

biogen idec

**User Defined Functions
via PROC FCMP:
The Good, the Bad, and the Ugly**

Jerry W. Lewis, PhD
x41237
BASUG 7 May 2010

Overview

Originally, SAS was not user extensible.

In the 1980's you could write new functions in Fortran or Assembly, and procedures in PL1.

Later SAS added SAS/TOOLKIT® to simplify language extensions.

Now with PROC FCMP (Function Compiler Procedure) it is possible to write your own functions in SAS and use them in data steps and many procedures.

Similar functionality can be achieved by a macro, provided that the computation can be performed by a single SAS expression.

Each of the last two approaches has advantages and disadvantages that will be discussed

FCMP Functions & Subroutines can be used

- Data Step
- Where statement
- ODS
- PROC CALIS
- PROC COMPILE
- PROC COMPUTAB
- PROC GA
- PROC GENMOD
- PROC MCMC
- PROC MODEL
- PROC NLIN
- PROC NLMIXED
- PROC NLP
- PROC PHREG
- PROC REPORT
COMPUTE blocks
- Risk Dimensions
procedures
- PROC SIMILARITY
- PROC SQL (no array
arguments)
- PROC Template
- Others?

Example 1: Clopper-Pearson Binomial Confidence Interval

- Observe s successes in n trials
- 1-sided
 - LCL = **BETAINV**(1-*conf*, s , $n-s$)
 - UCL = **BETAINV**(*conf*, $s+1$, $n-s$)
- 2-sided
 - LCL = **BETAINV**((1-*conf*)/2, s , $n-s$)
 - UCL = **BETAINV**(1-(1-*conf*)/2, $s+1$, $n-s$)

(cf. Hahn & Meeker *Statistical Intervals*)

PROC FCMP BinomialCL Code

```
PROC FCMP OUTLIB=sasuser.FCMP_Cat.fcns;
  FUNCTION BinomCL(s,n,direction,conf,sides);
    IF sides=2 THEN cnf = 1-(1-(conf))/2; ELSE cnf=conf;
    IF direction<0 THEN DO; /* lower confidence limit */
      IF s=0 THEN cl=0; ELSE cl=BETAINV(1-(cnf),s,n-s+1);
    END; ELSE DO; /* upper confidence limit */
      IF s=n THEN cl=1; ELSE cl=BETAINV(cnf,s+1,n-s);
    END;
    RETURN(cl);
  ENDSUB;
RUN;

OPTIONS CMPLIB=(sasuser.FCMP_Cat);
DATA Test; UCL=BinomCL(0,20,1,0.95,1); PROC PRINT; RUN;
```

SAS Macro BinomialCL Code

```
%MACRO BinomCL(s,n,direction,conf,sides);  
  %IF &sides=1 OR &sides=2 %THEN %DO;  
    %IF &sides=2 %THEN %LET conf = 1-(1-(&conf))/2;  
    %IF &direction<0 %THEN /* lower confidence limit */  
      IFN(&s=0,0,BETAINV(1-(&conf),&s,&n-(&s)+1));  
    %ELSE /* upper confidence limit */  
      IFN(&s=&n,1,BETAINV(&conf,&s+1,&n-(&s)));  
  %END;  
  %ELSE .  
%MEND BinomCL;  
  
DATA Test; UCL=%BinomCL(0,20,1,0.95,1); PROC PRINT; RUN;
```

Remarks

- A library of FCMP functions can be located on a common drive and shared by multiple users. `OPTION CMPLIB=` could be in the configuration or `autoexec.sas` file for convenience.
- Macros can be stored in a `.sas` file on a common drive and `%INCLUDED` by multiple users
- Some parameters will rarely be used, e.g. 95% 2-sided confidence intervals are standard.
- The macro approach can support optional parameters with defaults, FCMP cannot.

Optional Macro Parameters

```
%MACRO BinomCL(s,n,direction,conf=0.95,sides=2);  
  %IF "&direction"="" %THEN %LET direction = 1;  
  %IF &sides=1 OR &sides=2 %THEN %DO;  
    %IF &sides=2 %THEN %LET conf = 1-(1-(&conf))/2;  
    %IF &direction<0 %THEN /* lower confidence limit */  
      IFN(&s=0,0,BETAINV(1-(&conf),&s,&n-(&s)+1));  
    %ELSE /* upper confidence limit */  
      IFN(&s=&n,1,BETAINV(&conf,&s+1,&n-(&s)));  
  %END;  
  %ELSE .  
%MEND BinomCL;  
  
DATA Test; UCL=%BinomCL(0,20,sides=1); PROC PRINT; RUN;
```

Best of Both Worlds?

