

Instructions for use of the `grtfit` General Recognition Theory parameter estimation routine

Noah H. Silbert

Most recently updated on December 1, 2008

1 General Recognition Theory

General Recognition Theory (GRT; Ashby and Townsend, 1986, *inter alia*) is a multidimensional generalization of Signal Detection Theory (SDT; Green and Swets, 1966, *inter alia*). As such, it is a two stage model of perception and decision making¹. In GRT, it is assumed that the perceptual effect of stimulus presentation is, at least in part, random. The cumulative effect is modeled as a set of perceptual distributions defined on a perceptual space. It is also assumed that the perceptual space is exhaustively partitioned so that every possible perceptual effect corresponds to one (and only one) response option.

In the most general form of GRT, no assumptions about the function form of the perceptual distributions or decision bounds are made. While this allows for some very general tests of dimensional interaction, a more detailed picture can, in theory, be obtained via parameter estimation, which requires such assumptions. The parameter estimation algorithm `grtfit_st` (and `grtfit`) assumes that perceptual distributions are bivariate Gaussian, and that decision bounds are linear, piecewise linear, or based on likelihood ratios between the perceptual distributions (more on this later).

The algorithm also assumes that dimension interactions are what you're after². In fact, it assumes even more than this, namely four stimuli (or stimulus categories) and responses defined by the factorial combination of two levels on each of two dimensions (e.g., short and long duration, low and high frequency). That is, `grtfit_st` estimates parameters for four bivariate perceptual distributions and determines four response regions in a two dimensional space³. This 'two levels on each of two dimensions' stimulus-response structure will be re-

¹SDT and GRT can both also be applied to the study of memory (e.g., DeCarlo, 2002), though I haven't personally done so. In the interest of clarity, I will refer exclusively to perception in this document.

²There are two main lines of research in the GRT framework, one on dimension interaction (in which I participate), another on categorization (e.g., Ashby and Gott, 1988). In the former, the theory is typically referred to as GRT or MDSDT (multidimensional SDT); in the latter, it is usually called either GRT or decision bound theory

³`grtfit` uses more response regions; see section 6

ferred to herein as the 'standard' GRT task (the `_st` bit in `grtfit_st` means 'standard' in this sense).

Before describing the use of `grtfit_st` to probe interactions between dimensions, it is necessary to say more precisely what is meant by 'interactions between dimensions'. First and foremost, interaction here indicates the opposite of independence. If two dimensions are independent, they do not interact; if two dimensions interact, they are not independent.

1.1 Varieties of Independence

In GRT, there are three logically distinct ways in which dimensions may interact or be independent. This issue is usually cast in terms of (types of) independence, and I will discuss it using the terminology common to the GRT literature.

There are two main distinctions to keep in mind. First, there is the distinction between perceptual and decisional forms of independence. Second, with regard to perceptual relationships between dimensions, there are within-stimulus and between-stimuli notions of independence.

The within-stimulus notion of perceptual independence is called, perhaps not surprisingly, **perceptual independence**. Perceptual independence holds in a given perceptual distribution if, and only if, for all values of x and y , stochastic independence holds. That is, if, and only if, $f_{A_i B_j}(x, y) = g_{x, A_i B_j}(x)g_{y, A_i B_j}(y)$, where $f_{A_i B_j}(x, y)$ is the joint bivariate density for the stimulus at level i on dimension A and level j on dimension B , and $g_{x, A_i B_j}(x)g_{y, A_i B_j}(y)$ is the product of the marginal densities at the appropriate levels on the appropriate dimensions. Note that the definition of perceptual independence does not rely on any particular type of density. Note also that, if you are willing to assume that the perceptual distributions are Gaussian, then perceptual independence holds if, and only if, covariance (or correlation) is zero.

