Journal of Statistical Software

December 2006, Volume 17, Issue 7. *http://www.jstatsoft.org/*

Exact Hypothesis Tests for Log-linear Models with exactLoglinTest

Brian Caffo
Johns Hopkins Bloomberg School of Public Health

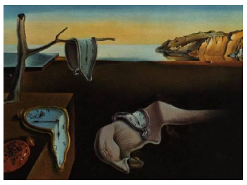
Abstract

This manuscript overviews exact testing of goodness of fit for log-linear models using the R package **exactLoglinTest**. This package evaluates model fit for Poisson log-linear models by conditioning on minimal sufficient statistics to remove nuisance parameters. A Monte Carlo algorithm is proposed to estimate P values from the resulting conditional distribution. In particular, this package implements a sequentially rounded normal approximation and importance sampling to approximate probabilities from the conditional distribution. Usually, this results in a high percentage of valid samples. However, in instances where this is not the case, a Metropolis Hastings algorithm can be implemented that makes more localized jumps within the reference set. The manuscript details how some conditional tests for binomial logit models can also be viewed as conditional Poisson log-linear models and hence can be performed via **exactLoglinTest**. A diverse battery of examples is considered to highlight use, features and extensions of the software. Notably, potential extensions to evaluating disclosure risk are also considered.

Keywords: Poisson models, exact testing, hypothesis testing, Fisher's exact test.

Goodness-of-fit tests for Poisson log-linear models are often performed by conditioning on the sufficient statistics for model parameters and then calculating P values from the resulting, parameter free, distribution. Such P values result in so called "exact tests" that guarantee that the actual Type I error rate is no larger than the nominal one. Such analyses are limited by the fact that the set of lattice points contained in the conditional distribution is difficult to manage, computationally or analytically. Hence, large sample chi-squared tests are generally more popular.

The R (R Development Core Team 2006a) software package **exactLoglinTest** attempts to address the computational difficulties of exact analyses using importance sampling and Markov chain Monte Carlo (MCMC). At the heart of **exactLoglinTest** is a sequentially rounded approximation to the conditional distribution developed and implemented in Booth and Butler (1999) and Caffo and Booth (2001).



Residence	Residence in 1985			
in 1980	Northeast	Midwest	South	West
Northeast	11,607	100	366	124
Midwest	87	13,677	515	302
South	172	225	17,819	270
West	63	176	286	10,192

Table 1: Residency data. Source Agresti (1990) page 423.

The manuscript is laid out as follows. In Section 1 notation and background information is presented. Section 2 gives the basics of the algorithms while Section 3 covers several example usages. Finally, Section 4 concludes with a discussion.

1. Problem formulation

Consider the data given in Table 1, which is a 4×4 contingency table compiled by the United States Census Bureau tabulating subjects' residence in 1985 by their residence in 1980. This data set will be used as motivation for the approach.

Suppose that observed counts, $y = (y_1, \ldots, y_n)$, are modeled as Poisson with means, $\mu = (\mu_1, \ldots, \mu_n)$, satisfying

$$\log \mu = x\beta.$$

Here "log" is presumed to act component-wise on vectors and the design matrix, x, is assumed to be full rank. For example, for the Residency data, we might consider the familiar model of independence. The columns of x would contain a constant (intercept term), three columns of indicators for the 1980 residence and three columns of indicators for the 1985 residence.

Consider investigating model fit using goodness-of-fit statistics. Specifically, let h be a test statistic of interest such that larger values of h support the alternative hypothesis, such as the Pearson Chi-Squared statistic

$$\sum_{i=1}^{n} \frac{(y_i - \hat{\mu}_i)^2}{\hat{\mu}_i}$$

or the residual deviance

$$2\sum_{i=1}^{n}\{y_i \log(y_i/\hat{\mu}_i) - (y_i - \hat{\mu}_i)\}.$$

Here the term $y_i \log(y_i/\hat{\mu}_i)$ is defined to be 0 when $y_i = 0$ and $\hat{\mu}_i$ are maximum likelihood estimates. For the Residency data set these statistics are both on the order $120,000$ on 9 degrees of freedom. The independence model results in such a poor fit because of the large counts down the diagonal; that is, people tend to stay in the same geographic region. In Section 3.1 a better fitting model for this data is considered that incorporates model parameters for each diagonal count.

For Poisson log-linear models, it is well known that the sufficient statistic for β under the null hypothesis is $x^\top y$, where a superscript $\top$ denotes a transpose. The existence of closed form sufficient statistics for the Poisson log-linear model yields a method for testing goodness of fit or eliminating nuisance parameters. In particular, because $x^\top y$ is sufficient for β, the conditional distribution of y given $x^\top y$ does not depend on any parameters. Furthermore, it can be shown that

$$\mathsf{P}(Y = y \mid x^\top y = s) \quad = \quad \left(\sum_{u \in \Gamma(s)} \prod_{i=1}^{n} \frac{1}{u_i!} \right)^{-1} \left(\prod_{i=1}^{n} \frac{1}{y_i!} \right) \quad = \quad C \prod_{i=1}^{n} \frac{1}{y_i!}, \tag{1}$$

where C is a normalizing constant, $u = (u_1, \ldots, u_n)^\top$ is a generic vector of non-negative counts and $\Gamma(s) = \{u \mid x^\top u = s\}$. Here $\Gamma(s)$ is the so called "reference set", or the set of allowable values of y given that $x^\top y = s$.

