```
install.packages("nleqslv")

library("nleqslv")

# Interval cutpoints defined by Holness and frequencies in each interval
#  as observed in Holness
Holness.x <- c(4.4, 8.8, 17.7)
Holness.freq <- c(34, 12, 9, 3)

HSolve.fn <- function (v, lims, freqs)
{
  # Defines dLL, the derivative of the log-likelihood function for the multinomial

  # 'v': parameters (intercept, slope) to be solved for
  # 'lims': vector of interval limits (not including 0 and Inf), defined in console
  # 'freqs': vector of frequencies for intervals
  dLL = numeric(2)
  lline.low = v[1] + v[2] * log(c(0, lims))
  lline.upp = v[1] + v[2] * log(c(lims, Inf))
  logx.low = c(0, log(lims))
  logx.upp = c(log(lims), 0)
  pn.diff = pnorm(lline.upp) - pnorm(lline.low)
  dLL[1] = sum(freqs * (dnorm(lline.upp) - dnorm(lline.low)) / pn.diff)
  dLL[2] = sum(freqs * (logx.upp * dnorm(lline.upp) - logx.low * dnorm(lline.low)) / pn.diff)
  dLL
}

HSolve.jac <- function (v, lims, freqs)
{
  # Defines DdLL, the jacobian of dLL from HSolve.fn.
  # DdLL is the Hessian matrix of the multinomial.

  # 'v': parameters (intercept, slope) to be solved for
  # 'lims': vector of interval limits (not including 0 and Inf), defined in console
  # 'freq': vector of frequencies for intervals, defined in console
  n = length(v)
  DdLL = matrix(numeric(n*n), n, n)
  lline.low = v[1] + v[2] * log(c(0, lims))
  lline.lowm = v[1] + v[2] * c(0, log(lims))
  lline.upp = v[1] + v[2] * log(c(lims, Inf))
  lline.uppm = v[1] + v[2] * c(log(lims), 0)
  logx.low = c(0, log(lims))
  logx.upp = c(log(lims), 0)
  pn.diff = pnorm(lline.upp) - pnorm(lline.low)
  summ.num11 = (-lline.uppm * dnorm(lline.upp) + lline.lowm * dnorm(lline.low)) * pn.diff
  - (dnorm(lline.upp) - dnorm(lline.low))^2
  summ.num12 = (-logx.upp * lline.uppm * dnorm(lline.upp) + logx.low * lline.lowm * dnorm(lline.low)) *
pn.diff
```

```
  - (logx.upp * dnorm(lline.upp) - logx.low * dnorm(lline.low))^2
  summ.num22 = (-logx.upp^2 * lline.uppm * dnorm(lline.upp) + logx.low^2 * lline.lowm *
dnorm(lline.low)) * pn.diff
  - (logx.upp^2 * dnorm(lline.upp) - logx.low^2 * dnorm(lline.low))^2
  DdLL[1,1] = sum(freqs * summ.num11 / pn.diff^2)
  DdLL[1,2] = sum(freqs * summ.num12 / pn.diff^2)
  DdLL[2,1] = DdLL[1,2]
  DdLL[2,2] = sum(freqs * summ.num22 / pn.diff^2)
  DdLL
}

HSolve.vst.derv <- function (x, freq, k)
{
  # Construct matrix of starting values to input into 'nleqslv'
  kp1 = k + 1

  # Calculate minimum and maximum possible means and SDs given
  #  frequency distribution defined by 'freq'
  mn.min = mean(rep(c(0, x), freq))
  mn.max = mean(rep(c(x, 2*x[3]), freq))
  sd.min = sd(rep(x, c(sum(freq[1:2]), freq[3:4])))
  sd.max = sd(rep(c(0, x[2:3], 2*x[3]), freq))

  # For mean and SD, construct vector of starting values by dividing
  #  interval of possible values into k subintervals
  # Expand into matrix containing all possible pairs (mean, SD)
  mn.seq = seq(mn.min, mn.max, (mn.max - mn.min) / k)
  sd.seq = seq(sd.min, sd.max, (sd.max - sd.min) / k)
  parms = expand.grid(mn.seq / sd.seq, rep(1, kp1) / sd.seq)
  return(parms)
}

