8 Feb 2013

eBay-Google Bayesian short course: Problem set 4 solutions (partial)

1. All of the example output involving BUGS in the class so far has been with WinBUGS; here for variety I'll illustrate classicBUGS. As usual I create three files: poisson1.bug specifies the model, poisson1.dat loads the data, and poisson1.in specifies the initial values.

poisson1.bug

```
model poisson1;

const

  n = 14;

var

  lambda, y[ n ], y.new;

data in "poisson1.dat";
inits in "poisson1.in";

{

  lambda ~ dgamma( 0.001, 0.001 );

  for ( i in 1:n ) {

    y[ i ] ~ dpois( lambda );

  }

  y.new ~ dpois( lambda );

}
```

poisson1.dat

```
list( y = c( 1, 2, 1, 1, 1, 2, 2, 4, 3, 6, 2, 1, 3, 0 ) )
```

poisson1.in

```
list( lambda = 2.0 )
```

My session with `classicBUGS` (under `Solaris UNIX`), in which I specified a burn-in of 1,000 iterations (far longer than necessary but no harm is done [as mentioned in the problem statement, with only a single parameter Gibbs sampling = IID sampling, and it doesn't matter where you start; you're in equilibrium after 1 iteration]) and a monitoring run of 10,000 iterations, went like this:

```
greco 339> bugs

Welcome to BUGS on 21 st Mar 2003  at 17:40:6
BUGS : Copyright (c) 1992 .. 1995 MRC Biostatistics Unit.
All rights reserved.
Version 0.603 for unix systems.
For general release: please see documentation for disclaimer.
The support of the Economic and Social Research Council (UK)
is gratefully acknowledged.

Bugs> compile( "poisson1.bug" )

  [here classicBUGS just repeated the .bug file]

Parsing model declarations.
Loading data value file(s).
Loading initial value file(s).
Parsing model specification.
Checking model graph for directed cycles.
Generating code.
Generating sampling distributions.
Generating initial values
Checking model specification.
Choosing update methods.
compilation took  00:00:00

Bugs>update( 1000 )

      time for    1000   updates was  00:00:00

Bugs>monitor( lambda )

Bugs>monitor( y.new )

Bugs>update( 10000 )

      time for    10000  updates was  00:00:01

Bugs>stats( lambda )

     mean        sd       2.5% :  97.5%  CI    median      sample
  2.066E+0   3.868E-1   1.391E+0   2.908E+0   2.038E+0     10000
```

```
Bugs>stats( y.new )

     mean         sd       2.5% :  97.5%  CI    median      sample
  2.077E+0    1.503E+0    0.000E+0    6.000E+0    2.000E+0      10000

Bugs>q( )

greco 340>
```

You can use the `stats` command in `classicBUGS` to get the usual summary statistics for each node in the graph. One way (perhaps the most stringent way we've looked at) to decide how long to monitor the chain to get 3-significant-figure accuracy for the posterior mean is the Brooks-Draper diagnostic: from the initial monitoring run of 10,000 iterations the estimated posterior mean for λ is `2.066E+0`, so $b = 0$; choosing $\alpha = 0.05$ for 95% Monte Carlo confidence and $k = 3$, taking 0.3868 for the current posterior SD estimate, and using $\rho_1 = 0$ because the series is IID yields

$$m \geq 4(1.960)^2 \left(\frac{0.3868}{10^{0-3+1}} \right)^2 \left(\frac{1+0}{1-0} \right) \doteq 22,989 \doteq 23,000. \tag{1}$$

The corresponding calculation for y_{n+1} yields $m \geq 347,000$, because the posterior SD is so much larger on the predictive scale (but even on my not-very-fast (550 Unix MHz) workstation this only takes 8 seconds).

After a similar `classicBUGS` session with a burn-in of length 1,000 and a monitoring run of 347,000 iterations, there are two new files in the directory in which I ran `BUGS`: `bugs.out` (which contains the MCMC data set) and `bugs.ind` (which tells me how `BUGS` has stored that data set):

```
greco 350> more bugs.ind

lambda              1               347000
y.new               347001          694000
```

(i.e., `BUGS` has put all of the simulated draws for λ in the first 347,000 rows and all of the y_{n+1} draws in the last 347,000). Looking also at `bugs.out` you can see that `classicBUGS` writes out two columns, one for the iteration number and one for the quantity being monitored:

```
greco 353> more bugs.out

1001        2.13775
1002        2.05825
1003        2.70570
1004        1.30209
1005        2.31741
```

