Monte Carlo Calibration of Distributions of Partition Statistics

John Sall, SAS Institute, Nov 18, 2002

Introduction

In recursive partitioning (decision trees) a search is made for each factor as to how to best make a split. Because of this searching over many possibilities, the p-values for the split itself do not have the claimed distribution. Most often a p-value multiplicity correction is made using the Bonferroni adjustment. The p-value is multiplied by B, where B is the total number of ways of arranging c levels into two groups (we evaluate only 2-group partitions in this study). For the unordered nominal case, $B = 2^{c-1}-1$. For the ordinal case, this is B=c-1. It turns out that this strongly overcorrects, and thus we investigated fitting the distribution to obtain better p-values.

Initial Study for Unordered Factor

First, we study the distribution of the unadjusted p-value and the Bonferroni-adjusted p-value.

All simulations were done in the null case, i.e. the data was completely random, and there should be no association between the response and factor except by random coincidence. 5000 trials were made at even numbers of levels between 2 and 80; the number of levels c is called nx in the simulations. Each sample had 500 observations. The response had two levels.

The script to do this was done in JMP's scripting language (JSL). A JSL function 'BestPartition' was created to interface with the partition searching code in JMP's Partition platform. The chi-square values were then stored in a JMP table with formulas to calculate the various p-values.

```
ny = 2;
n = 500;
nxx = 40; nxd = 2; // nx will go from 1*nxd to nxx*nxd
m = 5000;
g2vec = j(m*nxx,1,0);
nxvec = j(m*nxx,1,0);
gg2 = 0;

ij = 0;
for(j=0,j<nxx,j++,
 nx = j*nxd; if (nx==0,nx=2);
 show(nx);
 for(i=1,i<=m,i++,
   xx = j(n,1,randomInteger(nx)-1);
   yy = j(n,1,randomInteger(ny)-1);
   {l1,l2,gg2} = BestPartition(xx,yy);
   ij++;
   g2vec[ij] = gg2;
   nxvec[ij] = nx;
 )
);

dt = newTable("PValMonteCarloLarge.jmp");
dt<<NewColumn("G2",values(g2vec));
dt<<NewColumn("NX",nominal,values(nxvec));
dt<<NewColumn("PV", formula(1-ChiSquareDistribution(:G2,1)));
dt<<NewColumn("B",  formula(2^(:NX-1)-1));
dt<<NewColumn("PVB",formula(:PV*:B));
dt<<NewColumn("PVSQRTB",formula(:PV*Sqrt(:B)));
dt<<NewColumn("LogPV",formula(-Log(:PV)));
dt<<NewColumn("LogPVB",formula(-Log(:PVB)));
dt<<NewColumn("LogPVSqrtB",formula(-Log(:PV*Sqrt(:B))));
dt<<RunFormulas();

obj=Oneway(x(NX),Y(LogPV,LogPVB,LogPVSqrtB),Box Plots(1),GrandMean(0));

r = obj<<report;
r[1][FrameBox(1)]<<addGraphicsScript(hline(-log(.25));hline(-log(.5));hline(-log(.75)));
r[2][FrameBox(1)]<<addGraphicsScript(hline(-log(.25));hline(-log(.5));hline(-log(.75)));
r[3][FrameBox(1)]<<addGraphicsScript(hline(-log(.25));hline(-log(.5));hline(-log(.75)));
```

This creates a data table with 200,000 rows, 5000 for each number of factors. A similar script creates the data for the ordered case in which only splits in the original order are considered.

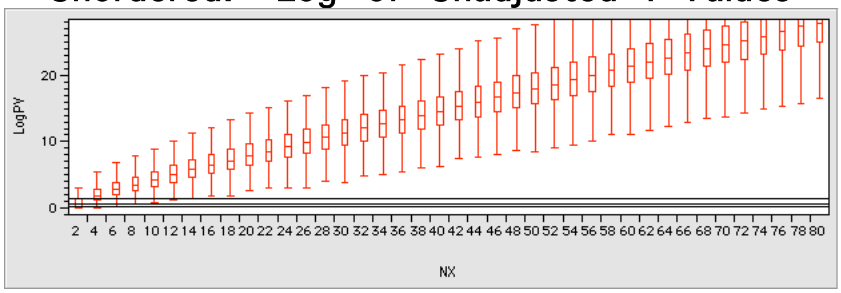
Results of the P-Values and adjusted P-Values for Unordered Case

The plots below show the box plots of the 5000 -log(pvalues) for each of the numbers of levels from 2 to 80 by 2. There are three reference lines drawn at -log(.25), -log(.5) and -log(.75). If the p-value distributions are right, the box plots should line up closely with the three reference lines. This occurs, as you would expect, for nx=2 levels. Notice

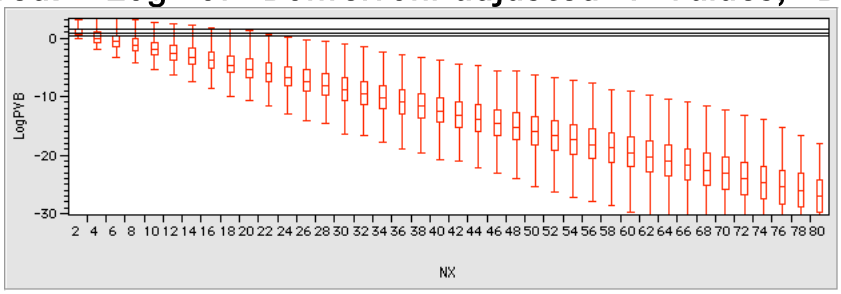
a. The unadjusted p-values are far too significant after the correct distribution for nx=2
b. The full adjustment by B makes things too-little significant, by far.
c. The pvalue-value transformation that makes sense is to use sqrt(B) instead of B.

The sqrt(B) transformation was suggested by plotting the resulting distribution quantiles and fitting a regression. This modified adjustment centers the distribution well, but the variance is far larger than what the p-value distribution should be.