To illustrate, consider the independence model for the Residency data. The sufficient statistics, $x^\top y$, can be shown to be the margins of the table. Hence $\Gamma(s)$ is the collection of all tables that satisfy the margins. In this case, the reference set is easily characterized. However, in general, $\Gamma(s)$ is either too complicated or contains too many elements to be characterized in any useful way.

A conditional P value calculates the probability of all tables with values of h at least as large as the observed value. Notationally, the conditional P value is:

$$\mathsf{P}\{h(y) \geq h(y_{obs}) \mid x^\top y = x^\top y_{obs}\} \quad = \sum_{\{y \in \Gamma(s_{obs})\}} \frac{I\{h(y) \geq h(y_{obs})\}}{C \prod_{i=1}^{n} y_i!}$$

where y_{obs} is the observed table and $s_{obs} = x^\top y_{obs}$.

When compared to a fixed nominal type I error rate, such a P value is "exact". It should be noted that the accolade "exact" is given to tests that guarantee the nominal type I error rate unconditionally. Thus a test that never rejects the null hypothesis is technically exact in any situation. In particular, this illustrates that exactness may be a desirable, but is not a sufficient property for a test to be acceptable. Moreover, this example (never rejecting) is particularly relevant in our setting because $\Gamma(s_{obs})$ may contain one or few elements. Hence the conditional P value will be exactly or near one regardless of the evidence in the data vis-a-vis the two hypotheses. However, it is also the case that the conservative conditional tests can produce P values that are smaller than those calculated via Chi-squared approximations (see Subsection 3.2 for an example).

A general problem with exact P values is their calculation. The reference set, $\Gamma(s_{obs})$, is often too large or complicated to enumerate. In this manuscript the use of Monte Carlo to approximate conditional P values is explored. In specific, consider simulating J complete tables, say $\{y^{(j)}\}_{j=1}^{J}$. Then a Monte Carlo approximation to the conditional P value is

$$\sum_{j=1}^{J} w^{(j)} I\{h(y^{(j)}) \geq h(y_{obs})\} / \sum_{j=1}^{J} w^{(j)},$$

where $\{w^{(j)}\}$ are importance weights, the ratio of the target mass function (given by Equation 1) evaluated at the simulated table to that of the mass function used for simulation. Note that when h is the Pearson statistic or the deviance, the fitted values, $\hat{\mu}$, do not change by iteration, because every simulated data set has the same sufficient statistics.

1.1. Binomial calculations

Conditional inference for binomial-logit models is a special case of conditional inference for Poisson log-linear models. Therefore, the testing method outlined can also be applied to binomial-logit model, provided some care in model specification.

Consider a binomial logit model of the form, $b_i \sim \text{Bin}(n_i, p_i)$ for $i = 1, \ldots, k$ and

$$\text{logit}(p_i) \quad = \quad z_i\gamma + x_i^\top \beta, \tag{2}$$

where γ is a scalar of interest and β is a p dimensional vector of nuisance parameters. Frequently, $x_i^\top$ contains only a stratum indicator and an intercept term. Conditioning on the sufficient statistic for β results in standard conditional logistic regression. For this purpose, we suggest the `coxph` function as described in Venables and Ripley (2002). Instead we consider the more general case where x_i is arbitrary. However, we stipulate that conditional inference in such settings is not always informative, as the loss of information from conditioning can sometimes be quite severe. For example, in many cases the reference set might contain only the observed data.

Consider testing $H_0 : \gamma = 0$ versus $H_a : \gamma < 0$ and the following model for the success $(y_{i1} = b_i)$ and failure $(y_{i2} = n_i - b_i)$ counts:

$$y_{ij} \sim \text{Poisson}(\mu_{ij}) \qquad \log(\mu_{i1}) = \alpha_i + x_i^\top \beta \qquad \log(\mu_{i2}) = \alpha_i, \tag{3}$$

for $j = 1, 2$ and $i = 1, \ldots, k$. The sufficient statistics for the α_i are $y_{i1} + y_{i2} = n_i$. Then it can be shown that the conditional distribution of $y_{i1}|y_{i+}$ is precisely the model given by Equation 2 where $p_i = \mu_{i1}/\mu_{i+}$.

Therefore, conditioning out the nuisance parameters $\{\alpha_i\}$ and β for the Poisson log-linear model yields exactly the same (null) conditional distribution as conditioning out β in Model 2. Furthermore, this exercise indicates exactly how to perform the calculations, which is pertinent, since **exactLoglinTest** only accepts models in the form of Poisson log-linear models. It is also possible to represent many multinomial and product multinomial data as instances of conditional Poisson models. We refer to (Agresti 1990, Chapter 8 and Section 8.6.7 in particular) for more details.

