



# Introduction to EEG Methods

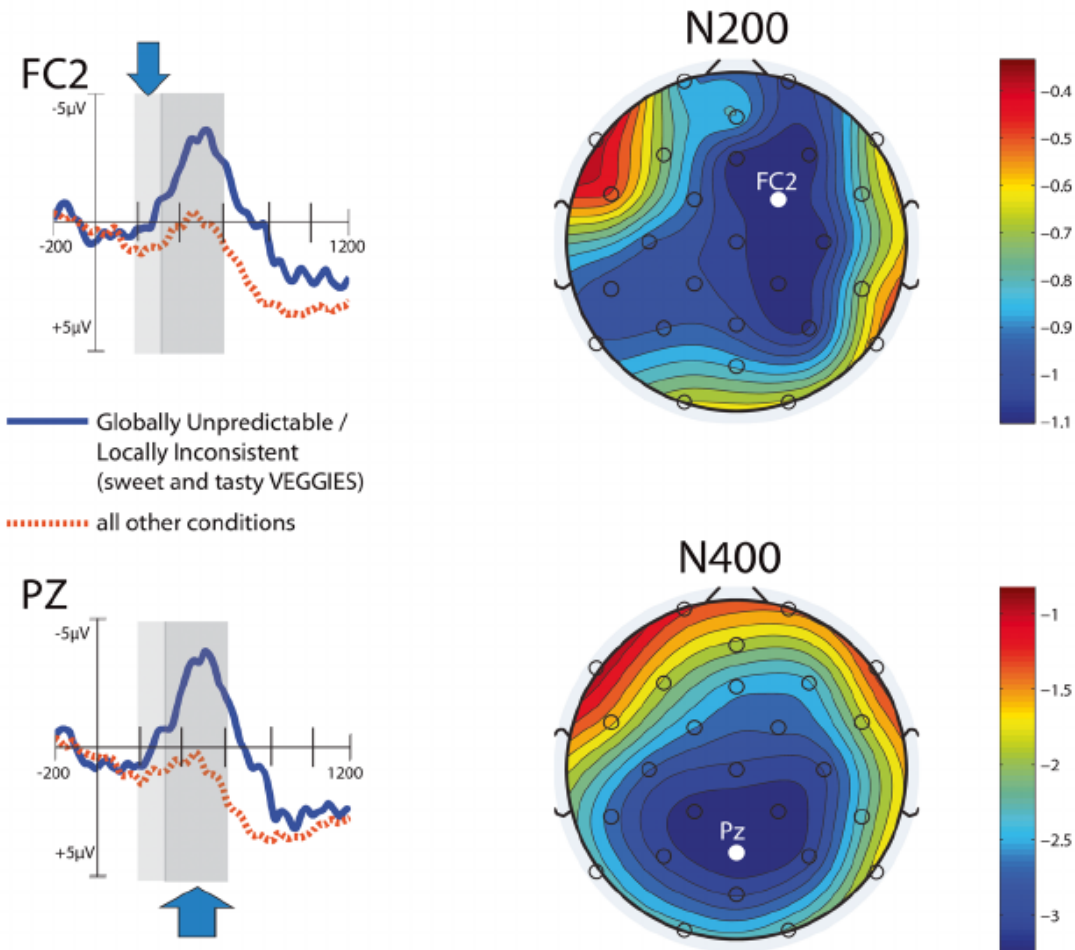
Jon Sprouse  
University of Connecticut

# Table of contents

1.	Introduction: The big picture	Luck 1
2.	Fundamentals	Luck 2
3.	Hands-on training and best practices	Luck 5
4.	The ERP processing pipeline in EEGLAB + ERPLAB	Luck 6, 7, 8
1.	Load the data.	
2.	Filter the data (high-pass, possibly low-pass).	
3.	ICA for artifact correction (optional).	
4.	Re-reference the data.	
5.	Add channel locations.	
6.	Epoch the data.	
7.	Artifact detection and rejection.	
8.	Average the epochs to create subject ERPs.	
9.	Average the subject ERPs to create a grand average ERP.	
10.	Plotting: waveforms, topoplots, difference waves	
11.	Measure amplitudes and latencies.	
12.	Run statistical tests.	
5.	Creating a script for EEGLAB + ERPLAB	
6.	Creating a script for Fieldtrip	

# What is Electroencephalography?

From Boudewyn et al. 2015



Electric head writing.

EEG is the measurement of voltage on the scalp, typically over time.

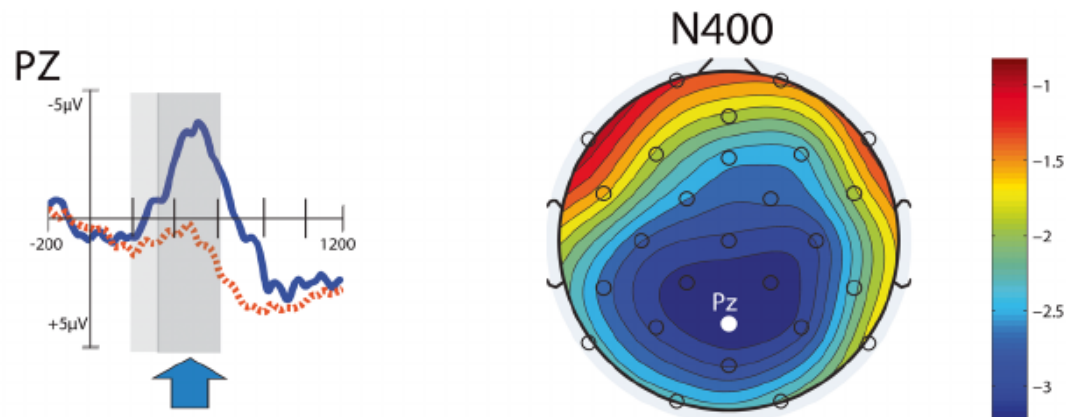
What is voltage?

Where does the electricity come from?

How is it measured?

# What is EEG good for?

<b>Basic answer:</b>	To detect a difference between conditions.
Acceptability Judgments:	One dimension of difference (higher/lower), provided at the end of a cognitive task.
Reaction times:	One dimension of difference (faster/slower), provided at the end of a cognitive task.
EEG:	Four dimensions of difference (polarity, amplitude, latency, scalp distribution), provided continuously during a cognitive task.





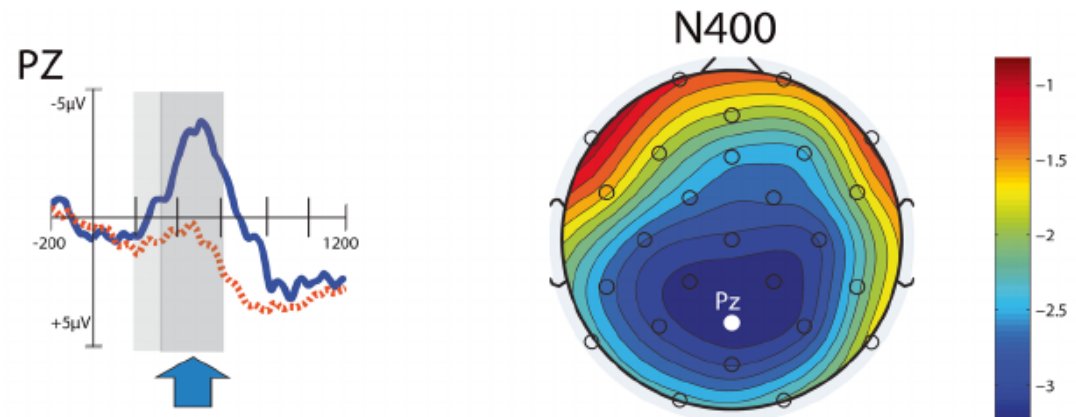
# What is EEG good for?

**A loftier goal:** To detect changes in mental states that can be plausibly linked to cognitive theories, by correlating specific combinations of polarity, amplitude, latency, and scalp distribution with hypothesized cognitive operations.

## An example:

A female chicken is a **hen**.

A female chicken is a **star**.

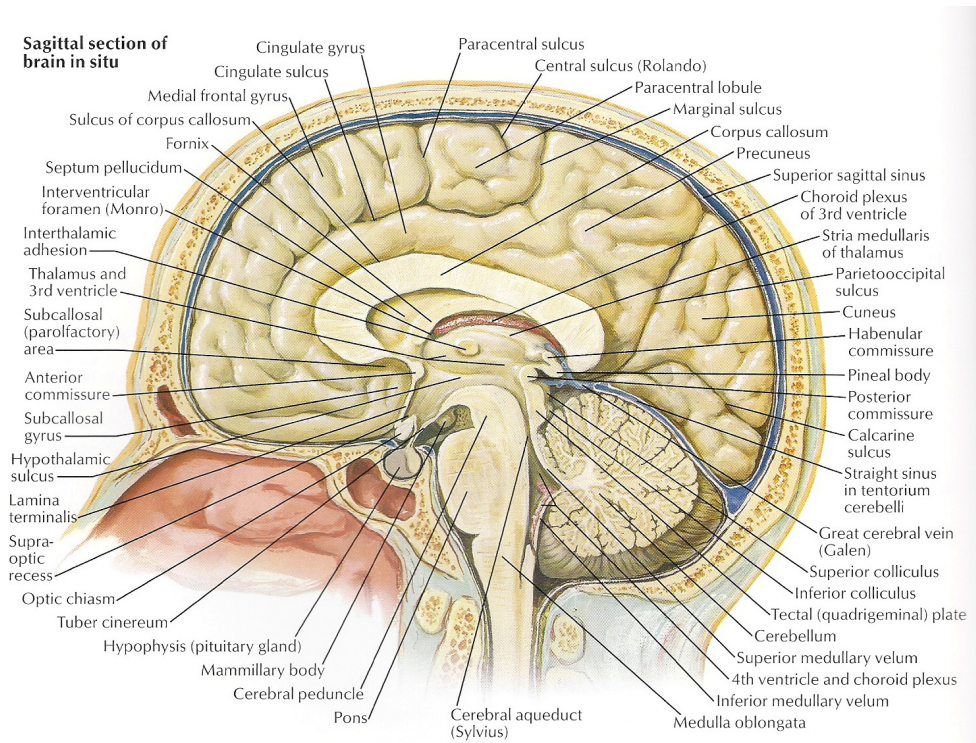


The negative-going deflection in voltage of the **incongruent word** (relative to the congruent word) is called an **N400**. It tends to occur between 300ms and 500ms after the onset of the critical word, and tends to have a central scalp distribution (the one above is central-posterior).

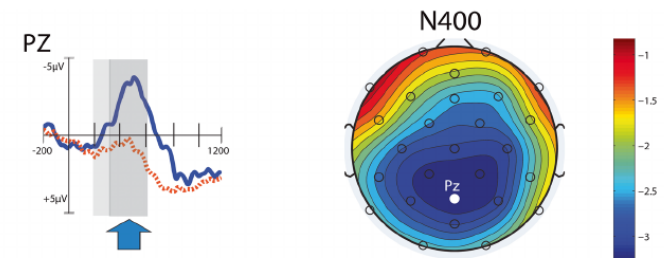
**What are some possible functional interpretations for this effect?**

# What is EEG good for?

**An unlikely goal:** To learn something about how the brain executes certain cognitive computations.



-VS-



There appears to be a mismatch between the complexity of the brain and the amount of information in scalp potentials.

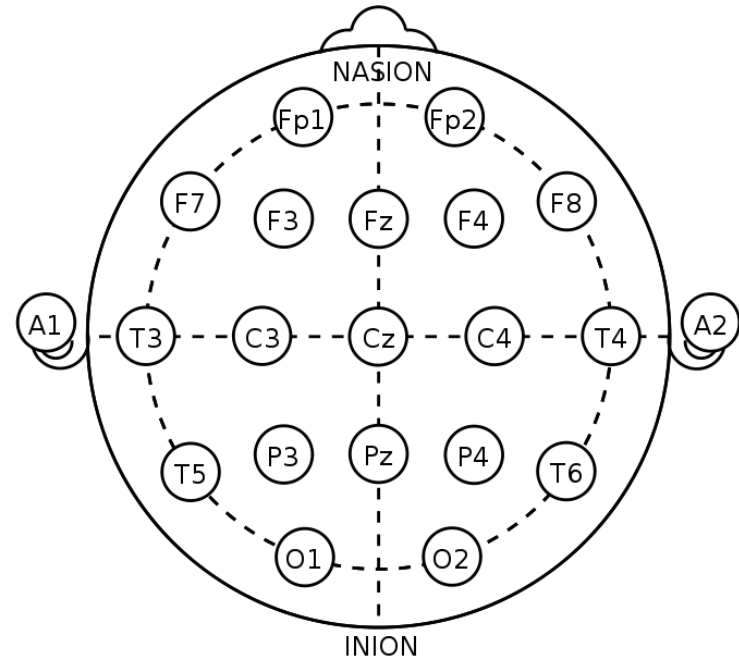
Which aspects of brain function lead to scalp potentials?

If you are interested in brain function, methods like ECoG and single-cell recordings are more likely to be useful.

# A first look at scalp EEG



**Brain Products Caps**

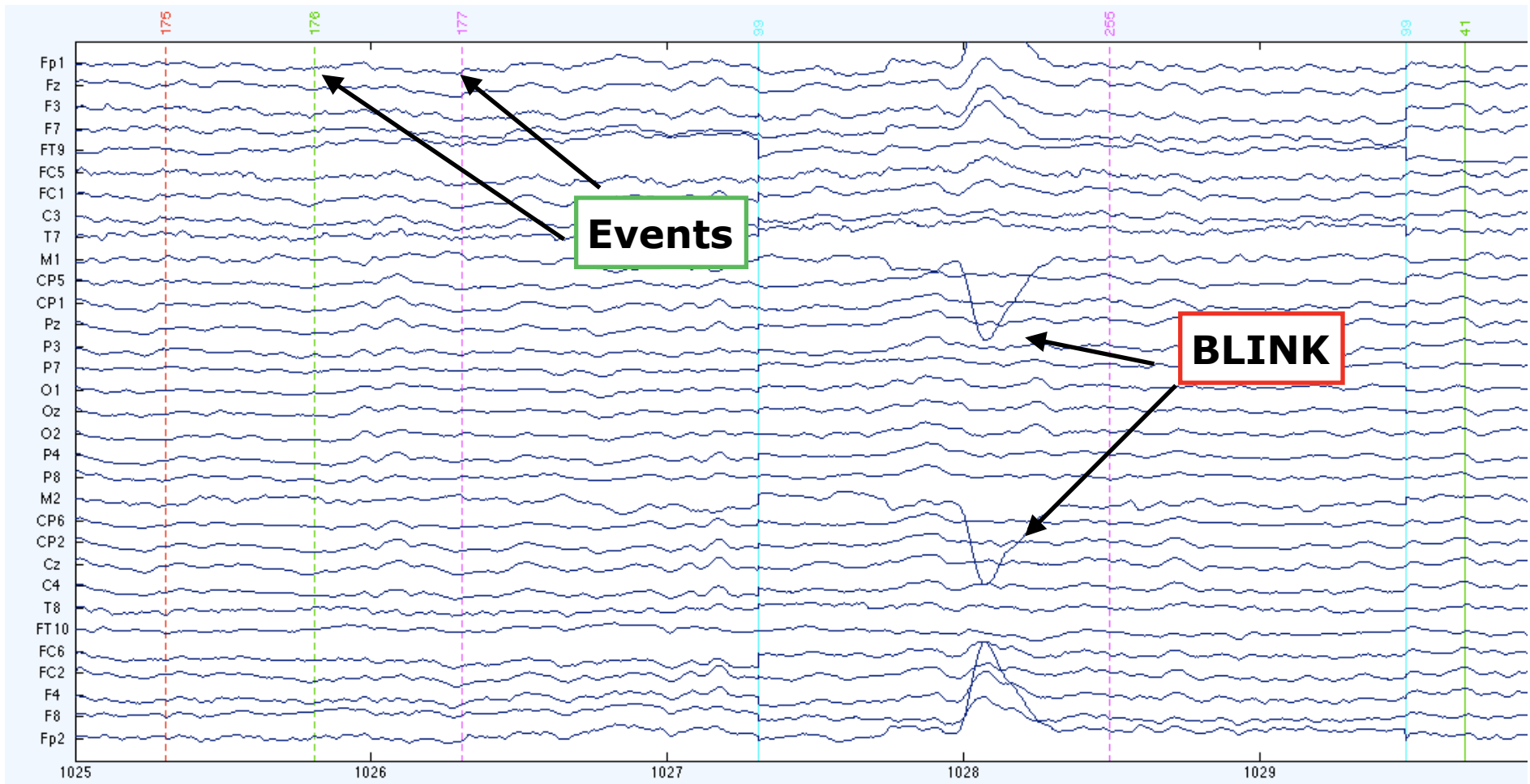


Typical EEG systems place electrodes in multiple locations on the scalp.

Don't worry about the details of the locations right now. This slide just exists to give you an intuitive understanding of the fact that we tend to record EEG from multiple locations simultaneously (often in multiples of 32 channels: 32, 64, 128, 256).

# A first look at scalp EEG (the trace)

With multiple electrodes across the scalp, the raw results of an EEG are a continuous stream of waves, one stream for each electrode (there is an independent y-axis for each stream, it just isn't typically labeled):



The raw output of an EEG is often called the **trace** -- because in the old days, it was a set of physical pens drawing on a scrolling piece of paper.

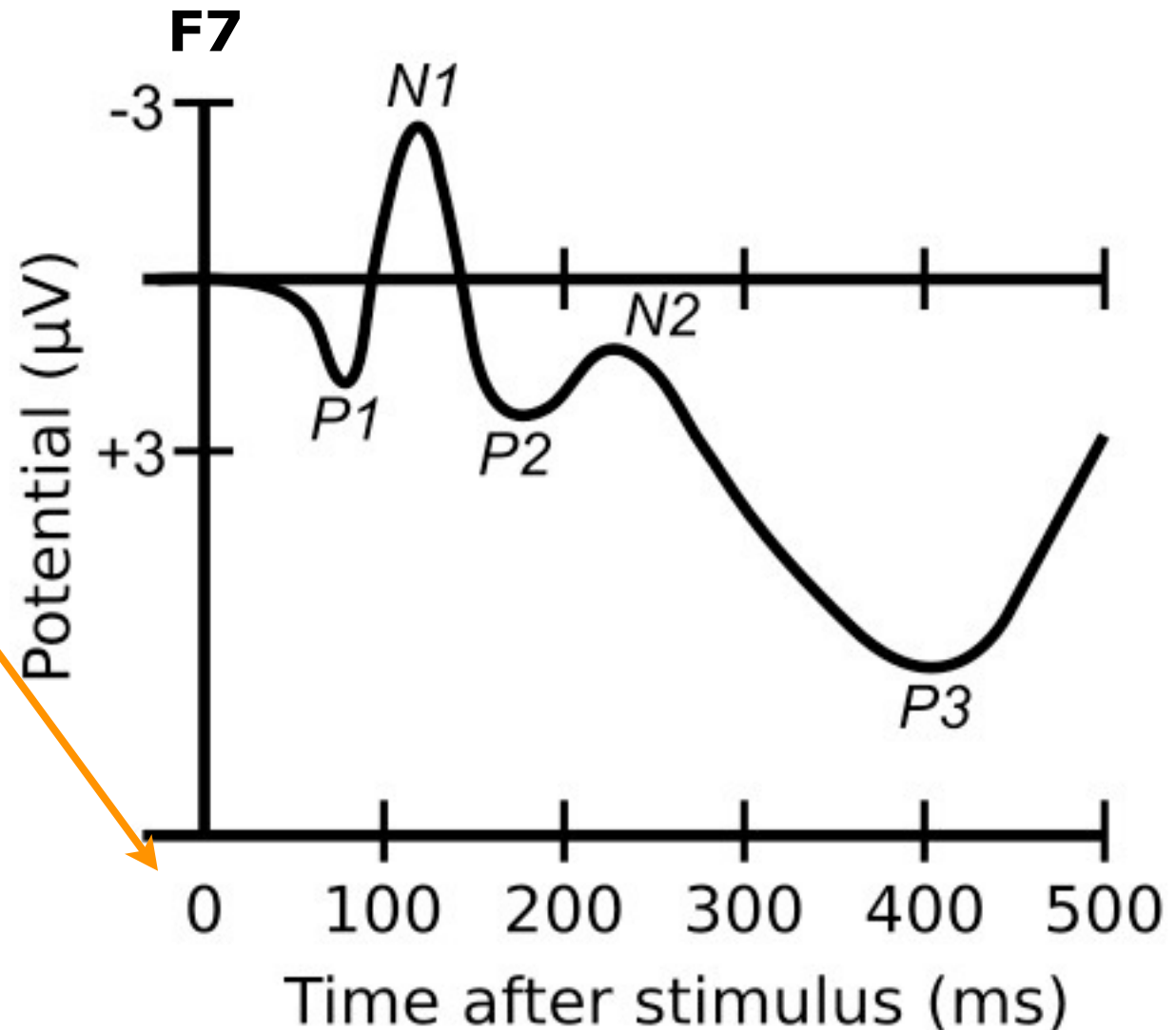
# Event-related Potentials (ERPs)

The first analysis technique we will look at in this class is the **event related potential (ERP)** technique.

The idea of the ERP technique is to time-lock the recording of the EEG signal to an event, such as playing a sound or displaying a word on the screen.

The **event** is represented by **time 0** on the x-axis.

The changes in amplitude on the scalp can then be correlated with the processing of that event over time.





# Event-related Potentials (ERPs)

It takes some time to get used to reading ERPs, but the general structure is relatively easy to learn.

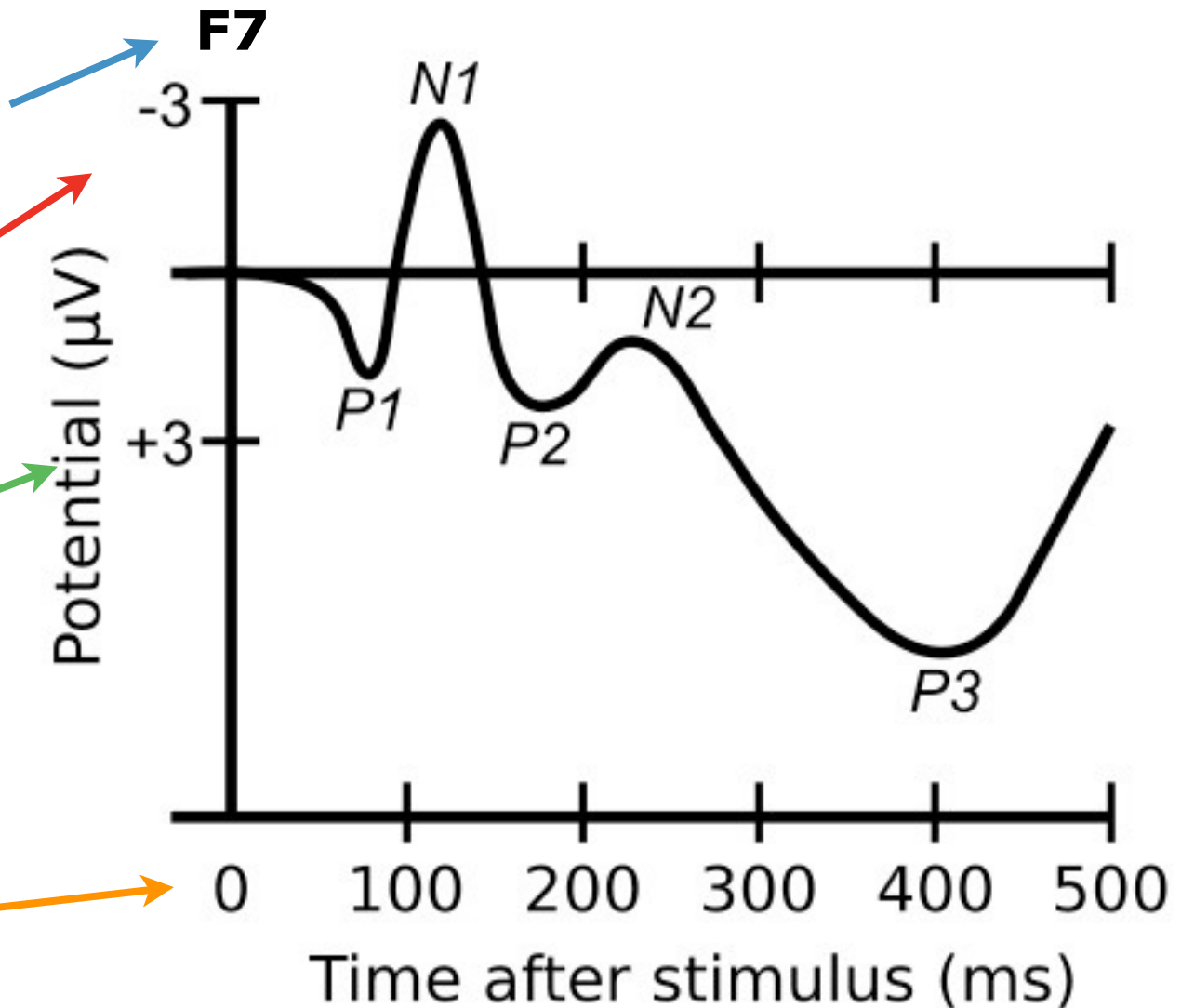
Somewhere in the figure will be an electrode label

The y-axis represents changes in voltage amplitude

Note that **negative** voltage is **up**, and **positive** is **down**. This is the traditional way to plot this. Luck now recommends positive up.

The x-axis represents time.

Note that the **event** is indicated by **time 0**.



# Event-related Potentials (ERPs)

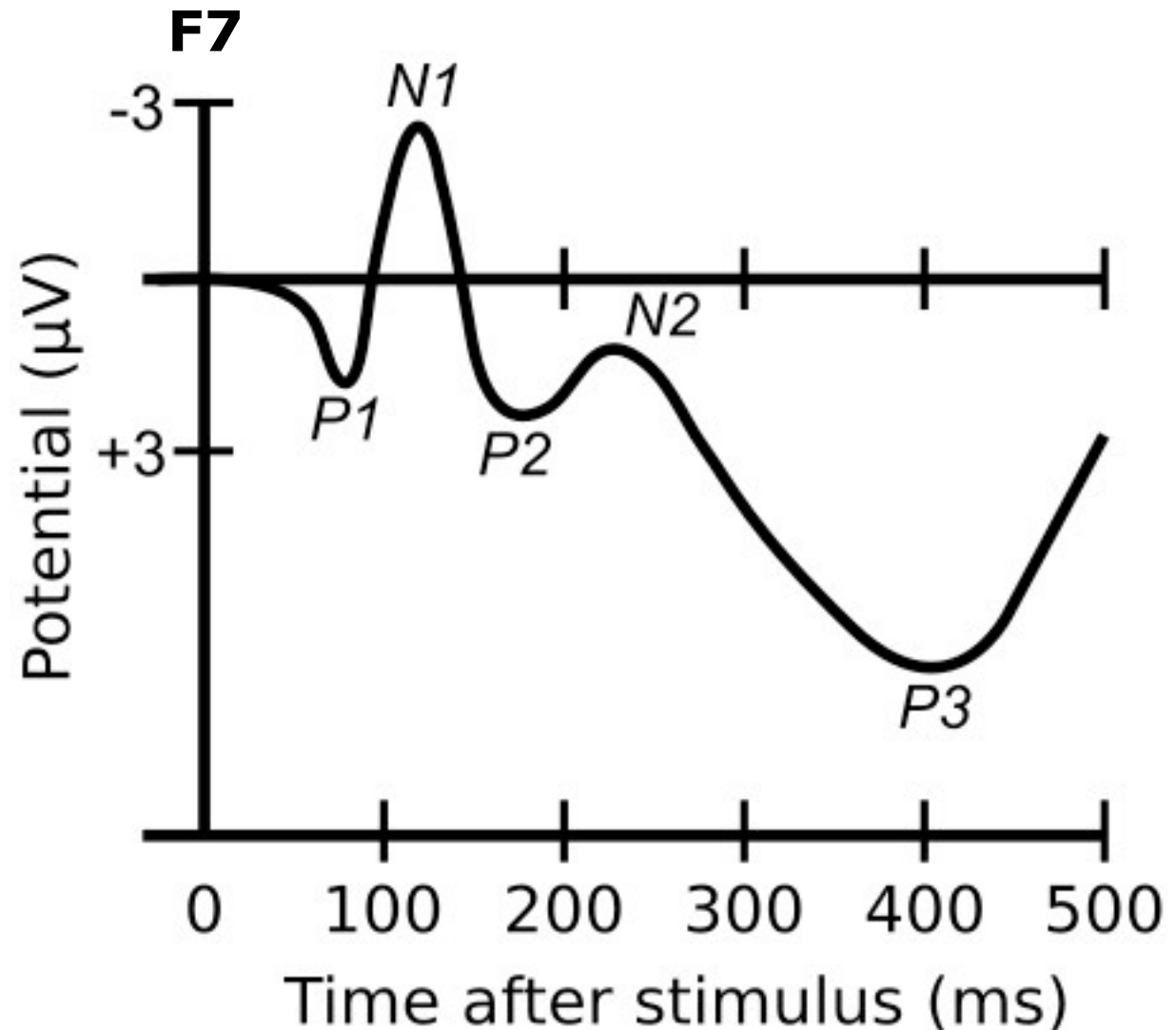
The deflections in the figure are often called **components** of the ERP. (Though the actual definition of component is a bit more complex... more later!)

Each component can be labeled using the following notation:

The letter represents the **polarity** of the component: **N** for negative, and **P** for positive.

The numbers represent the ordinal number of the component in a given polarity (P2 is the second positive deflection).

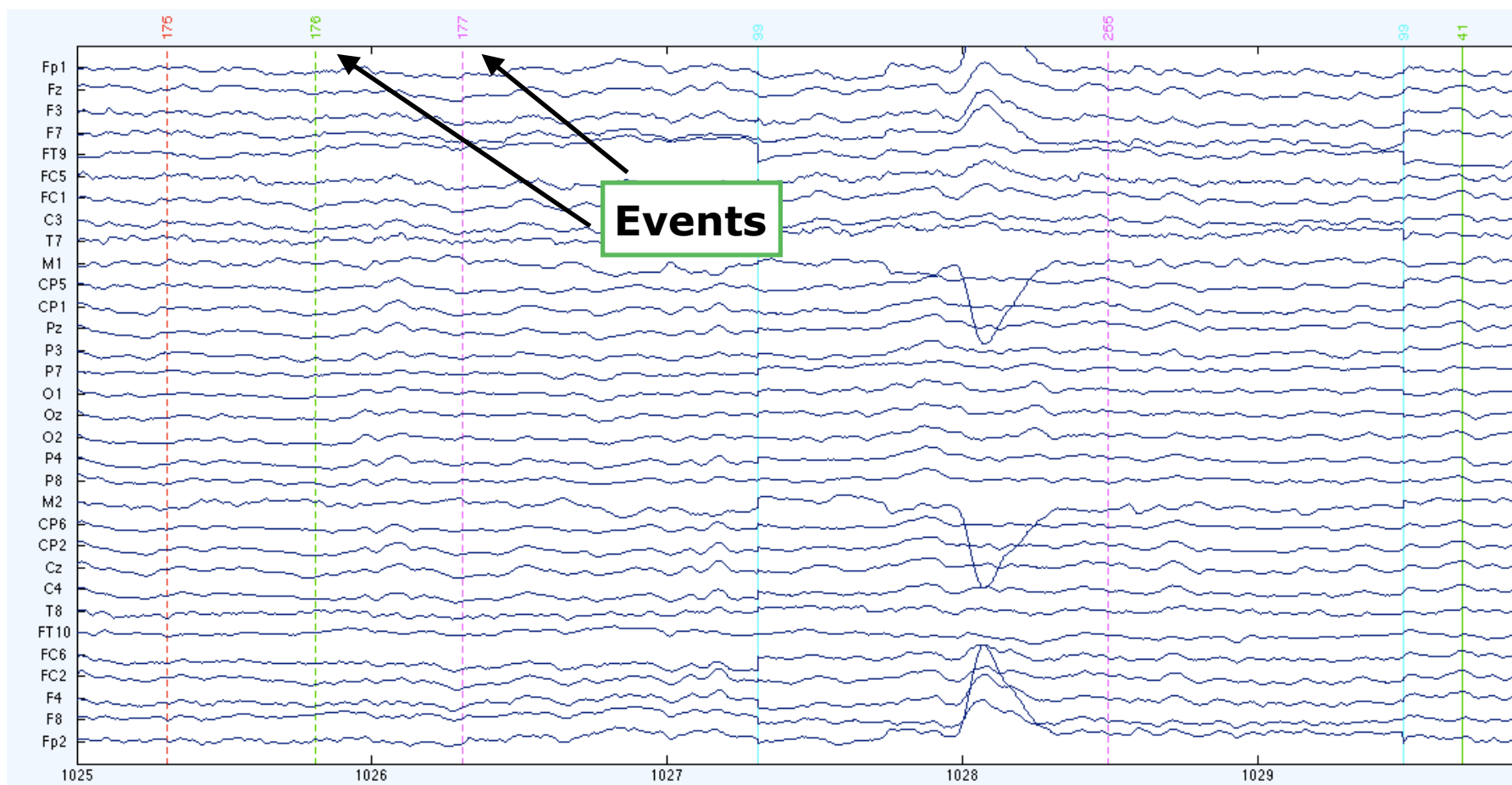
There are important questions to discuss in the future about what a component is, and how we can identify them in ERPs.



# Getting from the trace to ERPs

Step 1: Cut the trace into slices (called epochs) based on events of interest.

If you look closely at the trace below, you will see colored vertical lines with numbers on them. Those are **event markers**. The experimenter puts those in to indicate times in the trace where an event took place.

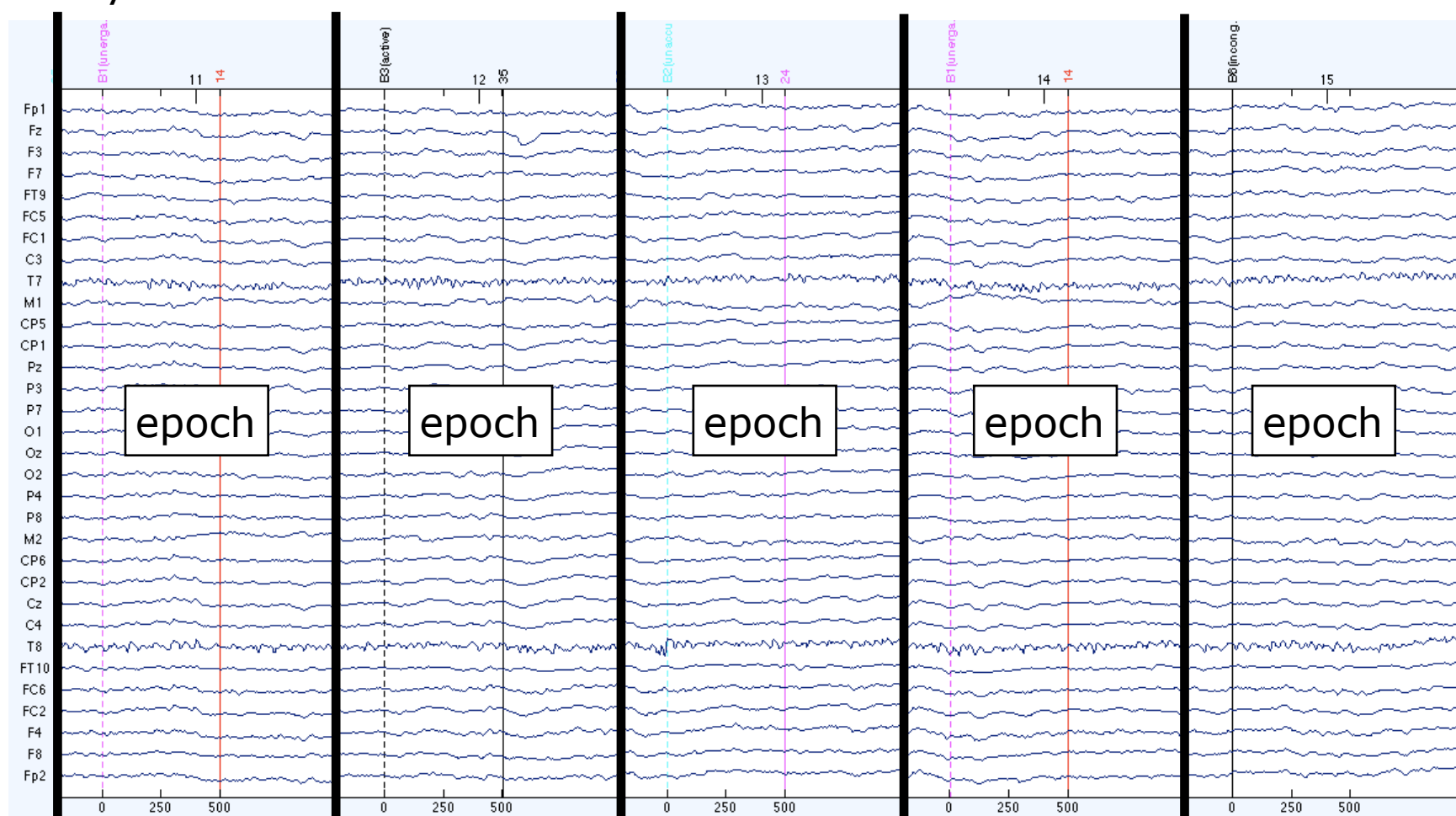




# Getting from the trace to ERPs

Step 1: Cut the trace into slices (called epochs) based on events of interest.

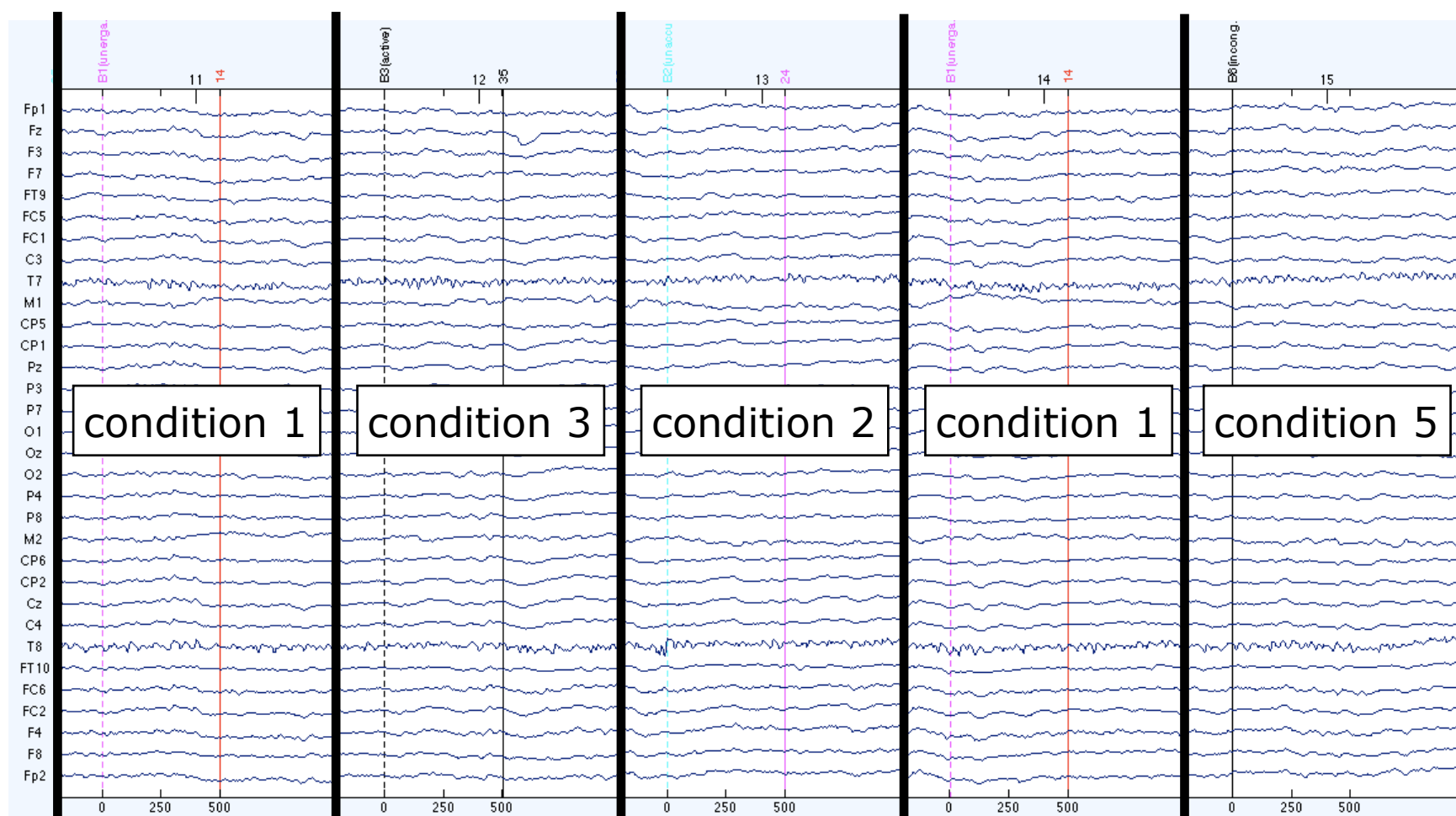
You can tell special computer programs (like Matlab toolbox ERPLAB) to cut out a slice of time around an event. Typically you cut out 100-200ms before the event, and 800-1000ms after the event. Here is a plot of a series of epochs after they've been cut out:



# Getting from the trace to ERPs

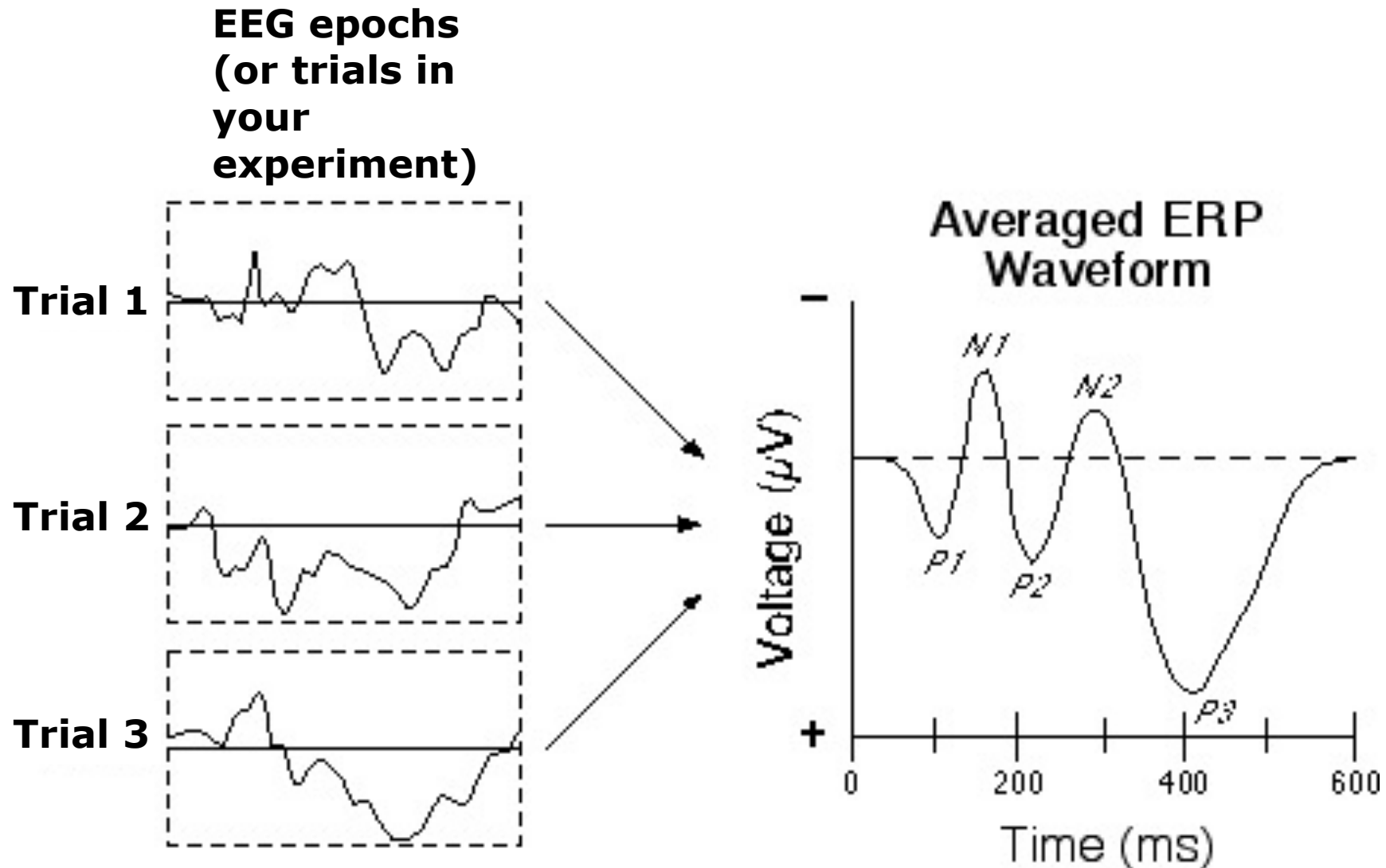
Step 2: Sort the epochs by condition. Put all epochs of the same condition together into a “bin” for that condition.

Most likely your experiment has (at least) two conditions. (The experiment below actually has 6 conditions.)



# Getting from the trace to ERPs

Step 3: Align all of the epochs (or trials) from one condition using the event as time-point 0, and average them together.



# Why average across trials?

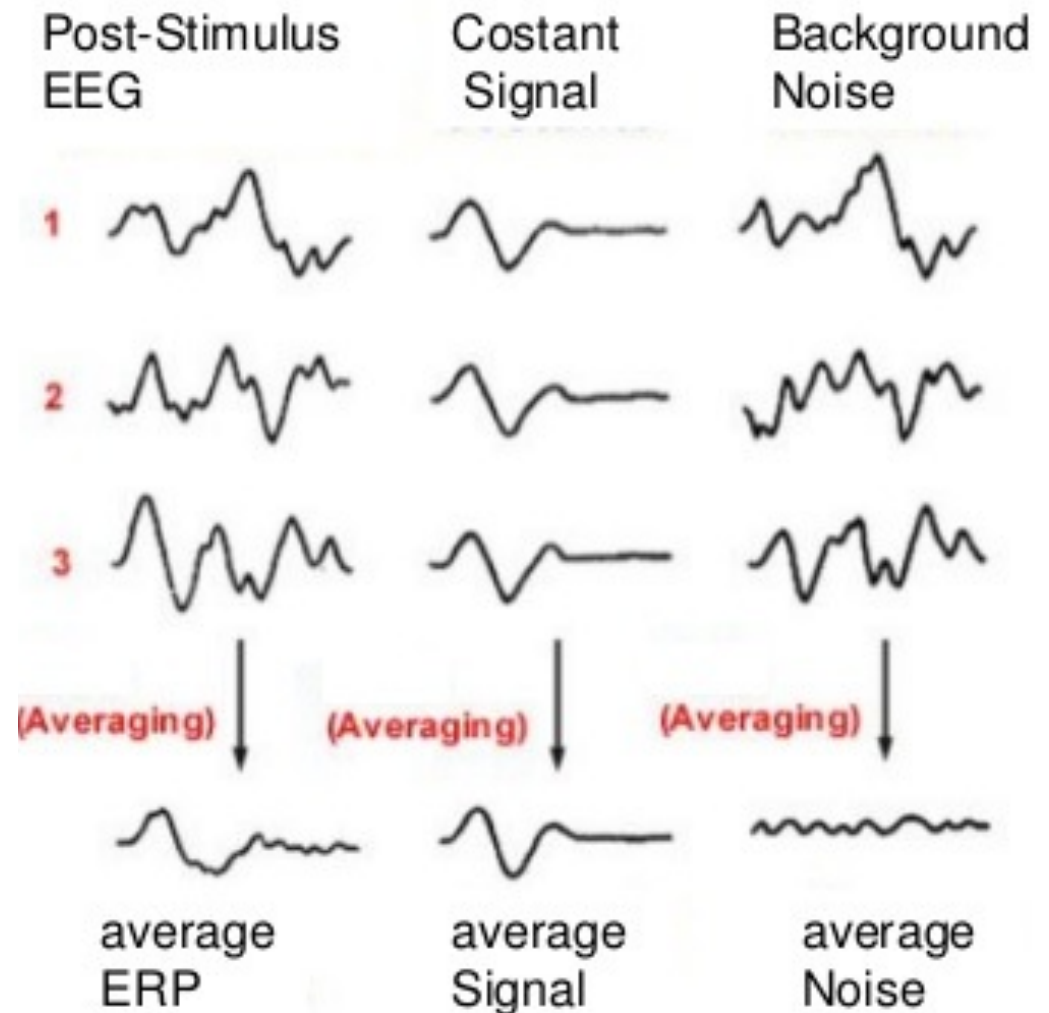
The reason why we average over trials is that EEG data contains lots of **noise**. By noise, we mean electrical activity that is **not related to the cognitive event** we are studying.

The theory behind averaging is as follows:

**Assumption 1:** The signal from the cognitive event is roughly identical in latency and amplitude across trials.

**Assumption 2:** The signal from noise sources is roughly random in latency and amplitude across trials.

Therefore, when averaging across trials, the **signal will remain roughly constant**, and the **noise will reduce to roughly zero**.

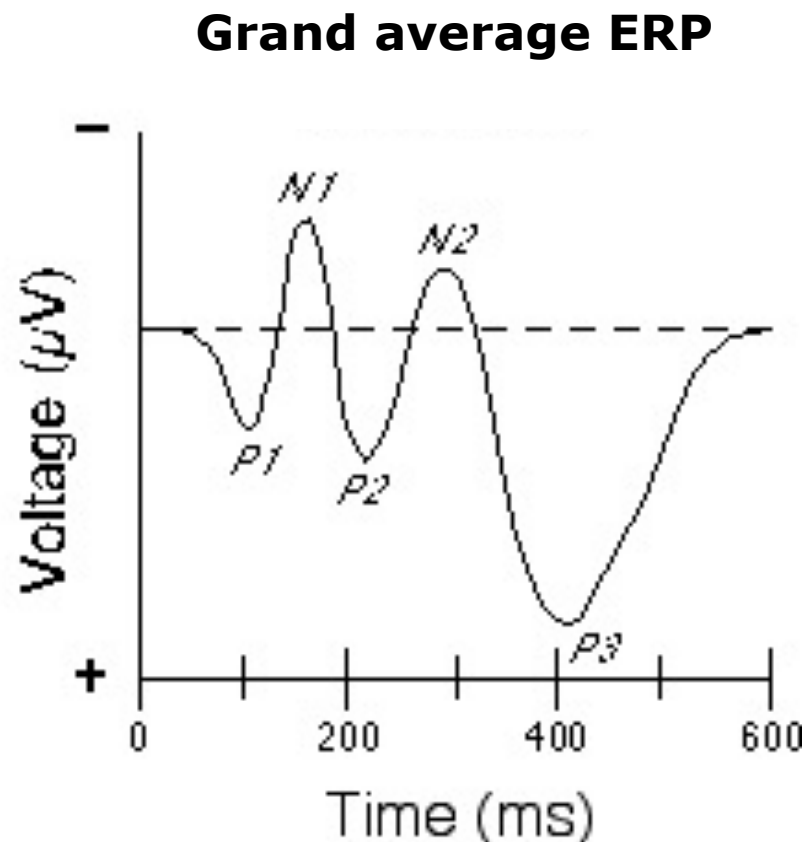
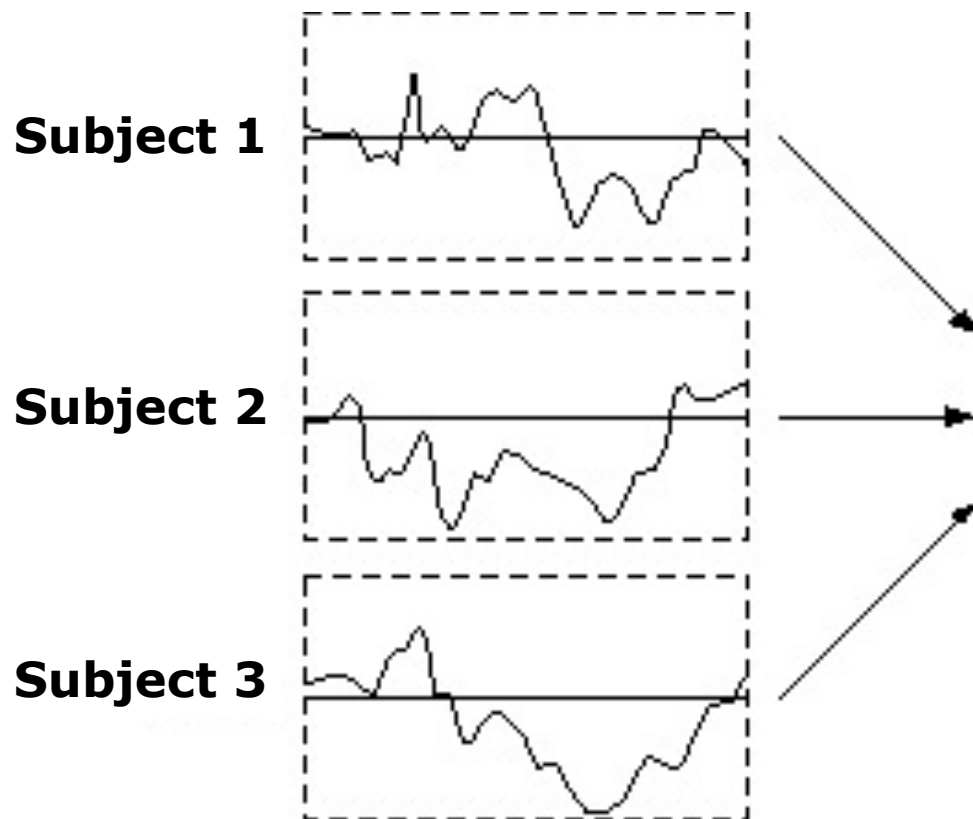


# Getting from the trace to ERPs

Step 4: Average the ERP from each subject into one ERP called the **grand average ERP** (it is grand because it is an average of averages).

I am cheating here by using the same picture as I used for trial level averaging... but I can do that because averaging is averaging.

## Subject-level ERPs



# Getting from the trace to ERPs

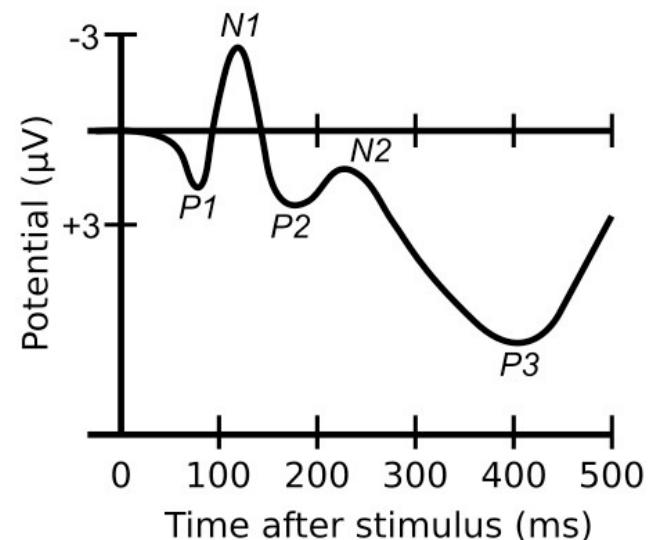
**Step 1:** Cut the trace into slices (called epochs) based on events of interest.

**Step 2:** Sort the epochs by condition. Put all epochs of the same condition together into a “bin” for that condition.

**Step 3:** Align all of the epochs (or trials) from one condition using the event as time-point 0, and average them together to create an ERP for each participant.

**Step 4:** Average the ERP from each participant into one ERP called the grand average ERP (it is grand because it is an average of averages).

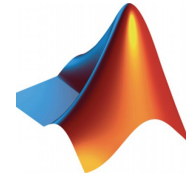
Please note that there are a couple of steps that I skipped where you clean up the data (filtering, artifact rejection)... but those aren't crucial to the fundamental logic of ERPs, they are just physical realities of measuring electricity from human scalps. We will cover all of these steps in detail in classes to follow.



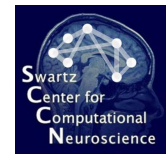
# Software for analyzing EEG

## Matlab + EEGLAB + ERPLAB

**MATLAB:** A proprietary computer language that specializes in matrix algebra. It costs money, but the university buys it for us.



**EEGLAB:** A free, open source toolbox for EEG analysis developed by the Swartz Center at UC San Diego.



**ERPLAB:** A free, open source plugin for EEGLAB developed by Steve Luck's lab at UC Davis.

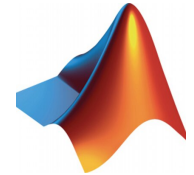
## Jon's opinion:

If you only want to do ERPs, this is probably the best software solution. EEGLAB and ERPLAB are mature toolboxes, with tons of documentation, tutorials, discussion boards, etc. They can be run either through a GUI or through scripting. Steve Luck developed ERPLAB to implement his ideas about how best to do ERP research, so it works well with his textbook. EEGLAB is also great for ICA, but not great for time-frequency analysis.

# Software for analyzing EEG

## Matlab + FieldTrip

**MATLAB:** A proprietary computer language that specializes in matrix algebra. It costs money, but the university buys it for us.



**FieldTrip:** A free, open source toolbox for EEG analysis developed by the Donders Institute in Nijmegen.



## Jon's opinion:

FieldTrip is the best option if you want to do time-frequency analysis (which we will talk about later in the course). It is also completely fine for ERPs. But it does not have a GUI, so you have to be comfortable with scripting (it is a set of Matlab functions that you can call to perform different EEG analyses).



# Software for analyzing EEG

## Python + MNE-Python

Python: A free, open-source computer language.



MNE: A free, open source toolbox for EEG analysis developed by a community of labs.



## Jon's opinion:

I don't have much experience with MNE-python. My impression is that it is completely adequate for ERPs and time-frequency, and that it excels at source localization. I think it may become an interesting option as Python slowly takes over market share from Matlab, so we will try to look at it together in this course. But be warned — for this one, I will be learning along with you.

