# CondMixTest: Maximum Likelihood Test for a Mixture Effect

Jeff Miller
Department of Psychology
University of Otago
Dunedin, New Zealand

April 25, 2018

## Contents

## 1 Overview

This document describes `CondMixTest`, a routine that can be used to compute a likelihood ratio test to see whether the difference between two conditions is a "uniform effect" or a "mixture effect", as is described further below (see Miller, 2006, for more details). The data from the two conditions can be any form of numerical measurements. As in the other `RawRT` routines, `CondMixTest` does the analysis separately for each combination of conditions indicated by a set of `CondSpecs`.

## 2 Detailed Introduction

What does this program do? This section describes the basic statistical scenario and defines some terminology.

Suppose a researcher collects numerical scores from two conditions and finds that the two conditions have significantly different means. To make the scenario a little more concrete, call the conditions "experimental" and "control", and suppose the mean is larger in the experimental condition.

Intuitively, the difference in means might reflect either a "uniform effect" or a "mixture effect", defined as follows: With a uniform effect, all of the scores in the experimental condition are increased relative to what they would have been in the control condition. With a mixture effect, however, only some of the scores in the experimental condition are affected; the rest of the scores in this condition are the same as they would have been without the manipulation (i.e., the same as they would have been in the control condition).

To take a traditional sort of example, suppose we want to evaluate the effectiveness of adding a certain chemical to a standard fertilizer. 100 corn plants are given the standard fertilizer as a control group, and 100 are given the standard fertilizer enhanced with the added chemical as an experimental group. Average productivity turns out to be higher for the experimental plants than the controls.

One interpretation of the higher scores for the experimental condition is that the added chemical facilitates growth of all corn plants, so productivity tends to be larger for each plant in this group than it would have been without the chemical. This would be an example of a uniform effect.

Another possibility, however, is that the added chemical facilitates growth for only a proportion of the plants, with other plants being completely unaffected by it (perhaps because of some genetic variation). This would be an example of a mixture effect.

Looking only at the means, there is no way to decide whether there was a uniform effect or a mixture effect. You might be able to see that there was a mixture effect by looking more closely at the observed distributions of scores, however. For example, you might find a bimodal frequency distribution in the experimental group; one mode might match the mode of the control group, and the other mode might be larger. In that case, you would have strong evidence of a mixture effect. But looking for a bimodal distribution isn't likely to be the most powerful approach, because (a) it requires a lot of data, and (b) it can only work for mixture effects that produce distinct modes.

`CondMixTest` tests between uniform and mixture effects with a likelihood ratio test taking into account the full distributions of scores in the two conditions. In brief, it works by fitting two different models to the data—one corresponding to a uniform effect, and the other corresponding to a mixture effect. These two models will be called the "uniform model" and the "mixture model" throughout this documentation.

Each model is fit via maximum likelihood estimation of its parameters. Roughly speaking, the likelihood of a data set under a given model is the product of the probabilities (under that model) of all of the actual observations. So, maximum likelihood estimation adjusts the model parameters so that the model assigns the highest likelihood that it can to the actual observed data set.

Once each model has been fit by maximum likelihood estimation, a chi-square likelihood ratio test is used to decide whether the mixture model fits significantly better than the uniform model. In brief, this test just looks at whether the data are significantly more likely under the mixture model than under the uniform model. If so, the researcher can conclude that there is a mixture effect. If not, the null hypothesis of a uniform effect model cannot be rejected. The uniform model is simpler, so it can never fit better than the mixture model.

The above description sounds fairly simple, but carrying out this sort of mixture analysis in practice is not simple at all, for a variety of reasons. Potential users wanting to gain an appreciation of the complications involved in this type of analysis should look at the article by Reynolds and Miller (2009) for a real-research example that illustrates some of the complications that must be considered in using `CondMixTest` with real data.

I will next get a bit more technical and introduce a little terminology which is used throughout the remainder of this documentation.

