

**Multivariate Empirical (MVE) Probability Distributions In Farm-Level Risk
Assessment and Policy Analysis: An Implementation of SAS Procedures**

By

Sung Chul No and Michael E. Salassi



**Staff Report No. 2004-09
July 2004**

**Department of Agricultural Economics and Agribusiness
Louisiana State University Agricultural Center
Baton Rouge, LA**

Multivariate Empirical (MVE) Probability Distributions In Farm-Level Risk Assessment and Policy Analysis: An Implementation of SAS Procedures

Sung Chul No and Michael E. Salassi¹

In their seminal paper, Richardson, Klose, and Gray (2000) described theoretical and procedural aspects of simulating multivariate non-normally distributed random variables using limited historical data. They first presented how to estimate parameters for a MVE probability distribution and simulate a MVE probability distribution with the estimated parameters. Then, they showed a numerical application of the MVE distribution in farm-level risk assessment and policy analysis in use of Simetar©, an add-in program to Microsoft Excel.

The purpose of this report is to demonstrate how SAS users adopt the estimation and simulation procedures of multivariate empirical (MVE) probability distributions demonstrated in the Richardson-Klose-Gray (RKG) paper. In particular, this report emphasizes procedural aspects so that SAS users are able to not only replicate their empirical results but also address similar empirical problems of interest.

RKG identified several problems as critical in modeling farm-level simulation for risk assessment and policy analysis. These include: Non-normally distributed random yields and prices, intra-temporal correlation of production across enterprises and fields, intra- and inter-temporal correlation of output prices², and heteroscedasticity of random variables over time due to policy changes. In addition, a simulation modeler faces a problem of having few observations on several enterprises in analysis of a farm.

To make these problems to be more specific, consider a simulation modeler faced with the analysis of a farm that has four enterprises (i.e., corn, soybean, wheat, and sorghum) with ten years of yield history. Simulation of variability of total revenues generated one, two, and three years ahead is of practical interest. However, it is impractical to parameterize an eight-variable probability distribution with ten observations. In this situation, a preferable approach is to construct an empirical distribution³ defined by the ten available observations. Assuming the data are distributed empirically avoids enforcing a specific distribution on the variables and does not limit the ability of the model to deal with correlation and heteroscedasticity.

In order for random variables generated from simulation to have the same autocorrelations (inter-temporal) and heteroscedasticity (intra-temporal) as historical data, complete simulation of a MVE probability distribution involves two major procedures: (1) Parameter estimation for a MVE probability distribution and (2) simulation of a MVE probability based on the estimated parameters. Parameter estimation, in turn, includes a sequential several steps shown in Figure 1. Simulation follows a six step procedure depicted in Figure 2. Each of simulation steps is carried

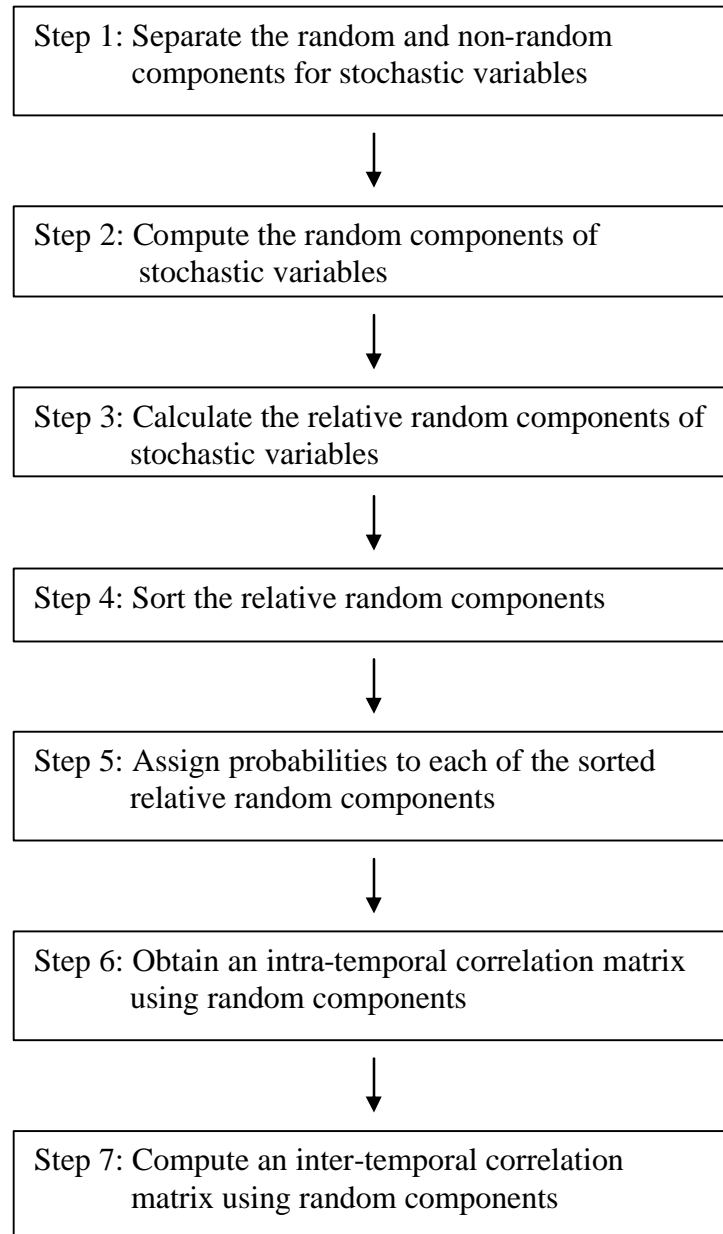
¹ Assistant Professor at Southern University and Professor at Louisiana State University, Baton Rouge, Louisiana.

² The RKG termed autocorrelation and cross-correlation among variables as inter-temporal correlation and intra-temporal correlation, respectively.

³ Law and Kelton provide an overview of the F(x) function for an empirical distribution and the inverse transform method of simulating from the F(x) for an empirical distribution.

out using SAS/IML. Below are a brief procedural comment on each step, SAS subroutines, and outputs. In Appendix, this report contains complete SAS codes with detailed annotations.

Figure 1. Parameter Estimation for a MVE Probability Distribution



Parameter Estimation for a MVE Probability Distribution

- The first step is to eliminate non-random component from the stochastic variables. The SAS procedure, PROC MEANS is used to separate the random and non-random components for each of the stochastic variables, as shown in the following code (Note that all SAS codes in upper cases are SAS default languages; in lower cases are SAS users' languages except ones inside comment delimiters (/*, */)):

```

30 * ===== *;
31 * PARAMETER ESTIMATION FOR A PROBABILITY DISTRIBUTION: SIX STEPS *;
32 * ===== *;
33 * STEP 1: Seperate the random and non-random *;
34 PROC MEANS MEAN STD CV VARDEF=N MIN MAX;
35 VAR cryld sbyld whyld sgyld crprc sbprc whprc sgprc;
36 OUTPUT OUT = meanout MEAN = m1 m2 m3 m4 m5 m6 m7 m8;
  :
43 END;

```

- The second step is to compute the random components of stochastic variables by subtracting the sample mean from the historical values. The SAS statement, SET is used and the variables, “e1” through “e8” contain the random components shown in column of 4 of Table 2.

```

44 * STEP 2: Compute the random components *;
45 DATA three;
46 SET two;
47 e1=cryld - m1; e2=sbyld - m2; e3=whyld - m3; e4=sgyld - m4;

```

Table 2: Steps for Estimating the Parameters for an Empirical Distribution for Variable Three

TAB2						
Observation	Variable	Nonrandom Comp.	Random Comp.	Relative Var.	Sorted Deviates	Probability
O1	48	53.3	-5.3	-0.09944	-0.13696	0.05
O2	46	53.3	-7.3	-0.13696	-0.09944	0.15
O3	48	53.3	-5.3	-0.09943	-0.09944	0.25
O4	54	53.3	0.7	0.01313	-0.09944	0.35
O5	65	53.3	11.7	0.21951	-0.06191	0.45
O6	52	53.3	-1.3	-0.02439	-0.06191	0.55
O7	50	53.3	-3.3	-0.06191	-0.02439	0.65
O8	48	53.3	-5.3	-0.09944	0.01313	0.75
O9	72	53.3	18.7	0.35084	0.21951	0.85
O10	50	53.3	-3.3	-0.06191	0.35084	0.95

- The third step is to calculate the relative random components of stochastic variables by taking the ratio of random components to the sample mean shown in column of 5 of Table 2.

```

53 * STEP 3: Calculate relative random components *;
54 d1=e1/m1; d2=e2/m2; d3=e3/m3; d4=e4/m4;
  :
58 RUN;

```

- The fourth step is to sort the relative random components to assign an empirical probability to each component. The SAS statement, SORT is used and the SAS dataset, “five” includes the sorted relative random components shown in column of 6 of Table 2.

```

59 * STEP 4: Sort the relative random components *;
60 PROC SORT DATA=three OUT=sd1;
61 BY d1;
62 RUN;
63 DATA stdat1;
64 SET sd1; std1 = d1;
65 KEEP std1;
66 RUN;
:
115 RUN;
116 DATA five;
117 MERGE stdat1 stdat2 stdat3 stdat4 stdat5 stdat6 stdat7 stdat8;
118 KEEP std1 std2 std3 std4 std5 std6 std7 std8;
119 RUN;

```

- The fifth step is to assign an empirical probability to each random component generated. The probability of occurrence is included in column of 7 of Table 2.

```

120 * STEP 5: Assign probabilities *;
121 DATA five;
122 MERGE three five;
:
136 max7=max7*1.000001; max8=max8*1.000001;
137 RUN;
:

```

- The sixth step is to calculate the M x M intra-temporal correlation matrix (ie., M=8). To accomplish this task, PROC CORR is used and the estimated matrix is shown in Table 3.1.

```

138 * STEP 6: Compute intra-temporal correlation matrix *;
139 PROC CORR DATA=one NOMISS NOPRINT OUTP=corrout1;
140 VAR cryld sbyld whyld sgyld
141 crprc sbprc whprc sgprc;
142 RUN;

```

Table 3.1: Intra - Temporal Correlation Matrix

	CORR							
	CornY	SoybeansY	WheatY	SorghumY	CornP	SoyP	WheatP	SorghumP
CornY	1	0.5826	-0.3793	0.4829	-0.3111	-0.3977	-0.3591	-0.3162
SoybeansY	0.5826	1	0.1621	0.4597	-0.0856	-0.0164	-0.0796	-0.2588
WheatY	-0.3793	0.16211	1	-0.0744	-0.0797	0.2014	0.2297	-0.3065
SorghumY	0.4829	0.4597	-0.0744	1	-0.0062	0.0726	-0.2146	-0.1445
CornP	-0.3111	-0.0856	-0.0797	-0.0062	1	0.7274	0.7489	0.9252
SoyP	-0.3977	-0.0164	0.2014	0.0727	0.7274	1	0.6584	0.4899
WheatP	-0.3591	-0.0796	0.2297	-0.2146	0.7490	0.6584	1	0.5656
SorghumP	-0.3162	-0.2588	-0.3065	-0.1445	0.9252	0.4899	0.5655845	1

- The seventh step is to calculate the inter-temporal correlation coefficients for the random variables. Current and one-period lagged random components are used to estimate the inter-temporal correlation coefficients, using PROC CORR. The estimated coefficients are reported in Table 3.2.

```

143 * STEP 7: Compute inter-temporal correlation matrix *;
144 PROC CORR DATA=three NOMISS NOPRINT OUTP=corrout2;
145 VAR e1 e11 e2 e12 e3 e13 e4 e14
:
:
149 SET corrout2;
150 RUN;

```

Table 3.2: Inter - Temporal Correlation Coefficient

AUTOCORR	
CornY	0.0536
SoybeansY	-0.1810
WheatY	-0.1187
SorghumY	0.0132
CornP	0.1531
SoybeansP	0.1426
WheatP	0.4231
SorghumP	-0.1577

In order for simulated values to capture the same intra- and inter-correlation properties as the historical data, the square root of the intra-temporal correlation matrix and each of the inter-temporal correlation matrices must be calculated. To achieve this objective, the SAS/IML is invoked. The IML reads each of unsorted random components (i.e., “d1”-“d8”) into “d11” through “d88”, respectively and makes a 10x8 design matrix (“reld”), using horizontal concatenation. The IML function, ROOT returns a square root of intra-temporal correlation matrix (an 8x8 matrix) for the prices and yields of corn, soybeans, wheat, and sorghum.

```

151 * PRE-STEPS FOR SIMULATION: obtain the square root of intra and inter correlation matrix *;
152 PROC IML symsize = 100000000; * invoke iml *;
153 START rkgsim; * start module *;
154 USE five; * open SAS dataset *;
:
:
163 READ ALL VAR{d1} INTO d11; * read SAS variables into IML*;
:
:
191 reld=d11||d22||d33||d44||d55||d66||d77||d88; * "reld" is horizontal concatenate *;
:
:
220 t=NROW(reld); * t = number of rows in "reld" *;
221 sum=reld[+,]; * sum of rows in "sum" *;
222 xpx=reld`*reld-sum`*sum/t;
223 s=DIAG(1/SQRT(VECDIAG(xpx)));
224 corr=s*xpx*s; * correlation matrix across variables
*;
234 cholcross=ROOT(corr); * compute a square root of the intra-correlation matrix *;

```

Similarly, the IML reads the corresponding autocorrelation coefficients of the random components (i.e., “e1”, “e11”, “e2”, “e12”, etc.) into “cor1” – “cor16”. For one, two, and three period-ahead simulations, a 3x3 symmetric matrix is constructed for each of eight variables as follows:

$$(1) \quad \begin{bmatrix} 1 & AR(1) & 0 \\ AR(1) & 1 & AR(1) \\ 0 & AR(1) & 1 \end{bmatrix}$$

where AR(1) is an estimate of a first-order autoregressive process for the corresponding variables. For instance, AR(1) = 0.0536 for corn price as shown in Table 3.2. Then, ROOT returns a square root of inter-temporal correlation matrix of (1) above for each of eight variables.

```

235 USE cor2; * open a SAS dataset *;
236 READ ALL VAR{e1} INTO cor1; * read "cryld" into "cor1" *;
237 READ ALL VAR{e11} INTO cor2;
:
:
252 auto1a=cor2[4,]; auto2a=cor4[6,]; * extract data from "cor2" *;
:
:
265 i1a=I(3); * generate an 3x3 identity matrix *;
266 shaper=SHAPE({0 1 0 1 0 1 0 1 0},3,3); * create a 3x3 matrix with 0s and 1s *;
267 auto1ma=(shaper*auto1a)+i1a; * make a composite matrix *;
:
:
274 auto8ma=(shaper*auto8a)+i1a;
275 chol1reva = ROOT(auto1ma); * create a square root of inter-correlation matrix *;
:
:
282 chol8reva = ROOT(auto8ma); * create a square root of inter-correlation matrix *;

```

Simulation of a MVE Probability Distribution

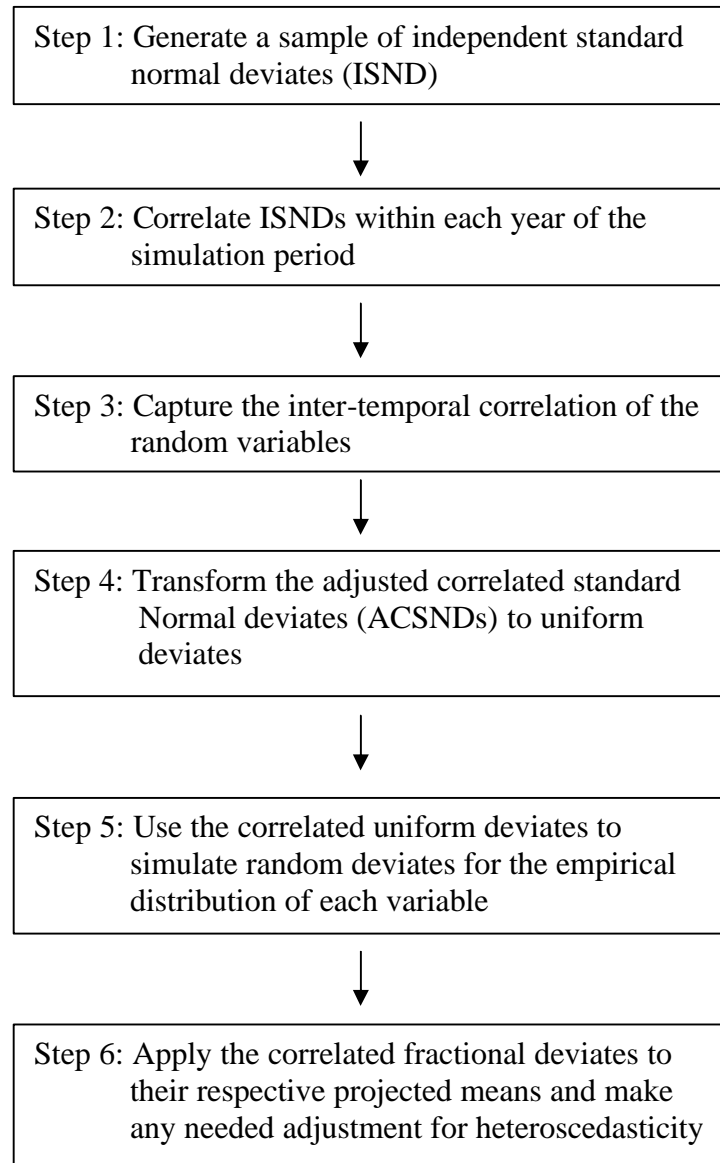
- The first step is to generate a sample of independent standard normal deviates (ISND). The number of ISDNs generated must equal the number of random variables. In this example, 24 ISDNs are needed for eight variables and three years. NORMAL is used to generate 24 random deviates with mean zero and variance one.

```

283 * ===== *;
284 * SIMULATION OF A MVE PROBABILITY DISTRIBUTION: SIX STEPS *;
285 * ===== *;
286 * STEP 1: Generate a sample of independent standard normal deviates (ISND) *;
287 n=4000; * replication n times *;
288 SEED = 12345678; * seed number *;
289 z=J(24,2,0); * create a storage matrix *;
290 CREATE one FROM z;
291 DO rep = 1 TO n; * Start do-loop *;
292 x=J(8,3,SEED); * create a storage matrix *;
293 x=NORMAL(x); * random generator with mean=0 and variance=1 *;
294 isnyr1a_1=x[1:8,1]; * create a 8x1 matrix for one year ahead*;
295 isnyr2a_1=x[1:8,2]; * create a 8x1 matrix for two year ahead*;
296 isnyr3a_1=x[1:8,3]; * create a 8x1 matrix for three year ahead*;

```

Figure 2. Simulation of a MVE Probability Distribution



- The second step is to correlate the ISNDs within each year of the simulation period ($t+k$, $k=1,2$, and 3) by multiplying the factored correlation matrix⁴ and the eight values in the ISND vector. The matrix multiplication is repeated once for each year (k) to be simulated, using the same matrix each time but a different set of eight ISNDs. The resulting eight values (ie., “csndcryld_1” ... “csndsgprc_1”) in each of three years of vectors are intra-temporally correlated standard normal deviates (CSNDs).

