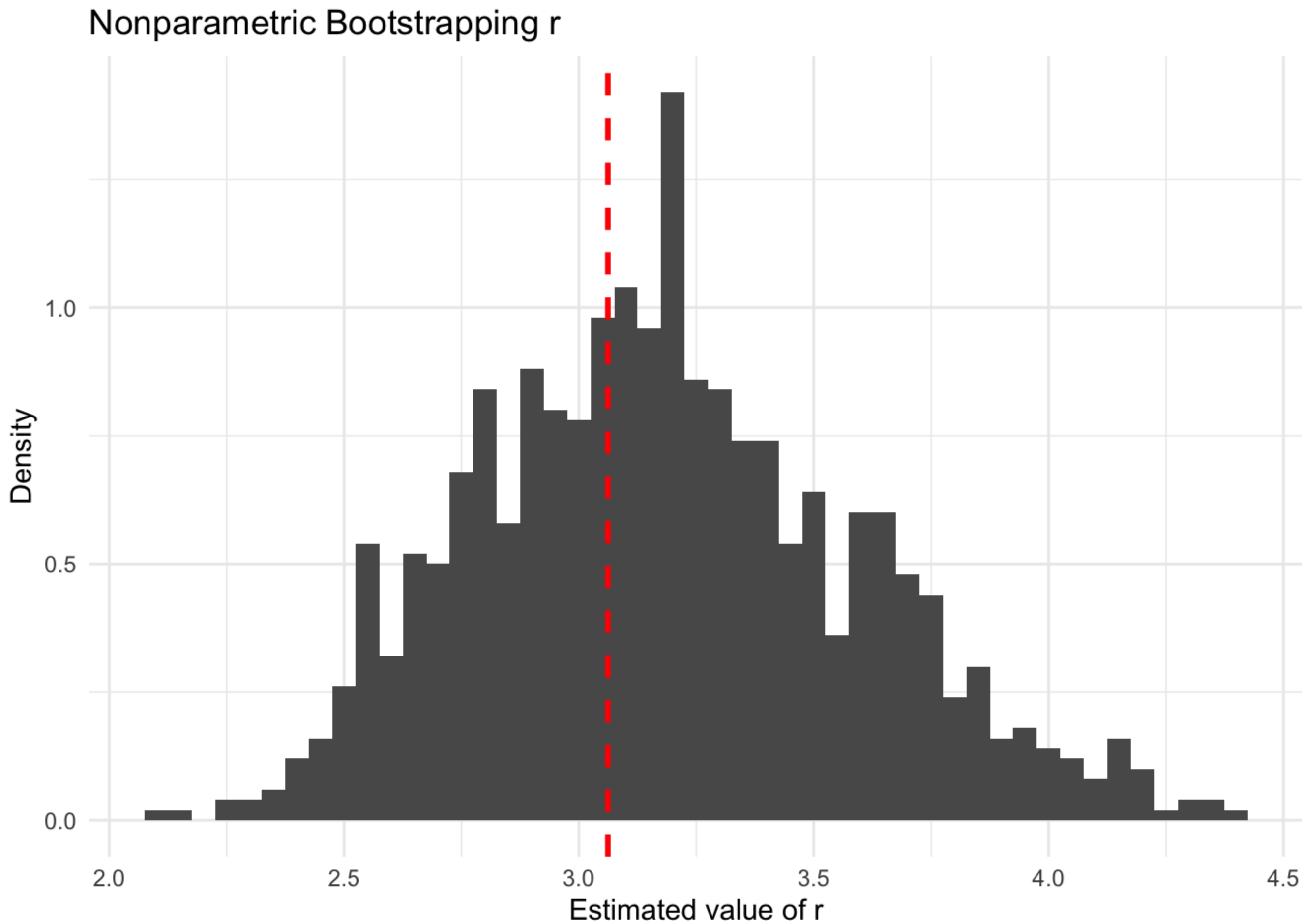
```
# find 5th and 95 percentiles of distribution
a <- quantile(parametric_df$r_estimate, .05)
b <- quantile(parametric_df$r_estimate, .95)

#Draw histogram for original sample with 3 gamma densities
ggplot(sample_df, aes(x=sample)) +
  geom_histogram(aes(y=after_stat(density)), binwidth = .2) +
  labs(title="Original Sample and 3 Gamma Distributions", x="Data from Gamma(3.0615, 1.7
688)", y="Density") +
  stat_function(fun=dgamma, args=list(shape=r, rate=lambda), color="blue", linewidth=1.
5) +
  stat_function(fun=dgamma, args=list(shape=a, rate=lambda_MOM), color="darkblue", linew
idth=1.5) +
  stat_function(fun=dgamma, args=list(shape=b, rate=lambda_MOM), color="lightblue", line
width=1.5) +
  theme_minimal()
```