## 2.1 Control Condition

Both models assume that there is some probability distribution for the observations in the control condition, call it $f_c$. For example, you might believe that the observations in the control condition have a normal distribution, so $f_c(x)$ is the function giving the normal curve over the region of $x$'s (data values) of interest.

## 2.2 Experimental Condition, Uniform Model

According to the uniform model, there is some other probability distribution for the observations in the experimental condition, call it $f_{eu}$. For example, if you thought the control distribution was normal, you would usually think that the observations in the experimental condition also had a normal distribution, although with a different mean and possibly a different standard deviation too. This other distribution is $f_{eu}$.

### 2.3 Experimental Condition, Mixture Model

According to the mixture model, there are two possibilities for each observation in the experimental condition. Some experimental observations are unaffected by the manipulation, so they simply come from the control distribution, $f_c$. The rest of the experimental observations are affected by the manipulation, however, so they come from some other probability distribution, $f_{em}$, representing the distribution of *affected* scores. (I will sometimes refer to this as the "effect-present distribution".) For example, if you thought the control distribution was normal, you would usually think that the affected observations in the experimental condition also had a normal distribution, although with a different mean and possibly a different standard deviation. Finally, the mixture model has one extra parameter, which is the "effect probability", $P$. This is the probability that an observation in the experimental condition is actually affected by the experimental manipulation—i.e., the probability that it comes from the effect-present distribution rather than the control distribution. Thus, $1 - P$ is the probability that the observation is not affected, so it comes from the control distribution.

### 2.4 The Likelihood Ratio Test

In brief, `CondMixTest` computes a likelihood ratio test to compare the fits of two models to the data from the experimental and control conditions (see Miller, 2006, for more details). Both models have the distribution $f_c$ in the control condition. The uniform model says that the data in the experimental condition come from some other distribution $f_{eu}$, whereas the mixture model says that these data are a mixture of $f_c$ and $f_{em}$. `CondMixTest` tries various parameter values for these models and finds the ones that give the maximum likelihood for your data. Using those maximum-likelihood parameter estimates, it then computes a likelihood ratio test to evaluate whether the more flexible mixture model fits significantly better than the simpler uniform model.

## 3 Setting up the Analysis

This section describes the commands needed to perform the mixture test analysis. For examples, see `DemoMixTest.m`.

An example of a series of commands for a mixture test analysis might look like this:

```
sDV = 'RT';    % Dependent variable affected by condition.
CondSpecs = {'SubNo'};   % Do analysis separately for combinations of these.
FactorName = 'Cond';     % The factor whose effect is being tested for mixture
    .
FactorLevels = [1 2];    % The levels of the factor defining the effect.
ControlDist  = ExGauMn(200,20,200);  % Initial guess for distribution in
    control condition.
ExptlUniDist = ExGauMn(250,25,100);  % Initial guess for distribution in
    experimental condition under uniform model.
ExptlMixDist = ExGauMn(300,30,100);  % Initial guess for ``effect present''
    distribution in experimental condition under mixture model.
StartingEffectP = .6;     % Initial guess as to the probability that the
    effect is present
SearchOptions = optimset('MaxFunEvals',$10^7$,'MaxIter',$10^6$);
[outResultTable, outDVNames] = CondMixTest(Trials,sDV,CondSpecs,FactorName,
    FactorLevels, ...
   ControlDist,ExptlUniDist,ExptlMixDist,StartingEffectP,'SearchOptions',
       SearchOptions,'Verbosity',3);
```

`Trials` is a trials table in the same format as all of the other `RawRT` routines.

`sDV` is a string indicating the name of the dependent variable within the trials table.

`FactorName` and `FactorLevels` define the effect for which a mixture explanation is being tested. `FactorName` is the name of a variable in the trials table; this variable codes the levels of the factor. `FactorLevels` indicates the levels of that factor defining the effect. That is, the effect being tested as a possible mixture compares the trials with `Trials.FactorName==FactorLevels(1)` against the trials with `Trials.FactorName==FactorLevels(2)`. In all examples, I will just consider factors with 2 levels, 1 and 2, and I will code them so that the values of the DV are smaller at level 1 than at level 2.

