Homework 3 - STAT 540

Amal Agarwal

Answer 1

(a) (i) Psudocode for Nelder-Mead simplex algorithm:

- Initialization: Since the maximization of the log-likelihood function is over a three-dimensional space (μ, ξ, σ), we minimize the negative log-likelihood function and start with an initial simplex S containing 4 vertices (x_0, x_1, x_2, x_4) such that $x_i \in \mathbb{R}^3$.
- Iterations: Repeat the following steps till termination condition is satisfied.
  - Ordering: In the current working simplex S, determine the indices h, s, l of the worst, second worst and the best vertex, such that

$$f_h = \max_j f_j, f_s = \max_{j \neq h} f_j, f_l = \min_{j \neq h} f_j$$

  - Calculate the centroid c of the best side which is the one opposite to the worst vertex as $c := \frac{1}{n} \sum_{j \neq h} x_j$
  - Transformation: Compute the new working simplex from the current one. First, replace only the worst vertex x_h with a better point by using reflection, expansion or contraction with respect to the best side. All test points lie on the line defined by x_h and c , and at most two of them are computed in one iteration. If this succeeds, the accepted point becomes the new vertex of the working simplex. If this fails, shrink the simplex towards the best vertex x_l . In this case, n new vertices are computed. Simplex transformations in the Nelder-Mead method are controlled by four parameters satisfying the following constraints: $\alpha > 0, 0 < \beta < 1, \gamma > 1, \gamma > \alpha, 0 < \delta < 1$.
    - Reflect: Compute the reflection point $x_r := c + \alpha(c - x_h)$ and $f_r := f(x_r)$. If $f_l \leq f_r < f_s$, accept x_r and terminate the iteration.
    - Expand: If $f_r < f_l$, compute the expansion point $x_e := c + \gamma(x_r c)$ and $f_e := f(x_e)$. If $f_e < f_r$, accept x_e and terminate the iteration. Otherwise (if $f_e \geq f_r$), accept x_r and terminate the iteration.
    - Contract: If $f_r \geq f_s$, compute the contraction point x_c by using the better of the two points x_h and x_r.
      - Outside: If $f_s \leq f_r < f_h$, compute $x_c := c + \beta(x_r - c)$ and $f_c := f(x_c)$. If $f_c \leq f_r$, accept x_c and terminate the iteration. Otherwise, perform a shrink transformation.
      - Inside: If $f_r \geq f_h$, compute $x_c := c + \beta(x_h - c)$ and $f_c := f(x_c)$. If $f_c < f_h$, accept x_c and terminate the iteration.

      Otherwise, perform a shrink transformation.

* Shrink: Compute n new vertices $x_j := x_l + \delta(x_j - x_l)$ and $f_j := f(x_j)$, for $j = 0, ..., n$, with $j \neq l$.

- Termination: When the working simplex S is sufficiently small in some sense i.e. some or all vertices x_j are close enough, terminate the algorithm.

(Reference: Scholarpedia)

(ii) Psudocode for Newton- Raphson algorithm:

- Start with an initial value $x^{(0)} = (\mu^{(0)}, \xi^{(0)}, \sigma^{(0)})$
- Run loop until the termination condition is satisfied.
  - $x^{(t+1)} = x^{(t)} - \nabla^2 l(x^{(t)}) \times \nabla l(x^{(t)})$ where $\nabla l(x^{(t)})$ is the gradient of log likelihood function evaluated at the the iterate $x^{(t)} = (\mu^{(t)}, \xi^{(t)}, \sigma^{(t)})$ and $\nabla^2 l(x^{(t)})$ is the hessian of log likelihood function evaluated at the the iterate $x^{(t)} = (\mu^{(t)}, \xi^{(t)}, \sigma^{(t)})$.
  - Check if $|x^{(t+1)} - x^{(t)}| < tol$ to terminate the loop. Here tol is some pre-specified tolerance.

(iii) Choosing the initial values: For both the above algorithms along with BFGS, the built-in function optim in R was used to calculate the MLE of (μ, ζ, σ) for the given data. In optim the initial values were passed as the maximizers of the log likelihood function over a grid of 1000 points in the 3-dimensional parameter space. The end-points of the grid were randomly chosen as $(1, 5) \times (0.1, 5) \times (0.05, 5.5)$.

