Abstract

Monitoring sleep via scalp electroencephalography (EEG) is one of the most common approaches. Researchers intend to find neural events, such as spindles, k-complexes, and sleep stages, in segmented EEG recording results. However, doing such data analysis by human labor is time consuming and results are suffered from low inter-rater agreement. To address these issues, objective and automated pipelines are proposed. The current study focuses on optimizing filter based and thresholding approaches in classifying spindles, and we use our approach as a step stone for searching the best naive machine learning pipeline, using raw signals, peak frequency, and dominant power spectral density as features. We argue that filtered based and thresholding approaches allow us to explicitly define neural event features of interest in a flexible way and subjectively look for neural events that matches all the features. Machine learning algorithms could yield better results than thresholding approaches but they require intensive feature engineering and model selection. The automated pipeline designed for the data in the current study was applied to the DREAMS sleep spindle dataset (Devuyst et al., 2008). Our pipeline yielded a sensitivity that is close to human experts.

The functional role of sleep in mammal primates remains debatable (Maquet, 2001; Rasch and Born, 2013; Stickgold and Walker, 2013; Heol, Alabantakis, Cirelli, and Tononi, 2016; Wallant, Maquet, and Phillips, 2016). One theory is that the occurrences of particular neural events during sleeps improve memories consolidation (need reference). It has become a challenge because identifying these neural events through viewing all the data is time consuming and objective, especially in high definition neural recordings, which contain thousands of data points in just a few seconds of recording. One of the neural recording techniques that is to record and monitor is electroencephalography (EEG).

Macro and micro structures are typically found in segmented recordings. Macro-structured neural events refer to segments that are usually 20 seconds to 30 seconds long data representing different levels of sleep compared to the awake condition (Liu, et al., 2015; Muller, et al., 2006; Ebrahimi, et al., 2008), namely sleep stages.

On the other hand, micro-structured neural events refer to local and short segments, such as spindles. Spindles typically occur during sleep stage 2, and they are believed to be generated from the thalamus (Tsanas and Clifford, 2015; Wallant et al., 2016). Based on the dominant frequency of a segment of a spindle, spindles are classified to slow spindles (9-10 Hz, Andrillo et al., 2011 and Molle et al., 2011; 10 – 12 Hz, De Gennaro and Ferrar, 2003; Anderer, et al., 2001; Terrier and Gottesmann, 1978) and fast spindles (13 – 15 Hz, Astori et al., 2013; De Gennaro and Ferrar, 2003; 12 – 14 Hz, Anderer, et al., 2001; Terrier and Gottesmann, 1978), which are believed to occur during different phases of slow oscillations (< 1 Hz) (Molle et al., 2011).

Measuring neural events and analyzing their relation with cognitive behaviors provide insights of how sleeps helps or hurt memory, as well as diagnostic measurements for various sleep disorders. It is unanswered how brain activities integrate past information to generate new memories, thus, neural events provide common and objective measurements of sleep so that we can connect these neural events with memory and describe how the brain processes information during sleep.

Given the importance of identifying these neural events to understand memory and sleep better, there have been many studies aim to classify spindles and k-complexes automatically (i.e., Bergmann et al., 2012; Schimicek et al., 1994; Perumalsamy et al., 2009; Palliyali et al., 2015; Subasi et al., 2005). Studies propose machine learning algorithms to classify these neural events, and usually, they make classifications based on one single EEG channel and long period (> 7 hours) recordings. Among all the automated algorithms, filtering based and thresholding approaches have been making progress to classify sleep stages, spindles, and k-complexes (e.g., Huupponen et al., 2000; Devuyst et al., 2010). Methods like template-based filtering using continuous wavelet transforms (CWTs) (Erdamar et al., 2012), artificial neural networks (ANN) (e.g., Günes et al., 2011), Support Vector Machines (SVMs) (e.g., Acir and Güzelis, 2004) and decision-trees (Duman et al., 2009) are investigated. However, there are not many studies classify spindles and other neural events simultaneously using a unified framework (Jobert et al., 1992; Koley and Dey, 2012; Jaleel et al., 2013; Camilleri et al., 2014; Parekh et al., 2015). One of the reasons is that spindles have unique symmetrical shape along the temporal axis, while other neural events do not have.

