Stat 534 Homework 2

1.

(a)

```r
mice <- data.frame(
  time = 1:6,
  ni = c(20, 14, 12, 11, 11, NA),
  ui = c(20, 10, 11, 7, 6, NA),
  mi = c(0, 4, 1, 4, 5, NA),
  Mi = c(0, 20, 30, 41, 48, 54)
)

lnlM0 <- function(param, data) {
  # data is a vector of 3 valules.  In order, they are:
  #   #capture occasions,
  #   # seen at least once (Mt+1),
  #   total number of captures

  # param is a vector of 2 values: N and p

  t <- data[1]
  Mt1 <- data[2]
  ndot <- data[3]

  N <- param[1]
  p <- param[2]

  lfactorial(N) - lfactorial(N-Mt1) + ndot*log(p) + (t*N-ndot)*log(1-p)
}

t <- 5
Mt1 <- 54
ndot <- sum(mice$ni[1:5])

mice.m0 <- optim(c(100, 0.2), lnlM0, data = c(t, Mt1, ndot), method = 'BFGS',
                 control = list(fnscale = -1), hessian = T)
mice.m0$par[1]
```

```
## [1] 115.397
```

```r
sqrt(diag(solve(-mice.m0$hessian)))[1]
```

```
## [1] 23.97924
```

(b)

```r
lnlMt <- function(param, data) {
  # data is a vector of t+2 values.  In order, they are:
  #   # capture occasions,
  #   # seen at least once (Mt+1),
  #   # captures at each occasion

  # param is a vector of t+1 values: N and t capture probabilities

  t <- data[1]
  Mt1 <- data[2]
  n <- data[-c(1:2)]

  N <- param[1]
  p <- param[-1]

  lfactorial(N) - lfactorial(N-Mt1) + sum(n*log(p)) + sum((N-n)*log(1-p))
  }

mice.mt <- optim(c(100, rep(0.2, 5)), lnlMt, data = c(t, Mt1, mice$ni[1:5]),
                 method = 'BFGS', control = list(fnscale = -1), hessian = T)

mice.mt$par[1]
```

```
## [1] 113.8964
```

```r
sqrt(diag(solve(-mice.mt$hessian)))[1]
```

```
## [1] 23.54166
```

(c)

```r
lnlMb <- function(param, data) {
   # data is a vector of 4 valules.  In order, they are:
  #   t, #capture occasions,
  #   Mt1, # seen at least once (Mt+1)
  #   M., total # marked
  #   m., total marked captures

  # param is a vector of 3 values: N, p, and c

  t <- data[1]
  Mt1 <- data[2]
  Mdot <- data[3]
  mdot <- data[4]

  N <- param[1]
  p <- param[2]
  c <- param[3]

  lfactorial(N) - lfactorial(N-Mt1) + Mt1*log(p) + (t*N - Mt1 - Mdot)*log(1-p) +
    mdot*log(c) + (Mdot - mdot)*log(1-c)
}
```

```
Mdot <- sum(mice$Mi[1:5])
mdot <- sum(mice$mi[1:5])

mice.mb <- optim(c(100, 0.2, 0.2), lnlMb, data = c(t, Mt1, Mdot, mdot),
                 method = 'BFGS', control = list(fnscale = -1), hessian = T)

mice.mb$par[1]
```

```
## [1] 66.80011
```

```
sqrt(diag(solve(-mice.mb$hessian)))[1]
```

```
## [1] 9.451303
```

(d)

```
mice.lnL <- c(M0 = mice.m0$value, Mt = mice.mt$value, Mb = mice.mb$value)
mice.par <- c(M0 = length(mice.m0$par),
              Mt = length(mice.mt$par),
              Mb = length(mice.mb$par))
mice.AIC <- -2*mice.lnL + 2*mice.par
mice.Nhat <- c(M0 = mice.m0$par[1], Mt = mice.mt$par[1], Mb = mice.mb$par[1])
rbind(mice.par, mice.Nhat, mice.AIC)
```

```
##                  M0        Mt        Mb
## mice.par    2.00000   6.00000   3.00000
## mice.Nhat 115.39700 113.89643  66.80011
## mice.AIC  -60.47867 -56.95052 -62.80249
```

