

## Appendix F: SPSS Macros

SPSS is a statistical program widely used throughout the social sciences, including the field of communication. The “point-and-click” interface makes it easy to interact with SPSS using only a mouse. However, much of the powerful features of SPSS can only be accessed through the command syntax. Each statistical routine can be run by typing commands into a syntax box and then running the command. For example, the command `DESCRIPTIVES VARIABLES = tvuse age income` would compute and display a series of descriptive statistics for the variables named “tvuse” “age” and “income” in the output window. Similarly, `REGRESSION/DEP = tvuse/METHOD = ENTER income` would conduct a linear regression predicting “tvuse” from “income.”

Although most of the “point-and-click” analyses SPSS is capable of conducting have a command syntax equivalent, there are many things you might want SPSS to do that it is not capable of doing straight from the manufacturer. But it is possible to construct your own statistical routines and execute them with a new command, a command that you define, by creating a MACRO. A macro is a SPSS program that instructs SPSS what to do when you execute a custom command.

In this Appendix, I include the syntax for the macros described in this book. To use each macro, first run the macro command set (listed here, stored on the CD with this book, or by downloading them from the book’s web page at <http://www.comm.ohio-state.edu/ahayes/smcs/>). After you run the command set, nothing visible will happen. However, the macro command set is actually defining a new SPSS command. To get the macro to perform, you must then execute the command syntax that the macro constructs. The format of the command syntax is provided at the beginning of each macro listed in this appendix.

Macros remain active in SPSS until you quit the program. There is no way to make the commands a permanent part of the SPSS software. So each time you execute SPSS, you must reload the macro if you want to use it. However, the macro must be executed only once per SPSS session.

**None of the macros listed here recognize user missing values. Missing data should be entered in the data file with a period “.” character. Missing data will be deleted by the macro prior to analysis.**

### Bootstrapping a Confidence Interval for a Mean

CD/WWW file: **BOOTMEAN.SPS**

This SPSS macro estimates a population mean and generates a 95% confidence interval via bootstrapping. The syntax is `BOOTMEAN X` where `X` is a variable in the open data file. Output provides the sample mean (‘‘Sample’’), the bootstrap estimate of the population mean based on 1000 resamples with replacement (‘‘Bootstrp’’), and the lower (‘‘Lo95%CI’’) and upper (‘‘Hi95%CI’’) limits of a 95% confidence interval for the population mean based on the bootstrapped sampling distribution. The 1000 bootstrap estimates are saved as an SPSS data file on the C drive with file name `bootmean.sav`.

```
DEFINE bootmean (!POSITIONAL !TOKEN (1)).
set mxloops = 10000.
preserve.
set length = none.
set seed = random.
matrix.
get dd/variables = !1/MISSING = OMIT.
```

```

compute reps = 1000.
compute n = nrow(dd).
compute mnb = make(reps,1,0).
compute dat=dd.
compute mni=dd.
compute mn=csum(dat)/n.
loop #n = 1 to reps.
  compute v=trunc(uniform(n,1)*n)+1.
  compute dat(:,1)=dd(v,1).
  compute mnb(#n,1)=csum(dat)/n.
end loop.
compute bmn = csum(mnb)/reps.
compute mnb = {-999;mnb}.
loop #i = 2 to reps+1.
  compute ix = mnb(#i,1).
  loop #k= #i to 2 by -1.
    compute k = #k.
    do if (mnb(#k-1,1) > ix).
      compute mnb(#k,1)=mnb(#k-1,1).
    else if (mnb(#k-1,1) <= ix).
      BREAK.
    end if.
  end loop.
  compute mnb(k,1)=ix.
end loop.
compute mnb = mnb(1:reps+1,1).
compute loci = mnb(25,1).
compute hici = mnb(976,1).
compute bw={mn, bmn, loci, hici, n}.
print/title = "BOOTSTRAP MEAN ESTIMATES, 1000 RESAMPLES".
print bw/title = " "
  /clabels = "Sample" "Bootstrp" "Lo95%CI" " " "Hi95%CI" "n"
  /format f9.4.
save mnb/outfile = 'c:\bootmean.sav'.
end matrix.
RESTORE.
!END DEFINE.

