STAT 206 Homework 4

Xin Feng(Vanessa)

10/25/2019

Due Monday, October 28, 5:00 PM

General instructions for homework: Homework must be submitted as pdf file, and be sure to include your name in the file. Give the commands to answer each question in its own code block, which will also produce plots that will be automatically embedded in the output file. Each answer must be supported by written statements as well as any code used. (Examining your various objects in the "Environment" section of RStudio is insufficient – you must use scripted commands.)

In lecture, we fit a gamma distribution to the weight of cat's hearts. We did this by adjusting the parameters so that the theoretical values of the mean and variance matched the observed, sample mean and variance. Since the mean and variance are the first two moments of the distribution, this is an example of the method of moments for estimation.

The method of moments gives a point estimate $\hat{\theta}$ of the parameters θ. To use a point estimate, we need to know how precise it is, i.e., how different it would be if we repeated the experiment with new data from the same population. We often measure imprecision by the standard error, which is the standard deviation of the point estimates $\hat{\theta}$. (You saw the standard error of the mean in your introductory statistics classes, but we are not computing the standard error of the mean here.)

If we actually did the experiment many times, getting many values of $\hat{\theta}$, we could take their standard deviation as the standard error. With only one data set, we need to do something else. There is usually no simple formula for standard errors of most estimates, the way there is for the standard error of the mean. Instead, we will see how to approximate the standard error of for our estimate of the gamma distribution computationally.

We can draw random values from a gamma distribution using the `rgamma()` function. For example, `rgamma(n=35,shape=0.57,scale=15)` would generate a vector of 35 random values, drawn from the gamma distribution with "shape" parameter a = 0.57 and "scale" s = 15. By applying the estimator to random samples drawn from the distribution, we can see how much the estimates will change purely due to noise.

Part I - Estimates and standard errors

1. Write a function, `gamma.est`, which takes as input a vector of data values, and returns a vector containing the two estimated parameters of the gamma distribution, with components named shape and scale as appropriate.

```
gamma.est <- function(value_x){
  value_mean <- mean(value_x)
  value_var <- var(value_x)
  shape <- value_mean^2/value_var
  scale <- value_var/value_mean
  return(c(shape, scale))
}
```

2. Verify that your function implements the appropriate formulas by showing that it matches the results from lecture for the cat heart data.

```r
library(MASS)
data(cats)
cats_para <- gamma.est(cats$Hwt)
cat("Estimate of a: ", cats_para[1], "\n")
```

```
## Estimate of a:  19.06531
```

```r
cat("Estimate of s: ", cats_para[2], "\n")
```

```
## Estimate of s:  0.5575862
```

```r
# The results are same as the one in Lab 3.
```

3. Generate a vector containing ten thousand random values from the gamma distribution with $a = 19$ and $s = 0.56$. What are the theoretical values of the mean and of the variance? What are their sample values?

```r
a <- 19
s <- 0.56
data_gamma <- rgamma(n=1e+4, shape=a, scale=s)
# Theoretical values of mean and variance:
cat("Mean=as: ", a*s, "\n")
```

```
## Mean=as:  10.64
```

```r
cat("Variance=as^2: ", a*s^2, "\n")
```

```
## Variance=as^2:  5.9584
```

```r
# Sample values of mean and variance:
cat("Mean: ", mean(data_gamma), "\n")
```