⁴ The factored correlation matrix refers to a square root of correlation matrix or Cholesky decomposed correlation matrix. In econometric literature, the term, Cholesky decomposition is more popular one.

```

297 * STEP 2: Correlate ISNDs *;
298 csndyr1_1= cholcross*isnyr1a_1; * create a 8x1 matrix *;
299 csndyr2_1= cholcross*isnyr2a_1;
300 csndyr3_1= cholcross*isnyr3a_1;
301 csndcryld1_1=csndyr3_1[1,]; * extract a single number from a 302 matrix *;
302 csndcryld2_1=csndyr2_1[1,];
303 csndcryld3_1=csndyr1_1[1,];
304 csndcryld_1=csndcryld1_1//csndcryld2_1//csndcryld3_1; * a 3x1 vertical concat*;
:
:
332 csndsgprc_1=csndsgprc1_1//csndsgprc2_1//csndsgprc3_1;

```

- The third step is to capture the inter-temporal correlation of the random variables. The values in the three 8 x 1 vectors of CSNDs are used in a second matrix multiplication to add the inter-temporal correlation to each random variable.

```

333 * Step 3: Capture the inter and intra-temporal correlation *;
334 acsdcryld_1 = chol1reva*csndcryld_1; * create a 3x1 matrix *;
:
:
341 acsdsgprc_1 = chol8reva*csndsgprc_1;

```

- The fourth step is to transform the ACSNDs from third step to uniform deviates. PROBNORM is used to integrate the standard normal distribution from minus infinity to the ACSND_i.

```

342 * Step 4: Transform the adjusted normal deviates to uniform deviates *;
343 cumcryld1_1 = PROBNORM(acsdcryld_1[1,]); * create correlated uniform deviates *;
344 cumcryld2_1 = PROBNORM(acsdcryld_1[2,]);
345 cumcryld3_1 = PROBNORM(acsdcryld_1[3,]);
:
:
371 cumsgprc1_1 = PROBNORM(acsdsgprc_1[1,]);
372 cumsgprc2_1 = PROBNORM(acsdsgprc_1[2,]);
373 cumsgprc3_1 = PROBNORM(acsdsgprc_1[3,]);

```

- The fifth step is to use the correlated uniform deviates generated from the fourth step to simulate random deviates for the empirical distribution of each variable Y_i, which requires a simple interpolation method. The resulting random deviates are appropriately correlated fractional deviates (CFD_i).

```

374 * Step 5: Interpolation of an emprical distribution *;
375 IF cumcryld1_1 >= 0.00 THEN IF cumcryld1_1 <= 0.05 THEN DO; * CORN YIELD: YEAR1 *;
376 cryld1_1 = ((cumcryld1_1 - pb[,1])*(cr_ylda[,2]-cr_ylda[,1]))/0.05 + cr_ylda[,1]; END;
:
:
467 IF cumsbyld2_1 >= 0.00 THEN IF cumsbyld2_1 <= 0.05 THEN DO; * SOYBEAN YIELD: YEAR 2 *;
:
:
674 IF cumcrprc2_1 >= 0.00 THEN IF cumcrprc2_1 <= 0.05 THEN DO; * CORN PRICE: YEAR 2 *;
:
:
904 IF cumsgprc3_1 >= 0.00 THEN IF cumsgprc3_1 <= 0.05 THEN DO; * SORGHUM PRICE: YEAR 3*;
:
:
924 IF cumsgprc3_1 > 0.85 THEN IF cumsgprc3_1 <= 0.95 THEN DO;
925 sgprc3_1 = ((cumsgprc3_1 - pb[,10])*(sg_prca[,11]-sg_prca[,10]))/0.1 + sg_prca[,10]; END;
926 IF cumsgprc3_1 > 0.95 THEN IF cumsgprc3_1 <= 1.00 THEN DO;
927 sgprc3_1 = ((cumsgprc3_1 - pb[,11])*(sg_prca[,12]-sg_prca[,11]))/0.05 + sg_prca[,11]; END;

```

- The sixth step is to combine the correlated fractional deviates with modelers' prospective projected means for yields and prices for the next three years or with experts' non-sampling information. In this last step, various price and yield scenarios can be simulated. The resulting projected parameters, such as minimum, maximum, and coefficient of variation of total revenue (i.e., price x yield), on average, are less likely to be overstated or understated due to ignoring the correlation among enterprises and across years.

```

926 * Step 6: Apply the correlated fractional deviates to projected means *;
927 cryldt_1=cryld1_1//cryld2_1//cryld3_1;          * create a 3x1 matrix *;
:
936 year1_1 = cryld1_1||sbyld1_1||whyld1_1||
937 sgyld1_1||crprc1_1||sbprc1_1||whprc1_1||sgprc1_1;      * create a 1x8 matrix *;
938 year1_1=year1_1`;          * transpose a 1x8 matrix *;
:
944 year3_1 = cryld3_1||sbyld3_1||whyld3_1||sgyld3_1
945 ||crprc3_1||sbprc3_1||whprc3_1||sgprc3_1;
946 year3_1=year3_1`;          * add projected means for simulation period below *;
947 projval = {118.7  37.2  53.3  43.4
948             1.960  4.520  2.910  3.232
949             121.1  37.9  54.4  44.3
950             2.000  4.710  2.990  3.375
951             123.5  38.7  55.5  45.2
952             2.060  4.870  3.090  3.482};          * add assumed expansion factors *;
953 expf = {1.0  1.0  1.0  1.0
954          1.0  1.0  1.0  1.0
955          1.0  1.0  1.0  1.0
956          1.0  1.0  1.0  1.0
957          1.0  1.0  1.0  1.0
958          1.4  1.4  1.4  1.4};
959 coln33 = {"CornY" "SoybeansY" "WheatY" "SorghumY" "CornP" "SoybeansP"
960 "WheatP" "SorghumP"};          *column name for table 3*;
961 rown33 = {"2000" "2001" "2002"};          *row name for table 3*;
:
988 crop24_1=year1_1//year2_1//year3_1;
989 projval_1=projval`;
990 expff=expf`;
991 cropexp_1 = crop24_1#expff;
992 cropexp1_1 = 1+cropexp_1;
993 ran1_1 = projval_1#cropexp1_1;
:
999 RUN rkgsim;          * run module*;
1000 QUIT;          * quit iml *;
1001 * ===== *
1002 * END OF SIMULATION PROCEDURE *
1003 * ===== *;

```

Lastly, the steps from one through six are repeated 4,000 times. The summary statistics are reported in the following table.

Table 4. Results of Simulating Yields and Prices for Three Years

Variable	Mean	Std Dev	Coeff of Variation	Minimum	Maximum
Corn_Yield1	118.9453902	30.9586715	26.0276346	80.0000000	165.0000000
Soybean_Yield1	37.1790042	7.4076173	19.9241949	26.0000000	47.0000000
Wheat_Yield1	53.2569486	7.7487497	14.5497439	46.0000000	72.0000000
Sorghum_Yield1	44.0865312	20.3084755	46.0650339	5.0000000	75.0000000
Corn_Price1	1.9837896	0.2626959	13.2421260	1.6399353	2.4163298
Soybean_Price1	4.4772691	0.4139921	9.2465309	3.9445957	5.3704404
Wheat_Price1	2.9002130	0.2697367	9.3005818	2.1626139	3.6209282
Sorghum_Price1	3.2138911	0.0146930	0.4571721	3.0573748	3.2442663
Corn_Yield2	122.1550130	32.0159238	26.2092591	81.6175232	168.3361415
Soybean_Yield2	37.8942444	7.4815421	19.7432150	26.4892473	47.8844086
Wheat_Yield2	54.2586939	7.9415331	14.6364252	46.9493433	73.4859287
Sorghum_Yield2	44.5548190	20.6200100	46.2800894	5.1036866	76.5552995
Corn_Price2	2.0230858	0.2667695	13.1862666	1.6734034	2.4656427
Soybean_Price2	4.6614011	0.4311968	9.2503699	4.1104083	5.5961890
Wheat_Price2	2.9768142	0.2576954	8.6567507	2.2220672	3.7204727
Sorghum_Price2	3.3563207	0.0146326	0.4359718	3.2101456	3.3887982
Corn_Yield3	124.9057971	32.1587590	25.7464103	83.2350463	171.6722831
Soybean_Yield3	38.6856087	7.6003050	19.6463368	27.0483871	48.8951613
Wheat_Yield3	55.4548696	8.2003899	14.7875020	47.8986867	74.9718574
Sorghum_Yield3	45.4391596	21.2030628	46.6625329	5.2073733	78.1105991
Corn_Price3	2.0809801	0.3796507	18.2438427	1.5890477	2.7314568
Soybean_Price3	4.7886460	0.6172764	12.8904161	4.0020560	6.1528102
Wheat_Price3	3.0556639	0.3115983	10.1974027	1.9789374	4.1468645
Sorghum_Price3	3.4547990	0.0213968	0.6193353	3.2580986	3.4991040

References

- Richardson, James W., Steven L. Klose, and Allan W. Gray. “An Applied Procedure for Estimating and Simulating Multivariate Empirical (MVE) Probability Distributions in Farm-Level Risk Assessment and Policy Analysis.” *Journal of Agricultural and Applied Economics*, 32(2000):299-315.
- Clements, A.M., Jr., H.P. Mapp, Jr., and V.R. Eidman. *A Procedure for Correlating Events in Farm Firm Simulation Models*. Technical Buletin T-131, Oklahoma Agricultural Experiment Station, August 1971.
- Law, A.M. and W.D. Kelton. *Simulation Modeling and Analysis*. 2nd edition, New York: McGraw-Hill Book Co., 1991.
- SAS Institute Inc. *SAS/STAT Guide for Personal Computes* (V. 8.2). Cary, NC, 1999.
- SAS Institute Inc. *SAS/IML Guide for Personal Computes* (V. 6). Cary, NC, 1985.

APPENDIX. COMPLETE SAS CODES WITH DETAILED ANNOTATIONS.

```

1  * =====*
2  * SAS Program: Simulating Multivariate Empirical (MVE) Probability *
3  * Distributions in Farm-Level Risk Assessment and Policy Analysis, *
4  * SAS Program developed by Sung Chul No and Michael E. Salassi *
5  * *
6  * SAS Program based on procedure by Richardson, Klose, and Gray *
7  * Journal of Agricultural and Applied Economics,32,2:299-315 Aug 2000*
8  * Program Compiler: Sung Chul No, April 23, 2002 *
9  * Filename c:\DEA7.SAS DATE: 7/07/04 *
10 * =====*;
11 DM "LOG; CLEAR; OUTPUT; CLEAR";
12 OPTION LINESIZE=100; OPTION PAGESIZE=175;
13 DATA one;
14 INPUT cryld sbyld whyld sgyld crprc sbprc whprc sgprc;
15     lcryld=LAG(cryld); lsbyld=LAG(sbyld); lwhyld=LAG(whyld);
16     lsgyld=LAG(sgyld); lcrprc=LAG(crprc); lsbprc=LAG(sbprc);
17     lwhprc=LAG(whprc); lsgprc=LAG(sgprc);
18 CARDS;
19 100.0 29.0 48.0 45.0 2.540 7.42 3.72 2.27
20 155.0 38.0 46.0 61.0 2.360 5.69 3.72 2.10
21 165.0 40.0 48.0 55.0 2.280 5.74 2.61 2.12

22 112.0 33.0 54.0 75.0 2.370 5.58 3.00 2.25
23 80.0 28.0 65.0 5.0 2.070 5.56 3.24 1.89
24 109.0 40.0 52.0 37.0 2.500 6.40 3.26 2.31
25 145.0 45.0 50.0 25.0 2.260 5.45 3.45 2.13
26 90.0 26.0 48.0 12.0 3.050 6.76 4.37 2.91
27 117.0 47.0 72.0 60.0 2.710 7.35 4.30 2.24
28 114.0 46.0 50.0 59.0 2.600 6.50 3.45 2.34
29 ;
30 * ===== *;
31 * PARAMETER ESTIMATION FOR A PROBABILITY DISTRIBUTION: SIX STEPS *;
32 * ===== *;
33 * STEP 1: Separate the random and non-random *;
34 PROC MEANS MEAN STD CV VARDEF=N MIN MAX;
35 VAR cryld sbyld whyld sgyld crprc sbprc whprc sgprc;
36 OUTPUT OUT = meanout MEAN = m1 m2 m3 m4 m5 m6 m7 m8;
37 RUN;
38 DATA two;
39 SET meanout;
40 DO i=1 TO 10;
41 MERGE one;
42 OUTPUT;
43 END;
44 * STEP 2: Compute the random components *;
45 DATA three;
46 SET two;
47 e1=cryld - m1; e2=sbyld - m2; e3=whyld - m3; e4=sgyld - m4;
48 e5=crprc - m5; e6=sbprc - m6; e7=whprc - m7; e8=sgprc - m8;
49 e11=LAG(e1); e12=LAG(e2); e13=LAG(e3); e14=LAG(e4);
50 e15=LAG(e5); e16=LAG(e6); e17=LAG(e7); e18=LAG(e8);
51 lcryld =e11; lsbyld=e12; lwhyld=e13; lsgyld=e14;
52 lcrprc=e15; lsbprc=e16; lwhprc=e17; lsgprc=e18;
53 * STEP 3: Calculate relative random components *;
54 d1=e1/m1; d2=e2/m2; d3=e3/m3; d4=e4/m4;
55 d5=e5/m5; d6=e6/m6; d7=e7/m7; d8=e8/m8;

```

```

56   d1L=LAG(d1); d2L=LAG(d2); d3L=LAG(d3); d4L=LAG(d4);
57   d5L=LAG(d5); d6L=LAG(d6); d7L=LAG(d7); d8L=LAG(d8);
58   RUN;
59   * STEP 4: Sort the relative random components *;
60   PROC SORT DATA=three OUT=sd1;
61     BY d1;
62   RUN;
63   DATA stdat1;
64     SET sd1; std1 = d1;
65   KEEP std1;
66   RUN;
67   PROC SORT DATA=three OUT=sd2;
68     BY d2;
69   RUN;
70   DATA stdat2;
71     SET sd2; std2 = d2;
72   KEEP std2;
73   RUN;
74   PROC SORT DATA=three OUT=sd3;
75     BY d3;
76   RUN;
77   DATA stdat3;
78     SET sd3; std3 = d3;
79   KEEP std3;
80   RUN;
81   PROC SORT DATA=three OUT=sd4;
82     BY d4;
83   RUN;
84   DATA stdat4;
85     SET sd4; std4 = d4;
86   KEEP std4;
87   RUN;
88   PROC SORT DATA=three OUT=sd5;
89     BY d5;
90   RUN;
91   DATA stdat5;
92     SET sd5; std5 = d5;
93   KEEP std5;
94   RUN;
95   PROC SORT DATA=three OUT=sd6;
96     BY d6;
97   RUN;
98   DATA stdat6;
99     SET sd6; std6 = d6;
100  KEEP std6;
101  RUN;
102  PROC SORT DATA=three OUT=sd7;
103    BY d7;
104  RUN;
105  DATA stdat7;
106    SET sd7; std7 = d7;
107  KEEP std7;
108  RUN;
109  PROC SORT DATA=three OUT=sd8;
110    BY d8;
111  RUN;