(f)

```
# here's the information you need to do the model average

se.N <- function(l) {
  sqrt(diag(solve(-l$hessian)))[1]
}

mice.Nse <- c(M0 = se.N(mice.m0), Mt = se.N(mice.mt), Mb = se.N(mice.mb))

round(rbind(lnL = mice.lnL,
  Npar = mice.par,
  AIC = mice.AIC,
  Nhat = mice.Nhat,
  Nse = mice.Nse), 2)
```

```
##          M0     Mt     Mb
## lnL   32.24  34.48  34.40
## Npar   2.00   6.00   3.00
## AIC  -60.48 -56.95 -62.80
## Nhat 115.40 113.90  66.80
## Nse   23.98  23.54   9.45
```

```
# the computations are:
deltaAIC <- mice.AIC - min(mice.AIC)
```

```
weight <- exp(-deltaAIC/2)
weight <- weight/sum(weight)

# model averaged estimate
N.ma <- sum(mice.Nhat*weight)

# Buckland formula
sum(weight*sqrt(mice.Nse^2 + (mice.Nhat - N.ma)^2))
```

```
## [1] 23.20562
```

```
# model averaged standard error, revised formula
sqrt(sum(weight*(mice.Nse^2 + (mice.Nhat - N.ma)^2)))
```

```
## [1] 26.04212
```

(h)

```
# fitting Mb using f0
lnlMb2 <- function(param, data) {
   # data is a vector of 4 valules.  In order, they are:
  #   t, #capture occasions,
  #   Mt1, # seen at least once (Mt+1)
  #   M., total # marked
  #   m., total marked captures

  # param is a vector of 3 values: f0, p, and c

  t <- data[1]
  Mt1 <- data[2]
  Mdot <- data[3]
  mdot <- data[4]

  f0 <- param[1]
  p <- param[2]
  c <- param[3]
  N <- f0 + Mt1

  lfactorial(N) - lfactorial(N-Mt1) + Mt1*log(p) + (t*N - Mt1 - Mdot)*log(1-p) +
    mdot*log(c) + (Mdot - mdot)*log(1-c)
}

mice.mb2 <- optim(c(15, 0.2, 0.2), lnlMb2, data = c(t, Mt1, Mdot, mdot),
                  method = 'BFGS', control = list(fnscale = -1), hessian = T)

mice.mb2$par[1]
```

```
## [1] 12.79317
```

```
sqrt(diag(solve(-mice.mb$hessian)))[1]
```

```
## [1] 9.451303
```

```
mice.mb2$value
```

```
## [1] 34.40125
```

```
mice.mb$value
```

```
## [1] 34.40125
```

(i)

```
# the 95% CI is
mice.mb$par[1] + c(-1, 1)*se.N(mice.mb)*qnorm(0.975)
```

```
## [1] 48.27590 85.32432
```

(j)

```
# fitting Mb using log f0
lnlMb3 <- function(param, data) {
   # data is a vector of 4 valules.  In order, they are:
  #   t, #capture occasions,
  #   Mt1, # seen at least once (Mt+1)
  #   M., total # marked
  #   m., total marked captures

  # param is a vector of 3 values: log(f0), p, and c

  t <- data[1]
  Mt1 <- data[2]
  Mdot <- data[3]
  mdot <- data[4]

  f0 <- exp(param[1])
  p <- param[2]
  c <- param[3]
  N <- f0 + Mt1

  lfactorial(N) - lfactorial(N-Mt1) + Mt1*log(p) + (t*N - Mt1 - Mdot)*log(1-p) +
    mdot*log(c) + (Mdot - mdot)*log(1-c)
}

mice.mb3 <- optim(c(log(15), 0.2, 0.2), lnlMb3, data = c(t, Mt1, Mdot, mdot),
                 method = 'BFGS', control = list(fnscale = -1), hessian = T)

# the 95% CI for the log(f0) is
mice.mb3$par[1] + c(-1, 1)*se.N(mice.mb3)*qnorm(0.975)
```

```
## [1] 1.102128 3.996276
```

```
# backtransform, the 95% CI for f0 is
exp(mice.mb3$par[1] + c(-1, 1)*se.N(mice.mb3)*qnorm(0.975))
```

