Supplement

Supplemental materials containing Notation and SAS programs and plots for the examples and simulations in "A Randomization Permutation (RP) Test for Single Subject Mediation", *Evaluation & the Health Professions*

Notation

Notation for the RP CI of the Mediation Effect

Notation	***Description***
X	observed predictor
M	observed mediator
Y	observed outcome
$\hat{M}$	predicted value of mediator ($\hat{M} = M - e_m$)
$\hat{Y}$	predicted value of outcome ($\hat{Y} = Y - e_y$)
M*	permuted value of mediator ($M^* = \hat{M} + e_m^*$)
Y*	permuted value of outcome ($Y^* = \hat{Y} + e_y^*$)
e_m	residual from regression of M on X in observed data
e_y	residual from regression of Y on M and X in observed data
e_m^*	permuted residual from regression of M on X reassigned to unpermuted data
e_y^*	permuted residual from regression of Y on M and X reassigned to unpermuted data
a	coefficient for X from regression of M on X in observed data
b	coefficient for M from regression of Y on X and M in observed data
c'	coefficient for X from regression of Y on X and M in observed data
ab	mediated effect in observed data
*a**	permuted coefficient for X from regression of M* on X
*b**	permuted coefficient for M from regression of Y* on X and M
*c'**	permuted coefficient for X from regression of Y* on X and M
*a*b**	mediated effect in permuted datasets

Example 5.1 Randomization to Treatment Condition

Our first example describes a single-case experimental design for understanding processes that may confer risk for developing an alcohol use disorder in a high-risk individual. We pattern the hypothetical data based on a study by Mayhugh et al. (2018) who examined patterns of momentary stress and craving among non-dependent, moderate-heavy alcohol users who drank as usual for three days and abstained from drinking for three days. Mayhugh et al. (2018) found evidence that on average drinking relieved stress relative to abstinence, and higher stress was associated with greater cravings. Data from one participant is shown in Table S1. Unstandardized estimates were derived by using the RP program with the raw data. The participant was randomly assigned days in which they could drink or had to abstain, in order to investigate the effect of abstinence on craving levels via increased stress. Table S2 contains the regression estimates for the unpermuted data. Results of the RP test suggest that, for this participant, abstinence reduced stress, which in turn predicted higher craving levels (median mediated effect of -.63 with 95% confidence limits: [-1.36, -0.01]). The estimates include adjustment for dependency in stress (lag 1 estimate = 0.07, $p = 0.67$) in the mediator equation and the dependency in cravings (lag 1 estimate = 0.29, $p = 0.07$) and stress (lag 1 estimate = -0.22, $p = 0.20$) in the outcome equation. Note that the mediated effect for this participant is opposite from the overall results of Mayhugh et al. (2018) demonstrating how single subject analysis may obtain different results for different participants.

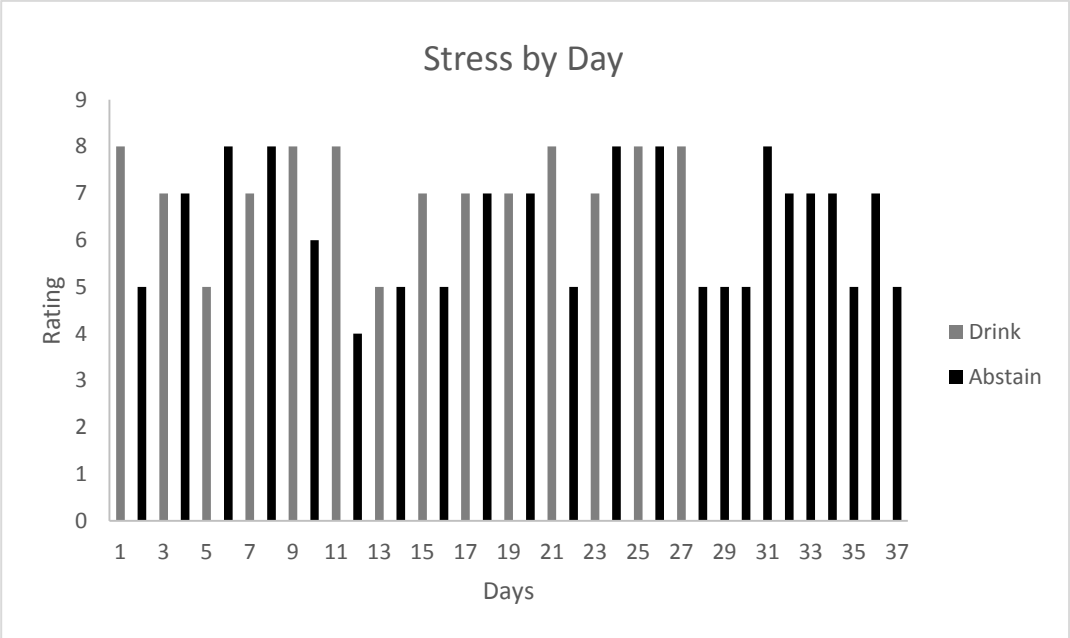

Figure S1. Stress by Type of Drinking Day

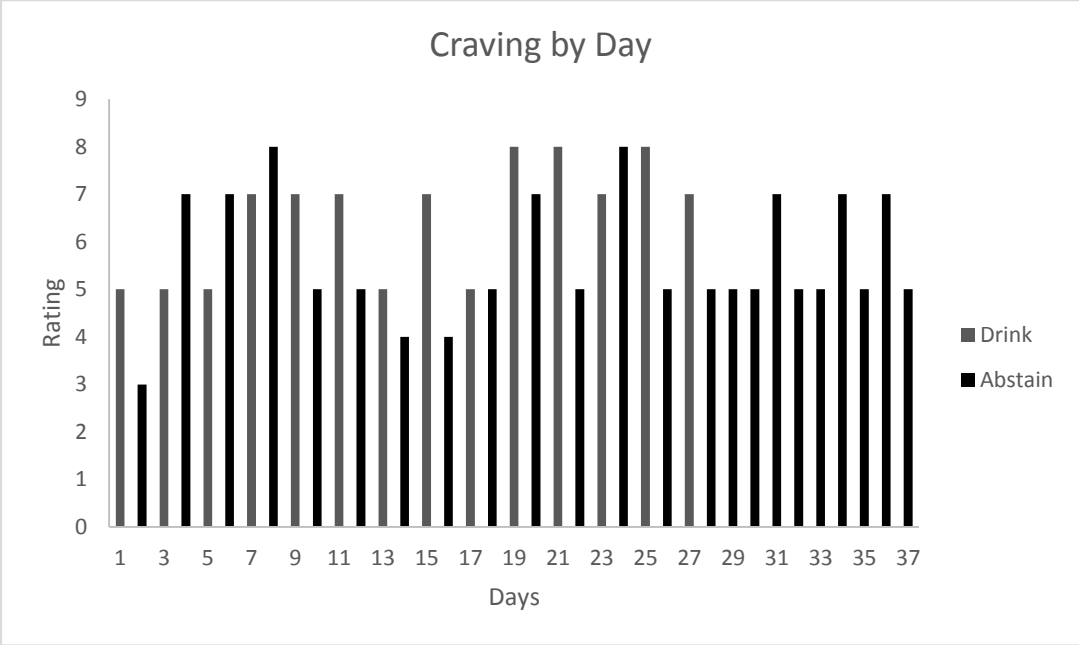

Figure S2. Craving by Type of Drinking Day

Data for Stress and Craving Example

Table S1. Example Data for Participant 18 (patterned on Mayhugh et al., 2018)

