Behavior Research Methods
2007, 39 (2), 291-302

Testing the race model inequality: An algorithm and computer programs

Rolf Ulrich
University of Tübingen, Tübingen, Germany

Jeff Miller
University of Otago, Dunedin, New Zealand

and

Hannes Schröter
University of Tübingen, Tübingen, Germany

In divided-attention tasks, responses are faster when two target stimuli are presented, and thus one is redundant, than when only a single target stimulus is presented. Raab (1962) suggested an account of this redundant-targets effect in terms of a race model in which the response to redundant target stimuli is initiated by the faster of two separate target detection processes. Such models make a prediction about the probability distributions of reaction times that is often called the *race model inequality*, and it is often of interest to test this prediction. In this article, we describe a precise algorithm that can be used to test the race model inequality and present MATLAB routines and a Pascal program that implement this algorithm.

When participants are required to respond as quickly as possible to the onset of any target stimulus, they usually respond faster when two target stimuli are presented than when only one is presented (see, e.g., Hershenson, 1962). This gain in reaction time (RT) with redundant stimuli has been termed the *redundant signals effect* (RSE). Several theoretical and experimental studies have been conducted to unravel the causes of this phenomenon (e.g., Colonius, 1988, 1990; Colonius & Townsend, 1997; Diederich, 1995; Giray & Ulrich, 1993; Miller, 1982; Mordkoff & Yantis, 1991; Schwarz, 1989, 1994; Townsend & Nozawa, 1995; Townsend & Wenger, 2004).

Raab (1962) was the first to suggest a detailed model for the RSE, a model that is based on a simple statistical principle. According to his *race model*, each stimulus is detected separately. In trials with redundant stimuli, a response is triggered as soon as the first stimulus is detected. In this view, RT is determined by the latency of a single detection process in trials with one stimulus, whereas it is determined by the faster of two stimulus detection processes in trials with redundant signals. Because the average time of the winner in a race is usually shorter than the average detection time of each single process, this race model predicts faster RTs in trials with redundant signals than in trials with only one stimulus.

Additional tests of the race model can be carried out at the level of RT distributions. More specifically, according to race models, the observed RT distributions should satisfy, for every value of t, the so-called *race model inequality* (Miller, 1982):

$$F_z(t) \leq F_x(t) + F_y(t), t > 0, \quad (1)$$

where F_x and F_y are the cumulative density functions (CDFs) of RT in the two single-stimulus conditions C_x and C_y, respectively, and F_z is the CDF of RT in the redundant-stimulus condition C_z. According to race models, $F_z(t)$ may approach $F_x(t) + F_y(t)$ for small values of t, especially when the detection times are strongly negatively correlated (Colonius, 1990; Ulrich & Giray, 1986). Yet, even in this case, the inequality must be satisfied according to race models. Contrary to this prediction, observed RT distributions often violate the race model inequality for small values of t (see, e.g., Diederich & Colonius, 1987; Giray & Ulrich, 1993; Miller, 1982, 1986; Plat, Praamstra, & Horstink, 2000).

Distributional tests using the race model inequality often show that the observed redundancy gain is actually larger than the race model can predict. More specifically, for small values of t, the CDF of RTs observed in redundant trials, $F_z(t)$, is often greater than the sum of the single-stimulus CDFs, $F_x(t) + F_y(t)$. Therefore, it has been suggested that the units of information from the redundant stimuli are somehow combined and that this combined activation triggers the response (see, e.g., Miller, 1982). Several quantitative models have been developed to describe this combination of information and the facili-

R. Ulrich, ulrich@uni-tuebingen.de

tation of RTs that results from such coactivation processes (e.g., Colonius & Townsend, 1997; Miller & Ulrich, 2003; Schwarz, 1989, 1994; Townsend & Nozawa, 1997).

The present article describes the statistical assessment of a potential violation of the race model inequality and shows how this assessment can be carried out with a computer. We provide program code in MATLAB and in Pascal that can be used to perform this assessment. This project was motivated primarily by two issues. First, there is no previously published detailed description of the algorithm for estimating the probability distributions underlying this test. Moreover, some previous descriptions apply only in the special case of equal numbers of RTs in all conditions (e.g., Miller, 1982). In fact, numerous researchers have asked the authors exactly how the test should be carried out. Second (and perhaps more alarming) is the fact that some commercial software (e.g., Excel) provides certain statistical functions for percentile estimation that might seem appropriate for testing the race model inequality, but which in fact are not appropriate.

The algorithm that is described in this article is appropriate for use with the most common test for evaluating a potential violation of Inequality 1 (see, e.g., Bucur, Allen, Sanders, Ruthruff, & Murphy, 2005; Krummenacher, Müller, & Heller, 2001). Recently, two potential alternative tests to assess this inequality have been suggested by Van Zandt (2002) and by Maris and Maris (2003). Van Zandt suggested that a bootstrapping procedure be used to estimate a confidence interval for the sample estimate $D(t) = F_z(t) - [F_x(t) + F_y(t)]$ at each value of t. Researchers could conclude that the race model inequality was violated whenever the bootstrap confidence interval indicated that $D(t)$ was significantly larger than zero. Although Van Zandt's procedure is statistically straightforward, its usefulness is limited for many typical RT studies. First, as noted by Van Zandt herself (p. 491), this procedure requires many observations per participant. Second, and perhaps more crucially, it is not clear how the bootstrapping results can be combined across multiple participants, or across multiple sessions for the same participant, if performance changes from session to session. In many RT studies, however, the number of observations per participant needs to be small because many conditions are tested, and it is therefore necessary to aggregate the results across participants in order to achieve a high level of statistical power.