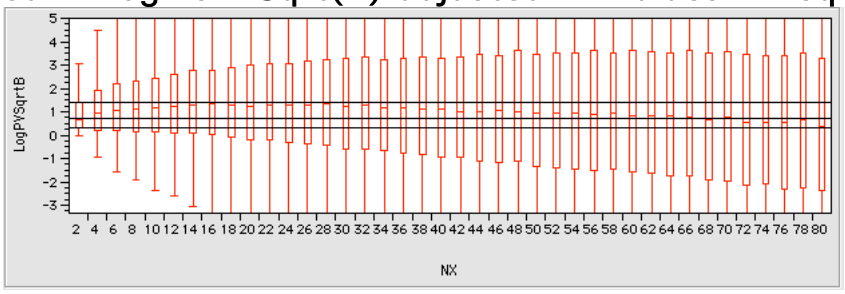
Unordered: -Log of Unadjusted P-Values



Unordered: -Log of Bonferroni-adjusted P-Values, $B=2^{c-1}-1$

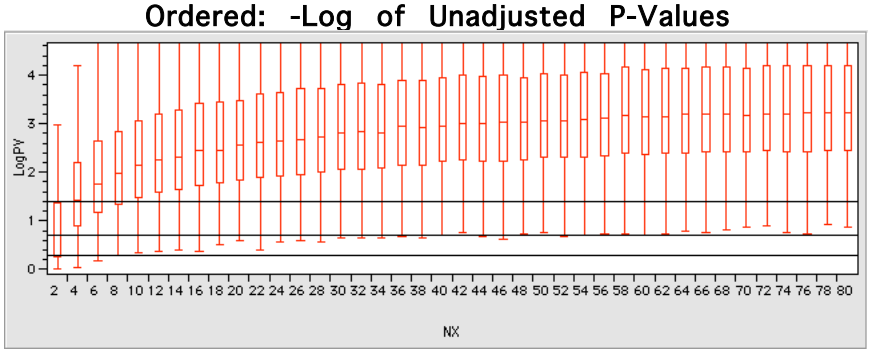


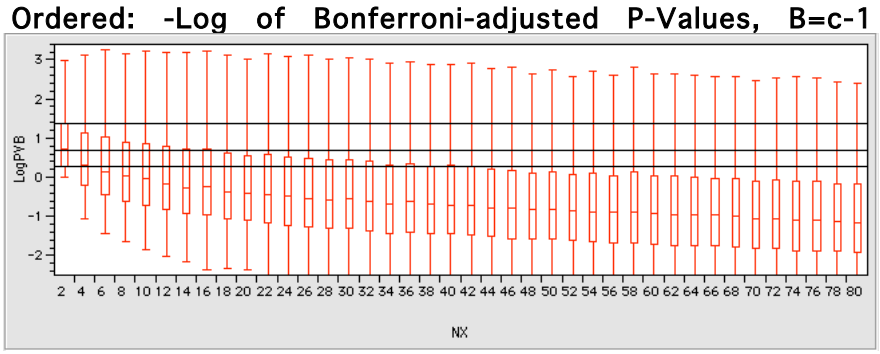
Unordered: -Log of Sqrt(B)-adjusted P-Values $B=\text{sqrt}(2^{c-1}-1)$

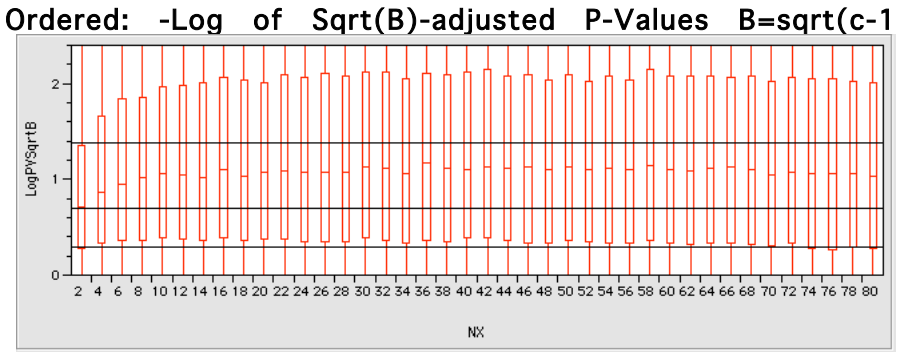


Initial Study for Ordinal Factor

The PValues and adjusted PValues are far better in the ordinal case, mostly because the number of combinations is much smaller. Still the ordinary pvalue is too significant. The Bonferroni pvalue suppresses it too much. Using Sqrt(B) instead of B for adjusting works out well for the center of the distribution, but the variance seem bigger than a chi-square.







Modeling the Test Statistic's Distribution -- Unordered Case

The conclusion from the above studies is that while a square root of B transformation to the p-value gets the right center of the distribution, it doesn't get the tails right. The null p-value distribution should look more uniform.

We undertook to fit the distribution empirically. The natural generalization of the chi-square distribution would be the gamma distribution. The gamma distribution is easily fit using the first two moments.

We wanted to fit a variety of number of X levels and numbers of Y levels to the data. The preliminary study above only covered 2 Y levels. We were especially concerned about fitting

the lower values, so we decided on a Fibonacci grid. Since the numbers of X levels needed to be kept small, since the computing time went up exponentially, we chose the Fibonacci numbers to 133 for NX, the number of levels of X. For NY, we chose the 7 Fibonacci numbers to 24. Each sample needed to be large enough to get counts into all the cells, and it seemed like n=800 was enough. To get a good handle on the test statistic distribution, we ran 5000 Monte Carlo trials. Thus the script to generate the data looked like this:

```
nyy = 7;  // ny will go Fibonacci, i.e. 2,3,5,8,13,21,34
nxx = 5;  // ny will go Fibonacci, i.e. 2,3,5,8,13,21
n = 800;  // number of rows in each data set in MonteCarlo
m = 5000; // number of MonteCarlo trials

g2vec = j(m*nxx*nyy,1,0);  nxvec = j(m*nxx*nyy,1,0);
nyvec = j(m*nxx*nyy,1,0);  lwvec = j(m*nxx*nyy,1,0);
gg2 = 0; lw = 0;  ij = 0;

ny = 1; nylast = 1;
for(k=1,k<=nyy,k++, ny = ny+nylast; nylast = ny-nylast;
  nx = 1; nxlast = 1;
  for(j=1,j<=nxx,j++, nx = nx+nxlast; nxlast = nx-nxlast;
    for(i=1,i<=m,i++,
      xx = j(n,1,randomInteger(nx)-1);
      yy = j(n,1,randomInteger(ny)-1);
      {l1,l2,gg2,lw} = BestPartition(xx,yy);
      ij++;
      g2vec[ij] = gg2; nxvec[ij] = nx; nyvec[ij] = ny; lwvec[ij] = lw;
    );
  );
);
dt = newTable("PValLW5000B.JMP");
dt<<NewColumn("G2",values(g2vec));
dt<<NewColumn("NX",nominal,values(nxvec));
dt<<NewColumn("NR",nominal,values(nyvec));
```

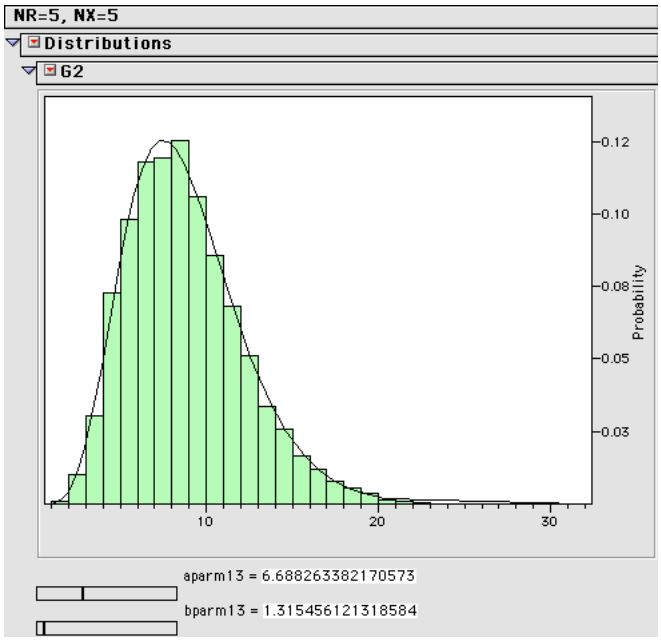
That produced a data set of 175,000 trials. We repeated it to make twice that many.