In most studies mentioned above, sophisticated machine learning algorithms yield better results in classifying complex neural events and “nonevents”; however, it is not useful for small datasets or practical use. Machine learning algorithms usually require large amount of data (> 1000 training samples) and at least some level of feature engineering, but neither of which is common approach of non-academic studies, especially many sleep studies are in clinical setting. Some open source programs can perform recursive model selection and manage to determine the best machine learning model and its corresponding hyper-parameters, which are determined by predicting power (i.e., accuracy, areas under the curve, etc.). For practical reasons, having large amount of data for training a machine learning model is arguably not realistic, because the amount of spindle examples and non-spindle examples we could extract from EEG recordings are not balanced. It is very common that we have lots of non-spindle examples and little spindle examples. For example, if we have 95 non-spindle examples and 5 spindle examples. We could find any machine learning models report a at least 95% accuracy by claiming everything a “non-spindle”, but that is not what we want to see in the practical field like clinical setting. For such reason, taking multiple standardized signal processing steps to apply a thresholding approach could avoid unconditional rejection. It might be true that a multi-step thresholding pipeline reduces signal-to-noise ratio, but detected neural events objectively contain defined-features that are given by the pipeline, which benefits researchers if they have unique features of some neural events in mind and want to look for them through large amount of data.

Also, prepare training data reminds a difficult problem in neuroscience. Neural events like spindles could occur with varying durations, and this makes it difficult to prepare training data for machine learning algorithms, because most machine learning algorithms require consistent size of training samples (Wallant et al., 2016).

The first focus of the current study is to design a filter-based and thresholding approach in order to detect spindles in a fast manner with flexible parameters that capture temporal and spectral variations (frequency, duration, amplitude, etc.). This thresholding pipeline will be the step stone for preparing representative spindles and non-spindles. In terms of non-spindles, we are mostly interested in those samples that are classified as spindles by this thresholding pipeline but rejected by human experts.

The second focus of the current is to classify spindles and non-spindles based on multi-channel criteria using short period of EEG recordings. Literature rarely implements algorithms to classify neural events based on multiple EEG channels (> 2) and/or short-term recordings (< 30 minutes). The current study aimed to implement filtering based and thresholding approaches to address multi-channel classification and these classifications are made on short period high definition EEG recordings. Also, this study focused on optimize parameters used to define a typical spindle in our thresholding model in order to evaluate features used in classifying spindles by human annotators. To make it a simpler classification problem and focus on optimization and validation processes, the current study mainly focused on classifying spindle and non-spindles.

Method

Data Acquisition

A total of 64 channels of EEG data were continuously recorded (1 kHz sampling) using an actiCHamp active electrode system (Brain Products, GmbH) while subjects napped on a bed inside a sound-attenuated testing booth (IAC Industries). Before EEG recording began, the impedance of all electrodes was optimized to be < 10 kOhms after application of electrolytic gel between scalp and the electrode tips. The reference electrode during recording was T9 (left mastoid) and two additional flat electrodes were placed around the eye for electrooculography (EOG) recording. Date used in the current study (https://osf.io/chav7/) contained 29 EEG sessions (15 subjects from 2 separate studies, 2 sessions for each subject except one subject). Each session is a 20-minute or 30-minute nap taken after scene learning and before a memory retrieval task. Due to the amount of data that needed to be viewed by human annotator, we don’t have complete results of spindles. Subject 12, 14, 19, 20, and 25 are showing one of the two days’ results. To match annotated spindle results, our pipeline automatically matches file names and then proceeds to statistical analysis. EEG recordings that do not have a corresponding annotated file are skipped. In total, we have 37 annotated files that contain sleep stages and locations of spindles.

Experimental Procedures

*Load 2 Working Memory Learning Task (Load 2)*

In each trial, participants were presented 2 images of scenes consecutively with \_\_ms in between. Each image was presented for 1000 ms. Followed by the last image presented, a blank screen was presented for 6000 ms and a probe image was presented for 2000 ms. During the presentation of the probe image, participants were expected to decide whether the probe image existed in the sequence of presented images. No matter participants responded or not, a scramble image would be presented for 1000 ms right after the presentation of the probe image, indicating the end of a trial. There were 100 trials in the task.

