Investigating EEG Burst Suppression for Coma Outcome Prediction

by Tiange Zhan

[S.B., C.S. M.I.T., 2017]

Submitted to the
Department of Electrical Engineering and Computer Science
in Partial Fulfillment of the Requirements for the Degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

Massachusetts Institute of Technology

June 2018

Author: _____
Department of Electrical Engineering and Computer Science
May 25, 2018

Certified by: _____
[Una-May O'Reilly, Principal Research Scientist] Thesis Supervisor
May 25, 2018

Certified by: _____
[Abdullah Al-Dujaili, Post-doctoral Associate] Thesis Co-Supervisor
May 25, 2018

Accepted by: _____
Katrina LaCurts, Chair, Master of Engineering Thesis Committee

Investigating EEG Burst Suppression for Coma Outcome Prediction
by Tiange Zhan
Submitted to the Department of Electrical Engineering and Computer Science

May 25, 2018
In Partial Fulfillment of the Requirements for the Degree of Master of Engineering in Electrical Engineering and Computer Science

ABSTRACT

Every year, over 300,000 incidents of cardiac arrest occur in the United States. Of the people who are successfully resuscitated and brought to the hospital, approximately 80% remain unconscious for some amount of time Marion [2009]. Predicting whether or not a patient will wake up from coma, as well as the patient's neurological function after waking up, is an important task in guiding treatment decisions for physicians and family of the patient. This project seeks to improve this prediction process by analyzing features of the patients' EEG recordings during coma with the aim to determine quantitative metrics which are predictive of patients' outcome. Specifically, we focus on the analysis of the similarity of bursts during burst suppression, which has been hypothesized to be linked with poor outcome. Our work confirms that similarity of bursts is indeed linked with poor outcome, and we also find that dynamic time warping gives a viable alternative to the previously used method of cross-correlation as a measure of similarity of bursts, with good predictive power for patient outcome.

Contents

List of Figures

Chapter 1

Introduction

Over 300,000 incidents of cardiac arrest occur every year in the United States. Approximately 80% of people who are successfully resuscitated from the incident remain unconscious for some amount of time after return of spontaneous circulation Marion [2009]. In fact, two thirds of the mortality in patients whose cardiac function is able to stabilize is due to the brain injury suffered during the cardiac arrest. Predicting whether or not a patient will wake up from coma, as well as the patient's neurological function after waking up, is an important task in guiding treatment decisions for physicians and family of the patient.

Until recently, much of this prognostication has been largely conducted manually by physicians, by a combination of tests of patients' reactivity to various stimuli and visual analysis of the patients' electroencephalogram (EEG) recordings. Both of these methods are often quite subjective, and they only capture the patients' neurologic state at the moment of the test Bateman [2001]. Developing effective quantitative

measures based on the entire EEG history would allow physicians to provide more accurate, objective prognoses, allowing medical teams to better focus care on patients who need it. Improving these quantitative predictions of outcome likelihood has thus been the major aim of this thesis project, with a primary focus to analyze specific features of the EEG across patients to determine their relevance in predicting outcome.

One of the features of the EEG which physicians and researchers often look to in this problem is the presence of burst suppression. Burst suppression is a type of pattern which occurs in EEG recordings, characterized by periods of little activity (known as suppressions), alternating with periods of high amplitude (bursts). Many patients in coma from cardiac arrest display some form of this pattern in varying severity for some amount of time, particularly in the earlier stages of recovery from the cardiac arrest. Generally, as patients recover, they will evolve from having an EEG which is nearly completely suppressed to an EEG which is nearly completely bursts. The more quickly this evolution occurs, the more likely the patient is to have a good outcome Noirhomme et al. [2014]. On the other hand, if a patient does not show this evolution at all and instead has an EEG which remains largely suppressed over time, this is a strong negative predictor for the patient's outcome. For this reason, physicians have traditionally used the presence of burst suppression as an indicator for poor prognosis Hofmeijer et al. [2014].

Indeed, in the research world, Cloostermans et al. [2012] has shown that burst suppression can occur pathologically in comatose patients in the first 48 hours after cardiac arrest. However, burst suppression can also be induced or altered in severity by drugs Yoon et al. [2012]. In fact, about 20% of patients showing burst suppression

after cardiac arrest end up with good neurological outcome Cloostermans et al. [2012]. Because of this, it is important to discover a way of discriminating the burst suppression cases which have poor outcome from those with good outcome, rather than the traditional approach of delivering poor prognosis for all the burst suppression cases.

It has been shown by one research group that when burst suppression occurs in a patient, if the bursts appear to be identical, the patient is likely to have a poor outcome Hofmeijer et al. [2014]. This work used cross-correlation to measure burst similarity and showed that patients with extremely similar bursts nearly certainly had poor outcome. Our project first confirmed this finding and then improved upon it by finding a better measure of similarity than the cross-correlation measure previously used. In particular, we investigated the use of dynamic time warping to define similarity between burst signals. Dynamic time warping is a common method of measuring similarity between signals which allows the signals to be stretched and compressed in time to find an optimal match between them. One of its advantages over other similarity measures is thus that it is more robust to differences in speed in the two signals. We believe that this advantage allows it to better capture similarity between bursts compared to the existing cross-correlation method.

