ESA Supplementary Publication Service

Document No. ESPS 8736

Rerandomization procedures and statistical tests used for testing null hypotheses about prey depletion effects

Supplement to

"Prey depletion by odonate larvae: combining evidence from mutiple field experiments"

by

Johnson, Dan M., Clay L. Pierce, Thomas H. Martin, Charles N. Watson, Robert E. Bohanan, and Philip H. Crowley

ECOLOGY Vol. 68 (1987)

Johnson, Dan M., Clay L. Pierce, Thomas H. Martin, Charles N. Watson, Robert E. Bohanan, and Philip H. Crowley. 1987. Prey depletion by odonate larvae: combining evidence from multiple field experiments. Ecology 68

APPENDICES

The logic behind rerandomization tests used by Johnson et al. (1987) is quite simple (Bradley 1968, Edgington 1987, Sokal & Rohlf 1981:787), but may be unfamiliar to many readers. Therefore, in these appendices we explain the procedures used for testing "Prey Depletion Effects" (Figure 1) and estimating the power of those tests (Table 2, Figure 2) in some detail. All programs were written by the senior author in TurboPascal and run on an IBM-PC microcomputer. The calculations used for One-Way and Two-Way Analyses of Variance are described by Sokal & Rohlf (1981:Boxes 9.1 and 11.1). Listings of these programs are also presented.

LITERATURE CITED:

Bradley, J.V. 1968. Distribution-free Statistical Tests. Prentice-Hall, Inc., Englewood Cliffs, NJ

Edgington, E.S. 1987. Randomization Tests, Second Edition. Marcel Dekker, Inc., New York, NY

Sokal, R.R. and F.J. Rohlf. 1981. Biometry, Second Edition. W.H. Freeman Co., San Francisco, CA

APPENDIX A

The following procedures were used in a program called RERANDOM to evaluate the null hypothesis of no "Prey Depletion Effect", compared to a one-sided alternative that Control treatment (NO) means exceed Odonate treatment (HD & LD) means for each prey category within each experiment:

1) Enter un-transformed data (numbers per square meter) for each replicate of Control and Odonate treatments;
2) Perform a One-Way Analysis of Variance and calculate the F-ratio associated with the Prey Depletion Effect;
3) If treatment means do not differ in the way predicted by the one-sided alternate hypothesis (Control > Odonate), associate a negative value with this F-ratio;
4) Pool all data (from both Control and Odonate enclosures) to represent the null hypothesis that all observations were drawn from the same population;
5) For each of 1000 rerandomization runs,
    a) select "Control" and "Odonate" observations at random <u>without</u> replacement from the pooled data set (with the same number of replicates for each treatment as in the original experiment), and calculate the resulting F-ratio for the Prey Depletion Effect in a One-Way Analysis of Variance;
    b) if "treatment" means do not differ in the way predicted by the one-sided alternate hypothesis ("Control" > "Odonate"), associate a negative value with the F-ratio;

c) count the number of these simulated F-ratios exceeding that observed in the original analysis (i.e., the number of times in 1000 that the observed value of F is exceeded by chance when all observations are known to be from the same population);

d) keep a sorted list of the largest 50 simulated F-ratios to determine the "critical value" of F, the value exceeded 5% of the time due to chance, for use in a subsequent power analysis.

To estimate the power of each experiment to detect prey depletion, RERANDOM used the following procedures:

1) Add the apparent Prey Depletion Effect (Control mean - Odonate mean) to each Odonate observation before pooling all data to represent the null hypothesis;
2) Specify the depletion effect to be detected as a proportion (i.e., 50% depletion, corresponding to an Odonate Treatment mean that is 50% of the Control mean);
3) For each of 1000 rerandomization runs,
   a) select "Control" and "Odonate" observations at random without replacement from the pooled data set (with the same number of replicates for each treatment as in the original experiment);
   b) calculate the mean of these "Control" observations;
   c) subtract from each "Odonate" observation a hypothetical depletion effect equal to the specified proportion of the calculated mean for "Control" observations (with the

condition that the resulting "Odonate" observation may not be less than zero);

d) calculate the resulting F-ratio for the Prey Depletion Effect in a One-Way Analysis of Variance;

e) if "treatment" means do not differ in the direction predicted by the one-sided alternate hypothesis ("Control" > "Odonate"), associate a negative value with this F-ratio;