*Load 5 Working Memory Learning Task (load 5)*

In each trial, procedures were similar to Load 2 Working Memory Learning Task, except participants were learning a sequence of 5 images instead and there were 40 trials in the task.

*Complete Pipeline of Experimental Procedure*

Participants performed a working memory learning task (load 2 and load 5 in two different days) and a recognition task (recognition 1) before they took a 20-minute or 30-minute nap in a dark room with EEG recording attached. After the nap, participants performed another recognition task (recognition 2). We had each participant’s EEG recordings during each sub-session of the pipeline. The current study focused on classifying neural events, namely spindles, during the 20-minute or 30-minute nap EEG recordings. Order of days to take the working memory learning task was counterbalanced.

*Manual annotated spindle*

[insert annotation procedures and criteria]

*Filter Based and Thresholding Approach Pipeline*

Preprocessing

Standard procedures of EEG preprocessing such lowpass filtering, artifact removal, downsampling, and etc (https://osf.io/fc3u5/wiki/home/). were performed in an automatic pipeline that went through each EEG recording. Data were lowpass at 200 Hz and a notch filter at 60 Hz was applied. Subject 5 to 10 were downsampled to 500 Hz due to their length and computation convenience. Artifact removal were performed based on MNE-python Independent Component Analysis (ICA) procedures (Gramfort et al., 2013; Gramfort et al., 2014). All the hyper-parameters for ICA processing was fixed except local amplitude change that was used for rejecting bad segments of data. This parameter was varied from 80 to 160 mV and was tested by “try-except” algorithm. Preprocessed data were saved for later used. No other human input was involved in the preprocessing stage. Preprocessed data were bandpassed between 0.1 Hz to 50 Hz.

Classification Pipeline

To classify spindles, we applied a filter based and thresholding approach to our data. Based on spindles that classified manually, features were bandpass frequency range, channels of interest, moving window size, threshold value (lower and upper boundaries), duration of spindles, decision making criteria based on number of channels. We defined spindles had dominant powers at 11 - 16 Hz with a duration varying between 0.5 to 2 seconds. Channels of interests are F3, F4, C3, C4, O1, and O2. After bandpass filtering data at these 6 channels, Root-Mean-Square (RMS) of individual channel was computed using a Gaussian moving window (formula 1 and 2). Harmonic mean of these 6 RMSs were also computed for later analysis. Threshold for classifying spindles was computed as formula 3 shows. If a segment of EEG recording exceeded a lower pass criterion but not exceeded an upper passing criterion above the mean for 0.5 seconds to 2 seconds, a possible spindle location was marked. This could be done on individual RMSs and the mean RMS (formula 3). To finalized a spindle, at a given possible location of spindle in the mean RMS, possible spindle locations must also be detected in at least 3 individual channels, with one second temporal variation. In other words, the temporal criteria allows 1 second of toleration. The naive thresholding pipeline takes no extra information except signals of the EEG recordings, thus, sleep stage information could be added to finalize spindle classification in order to reduce false alarm rate. Spindles marked in the first 300 and last 100 seconds are excluded because recording procedures are not consistent across subjects during these times.

*Validation Pipeline*

Validation of our filter based and thresholding approach is done between predicted spindles and manual annotated spindles. Temporally continuous data is segmented to 3 second non-overlapping epochs. A predicted spindle detected from our thresholding approach contains duration information, and we compute the time interval of the spindle. A manual annotated spindle does not contain duration information, thus, we define fixed time interval of 2 seconds for the spindle, which starts from 0.5 seconds before the annotated time stamp. If a given epoch overlaps both the time interval of a predicted spindle and a manual annotated spindle, this epoch is marked as a “hit”. If a given epoch overlaps only the time interval of a predicted spindle but not a manual annotated spindle, this epoch is marked as a “miss”. If a given epoch overlaps only the time interval of a manual annotated spindle but not a predicted spindle, this epoch is marked as a “false alarm”. If not any case above, this epoch is marked as a “correct rejection”. It is convention to favor using area under the curve (AUC) to measure validation results, which provides a sensitive, nonparametric criterion-free measure (Fawcett, 2006).

