Symbolic Computation for Statistical Inference: An Implementation in R

D. F. Andrews

University of Toronto

March 2004 rev 2006

This manual presents examples of the use of Symbolic Computation for Statistical Inference using R. A full discussion of the theory on which the computations are based is found in the book by Jamie Stafford and the author of the same title. This implementation uses simpler notation appropriate for computing in R. General procedures are defined which may be used for specific problems. These applications are illustrated by computing unbiased estimates of moments and cumulants, and the means, variances and other moments of

i) correlation coefficients
ii) maximum likelihood estimates
iii) more general M-estimates
iv) deviance or log likelihood ratio
v) generalized signed roots of deviance

The results are expressed as expansions in 1/n, one over the sample size. They may computed to any order. The resulting expressions can be evaluated for numerical data.

This introduction covers the case of independent and identically distributed random variables. Extensions to p-dimensional parameters have been developed and are discussed separately. The discussion presents the one dimensional, real case.

WHAT IS THE USE OF DOING IT SYMBOLICALLY

Symbolic calculation is useful for speed and generality.

The computed symbolic expressions involve simple averages and functions. As a result they are evaluated very quickly and precisely. This speed is useful in applications involving repeated calculations as for example in Monte Carlo. The expressions for symbolic moments are equivalent to bootstrap estimates with a large number of bootstrap samples. (The symbolic results are exact or correct to n^(-k/2) where the bootstrap estimates based on m bootstrap samples are correct to m^(-1/2).)

The expressions are general. Their application requires the definition only of expectations of simple random variables. For example, the calculation of properties of a likelihood ratio statistic requires only the definition of the expectation of the products of log-likelihood derivatives. These can be specified for any distribution.

SOURCE CODE

The basic procedures are contained in the file SCSI.1.0.2.txt. The file can be obtained from http://www.fisher.utstat.utoronto.ca/david/Sym2006/Sym2006.html . To use the procedures, begin the R code with

```
source(file="http://www.fisher.utstat.utoronto.ca/david/Sym2006/SCSI.1.0.2.txt")
```

Alternatively, save the file

http://www.fisher.utstat.utoronto.ca/david/Sym2006/SCSI.1.0.2.txt

to a convenient location and modify the following the source statement to point the that location.

SYMBOLIC FUNCTION S

The calculations are defined by the function S. This function has one argument, a string containing the expression to be evaluated.

USAGE

```
 S( text )
```

The functions are rather easy to use. For example, the function EA computes expectations, E, from averages, A. To compute the expected value of the product of two averages,

```
S( EA( A(X) * A(Y) ) )
```

produces the expression

```
(( 1  +  -1 *  1/n)) * (E(X) * E(Y))  +  ( 1/n) * E(X * Y)
```

The function CE computes cumulants, C, from expectations, E. The command

```
S( CE(EA(A(X) * A(Y))) )
```

produces

```
(C(X) * C(Y))  +  ( 1/n) * C(X,Y)
```

an expression involving the covariance.

Fisher computed unbiased estimates of cumulants. These are expressed in terms of averages. Here, the simple function AE produces the unbiased estimates expressed in averages, A, from expectations, E. The unbiased estimate of the fourth cumulant is computed by expressing the cumulant in terms of expectations and then applying AE.

```
ac4 <- S( AE( EC( C( X, X, X, X ) ) ) )
```

The result is rather lengthy but is simply evaluated numerically.

```
n <- 100; X <- rnorm(n); Eval( ac4 )
```

The function, S, returns an object which can be used by other R functions. The R function, Eval, evaluates objects numerically. For this reason, the results are expressed in simple text with no subscripts or foreign alphabets. We are comforted by the Red Green maxim: if folks don't find you pretty, they should find you useful. In some places, harmless redundant parentheses appear. Note however that the objects returned are rather special. Evaluation of addition, multiplication etc. of these objects should be done with the function s. Eval converts the objects to expressions which are then evaluated in R.