To perform our analysis, we found burst suppression episodes within a dataset of about 460 post-cardiac coma patients. Of the approximately 200 patients who displayed at least one burst suppression episode, we computed the similarities using both dynamic time warping and cross-correlation for all the pairs of bursts within each burst suppression episode. Among patients with burst suppression, our results show that these similarity numbers tend to be higher for patients with poor outcomes. In addition, we show that the burst similarity is indicative of poor outcome

on a continuous scale, rather than just binary. In other words, the more similar a patients' bursts are, the more likely the patient is to have a bad outcome. We show that cross-correlation is better when used on a continuous scale, whereas dynamic time warping similarity is more useful as a threshold; by thresholding on the dynamic time warping similarities, we are able to correctly identify about 40% of the patients with poor outcome with no false negatives (See Section 4.2.3).

The final contribution of this project is the release of all the source code used to conduct our analysis, making our burst suppression analysis tools available to the public in the hope that other researchers interested in this problem can build upon the work presented here.

The rest of this document will be organized as follows: first, we give an introduction to previous work conducted in this space, describing research on the general outcome prediction problem using EEG's, as well as specific research work focused on the burst suppression portion of the problem. Next, discuss the technical approach, giving details on the methodology of each step of the project. Finally, we present results and figures from each step of our methodology pipelines, focusing primarily on the results of the final similarity measures. In the appendix, we give an overview of the released code repository, along with detailed installation, usage, and development guides. We also make some remarks about issues specific to the dataset we used and give a guide to future users of our specific dataset.

Chapter 2

Related work

2.1 General EEG coma prediction models

Much of the previous which has been conducted in this space has sought to determine which features might be relevant in predicting patient outcomes. The papers which do this generally use statistical correlation to analyze and quantify the significance of the features they study. Here we provide a summary of this work and the features studied.

One such feature which has been studied is the neural network connectivity implied by the EEG Beudel et al. [2014]. For instance, Beudel et al. [2014], used the similarity between frequency strengths to create a connectivity graph representing the signals. They studied traits of this graph, such as the number of network nodes and connections, the clustering coefficient, and the average path length; the results showed that these features were helpful in predicting outcome. Forgacs et al. [2017] studied the anterior forebrain corticothalamic circuit integreity by visually categorizing the

power spectra of the EEG into four degrees of integrity, which they showed were linked to patient outcome. Moshirvaziri et al. [2016] studied a metric quantifying the complexity of the EEG signal, looking at the wavelet sub-band entropy, and found that the brain high-frequency oscillations were significantly more complex in the patients with better outcomes. Finally, Noirhomme et al. [2014] studied a variety of different features, including approximate entropy of the signal, the burst suppression ratio, and reactivity to stimulation, and concluded that these quantitative measures provide promising information in aiding clinical prognosis.

Other researchers, such as Temko et al. [2015] and Tjepkema-Cloostermans et al. [2017] have focused on improving the ways in which all the features are combined in making one final prediction. The Temko and Cloosterman groups, in particular, focus on using machine learning methods to train classifiers based on the features input.

2.2 Burst suppression coma prediction

Within the broader problem of predicting outcome using the EEG, several researchers have been focusing their studies on the analysis of burst suppression in these patients. Much of this work has focused on addressing the problem of discriminating burst suppression with good outcome from burst suppression with bad outcome. Some researchers have previously made strides in addressing this challenge. Hofmeijer et al. [2014] noticed that in some burst suppression cases in patients post-cardiac arrest, the burst shapes were extremely similar. They termed this phenomenon "burst suppression with identical bursts" and showed that it occurs exclusively after diffuse cerebral

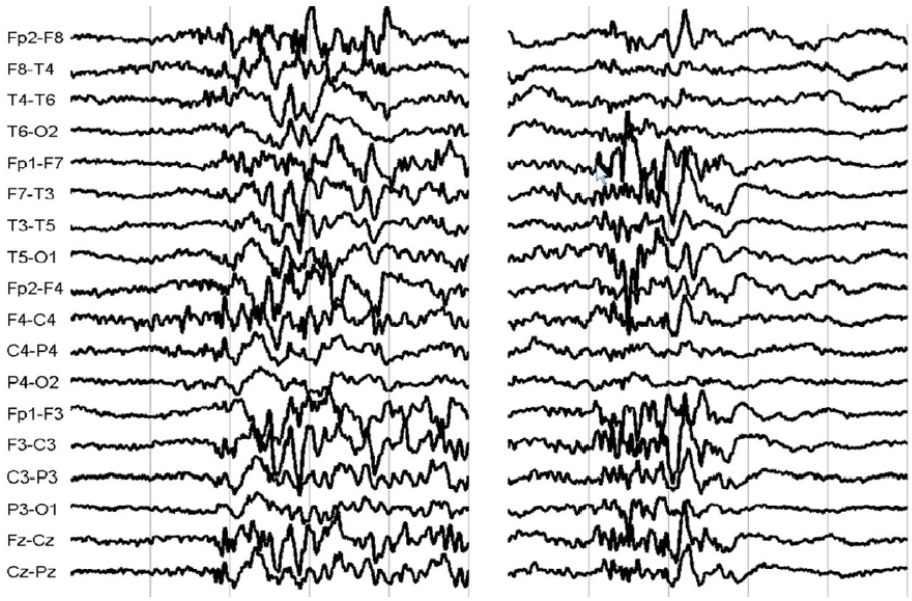


Figure 2.1: A example of an EEG signal with burst suppression with non-identical bursts

ischemia and is associated with poor outcome in patients Hofmeijer et al. [2014]. Figures 2.1 and 2.2 show EEG signals of two patients; Figure 2.1 displays an example of burst suppression with non-identical bursts, and Figure 2.2 is an example of burst suppression with identical bursts.