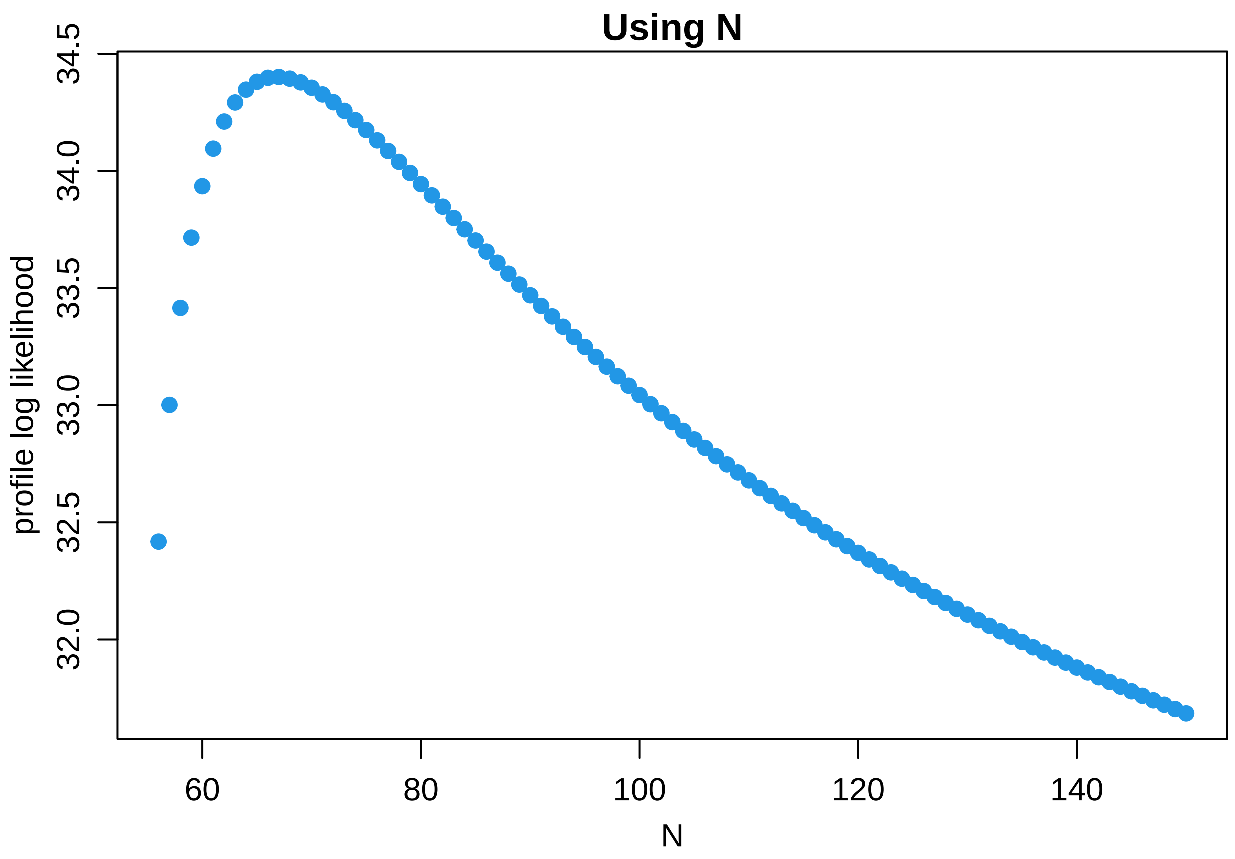
```
## [1]  3.010566 54.395187
```

```
# the 95% CI for N is
exp(mice.mb3$par[1] + c(-1, 1)*se.N(mice.mb3)*qnorm(0.975)) + Mt1
```

```
## [1]  57.01057 108.39519
```

(k)