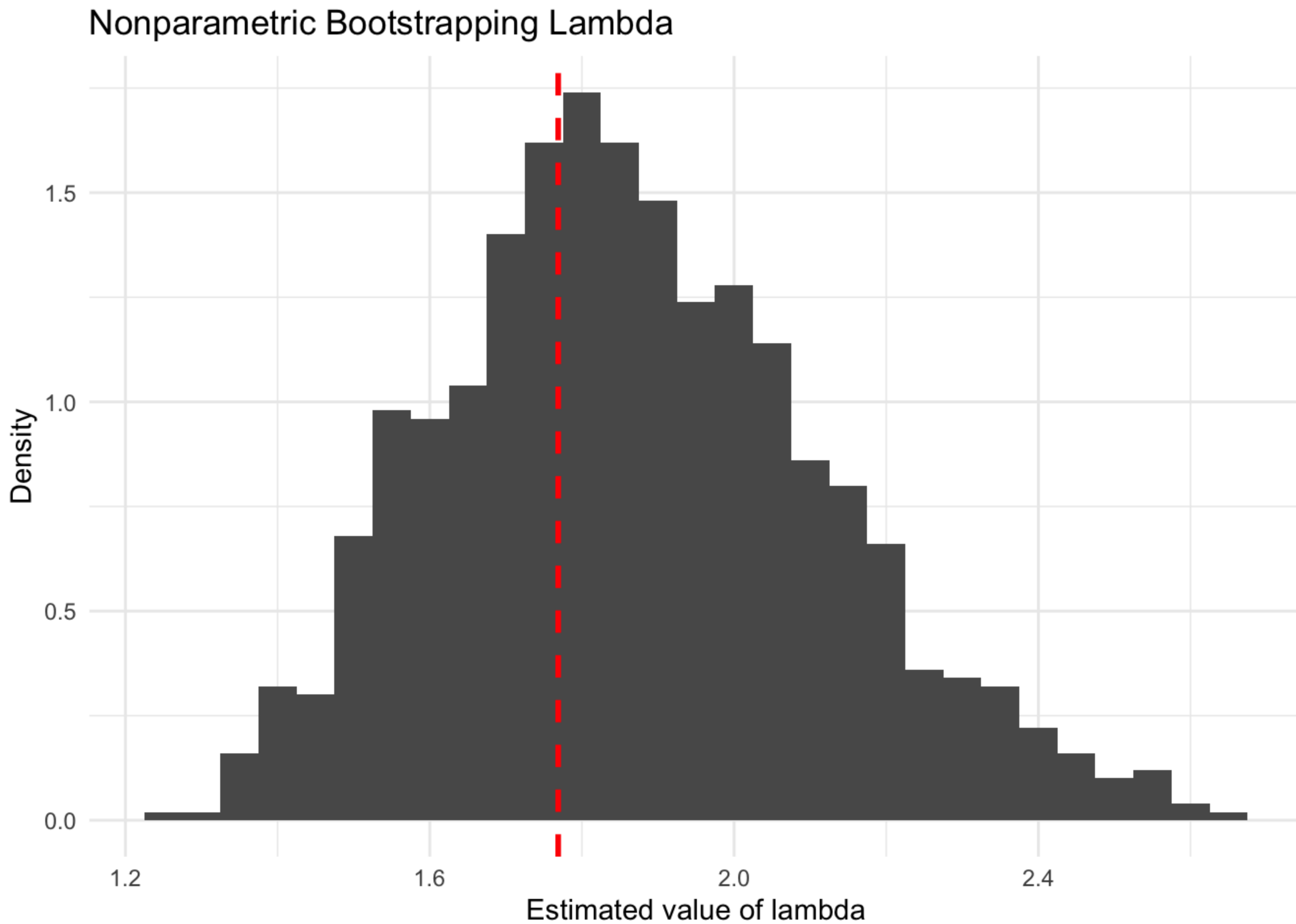
There is nothing too surprising about this data. We can see that the 5th percentile version underestimates the true value of r, making the curve skewed more to the left. The 95th percentile version overestimates the value of r, making the curve more skewed to the right.