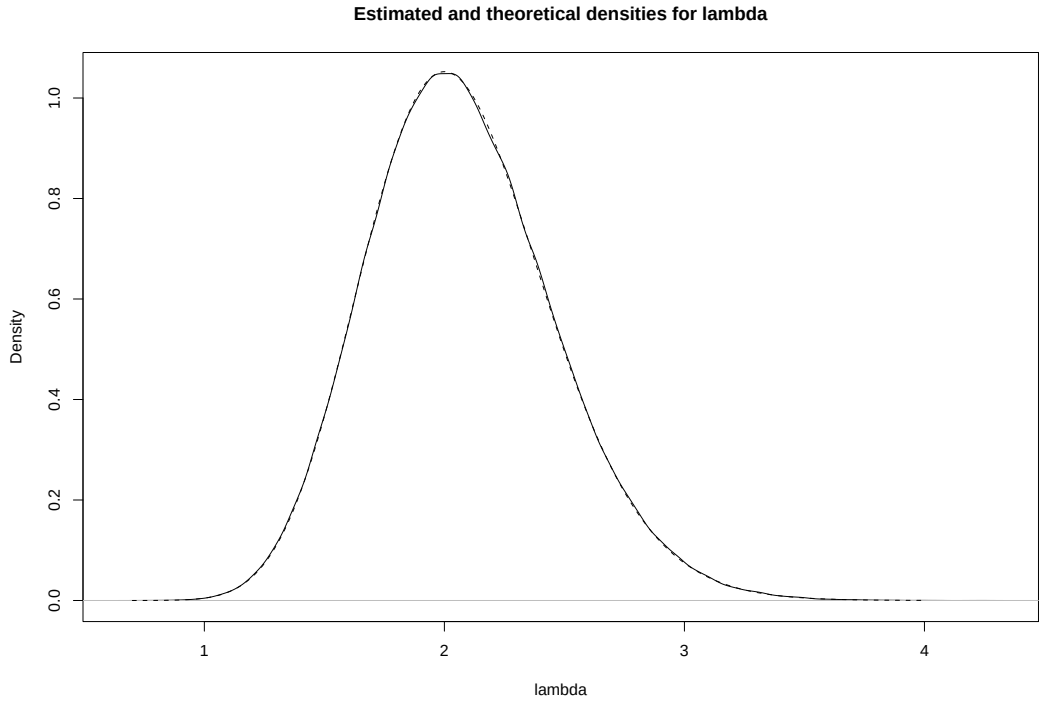

Figure 1: *A comparison of the true posterior distribution for λ (solid line) with its MCMC estimate (dotted line).*

Here's some `R` code to read the `bugs.out` file and make some comparisons between the Monte Carlo and theoretical results:

```
greco 361> R

R : Copyright 2003, The R Development Core Team
Version 1.6.2  (2003-01-10)

[Previously saved workspace restored]

> lambda <- matrix( scan( "bugs.out" ), 694000, 2,
    byrow = T )[ 1:347000, 2 ]

Read 1388000 items

> y.new <- matrix( scan( "bugs.out" ), 694000, 2,
    byrow = T )[ 347001:694000, 2 ]

Read 1388000 items

> mean( lambda )
```

```
[1] 2.072043

> alpha.star <- 29.001

> beta.star <- 14.001

> alpha.star / beta.star

[1] 2.071352

> sd( lambda )

[1] 0.3848807

> sqrt( alpha.star / beta.star^2 )

[1] 0.3846338

> mean( y.new )

[1] 2.070055

> sd( y.new )

[1] 1.488519

> sqrt( alpha.star * ( 1 + 1 / beta.star ) / beta.star )

[1] 1.48973

> postscript( "hw4-1.ps" )

> plot( density( lambda ), type = 'l', xlab = 'lambda',
    main = 'Estimated and theoretical densities for lambda' )

> lambda.grid <- seq( 0.7, 4.0, length = 500 )

> lines( lambda.grid, dgamma( lambda.grid, shape = 29.001,
    scale = 1 / 14.001 ), lty = 2 )

> dev.off( )

> postscript( "hw4-2.ps" )

> hist( y.new, probability = T, breaks = 0:14 - 0.5, xlab = 'y.new',
    main = 'Estimated and theoretical pmfs for y.new',
    ylab = 'Probability' )
```