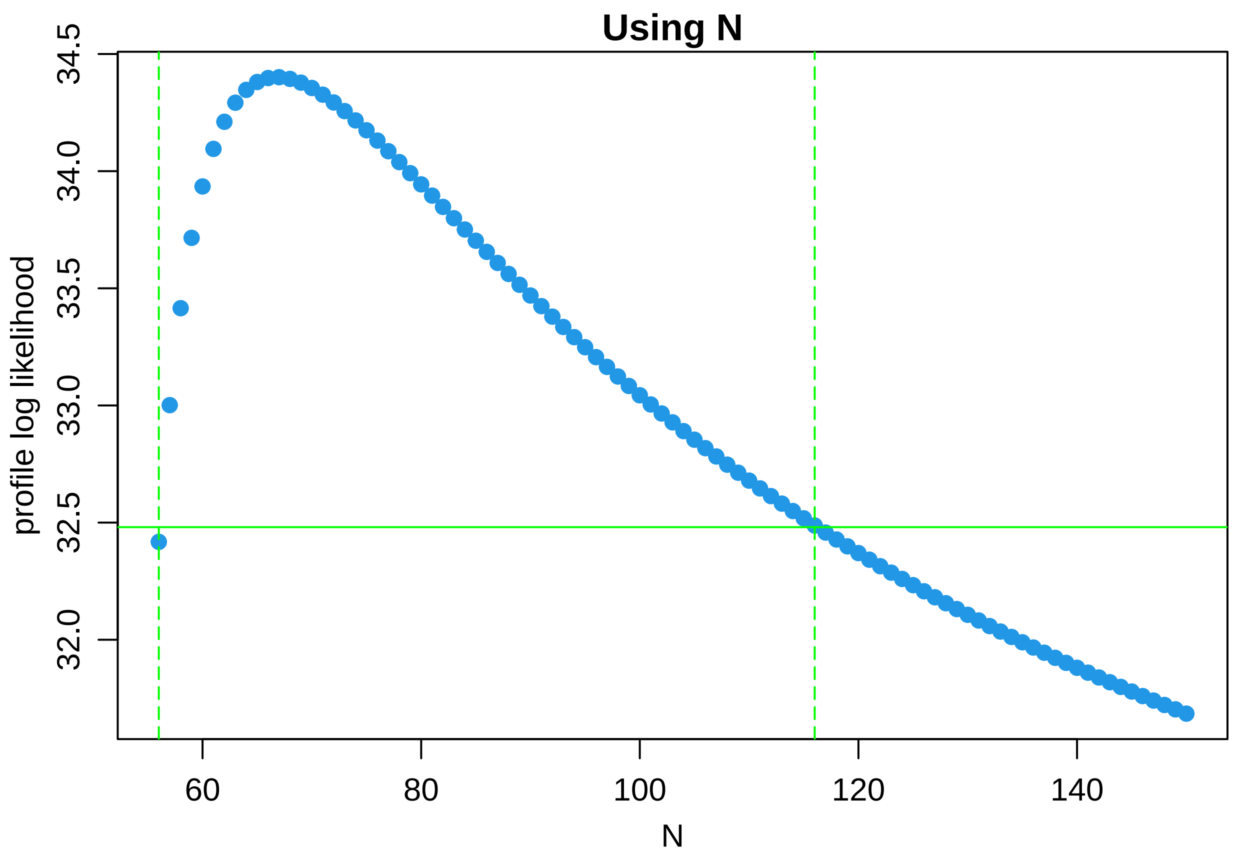
```
# profile lnl
lnlMbp <- function(param, data) {
  # par is parameters except for N, fn is the full lnl function,
  # N is the specific value of N and ... are any other arguments

  # param is the single value: N

  # data is a vector of 4 valules.  In order, they are:
  #   t, #capture occasions,
  #   Mt1, # seen at least once (Mt+1)
  #   M., total # marked
  #   m., total marked captures

  t <- data[1]
  Mt1 <- data[2]
  Mdot <- data[3]
  mdot <- data[4]

  N <- param[1]
  p <- Mt1/(N*t - Mdot)
  c <- mdot / Mdot

  lfactorial(N) - lfactorial(N-Mt1) + Mt1*log(p) + (t*N - Mt1 - Mdot)*log(1-p) +
    mdot*log(c) + (Mdot - mdot)*log(1-c)
}

# compute profile lnl for a range of N values
allN <- 56:150
alllnl_n <- rep(NA, length(allN))

# bucket to hold profile lnl value
for (i in 1:length(allN)) {
  alllnl_n[i] <- lnlMbp(allN[i], data = c(t, Mt1, Mdot, mdot))
}

# pdf('profile_1k.pdf', height=5, width=7)
par(mar=c(3,3,1,0)+0.3, mgp=c(2,0.8,0))

plot(allN, alllnl_n, pch=19, col=4,
  xlab='N', ylab='profile log likelihood',
  main='Using N')
```



```
#dev.off()
```

