example_with_comparison_to_alternative_optimizer.R

Carl Schmertmann

Tue Aug 06 11:36:26 2019

```
# experiments with TOPALS_fit function
```

```
rm(list=ls())
graphics.off()
if (.Platform$OS.type == 'windows') windows(record=TRUE)

library(splines)

## code for TOPALS_fit function
source('TOPALS_fit function.R')
```

```
# EXAMPLES: SMALL POPULATIONS WITH RANDOM DEATHS
```

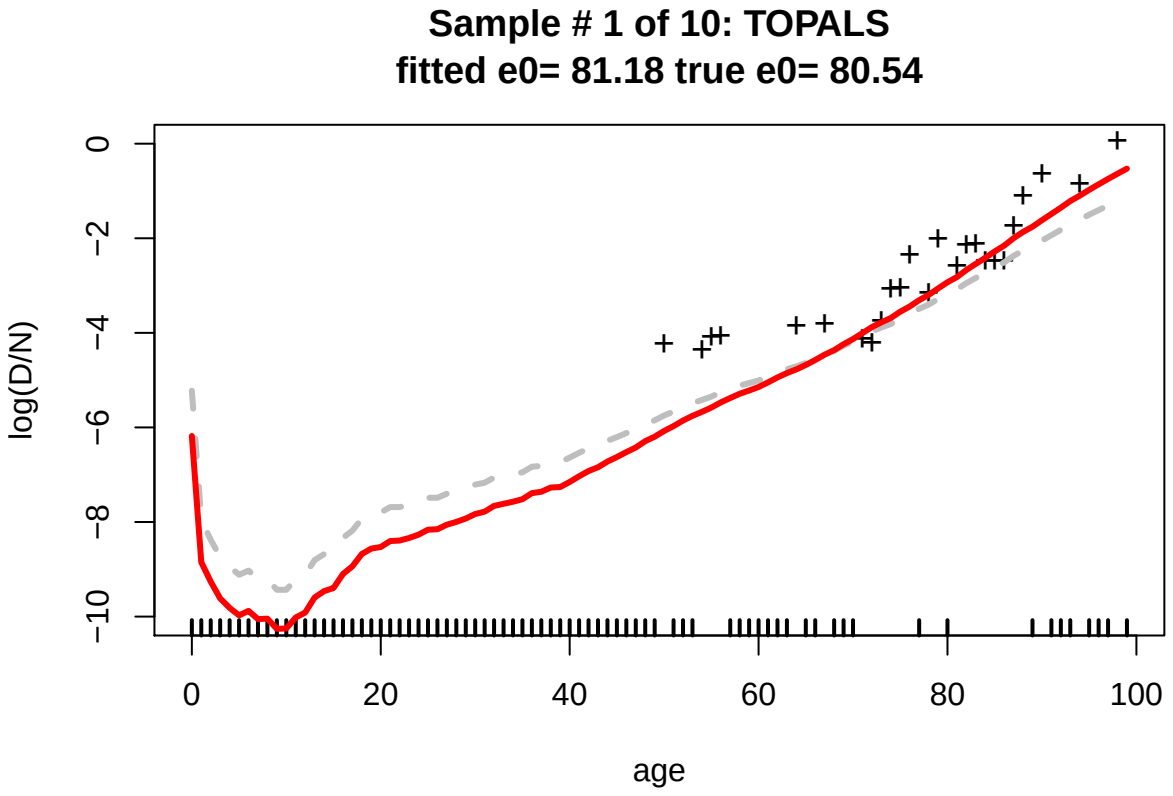
```
age = 0:99

## Population by Age (5000 females, age structure similar to Estonia 2010)
N = c(62, 62, 50, 65, 56, 56, 40, 50, 43, 50,
      42, 39, 34, 43, 45, 42, 53, 42, 45, 72,
      66, 65, 63, 67, 64, 78, 65, 69, 65, 60,
      70, 57, 46, 64, 58, 62, 59, 69, 69, 76,
      69, 56, 58, 61, 50, 52, 79, 65, 75, 78,
      73, 62, 76, 63, 83, 63, 61, 77, 84, 67,
      72, 62, 60, 60, 50, 55, 37, 48, 75, 51,
      59, 66, 71, 45, 46, 45, 44, 58, 50, 40,
      34, 42, 27, 35, 25, 25, 25, 18, 16,  5,
      2,  7,  2,  0,  5,  1,  1,  0,  1,  0)

## true mortality rates (Estonia 2010 from HMD)
mu = c(0.00246, 0.00064, 0.00026, 0.00014, 0.00014,
       0.00059, 0.00031, 0.00016,       0,       0,
       0.00034,       0, 0.00018,       0, 0.00016,
       0.00047, 0.00015, 0.00013,       0, 0.00023,
       0.00021,   2e-04,   1e-04, 0.00021, 0.00053,
       0.00073, 0.00021, 0.00085, 0.00033, 0.00033,
       0.00045, 0.00079, 0.00034, 0.00045, 0.00077,
       0.00066, 0.00077, 0.00099, 0.00074, 0.00096,
       0.00127, 0.00099, 0.00103,   7e-04, 0.00220,
       0.00177, 0.00263, 0.00247, 0.00171, 0.00189,
       0.00408, 0.00297, 0.00326, 0.00285, 0.00402,
       0.00441, 0.00584, 0.00485, 0.00475, 0.00484,
       0.00740, 0.00782, 0.00777, 0.01002, 0.01080,
       0.01277, 0.01026, 0.01350, 0.01316, 0.01467,
       0.01353, 0.01839, 0.02011, 0.02176, 0.02507,
       0.02801, 0.03008, 0.03849, 0.04071, 0.05160,
       0.05487, 0.06088, 0.06675, 0.07599, 0.08657,
       0.09597, 0.12556, 0.11733, 0.14262, 0.16010,
       0.17928, 0.20020, 0.22290, 0.24739, 0.27361,
       0.30150, 0.33094, 0.36176, 0.39377, 0.42671)
```

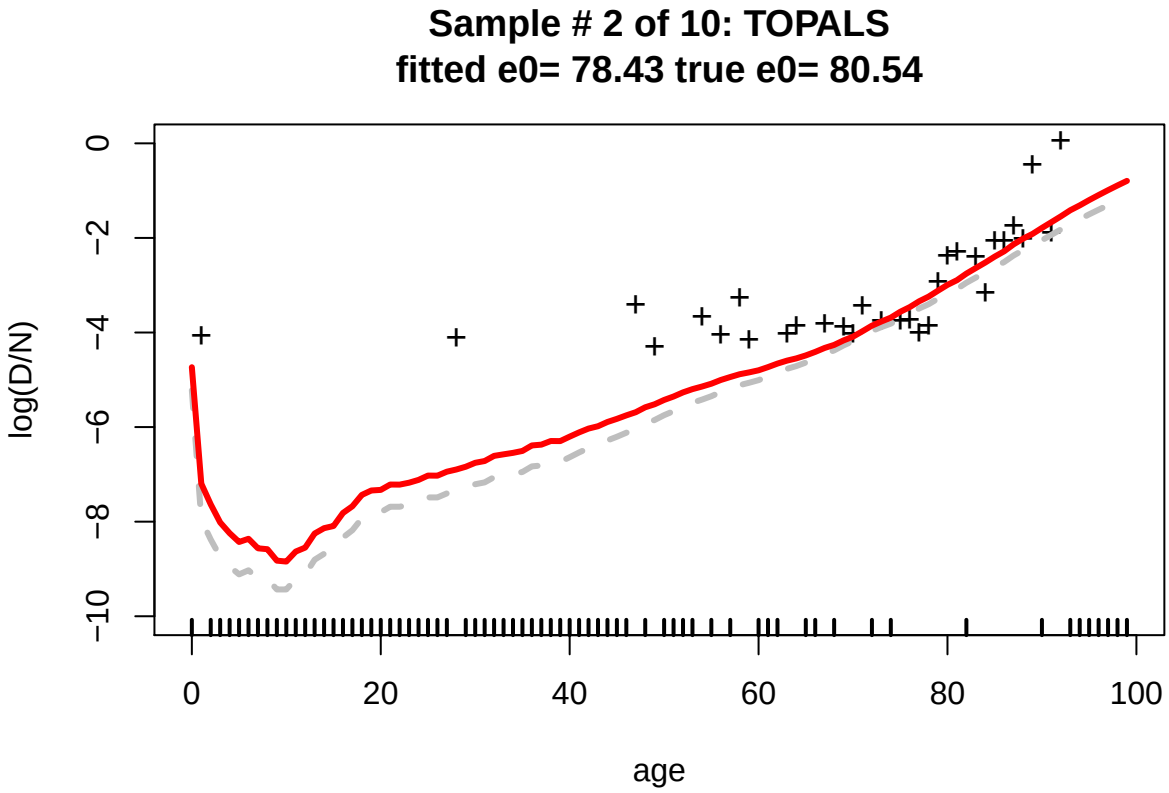
```
## A reasonable standard schedule of log mortality rates: USA females 2015 from HMD
this.std = c(-5.2232, -7.9576, -8.3774, -8.7403, -8.948,
             -9.1150, -9.0280, -9.2103, -9.2103, -9.4335,
             -9.4335, -9.2103, -9.1150, -8.8049, -8.6797,
             -8.6226, -8.3349, -8.1807, -7.9294, -7.8240,
             -7.7994, -7.6843, -7.6843, -7.6417, -7.5811,
             -7.4876, -7.4876, -7.4021, -7.3540, -7.2934,
             -7.2089, -7.1691, -7.0586, -7.0243, -6.9911,
             -6.9486, -6.8308, -6.8124, -6.7338, -6.7338,
             -6.6377, -6.5362, -6.4440, -6.3830, -6.2818,
             -6.2047, -6.1193, -6.0407, -5.9257, -5.8500,
             -5.7477, -5.6636, -5.5649, -5.4846, -5.4194,
             -5.3475, -5.2572, -5.1832, -5.1127, -5.0625,
             -5.0071, -4.9281, -4.8422, -4.7689, -4.7094,
             -4.6356, -4.5497, -4.4542, -4.3788, -4.2723,
             -4.1819, -4.0757, -3.9660, -3.8859, -3.8126,
             -3.6977, -3.6071, -3.4917, -3.4016, -3.2834,
             -3.1696, -3.0791, -2.9481, -2.8382, -2.7308,
             -2.6140, -2.5092, -2.3710, -2.2583, -2.1670,
             -2.0485, -1.935,  -1.8211, -1.6996, -1.6052,
             -1.5011, -1.4032, -1.3082, -1.2165, -1.1282)

# trapez approx of life expectancy from a logmx schedule over ages 0..99
e0 = function(logmx) {
  mx = exp(logmx)
  px = exp(-mx)
  lx = c(1,cumprod(px))
  return( sum(head(lx,-1) + tail(lx,-1)) / 2)
}
```