# A note about plotting

Eventually, you will have to make publication quality plots of your results. In my opinion, Matlab makes truly awful plots. This poses a problem if you are using one of the Matlab-centric processing pipelines (EEGLAB/ERPLAB or FieldTrip).



Personally, I think R makes some really nice plots. R is a computer language specifically created for statistics and plotting. So we may want to think about how to get our results into R for plotting.

- Option 1: Do all of the processing in Matlab, and then export the results to R. This works ok.
- Option 2: Use the [reticulate](#) package to call python from inside R. This allows you to use R to run MNE-python, so all of the processing is in R.
- Option 3: Use R to directly to process the EEG. There are currently no complete/mature solutions to this. But Matt Craddock is working on it: <http://www.mattcraddock.com/blog/2017/09/05/eegutils-an-r-package-for-eeg/>.

## Some common ERPs in language work

These represent the “bread and butter” of EEG in language. We try to identify a functional interpretation of these ERPs, and then use these ERPs as a marker of specific processes (ideally to answer questions about how certain bits of language are processed).

By the end of this course, you will absolutely be able to do this kind of work.

# Left Anterior Negativity (LAN)

The LAN is a negative-going deflection that peaks 300ms-500ms post stimulus, and is maximal over left-anterior electrodes.

## **morpho-syntactic violations**

case violations

agreement violations

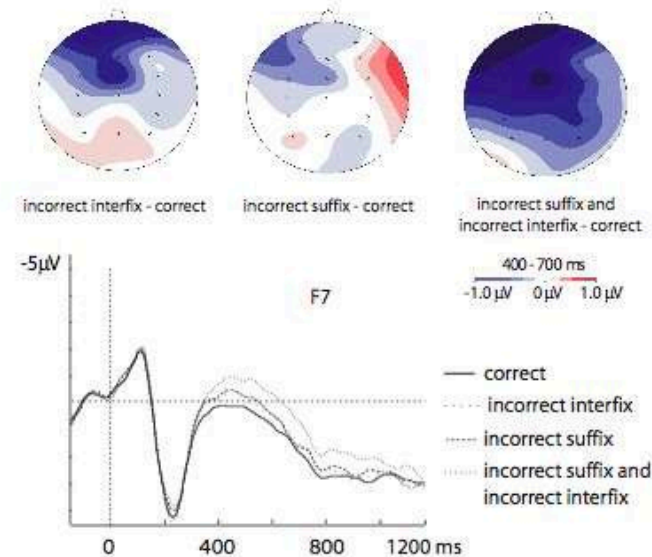
phrase structure violations

island constraint violations

## **grammatical processing**

garden-path sentences

long-distance dependencies



The most common interpretation of the LAN is as an index of morpho-syntactic processing. But there are two unresolved concerns with this:

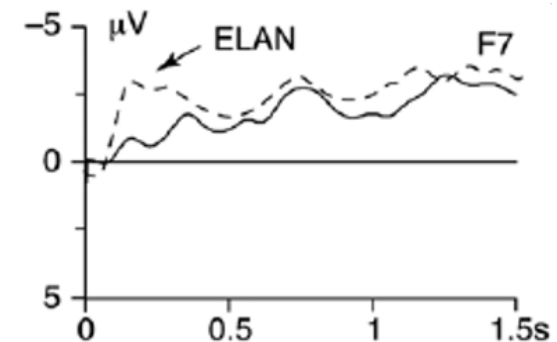
1. The LAN also arises in some grammatical sentences (see above).
2. The LAN has some replicability problems. This has led Tanner and Hall to suggest that it might be an illusion from combining participants with N400s and participants with P600s into one single grand-averaged ERP.

# Early Left Anterior Negativity (ELAN)

The ELAN is a negative-going deflection that peaks very early (100ms-250ms post stimulus), and is maximal over left-anterior electrodes, just like the LAN. First discovered by Neville et al. 1991, it is elicited by phrase structure violations:

\*The boys heard Joe's **about** stories the war.  
The boys heard Joe's stories **about** the war.

There are at least four functional interpretations in the literature, all focusing on the early latency:



**Syntax-first parsing:** Friederici and colleagues have suggested that if syntax were processed first, the early latency to the violation would make sense.

**Predictive pre-processing:** Lau et al. 2006 suggest that predictable items might be “pre-processed”, leading to the appearance of an early effect.

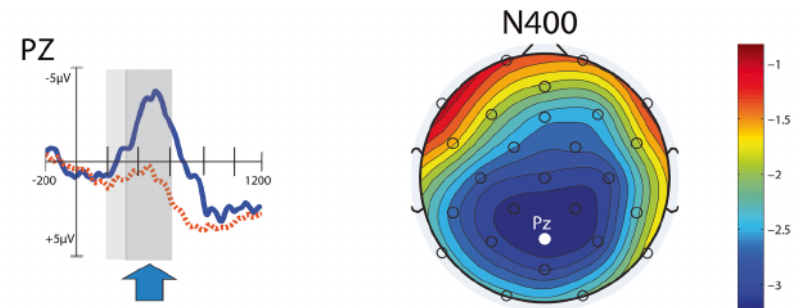
**Sensory pre-processing:** Dikker et al. 2009 suggest that predictable affixes could be processed in the sensory cortices before lexical access.

**ERP Artifact:** Steinhauer and Drury 2012 suggest that the ELAN might be an artifact from comparing conditions that aren't matched pre-stimulus.

# The N400

The N400 is a negative-going deflection that peaks 300-500ms post-stimulus, and is generally largest over central and centro-parietal electrodes. It was first discovered by Kutas and Hillyard 1980:

I spread the warm bread with **butter**.  
I spread the warm bread with **socks**.  
I spread the warm bread with **BUTTER**.



**Semantic Integration:** Hagoort and colleagues have suggested that the **increase** in the N400 for **semantically incongruent words** is an index of the difficulty of semantic integration.

**Activation of semantic features in the lexicon:** Kutas and colleagues have suggested that the **decrease** in the N400 for **predictable words** reflects the ease of activating their semantic features in memory (or pre-activating them).

# The P600

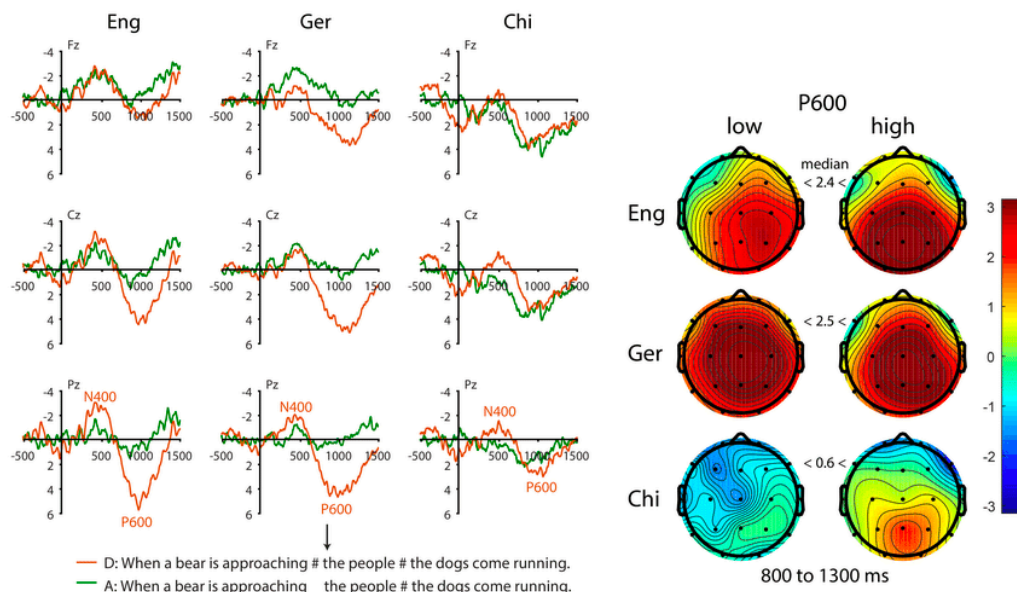
The P600 is a positive-going deflection that peaks 500-800ms post-stimulus, and is generally largest over centro-parietal electrodes. It often co-occurs with the LAN:

## violations

agreement violations  
phrase structure violations  
island constraint violations  
certain semantic violations

## grammatical processing

garden-path sentences  
gap-filling in dependencies

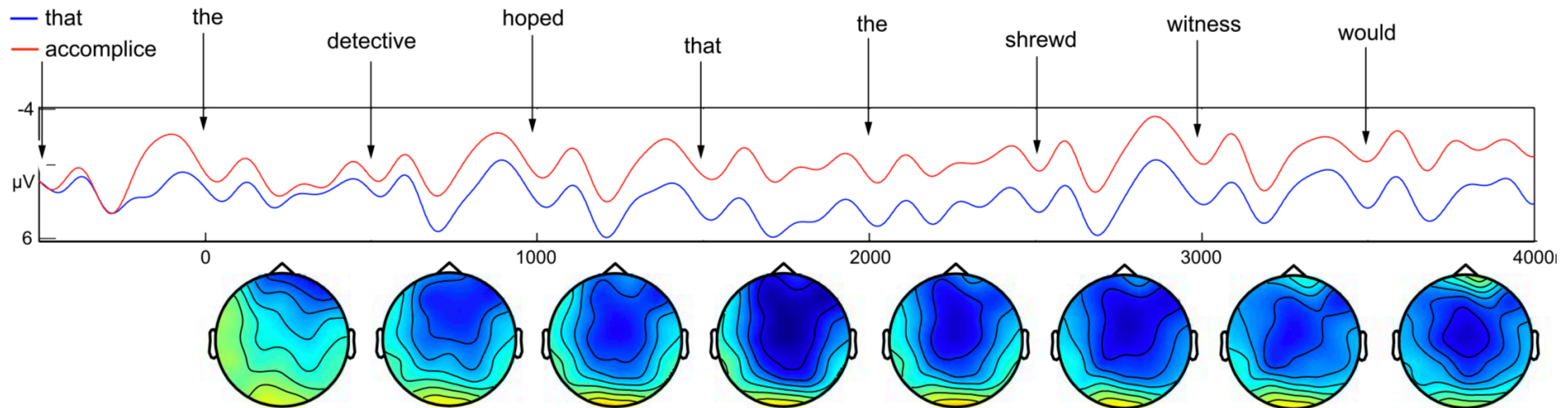


There are two questions driving most of the research into the functional interpretation of the P600

1. Is there a single interpretation for the P600, or multiple interpretations?
2. Is the P600 specific to language (as suggested by the eliciting conditions), or is it simply a delayed P3, which is a domain-general response to unexpected stimuli.

# Sustained Anterior Negativity (SAN)

**Sustained Anterior Negativity**: a negative-going deflection that tends to have an anterior scalp distribution, first reported by King and Kutas 1995. Here is a fairly canonical looking SAN from Phillips et al. 2005. They compared **wh-movement questions** to **declarative sentences**.



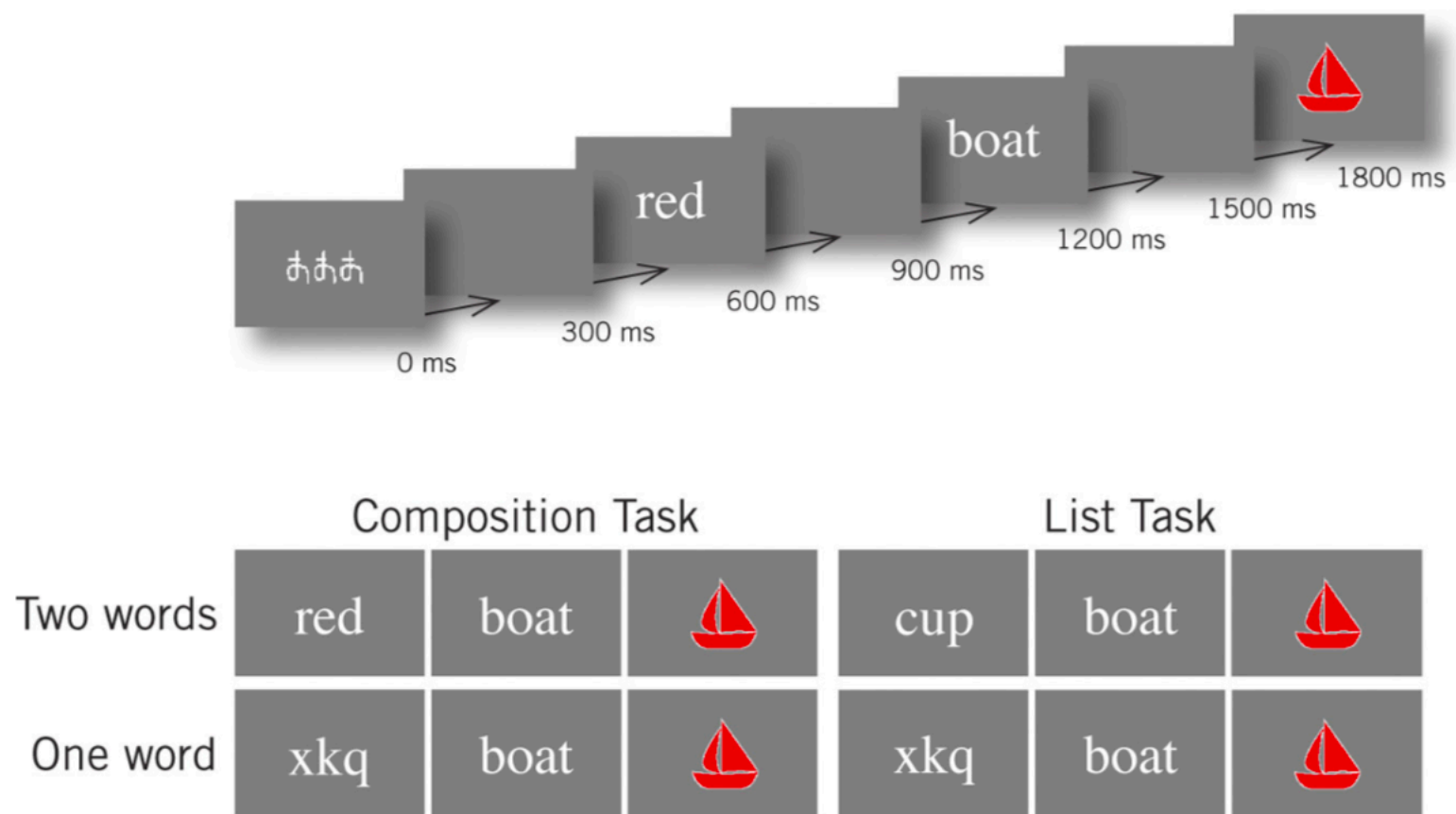
The SAN is a candidate for a correlate of the processes required by **move**, but the question is which process underlying dependencies drives the SAN: **the wh-word in the stack**, or **the predictive processes** searching for the gap location?



## Moving beyond ERPs

Here is a sample of some relatively recent papers that attempt to do new and interesting things with EEG (and MEG) recordings. I review these briefly here to give you a taste of the kind of work you can build toward after this course. (It is good to be aspirational.)

# Bemis and Pylkkänen 2011: composition



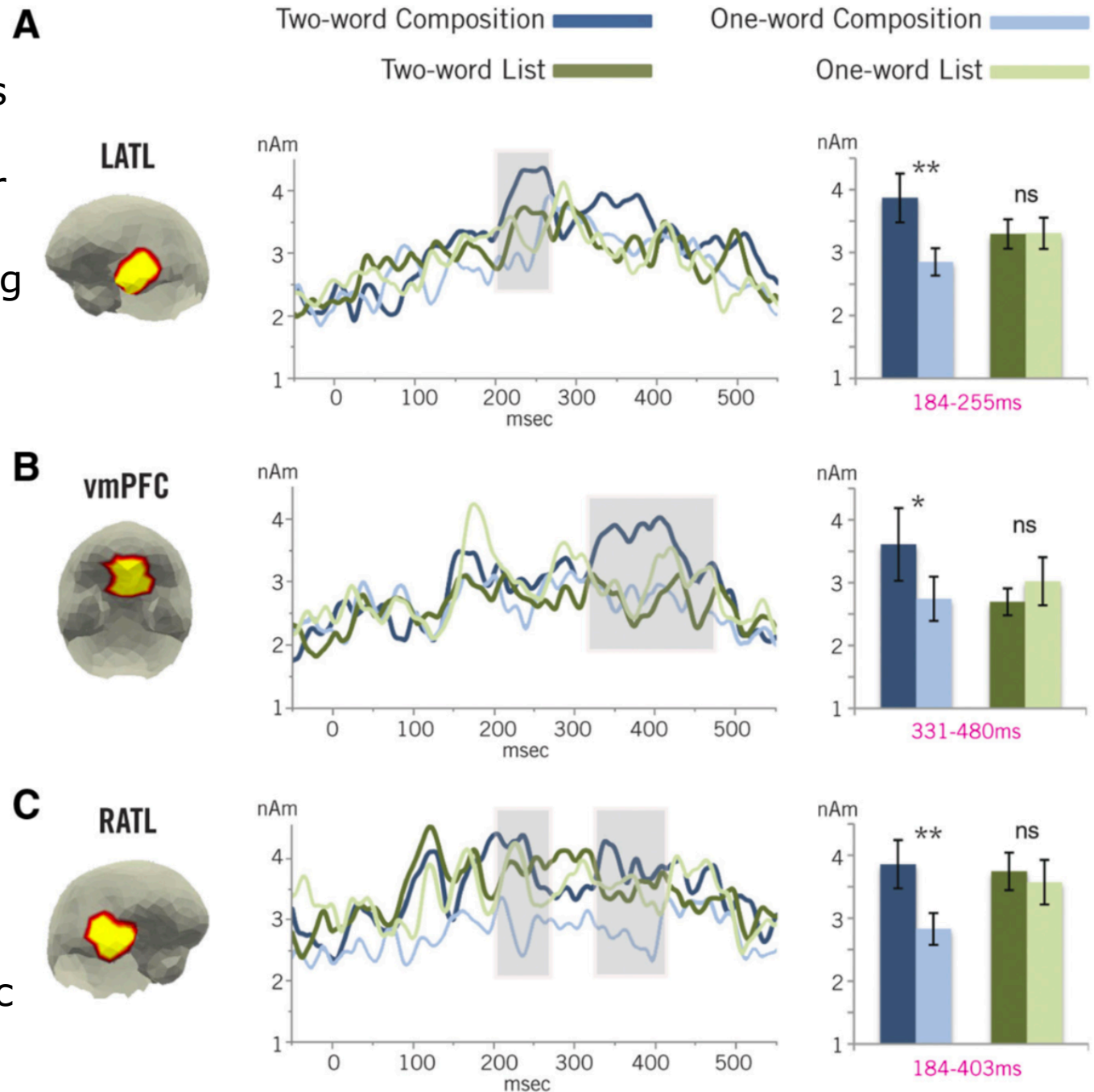
**Figure 1.** Experimental design. Our design crossed task (composition vs list) and number of words (two vs one). In each trial, participants indicated whether the target picture matched the preceding words. To satisfy this criterion, in the composition task, all preceding words were required to match, whereas in the list task, any matching word sufficed. A total of six colors and 25 shapes were randomly combined and used as stimuli. Half of the target pictures matched, but half did not. Only activity recorded at the matched nouns ("boat") was analyzed.

# Bemis and Pylkkänen 2011: composition

Their primary finding is activation in left ATL about 200-300ms after the second word. They point to this as indexing composition.

They also find activity in vmPFC 300-500ms after the second word. This looks like the Anterior Midline Field from previous Pylkkänen work.

The question moving forward is how to use this kind of minimal task to isolate syntactic processes.



# Brennan and Pylkkänen 2016: parsing effort

The story *Crybaby*, which is 1200+ words, but focused on PPs — 224 words inside of PPs.

They created two models: one left corner parser, and one bottom-up parser.

They then counted how many rules each parser would need to deploy at each word inside of PPs.

They then recorded MEG, and looked for correlations between the rule counts and brain areas showing activity in MEG.

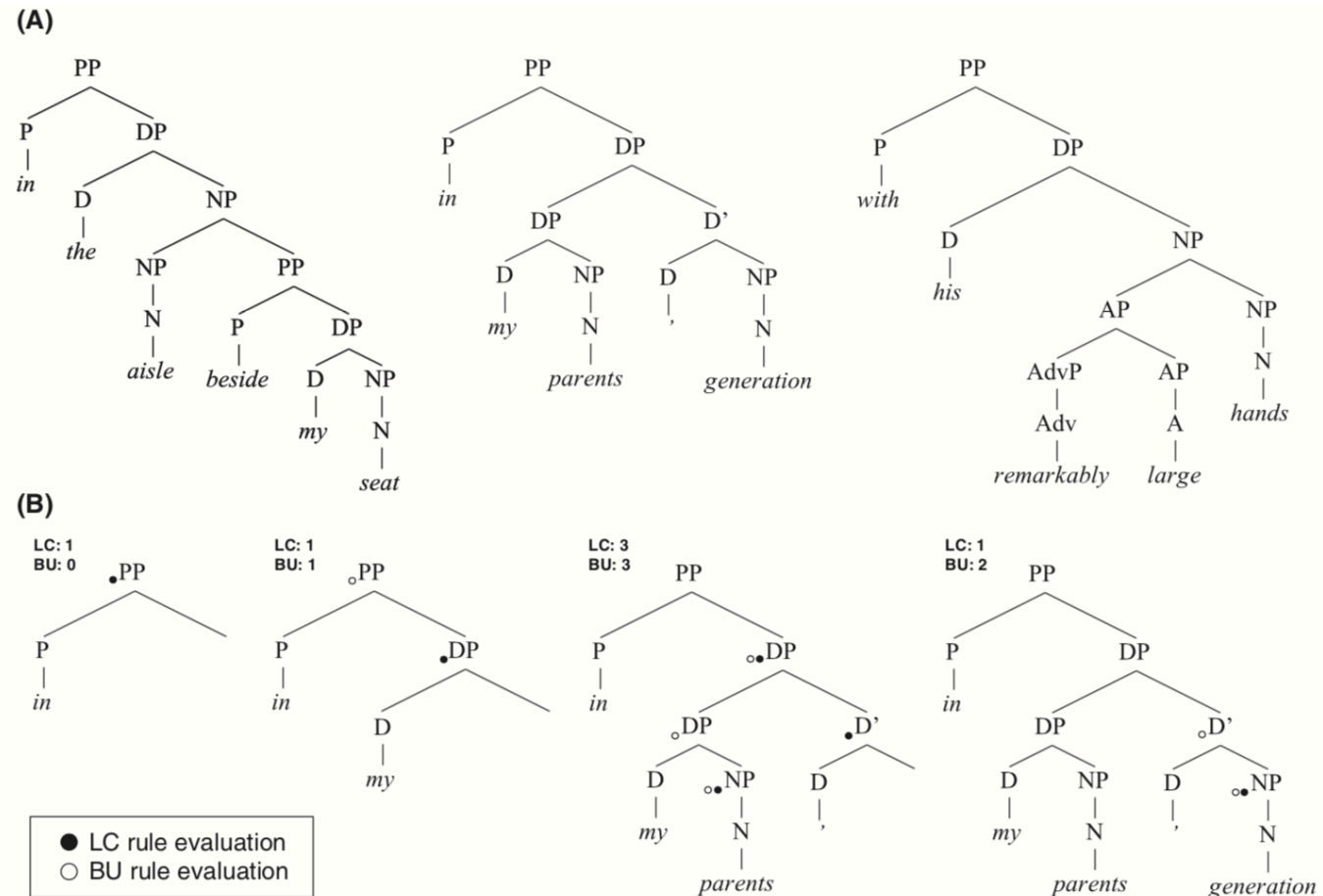
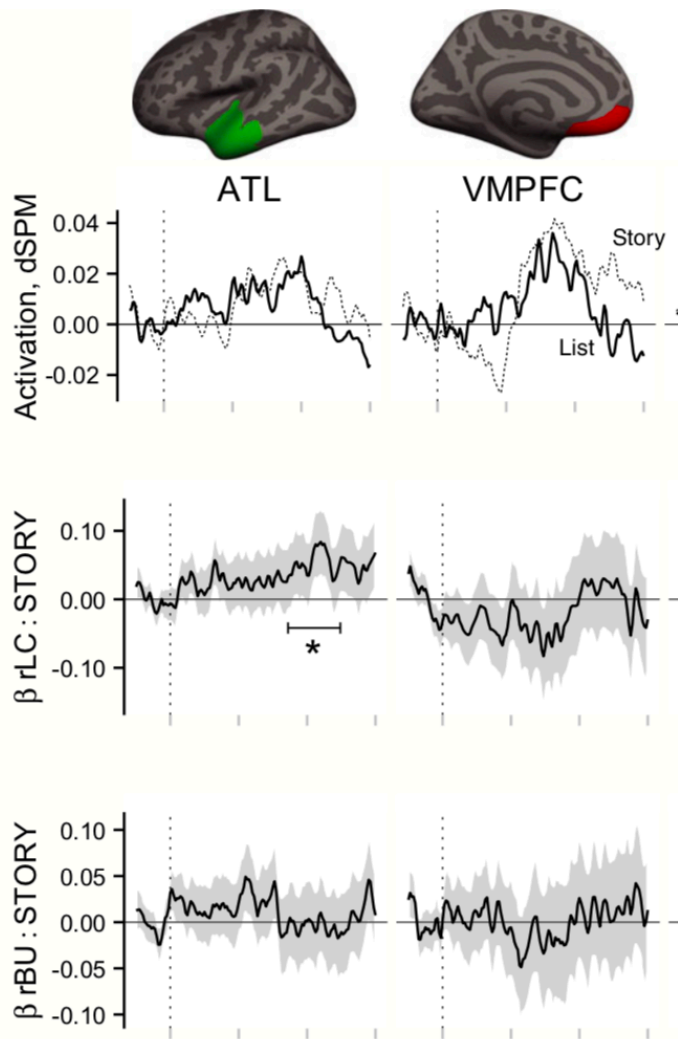


Fig. 1. (A) Three example trees for the prepositional phrases (PPs) covered by the grammar. (B) Word-by-word rule application dynamics for one example structure. Circles indicate the non-terminal node(s) that are recognized at each step according to the left-corner (LC, closed circle) and bottom-up (BU, open circle) strategies.

# Brennan and Pylkkänen 2016: parsing effort



The primary goal in this study was to identify brain areas that seemed to be tracking syntactic parsing operations.

Given previous work, they focused on left ATL, and the vmPFC.

They also manipulated the type of parsing work that they were looking for.

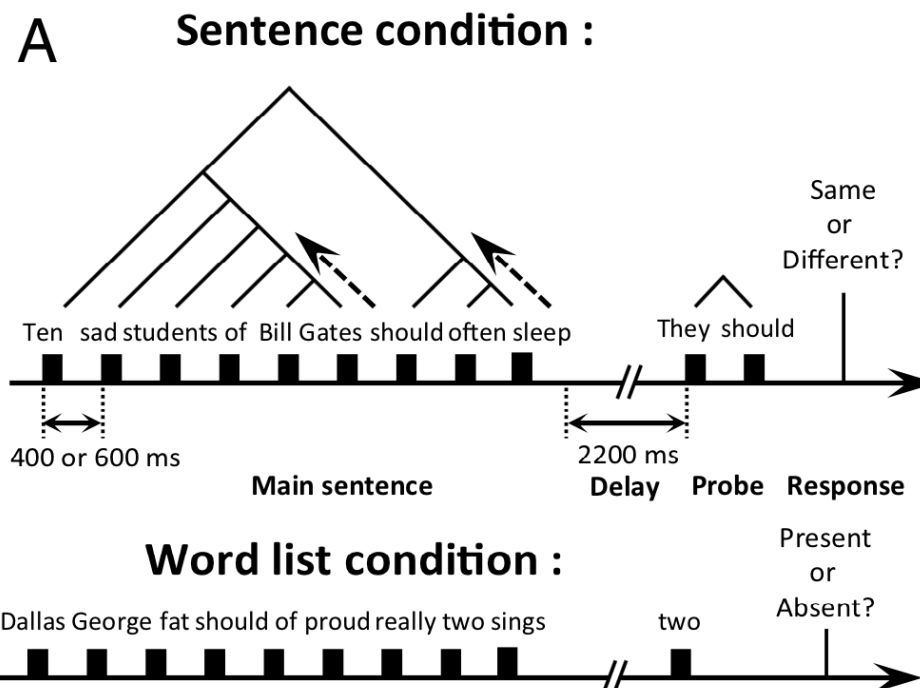
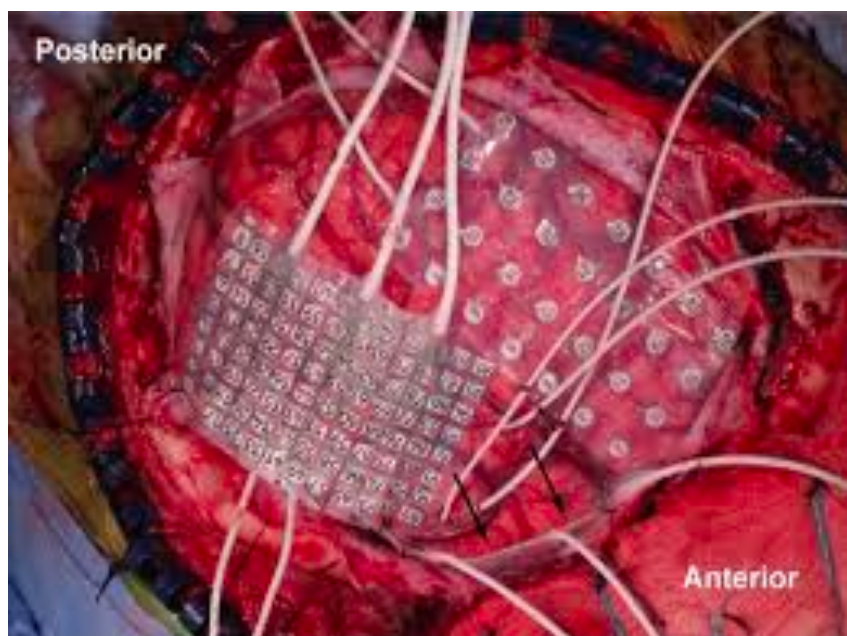
Their primary finding is significant correlations in left ATL for the left corner parser (but nothing for vmPFC or for the bottom-up parser).

The questions moving forward are (i) how to scale this up to more complex phenomena (beyond PPs), and (ii) how to compare two grammars within a single parser (likely a much smaller difference).

# Nelson et al. 2017: phrase boundaries

Nelson et al. 2017 recorded **intracranial** EEG from 12 participants (who were undergoing a separate medical procedure requiring intracranial EEG and several days of waiting) while they read sentences between 3 and 10 words long, one word at a time. Crucially, these sentences contained a subject noun phrase that varied in length.

They recorded **broadband gamma** (80-200Hz) as a proxy for neuronal activity.



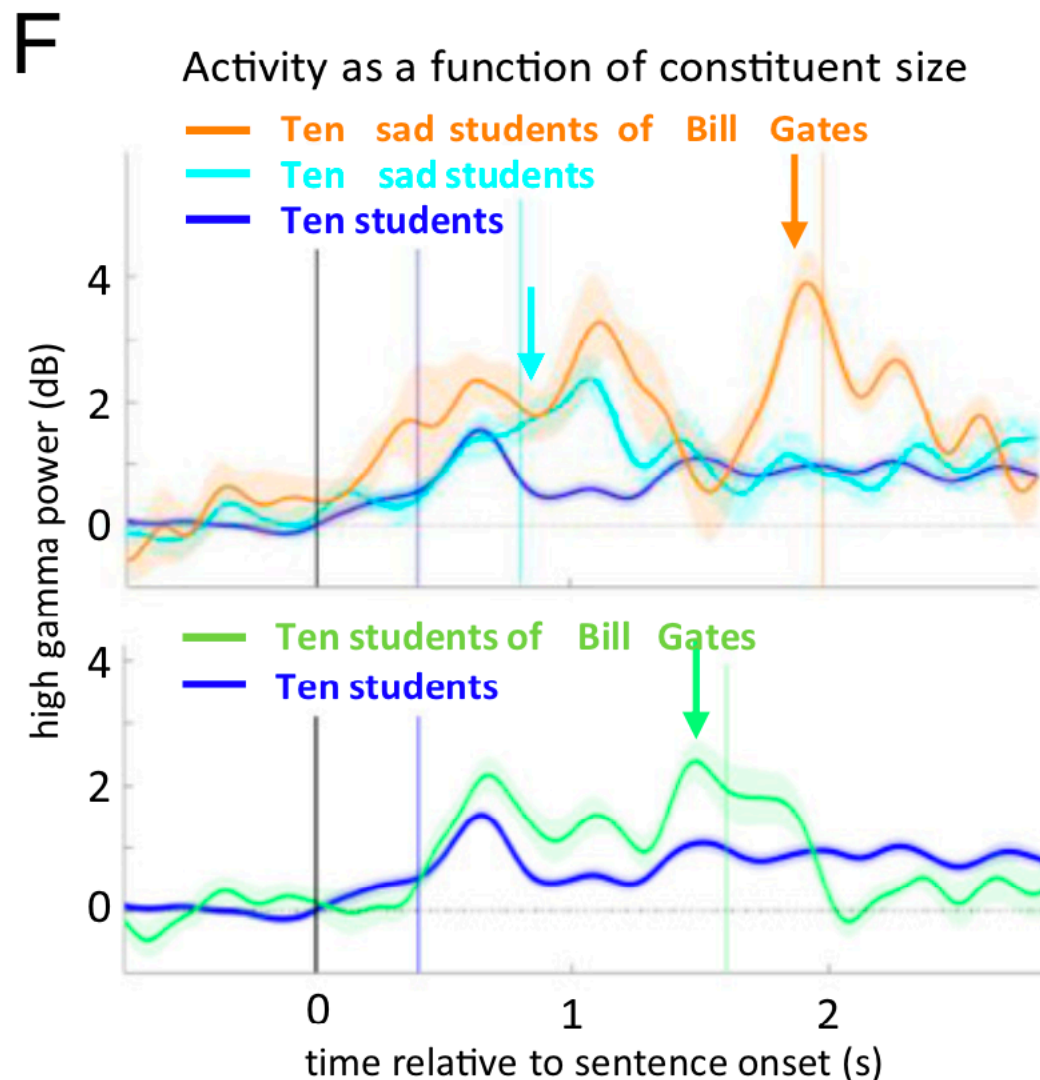


# Nelson et al. 2017: phrase boundaries

Their primary finding is that there is an increase in high gamma activity as the constituent length of the subject noun phrase increases, with a decrease in activity at both potential and actual constituent boundaries.

They interpret this pattern as a potential correlate of phrase-structure building (e.g., minimalist *merge*).

The two challenges to using this in our work are (i) ECoG can only be performed by neurosurgeons, and (ii) broadband gamma cannot be detected on the scalp.



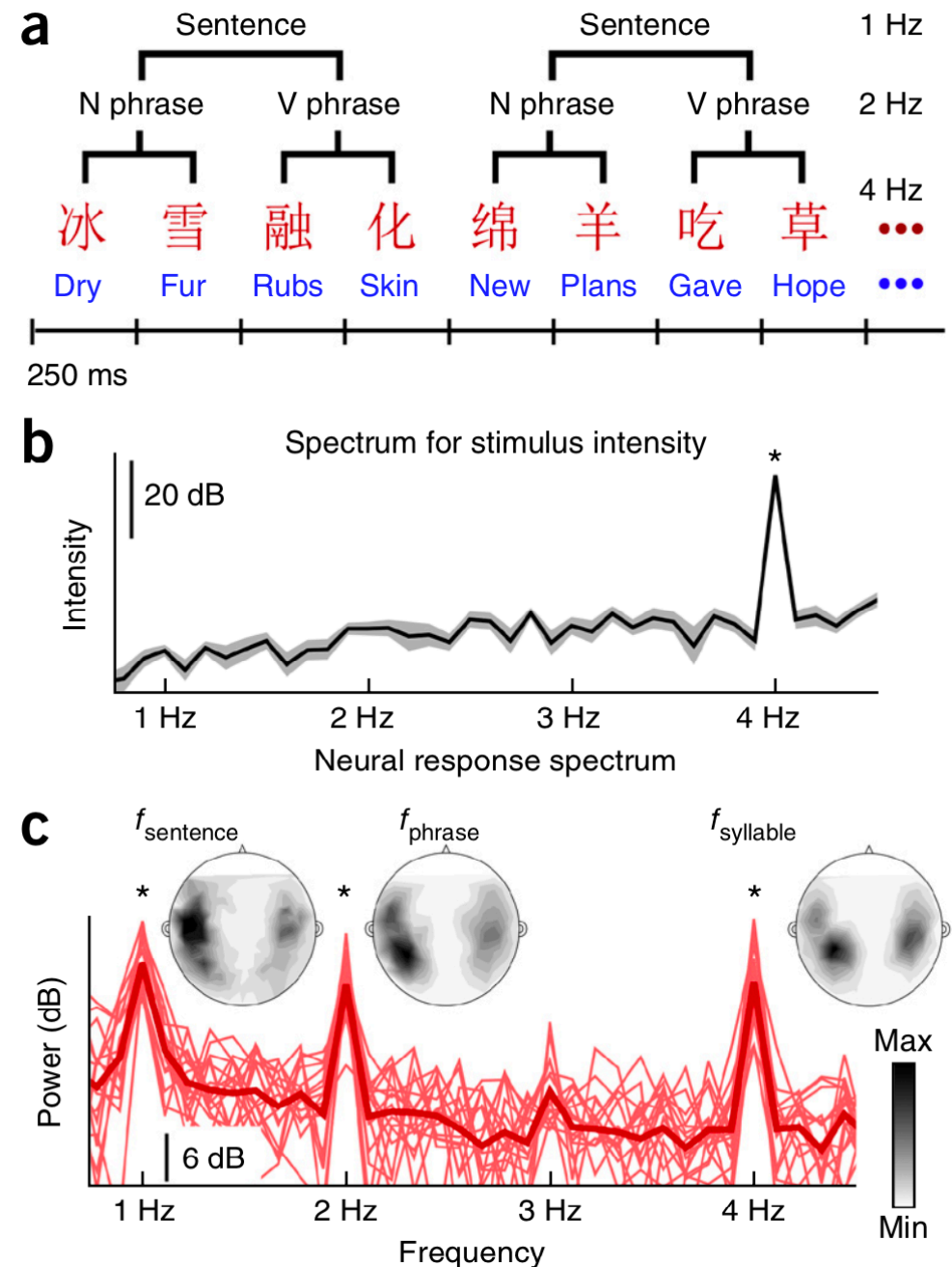
# Ding et al. 2016: hierarchical structure

Ding et al. used a method known as **steady state response** to demonstrate the existence of hierarchical (phrase structure) in the brain.

In a steady state design, the stimuli are presented continuously at a specific rate. The goal is to induce activity in the brain at that rate.

In this case, they presented monosyllabic Mandarin words, 4 words per second (4Hz), in continuous sequences such that each 4 word sequence formed a grammatical sentence in Mandarin.

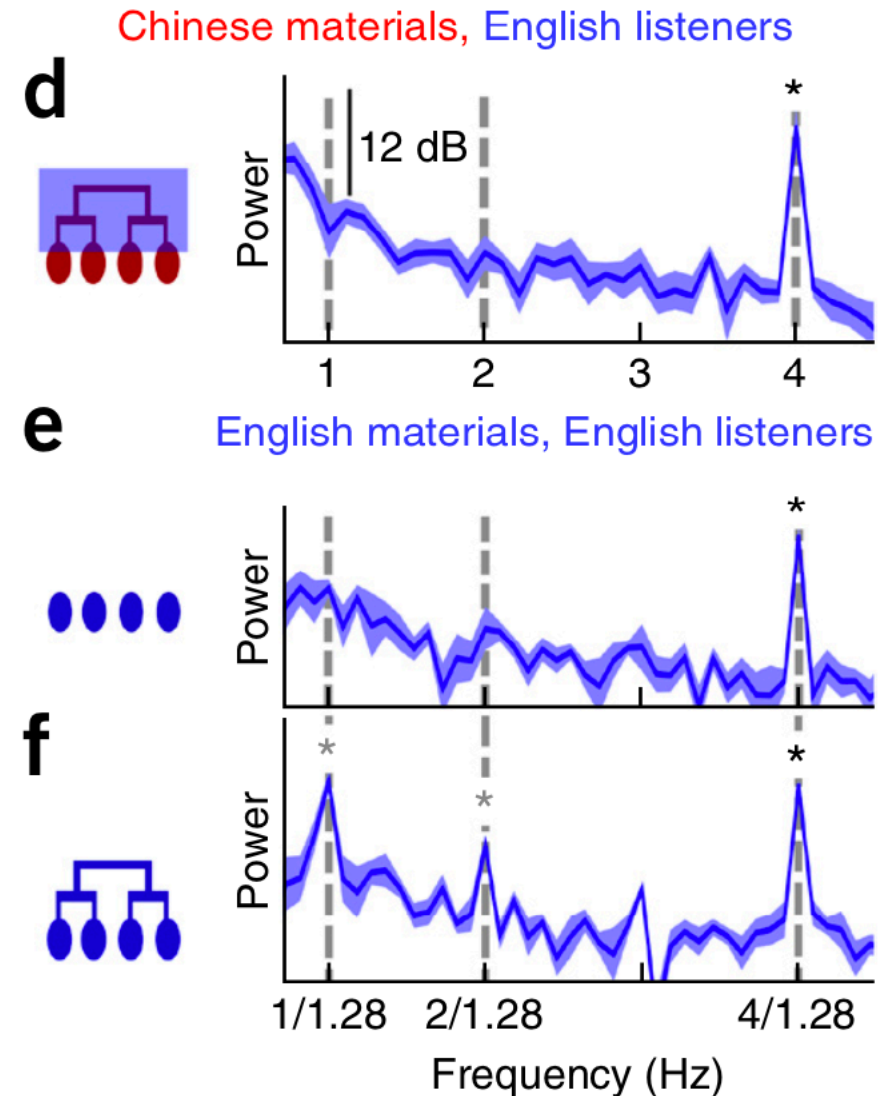
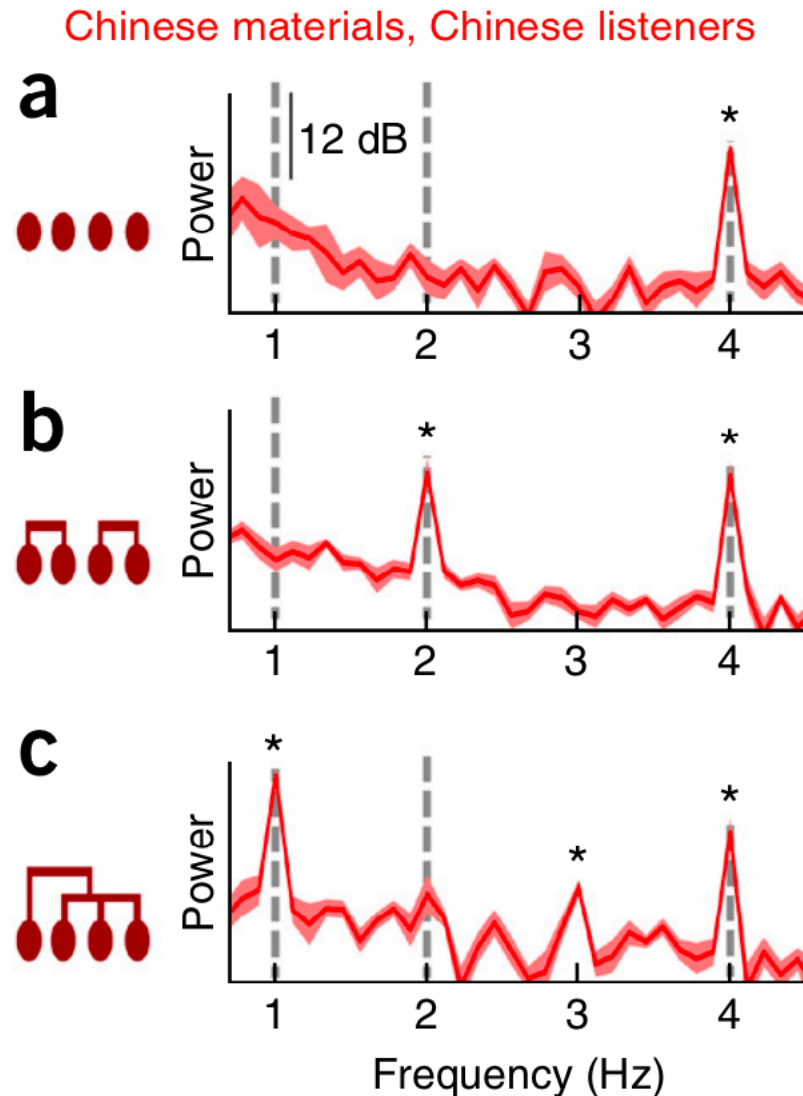
Crucially, this induced activity at more than just 4Hz — also 1Hz and 2Hz...at phrase units!





# Ding et al. 2016: hierarchical structure

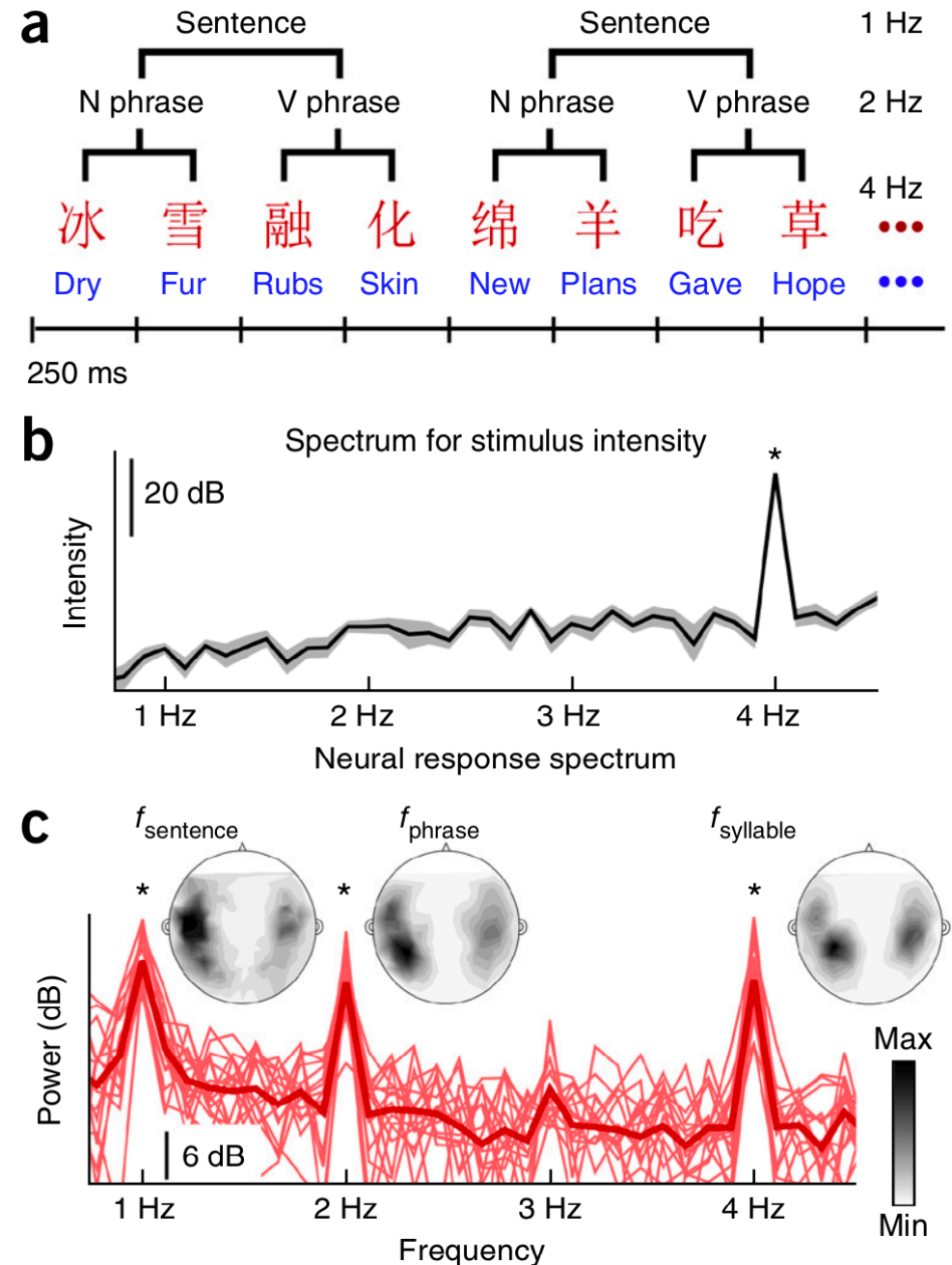
Some control conditions to really show that it is hierarchical structure that is driving the effect:



# Ding et al. 2016: hierarchical structure

The challenge for us is to figure out how to use the steady state response to probe **other aspects of syntax**.

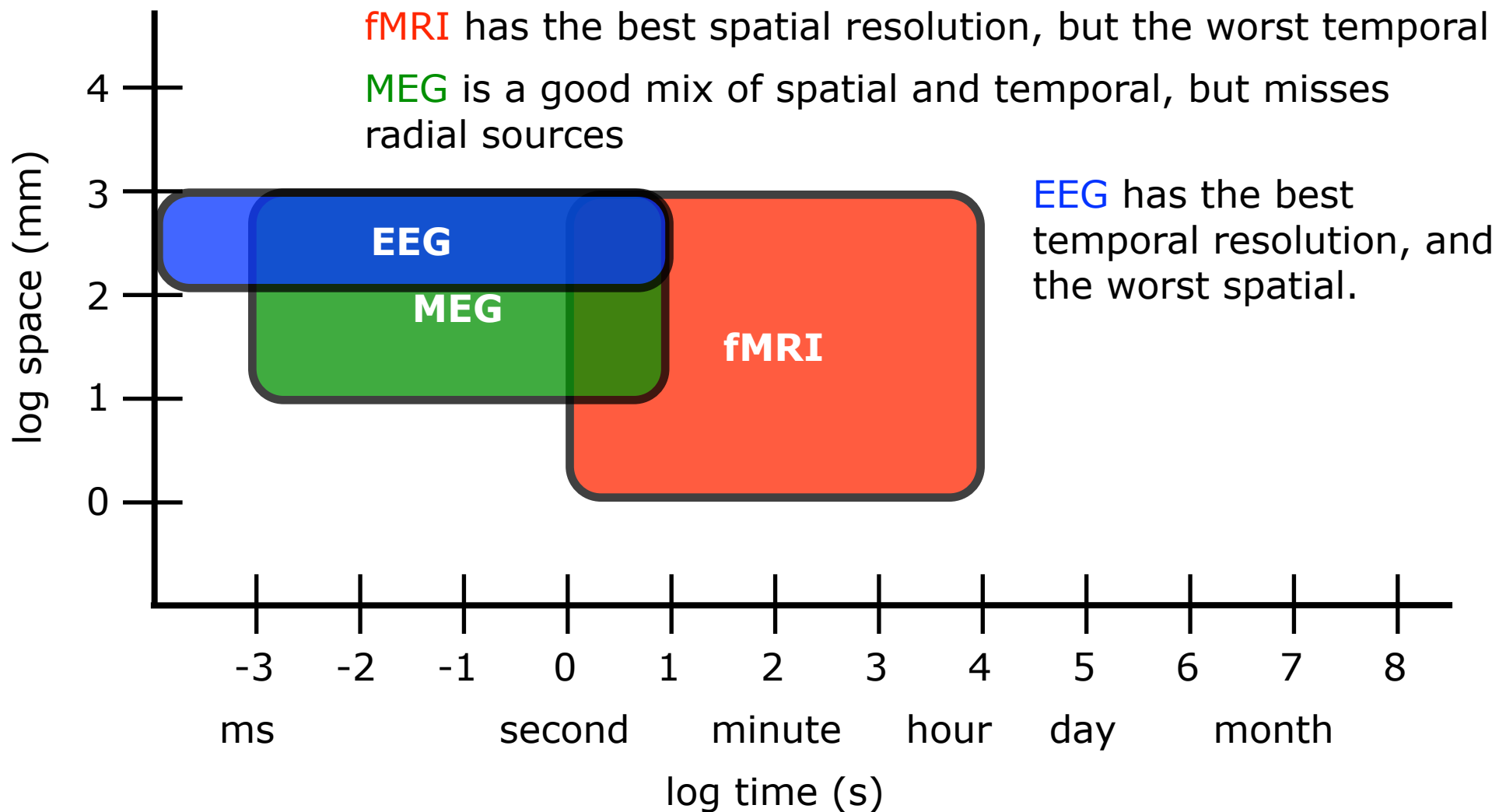
This is not easy, as it really relies on the assumption that the property in question will arise at the same time in each sentence, such that you can induce its rate of occurrence through the steady state.



# How does EEG compare to other methods?

It is sometimes useful to compare the pros and cons of different methods, to get a sense for when you might choose one over the other.

Do not worry about the details of why these methods have these properties yet. We will learn that over time in this course.



# Table of contents

1. Introduction: The big picture	Luck 1
2. <a href="#">Fundamentals</a>	Luck 2
3. Hands-on training and best practices	Luck 5
4. The ERP processing pipeline in EEGLAB + ERPLAB	Luck 6, 7, 8
1. Load the data.	
2. Filter the data (high-pass, possibly low-pass).	
3. ICA for artifact correction (optional).	
4. Re-reference the data.	
5. Add channel locations.	
6. Epoch the data.	
7. Artifact detection and rejection.	
8. Average the epochs to create subject ERPs.	
9. Average the subject ERPs to create a grand average ERP.	
10. Plotting: waveforms, topoplots, difference waves	
11. Measure amplitudes and latencies.	
12. Run statistical tests.	
5. Creating a script for EEGLAB + ERPLAB	
6. Creating a script for Fieldtrip	

# Three core content areas

Doing EEG research requires some amount of understanding of three different content areas: biology, electricity, and math. The question I'd like to address today is "How much do I need to know to get started?" Obviously, the best answer is as much as possible. But that is not realistic when one is first starting with EEG. So here I take a functional approach - When you can comfortably answer these questions (without looking the answer up), you are (probably) equipped well enough to jump into EEG research.

## **1. Biology**

Why does EEG research seem to focus on only a small number of effects? (Think about the finite list of ERPs in the literature...)

## **2. Electricity**

Why do you need at least two (and in practice three) electrodes to record a signal in EEG?

## **3. Math (trigonometry/periodic waves)**

How is it that an EEG signal can be represented in two, seemingly distinct, ways (the time-domain and the frequency-domain)?

# Biology

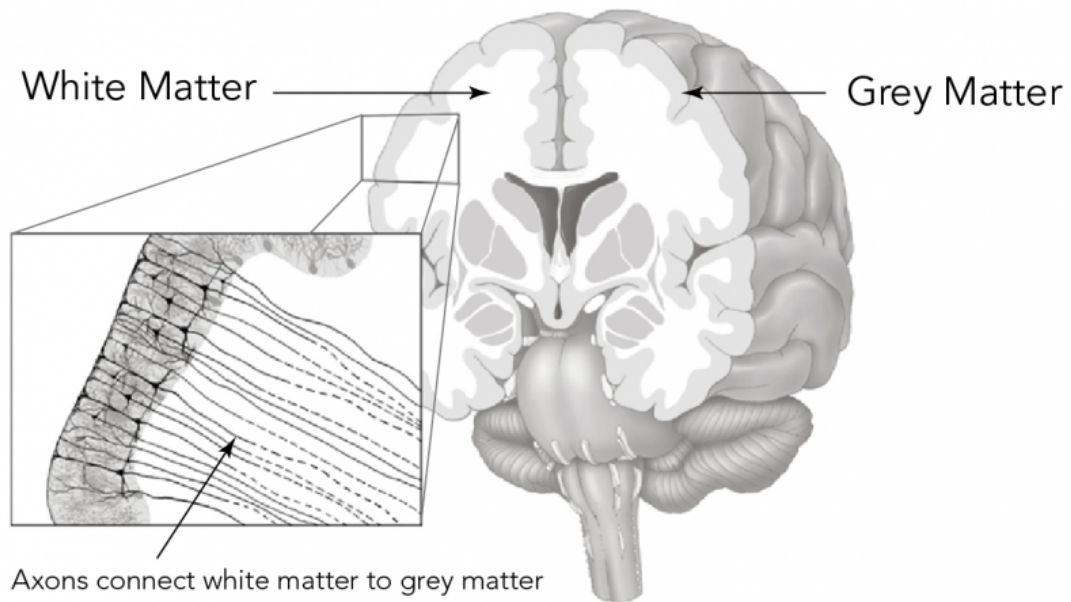
Scalp EEG measures the **summed post-synaptic potentials** of a large population of **pyramidal cells** in the **cerebral cortex** that are all **aligned in the same direction** and **firing together**.

The biology of EEG is rarely discussed in the cognitive EEG literature. This tells you something — the cognitive EEG literature is not really doing low-level neuroscience. It is doing cognitive neuroscience.

Therefore, today we will dive into this definition relatively superficially - we look at each of the important words, and try to put it together to a first approximation.

**Crucial conclusion:** If the effect you are looking for is not the result of post-synaptic potentials of a large number of aligned pyramidal cells firing together in the cerebral cortex, it probably won't be detected by scalp EEG. This is why the literature appears fixated on a small number of previously established effects, and why it is relatively rare for new effects to be reported. Scalp EEG is simply not sensitive to a (very!) large portion of activity in the brain.

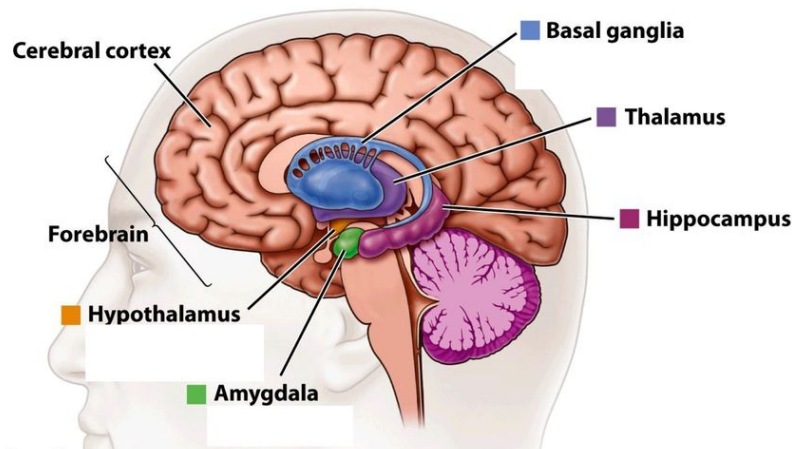
# The cerebral cortex (gray matter)



The cortex is the outer layer of the cerebrum, 2mm - 3mm thick, often called gray matter because of its coloring. It is made of the somas/bodies and dendrites of neurons. (White matter is made of the axons, and is white because of the myelin sheath.)

