2 The Event-Related Potential from a Scalp Electroencephalogram

Synopsis

Data 1 s of scalp EEG data sampled at 500 Hz during 1,000 trials in two conditions.

Goal Characterize the response of the EEG in the two conditions.

Tools Visualization, event-related potential, confidence intervals, bootstrapping.

2.1 Introduction

2.1.1 Background

Voltage recordings from the scalp surface—the electroencephalogram (EEG)—provide a powerful window into brain voltage activity. Some of the earliest human EEG recording occurred in 1924, when Dr. Hans Berger made a remarkable discovery: the EEG of a human subject at rest with eyes closed exhibits rhythmic activity, an approximately 10 Hz oscillation that he labeled the *alpha rhythm* [3]. Although now studied for nearly 100 years, the definitive functional role (if any) of the alpha rhythm remains unknown. Since then, many other EEG rhythms have been detected and labeled (typically with Greek letters), and the analysis of EEG rhythms remains an active area of research [4].

Compared to other modalities for measuring brain activity, the EEG possesses both advantages and disadvantages. Perhaps the most important advantages are that the EEG is noninvasive and permits a high temporal resolution (on the order of milliseconds). But the EEG measure also suffers from significant disadvantages, most notably, poor spatial resolution: a single scalp electrode detects the summed activity from approximately 10 cm^2 of cortex [5].

In this chapter, we consider EEG data recorded from a single scalp electrode. We analyze these data to determine what (if any) activity is evoked following two different types of stimuli presented to a human subject. In doing so, we continue demonstrating MATLAB

features and how this powerful tool can help us understand these time series data. We begin with a brief description of the EEG data.

2.1.2 Case Study Data

An undergraduate student volunteers to participate in a psychology study at his university. In this study, EEG electrodes (sampling rate 500 Hz, i.e., 500 samples per second) are placed on the student's scalp, and he is set in a comfortable chair in a dark, electrically isolated room. Upon entering the room, the student is instructed to place headphones over his ears and listen to a series of repeated sounds. The sounds consist of two tones: either a high-pitch tone or a low-pitch tone. A single tone is presented once every few seconds, and the student responds with a button press to the low-pitch tone. The tone presentation is repeated to collect the EEG response to numerous presentations of the two tones (figure 2.1). Our collaborator leading this research study has agreed to provide us with EEG data recorded at a single electrode for 1,000 presentations of the high-pitch tone and 1,000 presentations of the low-pitch tone. In each presentation, or trial, she provides us with 1 s of EEG data, such that the tone occurs at 0.25 s into the trial (gray-shaded regions in figure 2.1). She asks us to analyze these data to determine whether the EEG signal differs following the two tone presentations.

2.1.3 Goal

Our goal is to analyze the collection of EEG data recorded in the two conditions. We would like to understand the EEG features evoked by the stimulus, and how these features differ in the two conditions.

2.1.4 Tools

We develop two primary tools: visualization techniques and the *event-related potential*. In computing the latter, we consider two techniques to create confidence intervals for the ERP, one relying on the central limit theorem, the other a bootstrap technique. We also discuss how to implement a bootstrap test to assess the difference in response between the two conditions.

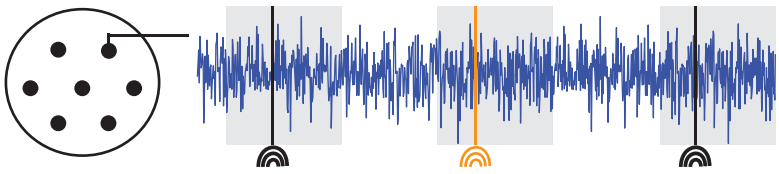

Figure 2.1

Cartoon illustration of EEG experiment. *Left*, EEG electrodes are placed on scalp surface of a human subject. *Right*, EEG activity (*blue*) recorded as a function of time during presentation of high-pitch tones (black) and low-pitch tones (*orange*).

2.2 Data Analysis

To access the EEG data for this chapter, visit

`http://github.com/Mark-Kramer/Case-Studies-Kramer-Eden`

and download the file `Ch2-EEG-1.mat`.

2.2.1 Visual Inspection

The best place to begin the data analysis is with visual inspection of the time series. The instructions to perform this initial analysis are simple: plot the data and look at them. Of course, before we can visualize the data, we must load them into MATLAB. To do so, we use the function load:

```
load('Ch2-EEG-1.mat')    %Load the EEG data.
```

where the file `Ch2-EEG-1.mat` resides in the current MATLAB directory.[1] To understand the outcome of issuing this command, let's examine the variables now present in the MATLAB workspace by asking `who`:

```
who                      %List variables in the workspace.
```

The `who` command lists all the variables that currently reside in the MATLAB workspace. We find there are three: `EEGa`, `EEGb`, and `t`. These correspond to the EEG data recorded in the two conditions (`EEGa` to condition A and `EEGb` to condition B) as well as a time axis (`t`). Let's further investigate the structure of the EEG variables by determining their size:

```
size(EEGa)               %Determine the dimensions of EEGa.
```

Upon executing this command, MATLAB returns the value `[1000, 500]`. The variable `EEGa` is a *matrix* with 1,000 rows and 500 columns. Our collaborator tells us that each row corresponds to a separate trial, and each column to a point in time. So, there are 1,000 total trials, each consisting of 500 time points. As a matter of convenience, we define a new variable to record the number of trials:

```
ntrials = size(EEGa,1); %Define variable to record no. of trials.
```

Notice that in using the `size` function, we supply two inputs; the second input requests the size of the first dimension (i.e., the number of trials). This variable will be useful later.

Q: Determine the size of the variable `EEGb`. How many rows and columns does it possess? Which dimension of the matrix corresponds to trials, which dimension to time?

1. An alternative approach to loading the data: the file `Ch2-EEG-1.mat` may be dragged from a folder on your computer and dropped into the MATLAB command window.

Both `EEGb` and `EEGa` are complicated variables that contain many elements. To understand these data, we might attempt to read the values contained in each element. For example, we can print to the MATLAB screen the EEG data for the first trial of condition A.

```
EEGa(1,:)        %Print to screen data from condition A, trial 1.
```

In this command, we index the first row of the matrix `EEGa` and print out all columns (corresponding to all moments of time).

Q: Upon issuing this command, what do you find? Does the printout help you understand these data?

A: You should observe a list of 500 numbers that begins

```
-0.1859 0.4499 1.0607 -0.4713 1.6867 0.9382 0.2212...
```

We might conclude that these numbers exhibit variability (i.e., the values are both positive and negative), but examining the data in this way is not particularly useful. For example, determining trends in the behavior (such as intervals of repeated activity) through inspection of these printed numbers alone is extremely difficult.

Printing out the data to the screen is not useful in this case. How else can we deepen our understanding of these data? Let's make a plot. Fortunately, plotting the data is easily done in MATLAB:

```
plot(EEGa(1,:))      %Plot the data from condition A, trial 1.
```

The results of this command are shown in figure 2.2a. Visualizing the data in this way, we immediately notice many features. First, let's consider the axes. The horizontal axis extends from 1 to 500. This corresponds to the 500 columns in the variable `EEGa`. It would be more informative to plot the EEG data as a function of time rather than indices. Fortunately, we possess a variable `t` in the workspace that corresponds to the time axis. Determining the `size` of the variable `t`, we find it is a vector with 1 row and 500 columns. Each column corresponds to a point in time. Plotting the variable `t` (figure 2.2b), we see it ranges from 0 to 1. This corresponds to the 1 s of EEG data recorded in each trial. We can also use the variable `t` to determine the *sampling interval*,

```
dt = t(2) - t(1);        %Determine the sampling interval.
```

The new variable `dt` corresponds to the time between samples.

