Psychophysiology, 52 (2015), 169–181. Wiley Periodicals, Inc. Printed in the USA.

DOI: 10.1111/psyp.12320

Regression-based estimation of ERP waveforms: II. Nonlinear effects, overlap correction, and practical considerations

NATHANIEL J. SMITH[a] AND MARTA KUTAS[b]

[a]School of Informatics, University of Edinburgh, Edinburgh, Scotland
[b]Departments of Cognitive Science and Neurosciences, University of California, San Diego, San Diego, California, USA

Abstract

A companion paper introduced the rERP framework, which recasts traditional event-related potential (ERP) averaging as a special case of a more flexible regression-based approach to estimating ERP waveforms. Here, we build on this foundation, showing how rERP analysis can also handle the estimation of nonlinear effects (a generalization of both the well-known approach of dichotomizing continuous covariates, and also of the ERP image technique), and can disentangle overlapping ERPs to temporally adjacent stimuli. We then consider how the use of rERPs impacts on other parts of the EEG analysis pipeline, including baselining, filtering, significance testing, and artifact rejection, and provide practical recommendations. Free software implementing these techniques is available.

Descriptors: Other, Language/Speech, Normal volunteers, EEG/ERP

As detailed in a companion article, the rERP framework provides a flexible way to adapt well-known regression techniques to the problem of estimating event-related brain potential (ERP) waveforms. Here, we first examine how this framework can be used to estimate nonlinear relationships between covariates and scalp potentials, and find that it generalizes both the common approach of dichotomizing/binning continuous covariates, and the ERP-image technique (Jung et al., 2001; Lorig & Urbach, 1995). Next, we show how regression's ability to distinguish partially confounded effects can be exploited in order to separate overlapping ERPs time-locked to temporally adjacent events. Finally, we discuss the practical considerations of baselining, filtering, significance testing, and rejecting artifacts in the rERP context. A list of free software packages implementing these methods is available at http://vorpus.org/rERP.

To illustrate these techniques, we use EEG data recorded by Urbach and Kutas (2008) from seven undergraduate participants engaging in a speeded go/no-go task, focusing on 673 artifact-free "go" trials.[1] Our question is how the scalp potential at a midline parietal site differs between trials with different response times (RTs). A preliminary analysis using traditional techniques (Figure 1) reveals a complex pattern, in which the three faster bins all show a positive peak at around 350 ms, but with variable latencies and different drop-offs, and the slowest bin remains flat from 200 ms onwards. Even after 800 ms, the four waves all remain rather different. In the next two sections, we demonstrate different ways that rERP analysis can be used to elicit more insight from such messy-seeming data.

Discovering Nonlinear Relationships Via Splines

Slope rERPs, as introduced in Smith & Kutas, 2015, provide a natural mechanism for analyzing continuous covariates like RT, but make a very restrictive assumption: that the scalp potential will at each latency vary linearly with the covariate. This is unlikely to be strictly true in most cases, and often we will not know what relationship actually holds. Fortunately, regression provides a standard set of techniques for estimating nonlinear relationships from data.

These techniques are powerful and, when used appropriately, can do anything that a simple linear model can do and more. However, this power does come at a cost in conceptual complexity, and their use may therefore demand greater sophistication on the part of the analyst, her analysis software, and her audience. So, before describing their application to EEG analysis, we want to emphasize that they are likely to add more value in some situations than others.

This work was supported in part by NIH grant T32-DC000041 to the Center for Research in Language, NIH grant T32-MH20002 to the Institute for Neural Computation, NICHD grant HD22614 to MK, funding from the Army Research Laboratories Cognition & Neuroergonomics Collaborative Technology Alliance, and funding from the European Union's 7th Framework Programme under grant agreement No. 270273 (XPERIENCE). We thank R. H. Baayen, K. A. DeLong, S. Frank, D. Groppe, T. F. Jaeger, R. T. Knight, R. Levy, S. Makeig, S. Narayan, M. Ramscar, M. Steedman, T. Urbach, and two anonymous reviewers for comments and discussion.

Address correspondence to: Nathaniel Smith, University of Edinburgh, Informatics Forum 3.29, 10 Crichton Street, Edinburgh, EH8 9AB, Scotland, UK. E-mail: nathaniel.smith@ed.ac.uk

1. Thanks to David Groppe for suggesting this example and preprocessing the data.

PSYCHOPHYSIOLOGY

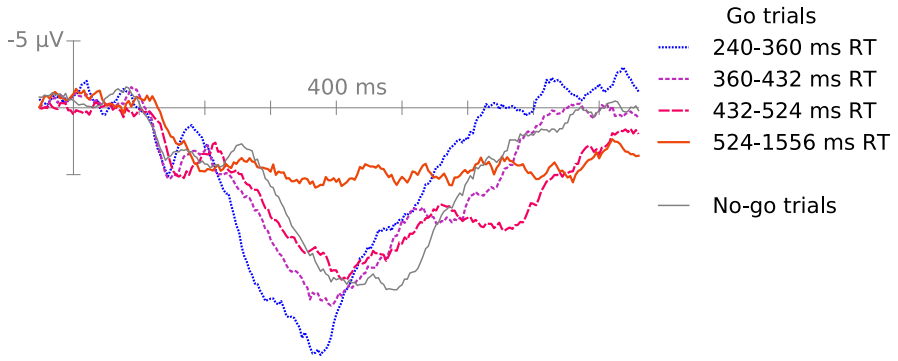

Figure 1. ERP waveforms from our example go/no-go response time task, with go trials broken down into four equal-sized bins by response time (RT), together with no-go trials for comparison.

Do You Need Nonlinear Regression?

The assumption of linearity is not so arbitrary as it may seem, and may in fact be a viable simplifying approximation in many cases.[2] Most relationships seen in the real world are smooth to a greater or lesser extent. (Smooth means that nearby points have similar values; the more a curve jumps or wiggles, the less smooth it is.) Locally, smooth functions can be reasonably approximated by a linear function. Furthermore, while U-shaped relationships do occur, they are the exception, rather than the rule. Most relationships are monotonic. If the factor we are studying has a monotonic effect on the ERP, and if all we care about is detecting whether or not an effect exists and determining its overall directionality, then a linear model may not be optimal, but it will get the job done.

Even if we are interested in the detailed shape of some relationship, we may not have enough data to estimate it. When we relax our linearity assumption, we still must make some assumptions about the range of possible curves. Otherwise, we cannot rule out pathological curves that, for instance, pass through every single data point. While there are a variety of techniques used to accomplish this, they generally take the form of somehow requiring our curves to be simpler and less wiggly. The fewer data points we have, the stronger these assumptions must be. When our data are sufficiently limited, we may find ourselves required to use constraints so strong that we are effectively doing linear regression after all. In such situations, again, we lose little by using conventional linear regression.

On the other hand, there are situations where it's particularly valuable to allow the regression model the flexibility to choose a nonlinear relationship to match our data. The most obvious is when discovering the shape of the relationship is itself a target of our analysis (e.g., Smith & Levy, 2013). A less obvious case occurs when our analysis includes control covariates—covariates whose effect we don't actually care about, but which are correlated with covariates that we do care about. For such covariates, we aren't particularly worried about giving them too much flexibility and allowing them to overfit the data—we don't plan to interpret them, and the worst that could happen is that they soak up more variance than they really deserve. This should be done in moderation, of course, but just as it's better to err on the side of including control covariates you don't need, it's better to have a model that is a little too flexible than a model that is a little too inflexible. Too inflexible a model may fail to fully account for the control factor's real effect, leaving some portion of it to be spuriously explained by the factors we do care about. Similar remarks apply to analyses of interactions between multiple covariates. In particular, a term representing the quadratic (nonlinear) effect of one covariate will be partially collinear with terms representing linear interactions between this covariate and other covariates that are correlated with it. If our model includes only the interaction terms but not the nonlinear terms, then we may get spurious results.

The third case where a nonlinear model is crucial is when we have reason to believe that our covariate may show a U-shaped effect. This is obvious, but in the rERP context, there is a less obvious consequence: if some covariate affects the latency of a component, then that covariate will effectively produce a U-shaped effect at some time points. An example occurs in Figure 1: at around 400–600 ms in the ERP, we see more positivity for medium-length RTs as compared to both short or long RTs. This is a U-shaped effect of RT on ERP amplitude, and so using a slope rERP to analyze the effect of RT here would be inappropriate.

There is a large literature on nonlinear regression; our goal is to show specifically how the basic idea of nonlinear regression (however implemented) applies to ERP analysis. Therefore, we give only a brief introduction here; for more information, see, for example, Wahba (1990) or Wood (2006).

The Spline Trick

In order to fit nonlinear models with linear least squares regression, we need a trick. That trick is to pick some particular set of basis functions, $f_1(x)$, … , $f_k(x)$, which are selected so that taking weighted sums of them, $\alpha_1 f_1(x) + \cdots + \alpha_k f_k(x)$, lets us construct a wide variety of nonlinear curves. Different sets of basis functions lead to different spaces of possible curves; the idea is that our regression fit will choose from these curves to pick the one that best matches the data. There are some good standard basis sets that are generally used. We must also decide how many basis functions

2. We already make this linearity assumption in other situations when analyzing ERP data. For instance, when choosing words to use in a language experiment, we often want to choose two sets of words that are matched on a large number of word properties that would otherwise become possible confounds, such as cloze probability or frequency. Ideally, we would choose pairs of words such that each pair was perfectly matched on all of these properties, but in practice we instead settle for ensuring that the average value of these properties is matched between the two groups. The reasoning is that if these properties are the same on average, then they cannot cause any differences in the ERPs between our two sets—which is only justified if we also assume that these properties have a linear effect on the ERP. If frequency has a nonlinear effect on the ERP, then the average ERP to a set containing one word with frequency 10 and one word with concreteness score 30 may not be the same as the average ERP to a set containing two words that both have frequency 20, even though the average frequency is the same in both cases.

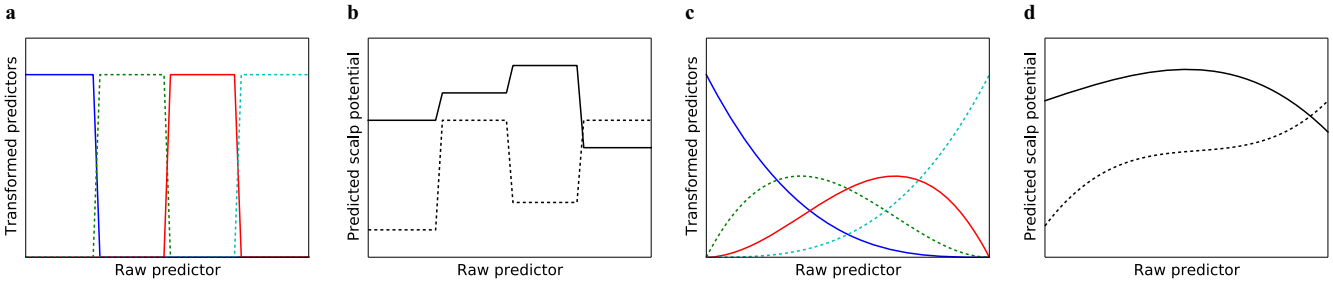

Figure 2. Two possible sets of basis functions for use in fitting nonlinear curves to data. a: A step-function (zero-degree B-spline) basis with four functions. By taking weighted sums of these functions, we can construct a variety of piecewise-constant curves. b: Two examples of such curves, as might result from fitting a regression model. c: A cubic (third-degree) B-spline basis with four functions. By taking weighted sums of these functions, we can construct a variety of smooth curves. d: Two examples of such curves, as might result from fitting a regression model.

we wish to use, since the more we use, the more flexible our space of possible curves—and the more data we will need to use to get sensible estimates. Two possible basis sets are shown in Figure 2, along with examples of the curves they can produce.

