Peer Reviewed

Title:
Scaling up psycholinguistics

Author:
Smith, Nathaniel J.

Acceptance Date:
01-01-2011

Series:
UC San Diego Electronic Theses and Dissertations

Degree:
Ph. D., UC San Diego

Permalink:
http://escholarship.org/uc/item/9hh0x7tq

Local Identifier:
b7183735

Abstract:
This dissertation contains several projects, each addressing different questions with different techniques. In chapter 1, I argue that they are unified thematically by their goal of 'scaling up psycholinguistics'; they are all aimed at analyzing large datasets using tools that reveal patterns to propose and test mechanism-neutral hypotheses about the brain's language processing architecture. In chapter 2, I investigate the well-known phenomenon that words which are more predictable in context are read faster than words which are not. I suggest that this is best understood as a special case of a phenomenon whereby more predictable events, whether linguistic or not, are in general processed more quickly than unpredictable ones, and propose a general model of why this happens. When combined with the constraints imposed by language's incrementally processed serial structure, this model predicts a logarithmic relationship between word probability and reading time, and I show that this in fact holds in two large data sets. In chapter 3, I turn to the question of how the brain produces fine- grained quantitative predictions; computationally, this seems impossible given the sparsity of the data it has available. This suggests it uses some kind of learning biases to guide generalization (a classic poverty of the stimulus argument), and I propose to study these biases by comparing subjective (cloze) probability with objective corpus probabilities. In Experiment 1, I find a number of candidate biases; in a series of follow-up experiments, I argue that some are probably artifactual, but others may in fact give clues to the brain's mechanisms for generating linguistic expectations from experience. Chapters 4 and 5 develop a methodology for extending ERP analysis to handle continuously varying stimulus attributes, partial confounding, and overlapping brain responses to events occurring in quick succession. These techniques are motivated by the desire to analyze EEG recorded in more naturalistic language paradigms (which contain all of the above challenges), but they are not specific to language studies, and have many potential applications for EEG/MEG research more generally

UNIVERSITY OF CALIFORNIA, SAN DIEGO

Scaling up psycholinguistics

A dissertation submitted in partial satisfaction of the
requirements for the degree
Doctor of Philosophy

in

Cognitive Science

by

Nathaniel J. Smith

Committee in charge:

Roger Levy, Chair
Marta Kutas, Co-Chair
Seana Coulson
Jeffrey L. Elman
Eric Halgren
Scott Makeig

2011

The dissertation of Nathaniel J. Smith is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

_____

_____

_____

_____

_____
Co-Chair

_____
Chair

University of California, San Diego

2011

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

Writing this thesis has been a long road with many twists and turns; it wouldn't have been possible without the assistance of many people. Foremost among these, of course, are my advisors, Roger Levy and Marta Kutas, who were always ready to give suggestions, critique, and support — thank you so much. There are dozens more people who I've learned from and enjoyed working with during these years at UCSD; it may take a village to raise a child, but in my case raising a thesis involved, at one time or another, half a dozen labs in three departments. I'm grateful to all of them, but I especially want to express my gratitude to the members of my committee, to the denizens of the Computational Psycholinguistics Lab, to the intrepid ERPers of the Kutas Lab (especially David Groppe, Tom Urbach, and Katherine Delong), and to the many fine folk at the Swartz Center for Computational Neuroscience. While less directly involved in this work, I also want to thank my early mentors, including Jerry Feldman, Srini Narayanan, and especially Eve Sweetser; more recently, I'm indebted to Rafael Núñez, Emo Todorov, and Ed Hutchins — they've influenced me more than they probably realize. Thanks are also owed to Ezra Van Everbroeck for patiently accomodating my habit of finding problems with the compute servers, usually on Sunday afternoons. And finally, I'd like to thank my family for their support, and — most of all — Shweta Narayan, whose been there since the beginning and has stuck with me anyway. You're wonderful. Thank you all.

Chapter 2, in part, has been submitted for publication. Smith, Nathaniel J.; Levy, Roger. The dissertation author was the primary investigator and author of this material.

Chapter 3, in part, is a revised version of the material as it appears in the proceedings of the 33rd annual conference of the Cognitive Science Society. Smith, Nathaniel J.; Levy, Roger, Cognitive Science Society, 2011. The dissertation author was the primary investigator and author of this paper.

Chapters 4 and 5, in full, are currently being prepared for submission for publication of the material. The dissertation author was the primary investigator and author of this material.

VITA AND PUBLICATIONS

2003 B. A. in Cognitive Science and Mathematics, University of California Berkeley.

2011 Ph. D. in Cognitive Science, University of California San Diego

Smith, N. J., & Levy, R. (2008). Optimal processing times in reading: a formal model and empirical investigation. In B. C. Love, K. McRae, & V. M. Sloutsky (Eds.), *Proceedings of the 30th Annual Meeting of the Cognitive Science Society* (pp. 595–600). Austin, TX: Cognitive Science Society.

Smith, N. J., Chan, W., & Levy, R. (2010). Is perceptual acuity asymmetric in isolated word recognition? Evidence from an ideal-observer reverse-engineering approach. In S. Ohlsson & R. Catrambone (Eds.), *Proceedings of the 32nd Annual Conference of the Cognitive Science Society* (pp. 1483–1488). Austin, TX: Cognitive Science Society.

Smith, N. J., & Levy, R. (2010). Fixation durations in first-pass reading reflect uncertainty about word identity. In S. Ohlsson & R. Catrambone (Eds.), *Proceedings of the 32nd Annual Conference of the Cognitive Science Society* (pp. 1313–1318). Austin, TX: Cognitive Science Society.

Kutas, M., DeLong, K., & Smith, N. J. (2011). A look around at what lies ahead: Prediction and predictability in language processing. In M. Bar (Ed.), *Predictions in the Brain*. Oxford, Oxford University Press.

Smith, N. J., & Levy, R. (2011). Cloze but no cigar: The complex relationship between cloze, corpus, and subjective probabilities in language processing. In L. Carlson, C. Hlscher, & T. Shipley (Eds.), *Proceedings of the 33rd Annual Conference of the Cognitive Science Society*. Austin, TX: Cognitive Science Society.

ABSTRACT OF THE DISSERTATION

Scaling up psycholinguistics

by

Nathaniel J. Smith

Doctor of Philosophy in Cognitive Science

University of California, San Diego, 2011

Roger Levy, Chair
Marta Kutas, Co-Chair

This dissertation contains several projects, each addressing different questions with different techniques. In chapter 1, I argue that they are unified thematically by their goal of 'scaling up psycholinguistics'; they are all aimed at analyzing large data-sets using tools that reveal patterns to propose and test mechanism-neutral hypotheses about the brain's language processing architecture.

In chapter 2, I investigate the well-known phenomenon that words which are more predictable in context are read faster than words which are not. I suggest that this is best understood as a special case of a phenomenon whereby more predictable events, whether linguistic or not, are in general processed more quickly than unpredictable ones, and propose a general model of why this happens. When combined with the constraints

imposed by language's incrementally processed serial structure, this model predicts a logarithmic relationship between word probability and reading time, and I show that this in fact holds in two large data sets.

In chapter 3, I turn to the question of how the brain produces fine-grained quantitative predictions; computationally, this seems impossible given the sparsity of the data it has available. This suggests it uses some kind of learning biases to guide generalization (a classic poverty of the stimulus argument), and I propose to study these biases by comparing subjective (cloze) probability with objective corpus probabilities. In Experiment 1, I find a number of candidate biases; in a series of follow-up experiments, I argue that some are probably artifactual, but others may in fact give clues to the brain's mechanisms for generating linguistic expectations from experience.

Chapters 4 and 5 develop a methodology for extending ERP analysis to handle continuously varying stimulus attributes, partial confounding, and overlapping brain responses to events occurring in quick succession. These techniques are motivated by the desire to analyze EEG recorded in more naturalistic language paradigms (which contain all of the above challenges), but they are not specific to language studies, and have many potential applications for EEG/MEG research more generally.

Chapter 1

Introduction

In this dissertation, I consider a number of questions whose connections are, perhaps, non-obvious.

In chapter 2, I investigate the well-known phenomenon that words which are more predictable in context are read faster than words which are not. I suggest that this is best understood as a special case of a phenomenon whereby more predictable events, whether linguistic or not, are in general processed more quickly than unpredictable ones, and propose a general model of why this happens. When combined with the constraints imposed by language's incrementally processed serial structure, this model predicts a logarithmic relationship between word probability and reading time, and I show that this in fact holds in two large data sets.

In chapter 3, I turn to the question of how the brain produces fine-grained quantitative predictions; computationally, this seems impossible given the sparsity of the data it has available. This suggests it uses some kind of learning biases to guide generalization (a classic poverty of the stimulus argument), and I propose to study these biases by comparing subjective (cloze) probability with objective corpus probabilities. In Experiment 1, I find a number of candidate biases; in a series of follow-up experiments, I argue that some are probably artifactual, but others may in fact give clues to the brain's mechanisms for generating linguistic expectations from experience.

Chapters 4 and 5 develop a methodology for extending ERP analysis to handle continuously varying stimulus attributes, partial confounding, and overlapping brain responses to events occurring in quick succession. These techniques are motivated by the

desire to analyze EEG recorded in more naturalistic language paradigms (which contain all of the above challenges), but they are not specific to language studies, and have many potential applications for EEG/MEG research more generally.

Why these projects, together in this thesis? To answer, let me digress to a now-classic paper of Newell (1973), who reviewed contemporary work in cognitive psychology, and expressed his concern that while many interesting phenomena were explored, the results merely accumulated. He asked: how can we proceed so that one day we will have, not dozens of little theories of particular phenomena, but one mature theory of our inner workings that is "adequate in power and commensurate with [our] complexity"? (p. 284)

This question is particularly apt for psycholinguists, since language and interaction is so central to our power and complexity as a species. We might imagine an adequate theory of low-level vision that made no reference to visual art — but what would it even mean to have a theory of "low-level language"? All of the complexity of human culture, individual development, and individual experience inheres in each human interaction. (And after all, what is culture but the accumulation of interactions?)

The chapters of this dissertation aren't bound by any single topic or methodology — but they all grow in some way out of my struggles with Newell's question.

At the core of his critique is our habit in psychology of generating binary oppositions — nature/nurture, serial/parallel, discrete/continuous, and so on and so forth — which are then argued back and forth as independent questions. It's this independence that bothered Newell the most; his prescription was to build unified models that would allow and require us to combine data from many ostensibly different experiments into a single whole, and he suggested production system models as the unifying tool.

When I look at these dichotomies, it's a different objection that first springs to mind. My experience is that even in typical, workaday computer programming, there are generally hundreds or thousands of mechanisms that could be used to accomplish any particular task. I would have no idea how to divide this space of solutions into simple categories like serial versus parallel or discrete versus continuous — they'd be some of one and some of the other, or, well, actually the distinction doesn't make sense in these cases, and so on down the list — they just wouldn't neatly fit. But the brain

isn't solving the kinds of algorithmically simple problems I'm thinking of, like table lookups and sorting lists — it's decoding the world, making decisions, and coordinating action. It's a supercomputer because it has to be. So the chances seem rather slim that any particular dichotomy will actually capture a useful generalization about the brain's mechanisms — and in fact, experience says that after a time, these dichotomies usually dissolve rather than resolve; the consensus conclusion is that it's some of one and some of the other, or, well, actually the distinction doesn't make sense in these cases...

None of which is to say that such oppositions can't be a useful tool for guiding our thinking. But this, I feel, is the greatest challenge we face in achieving Newell's dream: that there are far, far more viable theories than we can enumerate or articulate. So how will we ever find our way to the right ones?

Newell's motivation for advocating production systems instead of some other model were very simple: he "kn[ew] of only one model of the human control processes" (p. 301), and production systems were it. Yet in these more computationally sophisticated days, it's clear that there are far more ways to model the sequencing and coordination of operations — I can name a dozen broad classes just among the mainstream programming paradigms. If there are a thousand different ways the brain might solve some problem, and we pick one at random — well, we might get lucky. But it isn't likely. And perhaps there aren't a thousand, but a million, or more.

So for me, the idea of unification may be a valuable tool, but by itself it's no solution to the most urgent problem — there are still too many reasonable unified theories to identify the correct one. But what would help, then? I can't claim that my answers are novel, but I see three general steps we can take.

The first and most obvious is that we need more, bigger data. We could formalize this using information theory, but it's clear enough intuitively: if you want to distinguish between a million theories, you need a lot more data than if you just need to distinguish between two; in practice, this need is reflected by things like Bonferroni correction. Since big data is generally messy data, we'll need statistical techniques that let us handle that. And furthermore, to see the benefit of this data we'll need methods that let us extract more information from each data set than the binary opppositions that are typical of psychological statistics: bigger/smaller, interaction/no-interaction, related/unrelated.

You can't play twenty questions with nature and win — at least, not if you're asking yes/no questions. An example of this occurs in chapter 2, where the critical prediction is about the shape of a curve rather than its existence; novel methods allow us to extract this information from two large behavioral corpora.

Of course, more data by itself doesn't help if you don't know what questions to ask it, and even if we believe that there exist many viable theories, that doesn't mean we're smart enough to actually name and describe them all. So my second answer, related to the first, is that an even more important goal for new methodological tools is to help us find empirical patterns which can suggest new theories, or at least point us in the right direction. Our tools need to reveal patterns, not just test them. Chapter 3 provides an example of this usage, wherein we start with a very general question about how speakers generate their predictions about upcoming linguistic material, gather a large data set, ask the data what patterns we should be trying to explain, and then run follow-up experiments to test the resulting hypotheses.

Meanwhile, one of the great potential advantages of EEG over other psycholinguistic measures like eye-tracking is the fine temporal granularity and richness of the data that we acquire. In chapter 2, the central argument is based on how long people spend looking at words while reading. EEG potentially allows us a window into what they are doing *while* they are looking at these words. However, while corpus methods have become quite popular within behavioral psycholinguistics, the standard EEG analysis technique of ERP averaging is not able to handle analogous data sets. Chapters 4 and 5, therefore, are devoted to developing methods to allow us to analyze big messy EEG data-sets comparable to the large behavioral data-sets used in earlier chapters, and find the patterns in them.

So we need methods to distinguish between numerous theories, and we need methods to find these theories. My third suggestion is that we also need theories about the *space* of these theories — we need, if you will, meta-theories. Of course, we've always had these in some sense — the nature/nurture debate presupposes that all theories can be divided into one or the other camp — but perhaps we can do a better job of it. This is why I find it attractive to work at Marr's computational level, using, e.g., rational analysis approaches (Anderson, 1990; Oaksford & Chater, 1998): it lets us make

and test hypotheses about the overall shape of the language processing system, independently of any specific commitments to mechanism (which, let's face it, would certainly be wrong). A rational model is, in some sense, not itself a theory about the processing mechanisms, but a theory about what kind of theories about the processing mechanisms (i.e., the algorithmic and implementational levels) might work. This approach is seen in chapter 2, where we use rational arguments to postulate an extreme form of preparatory processing to explain predictability effects on processing times, and in chapter 3, where the initial question of how people make these predictions is motivated by computational considerations, and one of our results is that our participants seem to use methods similar to those that have been found particularly efficacious computationally. My focus on linguistic predictability in both cases also reflects this reasoning; the construct of predictability is meaningful without reference to any specific processing mechanisms.

Of course, each project also has its own logic, scientific questions, and results, and I certainly don't mean to imply that they even begin to exhaust the scope of an adequate theory of language. But one of the underlying motivations for each, the one that ties them together, is this dream that each takes a few fumbling steps towards — the dream of scaling up psycholinguistics to a theory whose power is commensurate with language's complexity.

References

Anderson, J. R. (Ed.). (1990). *The adaptive character of thought*. Hillsdale, New Jersey: Lawrence Erlbaum Associates.

Newell, A. (1973). You can't play 20 questions with nature and win: Projective comments on the papers of this symposium. In W. G. Chase (Ed.), *Visual information processing* (pp. 283–308). New York: Academic Press.

Oaksford, M., & Chater, N. (Eds.). (1998). *Rational models of cognition*. Oxford: Oxford University Press.

Chapter 2

A Rational Model of Predictability Effects on Cognitive Processing Times

Consider two similar sentences: *The children went outside to play* and *My brother came inside to play*. They have the same grammatical structure, and both contain the word *play*, used with approximately the same meaning. Yet something must be different, for native English speakers on average read *play* faster when it appears in the first sentence than the second. The difference is predictability (Ehrlich & Rayner, 1981; Kliegl, Nuthmann, & Engbert, 2006; McDonald & Shillcock, 2003a; Rayner & Well, 1996) — if we delete the word *play* and ask speakers to fill in the blank (Taylor, 1953), then their guesses on the first sentence are correct more often than on the second. It seems deeply intuitive that that more predictable items are processed more quickly, but it poses a theoretical conundrum. If the brain is capable of understanding the word *play* so quickly in one case, then why does it dawdle in others? And this result is not unique to human language comprehension; predictability regularly produces such speed-ups across diverse species and tasks (Carpenter & Williams, 1995; Froehlich, Herbranson, Loper, Wood, & Shimp, 2004; Janssen & Shadlen, 2005; Pang, Merkel, Egeth, & Olton, 1992) — but why?

Since the neural mechanisms involved presumably vary widely, explaining this pattern requires some way to generalize beyond the details of individual cognitive processes, which conventional theories of processing times are ill-equipped to do. Instead, we propose a rational-analysis approach to processing times. By 'rational', we mean

that we assume the brain has been sufficiently tuned through evolution that it functions as a near-optimal computational device, that on well-practiced tasks will find the best solution possible given the task's inherent constraints (Anderson, 1990; Oaksford & Chater, 1998). Previously, this approach has successfully explained behavior in such areas as sensory estimation (Ernst & Banks, 2002; Knill & Richards, 1996), decision making (Carpenter & Williams, 1995; Oaksford & Chater, 1994), and motor control (Harris & Wolpert, 1998; Todorov, 2004), but has been unable to explain *how long* it should take to *compute* such solutions. One might, after all, expect an 'optimal' computational device to perform all computations in near-zero time — but processing times are large and variable.

Previous rational theories have used two strategies to dodge this apparent contradiction. The first is to assume that there are various fundamental mental operations that inherently take some specified amount of time to perform, and restrict the rational analysis to finding the best way to combine these operations to accomplish higher-level tasks. For instance, Anderson and Milson (1989) assume that retrieving a single item from memory requires a fixed amount of time, and give a rational model of the order in which items should be retrieved so as to minimize the total time. Similarly, ACT-R models conventionally assume that executing a single production rule takes 50 ms (Anderson & Lebiere, 1998). But where do these numbers come from? In this approach they are inherently mysterious, arising from forces that are simply outside rationality's scope.

The second strategy is to model reaction time variability as arising from variations in the amount of perceptual data that must be acquired to accomplish some task (e.g., Carpenter & Williams, 1995; Laming, 1968; Norris, 2006, 2009; Stone, 1960). Since the external world provides data at its own pace, the only way to acquire this data is to wait for it to arrive, and if we need more data we must wait longer. These models have no place for reaction time effects to be driven by task-related variations in cognitive processing itself; the brain's computations are effectively assumed to occur instantaneously.

We tend to believe that cognitive processing times are both non-arbitrary and non-zero, which motivates the present work. Here, we present a third strategy, which

models variation in cognitive processing times as arising directly from rational optimization, without imposing the time required to perform mental operations as an extrinsic constraint.

2.1 A rational model of cognitive processing times

To bring cognitive processing times into the scope of rational analysis, we rely on one key idea: that any particular computation — whether it be a high-level task like understanding a word, or a lower-level operation like memory retrieval or production rule execution — can be neurally implemented in a wide variety of ways. Each possible implementation has an associated speed, and also, critically, some biologically imposed cost. When choosing an implementation, therefore, the brain must rationally balance the benefits of speed against the resources required to achieve it (Figure 2.1).

To motivate such a trade-off, consider the star-nosed mole. So far as we are aware, these moles hold the record for the fastest processing times ever observed in a complex sensorimotor task: they can detect a prey item and then program an appropriate tactile saccade in a scant $\sim$10 ms of central processing time (Catania & Remple, 2005). In comparison, humans seem terribly slow, typically requiring 100–150 ms just to program visual saccades. Presumably if our visual system were designed differently it could approach the mole's speed — but the reason these moles are so fast is that they survive by eating large numbers of very small prey items extremely quickly (Catania & Remple, 2005). Our survival does not depend so directly on the speed of our saccades, and so for us the resources required to achieve such speed would produce no commensurate payoff.

Our model's first claim is that such cases are typical, even on non-evolutionary timescales: that in general, task-level computation times are far from the physical limits of neural computation, and instead are determined by *a rational trade-off between the costs of constructing, maintaining, and using high-speed neural circuits versus the benefits of quick computation* — benefits ultimately determined by evolutionary fitness. These benefits will vary from task to task and situation to situation within the lifespan of an individual. For well-practiced behaviors, then, where the brain has the time, moti-

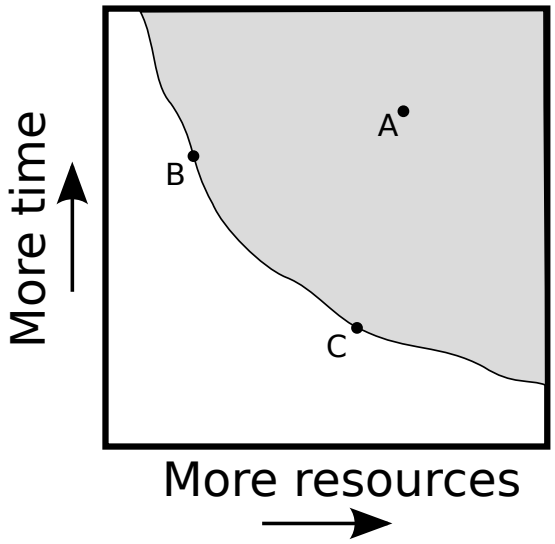

Figure 2.1: For any particular computation, there are many possible implementations, which are depicted here schematically as the shaded region. Each implementation achieves some speed and requires some expenditure of resources. While many combinations of speed and cost are possible, there are no implementations that are both extremely fast and extremely cheap. If we pick an implementation rationally, then we will never pick one like A, since there are other implementations (e.g., B and C) which are both faster and cheaper. But to decide between implementations on the frontier, like B and C, we must decide what relative value we place on speed versus resource expenditure. (Note that for simplicity, we assume that all implementations use differing methods which in the end accomplish identical computations; we are not proposing a model of speed-accuracy trade-offs. One could imagine extending this diagram with a third axis to indicate accuracy, but we do not explore this further here.)

vation, and experience needed to develop near-optimal computational techniques, it will allocate more computational resources to handling events for which speed is especially beneficial, and this will cause systematic, task-related differences in processing times.

Note that this claim means that two nearly identical stimuli — ones which require equally complex processing, and which do, in fact, receive equally complex processing — might still produce different reaction times solely because the relevant context gives the brain some reason to place a higher value on processing one of them quickly than the other.

But why would it be especially beneficial to process *play* quickly when it follows *The children went outside to*, but not when it follows *My brother came inside to*? Answering this question requires a second claim: that *at least some of the mechanisms by which the brain allocates resources are themselves slow enough that it would be useless to initiate them after the relevant event has occurred*. For instance, if we have a choice of either processing an event in 100 ms, or else allocating additional resources to process it in 50 ms, but this allocation will itself take 100 ms, then we might as well not bother. Any such resources can be effectively used only if they are allocated to an event before it occurs — the optimal strategy is to *prepare*. And intuitively, any resources we spend preparing are spent for good, but only produce a speed-up if spent on an event that does eventually occur; it is therefore better to invest resources preparing for events which are more likely to occur.

2.1.1 Deriving the generic predictability effect

To formalize this intuition, we define an overall cost function describing the trade-off between preparatory resource allocation and processing speed. Suppose that there are n events (e.g., the words of English) possible following a given context (like *The children went outside to...*). For each event i, the brain must determine what quantity of resources to allocate to this event before it occurs; we denote this quantity r_i. In reading, r_i summarizes the cost of such diverse processes as the long-term allocation of dedicated cortical circuitry to particularly common words and syntactic constructions, the medium-term maintenance in working memory of facts relevant to the current topic of conversation, and the immediate retrieval of the sensory form of specific anticipated

words. Our goal is to define a function $C(r_1, \ldots, r_n)$ to describe the total cost of some set of resource allocation choices $r_1, \ldots, r_n$; the rational resource allocation is the one which makes the cost $C(r_1, \ldots, r_n)$ as small as possible.

This cost needs two components, to capture the two competing priorities which are being trading off against each other. The first component is the cost of slow processing. To describe this, we need to know just how long it will take to process an event that has received any given quantity of resources — i.e., in order to decide between different points on the frontier shown in Figure 2.1, we need to know what the shape of this frontier actually is. This presumably varies for different tasks, so for now let us simply use $T(r)$ to denote the minimal time (in milliseconds) required to process an event which has received r resources. In Figure 2.1, B and C are two different points on the graph of $T(r)$, and Figure 2.2a shows two example forms that $T(r)$ might take. If the event that actually occurs is denoted A, then it will take $T(r_A)$ milliseconds to process.

The second component of the cost function is the cost of the resources being allocated, $r_1, \ldots, r_n$. For simplicity, we assume that these are measured on a scale where one unit of resources has the same value to the organism as one millisecond of additional speed in processing the event. In the end, the organism will only pay the processing time cost $T(r_A)$ for the event that actually occurs, but must pay the resource allocation costs $r_1, \ldots, r_n$ for all the potential events that it prepared for. Therefore, the total cost that will end up being paid to first prepare for and then process an event is:

$$C_{\text{actual}}(r_1, \ldots, r_n) = T(r_A) + \sum_{i=1}^{n} r_i.$$

However, in our model, resources must be allocated before A occurs, so the brain cannot optimize this actual cost, even in principle. What it can do is optimize the expectation of the actual cost, so that in the long run the total cost actually paid on average will be minimized. We assume access to a context-dependent probability distribution P over potential events, so that $P(A = i|\text{context})$ is the probability that i will be the event that actually occurs. The final cost function that we propose is

minimized by preparatory resource allocation is therefore

$$\begin{aligned} C(r_1, \ldots, r_n) &= E_P\big[C_{\text{actual}}(r_1, \ldots, r_n)\big|\text{context}\big] \\ &= E_P\Big[T(r_A) + \sum_{i=1}^{n} r_i\Big|\text{context}\Big] \end{aligned}$$

where $E_P[\cdot]$ denotes the expectation under the probability distribution P. Because $\sum_{i=1}^{n} r_i$ does not depend on A, we can pull it out of the expectation and simplify further:

$$\begin{aligned} &= E_P\big[T(r_A)\big|\text{context}\big] + \sum_{i=1}^{n} r_i \\ &= \sum_{i=1}^{n}\big[P(A = i|\text{context})T(r_i)\big] + \sum_{i=1}^{n} r_i \\ &= \sum_{i=1}^{n}\big[P(A = i|\text{context})T(r_i) + r_i\big]. \end{aligned}$$

This final formula has a simple structure, in which each event i makes two independent contributions to the total cost — its portion of the expected processing time, $P(A = i|\text{context})T(r_i)$, and its resource cost, r_i. These are depicted graphically in Figures 2.2b and 2.2c, respectively. Because the contributions of different events sum linearly, the combination of values for $r_1, \ldots, r_n$ which minimizes the total cost can be obtained by examining each r_i separately, and finding the value which minimizes its unique contribution to this total cost. This unique contribution, and its minimization, is illustrated in Figure 2.2d.

It is worth calling attention to a number of simplifying assumptions that we have made, and that a more nuanced model might relax. These include: (a) fast response is of equal value for each possible event; (b) if two events are allocated equal resources, then they produce equal processing times, so a single function $T(r)$ works for all events; (c) all information in the context is available in time to be used to optimize resource allocation; (d) resources spent preparing for one event give no benefits for other events; and (e) resource costs across possible events combine additively, with no limit on the

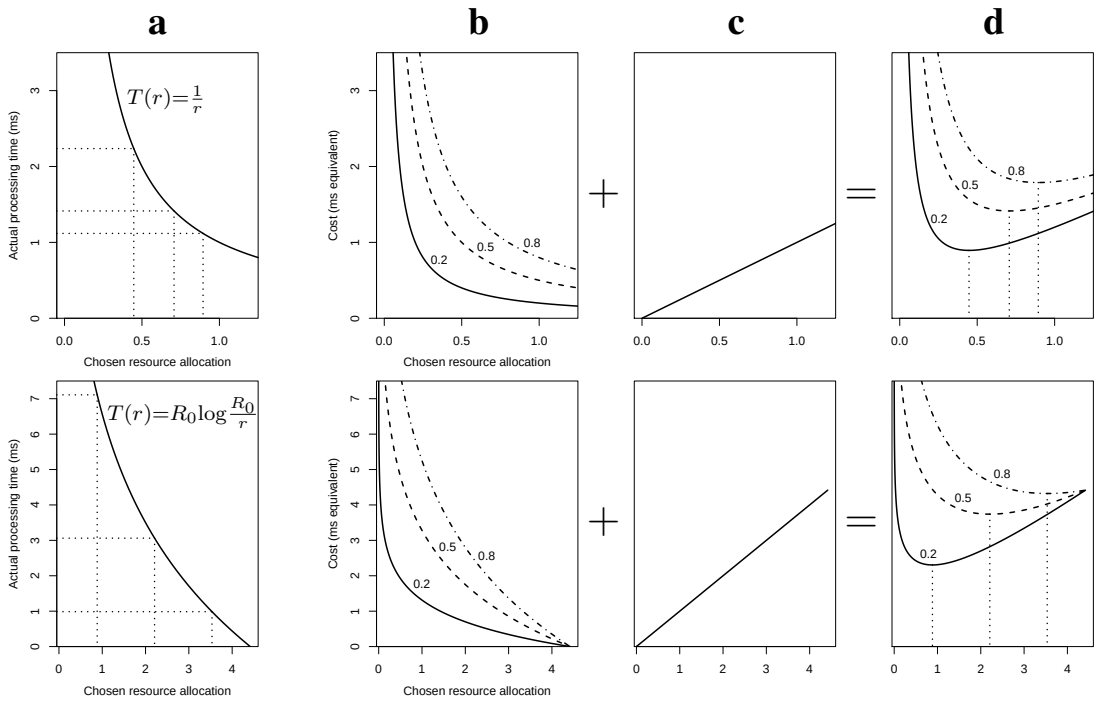

Figure 2.2: Finding the value of r_i which minimizes event i's unique contribution to the total expected processing cost. First row: $T(r) = 1/r$. Second row: $T(r) = R_0 \log(R_0/r)$. The first $T(r)$ is given purely as an example; it may or may not correspond to any real task. The second $T(r)$ is the one we derive for language processing, shown here for $R_0 = 4.4$ (estimated from self-paced reading data). The x-axis of each graph is the range of possible values that r_i may take. (**a**) Plots of $T(r)$. Note the similarity between these graphs and Figure 2.1. (Dotted lines are explained below.) (**b**) Plots of $P(A = i|\text{context})T(r_i)$, the contribution of each event to the expected processing time, for events with probability 0.2, 0.5, and 0.8. (**c**) Plots of r_i, the contribution of each event to the overall resource allocation cost. (**d**) The sum of **b** + **c** is the combined contribution of this event to the total cost. The rational value for r_i is given by the minimum of this function (dotted lines). For $T(r) = 1/r$ the events will be allocated 0.45, 0.71, and 0.89 units of resources respectively, and, as shown in **a**, these allocations would result in processing times of 2.2, 1.4, and 1.1 ms if such events are actually encountered. For $T(r) = R_0 \log(R_0/r)$, the events receive 0.9, 2.2, and 3.5 units of resources, and if encountered would be processed in 7.1, 3.1, and 1.0 ms.

total resources available. We leave a full examination of these issues for future work; our interest here is in deriving the effect of predictability on processing time.

To do this, we solve for the values of $r_1, \ldots, r_n$ that make the derivative of $C(r_1, \ldots, r_n)$ equal to zero; these are the values that minimize the total cost, and thus are rational. We denote these particular values as $r_1^*, \ldots, r_n^*$. When we take the derivative with respect to r_i, all but one of the terms in the summation disappear, leaving

$$\frac{\partial C(r_1, \ldots, r_n)}{\partial r_i} = P(A = i|\text{context})T'(r_i) + 1 = 0$$

where T' represents $dT(r)/dr$, the derivative of T. Therefore

$$r_i^* = \left(T'\right)^{-1}\left(\frac{-1}{P(A = i|\text{context})}\right). \tag{2.1}$$

where $(T')^{-1}$ represents the functional inverse of T' (not the reciprocal). The rational processing time t_i^* for stimulus i — a quantity more amenable to empirical measurement — is simply

$$t_i^* = T(r_i^*). \tag{2.2}$$

If we know the $T(r)$ that applies to some particular task, then Equations 2.1 and 2.2 give our model's corresponding predictions about resource allocations and processing times. For instance, if $T(r) = 1/r$, then we find by substitution and simplification that $r_i^* = \sqrt{P(A = i|\text{context})}$ and
$t_i^* = 1/\sqrt{P(A = i|\text{context})}$.

Note that this case, our model predicts the standard pattern of the processing time for an event t_i^* getting smaller as that event's predictability $P(A = i|\text{context})$ grows larger. This is not a coincidental feature of the particular example we chose; it holds for any $T(r)$ which is monotonic decreasing and strictly convex.[1] In plain English, this means that it holds whenever resource investment produces speed-ups in processing, but increasingly larger investments produce diminishing returns. This describes essentially any reasonable $T(r)$, which explains why predictability has a consistent effect across so

[1] See Appendix 2.A for proof, or, more intuitively, consider Figure 2.2, and how the curvature of the graph in the first column ultimately leads to the ordering of the minima in the last column.

many disparate tasks and species.

2.1.2 Deriving the form of the predictability effect in language comprehension

To make more detailed predictions, we need more information about $T(r)$'s precise form, which presumably varies depending on the situation, the computations required, and their underlying neural implementation. Therefore, we now focus on applying the model to natural language reading in particular, which as a test case has several attractive features. First, it is a skill that many adults have practiced heavily over multiple decades, and thus we can expect well-optimized behavior. Second, processing times can be measured using well-established methods (Just, Carpenter, & Woolley, 1982; Rayner, 1998). Finally, words in ordinary text exhibit a wide range of probabilities, allowing us to study contexts where some items are predicted with very low probability — predictions that would be difficult to induce by within-study training.

To apply our model to reading we need to answer two questions. First, what is the appropriate $T(r)$? In principle this could determined by measuring the benefits of fast understanding and the costs of fast computation, but this seems beyond our present capabilities. Second, at what temporal granularity does preparation — and its associated cost-benefit trade-off — occur? To apply our model to linguistic input, which unfolds sequentially over time, we must split that input up into discrete chunks which, we assume, are processed in sequence. But how should we make this division? By sentences? clauses? words? morphemes? letters/phonemes? or something else? If we turn to the literature to determine the granularity of language processing, then we realize that careful research has failed to find any units that are processed indivisibly: in auditory speech comprehension, where input truly is continuous in time, processing appears to be fully incremental, with even small amounts of phonetic material (e.g., the *ca–* in *candle*) processed immediately (Tanenhaus, Spivey-Knowlton, Eberhard, & Sedivy, 1995; Marslen-Wilson, 1975). Even in reading, input does not arrive nicely segmented—the eye regularly skips words, or views a single word in pieces over multiple fixations—and the core comprehension processes are presumably shared with spoken language. Put together, it seems we should not privilege *any* particular processing granularity. This is

only possible if we require a single law-like relationship between time and predictability to hold *simultaneously at all scales*.[2] Fortunately, it turns out that this requirement removes the need to measure $T(r)$ directly — there is only one form of $T(r)$ which is compatible with fully incremental processing of sequential input.

To see this formally, suppose that we have an event i (e.g., a word) which can be partitioned into a sequence of smaller events $i_1, \ldots, i_m$ (e.g., the phonetic segments in that word). If we define our model at the level of the word, then from Equations 2.1 and 2.2 we predict that the time to process word i will be

$$t_i^* = T\left(\left(T'\right)^{-1}\left(\frac{-1}{P(A = i|\text{context})}\right)\right).$$

Alternatively, if we define our model at segment granularity, we get predictions

$$t_{i_j}^* = T\left(\left(T'\right)^{-1}\left(\frac{-1}{P(A_j = i_j|\text{context}, i_1, \ldots, i_{j-1})}\right)\right).$$

The crucial point here is that both t_i^* and t_{ij}^* are functions only of the probabilities of i and i_j respectively—and that the function is the same in both cases. Next, we note that under the segment model, the total time to incrementally process word i is the sum of the times to process its parts, $\sum t_{i_j}^*$. Under the word model, of course, the time to process word i is t_i^*. If these quantities were to differ, then we could distinguish them empirically. The constraint imposed by incrementality, then, is that

$$t_i^* = \sum_{k=1}^{m} t_{i_j}^*$$

[2] If this were not so, then given a sufficiently large data set we would be able to compute probabilities at multiple granularities, and empirically determine the brain's language comprehension 'chunk size' — at least with respect to predictive processing — by checking which granularity best explained reading time behavior.

We can compute an analogous relationship for the probabilities: by the chain rule, we have

$$P(A = i|\text{context}) = \prod_{j=1}^{m} P(A_j = i_j|\text{context}, i_1, \ldots, i_{j-1}).$$

Or in short: times add and probabilities multiply, so the function which relates them must turn products into sums — and the only functions which have this property are logarithms. Since times are non-negative, the only way to satisfy the incrementality constraint is if

$$t_i^* = -R_0 \log P(A = i|\text{context})$$

for some free parameter $R_0 > 0$, where "log" denotes the natural logarithm. As shown in Appendix 2.B, this implies that

$$T(r) = R_0 \log \frac{R_0}{r}$$

where $0 \leq r \leq R_0$. This is the $T(r)$ which we suggest holds for language comprehension, and R_0 is interpretable as the quantity of resources needed to achieve instantaneous processing, $T(R_0) = 0$.

This reference to instantaneous processing might at first seem implausible, but note that as r asymptotically approaches R_0, $T(r)$ grows flatter, indicating that marginal increases in preparatory resource expenditure produce smaller and smaller returns in terms of processing time; such small returns are only justified if we are increasingly certain that the event will, in fact, need to be processed. If we plug our $T(r)$ into Equations 2.1 and 2.2, we find the rational resource allocation and subsequent processing time for for event i are

$$\begin{aligned} r_i^* &= R_0 \, P(i|\text{context}) \\ t_i^* &= -R_0 \log P(i|\text{context}). \end{aligned} \tag{2.3}$$

which means that the extreme, $r_i^* = R_0$ and $t_i^* = 0$, is predicted to occur only for an

event with probability one. This makes sense — if there is no uncertainty whatsoever about upcoming input then an organism can do all necessary processing in advance, and the input itself, when it does arrive, can be safely ignored. In reality, of course, perfect certainty is never achieved, so this model predicts that processing times for real input will always be greater than zero.

Equation 2.3 is useful for another reason — it shows that our general model, when applied to sequential input that is processed in a fully incremental fashion, makes a concrete, testable prediction: that the processing time for item i will be proportional to the logarithm of $P(A = i|\text{context})$, with $-R_0$ as the constant of proportionality.

2.2 Empirical validation

Previous authors have hypothesized such a logarithmic relationship between cognitive processing times and predictability under the moniker of *surprisal* (Hale, 2001; Levy, 2008), but not derived it from independently motivated principles such as rationality and incrementality. Nor has this hypothesis been tested; the difficulty and expense of accurately estimating word probabilities using traditional cloze norming (Taylor, 1953) have limited studies to examining only the upper range of probability ($\geq 10^{-2}$), primarily using small data-sets and factorial designs[3] (Ehrlich & Rayner, 1981; Rayner & Well, 1996; McDonald & Shillcock, 2003a, 2003b; Frisson, Rayner, & Pickering, 2005). As a result, while predictability's qualitative effect was first reported some decades ago, the quantitative form of this relationship has never been measured.

To avoid these limitations, we borrowed a probabilistic language model from the field of computational linguistics, and used it to algorithmically estimate word probabilities[4] in two large reading-time corpora. These corpora use different methodologies — one measures eye-movements and the other self-paced reading times — with different participant populations and different texts. The resulting data-sets are large enough to

[3]One exception is the work of Kliegl et al. (2006) and Kliegl (2007), but as their binning-based approach for assessing curve shape did not partial out confounds and was restricted to conditional word probabilities in the range 0.01–1, no clear picture emerged.

[4]Since we predict the same relationship holds regardless of the granularity we use to divide up the linguistic stream, we can pick whatever units we like, and using words facilitates the use of standard techniques like self-paced reading.

examine the fine-grained empirical relationship between probability and reading time over a wide range of probabilities.

2.2.1 Materials and Methods

Eye-tracking

Gaze durations (Rayner, 1998) were extracted from the English portion of the Dundee corpus (Kennedy, Hill, & Pynte, 2003), which records eye movements of 10 native speakers each reading 51,502 words of British newspaper text. Previous work (Demberg & Keller, 2008; Frank & Bod, 2011) has reported predictability effects in this corpus, but did not examine curve shape.

Self-paced reading

Moving-window self-paced reading (SPR) times (Just et al., 1982) were measured for 35 UCSD undergraduate native speakers each reading short (292–902 word) passages drawn from the Brown corpus of American English (2860–4999 total words per participant, mean 3912). 3 participants with comprehension-question performance at chance were excluded.

Probability estimation

Modified Kneser-Ney trigram word probabilities (Kneser & Ney, 1995; Chen & Goodman, 1998) were estimated from the British National Corpus (BNC Consortium, 2001) using SRILM v1.5.7 (Stolcke, 2002), and combined with a conditional bigram cache modeling word burstiness (Goodman, 2001). SPR analyses were adjusted for British/American spelling differences using VARCON (Atkinson, 2004).