Although the second test, suggested by Maris and Maris (2003), allows aggregation of data across participants, it suffers from another shortcoming that limits its application in experimental work. Their test, which is based on Kolmogorov's goodness-of-fit principle, requires that the number of trials in each single-stimulus condition be determined randomly for each single participant and condition, rather being constant across participants and conditions. This requirement is difficult to meet in most experimental work, because it also requires many trials per participant to ensure that a sufficient number of observations are available in each stimulus condition. In addition, and even more crucially, in order to control for potential confounds of stimulus contingencies (Mordkoff & Yantis, 1991), it is important that the experimenter prespecify a certain fixed number of trials for each stimulus condition rather than leaving this number to chance. The algorithm described in the following section does not require an especially large number of observations per participant, nor does it require that the number of trials be determined by chance. In addition, the algorithm allows the overall results to be aggregated across participants and displayed in a feasible manner.

ALGORITHM

In this section, we provide an exact specification of the steps necessary for testing potential violations of Inequality 1. It is convenient to break up this test into four parts. First, empirical CDFs have to be estimated for every participant and every stimulus condition, and these CDFs will be denoted as the *individual CDFs*. Specifically, let G_x, G_y, and G_z be the individual estimates of two single-stimulus CDFs, F_x and F_y, and the redundant CDF, F_z, respectively. Second, from G_x and G_y, one computes the bounding sum $B(t) = G_x(t) + G_y(t)$ for each participant, which provides an estimate for each participant of the upper bound on the right side of Inequality 1. Third, percentile values are computed from G_z and B for each participant at certain prespecified probabilities that must be the same for all participants. Fourth, the percentile values are aggregated across participants.

We will illustrate the first three steps with an example data set of a single hypothetical participant. Table 1 contains this example data set for each of the three experimental conditions.

Step 1: Compute G_x, G_y, and G_z

Because the computational steps are identical for G_x, G_y, and G_z, we will only describe the computation of G_x. Assume that a sample $\{x_1, x_2, \ldots, x_n\}$ of n RTs has been observed in condition C_x for a given participant. One orders this sample from the smallest value to the largest—that is, $x'_1 \leq x'_2 \leq \ldots \leq x'_n$—and generates a step function

Table 1
Illustrative Sets of Reaction Times for Each Experimental Condition for a Single Hypothetical Participant

Condition	n	Data Set
C_x	10	244, 249, 257, 260, 264, 268, 271, 274, 277, 291
C_y	16	245, 246, 248, 250, 251, 252, 253, 254, 255, 259, 263, 265, 279, 282, 284, 319
C_z	13	234, 238, 240, 240, 243, 243, 245, 251, 254, 256, 259, 270, 280

Note—n gives the number of observations per stimulus condition.

of G_x by plotting $p_i = i/n$ against x'_i for $i = 1 \ldots n$. For values of t smaller than x'_1, of course, p must be equal to zero. For our hypothetical participant, the ordered data set in condition C_x is

{244, 249, 257, 260, 264, 268, 271, 274, 277, 291},

and the p values

{.10, .20, .30, .40, .50, .60, .70, .80, .90, 1.0}

would be employed to generate the corresponding step function. The step function for this data set is shown as the dotted line in panel A of Figure 1. The step function estimates for G_y and for G_z are obtained in the analogous way from the observations in those conditions, and these are shown in panels B and C. Note that the number of observations need not be equal for all stimulus conditions; in practice, though, we recommend the use of at least 10 observations per condition.

Following the standard procedures for percentile estimation (see, e.g., Gilchrist, 2000), each step function is next used to generate a corresponding cumulative frequency polygon. For this purpose, the midpoint of each vertical step segment is determined as $[x'_i, (2i - 1)/(2n)]$, and the adjacent midpoints are connected by a straight line (i.e., linear interpolation). Specifically, it can be shown that for any value of t, the cumulative frequency polygon estimate $G_x(t)$ is given by Equation 2 (located at the bottom of the page). The solid lines in panels A–C of Figure 1 depict these cumulative frequency polygons for the three stimulus conditions. These values are stored, millisecond by millisecond, in the vectors G_x, G_y, and G_z in the MATLAB program shown in Appendix B, and in the vectors X.G, Y.G, and Z.G in the Pascal program in Appendix C.

The computation of a frequency polygon is more complicated when not all observations are distinct—that is, when the same value occurs more than once (a tie) within a single data set. Naturally, some ties may be observed when n is large and RT is rounded to the nearest millisecond. In this case Equation 2 no longer applies, and it thus needs to be adapted to allow for the possibility of ties. The adapted formula is presented in Appendix A. This adapted formula can be used in all cases, because it reduces to the formula in Equation 2 if no ties are present, and is incorporated in the computer programs shown in Appendixes B and C.

Condition C_z in Table 1 provides a numerical example of a data set with two tied values, and the corresponding cumulative frequency polygon for this data set is depicted in panel C of Figure 1. Although Equation 2 is a special case of the more general formula provided in Appendix A, we included this special case above to simplify the description of the complete algorithm.