```

```

112 DATA stdat8;
113   SET sd8; std8 = d8;
114 KEEP std8;
115 RUN;
116 DATA five;
117   MERGE stdat1 stdat2 stdat3 stdat4 stdat5 stdat6 stdat7 stdat8;
118   KEEP std1 std2 std3 std4 std5 std6 std7 std8;
119 RUN;
120 * STEP 5: Assign probabilities *;
121 DATA five;
122 MERGE three five;
123 RUN;
124 PROC MEANS DATA =five MIN MAX;
125   VAR d1 d2 d3 d4 d5 d6 d7 d8 ;
126   OUTPUT OUT=minmaxout MIN=min1 min2 min3 min4 min5 min6 min7 min8
127   MAX=max1 max2 max3 max4 max5 max6 max7 max8 ;
128 DATA six;
129   SET minmaxout;
130     min1=min1*1.000001; min2=min2*1.000001; min3=min3*1.000001;
131     min4=min4*1.000001; min5=min5*1.000001; min6=min6*1.000001;
132     min7=min7*1.000001; min8=min8*1.000001;
133
134     max1=max1*1.000001; max2=max2*1.000001; max3=max3*1.000001;
135     max4=max4*1.000001; max5=max5*1.000001; max6=max6*1.000001;
136     max7=max7*1.000001; max8=max8*1.000001;
137 RUN;
138 * STEP 6: Compute intra-temporal correlation matrix *;
139 PROC CORR DATA=one NOMISS NOPRINT OUTP=corrout1;
140 VAR cryld sbyld whyld sgyld
141     crprc sbprc whprc sgprc;
142 RUN;
143 * STEP 7: Compute inter-temporal correlation matrix *;
144 PROC CORR DATA=three NOMISS NOPRINT OUTP=corrout2;
145 VAR e1 e11 e2 e12 e3 e13 e4 e14
146     e5 e15 e6 e16 e7 e17 e8 e18;
147 RUN;
148 DATA cor2;
149 SET corrout2;
150 RUN;
151 * PRE-STEPS FOR SIMULATION: obtain the square root of intra and inter correlation matrix *;
152 PROC IML symsize = 100000000;                                * invoke iml *;
153 START rkgsim;                                                * start module *;
154 USE five;                                                    * open SAS dataset *;
155 READ ALL VAR{std1} INTO cr_yld;                               * read SAS variables into IML*;
156 READ ALL VAR{std2} INTO sb_yld;
157 READ ALL VAR{std3} INTO wh_yld;
158 READ ALL VAR{std4} INTO sg_yld;
159 READ ALL VAR{std5} INTO cr_prc;
160 READ ALL VAR{std6} INTO sb_prc;
161 READ ALL VAR{std7} INTO wh_prc;
162 READ ALL VAR{std8} INTO sg_prc;
163 READ ALL VAR{d1} INTO d11;                                    * read SAS variables into IML*;
164 READ ALL VAR{d2} INTO d22;
165 READ ALL VAR{d3} INTO d33;
166 READ ALL VAR{d4} INTO d44;
167 READ ALL VAR{d5} INTO d55;

```

```

168 READ ALL VAR{d6} INTO d66;
169 READ ALL VAR{d7} INTO d77;
170 READ ALL VAR{d8} INTO d88;
171 ***Steps for Estimating Variable Parameters for an Empirical Distribution***;
172 ***Example for Variable 3 - WHYLD****;
173 READ ALL VAR{whyld} INTO var3;
174 READ ALL VAR{e3} INTO dev3;
175 READ ALL VAR{m3} INTO mean3;
176 prob = {0.05, 0.15, 0.25, 0.35, 0.45, 0.55, 0.65, 0.75, 0.85, 0.95};
177 tab2 = var3|mean3|dev3|d33|wh_yld|prob;
178 coln = {"Observation" "Variable" "Nonrandom Comp." "Random Comp." "Relative Var."
179         "Sorted Deviates" "Probability"};
180 rown = {o1, o2, o3, o4, o5, o6, o7, o8, o9, o10};
181 PRINT,"=====
182 =====",
183 "Table 2:Steps for Estimating the Parameters for an Empirical Distribution for Variable Three"
184 "=====
185 =====", tab2 [rowname=row n colname=coln],
186 "-----
187 -----";
188 Print /,,;
189 ***End***Steps for Estimating Variable Parameters for an Empirical Distributin***;
190
191 reld=d11||d22||d33||d44||d55||d66||d77||d88; * "reld" is horizontal concatenate *;
192 cr_yld=cr_yld`; sb_yld=sb_yld`; wh_yld=wh_yld`; * transpose variables *;
193 sg_yld=sg_yld`; cr_prc=cr_prc`; sb_prc=sb_prc`;
194 wh_prc=wh_prc`; sg_prc=sg_prc`;
195 mincryld=MIN(cr_yld); minsbyld=MIN(sb_yld); * extract the minium *;
196 minwhyld=MIN(wh_yld); minsgyld=MIN(sg_yld);
197 mincrprc=MIN(cr_prc); minsbprc=MIN(sb_prc);
198 minwhprc=MIN(wh_prc); minsgprc=MIN(sg_prc);
199 maxcryld=MAX(cr_yld); maxsbyld=MAX(sb_yld); * extract the maximum *;
200 maxwhyld=MAX(wh_yld); maxsgyld=MAX(sg_yld);
201 maxcrprc=MAX(cr_prc); maxsbprc=MAX(sb_prc);
202 maxwhprc=MAX(wh_prc); maxsgprc=MAX(sg_prc);
203 mmcryld=INSERT(mincryld,maxcryld,0,2); * create a variable with the min and max *;
204 mmsbyld=INSERT(minsbyld,maxsbyld,0,2);
205 mmwhyld=INSERT(minwhyld,maxwhyld,0,2);
206 mmsgyld=INSERT(minsgyld,maxsgyld,0,2);
207 mmcrprc=INSERT(mincrprc,maxcrprc,0,2);
208 mmsbprc=INSERT(minsbprc,maxsbprc,0,2);
209 mmwhprc=INSERT(minwhprc,maxwhprc,0,2);
210 mmsgprc=INSERT(minsgprc,maxsgprc,0,2);
211 cr_ylda=INSERT(mmcryld,cr_yld,0,2); * insert "cr_yld" into "mmcryld" *;
212 sb_ylda=INSERT(mmsbyld,sb_yld,0,2);
213 wh_ylda=INSERT(mmwhyld,wh_yld,0,2);
214 sg_ylda=INSERT(mmsgyld,sg_yld,0,2);
215 cr_prca=INSERT(mmcrprc,cr_prc,0,2);
216 sb_prca=INSERT(mmsbprc,sb_prc,0,2);
217 wh_prca=INSERT(mmwhprc,wh_prc,0,2);
218 sg_prca=INSERT(mmsgprc,sg_prc,0,2); * creat a probability of occurrencevar below *;
219 pb={0.00 0.05 0.15 0.25 0.35 0.45 0.55 0.65 0.75 0.85 0.95 1.00};
220 t=NROW(reld); * t = number of rows in "reld" *;
221 sum=reld[+,]; * sum of rows in "sum" *;
222 xpx=reld`*reld-sum`*sum/t;
223 s=DIAG(1/SQRT(VECDIAG(xpx)));

```

```

224 corr=s*xpx*s; * correlation matrix across variables *;
225 rown3= {"CornY" "SoybeansY" "WheatY" "SorghumY" "CornP" "SoyP" "WheatP" "SorghumP"};
226 PRINT , "=====
227 =====",
228 "Table 3.1: Intra - Temporal Correlation Matrix",
229 "=====
230 =====", corr [ROWNAME=rown3 COLNAME=rown3],
231 "-----
232 -----";
233 PRINT /,;
234 cholcross=ROOT(corr); * compute a square root of the intra-correlation matrix *;
235 USE cor2; * open a SAS dataset *;
236 READ ALL VAR{e1} INTO cor1; * read "cryld" into "cor1" *;
237 READ ALL VAR{e11} INTO cor2;
238 READ ALL VAR{e2} INTO cor3;
239 READ ALL VAR{e12} INTO cor4;
240 READ ALL VAR{e3} INTO cor5;
241 READ ALL VAR{e13} INTO cor6;
242 READ ALL VAR{e4} INTO cor7;
243 READ ALL VAR{e14} INTO cor8;
244 READ ALL VAR{e5} INTO cor9;
245 READ ALL VAR{e15} INTO cor10;
246 READ ALL VAR{e6} INTO cor11;
247 READ ALL VAR{e16} INTO cor12;
248 READ ALL VAR{e7} INTO cor13;
249 READ ALL VAR{e17} INTO cor14;
250 READ ALL VAR{e8} INTO cor15;
251 READ ALL VAR{e18} INTO cor16;
252 auto1a=cor2[4,]; auto2a=cor4[6,]; * extract data from "cor2" *;
253 auto3a=cor6[8,]; auto4a=cor8[10,];
254 auto5a=cor10[12,]; auto6a=cor12[14,];
255 auto7a=cor14[16,]; auto8a=cor16[18,];
256 autocorr = auto1a//auto2a//auto3a//auto4a//auto5a//auto6a
257 //auto7a//auto8a; * "autocorr" is vertical concant *;
258 rown32 = {"CornY" "SoybeansY" "WheatY" "SorghumY" "CornP" "SoybeansP"
259 "WheatP" "SorghumP"}; * table rowname *;
260 PRINT , "=====
261 "Table 3.2: Inter - Temporal Correlation Coefficient",
262 "=====
263 "-----";
264 PRINT /,;
265 i1a=I(3); * generate an 3x3 identity matrix *;
266 shaper=SHAPE({0 1 0 1 0 1 0 1 0},3,3); * create a 3x3 matrix with 0s and 1s *;
267 auto1ma=(shaper*auto1a)+i1a; * make a composite matrix *;
268 auto2ma=(shaper*auto2a)+i1a;
269 auto3ma=(shaper*auto3a)+i1a;
270 auto4ma=(shaper*auto4a)+i1a;
271 auto5ma=(shaper*auto5a)+i1a;
272 auto6ma=(shaper*auto6a)+i1a;
273 auto7ma=(shaper*auto7a)+i1a;
274 auto8ma=(shaper*auto8a)+i1a;
275 chol1reva = ROOT(auto1ma); * create a square root of inter-correlation matrix *;
276 chol2reva = ROOT(auto2ma); * create a square root of inter-correlation matrix *;
277 chol3reva = ROOT(auto3ma); * create a square root of inter-correlation matrix *;
278 chol4reva = ROOT(auto4ma); * create a square root of inter-correlation matrix *;
279 chol5reva = ROOT(auto5ma); * create a square root of inter-correlation matrix *;

```

```

280 chol6reva = ROOT(auto6ma);          * create a square root of inter-correlation matrix *;
281 chol7reva = ROOT(auto7ma);          * create a square root of inter-correlation matrix *;
282 chol8reva = ROOT(auto8ma);          * create a square root of inter-correlation matrix *;
283 * ===== *;
284 * SIMULATION OF A MVE PROBABILITY DISTRIBUTION: SIX STEPS *;
285 * ===== *;
286 * STEP 1: Generate a sample of independent standard normal deviates (ISND) *;
287 n=4000;!                             * replication n times *;
288 SEED = 12345678;                       * seed number *;
289 z=J(24,2,0);                             * create a storage matrix *;
290 CREATE one FROM z;
291 DO rep = 1 TO n;                          * Start do-loop *;
292   x=J(8,3,SEED);                          * create a storage matrix *;
293   x=NORMAL(x);                             * random generator with mean=0 and variance=1 *;
294   isnyr1a_1=x[1:8,1];                       * create a 8x1 matrix for one year ahead*;
295   isnyr2a_1=x[1:8,2];                       * create a 8x1 matrix for two year ahead*;
296   isnyr3a_1=x[1:8,3];                       * create a 8x1 matrix for three year ahead*;
297 * STEP 2: Correlate ISNDs *;
298   csndyr1_1= cholcross*isnyr1a_1;           * create a 8x1 matrix *;
299   csndyr2_1= cholcross*isnyr2a_1;
300   csndyr3_1= cholcross*isnyr3a_1;
301   csndcryld1_1=csndyr3_1[1,];               * extract a single number from a 3x1 matrix *;
302   csndcryld2_1=csndyr2_1[1,];
303   csndcryld3_1=csndyr1_1[1,];
304   csndcryld_1=csndcryld1_1//csndcryld2_1//csndcryld3_1;           * a 3x1 vertical concat*;
305   csndsbyld1_1=csndyr3_1[2,];
306   csndsbyld2_1=csndyr2_1[2,];
307   csndsbyld3_1=csndyr1_1[2,];
308   csndsbyld_1=csndsbyld1_1//csndsbyld2_1//csndsbyld3_1;
309   csndwhyld1_1=csndyr3_1[3,];
310   csndwhyld2_1=csndyr2_1[3,];
311   csndwhyld3_1=csndyr1_1[3,];
312   csndwhyld_1=csndwhyld1_1//csndwhyld2_1//csndwhyld3_1;
313   csndsgyld1_1=csndyr3_1[4,];
314   csndsgyld2_1=csndyr2_1[4,];
315   csndsgyld3_1=csndyr1_1[4,];
316   csndsgyld_1=csndsgyld1_1//csndsgyld2_1//csndsgyld3_1;
317   csndcrprc1_1=csndyr3_1[5,];
318   csndcrprc2_1=csndyr2_1[5,];
319   csndcrprc3_1=csndyr1_1[5,];
320   csndcrprc_1=csndcrprc1_1//csndcrprc2_1//csndcrprc3_1;
321   csndsbrpc1_1=csndyr3_1[6,];
322   csndsbrpc2_1=csndyr2_1[6,];
323   csndsbrpc3_1=csndyr1_1[6,];
324   csndsbrpc_1=csndsbrpc1_1//csndsbrpc2_1//csndsbrpc3_1;
325   csndwhprc1_1=csndyr3_1[7,];
326   csndwhprc2_1=csndyr2_1[7,];
327   csndwhprc3_1=csndyr1_1[7,];
328   csndwhprc_1=csndwhprc1_1//csndwhprc2_1//csndwhprc3_1;
329   csndsgprc1_1=csndyr3_1[8,];
330   csndsgprc2_1=csndyr2_1[8,];
331   csndsgprc3_1=csndyr1_1[8,];
332   csndsgprc_1=csndsgprc1_1//csndsgprc2_1//csndsgprc3_1;
333 * Step 3: Capture the inter and intra-temporal correlation *;
334   acsdcryld_1 = chol1reva*csndcryld_1;           * create a 3x1 matrix *;
335   acsdsbyld_1 = chol2reva*csndsbyld_1;

```

```

336 acsdwhyld_1 = chol3reva*csndwhyld_1;
337 acsdsgyld_1 = chol4reva*csndsgyld_1;
338 acsdcrprc_1 = chol5reva*csndcrprc_1;
339 acsdsbprc_1 = chol6reva*csndsbsprc_1;
340 acsdwhprc_1 = chol7reva*csndwhprc_1;
341 acsdsgprc_1 = chol8reva*csndsgprc_1;
342 * Step 4: Transform the adjusted normal deviates to uniform deviates *;
343 cumcryld1_1 = PROBNORM(acsdcryld_1[1,]); * create correlated uniform deviates *;
344 cumcryld2_1 = PROBNORM(acsdcryld_1[2,]);
345 cumcryld3_1 = PROBNORM(acsdcryld_1[3,]);
346
347 cumsbyld1_1 = PROBNORM(acsdsbyld_1[1,]);
348 cumsbyld2_1 = PROBNORM(acsdsbyld_1[2,]);
349 cumsbyld3_1 = PROBNORM(acsdsbyld_1[3,]);
350
351 cumwhyld1_1 = PROBNORM(acsdwhyld_1[1,]);
352 cumwhyld2_1 = PROBNORM(acsdwhyld_1[2,]);
353 cumwhyld3_1 = PROBNORM(acsdwhyld_1[3,]);
354
355 cumsgyld1_1 = PROBNORM(acsdsgyld_1[1,]);
356 cumsgyld2_1 = PROBNORM(acsdsgyld_1[2,]);
357 cumsgyld3_1 = PROBNORM(acsdsgyld_1[3,]);
358
359 cumcrprc1_1 = PROBNORM(acsdcrprc_1[1,]);
360 cumcrprc2_1 = PROBNORM(acsdcrprc_1[2,]);
361 cumcrprc3_1 = PROBNORM(acsdcrprc_1[3,]);
362
363 cumsbprc1_1 = PROBNORM(acsdsbprc_1[1,]);
364 cumsbprc2_1 = PROBNORM(acsdsbprc_1[2,]);
365 cumsbprc3_1 = PROBNORM(acsdsbprc_1[3,]);
366
367 cumwhprc1_1 = PROBNORM(acsdwhprc_1[1,]);
368 cumwhprc2_1 = PROBNORM(acsdwhprc_1[2,]);
369 cumwhprc3_1 = PROBNORM(acsdwhprc_1[3,]);
370
371 cumsgprc1_1 = PROBNORM(acsdsgprc_1[1,]);
372 cumsgprc2_1 = PROBNORM(acsdsgprc_1[2,]);
373 cumsgprc3_1 = PROBNORM(acsdsgprc_1[3,]);
374 * Step 5: Interpolation of an empirical distribution *;
375 IF cumcryld1_1 >= 0.00 THEN IF cumcryld1_1 <= 0.05 THEN DO; * CORN YIELD: YEAR1 *;
376 cryld1_1 = ((cumcryld1_1 - pb[,1])*(cr_ylda[,2]-cr_ylda[,1]))/0.05 + cr_ylda[,1]; END;
377 IF cumcryld1_1 > 0.05 THEN IF cumcryld1_1 <= 0.15 THEN DO;
378 cryld1_1 = ((cumcryld1_1 - pb[,2])*(cr_ylda[,3]-cr_ylda[,2]))/0.1 + cr_ylda[,2]; END;
379 IF cumcryld1_1 > 0.15 THEN IF cumcryld1_1 <= 0.25 THEN DO;
380 cryld1_1 = ((cumcryld1_1 - pb[,3])*(cr_ylda[,4]-cr_ylda[,3]))/0.1 + cr_ylda[,3]; END;
381 IF cumcryld1_1 > 0.25 THEN IF cumcryld1_1 <= 0.35 THEN DO;
382 cryld1_1 = ((cumcryld1_1 - pb[,4])*(cr_ylda[,5]-cr_ylda[,4]))/0.1 + cr_ylda[,4]; END;
383 IF cumcryld1_1 > 0.35 THEN IF cumcryld1_1 <= 0.45 THEN DO;
384 cryld1_1 = ((cumcryld1_1 - pb[,5])*(cr_ylda[,6]-cr_ylda[,5]))/0.1 + cr_ylda[,5]; END;
385 IF cumcryld1_1 > 0.45 THEN IF cumcryld1_1 <= 0.55 THEN DO;
386 cryld1_1 = ((cumcryld1_1 - pb[,6])*(cr_ylda[,7]-cr_ylda[,6]))/0.1 + cr_ylda[,6]; END;
387 IF cumcryld1_1 > 0.55 THEN IF cumcryld1_1 <= 0.65 THEN DO;
388 cryld1_1 = ((cumcryld1_1 - pb[,7])*(cr_ylda[,8]-cr_ylda[,7]))/0.1 + cr_ylda[,7]; END;
389 IF cumcryld1_1 > 0.65 THEN IF cumcryld1_1 <= 0.75 THEN DO;
390 cryld1_1 = ((cumcryld1_1 - pb[,8])*(cr_ylda[,9]-cr_ylda[,8]))/0.1 + cr_ylda[,8]; END;
391 IF cumcryld1_1 > 0.75 THEN IF cumcryld1_1 <= 0.85 THEN DO;