Curve estimation

We used `mgcv` v1.6-2 (Wood, 2004, 2006) to predict reading times using penalized cubic spline functions (20 d.f.) of word log-probability, and as controls entered spline functions of position in text and the interaction of word length and log-frequency,

and factors indicating participant identity and (eye-tracking only) whether the previous word had been fixated. To capture spillover (Mitchell, 1984; Rayner, 1998), log-frequency/word-length interaction and probability terms were included for each word in an M-word window up to and including the current word, where M was chosen empirically to capture the effect present in each data set (eye-tracking: $M = 2$; SPR: $M = 4$). For further discussion of penalized spline smoothing, see Appendix 2.D. Total effect of probability was then estimated by summing effects over each word in this window. All words were analyzed except those at the beginning or end of a line, or for which some word in the window did not appear in the British National Corpus, appeared adjacent to punctuation, or contained numerals. Eye-tracking analyses excluded unfixated words; SPR analyses excluded outliers (reading times < 80 ms, > 1500 ms, or > 4 sd above participant-specific means). (Eye-tracking: $N = 166522$; SPR: $N = 51552$.) Fitting was by (penalized) least squares; confidence intervals were estimated by bootstrapping both participants and cases within participants, using the `mgcv` fitter's `weights` parameter to avoid replicating data across folds in its internal cross-validation routine. All reported results were robust to choice of spline basis, use of least squares estimation versus maximum likelihood estimation with the assumption of heavy-tailed (gamma-distributed) error, and the use of larger spillover windows (increased M).

Estimation of R_0 for reading

To produce the second row of Figure 2.2, we repeated the above regression analysis on the SPR data, but now constraining the word probability curves to be linear functions of the natural logarithm of word probability. Our estimate of R_0 is -1 times the sum of the resulting slopes.

2.2.2 Results

The total, unique effect of probability on reading time is shown in Figure 2.3. By 'total' we mean that these graphs show the total slowdown across the entire spillover region; this is the quantity to which our theory's predictions apply. By 'unique' we mean that these are corrected for the potential confounds detailed above. As predicted, the relationship between probability and reading time was found to be logarithmic over

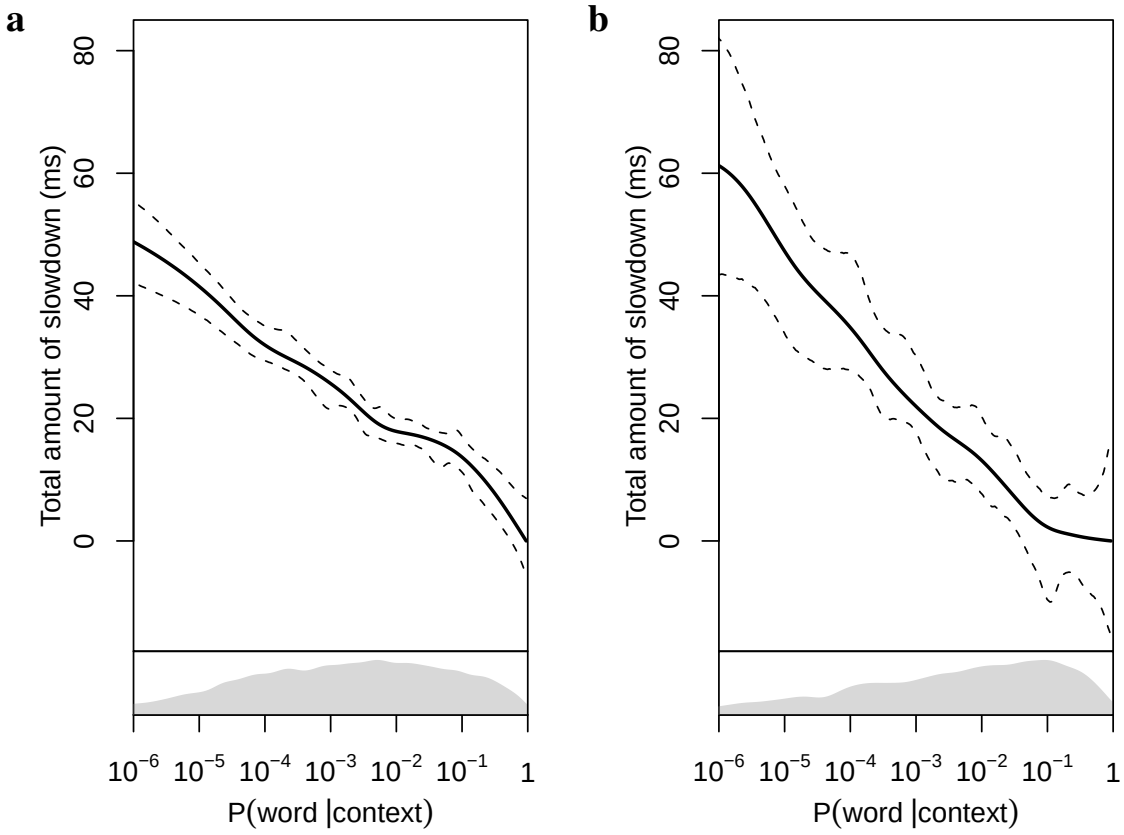

Figure 2.3: As predicted, the relationship between reading time and probability is approximately logarithmic. This relationship holds over at least six orders of magnitude. (Lower probability items occur, but in insufficient quantity to allow stable estimation — see Appendix 2.C. Lower panels show the amount of data available at each level of probability.) Curves are penalized splines with point-wise 95% confidence intervals, showing the total unique contribution of probability to reading time in first-pass gaze durations from the Dundee eye-tracking corpus (**a**) and self-paced reading times on a portion of the Brown corpus (**b**). By convention, we measure the effect of probability against the notional baseline of a perfectly predictable word. Confidence intervals do not include the uncertainty induced by measurement error in probability estimation. Note the general agreement on R_0 (curve slope) as measured by distinct methodologies.

the full usable range of our data, covering approximately six orders of magnitude, in both data sets.

To verify that this relationship also holds within individual participants, we repeated this analysis for each individual participant in the eye-tracking data. (The self-paced reading data contain insufficient observations per participant to allow such analysis.) As shown in Figure 2.4, the logarithmic effect is seen in at least 9 out of 10 participants.

This analysis also allows us to recover the time course of the predictability effect across the spillover region, as shown in Figure 2.5.

2.2.3 Discussion

We find these results very promising for our model. But before we can conclude much from them, we must consider alternate explanations for this logarithmic effect, and there are two that have previously been proposed in the literature. Neither, we argue here, provides a convincing account.

Relative entropy

Levy (2008) has argued that the goal of a comprehender is to build an internal model of their linguistic input — including the words, but more importantly, how they fit together and what they mean — in whatever representation the brain uses for such things. Furthermore, though the comprehender does not have enough information to uniquely pick out such a representation until the end of utterance (and possibly not even then), they would like to process words incrementally. A natural, mechanism-neutral way to represent the result of such incremental processing is as a probability distribution over whole-utterance representations. This probability distribution must be updated as each word arrives. Levy proposes that words which trigger 'large' changes in this distribution require more effort to process than words which trigger 'small' changes, and that Kullback-Leibler (KL) divergence be used to measure the size of such a change; in this theory, processing time for a word is assumed to be linearly proportional to the KL-divergence between old and new distributions. It turns out that this quantity equals the negative log-probability of that word (its surprisal).

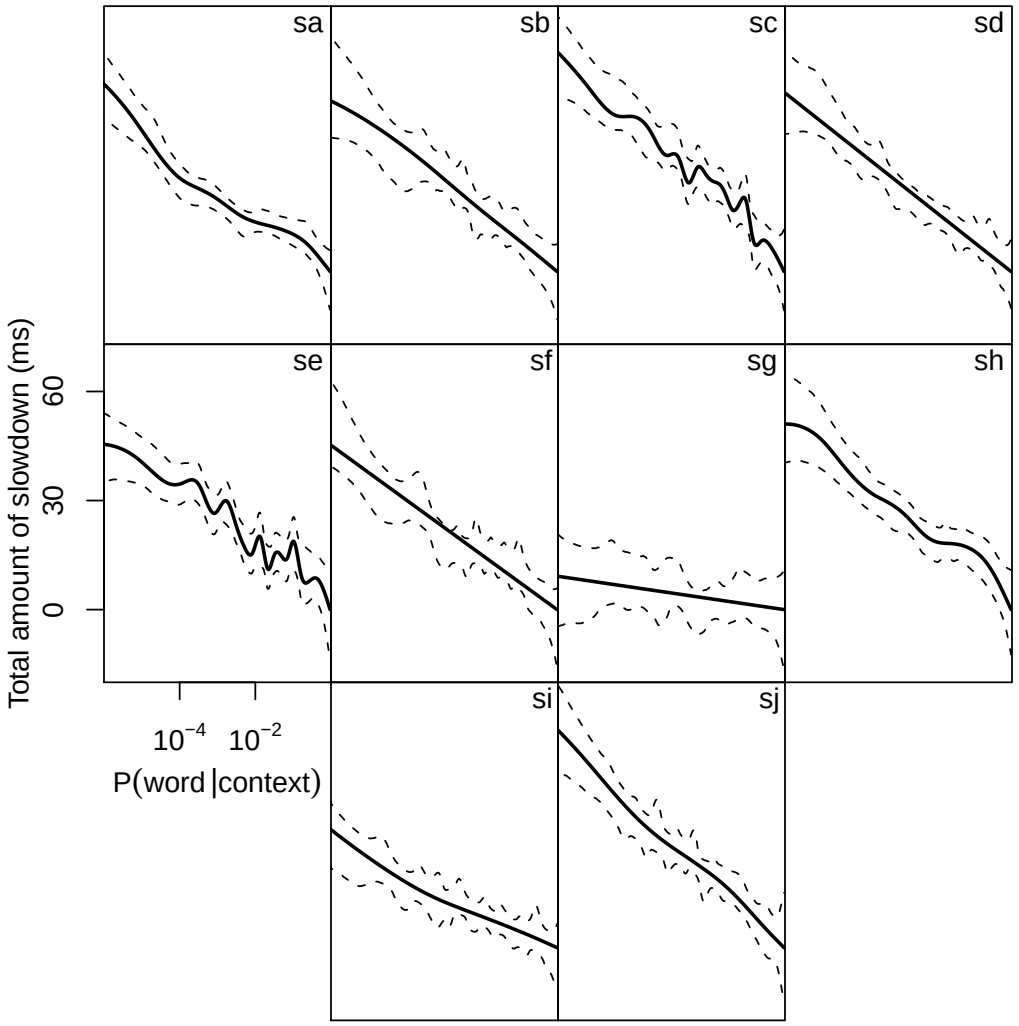

Figure 2.4: To visualize inter-individual variation, we break down the Dundee corpus data by participants and analyze each separately, shown here. Participant codes from the corpus are shown in the upper right of each panel. Dashed lines represent point-wise 95% confidence intervals. The variation in 'wiggliness' of the main curves results in part from noise and numerical instability in `mgcv`'s GCV-based penalization selection (Wood, 2004) allowing over-fitting in some cases (or possibly under-fitting in the case of participant 'sf') — see also Appendix 2.D. Despite this, 9 out of 10 participants show effects of log-probability with an overall linear trend, while no effect was found for participant 'sg'.

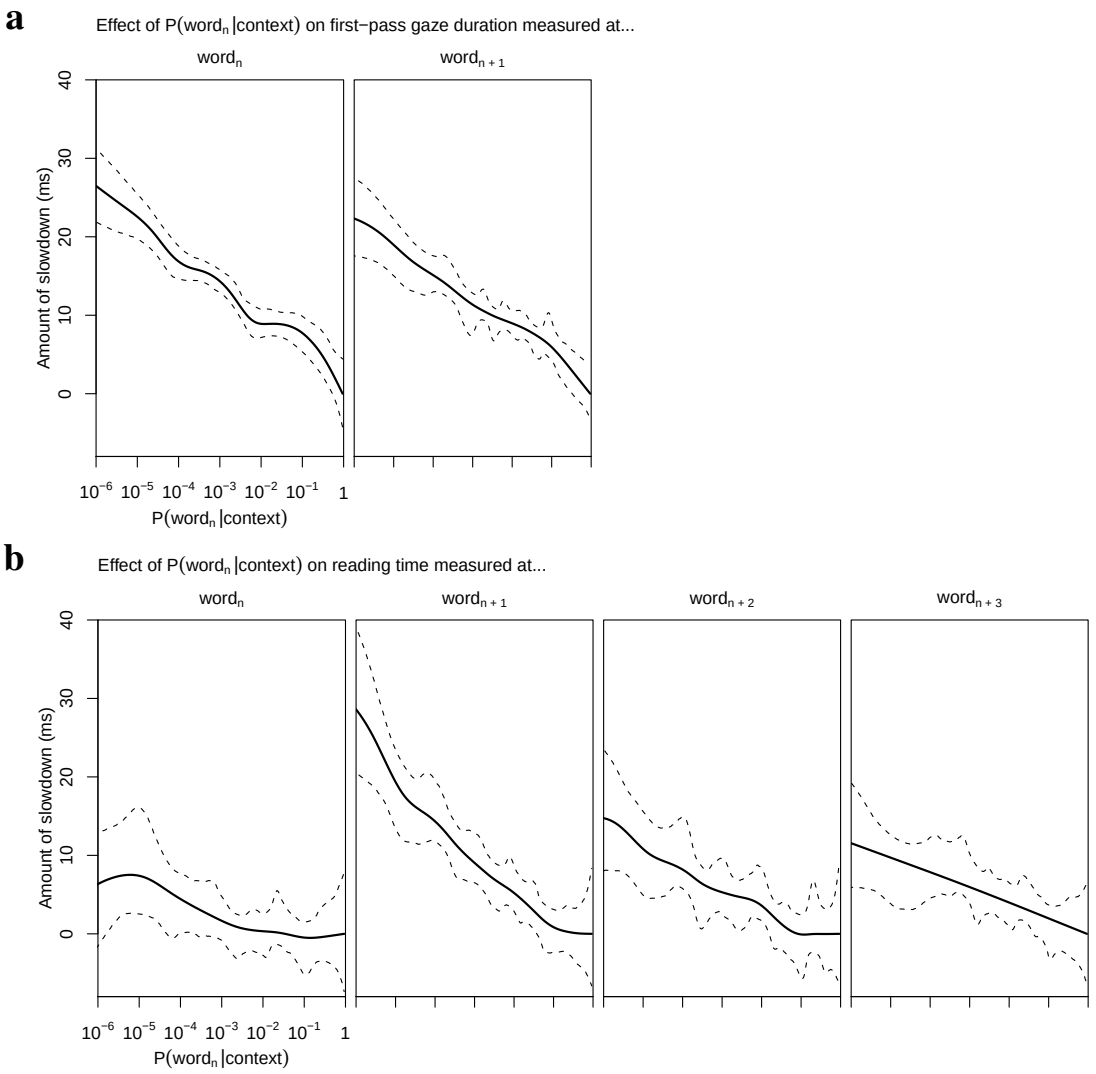

Figure 2.5: The same model fits shown in Figure 2.3, but showing the individual effects of the probability of word n on reading time measured at word n and on successive words (the spill-over region). Curves shown in Figure 2.3 were produced by summing the curves shown here. Dashed lines represent point-wise 95% confidence intervals. (**a**) First-pass gaze durations. (**b**) Self-paced reading times.

While this theory may well have merit as a conceptual model and motivation for future psycholinguistic work, our focus here is specifically on the striking empirical regularity of the predictability effect, and as an explanation of that regularity we believe this theory raises as many questions as it answers. To explain why predictability is measured in specifically log units rather than something else, it points out that this is the same as measuring the distance between certain probability distributions in specifically KL-divergence units, thus reducing one unexplained phenomena to another. Furthermore, it is unclear why an update that is larger in numerical magnitude should require a longer processing time.

Of course, the theory we propose based on rational resource allocation and incremental processing itself makes simplifying assumptions that might open it to similar criticism. But we believe that an explanation that explains aspects of linguistic behavior as arising, ultimately, from evolutionary fitness — and that also explains a pattern of behavior observed in many domains besides language — is fundamentally more satisfying than one which explains it only in terms of other theory-internal constructs.

The Bayesian Reader

Norris (2006, 2009) has proposed a model of word recognition as optimal perceptual discrimination: the Bayesian Reader. While the initial focus of Norris's work is on explaining results in isolated word recognition, the essential theory applies equally well to the reading of continuous text.

The Bayesian Reader is a rational model that uses the second strategy we referenced above; that is, it explains differences in reading times between words as being unrelated to cognitive processing, and instead as arising solely from sensory noise in the visual system. If the retina and visual system were able to exactly measure the light entering the eye, then a mere instant would suffice to recognize any word — but real sensory information contains noise. This noise can be reduced by taking multiple samples and combining them according to Bayesian principles: the posterior belief in the current word's identity is its prior probability of occurrence multiplied by the bottom-up likelihood of that word having produced each of the observed samples. This means that at any given level of desired accuracy, words with lower prior probability require more

samples to recognize, and as acquiring each sample requires the eyes stay fixed on the word for some time, words with lower probability will receive longer viewing times. The predictability effect thus arises naturally from the constraints of the visual system combined with principles of optimal Bayesian reasoning.

In fact, with no further assumptions, the model predicts a specifically logarithmic relationship between viewing time and prior probability. Intuitively, as each new sample arrives, the comprehender's posterior belief is updated by multiplying the old belief by the new sample's likelihood – or in log-space, by adding the new sample's log-likelihood. While samples are corrupted by stochastic noise, on average each sample contains about the same amount of information and has about the same likelihood, so on average, each sample increments the log-posterior by about the same amount. Therefore, if the log-posterior starts out at distance d from the recognition threshold, and each sample increments it by s on average, then threshold will be reached after about d/s samples. So viewing time is linearly proportional to d. But d is just the difference between the log-threshold and the log-prior, and accurate recognition requires a belief threshold of near one, which means the log-threshold is approximately zero. Thus d is, essentially, the negative log-prior, and viewing time is proportional to it.

This is an elegant model that accurately predicts several subtle results in isolated word recognition, particularly those involving task demands and the influence of visual neighbors (Pollatsek, Perea, & Binder, 1999), and we certainly believe that handling sensory uncertainty is an important component of language comprehension. However, it does not seem able to explain the results presented here.

Critically, in the Bayesian Reader, delay is caused by the visual system waiting to acquire new visual data; therefore, differential effects of probability should only occur at moments when the word is actually visible. The self-paced reading paradigm thus provides a useful test, because in this paradigm each word disappears from the screen as the next is shown. The optimal perceptual discrimination approach predicts that each viewing time should be affected only by the current word's predictability — there should be no spillover effects on the viewing time for following words. Yet, in our self-paced reading data (Figure 2.5b), almost all of the predictability effect appears on the following words; in fact, it isn't clear that predictability affects viewing time for the current word

at all. This is also compatible with results from the disappearing text paradigm used in eye-tracking, which show that even quite short fixations during ordinary reading are substantially longer than required to accumulate sufficient visual information to identify arbitrary words (Rayner, Liversedge, White, & Vergilino-Perez, 2003). It's unclear, therefore, why it would ever be necessary to alter the length of a fixation to acquire more visual information. Put together, these results suggest that the modulation of viewing durations predicted by optimal perceptual discrimination theory is unlikely to play more than a minor role in explaining the predictability effect observed for reading times.

2.3 Conclusion

Within psycholinguistics, our empirical result has important methodological implications for the interpretation of psycholinguistic data, since this ubiquitous reading-time effect turns out to be strongest and most variable for exactly those items whose conditional probability is too low to reliably measure or control by conventional means. Of more theoretical interest, the predictability effect's scale-free form also suggests that the various predictive processes which might be assumed to operate at the different levels of linguistic analysis — semantic/pragmatic, grammatical, lexical, and sublexical — produce effects which are interchangeable and indistinguishable.

More generally, our model predicts that these effects should take an similar logarithmic form for any domain in which an organism is highly skilled at incrementally processing sequentially accruing input. In other domains, we cannot make such detailed predictions without first deriving the appropriate relationship between resource investment and processing speed $T(r)$, and possibly extending the model to include other task-specific costs and benefits. But even in such cases, the model predicts the qualitative processing time advantage consistently observed for predictable events, and explains it as arising from a neural system capable of rationally tuning itself for greater processing speed via mechanisms which, crucially, both consume resources and are themselves somewhat slow.

2.4 Acknowledgements

This work was supported in part by NIH grant HD065829 to R.L., NIH grant T32-DC000041 to the Center for Research in Language, NIH grant T32-MH20002 to the Institute for Neural Computation, and funding from the Army Research Laboratories Cognition & Neuroergonomics Collaborative Technology Alliance. We thank: H. Lu and N. Katz for assistance in collecting self-paced reading data; V. Ferreira, T. F. Jaeger, R. Kliegl, M. Kutas, E. H. Levy, E. Mukamel, and K. Rayner for comments on the manuscript; and K. C. Catania and E. Todorov for valuable discussions.

Chapter 2, in part, has been submitted for publication. Smith, Nathaniel J.; Levy, Roger. The dissertation author was the primary investigator and author of this material.

2.A Derivation: Monotonicity of processing time

We claim that if $T(r)$ is a differentiable, strictly convex and monotonic decreasing function of r, then the rational processing time implied by Equations 2.1 and 2.2:

$$t_i^* = T\left(\left(T'\right)^{-1}\left(\frac{-1}{P(A=i|\text{context})}\right)\right)$$

is a monotonic decreasing function of $P(A = i|\text{context})$.

Proof: The derivative of a strictly convex function is a monotonic increasing function, and the inverse of a monotonic increasing function is also a monotonic increasing function. Therefore $(T')^{-1}(x)$ is a monotonic increasing function of x.

It is also clear that $-1/P(A = i|\text{context})$ is a monotonic increasing function of $P(A = i|\text{context})$.

The composition of two monotonic increasing functions is also monotonic increasing; therefore

$$(T')^{-1}\left(\frac{-1}{P(A=i|\text{context})}\right)$$

is a monotonic increasing function of $P(A = i|\text{context})$.

And finally, we know by assumption that $T(r)$ itself is monotonic decreasing.

Since the composition of a monotonic decreasing function with a monotonic increasing function is monotonic decreasing, we have that

$$T\left(\left(T'\right)^{-1}\left(\frac{-1}{P(A=i|\text{context})}\right)\right)$$

is monotonic decreasing. ■

2.B Derivation: $T(r)$ for logarithmic processing times

We wish to show that if

$$t_i^* = T\left(\left(T'\right)^{-1}\left(\frac{-1}{P(i|\text{context})}\right)\right) = -R_0 \log P(i|\text{context}) \tag{2.4}$$

then this implies that

$$T(r) = R_0 \log \frac{R_0}{r}.$$

However, the algebra turns out to be easier if we rewrite the problem in another form. Let $R(t)$ be the quantity of resources required to achieve a processing time of t. (Note that $R(t) = T^{-1}(t)$.) Our strategy will be to re-state our original optimization problem in terms of $R(t)$, and use this to re-derive Equation 2.4 in a new (and, it will turn out, simpler) form. We then solve this new equation to find an explicit formula for $R'(t)$, the derivative of $R(t)$ with respect to t, integrate it to find an explicit formula for $R(t)$, and then solve for $T(r)$.

First, we rewrite our original optimization problem into a formally equivalent problem: one where instead of selecting the quantity of resources $r_1, \ldots, r_n$ that we wish to spend, we select the processing times $t_1 = T(r_1), \ldots, t_n = T(r_n)$ that we wish to achieve:

$$C(t_1, \ldots, t_n) = \sum_{i=1}^{n} \big[P(A=i|\text{context})t_i + R(t_i)\big].$$

Finding the t_i^* which minimize this, we have

$$\frac{\partial C(t_1, \ldots, t_n)}{\partial t_i} = P(A = i|\text{context}) + R'(t_i^*) = 0$$
$$R'(t_i^*) = -P(A = i|\text{context})$$
$$t_i^* = (R')^{-1}\big(-P(A = i|\text{context})\big).$$

This shows that we can rewrite our premise above as

$$t_i^* = (R')^{-1}\big(-P(A = i|\text{context})\big) = -R_0 \log P(A = i|\text{context}).$$

We wish to solve for $R(t)$. Applying R' to both sides gives

$$-P(A = i|\text{context}) = R'\big(-R_0 \log P(i|\text{context})\big).$$

Now set $t = -R_0 \log P(i|\text{context})$, which implies that $P(A = i|\text{context}) = \exp(-t/R_0)$. Substituting, we find

$$-e^{-t/R_0} = R'(t)$$

and integrating both sides with respect to t gives

$$R(t) = R_0 e^{-t/R_0}.$$

But we want $T(r)$, which is the inverse of $R(t)$. So to find the inverse, we replace $R(t)$ with r and t with $T(r)$, and solve for $T(r)$ in

$$r = R_0 e^{-T(r)/R_0}$$

which gives

$$\frac{r}{R_0} = e^{-T(r)/R_0}$$
$$\log \frac{r}{R_0} = \frac{-T(r)}{R_0}$$

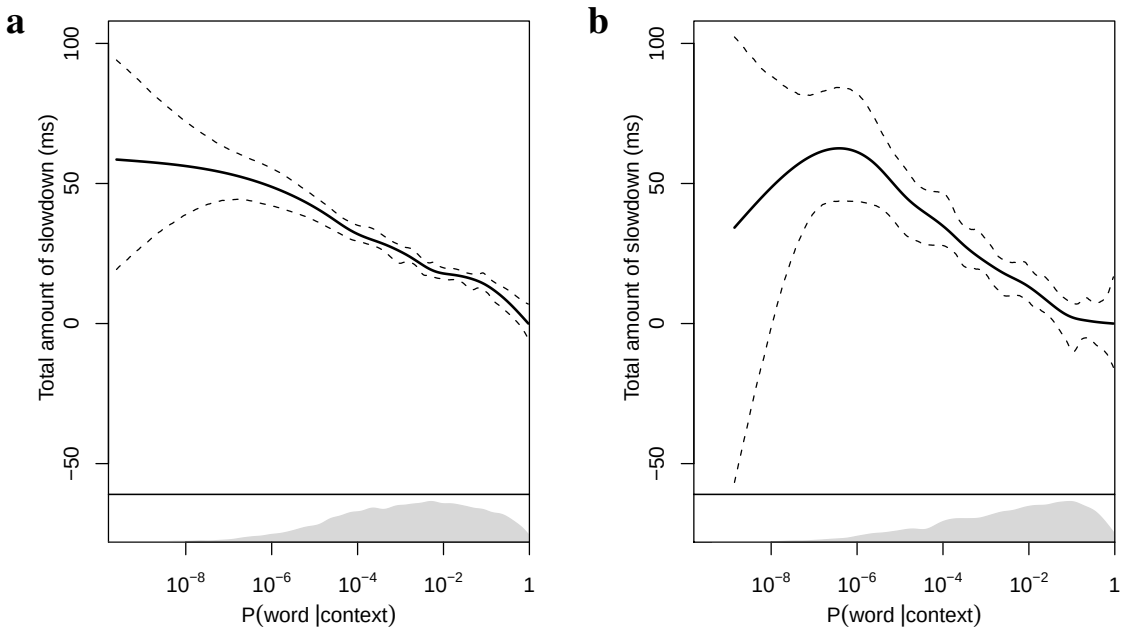

Figure 2.6: Here we reproduce Figure 2.3, but extending the x-axis to show the full range of probabilities. Data sparsity below about 10^{-7} (for gaze duration data, **a**) and about 10^{-6} (for self-paced reading data, **b**) cause a dramatic increase in the size of the confidence intervals, to the point that that very little can be said about the shape of the curve in this region. While our prediction of a straight line fit is compatible with the data, many other curves are as well.

$$T(r) = -R_0 \log \frac{r}{R_0} = R_0 \log \frac{R_0}{r}.$$

And this is the statement we wished to prove. ■

2.C Estimates for low probability words

Our figures in the main text do not show curve estimates for probabilities below 10^{-6}; Figures 2.6 and 2.7 demonstrate why.

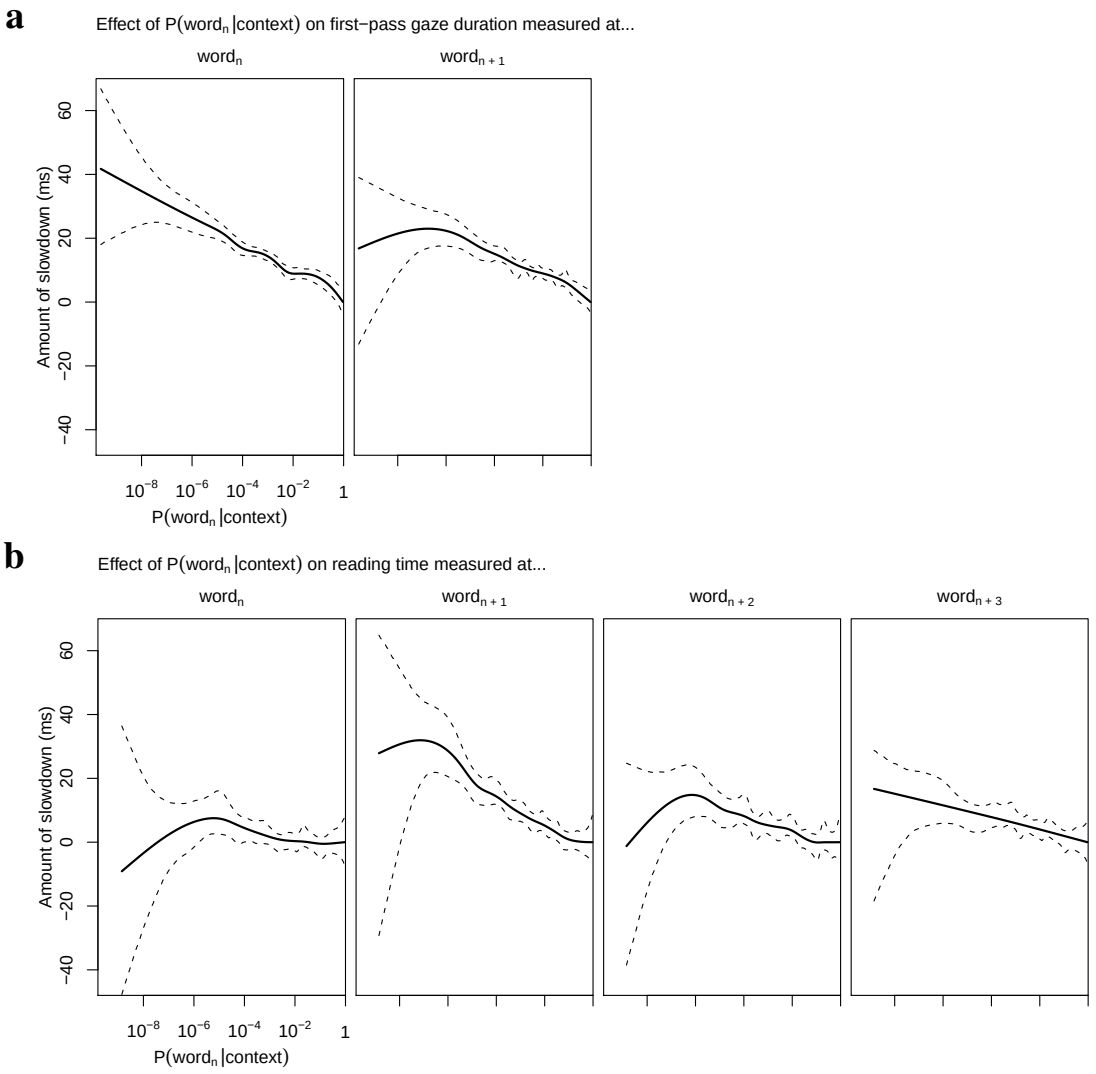

Figure 2.7: For completeness, we also show the individual word breakdowns across the full range of the data. (**a**) First-pass gaze durations. (**a**) Self-paced reading times.

2.D Validity of penalized spline regression

Penalized spline regression as implemented by `mgcv` (Wood, 2006, 2004) is a powerful and principled technique for estimating unknown non-linear relations. In order to fit nearly-arbitrary smooth curves, it uses a high-dimensional spline basis; in order to avoid the over-fitting that otherwise plagues high-dimensional models, it combines the standard maximum likelihood criterion with a curvature penalty term that biases the regression towards less 'wiggly' curves. Critically, the relative weight placed on the likelihood term (which attempts to follow the data) versus the penalty term (which attempts to make the line smoother and closer to a straight line) is determined by cross-validation. In theory, therefore, this method's fitted curves should be biased towards smoothness only to the extent that this helps it better match the true curve describing the underlying phenomenon.

But since our key empirical finding is that such a fit produces a straight line, it seems wise to verify that this is not an artifact introduced by penalization. We therefore repeated our analysis, but using two different methods to remove this potential bias.

First, we ran the same model fits, but entering raw probability in place of log-probability; in this case we predict that the splines should attempt to form a steep logarithmic curve (since we believe that is the true underlying relationship), while the penalization pushes towards a straight line. As expected, `mgcv`'s algorithm chose to apply very small penalization weights (ranging from 65 to 64000 times smaller than the corresponding weights chosen in the original analyses), which in turn allowed the resulting spline fit to form an highly-nonlinear, approximately logarithmic curve with substantial local variation around this underlying trend (Figure 2.8b; note that while the fit was performed using *raw* probability, we plot the result against *log* probability to facilitate comparison with other fits). Second, we fit our original model, but with penalization simply disabled; this produced similar results (Figure 2.8c). The three models thus agree that the underlying relationship is approximately logarithmic; next we would like to confirm that the local non-linear deviations from this trend that we see in models (b) and (c) are the result of over-fitting rather than a true effect. We verified this by performing 1000-fold cross-validation on all three models and both data sets, and found that in all cases, the original penalized model (Figure 2.8a) achieved a higher log-likelihood

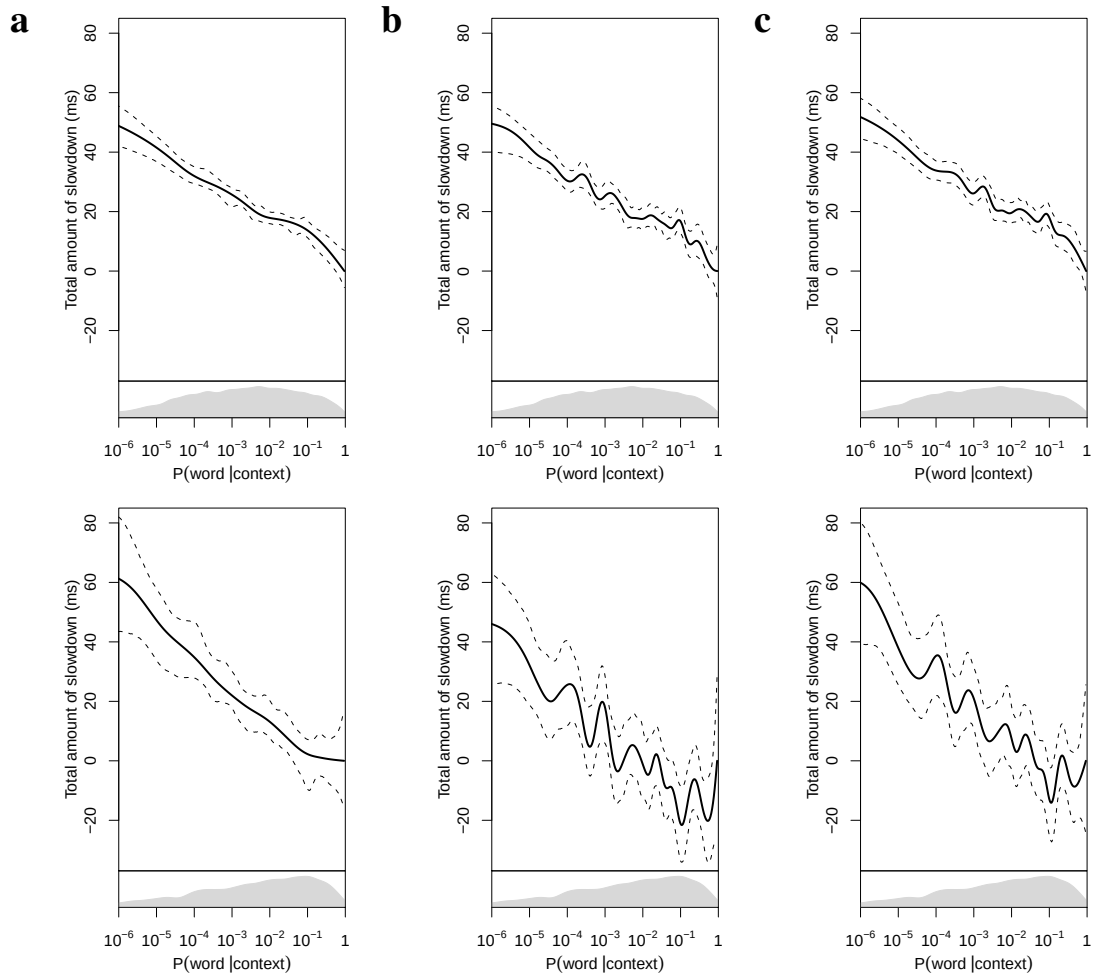

Figure 2.8: The effect of penalization in controlling over-fitting. (**a**) Our original, penalized model (a repeat of Figure 2.3). (**b**) The same model as in **a**, but fit with raw probability entered instead of log probability, then plotted in log-space. (**c**) The same model as in **a**, but fit without penalization. Upper panels show first-pass gaze durations; lower panels show self-paced reading times. That the lower panels show more wiggliness than the upper ones is presumably due to the relative sizes of the two data sets; in the absence of penalization, the smaller data set allows more overfitting than the larger. Dashed lines denote point-wise 95% confidence intervals.

than the other models on held-out data. Thus we conclude that, to the limits of our data, the underlying relationship between word probability and processing time is in fact logarithmic.

References

Anderson, J. R. (Ed.). (1990). *The adaptive character of thought*. Hillsdale, New Jersey: Lawrence Erlbaum Associates.

Anderson, J. R., & Lebiere, C. (1998). *The atomic components of thought*. Mahwah, New Jersey: Lawrence Erlbaum Associates.

Anderson, J. R., & Milson, R. (1989). Human memory: An adaptive perspective. *Psychological Review*, *96*, 703–719.

Atkinson, K. (2004). *The VARCON database, version 4.1*. Retrieved from http://wordlist.sourceforge.net/

BNC Consortium. (2001). *The British National Corpus* (version 2, BNC World) [Text corpus]. Retrieved from http://www.natcorp.ox.ac.uk/

Carpenter, R. H. S., & Williams, M. L. L. (1995). Neural computation of log likelihood in control of saccadic eye movements. *Nature*, *377*(6544), 59–62.

Catania, K. C., & Remple, F. E. (2005). Asymptotic prey profitability drives star-nosed moles to the foraging speed limit. *Nature*, *433*(7025), 519–522.

Chen, S. F., & Goodman, J. (1998). *An empirical study of smoothing techniques for language modeling* (Technical Report No. TR-10-98). Cambridge, MA: Computer Science Group, Harvard University. Retrieved from ftp://ftp.deas.harvard.edu/techreports/tr-10-98.ps.gz

Demberg, V., & Keller, F. (2008). Data from eye-tracking corpora as evidence for theories of syntactic processing complexity. *Cognition*, *109*(2), 193–210.

Ehrlich, S. F., & Rayner, K. (1981). Contextual effects on word perception and eye movements during reading. *Journal of Verbal Learning and Verbal Behavior*, *20*(6), 641–655.

Ernst, M. O., & Banks, M. S. (2002). Humans integrate visual and haptic information in a statistically optimal fashion. *Nature*, *415*(6870), 429–433.

Frank, S. L., & Bod, R. (2011). Insensitivity of the human sentence-processing system to hierarchical structure. *Psychological Science*, *22*(6), 829–834.

Frisson, S., Rayner, K., & Pickering, M. J. (2005). Effects of contextual predictability and transitional probability on eye movements during reading. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *31*(5), 862–877.

Froehlich, A. L., Herbranson, W. T., Loper, J. D., Wood, D. M., & Shimp, C. P. (2004). Anticipating by pigeons depends on local statistical information in a serial response time task. *Journal of Experimental Psychology: General*, *133*(1), 3145.

Goodman, J. T. (2001). *A bit of progress in language modeling, extended version* (Technical Report No. MSR-TR-2001-72). Redmond, WA: Microsoft Research.

Hale, J. (2001). A probabilistic Earley parser as a psycholinguistic model. In *Proceedings of NAACL-2001* (pp. 159–166). Stroudsburg, PA: Association for Computational Linguistics.

Harris, C. M., & Wolpert, D. M. (1998). Signal-dependent noise determines motor planning. *Nature*, *394*, 780–784.

Janssen, P., & Shadlen, M. N. (2005). A representation of the hazard rate of elapsed time in macaque area LIP. *Nature Neuroscience*, *8*(2), 234–241.

Just, M. A., Carpenter, P. A., & Woolley, J. D. (1982). Paradigms and processes in reading comprehension. *Journal of Experimental Psychology: General*, *111*(2), 228–238.

Kennedy, A., Hill, R., & Pynte, J. (2003). The Dundee corpus. In *Proceedings of the 12th European conference on eye movement.*

Kliegl, R. (2007). Toward a perceptual-span theory of distributed processing in reading: A reply to Rayner, Pollatsek, Drieghe, Slattery, and Reichle (2007). *Journal of Experimental Psychology: General*, *136*(3), 530–537.

Kliegl, R., Nuthmann, A., & Engbert, R. (2006). Tracking the mind during reading:

The influence of past, present, and future words on fixation durations. *Journal of Experimental Psychology: General*, *135*(1), 1235.

Kneser, R., & Ney, H. (1995). Improved backing-off for M-gram language modeling. In *Proc. ICASSP* (pp. 181–184).

Knill, D. C., & Richards, W. (Eds.). (1996). *Perception as Bayesian inference*. Cambridge: Cambridge University Press.

Laming, D. R. J. (1968). *Information theory of choice-reaction times*. London: Academic Press.

Levy, R. (2008). Expectation-based syntactic comprehension. *Cognition*, *106*, 1126–1177.

Marslen-Wilson, W. D. (1975). Sentence perception as an interactive parallel process. *Science*, *189*(4198), 226–228.

McDonald, S. A., & Shillcock, R. C. (2003a). Eye movements reveal the online computation of lexical probabilities during reading. *Psychological Science*, *14*(6), 648–652.

McDonald, S. A., & Shillcock, R. C. (2003b). Low-level predictive inference in reading: The influence of transitional probabilities on eye movements. *Vision Research*, *43*, 1735–1751.

Mitchell, D. C. (1984). An evaluation of subject-paced reading tasks and other methods for investigating immediate processes in reading. In D. E. Kieras & M. A. Just (Eds.), *New methods in reading comprehension research.* Hillsdale, New Jersey: Lawrence Erlbaum Associates.

Norris, D. (2006). The Bayesian reader: Explaining word recognition as an optimal Bayesian decision process. *Psychological Review*, *113*(2), 327–357.

