Robin's Viewer: Using Deep-Learning Predictions to Assist EEG Annotation

Robin Weiler[1], Marina Diachenko[1], Erika Juarez-Martinez[1], Arthur-Ervin Avramiea[1], Peter Bloem[2], Klaus Linkenkaer-Hansen[1]

[1]Department of Integrative Neurophysiology, Center for Neurogenomics and Cognitive Research (CNCR), Amsterdam Neuroscience, VU Amsterdam, Amsterdam, Netherlands

[2]Department of Computer Science, VU Amsterdam, Amsterdam, The Netherlands

*** Correspondence:**
Klaus Linkenkaer-Hansen
klaus.linkenkaer@cncr.vu.nl

Keywords: EEG, Viewer, Deep-learning, Artifacts, Open source, Python

Abstract

Deep learning has been successfully applied to EEG data for sleep staging, epileptic seizure detection, and artifact recognition. However, the performance of automated annotation is not yet sufficient to replace trained annotators in a clinical setting. Therefore, we propose a decision-support-system to help human annotators work faster and more effectively. As a first step towards addressing these challenges, we developed Robin's Viewer (RV) in Python, building on top of the plotting library Plotly and the popular M/EEG analysis toolbox MNE. The goal was to create a platform-independent, interactive web app, which is open-source and supports many common EEG-file formats to facilitate easy integration with a variety of EEG toolboxes. RV includes many common features of other EEG viewers, e.g., a view-slider, marking of bad channels and transient artifacts, and customizable preprocessing. The key feature distinguishing RV from existing EEG viewers is the possibility to visualize output predictions of deep-learning models that are trained to recognize patterns in EEG data. The result is a decision-support-system for scientists and clinicians who can use RV to annotate artifacts, sleep stages, abnormalities, and other classification tasks.

1 Introduction

Electroencephalography (EEG) is the most widely used method for measuring brain activity in humans (Niedermeyer and da Silva, 2005; Biasiucci et al., 2019). EEG measures neural oscillations, which are associated with the synchronous firing of neurons and correlate with brain functions such as perception, motor control, cognition, and states of arousal (Connors et al., 1997). EEG measurements are frequently performed in research of epilepsy (Smith, 2005; Acharya et al., 2013), sleep disorders (Diaz et al., 2016), mental disorders (Arns et al., 2008), and brain-computer interfaces (BCI) (Wolpaw et al., 1991). Unfortunately, EEG recordings also capture many non-neuronal artifact signals, such as muscle activity, eye movements, blinks, and heartbeats, but also non-biological ones, such as electrode movement, changing impedance, and interference from other electronic devices (Barua and Begum, 2014; Radüntz et al., 2015; Yang et al., 2018). While non-biological signals can often be filtered out using band-pass filters, biological artifacts can exist in frequency spectra that overlap with those of the neural oscillations scientists are trying to measure (Connors et al., 1997). Artifacts can be difficult to distinguish from abnormal EEG, for example in epilepsy patients, where

the latter is relevant for understanding the disorder. For these reasons, artifacts have to be removed prior to analyzing and interpreting EEG data. This process can be very time consuming, requires experienced neurologists, and introduces a certain degree of subjectivity (Biasiucci et al., 2019).

Given the need of neurologists to analyze large amounts of data, efforts have increased towards using various machine learning techniques for handling artifacts in EEG data (Shoker et al. 2005; Shao et al. 2008; Barua and Begum, 2014; Pardede et al., 2015; Radüntz et al., 2015; Yang et al., 2016, 2018; Sai et al., 2017; Wang et al, 2017; Craik et al., 2019; Roy et al., 2019; Sun et al., 2020; Boudaya et al., 2022; Jurczak et al., 2022; Liu et al., 2022; Diachenko, M., Houtman, S. J., Juarez-Martinez, E. L., Ramautar, J. R., Weiler, R., Mansvelder, H. D., Bruining, H., Bloem, P., & Linkenkaer-Hansen K. (2022). Improved manual annotation of EEG signals through convolutional neural network guidance. eNeuro. Unpublished manuscript.). Examples of machine learning techniques applied in artifact detection are support vector machines (SVMs) (Shoker et al. 2005; Shao et al. 2008; Bartels et al., 2010; Sai et al., 2017), k-nearest neighbor classifiers (k-NN) (Gao et al., 2010), and independent component analysis (ICA) (Barua and Begum, 2014; Radüntz et al., 2015). Recently, efforts have focused on developing deep-learning solutions for artifact detection (Pardede et al., 2015; Craik et al., 2019; Roy et al., 2019; Sun et al., 2020; Boudaya et al., 2022; Jurczak et al., 2022; Liu et al., 2022; Diachenko, M., Houtman, S. J., Juarez-Martinez, E. L., Ramautar, J. R., Weiler, R., Mansvelder, H. D., Bruining, H., Bloem, P., & Linkenkaer-Hansen K. (2022). Improved manual annotation of EEG signals through convolutional neural network guidance. eNeuro. Unpublished manuscript.), which are capable of learning very complex functions between raw input data and the corresponding output using a composition of several non-linear modules (LeCun et al., 2015).

However, even though there is an increasing amount of research using deep-learning models for EEG analysis, the performance of automated annotation is not yet sufficient to replace trained annotators in a clinical setting, and when artifact removal is completely automated, it does not offer neurologists, and potentially non-expert users, such as students, the opportunity to inspect the models' output and re-evaluate potential false predictions.

To this end, we developed an interactive web-application called Robin's Viewer (RV), a user-friendly EEG viewer with the main purpose of annotating time-series EEG data using deep-learning models as a decision-support-system. RV is open-source (MIT license) and supports many common EEG-file formats, allowing for it to be integrated into data-processing pipelines developed for a variety of EEG toolboxes. Since RV runs in the user's browser, it is also platform independent. RV was developed in Python and builds on the popular M/EEG analysis toolbox MNE (Gramfort et al., 2013), Plotly[1]: a Python-based plotting library, and Dash[2]: used for the interactive GUI and providing a locally hosted server to run the application. In RV, the user can load any EEG recording, apply common preprocessing methods (e.g., bandpass filtering, re-referencing, and downsampling), plot the data, use the GUI to navigate through the signal, and mark bad channels and artifacts. Our priority was to make EEG viewing and annotation as comfortable and efficient as possible for a wide variety of users. For this reason, we developed RV in close collaboration with machine-learning experts, neurologists, and non-expert users (e.g., students).

One of the key features distinguishing RV from conventional EEG viewers is the possibility to visualize output predictions of deep-learning models that are trained to recognize patterns in EEG

[1] https://github.com/plotly/plotly.py

[2] https://dash.plotly.com

data, and which assist users in marking artifacts. We included with RV a proof-of-concept deep-learning model trained to recognize artifacts in resting-state, multichannel EEG recordings (Diachenko, M., Houtman, S. J., Juarez-Martinez, E. L., Ramautar, J. R., Weiler, R., Mansvelder, H. D., Bruining, H., Bloem, P., & Linkenkaer-Hansen K. (2022). Improved manual annotation of EEG signals through convolutional neural network guidance. eNeuro. Unpublished manuscript.).

Ultimately, with this application we hope to combine the predictive power of deep-learning models and the knowledge of subject-matter experts such as neurologists, and hence, we invite especially the former to integrate their own models into RV as explained later. RV can be downloaded for free at https://github.com/RobinWeiler/RV.git, where we also provide detailed documentation and user manuals, with demos and example data. The purpose of this article is to give an overview of the design and key features of RV using artifact annotation as the running-example.

2 Methods

2.1 Design Requirements

Two main objectives were considered in the development of RV. The first was to provide an efficient and user-friendly EEG annotation interface that includes existing functionalities provided by other EEG viewers, i.e., supporting common EEG datafile formats for loading and saving, popular preprocessing options, bad channel marking, and artifact annotation. Among the preprocessing options, RV should include bandpass filtering, automatic (and manual) bad channel detection and interpolation, signal re-referencing and re-sampling. The second objective was to integrate deep-learning models trained for various classification tasks into the viewer to assist human annotations. To accomplish this, we defined the following design requirements.

First, RV should be able to plot EEG data of variable length with up to 128 channels. Moreover, to optimize the viewer experience, RV should allow for intuitive and user-friendly navigation of the EEG signal. For this, we considered a view-slider to scroll through the data in adjustable intervals, zooming buttons for both time and channel axes, and the ability to select specific channels for viewing.

Second, artifact annotation should be possible via selection of data in the plot using the mouse or trackpad (the interval selected can be highlighted across all channels simultaneously due to the high correlation between EEG channels). The user should be able to easily adjust the artifact annotations by expanding or narrowing the corresponding interval selection on the plot, or altogether erasing the annotation.