(l)

```
mice.mbp <- optim(80, lnlMbp, data = c(t, Mt1, Mdot, mdot), method = 'BFGS',
                    control = list(fnscale = -1), hessian = T)

#pdf('profile_1l.pdf', height=5, width=7)
par(mar=c(3,3,1,0)+0.3, mgp=c(2,0.8,0))
plot(allN, alllnl_n, pch=19, col=4,
  xlab='N', ylab='profile log likelihood',
  main='Using N')
abline(h=mice.mbp$value - qchisq(0.95, 1)/2, lty=1, col = "green")
abline(v=c(allN[1], allN[61]), col=c("green","green"), lty = 5)
```

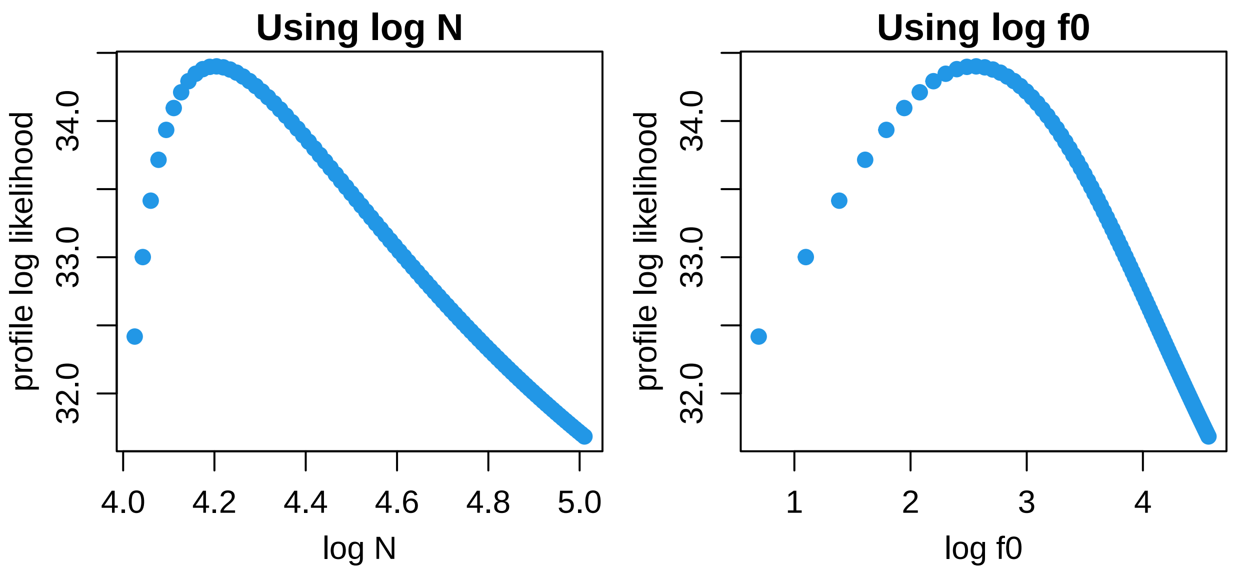


```
#dev.off()

# find the intersections of the horizontal line and the curve
#order(abs(alllnl_n - (mice.mbp$value - qchisq(0.95, 1)/2)))
#the 95\% profile CI for N is
c(allN[1], allN[61])
```

```
## [1]  56 116
```