But having selected our basis functions, how do we convince our regression model to actually do the fitting? It's surprisingly simple: we define a set of k different predictors, one for each basis function:

$$\begin{aligned} x_{1i} &= f_1(\text{RT on trial } i) \\ x_{2i} &= f_2(\text{RT on trial } i) \\ &\vdots \\ x_{ki} &= f_k(\text{RT on trial } i) \end{aligned}$$

Now the least squares fitting will, as usual, find whatever set of weights $\beta_1, \ldots, \beta_k$ causes these transformed predictors to work together to best match our data. As far as the fitting process is concerned, these are β coefficients like any others. But these individual βs are generally not very interpretable on their own; instead, we interpret them as together selecting the single curve in our space that best fits the data: the curve $\beta_1 f_1(x) + \cdots + \beta_k f_k(x)$. We've fooled a linear technique into fitting a nonlinear curve.

What basis functions should we use? The simplest option is to divide up the possible values for our factor into several nonoverlapping ranges. For each range, we define one basis function as

$$f_j(x) = \begin{cases} 1, & \text{if } x \text{ falls into the } j\text{th range of values} \\ 0, & \text{otherwise} \end{cases}$$

(Figure 2a). By taking weighted sums of these functions, our regression model is able to construct any function that is piecewise-constant on these ranges (Figure 2b). Notice that when we then use these basis functions to make predictors x_{ji}, these predictors will be very similar to dummy- or treatment-coded categorical predictors; we can think of them as indicating categorically whether our factor of interest falls into a particular range on each trial. Using this set of basis functions is therefore equivalent to the common practice of splitting a continuous covariate into two or more bins (e.g., a median split) and then averaging across each bin, that is, dichotomizing, as we did in Figure 1.[3]

But while a step-function basis is simple, it produces rather odd-looking staircase functions, which seem unlikely to actually represent any real brain response. A more standard way of constructing a basis for this kind of regression would be to use higher-order spline functions. There are a number of different families of splines (which you choose usually matters little in practice), but cubic B-splines are one of the most popular (Figure 2c). Splines have two desirable features: first, like the step functions we saw earlier, each basis function covers a fairly short range of the data, with minimal overlap. (In fact, our step functions themselves are a special case of splines, but built out of zero-degree constant functions instead of the cubic functions that are more commonly used.) As a result, the behavior of our fitted curve near any particular value will primarily be determined by the data that we actually observed near that value, since only those data points are able to affect the relevant β coefficients. But, unlike the step functions, they produce more realistic—and thus, potentially more accurate—smooth curves (Figure 2d).[4] As with traditional dichotomization, we can vary the size of the bin covered by each basis function by adjusting parameters called *knots*; in the case of the step functions, the knots are simply the break points in our dichotomization.

rERP Spline Regression

Figure 3 shows a nonlinear rERP analysis of the go trials from our go/no-go data. With standard ERPs or slope rERPs, each covariate has a single β value associated with each latency, and we can plot these values as a waveform. Nonlinear rERPs are harder to visualize, because at each latency we are now estimating an arbitrary curve representing the nonlinear effect our covariate has on the scalp potential. Or, put another way, for each of the infinitely many possible values of our covariate, we have a potentially distinct ERP waveform. So, to visualize nonlinear rERPs, instead of plotting individual waveforms, we use false-color images. In these figures, latency relative to the time-locking event runs along the x axis, and response time varies along the y axis. Each horizontal slice of the figure represents the predicted ERP waveform for a particular RT (compare

3. And just as with treatment-coded categorical predictors, in practice we generally drop one spline basis function from our analysis in order to avoid creating perfect collinearity with the intercept term.

4. Another common practice is to use a polynomial basis, i.e., one in which $f_1(x)$ is a linear function, $f_2(x)$ is a quadratic function, etc. This approach does produce smooth curves, but it fails on our first criterion: aberrations in the data in one localized region might cause the entire fitted curve to change radically. This is a significant obstacle to interpreting and trusting the results of polynomial regression; splines are generally preferred.

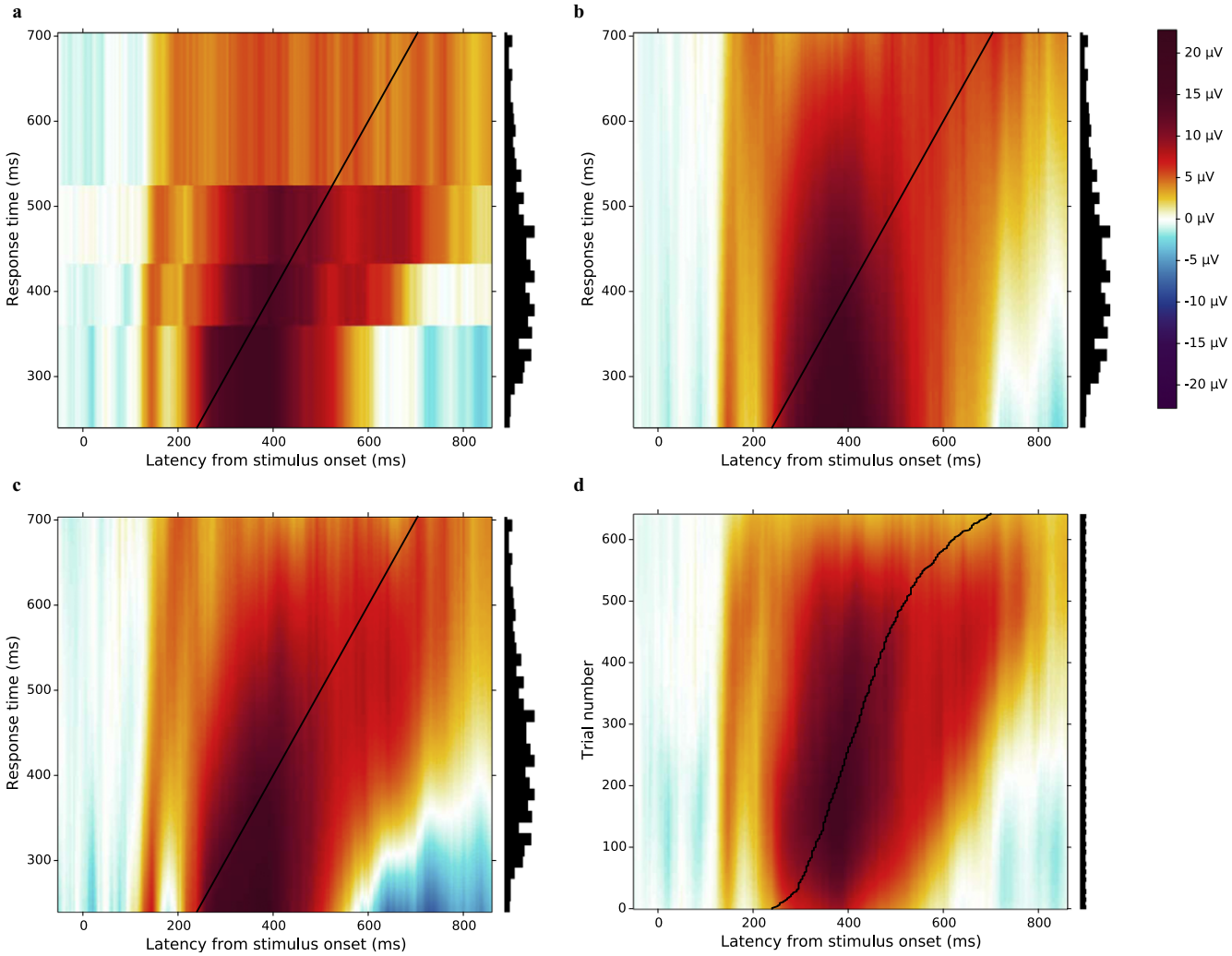

Figure 3. Four ways of analyzing the nonlinear effect of response time on scalp potential. In these plots, each horizontal strip represents the predicted ERP for a given RT, while each vertical strip shows how the predicted ERP at a particular latency varies with RT or trial number. Black lines indicate where the response (button press) occurs within each epoch, and a histogram on the right shows how many trials were observed with each RT. a: Regression using a step-function basis set containing four step functions. This is equivalent to the standard approach of splitting the data into four equal-sized bins and calculating the average over each; this figure is simply a different way of visualizing the go curves from Figure 1. b: Regression using a cubic B-spline basis set with four basis functions. c: Predictions from simple linear regression (slope rERPs). d: An ERP image of the same data for comparison (trials are sorted by RT, then vertically smoothed by a Gaussian-weighted moving average with $\sigma = 100$ trials). (a), (b), and (c) are rERP analyses; (a) and (d) are previously described methods from the literature.

Figure 3a to Figure 1; these two figures depict the same analysis in different ways). Each vertical slice of the figure represents the estimated nonlinear relationship between RT and scalp potential at one particular latency (comparable to Figure 2b,d).

Figure 3a,b demonstrate nonlinear rERPs using step-function and cubic B-spline bases, respectively. In these figures, there is a diagonal pattern visible in the 400–600 ms range, running roughly parallel to the black RT line. This diagonal pattern is what a latency effect looks like. But unsurprisingly, this is absent in the linear (slope) rERP predictions plotted in Figure 3c. As discussed above, latency effects create U-shaped effects at specific latencies, and because a slope rERP cannot capture U-shaped effects, the estimated slope coefficient during this period is essentially zero. This is another example of how slope rERPs are analogous to difference waves: if we had divided our RTs into just two bins and computed their difference, then the difference wave in this region would have been near zero.

On the right of each image, we plot a histogram showing how much data were available in different parts of the range of RT values. In general, the nonlinear curves will be more trustworthy in areas where more data are available, and we accordingly arrange our basis functions to allow more flexibility in these regions. This is equivalent to the standard practice of choosing dichotomization boundaries so as to produce equal-sized bins, rather than spacing boundaries equally in covariate space.