Q: What is the value of `dt`? We were told by our collaborator that the sampling frequency is 500 Hz. Is the value of `dt` consistent with this sampling frequency?

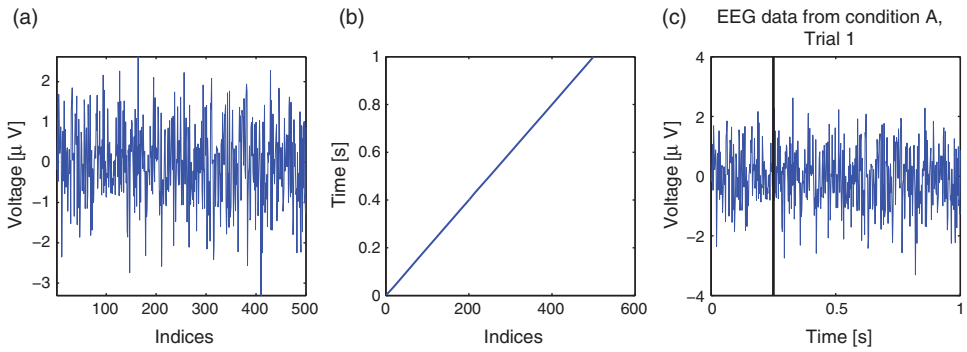

Figure 2.2

Initial exploration of the variables received. (*a*) Plot of EEG data from condition A in first trial. (*b*) Plot of variable `t`. (*c*) Plot of EEG data from condition A in first trial with appropriate axes, labels, and a vertical line indicating time of stimulus presentation.

A: Yes, it is consistent. We find that `dt` is 0.002 s, or 2 ms. The sampling frequency of 500 Hz corresponds to one sample of the EEG data every 1/(500 Hz) = 2 ms. If the two were not consistent, we would return to our collaborator and figure out what has gone wrong. In general, it's useful to ask such questions along the way to make sure we understand the formatting of the data and catch any potentially serious misunderstandings early in the analysis.

We can now combine the time axis with the EEG data to make a more complete plot. Let's also label the axes and give the plot a title.

```
plot(t,EEGa(1,:))          %Plot condition A, trial 1 data vs t.
xlabel('Time [s]')         %Label the x-axis as time.
ylabel('Voltage [\mu V]')  %Label the y-axis as voltage.
title('EEG data from condition A, Trial 1')    %Add a title.
```

This plot (figure 2.2c) provides a nice summary of the data in the first trial of condition A. A visual inspection of the plot suggests that these data exhibit complicated activity. We know from our collaborator that the stimulus occurs at time 0.25 s in each trial. This time is indicated in figure 2.2c as a vertical line. To add this vertical line to the figure, we first instruct MATLAB to freeze the plot (to not erase the current plot contents) using the `hold on` command:

```
hold on     %Freeze plot & indicate stimulus delivery time ...
plot([0.25, 0.25], [-4,4], 'k', 'LineWidth', 2)
hold off    %Release the current plot.
```

This `plot` command includes additional options that make the line black (`'k'`) and a bit wider (`'LineWidth', 2`). We end this code with `hold off` to release the plot; the next issuance of the `plot` command will erase the current plot contents. Releasing a plot is not always necessary but always polite.

Q: What else, if anything, can you say about the single trial of EEG data plotted in figure 2.2c? Does visual inspection reveal any particular change in the EEG activity following the stimulus presentation?

So far we have visualized only the data from condition A. Because we are interested in whether the EEG behaves differently in the two conditions, visualizing both conditions simultaneously would be of use. Using the `hold on` command, we can do this in MATLAB:

```
plot(t,EEGa(1,:))       %Plot condition A, trial 1, data vs t,
hold on                 %... freeze the plot, and then plot
plot(t,EEGb(1,:), 'r')  %... data from condition B, trial 1,
hold off                %... and release the plot.
xlabel('Time [s]')          %Label the x-axis as time.
ylabel('Voltage [\mu V]')   %Label the y-axis as voltage.
title('EEG data from conditions A (blue) and B (red), Trial 1')
```

Q: Compare the voltage traces from the first trial of conditions A and B as plotted in figure 2.3. What similarities and differences do you observe?

Q: The analysis has so far focused only on the first trial. Repeat this visual inspection of the data for different trials. What do you find? What similarities and differences exist between the two conditions across trials?

These techniques allow us to visualize the data one trial at a time. That is useful but can be time consuming, especially for a large number of trials. For the EEG data of interest here, each condition contains 1,000 trials, and to visualize each trial separately could require 2,000 plots. We can certainly create 2,000 plots, but the subsequent visual inspection would be time consuming and difficult. Fortunately, a more efficient visualization approach exists: we can display the entire structure of the data across both time and trials as an *image*. Doing so is straightforward in MATLAB:

```
imagesc(t,(1:ntrials),EEGa);    %Image the data from condition A.
xlabel('Time [s]')              %Label the x-axis.
```

```
ylabel('Trial #')                    %Label the y-axis.
hold on                              %Indicate time of stimulus onset.
plot([0.25, 0.25], [0 1000], 'k', 'LineWidth', 2)
hold off
```

The `imagesc` command allows us to visualize the entire matrix `EEGa` as a function of trial number and time (figure 2.4). Each row corresponds to a single trial of duration 1 s, and the color indicates the voltage, with warm (cool) colors indicating higher (lower) voltages.[2] This plot also indicates the time of stimulus presentation with a vertical black line as a cue to assist visual inspection.

Q: Upon close inspection of figure 2.4, what response, if any, do you observe following the stimulus presentation? (Look *really* carefully.) Repeat this visualization and analysis for `EEGb`. How do the two conditions compare?

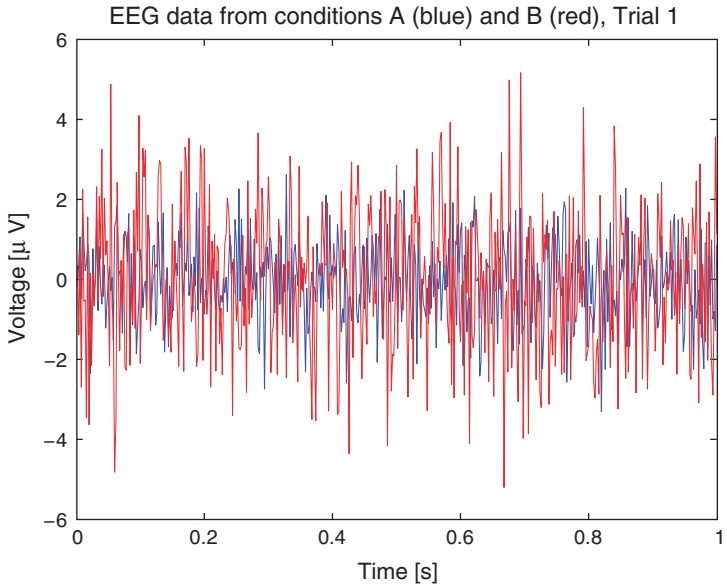

Figure 2.3
EEG data from first trial of condition A (*blue*) and condition B (*red*).