Norris, D. (2009). Putting it all together: A unified account of word recognition and reaction-time distributions. *Psychological Review*, *116*(1), 207–219.

Oaksford, M., & Chater, N. (1994). A rational analysis of the selection task as optimal data selection. *Psychological Review*, *101*, 608–631.

Oaksford, M., & Chater, N. (Eds.). (1998). *Rational models of cognition*. Oxford: Oxford University Press.

Pang, K., Merkel, F., Egeth, H., & Olton, D. S. (1992). Expectancy and stimulus frequency: a comparative analysis in rats and humans. *Perception and Psychophysics*, *51*(6), 607–615.

Pollatsek, A., Perea, M., & Binder, K. S. (1999). The effects of "neighborhood size" in reading and lexical decision. *Journal of Experimental Psychology: Human Perception and Performance*, *25*(4), 1142–1158.

Rayner, K. (1998). Eye movements in reading and information processing: 20 years of research. *Psychological bulletin*, *124*(3), 372–422.

Rayner, K., Liversedge, S. P., White, S. J., & Vergilino-Perez, D. (2003). Reading disappearing text. *Psychological Science*, *14*(4), 385–388.

Rayner, K., & Well, A. D. (1996). Effects of contextual constraint on eye movements in reading: A further examination. *Psychonomic Bulletin & Review*, *3*(4), 504–509.

Stolcke, A. (2002). SRILM — an extensible language modeling toolkit. In *Proc. intl. conf. on spoken language processing* (Vol. 2, pp. 901–904). Denver.

Stone, M. (1960). Models for choice-reaction time. *Psychometrika*, *25*, 251–260.

Tanenhaus, M. K., Spivey-Knowlton, M. J., Eberhard, K. M., & Sedivy, J. C. (1995). Integration of visual and linguistic information in spoken language comprehension. *Science*, *268*, 1632–1634.

Taylor, W. L. (1953). "Cloze procedure": A new tool for measuring readability. *Journalism Quarterly*, *30*, 415–433.

Todorov, E. (2004). Optimality principles in sensorimotor control. *Nature Neuroscience*, *7*(9), 907–915.

Wood, S. N. (2004). Stable and efficient multiple smoothing parameter estimation for generalized additive models. *Journal of the American Statistical Association*, *99*, 673–686.

Wood, S. N. (2006). *Generalized additive models: An introduction with R*. Boca Raton: Chapman and Hall/CRC.

Chapter 3

Cloze but no cigar: The complex relationship between cloze, corpus, and subjective probabilities in language processing

One conclusion suggested by the previous chapter's results is that the brain doesn't just have a general sense of some words being more probable than others; its model of the probabilistic structure of language is fine-grained and quantitative. But how are these fine-grained predictions made? In our analyses, we used a computer model trained on a large text corpus using state-of-the-art algorithms. The resulting trigram model is able to exploit a full *two words of context* when making predictions — at least, most of the time. Sometimes it uses less. The only way it can use more is that it knows that words which have been mentioned earlier in the document are generally more likely to occur again, which is pretty weak compared to what we usually think of as "using context". Of course, there exist computer models that can use more context and in less rigid ways (Elman, 1990), but the fundamental limitation here is not that there's anything wrong with our model — the reason we used this model is that there simply wasn't enough information in our corpus to do much better. If our language has 50,000 words (in fact it has more), then our trigram model has $50{,}000^3 = 1.25 \times 10^{14}$

parameters to estimate, and our corpus only has 10^8 words to estimate them from. Fortunately most of these parameters turn out not to matter (we rarely need to know the the probability of the word *averting* occuring after the phrase *billows redheads*), but even so, much of the art of building statistical language models is in choosing good ways to invent the probabilities that we can't actually measure.

In contrast with our trigram model, there's a rich tradition of research showing that the brain is able to efficiently exploit a wide variety of contextual cues, including such factors as lexical statistics, syntactic structure, real-world plausibility, the immediate physical and social environment, etc. (e.g., Bicknell, Elman, Hare, McRae, & Kutas, 2010; Hanna & Tanenhaus, 2004; Kamide, Altmann, & Haywood, 2003; Tanenhaus, Spivey-Knowlton, Eberhard, & Sedivy, 1995). Yet the typical experimental participant has on the same order of magnitude of linguistic exposure as our computer models do (e.g., this has been estimated at no more than 3.5×10^8 words for the average college student, Levy, Fedorenko, Breen, & Gibson, 2010). So there's an apparent contradiction here: the brain seems to be making these fine-grained, quantitative predictions, while exploiting large amounts of context, but it seems like it doesn't have enough information for this to be possible (Chomsky, 1980; Yarlett, 2008). Certainly part of the answer is that the brain is also exploiting various forms of non-linguistic knowledge, but it's not clear that these are able to provide the orders of magnitude more data that would be needed in order to make quantitatively accurate predictions in many ordinary situations. More likely, the brain is also finding good ways to invent probabilities in situations where they can't be measured. And we would like to know how it does this.

Figure 3.1 summarizes the relationship between the various concepts in play here. Out of all the language usage in the world (**a**), an individual experiences some subset (**b**). These experiences serve as the input to a learning process (**b**→**c**), which produces whatever internal representations of their language that their brain carries with it (**c**). The primary motivation for the research referenced above was to test how much of this knowledge is then available for efficient, online use (**d**).

Here, we instead focus on the knowledge itself (**c**), and the biases that make it possible to learn. While there's a great deal of work on the language learning process (**b**→**c**), the majority has focused on how children acquire their language's basic patterns

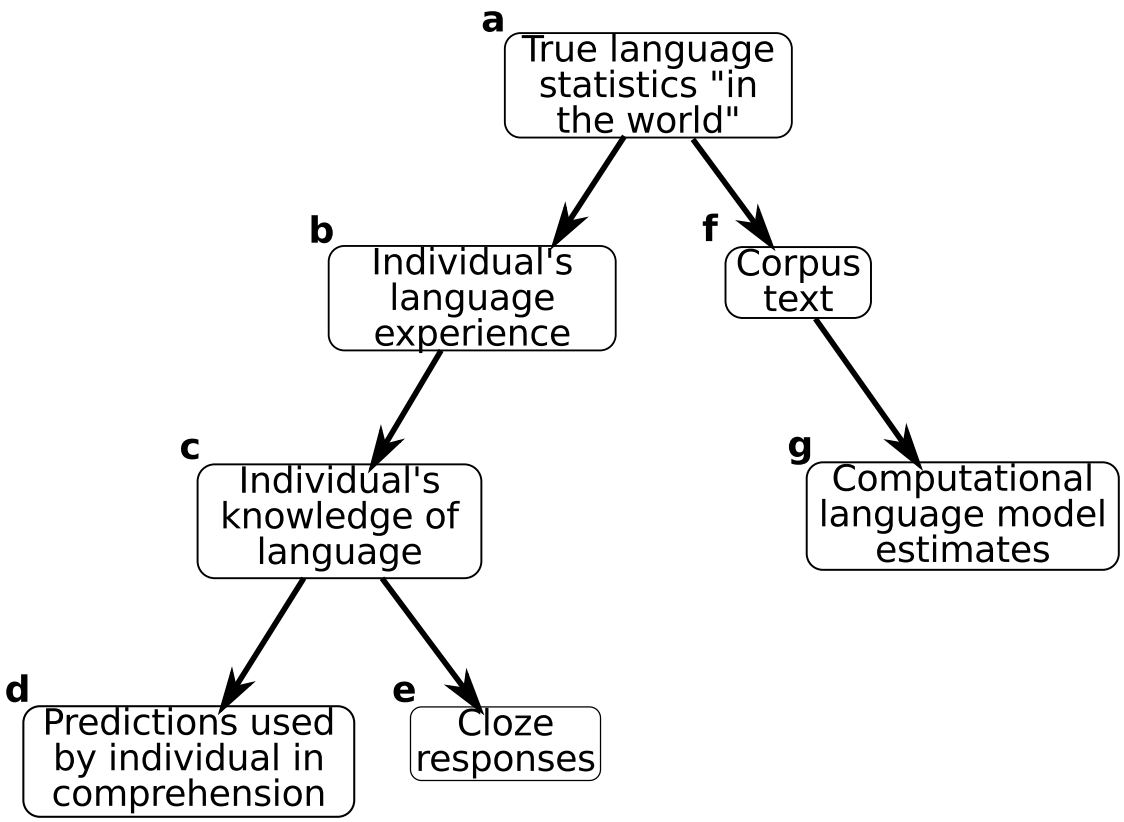

Figure 3.1: An informal illustration of the situation faced by those who wish to study linguistic prediction. Arrows indicate processes. Language is actually used in some particular ways in the real world (**a**); some subset of these uses are sampled by corpora (**f**), and may be used to train computational language models (**g**). A different subset (**b**) is encountered by each human language user, who uses these experiences to create some internal model of the statistics of their language (**c**). They then draw on this internal model to make predictions during online linguistic comprehension (**d**) and also, presumably, when responding in the cloze task (**e**). But the actual relationship between the items on the left side of the diagram remains obscure — do cloze completions match online predictions? Do online predictions match real-world statistics?

— how words are segmented, how tense is expressed, etc. In contrast, we focus on knowledge of fine-grained statistical patterns in language: it's one thing to know your language's tenses, but quite another to know exactly how often each of those tenses is used with each possible verb in each possible sentence frame.

To do this, we need some way to measure speaker's subjective probabilities, and we need some way to measure the 'real world' probabilities, so we can see how they compare; any systematic differences would then give us evidence for how the brain gen-

eralizes from its experience to new contexts. For the subjective probabilities, the most ecologically valid approach would be to somehow extract them from naturalistic online measures such as reading times. Unfortunately, these measures are extremely noisy. This isn't a big problem when trying to pick out overall patterns as in the last chapter, but it makes it infeasible to use them for measuring participant's subjective probabilities in any kind of more fine-grained way. Instead, we use the cloze task (Taylor, 1953) to elicit subjective probabilities (**e**).

The cloze task consists of presenting a large group of participants with sentence stems like *In the winter and* _____, and asking each to fill in the blank with some plausible continuation — "whatever comes to mind". Some might write *spring*, others *summer*, and so on. We then count up what proportion of participants responded with each word; this proportion is called the cloze probability of that word in that context. This task is ubiquitous in studies which wish to measure the predictability of various stimuli items, and is generally taken to measure subjective probabilities, though we actually know very little about how the offline and somewhat artificial nature of the task might affect responses. In previous work, Ong (2007) has attempted to predict cloze values based on computational models, but this was intended as a kind of labor-saving measure to avoid the expense of running cloze norms; we are interested in the cloze task in its own right.

As for the real world probabilities, we again cannot measure these directly; however, we can do better than the language models (**g**) that we used in the previous chapter. The Web 1T five-gram corpus (Brants & Franz, 2006) has recently become available; it counts the occurence of many five-word phrases in one trillion words of English web text. Because it is so large, it allows us to directly measure the probability of continuations for many four-word sentence stems (like *In the winter and*), by simply counting how often each continuation occurs and dividing by the number of times that the stem occured.

In this study, therefore, we selected four-word sentence-initial phrases from the Web 1T corpus, and ran each of them in the cloze task. If the following all hold:

1. The corpus accurately represents participant's language experience (i.e., the **a**→**b** and **a**→**f** sampling processes are similar)

2. Participants have accurate and complete internal models of their language input (i.e., **b**→**c** is veridical, and uninfluenced by learning biases)

3. When presented with isolated, out-of-context sentences, their subjective probabilities are based on accurately sampling from or marginalizing over the contexts implied by that sentence (e.g., if a sentence stem is one that rarely occurs outside of political news coverage, they recognize this and incorporate it into their predictions).

4. Cloze accurately measures subjective probabilities (i.e., the process **c**→**e** which generates cloze responses is simple probability matching)

then we expect that the cloze probabilities and the corpus probabilities will be identical, modulo noise. On the other hand, any differences that we do find must arise from violations of these assumptions. Violations of the first and last assumptions have methodological consequences for those relying on corpus or cloze data, respectively. Violations of the third assumption have methodological consequences for all of us who use isolated, out-of-context sentences in our experiemnts, and would give clues as to how people handle such stimuli. And most importantly, violations of the second assumption can give as information on how the brain learns and/or represents language statistics.

3.1 Experiment 1

3.1.1 Methods

Materials

We selected 300 four-word sentence initial stems from the Web 1T 5-gram corpus (Brants & Franz, 2006), which was compiled from one trillion words of English web text. (By 'stem' we mean nothing more or less than four words which begin a sentence.) The messy nature of this corpus required a complex selection procedure; we summarize the most important points: Our stems were required to have occurred often enough to allow reasonable probability estimates (median count 1906, minimum count 250), to meet a minimum perplexity threshold according to a separate trigram model

trained on the British National Corpus (*The British National Corpus, version 3 (BNC XML Edition)*, 2007), to induce mostly open-class word continuations ($\geq 90\%$), and to vary substantially in the range of probability for their most-likely and second-most-likely continuations. They were then screened by hand to eliminate obvious spam (any phrase used in a spam web page is produced by one person once and then duplicated all over the web, which causes it to be counted many times, as if it were in fact a commonly produced phrase), high-frequency stereotyped phrases (*e.g., Designated trademarks and brands...*), excessively technical usages that we judged participants were unlikely to have had much experience with (*The study protocol was...*), or web-specific usages (on the web, *If you leave the...* is usually followed by *...field blank...*, because web pages are very concerned about explaining web forms). Finally, whenever two stems were judged 'too similar' to each other (e.g., because they differed from each other only in the gender of pronouns), one of them was eliminated.

Procedure

Participants performed a computerized sentence continuation (cloze) task, in which they were given each stem and asked to type one or more words which naturally continued the sentence. Spelling was corrected by hand, with computer assistance.

Participants

140 students from UC San Diego participated for course credit. All were native English speakers. 114 participated via an online web form; of these, 6 were eliminated for admitting in a post-test questionnaire that they had used Google to find continuations. The remaining 26 participants performed the identical task in a lab environment. Because analyses (not shown) found no difference between the online and in-person groups, we pool their data.

3.1.2 Results and Discussion

As the online and in-person participant groups performed similarly in all analyses reported here, we present only pooled data ($N = 134$).

Table 3.1: Sample continuation distributions from Experiment 1. In each case, the left column is web corpus probability, and the right column is measured cloze probability. '—' denotes continuations that were never observed.

He played a key...			*After a cup of...*		
role	94%	42%	coffee	39.6%	28%
part	2.3%	3.8%	tea	39.1%	61%
			hot	3.0%	—
When she began to...			*The time needed to...*		
speak	5.2%	9.0%	complete	41%	6.7%
cry	2.5%	19%	reply	3.2%	—
work	2.2%	1.5%	finish	0.3%	10%
It usually takes the...			*In the winter and...*		
form	34%	1.5%	spring	66%	40%
shape	23%	—	early	13%	—
following	2.7%	—	summer	4.2%	32%
cake	—	8.3%	fall	2.3%	19%

When we started this project, the Web 1T corpus was the only corpus available that was large enough for our purposes, and so it was used to optimize the design of both this and the next experiment. However, a second large corpus has recently become available, derived from scanned books rather than the raw web (Michel et al., 2011; we use the subset containing American English since 1960, consisting of ∼89 billion words). This corpus seems more representative of real usage (i.e., it has no spam), but is smaller and our experimental design is not optimized to take full advantage of it; therefore, we perform all analyses with both corpora.

Fig. 3.2 shows the overall relationship between cloze probability and corpus probability; our first result is that their correlation is only moderate. So the next question is, if corpus probability does not determine cloze, then what does?

To find out, we used regression analysis to predict cloze probabilities given corpus probability and other predictors. Since we are trying to predict probability values which are necessarily bounded and normalized, standard gaussian least-squares regression is inappropriate. Instead, we fit cloze responses to a modified log-linear model of the form

$$P(\text{response}_{ij}|\text{stem}_i) = \frac{1}{Z_i} \times p_{ij}^{(\beta_0+\beta_1 \text{center}(\text{StemProp1}_i)+\cdots)}$$

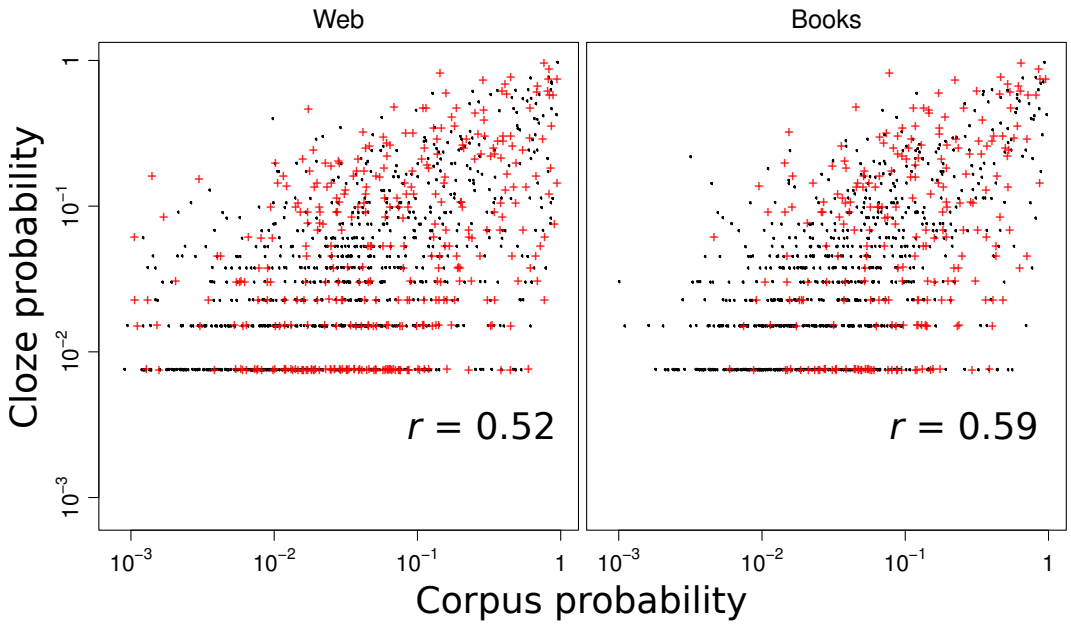

Figure 3.2: Cloze versus corpus probability. Each point represents a single stem/continuation pair that appeared in the corpus and was given by at least one cloze participant. (Regression analyses included responses that were given by zero participants, but they can't be shown on this log scale.) Correlations are computed in log space. Red crosses mark stem/continuation pairs that were selected for use as stimuli Experiment 2.

$$\times \exp(\gamma_1 \text{WordProp1}_j + \cdots)$$

Here, p_{ij} indicates the probability of continuation j given stem i computed from the corpus. Because we were interested in how people used context, however, we did not simply enter the raw corpus probability; instead, we defined p_{ij} as a weighted average of the five-gram, four-gram, trigram, and bigram probabilities from the corpus, with the weights chosen as part of the fitting process:

$$p_{ij} = \sum_{k=1}^{4} \alpha_k P_{\text{corpus}}(\text{word}_{ij} | \text{Last } k \text{ words of stem}_i)$$

$$\sum_{k=1}^{4} \alpha_k = 1.$$

Thus, the model is allowed to use the full context, or, alternatively, to use less than that, depending on which best explains the data. α_k can be interpreted as the proportion of

the time that participants chose to attend to k words of context when producing their response. β_0 is a free parameter that measures the baseline sensitivity of cloze to corpus probability. An β_0 of 0 would indicate no sensitivity, and a value of 1 would indicate that cloze matches corpus probability perfectly (at least, until the word-specific parameters come in to further influence matters). A value between 0 and 1 would indicate that cloze distributions are overall flatter (have higher entropy) than the corresponding corpus distribution, while a value greater than 1 would indicate that cloze distributions are more peaked (have lower entropy), as might happen if participants preferred to provide the most-probable continuation instead of probability matching. StemProp1, ... are properties of the stem which might modulate the overall effect of β_0. WordProp1, ... are word properties that might cause participants to use particular words more or less often than predicted by corpus probability alone. And Z_i is a normalizing constant (not a free parameter). This normalization is the reason we chose a log-linear model over more common forms of regression — it enforces the idea that different continuations for the same stem must compete for responses; if one becomes more popular, it is at the expense of others.

Stem predictors included the corpus constraint (defined as the probability of a stem's most-probable continuation, $\max_i P(\text{continuation}_i|\text{stem})$), and the total number of times that the stem was observed in the corpus (a proxy for participants likely amount of experience with each particular stem). Word predictors included familiarity, concreteness, imageability, and age of acquisition (from Wilson, 1988; Stadthagen-Gonzalez & Davis, 2006; Nelson, McEvoy, & Schreiber, 1998), word frequency and contextual diversity (from Brysbaert & New, 2009), word length, and a measure of interlexical priming from the stem to the target ('lexical prime probability'). This measure was computed by looking up the probability p_i that each word i in the stem would produce the continuation as a response in a free-association task (Nelson et al., 1998), and then combining these probabilities as $1 - \prod_i (1 - p_i)$. This implements the idea that each of the words in the stem has an independent chance of priming the continuation word, and so the overall probability of its being primed is the probability that at least one stem word primes it. Finally, we entered a factor indicating whether each participant had produced the given continuation word in a previous trial within the experiment, i.e., their response was a

Table 3.2: Estimated coefficients from a (modified) log-linear model regressing cloze responses against corpus probability and other measures. Shaded cells are significant. Positive/green cells indicate a response preference, while negative/red cells indicate a response dispreference; e.g., cloze participants use familiar words more than would be expected given corpus probability but may avoid long words. Not much should be read into the absolute magnitude of coefficients, since different predictors are on different scales.

	Web	Books
Corpus probability from		
all context	3%	11%
3 words context	97%	74%
2 words context	0%	15%
1 word context	0%	0%
is raised to some power		
Baseline	0.88	0.88
Corpus constraint ($\log_{10}$)	−0.41	−0.30
Stem frequency ($\log_{10}$)	−0.26	0.21
then modulated by factors:		
Familiarity	0.99	0.76
Concreteness	0.03	−0.01
Imageability	0.03	0.18
Age of acquisition	0.19	0.22
Frequency (SUBTLEX)	−0.57	−0.36
Contextual diversity	1.34	0.69
Length	−0.02	−0.11
Lexical prime probability	2.17	4.38
Used response before	0.57	0.56

repeat.

We analyzed the subset of the data for which all of these norming values were available, for which the continuation was recorded in the corpus, and, in the book corpus analysis, for which the stem occured at least 100 times in the corpus. (This allowed the analysis of 5015 responses for the web data, and 4636 for the book data.) The model was fit by maximum-likelihood, with all predictors entered simultaneously, and significance computed with the likelihood ratio test and corrected for multiple comparisons by sequential Bonferroni. The results of this analysis are shown in Table 3.2.

We start with the finding that the full corpus probability is *not* the best predictor

of cloze probability; the probability based on only three words of context dominated in both the web and books analyses. This is true despite the fact that for these stimuli, we are able to accurately calculate the probability of continuations given all four words, so this is not a simple matter of the full context 5-gram model being noisier than the reduced context 4-gram model. It seems that participants overwhelmingly ignore the first word of the prompt stem when producing their responses.

Reassuringly, though, the exponent β_0 on this reduced-context corpus probability is significantly greater than 0 (web: $\chi^2(1) = 992, p \ll 0.001$, books: $\chi^2(1) = 784, p \ll 0.001$), indicating that cloze is sensitive to corpus probability, as expected. However, it is also significantly smaller than 1 (web: $\chi^2(1) = 62, p \ll 0.001$, books: $\chi^2(1) = 44, p \ll 0.001$), indicating that to best explain the cloze distribution, we need to systematically flatten out the corpus distributions. This may indicate that cloze distributions are higher entropy than corpus distributions, and participants are more confused about upcoming linguistic material than would be expected of an optimal rational agent (compare Griffiths & Tenenbaum, 2006). However, it may be also be an artifact of the regression technique we used — we would also expect this effect if participants have a similar overall distribution as the corpus, but a different idea about which words are high probability and which are low. Unfortunately, technical limitations of the corpus data sets make it difficult to estimate the entropy of these distributions directly for comparison. In any case, this flattening is more pronounced for contexts that are particularly constraining (web: $\chi^2(1) = 36, p \ll 0.001$, books: $\chi^2(1) = 29, p \ll 0.001$). It is also affected by commonality of the stem. For high frequency stems in the web corpus, we must flatten out the corpus distribution even more than otherwise in order to fit the cloze distribution ($\chi^2(1) = 54, p \ll 0.001$); in the book corpus, we find the opposite effect, with high frequency stems requiring less flattening to explain cloze values ($\chi^2(1) = 8.6, p < 0.005$). This may indicate that on the web, high frequency phrases are ones that are contaminated by spam and other distributional oddities, making their corpus distributions particularly noisy, but in print, high frequency phrases are ones that participants genuinely have more experience with, and that they are able to use this experience to make better predictions.

For word properties, there is evidence that participants prefer to respond with

words which are familiar, concrete, have high contextual diversity, and which are primed by words in the stem (e.g., this may explain the *winter and fall* responses; in ordinary usage people would usually choose to say *fall and winter* instead, which makes *fall* an unlikely continuation according to the corpus; but in any case *fall* is somewhat primed by *winter*). They may avoid words which are long — perhaps because they require more effort to type — and also avoid words which are imageable or are acquired early — perhaps because in the formal context of an experiment they attempt to use more formal language. And, finally, they have a preference towards repeating themselves: if a word has been used once, then participants are about $\exp(0.56) = 1.75$ times more likely to produce it again on future trials.

3.2 Experiment 2

Having established that cloze and corpus probability vary in substantial and systematic ways, our next question is whether to attribute these effects to biases in the corpus sampling (Fig. 3.1, **a**→**f**), to biases in language acquisition and processing (**b**→**c**), or to biases in cloze task performance (**c**→**e**). If these effects were caused by the cloze task alone, then we would expect corpus probabilities to be more closely correlated with online subjective probabilities than cloze probabilities are, and therefore corpus probabilities should outperform cloze probabilities in explaining performance in an online comprehension task. So in this experiment, we pit cloze probabilities against corpus probabilities in explaining self-paced reading times.

3.2.1 Methods

Materials

Experiment 1 produced 2350 distinct stem-plus-continuation pairs for which we had both cloze and corpus predictability measurements (the points in Figure 3.2). From these we selected 179 four-word stems, then for each stem selected 2 target continuations, producing a total of 358 five-word sentence beginnings (the red points in Figure 3.2). We then completed each sentence and divided them into two 179-sentence

lists, so that no participant saw any stem or any target more than once. These stems and target continuations were selected to maximize our ability to distinguish cloze and web corpus probability according to a power analysis. No fillers were used. Several example stimuli are shown below:

1. In the winter and (fall/summer) the little town would hold fairs to raise the community spirit.

2. When she began to (cry/sing) he overheard and wanted to know what (had happened / song it was).

3. But this was no (problem / reason) for that amazing team (that they hired / to be fired).

Procedure

Participants read sentences in random order in a self-paced moving-window paradigm (Just, Carpenter, & Woolley, 1982) with a comprehension question presented after each sentence.

Participants

58 students students from UC San Diego participated for course credit. All were monolingual English speakers.

3.2.2 Results and Discussion

Comprehension questions

All subjects performed significantly above chance on comprehension question accuracy (minimum 77%, median 92%).

Reading times

We analyzed the total reading time for a region consisting of the target word plus the following word (to capture spillover; see Fig. 3.3). After removing sentences with

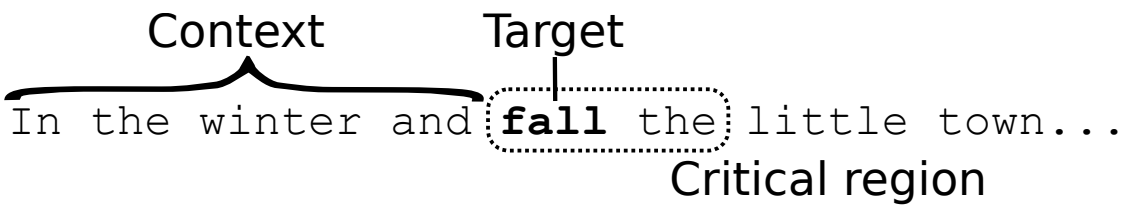

Figure 3.3: Sample stimulus for Experiment 2.

incorrect comprehension question answers or outlier reading times in the critical region (reading times <80 ms, >1500 ms, or >4 sd above participant-specific means, 1.7% of data removed), reading times were entered into a multi-level mixed-effects regression model using lme4 (Bates & Maechler, 2010). Our first question was whether cloze or corpus probabilities better explained reading times, so both were log-transformed (see previous chapter) and entered into the regression. In addition, for controls, we entered the log-frequency (from Brysbaert & New, 2009), word length, log-frequency/word-length interaction, and log-corpus probability for three different words: the word preceding the target, the target itself, and the word following the target. Because our stimuli were not optimized to produce corpus estimates of the probability of the word following the target, this probability was estimated using only a two-word context (i.e., an unsmoothed trigram model). As random effects, we allowed the intercept to vary by stem, and the intercept, slope of the cloze effect, and slope of the target word corpus probability effect to vary by subject (this structure selected by model comparison). Significance was assessed by assuming calculated t values were distributed as standard normal under the null hypothesis (Baayen, Davidson, & Bates, 2008).

Results are shown in Tables 3.3 and 3.4. We found that cloze was significant after controlling for corpus probability and the other factors described (web: $t = -2.17, p < 0.03$, books: $t = -2.47, p < 0.02$), but that after controlling for cloze and these other factors, corpus probability was not significant, or its effect went the wrong way (web: $t = 1.37$, *n.s.*, books: $t = 2.05, p < 0.04$). This would suggest that at least some of the biases we observe in cloze probability are also present in readers' subjective probabilities; the most substantial differences between cloze and corpus probabilities do *not* arise from the artificiality of the cloze task.

Table 3.3: Mixed-effects regression analysis of critical region reading times from experiment 2, using web corpus.

	β coef.	SE(β)	t	p
Intercept	472.67	39.64	11.92	$\ll$0.01
Target word (*fall*):				
Cloze probability (log10)	−9.05	4.14	−2.19	0.03
Web probability (log10)	4.92	3.58	1.37	0.17
Frequency (log10)	−11.14	6.97	−1.60	0.11
Length	6.90	4.92	1.40	0.16
Frequency × Length	0.40	1.09	0.37	0.71
Succeeding word (*the*):				
Web probability (log10)	−1.92	2.21	−0.87	0.39
Frequency (log10)	18.14	6.19	2.93	<0.01
Length	−10.37	3.66	−2.84	<0.01
Frequency × Length	−3.47	1.05	−3.32	<0.01
Previous word (*and*):				
Web probability (log10)	0.59	3.00	0.20	0.85
Frequency (log10)	6.00	4.66	1.29	0.20
Length	−6.02	3.75	−1.60	0.11
Frequency × Length	−2.37	0.83	−2.85	<0.01

Table 3.4: Mixed-effects regression analysis of critical region reading times from experiment 2, using books corpus.

	β coef.	SE(β)	z	p
Intercept	489.03	50.41	9.70	$\ll$0.01
Target word (*fall*):				
Cloze probability (log10)	−11.59	4.69	−2.47	0.01
Books probability (log10)	12.07	5.88	2.05	0.04
Frequency (log10)	−10.72	9.77	−1.10	0.27
Length	7.38	6.42	1.15	0.25
Frequency × Length	0.49	1.49	0.33	0.74
Succeeding word (*the*):				
Books probability (log10)	−2.48	2.95	−0.84	0.40
Frequency (log10)	22.52	7.40	3.04	<0.01
Length	−15.22	5.24	−2.91	<0.01
Frequency × Length	−5.04	1.47	−3.42	<0.01
Previous word (*and*):				
Books probability (log10)	−0.62	4.15	−0.15	0.88
Frequency (log10)	6.67	6.16	1.08	0.28
Length	−6.96	5.00	−1.39	0.16
Frequency × Length	−2.56	1.07	−2.39	0.02

3.2.3 Experiment 3

This suggests that we should focus on distinguishing between sampling differences and learning biases as the source of our effects. In particular, we were concerned about the over/under-representation of responses based on psycholinguistic attributes such as familiarity. It's very plausible that this could arise from a true psychological bias, but we also know that ordinary language experience, which has a substantial verbal/conversational component, is quite different from the kind of written text which makes its way into the corpora we are using. Maybe participants' language experience really does contain a higher proportion of 'familiar' words than are measured by our corpora?

Unfortunately we can't easily access participants' history of language experience to find out; but what we can do is find another corpus that is hopefully more representative. We use SUBTLEX (Brysbaert & New, 2009), which is a corpus of movie subtitles, which is a convenient source of dialogic speech. Critically, the word frequencies measured from SUBTLEX out-perform the word frequencies measured from most other corpora at explaining response times on word recognition tasks (Brysbaert & New, 2009). Thus, while SUBTLEX is far too small to use to compute the actual probabilities in context for our main study, we can expect that the overall mix of words which it contains are more representative of those experienced by our participants in their lives.

3.2.4 Methods

We used a log-linear model as described above to regress *Subtlex* single-word frequency counts against corresponding single-word frequency counts from the Web 1T and Google books corpora. We left out those predictors which did not make sense in this context (e.g., all of the stem-related predictors), leaving us with factors of familiarity, concreteness, imageability, age of acquisition, and length. Significance tests were performed on each factor via likelihood ratio test and corrected for multiple comparisons by sequential Bonferroni. (However, since the data sets involved are so large, this was largely superfluous; every effect was massively significant.)

Table 3.5: Estimated coefficients from a log-linear model regression SUBTLEX frequencies on Web 1T/Google books frequencies, plus other factors. Shaded cells are significant. Positive/green cells indicate that words with this property are over-represented in SUBTLEX as compared to the other corpora, while negative/red cells indicate that words with this property are underrepresented in SUBTLEX as compared to the other corpora.

	Web	Books
Web/Books frequency	0.57	0.65
Familiarity	0.51	0.43
Concreteness	−0.29	−0.26
Imageability	0.08	0.10
Age of acquisition	−0.11	−0.14
Length	−0.56	−0.55

3.2.5 Results and discussion

Results are given in Table 3.5. The words in SUBTLEX do differ from those in the Web 1T and Google books corpora; the average SUBTLEX word is more familiar, less concrete, more imageable, acquired earlier, and shorter. These differences are nearly identical for both the web and book analyses, suggesting that they may arise primarily from the difference between dialogic and written text.

Like our cloze data, SUBTLEX contains more usage of words with high familiarity than would be predicted from the Web 1T or Google books corpora (though the effect is roughly twice as strong for the cloze data). The estimated effects of imageability are also similar in magnitude between the two analyses, though they only reach significance for the SUBTLEX data. The other effects, however, do not line up very well; for instance, there is a substantial difference between our corpora and SUBTLEX in terms of concreteness, but there is no such difference between our corpora and the cloze responses; and the age-of-acquisition effects go in opposite directions.

However, we don't really expect that SUBTLEX should be a perfect match to cloze responses; after all, our participants probably both watch movies *and* surf the web, among other linguistically relevant activities. What we take from this experiment is that different types of language can differ dramatically in terms of the types of words which they contain. We conclude therefore that we cannot rule out the possibility that the effects of word type in Table 3.2 are artifacts of sampling differences between our corpora and our participants' experience, and thus should not interpret them as reflecting

learning biases.

We also note that if concreteness, familiarity, and so forth differ between language genres, then this may be another possible cause of effects which have been attributed to them, for instance due to mismatches between the genre implied by one's experimental stimuli or assumed by one's participants and the genre of the critical items embedded in those stimuli.

3.2.6 Experiment 4

We next turn to our participants' apparent disregard for the first word in each cloze stem. This also seems particularly plausible as a learning bias — indeed, it exactly mirrors the n-gram learning bias that we regularly impose on computational models to improve their performance on sparse data! Furthermore, we cannot think of a plausible mechanism for how this could arise as an artifact of our corpus sampling; why would four word phrases have similar statistics between our corpus and our participants' experience, while five word ones do not? However, it also seems plausible that it could be an artifact of the cloze task itself, where participants may only partially skim each prompt before responding in order to get through the experiment more quickly.

We reasoned that if this were the source of this effect, then we would predict that participants engaging in a more naturalistic reading task (Experiment 2) *would* include the full context in their online predictions, especially since the self-paced reading paradigm forces them to look at the first word of the text before they can get to the rest, and because in this task they were required to answer comprehension questions, which requires a certain attention to the sentences. (In the cloze task, by contrast, they could write literally anything without receiving any error feedback.)

We know that overall in Experiment 2, cloze probability better predicted reading times than corpus probability. But it might be that this is because cloze probabilities are just overall more accurate, because they better reflect our participants' language experience. What we want to know is whether the discrepancy between corpus and cloze probability that came specifically from ignoring the first word of each prompt is also reflected in the reading time variations.

3.2.7 Methods

We defined *full context cloze* as:

$$\text{Cloze}_{\text{Full context}} = \text{Cloze}_{\text{Measured}} \frac{P_{\text{corpus}}(\text{word} \mid \text{All } 4 \text{ words of stem})}{\left(\sum_{k=1}^{4} \alpha_k P_{\text{corpus}}(\text{word} \mid \text{Last } k \text{ words of stem})\right)^{\beta_1}} \frac{1}{Z}.$$

where the α_k and β_1 values were drawn from the regression of cloze on corpus probabilities, and Z is a normalizing factor. Effectively, we're using our log-linear model from Experiment 1 to estimate exactly how the use of a reduced context changes the cloze, and then correcting for that. If including the first word of the stem in our corpus probability calculation makes a particular continuation more likely, then we increase its cloze value; if including the first word in the corpus calculation makes a particular continuation less likely, then we likewise reduce its cloze. Otherwise, we leave the cloze alone. For example, for $P(\text{action}|\text{The consequences of their})$, we have $\text{Cloze}_{\text{Measured}} = 0.015$. This matches fairly closely to the web corpus probability given three words of context: 0.013. The corpus probability given the full context, however, is larger: 0.078. Using this procedure, therefore, we find a higher $\text{Cloze}_{\text{Full context}} = 0.108$. Critically, because the actual cloze was larger than the cloze predicted by the partial-context corpus probability model, the full-context cloze is also larger than the full-context corpus probability.

We then re-ran the mixed-effects regression analysis described above for Experiment 2, but now entering $\text{Cloze}_{\text{Full context}}$ and $\text{Cloze}_{\text{Measured}}$ in the place of cloze and corpus probability. The same controls and random-effects structure were used.

3.2.8 Results and Discussion

We were unable to reliably distinguish between $\text{Cloze}_{\text{Full context}}$ and $\text{Cloze}_{\text{Measured}}$. However, $\text{Cloze}_{\text{Measured}}$ trended in the correct direction, (web: $t = -1.08$, *n.s.*, books: $t = -1.13$, *n.s.*), while the effect of $\text{Cloze}_{\text{Full context}}$ was smaller and in the wrong direction (web: $t = 0.11$, *n.s.*, books: $t = 0.72$, *n.s.*). So, we can say at least that there is no evidence that our self-paced reading participants were more sensitive to context than our cloze participants; the trend is the reverse, if anything.

3.3 Conclusion and Future Directions

It's clear that cloze and corpus probabilities differ (Figure 3.2), and that these differences are highly systematic (Table 3.2). We've argued that there are at least three places where these differences can arise. First, our corpora may not be representative of our participants' linguistic experience (Figure 3.1, **a**→**b** versus **a**→**f**). A possible example of this would be the differences in word properties like familiarity between cloze continuations and corpus continuations. Secondly, cloze probabilities themselves may differ from participants' internal subjective probabilities due to biases in how they produce cloze responses (**c**→**e**). A likely example of this would be the preference towards repeating responses which have previously been given. Finally, participants' predictions may fail to accurately capture the statistics of the world (or even their experience) because they need to generalize from limited and sparse experience (**b**→**c**) — they need to learn rather than simply memorize.

Futhermore, we have some plausible candidates for effects caused by such learning. The most striking is that our cloze responses appear to based on only partial context, mimicking the n-gram smoothing commonly used in computational language modeling. Along the same lines, we found that free association norms can predict cloze responses above and beyond the actual corpus-measured co-occurrence of the relevant words; using a pairwise word association measure to predict upcoming words seems like a reasonable strategy for the brain to adopt in the presence of sparse data.

While promising, however, these results are clearly not the end of the story, and they suggest a number of directions for future work.

The first, and most obvious, is that of better nailing down the actual sources of the discrepancies between cloze and corpus probabilities that we found. While the differences between SUBTLEX and the Google corpora in terms of word properties like familiarity make it plausible that these particular discrepancies arise from the non-representativeness of our corpora, they do not assure it — particularly since the actual magnitude of the differences reported in Tables 3.2 and 3.5 are quite different. If we could somehow obtain a more representative sample of our participants' language exposure, then we could repeat this analysis; if sampling differences are indeed the root cause of these descrepancies, then we would expect our more representative corpus to

contain the same pattern of effects as reported in Table 3.2. Similarly, while it seems *a priori* somewhat implausible that our participants' apparent use of partial context could arise as a sampling artifact — this would require the four-word phrases in our corpus to have similar statistics to our participants' experience, while the five-word phrases did not — it would be good to test this. One method would be to extract sentences from SUBTLEX that are similar to those we used as stimuli, and confirm that using the full four words of context in the Google corpora outperforms using only three words of context at predicting the next word of the SUBTLEX sentences. Another potential strategy is illustrated by our (unsuccessful) Experiment 4 — with better chosen stimuli, and possibly higher quality materials, it should be possible to use online reading time data to further distinguish between sampling artifacts and learning biases.

Our results also have methodological implications, as many people use cloze and corpus probability estimates (usually with additional smoothing) as practical estimates of subjective predictability. Our results support the use of cloze for this purpose. It may be possible to do somewhat better, for instance by somehow correcting for the repetition bias we discovered, and it is entirely possible that the task introduces other unknown biases. (For instance, it might be interesting to see whether there is a bias towards particular word classes — e.g., when confronted with a sentence frame that selects for an NP, participants might prefer to use a simple noun, rather than adjective plus noun.) But overall, Experiment 2 confirms that cloze is a reasonable measure for this purpose. Of course, there are circumstances in which cloze cannot practically be used (e.g., our experiments in chapter 2), and in general cloze cannot measure the predictability of low-predictability events. In these situations our results may be useful in improving our ability to estimate subjective predictability from corpus measures.