2. The software

The software **exactLoglinTest** is an implementation of the algorithms presented in Booth and Butler (1999) and Caffo and Booth (2001) using the R open-source programming language (R Development Core Team 2006a). At the heart of both algorithms is a sequentially generated rounded normal approximation to the conditional distribution. Full descriptions of the algorithms are given in the technical references above while below we give a brief overview and summarize related methods.

Both algorithms use the normal approximation to the Poisson as their base. That is, for large μ, y will be approximately Normal$\{\mu, D(\mu)\}$, where $D(\mu)$ is a diagonal matrix with diagonal μ. Using the properties of the multivariate-normal distribution, an approximation for the distribution of y given $x^\top y$ can be found. This process fixes some of the cells of y and leaves some to be simulated. For example, in the independence model, $x^\top y$ is the margins of the table. Subtracting all rows except the last from the row margins yields the last row. The choice of which cells to fix and which to simulate from is somewhat arbitrary. The software does an iterative search to find appropriate cells, though the choice is not optimized.

The algorithm in Booth and Butler (1999) employed the sequential distributions from the normal approximation to the distribution of y given $x^\top y$ as a candidate distribution for importance sampling. A novel sequential rounding scheme was used to force the simulated data to be non-negative integers.

The normal approximation allows for negative cell entries, resulting in simulated tables that must be discarded and a potential loss in algorithmic efficiency. A refinement of the algorithm fixes not only $x^\top y$, but also some of the remaining free cells. Because the normal approximation is lower dimensional, a greater percentage of valid tables can be generated this way (as developed in Caffo and Booth 2001). However, this modification generates a Markov chain instead of iid samples and hence requires more thoughtful use of the algorithm. As such, one should switch to Markov chain sampling only when importance sampling produces an extremely low percentage of valid tables.

Both the importance sampling and MCMC methods are implemented in a function, `mcexact`. This function takes a log-linear model specification in the same form as a a call to `glm`. The function `mcexact` uses `glm` to construct a design matrix and obtain fitted means ($\hat{\mu}$). The algorithm proceeds to calculate goodness-of-fit statistics and subsequently runs the Monte Carlo algorithm. The returned object is a list of class “`bab`” or “`cab`” (acronyms for “Booth And Butler” and “Caffo And Booth” respectively) depending on which of the two algorithms was used. The object has available methods `summary`, `print` and `update`. Also included in the package are utility functions for more direct interaction with the simulated tables.

Below we discuss some of the several competing algorithms to the rounded normal approximations used in Booth and Butler (1999) and Caffo and Booth (2001). However, we note that direct software competitors to **exactLoglinTest** algorithms are few. From this perspective, the best developed are the network algorithms (see Mehta and Patel 1980, 1983) and related methods incorporated in the software **StatXact** (Mehta 1991). This proprietary software suite is considered the industry standard in this area. Being open source and freely available, **exactLoglinTest** is targeting a different user base. In addition, we note that **exactLoglinTest** differs from the **StatXact** software suite by focusing only on log-linear models via Monte Carlo calculations.

Of the many competing algorithms, perhaps the most general is due to Diaconis and Sturmfels (1998), who used computational algebra to provide simple random walk algorithms based on Markov bases. A limitation of this approach is that, in some settings, the computational algebra required to obtain the Markov bases is impractical. However, Dobra (2003) calculated the Markov bases for a large class of graphical models. Related algorithms using elegant random scan Gibbs samplers were given by Forster, McDonald, and Smith (1996); McDonald, Smith, and Forster (1999); Smith, Forster, and McDonald (1996). Furthermore, relevant recent developments in sequential importance sampling (Chen, Diaconis, Holmes, and Liu 2005; Chen, Dinwoodie, and Sullivant 2006) are applicable to this setting. We refer the reader to Caffo and Booth (2003) for an overview of Monte Carlo algorithms in this area.

In addition to Monte Carlo algorithms, there has been much relevant research in the area of fast exact enumeration calculations for specific models (such as in Patefield 1981; Booth, Capanu, and Heigenhauser 2005; Hirji, Mehta, and Patel 1987). Also, highly accurate saddlepoint approximations have been applied with success (Strawderman and Wells 1998).

As this brief literature review suggests, there are numerous competing algorithms. However, most users outside of this research area would have little interest in implementing any of the algorithms from scratch. The software package **exactLoglinTest** is an attempt to bridge this gap being user-friendly, open source software for performing Monte Carlo exact goodness-of-fit tests.

3. Examples

A copy of the package can be obtained at the Comprehensive R Archive Network, `http://CRAN.R-project.org/`. Refer to the "R Installation and Administration" manual (R Development Core Team 2006b), included with your R distribution, for details on how to install packages. In addition, a noweb version of this document can be found at the author's web site.

Assuming it is properly installed, one can load **exactLoglinTest** with

```
R> library("exactLoglinTest")
R> set.seed(1)
```

The `set.seed(1)` command is used to set the random number seed to a specific value, so that results can be reproduced.