```

```

392 cryld1_1 = ((cumcryld1_1 - pb[,9])*(cr_ylda[,10]-cr_ylda[,9]))/0.1 + cr_ylda[,9]; END;
393 IF cumcryld1_1 > 0.85 THEN IF cumcryld1_1 <= 0.95 THEN D0;
394 cryld1_1 = ((cumcryld1_1 - pb[,10])*(cr_ylda[,11]-cr_ylda[,10]))/0.1 + cr_ylda[,10]; END;
395 IF cumcryld1_1 > 0.95 THEN IF cumcryld1_1 <= 1.00 THEN D0;
396 cryld1_1 = ((cumcryld1_1 - pb[,11])*(cr_ylda[,12]-cr_ylda[,11]))/0.05 + cr_ylda[,11]; END;
397
398 IF cumcryld2_1 >= 0.00 THEN IF cumcryld2_1 <= 0.05 THEN D0;          * CORN YIELD: YEAR 2 *;
399 cryld2_1 = ((cumcryld2_1 - pb[,1])*(cr_ylda[,2]-cr_ylda[,1]))/0.05 + cr_ylda[,1]; END;
400 IF cumcryld2_1 > 0.05 THEN IF cumcryld2_1 <= 0.15 THEN D0;
401 cryld2_1 = ((cumcryld2_1 - pb[,2])*(cr_ylda[,3]-cr_ylda[,2]))/0.1 + cr_ylda[,2]; END;
402 IF cumcryld2_1 > 0.15 THEN IF cumcryld2_1 <= 0.25 THEN D0;
403 cryld2_1 = ((cumcryld2_1 - pb[,3])*(cr_ylda[,4]-cr_ylda[,3]))/0.1 + cr_ylda[,3]; END;
404 IF cumcryld2_1 > 0.25 THEN IF cumcryld2_1 <= 0.35 THEN D0;
405 cryld2_1 = ((cumcryld2_1 - pb[,4])*(cr_ylda[,5]-cr_ylda[,4]))/0.1 + cr_ylda[,4]; END;
406 IF cumcryld2_1 > 0.35 THEN IF cumcryld2_1 <= 0.45 THEN D0;
407 cryld2_1 = ((cumcryld2_1 - pb[,5])*(cr_ylda[,6]-cr_ylda[,5]))/0.1 + cr_ylda[,5]; END;
408 IF cumcryld2_1 > 0.45 THEN IF cumcryld2_1 <= 0.55 THEN D0;
409 cryld2_1 = ((cumcryld2_1 - pb[,6])*(cr_ylda[,7]-cr_ylda[,6]))/0.1 + cr_ylda[,6]; END;
410 IF cumcryld2_1 > 0.55 THEN IF cumcryld2_1 <= 0.65 THEN D0;
411 cryld2_1 = ((cumcryld2_1 - pb[,7])*(cr_ylda[,8]-cr_ylda[,7]))/0.1 + cr_ylda[,7]; END;
412 IF cumcryld2_1 > 0.65 THEN IF cumcryld2_1 <= 0.75 THEN D0;
413 cryld2_1 = ((cumcryld2_1 - pb[,8])*(cr_ylda[,9]-cr_ylda[,8]))/0.1 + cr_ylda[,8]; END;
414 IF cumcryld2_1 > 0.75 THEN IF cumcryld2_1 <= 0.85 THEN D0;
415 cryld2_1 = ((cumcryld2_1 - pb[,9])*(cr_ylda[,10]-cr_ylda[,9]))/0.1 + cr_ylda[,9]; END;
416 IF cumcryld2_1 > 0.85 THEN IF cumcryld2_1 <= 0.95 THEN D0;
417 cryld2_1 = ((cumcryld2_1 - pb[,10])*(cr_ylda[,11]-cr_ylda[,10]))/0.1 + cr_ylda[,10]; END;
418 IF cumcryld2_1 > 0.95 THEN IF cumcryld2_1 <= 1.00 THEN D0;
419 cryld2_1 = ((cumcryld2_1 - pb[,11])*(cr_ylda[,12]-cr_ylda[,11]))/0.05 + cr_ylda[,11]; END;
420
421 IF cumcryld3_1 >= 0.00 THEN IF cumcryld3_1 <= 0.05 THEN D0;          * CORN YIELD: YEAR 3 *;
422 cryld3_1 = ((cumcryld3_1 - pb[,1])*(cr_ylda[,2]-cr_ylda[,1]))/0.05 + cr_ylda[,1]; END;
423 IF cumcryld3_1 > 0.05 THEN IF cumcryld3_1 <= 0.15 THEN D0;
424 cryld3_1 = ((cumcryld3_1 - pb[,2])*(cr_ylda[,3]-cr_ylda[,2]))/0.1 + cr_ylda[,2]; END;
425 IF cumcryld3_1 > 0.15 THEN IF cumcryld3_1 <= 0.25 THEN D0;
426 cryld3_1 = ((cumcryld3_1 - pb[,3])*(cr_ylda[,4]-cr_ylda[,3]))/0.1 + cr_ylda[,3]; END;
427 IF cumcryld3_1 > 0.25 THEN IF cumcryld3_1 <= 0.35 THEN D0;
428 cryld3_1 = ((cumcryld3_1 - pb[,4])*(cr_ylda[,5]-cr_ylda[,4]))/0.1 + cr_ylda[,4]; END;
429 IF cumcryld3_1 > 0.35 THEN IF cumcryld3_1 <= 0.45 THEN D0;
430 cryld3_1 = ((cumcryld3_1 - pb[,5])*(cr_ylda[,6]-cr_ylda[,5]))/0.1 + cr_ylda[,5]; END;
431 IF cumcryld3_1 > 0.45 THEN IF cumcryld3_1 <= 0.55 THEN D0;
432 cryld3_1 = ((cumcryld3_1 - pb[,6])*(cr_ylda[,7]-cr_ylda[,6]))/0.1 + cr_ylda[,6]; END;
433 IF cumcryld3_1 > 0.55 THEN IF cumcryld3_1 <= 0.65 THEN D0;
434 cryld3_1 = ((cumcryld3_1 - pb[,7])*(cr_ylda[,8]-cr_ylda[,7]))/0.1 + cr_ylda[,7]; END;
435 IF cumcryld3_1 > 0.65 THEN IF cumcryld3_1 <= 0.75 THEN D0;
436 cryld3_1 = ((cumcryld3_1 - pb[,8])*(cr_ylda[,9]-cr_ylda[,8]))/0.1 + cr_ylda[,8]; END;
437 IF cumcryld3_1 > 0.75 THEN IF cumcryld3_1 <= 0.85 THEN D0;
438 cryld3_1 = ((cumcryld3_1 - pb[,9])*(cr_ylda[,10]-cr_ylda[,9]))/0.1 + cr_ylda[,9]; END;
439 IF cumcryld3_1 > 0.85 THEN IF cumcryld3_1 <= 0.95 THEN D0;
440 cryld3_1 = ((cumcryld3_1 - pb[,10])*(cr_ylda[,11]-cr_ylda[,10]))/0.1 + cr_ylda[,10]; END;
441 IF cumcryld3_1 > 0.95 THEN IF cumcryld3_1 <= 1.00 THEN D0;
442 cryld3_1 = ((cumcryld3_1 - pb[,11])*(cr_ylda[,12]-cr_ylda[,11]))/0.05 + cr_ylda[,11]; END;
443 * SOYBEAN YIELD *;
444 IF cumsbyld1_1 >= 0.00 THEN IF cumsbyld1_1 <= 0.05 THEN D0;          * SOYBEAN YIELD: YEAR 1 *;
445 sbyld1_1 = ((cumsbyld1_1 - pb[,1])*(sb_ylda[,2]-sb_ylda[,1]))/0.05 + sb_ylda[,1]; END;
446 IF cumsbyld1_1 > 0.05 THEN IF cumsbyld1_1 <= 0.15 THEN D0;
447 sbyld1_1 = ((cumsbyld1_1 - pb[,2])*(sb_ylda[,3]-sb_ylda[,2]))/0.1 + sb_ylda[,2]; END;

```

```

448 IF cumsbyld1_1 > 0.15 THEN IF cumsbyld1_1 <= 0.25 THEN DO;
449 sbyld1_1 = ((cumsbyld1_1 - pb[,3])*(sb_ylda[,4]-sb_ylda[,3]))/0.1 + sb_ylda[,3]; END;
450 IF cumsbyld1_1 > 0.25 THEN IF cumsbyld1_1 <= 0.35 THEN DO;
451 sbyld1_1 = ((cumsbyld1_1 - pb[,4])*(sb_ylda[,5]-sb_ylda[,4]))/0.1 + sb_ylda[,4]; END;
452 IF cumsbyld1_1 > 0.35 THEN IF cumsbyld1_1 <= 0.45 THEN DO;
453 sbyld1_1 = ((cumsbyld1_1 - pb[,5])*(sb_ylda[,6]-sb_ylda[,5]))/0.1 + sb_ylda[,5]; END;
454 IF cumsbyld1_1 > 0.45 THEN IF cumsbyld1_1 <= 0.55 THEN DO;
455 sbyld1_1 = ((cumsbyld1_1 - pb[,6])*(sb_ylda[,7]-sb_ylda[,6]))/0.1 + sb_ylda[,6]; END;
456 IF cumsbyld1_1 > 0.55 THEN IF cumsbyld1_1 <= 0.65 THEN DO;
457 sbyld1_1 = ((cumsbyld1_1 - pb[,7])*(sb_ylda[,8]-sb_ylda[,7]))/0.1 + sb_ylda[,7]; END;
458 IF cumsbyld1_1 > 0.65 THEN IF cumsbyld1_1 <= 0.75 THEN DO;
459 sbyld1_1 = ((cumsbyld1_1 - pb[,8])*(sb_ylda[,9]-sb_ylda[,8]))/0.1 + sb_ylda[,8]; END;
460 IF cumsbyld1_1 > 0.75 THEN IF cumsbyld1_1 <= 0.85 THEN DO;
461 sbyld1_1 = ((cumsbyld1_1 - pb[,9])*(sb_ylda[,10]-sb_ylda[,9]))/0.1 + sb_ylda[,9]; END;
462 IF cumsbyld1_1 > 0.85 THEN IF cumsbyld1_1 <= 0.95 THEN DO;
463 sbyld1_1 = ((cumsbyld1_1 - pb[,10])*(sb_ylda[,11]-sb_ylda[,10]))/0.1 + sb_ylda[,10]; END;
464 IF cumsbyld1_1 > 0.95 THEN IF cumsbyld1_1 <= 1.00 THEN DO;
465 sbyld1_1 = ((cumsbyld1_1 - pb[,11])*(sb_ylda[,12]-sb_ylda[,11]))/0.05 + sb_ylda[,11]; END;
466
467 IF cumsbyld2_1 >= 0.00 THEN IF cumsbyld2_1 <= 0.05 THEN DO; * SOYBEAN YIELD: YEAR 2 *;
468 sbyld2_1 = ((cumsbyld2_1 - pb[,1])*(sb_ylda[,2]-sb_ylda[,1]))/0.05 + sb_ylda[,1]; END;
469 IF cumsbyld2_1 > 0.05 THEN IF cumsbyld2_1 <= 0.15 THEN DO;
470 sbyld2_1 = ((cumsbyld2_1 - pb[,2])*(sb_ylda[,3]-sb_ylda[,2]))/0.1 + sb_ylda[,2]; END;
471 IF cumsbyld2_1 > 0.15 THEN IF cumsbyld2_1 <= 0.25 THEN DO;
472 sbyld2_1 = ((cumsbyld2_1 - pb[,3])*(sb_ylda[,4]-sb_ylda[,3]))/0.1 + sb_ylda[,3]; END;
473 IF cumsbyld2_1 > 0.25 THEN IF cumsbyld2_1 <= 0.35 THEN DO;
474 sbyld2_1 = ((cumsbyld2_1 - pb[,4])*(sb_ylda[,5]-sb_ylda[,4]))/0.1 + sb_ylda[,4]; END;
475 IF cumsbyld2_1 > 0.35 THEN IF cumsbyld2_1 <= 0.45 THEN DO;
476 sbyld2_1 = ((cumsbyld2_1 - pb[,5])*(sb_ylda[,6]-sb_ylda[,5]))/0.1 + sb_ylda[,5]; END;
477 IF cumsbyld2_1 > 0.45 THEN IF cumsbyld2_1 <= 0.55 THEN DO;
478 sbyld2_1 = ((cumsbyld2_1 - pb[,6])*(sb_ylda[,7]-sb_ylda[,6]))/0.1 + sb_ylda[,6]; END;
479 IF cumsbyld2_1 > 0.55 THEN IF cumsbyld2_1 <= 0.65 THEN DO;
480 sbyld2_1 = ((cumsbyld2_1 - pb[,7])*(sb_ylda[,8]-sb_ylda[,7]))/0.1 + sb_ylda[,7]; END;
481 IF cumsbyld2_1 > 0.65 THEN IF cumsbyld2_1 <= 0.75 THEN DO;
482 sbyld2_1 = ((cumsbyld2_1 - pb[,8])*(sb_ylda[,9]-sb_ylda[,8]))/0.1 + sb_ylda[,8]; END;
483 IF cumsbyld2_1 > 0.75 THEN IF cumsbyld2_1 <= 0.85 THEN DO;
484 sbyld2_1 = ((cumsbyld2_1 - pb[,9])*(sb_ylda[,10]-sb_ylda[,9]))/0.1 + sb_ylda[,9]; END;
485 IF cumsbyld2_1 > 0.85 THEN IF cumsbyld2_1 <= 0.95 THEN DO;
486 sbyld2_1 = ((cumsbyld2_1 - pb[,10])*(sb_ylda[,11]-sb_ylda[,10]))/0.1 + sb_ylda[,10]; END;
487 IF cumsbyld2_1 > 0.95 THEN IF cumsbyld2_1 <= 1.00 THEN DO;
488 sbyld2_1 = ((cumsbyld2_1 - pb[,11])*(sb_ylda[,12]-sb_ylda[,11]))/0.05 + sb_ylda[,11]; END;
489
490 IF cumsbyld3_1 >= 0.00 THEN IF cumsbyld3_1 <= 0.05 THEN DO; * SOYBEAN YIELD: YEAR 3 *;
491 sbyld3_1 = ((cumsbyld3_1 - pb[,1])*(sb_ylda[,2]-sb_ylda[,1]))/0.05 + sb_ylda[,1]; END;
492 IF cumsbyld3_1 > 0.05 THEN IF cumsbyld3_1 <= 0.15 THEN DO;
493 sbyld3_1 = ((cumsbyld3_1 - pb[,2])*(sb_ylda[,3]-sb_ylda[,2]))/0.1 + sb_ylda[,2]; END;
494 IF cumsbyld3_1 > 0.15 THEN IF cumsbyld3_1 <= 0.25 THEN DO;
495 sbyld3_1 = ((cumsbyld3_1 - pb[,3])*(sb_ylda[,4]-sb_ylda[,3]))/0.1 + sb_ylda[,3]; END;
496 IF cumsbyld3_1 > 0.25 THEN IF cumsbyld3_1 <= 0.35 THEN DO;
497 sbyld3_1 = ((cumsbyld3_1 - pb[,4])*(sb_ylda[,5]-sb_ylda[,4]))/0.1 + sb_ylda[,4]; END;
498 IF cumsbyld3_1 > 0.35 THEN IF cumsbyld3_1 <= 0.45 THEN DO;
499 sbyld3_1 = ((cumsbyld3_1 - pb[,5])*(sb_ylda[,6]-sb_ylda[,5]))/0.1 + sb_ylda[,5]; END;
500 IF cumsbyld3_1 > 0.45 THEN IF cumsbyld3_1 <= 0.55 THEN DO;
501 sbyld3_1 = ((cumsbyld3_1 - pb[,6])*(sb_ylda[,7]-sb_ylda[,6]))/0.1 + sb_ylda[,6]; END;
502 IF cumsbyld3_1 > 0.55 THEN IF cumsbyld3_1 <= 0.65 THEN DO;
503 sbyld3_1 = ((cumsbyld3_1 - pb[,7])*(sb_ylda[,8]-sb_ylda[,7]))/0.1 + sb_ylda[,7]; END;