2.2.1 Burst similarity quantification

Hofmeijer et al. [2014] has detailed a quantitative method used to identify burst suppression cases and measure the similarity between the bursts. They manually annotated 50 subsequent bursts for each patient displaying burst suppression. They truncated each burst to 500 milliseconds. For each pair of bursts, the "similarity" was taken to be the maximum cross-correlation value taken over a range of lags (from $-0.5s$ to $0.5s$, for sample rate s), where the cross-correlation between bursts b_1 and

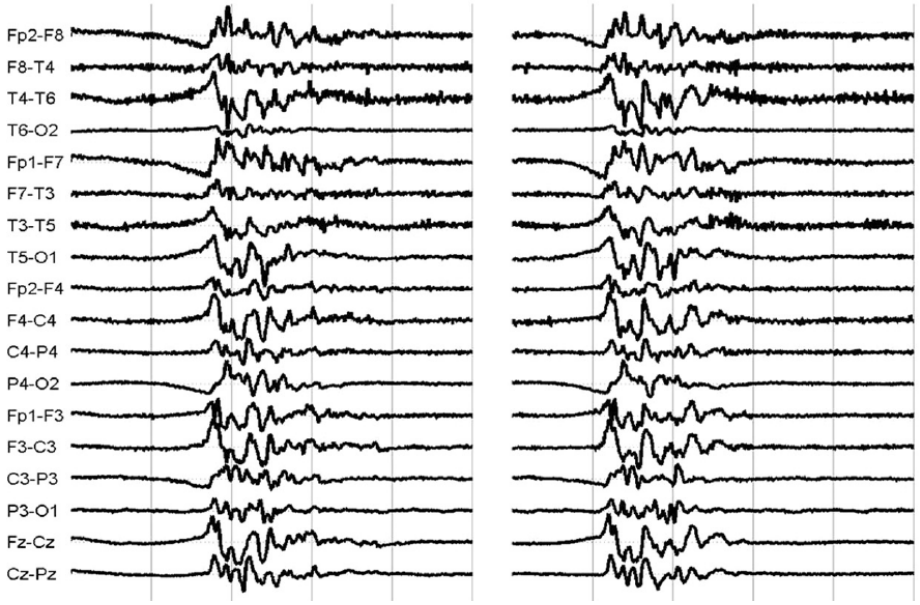


Figure 2.2: A example of an EEG signal with burst suppression with identical bursts

b_2 at lag l is calculated as:

$$(b_1 \star b_2)[l] = \sum_i b_1(i) b_2(i+l)$$

The final cross-correlation between two bursts is taken as the maximum cross-correlation for any lag:

$$(b_1 \star b_2) = \max_l (b_1 \star b_2)[l]$$

This resulted in one cross-correlation value for each pair of bursts; since there were 50 bursts per patient, each patient ended up with $\binom{50}{2} = 1225$ correlation values. Histograms of these correlations for patients with good outcome showed significant difference in both shape and mean from those with bad outcome.

A later paper by the same group Tjepkema-Cloostermans et al. [2017] built upon this work to algorithmically extract three features of the patient based on the simi-

larity of their bursts during burst suppression. In this paper, rather than manually annotating bursts, they calculated the envelope of the EEG signal using a Hilbert transform and defined a burst as any increase of at least 5 microvolts in the envelope. They calculated cross-correlations between all pairs of bursts, and summarized these numbers into three features: the mean burst correlation, the max burst correlation, and the fraction of correlations greater than 0.8. These three features were incorporated into a larger set of features used to predict patient outcome. On their dataset of post-cardiac arrest patients, they showed that these three features taken at 48 hours into the EEG are significantly predictive of patient outcomes.

This work was later built upon in a recent paper by the same group van Putten et al. [2018]. In this paper, rather than using specific clinical signals from the EEG, van Putten et al. [2018] inputted the EEG signal itself into a deep learning framework. They trained a convolutional neural network to predict patient outcome based on the inputted EEG signal and showed that the network achieves an AUC score of 0.89 on the task of distinguishing patients with poor outcome from those with good outcome.

To our knowledge, no previous work has been conducted on alternative methods of quantifying the similarity besides cross-correlation. We hypothesized that improved alternatives do exist, and the purpose of this project has been to investigate them and measure their efficacy.

Chapter 3

Methods

In this section, we first give an overview of our EEG analysis pipeline, before moving on to detailed descriptions of each step in the following sections.

3.1 Pipeline Overview

Figure 3.1 gives an overview of our EEG analysis pipeline. There are 5 main steps:

1. Preprocessing - We perform some basic cleaning of the EEG signals before any downstream analysis.

2. Detecting burst suppression - This step involves determining which patients display burst suppression, and at what times.

3. Describing burst suppression episodes - Before looking at the burst similarities, we would like to understand some basic information about our burst suppression