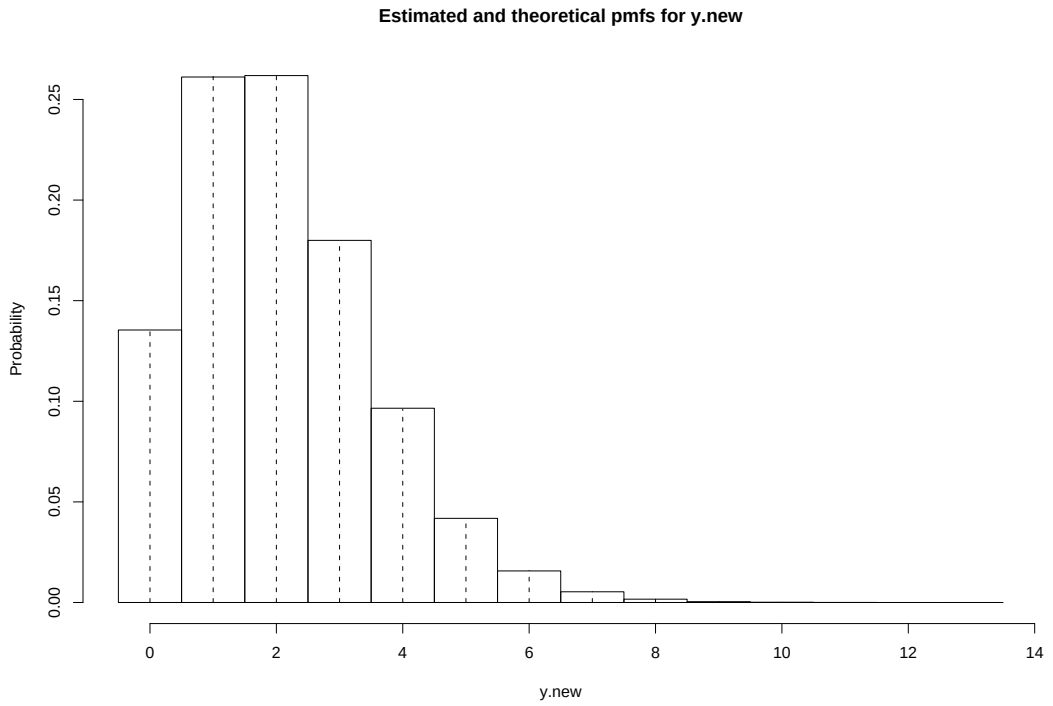

Figure 2: *A comparison of the true posterior predictive distribution for y_{n+1} (solid line) with its MCMC estimate (dotted line).*

```
> pmf <- exp( lgamma( alpha.star + 0:13 ) + alpha.star * log(
    beta.star / ( 1 + beta.star ) ) + 0:13 * log( 1 /
    ( 1 + beta.star ) ) - lgamma( alpha.star ) - lgamma( 0:13 + 1 ) )

> for ( i in 0:13 ) {

    segments( i, 0, i, pmf[ i + 1 ], lty = 2 )

  }

> dev.off( )
```

You can see that a monitoring run of 347,000 has indeed met (and actually exceeded, in this case) the target of 3-sigfig accuracy: the theoretical posterior means and SDs differ from their Monte Carlo estimates by 1 or 2 ticks in the fourth sigfig for both λ and y_{n+1}, and the kernel density traces and histograms based on the MCMC output (Figures 1 and 2) differ from their theoretical counterparts by amounts which are visually negligible.

As for the MCMC accuracy of the posterior SD, equation (17) from chapter 4 of the lecture notes indicates, when the marginal posterior is not far from Gaussian and the time series for the given parameter behaves like white noise, that the MCSE for the posterior SD is smaller than that for the posterior mean by a factor of $\sqrt{2}$, which means that the required

monitoring run length to get the same accuracy is smaller for the SD than the mean by a factor of 2. This calculation would apply here with reasonable force to λ (you can see from Figure 1 that its posterior distribution is approximately Gaussian) but would be less convincing for y_{n+1} (Figure 2 shows that its distribution is considerably more skewed). (to be continued)

2. (to be continued)

