Reaction Time Analysis for Interval Timing Research

Patrick Simen

1 Introduction: A Space of Timing Models

Reaction times are a rich form of data that has been widely used to understand how humans and other animals make simple, two-alternative perceptual discriminations, and how they time intervals. However, statistical techniques for analyzing reaction time data have been developed more extensively in the case of non-temporal, perceptual decision research than in timing research. This chapter describes how to apply reaction time (RT) analysis techniques from non-temporal decision research to the temporal domain, and it describes Matlab code that can be used to implement these analyses (see book's GitHub repository).

In order to motivate the application of these techniques for interval timing research, it helps first to consider a simple class of computational models of timing, the pacemaker-accumulator (PA) models (Creelman, 1962; Gibbon & Church, 1984; Killeen & Fetterman, 1988; Treisman, 1963). Simulating these models is interesting in its own right. For the purpose of this chapter, though, simulation is merely a tool that allows researchers to generate “fake” reaction time data that they can compare against their empirical data, test their intuitions, and debug their analysis code.

Importantly, starting with a simple, classic family of timing models also provides an example of how RT analysis can be used to select the “best” model within a class. Model selection is currently a topic of great interest in non-temporal decision research. Furthermore, starting with simulations may also help explain why the techniques used in non-temporal decision research have so far not reached the same level of use in timing research. That is, in many cases the sophistication of these techniques may seem to be wasted on the data in timing. Timing data, after all, typically has fewer degrees of freedom than decision data, as we will describe shortly. However, some examples from the literature will serve to illustrate the point that RT analysis can, contrary to this view, be extremely helpful in assessing models of timed behavior (e.g., Balci & Simen, 2014; Simen, Vlasov, & Papadakis, 2016).

1.1 *Fixed Clock-speed Pacemaker-accumulators*

We now consider two different types of timing model and ways to simulate them, thereby producing fake RT data. We can then apply the RT analysis methods described in the final section of this chapter to test whether these methods can properly determine which model best fits the fake data. This is an important step to undertake before applying the methods to real RT data from an experiment. If these methods cannot accurately determine which model generated the data even when we already know the answer, then they will not be useful in the real world. For this reason, model simulation should always be the first step in vetting a model-fitting procedure. We will restrict our attention in this chapter to the most basic and most classical type of timing model: the PA.

PAs are just about the simplest possible model of timing. They operate like a stopwatch, which counts up regular or irregular clock ticks until hitting some threshold either for making a decision (such as, whether the current interval is longer or shorter than another interval in memory) or for producing a response (such as pressing a lever to earn a reinforcement). We will refer to these two possibilities as decision and production tasks respectively. There are many possible variations on the basic PA idea, however, and different variations make different predictions, especially about production times. RT analysis techniques will therefore be useful in teasing these models apart.

Creelman (1962) developed an early PA model in which a source of clock pulses emits pulses randomly, at a fixed rate. The time between pulses was exponentially distributed, making the process of counting them up into a Poisson counting model. Since this basic process is central to several of the models for which we provide Matlab code, we begin with it. *Poisson_countermodel.m* contains the code in its entirety (see book's GitHub repository). It generates exponentially distributed random inter-pulse intervals, then adds them up, and checks when their sum has exceeded a threshold.

An important variation on this model allows for changes in pacemaker rate across trials, as well as variability in the pulse count threshold across trials. Gibbon and Church (1990) considered how these variations could account for a problem with Creelman's model, which is its conflict with a widely observed phenomenon in timing known as 'scalar invariance'. This benchmark phenomenon in interval timing research is one in which the standard deviation of remembered interval durations appears to grow linearly with the timed duration. This pattern yields a constant ratio of the standard deviation divided by the mean – a ratio known as the coefficient of variation (CV). Indeed, in production tasks, the production time distributions for different durations frequently superimpose when the data from different duration-conditions are divided by

their means. Creelman's model, in contrast, predicts decreasing CVs, and, thus, increased relative precision, as durations increase. Gibbon (1992) referred to this pattern as 'Poisson timing' (though as we shall see below, a Poisson timer that works on different principles yields perfect scalar invariance).