3.1. Residency data

Recall the Residency data (Table 1). The data can be obtained by the command

```
R> data("residence.dat")
```

Clearly a model of independence does not hold, as evidenced by the large diagonal counts. Instead we focus on the so called quasi-symmetry model (see Agresti 1990). This is a useful alternative model when testing marginal homogeneity. The extra term, `sym.pair`, in the data frame is used to fit a quasi-symmetry model. A Monte Carlo goodness-of-fit test of quasi-symmetry versus a saturated model involves the following command

```
R> resid.mcx <- mcexact(y ~ res.1985 + res.1980 + factor(sym.pair),
+     data = residence.dat, nosim = 10^2, maxiter = 10^4)
R> resid.mcx

                 deviance    Pearson
observed.stat 2.98596233 2.98198696
pvalue        0.46311695 0.46311695
mcse          0.03679595 0.03679595
```

The default method used for sampling is the importance sampling algorithm of Booth and Butler (1999). Because this method rejects simulated table with negative entries, the number of desired simulations `nosim` may not be met in `maxiter` iterations.

The returned object is a list of class "`bab`", storing the results as well as all of the relevant information necessary to restart the simulation. More information can be obtained with `summary`

```
R> summary(resid.mcx)

Number of iterations       =  100
t degrees of freedom       =  3
Number of counts           =  16
df                         =  3
Next update has nosim      =  100
Next update has maxiter    =  10000
Proportion of valid tables =  1

                 deviance    Pearson
observed.stat 2.98596233 2.98198696
pvalue        0.46311695 0.46311695
mcse          0.03679595 0.03679595
```

The "t degrees of freedom" refers to degrees of freedom used as a tuning parameter within the algorithm, while the `df` refers to the model degrees of freedom.

As it stands, the Monte Carlo standard error, `mcse`, is too large for the P value estimate to be useful. The simulation can be restarted using `update`

```
R> resid.mcx <- update(resid.mcx, nosim = 10^4, maxiter = 10^6)
R> resid.mcx

                  deviance     Pearson
observed.stat 2.985962330 2.981986964
pvalue        0.397222805 0.396772921
mcse          0.003596126 0.003594809
```

Here `nosim` is the number of additional simulations desired and `maxiter` is the maximum number of iterations allowed. It is important to note that `update` can only resume the simulation with a new Monte Carlo sample size. It does not allow users to change the model formulation; one must rerun `mcexact` independently to do that. In practice, we recommend that users employ a large number of simulations, updating results until twice the Monte Carlo standard error is below the number of significant digits required for reporting the P value.

This example illustrates the point that the underlying algorithms are very efficient when the cell counts are large. When this is the case, the large sample approximations are close to the conditional results, as we can see using the observed deviance and Pearson statistics given in the output above:

```
R> pchisq(c(2.986, 2.982), 3, lower.tail = FALSE)

[1] 0.3937887 0.3944088
```

3.2. Pathologists' tumor ratings

The following example is interesting in that the large sample results differ drastically from the conditional results. Moreover, the conditional results are *less* conservative. The data given in Table 2 cross classify two pathologists' tumor ratings and can be obtained in R with the command

```
R> data("pathologist.dat")
```

A uniform association model accounts for the ordinal nature of the ratings by assigning ordinal scores to the ratings (see Agresti 1990). We test the uniform association model against the saturated model with

```
R> path.mcx <- mcexact(y ~ factor(A) + factor(B) + I(A * B),
+     data = pathologist.dat, nosim = 10^4, maxiter = 10^4)
R> summary(path.mcx)
```

```
Number of iterations       =  4444
t degrees of freedom       =  3
Number of counts           =  25
df                         =  15
Next update has nosim      =  10000
Next update has maxiter    =  10000
Proportion of valid tables =  0.4444

                  deviance      Pearson
observed.stat 16.214534925 14.729278917
pvalue         0.037393837  0.126297722
mcse           0.001194588  0.002990041
```

It is worth comparing these results to the asymptotic Chi-squared results

```
R> pchisq(c(16.214, 14.729), 15, lower.tail = FALSE)
```

```
[1] 0.3679734 0.4711083
```

	Pathologist B				
Pathologist A	1	2	3	4	5
1	22	2	2	0	0
2	5	7	14	0	0
3	0	2	36	0	0
4	0	1	14	7	0
5	0	0	3	0	3

Table 2: Pathologist agreement data. Source Agresti (1990).

3.3. Alligator food choice data using MCMC

This example illustrates the algorithm from Caffo and Booth (2001) using the data and Poisson log-linear model from the alligator food choice data shown in Table 3. This data set and model is a good choice for MCMC as the percent of valid tables generated using `method = "bab"` is very small, less than 1% of the tables simulated. It is often the case that the MCMC algorithm will be preferable when the table is large and/or sparse. Using MCMC introduces further complications in reliably running and using the output of the algorithm. Throughout this example we consider the log-linear model

$$(FG, FL, FS, LGS),$$

where F = food choice, L = lake, S = size and G = gender.

The algorithm from Caffo and Booth (2001) uses local moves to reduce the number of tables with negative entries that the chain produces. This method can be invoked with the option `method = "cab"` of `mcexact`. The parameter `p` of `mcexact` is the average proportion of table entries left fixed. A chain with `p=.9` will leave most of the table entries fixed from one iteration to the next. A high value of `p` will often result in a high proportion of valid (non-negative) simulated tables. Unfortunately a large value of `p` can cause the chain to mix slowly, because the tables will be very similar from one iteration to the next. However, it is also possible that a small value of `p` will produce too many tables with negative entries. Hence the Metropolis/Hastings/Green algorithm will stay at the current table for long periods and again result in a slowly mixing chain. Therefore, adjusting the value of `p` is usually required.