Next, we needed to fit the distributions and see if they were well-modeled by a gamma distribution. We made the script that inserted slider controls so that we could adjust the parameters of the gamma distribution if the moments didn't fit it well. No such adjustments seemed necessary after the analysis was completed.

The script to do that was this:

```
dt = currentDataTable();
subdt = dt << Summary(Group(:NR, :NX), Mean(:G2), Variance( :G2));
means = Column("Mean(G2)")<<GetAsMatrix;
varis = Column("Variance(G2)")<<GetAsMatrix;
currentDataTable(dt);
close(subdt,nosave);
bparmest = varis:/means;
aparmest = means:/bparmest;

obj=Distribution(Stack(1), By(NR,NX),Continuous Distribution(Column(:G2), Quantiles(0), Moments (0), Horizontal Layout(1), Pro
Axis(1), Outlier Box Plot(0)));

r = obj<<report;
ni = nitems(r);
for(i=1,i<=ni,i++,
   eval(substitute(
   expr(
     ri = r[i];
     aparmi=aparmiz;
     bparmi = bparmiz;
     ri[FrameBox(1)]<<FrameSize(400,300);
     ri[FrameBox(1)]<<AddGraphicsScript( YFunction(GammaDensity(x,aparmi,bparmi),x));
     ri[OutlineBox(2)]<<Append(hlistBox(SliderBox(0,20,aparmi,ri[FrameBox(1)]<<reshow),GlobalBox(aparmi)));
     ri[OutlineBox(2)]<<Append(hlistBox(SliderBox(0,20,bparmi,ri[FrameBox(1)]<<reshow),GlobalBox(bparmi)));
    ),
    expr(ri),   asname("RI"||char(i)),
    expr(aparmi),asname("aparm"||char(i)),
    expr(bparmi),asname("bparm"||char(i)),
    expr(aparmiz),aparmest[i],
    expr(bparmiz),bparmest[i]
   ))
);
```

The result was 35 fitted histograms, one of which was this. All seemed to fit well.



Now that we knew that the moments would work well, we needed to actually fit the moments as a function of nx and nr, the number of levels in X and the response. We decided to use a Neural Net to do that. Instead of using sample moments for every case, we used true moments for cases when nx was 2, since they were known. We also up-weighted the low values of nx and nr. Despite these efforts, it was not fitting the lower values of nx and nr very well. So we fit log(mean) and log(variance).

The moments from the Monte Carlo trials with 10000 trials were:

NR	NX	mean	variance
2	2	1.03289355	2.15924722
2	3	1.82395394	3.4180102
2	5	3.33801921	5.94733507
2	8	5.33457997	8.55618652
2	13	8.55837734	12.9848002
3	2	1.98059792	3.86704342
3	3	3.34015908	6.0211452
3	5	5.35655314	8.14535297
3	8	8.00099435	10.7760409
3	13	12.0182627	15.045414
5	2	4.05030297	8.16992553
5	3	5.93834842	9.83905519
5	5	8.80880264	11.5370166
5	8	12.2207316	14.5373536
5	13	17.1384582	18.4692651
8	2	7.05195556	14.5488177
8	3	9.76369454	15.6254868
8	5	13.2898751	16.8179467
8	8	17.5631112	19.1399617
8	13	23.3418944	22.0938512
13	2	12.1125457	24.2801327
13	3	15.8248539	23.2344823
13	5	20.400277	24.1604374
13	8	25.4798889	25.0668846
13	13	32.6047175	27.3823254
21	2	20.4672271	42.5596023
21	3	25.2189345	38.948083
21	5	31.2937348	35.9884442
21	8	37.7197797	35.0057153
21	13	46.1683797	35.477833
34	2	34.1543912	73.5629112
34	3	40.6774513	62.8563349
34	5	48.6162962	54.427073
34	8	56.6623212	50.3206364
34	13	66.8548606	48.8430669

The resulting Neural Net expressions to get the a and b gamma parameters were:

```
if (nx==2) { mean = nr-1; variance = 2*mean; }        // special case for knowing true value
else {
        mean = (0.2760966063147902
        - 2.755053381082162 *squash(1.240416754698371  - 0.6032431624857673 *nx - 0.05642514914674863*nr)
        - 4.154202088411108 *squash(0.5312751474270642 - 0.1278894468108674 *nx - 0.6932686600047646 *nr)
        + 0.7851580431604441*squash(3.713613173491597  - 0.8278633814248664 *nx - 0.1197972373660215 *nr)
        + 0.8582241159788044*squash(-2.394641262806448 + 0.08769613494498227*nx + 0.240506138757354  *nr)
        + 1.423112997009518 *squash(-2.073886078453307 + 0.0630763758049051 *nx + 0.05684361604628144*nr))
        *1.326133750719967 + 1.436992834494008;
        mean = exp(mean);

        variance = (0.05730119378771649
        + 1.399124821317178 *squash(-1.246134861818054 + 0.02289272257494033*nx + 0.06337779889683313*nr)
        - 4.460610791451469 *squash(1.506225885258226  - 0.5470508563196945 *nx - 0.7192919712743862 *nr)
        + 1.329504815693271 *squash(-1.769242311377324 + 0.1492245418164505 *nx + 0.2375652258150597 *nr)
        - 0.7931959767628654*squash(1.688445772013076  + 0.3333298625240388 *nx - 0.06735956562280759*nr))
        *1.129158910274815 + 1.964942161096689;
        variance = exp(variance);
}
b = variance/mean;
a = mean/b;
```

Modeling the Test Statistic's Distribution -- Ordered Case

The ordered case was modeled in the same way as the unordered case, though larger values of nx could be used, since the method was still fast for large nx values.

The moments obtained (with true values where known) were these:

NX	NR	mean	variance
2	2	1	2
2	3	2	4
2	5	4	8
2	8	7	14
2	13	12	24
2	21	20	40
2	34	33	66
3	2	1.57359354	2.9130551
3	3	2.87339593	4.70423324
3	5	5.33302483	8.99601166
3	8	8.75238298	14.208193
3	13	14.5629357	24.6588076
3	21	23.3376848	37.4538116
3	34	38.1484803	62.7331599
5	2	2.15500279	3.50803611
5	3	3.76628021	5.76375731
5	5	6.57075969	9.52191659
5	8	10.4981056	14.9452036
5	13	16.6050626	22.3016945
5	21	26.3103357	35.9523162
5	34	42.2335801	58.9875065
8	2	2.73432389	4.13761761
8	3	4.46961511	6.6502585
8	5	7.60023844	10.6976152
8	8	11.748788	15.5820331
8	13	18.1466878	23.0851328
8	21	28.5218456	35.9005705
8	34	45.2393916	56.0657493
13	2	3.16909381	4.61507274
13	3	5.10006242	6.66176084
13	5	8.40773031	10.9513776
13	8	12.8626616	16.8976341
13	13	19.8248452	25.0213733
13	21	30.5588324	35.5264484
13	34	47.8286513	50.8815308
21	2	3.63069131	5.21411358
21	3	5.65440708	7.46877359
21	5	9.01601598	11.472946
21	8	13.7621884	16.3398467
21	13	20.9249317	23.5088116
21	21	32.1198467	33.3200839
21	34	49.3518525	45.9455547
34	2	3.96767946	5.26728105
34	3	6.21132316	8.06768618
34	5	9.74976575	11.6930219
34	8	14.589039	16.3861649