## Subcortical Structures

*Structures of forebrain located beneath cerebral cortex.*



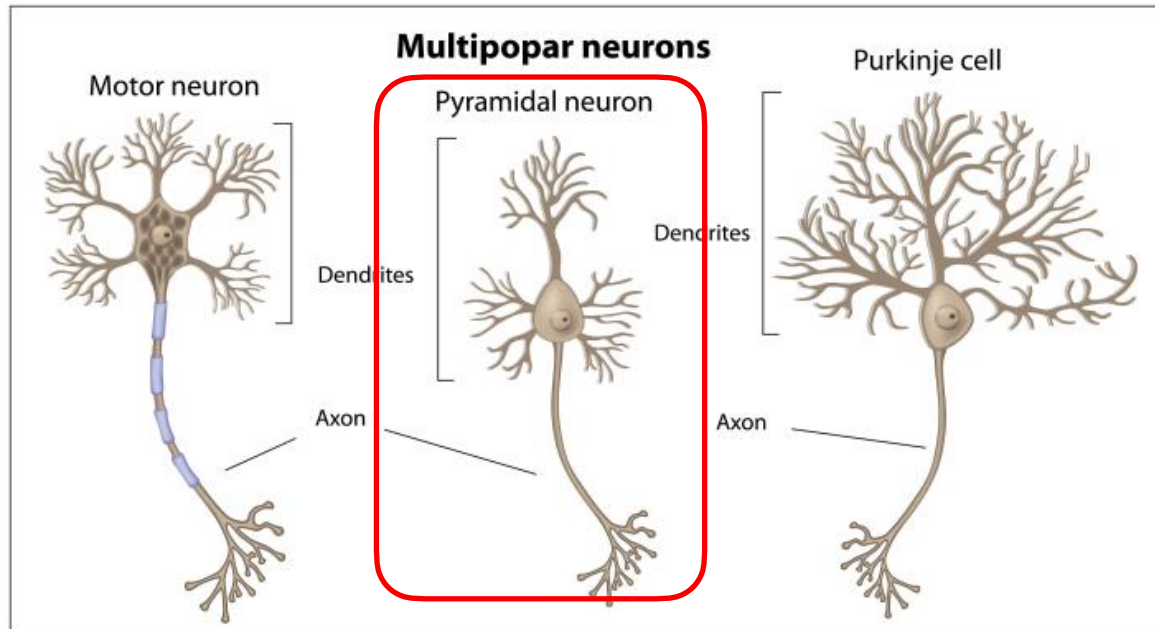
The cortex is absolutely important to cognition, but other parts of the brain, such as subcortical structures, may be relevant too.

HM had anterograde amnesia after a bilateral resection of the hippocampus.



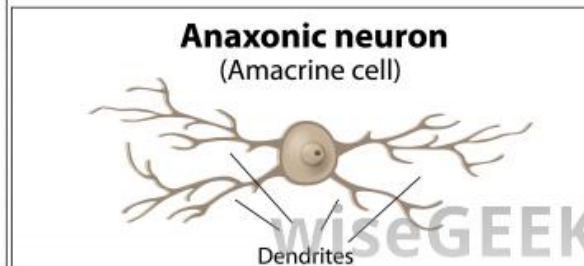
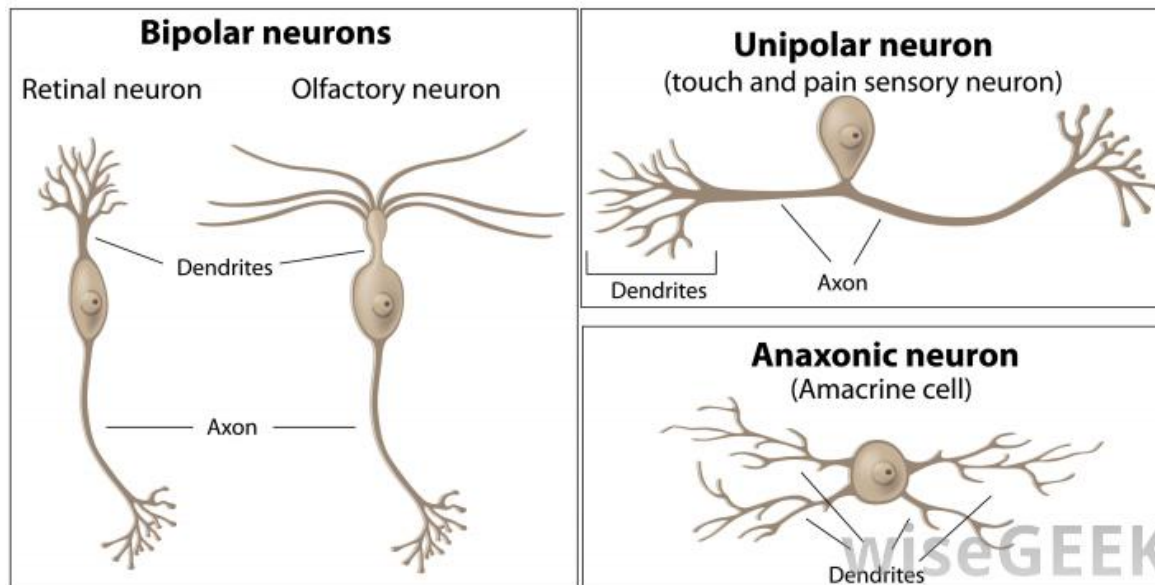
# Pyramidal cells

## Types of Neurons



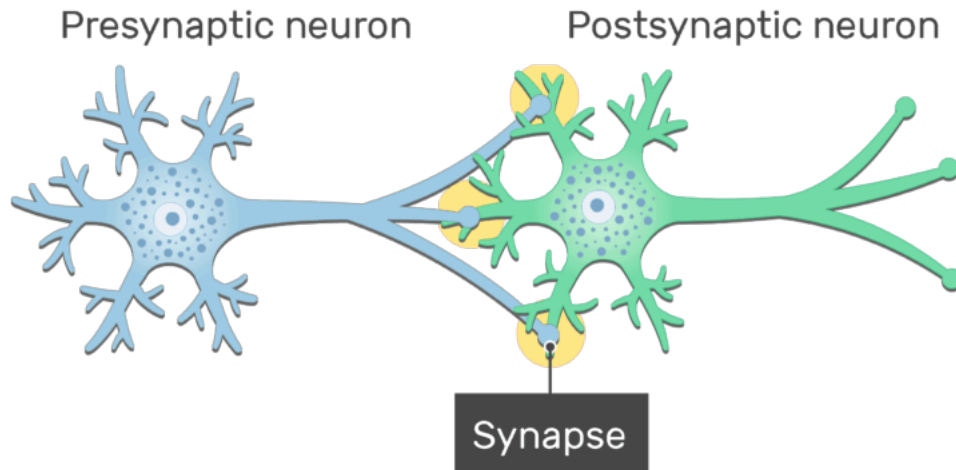
Pyramidal cells are very common in the cortex, and are undoubtedly critical for cognition.

That said, they are just one type of neuron. The others may also play a role.



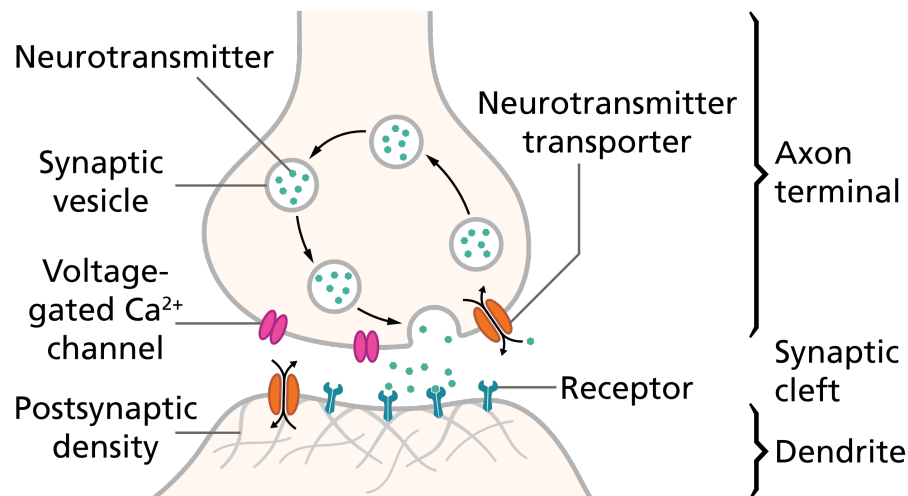


# Post-synaptic potentials



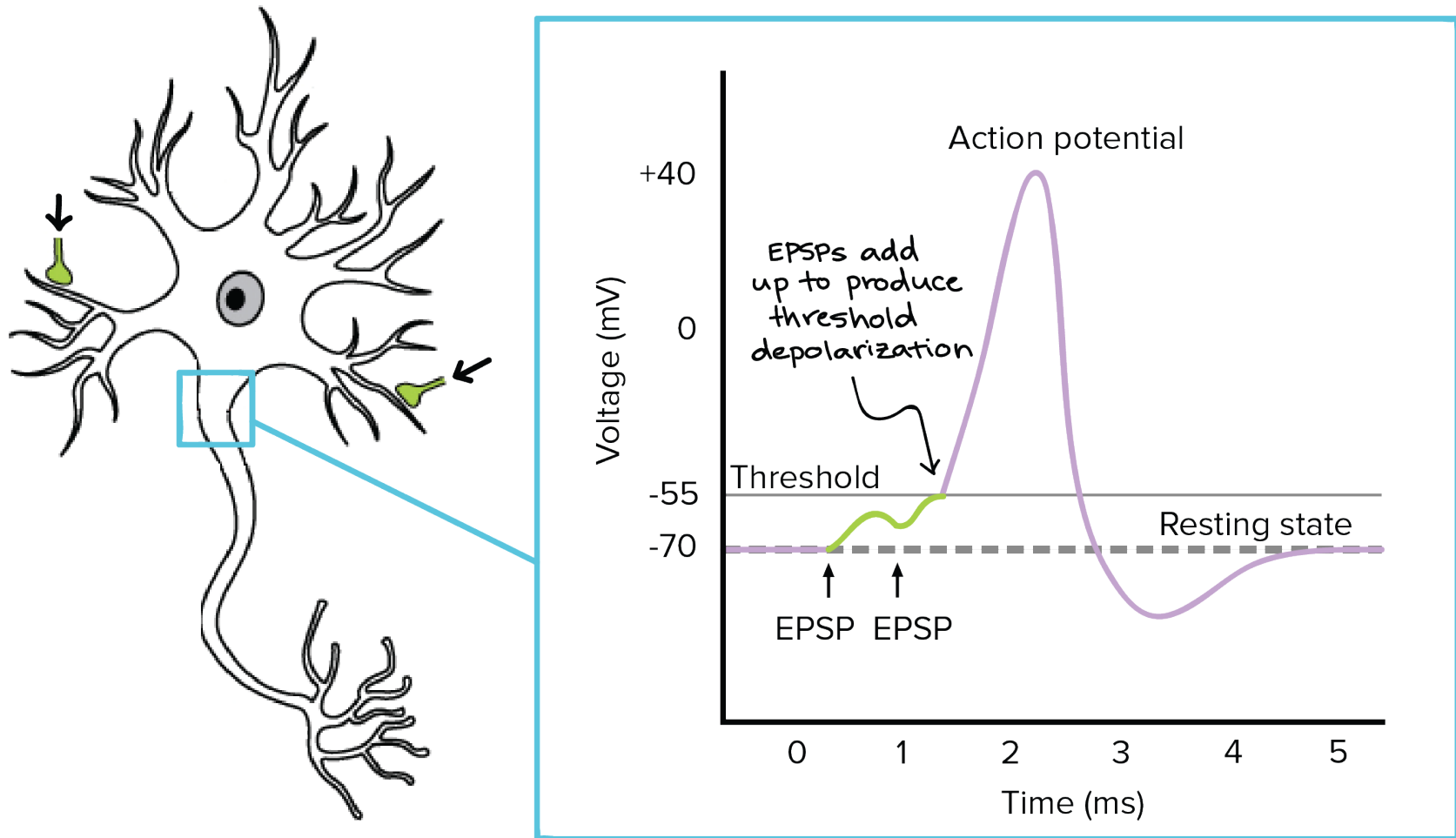
A synapse is the point where two neurons meet (typically an axon from one and a dendrite from the other).

Synapses can transfer a signal either electrically or chemically. Pyramidal cells use chemical synapses.



Here is a diagram for a chemical synapse. I am not an expert on this, so I won't even try to explain it (because I would fail).

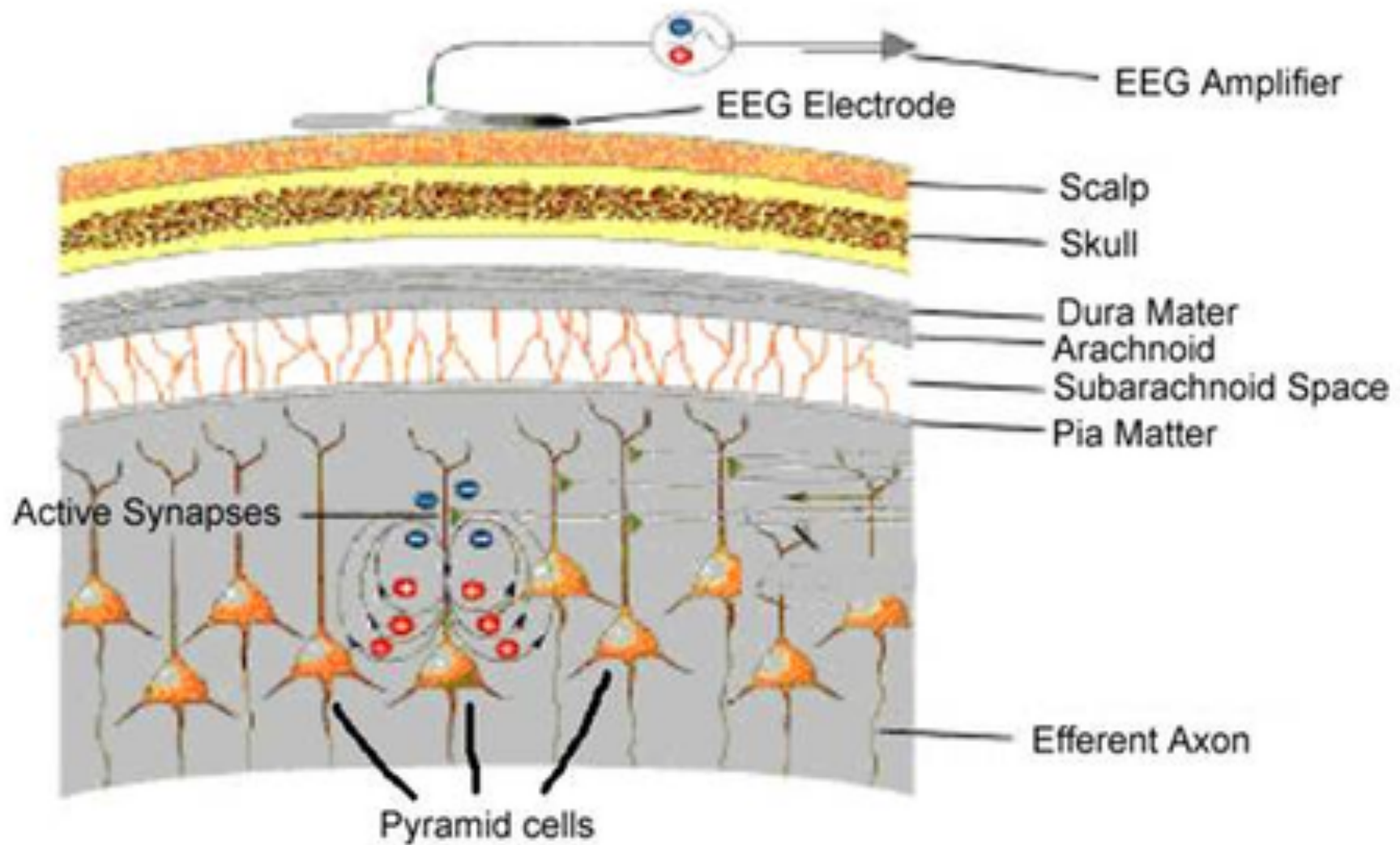
# NOT action potentials



Scalp EEG is not recording action potentials. When you think of a neuron firing, action potentials are probably what you are thinking of - the electrical activity inside of the neuronal body. For that, you would need intracellular recordings.

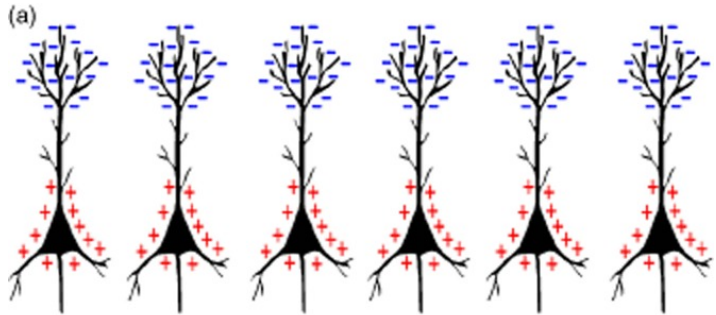
Instead, scalp EEG measures the PSPs that trigger the neuron to fire (or not).

# Getting to the scalp

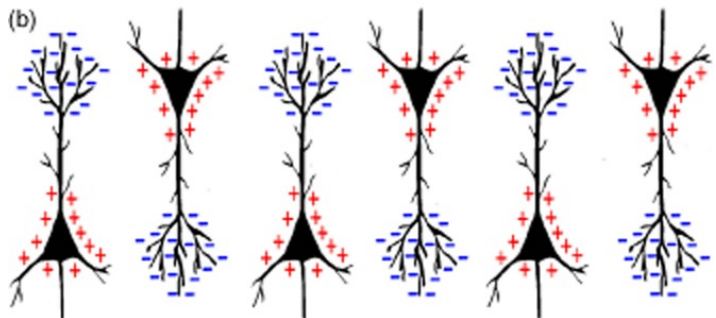


There is quite a bit of material between the cortex and the scalp electrode. That is why scalp EEG can only detect relatively large potentials.

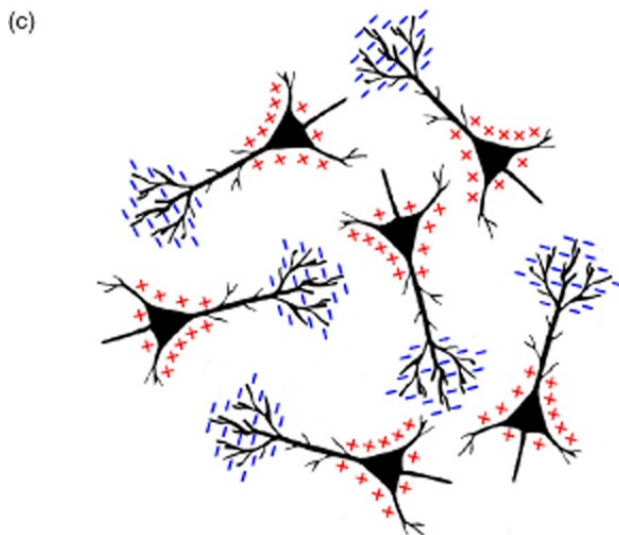
# Large requires spatial alignment



Measurable at the scalp



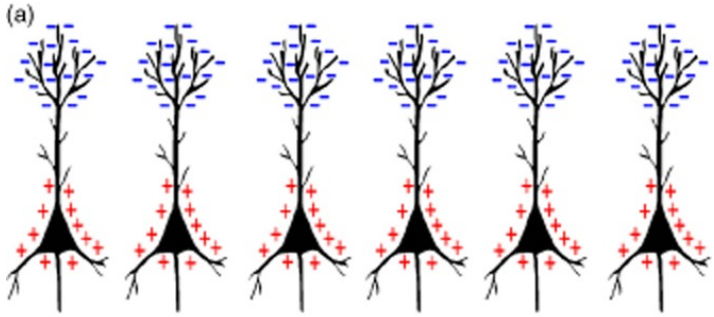
Not measurable at the scalp



Not measurable at the scalp

These images are from Jackson and Bolger 2014 (on the website).

# Large requires synchrony

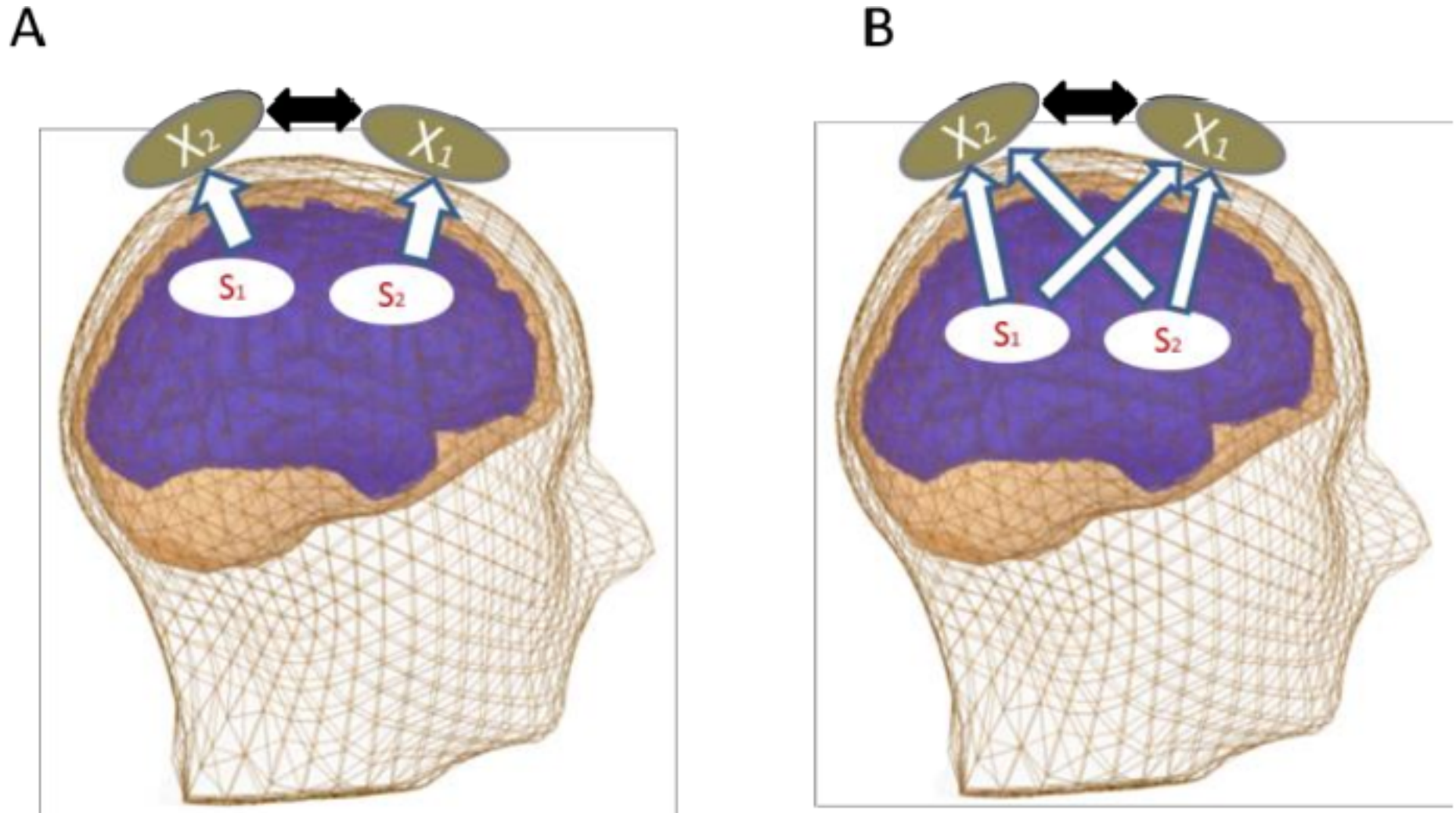


Even if you have alignment, it won't help you to generate a large signal if you don't also have firing synchrony.

Fortunately, neurons like to fire together. That is sort of their thing (and why they send signals to each other in the first place). But we just need to remember that when we see a signal large enough to make it the scalp, it is because a relatively large population of neurons fired together.



# Volume conduction



The signal we see is the summation of all of the signals generated in the brain. This is because of volume conduction — all of the matter in our heads is conductive. (Figure from Hassan and Wendling 2018.)

# Some links for the biology of EEG

Khan academy has a sequence of short videos that cover the anatomy and function of neurons:

<https://www.khanacademy.org/science/biology/human-biology/neuron-nervous-system/v/anatomy-of-a-neuron>

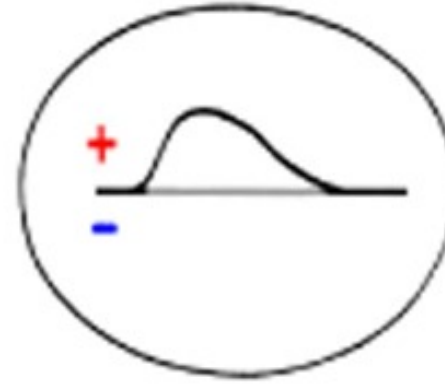
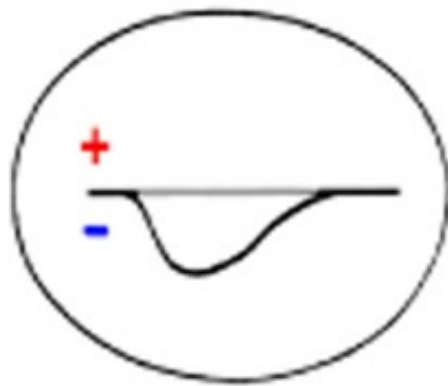
Here are two videos from the University of Oklahoma that explains excitatory post-synaptic potentials and inhibitory post-synaptic potentials (in detail):

**Excitatory:** <https://www.youtube.com/watch?v=PTgyBEmC-Es>

**Inhibitory:** <https://www.youtube.com/watch?v=jEvS7GrKP2U>

These two will be relevant if you want to know what the polarity of the EEG signal is telling us (spoiler: it is ambiguous).

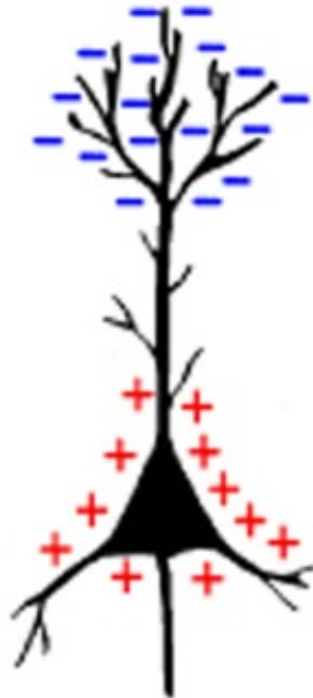
# Extra slide: interpreting polarity



Excitatory signal  
(EPSP) received  
at the dendrites

-or-

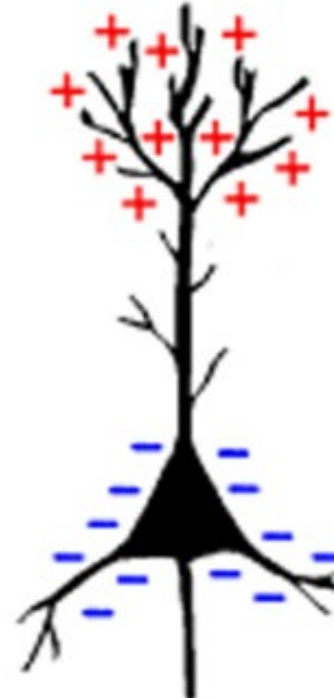
Inhibitory signal  
(IPSP) received  
near the soma



Inhibitory signal  
(IPSP) received  
at the dendrites

-or-

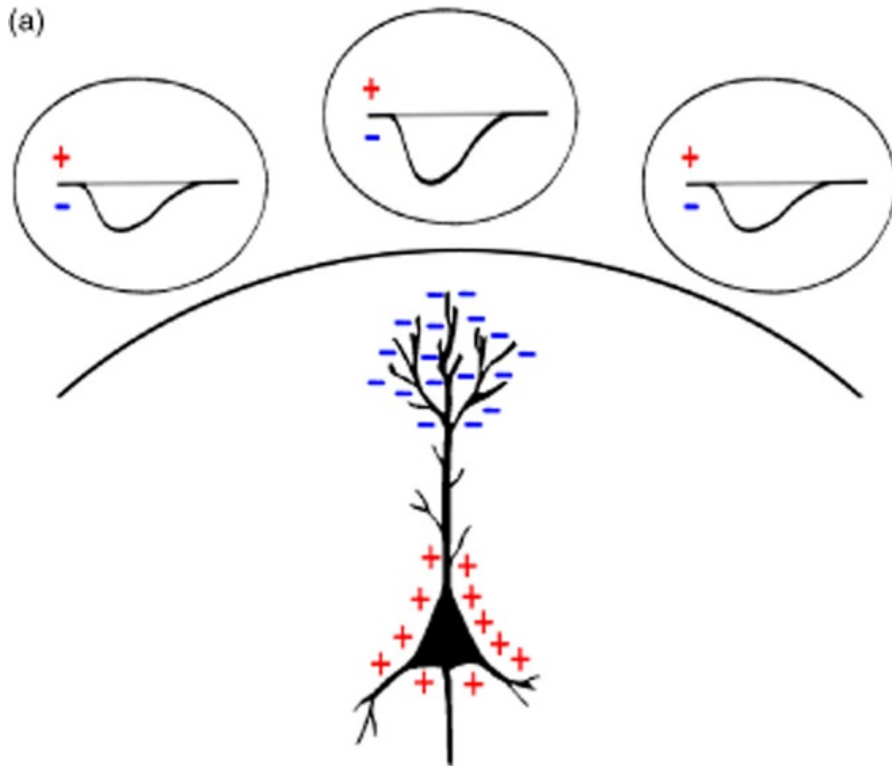
Excitatory signal  
(EPSP) received  
near the soma



These are from Jackson and Bolger 2014



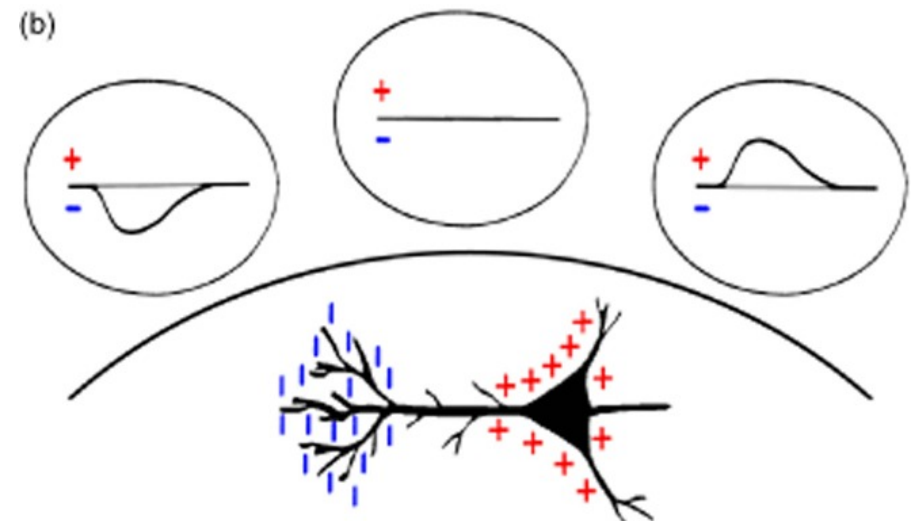
# Extra slide: interpreting polarity



**Tangential** dipoles are oriented along the scalp. This means the entire dipole (negative and positive) contributes to scalp EEG. If there were only one dipole active, this would lead to a gradient from negative to positive across the scalp.

**Radial** dipoles are oriented toward the scalp. This means one side of the dipole (negative or positive) contributes most of the signal. If there were only one dipole active, this would lead to a single polarity broadly distributed over the scalp.

These images are from Jackson and Bolger 2014 (on the website).



# Electricity - the water metaphor

EEG measures changes over time in **electrical potential** on the scalp.

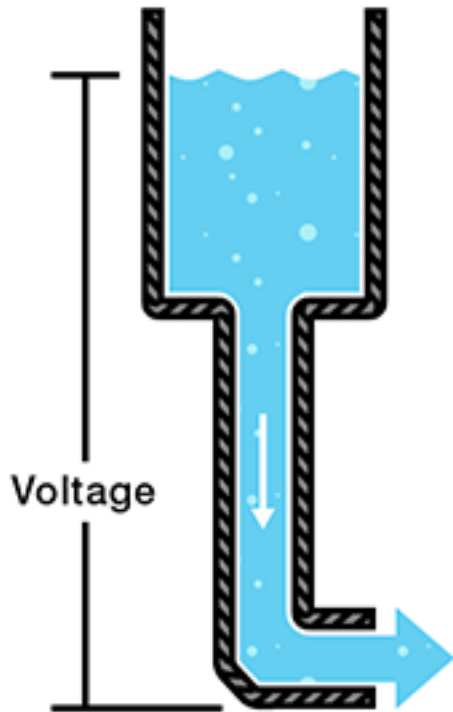
**Electrical potential** is named that because it is a type of potential energy (the concept from physics). It is the potential for electrons to flow from one location to another. It is also called **voltage** because the unit of measure is the **Volt**.

The best way to get an intuition for the concept of electrical potential is to imagine **a system of water in pipes**.

We use this metaphor because we humans tend to have physical **intuitions** about water flowing in pipes. But we don't have physical intuitions about electrons moving between the atoms of metal in a conductor.

There are a number of websites that will use this metaphor. I'll use this one today, but feel free to look for others: <https://learn.sparkfun.com/tutorials/voltage-current-resistance-and-ohms-law>

# Electrical potential = water pressure

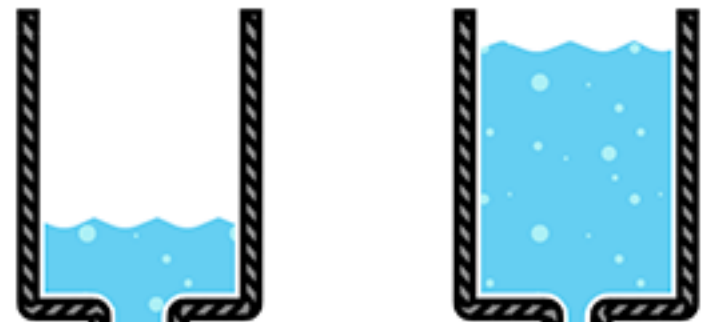


Potential is measured between two locations - it is the potential for water to flow between the two locations. Here it is the top of the tank and the spot at the bottom.

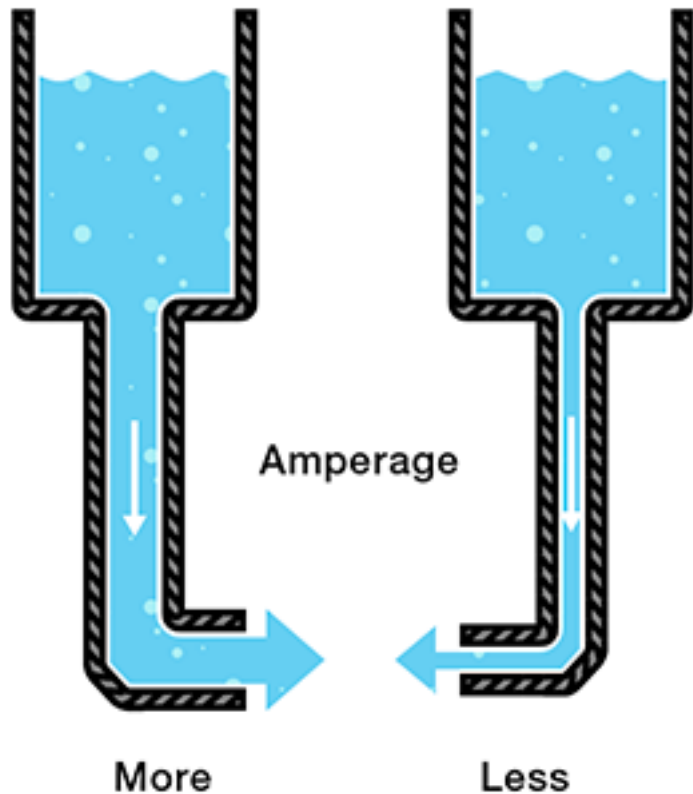
Electrical potential is measured between two locations - it is the potential for electrons to flow between the two locations, like two positions on the scalp.

The more water in the tank, the higher the pressure, so the greater the potential to flow.

The more electrical charge, the higher the voltage, so the greater the potential for electrons to flow.



# Electrical current = water flow



Both tanks have the same amount of water (so, same charge).

Both tanks are the same height, so there is the same potential (so, same voltage).

But the left tank has a wider spout. So the flow rate of the water will be faster on the left. In electrical terms we call this **current**, The unit of measure is Amp (Ampere). Current is sometimes also called amperage.

Why is the flow slower on the right? Because the tube is smaller. In electrical circuits this is called **resistance**. The unit of measure of resistance is Ohm.

There is a lawful relationship between voltage, current, and resistance. It is called Ohm's law:

Voltage = Current x Resistance

-or-

Current = Voltage/Resistance

# Why do we need at least two electrodes?

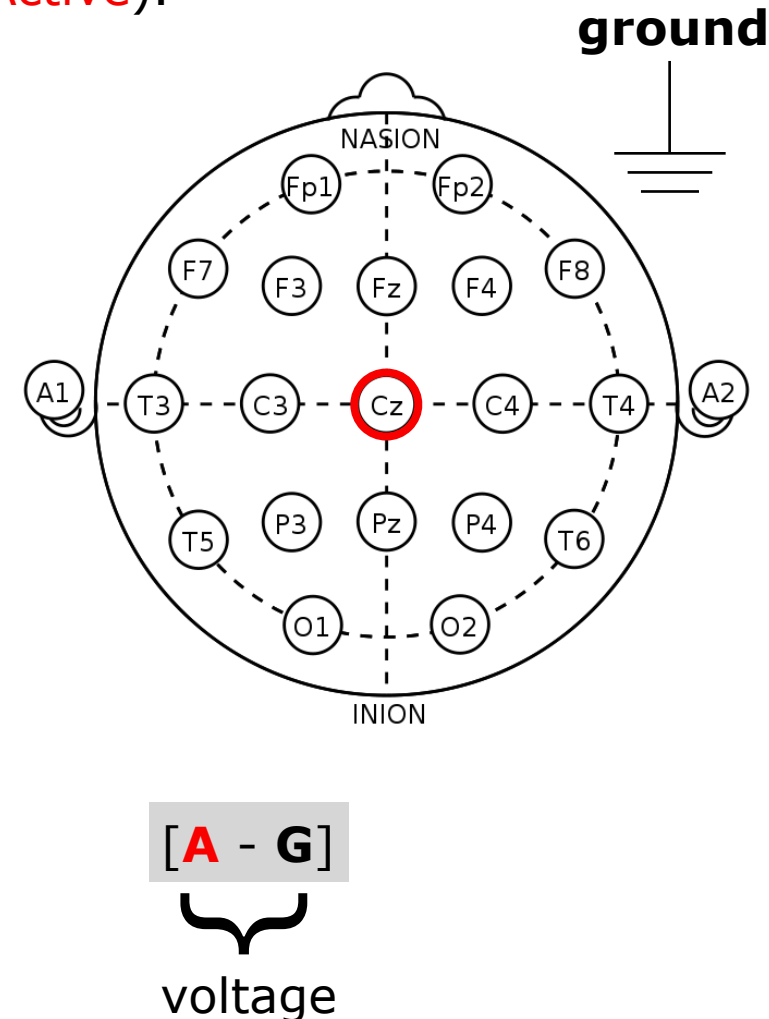
Because EEG measures scalp potential, that is, it measures voltage. And potential **just is** the relationship between two locations!

Let's call the electrode we want to measure **A** (for **Active**).

There is no such thing as "the voltage at A". If someone says that, they must mean it as a shorthand for "the voltage at A relative to **X**".

In electrical circuits the second location is typically called **ground** (it could be the actual ground, as in the earth, but is often just a part of the circuit itself).

The crucial thing to remember is that you cannot look inside this equation. It is the measurement of voltage:



# Why do modern amplifiers use three?

You might think that amplifiers just calculate the voltage between A and G, and amplify that:

$$[A - G]$$

In an ideal world, we would. But physical devices have noise in them. So if we amplified that, we'd have a noisy signal. (And that noise could be larger than the very small potentials generated by the brain.)

$$[A - G + \text{noise}]$$

Now, let's throw in a third electrode, called **R** for **reference**. We can calculate the voltage for A relative to G, and separately for R relative to G.

$$[A - G + \text{noise}]$$

$$[R - G + \text{noise}]$$

Now, for a critical assumption: the noise in these two independent voltage measurements will be **roughly equal** (though, obviously not perfectly). That is because the noise comes from the system itself, and the system is part of both measurements. So if we subtract them, we will end up the value  $[A - R]$  without much noise!

$$[A - G + \text{noise}]$$

$$- [R - G + \text{noise}]$$

---

$$[A - R]$$

# All about the reference

The amplification equation:  $[\text{active} - \text{ground}] - [\text{reference} - \text{ground}]$

Yields the voltage:  $[\text{active} - \text{reference}]$

The practical consequence of this is that you need to choose your reference electrode when doing data analysis. And, you need to pay attention to which reference electrode was chosen when you read an EEG paper.

## Some options

tip of the nose	left earlobe	right earlobe	average of mastoids
	left mastoid	right mastoid	average of all electrodes

We will discuss this a bit more when we do data analysis ourselves!

# Measuring through time requires sampling

The voltage on the scalp changes continuously through time. However, our physical amplifiers have to convert that continuous signal to discrete measurements. This is called **sampling**.

We will discuss sampling in more detail when we look at the math section in a few slides. For now, I just want to prime you for it.

The number of measurements that you take in a given time period is called the **sampling rate**. The typical unit of measure is Hertz (Hz), which is “samples per second”.

$1000 \text{ Hz} = 1000 \text{ samples per second} = 1 \text{ every millisecond.}$

Modern amps can sample quite fast — up to 20,000 Hz in standard amps; up to 250,000 Hz in the fanciest amps.

**Practical consideration:** You need to choose a sampling rate before you start recording EEG. This will impact the size of the computer files (more samples = more data). This will impact the time it takes to run stats (more samples = more time). For ERPs, this is about it. But for time-frequency analyses, we will see that the sampling rate directly constrains the frequencies you can analyze.

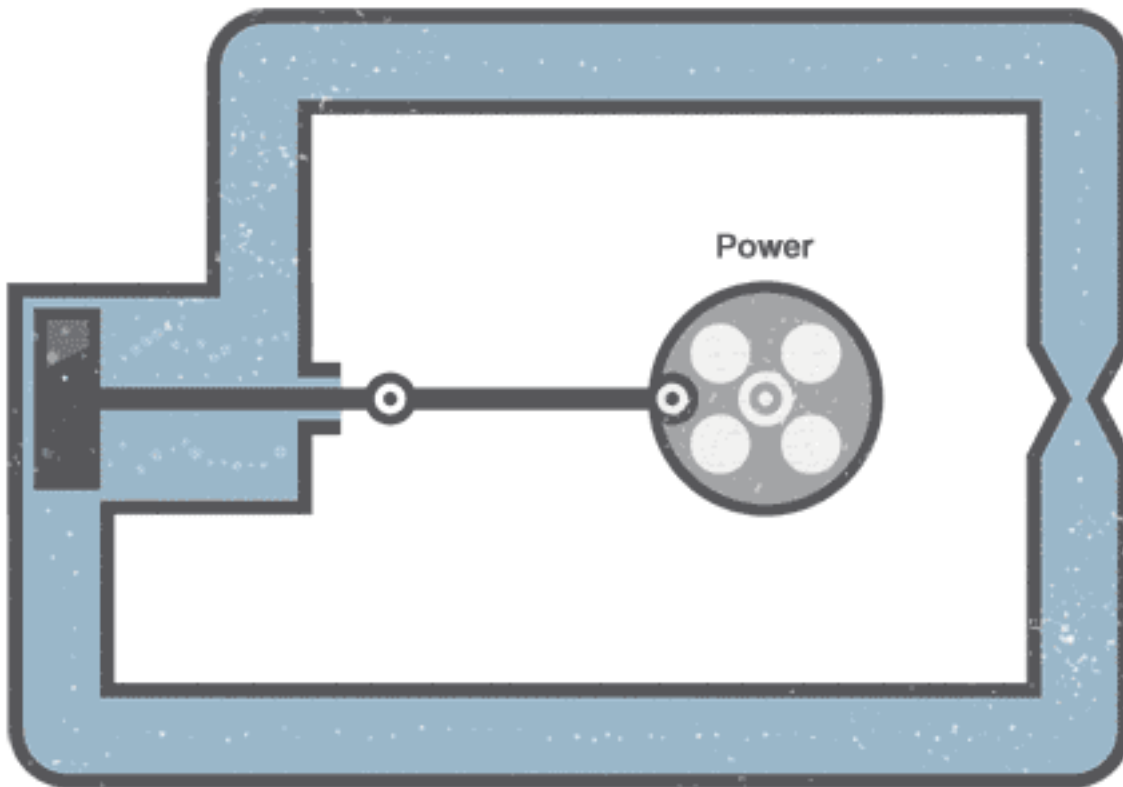


# Alternating Current vs Direct Current

Direct current (DC) is basically what we've been imagining so far as we've discussed electricity. The current flows in one direction.

Alternating current (AC) is when the current alternately flows in both directions.

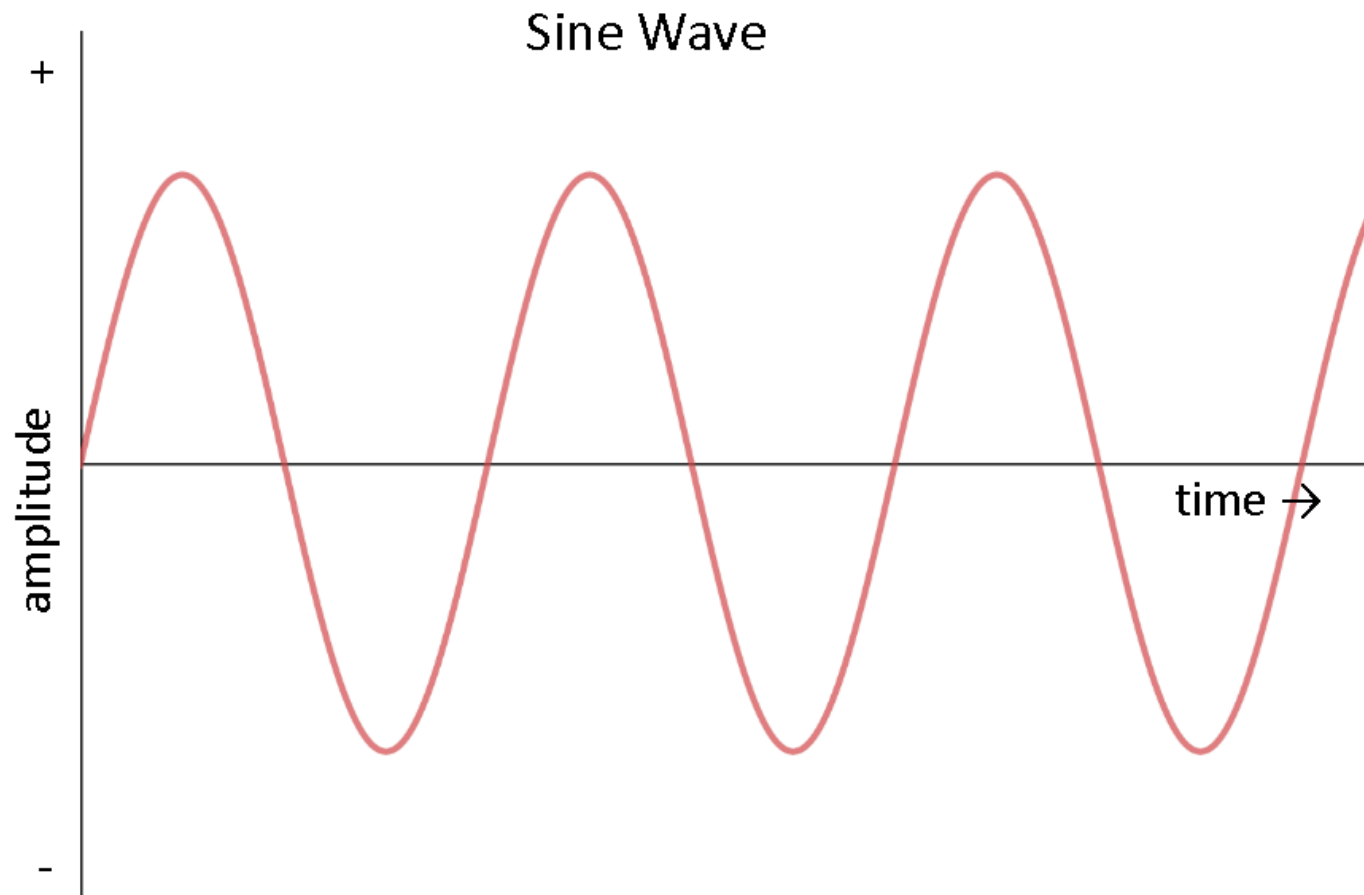
## Alternating Current: The Water Analogy



The electrical signals from individual neurons are DC. But neurons themselves tend to oscillate - firing repeatedly at a certain rate. This means that we primarily conceptualize EEG signals as AC.

# AC means wave math!

If you plot the potential of an AC signal as it changes over time, you will get a periodic function based on the rate at which the direction of the current alternates. It will look like a sine wave!



And that tells us that the math we need to know to analyze EEG signals is wave math (trigonometry and maybe a little linear algebra).

# Some links for the basics of electricity

Electricity is also rarely discussed in the cognitive EEG literature... but it will help you understand various aspects of the process of collecting and analyzing EEG signals.

**Tutorial on basics:** <https://learn.sparkfun.com/tutorials/voltage-current-resistance-and-ohms-law>

**AC-vs-DC:** <https://learn.sparkfun.com/tutorials/alternating-current-ac-vs-direct-current-dc> (the chemical electricity of neurons is like DC, but the oscillations we see in the EEG signal are like AC)

**Textbook:** <https://www.allaboutcircuits.com/textbook/> (especially worth reading about current flow vs electron flow notation)

**For the future:** We may want to explore the relationship between resistance (a DC concept) and impedance (an AC concept). Impedance is a complex number (magnitude and phase) - the real part is resistance and the imaginary part is reactance. But I haven't found an easy-to-digest tutorial on this yet.

# Waves and signal processing

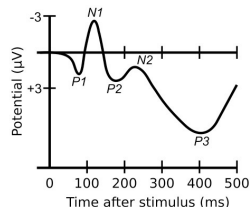
The signal that is recorded during EEG is a time-varying measure of voltage. It tends to have an oscillatory (wave) structure. So all of the analysis that you will do with EEG is based on the mathematics of waves (trigonometry).

This is the absolute best tutorial on waves and signal processing that I have found. I suggest we work through it together up to the section introducing the Fourier transform:

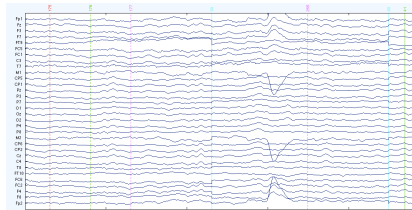
<https://jackschaedler.github.io/circles-sines-signals/>

**Crucial concept we are trying to get to:** The EEG signal can be represented in two different (but entirely equivalent) ways. We will use both of these representations!

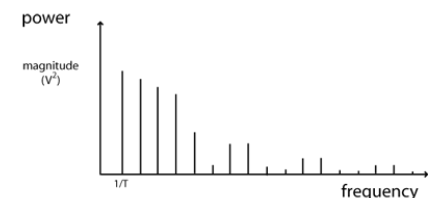
Time domain



EEG



Frequency domain



# The components of an EEG system

Stimulus computer



Acquisition computer



Cap and Electrodes



Amplifier and Battery



# The components of an EEG system

## Stimulus computer



This is the computer that presents the experiment to the participant, and records their behavioral responses. Its exact configuration depends on the types of experiments that you run in your research.

EEG requires **event codes** - markers in the EEG recording that indicate when a stimulus occurred (so that you can analyze the data relative to that event). This means that stimulus computers in EEG experiments must have a way to send event codes to the amplifier.

The standard way to do this is with a **parallel port** — the experimental software sends an electronic pulse out of the parallel port and into the amplifier. Parallel ports have excellent timing, so this is often a good option.

Other options include USB (timing is not great), or direct physical input — a photodiode or microphone that detects the stimulus directly.

# The components of an EEG system

## Stimulus computer



This is the computer that presents the experiment to the participant, and records their behavioral responses. Its exact configuration depends on the types of experiments that you run in your research.

## Other things it will likely need

### **Experimental Software:**

Psychopy, E-Prime, Matlab, etc.

### **Visual hardware:**

A good video card, and a monitor with fast response times (look for monitors marketed to gamers).

### **Audio hardware:**

A good sound card, and in-ear earphones (etymotic is the standard company).

### **Response collection:**

A response box or gamer keyboard.

# The components of an EEG system

## Acquisition computer



This is the computer that records the data from the amplifier. It typically does not require anything special other than a large hard drive and a USB port.

This is the computer that will run the data acquisition software from the EEG company.

My personal taste is to make sure that this computer also has a gaming monitor because gaming monitor often come with special buttons that allow you to change to pre-set brightness levels. If you are recording in the dark, you will want to be able to quickly change to a lower brightness level.

I also like to make sure the video card has two outputs. This lets me send the output to the stimulus computer during gelling so that I can see the measured impedances on the participant's computer (which is closer to where I am standing!).



# The components of an EEG system

## Amplifier and Battery



The Brain Products amplifiers that we use run on battery power. This is to increase safety (no connection to mains power), and portability. The EEG amplifier both amplifies and digitizes the signal (you can set the sampling rate through the software).

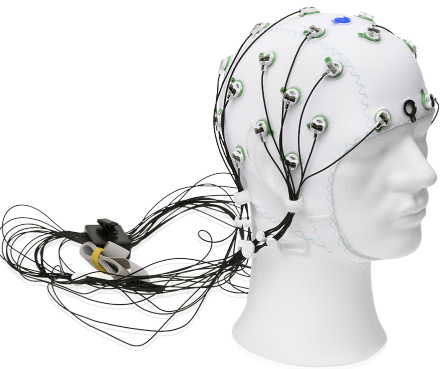
Now is a good time to start memorizing the differential amplification equation that modern EEG amplifiers use:

(Active-Ground) — (Reference-Ground)

Keep this in mind whenever you talk about voltage at A. And keep it in mind later when we talk about choosing a reference electrode... and re-referencing the data to change the reference electrode.

# The components of an EEG system

## Cap and Electrodes



The Brain Products system we use separates caps from electrodes. But in some systems they are integrated into one piece.

## **Brain Products Active Electrodes**

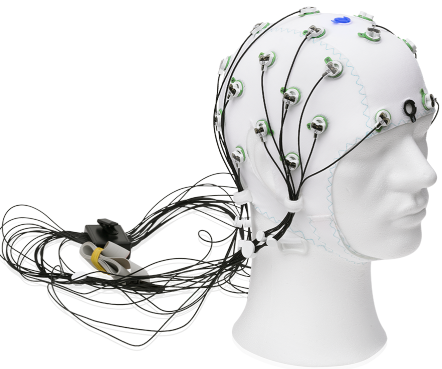
An electrode is simply a conductive element like a wire. So the most basic electrode we could have would simply be a piece of metal touching the scalp.

All modern EEG systems use silver as the metal because it is an excellent conductor. They actually mix silver and silver chloride together in a process called sintering (so the electrodes are sintered). This results in a cleaner signal and less variability in impedance, but I don't understand the chemistry of it.

Basic electrodes are passive - they simply conduct the electricity. Active electrodes have a mini-amplifier built into them to amplify the signal almost immediately. This minimizes the impact of environmental noise.

# The components of an EEG system

## Cap and Electrodes



The Brain Products system we use separates caps from electrodes. But in some systems they are integrated into one piece.

## Brain Products Caps

An EEG cap is typically a nylon (or other stretchy material) cap designed to hold the electrodes in place (so you don't have to glue anything to the participant).

For BP systems, the cap has a series of cups designed to hold the electrodes. You must insert the electrodes into each cup prior to the experiment.

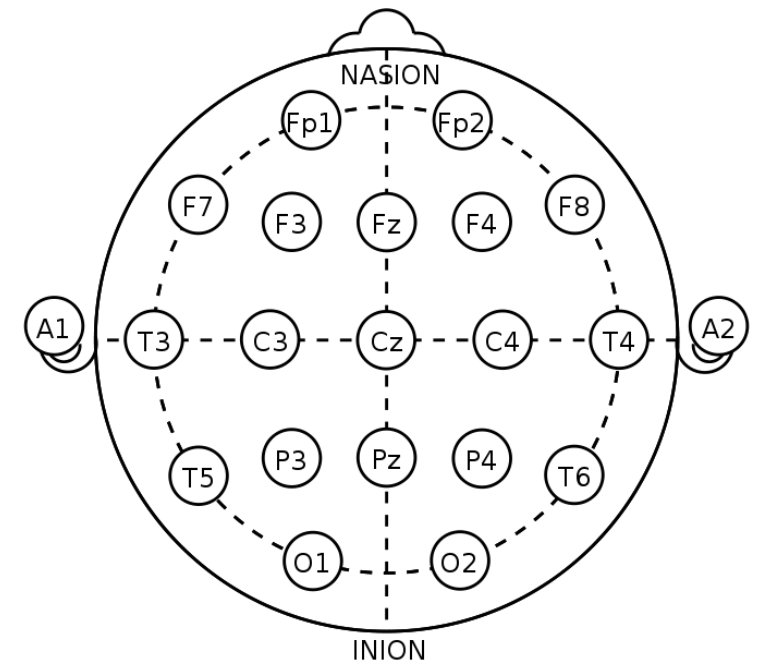
Caps come in different sizes because human heads come in different sizes. The sizing is done in centimeters (typically 52-64cm) in 2cm increments. BP caps have the size written on the tag. They also have distinct stitching colors for each size (though I have never memorized them).