The between-stimuli notion of perceptual, um, non-interaction is called **perceptual separability**. Perceptual separability holds for a pair of distributions that share a level on one dimension if the marginal densities on the shared dimension-level are equal across levels of the other dimension. Thus, for example, perceptual separability holds if $g_{x, A_1 B_1}(x) = g_{x, A_1 B_2}(x)$. That is, it holds if the perceptual effect on the x-axis (A dimension) is identical regardless of the level on the y-axis (B dimension). Although the definition permits us to do so, it is somewhat atypical to discuss perceptual separability with regard to a pair of stimuli. It is much more common to discuss perceptual separability with regard to *dimensions*. In this case, perceptual separability holds if, and only if, the definition given above holds for all levels on a given dimension. Still assuming the standard GRT setup, it holds if $g_{x, A_1 B_1}(x) = g_{x, A_1 B_2}(x)$ and $g_{x, A_2 B_1}(x) = g_{x, A_2 B_2}(x)$. The definitions given here all pertain to dimension A being (or failing to be) perceptually separable from dimension B . With the appropriate changes to subscripts, the same basic definition applies to perceptual separability of dimension B from dimensions A .

Finally, the decisional notion of independence is called **decisional separability**. Decisional separability holds if decisions about one dimension are not

affected by perception or decision making on the other dimension. That is, if, and only if, a decision bound is parallel to the appropriate coordinate axis.

All three forms of independence and examples of failures of each are illustrated in Figure 1, which shows a Gaussian GRT model with linear bounds for a space defined in terms of shape (square or rectangle) and hue (red or purple). Perceptual independence holds for the two distributions on the left (i.e., the ‘square’ distributions). Perceptual independence fails for the two distributions on the right; the ‘red rectangle’ distribution has negative covariance, whereas the ‘purple rectangle’ distribution has positive covariance.

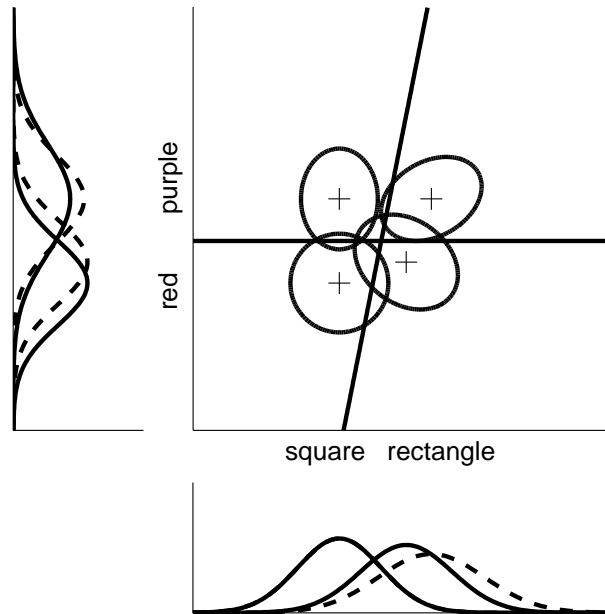


Figure 1: Illustrative two dimensional Gaussian GRT model. The ellipses represent contours of equal likelihood; the plus signs indicate the means of the distributions. The solid line marginal densities correspond to the first level on the other dimension (e.g., red or square); the dashed line marginals correspond to the second level (e.g., purple or rectangle). See text for details.

Perceptual separability holds on the shape dimension for the two ‘square’ distributions; the marginal shape densities (for squares) are identical across levels of hue. Perceptual separability fails for each of the other three pairs of marginal densities. It fails on the hue dimension for the ‘red’ marginals because the hue mean for the ‘red rectangle’ distribution is shifted up relative to the hue mean for the ‘red square’ distribution. It fails on the hue dimension for the ‘purple’ marginals because the hue variance is larger for the ‘purple square’ distribution than for the ‘purple rectangle’ distribution. Finally, it fails for

the ‘rectangle’ shape marginals because both the marginal shape means and variances differ across levels of hue.

Decisional separability holds for the hue decision bound, whereas It fails for the shape decision bound.

2 Data

The data for a standard GRT task consist of identification-confusion frequencies. Since there are four stimuli and four (matching) responses, the data are arranged in a four-by-four confusion matrix with rows corresponding to stimuli and responses corresponding to responses. The default arrangement of rows and columns is *aa*, *ab*, *ba*, *bb* (e.g., low frequency, short duration; low frequency, long duration; high frequency, short duration; high frequency, long duration). You should either make sure that your confusion matrices are arranged in this (or a suitably parallel) manner.

