# Chi-Squared Residuals

*davegoblue*

*March 11, 2016*

This document is to validate how R calculates chi-squared, Cramer's V, and standardized residuals for a simple test of categorical association. The test data frame will be a small 3x3 example.

```
testFrame <- data.frame(colI = c(5, 34, 33), colII = c(6, 47, 32),
                        colIII = c(9, 48, 14), row.names=c("A","B","C")
                        )
testFrame
```

```
##   colI colII colIII
## A    5     6      9
## B   34    47     48
## C   33    32     14
```

Next, expected values are calculated for each cell as N * Prow * Pcol. Further, residuals are calculated as (observed - expected):

```
expMatrix <- as.matrix(rowSums(testFrame)) %*% t(as.matrix(colSums(testFrame)))
expMatrix <- expMatrix / sum(testFrame)
round(expMatrix,1)
```

```
##   colI colII colIII
## A  6.3   7.5    6.2
## B 40.7  48.1   40.2
## C 24.9  29.5   24.6
```

```
resMatrix <- testFrame - expMatrix
round(resMatrix,1)
```

```
##   colI colII colIII
## A -1.3  -1.5    2.8
## B -6.7  -1.1    7.8
## C  8.1   2.5  -10.6
```

Then, we run (by hand) each of the chi-squared, df, Cramer's V, and standardized residuals:

```
testChiSq <- sum(resMatrix^2 / expMatrix)
nR <- nrow(resMatrix)
nC <- ncol(resMatrix)

dfChiSq <- (nR - 1) * (nC - 1)
cramerV <- sqrt(testChiSq / (nR * nC) / (min(nR, nC) - 1) )

## Pearson residuals - (obs - exp) / sqrt(exp)
prsMatrix <- resMatrix / sqrt(expMatrix)

## Standardized residuals - (obs - exp) / sqrt(V)
## Vij = sqrt(Expij * (1 - pRow) * (1 - pCol))
stdMatrix <- resMatrix /
             sqrt(expMatrix *
                   (as.matrix(1 - rowSums(expMatrix)/sum(expMatrix))
                    %*%
                    t(as.matrix(1-colSums(expMatrix)/sum(expMatrix)))
                    )
                  )

print(paste0("Chi-squared is ",round(testChiSq,2)," with df=",dfChiSq,
             " (p=",
             round(pchisq(testChiSq, df=dfChiSq, lower.tail=FALSE),4),
             ")"
             )
      )
```

```
## [1] "Chi-squared is 11.84 with df=4 (p=0.0185)"
```

```
print(paste0("Cramer's V is: ",round(cramerV,3)))
```

```
## [1] "Cramer's V is: 0.811"
```

```
## Pearson residuals - (obs - exp) / sqrt(exp)
round(prsMatrix,2)
```

```
##    colI colII colIII
## A -0.52 -0.53   1.11
## B -1.06 -0.16   1.24
## C  1.61  0.47  -2.14
## Standardized residuals - (obs - exp) / sqrt(V)
## Vij = sqrt(Expij * (1 - pRow) * (1 - pCol))
round(stdMatrix,2)
```

```
##    colI colII colIII
## A -0.66 -0.70   1.40
## B -1.94 -0.30   2.26
## C  2.41  0.73  -3.19
```

The results are compared with just running the chi-squared test in R, listing each of the 9 dimensions in order. Note that following as per residual calculations in chisq.test:

- “residuals”" are the Pearson residuals (observed - expected) / sqrt(expected)
- “stdres” are the standardized residuals (observed - expected) / sqrt(V)
- Vij = sqrt[ EXPij x (1 - P_i_ALL) x (1 - P_ALL_j ) ]

```
testChi <- chisq.test(testFrame)
for (intCtr in 1:length(testChi)) {
    print(testChi[intCtr])
}
```

```
## $statistic
## X-squared
##  11.84471
##
## $parameter
## df
##  4
##
## $p.value
## [1] 0.01854417
##
## $method
## [1] "Pearson's Chi-squared test"
##
## $data.name
## [1] "testFrame"
##
## $observed
##   colI colII colIII
## A    5     6      9
## B   34    47     48
## C   33    32     14
##
## $expected
##        colI    colII   colIII
## A  6.315789  7.45614  6.22807
## B 40.736842 48.09211 40.17105
## C 24.947368 29.45175 24.60088
##
## $residuals
##         colI      colII    colIII
## A -0.5235674 -0.5332688  1.110722
## B -1.0555108 -0.1574808  1.235227
## C  1.6122243  0.4695542 -2.137305
##
## $stdres
##         colI      colII    colIII
## A -0.6626947 -0.7049874  1.401389
## B -1.9365006 -0.3017705  2.258989
## C  2.4110425  0.7334312 -3.186093
## Borrowed from http://www.r-bloggers.com/example-8-39-calculating-cramers-v/
## With adaptations
cv.test <- function(x) {
    CV <- sqrt(chisq.test(x, correct=FALSE)$statistic /
               (sum(!is.na(x)) * (min(nrow(x), ncol(x)) - 1) )
               )

    print.noquote("Cramér V / Phi:")

    return(as.numeric(CV))
}

cv.test(testFrame)
```

```
## [1] Cramér V / Phi:
```

```
## [1] 0.8111964
```

The hand-calculated results match up with the R pchisq.test outputs in this particular example.