# Construct matrix of starting values to input into 'nleqslv' for frequency
#  data from Holness
Holness.vst <- HSolve.vst.derv(Holness.x, Holness.freq, 5)

# Execute 'nleqslv' for each vector of starting values in 'Holness.vst'
Holness.slns <- apply(Holness.vst, 1, nleqslv, fn = HSolve.fn, jac = HSolve.jac, lims = Holness.x, freqs =
Holness.freq)

# Save parameter results and convergence status from Holness.slns, and
#  keep only those for which convergence was obtained
Holness.parms <- t(rbind(sapply(Holness.slns, "[[", 1), sapply(Holness.slns, "[[", 3)))
Holness.parms <- Holness.parms[Holness.parms[,3] == 1,,drop = F]

# Calculate MLEs for log-scale mean and SD
# (Only one vector of parameters resulted from among those obtaining convergence)
Holness.mle <- c(mu = -Holness.slns[1,1], sg = 1) / Holness.slns[1,2]
```

```
ln.cond.mn <- function(mle.vec, cut.pt)
{
  # Calculate mean of a lognormal random variable conditional on it
  # exceeding the value ‘cut.pt’
  m = mle.vec["mu"]; s = mle.vec["sg"]
  u0 = (log(cut.pt) - m) / s
  mn = exp(m + s^2 / 2) * pnorm(-u0 + s) / pnorm(-u0)
  return(mn)
}

# Calculate mean estimate of high exposure group from Holness using MLEs
#  of log-scale mean and SD
Holness.mn = ln.cond.mn(Holness.mle, 8.8)

# Calculate expected frequencies by group from MLEs of log-scale mean and SD
exp.prb.upp <- pnorm((log(c(Holness.x, Inf)) - Holness.mle["mu"]) / Holness.mle["sg"])
exp.prb.low <- pnorm((log(c(0, Holness.x)) - Holness.mle["mu"]) / Holness.mle["sg"])
Holness.freq.exp <- 58 * (exp.prb.upp - exp.prb.low)

# Calculate test statistic and p-value for chi-square goodness-of-fit test
#  for estimated lognormal distribution
Holness.ln.chisq <- sum((Holness.freq - Holness.freq.exp)^2 / Holness.freq.exp)
Holness.ln.chisq.p <- 1 - pchisq(Holness.ln.chisq, 1)

ln.ran.grouping <- function (n, parms, B, lims)
{
  # Generate B samples of n observations from lognormal distribution
  #  with log-scale mean "parms.mu" and SD "parms.sg".
  # Group each sample using 'lims' to define the groups and generate
  #  frequencies.

  # Generate samples:
  tot = B * n
  ln.reals = exp(parms["sg"] * rnorm(tot) + parms["mu"])
  dsets = matrix(ln.reals, ncol = n)

  # Divide each sample into groups and calculate frequencies per group
  int.nums = NULL; lims = c(0, lims, Inf)
  for(i in 1:(length(lims)-1)) {
    int.id = 1 * (ln.reals >= lims[i] & ln.reals < lims[i+1])
    int.id.mat = matrix(int.id, ncol = n)
    int.nums = cbind(int.nums, rowSums(int.id.mat))
  }

  # Coalesce samples and frequencies
  lasted = list(data.sets = dsets, freqs = int.nums)
  return(lasted)
```

```
}

HBoot <- function(n, parms, cutoff, B, lims, k)
{
  # Generate B samples of n observations from lognormal distribution
  #  with log-scale mean "parms.mu" and SD "parms.sg"
  # For each sample, compute MLEs of mu and sigma, and estimates of mean
  #  of lognormal random variable conditioned on it exceeding 'cutoff'

  # Generate samples
  samples = ln.ran.grouping(n, parms, B, lims)$freqs

  # Compute MLEs of mu and sigma for each sample
  musgs = NULL
  for(i in 1:B)  {