3. (a) The likelihood function here is

$$l(\theta_1, \theta_2, \theta_3 | y_1, y_2, y_3) = c\, \theta_1^{y_1}\, \theta_2^{y_2}\, \theta_3^{y_3}, \tag{2}$$

and the log likelihood function is

$$ll(\theta|y) = c + y_1 \ln(\theta_1) + y_2 \ln(\theta_2) + y_3 \ln(\theta_3); \tag{3}$$

the MLEs are found by maximizing (3) subject to the constraint

$$g(\theta) = \theta_1 + \theta_2 + \theta_3 - 1 = 0. \tag{4}$$

Using method (i) mentioned in the problem statement, this is a job for Lagrange multipliers: the constrained extrema will satisfy the system of four equations in four unknowns defined for some λ by (4) plus

$$\begin{aligned}
\frac{\partial}{\partial \theta_1} ll(\theta|y) &= \frac{y_1}{\theta_1} = \lambda \frac{\partial}{\partial \theta_1} g(\theta) = \lambda \\
\frac{\partial}{\partial \theta_2} ll(\theta|y) &= \frac{y_2}{\theta_2} = \lambda \frac{\partial}{\partial \theta_2} g(\theta) = \lambda \\
\frac{\partial}{\partial \theta_3} ll(\theta|y) &= \frac{y_3}{\theta_3} = \lambda \frac{\partial}{\partial \theta_3} g(\theta) = \lambda.
\end{aligned} \tag{5}$$

Adding the three equations in (5) after multiplying equation i by θ_i and using (4) yields

$$y_1 + y_2 + y_3 = n = (\theta_1 + \theta_2 + \theta_3)\lambda = \lambda, \tag{6}$$

from which evidently, for $i = 1, 2, 3$,

$$\left(\hat{\theta}_i\right)_{\text{MLE}} = \frac{y_i}{n}, \tag{7}$$

a result that makes excellent intuitive sense. It remains to demonstrate that (7) defines the maximum of (3); your favorite multivariate calculus book will remind you that a *stationary point* of $ll(\theta|y)$ (a point at which all of its first partial derivatives are 0) is a local maximum if all of the eigenvalues of the *Hessian* (the matrix of second partial derivatives) of $ll(\theta|y)$ at the stationary point are negative. The Hessian of (3) is

$$H(\theta) = \begin{pmatrix} -\frac{y_1}{\theta_1^2} & 0 & 0 \\ 0 & -\frac{y_2}{\theta_2^2} & 0 \\ 0 & 0 & -\frac{y_3}{\theta_3^2} \end{pmatrix}; \tag{8}$$

evaluated at the stationary point (7) it becomes

$$H\left(\hat{\theta}_{\rm MLE}\right) = \begin{pmatrix} -\frac{n^2}{y_1} & 0 & 0 \\ 0 & -\frac{n^2}{y_2} & 0 \\ 0 & 0 & -\frac{n^2}{y_3} \end{pmatrix}. \tag{9}$$

The eigenvalues of (9) are evidently its diagonal elements, which you can see are indeed all negative, so (7) is indeed a local maximum; and (as you can check) it's a global maximum because approaching the relevant boundaries of the domain of (3), namely letting the θ_i get arbitrarily close to 0 or 1, the log likelihood function goes to $-\infty$.

Using method (ii) instead to find the MLEs, substitute (say) $\theta_3 = 1 - \theta_1 - \theta_2$ and $y_3 = n - y_1 - y_2$ into (3) to yield

$$ll(\theta_1, \theta_2 | y_1, y_2) = c + y_1 \ln(\theta_1) + y_2 \ln(\theta_2) + (n - y_1 - y_2) \ln(1 - \theta_1 - \theta_2); \tag{10}$$

setting the first partial derivatives of this with respect to θ_1 and θ_2 equal to 0 yields

$$\begin{aligned} \frac{\partial}{\partial \theta_1} ll(\theta|y) &= \frac{y_1}{\theta_1} - \frac{n - y_1 - y_2}{1 - \theta_1 - \theta_2} = 0 \\ \frac{\partial}{\partial \theta_2} ll(\theta|y) &= \frac{y_2}{\theta_2} - \frac{n - y_1 - y_2}{1 - \theta_1 - \theta_2} = 0. \end{aligned} \tag{11}$$

