

Review

The for loop has two components, a value `val` and a sequence of values `sequence`:

```
sequence = 1:5
for( val in sequence)
{
  print(val)
}
```

`val` will equal `sequence[1]` on the first execution of the block of code inside the braces, then `val` will equal `sequence[2]` for the second execution of the block of code inside the braces, etc.

```
## Here val is the value
## and c(0.25,1.8,9.75,10000) is the sequence
sequence = c(0.25,1.8,9.75,10000)
for( val in sequence)
{
  print(val)
}
```

```
## Here k is the value
## and 1:5 is the sequence
for( k in 1:5)
{
  print(k)
}
```

```
## Here k is the value
## and 1:3 is the sequence
N = c(10,20,30)
for (k in 1:length(N))
{
  n = N[k]
  print(n)
}
```

```
## Here n is the value
## and c(10,20,30) is the sequence
N = c(10,20,30)
for (n in N)
```

```
{
  print(n)
}
```

Clustered Data

Suppose we observe N clusters, each consisting of n values. This means we observe realizations of N groups of independent random variables. For notation, let Y_{ij} be the j^{th} random variable in cluster i . We have that:

$$\begin{aligned} \text{cluster 1} &= \{Y_{11}, Y_{12}, \dots, Y_{1n}\} \\ &\vdots \\ \text{cluster } i &= \{Y_{i1}, Y_{i2}, \dots, Y_{in}\} \\ &\vdots \\ \text{cluster } N &= \{Y_{N1}, Y_{N2}, \dots, Y_{Nn}\} \end{aligned}$$

The random variables in cluster i all have expected value μ_i , i.e. $EY_{ij} = \mu_i$. Also assume that $\mu_i < \mu_{i+1}$, $i = 1, \dots, N$. Additionally require that $\text{var}Y_{ij} = 1$, $\forall i, j$.

For simulation of this scenario, for each realization, randomly generate the μ_i by sorting N independent realizations from a standard normal distribution.

To do this in R

Let N denote the number of clusters and let n denote the number of values in each cluster, which is the same for all clusters. To generate a single realization of this scenario, letting the random variables in cluster i have distribution $N(\mu_i, 1)$, and arbitrarily choosing $N=20$ and $n=5$ we use the following R code:

```
N = 20
n = 5

## Randomly Generate the N cluster means
## using N(0,1) distribution, and sorting
## them in ascending order.

N.draws = rnorm(N)
MUs = sort(N.draws)

## Generate the matrix Y, which is a
## single realization of the scenario
```

```

#- Step 1:  Generate a matrix Z (N rows by n columns)
#-          of iid N(0,1) realizations

Z = array(rnorm(N*n), c(N,n))

#- Step 2:  Form a matrix M (N rows by n columns)
#-          with row i having all entries equal to MUs[i]

M = array(MUs, c(N,n))

#- Step 3:  Now we have that Y = Z+M, will generate
#-          a single realization of matrix Y with row i
#-          being iid draws from N(MUs[i], 1)

Y = Z + M

## Y is now ready to be used

```

Now we are in a position to use this single realization of the matrix Y_{ij} of clustered data to estimate population means μ_1, \dots, μ_N . Let's accomplish this by computing sample means of the data in each of the clusters, then sort them in ascending order yielding ordered sample means $\hat{\mu}_{(1)}, \dots, \hat{\mu}_{(N)}$. In R we have:

```
sample.means = sort(apply(Y,1, mean))
```

We should compare these ordered sample means to the populations means that they're supposed to estimate, we can get a picture by looking at the list of estimates and the list of estimands side by side. In R we have:

```
## Put the two column vectors into an N by 2 matrix SP:
SP = cbind(sample.means, MUs)
## View the results (Sample Mean, Population Mean)
SP

```

In the homework you will be asked to use simulation to estimate the bias, variance, and MSE of each of the ordered sample means $\hat{\mu}_{(1)}, \dots, \hat{\mu}_{(N)}$. Recall that the formula for MSE is:

$$\text{MSE}(\hat{\mu}_{(i)}) = E[(\hat{\mu}_{(i)} - \mu_i)^2]$$

Thus in order to handle the E in the formula, we take the average of `reps` realizations of the squared difference between the ordered sample means and corresponding population means, where the population means are generated to be different for each replication. We

can have the MSEs for all N clusters live in a vector `MSE.vec`, which is the mean of `reps` realizations of the vector: `(sample.means - MUs) ^ 2`. In R we would have the following setup, for which you could fill in the details:

```
N = 20
n = 5
reps = 1e3

squared.diff = array(0,N)

for ( r in 1:reps)
{
  ## Generate the population means for this
  ## replication

  N.draws = rnorm(N)
  MUs = sort(N.draws)

  ## Generate a single realization of Y
  ....paste code from above

  ## Compute the ordered sample means of Y
  ....paste code from above

  ## Compute the Squared Difference Vector,
  ## and add it to the vectors computed before

  squared.diff = squared.diff + (sample.means - MUs)^2
}

## Divide the sum of the squared difference vector
## by the number of vectors in the sum (which is reps)

MSE.vec = squared.diff/reps
```

Maximum Likelihood Estimation

Given X_1, \dots, X_n iid with probability density (or mass) function $f(x; \theta)$ where θ are the parameters of the distribution. The log-likelihood function is given by:

$$L(\theta) = \sum_{i=1}^n \log f(X_i; \theta)$$

The maximum likelihood estimator $\hat{\theta}$ is the point that maximizes the function $L(\theta)$. We will now focus on techniques for computing the maximum likelihood estimator.

