```
volume<-c(0.032,0.026,0.021,0.018,0.016,0.011,0.005)
n<-c(29,30,28,30,29,25,30)
p<-c(7,4,3,4,2,2,2)
mu_i<- -(1/volume)*log((n-p)/n)
data=data.frame(volume,n,p,mu_i)
LnL<-function(guess,data)
{tmp=volume*(guess-mu_i)*(n-p)-p*log((1-exp(-guess*volume))/(1-exp(-
mu_i*volume)))
    sum(tmp)}
LnL(1,data)
best<-optim(.5,LnL,gr=NULL,method='BFGS',control=list(trace=12,REPORT=1))
show(best)
```

Figure 6.17 R functions for solution of Example 6.9.

```
initial  value 41.888275
iter   2 value 2.241312
iter   3 value 2.086770
iter   4 value 1.200017
iter   5 value 1.055462
iter   6 value 1.023437
iter   7 value 1.023008
iter   8 value 1.023007
iter   9 value 1.023006
iter   9 value 1.023006
iter   9 value 1.023006
final  value 1.023006
converged
$par
[1] 6.948856

$value
[1] 1.023006

$counts
function gradient
      14        9

$convergence
[1] 0
```

Figure 6.18 Output from execution of R solution for Example 6.9.

then used to seek the minimum. The option BFGS specifies a variant of optimization engine used for the problem.

The result of execution of the program in Figure 6.17 is shown in Figure 6.18. In this case, the optimization finds the same best solution in nine iterations.

```
library("deSolve")
genlogist<-function(t,N,params){
  #generalized logistic with zero, one, two and three parameters (other
than k and K)
   k< -params[1]
   K< -params[2]
   theta1<-1
   theta2<-1
   theta3<-1
   model<-params[6]
   if (model>0) theta1<-params(16)
   if (model>1) theta2<-params[4]
   if (model>2) theta3<-params[5]
   dydt<-k*(N^theta2)*(1-(N/K)^theta1)^theta3
   list(dydt)
}
model<-3  #should be equal to the number of thetas <> 1
10->k
1e6->K
.7->theta1
.3->theta2
.5->theta3
params<-c(k,K,theta1,theta2,theta3,model)
N0< -1e1
times<-seq(0,8,by=0.1)
y< -ode(N0,times,genlogist,params,method='adams')
plot(y,log="y")
print(y)
```

Figure 7.10 R code to compute generalized logistic growth equation.

$$\mathrm{AIC} = n\ln\left(\frac{\mathrm{ESS}}{n}\right) + 2p$$
$$\mathrm{AIC_c} = n\ln\left(\frac{\mathrm{ESS}}{n}\right) + 2p + \frac{2p(p+1)}{n-p-1} \qquad (7.25)$$

where ESS is the residual sum of squares from a regression model that contains $p$ adjustable parameters and $n$ is the number of observations. In Table 7.3, these summary measures are reported. The three-theta version of the logistic, Equation (7.24), has the lowest value of AIC$_c$ and therefore is the preferred model among those in the logistic family.

The Gompertz model, Equation (7.20), was also fit to these data by ordinary least squares, to find the parameter estimates as well as metrics of adequacy shown in Table 7.4. A plot of the fit of the Gompertz and the three-theta logistic model to the data is shown in Figure 7.12.

