An algorithm for detecting ERP components using the grand average waveform as a template

Sven Lesche[1]

[1] Ruprecht-Karls-University Heidelberg

Author Note

Author Notes go here.

The authors made the following contributions. Sven Lesche: Conceptualization, Writing - Original Draft Preparation, Writing - Review & Editing.

Correspondence concerning this article should be addressed to Sven Lesche, Im Neuenheimer Feld 695, 69120 Heidelberg. E-mail: sven.lesche@psychologie.uni-heidelberg.de

Abstract

One or two sentences providing a **basic introduction** to the field, comprehensible to a scientist in any discipline.

Two to three sentences of **more detailed background**, comprehensible to scientists in related disciplines.

One sentence clearly stating the **general problem** being addressed by this particular study.

One sentence summarizing the main result (with the words "**here we show**" or their equivalent).

Two or three sentences explaining what the **main result** reveals in direct comparison to what was thought to be the case previously, or how the main result adds to previous knowledge.

One or two sentences to put the results into a more **general context**.

Two or three sentences to provide a **broader perspective**, readily comprehensible to a scientist in any discipline.

Keywords: keyword, more keyword, crazy keyword

Word count: X

An algorithm for detecting ERP components using the grand average waveform as a template

History of Extraction Methods

Importance of ERPs

Event-related potentials (ERPs) are a key tool used in a variety of fields of study. Clinical psychology [REF], cognitive psychology [REF], experimental psychology [REF] all make use of ERPs to investigate the neural processes. Both the amplitude as well as the latency of components are often used to gain insights into individual differences in cognitive processes [REF]. Accurate estimation of the latency of ERP components thus presents an important topic relevant to all applications of ERP research [REF].

something about N2 and P3 processes here also write something about its relevance for individual differences research

the first paragraph should already illustrate how imporant accurate latency extraction methods are

Prior Algorithms

Peak Latency

A number of algorithms have been developed hoping to ease identification of components and their latencies. The earliest version of an algorithm to automatically extract ERP latencies is the *peak latency* approach [REF]. It involves finding the point in time within a set measurement window that has the largest voltage deflection in the appropriate direction. This method was developed as it was easy to implement **something about historical ease of use Luck, 2014, p. 286 cites Donchin & Heffley, 1978 as historical example**. Even though an algorithm detecting the *peak latency* inside some measurement window is easy to implement and run, it can be corrupted by several factors.

be aware that luck lists the following issues: peak != component; high-freq noise; time-window/point differences in subjects; biased by noise-level; is

non-linear measure; influence of latency jitter; difficult to compare with RT

Issues with peak latency.

1. Choice of measurement window As the peak latency method is blind to the general structure within the ERP signal, it simply chooses the point in time with the largest voltage deflection within the measurement window. The "true" maximum signal may lay just outside of the measurement window, resulting in the algorithm picking an outside edge of the window to reflect the *peak latency*, even though the signal increases in amplitude just following that timepoint. This may be true even if there is a clear component-like structure fully contained in the measurement window. Later components may already influence the amplitude on the outer ranges of the measurement window, resulting in their latency being picked instead of the true component of interest. Simple *peak latency* algorithms can thus fail to pick the *local maximum* of a component of interest fully contained inside the measurement window in favor of a datapoint possibly influenced by later or earlier components. This problem is often referred to as the *superimposition problem* [REF, just luck, 2014?].

To combat the influence of other components, [REF] have modified the simple *peak latency* algorithm to only consider maxima inside the measurement window which are also larger than the surrounding datapoints. A spike just on the edge of the measurement window induced by a later component will not meet these criteria, as the last datapoint inside the measurement window will be smaller than the following datapoint. This method is more accurate in finding the *local* peak latency [REF] but still vulnerable to other issues with peak latency approaches.

2. Superimposition of other components The *peak latency* approach is also especially vulnerable to influences of other components on the ERP signal - **bad sentence**. Even though the inclusion of neighboring datapoints does help the algorithms find local optima, peak latency approaches will fail to detect the peak of the component of interest if the measurement window contains portions of non-relevant components.