Finally, Figure 3d demonstrates for comparison the ERP image technique (Jung et al., 2001; Lorig & Urbach, 1995). An ERP image is produced by sorting trials according to some factor of interest (RT, in this case), and stacking the raw data from these trials vertically. Then, we use a moving average to smooth out the raw data measured at each latency on adjacent trials, and plot the results as a two-dimensional pseudocolor image. This smoothing serves exactly the same purpose as our use of a limited set of basis functions, in that it allows us to pool data from trials with similar

RTs, to reduce the effect of noise while still being able to see whatever nonlinearities are present. In fact, while we focus here on the basis function approach for its simplicity, the use of such nonparametric smoothers is a well-studied alternative technique for peforming nonlinear regression (Hastie & Tibshirani, 1990). Conceptually, the main difference here between the ERP image and the rERP estimates is the *y* axis. For the ERP image, this is the trial number; we can think of the ERP image as a nonlinear rERP that has been stretched and squished vertically in order to flatten out the latency distribution over the *y* axis. Thus, if we had seen exactly the same brain responses but a different distribution of RTs (perhaps in another condition), the ERP image might look very different. For the rERP estimate, the *y* axis is the RT itself; a different distribution of RTs would potentially make our estimates more or less accurate at different points on the *y* axis, but the image itself would remain comparable across conditions.[5]

The rERP approach is also more general than ERP images. Figures 3a–3c were all produced by the same software routine; all that had to be changed was a single line specifying the predictors to be used. We could just as easily have added additional controls, or simultaneously estimated the nonlinear effects of multiple partially confounded covariates like word expectancy and word frequency, neither of which are possible with traditional ERP images.

To keep this example manageable, we've used only a small number of basis functions, which is equivalent to assuming that the underlying relation between RT and scalp potential is relatively smooth. In general, though, this is a parameter of our analysis that we can vary, and it results in a trade-off. As we add more flexibility to our model, we find ourselves trying to extract more information from a fixed amount of data. At one extreme, one could enter a different dummy-coded predictor for every stimulus that appears in the experiment. If we had unlimited data, this might even be the best option; in some experiments, every stimulus is, in fact, different (Laszlo & Federmeier, 2011). But in practice, such an analysis will rarely be useful. Instead, we have to strike a balance: as we weaken our model's assumptions, we increase the chance that it can accurately represent reality—but these assumptions about the form of the relationship between stimulus properties and brain response are exactly what allows the model to compare data across multiple trials to distinguish signal from noise. So, we have to strike some trade-off that allows us to pool data across multiple trials without introducing unacceptable biases. One of the great advantages of regression models is that they let us move mindfully between these extremes, and in cases where we do have a large amount of data, they allow us to use relatively mild assumptions and let the data speak.

More sophisticated regression strategies can make this trade-off by automatically optimizing some measure of overfitting based on the data itself; see Wood (2006) for details. These techniques have previously been applied to electroencephalography (EEG) by Hendrix (2009), Hendrix, Bolger, and Baayen (2014), Kryuchkova, Tucker, Wurm, and Baayen (2012), Tremblay (2009), and Tremblay and Baayen (2010).

Nonlinear rERPs are potentially applicable to any continuous covariate: RT, word expectancy or frequency, visual eccentricity, tone loudness, etc. In Figure 3, though, the parallelism between the black RT line and the dark red diagonal of increased positivity suggest that what we have in this case may be two overlapping effects: one time-locked to the stimulus, and the other time-locked to the response (Jung et al., 2001). So this analysis suggests that an even better approach in this instance might be not to use a generic nonlinear rERP, but to model this overlap explicitly. In the next section, we demonstrate how this can be done.

5. If we really wanted to reproduce the traditional ERP image's stretching and squishing, we also have the option of computing a nonlinear rERP, but entering rank RT as our predictor instead of raw RT.

Overlap Correction

ERP activity often lasts for a second or more beyond the time-locking event. A lot can happen in a second. In a response time task, participants may respond; in a language comprehension study, several more words may be presented; in ordinary life, the average second contains a rich, multidimensional stream of sensory stimulation, multiple visual saccades, and other complex motor actions. If there are multiple events happening within a single second, and ERPs to these events last for a second or more, then these ERPs must be active simultaneously—they overlap.

There are often practical advantages to presenting stimuli with short interstimulus intervals (ISIs). Short ISIs allow investigations into interactions between the processing of successive stimuli (e.g., priming and habituation effects), improve performance on selective attention tasks, and make it possible to collect more data in a limited amount of time (Woldorff, 1993). But the traditional averaging technique provides no reliable way to isolate and reconstruct the ERPs attributable to temporally adjacent events. Particularly pernicious are situations in which adjacent stimuli have properties that are correlated with each other, creating a new kind of confounding problem. Previously, we've discussed situations where there is confounding between the different properties of a single event (e.g., word frequency is correlated with word expectancy), and thus effects that are caused by one may be incorrectly attributed to the other. But equally problematic is confounding between properties of adjacent events. For example, in naturalistic English text, the frequency of word n is correlated with that of word $n + 1$. If we're not careful, then in a sentence reading experiment we may conclude that we have found an ERP effect of word frequency at, say, 100 ms poststimulus onset, when in fact the component in question is driven by the frequency of the previous word at a later latency. Woldorff (1993) discusses analogous problems with confounding that arise in selective attention experiments, where, for example, in a design with "attend to visual flash" and "attend to auditory click" conditions, the estimated ERP for unattended auditory events will be contaminated by overlap with attended visual events, while the estimated ERP for attended auditory events will be contaminated by overlap with *un*attended visual events. An apparent effect of attention on auditory processing thus might actually arise from an effect of attention on visual processing, and vice versa.

The simplest method to reduce the effect of overlap is to randomly vary (jitter) ISIs, which "smears" the contributions of adjacent events to the ERP estimate. This provides only a partial solution at best. The best known technique for further correcting ERP estimates for overlap is Adjar (Woldorff, 1993), which is a relatively complex procedure requiring the manual computation of a number of successive approximations with uncertain accuracy. The technique proposed here is similar to Adjar in spirit, but lets the computer do the work of finding exact solutions automatically, and allows us to use results from the regression literature to guarantee theoretical rigor and characterize the conditions required to achieve accurate results. And, because it is part of the general rERP framework, the technique applies equally well to standard ERPs, slope rERPs, nonlinear rERPs, and combinations of these.

Techniques related to the one proposed here have previously been used for estimating hemodynamic response functions in

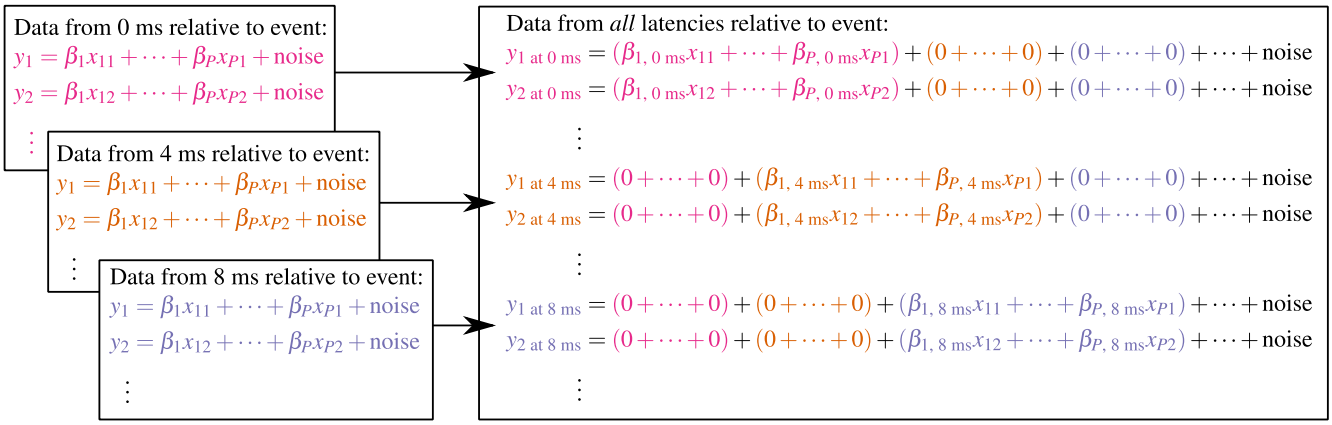

Figure 4. An illustration of the procedure for converting a set of separate regression models for different latencies into a single combined model for all latencies, as required for overlap correction. On the left, we see three of the models that previously we would have fit to different subsets of our data. (The particular latencies shown reflect a 250 Hz sampling rate.) On the right, we see how they are combined by, for each model, replacing each predictor by a new one that takes on the same values on the subset of data that its model was previously fit to, and zero elsewhere.

event-related fMRI (Hinrichs et al., 2000), and analyzing skin conductance responses (Bach & Friston, 2013).

Encoding Overlap in a Regression Equation

From by-epoch regression to continuous-time regression. Previously, we calculated rERPs by fitting a separate regression model for each latency, and then collected the resulting β coefficients from each model together to form individual waveforms. The first step to accomplishing overlap correction is to combine these many regression models into a single, giant model. To do this, we use the same "zero trick" that we used in the companion article (Smith & Kutas, 2015) to combine two intercept-only "classic" ERP models into a single model with dummy coding. There, this trick was used to replace the binning process, so that instead of dividing our events into separate groups before analysis, the division effectively happened as part of the regression fitting. Now we'll replace epoching in a similar manner, giving a regression model that is applied directly to continuous-time EEG.

If we have data for N trials, P predictors, and L distinct latencies, then before, we would fit L different regression models, where each regression model had entries of the form:

$$\begin{aligned} y_{\text{trial } 1,\text{latency } 0} &= \beta_1 x_{\text{predictor } 1 @ \text{trial } 1} + \cdots + \beta_P x_{\text{predictor } P @ \text{trial } 1} \\ &\vdots \\ y_{\text{trial } N,\text{latency } 0} &= \beta_1 x_{\text{predictor } 1 @ \text{trial } N} + \cdots + \beta_P x_{\text{predictor } P @ \text{trial } N} \end{aligned}$$

To coalesce these models into one, we have to replace each of our P predictors $x_{\text{predictor } p}$ by L new predictors $x_{\text{predictor } p,\ \text{latency } l}$, giving $P \times L$ predictors in total. The β value for $x_{\text{predictor } p}$ was in the model for latency l. Since we'll be fitting our new model on all of the original $y_{\text{trial } i,\ \text{latency } l}$ data points, without epoching, we have to define what values these predictors take for each of $N \times L$ data points, as follows:

$$x_{\text{predictor } p,\text{latency } l_1 @ \text{trial } i,\text{latency } l_2} = \begin{cases} x_{\text{predictor } p @ \text{trial } i}, & \text{if } l_1 = l_2 \\ 0, & \text{otherwise} \end{cases}$$

This looks complicated, but all it's doing is saying in regression language that, when we want to predict the scalp potential at 100 ms after some time-locking event, we should multiply the predictor values for the event we're looking at by the β values that are 100 ms into our rERP waveform, and ignore the other β values. The overall process is shown schematically in Figure 4. By itself, this transformation has no effect whatsoever: fitting this model once will give exactly the same results as fitting each of the L original models one at a time. The only differences are that we get our entire waveforms in one go, instead of having to assemble them one point at a time, and that our computer will have to work a bit harder—fitting these models takes a few seconds instead of tens of milliseconds.