Table 2
Percentiles of G_z and B As a Function of p

p	z_p	b_p
.05	234.6	244.0
.15	238.6	245.6
.25	240.4	247.3
.35	242.3	249.3
.45	244.1	250.9
.55	248.9	252.3
.65	253.9	253.6
.75	256.8	254.9
.85	265.1	257.8
.95	278.5	259.8

Step 2: Compute $B(t) = G_x(t) + G_y(t)$

In order to compute the sum of G_x and G_y, we recommend evaluating the functions G_x and G_y at each millisecond starting at $t = 1$ msec. The results of the two function evaluations can simply be added, and this sum can be stored in an array, which we will call B. Because the sum is only compared against the values of G_z, the computation of the sum can be terminated at the first value of t for which the sum exceeds a value of 1.0. Let us denote this time point as t_{max}. Thus, B has the form $[B(1), B(2), B(3), \ldots, B(t_{max})]$. The function $B(t)$ computed for the data in Table 1 is shown as the dashed line in panel D of Figure 1.

Step 3: Determine Percentiles

The third step involves the determination of percentile values for G_z and B, and many researchers may want to determine these percentile values for G_x and G_y as well, for completeness in presenting the data. It is common practice to employ n_p equally spaced probabilities of the form

$$p_i = \frac{(i - 0.5)}{n_p} \quad \text{for } i = 1 \ldots n_p. \tag{3}$$

For example, with $n_p = 10$, the values of p_i would be .05, .15, .25, . . . , .95. Let us denote the percentiles for G_z as $z_{.05}, z_{.15}, \ldots, z_{.95}$. Each value is obtained by searching through the array $G_z(1), G_z(2), G_z(3), \ldots, G_z(z'_n)$ to locate adjacent values $G_z(i)$ and $G_z(i+1)$ that bracket the desired probability, p_i, and then using linear interpolation to find the appropriate percentile value within the bracketed range. As an example, this procedure is illustrated graphically for G_y in panel E of Figure 1. The analogous procedure also has to be carried out using the array B to obtain the associated percentiles $b_{.05}, b_{.15}, \ldots, b_{.95}$. The resulting percentile arrays for the data in Table 1 are shown in Table 2.

$$G_x(t) = \begin{cases} 0 & \text{if } t < x'_1, \\ \frac{1}{n} \cdot \left(i - \frac{1}{2} + \frac{t - x'_i}{x'_{i+1} - x'_i} \right) & \text{if } x'_i \le t < x'_{i+1} \text{ and } i \ne n, \\ 1 & \text{if } t \ge x'_n \end{cases} \tag{2}$$