```

### Bootstrapping A Confidence Interval and $p$ -value for the Difference Between Two Group Means

CD/WWW file: **BOOTDIFF.SPS**

This SPSS macro generates a 95% confidence interval for the difference between two means by bootstrapping the sampling distribution. It also generates a bootstrapped  $p$ -value to test the null hypothesis that the two population means are the same. The syntax is **BOOTDIFF var = Y/grp = X**, where **Y** is the name of the outcome variable in the open SPSS data file, and **X** is a variable that codes group membership.

The output contains descriptive statistics for the sample, the upper and lower bounds on a bootstrapped 95% confidence interval for  $\mu_1 - \mu_2$  and several bootstrapped  $p$ -values based on 1000 bootstrap resamples. The two tailed  $p$ -value (**2-tail p**) is the proportion of sample mean differences in the bootstrapped sampling distribution of the sample mean difference at least as large as the obtained mean difference in absolute value. The right-tailed  $p$ -value (**right-p**) is the proportion of sample mean differences in the bootstrapped sampling distribution greater than the obtained difference between the means. The left-tailed  $p$ -value (**left-p**) is the proportion of sample mean differences in the bootstrapped sampling distribution that are less than the opposite of the absolute value of the obtained difference between the means.

The 1,000 bootstrap estimates are saved as an SPSS data file on the C drive with file name bootn.sav.

```
DEFINE BOOTDIFF (var = !charend('')/grp = !tokens(1)).
SET MXLOOP = 100000.
preserve.
set length = none.
set seed = random.
MATRIX.
get xy/variables = !grp !var/missing = omit.
compute y = xy(:,2). compute x = xy(:,1).
compute v=design(x).
compute my = csum(y)/nrow(y).
compute ni = csum(v).
compute v = 2-v.
compute y1=make(ni(1,1),1,0).
compute y2=make(ni(1,2),1,0).
compute y1c=1.
compute y2c= 1.
loop cse = 1 to nrow(x).
  do if (v(cse,1))=1.
    compute y1(y1c,1)=y(cse,1).
    compute y1c=y1c+1.
  else if (v(cse,1))=2.
    compute y2(y2c,1)=y(cse,1).
    compute y2c=y2c+1.
  end if.
end loop.
compute y1d = y1.
compute y2d = y2.
compute m1s=csum(y1)/ni(1,1).
```

```

compute m2s = csum(y2)/ni(1,2).
compute y1t=y1-m1s+my.
compute y2t =y2-m2s+my.
compute res=make(1001,1,0).
compute res3=res.
loop bootn = 1 to 1001.
  do if (bootn > 1).
    compute u=trunc(uniform(ni(1,1),1)*ni(1,1))+1.
    compute y1=y1t(u,1).
    compute y1e = y1d(u,1).
    compute u=trunc(uniform(ni(1,2),1)*ni(1,2))+1.
    compute y2b=y2t(u,1).
    compute y2e=y2d(u,1).
    compute m1=csum(y1e)/ni(1,1).
    compute m2 = csum(y2e)/ni(1,2).
    compute res3(bootn,1)=(m1-m2).
  end if.
  do if (bootn = 1).
    compute v1 = (ni(1,1)*cssq(y1)-(csum(y1)**2))/(ni(1,1)*(ni(1,1)-1)).
    compute v2 = (ni(1,2)*cssq(y2)-(csum(y2)**2))/(ni(1,2)*(ni(1,2)-1)).
    compute v1s = sqrt(v1).
    compute v2s = sqrt(v2).
    compute m1s=csum(y1)/ni(1,1).
    compute m2s = csum(y2)/ni(1,2).
  end if.
  compute m1=csum(y1)/ni(1,1).
  compute m2 = csum(y2)/ni(1,2).
  compute res(bootn,1)=(m1-m2).
end loop.
compute res2=res(2:1001,1).
compute rest = (abs(res2) >= abs(res(1,1))).
compute ptwo=csum(rest)/(1000).
compute rest = (res2 >= abs(res(1,1))).
compute pright = csum(rest)/(1000).
compute rest = (res2 <= -abs(res(1,1))).
compute pleft = csum(rest)/(1000).
compute pval = {ptwo; pright; pleft}.
compute res3=res3(2:1001,1).
compute mnb = {-999;res3}.
loop #i = 2 to 1001.
  compute ix = mnb(#i,1).
  loop #k= #i to 2 by -1.
    compute k = #k.
    do if (mnb(#k-1,1) > ix).
      compute mnb(#k,1)=mnb(#k-1,1).
    else if (mnb(#k-1,1) <= ix).
      BREAK.
    end if.
  end loop.
  compute mnb(k,1)=ix.
end loop.
compute res3 = mnb.