The function s is rather simple. It can be used for the types of expression illustrated here. For clarity and other reasons, it is useful to break up a complex expression and assign the intermediate results as is illustrated in the correlation example below.

The symbolic calculations involve simple objects and basic transformations described below. These simple operations permit quite complex calculations such as signed roots of log likelihood ratios.

FUNDAMENTAL OBJECTS

The objects of statistics are typically expressed as averages, expectations (moments) and cumulants denoted here by A, E, C respectively. The standard calculations of statistics may be described as transformations of objects of one type into object to another type. For example, consider the estimate of a population mean. The estimate is the sample average, denoted here by A(X). The variance of this estimate is expressed in terms of expectations E: (E(X*X) – E(X)*E(X))/n. An object composed of averages, A, has been transformed into an object composed of expectations, E. Since almost all estimates can be expressed in terms of averages, and most statistical properties can be expressed in terms of expectations, this is a very common and useful transformation. The calculation involves of a transformation of averages into expectations. Several useful transformations have been implemented. The objects and the transformations are described below.

A: AVERAGES

Averages are denoted by A. Here we create and print the average of the product of two variables X and Y. The input and output have been cut and pasted from R. The input lines are shown here with the prompt "<". These commands may be executed by cutting the line from the text – without the prompt "<" -- and pasting the cut into R.

```
> axy <- S(   A(X*Y)   );   axy
A(X * Y)
```

One linear property of averages has been implemented as can be seen from the following.

```
> S(  A(2*X + 3*Y)  )
(2) * A(X)  +
 (3) * A(Y)
```

EVAL: NUMERICAL EVALUATION

The assigned expressions may be evaluated for data using the function Eval, and compared to the stand numerical calculation. The requires the definition of n, the sample size, and values for the variables. The evaluation of the symbolic expression is given with the standard numerical result.

```
> n<- 10; X <- 1:n; Y<- n:1;
> c(  Eval(axy), mean(X*Y))
[1] 22 22
```

The standard R syntax may be used for arithmetic expressions. For example, here we give the expression for the Gaussian MLE estimate of variance.

```
>  estvarx <- S(   A(X*X) - A(X)*A(X)   );
> estvarx
A(X * X)  +
  (-1) * (A(X) * A(X))
```

The expression is compared with the standard R numerical calculation.

```
> c( Eval(estvarx), var(X)*(n-1)/n )
[1] 8.25 8.25
```

E: EXPECTED VALUES OR MOMENTS

Expected values, or moments, are denoted by E. Here we give the expected value of a linear combination of random variables.

```
>S(  E(2*X + 3*Y)  )
(2) * E(X)  + (3) * E(Y)
```

CHANGING THE TYPES OF OBJECTS

The objects, A, C, and E are related by simple, fundamental transformations. Each type may be transformed to another. These transformations correspond to common statistical calculations. The transformations are denoted by pairs of letters. The first letter denotes the type of the result, and the second, the type of the argument. Thus EA produces expectations from averages. It is applied to averages. The transformations may be composed. The notation helps in checking whether the calculation is meaningful. Thus CE(EA(A(...))) makes sense and all letters except the first come in pairs. But EC(EA(A(...))) does not because the first operator, EC, should have cumulants (C) as its argument where in fact it has expectations produced by EA(...).

EA: EXPECTATION Computing expected values of expressions involving averages.