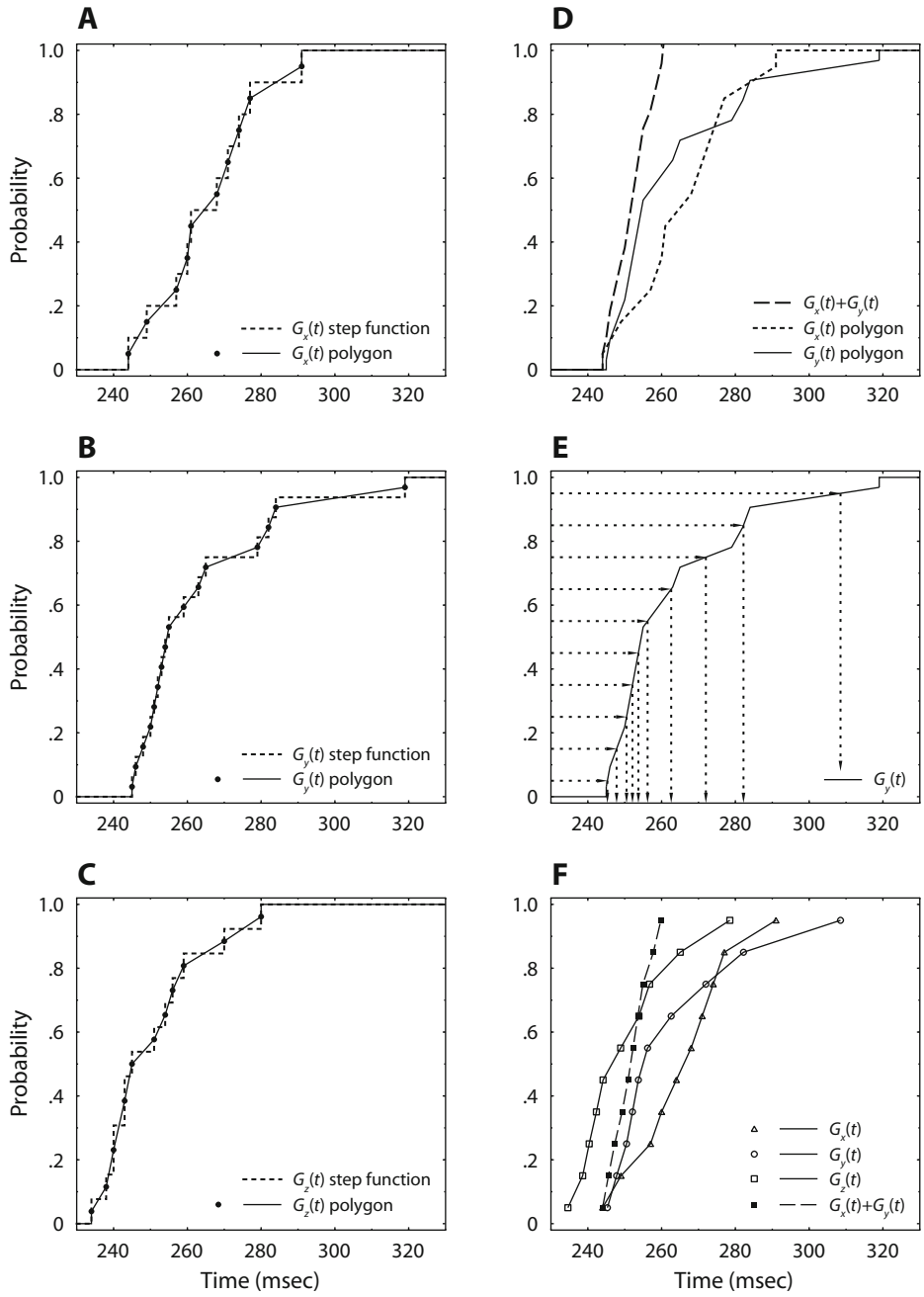

Figure 1. Illustration of computations for the data in Table 1. Panels A, B, and C show the estimates of G_x, G_y, and G_z, respectively. In these panels, the dashed line depicts the step function estimate and the solid line shows the corresponding cumulative frequency polygon. Panel D shows the function (dashed line) representing the race model bound, $B(t) = G_x(t) + G_y(t)$. Panel E shows how to obtain 10 estimated percentile points corresponding to the percentiles .05, .15, .25, . . . , .95 for the function G_y. As is illustrated in this panel, the point associated with any percentile can be determined by finding the t value associated with the desired percentile on the cumulative frequency polygon. Finally, panel F summarizes the analysis by displaying the 10 estimated percentile points for each of the four functions of interest: G_x, G_y, G_z, and $B = G_x + G_y$.

Step 4: Aggregation Across Participants

In this step, the summary measure for each percentile value of each function is calculated by simply finding the average of that percentile value across participants (cf. Vincent, 1912).[1] For example, the summary measure for $z_{.05}$ is the average of the $z_{.05}$ values computed for the individual participants, as described in Steps 1–3.

The main goal of the analysis is usually to see whether the race model is significantly violated. Within the literature examining the RSE (e.g., Bucur et al., 2005; Krumme-

nacher et al., 2001), one common method for addressing this question is to compute separate paired *t* tests at each of the percentiles under examination. For example, each participant supplies a pair of scores $z_{.05}$ and $b_{.05}$, and a *t* test is used to see whether there is a significant difference between the two types of scores. If the mean for z_p is significantly less than the mean for b_p, it is concluded that the race model inequality is significantly violated at the *p*th percentile. Typically, the race model is rejected as insufficient to account for the data if there is a significant violation at any percentile. Although this test imposes a certain inflation of the Type I error rate because of the computation of multiple *t* tests (i.e., across numerous percentiles), this inflation may be eliminated by adjusting the overall significance level with a Bonferroni-type correction.

Monte Carlo simulations have indicated that this correction is somewhat conservative, so it is unlikely that violations will be identified by chance when the race model is correct (Kiesel, Miller, & Ulrich, 2005).

DISCUSSION AND CONCLUSIONS

Although tests of the race model inequality have been carried out for more than 20 years, this article provides the first detailed description of an appropriate algorithm for conducting such tests. Such a description seems especially necessary at this point because some users might be tempted to use standard software packages that are now readily available (e.g., Excel), but that in fact provide built-in functions that would give incorrect results when used to compute these tests.

The present article provides an algorithm built on an explicit solution to the problem of estimating CDFs of RTs when ties are present. Ties will tend to occur most frequently when sample sizes are large, when RT variance is small, and when the resolution of the RT clock is poor. Even if ties are expected to be infrequent, however, it is important to have a method of dealing with them, because they could occur in any real data set.

It should be emphasized that the sample size limits the estimation of upper and lower percentile points. Specifically, there is no satisfactory method to estimate a percentile point less than $p = 1/(2n)$ or greater than $p = (2n - 1)/(2n)$; points in either of these ranges would require extrapolation outside the range of the observed data or strong assumptions about the underlying RT distribution. Indeed, in cases in which there are ties at the smallest or largest sample values, the range of estimable percentile points narrows even further. In practice, however, this limitation is not too serious, because computer simulations indicate that the race model inequality can usually be tested quite effectively even with as few as 10 RTs per condition (Kiesel et al., 2005). Although such a small sample size might be necessary in clinical or developmental studies or in studies with low-probability conditions, more observations would certainly increase the accuracy of the test. On the basis of the simulation studies conducted by Kiesel et al., we recommend employing at least 20 RTs per condition.

The MATLAB and Pascal implementations of Steps 1, 2, and 3 of the algorithm that are provided in the appendixes incorporate the most sophisticated version of the algorithm presented in this article, and we hope that these could be easily adapted to any specific software requirements of researchers working in this area. Step 4—computation of *t* tests—can then be carried out via any standard statistical package. In addition, we can also provide a self-contained Windows program called RMITest that computes all four steps; this program can be downloaded from the authors' Web sites (e.g., psy.otago.ac.nz/miller/index.html).