Day	Treatment Condition	Stress	Craving
1	0	8	5
2	1	5	3
3	0	7	5
4	1	7	7
5	0	5	5
6	1	8	7
7	0	7	7
8	1	8	8
9	0	8	7
10	1	6	5
11	0	8	7
12	1	4	5
13	0	5	5
14	1	5	4
15	0	7	7
16	1	5	4
17	0	7	5
18	1	7	5
19	0	7	8
20	1	7	7
21	0	8	8
22	1	5	5
23	0	7	7
24	1	8	8

25	0	8	8
26	1	8	5
27	0	8	7
28	1	5	5
30	1	5	5
32	1	5	5
33	1	8	7
34	1	7	5
35	1	7	5
36	1	7	7
37	1	5	5
38	1	7	7
39	1	5	5

Note. Treatment Condition: 1 = abstained days. 0 = normal drinking days. Stress rating: 0-10, with 0 "no stress" and 10 "extreme stress." Craving rating: 0 – 10, with 0 "no craving" and 10 "extreme craving."

Table S2. Coefficients for data analysis of stress and craving example observed data

Effect	Estimate	SE	t-value	*p*
Model for M (Stress)				
Intercept	6.62	1.13	5.88	<.0001
X (treatment)	-0.84	0.44	-1.93	0.06
mlag	0.07	0.17	0.42	0.67
Model for Y (Craving)				
intercept	.92	1.17	0.79	0.44
X (treatment)	-0.46	0.33	-1.37	0.18
M (stress)	0.77	0.12	6.16	<.0001
mlag	-0.22	0.17	-1.30	0.20
ylag	0.29	0.16	1.86	0.07

Examples 5.2 – Randomization to Treatment Dose

Randomization tests can also be employed to examine dose-response relationships. Examining dose-response relationships may uncover for whom certain foods or food additives have an adverse effect, as well as the process through which these individuals are affected. For example, although the majority of children do not have a behavioral response to food additives, some children do exhibit a significant increase in their negative behaviors following consumption of artificial food colors (Weiss et al., 1980). Randomization tests can evaluate potential mechanisms, such as the secretion of histamine, by which food additives influence behavior for each affected child (Stevens et al., 2013). We present data in Table S2 based on a hypothetical study where the participant was randomly assigned to different doses of food additives on different measurement occasions, and repeated measures of plasma histamine and behavior problems were assessed. Unstandardized estimates were derived by using the RP program with the raw data. Table S4 contains the regression estimates for the unpermuted data. Results of the RP test suggest that, for this participant, food additives were related to plasma histamine, which in turn predicted more behavior problems (median mediated effect = .043 with 95% confidence limits: [0.02, 0.08]). The estimates include adjustment for dependency in plasma histamine (lag 1 estimate = 0.02, $p = 0.90$) in the mediator equation, and the dependency in behavior problems (lag 1 estimate = 0.02, $p = 0.41$) and histamine (lag 1 estimate = -.24, p=0.22) in the outcome equation.

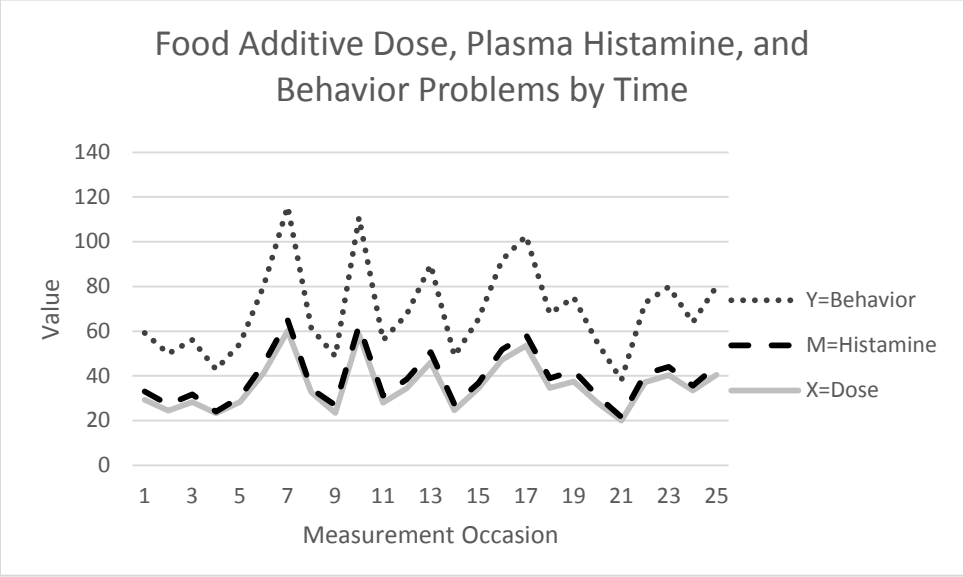

Figure S3. Food Additive Dose, Plasma Histamine, and Behavior Problems by Measurement Occasion

Data Generated for Food Additive Example

Table S3. Generated Food Additive Data

Food additive dose	Plasma histamine	Behavior problems
29.26026229	3.791956763	26.14736031
24.45227327	2.791556607	22.70840516
28.37732128	3.429541122	24.2316014
23.25452487	0.801804582	18.86195983
28.33333934	2.113872915	23.89892719
41.29237438	4.015345574	34.91735358
60.15941034	4.845555268	50.70281567
32.56765113	1.679352154	27.02652129
23.47745044	3.376875053	21.93325987
58.50102009	4.00270083	48.39413637
28.05398867	3.117020765	24.73818423
34.49790959	3.952123632	29.08729129
46.13176752	4.526035094	38.98349655
24.62263289	2.715782145	21.61648996
34.44642871	2.251969859	28.64847749
47.14375366	4.690264508	40.08234145
53.45387818	4.868659553	44.21529719
34.58383914	4.28629195	28.92258825
37.42956535	4.982098686	32.70912239
28.05750829	3.011423054	23.6612125
19.96655761	1.563138217	16.66333264
37.24464284	3.629012636	31.86173776