To work with the individual iterations of the chain, the program allows for the option to save the chain of goodness-of-fit statistics with the option `savechain = TRUE`. If using importance

			Primary Food Choice				
Lake	Gender	Size	Fish	Invert	Reptile	Bird	Other
1	Male	Small	7	1	0	0	5
	Male	Large	4	0	0	1	2
	Female	Small	16	3	2	2	3
	Female	Large	3	0	1	2	3
2	Male	Small	2	2	0	0	1
	Male	Large	13	7	6	0	0
	Female	Small	3	9	1	0	2
	Female	Large	0	1	0	1	0
3	Male	Small	3	7	1	0	1
	Male	Large	8	6	6	3	5
	Female	Small	2	4	1	1	4
	Female	Large	0	1	0	0	0
4	Male	Small	13	10	0	2	2
	Male	Large	9	0	0	1	2
	Female	Small	3	9	1	0	1
	Female	Large	8	1	0	0	1

Table 3: Alligator data. Source Agresti (1990) page 269.

sampling, i.e., `method = "bab"`, then both the statistic values and the importance weights on the log scale are saved. Consider the chain of goodness-of-fit statistics for the alligator food choice data:

```
R> data("alligator.dat")
R> alligator.mcx <- mcexact(y ~ (lake + gender + size) * food +
+     lake * gender * size, data = alligator.dat, nosim = 10^3,
+     method = "cab", savechain = TRUE, batchsize = 100, p = 0.75)
R> summary(alligator.mcx)
```

```
Number of iterations       =  1000
t degrees of freedom       =  3
Number of counts           =  80
df                         =  40
Number of batches          =  10
Batchsize                  =  100
Next update has nosim      =  1000
Proportion of valid tables =  0.196

                 deviance     Pearson
observed.stat 50.26368862 52.56768703
pvalue         0.20000000  0.24900000
mcse           0.07231182  0.08921827
```

The chain of goodness-of-fit statistics are saved in `alligator.mcx$chain`. The saved chain is discarded if the simulations are resumed with `update`, even if `savechain = TRUE` when the simulation is resumed.

We would want to look at the autocorrelation function of the goodness-of-fit statistics. The result of

```
R> acf(alligator.mcx$chain[, 1])
R> acf(alligator.mcx$chain[, 2])
```

is shown in Figure 1.

It is also usually useful to look at the chain of P values for the deviance statistics

```
R> top <- cumsum(alligator.mcx$chain[, 1] >= alligator.mcx$dobs[1])
R> bottom <- 1:alligator.mcx$nosim
R> dev.p <- top/bottom
R> plot(dev.p, type = "l", ylab = "P value", xlab = "iteration")
R> title("Deviance P value by iteration")
```

and Pearson statistics.

```
R> top <- cumsum(alligator.mcx$chain[, 1] >= alligator.mcx$dobs[1])
R> bottom <- (1:alligator.mcx$nosim)
R> pearson.p <- top/bottom
R> plot(pearson.p, type = "l", ylab = "Pvalue", xlab = "iteration")
R> title("Pearson P value by iteration")
```

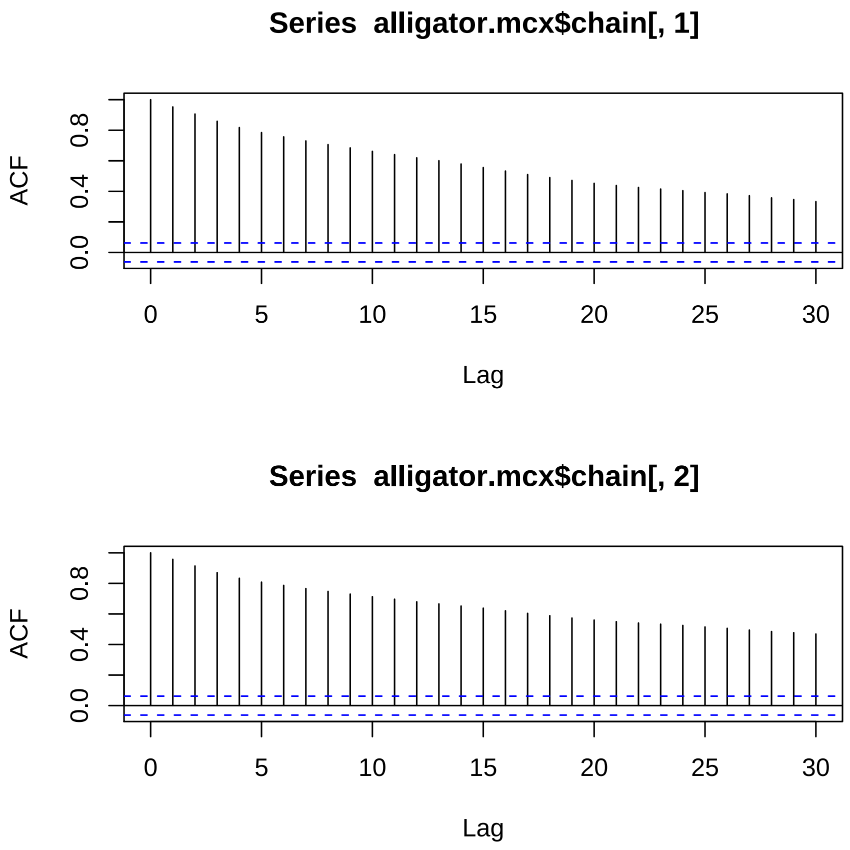


Figure 1: Autocorrelation function of the goodness-of-fit statistics.