One way to interpret our new, expanded predictors is as interactions between our original per-event predictors, and a second dummy-coded predictor that is 1 for data points measured at a certain latency from the time-locking event, and 0 otherwise:

$$x_{\text{predictor } p,\text{latency } l @ \text{data point } d} = x_{\text{predictor } p,\text{event } i} \times x_{\text{latency to event } i \text{ is } l @ \text{data point } d}$$

Like any dummy coding scheme, this approach allows our model to make independent estimates for each value of our predictor. When our predictor is latency, this means we are allowing the waveform to vary arbitrarily from one time point another, constrained only by the data. Now that our model is written in this form, though, we can consider using other codings instead. For example, we could impose a smoothness constraint on our rERP waveforms by encoding the latency predictor using a spline basis—and doing exactly this is one component of the generalized additive modelling approach used by Hendrix (2009), Hendrix et al. (2014), Kryuchkova et al. (2012), Tremblay (2009), and Tremblay and Baayen (2010). However, this is not as advantageous as it might at first appear. Encoding latency with a spline basis is essentially a way of applying noncausal smoothing to our ERP waveforms a priori before estimating them, which is not generally recommended (Luck, 2005). Like all smoothing, it will introduce distortions; but, unlike the use of post hoc low-pass filtering, which allows us to directly compare before and after waveforms, the consequences of spline-based smoothing may be hard to understand, especially when the degree of smoothing is selected by a method that is automatic, and thus opaque. And the usual justification for spline smoothing—that it allows us to pool data from similar measurements, amplifying the signal while reducing the noise—applies

only weakly here. In EEG recordings, the noise at adjacent time points may be just as similar as the signal is; to really increase our signal-to-noise ratio we have to pool data from multiple events that are more widely separated in time.[6] Therefore, for now we recommend sticking with the dummy-coded latency approach described above, which is the direct analogue to traditional ERP averaging in the continuous regression setting.

Overlap as confounding. Once we have combined our regression models into this expanded form, it's easy to see how to correct for overlap. Overlap occurs when a single EEG measurement (y value) is affected by multiple events—say, event 1 occurred 500 ms before this measurement, and event 2 occurred 100 ms before this measurement. If we assume that the ERPs to event 1 and event 2 sum, then this means that measurement y is

$$y = \text{ERP to event } 1_{500\,\text{ms}} + \text{ERP to event } 2_{100\,\text{ms}} + \text{noise}.$$

But the ERP to event 1 at 500 ms is simply the β values for 500 ms multiplied by the predictors for event 1, and similarly for the ERP to event 2 at 100 ms. To represent this overlap, in the combined model we allow the combined-model predictors for 500 ms to take on the appropriate values for event 1 at the same time as the combined-model predictors for 100 ms take on the appropriate values for event 2. More formally, we adjust our predictor definition like so:

$$x_{\text{predictor } p,\text{latency } l@\text{data point } d} = \begin{cases} x_{\text{predictor } p@\text{event } i}, & \text{if } d \text{ was measured } l \text{ ms after some event } i \\ 0, & \text{otherwise} \end{cases}$$

This transforms the problem of disentangling overlapping ERPs into the problem of disentangling partially confounded predictors in a regression model, and, as discussed in the companion article (Smith & Kutas, 2015), least squares fitting solves such problems automatically. In essence, the fitting process will consider all possible β waveforms, and pick the ones that best match the data after being time-locked to successive events and summed.

Under what circumstances should we expect this to work? We do not currently have a way to calculate precise variance inflation factor (VIF) values for continuous-time regression, because to be done properly this will have to take into account the temporal dependencies in the EEG background noise.[7] Nonetheless, the fundamental rule of partial collinearity still holds: the higher the correlations between our predictors, the more data are needed. In practice, there are two mechanisms that reduce collinearity between our expanded predictors: variation in ISI, and differences between stimuli. This is intuitive. If our experiment always uses an ISI of 500 ms exactly, and our stimuli are identical, then no method will ever be able to tell whether some component in the EEG occurs at 100-ms latency to one event versus 600-ms latency to another, because these latencies always co-occur. In the regression model, this means the predictors $x_{\text{predictor } p,\ 100\,\text{ms}}$ and $x_{\text{predictor } p,\ 600\,\text{ms}}$ will have identical values for every data point, creating perfect collinearity (aside from the first and last trials). But if we add some trials that use different ISIs, then on those trials there will be data points where one of these predictors is zero and the other is not, breaking the collinearity. The more variation there is in our ISI, the less correlated our predictors will be, and the better least squares will be at correcting for overlap.

But even if we have a fixed ISI, all is not lost. A fixed ISI means that the expanded predictors for the 100-ms and 600-ms latencies will always be zero or nonzero at the same times. But when they are nonzero, their values are determined by the corresponding events. If our predictor reflects word expectancy, for example, then the 100-ms predictor will reflect the expectancy for word n at the same time the 600-ms predictor reflects the expectancy for word $n - 1$, and these values will generally be different, again breaking the perfect collinearity. The extent to which this form of variability will help, of course, depends on the predictors in question. If the original predictors are entirely uncorrelated between nearby events, then our expanded predictors will be orthogonal; in this case, applying overlap correction is harmless but unnecessary. If the predictors are perfectly correlated, then jitter is our only hope. But if we have partial correlation between nearby events, then overlap correction can potentially be both effective and useful even in the absence of ISI variability. This may differ on a predictor-by-predictor basis within a single model; intercept terms, for example, never vary across stimuli, and thus require jitter to estimate. But as we saw before, collinearity problems affect only the collinear predictors,[8] and for some questions it may not matter how poorly estimated the intercept βs are so long as the other βs are estimated reliably.

Best of all, for overlap correction purposes, is to have as much variability as possible in both the ISI and the stimuli. Naturally, this must be balanced against other experimental considerations, and as long as there is enough variability somewhere, and we have enough data, then overlap correction can work. In any case, the least squares fitting process will automatically take full advantage of whatever variability is present, without any need for human intervention.

Simulation

At least, that's what the theory says. To verify this, we simulated three experiments involving high degrees of overlap. In the first, a continuous train of identical stimuli is presented, with each ISI randomly selected to fall between 200 ms and 400 ms in increments of 16.7 ms (the achievable ISIs for visual presentation using a monitor with a 60 Hz refresh rate). We time-locked a known ERP waveform (1,100 ms long) to each stimulus, and summed these time-shifted ERPs with phase-shuffled EEG background noise to produce simulated recordings. We then attempted to estimate the ERP from these simulated data, both with and without overlap correction. In this experiment, each data point has an average of 3.8 simultaneous ERPs overlapping with it.

The second simulated experiment is similar to the first, except that we used a fixed ISI of 300 ms, and now the stimuli themselves vary along an arbitrary continuous dimension (a stand-in for properties like visual contrast or expectancy), and this variation has a linear effect on the evoked potential:

6. Correlations between the EEG noise at nearby time points will also tend to thwart automatic smoothness selection methods (Wood, 2006), reducing their ability to prevent overfitting.

7. A useful target for further research would be to derive a VIF-like formula that will predict ahead of time how badly overlap impacts an arbitrary experimental design, as compared to a similar design without overlap, while taking into account the $1/f$ structure of the EEG background noise (Smith, 2011).

8. A demonstration on simulated data is given in the supporting information.

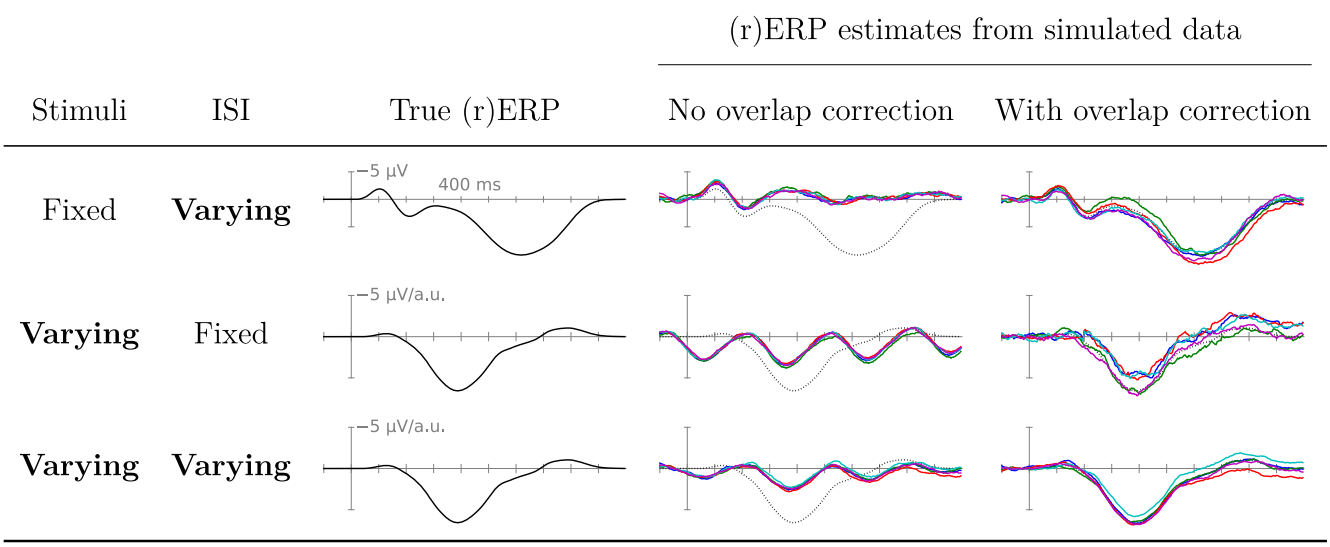

Figure 5. First row: An intercept ERP from a simulated experiment with ISIs jittered randomly between 200 and 400 ms. Second row: A slope rERP from a simulated experiment with fixed 300 ms ISI, and $r = 0.9$ trial-to-trial correlation in the simulated stimulus property (measured in arbitrary units, a.u.). Third row: Same as the second row, but with ISIs varying between 200 and 400 ms. Each plot shows estimates from five simulated runs of the experiment, along with the true (r)ERP for comparison (dotted line).

$$\text{Stimulus ERP}_n = \text{Intercept rERP} + \text{Stimulus property}_n \times \text{Slope rERP}$$

The simulated stimulus properties were selected so that on average across the experiment they are centered around zero, but on adjacent trials are highly correlated ($r = 0.9$); thus, the effect of event n is confounded with the effect of event $n - 1$. This is similar to what occurs when, for example, trying to estimate the effect of word frequency for word-by-word presentation of naturalistic text (though our simulation has much higher item-to-item correlation, making the overlap correction problem more challenging). As discussed above, with a fixed ISI it is impossible to reliably estimate the intercept rERP, and our estimates of this waveform were highly noisy (see online supporting information), but the slope rERP in the same model can be estimated reliably, even with this high degree of correlation and high degree of overlap.

The third simulated experiment is identical to the second, except that we replaced the fixed ISI with one that varied between 200–400 ms as in the first experiment.

The results, using 2,000 trials (~10 min) of simulated data, are shown in Figure 5. As expected, we find that if we omit overlap correction our (r)ERP estimates are entirely wrong. With overlap correction enabled, our method is able to recover the correct waveform even under these rather adverse conditions, so long as jitter or stimulus variation is present—and having both is best of all. (Of course, 2,000 trials is an unusually large number of trials to present in 10 min, but the high trial density is exactly what makes this a challenging case for overlap correction.) The supplementary information contains a more detailed set of simulations, showing the intercept rERPs for Experiments 2 and 3, as well as performance for different data set sizes.