(m)

```
lnlMbpln <- function(param, data) {
  # profile lnl for Mb but using log N
  # param is a single value, log N

  N <- exp(param[1])  # compute N to pass to lnlM0p

  lnlMbp(N, data)
}

lnlMbplf0 <- function(param, data) {
  # profile lnl for Mb but using log f0
  # param is a single value, log f0

  N <- exp(param[1]) + data[2]    # f0 + Mt+1

  lnlMbp(N, data)
}

# compute profile lnl for a range of N values
allN <- 56:150
alllnl_ln <- rep(NA, length(allN))
alllnl_lf0 <- rep(NA, length(allN))

# bucket to hold profile lnl values
for (i in 1:length(allN)) {
  alllnl_ln[i] <- lnlMbpln(log(allN[i]), data = c(t, Mt1, Mdot, mdot))
  alllnl_lf0[i] <- lnlMbplf0(log(allN[i]-Mt1), data = c(t, Mt1, Mdot, mdot))
}

#pdf('profile_1m.pdf', height=3.5, width=7.5)
par(mar=c(3,3,1,0)+0.3, mgp=c(2,0.8,0))
par(mfrow=c(1,2))

plot(log(allN), alllnl_ln, pch=19, col=4,
  xlab='log N', ylab='profile log likelihood',
  main='Using log N')

plot(log(allN-Mt1), alllnl_lf0, pch=19, col=4,
  xlab='log f0', ylab='profile log likelihood',
  main='Using log f0')
```