```

```
print/title = "** BOOTSTRAP TWO GROUP MEAN DIFFERENCE **".
compute res2={m1s, v1s, ni(1,1); m2s,v2s, ni(1,2)}.
print res2/title = "Obtained Results"/rlabels = "group 1" "group 2"
    /clabels = "Mean" "SD" "n"/format F8.4.
compute tob=res(1,1).
compute bootn = 1000.
print tob/title = "Obtained Mean Difference (Mean 1 - Mean 2)"
    /format F8.4.
compute loci = res3(25,1).
compute hici = res3(976,1).
compute bw={loci, hici}.
print/title = "BOOTSTRAP 95% CONFIDENCE INTERVAL FOR DIFFERENCE".
print bw/title = " " /clabels = "Lo95%CI " "Hi95%CI"/format f9.4.
print pval/title = "Bootstrap p-values"
    /rlabels "2-tail p" "Right-p" "Left-p"/format F8.4.
print bootn/title = "Number of bootstrap samples"/format F8.0.
compute res2=res(2:1001,1).
compute res3=res3(2:1001,1).
save res3/outfile = 'c:\bootn.sav'.
END MATRIX.
RESTORE.
!END DEFINE.
```

### Generating Agreement Estimates From a Crosstabulation of Categorical Judgments

CD/WWW file: **AGREE.SPS**

This macro computes five measures of agreement (Holsti, Scott's  $\pi$ , Cohen's  $\kappa$ , Krippendorff's  $\alpha$ , and  $I_r$ ) for two judges asked to categorize the same set of units into a set of  $k$  categories. The syntax is `AGREE var = judge1 judge2` where `judge1` and `textttjudge2` are the names of the variables in the active SPSS data file corresponding to the judgments of judge 1 and 2, respectively. The data file is to be structured so that each object being judged is represented as a row, and the columns represent each judge's categorization of that object.

If a judge failed to code an object, it should be represented with a period (".") in the data file. Such objects will be discarded from the analysis.

```
DEFINE AGREE (var = !cmdend).
set mxloop = 1000000.
MATRIX.
get mat/file = */variables = !var/MISSING = OMIT.
compute j1 = mat(:,1).
compute j2 = mat(:,2).
compute j1d = design(j1).
compute j2d = design(j2).
compute j1e={j1,j1d}.
compute j2e={j2, j2d}.
compute j5 = {j1;j2}.
compute ad=abs(mmin(j5)).
compute j5 = j5+ad.
compute nnn = make(1,1,999).
compute res = {nnn;j5}.
loop #i = 2 to nrow(j5).
  compute ix = res(#i,1).
  loop #k= #i to 2 by -1.
    compute k = #k.
    do if (res(#k-1,1) > ix).
      compute res(#k,1)=res(#k-1,1).
    else if (res(#k-1,1) <= ix).
      BREAK.
    end if.
  end loop.
  compute res(k,1)=ix.
end loop.
compute res = res(2:nrow(j5),:).
compute j5 = res.
compute j5d = design(j5).
loop i = 1 to nrow(j5d).
  compute j5d(i,:)=j5d(i,:)*j5(i,1).
end loop.
compute j6=cmax(j5d). compute
mat2=make(ncol(j6),ncol(j6),0).
loop i = 1 to nrow(j1).
  loop k1 = 1 to ncol(j6).
```

```

do if (j1(i,1)+ad)=j6(1,k1).
    break.
end if.
end loop.
loop k2 = 1 to ncol(j6).
    do if (j2(i,1)+ad)=j6(1,k2).
        break.
    end if.
end loop.
compute k3 = {k1, k2}.
compute mat2(k1,k2)=mat2(k1,k2)+1.
end loop.
compute mat = mat2.
compute rowa = rsum(mat).
compute rowb = t(csum(mat)).
compute k = ncol(mat).
compute rowave = (rowa+rowb)/2.
compute n = csum(rowa).
compute expk = trace((rowa*t(rowb)/(n*n))).
compute exps = trace((rowave*t(rowave)/(n*n))).
compute agr = trace(mat).
compute holsti = trace(mat)/n.
compute pi = (holsti-exps)/(1-exps).
compute kappa = (holsti-expk)/(1-expk).
do if ((agr/n) >= (1/k)).
    compute ir = sqrt(((agr/n)-(1/k))*(k/(k-1))).
else.
    compute ir =0.
end if.
compute b = rowa+rowb.
compute n2 = nrow(rowa).
compute x = make(1,n2+1,0).
compute k = 0.
compute ma = x.
loop j = 1 to ((2&*n2)).
    compute k=k+1.
    compute x(1,1)=x(1,1)+1.
    loop i = 1 to n2.
        do if x(1,i) = 2.
            compute x(1,i)=0.
            compute x(1,i+1)=x(1,i+1)+1.
        end if.
    end loop.
    do if (rsum(x) = 2).
        compute ma={ma;x}.
    end if.
end loop.
compute ma = ma(2:nrow(ma),1:(ncol(ma)-1)).
loop i = 1 to nrow(ma).
    compute ma(i,:) = ma(i,*)&t(b).
end loop.
compute ma = sscp(ma).

