APPENDIX A

Artifact detection algorithms

The rejection matrix is stored in EEG.artifacts.BCT.

The input of each artifact algorithm is the EEGLAB structure and some parameters. The output of each artifact detection algorithm is the EEGLAB structure and a logical matrix containing the data rejected by that algorithm. The algorithms update the rejection matrix in the EEGLAB structure, such as it contains the old and new rejected data. The data already rejected is not used for the estimation of relative thresholds.

The following algorithms are particularly sensitive to the detection of non-functional channels:

- ***eega_tRejCorrCh:*** This algorithm relies on the high correlation existing between adjacent channels, especially in high-density systems. The correlation between all channels is computed in sliding time windows (4 s length, 2 s step by default), and for each channel and time window, the average over the stronger correlations (top 5% by default) is the measure used for defining bad data. The algorithm rejects channels per time window with a top correlation lower than a threshold. Because the method's measure is independent of the signal amplitude and its distribution is not normal, we recommend using an absolute threshold (0.4 by default). However, the function supports the estimation of a unique relative threshold for all electrodes.
- ***eega_tRejPwr:*** This algorithm identifies segments of bad data based on the power at different frequency bands. It applies FFT in sliding time windows (4 s length, 2 s step by default), and computes the average power in each frequency band, electrode, and time

window, and expresses it in decibels relative to the median value across all time windows and electrodes. By default, the algorithm z-scores the data per channel, uses relative thresholds, and rejects data when the power in the frequency band [1, 10] Hz is below the threshold or the power in the frequency band [20, 40] Hz is above the threshold. Notice that for this algorithm, the same threshold is used for all electrodes.

The following algorithms are more specifically sensitive to motion artifacts, but they will also partially identify non-working channels and other types of artifacts.

- ***eega_tRejAmp:*** This algorithm rejects samples with an amplitude below or above a threshold. By default, it sets the thresholds relatively and per electrode and applies a 50 ms mask (it removes the 50ms before and after any segment rejected by the algorithm).
- ***eega_tRejTimeVar:*** This algorithm computes the variance of the signal in a sliding time window (0.5 s length, 0.1 s step by default) and rejects samples with a variance above or below a threshold. By default, it sets the thresholds relatively and per electrode.
- ***eega_tRejRunningAvg:*** This algorithm computes for each sample two weighted running averages and rejects samples with a too high fast running average or with a too high difference between the fast and the slow running average. The fast-running average is computed as $AvgF_j = 0.800 \times AvgF_{j-1} + 0.200 \times X_j$ where X_j is the data at sample *j*, and the slow running average as $AvgS_j = 0.975 \times AvgF_{j-1} + 0.025 \times X_j$. By default, it sets the thresholds relatively and by electrode and applies a 50 ms mask to the bad segments.
- ***eega_tRejFastChange:*** This algorithm rejects data when the maximum change in a given time window (20 ms by default) is larger than a threshold. It specifically detects jumps in

the signal and can identify some non-neural activity like heartbeats. By default, it sets the thresholds relatively and by electrode.

- ***eega_tRejAmpElecVar:*** This algorithm rejects data based on the variance of the signal across electrodes. If the amplitude for a given electrode is too far away from the median of all electrodes, it is rejected. Notice that for this algorithm, the threshold is by definition relative and defined for all the electrodes. By default, it applies a 50 ms mask to the bad segments.

The following algorithms can be used to reject/re-include data according to the rejection matrix *EEG.artifacts.BCT*.

- ***eega_tIncShortBad:*** This function includes segments of rejected data that are shorter than a lower threshold (20 ms by default).
- ***eega_tRejShortGood:*** This function rejects segments of good data between segments of bad data if they are shorter than a lower threshold (2 s by default).
- ***eega_tMask:*** This function applies a mask to the bad segments (50 ms by default).
- ***eega_tRejChPercSmpl:*** This function rejects all the data for channel and epoch if more than an upper limit of samples were rejected (50% by default).
- ***eega_tRejSmplPercCh:*** This function rejects all the data at a particular time-point if more than an upper limit of channels were rejected (30% by default).

We also provide a function to run a combination of multiple artifact detection algorithms.

- ***eega_tArtifacts:*** This function runs a combination of artifact detection algorithms to ensure proper artifact detection. The function performs a defined number of loops of rejection.