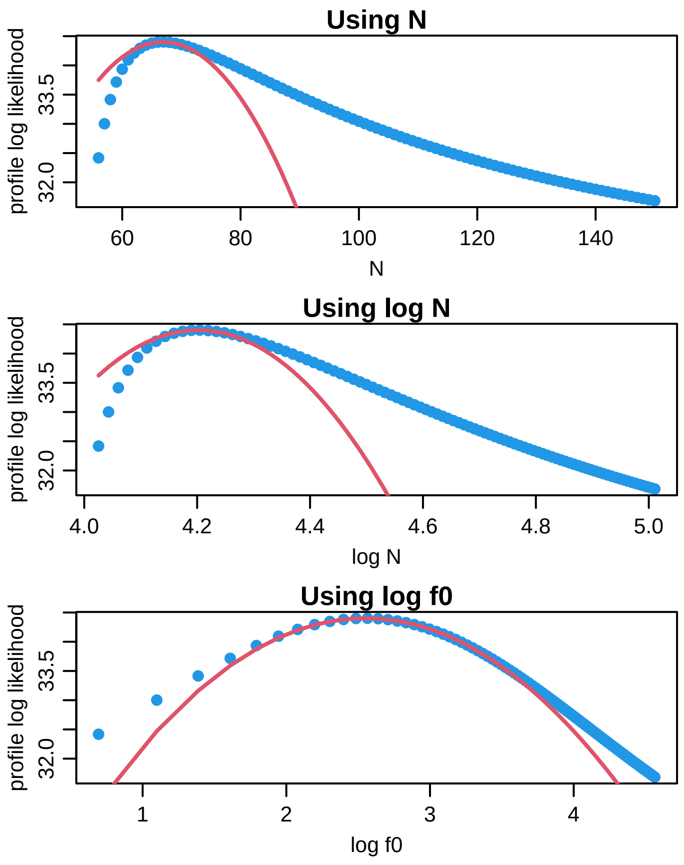
```
#dev.off()
```

(n)

```
addQuad <- function(mle, se, lnl, Xvalues, plot=T) {
#  input:
#    mle: estimate
#    lnl: lnl at that estimate
#    se: se at that estimate
#    Xvalues: X values at which to plot the quadratic approx.
#    plot: T/F: whether or not to plot the quad. approx.
#  output: (invisible):
#    data.frame with X values and the quad. approx at those values

#  draws a plot of the quadratic if plot=T
#  if you want to add the quadratic approx. to a pre-existing plot,
#    set plot=F, save the output of this function,
#    then add the quadratic approx to your plot,
#  i.e. use points() or lines():  see code below for example

  C <- -1/(2*se^2)
  lnlX <- lnl + C*(Xvalues - mle)^2
  if (plot) {
    plot(Xvalues, lnlX)
    }
  invisible(data.frame(x=Xvalues, y=lnlX))
  }
```

```
# se of Nhat = sqrt(-1/H) where H is scalar.
# The c() is to tell R this is a scalar, not a 1x1 matrix.

# for N
temp_n <- addQuad(mle = mice.mbp$par, se = sqrt(c(-1/mice.mbp$hessian)),
                lnl = mice.mbp$value, 56:150, plot=F)

# for log N
mice.mbpln <- optim(log(80), lnlMbpln, data = c(t, Mt1, Mdot, mdot), method = 'BFGS',
                    control = list(fnscale = -1), hessian = T)

temp_ln <- addQuad(mle = mice.mbpln$par, se = sqrt(c(-1/mice.mbpln$hessian)),
                   lnl = mice.mbpln$value, log(56:150), plot=F)

# for log f0
mice.mbplf0 <- optim(log(80-Mt1), lnlMbplf0, data = c(t, Mt1, Mdot, mdot),
                     method = 'BFGS', control = list(fnscale = -1), hessian = T)

temp_lf0 <- addQuad(mle = mice.mbplf0$par, se = sqrt(c(-1/mice.mbplf0$hessian)),
                lnl = mice.mbplf0$value, log(56:150-Mt1), plot=F)

#pdf('profile_1n.pdf', height=7, width=4.5)
par(mar=c(3,3,1,3)+0.3, mgp=c(1.9,0.7,0))
par(mfrow=c(3,1))

plot(allN, alllnl_n, pch=19, col=4,
  xlab='N', ylab='profile log likelihood',
  main='Using N')
# add it to the profile lnL plot
with(temp_n, lines(x, y, col=2, lwd=2))

plot(log(allN), alllnl_ln, pch=19, col=4,
  xlab='log N', ylab='profile log likelihood',
  main='Using log N')
# add it to the profile lnL plot
with(temp_ln, lines(x, y, col=2, lwd=2))
```

```r
plot(log(allN-Mt1), alllnl_lf0, pch=19, col=4,
  xlab='log f0', ylab='profile log likelihood',
  main='Using log f0')
# add it to the profile lnL plot
with(temp_lf0, lines(x, y, col=2, lwd=2))
```



```r
#dev.off()
```

