## R code for contingency tables

```
# prop.r:  R code for proportions and
#   contingency tables

# N.B. many of these calculations are as easy by hand
#    as by R functions

# ci or test for a single proportion: prop.test(),
#   default PI0 = 0.5
# two ways to specify data

# 1) 2 column matrix with # success, # failure
prop.test(t(c(20,40)))

# 2) 2 arguments y, n = # success, # trials
prop.test(20,60)
```

```
# can specify your own Pi0 (p=, or 3rd argument)
prop.test(t(c(20,40)), p=0.7)

# all above give you tests (Chi-square = Z^2) and ci's
#   ci's from inverting a score test
#   i.e. using null probability to calculate variance
# so in first 2 examples, upper ci limit is 0.4679
#  p <- 0.4679; p - 1.96*sqrt(p*(1-p)/60) = 0.5

# N.B. the above uses a continuity correction
#   'when needed'.  I don't know R's 'when needed'
# R does report whether it is or is not used
# if you really don't want it (I really don't want it)
# I recommend add ``correct = F'' to the call
# difference / equality of two proportions:
# 1) Chi-square test of equality + ci for difference
prop.test(rbind(c(20,40), c(50,50)))

# note above uses continuity correction, now turn off
prop.test(rbind(c(20,40), c(50,50)), correct=F)

# different ci and p-value.

# 2) Chi-square test only
chisq.test(rbind(c(20,40), c(50,50)), correct=F)

# chisq.test() can be used for general R x C tables
# my understanding is that prop.test() can only be
#   used for R x 2 tables
# can pass two factor variables to chisq.test(),
#  e.g. chisq.test(factor1, factor2, correct=F)
# and chisq.test will construct the contingency
#  table, then calculate the Chi-square statistic
# Chi-square test against specified probabilities
# E.g. count # events per 5 minutes for 100 5 minute
#    periods.  Data is 13,30,27,14,10,3,1,1,1
# i.e. 0 events in 13 periods, 1 event in 30 periods,
$    ... 8 events in 1 period
# are these consistent with a Poisson(mean=2.5)
#   distribution?
# have to expand one category so probabilities
#   sum to 1
# P[8 events] is turned into P[8 or more]
#   = 1-P[7 or less]

chisq.test(c(13,30,27,14,10,3,1,1,1),
   p=c(dpois(0:7,2.5),1-ppois(7,2.5)))
```

## Odds ratios

```
# if you don't want to calculate odds ratios
#  'by hand', the oddsratio() set of functions in
#  the epitools package provides many different
#  ways to estimate / calculate ci's for odds
# the differences among the approaches are 557
#  (Categorical Data) material, not 511 material.

# fisher.test() calculates the odds ratio we've used
```

## Fisher exact test

```
# Fisher exact test: not really necessary with
#   this size sample.
# one of the stat computing miracles of the late
#   1990's was a very fast algorithm for enumerating
#   possible contingency tables
# also gives you the estimated odds ratio

fisher.test(rbind(c(20,40), c(50,50)))

# Here's when you need Fisher: lecture example
fisher.test(rbind(c(1,7), c(4,4)))

# very different p-value than from:
chisq.test(rbind(c(1,7), c(4,4)),correct=F)
# (but continuity corrected version is close)
# can also give two columns (response on
#   variable 1, response on variable 2)
# fisher.test will compute the contingency table
# just like chisq.test()

# both chisq.test() and fisher.test() can be used
#   with general R x C tables
```

## McNemar’s test

```
# McNemar’s test
# can do it by hand (binomial tail probability)
#   or by mcnemar.test()
# made up data: 20 pairs with ++, 5 with +-,
#   20 with -+, 30 with --

mcnemar.test(rbind(c(20,5), c(20,30)), correct=F)

# by hand, to get "exact" p-value
2*pbinom(5,25,0.5)
# 5 from #(+-), 25 from #(+-) + #(-+)
# if #(+-) > n/2, add lower=F to get correct tail
```