Note: θ could be more than one parameter, in the case of the normal distribution, $\theta = (\mu, \sigma^2)$, whereas in the case of the exponential distribution, $\theta = \lambda$. For the purposes of this course, you will only be maximizing $L(\theta)$ over one parameter, treating the other parameters as constants.

- We could find $\hat{\theta}$ by solving the equation $L'(\hat{\theta}) = 0$, if possible.
- We could find $\hat{\theta}$ by applying Newton's Method to the function $L(\theta)$ if possible.

Example: The Poisson Distribution with parameter $\theta > 0$ has probability mass function:

$$f(x; \theta) = \frac{\theta^x e^{-\theta}}{x!} \quad \text{for } x = 0, 1, \dots$$

Consider data X_1, \dots, X_n iid with probability mass function $f(x; \theta)$, we have the log-likelihood function:

$$L(\theta) = \sum_{i=1}^n \log f(X_i; \theta) = \sum_{i=1}^n \log \left(\frac{\theta^{X_i} e^{-\theta}}{X_i!} \right)$$

Recall that $\log(ab/c) = \log(a) + \log(b) - \log(c)$ and $\log(a^b) = b \log(a)$. Applying these identities we have:

$$L(\theta) = \sum_{i=1}^n (X_i \log \theta - \theta - \log(X_i!)) = \log \theta \sum_{i=1}^n X_i - n\theta - \sum_{i=1}^n \log(X_i!)$$

Now we can take the derivative with respect to θ :

$$L'(\theta) = \frac{1}{\theta} \sum_{i=1}^n X_i - n$$

Now solve the equation $L'(\hat{\theta}) = 0$

$$\frac{1}{\hat{\theta}} \sum_{i=1}^n X_i = n \quad \text{implies} \quad \hat{\theta} = \frac{1}{n} \sum_{i=1}^n X_i$$

Example: The Geometric Distribution with parameter $0 < \theta < 1$ has probability mass function:

$$f(x; \theta) = (1 - \theta)^{x-1} \theta \quad \text{for } x = 1, 2, \dots$$

Consider data X_1, \dots, X_n iid with probability mass function $f(x; \theta)$, we have the log-likelihood function:

$$L(\theta) = \sum_{i=1}^n \log f(X_i; \theta) = \sum_{i=1}^n \log ((1 - \theta)^{X_i-1} \theta)$$

After some simplification we have:

$$L(\theta) = \log(1 - \theta) \left(\sum_{i=1}^n X_i - n \right) + n \log \theta$$

We could find the $\hat{\theta}$ that maximizes $L(\theta)$ using Newton's method, which is accomplished by choosing a starting value θ_0 and then iterating with the equation:

$$\theta_{k+1} = \theta_k - \frac{L'(\theta_k)}{L''(\theta_k)} \quad \text{for } k = 0, 1, \dots$$

One should stop iterating when θ_k is not changing much with each iteration. I will choose to stop when:

$$|\theta_{k+1} - \theta_k| < 10^{-6}$$

We must find $L'(\theta)$ and $L''(\theta)$ in order to use Newton's method, which are given by:

$$L'(\theta) = n\theta^{-1} - \left(\sum_{i=1}^n X_i - n \right) (1 - \theta)^{-1}$$

$$L''(\theta) = -n\theta^{-2} - \left(\sum_{i=1}^n X_i - n \right) (1 - \theta)^{-2}$$

Lets generate a single realization of X_1, \dots, X_n iid Geometric($\theta = 0.25$), $n = 100$ and find the MLE using Newton's method in R, with the starting point $\theta_0 = 0.6$:

```
n=100
## Generate n realizations
X=rgeom(n,prob=0.25)

## Set the starting value
oldTheta = 0.6

## Let k count the number of iterations
k=0
distance = Inf
while( distance >= 1e-6 )
{
  k=k+1
  Lprime = n/oldTheta - (sum(X) - n)/(1-oldTheta)
  Lprimeprime = -n*oldTheta^(-2) - (sum(X)-n)*(1-oldTheta)^(-2)

  ## Newton's Step
  newTheta = oldTheta - Lprime/Lprimeprime
```

```
## Compute distance we just stepped
distance = abs(newTheta-oldTheta)

## Update oldTheta for the next iteration
oldTheta = newTheta

## Print our progress
cat("k=",k,"newTheta=",newTheta,"Distance=",distance,"\n")
}
MLE = newTheta
```

One could find the closed form solution for the MLE here, by solving $L'(\hat{\theta}) = 0$ for $\hat{\theta}$:

$$n\hat{\theta}^{-1} = \left(\sum_{i=1}^n X_i - n \right) (1 - \hat{\theta})^{-1}$$

Multiply both sides by $(1 - \hat{\theta})$ and divide both sides by n :

$$\hat{\theta} = \frac{1}{\bar{X}}$$