

Using SAS® To Control Multistream Binomial Processes

Peter Wludyka, University of North Florida, Jacksonville, FL & Sheri Jacobs of Vistakon, Inc.

ABSTRACT

Process control schemes for multistream binomial processes are described. A multistream process is one in which the streams are uncorrelated or very weakly correlated. When each of the J streams are identical (a homogeneous multistream process) one can use a group control chart (a k -sigma p -chart on which only the largest and smallest of the J sample proportions nonconforming are plotted). A test (runs rule) that signals when the same stream produces the largest proportion nonconforming R consecutive times is presented along with two SAS® programs useful for designing runs rule schemes, p -charts, and Chi-Squared control charts.

INTRODUCTION

A multistream process is best introduced by means of an example. Consider an injection molding process that makes a plastic toy. Tubes running from a reservoir carry liquid plastic into molds in an apparatus (a frame) which contains four molds. Suppose for simplicity that one tube is used to fill each toy. Each mold location can be considered to be a "stream" and this will be referred to as a 4-stream process. After cooling and separation from the mold a toy can be inspected and classified as defective (nonconforming) or nondefective (conforming). A process quality control scheme is a methodology for determine whether the process is producing "conforming units" in a stable and predictable manner; that is, one may decide whether the process is in-control or out-of-control. The control schemes described here require that periodically samples of product be inspected. One might be concerned with two ways in which the process could cease to be in-control.

1. A process change that affects all of the streams
2. A change that impacts on one or more streams

The first might correspond with to a poor grade of plastic being used; the second to a clog or other problem with an individual tube.

Note that identifying which type of out-of-control condition occurs will be useful for bringing the process back under control.

PROCESS MODELING AND CONTROL CHARTS

Control charts can be used to control processes (see Montgomery). In order to use control charts a statistical model for the process is required. Suppose that n frames are selected for inspection at the end of each epoch. Then each stream will be sampled n times also. If the streams are uncorrelated then the process is called a J -stream multistream process. A plausible process model then is that each stream is a binomial process with n trials and success probability p , where p is the proportion of items in the process (stream) that are nonconforming. When p is the same for all the streams the process is a homogeneous multistream process. (See Jacobs and Wludyka for details). The process is out-of-control if

1. The overall p changes
2. p changes for one or more streams

A frequently used tool for detecting changes in a binomial process is the p -chart (see Montgomery). The sample proportion nonconforming is plotted on the chart at each epoch and a signal occurs whenever the sample proportion plots outside the k -sigma control limits. Typically $k = 3$ since experience has shown this to be a useful control band width. For the multistream process one may

1. construct a single overall p -chart

2. Construct J individual p -charts (one for each stream).

3. Use a Chi-squared chart

In (2) it is unwise to set $k = 3$ since the false alarm rate will usually be much higher than most quality engineers would find reasonable. That is, k should be greater than three. See Jacobs and Wludyka for methods for determining the proper k . Note that is it sufficient to plot only the largest and smallest of the J sample (stream) proportions on a single chart for (2). This chart is called a group p -chart. The Chi-squared chart will not be described in this paper; however, the simulation program provided produces estimated ARLs (average run lengths) for this chart.

RUNS RULES

A runs rule is a simple tool for determining whether one (or more) of the streams have shifted (p has changed). At each epoch the stream for which the proportion nonconforming is largest is determined. This will be used to produce a one-sided runs rule. This has the advantage of being simple to use. When the same stream has the largest proportion nonconforming for r consecutive times that will be defined to be a run of length r . When r equals some constant R (chosen by the user) then an out-of-control signal occurs. R is chosen such that the in-control process will have a large average run length (ARL). The ARL is defined as the average number of samples until a signal occurs. When the process is out-of-control a short ARL is desired. Since ties can occur for the stream with the highest sample proportion nonconforming there are several ways in which a "run" can be defined. Four rules are:

1. The run is maintained on a tie only if the run stream is included in the tie
2. The run length is maintained on all ties
3. The run continues (is increased) on a tie if the run stream is in the tie
4. The run continues on a tie that includes the run stream and is maintained on a tie that does not include the run stream.

Each of these has a variant in which the run starts over whenever there is a j -way tie at zero, which is called a zero restart rule. These rules are denoted Rule 1a, ..., Rule 4a.

