Assignment 2 Question 2

Sheen Thusoo

Part f)

```r
data <- read.csv("EconomicMobility.csv")
```

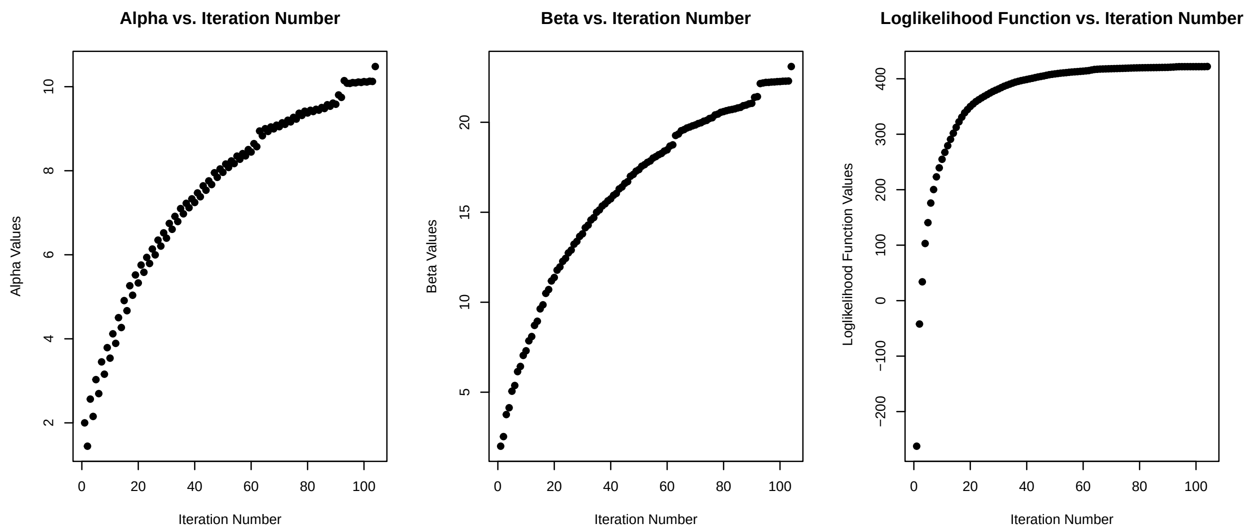
```r
gradientAscentWithSolutionPath <- function(theta,
                                           rhoFn,
                                           gradientFn,
                                           lineSearchFn,
                                           testConvergenceFn,
                                           maxIterations = 100,
                                           tolerance = 1E-6,
                                           relative = FALSE,
                                           lambdaStepsize = 0.01,
                                           lambdaMax = 0.5) {
  # Pre-allocate a matrix to track theta at each iteration.
  SolutionPath = matrix(NA,
                        nrow = maxIterations + 1,
                        ncol = length(theta))
  rhoPath = array(NA, dim = maxIterations + 1)
  SolutionPath[1, ] = theta
  rhoPath[1] = rhoFn(theta)

  for (i in 1:maxIterations) {
    g         <- gradientFn(theta) # Unnormalized gradient.
    glength   <- sqrt(sum(g ^ 2))  # Gradient vector length.
    g         <- g / glength       # Unit vector gradient.
    lambda    <- lineSearchFn(theta, rhoFn, g,
                          lambdaStepsize = lambdaStepsize,
                          lambdaMax = lambdaMax)
    thetaNew  <- theta + lambda * g
    converged <- testConvergenceFn(thetaNew, theta,
                                   tolerance = tolerance,
                                   relative = relative)

    theta = thetaNew #Reza added this update
    SolutionPath[i + 1,] = theta
    rhoPath[i + 1] = rhoFn(theta)
    if (converged) break
  }

  ## Return information about the gradient descent procedure.
  return(list(theta = theta, converged = converged,
```

```r
              iteration = i, fnValue = rhoFn(theta),
              SolutionPath = SolutionPath[1:i, ], rhoPath = rhoPath[1:i]))
}

gridLineSearch <- function(theta,
                           rhoFn,
                           g,
                           lambdaStepsize = 0.01,
                           lambdaMax = 1) {
  ## Define equally-spaced grid of lambdas to search over.
  lambdas <- seq(from = 0, by = lambdaStepsize, to = lambdaMax)
  ## Evaluate the objective rho at each such lambda.
  rhoVals <- Map(function(lambda) {rhoFn(theta + lambda * g)}, lambdas)
  ## Return the lambda that gave the minimum objective.
  return(lambdas[which.max(rhoVals)])
}

testConvergence <- function(thetaNew,
                            thetaOld,
                            tolerance = 1E-10,
                            relative = FALSE) {
   sum(abs(thetaNew - thetaOld)) <
    if (relative) tolerance * sum(abs(thetaOld)) else tolerance
}
```