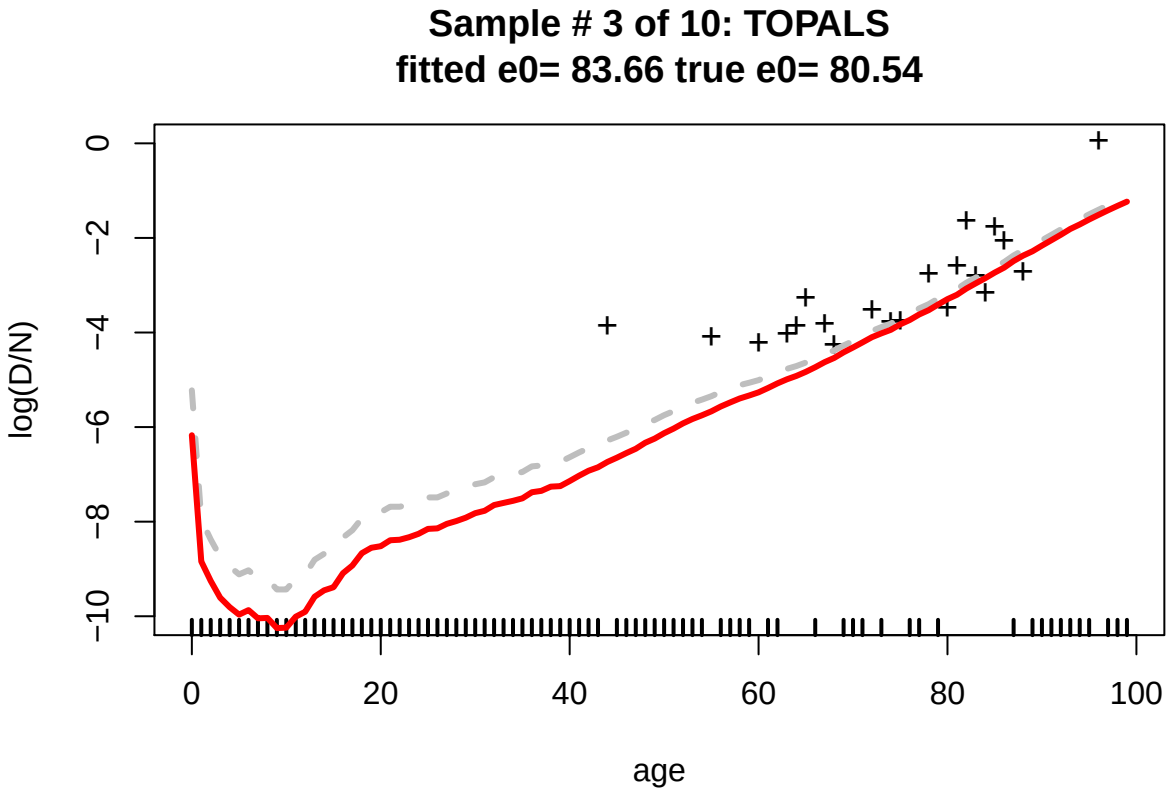
```
# EXAMPLE 1: TOPALS Defaults
```

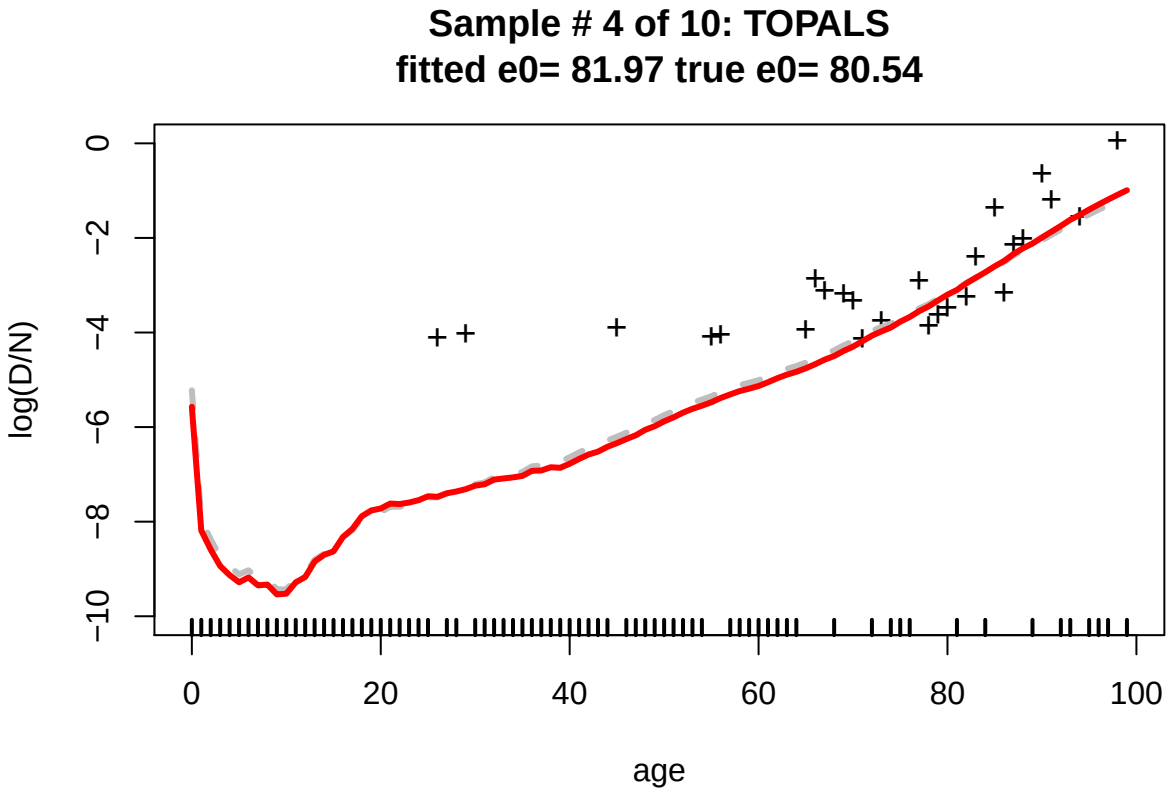
```
set.seed(6447100)   # change this if you want a different random dataset

## Draw random samples of deaths D ~ Poisson(N*mu). Fit and display TOPALS model.

nsim = 10

B  = bs( 0:99, knots=c(0,1,10,20,40,70), degree=1 )

true_e0 = round( e0(log(mu)), 2)

for (i in 1:nsim) {
    D = rpois(100, N*mu)   # random deaths

    fitted_alpha = TOPALS_fit( N, D, this.std)

    fitted_logmx = this.std + B %*% fitted_alpha

    this.e0    = e0(fitted_logmx)
    this.title = paste0('Sample # ',i, ' of ',nsim,': TOPALS\nfitted e0= ', round(this.e0,2),
                        ' true e0= ', true_e0)

    plot( age, log(D/N), ylim=c(-10,0), pch='+', cex=1.2, main=this.title)
```

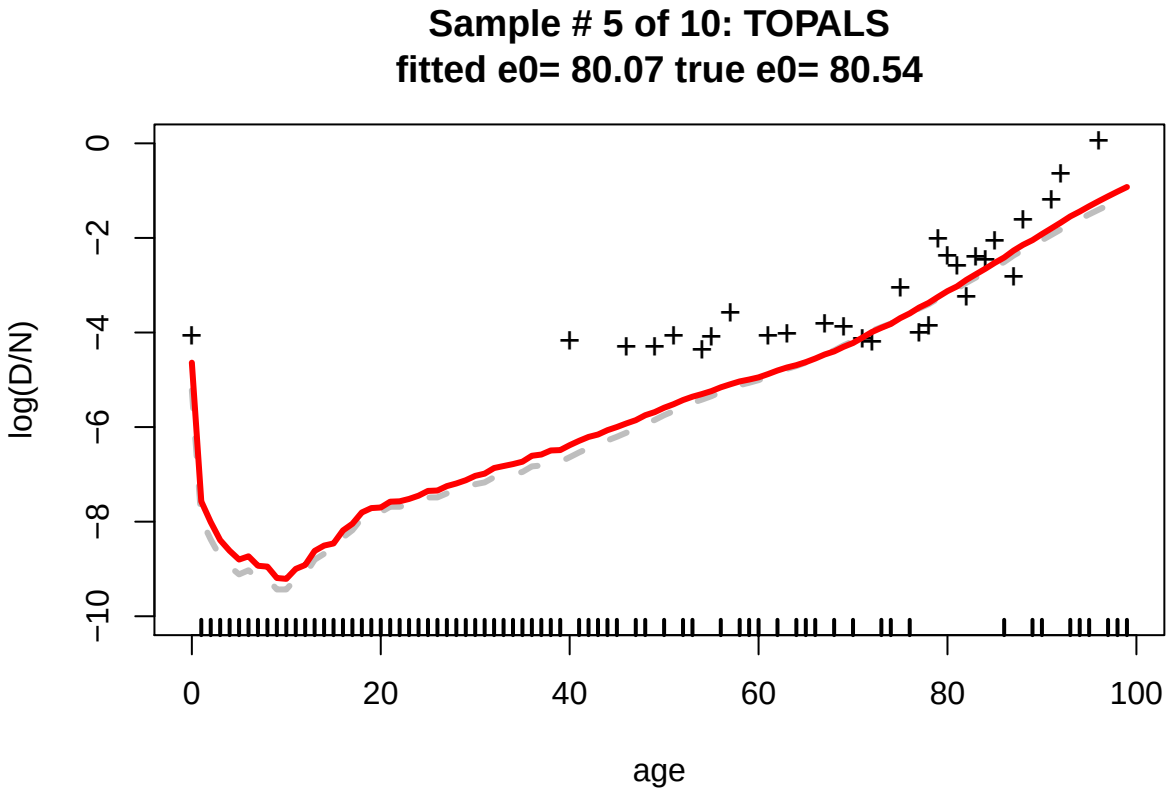
```
    rug( age[D==0], lwd=2)
    lines(age, this.std, type='l', lty=2, col='grey', lwd=3)
    lines(age, fitted_logmx, lty=1, col='red', lwd=3)

}
```

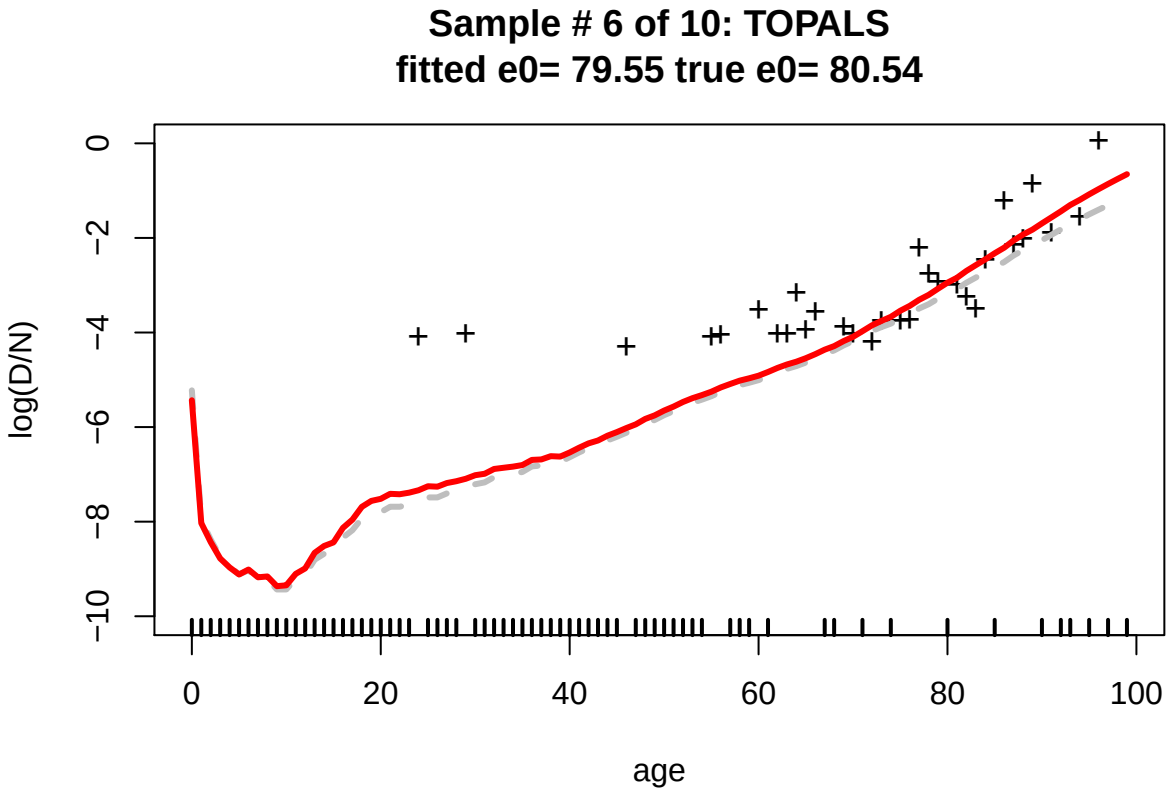


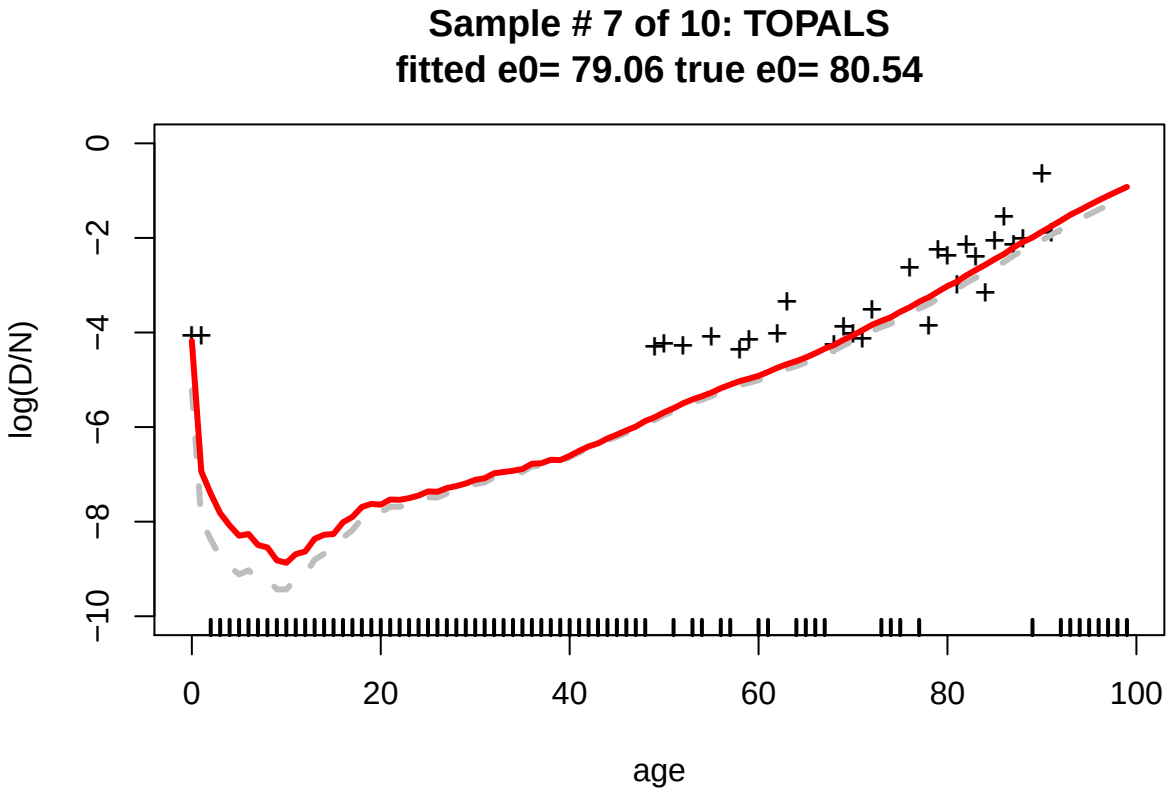

Sample # 2 of 10: TOPALS
fitted e0= 78.43 true e0= 80.54
log(D/N)
0
−2
−4
−6
−8
−10
0
20
40
60
80
100
age