2. We have used the default color settings. There are many other options; check out `help colormap` for details.

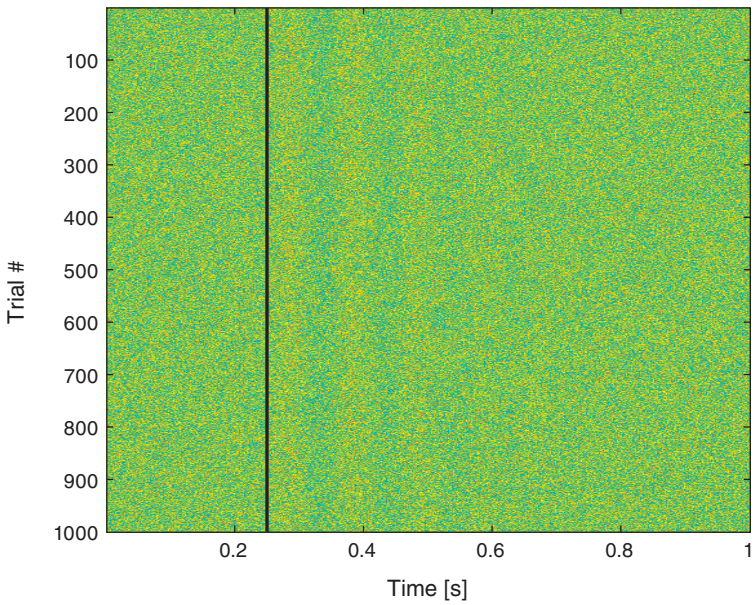

Figure 2.4

Image of EEG data from condition A for all trials. Time of stimulus presentation is indicated with a vertical black line.

2.2.2 Plotting the ERP

Visual inspection of the EEG data has so far come up empty. The EEG traces appear noisy or perhaps rhythmic, but from visual inspection of the individual trials it's difficult to make a decisive conclusion of underlying structure (e.g., figure 2.3). To further investigate the activity in these data, we compute the *event-related potential* (*ERP*). To compute the ERP, we first assume that each trial evokes an instantiation of the same underlying brain process. So, in this case, we assume that the same brain response is evoked 1,000 times (once for each trial) for each condition. However, the evoked response due to the stimulus is small and hidden in the EEG signal by other ongoing activity unrelated to the stimulus (e.g., daydreaming, thoughts of dinner, thoughts of homework). Therefore, to tease out the weak evoked effect, we average the EEG responses across trials. Ideally, EEG activity unrelated to the stimulus will cancel out in the average, while EEG activity evoked by the stimulus will sum constructively. The procedure to perform and display this averaging can be done in MATLAB:

```
load('Ch2-EEG-1.mat')        %Load the EEG data.
plot(t, mean(EEGa,1))        %Plot the ERP of condition A.
xlabel('Time [s]')           %Label the x-axis as time,
ylabel('Voltage [\mu V]')    %...and the y-axis as voltage,
title('ERP of condition A') %...and provide a useful title.
```

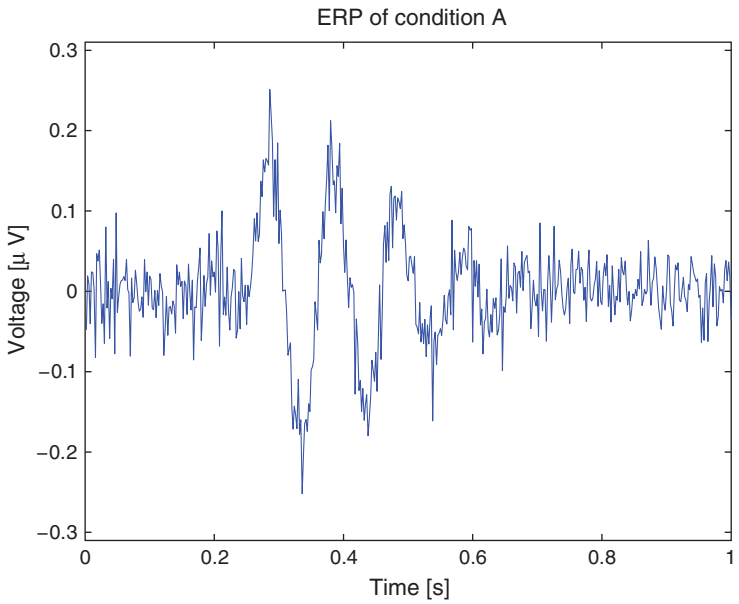

Figure 2.5
ERP for condition A computed by averaging EEG response over all trials.

Notice that in the second line, we use the MATLAB function `mean` to compute the mean of `EEGa` over the first dimension (the number of trials). The result is the ERP for condition A (figure 2.5).

Q: Consider the ERP for condition A plotted in figure 2.5. Update this figure to include a vertical line at the location of the stimulus, and the ERP for condition B. How, if at all, do the ERPs for Conditions A and B differ?

The ERP in figure 2.5 shows the mean voltage across trials at each moment in time. Visual inspection suggests that before stimulus presentation (i.e., times 0 s to 0.25 s) the EEG fluctuates around zero. Then, after stimulus presentation, the ERP increases and decreases substantially above and below zero. Which, if any, of these deviations following stimulation are significant? To address this, we make use of the trial structure of the EEG data to compute *confidence bounds* for the ERP. We do so in two ways.

2.2.3 Confidence Intervals for the ERP (Method 1)

To compute the ERP we average the EEG data across many trials. Because of this, we may make use of a powerful theorem in statistics—the *central limit theorem* (CLT)—to include approximate confidence bounds in the ERP figure. Briefly, this theorem states that the mean of a sufficiently large number of independent random variables, each with finite

mean and variance, will be approximately normally distributed. Remember that the ERP at each moment in time is the sum of EEG activity across trials (then scaled by a constant, the number of trials). Let's assume that the trials are independent (i.e., one trial does not depend on any other trial). Let's also assume that the EEG data at each moment in time have finite mean and variance. With those assumptions, we have satisfied the CLT and may therefore conclude that the ERP at each moment in time is approximately normally distributed.

Q: To use the CLT, we make two assumptions about the EEG data. Are these assumptions reasonable?

A: We assume that the EEG data are independent across trials. This assumption may fail if, for example, the activity in one trial influences the activity in the next trial. We also assume that the EEG data are "well-behaved" (i.e., have finite mean and variance). That is a reasonable assumption for physical data we observe from the brain; we expect the EEG data to always remain finite and not diverge to plus or minus infinity.

This conclusion—that the ERP at each moment in time is approximately normally distributed—is useful because the normal distribution (also known as the Gaussian distribution or bell curve) possesses many convenient properties. First, a normal distribution is relatively simple; it can be completely specified with two parameters: the mean value and the standard deviation (figure 2.6). Second, 95% of the values drawn from a normal distribution lie within approximately two standard deviations of the mean (figure 2.6).