# The 10-20 system of electrode placement

<b>Nasion:</b>	Indent between the eyes and above the nose.
<b>Inion:</b>	Prominent point on the occipital ridge of the skull.
<b>Preauricular point:</b>	Prominent point in front of the ear that can be felt when you open/close your jaw.
<b>Mastoid process:</b>	Prominent point at the base of the skull behind the ear.

In the 10-20 system, electrodes are placed around the skull based on measurements between these anatomical landmarks.

For example, along the midline from nasion to inion, the first electrode is placed 10% of the total distance (nasion to inion) from the nasion, the next is 20%, the next is 20%, and so on, until the last is 10% from inion.



# The 10-20 system naming convention

**Fp** is for electrodes placed on the forehead, and means “fronto-polar”.

**F** is for electrodes placed outside/near the frontal lobe.

**T** is for electrodes placed outside/near the temporal lobe.

**C** is for electrodes placed in the central portion of the skull.

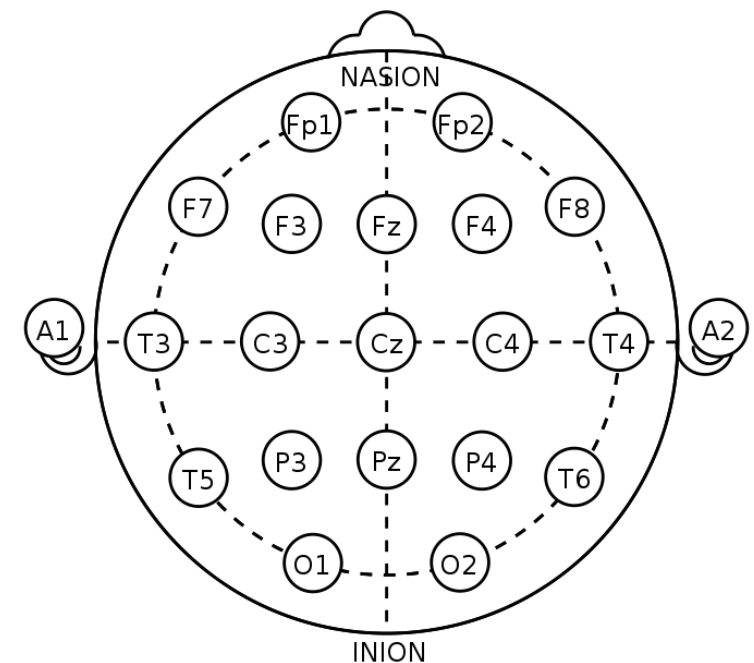
**P** is for electrodes placed outside/near the parietal lobe.

**O** is for electrodes placed outside/near the occipital lobe.

The left-right hemisphere placement is indicated by numbers: even numbers indicate right, odd numbers indicate left.

The magnitude of the number indicates the distance from the midline: larger is farther from the midline.

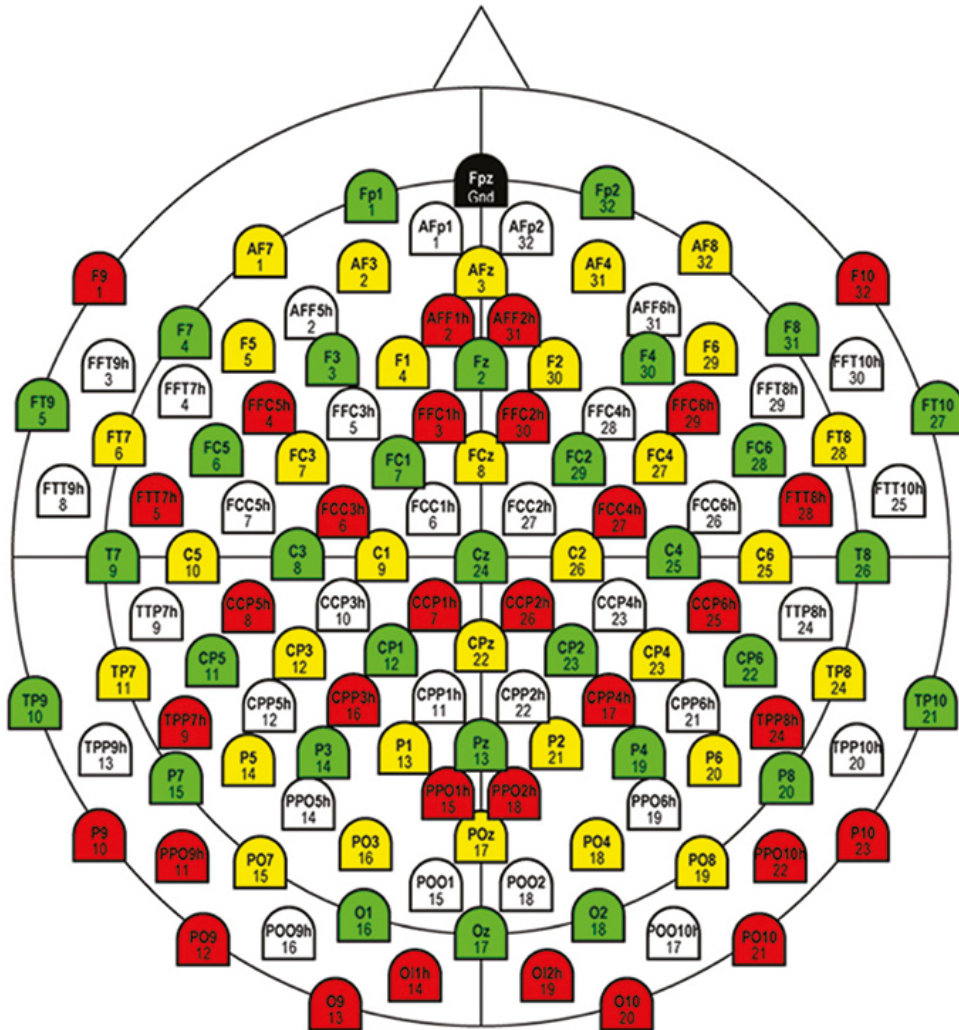
The midline is indicated by the letter **z** (zero) to avoid confusion between the numeral 0 and the letter O.



# The BP system uses numbers



128ch Standard Layout (previous version)



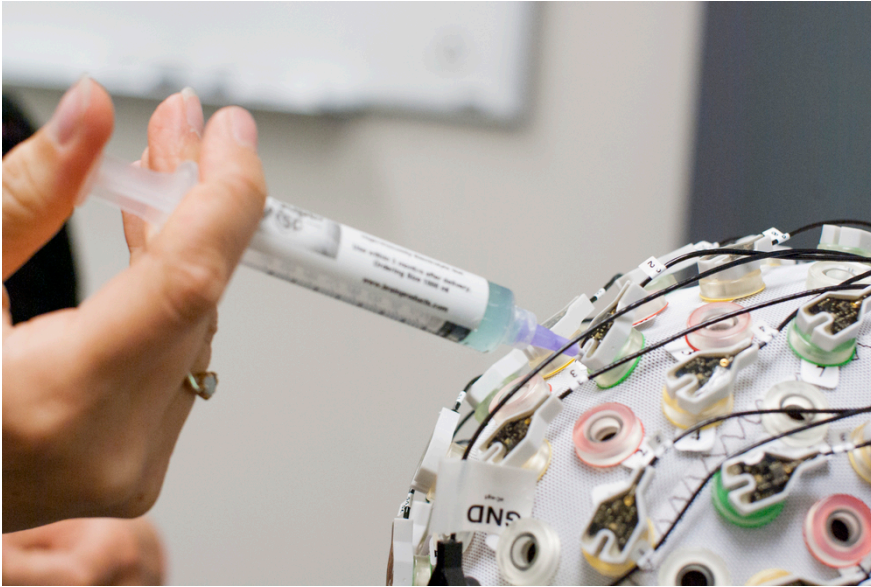
The Brain Products system we use marks each electrode cup on the cap with a number (1-32).

Our caps have 64 cups for up to 64 electrodes.

The green cups are for the first 32 electrodes (1-32). The yellow cups are for the next 32 (33-64). However, please note that the yellow cups are also labeled 1-32. This is because the electrodes are also labeled 1-32, and this allows every electrode set to be used for 32 channels or 64 channels.



# Gel and impedance



The last component of the system is the electroconductive gel. This gel acts like a wire to make a connection between the person's scalp and the electrode (which is sitting on top of their hair). Electroconductive gel is simply water, salt, and an emulsifier (like gelatin) to give it some structure.

The quality of the connection between the scalp and the electrode is typically evaluated by measuring the impedance between the scalp and the electrode. For our BP system, the goal is typically to get the impedances of all electrodes below 20K Ohms.

We will learn how to do this in the next segment of the course!





# Table of contents

1. Introduction: The big picture	Luck 1
2. Fundamentals	Luck 2
3. <a href="#">Hands-on training and best practices</a>	Luck 5
4. The ERP processing pipeline in EEGLAB + ERPLAB	Luck 6, 7, 8
1. Load the data.	
2. Filter the data (high-pass, possibly low-pass).	
3. ICA for artifact correction (optional).	
4. Re-reference the data.	
5. Add channel locations.	
6. Epoch the data.	
7. Artifact detection and rejection.	
8. Average the epochs to create subject ERPs.	
9. Average the subject ERPs to create a grand average ERP.	
10. Plotting: waveforms, topoplots, difference waves	
11. Measure amplitudes and latencies.	
12. Run statistical tests.	
5. Creating a script for EEGLAB + ERPLAB	
6. Creating a script for Fieldtrip	

# Table of contents

1. Introduction: The big picture
2. Fundamentals
3. Hands-on training and best practices
4. [The ERP processing pipeline in EEGLAB + ERPLAB](#)
  1. Load the data.
  2. Filter the data (high-pass, possibly low-pass).
  3. ICA for artifact correction (optional).
  4. Re-reference the data.
  5. Add channel locations.
  6. Epoch the data.
  7. Artifact detection and rejection.
  8. Average the epochs to create subject ERPs.
  9. Average the subject ERPs to create a grand average ERP.
  10. Plotting: waveforms, topoplots, difference waves
  11. Measure amplitudes and latencies.
  12. Run statistical tests.
5. Creating a script for EEGLAB + ERPLAB
6. Creating a script for Fieldtrip

Luck 1

Luck 2

Luck 5

Luck 6, 7, 8

# Table of contents

1. Introduction: The big picture
2. Fundamentals
3. Hands-on training and best practices
4. The ERP processing pipeline in EEGLAB + ERPLAB
  1. Load the data.
  2. Filter the data (high-pass, possibly low-pass).
  3. ICA for artifact correction (optional).
  4. Re-reference the data.
  5. Add channel locations.
  6. Epoch the data.
  7. Artifact detection and rejection.
  8. Average the epochs to create subject ERPs.
  9. Average the subject ERPs to create a grand average ERP.
  10. Plotting: waveforms, topoplots, difference waves
  11. Measure amplitudes and latencies.
  12. Run statistical tests.
5. Creating a script for EEGLAB + ERPLAB
6. Creating a script for Fieldtrip

Luck 1

Luck 2

Luck 5

Luck 6, 7, 8

# Loading the data

Before any analysis, whether it is ERP or time-frequency (or ICA, or anything else), the first step is to load the data set into your analysis software.

In this case, we are fundamentally using EEGLAB for data analysis. ERPLAB is a plugin that adds functionality to EEGLAB. So we need to load our data into EEGLAB.

Each EEG manufacturer records EEG data in its own format. This makes loading data into any analysis software complicated.

When it comes to EEGLAB, manufacturers (or user communities) will typically write a plugin that allows you to import their data. Here is EEGLAB's page about this: [https://sccn.ucsd.edu/wiki/A01:\\_Importing\\_Continuous\\_and\\_Epoched\\_Data](https://sccn.ucsd.edu/wiki/A01:_Importing_Continuous_and_Epoched_Data)

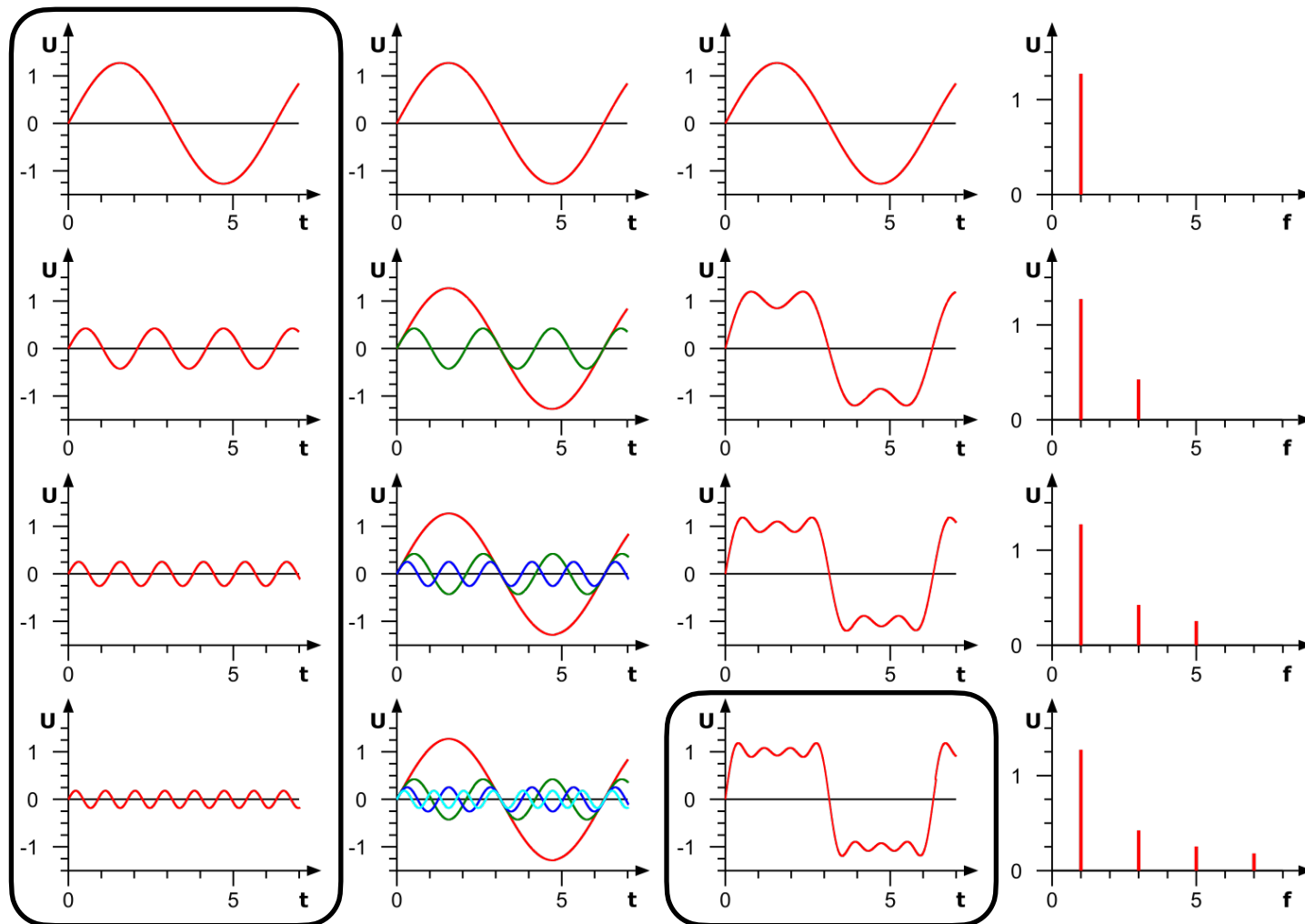
I have posted the plugin for our data on the course website: <http://sprouse.uconn.edu/courses/eeg-methods/bvaio1.57.tar.gz>

# Table of contents

1. Introduction: The big picture	Luck 1
2. Fundamentals	Luck 2
3. Hands-on training and best practices	Luck 5
4. The ERP processing pipeline in EEGLAB + ERPLAB	Luck 6, 7, 8
1. Load the data.	
2. Filter the data (high-pass, possibly low-pass).	
3. ICA for artifact correction (optional).	
4. Re-reference the data.	
5. Add channel locations.	
6. Epoch the data.	
7. Artifact detection and rejection.	
8. Average the epochs to create subject ERPs.	
9. Average the subject ERPs to create a grand average ERP.	
10. Plotting: waveforms, topoplots, difference waves	
11. Measure amplitudes and latencies.	
12. Run statistical tests.	
5. Creating a script for EEGLAB + ERPLAB	
6. Creating a script for Fieldtrip	

# Filtering: Fourier Analysis

**The big idea:** Any complex signal, including an EEG signal or the square wave below, can be represented as the sum of some number of pure sine waves (each with their own amplitude and frequency).

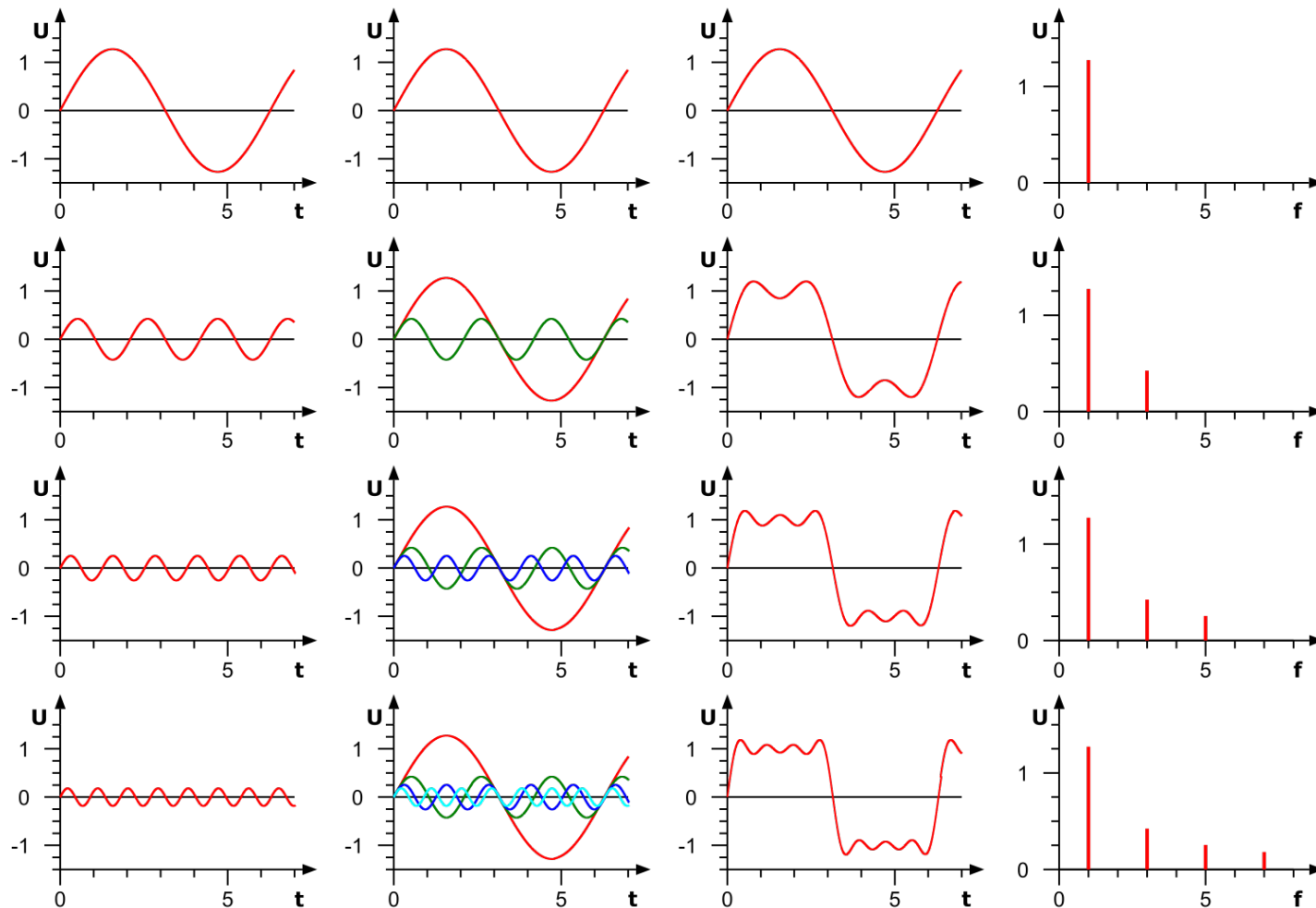


# Filtering: Fourier Analysis

The first column shows the four constituent waves.

The second column shows the progressive overlay of the four waves.

The third column shows the summation of the overlaid waves.

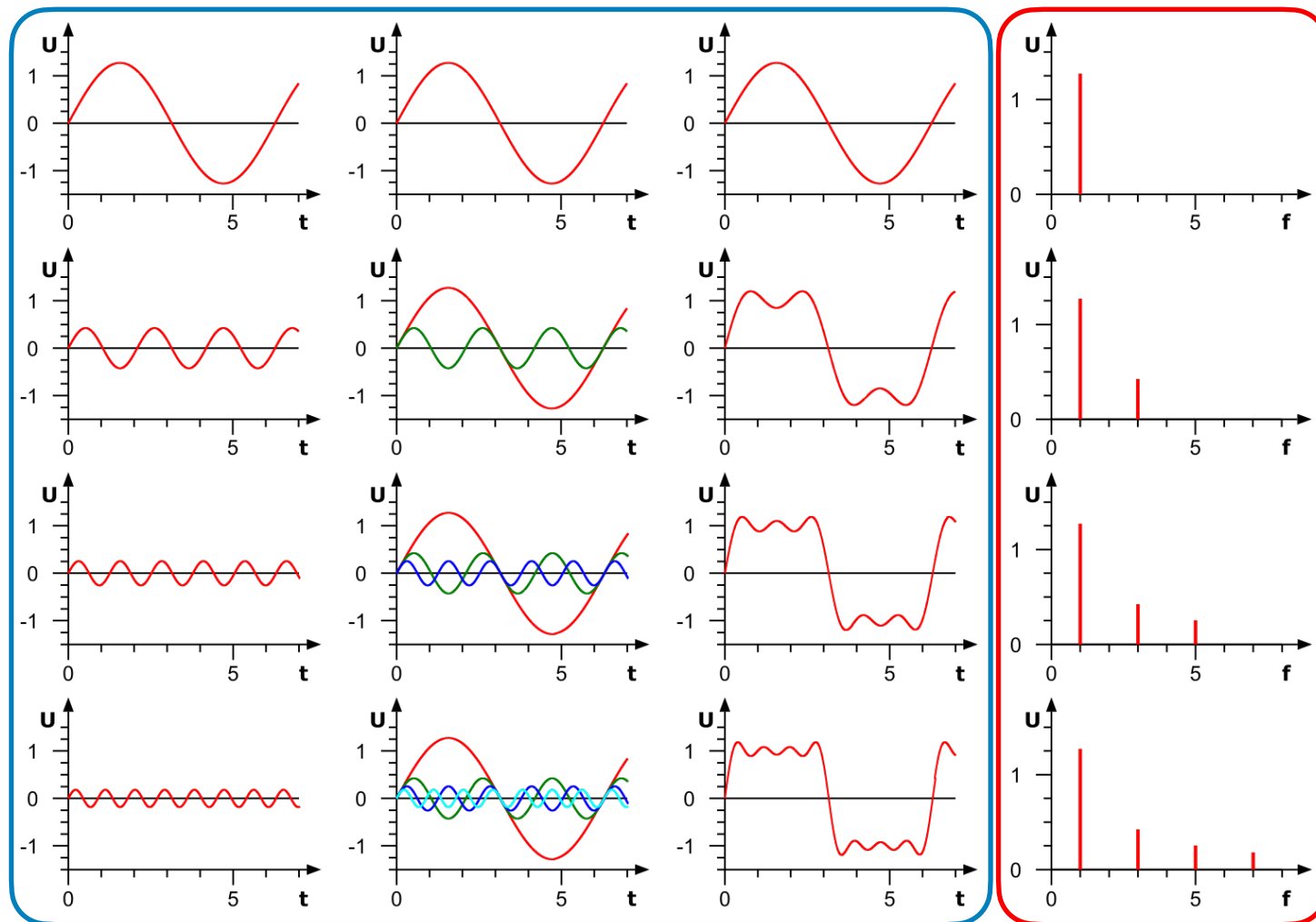




# Filtering: Time domain and Frequency domain

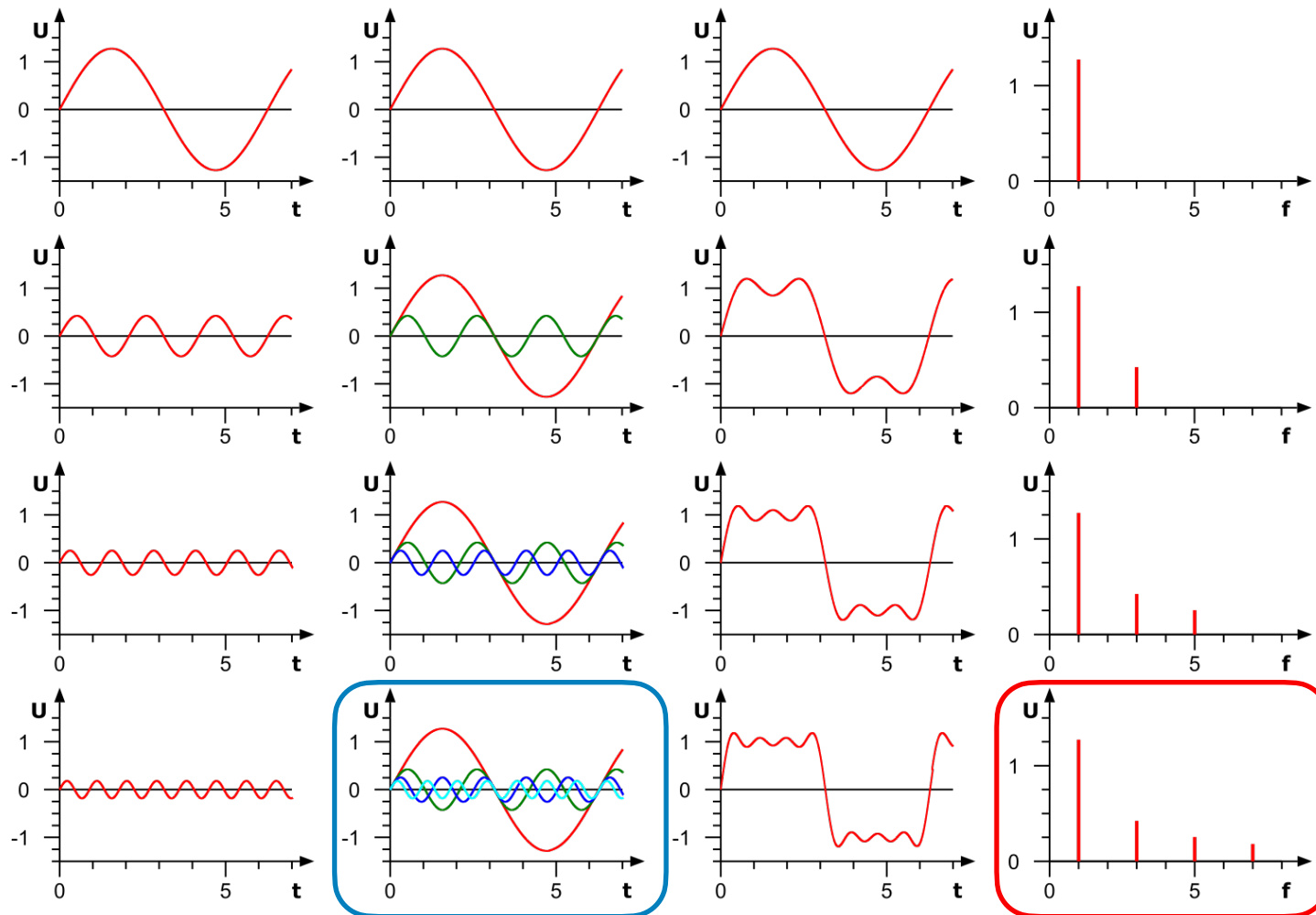
The first three columns show time-domain representations: the x-axis is time, and the y-axis is amplitude.

The fourth column shows a frequency-domain representation: the x-axis is frequency, and the y-axis is amplitude.



# Filtering: Time domain and Frequency domain

The time-domain and frequency-domain are equivalent. This means that a complex wave can be represented either as a **set of waves in the time domain**, or as a **set of frequency amplitudes in the frequency-domain**. Both are equivalent, so you can use whichever is easiest to work with.



# Filtering: Interactive Fourier demo

Sine waves are really representations of circular paths. So if we want to be super precise, we would say that the underlying idea of Fourier Analysis is that any complex signal can be represented as the sum of some number of circular paths.

This interactive demo reminds us that sine waves are circular paths:

<https://jackschaedler.github.io/circles-sines-signals/sincos.html>

This interactively demonstrates Fourier Analysis, with both time-domain and frequency-domain representations:

[https://jackschaedler.github.io/circles-sines-signals/dft\\_introduction.html](https://jackschaedler.github.io/circles-sines-signals/dft_introduction.html)

And this is a youtube video demonstrating Fourier Analysis from a slightly different perspective. I don't know if this is better, but it is neat:

<https://www.youtube.com/watch?v=spUNpyF58BY>

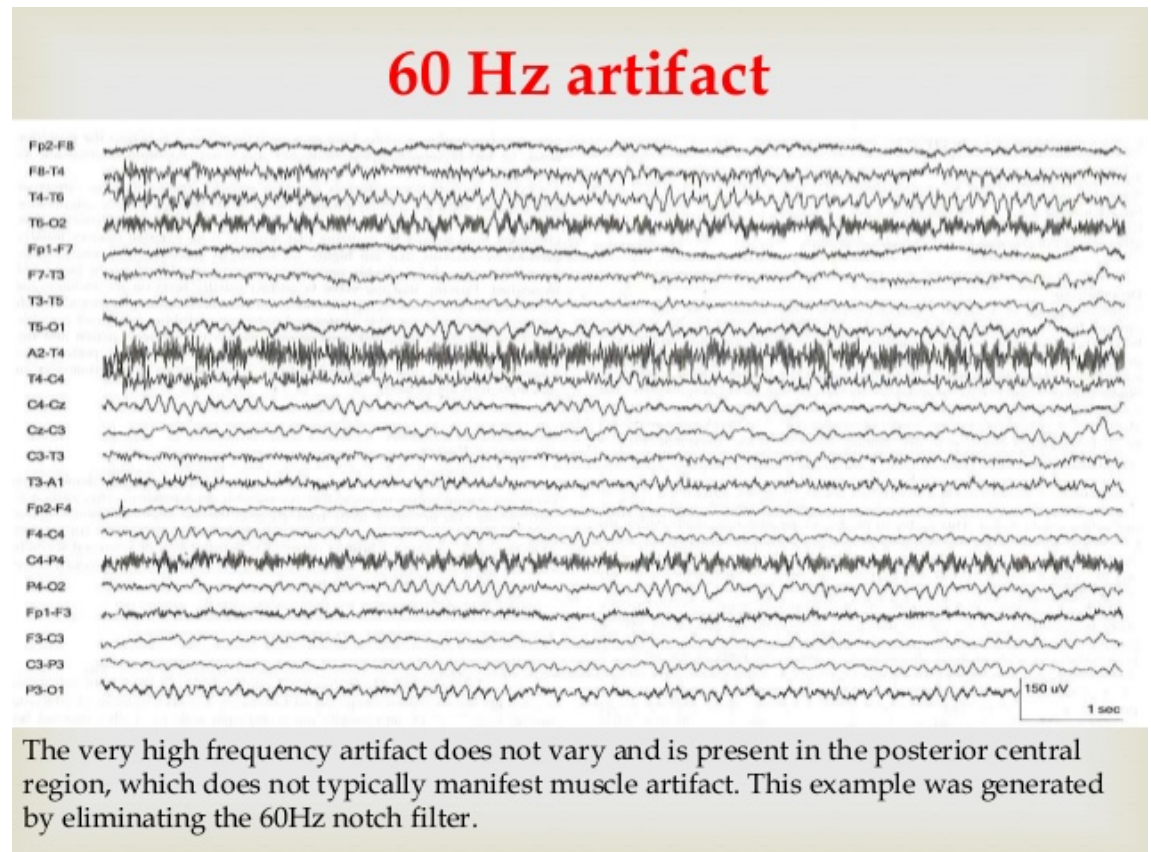
# Filtering in the frequency-domain

Filtering means to remove frequencies (that we do not want) from the signal.

Why might we want to eliminate some frequencies? Because some frequencies are very clearly not cognitive activity.

The electrical system in the US is AC, with a frequency of 60Hz. The magnetic fields created by electrical wiring in buildings can induce 60Hz AC in EEG electrodes.

We will look at artifacts in more detail in the next section.

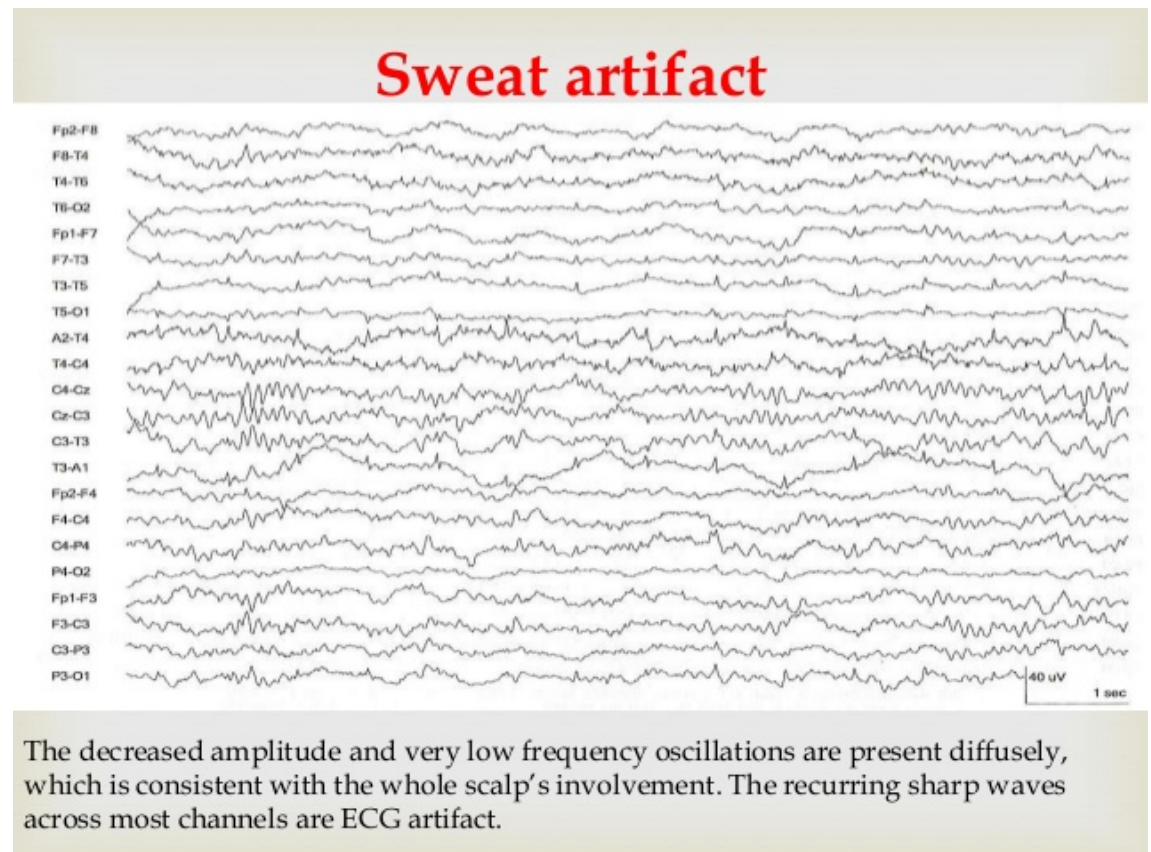


# Filtering in the frequency-domain

Filtering means to remove frequencies (that we do not want) from the signal.

Why might we want to eliminate some frequencies? Because some frequencies are very clearly not cognitive activity.

Sweat creates relatively slow waves (reminiscent of deep sleep).



We will look at artifacts in more detail in the next section.



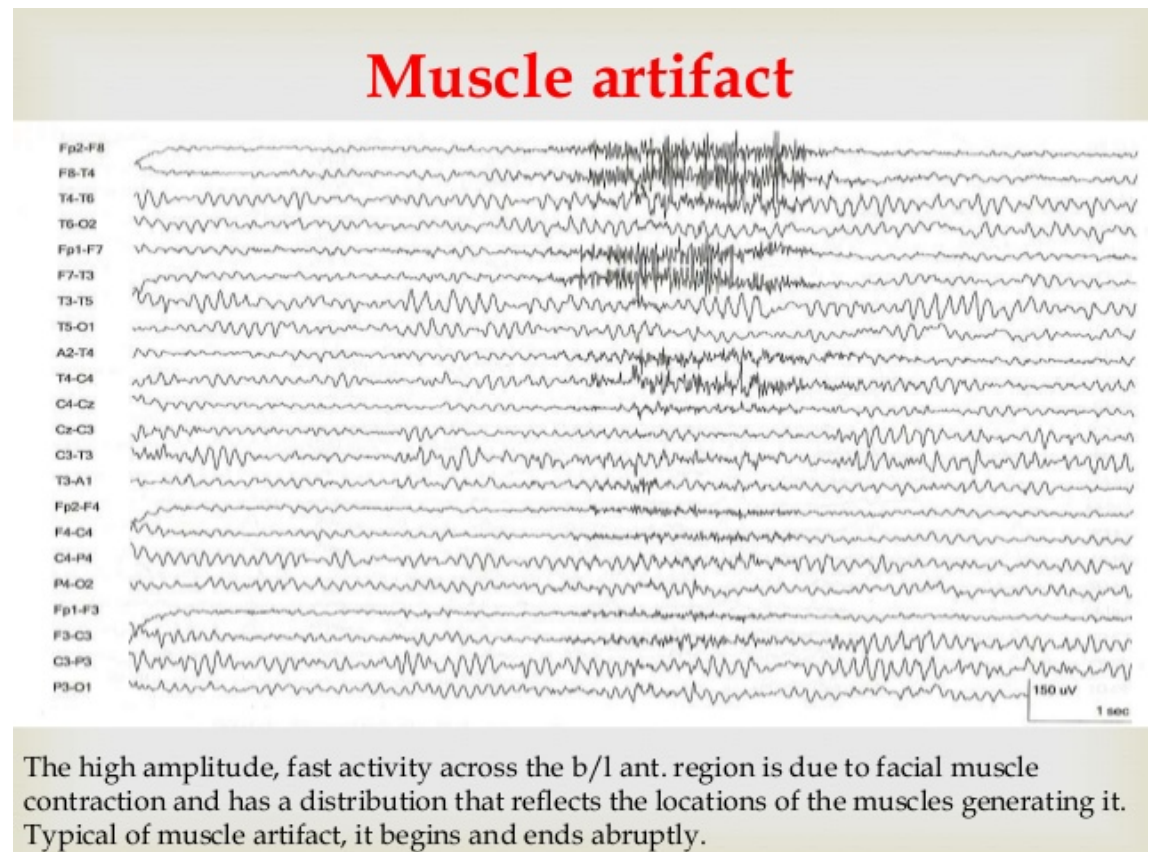
# Filtering in the frequency-domain

Filtering means to remove frequencies (that we do not want) from the signal.

Why might we want to eliminate some frequencies? Because some frequencies are very clearly not cognitive activity.

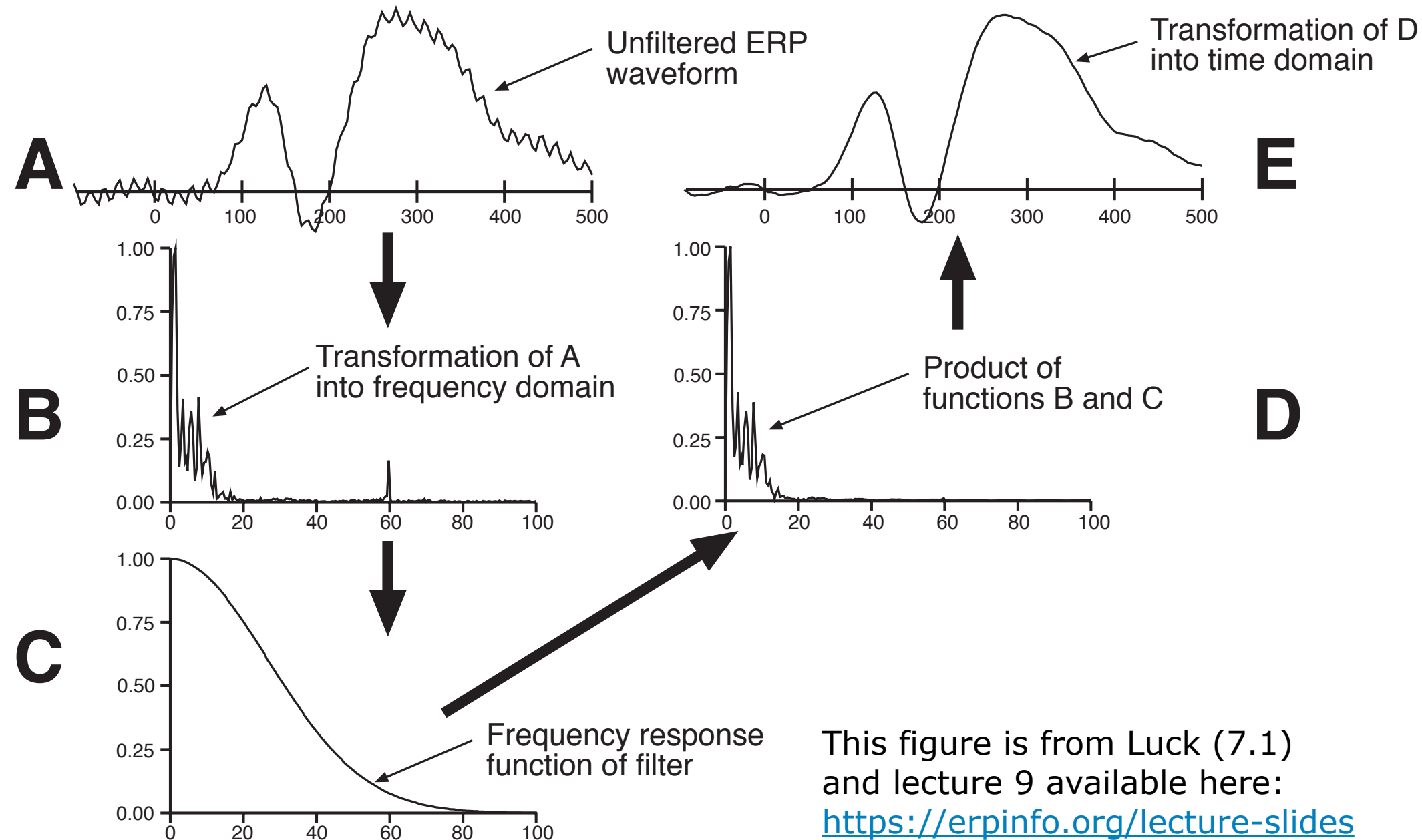
Muscles contractions involve high frequency electrical activity.

We will look at artifacts in more detail in the next section.



<https://www.slideshare.net/SudhakarMarella/eeg-artifacts-15175461>

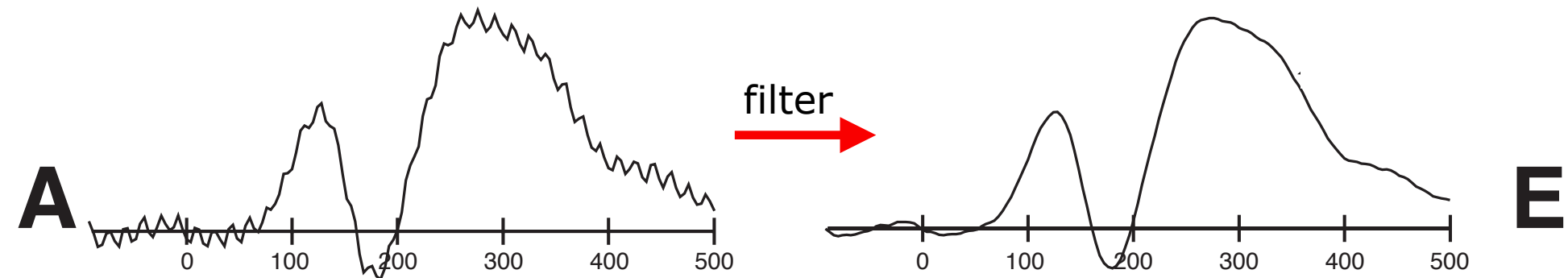
# Filtering in the frequency-domain



This figure is from Luck (7.1) and lecture 9 available here: <https://erpinfo.org/lecture-slides>

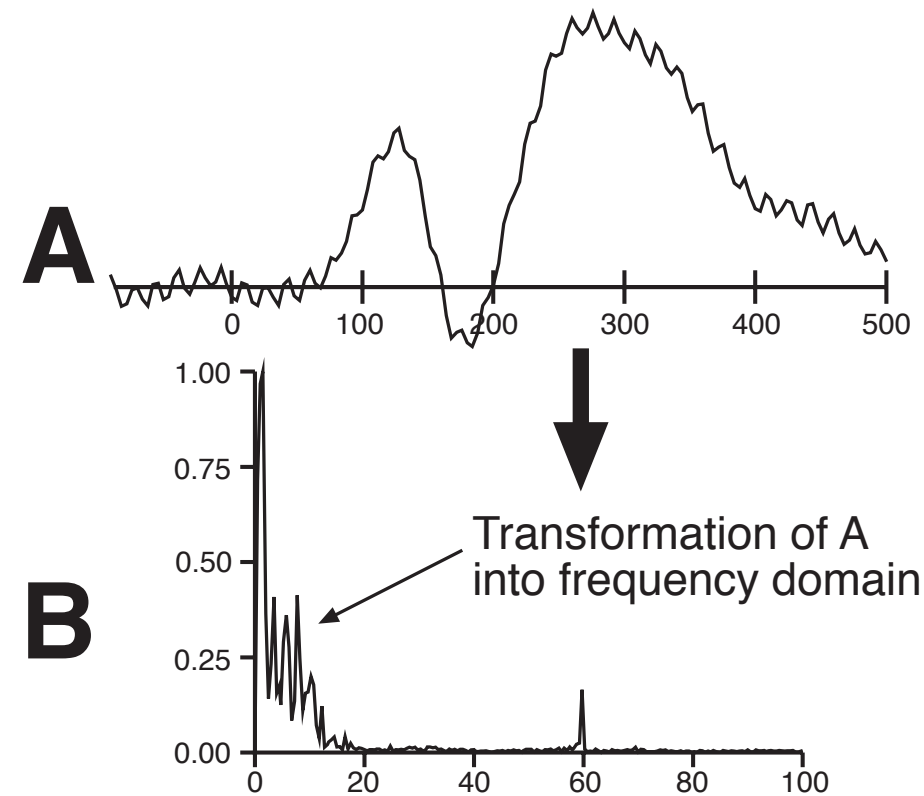


# Filtering in the frequency-domain



This is the transformation we are trying to achieve with our filter. Our filter will attenuate higher frequencies (and pass lower frequencies through). In practice, this will remove 60Hz electrical noise and some muscle artifacts.

# Filtering in the frequency-domain



The first step is to convert the EEG signal into a frequency-domain representation.

While it is possible to do filtering in the time-domain, I find it easier to think about it as a frequency-domain operation. This is partly because filtering is about frequency elimination, and partly because the math is simpler.

The algorithm that allows us to convert a time-domain representation into a frequency-domain representation is called the **Fourier Transform**. We are not going to learn the details of the Fourier Transform right now. It will figure more prominently later in the course when we do time-frequency analysis. So for now, just think of it like a magic black box that converts time-domain representations to frequency-domain representations, and file away a promissory note that we still need to learn the details of the Fourier Transform.

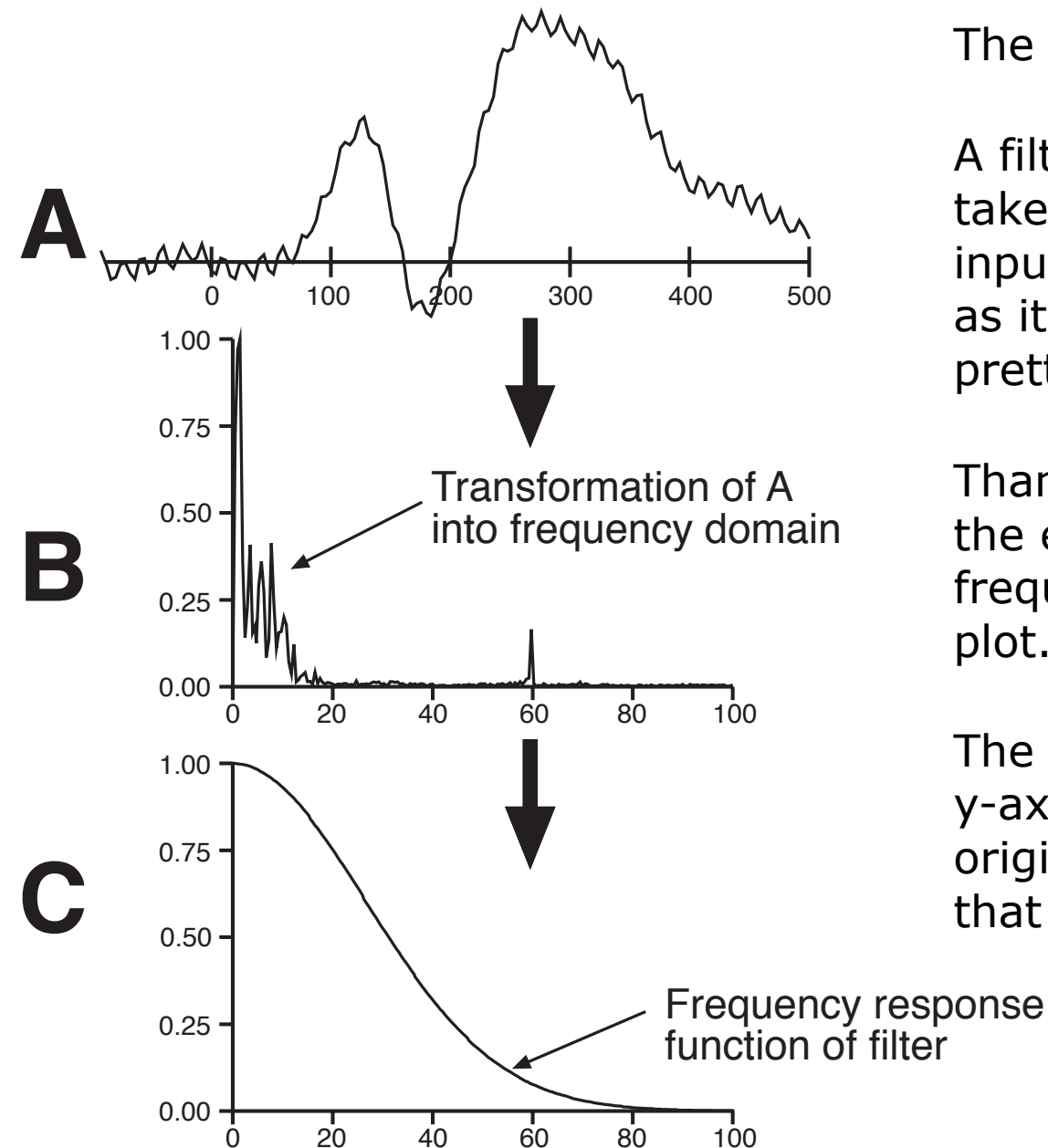
# Filtering in the frequency-domain

The next step is to design your filter.

A filter is essentially a function that takes the original EEG signal as its input, and yields the filtered EEG signal as its output. Filter functions can be pretty complicated looking.

Thankfully, we can visually represent the effect of the function in the frequency domain with a fairly simple plot.

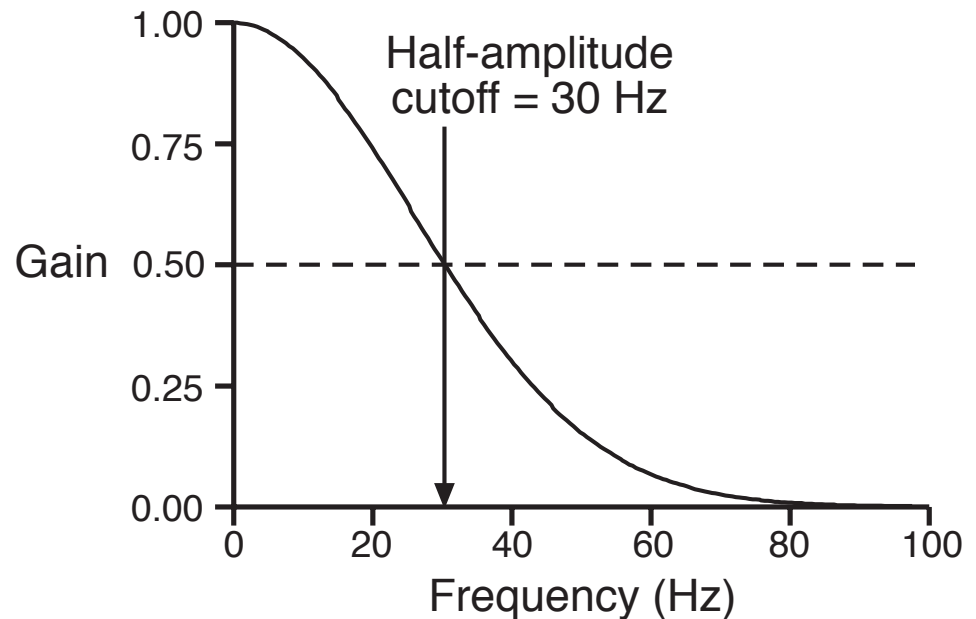
The x-axis of this plot is frequency. The y-axis is gain: the proportion of the original amplitude of that frequency that the filter will yield in the output.



# A closer look at the frequency response function

The basic idea of the FRF is that it tells you what proportion of a frequency's amplitude will survive the filter.

A gain of 1 means that all of the frequency's amplitude will survive the filter. A gain of 0 means that none of the amplitude will survive the filter (the frequency will be eliminated entirely). A gain of .5 means that 50% of the amplitude will survive the filter.



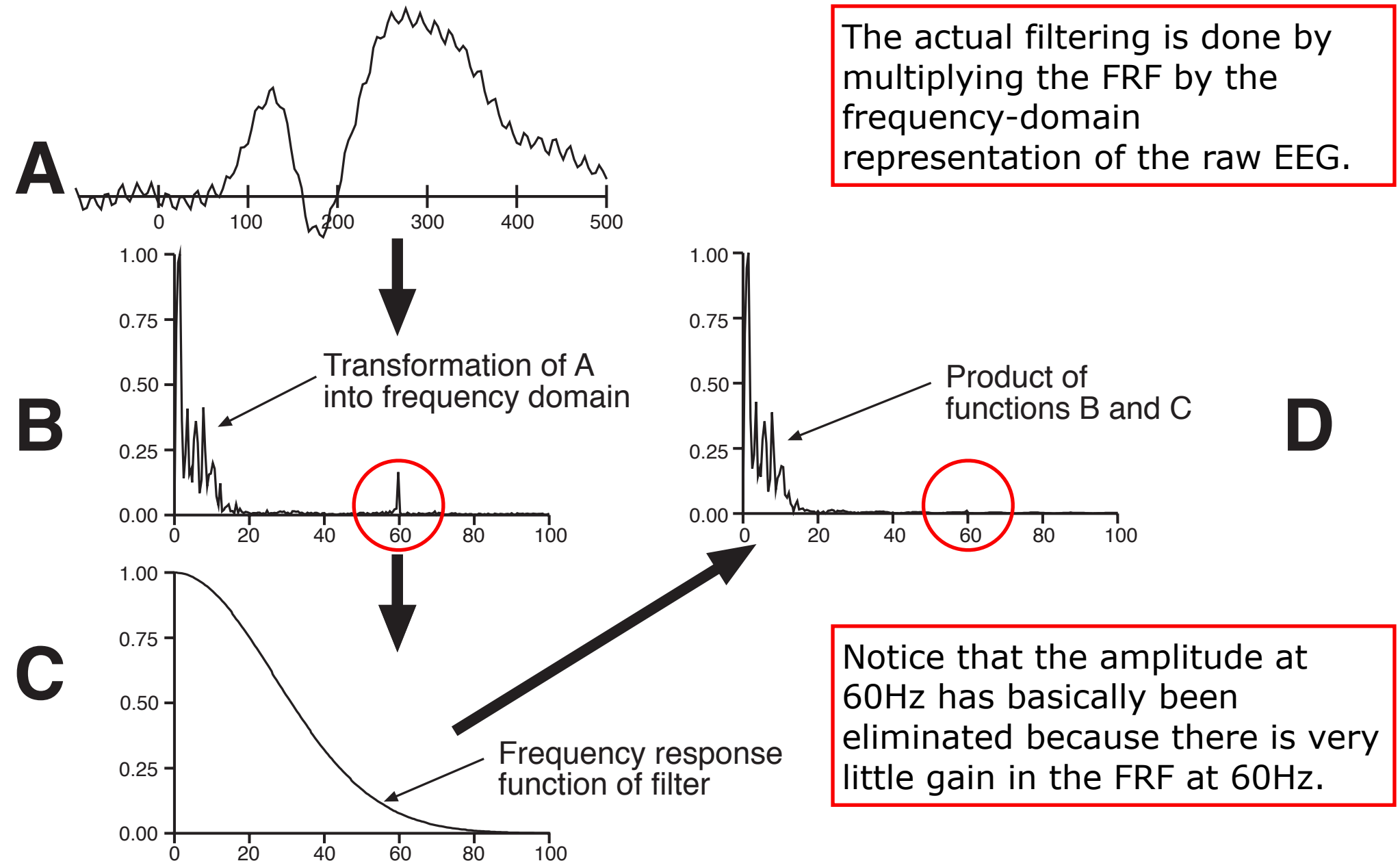
From figure 5.1 in Luck 2005,  
and lecture 9 of the slides

FRFs are typically described with two numbers: (i) the half-amplitude cutoff, which tells you the frequency that will have a gain of .5, and (ii) the roll-off, which tells you the slope of the curve.