Within each loop, it applies the specified algorithms on the data remaining from the previous loops. After each loop, it updates the rejection matrix. Both the algorithms to run and its parameters are provided to the function as a structure. For some algorithms to work correctly, it is important that the data is filtered—especially low-pass filtered to remove line noise. It can be specified that the data needs to be low or high pass filter before performing the artifact rejection (notice that the function never modifies the output data, the filtered data are only used for the computation). The function can either reset or keep (default) the initial rejection matrix.

APPENDIX B

Description of the function to define BT and BC

We created a single function to define bad times (BT), bad channels per epoch (BC), and bad channels during the whole recording (BCall) on both continuous and epoched data. Notice that the definition of bad times and channels based on the rejection matrix are linked. For example, if we define BT as samples with more than 15% of rejected electrodes and that for one subject, 18% of the channels are non-functional, we will define all times as bad and discard the entire recording. To avoid this kind of problem, we created a function that can iteratively approximate the final thresholds used for defining BT, BC, and BCall.

- ***eega_tDefBTBC:*** This function defines *EEG.artifacts.BT* as a logical matrix with size 1 x samples x epochs signaling the bad samples, *EEG.artifacts.BC* as a logical matrix with size channels x 1 x epochs specifying the bad channels during each epoch, and *EEG.artifacts.BCall* as a logical vector with a length equal to the number of channels, indicating non-functional channels for the entire recording. The thresholds used to define

BT, BC, and Bcall have to be provided as input. If more than one threshold is provided for each definition, they will be applied as follows. First, the function defines bad times as those where the proportion of rejected channels (excluding bad-channel in BC) at each sample is above the first threshold. Second, it defines bad channels during the whole recording as those for which the proportion of total rejected samples (excluding bad times in BT) is above the first threshold. Third, it defines bad channels per epoch in an analog way, but computing the rejected samples per epoch and using the corresponding first threshold. It repeats the process by looping over all the thresholds provided. This iterative approximation to the thresholds allows a better definition of BT, BC, and Bcall. The function also allows applying a mask to bad-times (non applied by default), ignoring bad times shorter than a lower limit (0 by default), and marking as bad-times short good-times between two bad-times (0 by default).

APPENDIX C

Description of the artifact correction algorithms

We provide a function to perform the target PCA correction and two separate functions to perform the spatial interpolation of channels. One function interpolates channels non-functional during the entire recording (or an entire epoch when applied on epoched data). The second function interpolates segments of rejected data. All functions mark corrected data as good in the rejection matrix and indicate which data has been interpolated in another logical matrix, *EEG.artifacts.CCT* (true indicates corrected data).

- ***eega_tTargetPCAxElEEG*:** This functions corrects transient artifacts using target PCA. It concatenates the data segment to correct, applies PCA to this subset of data, and removes

the first *n* components. Finally, the interpolated segments are spliced back into the data and aligned to it to avoid discontinuities. By default, the segments are concatenated relatively to the immediate previous sample. This alignment introduces drifts in the signal; thus, after artifact correction using this method, the data should be high-pass filtered again.Small drifts might remain affecting frequencies below the filter frequency; therefore, noticeable on long data segments (or continuous data). However, these drifts will not affect baseline-corrected epoched data or frequencies above the high-pass filter. By default, the correction is restricted to data segments indicated as good data in *EEG.artifacts.BT*, shorter than 100 ms, and to the channels indicated as good in *EEG.artifacts.BC*. Applying the method to very short data segments guarantees that it is implemented on sudden short signal changes as jumps or heartbeats. We recommend removing 90 % of the variance because we have observed that this is the minimum percentage of variance removal that results in a proper artifact correction.

- ***eega_tInterpSpatialSegmentEEG*:** This function uses spherical spline to spatially interpolate channels not working during limited periods. The interpolated segments are spliced back into the data and aligned to it to avoid discontinuities. By default, the segments are concatenated relatively to the immediate previous sample. As before, this alignment introduces drifts in the signal, requiring subsequent high-pass filtering. The signal is reconstructed only during segments indicated as good data in *EEG.artifacts.BT* and when the proportion of channels rejected in the rejection matrix is lower than a threshold. By default, segments shorter than 100 ms are not interpolated, and a 1 s mask is applied to all segments before interpolation.

- ***eega_tInterpSpatialEEG:*** This function uses a spherical spline to interpolate channels identified in *EEG.artifacts.BC* as non-functional during an entire epoch (or the entire recording when applied to continuous data).