40.36000065	3.682294551	35.79719281
33.45210155	2.242046111	28.27607536
40.52417633	3.999040922	35.39336362

Table S4. Coefficients for data analysis of food additive example observed data

Effect	Estimate	SE	t-value	*p*
Model for M (Plasma histamine)				
Intercept	0.60	0.74	0.80	0.43
X (food additive dose)	0.08	0.02	4.75	0.0001
mlag	0.02	0.15	0.13	0.90
Model for Y (Behavior problems)				
intercept	0.99	0.70	1.42	0.17
X (food additive dose)	0.76	0.02	39.98	<.0001
M (plasma histamine)	0.59	0.19	3.18	0.01
mlag	-0.24	0.19	-1.27	0.22
ylag	0.02	0.02	0.84	0.41

Similarly, single-case experimental designs may be useful in evaluating the mechanisms through which short-acting pharmacological interventions achieve their effects. Randomization tests could evaluate whether the effect of oxytocin, administered as a nasal spray, on increased cooperation operates via increased recognition of happy faces (Gossen et al., 2012; Rilling et al., 2011; Schulze et al., 2011; Shin et al., 2015; Shin et al., 2018). Table S3 shows data from a hypothetical study where treatment times were randomly assigned to intranasally-administered oxytocin doses, and measures of a participant's facial emotion recognition and prosocial behavior were taken at each treatment occasion. Unstandardized estimates were derived by using the RP program with the raw data. Table S6 contains the regression estimates for the unpermuted data. Results of the RP test suggest that, for this participant, oxytocin was not related to prosocial behavior via facial emotion recognition because the confidence interval contains zero (median mediated effect = .02 with 95% confidence limits: [-0.00, 0.05]). The estimates include adjustment for dependency in facial emotion recognition, (lag 1 estimate = -0.05, p = 0.75) in the mediator equation and the dependency in prosocial behavior (lag 1 estimate = 0.00, p = 0.98) and facial emotion recognition (lag 1 estimate = -.15, p =.55) in the outcome equation.

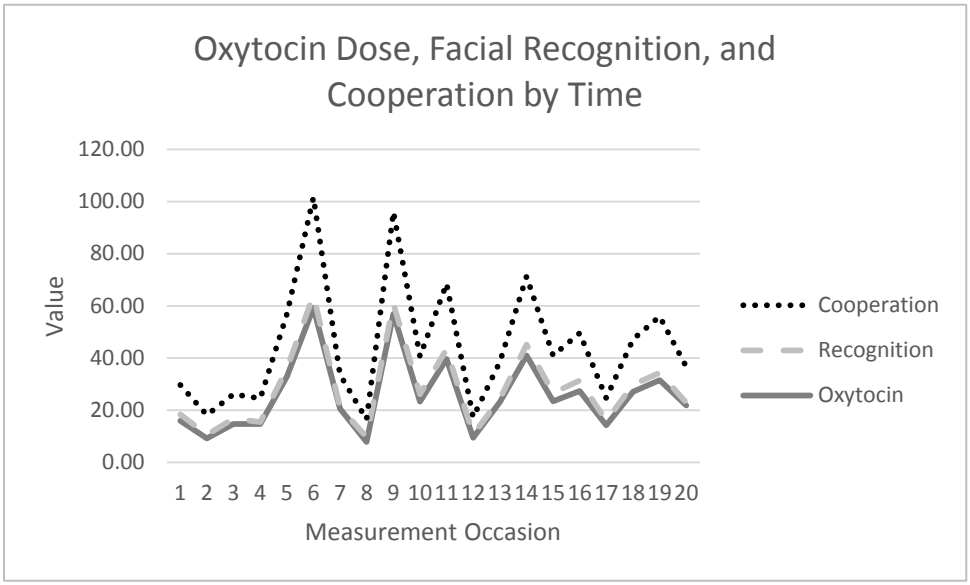

Figure S4. Oxytocin Dose, Facial Recognition, and Cooperation by Measurement Occasion

Data Generated for Oxytocin Example

Table S5. Generated Oxytocin Data

Intranasal oxytocin dose	Facial recognition	Cooperation
15.96	2.46	11.27
9.23	1.27	7.65
14.73	2.06	9.34
14.67	0.75	9.14
32.81	3.17	20.74
59.22	4.75	37.58
20.59	0.48	12.56
7.87	1.82	6.75
56.90	3.84	35.25
23.30	2.83	14.51
39.58	3.87	25.05
9.47	1.20	6.57
23.23	1.13	14.24
41.00	4.08	26.19
23.42	3.17	14.32
27.40	3.98	18.21
14.28	1.63	8.79
27.14	2.62	17.48
31.50	2.80	21.60
21.83	1.08	13.81

Table S6. Coefficients for data analysis of oxytocin example observed data

Effect	Estimate	SE	t-value	*p*
Model for M (Facial recognition)				
Intercept	0.74	0.57	1.29	0.22
X (oxytocin dose)	0.07	0.01	5.34	<.0001
mlag	-0.05	0.15	-0.33	0.75
Model for Y (Cooperation)				
intercept	0.88	0.55	1.61	0.13
X (oxytocin dose)	0.59	0.02	29.95	<.0001
M (facial recognition)	0.35	0.22	1.55	0.14
mlag	-0.15	0.24	-0.62	0.55
ylag	0.00	0.03	0.03	0.98

