

Binary response data

A “Bernoulli trial” is a random variable that has two points in its sample space. The two points may be denoted “success/failure,” “heads/tails,” “yes/no,” “0/1,” etc.

The probability distribution of a Bernoulli trial X is completely determined by the value $P(X = 1)$, since $P(X = 0) = 1 - P(X = 1)$. The numerical value of $P(X = 1)$, often denoted p , is called the “success probability.”

A sequence of iid Bernoulli trials

$$X_1, X_2, \dots, X_n$$

arises in many applications. For example, suppose every 100th part manufactured on an assembly line is tested, and the result of the test is a designation of “success” or “failure.” These data are Bernoulli, since there are only two possible outcomes for each test. To determine whether the X_i are iid (independent and identically distributed), the following must be considered.

- Is every test equally likely to yield a success? If so, the X_i are identically distributed. However, for example, if the later tests are more likely to be successful (e.g. because the equipment was cold at the beginning of the trial and therefore yielded more faulty parts), the X_i are not identically distributed.
- Are the test results for the parts independent of each other? If so, the X_i are independent. This would not hold, for example, if a supply of raw material (say plastic pellets) were replenished periodically. If there is variability in the quality of raw material batches, then parts produced from a common batch will be positively correlated, with either a higher than average failure rate (if the raw material batch is poorer than average) or a lower than average failure rate (if the raw material batch is better than average).

A Bernoulli probability formula

Suppose the X_i are independent, but not identically distributed Bernoulli trials, where $P(X_i = 1) = p_i$. The probability of the observed value X_i is

$$P(X_i) = p_i^{X_i}(1 - p_i)^{1-X_i}.$$

This formula is valid for both $X_i = 1$, in which case the result is p_i , and $X_i = 0$, in which case the result is $1 - p_i$. More generally, the probability of a particular sequence X_1, \dots, X_n being observed is

$$p_1^{X_1}(1 - p_1)^{1-X_1} \dots p_n^{X_n}(1 - p_n)^{1-X_n} = \prod_i p_i^{X_i}(1 - p_i)^{1-X_i},$$

where here we are using the fact that probabilities of independent random variables multiply.

The binomial distribution for the total number of successes

The total number of successes is

$$T = X_1 + \dots + X_n = \sum_i X_i.$$

For example, if $n = 4$, the sequences 1100 and 1001 would both give $T = 2$.

If the X_i are iid Bernoulli trials, the probability distribution of T is the “binomial distribution,” whose probabilities are

$$P(T = k) = \binom{n}{k} p^k (1 - p)^{n-k},$$

The “cumulative probabilities” of the binomial distribution are

$$P(T \leq k) = P(T = 0) + P(T = 1) + \dots + P(T = k).$$

These can be calculated in R using

```
pbinom(k, n, p)
```

which returns the value $P(X \leq k)$ for n iid Bernoulli trials with success probability p .

Although this function is available in R, it is worth considering how it is calculated. If the binomial coefficient

$$\binom{n}{k} = \frac{n!}{(n-k)!k!},$$

overflow is possible. On the log scale this becomes

$$\log \binom{n}{k} = \sum_{j=n-k+1}^n \log j - \sum_{j=1}^k \log j.$$

The following R code calculates the binomial probability $P(X = k)$:

```
P <- 0
if ((k>0) & (k<n)) { P <- sum(log(seq(n-k+1, n))) - sum(log(seq(1, k))) }
P <- P + k*log(p) + (n-k)*log(1-p)
P <- exp(P)
```

The normal and Poisson approximations to the binomial distribution

The binomial distribution is difficult to work with since the cumulative probabilities

$$P(T \leq k) = \sum_{j \leq k} P(T = j)$$

cannot be expressed in a simple formula. Thus it is common to use some form of approximation to the binomial distribution.