Other required features for the viewer are the ability to measure both the amplitude (in µV) of the signal at a specific time point, as well as the predominant frequency (in Hz) within a selected time range. Additionally, the user should have access to summary information regarding the total length of intervals selected as artifacts, the amount of clean data, and other statistics related to artifact removal.

RV should integrate with other modern EEG toolboxes, allowing the users to inspect and annotate their data in RV and continue working with the annotated data in their toolbox of choice.

The feature that distinguishes RV from other viewers is the integration of output from deep-learning models with EEG annotations. In this regard, RV should provide a clear and effective visualization of model predictions, allowing the user to inspect the models' confidence that certain segments are clean or artifacts. With the support of the deep-learning model, the user will be able to annotate the

EEG recordings faster and more efficiently. RV should include at least one pre-trained deep learning model as a proof-of-concept. Finally, the users should be able to construct, train, and assess their own models for the purpose of EEG artifact recognition, and load their models in RV.

2.2 Programming Environment

We developed RV in Python and built it around the following main libraries: Plotly, Dash, and MNE. A full list of all the required libraries can be found in the "requirements.txt" file included on GitHub. Because of Python's portability and the fact that Plotly and Dash allow RV to be served in the user's web browser, our application is largely platform independent. We tested RV on the three major platforms MacOS, Windows, and (Ubuntu) Linux, as well as the popular web browsers Safari, Google Chrome, and Mozilla Firefox.

All of the plots and visualizations, as well as parts of the interactive GUI, including the slider and artifact annotation mechanism, included in RV, were written using Plotly. We used Dash to deploy and host the application in the user's web browser. Most of the buttons and other interactive elements of RV were also developed in Dash using the integrated callbacks.

MNE (Gramfort et al., 2013) is a popular M/EEG analysis toolbox built in Python, which features a wide variety of common techniques applied to EEG data. We used the data structures provided by MNE as the main underlying data structures for RV. All the preprocessing applied by RV relied on MNE code, including automatic bad-channel detection, which was implemented using Autoreject (Jas et al., 2017) and which is also based on MNE. However, when needed, the user can pre-process their data outside of RV and then plot it directly using the techniques outlined in Section 3.8, skipping the pre-processing steps applied by RV.

2.3 Deep-Learning Model

A convolutional neural network was trained on expert-annotated resting-state EEG data from typically developing children and children with neurodevelopmental disorders, with the purpose of recognizing artifacts (Diachenko, M., Houtman, S. J., Juarez-Martinez, E. L., Ramautar, J. R., Weiler, R., Mansvelder, H. D., Bruining, H., Bloem, P., & Linkenkaer-Hansen K. (2022). Improved manual annotation of EEG signals through convolutional neural network guidance. eNeuro. Unpublished manuscript.). This model was integrated into RV as a proof-of-concept decision-support system for EEG artifact annotation. The model reached 80% balanced accuracy (a performance metric based on the average of sensitivity and specificity which is used to assess accuracy in imbalanced data, i.e., where one target class appears more often than the other) in distinguishing artifact from non-artifact EEG patterns and, importantly, detected valid artifacts missed by the trained annotator. The study by Diachenko et al. (2022) also showed that the model could potentially be improved with more- and better-annotated data. In this regard, RV does not only increase the quality and speed of manual annotation of EEG artifacts, but, vice versa, it can also serve to improve deep-learning models used for EEG analysis in the long run, when it is used to produce more high-quality annotated EEG data for future model training.

RV also allows the model to be used for automatic artifact annotation, i.e., it can automatically annotate portions of the signal predicted with confidence that exceeds a customizable threshold, which will be explained in more detail in Section 3.5. For the data where the model confidence does not cross the fixed threshold, the user has to make the final decision.

3 Results

In the following, we give an overview of the design and key features of RV and its graphical user interface (GUI) using artifact annotation as the running-example.

3.1 Loading EEG Data

The initial screen of RV consists of a pop-up window where the user can load EEG data and make use of the preprocessing and visualization methods provided by the application. This window can be seen in Figure 2 and can always be reopened when closed by using the "Preprocessing" button in the top-left corner of the menu bar (see Section 3.3.1.).

RV follows the following file structure: There is the "data" directory, from which files can be loaded but not overwritten. This is done to ensure that the user never accidentally overwrites their raw EEG-data files. Then, there is the "save_files" directory, which is where all files annotated in RV will be saved, and where files can be overwritten when the user explicitly chooses to do so in the "Save to" pop-up window (more information on RV's saving functionality in Section 3.7.). Additionally, there is the "temp_raw.fif" file located in the main RV directory, which is a temporary working save-file updated automatically with every annotation made to the data. All files located in both the "data" and "save_files" directory, as well as the "temp_raw.fif" file, can be loaded in RV as this is where RV has access to reading files. Currently, the following file-formats are supported for loading: .edf, .bdf, .fif, and .set. However, note that the user can also choose to load the data outside of RV (e.g., to load a file-format not currently supported) and pass it as a MNE Raw data-structure to the "run_viewer" function (explained in more detail in Section 3.8.).

After loading EEG data, the user can use the "Stats" button in order to view some selected statistics surrounding the annotated data, which allows for a quick overview over the data before plotting (Figure 3). Currently, the integrated statistics consist of the recording's total length (in seconds), the amount of annotated and non-annotated data (in seconds), as well as the amount of consecutive, non-annotated intervals longer than two seconds (this is useful because, if most of the remaining clean data segments are shorter than two seconds, the recording's quality is likely not sufficient for further analysis). Furthermore, we included a histogram to show the distribution of the length of non-annotated data segments. These statistics are intended to give the annotator a feeling for the quality of the data and whether this data can be used for further analysis. We invite researchers to further extend this list of statistics, or to create a feature request on GitHub.

3.2 Preprocessing and Visualization Settings

RV offers a handful of popular EEG preprocessing methods within the file-loading window seen in Figure 2, which rely on the MNE package (Gramfort et al., 2013). These include bandpass filtering, re-referencing, bad channel interpolation, and automatic bad-channel detection, which, as mentioned in Section 2.2., is based on the Autoreject library (Jas et al., 2017). The flowchart in Figure 4 shows the order in which preprocessing and visualization settings are applied to the data.

For the high- and low-pass filter, the user needs to provide the values (in Hz) in the respective input fields. The bandpass filter is a FIR filter with a Blackman window (more filters will be added in future versions of RV), implemented in MNE. If the loaded EEG data has previously been preprocessed (because it is a save-file or because it was preprocessed outside of RV), those respective filter values will be extracted and filled in automatically. Then, if no changes are made to these values, or fields are empty, the data will be left in its raw state.

Below the bandpass filter, the user can choose from average-, Cz-, and no custom reference via the respective dropdown-menu.

For automatic bad-channel detection, there two options: skip the step altogether ("None"), or use automatic bad-channel detection ("Autoreject"). The latter was implemented using Autoreject (Jas et al., 2017). More methods for automatic bad-channel detection will be added in future versions of RV and, of course, bad channels can also be marked manually in RV, which will be explained in Section 3.4.1.

Interpolation of bad channels can be activated in this window as well, and is performed when there are marked bad channels (also if the automatic bad-channel detection algorithm marks them) and when the loaded file contains channel-position information. If the latter is not provided, RV will display an error message.

Aside from the preprocessing options, there are several settings which only affect the visualization and are not reflected in the save-file when saved. First, the user can provide a resampling frequency (in Hz). By default, this is initialized with the original sampling frequency of the loaded data. We highly recommend downsampling the data before plotting, as this will have a big impact on the overall performance and usability of RV. This resampling, as mentioned, will only be used for plotting and does not affect the sampling frequency of the data saved in the end.

Another option that affects the visualization of the signal is the custom scaling factor, which is the scaling the data should have in relation to volts, the default unit used by MNE. By default, this is equal to 1e-6 as the EEG data is scaled from volts to microvolts for easier plotting. However, when loading source-reconstructed data, for example, one should put 1e-12 in this field so that RV can scale the data accordingly. Additionally, there is an input field allowing the user to change the space between channels for plotting, called the "Channel-offset". By default, this is equal to 40 μV, which allows enough space between individual channels while still spreading them out as much as possible to preserve detail in the traces. Entering 0 here will collapse all channels on top of each other (called butterfly mode), which can be useful to identify potential bad channels with high variance, while numbers bigger than 40 will spread the channels further apart, requiring the user to scroll up and down by clicking and dragging the channel-axis labels.