APPENDIX D

Description of the function for ICA and DSS

We created a function that performs the ICA combined with wavelet-thresholding ICA and afterward uses the iMARA algorithm to identify components associated with non-neural artifacts automatically. For short recordings, the function can run the analysis on subsets of electrodes or use PCA to reduce the dimensionality.

- ***eega_pcawtica:*** This function applies ICA to the removal of non-neural sources. The function operates as follows. *(1)* It makes a copy of the data, which is high-pass filtered (at 2 Hz by default) and low-pass filter (at 40 Hz by default). *(2)* It removes bad samples and bad channels. *(3)* If channel subsets are provided, it restricts the analysis to them. *(4)* It performs a first PCA (optional) + ICA. *(5)* It uses wavelet-thresholding to estimate transient artifacts and subtracts them from the data. *(6)* It performs a second PCA (optional) + ICA on the artifacts-free data. *(7)* It identifies the components to remove using iMARA. *(8)* It estimates the artifacts using those components. *(9)* It removes the estimated artifacts from the original data.

We provide a function, dss_denoise_EEG, which performs a DSS on an EEGLAB structure containing epoched data.

- ***dss_denoise_EEG:*** This function performs a DDS. It applies a first PCA, discards the last components, and normalizes the retained components. Then, it computes the average ERP, applies a second PCA to the average. The filter is the conjunction of the two PCA decompositions. Afterward, it projects the data into the filter and keeps only the first components. Finally, it projects the data back to the sensor space. This function enables designing a single filter and applying it to all the epochs or designing different filters for each condition (in this case, experimental factors have to be previously defined using *eega_definefactors*). The number of components to retain in the first and second PCA are provided as inputs to the function.

APPENDIX E

Other functions

Usually, a delay exists between the time the stimulation computer gives the order of presenting a stimulus and its actual presentation. Therefore, the latency of the events is usually corrected by the latency of DINs. We provide a function that performs this latency correction.

- ***eega_latencyevent:*** This function corrects the latency of certain types of events (indicated as input) using the latency information of the first DIN event (its name also has to be indicated as input).

For segmenting the data, we provide a function that, besides epoching the data, also epochs the logical matrices defining artifacts.

- ***eega_epoch:*** This function epochs the data and the logical matrices in the *EEG.artifacts* field.

We also provide two functions for identifying bad epochs, either based on the amount of rejected data or on the distance to the average evoked response.

- ***eega_tDefBEbaddata:*** This function identifies bad epochs based on the amount of rejected data and saves it as a logical vector in *EEG.artifacts.BE*. The function defines epoch *k* as bad based on *(1)* the proportion of bad data in *EEG.artifacts.BCT(:,:,k)*, *(2)* of bad samples in *EEG.artifacts.BT(1,:,k)*, *(3)* of bad channels in *EEG.artifacts.BC(:,1,k)*, and *(4)* of corrected data in *EEG.artifacts.CCT(:,:,k)*.
- ***eega_tDefBEdist*:** This function identifies bad epochs based on the Euclidean distance of the average referenced response during epoch *k* to the average response across all epochs. It calculates the Euclidean distance to the average at each sample for all epochs. Then it establishes a threshold based on a certain number of interquartile ranges from the third quartile. If, for epoch k, the distance to the average response is bigger than the threshold during a certain proportion of the epoch, the function rejects the epoch.

After data has been segmented and the epochs with artifacts rejected, additional steps are usually performed to obtain the ERP responses. These typically include average referencing, and eventually, data normalization and baseline correction. While normalizing the data is not necessary, this process can improve statistical power. We provide a function for data normalization flexible to normalized based on different dimensions.

- ***eega_normalization:*** This function z-scores the data. The function can apply the normalization over single or all electrodes and single or all epochs. It is also possible to

define specific values for the mean or the standard deviation. For example, setting the mean to 0 implies only dividing by the standard deviation.

In classical ERPs analysis, the responses across trials belonging to the same experimental condition are averaged to obtain an ERP per condition and subject. While the selection of trials usually needs to be adapted to specific requirements of each experiment, we provide two functions that can facilitate this process. The first defines experimental factors across trials and the second averages across any of them.

- ***eega_definefactors*:** This function creates a field, *F*, in the EEG structure that specifies different experimental factors. *F* is a cell array containing all the possible experimental factors. The values for each experimental factor are determined based on the information present within each epoch in *EEG.epoch(i).(field of interest).*
- ***eega_avgdatabyfactors:*** This function averages across one or multiple of the factors previously defined.