Response-Time Effect Revisited

Our nonlinear analysis of the go trials from the go/no-go data above gave us reason to suspect that there might be two ERPs occurring simultaneously: one time-locked to the stimulus, and a second, overlapping ERP time-locked to the button press response. Our overlap correction technique allows us to estimate the true forms of these two ERPs separately, as shown in Figure 6. The two curves in Figure 6a are chosen by the fitting process so as to make their overlapped predictions (Figure 6b) match the actual trial-by-trial data as closely as possible in the least squares sense—compare Figure 6b to Figure 3a,b. As suspected, we find that the best explanation involves a large response-locked positivity, which begins somewhat before the button press is registered.

Note that in Figure 6a we use fairly long time windows, with all curves returning to zero before the edge of the analysis window. When not using overlap correction, it doesn't much matter how large an analysis window one uses; if it's too short, then it merely means that we can't see all of the curve. But when fitting the overlap correction model, all ERPs are assumed to be exactly zero everywhere outside of our analysis window. If this is not true, then any effects outside this window will not be subject to overlap control, and their overlap may contaminate the parts of the waveform that we do analyze. (This is another example of the principle that leaving predictors out of one's model may cause their effects to be spuriously attributed to whichever predictors remain.) So when using overlap correction to estimate waveforms, it's important to err on the side of using a long time window.

Validation

This approach to overlap correction treats the brain as a linear time-invariant (LTI) system; that is, it assumes that the brain response to seeing two items in quick succession is simply the sum of the brain response to seeing each of those items in isolation, with an appropriate temporal shift to account for time-locking. Theory and simulations confirm that, if the LTI assumption is true, then our method is capable of recovering the true underlying ERPs. But does this assumption accurately describe neural processing?

There are two ways we can respond to this concern. One is to perform various model-checking procedures, such as running the

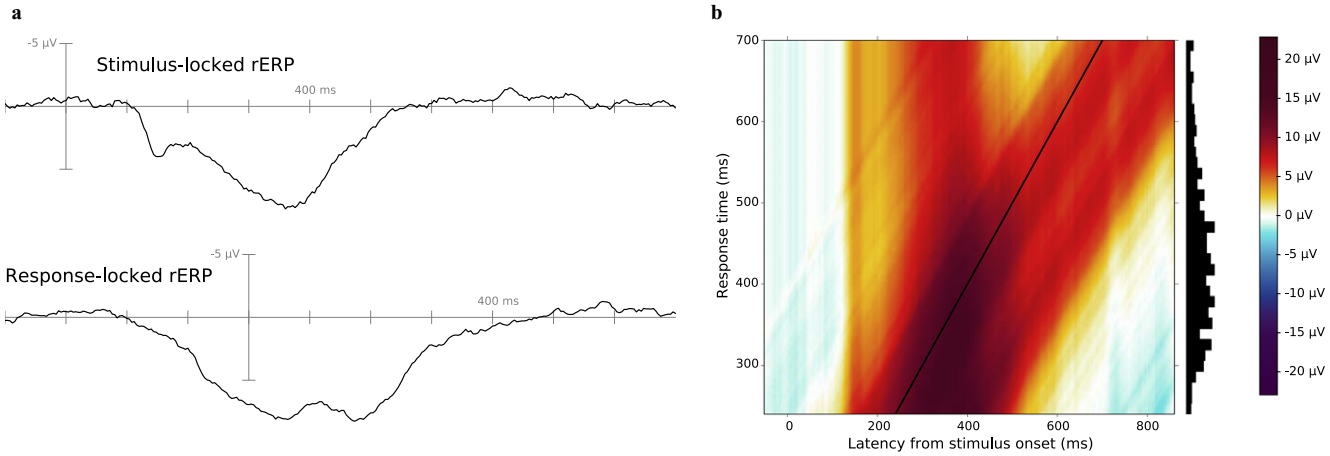

Figure 6. a: rERP estimates time-locked separately to stimulus presentation and button-press response, estimated with overlap correction to go trials from Urbach and Kutas (2008). b: The predictions made by this model. Compare to Figures 3a,b.

same experiment with both short and long ISIs, and checking whether the estimated rERPs are the same in the two cases, or simply checking whether overlap correction improves our model's fit. Burns, Bigdely-Shamlo, Smith, Kreutz-Delgado, and Makeig (2013) performed the latter experiment, analyzing EEG from a task where images were presented at high-speed (12 per second), and found that, when performing cross-validation, the overlap-corrected rERPs explained a higher proportion of the variance in the EEG signal than traditional ERPs from averaging, while using the same number of parameters. This doesn't tell us that the LTI assumption is correct, but it suggests that, for this task, using this form of overlap correction brings us closer to the truth than using no overlap correction at all. In addition, studies in fMRI suggest that the LTI assumption holds approximately for the hemodynamic response (Boynton, Engel, Glover, & Heeger, 1996; Cohen, 1997; Dale & Buckner, 1997), and to the extent that the hemodynamic response reflects the same generators as EEG, this suggests that it will be a useful approximation in this context as well.

In the long run, though, focusing on *whether* the brain violates the LTI assumption is probably not the most productive approach—it surely does. Instead, we should ask *how* the brain violates the LTI assumption. The overlap correction method proposed here should be viewed as a method for determining precisely which aspects of our data can be explained as arising from "mere overlap"—and thus which aspects of it cannot. If we find that overlap-corrected rERPs estimated from short-ISI and long-ISI conditions differ, then this would mean that we have successfully learned something new: that in our task ISI has effects that go above and beyond mere overlap. And the rERP approach is flexible enough to let us integrate covariates like ISI or the identity of the previous item (relevant for priming or habituation effects) directly into our analysis of the event-locked waveform, while still correcting for overlap with waveforms time-locked to other events.

In our go/no-go task, there must be something different about the neural processing between slow trials and fast trials that makes them slow or fast. So we expect there to be effects of RT that go beyond the timing of the response—but previously, it was difficult to look for these, since the massive positivity time-locked to the response overwhelms any other effects. Using our model of overlap correction, though, we can now subtract off the model predictions from the actual data (producing *residuals*), and then look to see if there are any patterns that remain unexplained. Figure 7a shows a nonlinear rERP analysis of the residuals from the overlap-corrected rERPs in Figure 6. Two things might strike us here.

First, all the amplitudes are much smaller than before; the overlap correction model has successfully accounted for most of the systematic variability in this data (or at least, that which can be captured by a nonlinear rERP). But there are some intriguing regularities in what's left: at around 200 ms, we see a roughly linear effect where slow trials are more positive than fast trials (red above blue); at around 300 ms, this switches to the opposite pattern (blue above red). A somewhat crude way to visualize this in more familiar terms is to subtract off only the response-locked rERP as estimated by our model, and then compute ERP estimates for dichotomized RT bins (see Figure 7b). This graph is directly comparable to Figure 1.

Figure 1 showed a RT-related latency shift in the positivity peaking near 350 ms, and sustained differences between our four RT bins extending to 800 ms and beyond. Figure 7b shows that both of these apparent effects can be accounted for by mere overlap with a response-locked positivity. But even after correcting for the response-locked ERP, we see residual amplitude differences between fast and slow trials in the 150–250 ms and 250–400 ms time windows, which call for further examination. In addition, we can now see that the no-go trials in fact deviate substantially from the go trials, especially in the 400–700 ms range; previously, they appeared similar, but this turns out to have been an artifact of the response-locked positivity in the go trials coincidentally matching a stimulus-locked positivity in the no-go trials.

But Figure 7b is (in statistical terms) a somewhat naughty approach, because the response-locked rERP in Figure 6a was calculated on the assumption that the stimulus-locked ERP did *not* vary with RT; if the stimulus-locked rERP had been allowed to vary, then the response-locked rERP might have come out differently. We're effectively assuming that all the unexplained variance in our original overlap-corrected model can be attributed to the stimulus-locked ERP, rather than the response. A more theoretically sound method of doing this analysis would be to allow the stimulus-locked rERP to vary in a nonlinear fashion (e.g., by using a step-function basis), and estimating it simultaneously with the

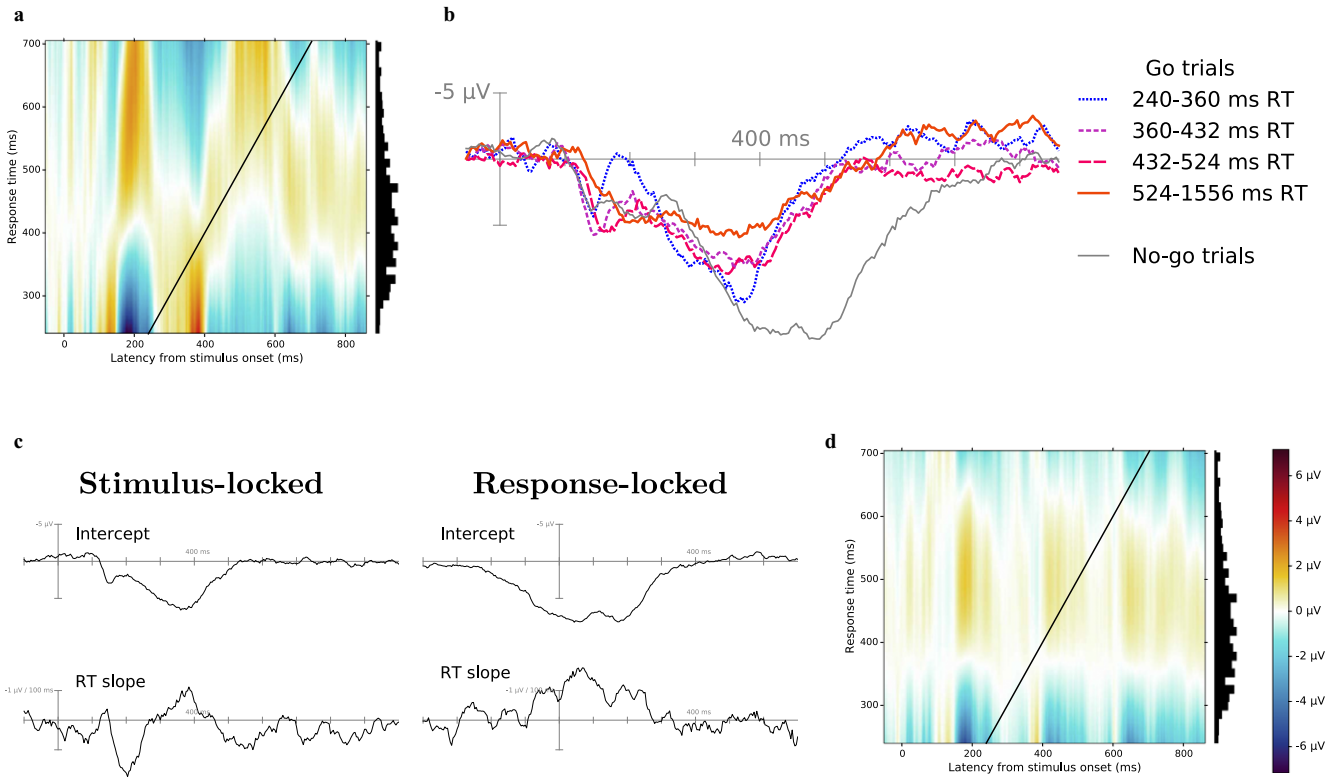

Figure 7. a: A nonlinear rERP (cubic spline with 4 *df*) analysis of the residual amplitudes left by subtracting the overlap model's predictions (Figure 6b) from the single-trial EEG data. b: ERPs formed by subtracting out Figure 6a's response-locked rERP (but not the stimulus-locked rERP) from the single-trial data, and then averaging across four RT bins. Compare to Figure 1. c: A more sophisticated overlap-corrected rERP model, in which both stimulus- and response-locked ERPs are assumed to vary linearly with RT. Because the RT predictor was centered, intercept rERPs here estimate ERP activity on trials with the mean RT (460 ms), while RT slope rERPs estimate how this changes as the RT varies above or below this. Tick marks on RT slope graphs indicate 1 μV change in scalp amplitude per 100 ms change in RT. d: Nonlinear rERP analysis of residual amplitudes from this more sophisticated model. Notice that images (a) and (d) share a different color bar from that used in previous figures.

