Extensive Assignment 2

Eric Hare, Samantha Tyner

March 31, 2014

1 Exploratory Analysis

2 Newton-Raphson Algorithm

In the spirit of conducting as much of this analysis "by-hand" as possible, we elected to write our own Newton-Raphson function based on the STAT 520 course notes. This function accepts four required parameters: a function which returns the value of the log likelihood for the parameters which we are interested in maximizing, functions that return the values of the gradient and hessian, and starting values for the parameters that these functions accept.

Other parameters include:

- lower - Lower bounds of the parameter space
- upper - Upper bounds of the parameter space
- tol - The three tolerance values for which we end the algorithm upon reaching
- max.iterations - The maximum number of iterations of the algorithm before terminating and returning
- step.halving - A boolean indicating whether to perform the step-halving procedure
- debug - A boolean indicating whether to print debug messages

This function will return the Maximum Likelihood Estimates of the parameters given in the loglik, gradient, and hessian functions. The full code is reproduced starting on page 4.

```
newton.raphson <- function(loglik, gradient, hessian, start, lower = rep(-Inf,
    length(start)), upper = rep(Inf, length(start)), tol = rep(0.01, 3), max.iterations = 100,
    step.halving = TRUE, debug = FALSE, ...) {
    current <- start
    conditions <- TRUE

    iterations <- 0
    while (TRUE) {
        new <- as.vector(current - solve(hessian(current, ...)) %*% gradient(current,
            ...))
        new[new < lower] <- lower[new < lower] + tol[1]
        new[new > upper] <- upper[new > upper] - tol[1]

        if (!(any(abs(gradient(new, ...)) > tol[1]) | loglik(new, ...) - loglik(current,
            ...) > tol[2] | dist(rbind(current, new))[1] > tol[3]))
            break

        if (debug)
```

```
            cat(paste("Current loglik is", loglik(current, ...), "\n"))
        if (debug)
            cat(paste("New is now", new, "\n"))
        if (debug)
            cat(paste("New loglik is", loglik(new, ...), "\n"))

        if (step.halving & (loglik(new, ...) < loglik(current, ...))) {
            if (debug)
                cat("Uh oh, time to fix this\n")
            m <- 1
            while (m < max.iterations & loglik(new, ...) < loglik(current, ...)) {
                new <- as.vector(current - (1/(2 * m)) * solve(hessian(current,
                  ...)) %*% gradient(current, ...))
                m <- 2 * m
            }
            if (debug)
                cat(paste("We have fixed it! its now", new, "\n"))
            if (debug)
                cat(paste("And the new loglik is finally", loglik(new, ...),
                  "\n"))
        }

        iterations <- iterations + 1
        if (iterations > max.iterations) {
            if (debug)
                cat(paste("Didn't converge in", max.iterations, "iterations\n"))
            break
        }

        if (debug)
            cat("\n")

        current <- new
    }

    return(list(loglik = loglik(new, ...), par = new))
}
```

We will use this function to determine the MLEs for the three models presented henceforth.

3 Model Formulation

4 Code Appendix

```
library(pscl)
library(plyr)
library(ggplot2)
library(reshape2)
library(xtable)
library(dplyr)
newton.raphson <- function(loglik, gradient, hessian, start, lower = rep(-Inf,
    length(start)), upper = rep(Inf, length(start)), tol = rep(0.01, 3), max.iterations = 100,
    step.halving = TRUE, debug = FALSE, ...) {
    current <- start
    conditions <- TRUE

    iterations <- 0
    while (TRUE) {
        new <- as.vector(current - solve(hessian(current, ...)) %*% gradient(current,
            ...))
        new[new < lower] <- lower[new < lower] + tol[1]
        new[new > upper] <- upper[new > upper] - tol[1]

        if (!(any(abs(gradient(new, ...)) > tol[1]) | loglik(new, ...) - loglik(current,
            ...) > tol[2] | dist(rbind(current, new))[1] > tol[3]))
            break

        if (debug)
            cat(paste("Current loglik is", loglik(current, ...), "\n"))
        if (debug)
            cat(paste("New is now", new, "\n"))
        if (debug)
            cat(paste("New loglik is", loglik(new, ...), "\n"))

        if (step.halving & (loglik(new, ...) < loglik(current, ...))) {
            if (debug)
                cat("Uh oh, time to fix this\n")
            m <- 1
            while (m < max.iterations & loglik(new, ...) < loglik(current, ...)) {
                new <- as.vector(current - (1/(2 * m)) * solve(hessian(current,
                  ...)) %*% gradient(current, ...))
                m <- 2 * m
            }
            if (debug)
                cat(paste("We have fixed it! its now", new, "\n"))
            if (debug)
                cat(paste("And the new loglik is finally", loglik(new, ...),
                  "\n"))
        }
```

```
        iterations <- iterations + 1
        if (iterations > max.iterations) {
            if (debug)
                cat(paste("Didn't converge in", max.iterations, "iterations\n"))
            break
        }

        if (debug)
            cat("\n")

        current <- new
    }

    return(list(loglik = loglik(new, ...), par = new))
}
```