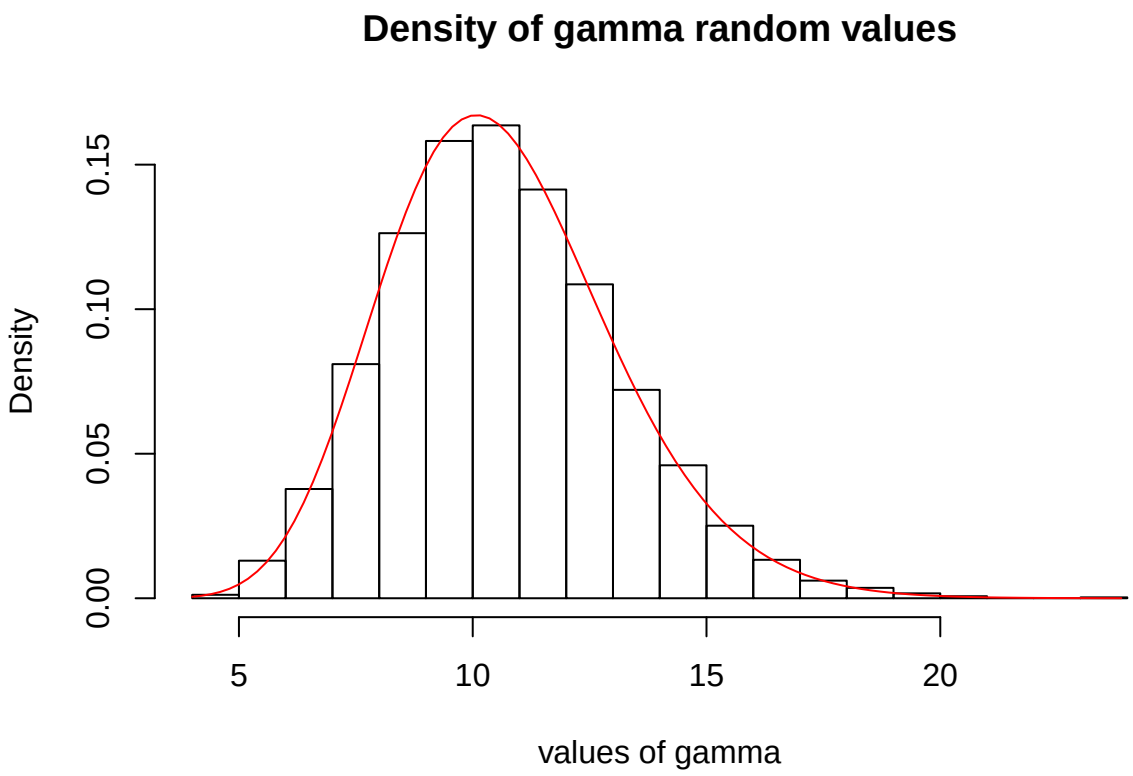
```
## Mean:  10.6807
```

```r
cat("Variance: ", var(data_gamma), "\n")
```

```
## Variance:  6.090792
```

4. Plot the histogram of the random values, and add the curve of the theoretical probability density function.

```r
hist(data_gamma, main="Density of gamma random values", xlab="values of gamma",
     probability=T)
curve(dgamma(x,shape=a,scale=s),
      from=min(data_gamma), to=max(data_gamma), add=T, col=2)
```



5. Apply your `gamma.est` function to your random sample. Report the estimated parameters and how far they are from the true values.

```
result <- gamma.est(data_gamma)
cat("Estimated of shape: ", result[1], ", and true value is: ", a, ".
    The difference between them is: ", abs(result[1]-a), "\n")
```

```
## Estimated of shape:  18.72949 , and true value is:  19 .
##     The difference between them is:  0.2705083
```

```
cat("Estimated of scale: ", result[2], ", and true value is: ", s, ".
    The difference between them is: ", abs(result[2]-s), "\n")
```

```
## Estimated of scale:  0.5702613 , and true value is:  0.56 .
##     The difference between them is:  0.01026128
```

```
# According the result, the differences between these two groups of value are little.
```

6. Write a function, `gamma.est.se`, to calculate the standard error of your estimates of the gamma parameters, on simulated data drawn from the gamma distribution. It should take the following arguments: true shape parameter `shape` (or a), true scale parameter `scale` (or s), size of each sample `n`, and number of repetitions at that sample size `B`. It should return two standard errors, one for the shape parameter a and one for the scale parameter s. (These can be either in a vector or in a list, but should be named clearly.) It should call a function `gamma.est.sim` which takes the same arguments as `gamma.est.se`, and returns an array with two rows and B columns, one row holding shape estimates and the other row scale estimates. Your `gamma.est.se` function should not, itself, estimate any parameters or generate any random values

```
gamma.est.sim <- function(a, s, n, B){
  est_a <- c()
  est_s <- c()
  for(i in 1:B){
    value_gamma <- rgamma(n, a, s)
    est_para <- gamma.est(value_gamma)
    est_a <- c(est_a, est_para[1])
    est_s <- c(est_s, est_para[2])
  }
  est_result <- rbind(est_a, est_s)
  dimnames(est_result) <- list(rownames=c("shape","scale"))
  return(est_result)
}

gamma.est.se <- function(a, s, n, B){
  est_result <- gamma.est.sim(a, s, n, B)
  sd_a <- sd(est_result[1,])
  sd_s <- sd(est_result[2,])
  return (c(sd_a, sd_s))
}

# Generate 1000 random value following gamma distribution, and get the estimate of a and s.
# Cycle 100 times, calculate the standard error of the estimates of the gamma parameters.
gamma.est.se(a, s, 1e+3, 100)
```

```
## [1] 0.85935136 0.07949319
```

Part II - Testing with a *stub*

To check that `gamma.est.se` works properly, write a *stub* or *dummy* version of `gamma.est.sim`, which takes the correct arguments and returns an array of the proper size, but whose entries are fixed so that it's easy for us to calculate what `gamma.est.se` ought to do.