To obtain a summary of the preprocessing output for a group of subjects, we created a function that prints a table with the number of channels, samples, and epochs, the number of rejected and interpolated points, and the number of bad times, bad channels, and bad epochs. The table is created for some critical points during the pipeline: before epoching the data, before the rejection of bad epochs, and on the final stage.

- ***eega_printsummary:*** This function takes all the files in a given folder adhering to a specific name and creates a summary report for them.

Here we present the preprocessing reports obtained using APICE(3) for Datasets 1 and 2 before epoching.

Table S1. Example of a preprocessing report. The summary table corresponds to Dataset 1 after artifacts detection and correction using APICE(3). Column 1 (Subject): subject name. Column 2 (Ch): number of channels. Column 3 (Smpl): number of samples per epoch. Column 4 (Ep): number of epochs. Column 5 (BCT(%)): percentage of bad data computed as the number of bad data points relative to the total number of points (channels x samples x epochs). Column 6 (CCT(%)): percentage of interpolated data computed as the number of interpolated data points relative to the total number of points (channels x samples x epochs). Column 7 (BT(%)): percentage of bad times, computed as the number of bad times relative to the total number of time points (samples x epochs). Column 8 (BC(%)): percentage of bad channels computed as the number of bad channels relative to the total channels (channels x epochs).

Subject	Ch	Smpl	Ep	BCTx100	CCTx100	BTx100	BCx100
Session 20180926 S02.set	128	474632	1	0.19	8.31	0.29	0
Session 20180927 S03.set	128	473612	1	6.02	11.51	8.54	0
Session 20180927 S04.set	128	473702	1	1.74	11.39	2.92	0
Session 20180927 S05.set	128	475142	1	7.10	12.00	9.96	0
Session 20181003 S06.set	128	473703	1	9.96	6.93	13.98	0
Session 20181003 S07.set	128	473223	1	3.04	8.32	4.74	0
Session 20181004 S08.set	128	473762	1	0.69	7.23	0.89	0
Session 20181004 S09.set	128	473552	1	2.74	10.86	4.28	0
Session 20181004 S10.set	128	474842	1	0.88	8.05	1.59	0
Session 20181005 S11.set	128	473718	1	1.05	5.76	1.56	0
Session 20181005 S13.set	128	475021	1	1.53	11.81	2.88	0
Session 20181022 S14.set	128	476837	1	1.86	6.92	3.20	0
Session 20181022 S15.set	128	473117	1	1.79	13.52	2.98	0
Session 20181022 S16.set	128	474362	1	6.21	17.60	10.84	0
Session 20181023 S18.set	128	472562	1	2.50	10.60	3.76	0
Session 20181023 S20.set	128	474812	1	12.79	12.03	18.21	0
Session 20181024 S21.set	128	475622	1	3.98	7.66	6.77	0
Session 20181024 S22.set	128	473432	1	3.49	15.63	6.30	0
Session 20181128 S23.set	128	473793	1	1.87	11.14	3.19	0
Session 20181129 S24.set	128	473417	1	2.46	9.65	4.00	0
Session 20181129 S25.set	128	473642	1	10.85	15.92	16.79	0
Session 20181211 S29.set	128	473285	1	1.77	16.92	3.01	0
Session 20181211 S30.set	128	474557	1	3.09	10.92	5.64	0
Session 20181211 S31.set	128	473957	1	12.92	19.65	18.30	0

Table S2. Example of a preprocessing report, analog to Table S1. The summary table corresponds to Dataset 2 after artifacts detection and correction using APICE(3).