- Since FCMP is more flexible computationally and Macro is more flexible in the user interface, why not combine them?

```
%OPTIONS CMPLIB=(sasuser.FCMP_Cat);  
MACRO BinomCL(s,n,direction,conf=0.95,sides=2);  
    BinomCL(&s,&n,&direction,&conf,&sides);  
%MEND BinomCL;  
  
DATA Test; UCL=%BinomCL(0,20,sides=1); PROC PRINT; RUN;
```

Alternate FCMP Syntax: IF Expressions

```
PROC FCMP OUTLIB=sasuser.FCMP_Cat.fcns;
  FUNCTION BinomCL2(s,n,direction,conf,sides);
    cnf = IF sides=2 THEN 1-(1-(conf))/2 ELSE conf;
    IF direction<0 THEN /* lower confidence limit */
      c1 = IF s=0 THEN 0 ELSE BETAINV(1-(cnf),s,n-s+1);
    ELSE /* upper confidence limit */
      c1 = IF s=n THEN 1 ELSE BETAINV(cnf,s+1,n-s);
    END;
    RETURN(c1);
  ENDSUB;
RUN;
```

Alternate FCMP Syntax: Multi-line WHEN/OTHERWISE Clauses

```
PROC FCMP OUTLIB=sasuser.FCMP_Cat.fcns;
FUNCTION BinomCL3(s,n,direction,conf,sides);
  cnf = IF sides=2 THEN 1-(1-(cnf))/2 ELSE conf;
  SELECT;
    WHEN (direction<0)      /* lower confidence limit */
      IF s=0 THEN cl=0;
      ELSE cl=BETAINV(1-(cnf),s,n-s+1);
    OTHERWISE;              /* upper confidence limit */
      IF s=n THEN cl=1;
      ELSE cl=BETAINV(cnf,s+1,n-s);
  END;
  RETURN(cl);
ENDSUB;
RUN;
```

Example 2: Skellam Quantiles

- $\text{Poisson}(\lambda_1) - \text{Poisson}(\lambda_2) \sim \text{Skellam}(\lambda_1, \lambda_2)$
- CDF (cf. Johnson & Kotz *Univariate Discrete Distributions*):
 $\text{PROBCHI}(2*\lambda_2, -2*\text{FLOOR}(x), 2*\lambda_1) \quad (x < 0)$
 $1 - \text{PROBCHI}(2*\lambda_1, 2*(1 + \text{FLOOR}(x)), 2*\lambda_2) \quad (x \geq 0)$
- Quantile:
Smallest integer x satisfying $\text{CDF} \geq p$
Use Cornish-Fisher asymptotic expansion as a starting value & step up or down till done
- FCMP works quite well, macro approach easily hits SAS's character limit for a single expression.