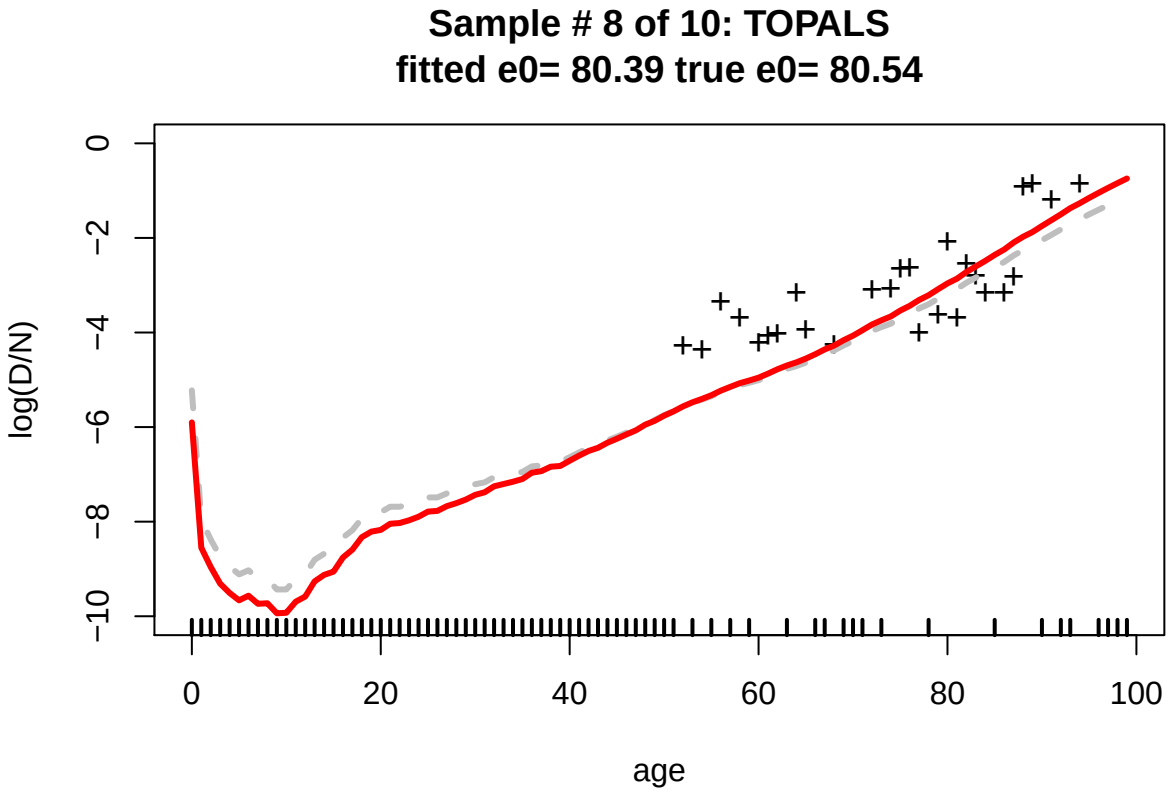

Sample # 3 of 10: TOPALS
fitted e0= 83.66 true e0= 80.54
log(D/N)
0
−2
−4
−6
−8
−10
0
20
40
60
80
100
age

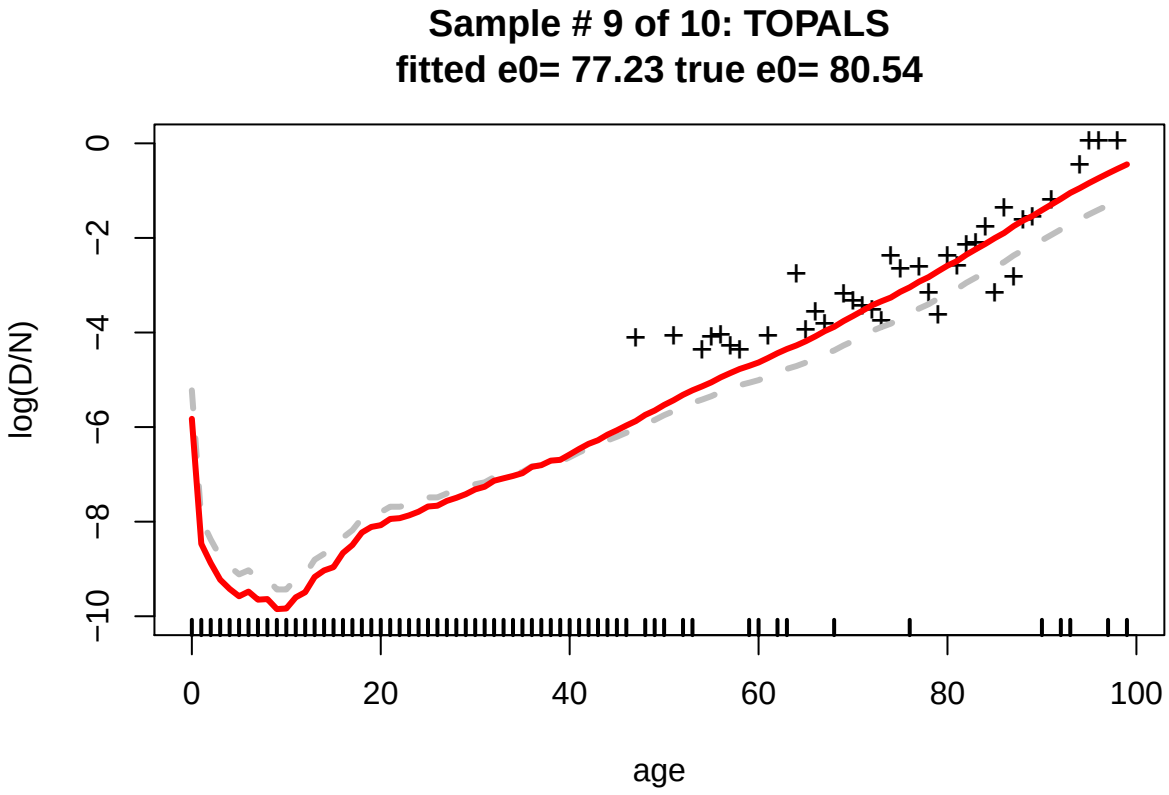

Sample # 4 of 10: TOPALS
fitted e0= 81.97 true e0= 80.54
log(D/N)
0
−2
−4
−6
−8
−10
0
20
40
60
80
100
age

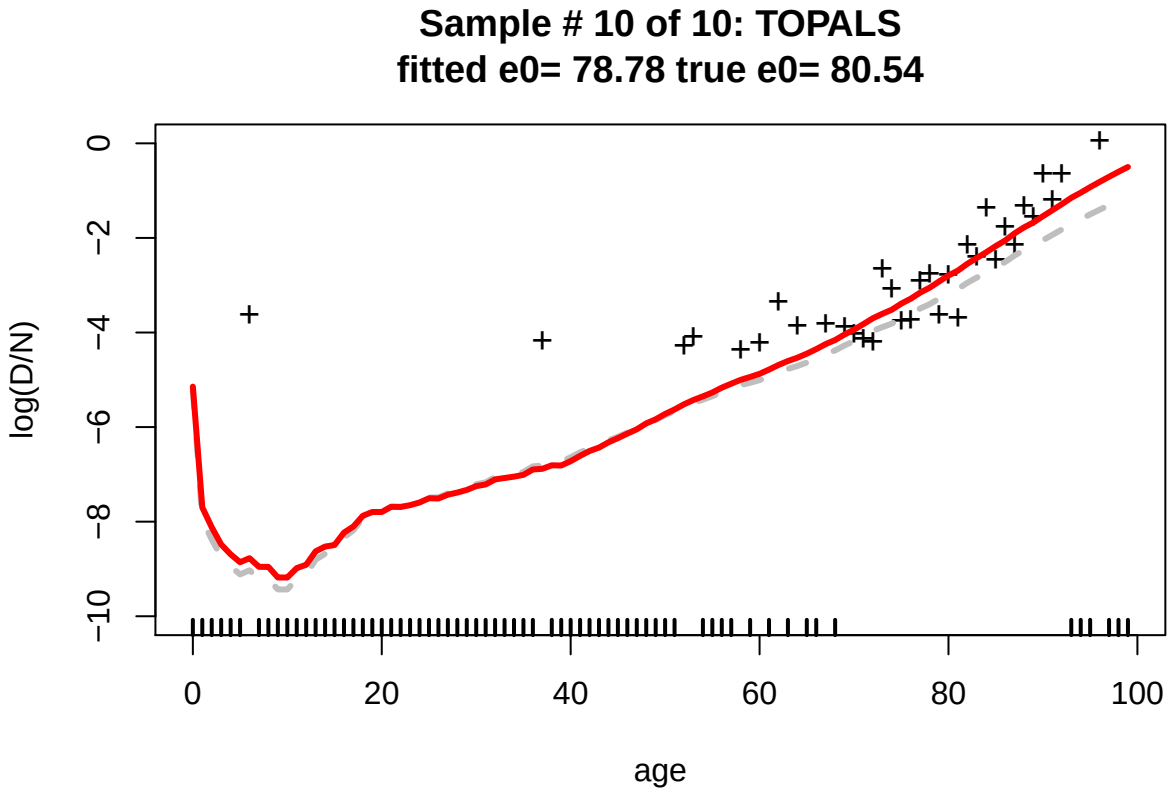

Sample # 5 of 10: TOPALS
fitted e0= 80.07 true e0= 80.54
log(D/N)
0
−2
−4
−6
−8
−10
0
20
40
60
80
100
age


Sample # 6 of 10: TOPALS
fitted e0= 79.55 true e0= 80.54
log(D/N)
0
−2
−4
−6
−8
−10
0
20
40
60
80
100
age


Sample # 7 of 10: TOPALS
fitted e0= 79.06 true e0= 80.54
log(D/N)
0
−2
−4
−6
−8
−10
0
20
40
60
80
100
age


Sample # 8 of 10: TOPALS
fitted e0= 80.39 true e0= 80.54
log(D/N)
0
−2
−4
−6
−8
−10
0
20
40
60
80
100
age