Treisman (1963) developed an early variant of this PA model that did not make specific assumptions about the distribution of clock pulses except for a key constraint on their statistics. The constraint is that the inter-pulse intervals are mostly relatively short in some trials, compared to the global average over trials, and mostly relatively long in others. That is, the clock speed varies from trial to trial (but always around a fixed average). Another way to say it is that, across trials, the i th inter-pulse interval following the clock-start is correlated with the j th inter-pulse interval in the same trial, for all intervals i and j occurring prior to the end of the timed duration. Simulations of this model demonstrate that it takes only a remarkably small amount of such correlation (i.e., variation in the pacemaker rate) across trials to recover the pattern of constant CVs. *Treisman63.m* contains code (see book's GitHub repository) that allows any desired distribution of inter-pulse intervals to be simulated while still observing the constraints. Part of what the code produces is a correlation matrix for selecting random inter-pulse intervals with the specified level of correlations. As users can see for themselves, correlations can be reduced nearly to 0 in a Creelman-style PA model but still obtain constant CVs.

1.2 *Variable Clock-speed Pacemaker-accumulators*

The models in the preceding section used a pacemaker that emits random pulses at a constant rate on average, and stores different pulse-count totals to encode different durations T . In contrast, the models in this section use a fixed pulse-count threshold (call it θ) and a variable pacemaker rate, $A = \theta/T$. When the pulses are purely excitatory, the model is equivalent to Killeen and Fetterman's (1988) Behavioral Theory of Timing (BeT) model (although the "pulses" in that case are interpreted as transitions between states of behavior). When we add negative pulses emitted at a rate proportional to the positive pulse emission, the resulting process closely approximates a process known as a diffusion process (Simen et al., 2013). Such processes are idealized, one-dimensional Brownian motion systems, in which a particle (equivalent to a pulse-count) drifts upward while being continuously perturbed by noise at every moment. This kind of perturbation (in three dimensions) accounts for the diffusion of particles in a liquid or gas over time. One-dimensional drift-diffusion models (DDMs) are leading models of two-alternative perceptual decision making, where the "particle position" represents something like the log odds ratio of one hypothesis vs. the alternative. In either case, BeT or the

time-adaptive, opponent Poisson DDM (TopDDM) of Simen et al. (2011, 2013), scalar invariance falls out of the models automatically. The code in *opponent_Poisson_Appendix3.m* explicitly compares the simulation of actual pulse-counting to a diffusion approximation of pulse-counting (specifically, the Euler-Maruyama method, in which spike counting *per se* does not occur). The primary computational difference between these approaches is that in the former, a sequence of pulse-times is generated and then summed; in the latter, properly scaled deterministic and random noise increments are added to a running sum at every one of a sequence of time steps. The primary theoretical difference is that the pulses are finite in number and spaced out at separate points in time, in the former, while they are idealized as occurring infinitely often at every moment in the latter. Since the latter theoretical assumption of the diffusion model is not likely to be true in reality, it is worth noting that this assumption allows some very simple approximate mathematical expressions to be used to fit RT distributions.

2 Reaction Time Analysis Methods

Built-in Matlab functions that are particularly useful for reaction time analysis include sorting functions (`sort`), plotting functions (`plot`, `hist`, `ksdensity`), and many of the features of the Matlab Statistics Toolbox (this toolbox is an add-on product that comes with its own licensing fee). These functions are incorporated into several of the functions accompanying this chapter.

Growing numbers of researchers these days are also opting for alternatives to Matlab, such as the free, open-source, Python programming and scripting language, or the statistical programming language, R, in addition to open-source versions of Matlab such as Octave. Python, for example, operates in many ways like Matlab, particularly when the NumPy and SciPy numerical and scientific computing packages and the Matplotlib graphics packages are imported (see, e.g., Anaconda, for a complete development environment for scientific computing in Python). Here, however, we restrict our attention to programs in Matlab. The primary virtue of Matlab, in the author's opinion, is that, although it must be purchased and renewed yearly for updates, the software is stable and platform-independent across Mac, Windows, and Linux computing platforms; documentation for each function is generally trustworthy and well organized; and there are fewer of the problems that seem to accompany open-source software (e.g., the confusion surrounding multiple, slightly different parallel versions – e.g., Python 2 vs. Python 3 – and updates that frequently break functioning software until workarounds are developed). Whether those features are worth the cost is a matter of personal opinion.