(b) The following table gives the required information:

Table 1: ML estimates and standard errors for the three algorithms

	ML estimates for (μ, ξ, σ)	Standard errors for (μ, ξ, σ)
BFGS	(1.65,0.18,0.098)	(0.0021,0.0205,0.0017)
Nelder-Mead	(1.65,0.18,0.098)	(0.0021,0.0205,0.0017)
Newton Raphson	(1.65,0.18,0.098)	(0.0021,0.0205,0.0017)

Standard error estimates of (μ, ξ, σ) were found by inverting the observed information matrix and taking the square root of the diagnol entries which correspond to (μ, ξ, σ) respectively. This is using the asymptotic theory for MLE's which gives the following convergence in law.

$$\sqrt{n}(\hat{\theta_n} - \theta) \longrightarrow N(0, I^{-1}(\theta))$$

Now we know that the observed information is the negative of the Hessian matrix. For the first two algorithms BFGS and Nelder-Mead, the negative Hessian matrix can be extracted from the optim function. For the Newton-Raphson algorithm, the Hessian matrix was computed for the last iterate using "hessian" function from the package "pracma". Note that in all the above cases the observed information is for the whole dataset (not just one data point) and so division by n is not required.

(c) The following figure gives the fitted GEV density for all the three optimization algorithms over histogram of the data.

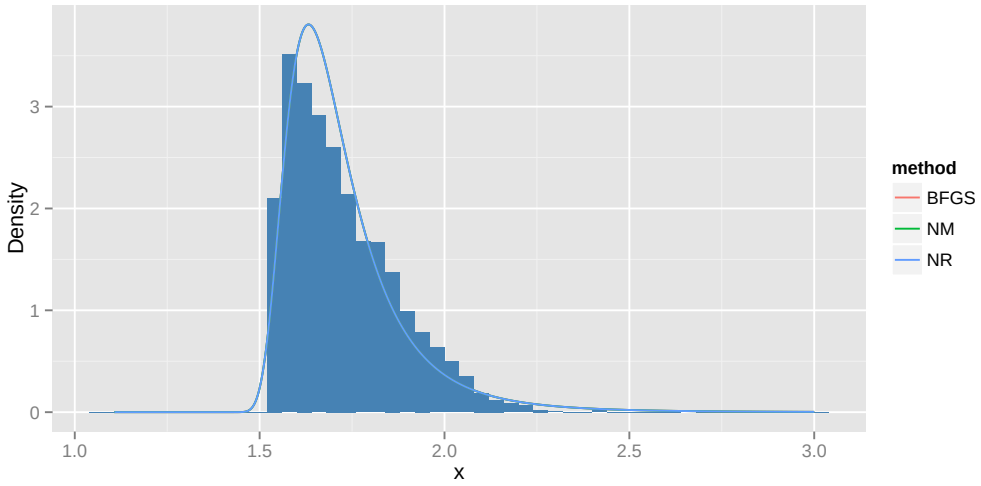

Figure 1: Plot of fitted density for three algorithms over the histogram of the data

Clearly the lines seem to coincide which shows that the parameter estimates are really close.

Answer 2

(a) (i) Pseudocode for the first Metropolis Hastings algorithm (Variable one at a time) for the given target density:

- Start off with $x^0 = (\mu^{(0)}, \xi^{(0)}, \sigma^{(0)})$ chosen suitably after a series of pilot runs. Choose the length of the Markov chain as m.
- Run loop from $t = 0 : (m-1)$
  - Assign the current state as $x^{(t)} = (\mu^{(t)}, \xi^{(t)}, \sigma^{(t)})$.
  - Generate a candidate μ^* drawn from univariate normal distribution with mean as $\mu^{(t)}$ and variance τ_μ^2 being the tuning parameter.
  - Accept μ^* as next state $\mu^{(t+1)}$ with the following acceptance probability α_μ, else assign the current state as next state.

$$\alpha_\mu(\mu^*, \mu^{(t)}) = \min\left(1, \frac{h_\mu(\mu^*, \xi^{(t)}, \sigma^{(t)}) q_\mu(\mu^*, \mu^{(t)})}{h_\mu(\mu^{(t)}, \xi^{(t)}, \sigma^{(t)}) q_\mu(\mu^{(t)}, \mu^*)}\right) = \min\left(1, \frac{h_\mu(\mu^*, \xi^{(t)}, \sigma^{(t)})}{h_\mu(\mu^{(t)}, \xi^{(t)}, \sigma^{(t)})}\right)$$

where h_μ is the conditional target kernel for μ which the product of the likelihood function and the uniform prior (contains no term for the given prior) and q_μ is univariate normal proposal density which cancels out being symmetric.