We also use AUC to optimize both upper and lower thresholds (formula 3). To avoid data overfitting, cross validation loops are nested to the validation pipeline, such that performance is measured only on novel left-out data samples. AUCs are computed across a parameter space of thresholds. Due to the thresholding approach is not a typical “train-test”, a k-folds cross validator is applied to select samples of predictions and the true labels (spindles and non-spindles marked by human annotators) and compare them. The mean AUC is computed from 10-folds (Python::sklearn.K\_folds) cross validation at individual level.

With other parameters set as fixed (bandpass filter between 11 Hz to 16 Hz, convolution kernel in size of 500 samples, leaving the first 100 seconds and the last 300 seconds of sample aside, restrict to 6 channels of interest, duration varying between 0.5 to 2 seconds, and a temporal toleration of 1 second), we perform a grid search for the lower and upper threshold values. The lower threshold varies between 0.1 to 1, with 0.1 as the step size. The upper threshold varies between 2 to 3.5, with 0.1 as the step size. In total, 160 pairs of lower and upper threshold values are tested to find the best pair. During the grid search, samples of miss and false alarm are collected from all individual subjects, these samples are labeled as “1” and “0” to specify spindle and non spindle samples, which is done under each pair of lower and upper threshold values (Figure 2, bottom panel). Each sample of the data is vectorized raw EEG segment (3 seconds, size of 6 x 1500) concatenated with the peak power (size of 6 x 1) and the frequency at the peak power (size of 6 x 1). Therefore, each sample is 9012 samples in size. After we determine the best pair of thresholds, we provide the sample data (miss and false alarm segments) and labels to the a Python library that automate the process of exploring thousands of possible pipelines to find the best one for this sample and label (https://github.com/rhiever/tpot). This library provides us the benchmark comparison of a naive machine learning model without extended feature engineering.

To compare the machine learning model with our thresholding model, the machine learning model is cross validated with individual subject (10-folds cross validation), and the thresholding model is cross validated with the best thresholds also at the individual level (10-fold cross validation).

Results

Figure 1 shows density distribution of each subject at each recording. Density distribution is estimated by a kernel density in Python::seaborn.violiplot. Number of points in the discrete grid used to compute the kernel density estimate is 20. We observe that the peaks of the density distributions do not occur at similar time, which suggests a high between-subject variability in terms of sleep.

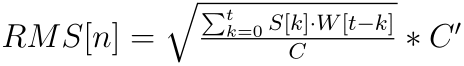
The validation pipeline with 36 individual EEG nap data yielded that the best lower threshold is 0.4 and the best upper threshold is 3.5.

Formula 1

euqation 1-1.png

W denotes the window function. The standard deviation of the function is defined by the moving window size (number of samples). n is the length of the moving window size and the standard deviation is computed by half of n/0.68.

Formula 2



Convolution is applied to compute the RMS. Computed RMS uses Python::numpy::convolve and remains the same sampling rate as the target signal. C and C’ are terms to change the scales of the results.