response-locked rERP. This is perfectly doable with our method, but it turns out that, if we try on these data, the stimulus- and response-locked rERPs are too highly collinear to produce useful estimates. This is unsurprising, since, as Figure 3a shows, the stimulus-locked step-function basis is already able to capture a significant portion of the response-locked effect. One way to deal with this is to first fit one model—allowing it to capture all the variance that it can, resolving all ambiguities in its favor—and then examine what's left, as we did in Figure 7a (Kryuchkova et al., 2012; Laszlo & Federmeier, 2014). Another would be to gather more data. And a third is to fit a model that is flexible enough to capture the effects we care about, but not so flexible as to create massive collinearity problems. (When interpreting our results, however, we must keep in mind that no data analysis alone, no matter how clever, can rule out the interpretation where the entire pattern we observe is due to an extremely complicated nonlinear stimulus-locked ERP—this is the fundamental problem that the collinearity here is warning us about.)

Fortunately, as shown in Figure 2c, stimulus-locked linear (slope) rERPs are unable to capture the response-locked effect, which means that we can safely use them in our overlap analysis to analyze the residual effects of RT without creating too much collinearity. Further support for this strategy comes from our discussion of Figure 7a, in which the major unmodeled patterns appear to be roughly linear. Therefore, we fit a more sophisticated model, in which both the stimulus- and response-locked ERPs are allowed to vary linearly with RT (Figure 7c–d.[9]) Consistent with the cruder analysis in Figure 7b, we find that, in the stimulus-locked rERPs, slower responses are more negative than faster responses at around 200 ms, but from 300–400 ms the pattern reverses. Turning to the response-locked rERPs, we also now see that faster responses seem to be associated with an overall greater positivity than slower ones. In a real study, we would want to use statistical tests to confirm that these patterns were real, or we might continue by examining the residuals from this more sophisticated model for more unmodeled patterns (Figure 7d). Our goal here is not to provide a definitive analysis of these go/no-go data, but to illustrate how these models can be used and interpreted.

Practical Considerations

Having completed our discussion of the core rERP methods, we now turn to a number of practical issues that arise when using them as part of a full analysis pipeline.

9. Notice that to compute these rERPs, we have combined one intercept + slope rERP analysis for the stimulus-locked events with a second, separate intercept + slope rERP analysis for the response-locked events. This is accomplished by writing down the standard predictors for each of these two models, and then using the zero trick yet again to combine the two into a single model, which we fit using overlap correction as described above.

Baselining and Temporal Filtering

Baselining corrects for drift in the EEG signal by subtracting off the average potential measured over some prestimulus baseline period; temporal filtering smoothes away high-frequency noise that may be known to be irrelevant to some analysis of interest. Both are commonly used in ERP research. When working with traditional ERPs, these operations can be applied either to the individual epochs before averaging, or to the estimated ERP waveforms after averaging. It turns out that with averaging-based ERPs the results come out identical whichever order you use (Luck, 2005). Therefore, it's common to do averaging first—it's faster to baseline or filter an averaged waveform than many individual epochs, and, because filtering in particular has the potential to create harmful distortions, saving it for the end is useful because it allows one to try multiple filter settings, and apply it only where necessary (e.g., for display, but not statistical analysis; Luck, 2005).

Remarkably, in this respect, least squares rERPs estimated without overlap correction turn out to act just like those derived by averaging: you can baseline or filter either your original epochs or your rERP waveforms, and you get the same results either way (see supporting information for proof). Therefore, the same recommendations apply.

When using overlap correction, the equivalence theorem does not hold, but there still doesn't seem to be any reason to apply baselining or filtering first. (And since the continuous-time regression model replaces epoching, it's not clear how baselining before regressing would even work.) The simplest and safest approach still seems to be to apply these to estimated rERP waveforms only.

Baselining makes just as much sense for slope and nonlinear rERPs as it does for categorical rERPs; these estimates are just as likely to be thrown off by outliers caused by low-frequency drift.

Spatial Filtering

A number of useful transformations on EEG data can be interpreted as the application of linear spatial filters: spatial principal component analysis (PCA) or independent component analysis (ICA) decomposition, rereferencing, the Laplacian transformation, some forms of blink correction, etc. Similar to baselining and filtering, these have an identical effect whether they are applied to EEG data before regressing or afterwards to derived rERP waveforms. Unlike baselining and temporal filtering, this is true regardless of whether you use overlap correction. (See supporting information.)

Significance Testing

After estimating an ERP waveform, we must determine whether the patterns we think we see are real, or whether we're just looking at noise and fooling ourselves. Thus, we need ways to perform statistical significance testing.

Regression models come with a rich set of tools for testing hypotheses about β coefficients and combinations of β coefficients, and in rERP analysis, all our estimated and predicted waveforms are β coefficients and combinations of β coefficients. In principle, therefore, we should be able to use our rERP model to directly test null hypotheses like "at least one point within this latency window is nonzero." But, the most straightforward way of applying these statistical tests to rERP analyses turns out not to work. Least squares regression itself is an unbiased technique regardless of how our noise is distributed. But the textbook statistical tests for regression models require much stronger assumptions, including in particular the assumption that noise is uncorrelated across data points; that is, they require that knowing the amplitude of the EEG background activity at time t tells you nothing about its amplitude at time $t + 1$. This is very far from true, and thus more work will be needed before we can use rERP models to directly derive trustworthy parametric p values for activity spread over multiple time points (Smith, 2011). Another natural extension to the approach described here would be to estimate rERP waveforms using mixed effects regression to handle crossed random effects of both items and participants (Baayen, Davidson, & Bates, 2008; Clark, 1973), but this runs into a similar problem: these methods require distributional assumptions about how waveforms vary across items and participants, and understanding what assumptions are appropriate will require further investigation.

In the meantime, the traditional methods for testing significance of features in ERP waveforms remain just as valid for rERPs. Specifically, we can estimate per-participant rERPs, extract some feature from each waveform—such as average amplitude over a window, or peak latency—and then submit it to an analysis of variance (ANOVA) across participants. The waveforms used can be individual β waveforms, or combined prediction waveforms. This can handle arbitrary rERP designs (including those with overlap correction) and arbitrary features. It cannot account for crossed random effects, but neither can current ERP analysis methods.

In the special case where one wants to test mean amplitude over a latency window in an rERP design and is *not* using overlap correction, then there are three approaches that might seem reasonable:

1. As above, calculate rERP waveforms for each participant and electrode, baseline them, compute mean rERP amplitude over the window, then enter these into an ANOVA.
2. On a trial-by-trial basis within each participant and electrode, baseline each trial, calculate mean amplitude over the window, then perform a repeated measures regression across participants and electrodes (Dambacher, Kliegl, Hofmann, & Jacobs, 2006; Lorch & Myers, 1990).
3. On a trial-by-trial basis within each participant and electrode, baseline each trial, calculate mean amplitude over the window, then submit these to a mixed effects regression (Amsel, 2011; Frank, Otten, Galli, & Vigliocco, 2013).

Options 1 and 2 turn out to be mathematically identical (see supporting information). Options 2 and 3 are conceptually very similar: repeated measures regression/ANCOVA (analysis of covariance) and mixed effects models are two different statistical techniques for answering the same questions, with the latter being more modern, powerful, and prone to complications (Barr, Levy, Scheepers, & Tily, 2013). And, in this case, using only single data points derived from each baselined epoch minimizes the problems described above involving waveform covariation and correlations in the background noise. Thus, if handling crossed random effects is important and appropriate sphericity correction is available, there may be cases where it makes sense to use least squares to estimate rERPs for plotting but then use Option 3 for testing. In all other cases, the best bet for the moment is to stick to the traditional Option 1.

Artifact Rejection

A common problem in EEG analysis is the presence of artifacts such as blinks or muscle activity, which create large bursts of noise that may contaminate ERP/rERP estimates. Thus, it's common to

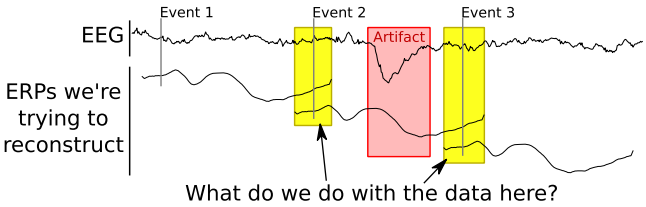

Figure 8. A schematic depiction of the challenge when combining artifact rejection and overlap correction. Top: EEG data we measured. Bottom: ERPs embedded in this EEG that we are trying to reconstruct. When the data contains an (uncorrectable) artifact, we must reject the artifact from our analysis. But if we reject event 2 as a whole, we must also reject the portions of its data that overlap with other events.

use some sort of method to identify and discard data containing such artifacts. When not using overlap correction, rERP analysis takes epoched data as input, and we can apply artifact rejection to these epochs just as we do with traditional averaging. When using overlap correction, things are more complicated.

Consider the situation depicted in Figure 8, where one epoch contains an artifact and also overlaps with other artifact-free epochs. The usual way of handling this in traditional ERP analysis would be to drop event 2, while keeping events 1 and 3. But we can't do that here; if we analyzed this data as if event 2 had not occurred, then it would cause event 2's effects in the marked regions to contaminate our analysis of events 1 and 3. (This is another example of how dropping a control variable creates inconsistent estimates.) Traditionally, rejecting an event and rejecting its data are the same thing. Here, the mapping between events and data is not so simple, and artifact rejection must be applied to data, not to events.

One option is to reject just the portion of the data where the artifact actually occurs (the red box in Figure 8), and continue to analyze the remainder of event 2's epoch, along with all of the epochs for events 1 and 3. Technically, this is easy to do; when using the continuous-time regression model for overlap correction, every data point enters as its own row in the regression, so we can reject an arbitrary subset of data points by simply dropping them from the regression. However, some might find it bothersome to reject part of an epoch while keeping the rest, and it may cause difficulties for baseline correction if different portions of our waveform are estimated from different subsets of our data and thus exposed to different levels of drift. If we require every epoch to either be in or out as a whole, then we must reject all of event 2's epoch, and all of the epochs that overlap it, and all of the epochs that overlap those, and so on. In some cases, like our go/no-go data, this is not bothersome, since we never have more than two events overlapping. In others, for example, a sentence-reading study where events are successive words presented in rapid serial visual presentation (RSVP), it would be unfortunate to have to throw out the entirety of every sentence that contains a blink anywhere within it, and we might prefer to reject partial epochs instead.