Researchers thus have to carefully choose the measurement window in order to **not include other components**, **keep it homogenous across all subjects**, **fully include the component of interest**

3. high frequency noise The influence of high frequency noise presents another issue for peak latency algorithms. The maximum voltage deflection may not reflect the true point in time at which the component reaches its maximum. Rather it may be the result of high frequency noise inducing a spike in the signal, independent of any cognitive process. This is problematic especially in the identification of the latency of later ERP components, as the larger measurement window most commonly applied here will increase the likelihood of high frequency noise impacting the maximum voltage deflection [REF, luck? or liesemeier?].
4. The peak latency is also just not that relevant Lastly, as [Luck, 2014] so aptly states: *There is nothing special about the point at which the voltage reaches a local maximum.* The largest deflection does not inherently relate to any physiological or psychological process and may not even reflect the true maximum of the component of interest due to the *superimposition problem.* [REF luck rule paper] visually demonstrates how peak latency may be a result of the overlap of multiple components and not related to any single component.

Fractional Area Latency

Fractional Area Latency approaches hope to remedy some of the problems associated with *peak latency* algorithms. *Fractional Area Latency* revolves around the area under the ERP signal in a given measurement window and finding the point in time that divides that area into a given fraction before and a after that point in time. The timepoint halving the area under the signal, for example, is referred to as the *50% area latency.* This approach is much less susceptible to the influence of high frequency noise, as short spikes in the signal do not influence the area under the signal much. It remains highly dependent on the measurement window [Luck, 2005]. Choosing a shorter window may result in only a part of

the area of the component of interest being captured. A wider measurement window on the other hand might be influenced by surrounding components. *Fractional Area* measures thus work best for investigating an isolated component [Luck, 2014], limiting the applicability of algorithms based on this.

Jackknifing

Another approach towards dealing with noisy subject-level ERPs is to try and remove that noise by averaging multiple subject-level ERPs. This technique is referred to as *jackknifing*. It uses the grand average of N subject-level ERPs and generates N sub-grand averages by removing one of the subject-level ERPs from this grand average. This results in N ERPs with higher signal-to-noise ratios as they are the average of $N-1$ subject-level ERPs. Both *peak* and *area* based measures can then be applied to the jackknifed data to extract latencies. As jackknifed signals share $\frac{N-2}{N} * 100$ percent of the subject-level signals that are averaged, each jackknifed-subaverage is quite similar to all others. This artificially decreases the error variance, which needs to be corrected for when testing for significant differences between groups [(Ulrich & Miller, 2001), aus Sadus (2023)]. Because Latencies extracted from jackknifed ERPs are based on averaged waveforms, they can not be readily associated with any single subject, preventing this method from generating individual-level latency estimates needed for individual differences research. This problem was adressed by [Smulders 2010, aus Sadus (2023)] who introduced a transformation able to generate individual-level latency estimates.

Comparison of Algorithms

[Kiesel 2008] compared a number of extraction methods for a variety of ERP components and found [SUMMARY HERE]. Their analyses indicates **drawbacks of each method summarized** [Sadus 2023] used a multiverse approach to investigate P3 stuff and utilizied **these** algorithms as well as manual extraction of ERP latencies ...

Look at comparison studies report what they thought is best. Sadus et al. as last the conclusion from this needs to be made clear: no algorithm provides a sufficiently

reliable method of extracting component latencies. - Manual inspection seems to reign supreme, **talk about the practical problems associated with this**, **also talk about the methods problems, low objectivity**

Our algorithm

Motivation

When visually inspecting ERP signals, the goal is to identify a pattern within the signal that resembles the component of interest in shape, size and location. None of the previously described algorithms used in most ERP research aim to replicate this human behavior but rather focus on generating similar decisions, not similar decision processes. The goal of this work is to introduce a new method for extracting component latencies that more closely resembles the manual extraction process.

template matching

Introduction

The general question each ERP-researcher asks themselves when investigating ERP signals is: "Where in this ERP signal is the component of interest?" Most researchers have some mental representation of what the component *should* look like and where it *should* generally appear that helps them identify a specific component in some noisy ERP. Finding a given pattern inside a noisy signal is not a novel task in the field of signal processing. Algorithms aiming to detect the appearance of a pattern - a *template* -inside audio-, video- or radio signals have been around since **HISTORY** [REF] and a large amount of research has gone into perfecting these *template matching* algorithms [REF].