    vst.i = HSolve.vst.derv(lims, samples[i,], k)
    sln.i.lst = apply(vst.i, 1, nleqslv, fn = HSolve.fn, jac = HSolve.jac, lims = lims, freqs = samples[i,])
    sln.i = t(rbind(sapply(sln.i.lst, "[[", 1), sapply(sln.i.lst, "[[", 3)))
    if(min(sln.i[,3]) > 1.5) musg.i = rep(NA, 3)
    else {
      z.i = sln.i[sln.i[,3] == 1,,drop = F]
      mults.i = 1 * (max(c(max(z.i[,1]) - min(z.i[,1]), max(z.i[,2]) - min(z.i[,2]))) > 1e-3)
      musg.i = c(c(-z.i[1,1], 1) / z.i[1,2], mults.i)
    }
    musgs = rbind(musgs, c(samples[i,], musg.i))
  }

  # Compute estimates of conditioned mean
  mus = musgs[,5]; sgs = musgs[,6]
  u0s = (log(cutoff) - mus) / sgs
  mns.vec = exp(mus + sgs^2 / 2) * pnorm(sgs - u0s) / pnorm(-u0s)

  # Coalesce results
  rest = cbind(musgs, u0s, mns.vec)
  return(rest)
}

HB.spec.freqs <- function(n, cutoff, freq.mat, lims, k)
{
  # For set of pre-defined grouped lognormal samples, compute MLEs
  #  of mu and sigma, and estimates of mean of lognormal random variable
  #  conditioned on it exceeding 'cutoff'

  # Compute MLEs of mu and sigma for each sample
  musgs = NULL
  for(i in 1:nrow(freq.mat))  {
    vst.i = HSolve.vst.derv(lims, freq.mat[i,], k)
    sln.i.lst = apply(vst.i, 1, nleqslv, fn = HSolve.fn, jac = HSolve.jac, lims = lims, freqs = freq.mat[i,])
```

```
    sln.i = t(rbind(sapply(sln.i.lst, "[[", 1), sapply(sln.i.lst, "[[", 3)))
    if(min(sln.i[,3]) > 1.5) musg.i = rep(NA, 3)
    else {
      z.i = sln.i[sln.i[,3] == 1,,drop = F]
      mults.i = 1 * (max(c(max(z.i[,1]) - min(z.i[,1]), max(z.i[,2]) - min(z.i[,2]))) > 1e-3)
      musg.i = c(c(-z.i[1,1], 1) / z.i[1,2], mults.i)
    }
    musgs = rbind(musgs, c(freq.mat[i,], musg.i))
  }

  # Compute estimates of conditional mean
  mus = musgs[,5]; sgs = musgs[,6]
  u0s = (log(cutoff) - mus) / sgs
  mns.vec = exp(mus + sgs^2 / 2) * pnorm(sgs - u0s) / pnorm(-u0s)

  # Coalesce results
  rest = cbind(musgs, u0s, mns.vec)
  return(rest)
}

# Based on grouped data form Holness (1989), generate 10,000 bootstrap
#  estimates of mean of lognormal random variable conditioned on it
#  exceeding 8.8.
HBoot.mns.tmp = HBoot(58, Holness.mle, 8.8, 1e4, HSolve.x, 5)

# Separate estimates based on samples that yielded convergence from those
#  that did not
HBoot.ms.nona = HBoot.mns.tmp[!is.na(HBoot.mns.tmp[,5]),]
HBoot.ms.na = HBoot.mns.tmp[is.na(HBoot.mns.tmp[,5]),]

# Extract samples that yielded multiple sets of estimates
HBoot.ms.mults = HBoot.ms.nona[HBoot.ms.nona[,7] > 1.5,]

# Compute percentiles of mean distributions based on samples yielding
#  convergence
Bpercs = c(0.01, 0.05, 0.10, 0.50, 0.90, 0.95, 0.99)
qmn.nona = quantile(HBoot.ms.nona[,9], Bpercs)

# Rerun samples which did not yield convergence, using more starting
#  values for parameters
# Bind new results into one matrix
HBoot.na.rerun = HB.spec.freqs(58, 8.8, HBoot.ms.na[,1:4], HSolve.x, 20)
HBoot.mns = rbind(HBoot.ms.nona, HBoot.na.rerun)
rm(HBoot.mns.tmp)

# Extract means from matrix of estimates and compute percentiles
HBoot.mns = HBoot.mns[,9]
HBoot.mn.perc = quantile(c(HBoot.mns), Bpercs)
```