This is a system of two linear equations in the two unknowns θ_1 and θ_2 whose solution for $i = 1, 2$ is

$$\left(\hat{\theta}_i\right)_{\rm MLE} = \frac{y_i}{n}, \tag{12}$$

and then from the constraint on the parameters and by the functional invariance property of MLEs you get immediately that $\left(\hat{\theta}_3\right)_{\rm MLE} = 1 - \left(\hat{\theta}_1\right)_{\rm MLE} - \left(\hat{\theta}_2\right)_{\rm MLE} = \frac{y_3}{n}$ and $\hat{\gamma} = \left[\left(\hat{\theta}_1\right)_{\rm MLE} - \left(\hat{\theta}_2\right)_{\rm MLE}\right]$. Your favorite book on mathematical statistics will tell you that when the parameter θ of interest is a vector the analogue of Fisher information is a matrix given by

$$\hat{I} = -H\left(\hat{\theta}_{\rm MLE}\right), \tag{13}$$

and that for large n the repeated-sampling distribution of $\hat{\theta}_{\rm MLE}$ is approximately multivariate Gaussian with mean θ and covariance matrix $\hat{V} = \hat{I}^{-1}$, the inverse of the Fisher information matrix. Here, because of the constraint on the parameters, (9) can't be used to compute the Hessian; you need to apply (13) to the method–(ii) parameterization. At this point (if not before) `Maple` starts to become quite useful:

```
greco 2841> maple

    |\^/|     Maple 7 (SUN SPARC SOLARIS)
._|\|   |/|_. Copyright (c) 2001 by Waterloo Maple Inc.
 \  MAPLE  /  All rights reserved. Maple is a registered trademark of
 <____ ____>  Waterloo Maple Inc.
      |       Type ? for help.
```

```
> assume( n > 0, y1 > 0, y2 > 0, theta1 > 0, theta1 < 1, theta2 > 0,
    theta2 < 1 );

> with( linalg ):

Warning, the protected names norm and trace have been redefined and
  unprotected

> ll := ( y1, theta1, y2, theta2, n ) -> y1 * ln( theta1 ) +
    y2 * ln( theta2 ) + ( n - y1 - y2 ) * ln( 1 - theta1 - theta2 );

ll := (y1, theta1, y2, theta2, n) ->

    y1 ln(theta1) + y2 ln(theta2) + (n - y1 - y2) ln(1 - theta1 - theta2)

> hessian( ll( y1, theta1, y2, theta2, n ), [ theta1, theta2 ] );

     [    y1~           n~ - y1~ - y2~              n~ - y1~ - y2~     ]
     [- -------- - ----------------------- , - ----------------------]
     [         2                          2                          2]
     [  theta1~    (1 - theta1~ - theta2~)      (1 - theta1~ - theta2~) ]
     [                                                                ]
     [       n~ - y1~ - y2~            y2~           n~ - y1~ - y2~     ]
     [- ----------------------- , - -------- - ----------------------]
     [                         2           2                          2]
     [  (1 - theta1~ - theta2~)      theta2~    (1 - theta1~ - theta2~) ]

> H := subs( n - y1 - y2 = y3, 1 - theta1 - theta2 = theta3,
    hessian( ll( y1, theta1, y2, theta2, n ), [ theta1, theta2 ] ) );

                [    y1~        y3               y3         ]
                [- -------- - -------         - -------     ]
                [         2         2                2      ]
                [  theta1~    theta3           theta3       ]
           H := [                                          ]
                [         y3               y2~        y3   ]
                [     - -------          - -------- - -------]
                [             2                  2         2]
                [       theta3           theta2~    theta3 ]

> Ihat := simplify( eval( - matrix( H ), [ theta1 = y1 / n,
    theta2 = y2 / n, theta3 = y3 / n ] ) );

                      [    2                      2     ]
                      [  n~  (y3 + y1~)           n~    ]
                      [- --------------         - ---   ]
                      [      y3 y1~               y3     ]
```

```
                Ihat := -[                                  ]
                         [         2              2         ]
                         [       n~             n~  (y3 + y2~)]
                         [     - ---          - --------------]
                         [       y3                 y3 y2~    ]
```

```
> V := subs( y1 + y2 + y3 = n, y1 + y2 = n - y3, y1 + y3 = n - y2,
    y2 + y3 = n - y1, simplify( inverse( Ihat ) ) );
```

```
                         [(n~ - y1~) y1~        y1~ y2~   ]
                         [--------------      - -------   ]
                         [       3                 3      ]
                         [     n~                n~       ]
                    V := [                                ]
                         [    y1~ y2~       (n~ - y2~) y2~]
                         [  - -------       --------------]
                         [        3                3      ]
                         [      n~               n~       ]
```

