Seminar 2

# 1 Synopsis.

**What this session is about.** In the previous seminar you saw how to estimate things using the Monte Carlo method. You now have to use this technique again in conjunction with the theory to examine relative efficiency of moment and maximum likelihood estimators.

You will see on a simple example that, as expected from the theory, MLE estimators outperform moment estimators, at least for large sample size. For small sample sizes, this may not be the case.

**What you need.** You need to be fully acquainted with the content of previous seminars and practice assignment. You will also need access to a computer equiped with R.

# 2 Background

Assume we have data with a histogram like that in Figure 1. If we were to fit a distribution to such data, a good initial choice might be a $\gamma(a, r)$ which we know can have a similar shape for well chosen $a$ and $r$.

## 2.1 Moment estimator of $a, r$

Since (check the theory) the mean of a $\gamma(a, r)$ is $r/a$, and the variance is $r/a^2$, the moment estimators of $a$ and $r/a$ are quite simple, and can be obtained from the two equations

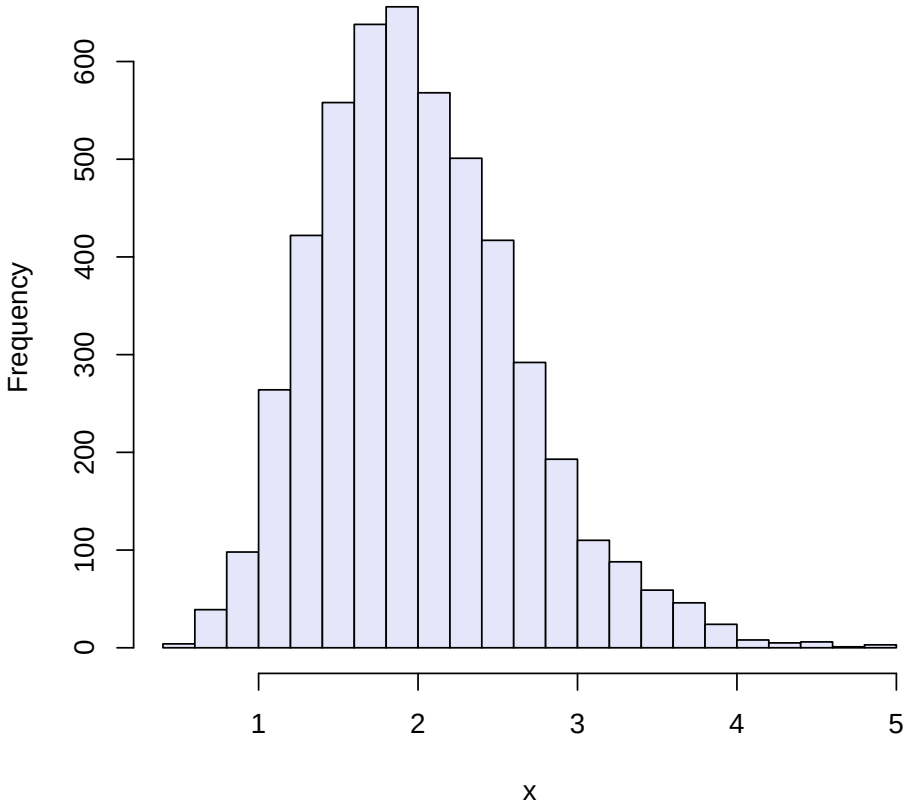
$$r/a = \overline{X} \tag{1}$$

$$r/a^2 + r^2/a^2 = \frac{1}{n}\sum_{i=1}^{n} X_i^2 \tag{2}$$

## 2.2 Maximum likelihood estimators of $a, r$

Maximum likelihood estimators are much more complex in this case, but do not be intimidated by the formidable-looking formulae below! Conceptually, it is quite simple. Remember that the density