A RUNS RULE ARL PROGRAM

Given the process proportion nonconforming, p , one can then choose values for n and R that produce ARLs that have suitable properties. Note that p is usually estimated from in-control process data. The program below produces a series of tables that aid the user the selecting a control scheme.

```
/* ARLs for Runs Rules */
data rule2mat;
j=4; /* the number of streams */
do p0=0.023; /* proportion nonconforming */
do n=75; /* sample size per stream */
do idelta=0.0, .01, .02, .03, .04, .05, .08, .10;
delta=idelta; /* process shift vales */
p1=p0+delta; p2=p0;
alltie =
probbnml(p1,n,0)*(probbnml(p2,n,0))**(j-1);
/* calculate t1 */
q1 = 0; q2=0;
t1 = 0; t2in1=0; t2out1=0; t2=0;
```

STATISTICS AND DATA ANALYSIS

```

do i=1 to n;
  py2eqi = probbnml(p2,n,i)-
  probbnml(p2,n,i-1);
  py2lti = probbnml(p2,n,i-1);
  py1lti = probbnml(p1,n,i-1);
  py1eqi = probbnml(p1,n,i)-
  probbnml(p1,n,i-1);
  t1wstep = py2eqi**2*py2lti**(j-
3)*py1lti;
  t2in1stp = py2eqi*py2lti**(j-2)*py1eqi;
  t2ot1stp = py2eqi**2*py2lti**(j-
3)*py1lti;
  q1 = q1+ py1eqi*py2lti**(j-1);
  q2 = q2 + py2eqi*py2lti**(j-2)*py1lti;
  t = 1-(q1+(j-1)*q2);
  alltie = alltie + py1eqi*py2eqi**(j-1);
  do w = 2 to j-1;
    c1 =
gamma(j)/(gamma(w+1)*gamma(j-w));
    c2 = gamma(j-
1)/(gamma(w)*gamma(j-w));
    if w = j-1 then c3 = 0;
    else c3 = gamma(j-
1)/(gamma(w+1)*gamma(j-w-1));

    if w = 2 then do;
      t1 = t1+c1*t1wstep;
      t2in1 = t2in1 + c2*t2in1stp;
      t2out1 = t2out1 + c3*t2ot1stp;
      t2 = t2 + c2*t2in1stp +
c3*t2ot1stp;
    end;
    else do;
      t1wstep =
t1wstep*(py2eqi/py2lti);
      t1 = t1+c1*t1wstep;
      t2in1stp =
t2in1stp*(py2eqi/py2lti);
      t2in1 = t2in1 + c2*t2in1stp;
      t2 = t2 + c2*t2in1stp;

      t2ot1stp =
t2ot1stp*(py2eqi/py2lti);
      t2out1 = t2out1 +
c3*t2ot1stp;

      t2 = t2 + c3*t2ot1stp ;

    end;
  /* output; */
end;
z0 = (1-p2)**((j-1)*n)*(1-p1)**n;
output;
end; end;
proc print;
  var j n p0 delta q1 q2 t1 t2 t z0;
run;

data probs;
  set rule2mat;

```

```

  keep j n q1 q2 t t1 t2 p0 delta z0;
/* arl routine for rule 1 follows */
proc iml;
  use probs;
  /* show datasets; show contents;*/
  labtab = J(1,3,0);
  ptab = J(8,8,0); ptaba = J(8,8,0);
  do step = 1 to 8; /* print step;*/
    read next; /* list;*/
    do instep= 2 to 8; r=instep; /* print r;
*/
    /* print j t1 t2 t q1 p0 ; */
    P=J(2*r-1,2*r-1,0);
    /* print P;*/
    if r > 2 then do;
      do row = 4 to 2*r-1 by 2;

p[row,1]=t1;p[row,2]=0;p[row,3]=(J-
1)*q2;p[row,row]=t-t1;if row < 2*r-2 then
p[row,row+2]=q1;

p[row+1,1]=t2;p[row+1,2]=q1;p[row+1,3]=(J-
2)*q2;p[row+1,row+1]=t-t2; if row < 2*r-2 then
p[row+1,row+3]=q2;
      end;end;
      p[1,1]=t; p[1,2]=q1; p[1,3]=(J-
1)*q2;
      p[2,1]=t1; p[2,2]=t-t1; p[2,3]=(J-
1)*q2; if r > 2 then p[2,4]=q1;
      p[3,1]=t2; p[3,2]=q1; p[3,3]=t-
t2+(J-2)*q2; if r > 2 then p[3,5]=q2;
      /* print P;*/
      Imat=I(2*r-1); /* print Imat;*/
      one=J(2*r-1,1,1); /* print one;*/
      arl=inv(Imat-P)*one;
      ptab[step,1]=delta;ptab[step,r]=arl[1];
      /* print 'arl using rule 1';
      print arl; */
      pzero = p;
      do rstep = 2 to 2*r-1;
        pzero[rstep,1]=p[rstep,1]+z0;

pzero[rstep,rstep]=p[rstep,rstep]-z0;
      end;
      arlz = inv(Imat-pzero)*one;

ptaba[step,1]=delta;ptaba[step,r]=arlz[1];
end;end;
print 'arl table for rule 1';
tag1 = {J n p0};

labtab[1,1]=j;labtab[1,2]=n;labtab[1,3]=p0;
print labtab[colname=tag1];
tag2= {delta r2ARL r3ARL r4ARL r5ARL
r6ARL r7ARL r8ARL};
print ptab[colname=tag2 format = 10.2];
print 'ARL Table for Rule 1a';
print ptaba[colname=tag2 format = 10.2];

/* arl routine for rule 2 follows */