2.

```r
library(RMark)
snouter1 <- import.chdata("snouter1.txt")
snouter2 <- import.chdata("snouter2.txt")

# for snouter1
run.snouter1 <- function() {
f0 <- list(formula=~1)
f0s <- list(formula=~1, share=TRUE)
ft <- list(formula=~time, share=TRUE)
fT <- list(formula=~Time, share=TRUE)
ftb <- list(formula=~time)
ftb2 <- list(formula = ~time + c, share=TRUE)

m0 <- mark(snouter1, model='Closed', model.parameters=list(p=f0s))
mt <- mark(snouter1, model='Closed', model.parameters=list(p=ft))
mT <- mark(snouter1, model='Closed', model.parameters=list(p=fT))
mb <- mark(snouter1, model='Closed', model.parameters=list(p=f0, c=f0))
mtb <- mark(snouter1, model='Closed', model.parameters=list(p=ftb, c=ftb))
mtb2 <- mark(snouter1, model='Closed', model.parameters=list(p=ftb2))
return(collect.models())
}

snouter1.models <- run.snouter1()
```

```r
print(model.table(snouter1.models, model.name=F), digits = 4)
```

```
##             p      c f0 model npar  AICc DeltaAICc    weight Deviance
## 5     ~time ~time ~1   mtb    8 -1296     0.000 7.150e-01     8.76
## 6 ~time + c        ~1  mtb2    6 -1294     2.198 2.382e-01    15.01
## 3     ~time        ~1    mt    5 -1290     6.226 3.179e-02    21.06
## 2        ~1    ~1 ~1    mb    3 -1288     7.735 1.495e-02    26.61
## 4     ~Time        ~1    mT    3 -1273    22.915 7.556e-06    41.79
## 1        ~1        ~1    m0    2 -1257    39.342 2.048e-09    60.22
```

```r
summary(snouter1.models[['mtb2']])
```

```
## Output summary for Closed model
## Name : p(~time + c)c()f0(~1)
##
## Npar :  6
## -2lnL:  -1305.779
## AICc :  -1293.702
##
## Beta
##                   estimate        se        lcl        ucl
## p:(Intercept)  -0.1088786 0.1525671 -0.4079102  0.1901530
## p:time2        -0.3403559 0.2184088 -0.7684372  0.0877254
## p:time3        -0.3176682 0.2815403 -0.8694871  0.2341507
## p:time4         0.2082565 0.3596322 -0.4966227  0.9131356
## p:c            -1.2841411 0.3652263 -1.9999847 -0.5682975
## f0:(Intercept)  3.3081253 0.5769002  2.1774009  4.4388497
##
##
## Real Parameter p
##          1         2         3        4
##  0.4728072 0.3895428 0.3949512 0.524824
##
##
## Real Parameter c
##          2         3         4
##  0.1501563 0.1530745 0.2341968
##
##
## Real Parameter f0
##         1
##  27.33384
```

```r
snouter1.models[['mtb2']]$results$derived
```

```
## $`N Population Size`
##   estimate       se      lcl      ucl
## 1 300.3338 15.76889 282.5598 351.1538
```

```r
# for snouter2
run.snouter2 <- function() {
f0 <- list(formula = ~1)
f0s <- list(formula = ~1, share=TRUE)
ft <- list(formula = ~time, share=TRUE)
fT <- list(formula = ~Time, share=TRUE)
ftb <- list(formula = ~time)
ftb2 <- list(formula = ~time + c, share=TRUE)

m0 <- mark(snouter2, model='Closed', model.parameters=list(p=f0s))
mt <- mark(snouter2, model='Closed', model.parameters=list(p=ft))
mT <- mark(snouter2, model='Closed', model.parameters=list(p=fT))
mb <- mark(snouter2, model='Closed', model.parameters=list(p=f0, c=f0))
mtb <- mark(snouter2, model='Closed', model.parameters=list(p=ftb, c=ftb))
mtb2 <- mark(snouter2, model='Closed', model.parameters=list(p=ftb2))
return(collect.models())
}

snouter2.models <- run.snouter2()
```

```r
print(model.table(snouter2.models, model.name=F), digits = 4)
```

```
##            p     c f0 model npar   AICc DeltaAICc    weight Deviance
## 5      ~time ~time ~1   mtb    8 -573.5     0.000 6.610e-01    3.459
## 3      ~time       ~1    mt    5 -571.4     2.127 2.282e-01   11.737
## 6 ~time + c        ~1  mtb2    6 -570.0     3.572 1.108e-01   11.140
## 1         ~1       ~1    m0    2 -553.4    20.167 2.761e-05   35.863
## 2         ~1    ~1 ~1    mb    3 -551.4    22.103 1.048e-05   35.778
## 4      ~Time       ~1    mT    3 -551.4    22.161 1.018e-05   35.836
```

```r
summary(snouter2.models[['mt']])
```

```
## Output summary for Closed model
## Name : p(~time)c()f0(~1)
##
## Npar :  5
## -2lnL:  -581.5065
## AICc :  -571.3998
##
## Beta
##                   estimate        se        lcl        ucl
## p:(Intercept)   -1.5348103 0.2289299 -1.9835128 -1.0861077
## p:time2         -0.9332444 0.2503814 -1.4239919 -0.4424969
## p:time3         -0.7770193 0.2398529 -1.2471311 -0.3069076
## p:time4         -0.0630834 0.2051230 -0.4651245  0.3389576
## f0:(Intercept)   5.2511536 0.2559337  4.7495235  5.7527836
##
##
## Real Parameter p
##         1         2        3         4
##  0.177291 0.0781282 0.090148 0.1682762
##
##
## Real Parameter c
##          2        3         4
##  0.0781282 0.090148 0.1682762
##
##
## Real Parameter f0
##         1
##  190.7862
```

```
snouter2.models[['mt']]$results$derived
```

```
## $`N Population Size`
##   estimate       se      lcl      ucl
## 1 332.7862 48.82862 258.4496 454.5762
```

```
cleanup(ask=F)
```