A third area for further research would be to study the effect of individual differences in this task. For instance, working memory span might influence participants' ability to identify the kind of context in which an isolated phrase is likely to have been drawn from, and we might expect participants who are avid readers and conversationalists to have more accurate, less-smoothed probability estimates than other participants who have less linguistic experience.

And finally, there is much more to be done on our central question: how do

speakers produce predictions in novel contexts? The partial context and free association norm effects that we found may be two examples of a more general strategy of similarity-based generalization (Yarlett, 2008). For example, here are two cloze responses from Experiment 1:

1. *It was no great* __fun__

2. *It was no great* __deal__

Neither strikes us as particularly felicitous English, but both can be easily repaired. In (1), deleting the negative to produce *It was great fun* gives a much more common phrase, and in (2), replacing *great* with its synonym *big* produces the common idiom *It was no big deal*. None of which proves anything, since this is just our intuitive sense of how two arbitrarily picked responses might have arisen — but it does hint at a hypothesis for how participants produce cloze responses (and, potentially, linguistic predictions in general), in which near-neighors of the context that is actually observed can also influence the response. We modeled one sort of near-neighbor in our analyses here — those in which the initial words may differ, but the final words are the same — but these examples suggest that we may need a broader definition. The techniques we use here could also be used to test any particular theory one might have about such generalization. One starting point might be to test whether participants are sensitive to different parts of the context depending on where the relevant information falls — for instance, are they ignoring the first word because they use an n-gram model, or because it happens that in our experiment, the first word is the least informative and therefore the best candidate for being marginalized over? Unfortunately limitations of our current corpora make it difficult to compute the probability of a non-contiguous string, which is what would be needed to test which words in the context are most informative about the continuation — but these problems are surmountable, and doing so could give new insight into how speakers turn past experience into predictions.

3.4 Acknowledgments

We thank Erin Bennett, Tiffany Chiou, Megha Ram, Maria Sokolov, and Daphne Tan for assistance in collecting data. This research was partially supported by NIH Training Grant T32-DC000041 to the Center for Research in Language at UC San Diego to NJS, NSF grant 0953870 to RL, and funding from the Army Research Laboratory's Cognition & Neuroergonomics Collaborative Technology Alliance.

Chapter 3, in part, is a revised version of the material as it appears in the proceedings of the 33rd annual conference of the Cognitive Science Society. Smith, Nathaniel J.; Levy, Roger, Cognitive Science Society, 2011. The dissertation author was the primary investigator and author of this paper.

References

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*(4), 390–412.

Bates, D., & Maechler, M. (2010). lme4: Linear mixed-effects models using s4 classes [Computer software manual]. Retrieved from http://CRAN.R-project.org/package=lme4 (R package version 0.999375-37)

Bicknell, K., Elman, J. L., Hare, M., McRae, K., & Kutas, M. (2010). Effects of event knowledge in processing verbal arguments. *Journal of memory and language*, *63*(4), 489–505.

Brants, T., & Franz, A. (2006). *Web 1T 5-gram version 1.1, LDC2006T13*. Philadelphia, PA: Linguistic Data Consortium.

The British National Corpus, version 3 (BNC XML edition). (2007). Retrieved from http://www.natcorp.ox.ac.uk/ (Distributed by Oxford University Computing Services on behalf of the BNC Consortium)

Brysbaert, M., & New, B. (2009). Moving beyond Kučera and Francis: A critical evaluation of current word frequency norms and the introduction of a new and improved word frequency measure for American English. *Behavior Research Methods*, *41*, 977–990.

Chomsky, N. (1980). *Rules and representations*. Oxford: Basil Blackwell.

Elman, J. L. (1990). Finding structure in time. *Cognitive Science*, *14*, 179–211.

Griffiths, T. L., & Tenenbaum, J. B. (2006). Optimal predictions in everyday cognition. *Psychological Science*, *17*(9), 767–773.

Hanna, J. E., & Tanenhaus, M. K. (2004). Pragmatic effects on reference resolution in a collaborative task: evidence from eye movements. *Cognitive Science*, *28*, 105–115.

Just, M. A., Carpenter, P. A., & Woolley, J. D. (1982). Paradigms and processes in reading comprehension. *Journal of Experimental Psychology: General*, *111*(2), 228–238.

Kamide, Y., Altmann, G., & Haywood, S. L. (2003). The time-course of prediction in incremental sentence processing: Evidence from anticipatory eye movements. *Journal of Memory and Language*, *49*(1), 133–156.

Levy, R., Fedorenko, E., Breen, M., & Gibson, T. (2010). *The processing of extraposed structures in english.* Manuscript submitted for publication.

Michel, J., Shen, Y. K., Aiden, A. P., Veres, A., Gray, M. K., Team, T. G. B., et al. (2011). Quantitative analysis of culture using millions of digitized books. *Science*, *331*(6014), 176–182.

Nelson, D. L., McEvoy, C. L., & Schreiber, T. A. (1998). *The University of South Florida word association, rhyme, and word fragment norms.* Retrieved from http://www.usf.edu/FreeAssociation/

Ong, J. K. Y. (2007). *The predictability problem*. Doctoral dissertation, Potsdam University. Retrieved from http://opus.kobv.de/ubp/volltexte/2007/1502/ (Wir versuchen herauszufinden, ob das subjektive Ma der Cloze-Vorhersagbarkeit mit der Kombination objektiver Mae (semantische und n-gram-Mae) geschtzt werden kann, die auf den statistischen Eigenschaften von Textkorpora beruhen. Die semantischen Mae werden entweder durch Abfragen von Internet-Suchmaschinen oder durch die Anwendung der Latent Semantic Analysis gebildet, whrend die n-gram-Wortmae allein auf den Ergebnissen von Internet-Suchmaschinen basieren. Weiterhin untersuchen wir die Rolle der Cloze-Vorhersagbarkeit in SWIFT, einem

Modell der Blickkontrolle, und wgen ab, ob andere Parameter den der Vorhersagbarkeit ersetzen knnen. Unsere Ergebnisse legen nahe, dass ein computationales Modell, welches Vorhersagbarkeitswerte berechnet, nicht nur Mae beachten muss, die die Relatiertheit eines Wortes zum Kontext darstellen; das Vorhandensein eines Maes bezglich der Nicht-Relatiertheit ist von ebenso groer Bedeutung. Obwohl hier jedoch nur Relatiertheits-Mae zur Verfgung stehen, sollte SWIFT ebensogute Ergebnisse liefern, wenn wir Cloze-Vorhersagbarkeit mit unseren Maen ersetzen.)

Stadthagen-Gonzalez, H., & Davis, C. J. (2006). The Bristol norms for age of acquisition, imageability and familiarity. *Behavior Research Methods*, *68*, 598–605.

Tanenhaus, M. K., Spivey-Knowlton, M. J., Eberhard, K. M., & Sedivy, J. C. (1995). Integration of visual and linguistic information in spoken language comprehension. *Science*, *268*, 1632–1634.

Taylor, W. L. (1953). "Cloze procedure": A new tool for measuring readability. *Journalism Quarterly*, *30*, 415–433.

Wilson, M. D. (1988). The MRC psycholinguistic database: Machine Readable Dictionary, version 2. *Behavioural Research Methods, Instruments and Computers*, *20*(1), 6–11.

Yarlett, D. (2008). *Language learning through similarity-based generalization*. Unpublished doctoral dissertation, Stanford University.

Chapter 4

Regression-based ERP analysis: I. Pointwise regression techniques

EEG activity recorded at the scalp measures ongoing activity from many different parts of the brain, as well as various non-brain artifacts. Only a small portion of this summed activity reflects processing related to any particular task, and so a major challenge in analyzing and interpreting the data from EEG experiments is to distinguish the task-related 'signal' from the unrelated 'noise'. One strategy is to estimate the event related potential (ERP), i.e., that portion of the EEG signal which is consistently present across trials and time-locked to some event of interest. The usual method for estimating the ERP is to extract time-locked epochs from the continuous EEG signal, align them, and compute their point-by-point average. But the averaging approach places significant limitations on experimental design: stimuli must be chosen to fall into a small set of discrete categories, and these categories must differ *only* along the axes of interest, while all other stimulus properties that might affect neural processing must be held constant. In some experiments, such as those investigating language processing, it is generally impossible to design stimuli that fully satisfy the latter requirement — every word has a vast array of properties like length, frequency, orthographic neighborhood size, and so on, which cannot be perfectly matched across conditions. And even when a rigid categorical design is possible, there remains value in being able to study the brain's response to more naturalistic stimuli that vary in continuous ways along multiple axes. Here, we describe an alternative method for estimating ERPs by using multiple regression in place

of averaging. This approach allows us to relax the above restrictions while retaining the essential attributes of the standard approach.

We are, of course, not the first to consider the problem of analyzing EEG with multiple and non-categorical factors. One piece of prior art is the 'ERP image' technique, which allows one to visualize the effect of a single continuous factor such as reaction time on the ERP (Jung et al., 2001; Lorig & Urbach, 1995). In addition, a number of groups have used forms of multiple regression to analyze EEG data, in a variety of ways. One approach is to divide a continuous factor into a number of bins, calculate the conventional ERP for each bin, and then derive some measure such as peak latency or average activity which is entered into a linear regression analysis (e.g., King & Kutas, 1998; DeLong, Urbach, & Kutas, 2005). (A variation on this approach is to use bins so small that they contain only a single item; e.g., Laszlo & Federmeier, 2011.) Others have calculated the average activation over some window within single trials, and then entered these into a mixed-effects regression analysis (Amsel, 2011; Dambacher, Kliegl, Hofmann, & Jacobs, 2006). Alternatively, some have proposed calculating a separate regression model at each possible latency, similar to the mass univariate techniques used in PET/fMRI analysis (Hauk, Davis, Ford, Pulvermuller, & Marslen-Wilson, 2006; Hauk, Pulvermller, Ford, Marslen-Wilson, & Davis, 2009; Rousselet, Pernet, Bennett, & Sekuler, 2008; Pernet, Chauveau, Gaspar, & Rousselet, 2011). And yet others have proposed using a non-parametric regression approach based on generalized additive models (GAMs) (Hendrix, 2009; Tremblay, 2009; Tremblay & Baayen, 2010). However, none these approaches have seen broad adoption, presumably in part due to confusion over how, why, when, and whether they should be used, and how they relate to more traditional methods like averaging.

Our goal is not to add to this confusion by proposing yet another new method. On the contrary: our goal is to show that all of these apparently disparate approaches (or close variants of them that serve the same purposes) can be unified into a single conceptual framework, and to present this framework in a way that makes it accessible to researchers in the field. Critically, we argue that this framework should not be viewed as an alternative to classic ERP averaging, but rather an extension that arises naturally from a careful consideration of why we average in the first place. The result is a method

which combines the essential familiarity of ERP analysis with the flexibility and power of regression.

Because regression, as a technique, has been studied over more than two centuries, there are standard ways of handling most situations that arise in practice. It turns out that if we have an ERP experiment, then these conventions suggest that we simply calculate ERPs. If there is a mild violation of the ERP assumptions — e.g., a partially uncontrolled nuisance variable that is correlated with our variable of interest — then we can statistically control for that nuisance variable while calculating what is effectively an otherwise traditional ERP. Yet the use of a unified framework also allows us to handle even the most complex situations in a straightforward manner. For example, previously, if one wished to use the ERP image technique to obtain a detailed picture of one continuous factor's effect, one could not also use a multiple regression technique to distinguish the simultaneous effects of multiple continuous factors. In this framework, we can easily combine the two methods if appropriate.

4.1 Least-squares estimation for ERPs

We begin by reviewing the theory underlying traditional ERP averaging. ERP analysis starts from the presumption that whenever a particular type of event occurs (e.g., a stimulus appears on a display), then the brain produces a fixed pattern of neural activity — the ERP itself — that is time-locked to that event. In an ideal world, we could just present our stimulus once, measure this response, and be done. But reality, of course, is never that simple. In this case, the difficulty is that there is always a great deal of other ongoing neural activity that doesn't care about our experimental manipulation at all (or if it does care, then not in a way that matches the assumptions of ERP analysis), and what we actually measure on any single trial will be the sum of the ERP activity and this background activity. So the problem is: given that we can't measure the ERP directly, how can we estimate it from the data that we *can* measure?

A note on terminology: the word 'ERP' is often used interchangeably to refer to either the underlying time-locked brain activity that we are trying to measure, or to the output of a numerical procedure applied to data to estimate that activity. In the

present context, this seems likely to cause confusion, so we reserve 'ERP' to refer to the postulated brain activity, and say 'ERP estimate' when discussing any estimate based on data.

To make things simpler, let's focus on estimating the value of the ERP at one single electrode and latency — e.g., 136 ms post-event at electrode Cz. (Once we've worked out how to do that, we can estimate the rest of the ERP by applying our technique again at all the other electrodes and latencies.) The value we're trying to estimate, then, is just a single number, which we call β. To estimate β, we use the measurements we've made of the scalp potential at 136 ms post-event, at Cz, on many trials — this is a list of numbers, which we call y_1, y_2, y_3, ..., y_n. And our assumption is that the physical process which produced these numbers was the summation of the true ERP plus each trial's background "noise":

$$\begin{aligned} y_1 &= \beta + \text{noise}_1 \\ y_2 &= \beta + \text{noise}_2 \\ &\vdots \\ y_n &= \beta + \text{noise}_n \end{aligned}$$

Or, for short, we just write

$$y_i = \beta + \text{noise}_i.$$

Notice that on every trial, the value of the noise (at this latency and electrode) is different, but the value of the ERP is always the same.

At this point, most texts on ERP would suggest that we just estimate β by taking the average of the y_i values. But why, mathematically, is that a good idea?

We know the values of y_i, but not β or noise_i. If we knew what noise_i was, we could just solve for β; contrariwise, if we knew β, we could solve for the noise: $y_i - \beta = \text{noise}_i$. We know neither, so we need some other technique, and the oldest, simplest, and most widely studied heuristic for solving such problems is to use the principle of least-squares. This principle says that we should choose β to be the number

which makes the total squared noise

$$\text{squared noise} = \sum_{i=1}^{n} (\text{noise}_i)^2 = \sum_{i=1}^{n} (y_i - \beta)^2$$

as small as possible. This is an exercise straight out of freshman calculus — to minimize this formula, we first take the derivative:

$$\begin{aligned} \frac{d}{d\beta}\text{squared noise} &= \frac{d}{d\beta} \sum_{i=1}^{n} (y_i - \beta)^2 \\ &= \sum_{i=1}^{n} -2(y_i - \beta). \end{aligned}$$

Then we set it equal to zero:

$$\sum_{i=1}^{n} -2(y_i - \beta) = 0$$

And finally use algebra to solve for β:

$$\begin{aligned} -2 \left(\sum_{i=1}^{n} y_i - \sum_{i=1}^{n} \beta \right) &= 0 \\ \sum_{i=1}^{n} y_i &= \sum_{i=1}^{n} \beta \\ \sum_{i=1}^{n} y_i &= n\beta \\ \frac{1}{n} \sum_{i=1}^{n} y_i &= \beta. \end{aligned}$$

That final formula should be familiar — it says that according to the least-squares principle, the best way to estimate β is to take the average of our measured values, y_1, …, y_n. From a certain perspective, we might even say that this is the whole reason why using averaging to estimate ERPs makes sense in the first place.

4.2 From averaging to regression

We've seen that the traditional averaging method for estimating ERPs can be justified as being the least-squares solution to the equation

$$y_i = \beta + \text{noise}_i.$$

This chapter's central observation is this: the above equation is a particularly simple example of the general least-squares linear regression formula, as can be found in any statistics textbook:

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

So what happens if we estimate ERPs using full-fledged linear regression, instead of just the simplified version?

Just as before, the y_i values are set to equal the measured scalp potential at a single electrode, at a single latency, across different trials. The x_{ji} values (the *predictors*) are set to indicate various properties of the stimulus presented on trial i, coded numerically. (The original ERP equation that we derived above effectively has a single predictor, x_{1i}, whose value is always 1. We can think of this as a particularly vague property that simply says yes, this stimulus is a stimulus.) And, once we've measured the y_i values and specified the x_{ji} values, we again use the principle of least-squares to find those values for $\beta_1, \beta_2, \ldots$ which together minimize the total squared noise. Each β value then gives an estimate of some portion of the ERP at this electrode and latency. Alternatively, for any particular stimulus i, we can compute the sum $\beta_1 x_{1i} + \beta_2 x_{2i} + \cdots$, which is the model's *prediction* for the ERP to this stimulus at this latency.

Actually calculating these β values in practice is somewhat more complicated than just taking the average, but not by much — there is a standard formula for them which any modern computer can use to reliably calculate them in a few milliseconds. We then repeat these calculations many times, once for each electrode and latency. As we do, we keep the same xs — since these represent properties of the event that each trial is time-locked too, which do not vary across electrodes or latencies — but swap out

the y values to represent the measurements made at each electrode and latency across our different trials. Finally, we gather up all the computed β_1 values to make one waveform, all the β_2 values to make a second waveform, and so on for all of the βs. Then these waveforms can then be graphed, smoothed, entered into statistical analysis, and generally treated exactly as if they were ERP estimates obtained from averaging. Likewise, we can combine βs together to compute the predicted waveforms for particular stimuli, and these predictions too can be analyzed like ERP estimates obtained from averaging. To distinguish them while emphasizing their similarity, we refer to these ERP estimates obtained from regression models as *rERPs*.

While rERPs can be treated much like traditional ERP estimates, there is an important shift in our perspective. Many of us are used to thinking of data analysis in terms of deciding how to manipulate our data to achieve a desired result. In the regression framework, we instead focus on setting up the predictors which best describe our experiment, and then our computer's least-squares program takes care of automatically deriving the optimal data processing method from this description.

But how do we define the xs, and how do we interpret the resulting rERP waveforms? Some examples should make this clearer — starting with some that show how familiar ERP ideas can be expressed in the traditional language of linear regressions, and then moving on to more novel capabilities. To make this more concrete, let's pretend that we have a hypothetical experiment to analyze, a variant of the standard oddball paradigm (Squires, Squires, & Hillyard, 1975). In our version, a mixture of standards and targets are displayed at randomly varying eccentricities from fixation (Figure 4.1). This gives us one two-level categorical factor (standard versus target), and one continuous factor (eccentricity).

4.2.1 The basic ERP as an intercept term

The simplest example is the one we have already seen — suppose we define just a single predictor, as:

$$x_{1i} = 1$$

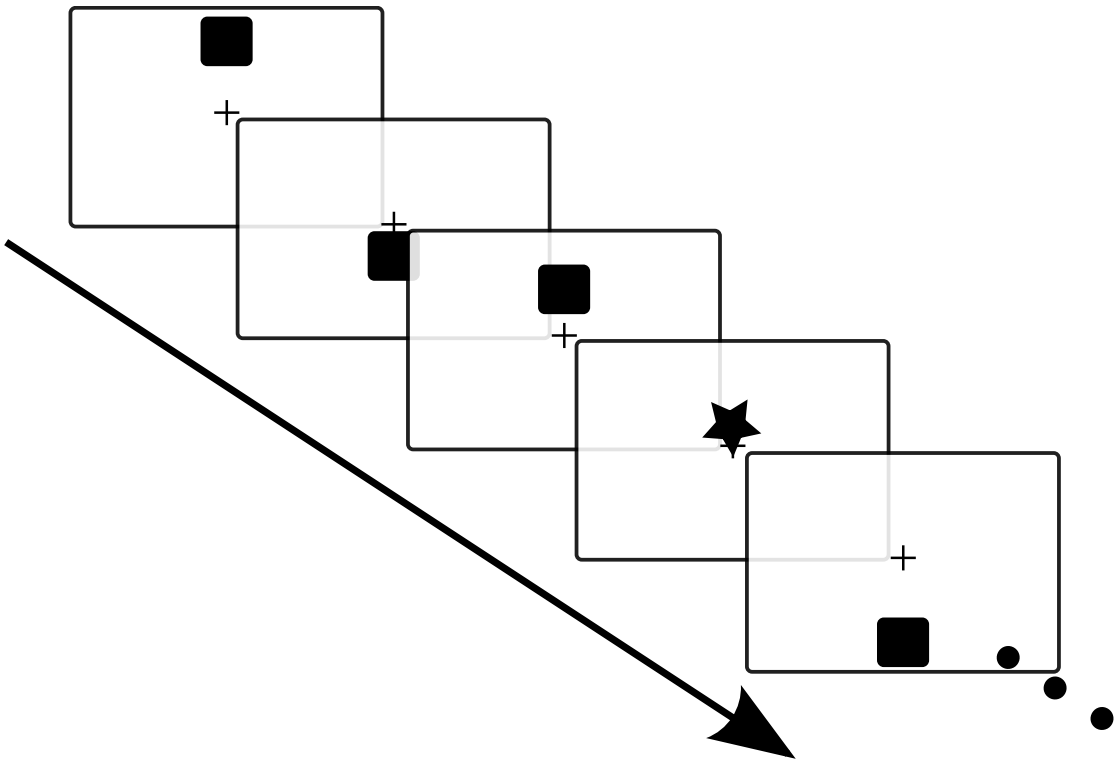

Figure 4.1: A hypothetical oddball experiment used to illustrate the rERP technique. A series of standards (squares) are interspersed occasionally with targets (stars). In addition, each item appears at some varying eccentricity (positive or negative) from fixation.

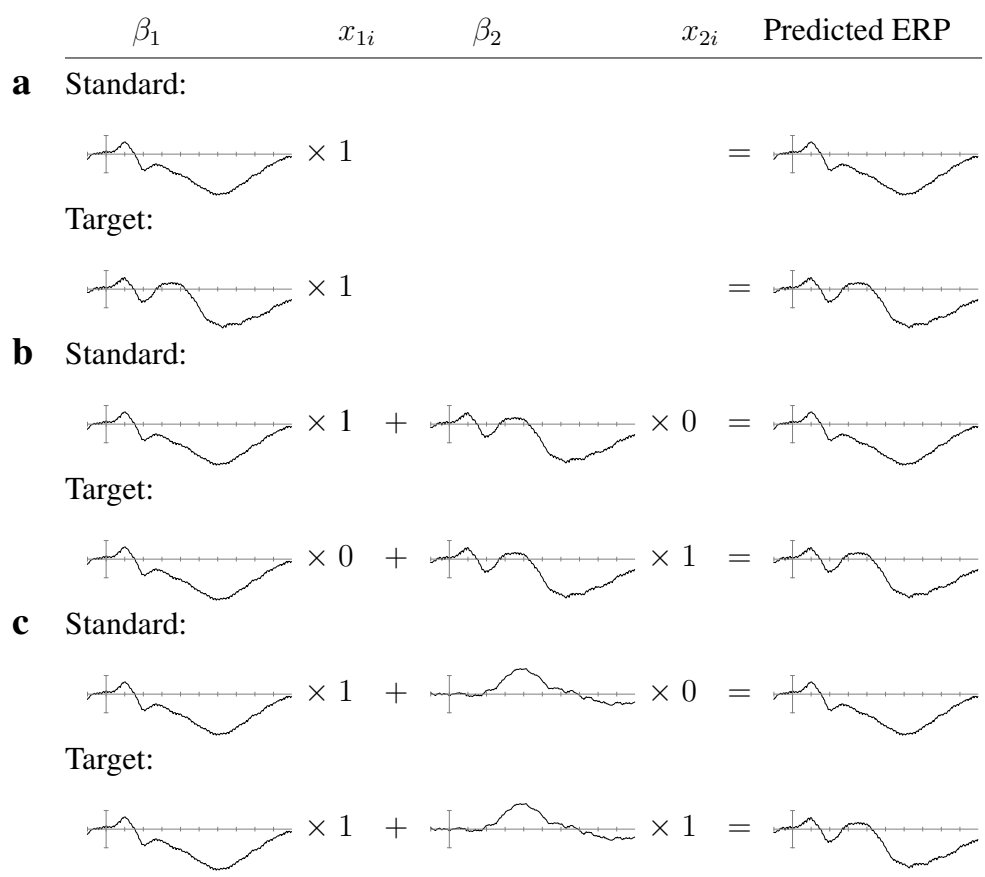

Figure 4.2: The relationship between xs (predictors), β coefficients (rERPs), and predicted ERPs, for various stimuli. Data is fictional. (**a**) Intercept-only models fitted to subsets of the data. (**b**) Dummy coding. (**c**) Treatment coding.

In linear regression terminology, this predictor is known as an intercept term. Then our regression equation is

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

and as we saw above, when we find the least-squares solution, β_1 will simply end up equal to the mean of the y_i values. (Figure 4.2a)

Therefore, this is not only a legitimate method for estimating the activity time-locked to some event, but it produces results that are exactly identical to the conventional averaging technique. However, if we have categorical factors with multiple levels, then it requires us to fit two different models — one on the subset of trials that represent standards, and another on the subset of trials that represent oddballs.

4.2.2 Multiple ERPs via dummy coding

Instead of fitting multiple models, we can estimate both ERPs at once within a single regression model by using a trick known as *dummy coding*, which is one of the standard ways to handle categorical variables within regression models.[1] If we define two different predictors, like so:

$$x_{1i} = \begin{cases} 1 & \text{if stimulus } i \text{ is a standard} \\ 0 & \text{if stimulus } i \text{ is a target} \end{cases} \qquad x_{2i} = \begin{cases} 0 & \text{if stimulus } i \text{ is a standard} \\ 1 & \text{if stimulus } i \text{ is a target} \end{cases}$$

and then plug them into the standard regression equation:

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

then least-squares fitting will set β_1 to the average of the standard trials, and β_2 to the average of the target trials (Figure 4.2b).

It's easy to see why this happens. If i is a standard trial, then x_{1i} is 1 and x_{2i} is 0, so we have

$$y_i = \beta_1 \times 1 + \beta_2 \times 0 + \text{noise}_i = \beta_1 + \text{noise}_i$$

Of, if i is a target trial, x_{1i} is 0 and x_{2i} is 1, so we have

$$y_i = \beta_1 \times 0 + \beta_2 \times 1 + \text{noise}_i = \beta_2 + \text{noise}_i.$$

So effectively we end up fitting β_1 and β_2 on two different subsets of the data. The only difference from the previous example is that before, we did this explicitly; now, we just defined our xs and the appropriate data splitting happened automatically as a result of the least-squares fitting process.

[1] This is the default method of coding categorical variables used by SAS (SAS Institute Inc., 2010), and is also used by default by R (R Development Core Team, 2010) for models that do not contain an intercept term.

4.2.3 Difference ERPs via treatment coding

However, a more common method for handling categorical variables in regression is by *treatment coding*.[2] This consists of dummy-coding all but one of the levels of our factor (this level that's left out is known as the *reference level*), and adding an intercept term. For example, taking standards as our reference level, we have:

$$x_{1i} = 1 \qquad\qquad x_{2i} = \begin{cases} 0 & \text{if stimulus } i \text{ is a standard} \\ 1 & \text{if stimulus } i \text{ is a target} \end{cases}$$

With this coding scheme, least-squares fitting will set β_1 to the average of the standard trials, and β_2 to the difference between the target trials and the standard trials, i.e., β_2 will be a conventional *difference ERP* (Figure 4.2c). Hence the name 'treatment coding': usually, we choose a control condition for our reference level, and each β_j then tells you how you expect your outcome values to change as you apply the relevant non-control 'treatment'.

This also illustrates a very important aspect of interpreting regression formulas, which is that the fitted value of each β_j depends not just on how we defined the jth predictor, but on *all* the predictors, $x_{1i}, \ldots, x_{ni}$. In both this example and the previous one, we used the exact same definition for x_{2i}, but β_2 changed completely depending on the how x_{1i} was defined. (This is why we prefer the name treatment coding for this particular coding scheme, even though SPSS and many references refer to it as, simply, 'dummy coding'. As we've seen, using 0/1 dummy coding for some levels of a factor can produce very different results depending on how other levels are coded, but treatment coding refers specifically to this scheme in which all-but-one level are dummy-coded.) Likewise, we saw above that if all you have is an intercept, then the corresponding β will give the grand mean of all your data, but here it gives the mean of the standards only. This behavior is actually the source of one of regression's great strengths, but it can be quite confusing when first encountered.

The key to understanding what will happen in these models is to remember that least-squares fitting is focused entirely on the predicted values (the rightmost column in

[2] This is the default method of coding categorical variables in R and SPSS (IBM, 2010).

Figure 4.2); it will pick whichever β values make these predictions match the data as closely as possible. In this case, in order to make the predictions for standards as accurate as possible, the fitting algorithm has to set β_1 to be equal to the ERP for standards, and then for targets, our predictors are defined in such a way that it has to use β_1 there as well. So to make the prediction for targets come out, it has to set β_2 to be whatever value will give the right predictions *after being summed with* β_1.

Imagine what would happen if it instead set β_2 to the ERP for targets, as in the pure dummy coding example above. Its final prediction for targets would then be the sum of the ERP to standards and the ERP to targets, resulting in a doubled N1, P2 etc. This, obviously, is not supported by the data! Since β_1 does a good job of explaining the initial N1/P2 complex for targets, there is nothing left in this time period for β_2 to do. The more general principle is that in cases like this, where there is some potential redundancy between predictors — where there are multiple βs that have some chance of explaining the same pattern in the data — then the least-squares fitting process will automatically divy up responsibility between them, in whatever way explains the data best. Such redundancy is better known as *collinearity* or *confounding*, and one of the great strengths of regression is that — given sufficient data — it can accurately analyze data with even high degrees of collinearity.

Of course, this only works when your predictors are partially, but not completely, redundant. An example of complete redundancy would be if we entered the same predictor twice, or — less obviously — if we used the simple dummy coding scheme above, but for two different factors at once. For instance, suppose in our oddball experiment we added a second categorical variable by making some of the displays red, and some blue. Then we could define four predictors:

$$x_{1i} = \begin{cases} 1 & \text{if stimulus } i \text{ is a standard} \\ 0 & \text{if stimulus } i \text{ is a target} \end{cases} \qquad x_{2i} = \begin{cases} 0 & \text{if stimulus } i \text{ is a standard} \\ 1 & \text{if stimulus } i \text{ is a target} \end{cases}$$

$$x_{3i} = \begin{cases} 1 & \text{if stimulus } i \text{ is red} \\ 0 & \text{if stimulus } i \text{ is blue} \end{cases} \qquad x_{4i} = \begin{cases} 0 & \text{if stimulus } i \text{ is red} \\ 1 & \text{if stimulus } i \text{ is blue} \end{cases}$$

But this would be a bad idea, and most well-written regression programs will give an

error if you attempt to fit such a model. To see why, suppose that there is some component — perhaps the N1 again — which is identical between all four conditions. Then one way to explain it would be to say that seeing a standard causes and N1, and so does seeing a target. Another way to explain it would be to say that N1s are caused by red items and also by blue items. Or perhaps standards and targets both cause a positivity during this window, while red-ness and blue-ness both cause an even greater negativity to cancel it out. The end result is that there are many different combinations of β values which all produce the same predicted ERP; and since least-squares fitting works by trying to find the best match between predictions and data, it has no way to distinguish between different β values that make the same prediction.

This problem does not arise for treatment coding, which is why it is generally preferred over strict dummy coding. Following the treatment coding idea, we would handle this same data by choosing as our predictors an intercept (x_{1i}), the dummy coded versions of all *non*-reference levels for both of our categorical factors (x_{2i} and x_{3i}), and then adding an interaction term (x_{4i}):

$$x_{1i} = 1 \qquad x_{2i} = \begin{cases} 0 & \text{if stimulus } i \text{ is a standard} \\ 1 & \text{if stimulus } i \text{ is a target} \end{cases}$$

$$x_{3i} = \begin{cases} 0 & \text{if stimulus } i \text{ is red} \\ 1 & \text{if stimulus } i \text{ is blue} \end{cases} \qquad x_{4i} = \begin{cases} 0 & \text{if stimulus } i \text{ is either red or a standard} \\ 1 & \text{if stimulus } i \text{ is a blue target} \end{cases}$$

(Notice that $x_{4i} = x_{2i} \times x_{3i}$, which is the general rule for defining interaction terms.) In this case, β_1 would give the ERP for red standards, β_2 would give the difference ERP between red targets and red standards, β_3 would give the difference ERP between blue standards and red standards, and β_4 is a difference-of-differences ERP — if we first calculated the difference ERP between blue targets and blue standards, and then calculated the difference ERP between red targets and red standards, then β_4 would be the difference between these two difference ERPs. β_4 will be significantly different from zero exactly when there is a non-additive interaction between our two factors.

4.2.4 'Slope' ERPs from numeric predictors

By this point it's hopefully clear how standard ideas from ERP analysis correspond closely to some of the conventional ideas from regression theory. Now we'll see how the regression approach also allows us to analyze a continuous factor like eccentricity. (This approach has been used to study work recognition in EEG; Hauk et al., 2006, 2009.[3]) The core idea is very simple. We can simply define a predictor that indicates the eccentricity of each trial. In practice, we'll always want to include an intercept term as well; otherwise, we're assuming that whenever our predictor is zero, the ERP will be perfectly zero as well. (There may be some cases where this makes sense — perhaps if the predictor indicated visual contrast, then we'd expect no response at all to a zero-contrast display. But such cases are unlikely to arise often in practice.) To look at the eccentricity effect, we can define predictors:

$$x_{1i} = 1, \qquad x_{2i} = \text{eccentricity on trial } i\text{, in degrees of visual angle.}$$

Now β_1 will give a kind of baseline ERP — the ERP we expect to see for items with $x_{2i} = 0$, i.e., items displayed in the center of the screen. β_2 estimates how much this ERP changes with each 1° change in visual angle (Figure 4.3). It is the slope of the regression line that relates x_{2i} and y_i.

Notice the similarity between these predictors and the predictors we used for treatment coding above. In both cases, we have an intercept term x_{1i} whose corresponding parameter β_1 gives us a kind of baseline ERP, and then there is an additional term x_{2i} which codes for deviations from this baseline. With treatment coding, this second parameter β_2 measures the difference in scalp voltage between targets and standards, and has units of μV. Here, β_2 measures the difference in scalp voltage corresponding to any 1° change in eccentricity, and has units of μV per degree of visual angle. Our slope rERP β_2 is a new variant of the familiar difference ERP.

At this point, though, we might get worried. What if going from 0° to 1° is different than going from 10° to 11°? As currently phrased, our model assumes that

[3] Hauk et al have also argued that ERP averaging is a special case of regression, on the grounds that both are linear operations. However, we believe that the fact that both are the least-squares solution to the problem of estimating ERPs embedded in noise is a deeper connection.

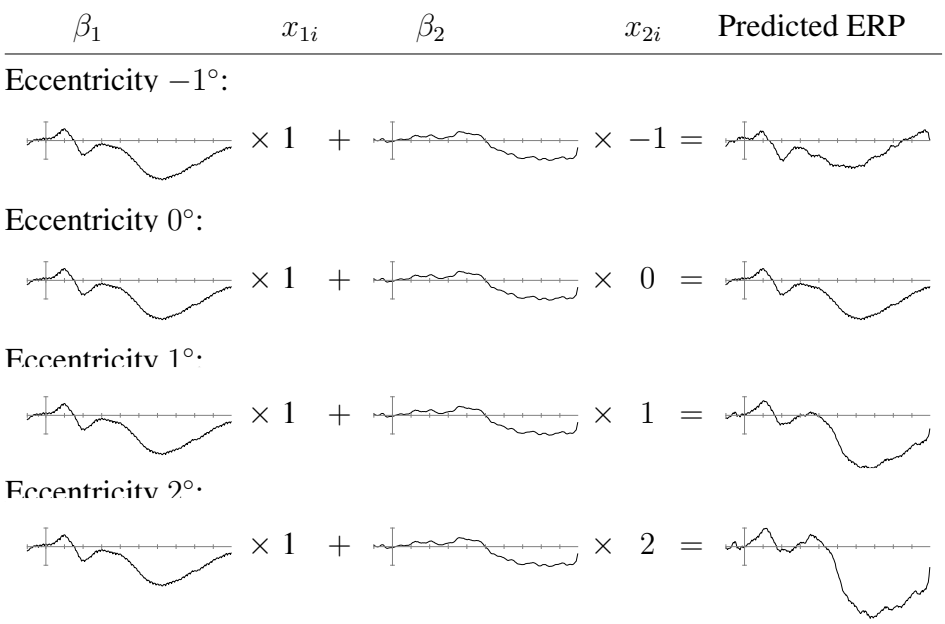

Figure 4.3: The relationship between xs (predictors), β coefficients (rERPs), and predicted ERPs, for various stimuli in a model containing a single continuous predictor. Data is fictional.

these are the same — it assumes that the effect of eccentricity is *linear*. When this assumption is true, then it gives this technique additional statistical power. We can think of this as the fitting process automatically finding all the 0° trials and comparing them to the 1° trials, and then pooling this together with comparisons of 10° trials versus 11° trials, and all other such comparisons; this increases the effective data set size, and allows for more accurate estimates (Cohen, 1983).

Nevertheless, we may not wish to take this linearity on faith, and we will return to discuss methods for estimating non-linear relationships in a later section. For now, we settle for pointing out that if we know how the ERP response to our stimulus property deviates from non-linearity, then we can transform this property when defining our predictor. For instance, word frequency famously has a logarithmic effect on reaction times (Howes & Solomon, 1951; McCusker, 1977); for word stimuli we might want to define a predictor like $x_{2i} = \log(\text{Frequency of word } i)$. In fact, we are allowed to apply arbitrary transformations to stimulus properties when defining our predictors — the 'linear' in 'linear regression' means that the relationship between our outcome (the predicted ERP) and our predictors (the x_{ji}) has to be linear. The relationship between

the properties we care about and the x_{ji} values that we actually enter into the regression can be anything we like. There are only two things we must keep in mind when using such transformations: first, that the resulting β coefficient will describe how the ERP changes with each unit change in x_{ji}, which may be different from a unit change in the original stimulus property. And second, that — as usual — transforming one predictor in our regression may cause the other β values to change as well, in order to keep the overall predicted ERP as close as possible to the data.

One particularly simple and useful transformation is to *center* our predictors; that is, to subtract the mean value off of each before entering them into the regression model. Assuming our model has an intercept term, this will not change the model's predictions at all. But recall that the β for the intercept term corresponds to the predicted ERP when all other predictors are zero. If we change our other predictors so that they have the value zero at their average value, rather than at their natural zero point (which may not correspond to any possible stimulus), then the intercept term's β will become interpretable as the ERP for an 'average stimulus'.

Of course, even if we don't center our predictors, we can always plug the average value of our predictors into the regression formula to calculate the predicted ERP for an 'average stimulus', and this estimate will be identical to the one we would have gotten by centering our predictors and re-fitting our model. So centering our predictors has largely an aesthetic, rather than substantive, effect on the analysis. But aesthetics matter; it makes interpreting such models just a little bit simpler for a trivial amount of effort, so it still seems worth following as a standard practice.

4.2.5 Putting it all together

So far we've analyzed categorical factors, and a continuous factor, but what happens when you have both? By this point, the answer is hopefully obvious: we simply put both sets of predictors into a single model. (This combined approach has been used to study the EEG response to face recognition: Rousselet et al., 2008, 2009, 2010.) To analyze our hypothetical oddball experiment, we'd want a model something like this:

$$x_{1i} = 1$$

$$x_{2i} = \begin{cases} 0 & \text{if stimulus } i \text{ is a standard} \\ 1 & \text{if stimulus } i \text{ is a target} \end{cases}$$

$$x_{3i} = \text{eccentricity on trial } i\text{, in degrees of visual angle}$$

(See Figure 4.4). The resulting βs have the same interpretration as they did in the previous examples in which these same predictors occurred — β_1 gives the predicted ERP for standards with zero eccentricity, β_2 gives the difference between targets and standards (under the assumption that this is the same for all eccentricities), and β_3 gives the change in the predicted ERP for each unit change in eccentricity.

The only difference from before is that now, since the predictors all appear in a single model, the predictors must work together cooperatively to explain the data. If you have two predictors which are correlated with each other, then the model will be forced to choose which of them — or what combination of them — best explains the data. Therefore, if you were to actually run this or a similar experiment, then this is the very simplest model we would ever recommend using for analysis. (Please ignore our earlier examples that used only a subset of these terms — those were presented for purely pedagogical purposes!) The reason for using as rich as possible a model by default is simple. If you have a designed experiment with orthogonal factors, where each x vector of predictor values is orthogonal to all other predictors in your model, then it doesn't matter whether you analyze each factor one at a time in separate models, or all at once in a single model — you'll get exactly the same β coefficients out either way. On the other hand, if you don't have an orthogonal model, then you have, by definition, a confounded model — and in the presence of confounding, analyzing single factors at a time may be actively misleading. In fact, it might be wise to go further, and add (at least) a term to represent the interaction between targets versus standards and eccentricity. Such a term would be defined as $x_{4i} = x_{2i} \times x_{3i}$, and if it were included, then β_3 would denote the eccentricity effect for standards, while β_4 would denote the *difference* between the eccentricity effect for standards and the eccentricity effect for targets — note the similarity to our earlier discussion of categorical interactions.