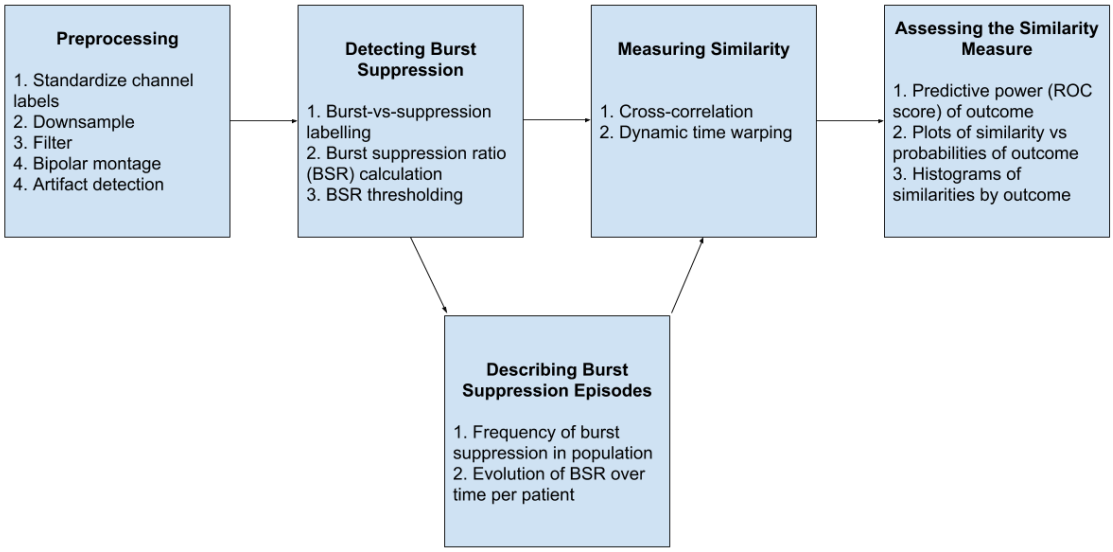


Figure 3.1: This diagram outlines the pipeline of our project, and the substeps involved in each portion.

cases such as the typical length of a burst suppression episode in order to better guide our similarity analysis.

4. Measuring similarity within the burst suppression episodes - This represents the most important portion of our analysis. For each burst suppression episode found, we compute similarity numbers on all the pairs of bursts within the episode.

5. Assessing the similarity measure - Using the similarity numbers computed, we analyze how well, if at all, they are correlated with patient outcome.

3.2 Preprocessing

The first step in the pipeline is preprocessing of the data. Before EEGs are used to show to physicians, it is necessary to perform some standard preprocessing steps. We preprocess our EEGs in a similar manner to reproduce the data as a physician would see it. For each EDF (European Data Format) file in our dataset containing a segment of EEG recording data, we perform the following preprocessing:

1. Standardize the channel labels across hospitals
2. Downsample the signal to a common sample rate across all segments
3. Filter the signal data
4. Create a bipolar montage across the channels
5. Detect artifacts

3.2.1 Nomenclature standardization and downsampling

Each hospital in our dataset used different nomenclature for their labels of the EEG channels, so the first step was to standardize all of these labels into a common nomenclature. For the downsampling of the signal, we choose a common sample rate of 200Hz, which was the lowest sample rate among all the segments in our dataset.

3.2.2 Filtering and Bipolar Montage

The filtering of the signal data consists of three filters:

1. High pass filter - 0.5 Hz

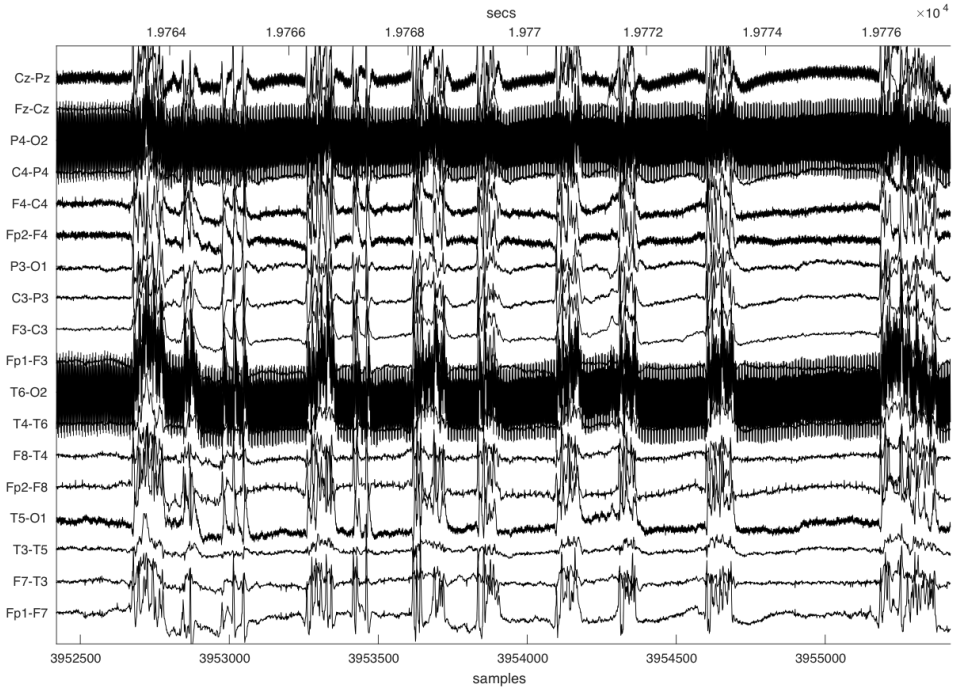
2. Notch filter - 60 Hz

3. Low pass filter - 50 Hz

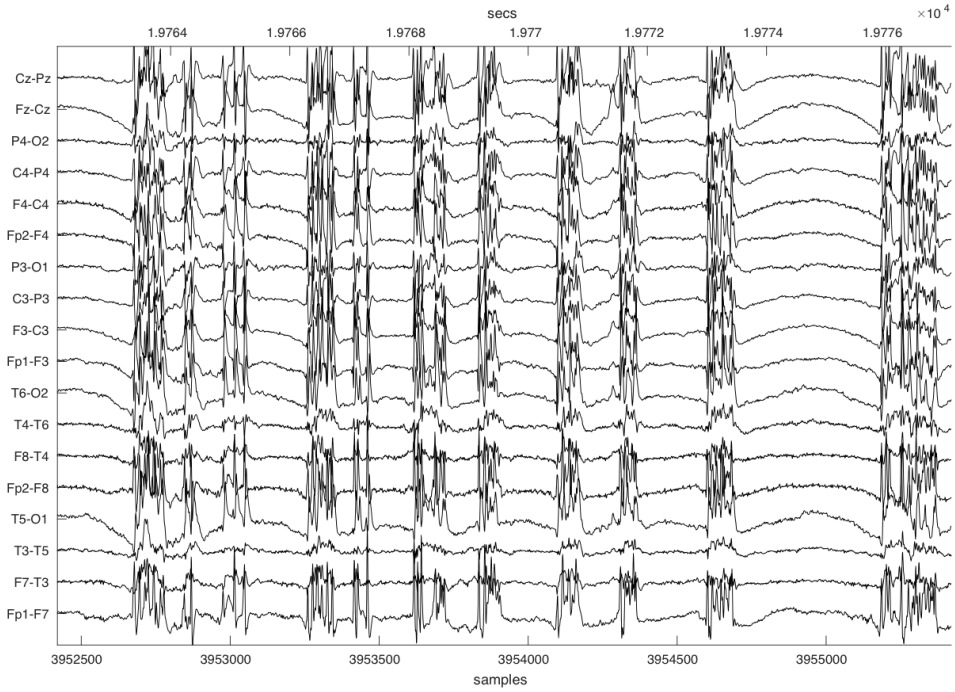
The high pass filter removes long-term trends from the signal, while the notch filter is designed to remove powerline noise. All three filters are implemented as Butterworth filters in MATLAB. The high and low pass filters are 4th-order Butterworth filters, and the notch filter is a 6th-order bandstop Butterworth filter acting between 55 and 65 Hz.

Following the filters, we create a bipolar montage of the EEG channels. The bipolar montage is a method of defining a reference voltage for the electrical signals, and is widely used by EEG monitors to display a patient's EEG to physicians. We define new bipolar montage channels to be the voltage difference between two of the original channels' data, creating a total of 19 new bipolar channels: Fp1-F7, F7-T3, T3-T5, T5-O1, Fp2-F8, F8-T4, T4-T6, T6-O2, Fp1-F3, F3-C3, C3-P3, P3-O1, Fp3-F4, F4-C4, C4-P4, P4-O2, Fz-Cz, and Cz-Pz.

Figure 3.2 shows an example of an EEG signal before and after these preprocessing steps. Both the before and after plots are shown with bipolar montage referencing for clearer comparision. As can be seen, the preprocessing significantly cleans up the signal and allows for clearer analysis. For example, this signal in particular benefits greatly from the removal of high frequency noise.



(a) Before preprocessing



(b) After preprocessing

Figure 3.2: An example of an EEG signal before and after the preprocessing (not including artifact detection). Note that the bottom x-axis is in samples, while the top x-axis is in seconds, with a sample rate of 200 samples per second.

3.2.3 Artifact Detection

Once the EEG signal is filtered, we need to detect artifacts in the recording. This portion of the preprocessing identifies segments of the EEG signal which are affected by electrical activity outside of the patient's brain, and therefore should be excluded from our analysis. To perform artifact detection, we divide the signal into chunks of 5 seconds. For each channel in each chunk, we check for two types of artifact:

1. Too much signal - We define this as when the maximum absolute value of the voltage within the chunk is greater than a threshold value of 500 microvolts

2. No signal - We define this as when the standard deviation of the voltage within the chunk is smaller than a threshold value of 0.0001 microvolts

If the signal in any channel in a chunk contains artifact, we mark all samples in all channels within that chunk as being artifact. The end result is a vector a marking whether or not each time sample is an artifact. In other words, $a_t = \vee_j A_t^{(j)}$, where $\vee_j$ represents the logical or operator taken over all j, and $A_t^{(j)}$ indicates whether or not sample t in channel j is detected as artifact.

Figure 3.3 shows an example of the artifact detection pipeline marking a portion of an EEG as having no signal. This 45 second chunk captures the moment when the EEG machine actually begins recording, and the pipeline correctly marks everything before this as artifact.