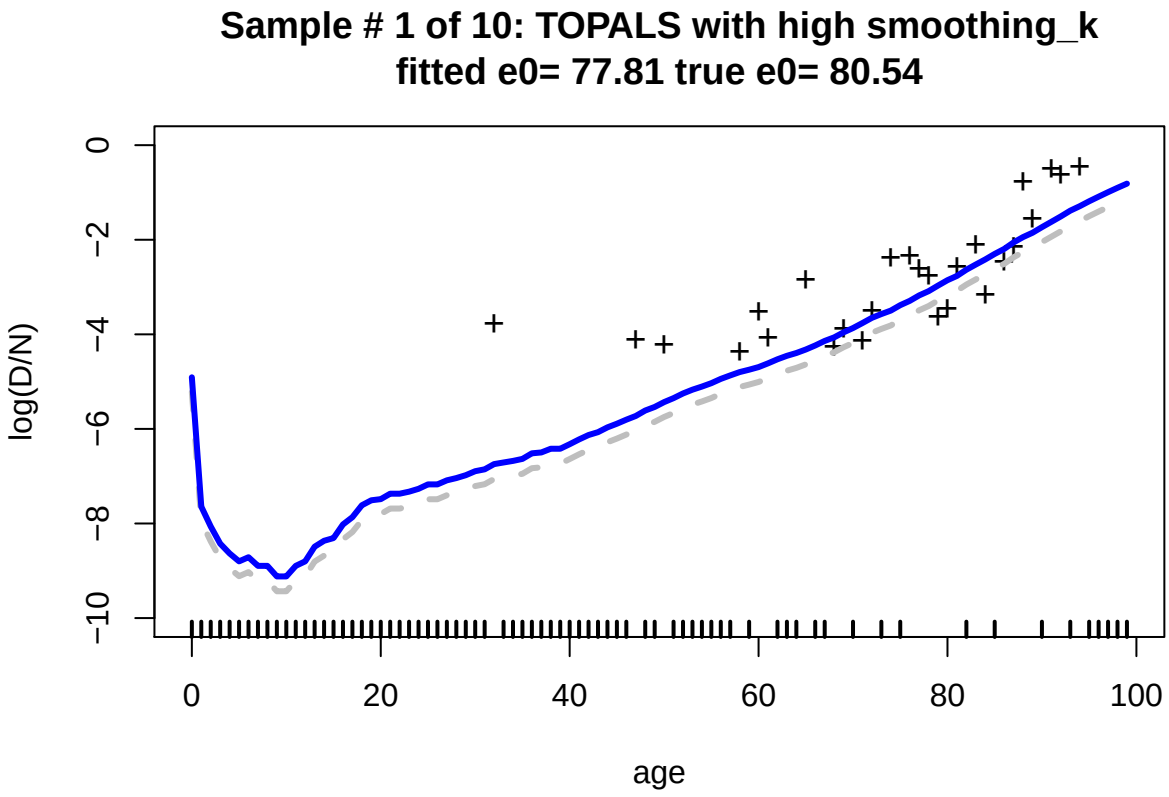

Sample # 9 of 10: TOPALS
fitted e0= 77.23 true e0= 80.54
log(D/N)
0
−2
−4
−6
−8
−10
0
20
40
60
80
100
age

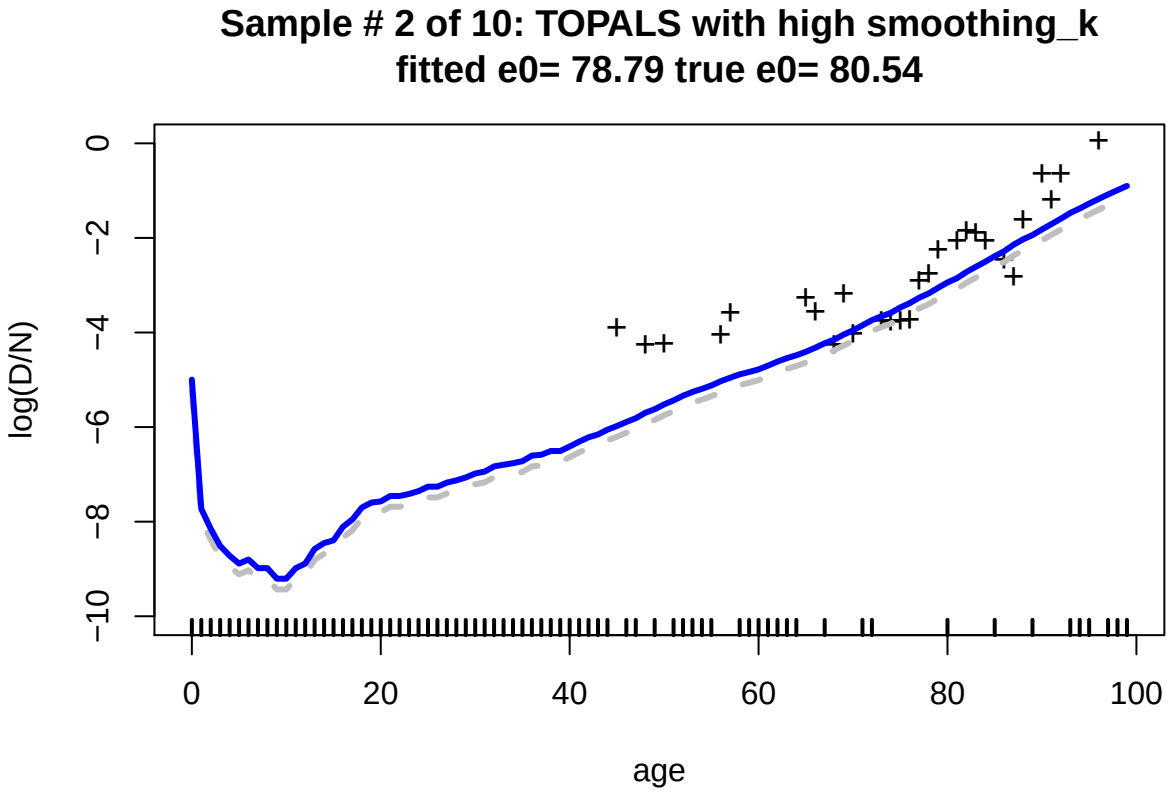


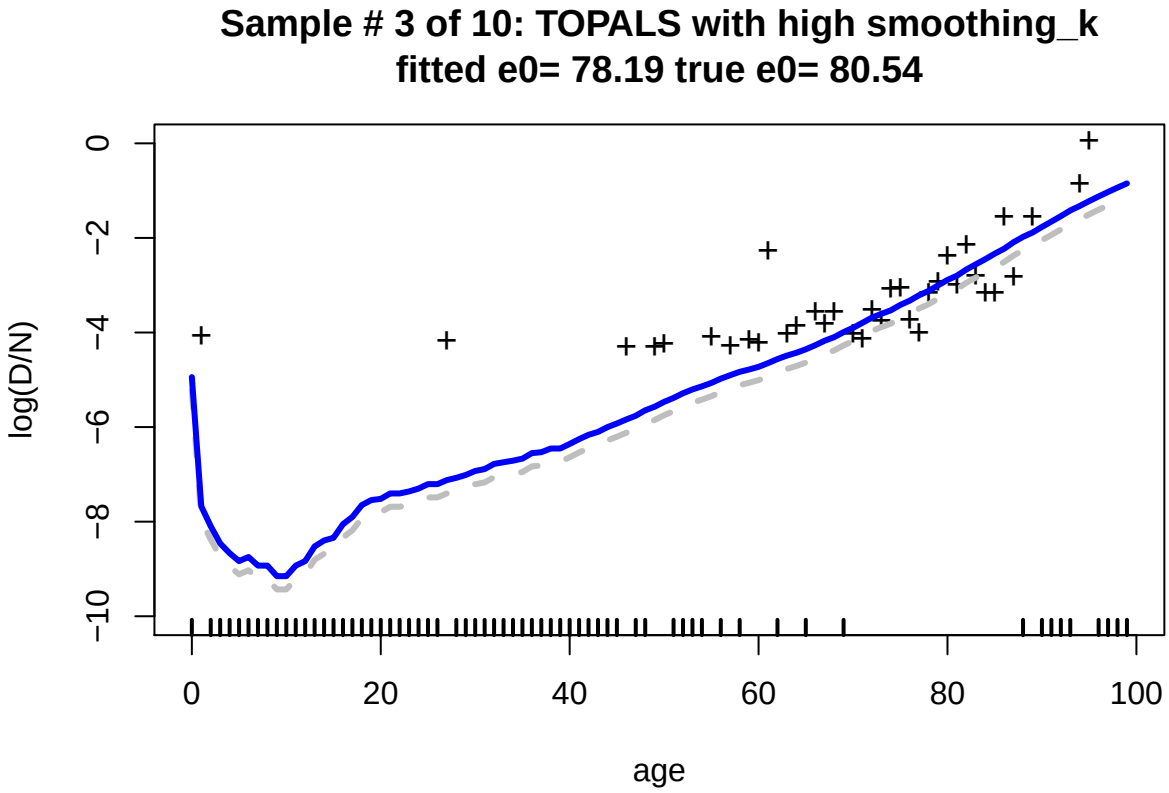

```
# EXAMPLE 2: Very high penalty, in which case
# TOPALS = indirect standardization
#        = up and down shifts of standard schedule
```

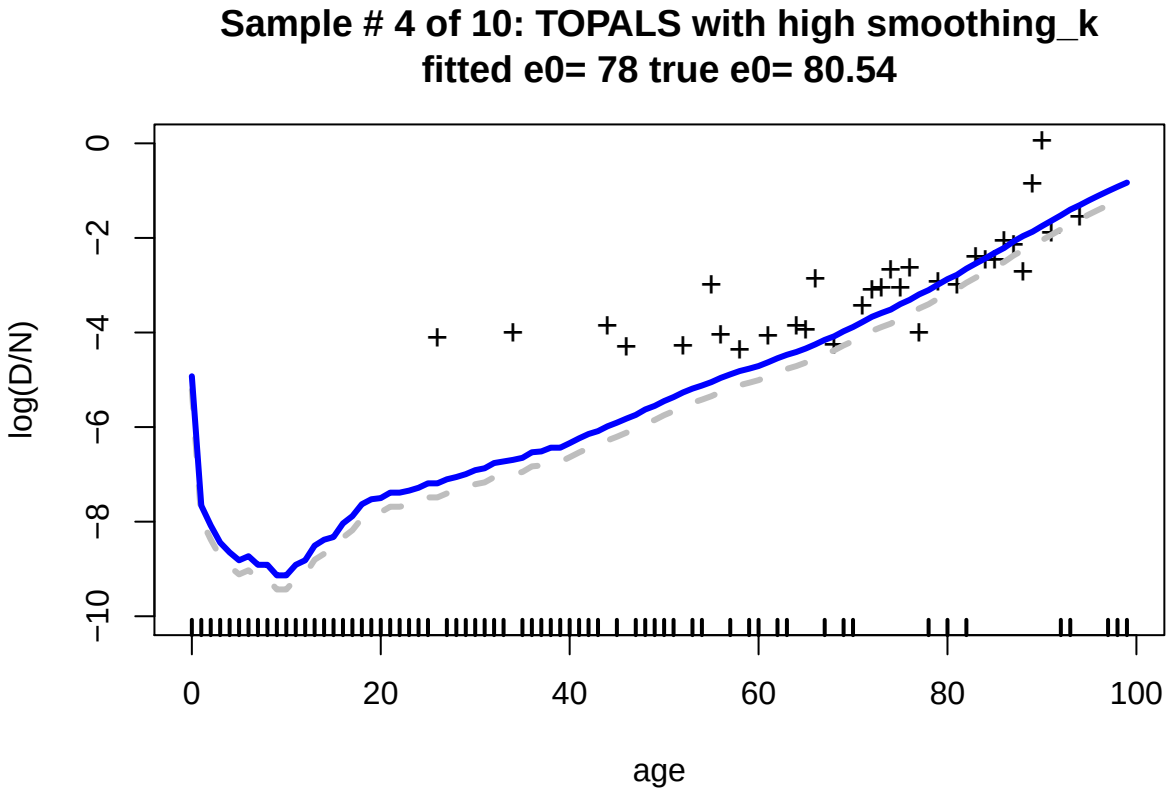
```
nsim = 10

B  = bs( 0:99, knots=c(0,1,10,20,40,70), degree=1 )

true_e0 = round( e0(log(mu)), 2)

for (i in 1:nsim) {
  D = rpois(100, N*mu)    # random deaths

  fitted_alpha = TOPALS_fit( N, D, this.std, smoothing_k = 10000)  #<<<<<<

  fitted_logmx = this.std + B %*% fitted_alpha

  this.e0    = e0(fitted_logmx)
  this.title = paste0('Sample # ',i, ' of ',nsim,': TOPALS with high smoothing_k\nfitted e0= ', round(tl
                      ' true e0= ', true_e0)

  plot( age, log(D/N), ylim=c(-10,0), pch='+', cex=1.2, main=this.title)
  rug( age[D==0], lwd=2)
  lines(age, this.std, type='l', lty=2, col='grey', lwd=3)
  lines(age, fitted_logmx, lty=1, col='blue', lwd=3)
```