The normal approximation to the binomial distribution

To derive the normal approximation, recall that the expected value and variance of each X_i are

$$EX_i = p \qquad \text{var } X_i = p(1 - p).$$

Therefore

$$ET = np \quad \text{var } T = np(1 - p).$$

Thus the standardization of T is

$$\frac{T - np}{\sqrt{np(1 - p)}},$$

which by the central limit theorem will approximately follow a standard normal distribution when n is not too small.

Suppose we want to find the tail probability $P(T > k)$. Standardizing yields

$$P\left(\frac{T - np}{\sqrt{np(1 - p)}} > \frac{k - np}{\sqrt{np(1 - p)}}\right) = 1 - F\left(\frac{k - np}{\sqrt{np(1 - p)}}\right),$$

where F is the standard normal CDF (`pnorm` in R).

The Poisson approximation to the binomial distribution

A different approximation, called the “Poisson approximation,” is more accurate than the normal approximation when p is small. A Poisson random variable G has sample space $0, 1, 2, \dots$, with probabilities

$$P(G = k) = e^{-\lambda} \lambda^k / k!,$$

where $\lambda > 0$ is a parameter.

Using $\lambda = np$ provides a good approximation to the binomial distribution with sample size n and success probability p , if p is small.

Comparisons of the normal and Poisson approximations

The following program uses R to calculate the exact Binomial right tail probability $P(T > k)$ and the approximations to this value using the normal and Poisson distributions.

```

n <- 20
k <- 2

## Consider different success probabilities.
for (p in c(0.01, 0.1, 0.2, 0.3, 0.4, 0.5))
{
  ## The probability from the normal approximation.
  N <- 1 - pnorm((k-n*p)/sqrt(n*p*(1-p)))

  ## The probability from the Poisson approximation.
  P <- 1 - ppois(k, n*p)

  ## The exact value from the binomial distribution.
  B <- 1 - pbinom(k, n, p)

  print(c(N, P, B))
}

```

Based on the output to this program you will see that the normal approximation is closer to the exact value than the Poisson approximation when $p > 0.3$, but for smaller values of p , the Poisson approximation is closer.

The value 0.3 is not always the point where the Poisson approximation becomes valid – the relative performances of the Poisson and normal approximations depend on the sample size n and on p .

Bivariate binary data – contingency tables

Suppose each individual in a random sample is measured in terms of two different binary variables X and Y . For example, individuals in a drug trial may have their gender recorded, as well as their response to the drug (yes/no). Such bivariate binary data is usually presented as a contingency table:

	Response	
	Y	N
F	n_{11}	n_{10}
M	n_{01}	n_{00}

The contingency table describes a sample of size $n = n_{11} + n_{10} + n_{01} + n_{00}$ from the probability distribution

	$Y = 1$	$Y = 0$
$X = 1$	p_{11}	p_{10}
$X = 0$	p_{01}	p_{00}

The expected value of n_{ij} is np_{ij} , and n_{ij} follows a binomial distribution with parameters p_{ij} and n . However n_{11} , n_{10} , n_{01} , and n_{00} are not independent.

Suppose we wish to simulate a sample of size n from the contingency table specified above. Put the four cells of the contingency table in the arbitrary order 11, 10, 01, 00, so the cumulative probabilities are

$$\begin{aligned} c_1 &= p_{11} \\ c_2 &= p_{11} + p_{10} \\ c_3 &= p_{11} + p_{10} + p_{01} \\ c_4 &= 1. \end{aligned}$$

If U is uniformly distributed on $(0, 1)$,

$$\begin{aligned} P(U < c_1) &= c_1 = p_{11} \\ P(c_1 \leq U < c_2) &= c_2 - c_1 = p_{10} \\ P(c_2 \leq U < c_3) &= c_3 - c_2 = p_{01} \\ P(c_3 \leq U) &= c_4 - c_3 = p_{00}. \end{aligned}$$

We can simulate the n_{11} , n_{01} , n_{10} , and n_{00} by simulating uniform random values U_1, \dots, U_n . For each U_i we add 1 to one of the cells, as follows.