Both are shown in Figure 2.

Note that there is an extremely slow decay in the autocorrelations of the chain of goodness-of-fit statistics and that the P values for the two statistics do not seem to have converged. Therefore, a much longer final run should be performed (not shown). Also, `mcexact` uses batch means to estimate the Monte Carlo error, and so this suggests we should use very large batch sizes.

With regard to batch sizes, `mcexact` does not require the total number of simulations to be a multiple of the batch size. If the algorithm terminates in the middle of completing a batch, that batch is not used in the P value calculations. However, the simulations are not wasted if the algorithm is resumed with `update`.

A final run of this data, discarding all of the initial runs, could be performed by setting `flush = TRUE` as an argument to `update`. Here, `flush = TRUE`, tells `update` to throw out all of the data used in the initial tinkering, except that it starts the new chain from the final table from the initial runs. This is a harmless way to burn in the chain without throwing away samples from the final run. The chain can be restarted at the default starting value, the observed data, by rerunning `mcexact`.

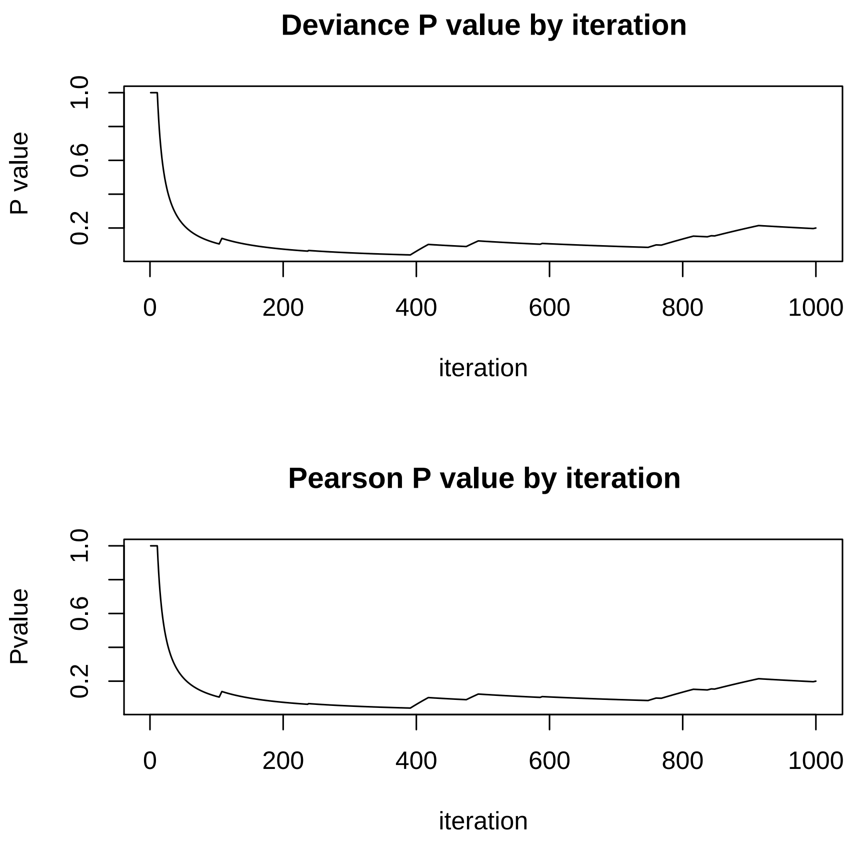


Figure 2: Chain of deviance and Pearson P values.

3.4. Exact score test for binomial counts

The data given below are contained in the R dataset `Titanic` (see Mac and Dawson 1995). Refer to the help page for the data set for more information. The data cross-classify survival counts of the Titanic passengers by class, gender and age.

```
R> data("Titanic")
R> ftable(Class ~ Survived + Sex + Age, Titanic)
```

```
                      Class 1st 2nd 3rd Crew
Survived Sex    Age
No       Male   Child         0   0  35    0
                Adult       118 154 387  670
         Female Child         0   0  17    0
                Adult         4  13  89    3
Yes      Male   Child         5  11  13    0
                Adult        57  14  75  192
         Female Child         1  13  14    0
                Adult       140  80  76   20
```

A reorganized version of the same data is provided with the **exactLoglinTest** package

```
R> data("titanic.dat")
R> titanic.dat$alpha <- rep(1:16, 2)
```

The variable `alpha` is added to correspond to the α_i terms from Equation 3.