Since the AIC$_c$ for the Gompertz model is substantially less than any for the logistic family, from a model adequacy viewpoint, Gompertz would be preferred.

```
#fit listeria data - 1.5o skim milk
#Xanthiakos, K., D. Simos, A. S. Angelidis, G. J. Nychas and K. Koutsoumanis
(2006).
#"Dynamic modeling of Listeria monocytogenes growth in pasteurized milk."
Journal of Applied Microbiology 100(6): 1289-1298.
time<-
c(0,143.39622,260.37735,383.01886,500,598.11322,696.22644,792.45282,884.90564,10
52.8302,1205.6604,1371.69812)
#time in hours
N<-
c(5204.991205,5972.002763,4536.49044,4859.258466,12719.79172,12719.79172,31084.2
2186,84211.22791,300337.8033,4859258.466,25292397.58,61807332.42)
#N CFU/mL
A<-data.frame(time=time,N=N,lnN=log(N))
attach(A)

source("genlogist.R")
#-----------------------------------------------
ObjFunc<-function(paramsin){
  #computes ESS

  if (modell==0){
    paramsin["theta1"]<-1
    paramsin["theta2"]<-1
    paramsin["theta3"]<-1}
  if (modell==1){
    paramsin["theta2"]<-1
    paramsin["theta3"]<-1}
  if (modell==2){
    paramsin["theta3"]<-1}
  params<-
c(exp(paramsin["lnk"]),exp(paramsin["lnK"]),paramsin["theta1"],paramsin["theta2"
],paramsin["theta3"],modell)

  y<-ode(N0,timepoints,genlogist,params,method="daspk")
  pred<-y[,"1"]
  ESS<-log(obsN)-log(pred)

  A<-c(ESS^2)
  ESS<- sum(A)
  plot(y,log="y")
  points(timepoints,obsN)
  working<<-data.frame(t=timepoints,obsN=obsN,predN=pred)
  return(ESS)
}
#-----------------------------------------------
timepoints<<-A$time
N0<<-5000
obsN<<-A$N
modell<<-3
params<-c()
params["lnk"]<- -4.69
params["lnK"]<-33.64
params["theta1"]<-0.0615
params["theta2"]<-3.69
params["theta3"]<-103.534
best<-optim(params,ObjFunc,gr=NULL,method="Nelder-
Mead",control=list(trace=6,reltol=1e-10,maxit=5000))
#best<- genoud(ObjFunc,
nvars=6,pop.size=20,starting.values=params,optim.method="Nelder-Mead")
print(best)
print(working)
```

Figure 7.11 R listing for fitting *Listeria* data.

**TABLE 7.3 Summary of Fits of Alternative Logistic Models to *Listeria* Data**

| Equation | $k$ (1/h) | $K$ (#/ml) | $\theta_1$ | $\theta_2$ | $\theta_3$ | ESS | $p$ | AIC | AIC$_c$ |
|---|---|---|---|---|---|---|---|---|---|
| 7.18 | 5.37E−03 | 1.80E+11 | | | | 31.280 | 2 | 63.5514 | 16.8302 |
| 7.21 | 5.37E−03 | 1.16E+09 | 2.5735 | | | 31.280 | 3 | 65.5513 | 20.4968 |
| 7.23 | 9.21E−05 | 3.88E+07 | 0.0072 | 1.6705 | | 2.577 | 4 | 37.5950 | −4.7452 |
| 7.24 | 9.20E−03 | 4.07E+14 | 0.0615 | 3.6896 | 103.6234 | 1.173 | 5 | 30.1495 | −7.9050 |

**8.5.** Apply the Cochran–Armitage test of trend to the *Cryptosporidium* data in Table 8.6.

**8.6.** Fit the *S. anatum* data (Table 8.12) to the exponential model, and determine the statistical significance of the improvement in fit for the beta-Poisson model relative to the exponential model. Compute the AIC statistic. ■

## APPENDIX

The best-fit dose–response parameters can be determined by unconstrained optimization. In this Appendix, we illustrate the use of *R* to compute the best-fit parameters of a dose–response relationship.

We will consider the rotavirus data set presented in Table 8.4. We wish to compute best-fit values of $\alpha$ and $N_{50}$ for the beta-Poisson fit. The fundamental process requires finding the values of $\alpha$ and $N_{50}$ that minimize the deviance of Equation (8.33).

The listing for solving this problem in *R* is given in Figure 8.14. After storing the data, the dose–response function is given in pred. beta-Poisson, and deviance is given in the function deviance. The optimization is done using the built-in function optim, using the Nelder–Mead simplex algorithm [99].

The output of the optimization is shown in Figure 8.15. The best-fitting parameter values are found to be $\alpha = 0.2649917$ and $N_{50} = 5.5962875$. The minimum deviance ($Y$) is given as 6.81529. These are the values noted in Example 8.1.

```
dose<-c(90000,9000,900,90,9,0.9,0.09,0.009)
total<-c(3, 7,8,9,11,7,7,7)
positives<-c(3,5,7,8,8,1,0,0)
dataframe=data.frame(dose=dose,total=total,positives=positives)
#---------------------------------------------------------------
#Function to Return Predicted Value Given Parameters
pred.betaPoisson<-function(alpha,N50,data){
  f<-1-(1+data$dose*(2^(1/alpha)-1)/N50)^-alpha
  return(f)
}
#---------------------------------------------------------------
#Function to Return Deviance
deviance<-function(params,data){
  alpha=params[1]
  N50=params[2]
  fpred<-pred.betaPoisson(alpha,N50,data)
  fobs<-data$positives/data$total
  Y1<-data$positives*log(fpred/(fobs+1e-15))  #we add small number to
prevent taking log(0)
  Y2<-(data$total-data$positives)*log((1-fpred)/(1-fobs+1e-15))
  Y<--2*(sum(Y1)+sum(Y2))
  return(Y)
}
#---------------------------------------------------------------
best<-optim(c(0.5,10),deviance,gr=NULL,dataframe,method="Nelder-
Mead",control=list(trace=10))
print(best)
```