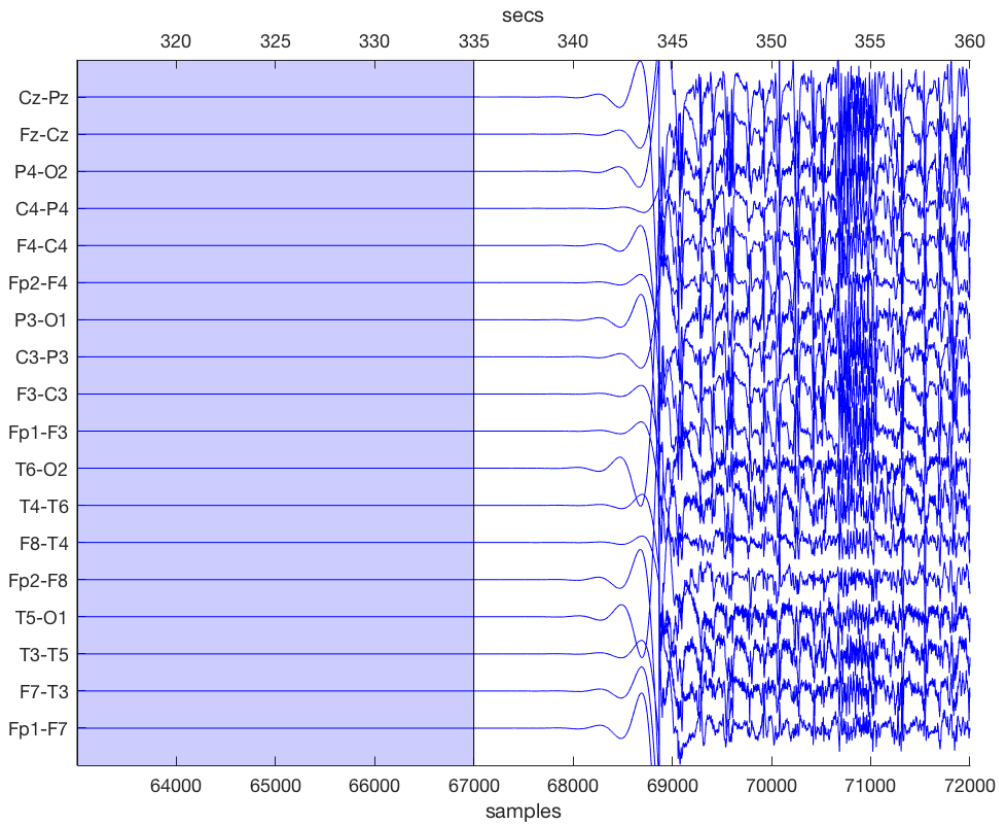


Figure 3.3: An example of the artifact detection pipeline marking a portion of an EEG as having no signal. The blue shaded region represents the artifact region. Note that the bottom x-axis is in samples, while the top x-axis is in seconds, with a sample rate of 200 samples per second.

3.3 Burst Suppression Detection

3.3.1 Burst-vs-Suppression Labels

We detect burst suppression according to a method described by Westover et al. [2013]. The method involves first labelling each time step as part of either a burst (1) or a suppression (0). This is first done locally within each channel by calculating a recursive local variance and thresholding bursts as the time points with high enough variance, according to the following equations:

$$\mu_t^{(i)} = \beta\mu_{t-1}^{(i)} + (1-\beta)x_t^{(i)}$$

$$v_t^{(i)} = \beta v_{t-1}^{(i)} + (1-\beta)(x_t^{(i)} - \mu_t^{(i)})^2$$

$$z_t^{(i)} = \mathbb{1}[v_t^{(i)} < \theta]$$

, where the input $x_t^{(i)}$ represents the signal at time t in channel i, and the output $z_t^{(i)}$ represents the local burst-vs-suppression label for the channel. We initialize $\mu_0^{(i)} = x_0^{(i)}$ and $v_0^{(i)} = 0$. β governs the weight of the past values in the current local estimate and is related to the forgetting factor τ: $\beta = e^{-1/(\tau * s)}$, where s is the sample rate. τ, as well as the threshold θ on the local variance, are parameters of the algorithm, and they are set to the optimal values found by Westover et al. [2013], namely, 0.1047, and 1.75, respectively.

The local burst-vs-suppression labels (also referred to as the "local zs") are aggregated into a global burst-vs-suppression label (also referred to as the "global zs")

across all channels through a simple voting procedure of all the channels.

$$z_t^{GLOBAL} = \mathbb{1}[\frac{1}{n}\sum_{i=1}^{n} z_t^{(i)} \geq \text{agree-percent}]$$

In other words, the global burst-vs-suppression label is set to 0 if at least "agree-percent" of the channels label the time step as a suppression, where the "agree-percent" parameter will be set the value recommended by An et al. [2015], namely, 60%.

Once the global burst-vs-supppression labels are computed, we perform some smoothing on the labels. We define that a suppression cannot last for fewer than 0.5 seconds, so any continuous time ranges where the burst-vs-suppression label is 0 for fewer than $0.5 * srate$ samples is considered to not be a true suppression, and the labels are flipped to 1.

3.3.2 Burst Suppression Ratio

The next step is to compute the burst suppression ratio using the global burst-vs-suppression labels. The burst suppression ratio (BSR) r is defined as the percentage of time steps labelled as suppression in a sliding window of 1 minute.

$$r_t = 1 - \frac{1}{60s}\sum_{t'=\min(0,t-60s)}^{t} z_t^{GLOBAL}$$

, where s is the sample rate in samples per second.

If any of the points within the window are marked as being artifact, then the BSR

value for the window is considered to be undefined.

3.3.3 Burst Suppression Episodes

Using the burst suppression ratio, we can define a range of time to be an instance of burst suppression if the suppression percentage is greater than a certain threshold value. Specifically, if $r_t \geq 0.50$, we label the range $t - 60s$ to t as being part of a burst suppression episode. While we recognize that it is possible to display burst suppression with burst suppression ratios lower than 50%, we choose to focus on the cases within our threshold as the most interesting ones—in particular, the ones which may contain identical bursts.

We define one burst suppression episode as a contiguous time range t_{start} to t_{end} where $r_{t_i} \geq 0.50$ for all t_i in $[t_{start}, t_{end}]$. We apply some smoothing to this, so that if there are two consecutive time ranges $[t_{i_1}, t_{j_1}]$ and $[t_{i_2}, t_{j_2}]$ such that $i_2 - j_1 \leq \delta s$, for some small duration in seconds δ and sample rate s, we consider the whole time range $[t_{i_1}, t_{j_2}]$ to be one continuous burst suppression episode. We set $\delta = 1$ second. Most burst suppression episodes are on the order of many minutes long (see 4.2), so it is reasonable that a dip in the BSR below our threshold value for only one second does not count as a new episode. In addition, we define burst suppression episodes to be longer than a minimum duration of 5 seconds, so any time ranges shorter than this are removed from the final set of burst suppression episodes. This is a very liberal miminum duration for a burst suppression episode. In later steps of our analysis, we filter the episodes more heavily, so that only episodes which are half an hour or longer in duration are actually considered in the final results, so in this step, we can simply

keep virtually all the detected episodes.