3 The Algorithm

`grtfit.st` is, in essence, a hill climbing algorithm. When the function is called, you provide it with, among other things, an observed confusion matrix, an initial set of model parameters, a specification of which parameters are free to vary, and the number of improvements in model fit that you want the algorithm to find. On each step, `grtfit.st` takes the ‘old’ set of parameters (on the first step, this is just the initial parameters you provide) and adds a bit of random noise to each free parameter to get a new set. It evaluates the fit with the new set of parameters and, if the fit is better for the new parameters than it is for the old parameters, it accepts (i.e., saves) the new set and then uses it as the old set for the next step.

If the fit is worse (or equal) for the new set than for the old set, new random noise is added to the old parameters and the fit is again evaluated. If no improvement in fit is obtained for 10 tries in a row, the magnitude of the random noise is reduced, and the old parameters continue to be used as the old parameters for another 10 tries. The idea is that 10 failed attempts to improve the fit suggests that you’re stuck in either a local or the global minimum, so you should be sure that you’ve explored that region of parameter space thoroughly before giving up on it. If no improvement in fit is obtained after the next 10 tries, the magnitude of the random noise is increased and the new set of parameters is accepted, regardless of the relative fit. The idea here is to escape from local minima and find a new local, or with a little luck, the global minimum.

4 The Function

A note on notation: The levels on each dimension for each stimulus (and perceptual distribution) will be specified by a two-character string where the left

character indicates the x-dimension and the right character indicates the y-dimension; *a* indicates the first level, *b* the second. Thus, the stimulus at the first level on each dimension (e.g., the low frequency, short duration tone) is indicated by *aa*, the stimulus at the first level on the x-dimension (here frequency) and second on the y-dimension (here duration) is indicated by *ab*, etc...

4.1 .m files

First of all, in order to use `grtfit_st`, you need to have the following Matlab functions:

`grtfit_st.m` is the function you call to run the algorithm; it contains calls to `bivar_norm.m` and `dec_reg.m`.

`bivar_norm.m` provides numerical approximations to four bivariate Gaussian densities.

`dec_reg.m` defines the decision regions; if the decision bounds are linear or piecewise linear, it contains a call to `bound_spec.m`.

`bound_spec.m` defines the decision bounds for the linear and piecewise linear decision models.

4.2 Inputs and Outputs

A call to `grtfit_st.m` returns three output arrays and requires seven input arrays.

4.2.1 Outputs

The output arrays are:

`fitps` is a structure with fields for each parameter and values for each field for each accepted set of parameters, so if you call for 500 sets, `fitps` is a 1x500 structure (i.e., it contains 500 sets of parameters, each of which either fit better than the last set or was accepted after a big jump taken after 20 failures to improve the fit).

`best_fitps` is the best fitting set of parameters from `fitps`

`best_pM` is the predicted confusion matrix for the parameters in `best_fitps`.

In case you haven't used structures in Matlab, they have fields that can be a subset of the available arrays (e.g., matrix, string, scalar). `fitps` has the following fields when fitting a model with linear decision bounds:

`maa`, `mab`, `mba`, & `mbb` are all 2x1 vectors containing the *x* and *y* means of the `aa`, `ab`, `ba`, `bb` densities, respectively. The (1,1) element is the *x* mean, the (2,1) element the *y* mean.

`caa`, `cab`, `cba`, & `cbb` are 2x2 covariance matrices for the `aa`, `ab`, `ba`, `bb` densities, respectively. The (1,1) element is the x variance, the (2,2) element is the y variance, and the (2,1) = (1,2) elements are the covariance.

`cx`, `dx`, `cy`, `dy` are the decision bound parameters. With linear decision bounds, `cx`, `cy` indicate slopes, `dx`, `dy` intercepts.

`LL`, `g2`, & `bic` are fit statistics. `LL` is the log likelihood, `g2` is the likelihood ratio statistic G^2 , and `bic` is the BIC, discussed and defined below in section 4.3.1.

Typing, e.g., `fitps(23).caa` would give you the covariance matrix for the ‘low-low’ stimulus for the 23rd step of the algorithm.

4.2.2 Inputs

The required inputs are:

`oM` is the 4x4 observed confusion matrix, containing counts (i.e., raw frequencies, not relative frequencies or estimated probabilities).