Figure 8.14 Program listing in *R* to compute best-fit parameters.

***Bootstrap*** Typical exposure data, such as those discussed in Chapter 6, consist of repeated measurements of a system—possibly over time or at different locations. If there is presumed to be a systematic variation with time (e.g., seasonality) or with location ("hot spots"), somewhat more complex bootstrap structuring must be used.

Where it can be regarded that systematic temporal or spatial variation can be neglected (or is minor relative to the intrinsic variability between samples), then the set of "*N*" samples can be subject to be bootstrap by constructing individual bootstrap replicates each consisting of "*N*" observations drawn from the original data *with replacement*. From each of the individual bootstrap replicates, the statistic of interest (e.g., sample mean) is computed, and the ensemble of values of that statistic over all of the bootstrap replicates is an estimator of the uncertainty distribution of the statistic.

This is illustrated by reference to the data in Table 6.5, where weekly *Cryptosporidium* measurements were taken in a water source over the course of a year. Therefore, each bootstrap replicate consists of 52 random draws from the data (note each observation consists in this case of a number of oocysts found and the volume of water that was examined). This computation is especially straightforward in R, which has a bootstrap capability in one of the distributed packages.

The computation is shown in Figure 9.5. This will produce the cdf of the bootstrap estimate of the density of oocysts (defined by the sum of oocyst counts divided by the sum of volumes examined). The result of this computation (for 10,000 bootstrap replicates) is shown in Figure 9.6. The cdf in this figure could be used as an input into an overall risk characterization (or, more directly, the 10,000-individual bootstrapped values of density).

***Bayesian*** The Bayesian approach to analysis of exposure data is illustrated with reference to the count data (in 1 ml samples) given in Example 6.5. In that example, it was shown that the data were statistically different from what would be predicted

```
library(boot) #requires the bootstrap package
oocysts<-c(0,  0,0,0,0,0,0,0,0,0,0,0,0,
           0,0,0,0,0,0,0,0,0,0,0,0,0,
           0,0,0,0,0,0,0,1,1,1,1,1,1,
           1,1,1,1,1,1,2,2,2,2,3,3,3)
volume<-c(48,  51,52,54.9,55,55,55,57,59,59,85.2,
          100,100,100,100,100,100,100,100.4,
          100.4,100.6,100.7,101.7,102,102,102.2,
          102.2,103.3,185.4,189.3,189.3,190,191.4,
          18.4,74.1,99.9,100,100,100,100,100,101.1,
          101.3,183.5,193,95.8,223.7,223.7,227.1,
          89.9,98.4,100)
data<-data.frame(oocysts=oocysts,volume=volume)
poissonmean<-function(data,weights) {
  A<-sum(oocysts*weights)
  B<-sum(volume*weights)
  return(A/B)
}
bootstrappedpoissonmean<-boot(data,poissonmean,R=10000)
density<-bootstrappedpoissonmean$t
fig<-ecdf(density)
plot(fig,xlab="mean density #/L",ylab='cumulative<=',main='10,000 bootstrap
replicates')
```

Figure 9.5 R code for bootstrap analysis of mean density for data in Table 6.5.