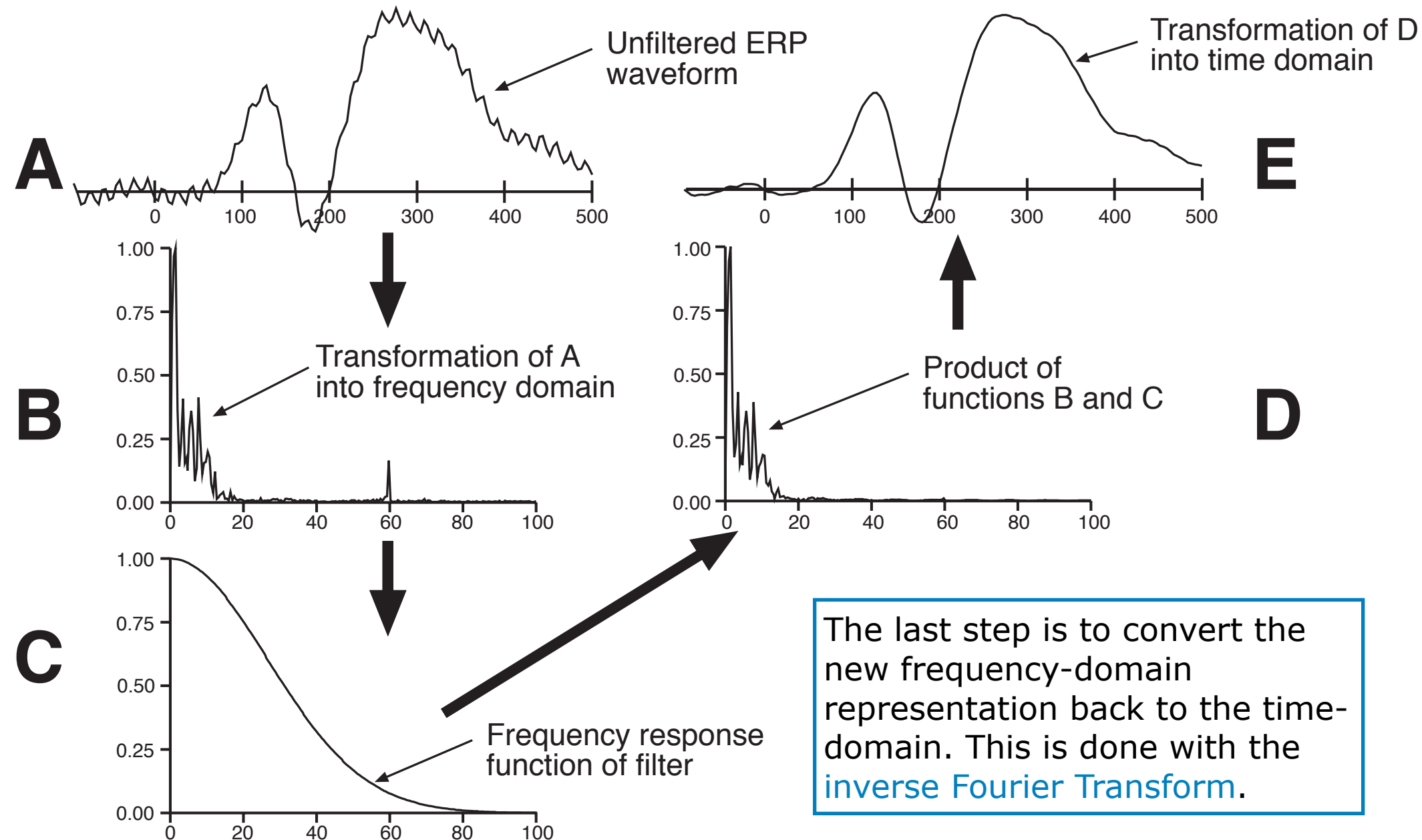
The roll-off is a slope (rise over run). The units are decibels (dB) over octaves. We will look at dBs in more detail later. Octave means doubling of frequency.

# Filtering in the frequency-domain

The actual filtering is done by multiplying the FRF by the frequency-domain representation of the raw EEG.

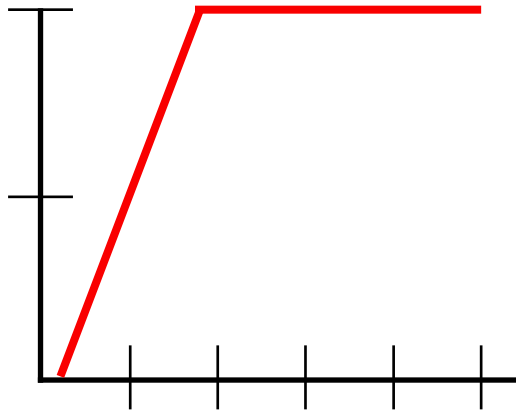


# Filtering in the frequency-domain

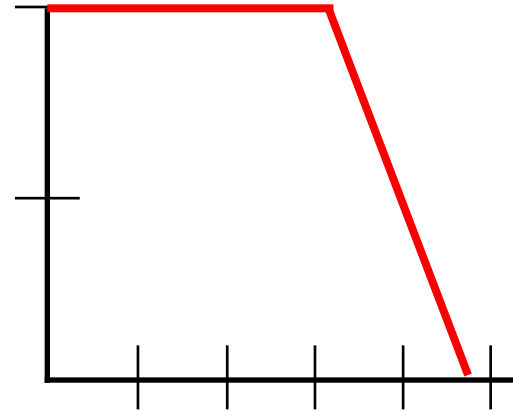


# General types of filters

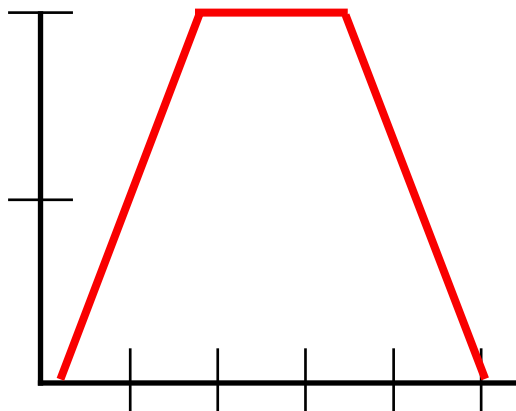
**high-pass**



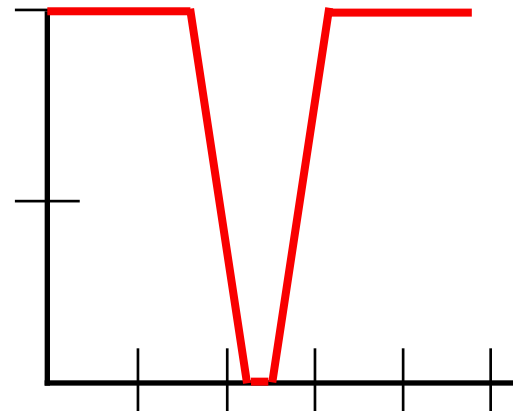
**low-pass**



**band-pass**



**notch**



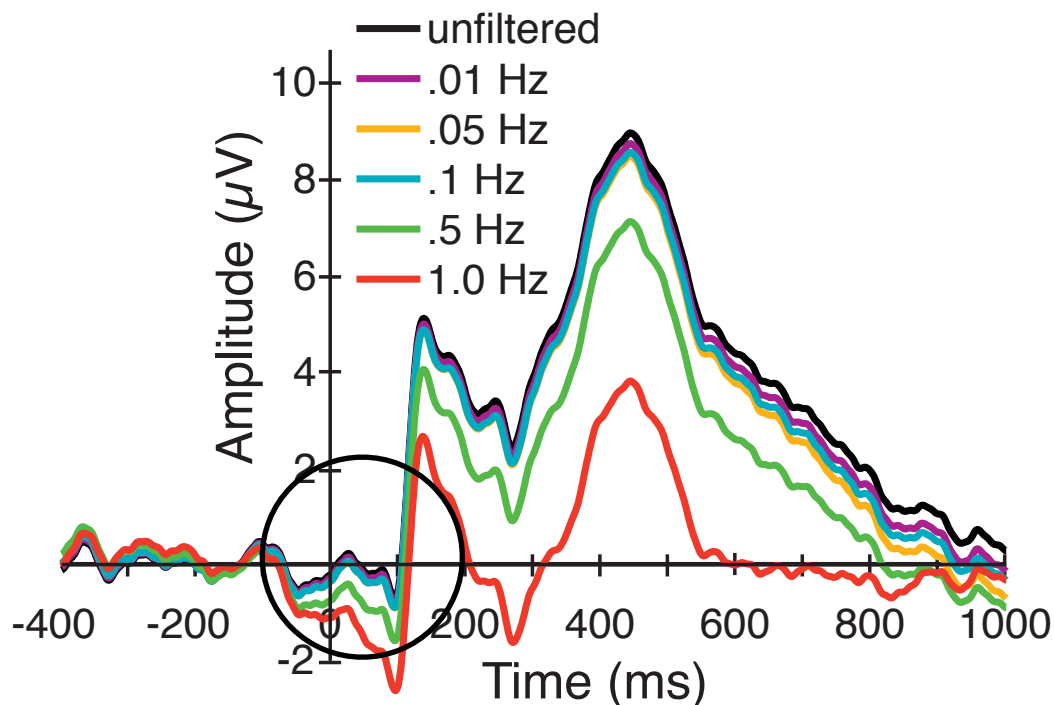
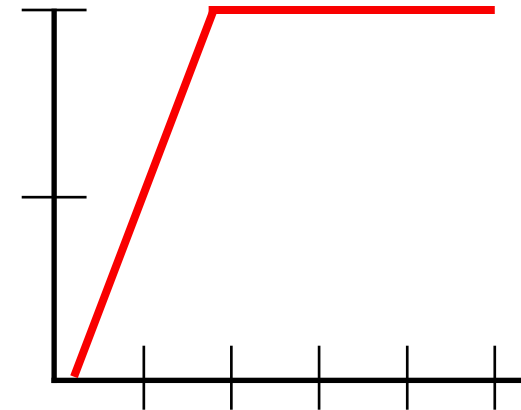
The exact shapes are dictated by the half-amplitude cut-off and roll-off parameters that you set. These are just basic schematics.

# High-pass filtering - Luck's recommendation

The goal of high-pass filtering is to remove the slow-drifts that can happen during EEG recording (due to underlying biological factors).

Luck recommends a high-pass filter with a half-amplitude **cutoff  $\leq .1$  Hz**

**high-pass**



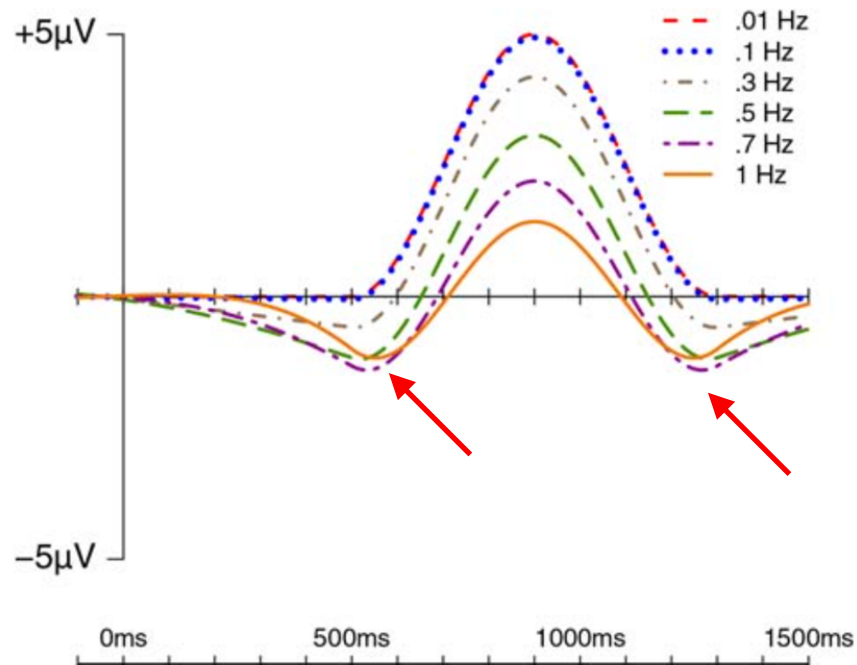
Cutoffs that are  $\leq .1$  Hz don't change the ERP much.

Cutoffs that are  $> .1$  Hz suppress slow-wave amplitude, and induce artificial negative-going deflections.

This image is from Luck's lecture 9:  
<https://erpinfo.org/lecture-slides>



# High-pass filtering - Luck's recommendation



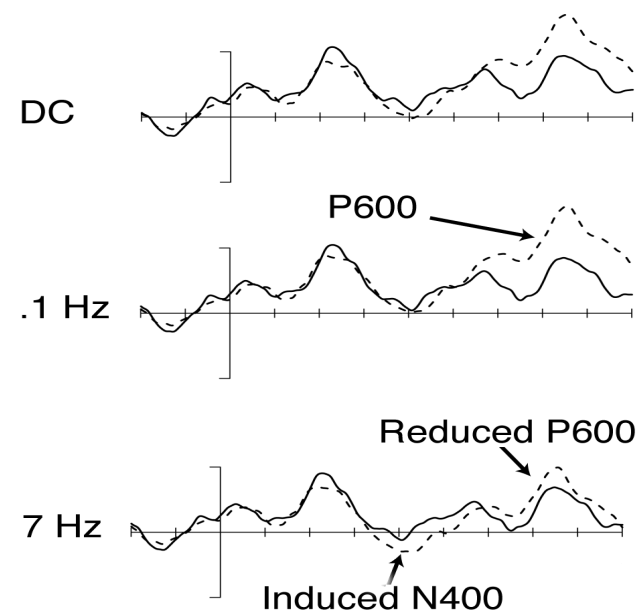
This is a simulated P600 from Tanner, Morgan-Short, and Luck 2015 (taken from Luck's lecture 9).

High cutoffs lead to a suppressed P600 effect and artifactual negativities.

There is basically no distortion at .1 Hz

The same pattern holds for real data (a syntactic violation that triggers a P600). This is from the same study and lecture.

High cutoff values reduce the P600 and induce a negativity.



# How to high-pass filter in ERPLAB

High-pass filtering should be done on the continuous EEG data.

The reason for this is not obvious from the frequency-domain representation of filtering, but it will become clearer when we look at the time-domain representation of filtering. I want to postpone that until later in the course when we build the math necessary to look at filtering in the time-domain.

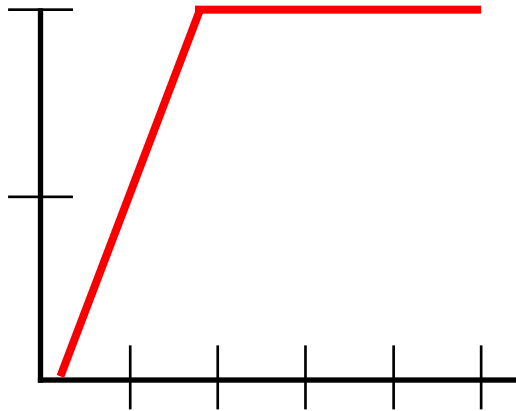
Actually filtering in ERPLAB is surprisingly easy. The GUI provides you with a number of filter types to choose from, and allows you to set the cutoff and roll-off parameters. It also shows you a plot of the frequency response function so that you can really see what your filter will do.

<https://github.com/lucklab/erplab/wiki/Filtering>

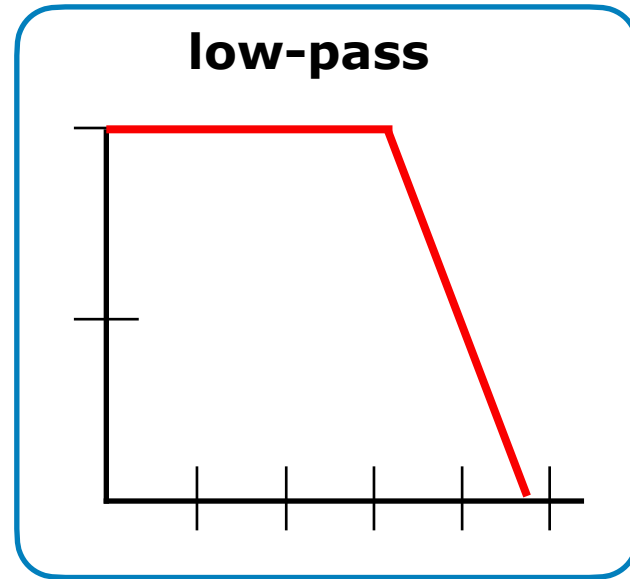
Luck recommends a high-pass filter with a half-amplitude cutoff of .1 Hz and a roll-off of 12db/octave. I think that will probably be a good choice for most, if not all, of the ERP experiments that are likely to run in your own research.

# Low-pass filtering

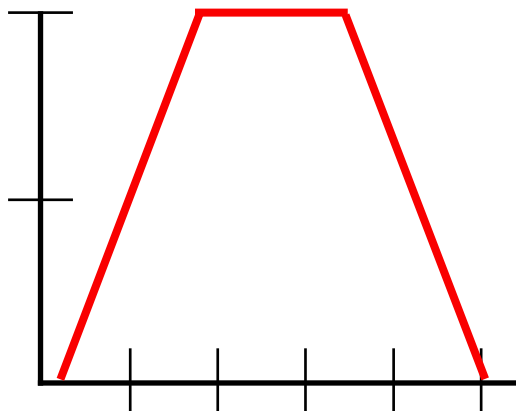
**high-pass**



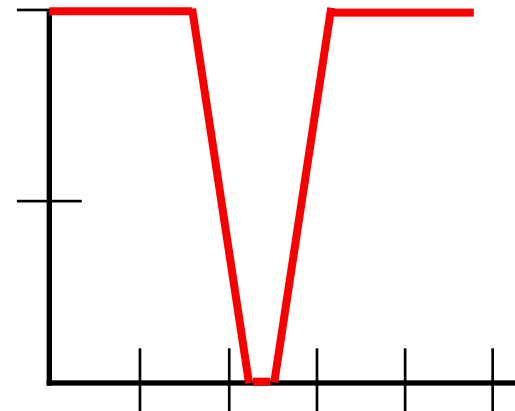
**low-pass**



**band-pass**

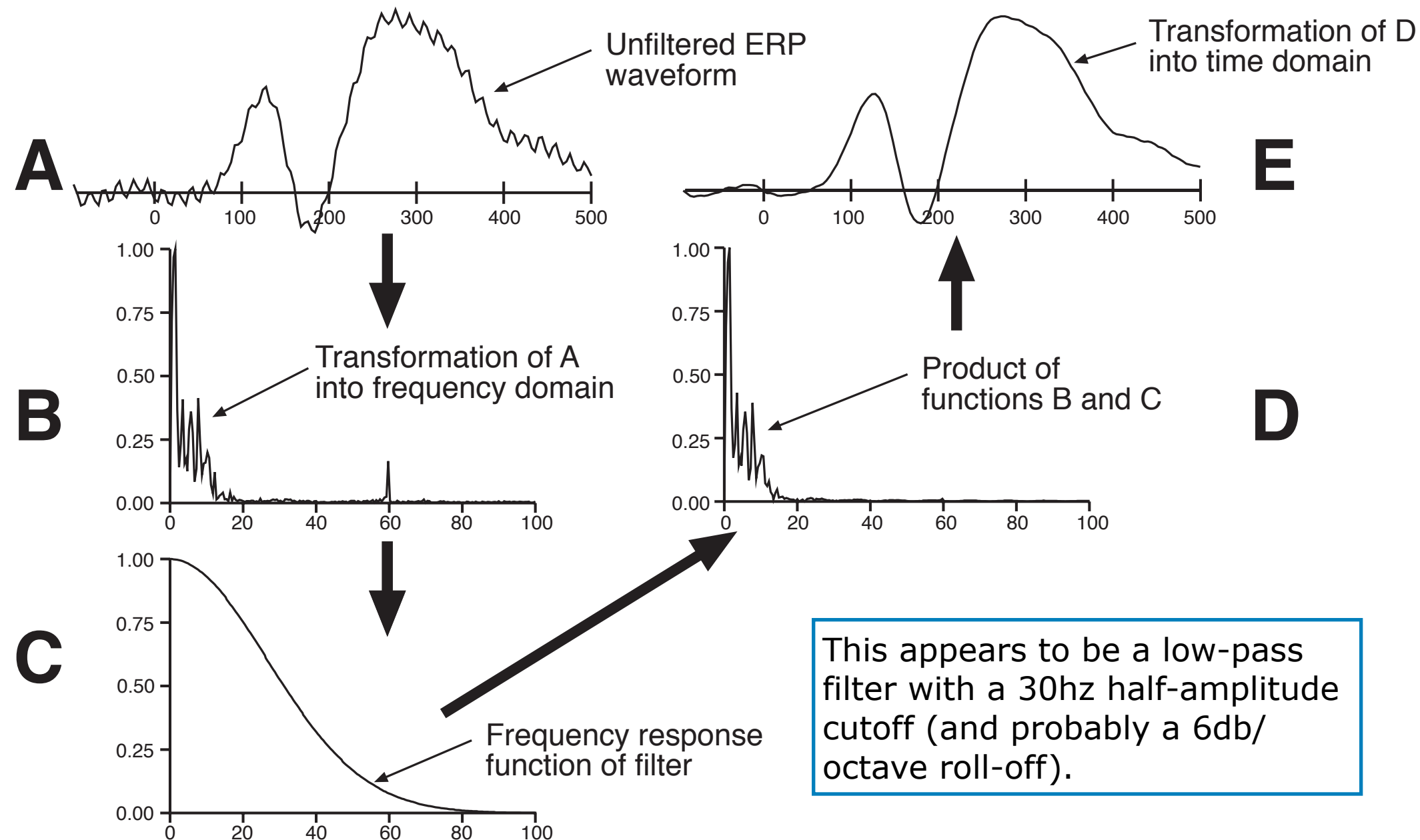


**notch**



The exact shapes are dictated by the half-amplitude cut-off and roll-off parameters that you set. These are just basic schematics.

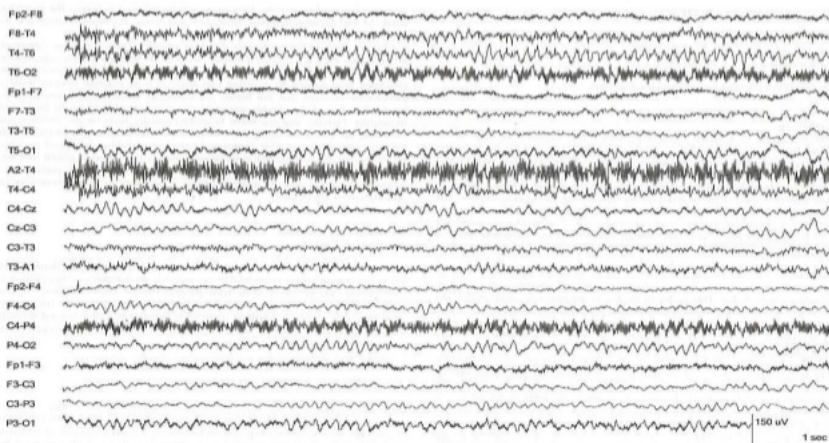
# We have already seen a low-pass filter



# Why low-pass filter?

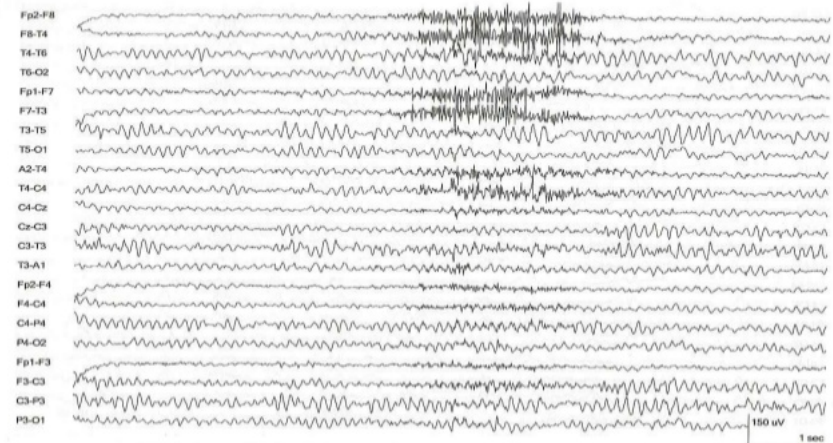
Low-pass filtering is done to **remove sources of high-frequency noise**, such as 60hz line noise or muscle artifacts.

## 60 Hz artifact



The very high frequency artifact does not vary and is present in the posterior central region, which does not typically manifest muscle artifact. This example was generated by eliminating the 60Hz notch filter.

## Muscle artifact



The high amplitude, fast activity across the b/l ant. region is due to facial muscle contraction and has a distribution that reflects the locations of the muscles generating it. Typical of muscle artifact, it begins and ends abruptly.

This raises an important question: Does low-pass filtering make artifact detection easier or harder? The answer will vary from data set to data set. Luck's example is that 60Hz line noise can make blinks difficult to detect with a moving window peak-to-peak algorithm. So, sometimes you may want to filter before artifact detection, sometimes you may want to filter after. (I typically filter before... I find it easier to identify artifacts in filtered data.)

# When to low-pass filter?

Luck has placed the low-pass filtering step after creating subject ERPs. But, in fact, low-pass filtering can happen at pretty much any point in the processing pipeline: continuous EEG, epochs, subject ERPs, or grand average ERPs. I like to do it before artifact detection.

One reason for this flexibility is that low-pass filters don't need long time segments to behave correctly, so they can be applied to epochs or ERPs. This will become clearer when we look at filters in the time-domain, so for now, just remember that as a fact.

Another reason for this flexibility is that most of the filters used in ERP research are linear transformations (or approximate linear transformations).

Linear transformations can be applied in any order relative to other linear transformations, and the end result will be equal. Re-referencing, baseline correction, and averaging are all linear transformations, so filtering can be done pretty much anywhere in the pipeline.

Here is a nice visual introduction to linear transformations as part of an awesome series on linear algebra: [https://www.youtube.com/watch?v=kYB8IZa5AuE&t=0s&list=PLZHQObOWTQDPD3MizzM2xVFitgF8hE\\_ab&index=4](https://www.youtube.com/watch?v=kYB8IZa5AuE&t=0s&list=PLZHQObOWTQDPD3MizzM2xVFitgF8hE_ab&index=4)

# How to low-pass filter

As we've already seen, ERPLAB makes it very easy to filter EEG data:

<https://github.com/lucklab/erplab/wiki/Filtering>

Here are some practical questions:

1. What cutoff and rolloff should you use?

30Hz is a fairly typical cutoff. Luck recommends a rolloff of 12db/octave.

2. Should you filter before or after measuring amplitudes and onsets?

Luck recommends filtering before measuring ERP onsets, but cautions that you often need even lower cutoffs (10Hz), and that you should be an expert before doing that.

Luck points out that filtering is typically unnecessary if you are measuring mean amplitude in a wide window (e.g, 300-500ms). The wide window will already attenuate high frequency noise.

# Promissory notes from this section

1. The Fourier Transform and the inverse Fourier Transform
2. Time-domain filtering
3. Types of filters (IIR, FIR)
4. An explanation for why high-pass filtering requires more time (time/frequency trade-offs and edge artifacts)



# Table of contents

1. Introduction: The big picture
2. Fundamentals
3. Hands-on training and best practices
4. **The ERP processing pipeline in EEGLAB + ERPLAB**
  1. Load the data.
  2. Filter the data (high-pass, possibly low-pass).
  3. **ICA for artifact correction (optional).**
  4. Re-reference the data.
  5. Add channel locations.
  6. Epoch the data.
  7. Artifact detection and rejection.
  8. Average the epochs to create subject ERPs.
  9. Average the subject ERPs to create a grand average ERP.
  10. Plotting: waveforms, topoplots, difference waves
  11. Measure amplitudes and latencies.
  12. Run statistical tests.
5. Creating a script for EEGLAB + ERPLAB
6. Creating a script for Fieldtrip

Luck 1

Luck 2

Luck 5

Luck 6, 7, 8

# Artifact correction = ICA

ICA is a supremely advanced topic in terms of both the underlying math, and the skills required to use it “safely”. So, for pedagogical purposes, I don’t want to cover it right now. There are number of other skills you need to build first (artifact identification, artifact rejection, and interpreting wave-forms and scalp maps).

Luck (and others) suggest doing ICA early in your pipeline because it should be run on the continuous EEG (or very large epochs). So I mention it here so that you know this is where it would go in your pipeline.

I also want to mention that ICA might not be right for your experiment, so you shouldn’t feel like you have to learn it. For example, if you are running a vision-based experiment, blinks are not just artifacts — they also tell you when the participant was not looking at the experiment. So you may want to throw out those trials, not clean them.

# Table of contents

1. Introduction: The big picture
2. Fundamentals
3. Hands-on training and best practices
4. **The ERP processing pipeline in EEGLAB + ERPLAB**
  1. Load the data.
  2. Filter the data (high-pass, possibly low-pass).
  3. ICA for artifact correction (optional).
  4. **Re-reference the data.**
  5. Add channel locations.
  6. Epoch the data.
  7. Artifact detection and rejection.
  8. Average the epochs to create subject ERPs.
  9. Average the subject ERPs to create a grand average ERP.
  10. Plotting: waveforms, topoplots, difference waves
  11. Measure amplitudes and latencies.
  12. Run statistical tests.
5. Creating a script for EEGLAB + ERPLAB
6. Creating a script for Fieldtrip

Luck 1

Luck 2

Luck 5

Luck 6, 7, 8

# Active, ground, and reference

Voltage is electrical potential. Potential always requires two locations. For us, that means two electrodes. Let's call them **active** and **ground**.

We call the first **active** because it is the electrode near the cortical activity that you want to measure.

We call the other **ground** because the physical measurement of electrical potential must be relative to a ground circuit.

This means that the voltage at our active electrode should be given by this equation: **active - ground**

The problem is that the ground circuit in an EEG amplifier has a lot of electrical noise in it. That noise is much larger than the neural activity that we are trying to measure. So the equation above would just yield noise.

One way around this problem is to calculate the voltage at a second electrode site. We will call it **reference** because, as we will see in a minute, it will become the new reference for calculating the voltage for **active**.

Here is the equation for calculating the voltage at **reference**: **reference - ground**

# Eliminating the ground (and noise)

The trick in EEG amplifiers is that instead of simply amplifying the voltage at one electrode, they actually amplify the voltage at two electrodes, and calculate the difference between them:

$$(\text{active} - \text{ground}) - (\text{reference} - \text{ground})$$

The result of this differential amplification is a signal that has no ground in it (the two ground terms cancel out):

$$(\text{active} - \text{reference})$$

This means that all of the noise in the ground circuit (in principle) has also been subtracted out... leaving a clean signal.

This means that, if one could find a **reference** electrode with no neural activity, one could (in principle) have a completely clean measure of the voltage at **active** relative to the **reference**.

# Choosing a reference

The ideal choice of reference would be one with no neural activity. This is because any neural activity at the reference electrode will be subtracted from the neural activity at the active electrode... leading to complex effects.  $(\text{A} - \text{G}) - (\text{R} - \text{G})$

Unfortunately, there is no electrically neutral site on the human body that could be used as a reference. Luck calls this the *no-Switzerland principle*.

This means that all choices of reference will have an impact on the voltages that you record at active electrodes. So you have to think carefully about this.

## Some options

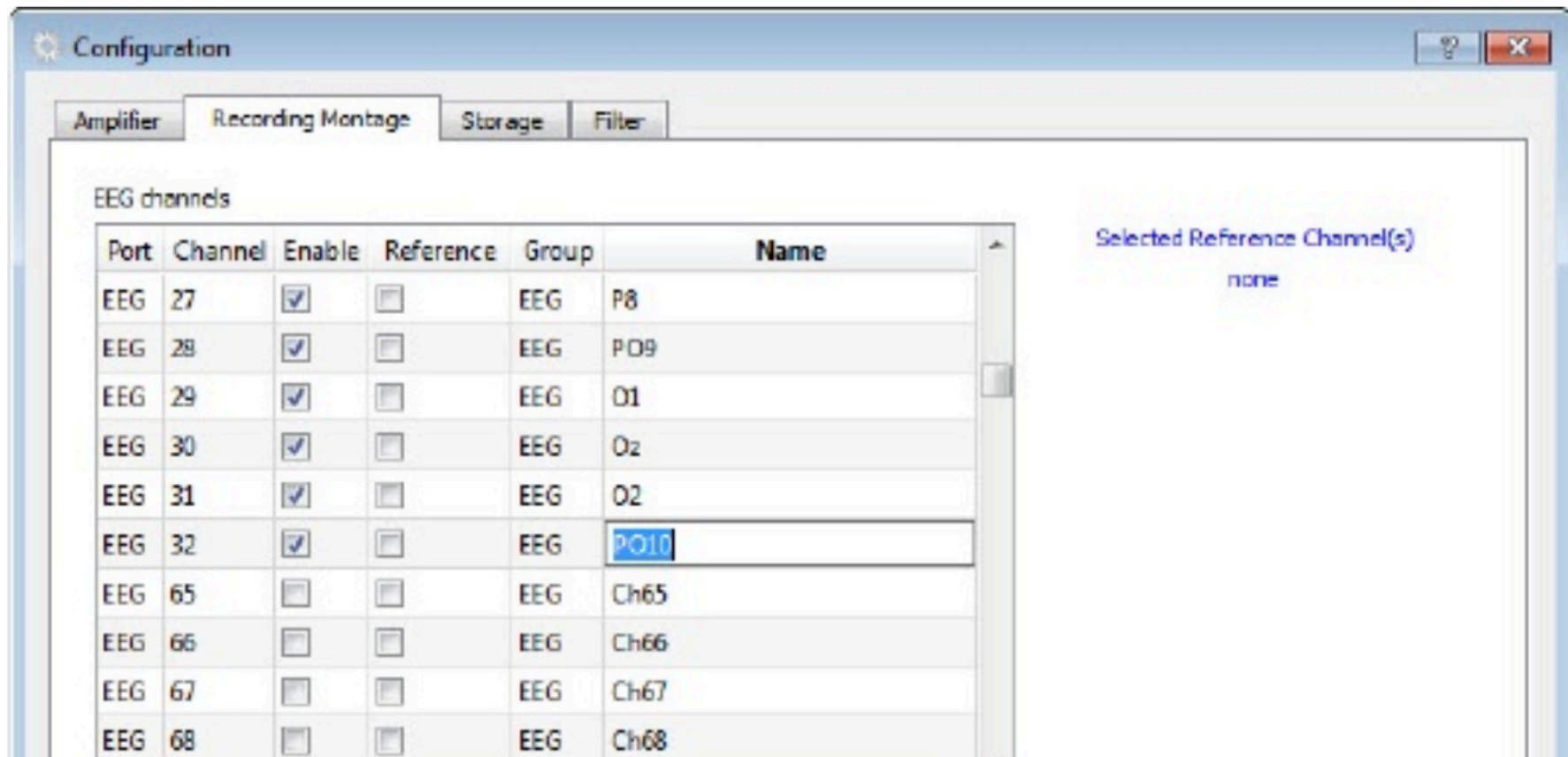
tip of the nose	left earlobe	right earlobe	average of mastoids
	left mastoid	right mastoid	average of all electrodes

Different references will yield different patterns of results, both on the scalp, and in terms of waveform morphology. It is worth thinking about why this would be so... and also worth trying different references on your data sets.

# Online reference

An [online reference](#) is one that is used during EEG recording. The software that we use to record EEG with our system is called **pycorder** (python + recorder). You can set the reference during recording by clicking configuration, then the recording montage tab, and checking the box in the reference column for the channel(s) that you want to be the reference during recording.

*Figure 2-11.* Configuring channel settings



# What if you forget to choose a reference?

Pycorder will work, and record EEG data, even if no reference is set. This is actually how the data set for this class was recorded.

When I realized that I was recording this way, I contacted Brain Products to find out what was happening. How could the amplifier be working without a reference being set?

I don't fully understand the details behind their answer. They said that the amplifier uses a "virtual reference" when none is set. This allows it to record EEG even if the user does not set a reference. They said that the only impact this should have is that the recorded data will look a bit noisier than it would otherwise be, because the virtual reference will not allow for as much of the noise to be eliminated. (Again, I don't know why.)

In such a case, you have to apply a reference **offline** (after recording the data), which will remove the rest of the noise.



# What if you chose the wrong reference?

What if you choose one reference during recording, only to later realize that you should have chosen a different reference?

In that case, you can **re-reference** your data **offline**.

The reason you can do this is that the effect of differential amplification (the equation from earlier) is to yield **A-R**. It is a simple subtraction. You can do the same thing with your data set in software.

So let's imagine that your **online reference** was the left mastoid, let's call it LM.

The voltage at the Cz electrode:  $Cz - LM$

The voltage at the right mastoid:  $RM - LM$

Now, let's say that you want to re-reference to the right mastoid. You can simply subtract RM from your active electrodes. The LM term will cancel out.

Re-reference to RM:  $(Cz - LM) - (RM - LM) = (Cz - RM)$

# The linked mastoids

One very common reference in the language processing literature is called **the linked mastoids**.

This name is a holdover from the past, when EEG was a completely mechanical process (no software). People would literally link the two mastoids using a wire. The result is an **average of the two mastoids**.

Since this is a popular reference, let's see how to calculate it.

What we want:  $Cz - (LM + RM)/2 = Cz - \frac{LM}{2} - \frac{RM}{2}$

What we have:  $(Cz - LM) \text{ and } (RM - LM)$

What we need to do:  $(Cz - LM) - \frac{(RM - LM)}{2}$  Subtract 1/2 of RM

Some algebra:  $Cz - LM - \frac{RM}{2} + \frac{LM}{2} = Cz - \frac{LM}{2} - \frac{RM}{2}$

# Re-referencing in ERPLAB

ERPLAB makes it very easy to re-reference your data.

Here is the general page for re-referencing: <https://github.com/lucklab/erplab/wiki/EEG-and-ERP-Channel-Operations>

And here is a section of the tutorial, with specific instructions for re-referencing to the linked mastoids: <https://github.com/lucklab/erplab/wiki/Creating-and-Modifying-Channels-with-Channel-Operations:-Tutorial>

# Table of contents

1. Introduction: The big picture	Luck 1
2. Fundamentals	Luck 2
3. Hands-on training and best practices	Luck 5
4. The ERP processing pipeline in EEGLAB + ERPLAB	Luck 6, 7, 8
1. Load the data.	
2. Filter the data (high-pass, possibly low-pass).	
3. ICA for artifact correction (optional).	
4. Re-reference the data.	
5. Add channel locations.	
6. Epoch the data.	
7. Artifact detection and rejection.	
8. Average the epochs to create subject ERPs.	
9. Average the subject ERPs to create a grand average ERP.	
10. Plotting: waveforms, topoplots, difference waves	
11. Measure amplitudes and latencies.	
12. Run statistical tests.	
5. Creating a script for EEGLAB + ERPLAB	
6. Creating a script for Fieldtrip	

# Adding channel locations

The data that we record will likely have channel labels (e.g., Cz), but no information about where that channel is in space.

In order to do things that require space information, like plotting scalp maps or determining which electrodes are near each other, we have to load channel location information into our data set.

EEGLAB comes with standard channel locations built in. We just have to tell it to apply those channel locations to the channel labels in our data set.

Here is the EEGLAB page for this: [https://sccn.ucsd.edu/wiki/A03:\\_Importing\\_Channel\\_Locations](https://sccn.ucsd.edu/wiki/A03:_Importing_Channel_Locations)

Channel locations need to be added every time you create new channels. The re-referencing procedure in ERPLAB creates new channels (if you use the nchan syntax), so you have to add (or re-add) channel locations after re-referencing.

# Table of contents

1. Introduction: The big picture
2. Fundamentals
3. Hands-on training and best practices
4. **The ERP processing pipeline in EEGLAB + ERPLAB**
  1. Load the data.
  2. Filter the data (high-pass, possibly low-pass).
  3. ICA for artifact correction (optional).
  4. Re-reference the data.
  5. Add channel locations.
  6. **Epoch the data.**
  7. Artifact detection and rejection.
  8. Average the epochs to create subject ERPs.
  9. Average the subject ERPs to create a grand average ERP.
  10. Plotting: waveforms, topoplots, difference waves
  11. Measure amplitudes and latencies.
  12. Run statistical tests.
5. Creating a script for EEGLAB + ERPLAB
6. Creating a script for Fieldtrip

Luck 1

Luck 2

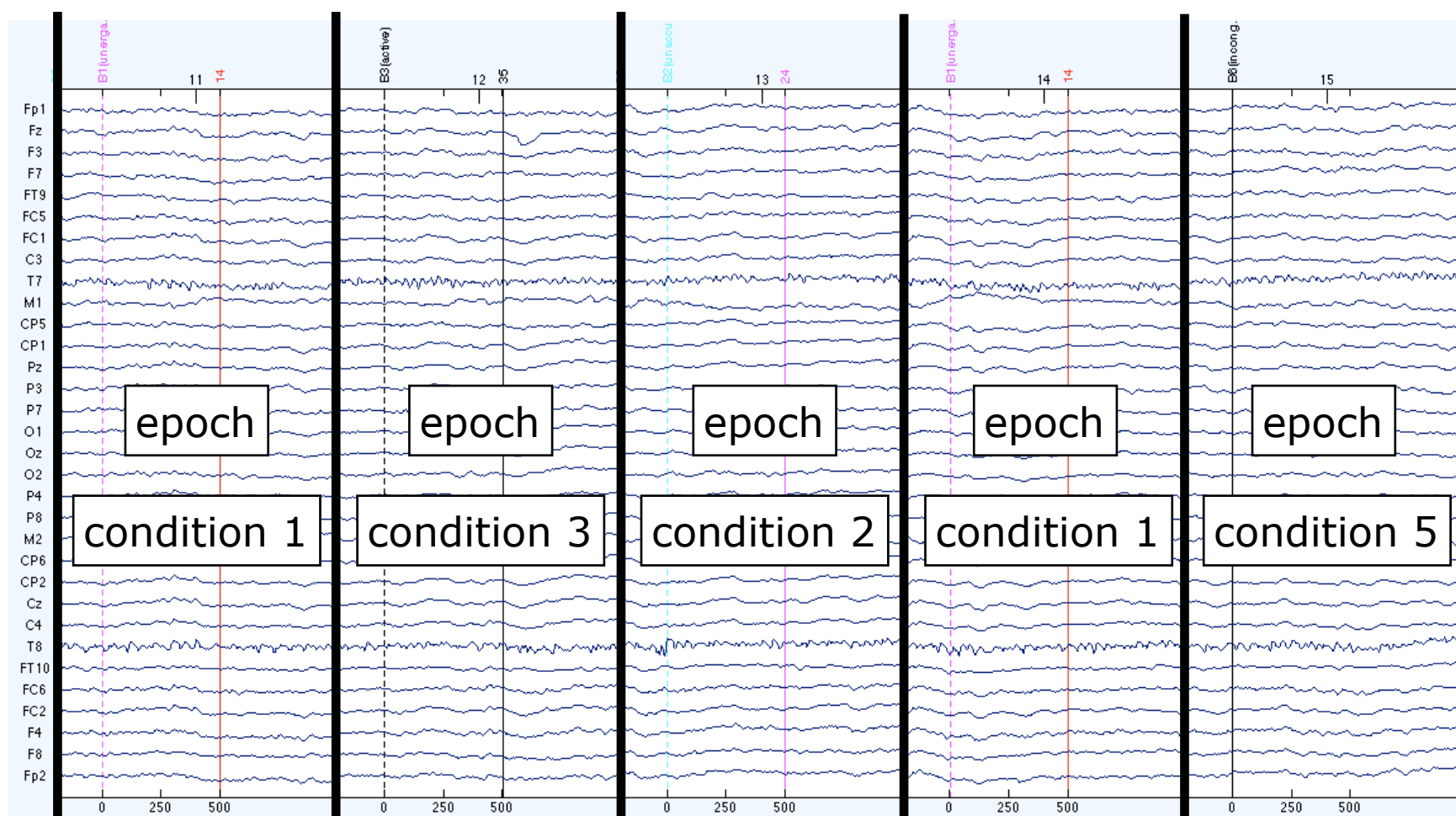
Luck 5

Luck 6, 7, 8

# Epoching the data

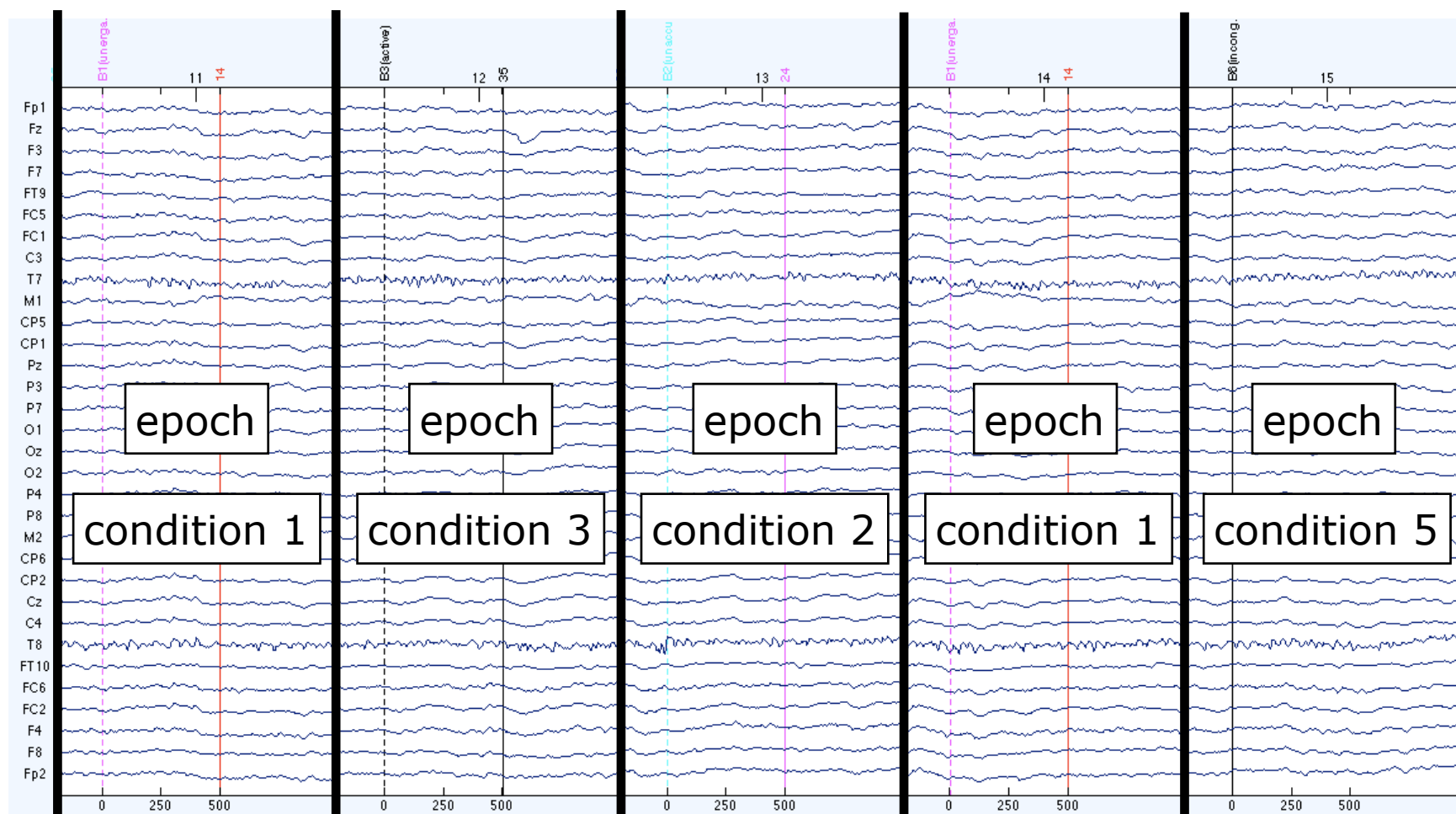
The next step is to cut the continuous EEG into smaller epochs, with each epoch containing a single trial.

This is where event codes come into play. We can use the event codes in the EEG data to identify the different trials. These show up as lines in EEGLAB.



# Epoching the data

To create these epochs we need to tell ERPLAB two things: (i) which event codes represent each type of trial, and (ii) how much time to cut out around each event code.



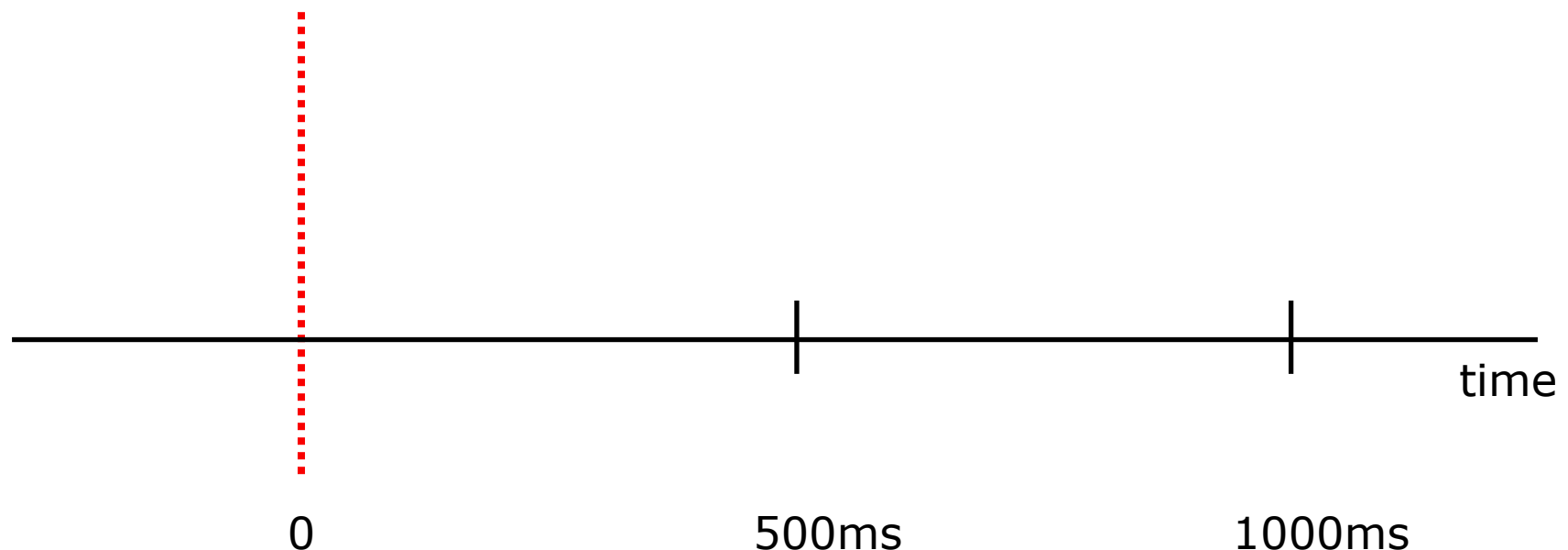


# Choosing the rightward bound

Typically, the event code represents time point 0 in creating an ERP — the point that is used to time-lock the ERP.

The choice of the rightward bound of the epoch is typically based on two factors. The first is the presentation rate of your trials. You typically want to create an epoch that is an integer multiple of the presentation rate, so that the epoch contains a whole number of trials.

For example, if the presentation rate is 500ms per trial, then the options for epoch end-point are 500ms, 1000ms, 1500ms, etc.

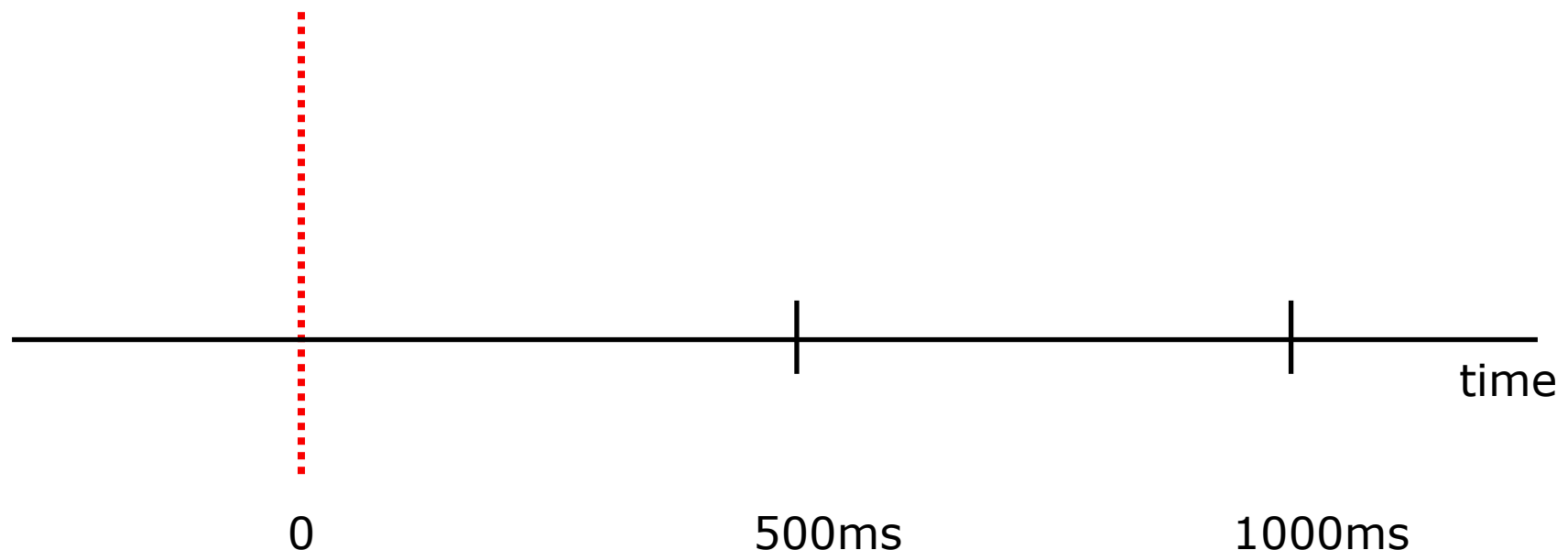


# Choosing the rightward bound

The second factor affecting the rightward bound of the epoch is the ERP that you are interested in detecting.

Different ERPs occur in different (post-stimulus) time windows: the LAN is 300-500ms, the N400 is 300-500ms, the P600 is 500-800ms, and sustained negativities can be even later (1000ms to 3000ms in one experiment I ran).

You want to choose a rightward bound that will allow you to see your ERP of interest. For example, if you were interested in P600s with a 500ms presentation rate, you would probably choose an epoch with a rightward bound of 1000ms.

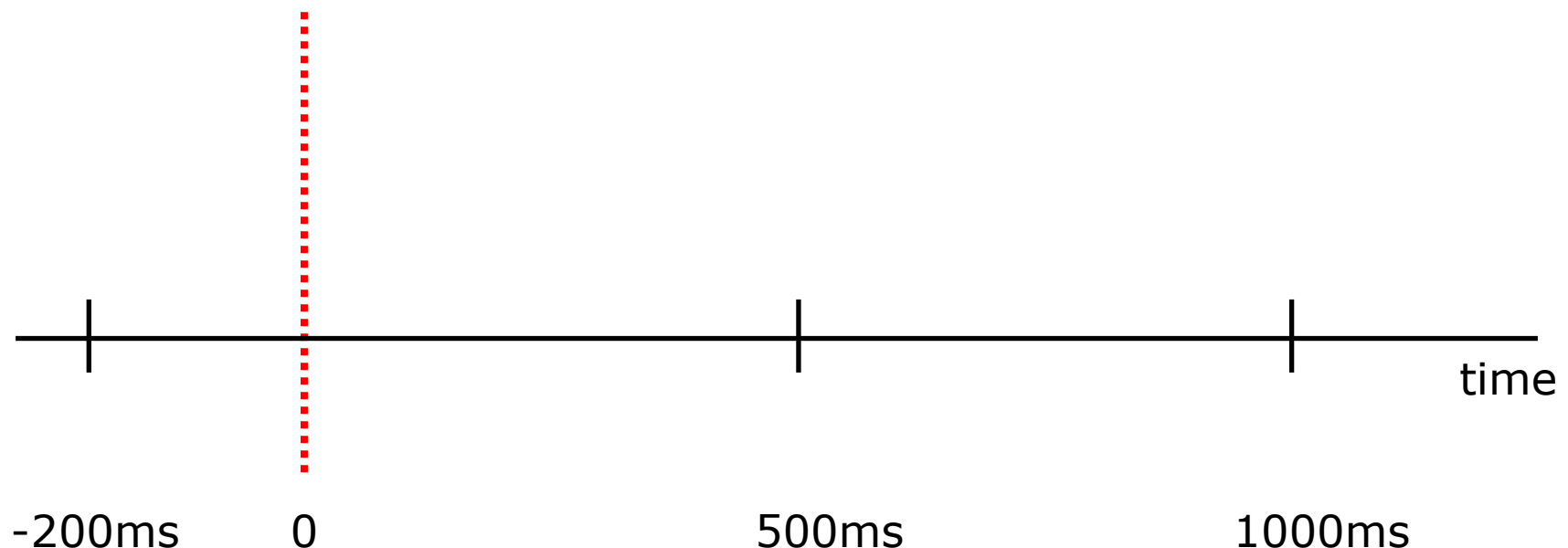


# Choosing the leftward bound

The leftward bound of the epoch will typically not be 0. This might sound strange at first, but the reason for this is that you will typically want a pre-stimulus period of activity that you can use for a [baseline](#).