34	13	22.0315512	22.4350861
34	21	33.2207545	31.0549042
34	34	50.2795559	48.3213961
55	2	4.36056224	5.97219395
55	3	6.62107453	8.23472973
55	5	10.4194459	12.0529338
55	8	15.3517718	15.4801469
55	13	22.7373549	22.3226465
55	21	33.8420219	30.2156008
55	34	51.3338428	48.2314163
89	2	4.61304146	5.93364097
89	3	7.01565744	8.16589692
89	5	10.8904836	11.6840368
89	8	15.8364564	15.8549206
89	13	23.3234238	22.4781522
89	21	34.5887655	31.3836077
89	34	51.9730612	46.7316526

The neural net expressions for the gamma parameters were this:

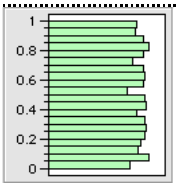
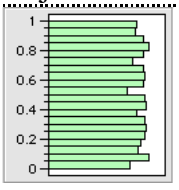
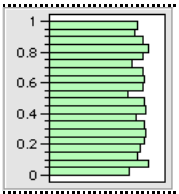
```
if (nx==2) { mean = nr-1; variance = 2*mean; }        // special case for knowing true value
else {
         mean=(1.301769477083182
         - 7.94692504665389 * squash(-0.0210123408666232 - 0.2518613156386489 *  nx - 0.06727041462914558 *  nr)
         - 4.077814018416775 * squash(0.2235513471395557 - 0.006010078927027169 *  nx - 0.6982453780867636 *  nr)
         + 2.191896597732866 * squash(2.35102691594594 - 0.3569390844668006 *  nx - 0.1245875970073678 *  nr)
         + 0.928447356341446 * squash(-2.223583104899824 + 0.003107036751429028 *  nx + 0.2500806295975323 *  nr)
         - 1.218519901635325 * squash(1.951502740399493 - 0.001311840317569508 *  nx - 0.08166483883705047 *  nr))
         * 1.138081563249612 + 1.786178322906917;
         mean = exp(mean);

         variance = (-0.3831532845610621
         - 1.469765640428693 * squash(2.637404535262809 - 0.00201549213395159 *  nx - 0.09140700525110769 *  nr)
         + 3.564959674389817 * squash(-0.1886542988420415 + 0.0023325631538162 *  nx + 0.6366501165090098 *  nr)
         + 1.32083818385447 * squash(-3.227088025889567 + 0.1680736675939787 *  nx + 0.1473719725610988 *  nr)
         - 0.6125564910317741 * squash(2.991316133978449 - 0.0001063246797539832 *  nx - 0.3154312734939435 *  nr)
         - 5.53975146770328 * squash(-0.104495544378832 - 0.3302280397694065 *  nx - 0.26616402089296 *  nr)
         - 2.296828210396418 * squash(-1.027867872780368 + 0.115714146444512 *  nx + 0.06136431966040862 *  nr))
         * 0.9654108489785114 + 2.222724833424783;
         variance = exp(variance);
}
b = variance/mean;
a = mean/b;
```

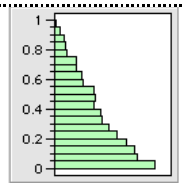
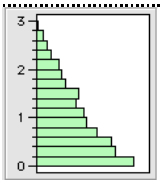
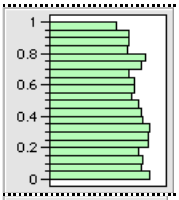
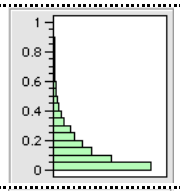
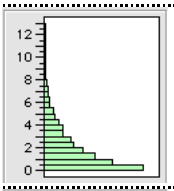
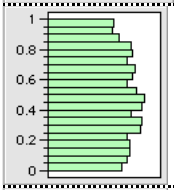
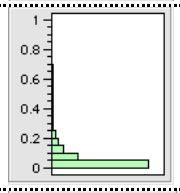
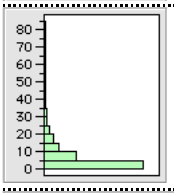
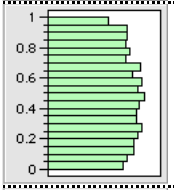
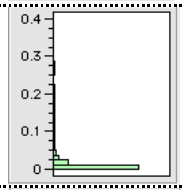
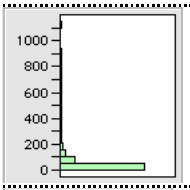
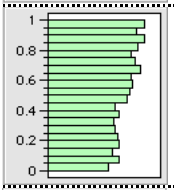
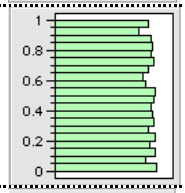
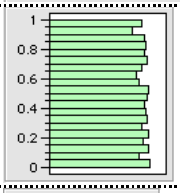
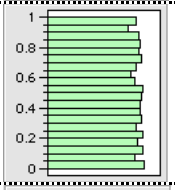
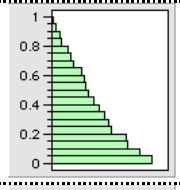
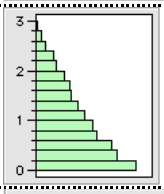
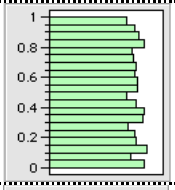
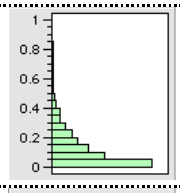
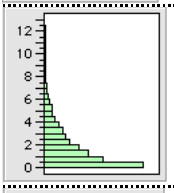
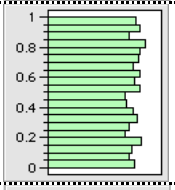
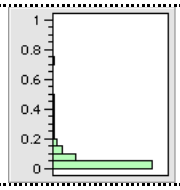
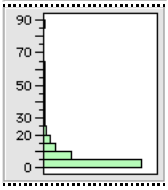
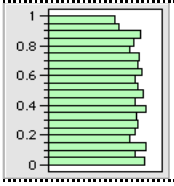
Evaluation of Resulting p-values

We reran the simulations calculating pvalues corresponding to the fitted gamma distributions. For the unordered case, this produced the following histograms, shown in two tables.