f) count the number of these simulated F-ratios that exceed the "critical value" of F from the first rerandomization analysis (i.e., the number of times in 1000 that the null hypothesis would be rejected when it is known to be false).

```
program          RERANDOM;
                              {randomization test for analysis of variance}
                              {and power analysis for "prey depletion"}
const                  a =  2;
                      nc =  3;
                      no = 15;
                      nt = 18;
                  Rerand = 1000;
                   alpha = 0.05;
              Depletion = 0.5;

type                  Data = record
                               Number:Real;
                             end;

var                   YControl, YRerandControl                    :Array[1..nc] of Real;
                      YOdonate, YRerandOdonate                    :Array[1..no] of Real;
                      YPooled                                   :Array[1..nt] of Real;
                      GrandTotal, SumSqObs, SumSqGroup, CorrectionTerm     :Real;
                      SumControl, SumOdonate                               :Real;
                      SSTotal, SSGroups, SSWithin, MSAmong, MSWithin       :Real;
                      N, dfAmong, dfWithin, dfTotal, Count_alpha           :Integer;
                      F, F_actual, F_critical                              :Real;
                      FRerand, FRerandSorted                 :Array[1..200] of Real;
                      Replicate, aB, bt, i, j, jj                          :Integer;
                      Heading                                :String[25];
                      DataRecord  :Data;
                      TheFile     :file of Data;

                      MeanYControl, MeanYOdonate                           :Real;
                      Count_beta                                           :Integer;
                      Again                                                :Char;

                      Prospect, Counter, PR     :Integer;
                      Previous  :Array[1..nc] of Integer;


const    d = 0.1;


label  10, 20, 30, 40, 50;


begin
  FillChar(YControl,SizeOf(YControl),0);
  FillChar(YOdonate,SizeOf(YOdonate),0);
  FillChar(FRerandSorted,SizeOf(FRerandSorted),0);
  FillChar(Previous,SizeOf(Previous),0);

  {read in data for one experiment}
  Writeln('RERANDOM randomizes a One-Way Analysis of Variance & Power Analysis')
;
  Writeln('for the "Odonate Effect" in a Prey Depletion Experiment.');
  Writeln;
  Write('Enter File Name for this run:  ');
  Readln(Heading);
  Assign(TheFile, Heading);
  Reset(TheFile);
```

```
for Replicate := 1 to nc do
  begin
    Read(TheFile, DataRecord);
    YControl[Replicate] := DataRecord.Number;
  end;
for Replicate := 1 to no do
  begin
    Read(TheFile,DataRecord);
    YOdonate[Replicate] := DataRecord.Number;
  end;
Close(TheFile);


{calculate original analysis of variance}
dfAmong      := a - 1;
dfWithin     := nt - a;
dfTotal      := nt - 1;

GrandTotal   := 0;
SumSqObs     := 0;
SumControl   := 0;
SumOdonate   := 0;
SumSqGroup   := 0;

for i := 1 to nc do
  begin
    GrandTotal := GrandTotal + YControl[i];
    SumSqObs   := SumSqObs   + sqr(YControl[i]);
    SumControl := SumControl + YControl[i];
  end;
for i := 1 to no do
  begin
    GrandTotal := GrandTotal + YOdonate[i];
    SumSqObs   := SumSqObs   + sqr(YOdonate[i]);
    SumOdonate := SumOdonate + YOdonate[i];
  end;

MeanYControl := SumControl/nc;
MeanYOdonate := SumOdonate/no;


SumSqGroup      := sqr(SumControl)/nc + sqr(SumOdonate)/no;
CorrectionTerm := sqr(GrandTotal)/nt;

SSTotal         := SumSqObs     - CorrectionTerm;
SSGroups        := SumSqGroup   - CorrectionTerm;
SSWithin        := SSTotal      - SSGroups;

MSAmong         := SSGroups/dfAmong;
MSWithin        := SSWithin/dfWithin;

F_actual        := MSAmong/MSWithin; {a sign may be attached below}

if MeanYControl < MeanYOdonate then
  begin
    F_actual := -1.0*F_actual;
    Writeln;
    Writeln('A negative sign is attached to the F value because means');
    Writeln('did not differ in the predicted direction.');
```

```
      Writeln;
    end;
  {testing one-sided alternative hypothesis}


  ClrScr;
  Writeln('                          ',Heading);
  Writeln('Control data: ');
  for Replicate := 1 to nc do
    Write(YControl[Replicate]:5:0);
  Writeln;
  Writeln('Odonate data: ');
  for Replicate := 1 to no do
    Write(YOdonate[Replicate]:5:0);
  Writeln;


  Writeln;
  Writeln('ORIGINAL ANALYSIS OF VARIANCE for ',Heading);
  Writeln('SOURCE       df              SS                MS          F');
  Writeln('Treatment    ',dfAmong :2,'     ',SSGroups:12:2,'     ',MSAmong:12:2,'
    ',F_actual:4:2);
  Writeln('Within       ',dfWithin:2,'     ',SSWithin:12:2,'     ',MSWithin:12:2);
  Writeln('Total        ',dfTotal :2,'     ',SSTotal:12:2);


  {pool data to represent null hypothesis for the randomization test}
  for Replicate := 1 to nc do
    YPooled[Replicate] := YControl[Replicate];
  for Replicate := 1 to no do
    YPooled[nc + Replicate] := YOdonate[Replicate];

  Randomize;
  Count_alpha := 0;
  aB := Round(alpha*Rerand);{number equivalent to alpha}

  {randomization loop to estimate alpha}
  for bt := 1 to Rerand do
    begin
      for Replicate := 1 to nc do
        begin {sampling without replacement}
20:       Prospect := Random(nt+1);
          for PR := Replicate downto 1 do
            if (Prospect <> Previous[PR]) and (Prospect <> 0)
              then
                begin