No matter the implementation of the template matching algorithm, they all aim to answer the question "Does this (shorter) template appear in my (larger) signal?". To achieve this, a researcher needs to specify two things. First, the template that they want to search the signal for. And second, the *similarity measure* by which the algorithm can figure out how well the template fits in a given spot of the signal.

Template generation

Depending on the field, the template is easily specified. If you are looking to extract a particular audio-signal or some image [REF].... It becomes more challenging when it is not exactly certain what template you are looking for. Recent research in image processing has attempted to use template matching to process faces, for example. You cannot just use "the ideal nose" describe more here... [REF] **more about history of pattern matching here**... A similar issue has plagued template matching approaches in ERP research. The variance in ERP signals introduced by the task used or the type of participants prevents a successful implementation of template matching algorithms using one idealized template over all types of studies.

Similarity measures

A number of similarity measures have been proposed to capture the degree of similarity between a template and the signal. These include, but are not limited to: the Sum of Absolute Differences (SAD), the Mean Square Error (MSE), the Geometric Distance, Mutual Information or Cross-Correlation (CC)[[mahalakshmi2012image]] [REFs]. We have chosen to implement algorithms based on two of these measures, one minimizing the Sum of Squared Differences (MINSQ) and one maximizing the correlation between the template and the signal (CORR).

Our goal is to show that the concepts used in signal processing to find specific patterns can be adopted by ERP researchers to use *template matching* algorithms when identifying ERP components.

Prior attempts

There have been some attempts at using an idealized component structure, like the COMPONENT[depending on citation] as a template to identify the component in noisy subject-level data. The variance of ERP signal structure within and especially between ERP studies has presented a challenge to this approach [REF]. The morphology of the P3, for example, varies depending on the task used and the participants studied [REF]. This

compromises the ability to use a heavily idealized P3-like component as a template to search for in subject-level ERPs [Does it work with GA?]. **want to talk about the Hidden-semi-markov-chain approach?**

our solution

This problem can be addressed by using using an experiment-specific template of the component of interest that will reflect influences of the task and participant sample on the morphology of the idealized ERP component. The grand average enables us to generate this sort of idealized ERP component. The grand average's high signal-to-noise ratio leads to a more clear component structure from which the researcher can specify the times during which the component of interest occurs. This part of the grand average's signal can then be used as a template and matched to subject-level ERPs.

Differences to other template matching studies

Template Matching in ERP studies differs from classical approaches in the sense that the template may change shape based on individual differences. The question shifts from "Does the template appear in the signal" to "Which transformation of the template appears in the signal". Of course some aspects of both questions are present in traditional approaches and our algorithm should also be able to detect cases in which no clear match is found for the component of interest. But the goal of this algorithm is to accurately measure individual differences in the latency of ERP components. Since not all subjects' ERP components have the same size, shape and location as the grand averages', transformations of the template allow for a more realistic representation of individual differences in ERP signals. Finding the transformation of the template that best fits each subject-level ERP thus allows us to identify how the subject-level signal differs from the grand average. This is precisely the issue that individual differences research using ERPs is faced with and that we hope to provide an algorithm for.

Subject-level transformations as the goal

Linear transformations of the grand average reflect individual differences in the shape, size and location of ERP components. All ERP signals can be transformed into a sum of sines function with the properties amplitude, frequency and phase for each sine term. ERP signals can thus be modified by multiplying all amplitude-factors **this is terrible, do math notation!** by a factor a. This effectively transforms the amplitudes of the overall signal by that factor. Multiplying all frequency factors by a factor b can change the latency of peaks in the signal. It effectively squishes or stretches the signal along the x axis. We have limited our transformations to these to operations, with future studies inspecting the impact of manipulating the phase parameter. This would effectively "shift" the signal. ** Explain why this was done show examples of the effect of this**

Figuring out the optimal transformation

We can produce an infinite number of transformed grand averages each one version of our "template". The algorithm then finds the one template that best fits the subject-level ERP signal and returns the parameters used to generate this template. A subject whos P3 peaks at 400ms, while the grand average's P3 is located around 600ms would most likely receive a parameter around $b = 1,5$. All frequency parameters in the sum of sine functions would be multiplied by 1,5, effectively "speeding up" the signal and thus "squishing" it more closely to the origin.