3.3.4 Burst Suppression Detection Visualization

Figure 3.4 presents a visualization of our burst suppression detection pipeline. The EEG in the figure displays burst suppression, and one can visually verify that the global burst-vs-suppression label follows the signal, and that the burst suppression ratio falls within our threshold for burst suppression detection. Figure 3.5 presents a similar plot, but with only one EEG channel and a cleaner, zoomed version of the BSR and z_t^{GLOBAL} signals. This visualization colors the EEG during any burst suppression episode as pink, highlighting the burst portions of the episode with a deeper red color. Again, we verify that the signal plotted in the figure is marked as burst suppression and colored pink, as desired, and that the expected burst regions are properly identified and colored red.

3.4 Describing the Data

Once we can find burst suppression episodes, we can run some basic descriptive analyses on the episodes themselves. This is to help us gain better understanding of what the episodes and bursts tend to look like, which guides us in making decisions about parameters in later pipeline steps. We plot histograms of

1. the number of episodes per patient

2. the duration of episodes across all patients

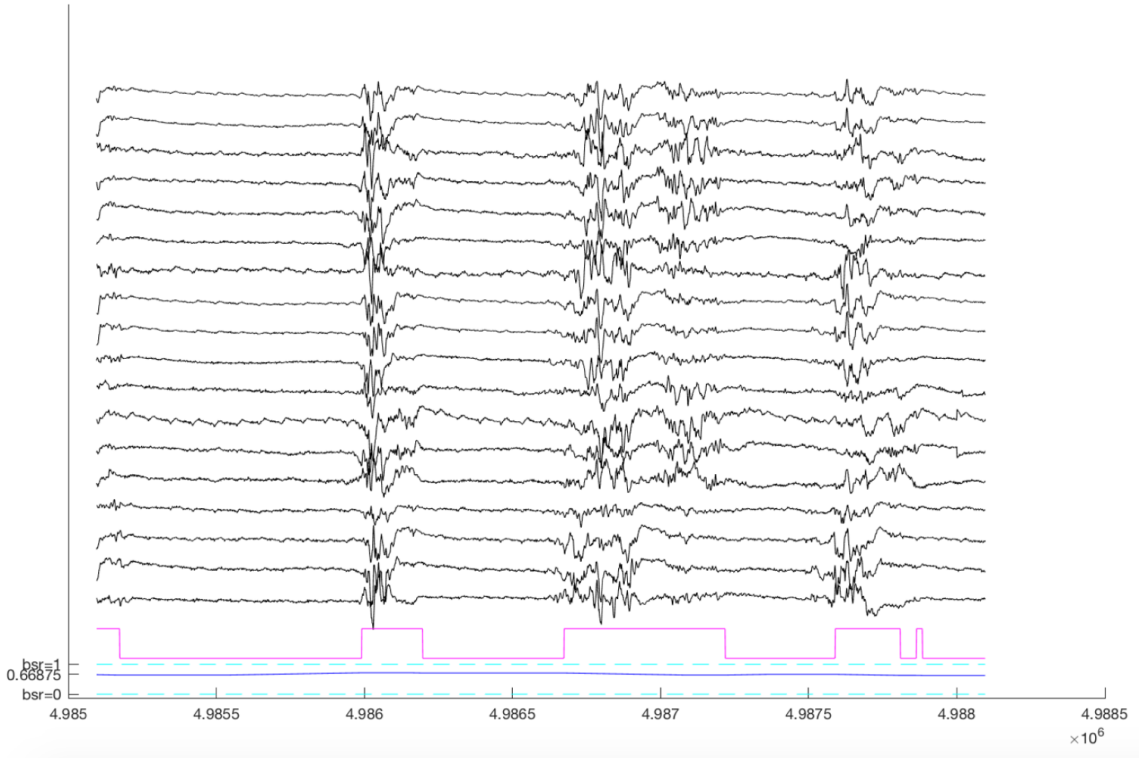


Figure 3.4: A example of an EEG signal along with its computed burst-vs-suppression label and the burst suppression ratio. The pink line represents z_t^{GLOBAL}, and the blue line represents r_t. The calculated burst suppression ratio in this plot is around 67%, and one can visually verify that indeed, about 2/3 of the plot time is in suppression. The x-axis is in units of samples, with a sample rate of 200 samples per second.

Figure 3.5: An EEG signal along with its z^{GLOBAL} and r signals. The pink shading represents that this portion is detected as burst suppression, and the red shading represents regions labeled as bursts. The x-axis is in units of samples, with a sample rate of 200 samples per second. The plot depicts a patient's signal from start index 251970 to end index 253470.

3. the duration of bursts across all patients and episodes

Using these histograms, we can better answer questions such as:

1. What is the minimum duration of a burst suppression episode that a patient must have in order to be considered as a "burst suppression patient"?

2. What is the maximum duration a burst can have before it is considered too "abnormal" to be used to our similarity analysis?

We present these histograms in section 4.1.1.