AUTHOR NOTE

This research was supported by grants from the Marsden Fund, administered by the Royal Society of New Zealand, and from the Department of Psychology at the University of Otago. We thank Andrea Kiesel, Christopher Hone, and James Townsend for constructive comments on an earlier version of the manuscript. Correspondence concerning this article may be addressed to R. Ulrich or H. Schröter, Abteilung für Allgemeine & Biologische Psychologie, Psychologisches Institut, Universität Tübingen, Friedrichstr. 21, 72072 Tübingen, Germany (e-mail: ulrich@uni-tuebingen.de or hannes.schroeter@uni-tuebingen.de), or to J. Miller, Department of Psychology, University of Otago, Dunedin, New Zealand (e-mail: miller@psy.otago.ac.nz).

REFERENCES

Bucur, B., Allen, P. A., Sanders, R. E., Ruthruff, E., & Murphy, M. D. (2005). Redundancy gain and coactivation in bimodal detection: Evidence for the preservation of coactive processing in older adults. *Journals of Gerontology*, **60B**, P279-P282.

Colonius, H. (1988). Modeling the redundant signals effect by specifying the hazard function. *Perception & Psychophysics*, **43**, 604-606.

Colonius, H. (1990). Possibly dependent probability summation of reaction time. *Journal of Mathematical Psychology*, **34**, 253-275.

Colonius, H., & Townsend, J. T. (1997). Activation-state representation of models for the redundant-signals-effect. In A. A. J. Marley (Ed.), *Choice, decision, and measurement: Essays in honor of R. Duncan Luce* (pp. 245-254). Mahwah, NJ: Erlbaum.

Diederich, A. (1995). Intersensory facilitation of reaction time: Evaluation of counter and diffusion coactivation models. *Journal of Mathematical Psychology*, **39**, 197-215.

Diederich, A., & Colonius, H. (1987). Intersensory facilitation in the motor component? A reaction time analysis. *Psychological Research*, **49**, 23-29.

Gilchrist, W. G. (2000). *Statistical modelling with quantile functions*. Boca Raton, FL: Chapman & Hall/CRC.

Giray, M., & Ulrich, R. (1993). Motor coactivation revealed by response force in divided and focused attention. *Journal of Experimental Psychology: Human Perception & Performance*, **19**, 1278-1291.

Hershenson, M. (1962). Reaction time as a measure of intersensory facilitation. *Journal of Experimental Psychology*, **63**, 289-293.

Jiang, Y., Rouder, J. N., & Speckman, P. L. (2004). A note on the sampling properties of the Vincentizing (quantile averaging) procedure. *Journal of Mathematical Psychology*, **48**, 186-195.

Kiesel, A., Miller, J., & Ulrich, R. (2005). *Systematic biases and Type I error accumulation in tests of the race models*. Manuscript submitted for publication.

Krummenacher, J., Müller, H. J., & Heller, D. (2001). Visual search for dimensionally redundant pop-out targets: Evidence for parallel-coactive processing of dimensions. *Perception & Psychophysics*, **63**, 901-917.

Maris, G., & Maris, E. (2003). Testing the race model inequality: A nonparametric approach. *Journal of Mathematical Psychology*, **47**, 507-514.

Miller, J. (1982). Divided attention: Evidence for coactivation with redundant signals. *Cognitive Psychology*, **14**, 247-279.

Miller, J. (1986). Timecourse of coactivation in bimodal divided attention. *Perception & Psychophysics*, **40**, 331-343.

Miller, J., & Ulrich, R. (2003). Simple reaction time and statistical facilitation: A parallel grains model. *Cognitive Psychology*, **46**, 101-151.

MORDKOFF, J. T., & YANTIS, S. (1991). An interactive race model of divided attention. *Journal of Experimental Psychology: Human Perception & Performance*, **17**, 520-538.

PLAT, F. M., PRAAMSTRA, P., & HORSTINK, M. W. I. M. (2000). Redundant-signals effects on reaction time, response force, and movement-related potentials in Parkinson's disease. *Experimental Brain Research*, **130**, 533-539.

RAAB, D. H. (1962). Statistical facilitation of simple reaction times. *Transactions of the New York Academy of Sciences*, **24**, 574-590.

RATCLIFF, R. (1979). Group reaction time distributions and an analysis of distribution statistics. *Psychological Bulletin*, **86**, 446-461.

SCHWARZ, W. (1989). A new model to explain the redundant-signals effect. *Perception & Psychophysics*, **46**, 498-500.

SCHWARZ, W. (1994). Diffusion, superposition, and the redundant-targets effect. *Journal of Mathematical Psychology*, **38**, 504-520.

TOWNSEND, J. T., & NOZAWA, G. (1995). Spatio-temporal properties of elementary perception: An investigation of parallel, serial, and coactive theories. *Journal of Mathematical Psychology*, **39**, 321-359.

TOWNSEND, J. T., & NOZAWA, G. (1997). Serial exhaustive models can violate the race model inequality: Implications for architecture and capacity. *Psychological Review*, **104**, 595-602.

TOWNSEND, J. T., & WENGER, M. J. (2004). A theory of interactive parallel processing: New capacity measures and predictions for a response time inequality series. *Psychological Review*, **111**, 1003-1035.

ULRICH, R., & GIRAY, M. (1986). Separate-activation models with variable base times: Testability and checking of cross-channel dependency. *Perception & Psychophysics*, **39**, 248-254.