                  YRerandControl[Replicate] := YPooled[Prospect];
                  Previous[Replicate] := Prospect;
                end
              else GoTo 20;
        end;
      Counter := 0;
      for Replicate := 1 to nt do
        begin {avoiding data assigned to controls}
          for PR := 1 to nc do
            if Replicate = Previous[PR] then goto 40;
          Counter := Counter + 1;
          YRerandOdonate[Counter] := YPooled[Replicate];
40:     end;
```

```
      GrandTotal := 0;
      SumSqObs   := 0;
      SumControl := 0;
      SumOdonate := 0;
      SumSqGroup := 0;

      for i := 1 to nc do
        begin
          GrandTotal := GrandTotal + YRerandControl[i];
          SumSqObs   := SumSqObs   + sqr(YRerandControl[i]);
          SumControl := SumControl + YRerandControl[i];
        end;
      for i := 1 to no do
        begin
          GrandTotal := GrandTotal + YRerandOdonate[i];
          SumSqObs   := SumSqObs   + sqr(YRerandOdonate[i]);
          SumOdonate := SumOdonate + YRerandOdonate[i];
        end;
      SumSqGroup      := sqr(SumControl)/nc + sqr(SumOdonate)/no;
      CorrectionTerm := sqr(GrandTotal)/nt;

      SSTotal         := SumSqObs   - CorrectionTerm;
      SSGroups        := SumSqGroup - CorrectionTerm;
      SSWithin        := SSTotal    - SSGroups;

      MSAmong         := SSGroups/dfAmong;
      MSWithin        := SSWithin/dfWithin;

      F               := MSAmong/MSWithin;

      if SumControl/nc < SumOdonate/no then F := -1.0*F;
      {counting only those values of F that exceed F_actual when means differ
       in expected directions}
      if F >= F_actual then
        Count_alpha := Count_alpha + 1;

      for j :=  1 to aB do  {find critical value exceeded by alpha*Rerand runs}
        if F > FRerandSorted[j] then
              begin
                for jj := aB downto j+1 do
                  FRerandSorted[jj] := FRerandSorted[jj-1];
                FRerandSorted[j] := F;
                Goto 10;
              end;
10: end; {end of randomization loop for alpha}

  Writeln;
  Writeln(Count_alpha:4,' of ',Rerand:4,' randomization runs exceeded the origin
al F = ',F_actual:6:2,' ;');
  Writeln('thus the probability of Type I error is ',Count_alpha/Rerand:5:3);
  Writeln;
  Writeln('The critical value, exceeded by ',aB:4,' runs, is F_[',alpha:3:2,'] =
 ',FRerandSorted[aB]:4:2,' .');

  F_critical := FRerandSorted[aB];
```

```
{randomization power analysis for prey depletion anova}

  {pool data to represent null hypothesis for power rerandomization}
  {removing apparent "Odonate Effect" by adding mean difference to YOdonate}
  for Replicate := 1 to nc do
    YPooled[Replicate] := YControl[Replicate];
  for Replicate := 1 to no do
        YPooled[nc+Replicate] := YOdonate[Replicate] + (MeanYControl - MeanYOdon
ate);

  Writeln;
  Writeln('An estimate of the power to detect 50% prey depletion, ');
  Write('based on ',Rerand:4,' randomization runs is:');

      Randomize;
      Count_beta := 0;

      {second randomization loop for power with specified %depletion}
      for bt := 1 to Rerand do
        begin
          SumControl := 0;
          for Replicate := 1 to nc do
            begin {sampling without replacement}
30:           Prospect := Random(nt+1);
              for PR := Replicate downto 1 do
                if (Prospect <> Previous[PR]) and (Prospect <> 0)
                  then
                    begin
                      YRerandControl[Replicate] := YPooled[Prospect];
                      Previous[Replicate] := Prospect;
                      SumControl := SumControl + YRerandControl[Replicate];
                    end
                  else GoTo 30;
            end;

          Counter := 0;
          for Replicate := 1 to nt do
            begin {avoiding data assigned to controls}
              for PR := 1 to nc do
                if Replicate = Previous[PR] then goto 50;
              Counter := Counter + 1;
              YRerandOdonate[Counter] := YPooled[Replicate]
                                              - (Depletion)*SumControl/nc;
              if YRerandOdonate[Counter] < 0
                then YRerandOdonate[Counter] := 0; {can't have negative prey}
50:           end;

          GrandTotal := 0;
          SumSqObs   := 0;
          SumControl := 0;
          SumOdonate := 0;
          SumSqGroup := 0;

          for i := 1 to nc do
            begin
              GrandTotal := GrandTotal + YRerandControl[i];
              SumSqObs   := SumSqObs   + sqr(YRerandControl[i]);
              SumControl := SumControl + YRerandControl[i];
            end;
```

```
        for i := 1 to no do
          begin
            GrandTotal  := GrandTotal  + YRerandOdonate[i];
            SumSqObs    := SumSqObs    + sqr(YRerandOdonate[i]);
            SumOdonate  := SumOdonate  + YRerandOdonate[i];
          end;
        SumSqGroup      := sqr(SumControl)/nc + sqr(SumOdonate)/no;
        CorrectionTerm  := sqr(GrandTotal)/nt;

        SSTotal         := SumSqObs    - CorrectionTerm;
        SSGroups        := SumSqGroup  - CorrectionTerm;
        SSWithin        := SSTotal     - SSGroups;

        MSAmong         := SSGroups/dfAmong;
        MSWithin        := SSWithin/dfWithin;

        F               := MSAmong/MSWithin;
        if SumControl/nc < SumOdonate/no then F := -1.0*F;
        {counting those that differ in expected direction only}

        if F >= F_critical then
          Count_beta := Count_beta + 1;

      end; {end of second randomization loop}

    Writeln('    ',Count_beta/Rerand:5:3);
    Sound(440);
    Delay(500);
    NoSound;
end.
```