Another visualization option that significantly improves RV's performance and responsiveness is the ability to load and visualize the recording in segments of a specified length (in seconds). When using this option, the user can move between the segments using the left- and right-arrow buttons in the menu bar of the GUI (shown in the following section), just below the "Preprocessing" button. When the recording is segmented in this way, fewer datapoints are plotted at once, compared to the case where the entire signal is loaded. By default, the recording is not segmented.

Underneath the input field to set the segment size, there is an option to enable/disable the adjustable view-slider, which can be used to scroll through the recording (or segment) continuously, i.e., without switching between segments, and which will be described in more detail in Section 3.3.4. However, the view-slider cannot be used to improve RV's performance as the rest of the data outside of the timeframe shown by the view-slider still has to be preloaded and plotted in the background. By default the view-slider is active and initialized to span a ten-second timeframe.

Finally, by pressing the "Select channels" button, the user can select the channels to be plotted (Figure 5). Channels can be selected from a sensor-topography plot (if the loaded data-file contains

channel-position information) or by choosing specific channel names to be plotted in a dropdown-menu. If no selection is made, all channels will be plotted.

Pressing the "Plot" button at the bottom will close the "Preprocessing" window and, after a brief period in which the signal is loaded, open up the main GUI where the preprocessed EEG data will be displayed. The "Preprocessing" window can be reopened at any time using the "Preprocessing" button in the menubar of the application (Section 3.3.1.), and all the preprocessing and visualization settings can be changed. Then, clicking the "Plot" button will reset the current view and hence all annotations and marked bad channels. This is because changing the bandpass filter settings can influence the timing of artifacts. Therefore, all relevant data has to be saved before changing these settings and re-plotting.

3.3 Graphical User Interface (GUI)

The GUI of RV is divided into six components, as indicated in Figure 6, each of which will be explained in detail in the following subsections.

3.3.1 Menu Bar

The menu bar is located at the top of the screen. Its primary function is to contain buttons to open different pop-up windows for file loading and preprocessing, saving, statistics, power spectrum, help, and quitting, but it also hosts a dropdown-menu to mark bad channels, as well as left and right arrow buttons used to navigate the currently viewed timeframe if the data is loaded in segments. The "Preprocessing" button opens the file-selection and preprocessing window mentioned in Section "Loading EEG Data". The "Save to" button opens up a pop-up window allowing the user to save the annotated data (explained in Section 3.7.).

In the middle of the menu bar, the bad-channel selection dropdown-menu allows the user to select bad channels out of a list of all available EEG channels. For more information on bad-channel marking, see Section 3.4. Next to the dropdown-menu, there is the "Redraw" button, which, as the name suggests, redraws the plot. This button should be used after marking all bad channels in order to update the plot with the newly marked bad channels. The updated plot will highlight newly marked bad channels with gray. Additionally, once the "Redraw" button is clicked, the built-in deep-learning model, if activated (see Section 3.5.2.), will recalculate its predictions while considering the newly marked bad channels. Note that, in contrast to the "Plot" button described in Section 3.2., the "Redraw" button does not reset currently drawn artifacts.

The "Stats" button opens the same pop-up window described in Section 3.1. and shown in Figure 3, containing statistics surrounding the annotated data. Next to the "Stats" button, there is the "Power Spectrum" button, which opens a pop-up window showing the most prominent frequency, as well as the respective power spectrum of a segment of the plotted EEG data previously selected. SciPy's (Virtanen et al., 2020) implementation of the Welch method (Welch and Peter, 1967) is used for the power spectrum generation. How to select such a segment will be explained in the following Section 3.3.2. - button 4.

Finally, on the top-right, there are two buttons grouped together. On the left, the "Help" button opens a pop-up window containing links to the documentation and user manual. On the right, the "Quit" button displays a pop-up window asking for confirmation to shut down RV, shuts down the local server RV is running on, and after this the RV interface cannot be used anymore.

3.3.2 Taskbar

The taskbar, seen in Figure 6, allows the user to interact with the EEG plot. From left to right, there are the following buttons:

1. Take a picture of the (annotated) EEG signal and download it as a .png file.

2. Select an area to zoom in on (click-and-drag).

3. Move view along the channel- or time-axis, or both simultaneously (click-and-drag).

4. Select a segment of the data for which to calculate and display the main underlying frequency, as well as a power spectrum in "Power-spectrum" pop-up window, which can be reopened using the "Power spectrum" button in the menu bar (click-and-drag). It is possible to select all channels or only a few desired ones. The latter can be facilitated via the legend explained in Section 3.3.5.

5. (Activated by default) Select a segment of the plot to annotate, which creates a semi-transparent red box spanning the entire vertical axis in view (currently only annotations across all channels are supported) across the selected time interval (click-and-drag). These annotations can be freely adjusted in their position and size, or removed entirely (with button 6.), when clicked on.

6. Delete the currently selected annotation.

7. Zoom in.

8. Zoom out.

9. Zoom out as much as necessary to show all channels for the entire duration of the recording (or segment), including all peaks in the data (potentially produced by artifacts).

10. Display a ruler from both axes to the datapoint currently hovered on.

3.3.3 Labeled Buttons

By default, there are three labeled buttons in the top-left of the plot. The first two buttons are used to reset the view, either across the channel- ("Reset channel-axis") or time-axis ("Reset time-axis"). The former will reset the view to fit all channels in the plot while the latter will reset the plotted timeframe to show the initial configuration, e.g., the first ten seconds of the recording (or segment) when the 10-second view-slider is activated.

Furthermore, there is the button to hide marked bad channels from the plot ("Hide/show bad channels"). Once bad channels have been hidden, when this button is clicked again, bad channels are once again visible. It is important to note that, after marking bad channels, the user will first have to use the "Redraw" button in the menu bar in order for the changes to have an effect, and for the "Hide/show bad channels" button to work accordingly. Bad channel marking is explained in more detail in Section 3.4.

An additional fourth button, called "Highlight model-channels", appears only when the integrated deep-learning model is activated. This button marks all channels the model used to generate its predictions in blue. These channels are determined by the model's implementation and cannot be changed. However, knowing which channels the model considered can be helpful in understanding

its predictions, e.g., if an artifact was not marked by the model because it occurred on a channel that was not part of its input.

3.3.4 View-Slider

The view-slider, once activated in the "Preprocessing" window (Figure 2), is located at the bottom of the screen (Figure 6), and can be used to continuously scroll through the recording (horizontally along the time-axis) by clicking and dragging it. The slider's range, by default initialized to ten seconds, can be freely adjusted from showing the entire recording (or segment) at once, down to merely several milliseconds, by clicking and dragging the small rectangles at its edges. It should be noted that the same functionality as the view-slider can also be achieved by clicking and dragging on the channel- or time-axis' labels, or selecting the third button in the taskbar (see Figure 6) and then clicking and dragging on the plot directly.

3.3.5 Legend

At the right side of the plot, there is a scrollable legend showing the names of all available EEG channels (Figure 6). Clicking on any channel name hides the channel from the plot. Double-clicking hides all channels except for the selected one, and the user can follow up by adding more channels to display by clicking their names once. Hiding all but a few channels is advised to retrieve the most prominent underlying frequency and power spectrum of a segment mentioned in Section 3.3.1., if it is desired to only include specific channels, as this will make it easier to only select the desired channels in an interval.

3.3.6 Plot

As can be seen in Figure 6, the majority of the screen is occupied by the plot which dynamically adapts its size based on RV's window size to fill as much space as possible. All selected channels are spread across the vertical axis with an offset of 40 μV between them by default, unless specified otherwise by the user in the visualization settings. Time, on the horizontal axis, is displayed in seconds. If the view-slider is enabled, the first ten seconds of the recording (or segment) are displayed across all selected channels. Otherwise, the whole recording (or segment) is shown.

The EEG channels are plotted in black, with the exception of bad channels, which are shown in gray. The user can hover over any given point in the plot in order to display the time (in seconds) and amplitude (in μV if no custom scaling was used) values, rounded to three decimals, of the selected channel. The deep-learning predictions, if activated, are plotted below the EEG channels and will be explained in more detail in Section "Deep-Learning Model Predictions". Figure 7A shows an example plot with several annotated artifacts and with the integrated deep-learning model activated.

3.4 Bad-Channel Marking

Bad channels are EEG electrodes which did not record actual brain activity, for example because they were disconnected from the patient's scalp, and thus cannot be used for further analysis and interpretation. In RV, we offer two possibilities to mark bad channels: The manual way of inspecting the EEG data first and then marking the bad channels through the respective dropdown-menu in the menu bar (Figure 6), and the automatic method selectable in the "Preprocessing" pop-up window (Figure 2) implemented using the Autoreject library (Jas et al., 2017).