$$\begin{aligned} U < c_1 & \text{ add 1 to } n_{11} \\ c_1 \leq U < c_2 & \text{ add 1 to } n_{10} \\ c_2 \leq U < c_3 & \text{ add 1 to } n_{01} \\ c_3 \leq U < c_4 & \text{ add 1 to } n_{00}. \end{aligned}$$

The odds ratio and log odds ratio

For a univariate Bernoulli trial with success probability p , the “odds” is the ratio of the success probability to the failure probability:

$$p/(1 - p).$$

In a contingency table, if we know that $X = 1$, the odds of Y are p_{11}/p_{10} . Similarly, if we know that $X = 0$, the odds of Y are p_{01}/p_{00} . The “odds ratio” is the ratio of the odds when $X = 1$ to the odds when $X = 0$:

$$\frac{p_{11}/p_{10}}{p_{01}/p_{00}} = \frac{p_{11}p_{00}}{p_{01}p_{10}}.$$

The odds ratio relates the distribution of Y for individuals with $X = 1$ to the distribution of Y for individuals with $X = 0$. If the odds ratio is positive, concordant responses ($X = 1, Y = 1$ or $X = 0, Y = 0$) are more common than discordant responses ($X = 1, Y = 0$ or $X = 0, Y = 1$). If the odds ratio is negative, discordant responses are more common.

Tests of independence

An important question about a contingency table is whether X and Y are independent. When this is that case, two numbers p and q can be found so that the population probability distribution can be written

$$\frac{pq \quad | \quad p(1 - q)}{(1 - p)q \quad | \quad (1 - p)(1 - q)}$$

It's easy to check that the contingency table can be written this way if and only if the odds ratio is one (the log odds ratio is zero).

A nice property of the odds ratio is that if the roles of X and Y are switched, the odds ratio is unchanged. The odds of X when $Y = 1$ are p_{11}/p_{01} and the odds of X when $Y = 0$ are p_{10}/p_{00} . Viewed this way, the odds ratio is

$$\frac{p_{11}/p_{01}}{p_{10}/p_{00}} = \frac{p_{11}p_{00}}{p_{10}p_{01}},$$

which agrees with the formula for the odds ratio given above.

The log odds ratio

It is common to work with the log-transformed odds ratio,

$$\log p_{11} + \log p_{00} - \log p_{10} - \log p_{01}.$$

where \log always denotes the natural logarithm. The log odds ratio is zero when X and Y are independent, is positive when concordant responses are more common than discordant responses, and is negative when discordant responses are more common than concordant responses.

In some ways the log odds ratio is like a correlation coefficient for bivariate binary data (the Pearson correlation coefficient is usually applied to bivariate continuous data). However the log odds ranges over $(-\infty, \infty)$ whereas the correlation coefficient is restricted to $(-1, 1)$.

Estimation and inference for the log odds ratio

The population odds ratio and population log odds ratios depend on the unknown population structure (i.e. the values p_{11} , p_{10} , p_{01} and p_{00}). The sample odds ratio

$$\frac{n_{11}n_{00}}{n_{10}n_{01}},$$

and the sample log odds ratio

$$\log n_{11} + \log n_{00} - \log n_{01} - \log n_{10}.$$

These are obtained by substituting the estimated cell probability $\hat{p}_{ij} = n_{ij}/n$ for the population cell probability p_{ij} .

The standard error of the log odds ratio is approximately

$$\sqrt{\frac{1/p_{11} + 1/p_{10} + 1/p_{01} + 1/p_{00}}{n}}.$$

Since we don't know the p_{ij} , in practice the plug-in estimate of the standard error is used