```

```
compute pm = (csum(rsum(ma))-trace(ma))/2.
compute pf = n-agr.
compute alpha = 1- (((2*n)-1)*(pf/pm)).
print/title = "MEASURES OF AGREEMENT IN A CROSSTABULATION".
compute j6 = j6-ad.
print mat/title = "Table Being Analyzed".
print j6/title = "Data Codes Corresponding to Rows and Columns:".
compute agre = {holsti; pi; kappa; alpha; ir}.
print agre/title = " "
      /rlabels = "Holsti" "Scott pi" "Kappa" "Alpha" "I-sub-r"
      /format = F10.4.
END MATRIX.
!END DEFINE.
```



## Permutation Test of the Null Hypothesis of Random Association Between $X$ and $Y$

CD/WWW file: **PERMREG.SPS**

This macro generates one and two-tailed  $p$ -values for a permutation test of random association between  $X$  and  $Y$ . The test statistic used is Pearson's coefficient of correlation or the simple regression weight estimating  $Y$  from  $X$  (these are mathematically identical test statistics and will produce the same  $p$ -value). The obtained result ( $r$  or  $b$ ) is identified as "Obt" in the output. The syntax is `permreg dv = y/iv = x/perm = z`, where  $y$  and  $x$  are the variable names of the outcome and predictor variables, respectively, and  $z$  is the number of permutations desired.

```

DEFINE permreg (dv =!charend('/')/iv =!charend ('/'))
    /perm = !charend('/')).
set mxloop = 10000000.
preserve.
set length = none.
set seed = random.
matrix.
get xy/file = */variables = !dv !iv/names = nm/missing omit.
compute nm = t(nm).
print nm/title = "Variables:"/rlabels = "DV =" "IV =" /format a8.
compute n=nrow(xy).
compute perm = trunc(!perm).
compute res = make(perm,2,0).
compute ones = make(n,1,1).
loop i = 1 to perm.
    compute xyc = xy-(ones*(csum(xy)&/n)).
    compute vcov = (1/(n-1))*(t(xyc)*xyc).
    compute var = diag(vcov).
    compute r = mdiag(1/sqrt(var))*vcov*mdiag(1/sqrt(var)).
    compute b = mdiag(1/var)*vcov.
    compute res(i,1) = r(2,1).
    compute res(i,2) = b(2,1).
loop cse = 1 to n.
    compute k = trunc(uniform(1,1)*(n-cse+1))+cse.
    compute temp =xy(cse,2).
    compute xy(cse,2) = xy(k,2).
    compute xy(k,2) = temp.
end loop.
end loop.
compute ser = sqrt((1-(res(1,1)*res(1,1)))/(n-2)).
compute seb = sqrt((var(1,1)*(1-(res(1,1)*res(1,1))))
    /((n-2)*var(2,1))).
compute tstat = res(1,1)/ser.
compute pval=2*(1-tcdf(abs(tstat), (n-2))).
compute res2 = csum((abs(res(:,1))>= abs(res(1,1))))/perm.
compute res1g = csum((res(:,1) >=res(1,1)))/perm.
compute res1L = csum((res(:,1) <=res(1,1)))/perm.
compute se = {ser,seb;pval,pval}.
compute obt ={res(1,:);se}.
print obt/title = "Obtained Results"

```

From the CD accompanying Hayes, A. F. (2005). *Statistical Methods for Communication Science*. Mahwah, NJ: Lawrence Erlbaum Associates.