SAS Program for Data Examples

```
*import data from Table S1, S2, or
S3 as a;

/**Creating lag variables**/
data a1;
set a;
mlag = lag1(m);
ylag = lag1(y);
run;

/*This program conducts mediation
permutation tests*/

%macro
permmed(dataname,x,m,y,npermute,max
iter,seed,mlag,ylag);

%* Make a listwise deleted dataset.
;
data listwise; set &dataname;
      if (&x ne .) and (&m ne .)
and (&y ne .);
/*    if (&x ne .) and (&m ne .)
and (&y ne .) and (&mlag ne .) and
(&ylag ne .);*/
      rename
            &x = x &m = m &y = y;
/*          &x = x &m = m &y = y
&mlag = mlag &ylag = ylag;*/
run;

%* Find number of cases in listwise
deleted dataset. ;
proc means data=listwise noprint;
      output out=meanout n(x) =
nobs;
run;
data _NULL_; set meanout;
      call symput('nobs',nobs);
run;

%* Model 2: Regress y on x, m mlag
ylag ;
proc reg data=listwise
outest=model2 tableout noprint;
      model y = x m mlag ylag;

%* Save predicted values and
residuals, which are needed for
permutation tests. ;
      output out=predres2 p=yhat
r=yres;
run;

%* Model 3: Regress m on x mlag;
proc reg data=listwise
outest=model3 tableout noprint;
      model m = x mlag;

%* Save predicted values and
residuals, which are needed for
permutation tests. ;
      output out=predres3 p=mhat
r=mres;
run;

%* Gather results. ;
data parm2; set model2;
      if _TYPE_='PARMS';
      b = m; cprime = x; b02 =
intercept; byl = ylag; by2 = mlag;
keep b cprime b02 byl by2;
run;


data se2; set model2;
      if _TYPE_='STDERR';
      seb = m; keep seb;
run;

data parm3; set model3;
      if _TYPE_='PARMS';
      a = x; b03 = intercept; keep
a b03; bml = mlag; keep a b03 bml;
run;
data se3; set model3;
      if _TYPE_='STDERR';
      sea = x; keep sea;
run;
data origresult; merge parm2 se2
parm3 se3;
      vara = sea*sea;
      varb = seb*seb;
      sobelse = sqrt(a*a*varb +
b*b*vara);
run;

%* Merge predicted values and
residuals for models 2 and 3. ;
data predres; merge predres2
predres3;
run;

%*** Non-iterated permutation
confidence limits *** ;

%* Make npermute copies of the
original data. ;
```

```
proc iml;
      use predres;
      read all var{ x m mlag y ylag
yhat yres mhat mres} into orig;
      copies = orig;
      do i = 2 to &npermute;
            copies = copies//orig;
      end;
      varnames = {'x' 'm' 'mlag'
'y' 'ylag' 'yhat' 'yres' 'mhat'
'mres'};
      create origcopies from
copies[colname = varnames];
      append from copies;
quit;

%* Make shuffling variables to
permute the residuals mres and
yres. ;
data origcopiesbycopy; set
origcopies;
      copynum = ceil(_N_/&nobs);
      shufflevaryres =
ranuni(&seed);
      shufflevarmres = ranuni(0);
run;

%* Make a dataset with yres
shuffled within each dataset copy.
;
proc sort data=origcopiesbycopy
out=shuffleyres;
      by copynum shufflevaryres;
run;

%* Make a dataset with mres
shuffled within each dataset copy.
;
proc sort data=origcopiesbycopy
out=shufflemres;
      by copynum shufflevarmres;
run;

%* Merge shuffled residual mres and
yres with original data in each
dataset copy. ;
data shuffled; merge
origcopiesbycopy(keep=copynum x m
mlag y ylag mhat yhat)
shuffleyres(keep=yres)
shufflemres(keep=mres);
      %* Find mstar and ystar, the
new values of m and y based on
shuffling the residuals. ;
      mstar =     mhat + mres;
      ystar = yhat + yres;
run;

%* Model 2: Regress ystar on x, m.
;
proc reg data=shuffled noprint
outest=model2shuf;
      by copynum;
      model ystar = x m mlag ylag;
run;

%* Model 3: Regress mstar on x. ;
proc reg data=shuffled noprint
outest=model3shuf;
      by copynum;
      model mstar = x mlag;
run;

%* Gather results. ;
data model2shuf; set model2shuf;
      if _TYPE_ = 'PARMS';
      bperm = m; keep bperm;
run;
data model3shuf; set model3shuf;
      if _TYPE_ = 'PARMS';
      aperm = x; keep aperm;
run;
data shufresult; merge model2shuf
model3shuf;
run;


%* Include results for original
data. ;
data origshufresult; set
origresult(rename=(a=aperm
b=bperm)) shufresult;
      abperm = aperm*bperm;
run;

%* Get confidence limits. ;
proc univariate data=origshufresult
noprint;
      var abperm;
      output out=permcl pctlpts =
2.5 97.5 50 pctlpre = perm pctlname
= lcl ucl median;
run;


%*** Iterated permutation
confidence limits *** ;

%* Iterate twice: 1 = lower
confidence limit, 2 = upper
confidence limit ;
```

```
%do direction = 1 %to 2;

      data clguess; set origresult;
            %if &direction = 1
%then %do;
                  clab = a*b -
1.96*sobelse;
            %end;
            %else %do;
                  clab = a*b +
1.96*sobelse;
            %end;
      run;

      %* Initialize loop counter
and break checker. ;
      %let loop = 0;
      %let break = 0;

      %* Search for confidence
limit. ;
      %do %until ((&break = 1) or
(&loop = &maxiter));

            %* Increment loop
counter. ;
            %let loop =
%eval(&loop+1);

            %* Analyze confidence
limit into its components. ;
            data clguess; set
clguess;
                  %* Make up terms
for quadratic equation solutions. ;
                  seaoverseb =
sea/seb;
                  term1 = (-1)*(a-
(b*seaoverseb));
                  term2 = (a-
(b*seaoverseb))**2;
                  term3 =
4*seaoverseb*clab;
                  term4 =
2*seaoverseb;
                  term5 =
(a+(b*seaoverseb));
                  term6 =
(a+(b*seaoverseb))**2;
                  %* Find the two
possible solutions for clb. ;
                  %if &direction =
1 %then %do;

      clb1=(term5+sqrt(term6-
term3))/term4;

      clb2=(term5-sqrt(term6-
term3))/term4;
                  %end;
                  %else %do;

      clb1=(term1+sqrt(term2+term3)
)/term4;

      clb2=(term1-
sqrt(term2+term3))/term4;
                  %end;
                  %* Pick the
solution that puts clb closer to b.
;
                  clb1dist =
abs(clb1-b);
                  clb2dist =
abs(clb2-b);
                  if clb1dist <
clb2dist then clb = clb1;
                  else clb = clb2;
                  %* Make cla ;
                  cla = clab/clb;
      %* Make clab into a macro
parameter for easier reference ;
                  call
symput('clab',clab);
            run;

            %* Merge confidence
limit components with copies of
original data created above. ;
            %* This will allow for
new predicted values and residuals
to be made based on ;
            %* using the confidence
limit components rather than the
sample values. ;
            data origcopiesbycopyi;
if _N_=1 then set clguess(keep=cla
clb cprime b02 b03 byl bml); set
origcopiesbycopy;


%* Make new residuals based on cla
and clb and sample values of other
coefficients ;
                  %* (cprime, b02,
and b03). ;
                  yhati = b02 +
clb*m + cprime*x + by2*mlag +
byl*ylag;
                  yresi = y -
yhati;
                  mhati = b03 +
cla*x + bml*mlag;
```

```
                  mresi = m -
mhati;
            run;

            %* On the first loop,
make variables for later shuffling
of yresi and mresi. ;
            %if &loop = 1 %then
%do;

                  data shufvars;
                        do i = 1 to
&nobs*&npermute;

      shufflevaryresi =
ranuni(&seed);

      shufflevarmresi = ranuni(0);

      output;
                        end;
                  run;

            %end;

            %* On subsequent loops,
re-use same shuffling variables
made up on the first loop. ;

            %* Make a dataset with
yresi shuffled within each dataset
copy. ;
            data shuffleyresi;
merge origcopiesbycopyi
shufvars(keep=shufflevaryresi);
            run;
            proc sort
data=shuffleyresi;
                  by copynum
shufflevaryresi;
            run;

            %* Make a dataset with
mresi shuffled within each dataset
copy. ;
            data shufflemresi;
merge origcopiesbycopyi
shufvars(keep=shufflevarmresi);
            run;
            proc sort
data=shufflemresi;
                  by copynum
shufflevarmresi;
            run;

            %* Merge shuffled
residuals yresi and mresi with
original data in each dataset copy.
;
            data shuffledi; merge
origcopiesbycopyi
shuffleyresi(keep=yresi)
shufflemresi(keep=mresi);
                  %* Calculate
ystari and mstari, the new values
of y and m based on shuffling the
residuals. ;
                  ystari = yhat +
yresi;
                  mstari = mhat +
mresi;
            run;

            %* Model 2: Regress
ystari on x, m. ;
            proc reg data=shuffledi
noprint outest=model2shufi;
                  by copynum;
                  model ystari = x
m mlag ylag;
            run;

            %* Model 3: Regress
mstari on x. ;
            proc reg data=shuffledi
noprint outest=model3shufi;
                  by copynum;
                  model mstari = x
mlag;
            run;

            %* Gather results. ;
            data model2shufi; set
model2shufi;
                  if _TYPE_ =
'PARMS';
                  bpermi = m; keep
bpermi;
            run;
            data model3shufi; set
model3shufi;
                  if _TYPE_ =
'PARMS';
                  apermi = x; keep
apermi;
            run;
            data shufresulti; merge
model2shufi model3shufi;
                  abpermi =
apermi*bpermi;
            run;

            %* Include current clab
guess value. ;
```

```
            data origshufresulti;
set clguess(rename=(clab=abpermi))
shufresulti;
            run;

            %* Make a frequency
table with cumulative frequencies
(percentile ranks) ;
            %* of abpermi. ;
            proc freq
data=origshufresulti noprint;
                  tables abpermi
/out=freqtable outcum;
            run;

            %* Merge current
confidence limit guess, clab, with
frequency table. ;
            %* Find percentile rank
of current confidence limit guess,
clab, ;
            %* in the distribution.
;
            %* Do this by choosing
the nearest value of abpermi (they
should be equal ;
            %* within rounding) and
getting its percentile rank. ;
            data freqtable; set
freqtable;
                  clab = &clab;
                  diff = abs(clab -
abpermi);
            run;
            proc sort
data=freqtable;
                  by diff;
            run;
            data getptilerank; set
freqtable;
                  if _N_ = 1;
                  prankclab =
CUM_PCT;
                  %* Check if error
is small enough to exit the loop. ;
                  %if &direction =
1 %then %do;
                        abserror =
abs(prankclab - 2.5);
                  %end;
                  %else %do;
                        abserror =
abs(prankclab - 97.5);
                  %end;
                  if abserror le
0.5 then break = 1;
                  else break = 0;
                  %* Save break as
a macro variable so the loop can
check its value. ;
                  call
symput('break',break);
            run;

            %* If looping
continutes, choose as the next
guess for the value of the
confidence ;
            %* limit the value of
abpermi at the target percentile
rank. ;
            proc univariate
data=origshufresulti noprint;
                  var abpermi;
                  output
out=univout pctlpts = 2.5 97.5
pctlpre = clab pctlname = lower
upper;
            run;

            %* Merge the next guess
of clab back into the clguess
dataset to be ;
            %* read at the top of
the loop. ;
            data clguess; merge
clguess(drop=clab) univout;
                  %if &direction =
1 %then %do;
                        clab =
clablower;
                  %end;
                  %else %do;
                        clab =
clabupper;
                  %end;
            run;

      %end;

      %* Save the confidence limit
that was just found. ;
      %if &direction = 1 %then %do;
            data loweri; length
ilowerstatus $ 15; set
getptilerank;
                  if abserror le
0.5 then do;
                        ipermlcl =
clab;

      ilowerstatus = '(converged)';
                  end;
                  else do;
```

```
                        ipermlcl =
.;

      ilowerstatus = '(not
converged)';
                  end;
            run;
      %end;
      %else %do;
            data upperi; length
iupperstatus $ 15; set
getptilerank;
                  if abserror le
0.5 then do;
                        ipermucl =
clab;

      iupperstatus = '(converged)';
                  end;
                  else do;
                        ipermucl =
.;

      iupperstatus = '(not
converged)';
                  end;
            run;
      %end;
%end;
```

```
%* Merge lower and upper iterated
confidence limits with non-iterated
confidence limits. ;
data allresult; merge permcl loweri
upperi;
      file print;
      put 'Permutation 95%
confidence limits'
            /'Lower: ' permlcl
            /'Upper: ' permucl
            /'Median: ' permmedian
            /'Iterated permutation
95% confidence limits'
            /'Lower: ' ipermlcl ' '
ilowerstatus
            /'Upper: ' ipermucl ' '
iupperstatus;
run;

%mend permmed;
/**run code below to get
unstandardized estimates**/

%permmed(dataname=a1,x=x,m=m,y=y,np
ermute=500,maxiter=1,seed=52343);
title2 'Results from Study of Food
Additives and Plasma Histamine and
Behavior Problems';
proc print data=allresult; run;
```