2.1 *Using Moments to Characterize the Data and Evaluate Conformance to the Scalar Invariance Pattern*

The shape of a probability distribution can be characterized by its moments: that is, by the mean, variance, skewness, kurtosis, etc. Mathematically, the n th central moment is defined as the expected value of the n th power of the data minus the mean (when such expectation exists). In the case of skewness and kurtosis, the third and fourth central moments are normalized through division by the standard deviation. In Matlab, these moments can be computed with a single function call. For this, we will assume that the reaction time values are contained in a vector having n elements in the variable `rt`. With such a variable, we can compute:

$$\text{mean}(\text{rt}), \text{var}(\text{rt}), \text{std}(\text{rt}), \text{skewness}(\text{rt}), \text{kurtosis}(\text{rt})$$

Documentation for any of these can be easily obtained at the Matlab command prompt by typing, e.g.:

```
doc mean
```

In the literature on timing, the most famous phenomenon that can be observed by measuring RTs is the constancy of relative temporal precision (Gibbon, 1977), that is, the constant ratio between the first two central moments of production time distributions. In studies that require animals to learn to press a lever after a delay of T seconds from a stimulus to obtain a reward, production times are found to be typically accurate: the average of the production time distribution is close to T . The precision of the production time across trials, however, decreases as T increases. Such a relationship can be captured by computing the CV, which is the standard deviation (square root of the variance) divided by the mean of the response times contained in the vector variable `rt`. In Matlab, the CV can be obtained with:

$$\text{CV} = \text{std}(\text{rt}) / \text{mean}(\text{rt})$$

CVs are often found to be roughly constant as a function of T across groups of participants. Violations of CV constancy are often observed if multiple durations T are mixed in a single experiment, with both human and non-human participants (Bizo, Chu, Sanabria, & Killeen, 2006).

2.2 *Skewness and Model Selection*

Skewness, the third standardized moment, captures how symmetric a distribution is. Gaussian distributions (a shape that is often used to fit production

time data) have zero skewness. Positive skewness, in which the mean is greater than the mode, and the tail on the right is heavier than on the left, is widely observed in two-alternative perceptual decision RTs, but is less often observed in timing tasks, which typically yield skewness levels closer to 0. Nevertheless, skewness may provide a fairly robust test of the predictions of a timing model, because models that predict a Gaussian distribution of RTs can be tested by verifying whether skewness equals 0. Simple Poisson counter models, such as BeT, predict skewness equal to twice the CV. Diffusion models such as Top-DDM predict skewness equal to three times the CV (see Simen et al., 2011, 2013, for discussion of proofs; these properties can be verified by running the model simulation code provided in the GitHub).

It should be noted that when the CV is small, skewness is, therefore, also expected to be small. Thus, if conditions can be created that generate large CVs, such as by driving attentional resources away from the primary task, then skewness should be expected to increase for models such as BeT and DDM, but not for models that predict a normal distribution of RTs, such as the information processing implementation (Gibbon, Church, & Meck, 1984) of scalar expectancy theory (SET; Gibbon 1977).

In summary, skewness is worth computing as it helps in model selection. Matlab has a built-in skewness function that can be used in this way. The Matlab Statistics Toolbox also includes helpful functions for assessing normality or deviations from normality in a set of RT data in variable `rt` (e.g., `normplot(rt)`).

2.3 *Quantiles and Timescale Invariance*

In Section 2.1, we have discussed how the CV tends to remain the same for different base intervals. Another famous feature of timing data that implies constant CVs, but is a stronger form of invariance, is called scalar invariance or timescale invariance. This is a phenomenon in which the entire distribution of RTs can be rescaled, through division by the mean, so that the shape of the RT distributions with different means superimpose on each other perfectly after rescaling.