```

```

proc iml;
    use probs;
    /* show datasets; show contents;*/
    labtab = J(1,3,0);
    ptab = J(8,8,0); ptaba = J(8,8,0);
    do step = 1 to 8; /* print step;*/
    read next; /* list;*/
    do instep= 2 to 8; r=instep; /* print r;
    */
    /* print j t1 t2 t q1 p0 ; */
    P=J(2*r-1,2*r-1,0);
    /* print P;*/
    if r > 2 then do;
        do row = 4 to 2*r-1 by 2;
            p[row,1]=0;p[row,2]=0;p[row,3]=(J-
            1)*q2;p[row,row]=t;if row < 2*r-2 then
            p[row,row+2]=q1;

            p[row+1,1]=0;p[row+1,2]=q1;p[row+1,3]=(J-
            2)*q2;p[row+1,row+1]=t; if row < 2*r-2 then
            p[row+1,row+3]=q2;
            end;end;
            p[1,1]=t; p[1,2]=q1; p[1,3]=(J-1)*q2;
            p[2,1]=0; p[2,2]=t; p[2,3]=(J-1)*q2;
        if r > 2 then p[2,4]=q1;
            p[3,1]=0; p[3,2]=q1; p[3,3]=t+(J-
            2)*q2; if r > 2 then p[3,5]=q2;
            /* print P;*/
            Imat=I(2*r-1); /* print Imat;*/
            one=J(2*r-1,1,1); /* print one;*/
            arl=inv(Imat-P)*one;
            ptab[step,1]=delta;ptab[step,r]=arl[1];
            /* print 'arl using rule 2';
            print arl; */
            pzero = p;
            do rstep = 2 to 2*r-1;
                pzero[rstep,1]=p[rstep,1]+z0;

            pzero[rstep,rstep]=p[rstep,rstep]-z0;
            end;
            arlz = inv(Imat-pzero)*one;

            ptaba[step,1]=delta;ptaba[step,r]=arlz[1];
            end;end;
            print 'arl table for rule 2';
            tag1 = {J n p0};

            labtab[1,1]=j;labtab[1,2]=n;labtab[1,3]=p0;
            print labtab[colname=tag1];
            tag2= {delta r2ARL r3ARL r4ARL r5ARL
            r6ARL r7ARL r8ARL};
            print ptab[colname=tag2 format=10.2];
            print 'ARL Table for Rule 2a';
            print ptaba[colname=tag2 format = 10.2];
        /* arl routine for rule 3 follows */
        proc iml;
            use probs;
            /* show datasets; show contents;*/
            labtab = J(1,3,0);