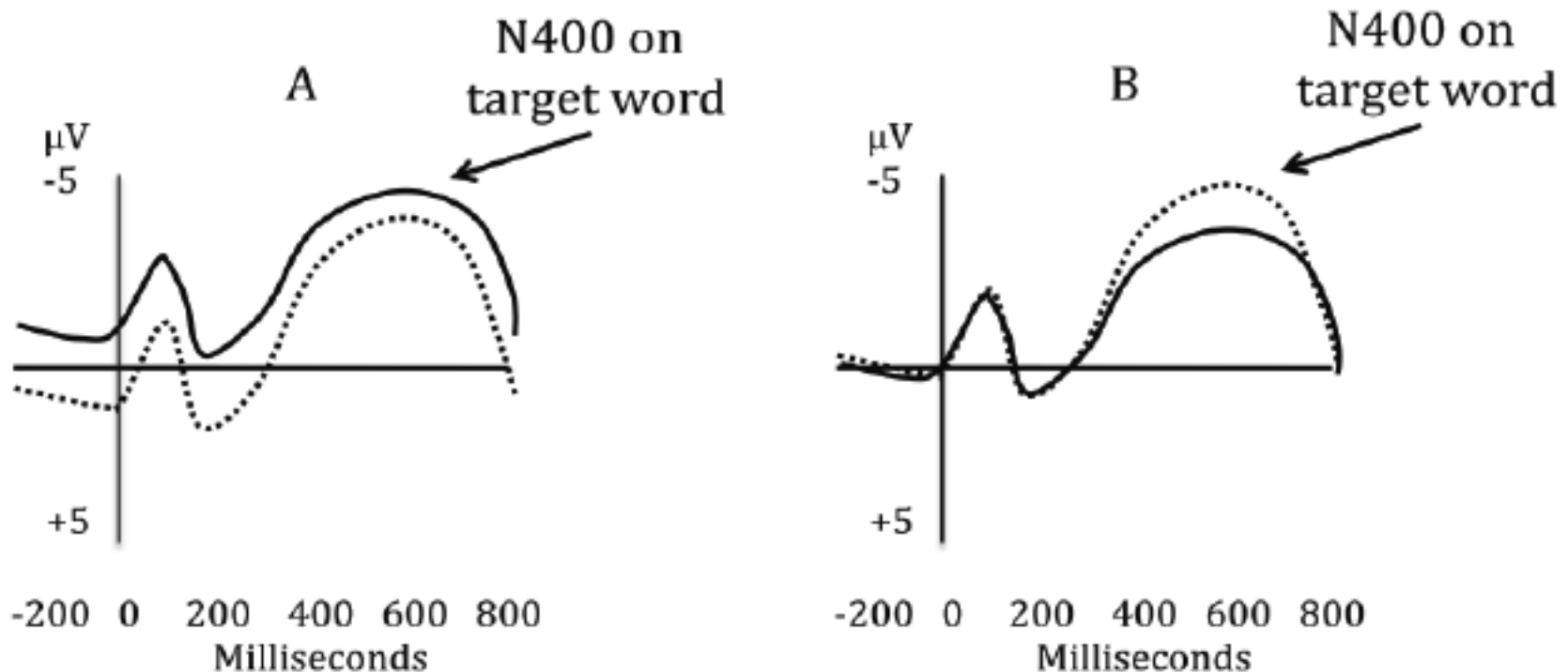
The pre-stimulus baseline period will typically be either 100ms or 200ms. I do not know of any systematic studies of the choice between the two. I tend to use 200ms.

But the real question for us at this point is this — What is the purpose of a baseline?



# Baselines in EEG

The logic behind a **baseline** is captured right there in the name - the goal is to determine what the activity level was prior to the onset of the stimulus, so that you can detect any changes from that baseline activity that are **driven by the stimulus**.



In plot A, there appears to be a difference between the two conditions (black is higher than dotted). But that difference begins before the stimulus (0ms), so it cannot be due to the stimulus. Baseline correction eliminates this pre-stimulus difference, allowing us to see the effect driven by the stimulus (an N400).

# Baseline correction: Absolute baseline

There are several different types of baseline correction that one can do to EEG signals. Here is a brief discussion: [http://bjornherrmann.com/baseline\\_correction.html](http://bjornherrmann.com/baseline_correction.html).

ERPs typically use what is known as the **absolute baseline**. We will see the others when we look at time-frequency analysis later in the semester.

The first step to calculate an absolute baseline is to identify the baseline period. For us, it will be -200ms to 0ms.

The second step is to calculate the mean amplitude in the baseline period. We do this for each epoch separately (they each have their own baseline).

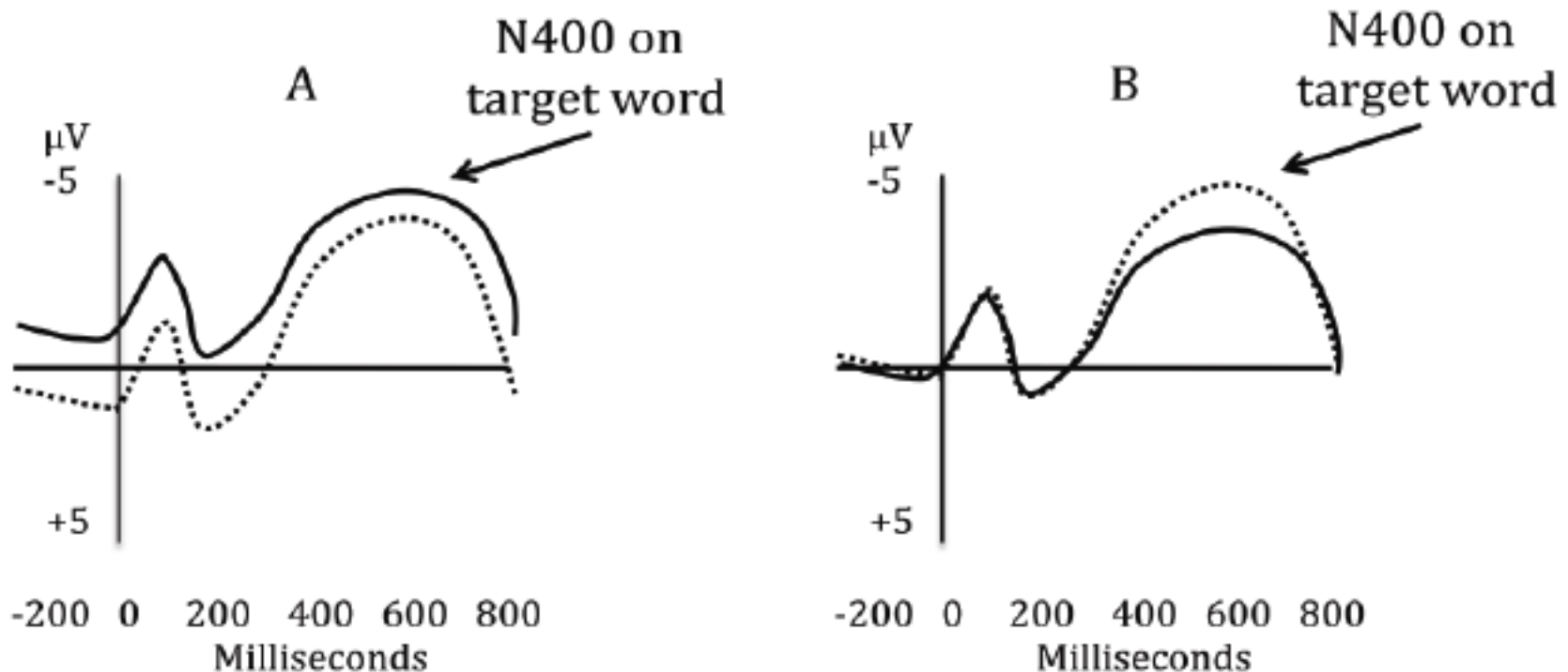
The final step is to take the mean amplitude during the baseline period, let's call it  $B$ , and subtract  $B$  from each data point during the entire epoch.

If you have taken statistics courses, you will recognize this as a type of **mean centering**. We are centering the entire time series around the mean of the baseline period.

This has the effect of moving the waveform up or down, such that it is centered around 0 during the baseline period.

# The absolute baseline in action

Plot A shows the raw waveforms for two conditions before baseline correction, and plot B shows the two waveforms after absolute baseline correction.



You can see that the solid waveform is moved down to the 0 line by the correction, and the dotted waveform is moved up. This makes the two waveforms roughly equal in amplitude during the baseline period, so that any differences that we see post-stimulus can be better attributed to the stimulus. In this case, there is an N400.

# Epoching data in ERPLAB

Epoching data in ERPLAB is a two step process.

The first step is to create a data structure that ERPLAB calls an **eventlist**. The eventlist tells ERPLAB which event codes represent important trials, and how to divide the event codes into different **bins** representing the **conditions** of your experiment.

Here is the ERPLAB tutorial for creating an eventlist using bins for distinct conditions: <https://github.com/lucklab/erplab/wiki/Advanced-EventList-Options:-Tutorial>

The second step is to tell ERPLAB to cut the epochs out of the continuous EEG according to the eventlist. ERPLAB calls this **extracting bin-based epochs**.

Here is the tutorial for extracting bin-based epochs: <https://github.com/lucklab/erplab/wiki/Creating-Bin--Based-EEG-Epochs:-Tutorial>

# Demeaning and Detrending

- Demean:** To subtract the mean amplitude of the entire epoch from each time point in the epoch (i.e., mean centering the epoch).
- Detrend:** To look for severe linear trends over the entire time period of the epoch, and regress them out.

These are two optional pre-processing steps that can be applied to epochs.

The goal for both is to minimize the amount of data that is lost to artifact detection.

Demeaning minimizes the chances that a vertical offset (increase in amplitude) over the entire epoch will trigger a threshold-based artifact detection algorithm.

Detrending minimizes the chances that a linear trend (like a slow skin potential or sweat artifact) will trigger a threshold-based artifact detection algorithm.

I have tested both against several of my own data sets, including data sets that find sustained ERPs. They do not appear to eliminate real ERPs. They do appear to increase the amount of data that survives artifact detection.



# Demeaning and Detrending in ERPLAB

ERPLAB gives you the option to demean during filtering: <https://github.com/lucklab/erplab/wiki/Filtering>

You could also do it using the baseline function `pop_blcerp()` and setting the baseline window to 'whole' (for the entire epoch). This is because the absolute baseline correction is just demeaning with the mean from the pre-stimulus baseline!

I don't believe there is a GUI option for detrending. But ERPLAB does have a function called `lindetrend()` that you can use to do it if you are using a script for your data analysis.

We will discuss creating an analysis script later in this course.

# Table of contents

1. Introduction: The big picture
2. Fundamentals
3. Hands-on training and best practices
4. **The ERP processing pipeline in EEGLAB + ERPLAB**
  1. Load the data.
  2. Filter the data (high-pass, possibly low-pass).
  3. ICA for artifact correction (optional).
  4. Re-reference the data.
  5. Add channel locations.
  6. Epoch the data.
  7. **Artifact detection and rejection.**
  8. Average the epochs to create subject ERPs.
  9. Average the subject ERPs to create a grand average ERP.
  10. Plotting: waveforms, topoplots, difference waves
  11. Measure amplitudes and latencies.
  12. Run statistical tests.
5. Creating a script for EEGLAB + ERPLAB
6. Creating a script for Fieldtrip

Luck 1

Luck 2

Luck 5

Luck 6, 7, 8

# Artifact detection and rejection

<b>Artifact:</b>	Anything in the EEG signal that was not caused by neural activity.
<b>Artifact detection:</b>	Identifying artifacts in the EEG signal. This can be done manually, automatically, or both.
<b>Artifact rejection:</b>	Deleting epochs (or segments of the continuous EEG) that contain artifacts.
<b>Artifact correction:</b>	Subtracting artifact components from the EEG signal without deleting any epochs/segments.

The goals of this section are to (i) learn how to identify artifacts in EEG, (ii) learn how to use automatic artifact detection algorithms to make that detection rigorous, and (iii) learn how to remove artifacts from the data set.

# An introduction to EEG artifacts

Learning to identify artifacts in EEG just takes time and practice. Actually, noticing that there is an artifact in continuous EEG is easy — it looks different from clean EEG. But learning to identify the cause of the artifact takes some practice.

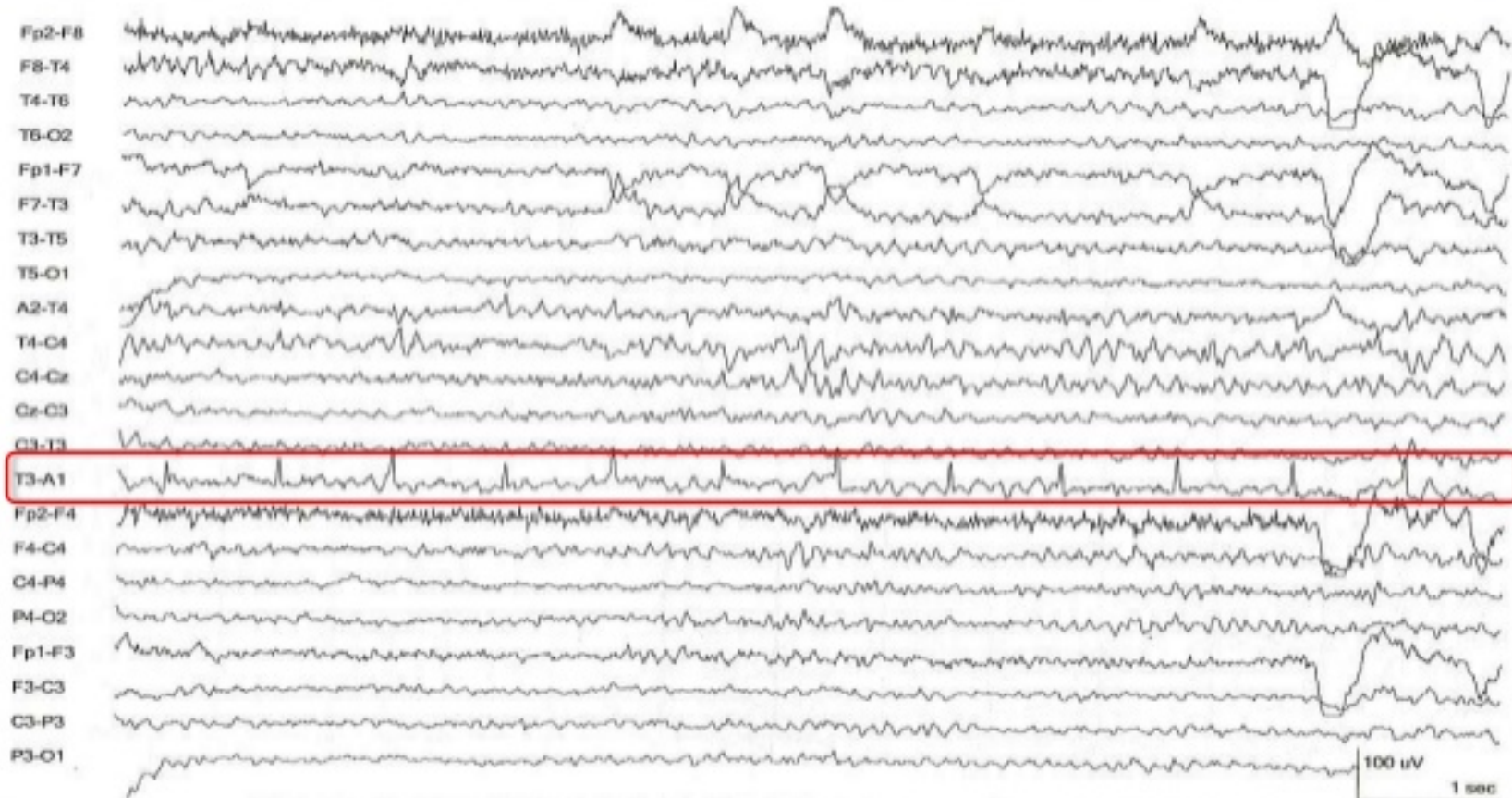
Here is a slideshow with images of artifacts in EEG, along with some discussion of how they are generated: <https://www.slideshare.net/SudhakarMarella/eeg-artifacts-15175461>

Here is a website with with similar images and discussion: <https://emedicine.medscape.com/article/1140247-overview#a2>

Let's look at some artifacts together!

# An introduction to EEG artifacts

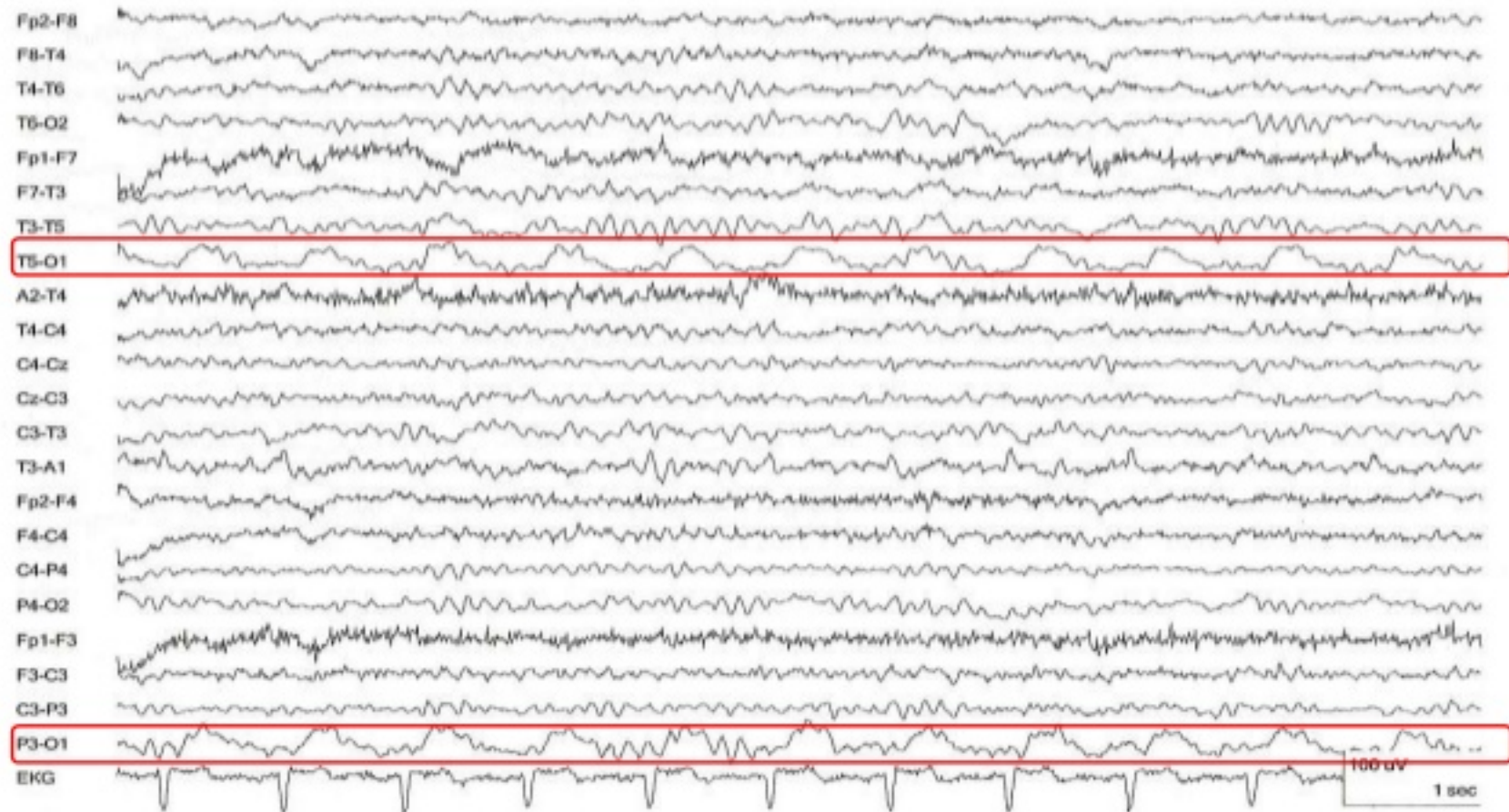
## Cardiac artifact



ECG artifact is identified by its fixed period and morphology and is limited to T3-A1 channel in this bipolar montage

# An introduction to EEG artifacts

## Pulse artifact

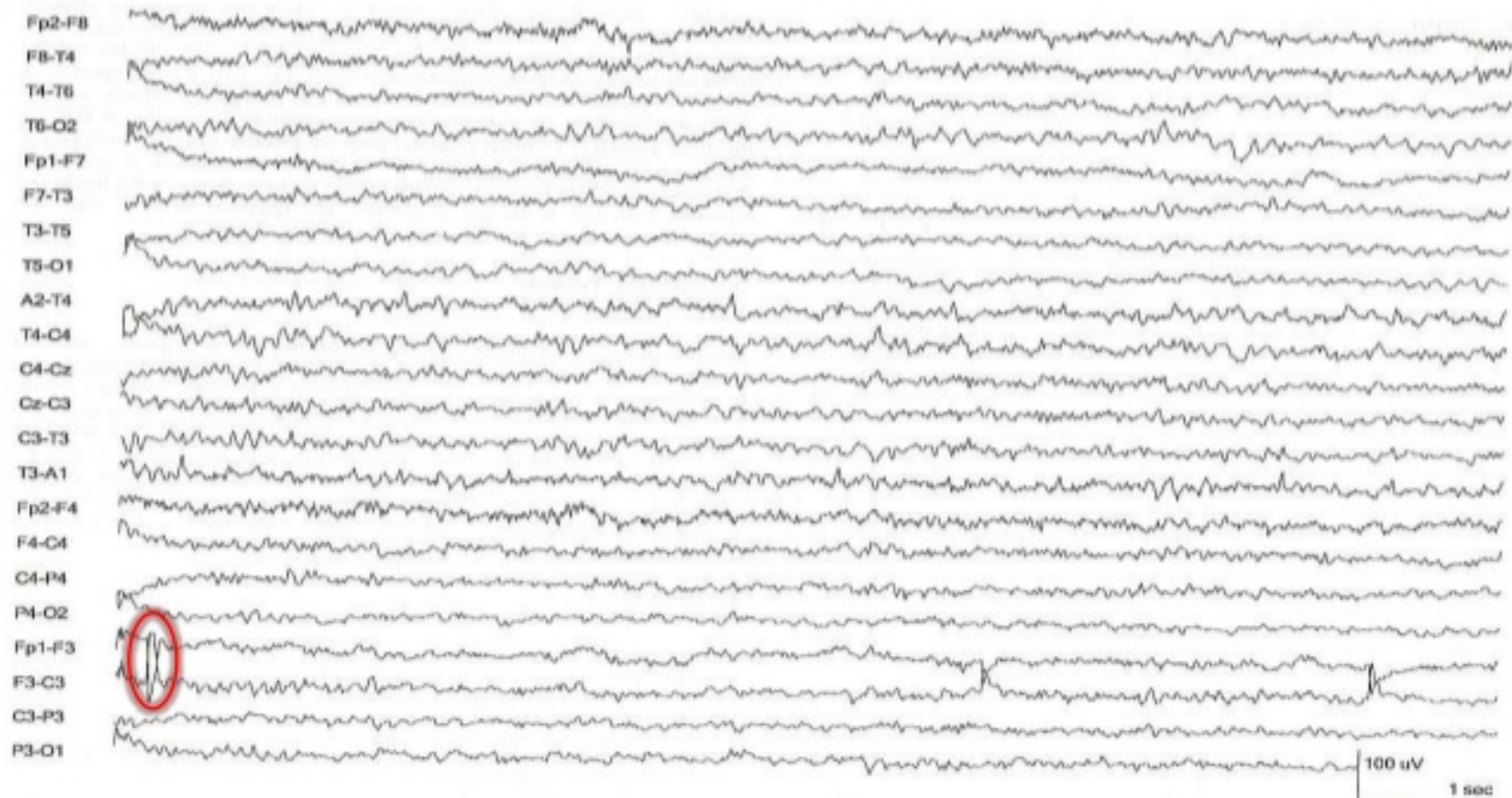


Focal slow waves at the left occiput follow each heart beat, as indicated in the EKG channel. The slow waves are artifact due to electrode movement and were eliminated by repositioning the O1 electrode



# An introduction to EEG artifacts

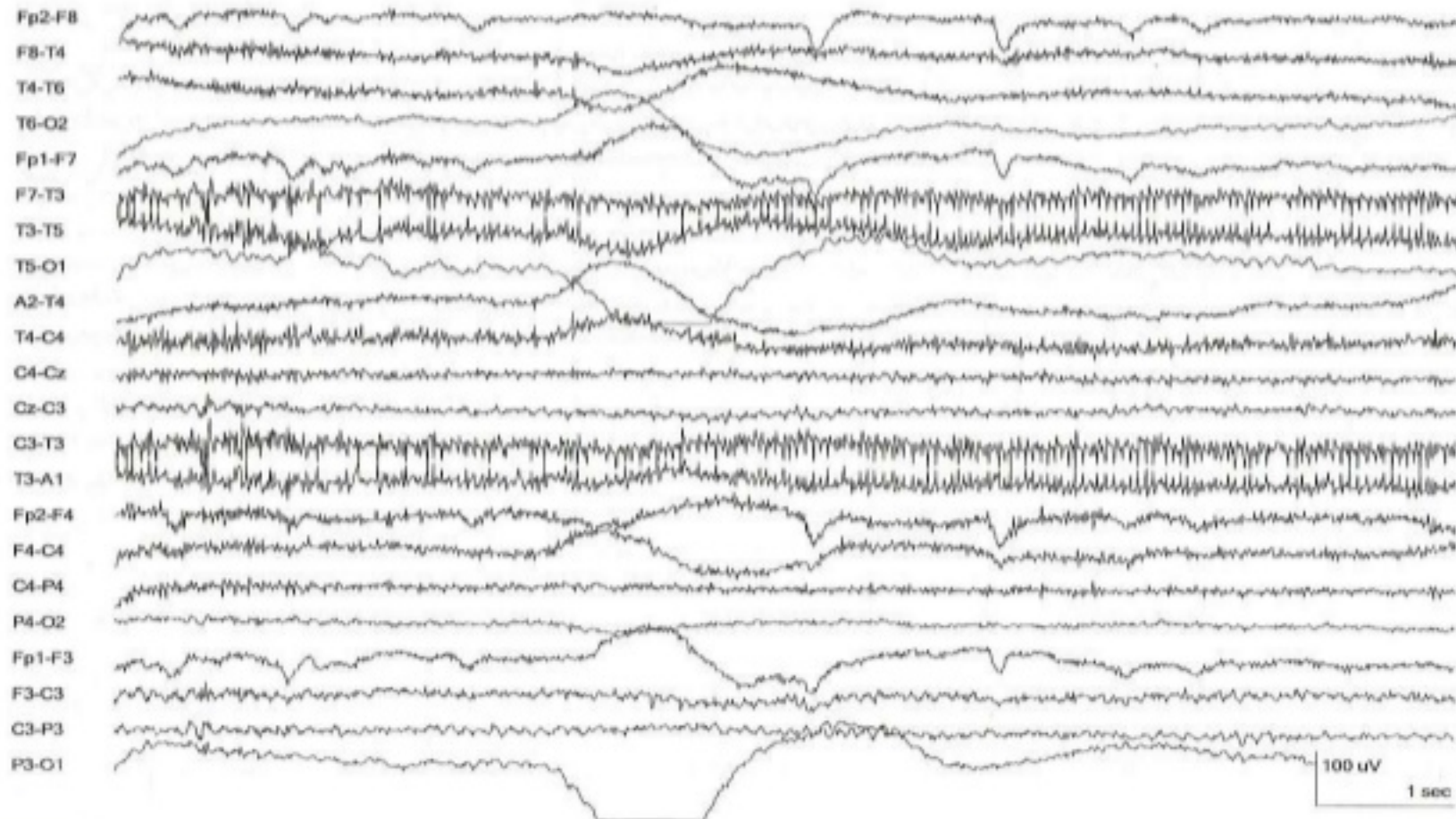
## Electrode pop



The nearly vertical rise followed by the slower fall at the F3 electrode is typical of electrode pop artifact. Also typical is an amplitude that is much greater than the surrounding activity, a field that is limited to one electrode, and repeated recurrence within a short time

# An introduction to EEG artifacts

## Lead movement

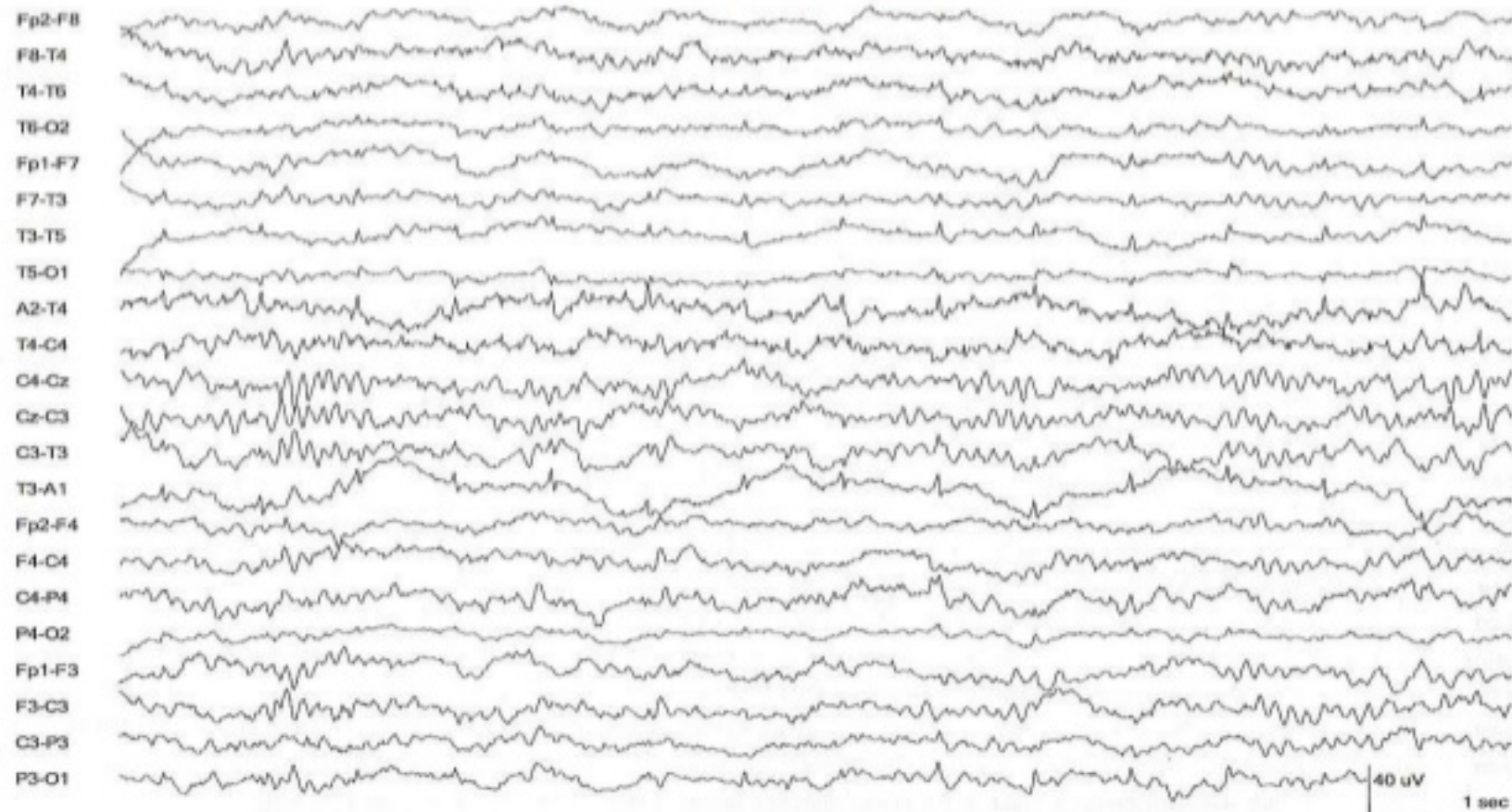


Multiple channels demonstrate the artifact through activity that is both unusually high amplitude and low frequency and also disorganized without a plausible field



# An introduction to EEG artifacts

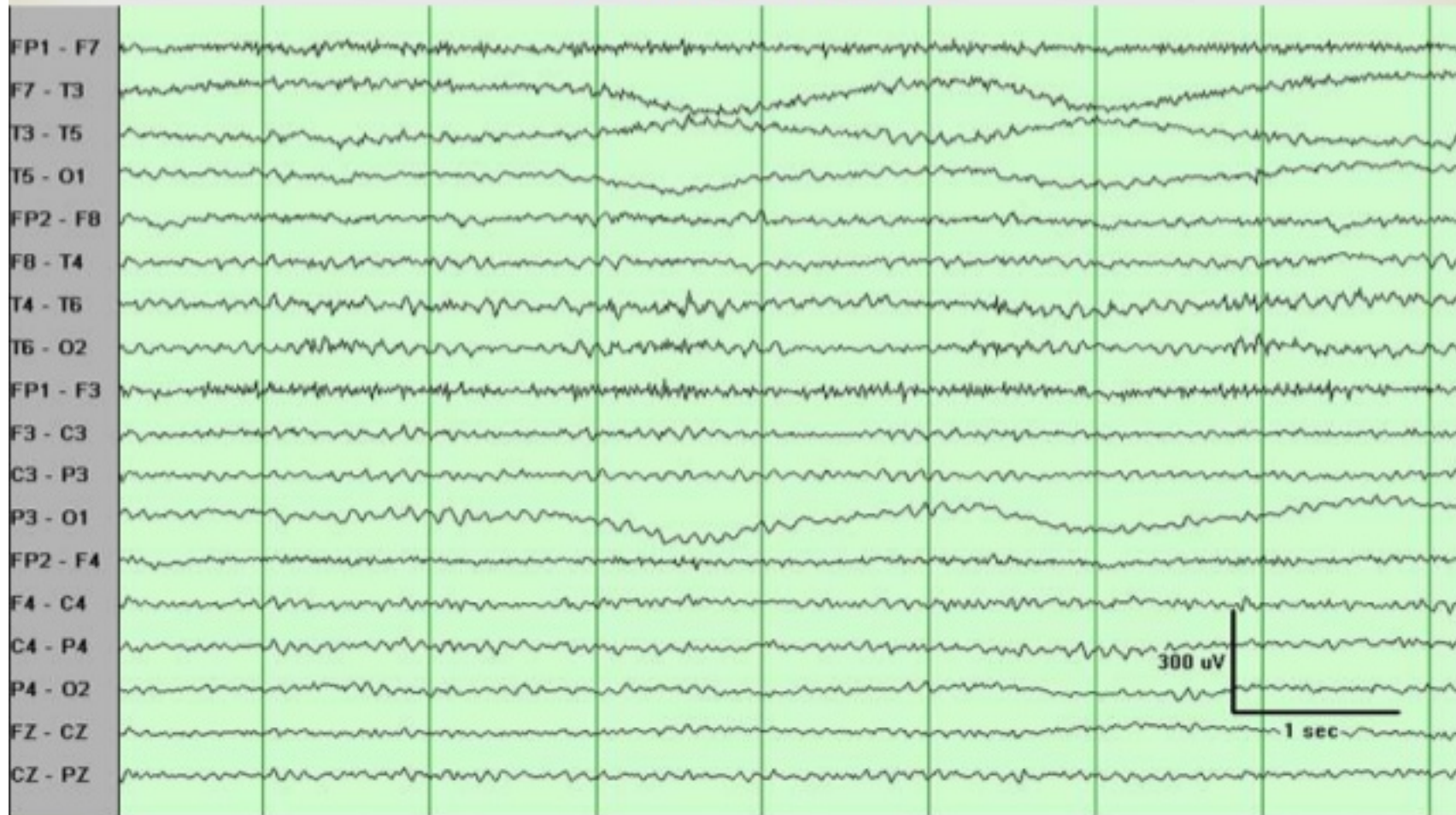
## Sweat artifact



The decreased amplitude and very low frequency oscillations are present diffusely, which is consistent with the whole scalp's involvement. The recurring sharp waves across most channels are ECG artifact.

# An introduction to EEG artifacts

## Sweat artifact

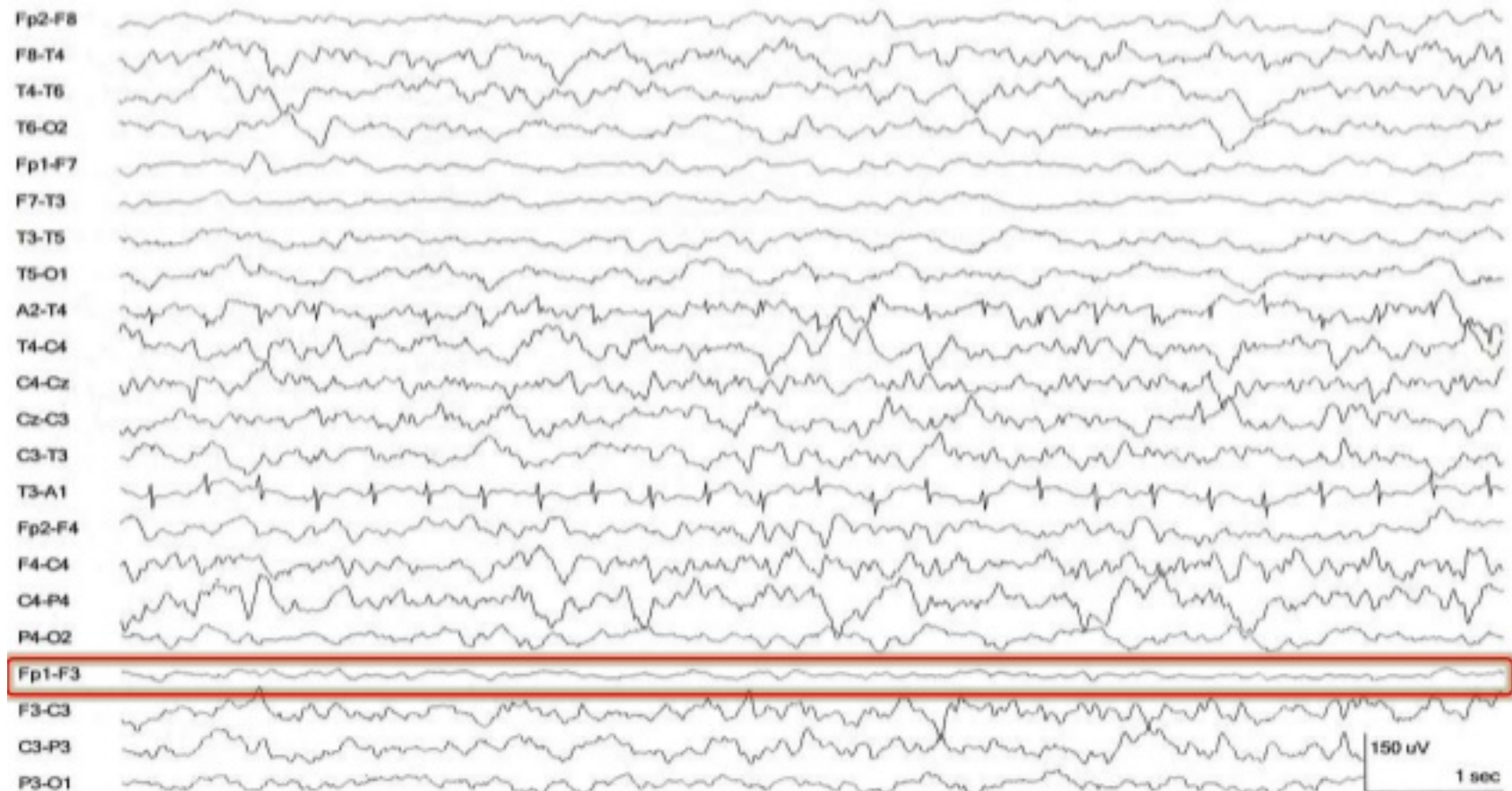


This is characterized by very low-frequency (here, 0.25- to 0.5-Hz) oscillations. The distribution here (midtemporal electrode T3 and occipital electrode O1) suggests sweat on the left side



# An introduction to EEG artifacts

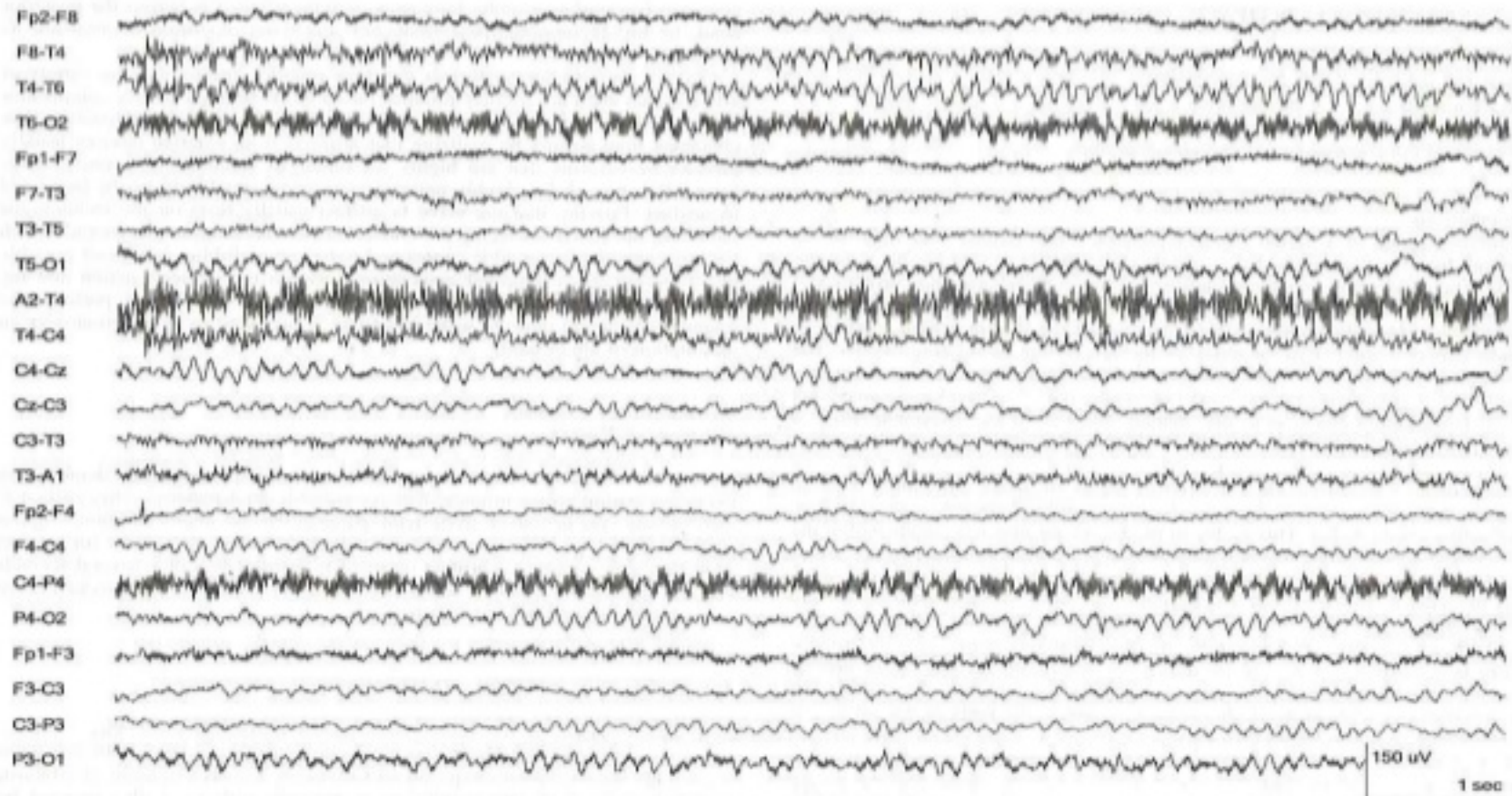
## Salt bridge artifact



Activity in channels that include left frontal electrodes is much lower in amplitude and frequency than the remaining background. The lack of these findings when viewed in a referential montage confirms that an electrolyte bridge is present among the electrodes involved.

# An introduction to EEG artifacts

## 60 Hz artifact

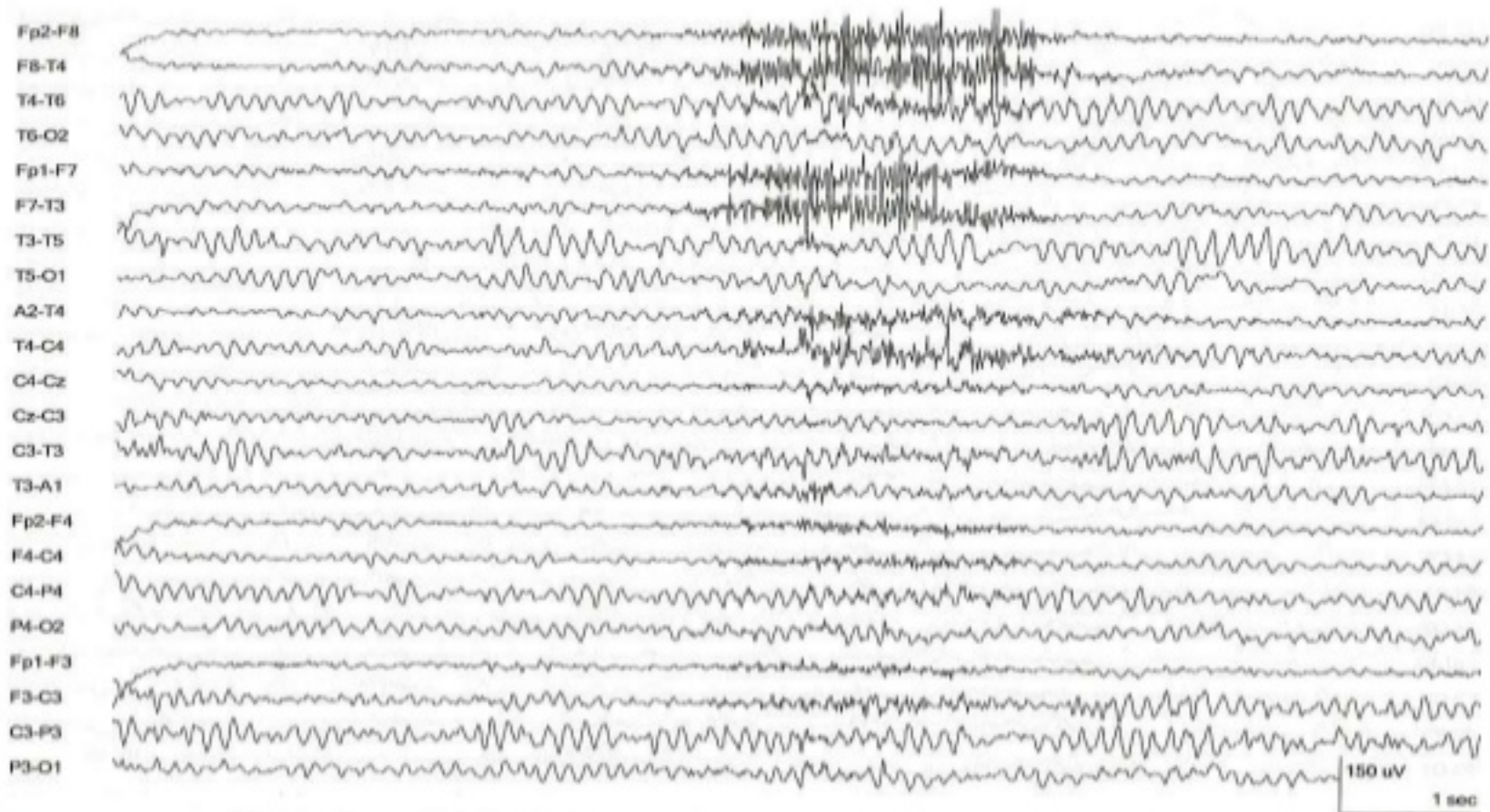


The very high frequency artifact does not vary and is present in the posterior central region, which does not typically manifest muscle artifact. This example was generated by eliminating the 60Hz notch filter.



# An introduction to EEG artifacts

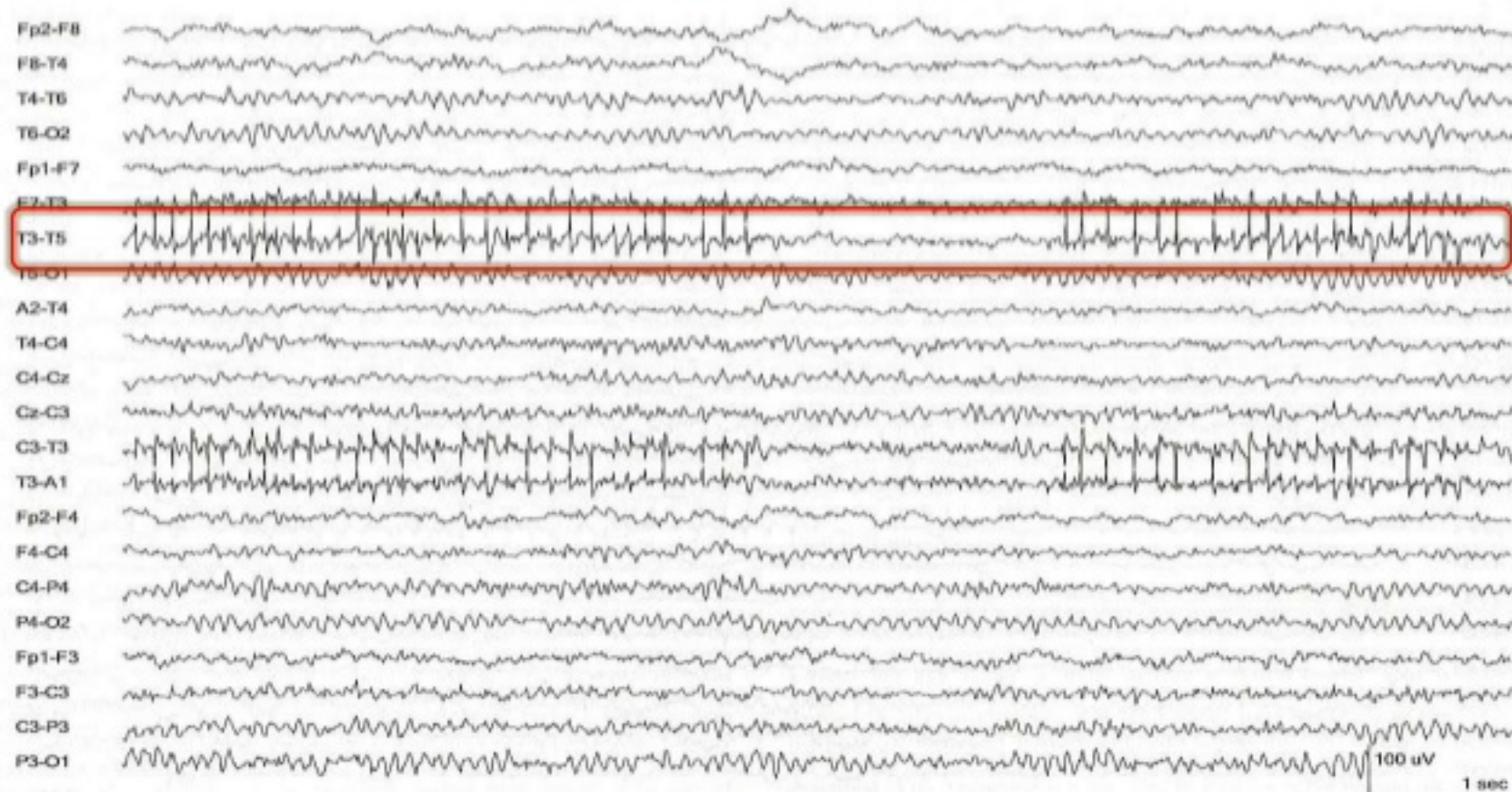
## Muscle artifact



The high amplitude, fast activity across the b/l ant. region is due to facial muscle contraction and has a distribution that reflects the locations of the muscles generating it. Typical of muscle artifact, it begins and ends abruptly.

# An introduction to EEG artifacts

## EMG artifact

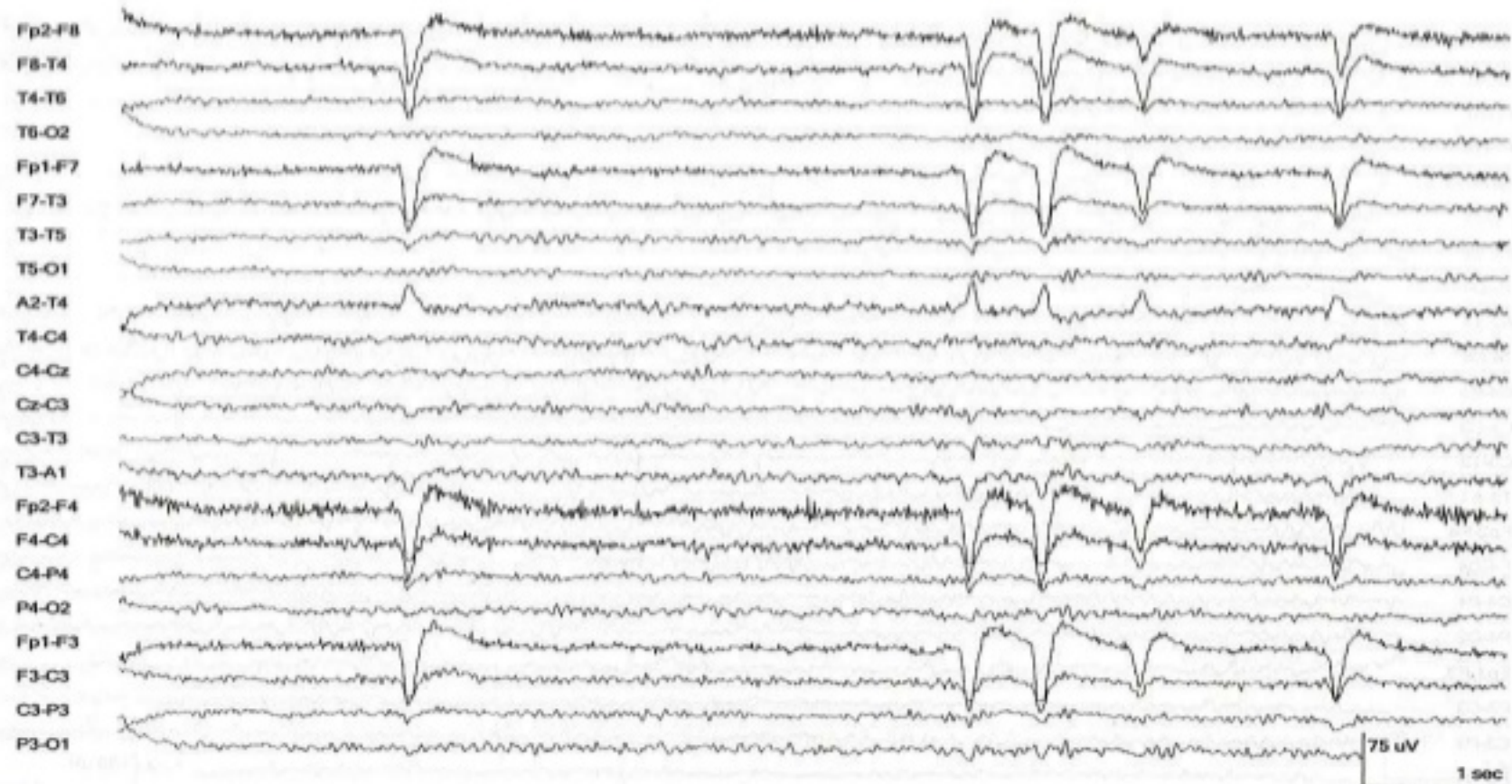


The stereotyped potentials at the T3 electrode are EMG artifact. The potentials' duration are briefer than cerebrally generated spikes, and unlike cerebrally generated activity, they have a field limited to one electrode.



# An introduction to EEG artifacts

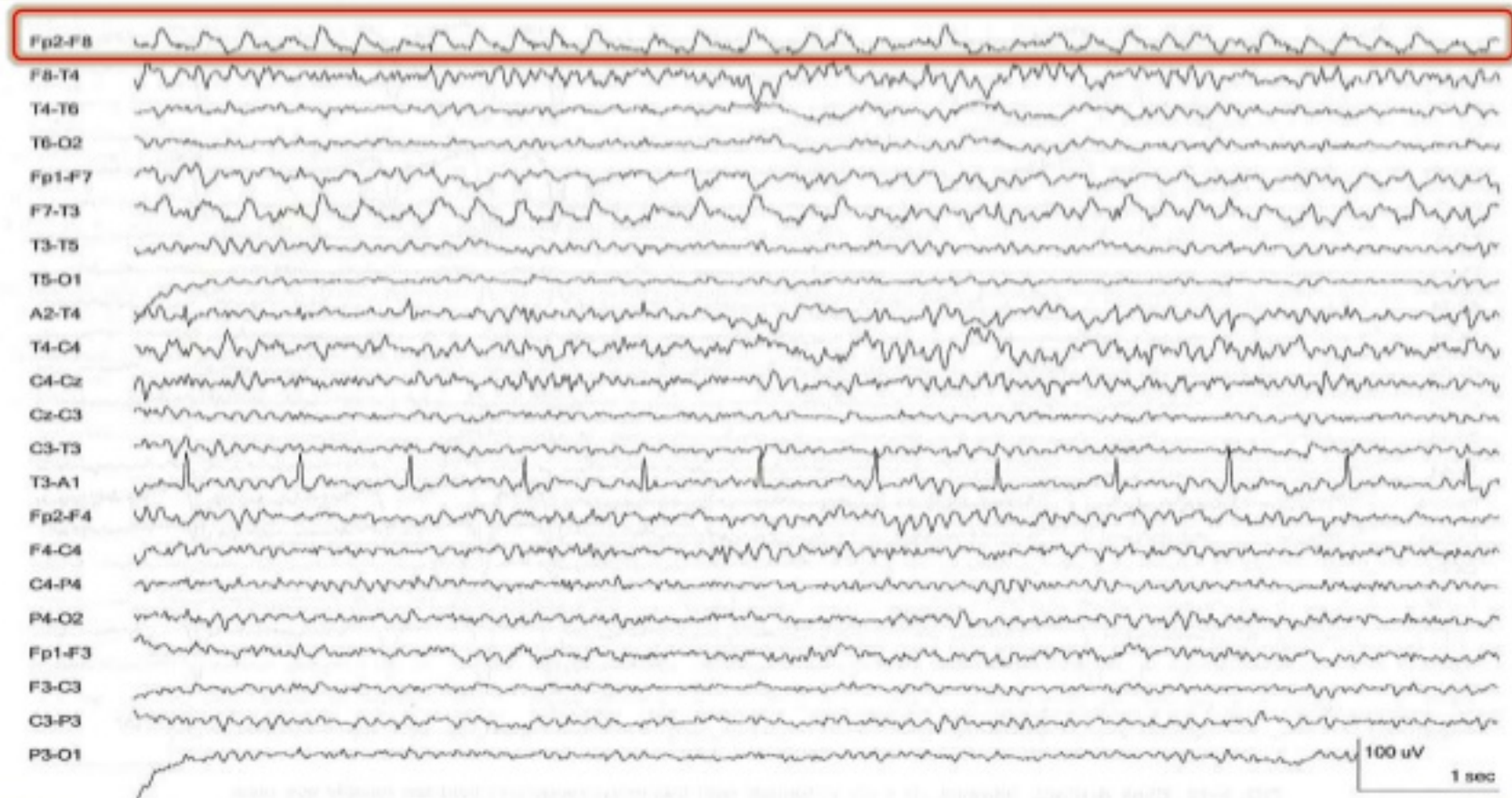
## Blink artifact



Bifrontal, diphasic potentials with this morphology and field are reliably eye blink artifact.