Formula 3

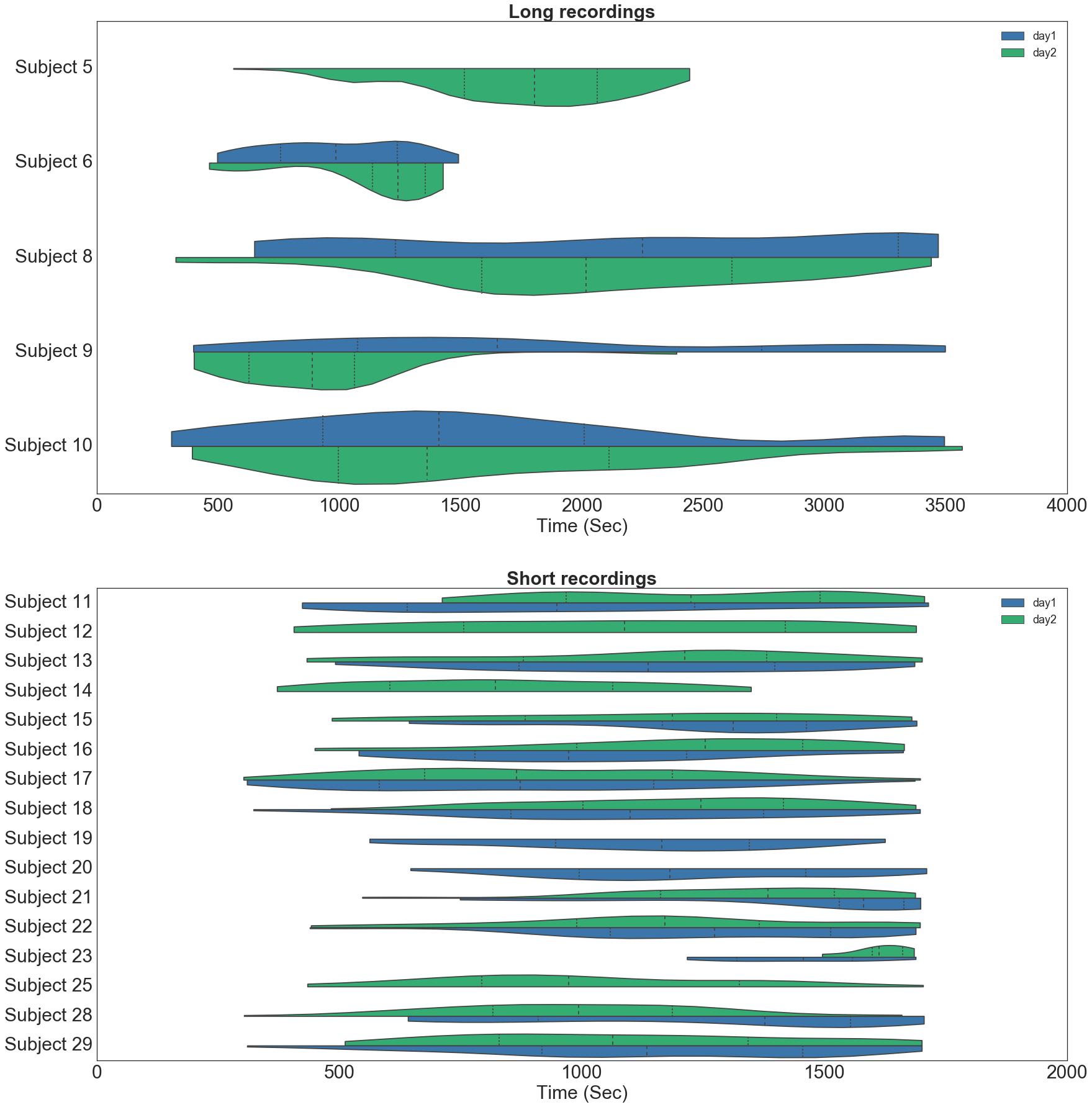
equation 31-1.png

equation 32-1.png

Trimmed mean and trimmed standard deviation take account of 95% of the data to avoid influence of outliers. High pass criterion means the RMS is higher than the criterion value, and low pass criterion means the RMS is lower than the criterion value. Thresholds are determined at relatively individual channel.