```

```

            ptab = J(8,8,0); ptaba = J(8,8,0);
            do step = 1 to 8; /* print step;*/
            read next; /* list;*/
            do instep= 2 to 8; r=instep; /* print r;
            */
            /* print j t1 t2 t q1 p0 ; */
            P=J(2*r-1,2*r-1,0);
            /* print P;*/
            if r > 2 then do;
                do row = 4 to 2*r-1 by 2;
                    p[row,1]=t1; p[row,2]=0;
                    p[row,3]=(J-1)*q2; if row < 2*r-2 then
                    p[row,row+2]=q1+t-t1;

                    p[row+1,1]=t2;p[row+1,2]=q1;p[row+1,3]=(J-2)*q2;
                    if row < 2*r-2 then p[row+1,row+3]=q2+t-t2;
                    end;end;
                    p[1,1]=t; p[1,2]=q1; p[1,3]=(J-1)*q2;
                    p[2,1]=t1; p[2,2]=0; p[2,3]=(J-1)*q2;
                if r > 2 then p[2,4]=q1+t-t1;
                    p[3,1]=t2; p[3,2]=q1; p[3,3]=(J-2)*q2;
                if r > 2 then p[3,5]=q2+t-t2;
                    /* print P;*/
                    Imat=I(2*r-1); /* print Imat;*/
                    one=J(2*r-1,1,1); /* print one;*/
                    arl=inv(Imat-P)*one;
                    ptab[step,1]=delta;ptab[step,r]=arl[1];
                    /* print 'arl using rule 3';
                    print arl; */
                    pzero = p;
                    do rstep = 2 to 2*r-1;
                        pzero[rstep,1]=p[rstep,1]+z0;

                    pzero[rstep,rstep]=p[rstep,rstep]-z0;
                    end;
                    arlz = inv(Imat-pzero)*one;

                    ptaba[step,1]=delta;ptaba[step,r]=arlz[1];
                    end;end;
                    print 'arl table for rule 3';
                    tag1 = {J n p0};

                    labtab[1,1]=j;labtab[1,2]=n;labtab[1,3]=p0;
                    print labtab[colname=tag1];
                    tag2= {delta r2ARL r3ARL r4ARL r5ARL
                    r6ARL r7ARL r8ARL};
                    print ptab[colname=tag2 format=10.2];
                    print 'ARL Table for Rule 3a';
                    print ptaba[colname=tag2 format = 10.2];
                /* arl routine for rule 4 follows */
                proc iml;
                    use probs;
                    /* show datasets; show contents;*/
                    labtab = J(1,3,0);
                    ptab = J(8,8,0); ptaba = J(8,8,0);
                    do step = 1 to 8; /* print step;*/
                    read next; /* list;*/
                    do instep= 2 to 8; r=instep; /* print r;
                    */

```

STATISTICS AND DATA ANALYSIS

```

/* print j t1 t2 t q1 p0 ; */
P=J(2*r-1,2*r-1,0);
/* print P;*/
if r > 2 then do;
    do row = 4 to 2*r-1 by 2;

p[row,1]=0;p[row,2]=0;p[row,3]=(J-
1)*q2;p[row,row]=t1;if row < 2*r-2 then
p[row,row+2]=q1+t-t1;

p[row+1,1]=0;p[row+1,2]=q1;p[row+1,3]=(J-
2)*q2;p[row+1,row+1]=t2; if row < 2*r-2 then
p[row+1,row+3]=q2+t-t2;
        end;end;
        p[1,1]=t;p[1,2]=q1;p[1,3]=(J-1)*q2;
        p[2,1]=0;p[2,2]=t1;p[2,3]=(J-1)*q2; if
r > 2 then p[2,4]=q1+t-t1;
        p[3,1]=0;p[3,2]=q1;p[3,3]=t2+(J-2)*q2;if
r > 2 then p[3,5]=q2+t-t2;
        /* print P;*/
        Imat=I(2*r-1); /* print Imat;*/
        one=J(2*r-1,1,1); /* print one;*/
        arl=inv(Imat-P)*one;
        ptab[step,1]=delta;ptab[step,r]=arl[1];
        /* print 'arl using rule 2';
        print arl; */
        pzero = p;
        do rstep = 2 to 2*r-1;
            pzero[rstep,1]=p[rstep,1]+z0;

pzero[rstep,rstep]=p[rstep,rstep]-z0;
        end;
        arlz = inv(Imat-pzero)*one;

ptaba[step,1]=delta;ptaba[step,r]=arlz[1];
        end;end;
        print 'arl table for rule 4';
        tag1 = {J n p0};

labtab[1,1]=j;labtab[1,2]=n;labtab[1,3]=p0;
        print labtab[colname=tag1];
        tag2= {delta r2ARL r3ARL r4ARL r5ARL
r6ARL r7ARL r8ARL};
        print ptab[colname=tag2 format=10.2];
        print 'ARL Table for Rule 4a';
        print ptaba[colname=tag2 format = 10.2];

```

Program output and interpretation

A series of tables are produced which can be used to pick R and n .

Table 1: Runs Rule ARL Output

arl table for rule 1

LABTAB	J	N	P0
	4	75	0.023
DELTA	R4ARL	R5ARL	R6ARL
0.00	195.01	926.44	4395.88
0.01	73.95	200.92	522.68
0.02	23.50	43.29	77.56
0.03	11.74	18.04	26.83
0.04	7.77	10.90	14.72
0.05	6.05	8.07	10.36
0.08	4.41	5.59	6.79
0.10	4.15	5.20	6.27

These are partial outputs; that is those for Rule1 (Table 1) and Rule 2 (Table 2) for selected values of R . Continuing with the TOY process, we have $J = 4$ streams in which the in-control proportion nonconforming is estimated to be 0.023. If one takes samples of size $n = 75$, then the in-control ARL using Rule 1 is 926.44 with $R = 5$ and the in-control ARL for Rule 2 is 492.71 with $R = 5$ (note that in-control corresponds to $\delta = 0$). If the process shifts upward by 0.03 for a single stream (so that the stream now produces 5.3% nonconforming) then using Rule 1 it will take on average 18 samples to detect the shift and 16.32 samples using Rule 2.

