**Section 6.2.2 – Permutation test for independence**

Permutation tests are a general way to perform hypothesis tests. The test is most often performed via Monte Carlo simulation to obtain a very good estimate of the exact distribution of a statistic of interest. In simple cases, Monte Carlo distribution is not needed as demonstrated next.

The actual distribution for X2 is not , but rather closely related to the hypergeometric distribution. Below is the table from the lady tasting tea example but now with X2 included:

|  |  |  |  |
| --- | --- | --- | --- |
| m | P(M = m) |  | X2 |
| 0 | 0.0143 | 0 | 8 |
| 1 | 0.2286 | 1/9 | 2 |
| 2 | 0.5143 | 1 | 0 |
| 3 | 0.2286 | 9 | 2 |
| 4 | 0.0143 | >9 | 8 |

Therefore, the exact distribution for X2 (think of X2 as a random variable) is

P(X2 = 0) = 0.5143,

P(X2 = 2) = 0.2286 + 0.2286 = 0.4572, and

P(X2 = 8) = 0.0143 + 0.0143 = 0.0286,

where rounding error leads to these probabilities here not summing exactly to 1. Below is a plot of the CDF for this distribution and a  (code in Tea.R):



There are some differences between the two CDFs!

A permutation test for independence can use this exact distribution then to calculate p-values. Thus, if X2 was observed to be 2, then the p-value is P(X2 ≥ 2) = 0.4572 + 0.0286 = 0.4858.

When the sample and/or contingency table size is large, it will not be as easy to use the hypergeometric or multiple hypergeometric distributions. Instead, a form of Monte Carlo simulation can be used to obtain a very good approximation to the exact distribution of X2. Similar to what was done in Chapter 3, a large number, say B, of contingency tables are simulated under independence. The test statistic is calculated for each of these tables, say , to build an approximation to the exact distribution. The p-value is then .

The difference between Chapter 3 and here is HOW the contingency tables are simulated. We are now going to keep the row AND column totals fixed! Consider again the hypothetical table for the lady tasting tea example:

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Lady’s response |  |
|  |  | Milk = 1 | Tea = 2 |  |
| Actual | Milk = 1 | 3 | 1 | 4 |
| Tea = 2 | 1 | 3 | 4 |
|  |  | 4 | 4 | 8 |

We can re-write this table in a “raw” data representation as shown on the left below.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Row | Column |  | Row | Column |  | Row | Column |
| 1 | z1 = 1 |  | 1 | z2 = 1 |  | 1 | z1 = 1 |
| 1 | z2 = 1 |  | 1 | z1 = 1 |  | 1 | z2 = 1 |
| 1 | z3 = 1 |  | 1 | z3 = 1 |  | 1 | z7 = 2 |
| 1 | z4 = 2 |  | 1 | z4 = 2 |  | 1 | z4 = 2 |
| 2 | z5 = 1 |  | 2 | z5 = 1 |  | 2 | z5 = 1 |
| 2 | z6 = 2 |  | 2 | z6 = 2 |  | 2 | z8 = 2 |
| 2 | z7 = 2 |  | 2 | z7 = 2 |  | 2 | z3 = 1 |
| 2 | z8 = 2 |  | 2 | z8 = 2 |  | 2 | z6 = 2 |

Notice there are four 1’s and four 2’s in the “row” portion and four 1’s and four 2’s in the “column” portion. Also, notice there are three times where row = 1 and column = 1, just like in the 2×2 contingency table representation. The zj, for j = 1, …, 8, labels are included to help us track the 1’s and 2’s when we re-order them shortly.

Under independence, the row numbers and the column numbers are equally likely to be paired up! Thus, we could “mix up” or “permute” the row numbers and also do the same for the column numbers to obtain one possible contingency table. Because the total number of 1’s and 2’s is fixed for both the row and column variable, we can actually just permute the column (or row) values only. Thus, there are 8! = 40,320 different permutations with each having an equal probability to occur due to independence.

The middle and right side raw data representations show two possible permutations (X2 = 2 in the middle and X2 = 0 on the right). In total, there are 20,736 different permutations that lead to X2 = 0. Notice that 20,736/40,320 = 0.5143, which is the same probability as obtained earlier with the hypergeometric.

