8 Goodness of Fit Test

For a continuous random variable X, that is, for a random variable whose range is a non-countable subset of $\mathbb{R}$, the probability of any particular outcome x is zero:

$$\mathbb{P}[X = x] = 0 \ ,$$

so defining a probability distribution as we did for discrete random variables makes little sense. Instead one defines a *probability density function (pdf)* $p_X(x)$ such that for any two numbers a and b in the range of X and such that $a \leq b$ the following identity holds:

$$\mathbb{P}[a < X \leq b] = \int_a^b p_X(x)\, dx \ .$$

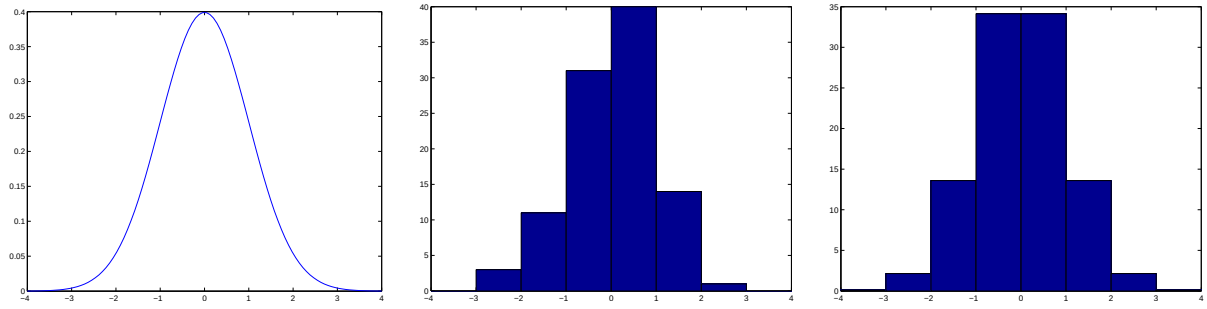
Thus, the density itself is not a probability. Instead, the integral of the density over any interval yields the probability that the random variable X takes on values in that interval. Since the probability that X takes on *some* value in its range $\mathcal{X}$ is one, it must be the case that

$$\int_{\mathcal{X}} p_X(x)\, dx \ = \ 1 \ .$$

For instance, the probability density for a Gaussian random variable G with mean m and standard deviation σ is

$$p_G(x) = \frac{1}{\sqrt{2\pi}\sigma}\, e^{-\frac{1}{2}\left(\frac{x-m}{\sigma}\right)^2} \ .$$

Figure 26 (a) shows the Gaussian pdf for $m = 0$ and $\sigma = 1$.



(a) (b) (c)

Figure 26: (1) The probability density function of a Gaussian random variable with zero mean and unit standard deviation. (b) Histogram of a set of 100 data points. (c) Expected number of samples in each of the bins in part (a) for a true Gaussian distribution with zero mean and unit standard deviation.

Suppose now that we are given a single histogram of, say, $n = 100$ data points, such as the one in Figure 26 (b). If the histogram has k bins, let

$$b_0 \ , \ \ldots \ , \ b_k$$

be the $k+1$ bin boundaries. More precisely, each boundary other than b_0 is assumed to be part of the bin to its left, so that data point x is in bin i for $i = 1, \ldots, k$ if

$$b_{i-1} < x \leq b_i \ .$$

Data points x for which

$$x \leq b_0 \quad \text{or} \quad b_k < x$$

are not counted. Thus, the histogram in question is specified by the $k+1$ bin boundaries and by the k observed bin counts $o_1, \ldots, o_k$.

The question arises naturally, whether this histogram is consistent with a given pdf. For instance, we could ask whether it is plausible that the data summarized by this histogram was drawn from a Gaussian population with the pdf in Figure 26 (a). If the histogram is recomputed from a new sample of points, it will be in general different from the first, because of the randomness of the data. In other words, each of the bin counts o_i in the given histogram is a particular outcome of a random variable O_i, and one should not expect any "exact" consistency with the null hypothesis. Are the inconsistencies small enough that they can be attributed to randomness? Goodness of fit tests are meant to answer questions like this.