7. Write `gamma.est.sim` so that the entries in the first row of the returned array alternate between `shape` and `shape+1`, and those in the second row alternate between `scale` and `scale+n`. For example `gamma.est.sim(2,1,10,10)` should return (row names are optional)

```
##         [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
## shapes    2    3    2    3    2    3    2    3    2     3
## scales    1   11    1   11    1   11    1   11    1    11
```

and `gamma.est.sim(2,8,5,7)` should return

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7]
## [1,]    2    3    2    3    2    3    2
## [2,]    8   13    8   13    8   13    8
```

```
gamma.est.sim <- function(a, s, n, B){
  est_a <- c()
  est_s <- c()
```

```
  shapes <- rep(c(a, a+1), B)
  scales <- rep(c(s, s+n), B)
  value_para <- rbind(shapes, scales)
  for(i in 1:B){
    value_gamma <- rgamma(n, value_para[1,i], value_para[2,i])
    est_para <- gamma.est(value_gamma)
    est_a <- c(est_a, est_para[1])
    est_s <- c(est_s, est_para[2])
  }
  est_result <- rbind(est_a, est_s)
  rownames(est_result) <- c("shape","scale")
  return(est_result)
}
```

8. Calculate the standard deviations of each *row* in the two arrays above.

```
test_array_1 <- gamma.est.sim(2, 1, 10, 10)
cat("Test 1-Standard deviation of a: ", sd(test_array_1[1,]), "\n")
```

```
## Test 1-Standard deviation of a:  2.481263
```

```
cat("Test 1-Standard deviation of s: ", sd(test_array_1[2,]), "\n")
```

```
## Test 1-Standard deviation of s:  0.2704128
```

```
test_array_2 <- gamma.est.sim(2, 8, 5, 7)
cat("Test 2-Standard deviation of a: ", sd(test_array_2[1,]), "\n")
```

```
## Test 2-Standard deviation of a:  119.1033
```

```
cat("Test 2-Standard deviation of s: ", sd(test_array_2[2,]), "\n")
```

```
## Test 2-Standard deviation of s:  0.06355579
```

9. Run your `gamma.est.se`, with this version of `gamma.est.sim`. Do its standard errors match the standard deviations you just calculated? Should they?

```
gamma.est.se(2, 1, 10, 10)
```

```
## [1] 2.512000 0.734428
```

```
gamma.est.se(2, 1, 10, 10)
```

```
## [1] 1.4445867 0.1961109
```

```
# Because the function gamma.est.se has the process of gamma.est.sim,
# it will re-create random values following gamma distribution.
# Thus, The result of function gamma.est.se is very close to the result of
# the standard deviations last question,
# but due to the randomness of the number of generations, it is not exactly equal.
```

Part III - Replacing the stub

10. Write the actual `gamma.est.sim`. Each of the B columns in its output should be the result of applying `gamma.est` to a vector of `n` random numbers generated by a different call to `rgamma`, all with the same shape and scale parameters.

```
gamma.est.sim <- function(a, s, n, B){
  est_a <- c()
  est_s <- c()
  for(i in 1:B){
    value_gamma <- rgamma(n, a, s)
    est_para <- gamma.est(value_gamma)
    est_a <- c(est_a, est_para[1])
    est_s <- c(est_s, est_para[2])
  }
  est_result <- rbind(est_a, est_s)
  dimnames(est_result) <- list(rownames=c("shape","scale"))
  return(est_result)
}
```

11. Run `gamma.est.se`, calling your new `gamma.est.sim`, with `shape=2`, `scale=1`, `n=10` and `B=1e5`. Check that the standard error for `shape` is approximately 1.6 and that for `scale` approximately 0.54. Explain why your answers are not exactly 1.6 and 0.54.

```
test_3 <- gamma.est.se(2, 1, 10, 1e5)
cat("The result (", test_3[1], ", ", test_3[2], ") is approximately (1.6, 0.54)\n")
```

```
## The result ( 1.641958 ,  0.5437387 ) is approximately (1.6, 0.54)
```

```
# During the 1e5 cycling, each time we create new group of values following the
# same gamma distribution, due to the randomness of the number of generations,
# it is not exactly equal, but the difference is little.
```