Table 2: Runs Rule ARLs for Rule 2

arl table for rule 2

LABTAB	J	N	P0
	4	75	0.023
DELTA	R4ARL	R5ARL	R6ARL
0.00	122.82	492.71	1972.28
0.01	55.46	137.12	322.56
0.02	20.37	35.57	60.14
0.03	10.91	16.32	23.57
0.04	7.49	10.36	13.78
0.05	5.93	7.86	10.01
0.08	4.40	5.57	6.76
0.10	4.14	5.20	6.26

A quality engineer might on this basis decide to use Rule 2 with $R = 5$; or, decide to examine other (larger) sample sizes to increase the power of the detection scheme. For example, increasing the sample size to 150 reduces the ARL for a shift of 0.03 to about 9 for each of the rules examined.

Given J , p , and n the program outputs ARLs for $R = 2, \dots, 8$ (and can be easily modified to larger values of R) for each of the eight rules so that an engineer can examine several scenarios before choosing a rule.

PROCESS SHIFTS

Runs rules are of no value for detecting an overall process shift. For that reason it is often wise to use the runs rule in conjunction with another tool, such as a process p -chart, in which the stream data is aggregated to form a single p -chart. It is also possible to use a Chi-Squared chart with a runs rule or a group p -chart (see Jacobs and Wludyka for a thorough explanation). Using a control chart along with a runs rule will decrease all ARLs, especially the in-control ARL, so care must be used when employing such a control scheme.

The advantage of a dual scheme is that both stream shifts and overall shifts are more easily detected.

If one plans to use a p -chart with a runs rule it is probably wise to choose a Runs Rule with an in-control ARL of 500 or more. Then using a p -chart with $k = 3$ will probably likely produce a scheme with in-control ARL of about 225.

SIMULATION TOOLS FOR CONTROL SCHEMES

Monte Carlo methods can be used to estimate the ARLs associated with a complex control scheme.

Program for Runs Rule with a P-chart ARLs

```

/*****
*****
*      simulates ARLs: Runs rule 2 (maintain
run on ties) + p-chart
*      overall p-chart k = (user's choice)
*      group p-charts k1,k2 = equated
*      ChiSquare chart based on nominal ARL0
(user specified)
*      Default: case 1 to case 2
(ALPHA{1}=DELTAJ and DELTA + 0)
*      Modify: case 1 to case 1 ( ALPHA{1} = 0
and DELTA=DELTAJ)
*      signal(1) = Overall p-chart
*      signal(2) = pj
*      signal(3) = chi-square overall
*      signal(4) = chi-square for j-columns;
*      signal 10 - 20 for individual pj
*      signal(13) = chisq overall
*      signal(14) = chisq for j cols
*****
*****/
data simdat1;
  do jstrs=3; /*users list for loop*/
  do n=50; /*user list for loop*/
    /* determine control limits */
    p0=0.10;
    inum=1;jnum=jstrs;
    kovp = 3.0;
    ovp = p0;
    ovpucl = min(1,ovp + kovp*sqrt((ovp*(1-
ovp))/(inum*jnum*n)));
    ovplcl = max(0, ovp - kovp*sqrt((ovp*(1-
ovp))/(inum*jnum*n)));