3.4.1 Manual Bad-Channel Marking

In the menubar of the application (Figure 6), users can select channels to mark them as "bad" using a dropdown-menu. When this dropdown-menu is clicked, all available EEG channels are listed for selection. The user can also type the beginning of a channel name in the dropdown-menu text box, which will narrow down the option selection. Note that selecting bad channels does not update the plot automatically, because it takes a while to redraw the signals. For this reason, the "Redraw" button in the menubar (Figure 6) must be used after marking all bad channels. The updated plot will then highlight newly marked bad channels with gray. Once at least one bad channel has been marked, and the plot has been redrawn, the "Hide/show bad channels" is enabled, from the section "Labeled buttons" of Figure 6. This button allows for bad channels to be hidden from the plot, as explained in Section 3.3.3.

3.4.2 Automatic Bad-Channel Detection

Automatic bad-channel detection is performed using the Autoreject library (Jas et al., 2017) and can be activated in the "Preprocessing" pop-up window (Figure 2). When this method is used, first, an EEG recording is cut into 1-second segments with an overlap of 50%. Then, a cross-validation algorithm is applied to the segments to identify "good" and "bad" channels. Then, for each channel, the ratio of segments for which that channel is "bad" is calculated. The channel is called "bad" if the ratio is more than or equal to 0.8 and will be selected in the bad-channel dropdown-menu in the menu bar seen in Figure 6. Note that, for future versions of RV, we are planning to add more automatic bad-channel detection methods.

3.5 Deep-Learning Model Predictions

There are two possibilities for visualizing deep-learning model predictions within RV, both of which will be explained below. The first option is to load predictions from a file; the second option is to run the integrated deep-learning model on the currently loaded EEG recording in RV. Once loaded, predictions can be used to automatically annotate the data. Note that RV can also be used without the deep-learning model, if desired.

3.5.1 Loading Predictions from Files

The first option for displaying predictions in RV is for the user to run the deep-learning model of choice over each EEG recording individually with their own scripts outside of RV, saving the predictions (scaled between zero and one) in separate files, and then loading them in RV for each recording. Note that there must be one prediction per timepoint in the raw recording. RV assumes that the loaded predictions have the same sampling frequency, and hence align with, the loaded raw EEG data (i.e., without the downsampling applied). If this is not the case, the predictions will not be scaled correctly across the time-axis. Note that if the deep-learning model is integrated into RV and run accordingly (see following Section 3.5.2.), predictions can be returned along with the corresponding sampling frequency, allowing for custom scaling along the time-axis.

Loading prediction files is very similar to loading EEG data (described in detail in Section 3.1.): After generating the model-prediction files, they must be placed in the "data" directory. To then load these files, the user can either drag-and-drop them into the "Drag-and-drop or click here to select model output" area in the "Preprocessing" pop-up window seen in Figure 2, or click on said area, upon which a file selection window will open and allow to select the file to load (the selected file will still have to be inside the "data" directory in order to be loadable). In contrast to loading an EEG recording, however, multiple prediction files can be loaded, allowing the user to compare several

models simultaneously (see Section 3.5.3 for more detail). Currently supported file formats for prediction files are .npy, .txt, and .csv. Note that because there must be one prediction per timepoint in the raw EEG recording, the files must have only one column and each row must correspond to one prediction. The .txt and .csv files must not have a header.

After loading a new EEG recording, or when wanting to remove the predictions from the plot, one can use the "Remove model predictions" button underneath the dashed file selection area.

3.5.2 Running the Integrated Deep-Learning Model

Underneath the "Remove model predictions" button in Figure 2, there is an option to run the integrated deep-learning model defined in the "run_model.py" file. The model that is currently included was pre-trained to recognize artifacts (Diachenko, M., Houtman, S. J., Juarez-Martinez, E. L., Ramautar, J. R., Weiler, R., Mansvelder, H. D., Bruining, H., Bloem, P., & Linkenkaer-Hansen K. (2022). Improved manual annotation of EEG signals through convolutional neural network guidance. eNeuro. Unpublished manuscript.). It is important to note that the model will be passed both the preprocessed- and the raw EEG data. This is done to allow for custom preprocessing of the model's input, independent of the preprocessing applied to the viewed data. In the case of the integrated model by Diachenko et al., for example, this is necessary because this model specifically requires time-frequency plots generated from segments of the time-series EEG data as input. Note that, because the model is sensitive to bad channels, they should be marked before running the model and evaluating the predictions.

If the user wants to implement their own model, it should follow the following guidelines: Input is the preprocessed- and raw EEG data; output is a one-dimensional array, each prediction scaled between zero and one, as well as the list of channel names that the predictions are based on, if applicable, and the sampling rate of the predictions, in order for RV to correctly match the predictions to the time-axis. For more information on how to implement a custom model, please refer to the docstrings in the "run_model.py" file or the documentation on GitHub.

3.5.3 Predictions Visualization

As mentioned previously, the deep-learning models' predictions are plotted below the EEG channels (Figure 7A), and are colored according to a colorbar (Figure 7B), where the color changes from blue to white to red for predictions ranging from 0 to 0.5 to 1, respectively. Since the integrated deep-learning model was trained to recognize artifacts, in this case, 0 (blue) corresponds to a high likelihood of clean data, while 1 (red) indicates a high likelihood for an artifact being present. When hovering over the predictions, RV shows the time (in seconds) and the corresponding prediction value. In the legend and on the y-axis, if more than one model output is loaded via the methods described above, the prediction channels will be named "M0", "M1", etc. where, if the user loaded previously generated predictions, they will be in the order of the list displayed under "Selected model output" in the "Preprocessing" window (Figure 2). If the user ran the integrated model on the data, it will always be the bottom one.

3.6 Annotations

In RV, annotations can be made manually using the GUI or automatically using the integrated (or loaded) deep-learning model predictions. Annotations are saved in the "annotations" attribute of the saved MNE Raw data-structure, each with its own onset-time, duration, and, because we are primarily focused on artifact annotation for this version of RV, with the description "bad_artifact". The user will be able to set the description of the annotation in the GUI in a future version of RV.

3.6.1 Manual Annotation

As mentioned in Section 3.3.2., annotations can be made when the (5.) button in the taskbar (see Figure 6) is activated, which it is by default upon loading. While this mode is active, the user can select a segment of the plot to annotate by clicking and dragging on the plot, which creates an opaque red box spanning the entire vertical axis in view across the selected time interval. It should be noted that, currently, only annotations across all channels are supported. Annotations can be freely adjusted in their position and size when selected: The selected annotation will be highlighted and its corners can be dragged in order to adjust its size. To move it, the user can click anywhere on the annotation, and, after having selected it, drag it to the desired position. If the user wants to delete the annotation, they first have to select it and then click the (sixth) button in the taskbar which deletes the currently selected annotation. Note that, if annotations are overlapping, they will be merged in the underlying data structure and will also be displayed merged when the plot was redrawn.

3.6.2 Automatic Annotation

It is possible to let the integrated deep-learning model create annotations automatically according to a customizable threshold specified in the "Preprocessing" pop-up window (see Section 3.2. and Figure 2). Then, for every datapoint where the supplied model outputs a prediction higher than the given threshold, an annotation will be drawn. These automatic annotations will be given the same description as the manually created ones ("bad_artifact").

3.7 Saving Annotated Data

As mentioned in Section 3.1., RV always keeps one temporary working save-file, which is updated with every annotation made to the data. This file can be loaded in case RV is accidentally closed before saving. This automatic save-file is called "temp_raw.fif" and is located in the main RV directory. To properly save data, the user has to use the "Save to" button in the menu bar (see Figure 6), which will open a small pop-up window giving the option to either save the data with a given filename and the .fif file-extension, or overwrite the loaded file (the latter only being possible if the loaded file is a save-file in the "save_files" directory; files in the "data" directory cannot be overwritten as explained in Section 3.1.). RV uses the saving functionality of MNE, hence we currently only support saving to the native MNE .fif format as there is a risk of information loss when saving to external formats. For future versions of RV, we will investigate possibilities to export to other EEG-file formats, e.g., .edf and .set. Once files are saved in the "save-files" directory, they can also be loaded through the same steps as outlined in Section 3.1. If a save-file already exists with the same name, it will be saved with an increasing integer added to the end of the name, unless the user chooses the option to overwrite the loaded save-file. When choosing to overwrite the loaded save-file, an additional confirmation pop-up window opens.