$$\sqrt{1/n_{11} + 1/n_{10} + 1/n_{01} + 1/n_{00}}.$$

The following simulation evaluates the coverage properties of the 95% confidence interval based on the plug-in standard error estimate for the log odds ratio. Note that if any of the cell counts n_{ij} are zero, the standard error is infinite. In this case the confidence interval always covers the true value.

```
## Simulate a 2x2 table with sample size n and cell probabilities
## given in P.
simtab <- function(P, n)
{
  ## Convert to cumulative probabilities.
  CP <- cumsum(P)

  ## Storage for the data being simulated.
  N <- array(0, c(2,2))

  ## Simulate one contingency table.
  for (j in (1:n))
  {
    J <- 1 + sum(runif(1) > CP)
    if (J == 1)      { N[1,1] <- N[1,1] + 1 }
    else if (J == 2) { N[1,2] <- N[1,2] + 1 }
    else if (J == 3) { N[2,1] <- N[2,1] + 1 }
    else if (J == 4) { N[2,2] <- N[2,2] + 1 }
  }

  return(N)
}

## The sample size.
n <- 50

## Array of coverage indicators.
```

```

C <- array(0, 1000)

for (k in (1:1000))
{
  ## Generate the four cell probabilities (p_11, p_10, p_01, p_00).
  P <- 0.1 + 0.8*runif(4)
  P <- P / sum(P)

  N <- simtab(P, n)

  ## The sample log-odds ratio and its standard error.
  LR <- log(N[1,1]) + log(N[2,2]) - log(N[1,2]) - log(N[2,1])
  SE <- sqrt(1/N[1,1] + 1/N[1,2] + 1/N[2,1] + 1/N[2,2])

  ## The population log-odds ratio.
  PLR <- log(P[1]) + log(P[4]) - log(P[2]) - log(P[3])

  ## Check for coverage.
  if (!is.finite(SE))
    { C[k] <- 1 }
  else
    { C[k] <- ((LR-1.96*SE < PLR) & (LR+1.96*SE > PLR)) }
}

```

Marginal probabilities

The *marginal probability of X* is the probability of observing a given value of X while ignoring the accompanying Y value. These probabilities are given by

$$\begin{aligned}
P(X = 1) &= p_{11} + p_{10} \\
P(X = 0) &= p_{01} + p_{00}.
\end{aligned}$$

Similarly, the marginal probabilities of Y are

$$P(Y = 1) = p_{11} + p_{01}$$

$$P(Y = 0) = p_{10} + p_{00}.$$

Marginal probabilities and joint probabilities can be estimated as follows.

		Y = 1	Y = 0
	Marginal	$(n_{11} + n_{01})/n$	$(n_{10} + n_{00})/n$
X = 1	$(n_{11} + n_{10})/n$	n_{11}/n	n_{10}/n
X = 0	$(n_{01} + n_{00})/n$	n_{01}/n	n_{00}/n

Expressed a different way, the population X and Y marginal distributions are

X	P(X)	Y	P(Y)
1	$p_{11} + p_{10}$	1	$p_{11} + p_{01}$
0	$p_{01} + p_{00}$	0	$p_{10} + p_{00}$.

and their estimates are

X	P(X)	Y	P(Y)
1	$(n_{11} + n_{10})/n$	1	$(n_{11} + n_{01})/n$
0	$(n_{01} + n_{00})/n$	0	$(n_{10} + n_{00})/n$.

Given a contingency table whose X and Y measurements may or may not be independent, let

$$p_X = (n_{11} + n_{10})/n \quad p_Y = (n_{11} + n_{01})/n.$$

The following table is the closest exactly independent table to the observed table.

	Y = 1	Y = 0
X = 1	$np_X p_Y$	$np_X(1 - p_Y)$
X = 0	$n(1 - p_X)p_Y$	$n(1 - p_X)(1 - p_Y)$

To simplify notation below, we will refer to the values in this table as follows.

	$Y = 1$	$Y = 0$
$X = 1$	E_{11}	E_{10}
$X = 0$	E_{01}	E_{00}

χ^2 tests of association

We have already seen one way to test for independence between X and Y in a contingency table – test whether the log odds ratio is zero using a normal approximation as a reference distribution.