```

```

    ovpnucl =int(n*inum*jnum*( min(1,ovp +
kovp*sqrt((ovp*(1-ovp))/(inum*jnum*n))));
    ovplcl =int(n*jnum*inum*( max(0, ovp -
kovp*sqrt((ovp*(1-ovp))/(inum*jnum*n))));
    /* determine k1 and k2 by equating arl0
to p-chart */
    uclpx=(ovp+kovp*sqrt((ovp*(1-
ovp))/(Jnum*n)))*(Jnum*n);
    lclpx=(ovp-kovp*sqrt((ovp*(1-
ovp))/(Jnum*n)))*(Jnum*n);
    if lclpx <=0 then powp0=1-
probbnml(ovp,Jnum*n,floor(uclpx));
    else powp0 = 1-probbnml(ovp,Jnum*n,
floor(uclpx))+probbnml(ovp,Jnum*n,floor(lclpx));
    /* */
    pj0=p0;
    k1=2.9;
    do until (diff < 0 or k1 > 6);
    k1=k1+.01;
    uclpx=(pj0+k1*sqrt((pj0*(1-pj0))/(n)))*(n);
    lclpx=(pj0-k1*sqrt((pj0*(1-pj0))/(n)))*(n);
    if lclpx <=0 then powpj0=1-
probbnml(pj0,n,floor(uclpx));
    else powpj0 = 1-probbnml(pj0,n,
floor(uclpx))+probbnml(pj0,n,floor(lclpx));
    powpj0 = 1-(1-powpj0)**jnum;
    diff = powpj0 - powp0;
    end;
    k2=k1-.01;
    uclpx2=(pj0+k2*sqrt((pj0*(1-pj0))/(n)))*(n);
    lclpx2=(pj0-k2*sqrt((pj0*(1-pj0))/(n)))*(n);
    /*
put 'powp0 ='powp0 '      powpj0= 'powpj0;
    */
    /* end k1 and k2 */
    kpj = k1;
    pj0 = p0;
    pjuc1 = min(1,pj0 + kpj*sqrt((pj0*(1-
pj0))/(inum*n)));
    pjlc1 = max(0,pj0 - kpj*sqrt((pj0*(1-
pj0))/(inum*n)));
    pjnucl =int( min(1,pj0 +
kpj*sqrt((pj0*(1-pj0))/(inum*n))));
    pjnlcl = int( max(0,pj0 -
kpj*sqrt((pj0*(1-pj0))/(inum*n))));
    kpj=k2;
    pjuc2 = min(1,pj0 + kpj*sqrt((pj0*(1-
pj0))/(inum*n)));
    pjlc2 = max(0,pj0 - kpj*sqrt((pj0*(1-
pj0))/(inum*n)));
    pjnucl2=int( min(1,pj0 +
kpj*sqrt((pj0*(1-pj0))/(inum*n))));
    pjnlcl2 = int( max(0,pj0 -
kpj*sqrt((pj0*(1-pj0))/(inum*n))));
    ncsarl = 800;
    csqcl = cinv(1-(1/ncsarl),inum*jnum);
    csqclq = cinv(1-(1/ncsarl),jnum);
    do deltax = 0.0,0.01,0.02,0.05,0.10,.2;
    array tsig(30);
    do init = 1 to 30;

```

STATISTICS AND DATA ANALYSIS

```

        tsig(init)=0;
end;
seed1 = -1;
numreps = 1000;
totr=0; count=0;repct=0;
do reps = 1 to numreps;

    R=5;
    if jstrs = 3 then r = 6;
    if (jstrs = 3 and n=20 and p0=.01) then
r = 5;
        if jstrs = 4 then r=5;
        if jstrs = 5 then r=4;
            if (jstrs=5 and n=100 and p0 >
.05) then r=5;
            if (jstrs=5 and n=200 and p0 >
.01) then r=5;

            if jstrs = 8 then r=4;
            if jstrs = 10 then r=4;
                if (jstrs = 10 and n = 20 ) then
r = 3;
                    if (jstrs = 10 and n = 50 and p0
= .01) then r=3;
                        if jstrs > 10 then r = 3;
                        totr=totr+count;
                        /*
                        put 'count =' count totr;
                        */
                        count =0;
                        rsig=0;
                        rlength=1;
                        prevmaxj=-1;

                        do until ( rsig = 1 or count > 3000 );
                        repct=repct+1;
                        count = count+1;
                        /*
                        put rlength;
                        */

                        inum=1;
                        jnum=JSTRS;
                        /* p0=0.2; */
                        delta = 0; /* deltaj; */
                        array alpha(20);
do ainit = 1 to 20;
                    alpha{ainit}=0.0;
end;

                    alpha{1}= deltaj; /* 0 */
                    alpha{2}=0.0;
                    alpha{3}=0.0;

                    array signal(30);
do ksig = 1 to 30;
                    signal(ksig) = 0;
end;

                    array p(1,20);
                    array y(1,20);