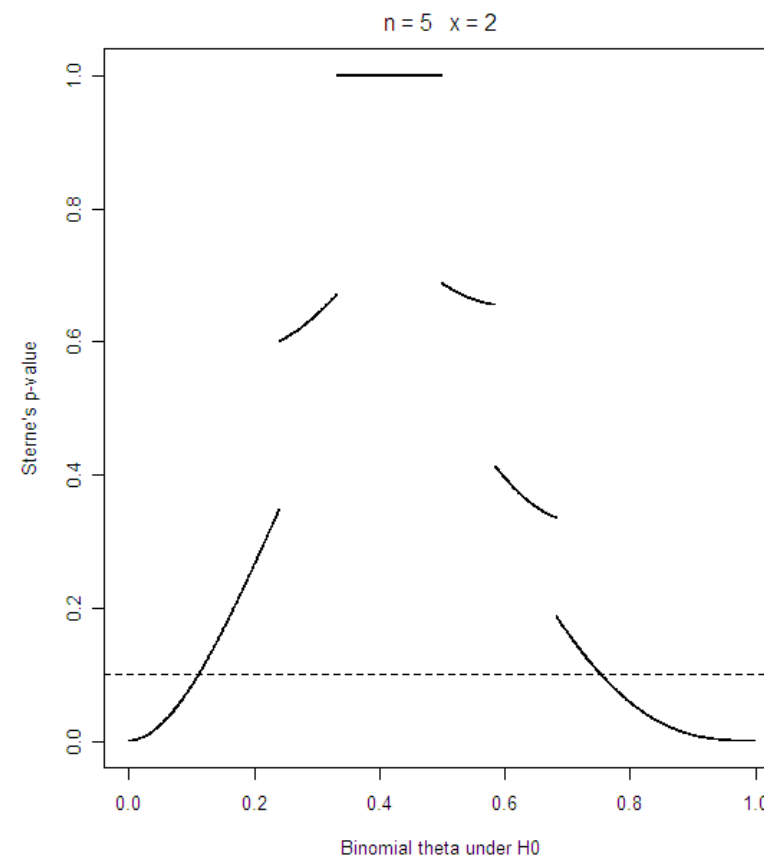
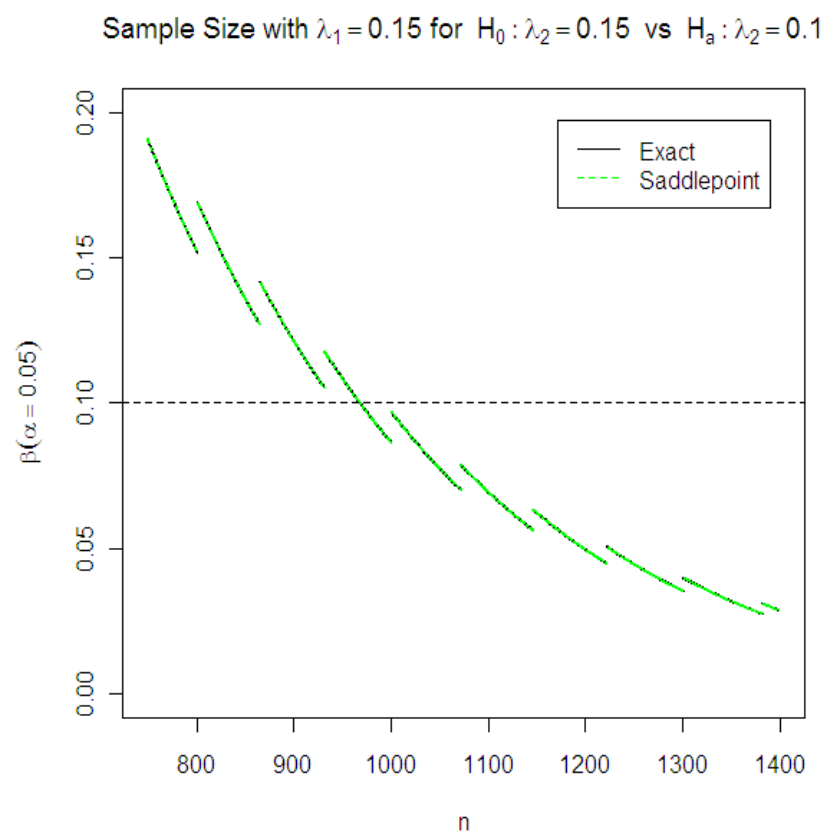
PROC FCMP qskellam code

```
PROC FCMP OUTLIB=sasuser.FCMP_Cat.fcns INLIB=sasuser.FCMP_Cat;
FUNCTION qskellam(p, lambda1, lambda2);
    /* start with special cases that the algorithm cannot handle */
    IF p<0 OR p>1 OR lambda1<0 OR lambda2<0 THEN RETURN(.);
    IF p=0 THEN RETURN(IF lambda2=0 THEN 0 ELSE .);
    IF p=1 THEN RETURN(IF lambda1=0 THEN 0 ELSE .);
    IF lambda1=0 AND lambda2=0 THEN RETURN(0); /* degenerate Poisson */
    IF lambda2=0 THEN RETURN(QUANTILE('POISSON',p,lambda1));
    IF lambda1=0 THEN RETURN(-QUANTILE('POISSON',1-p,lambda2));
    /* Cornish-Fisher approximations */
    z = PROBIT(p);
    mu = lambda1-lambda2;
    vr = lambda1+lambda2;
    sg = SQRT(vr);
    c0 = mu+z*sg;
    c1 = (z**2-1)*mu/vr/6;
    c2 = -( c1*mu - 2*lambda1*lambda2*(z**2-3)/vr )*z/12/vr/sg;
    /* test and linear search (slow if p extreme or lambda1+lambda2 small) */
    q0 = ROUND(c0+c1+c2);
    p = p*(1-2**-46); /* fuzz factor */
    /* find smallest x such that F(x) >= p */
    up = pskellam(q0,lambda1,lambda2) < p;
    DO WHILE (pskellam(q0,lambda1,lambda2) < p);
        q0 = q0+1;
    END;
    DO WHILE ((^up) AND (pskellam(q0-1,lambda1,lambda2) > p));
        q0 = q0-1;
    END;
    RETURN(q0);
ENDSUB;
RUN;
```