Simulation Programs

The RP simulation program has three parts. In the first program, data are generated, in the second the RP test is conducted, and in the third, the results are aggregated.

Data Generation Program

The data generation program requires a CSV files, labeled here as "N1PermConditions.csv" which contains parameter information for each simulated condition.

The following program generates data with a .8 lag on both M and Y. The user can change the magnitude of the lag by replacing the second argument to define the variables "phim" and "phiy". Note: for lag with a positive value, a negative value must be input.

```
PROC IMPORT OUT= WORK.conds

      DATAFILE="\N1PermConditions.c
sv"
            DBMS=CSV REPLACE;
     GETNAMES=YES;
     DATAROW=2;
RUN;

OPTIONS PS=59 LS=80 REPLACE
NONOTES;
FILENAME NULLOG DUMMY 'C:\NULL';
proc printto log = null;
RUN;

data conds; length ii $8; set
conds;
i+1;
ii=left(put(i, 8.));
call symput ('FILE'||ii, file);
call symput ('NREP'||ii, NREP);
call symput ('NOBS'||ii, NOBS);
call symput ('BMX'||ii, BMX);
call symput ('BYM'||ii, BYM);
call symput ('BYX'||ii, BYX);
call symput ('h'||ii, h);
call symput ('ERROR'||ii, ERROR);
call symput ('errorm'||ii, errorm);
call symput ('errory'||ii, errory);
call symput ('n', _n_);
drop i;
run;


%MACRO LAG(file, nrep, nobs, BMX,
BYM, BYX,  h, error, errorm,
errory);
DATA Summary; SET _null_;
%DO i=1 %TO &nrep;
proc iml;
id=j(&nobs,1);
x=j(&nobs,1);
call randgen(x, "bern", .5);

/*User can adjust arguments in line
below to change magnitude of lag*/
phim={1 -0.8};
thetam={1};
m=armasim(phim, thetam, 0, 1,
&nobs, 123&i)+&BMX*x;;

/*User can adjust arguments in line
below to change magnitude of lag*/
phiy={1 -0.8};
thetay={1};
y=armasim(phiy, thetay, 0, 1,
&nobs, 234&i)+&BYM*m + &BYX*x;

create data var {id x m y};
append;
close data;

/*Appends a replication number to
each observation*/
/*Create lagged variables for M and
Y*/
data data; set data;
rep=&i;
Mlag=lag1(M);
Ylag=lag1(Y);
NREP=&NREP;
NOBS=&NOBS;
BMX=&BMX;
BYM=&BYM;
BYX=&BYX;
```

```
h=&h;
ERROR=&ERROR;
errorm=&errorm;
errory=&errory;
FILE="&FILE";
run;


/*Saves all the replications to a
single 'summary' datafile*/
data new; set summary;
run;

data summary; set new data;
if mlag="." then delete;
RUN;

Data data.&file; set summary;
run;

%END;

run;
%MEND LAG;

%MACRO Condloop;

      %do k=1 %to &n;
%LAG(FILE=&&FILE&k,NREP=&&NREP&k,NO
BS=&&NOBS&k,BMX=&&BMX&k,BYM=&&BYM&k
,BYX=&&BYX&k,h=&&h&k,ERROR=&&ERROR&
k,errorm=&&errorm&k,errory=&&errory
&k);
      %end;
%mend condloop;
%condloop; run;
```

RP Simulation Program

The simulation program requires a CSV files, labeled here as "conds2.csv" which contains the number of observations, replications, and permutations for each generated dataset.