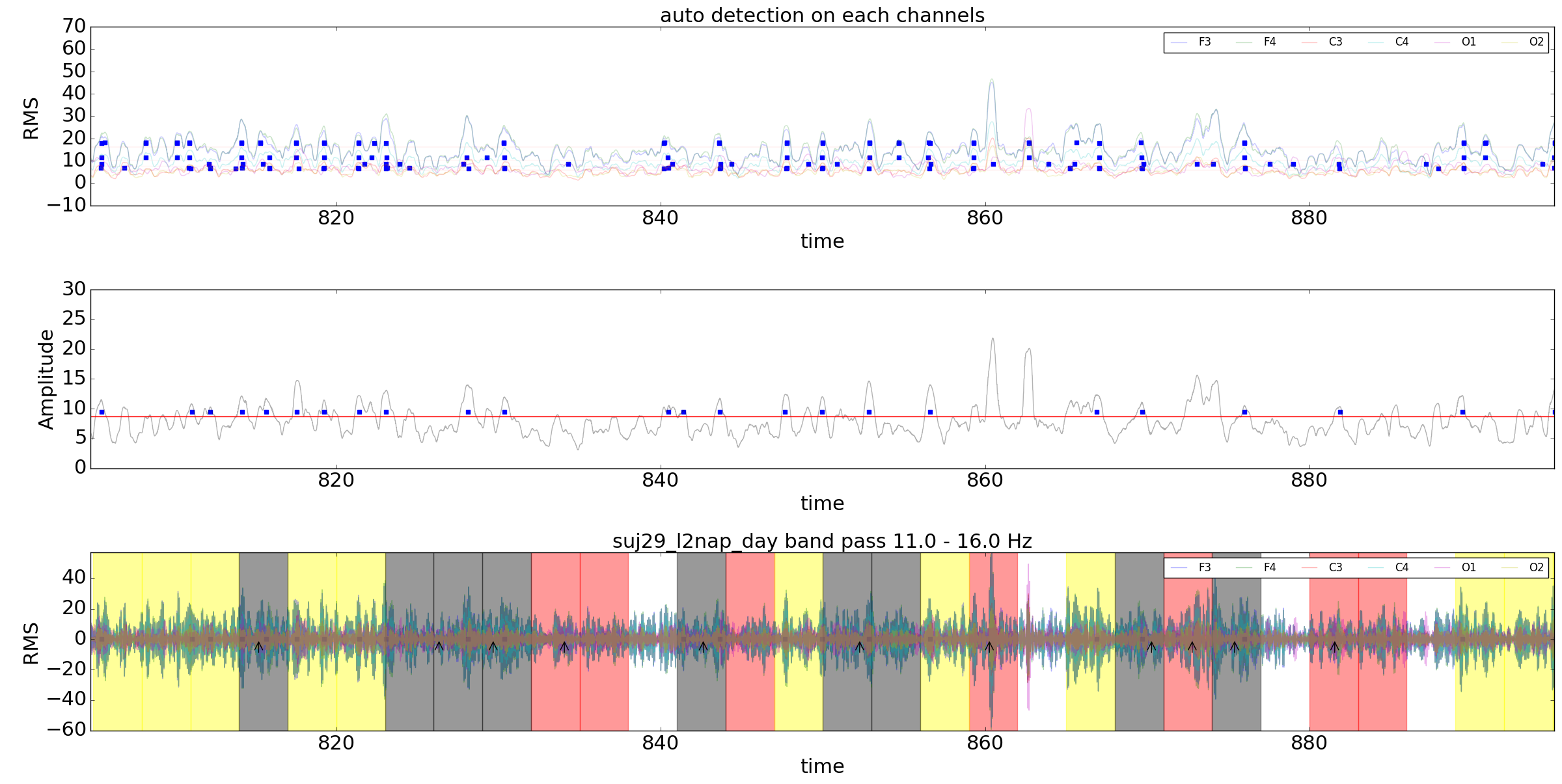
Manual Annotation of Spindles

(figure 1)