The algorithm that can be used to perform the permutation test by Monte Carlo simulation is as follows:

* + 1. Randomly permute the column numbers while keeping the row numbers unchanged.
		2. Calculate X2 for the simulated data set; denote this statistic as  to differentiate it from the original data’s X2 value
		3. Repeat steps 1 and 2 B times, where B is a large number (e.g., 1,000 or more).
		4. Plot a histogram 
		5. Calculate ; this is the p-value for a hypothesis test

Note that step 1 is just sampling the column numbers without replacement.

Example: Fiber enriched crackers (FiberExact.R)

|  |  |  |
| --- | --- | --- |
|  |  | Bloating severity |
|  |  | None | Low | Medium | High |
| Fiber source | None | 6 | 4 | 2 | 0 |
| Bran | 7 | 4 | 1 | 0 |
| Gum | 2 | 2 | 3 | 5 |
| Both | 2 | 5 | 3 | 2 |

> diet <- read.csv(file = "C:\\data\\Fiber.csv")

> head(diet)

 fiber bloat count

1 bran high 0

2 gum high 5

3 both high 2

4 none high 0

5 bran medium 1

6 gum medium 3

> # Match order given in table

> diet$fiber <- factor(x = diet$fiber, levels = c("none",

 "bran", "gum", "both"))

> diet$bloat <- factor(x = diet$bloat, levels = c("none",

 "low", "medium", "high"))

> diet.table <- xtabs(formula = count ~ fiber + bloat, data = diet)

> diet.table

 bloat

fiber none low medium high

 none 6 4 2 0

 bran 7 4 1 0

 gum 2 2 3 5

 both 2 5 3 2

> fisher.test(x = diet.table)

 Fisher's Exact Test for Count Data

data: diet.table

p-value = 0.06636

alternative hypothesis: two.sided

Fisher’s exact test produces a p-value of 0.0664. In a previous set of notes, we performed the Pearson chi-square test for independence resulting in a p-value of 0.0496.

One way to perform the permutation test is through using the simulate.p.value = TRUE argument in chisq.test():

> set.seed(8912)

> chisq.test(x = diet.table, correct = FALSE,

 simulate.p.value = TRUE, B = 1000)

 Pearson's Chi-squared test with simulated p-value (based

 on 1000 replicates)

data: diet.table

X-squared = 16.943, df = NA, p-value = 0.03896

There were B = 1,000 different permutations used. Is this enough? Because the p-value here is simply a “observed proportion of successes”, we could use a confidence interval for the probability of success (Chapter 1) to obtain a range which we expect the exact p-value to fall within:

> set.seed(8912)

> save.p <- chisq.test(x = diet.table, correct = FALSE,

 simulate.p.value = TRUE, B = 1000)

> library(package = binom)

> binom.confint(x = round(save.p$p.value\*1000,0), n = 1000,

 conf.level = 0.95, methods = "wilson")

 method x n mean lower upper

1 wilson 39 1000 0.039 0.02865896 0.05286931

One could also take a larger number of permutations:

> set.seed(8912)

> save.p2 <- chisq.test(x = diet.table, correct = FALSE,

 simulate.p.value = TRUE, B = 100000)

> binom.confint(x = round(save.p2$p.value\*100000,0), n =

 100000, conf.level = 0.95, methods = "wilson")

 method x n mean lower upper

1 wilson 4552 1e+05 0.04552 0.04424545 0.04682946

Similar to Chapter 3, I would like to actually see the histogram for all of the  values. By plotting a  on this histogram too, one can determine how well the  distribution approximates the exact distribution. Unfortunately, chisq.test() does not provide the histogram so we will need to create the plot ourselves. Below is the process that I used to perform the test and create the plot:

> # Put the data into its raw form – could just use diet

> set1 <- as.data.frame(as.table(diet.table))

> tail(set1) # Notice 2 obs. for fiber = both and bloat =

 high

 fiber bloat Freq

11 gum medium 3

12 both medium 3

13 none high 0

14 bran high 0

15 gum high 5

16 both high 2

> set2 <- set1[rep(x = 1:nrow(set1), times = set1$Freq), -3]

> tail(set2) # Notice 2 rows for fiber = both and bloat =

 high

 fiber bloat

15.1 gum high

15.2 gum high

15.3 gum high

15.4 gum high

16 both high

16.1 both high

> # Check

> xtabs(formula = ~ set2[,1] + set2[,2])

 set2[, 2]

set2[, 1] none low medium high

 none 6 4 2 0

 bran 7 4 1 0

 gum 2 2 3 5

 both 2 5 3 2

> X.sq <- chisq.test(set2[,1], set2[,2], correct = FALSE)