A second approach to testing the independence of X and Y is based on the χ^2 statistic:

$$\chi^2 = \sum_{ij} (n_{ij} - E_{ij})^2 / E_{ij}.$$

Under the null hypothesis that X and Y are independent, the χ^2 statistic approximately has a χ_1^2 distribution (i.e. a χ^2 distribution with 1 degree of freedom). The following simulation checks the accuracy of this approximation for four different null population structures. You may get NA as a result, which occurs when a cell in the observed table is zero. This issue will be considered in detail below.

```
## The sample size.
n <- 75

## Storage for chi^2 test statistics (column 1) and p-values (column 2).
R <- array(0, c(5000,2))

## Null population structures.
for (px in c(0.2, 0.3, 0.4, 0.5))
{
  ## Use equal marginal distributions.
  py <- px

  for (k in (1:5000))
  {
```

```

## Simulate a null data set.
X <- (runif(n) < px)
Y <- (runif(n) < py)

## Put the data into a contingency table.
Q <- array(0, c(2,2))
Q[1,1] <- sum(X*Y)
Q[1,2] <- sum(X*(1-Y))
Q[2,1] <- sum((1-X)*Y)
Q[2,2] <- sum((1-X)*(1-Y))

## Estimated X and Y marginal distributions.
pxh <- mean(X == 1)
pyh <- mean(Y == 1)

## Construct the closest exactly independent table to the
## observed table.
E <- array(0, c(2,2))
E[1,1] <- n*pxh*pyh
E[1,2] <- n*pxh*(1-pyh)
E[2,1] <- n*(1-pxh)*pyh
E[2,2] <- n*(1-pxh)*(1-pyh)

## The chi^2 statistic.
R[k,1] <- sum( (E-Q)^2 / E )

## The p-value based on the 1df chi^2 distribution.
R[k,2] <- 1-pchisq(R[k,1], 1)
}

## These should be close to 0.05 and 0.01, since all the simulated
## data sets are null.
B <- c(mean(R[,2] < 0.05), mean(R[,2] < 0.01))
print(B)

```

```
}
```

In the previous program, it is possible that some of the values in E are zero, in which case the χ^2 statistic is undefined. One way to work around this is to add a “pseudo-count” to each cell count (below the value 0.1 is used). If the sample size is large this works well, but if the sample size is not large, or if the marginal distribution of X or Y is very skewed, the p-values are biased downward substantially.

```
## The sample size.
```

```
n <- 50
```

```
## Storage for chi^2 test statistics (column 1) and p-values (column 2).
```

```
R <- array(0, c(5000,2))
```

```
## Null population structures.
```

```
for (px in c(0.01, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5))
```

```
{
```

```
  ## Use equal marginal distributions.
```

```
  py <- px
```

```
  for (k in (1:5000))
```

```
  {
```

```
    ## Simulate a null data set.
```

```
    X <- (runif(n) < px)
```

```
    Y <- (runif(n) < py)
```

```
    ## Put the data into a contingency table, adding a pseudocount of 1
```

```
    ## to each cell.
```

```
    Q <- array(0, c(2,2))
```

```
    Q[1,1] <- sum(X*Y) + 0.1
```

```
    Q[1,2] <- sum(X*(1-Y)) + 0.1
```

```
    Q[2,1] <- sum((1-X)*Y) + 0.1
```

```
    Q[2,2] <- sum((1-X)*(1-Y)) + 0.1
```

```

## Estimated X and Y marginal distributions, including pseudo-counts.
pxh <- (Q[1,1] + Q[1,2]) / sum(Q)
pyh <- (Q[1,1] + Q[2,1]) / sum(Q)

## Construct a null table that is exactly independent.
E <- array(0, c(2,2))
E[1,1] <- n*pxh*pyh
E[1,2] <- n*pxh*(1-pyh)
E[2,1] <- n*(1-pxh)*pyh
E[2,2] <- n*(1-pxh)*(1-pyh)

## The chi^2 statistic.
R[k,1] <- sum( (E-Q)^2 / E )

## The p-value based on the 1df chi^2 distribution.
R[k,2] <- 1-pchisq(R[k,1], 1)
}

## These should be close to 0.05 and 0.01, since all the simulated
## data sets are null.
B <- c(mean(R[,2] < 0.05), mean(R[,2] < 0.01))
print(B)
}

```

Fisher's exact test

Fisher's exact test is another approach to testing the null hypothesis of no association between X and Y in a contingency table. The key idea is to restrict the null distribution to tables that have the same row and column totals as the observed table. First we need to be able to list all these tables.