SOLVE (root finding function)

```
PROC FCMP;  
  FUNCTION quad(x);  
    pi = 3.1415926535897932;  
    RETURN( (x-pi)*(x-0.5*pi) );  
  ENDSUB;  
  root = SOLVE("quad",{.},0,.);  
  /* 9 iterations, off by 1 in 11th figure */  
  PUT root=;  
  /* find the other root */  
  array solvopts[1] initial (4);  
  root = SOLVE("quad",solvopts,0,.);  
  PUT root=;  
RUN;
```

SOLVE's Target Must be Attainable



Special Functions in FCMP

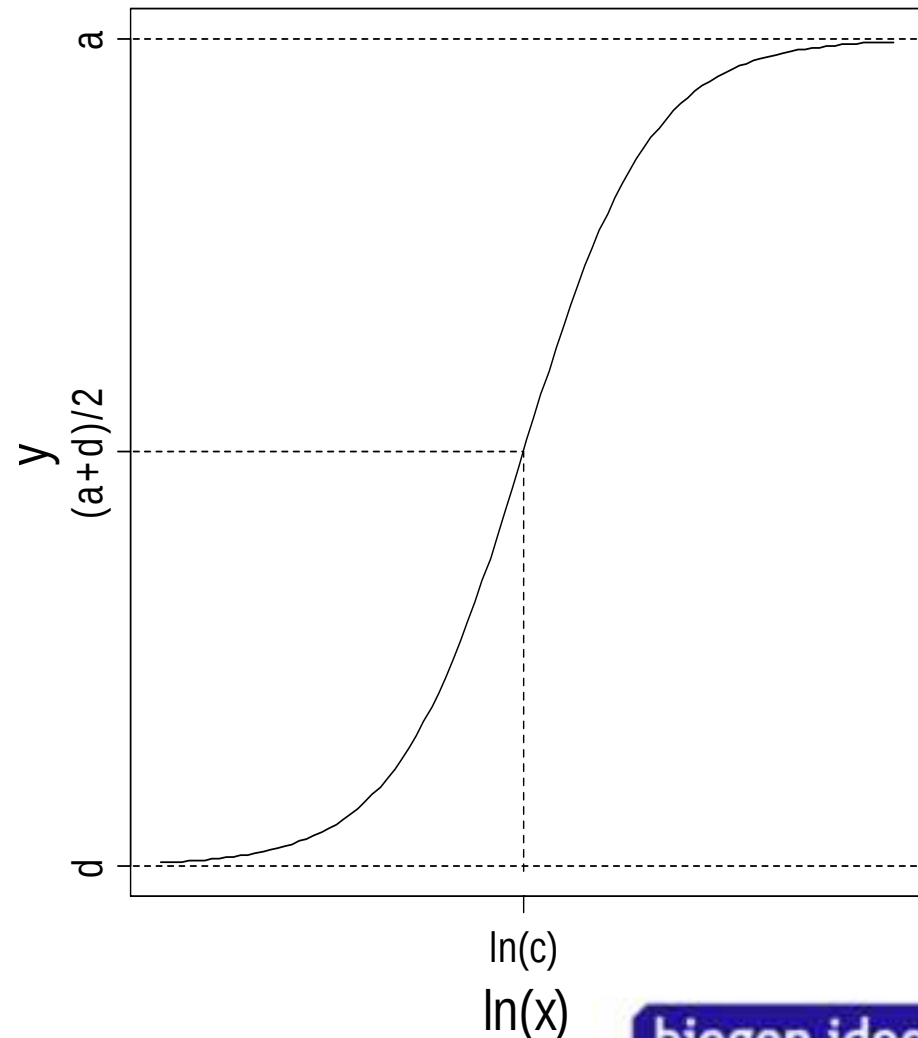
- Matrix CALL routines (subset of IML)
- C Helper functions and CALL routines
- SOLVE root finding function
- Dynamic array resizing
- Call SAS code (macros, data steps, functions) within FCMP routines
- Traverse directory hierarchy
- Excel equivalent functions

Conditionally Linear Models

Example: 4PL bioassay model

$$y = a + \frac{d - a}{1 + \left(\frac{x}{c}\right)^b}$$

- Good starting values are critical for PROC NLIN
- Logit linear in $\{b, c\}$ given $\{a, d\}$
$$\ln\left(\frac{y-a}{d-y}\right) = -b\{\ln(x) - \ln(c)\}$$
- Linear in $\{a, d\}$ given $\{b, c\}$, i.e.
$$RSS(b, c, \hat{a}, \hat{d}) \leq RSS(b, c, a, d)$$
- Could do more stable 2D optimization if SAS understood conditional linearity ('plinear' algorithm in R / S-PLUS)
- With FCMP dimension reduction in PROC NLIN should be possible