A good test of this form of invariance is to examine a feature of the RT data that is widely used in 2AFC research. To obtain a compact empirical description of the distribution, the RTs are divided into quantiles. For example, after the RTs are sorted from fastest to slowest, it is possible to calculate the 10th, 30th, 50th, 70th, and 90th percentile by taking the 10% fastest, 30% fastest, 50% fastest, 70% fastest, and 90% fastest RT, respectively. If there is scalar invariance, the plot of the quantiles from one duration condition should line up with the corresponding quantiles from another condition.

The following Matlab commands will compute the quantiles by sorting the reaction time data in the vector variable `rt` :

```
rt_sorted=sort(rt);
```

determine how many trials there are

```
lrt=length(rt);
```

and compute the quantiles of reaction time data stored in the Matlab variable `rt`:

```
quantiles1=rt_sorted(floor(lrt*[0.10.30.50.70.9]));
```

Supposing that there are quantiles from a different data set stored in the variable `quantiles2`, the two RT distributions can be examined for time scale invariance by plotting them against each other:

```
plot(quantiles1,quantiles2,'o');
```

These plot points should form nearly a straight line, with slope equal to the ratio of the average of the timed durations in the two different conditions. Additional analyses based on linear regression can be computed and the best fitting line can be superimposed on the data (a quick but incomplete way of doing this is to go to the Tools menu of the figure window in Matlab, select the Basic Fitting menu option, and then select “linear” in the popup dialog box that appears).

2.4 Maximum Likelihood Fitting of Distributions to RT Data

The most sophisticated method for testing model predictions is by fitting model parameters to empirical RT distributions. In two-alternative perceptual decision research, this approach is powerful and widely used. Despite having a similar goal, different fitting methods greatly vary in terms of their computational speed, their robustness (resilience from data not actually produced by the process under investigation, but instead delayed by distraction, for example), and the amount of data they require for fitting accurately. For this reason, a large amount of work is devoted to comparing the fitting methods and trying to find new ones.

Maximum likelihood fitting is a standard technique for fitting models to data in many areas. The *likelihood function* is the point-by-point product of a

candidate probability density function's values evaluated at each of the observed RTs. For example, a Gaussian density has two parameters, a mean, and a variance. To calculate the likelihood of the RT data for a given value of mean and variance, we multiply the probability density value at each RT point to get the overall likelihood of the data, given the particular parameter values used. If a new set of values for the mean and variance parameters gives a higher likelihood, then we prefer that set of parameters to the previous set. Intuitively, those parameters that make the data appear most likely are themselves likely to be the parameters that generated the data in reality. Thus, an iterative procedure can be implemented, wherein a parameter value is chosen (or multiple values are chosen, for models with more than one parameter), the likelihood of the data is computed, and then a new parameter is chosen to see if it gives a higher likelihood. This could be done purely randomly, but search procedures in Matlab's Statistics Toolbox, and also in its Optimization Toolbox, can be used to do much more intelligent searching over parameter space. The code for implementing such a search is shown in *fit_models.m* (see book's GitHub repository). It is relatively concise, because the *fitdist* function hides a large degree of complexity.

The maximum likelihood method is particularly useful in timing research, because PA models produce RT distributions with simple, closed-form expressions, such as the normal, gamma, or inverse Gaussian distributions. The most common distributions are built in to Matlab's Statistics Toolbox, allowing samples to be drawn easily. Moreover, because of the closed-form expression, the fit to the data can be done by simply modifying the parameters of the distribution and finding the best fit rather than by sampling the data and simulating the model output as in the case of models with no closed-form expression. Instead, a computation-intensive process is required to evaluate the likelihood function for the data of models without a known, closed-form RT distribution, so that maximum likelihood methods are frequently outperformed by methods such as the chi-square method (Ratcliff & Tuerlinckx, 2002). Furthermore, new methods such as hierarchical Bayesian methods (e.g., Wiecki, Sofer, & Frank, 2013) are becoming increasingly interesting to researchers, given their ability to fit small amounts of data and to infer population-level parameters efficiently. For example, they can infer a set of parameter values that represents patients with attention deficit hyperactivity disorder versus a set of values that represents neurotypical control participants. We do not go into those methods here, since they are at the forefront of development in the two-alternative decision domain, and have not yet (to my knowledge) been widely used in fitting timing data.