`ControlDist`, `ExptlUniDist`, and `ExptlMixDist` specify the distributions to which the values of the dependent variable should be fit. Any of the distributions described in the CUPID documentation can be used with `CondMixTest`. The parameters specified for these distributions provide the starting values, and `fminsearch` will try to adjust these values to find the maximum likelihood fits, separately for both the uniform and mixture models. Note that the same control distribution is used for both the uniform model and the mixture model, although the parameter values are estimated separately for the two models. The `ExptlMixDist` distribution is the one used in fitting the effect-present trials in the experimental condition within the mixture model. The unaffected trials in this condition are of course fit to the `ControlDist` distribution in the mixture model.

In most applications that I can think of, I would expect the same distributional family (e.g., normal, ex-Gaussian, etc) to be used for all three types of fits (i.e., control, experimental within uniform model, effect-present experimental within mixture model). The program does not require this, however. The chi-square test and associated $p$ value are only meaningful, however, if the total number of parameters in the mixture model (counting MixP) is exactly one more than the number of parameters in the uniform model.

`StartingEffectP` provides the starting value for the probability that the effect is present within the mixture model. `fminsearch` will also try to adjust this value to find the maximum likelihood fit under the mixture model. For example, a `StartingEffectP` of 0.35 indicates that the search should start with a mixture model in which the effect is present 35% of the time. Of course the final value at the end of the search is whatever value maximizes likelihood. To prevent division by zero, you may not use a `StartingEffectP` value of 0 or 1. Also, for technical reasons that I do not understand very well, the `fminsearch` seems especially likely to get caught in a local minimum if you use `StartingEffectP`=0.5, so I strongly recommend avoiding this value.

A little more explanation of the concept of "starting values" may be useful. A well-known problem with iterative parameter search routines, like `fminsearch`, is that they can get stuck at local minima—parameter combinations that are best within a certain region of the parameter space but not best overall. For that reason, it is pretty important that your starting values should be good ballpark guesses as to what the true values of the parameters might be. For example, if you are assuming normal distributions, then you should pick starting values for its $\mu$ and $\sigma$ that would be realistic for the type of data you actually have. Similarly, you need to provide a reasonably good guess about the probability that the effect is present. It doesn't matter *exactly* which starting values you choose. The maximum-likelihood parameter search routine will always try to adjust the parameters from the starting values that you give it, in order to find parameters that give an even better fit to your data. So, for example, it probably won't matter whether you start with $\mu = 100$ and $\sigma = 20$ or $\mu = 95$ and $\sigma = 18$; these are similar enough that the parameter search routine fill find the same maximum-likelihood point from either one of them. But the parameter search routine can get stuck in a local minimum if your initial guesses are really far off the mark, so you need to have reasonably good starting values. For example, if you started with $\mu = 0$ and $\sigma = 1$ where something like the actual values were more like 100 and 20, then there is a good chance that the parameter search routine would not converge on the best values.

One way to combat the local minimum problem is to try starting the parameter search from different points in the parameter space, hoping that it will find the real overall minimum from at least one of the starting points. This is especially important to the extent that you are unsure of the starting values, because trying many different ones increases your chances of finding the best fit. Within `CondMixTest`, it is also very important if you are trying to compute mixture tests in many different combinations of conditions (e.g., for many different experimental participants), because you might need different starting values for different combinations. The only way to do this automatically is to try lots of different starting values for everybody.