PROC FCMP 4PL Code

```
PROC FCMP OUTLIB=sasuser.FCMP_Cat.plinear;
FUNCTION plin4PL(x,b,c,dsNam$);
  /* 4PL (4-parameter logistic) model fn reduced
     to 2D by conditional linearity */
  /* in 4PL model, a & d are linear (closed form
     LS estimates) given b & c */
  ARRAY xx[8,1] / NOSYMBOLS;
  ARRAY yy[8,1] / NOSYMBOLS;
  ARRAY temp[8,1];
  ARRAY Xmat[8,2]; ARRAY Xp[2,8];
  ARRAY XpX[2,2]; ARRAY Xpy[2,1];
  ARRAY XpXinv[2,2]; ARRAY ad[2,1];
  rc = READ_ARRAY(dsNam,xx,'x');
  rc = READ_ARRAY(dsNam,yy,'y');
  DO i=1 TO DIM1(Xmat);
    temp[i] = (xx[i]/c)**b;
    Xmat[i,1] = temp[i]/(1+temp[i]); /* a */
    Xmat[i,2] = 1/(1+temp[i]); /* d */
  END;
  /* set up and solve Normal Equations for a & d
     given b & c */
  CALL TRANSPOSE(Xmat,Xp);
  CALL MULT(Xp,Xmat,XpX);
  CALL MULT(Xp,yy,Xpy);
  CALL INV(XpX,XpXinv);
  CALL MULT(XpXinv,Xpy,ad);
  /* extract the a & d estimates and calculate
     yhat given b & c */
  a = ad[1]; d = ad[2];
  return( a+(d-a)/(1+(x/c)**b) );
ENDSUB;
RUN;
```

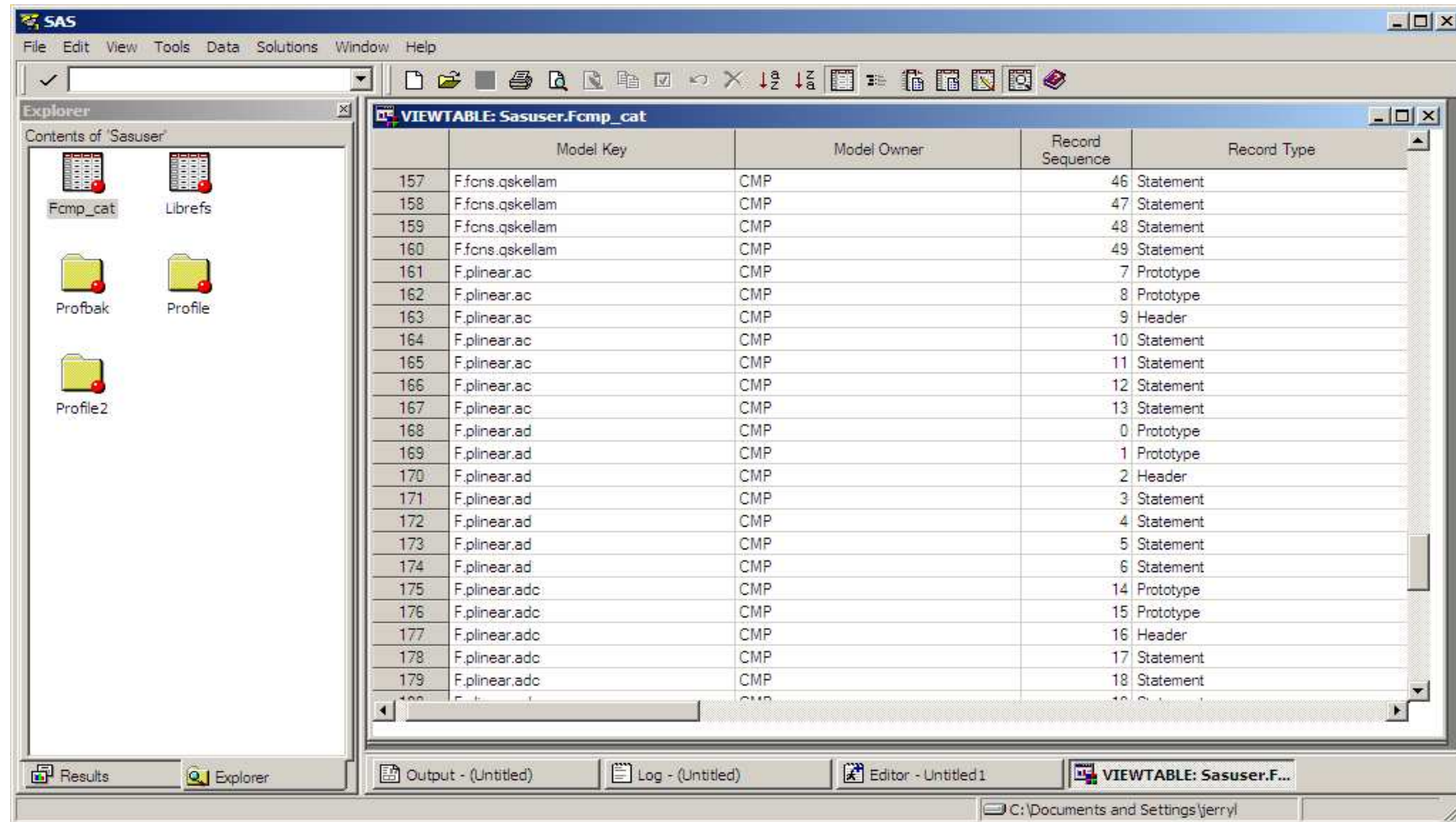
```
/* goal is to use */
PROC NLIN DATA=bioassay;
  PARAMETERS b=1.39 c=3000;
  MODEL y = plin4PL(x,b,c,'bioassay');
RUN;

/* instead of */
PROC NLIN DATA=bioassay;
  PARAMETERS b=1.39 c=3000 d=286 a=747000;
  MODEL y = a+(d-a)/(1+(x/c)**b);
RUN;
```