The transformation that produces the expected value of an expression involving averages is denoted by EA. This transformation assumes that the A's represent the average of n independent realizations of random variables. The expected value of the estimate of variance may be computed by

```
> eestvar <- S(   EA(A(X*X) - A(X)*A(X))   )
> eestvar
(( -1 *  1/n  +  1)) * E(X * X)  +
  (( -1  +   1/n)) * (E(X) * E(X))
```

AE: UNBIASED ESTIMATES Computing unbiased estimates of expectations

The above example shows the bias of the MLE estimate of variance. The unbiased estimate of variance may be simply computed using the operator AE.

```
>  uestvar <- S(   AE(E(X*X)- E(X)*E(X))   );   uestvar
((  1/(n-1)  +  1)) * A(X * X)  +
  (( -1  +  -1 *  1/(n-1))) * (A(X) * A(X))
```

This result can be checked numerically by evaluating the expression and comparing it to the usual estimate of variance.

```
  > c( Eval(uestvar), var(X) )
  [1] 9.166667 9.166667
```

CE: Computing a cumulant expression from one involving expectations

Moments or expectations can be reexpressed in terms of cumulants. The following example shows that the variance is just the second cumulant.

```
> S( CE(E(X*X)- E(X)*E(X))   )
C(X,X)
```

EC: Computing a expectation expression from one involving cumulants

Cumulants can be expressed in terms of expectatations.

```
  >S(  EC(C(X,X))  )
```

```
(-1) * (E(X) * E(X))  +
  E(X * X)
```

BE: Computing the bootstrap estimate from a parameter expressed as expecations.

Most statistical parameters may be expressed in terms of expectations. Expectations are just averages with respect to a distribution. The basic bootstrap estimate computes the corresponding expression in terms of averages with respect to the empirical distribution. These averages are just the simple averages denoted by A. So the symbolic computation of a simple bootstrap estimate is achieved by simply changing the E's to A's. (The symbolic bootstrap is not computationally intensive.) The operator BE does this. Here we compute the bootstrap estimate of a variance.

```
> S(  BE(E(X*X)- E(X)*E(X))  )
A(X * X)  + (-1) * (A(X) * A(X))
```

EXAMPLE: The bias and variance of the bootstrap estimate of variance

In this example we computed the variance of the bootstrap estimate of variance. First we compute the bootstrap estimate of variance called bvar.

```
> varx <- S(  E(X*X)- E(X)*E(X)  )
> bvar <- S( BE(varx) ); bvar
A(X * X)  +  (-1) * (A(X) * A(X))
```

The bias of the bootstrap estimate is simply computed

```
> S( EA(bvar) - varx )
(-1 *  1/n) * E(X * X)  +  ( 1/n) * (E(X) * E(X))
```

The bootstrap variance has a bias which is just the variance times -1/n. It underestimates the variance.

Next we compute varbvar, the variance of the bootstrap estimate of variance, bvar.

```
varbvar <- S( EA(bvar*bvar) – EA(bvar)*EA(bvar)  )
```

The resulting expression is rather long. It is more simply expressed in terms of cumulants using the operator CE.

```
> S(  CE(varbvar)  )
(( 2 *  1/n  +  -2 *  1/n^2)) * (C(X,X) * C(X,X))  +
  ((  1/n  +  -2 *  1/n^2  +  1/n^3)) * C(X,X,X,X)
```

The variance of bvar depends on the the square of the variance, C(X,X), but also on the fourth cumulant, C(X,X,X,X) – to the same order, 1/n. This shows the heavy dependence of variance of the estimate on distribution shape.

ASYMPTOTIC EXPANSIONS

The procedures described above apply to linear combinations of products of fundamental. To extend calculations to logarithms, MLE's etc., we need to first convert these into this form of linear combinations of products. The standard conversion is achieved by expanding objects in Taylor expansions. These expansions approximate the object to order n^(-a/2). Machine computations make precise approximation possible.

The first step in investigating an estimate is to compute it's expansion. The function APPROX, produces an expansion. Here we produce the expansion of an average A(X) to order n^-(4/2)

```
> S( APPROX(A(X),4) )
  ( E((X))) +  Z(X)
```

The average, A(X), has been replaced by its expectation and the average deviation Z(X) = A(X – E(X)). Average deviations, Z, are small, typically of order, in probability, of n^(-1/2).