```

```

504 IF cumsbyld3_1 > 0.65 THEN IF cumsbyld3_1 <= 0.75 THEN D0;
505 sbyld3_1 = ((cumsbyld3_1 - pb[,8])*(sb_ylda[,9]-sb_ylda[,8]))/0.1 + sb_ylda[,8]; END;
506 IF cumsbyld3_1 > 0.75 THEN IF cumsbyld3_1 <= 0.85 THEN D0;
507 sbyld3_1 = ((cumsbyld3_1 - pb[,9])*(sb_ylda[,10]-sb_ylda[,9]))/0.1 + sb_ylda[,9]; END;
508 IF cumsbyld3_1 > 0.85 THEN IF cumsbyld3_1 <= 0.95 THEN D0;
509 sbyld3_1 = ((cumsbyld3_1 - pb[,10])*(sb_ylda[,11]-sb_ylda[,10]))/0.1 + sb_ylda[,10]; END;
510 IF cumsbyld3_1 > 0.95 THEN IF cumsbyld3_1 <= 1.00 THEN D0;
511 sbyld3_1 = ((cumsbyld3_1 - pb[,11])*(sb_ylda[,12]-sb_ylda[,11]))/0.05 + sb_ylda[,11]; END;
512 * WHEAT YIELD *;
513 IF cumwhyld1_1 >= 0.00 THEN IF cumwhyld1_1 <= 0.05 THEN D0; * WHEAT YIELD: YEAR 1 *;
514 whyld1_1 = ((cumwhyld1_1 - pb[,1])*(wh_ylda[,2]-wh_ylda[,1]))/0.05 + wh_ylda[,1]; END;
515 IF cumwhyld1_1 > 0.05 THEN IF cumwhyld1_1 <= 0.15 THEN D0;
516 whyld1_1 = ((cumwhyld1_1 - pb[,2])*(wh_ylda[,3]-wh_ylda[,2]))/0.1 + wh_ylda[,2]; END;
517 IF cumwhyld1_1 > 0.15 THEN IF cumwhyld1_1 <= 0.25 THEN D0;
518 whyld1_1 = ((cumwhyld1_1 - pb[,3])*(wh_ylda[,4]-wh_ylda[,3]))/0.1 + wh_ylda[,3]; END;
519 IF cumwhyld1_1 > 0.25 THEN IF cumwhyld1_1 <= 0.35 THEN D0;
520 whyld1_1 = ((cumwhyld1_1 - pb[,4])*(wh_ylda[,5]-wh_ylda[,4]))/0.1 + wh_ylda[,4]; END;
521 IF cumwhyld1_1 > 0.35 THEN IF cumwhyld1_1 <= 0.45 THEN D0;
522 whyld1_1 = ((cumwhyld1_1 - pb[,5])*(wh_ylda[,6]-wh_ylda[,5]))/0.1 + wh_ylda[,5]; END;
523 IF cumwhyld1_1 > 0.45 THEN IF cumwhyld1_1 <= 0.55 THEN D0;
524 whyld1_1 = ((cumwhyld1_1 - pb[,6])*(wh_ylda[,7]-wh_ylda[,6]))/0.1 + wh_ylda[,6]; END;
525 IF cumwhyld1_1 > 0.55 THEN IF cumwhyld1_1 <= 0.65 THEN D0;
526 whyld1_1 = ((cumwhyld1_1 - pb[,7])*(wh_ylda[,8]-wh_ylda[,7]))/0.1 + wh_ylda[,7]; END;
527 IF cumwhyld1_1 > 0.65 THEN IF cumwhyld1_1 <= 0.75 THEN D0;
528 whyld1_1 = ((cumwhyld1_1 - pb[,8])*(wh_ylda[,9]-wh_ylda[,8]))/0.1 + wh_ylda[,8]; END;
529 IF cumwhyld1_1 > 0.75 THEN IF cumwhyld1_1 <= 0.85 THEN D0;
530 whyld1_1 = ((cumwhyld1_1 - pb[,9])*(wh_ylda[,10]-wh_ylda[,9]))/0.1 + wh_ylda[,9]; END;
531 IF cumwhyld1_1 > 0.85 THEN IF cumwhyld1_1 <= 0.95 THEN D0;
532 whyld1_1 = ((cumwhyld1_1 - pb[,10])*(wh_ylda[,11]-wh_ylda[,10]))/0.1 + wh_ylda[,10]; END;
533 IF cumwhyld1_1 > 0.95 THEN IF cumwhyld1_1 <= 1.00 THEN D0;
534 whyld1_1 = ((cumwhyld1_1 - pb[,11])*(wh_ylda[,12]-wh_ylda[,11]))/0.05 + wh_ylda[,11]; END;
535
536 IF cumwhyld2_1 >= 0.00 THEN IF cumwhyld2_1 <= 0.05 THEN D0; * WHEAT YIELD: YEAR 2 *;
537 whyld2_1 = ((cumwhyld2_1 - pb[,1])*(wh_ylda[,2]-wh_ylda[,1]))/0.05 + wh_ylda[,1]; END;
538 IF cumwhyld2_1 > 0.05 THEN IF cumwhyld2_1 <= 0.15 THEN D0;
539 whyld2_1 = ((cumwhyld2_1 - pb[,2])*(wh_ylda[,3]-wh_ylda[,2]))/0.1 + wh_ylda[,2]; END;
540 IF cumwhyld2_1 > 0.15 THEN IF cumwhyld2_1 <= 0.25 THEN D0;
541 whyld2_1 = ((cumwhyld2_1 - pb[,3])*(wh_ylda[,4]-wh_ylda[,3]))/0.1 + wh_ylda[,3]; END;
542 IF cumwhyld2_1 > 0.25 THEN IF cumwhyld2_1 <= 0.35 THEN D0;
543 whyld2_1 = ((cumwhyld2_1 - pb[,4])*(wh_ylda[,5]-wh_ylda[,4]))/0.1 + wh_ylda[,4]; END;
544 IF cumwhyld2_1 > 0.35 THEN IF cumwhyld2_1 <= 0.45 THEN D0;
545 whyld2_1 = ((cumwhyld2_1 - pb[,5])*(wh_ylda[,6]-wh_ylda[,5]))/0.1 + wh_ylda[,5]; END;
546 IF cumwhyld2_1 > 0.45 THEN IF cumwhyld2_1 <= 0.55 THEN D0;
547 whyld2_1 = ((cumwhyld2_1 - pb[,6])*(wh_ylda[,7]-wh_ylda[,6]))/0.1 + wh_ylda[,6]; END;
548 IF cumwhyld2_1 > 0.55 THEN IF cumwhyld2_1 <= 0.65 THEN D0;
549 whyld2_1 = ((cumwhyld2_1 - pb[,7])*(wh_ylda[,8]-wh_ylda[,7]))/0.1 + wh_ylda[,7]; END;
550 IF cumwhyld2_1 > 0.65 THEN IF cumwhyld2_1 <= 0.75 THEN D0;
551 whyld2_1 = ((cumwhyld2_1 - pb[,8])*(wh_ylda[,9]-wh_ylda[,8]))/0.1 + wh_ylda[,8]; END;
552 IF cumwhyld2_1 > 0.75 THEN IF cumwhyld2_1 <= 0.85 THEN D0;
553 whyld2_1 = ((cumwhyld2_1 - pb[,9])*(wh_ylda[,10]-wh_ylda[,9]))/0.1 + wh_ylda[,9]; END;
554 IF cumwhyld2_1 > 0.85 THEN IF cumwhyld2_1 <= 0.95 THEN D0;
555 whyld2_1 = ((cumwhyld2_1 - pb[,10])*(wh_ylda[,11]-wh_ylda[,10]))/0.1 + wh_ylda[,10]; END;
556 IF cumwhyld2_1 > 0.95 THEN IF cumwhyld2_1 <= 1.00 THEN D0;
557 whyld2_1 = ((cumwhyld2_1 - pb[,11])*(wh_ylda[,12]-wh_ylda[,11]))/0.05 + wh_ylda[,11]; END;
558
559 IF cumwhyld3_1 >= 0.00 THEN IF cumwhyld3_1 <= 0.05 THEN D0; * WHEAT YIELD YEAR 3*;

```

```

560 whyld3_1 = ((cumwhyld3_1 - pb[,1])*(wh_ylda[,2]-wh_ylda[,1]))/0.05 + wh_ylda[,1]; END;
561 IF cumwhyld3_1 > 0.05 THEN IF cumwhyld3_1 <= 0.15 THEN D0;
562 whyld3_1 = ((cumwhyld3_1 - pb[,2])*(wh_ylda[,3]-wh_ylda[,2]))/0.1 + wh_ylda[,2]; END;
563 IF cumwhyld3_1 > 0.15 THEN IF cumwhyld3_1 <= 0.25 THEN D0;
564 whyld3_1 = ((cumwhyld3_1 - pb[,3])*(wh_ylda[,4]-wh_ylda[,3]))/0.1 + wh_ylda[,3]; END;
565 IF cumwhyld3_1 > 0.25 THEN IF cumwhyld3_1 <= 0.35 THEN D0;
566 whyld3_1 = ((cumwhyld3_1 - pb[,4])*(wh_ylda[,5]-wh_ylda[,4]))/0.1 + wh_ylda[,4]; END;
567 IF cumwhyld3_1 > 0.35 THEN IF cumwhyld3_1 <= 0.45 THEN D0;
568 whyld3_1 = ((cumwhyld3_1 - pb[,5])*(wh_ylda[,6]-wh_ylda[,5]))/0.1 + wh_ylda[,5]; END;
569 IF cumwhyld3_1 > 0.45 THEN IF cumwhyld3_1 <= 0.55 THEN D0;
570 whyld3_1 = ((cumwhyld3_1 - pb[,6])*(wh_ylda[,7]-wh_ylda[,6]))/0.1 + wh_ylda[,6]; END;
571 IF cumwhyld3_1 > 0.55 THEN IF cumwhyld3_1 <= 0.65 THEN D0;
572 whyld3_1 = ((cumwhyld3_1 - pb[,7])*(wh_ylda[,8]-wh_ylda[,7]))/0.1 + wh_ylda[,7]; END;
573 IF cumwhyld3_1 > 0.65 THEN IF cumwhyld3_1 <= 0.75 THEN D0;
574 whyld3_1 = ((cumwhyld3_1 - pb[,8])*(wh_ylda[,9]-wh_ylda[,8]))/0.1 + wh_ylda[,8]; END;
575 IF cumwhyld3_1 > 0.75 THEN IF cumwhyld3_1 <= 0.85 THEN D0;
576 whyld3_1 = ((cumwhyld3_1 - pb[,9])*(wh_ylda[,10]-wh_ylda[,9]))/0.1 + wh_ylda[,9]; END;
577 IF cumwhyld3_1 > 0.85 THEN IF cumwhyld3_1 <= 0.95 THEN D0;
578 whyld3_1 = ((cumwhyld3_1 - pb[,10])*(wh_ylda[,11]-wh_ylda[,10]))/0.1 + wh_ylda[,10]; END;
579 IF cumwhyld3_1 > 0.95 THEN IF cumwhyld3_1 <= 1.00 THEN D0;
580 whyld3_1 = ((cumwhyld3_1 - pb[,11])*(wh_ylda[,12]-wh_ylda[,11]))/0.05 + wh_ylda[,11]; END;
581 * SORGHUM YIELD *;
582 IF cumsgyld1_1 >= 0.00 THEN IF cumsgyld1_1 <= 0.05 THEN D0; * SORGHUM YIELD: YEAR 1 *;
583 sgyld1_1 = ((cumsgyld1_1 - pb[,1])*(sg_ylda[,2]-sg_ylda[,1]))/0.05 + sg_ylda[,1]; END;
584 IF cumsgyld1_1 > 0.05 THEN IF cumsgyld1_1 <= 0.15 THEN D0;
585 sgyld1_1 = ((cumsgyld1_1 - pb[,2])*(sg_ylda[,3]-sg_ylda[,2]))/0.1 + sg_ylda[,2]; END;
586 IF cumsgyld1_1 > 0.15 THEN IF cumsgyld1_1 <= 0.25 THEN D0;
587 sgyld1_1 = ((cumsgyld1_1 - pb[,3])*(sg_ylda[,4]-sg_ylda[,3]))/0.1 + sg_ylda[,3]; END;
588 IF cumsgyld1_1 > 0.25 THEN IF cumsgyld1_1 <= 0.35 THEN D0;
589 sgyld1_1 = ((cumsgyld1_1 - pb[,4])*(sg_ylda[,5]-sg_ylda[,4]))/0.1 + sg_ylda[,4]; END;
590 IF cumsgyld1_1 > 0.35 THEN IF cumsgyld1_1 <= 0.45 THEN D0;
591 sgyld1_1 = ((cumsgyld1_1 - pb[,5])*(sg_ylda[,6]-sg_ylda[,5]))/0.1 + sg_ylda[,5]; END;
592 IF cumsgyld1_1 > 0.45 THEN IF cumsgyld1_1 <= 0.55 THEN D0;
593 sgyld1_1 = ((cumsgyld1_1 - pb[,6])*(sg_ylda[,7]-sg_ylda[,6]))/0.1 + sg_ylda[,6]; END;
594 IF cumsgyld1_1 > 0.55 THEN IF cumsgyld1_1 <= 0.65 THEN D0;
595 sgyld1_1 = ((cumsgyld1_1 - pb[,7])*(sg_ylda[,8]-sg_ylda[,7]))/0.1 + sg_ylda[,7]; END;
596 IF cumsgyld1_1 > 0.65 THEN IF cumsgyld1_1 <= 0.75 THEN D0;
597 sgyld1_1 = ((cumsgyld1_1 - pb[,8])*(sg_ylda[,9]-sg_ylda[,8]))/0.1 + sg_ylda[,8]; END;
598 IF cumsgyld1_1 > 0.75 THEN IF cumsgyld1_1 <= 0.85 THEN D0;
599 sgyld1_1 = ((cumsgyld1_1 - pb[,9])*(sg_ylda[,10]-sg_ylda[,9]))/0.1 + sg_ylda[,9]; END;
600 IF cumsgyld1_1 > 0.85 THEN IF cumsgyld1_1 <= 0.95 THEN D0;
601 sgyld1_1 = ((cumsgyld1_1 - pb[,10])*(sg_ylda[,11]-sg_ylda[,10]))/0.1 + sg_ylda[,10]; END;
602 IF cumsgyld1_1 > 0.95 THEN IF cumsgyld1_1 <= 1.00 THEN D0;
603 sgyld1_1 = ((cumsgyld1_1 - pb[,11])*(sg_ylda[,12]-sg_ylda[,11]))/0.05 + sg_ylda[,11]; END;
604
605 IF cumsgyld2_1 >= 0.00 THEN IF cumsgyld2_1 <= 0.05 THEN D0; * SORGHUM YIELD: YEAR 2 *;
606 sgyld2_1 = ((cumsgyld2_1 - pb[,1])*(sg_ylda[,2]-sg_ylda[,1]))/0.05 + sg_ylda[,1]; END;
607 IF cumsgyld2_1 > 0.05 THEN IF cumsgyld2_1 <= 0.15 THEN D0;
608 sgyld2_1 = ((cumsgyld2_1 - pb[,2])*(sg_ylda[,3]-sg_ylda[,2]))/0.1 + sg_ylda[,2]; END;
609 IF cumsgyld2_1 > 0.15 THEN IF cumsgyld2_1 <= 0.25 THEN D0;
610 sgyld2_1 = ((cumsgyld2_1 - pb[,3])*(sg_ylda[,4]-sg_ylda[,3]))/0.1 + sg_ylda[,3]; END;
611 IF cumsgyld2_1 > 0.25 THEN IF cumsgyld2_1 <= 0.35 THEN D0;
612 sgyld2_1 = ((cumsgyld2_1 - pb[,4])*(sg_ylda[,5]-sg_ylda[,4]))/0.1 + sg_ylda[,4]; END;
613 IF cumsgyld2_1 > 0.35 THEN IF cumsgyld2_1 <= 0.45 THEN D0;
614 sgyld2_1 = ((cumsgyld2_1 - pb[,5])*(sg_ylda[,6]-sg_ylda[,5]))/0.1 + sg_ylda[,5]; END;
615 IF cumsgyld2_1 > 0.45 THEN IF cumsgyld2_1 <= 0.55 THEN D0;