As usual, there is a tension here. As we add more flexibility to our model, then we find ourselves trying to extract more information from a fixed amount of data. At

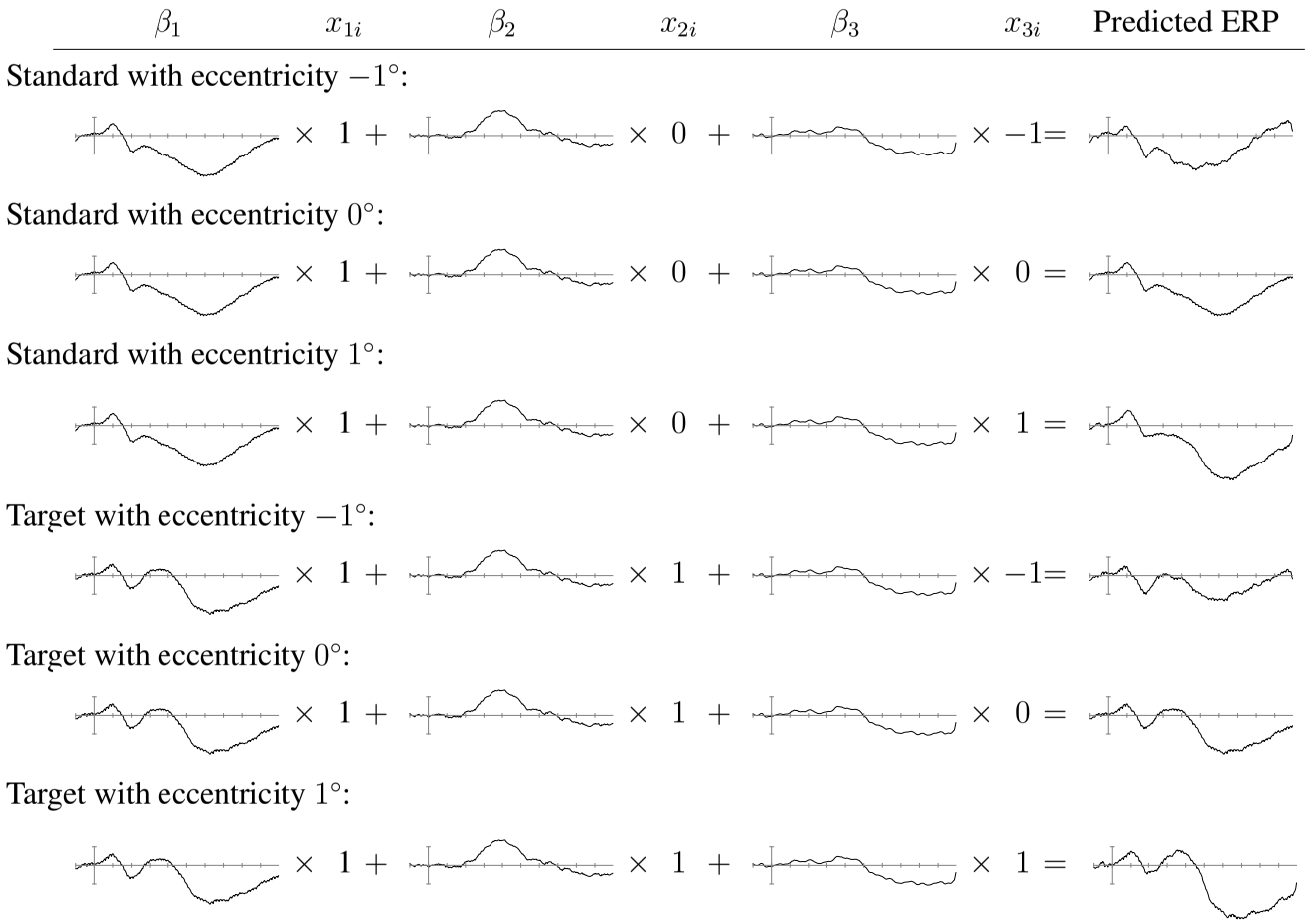

Figure 4.4: The relationship between xs (predictors), β coefficients (rERPs), and predicted ERPs, for various stimuli in a model containing both a categorical and continuous predictor. Data is fictional.

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. But in practice, such an analysis is unlikely to 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 pool data across multiple trials to achieve stable estimates. So we have to strike some trade-off that allows us to achieve acceptable stability with maximum accuracy. One of the great advantages of regression models is that they let us move mindfully between these extremes, and to use relatively mild assumptions when possible to let the data speak.

As more terms and interactions are added to a model, it may become a challenge to interpret the various β coefficients. The key to achieving clarity in such situations is always to consider how the coefficient under consideration affects the overall predicted ERP. We find that drawing pictures like those shown in Figure 4.2 often helps.

4.3 Non-linear regression

The models we've discussed so far make a very specific assumption about the effect that any continuous factor will have on the ERP: that it will be linear. This is unlikely to be strictly true in most cases, and often we will not know what relationship actually holds in its place (indeed, this may itself be our research question!). Fortunately, the regression framework provides a variety of standard techniques for estimating non-linear relationships from data.

Such techniques are powerful and, when used appropriately, can do anything that a simple linear model can do and more. But 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.

The assumption of linearity is not so arbitrary as it may seem, and may in fact be

a viable simplifying approximation in many cases.[4] 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. For instance, consider the logarithmic relationship between predictability and reading time that we graphed in a previous chapter. While this curve is highly non-linear, it only begins to bend sharply at low probabilities. If we had only looked at word probability in the range of 0.05–1.0 — as most ERP studies do — then this curvature would have been mild, and a linear model would have captured the data nearly as well as a logarithmic one. Furthermore, while U-shaped relationships do occur in neuropsychological research, 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 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. Even 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 variety of techniques used to accomplish this, they in general take the form of somehow requiring our curves to be simpler and less wiggly. The fewer data we have, the stronger these assumptions must be. When our data is 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

[4] It's also worth noting that 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 which are matched on a large number of word properties that would otherwise become possible confounds, such as cloze, frequency, concreteness, orthographic neighborhood size, and so forth. Ideally, we would choose pairs of words such that each pair was perfectly matched on all of these properties, but in practice we often 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 concreteness, say, has a non-linear effect on the ERP, then even with all other confounds controlled for, the average ERP to a set containing one word with concreteness score 1.0 and one word with concreteness score 3.0 may not be the same as the average ERP to a set containing two words that both have concreteness scores of 2.0, even though the average concreteness is the same in both cases.

linear regression to start with.

On the other hand, there are situations where allowing non-linearities in our regression is particularly valuable. The most obvious is when the shape of the relationship is itself a target our of our analysis (as, for example, in the previous chapter). A less obvious case occurs when our analysis includes control factors — factors whose effect we don't actually care about, but that are correlated with factors that do care about. It's wise to include them in our model anyway in order to ensure that we don't accidentally attribute whatever effect they do have to our factors of interest. For such factors, we aren't particularly worried about giving them too much flexibility and allowing them to overfit the data — we don't plan to try interpreting them anyway, and the worst that could happen is that they soak up more variance than they really deserve. This is far better than a model which is too inflexible, and fails 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.

The third case where a non-linear model is useful is when we have reason to believe that there may be a U-shaped effect. That may be obvious, but in the ERP context, there is a less obvious consequence: if some factor cause the latency of a component to vary, then that factor will produce a U-shaped effect at some time points. For instance, on the go trials in the go/no-go task we analyze below, there is a positivity associated with the response, and in the ERP this positivity therefore has a latency which varies with reaction time (RT). A fast response is around 300 ms; a slow response around 600 ms. Therefore, at around 400–600 ms in the ERP, you see an increased positivity for medium-length RTs as compared to both short or long RTs. This is a U-shaped effect of RT on ERP amplitude.

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

In order to fit non-linear models with linear least-squares regression, we need a trick. That trick is to pick some particular set of *basis functions*, $f_1(x), \ldots, f_k(x)$, which are chosen so that taking weighted sums of them, $\alpha_1 f_x(x) + \cdots + \alpha_k f_k(x)$, lets

us construct a wide variety of non-linear curves. Different sets of basis functions lead to different spaces of possible curves, and there are some good standard sets which are generally used. More important is the choice of how many basis functions we wish to use, since the more we use, the more flexible our space of possible curves — and the more data we need to get sensible estimates. The clever part is that when it comes time to do the regression, we define a set of different predictors, one for each basis function:

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

Now the least-squares fitting will do its usual thing, and find whatever set of weights $\beta_1, \ldots, \beta_k$ cause these predictors to best match our data. As a result, the individual βs are generally not very interpretable on their own, but they tell us which curve in our space of many possible curves best fits the data: it's the curve you get by taking the weighted sum $\beta_1 f_1(x) + \cdots + \beta_k f_k(x)$. We've tricked a linear technique into fitting a non-linear curve.

As usual, then, we redo this fitting process at each latency relative to the time-locked event, and arrive at a set of β values for each time point. That part's straightforward. The challenge is what to do with these β values. Before, we had only a single β value at each time point, so we simply graphed that against time. Now, we have a set of β values, none of which are meaningful on their own — really, what we have is a full curve at each time point. In order to visualize the curve for a particular time point, we use a colormap to convert each point on that curve into a dot of color, and we stack these dots vertically. When we repeat this process at all of the different time points, we produce a two-dimensional colormapped image with time on the x-axis, our factor of interest (e.g., RT) on the y-axis, and the color of each point representing the predicted amplitude of the ERP response to a particular value of our factor at a particular point in time.

The simplest way of creating basis functions is to divide up the possible values

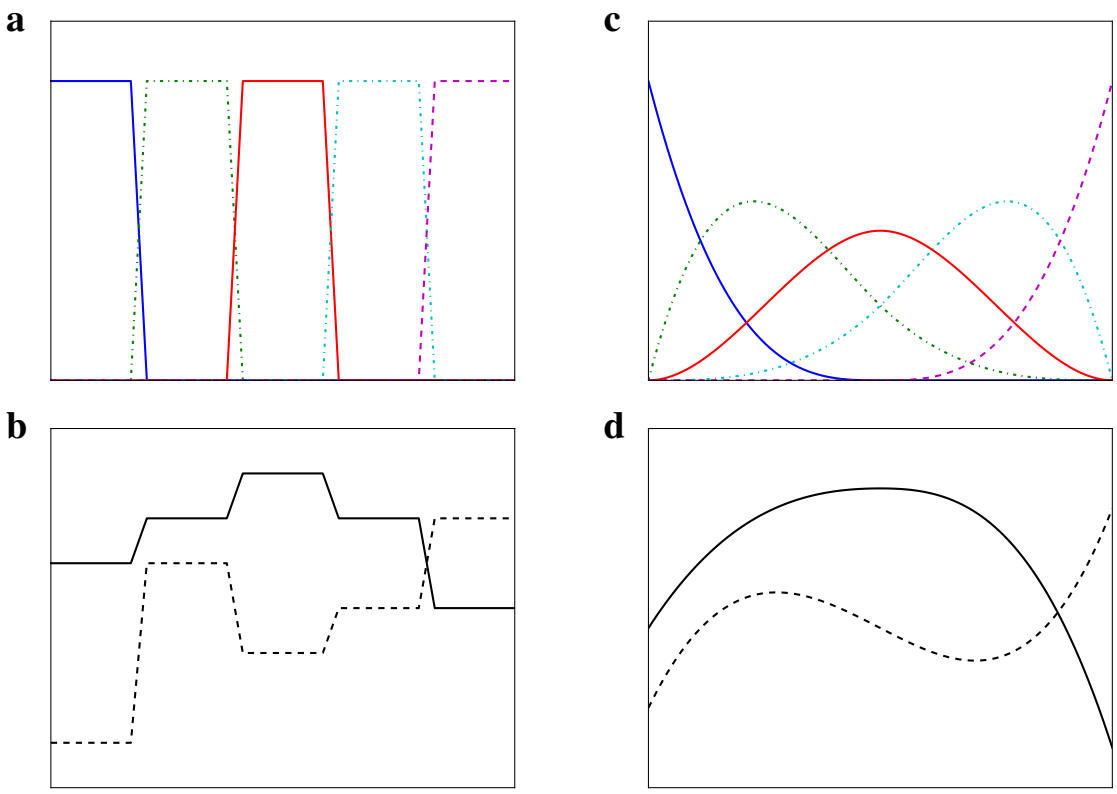

Figure 4.5: Two sets of possible basis functions for use in fitting non-linear curves to data. (**a**) A step-function basis with 5 functions. (**b**) By taking weighted sums of the step functions in (**a**), we can construct any piecewise-constant function. Two examples are shown here. (**c**) A cubic B-spline basis 5 functions, as constructed by the function `bs()` in R. (**d**) By taking weighted sums of the spline functions in (**c**), we can construct a wide variety of smooth curves.

for our factor into several non-overlapping ranges. For each each range, we define one basis function as

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

(See Figure 4.5a). By taking weighted averages of these functions, our regression model is able to construct any function which is piecewise-constant on these ranges (Figure 4.5b). Notice that when we then use these basis functions to make predictors x_{ji}, these predictors will be very similar to the dummy- or treatment-coded categorical predictors above; 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 factor into two or more bins (e.g., a median split) and then averaging across each bin (Kutas & Hillyard, 1984, e.g.).

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 spline functions. While there are a number of different types of splines (which you choose usually matters little in practice), one example is shown in Figure 4.5c. The characteristic features of splines are twofold: first, like the step functions we saw earlier, each basis function covers a fairly short range of the data, with minimal overlap. As a result, the behavior of our fitted curve near any particular value will primarily be determined by the data which 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 smooth curves (Figure 4.5d).[5]

In Figure 4.6, we demonstrate the results of using these two sets of basis functions to analyze data from a mid-line parietal site in 1077 go trials from 7 participants

[5]It is sometimes suggested that one could 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, and better methods are available.

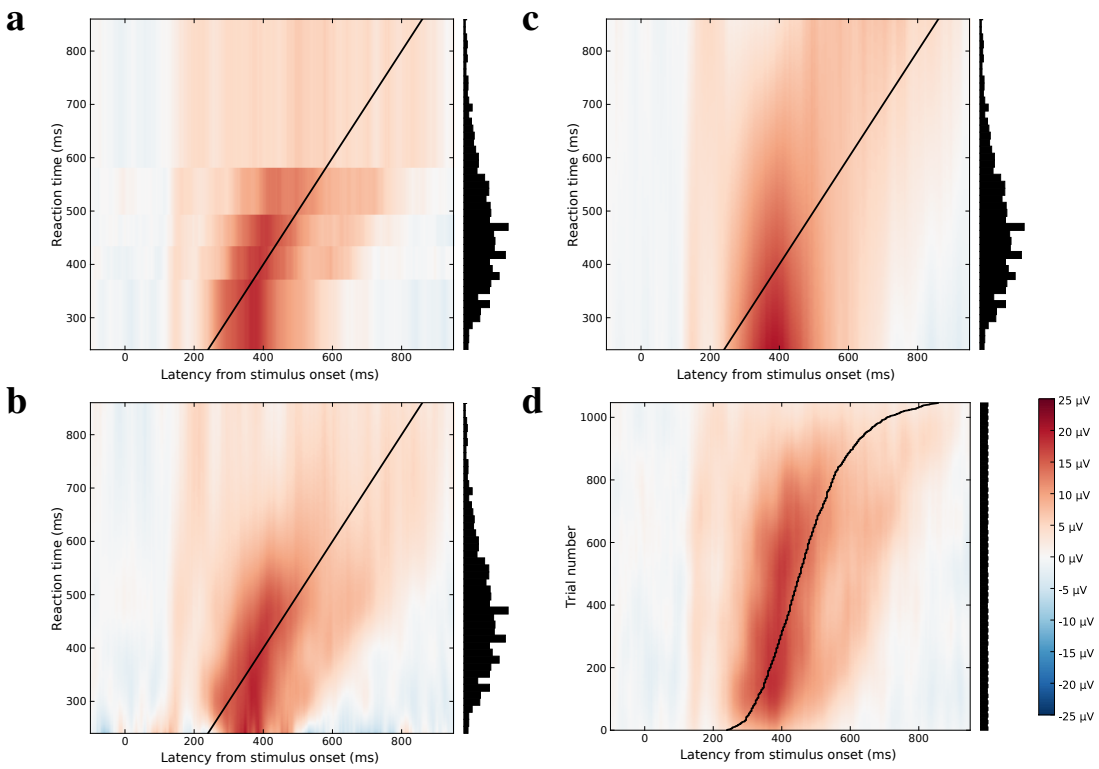

Figure 4.6: Demonstration of non-linear rERPs applied to analyzing an effect on the latency of an ERP positivity. (**a**) Regression using a step-function basis set containing 5 step-functions. This is equivalent to the standard approach of splitting the data into 5 equal-sized bins and calculating the average over each. Each horizontal line across the image represents one predicted ERP. (**b**) Regression using a cubic B-spline basis set with 5 degrees of freedom. (**c**) Simple linear regression, as described in previous sections, but plotted as an image for comparison. (**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 $\sigma = 75$ trials). In each plot, a black line shows where the response fell within each epoch, and a histogram on the right shows how many trials were observed in each range of the y-axis. (**a**)–(**c**) are rERPs. (**a**) and (**d**) are standard methods in the literature. The difference in the histogram and overlaid RT plot between (**d**) and the other images is due to the difference in y-axes.

in a go/no-go task[6]. We also include the result of using a simple linear model to analyze these data (Figure 4.6c). We could plot this as a conventional waveform, but since a line is just a particularly simple curve, we also have the option of plotting it using the same two-dimensional technique as we use for the non-linear regression estimates. We do so here, for comparison.

Notice that in Figure 4.6a and **b**, there is a pronounced diagonal pattern visible in the 400–600 ms range. This diagonal pattern is what a latency effect looks like. But it is absent in Figure 4.6c; during this period, the estimated slope coefficient is essentially zero. This is because in this region, the latency effect creates a U-shaped relationship between our factor RT and the ERP response: the ERP shows a positivity to RTs on the order of 450 ms, but not RTs which are either faster or slower. The linear model is unable to capture this U-shaped relationship. The linear model does successfully capture the effect before and after this region, but if we looked only at it, we might have instead interpreted this effect as a positivity to short RTs at $\sim$400 ms followed by a negativity at $\sim$600 ms. (Of course, in principle it is still possible that some complicated combination of fixed-latency positivities and negativities would produce the observed pattern, but once we've see the non-linear fit the varying-latency interpretation becomes much more plausible.) This is another way that slope rERPs are similar to conventional difference waves — if we had divided our RTs into just two bins and averaged them, then we would face the same ambiguity. Detecting a latency effect requires the use of more than two bins, or, equivalently, some form of non-linear regression.

On the right of each image, we plot a histogram showing how much data was available in different parts of the range of RT values. In general, the curves will be more trustworthy in areas where more data is available. In the top portion of Figure 4.6b we can see the effects of data sparsity reflected in the more smeared out curve estimates.

Finally, we also include an ERP image (Jung et al., 2001; Lorig & Urbach, 1995) of this same data (Figure 4.6d). An ERP image is produced by simply sorting a number of trials according to some factor of interest (RT, in this case), and stacking the raw data from these trials on top of each other. 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

[6]Data from Urbach & Kutas, 2008. Thanks to Tom Urbach for sharing this data.

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 non-linearities are present. In fact, while we focus here on the basis function approach for its simplicity, the use of such non-parametric smoothers is a well-studied alternative technique for peforming non-linear regression (e.g., Hastie & Tibshirani, 1990).

The main difference between the ERP image and the non-linear rERP estimate is the y-axis. For the ERP image, this is the trial number; if we had seen a different distribution of RTs, then our image would be stretched and squished accordingly. 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 would not change the overall shape of our image.

This invariance is arguably an advantage of the rERP approach, but probably doesn't make a huge difference in practice. What's more important is that by computing rERPs, you can use the same technique to analyze categorical effects, linear effects, non-linear effects. (In fact, Figure 4.6a, b, c, and Figure 4.7a and b were all produced by the exact some software routine, with the only thing changed being the matrix of predictor values x_{ji} that it was given as input.) Furthermore, in situations where it would be useful, these techniques can all be combined, or further bells and whistles added (such as those described in the next chapter). So, for example, we can have linear factors as controls on our non-linear rERP estimate, or even fit multiple non-linear effects simultaneously. In that case, the overall predicted ERP for any particular stimulus would be calculated by reading out the particular row of each rERP image which corresponded to this stimulus's properties, and adding these rows together, just as in Figure 4.2.

In Figure 4.6, in order to facilitate comparison with the linear model, we used a small number of basis functions, which has the effect of strongly smoothing the data. To demonstrate how changing the size of our basis function set affects our results, Figure 4.7 repeats this analysis with ten times as many basis functions, and as you see, this produces pictures which are both more detailed and noisier.

More sophisticated regression strategies can make this trade-off by automatically optimizing some measure of overfitting based on the data itself (Wood, 2004, 2006),

and are currently our preferred method for fitting such models, at least when appropriate software is available. These techniques have previously been applied to EEG by Hendrix (2009), Tremblay (2009), and Tremblay and Baayen (2010). Besides using automatic smoothness selection, the main difference between their approach and the one described here is that they use two-dimensional splines that are functions of both their factor of interest (such as RT) and also latency. To understand this distinction, consider Figure 4.6b. In this figure, each column of data has been smoothed by fitting splines that are functions of RT, but as you move across rows, the values can potentially change in arbitrary ways, since a new separate set of splines was fit at each latency. We have, in effect, treated latency as a categorical variable, with every latency given a bin to itself, and we can think of these figures as showing how the non-linear interaction of latency with RT affects scalp potential. They, instead, use a two-dimensional spline basis which smooths across both columns (RT) and rows (latency). We are not convinced that adding latency smoothing is useful. As can be seen from our figures, even without latency smoothing we generally do not see radical changes as we move across rows of the figure, simply because this is not supported by the data; additional smoothing does not seem necessary. On the other hand, it would carry risks; it is effectively a form of low-pass filtering, and like any low-pass filtering, might distort the temporal structure of our data, for instance by creating the appearance of an effect starting earlier than it actually does (Luck, 2005). And finally, we know that ERPs are usually more wiggly immediately after a stimulus is presented (e.g., in the N1/P1 complex), and are smoother later in the epoch. Current methods for automatic smoothness selection apply a uniform criterion across the epoch, which does not seem appropriate for ERP data. Indeed, they avoid this issue by performing separate analyses within multiple short epochs (0–250 ms post-stimulus, 200–450 ms, etc.), allowing for different smoothness criteria in each; but one might just as well take this further yet and analyze each latency separately, as we suggest. We do, however, return to this idea of treating latency as a categorical predictor in the next chapter, though for other purposes.

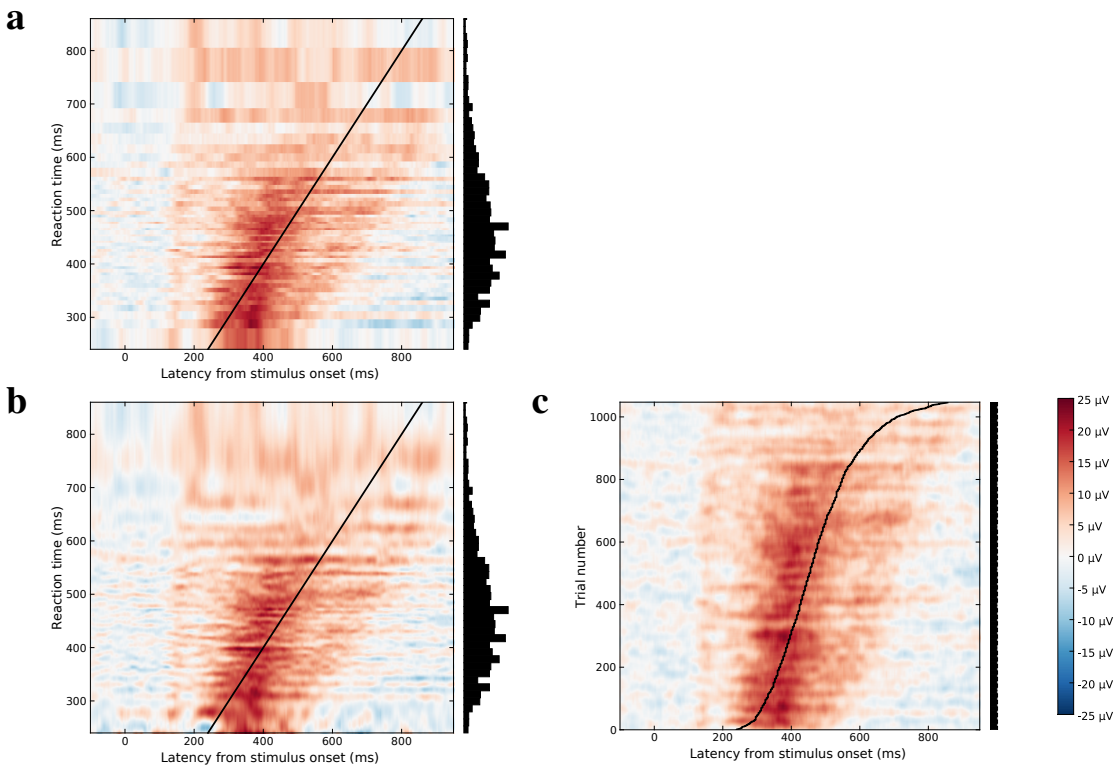

Figure 4.7: For techniques which can detect non-linear effects, we have a trade-off to make. Higher degrees of smoothing produce more stable estimates; but they may also hide structure present in the data. To demonstrate the effect of this change, we repeat the analyses from Figure 4.6, but with less smoothing. (**a**) Regression using 50 step functions as our basis set. (**b**) Regression using a cubic B-spline basis set with with 50 degreees of freedom. (**c**) An ERP image for comparison (smoothed by a gaussian-weighted moving average with $\sigma = 10$ trials).

4.4 Further considerations

It should be clear, now, how the rERP approach works to analyze both simple and complex designs. But there is more to ERP analysis than just averaging, and here we discuss how the rERP approach interacts with some of these ancillary issues.

4.4.1 Baselining

Because the potential at any scalp electrode tends to drift over time, and because we are generally interested in how brain activity changes response to some particular event, when computing ERPs we usually subtract off the average potential observed during some short pre-event time-window. This is known as *baselining*. The same logic still applies when using regression in place of averaging, so we recommend that you continue to baseline as normal.

There are two slightly different procedures you might use to accomplish this. You could subtract off the baseline from each epoch, and then regress, or you could regress first, and then subtract off the baseline from each of the resulting β waveforms. When it comes to traditional average, this is a distinction without a difference, since either procedure produces the same results. Surprisingly, perhaps, the same holds true for the pointwise least-squares technique described in this paper — you are free to baseline either before or after regressing, and will achieve the same results either way (see Appendix for proof).

4.4.2 Filtering

It is sometimes useful to apply a digital filter — usually low-pass — to reduce the effect of artifacts like muscle activity on the ERP. Luck (2005) recommends that this kind of filtering should be applied only for visualization purposes. Conveniently, it turns out that for traditional ERPs, it makes no difference whether we filter our data and then average, or average our data and then filter it. This means that the workflow in which we perform our slower and more data-intensive preprocessing and averaging once and then try out different filter settings (while saving the raw ERPs for statistical analysis) is not only more virtuous than filtering first, but also produces the same results more quickly.

Fortunately, this same property again holds for the pointwise rERPs described in this chapter: it makes no difference whether you filter and then regress, or regress and then filter (see Appendix). Therefore we recommend that you always perform regression on the raw EEG data, before any digital filtering is applied.

4.4.3 Significance testing and averaging over windows

At present, the most common approach for performing statistical tests on ERP data is to first calculate single-subject ERPs, then pick a time window for analysis, and compute the average amplitude of the ERP over the time window, which for each subject results in a single number per condition and channel. After calculating these numbers for each subject, they are entered in an ANOVA, with condition and channel entered as factors.

This approach can be applied straightforwardly to rERPs. We have two choices: either we can average over the β values in some time-window directly, or, if we want to compare two conditions that aren't straightforwardly represented by a single β waveform apiece, we can combine several β waveforms together as in Figure 4.2 to produce a predicted ERP for each of the conditions we want to compare, and then average over our time-window in the predicted ERP.[7] Either way, everything works pretty much the same way as it does with analyzing traditional ERPs. Let's call this method 1.

There is another approach which might seem tempting, now that we're using regression, which is to take use regression's standard hypothesis testing tools, such as t and F tests. It's not clear that this would serve much use, though, for the same reason that when analyzing conventional ERPs, we don't often use t tests to compare our averages: there just isn't much power to detect an effect at a single time point, and then even if there were, we'd still have to come up with some way to combine our results from multiple subjects.

Still, if we are intent on using regression to perform our statistics, then we can press on to overcome these difficulties. The first problem, the lack of power, comes

[7] Sometimes people recommend playing complicated games with factorial coding schemes in order to rearrange a model so that its βs will straightforwardly represent the conditions you want to compare. For our present purposes this would be a waste of time, since the suggested method will produce the exact same result with much less bother.

from entering individual time points into our regression. To overcome this, we could calculate the average amplitude within some time-window of interest in each individual trial, and then enter these averages into regression instead. Then, we need some way to make inferences across subjects. Within the regression framework, there are two standard options for doing this. The more traditional method is to first perform a standard regression within each subject, and then treat the resulting regression coefficients as a new set of data, and enter them into a t-test or ANOVA (Lorch & Myers, 1990). Let's call this method 2. The more modern approach (method 3) would be to use mixed-effects regression instead of standard least squares (Baayen, Davidson, & Bates, 2008). To our knowledge, two groups have previously used this general approach for EEG analysis, both choosing to use mixed-effects regression for generalizing across subjects (Dambacher et al., 2006; Amsel, 2011).

Mixed-effects models have many potential benefits, but also require more sophistication on the part of their users: at the present time, there does not yet exist a single standard method for extracting p values from such models, and all of the methods which do currently exist can yield highly anti-conservative results when used without proper precautions (Barr, Levy, Scheepers, & Tily, 2011). We fully expect that these issues will be resolved in time, and mixed-effect models will become a standard tool for EEG analysis; but for now, sticking to the more traditional method is also perfectly justifiable.

So, if we put aside method 3 for now, then we should ask whether method 2 has any advantage over method 1. The answer is no — in fact, though it may not be obvious at first, these methods are identical! The reason is that, just like baselining and filtering, calculating the average amplitude within a window is an operation that can be performed either before or after regression, with identical results (see Appendix); therefore we end up entering the exact same numbers into our ANOVA either way. So this method isn't just justified by the weight of ERP tradition; it's also the most straightforward way to perform repeated-measures regression analysis.

Certainly it will be possible to come up with more sophisticated approaches, but we have to be careful. Under almost any conditions, least-squares regression will produce unbiased estimates that, given sufficient data, will reliably converge to the correct

answer. But that's just for the βs; to get accurate p values, we need to make much more stringent assumptions: that the noise is normally distributed, and that each noise term is uncorrelated from all others, and that each noise term has the same variance. (Mixed-effect models then add even more assumptions about the form of the inter-subject variability.) If these are violated, then our p values may be radically off. The next chapter contains more discussion of how these issues apply to EEG analysis, along with a proposal for handling these issues to produce more powerful hypothesis tests; but unless you have a particular reasons to need this additional power, you won't go far wrong sticking with the traditional methods here.

4.4.4 Collinearity: Curse or blessing?

We mentioned above that we consider regression's ability to handle collinearity between predictors to be one of its great strengths. But we find that even those who use regression regularly are often nervous or confused about how collinearity effects their analyses, so it's worth spending some time examining the details here.

First, what is collinearity? It's often used to refer to two rather distinct phenomena. The first is when we have predictors which are perfectly redundant with each other. This can always be avoided by setting up our model properly; it's usually caused by some sort of error like, for example, accidentally entering the same predictor twice. It's easy to see why this would cause problems. Remember that by definition, least-squares fitting finds whatever combination of β values results in a prediction which is closest to the data. If we had only entered our predictor once, then least-squares might find that the best possible predictions are obtained by assigning it a β value of 1. When we enter our predictor twice, then we now have two β values, β_1 and β_2, and as long as $\beta_1 + \beta_2 = 1$, the model will achieve this best possible prediction. We can get this prediction by setting $\beta_1 = 1, \beta_2 = 0$, or $\beta_1 = 0, \beta_2 = 1$, or even $\beta_1 = 100, \beta_2 = -99$. That is, there is no best possible set of βs; instead, there is an infinite collection of equally good βs, and the least-squares fit is formally undefined. This is certainly a serious problem, and probably the source of some of the dire warnings about collinearity that have entered the experimental folklore, but we can always fix this kind of problem by fixing our model

specification.[8]

What comes up more often in practice is the second kind of collinearity, when some of our predictors are correlated, but not identical. The most important thing to know about using such predictors is that least-squares linear regression works just fine in their presence; in particular, it still produces efficient and unbiased estimates of the β coefficients, just like it does for orthogonal designs. For example, suppose you have two predictors, which are correlated with each other: "flurbiness", which does affect brain response, and "norbiness", which does not. Unfortunately, we don't know that — as far as we're concerned, either or both might affect the EEG. But if we enter both as predictors into a regression model and fit it with sufficient data, we will find out: our model will assign a significantly non-zero β coefficient to our predictor that codes flurbiness, but not to our predictor that codes for norbiness.

This is not to say that this weaker form of collinearity has no effect at all — it has several. Firstly, as we've already seen, adding collinear predictors into your model can alter the β coefficients corresponding to other correlated predictors. Some references view this as undesireable, but we don't see why. In our example above, if we had only entered norbiness as a predictor in our model, then of course linear regression would correctly point out that there was a relationship between norbiness and EEG amplitude. This is misleading, but inevitable — no statistical method can magically infer that we have asked the wrong question. The wonderful thing is that once we add flurbiness to our model, the effect attributed to norbiness goes away. This is only possible because the effect attributed to one predictor (i.e., norbiness) can vary depending on which other predictors (like flurbiness) are also entered into the model.

But this does raise the question of which predictors you should enter. The short answer is, you should enter all the ones that might matter. Why? Well, it's obvious why adding more predictors might be helpful — it might prevent us from publishing a paper declaring the important cognitive effects of norbiness, only to end up with egg on our face once someone notices that we forgot to include the flurbiness control. But we might

[8] Another option is to use various variants on least-squares which resolve these ambiguities by arbitrarily picking one of the equally good sets of coefficients, such as the Lasso (Tibshirani, 1996). This is fine if all you care about is making predictions and measuring overall goodness of fit, but if you want to be able to interpret the β coefficients themselves then picking them arbitrarily raises substantial obstacles. Therefore, we don't recommend the use of such regression methods for EEG analysis.

worry that there is a countervailing cost to adding more predictors; that — as per the general rule we've seen several times now — the additional flexibility they afford will make our estimates more noisy and unreliable.

To a certain extent this does happen, but the picture is more complicated than in previous cases. The key to understanding how collinearity affects the stability of our estimates is the *variance inflation factor*, or VIF. The VIF measures how the sampling variance of one estimated β coefficient is affected by the presence of collinearity between its predictor and the other predictors in the model, and is equal to $1/(1 - R^2)$ (see, e.g., chapter 13 of Fox, 2008). The R^2 here is not taken from the model we actually want to fit, but instead measures how much of the variance in this *predictor* can be explained by the *other* predictors in our model. So if we have a model with just two predictors, and the correlation between these two predictors is $r = 0.5$, then the VIF will be $1/(1 - 0.5^2) = 1.33$. In practical terms, what this number means is that if we know that we would need, say, 60 trials in order to get an acceptably accurate estimate of the ERP for a single predictor, then we need $60 \times 1.33 = 80$ trials to get acceptably accurate estimates of the ERP to these two predictors. (In general, of course, if your model has more than two predictors, you can't just look at the pairwise correlations — to compute the VIF you need to compute the full coefficient of determination R^2 for each predictor.)

The first conclusion we can draw from this is quite surprising: if we add additional *uncorrelated* predictors, then this has absolutely no effect on the noisiness of our estimates. Any predictor in an orthogonal design has a VIF of 1, which means that we can estimate a hundred parameters just as well as we can estimate one of them, from the same amount of data. (Of course, it may be quite difficult to find a hundred predictors which are all uncorrelated with each other!) Notice also that if we have two predictors which are correlated with each other but both are uncorrelated with a third, then we will need more data to estimate the effect of the first two predictors, but their presence in the model will not affect estimates for the third predictor.

The second conclusion is that whenever we do encounter a VIF > 1, then we will see it for multiple predictors at once — because correlation always goes two ways. This is closely related to the underlying cause of variance inflation. Recall how when we had

two identical predictors, the model knew that their β coefficients should sum to one, but could not distinguish between an option like $\beta_1 = 0, \beta_2 = 1$ and one like $\beta_1 = 100, \beta_2 = -99$. When our predictors are not identical, but merely similar, then our model can distinguish these cases — but the more similar (correlated) our predictors are, the more difficulty it has in doing so. This has the effect of increasing the sampling variance for all the involved βs, as measured by the VIF, and that extra variance manifests in a distinctive way. In orthogonal designs, when we don't have enough data, then we get noise our estimates, but the noise in one β is independent of the noise in another. In designs containing collinearity, when we don't have enough data, the noise may produce strange, structured behavior, in which different βs trade-off against each other in implausible ways. (One classic sign of this kind of behavior is when one β is large and positive, and another is large and negative.) This is fully expected within the theory of linear regression, accounted for in ordinary statistical tests, and not really different in principle from the ordinary noise we always expect to see when performing statistical estimation. But when it comes time to interpret such designs, it's important to keep an eye out for this behavior.

And the third conclusion we can draw by examining the formula for the VIF is that while correlation in your design does increase the amount of data you require, it may still be tractable to disentangle the effects of quite highly correlated predictors. If flurbiness and norbiness were correlated at the level of $r = 0.7$ (which, in our experience, would be on the high side for ordinary experimental materials), then we would need to collect double the usual amount of data to distinguish them. This would be a hassle but is probably not an insurmountable barrier. And in our own work, we have successfully used regression to disentangle reading-time effects with predictors whose correlation was $r = 0.98$ (Smith & Levy, 2010). This requires 25 times more data than one would otherwise need, but in a corpus study this is quite achievable, and one of the key advantages of regression is that it enables one to analyze relatively unstructured data like corpora. Collinearity can still become a problem as the number of predictors increases (each one adding just a bit to the R^2 for all the others), and it is often wise to compute the VIF while planning an experiment rather than waiting until the analysis stage, but in most cases these issues should be manageable.

Even so, there will of course be cases where the collinearity is too high, and we cannot gather enough data to achieve tight estimates of our β parameters. Would this, then, be a good time to toss out some predictors and simplify our model?

Certainly the temptation may be there, but it should be resisted. There are two main cases we can see. First, the problematic predictors may be useful as controls in your model, but not actually the target of your scientific study. For example, Hauk et al. (2009) wished to include bigram and trigram word frequencies as controls in their analysis. These factors are highly correlated with each other. To reduce this collinearity, they decided to use PCA to produce a single new predictor that was roughly the average of the two original frequency predictors, and entered that instead as their control. We have no reason to think that this choice had any substantive effect on their results, but it was completely unnecessary and potentially harmful; they would have done better to skip it and include the original predictors directly. It's true that the rERPs for highly-collinear control factors will be very noisy and should not be trusted, but if you aren't going to interpret them anyway, then this is no problem. As we've seen above, this untrustworthiness affects only the highly-collinear predictors. If your controls are strongly correlated with each other, but only weakly correlated with the predictors that are the actual targets of your analysis, then you can leave in all your control predictors without worrying that their instability will infect your other βs. And while these predictors will not produce useful estimates themselves, they will continue to provide a valuable control on those βs, reducing the chance of finding spurious results due to confounding.

The other problematic case is when one or more of the factors that are the target of your study are themselves too highly correlated with the other predictors in your model to achieve usable estimates. This is the case in which it will be most tempting to remove some other control factors "to increase power". But if the presence of those control factors means that you don't have enough data to estimate your effect, then, well, you don't have enough data to estimate your effect. Technically speaking, removing those control factors will indeed give you more statistical power, but it comes at the expense of being able to distinguish between those controls and the factor you care about. When you remove a control variable, what you're saying is that you know *a priori* that it has no effect; but if you know that, then why did you include it in the first

place? And if it does have an effect, then that power you are gaining may be the power to misattribute this effect to the factor you care about. Better to compute your VIFs before you start gathering data, and if they show that you will have a problem, figure out then how to add more trials or to reduce the collinearity in your stimuli.

So in general, there is little justification for ever removing predictors from your model. Note in particular that we do not make an exception for removing predictors which do not reach significance. As methodology papers will tell us over and over, significance gives us some reason to believe in an effect's existence, but insignificance does not tell us that an effect does not exist — and in particular, it does not tell us that this predictor is not providing an important control for other predictors in the model. It is also entirely possible that if we have two predictors which are correlated with each other, then either one will be significant when included individually, but when we include both then neither reach significance. This is easy to interpret: it means that we have enough data to see that at least one of these predictors has an effect, but not enough data to distinguish which of them is truly responsible. This is useful information. Stepwise regression techniques will generally pick one of these two predictors at random, which is why they aren't a very good tool for telling us which of a set of predictors are the ones that really matter — when they aren't sure, they make things up. This is fine if your goal is model simplicity, and a simple model can be very useful if you're trying to predict the future in a practical way. If a medical researcher can accurately predict cancer risk based on only 3 tests instead of 10, then that means that they can screen patients more economically, and it doesn't matter whether the tests they picked are the most direct reflections of the underlying process, or merely confounded with them in a useful way — these are the situations for which stepwise techniques were developed. But in scientific research, we are usually interested in whether we can be certain that some factor has an effect above and beyond any others which may be confounded with it. The best way to answer that question is to simply fit our full model with all our predictors in it, and then test and report the βs for whichever factors are of interest.

Finally, a few words on another response to collinearity that is sometimes used: that of orthogonalizing predictors before fitting the model (e.g. Hauk et al., 2006; Rousselet et al., 2009). In this approach, we use PCA or a similar technique to replace our

original predictors with a new set of predictors which together span the same space, but are themselves orthogonal to each other. This does produce a regression model where all the VIFs are 1, which might seem like an improvement. However, because we had to change our predictors to achieve this, this improvement is largely illusory. We can, indeed, get more stable estimates of our βs — but we can no longer interpret those βs, so what use is being able to estimate them precisely? And it is *only* the new βs that are estimated more precisely: the new model will make exactly the same predictions as the old model, and with exactly the same amount of noise. Any comparisons between different stimuli which we compute in the new model will come out just as they did in the old (the difference in how the stimuli are coded between the two models exactly cancels out the difference in βs between the two models). So again, we see little reason to recommend this technique; simply entering the predictors you care about unchanged seems like the best standard approach.

4.4.5 Which statistic to use? β, R^2, r, t, p, ...?

While our discussion has so far focused on β values as the output from fitting a regression model, there are a large number of ancillary statistics that can also be extracted, and that most regression software will compute by default. Within EEG analysis, R^2 has been taken as a measure of model fit, and differences in R^2 (or its close cousin the correlation coefficient r) have been used to infer differences in ERP activity between scalp locations (DeLong et al., 2005; Rousselet et al., 2008, 2010), across time (Rousselet et al., 2008, 2009, 2010), and between different participant populations (Rousselet et al., 2009, 2010). Regression models also allow one to calculate t and p statistics, and in analyses of PET/fMRI data these are often plotted instead of β coefficients. However, we have reservations about using these statistics in these ways with EEG.