`inits` is a 1x1 structure containing the initial set of (old) parameters used in the algorithm (often the best fitting parameters – `best_fitps` – from a previous run; it must have the same fields as `fitps` and `best_fitps`).

`fp` is a structure with fields that take logical values (i.e., 1 or 0) that indicate which parameters are free to vary.

1. The following fields determine whether a parameter is free to vary (1) or not (0).
 - `cx` = 1 allows the slope on the bound partitioning the y-axis to vary (this field is called `cx` because it determines the slope of a linear function of x). `cx` = 0 forces the slope on this bound to be zero (i.e., decisional separability holds). *Mutatis mutandis* for `cy`.
 - `dx` = 1 allows the intercept of the bound partitioning the y-axis to vary. `dx` = 0 fixes the intercept at the mean value of all of the marginal y-axis means. *Mutatis mutandis* for `dy`.
 - `mab` = 1 allows the mean vector for the distribution at the low level on the x-axis and high level on the y-axis to vary; `mab` = 0 fixes the mean vector at the initial value provided as input. *Mutatis mutandis* for `mba` (high-low) and `mbb` (high-high).
 - `raa` = 1 allows the correlation (covariance) of the low-low distribution to vary; `raa` = 0 fixes it at zero. *Mutatis mutandis* for `rab` (low-high), `rba` (high-low), and `rbb` (high-high).

2. The following fields determine whether or not sets of parameters will be forced to be equal or not. Note that, in `grtfit_st`, this only affects means, since shifts in means vs. variances are not identifiable in the ‘standard’ GRT experiment. Thus, for example, forcing x means to be equal across levels of y is equivalent to forcing perceptual separability to hold (at the given level of x).

- `mxa = 1` forces the means at the low (a) level of the x-axis to be equal (i.e., the x means for the low-low and low-high distributions). `mxa = 0` allows these means to be different. *Mutatis mutandis* for `mxb` (x means at the high level), `mya` (y means at the low level), and `myb` (y means at the high level).

`nfits` is the number of fits wanted (i.e., `fitps` will be of size 1-by-`nfits`).

`dbtype` is a string, and must either ‘`linear`’ or ‘`maxpost`’, depending on which decision bound model you want to fit. In the ‘`maxpost`’ model, responses are determined by which perceptual distribution has the maximum posterior probability of having produced any given perceptual effect, so the form of ‘`maxpost`’ decision bounds is determined by the shape and location of the perceptual distributions. In general, they are piecewise quadratic curves. The form of ‘`linear`’ bounds is obvious.

`mod_id` is a string indicating which model the algorithm is working on (useful for you, the user, just so you can see how much progress has been made in, say, the last day or so).

4.3 Function Calls

4.3.1 How

A generic call to `grtfit_st`, then, looks like:

```
[fitps, best_fitps, best_pM] = grtfit_standard(oM,inits,fp,nfits,dbtype,mod_id);
```

A more realistic version could look like:

```
[nhs_PI,nhs_min_PI,nhs_PIM] = grtfit_st(nhs_oM,inits,fp_rvm,500,'linear','nhs_PI');
```

and produces regular updates that look like:

```
N free parameters = 8
nhs_PI
working on fit 0 out of 500
complexity = 39.9694
bic_min = 453.2116
bic_old = 453.2116
```

```

bic_new = 453.2116
n_rejected = 1

nhs_PI
working on fit 1 out of 500
complexity = 39.9694
bic_min = 453.2116
bic_old = 453.2116
bic_new = 456.9749
n_rejected = 2

...

```

On the first step, it tells you how many free parameters your model has (important with standard GRT, since the model in its most general form has more free parameters (14 or so, depending on the type of decision bound) than the data do degrees of freedom (12), so the data cannot constrain the most general form of the model). If your model has 12 free parameters, `grtfit_st` let's you know that this isn't such a good idea, and if your model has more than 12 free parameters, it returns an error message and won't do your bidding.