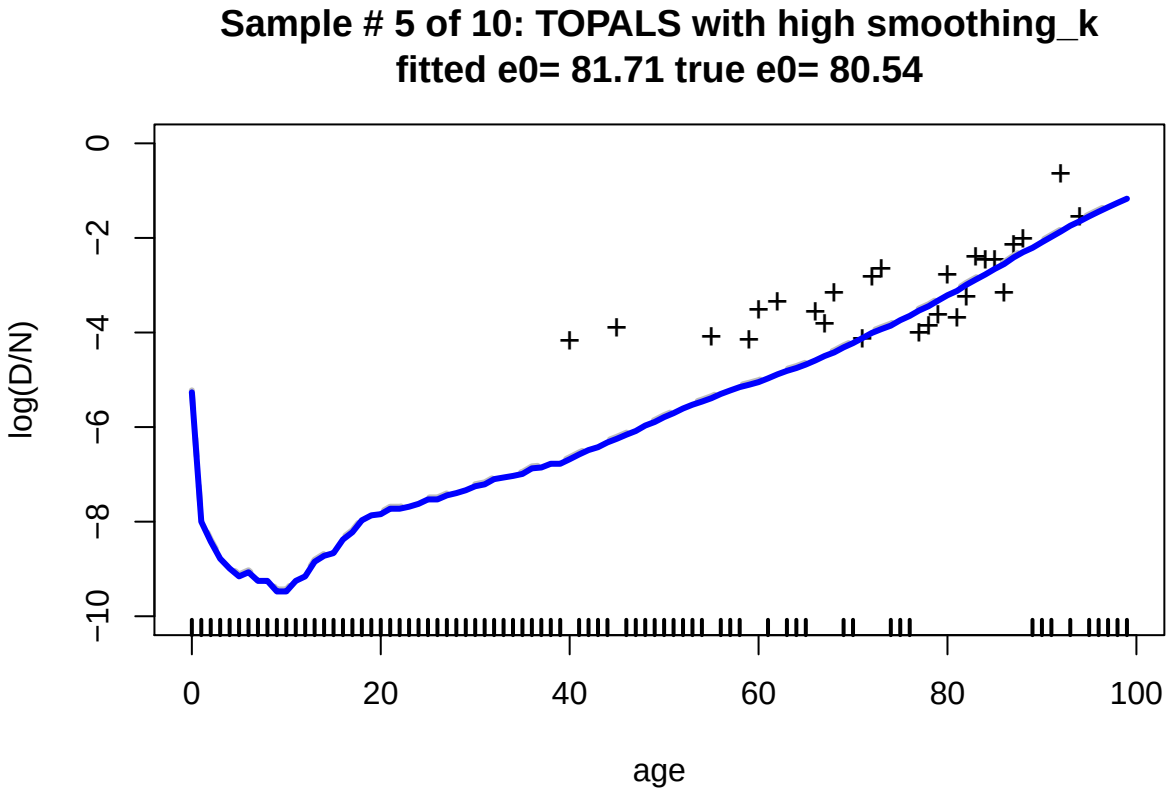
```
}
```

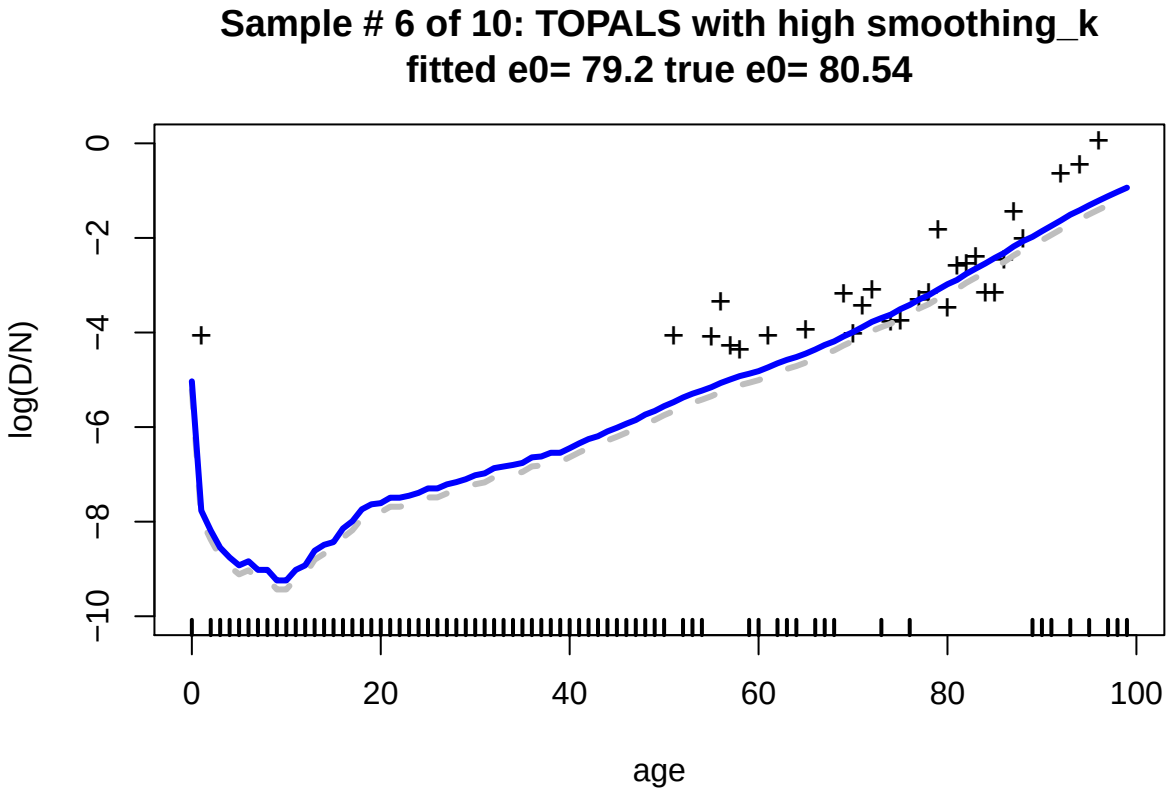

Sample # 1 of 10: TOPALS with high smoothing_k
fitted e0= 77.81 true e0= 80.54
log(D/N)
0
−2
−4
−6
−8
−10
0
20
40
60
80
100
age

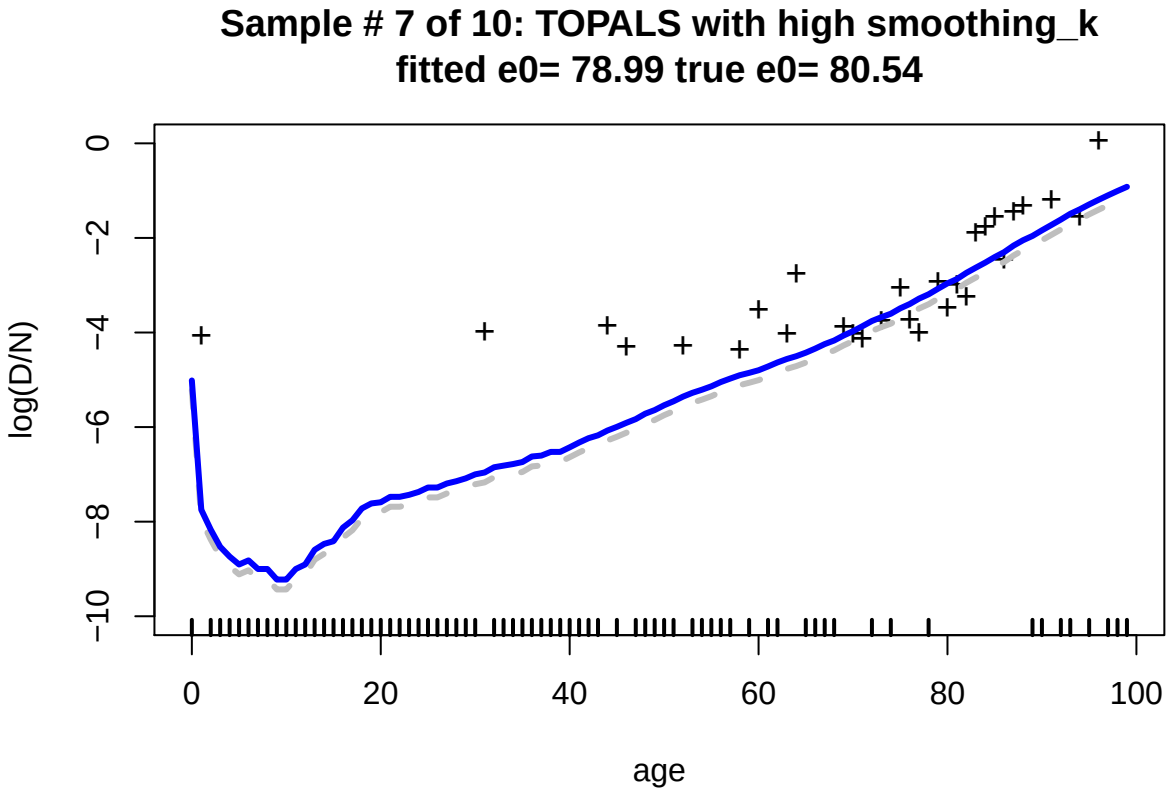

Sample # 2 of 10: TOPALS with high smoothing_k
fitted e0= 78.79 true e0= 80.54
log(D/N)
0
−2
−4
−6
−8
−10
0
20
40
60
80
100
age

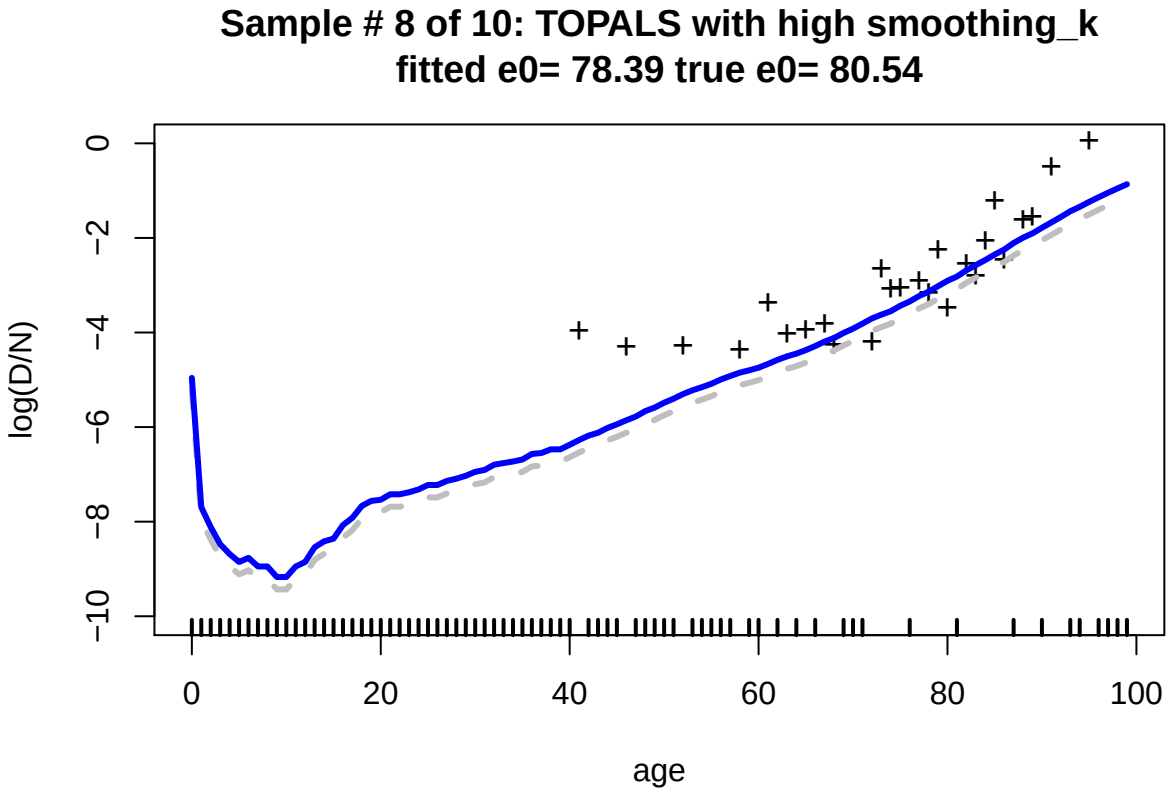

Sample # 3 of 10: TOPALS with high smoothing_k
fitted e0= 78.19 true e0= 80.54
log(D/N)
0
−2
−4
−6
−8
−10
0
20
40
60
80
100
age