When working with data sets or software packages not originally intended for use with overlap correction, we may be able to determine which epochs contain artifacts, but not where within the epoch they occur. In this case, the best available option is probably to treat the whole epoch as if it were artifactual, and reject all the data it contains from all the epochs that it overlaps.

In general, these issues mean that the use of continuous regression will benefit from the use of artifact correction techniques such as blink correction (Joyce, Gorodnitsky, & Kutas, 2004) or ICA (Jung et al., 2000), the adoption of regression methods that are more robust to outliers (Maronna, Martin, & Yohai, 2006) and thus depend less on explicit artifact rejection, and the development of techniques for precisely detecting the location and extent of artifacts in continuous, nonepoched data.

Conclusion

Smith & Kutas (2015) introduced the fundamentals of the rERP framework, examined its relation to classic ERP averaging, and reviewed the theoretical results that guide us in considering trade-offs between different designs. Here, we've built on this foundation, and found that, in addition to handling classic ERP designs, the rERP framework straightforwardly generalizes to even complex analyses involving nonlinear effects and overlapping brain responses to temporally adjacent events. It also—with some minor adjustments—fits naturally into a traditional ERP analysis pipeline. In the process, we've found that the concepts introduced to understand trade-offs in simple linear designs—like the effects of partial and perfect collinearity—are also sufficient to understand even complicated situations—like the interplay between overlap correction and nonlinear effects in a combined analysis. rERP analysis unifies classic ERPs, difference ERPs, linear slope ERPs, dichotomization, ERP images, single-trial analysis of mean window amplitudes, and overlap correction into a single flexible and conceptually coherent system.

A list of free software packages implementing these methods is available at http://vorpus.org/rERP.

References

Amsel, B. D. (2011). Tracking real-time neural activation of conceptual knowledge using single-trial event-related potentials. *Neuropsychologia*, *49*, 970–983. doi: 10.1016/j.neuropsychologia.2011.01.003

Baayen, R. H., Davidson, D. J., & Bates, D. M. (2008). Mixed-effects modeling with crossed random effects for subjects and items. *Journal of Memory and Language*, *59*, 390–412. doi: 10.1016/j.jml.2007.12.005

Bach, D. R., & Friston, K. J. (2013). Model-based analysis of skin conductance responses: Towards causal models in psychophysiology. *Psychophysiology*, *50*, 15–22. doi: 10.1111/j.1469-8986.2012.01483.x

Barr, D. J., Levy, R., Scheepers, C., & Tily, H. J. (2013). Random effects structure for confirmatory hypothesis testing: Keep it maximal. *Journal of Memory and Language*, *68*, 255–278. doi: 10.1016/j.jml.2012.11.001

Boynton, G. M., Engel, S. A., Glover, G. H., & Heeger, D. J. (1996). Linear systems analysis of functional magnetic resonance imaging in human V1. *Journal of Neuroscience*, *13*, 4207–4221.

Burns, M. D., Bigdely-Shamlo, N., Smith, N. J., Kreutz-Delgado, K., & Makeig, S. (2013). Comparison of averaging and regression techniques for estimating event related potentials. Proceedings of the IEEE Engineering in Medicine & Biology Science Conference. Osaka, Japan.

Clark, H. (1973). The language-as-fixed-effect fallacy. *Journal of Verbal Learning and Verbal Behavior*, *12*, 335–359.

Cohen, M. S. (1997). Parametric analysis of fMRI data using linear systems methods. *NeuroImage*, *6*, 99–103.

Dale, A. M., & Buckner, R. L. (1997). Selective averaging of rapidly presented individual trials using fMRI. *Human Brain Mapping*, *5*, 329–340.

Dambacher, M., Kliegl, R., Hofmann, M., & Jacobs, A. M. (2006). Frequency and predictability effects on event-related potentials during reading. *Brain Research*, *1084*, 89–103. doi: 10.1016/j.brainres.2006.02.010

Frank, S. L., Otten, L. J., Galli, G., & Vigliocco, G. (2013). Word surprisal predicts N400 amplitude during reading. Proceedings of the

51st Annual Meeting of the Association for Computational Linguistics (pp. 878–883). Sofia, Bulgaria.
Hastie, T. J., & Tibshirani, R. J. (1990). *Generalized additive models*. New York, NY: Chapman and Hall.
Hendrix, P. (2009). Electrophysiological effects in language production: A picture naming study using generalized additive modeling (Unpublished Master's thesis). Radboud University, Nijmegen, The Netherlands.
Hendrix, P., Bolger, P., & Baayen, H. (2014). Distinct ERP signatures of word frequency, phrase frequency, and prototypicality in speech production. Manuscript submitted for publication.
Hinrichs, H., Scholz, M., Tempelmann, C., Woldorff, M. G., Dale, A. M., & Heinze, H.-J. (2000). Deconvolution of event-related fMRI responses in fast-rate experimental designs: Tracking amplitude variations [Supplement 2]. *Journal of Cognitive Neuroscience*, *12*, 76–89. doi: 10.1162/089892900564082
Joyce, C. A., Gorodnitsky, I. F., & Kutas, M. (2004). Automatic removal of eye movement and blink artifacts from EEG data using blind component separation. *Psychophysiology*, *41*, 313–325.
Jung, T.-P., Makeig, S., Humphries, C., Lee, T.-W., McKeown, M. J., Iragui, V., & Sejnowski, T. J. (2000). Removing electroencephalographic artifacts by blind source separation. *Psychophysiology*, *37*, 163–178.
Jung, T.-P., Makeig, S., Westerfield, M., Townsend, J., Courchesne, E., & Sejnowski, T. J. (2001). Analysis and visualization of single-trial event-related potentials. *Human Brain Mapping*, *14*, 166–185. doi: 10.1002/hbm.1050
Kryuchkova, T., Tucker, B. V., Wurm, L., & Baayen, R. H. (2012). Danger and usefulness in auditory lexical processing: Evidence from electroencephalography. *Brain and Language*, *122*, 81–91.
Laszlo, S., & Federmeier, K. D. (2011). The N400 as a snapshot of interactive processing: Evidence from regression analyses of orthographic neighbor and lexical associate effects. *Psychophysiology*, *48*, 176–186. doi: 10.1111/j.1469-8986.2010.01058.x
Laszlo, S., & Federmeier, K. D. (2014). Never seem to find the time: Evaluating the physiological time course of visual word recognition with regression analysis of single item ERPs. *Language, Cognition, and Neuroscience*, *29*, 642–661. doi: 10.1080/01690965.2013.866259
Lorch, R. F., & Myers, J. L. (1990). Regression analyses of repeated measures data in cognitive research. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *16*, 149–157.
Lorig, T. S., & Urbach, T. P. (1995). Event-related potential analysis using Mathematica. *Behavior Research Methods, Instruments, & Computers*, *27*, 358–366. doi: 10.3758/BF03200430
Luck, S. J. (2005). *An introduction to the event-related potential technique*. Cambridge, MA: MIT Press.
Maronna, R. A., Martin, R. D., & Yohai, V. J. (2006). *Robust statistics: Theory and methods*. New York, NY: Wiley.
Smith, N. J. (2011). Scaling up psycholinguistics (Doctoral dissertation, UC San Diego). Retrieved from http://gradworks.umi.com/34/72/3472953.html
Smith, N. J., & Kutas, M. (2015). Regression-based estimation of ERP waveforms: I. The rERP framework. *Psychophysiology*. Advance online publication, *52*, 157–168. doi: 10.1111/psyp.12317.
Smith, N. J., & Levy, R. (2013). The effect of word predictability on reading time is logarithmic. *Cognition*, *128*, 302–319.
Tremblay, A. (2009). Processing advantages of lexical bundles: Evidence from self-paced reading, word and sentence recall, and free recall with event-related brain potential recordings (Unpublished doctoral dissertation). University of Alberta, Edmonton, Alberta, Canada.
Tremblay, A., & Baayen, R. H. (2010). Holistic processing of regular four-word sequences: A behavioral and ERP study of the effects of structure, frequency, and probability on immediate free recall. In D. Wood (Ed.), *Perspectives on formulaic language: Acquisition and communication* (pp. 151–173). London, UK: Continuum.
Urbach, T. P., & Kutas, M. (2008). Cognitive aging: Evidence against a single factor [Abstract]. *Psychophysiology*, *45*, S113.
Wahba, G. (1990). *Spline models for observational data*. Philadelphia, PA: SIAM.
Woldorff, M. G. (1993). Distortion of ERP averages due to overlap from temporally adjacent ERPs: Analysis and correction. *Psychophysiology*, *30*, 98–119. doi: 10.1111/j.1469-8986.1993.tb03209.x
Wood, S. N. (2006). *Generalized additive models: An introduction with R*. Boca Raton, FL: Chapman and Hall/CRC.

(Received November 19, 2013; Accepted June 23, 2014)

Supporting Information

Additional Supporting Information may be found in the online verson of this article:

Appendix S1: Detailed simulations of overlap correction method.
Appendix S2: Derivations for baselining, filtering, windowing.
Figure S1: Intercept-only model, with jitter.
Figure S2: Intercept + slope model, no jitter.
Figure S3: Intercept + slope model, with jitter.
Table S1: Root mean squared error between estimated rERP and true rERP.

Supplementary information for *Regression-based estimation of ERP waveforms: II. Non-linear effects, overlap correction, and practical considerations*

Nathaniel J. Smith and Marta Kutas

S1 Detailed simulations of overlap correction method

The main text describes three simulated experiments used to test the overlap correction technique. In the first (*Intercept-only, with jitter*), a series of identical stimuli are presented, with ISI jittered between 200 and 400 ms in 16.7 ms intervals. In the second (*Intercept + slope, no jitter*), a fixed 300 ms ISI is used, and the ERP response to the stimuli varies according to a linear model:

$$\text{Stimulus ERP}_n = \text{Intercept rERP} + \text{Stimulus property}_n \times \text{Slope rERP}$$

The stimulus properties are sampled from an AR(1) model:

$$\text{Stimulus property}_n = 0.9 \times \text{Stimulus property}_{n-1} + \text{Normal}(\mu = 0, \sigma = 1)$$

The resulting stimulus properties have a mean value of zero, and strong trial-to-trial correlations; the correlation between trial n and trial $n+k$ is $r = 0.9^k$. The third experiment (*Intercept + slope, with jitter*), combines these manipulations; it uses the slope rERP model of experiment 2 and the jittered stimulus presentation schedule of experiment 1.

Fig. S1-S3 show the results of analyzing these simulated experiments on varying amounts of stimuli, both with and without overlap correction. Table S1 reports the root mean squared error (RMSE) between the estimated rERP and the true rERP for each of these conditions. To visualize sampling noise, the figures show results from five randomly selected simulation runs. RMSE calculations are based on estimates from 10,000 simulation runs. All waveforms were baselined before plotting or RMSE calculation.