On the first step, it also tells you the 'complexity' of the model. This is because the fit it evaluates is the Bayesian Information Criterion (BIC; see, e.g., Kass and Raftery, 1995), which is calculated as $BIC = k \log(N) - 2 \log(L)$, where k is the number of free parameters in the model, N is the number of observations, and L is the (multinomial) likelihood. The first term is often called 'complexity', so I call it that here. I was curious about the magnitudes of the complexity terms for the various models I have been fitting, so I made it tell me explicitly. But I digress...

On every step, it tells you which model is being fit (i.e., `mod_id`), what the best (i.e., smallest) BIC thus far is, what the 'old' BIC is (i.e., the BIC for the current 'old' set of parameters), what the 'new' BIC is (i.e., the BIC just calculated and compared to the 'old' BIC), and how many times it has failed to find an improvement in fit (i.e., `n_rejected`).

When it's done with the `nfits` steps, it returns the outputs discussed above in section 4.2.1. You decide what to do next. One possibility is to use `grt_plotfit.m` to produce stunningly beautiful graphical representations of your model fits.

4.4 Testing independence and separability

The logic behind using `grtfit_st` to test for (the absence of) these three types of independence is fairly straightforward. Fit a model that exhibits a particular form of independence, and fit a model that allows that form of independence to fail, and compare the fit of the two models. Whichever model has the lower BIC wins. BIC is a reasonably good fit statistic to use for this kind of test, since it takes into account both goodness of fit (the $-2 \log(L)$ term) and the complexity of the model (the $k \log(N)$ term). A model that allows some form of

independence to fail will always have more free parameters than a model that forces that form of independence to hold, so the complexity term will be larger for the former than for the latter. The extra parameters will enable that model to fit better, though, so the fit term will, in general, be smaller for the former than for the latter. The relative magnitude of these differences determines the winner.

Here are some more concrete examples. Say you want to test perceptual independence. You would fit a model that forces perceptual independence to hold (i.e., that makes the covariance for each perceptual distribution zero), and you would fit a model that allows perceptual independence to fail (i.e., that lets the covariances vary freely). Similarly for perceptual and/or decisional separability.

Recall that these three forms of independence are logically distinct. This means that they can hold or fail regardless of whether the others hold or fail. The upshot is that a thorough `grtfit_st` treatment of a data set requires fitting a large number of models - one for each combination of each type of independence holding or failing. In fact, it's a bit more complicated than that, as, for example, perceptual separability may hold for one dimension and not the other, or it may hold for both.

This brings us back to the number of free parameters relative to the degrees of freedom in the data. Recall that the data collected in a standard GRT task has 12 degrees of freedom. The most general Gaussian GRT model with linear bounds has two means, two variances, and a covariance for each perceptual distribution ($4 * (2 + 2 + 1) = 20$) and a slope and intercept for each decision bound ($2 * 2 = 4$), which gives us twice as many free parameters as degrees of freedom. We can (and do) fix the means of the 'low-low' distribution at (0,0), and it turns out we can ignore variances and let shifts in the means of the other three distributions do the work, so that reduces the number of free parameters to 14, which is still two too many.

One easy, and common, way to reduce the number of free parameters still further is to assume that decisional separability holds (i.e., that the slope parameters are zero). This gives you exactly 12 free parameters, which is okay, but not ideal. Assuming perceptual independence fixes four otherwise free (covariance) parameters, and assuming perceptual separability fixes four otherwise free (mean) parameters.

More soon (December 1, 2008)

5 Using `grt_plotfit.m`

5.1 Inputs and Outputs

6 Base Rate Manipulation

More soon ... (9.14.08)

References

- Ashby, F. G. and Gott, R. E. (1988). Decision rules in the perception and categorization of multidimensional stimuli. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 14(1):33–53.
- Ashby, F. G. and Townsend, J. T. (1986). Varieties of perceptual independence. *Psychological Review*, 91(2):154–179.
- DeCarlo, L. T. (2002). Signal detection theory with finite mixture distributions: Theoretical developments with applications to recognition memory. *Psychological Review*, 109(4):710–721.
- Green, D. M. and Swets, J. A. (1966). *Signal Detection Theory and Psychophysics*. John Wiley and Sons, Inc., New York.
- Kass, R. E. and Raftery, A. E. (1995). Bayes factors. *Journal of the American Statistical Association*, 90:773–795.