EXPANSION OF FUNCTIONS

Functions of random variables are expanded using standard Taylor series. The following example gives the expansion of f(A(X)). Note that the expansion stops at the term of order n^(-4/2). This order was specified above in the approximation of A(X).

```
>S( f( APPROX(A(X),4) ) )
  ( f(( E((X))))) +
 ( f(( E((X))),1)) * Z(X) +
 (0.5 *  f(( E((X))),2)) * (Z(X) * Z(X)) +
 (0.16666666667 *  f(( E((X))),3)) * (Z(X) * Z(X) * Z(X)) +
 (0.041666666667 *  f(( E((X))),4)) * (Z(X) * Z(X) * Z(X) * Z(X))
```

where the i'th derivative of f is denoted by f(, i).

Some common functions including log, sqrt and powers, denoted by "^" have been implemented. The i'th derivative of f is denoted by f(, i).

```
> S( log( APPROX(A(X),4) ) )
  ( log(( E((X))))) +
 ( (( E((X))))^-1) * Z(X) +
 (-0.5 *  (( E((X))))^-2) * (Z(X) * Z(X)) +
 (0.333333333333333 *  (( E((X))))^-3) * (Z(X) * Z(X) * Z(X)) +
 (-0.25 *  (( E((X))))^-4) * (Z(X) * Z(X) * Z(X) * Z(X))
```

EZ: Computing expected values of expressions involving Z's

The transformation EA computes expectations of expressions involving averages, A. Asymptotic expansions involve standardized averages, Z. The transformation EZ computes expected values from Z's. Here we compute the expected value of the square root of an average – to order n^(-4/2)

```
> S( EZ( sqrt( APPROX(A(X), 4) ) ) )
  ( (( E((X)))^(0.5))) +
 (-0.125 *  (( E((X)))^(-1.5))*1/n) * E(z(X) * z(X)) +
 (0.0625 *  (( E((X)))^(-2.5))*1/n^2) * E(z(X) * z(X) * z(X)) +
   (-0.1171875 *  (( E((X)))^(-3.5))*1/n^2) * (E(z(X) * z(X)) *
E(z(X) * z(X)))
```

The result is expressed in terms of expectations of centered random variables z(X) = X – E(X).

EXAMPLE: CORRELATION

The properties of the correlation coefficient are known for the Gaussian distribution. Here we show how the moments of the statistic can be computed in general. This is illustrated by evaluating the moments for the empirical distribution. Defining other functions for E, will yield moments for any distribution.

The expansion to order n^(2/2) is defined. The first step specifies expansions of the first and second sample moments required.

```
> sax <- S( APPROX(A(X), 2) )
> say <- S( APPROX(A(X), 2) )
> saxx <- S( APPROX(A(X*X), 2) )
> saxy <- S( APPROX(A(X*Y), 2) )
> sayy <- S( APPROX(A(Y*Y), 2) )
```

Expansions of square roots are used to define the statistic.

```
> cxx <- S( saxx - sax*sax )
> cxy <- S( saxy - say*say )
> cyy <- S( sayy - sax*say )
> corxy <- S(cxy*cxx^(-1/2)*cyy^(-1/2) )
```

The expectation and variance are computed below.

```
> Ecorxy <- S( EZ(corxy) )
> Vcorxy <- S( EZ(corxy*corxy)-Ecorxy*Ecorxy )
```

The resulting expressions are lengthy. However, they may be evaluated numerically. Here we evaluate them for the empirical distribution. In this case, E() is an average. The required functions are defined.

```
> E<- function(x){mean(x)}
```

Gaussian data were generated with true correlation 1/sqrt(2).

```
> set.seed(42)
> n <- 100; z1 <- rnorm(n); z2 <- rnorm(n)
> X <- z1 - mean(z1)
> Y <- z1+z2 - mean(z1+z2)
```

The expectation and variance of the statistic are computed for the empirical distribution.

```
> Eval(Ecorxy)
[1] 0.753956
> Eval(Vcorxy)
[1] 0.002475547
```

The asymptotic variance for the Gaussian distribution with correlation, r is (1-r^2)^2 /n. In this case the value would be 0.0025. The expression for the variance permits evaluation for any distribution.