The following program analyzes multiple generated datasets using the RP test and modeling the lag in both M and Y.

```
/*This program conducts mediation
permutation tests*/
TITLE 'SIMULATION OF N=1
PERMUTATION';


PROC IMPORT OUT= WORK.conds
      DATAFILE="\conds2.csv"
       DBMS=CSV REPLACE;
     GETNAMES=YES;
     DATAROW=2;
       GUESSINGROWS=1001;
RUN;
OPTIONS PS=59 LS=80 REPLACE
NONOTES;
FILENAME NULLOG DUMMY 'C:\NULL';
proc printto log=null;

RUN;

/*This data step reads through the
csv file and creates a macro
variable for each simulation
parameter */
/*and appends a number to each
value corresponding to the
condition it is associated with.*/
data conds; length ii $8; set
conds;
i+1;
ii=left(put(i, 8.));
call symput ('dataname'||ii,
dataname);
call symput ('x'||ii, x);
call symput ('m'||ii, m);
call symput ('y'||ii, y);
call symput ('npermute'||ii,
npermute);
call symput ('maxiter'||ii,
maxiter);
call symput ('seed'||ii, seed);
call symput ('nobs'||ii, nobs);
call symput ('nsim'||ii, nsim);
/*call symput ('mlag'||ii, mlag);*/
/*call symput ('ylag'||ii, ylag);*/
call symput ('n', _n_);
drop i;
run;


/*PROC DATASETS LIB=WORK KILL
NOLIST;*/
/*run;*/

data summary2; set _null_;
%macro
permmed(dataname,x,m,y,npermute,max
iter,seed, nobs, nsim,mlag,ylag);

data summary; set _null_;
DATA SIMall; SET data.&dataname;
run;

%split(nobs=&&nobs&i);
run;


data new2; set summary2;
run;

data summary2; set new2 summary ;
run;

%mend permmed;
run;
quit;


%MACRO split (nobs);

      %do k=1 %to &&nsim&i;

Data sim; set
simall(where=(Rep=&k));

run;


* Make a listwise deleted dataset.
;
```

```
data listwise; set sim;
/*    if (&x ne .) and (&m ne .)
and (&y ne .);*/
/*    rename*/
/*          &x = x &m = m &y = y;*/
/*    if (&x ne .) and (&m ne .)
and (&y ne .) and (&mlag ne .) and
(&ylag ne .);*/
/*    rename*/
/*          &x = x &m = m &y = y
&mlag = mlag &ylag = ylag ;*/
run;

* Find number of cases in listwise
deleted dataset. ;
proc means data=listwise noprint;
      output out=meanout n(x) =
nobs;
run;
data _NULL_; set meanout;
      call symput('nobs',nobs);
run;


* Model 2: Regress y on x, m, mlag
ylag ;
proc reg data=listwise
outest=model2 tableout noprint;
model y = x m mlag ylag;      */;

      * Save predicted values and
residuals, which are needed for
permutation tests. ;
      output out=predres2 p=yhat
r=yres;
run;

* Model 3: Regress m on x mlag ;
proc reg data=listwise
outest=model3 tableout noprint;
      model m = x mlag;

      * Save predicted values and
residuals, which are needed for
permutation tests. ;
      output out=predres3 p=mhat
r=mres;
run;


* Gather results. ;
data parm2; set model2;
      if _TYPE_='PARMS';

      b = m; cprime = x; b02 =
intercept; byl = ylag; by2 = mlag;
keep b cprime b02 byl by2;
run;
data se2; set model2;
      if _TYPE_='STDERR';
      seb = m; keep seb;
run;

data parm3; set model3;
      if _TYPE_='PARMS';
      a = x; b03 = intercept; bml =
mlag; keep a b03 bml;
run;
data se3; set model3;
      if _TYPE_='STDERR';
      sea = x; keep sea;
run;
data origresult; merge parm2 se2
parm3 se3;
      vara = sea*sea;
      varb = seb*seb;
      sobelse = sqrt(a*a*varb +
b*b*vara);
run;

* Merge predicted values and
residuals for models 2 and 3. ;
data predres; merge predres2
predres3;
run;

*** Non-iterated permutation
confidence limits *** ;

* Make npermute copies of the
original data. ;
proc iml;
      use predres;
      read all var{x m mlag y ylag
yhat yres mhat mres} into orig;
      copies = orig;
      do i = 2 to &npermute;
            copies = copies//orig;
      end;
      varnames = {'x' 'm' 'mlag'
'y' 'ylag' 'yhat' 'yres' 'mhat'
'mres'};
      create origcopies from
copies[colname = varnames];
      append from copies;
quit;

* Make shuffling variables to
permute the residuals mres and
yres. ;
data origcopiesbycopy; set
origcopies;
      copynum = ceil(_N_/&nobs);
```

```
      shufflevaryres =
ranuni(&seed);
      shufflevarmres = ranuni(0);
run;

* Make a dataset with yres shuffled
within each dataset copy. ;
proc sort data=origcopiesbycopy
out=shuffleyres;
      by copynum shufflevaryres;
run;

* Make a dataset with mres shuffled
within each dataset copy. ;
proc sort data=origcopiesbycopy
out=shufflemres;
      by copynum shufflevarmres;
run;


* Merge shuffled residual mres and
yres with original data in each
dataset copy. ;
data shuffled; merge
origcopiesbycopy(keep=copynum x m
mlag y ylag mhat yhat)
shuffleyres(keep=yres)
shufflemres(keep=mres);
      %* Find mstar and ystar, the
new values of m and y based on
shuffling the residuals. ;
      mstar =     mhat + mres;
      ystar = yhat + yres;
run;


* Model 2: Regress ystar on x, m. ;
proc reg data=shuffled noprint
outest=model2shuf;
      by copynum;
      model ystar = x m mlag ylag;
run;


* Model 3: Regress mstar on x. ;
proc reg data=shuffled noprint
outest=model3shuf;
      by copynum;
      model mstar = x mlag;
run;

* Gather results. ;
data model2shuf; set model2shuf;
      if _TYPE_ = 'PARMS';
      bperm = m; keep bperm;
run;
data model3shuf; set model3shuf;
      if _TYPE_ = 'PARMS';
      aperm = x; keep aperm;
run;
data shufresult; merge model2shuf
model3shuf;
run;

* Include results for original
data. ;
data origshufresult; set
origresult(rename=(a=aperm
b=bperm)) shufresult;
      abperm = aperm*bperm;
run;

* Get confidence limits. ;
proc univariate data=origshufresult
noprint;
      var abperm;
      output out=permcl pctlpts =
2.5 97.5 pctlpre = perm pctlname =
lcl ucl;
run;


*** Iterated permutation confidence
limits *** ;

* Iterate twice: 1 = lower
confidence limit, 2 = upper
confidence limit ;
%do direction = 1 %to 2;

      data clguess; set origresult;
            %if &direction = 1
%then %do;
                  clab = a*b -
1.96*sobelse;
            %end;
            %else %do;
                  clab = a*b +
1.96*sobelse;
            %end;
      run;

      * Initialize loop counter and
break checker. ;
      %let loop = 0;
      %let break = 0;

      * Search for confidence
limit. ;
      %do %until ((&break = 1) or
(&loop = &maxiter));

            %* Increment loop
counter. ;
```

```
            %let loop =
%eval(&loop+1);

            * Analyze confidence
limit into its components. ;
            data clguess; set
clguess;
                  %* Make up terms
for quadratic equation solutions. ;
                  seaoverseb =
sea/seb;
                  term1 = (-1)*(a-
(b*seaoverseb));
                  term2 = (a-
(b*seaoverseb))**2;
                  term3 =
4*seaoverseb*clab;
                  term4 =
2*seaoverseb;
                  term5 =
(a+(b*seaoverseb));
                  term6 =
(a+(b*seaoverseb))**2;
                  * Find the two
possible solutions for clb. ;
                  %if &direction =
1 %then %do;

      clb1=(term5+sqrt(term6-
term3))/term4;

      clb2=(term5-sqrt(term6-
term3))/term4;
                  %end;
                  %else %do;

      clb1=(term1+sqrt(term2+term3)
)/term4;

      clb2=(term1-
sqrt(term2+term3))/term4;
                  %end;
                  %* Pick the
solution that puts clb closer to b.
;
                  clb1dist =
abs(clb1-b);
                  clb2dist =
abs(clb2-b);
                  if clb1dist <
clb2dist then clb = clb1;
                  else clb = clb2;
                  * Make cla ;
                  cla = clab/clb;
                  * Make clab into