If you now do this exercise over again two more times, using the constraint to successively remove first θ_1 and then θ_2 from the log likelihood function instead of θ_3, and if you substitute (12) into the final expression `Maple` came up with above, you'll see a pattern that permits you to work out the covariance matrix for the MLE $\hat{\theta} = \left(\hat{\theta}_1, \hat{\theta}_2, \hat{\theta}_3\right)$ of the entire parameter vector:

$$\hat{V} = \begin{pmatrix} \frac{\hat{\theta}_1\left(1-\hat{\theta}_1\right)}{n} & -\frac{\hat{\theta}_1\hat{\theta}_2}{n} & -\frac{\hat{\theta}_1\hat{\theta}_3}{n} \\ -\frac{\hat{\theta}_1\hat{\theta}_2}{n} & \frac{\hat{\theta}_2\left(1-\hat{\theta}_2\right)}{n} & -\frac{\hat{\theta}_2\hat{\theta}_3}{n} \\ -\frac{\hat{\theta}_1\hat{\theta}_3}{n} & -\frac{\hat{\theta}_2\hat{\theta}_3}{n} & \frac{\hat{\theta}_3\left(1-\hat{\theta}_3\right)}{n} \end{pmatrix}. \tag{14}$$

The diagonal entries of this matrix (the sampling variances of the parameters) are familiar from our earlier work with the Bernoulli/ binomial likelihood: (14) says that for each component of the parameter vector

$$\widehat{SE}\left(\hat{\theta}_i\right) = \sqrt{\frac{\hat{\theta}_i\left(1-\hat{\theta}_i\right)}{n}}. \tag{15}$$

(14) can also be used to work out a large-sample standard error for $\hat{\gamma}$, which can be written as a linear combination of the components of the $\hat{\theta}$ vector: $\hat{\gamma} = a^T\hat{\theta}$ for $a^T = (1, -1, 0)$. Your mathstat book will also no doubt mention that if θ is a random vector with covariance matrix $\hat{V}$ then in repeated sampling

$$V(\hat{\gamma}) = V\left(a^T\hat{\theta}\right) = a^T\hat{V}a. \tag{16}$$

```
> assume( theta1hat > 0, theta1hat < 1, theta2hat > 0, theta2hat < 1,
    theta3hat > 0, theta3hat < 1, n > 0 );
```

```
> Vhat := matrix( 3, 3, [ theta1hat * ( 1 - theta1hat ) / n, - theta1hat *
    theta2hat / n, - theta1hat * theta3hat / n, - theta1hat *
    theta2hat / n, theta2hat * ( 1 - theta2hat ) / n, - theta2hat *
    theta3hat / n, - theta1hat * theta3hat / n, - theta2hat *
    theta3hat / n, theta3hat * ( 1 - theta3hat ) / n ] ):

> a := matrix( 3, 1, [ 1, -1, 0 ] );

                                        [ 1]
                                        [  ]
                                   a := [-1]
                                        [  ]
                                        [ 0]

> evalm( transpose( a ) &* Vhat &* a );

    [theta1hat~ (1 - theta1hat~)     theta1hat~ theta2hat~
    [-------------------------- + 2 ---------------------
    [           n~                           n~

       theta2hat~ (1 - theta2hat~)]
     + --------------------------]
                  n~             ]
```

Of course the other way to compute the repeated-sampling variance of $\hat{\gamma}$ is to do so directly:

$$V(\hat{\gamma}) = V\left(\hat{\theta}_1 - \hat{\theta}_2\right) = V\left(\hat{\theta}_1\right) + V\left(\hat{\theta}_2\right) - 2C\left(\hat{\theta}_1, \hat{\theta}_1\right), \tag{17}$$

where $C\left(\hat{\theta}_1, \hat{\theta}_1\right)$ is the covariance of $\hat{\theta}_1$ and $\hat{\theta}_2$, which is estimated in (14) to be $-\frac{\hat{\theta}_1\hat{\theta}_2}{n}$. Putting this together either way yields

$$\widehat{SE}(\hat{\gamma}) = \sqrt{\frac{\hat{\theta}_1\left(1-\hat{\theta}_1\right)}{n} + 2\frac{\hat{\theta}_1\hat{\theta}_2}{n} + \frac{\hat{\theta}_2\left(1-\hat{\theta}_2\right)}{n}}. \tag{18}$$

With the data from the CBS News poll the MLEs and their estimated standard errors (in parentheses) work out as follows: $\hat{\theta}_1 = 0.502\,(0.0131), \hat{\theta}_2 = 0.403\,(0.0129), \hat{\theta}_3 = 0.0947$ (0.00770), and $\hat{\gamma} = 0.0995\,(0.0249)$.

(b) (to be continued)