# An introduction to EEG artifacts

## Eye flutter artifact

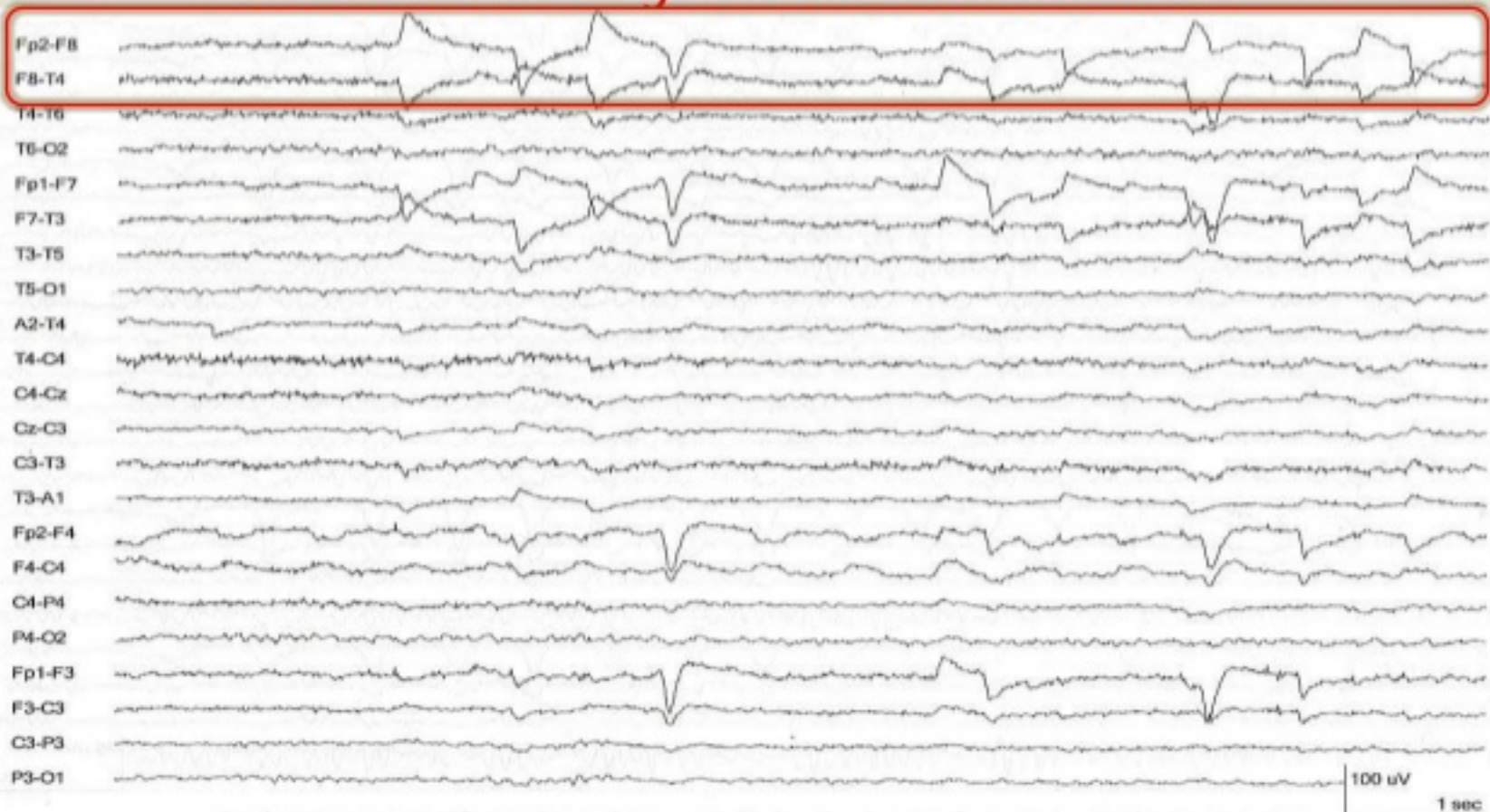


Medium amplitude, low frequency activity that is confined to the frontal poles is identified as ocular artifact through its morphology. Compared to blink artifact, flutter artifact typically has a lower amplitude and a more rhythmic appearance



# An introduction to EEG artifacts

## Lateral eye movement



Although a horizontal, frontal dipole is the key finding with lateral eye movements, the artifact is also distinguished by its morphology, which has a more abrupt transition between the positive and negative slopes than blinks and most flutter. The initial gaze in this segment is to the right.

# Manual artifact detection and rejection

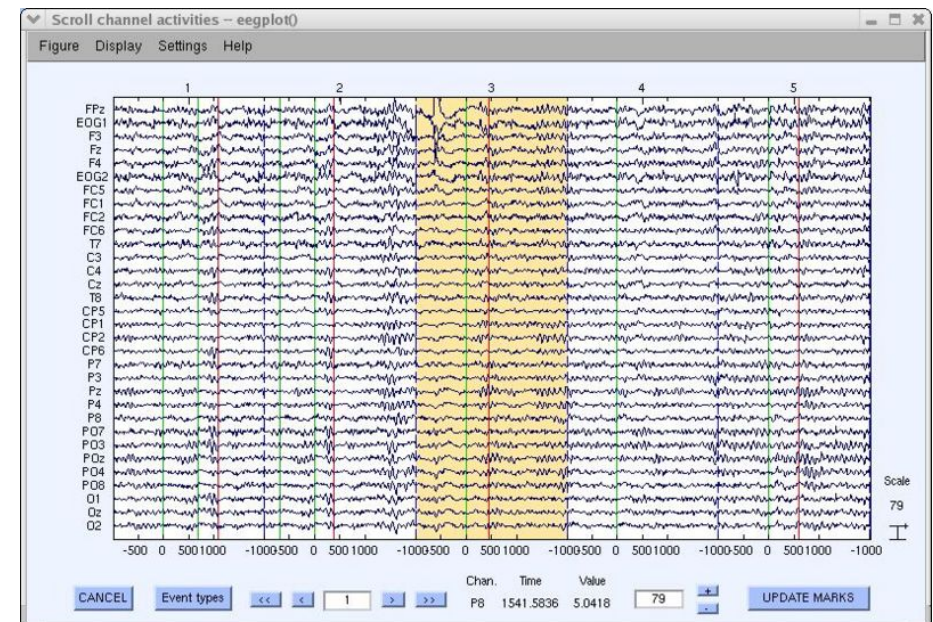
Manual artifact detection is just what it sounds like, looking through the epochs, and using your own mind to identify epochs that contain artifacts.

EEGLAB provides a nice interface for doing this. EEGLAB presents the epochs in sequence (with an appearance similar to continuous EEG) so you can quickly scroll through them. If you click on an epoch, it is marked as an artifact.

Here is the tutorial page for all artifact detection in EEGLAB: [https://scn.ucsd.edu/wiki/Chapter\\_01:\\_Rejecting\\_Artifacts](https://scn.ucsd.edu/wiki/Chapter_01:_Rejecting_Artifacts)

There are two ways to access manual artifact detection in EEGLAB. The first only accesses manual detection/rejection: **Tools -> Reject data epochs -> Reject by inspection.**

The second accesses all of the tools: **Tools -> Reject data epochs -> Reject data (all methods).**



# Automatic artifact detection and rejection

Automatic artifact detection uses mathematical algorithms to identify artifacts. This removes human judgment from the equation.

Both EEGLAB and ERPLAB come with automatic detection tools.

**EEGLAB:** [https://sccn.ucsd.edu/wiki/Chapter\\_01:\\_Rejecting\\_Artifacts](https://sccn.ucsd.edu/wiki/Chapter_01:_Rejecting_Artifacts)

**ERPLAB:** <https://github.com/lucklab/erplab/wiki/Artifact-Detection:-Tutorial>

These algorithms often need one or more parameters to be set before they will run. The general procedure is as follows:

1. Choose a value for the parameter.
2. Run the detection algorithm on the participant's data.
3. Look at the results. The goal is for the algorithm to have identified all of the artifacts that you can see, and no epochs that are clearly devoid of artifacts.
4. If the algorithm detected too few or too many artifacts, change the parameter value, and run it again.

The point of this is to make the detection process rigorous. Humans can apply criteria unequally; algorithms apply criteria uniformly.

# Fully automatic detection/rejection

In principle, it is possible to use these algorithms to create a fully automated pipeline for artifact detection and rejection.

This is **not recommended** for most (normal-sized) ERP experiments. The parameters that work well for one participant may not work well for another. Judicious human intervention can help to optimize the algorithms.

That said, there are situations where fully automated algorithms can be helpful:

1. If you are in a rush, and just want to see some basic results.
2. If you have so much data that it would be impossible for a human to look at it all.
3. If you need to standardize the analysis pipeline.

We won't look at any of these now. You can google for them if you want. They have acronym names like FASTER, PREP, HAPPE, FORCEe, etc.

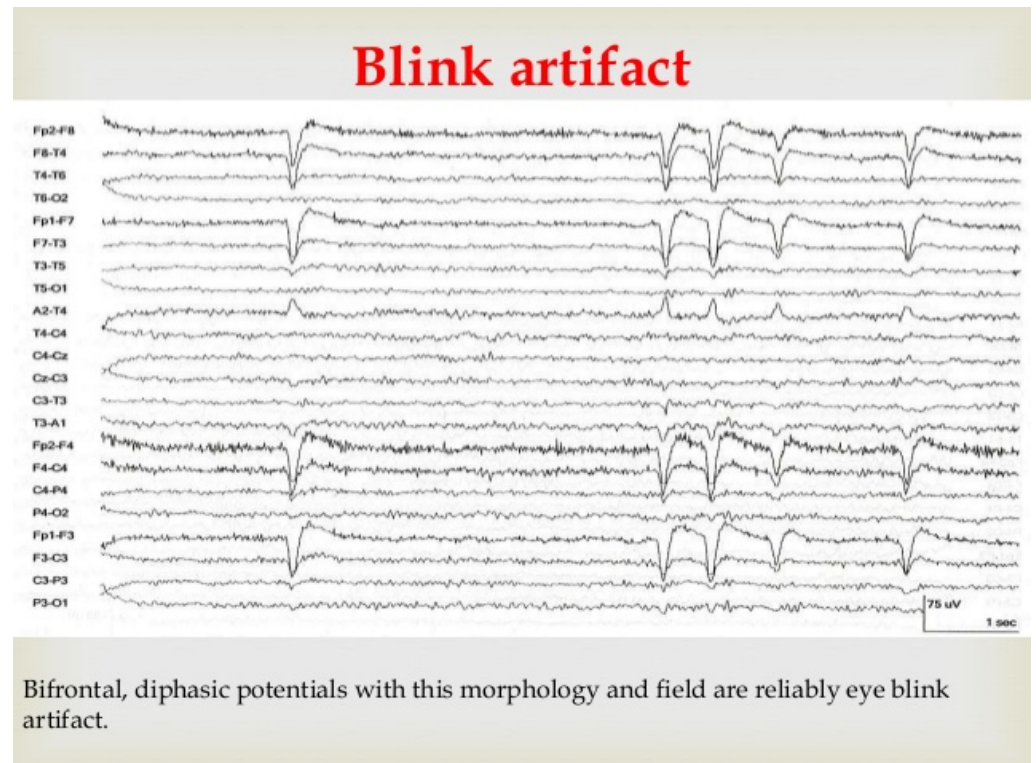
# Artifact correction = ICA

**Independent Components Analysis (ICA)** is a statistical technique for separating a complex signal (like EEG) into statistically independent subcomponents.

As you can imagine, ICA is a very powerful analysis tool. If you have reason to believe that your complex signal is a mixture of separate subcomponents (like EEG), you can use it to try to tease apart those subcomponents.

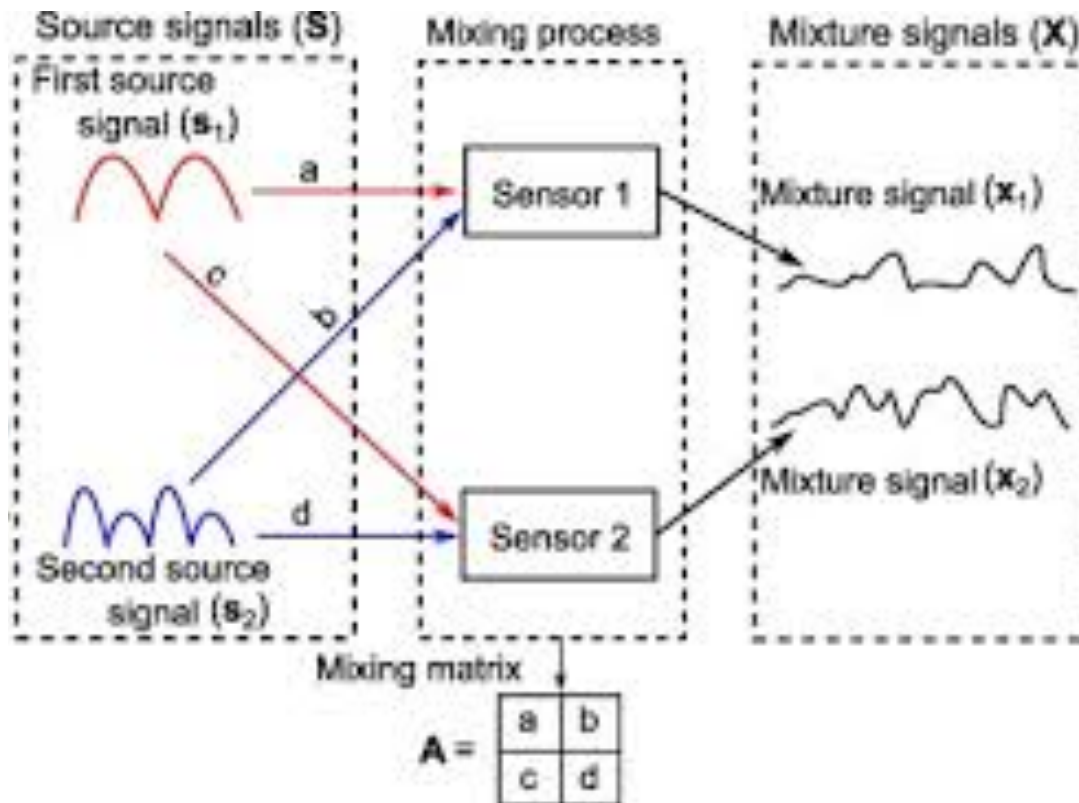
One rather straightforward, and also valuable, use of ICA is to separate out the subcomponents of the EEG signal that come from artifacts like blinks.

**The big idea** is that if you can identify that subcomponent, you can then **subtract** that subcomponent from the signal and have **artifact-free EEG**!



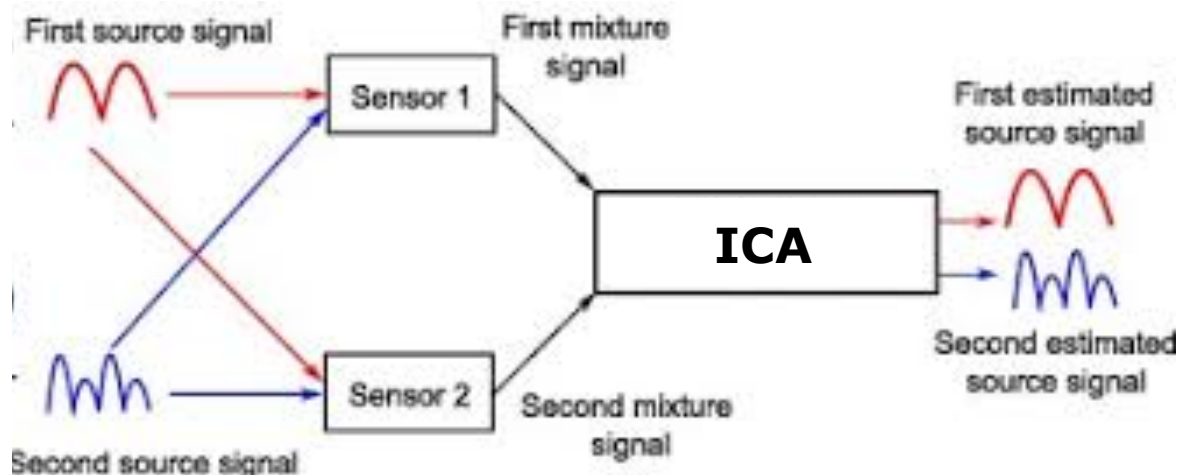


# The logic of ICA



The first step is to remember that the signal at any given electrode is a **mixture** of independent sources.

Every electrode will contain activity from all sources... just in different proportions.



ICA is a mathematical procedure for decomposing the mixtures in all electrodes back into statistically independent component sources. The hope is that these match the actual component sources!

# Artifact correction = ICA

Here is a visual representation of the ICA process.

The raw EEG has a blink in it, as well as some muscle artifact at the temporal electrodes.

ICA decomposes the signal into a number of independent subcomponents. This subcomponents have a time-course and a scalp distribution.

You can then identify the components that capture the blink and the muscle artifact, and recompose the signal using all of the **other** components - yielding cleaner EEG!

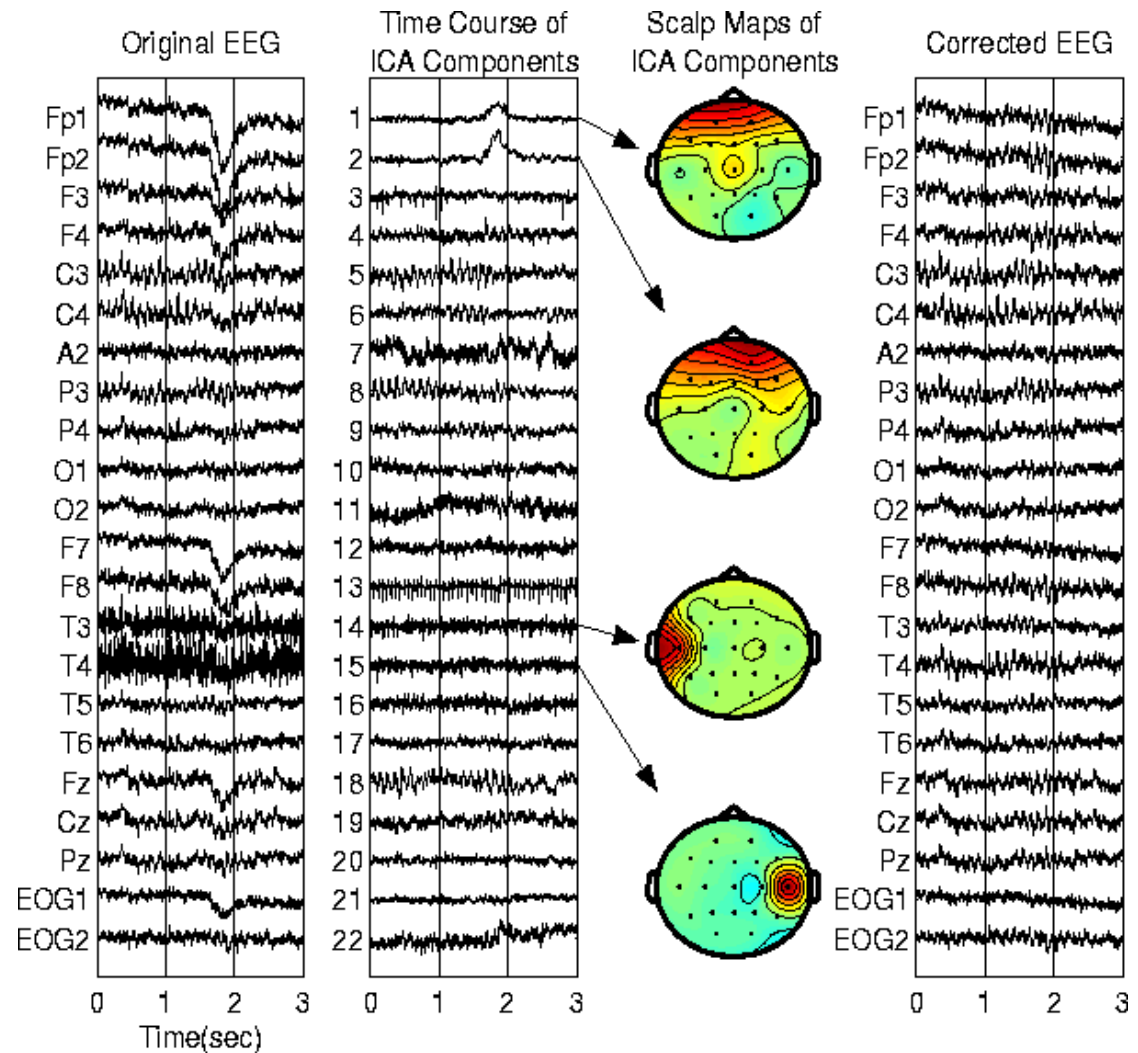


Image from: [https://sccn.ucsd.edu/~jung/Site/EEG\\_artifact\\_removal.html](https://sccn.ucsd.edu/~jung/Site/EEG_artifact_removal.html)

# Artifact correction = ICA

Learning the basic patterns of different types of artifacts:

<https://labeling.ucsd.edu/tutorial/labels>

Practice adding your own labels (with answers):

<https://labeling.ucsd.edu/tutorial/practice>



# Makoto's pipeline

The group at the Swartz center (the group behind EEGLAB) have developed (and are actively developing) a number of methods for using ICA in EEG research. They use it as a primary analysis technique, not just for artifact correction.

I highly recommend exploring the tutorials and tools that they have created if you are interested in ICA.

I also recommend checking out Makoto Miyakoshi's preprocessing pipeline for lots of tips and tricks for performing ICA and data cleaning: [https://scn.ucsd.edu/wiki/Makoto%27s\\_preprocessing\\_pipeline](https://scn.ucsd.edu/wiki/Makoto%27s_preprocessing_pipeline).

# Table of contents

1. Introduction: The big picture
2. Fundamentals
3. Hands-on training and best practices
4. **The ERP processing pipeline in EEGLAB + ERPLAB**
  1. Load the data.
  2. Filter the data (high-pass, possibly low-pass).
  3. ICA for artifact correction (optional).
  4. Re-reference the data.
  5. Add channel locations.
  6. Epoch the data.
  7. Artifact detection and rejection.
  8. **Average the epochs to create subject ERPs.**
  9. Average the subject ERPs to create a grand average ERP.
  10. Plotting: waveforms, topoplots, difference waves
  11. Measure amplitudes and latencies.
  12. Run statistical tests.
5. Creating a script for EEGLAB + ERPLAB
6. Creating a script for Fieldtrip

Luck 1

Luck 2

Luck 5

Luck 6, 7, 8

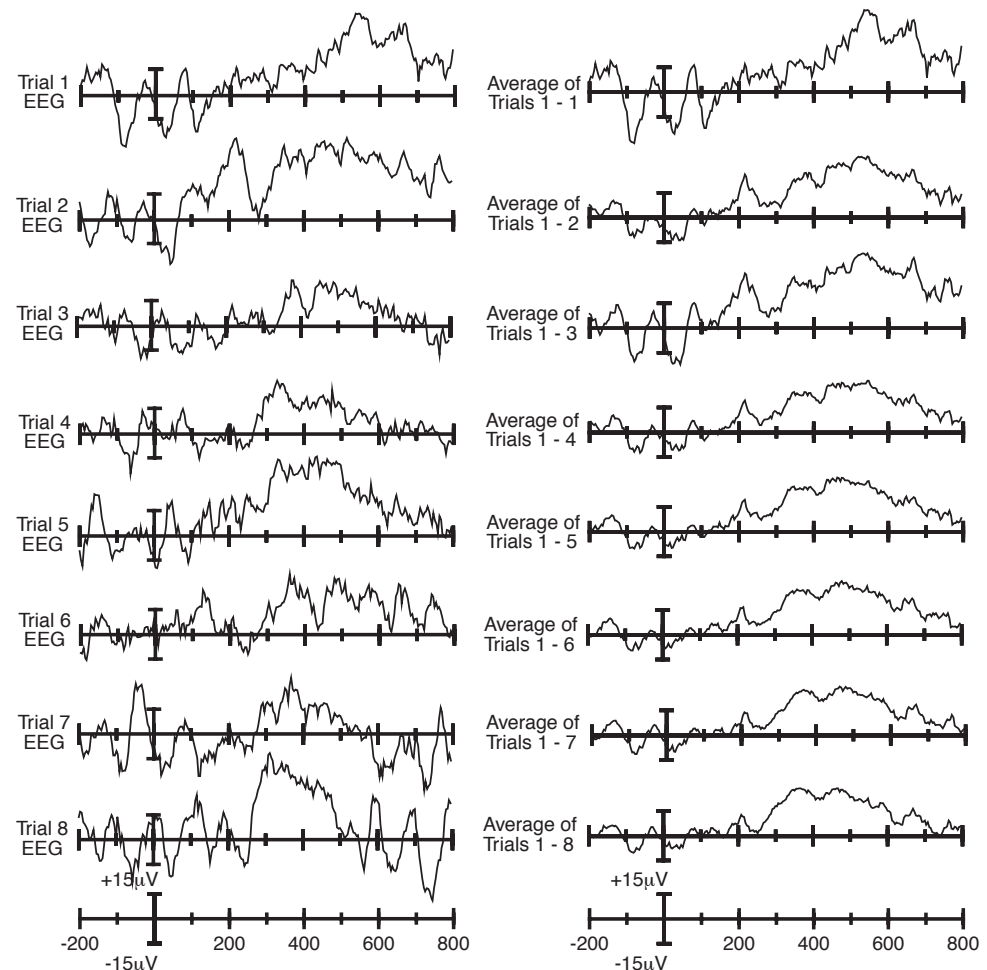
# Creating an ERP is just averaging over trials

Recall from our first class that an ERP is just an average over the trials. This procedure maintains time-locked and phase-locked activity, but diminishes all other activity. The assumption here is that cognitive activity related to the stimulus will be time-locked and phase-locked, and noise (activity we don't care about) will be randomly distributed so that it averages to 0.

This is figure 4.1 from Luck's textbook, taken from lecture 8 at the ERPLAB website.

It shows 8 trials, and the consequences of averaging them (in a cascade fashion).

You can see that the averaging diminishes the parts of the signal that are not identical across trials, slowly revealing a positive wave from 300-600ms.



# Create subject ERP in ERPLAB

We are now ready to create ERPs for each subject.

ERPLAB makes this incredibly simple. There is literally a command called **compute averaged ERP**.

Here is the tutorial link: <https://github.com/lucklab/erplab/wiki/Creating-Averaged-ERPs:-Tutorial>

This command gives you the option (set by default) to exclude epochs that have been marked as artifacts.

This command will create an ERP for every bin in the data set.

# Table of contents

1. Introduction: The big picture	Luck 1
2. Fundamentals	Luck 2
3. Hands-on training and best practices	Luck 5
4. The ERP processing pipeline in EEGLAB + ERPLAB	Luck 6, 7, 8
1. Load the data.	
2. Filter the data (high-pass, possibly low-pass).	
3. ICA for artifact correction (optional).	
4. Re-reference the data.	
5. Add channel locations.	
6. Epoch the data.	
7. Artifact detection and rejection.	
8. Average the epochs to create subject ERPs.	
9. Average the subject ERPs to create a grand average ERP.	
10. Plotting: waveforms, topoplots, difference waves	
11. Measure amplitudes and latencies.	
12. Run statistical tests.	
5. Creating a script for EEGLAB + ERPLAB	
6. Creating a script for Fieldtrip	

# Create grand average ERPs in ERPLAB

Grand average ERPs are created by averaging over all of the subject ERPs.

They have exactly the same character as subject ERPs — they maintain activity that is time-locked and phase-locked, and diminish all other activity.

Here is the link for computing grand averaged ERPs: [https://github.com/lucklab/erplab/wiki/Averaging-Across-ERPSETS-\(Creating-Grand-Averages\)](https://github.com/lucklab/erplab/wiki/Averaging-Across-ERPSETS-(Creating-Grand-Averages))

# Table of contents

1. Introduction: The big picture
2. Fundamentals
3. Hands-on training and best practices
4. **The ERP processing pipeline in EEGLAB + ERPLAB**
  1. Load the data.
  2. Filter the data (high-pass, possibly low-pass).
  3. ICA for artifact correction (optional).
  4. Re-reference the data.
  5. Add channel locations.
  6. Epoch the data.
  7. Artifact detection and rejection.
  8. Average the epochs to create subject ERPs.
  9. Average the subject ERPs to create a grand average ERP.
  10. **Plotting: waveforms, topoplots, difference waves**
  11. Measure amplitudes and latencies.
  12. Run statistical tests.
5. Creating a script for EEGLAB + ERPLAB
6. Creating a script for Fieldtrip

Luck 1

Luck 2

Luck 5

Luck 6, 7, 8

# Two types of plots: waveforms and scalp maps

**Waveform:** This is sometimes called a timecourse plot, or just an ERP. It is a plot of voltage amplitude over time.

ERPs are ultimately three dimensional: voltage x time x space. Waveform plots emphasize amplitude and time, but sacrifice space.

This is a single electrode, so we have no information about how the ERP is distributed across the scalp.

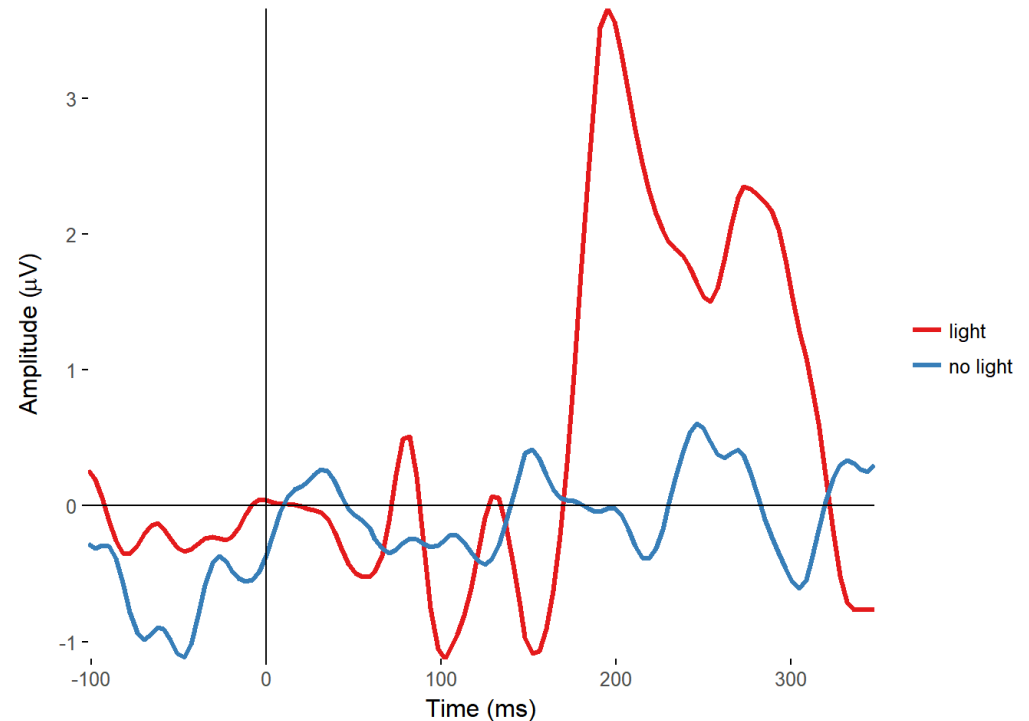


Image from: <http://www.mattcraddock.com/blog/2017/09/05/eegutils-an-r-package-for-eeg/> (He created code to make ERP plots in R, which as we will see, are much prettier than the ones created by ERPLAB.)



# Can we add space to waveforms?

The waveform plot for a single electrode obviously has no space information. But what if we plotted all of the electrodes? This is called a topographically arranged waveform plot:

This does give us space information, but it is difficult to take it all in at once.

One problem is that the individual waveforms become small and hard to see.

Another problem is that it is hard for humans to take in this much information at once.

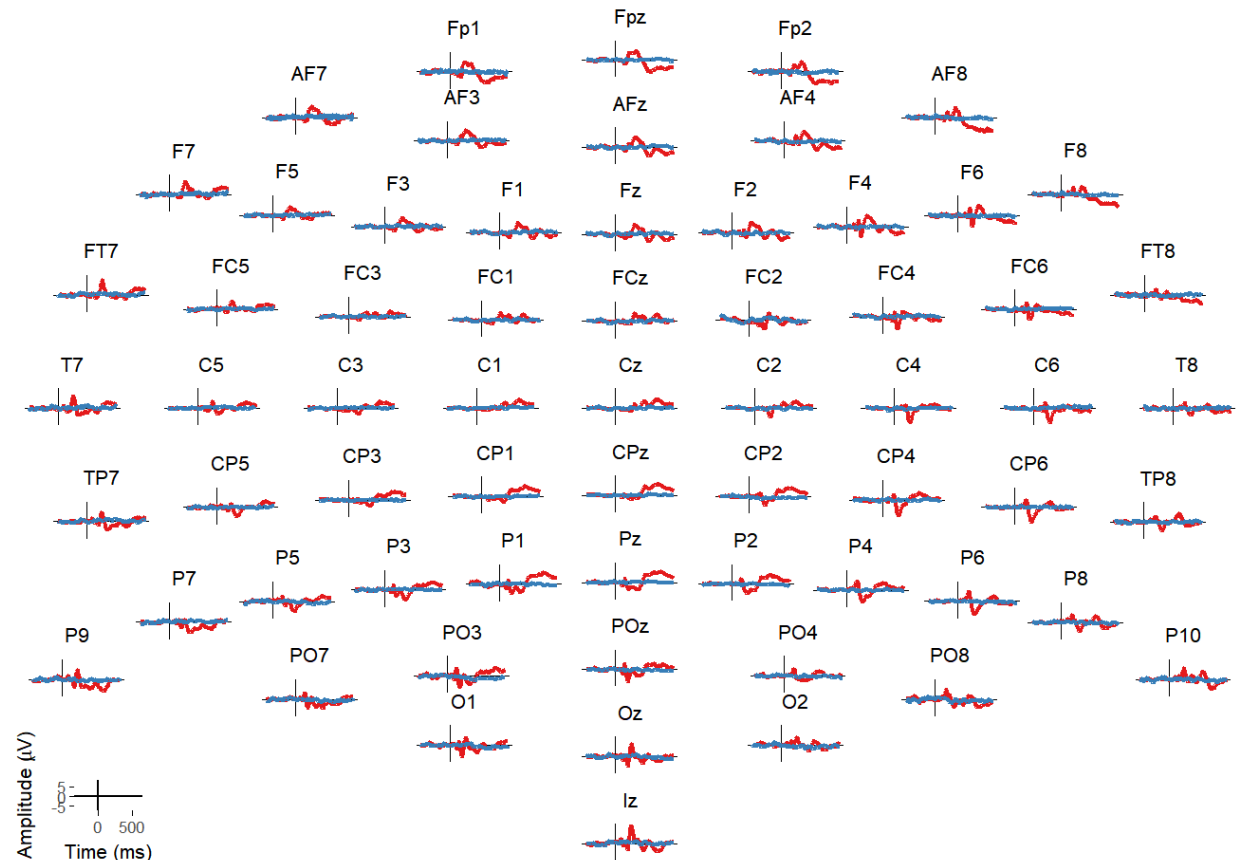


Image from: <http://www.mattcraddock.com/blog/2017/09/05/eegutils-an-r-package-for-eeg/>

# Two types of plots: waveforms and scalp maps

**Topoplot:** This is also called a (topographic) scalp map, or an isovoltage map. It is a plot of amplitude (or a difference in amplitude between two conditions) at a single moment in time, or the mean of a time window, across all electrodes.

Topoplots emphasize amplitude and space, but sacrifice time.

One thing to note is we only have measurements at each electrode location. We have no information for the space between the electrodes. So the plotting algorithm makes an educated guess called interpolation.

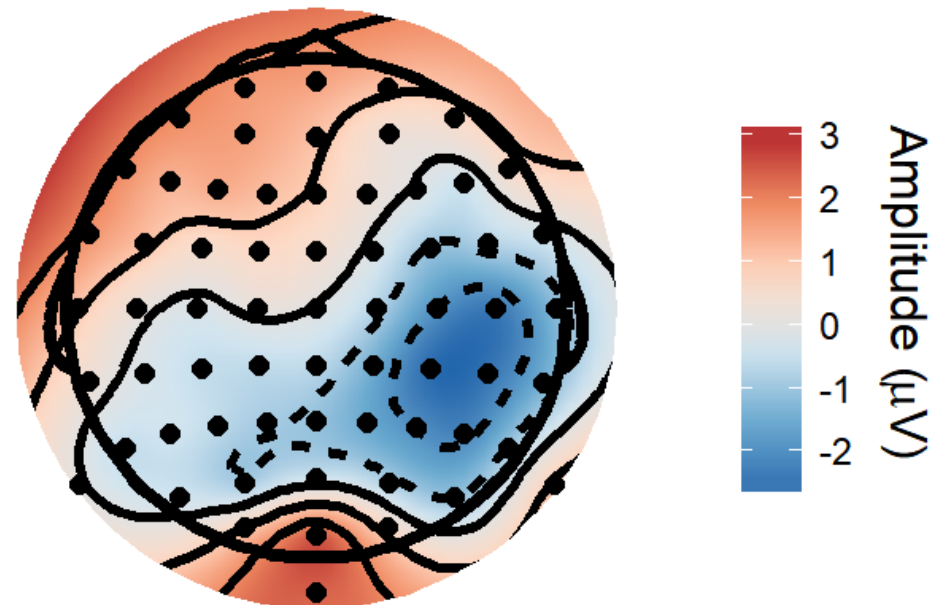
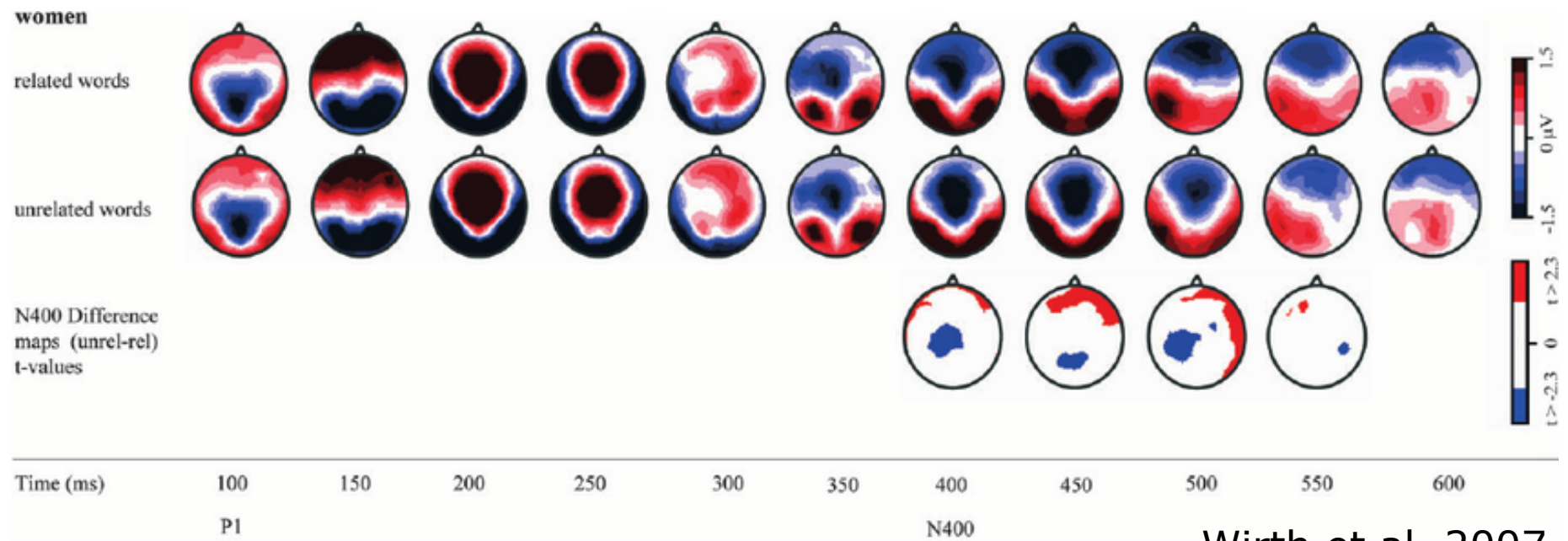


Image from: <http://www.mattcraddock.com/blog/2017/09/05/eegutils-an-r-package-for-eeg/> (He created code to make ERP plots in R, which as we will see, are much prettier than the ones created by ERPLAB.)

# Can we add time to topoplots?

Yes, we can create a time series of topoplots. This plot shows the voltage for one sample every 50 ms in an N400 experiment.



Wirth et al. 2007

This is a useful plot. You will see plots like this often in papers. But you can see that quite a bit of time information is lost. In this case, we lose the 50ms between each snapshot. Capturing all of the time information would be unwieldy.

There is also the option of creating a movie of the voltage over time. But this only works on devices, not in papers.

# Making plots with ERPLAB

ERPLAB has a GUI for generating both waveforms and topoplots. It really makes it easy to generate useful plots.

**Waveforms tutorial:** <https://github.com/lucklab/erplab/wiki/Plotting-Averaged-ERP-Waveforms:-Tutorial>

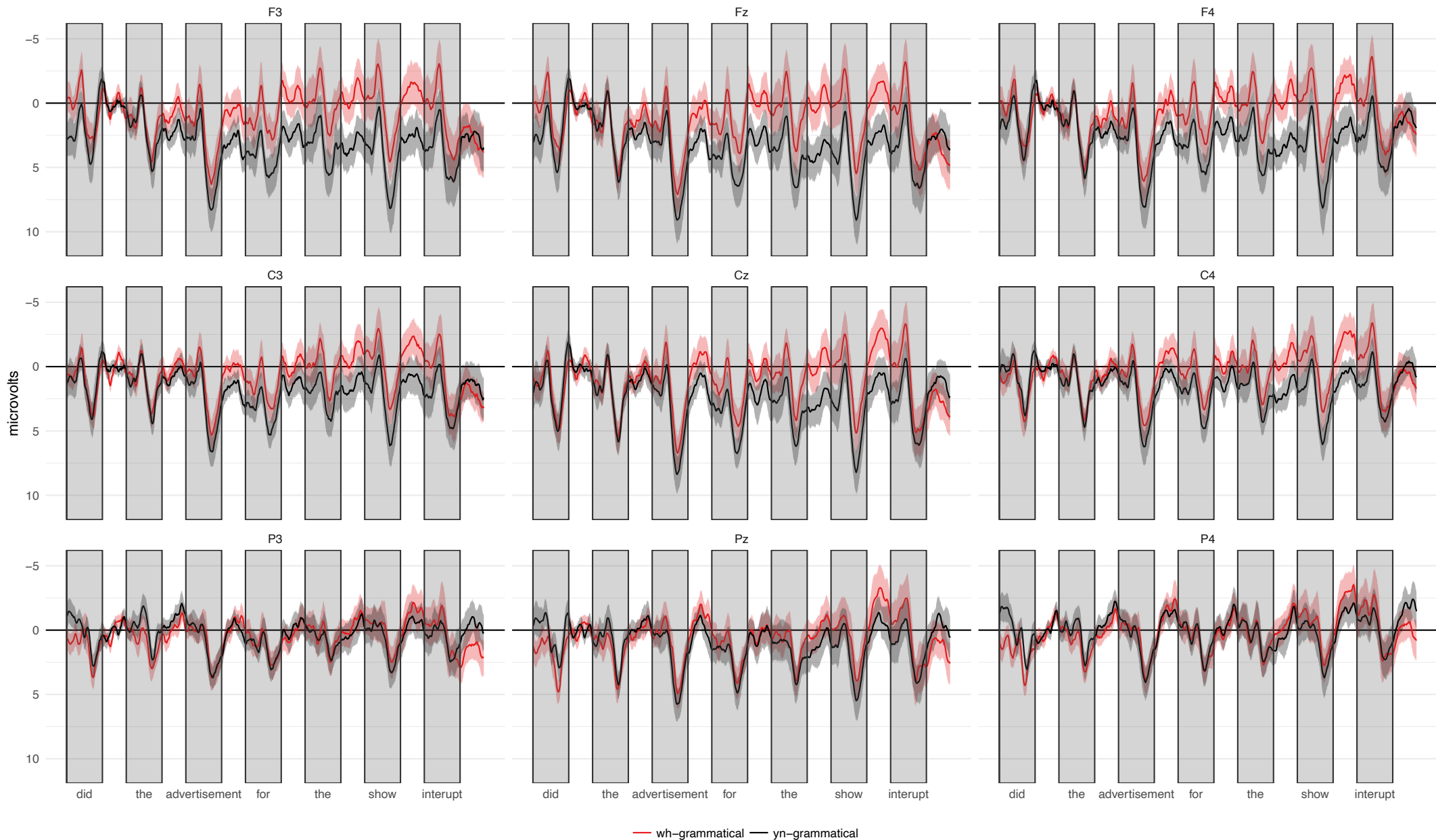
**Waveforms manual:** <https://github.com/lucklab/erplab/wiki/Plotting-ERP-Waveforms>

**Topoplots:** <https://github.com/lucklab/erplab/wiki/Topographic-Mapping>

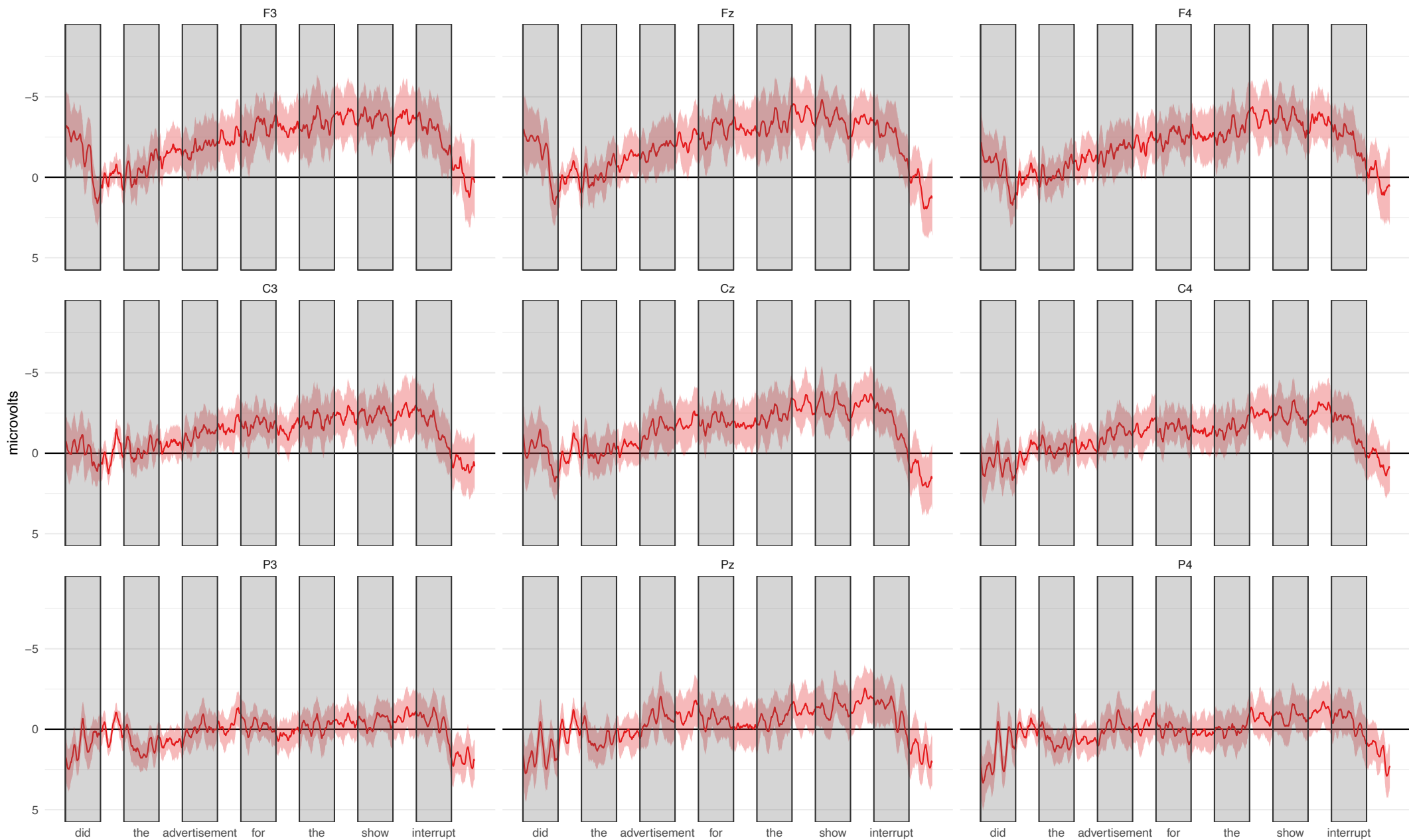
The ERPLAB GUI for creating plots is a really terrific tool. However, the plots that it generates are merely functional... they are useful for exploring your data, and possibly even for sharing it with labmates. But the plots are not publication quality. You should not use them in papers, or even posters.

If you want to make publication quality plots, you need to move beyond the ERPLAB GUI. One option is to learn how to use the Matlab plotting routines directly. Personally, I don't think Matlab plots are all that easy to use, or all that pretty. Another option is to use R or Python to generate plots. I personally prefer R. We will discuss publication quality plots later in the course.

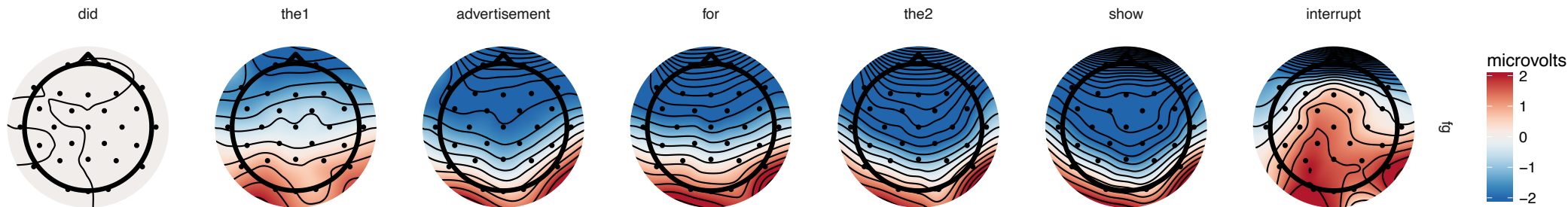
# Waveform with individual conditions



# Difference waves



# Difference topoplots



Difference waves can also be plotted in topoplots. The plot above is the average amplitude of the difference wave in the 300-500ms window for each word in the sentence.

Luck recommends publishing both individual condition waveforms and difference waves, and I agree. It provides all possible information.

For topoplots, I personally tend to only present differences (differences in mean amplitude between two conditions in a given time window). I find it easier to interpret difference topoplots than individual condition topoplots (because they are not plotted on top of each, and require you to compare the shade of the color)



# Creating difference waves in ERPLAB

To create difference waves, you have to tell ERPLAB to subtract one ERP from another.

To do this, we need to use something called bin operations — we create a new bin in ERPLAB by combining existing bins in different ways. In this case, we will subtract one bin from another.

Here is the link for bin operations: <https://github.com/lucklab/erplab/wiki/ERP-Bin-Operations>



# Table of contents

1.	Introduction: The big picture	Luck 1
2.	Fundamentals	Luck 2
3.	Hands-on training and best practices	Luck 5
4.	The ERP processing pipeline in EEGLAB + ERPLAB	Luck 6, 7, 8
1.	Load the data.	
2.	Filter the data (high-pass, possibly low-pass).	
3.	ICA for artifact correction (optional).	
4.	Re-reference the data.	
5.	Add channel locations.	
6.	Epoch the data.	
7.	Artifact detection and rejection.	
8.	Average the epochs to create subject ERPs.	
9.	Average the subject ERPs to create a grand average ERP.	
10.	Plotting: waveforms, topoplots, difference waves	
11.	Measure amplitudes and latencies.	
12.	Run statistical tests.	
5.	Creating a script for EEGLAB + ERPLAB	
6.	Creating a script for Fieldtrip	

# Types of measurements: peak amplitude

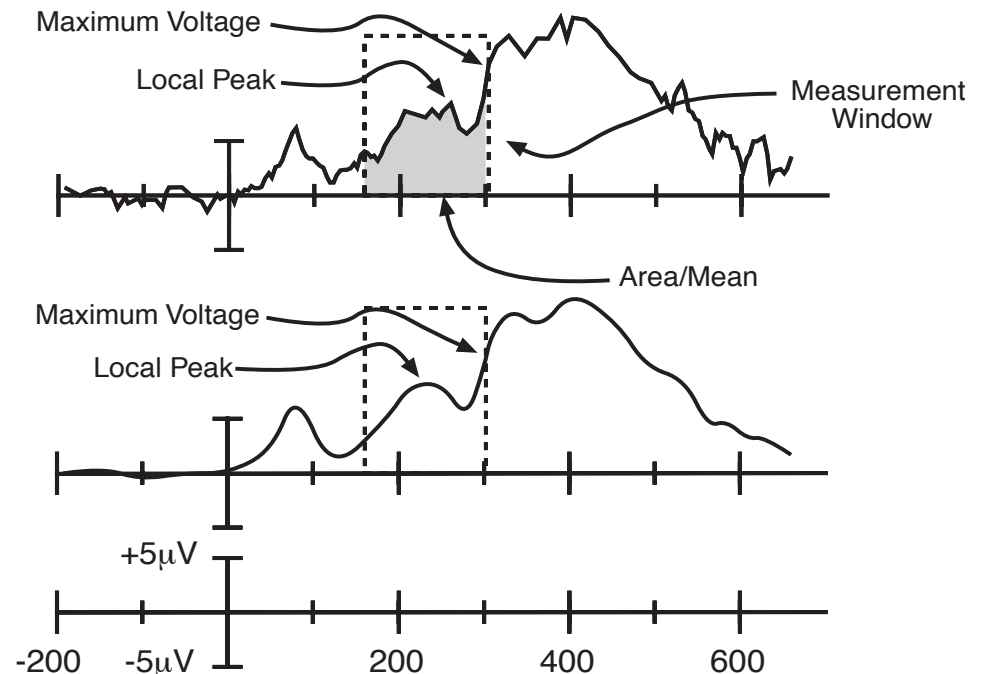
The first thing to keep in mind is that most (probably all) measurements that you make will be within a specified **time window**, such as 150-300 ms or 300-500ms. So every measurement we make is within a time window.

**peak amplitude:** The maximum amplitude within a time window.

**local peak amplitude:** The maximum amplitude within a time window that has lower amplitude measurements on either side.

Figure 9.1 from Luck, taken from lecture 10 available on the ERPLAB website.

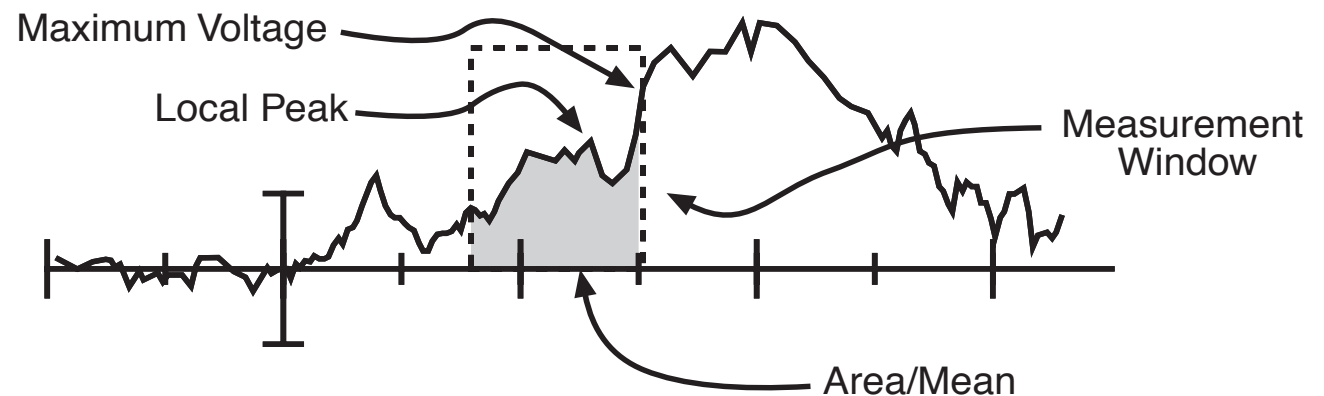
It shows the difference between peak amplitude and local peak amplitude (top), and also the effect of low-pass filtering (bottom), which moves the local peak a bit.