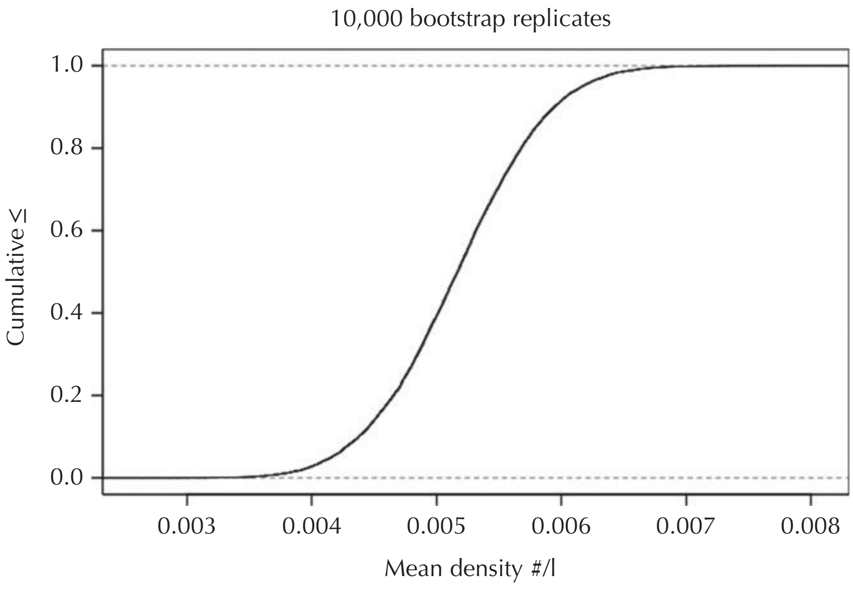

Figure 9.6 Bootstrap cdf for *Cryptosporidium* density estimate (Table 6.5).

from a Poisson distribution. We want to determine the uncertainty of negative binomial parameters that might be used to characterize these data.

The starting point is Equation (9.2). The likelihood function for the unknown parameters ($\mu$,$k$) given data "$x$" can be written as

$$p(x|\Theta) = \prod_i \frac{\Gamma(x_i + k)}{\Gamma(k)x_i!}\left(\frac{\mu}{k+\mu}\right)^{x_i}\left(\frac{k+\mu}{k}\right)^{-k} \tag{9.4}$$

Equation (9.4) arises from Equation (6.47) recognizing that the volume in this problem is constant ($V = 1$ ml for all samples).

To apply a Bayesian approach, we need a prior distribution, $p(\Theta)$. If we actually had prior information, we could use an "informative" prior. In this problem, we assume we have been given this data without prior knowledge, so a flat prior over a broad range of parameters is appropriate. So we will assume that the joint prior is independent, that is, the product of the marginal priors:

$$p(\mu,k) = p(\mu)p(k) \tag{9.5}$$

And the marginal priors are given by

$$\begin{aligned} p(\mu) &= 0 && \text{if } \mu < 1 \text{ or } \mu > 500 \\ p(\mu) &= \frac{1}{499} && \text{otherwise} \end{aligned} \tag{9.6}$$

and

$$\begin{aligned} p(k) &= 0 && \text{if} \quad k < 0.01 \text{ or } k > 20 \\ p(k) &= \frac{1}{19.99} && \text{otherwise} \end{aligned} \tag{9.7}$$

We now evaluate Equation (9.2) by first evaluating the denominator (which is a double integral due to the presence of two parameters). We can then evaluate the posterior distribution, $p(\Theta|x)$, at a series of grid points to show the bivariate distribution.

The computations are done in *R*, which has an available package, cubature, which performs the needed numerical integration. The computation is shown in Figure 9.7.

The resulting contours of the posterior distribution for the negative binomial parameters are shown in Figure 9.8. Note that the contours are "clipped" at the top since the prior distribution, Equation (9.7), for *k* disallows values greater than 20. This indicates that there is some influence of the prior distribution

```
require(cubature) #numerical integration package (needs to be installed)
require(lattice)  #for contour plotting
observations<<-c(27,30,60,60,70,70,74,80,81,82,84,84,93,98,98,101,105,110)#note global
assign

#-----------Prior Distribution of Parameters--------
prior<-function(mu,k){
  pmu<- ((mu>1)&(mu<500))/499
  pk<-((k>0.01)&(k<20))/19.99
  A<-pmu*pk
  return(A)
}
#-------------Neg Binomial Distribution-------------
NB<-function(mu,k,x){
  A<-gamma(x+k)/(gamma(k)*factorial(x))
  B<-(mu/(k+mu))^x
  C<-((k+mu)/k)^(-k)
  return(A*B*C)
}
#------------Likelihood Function--------------------
Likelihood<- function(mu,k,data){
  L<-NB(mu,k,data)
  Lik<-prod(L)
  return(Lik)
}

#-------------Integrand-----------------------------
# This is å product of the prior and the likelihood
Integrand<-function(y){
  mu<-y[1]
  k<-y[2]
  A<-Likelihood(mu,k,observations)  #note reference to global variable
  B<-prior(mu,k)
  return(A*B)
}
#=====================================================
# First we determine the denominator by integration

I<-adaptIntegrate(Integrand,c(1,.01),c(500,20),tol=1e-5)

#---------------------------------------------------
#Now evaluate the posterior over a grid
mu<-seq(from=60, to=102, by=1)
k<-seq(from=4, to=20, by=.1)
values<-expand.grid(mu=mu,k=k)
posterior<-vector(mode="numeric",length=dim(values)[1])
for (i in 1:dim(values)[1]){
  posterior[i]<-Integrand(c(values[i,1],values[i,2]))
  posterior[i]<-posterior[i]/I$integral
}

tableau<-(cbind(values,posterior))
contourplot(posterior~mu*k,data=tableau,cuts=12,xlim=c(62,97),ylim=c(4.0,20),label.style='al
ign',font=2,ps=17)
```