We view each person's survival as a binary outcome. Furthermore, we use a model where a person's age, gender and class are additive effects on the logit scale. The corresponding Poisson log-linear model has model formula

```
y ~ (factor(class) + factor(age) + factor(sex)) : factor(surv) +
     factor(surv) + factor(alpha)
```

Because `mcexact` was designed for goodness-of-fit tests, one must work more directly with the simulated data in cases with an alternate goal. We use this example to illustrate the `simlu-ateConditional` command, which only performs the simulation. Its result is the simulated y values in a matrix, with each row being a simulation. When `method = "bab"` the importance weights, on the log scale, are represented in the final column.

Consider the gender effect in specific. To calculate an exact P value `simulateConditional` is used to simulate tables conditioning on all of the parameters, setting the interaction `fac-tor(surv) : factor(sex)` to 0.

```
R> chain <- simulateConditional(y ~ factor(surv) + (factor(class) +
+     factor(age)):factor(surv) + factor(alpha), dat = titanic.dat,
+     nosim = 10^4, method = "cab", p = 0.1)
```

A P value for a score test of $H_0 : \gamma = 0$ versus $H_a : \gamma < 0$ simply counts the proportion of tables with sufficient statistic for γ is smaller than the observed value. Using the notation from Equation 3 the sufficient statistic for γ is $s_\gamma = \sum_i z_i y_i \equiv z^\top y$. We calculate the chain of sufficient statistics and the observed sufficient statistic below.

```
R> z <- titanic.dat$sex * titanic.dat$surv
R> sgamma <- chain %*% z
R> sgamma.obs <- titanic.dat$y %*% z
R> mean(sgamma <= sgamma.obs[1])

[1] 0
```

Apparently, none of the simulated tables have sufficient statistics for γ below that of the observed, suggesting that we would reject the hypothesis that $\gamma = 0$.

3.5. Application to disclosure limitation

Though there are certainly more rigorous procedures available (see Dobra *et al.* 2002), **exact-LoglinTest** is a useful tool for exploring disclosure limitation in contingency tables. Consider the Czech Auto Worker's data given in Table 4. We investigate the potential disclosure risk from releasing all two-way marginals from this table. The following code will load the Czech auto worker data into a data frame:

				B	no		yes	
F	E	D	C	A	no	yes	no	yes
neg	small	small	no		44	40	112	67
			yes		129	145	12	23
		large	no		35	12	80	33
			yes		109	67	7	9
	large	small	no		23	32	70	66
			yes		50	80	7	13
		large	no		24	25	73	57
			yes		51	63	7	16
pos	small	small	no		5	7	21	9
			yes		9	17	1	4
		large	no		4	3	11	8
			yes		14	17	5	2
	large	small	no		7	3	14	14
			yes		9	16	2	3
		large	no		4	0	13	11
			yes		5	14	4	4

Table 4: Czech auto workers data. Source Dobra *et al.* (2002) originally appeared in Edwards and Havranek (1985).

```
R> data("czech.dat")
```

We will explore disclosure risk by simulating tables from the hypergeometric distribution obtained by conditioning on all two way margins. However, it is necessary to save all of the simulated table entries, not just the deviance and Pearson statistics. The function `simulateConditional` performs this task. Recall, this function returns the simulated tables in a matrix with each row being a complete simulated table.

Below we run the chain

```
R> chain <- simulateConditional(y ~ (A + B + C + D + E + F)^2,
+     data = czech.dat, method = "cab", nosim = 10^3, p = 0.4)
```

Now, the variable `chain` is a matrix so that each row is a simulated table with two-way margins equal to those of the original table. We were particularly concerned with cells 39, 48, and 55 which contained only one, two and two individuals in the observed data respectively. Consider the proportion of tables which have greater than zero but fewer than three individuals

```
R> mean(chain[, 39] > 0 & chain[, 39] < 3)

[1] 0.4

R> mean(chain[, 48] > 0 & chain[, 48] < 3)

[1] 0.534
```

```
R> mean(chain[, 55] > 0 & chain[, 55] < 3)

[1] 0.603
```

This generalized hypergeometric model was used because it fixes all two-way margins. However, that model need not fit the data well (in fact, it doesn't). Therefore, in addition to simulating from the generalized hypergeometric density, it is also of interest to simulate from other densities, such as a uniform distribution on tables with these margins. Though the normal approximations for **exactLoglinTest** were tailored specifically to the hypergeometric density, it allows for other target distributions. Here the density must be specified *on the log scale*. Multiplicative constants (additive on the log scale) can be discarded. Therefore, to specify a uniform density on the log scale, a function that returns 0 is supplied to the `dens =` option.

```
R> chain2 <- simulateConditional(y ~ (A + B + C + D + E + F)^2,
+     data = czech.dat, method = "cab", nosim = 10^3, p = 0.4,
+     dens = function(y) 0)
R> mean(chain2[, 39] > 0 & chain2[, 39] < 3)

[1] 0.055

R> mean(chain2[, 48] > 0 & chain2[, 48] < 3)

[1] 0.603

R> mean(chain2[, 55] > 0 & chain2[, 55] < 3)

[1] 0.979
```

For each cell under consideration, the algorithm discovered tables that satisfy the margins, but do not have a one or two cell count. Hence, the disclosure risk in releasing the two-way marginals seems minimal.