FCMP Library is a SAS data set

- Routines only exist within the given FCMP instance unless explicitly saved with OUTLIB option
- SAVE with a 3-level name
 - 1st 2 levels ~ permanent SAS dataset
 - 3rd level ~ “package”/“packet” (poorly documented)
- OPTIONS CMPLIB=<dsList>
statement to access FCMP compiled functions outside of the PROC
- INLIB <dsList>
FCMP option to link previously compiled functions
- Both CMPLIB and INLIB use only 2-level (dataset) names
- Use LIBREF to define physical location of 1st level of name
- Name collisions between datasets resolved by order in CMPLIB list (**priority is right to left !!!**)
- IF there are name collisions between packages/packets within a dataset, only the last (alphabetically) package/packet routine will be accessible in DATA step, though pkg.routine may work elsewhere
- You can replace some members in package/packet without disturbing the others

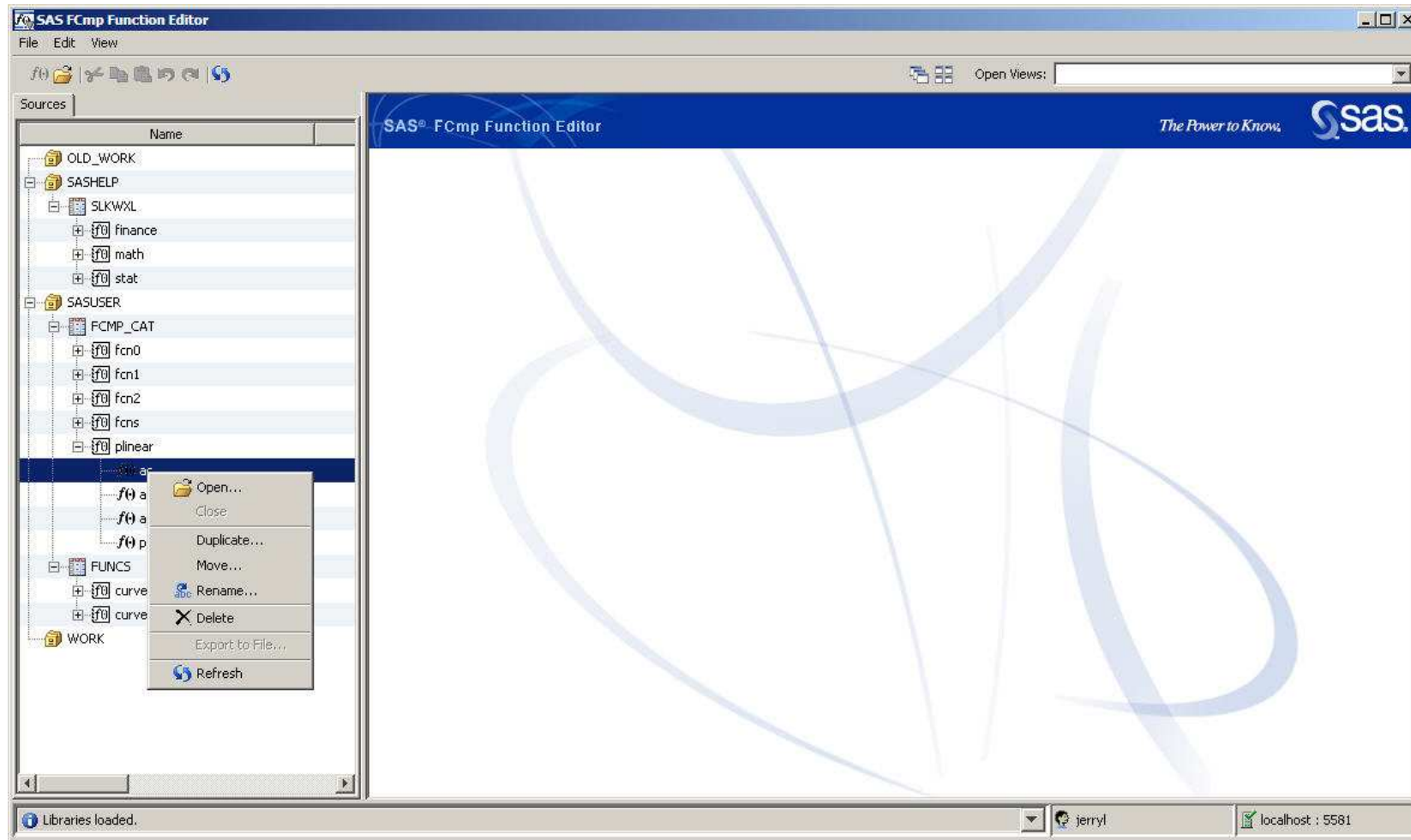
FCMP Functions are Stored in SAS Data Sets



