Tutorials

by William B. King, Ph.D.
Coastal Carolina University

I think, therefore I R.

| Table of Contents | Function Reference | Function Finder | R Project |

CHI SQUARE GOODNESS OF FIT TEST

Syntax

From the help page, the syntax of the `chisq.test()` function is...

```
chisq.test(x, y = NULL, correct = TRUE,
           p = rep(1/length(x), length(x)), rescale.p = FALSE,
           simulate.p.value = FALSE, B = 2000)
```

This function is used for both the goodness of fit test and the test of independence, and which it does depends upon what kind of data you feed it. If "x" is a numeric vector or a one-dimensional table of numerical values, a goodness of fit test will be done (or attempted), treating "x" as a vector of observed frequencies. If "x" is a 2-D table, array, or matrix, then it is assumed to be a contingency table of frequencies, and a test of independence will be done.

Ignore "y". (See the help page if you must know.) The "correct=TRUE" option applies the Yates continuity correction when "x" is a 2x2 table. Set this to FALSE if the correction is not desired. For the goodness of fit test, set "p" equal to the null hypothesized proportions or probabilies for each of the categories represented in the vector "x". These must add exactly to 1. In the examples, I'll also illustrate how this vector can be set to the expected frequencies. The default is equal frequencies or probabilities. The remaining options can be ignored.

Textbook Problems

"Professor Bumblefuss takes a random sample of students enrolled in Statistics 101 at ABC University. He finds the following: there are 25 freshman in the sample, 32 sophomores, 18 juniors, and 20 seniors. Test the null hypothesis that freshman, sophomores, juniors, and seniors are equally represented among students signed up for Stat 101."

This is a goodness of fit test with equal expected frequencies. The "p" vector does not need to be specified, since equal frequencies is the default. The observed frequencies can be fed directly into the test, using `c()` to collect them into a vector, or the frequencies can be put into a vector first and the name of the vector fed to the test.

```
> chisq.test(c(25,32,18,20))          # or freqs=c(25,32,18,20); chisq.test(freqs)

        Chi-squared test for given probabilities

data:  c(25, 32, 18, 20)
X-squared = 4.9158, df = 3, p-value = 0.1781
```

Either way, the null hypothesis cannot be rejected at alpha = .05.

"Professor Iconoclast argues that Bumblefuss is wrong, and that the number of freshman and sophomores enrolled is each twice the number of juniors and the number of seniors. Test Iconoclast's hypothesis from these same data."

Now the expected frequencies are no longer equal, so a "p" vector must be specified. A little algebra leads us to the expected probabilities of 1/3, 1/3, 1/6, and 1/6. Since these numbers are awkward to enter as decimal numbers, they are best entered as fractions, letting R do the arithmetic for us. (You should always use fractions, when possible.)

```
> null.probs = c(1/3,1/3,1/6,1/6)
> freqs = c(25,32,18,20)
> chisq.test(freqs, p=null.probs)      # p is not the second option, so must be labeled

        Chi-squared test for given probabilities

data:  freqs
X-squared = 2.8, df = 3, p-value = 0.4235
```

A warning: Since R does not expect the "p" vector to come second inside the `chisq.test()` function, it MUST be given its correct name, "p=". It doesn't appear that we can reject Iconoclast's null hypothesis either.

The chi-square distribution is a continuous probability distribution, which is being used here as an approximation to the discrete case. To be an accurate approximation, the expected frequencies must be sufficiently large. R will issue a warning message if any of the EFs fall below 5, but it's good practice to check them nevertheless. This leads us to a device we will use often in future tutorials--storing the output of an R procedure to an object in the workspace, and then extracting the information we want. Usually, the printed output of an R procedure is only a small part of what R has calculated. Here's how to see more.

```
> results = chisq.test(freqs, p=null.probs)
> results

        Chi-squared test for given probabilities

data:  freqs
X-squared = 2.8, df = 3, p-value = 0.4235

> results$expected
[1] 31.66667 31.66667 15.83333 15.83333
> results$residuals
[1] -1.18469776  0.05923489  0.54451008  1.04713477
```

The "results" object now contains a list of the stuff R has calculated. To see what is in the list, see the section headed "Value" on the help page for the `chisq.test()` function. The query "results$expected" shows the expected frequencies under the null hypothesis. The query "results$residuals" shows the unsquared chi values in each cell, allowing us to see where the biggest deviations are from frequencies the null predicted.

One more example will demonstrate another trick for specifying expected frequencies. "Last year, the same study was done on the Stat 101 class, and frequencies observed were: 30 freshman, 28 sophomores, 28 juniors, and 11 seniors. Test to see if this year's results matched last years, to within the limits of random sampling error." It's an awkward sample size, so a good way to handle the expected probabilities is like this.

```
> new.freqs = c(30,28,28,11)
> freqs -> old.freqs
> chisq.test(new.freqs, p=old.freqs/sum(old.freqs))

        Chi-squared test for given probabilities

data:  new.freqs
X-squared = 10.8353, df = 3, p-value = 0.01265
```

Looks like we're significantly off from what was observed last year.