To avoid the situation where, e.g., high-frequency artifacts are filtered away during viewing but not when subsequently applying quantitative analysis algorithms, the save-file will contain the preprocessed data after applying bandpass filters and setting the custom reference. Visualization settings, such as downsampling, will not be included as these are exclusively used for plotting.

3.8 Integration into EEG Toolboxes

Since RV supports many common EEG-file formats for loading, it can easily be integrated into data-processing pipelines developed with different EEG toolboxes. Aside from loading data in RV and running it through the integrated preprocessing pipeline (Figure 4), the user can preprocess their data in a script outside of RV before loading it, or pass it to the "run_viewer" function as a MNE Raw

data-structure. The latter allows to circumvent the restriction of only loading and saving data in the "data" and "save_files" directories, respectively (for more details, please refer to the documentation on GitHub). Then, because preprocessing has been done externally, all the preprocessing steps (except the visualization settings) within RV can be skipped (by not changing the settings' parameters), and the data can be visualized and annotated immediately. This way, the user merely has to load the afterwards saved file in their own scripts again to continue their custom pipeline.

In principle this would even be possible using toolboxes written in programming languages other than Python, for example MATLAB, but caution is advised when switching between languages as we did not test this thoroughly.

4 Discussion

We developed an open-source EEG viewer that facilitates EEG annotation with the optional support of deep-learning model predictions. RV offers common EEG viewer functionalities with an emphasis on tools surrounding artifact annotation. The integration of deep-learning model predictions into RV facilitates an efficient decision-support system for neurologists. An important advantage of RV is that it can be easily integrated into pipelines developed with other EEG toolboxes. Moreover, because RV is programmed in Python, and it gets served in the user's web browser, RV can be run on any operating system.

The main limitation of RV lies in its performance, specifically if neither downsampling nor segmentation is used, as plotting the entire signal can burden memory significantly and result in a long initial loading time and slow-responding GUI. Therefore, for EEG recordings with a high number of channels, or for long EEG recordings, segmenting the data (resulting in smaller amounts of data being loaded into memory and plotted at one given time) and downsampling (which reduces the amount of datapoints plotted overall) becomes indispensable. Another limitation is that the current version of RV was developed for binary classification (artifact or clean data), and cannot handle multi-class deep-learning predictions yet. However, this will be revised in a future version of RV. Additionally, there is currently no possibility to pass labels to annotations: all annotations have "bad_artifact" as their saved description unless specified otherwise in the source code.

This paper marks the beginning of Robin's Viewer Version 1.0. Many additional features and improvements are currently in development for future versions. The main working points are the option to add labels to annotations to support multi-class deep-learning predictions and annotations, as well as functionality to interact with the RV GUI using keyboard keys and combinations of keys. Once the implemented deep-learning model is accurate enough, we will also consider adding an option where the user is presented only with segments for which the model's confidence stayed under a certain threshold to decide between artifact and non-artifact, e.g., predictions between 0.3 and 0.7, which would speed up the annotation process further. Additionally, we want to reduce the computational cost of RV and improve its speed. In this regard, we are investigating a more sophisticated preloading technique where the data outside of the currently plotted timeframe is loaded on the fly as the user is using the view-slider to scroll through the recording, or moving RV to a remote server, which would offload most of the computations from the client- to the server-side.

Finally, we invite other Python programmers, deep-learning researchers, and neurologists alike to collaborate and build on RV, according to their requirements, or to make feature requests on GitHub[3].

5 Conflict of Interest

KL-H is a shareholder of Aspect Neuroprofiles BV, which develops physiology-informed prognostic measures for neurodevelopmental disorders. The rest of the authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

6 Author Contributions

RW, MD, EJ-M, AEA, PB, and KL-H conceived design and functional requirements. RW wrote the paper; MD, EJ-M, AEA, and KL-H revised and edited the paper. RW wrote the code. MD provided the deep-learning model trained to recognize artifacts (under supervision of PB and KL-H) and code to run it included in RV. RW, MD, EJ-M, and AEA tested the software. KL-H provided data.

7 Funding

Supported by the program committees 'Compulsivity Impulsivity Attention' of Amsterdam Neuroscience (to KL-H), the Netherlands Organization for Scientific Research (NWO) Social Sciences grant 406.15.256 (to AEA and KLH), the NWO Dutch National Research Agenda, NWA-ORC Call (NewTdec: NWA.1160.18.200), and ZonMW Top grant (2019/01724/ZONMW) (to KL-H).

8 Acknowledgments

We thank the students from the courses "Rhythms of the Brain" (2021) and "Human Neurophysiology" (2022) at the Vrije Universiteit Amsterdam, as well as Jennifer Ramautar and Shilpa Anand at the N=You Neurodevelopmental Precision Center, for testing RV and giving feedback. Furthermore, we thank Laureen Weiler for designing the logo of RV.

9 References

Acharya, U.R., Sree, S.V., Swapna, G., Martis, R.J. and Suri, J.S., 2013. Automated EEG analysis of epilepsy: a review. Knowledge-Based Systems, 45, pp.147-165.

Arns, M., Gunkelman, J., Breteler, M. and Spronk, D., 2008. EEG phenotypes predict treatment outcome to stimulants in children with ADHD. Journal of integrative neuroscience, 7(03), pp.421-438.

Bartels, G., Shi, L.C. and Lu, B.L., 2010, August. Automatic artifact removal from EEG-a mixed approach based on double blind source separation and support vector machine. In 2010 Annual International Conference of the IEEE Engineering in Medicine and Biology (pp. 5383-5386). IEEE.

Barua, S. and Begum, S., 2014. A review on machine learning algorithms in handling EEG artifacts. In The Swedish AI Society (SAIS) Workshop SAIS, 14, 22-23 May 2014, Stockholm, Sweden.

[3] https://github.com/RobinWeiler/RV.git

Biasiucci, A., Franceschiello, B. and Murray, M.M., 2019. Electroencephalography. Current Biology, 29(3), pp.R80-R85.

Boudaya, A., Chaabene, S., Bouaziz, B., Batatia, H., Zouari, H., Jemea, S.B. and Chaari, L., 2022. A Convolutional Neural Network for Artifacts Detection in EEG Data. In Proceedings of International Conference on Information Technology and Applications (pp. 3-13). Springer, Singapore.

Connors, B.W. and Amitai, Y., 1997. Making waves in the neocortex. Neuron, 18(3), pp.347-349.

Diaz, B.A., Hardstone, R., Mansvelder, H.D., Van Someren, E.J. and Linkenkaer-Hansen, K., 2016. Resting-state subjective experience and EEG biomarkers are associated with sleep-onset latency. Frontiers in psychology, 7, p.492.

Craik, A., He, Y. and Contreras-Vidal, J.L., 2019. Deep learning for electroencephalogram (EEG) classification tasks: a review. Journal of neural engineering, 16(3), p.031001.

Gao, J., Lin, P., Yang, Y., Wang, P. and Zheng, C., 2010. Real-time removal of ocular artifacts from EEG based on independent component analysis and manifold learning. Neural Computing and Applications, 19(8), pp.1217-1226.

Gramfort, A., Luessi, M., Larson, E., Engemann, D.A., Strohmeier, D., Brodbeck, C., Goj, R., Jas, M., Brooks, T., Parkkonen, L. and Hämäläinen, M., 2013. MEG and EEG data analysis with MNE-Python. Frontiers in neuroscience, p.267.

Jas, M., Engemann, D.A., Bekhti, Y., Raimondo, F. and Gramfort, A., 2017. Autoreject: Automated artifact rejection for MEG and EEG data. NeuroImage, 159, pp.417-429.

Jurczak, M., Kołodziej, M. and Majkowski, A., 2022. Implementation of a Convolutional Neural Network for Eye Blink Artifacts Removal From the Electroencephalography Signal. Frontiers in Neuroscience, 16.

LeCun, Y., Bengio, Y. and Hinton, G., 2015. Deep learning. nature, 521(7553), pp.436-444.

Liu, Y., Höllerer, T. and Sra, M., 2022. SRI-EEG: State-Based Recurrent Imputation for EEG Artifact Correction. Frontiers in Computational Neuroscience, 16.

Niedermeyer, E. and da Silva, F.L. eds., 2005. Electroencephalography: basic principles, clinical applications, and related fields. Lippincott Williams & Wilkins.

Pardede, J., Turnip, M., Manalu, D.R. and Turnip, A., 2015. Adaptive recurrent neural network for reduction of noise and estimation of source from recorded eeg signals. ARPN Journal of Engineering and Applied Sciences, 10(3).

Radüntz, T., Scouten, J., Hochmuth, O. and Meffert, B., 2015. EEG artifact elimination by extraction of ICA-component features using image processing algorithms. Journal of neuroscience methods, 243, pp.84-93.