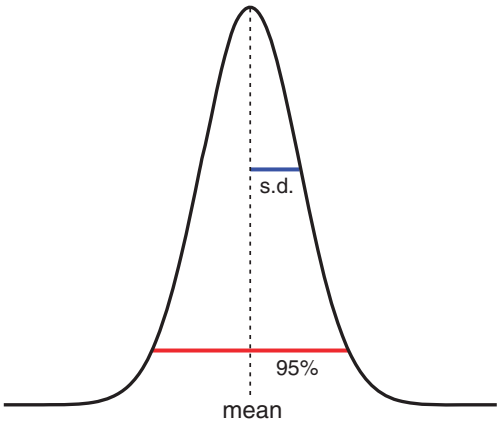

Figure 2.6

Canonical normal distribution showing mean (*dotted vertical line*) and standard deviation (*blue*). Ninety-five percent of values lie within the interval indicated by the red bar.

Therefore, to construct a 95% confidence interval for the ERP, we need to determine the mean and standard deviation of the mean across trials at each point in time.[3] To compute the mean in MATLAB is easy:

```
mn = mean(EEGa,1);        %Compute mean across trials (the ERP).
```

We again note that the second input to the function `mean` specifies the dimension in which we compute the mean, in this case, across the first dimension of the variable `EEGa` corresponding to the trials. To compute the standard deviation of the mean, we start by computing the standard deviation of the data:

```
sd = std(EEGa,1);         %Compute std across trials.
```

But we're not interested in the standard deviation of the EEG data across trials; instead, we're interested in the standard deviation of the estimate of the mean. To calculate the standard deviation of the mean, we divide the standard deviation of the data by the square root of the number of trials (i.e., the number of terms used to compute the mean; see the appendix at the end of this chapter). In MATLAB,

```
sdmn=sd/sqrt(ntrials);  %Compute the std of the mean.
```

Now, having found the mean and the standard deviation of the mean, we can compute a 95% confidence interval for the ERP. We again exploit the observation, based on the central limit theorem, that the ERP is normally distributed at each instant of time. With these calculations, the following MATLAB code plots the ERP and the 95% confidence interval:

```
plot(t, mn, 'LineWidth', 3)      %Plot the ERP of condition A.
hold on                          %Freeze the plot,
plot(t, mn+2*sdmn);              %... and include the upper CI,
plot(t, mn-2*sdmn);              %... and the lower CI.
hold off                         %Release the plot.
xlabel('Time [s]')               %Label the x-axis as time.
ylabel('Voltage [\mu V]')        %Label the y-axis as voltage.
title('ERP of condition A')      %Provide a useful title.
```

The ERP computed with confidence intervals allows us to ask specific questions about the data. For example, does the ERP ever differ significantly from zero? To answer this, we

3. We could instead write the *sample* mean because we use the observed data to estimate the theoretical mean that we would see if we kept repeating this experiment. This distinction is not essential to our goals here, but it is important when talking to your statistics-minded colleagues. Throughout the book, we omit the term *sample* when referring to sample means, variances, covariances, and so forth, unless this distinction is essential to the discussion.

look for intervals of the ERP for which the confidence intervals do not include zero. To aid visual inspection, we add to the ERP plot a horizontal line at 0:

```
hold on                          %Freeze the plot,
plot(t, zeros(size(mn)), 'k')    %... add horizontal line at 0,
hold off                         %... and release the plot.
```

Q: What is the role of the MATLAB function `zeros` in this code? *Hint*: If you have not encountered `zeros` before, look it up in MATLAB Help.

We find three time intervals at which the confidence intervals of the ERP do not include zero: near 0.27 s, near 0.37 s, and near 0.47 s (see figure 2.7). These results suggest that for an interval of time following the stimulus presentation in condition A, the observed ERP is not a random fluctuation about zero but instead contains consistent structure across trials.

Q: Construct the ERP with confidence intervals for condition B. As for condition A, you should find that before stimulus presentation the ERP fluctuates around zero. What intervals of time, if any, differ significantly from zero?

2.2.4 Comparing ERPs

In the previous section, we implemented a procedure to compute confidence intervals for the ERPs in conditions A and B. To investigate *differences* between the ERPs in the two

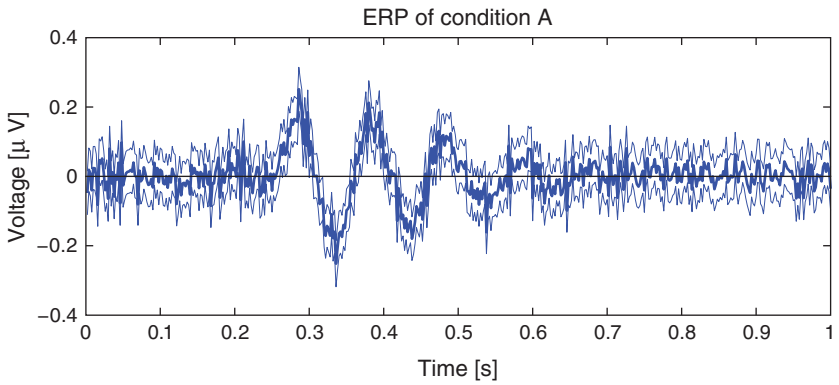

Figure 2.7

The ERP for condition A with 95% confidence intervals. The thick line indicates the ERP (i.e., the mean of the EEG across trials) while the thin lines indicate the 95% confidence intervals.

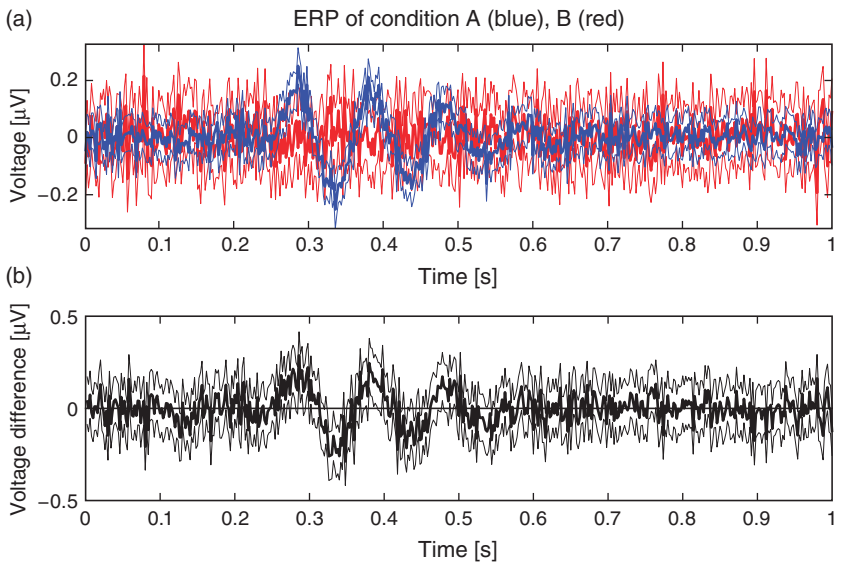

Figure 2.8

(*a*) ERPs with confidence intervals for condition A (*blue*) and condition B (*red*). (*b*) The differenced ERP. Thick lines indicate the mean; 95% of values lie between the thin lines.

conditions, we can use a similar approach. To start, let's plot the ERPs with confidence intervals for both conditions and attempt to identify periods for which the confidence intervals do not overlap (such intervals would correspond to significant differences between the responses of the two conditions). The plot of both ERPs (figure 2.8a) is rather messy; it's difficult to determine through visual inspection alone in which intervals the ERPs exhibit significant separation.