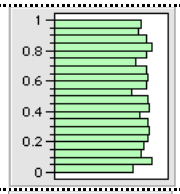
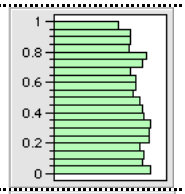
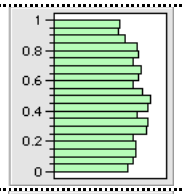
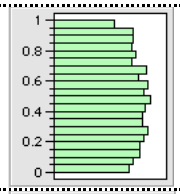
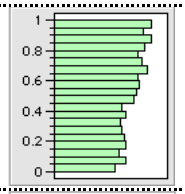
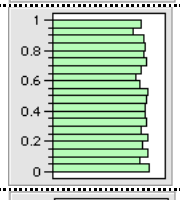
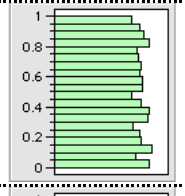
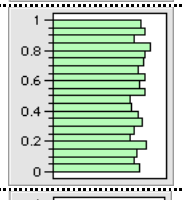
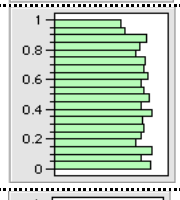
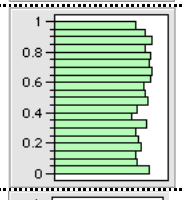
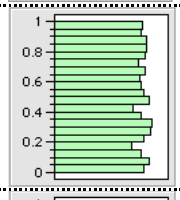
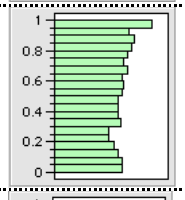
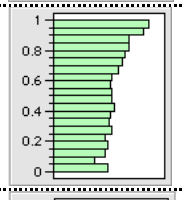
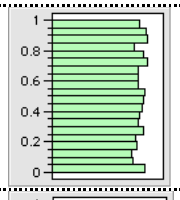
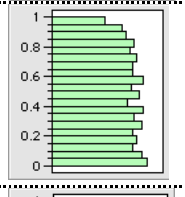
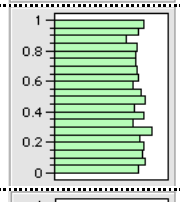
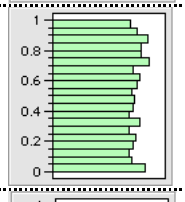
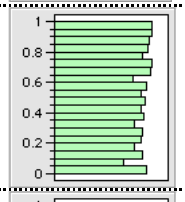
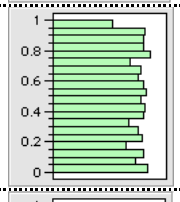
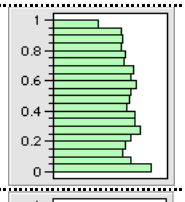
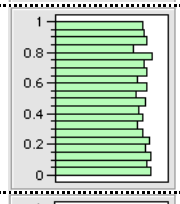
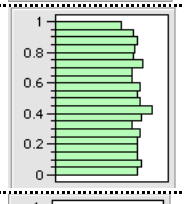
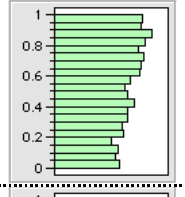
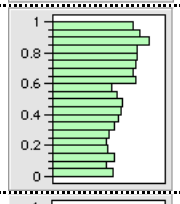
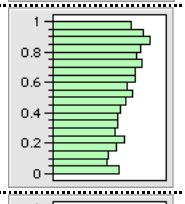
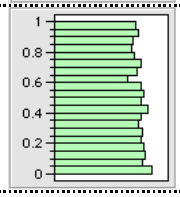
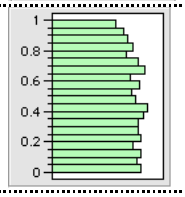
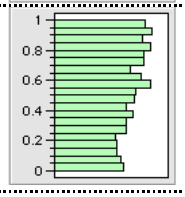
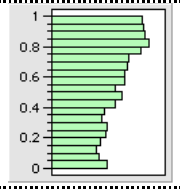
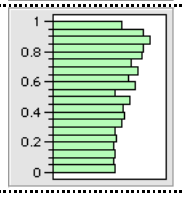
In the first table, we compare the fitted p-values with the unadjusted and Bonferroni-adjusted pvalues that are used traditionally. Only the nx=2 distributions actually are Uniform(0,1). The Bonferroni values are even on different scales, because many values far exceed 1 in value.

In the second table we show the fitted p-values distributions across all the nx and nr values in the Fibonacci grid.

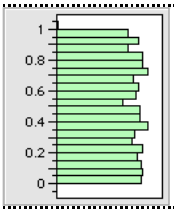
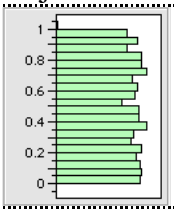
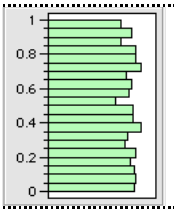
	Unadjusted P-Value	Bonferroni Adjusted	Fitted P-Value
NR=2, NX=2			

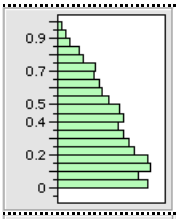
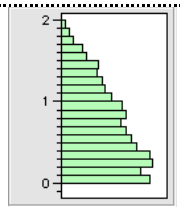
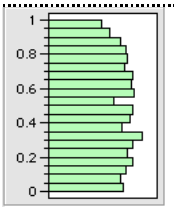
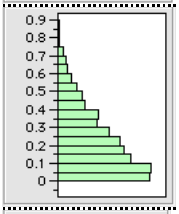
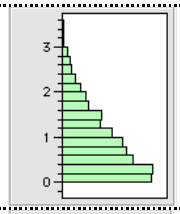
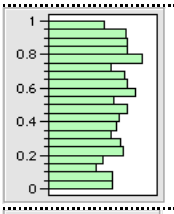
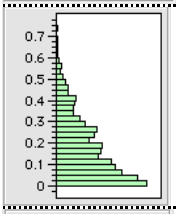
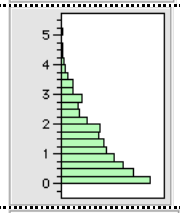
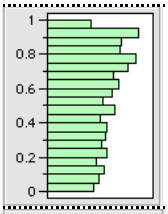
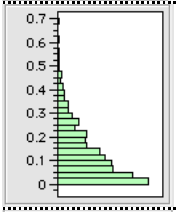
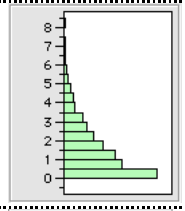
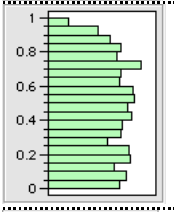
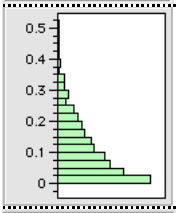
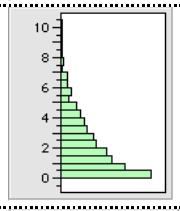
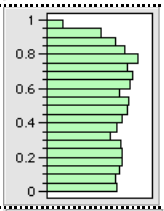
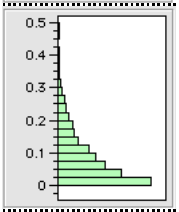
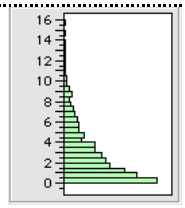
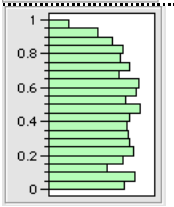
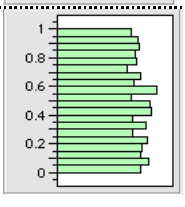
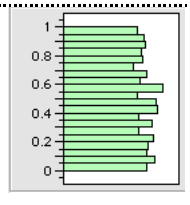
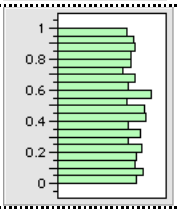
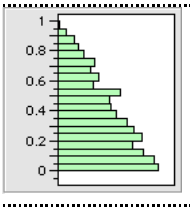
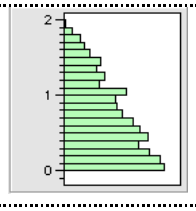
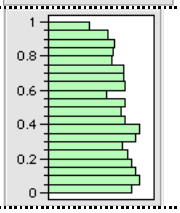
NR=2, NX=3			
NR=2, NX=5			
NR=2, NX=8			
NR=2, NX=13			
NR=3, NX=2			
NR=3, NX=3			
NR=3, NX=5			
NR=3, NX=8			