4. Discussion

Exact tests are useful tools for investigating goodness-of-fit for contingency table data. The primary limitation of these tests is the difficulty in implementing them. The software **exactLoglinTest** can help solve this problem for many examples. The program uses sequentially rounded normal approximations to the relevant conditional distribution to produce Monte Carlo approximations to P values. In this manuscript we investigated three straightforward examples of **exactLoglinTest** and considered two potentially useful extensions of the program.

Acknowledgments

The author would like to thank the editor, associate editor and referees for their insightful comments

References

Agresti A (1990). *Categorical Data Analysis.* Wiley, New York.

Booth J, Butler R (1999). "An Importance Sampling Algorithm for Exact Conditional Test in Log-Linear Models." *Biometrika*, **86**, 321–332.

Booth JG, Capanu M, Heigenhauser L (2005). "Exact Conditional P Value Calculation for the Quasi-symmetry Model." *Journal of Computational and Graphical Statistics*, **14**(3), 716–725.

Caffo BS, Booth JG (2001). "A Markov Chain Monte Carlo Algorithm for Approximating Exact Conditional Probabilities." *Journal of Compuatational and Graphical Statistics*, **10**, 730–745.

Caffo BS, Booth JG (2003). "Monte Carlo Conditional Inference for a Log-linear and Logistic Models: A Survey of Current Methodology." *Statistical Methods in Medical Research*, **12**(2), 109–123.

Chen Y, Diaconis P, Holmes SP, Liu JS (2005). "Sequential Monte Carlo Methods for Statistical Analysis of Tables." *Journal of the American Statistical Association*, **100**(469), 109–120.

Chen Y, Dinwoodie IH, Sullivant S (2006). "Sequential Importance Sampling for Multiway Tables." *The Annals of Statistics*, **34**(1), 523–545.

Diaconis P, Sturmfels B (1998). "Algebraic Algorithms for Sampling from Conditional Distributions." *The Annals of Statistics*, **26**, 363–397.

Dobra A (2003). "Markov Bases for Decomposable Graphical Models." *Bernoulli*, **9**(6), 1093–1108.

Dobra A, Tebaldi C, West M (2002). "Reconstruction of Contingency Tables with Missing Data." *Technical report*, Duke University.

Edwards DE, Havranek T (1985). "A Fast Procedure for Model Search in Multidimesional Contingency Tables." *Biometrika*, **72**, 339–351.

Forster JJ, McDonald JW, Smith PWF (1996). "Monte Carlo Exact Conditional Tests for Log-linear and Logistic Models." *Journal of the Royal Statistical Society B*, **58**, 445–453.

Hirji KF, Mehta CR, Patel NR (1987). "Computing Distributions for Exact Logistic Regression." *Journal of the American Statistical Association*, **82**, 1110–1117.

Mac RJ, Dawson G (1995). "The 'Unusual Episode' Data Revisited." *Journal of Statistics Education*, **3**(3), 6–6.

McDonald JW, Smith PWF, Forster JJ (1999). "Exact Tests of Goodness of Fit of Log-linear Models for Rates." *Biometrics*, **55**, 620–624.

Mehta CR (1991). "**StatXact**: A Statistical Package for Exact Nonparametric Inference." *The American Statistician*, **45**, 74–75.

Mehta CR, Patel NR (1980). "A Network Algorithm for the Exact Treatment of the $2 \times K$ Contingency Table." *Communications in Statistics: Simulation and Computation*, **9**, 649–664.

Mehta CR, Patel NR (1983). "A Network Algorithm for Performing Fisher's Exact Test in $R \times C$ Contingency Tables." *Journal of the American Statistical Association*, **78**, 427–434.

Patefield WM (1981). "[Algorithm AS 159] An Efficient Method of Generating Random $R \times C$ Tables with Given Row and Column Totals." *Applied Statistics*, **30**, 91–97.

R Development Core Team (2006a). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL http://www.R-project.org/.

R Development Core Team (2006b). *R Installation and Administration*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-09-7, URL http://www.R-project.org/.

Smith PWF, Forster JJ, McDonald JW (1996). "Monte Carlo Exact Tests for Square Contingency Tables." *Journal of the Royal Statistical Society A*, **159**, 309–321.

Strawderman RL, Wells MT (1998). "Approximately Exact Inference for the Common Odds Ratio in Several 2×2 Tables (C/R: P1307-1320)." *Journal of the American Statistical Association*, **93**, 1294–1307.

Venables WN, Ripley BD (2002). *Modern Applied Statistics with S*. Springer-Verlag, New York, 4th edition.

Affiliation:

Brian Caffo
Johns Hopkins Bloomberg School of Public Health
615 N Wolfe Street,
Baltimore, Maryland, 21205, United States of America
E-mail: bcaffo@jhsph.edu
URL: http://www.biostat.jhsph.edu/~bcaffo/

Journal of Statistical Software http://www.jstatsoft.org/
published by the American Statistical Association http://www.amstat.org/

Volume 17, Issue 7 *Submitted:* 2006-02-07
December 2006 *Accepted:* 2006-11-10