Figure 9.7 R code for determining the posterior distribution for the negative binomial distribution parameters fit to data in Example 6.5.

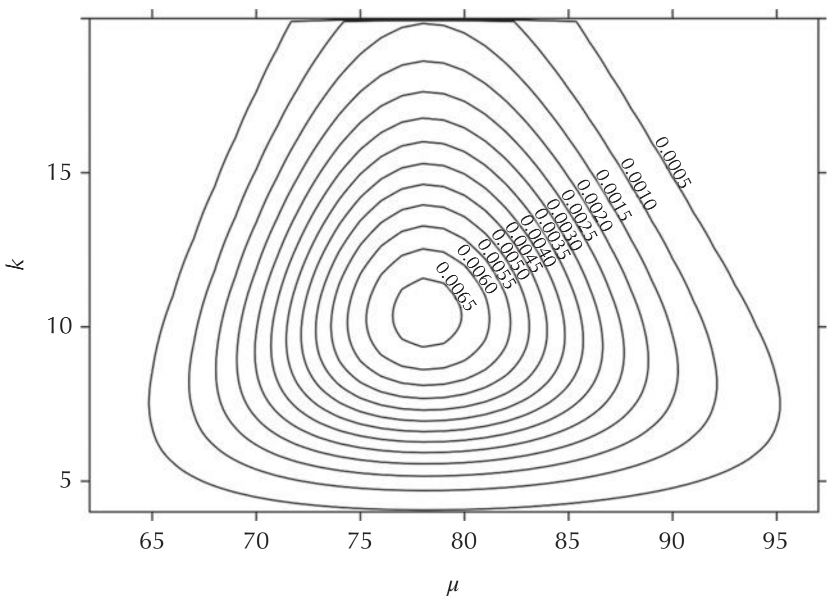

Figure 9.8 Contours (numbers indicate values of the density) of the posterior distribution of the negative binomial parameters.

on the final estimates.[2] While this is useful for graphically depicting the spread of parametric uncertainty, for a full risk assessment, it would be useful to draw random samples of the posterior.

To draw random samples from the posterior, we use a variant of the acceptance–rejection method noted previously—since this is a bivariate distribution, we draw random pairs ($\mu^*, k^*$) uniformly from the range of possible values (in this case, based on the priors, the ranges indicated in Eqs. (9.6) and (9.7)), as well as a random uniform number, $z^*$. Since the computation of the posterior indicated that no value of the density was in excess of 0.008 (Fig. 9.8), $z^*$ is chosen uniformly from the range <0, 0.008>. Then, if the value of the posterior evaluated at ($\mu^*, k^*$) is greater than $z^*$, the point ($\mu^*, k^*$) is accepted as one point sampled from the posterior. In Figure 9.9, the code snippet for generating such a sample (which is executed immediately subsequent to the code in Fig. 9.7) is presented.

A set of 2000 samples from the posterior distribution is given in Figure 9.10. To get 2,000 accepted points, over 150,000 random trials needed to be conducted. While this was done sufficiently rapidly (seconds of execution time on a desktop computer), it is inefficient. To achieve greater efficiency, there are particular "tricks" of drawing random numbers from a distribution that more closely envelopes the posterior;

[2] As noted earlier, different prior distributions may result in different posterior distributions, unless there is a large amount of data. The prior must reflect the true state of knowledge **before the analysis is conducted**. However, one might do a sensitivity analysis using different priors to determine the degree that they influence the posterior.