Component latencies can also be recovered from this method. Researchers need to specify a time-point of the grand average denoting the latency of the component of interest in the grand average. The transformation parameters resulting in the best fit of the grand average to the subject level signal can then just be applied to the grand average latency to recover the subject-level latency of the component.

The algorithm is naive to the method used to generate the component latency of the grand average. It simply applies the optimal transformation to a given latency value. Researchers can use whatever tool they want to identify component latencies in the grand

average waveform. Both peak and area latency measures, 50% area or onset / offset latencies can be supplied to the algorithm. It will then return the transformed latency based on the transformation parameters.

Even though the algorithm allows for the use of peak latency measures when specifying the grand average component latency, it is by no means a peak based algorithm. Approaches based on both similarity measures, Squared Sum of Differences and Cross-Correlation are not impacted by high frequency noise.

Talk about how this matches the entire component and is also similar to area measures. Talk about how this is more like z-Standardizing the Dataset. You can extract peak latencies, but you can also just leave it be defined by the stretching parameter

Uncertain parameters

Validation

We will investigated psychometric properties of the latency values generated by this algorithm. We will estimate reliability of extraction methods **explain how we hope to validate this algorithm what are we investigating: measurement window, filtering, sub-GAs** explain how we hope to make ERP extraction more objective and easier**

** you can say something about moving averages clearing up signal-to-noise ration (in [[william2020erp]] cited: "Stein's Unbiased Risk [21]. This thresholding technique has better signal to noise ratio and outperforms other methods such as moving average and Savitzky Golay") **

** in methods, talk about how actually the subject-signal is transformed also talk about the weighting function of the minsq algorithm **

Method

The Method section is usually a good place to start embedding your data-child documents

Describe your method here. You can embed pictured and reference their label (see

Figure 1). You need to call `\@ref(TYPE:CHUNK-NAME)` to embed reference the output of an r chunk.