AVERAGES OF FUNCTIONS: ROOTS OR INVERSES

Likelihood and M-estimates are defined in terms of averages of functions. For example, M-estimates are defined as the root, thetahat, of the equation A(psi(thetahat, X)) = 0. To shorten the expressions, the dependence of the function on X is supressed, and we write psi(thetahat) for psi(theahat,X) . This gives the the equation A(psi(thetahat)) = 0. The solution thetahat is the inverse average function denoted here by InverseA. The expansion of thethat about theta is computed below.

```
> S( InverseA( psi(APPROX(theta,2)) ) )
  ( theta) +
 (-1 *  1/E(psi(( theta),1))) * Z(psi(( theta))) +
 ( 1/E(psi(( theta),1))^2) *(Z(psi(( theta)))*Z(psi(( theta),1)))  +
  (-0.5 *  1/E(psi(( theta),1))^3*E(psi(( theta),2))) * (Z(psi((
theta))) * Z(psi(( theta))))
```

where the order of the expansion if specified by the argument of APPROX.

A LIKELIHOOD EXAMPLE

The maximum likelihood estimate is defined as above where psi is the score function, the first derivative of log-likelihood function l. The estimate is computed below.

```
>S( InverseA( l(APPROX(theta,2),1) ) )
  ( theta) +
 (-1 *  1/E(l(( theta),2))) * Z(l(( theta),1)) +
 ( 1/E(l(( theta),2))^2) * (Z(l(( theta),1)) * Z(l(( theta),2)))  +
  (-0.5 *  1/E(l(( theta),2))^3*E(l(( theta),3))) * (Z(l(( theta),
1)) * Z(l(( theta),1)))
```

The average deviance or twice the drop in average log-likelihood is computed below to order n^(-4/2). The usual theta has been changed to t0 to shorten the printed result.

```
   > theta0 <- S( APPROX(t0, 4) )
   > thetahat <- S( InverseA( l(theta0, 1) ) )
   > hadev <- S( A(l(thetahat)) - A(l(theta0)) )
   > S( EZ(hadev + hadev) )
 (-1 *  1/E(l(( t0),2))*1/n) * E(z(l(( t0),1)) * z(l(( t0),1))) +
 ( 1/E(l(( t0),2))^2*1/n^2) * E(z(l(( t0),1)) * z(l(( t0),1)) *
     z(l(( t0),2)))  +
  (-0.333333333333333 *  1/E(l(( t0),2))^3*1/n^2*E(l(( t0),3))) *
     E(z(l(( t0),1)) * z(l(( t0),1)) * z(l(( t0),1))) + ….
```

The resulting expression can be evaluated by defining functions for l , its derivatives and their expected values.

SIGNED ROOTS

This lengthy example illustrates the use of symbolic computation in a complex calculation.

The signed root of the deviance or log likelihood ratio is relatively symmetric and has mean and variance that are relatively constant. It is therefore useful as a basis for constructing confidence intervals. However in the typical case where the underlying distribution of the data is not known exactly, the moments of the signed root must be estimated. In this example, the variance of the signed root is computed and used to calculate confidence intervals.

Let f be an arbitrary function corresponding to the likelihood. We do not assume that it is a log density. Its first derivative is used to define the estimate in the same way as the M-estimate or the mle above. This is used to compute dev, the generalized average deviance.

```
theta0 <- S( APPROX(t0, 4) ) # point of expansion
thetahat <- S( InverseA( f(theta0, 1) ) ) # estimate
adev <- S( A(f(thetahat)) - A(f(theta0)) )
```

Next, the signed root of the average deviance is computed. The Taylor expansion of A(f(thetahat, 1)) gives an expression for A(f(theta0, 1)). Substituting this in the Taylor expansion of adev gives an expression with (thetahat – theta0)^2 as a factor. The expansion of the signed root is computed by taking the square root of the second factor.