To facilitate the process of trying multiple starting points, `CondMixTest` allows you to specify multiple starting points like this:

```
% Try multiple search starting points for each of these distributions:
ControlDist  = [ {ExGauMn(200,20,200)} {ExGauMn(300,30,100)} ];
```

```
ExptlUniDist = [ {ExGauMn(250,25,100)} {ExGauMn(350,35,100)} {ExGauMn
    (200,20,150)} {ExGauMn(300,30,150)}  ];
ExptlMixDist = [ {ExGauMn(300,30,100)} {ExGauMn(400,40,100)} {ExGauMn
    (200,20,200)} {ExGauMn(300,30,200)}  ];
% Try multiple starting points for the probability that the effect is present:
StartingEffectP = [.5 .7 .9];
```

For each of the three required distributions, you can provide a cell array of two or more distributions rather than a single distribution, as shown, and each of the distributions in the cell array is used as a different starting point for that distribution. For this example, when fitting the control condition for the uniform model, `CondMixTest` would start the search once parameter values (200,20,200) and once with (300,30,100), and it would use the better of the two finishing points as its best guess for the global minimum. Similarly, when fitting the experimental condition for the uniform model, it would start with four different parameter values and takes the best of all four finishing points that it finds. Things are a bit more complicated when fitting the mixture model, because this model depends on the control distribution, the effect-present experimental distribution, and the mixture probability, so `CondMixTest` tries all possible combinations of starting values. When fitting the mixture model for this example, `CondMixTest` would actually try 24 different starting points defined by all the combinations of `ControlDist`, `ExptlMixDist`, and `StartingEffectP` (i.e., $2 \times 4 \times 3 = 24$), using the best finishing point across all 24. You can use as many starting point as you like; the only cost is computational time.

The final two input parameter pairs are optional. `'SearchOptions',SearchOptions` is used to pass a structure of settings to `fminsearch`. If this pair is omitted, `fminsearch` is called with its default settings. But you can override the default settings by creating a structure of alternative settings using MATLAB's `optimset` function, as in this example:

`SearchOptions = optimset('MaxFunEvals',`$10^7$`,'MaxIter',`$10^6$`);`

You can use any `optimset` options you like. I always increase the default maximum numbers of function evaluations and iterations so that `fminsearch` will work longer and have a better chance of finding the true global minimum—the only cost is computer time.

The parameter pair `'Verbosity',iVerb` can be used to request more or less progress reporting output from `CondMixTest`. `iVerb` is an integer 0–3, with higher integers giving more verbose output. Experiment to see what the options do (the default is 1).

## 4 Results

The main results of the `CondMixTest` command are stored in a MATLAB table similar to that shown in Table 1. Each line of the table holds the results of the analysis for one combination of the variables specified in `CondSpecs`.

The left-most variable(s) on each line indicate the condition code(s) for the current line (here, just `SubNo`).

The next set of variables (moving left to right) is the set of parameter estimates for the control condition, uniform model, with one variable for each parameter of the model. In this example, there are three parameters corresponding to the three parameters of the ExGauMn distribution: normal $\mu$, normal $\sigma$, and mean of the exponential.

Similarly, the next set of variables is the set of parameter estimates for the experimental condition, uniform model.

The next variable, `UniLnLik`, is the overall log-likelihood of the data set under the uniform model.

`EffectP` is the maximum-likelihood estimate of the effect-present probability $P$ under the mixture model.

The next two sets of variables are the parameter estimates for the control and experimental conditions under the mixture model.

To elaborate a little bit on the meanings of these parameter estimates, those for the experimental condition correspond to the estimates for the distributions $f_{eu}$ and $f_{em}$ within the uniform and mixture models, respectively. The estimates for the control distribution are estimates for $f_c$, but note that these estimates differ across the two models. In the uniform model, the estimates of $f_c$ are influenced only by the scores in the