The R^2 for a regression model is the percentage of variance which that model explains; that is, it is the ratio between the power (μV^2) of the estimated ERPs, and the total power measured at the scalp, which is the sum of the power due to ERP effects and the power due to background noise. As such, differences in R^2 are difficult to interpret: they might be caused by differences in the ERP to our stimuli, or they might be caused

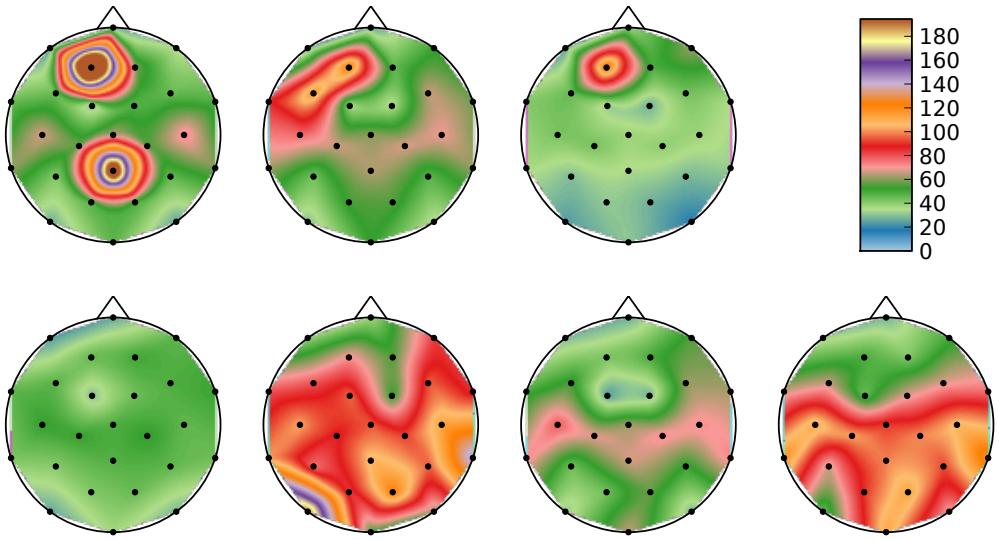

Figure 4.8: Average power (μV^2) at each electrode from a 500 ms baseline interval in the previously mentioned go/no-go task, for each of 7 participants. This is an estimate of the residual background noise which forms a portion of the denominator when calculating R^2 or r values. In each case, noise power varies by at least a factor of 2 over the scalp (and usually by much more), which means that R^2 will vary by almost as much, even in the absence of any real differences in effect topography. Different participants have substantially different different spatial distributions, and substantially different total power, which means that R^2 maps are also incomparable between participants.

by differences in the background noise. The simple correlation coefficient r is derived from a similar ratio, and has the same problem. Intuitively, there is good reason to worry that background noise might vary substantially over the scalp or between participants, because we know that there exist powerful noise sources with specific localization, such as alpha or muscle artifact. To check this more rigorously, we extracted 500 ms of pre-stimulus activity from the 7 participants used to demonstrate non-linear regression (Figures 4.6 and 4.7), and for each we computed the average power present in each scalp electrode. The results are shown in Figure 4.8. These are reasonably clean and artifact-free data typical of that analyzed in ERP experiments, and represent an estimate of baseline brain activity in the absence of any stimulus. They contain huge amounts of variation. Not only does power vary greatly over the scalp within each individual, and overall vary greatly between individuals, but the scalp distribution varies greatly between individuals as well.

This presents a substantial potential confound for our ability to interpret R^2 or r, since any differences we see may be due to irrelevant differences in background noise rather than differences in event-related brain activity. Such differences will not necessarily arise in all cases (for instance, compare Figure 1 of DeLong et al., 2005 to DeLong, Urbach, Groppe, & Kutas, 2011, Figure 2), and when they do arise, they may themselves be of scientific interest. But if our goal is to understand how EEG power varies between conditions or participant groups, then it makes more sense to perform one computation to estimate the ERP effects alone (i.e., the βs from an rERP analysis), and a second to estimate changes in non-time-locked noise (e.g., by comparing baseline noise between participants, or performing an event-related spectral perturbation (ERSP) analysis, Makeig, 1993). The ratio between these quantities is of interest mostly because of its ability to tell us how well our measurement tools are working (e.g., measuring signal-to-noise ratio); it isn't a reliable tool for answering questions about the brain's activity itself.

This issue of background noise also makes R^2 and r less meaningful for EEG analysis than in some other situations. With EEG, we know a priori that some substantial part of the total variance is *not* explicable by *any* model of how the brain processes our stimuli, simply because that variance arises from other processes which are indifferent

to and independent of our stimuli. Therefore even good very good models of event-related brain processes will still produce rather low R^2 values, and this is in no way a strike against them.

Finally, a brief comment on the occasional practice of dividing trials up into a number of bins (e.g., 10 bins by deciles), estimating ERP amplitude within each bin, entering these amplitudes into a regression analysis, and then computing R^2 or r for this final regression (e.g., DeLong et al., 2005). The practice of averaging before regression is not particularly useful — as we've seen, the averaging itself is a kind of non-linear regression and useful in its own right, but if our end goal is to perform a linear regression then binning first is somewhat unprincipled and unlikely to affect the outcome much. But it will have a very large effect on the computed R^2. Consider that if we go to the extreme of using only two bins, then we will have just two data points to enter into the regression, and it is always possible to draw a line through two points. Therefore a linear model can explain all of the variance; we will always find that $R^2 = 1$. As we add more bins, the number of data points in each bin goes down, and so the average over each bin becomes noiser, and the R^2 drops. Eventually, we're left with individual data points, and the rather low R^2 that we get from the rERP models discussed here. In between these limits, we can achieve any R^2 we like simply by choosing how many bins to use. So while a scatter-plot of the individual points can be highly informative, the actual R^2 values derived from such regressions are not meaningful.

We aren't saying that R^2 can never be useful — for instance, it's a perfectly good statistic for doing model comparison, and there are good reasons to want to know what our signal-to-noise ratio looks like, even if it isn't the primary goal of research. We are merely arguing that one must consider whether it is the appropriate tool for the situation at hand.

Turning to t and p values, we face a similar set of issues. These statistics can certainly be useful for determining whether an effect exists; that is, literally, what they're designed for. But as general tools for understanding the brain's activity, they have the same limitation as R^2, in that they are affected just as much by the strength of the background noise as they are by the signal. In addition to this, they are also affected by our sample size, meaning that unlike even R^2, t and p values are not comparable across

studies. In PET and fMRI studies, it is common to plot t or p scores directly. These statistics are good at determining the existence of effects, so if the goal of analysis is to determine whether there is or is not an effect in each voxel individually, then examining them makes perfect sense. But in EEG analysis, we know *a priori* that if there is any effect at all, then volume conduction will ensure that it is present to some extent in most of our electrodes (Lutzenberger, Elbert, & Rockstroh, 1987).

So in EEG, unlike PET/fMRI, there are generally two questions we are interested in. First, is there an effect present at all? At this stage, whether an effect is present at one electrode versus another is almost never of interest, though we may still perform a significance test at each electrode for computational convenience, so that if we find significance at at least one electrode we can conclude that some effect is in fact present. Once we know that there's an effect, we ask our second question: what is its distribution over the scalp? At this stage, again, checking for significance at individual electrodes is not particularly useful; it may give some coarse picture of where the effect is strong, but we're just as interested in knowing where it is weak, and, more importantly, in exactly *how* strong or weak it is at each point. There's a specific way of defining effect strength across the scalp that has good theoretical justification. According to the mathematics of volume conduction, a certain amount of the electrical activity generated by each point in the brain will project linearly to each point on the scalp. That is, there is some fixed multiplicative scaling factor that determines how strongly any given activity at a particular source will appear at each electrode. If we could determine these scaling factors, then they would tell us everything that we could possibly learn about the source's location using EEG. So finding these factors is useful both for intuitive reasoning about different components and for more formal localization procedures.

In practice, we can't quite hope to figure out the actual scaling factors, since we do not know how strong the source activity is; this makes it impossible to distinguish between a strong source with small scaling factors, or a weak source with strong scaling factors. But what we can do is estimate these scaling factors up to a multiplicative constant. To do this, we need some statistic that increases by a factor of s whenever the event-related portion of the EEG signal increases by a factor of s; then we could estimate this statistic for each electrode. Conveniently, slope rERPs are just such a

statistic, and the β values from rERP regression are therefore estimates of the forward projection constants implied by volume conduction theory. This is why we suggest that the βs from linear rERPs are the most appropriate tool for most everyday ERP analysis needs.

If we're doing non-linear regression, then this becomes more complicated, since we no longer have a single β describing effect slope at each electrode. However, non-linearities add another interesting potential source of information to our understanding of effect distributions. Suppose that there were only one source present whose activity varied with our stimulus property. Whatever that source's response curve looks like, it must be projected unchanged (except for uniform scaling) to all of our scalp electrodes. That is, you can't get two different non-linearities from a single source. If we see different non-linearities at different points on the scalp, then, this is strong evidence that we are observing are multiple sources with different sensitivities to our stimuli. In fact, since the non-linearities observed at the scalp must be weighted sums of the non-linearities present at the source, the way our estimated non-linearities vary across the scalp contains some kind of information about the response and distribution of the underlying sources. An open question is whether it would be possible to decode this information somehow to learn about the individual sources, possibly via a technique such as independent component analysis (Makeig, Bell, Jung, & Sejnowski, 1996).

4.5 Discussion

We've only scratched the surface of the regression literature here; there's extensive work on such topics as mixed-effects regression, robust regression, and so forth, any of which may prove valuable for work on EEG. We hope that another benefit of our framework is that it will enable such techniques to be adapted when appropriate. Another potential area for future work is the application of these techniques to areas besides ERP analysis. For instance, we expect that most of these techniques could be applied unchanged to MEG analysis, and even within EEG, there are a variety of other analysis strategies which might benefit from these tools. For example, independent component analysis (Makeig et al., 1996) begins by decomposing single-trial scalp potential into

traces of component activation. These activations are then analyzed in various ways, for instance by dividing them into bins and averaging, just as with standard ERPs; they could just as well be entered into a regression model as described here, and with similar benefits. Likewise, in time-frequency analysis a standard measure is the event-related spectral perturbation (ERSP; Makeig, 1993), which is calculated by transforming single trial data to the time-frequency domain, discarding phase information, and then averaging. This averaging too could be replaced by regression.

But in this paper, we've instead focused on laying out the basic principles and considerations for extending classic ERP analysis to include least-squares regression techniques. These techniques are not themselves new; they been used extensively for centuries, and standard ERP averaging is itself an example of this methodology. For each of the individual situations we consider, either the method we propose has been been previously suggested independently, or a close variant of it has. But what hasn't been appreciated is that regression provides a single framework that encompasses each of these methods, unifies them conceptually, and lets us choose in a principled way how to apply them to each situation that arises in practice. Furthermore, bringing them into the same framework lets us mix-and-match the capabilities of each technique to adapt to our specific situation, and these adaptations are derived automatically by the least-squares fitting process; all we have to do is describe our design by defining a set of predictors.

While the primary value of these models is for analyzing complex data sets which contain confounding and which in practice may arise from observational/corpus-type work rather than from designed experiments, we are not advocating that anyone give up orthogonal designs for ones containing collinearity, or give up designed experiments for corpus studies. As we learned from our examination of the variance inflation factor, orthogonal designs extract more information from a given set of data than designs containing collinearity (though an orthogonal design which contains continuous predictors may tell you even more than one which uses dichotimization to produce categorical predictors, Cohen, 1983). A proper randomized experiment can tell you things about causality that a corpus study cannot. But this only works when you can randomly assign factors to conditions while holding everything else constant, and there arise situations

where this is prohibitively difficult or undesireable for other reasons. In our field of language research, there are so many potentially relevant factors, all correlated with each other, that it is standard to pick some subset of these factors to control experimentally, and even that is only done approximately. Furthermore, even if it were easy to control these factors exactly, doing so would force us to choose very unusual words, which might distort our results in other ways (Blalock, 1967). And while only a randomized experiment lets us infer causality, only large studies using naturalistic stimuli let us examine how a wide variety of factors might interact in real-world situations (since the world is, of course, not orthogonal).

Therefore, we see these kinds of research as complementary, and the advantage of a flexible analysis technique is that it lets us use whatever design is most appropriate for the question at hand, rather than redefining our questions to match our tools.

4.6 Acknowledgements

Chapters 4 and 5, in full, are currently being prepared for submission for publication of the material. The dissertation author was the primary investigator and author of this material.

4.A Derivations for baselining, filtering, windowing

Here we give the mathematical derivations which show that many standard operations — including baselining, filtering, and taking averages over latency windows — can be performed either on individual epochs before using least-squares regression to estimate rERPs, or else on the resulting rERPs themselves, and the exact same result will be obtained either way.

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

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

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 written as

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

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:

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

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.

The key fact about the operations we consider here is that 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 B_F:

$$B_F = (X^T X)^{-1} X^T (YF).$$

Alternatively, we could first compute the matrix of rERP values, B, and then apply our

transformation to these β values directly, which in matrix notation would be written BF.

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 $B_F = BF$. But this is now easy to see, simply by associativity of matrix multiplication:

$$B_F = (X^T X)^{-1} X^T (YF) = \big((X^T X)^{-1} X^T Y\big) F = BF.$$

References

Amsel, B. D. (2011). Tracking real-time neural activation of conceptual knowledge using single-trial event-related potentials. *Neuropsychologia*, *49*(5), 970–983.

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*(4), 390–412.

Barr, D. J., Levy, R., Scheepers, C., & Tily, H. J. (2011). *Random effects structure in mixed-effects models: Keep it maximal.* Manuscript submitted for publication.

Blalock, H. M. (1967). Causal inferences in natural experiments: Some complications in matching designs. *Sociometry*, *30*(3), 300–315.

Cohen, J. (1983). The cost of dichotomization. *Applied Psychological Measurement*, *7*(3), 249–253.

Dambacher, M., Kliegl, R., Hofmann, M., & Jacobs, A. M. (2006). Frequency and predictability effects on event-related potentials during reading. *Brain Research*, *1084*(1), 89–103.

DeLong, K. A., Urbach, T. P., Groppe, D. M., & Kutas, M. (2011, September). Overlapping dual ERP responses to low cloze probability sentence continuations. *Psychophysiology*, *48*(9), 1203–1207. Retrieved from http://onlinelibrary.wiley.com/doi/10.1111/j.1469-8986.2011.01199.x/abstract

DeLong, K. A., Urbach, T. P., & Kutas, M. (2005). Probabilistic word pre-activation during language comprehension inferred from electrical brain activity. *Nature Neuroscience*, *8*, 1117–1121.

Fox, J. (2008). *Applied regression analysis and generalized linear models* (2nd ed.). Thousand Oaks, California: SAGE.

Hastie, T. J., & Tibshirani, R. J. (1990). *Generalized additive models*. New York: Chapman and Hall.

Hauk, O., Davis, M. H., Ford, M., Pulvermuller, F., & Marslen-Wilson, W. D. (2006). The time course of visual word recognition as revealed by linear regression analysis of ERP data. *NeuroImage*, *30*, 1383–1400.

Hauk, O., Pulvermller, F., Ford, M., Marslen-Wilson, W. D., & Davis, M. H. (2009). Can I have a quick word? early electrophysiological manifestations of psycholinguistic processes revealed by event-related regression analysis of the EEG. *Biological Psychology*, *80*(1), 64–74.

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.

Howes, D. H., & Solomon, R. L. (1951). Visual duration threshold as a function of word-probability. *Journal of Experimental Psychology*, *41*(6), 401–410.

IBM. (2010). *Spss statistics* [Computer software].

Jung, T., 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*(3), 166–185.

King, J. W., & Kutas, M. (1998). Neural plasticity in the dynamics of human visual word recognition. *Neuroscience Letters*, *244*(2), 61–64.

Kutas, M., & Hillyard, S. A. (1984). Brain potentials reflect word expectancy and semantic association during reading. *Nature*, *307*, 161–163.

Laszlo, S., & Federmeier, K. D. (2011). The n400 as a snapshot of interactive pro-

cessing: Evidence from regression analyses of orthographic neighbor and lexical associate effects. *Psychophysiology*, *48*(2), 176–186.

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*(3), 358–366.

Luck, S. J. (2005). *An introduction to the event-related potential technique*. Cambridge, MA: MIT Press.

Lutzenberger, W., Elbert, T., & Rockstroh, B. (1987). A brief tutorial on the implications of volume conduction for the interpretation of the EEG. *Journal of Psychophysiology*, *1*, 81–89.

Makeig, S. (1993). Auditory event-related dynamics of the EEG spectrum and effects of exposure to tones. *Electroencephalography and Clinical Neurophysiology*, *86*, 283–293.

Makeig, S., Bell, A. J., Jung, T.-P., & Sejnowski, T. J. (1996). Independent component analysis of electroencephalographic data. In *Advances in neural information processing systems 8.* Cambridge, MA: MIT Press.

McCusker, L. M. (1977, April). *Some determinants of word recognition: Frequency.* Paper presented at the 24th Annual Convention of the Southwestern Psychological Association, Fort Worth, TX.

Pernet, C. R., Chauveau, N., Gaspar, C., & Rousselet, G. A. (2011). LIMO EEG: a toolbox for hierarchical LInear MOdeling of ElectroEncephaloGraphic data. *Computational Intelligence and Neuroscience*, *2011*.

R Development Core Team. (2010). R: A language and environment for statistical computing [Computer software and manual]. Vienna, Austria. Retrieved from http://www.R-project.org (ISBN 3-900051-07-0)

Rousselet, G. A., Gaspar, C. M., Pernet, C. R., Husk, J. S., Bennett, P. J., & Sekuler, A. B. (2010). Healthy aging delays scalp EEG sensitivity to noise in a face

discrimination task. *Frontiers in Psychology*.

Rousselet, G. A., Husk, J. S., Pernet, C. R., Gaspar, C. M., Bennett, P. J., & Sekuler, A. B. (2009). Age-related delay in information accrual for faces: Evidence from a parametric, single-trial EEG approach. *BMC Neuroscience*, *10*(1), 114.

Rousselet, G. A., Pernet, C. R., Bennett, P. J., & Sekuler, A. B. (2008). Parametric study of EEG sensitivity to phase noise during face processing. *BMC Neuroscience*, *9*(1), 98.

SAS Institute Inc. (2010). *SAS/STAT 9.22 user's guide* [Computer software and manual]. Cary, NC: SAS Institute Inc.

Smith, N. J., & Levy, R. (2010). Fixation durations in first-pass reading reflect uncertainty about word identity. In S. Ohlsson & R. Catrambone (Eds.), *Proceedings of the thirty-second annual meeting of the Cognitive Science Society* (pp. 1313–1318). Austin, TX: Cognitive Science Society.

Squires, N. K., Squires, K. C., & Hillyard, S. A. (1975). Two varieties of long-latency positive waves evoked by unpredictable auditory stimuli in man. *Electroencephalography and Clinical Neurophysiology*, *38*(4), 387–401.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B (Methodological)*, *58*(1), 267–288.

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.

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 and New York: Continuum.

Urbach, T. P., & Kutas, M. (2008). Cognitive aging: Evidence against a single factor. *Psychophysiology*, *45*(S1), S113.

Wahba, G. (1990). *Spline models for observational data*. Philadelphia, PA: SIAM.

Wood, S. N. (2004). Stable and efficient multiple smoothing parameter estimation for generalized additive models. *Journal of the American Statistical Association*, *99*, 673–686.

Wood, S. N. (2006). *Generalized additive models: An introduction with R*. Boca Raton: Chapman and Hall/CRC.

Chapter 5

Regression-based ERP analysis: II. Continuous-time techniques

In the previous chapter, we explained the fundamentals of using regression models to estimate ERPs (or *rERPs*), which allowed us to analyze designs containing continuous and partially founded factors. Here, we extend this approach further, enabling it to properly estimate the underlying form of ERPs which overlap in time, and to explicitly handle the temporal correlations present in the EEG background noise.

This latter change has the potential to produce a more efficient statistical estimator of the rERP, i.e., one which produces better estimates of rERPs from fewer data, and also enable the use of parametric statistical tests. One the advantages of rERPs is that they allow us to analyze rich designs that contain many factors of interest — but the use of such designs creates a difficulty for statistical testing. When there are many interesting hypotheses to test, it isn't enough to check for $p < 0.05$ — we must use some form of multiple comparison correction (Groppe, Urbach, & Kutas, in press), which require much stricter significance thresholds. More powerful statistics, therefore, may be a practical necessity for enabling the analysis of large, rich data sets such as those that arise in language research. However, the complex structure of the EEG signal means that using these tests successfully may prove challenging in practice; here we explore the theory underlying them, and their prospects for future use.

5.1 One regression for all latencies

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 basic idea that enables the extensions described in this chapter is to combine these many regression models into a single, giant model. We do this by using the same trick which we used in the 'dummy coding' example in the previous chapter. Recall that there, we started by fitting model containing only an intercept term ($x_i = 1$), and found that the resulting coefficient β was equal to the average of the data we entered into the model. This let us compute ERPs to each of k conditions by taking the subset of our trials that occurred in that condition, and fitting our model on that subset. The dummy coding idea was to instead fit a single model on all of the data, where we defined new predictors $x_{1i}, x_{2i}, \ldots, x_{ki}$, such that $x_{ji} = x_i = 1$ if the data point y_i is one of the data points which was included in the jth subset of the data, and zero otherwise. Because each of the resulting predictors was non-zero on different, non-overlapping subsets of the data, this produced exactly the same fitted coefficients $\beta_1, \ldots, \beta_k$ as we would have gotten by fitting the original k separate models.

But in all of the above, we were still working on that subset of our data which was measured at a single latency relative to an event, and then repeating our model fitting at each latency. Now, assume we have some model we'd like to fit, defined by predictors $x_{1i}, \ldots, x_{ki}$ chosen as per the discussion in the previous chapter, and we'd now like to combine *all* of our data into a single model. To do this, we define a new set of $k \times m$ predictors, where m is the number of distinct measured latencies that fall within our chosen epoch. These predictors are each defined to take on the same value as one of the old predictors for data which was measured at one particular latency, and zero otherwise. For instance, with a 250 Hz sampling rate, we might have (see Figure 5.1):

$$x_{1 \text{ at 0 ms}, i} = \begin{cases} x_{1i} & \text{if } y_i \text{ was measured 0 ms after an event} \\ 0 & \text{otherwise} \end{cases}$$

$$x_{1 \text{ at 4 ms}, i} = \begin{cases} x_{1i} & \text{if } y_i \text{ was measured 4 ms after an event} \\ 0 & \text{otherwise} \end{cases}$$

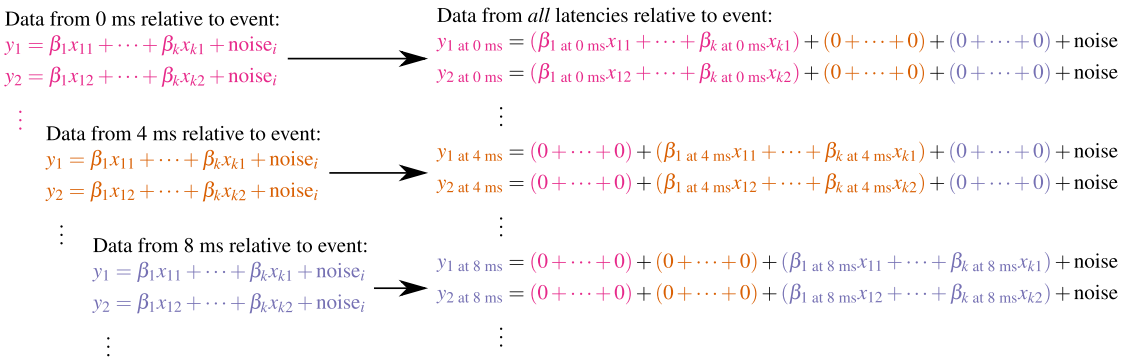

Figure 5.1: Going from one regression model for each latency (previous chapter) to a combined regression model for all latencies (this chapter). On the left, we see three of the models that in the previous chapter we would have fit to different subsets of our data. (The particular latencies shown reflect 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 it previously applied to, and zero elsewhere.

$$\vdots$$

$$x_{2 \text{ at 0 ms}, i} = \begin{cases} x_{2i} & \text{if } y_i \text{ was measured 0 ms after an event} \\ 0 & \text{otherwise} \end{cases}$$

$$x_{2 \text{ at 4 ms}, i} = \begin{cases} x_{2i} & \text{if } y_i \text{ was measured 4 ms after an event} \\ 0 & \text{otherwise} \end{cases}$$

$$\vdots$$

One way to think about this is that we are taking latency as being a categorical factor, and then defining a new set of predictors which represent the interaction of latency with each of our original predictors.

The same reasoning as before now applies: because each β only affects the model's predictions for the subset of the data which occurred at a single latency, the estimated β coefficients we derive by fitting this model to all of our data will match exactly to the β coefficients derived from fitting our original collection of models to subsets of our data. (For instance, $\beta_{1 \text{ at 0 ms}}$ will match β_1 as fit to the 0 ms data, while $\beta_{1 \text{ at 4 ms}}$ will match β_1 fit on the 4 ms data.) On the one hand, this makes it clear that this

is a statistically acceptable thing to do. On the other, it may raise the question of why we should bother doing this in the first place. The answer, of course, is that it allows several new extensions. When these extensions aren't used, then the rERPs in this chapter are identical to the ones in the previous chapter.

5.1.1 A note on baselining and filtering

Before we discuss these extensions and their advantages, though, a warning is in order: while most results described in the previous chapter still hold true for the combined models we discuss here, one does not. Previously, linear transformations on individual epochs such as baselining or filtering could be applied either on the raw data, or on the resulting rERP, with identical results. When using the extensions described here, the proof given in the previous chapter's Appendix does *not* go through, and it can make a difference when you filter or baseline. In this chapter, we will perform baselining after rERP estimation in those cases where it makes sense to do so.

5.2 Overlap correction

The ERP to any given event may be quite long, persisting for a second or more. When events occur in quick succession, then, their ERPs necessarily overlap. This makes it difficult to determine the actual ERP to any individual event. One approach to handling this problem is to simply not present stimuli in quick succession, but this is not always a viable option, both because it can greatly reduce the amount of data that can be gathered in a practical amount of time, and because often it is necessary to present stimuli more quickly than once per second in order to elicit the desired brain response. For example: (a) reading at one word per second is quite unnatural, (b) selective attention tasks are performed better with faster rates of presentation (Woldorff, 1993), (c) response-locked potentials in a speeded reaction time task necessarily occur soon after stimulus presentation, and (d) we may wish to use presentation rate itself as an independent variable (e.g. Kutas, Lindamood, & Hillyard, 1984).

In some cases the resulting overlap can still be tolerated, but in others it may be very important to correct for. For instance, we may wish present two stimuli in

quick succession, and then investigate how the processing of the second stimulus varies depending on what stimulus was presented first (e.g., there may be priming effects). But even if we do find such differences in the averaged ERP, it may be difficult to know whether these are caused by differences in processing of the second stimulus, versus lingering differences in the brain response to the various first stimuli themselves (Woldorff, 1993). Another example where overlap can cause critical difficulties is if there are sequential dependencies in your stimulus sequence, in which case a naive analysis may detect spurious effects. This was the problem which initially motivated this investigation: a straightforward rERP analysis of sentences presented in RSVP was finding significant effects of word frequency at 0 ms, which is clearly impossible. So what went wrong? In natural language, the frequencies of adjacent words are correlated, so the overlapping effect of the previous word was introducing an unexpected, uncontrolled confound.

Therefore it would be convenient to have a analytical technique that could take EEG recorded to events presented in near temporal proximity, and from it estimate what the ERPs to individual events were, correcting for any overlap. This necessarily requires some assumption about what happens when two ERPs overlap, and here we make the simplest assumption that they sum linearly. Aside from providing computational convenience, this assumption seems physiologically plausible in some cases (e.g., if the overlapping components arise from distinct and non-interacting cortical sources). And even when it isn't plausible, it still provides a useful baseline: we can use our technique to compute what ERPs would have to be summed to best reproduce some observed effect, and if these do not match the ERPs observed in isolated presentation, or if there are systematic patterns in the data that are not captured by this simple linear summation, then this is evidence that there must be some more complex interaction between the brain processes triggered by our adjacent events.

But how can we estimate these ERPs? The combined regression model actually makes this easy. Suppose we make a measurement of scalp potential y_i at some particular time, and suppose that we know that some event happened 100 ms before our measurement, and that another event happened 300 ms before our measurement. Then according to our linearity assumption, y_i is the sum of the ERP to the first event at 100

ms latency, the ERP to the second event at 300 ms latency, and background noise:

$$y_i = \text{First event ERP}_{\text{at 100 ms}} + \text{Second event ERP}_{\text{at 300 ms}} + \text{noise}_i$$

According to the rERP setup, these ERPs can each be written as the sum of the some β coefficients and event-specific predictor values:

$$\begin{aligned} y_i = {} & \beta_{\text{1 at 100 ms, } i}\ x_{\text{1 at 100 ms for first event, } i} + \cdots + \beta_{\text{k at 100 ms, } i}\ x_{\text{k at 100 ms for first event, } i} \\ & + \beta_{\text{1 at 300 ms, } i}\ x_{\text{1 at 300 ms for second event, } i} + \cdots + \beta_{\text{k at 300 ms, } i}\ x_{\text{k at 300 ms for second event, } i} \\ & + \text{noise}_i. \end{aligned}$$

In the previous section, we observed that only one set of predictors would be non-zero for any given y_i. That assumed a standard ERP experimental design, in which each epoch is treated as an independent, non-overlapping measurement. If overlapping is present, then that means that multiple ERPs are contributing to individual y_is, and to account for this, we can in the combined model simply set the appropriate sets of predictors to be non-zero simultaneously. Then the least-squares fitting process will, as usual, automatically do the grunt work of pulling apart the variance in the data and attributing it to the appropriate predictors. (An equivalent procedure has been proposed for estimating hemodynamic response functions in the context of event-related fMRI: Burock, Buckner, Woldorff, Rosen, & Dale, 1998.)

It may be useful to compare this to another approach for handling confounding due to sequential correlations in our stimuli, which is a problem that also arises in, for example, reading time studies. There, we sometimes include as predictors both properties of the current word and properties of the previous one or more words. We could do this in our rERP analysis as well. For instance, suppose that we present a word every 300 ms exactly, and we do an rERP analysis that includes both current word frequency and previous word frequency, without the kind of overlap correction described here. This will work to some extent, but has a somewhat disturbing feature. The estimated effect of the current word's frequency at 600 ms may be totally unlike the estimated effect of the previous word's frequency at 300 ms — which is 600 ms after that word was presented. It makes no sense that the same measurement of the scalp potential

should be entered at two different times into the same analysis, and be explained as arising due to two completely different ERP effects triggered by a single event. The solution is to enforce a constraint on our model-fitting process that the previous-word effect at 300 ms should be the same as the current word effect at 600 ms — then that constraint produces exactly the procedure described here.

As usual, performing regression requires that the information to distinguish between our predictors be present in the data. If we have only one type of stimulus, and we present it every 100 ms exactly, then our data will simply not be able to tell us whether some deflection occurs at 100 ms latency or 200 ms latency, because these are always observed together. In the regression model, this manifests as our old friend collinearity — because of the regularity of our presentation schedule, $x_{\text{j at 100 ms, } i}$ and $x_{\text{j at 200 ms, } i}$ will be non-zero at exactly the same times, and because we have only one type of stimulus, when they are non-zero they will have the same values. Therefore they will be perfectly collinear. To get accurate estimates, we need to break this collinearity somehow, and there are two options. One is to jitter our presentation (which will alter the pattern of zeros). The other is to vary our stimuli in such a way that the values of x_{ji} for adjacent stimuli are relatively uncorrelated. Either technique can work on its own; when used together, they complement each other. Note that in theory we do *not* need to ever observe any non-overlapping ERPs — we just need to make sure that the overlapping is not perfectly consistent. In fact, you can never even achieve collinearity in practice, because the first and last stimuli will necessarily break the pattern. But if this is the only thing breaking collinearity, then you effectively have only one or two epochs worth of data, which is unlikely to produce satisfactory results.

Most models will include an intercept term, which will always, by definition, be perfectly correlated between adjacent items. If the goal of analysis is to understand the effects of other predictors, then it may not be important to get an accurate estimate of the intercept rERP; recall from our previous discussion of collinearity that even if some predictors are highly correlated with each other, this does not necessarily affect our ability to estimate the effects of other predictors in the model. If, however, an accurate estimate of the overlap-corrected intercept rERP is important, then you will have to rely on jitter to make this possible.

The resulting β values and any predicted ERPs made from them can be plotted, filtered, and entered into statistical analysis just like any other rERP or ERP estimate.

5.2.1 Simulation

To demonstrate this method, we simulated an experiment in which a continuous train of stimuli are presented, with each SOA randomly selected to fall between t_{min} and t_{max} ms, in increments of 16.7 ms (which simulates the possible SOAs for visual presention using a monitor with a fixed 60 Hz refresh rate). We time-locked a known ERP waveform to each stimulus, and then summed these time-shifted ERPs with simulated EEG background noise. (This background noise was generated by taking the Fourier transform of a stretch of real EEG data, randomly shuffling the phase components, and then transforming back to the time domain.) The question is, how accurately can we reconstruct our known ERP waveform given only this noisy data?

In Simulation 1, we set $t_{\text{min}} = 200$ ms, $t_{\text{max}} = 400$ ms, and each stimulus was assumed to produce an identical ERP. As discussed above, this is something of a worst case for our technique, since in real experiments there will usually be some variation between stimuli, which our method is able to exploit. Figure 5.2 shows the true ERP that we embedded in the data, and our attempts to reconstruct it from varying amounts of data, using both the conventional time-locked averaging and our overlap-corrected rERP technique. (In both cases, we applied baseline correction to the final curves before plotting.) Averaging converges to a stable answer on much fewer data — but it is the wrong answer, as components from previous and successive trials appear overlaid on the estimated ERP. The overlap-corrected rERP requires more data, but accurately estimates the correct ERP waveform — and since the large degree of overlap means that these trials come quite quickly, even the bottom row of our figure is based on only ten minutes worth of simulated EEG.

In Simulation 2, we set $t_{\text{min}} = t_{\text{max}} = 300$ ms, i.e., we used no jitter. Instead, we simulated a continuous factor with a linear effect on the ERP. This factor was randomly assigned a value between 0 and 2 independently for each simulated stimulus, and the simulated ERP for each stimulus was multiplied by this factor before being summed with the overlapping ERPs and background noise. As discussed above, this is sufficient

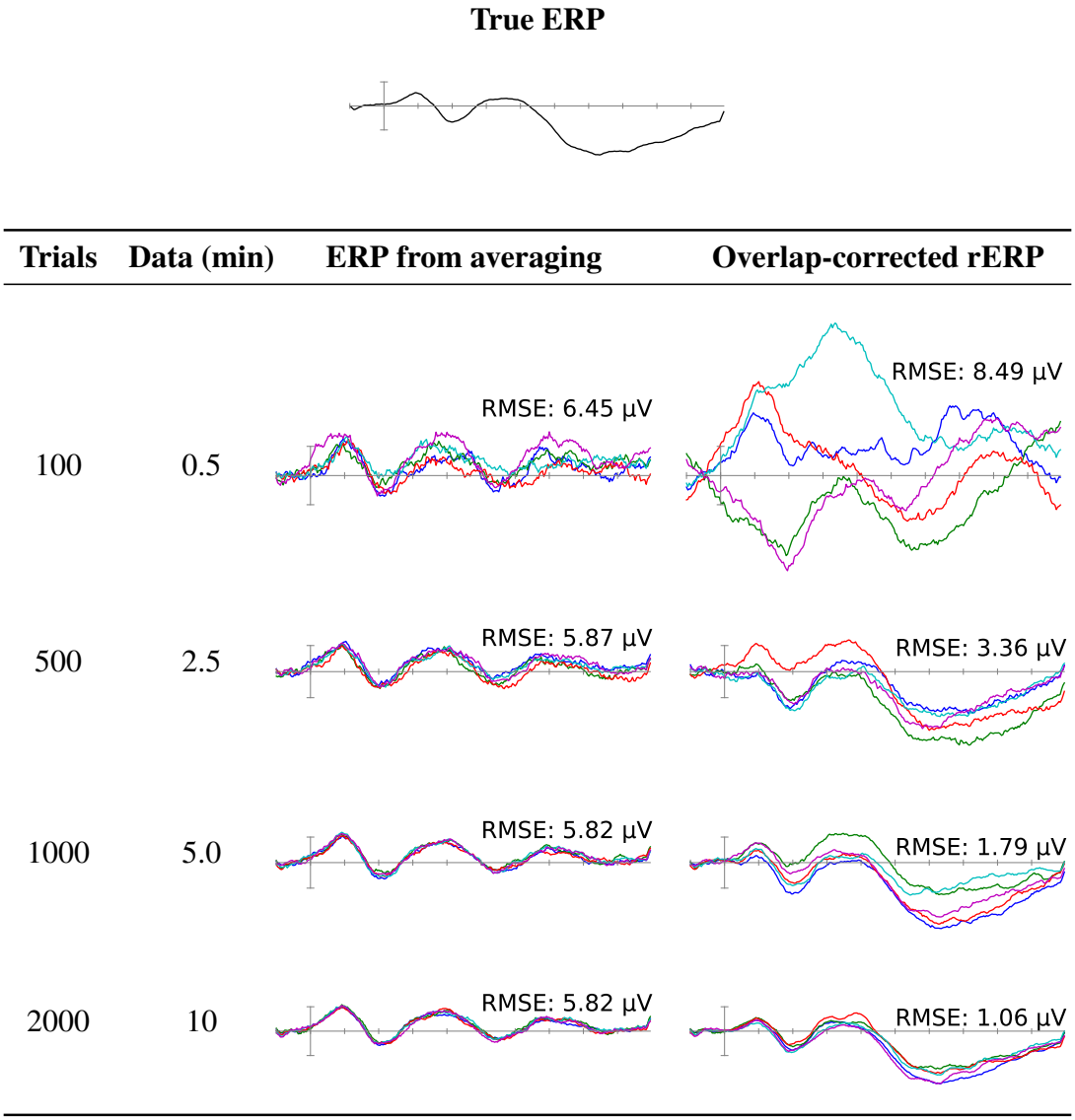

Figure 5.2: Simulation 1: Conventional averaged ERPs, and overlap-corrected rERPs, estimated from various amounts of simulated data with SOAs randomly varying in the 200–400 ms range. Each analysis was run on 100 simulated data sets. 5 of the resulting ERPs/rERPs are plotted here to help visualize variability between runs; in an actual experiment, you would just see one of these waveforms selected at random. All 100 were used to calculate the root-mean-squared-error (RMSE). This is an estimate of the standard error; a smaller RMSE means that the estimated values are closer to the true value.

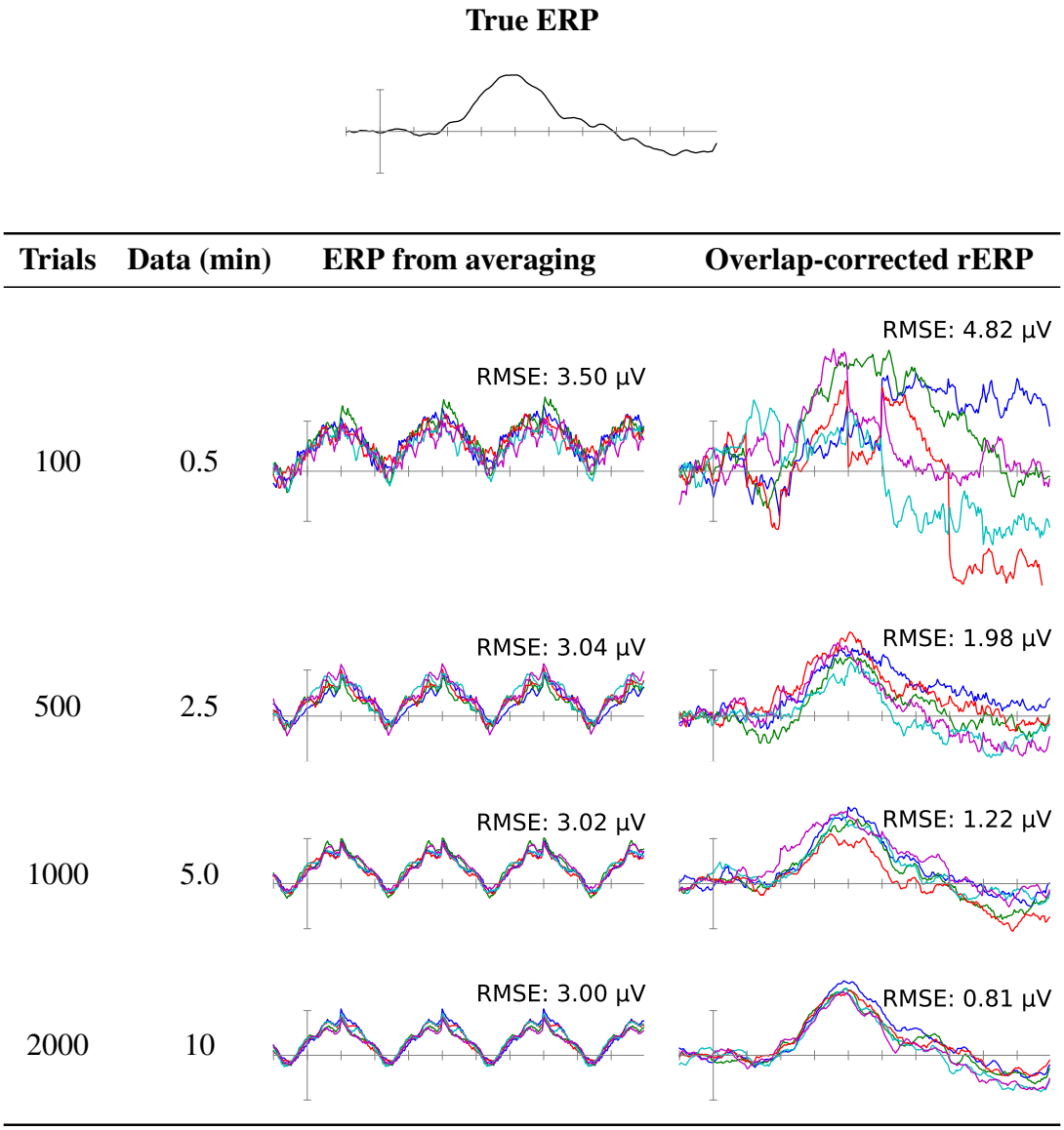

Figure 5.3: Simulation 2: Conventional averaged ERPs, and overlap-corrected rERPs, estimated from various amounts of simulated data, with a fixed SOA of 300 ms and a continuous factor which acts linearly on the ERP. (It is not standard to use simple averaging to estimate an ERP in this case, but since our factor had an average value of 1, simple averaging would accurately reconstruct the true ERP if not for the presence of overlap.) As before, we plot analyses of 5 simulated data sets, while 100 simulated ERP/rERP estimates were used to calculate the RMSE (standard error).

to break collinearity, so it should be possible to accurately reconstruct the true ERP from the overlapped data even without any jitter. As we see from Figure 5.3, this theory correctly describes (simulated) practice.