```

```

616 sgyl2_1 = ((cumsgyld2_1 - pb[,6])*(sg_ylda[,7]-sg_ylda[,6]))/0.1 + sg_ylda[,6]; END;
617 IF cumsgyld2_1 > 0.55 THEN IF cumsgyld2_1 <= 0.65 THEN D0;
618 sgyl2_1 = ((cumsgyld2_1 - pb[,7])*(sg_ylda[,8]-sg_ylda[,7]))/0.1 + sg_ylda[,7]; END;
619 IF cumsgyld2_1 > 0.65 THEN IF cumsgyld2_1 <= 0.75 THEN D0;
620 sgyl2_1 = ((cumsgyld2_1 - pb[,8])*(sg_ylda[,9]-sg_ylda[,8]))/0.1 + sg_ylda[,8]; END;
621 IF cumsgyld2_1 > 0.75 THEN IF cumsgyld2_1 <= 0.85 THEN D0;
622 sgyl2_1 = ((cumsgyld2_1 - pb[,9])*(sg_ylda[,10]-sg_ylda[,9]))/0.1 + sg_ylda[,9]; END;
623 IF cumsgyld2_1 > 0.85 THEN IF cumsgyld2_1 <= 0.95 THEN D0;
624 sgyl2_1 = ((cumsgyld2_1 - pb[,10])*(sg_ylda[,11]-sg_ylda[,10]))/0.1 + sg_ylda[,10]; END;
625 IF cumsgyld2_1 > 0.95 THEN IF cumsgyld2_1 <= 1.00 THEN D0;
626 sgyl2_1 = ((cumsgyld2_1 - pb[,11])*(sg_ylda[,12]-sg_ylda[,11]))/0.05 + sg_ylda[,11]; END;
627
628 IF cumsgyld3_1 >= 0.00 THEN IF cumsgyld3_1 <= 0.05 THEN D0; * SORGHUM YIELD: YEAR 3 *;
629 sgyl3_1 = ((cumsgyld3_1 - pb[,1])*(sg_ylda[,2]-sg_ylda[,1]))/0.05 + sg_ylda[,1]; END;
630 IF cumsgyld3_1 > 0.05 THEN IF cumsgyld3_1 <= 0.15 THEN D0;
631 sgyl3_1 = ((cumsgyld3_1 - pb[,2])*(sg_ylda[,3]-sg_ylda[,2]))/0.1 + sg_ylda[,2]; END;
632 IF cumsgyld3_1 > 0.15 THEN IF cumsgyld3_1 <= 0.25 THEN D0;
633 sgyl3_1 = ((cumsgyld3_1 - pb[,3])*(sg_ylda[,4]-sg_ylda[,3]))/0.1 + sg_ylda[,3]; END;
634 IF cumsgyld3_1 > 0.25 THEN IF cumsgyld3_1 <= 0.35 THEN D0;
635 sgyl3_1 = ((cumsgyld3_1 - pb[,4])*(sg_ylda[,5]-sg_ylda[,4]))/0.1 + sg_ylda[,4]; END;
636 IF cumsgyld3_1 > 0.35 THEN IF cumsgyld3_1 <= 0.45 THEN D0;
637 sgyl3_1 = ((cumsgyld3_1 - pb[,5])*(sg_ylda[,6]-sg_ylda[,5]))/0.1 + sg_ylda[,5]; END;
638 IF cumsgyld3_1 > 0.45 THEN IF cumsgyld3_1 <= 0.55 THEN D0;
639 sgyl3_1 = ((cumsgyld3_1 - pb[,6])*(sg_ylda[,7]-sg_ylda[,6]))/0.1 + sg_ylda[,6]; END;
640 IF cumsgyld3_1 > 0.55 THEN IF cumsgyld3_1 <= 0.65 THEN D0;
641 sgyl3_1 = ((cumsgyld3_1 - pb[,7])*(sg_ylda[,8]-sg_ylda[,7]))/0.1 + sg_ylda[,7]; END;
642 IF cumsgyld3_1 > 0.65 THEN IF cumsgyld3_1 <= 0.75 THEN D0;
643 sgyl3_1 = ((cumsgyld3_1 - pb[,8])*(sg_ylda[,9]-sg_ylda[,8]))/0.1 + sg_ylda[,8]; END;
644 IF cumsgyld3_1 > 0.75 THEN IF cumsgyld3_1 <= 0.85 THEN D0;
645 sgyl3_1 = ((cumsgyld3_1 - pb[,9])*(sg_ylda[,10]-sg_ylda[,9]))/0.1 + sg_ylda[,9]; END;
646 IF cumsgyld3_1 > 0.85 THEN IF cumsgyld3_1 <= 0.95 THEN D0;
647 sgyl3_1 = ((cumsgyld3_1 - pb[,10])*(sg_ylda[,11]-sg_ylda[,10]))/0.1 + sg_ylda[,10]; END;
648 IF cumsgyld3_1 > 0.95 THEN IF cumsgyld3_1 <= 1.00 THEN D0;
649 sgyl3_1 = ((cumsgyld3_1 - pb[,11])*(sg_ylda[,12]-sg_ylda[,11]))/0.05 + sg_ylda[,11]; END;
650 * CORN PRICE *;
651 IF cumcrprc1_1 >= 0.00 THEN IF cumcrprc1_1 <= 0.05 THEN D0; * CORN PRICE: YEAR 1 *;
652 crprc1_1 = ((cumcrprc1_1 - pb[,1])*(cr_prca[,2]-cr_prca[,1]))/0.05 + cr_prca[,1]; END;
653 IF cumcrprc1_1 > 0.05 THEN IF cumcrprc1_1 <= 0.15 THEN D0;
654 crprc1_1 = ((cumcrprc1_1 - pb[,2])*(cr_prca[,3]-cr_prca[,2]))/0.1 + cr_prca[,2]; END;
655 IF cumcrprc1_1 > 0.15 THEN IF cumcrprc1_1 <= 0.25 THEN D0;
656 crprc1_1 = ((cumcrprc1_1 - pb[,3])*(cr_prca[,4]-cr_prca[,3]))/0.1 + cr_prca[,3]; END;
657 IF cumcrprc1_1 > 0.25 THEN IF cumcrprc1_1 <= 0.35 THEN D0;
658 crprc1_1 = ((cumcrprc1_1 - pb[,4])*(cr_prca[,5]-cr_prca[,4]))/0.1 + cr_prca[,4]; END;
659 IF cumcrprc1_1 > 0.35 THEN IF cumcrprc1_1 <= 0.45 THEN D0;
660 crprc1_1 = ((cumcrprc1_1 - pb[,5])*(cr_prca[,6]-cr_prca[,5]))/0.1 + cr_prca[,5]; END;
661 IF cumcrprc1_1 > 0.45 THEN IF cumcrprc1_1 <= 0.55 THEN D0;
662 crprc1_1 = ((cumcrprc1_1 - pb[,6])*(cr_prca[,7]-cr_prca[,6]))/0.1 + cr_prca[,6]; END;
663 IF cumcrprc1_1 > 0.55 THEN IF cumcrprc1_1 <= 0.65 THEN D0;
664 crprc1_1 = ((cumcrprc1_1 - pb[,7])*(cr_prca[,8]-cr_prca[,7]))/0.1 + cr_prca[,7]; END;
665 IF cumcrprc1_1 > 0.65 THEN IF cumcrprc1_1 <= 0.75 THEN D0;
666 crprc1_1 = ((cumcrprc1_1 - pb[,8])*(cr_prca[,9]-cr_prca[,8]))/0.1 + cr_prca[,8]; END;
667 IF cumcrprc1_1 > 0.75 THEN IF cumcrprc1_1 <= 0.85 THEN D0;
668 crprc1_1 = ((cumcrprc1_1 - pb[,9])*(cr_prca[,10]-cr_prca[,9]))/0.1 + cr_prca[,9]; END;
669 IF cumcrprc1_1 > 0.85 THEN IF cumcrprc1_1 <= 0.95 THEN D0;
670 crprc1_1 = ((cumcrprc1_1 - pb[,10])*(cr_prca[,11]-cr_prca[,10]))/0.1 + cr_prca[,10]; END;
671 IF cumcrprc1_1 > 0.95 THEN IF cumcrprc1_1 <= 1.00 THEN D0;

```

```

672 crprc1_1 = ((cumcrprc1_1 - pb[,11])*(cr_prca[,12]-cr_prca[,11]))/0.05 + cr_prca[,11]; END;
673
674 IF cumcrprc2_1 >= 0.00 THEN IF cumcrprc2_1 <= 0.05 THEN D0;      * CORN PRICE: YEAR 2 *;
675 crprc2_1 = ((cumcrprc2_1 - pb[,1])*(cr_prca[,2]-cr_prca[,1]))/0.05 + cr_prca[,1]; END;
676 IF cumcrprc2_1 > 0.05 THEN IF cumcrprc2_1 <= 0.15 THEN D0;
677 crprc2_1 = ((cumcrprc2_1 - pb[,2])*(cr_prca[,3]-cr_prca[,2]))/0.1 + cr_prca[,2]; END;
678 IF cumcrprc2_1 > 0.15 THEN IF cumcrprc2_1 <= 0.25 THEN D0;
679 crprc2_1 = ((cumcrprc2_1 - pb[,3])*(cr_prca[,4]-cr_prca[,3]))/0.1 + cr_prca[,3]; END;
680 IF cumcrprc2_1 > 0.25 THEN IF cumcrprc2_1 <= 0.35 THEN D0;
681 crprc2_1 = ((cumcrprc2_1 - pb[,4])*(cr_prca[,5]-cr_prca[,4]))/0.1 + cr_prca[,4]; END;
682 IF cumcrprc2_1 > 0.35 THEN IF cumcrprc2_1 <= 0.45 THEN D0;
683 crprc2_1 = ((cumcrprc2_1 - pb[,5])*(cr_prca[,6]-cr_prca[,5]))/0.1 + cr_prca[,5]; END;
684 IF cumcrprc2_1 > 0.45 THEN IF cumcrprc2_1 <= 0.55 THEN D0;
685 crprc2_1 = ((cumcrprc2_1 - pb[,6])*(cr_prca[,7]-cr_prca[,6]))/0.1 + cr_prca[,6]; END;
686 IF cumcrprc2_1 > 0.55 THEN IF cumcrprc2_1 <= 0.65 THEN D0;
687 crprc2_1 = ((cumcrprc2_1 - pb[,7])*(cr_prca[,8]-cr_prca[,7]))/0.1 + cr_prca[,7]; END;
688 IF cumcrprc2_1 > 0.65 THEN IF cumcrprc2_1 <= 0.75 THEN D0;
689 crprc2_1 = ((cumcrprc2_1 - pb[,8])*(cr_prca[,9]-cr_prca[,8]))/0.1 + cr_prca[,8]; END;
690 IF cumcrprc2_1 > 0.75 THEN IF cumcrprc2_1 <= 0.85 THEN D0;
691 crprc2_1 = ((cumcrprc2_1 - pb[,9])*(cr_prca[,10]-cr_prca[,9]))/0.1 + cr_prca[,9]; END;
692 IF cumcrprc2_1 > 0.85 THEN IF cumcrprc2_1 <= 0.95 THEN D0;
693 crprc2_1 = ((cumcrprc2_1 - pb[,10])*(cr_prca[,11]-cr_prca[,10]))/0.1 + cr_prca[,10]; END;
694 IF cumcrprc2_1 > 0.95 THEN IF cumcrprc2_1 <= 1.00 THEN D0;
695 crprc2_1 = ((cumcrprc2_1 - pb[,11])*(cr_prca[,12]-cr_prca[,11]))/0.05 + cr_prca[,11]; END;
696
697 IF cumcrprc3_1 >= 0.00 THEN IF cumcrprc3_1 <= 0.05 THEN D0;      * CORN PRICE: YEAR 3 *;
698 crprc3_1 = ((cumcrprc3_1 - pb[,1])*(cr_prca[,2]-cr_prca[,1]))/0.05 + cr_prca[,1]; END;
699 IF cumcrprc3_1 > 0.05 THEN IF cumcrprc3_1 <= 0.15 THEN D0;
700 crprc3_1 = ((cumcrprc3_1 - pb[,2])*(cr_prca[,3]-cr_prca[,2]))/0.1 + cr_prca[,2]; END;
701 IF cumcrprc3_1 > 0.15 THEN IF cumcrprc3_1 <= 0.25 THEN D0;
702 crprc3_1 = ((cumcrprc3_1 - pb[,3])*(cr_prca[,4]-cr_prca[,3]))/0.1 + cr_prca[,3]; END;
703 IF cumcrprc3_1 > 0.25 THEN IF cumcrprc3_1 <= 0.35 THEN D0;
704 crprc3_1 = ((cumcrprc3_1 - pb[,4])*(cr_prca[,5]-cr_prca[,4]))/0.1 + cr_prca[,4]; END;
705 IF cumcrprc3_1 > 0.35 THEN IF cumcrprc3_1 <= 0.45 THEN D0;
706 crprc3_1 = ((cumcrprc3_1 - pb[,5])*(cr_prca[,6]-cr_prca[,5]))/0.1 + cr_prca[,5]; END;
707 IF cumcrprc3_1 > 0.45 THEN IF cumcrprc3_1 <= 0.55 THEN D0;
708 crprc3_1 = ((cumcrprc3_1 - pb[,6])*(cr_prca[,7]-cr_prca[,6]))/0.1 + cr_prca[,6]; END;
709 IF cumcrprc3_1 > 0.55 THEN IF cumcrprc3_1 <= 0.65 THEN D0;
710 crprc3_1 = ((cumcrprc3_1 - pb[,7])*(cr_prca[,8]-cr_prca[,7]))/0.1 + cr_prca[,7]; END;
711 IF cumcrprc3_1 > 0.65 THEN IF cumcrprc3_1 <= 0.75 THEN D0;
712 crprc3_1 = ((cumcrprc3_1 - pb[,8])*(cr_prca[,9]-cr_prca[,8]))/0.1 + cr_prca[,8]; END;
713 IF cumcrprc3_1 > 0.75 THEN IF cumcrprc3_1 <= 0.85 THEN D0;
714 crprc3_1 = ((cumcrprc3_1 - pb[,9])*(cr_prca[,10]-cr_prca[,9]))/0.1 + cr_prca[,9]; END;
715 IF cumcrprc3_1 > 0.85 THEN IF cumcrprc3_1 <= 0.95 THEN D0;
716 crprc3_1 = ((cumcrprc3_1 - pb[,10])*(cr_prca[,11]-cr_prca[,10]))/0.1 + cr_prca[,10]; END;
717 IF cumcrprc3_1 > 0.95 THEN IF cumcrprc3_1 <= 1.00 THEN D0;
718 crprc3_1 = ((cumcrprc3_1 - pb[,11])*(cr_prca[,12]-cr_prca[,11]))/0.05 + cr_prca[,11]; END;
719 * SOYBEAN PRICE *;
720 IF cumsbprc1_1 >= 0.00 THEN IF cumsbprc1_1 <= 0.05 THEN D0;      * SOYBEAN PRICE: YEAR 1 *;
721 sbprc1_1 = ((cumsbprc1_1 - pb[,1])*(sb_prca[,2]-sb_prca[,1]))/0.05 + sb_prca[,1]; END;
722 IF cumsbprc1_1 > 0.05 THEN IF cumsbprc1_1 <= 0.15 THEN D0;
723 sbprc1_1 = ((cumsbprc1_1 - pb[,2])*(sb_prca[,3]-sb_prca[,2]))/0.1 + sb_prca[,2]; END;
724 IF cumsbprc1_1 > 0.15 THEN IF cumsbprc1_1 <= 0.25 THEN D0;
725 sbprc1_1 = ((cumsbprc1_1 - pb[,3])*(sb_prca[,4]-sb_prca[,3]))/0.1 + sb_prca[,3]; END;
726 IF cumsbprc1_1 > 0.25 THEN IF cumsbprc1_1 <= 0.35 THEN D0;
727 sbprc1_1 = ((cumsbprc1_1 - pb[,4])*(sb_prca[,5]-sb_prca[,4]))/0.1 + sb_prca[,4]; END;