The hypothesis that the data was indeed drawn from the stated distribution is called the *null hypothesis*. This Section considers the particular type of goodness of fit test called the chi square test, often written "χ^2 test."

The χ^2 *test* takes as its input

- An observed *histogram* $o_1, \ldots, o_k$ with bin boundaries $b_0, \ldots, b_k$.
- A *null hypothesis* in the form of a candidate distribution $p_X(x)$.
- A number α called the *significance level* (usually small; for instance, $\alpha = 0.05$.)

The test returns one of the following answers:

- *Reject* the null hypothesis if there is a probability of at most α that the given histogram (or a histogram that fits the null hypothesis even less well) was generated by the candidate distribution.
- *Do not reject* the null hypothesis otherwise.

Note that if the null hypothesis is not rejected it is not automatically accepted: failure to reject the null hypothesis at a significance level α merely means that evidence against the hypothesis is not overwhelming. It does *not* mean that there is evidence in favor of the hypothesis. Failing to realize this is perhaps the most common misunderstanding about hypothesis testing. We return to this point after describing how the chi square test works.

It is conceptually straightforward to compute the *expected* histogram that would be obtained under the null hypothesis. The expected *fraction* of points in bin i is equal to the probability that a

point falls in that bin:

$$p_i = \int_{b_{i-1}}^{b_i} p_X(x)\, dx \ , \tag{51}$$

and the expected *number* of points in bin i is then

$$e_i = np_i = n \int_{b_{i-1}}^{b_i} p_X(x)\, dx \ . \tag{52}$$

Note that the numbers e_i are generally not integer, in contrast with the observed values o_i. This is because the expected count for bin i is the average of the counts one would obtain from a large number of histograms. The average of a set of integers is generally not an integer.

Since the bin counts o_i are outcomes of random variables O_i, it should be possible to compute the probability distribution of each O_i under the null hypothesis. The probability that a data point falls in bin i is p_i, and if the data are drawn independently from each other the count O_i is a binomial random variable with probability distribution

$$p_{O_i}(j) = \binom{n}{j} p_i^j (1-p_i)^{n-j} \ .$$

as shown in Section 3. The mean of this random variable is then

$$m_{O_i} = np_i = e_i$$

(see equation (52), and its variance is

$$s_{O_i}^2 = np_i(1-p_i) = np_i - np_i^2 \ .$$

If the bins are not too large, the probability p_i is substantially smaller than 1, and p_i^2 is negligible with respect to p_i. One therefore obtains

$$e_i = m_{O_i} \approx \sigma_{O_i}^2 \ .$$

To measure the discrepancy between the observed histogram o_i and the histogram e_i computed under the null hypothesis (for $i = 1, \ldots, n$), it is then natural to use the measure

$$\overline{\chi}^2 = \sum_{i=1}^{k} \frac{(o_i - e_i)^2}{e_i}; .$$

The ratios

$$\frac{o_i - e_i}{\sqrt{e_i}} \tag{53}$$

are the bin discrepancies normalized by their standard deviations $\sigma_{O_i} = \sqrt{e_i}$. This normalization makes the measure of discrepancy independent of the number n of data points (but not of the number k of bins). Squaring the discrepancies ensures that they are all counted with a positive sign, and adding the squares amounts to giving each bin the same importance.

Since the observed bin values o_i are outcomes of the random variables O_i, the value $\overline{\chi}^2$ is itself an outcome of the random variable[36]

$$\chi^2 = \sum_{i=1}^{k} \frac{(O_i - e_i)^2}{e_i};.$$

For a sufficiently large number n of data points, the binomial distribution of O_i is well approximated by a Gaussian distribution with mean and variance both equal to e_i. In turn, if the O_i are Gaussian, and if one accounts for the constraint

$$\sum_{i=1}^{k} O_i = n$$

that the sum of the histogram counts must equal the given number n of data points, it can be proven that the random variable χ^2 is distributed according to the following *chi square probability density function with $d = k - 1$ degrees of freedom*:

$$p_{\chi^2}(x;\, d) = \frac{1}{\Gamma(d/2)} \frac{1}{2^{d/2}} x^{d/2-1} e^{-x/2} \quad \text{for} \quad x \geq 0 \,.$$