To facilitate further inspection of the data, we compute the *difference* between the ERPs in the two conditions. In the differenced signal, large deviations between the two conditions will appear as large differences from zero. To determine whether a deviation is significantly different from zero, we need to determine the confidence interval for the differenced ERP. This requires we propagate the standard deviation of the mean for both ERPs to the new differenced ERP. The propagated standard deviation of the mean at a fixed moment in time is computed as

$$\sigma = \sqrt{\frac{\sigma_A^2}{K} + \frac{\sigma_B^2}{K}}, \tag{2.1}$$

where σ_A is the standard deviation of condition A, σ_B is the standard deviation of condition B, and K is the number of trials. In MATLAB we compute the differenced ERP and

standard deviation of the mean of this difference as follows:

```
mnA=mean(EEGa,1);                %ERP of condition A,
sdmnA=std(EEGa,1)/sqrt(ntrials);%...and standard dev of mean.
mnB=mean(EEGb,1);                %ERP of condition B,
sdmnB=std(EEGb,1)/sqrt(ntrials);%...and standard dev of mean.
mnD=mnA-mnB;                     %The differenced ERP,
sdmnD=sqrt(sdmnA.^2 + sdmnB.^2);%...and its standard dev.
```

In this code we first compute the ERP and standard deviation of the mean for each condition. We then compute the differenced ERP (`mnD`) and the standard deviation of the mean of this difference (`sdmnD`) using equation (2.1). We note that $\texttt{sdmnA} = \sqrt{\sigma_A^2/K}$ and therefore $\texttt{sdmnA.^2} = \sigma_A^2/K$, with similar expressions for condition B. The resulting differenced ERP with 95% confidence intervals is shown in figure 2.8b. The hope is that from this figure we can more easily identify significant differences between the two conditions.

Q: Examine the differenced ERP in figure 2.8b. In what intervals of time do the EEG responses in the two conditions significantly differ?

2.2.5 Confidence Intervals for the ERP (Method 2)

So far we have computed confidence intervals for the ERPs by relying on the central limit theorem and approximating the average voltage values at each point in time as normally distributed. That's a completely reasonable approach. And because the normal distribution is so well-behaved, it's easy to compute the 95% confidence intervals. An alternative approach to generate confidence intervals is through a *bootstrap* procedure. Bootstrapping is a resampling method that allows us to estimate the sampling distribution of many different statistics. In this chapter, we implement a *nonparametric bootstrap*. To do so, we generate new *pseudodata* from the observed EEG data.[4] We begin by using a bootstrapping procedure to create confidence intervals for the ERPs observed in each condition.

We implement the bootstrapping procedure to compute pointwise confidence intervals. By pointwise we mean that the confidence intervals are computed separately for each point in time, and interactions across time are not considered. The prescription for the bootstrapping procedure follows four steps:

4. Briefly, there is strong theoretical justification for the nonparametric bootstrap. The fundamental idea is that resampling the data with replacement is equivalent to sampling new pseudodata from the empirical cumulative distribution function (eCDF) of the observed data. For a large sample of independent, identically distributed random variables, the distribution of the pseudodata generated from the eCDF will be close to the true distribution of the data. Note the important caveat that the variables are independent, identically distributed; this assumption fails in many cases, such as for time series. Here, we assume that each trial is drawn independently from the same distribution (i.e., the trials are independent, identically distributed variables). For more details, see [6].

1. Sample with replacement 1,000 trials of the EEG data from condition A.
2. Average these 1,000 trials to create a resampled ERP.
3. Repeat these two steps 3,000 times to create a distribution of ERPs.
4. For each time point, identify the values greater than 2.5% and greater than 97.5% of all 3,000 values. This range determines the 95% confidence interval of the ERP for that time point.

Let's now implement each step in MATLAB. In step 1 we must sample with replacement from the EEG data. To visualize this procedure, imagine placing 1,000 marbles in an opaque bag. Each marble is assigned a unique integer value from 1 to 1,000. Now, reach your hand into the bag, grab a marble, record its number, and replace the marble in the bag. We assume that each marble is equally likely to be selected at each draw (i.e., there are no special features that allow some marbles to be drawn more often). Repeat this procedure 1,000 times to create a list of 1,000 integers. Notice that after recording the drawn marble's number, we replace it in the bag. So, we could potentially draw the same marble 1,000 times, although that's extremely unlikely. Performing this sampling with replacement procedure by hand would, of course, be extremely time consuming (e.g., who will paint integers on each marble?). Fortunately, MATLAB provides a function to perform sampling with replacement:

```
%Draw 1000 integers with replacement from (1,1000);
i=randsample(ntrials,ntrials,1);
```

The first and second inputs to `randsample` specify the maximum integer to draw and the number of integers to draw, respectively. In this case, both inputs equal the number of trials (1,000). The last input is a flag that tells `randsample` to sample with replacement.

Q: Execute this command and examine the values of `i` returned. What values do you find?

The result `i` provides a list of integers between 1 and 1,000. These values specify the trials to use in creating the resampled EEG. This resampled EEG will contain the same number of trials as the original EEG (i.e., 1,000 trials) but in a different order and with possibly repeated trials. For example, if the sampling with replacement procedure returns

```
i = [10, 941, 3, 400, 10, . . .
```

then the first and fifth trials of the resampled EEG will equal the tenth trial of the original EEG. We create the resampled EEG in MATLAB as follows:

```
EEG0 = EEGa(i,:);            %Create the resampled EEG.
```

In this code we use the variable `i` as the index to the rows of `EEGa`.

Q: What is the `size` of the new variable `EEG0`? Is this size consistent with the original EEG datasets?

That completes step 1 of the resampling procedure. Step 2 is easy: we create a resampled ERP from the resampled EEG data. Computing the resampled ERP requires only one line of code in MATLAB:

```
ERP0 = mean(EEG0,1);          %Create the resampled ERP.
```

Q: What is the difference between the resampled EEG and the resampled ERP? Explain your answer in words.

Q: Plot the resampled ERP that we created. What does it look like?

In the first two steps of the resampling procedure we created a single resampled ERP. In step 3 we are instructed to repeat this procedure 3,000 times and create a distribution of ERPs. How can we do so? One potential solution is to cut and paste the code we developed over and over again, for example:

```
i=randsample(ntrials,ntrials,1);  %Draw integers,
EEG1 = EEGa(i,:);                 %... create resampled EEG,
ERP1 = mean(EEG1,1);              %... create resampled ERP.

i=randsample(ntrials,ntrials,1);  %Draw integers,
EEG2 = EEGa(i,:);                 %... create resampled EEG,
ERP2 = mean(EEG2,1);              %... create resampled ERP.

i=randsample(ntrials,ntrials,1);  %Draw integers,
EEG3 = EEGa(i,:);                 %... create resampled EEG,
ERP3 = mean(EEG3,1);              %... create resampled ERP.
```

In these lines, we have created three resampled ERPs, each with its own variable name. We could, of course, repeat this procedure and eventually define the variable `ERP3000`.

Q: Is defining the resampled ERPs in this way a good idea?

A: No! We should let the computer execute this repeated procedure for us.

> If you find yourself cutting and pasting the same code over, you're probably doing something inefficient, inelegant, and error-prone.