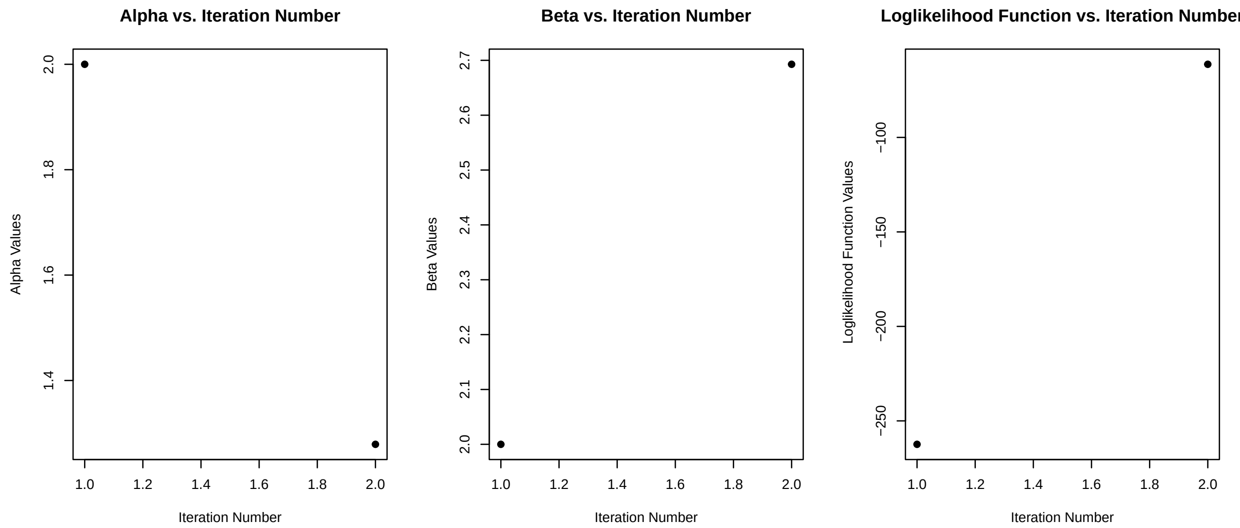
```r
rho <- function(x) {
  alpha = x[1]
  beta = x[2]
  loglikelihood <- 0
  P <- data$Commute
  for (y in P) {
    loglikelihood <- loglikelihood + alpha*log(beta) +
      (alpha -1)*log(y) - log(gamma(alpha)) - y*beta
  }
  return(loglikelihood)
}

g <- function(x) {
  alpha = x[1]
  beta = x[2]
  y <- data$Commute
  grad1 <- sum(log(beta) + log(y) - digamma(alpha))
  grad2 <- sum((alpha/beta) - y)
  return(c(grad1 , grad2))
}
```

```r
# Starting at (2, 2) as mentioned in Piazza
Optim1 = gradientAscentWithSolutionPath(rhoFn = rho, gradientFn = g, theta = c(2,2),
           lineSearchFn = gridLineSearch, testConvergenceFn = testConvergence,
           maxIterations = 1000, lambdaMax = 5)

par(mfrow = c(1, 3))
plot(Optim1$SolutionPath[,1], pch = 19, main='Alpha vs. Iteration Number',
```

```
     xlab='Iteration Number', ylab='Alpha Values')
plot(Optim1$SolutionPath[,2], pch = 19, main='Beta vs. Iteration Number',
     xlab='Iteration Number', ylab='Beta Values')
plot(Optim1$rhoPath, pch = 19, main='Loglikelihood Function vs. Iteration Number',
     xlab='Iteration Number', ylab=' Loglikelihood Function Values')
```




```
# Starting at (2, 2) as mentioned on Piazza
# lambda=0,1,2,3,4,5
Optim2 = gradientAscentWithSolutionPath(rhoFn = rho, gradientFn = g, theta = c(2,2),
           lineSearchFn = gridLineSearch, testConvergenceFn = testConvergence,
           maxIterations = 1000, lambdaStepsize = 1, lambdaMax = 5)

par(mfrow = c(1, 3))
plot(Optim2$SolutionPath[,1], pch = 19, main='Alpha vs. Iteration Number',
     xlab='Iteration Number', ylab='Alpha Values')
plot(Optim2$SolutionPath[,2], pch = 19, main='Beta vs. Iteration Number',
     xlab='Iteration Number', ylab='Beta Values')
plot(Optim2$rhoPath, pch = 19, main='Loglikelihood Function vs. Iteration Number',
     xlab='Iteration Number', ylab=' Loglikelihood Function Values')
```




```r
# Starting at (2, 2) as mentioned on Piazza
# lambda=0.0001,0.0002,...,0.01
Optim3 = gradientAscentWithSolutionPath(rhoFn = rho, gradientFn = g, theta = c(2,2),
           lineSearchFn = gridLineSearch, testConvergenceFn = testConvergence,
           maxIterations = 1000, lambdaStepsize = 0.0001, lambdaMax = 0.01)

par(mfrow = c(1, 3))
plot(Optim3$SolutionPath[,1], pch = 19, main='Alpha vs. Iteration Number',
     xlab='Iteration Number', ylab='Alpha Values')
plot(Optim3$SolutionPath[,2], pch = 19, main='Beta vs. Iteration Number',
     xlab='Iteration Number', ylab='Beta Values')
plot(Optim3$rhoPath, pch = 19, main='Loglikelihood Function vs. Iteration Number',
     xlab='Iteration Number', ylab=' Loglikelihood Function Values')
```