APPENDIX B

The following procedures were used in a program called RERANTWO to evaluate the probabilities associated with null hypotheses concerning "Prey Depletion Effects" in a Two-Way Analysis of Variance combining data from Experiments A, B and C:

1) Enter data for each experiment (numbers per square meter) for each replicate of Control and Odonate treatments;
2) Perform a Two-Way Analysis of Variance and calculate the F-ratio (Model I) associated with the Prey Depletion Effect;
3) If treatment means do not differ in the way predicted by the one-sided alternate hypothesis (Control > Odonate), associate a negative value with this F-ratio;
3) Pool all data (from both Control and Odonate enclosures) within each of the three experiments to represent the null hypothesis that observations within each experiment were sampled from the same population;
4) For each of 1000 rerandomization runs,
    a) select "Control" and "Odonate" observations for each of three experiments at random <u>without</u> replacement from the pooled data set for that experiment (with the same number of replicates for each treatment as in the original experiment), and calculate the resulting F-ratios for the Prey Depletion Effect in a Two-Way Analysis of Variance;

b) if "treatment" means do not differ in the way predicted by the one-sided alternate hypothesis ("Control" > "Odonate"), associate a negative value with the F-ratio;

c) count the number of these simulated F-ratios that exceed those observed in the original analysis (i.e., the number of times in 1000 that the observed value of F is exceeded by chance when all observations within each experiment are known to be from the same population);

d) keep a sorted list of the largest 50 simulated F-ratios to determine the "critical value" of F, the value exceeded 5% of the time due to chance, for use in a subsequent power analysis.

To estimate the power of the Two-Way Analysis of Variance to reject false null hypotheses for the "Prey Depletion Effect", RERANTWO used the following procedures:

1) Add the apparent Prey Depletion Effect (Control mean - Odonate mean) within each experiment to each observation from an Odonate enclosure before partly pooling the data (within experiments); this leaves the apparent Experiment Effect in place, but removes the apparent Prey Depletion Effect to represent the null hypothesis before specific hypothetical prey depletion effects are imposed on the data;

2) Specify the hypothetical prey depletion effect to be imposed (i.e., 10% depletion indicates that the Odonate treatment mean should be 90% of the Control mean);

3) For each of 1000 rerandomization runs,

a) select "Control" and "Odonate" observations at random without replacement from the pooled data set for each experiment (with the same number of replicates for each treatment as in the original experiment);

b) calculate the mean of "Control" observations within each "experiment";

c) subtract from each "Odonate" observation a hypothetical depletion effect equal to the specified proportion of the mean for "Control" observations within each "experiment" (with the condition that the resulting "Odonate" observation may not be less than zero);

d) calculate the resulting F-ratio (Model I) for the Prey Depletion Effect in a Two-Way Analysis of Variance;

e) if overall "treatment" means do not differ in the direction predicted by the one-sided alternate hypothesis ("Control" > "Odonate"), associate a negative value with this F-ratio;