Roy, Y., Banville, H., Albuquerque, I., Gramfort, A., Falk, T.H. and Faubert, J., 2019. Deep learning-based electroencephalography analysis: a systematic review. Journal of neural engineering, 16(5), p.051001.

Sai, C.Y., Mokhtar, N., Arof, H., Cumming, P. and Iwahashi, M., 2017. Automated classification and removal of EEG artifacts with SVM and wavelet-ICA. IEEE journal of biomedical and health informatics, 22(3), pp.664-670.

Shao, S.Y., Shen, K.Q., Ong, C.J. and Wilder-Smith, E.P., 2008. Automatic EEG artifact removal: a weighted support vector machine approach with error correction. IEEE Transactions on Biomedical Engineering, 56(2), pp.336-344.

Shoker, L., Sanei, S. and Chambers, J., 2005. Artifact removal from electroencephalograms using a hybrid BSS-SVM algorithm. IEEE Signal Processing Letters, 12(10), pp.721-724.

Smith, S.J., 2005. EEG in the diagnosis, classification, and management of patients with epilepsy. Journal of Neurology, Neurosurgery & Psychiatry, 76(suppl 2), pp.ii2-ii7.

Sun, W., Su, Y., Wu, X. and Wu, X., 2020. A novel end-to-end 1D-ResCNN model to remove artifact from EEG signals. Neurocomputing, 404, pp.108-121.

Virtanen, P., Gommers, R., Oliphant, T.E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J. and Van Der Walt, S.J., 2020. SciPy 1.0: fundamental algorithms for scientific computing in Python. Nature methods, 17(3), pp.261-272.

Wang, S., Guo, B., Zhang, C., Bai, X. and Wang, Z., 2017, October. EEG detection and de-noising based on convolution neural network and Hilbert-Huang transform. In 2017 10th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI) (pp. 1-6). IEEE.

Welch, P., 1967. The use of fast Fourier transform for the estimation of power spectra: a method based on time averaging over short, modified periodograms. IEEE Transactions on audio and electroacoustics, 15(2), pp.70-73.

Wolpaw, J.R., McFarland, D.J., Neat, G.W. and Forneris, C.A., 1991. An EEG-based brain-computer interface for cursor control. Electroencephalography and clinical neurophysiology, 78(3), pp.252-259.

Yang, B., Duan, K. and Zhang, T., 2016. Removal of EOG artifacts from EEG using a cascade of sparse autoencoder and recursive least squares adaptive filter. Neurocomputing, 214, pp.1053-1060.

Yang, B., Duan, K., Fan, C., Hu, C. and Wang, J., 2018. Automatic ocular artifacts removal in EEG using deep learning. Biomedical Signal Processing and Control, 43, pp.148-158.

10 Figure Captions

Figure 1: "Preprocessing" pop-up window and initial screen of RV. There are four sections: EEG-file selection, Preprocessing settings, visualization settings, and deep-learning model predictions. Clicking the "Plot" button will close this window and, after a short loading screen, lead to the main GUI.

Figure 2: "Stats" pop-up window offering an overview over the annotated data. Currently, the integrated statistics consist of the recording's total length (in seconds), the amount of annotated and non-annotated data (in seconds), as well as the amount of consecutive, non-annotated intervals longer

than two seconds. Finally, there is a histogram to show the distribution of the length of non-annotated data.

Figure 3: Flowchart of RV's preprocessing and visualization settings pipeline.

Figure 4: Channel-selection pop-up window where the user can select specific channels to plot. At the top, there is a dropdown-menu to select channels out of a list of all channels. Alternatively, the channels can be selected on the scalp-topography plot below.

Figure 5: Graphical User Interface (GUI) of RV with its five subsections highlighted: Menu bar, taskbar, labeled buttons, view-slider, legend, and plot.

Figure 6A: Example timeframe of an EEG file displayed in RV. One can see an annotated artifact (in red) and the respective deep-learning model predictions at the bottom of the plot, in accordance with the colorbar in Figure 7B.

Figure 6B: Colorbar of deep-learning model predictions seen in Figure 6A, where the color indicates predicted probability of an artifact. Red (1) corresponds to high probability and blue (0) to low probability.

EEG-file selection

Drag-and-drop or click here to select EEG file

Selected file:

Stats

Pre-processing

High-pass: in Hz Low-pass: in Hz

Reference: None

Bad-channel detection: None

Interpolate bad channels

Visualization

Resampling rate: in Hz

Scale: default: 1e-6

Channel-offset: default: 40 (µV

Segments (in sec): default: None

Activate view-slider

Select channels

Deep-learning model

Drag-and-drop or click here to select model output

Selected model output:

Remove model predictions

Run model from run_model.py

Annotate according to model with threshold default: 0.5

Plot

Recording length (in sec):

181

Amount of annotated data (in sec):