```

```

728 IF cumsbprc1_1 > 0.35 THEN IF cumsbprc1_1 <= 0.45 THEN D0;
729 sbprc1_1 = ((cumsbprc1_1 - pb[,5])*(sb_prca[,6]-sb_prca[,5]))/0.1 + sb_prca[,5]; END;
730 IF cumsbprc1_1 > 0.45 THEN IF cumsbprc1_1 <= 0.55 THEN D0;
731 sbprc1_1 = ((cumsbprc1_1 - pb[,6])*(sb_prca[,7]-sb_prca[,6]))/0.1 + sb_prca[,6]; END;
732 IF cumsbprc1_1 > 0.55 THEN IF cumsbprc1_1 <= 0.65 THEN D0;
733 sbprc1_1 = ((cumsbprc1_1 - pb[,7])*(sb_prca[,8]-sb_prca[,7]))/0.1 + sb_prca[,7]; END;
734 IF cumsbprc1_1 > 0.65 THEN IF cumsbprc1_1 <= 0.75 THEN D0;
735 sbprc1_1 = ((cumsbprc1_1 - pb[,8])*(sb_prca[,9]-sb_prca[,8]))/0.1 + sb_prca[,8]; END;
736 IF cumsbprc1_1 > 0.75 THEN IF cumsbprc1_1 <= 0.85 THEN D0;
737 sbprc1_1 = ((cumsbprc1_1 - pb[,9])*(sb_prca[,10]-sb_prca[,9]))/0.1 + sb_prca[,9]; END;
738 IF cumsbprc1_1 > 0.85 THEN IF cumsbprc1_1 <= 0.95 THEN D0;
739 sbprc1_1 = ((cumsbprc1_1 - pb[,10])*(sb_prca[,11]-sb_prca[,10]))/0.1 + sb_prca[,10]; END;
740 IF cumsbprc1_1 > 0.95 THEN IF cumsbprc1_1 <= 1.00 THEN D0;
741 sbprc1_1 = ((cumsbprc1_1 - pb[,11])*(sb_prca[,12]-sb_prca[,11]))/0.05 + sb_prca[,11]; END;
742
743 IF cumsbprc2_1 >= 0.00 THEN IF cumsbprc2_1 <= 0.05 THEN D0; * SOYBEAN PRICE: YEAR 2 *;
744 sbprc2_1 = ((cumsbprc2_1 - pb[,1])*(sb_prca[,2]-sb_prca[,1]))/0.05 + sb_prca[,1]; END;
745 IF cumsbprc2_1 > 0.05 THEN IF cumsbprc2_1 <= 0.15 THEN D0;
746 sbprc2_1 = ((cumsbprc2_1 - pb[,2])*(sb_prca[,3]-sb_prca[,2]))/0.1 + sb_prca[,2]; END;
747 IF cumsbprc2_1 > 0.15 THEN IF cumsbprc2_1 <= 0.25 THEN D0;
748 sbprc2_1 = ((cumsbprc2_1 - pb[,3])*(sb_prca[,4]-sb_prca[,3]))/0.1 + sb_prca[,3]; END;
749 IF cumsbprc2_1 > 0.25 THEN IF cumsbprc2_1 <= 0.35 THEN D0;
750 sbprc2_1 = ((cumsbprc2_1 - pb[,4])*(sb_prca[,5]-sb_prca[,4]))/0.1 + sb_prca[,4]; END;
751 IF cumsbprc2_1 > 0.35 THEN IF cumsbprc2_1 <= 0.45 THEN D0;
752 sbprc2_1 = ((cumsbprc2_1 - pb[,5])*(sb_prca[,6]-sb_prca[,5]))/0.1 + sb_prca[,5]; END;
753 IF cumsbprc2_1 > 0.45 THEN IF cumsbprc2_1 <= 0.55 THEN D0;
754 sbprc2_1 = ((cumsbprc2_1 - pb[,6])*(sb_prca[,7]-sb_prca[,6]))/0.1 + sb_prca[,6]; END;
755 IF cumsbprc2_1 > 0.55 THEN IF cumsbprc2_1 <= 0.65 THEN D0;
756 sbprc2_1 = ((cumsbprc2_1 - pb[,7])*(sb_prca[,8]-sb_prca[,7]))/0.1 + sb_prca[,7]; END;
757 IF cumsbprc2_1 > 0.65 THEN IF cumsbprc2_1 <= 0.75 THEN D0;
758 sbprc2_1 = ((cumsbprc2_1 - pb[,8])*(sb_prca[,9]-sb_prca[,8]))/0.1 + sb_prca[,8]; END;
759 IF cumsbprc2_1 > 0.75 THEN IF cumsbprc2_1 <= 0.85 THEN D0;
760 sbprc2_1 = ((cumsbprc2_1 - pb[,9])*(sb_prca[,10]-sb_prca[,9]))/0.1 + sb_prca[,9]; END;
761 IF cumsbprc2_1 > 0.85 THEN IF cumsbprc2_1 <= 0.95 THEN D0;
762 sbprc2_1 = ((cumsbprc2_1 - pb[,10])*(sb_prca[,11]-sb_prca[,10]))/0.1 + sb_prca[,10]; END;
763 IF cumsbprc2_1 > 0.95 THEN IF cumsbprc2_1 <= 1.00 THEN D0;
764 sbprc2_1 = ((cumsbprc2_1 - pb[,11])*(sb_prca[,12]-sb_prca[,11]))/0.05 + sb_prca[,11]; END;
765
766 IF cumsbprc3_1 >= 0.00 THEN IF cumsbprc3_1 <= 0.05 THEN D0; * SOYBEAN PRICE: YEAR 3 *;
767 sbprc3_1 = ((cumsbprc3_1 - pb[,1])*(sb_prca[,2]-sb_prca[,1]))/0.05 + sb_prca[,1]; END;
768 IF cumsbprc3_1 > 0.05 THEN IF cumsbprc3_1 <= 0.15 THEN D0;
769 sbprc3_1 = ((cumsbprc3_1 - pb[,2])*(sb_prca[,3]-sb_prca[,2]))/0.1 + sb_prca[,2]; END;
770 IF cumsbprc3_1 > 0.15 THEN IF cumsbprc3_1 <= 0.25 THEN D0;
771 sbprc3_1 = ((cumsbprc3_1 - pb[,3])*(sb_prca[,4]-sb_prca[,3]))/0.1 + sb_prca[,3]; END;
772 IF cumsbprc3_1 > 0.25 THEN IF cumsbprc3_1 <= 0.35 THEN D0;
773 sbprc3_1 = ((cumsbprc3_1 - pb[,4])*(sb_prca[,5]-sb_prca[,4]))/0.1 + sb_prca[,4]; END;
774 IF cumsbprc3_1 > 0.35 THEN IF cumsbprc3_1 <= 0.45 THEN D0;
775 sbprc3_1 = ((cumsbprc3_1 - pb[,5])*(sb_prca[,6]-sb_prca[,5]))/0.1 + sb_prca[,5]; END;
776 IF cumsbprc3_1 > 0.45 THEN IF cumsbprc3_1 <= 0.55 THEN D0;
777 sbprc3_1 = ((cumsbprc3_1 - pb[,6])*(sb_prca[,7]-sb_prca[,6]))/0.1 + sb_prca[,6]; END;
778 IF cumsbprc3_1 > 0.55 THEN IF cumsbprc3_1 <= 0.65 THEN D0;
779 sbprc3_1 = ((cumsbprc3_1 - pb[,7])*(sb_prca[,8]-sb_prca[,7]))/0.1 + sb_prca[,7]; END;
780 IF cumsbprc3_1 > 0.65 THEN IF cumsbprc3_1 <= 0.75 THEN D0;
781 sbprc3_1 = ((cumsbprc3_1 - pb[,8])*(sb_prca[,9]-sb_prca[,8]))/0.1 + sb_prca[,8]; END;
782 IF cumsbprc3_1 > 0.75 THEN IF cumsbprc3_1 <= 0.85 THEN D0;
783 sbprc3_1 = ((cumsbprc3_1 - pb[,9])*(sb_prca[,10]-sb_prca[,9]))/0.1 + sb_prca[,9]; END;

```

```

784 IF cumsbprc3_1 > 0.85 THEN IF cumsbprc3_1 <= 0.95 THEN D0;
785 sbprc3_1 = ((cumsbprc3_1 - pb[,10])*(sb_prca[,11]-sb_prca[,10]))/0.1 + sb_prca[,10]; END;
786 IF cumsbprc3_1 > 0.95 THEN IF cumsbprc3_1 <= 1.00 THEN D0;
787 sbprc3_1 = ((cumsbprc3_1 - pb[,11])*(sb_prca[,12]-sb_prca[,11]))/0.05 + sb_prca[,11]; END;
788 * WHEAT PRICE *;
789 IF cumwhprc1_1 >= 0.00 THEN IF cumwhprc1_1 <= 0.05 THEN D0; * WHEAT PRICE: YEAR 1 *;
790 whprc1_1 = ((cumwhprc1_1 - pb[,1])*(wh_prca[,2]-wh_prca[,1]))/0.05 + wh_prca[,1]; END;
791 IF cumwhprc1_1 > 0.05 THEN IF cumwhprc1_1 <= 0.15 THEN D0;
792 whprc1_1 = ((cumwhprc1_1 - pb[,2])*(wh_prca[,3]-wh_prca[,2]))/0.1 + wh_prca[,2]; END;
793 IF cumwhprc1_1 > 0.15 THEN IF cumwhprc1_1 <= 0.25 THEN D0;
794 whprc1_1 = ((cumwhprc1_1 - pb[,3])*(wh_prca[,4]-wh_prca[,3]))/0.1 + wh_prca[,3]; END;
795 IF cumwhprc1_1 > 0.25 THEN IF cumwhprc1_1 <= 0.35 THEN D0;
796 whprc1_1 = ((cumwhprc1_1 - pb[,4])*(wh_prca[,5]-wh_prca[,4]))/0.1 + wh_prca[,4]; END;
797 IF cumwhprc1_1 > 0.35 THEN IF cumwhprc1_1 <= 0.45 THEN D0;
798 whprc1_1 = ((cumwhprc1_1 - pb[,5])*(wh_prca[,6]-wh_prca[,5]))/0.1 + wh_prca[,5]; END;
799 IF cumwhprc1_1 > 0.45 THEN IF cumwhprc1_1 <= 0.55 THEN D0;
800 whprc1_1 = ((cumwhprc1_1 - pb[,6])*(wh_prca[,7]-wh_prca[,6]))/0.1 + wh_prca[,6]; END;
801 IF cumwhprc1_1 > 0.55 THEN IF cumwhprc1_1 <= 0.65 THEN D0;
802 whprc1_1 = ((cumwhprc1_1 - pb[,7])*(wh_prca[,8]-wh_prca[,7]))/0.1 + wh_prca[,7]; END;
803 IF cumwhprc1_1 > 0.65 THEN IF cumwhprc1_1 <= 0.75 THEN D0;
804 whprc1_1 = ((cumwhprc1_1 - pb[,8])*(wh_prca[,9]-wh_prca[,8]))/0.1 + wh_prca[,8]; END;
805 IF cumwhprc1_1 > 0.75 THEN IF cumwhprc1_1 <= 0.85 THEN D0;
806 whprc1_1 = ((cumwhprc1_1 - pb[,9])*(wh_prca[,10]-wh_prca[,9]))/0.1 + wh_prca[,9]; END;
807 IF cumwhprc1_1 > 0.85 THEN IF cumwhprc1_1 <= 0.95 THEN D0;
808 whprc1_1 = ((cumwhprc1_1 - pb[,10])*(wh_prca[,11]-wh_prca[,10]))/0.1 + wh_prca[,10]; END;
809 IF cumwhprc1_1 > 0.95 THEN IF cumwhprc1_1 <= 1.00 THEN D0;
810 whprc1_1 = ((cumwhprc1_1 - pb[,11])*(wh_prca[,12]-wh_prca[,11]))/0.05 + wh_prca[,11]; END;
811
812 IF cumwhprc2_1 >= 0.00 THEN IF cumwhprc2_1 <= 0.05 THEN D0; * WHEAT PRICE: YEAR 2 *;
813 whprc2_1 = ((cumwhprc2_1 - pb[,1])*(wh_prca[,2]-wh_prca[,1]))/0.05 + wh_prca[,1]; END;
814 IF cumwhprc2_1 > 0.05 THEN IF cumwhprc2_1 <= 0.15 THEN D0;
815 whprc2_1 = ((cumwhprc2_1 - pb[,2])*(wh_prca[,3]-wh_prca[,2]))/0.1 + wh_prca[,2]; END;
816 IF cumwhprc2_1 > 0.15 THEN IF cumwhprc2_1 <= 0.25 THEN D0;
817 whprc2_1 = ((cumwhprc2_1 - pb[,3])*(wh_prca[,4]-wh_prca[,3]))/0.1 + wh_prca[,3]; END;
818 IF cumwhprc2_1 > 0.25 THEN IF cumwhprc2_1 <= 0.35 THEN D0;
819 whprc2_1 = ((cumwhprc2_1 - pb[,4])*(wh_prca[,5]-wh_prca[,4]))/0.1 + wh_prca[,4]; END;
820 IF cumwhprc2_1 > 0.35 THEN IF cumwhprc2_1 <= 0.45 THEN D0;
821 whprc2_1 = ((cumwhprc2_1 - pb[,5])*(wh_prca[,6]-wh_prca[,5]))/0.1 + wh_prca[,5]; END;
822 IF cumwhprc2_1 > 0.45 THEN IF cumwhprc2_1 <= 0.55 THEN D0;
823 whprc2_1 = ((cumwhprc2_1 - pb[,6])*(wh_prca[,7]-wh_prca[,6]))/0.1 + wh_prca[,6]; END;
824 IF cumwhprc2_1 > 0.55 THEN IF cumwhprc2_1 <= 0.65 THEN D0;
825 whprc2_1 = ((cumwhprc2_1 - pb[,7])*(wh_prca[,8]-wh_prca[,7]))/0.1 + wh_prca[,7]; END;
826 IF cumwhprc2_1 > 0.65 THEN IF cumwhprc2_1 <= 0.75 THEN D0;
827 whprc2_1 = ((cumwhprc2_1 - pb[,8])*(wh_prca[,9]-wh_prca[,8]))/0.1 + wh_prca[,8]; END;
828 IF cumwhprc2_1 > 0.75 THEN IF cumwhprc2_1 <= 0.85 THEN D0;
829 whprc2_1 = ((cumwhprc2_1 - pb[,9])*(wh_prca[,10]-wh_prca[,9]))/0.1 + wh_prca[,9]; END;
830 IF cumwhprc2_1 > 0.85 THEN IF cumwhprc2_1 <= 0.95 THEN D0;
831 whprc2_1 = ((cumwhprc2_1 - pb[,10])*(wh_prca[,11]-wh_prca[,10]))/0.1 + wh_prca[,10]; END;
832 IF cumwhprc2_1 > 0.95 THEN IF cumwhprc2_1 <= 1.00 THEN D0;
833 whprc2_1 = ((cumwhprc2_1 - pb[,11])*(wh_prca[,12]-wh_prca[,11]))/0.05 + wh_prca[,11]; END;
834
835 IF cumwhprc3_1 >= 0.00 THEN IF cumwhprc3_1 <= 0.05 THEN D0; * WHEAT PRICE: YEAR 3 *;
836 whprc3_1 = ((cumwhprc3_1 - pb[,1])*(wh_prca[,2]-wh_prca[,1]))/0.05 + wh_prca[,1]; END;
837 IF cumwhprc3_1 > 0.05 THEN IF cumwhprc3_1 <= 0.15 THEN D0;
838 whprc3_1 = ((cumwhprc3_1 - pb[,2])*(wh_prca[,3]-wh_prca[,2]))/0.1 + wh_prca[,2]; END;
839 IF cumwhprc3_1 > 0.15 THEN IF cumwhprc3_1 <= 0.25 THEN D0;