In this expression, $\Gamma(z)$ is the gamma function:

$$\Gamma(z) = \int_0^\infty t^{z-1} e^{-t}\, dt \,.$$

The following special values of the gamma function are sufficient to compute the coefficients of the chi square probability density function:

$$\begin{aligned}
\Gamma(0) &= 0 \\
\Gamma(1) &= 1 \\
\Gamma(n) &= (n-1)! \qquad \text{for integer } n > 1 \\
\Gamma(1/2) &= \sqrt{\pi} \;\approx\; 1.77245 \\
\Gamma(n+1/2) &= \sqrt{\pi} \frac{1 \cdot 3 \cdot 5 \cdot \ldots \cdot (2n-1)}{2^n} \qquad \text{for integer } n > 0.
\end{aligned}$$

Thanks to the centering and normalization performed in equation (53), the number $d = k - 1$ of degrees of freedom turns out to be the only (known!) parameter of this pdf. Because of this, one can compute the probability of obtaining an observed histogram whose χ^2 deviation from the expected histogram is at least $\overline{\chi}$:

$$p = \mathbb{P}[\chi^2 \geq \overline{\chi}^2] = \int_{\overline{\chi}^2}^\infty p_{\chi^2}(x;\, d)\, dx \,.$$

[36] Here we depart from the convention of using uppercase for the random variable and lowercase for its values. This is because the uppercase Greek letter chi looks the same as the uppercase Latin letter X. Instead, we use the lowercase Greek letter χ to denote the random variable, and the over-lined symbol $\overline{\chi}^2$ to denote its outcome values.

If this probability p is small (in particular, smaller than the pre-specified significance value α), then it is very unlikely (there is a probability smaller than α) that data that conform with the null hypothesis could have produced the given histogram. Thus,

> If $p \leq \alpha$, we reject the null hypothesis at a significance level α. That is, we reject the null hypothesis, but there is a probability α that we are wrong in doing so.

It is important to observe that if $p > \alpha$, the null hypothesis may still be unlikely. It is just not unlikely enough that we feel justified in rejecting it outright. Suppose for instance, that we set a significance level $\alpha = 0.05$ (a typical value), and that we obtain a value $\overline{\chi^2}$ for the discrepancy measure χ^2 between observed and theoretical histogram such that

$$p = \mathbb{P}[\chi^2 \geq \overline{\chi}^2] = 0.1 \ .$$

This means that, on the average, in ten percent of the cases an n-point data sample that is drawn from the pdf specified in the null hypothesis will generate a histogram that differs from the theoretical one by at least $\overline{\chi^2}$. So the observed discrepancy $\overline{\chi^2}$ is still rather unlikely under the null hypothesis: in ninety percent of the cases we ought to observe a smaller discrepancy, if the null hypothesis is indeed true. We do not feel authorized to reject the hypothesis outright, since we stipulated to do so only under the demanding criterion that the probability of obtaining the given histogram (or an even worse one) under the null hypothesis is exceedingly small ($p \leq \alpha$). However, jumping to the conclusion that then the null hypothesis is correct is unwarranted: the observed discrepancy (or a worse one) is still supposed to arise in only ten percent of the histograms we would obtain under the null hypothesis, so the fact that we did indeed see a discrepancy that large should at least make us suspicious of the null hypothesis. Because of these considerations, it has become more common to report the p value, rather than a reject/no reject decision.

Even a much greater value of p would not authorize us to accept the null hypothesis, but for a different reason. Suppose for instance that the discrepancy $\overline{\chi}^2$ is so small that

$$p = \mathbb{P}[\chi^2 \geq \overline{\chi}^2] = 0.99 \ .$$

This means that the observed histogram is nearly identical to the theoretical one. Does this tell us that the underlying distribution from which the data was drawn is equal to the distribution specified by the null hypothesis? The answer is no, because histograms are incomplete summaries of distributions. There may be two different distributions that yield the same histogram, and this is especially the case when the histograms have few bins. Consider for instance the theoretical histogram in Figure 26 (a). One way to produce data with that histogram is to deterministically output the number -2.5 three times, followed by the number -1.5 twelve times, then -0.5 32 times, and so forth. This is clearly not a Gaussian random variable. It is not even a random variable, nor are the drawings independent. In brief, very different histograms imply different distributions (with high probability), but the converse is not true.