Subject	Ch	Smpl	Ep	BCTx100	CCTx100	BTx100	BCx100
SLCat_test_04	128	447924	1	3.66	8.67	5.06	0
SLCat_test_05	128	363156	1	4.12	18.61	5.49	0
SLCat_test_06	128	246804	1	7.15	21.62	9.77	0
SLCat_test_09	128	421924	1	2.71	20.52	4.19	0
SLCat_test_10	128	361000	1	7.56	28.56	11.45	0
SLCat_test_12	128	270476	1	30.40	29.93	43.93	0
SLCat_test_14	128	404125	1	27.87	34.45	36.17	0
SLCat_test_15	128	319828	1	23.35	35.47	32.93	0
SLCat_test_16	128	340604	1	21.38	18.11	26.51	0
SLCat_test_19	128	818724	1	29.15	36.41	41.36	0
SLCat_test_24	128	315424	1	18.75	21.15	25.14	0
SLCat_test_28	128	278088	1	1.88	25.42	1.93	0
SLCat_test_30	128	260272	1	6.10	44.11	7.55	0
SLCat_test_32	128	231316	1	7.86	22.00	12.39	0
SLCat_test_34	128	259744	1	24.76	19.90	32.95	0
SLCat_test_35	128	356780	1	8.32	25.53	13.27	0
SLCat_test_36	128	374336	1	9.23	25.92	12.83	0
SLCat_test_38	128	366024	1	6.27	24.36	8.38	0
SLCat_test_39	128	341388	1	9.94	20.11	15.52	0
SLCat_test_40	128	337004	1	12.64	18.02	17.25	0
SLCat_test_41	128	298860	1	6.44	30.44	11.06	0
SLCat_test_43	128	325968	1	6.69	22.01	9.50	0
SLCat_test_44	128	281164	1	13.08	31.35	17.39	0
SLCat_test_45	128	422608	1	4.40	23.58	7.45	0

When preprocessing a dataset, a series of processes are sequentially applied to many files. We have created a function that sequentially applies a series of specified functions to a set of files. The function operates on all the specified files in an input folder and saves the final files in an output folder.

- ***eega_RunAll:*** This function considers all the files adhering to a provided file name in a specified folder and saves the resulting EEGLAB structure in a specified output folder. The functions to apply and their parameters are provided to the function as a string, corresponding to the function name, followed by a cell array containing its inputs (excluding EEGLAB structure). It can run the functions for all the files found in the input folder or limit the output files that do not exist. It can create the output name by concatenating a string provided as input with the input file name.

APPENDIX F

Grand average ERPs for the different pipelines

The responses obtained using different pipelines were compared by running a one-sample t-test (FDR corrected by the number of samples) between each pair of preprocessing approaches. When significant differences were observed ($p \leq 0.05$ FDR corrected), they were reported. Notice that the grand average ERP might appear different between two conditions, but the difference might not be significant because the effect is driven by a single (few) subject(s). Significant differences across pipelines denote a consistent difference across subjects.