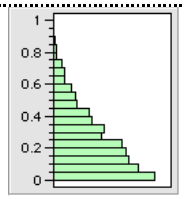
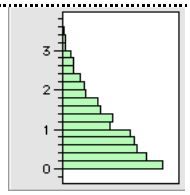
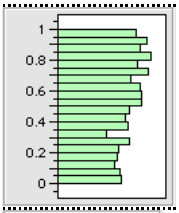
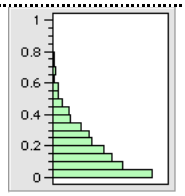
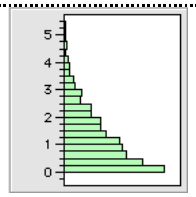
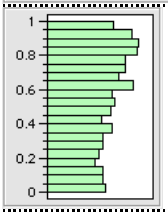
	nx=2	nx=3	nx=5	nx=8	nx=13

nr=2					
nr=3					
nr=5					
nr=8					
nr=13					
nr=21					

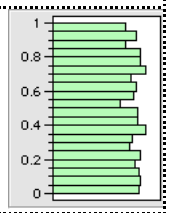
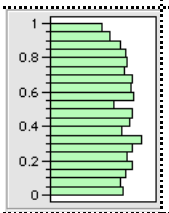
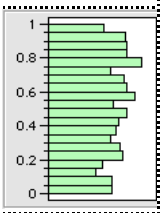
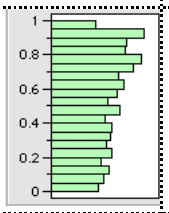
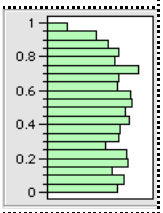
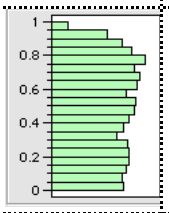
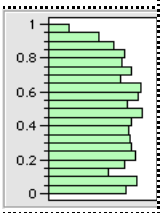
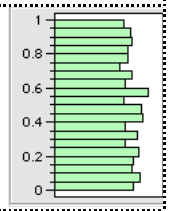
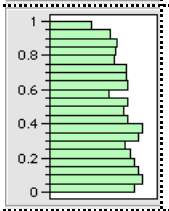
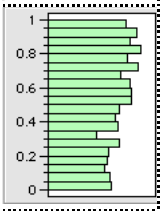
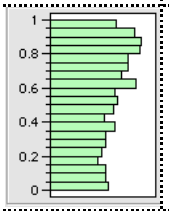
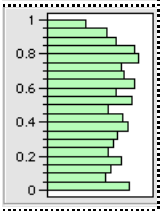
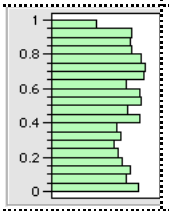
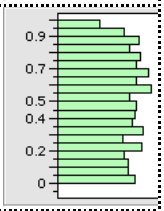
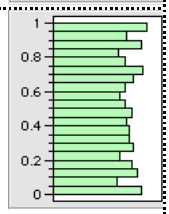
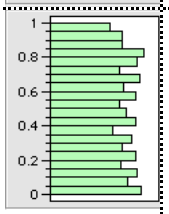
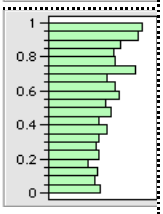
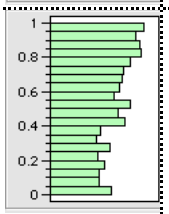
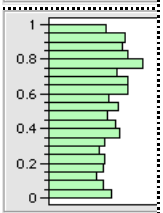
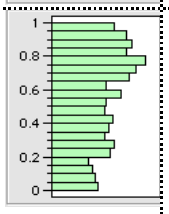
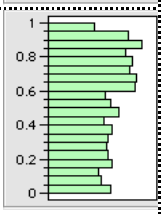
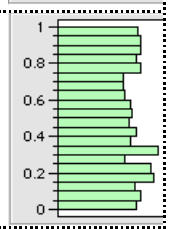
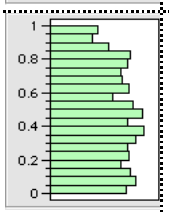
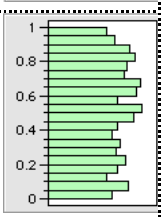
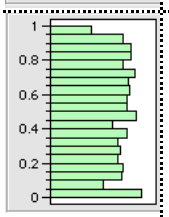
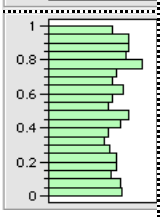
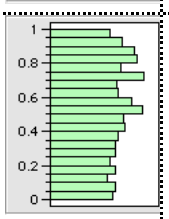
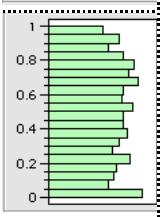
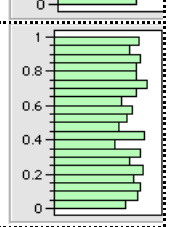
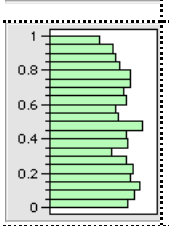
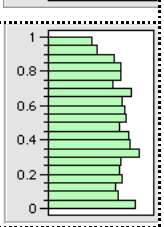
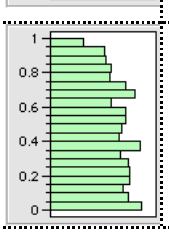
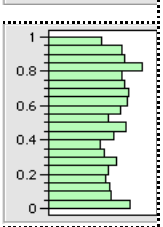
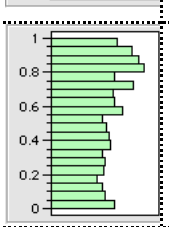
For the ordered case, first compare the fitted p-values with the unadjusted and the Bonferroni-adjusted p-values.