2.5 *Model Complexity*

Maximum likelihood fitting offers one other useful property, which is that likelihood methods can easily be adapted so as to penalize for model complexity. Occam's Razor is the principle that the simplest explanation of real data should be preferred, all else being equal. When one statistical model has more parameters than another, it has more flexibility to fit a wider range of data patterns. In the extreme, a model that has as many parameters as data points is likely to fit the data perfectly. However, such a model is not at all likely to generalize well to data that has not been fit. Such models are said to overfit observed data, at the risk of failing to fit unobserved data. To combat this risk, tests such as the Akaike Information Criterion (Akaike, 1974), and the Bayesian Information Criterion (Schwarz, 1978), can be used to rule out overly complex models. Both of these methods simply add a penalty to the logarithm of the computed likelihood function, with the amount of the penalty depending on the number of parameters. Thus, a model A that fits the data less well, but with fewer parameters than model B, may in the end have a higher likelihood score. In such a case, we select model A over model B.

If the log likelihood of the data is computed by the fitting methods described in the previous section and stored in variable ll , it can be adjusted for parameter penalties as follows, where the number of parameters in the model is k (for example, Creelman's model has a pulse rate parameter and a pulse-count threshold parameter, so $k = 2$):

$$AIC = 2 * k - 2 * ll;$$

Given that the log likelihood is subtracted from the parameter penalty in the standard formulation of the AIC, the goal is to select the model with the minimum AIC score.

The Bayesian Information Criterion is also used for model selection and tends to implement a stronger penalty for parameters:

$$BIC = k * \log(\text{length}(rt)) - 2 * ll;$$

2.6 *Outlier Treatment*

One factor that bedevils RT research is the presence of contaminated data. If a participant does not pay attention during a trial of an experiment, they may issue a response that is far later, or far earlier, than would normally occur. There is a host of different approaches to removing outliers from data, though none can be assured of doing it correctly (see, e.g., Ratcliff, 1993). After all, a very

long RT may just be ... a very long RT. Still, the presence of just one unusually long RT that results from inattention can throw off maximum likelihood fitting methods, because the likelihood of such a data point is so low according to the true parameter values.

Here is a simple technique for eliminating outliers that is by no means guaranteed to work perfectly, but in any case can be adapted by users to be more or less conservative as they see fit. It is included here primarily to emphasize an incredibly useful technique for indexing RT data that is outside an outlier cutoff range.

We can eliminate unusually long RTs by computing the standard deviation of the data, and then keeping only those data that are smaller than some number of standard deviations above the mean, for example 3:

$$rt = rt(rt \leq \text{mean}(rt) + 3 * \text{std}(rt));$$

The syntax inside the outer parentheses creates a vector of logical 1s and 0s. Only those elements of the `rt` array that have a logical 1 in the corresponding array created by the `<=` operation will be assigned to `rt`. The result is that any RT greater than 3 standard deviations above the mean will be deleted from the array `rt`.

3 Conclusion

The Matlab code that accompanies this chapter (see book's GitHub repository) is intended to help researchers who are new to RT analysis to begin analyzing their data with the aim of model selection. Arguably the best thing about Matlab is its extensive, easily searchable Help documentation (for Matlab version 2014a or later, type `doc` at the Matlab command prompt to bring up the documentation viewer). This author learned Matlab simply by progressing through the Matlab documentation and trying out the examples, which are provided in nearly every help topic at the command prompt. The MathWorks website also includes tutorial videos and forums for getting help from other users. After consulting the Matlab tutorial documentation section (called *Getting Started With Matlab* in version 2014b), I recommend that new users try out some of the functions provided with this chapter, using the debugger in Matlab to step through lines of code one by one to see how variables in memory are changing as a script or function is executing, and to learn how particular Matlab built-in commands are used for RT analysis. Experienced users may wish to use this code as a stepping-off point for investigating methods more

widely used in the two-alternative decision making literature, some of which, such as hierarchical Bayesian model fitting, have yet to be used widely in timing research.