f) count the number of these simulated F-ratios that exceed the "critical value" of F for the Prey Depletion Effect from the first rerandomization analysis (i.e., the number of times in 1000 that the null hypothesis would be rejected when it is known to be false).

```
program          RERANTWO; {two-way anova and power for prey depletion}
                           {with rerandomization tests for alpha}
const              a =     2   ;{treatments, Control & Odonate}
                   b =     3   ;{experiments, A, B, C}
                  nc =     3   ;{control replicates per experiment}
                  no =    15   ;{odonate replicates per experiment}
                  nt =    18   ;{total replicates per experiment}
                 net =    54   ;{total in all experiments}
               alpha =     0.05;{probability of Type I error}
              Rerand = 1000    ;{number of bootstrap runs}
                   d =     0.1 ;{increment in prey depletion}

type               Data = record
                            Number:Real;
                          end;

var     YControl, YRerandControl              :Array[1..b,1..nc] of Real;
        YOdonate, YRerandOdonate              :Array[1..b,1..no] of Real;
        YPartlyPooled                        :Array[1..b,1..nt] of Real;
        GrandTotal, CorrectionTerm, Control, Odonate      :Real;
        ExControl, ExOdonate,SumControl, SumOdonate, SumTotal:Array[1..b] of Real
;
        SumSqObs, SumSqTrtmnt, SumSqExpt, SumSqCells            :Real;
        SSTotal, SSTrtmnt, SSExpt, SSInter, SSCells, SSWithin   :Real;
        MSTrtmnt, MSExpt, MSInter, MSWithin                     :Real;
        dfTrtmnt, dfExpt, dfInter, dfWithin, dfTotal          :Integer;
        N, aB, Count_T_alpha, Count_E_alpha, Count_I_alpha:Integer;
        F_Trtmnt, F_Expt, F_Inter, F, F_T, F_E, F_I, F_critical:Real;
        F_TRerandSorted, F_ERerandSorted, F_IRerandSorted:Array[1..100] of Real;
        Replicate,bt,i,j,jj,x,PR, Prospect, Counter           :Integer;
        Heading, Name                        :String[25];
        l, Proceed                           :Char;
        DataRecord                           :Data;
        TheFile                              :file of Data;
        MeanYControl, MeanYOdonate           :Real;
        Count_T_beta                         :Integer;
        Depletion                            :Real;
        Previous              :array[1..nc] of Integer;

label   1, 2, 3, 4, 5, 10, 20, 30, 40, 50;

begin
  FillChar(YControl,SizeOf(YControl),0);
  FillChar(YOdonate,SizeOf(YOdonate),0);
  FillChar(YRerandControl,SizeOf(YRerandControl),0);
  FillChar(YRerandOdonate,SizeOf(YRerandOdonate),0);
  FillChar(YPartlyPooled,SizeOf(YPartlyPooled),0);
  FillChar(SumControl,SizeOf(SumControl),0);
  FillChar(SumOdonate,SizeOf(SumOdonate),0);
  FillChar(SumTotal,SizeOf(SumTotal),0);
  FillChar(F_ERerandSorted,SizeOf(F_ERerandSorted),0);
  FillChar(F_TRerandSorted,SizeOf(F_TRerandSorted),0);
  FillChar(F_IRerandSorted,SizeOf(F_IRerandSorted),0);

  {read in data for one experiment}
  Writeln('RERANTWO does a Two-Way Analysis of Variance and Power');
  Writeln('             for "Odonate Effect" in Prey Depletion Experiments,');
  Writeln('                 using rerandomization tests to determine alpha.');
  Writeln;
```

```
  Write('Enter Heading for this run:  ');
  Readln(Heading);
  Writeln('What data files have experimental results in them?');
  for x:= 1 to b do
    begin
      Readln(Name);
      Assign(TheFile, Name);
      Reset(TheFile);

      for Replicate := 1 to nc do
        begin
          Read(TheFile, DataRecord);
          YControl[x,Replicate] := DataRecord.Number;
        end;
      for Replicate := 1 to no do
        begin
          Read(TheFile, DataRecord);
          YOdonate[x,Replicate] := DataRecord.Number;
        end;
    end;

  {calculate original analysis of variance}
  N := (nc + no)*b;
  dfTrtmnt     := a - 1;
  dfExpt       := b - 1;
  dfInter      := (a-1)*(b-1);
  dfWithin     := N - a*b;
  dfTotal      := N - 1;

  GrandTotal   := 0;
  SumSqObs     := 0;
  SumSqTrtmnt  := 0;
  SumSqExpt    := 0;
  SumSqCells   := 0;

  {looping over independent experiments}
  for x:= 1 to b do
    begin
      for i := 1 to nc do
        begin