Data From a Table Object

The object entered into `chisq.test()` for "x=" must be a vector of observed frequencies. If the data are in some other form, a vector must be extracted somehow. I will use the built-in data set "HairEyeColor" to demonstrate. This is a 3-D table that crosstabulates hair color, eye color, and sex for 592 statistics students. Suppose we are interested only in eye color.

```
> dimnames(HairEyeColor)
$Hair
[1] "Black" "Brown" "Red"   "Blond"

$Eye
[1] "Brown" "Blue"  "Hazel" "Green"

$Sex
[1] "Male"   "Female"

> eyes = margin.table(HairEyeColor, 2)      # eye color in margin 2
> eyes
Eye
Brown  Blue Hazel Green
  220   215    93    64
```

Suppose a certain physiologist proposes that 50% of people should have brown eyes, 25% blue eyes, 15% hazel eyes, and 10% green eyes. (I don't know why. I made these numbers up!) Does the sample at hand agree with the physiologist's hypothesis?

```
> chisq.test(eyes, p=(.5, .25, .15, .1))
Error: unexpected ',' in "chisq.test(eyes, p=(.5,"
>
> # Okay, I have to remember how to create a vector! Sheesh!
>
> chisq.test(eyes, p=c(.5, .25, .15, .1))

        Chi-squared test for given probabilities

data:  eyes
X-squared = 50.4324, df = 3, p-value = 6.462e-11
```

The physiologist's hypothesis is not supported. (Note: See the More Descriptive Statistics tutorial for the `margin.table()` function.

Data From a Data Frame

If a data frame contains categorical data in one variable, it can be extracted into a 1-D frequency table using the `table()` function.

```
> data(survey, package="MASS")
> str(survey)
'data.frame':   237 obs. of  12 variables:
 $ Sex   : Factor w/ 2 levels "Female","Male": 1 2 2 2 2 1 2 1 2 2 ...
 $ Wr.Hnd: num  18.5 19.5 18 18.8 20 18 17.7 17 20 18.5 ...
 $ NW.Hnd: num  18 20.5 13.3 18.9 20 17.7 17.7 17.3 19.5 18.5 ...
 $ W.Hnd : Factor w/ 2 levels "Left","Right": 2 1 2 2 2 2 2 2 2 2 ...
 $ Fold  : Factor w/ 3 levels "L on R","Neither",..: 3 3 1 3 2 1 1 3 3 3 ...
 $ Pulse : int  92 104 87 NA 35 64 83 74 72 90 ...
 $ Clap  : Factor w/ 3 levels "Left","Neither",..: 1 1 2 2 3 3 3 3 3 3 ...
 $ Exer  : Factor w/ 3 levels "Freq","None",..: 3 2 2 2 3 3 1 1 3 3 ...
 $ Smoke : Factor w/ 4 levels "Heavy","Never",..: 2 4 3 2 2 2 2 2 2 2 ...
 $ Height: num  173 178  NA 160 165 ...
 $ M.I   : Factor w/ 2 levels "Imperial","Metric": 2 1 NA 2 2 1 1 2 2 2 ...
 $ Age   : num  18.2 17.6 16.9 20.3 23.7 ...
> table(survey$Smoke)

Heavy Never Occas Regul
   11   189    19    17
>
> smokers = table(survey$Smoke)
> smokers

Heavy Never Occas Regul
   11   189    19    17
```

These are the responses of 237 statistics students at the University of Adelaide to a number of questions. The "Smoke" variable contains their responses to a question about smoking habits, and resulted in each student being classified into one of four categories: Never, Occasional, Regular, and Heavy. We have extracted these data and have placed them into a 1-D table of frequencies called "smokers". This table can now be subjected to any reasonable (or even unreasonable) goodness of fit test. For example, we can test the null hypothesis that 70% of statistics students are nonsmokers, and that the other 30% are divided equally among the remaining categories. The only thing we have to be careful of is that our null probabilities are given in the same order as the frequencies are given in the "smokers" table. (R has put them in a somewhat odd order, because R doesn't really understand the English language, so it just arranged them alphabetically.)

```
> chisq.test(smokers, p=c(.1, .7, .1, .1))

        Chi-squared test for given probabilities

data:  smokers
X-squared = 12.8983, df = 3, p-value = 0.004862

> chisq.test(smokers, p=c(.1, .7, .1, .1))$resid

    Heavy     Never     Occas     Regul
-2.593669  1.851706 -0.946895 -1.358588
```

The hypothesis is rejected. It appears in particular that we have very much overestimated the percentage of "Heavy" smokers. Two things. One, the last part of this example shows that we do not need to store the output of the procedure to request residuals. All we need to do is stick "$resid" on the end of the command. We could have done the same with "$expected" or any other element of the outputted results. Also demonstrated is the convenient fact that, if we are too lazy to type the entire word "residuals," we can use just enough of it that R is able to recognize what we are asking for.

In an undergraduate statistics class, you would probably be asked to calculate the expected frequencies. That's not hard. There are 236 subjects in the "smokers" vector (and one missing value). 10% of 236 is 23.6, and 70% is 165.2. What if we tried to enter those actual expected frequencies into the p= option? The function would fail (try it), because those probabilities MUST add to one. Well, sort of. Here's how to do it.

```
> chisq.test(smokers, p=c(23.6, 165.2, 23.6, 23.6), rescale.p=TRUE)

        Chi-squared test for given probabilities

data:  smokers
X-squared = 12.8983, df = 3, p-value = 0.004862
```

The rescale.p=TRUE option does the trick. Set it any time your p= values don't add exactly to 1.

---

revised 2016 January 27