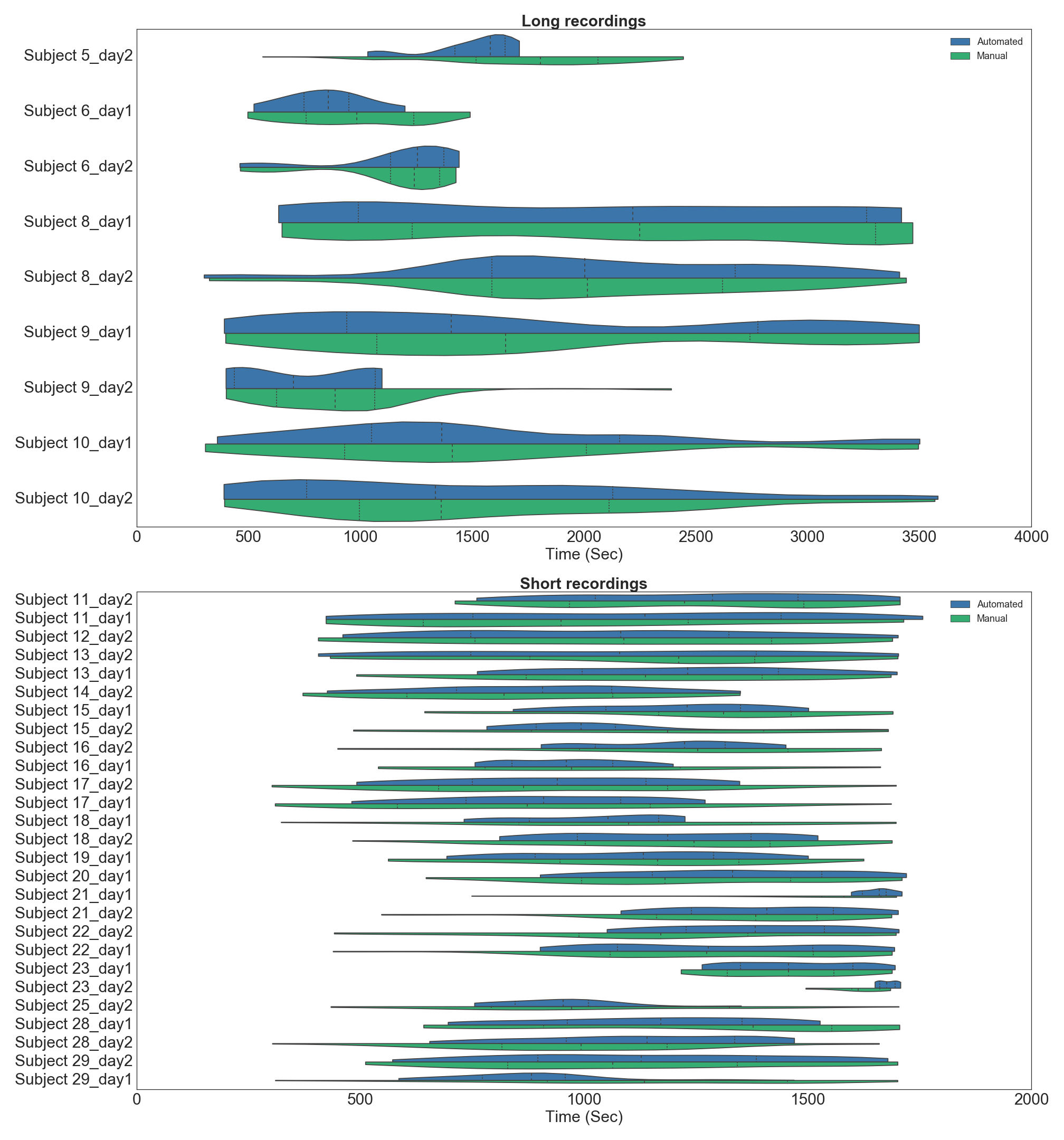
Temporal density of spindles based on result classified by human annotators. Two human annotators marked spindle locations on two subsets of the EEG recordings. One worked on those who had long recordings (> 30 minutes), and the other worked on those who had short recordings (< 20 minutes). Annotation data was not completed, thus, we show one of two days for subject 12, 14, 19, 20, and 25. In order to compare with automated marked annotations, spindles marked in the first 300 seconds and the last 100 seconds are excluded. Density is computed by Python::seaborn.violinplot, and both ends of the density distribution are cut by the first and the last spindle event. Three dash lines in one half of a violin plot represent 25%, 50%, and 75% quartile of time.

Figure 2 (visualize example segment of classification)