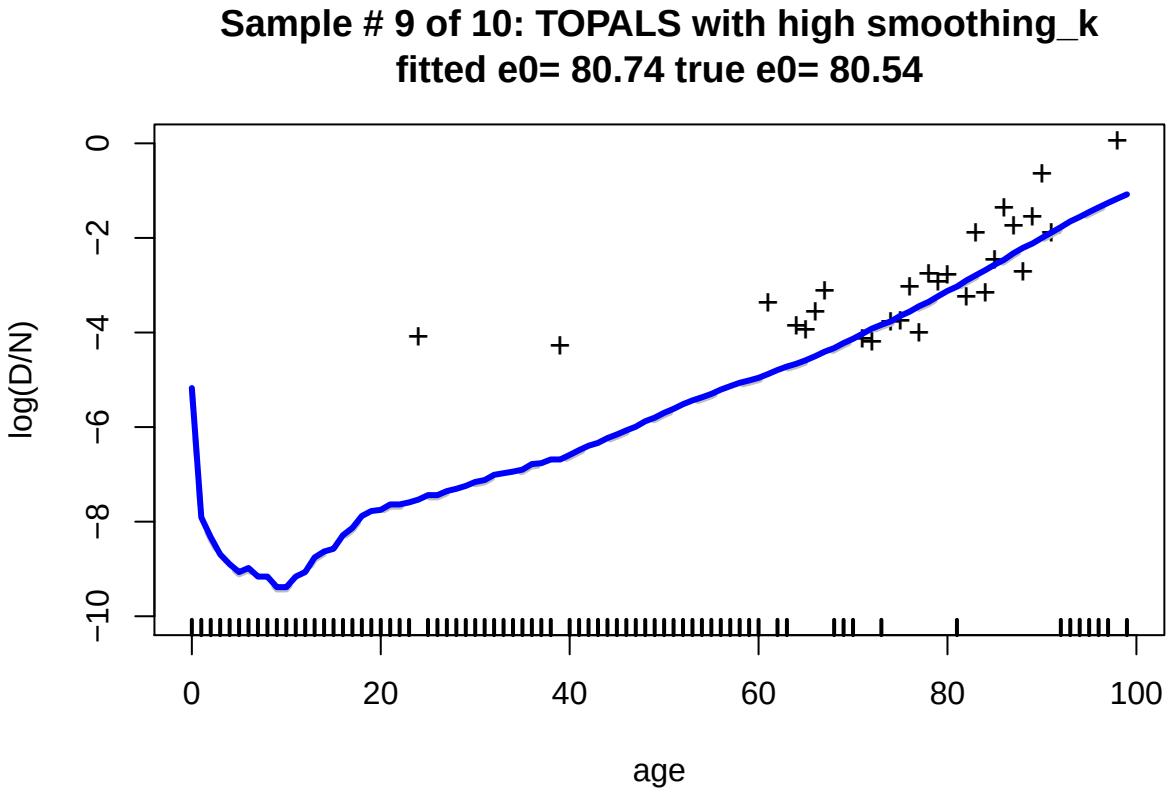

Sample # 4 of 10: TOPALS with high smoothing_k
fitted e0= 78 true e0= 80.54
log(D/N)
0
−2
−4
−6
−8
−10
0
20
40
60
80
100
age

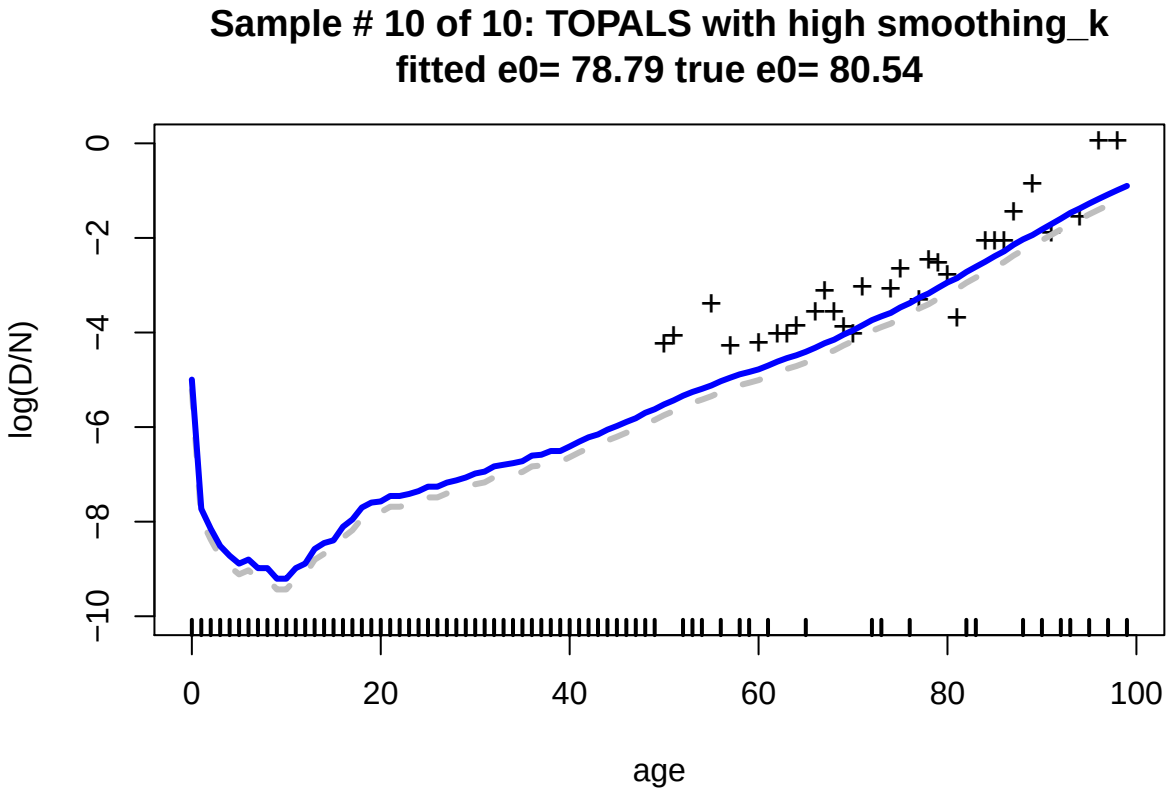

Sample # 5 of 10: TOPALS with high smoothing_k
fitted e0= 81.71 true e0= 80.54
log(D/N)
0
−2
−4
−6
−8
−10
0
20
40
60
80
100
age


Sample # 6 of 10: TOPALS with high smoothing_k
fitted e0= 79.2 true e0= 80.54
log(D/N)
0
−2
−4
−6
−8
−10
0
20
40
60
80
100
age


Sample # 7 of 10: TOPALS with high smoothing_k
fitted e0= 78.99 true e0= 80.54
log(D/N)
0
−2
−4
−6
−8
−10
0
20
40
60
80
100
age


Sample # 8 of 10: TOPALS with high smoothing_k
fitted e0= 78.39 true e0= 80.54
log(D/N)
0
−2
−4
−6
−8
−10
0
20
40
60
80
100
age


Sample # 9 of 10: TOPALS with high smoothing_k
fitted e0= 80.74 true e0= 80.54
log(D/N)
0
−2
−4
−6
−8
−10
0
20
40
60
80
100
age

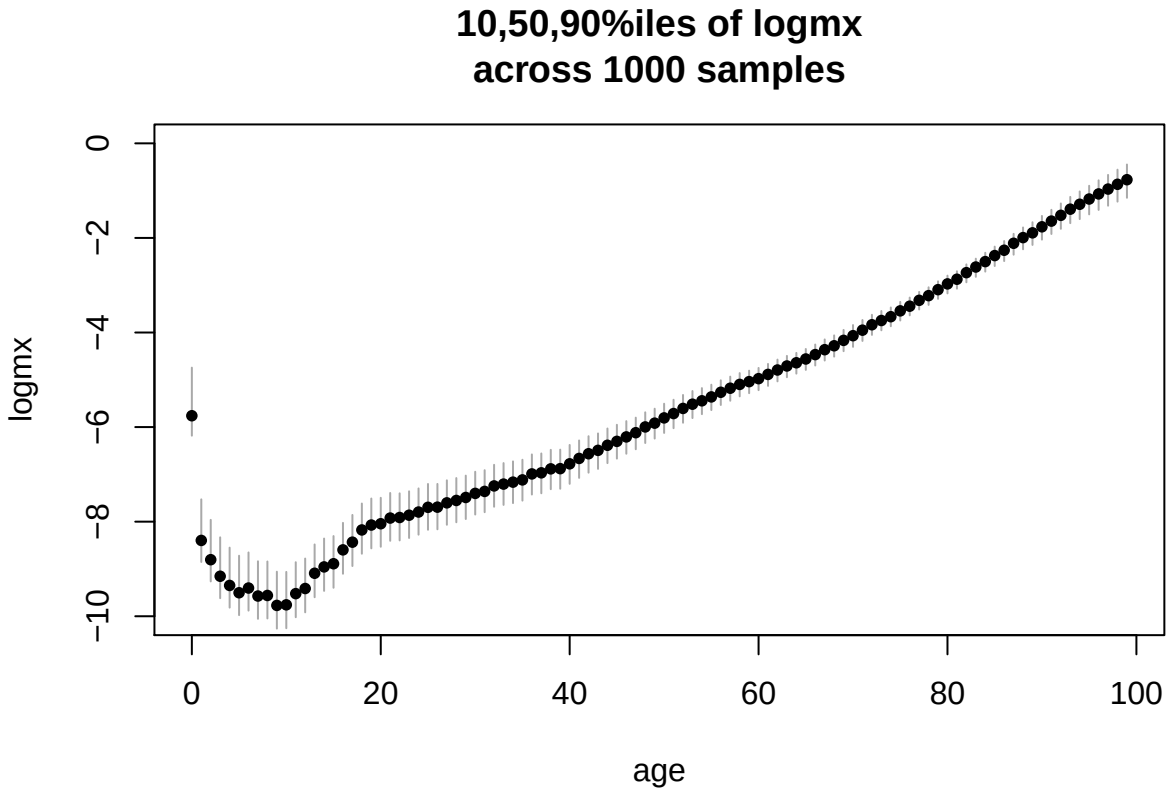



```
# EXAMPLE 3: applying TOPALS_fit to a large number of
#  datasets in one command
```

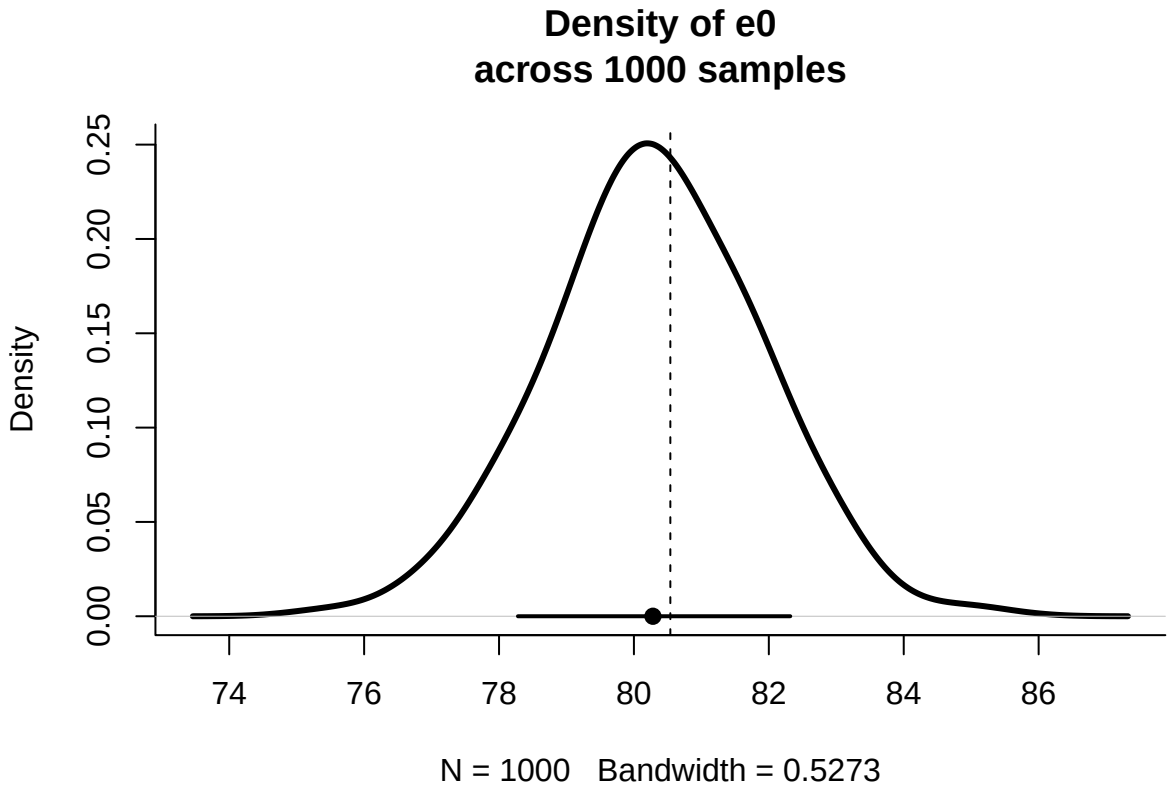
```
npop = 1000   # number of populations

# each col of D is a sample of deaths over ages 0..99
D = matrix( rpois( npop*100, N*mu), nrow=100)

# estimate TOPALS parameters for all npop populations in one command
# on a std desktop PC (circa 2014) this takes <1 sec for 1000 fits
system.time ({a = sapply( 1:ncol(D), function(i) TOPALS_fit(N, D[,i], std=this.std))})
```

```
##    user  system elapsed
##    0.90    0.01    0.92
```

```
L = this.std + B %*% a   # 100 x npop matrix of fitted logmx schedules

Lquant = t( apply(L, 1, quantile, c(.10,.50,.90)) )  # 100 x 3

plot(age, Lquant[,'50%'], type='n', ylim=c(-10,0),
     xlab='age',ylab='logmx',
     main=paste('10,50,90%iles of logmx\nacross',npop,'samples'))
segments(age, Lquant[,'10%'], age, Lquant[,'90%'], col='darkgrey' )
points(age, Lquant[,'50%'], pch=16, cex=.80)
```