VAN ZANDT, T. (2002). Analysis of response time distributions. In H. Pashler & J. Wixted (Eds.), *Stevens' Handbook of experimental psychology: Vol. 4. Methodology in experimental psychology* (3rd ed., pp. 461-516). New York: Wiley.

VINCENT, S. B. (1912). The function of the vibrissae in the behavior of the white rat. *Animal Behavior Monographs*, **1**(No. 5), 1-84.

NOTE

1. The term *Vincentizing* has been used in the RT literature to refer to two slightly different procedures for estimating quantiles of a group RT distribution (Jiang, Rouder, & Speckman, 2004). The full Vincentizing procedure, which we do not recommend and which is not implemented in the present algorithm, proceeds in two steps (Ratcliff, 1979; Van Zandt, 2002): First, the procedure uses an alternative, nonstandard approach for estimating quantiles, and these are sometimes called "Vincentiles" (Van Zandt, 2002). It has been shown, however, that Vincentiles do not correspond well to the percentiles of nonsymmetric distributions, and thus Vincentiles should not be used for estimating the CDF of an RT distribution (Van Zandt, 2002). In a second step, these Vincentiles are averaged across participants to compute a group quantile. Because Vincentiles are poor quantile estimators at the level of a single participant, the resulting group average must also be a poor estimator. The term *Vincentizing* is also sometimes used to refer to the simpler procedure of merely averaging standard quantile estimates across participants, without including the first step involving the poor quantile estimators (see, e.g., Jiang et al., 2004). In summary, the present algorithm employs "Vincentizing" in the latter sense, so its estimation of RT quantiles rests on solid statistical techniques (Gilchrist, 2000).

APPENDIX A
Cumulative Frequency Polygon for Data Sets With Ties

A data set with ties can be arranged as in Table A1. Assume that a data set has a total of n observations, with only $k < n$ distinct observations as a result of some replications of the same data values. We order these k distinct observations from the smallest value to the largest—that is, $x'_1 < x'_2 < \ldots < x'_k$. The column n_i indexes how often the observation x'_i occurs in the data set. For example, if x'_i is observed three times, then $n_i = 3$. The fourth column of the table contains the cumulative number of observations, s_i, which is the number of observations that are less than or equal to x'_i. With this notation in mind, it can be shown that

$$G_x(t) = \begin{cases} 0 & \text{if } t < x'_1, \\ \frac{1}{n} \cdot \left(s_{i-1} + \frac{n_i}{2} + \frac{n_i + n_{i+1}}{2} \times \frac{t - x'_i}{x'_{i+1} - x'_i} \right) & \text{if } x'_i \le t < x'_{i+1} \text{ and } i \ne k, \\ 1 & \text{if } t \ge x'_k, \end{cases} \tag{A1}$$

with $s_0 = 0$. Note that this expression is equivalent to Equation 2 if there are no ties in the data set.

Table A1
Data Set With Ties

i	x'_i	n_i	$s_i = \sum_{j=1}^{i} n_j$
1	x'_1	n_1	$s_1 = n_1$
2	x'_2	n_2	$s_2 = n_1 + n_2$
3	x'_3	n_3	$s_3 = n_1 + n_2 + n_3$
⋮	⋮	⋮	⋮
k	x'_k	n_k	$s_k = n_1 + \cdots + n_k = n$

APPENDIX B
MATLAB Code for Testing the Race Model Inequality

This appendix presents a MATLAB version of the algorithm described in the text, including handling of ties as described in Appendix A. This MATLAB program performs Steps 1, 2, and 3 of the algorithm, and it is written as a function that can be integrated into any larger program or simply called at the command window to analyze the data set of a single participant. The function requires as input (1) the vectors X, Y, and Z of RTs in conditions C_x, C_y, and C_z, respectively; (2) a vector, P, of percentile points at which to test the inequality; and (3) a boolean variable, Plot, indicating whether a plot should be produced (i.e., Plot = true) or not (Plot = false). It returns the desired percentile values Xp, Yp, and Zp for conditions C_x, C_y, and C_z, respectively, and the percentiles Bp for the bound B, as well as a plot of the key values if one is requested.

```
function [Xp Yp Zp Bp] = RaceModel(X,Y,Z,P,Plot);
   %X,Y,Z are arrays with RTs for conditions Cx, Cy, Cz, respectively.
   %P is an array which contains the probabilities for computing
   % percentiles.
   %If Plot==true, a plot of the result is generated.

               %%% Step 1: Determine Gx, Gy, and Gz %%%

%Check for ties
   [Ux Rx Cx]=ties(X);
   [Uy Ry Cy]=ties(Y);
   [Uz Rz Cz]=ties(Z);

%Get maximum t value
   tmax=ceil(max([X Y Z]));
   T=1:1:tmax;

%Get function values of G
   [Gx]=CDF(Ux,Rx,Cx,tmax);
   [Gy]=CDF(Uy,Ry,Cy,tmax);
   [Gz]=CDF(Uz,Rz,Cz,tmax);

                %%% Step 2: Compute B = Gx plus Gy %%%

for t=1:tmax;
   B(t)=Gx(t)+Gy(t);
end

%Check whether requested percentiles can be computed
OKx = check(Ux(1),P(1),Gx);
if OKx == false
   disp('Not enough X values to compute requested percentiles')
   Xp=NaN;Yp=NaN;Zp=NaN;Bp=NaN;
   return
end
OKy = check(Uy(1),P(1),Gy);
if OKy == false
   disp('Not enough Y values to compute requested percentiles')
   Xp=NaN;Yp=NaN;Zp=NaN;Bp=NaN;
   return
end
OKz = check(Uz(1),P(1),Gz);
if OKz == false
   disp('Not enough Z values to compute requested percentiles')
   Xp=NaN;Yp=NaN;Zp=NaN;Bp=NaN;
   return
end

                %%% Step 3: Determine percentiles %%%

[Xp] = GetPercentile(P,Gx,tmax);
[Yp] = GetPercentile(P,Gy,tmax);
[Zp] = GetPercentile(P,Gz,tmax);
[Bp] = GetPercentile(P,B,tmax);
```