- Generate a candidate ξ^* drawn from univariate normal distribution with mean as $\xi^{(t)}$ and variance τ_ξ^2 being the tuning parameter.
- Accept ξ^* as next state $\xi^{(t+1)}$ with the following acceptance probability α_ξ, else assign the current state as next state.

$$\alpha_\xi(\xi^*, \xi^{(t)}) = \min\left(1, \frac{h_\xi(\mu^{(t+1)}, \xi^*, \sigma^{(t)}) q_\xi(\xi^*, \xi^{(t)})}{h_\xi(\mu^{(t+1)}, \xi^{(t)}, \sigma^{(t)}) q_\xi(\xi^{(t)}, \xi^*)}\right) = \min\left(1, \frac{h_\xi(\mu^{(t+1)}, \xi^*, \sigma^{(t)})}{h_\xi(\mu^{(t+1)}, \xi^{(t)}, \sigma^{(t)})}\right)$$

  where h_ξ is the conditional target kernel for ξ which is the product of the likelihood function and the uniform prior (contains no term for the given prior) and q_ξ is univariate normal proposal density which cancels out being symmetric.
- Generate a candidate σ^* drawn from univariate truncated normal distribution with bounds as $(0, \infty)$. Again take the mean of this proposal as $\sigma^{(t)}$ and variance as τ_σ^2 being the tuning parameter.
- Accept σ^* as next state $\sigma^{(t+1)}$ with the following acceptance probability α_σ, else assign the current state as next state.

$$\alpha_\sigma(\sigma^*, \sigma^{(t)}) = \min\left(1, \frac{h_\sigma(\mu^{(t+1)}, \xi^{(t+1)}, \sigma^*) q_\sigma(\sigma^*, \sigma^{(t)})}{h_\sigma(\mu^{(t+1)}, \xi^{(t+1)}, \sigma^{(t)}) q_\sigma(\sigma^t, \sigma^*)}\right)$$

  where h_σ is the conditional target kernel for σ which is the product of the likelihood function and the uniform prior and q_σ is univariate truncated normal proposal density.
- Assign the next state as $x^{(t+1)} = (\mu^{(t+1)}, \xi^{(t+1)}, \sigma^{(t+1)})$.

(ii) Pseudocode for the second Metropolis Hastings algorithm (Variable one at a time) for the given target density is same as the first one above with just the change in proposal density q_σ for σ as gamma density instead of univariate truncated normal. The mean of q_σ is chosen as the current state and the variance is kept as the tuning parameter.

(iii) As before, starting values for both the algorithms were chosen as the maximizers of the log likelihood function over a grid of 1000 points in the 3-dimensional parameter space. The end-points of the grid were randomly chosen as $(1, 5) \times (0.1, 5) \times (0.05, 5.5)$.

(b) For both the algorithms, the appropriate chain length was chosen as 2500 since the all the diagnostics shown below suggest that the convergence takes place till after this point. Besides, the relative error i.e. the ratio of MCMC standard error to the estimate drops below 10^{-2} for $n > 2500$ which is reasonably good.

(i) Diagnostics for algorithm 1: Plot of the ACF of samples is given as:

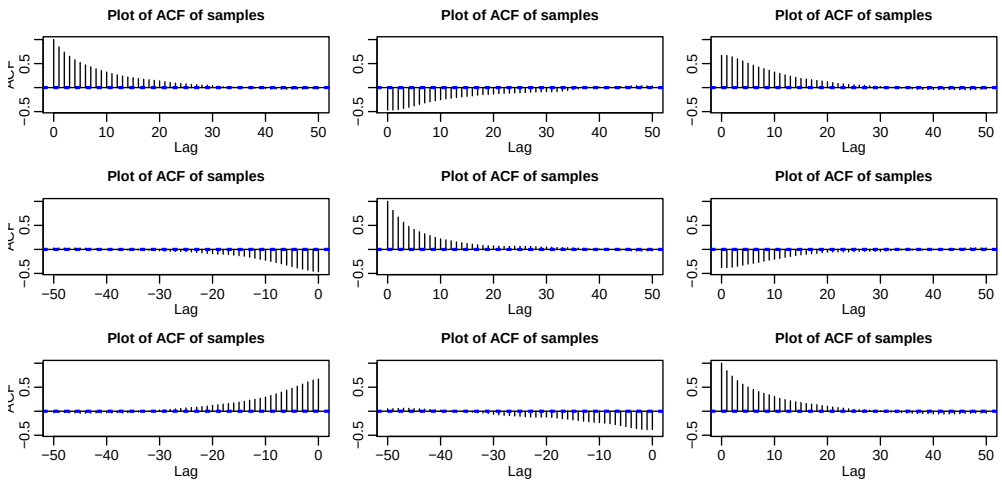

Figure 2: Plot of ACF of samples

For all the 3 components, the above plot is reasonably good which indicates that the tuning parameter value of $(4.36 \times 10^{-6}, 4.20 \times 10^{-4}, 2.88 \times 10^{-6})$ may not be the best but still it works reasonably good.

For 3 different starting values as (1.65,0.17,0.098), (1.64,0.18,0.091) and (1.65,0.17,0.127), labelled as 1, 2 and 3, plots of estimates of expected values of (μ, ξ, σ) and Monte Carlo standard errors with sample size are given below:

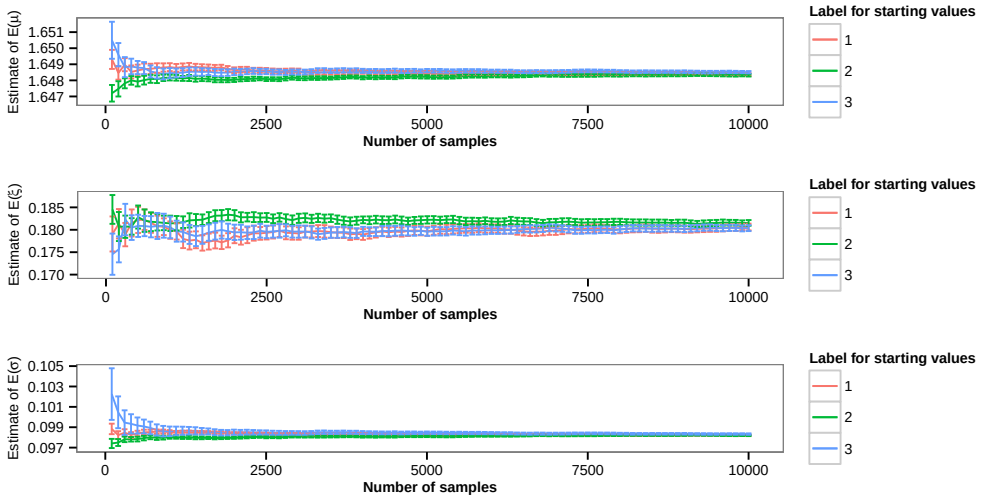

Figure 3: Plot of estimates vs. sample size with error bars for (μ, ξ, σ)

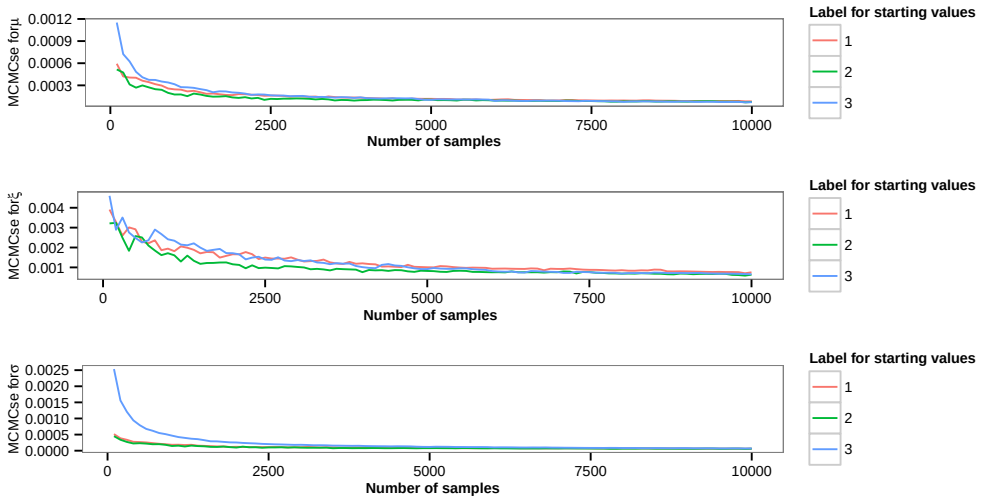

Figure 4: Plot of MCMC se vs. sample size for (μ, ξ, σ)