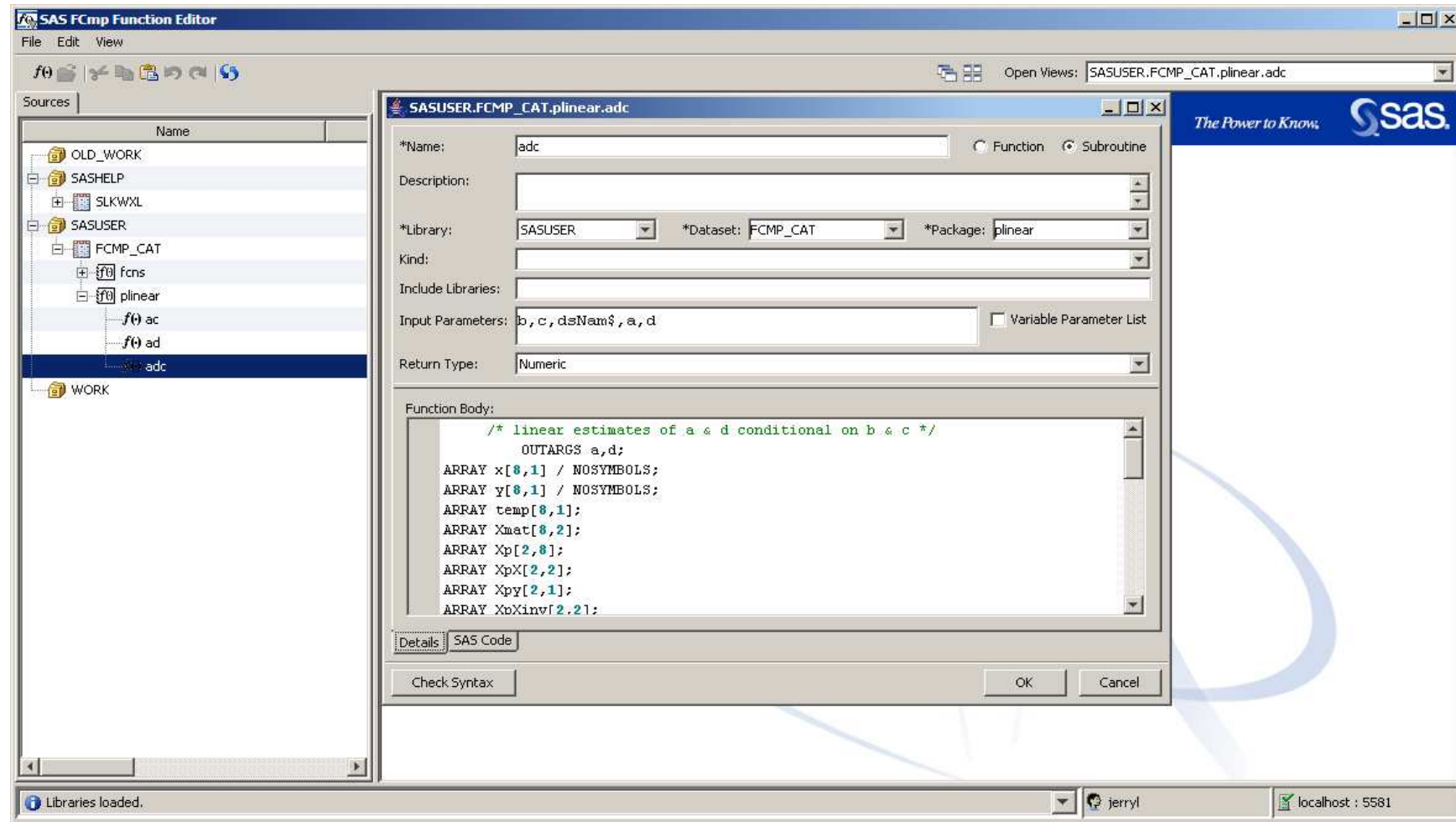
The screenshot shows the SAS software interface. On the left, the Explorer window displays the contents of the 'Sasuser' directory, including folders 'Fcmp_cat', 'Librefs', 'Profbak', 'Profile', and 'Profile2'. The main window displays a table titled 'VIEWTABLE: Sasuser.Fcmp_cat'. The table has five columns: 'Model Key', 'Model Owner', 'Record Sequence', and 'Record Type'. The table lists various FCMP functions and their associated record types.