APPENDIX B (Continued)

```
%Generate a plot if requested
if Plot == true
   plot(Xp,P,'o-',Yp,P,'o-',Zp,P,'o-',Bp,P,'o-')
   axis([min([Ux Uy Uz])-10 tmax+10 -0.03 1.03])
   grid on
   title('Test of the Race Model Inequality','FontSize',16)
   xlabel('Time t (ms)','FontSize',14)
   ylabel('Probability','FontSize',14)
   legend('G_x(t)','G_y(t)','G_z(t)','G_x(t)+G_y(t)',4)
end

% return to calling routine.

function OK=check(U1,P1,G)
   OK=true;
   for t=(U1-2):(U1+2);
      if (G(t)>P1) && (G(t-1)==0)
         OK=false;
         return
      end
   end

%END of check

function [Tp] = GetPercentile(P,G,tmax)
%Determine minimum of |G(Tp(i))-P(i)|
np=length(P);
for i=1:np;
cc=100;
  for t=1:tmax
   if abs(G(t)-P(i)) < cc
      c=t;
      cc=abs(G(t)-P(i));
   end
  end
  if P(i) > G(c)
     Tp(i)=c+(P(i)-G(c))/(G(c+1)-G(c));
   else
     Tp(i)=c+(P(i)-G(c))/(G(c)-G(c-1));
  end
end
% End of GetPercentile

function [U R C]=ties(W);
% Count number k of unique values
% and store these values in U.
   W=sort(W);
   n=length(W);
   k=1;
   U(1)=W(1);
   for i=2:n
    if W(i)~=W(i-1)
      k=k+1;
      U(k)=W(i);
    end
   end
% Determine number of replications R
   R=zeros(1,k);
   for i=1:k
      for j=1:n
        if U(i)==W(j)
          R(i)=R(i)+1;
        end
      end
```

APPENDIX B (Continued)

```
   end
%Determine the cumulative frequency
   C=zeros(1,k);
   C(1)=R(1);
   for i=2:k
      C(i)=C(i-1)+R(i);
   end
%END of Ties

function [G]=CDF(U,R,C,maximum);
G=zeros(1,maximum);
k=length(U); n=C(k);
for i=1:k;
   U(i)=round(U(i));
end
for t=1:U(1);
   G(t)=0;
end;
for t=U(1):U(2);
   G(t)=(R(1)/2+(R(1)+R(2))/2*(t-U(1))/(U(2)-U(1)))/n;
end;
for i=2:(k-1);
   for t=U(i):U(i+1);
      G(t)=(C(i-1)+R(i)/2+(R(i)+R(i+1))/2*(t-U(i))/(U(i+1)-U(i)))/n;
   end
end
for t=U(k):maximum;
   G(t)=1;
end;
% End of RaceModel
```

APPENDIX C
Pascal Code for Testing the Race Model Inequality

This appendix presents an object-oriented Pascal version of the algorithm described in the text, including handling of ties as described in Appendix A. The algorithm can be used to perform Steps 1–3 for each experimental participant, and the *t* tests of Step 4 can then be computed with any standard statistical package. As is illustrated in the main program at the end, the user needs to load observed RTs for each participant into the Raw arrays for the stimulus conditions C_x, C_y, and C_z and then call the Tabulation procedure for each condition. After that, the value of a desired percentile point for any distribution can be computed with the TofG function, and the value of the race model bound at any desired percentile point can be computed with the TofSumG function.

```
Program VCDFTest;

Const MaxArraySize = 2000;

Type
   RealArray = Array[0..MaxArraySize] of Real;
   IntArray = Array[0..MaxArraySize] of Integer;

   StimulusType = Object
      Raw, Unique, G : RealArray;
      N, S : IntArray;
      K, TotalN, CheckAt : Integer;
      Constructor Construct;
      Function GofT(t : Real) : Real;
      Function TofG(DesiredG : Real) : Real;
      Procedure Tabulations;
      End;

Var X, Y, Z : StimulusType;
```