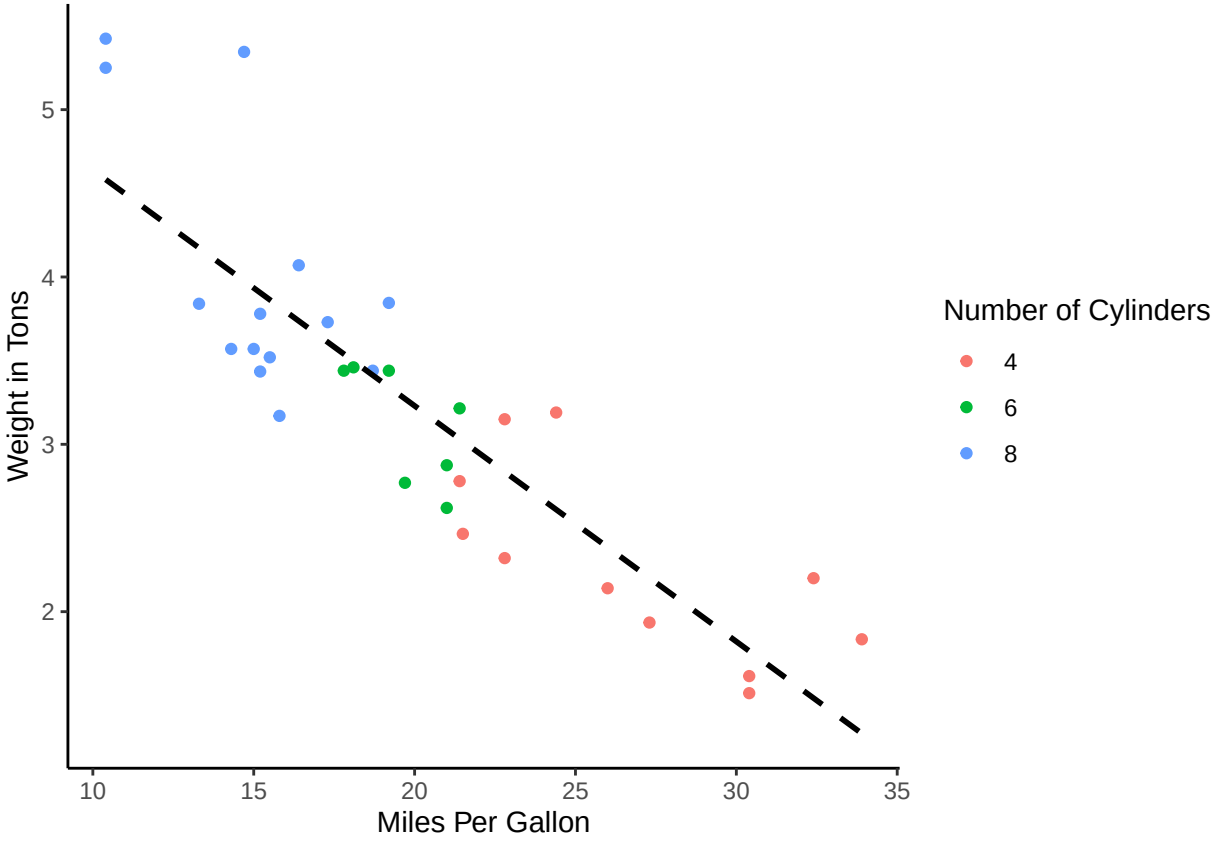

Figure 1

Plot of mpg over wt

You can also refer in inline code to data objects. We used the dataset `mtcars` for our analysis. It consists of data of 32 cars. The syntax "r command" in backticks will evaluate the command using your R-engine.

Results

Write your results here. You can add chunks for additional analysis. But to ensure readability, I would recommend conducting all anaylsis inside the appropriately named children.

You can cite R-packages used by calling the object `r_citiations`. If you want to

Table 1

The top five cars by Miles Per Gallon (MPG)

car	mpg	disp
Pontiac Firebird	19.20	400.00
Hornet Sportabout	18.70	360.00
Merc 450SL	17.30	275.80
Merc 450SE	16.40	275.80
Ford Pantera L	15.80	351.00

Note. This table was generated using papaja::apa_table()

only cite R itself within your text, but refer to all packages used in a footnote, call `r_citations$r` in text and `r_citations$pkgs` after the end of the sentence. This report was generated using R [Version 4.1.3; R Core Team (2022)][1].

You can print tables using the wonderful `apa_table` command provided to you by `papaja` (see Table 1). Here it is best to set the caption using the `caption` argument provided by `apa_table()`.

As you can see, the best car is Pontiac Firebird.

Discussion

Here you can discuss your results.

[1] We, furthermore, used the R-packages *knitr* (Version 1.41; Xie, 2015), *papaja* (Version 0.1.1; Aust & Barth, 2022), and *tidyverse* (Version 1.3.2; Wickham et al., 2019).

References

Aust, F., & Barth, M. (2022). *papaja: Prepare reproducible APA journal articles with R Markdown*. https://github.com/crsh/papaja

R Core Team. (2022). *R: A language and environment for statistical computing*. R Foundation for Statistical Computing. https://www.R-project.org/

Wickham, H., Averick, M., Bryan, J., Chang, W., McGowan, L. D., François, R., Grolemund, G., Hayes, A., Henry, L., Hester, J., Kuhn, M., Pedersen, T. L., Miller, E., Bache, S. M., Müller, K., Ooms, J., Robinson, D., Seidel, D. P., Spinu, V., ... Yutani, H. (2019). Welcome to the tidyverse. *Journal of Open Source Software*, *4*(43), 1686. https://doi.org/10.21105/joss.01686

Xie, Y. (2015). *Dynamic documents with R and knitr* (2nd ed.). Chapman; Hall/CRC. https://yihui.org/knitr/

Table A1

Best car only

car	mpg	disp
Pontiac Firebird	19.20	400.00

Appendix A

Talking about appendices

First-level headers create appendix-sections labelled A-Z. You can print tables here aswell and refer to them in your main part. They will receive a prefix to their Table/Figure Number based on the appendix section they are in (see Table A1).

Appendix B

Another section

this creates another appendix section