```

```

        array pjhat(20);
        array yjsum(20);
do i=1 to inum;
    do j=1 to jnum;
        p(i,j) = p0 + delta + alpha(j);

end;end;
/* n = 50; */

/* generate random samples */
do i=1 to inum;
    do j=1 to jnum;
        pij=p(i,j);
        y(i,j) = ranbin(seed1,n, p(i,j));
end;end;
/* calculate phat,pjhat, chisq overall, chisq-
for-j-columns
    test for signals */
/* calc run length */
array ystr(20);
maxj = 1;
tiesw = 0;
ystr(1)=y(1,1);
do jstep = 2 to jnum;
    ystr(jstep) = y(1,jstep);
    if ystr(maxj) ge ystr(jstep)
then
        if ystr(maxj) eq
ystr(jstep) then
            tiesw =1;
            else maxj = maxj;
            else do; maxj = jstep ; tiesw =
0; end;

        end;
        /*
        put 'before' ystr1 ystr2 ystr3 ystr4 ystr5 maxj
        tiesw prevmaxj rlength count;
        */
        if tiesw = 0 then
        if maxj=prevmaxj then rlength = rlength +1;
        else do;
            prevmaxj=maxj;
            rlength=1;
        end;
        /*
        put 'after ' ystr1 ystr2 ystr3 ystr4 ystr5 maxj
        tiesw prevmaxj rlength count;

        */
        if rlength = r then rsig =1;

        /* end calc run length */
        ysum = 0;
        zsqsum = 0;
        zsqsumq = 0;
        do j = 1 to jnum;
            yjsum(j) = 0;

        do i = 1 to inum;

```

```

        ysum = ysum+y(i,j);
        yjsum(j) = yjsum(j) + y(i,j);
        zsqsum = zsqsum +(y(i,j)-n*p0)**2;
    end;
    pjhat(j)=yjsum(j)/(n*inum);
    if pjhat(j) gt pjuc1 or pjhat(j) lt pjlc1 then
        signal(2) = 1;
    if pjhat(j) gt pjuc12 or pjhat(j) lt pjlc12 then
        signal(4) = 1;
    if pjhat(j) gt pjuc1 or pjhat(j) lt pjlc1 then
        signal(j+9) = 1;
        else signal(j+9) = 0;
    zsqsumq = zsqsumq + (yjsum(j) - n*inum*p0)**2;
    end;
    phat=ysum/(n*inum*jnum);
    csqst = (zsqsum)/(p0*(1-p0)*n);
    csqstq = (zsqsumq)/(p0*(1-p0)*n*inum);
    if csqst gt csqcl then signal(3) = 1;
        else signal(3) = 0;
    /* if csqstq gt csqclq then signal(4) = 1;
        else signal(4) = 0; */
    if phat gt ovpucl or phat lt ovplcl then
        signal(1) = 1;
        else signal(1) = 0;
    do ksig2 = 1 to 20;
        tsig(ksig2)=tsig(ksig2)+signal(ksig2);
    end;
    if signal(1)=1 then rsig=1;
    end;
    end;
    end;
    totr=totr+count;
    arlRp=totr/numreps;
    arlp=repct/tsig1;
    arlpj=repct/tsig2;
    arlpj2=repct/tsig4;
    arlchi=repct/tsig3;
    output;
    end;
END;
run;
/*
proc print;
    by jstrs n deltaj;
    var repct arlRp arlp arlpj arlchi tsig1
    tsig2 tsig3 tsig4 tsig5 tsig6 tsig7 tsig7 tsig8
    tsig9
    tsig10 tsig11 tsig12 tsig13 tsig14
    tsig15
    r csqcl csqst csqclq csqstq
    p1 p2 p3 p4 p5 p6 p7 p8 p9 p10 p11 p12
    p13 p14 p15 p16
    alpha1 alpha2 alpha3 alpha4;
    /*
title 'CASE 1 TO CASE 2'; /* if delta > 0 then
title 'case 1 to case 2';*/
proc tabulate data=simdat1;
    BY JNUM;
    class p0 deltaj n;

```

```

var arlp arlchi arlpj arlpj2 arlRp r k1
k2;
table p0,deltaj,n*(arlp*f=6.1
arlchi*f=6.1 arlpj*f=6.1 arlpj2*f=6.1
arlRp*f=6.1 r*f=1.0 k1*f=4.2 k2*f=4.2)*(MEAN);
KEYLABEL
MEAN='ARL';
run;

```

Program output and interpretation

Returning to the TOY process (see Table 3, where ARLP is the average run length for a 3-sigma p-chart, ARLCHI is the ARL for a Chi-Squared chart, ARLPJ is the ARL for a group p-chart with $K1 = 4.07$ sigma limits, ARLPJ2 is the ARL for a group p-chart with $K2=4.06$ sigma limits) one sees that for a scheme in which a $k=3$ sigma overall p-chart is used in conjunction with a runs rule with $R = 5$ the in-control ARL is about 149. This drop should be attributed to the fact that the in-control ARL for the p-chart is about 212. A user might consider increasing k with the overall p-chart.