The above plots show that for all the 3 components and for different starting alues, the estimates converge to same value and MCMC standard errors converge to 0. This verifies the algorithm to some extent. Plotting the estimated density after $n/2$ and after n:

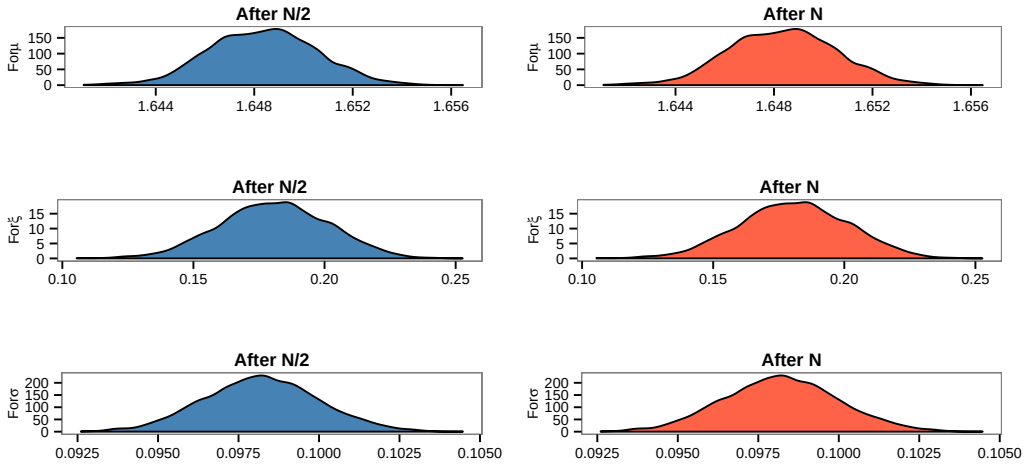

Figure 5: Plot of estimated density for (μ, ξ, σ) after N/2 and N

Plotting the estimated density for different starting values for all the components:

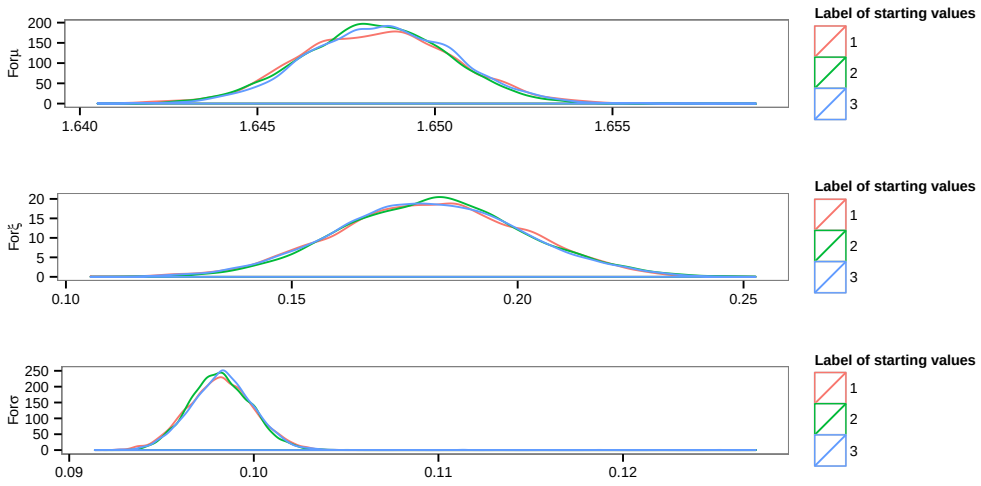

Figure 6: Plot of marginal density for (μ, ξ, σ)

From the above plots, the estimated densities after $n/2$ and after n look reasonably identical for all the 3 components. Also the estimated density for different starting values also overlap to a good extent. This further verifies the robustness and convergence of the algorithm.