In Simulation 3, we combined the jitter from Simulation 1 ($t_{\text{min}} = 200$ ms, $t_{\text{max}} = 400$ ms) with the stimuli from Simulation 2 (a continuous factor varying randomly between 0 and 2). Figure 5.4 shows the results. As compared to Simulation 2, we see that adding jitter on top of the previous inter-stimulus variability allows for improved estimates (smaller RMSE); our technique is able to efficiently exploit both of these sources of disambiguating information in combination.

5.2.2 Analyzing a response-related potential

In the previous chapter, we used a positivity associated with giving a response to illustrate how non-linear regression could detect components whose latency varied with some continuous factor. The non-linear regression approach is well-suited to detecting such latency effects in general, regardless of the factor which produces them. In this particular case of a response-related potential, though, there's another way to think about what's happening: there are in fact two events — the presentation of the stimulus, and the participant response — whose time-locked brain responses overlap.[1] This suggests that we might do even better to analyze that data with this method, simultaneously estimating the rERP time-locked to the stimulus presentation (time zero in the previous figures) and the rERP time-locked to the response (the thick black line on the previous figures). Figure 5.5 shows the resulting rERPs.[2] As expected, the rERP time-locked to the response shows a positivity that begins precedes ∼200 ms before the button press, and continues ∼300 ms after it.

In estimating these rERPs, we assumed that every stimulus and every response produced exactly the same ERP, regardless of response latency; i.e., the only difference between a fast response and a slow responses is that they happen at different times. We might wonder how accurate this is — is there different componentry between fast and slow responses, reflecting their production via different routes? To find out, we

[1] Thanks to David Groppe for suggesting this application.

[2] Data from Urbach & Kutas, 2008. Thanks to Tom Urbach for sharing this data.

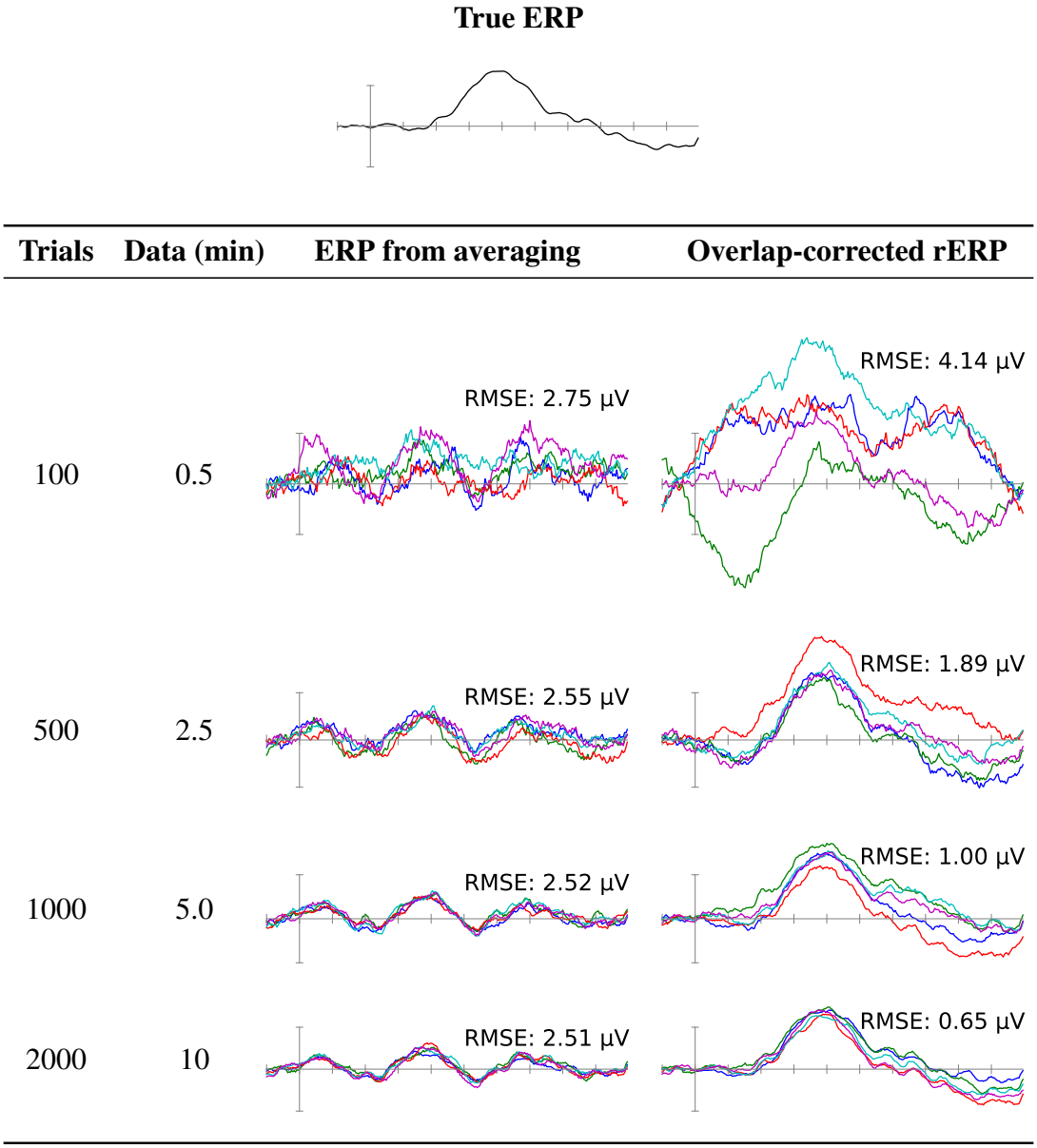

Figure 5.4: Simulation 3: Conventional averaged ERPs, and overlap-corrected rERPs, estimated from various amounts of simulated data, with SOAs varying randomly between 200–400 ms and a continuous factor which acts linearly on the ERP. As before, we plot analyses of 5 simulated data sets, while 100 simulated ERP/rERP estimates were used to calculate the RMSE (standard error).

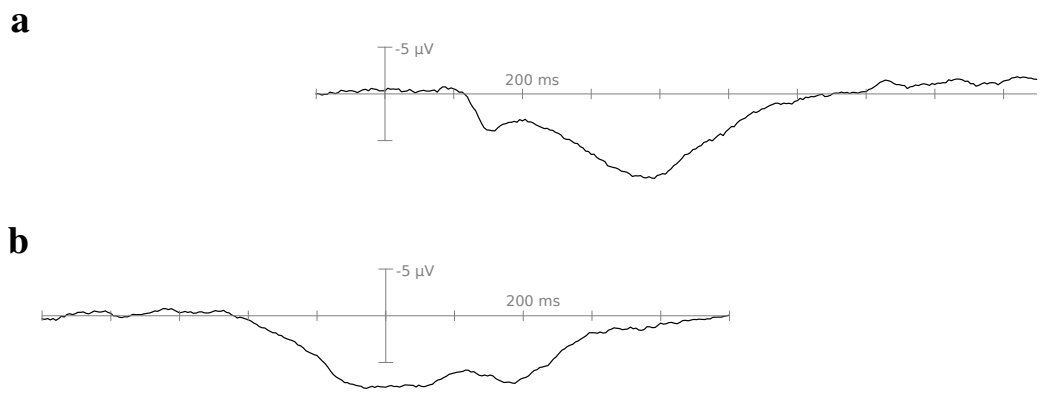

Figure 5.5: rERP estimates time-locked to stimulus presentation (**a**) and response (**b**) from the go/no-go experiment data used in the previous chapter, as estimated by our overlap correction technique. In **a**, 0 ms corresponds to the stimulus presentation time. In **b**, 0 ms indicates the button press time.

calculated the predicted ERP at a range of different response times from this model by taking the stimulus-locked rERP and adding a shifted copy of the response-locked rERP. These predicted ERPs are shown in Figure 5.6a. For comparison, Figure 5.6b shows a non-linear model fit from the previous chapter on the same data. This is a very flexible model (with 13150 free parameters) which gives a fairly direct representation of the trial-level data. Yet our new model, with only 514 parameters, seems to capture all of the structure that the flexible model does. Figure 5.6c shows the difference between these images, with a 'zoomed-in' colormap to make any differences more visible. No structure jumps out at us — especially for values in the 300–600 ms range on the y-axis, which is where most of our data lies.

We don't want to overinterpret this, since this is an illustration of our technique, not a rigorous study; if we were doing this for real, we'd certainly analyze more than one electrode, look at different participants individually, and so forth. But as far as this example goes, we don't see any evidence that fast and slow responses differ systematically in any way besides a simple shift in the response-locked positivity.

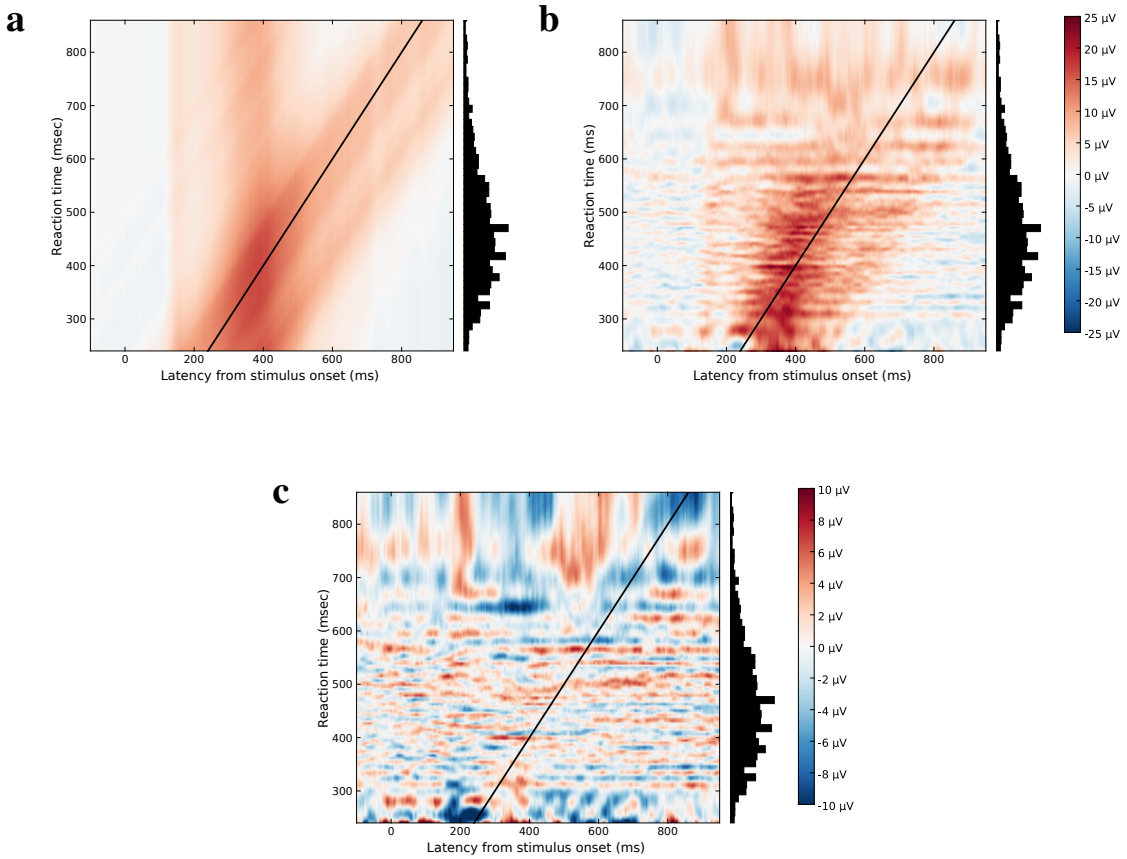

Figure 5.6: (**a**) Predicted ERPs at different response latencies, according to the overlap model (Figure 5.5). (**b**) Predicted ERPs from a non-linear regression model fitting a cubic spline basis with 50 degrees of freedom at each time point (a duplicate of Figure 4.7b). (**c**) The difference **b** − **a**. Note that, to make details more visible, this image uses a different colormap than the above two. There is no clear structure visible in this image, suggesting that the model in (**a**) has successfully captured the important features of the data.

5.2.3 Comparison to Adjar

Perhaps the best-known method for overlap correction is ERP studies is the *Adjar* technique, introduced by Woldorff (1988, 1993). Adjar and our technique are very similar in spirit: they both assume that the ERPs from adjacent trials simply sum, with an offset, and then attempt to reconstruct what the ERPs would have to be in order to produce the observed data; and they both do this by, at some level, taking that observed data, and attempting to work out what variations in the scalp potential time-locked to one event should be attributed to the previous event instead. However, in our technique, this attribution problem is solved implicitly by the standard least-squares fitting machinery; aspects of the data which are explained by one predictor cannot also be explained by another, so the fitting process must choose which to use. In Adjar, this problem is solved explicitly, by an iterative process of calculating ERPs, convolving them with the SOA distribution to estimate how they contribute to the ERPs to adjacent stimuli, and then subtracting off this contribution. To simplify calculations, some parts of the analysis use pooled averages rather than stimulus-specific averages; therefore, it is necessary to ensure that all combinations of stimuli orderings are observed, and in roughly equal proportion. This also limits Adjar's ability to handle the kind of inter-stimulus variability that our technique exploited in Simulations 2 and 3, and even with such simplifications, the analyst is still required to make some reasonably complex judgements about how best to implement the procedure in their particular case.

While we are generally in favor of the use of judgement when performing statistical analyses, this degree of manual attention seems excessive given the availability of computers with sophisticated linear algebra libraries and copious memory. We prefer to leave such grunt work to them. Overall, we see our procedure as essentially a modernized implementation of the principles underlying Adjar, in which we as analysts take on the job of accurately describing what happened in the experiment (by setting up our predictors), and the computer then automatically derives the optimal method for disentangling these effects.

The other advantage of our technique, of course, is that it fits neatly into the rERP framework, which means that it can be straightforwardly combined with all of the other capabilities that framework provides, such as the use of continuous or non-linear

predictors.

5.3 Generalized Least Squares

5.3.1 Motivation

Now that we've seen the advantages of using a combined regression model for overlap correction, we turn to the more speculative portion of the paper. Our motivation here is that, if we are to believe statistical theory, it is possible to both extract more information from our data, and to perform powerful statistical tests directly within our regression model.

These tests are the more tempting part of the package, because if they can be made to work then they may allow us to detect significant effects within individual participants, rather than only at the population level. This would give us a more detailed view into these effects by allowing us to more finely parse individual variation, and, potentially, give us vastly more power to detect particular effects. This is because, intuitively, the relevant N is the the number of EEG samples we can record or the number of trials that we can run rather than the number of participants that we are able to bring into our lab, which is many orders of magnitude smaller. This would be particularly valuable for the corpus-style studies that rERP enables, because studies which cast a wide net rely heavily on multiple-comparison correction methods which are very hungry for power.

And it's not clear how else we could get this power. Current methods are sufficient for detecting single effects, but it is a rare ERP study that reports an effect at $p < 0.001$, and increasing our participant counts by an order of magnitude would be impractical. Alternatively, we might try applying non-parametric methods such as bootstrap or permutation tests within individual participants, but these face two significant challenges that would have to be overcome. The first is that there simply isn't any really good way to apply them to data sets containing overlapping. For epoched data, we can resample epochs. If we have no epochs, but only a long contiguous stream of overlapping ERPs embedded in EEG noise, then we resample... what? There are proposals in the literature (Lahiri, 2003), but this remains an active area of research, and solutions

necessarily depend on having a reasonably accurate model of the structure of your data — which makes them rather similar to the parametric approaches we investigate here. Secondly, even if we worked out a way to make them work, multiple-comparison corrections require us to be able to calculate very small p values. For example, if we check all points in a 1-second rERP sampled at 250 Hz with Bonferroni correction, then we need $p < 0.0002$ for significance at the 0.05 level. Simply measuring such a small p value with the bootstrap would require on the order of 500,000 resamples, which is computationally prohibitive — and this is a relatively modest application of such corrections. This challenge too may be solvable, but it isn't clear how.

So, we turn to the aforementioned statistical theory. To understand where our opportunity lies, consider the list of requirements for using least-squares regression given in every statistics textbook. It goes something like: the errors must have a gaussian distribution, and be homoskedastic, and uncorrelated. This is something of an overstatement; in fact, none of these are required for least-squares regression to be unbiased (it gets the right answer on average) and consistent (given enough data, it's always close to right). However, when these conditions do hold, theory says, we get two benefits. First, if the errors are homoskedastic and uncorrelated, this implies that the ordinary least-squares estimator is the best possible unbiased estimator that can be computed using linear algebra (that is, given limited data, it will do better than the alternatives; this is the "Gauss-Markov theorem"). And secondly, all three conditions together are needed to ensure that the standard t- and F-tests will give accurate p values.

For the regression models in this chapter, the errors under discussion are the EEG background noise, sampled at successive points in time. How well does it meet these conditions? These scalp potentials have a distribution which is fairly close to gaussian (though with slightly heavier tails); close enough for most purposes. The homoskedasticity requirement says that we expect the background noise to be equally noisy at all moments, or at least not to vary with our experimental conditions; this is also quite reasonable, though again not perfect — any stimulus-related induced change in EEG power will manifest as a departure from homoskedasticity. These deviations, however, should be rather small compared to the overall EEG power. And finally, the requirement the the noise be uncorrelated says that if we pick a moment at random to observe the scalp

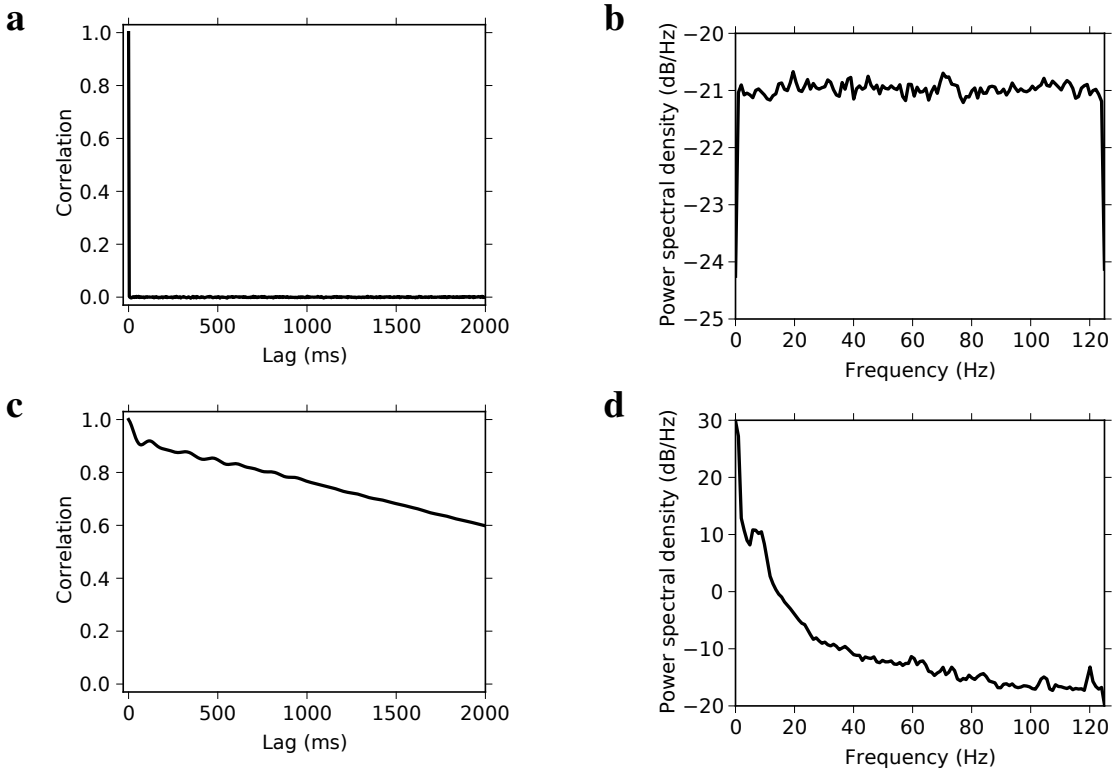

Figure 5.7: (**a**) The autocorrelation function (ACF) of a signal where each point is sampled independently. It has the value one at lag zero, and zero everywhere else. (**b**) The power spectral density (PSD) for this same signal. (**c**) The autocorrelation function computed from a $\sim$2 minute stretch of EEG data. It has the value one at lag zero, and decreases linearly as lag increases. (**d**) The well-known 'pink noise' power spectral distribution for EEG, with its $1/f$ shape (plus a hump at 10 Hz due to alpha — the corresponding 10 Hz wiggles in the ACF are just barely visible).

potential, then this should tell us absolutely nothing about what the scalp potential will be when we take our next observation, 4 ms later. This assumption is clearly laughable. Intuitively, we know that nearby points in an EEG recording will have, on average, somewhat similar values.

We can quantify this violation by computing what is known as the *autocorrelation function*, ACF(n). This function is measured by taking random pairs of points that are n milliseconds apart (n is known as the *lag*), and computing their correlation. For any signal, ACF$(0) = 1$, because each point is always perfectly correlated with itself; but as n increases, we expect to see ACF(n) decrease. According to our regression requirements, the autocorrelation function should drop immediately to zero

and stay there, as shown in Figure 5.7a. A useful fact is that for any signal, the ACF and the power spectral density (PSD) form a Fourier transform pair. Since the Fourier transform of a point impulse function is a flat line, this means that a signal containing uncorrelated values has a flat power spectrum, as shown in Figure 5.7b. Such a signal is therefore called *white noise*, since light with an analogous spectrum would appear white.

The ACF for actual EEG data looks quite different: rather than dropping instantly to zero, or even dropping quickly to zero, it diminishes in a leisurely linear fashion (Figure 5.7c). This property of the EEG is equivalent to another, much better known property: that it has a $1/f$ or 'pink' spectrum (Figure 5.7d). When we take the Fourier transform, the large values of the PSD at low frequencies correspond to a large DC component in the ACF; in a pink noise signal, arbitrarily distant points in time are still quite correlated.[3]

So clearly the 'requirement' of uncorrelated errors is massively violated for the regression models we consider here. This doesn't invalidate their use, any more than it invalidates the use of conventional EEG averaging — they will still give unbiased estimates for our β coefficients. But it means that we cannot use standard parametric significance tests, and it means that there may be other ways of performing our estimation which produce better estimates of the underlying ERP signal, from fewer trials.

5.3.2 Mathematics

In fact, such a method exists, and is known as *generalized least squares* (GLS), to distinguish it from the *ordinary least squares* (OLS) that we've been using so far.[4]

[3]One might quibble at this reference to non-zero correlation at arbitrarily large lags, since clearly the pictured ACF will hit zero eventually. But this is actually something of an illusion, caused by our estimating the ACF from a finite stretch of data. Technically speaking, a true pink noise signal is non-stationary, which implies that neither its PSD nor ACF are well-defined; as you sample longer and longer signals, there is more and more opportunity for very large, very low-frequency noise to intrude, and so the empirical PSD records more and more power at lower frequencies, and the empirical ACF drops off more and more slowly. However, actual measured EEG data stops being pink before this can occur, because the high-pass filter applied during the acquisition process suppresses these very low frequencies. (Indeed, this is the whole reason that such a filter is necessary: our analog/digital converters would quickly saturate if exposed to real pink noise.) But in any case, the correlations in EEG are very strong on the time-scale of typical experimental manipulations.

[4]NB: The word "generalized" in "generalized least squares" has nothing to do with its usage in the term "general linear model" (GLM). "General linear model" means either the same thing as "linear model"

The essential idea is this: if we know what the correlations and heterskedasticity in our errors looks like, then we can correct for it using some linear algebra. Having done so, we regain the nice properties that we expect from linear regression: GLS is the best possible unbiased estimator for our βs that can be computed using linear algebra, and (given gaussian errors) the standard significance tests again begin to produce accurate p values (Aitken, 1935).

Unfortunately, this approach does not lead to better estimates of the ERPs measured in traditional orthogonal designs that use widely separated epochs; in such cases, the GLS estimates are identical to those produced by OLS or averaging. (For details, see Appendix 5.A.) However, in the presence of collinearity (whether arising from the characteristics of the stimuli themselves, or introduced by overlap), our rERP estimates can be improved. And, in all cases, the GLS model opens the possibility of using the powerful statistical techniques we propose here — even with traditional orthogonal designs.

Here's how it works. In OLS, we assume that the data $\boldsymbol{y}$ were generated by additively combining some deterministic function of our predictors, $X\boldsymbol{\beta}$, with random noise, $\boldsymbol{\epsilon}$:

$$\boldsymbol{y} = X\boldsymbol{\beta} + \boldsymbol{\epsilon}$$

Furthermore, we assume that each entry in the vector $\boldsymbol{\epsilon}$ was generated by taking an independent sample from a normal distribution $N(0, \sigma^2)$. A different, equivalent, way of stating this assumption would be to say that the whole vector $\boldsymbol{\epsilon}$ was generated by taking a *single* sample from *multivariate* normal distribution $N(0, \sigma^2 I)$, where I is the identity matrix. (This is where the homoskedasticity and uncorrelated errors ideas come from — homoskedasticity just means that all the entries on the diagonal of this matrix are the same, σ^2, and uncorrelated errors just means that all the off-diagonal entries are

alone, or else it means the multivariate extension of linear modelling in which many sets of data are regressed onto the same set of predictors in an efficient way — see previous chapter's Appendix. It also has nothing to do with "generalized linear modeling" (also abbreviated GLM), which is the extension of standard linear regression to situations in which data may have a non-gaussian distribution, and there may be a non-linear transformation applied to the data (the *linking function*) before fitting is performed (McCullagh & Nelder, 1989). The use of the word "general" for all of these different techniques presumably reflects only the regrettably limited size of the English lexicon.

zero.)

Then the maximum likelihood estimate $\hat{\boldsymbol{\beta}}$ is the ordinary least squares estimator

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

That's the formula we use for estimation, but to understand how $\hat{\boldsymbol{\beta}}$ is distributed, it's useful to expand it further:

$$\begin{aligned}\hat{\boldsymbol{\beta}} &= (X^T X)^{-1} X^T (X\boldsymbol{\beta} + \boldsymbol{\epsilon}) \\ &= \boldsymbol{\beta} + (X^T X)^{-1} X^T \boldsymbol{\epsilon}\end{aligned}$$

In general, if A is a matrix and $\boldsymbol{b}$ is distributed according to a multivariate normal $N(\boldsymbol{\mu}, \Sigma)$, then $A\boldsymbol{b}$ is distributed according to a multivariate normal distribution $N(A\boldsymbol{\mu}, A\Sigma A^{)}$. Therefore, we have that the covariance matrix of $(X^T X)^{-1} X^T \boldsymbol{\epsilon}$ is

$$\begin{aligned}&(X^T X)^{-1} X^T \sigma^2 I \big((X^T X)^{-1} X^T\big)^T \\ &\quad = \sigma^2 (X^T X)^{-1} X^T X \big((X^T X)^{-1}\big)^T \\ &\quad = \sigma^2 \big((X^T X)^{-1}\big)^T \\ &\quad = \sigma^2 (X^T X)^{-1} \qquad \text{(because } X^T X \text{ is symmetric)}\end{aligned}$$

So our estimate $\hat{\boldsymbol{\beta}}$ is the sum of the actual, true $\boldsymbol{\beta}$, plus some gaussian noise with mean zero and a sampling covariance matrix that is known except for the scaling factor, σ^2. Therefore, the expected value of $\hat{\boldsymbol{\beta}}$ is $\boldsymbol{\beta}$ (i.e., it is an unbiased estimator), and we can use the covariance matrix $\sigma^2 (X^T X)^{-1}$ to construct significance tests. For example, the diagonal elements of $\sigma^2 (X^T X)^{-1}$ give the sampling variances for each of our βs, and if we take the square root of the sampling variance we get the standard error.

In GLS, we use the same setup:

$$\boldsymbol{y} = X\boldsymbol{\beta} + \boldsymbol{\epsilon}$$

But, now we assume that $\boldsymbol{\epsilon}$ was generated by taking a sample from a multivariate normal distribution $N(0, \sigma^2 \Sigma)$, where Σ is an arbitrary covariance matrix. (Technically we

don't really need to separate out the σ^2 scaling factor here; we could just merge it into the overall matrix, since the matrix is arbitrary anyway. But in ordinary least-squares, it turns out that we can estimate $\boldsymbol{\beta}$ without estimating σ^2; similarly, in generalized least squares, we only have to know Σ up to a multiplicative factor. So separating out σ^2 emphasizes this.) Under this assumption, the maximum likelihood estimate for $\hat{\boldsymbol{\beta}}$ becomes

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

We could use this expression directly to compute $\hat{\boldsymbol{\beta}}$, but there's an easier way.

Suppose that we have some matrix S. Then we can take the basic equation that defines our data, and multiply both sides by S:

$$S\boldsymbol{y} = SX\boldsymbol{\beta} + S\boldsymbol{\epsilon}.$$

Since we did the same thing to both sides of the equation, the equality still holds. In particular, the solution $\boldsymbol{\beta}$ that solves this equation is the same as the one that solved our earlier equation. What we've done is basically change our original regression problem into a new regression problem:

$$\underline{\boldsymbol{y}} = \underline{X}\boldsymbol{\beta} + \underline{\boldsymbol{\epsilon}}$$

where $\underline{\boldsymbol{y}} = S\boldsymbol{y}$, $\underline{X} = SX$, $\underline{\boldsymbol{\epsilon}} = S\boldsymbol{\epsilon}$. This new regression problem has the same solution as our old one, but — and this is critical — it has a different noise term. By assumption, $\boldsymbol{\epsilon}$ has covariance matrix $\sigma^2\Sigma$. So we can use the same formula we used above for linear transformations of random vectors, and infer that $\underline{\boldsymbol{\epsilon}}$ has covariance matrix $\sigma^2 S\Sigma S^T$.

So, here's the trick: let's choose our S so that $S\Sigma S^T$ becomes the identity matrix. Since Σ is a covariance matrix, it must be positive definite; that means that we can take its Cholesky decomposition, $CC^T = \Sigma$, and then invert that, $S = C^{-1}$. Now we have

$$\begin{aligned} S\Sigma S^T &= C^{-1}\Sigma\left(C^{-1}\right)^T \\ &= C^{-1}CC^T\left(C^T\right)^{-1} \\ &= I. \end{aligned}$$

What this means in English is that S is a matrix which turns our correlated noise, $\boldsymbol{\epsilon}$, into white noise, $\underline{\boldsymbol{\epsilon}}$.

The result is that if we use this choice of S, our new regression model using $\underline{\boldsymbol{y}}$ and $\underline{X}$ is one where the noise has a simple $\sigma^2 I$ covariance matrix, so we can use ordinary least squares without guilt. When we do, we find that the OLS estimate is

$$\begin{aligned}\hat{\boldsymbol{\beta}} &= (\underline{X}^T\underline{X})^{-1}\underline{X}^T\underline{\boldsymbol{y}} \\ &= (X^T S^T S X)^{-1} X^T S^T S \boldsymbol{y} \\ &= (X^T \Sigma^{-1} X)^{-1} X^T \Sigma^{-1} \boldsymbol{y}.\end{aligned}$$

which is the same as the GLS estimator for our original problem, as shown above.

So the easy way to compute the GLS solution to a regression problem is to:

1. Figure out what kind of linear transformation you would need to apply to your noise in order to turn it into white noise.
2. Apply this linear transformation to both your data and your predictors.
3. Regress your transformed data on your transformed predictors, using ordinary least squares.

The β coefficients estimated by this transformed model are, as usual, unbiased estimators of your rERP. Given enough data, the transformed model and the original model will both produce the same answer. Actually, for purely orthogonal designs, it turns out that they will always produce *exactly* the same answer (Appendix 5.A). But in the presence of collinearity then they may differ, and when this occurs the transformed model will on average give more accurate results with fewer data. Furthermore, in the transformed model, we can use standard parametric statistical tests. This happens because in GLS, the sampling covariance matrix for our βs is estimated to be $(X^T\Sigma X)^{-1}$ rather than $(X^T X)^{-1}$. What this is does is account for correlations between our βs which are caused by them being estimated from correlated data. The $(X^T X)^{-1}$ form assumes that all correlations between β estimates are caused by collinearities in our design; for EEG this is not true, and therefore using this form gives wrong answers. The $(X^T\Sigma X)^{-1}$

form, on the other hand, more accurately allows for correlations to arise both from collinearities and directly from the data.

5.3.3 Finding S

But before we can take advantage of GLS, we need to find the proper whitening transform S for our data. This transform needs to be efficient to apply, it needs to accurately correct for the correlations which are present, and — the trickiest part — it needs to be flexible enough to work for different participants, scalp locations, recording conditions, and so forth, all of which may produce somewhat different autocorrelation structures.

Fortunately, there is a straightforward strategy for estimating the autocorrelation structure from data; the general approach is known as *feasible generalized least squares* (FGLS). First, we fit our model by OLS; our eventual goal is to use GLS, but we can't until we know S. So this isn't the best fit possible — but it does give us an unbiased estimate of our βs, from which we can calculate the model residuals (i.e., the differences between the predicted values and the actual data). These residuals give us a sub-optimal but unbiased estimate of the background noise signal whose correlation structure we want to model. We then use our favorite technique to estimate S based on the residuals, and then we re-do our regression fit, now using GLS with this S matrix instead of OLS.

But this S was based on unnecessarily noisy data; now that we have a model fit using GLS, we can get a new, better estimate of the residuals from it, re-fit S to them, re-run the GLS and around and around we go until we reach convergence (usually about 3–4 iterations).

The question then becomes, how should we estimate S, our whitening transform, from a non-white time-series, our residuals?

ARMA models

The most common way of modeling empirical correlations in time-series data is to use an ARMA model. The AR stands for *auto-regressive*: an AR(p) models assumes that each new value in our time series is generated by taking a weighted sum of the previous p values, plus some new noise. (One way to think of this is that it assumes

our time-series was produced by running an order p IIR filter over white noise.) The MA standards for *moving average*: an MA(q) model assumes that we have a stream of gaussian variates, and each value in our time series is produced by taking a weighted sum of the last q of these variates. (This corresponds to running an FIR filter with order q over white noise.) An ARMA(p, q) model assumes that we took a time-series from an AR(p) model, and another time-series from an MA(q) model, and added them together.

These models are widely used for handling time-series in economics, climatology, fMRI, etc.; they're the default tool for solving the kind of problem we face here, of estimating the structure of a non-white noise signal so that it can be whitened. Unfortunately, though, the correlations in EEG noise are much worse than they are designed to handle. In an AR(p) model, the auto-correlation function for points more than p steps apart drops off exponentially quickly. For an MA(q) model, it's even worse: at a lag of q samples it drops off immediately to zero. Therefore, these models are incapable of accurately fitting pink noise, where the correlations drop off much more slowly. We would need such a large p or q that we'd be fitting about one parameter for every data point, which is not a recipe for successful statistics.

So, what else can we do?

Option 1: Keep it simple

In a traditional ERP experiment, we have a large number of trials, and from each we extract a fixed-length epoch of data time-locked to some event of interest. A more-or-less reasonable assumption (depending on how widely spaced our trials are) is that background noise from different trials is not correlated; only noise within a trial is correlated. And it's also reasonable to assume that the correlations within each trial are the same.

This suggests a trivial method for estimating S: just calculate the empirical sample covariance matrix for your data across all of your trials, $\hat{\Sigma}$, and then calculate $\hat{S} = \text{Cholesky}(\hat{\Sigma})^{-1}$. However, this method's simplicity is marred somewhat in practice when the number of data points in each epoch exceeds the number of trials, because $\hat{\Sigma}$ will be rank-deficient. This means that we can't depend on either the Cholesky decomposition or the matrix inverse to be computable. This can also be solved in a crude but

effective manner: we just add λI to our empirical covariance matrix, where λ is a small positive number.

The easiest way to think about this is in terms of the eigenvalues of $\hat{\Sigma}$ — to be a valid covariance matrix, they must all be positive and non-zero. When we add λI to a matrix, then we add λ to each of its eigenvalues. So we will be able to perform our calculations as long as λ is sufficiently large (and something like 10^{-6} is sufficiently large). This also has a possible secondary benefit, which is that this method tends to overfit the covariance matrix, and adding λI tends to smooth the resulting matrix. As a result, using a larger value of λ will reduce apparent performance on our original data, but probably get us closer to the true underlying covariance matrix. This is demonstrated in Figure 5.8: with a small λ, we do a better job of whitening the original data, but a worse job of whitening a second set of epochs drawn from the same recording. It isn't yet clear, though, whether this overfitting is a good or bad thing in practice, since it applies only to our model of the noise.

This approach is extremely simple to implement. It also has the advantage that, since it makes essentially no assumptions about the form of the background noise, it may be robust to strange distributions introduced by artifacts such as eye-blinks or muscle noise. However, we cannot use this approach if we have overlapping ERPs, because they cannot be divided into independent epochs. So let's consider some other approaches.

Option 2: Custom-designed whitening filter

Our overal goal here is to fix the autocorrelation function of our background noise; but since the autocorrelation function is equivalent to the power spectrum, another way to think about this is that we want to find some filter that will take the $1/f$ distribution from Figure 5.7d and flatten it out into the distribution Figure 5.7b. So, why not use some standard technique to estimate the PSD for our data, and then use standard digital filter design techniques to create a filter whose frequency response is tuned to be the inverse of the PSD?

Applying filters to EEG is a very tricky subject in general (Luck, 2005), but here we sidestep most of those issues, since we are not actually filtering our ERP estimates (i.e., the βs). Remember that no matter what strange and curious transformation we

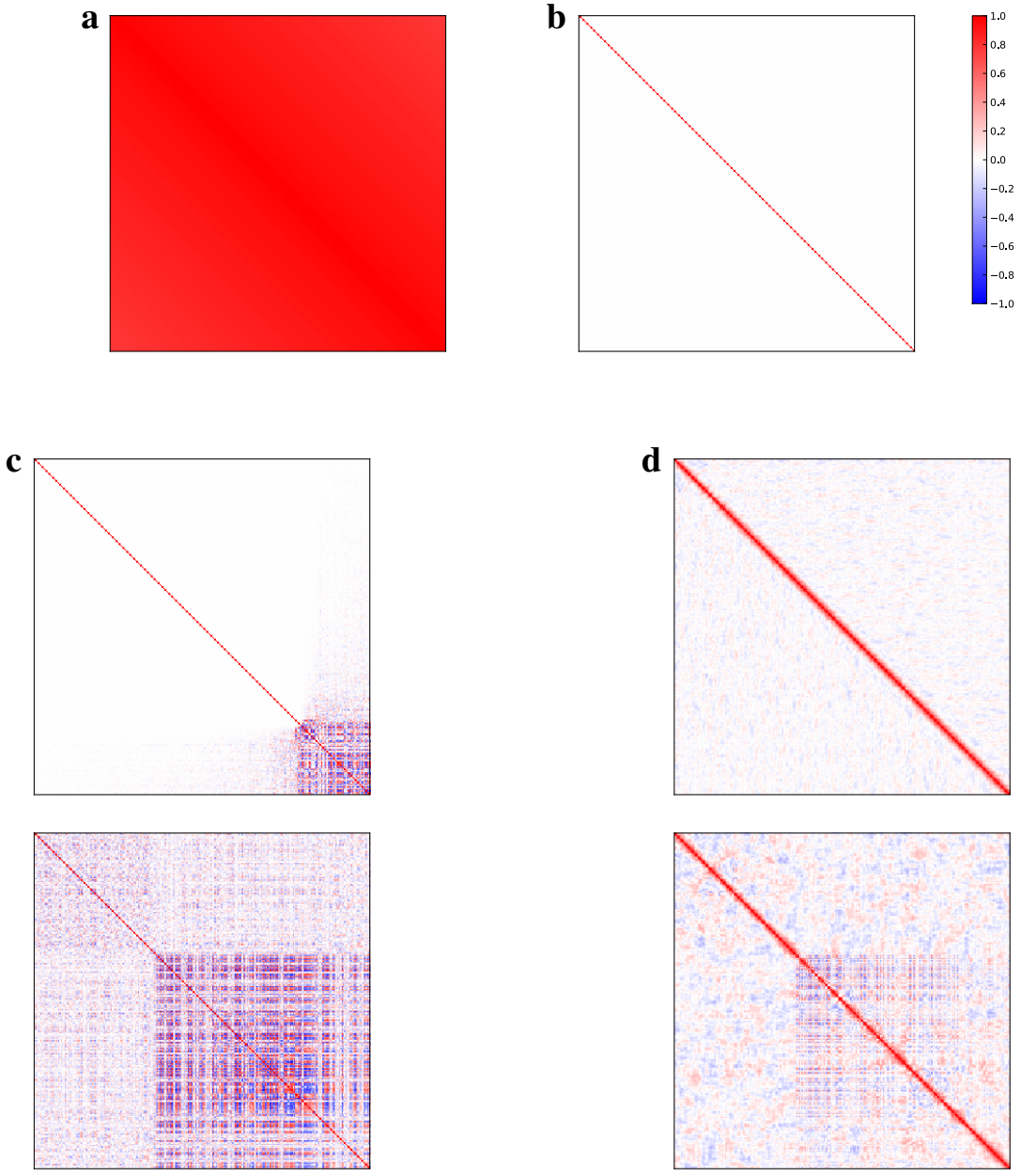

Figure 5.8: Plots of correlations between different points in 200 one-second epochs of EEG data. (**a**) The correlation matrix for raw EEG data. (**b**) The ideal correlation matrix for white noise. Our goal is to transform **a** into **b**. (**c**) 'Option 1' applied to these data, with $\lambda = 10^{-6}$. Top: the data that was used to calculate the covariance matrix, after being whitened. Bottom: A different set of data, after being whitened using the same covariance matrix. (**d**) The same, but with $\lambda = 100$. The top image here actually contains more total residual correlation than the top image in **c**, but it is distributed more uniformly, and this transformation generalizes much more successfully to the new data below.