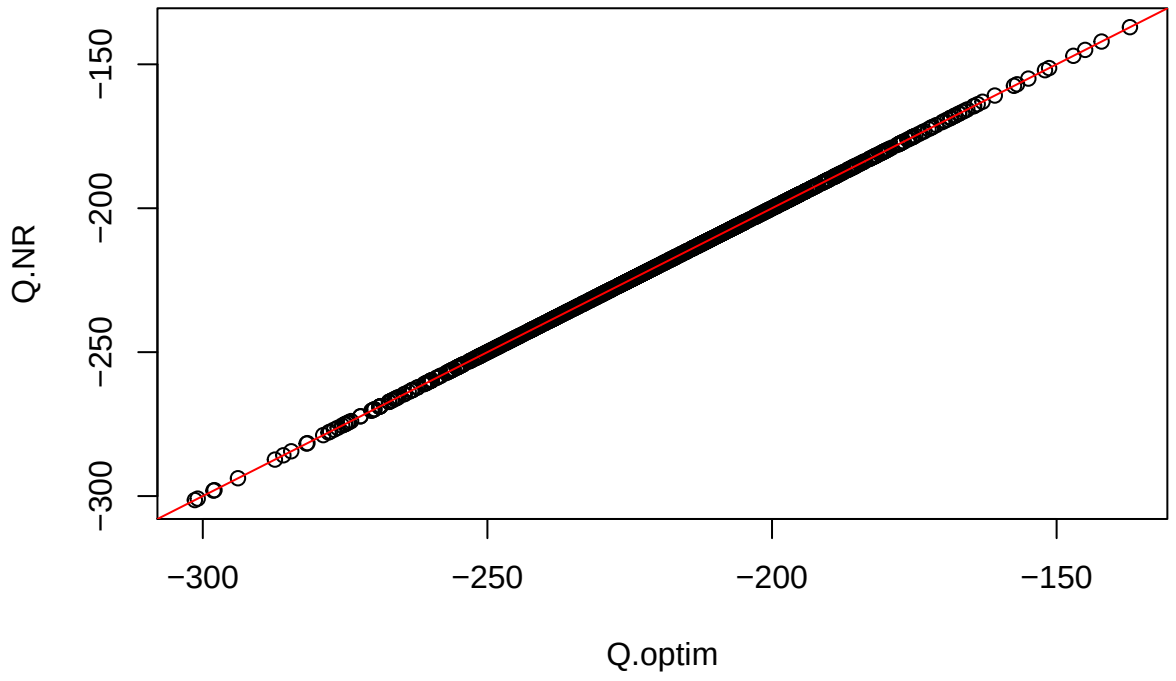

```
## estimated life expectancies across samples
esim = apply(L, 2, e0)
plot(density(esim, adjust=1.5), main=paste('Density of e0\nacross',npop,'samples'),
     lwd=3, bty='l')
equant = quantile(esim, c(.10,.50,.90))
segments( equant['10%'], 0, equant['90%'], 0, lwd=2)
points( equant['50%'], 0, pch=16, cex=1.2)
abline(v= true_e0, lty=2)
```



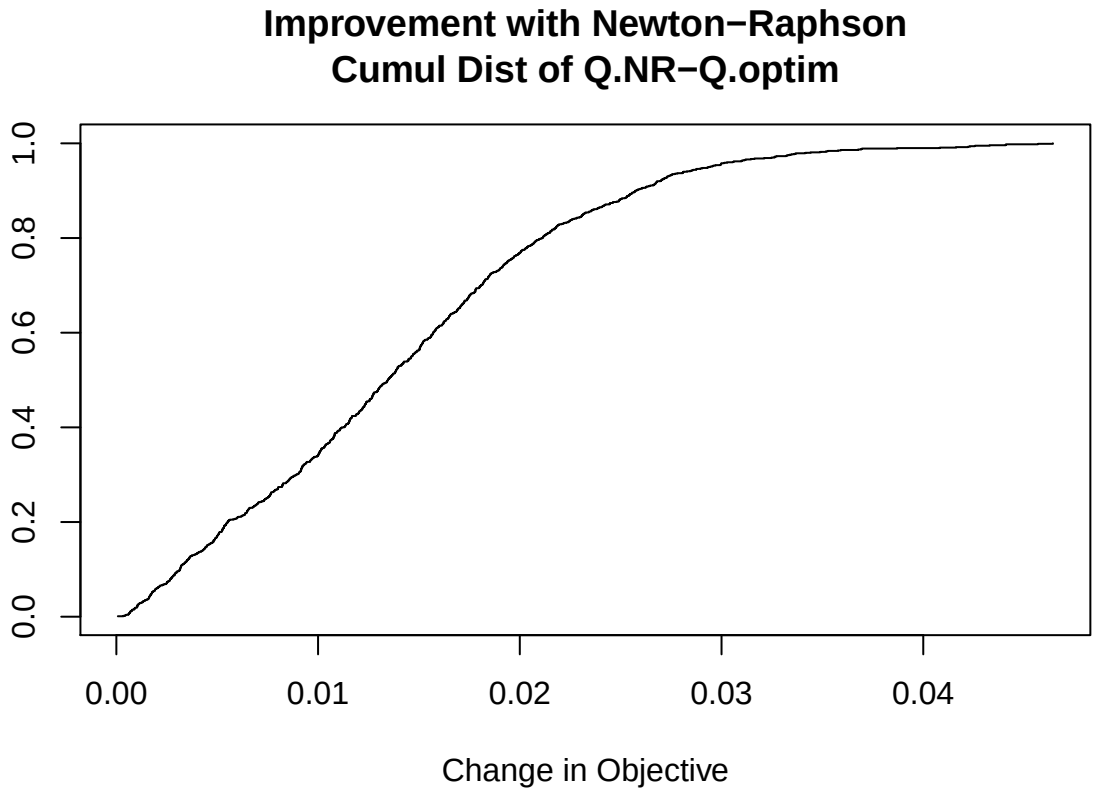

```
## compare to optim() approach, which seems to take about 20-30 times longer
## and be slightly less accurate

# control params for nonlinear fitting
this.control = list(maxit=1000,fnscale= -1, parscale=rep(.01,ncol(B)))

alpha0 = rep(0, 7)  # initial offsets (all 0 means start at standard schedule)

Q = function(alpha,N,D,std, smoothing_k=1) {
  lambda.hat = as.numeric( std + B %*% alpha)
  penalty    = smoothing_k * sum( diff(alpha)^2 )
  return( sum(D * lambda.hat - N * exp(lambda.hat)) - penalty)
}

system.time( {
  a.optim =
      sapply(1:ncol(D), function(j) {
      optim( alpha0, Q,
             D=D[,j], N=N,
             std=this.std,
             method='BFGS', control=this.control)$par }) }
)
```

```
##    user  system elapsed
##   20.67    0.00   20.71
```

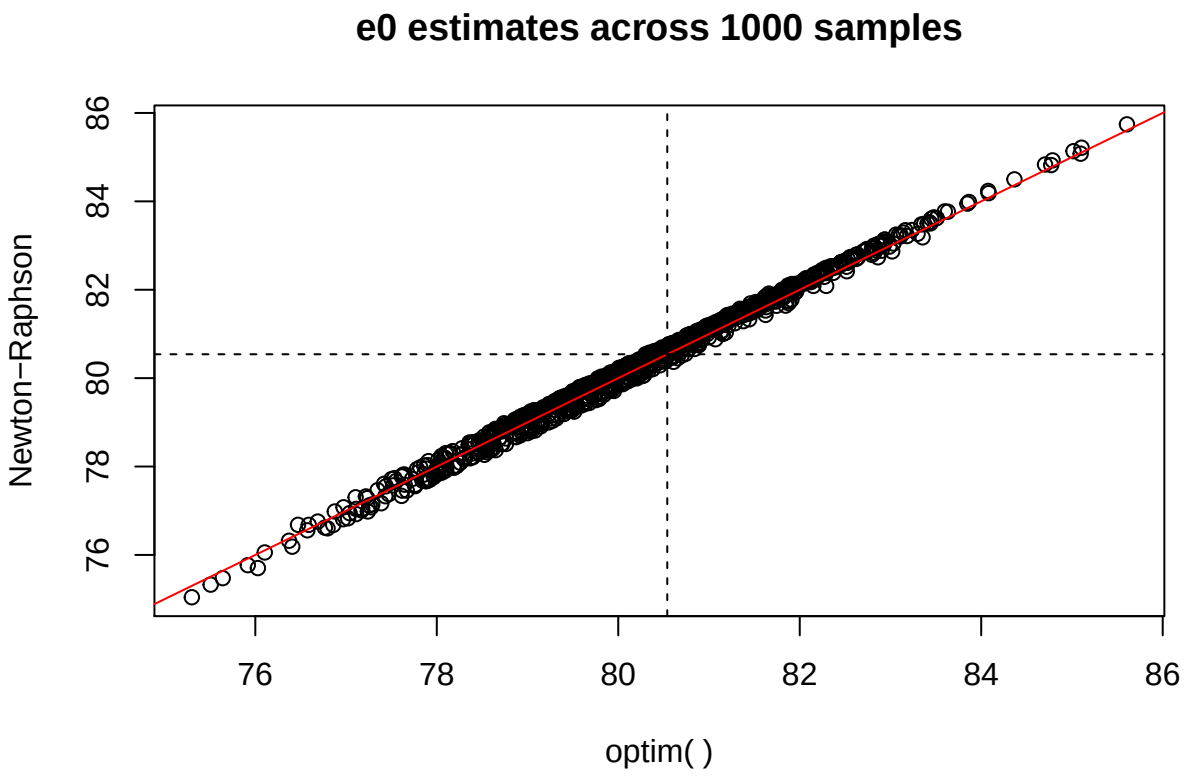
```
Q.optim = sapply(1:ncol(D),  function(j)  Q(a.optim[,j], N=N,D=D[,j],std=this.std) )
Q.NR    = sapply(1:ncol(D),  function(j)  Q(a[,j],       N=N,D=D[,j],std=this.std) )

plot(Q.optim, Q.NR)
abline(0,1,col=2)
```



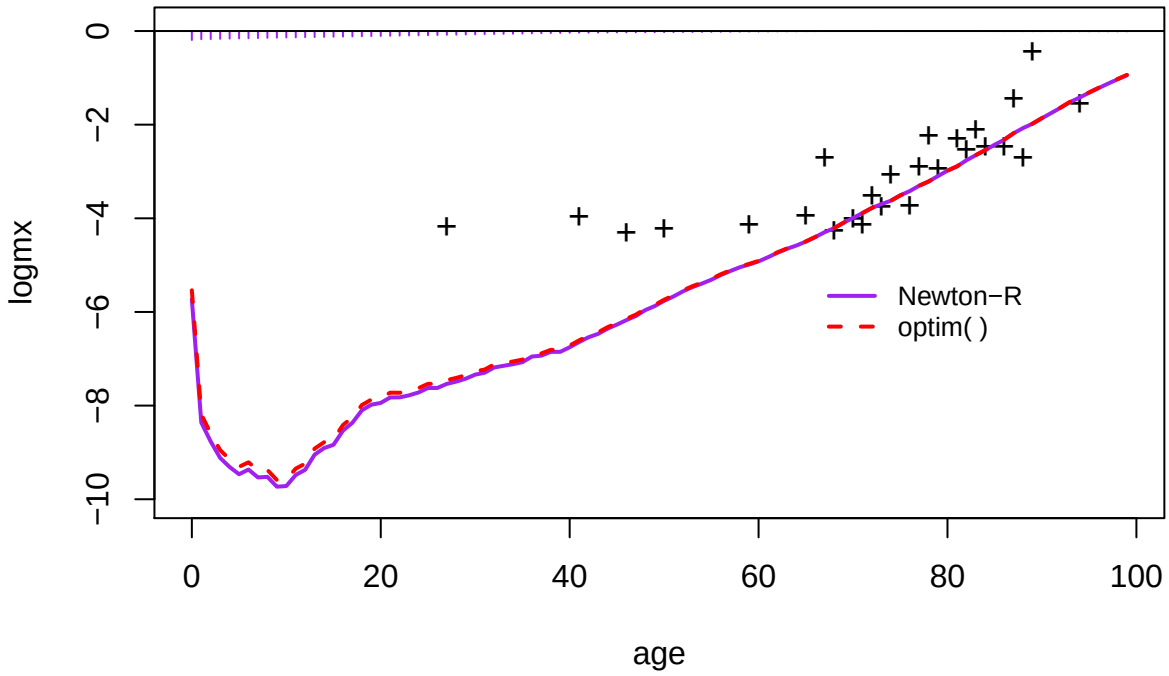

```
plot( sort( Q.NR - Q.optim), seq(Q.NR)/length(Q.NR), type='s',
      xlab='Change in Objective',ylab='',
      main='Improvement with Newton-Raphson\nCumul Dist of Q.NR-Q.optim')
```