The null hypothesis is sometimes specified by giving a *family* of densities, rather than a single density. For instance, one might hypothesize that the histogram comes from a Gaussian random

variable, without specifying its mean or variance. In that case, the parameters can be estimated from the histogram:

$$m \approx \frac{1}{n} \sum_{i=1}^{k} c_i \, o_i \quad \text{and} \quad \sigma^2 \approx \frac{1}{n-1} \sum_{i=1}^{k} (c_i - m)^2 \, o_i$$

where

$$c_i = \frac{b_{i-1} + b_i}{2} \quad \text{for} \quad i = 1, \ldots, k$$

are the bin centers. Alternatively, these parameters can be estimated somewhat more accurately from the data samples d_j for $j = 1, \ldots, n$:

$$m \approx \frac{1}{n} \sum_{j=1}^{n} d_j \quad \text{and} \quad \sigma^2 \approx \frac{1}{n-1} \sum_{j=1}^{n} (d_j - m)^2 \; .$$

The denominator $n - 1$ instead of n in the expressions for the variance is more a curiosity than a deep concept: it turns out that the expression with $1/n$ would yield a *biased* estimator of the variance, one that is consistently smaller than it should be, by a factor $n/(n-1)$. However, n is typically large enough that the difference is insignificant.

When parameters are estimated from the data, the degrees of freedom d of the χ^2 distribution must be reduced accordingly, by subtracting the number of parameters being estimated. For instance, if mean and variance must be estimated, we set

$$d = k - 1 - 2 = k - 3$$

instead of just $d = k - 1$ as done earlier. The need for this correction will not be proven here. Intuitively, this reflects the fact that only $k - 3$ bin counts can be chosen independently. One of the other three is determined by the fact that the bin counts must add up to n. The other two can be inferred from the others through the values of m and σ.

Note the somewhat conflicting assumptions underlying the χ^2 test: on one hand, we need the expected number of samples $e_i = np_i$ in each bin to be large enough that the corresponding observed value O_i becomes approximately a Gaussian random variable (rather than a binomial). On the other hand, the bins must be small enough so that p_i^2 can be neglected relative to p_i. So we need small bins with large counts. This can be obtained with small bins and large sample sizes (n), but this is often an unattainable luxury in practice.

A practical recipe for approximating the proper conditions for the chi square test is to require that each bin have at least a count of five. If this is not the case, adjacent bins are merged, to make larger bins with greater counts. Of course, counts cannot be too large, or else p_i^2 is no longer negligible relative to p_i. One way to enforce a tight bound on the number of samples per bin is the procedure of *histogram equalization*: rather than choosing a fixed number of bins, we can choose a fixed count per bin, and then size each bin so that it has the required count. All bins are different in width, but they all have approximately the same count. The approximation comes from the fact that the number n of samples may not be an integer multiple of the required bin count. For

instance, with a desired bin count of 10 and $n = 203$, there are 19 bins with count 10, plus a bin with count 13. The alternative of 20 bins with count 10 and one bin with count 3 is not acceptable, because of the need to have at least five samples per bin.

Figure 27 shows the result of histogram equalization with a desired count of 10, applied to the histogram in Figure 26 (a). As expected, this is a boring histogram: all counts are equal to 10 ($n = 100$ happens to be a multiple of 10, so there is not even a leftover at the end). The bins, however, are all different in width. Where the original histogram has high counts, the equalized version has narrow bins: more samples require more bins if each bin holds the same number of samples. The two histograms convey essentially the same information, but in different ways.