This done by noting that the Taylor expansion of the deviance can be factored

```
dt <- S( thetahat - theta0 ) #deviation of estimate
el2 <- S( A(f(theta0,2)) ) #average 2nd derivative
mel2 <- S( -el2 ) # times -1
mel2s <- mel2 #make a copy to get expected value
mel2s[2] <- mel2s[3] #set second term to 0
rel2 <- S( sqrt(mel2s) ) #take its square root
rd <- S( rel2 * dt ) # times deviation
fac <- mel2 # first term of factor
for( i in 3:(length(thetahat)-1)) {
      if(i == 3){
             rdp <- dt
      }
      else{
             rdp <- S(rdp * dt )
      }
      deri <- S(paste(" A(f(theta0,",i,”)) ")) #ith derivative
      con <- 2/gamma(i+1)- 2/gamma(i) #make constant an approx.
      scon<- S(paste("APPROX(",con,",",length(thetahat) -1,")"))
      fac <- S( fac + scon*deri*rdp ")
}
rfac <- S( sqrt(fac) ")
rdev <- S( dt * rfac ") #signed root
```

The variance is computed as follows.

```
erdev <- S( EZ(rdev) ) #expected value
erdev2 <- S( EZ(rdev * rdev) )
vrdev <- S( erdev2 - erdev * erdev ) # variance
```

This variance can be used to compute confidence intervals. This is illustrated for the scale parameter of a standard exponential distribution. The estimate is specified by defining f to be the log likelihood.

```
f<- function(x,i=0){
     t1 <- ifelse(i == 0, t1 <- -log(x),
          t1 <- (-1)^(i) * gamma(i) * x^(-i))
     t2 <- X* (-1)^(i+1)*gamma(i+1)*x^(-(i+1))
      (t1+t2)
}
```

The numerical average deviance is given by

```
numf <- function(x,sigma){
     -log(sigma) - mean(x)/sigma
     }
numavd <- function(x,sigma){
     2*(numf(x,mean(x)) - numf(x,sigma))
     }
```

The “exact” confidence interval is

```
numci <- function(x){
     mean(x)*2*length(x)*1/(qchisq(c(0.975,0.025),2*n))
     }
```

The symbolic interval can be compared to the “exact” interval for numerical data.

```
set.seed(42)
n <- 20
X<- -log(runif(n))
t0 <- mean(X)
```

Here the expectations, E(), are defined to the expectations with respect to the empirical distribution. (We leave the definition of E() for known distributions to those who know precisely the distribution of the data.)

```
E<- mean
evd <- Eval(vrdev)
deveq <- function(t0){
     numavd(X,t0)-(1.96^2)*evd
     }
c(uniroot(deveq,c(mean(X)/2, mean(X)))$root,
     uniroot(deveq,c(mean(X),mean(X)+3))$root)
 [1] 0.4247848 0.9980578
```

This can be compared to the “exact” interval which could be used if the distribution were known

```
   numci(X)
   [1] 0.4258268 1.0342261
```

This is compared with 5 bootstrap BCA intervals based on 1000 bootstrap samples

```
library(boot)
for( i in 1:5){
      boot.X <-boot(X,function(x,i) mean(x[i]),R=1000)
      print(boot.ci(boot.X,conf=0.975,type="bca")[[4]][c(4,5)])
}
```

to give

```
   [1] 0.3852296 1.0634212
   [1] 0.4014010 0.9680726
   [1] 0.3775158 0.9894336
   [1] 0.3977897 0.9720249
   [1] 0.3809278 1.0057228
```

The symbolic intervals are computed very quickly. The bootstrap intervals even with 1000 bootstrap samples show considerable variability.