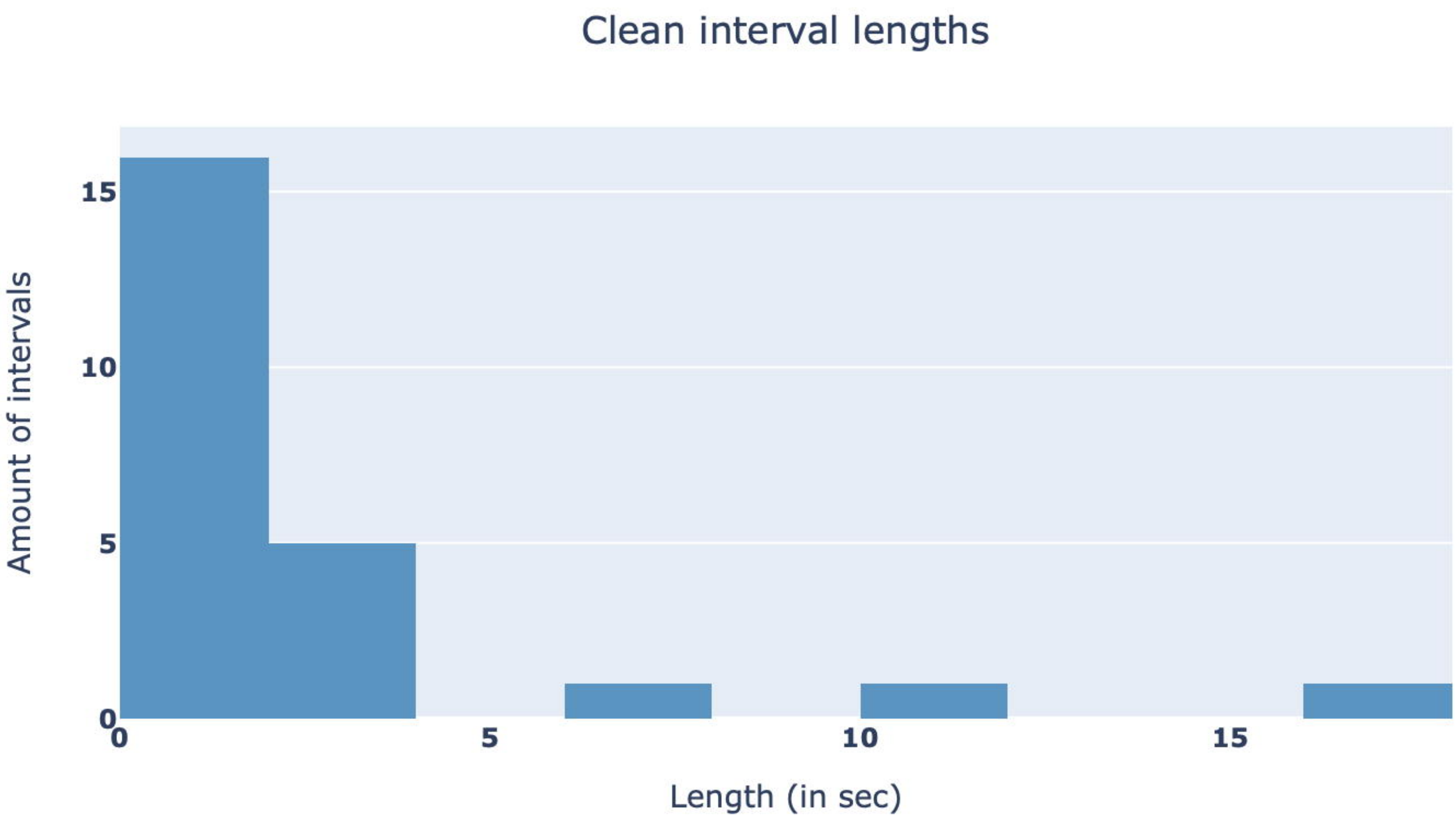
113.74

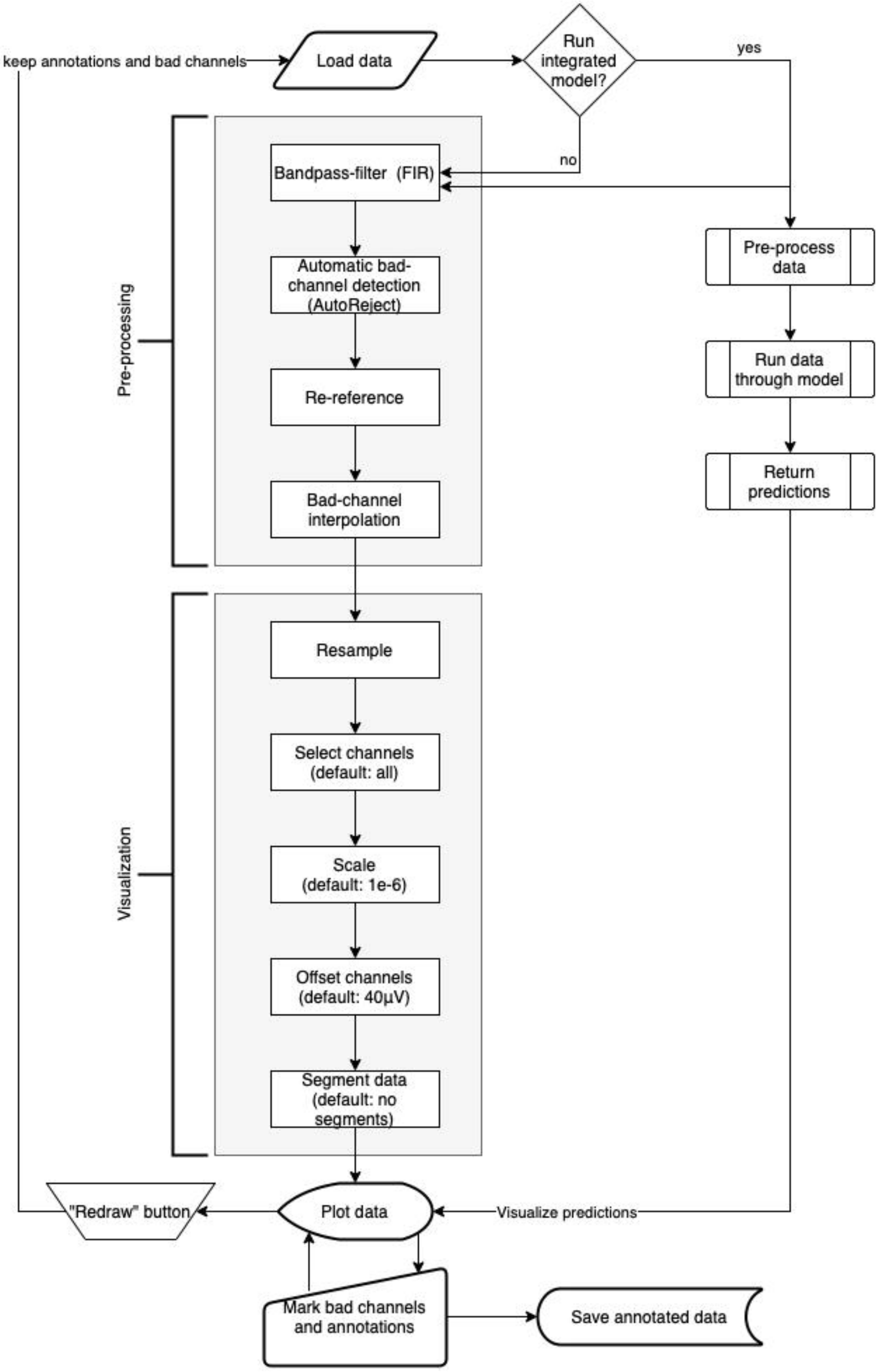
Amount of clean data left (in sec):

67.26

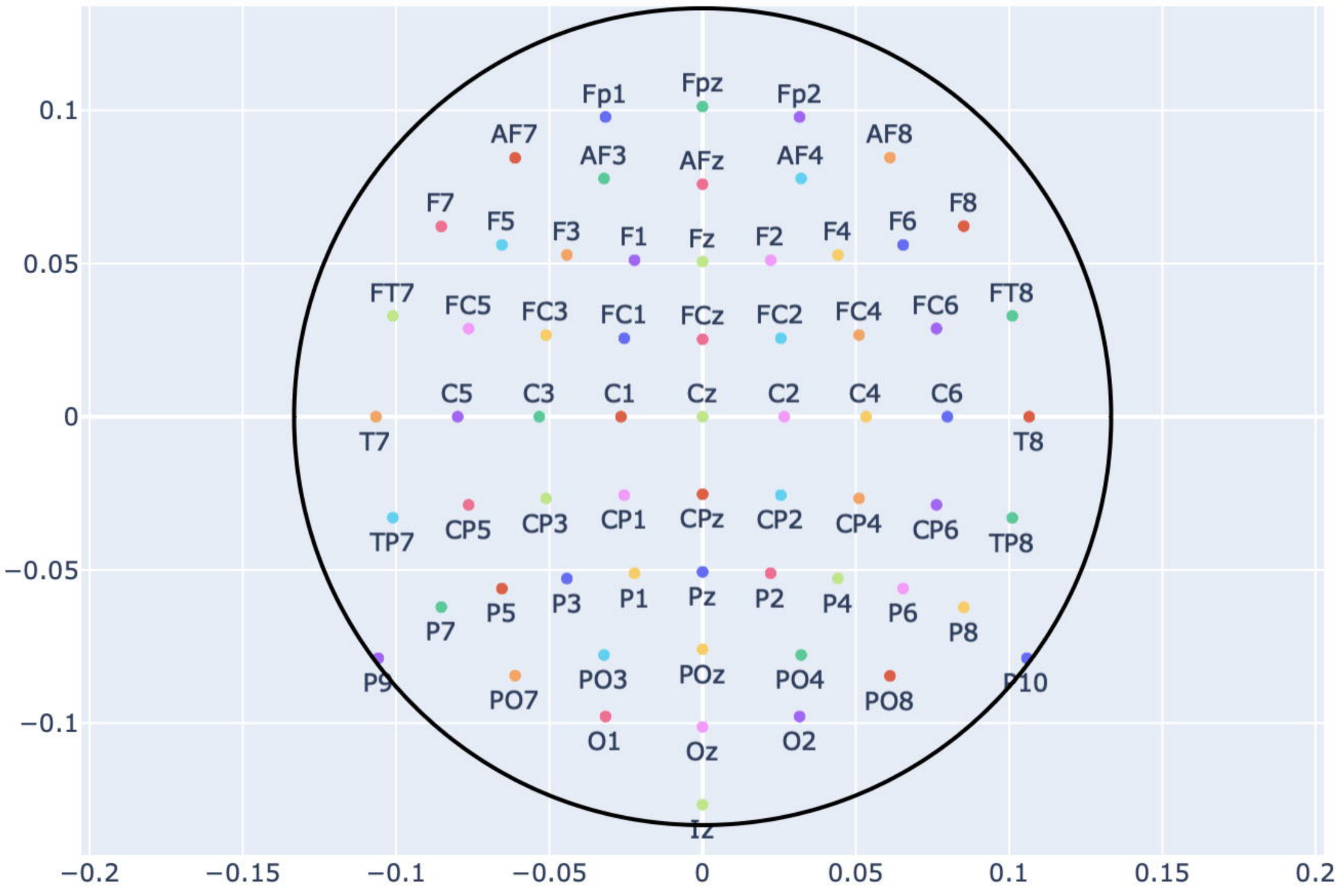
Amount of clean intervals longer than 2 seconds:

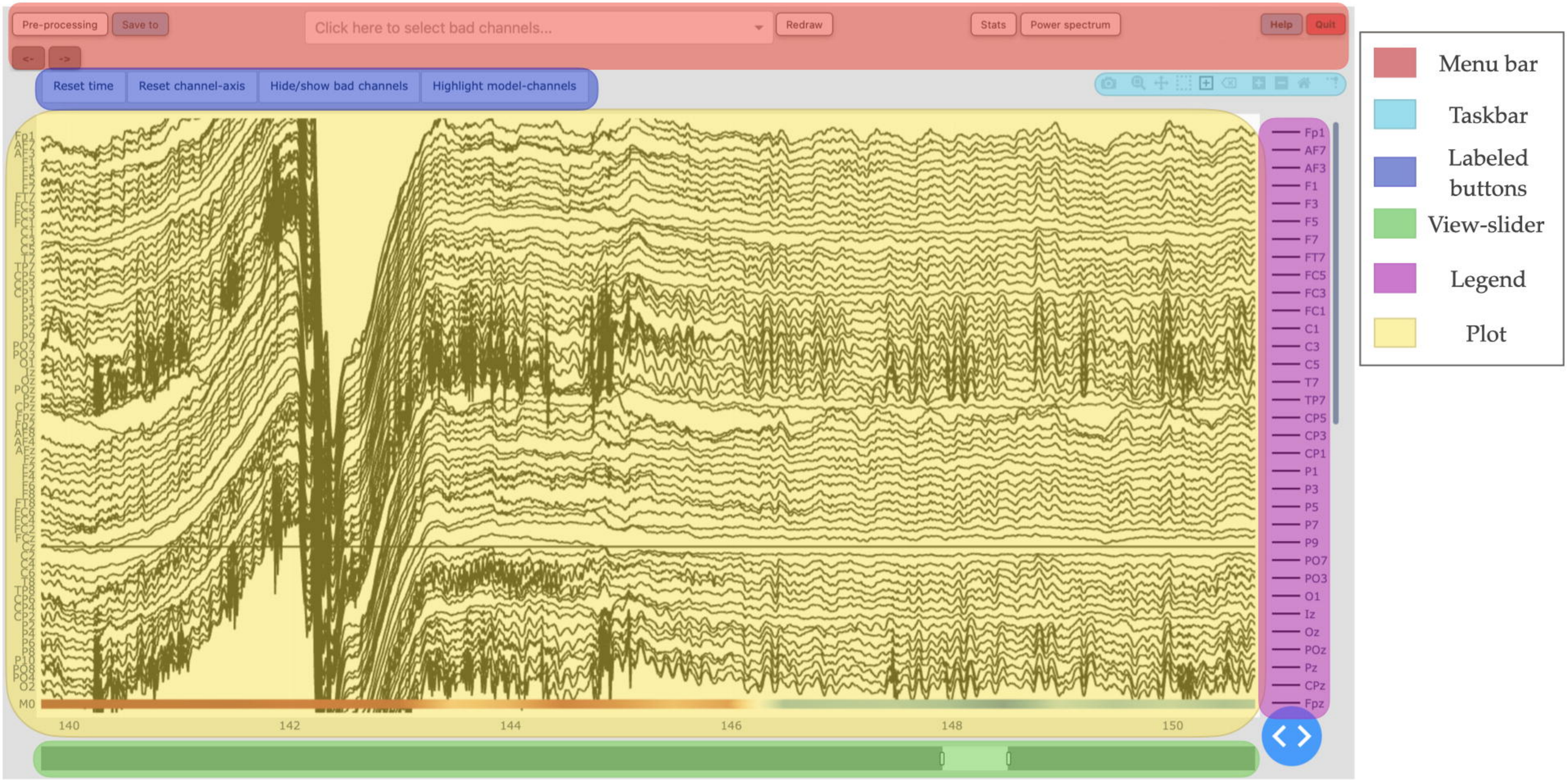
25

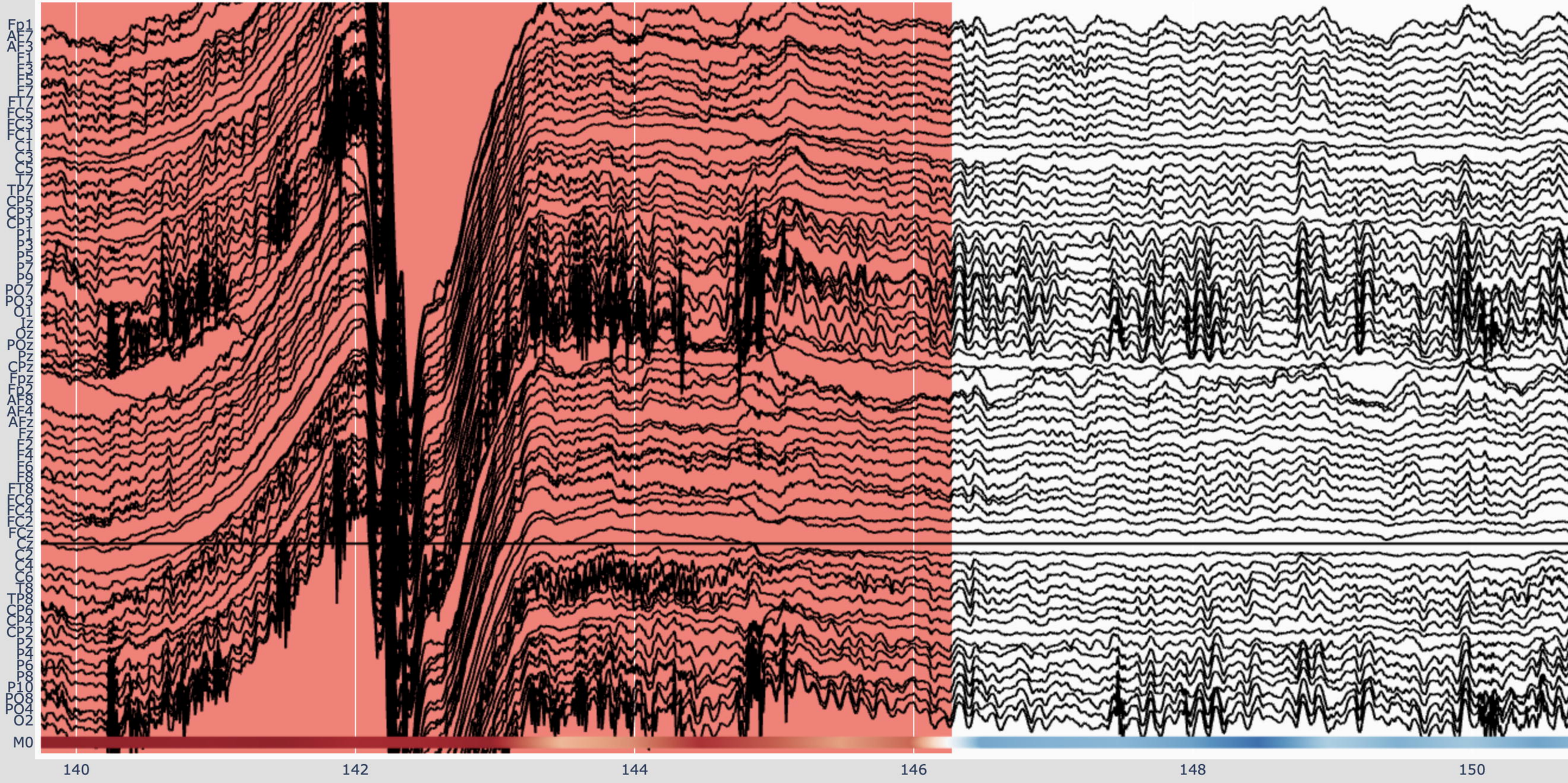



keep annotations and bad channels
Load data
Run integrated model?
yes
no
Pre-processing
Bandpass-filter (FIR)
Automatic bad-channel detection (AutoReject)
Re-reference
Bad-channel interpolation
Pre-process data
Run data through model
Return predictions
Visualization
Resample
Select channels (default: all)
Scale (default: 1e-6)
Offset channels (default: 40µV)
Segment data (default: no segments)
"Redraw" button
Plot data
Visualize predictions
Mark bad channels and annotations
Save annotated data

Click here to select channels to plot or select them in the plot below...




Pre-processing
Save to
Click here to select bad channels...
Redraw
Stats
Power spectrum
Help
Quit
<-
->
Reset time
Reset channel-axis
Hide/show bad channels
Highlight model-channels
140
142
144
146
148
150
Menu bar
Taskbar
Labeled buttons
View-slider
Legend
Plot


Fp1
AF7
AF3
F1
F3
F5
F7
FT7
FC5
FC3
FC1
C1
C3
C5
T7
TP7
CP5
CP3
CP1
P1
P3
P5
P7
P9
PO7
PO3
O1
Iz
Oz
POz
Pz
CPz
Fpz
Fp2
AF8
AF4
AFz
Fz
F2
F4
F6
F8
FT8
FC6
FC4
FC2
FCz
Cz
C2
C4
C6
T8
TP8
CP6
CP4
CP2
P2
P4
P6
P8
P10
PO8
PO4
O2
M0
140
142
144
146
148
150

1
0.8
0.6
0.4
0.2
0