---

```
/rlabels "Obt" "se(Obt)" "sig(two)"
/clabels "r" "b"/format f8.4.
compute p={res2;res1g;res1L}.
print p/title = "Approximate Permutation Test Results"
/rlabels "Two-tail" ">= Obt" "<= Obt"
/clabels "prob"/format f8.4.
print perm/title = "Number of Permutations:".
end matrix.
restore.
!end define.
```

## Inference in Linear Regression Using a Heteroscedasticity-Consistent Standard Error Estimator

CD/WWW file: **HC3REG.SPS**

This macro generates OLS regression output but bases tests of significance for the regression parameter estimates on the HC3 estimator of the standard error (see Long & Erwin, 2000). The syntax is `HC3REG dv = outcome/iv = varlist` where `outcome` is the name of the outcome variable being estimated and `varlist` is a list of one or more predictor variables. Parts of this macro were written by Li Cai.

```

DEFINE hc3reg (dv =!charend('')/iv =!charend ('')).
sex mxloop = 100000000.
MATRIX.
get x/file = */variables = !dv !iv/names = dv/missing = omit.
compute y=x(:,1).
compute x=x(:,2:ncol(x)).
compute pr = ncol(x).
compute n = nrow(x).
compute iv = t({"Constant", dv(1,2:ncol(dv))}).
compute con = make(n,1,1).
compute x={con,x}.
compute dv=dv(1,1).
compute b = inv(t(x)*x)*t(x)*y.
compute k = nrow(b).
compute invXtX = inv(t(x)*x).
compute h = x(:,1).
loop i=1 to n.
    compute h(i,1)= x(i,:)*invXtX*t(x(i,:)).
end loop.
compute resid = (y-(x*b)).
compute pred = x*b.
loop i=1 to k.
    compute x(:,i) = (resid&/(1-h))&*x(:,i).
end loop.
compute hc = invXtX*t(x)*x*invXtX.
print dv/title = "Criterion Variable"/format A8.
compute sebh = sqrt(diag(hc)).
compute te = b&/sebh.
compute p = 2*(1-tcdf(abs(te), n-nrow(b))).
compute oput = {b,sebh, te, p}.
print oput
    /title = "HC3 Heteroscedasticity-Consistent Regression Results"
    /clabels = "B(OLS)" "SE(HC3)" "t" "P>|t|"/rnames = iv/format f10.4.
END MATRIX.
!END DEFINE.

```

## A Randomization Test Equivalent of a Single Factor ANOVA

CD/WWW file: **RANDGRP.SPS**

This macro generates a  $p$ -value for an  $F$ -ratio from an analysis of variance through random reassignment of the outcome variable scores to groups. The syntax is `randgrp y = dv/grp = group/nperm = z` where `dv` is the variable name of the dependent variable, `group` is the variable name that holds the group codes, and `z` is the number of permutations desired.

The  $F$  values from the rerandomizations are saved as an SPSS data file with the file name "permdisf.sav".

```

DEFINE RANDGRP (y = !charend('/')/grp = !charend('/')/nperm = !charend('/')).
SET MXLOOPS = 10000001.
preserve.
set length = none.
set seed = random.
MATRIX.
get dat/variables = !grp !y/MISSING = OMIT.
compute n = nrow(dat).
compute y = dat(:,2).
compute grp = dat(:,1).
compute v=design(grp).
compute my = csum(y)/nrow(y).
compute ngrp = t(csum(v)).
compute gsum = t(v)*y.
compute gps=nrow(ngrp).
loop i = 1 to ncol(v).
  loop j = 1 to n.
    do if v(j,i) = 1.
      compute dat(j,1)=i.
    end if.
  end loop.
end loop.
compute mns = gsum&/ngrp.
compute mns = {ngrp,mns}.
print mns/title = " Group Statistics"/clabels = "n" "Mean"/format F12.4.
compute results=uniform(!nperm,1).
compute pos=0.
compute sxs=(cssq(dat)).
compute sxs=sxs(1,2).
compute sts=csum(dat).
compute sts=(sts(1,2)*sts(1,2))/n.
loop perm = 1 to !nperm.
  loop ce = 1 to n.
    do if perm > 1.
      compute k = trunc(uniform(1,1)*(n-ce+1))+ce.
      compute temp =dat(ce,2).
      compute dat(ce,2) = dat(k,2).
      compute dat(k,2) = temp.
    end if.
  end loop.
  compute ttl=0/ngrp.