```

```

840 whprc3_1 = ((cumwhprc3_1 - pb[,3])*(wh_prca[,4]-wh_prca[,3]))/0.1 + wh_prca[,3]; END;
841 IF cumwhprc3_1 > 0.25 THEN IF cumwhprc3_1 <= 0.35 THEN D0;
842 whprc3_1 = ((cumwhprc3_1 - pb[,4])*(wh_prca[,5]-wh_prca[,4]))/0.1 + wh_prca[,4]; END;
843 IF cumwhprc3_1 > 0.35 THEN IF cumwhprc3_1 <= 0.45 THEN D0;
844 whprc3_1 = ((cumwhprc3_1 - pb[,5])*(wh_prca[,6]-wh_prca[,5]))/0.1 + wh_prca[,5]; END;
845 IF cumwhprc3_1 > 0.45 THEN IF cumwhprc3_1 <= 0.55 THEN D0;
846 whprc3_1 = ((cumwhprc3_1 - pb[,6])*(wh_prca[,7]-wh_prca[,6]))/0.1 + wh_prca[,6]; END;
847 IF cumwhprc3_1 > 0.55 THEN IF cumwhprc3_1 <= 0.65 THEN D0;
848 whprc3_1 = ((cumwhprc3_1 - pb[,7])*(wh_prca[,8]-wh_prca[,7]))/0.1 + wh_prca[,7]; END;
849 IF cumwhprc3_1 > 0.65 THEN IF cumwhprc3_1 <= 0.75 THEN D0;
850 whprc3_1 = ((cumwhprc3_1 - pb[,8])*(wh_prca[,9]-wh_prca[,8]))/0.1 + wh_prca[,8]; END;
851 IF cumwhprc3_1 > 0.75 THEN IF cumwhprc3_1 <= 0.85 THEN D0;
852 whprc3_1 = ((cumwhprc3_1 - pb[,9])*(wh_prca[,10]-wh_prca[,9]))/0.1 + wh_prca[,9]; END;
853 IF cumwhprc3_1 > 0.85 THEN IF cumwhprc3_1 <= 0.95 THEN D0;
854 whprc3_1 = ((cumwhprc3_1 - pb[,10])*(wh_prca[,11]-wh_prca[,10]))/0.1 + wh_prca[,10]; END;
855 IF cumwhprc3_1 > 0.95 THEN IF cumwhprc3_1 <= 1.00 THEN D0;
856 whprc3_1 = ((cumwhprc3_1 - pb[,11])*(wh_prca[,12]-wh_prca[,11]))/0.05 + wh_prca[,11]; END;
857 * SORGHUM PRICE *;
858 IF cumsgprc1_1 >= 0.00 THEN IF cumsgprc1_1 <= 0.05 THEN D0; * SORGHUM PRICE: YEAR 1 *;
859 sgprc1_1 = ((cumsgprc1_1 - pb[,1])*(sg_prca[,2]-sg_prca[,1]))/0.05 + sg_prca[,1]; END;
860 IF cumsgprc1_1 > 0.05 THEN IF cumsgprc1_1 <= 0.15 THEN D0;
861 sgprc1_1 = ((cumsgprc1_1 - pb[,2])*(sg_prca[,3]-sg_prca[,2]))/0.1 + sg_prca[,2]; END;
862 IF cumsgprc1_1 > 0.15 THEN IF cumsgprc1_1 <= 0.25 THEN D0;
863 sgprc1_1 = ((cumsgprc1_1 - pb[,3])*(sg_prca[,4]-sg_prca[,3]))/0.1 + sg_prca[,3]; END;
864 IF cumsgprc1_1 > 0.25 THEN IF cumsgprc1_1 <= 0.35 THEN D0;
865 sgprc1_1 = ((cumsgprc1_1 - pb[,4])*(sg_prca[,5]-sg_prca[,4]))/0.1 + sg_prca[,4]; END;
866 IF cumsgprc1_1 > 0.35 THEN IF cumsgprc1_1 <= 0.45 THEN D0;
867 sgprc1_1 = ((cumsgprc1_1 - pb[,5])*(sg_prca[,6]-sg_prca[,5]))/0.1 + sg_prca[,5]; END;
868 IF cumsgprc1_1 > 0.45 THEN IF cumsgprc1_1 <= 0.55 THEN D0;
869 sgprc1_1 = ((cumsgprc1_1 - pb[,6])*(sg_prca[,7]-sg_prca[,6]))/0.1 + sg_prca[,6]; END;
870 IF cumsgprc1_1 > 0.55 THEN IF cumsgprc1_1 <= 0.65 THEN D0;
871 sgprc1_1 = ((cumsgprc1_1 - pb[,7])*(sg_prca[,8]-sg_prca[,7]))/0.1 + sg_prca[,7]; END;
872 IF cumsgprc1_1 > 0.65 THEN IF cumsgprc1_1 <= 0.75 THEN D0;
873 sgprc1_1 = ((cumsgprc1_1 - pb[,8])*(sg_prca[,9]-sg_prca[,8]))/0.1 + sg_prca[,8]; END;
874 IF cumsgprc1_1 > 0.75 THEN IF cumsgprc1_1 <= 0.85 THEN D0;
875 sgprc1_1 = ((cumsgprc1_1 - pb[,9])*(sg_prca[,10]-sg_prca[,9]))/0.1 + sg_prca[,9]; END;
876 IF cumsgprc1_1 > 0.85 THEN IF cumsgprc1_1 <= 0.95 THEN D0;
877 sgprc1_1 = ((cumsgprc1_1 - pb[,10])*(sg_prca[,11]-sg_prca[,10]))/0.1 + sg_prca[,10]; END;
878 IF cumsgprc1_1 > 0.95 THEN IF cumsgprc1_1 <= 1.00 THEN D0;
879 sgprc1_1 = ((cumsgprc1_1 - pb[,11])*(sg_prca[,12]-sg_prca[,11]))/0.05 + sg_prca[,11]; END;
880
881 IF cumsgprc2_1 >= 0.00 THEN IF cumsgprc2_1 <= 0.05 THEN D0; * SORGHUM PRICE: YEAR 2 *;
882 sgprc2_1 = ((cumsgprc2_1 - pb[,1])*(sg_prca[,2]-sg_prca[,1]))/0.05 + sg_prca[,1]; END;
883 IF cumsgprc2_1 > 0.05 THEN IF cumsgprc2_1 <= 0.15 THEN D0;
884 sgprc2_1 = ((cumsgprc2_1 - pb[,2])*(sg_prca[,3]-sg_prca[,2]))/0.1 + sg_prca[,2]; END;
885 IF cumsgprc2_1 > 0.15 THEN IF cumsgprc2_1 <= 0.25 THEN D0;
886 sgprc2_1 = ((cumsgprc2_1 - pb[,3])*(sg_prca[,4]-sg_prca[,3]))/0.1 + sg_prca[,3]; END;
887 IF cumsgprc2_1 > 0.25 THEN IF cumsgprc2_1 <= 0.35 THEN D0;
888 sgprc2_1 = ((cumsgprc2_1 - pb[,4])*(sg_prca[,5]-sg_prca[,4]))/0.1 + sg_prca[,4]; END;
889 IF cumsgprc2_1 > 0.35 THEN IF cumsgprc2_1 <= 0.45 THEN D0;
890 sgprc2_1 = ((cumsgprc2_1 - pb[,5])*(sg_prca[,6]-sg_prca[,5]))/0.1 + sg_prca[,5]; END;
891 IF cumsgprc2_1 > 0.45 THEN IF cumsgprc2_1 <= 0.55 THEN D0;
892 sgprc2_1 = ((cumsgprc2_1 - pb[,6])*(sg_prca[,7]-sg_prca[,6]))/0.1 + sg_prca[,6]; END;
893 IF cumsgprc2_1 > 0.55 THEN IF cumsgprc2_1 <= 0.65 THEN D0;
894 sgprc2_1 = ((cumsgprc2_1 - pb[,7])*(sg_prca[,8]-sg_prca[,7]))/0.1 + sg_prca[,7]; END;
895 IF cumsgprc2_1 > 0.65 THEN IF cumsgprc2_1 <= 0.75 THEN D0;

```

```

896 sgprc2_1 = ((cumsgprc2_1 - pb[,8])*(sg_prca[,9]-sg_prca[,8]))/0.1 + sg_prca[,8]; END;
897 IF cumsgprc2_1 > 0.75 THEN IF cumsgprc2_1 <= 0.85 THEN D0;
898 sgprc2_1 = ((cumsgprc2_1 - pb[,9])*(sg_prca[,10]-sg_prca[,9]))/0.1 + sg_prca[,9]; END;
899 IF cumsgprc2_1 > 0.85 THEN IF cumsgprc2_1 <= 0.95 THEN D0;
900 sgprc2_1 = ((cumsgprc2_1 - pb[,10])*(sg_prca[,11]-sg_prca[,10]))/0.1 + sg_prca[,10]; END;
901 IF cumsgprc2_1 > 0.95 THEN IF cumsgprc2_1 <= 1.00 THEN D0;
902 sgprc2_1 = ((cumsgprc2_1 - pb[,11])*(sg_prca[,12]-sg_prca[,11]))/0.05 + sg_prca[,11]; END;
903
904 IF cumsgprc3_1 >= 0.00 THEN IF cumsgprc3_1 <= 0.05 THEN D0; * SORGHUM PRICE: YEAR 3*;
905 sgprc3_1 = ((cumsgprc3_1 - pb[,1])*(sg_prca[,2]-sg_prca[,1]))/0.05 + sg_prca[,1]; END;
906 IF cumsgprc3_1 > 0.05 THEN IF cumsgprc3_1 <= 0.15 THEN D0;
907 sgprc3_1 = ((cumsgprc3_1 - pb[,2])*(sg_prca[,3]-sg_prca[,2]))/0.1 + sg_prca[,2]; END;
908 IF cumsgprc3_1 > 0.15 THEN IF cumsgprc3_1 <= 0.25 THEN D0;
909 sgprc3_1 = ((cumsgprc3_1 - pb[,3])*(sg_prca[,4]-sg_prca[,3]))/0.1 + sg_prca[,3]; END;
910 IF cumsgprc3_1 > 0.25 THEN IF cumsgprc3_1 <= 0.35 THEN D0;
911 sgprc3_1 = ((cumsgprc3_1 - pb[,4])*(sg_prca[,5]-sg_prca[,4]))/0.1 + sg_prca[,4]; END;
912 IF cumsgprc3_1 > 0.35 THEN IF cumsgprc3_1 <= 0.45 THEN D0;
913 sgprc3_1 = ((cumsgprc3_1 - pb[,5])*(sg_prca[,6]-sg_prca[,5]))/0.1 + sg_prca[,5]; END;
914 IF cumsgprc3_1 > 0.45 THEN IF cumsgprc3_1 <= 0.55 THEN D0;
915 sgprc3_1 = ((cumsgprc3_1 - pb[,6])*(sg_prca[,7]-sg_prca[,6]))/0.1 + sg_prca[,6]; END;
916 IF cumsgprc3_1 > 0.55 THEN IF cumsgprc3_1 <= 0.65 THEN D0;
917 sgprc3_1 = ((cumsgprc3_1 - pb[,7])*(sg_prca[,8]-sg_prca[,7]))/0.1 + sg_prca[,7]; END;
918 IF cumsgprc3_1 > 0.65 THEN IF cumsgprc3_1 <= 0.75 THEN D0;
919 sgprc3_1 = ((cumsgprc3_1 - pb[,8])*(sg_prca[,9]-sg_prca[,8]))/0.1 + sg_prca[,8]; END;
920 IF cumsgprc3_1 > 0.75 THEN IF cumsgprc3_1 <= 0.85 THEN D0;
921 sgprc3_1 = ((cumsgprc3_1 - pb[,9])*(sg_prca[,10]-sg_prca[,9]))/0.1 + sg_prca[,9]; END;
922 IF cumsgprc3_1 > 0.85 THEN IF cumsgprc3_1 <= 0.95 THEN D0;
923 sgprc3_1 = ((cumsgprc3_1 - pb[,10])*(sg_prca[,11]-sg_prca[,10]))/0.1 + sg_prca[,10]; END;
924 IF cumsgprc3_1 > 0.95 THEN IF cumsgprc3_1 <= 1.00 THEN D0;
925 sgprc3_1 = ((cumsgprc3_1 - pb[,11])*(sg_prca[,12]-sg_prca[,11]))/0.05 + sg_prca[,11]; END;
926 * Step 6: Apply the correlated fractional deviates to projected means *;
927 cryldt_1=cryld1_1//cryld2_1//cryld3_1; * create a 3x1 matrix *;
928 sbyldt_1=sbyld1_1//sbyld2_1//sbyld3_1;
929 whyldt_1=whyld1_1//whyld2_1//whyld3_1;
930 sgyldt_1=sgyld1_1//sgyld2_1//sgyld3_1;
931 crprct_1=crprc1_1//crprc2_1//crprc3_1;
932 sbprct_1=sbprc1_1//sbprc2_1//sbprc3_1;
933 whprct_1=whprc1_1//whprc2_1//whprc3_1;
934 sgprct_1=sgprc1_1//sgprc2_1//sgprc3_1;
935
936 year1_1 = cryld1_1||sbyld1_1||whyld1_1||
937 sgyld1_1||crprc1_1||sbprc1_1||whprc1_1||sgprc1_1; * create a 1x8 matrix *;
938 year1_1=year1_1`; * transpose a 1x8 matrix *;
939
940 year2_1 = cryld2_1||sbyld2_1||whyld2_1||sgyld2_1
941 ||crprc2_1||sbprc2_1||whprc2_1||sgprc2_1;
942 year2_1=year2_1`;
943
944 year3_1 = cryld3_1||sbyld3_1||whyld3_1||sgyld3_1
945 ||crprc3_1||sbprc3_1||whprc3_1||sgprc3_1;
946 year3_1=year3_1`; * add projected means for simulation period below *;
947 projval = {118.7 37.2 53.3 43.4
948 1.960 4.520 2.910 3.232
949 121.1 37.9 54.4 44.3
950 2.000 4.710 2.990 3.375
951 123.5 38.7 55.5 45.2

```

```

952          2.060  4.870  3.090  3.482};          * add assumed expansion factors *;
953 expf =      {1.0  1.0  1.0  1.0
954             1.0  1.0  1.0  1.0
955             1.0  1.0  1.0  1.0
956             1.0  1.0  1.0  1.0
957             1.0  1.0  1.0  1.0
958             1.4  1.4  1.4  1.4};
959 coln33 = {"CornY" "SoybeansY" "WheatY" "SorghumY" "CornP" "SoybeansP"
960 "WheatP" "SorghumP"};          *column name for table 3*;
961 rown33 = {"2000" "2001" "2002"};          *row name for table 3*;
962 IF rep = 1 THEN DO;          *start if-statement *;
963   projval1 = projval[1,1:8];
964   projval2 = projval[1,9:16];
965   projval3 = projval[1,17:24];
966   projvalsum = projval1//projval2//projval3;
967   PRINT,"=====
968   =====",
969   "Table 3.3: Projected Means for Simulation Period",
970   "=====
971   =====", projvalsum [ROWNAME=rown33 COLNAME=coln33],
972   "-----
973   -----";
974   PRINT /,;;
975   expf1 = expf[1,1:8];
976   expf2 = expf[1,9:16];
977   expf3 = expf[1,17:24];
978   expfsum = expf1//expf2//expf3;
979   PRINT,"=====
980   =====",
981   "Table 3.4: Assumed Expansion Factors",
982   "=====
983   =====", expfsum [ROWNAME=rown33 COLNAME=coln33],
984   "-----
985   -----";
986   PRINT /,;;
987   END;          * end if-statement *;
988   crop24_1=year1_1//year2_1//year3_1;
989   projval_1=projval`;
990   expff=expf`;
991   cropexp_1 = crop24_1#expff;
992   cropexp1_1 = 1+cropexp_1;
993   ran1_1 = projval_1#cropexp1_1;
994   z = j(24,1,rep)||ran1_1;
995   APPEND from z;          * add observations from "z" *;
996   END;
997   CLOSE one;
998   FINISH;          * finish module *;
999   RUN rkgsim;          * run module *;
1000  QUIT;          * quit iml *;
1001  * ===== *
1002  * END OF SIMULATION PROCEDURE *
1003  * ===== *;
1004  DATA seven;          * create a SAS dataset*;
1005  SET one;
1006  OUTPUT;
1007  KEEP col2;

```

```

1008 RUN;
1009 PROC IML;                                * invoke iml      *;
1010 USE seven;                               * use SAS dataset *;
1011 t=4000;
1012 READ ALL VAR{col2} INTO x1;              * read "col2"    *;
1013 det1 = SHAPE(x1,t,24);                   * create a 24x1 matrix *;
1014 avg = det1[+,]#1/t;                       * compute a simple mean*;
1015 coln ={ "Corn_Yield1" "Soybean_Yield1" "Wheat_Yield1" "Sorghum_Yield1"
1016         "Corn_Price1" "Soybean_Price1" "Wheat_Price1" "Sorghum_Price1"
1017         "Corn_Yield2" "Soybean_Yield2" "Wheat_Yield2" "Sorghum_Yield2"
1018         "Corn_Price2" "Soybean_Price2" "Wheat_Price2" "Sorghum_Price2"
1019         "Corn_Yield3" "Soybean_Yield3" "Wheat_Yield3" "Sorghum_Yield3"
1020         "Corn_Price3" "Soybean_Price3" "Wheat_Price3" "Sorghum_Price3" };
1021 CREATE simprice FROM avg[COLNAME=coln];    * create a SAS dataset,"simprice" *;
1022 APPEND FROM avg;                          * add observations *;
1023 QUIT;                                      * quit iml *;
1024 DATA Eight;
1025 SET Seven;
1026   count+1;
1027   x1=(count/24) + 1;
1028   x=MOD(count, 24);
1029   IF x = 0 THEN x2=INT(x1)-1;
1030   ELSE x2=INT(x1);
1031   KEEP col2 x2;
1032 RUN;
1033 PROC TRANSPOSE DATA=Eight OUT=Nine ;
1034   BY x2;
1035 RUN;
1036 DATA nine; SET nine;
1037 RENAME col1 = Corn_Yield1 ; RENAME col2 = Soybean_Yield1 ;
1038 RENAME col3 = Wheat_Yield1 ; RENAME col4 = Sorghum_Yield1 ;
1039 RENAME col5 = Corn_Price1 ; RENAME col6 = Soybean_Price1 ;
1040 RENAME col7 = Wheat_Price1 ; RENAME col8 = Sorghum_Price1 ;
1041 RENAME col9 = Corn_Yield2 ; RENAME col10 = Soybean_Yield2 ;
1042 RENAME col11 = Wheat_Yield2 ; RENAME col12 = Sorghum_Yield2 ;
1043 RENAME col13 = Corn_Price2 ; RENAME col14 = Soybean_Price2 ;
1044 RENAME col15 = Wheat_Price2 ; RENAME col16 = Sorghum_Price2 ;
1045 RENAME col17 = Corn_Yield3 ; RENAME col18 = Soybean_Yield3 ;
1046 RENAME col19 = Wheat_Yield3 ; RENAME col20 = Sorghum_Yield3 ;
1047 RENAME col21 = Corn_Price3 ; RENAME col22 = Soybean_Price3 ;
1048 RENAME col23 = Wheat_Price3 ; RENAME col24 = Sorghum_Price3 ;
1049 PROC MEANS MEAN STD CV VARDEF=N MIN MAX;
1050 VAR Corn_Yield1 Soybean_Yield1 Wheat_Yield1 Sorghum_Yield1
1051      Corn_Price1 Soybean_Price1 Wheat_Price1 Sorghum_Price1
1052      Corn_Yield2 Soybean_Yield2 Wheat_Yield2 Sorghum_Yield2
1053      Corn_Price2 Soybean_Price2 Wheat_Price2 Sorghum_Price2
1054      Corn_Yield3 Soybean_Yield3 Wheat_Yield3 Sorghum_Yield3
1055      Corn_Price3 Soybean_Price3 Wheat_Price3 Sorghum_Price3;
1056 RUN;

```