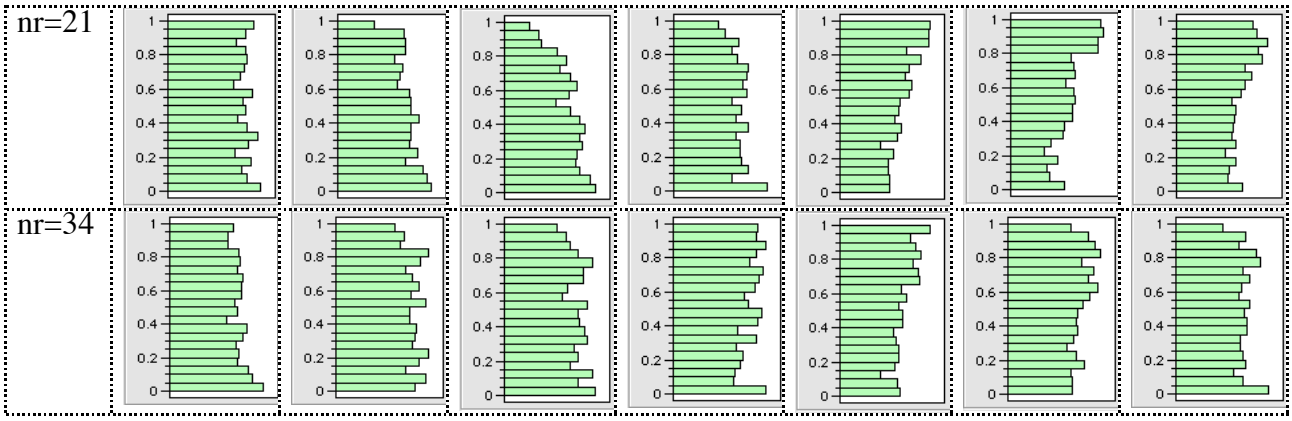
	Unadjusted P-Value	Bonferroni Adjusted	Fitted P-Value
NR=2, NX=2			

NR=2, NX=3			
NR=2, NX=5			
NR=2, NX=8			
NR=2, NX=13			
NR=2, NX=21			
NR=2, NX=34			
NR=3, NX=2			
NR=3, NX=3			

NR=3, NX=5			
NR=3, NX=8			

Then look at a table of all the fitted p-value distributions in the Fibonacci grid. The distributions in the first column are true uniforms--the others fitted from moments. Some of the distributions look slightly less uniform than the first column distributions, but in general the fit is reasonable, much more reasonable than the unadjusted p-values and Bonferroni p-values of tradition.

	nx=2	nx=3	nx=5	nx=8	nx=13	nx=21	nx=34
nr=2							
nr=3							
nr=5							
nr=8							
nr=13							

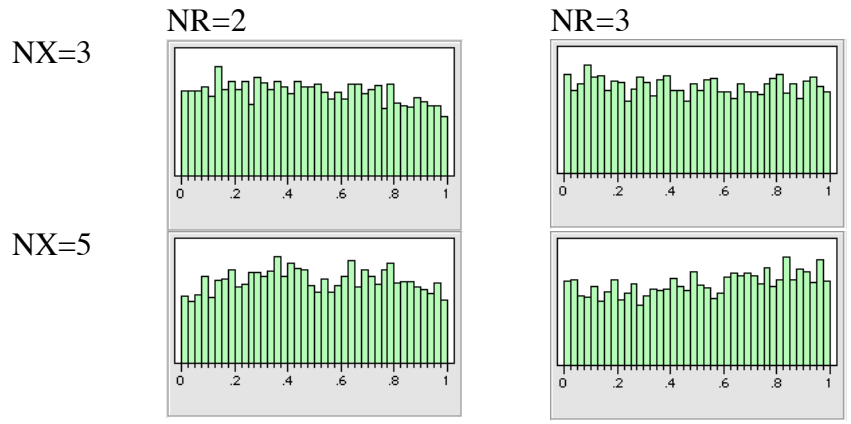


Imbalance of X-level frequencies

All the simulations so far have used balanced classifications, with roughly an equal number of occurrences for each level. To check the pvalues for unbalanced distributions, we made the X with half the cases on the last level, with the other levels equal to each other. This was done by modifying one line in the simulation script:

```
xx = j(n,1,min(randomInteger(nx*2),nx)-1);
```

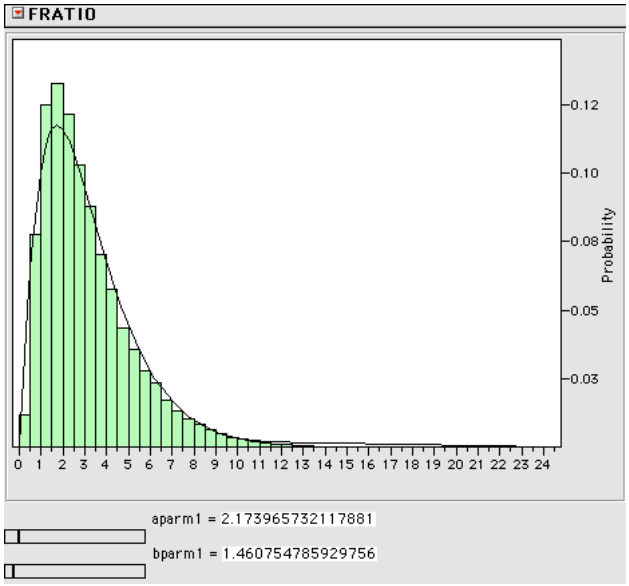
The resulting p-value distributions looked reasonable.



These results suggested that an adjustment for imbalance was not needed, though we didn't test any more stressful situations. If the test statistic distribution did on these X-level rates, then we would try using functions of the unbalance, starting with the marginal entropy-equivalent balanced number of levels, which is NXE=-exp(Sum(p[i]*log(p[i])))

Continuous Responses with Categorical X's

The preliminary study on the distribution of continuous responses turned up distributions that the gamma distribution did not fit very well. The fit below was for the F statistic for a continuous normal response with an ordered X with 13 levels. This was among the worst fitting among the distributions examined.



However, we decided to go ahead with gamma distributions anyway and see how the p-value distribution looked in the NULL case.