A better approach to create the 3,000 resampled ERPs is with a *for-loop*. We do so in MATLAB with the `for` command:

```
ERP0 = zeros(3000,size(EEGa,2));     %Create empty ERP variable.
for k=1:3000                         %For each resampling,
    i=randsample(ntrials,ntrials,1);%... choose the trials,
    EEG0 = EEGa(i,:);                %... create resampled EEG,
    ERP0(k,:) = mean(EEG0,1);        %... save resampled ERP.
end
```

In the first line, we define an empty matrix with 3,000 rows and 500 columns; each row will contain a resampled ERP of length 500 elements, corresponding to the 1 s of EEG data (sampled at 500 Hz) considered here. In the second line, the for-loop begins. For each value of k from 1 to 3,000, we draw the trial indices with replacement (third line), create the resampled EEG (fourth line), and compute and save the resampled ERP (fifth line). This completes step 3 of the bootstrapping procedure.

A note about this segment of MATLAB code: We did not need to include the first line. The purpose of this line is to create an empty variable, in this case a matrix filled with zeros. Then, within the for-loop, we add information to this variable (we fill each row with a resampled ERP). By defining the variable `ERP0` before executing the for-loop, we make MATLAB more efficient, and the code will execute faster. In addition, defining the variable `ERP0` in advance forces us to consider the dimensions of the result, which can be useful in verifying our understanding of the variables that already exist (i.e., `EEGa`) and that we newly create (i.e., here `ERP0`).

Step 4 of the bootstrapping procedure is to determine for each time point the values greater than 2.5% and greater than 97.5% of all values. There are many ways to perform this operation in MATLAB, perhaps the easiest being to *sort* from smallest to largest the 3,000 resampled ERP values at each time point. With the resampled values sorted in this way, we then find the resampled ERP value at index $0.025 \times 3000 = 75$ and $0.975 \times 3000 = 2925$. These indices correspond to the resampled ERP values greater than 2.5% of all values and greater than 97.5% of all values, respectively, and therefore define the lower and upper confidence intervals at each moment in time. We can compute both confidence intervals in MATLAB, and (at last) plot the ERP for condition A with confidence intervals computed using the bootstrapping procedure:

```
sERP0=sort(ERP0);        %Sort each column of resampled ERP.
ciL  =sERP0(0.025*size(ERP0,1),:);  %Determine lower CI.
ciU  =sERP0(0.975*size(ERP0,1),:);  %Determine upper CI.
```

```
mnA = mean(EEGa,1);      %Determine ERP for condition A,
plot(t, mnA, 'LineWidth', 3)        %... and plot it.
hold on                  %Freeze the current plot,
plot(t,ciL)              %... and plot lower CI,
plot(t,ciU)              %... and upper CI.
hold off                 %Release the current plot.

ylabel('Voltage [\mu V]')    %Label the y-axis as voltage.
xlabel('Time [s]')           %Label the x-axis as time.
title('ERP of condition A with bootstrap confidence intervals')
```

The results are plotted in figure 2.9. We can use these results to identify, for example, intervals in which the ERP differs significantly from zero by finding periods in which the confidence intervals do not include zero. The advantage of the bootstrapping procedure over other approaches is that this procedure requires few assumptions about the distribution of the statistic of interest, and that we use the observed data to probe the distribution of the statistic. The disadvantage of the bootstrapping procedure is that it is computationally intensive. Here we considered 3,000 resamplings, but we could easily consider more.

Q: Compare the confidence intervals plotted in figures 2.7 and 2.9. How are the two results similar or different? What happens to the confidence intervals if you change the number of resamplings in step 3 from 3,000 to 10,000?

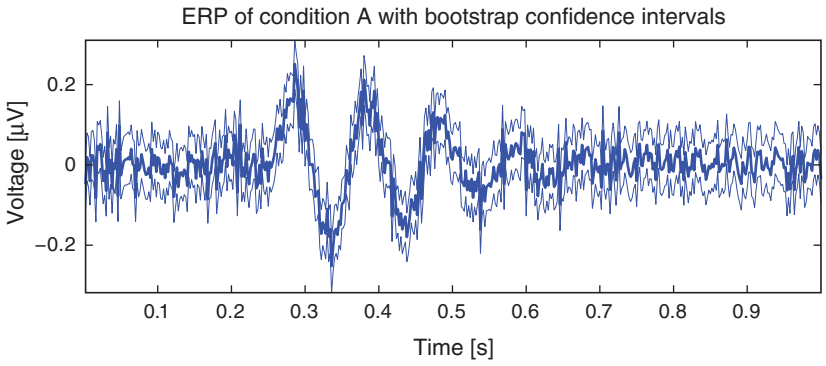

Figure 2.9

ERP with confidence intervals computed using the bootstrapping procedure for condition A. Thick line indicates the mean; 95% of values lie between the thin lines.

Q: Compute the confidence intervals using the bootstrapping procedure for the ERP of condition B. What do you find?

2.2.6 A Bootstrap Test to Compare ERPs

The bootstrapping procedure provides a powerful technique to construct confidence intervals for the ERPs using only the observed EEG measurements. We can apply a similar technique to search for significant differences between the ERPs in conditions A and B. To do so, we first choose a *statistic*, a measure of some attribute of the difference between the two ERPs. There are many choices, some informative and some not. Let's choose as our statistic the maximum absolute value of the difference in the ERPs across time. Computing this statistic is straightforward in MATLAB:

```
mnA = mean(EEGa,1);          %Determine ERP for condition A.
mnB = mean(EEGb,1);          %Determine ERP for condition B.
mnD = mnA-mnB;               %Compute the differenced ERP.
stat = max(abs(mnD));        %Compute the statistic.
```

The variable `stat` is a number, in this case 0.2613.

Q: Given the value we determined for `stat`, are the ERPs for the two conditions different?

In isolation, the numerical value for `stat` is not very useful or interesting. Is the value for `stat` consistent with noisy scalp EEG data lacking an evoked response? Or is the value for `stat` large and unexpected to occur unless the ERPs in the two conditions are different? To make the statistic useful, we need `stat` to be interpretable, which we pursue here through a bootstrapping procedure. We assume that no difference exists between the two conditions; in the language of statistics, this is called the null hypothesis. If the null hypothesis holds, then we can pool all the EEG signals together from both conditions (for a total of 2,000 trials) and draw from this combined distribution to create resampled ERPs representative of either condition.

It may seem odd to create pseudodata by selecting trials across *both* conditions; intuitively, we may expect the data to differ in these two conditions and feel uncomfortable making a pseudodata set that includes trials from both conditions. But under the null hypothesis, we assume no difference between the EEG responses in conditions A and B, and we are therefore free to create pseudodata drawing from trials in both conditions. We do so with the goal of creating a distribution of values for `stat` under the null hypothesis that conditions A and B exhibit no difference. We then compare the observed value of `stat` with this distribution of `stat` values. If there *is* a difference between the two conditions,

we expect to find the observed value of `stat` to be very different from the distribution of `stat` values generated from the pseudodata under the null hypothesis.

To create the distribution of `stat` values under the null hypothesis of no difference between the two conditions, we perform a bootstrap test. The idea is similar to the bootstrapping procedure used to construct the confidence intervals for the ERP (see figure 2.9). We proceed as follows:

1. Merge the 1,000 trials each of EEG data from conditions A and B to form a combined distribution of 2,000 trials.
2. Sample with replacement 1,000 trials of EEG data from the combined distribution, and compute the resampled ERP.
3. Repeat step 2 and compute a second resampled ERP.
4. Compute the statistic, the maximum absolute value of the difference between the two resampled ERPs.
5. Repeat steps 2–4 3,000 times to create a distribution of statistic values.
6. Compare the observed statistic to this distribution of statistic values. If the observed statistic is greater than 95% of the bootstrapped values, then reject the null hypothesis that the two conditions are the same.