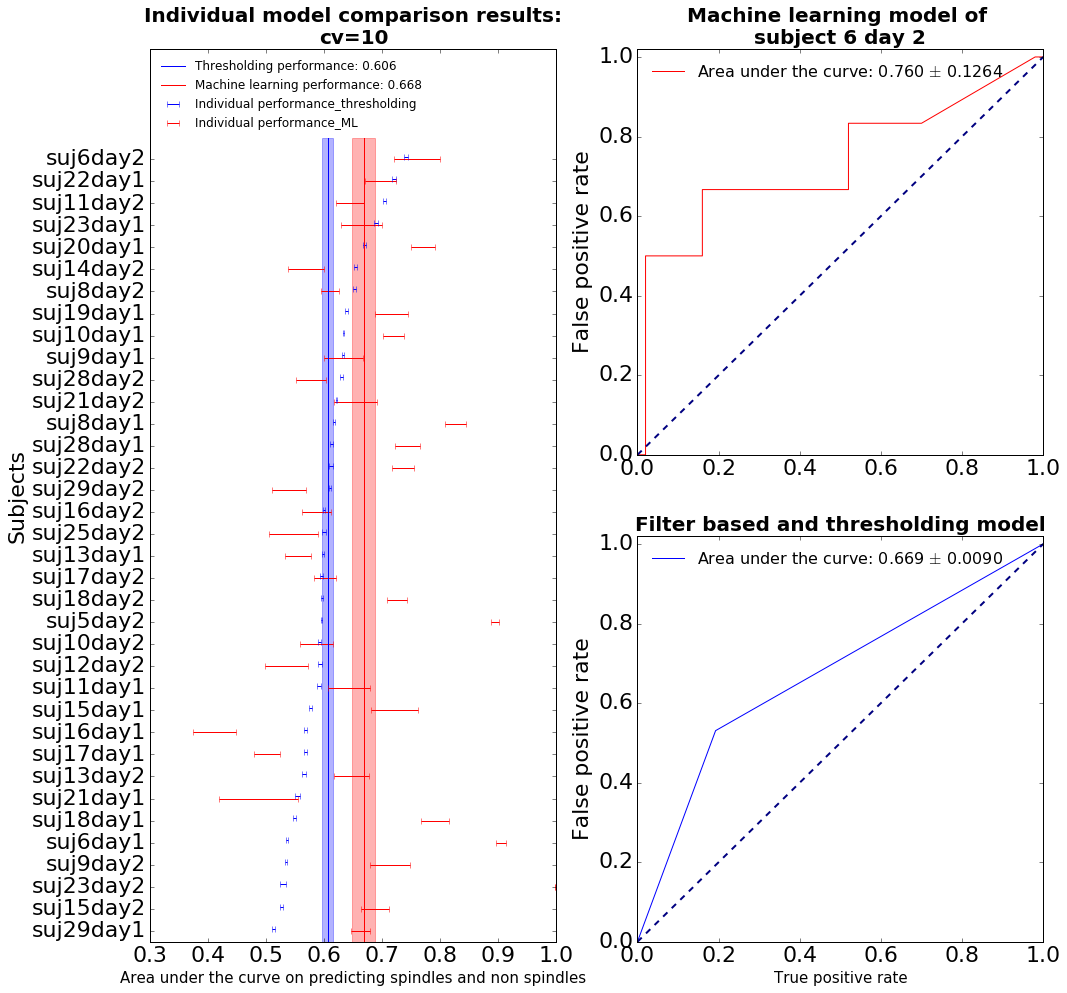
The top panel shows the root-mean-square (RMS) of individual channels (F3, F4, C1, C2, O1, O2). Each blue square represents a possible spindle at the level of individual channel. These possible spindles are detected by the thresholding algorithm presented above. The middle panel shows the average RMS of all 6 channels of interest. Each blue square represents a possible spindle at the mean level. These possible spindles are also detected by the same algorithm. At a given time point (with 1 second of variation), only when detection criteria were simultaneously met in both the average level and in at least 3 original channels a spindle event is included in the bottom panel, and classified as a spindle for later analysis. The bottom panel shows the validation results between manual annotated spindles and automated annotated spindles. In validation, temporal continuous data are segmented to 3 seconds epochs. A hit is color coded black, a miss is color coded red, a false alarm is color coded yellow, and a correct rejection is color coded white. Comparison algorithm is described in the method section.

Figure 3 (comparison between auto spindle and manual spindles in terms of density distribution)



A comparison between human marked annotations and automated marked annotations. Automated marked annotations are made by the optimal threshold parameters. [insert best parameters]. Automated marked annotations are more consistent with human marked annotations for long recordings but not for short recordings, in terms of spindle density. Spindle density is computed by Python::searbon.violinplot, with kernel size of 20. In general, automated approach classify more spindle case than human annotators.

Figure 4. Validation results



A comparison between the best machine learning model without a feature engineering with a filter based and thresholding model at individual level. The left panel shows the machine learning model validation results in red and thresholding model validation result in blue.