a macro parameter for easier
reference ;
                  call
symput('clab',clab);
            run;

            * Merge confidence
limit components with copies of
original data created above. ;
            * This will allow for
new predicted values and residuals
to be made based on ;
            * using the confidence
limit components rather than the
sample values. ;
/*          data origcopiesbycopyi;
if _N_=1 then set clguess(keep=cla
clb cprime b02 b03 byl bml); set
origcopiesbycopy;*/
            data origcopiesbycopyi;
if _N_=1 then set clguess(keep=cla
clb cprime b02 b03 byl by2 bml);
set origcopiesbycopy;
                  %* Make new
residuals based on cla and clb and
sample values of other coefficients
;
                  %* (cprime, b02,
and b03). ;
                  yhati = b02 +
clb*m + cprime*x + by2*mlag +
byl*ylag;
                  yresi = y -
yhati;
                  mhati = b03 +
cla*x + bml*mlag;
                  mresi = m -
mhati;
            run;

            * On the first loop,
make variables for later shuffling
of yresi and mresi. ;
            %if &loop = 1 %then
%do;

                  data shufvars;
                        do i = 1 to
&nobs*&npermute;

      shufflevaryresi =
ranuni(&seed);

      shufflevarmresi = ranuni(0);

      output;
                        end;
                  run;
```

```
            %end;

            * On subsequent loops,
re-use same shuffling variables
made up on the first loop. ;

            * Make a dataset with
yresi shuffled within each dataset
copy. ;
            data shuffleyresi;
merge origcopiesbycopyi
shufvars(keep=shufflevaryresi);
            run;
            proc sort
data=shuffleyresi;
                  by copynum
shufflevaryresi;
            run;

            * Make a dataset with
mresi shuffled within each dataset
copy. ;
            data shufflemresi;
merge origcopiesbycopyi
shufvars(keep=shufflevarmresi);
            run;
            proc sort
data=shufflemresi;
                  by copynum
shufflevarmresi;
            run;

            * Merge shuffled
residuals yresi and mresi with
original data in each dataset copy.
;
            data shuffledi; merge
origcopiesbycopyi
shuffleyresi(keep=yresi)
shufflemresi(keep=mresi);
                  * Calculate
ystari and mstari, the new values
of y and m based on shuffling the
residuals. ;
                  ystari = yhat +
yresi;
                  mstari = mhat +
mresi;
            run;

            * Model 2: Regress
ystari on x, m. ;
            proc reg data=shuffledi
noprint outest=model2shufi;
                  by copynum;
                  model ystari = x
m mlag ylag;
            run;

* Model 3: Regress mstari on x. ;
            proc reg data=shuffledi
noprint outest=model3shufi;
                  by copynum;
                  model mstari = x
mlag;
            run;

            * Gather results. ;
            data model2shufi; set
model2shufi;
                  if _TYPE_ =
'PARMS';
                  bpermi = m; keep
bpermi;
            run;
            data model3shufi; set
model3shufi;
                  if _TYPE_ =
'PARMS';
                  apermi = x; keep
apermi;
            run;
            data shufresulti; merge
model2shufi model3shufi;
                  abpermi =
apermi*bpermi;
            run;

            * Include current clab
guess value. ;
            data origshufresulti;
set clguess(rename=(clab=abpermi))
shufresulti;
            run;

            * Make a frequency
table with cumulative frequencies
(percentile ranks) ;
            * of abpermi. ;
            proc freq
data=origshufresulti noprint;
                  tables abpermi
/out=freqtable outcum;
            run;

            * Merge current
confidence limit guess, clab, with
frequency table. ;
            * Find percentile rank
of current confidence limit guess,
clab, ;
            * in the distribution.
;
```

```
            * Do this by choosing
the nearest value of abpermi (they
should be equal ;
            * within rounding) and
getting its percentile rank. ;
            data freqtable; set
freqtable;
                  clab = &clab;
                  diff = abs(clab -
abpermi);
            run;
            proc sort
data=freqtable;
                  by diff;
            run;
            data getptilerank; set
freqtable;
                  if _N_ = 1;
                  prankclab =
CUM_PCT;
                  * Check if error
is small enough to exit the loop. ;
                  %if &direction =
1 %then %do;
                        abserror =
abs(prankclab - 2.5);
                  %end;
                  %else %do;
                        abserror =
abs(prankclab - 97.5);
                  %end;
                  if abserror le
0.5 then break = 1;
                  else break = 0;
                  * Save break as a
macro variable so the loop can
check its value. ;
                  call
symput('break',break);
            run;

            * If looping
continutes, choose as the next
guess for the value of the
confidence ;
            * limit the value of
abpermi at the target percentile
rank. ;
            proc univariate
data=origshufresulti noprint;
                  var abpermi;
                  output
out=univout pctlpts = 2.5 97.5
pctlpre = clab pctlname = lower
upper;
            run;

            * Merge the next guess
of clab back into the clguess
dataset to be ;
            * read at the top of
the loop. ;
            data clguess; merge
clguess(drop=clab) univout;
                  %if &direction =
1 %then %do;
                        clab =
clablower;
                  %end;
                  %else %do;
                        clab =
clabupper;
                  %end;
            run;

      %end;

      * Save the confidence limit
that was just found. ;
      %if &direction = 1 %then %do;
            data loweri; length
ilowerstatus $ 15; set
getptilerank;
                  if abserror le
0.5 then do;
                        ipermlcl =
clab;

      ilowerstatus = '(converged)';
                  end;
                  else do;
                        ipermlcl =
.;

      ilowerstatus = '(not
converged)';
                  end;
            run;
      %end;
      %else %do;

            data upperi; length
iupperstatus $ 15; set
getptilerank;
                  if abserror le
0.5 then do;
                        ipermucl =
clab;

      iupperstatus = '(converged)';
                  end;
                  else do;
                        ipermucl =
.;
```

```
      iupperstatus = '(not
converged)';
                  end;
            run;
      %end;

%end;

* Merge lower and upper iterated
confidence limits with non-iterated
confidence limits. ;
data allresult; merge permcl loweri
upperi;
      file print;
      put 'Permutation 95%
confidence limits'
            /'Lower: ' permlcl
            /'Upper: ' permucl
            /'Iterated permutation
95% confidence limits'
            /'Lower: ' ipermlcl ' '
ilowerstatus
            /'Upper: ' ipermucl ' '
iupperstatus;
data new; set summary;
run;

data summary; set new allresult;
run;
%end;
%mend split;

%MACRO Condloop;

      %do i=1 %to &n;

/*   ,mlag=&&mlag&i,ylag=&&ylag&i*/
%permmed(dataname=&&dataname&i,x=&&
x&i,m=&&m&i,y=&&y&i,npermute=&&nper
mute&i,maxiter=&&maxiter&i,seed=&&s
eed&i,NSIM=&&NSIM&i,NOBS=&&NOBS&i);
      %end;
      %mend Condloop;
%Condloop; run;

data data.summary2; set summary2;
run;
```