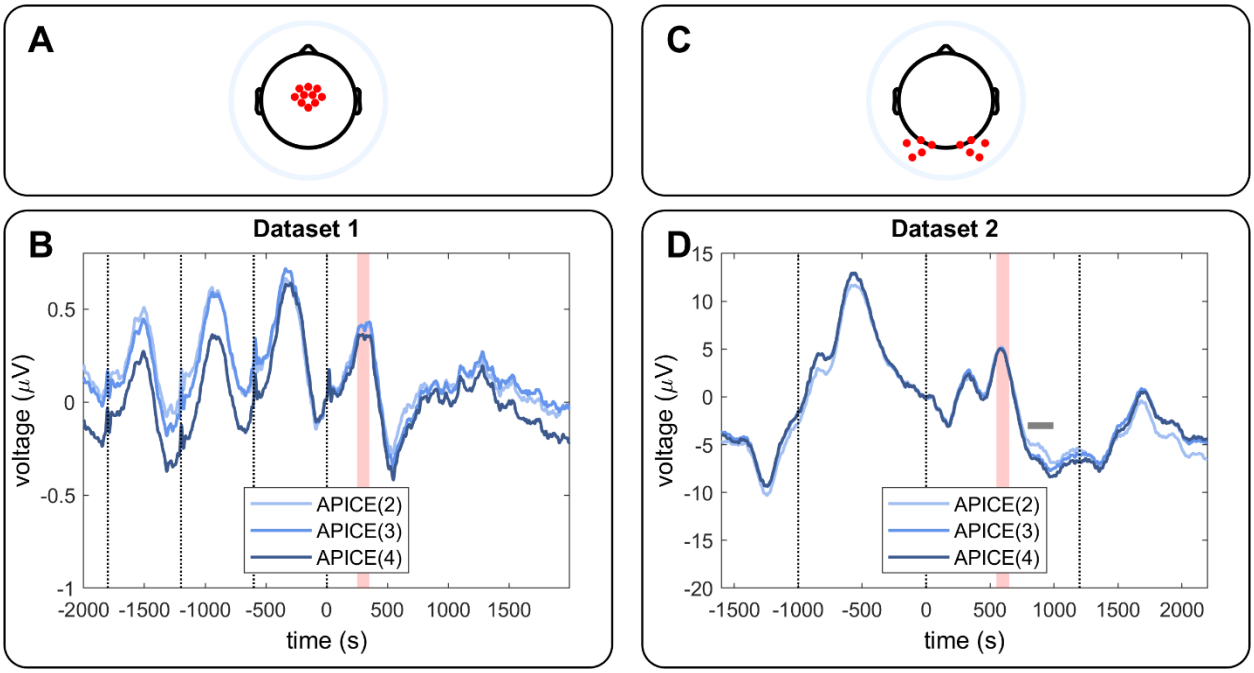

Figure S1. Grand average ERPs obtained using APICE with varying thresholds for artifact rejection. **(A)** Central electrodes (in red) considered for dataset 1. **(B)** Grand average ERP for the electrodes of interest for dataset 1. The peaks after the dotted lines (syllables' onset) correspond to the auditory ERP following each syllable. The shaded area shows the time windows where the SME was computed (250-350 ms, peak of the auditory response to the last syllable of the epoch). No significant differences were observed between ERPs ($p > 0.05$, FDR corrected). **(C)** Occipital electrodes (in red) considered for dataset 2. **(D)** Grand average ERP for the electrodes of interest for dataset 2. The first two dotted lines indicate the onset of two images, and the third dotted line the appearance of the attention grabber. P1 and P400 are visible after the onset of the images, followed by the visual response to the attention grabber. The shaded area shows the time windows where the SME was computed (550-650 ms, P400 to the second image). The gray line shows where significant differences ($p \leq 0.05$, FDR corrected) were observed between APICE(2) and APICE(4). No other significant differences were observed between ERPs ($p > 0.05$, FDR corrected).