The MATLAB code to implement this procedure is similar to the bootstrapping procedure we have already implemented to compute the confidence intervals for the ERP:

```
EEG = [EEGa; EEGb];          %Merge EEG data from all trials.
statD = zeros(3000,1);       %Empty variable to hold results.
for k=1:3000                 %For each resampling,
    i=randsample(2*ntrials,ntrials,1);  %... choose trials,
    EEG0 = EEG(i,:);         %... create resampled EEG,
    mnA = mean(EEG0,1);      %... create resampled ERP.

    i=randsample(2*ntrials,ntrials,1);  %Re-choose trials,
    EEG0 = EEG(i,:);         %... create resampled EEG,
    mnB = mean(EEG0,1);      %... create resampled ERP.

    mnD = mnA-mnB;           %Compute differenced ERP,
    statD(k)= max(abs(mnD));%... and the statistic.
end
```

In this code, we first combine `EEGa` and `EEGb` in a new variable `EEG`. We then resample from `EEG` two times and each time compute a resampled ERP. We use the resampled ERPs to compute the statistic, which is saved in the variable `statD`. We perform this operation

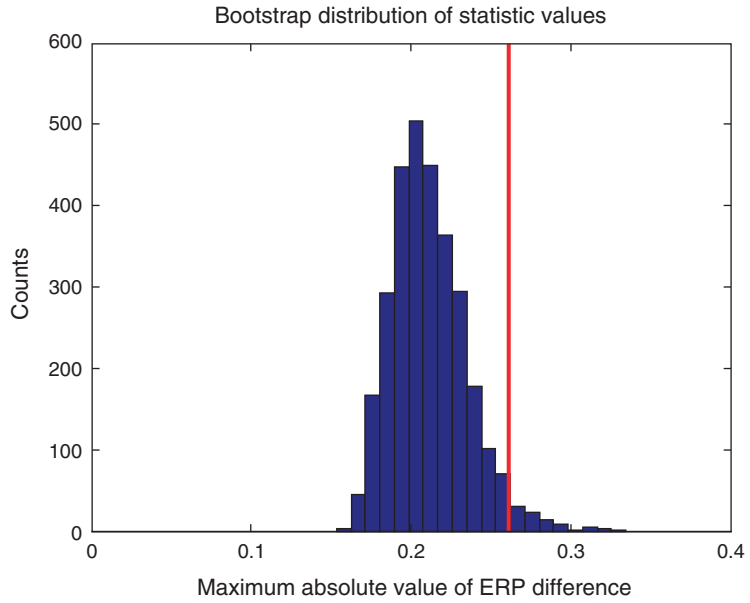

Figure 2.10

Distribution of values for the test statistic under the null hypothesis of no difference between the two conditions. Red line indicates observed statistic from EEG data.

in a for-loop to repeat the procedure 3,000 times. A histogram of the values for `statD` is shown in figure 2.10.

Q: Given the distribution of `statD` values shown in figure 2.10, and the value of `stat` computed from the original data, can we conclude that the difference between the two conditions is significant with this statistic?

A: Yes. Under the null hypothesis, the distribution of the statistic ranges from approximately 0.15 to 0.33 (figure 2.10). The observed statistic `stat` = 0.2613 exceeds most values in this distribution. Computing `length(find(statD > stat))` we find in this example that only 88 of the 3,000 values in the distribution exceed the observed statistic. This corresponds to a proportion of $88/3000 = 0.029$. We therefore reject the null hypothesis of no difference between the ERPs of conditions A and B. This result may be surprising, given how similar the two ERPs appear and the large variability in their differences (figure 2.8).

This result illustrates the power of the bootstrapping procedure. We proposed a complicated statistic (the maximum absolute value of the difference between the two resampled ERPs).

For this statistic, we do not possess an obvious formula to decide whether the resulting statistic is significant (we cannot rely on the CLT, for example). To determine significance, we employ a bootstrapping procedure (also known as a permutation test), which we can perform even for the relatively complicated statistic. In this way, we may devise complicated measures of data and construct error bars or compute statistical significance, provided our computational resources are sufficient.

Summary

In this chapter, we considered scalp EEG data recorded from a single electrode during an auditory task. The task consisted of two conditions, and we sought to uncover the difference in the EEG responses between the two conditions. We began with a visual inspection of the EEG recordings from individual trials and from all trials, and concluded that the data were quite noisy; any evoked response due to the stimulus was not obvious in the single-trial data. To emphasize the evoked signal, we computed the ERP, which involved averaging the EEG signal across trials. By doing so, we uncovered interesting structure in condition A, but not much in condition B. We then developed two techniques to add error bars to an ERP. One technique relied on the central limit theorem, and the other technique involved a computationally expensive bootstrapping procedure. Both techniques suggested that the ERP in condition A differed significantly from zero following the stimulus at time 0.25 s. Finally, we assessed whether the two ERPs differed. We did so through visual inspection, by comparing the differences in the ERPs, and by computing a statistic and assessing its significance through a bootstrapping procedure. Using the last procedure, we concluded that the ERP in the two conditions significantly differed.

Problems

2.1. Consider an alternative statistic to assess the difference in the ERPs between conditions A and B: the integrated area of the squared difference between the two ERPs. Compute this statistic for the data, and use a bootstrapping procedure to test the null hypothesis of no difference in this statistic between the two conditions.

2.2. Load the file `Ch2-EEG-2.mat`, available at

```
http://github.com/Mark-Kramer/Case-Studies-Kramer-Eden
```

into MATLAB. You'll find two variables:
`EEG` = the EEG data, consisting of 1,000 trials, each observed for 1 s;
`t` = the time axis, which ranges from 0 s to 1 s.
These data have a similar structure to the data studied in this chapter. To collect these data, a stimulus was presented to the subject at 0.25 s. Analyze these data for the presence of an evoked response. To do so, answer the following questions.

a. What is the time between samples (`dt`) in seconds?

b. Examine some individual trials of these data. Explain what you observe in pictures and words. From your visual inspection, do you expect to find an ERP in these data?

c. Compute the ERP for these data, and plot the results. Do you observe an ERP (i.e., times at which the 95% confidence intervals do not include zero)? Include 95% confidence intervals in your ERP plot, and label the axes. Explain in a few sentences the results of your analysis, as you would to a collaborator who collected these data.

2.3. Load the file `Ch2-EEG-3.mat`, available at

```
http://github.com/Mark-Kramer/Case-Studies-Kramer-Eden
```

into MATLAB. You'll find two variables:
`EEG` = the EEG data, consisting of 1,000 trials, each observed for 1 s;
`t` = the time axis, which ranges from 0 s to 1 s.
These data have a similar structure to the data studied in this chapter. To collect these data, a stimulus was presented to the subject at 0.25 s. Analyze these data for the presence of an evoked response. To do so, answer the following questions.

a. What is the time between samples (`dt`) in seconds?

b. Examine some individual trials of these data. Explain what you observe in pictures and words. From your visual inspection, do you expect to find an ERP in these data?

c. Compute the ERP for these data, and plot the results. Do you observe an ERP (i.e., times at which the 95% confidence intervals do not include zero)? Include 95% confidence intervals in your ERP plot, and label the axes. Explain in a few sentences the results of your analysis, as you would to a collaborator who collected these data.