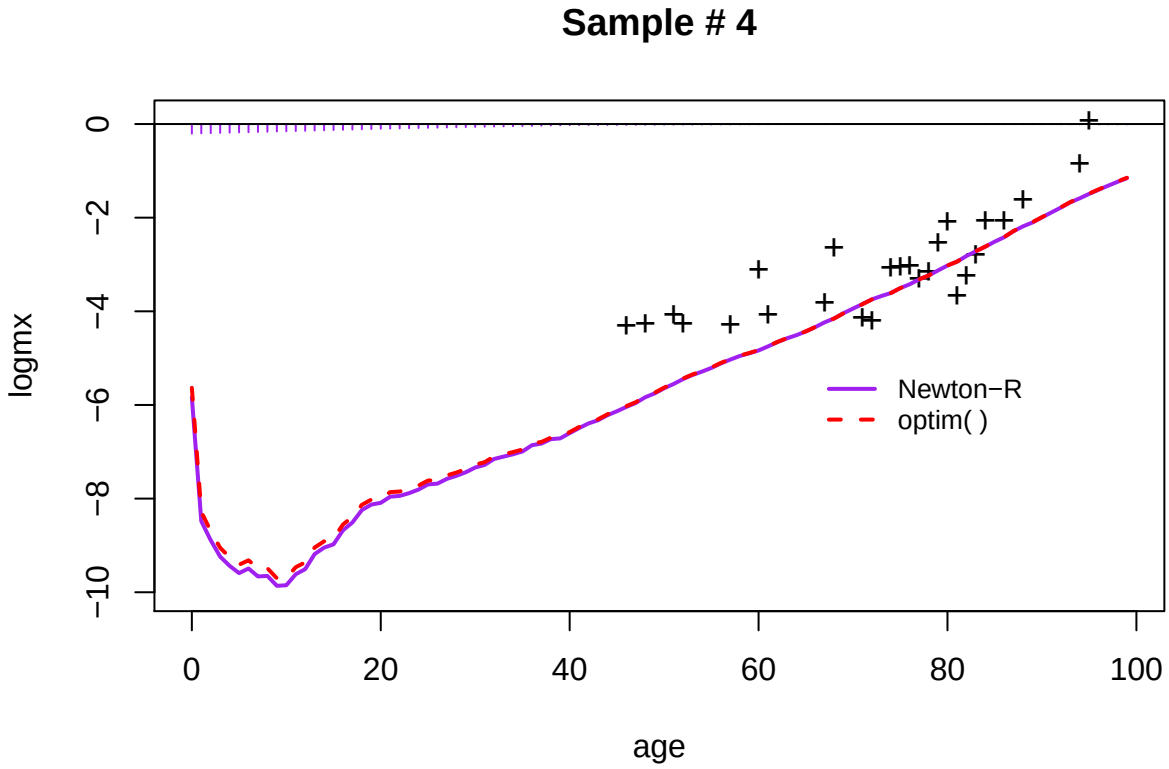


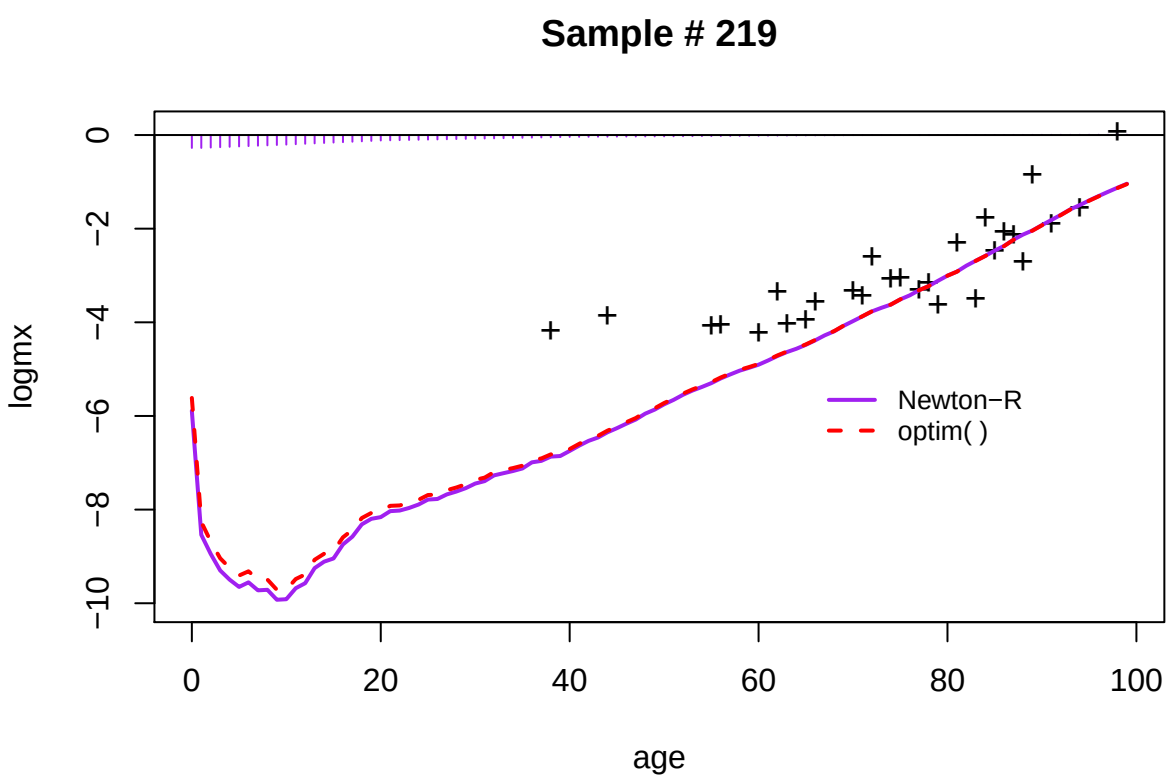

```
L.optim = this.std + B %*% a.optim

esim.optim = apply(L.optim, 2, e0)

plot(esim.optim, esim, ylab='Newton-Raphson', xlab='optim( )',
     main='e0 estimates across 1000 samples')
abline(0,1,col=2)
abline(h=true_e0, v=true_e0, lty=2)
```

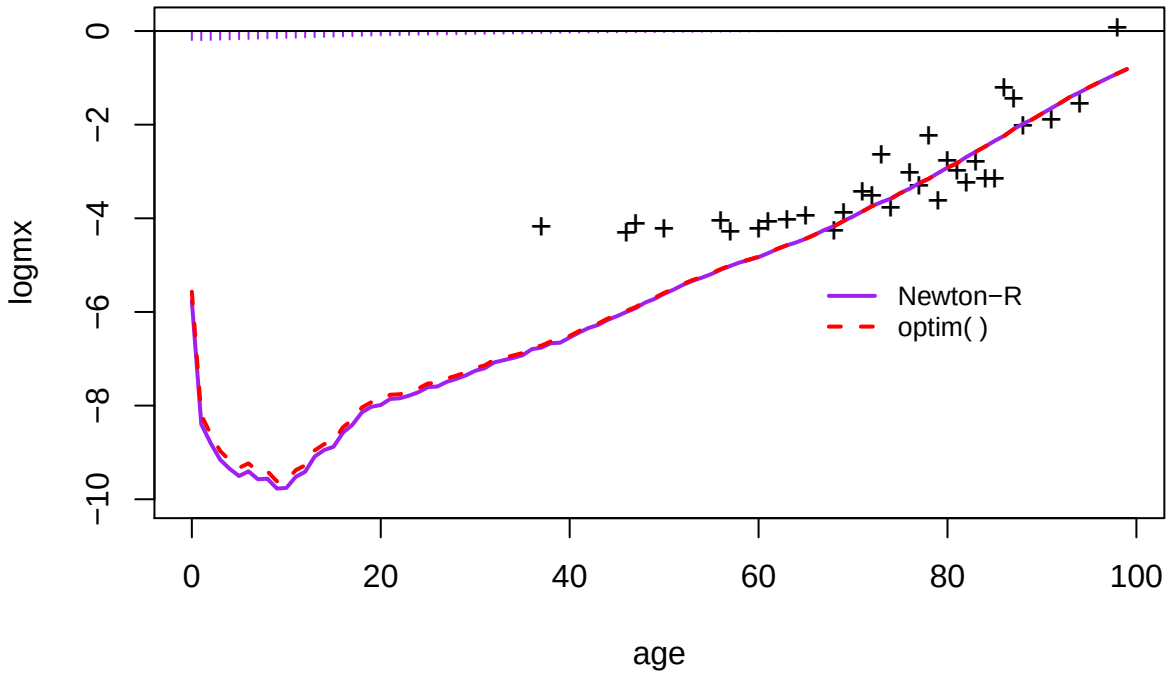


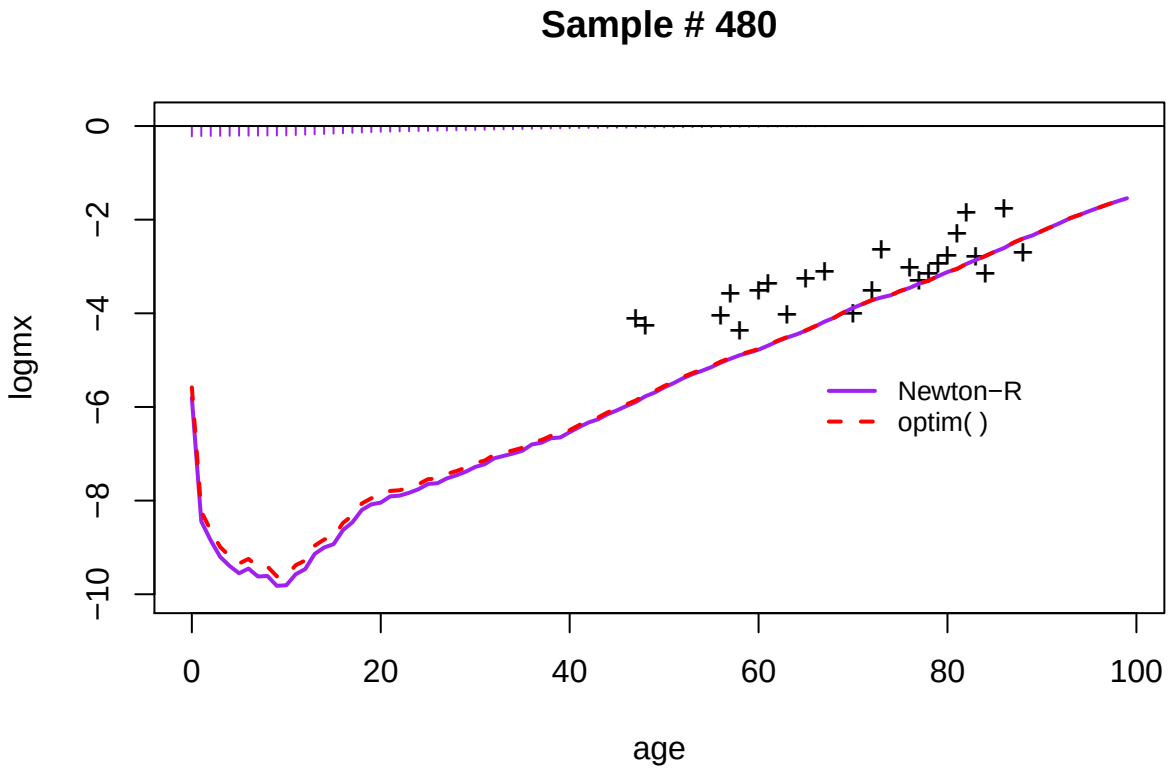

```
# a few examples

sel = sample(ncol(D), 5)
for (j in sel) {
  plot( age, log(D[,j]/N), pch='+', cex=1.2, main=paste('Sample #',j),
        ylim=c(-10,.10), ylab='logmx')
  lines(age, L[,j], col='purple', lwd=2)
  lines(age, L.optim[,j], col='red', lwd=2, lty=2)
  segments(age,0, age, L[,j]-L.optim[,j], col='purple')
  abline(h=0)
  legend( 65,-5, c('Newton-R','optim( )'), lwd=2,lty=c(1,2),
          col=c('purple','red'), bty='n', cex=.80)
}
```


Sample # 318
logmx
0
−2
−4
−6
−8
−10
0
20
40
60
80
100
age
Newton−R
optim( )


Sample # 4
logmx
0
−2
−4
−6
−8
−10
0
20
40
60
80
100
age
Newton−R
optim( )


Sample # 219
logmx
0
−2
−4
−6
−8
−10
0
20
40
60
80
100
age
Newton−R
optim( )


Sample # 32
logmx
0
−2
−4
−6
−8
−10
0
20
40
60
80
100
age
Newton−R
optim( )


Sample # 480
logmx
0
−2
−4
−6
−8
−10
0
20
40
60
80
100
age
Newton−R
optim( )