The moments of F statistics over 5000 Monte Carlo samples for *unordered-X* samples were this:

NX	F Mean	F Variance
2	1.0081888	2.0492372
3	1.81826093	3.33011866
5	3.2770996	5.53976317
8	5.23020957	8.11351644
13	8.57822411	13.1827308
21	13.458647	19.8100357

We used a neural net to fit the log moments, with the resulting expressions for the gamma parameters:

```
logMean = (0.3247401514044082
- 3.377924250520082 * squash(0.6928893526465474 - 0.5108055293941157 *  nx)
+ 1.404646616649575 * squash((-1.548447235304804) + 0.1424129088976174 *  nx)
- 0.8111344305114346 * squash(0.9552247732829481 - 0.102707945360597 *  nx))
* 0.9679128513133213 + 1.366049129723897;
mean = exp(logMean );

logVariance = (0.09999563736950218
+ 1.499493412598702 * squash((-2.501182784952456) + 0.2120655661008282 *  nx)
- 0.2982074540555843 * squash(0.3707045401415789 - 0.04351974257350025 *  nx)
- 3.335570427975299 * squash(0.7003618842539239 - 0.5107079416841145 *  nx))
* 0.8480260134033467 + 1.881842508423816;
variance = exp(logVariance );
b = variance/mean;
a = mean/b;
```

The moments of F statistics over 5000 Monte Carlo samples for *ordered-X* samples were this:

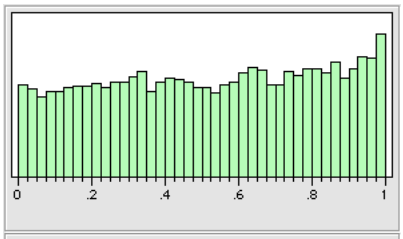
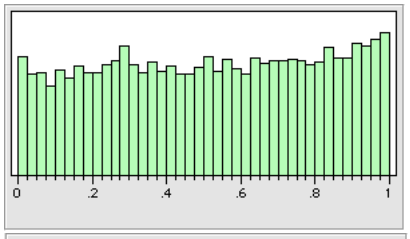
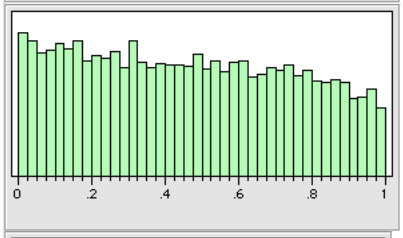
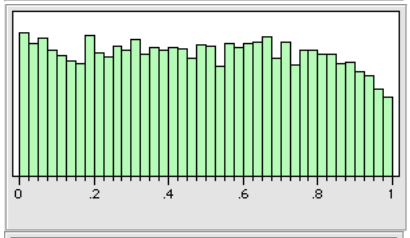
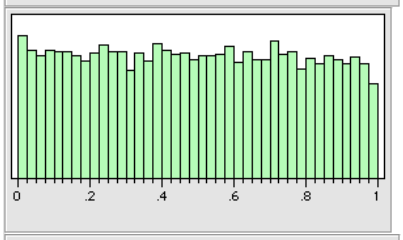
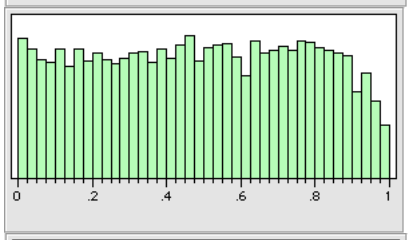
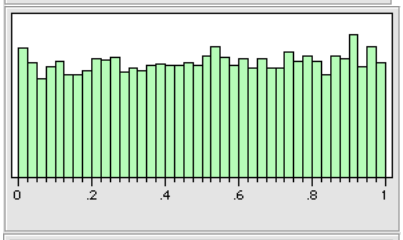
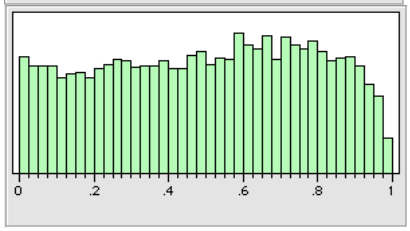
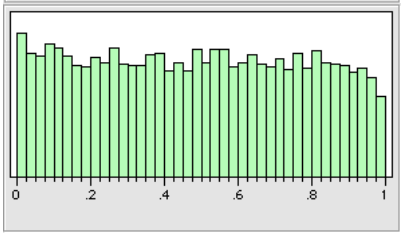
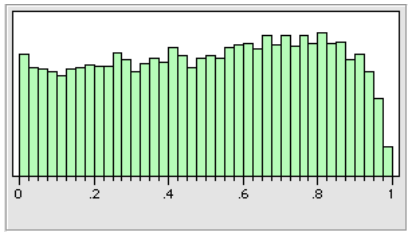
NX	F Mean	F Variance
2	0.97898017	1.94658557
3	1.53383408	2.61049221
5	2.16492664	3.52630601
8	2.68634194	4.25617303

13	3.21531017	4.85654314
21	3.57907503	5.27199196

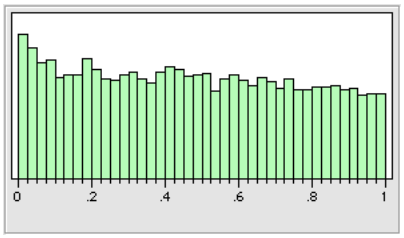
The neural net fit for the gamma parameters (not using logs) for unordered was this:

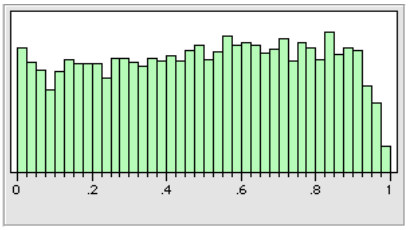
```
mean = (-0.1057020494568139
- 3.414250949476012 * squash(0.7230383201277949 - 0.4643842977130933 * nx)
+ 1.532971661689726 * squash(-1.692665100927281 + 0.170743298371788 * nx))
* 0.9951901912449169 + 2.359744670193956;

variance = (0.005509654653496981
+ 1.419586807289863 * squash(-1.331713402887075 + 0.1372340412035706 * nx)
- 3.445110164564215 * squash(0.903536453165965 - 0.4574306647384537 * nx))
* 1.295975255042254 + 3.744681986456623;
b = variance/mean;
a = mean/b;
```

Fitted P-values were calculated for 10000 samples resulting the following p-value distributions:

X Levels	UnOrdered X Levels	Ordered X Levels
NX=2		
NX=3		
NX=5		
NX=8		
NX=13		

NX=21





We decided that these p-value distributions looked acceptable, certainly much better than the unadjusted F's or the Bonferroni-adjusted F's that are commonly used. We concede that the actual distribution does not look like a gamma (except for NX=2), but the p-values look reasonable despite that. We hope to get better advice concerning the tail the distribution.

Continuous Responses with Continuous X's

This is the limiting distribution for the previous ordered case, as NX goes to N.

Discussion

The goal here is to make fair comparisons to choose partition splits among X variables where the number of levels and the type of X (unordered categories, ordered categories, and continuous) is different among the terms. We need to compare the taste of apples and oranges of different sizes.

It is clear that the using raw statistics or unadjusted p-values is not very fair, since it unfairly chooses factors with many levels. It is also clear that using Bonferroni-adjusted p-values is not fair, since it prefers terms with few levels, especially for unordered categories.

The Monte-Carlo derived p-value formulas certainly produce better null-case p-value distributions than the unadjusted or Bonferroni-adjusted methods. Of course, since we are calibrating the tests so that it produces fair tests under the null case, where there is no relationship. This does not guarantee us good behavior when there really are relationships, however, significance testing is what we know how to do.

Ultimately, we want to be able to use these p-values, with further selection-bias adjustments, to make judgments on how much to grow the tree. Making further adjustments would be a good subject for further, more complicated, experiments.