In the *Intercept + slope, no jitter* condition (Fig. S2) we see that, as discussed in the main text, the intercept term cannot be reliably estimated with overlap correction enabled; we get extremely noisy results. But these waveforms are not entirely arbitrary. What's happening is, essentially, that the model knows very well what the steady-state response ought to look like, because it has plenty of data showing this. Then out of all the waveforms that, when overlapped, produce the same steady-state response, it chooses the ones which best match the first and last trials in our simulated experiment, since these are the only trials that are observed with only partial overlap. So while the model can identify the steady-state response, when it comes to uniquely picking out which of the waveforms that correspond to this steady-state response was actually present, we effectively have an N of $\sim$1. In the companion article, we discussed how removing confounded predictors from

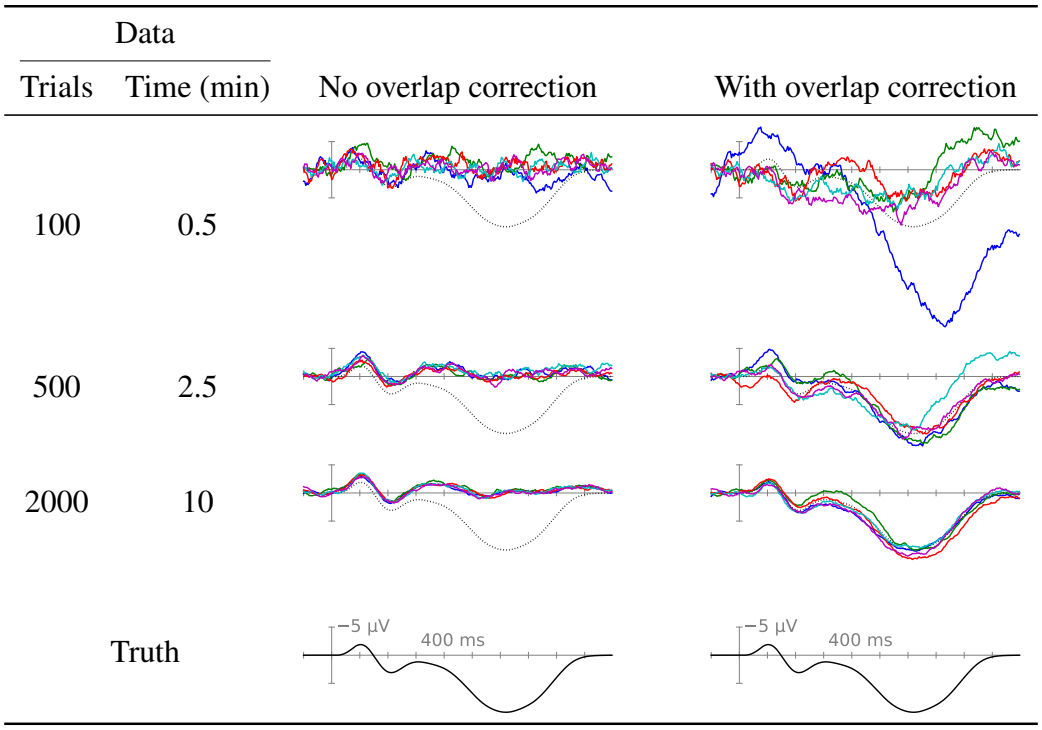

Figure S1: Intercept-only model, with jitter: Comparison between estimated waveforms from five simulated runs, and true waveforms.

an analysis may increase your power, but it does this by increasing your power to detect spurious effects. This is an example: when not using overlap corection, the intercept rERP is stable and reliably wrong. When using overlap correction (and thus correcting for the confounding effects of temporally adjacent events), the rERP estimate is highly variable, thus correctly and reliably indicating that the true underlying rERP cannot be estimated from this data.

Within this simulation, the intercept and slope rERPs are estimated simultaneously together within each model fit. Notice how these fits produce correct estimates of the slope rERPs, even while failing to estimate the intercept rERP. This gives an extreme example of how collinearity only affects estimates for the collinear predictors, and does not 'infect' other predictors within the same model.

Finally, notice that while our technique can use stimulus variability to successfully estimate the slope rERP even with a fixed ISI (Fig. S2), it does even better when it has both stimulus variability and jitter together (Fig. S3; see also Table S1, where in the right column, the *with jitter* values are systematically lower than *no jitter* values). This demonstrates how the least-squares fitting process automatically integrates all data available.

S2 Derivations for baselining, filtering, windowing

Here we give the mathematical derivations which show that many standard operations — including baselining, temporal and spatial filtering, and taking averages over latency windows — can be performed either on raw EEG before using least-squares regression to estimate rERPs, or else

Data		No overlap correction		With overlap correction	
Trials	Time (min)	Intercept rERP	Slope rERP	Intercept rERP	Slope rERP
100	0.5				
500	2.5				
2000	10				
Truth		−5 µV, 400 ms	−5 µV/a.u., 400 ms	−5 µV, 400 ms	−5 µV/a.u., 400 ms

Figure S2: Intercept + slope model, no jitter: Comparison between estimated waveforms from five simulated runs, and true waveforms.

Data		No overlap correction		With overlap correction	
Trials	Time (min)	Intercept rERP	Slope rERP	Intercept rERP	Slope rERP
100	0.5				
500	2.5				
2000	10				
Truth		−5 µV 400 ms	−5 µV/a.u. 400 ms	−5 µV 400 ms	−5 µV/a.u. 400 ms

Figure S3: Intercept + slope model, with jitter: Comparison between estimated waveforms from five simulated runs and true waveforms.

Table S1
Root mean squared error between estimated rERP and true rERP

		No overlap correction		With overlap correction	
Trials	Time (min)	Intercept rERP	Slope rERP	Intercept rERP	Slope rERP
		Intercept-only model, with jitter (Fig. S1)			
100	0.5	5.22		5.34	
500	2.5	5.13		1.79	
2000	10	5.09		0.78	
		Intercept + slope model, no jitter (Fig. S2)			
100	0.5	5.87	2.35	23.20	2.79
500	2.5	5.76	2.19	21.31	1.23
2000	10	5.73	2.18	25.80	0.79
		Intercept + slope model, with jitter (Fig. S3)			
100	0.5	5.25	2.15	5.71	1.83
500	2.5	5.14	1.99	1.80	0.70
2000	10	5.09	1.99	0.81	0.43

Note. Darker shading corresponds to larger error values.

on the resulting rERPs themselves, and the exact same result will be obtained either way. This theorem underlies the discussion of baselining and filtering in the main text. It also justifies the equivalence between 'option 1' and 'option 2' in the discussion of statistical significance testing, because repeated-measures regression works by first running a standard regression model across mean window amplitudes, and then submitting these to an ANOVA.

This is most clear if we switch to matrix notation for writing our regression model. Instead of

$$y_i = \beta_1 x_{1i} + \beta_2 x_{2i} + \cdots + \text{noise}_i$$

we can write the mathematically identical expression

$$\boldsymbol{y} = X\boldsymbol{\beta} + \textbf{noise}. \qquad (*)$$

Here we use the convention that column vectors are written in bold, while matrices are written in uppercase; the vectors are

$$\boldsymbol{y} = < y_1, \ldots, y_n >^T$$
$$\boldsymbol{\beta} = < \beta_1, \ldots, \beta_n >^T$$
$$\textbf{noise} = < \text{noise}_1, \ldots, \text{noise}_n >^T$$

and X is the *design matrix*, i.e., a matrix formed by putting the values of x_{1i} into the first column, x_{2i} into the second, and so on:

$$X_{ij} = x_{ji}.$$

The least-squares solution to equation $(*)$ can now be expressed using the so-called 'normal equations':

$$\hat{\boldsymbol{\beta}} = (X^T X)^{-1} X^T \boldsymbol{y}.$$

(Here the hat on the $\hat{\boldsymbol{\beta}}$ is to remind us that this value is our estimate of $\boldsymbol{\beta}$, and probably has a different value than the actual $\boldsymbol{\beta}$ that generated our data.)

One useful thing about this notation is that if we are performing m regressions with the same predictors but different data, then we can combine these into a single formula. Each regression has a different column vector of data $\boldsymbol{y}$. If we combine these to be the columns of a single large matrix Y, then we can write

$$Y = XB + \text{Noise}.$$

where B (the uppercase form of β) and Noise are now matrices. There's nothing particularly deep about combining the regression models in this way; it's basically just a notational trick, where we use the standard rules of linear algebra to express that we want to fit m regression models in parallel. And here, too, we can write a single formula for the least-squares solution to all of these regressions at once:

$$\hat{B} = (X^T X)^{-1} X^T Y.$$

Again, if you work through the matrix multiplications, this is just saying that the single-regression equation above should be applied to each column of Y separately.

But this final expression is very useful when it comes to thinking about how the rERP is affected by these operations. Recall that $\boldsymbol{y} = < y_1, \ldots, y_n >^T$ are the values of the EEG at a single electrode and latency but on different trials. Within a single electrode we have one $\boldsymbol{y}$ vector for each latency, and we can collect these together to make a single Y matrix for that electrode. If we order these column vectors by latency then — this is critical — each *row* of Y will contain the raw EEG data from a single epoch.

Baselining, temporal filtering, and taking the average over a window all have a critical attribute in common: they transform each epoch of data by applying a linear transformation to that epoch alone. So they take our matrix Y, and replace each row by a linear transformation of that row. That means that they can be written as a right-multiplication of our Y matrix by another matrix. For averaging over a window containing k points, this matrix would have a single column, with entries $< 0, \ldots, 0, 1/k, \ldots, 1/k, 0, \ldots, 0 >^T$. We can construct the matrix that performs baseline correction by starting with a matrix that has n columns, each of which computes the average of the baseline window, and subtracting it off from an $n \times n$ identity matrix. And filtering, which operates by replacing each point in an epoch with some linear combination of the points in that epoch, would use a matrix in which each column contained a shifted copy of the filter's impulse response function. In any case, let's call the matrix which performs the desired operation F.

If we first perform this transformation on the raw data and then enter these transformed values — YF, in matrix notation — into a regression, we get a matrix of new β values, which we denote $\hat{B}_F$:

$$\hat{B}_F = (X^T X)^{-1} X^T (YF).$$

Alternatively, we could first compute the matrix of rERP values, $\hat{B}$, and then apply our transformation to these β values directly, which in matrix notation would be written $\hat{B}F$.

So the claim is that it doesn't matter whether we average/filter our raw data, or instead average/filter our resulting rERP; that is, we claim that $\hat{B}_F = \hat{B}F$. But this is now easy to see, because matrix multiplication is associative, which allows us to rearrange parentheses:

$$\hat{B}_F = (X^T X)^{-1} X^T (YF) = \big((X^T X)^{-1} X^T Y\big)F = \hat{B}F.$$

If we are using the continuous-time regression model necessary for overlap correction (and actually using it in a non-trivial way, i.e., not just doing an analysis that could be equally well done by epoched regression), then the above proof does not go through, because now all the time points from each epoch go into a single column of Y. However, even in this case, we still regress each *electrode* separately. Therefore, the above proof still applies to spatial linear transformations, which can be treated as right multiplication of our Y matrix of data, where each column in Y has data from a single electrode.