APPENDIX C (Continued)

```
Procedure SortRealArray(Var RA : RealArray; TotalN : Integer);
Var I, J, D : Integer;
   Temp : Real;
Begin
D := 2;
While 2 * D < TotalN Do D := 2 * D;
D := D - 1;
Repeat
   For I :=1 to TotalN - D Do Begin
      J := I;
      While J >= 1 Do Begin
         If RA[J] > RA[J+D] Then Begin
            Temp := RA[J];
            RA[J] := RA[J+D];
            RA[J+D] := Temp;
            End
          Else J := 0;
         J := J - D;
         End;
      End;
    D := D div 2;
  Until D = 0;
End;

Constructor StimulusType.Construct;
Begin
End;

Procedure StimulusType.Tabulations;
Var I, J : Integer; CurrentRaw : Real;
Begin
SortRealArray(Raw,TotalN);
{ Next:
   o Determine the number of unique values, K.
   o Store these values in the Unique array.
   o Count the number of occurrences of each value in N. }
CurrentRaw := Raw[1] - 1;
K := 0;
For I := 1 to TotalN Do
If Raw[I] = CurrentRaw Then Inc(N[K]) Else Begin
   Inc(K);
   N[K] := 1;
   CurrentRaw := Raw[I];
   Unique[K] := CurrentRaw;
   End;
{ Next, compute the values of the sum array, S, from the values in the
  count array, N, for the K different distinct values: }
S[0] := 0;
For I := 1 to K Do S[I] := N[I] + S[I-1];
{ Next, make the G array: }
G[0] := 0;
CheckAt := 1;
I := 0;
Repeat
   Inc(I);
   G[I] := GofT(I);
Until (G[I] >= 1) or (I = MaxArraySize);
If G[I] < 1 Then Begin
   Writeln('Fatal Error: Halting because tabulation routine failed to
      reach');
   Writeln('cumulative probability of 1.0.');
   Halt(1);
   End;
```

APPENDIX C (Continued)

```
For J := I+1 to MaxArraySize Do G[J] := 1;
End;

Function StimulusType.GofT(t : Real) : Real;
{ Compute the estimated CDF value G for an RT equal to t. }
Begin
If t < Unique[1] Then GofT := 0 Else
If t > Unique[K] Then GofT := 1 Else Begin
   While Unique[CheckAt] > t Do Dec(CheckAt);
   While Unique[CheckAt+1] < t Do Inc(CheckAt);
   GofT := 1.0 / S[K] * (S[CheckAt-1] + N[CheckAt]/2.0 +
      (N[CheckAt]+N[CheckAt+1])/2.0*(t-Unique[CheckAt]) /
      (Unique[CheckAt+1]-Unique[CheckAt]) );
   End;
End;

Function StimulusType.TofG(DesiredG : Real) : Real;
{ Compute the t value yielding the estimated CDF value DesiredG. }
Var t : Integer;
Begin
t := 0;
Repeat
   Inc(t);
Until (G[t] >= DesiredG) or (t = MaxArraySize);
If ( (G[t]   < DesiredG) and (G[t-1] <= 0) ) or
   ( (G[t-1] < DesiredG) and (G[t]   >= 1) ) or (G[t] < DesiredG) Then
     Begin
   Writeln('Fatal error: Halting because TofG cannot find desired
     percentile ', DesiredG:5:3,'.');
   Halt(1);
   End;
TofG := t - 1 + (DesiredG - G[t-1]) / (G[t] - G[t-1]);
End;

Function TofSumG(DesiredG : Real; Var G1, G2 : RealArray) : Real;
{ Compute the t value yielding the estimated summed CDF value DesiredG. }
Var t : Integer; Sum : Real;
Begin
t := 0;
Repeat
   Inc(t);
   Sum := G1[t]+G2[t];
Until (Sum >= DesiredG) or (t = MaxArraySize);
If ( (Sum > DesiredG) and (G1[t-1]+G2[t-1] <= 0) ) or (Sum < DesiredG)
   Then Begin
   Writeln('Fatal error: Halting because TofSumG cannot find desired
      percentile ', DesiredG:5:3,'.');
   Halt(1);
   End;
TofSumG := t - 1 + (DesiredG - G1[t-1] - G2[t-1]) / (G1[t] + G2[t] -
  G1[t-1] - G2[t-1]);
End;

Procedure ReadArray(Var InFile : Text; Var RA : RealArray; Var TotalN :
  Integer);
Begin
TotalN := 0;
Repeat
   Inc(TotalN);
   Read(InFile,RA[TotalN]);
Until EOLN(InFile);
Readln(InFile);
End;
```

APPENDIX C (Continued)

```
{ The following is a simple main program demonstrating the
  use of the above routines to analyze one individual's data. }

Const NPctiles = 10;

Var InFile, OutFile : Text;
   I : Integer;
   DesiredG : Real;
Begin
X.Construct; Y.Construct; Z.Construct;

Assign(InFile,'VCDFTest.Dat');  Reset(InFile);
Assign(OutFile,'VCDFTest.Out'); Rewrite(OutFile);

With X Do Begin
   ReadArray(InFile,Raw,TotalN);
   Tabulations;
   End;
With Y Do Begin
   ReadArray(InFile,Raw,TotalN);
   Tabulations;
   End;
With Z Do Begin
   ReadArray(InFile,Raw,TotalN);
   Tabulations;
   End;

For I := 1 to NPctiles Do Begin
   DesiredG := (I - 0.5) / NPctiles;
   Writeln(OutFile,DesiredG:6:3,X.TofG(DesiredG):10:3,
   Y.TofG(DesiredG):10:3,
   Z.TofG(DesiredG):10:3,
   TofSumG(DesiredG,X.G,Y.G):10:3);
   End;

Close(InFile);
Close(OutFile);
End.
```

(Manuscript received September 30, 2005;
revision accepted for publication February 10, 2006.)