> X.sq$statistic

X-squared

 16.94267

Warning message:

In chisq.test(set2[, 1], set2[, 2], correct = FALSE) :

 Chi-squared approximation may be incorrect

> # Do one permutation to illustrate

> set.seed(4088)

> set2.star <- data.frame(row = set2[,1], column =

 sample(set2[,2], replace = FALSE))

> xtabs(formula = ~ set2.star[,1] + set2.star[,2])

 set2.star[, 2]

set2.star[, 1] none low medium high

 none 3 6 2 1

 bran 6 4 1 1

 gum 5 4 3 0

 both 3 1 3 5

> X.sq.star <- chisq.test(set2.star[,1], set2.star[,2],

 correct = FALSE)

> X.sq.star$statistic

X-squared

14.63903

Warning message:

In chisq.test(set2.star[, 1], set2.star[, 2], correct = FALSE) :

 Chi-squared approximation may be incorrect

> # Permutation test

> B <- 1000

> X.sq.star.save <- matrix(data = NA, nrow = B, ncol = 1)

> set.seed(1938)

> for(i in 1:B) {

 set2.star <- data.frame(row = set2[,1], column =

 sample(set2[,2], replace = FALSE))

 X.sq.star <- chisq.test(set2.star[,1], set2.star[,2],

 correct = FALSE)

 X.sq.star.save[i,1] <- X.sq.star$statistic

 }

There were 50 or more warnings (use warnings() to see the first 50)

> mean(X.sq.star.save >= X.sq$statistic)

[1] 0.039

> summarize <- function(result.set, statistic, df, B,

 color.line = "red") {

 par(mfrow = c(1,3), mar = c(5,4,4,0.5))

 # Histogram

 hist(x = result.set, main = "Histogram", freq = FALSE,

 xlab = expression(X^{"2\*"}))

 curve(expr = dchisq(x = x, df = df), col = color.line,

 add = TRUE, lwd = 2)

 segments(x0 = statistic, y0 = -10, x1 = statistic, y1 =

 10)

 # Compare CDFs

 plot.ecdf(x = result.set, verticals = TRUE, do.p =

 FALSE, main = "CDFs", lwd = 2, col = "black", xlab =

 expression(X^"2\*"), ylab = "CDF")

 curve(expr = pchisq(q = x, df = df), col = color.line,

 add = TRUE, lwd = 2, lty = "dotted")

 legend(x = df, y = 0.4, legend = c(expression(Perm.),

 substitute(chi[df1]^2, list(df1 = df))), lwd =

 c(2,2), col = c("black", color.line), lty =

 c("solid", "dotted"), bty = "n")

 # QQ-Plot

 chi.quant <- qchisq(p = seq(from = 1/(B+1), to = 1-

 1/(B+1), by = 1/(B+1)), df = df)

 plot(x = sort(result.set), y = chi.quant, main = "QQ-

 plot", xlab = expression(X^{"2\*"}), ylab = "Chi-

 square quantiles")

 abline(a = 0, b = 1)

 par(mfrow = c(1,1))

 # p-value

 mean(result.set >= statistic)

 }

> summarize(result.set = X.sq.star.save, statistic =

 X.sq$statistic, df = (nrow(diet.table)-1) \*

 (ncol(diet.table)-1), B = B)

[1] 0.039



The p-value here is very similar to what we had before. Also, we see that a  distribution does a very good job with approximating the distribution of X2.

Question: How could we find estimates of P(X2 = x2) to obtain the estimated PMF? Note that the lowercase x2 is used to emphasize a particular observed value.

Final comments:

* Computational Statistics and Nonparametrics courses often discuss permutation tests for other statistical problems.
* An example of when the  distribution approximation works poorly is for the following data:

> c.table <- array(data = c(0, 1, 0,

 7, 1, 8,

 0, 1, 0,

 0, 1, 0,

 0, 1, 0,

 0, 1, 0,

 0, 1, 0,

 1, 0, 0,

 1, 0, 0), dim = c(3,9))

> c.table

 [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9]

[1,] 0 7 0 0 0 0 0 1 1

[2,] 1 1 1 1 1 1 1 0 0

[3,] 0 8 0 0 0 0 0 0 0

Run the analysis methods for this section to examine it more closely.