Table 3: Simulation Output

P0 0.023	N							
	75							
	ARLP	ARLCHI	ARLPJ	ARLPJ2	ARLP	R	K1	K2
	ARL	ARL	ARL	ARL	ARL	ARL	ARL	ARL
DELTAJ								
0	211.6	149.1	707.9	140.8	149.4	5	4.07	4.06
0.01	32.4	64.5	228.8	62.1	58.9	5	4.07	4.06
0.02	48.9	24.3	64.8	22.4	28.7	5	4.07	4.06
0.05	8.5	3.9	5.2	8.0	5.2	5	4.07	4.06
0.1	2.1	1.2	1.4	1.2	2.0	5	4.07	4.06
0.2	1.0	1.0	1.0	1.0	1.0	5	4.07	4.06

IMPLEMENTATION

Below is a SAS program that produces a 3-sigma p-chart for the TOY process. The streams are denoted by S1, ..., S4, which contain the number of defective items in each sample. There is data for eight epochs and the proportion defective is known to be 0.023. Since each of the four streams has sample size 75, the sample size for the overall p-chart is 300.

```

options;
data toydat;
    input s1 s2 s3 s4 sample;
    defect = s1+s2+s3+s4;
    cards;
1 2 1 2 1
0 2 3 0 2
2 0 2 4 3
2 3 0 5 4
4 2 1 7 5
3 4 2 4 6
1 1 3 5 7
2 3 1 4 8
;
proc shewhart data=toydat graphics;

```

STATISTICS AND DATA ANALYSIS

```
pchart defect*sample / subgroupn = 300  
p0 = 0.023;  
run;
```

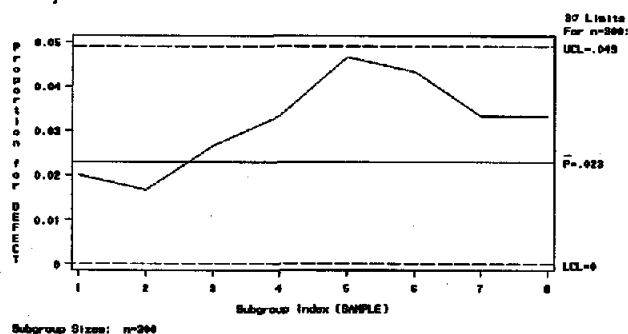


Figure 1: P-chart for TOY process

Using runs Rule 2 with $R = 5$, observe that at epoch 1 (sample 1) there is a tie for the max stream so there is no run-stream; at epoch 2 stream 3 has the largest proportion defective and the run length is one (that is, $r = 1$); at epoch 3 stream four has the largest proportion defective (there is a new max stream), and hence $r = 1$. Also, at epoch 4 stream 4 is the max and hence $r = 2$; at epoch 5 stream four is the max and hence $r = 3$; at epoch 6 there is a tie so r remains 3; at epoch 7 stream 4 is the max so $r = 4$ and at epoch 8 stream 4 is the max so $r = 5$. Since the run length is equal to $R = 5$, there is an out-of-control signal at epoch 8.

CONCLUSION

Several process control schemes useful for controlling multistream binomial processes have been discussed. A SAS® program for calculating ARLs for runs rules is provided. This program is a useful tool when designing a runs rule scheme. Next a SAS® monte carlo simulation program is provided that produces estimated ARLs that are useful for complex schemes, such as those using a runs rule in conjunction with an overall p -chart. Simulation study has shown this to be an effective process control scheme for multistream binomial processes.

DOWNLOADING SAS® PROGRAMS

Source code can be downloaded from the University of North Florida Center for Research and Consulting in Statistics web page (www.unf.edu/coas/math-stat/CRCS) as technical reports #080101 and #080201.

REFERENCES

Jacobs, S., and Wludyka, P., "Runs Rules and P-Charts for Multistream Binomial Processes," to appear (and CRCS technical report #080201).

Montgomery, D. C. (1997). *Introduction to Statistical Quality Control*, 3rd ed., John Wiley and Sons, New York.

CONTACT INFORMATION

Your comments and questions are valued and encouraged.
Contact the authors at:

Peter Wludyka
Associate Professor of Statistics
University of North Florida
Jacksonville, Florida
Work Phone: 904-620-1048
Fax: 904-620-2818
Email: pwludyka@unf.edu
Web: www.unf.edu/coas/math-stat/~pwludyka

Sheri Jacobs
Quality Engineer
Vistakon, Inc.
Jacksonville, Florida
Work Phone: 904-443-1607
Fax: 904-443-1607
Email: sjacobs@visius.jnj.com