I have not addressed the very useful technique of fitting RTs in retrospective timing tasks, as was done for example in Balci and Simen (2014). In such tasks, estimates of time intervals are used as the inputs to a decision process. Balci and Simen (2014) applied this technique to data from a temporal bisection task, in which intervals are presented and the participant must categorize them as being either closer to a short reference interval, or closer to a long reference interval. Choice probabilities for long and short choices and corresponding RTs in this case are not necessarily directly related to the mechanism of time estimation, but the RT data here provide important information about the mechanism by which temporal discriminations are made after an interval is over. Because the literature on non-temporal, two-alternative decision making covers this type of RT analysis extensively, and because fitting two-choice RT data is more complicated than fitting “1-choice” production times, I do not address such methods here. However, they are really just extensions of the approaches described here. Ratcliff and Tuerlinckx (2002) offer a comprehensive discussion of how model fitting is done, as just one example, and a number of fitting algorithms and tutorials exist in the two-choice perceptual decision domain.

References

- Akaike, H. (1974). A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19, 716–723.
- Balci, F., & P. Simen (2014). Decision processes in temporal discrimination. *Acta Psychologica*, 149, 157–168.
- Bizo, L.A., J.Y.M. Chu, F. Sanabria, & P.R. Killeen (2006). The failure of Weber's law in time perception and production. *Behavioral Processes*, 71, 201–210.
- Creelman, C.D. (1962). Human discrimination of auditory duration. *The Journal of the Acoustical Society of America*, 34, 582–593.
- Gibbon, J. (1977). Scalar expectancy theory and Weber's law in animal timing. *Psychological Review*, 84, 279–325.
- Gibbon, J. (1992). Ubiquity of scalar timing with a Poisson clock. *Journal of Mathematical Psychology*, 35, 283–293.
- Gibbon, J., & R.M. Church (1984). Sources of variance in an information processing theory of timing. In Roitblat, H.L., T.G. Bever, & H.S. Terrace (Eds.), *Animal Cognition* (pp. 465–488). Erlbaum.

- Gibbon, J., & R.M. Church (1990). Representation of time. *Cognition*, 37, 23–54.
- Gibbon, J., R.M. Church, & W.H. Meck (1984) Scalar timing in memory. In Gibbon, J. and L.G. Allan (Eds.), *Annals of the New York Academy of Sciences: Timing and Time Perception*, Vol. 423 (pp. 52–77), New York Academy of Sciences.
- Killeen, P.R., & J.G. Fetterman (1988). A behavioral theory of timing. *Psychological Review*, 95(2), 274–295.
- Ratcliff, R. (1993). Methods for dealing with reaction time outliers. *Psychological Bulletin*, 114, 510–532.
- Ratcliff, R., & F. Tuerlinckx (2002). Estimating parameters of the diffusion model: Approaches to dealing with contaminant reaction times and parameter variability. *Psychonomic Bulletin and Review*, 9(3), 438–481.
- Schwarz, G. (1978). Estimating the dimension of a model. *The Annals of Statistics*, 6(2), 461–464.
- Simen, P., F. Balci, L. deSouza, J.D. Cohen, & P. Holmes (2011). A model of interval timing by neural integration. *Journal of Neuroscience*, 31, 9238–9253.
- Simen, P., F. Rivest, E.A. Ludvig, F. Balci, & P.R. Killeen (2013). Timescale invariance in the pacemaker-accumulator family of timing models. *Timing & Time Perception*, 1, 159–188.
- Treisman, M. (1963). Temporal discrimination and the indifference interval: Implications for a model of the ‘internal clock’. *Psychological Monographs*, 77, 1–31.
- Wiecki, T., I. Sofer, & M.J. Frank (2013). HDDM: Hierarchical Bayesian estimation of the drift-diffusion model in Python. *Frontiers in Neuroinformatics*, 7, 14.