For a given table of observed counts, define the following quantities

$$\begin{aligned}
 n_{1+} &= n_{11} + n_{10} && \text{row 1 total} \\
 n_{0+} &= n_{01} + n_{00} && \text{row 2 total} \\
 n_{+1} &= n_{11} + n_{01} && \text{column 1 total}
 \end{aligned}$$

Now suppose another table has count \tilde{n}_{11} in the upper left cell. For this table to have the same row and column totals as the observed table, the other counts must be

$$\begin{aligned}\tilde{n}_{10} &= n_{1+} - \tilde{n}_{11} \\ \tilde{n}_{01} &= n_{+1} - \tilde{n}_{11} \\ \tilde{n}_{00} &= n_{0+} - \tilde{n}_{01}.\end{aligned}$$

Thus the value of \tilde{n}_{11} uniquely determines the remaining cell counts of a contingency table that has the same row and column totals as the observed table.

Since

$$0 \leq \tilde{n}_{11} \leq \min(n_{1+}, n_{+1}),$$

it follows that there are at most $\min(n_{1+}, n_{+1})$ contingency tables with the same row and column totals as the observed table. In general, there will be fewer than this number, since some of the \tilde{n}_{ij} values determined by the above equations may be negative, in which case that particular solution must be ignored.

The following program prints out all contingency tables with the same row and column totals as a given table Q.

```
for (k in (0:sum(Q[1,])))
{
  B <- array(0, c(2,2))

  B[1,1] <- k
  B[1,2] <- sum(Q[1,]) - k
  B[2,1] <- sum(Q[,1]) - k
  B[2,2] <- sum(Q[2,]) - B[2,1]

  if (min(B) >= 0)
  {
    print(B)
  }
}
```

Now that we can list the contingency tables with the same row and column marginals as the observed table, we need to determine the probability of each of them. It turns out that these probabilities are given by the *hypergeometric distribution*:

$$\binom{n_{1+}}{n_{11}} \binom{n_{0+}}{n_{01}} / \binom{n}{n_{+1}}$$

The following program illustrates the hypergeometric distribution by generating a random contingency table, enumerating all the tables having the same row and column marginals, and calculating the probability of each. You can check that these probabilities sum to one.

```
## Calculate the log binomial coefficient 'm choose n'.
lbincoeff <- function(m, n)
{
  c <- 0
  if ( (n<m) & (n>0) )
  {
    c <- sum(log(1:m)) - sum(log(1:n)) - sum(log(1:(m-n)))
  }
  return(c)
}

## Generate a random 'actual table'.
Q <- array(ceiling(10*runif(4)), c(2,2))

## Total the probabilities to check that they sum to 1.
T <- 0

for (k in (0:sum(Q[1,])))
{
  ## Generate a table with k in the 1,1 position having the same row
  ## and column totals as Q.
  B <- array(0, c(2,2))
  B[1,1] <- k
}
```

```

B[1,2] <- sum(Q[1,]) - k
B[2,1] <- sum(Q[,1]) - k
B[2,2] <- sum(Q[2,]) - B[2,1]

## Ignore tables with negative cell counts.
if (min(B) >= 0)
{
  ## Calculate the hypergeometric probability.
  H <- lbincoeff(B[1,1]+B[1,2], B[1,1])
  H <- H + lbincoeff(B[2,1]+B[2,2], B[2,1])
  H <- H - lbincoeff(sum(B), B[1,1]+B[2,1])
  T <- T + exp(H)

  ## Print out the table and its probability.
  print(B)
  print(exp(H))
}
}

```

The final step in Fisher's exact test is to select from among the tables having the same row and column marginals as the observed table those tables having equal or greater evidence for dependence as the observed table. Here we can measure dependence using the log odds ratio. The p-value from Fisher's exact test is therefore the sum of probabilities (from the hypergeometric distribution) of all tables having log odds ratio greater than or equal to the observed table. This is the one sided test for a positive dependence between X and Y . For the two sided test, select all tables with the absolute value of the log odds ratio greater than the absolute value of the observed log odds ratio.