          GrandTotal  := GrandTotal + YControl[x,i];
          SumSqObs    := SumSqObs   +sqr(YControl[x,i]);
          SumControl[x] := SumControl[x] + YControl[x,i];
        end;
      for i := 1 to no do
        begin
          GrandTotal  := GrandTotal + YOdonate[x,i];
          SumSqObs    := SumSqObs   + sqr(YOdonate[x,i]);
          SumOdonate[x] := SumOdonate[x] + YOdonate[x,i];
        end;
      ExControl[x] := SumControl[x]/nc; {means for original data}
      ExOdonate[x] := SumOdonate[x]/no; {for each experiment}

      SumSqCells      := SumSqCells  + sqr(SumControl[x])/nc + sqr(SumOdonate[x])
/no;
      SumSqExpt       := SumSqExpt   + sqr(SumControl[x] + SumOdonate[x])/nt;
    end;

  Control := 0;
```

```
Odonate := 0;
for x := 1 to b do
  begin
    Control := Control + SumControl[x];
    Odonate := Odonate + SumOdonate[x];
  end;
MeanYControl := Control/(b*nc);{overall means from original data}
MeanYOdonate := Odonate/(b*no);
SumSqTrtmnt := sqr(Control)/(b*nc) + sqr(Odonate)/(b*no);
CorrectionTerm := sqr(GrandTotal)/N;

SSTotal          := SumSqObs     - CorrectionTerm;
SSCells          := SumSqCells   - CorrectionTerm;
SSTrtmnt         := SumSqTrtmnt - CorrectionTerm;
SSExpt           := SumSqExpt    - CorrectionTerm;
SSInter          := SSCells - SSTrtmnt - SSExpt;
SSWithin         := SSTotal - SSCells;

MSTrtmnt         := SSTrtmnt/dfTrtmnt;
MSExpt           := SSExpt/dfExpt;
MSInter          := SSInter/dfInter;
MSWithin         := SSWithin/dfWithin;

F_Trtmnt         := MSTrtmnt/MSWithin; {a sign may be attached below}
F_Expt           := MSExpt/MSWithin;
F_Inter          := MSInter/MSWithin;

 if MeanYControl < MeanYOdonate
   then   {assign negative F so only those in predicted direction count}
     begin
       F_Trtmnt := -1.0*F_Trtmnt;
       Writeln(Lst);
       Writeln(Lst,'A negative sign is attached to F_Trtmnt because');
       Writeln(Lst,'means did not differ in the predicted direction.');
       Writeln(Lst);
     end;


ClrScr;
Writeln(Lst);
Writeln(Lst,'                          ',Heading);
Writeln(Lst,'Control data: ');
for x := 1 to b do
  begin
    for Replicate := 1 to nc do
      Write(Lst,YControl[x,Replicate]:4:0,',');
    Writeln(Lst);
  end;
Writeln(Lst,'Odonate data: ');
for x:= 1 to b do
  begin
    for Replicate := 1 to no do
        Write(Lst,YOdonate[x,Replicate]:4:0,',');
    Writeln(Lst);
  end;



Writeln(Lst);
```

```
  Writeln(Lst,'ORIGINAL ANALYSIS OF VARIANCE for ',Heading);
  Writeln(Lst,'SOURCE       df                SS                MS        F');
  Writeln(Lst,'Treatments   ',dfTrtmnt:2,'     ',SSTrtmnt:12:2,'     ',MSTrtmnt:12:
2,'     ',F_Trtmnt:4:2);
  Writeln(Lst,'Experiments  ',dfExpt:2,'     ',SSExpt:12:2,'     ',MSExpt:12:2,'
 ', F_Expt:4:2);
  Writeln(Lst,'Interaction  ',dfInter:2,'     ',SSInter:12:2,'     ',MSInter:12:2,'
   ', F_Inter:4:2);
  Writeln(Lst,'Within       ',dfWithin:2,'     ',SSWithin:12:2,'     ',MSWithin:12:
2);
  Writeln(Lst,'Total        ',dfTotal:2,'     ',SSTotal:12:2);
  Writeln(Lst);

  {pool data within experiments for null hypothesis for rerandomization tests}
  for x := 1 to b do  {looping over experiments}
    begin
      for Replicate := 1 to nc do
        YPartlyPooled[x,Replicate] := YControl[x,Replicate];
      for Replicate := 1 to nc do
        YPartlyPooled[x,nc + Replicate] := YOdonate[x,Replicate];
    end;

  Randomize;
  aB := Round(alpha*Rerand);
  Count_T_alpha := 0;
  Count_E_alpha := 0;
  Count_I_alpha := 0;

  {rerandomization loop to estimate alpha}
  for bt := 1 to Rerand do
    begin
      for x := 1 to b do
        begin
          FillChar(Previous,SizeOf(Previous),0);
          for Replicate := 1 to nc do
            begin  {sampling without replacement}
2:            Prospect := Random(nt+1);
              for PR := Replicate downto 1 do
                if (Prospect <> Previous[PR]) and (Prospect <> 0)
                  then
                    begin
                      YRerandControl[x,Replicate] := YPartlyPooled[x,Prospect];
                      Previous[Replicate] := Prospect;
                    end
                  else GoTo 2;
            end;
          Counter := 0;
          for Replicate := 1 to nt do
            begin {avoiding data assigned to controls}
              for PR := 1 to nc do