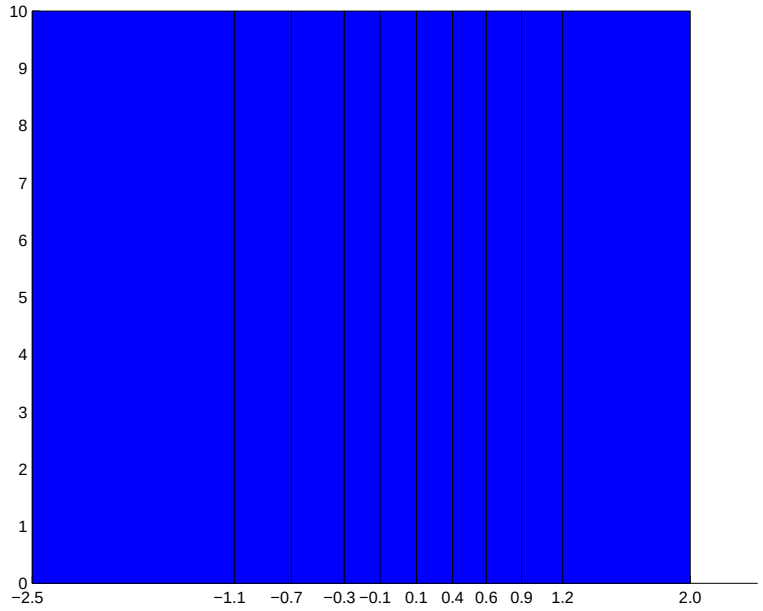


Figure 27: The equalized version of the histogram in Figure 26 (a).

The following function `chi2test`, written in Matlab, computes the chi square statistics for an arbitrary histogram and a null hypothesis that specifies an arbitrary probability density function. The Statistics Toolbox in Matlab has its own version of chi square, called `chi2gof` (for "goodness of fit"). However, writing your own is a good exercise, both for programming and for math, and saves you the expense of an additional toolbox. The interface of the function (what parameters to pass and what results the function returns) is nontrivial, so a lengthy comment above the code is appropriate.

```
% function [reject, chi2, p, dof, edges, o, e] = chi2test(d, count, alpha, ecdf)
%
% Performs the chi-square goodness-of-fit test on the data in o.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Inputs:
```

```
%
% d: a vector of observed data (the samples themselves, not a histogram)
%
% count: a vector of two integers specifying the minimum and maximum number
%   of samples per bin. Default: count = [5 max(5, floor(length(d)/10))].
%   In this way, about ten bins are usually used, and none of them has
%   fewer than five samples.
%
% alpha: the significance level. Default: alpha = 0.05
%
% ecdf: a cell array that specifies the cumulative probability
%   distribution function of the null hypothesis. The first element of the
%   cell array is a function handle. Subsequent elements, if any, are
%   parameters (arrays, possibly scalar or vector). If
%       ecdf = {f, p1, ..., pn}
%   then it must be possible to call
%       f(x, {p1, ..., pn})
%   where x is a vector of values.
%   Default: ecdf = {@(x) (1 + erf(x/sqrt(2)))/2}
%   (the cumulative distribution of a Gaussian with mean 0 and standard
%   deviation 1).
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Outputs:
%
% reject: true if the null hypothesis is rejected; false otherwise
%
% chi2: the chi-square statitstic
%
% p: the probability of observing the given statistics or a greater value
%
% dof: the degrees of freedom
%
% edges: the bin edges actually used for the statistics
%
% o: the histogram of observed data with the given bin edges
%
% e: the estimated histogram on the same bins
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [reject, chi2, p, dof, edges, o, e] = chi2test(d, count, alpha, ecdf)

if nargin < 2 || isempty(count)
    count = [5 max(5, floor(length(d)/10))];
end

if nargin < 3 || isempty(alpha)
    alpha = 0.05;
end
```

```
if nargin < 4 || isempty(ecdf)
    ecdf = {@(x) (1 + erf(x/sqrt(2)))/2};
end

% Check that there are enough data
d = d(:)';
n = length(d);
if n < 5
    error('Need at least five data points for chi-square statistic')
end
if count(2) < count(1)
    error('Maximum number of samples per bin cannot be smaller than minimum number')
end
if n < count(2)
    count(2) = n;
end

% Determine the bin edges and bin counts by histogram equalization
d = sort(d);
j = count(2):count(2):n;    % Index of highest data point in each bin
if j(end) < n % Data do not divide evenly into the bins
    if n - j(end) < count(1)
        % Not enough data in the last bin: merge it with the one before
        j(end) = n;
    else
        % Enough data to add one bin at the end
        j = [j, n];
    end
end
% Internal bin edges first
edges = (d(j(1:(end-1))) + d(j(1:(end-1)) + 1)) / 2;
% Now add an edge at the beginning and one at the end
small = mean(d(j(1:(end-1)) + 1) - d(j(1:(end-1))));
edges = [d(1) - small, edges, d(end) + small];

% Observed bin counts
o = [j(1), j(2:end) - j(1:(end-1))];

% Compute expected bin counts under the null hypothesis
if length(ecdf) == 1     % No parameters
    c = ecdf{1}(edges);
else
    c = ecdf{1}(edges, ecdf(2:end));
end
e = n * (c(2:end) - c(1:(end-1)));

% Degrees of freedom
dof = length(o) - 1;    % -1 because of the constraint on the number of data points
for k = 2:length(ecdf)
    dof = dof - numel(ecdf{k});
```

```
end

chi2 = sum((o - e).^2 ./ e);

p = 1 - chi2cdf(chi2, dof);

reject = p < alpha;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function p = chi2cdf(x, dof)

if dof < 1 || dof ~= round(dof)
    error('Degrees of freedom must be a positive integer')
end

if any(x < 0)
    error('The chi-square distribution is only defined for nonnegative x')
end

p = gammainc(x/2, dof/2);

% Just to make sure that round-off does not put p out of range
p(p < 0) = 0;
p(p > 1) = 1;
```