The following R function calculates the p-value for Fisher's exact test.

```

FEP <- function(Q)
{
  ## Fisher's test can't handle these cases so give them a p-value of 1.
  if (min(colSums(Q)) == 0) { return(1) }

```

```

if (min(rowSums(Q)) == 0) { return(1) }

## The log odds ratio for the input table Q.
AL <- log(Q[1,1]) + log(Q[2,2]) - log(Q[1,2]) - log(Q[2,1])

## This will be the p-value.
PV <- 0

for (k in (0:sum(Q[1,])))
{
  ## Generate a table with k in the 1,1 position having the same row
  ## and column totals as Q.
  B <- array(0, c(2,2))
  B[1,1] <- k
  B[1,2] <- sum(Q[1,]) - k
  B[2,1] <- sum(Q[,1]) - k
  B[2,2] <- sum(Q[2,]) - B[2,1]

  ## Ignore tables with negative cell counts.
  if (min(B) >= 0)
  {
    ## The log odds ratio for the constructed table B.
    LOR <- log(B[1,1]) + log(B[2,2]) - log(B[1,2]) - log(B[2,1])

    ## Do the one-sided test.
    if (LOR >= AL)
    {
      ## Calculate the hypergeometric probability.
      H <- lbincoeff(B[1,1]+B[1,2], B[1,1])
      H <- H + lbincoeff(B[2,1]+B[2,2], B[2,1])
      H <- H - lbincoeff(sum(B), B[1,1]+B[2,1])

      ## Update the p-value.
      PV <- PV + exp(H)
    }
  }
}

```

```

    }
  }
}

return(PV)
}

```

The following simulation looks at the distribution of p-values when applying Fisher's exact test to null data. Note that unlike most other p-values, the p-values from Fisher's exact test will not have a uniform distribution under null data, and will generally have expected value greater than 1/2. Thus the procedure is somewhat conservative.

```

## The sample size.
n <- 20

## Storage for the p-values.
F <- array(0, 1000)

for (k in (1:1000))
{
  ## The marginal row and column probabilities.
  px <- 0.1+0.8*runif(1)
  py <- 0.1+0.8*runif(1)

  ## Generate a data set from a null model.
  X <- (runif(n) < px)
  Y <- (runif(n) < py)
  N <- array(0, c(2,2))
  N[1,1] <- sum(X*Y)
  N[1,2] <- sum(X*(1-Y))
  N[2,1] <- sum((1-X)*Y)
  N[2,2] <- sum((1-X)*(1-Y))

  F[k] <- FEP(N)
}

```

```
}
```

The following simulation evaluates the p-values produced by Fisher's exact test under alternative distributions.

```
## The sample size.
n <- 20

B <- array(0, c(100,2))

for (k in (1:100))
{
  ## Generate the population structure.
  P <- runif(4)
  P <- P / sum(P)

  N <- simtab(P, n)

  ## The p-value from Fisher's exact test.
  fep <- FEP(N)

  ## Generate p-values for 1000 null data sets.
  F <- array(0, 1000)
  for (r in (1:1000))
  {
    X <- (runif(n) < runif(1))
    Y <- (runif(n) < runif(1))
    N1 <- array(0, c(2,2))
    N1[1,1] <- sum(X*Y)
    N1[1,2] <- sum(X*(1-Y))
    N1[2,1] <- sum((1-X)*Y)
    N1[2,2] <- sum((1-X)*(1-Y))

    F[r] <- FEP(N1)
  }
}
```

```
B[k,] <- c(fep, mean(F < fep))  
}
```