                if Replicate = Previous[PR] then Goto 4;
              Counter := Counter + 1;
              YRerandOdonate[x,Counter] := YPartlyPooled[x,Replicate];
4:          end;
        end;

      GrandTotal := 0;
      SumSqObs   := 0;
      SumSqTrtmnt:= 0;
```

```
FillChar(SumControl,SizeOf(SumControl),0);
FillChar(SumOdonate,SizeOf(SumOdonate),0);
SumSqExpt  := 0;
SumSqCells := 0;

{looping over simulated experiments}
for x := 1 to b do
  begin
    for i := 1 to nc do
      begin
        GrandTotal  := GrandTotal + YRerandControl[x,i];
        SumSqObs    := SumSqObs   + sqr(YRerandControl[x,i]);
        SumControl[x] := SumControl[x] + YRerandControl[x,i];
      end;
    for i := 1 to no do
      begin
        GrandTotal  := GrandTotal + YRerandOdonate[x,i];
        SumSqObs    := SumSqObs   + sqr(YRerandOdonate[x,i]);
        SumOdonate[x] := SumOdonate[x] + YRerandOdonate[x,i];
      end;
    SumSqCells:= SumSqCells + sqr(SumControl[x])/nc + sqr(SumOdonate[x])/n
o;
    SumSqExpt := SumSqExpt  + sqr(SumControl[x] + SumOdonate[x])/nt;
  end;

Control := 0;
Odonate := 0;
for x := 1 to b do
  begin
    Control := Control + SumControl[x];
    Odonate := Odonate + SumOdonate[x];
  end;
SumSqTrtmnt := sqr(Control)/(b*nc) + sqr(Odonate)/(b*no);
CorrectionTerm := sqr(GrandTotal)/N;

SSTotal          := SumSqObs      - CorrectionTerm;
SSCells          := SumSqCells    - CorrectionTerm;
SSTrtmnt         := SumSqTrtmnt   - CorrectionTerm;
SSExpt           := SumSqExpt     - CorrectionTerm;
SSInter          := SSCells - SSTrtmnt - SSExpt;
SSWithin         := SSTotal     - SSCells;

MSTrtmnt         := SSTrtmnt/dfTrtmnt;
MSExpt           := SSExpt/dfExpt;
MSInter          := SSInter/dfInter;
MSWithin         := SSWithin/dfWithin;

F_T              := MSTrtmnt/MSWithin;
F_E              := MSExpt/MSWithin;
F_I              := MSInter/MSWithin;

if (Control/(b*nc)) < (Odonate/(b*no)) then F_T := -1.0*F_T;

if F_T >= F_Trtmnt then
  Count_T_alpha := Count_T_alpha + 1;
if F_E >= F_Expt then
  Count_E_alpha := Count_E_alpha + 1;
if F_I >= F_Inter then
  Count_I_alpha := Count_I_alpha + 1;
```

```
                                                    Depletion := 0;
                                                end;
                                            end;
                                        end;
                                    end;
                                end;
                            end;
                        end;
                    end;
                end;
            end;
        end;
    end;
    for j := 1 to aB do {find critical value exceeded by alpha*Rerand runs}
        if F_T > F_TRerandSorted[j] then
            begin
            for jj := aB downto j+1 do
                F_TRerandSorted[jj] := F_TRerandSorted[jj-1];
            F_TRerandSorted[j] := F_T;
            Goto 10;
            end;
    for j := 1 to aB do
10:     if F_E > F_ERerandSorted[j] then
            begin
            for jj := aB downto j+1 do
                F_ERerandSorted[jj] := F_ERerandSorted[jj-1];
            F_ERerandSorted[j] := F_E;
            Goto 20;
            end;
    for j := 1 to aB do
20:     if F_I > F_IRerandSorted[j] then
            begin
            for jj := aB downto j+1 do
                F_IRerandSorted[jj] := F_IRerandSorted[jj-1];
            F_IRerandSorted[j] := F_I;
            Goto 30;
            end;
30: end; {end of first rerandomization loop for alpha}

  Writeln(lst);
  Writeln(lst,'SOURCE                F Model I          alpha       F_I','alpha:3:2,' ')
;
  Writeln(lst,'Treatments       ','F_Trtmnt:4:2,'
and "5:3,'      ','F_TRerandSorted[aB]:4:2);
  Writeln(lst,'Experiments      ','F_Expt:4:2   ,'
and  "5:3,'      ','F_ERerandSorted[aB]:4:2);
  Writeln(lst,'Interaction      ','F_Inter:4:2  ,'
and  "5:3,'      ','F_IRerandSorted[aB]:4:2);
  Writeln(lst);

{partly pool data to represent null hypothesis for the power rerandomization}
{removing apparent "Odonate Effect" within each experiment}
{but leaving the apparent "Experiment Effect" to affect interaction term.}
for x := 1 to b do
  begin
  for Replicate := 1 to nc do
    YPartlyPooled[x,Replicate] := YControl[x,Replicate];
  for Replicate := 1 to no do
    begin
    for x := 1 to b do
      YPartlyPooled[x,nc + Replicate] := YOdonate[x,Replicate] +