Figure 1: Histogram of data to fit




of the distribution $\gamma(a, r)$ is:

$$f(x) = \frac{a^r}{\Gamma(r)} x^{r-1} e^{-ax}$$

Hence, the likelihood associated with a sample $(x_1, \ldots, x_n)$ is:

$$L\left(r; x_1, \ldots, x_n\right) = \frac{a^{nr}}{(\Gamma(r))^n} \left[\prod_{i=1}^{n} x_i^{(r-1)}\right] e^{-a\sum_{i=1}^{n} x_i}, \tag{3}$$

and the log-likelihood:

$$\ell\left(r; x_1, \ldots, x_n\right) = nr \log\left(a\right) - n \log \Gamma\left(r\right) + \sum_{i=1}^{n} \left(r-1\right) \log(x_i) - a \sum_{i=1}^{n} x_i. \tag{4}$$

### 2.2.1 The usual approach

We *might* take the derivatives of (4), equate them to zero and solve for $r$ and $a$. For $a$ we would have

$$\frac{nr}{a} - \sum_{i=1}^{n} x_i = 0; \tag{5}$$

this gives for $a$ the same as the moment estimator (check (1)).

However, when we take the derivative with respect to $r$ we have:

$$n \log(a) - n\psi(r) + \sum_{i=1}^{n} \log(x_i) = 0 \tag{6}$$

where

$$\psi(r) = \frac{\partial \log \Gamma(r)}{\partial r} \tag{7}$$

is the so-called psi (or digamma) function. Since $r$ is buried under $\psi(r)$, (6) is hard to solve for $r$.

### 2.2.2 Directly maximizing the (log) likelihood

A better alternative in this case might be to maximize the likelihood directly, giving function $\ell(r; x_1, \ldots, x_n)$ to a computer routine that finds the maximum.

First we have to code (4):

```
> gamma.lik <- function(par, x) {
+  a <- par[1] ; r <- par[2]
+  n <- length(x)
+  ll <- n*r*log(a) - n*log(gamma(r)) +
+        (r-1)*sum(log(x)) - a*sum(x)
+  return(ll)
+  }
```

Notice: we pass the two parameters in the form of a vector `par`; a second argument, `x`, contains the values of the sample to which the likelihood is associated.

Once we have coded the log likelihood, we pass it to function `optim`. This is a general purpose function which optimizes the value of a function with respect to the parameters passed in `par`. We will first generate data from a $\gamma(a = 5, r = 10)$ and then estimate the (known, because we have set their value) parameters:

```
> sample <- rgamma(5000, rate=5, shape=10)
> mean(sample)            #  Should be close to 10/5

[1] 1.998211

> var(sample)             #  Should be close to 10/5^2 = 0.40

[1] 0.3937126
```

Now we can invoke `optim`. The first argument is a vector with initial guesses of the values of the parameters, the second argument the name of the function to optimize, the third some control values. The `fnscale=-1` means that values returned from the function optimized are multiplied by -1. This is done because, left to its own devices, `optim` *minimizes* a function. A way to obtain the desired maximization is to multiply by -1.

```
> par <- c(1,1)
> optim(par,gamma.lik,control=list(fnscale=-1), x=sample)

$par
[1]  5.113283 10.217077

$value
[1] -4578.748

$counts
function gradient
      75       NA

$convergence
[1] 0

$message
NULL
```

The values that maximize the function are returned in `$par`. We can see that they are close to the (in this case) known true values of 5 and 10.

For the sake of comparison, we can compute the moment estimators. Replacing (1) in (2),

$$\overline{X}/a + \overline{X}^2 = \frac{1}{n}\sum_{i=1}^{n} X_i^2 \tag{8}$$

from which,

$$\begin{aligned} a &= \frac{\overline{X}}{\frac{1}{n}\sum_{i=1}^{n} X_i^2 - \overline{X}^2} \\ r &= \frac{\overline{X}^2}{\frac{1}{n}\sum_{i=1}^{n} X_i^2 - \overline{X}^2} \end{aligned}$$

In this case,

```
> n <- length(sample)
> a <- mean(sample) / ( sum(sample^2)/n - mean(sample)^2 )
> a

[1] 5.076319

> r <- a * mean(sample)
> r

[1] 10.14356
```

In this particular instance, the moment estimates are a bit closer to the true values than the MLE. Comparison would require to compute both estimators a large number of times, much as you did in Activity 1.