(ii) Diagnostics for algorithm 2: Plot of the ACF of samples is given as:

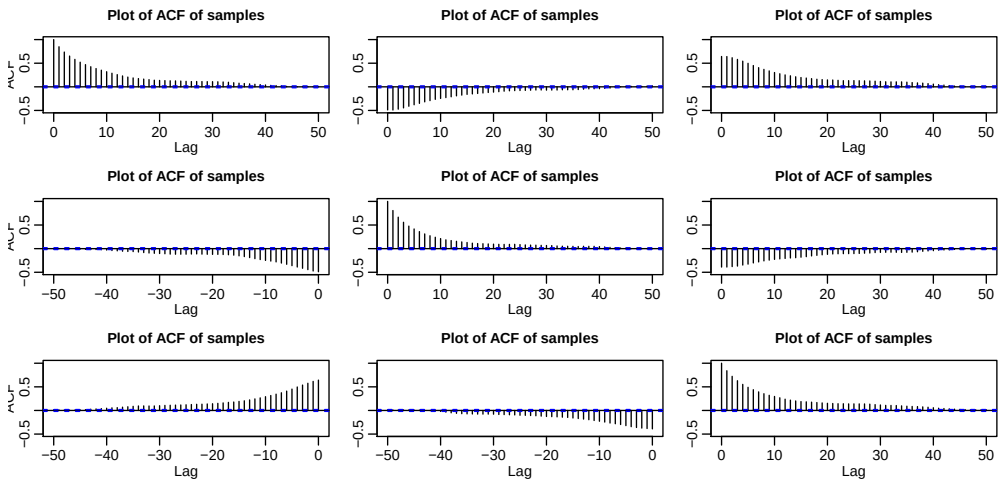

Figure 7: Plot of ACF of samples

For all the 3 components, the above plot is reasonably good which indicates that the tuning parameter values of $(4.46 \times 10^{-6}, 4.33 \times 10^{-4}, 2.94 \times 10^{-6})$ may not be the best but still it works reasonably good.

For 3 different starting values as (1.65,0.17,0.098), (1.62,0.16,0.102) and (1.68,0.2,0.093), labelled as 1, 2 and 3, plots of estimates of expected values of (μ, ξ, σ) and Monte Carlo standard errors with sample size are given below:

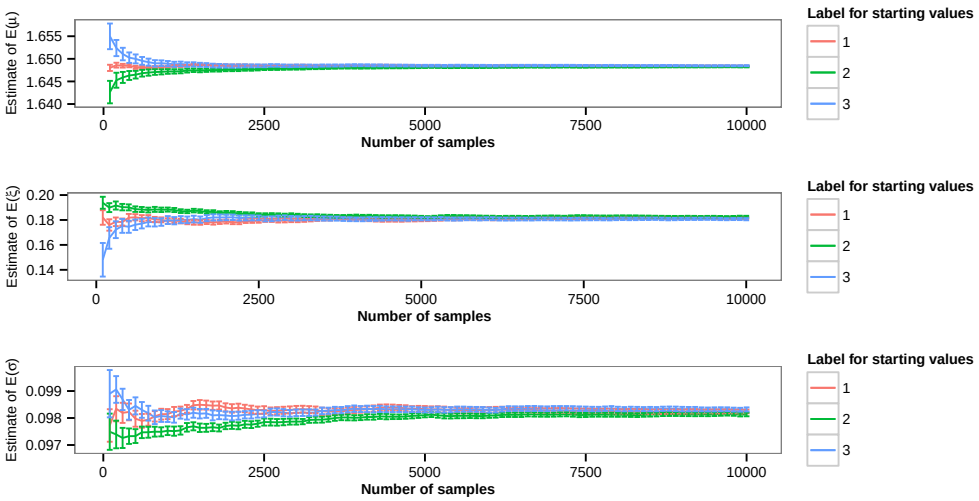

Figure 8: Plot of estimates vs. sample size with error bars for (μ, ξ, σ)