```

```
loop ce=1 to n.
  compute ttl(dat(ce,1),1)=ttl(dat(ce,1),1)+dat(ce,2).
end loop.
compute test = 0.
loop ce = 1 to nrow(ngrp).
  compute test = test+((ttl(ce,1)*ttl(ce,1))/ngrp(ce,1)).
end loop.
compute results(perm,1)=test.
end loop.
loop k = 1 to !nperm.
  do if results(k,1) >= results(1,1).
    compute pos=pos+1.
  end if.
end loop.
compute results = ((results-sts)/(gps-1))/((sxs-results)/(n-gps)).
compute p = {abs(results(1,1)),(pos!/nperm)}.
print p/title = "Results of Approximate Randomization Test"/clabels = " F"
      "p-value"/format F12.4.
print !nperm/title = "Number of randomizations requested".
save results/outfile = 'permdisf.sav'/variables perm.
END MATRIX.
restore.
!END DEFINE.
```

### A Demonstration of the Sampling Distribution of the Sample Mean

CD/WWW file: MEANSAMP.SPS

This macro will empirically generate an approximation of the sampling distribution of the sample mean when taking a random sample of size =  $n$  from the population stored on the C drive in file name POP.SAV. This file should contain a single column labeled  $Y$ , with measurements of each member of the population on  $Y$  residing in each row. After activating the macro, the syntax is `meansam size = n samples = rep`, where  $n$  is the sample size and `rep` is the number of samples from this population desired. Each sample mean will be displayed as a row in the active data window.

```
DEFINE MEANSAMP (size = !tokens(1)/samples = !cmdend).
preserve.
SET MXLOOP = !samples.
set seed = random.
MATRIX.
compute s = !size.
compute sam = !samples.
compute result = uniform(sam,1).
get tv/file = 'c:\pop.sav'/variable = y/missing = omit/sysmis = omit.
compute n = nrow(tv).
compute popmean = csum(tv)/n.
do if (s > n).
  print/title = "ERROR: SAMPLE SIZE IS LARGER THAN POPULATION".
  print/title = "DISREGARD OUTPUT THAT FOLLOWS".
  end.
end if.
loop rep = 1 to sam.
  loop cse = 1 to n.
    compute k = trunc(uniform(1,1)*(n-cse+1))+cse.
    compute temp=tv(cse,1).
    compute tv(cse,1) = tv(k,1).
    compute tv(k,1) = temp.
  end loop.
  compute sample = tv(1:s,1).
  compute result(rep,1) = csum(sample)/s.
end loop.
save result/outfile = */variables = smean.
compute mat = {s;sam;popmean; n}.
print mat/title = "Simulation Characteristics"/
  rlabels = "n" "Samples" "PopMean" "PopSize"/format = F10.4.
END MATRIX.
GRAPH/HISTOGRAM=smean.
restore.
!END DEFINE.
```

## A Demonstration of the Sampling Distribution of the Sample Correlation

CD/WWW file: **RSAMP.SPS**

This macro will simulate sampling from a bivariate population with correlation  $\rho$ . After the macro is activated, the syntax is **rsamp size = n samples = reps rho = corr**, where **n** is the size of each sample, **reps** is the number of samples of  $n$  you want to take, and **corr** is the population correlation between the two variables. Each sample is stored in a row of the active data matrix.

```
DEFINE RSAMP (size = !tokens(1)/samples = !tokens(1)/rho = !cmdend).
preserve.
SET MXLOOP = !samples.
set seed = random.
MATRIX.
compute rpop = !rho.
compute reps = !samples.
compute n = !size.
compute res = make(reps,1,1).
compute ones = make(n,1,1).
loop i = 1 to reps.
  compute x = sqrt(-2*ln(uniform(n,1)))&cos((2*3.14159265358979)*uniform(n,1)).
  compute y = rpop*x+sqrt(1-(rpop*rpob))*sqrt(-2*ln(uniform(n,1)))&
    cos((2*3.14159265358979)*uniform(n,1)).
  compute xy={x,y}.
  compute xyc = xy-(ones*(csum(xy)&/n)).
  compute vcov = (1/(n-1))*(t(xyc)*xyc).
  compute var = diag(vcov).
  compute r = mdiag(1/sqrt(var))*vcov*mdiag(1/sqrt(var)).
  compute res(i,1) = r(2,1).
end loop.
compute mat = {n;reps;rpob}.
print mat/title = "Simulation Characteristics"/rlabels = "n" "Samples" "rho"
  /format = F10.4.
save res/outfile = */variables = r.
end matrix.
restore.
GRAPH/HISTOGRAM=r.
!END DEFINE.
```