Problem 4l

```
find_estimators_nonparam <- function(){
  new_sample <- sample(sample,200,replace=T)

  new_xbar <- mean(new_sample)
  new_sample_var <- var(new_sample)

  new_r_MOM <- (new_xbar)^2 / new_sample_var
  new_lamda_MOM <- new_xbar / new_sample_var

  return(c(new_r_MOM,new_lamda_MOM))
}

nonparam_df <- data.frame(
  r_estimate_nonparam = c(),
  lambda_estimate_nonparam = c()
)

for (i in 1:1000) {
  estimators <- find_estimators_nonparam()
  nonparam_df <- rbind(nonparam_df, data.frame(r_estimate_nonparam=estimators[1], lambda
_estimate_nonparam=estimators[2]))
}

#Draw histogram for r estimates
ggplot(nonparam_df, aes(x=r_estimate_nonparam)) +
  geom_histogram(aes(y=after_stat(density)), binwidth = .05) +
  labs(title="Nonparametric Bootstrapping r", x="Estimated value of r", y="Density") +
  geom_vline(aes(xintercept = r), color = "red", linetype = "dashed", linewidth = 1) +
  theme_minimal()
```




```
#Draw histogram for lambda estimates
ggplot(nonparam_df, aes(x=lambda_estimate_nonparam)) +
  geom_histogram(aes(y=after_stat(density)), binwidth = .05) +
  labs(title="Nonparametric Bootstrapping Lambda", x="Estimated value of lambda", y="Den
sity") +
  geom_vline(aes(xintercept = lambda), color = "red", linetype = "dashed", linewidth =
1) +
  theme_minimal()
```



The shapes of these histograms is very similar to the shapes of the histograms from problem 4G. Again, the true values of the parameters r and lamba are close to the center of the histogram. One difference is that the nonparametric histograms appear to have less spread than the parametric ones. This is further evidenced by our calculation of the standard errors below

```
#Approximate the standard errors
SE_r_nonparam <-  sd(nonparam_df$r_estimate)
SE_lambda_nonparam <- sd(nonparam_df$lambda_estimate)

SE_r_nonparam
```

```
## [1] 0.4068081
```

```
SE_lambda_nonparam
```

```
## [1] 0.2496493
```

Problem 4J

```
fun <-  function(para, x) {
    r_para <-  para[1]
    lambda_para <-  para[2]
    -sum(r_para * log(lambda_para) + (r_para - 1) * log(x) - lambda_para * x - log(gamma
(r_para)))
}


mle <-  optim(par = c(r_MOM, lambda_MOM), fn = fun, x = sample)

r_MLE <- mle$par[1]
lambda_MLE <- mle$par[2]

r_MLE
```

```
## [1] 2.997137
```

```
lambda_MLE
```

```
## [1] 1.759656
```

These MLE estimates are very close to the true values of r and lambda. r_MLE is off by just:

```
r - r_MLE
```

```
## [1] 0.06437108
```

and lambda_MLE is off by just:

```
lambda - lambda_MLE
```

```
## [1] 0.00915168
```