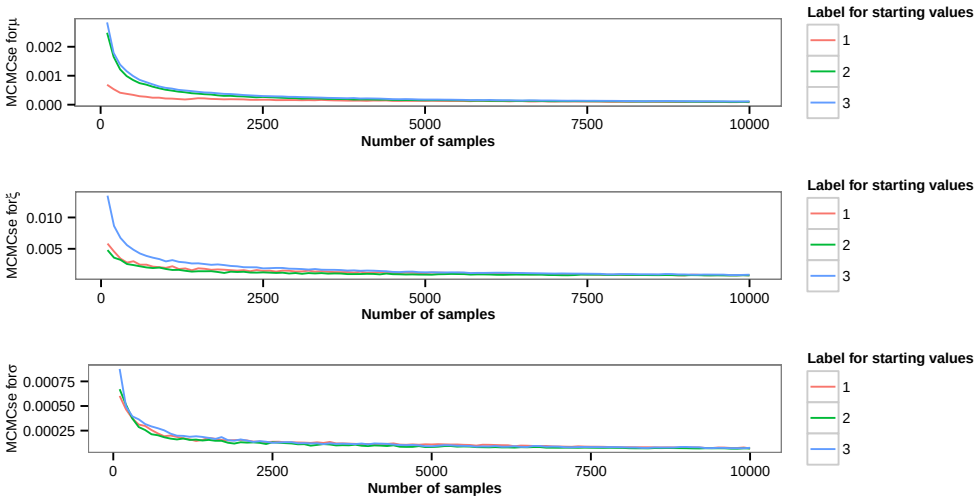

Figure 9: Plot of MCMC se vs. sample size for (μ, ξ, σ)

The above plots show that for all the 3 components and for different starting alues, the estimates converge to same value and MCMC standard errors converge to 0. This verifies the algorithm to some extent.

Plotting the estimated density after $n/2$ and after n:

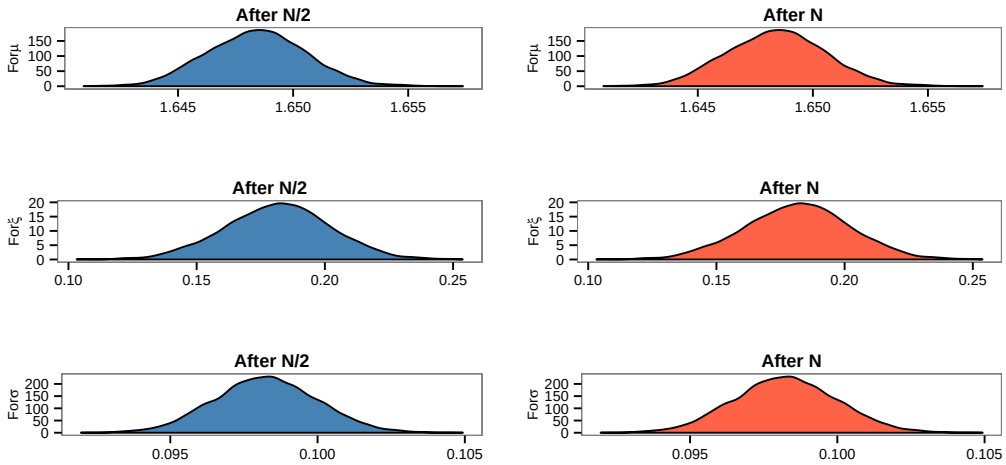

Figure 10: Plot of estimated density for (μ, ξ, σ) after N/2 and N

Plotting the estimated density for different starting values for all the components:

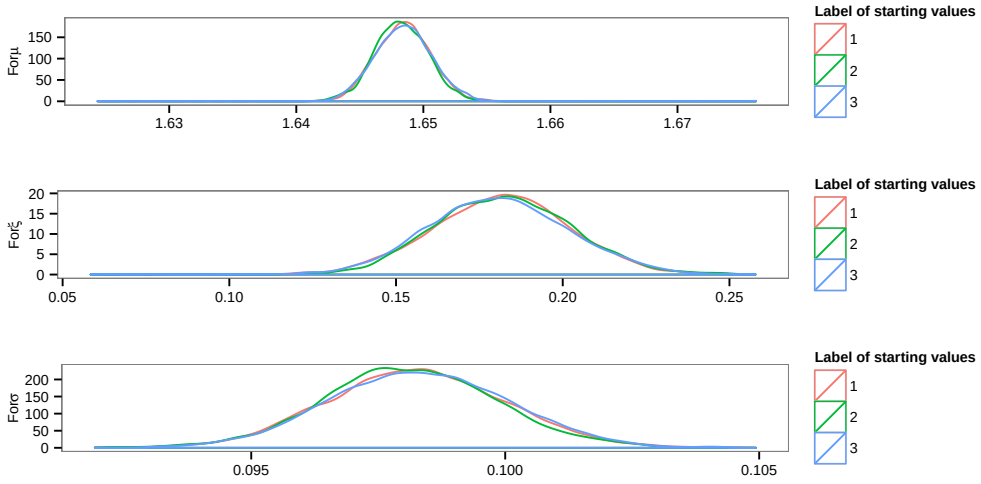

Figure 11: Plot of marginal density for (μ, ξ, σ)

From the above plots, the estimated densities after $n/2$ and after n look reasonably identical for all the 3 components. Also the estimated density for different starting values also overlap to a good extent. This further verifies the robustness and convergence of the algorithm.

(c) The following table gives the required information:

Table 2: Posterior estimates, standard deviations and MCMC standard errors

	Posterior estimates for (μ, ξ, σ)	MCMC standard errors	Posterior standard deviations
Algorithm 1	(1.65, 0.18, 0.098)	(4e-04, 0.0026, 3e-04)	(0.0022, 0.0208, 0.0018)
Algorithm 2	(1.65, 0.18, 0.098)	(4e-04, 0.0034, 4e-04)	(0.0022, 0.0208, 0.0018)

(d) The following table gives the required information:

Table 3: Comparing confidence and credible intervals for different algorithms

	Conf. int. for μ	Conf. int. for ξ	Conf. int. for σ
BFGS	(1.6442, 1.6525)	(0.1407, 0.2212)	(0.0948, 0.1015)
Nelder-Mead	(1.6442, 1.6525)	(0.1406, 0.221)	(0.0948, 0.1015)
Newton Raphson	(1.6442, 1.6524)	(0.1408, 0.2212)	(0.0948, 0.1015)
	Cred. int. for μ	Cred. int. for ξ	Cred. int. for σ
Algorithm 1	(1.6441, 1.6526)	(0.1396, 0.2204)	(0.0948, 0.1018)
Algorithm 2	(1.6443, 1.6527)	(0.1403, 0.2216)	(0.0948, 0.1017)

Answer 3

(a) Among the three optimization algorithms, viz. BFGS, Nelder-Mead and Newton-Raphson, we observed earlier that the Newton Raphson algorithm does not converge if the initial values of the parameters (μ, ξ, σ) are not too close to the MLE's. Also, there is not any apparent difference between BFGS and Nelder-Mead algorithm since the standard errors are exactly same. Therefore best optimization algorithm to compute the MLE's was chosen randomly as Nelder-Mead. With the given parameters 1000 data points were simulated for one set. The number of sets determine the Monte Carlo sample size.

For the three parameters (μ, ξ, σ), the estimates of mean square errors were obtained as (0.000945, 0.000883, 0.000684). The corresponding Monte-Carlo standard errors for these approximations were (0.000111, 9e-05, 6.2e-05).

(b) The coverage probabilties for 95% confidence intervals for (μ, ξ, σ) were obtained as (0.9536, 0.9588, 0.9485). The corresponding Monte-Carlo standard errors for these approximations were (0.0151, 0.0143, 0.0159).

(c) The simulation study was carried out by generatng datasets one by one and calculating sample mean estimates and standard for all the three parameters using Nelder Mead algorithm. We define the relative error for each parameter as the ratio of Monte Carlo standard error to the point estimate (sample mean) upto a particular Monte Carlo sample size. The generation of datasets was stopped when the maximum of the relative error for all parameters becomes less than an arbitrary tolerance of 5×10^{-3}. The number of

datasets chosen gives the Monte Carlo sample size of 194 for the chosen tolerance. Also at each iterate the log likelihood function was evaluated on a grid of 125 points in the parameter space. For the first iterate, the grid end-points were initialized arbirarily at $(-10, 10) \times (-10, 10) \times (0.1, 10.1)$. For the successive iterates, the grid end-points were fixed as the 95% confidence bounds obtained in the previous iterate. Note that this approach to choose initial values exploits the information from successive simulated datasets, thereby making the estimates and standard errors more and more accurate as the number of iterations increase.