Table 1: Example results table produced by `DemoMixTest.m`. There are too many variables in the table to fit in the width of a single page, so the ... markers indicate continuations of previous lines.

```
SubNo  CntrlUni_mu  CntrlUni_sigma  CntrlUni_exmean
-----  -----------  --------------  ---------------
1      197.86       14.917          199.65
2      202.63       9.7261          218.91
3      207.86       30.137          185.45
4      208.98       28.399          208.49
5      188.48       7.9285             205

...  ExptlUni_mu  ExptlUni_sigma  ExptlUni_exmean  UniMaxLnLik
...  -----------  --------------  ---------------  -----------
...  277.01       58.953          193.82           -1288.7
...  270.56       42.879          220.29           -1299.2
...  263.55       67.944          219.77           -1301.7
...  260.21       47.473          223.12             -1305
...  279.17       60.378           176.9             -1282

...  EffectP  CntrlMix_mu  CntrlMix_sigma  CntrlMix_exmean
...  -------  -----------  --------------  ---------------
...  0.71779  196.93            13.26      200.65
...   0.7311  203.48           9.0452       218.3
...   0.6412   200.4           27.275      192.76
...  0.75402  191.62           10.348      225.34
...  0.68969   182.2      6.3243e-12      214.97

...  ExptlMix_mu  ExptlMix_sigma  ExptlMix_exmean  MixMaxLnLik  ChiSq       p
...  -----------  --------------  ---------------  -----------  -----   ----------
...  308.58       2.4481e-11      190.95           -1281.8      13.91   0.00019094
...  292.64       1.6867e-10      223.27           -1296.6      5.156     0.023162
...  296.93       2.5578e-11      237.07           -1299.1      5.222       0.0223
...  280.51       5.3815e-11      225.15           -1300.9      8.275     0.004019
...  312.97        0.0014658      164.24           -1275.3      13.34   0.00025925
```

control condition. In the mixture model, the estimates of $f_c$ are influenced by the scores in the experimental condition as well.

`MixLnLik` is the overall log-likelihood of the data set under the mixed model.

`ChiSq` is the chi-square value of the likelihood ratio test.

Finally, the variable `p` is the attained significance level of the observed `ChiSq` value, judged against a chi-square distribution with one degree of freedom. (The mixture probability is the one extra parameter in the mixture model, corresponding to the one df of the chi-square test.) If $p < .05$ (or whatever your significance cutoff is), then the mixture model fits significantly better than the uniform model.

The right-most two columns in the table are the log-likelihood values corresponding to the best-fitting uniform and mixture models, respectively. Each of these is the natural log of the likelihood of the data, under the particular model. Note that larger values indicate higher likelihood (i.e., better fit to the model). If you analyze the same data set in several different runs (e.g., with different starting parameter values), you should use as the "final" parameter estimates the run that gives the largest (i.e., least negative) log-likelihood value across all of the different runs for that data set.

In addition to the table illustrated in Table 1, `CondMixTest` also returns two further outputs that might be of use to a few users. The first is a list of the names of the variables in `outResultTable`. The second is a data structure that could be called `SearchReports`. This is an elaborate data structure with reports about the ending points of searches with different starting points. You can examine it to see how well the searches converged on the same best parameter values from different starting points, which may give you some insight into the extent of problems with local minima.

## 5 Precautions: Distributional Assumptions and Outliers

A limitation of `CondMixTest` is that the researcher has to specify the probability distribution family (e.g., normal, lognormal, gamma) of the observations. In some cases, the correct family may be known on a priori grounds or from extensive prior research in the area. If not, the researcher must examine the control condition data carefully to try to determine a reasonable distributional family for this condition before starting `CondMixTest`. Note that it is probably not a good idea to examine the experimental condition data in the same fashion, because you don't know when you start whether this condition should be fit by a single family or by a mixture.

Another important limitation of `CondMixTest` is that it is rather sensitive to the presence of outliers. It is thus extremely important to check the data carefully for outliers, and exclude any that are found. If there are some identifiable points that may or may not be outliers—you cannot tell for sure—then it would be prudent to run `CondMixTest` twice (i.e., once with the possible outliers and once without them) and to interpret the results only if the two runs lead to the same overall conclusion.

## 6 Release History

- First MATLAB release in RawRT: April 2018.
- Prehistory: some executable versions of the Pascal program were released in 2005–2007.

## References

Miller, J. O. (2006). A likelihood ratio test for mixture effects. *Behavior Research Methods*, 38(1):92–106.

Reynolds, A. and Miller, J. O. (2009). Display size effects in visual search: Analyses of reaction time distributions as mixtures. *Quarterly Journal of Experimental Psychology*, 62(5):988–1009.