Let us understand the easy part first: the function `chi2cdf` at the bottom. This computes the *cumulative distribution function* of the χ^2 random variable, that is,

$$c_{\chi^2}(x;\, d) = \mathbb{P}[\chi^2 \le x] = \int_0^x p_{\chi^2}(\xi;\, d)\, d\xi = \int_0^x \frac{1}{\Gamma(d/2)}\, \frac{1}{2^{d/2}}\, \xi^{d/2-1}\, e^{-\xi/2}\, d\xi\ .$$

Fortunately, Matlab provides a function `gammainc` thus defined:

$$P(x, a) = \frac{1}{\Gamma(a)} \int_0^x t^{a-1}\, e^{-t}\, dt\ .$$

The change of variable

$$\xi = 2t$$

then yields

$$P(x, a) \;=\; \frac{1}{\Gamma(a)} \int_0^{2x} \frac{\xi^{a-1}}{2^{a-1}}\, e^{-\xi/2}\, \frac{1}{2}\, d\xi \;=\; \frac{1}{\Gamma(a)}\, \frac{1}{2^a} \int_0^{2x} \xi^{a-1}\, e^{-\xi/2}\, d\xi \;=\; p_{\chi^2}(2x; 2a)$$

so that

$$p_{\chi^2}(x;\, d) = P(x/2,\, d/2)\ .$$

This is what `chi2cdf` does.

Turning our attention to the main function `chi2test`, the input and output parameters are explained in the comments. The more esoteric feature used in this code is that of a *function handle*, which is a way to pass a function as an argument to another function. In our case, we

need a mechanism for passing the cumulative distribution function $c_X(x)$ corresponding to the density $p_X(x)$ specified in the null hypothesis, in order to compute the bin probabilities (51):

$$p_i \;=\; \int_{b_{i-1}}^{b_i} p_X(x)\,dx \;=\; \int_{-\infty}^{b_i} p_X(x)\,dx \;-\; \int_{-\infty}^{b_{i-1}} p_X(x)\,dx \;=\; c_X(b_i) \;-\; c_X(b_{i-1})\;.$$