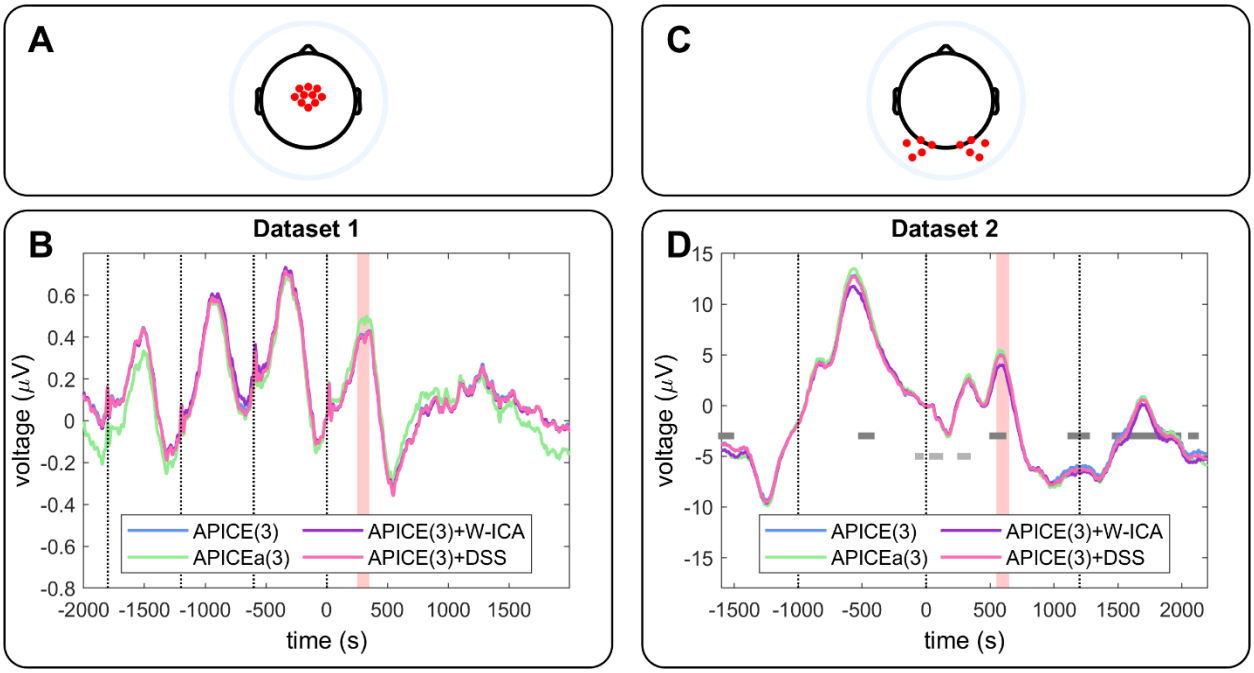

Figure S2. Grand average ERPs obtained using APICE and APICE variations. Analog to figure S1. **(B)** The blue line corresponding to APICE(3) is below the violet line corresponding to APICE(3)+W-ICA. No significant differences were observed between ERPs ($p > 0.05$, FDR corrected). **(D)** The blue line corresponding to APICE(3) is below the green line corresponding to APICEa(3). The dark gray line shows where significant differences were observed between APICE(3) and APICE(3)+W-ICA, and the light gray line between APICE(3) and APICE(3)+DSS ($p \leq 0.05$, FDR corrected). No other significant differences were observed between ERPs ($p > 0.05$, FDR corrected).

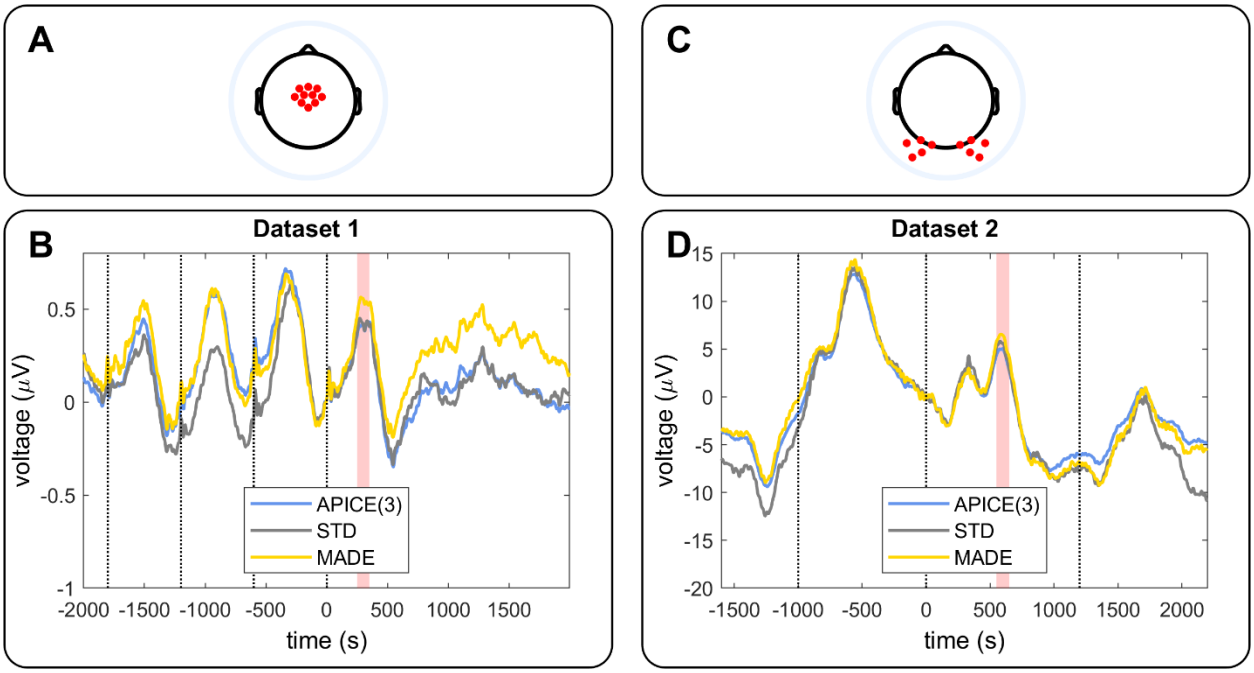

Figure S3. Grand average ERPs obtained using APICE(3), the Standard pipeline (STD), and MADE. Analog to figure S1. No significant differences were observed between ERPs ($p > 0.05$, FDR corrected).

These results suggest that any preprocessing approach introduced no important biases. While APICE(3) shows a slightly higher amplitude ERPs than APICE(3)+W-ICA and APICE(3)+DSS, the effect is very small in magnitude. The effect is probably due to a slight reduction of the neural response due to the cleaning methods applied to the data. The fact that it reaches significance is not surprising given that the pipelines APICE(3)+W-ICA and APICE(3)+DSS only differ from APICE(3) in the addition of ICA/DSS. These methods reduce the variance in the data and therefore tend to consistently decrease the signal's amplitude across subjects due to some reduction of the neural signal (Haresign, et al., 2021).