# Types of measurements: mean amplitude

**mean amplitude:** The mean of the amplitude values in the time window.

Figure 9.1 from Luck, taken from lecture 10 available on the ERPLAB website.



Mean amplitude is often superior to peak amplitude measures for a number of reasons that Luck describes in chapter 9, such as:

There is nothing special about peaks. They are not the same as components.

Components exist through time. Mean amplitude reflects that.

Peaks are sensitive to high-frequency noise. It causes systematic increases in peak amplitude, and can cause shifts in the location (latency) of the peak that are cognitively implausible. Mean amplitude is not affected this way.

# Types of measurements: signed area

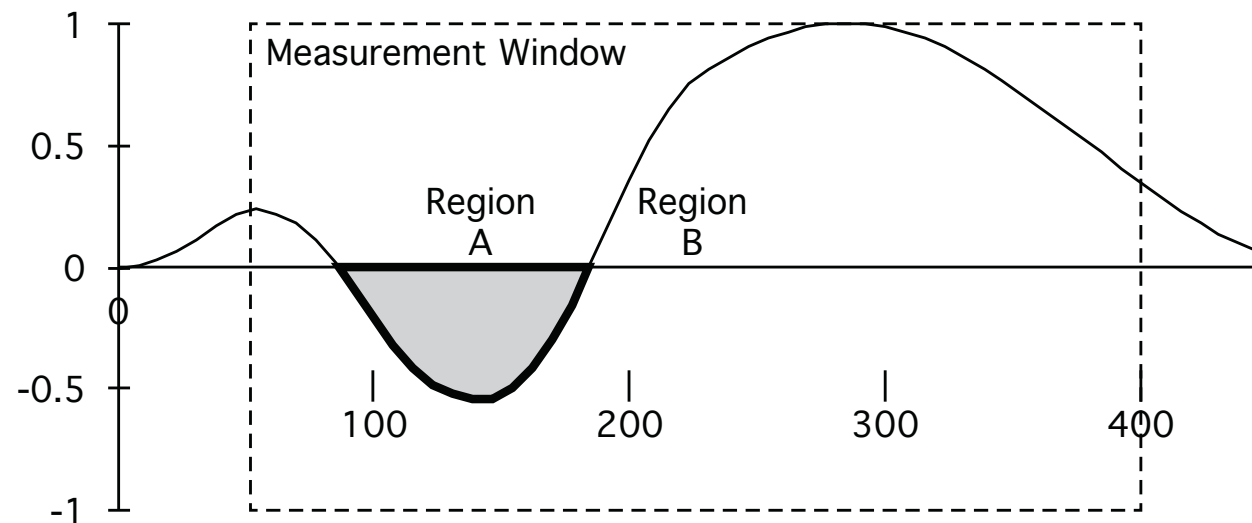
Mean amplitude requires you to know the time-window of interest. You can usually determine this from previous research (you shouldn't determine this post-hoc from the results, because that would bias your results).

But if you don't know the right time window from previous research, you could use a signed area measure instead of a mean amplitude measure.

**signed area:** The area under the curve relative to the zero amplitude line.

Figure 9.4 from Luck, taken from lecture 10 available on the ERPLAB website.

Notice that signed area allows us to specify a very large time window and identify the negative wave of interest... but it complicates the statistics because of bias (see Luck).



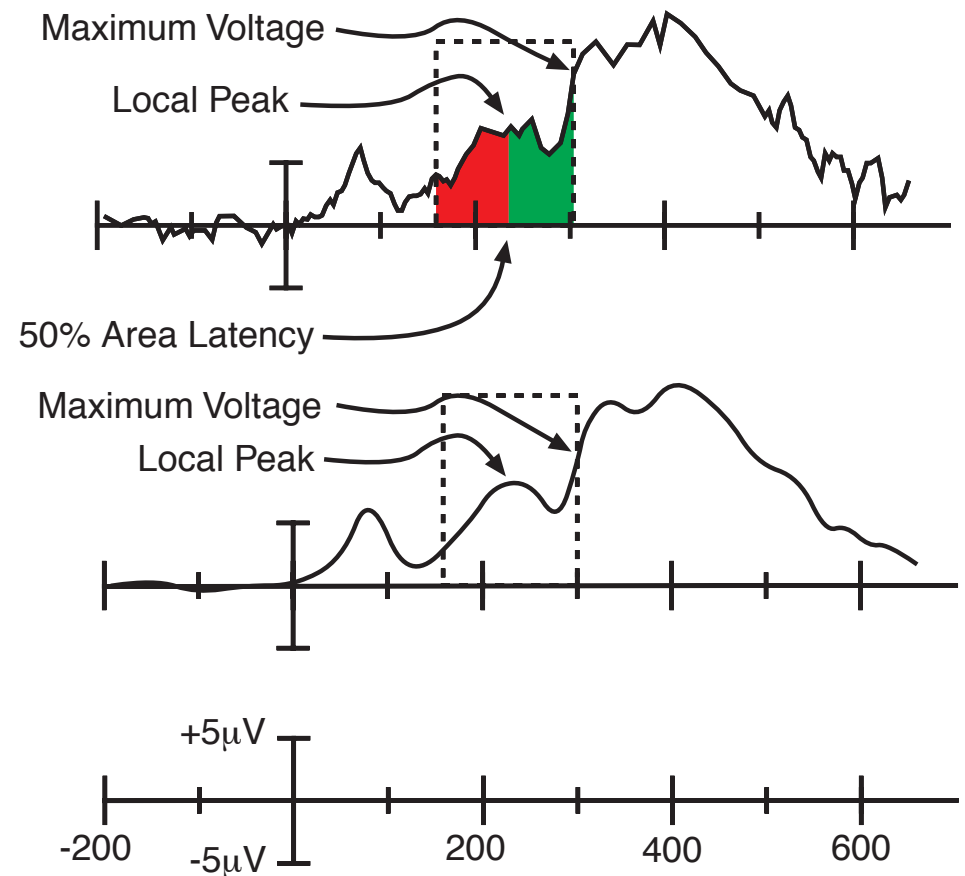
# Types of measurements: peak latency

**peak latency:** The latency of a peak in a time window; either simple or local.

Latency is all about measuring the timing of a process. Since peaks are easy to see, it is tempting to measure their latency.

But peak latency has all of the shortcomings of peak amplitude - the maximum may not be a true peak, peaks are distorted by noise, etc.

Because of this, Luck recommends using area latency (specifically 50% area).

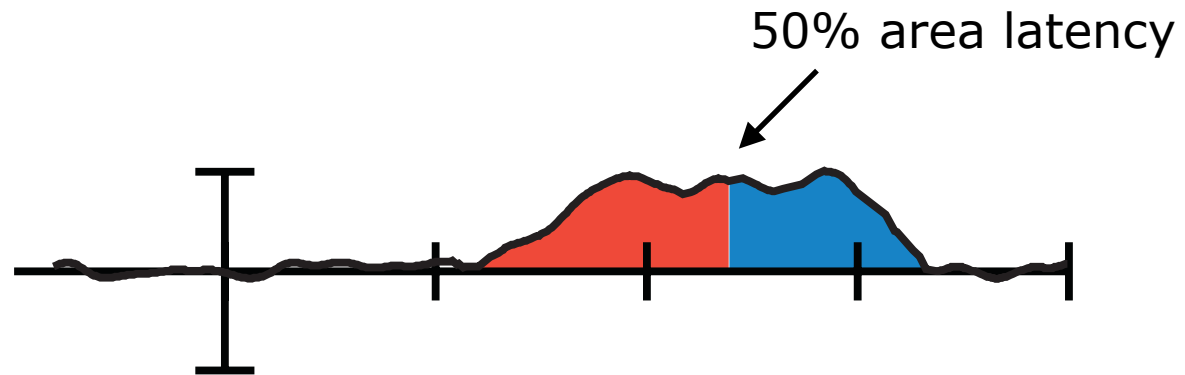


A modified version of Luck figure 9.1, taken from lecture 10.

# Types of measurements: fractional area latency

## **fractional area latency:**

This is an estimate for midpoint latency. Calculate the area under a wave, then identify the time point that divides the area into the desired fraction, such as 50% for a midpoint.



This is from Luck figure 9.5 and lecture 10.

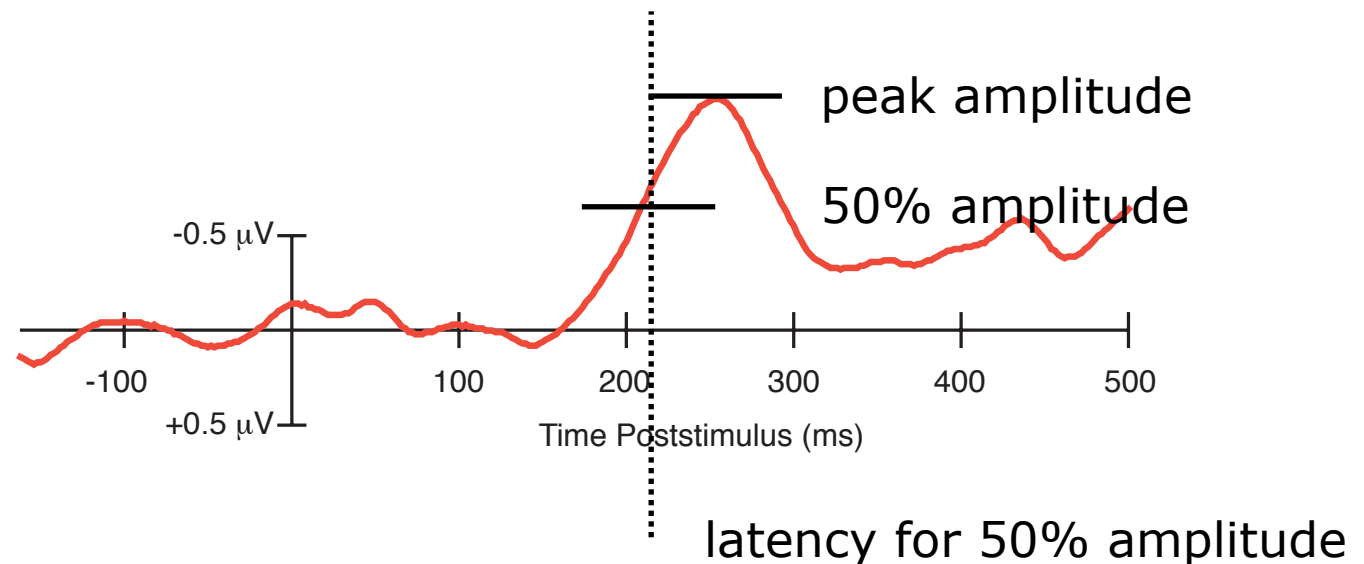
Fractional area measures avoid the problems with peak latency - they are not influenced by noise, or the method of identifying peaks. Luck recommends fractional area for most latency measures.

# Types of measurements: fractional peak latency

## **fractional peak latency:**

This is an estimate for onset latency. First calculate the peak amplitude. Then calculate a fraction of the peak amplitude (such as 50%). Finally, identify the time point where the fraction of the peak amplitude is reached.

This figure is from Luck's lecture 10.



This is Luck's preferred method for measuring onset latency. Onset latency might be relevant if you are interested in estimating an upper bound for how long it takes the brain to distinguish two conditions. Onset latency might also be more relevant to your theory of timing than midpoint latency if you believe that the crucial question is when a process begins rather than when it hits its midpoint or peak.



# The ERPLAB measurement tool

ERPLAB has a measurement tool for measuring amplitudes and latencies. It will export these as text files so that you can import them into R to run statistics.

Here is the link for a tutorial on using the measurement tool in ERPLAB:  
<https://github.com/lucklab/erplab/wiki/Measuring-amplitudes-and-latencies-with-the-ERP-Measurement-Tool:-Tutorial>

Here is the manual page: <https://github.com/lucklab/erplab/wiki/ERP-Measurement-Tool>

Note that the measurement tool comes with a viewer that allows you to see how the measurement is being made for each subject ERP!

# Table of contents

1.	Introduction: The big picture	Luck 1
2.	Fundamentals	Luck 2
3.	Hands-on training and best practices	Luck 5
4.	The ERP processing pipeline in EEGLAB + ERPLAB	Luck 6, 7, 8
1.	Load the data.	
2.	Filter the data (high-pass, possibly low-pass).	
3.	ICA for artifact correction (optional).	
4.	Re-reference the data.	
5.	Add channel locations.	
6.	Epoch the data.	
7.	Artifact detection and rejection.	
8.	Average the epochs to create subject ERPs.	
9.	Average the subject ERPs to create a grand average ERP.	
10.	Plotting: waveforms, topoplots, difference waves	
11.	Measure amplitudes and latencies.	
12.	Run statistical tests.	
5.	Creating a script for EEGLAB + ERPLAB	
6.	Creating a script for Fieldtrip	

# Roadmap for statistical analysis in EEG

The driving issue behind (null hypothesis) statistical analysis for EEG data is the **multiple comparisons problem**. Much of what follows discusses that.

Once the multiple comparisons problem is in focus, the choice we need to make is how we want to address it:

**ANOVAs** can be used if you (i) know the time window of the effect, and (ii) the scalp distribution of the effect. They solve the multiple comparison problem by reducing the test to one (omnibus) ANOVA. There is a script for this on the website.

**Mass univariate permutation tests** can be used if you want to test all time points and all channels, and want strong control over the familywise error rate. The only downside is that you lose some statistical power.

**Cluster-based mass univariate permutation tests** can be used if you want to test all time points and all channels, and are ok with restricting your view to effects that are broadly distributed in time and/or space. This increases statistical power, but provides only weak control over the familywise error rate.

# Two approaches to NHST

It turns out that there are two major approaches to NHST. They are very similar in mathematical appearance, so it is easy to think that they are identical. But they differ philosophically (and in some details), so it is important to keep them separated.

**Ronald Fisher** was the first person to try to wrangle the growing field of statistics into a unified approach to hypothesis testing. His NHST was the first attempt, and may still be the closest to the way scientists think about NHST. We'll start with the **Fisher approach**.

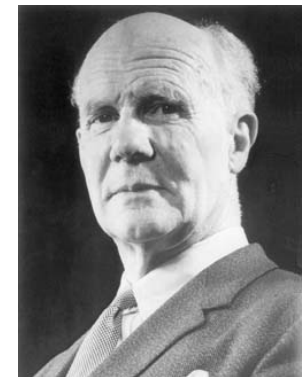


Ronald A. Fisher  
(1890-1962)

**Neyman** and **Pearson** were fans of Fisher's work, but thought there were some deficiencies in his approach. So they tried to rectify that. It turns out that they simply had a different conception of probability and hypothesis testing. We'll talk about the Neyman-Pearson approach second.



Jerzy Neyman  
(1894-1981)



Egon Pearson  
(1895-1980)

# Fisher's NHST

Under Fisher's NHST, there is only one hypothesis under consideration. Perhaps ironically, it is the most uninteresting hypothesis you could consider. It is called the **null hypothesis**, or  $H_0$ .

**H<sub>0</sub>:** The **null hypothesis**. This states that there is no effect in your data (e.g., no difference between conditions, no interaction term, etc).

For Fisher's NHST, the goal of an experiment is to **disprove the null hypothesis**.



"Every experiment may be said to exist only in order to give the facts a chance of disproving the null hypothesis." - Fisher (1966)

To do this, Fisher's NHST calculates the probability of **the observed data** under **the assumption that the null hypothesis is true**, or  $p(\text{data} | \text{null hypothesis})$ .

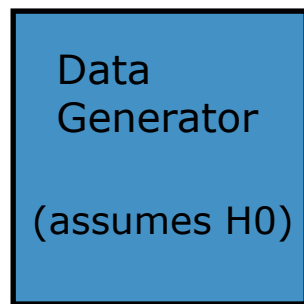
This leads to **Fisher's disjunction**:

If  $p(\text{data} | \text{null hypothesis})$ , called the p-value, is low, then you can conclude either: (i) the null hypothesis is incorrect, or (ii) a rare event occurred.

# Fisher's NHST logic, stated a different way

There are two steps to a statistical test under Fisher's NHST approach:

**Step 1:** Calculate  $P(\text{data} \mid \text{null hypothesis})$



data1  
data2  
data3  
...

One way to think about this is that you are creating a data generating device that assumes the null hypothesis, and generates **all possible data sets**.

Then you use the distribution of generated data to calculate the probability of the observed data

$$P(\text{data} \mid H_0) = \frac{\text{observed data}}{\text{generated data}}$$

**Step 2:** Make an inference about the null hypothesis

For Fisher,  $p(\text{data} \mid H_0)$  is a measure of the strength of evidence against the null hypothesis. If it is low, that is either strong evidence against the null hypothesis, or evidence that something really rare occurred.

# The math underlying NHST

Though the logic is enough to interpret the results that R and lmerTest give us, you may want to study the math that NHST approaches use to generate the reference distribution for the null hypothesis. It will give you the flexibility to run (and even create) your own analyses, and it will help you understand the hypothesis tests at a deeper level.

There are basically three approaches to generating the null reference distribution in NHST. I will review each briefly in the next few slides:

## **1. Randomization methods.**

The basic idea is to take your observed data points, and randomize the condition labels that you attach to them.

## **2. Bootstrap methods.**

The basic idea is to use your sample as a population, and sample from it to generate a (population-based) reference distribution.

## **3. Analytic methods.**

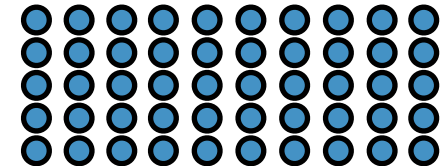
Most people imagine analytic methods when they think of stats. The idea here is that there are test statistics whose distribution is invariant under certain assumptions. We can use these known distributions to calculate p-values analytically (with an equation).



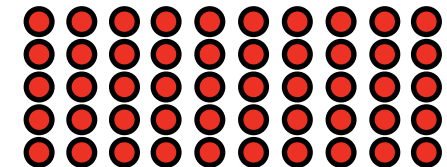
# Randomization Methods

Let's use an example to demonstrate how to generate a reference distribution for the null hypothesis using randomization. Let's focus on two conditions for simplicity:

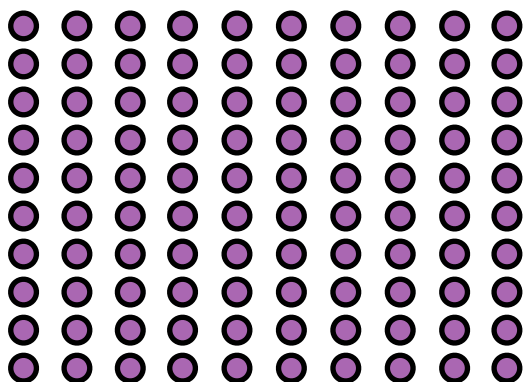
**control:** What do you think that Jack stole \_\_\_?



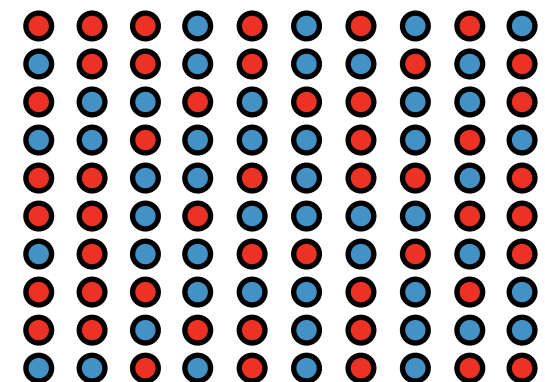
**target:** \*What do you wonder whether Jack stole \_\_\_?



Here is the critical insight of randomization tests: Even though I have **labeled** these observations **control** and **target**, under the null hypothesis they are all just from the same label, **null**. So, this assignment of labels is arbitrary under the null hypothesis. And if the assignment is arbitrary, then I should be able to randomly re-arrange the labels.

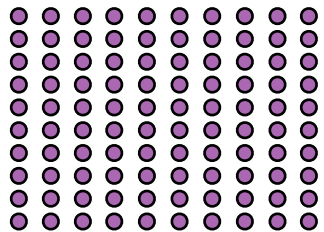


Randomly assign labels to these points because these labels are arbitrary under the null hypothesis.

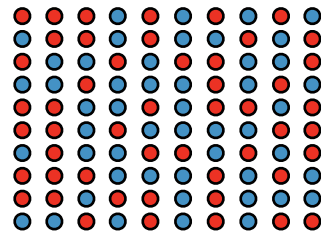


# Randomization: generating the distribution

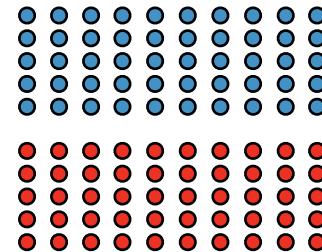
Start with the full data set



Randomly assign labels



Calculate the test statistic

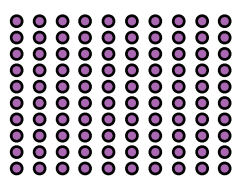


$$\rightarrow \bar{x}_c = 1$$

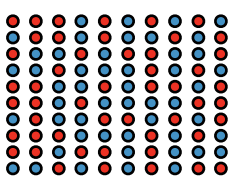
$$\rightarrow \bar{x}_t = .3$$

$$\rightarrow \bar{x}_c - \bar{x}_t = .7$$

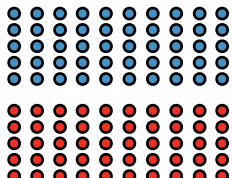
Then we repeat the process. With small samples, we can create every possible combination of labels, and have a complete distribution of possible test statistics. With large samples, this isn't possible, so we collect a large number of randomizations, like 10,000, and approximate the distribution.



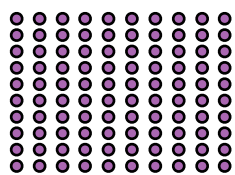
→



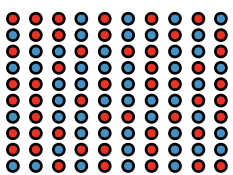
→



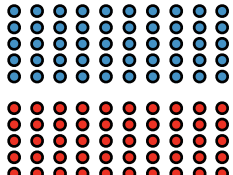
$$\bar{x}_c - \bar{x}_t = -.5$$



→



→



$$\bar{x}_c - \bar{x}_t = .3$$

We then collect all of the test statistics together to form a reference distribution under the null hypothesis.

See [randomization.r](#) for code to do this!

... to completion or 10,000

# Randomization: calculating a p-value

Now that we have a reference distribution, we ask the following question: **What is the probability of obtaining the observed result, or one more extreme, given this reference distribution?**

We say “or one more extreme” for two reasons. First, we can’t just ask about one value because our response scale is continuous (most likely, the probability of one value is 1/the number of values in our distribution). Second, if we have to define a bin, “more extreme” results make sense, because those are also results that would be less likely under the null hypothesis.

If you calculated all possible randomizations, then you can use this formula for p-values:

$$p = \frac{\text{observations equal, or more extreme}}{\text{number of randomizations}}$$

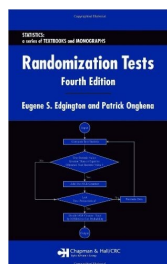
If you randomly sampled the randomizations, then the above will underestimate the true p-value (because your sampled distribution is missing some extreme values). You can correct for this by adding 1 to the numerator and denominator:

$$p = \frac{\text{observations equal, or more extreme} + 1}{\text{randomly sampled randomizations} + 1}$$

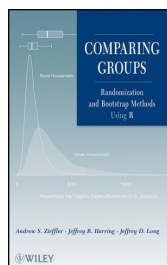
# Randomization: More info

Randomization tests are incredibly powerful and incredibly flexible. I would say that if you want to do pure NHST, without mixed effects, then randomization tests should be your first choice.

Even Fisher admitted that randomization tests should be the gold standard for NHST. But in the 1930s, computers weren't accessible enough to make randomization tests feasible for anything but very small experiments. So he developed analytic methods for larger experiments. But he said that the analytic methods are only valid insofar as they give approximately the same result as randomization methods.



The best reference for randomization tests is Edgington and Onghena (now 2007), **Randomization Tests**. Be warned that it is written like a reference, and not like a textbook. But if you need to know something about randomization tests, it is fantastic.



For a textbook experience, I like Zieffler, Harring, and Long's (2011) **Comparing Groups: Randomization and Bootstrap Methods using R**. It is an introduction to NHST using Randomization and Bootstrap methods, which is a nice idea in the computer age.

# Analytic methods

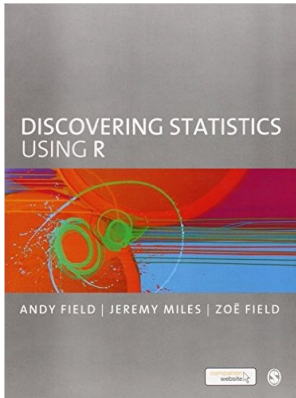
Because randomization and bootstrap methods are so computationally intensive, early 20th century statisticians could not use them. These people were smart. They developed analytic methods that give approximately the same result as randomization and bootstrap methods. And then shared them with the world.

The basic idea of analytic methods is that we need test statistics that have known, or easily calculable, reference distributions. We can't use the mean, because the distribution of the mean will vary based on the experiment (the data type, the design, etc). We need statistics that are relatively invariant, so that we can calculate the distribution once, and use it for every experiment in all of the different areas of science.

There are both **parametric** and **non-parametric** analytic methods, just like there are both parametric and non-parametric bootstrap methods. And there are a ton of different test statistics with different properties that are suited for different experimental situations.

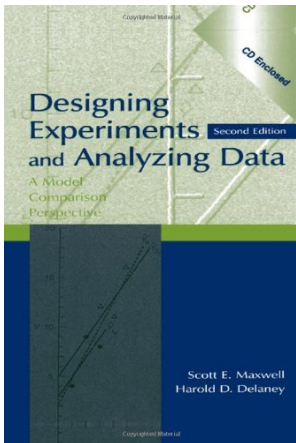
For pedagogical reasons, I am going to focus on the **F statistic**.

# Analytic methods: more information



## **Discovering Statistics Using R Field, Miles, and Field**

This book is a comprehensive introduction to (analytic) statistics, and it is a great introduction to R (and plotting with R). It is very readable (and at times, amusing), and covers all of the things that are covered in fundamental statistics courses.



## **Designing Experiments and Analyzing Data Maxwell and Delaney**

This is probably the best book on the model comparison approach to F-tests there is. It is also a beast of a book. But well worth it if you really want to understand F-tests. There is no R here. This is math.

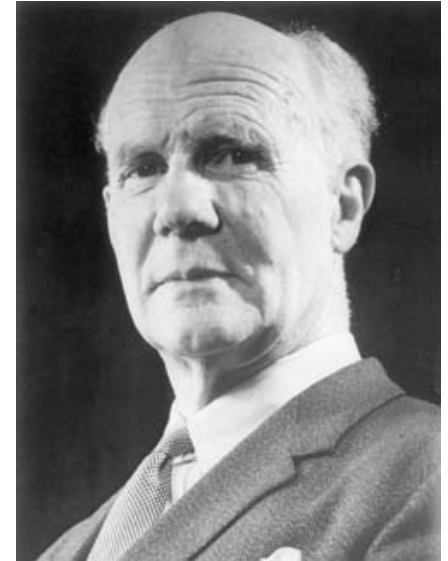
# Neyman-Pearson NHST

Neyman and Pearson were fans of Fisher's work, but they thought there was a logical problem with his approach.

While it is all well and good to say that the p-value is a measure of strength of evidence against the null hypothesis, at some point you have to **make a decision to reject the null hypothesis, or not.**



Jerzy Neyman  
(1894-1981)



Egon Pearson  
(1895-1980)

Fisher himself had suggested that  $p < .05$  was a good criterion for deciding whether to reject the null hypothesis or not.

Neyman and Pearson decided to take this one step further, and really work out what it would mean to base a statistical theory on the idea of decisions to reject the null hypothesis.



# Neyman-Pearson NHST

**Tenet 1:** There are two states of the world: the null hypothesis is either true or false.

**Tenet 2:** You can never know if the null hypothesis is true or false.

This actually follows from the philosophy of science and the problem of induction.

In the absence of certainty about the state of the world, all you can do is make a decision about how to proceed based on the results of your experiment. You can choose to reject the null hypothesis, or you can choose not to reject the null hypothesis.

This sets up four possibilities: two states of the world and two decisions that you could make.

		State of the World	
		$H_0$ True	$H_0$ False
Decision	Reject $H_0$	Type I Error	Correct Action
	Accept $H_0$	Correct Action	Type II Error

# Neyman-Pearson NHST

		State of the World	
		$H_0$ True	$H_0$ False
Decision	Reject $H_0$	Type I Error	Correct Action
	Accept $H_0$	Correct Action	Type II Error

**Type I Error:** This is when the null hypothesis is true, but you mistakenly reject it.

**Type II Error:** This is when the null hypothesis is false, but you mistakenly fail to reject it.

Take a moment to really think about what these two errors are. What do you think about the relative importance of each one?

# Neyman-Pearson NHST

		State of the World	
		$H_0$ True	$H_0$ False
Decision	Reject $H_0$	Type I Error	Correct Action
	Accept $H_0$	Correct Action	Type II Error

Neyman-Pearson, and many others, have suggested that Type I errors are more damaging than Type II errors.

The basic idea is that science is focused on rejecting the null hypothesis, not accepting it. (To publish a paper, you have to reject the null hypothesis.) So a Type I error would mean making a decision (or publishing a result) that misleads science.

Type II errors are also important, but not equally so. Failing to reject the null hypothesis is simply a failure to advance science. It doesn't (necessarily) mislead the way that a Type I error does.

# Neyman-Pearson NHST

**Type I Error:** This is when the null hypothesis is true, but you mistakenly reject it.

If you accept the importance of Type I errors, then you will want to keep the rate of Type I errors as low as possible.

Under the Neyman-Pearson approach, which emphasizes the decision aspect of science, you can control your Type I error rate by **always using the same criterion for your decisions**.

**alpha level / alpha criterion:** This is the criterion that you use to make your decision. By keeping it constant, you keep the number of Type I errors that you will make constant too. For example, if you set your alpha level to .05, then you only decide to reject the null hypothesis if your p-value is less than .05. Similarly, if you set your alpha level to .01, then you only decide to reject the null hypothesis if your p-value is less than .01.

Take a moment to think about how setting an alpha level will control your Type I error rate.

# Neyman-Pearson NHST

There is an important relationship between your alpha level and the number of Type I errors that you will make:

If you apply the same alpha level consistently over the long-run, your Type I error rate will be less than or equal to your alpha level.

## **Here's a thought experiment:**

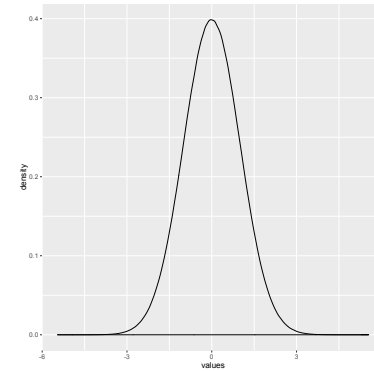
1. Imagine that the null hypothesis is TRUE.
2. Now, imagine that you run an experiment and derive a test statistic.
3. Next, imagine that you run a second experiment and derive a test statistic.
4. And then, imagine that you ran the experiment 10,000 times...
5. This should be familiar. You just derived a reference distribution of the test statistic under the null hypothesis!
6. Now set your alpha level (decision criterion) to .05. Given the distribution you just derived, how often will you derive a p-value less than .05? In short, how often would you make a Type I Error?

We can run this in R. There is code for it in [alpha.demonstration.r](https://alpha.demonstration.r).

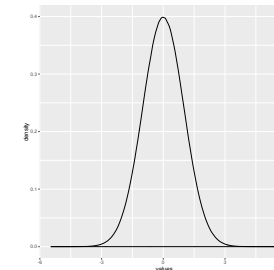
# Graphical version: the alpha level

Here is how the alpha level works:

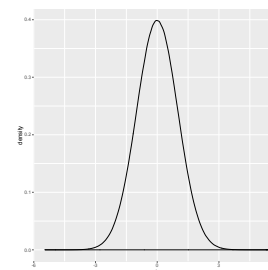
- 1.** Imagine that the null hypothesis is true for your phenomenon.
- 2.** And let's run an experiment testing this difference 10,000 times, saving the statistic each time.
- 3.** The result will be a distribution of real-world test statistics, obtained from experiments where the null hypothesis is true.
- 4.** But also notice that this distribution will be nearly identical to the hypothetical null distribution for your test statistic (because the null hypothesis was true in the real world). This will be important later.



real world  
distribution of stats

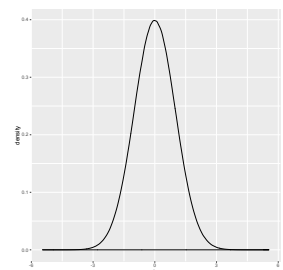


real world  
distribution



null distribution

=



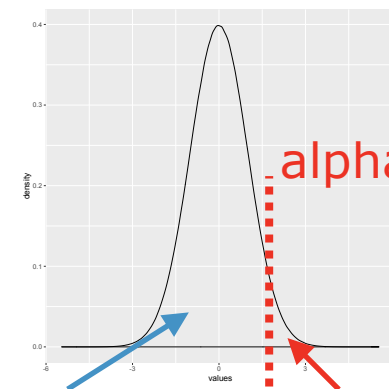
# Graphical version: the alpha level

5. Now let's choose a **threshold** to cut the distribution into two decisions: **non-significant** and **significant**

Remember we call this the **alpha level**.

Also remember that this is a criterion chosen based on the null distribution (because this is a null hypothesis test).

null distribution



accept the null

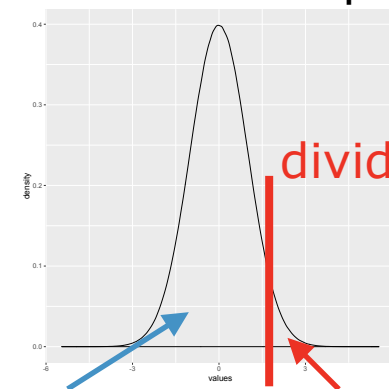
reject the null

6. Now we apply this threshold to each of our 10,000 experiments, one at a time as we run them.

So for each experiment, we can label it as a **correct decision** (accept the null) or a **false positive** (reject the null).

And to make life easier, we can visualize this as a distribution of results, with a dividing line between the two types.

real world experiments



correct decisions

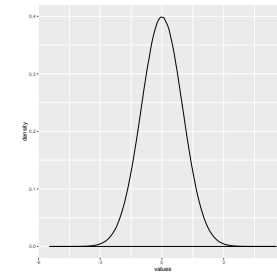
false positives

# Graphical version: the alpha level

7. Now here is the final question. How many false positives happened in our 10,000 experiments?

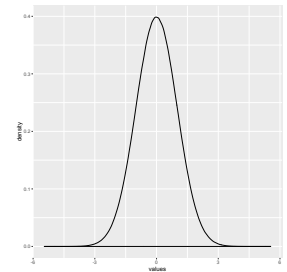
We could count them. But what I want to show you is the consequence of the identity that happened back in step 4.

real world  
distribution



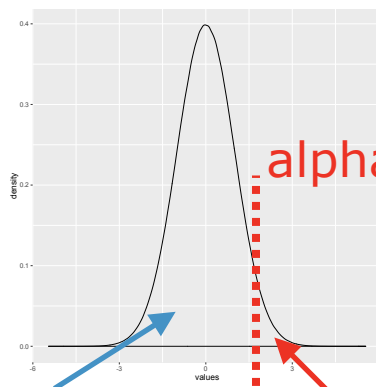
null distribution

=



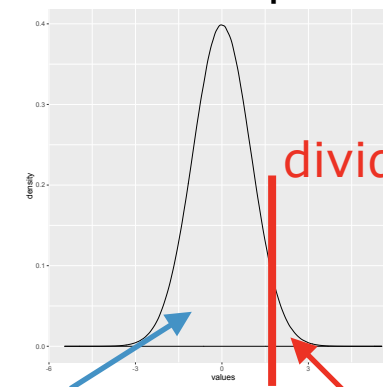
Because our real world distribution is identical to the null distribution (the null hypothesis is true), our alpha level is identical to the dividing line between correct decisions and false positives:

null distribution



=

real world experiments



accept the null

reject the null

correct decisions

false positives

In this way, the alpha level is the **maximum type I error rate** (because the maximum number of errors occurs when the null is true).



# Neyman-Pearson NHST

It is important to understand the relationship between these concepts:

- p-value:** The probability of obtaining a test statistic equal to, or more extreme than, the one you observed under the null hypothesis.
- $\alpha$ -level:** The threshold below which you decide to reject the null hypothesis
- Type I Error:** This is when the null hypothesis is true, but you mistakenly reject it.

If you consistently base your decisions on the alpha level, then your Type I error rate will either be less than or equal to your alpha level!




We say that it might be less because we admit that the null hypothesis might be false for some experiments. Every time the null hypothesis is false, you make one less Type I Error, so the rate goes down a bit!

Multiple comparisons

# Multiple comparisons

When people say “multiple comparisons”, what they mean is running more than one statistical test on a single set of experimental data.

The simplest design where this will arise is a one-factor design with three levels. Maybe something like this:

-  What do you think that John bought?
-  What do you wonder whether John bought?
-  What do you wonder who bought?

An F-test (ANOVA) or linear mixed effects model on this design will ask the following question:

**What is the probability of the data under the assumption that the three means are equal?**

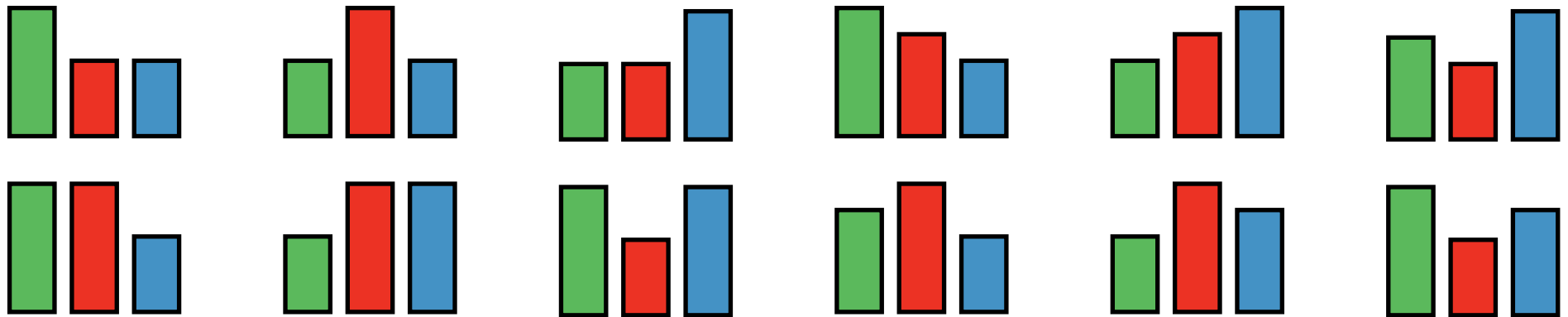
**null hypothesis**



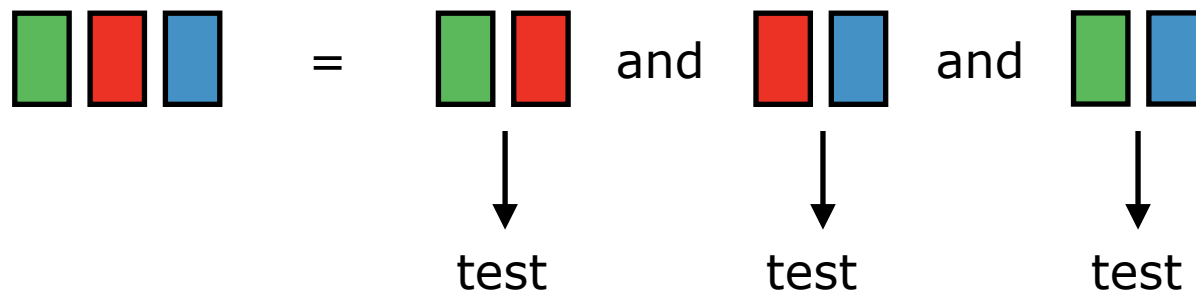
How many patterns of results will yield a low p-value under this null hypothesis?

# A significant result tells us relatively little

Here are all (I think?) of the patterns of results that will yield a significant result in a one-way / three-level test. As you can see, a significant result doesn't tell us very much.



If we want to know which of these patterns is the one in our data, we need to compare each level to every other level one pair at a time:



# The multiple comparison problem

# Review: Neyman-Pearson NHST

		State of the World	
		$H_0$ True	$H_0$ False
Decision	Reject $H_0$	Type I Error	Correct Action
	Accept $H_0$	Correct Action	Type II Error

**Type I Error:** This is when the null hypothesis is true, but you mistakenly reject it.

**Type II Error:** This is when the null hypothesis is false, but you mistakenly fail to reject it.

**$\alpha$ -level:** The threshold at which you decide to reject the null hypothesis.

# There are different error rates

## **Per Comparison Error Rate:**

The probability that any one comparison is a Type I error (number of errors/number of comparisons). You set this by choosing a threshold for your decisions. We call the threshold  $\alpha$ . Let's call the error rate PCER.

$$\text{PCER} = \frac{\text{number of errors}}{\text{number of statistical tests}}$$

## **Experimentwise Error Rate:**

The probability that an experiment contains at least one Type I error. We can call this rate EWER.

$$\text{EWER} = \frac{\text{number of experiments with 1 or more errors}}{\text{number of experiments}}$$

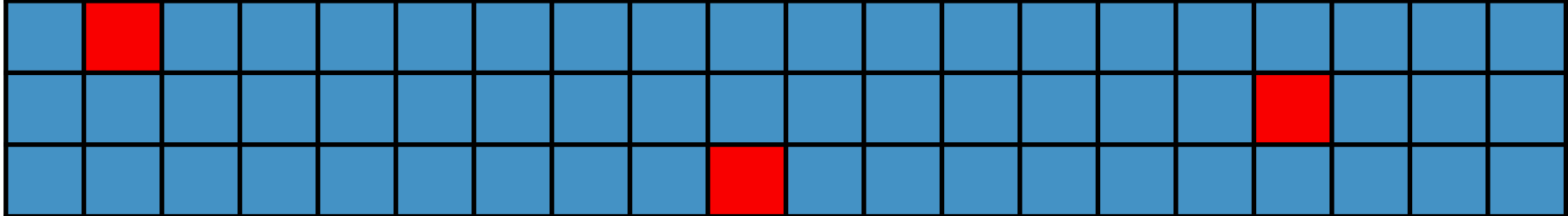
## **Familywise Error Rate:**

This is just like experimentwise error, but allows you to define sub-groups of comparisons inside of an experiment called a "family". So this is the probability that a family contains at least one error. In most experiments, there is just one family, so this will be equal to the experimentwise error rate. Let's call it FWER.

# Visualizing the different error rates

Imagine your experiment has 3 comparisons, and you run that experiment 20 times. Let's say you set  $\alpha$  to .05. Here are your results:

	e1	e2	e3	...	e20
comp1					
comp2					
comp3					



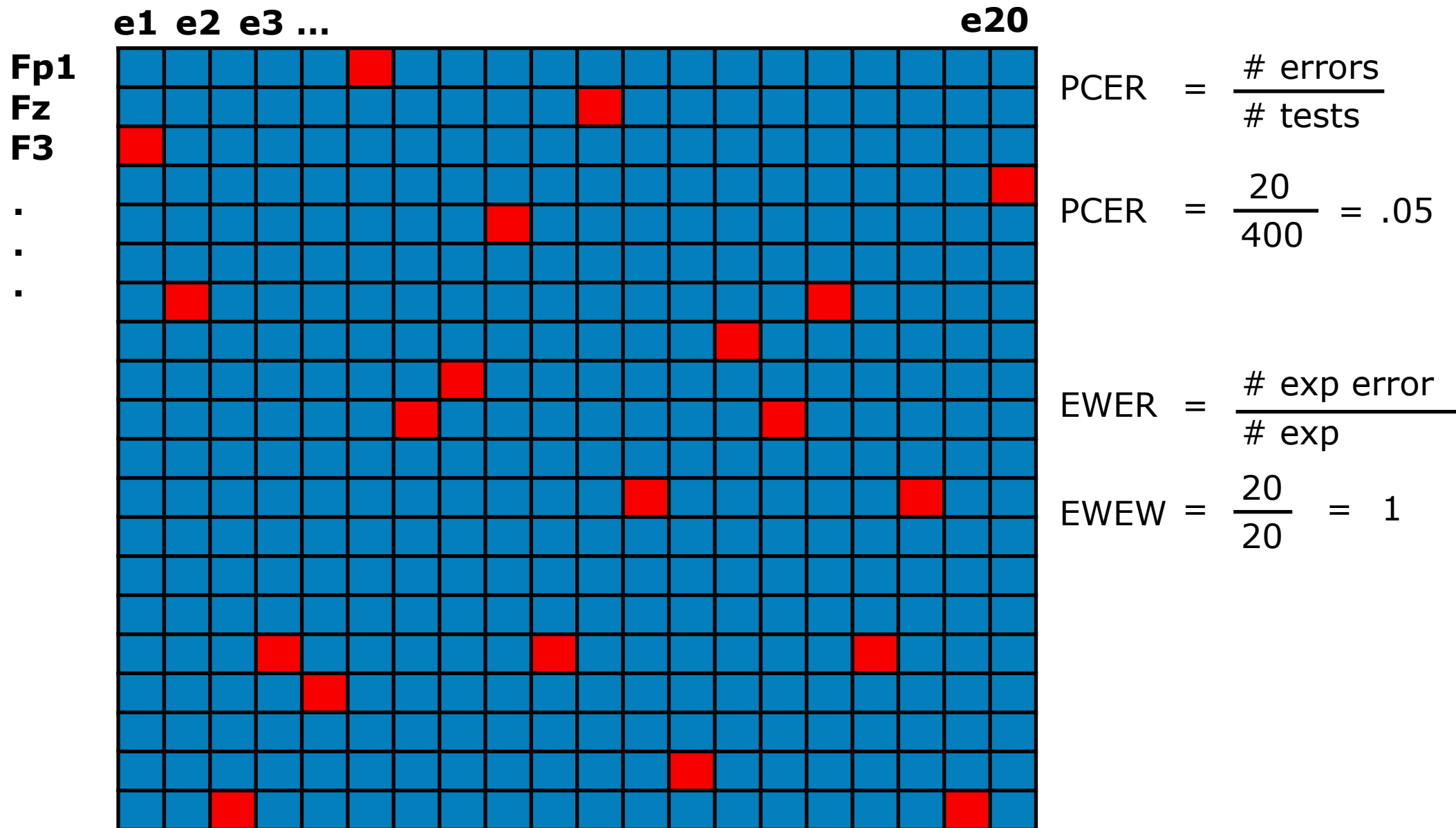
$$\text{PCER} = \frac{\text{number of errors}}{\text{number of statistical tests}} = \frac{3}{60} = .05$$

$$\text{EWER} = \frac{\text{number of experiments w/errors}}{\text{number of experiments}} = \frac{3}{20} = .15$$

When you make multiple comparisons, EWER is larger than PCER. This is the **multiple comparisons problem!**

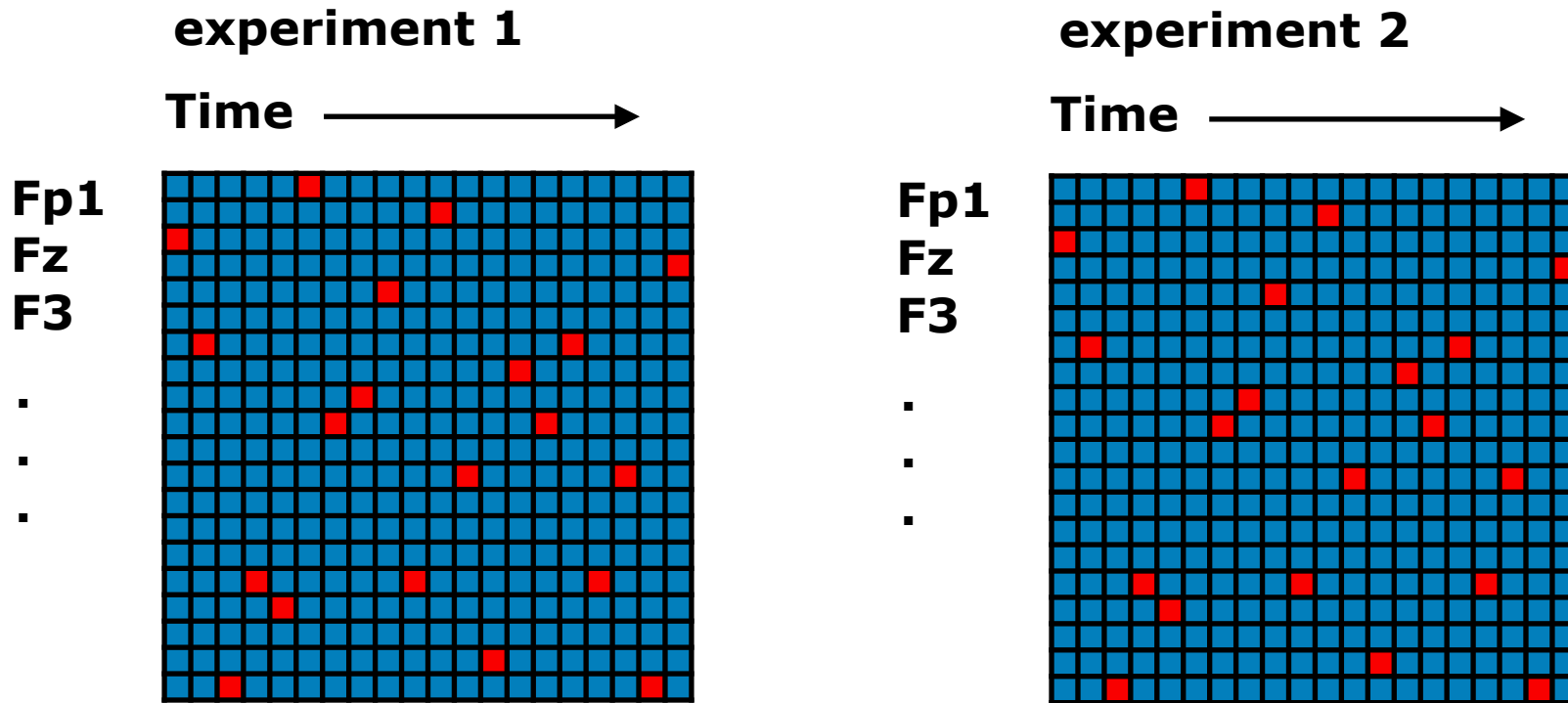


# EEG - multiple comparisons with channels



Once you reach 20 channels, it is possible that every experiment will have at least one false positive in it. And we haven't even considered time yet...

# EEG - multiple comparisons with channels



You will often see people say that EEG suffers from an “extreme version” of the multiple comparison problem. This is because each experiment potentially has a lot of comparisons in it: # of electrodes x # of time points.

If you tested each electrode x time point comparison, you are almost guaranteed to have an EWER of 1. In other words, you will definitely find statistically significant results in an EEG experiment using NHST and an alpha criterion of .05.

# An equation for relating EWER to $\alpha$

The relationship between  $\alpha$  and EWER is lawful, and follows this equation:

	e1	e2	e3	...														e20
comp1																		
comp2																		
comp3																		

$$\text{EWER} = 1 - (1 - \alpha)^C \quad \text{where } C \text{ is the number of comparisons.}$$

So for 3 comparisons and an  $\alpha$  set to .05, the maximum EWER will be:

$$\text{EWER} = 1 - (1 - .05)^3 = .142625$$

There is code in [multiple.comparisons.r](#) to demonstrate EWER, and how the EWER will always be more than PCER.

The take-home message is that multiple comparisons increases your type I error rate for the entire experiment!

# Controlling Experimentwise Error

## The Dunn Correction

(Mistakenly called the “Bonferroni correction” in the literature)

# Controlling EW/FW error

So now you can see that setting an alpha level of .05 for each comparison only controls error at the comparison level. If you want to control errors at the experiment (or family) level, you need to make an adjustment to your decision criterion.

Luckily, the equation for EW/FW error tells us exactly how to do that:

$$\text{EWER} = 1 - (1 - \alpha)^C$$

Since EW/FW error is dependent on  $\alpha$ , all we have to do is choose an  $\alpha$  that gives us the EWER that we want!



You could do this through guessing-and-testing if you want, but statistician Olive Dunn figured out a much faster way using one of mathematician Carlo Bonferroni's inequalities:

$$X \geq 1 - (1 - (X/C))^C$$

As you can see, this inequality looks very similar to the EWER equation above...

# The Dunn Correction (“Bonferroni correction”)

Here is how you can use Bonferroni’s inequality to set your  $\alpha$ , and control EWER:

First, replace the  $X$  with EWER because that is what we care about. (And  $C$  is the number of comparisons).

Next, notice that the term  $\text{EWER}/C$  is in the position that  $\alpha$  occupies in the EWER equation.

From that, it follows that if we set  $\alpha$  to  $\text{EWER}/C$  we can keep our EWER at or below the number we want!

$$\begin{array}{ccc} X \geq 1 - (1 - (X/C))^C & & \\ \downarrow & & \downarrow \\ \text{EWER} \geq 1 - (1 - (\text{EWER}/C))^C & & \\ & & \swarrow \\ \text{EWER} = 1 - (1 - \alpha)^C & & \end{array}$$

$$\alpha = \frac{\text{EWER}}{C}$$

The **Dunn correction** states that we can control our experimentwise error rate (EWER) by setting our decision threshold ( $\alpha$ ) to our intended experimentwise error rate divided by the number of comparisons ( $\text{EWER}/C$ ). See [multiple.comparisons.r](https://www.psychstat.org/multiplecomparisons.r) for a demo!

# The Dunn correction is too conservative

If we use the Dunn correction in EEG, we are going to have to set our alpha to a ridiculously small number:

32 channels x 1000 time points leads to 32,000 comparisons

$$\alpha = \frac{\text{EWER}}{C}$$

If we plug that into the Dunn equation, we get the following critical p-value:

$$.0000015625 = \frac{.05}{32000}$$

We are very unlikely to ever reach p-values of this level, even with large, robust results. This tells us that the Dunn correction is too conservative. It is terrific for guaranteeing an upper limit on the EWER, but when the number of comparisons is large, that conservatively works against us, as it only allows exceedingly large effects through (leading to lots of Type II errors, and therefore low statistical power). So we need something that has a good balance between control of the EWER and type II errors.

# A note on the name: Dunn vs Bonferroni

Olive Dunn proposed the correction in a paper in 1961, but didn't name it. She cites Bonferroni once for the use of the inequality. The paper is super mathy!

Bonferroni was a male mathematician who never worked on statistics.

The field seems to have named the correction sometime after Dunn's 1961 paper, and chose Bonferroni for the name. The question is why.

One possibility is to give credit for the use of the inequalities. But that doesn't go through. All statistical tests are named after the statisticians who discovered them, including correction procedures: Turkey, Scheffe, Fisher, etc. We don't name things after the mathematicians whose math they used (Euler, Leibniz, etc).

Another possibility is that Dunn did not invent the correction. Perhaps it was around before her, and called the Bonferroni correction, and she just did the mathematical work to figure out its properties. In that case, the name should be Dunn-Bonferroni. All other modifications of existing tests do this appending: the Holm-Bonferroni correction is a modification of Dunn's correction proposed by Holm in 1979 (where he doesn't cite Dunn!).



# Table of contents

1. Introduction: The big picture	Luck 1
2. Fundamentals	Luck 2
3. Hands-on training and best practices	Luck 5
4. The ERP processing pipeline in EEGLAB + ERPLAB	Luck 6, 7, 8
1. Load the data.	
2. Filter the data (high-pass, possibly low-pass).	
3. ICA for artifact correction (optional).	
4. Re-reference the data.	
5. Add channel locations.	
6. Epoch the data.	
7. Artifact detection and rejection.	
8. Average the epochs to create subject ERPs.	
9. Average the subject ERPs to create a grand average ERP.	
10. Plotting: waveforms, topoplots, difference waves	
11. Measure amplitudes and latencies.	
12. Run statistical tests.	
5. <a href="#">Creating a script for EEGLAB + ERPLAB</a>	
6. Creating a script for Fieldtrip	

# An example script for ERPLAB

I have posted an example script for ERPLAB on the course website.

It may require some light editing to work on your computer (and to fix any errors that I inadvertently made).

The ERPLAB wiki has a nice tutorial for learning how to script in ERPLAB:  
<https://github.com/lucklab/erplab/wiki/Scripting-Guide>.

# Table of contents

1. Introduction: The big picture	Luck 1
2. Fundamentals	Luck 2
3. Hands-on training and best practices	Luck 5
4. The ERP processing pipeline in EEGLAB + ERPLAB	Luck 6, 7, 8
1. Load the data.	
2. Filter the data (high-pass, possibly low-pass).	
3. ICA for artifact correction (optional).	
4. Re-reference the data.	
5. Add channel locations.	
6. Epoch the data.	
7. Artifact detection and rejection.	
8. Average the epochs to create subject ERPs.	
9. Average the subject ERPs to create a grand average ERP.	
10. Plotting: waveforms, topoplots, difference waves	
11. Measure amplitudes and latencies.	
12. Run statistical tests.	
5. Creating a script for EEGLAB + ERPLAB	
6. <a href="#">Creating a script for Fieldtrip</a>	

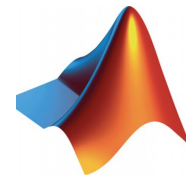
# An example script for Fieldtrip

I have posted an example script for Fieldtrip on the course website.

There is extensive documentation (tutorials, etc) for Fieldtrip: <http://www.fieldtriptoolbox.org/>

## Matlab + FieldTrip

**MATLAB:** A proprietary computer language that specializes in matrix algebra. It costs money, but the university buys it for us.



**FieldTrip:** A free, open source toolbox for EEG analysis developed by the Donders Institute in Nijmegen.



## Jon's opinion:

FieldTrip is the best option if you want to do time-frequency analysis (which we will talk about later in the course). It is also completely fine for ERPs. But it does not have a GUI, so you have to be comfortable with scripting (it is a set of Matlab functions that you can call to perform different EEG analyses).