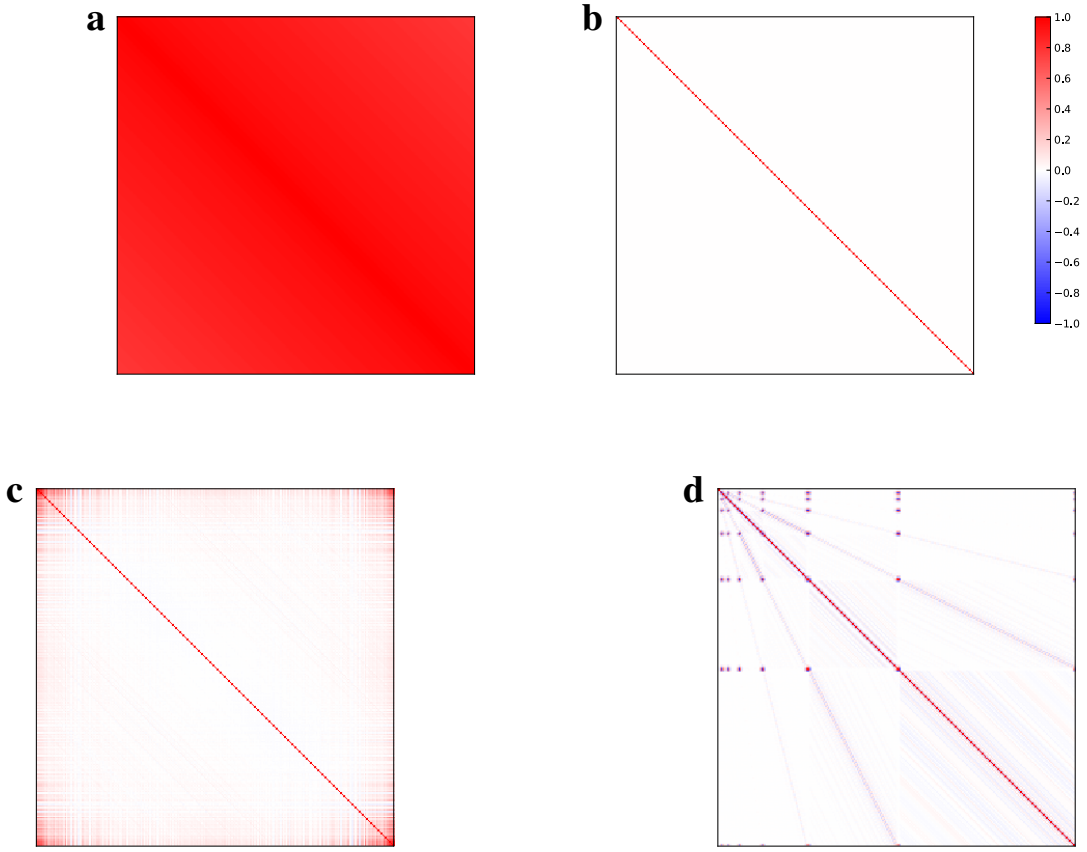

Figure 5.9: Plots of correlations between different points in one-second epochs of EEG data. (**a**) The correlation matrix for raw EEG data. (**b**) The ideal correlation matrix for white noise. Our goal is to transform **a** into **b**. (**c**) 'Option 2' applied to these data, using a 249-tap FIR filter. Residual correlations in the corners are due to edge effects impacting the filter's ability to whiten accurately. (**d**) 'Option 3' applied to these data, using the `sym4` wavelet family with a seven-level decomposition. The faint boxy pattern reflects the different wavelet decomposition scales; the grid of points at the corners reflects edge effects, which occur separately at each scale. The faint diagonal lines reflect residual correlation between wavelets at different scales but the same location.

might choose use for S, we will still get an unbiased estimate of our regression coefficients. Our filter will, in effect, only be applied to the background noise. So most of the issues which people usually worry about when filtering — phase delay, the necessary trade-offs between time-domain distortion versus frequency-domain filter accuracy, and so forth — are simply irrelevant, because we don't care whether the background noise becomes distorted, so long as its spectrum gets flat.

An appropriate tool for designing such filters is the so-called windowed technique (e.g. Smith, 1997, ch. 17) which takes an arbitrary frequency response curve and produces an FIR filter of arbitrary order which approximates that frequency response. This is available as a standard function in most signal processing libraries; see, for example, the `firwin2` function in SciPy (Jones, Oliphant, Peterson, et al., 2011), or the `fir2` function in the Matlab Signal Processing Toolbox (Mathworks, 2011). See Figure 5.9c.

Option 3: The wavelet trick

The discrete wavelet transform (DWT) is analogous to the Fourier transform — it takes a time series, and applies a linear transformation to represent it in another basis. A wavelet basis has a particular structure, so that if we apply the DWT to a signal with n samples, then we get out $n/2$ samples representing the highest frequency portion of the signal, $n/4$ samples representing the next highest frequency portion, and so on until we run out of samples. (This means that the wavelet decomposition of a signal has coefficients on about $\log_2(n)$ different scales.)

This makes it a useful tool for time/frequency analysis, and wavelets have been applied to EEG analysis for this purpose. Our usage here is completely unrelated to this. For us, wavelets are interesting because of one key theorem: it turns out that if we take the wavelet decomposition of a $1/f$ signal, then the resulting wavelet coefficients are (approximately) uncorrelated (Wornell, 1990; Tewfik & Kim, 1992).[5] This holds quite generally; the only requirement is the technical one that we use a wavelet basis with at

[5]This also, incidentally, provides a useful way to think of what makes a noise into $1/f$ noise. White noise is the concatenation of a series of independent random numbers. Pink noise is the summation of a series of transient fast waves (the fine-scale wavelet basis functions) and transient slow waves (the coarse-scale wavelet basis functions), with the slow waves overall larger than the fast ones, and the amplitude of each wave is an independent random number.

least 4 vanishing moments. (This means that when the wavelet decomposition divides the signal into low-frequency and high-frequency components, polynomial curves of up to degree 4 are always considered low frequency components.) And since the DWT is a linear transformation, we don't actually have to compute the matrix S and then multiply our various other matrices by it — we can just use an off-the-shelf DWT library to transform our data directly. See Figure 5.9d.

The one thing we have to watch out for is that while the DWT decorrelates the data, it does not produce homoskedastic data. This is unsurprising if we think about it; since EEG has a $1/f$ spectrum, the low-frequency parts of the decomposition must have higher power (i.e., variance) than the high-frequency parts. So applying the DWT is enough to get us from a $1/f$ noise covariance matrix to a diagonal covariance matrix, but according to GLS theory, we can't stop there — we still need to rescale the wavelet coefficients obtained at each decomposition scale to equalize their variance, and to do that we need to estimate them inside an FGLS fitting loop.

Fadili and Bullmore (2002), who first proposed the usage of wavelets for GLS whitening in the context of fMRI, spend some energy on deriving a theoretical formula for the relative variance expected at each scale. We prefer a simpler approach, of computing the sample variance of the wavelet decomposition of our residual time series at each level of decomposition; this is easy to implement, requires only a handful of additional free parameters, and gains some robustness against possible real-world deviations from the theoretical $1/f$ noise model.

As for a choice of basis, we currently prefer the 4th order symlets (also known as `sym4`). Why? On the one hand, we want a wavelet basis that has 4 vanishing moments, in order to ensure that the whitening theorem applies. On the other hand, we would like to use a wavelet with as few coefficients as possible, in order to minimize edge effects, which tend to reduce the efficacy of our whitening (Fadili & Bullmore, 2002). The shortest possible wavelets with 4 vanishing moments have 8 coefficients. It's likely that any wavelets that achieved this lower bound would work equally well. Among these, `sym4` is the one that comes closest to having linear phase, which means it will tend to distort our signal less than the alternatives. (Usually linear phase is mentioned in the context of filters, but the wavelet decomposition basically works by using a band-

pass filter to compute each set of wavelet coefficients, so it makes sense here as well. All filters introduce some delay into their output, and the amount of delay may vary for different frequencies, introducing distortion; linear phase filters introduce the same delay at all frequencies.) Like we said above, we have no reason to think that having linear phase actually matters for this application. But we have to pick something, so superstition seems is as good a reason as any.

Which option should we pick?

The options proposed here make different tradeoffs between computational practicality and their flexibility for handling unusual data.

Option 1, the empirical covariance approach, makes no assumptions about the form of the noise, making it infinitely adaptable. It requires epoched data to estimate, which is a limit to its use; but this could conceivably be fixed by imposing some assumptions about its structure (e.g., if we assume the background noise is stationary, that would imply that Σ was Toeplitz). But such an approach would still require explicitly forming the covariance matrix $\hat{\Sigma}$. If we have n data points, then this matrix has n^2 entries. If we wanted to form this matrix for the 10 minutes of overlapped data shown in Figure 5.2, then we would need 180 gigabytes of RAM (or 90 gigabytes for single precision). Hardware that can store such matrices is not widely available, and inverting such a matrix is even more of a challenge.

So this was part of the motivation for the next two options: both allow us to calculate the effect of multiplying by S without explicitly forming the matrix S in memory. The filtering option is conceptually straightforward, and has the advantage that it can straightforwardly adapt to artifacts with a periodic nature, since their effect is to alter the spectral distribution of the signal. (Examples include alpha, 60 or 50 Hz line noise, heartbeat and muscle artifact.) The wavelet option is less flexible, in that it assumes that the noise has a clean $1/f$ spectrum; its flexibility is limited to adapting to different exponents on the $1/f$, which correspond to different allocations of power between the different levels of decomposition.

On the other hand, when working with continuous data containing overlap, we already have extra motivation to use data-sparing artifact correction techniques such as

ICA (Makeig, Bell, Jung, & Sejnowski, 1996). As a practical matter, since we have no epochs, it is unclear how we should define the boundaries of the data to be rejected around an artifact, and each span of data which we reject will impact on multiple trials. If we are using either the filter- or wavelet-based GLS technique, then this adds an additional motivation, since every time we reject a span of data we will introduce new boundary artifacts (Figure 5.9). So it's unclear that filter approach's greater flexibility is important in practice; we may be removing the relevant artifacts in a pre-processing step regardless of which option we use.

The other advantage of the wavelet approach is that it can be implemented extremely efficiently. Not only does it allow us to avoid forming the S matrix, but the discrete wavelet transform can be performed extremely quickly (faster, even, than the FFT), and the simplicity of the required parameter estimation allows us to perform the full FGLS iterative fitting algorithm in only a single pass through our data, and without forming the X matrix in memory (see Appendix 5.B).

5.3.4 rERPs with less noise

One of the motivations for making GLS work is that, at least in some cases, it should give us more accurate rERP estimates. This benefit is probably of less importance than the statistical tests described below, but even so, it's worth examining how it plays out in practice. In Simulation 4, we repeated Simulation 1 above (Figure 5.2, but this time we computed filter- and wavelet-based GLS estimates in addition to the OLS estimates we used previously. To make this test more challenging, we also cranked up the degree of overlap: in this simulated experiment, the SOA varies randomly between 150 and 250 ms, meaning that there are at least four and potentially as many as eight ERPs overlapping in the stimulus at any one time.

The results are shown in Figure 5.10. The GLS estimates outperform the OLS estimates in terms of RMSE in all cases except for the 4000 trial wavelet GLS, which is very slightly worse.

It's also instructive to consider what the noise looks like. By the time we get to, say, the 4000 trial OLS estimate, most of the jaggy high-frequency noise is gone; the really persistent errors are in the form of slow drifts away from the true ERP. This is

True ERP

rERP estimates

Trials	Data (min)	OLS	Filter GLS	Wavelet GLS
500	1.67	RMSE: 5.34 μV	RMSE: 3.50 μV	RMSE: 3.54 μV
1000	3.33	RMSE: 2.63 μV	RMSE: 2.22 μV	RMSE: 2.11 μV
2000	6.67	RMSE: 1.43 μV	RMSE: 1.36 μV	RMSE: 1.37 μV
4000	13.33	RMSE: 0.94 μV	RMSE: 0.90 μV	RMSE: 0.89 μV

Figure 5.10: Simulation 4: Using GLS to perform overlap correction under extreme conditions. We constructed simulated EEG data with SOAs randomly varying in the 150–250 ms range, then reconstructed the ERPs from this data using three different estimation techniques. Each simulation was run 100 times. 5 estimates are plotted here to visualize the sampling error, and the root-mean-squared-error (RMSE) for each set was calculated against the true ERP.

because of the $1/f$ character of the EEG noise: the strongest noise is the low frequency noise, so it persists the longest. This is why some kind of detrending (e.g., in the simplest case, baselining) is so important for EEG — because the drift, the low frequency noise, is huge. We all know that when the whole EEG signal drifts off in some direction this isn't really telling us much about the brain response to any particular event; it's not that we should ignore it — it might in fact reflect a real, slow wave brain potential — but it's just as likely that the EEG is just drifting randomly because that's what it does. Since these drifts are huge, standard least-squares pays much more attention to them than it should. This is where GLS's advantage comes from: it downweights the influence of low-frequency noise exactly enough to counteract OLS's tendency to overweight it.

In fact, part of the reason why GLS's advantage over OLS in these examples is relatively modest is that we baselined all of the rERPs after estimating them. This helped the OLS estimates much more than it helped the GLS ones, basically because it's replicating some of the detrending that the GLS estimators are doing in a more principled way.

5.3.5 The linear hypothesis test

Our second and primary motivation for investigating GLS estimation is for the statistical tests which it may enable. In particular, we consider the *linear hypothesis test* (Fox, 2008), a standard but not necessarily well-known method. Its advantage over other significance tests, such as the better known t test or ANOVA-style F tests, is that it allows both simple and complex hypotheses to be tested in a uniform manner. Any null hypothesis which can be expressed as a set of linear constraints on β coefficients in a single regression model can be tested. The result is an ordinary F value, with numerator degrees of freedom equal to the number of linear constraints included in your hypothesis, and denominator degrees of freedom equal to the usual residual degrees of freedom in your model (i.e., the number of data points minus the number of βs).

With the combined-regression rERP/GLS estimation approach, *all* our rERP values across different latencies and predictors are β coefficients within a single regression model,[6] which makes the linear hypothesis test a very flexible tool.

[6] At least, those measured at a single electrode. There are also multivariate versions of the linear

Here are some examples of different null hypotheses that we can test using this method:

$$\beta_{150\text{ ms}} = 0$$

In English: "Is the point rERP amplitude at 150 ms significantly different from zero?" This is effectively a t test for this coefficient; indeed, the F value we get out is just the square of the t value we would get if we ran this test in the traditional way.

$$\frac{1}{26}\left(\beta_{200\text{ ms}} + \beta_{204\text{ ms}} + \cdots + \beta_{300\text{ ms}}\right) = 0$$

In English: "Is the mean rERP amplitude in the 200–300 ms window significantly different from zero?"

$$\beta_{300\text{ ms}} = \frac{1}{26}\left(\beta_{-100\text{ ms}} + \beta_{-96\text{ ms}} + \cdots + \beta_{-4\text{ ms}} + \beta_{0\text{ ms}}\right)$$

In English: "Is the point rERP amplitude at 300 ms significantly different from the mean amplitude observed during the baseline period?"

$$\frac{1}{25}\left(\beta_{300\text{ ms}} + \beta_{304\text{ ms}} + \cdots + \beta_{396\text{ ms}}\right) = \frac{1}{25}\left(\beta_{400\text{ ms}} + \beta_{404\text{ ms}} + \cdots + \beta_{496\text{ ms}}\right)$$

In English: "Does the average rERP amplitude change significantly between the 300–400 ms window and the 400–500 ms window?"

$$\beta_{\text{factor A, 400 ms}} = \beta_{\text{factor B, 400 ms}}$$

In English: "Do factor A and factor B have different effects at 400 ms?"

We aren't restricted to writing single equations, either; we can write a system of

hypothesis test that might be useful for testing hypotheses across multiple electrodes simultaneously. However, these would require that we use a single S matrix for all electrodes, which seems unlikely to work well since different places on the scalp exhibit have rather different spectral distributions (for instance, consider the topographic distribution of alpha power).

equations, which produces an omnibus test:

$$\begin{cases} \beta_{400\text{ ms}} = 0 \\ \beta_{404\text{ ms}} = 0 \\ \quad\vdots \\ \beta_{600\text{ ms}} = 0 \end{cases}$$

In English: "Does the rERP amplitude differ significantly from zero at any point between 400 and 600 ms?" This is exactly analogous to a test for a main effect in ANOVA. (In fact, if we instead wrote a set of equations here stating that the means of all our groups were the same, then the linear hypothesis test would give us the same F value and degrees of freedom as we would get when running an ANOVA the normal way.) Like a test for a main effect, this can be significant even if none of the individual tests are; this happens when there is moderate evidence for multiple effects, so that we don't have enough power to identify any of them specifically, but can still be fairly certain that at least one of them is real.

Perhaps the single most useful test for ERP work, then, is:

$$\begin{cases} \beta_{400\text{ ms}} = \frac{1}{26}\left(\beta_{-100\text{ ms}} + \beta_{-96\text{ ms}} + \cdots + \beta_{-4\text{ ms}} + \beta_{0\text{ ms}}\right) \\ \beta_{404\text{ ms}} = \frac{1}{26}\left(\beta_{-100\text{ ms}} + \beta_{-96\text{ ms}} + \cdots + \beta_{-4\text{ ms}} + \beta_{0\text{ ms}}\right) \\ \quad\vdots \\ \beta_{600\text{ ms}} = \frac{1}{26}\left(\beta_{-100\text{ ms}} + \beta_{-96\text{ ms}} + \cdots + \beta_{-4\text{ ms}} + \beta_{0\text{ ms}}\right) \end{cases}$$

In English: "Does the rERP amplitude differ significantly from the mean baseline amplitude at any point between 400 and 600 ms?" This corresponds somewhat to the common approach of calculating the average amplitude over the 400–600 ms window (after baselining) and entering this into an ANOVA. We do that, but it's not really the average amplitude we're usually interested in. Really, what we want to know is whether there's anything happening there at all, and we average because it's a convenient way to pool power across adjacent points while still being able to use available statistical tools. This omnibus test also allows us to pool power across adjacent points, but is able also

to detect patterns like a positivity in the 400–500 ms window followed by a negativity in the 500–600 ms window, which together sum to a zero mean amplitude over the 400–600 ms window.

Of course, normally if we saw that pattern of a positivity followed by a negativity in our ERP, then we'd pick a different window. But this is (a) statistically speaking, a sin (Kriegeskorte, Simmons, Bellgowan, & Baker, 2009), and (b) impractical when running large scale automated tests. So this is the form of the linear hypothesis test that seems most generally useful.

But what if the comparison we want to make is not easily expressed as a single β coefficient? For example, we discussed a coding scheme in the previous chapter for an experiment containing two categorical factors, standard vs. target and red vs. blue. Suppose we want to know whether our targets show an effect of color. Recall that according to the scheme we used, a red target was coded with a combination of predictor values like:

$$x_{1i} = 1 \qquad x_{2i} = 1 \qquad x_{3i} = 0 \qquad x_{4i} = 0$$

while a blue target was

$$x_{1i} = 1 \qquad x_{2i} = 0 \qquad x_{3i} = 1 \qquad x_{4i} = 1$$

So there is no single β that reflects the comparison we want to make. One solution would be to figure out some other coding scheme in which such a single β would exist, so that we could use a t test. But an easier approach would be to take advantage of the linear hypothesis test. In this model, the predicted ERP for red targets is $\beta_1 + \beta_2$, and for blue targets is $\beta_1 + \beta_3 + \beta_4$, so if we want to know if these differ at, say, 300 ms, we can test with the null

$$\beta_{1at300ms} + \beta_{2 \text{ at 300 ms}} = \beta_{1 \text{ at 300 ms}} + \beta_{3 \text{ at 300 ms}} + \beta_{4 \text{ at 300 ms}}$$

The resulting F value will again be the square of the t value we would have gotten by

recoding our predictors.

5.3.6 Simulating the linear hypothesis test

So: does it work? To find out, we analyzed simulated EEG recordings using OLS, filter-based GLS, and wavelet-based GLS, and then tested the following hypothesis:

$$\begin{cases} \beta_{500\text{ ms}} = \frac{1}{25}\left(\beta_{-100\text{ ms}} + \beta_{-96\text{ ms}} + \cdots + \beta_{-4\text{ ms}}\right) \\ \beta_{504\text{ ms}} = \frac{1}{25}\left(\beta_{-100\text{ ms}} + \beta_{-96\text{ ms}} + \cdots + \beta_{-4\text{ ms}}\right) \\ \qquad \vdots \\ \beta_{700\text{ ms}} = \frac{1}{25}\left(\beta_{-100\text{ ms}} + \beta_{-96\text{ ms}} + \cdots + \beta_{-4\text{ ms}}\right) \end{cases}$$

The simulated EEG was created using the same method as Simulation 4, with 2000 trials and SOAs randomly varying between 150–250 ms. In Simulation 5, however, we left out the actual ERP and included only background noise; i.e., the null hypothesis was true. Under these conditions, a properly calibrated statistical test should produce p values which are distributed uniformly between zero and one; Figure 5.11 shows what we actually got.

The OLS results, as expected, deviate sharply from the ideal. They're conservative, at least, which is better than being anti-conservative — but in other tests (not shown) we've seen the OLS regressions give results that are equally skewed the other way, or where all p values are either < 0.01 or > 0.99 with nothing in between. It seems unwise to put much trust in them in general. The GLS results come much closer to the ideal; the wavelet GLS is somewhat anti-conservative in this test, and the filter GLS results are perfect.

In Simulation 6, we repeated the above test, but this time with a real effect present. We used the same true ERP as shown in Figure 5.10, which has a positivity in the 500–700 ms window which we tested. (This simulation thus corresponds exactly to the 3rd line of that figure.) Our results are shown in Figure 5.12. All of the hypothesis tests shown here had excellent power to detect this effect. In particular the filter-based GLS, our most trustworthy test from the previous simulation, produced a $p < 0.05$ on 94.8% of runs, and $p < 10^{-4}$ on 32.6%.

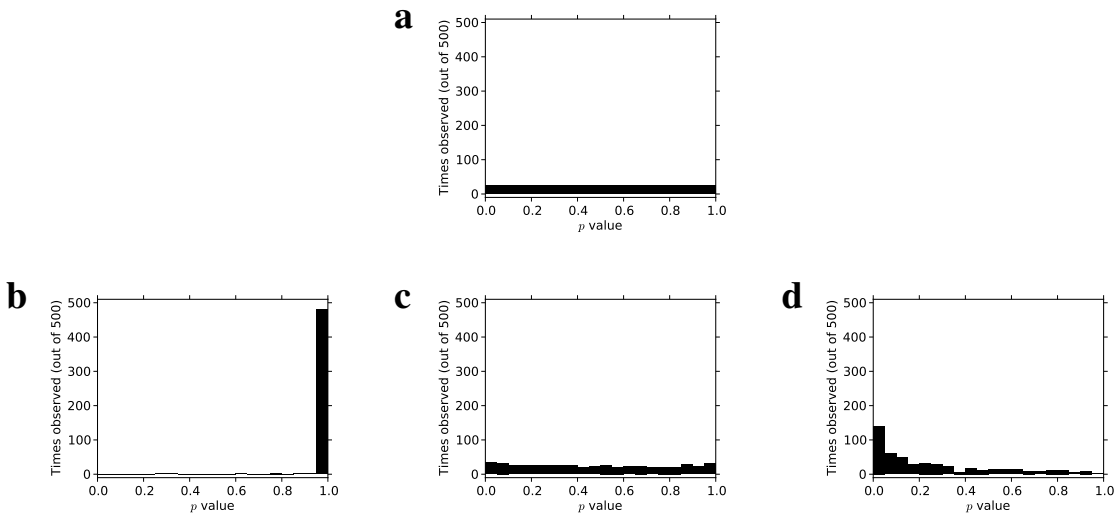

Figure 5.11: Simulation 5: Distribution of p values in a linear hypothesis test on simulated data, where the null was true. (**a**) The ideal distribution — 5% of p-values are less than 0.05, 1% are less than 0.01, etc. (**b**) The results from simple OLS fitting, i.e., pretending that the EEG background noise is actually white noise. (**c**) Filter-based GLS fitting. (**d**) Wavelet-based GLS fitting.

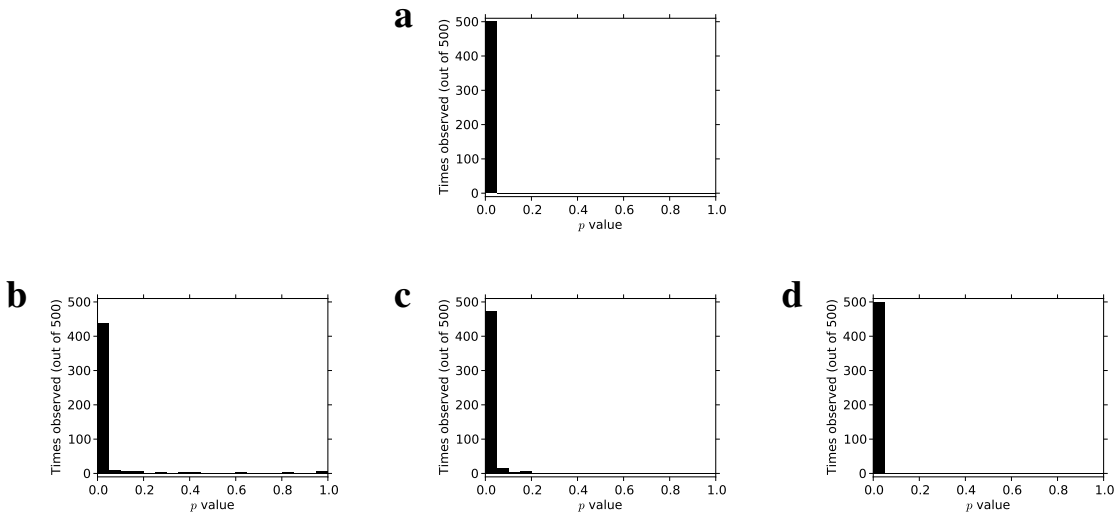

Figure 5.12: Simulation 6: Distribution of p values in a linear hypothesis test on simulated data, where an effect existed (the positivity in the 500–700 ms range shown in Figure 5.10). (**a**) The ideal distribution — a sufficiently powerful test would hope to detect a significant effect on every trial. (**b**) The results from simple OLS fitting, i.e., pretending that the EEG background noise is actually white noise. (**c**) Filter-based GLS fitting. (**d**) Wavelet-based GLS fitting.

These results, however, are quite preliminary. The success of the filter-based GLS in particular should be taken with a grain of salt, since our method for simulating EEG (phase shuffling) produces a noise signal with a fixed spectrum and stationary distribution — which is exactly what the filter-base GLS assumes about the data. Real EEG may prove less congenial, and the anti-conservativity demonstrated by the wavelet technique, while mild, suggests that these tests may be somewhat sensitive to mild mispecification. Nevertheless, we find these initial results promising; as these techniques undergo more rigorous testing then further modifications may be required, but the basic theory and techniques developed here should put such work on a solid foundation.

5.4 Acknowledgements

Chapters 4 and 5, in full, are currently being prepared for submission for publication of the material. The dissertation author was the primary investigator and author of this material.

5.A GLS for orthogonal designs

Suppose that we have a collection of epochs (as opposed to long segments of continuous data), and that we believe that they are widely separated enough that data in one epoch is uncorrelated with data in other epochs. (Or, at least, we are willing to make this assumption for purposes of GLS analysis.) Furthermore, suppose that our predictors are orthogonal. Then, we claim, the GLS regression estimate will coincide with the OLS regression estimate. We show this here.

In a mild abuse of notation, let S denote the matrix which whitens *a single epoch*. (Previously, we used S to denote the matrix which whitened the entire data set in one go; for our current setup, the old S would have a block diagonal form composed of copies of the new S.) Also, let X_{ji} be the matrix of predictor values for factor j in epoch i.

Then we have

$$X = \begin{pmatrix} X_{11} & X_{21} & \cdots \\ X_{12} & X_{22} & \cdots \\ X_{13} & X_{23} & \\ \vdots & & \ddots \end{pmatrix} \qquad \underline{X} = \begin{pmatrix} SX_{11} & SX_{21} & \cdots \\ SX_{12} & SX_{22} & \cdots \\ SX_{13} & SX_{23} & \\ \vdots & & \ddots \end{pmatrix}$$

According to the standard regresson formula, $\hat{\boldsymbol{\beta}} = (X^T X)^{-1} X^T \boldsymbol{y}$. We begin by examining the first part of this formula. We have

$$\underline{X}^T \underline{X} = \begin{pmatrix} X_{11}^T S^T & X_{12}^T S^T & X_{13}^T S^T & \cdots \\ X_{21}^T S^T & X_{22}^T S^T & X_{23}^T S^T & \cdots \\ \vdots & & & \ddots \end{pmatrix} \begin{pmatrix} SX_{11} & SX_{21} & \cdots \\ SX_{12} & SX_{22} & \cdots \\ SX_{13} & SX_{23} & \\ \vdots & & \ddots \end{pmatrix}$$

$$= \begin{pmatrix} \sum_i X_{1i}^T S^T S X_{1i} & \sum_i X_{1i}^T S^T S X_{2i} & \cdots \\ \sum_i X_{2i}^T S^T S X_{1i} & \sum_i X_{2i}^T S^T S X_{2i} & \\ \vdots & & \ddots \end{pmatrix}$$

Because we are assuming no overlap, the X_{ji} matrices are scalar matrices, $X_{ji} = x_{ji} I$. Therefore we can simplify further:

$$= \begin{pmatrix} \left(\sum_i x_{1i} x_{1i}\right) S^T S & \left(\sum_i x_{1i} x_{2i}\right) S^T S & \cdots \\ \left(\sum_i x_{2i} x_{1i}\right) S^T S & \left(\sum_i x_{2i} x_{2i}\right) S^T S & \\ \vdots & & \ddots \end{pmatrix}$$

Because we are assuming that the design is orthogonal, we know that $\sum_i x_{ji} x_{ki} = 0$ for $j \neq k$. (This is the definition of orthogonality.) Therefore, this is a block diagonal

matrix; letting $c_j = \sum_i x_{ji}^2$, we have:

$$\underline{X}^T \underline{X} = \begin{pmatrix} c_1 S^T S & & \\ & c_2 S^T S & \\ & & \ddots \end{pmatrix}$$

and

$$\left(\underline{X}^T \underline{X}\right)^{-1} = \begin{pmatrix} \frac{1}{c_1}\left(S^T S\right)^{-1} & & \\ & \frac{1}{c_2}\left(S^T S\right)^{-1} & \\ & & \ddots \end{pmatrix} \qquad (*)$$

Turning to the other part of our regression formula, let $\boldsymbol{y}_i$ represent the data from epoch i. Then we have

$$\underline{X}^T \underline{\boldsymbol{y}} = \begin{pmatrix} X_{11}^T S^T & X_{12}^T S^T & X_{13}^T S^T & \dots \\ X_{21}^T S^T & X_{22}^T S^T & X_{23}^T S^T & \dots \\ \vdots & & & \ddots \end{pmatrix} \begin{pmatrix} S\boldsymbol{y}_1 \\ S\boldsymbol{y}_2 \\ \vdots \end{pmatrix}$$

$$= \begin{pmatrix} \sum_i X_{1i}^T S^T S \boldsymbol{y}_i \\ \sum_i X_{2i}^T S^T S \boldsymbol{y}_i \\ \vdots \end{pmatrix}$$

And, again using the fact that X_{ji} is diagonal,

$$= \begin{pmatrix} S^T S \sum_i X_{1i}^T \boldsymbol{y}_i \\ S^T S \sum_i X_{2i}^T \boldsymbol{y}_i \\ \vdots \end{pmatrix} \qquad (**)$$

When we multiply $(*)$ and $(**)$ together to produce the full regression formula $(\underline{X}^T \underline{X})^{-1} \underline{X}^T \boldsymbol{y}$, we see that the $(S^T S)^{-1}$ and $S^T S$ terms cancel out, and the rest of the formula is identical to what it would have been if they had never been there in the first place, i.e., if we had used OLS instead of GLS. However, the $(\underline{X}^T \underline{X})^{-1}$ is different

between the GLS and OLS models, which means that the statistical tests which rely on it will be different as well. If we do have an orthogonal design in which we wish to use GLS for statistical testing, then this suggests a possible optimization: we can perform the fit itself using OLS, but then calculate $(X^T S^T S X)^{-1}$ as the unscaled covariance matrix for use in our tests. However, perfectly orthogonal designs are rare (e.g., because of artifact rejection), and saving a few iterations of the FGLS algorithm may not be worth the bother in any case.

5.B Implementation considerations

Here we discuss several details of how we efficiently implement the regression models considered in this chapter. If we have k predictors for which we would like to compute rERPs, we want each of these waveforms to be n points long, and we have a total of m data points, then our X matrix will have $k \times n \times m$ entries. A somewhat complex but plausible experiment might have $k = 5$, $n = 275$, and $m = 45 \times 60 \times 250$ (i.e., 45 minutes of data at 250 Hz). This gives a matrix with almost one billion entries, requiring more than seven gigabytes of RAM in a straightforward implementation.

This memory usage can be reduced (and subsequent calculations made much faster) by storing X in a sparse format, but even this is not a panacea; if we use a whitening filter then this will remove a great deal of sparsity (though of the options considered, the wavelet transform does the least harm in this respect). However, there is no need to ever actually form the X matrix in memory.

To compute a least-squares regression, we need to compute $(X^T X)^{-1} X^T y$. This can be done without X, so long as we know $X^T X$ and $X^T y$. We may also wish to know the sum of squared residuals, but computing the residuals themselves requires using the full X matrix. Fortunately, this is also unnecessary. The sum of squared residuals can be written as

$$\begin{aligned}(\boldsymbol{y} - X\hat{\boldsymbol{\beta}})^T (\boldsymbol{y} - X\hat{\boldsymbol{\beta}} \\ = \boldsymbol{y}^T \boldsymbol{y} - 2\hat{\boldsymbol{\beta}}^T X^T \boldsymbol{y} + \hat{\boldsymbol{\beta}}^T X^T X \hat{\boldsymbol{\beta}}\end{aligned}$$

Notice that this can be computed using only the matrices above plus $\boldsymbol{y}^T\boldsymbol{y}$ (i.e., the sum of squared data). The largest of these matrices, X^TX, now has only $(k \times n)^2$ entries. In our example this comes to about 2 million, which fits easily into RAM on an average cell phone. We also need to save a count of how many data points we have in total.

To compute these matrices, we observe that if we have a block matrix

$$A = \begin{pmatrix} A_1 \\ A_2 \end{pmatrix}$$

then

$$A^TA = A_1^TA_1 + A_2^TA_2.$$

This means that we can compute some rows for X, compute their cross product, accumulate the result into a buffer, discard the rows, and repeat. In a single pass, we can construct everything we need to perform this regression, without ever loading X itself into memory. This approach is more difficult if we are using whitening transforms, since they need to act on more than one row of X at a time, but by computing some reasonable number of rows at a time and taking some care with the edge conditions then it can still be used. (Even an experiment using long streams of closely spaced stimuli will usually include a number of breaks. In this case we will generally treat each stimulus block as a separate data set for whitening, so the easiest approach is to just break up X at these boundaries.)

If you have access to an incremental QR or Cholesky algorithm, then they can also be used in this way, with a possible gain in numerical stability.

In the wavelet case, there are a few more tricks that can be used to speed things up even more. First, the `sym4` wavelet basis is orthogonal; i.e., if W is the matrix representing the DWT, then $W^TW = I$. This means that when doing our FGLS iteration, we don't have to first construct X, perform an OLS regression, and then construct $\underline{X}$ to perform the GLS regressions. We can define $\underline{X} = \text{DWT}(X), \underline{\boldsymbol{y}} = \text{DWT}(\boldsymbol{y})$ and use

these from the beginning, because

$$\begin{aligned}\hat{\boldsymbol{\beta}} &= (\underline{X}^T\underline{X})^{-1}\underline{X}^T\underline{\boldsymbol{y}} \\ &= (X^TW^TWX)^{-1}X^TW^TW\boldsymbol{y} \\ &= (X^TX)^{-1}X^T\boldsymbol{y}.\end{aligned}$$

That is, if all we do is apply the DWT, then we still get the exact same result we would have gotten from performing OLS with X itself. It's only after we start weighting the different decomposition scales to correct for heteroskedasticity that the OLS and GLS estimates begin to differ.

Furthermore, we can both estimate and update these weights without redoing any large matrix computations. Here, the trick is that instead of storing $\underline{X}^T\underline{X}$, $\underline{X}^T\underline{\boldsymbol{y}}$, $\underline{y}^T\underline{y}$, we store a separate version of each of these matrices for each decomposition scale. So if $\underline{X}_0$ is the matrix with the rows of $\underline{X}$ which contain approximation coefficients, $\underline{X}_1$ has the first set of detail coefficients, etc., then we store $\underline{X}_0^T\underline{X}_0$, $\underline{X}_1^T\underline{X}_1$, etc. Notice that these matrices together sum to $\underline{X}^T\underline{X}$, and likewise for the other matrices we need. If the longest contiguous stream of data which we analyze contains l samples (and as mentioned above, this will usually be a small fraction of the total data gathered in an experiment), then this increases our memory storage by a factor of about $\log_2 l$. Even if we sampled at 250 Hz for an hour straight, this would still only come to an increase of about 20 times, requiring about 40 million entries for the X^TX matrices in our example above — beginning to push the limits of a cell-phone, perhaps, but still comfortably within the range of an ordinary computer.

Having split up our matrices in this way, we can easily reweight the coefficients for each decomposition scale. We construct $\underline{X}^T\underline{X}$ for our regression by taking the weighted sum of the individual $\underline{X}_i^T\underline{X}_i$ matrices, where the weights are the reciprocals of the estimated standard deviation of the noise at each level. Then, we need to look to the residuals to re-estimate these standard deviations for the next iteration.

The sum of squared residuals for the 3rd level of detail coefficients, say, is

$$\underline{\boldsymbol{y}}_3^T\underline{\boldsymbol{y}}_3 - 2\hat{\boldsymbol{\beta}}^T\underline{X}_3^T\underline{\boldsymbol{y}}_3 + \hat{\boldsymbol{\beta}}^T\underline{X}_3^T\underline{X}_3\hat{\boldsymbol{\beta}}.$$

This and a count of the number of rows in $\underline{X}_3$ is enough to let us compute the sample standard deviation for these residuals.

The end result is that in the wavelet case, it's only the initial pass through the data constructing X that takes much time; the actual fitting process is extremely fast.

References

Aitken, A. C. (1935). On least squares and linear combinations of observations. *Proceedings of the Royal Society of Edinburgh*, *55*, 42–48.

Burock, M. A., Buckner, R. L., Woldorff, M. G., Rosen, B. R., & Dale, A. M. (1998). Randomized event-related experimental designs allow for extremely rapid presentation rates using functional MRI. *NeuroReport*, *9*, 3735–3739.

Fadili, M. J., & Bullmore, E. T. (2002). Wavelet-generalized least squares: a new BLU estimator of linear regression models with $1/f$ errors. *NeuroImage*, *15*(1), 217–232.

Fox, J. (2008). *Applied regression analysis and generalized linear models* (2nd ed.). Thousand Oaks, California: SAGE.

Groppe, D. M., Urbach, T. P., & Kutas, M. (in press). Mass univariate analysis of event-related brain potentials/fields i: A critical tutorial review. *Psychophysiology*.

Jones, E., Oliphant, T., Peterson, P., et al. (2011). *SciPy: Open source scientific tools for Python* [Computer software]. Retrieved from http://www.scipy.org/

Kriegeskorte, N., Simmons, W. K., Bellgowan, P. S. F., & Baker, C. I. (2009). Circular analysis in systems neuroscience: the dangers of double dipping. *Nature Neuroscience*, *12*(5), 535–540.

Kutas, M., Lindamood, T. E., & Hillyard, S. A. (1984). Word expectancy and event-related brain potentials during sentence processing. In S. Kornblum & J. Requin (Eds.), *Preparatory states and processes* (pp. 217–237). Hillsdale, N. J.: Lawrence Erlbaum.

Lahiri, S. N. (2003). *Resampling methods for dependent data*. New York: Springer-Verlag.

Luck, S. J. (2005). *An introduction to the event-related potential technique*. Cambridge, MA: MIT Press.

Makeig, S., Bell, A. J., Jung, T.-P., & Sejnowski, T. J. (1996). Independent component analysis of electroencephalographic data. In *Advances in neural information processing systems 8.* Cambridge, MA: MIT Press.

Mathworks. (2011). *Matlab signal processing toolbox* [Computer software].

McCullagh, P., & Nelder, J. (1989). *Generalized linear models*.

Smith, S. W. (1997). *The scientist and engineer's guide to digital signal processing*. San Diego, CA: California Technical Publishing. Retrieved from http://www.dspguide.com/

Tewfik, A. H., & Kim, M. (1992). Correlation structure of the discrete wavelet coefficients of fractional brownian motion. *Information Theory, IEEE Transactions on*, *38*(2), 904–909.

Urbach, T. P., & Kutas, M. (2008). Cognitive aging: Evidence against a single factor. *Psychophysiology*, *45*(S1), S113.

Woldorff, M. G. (1988). Adjacent response overlap during the erp averaging process and a technique (Adjar) for its estimation and removal. *Psychophysiology*, *25*, 490.

Woldorff, M. G. (1993). Distortion of ERP averages due to overlap from temporally adjacent ERPs: analysis and correction. *Psychophysiology*, *30*(1), 98–119.

Wornell, G. W. (1990). A Karhunen-Loeve-like expansion for $1/f$ processes via wavelets. *Information Theory, IEEE Transactions on*, *36*(4), 859–861.

Chapter 6

Conclusion

In this dissertation we've considered several questions. Why can brains process things faster, just because they're more predictable? How do they make these predictions? And, having seen the benefits that accrued from the use of rich regression models in chapters 2 and 3, how can we extend these to ERP analysis while respecting its unique challenges?

In the introduction, I argued that a particular combination of approaches, — the use of large data sets, analytical tools that reveal patterns instead of just testing hypotheses, and a theoretical stance that allows for mechanism-neutral hypotheses — is particularly useful for 'scaling up' psycholinguistics; none of these ideas are new, but I believe they are underutilized. By now, I hope I've also convinced you of their merit in shedding light on issues that would otherwise be difficult or impossible to approach (e.g., measuring the relationship between predictability and reading time, or detecting our cloze participants' habit of ignoring part of their prompt), and, particularly in chapter 4, shown how this point of view can lead to methods which address a variety of common, practical problems.