MADE using different thresholds for rejection

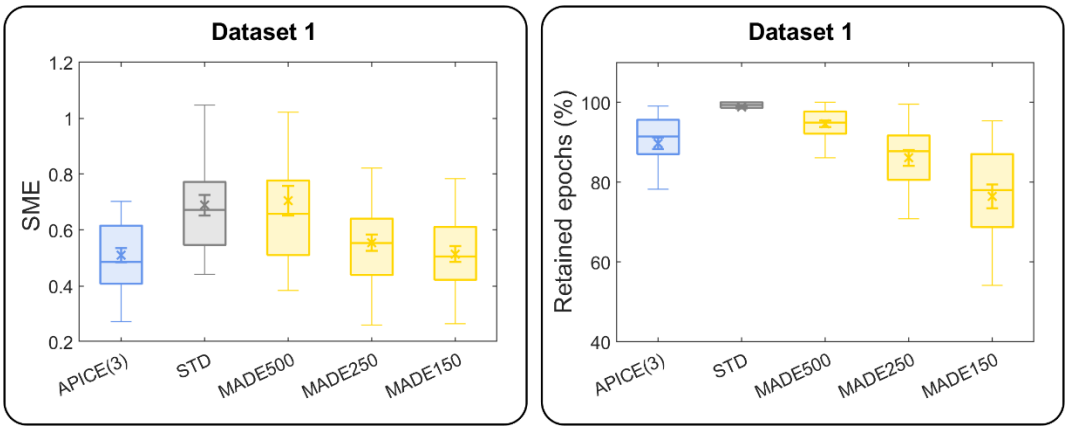

Figure S4. Comparison of APICE's performance with the Standard pipeline (STD) and the MADE pipeline using different thresholds for rejection (500 µV, 200 µV, 150 µV) for dataset 1. The boxplot shows the median, 25 and 75 percentiles, and the whiskers 1.5 interquartile ranges. The cross shows the mean and the error bar the standard error. **(A)** SME for dataset 1. **(B)** Retained epochs for dataset 1.

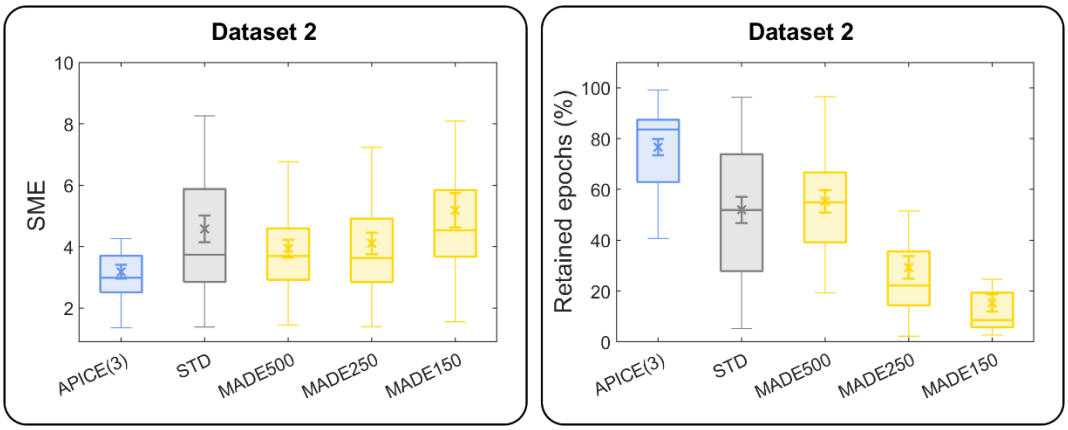

Figure S5. Comparison of APICE's performance with the Standard pipeline (STD) and the MADE pipeline using different thresholds for rejection (500 µV, 200 µV, 150 µV) for dataset 2. The boxplot shows the median, 25 and 75 percentiles, and the whiskers 1.5 interquartile ranges. The cross shows the mean and the error bar the standard error. **(A)** SME for dataset 1. **(B)** Retained epochs for dataset 1.

References

Haresign, I. M., Phillips, E., Whitehorn, M., Noreika, V., Jones, E. J. H., Leong, V., & Wass, S. V. (2021). *Automatic classification of ICA components from infant EEG using MARA* [Preprint]. Neuroscience. https://doi.org/10.1101/2021.01.22.427809