                  (ExControl[x] - ExOdonate[x]);
    end; {avoid negative prey}
  end;

  if YPartlyPooled[x,nc+Replicate] > 0
    then YPartlyPooled[x,nc+Replicate] := 0;

  Writeln(lst,'Estimates of power to detect specific prey depletion effects.');
  Writeln(lst,'each based on ',Rerand:4,' rerandomization runs, for Model I are');
  Writeln;
```

```
  while Depletion <= 0.8 do
    begin
      Depletion := Depletion + d;

      Count_T_beta := 0;

      {second bootstrap loop for power with specific % depletion}
      for bt := 1 to Rerand do
        begin
          FillChar(SumControl,SizeOf(SumControl),0);
          for x:= 1 to b do
            begin
              FillChar(Previous,SizeOf(Previous),0);
              for Replicate := 1 to nc do
                begin  {sampling without replacement}
3:                Prospect := Random(nt+1);
                  for PR := Replicate downto 1 do
                    if (Prospect <> Previous[PR]) and (Prospect <> 0)
                      then
                        begin
                          YRerandControl[x,Replicate] := YPartlyPooled[x,Prospec
t];
                          Previous[Replicate] := Prospect;
                          SumControl[x] := SumControl[x] + YRerandControl[x,Repl
icate];
                        end
                      else GoTo 3;
                end;
              Counter := 0;
              for Replicate := 1 to nt do
                begin {avoiding data assigned to controls}
                  for PR := 1 to nc do
                    if Replicate = Previous[PR] then Goto 5;
                  Counter := Counter + 1;
                  YRerandOdonate[x,Counter] := YPartlyPooled[x,Replicate] -
                                               (Depletion*SumControl[x]/nc);
                  if YRerandOdonate[x,Counter] < 0 then YRerandOdonate[x,Counter
] := 0;
5:                end;
            end;
          GrandTotal  := 0;
          SumSqObs    := 0;
          SumSqTrtmnt:= 0;
          FillChar(SumControl,SizeOf(SumControl),0);
          FillChar(SumOdonate,SizeOf(SumOdonate),0);
          SumSqExpt   := 0;
          SumSqCells  := 0;

          {looping over simulated experiments}
          for x := 1 to b do
            begin
              for i := 1 to nc do
                begin
                  GrandTotal := GrandTotal + YRerandControl[x,i];
                  SumSqObs   := SumSqObs   + YRerandControl[x,i]*YRerandControl[
x,i];
                  SumControl[x] := SumControl[x] + YRerandControl[x,i];
                end;
              for i := 1 to nc do
```

```
                begin
                  GrandTotal  := GrandTotal  + YRerandOdonate[x,i];
                  SumSqObs    := SumSqObs    + YRerandOdonate[x,i]*YRerandOdonate[
x,i];
                  SumOdonate[x] := SumOdonate[x] + YRerandOdonate[x,i];
                end;
              SumSqCells      := SumSqCells + sqr(SumControl[x])/nc + sqr(SumOdon
ate[x])/no;
              SumSqExpt       := SumSqExpt  + sqr(SumControl[x] + SumOdonate[x])/
nt;
            end;

          Control := 0;
          Odonate := 0;
          for x := 1 to b do
            begin
              Control := Control + SumControl[x];
              Odonate := Odonate + SumOdonate[x];
            end;
          SumSqTrtmnt := sqr(Control)/(b*nc) + sqr(Odonate)/(b*no);
          CorrectionTerm := sqr(GrandTotal)/N;

          SSTotal          := SumSqObs      - CorrectionTerm;
          SSCells          := SumSqCells    - CorrectionTerm;
          SSTrtmnt         := SumSqTrtmnt   - CorrectionTerm;
          SSExpt           := SumSqExpt     - CorrectionTerm;
          SSInter          := SSCells - SSTrtmnt - SSExpt;
          SSWithin         := SSTotal     - SSCells;

          MSTrtmnt         := SSTrtmnt/dfTrtmnt;
          MSExpt           := SSExpt/dfExpt;
          MSInter          := SSInter/dfInter;
          MSWithin         := SSWithin/dfWithin;

          F_T              := MSTrtmnt/MSWithin;
          if (Control/(b*nc)) < (Odonate/(b*no)) then F_T := -1.0*F_T;


          if F_T >= F_TRerandSorted[aB] then
            Count_T_beta := Count_T_beta + 1;
        end; {end of rerandomization loop for power}

    Write(Lst,'  ',100*Depletion:3:0,'% ',Count_T_beta/Rerand:4:3);
  end; {of depletion % loop}
  end.
```