Program to Aggregate Simulation Results

The following program aggregates the results from 14 datasets, labeled here "D71-D84", and computes power and Type 1 error rates.

```
data SummaryPermSim; set summary2;
run;


data SummaryPermSim; set
SummaryPermSim;
                  if _n_ <=500        then
dataname="D71";
   else if 500<_n_<=1000     then
dataname="D72";
   else if 1000<_n_<=1500    then
dataname="D73";
         else if 1500<_n_<=2000    then
dataname="D74";
         else if 2000<_n_<=2500    then
dataname="D75";
         else if 2500<_n_<=3000    then
dataname="D76";
   else if 3000<_n_<=3500    then
dataname="D77";
         else if 3500<_n_<=4000    then
dataname="D78";
         else if 4000<_n_<=4500    then
dataname="D79";
         else if 4500<_n_<=5000    then
dataname="D80";
         else if 5000<_n_<=5500    then
dataname="D81";
         else if 5500<_n_<=6000    then
dataname="D82";
         else if 6000<_n_<=6500    then
dataname="D83";
         else if 6500<_n_<=7000    then
dataname="D84";
          run;


data clzero; set SummaryPermSim;
zeroincl=0;
lclneg=0;
uclpos=0;
if permlcl le 0 then lclneg=1;
if permucl ge 0 then uclpos=1;
if lclneg=1 and uclpos=1 then zeroincl=1;
if lclneg=1 and uclpos=0 then zeroincl=0;
/*significant*/;
if lclneg=0 and uclpos=1 then zeroincl=0;
/*significant*/;
if lclneg=0 and uclpos=0 then zeroincl=1;
keep permlcl permucl lclneg uclpos zeroincl
dataname;
run;


proc freq data=clzero;
by dataname;
tables zeroincl/ out=type1;
run;


data a;
input dataname $ truemed a b cp n;
cards;
D71 0 0      0.59   0
      200
D72 0 0.59   0      0
      200
D73 0 0      0.59   0.59   200
D74 0 0.59   0      0.59   200
D75 1 0.59   0.59   0      200
D76 1 0.59   0.59   0.59   200
D77 1 1      1      0
      200
D78 0 0      0.59   0
      500
D79 0 0.59   0      0
      500
D80 0 0      0.59   0.59   500
D81 0 0.59   0      0.59   500
D82 1 0.59   0.59   0      500
D83 1 0.59   0.59   0.59   500
```

```
D84 1  1              1              0
       500
;
run;

data b; merge type1 a;
by dataname;
run;

/*this data file has the type 1 error rates and
power for all 49 conditions*/
data type1_power; set b;
if zeroincl=0; /*significant*/;
if truemed=0 then type1rate = count/500;
else type1rate=".";
if truemed=1 then power = count/500; else
power=".";
run;

data data.clzero; set clzero; run;
data data.summarypermsim; set
summarypermsim;run;
data data.type1_power; set
type1_power;run;
```