2.4. Load the file `Ch2-EEG-4.mat`, available at

```
http://github.com/Mark-Kramer/Case-Studies-Kramer-Eden
```

into MATLAB. You'll find two variables:
`EEG` = the EEG data, consisting of 1,000 trials, each observed for 1 s;
`t` = the time axis, which ranges from 0 s to 1 s.
These data have a similar structure to the data studied in this chapter. To collect these data, a stimulus was presented to the subject at 0.25 s. Analyze these data for the presence of an evoked response. To do so, answer the following questions.

a. What is the time between samples (`dt`) in seconds?

b. Examine some individual trials of these data. Explain what you observe in pictures and words. From your visual inspection, do you expect to find an ERP in these data?

c. Compute the ERP for these data, and plot the results. Do you observe an ERP (i.e., times at which the 95% confidence intervals do not include zero)? Include 95% confidence intervals in your ERP plot, and label the axes. Explain in a few sentences the results of your analysis, as you would to a collaborator who collected these data.

2.5. In the previous problem, you considered the dataset `Ch2-EEG-4.mat` and analyzed these data for the presence of an ERP. To do so, you averaged the EEG data across trials. The results may have surprised you. Modify your analysis of these data in some way to better illustrate the appearance (or lack thereof) of an evoked response. Explain what's happening in these data, as you would to a colleague or experimental collaborator.

2.6. Load the file `Ch2-EEG-5.mat`, available at

`http://github.com/Mark-Kramer/Case-Studies-Kramer-Eden`

into MATLAB. You'll find two variables:
`EEG` = the EEG data, consisting of 1,000 trials, each observed for 1 s.
`t` = the time axis, which ranges from 0 s to 1 s.
These data have a similar structure to the data studied in this chapter. To collect these data, a stimulus was presented to the subject at 0.25 s. Analyze these data for the presence of an evoked response. To do so, answer the following questions.

a. What is the time between samples (`dt`) in seconds?

b. Examine some individual trials of these data. Explain what you observe in pictures and words. From your visual inspection, do you expect to find an ERP in these data?

c. Compute the ERP for these data, and plot the results. Do you observe an ERP (i.e., times at which the 95% confidence intervals do not include zero)? Include 95% confidence intervals in your ERP plot, and label the axes. Explain in a few sentences the results of your analysis, as you would to a collaborator who collected these data.

2.7. Compare the datasets `Ch2-EEG-3.mat` and `Ch2-EEG-4.mat` studied in the previous problems. Use a bootstrap procedure to test the hypothesis that the evoked response is significantly different in the two datasets.

2.8. The goal of this problem is to explore the central limit theorem. We informally defined the CLT in this chapter. Let's perform a numerical experiment to test it.

a. Generate a large set of independent random numbers. These are many options to choose from. For example, use MATLAB to produce 500 uniformly distributed random numbers:

```
R = rand(500,1);
```

Describe this list of random numbers. Include a histogram (with axes labeled).

b. Compute the mean of this list of numbers. What value do you find for the mean? Does the mean value make sense when compared to the histogram you created?

c. Generate another instance of the independent random numbers you chose in part (a) and compute its mean, Repeat 5,000 times, saving the result each time. In the end, you should have 5,000 values, each corresponding to a mean of a different realization of the variable `R`. Describe the distribution of these mean values of `R`. What do you find? Is your result consistent with the CLT? (*Hint*: Make a histogram and consider its shape . . .)

d. Repeat your analysis with a different choice for the independent random numbers. Note that MATLAB provides many different options. For your new choice of random numbers, describe a single instance of 500 values, as in part (a), and then describe the distribution of the mean of these random numbers, as in part (c).

Appendix: Standard Error of the Mean

Assume the values,

$$x = \{x_1, x_2, x_3, \ldots, x_K\},$$

are independent data drawn from a distribution with a theoretical mean μ and theoretical variance σ^2. The sample mean of x is

$$\bar{x} = \frac{1}{K}\sum_{k=1}^{K} x_k \,.$$

Since each of the values x_k is a sample of a random variable with a probability distribution, so is the sample mean. We are interested in computing the properties of the distribution of the sample mean of the original data.

First, we compute the expected value, or theoretical mean, of the sample mean,

$$E\big[\bar{x}\big] = \frac{1}{K}\sum_{k=1}^{K} E\big[x_k\big] = \frac{K\mu}{K} = \mu \,.$$

In other words, the sample mean has the same expected value, or theoretical mean, as the original data. This is why we often use the sample mean to estimate the theoretical mean of the data.

Next, we compute the theoretical variance of the sample mean of the data. The variance is the squared deviation of the sample mean of the data from the theoretical mean,

$$\begin{aligned}\mathrm{Var}\Big(\bar{x}\Big) &= E\Big[\Big(\frac{1}{K}\sum_{k=1}^{K} x_k - \mu\Big)^2\Big] \\ &= E\Big[\Big(\frac{1}{K}\sum_{k=1}^{K} \big(x_k - \mu\big)\Big)^2\Big] \\ &= \frac{1}{K^2} E\Big[\Big(\sum_{k=1}^{K} \big(x_k - \mu\big)\Big)^2\Big].\end{aligned}$$

The expected value in the last line of this expression is the theoretical variance of the sum of zero-mean random variables $(x_k - \mu)$. Assuming that the variables x_k are independent, the variance of the sum becomes the sum of the variances, so

$$\begin{aligned}\mathrm{Var}\Big(\bar{x}\Big) &= \frac{1}{K^2}\sum_{k=1}^{K} E\Big[\big(x_k - \mu\big)^2\Big] \\ &= \frac{1}{K^2}\Big(\mathrm{Var}\big[x_1\big] + \mathrm{Var}\big[x_2\big] + \mathrm{Var}\big[x_3\big] + \cdots + \mathrm{Var}\big[x_K\big]\Big) \\ &= \frac{1}{K^2}\Big(\sigma^2 + \sigma^2 + \sigma^2 + \cdots + \sigma^2\Big) \\ &= \frac{1}{K^2}\Big(K\sigma^2\Big) \\ &= \frac{1}{K}\sigma^2,\end{aligned}$$

where Var with a capital V refers to the theoretical variance of a random variable (we use lowercase var to refer to the sample variance computed from the data). The last expression defines the variance of the sample mean of x in terms of the variance of each observation of x. To know σ^2, we would need information about the distribution from which each x_k was drawn. We usually do not have this information, so instead we assume that each x_k is drawn from the same distribution, and approximate σ^2 as the sample variance of x. The advantage of this approach is that we can then estimate σ^2 from the observed data, that is, $\hat{\sigma}^2 = \mathrm{var}\Big[x\Big]$. Then

$$\mathrm{Var}\Big(\bar{x}\Big) \approx \frac{1}{K}\mathrm{var}\Big[x\Big].$$

In words, to find the theoretical variance of the sample mean of x we first compute the sample variance of x and then divide this sample variance by the number of observations K.

The sample standard deviation of the sample mean of x, which is also called the *standard error of the mean* (sem), is then

$$\begin{aligned}\mathrm{sem}[x] &= \sqrt{\mathrm{Var}\left[\bar{x}\right]} \\ &\approx \sqrt{\frac{1}{K}\,\mathrm{var}[x]} \\ &= \frac{\mathrm{std}[x]}{\sqrt{K}},\end{aligned}$$

where $\mathrm{std}[x]$ is the sample standard deviation of x.