	Model Key	Model Owner	Record Sequence	Record Type
157	F.fcn.sqskellam	CMP	46	Statement
158	F.fcn.sqskellam	CMP	47	Statement
159	F.fcn.sqskellam	CMP	48	Statement
160	F.fcn.sqskellam	CMP	49	Statement
161	F.plinear.ac	CMP	7	Prototype
162	F.plinear.ac	CMP	8	Prototype
163	F.plinear.ac	CMP	9	Header
164	F.plinear.ac	CMP	10	Statement
165	F.plinear.ac	CMP	11	Statement
166	F.plinear.ac	CMP	12	Statement
167	F.plinear.ac	CMP	13	Statement
168	F.plinear.ad	CMP	0	Prototype
169	F.plinear.ad	CMP	1	Prototype
170	F.plinear.ad	CMP	2	Header
171	F.plinear.ad	CMP	3	Statement
172	F.plinear.ad	CMP	4	Statement
173	F.plinear.ad	CMP	5	Statement
174	F.plinear.ad	CMP	6	Statement
175	F.plinear.adc	CMP	14	Prototype
176	F.plinear.adc	CMP	15	Prototype
177	F.plinear.adc	CMP	16	Header
178	F.plinear.adc	CMP	17	Statement
179	F.plinear.adc	CMP	18	Statement

Library Maintenance: FCMP Function Editor



Library Maintenance: FCMP Function Editor 2



Odds & Ends

- OUTARGS defines arguments that can be modified (to return new values) within a SUBROUTINE
- FCMP supports recursive calling of routines
- Throwing errors – no facility for throwing a native SAS error statement, though you can PUT failure information
- Where PUT writes depends on when it is called:
 - Inside PROC FCMP, PUT writes to listing
 - Outside PROC FCMP, PUT writes to log
 - FILE LOG; or FILE PRINT; can modify this behavior
- Editor syntax highlighting does not always work properly. After editing, correct syntax may be highlighted in red even though there is no error.
- Line numbers in legitimate error messages may not agree with log line numbers

The Good,

- Ability to modularize programming tasks, making programs easier to read, write and modify.
- Familiar SAS syntax.
- All variables are local unless specified in OUTARGS.
- Supports shared libraries of routines and does not seem to have record locking issues in use.

the Bad,

- Documentation does not adequately address many practical issues.
- No support for optional arguments with default values.
- READ_ARRAY must be hard-coded for column names (reportedly fixed in next release, ~ Q1 2011)
- SOLVE does not fully support finding roots of functions with discontinuities (may be addressed in a future release).
- No hooks to SAS error reporting routines.
- Difficult to quickly reach people knowledgeable in FCMP through SAS's HELP line.

and the Ugly

- Syntax highlighting does not work properly.
- Line numbers in legitimate error messages may be wrong.
- Some messages to log are misleading
- Inability to resolve name collisions between packages/packets
- May not be able to use FCMP Function Editor off-line
- Library maintenance tasks seem poorly thought out initially (get TS2M2 if you haven't already!).

Summary

- PROC FCMP is a welcome addition that has the potential for many uses.
- Documentation is somewhat sketchy, so you need to play with it.
- There remains considerable room for improvement in future versions.

Questions/Discussion?

Jerry W. Lewis, PhD
Biogen Idec
14 Cambridge Center
Cambridge, MA 02143
Jerry.Lewis@BiogenIdec.com