In order to pass a function, we create a function handle. There are two was to do this. The one that is easier to understand defines the function in a separate file, say `f.m`, and then passes

```
@f
```

as an argument to `chi2test`. For instance, the uniform density

$$p_U(x) = 1 \quad \text{for} \quad 0 \le x \le 1$$

has cumulative distribution function

$$c_U(x) = \begin{cases} \int_0^x 1\,d\xi = x & \text{for } 0 \le x \le 1 \\ 0 & \text{for } x < 0 \\ 1 & \text{for } x > 1 \end{cases} \quad .$$

So we can write

```
function c = ucdf(x)

c = max(min(x, 1), 0);
```

in a separate file, and then call, say

```
reject = chi2test(d, [], [], {@ucdf})
```

Note that instead of passing the handle itself, `@ucdf`, we wrap it into the only element of a cell structure (the braces around it). This has nothing to do with the handle issue: since some cumulative distribution functions have parameters (such as the mean and variance of a Gaussian, for instance), we bundle these in the same cell structure:

```
{@gauss, m, sigma}
```

so we can have a variable number of parameters.

A more compact way to pass the uniform cumulative distribution is to define the function `ucdf` on the fly, so to speak, and without giving it a name. This is possible through the Matlab notion of an *anonymous function*:

```
reject = chi2test(d, [], [], {@(x) max(min(x, 1), 0)})
```

Note the last argument (still wrapped into a cell structure): the `@` declares this to be a function handle; the `(x)` is the argument list for the function, and the rest is any Matlab expression (which could be arbitrarily complicated) that defines the function body. Of course, this syntax is convenient only when the function body is a short snippet. For longer functions, readability demands that the body be written in a separate file, and the function be named.

The rest of the code is relatively straightforward: after checking input arguments and defining an anonymous function for the cumulative function of a Gaussian as a default fourth parameter:

```
ecdf = {@(x) (1 + erf(x/sqrt(2)))/2};
```

a few more lines of code check for correctness of the data: at least five data points, consistent minimum and maximum bin count.

A dozen lines of code (from `d = sort(d);` to `o = [j(1), j(2:end) - j(1:(end-1))];`) then perform histogram equalization.

The following `if` group calls the cumulative function (without or with parameters) and computes the theoretical histogram. The last four lines of code compute the number of degrees of freedom, the χ^2 statistic, the probability

$$p = \mathbb{P}[\chi^2 \geq \overline{\chi}^2]$$

(by calling `chi2cdf`), and the decision whether to reject the null hypothesis.

Here is a sample call:

```
[reject, chi2, p, dof] = chi2test(randn(100, 1))
reject =
     0
chi2 =
    4.38
p =
    0.88
dof =
     9
```

The number formatting was shrunk to save space (see `help format` for how to do this). The hypothesis that the 100 samples generated by the pesudo-random Gaussian generator `randn` are drawn from a Gaussian distribution with zero mean and variance 1 (the default function handle provided by `chi2test`) was not rejected. A χ^2 value of 4.38 or greater was deemed to occur with a probability $p = 0.88$, so there is no evidence that the data do not come from a Gaussian.

Just removing an `n` (going from `randn` to `rand`) would of course change the conclusion:

```
[reject, chi2, p, dof] = chi2test(rand(100, 1))
reject =
     1
chi2 =
  152.38
p =
     0
dof =
     9
```

This is because data drawn from a uniform distribution does not lead to a histogram that is consistent with a Gaussian. The null hypothesis of a uniform distribution, on the other hand, explains uniform data well:

```
[reject, chi2, p, dof] = chi2test(rand(100, 1), [], [], {@(x) max(min(x, 1), 0)})
reject =
     0
chi2 =
    6.55
p =
    0.68
dof =
     9
```

Caveat: the current code does not work when the distribution in the null hypothesis is defined over a proper subset of the distribution from which the data is drawn. For instance, the call

```
[reject, chi2, p, dof] = chi2test(randn(100, 1), [], [], {@(x) max(min(x, 1), 0)})
```

would fail, because Gaussian samples (`randn`) are defined on the entire real line, but the uniform distribution is defined only on $[0, 1]$. As a result, some of the bins where $o_i \neq 0$ have $e_i = 0$, and the χ^2 statistic is undefined because of divisions by zero.

Exercise: fix this by redefining the bins if this occurs.