3.5 Measuring burst similarity

Once burst suppression episodes are found, we calculate a similarity measure for the bursts in each episode. For each episode, we first take all the bursts (defined as continuous regions where the z^{GLOBAL} signal equals 1). We filter out bursts which are too short or too long to be considered a true burst, keeping only bursts whose lengths are between $min\_burst\_time = 0.1$ seconds and $max\_burst\_time = 5$ seconds. We chose these parameters based on the recommendation of the neurologists who were collaborating with us. Before computing similarities, we normalize each burst individually by its mean and standard deviation:

$$\hat{x}[i] = \frac{x[i] - \mu}{s}, \qquad \forall i = 1, 2, ..., (n := len(x))$$

where $\mu = 1/n \sum_{i=1}^{n} x[i]$ is the mean value of the burst and $s = \sqrt{1/n \sum_{i=1}^{n} (x[i] - \mu)^2}$ is the variance. We normalize each burst individually, rather than by the entire burst

dataset, because it is has been shown that normalization of an entire dataset gives suboptimal results compared to individual normalization when using dynamic time warping Rakthanmanon et al. [2012].

After normalization, we trim all the bursts to take at most the first $burst\_trim\_time$ seconds of the burst. We try two values of $burst\_trim\_time$. The first is 500 milliseconds, in order to match the work done by Hofmeijer et al. [2014] for direct comparison with their methods. The second is 5 seconds, or no splicing, since each burst is at most 5 seconds long; this allows the similarity measure to take into account the whole burst, not just the beginning portion of it. After this splicing, we compute the similarity between all pairs of bursts within the episode using two methods: cross-correlation and dynamic time warping.

3.5.1 Cross-correlation

We use cross-correlation as our baseline method for similarity quantification, as this is the only similarity measure which has been used on EEG bursts in existing work. The cross-correlation between two signals is calculated as the dot product between the signals shifted at various lags (Weisstein):

$$(X \star Y)[l] = \sum_n X[n]Y[n+l]$$

For each burst pair (X, Y), we calculate $(X \star Y)[l]$ for all l from $-max(len(X), len(Y))$ to $+max(len(X), len(Y))$. The final cross-correlation similarity is taken to be the max of the cross-correlations at all lags; this represents the similarity when the two signals are optimally aligned. Our implementation uses the cross-correlation function in MATLAB.

3.5.2 Dynamic time warping

Dynamic time warping is a commonly used measure of distance between two signals. For two signals $X = \{x_1, ..., x_N\}$ and $Y = \{y_1, ..., y_M\}$, dynamic time warping seeks to find an optimal alignment (or warping path) of samples from X with samples from Y, and the distance between X and Y is given as the distance at this optimal alignment. Formally, DTW finds an alignment $p = \{(n_1, m_1), ..., (n_L, m_L)\}$ which minimizes

$$C_p(X, Y) := \sum_{l=1}^{L} c(x[n_l], y[m_l])$$

where c is the local cost function which measures the distance between a sample of X and a sample of Y. We use the Euclidean distance as our local cost function:

$$c(x, y) = \sqrt{(x - y)^2} = |x - y|$$

In order to prevent unreasonable alignments, DTW constrains the set of valid alignments to meet the following requirements:

- Boundary requirement - $(n_1, m_1) = (1, 1)$ and $(n_L, m_L) = (N, M)$
- Monotonicity - $n_1 \leq n_2 \leq ... \leq n_L$ and $m_1 \leq m_2 \leq ... \leq m_L$
- Step size - $(n_{i+1}, m_{i+1}) - (n_i, m_i) \in \{(0, 1), (1, 0), (1, 1)\}$
- Warp amount - No element in the alignment can be further than *maxsamp* samples from the straight-line alignment between X and Y. *maxsamp* is a parameter of the DTW algorithm, which we set to be 200, corresponding to 1 second with a sample rate of 200Hz.

With these constraints on alignments, the final DTW distance between X and Y is given as

$$DTW(X, Y) = \min_{p \in P_{valid}} C_p(X, Y)$$

where P_{valid} is the set of all valid alignments. For our implementation, we use the DTW function in MATLAB.

Chapter 4

Results

In this section, we begin by presenting some results from running burst suppression detection on our dataset. This includes histograms depicting the burst suppression episodes and bursts, as well as plots describing the BSR for all the patients over itme. We then follow by presenting our main results on burst similarity. This includes comparisons of the histograms of similarities for good and bad patients, as well as plots depicting how predictive the similarities are of patient outcome. In the results that follow, we use "good" outcome to refer to an outcome of 1 or 2 on the Cerebral Performance Categories (CPC) scale, and "bad" outcome to refer to an outcome of 3, 4, or 5 on the CPC scale (see Section B in the appendix for details) Safar and Grenvik [1981].

4.1 Describing Burst Suppression Episodes

Running burst suppression detection on our dataset revealed that 228 out of 460 patients had at least one burst suppression episode, and 197 out of 460 displayed at least one burst suppression episode which lasted at least 30 minutes.

4.1.1 Histograms

We plotted histograms describing the number of burst suppression episodes, their durations, and the durations of their bursts. Figure 4.1 depicts the distribution of the number of burst suppression episodes each patient had. It reveals that the majority of patients had no burst suppression episodes, which is to be expected, as burst suppression is considered a relatively rarer occurrence. It also reveals that a number of patients had one to ten episodes; it seems that if a patient has burst suppression, he or she is likely to have multiple episodes. Figure 4.2 depicts the distribution of the duration of burst suppression episodes for all episodes across all patients. The vast majority of episodes are less than 2 hours long, and about half of the episodes are less than 30 minutes long. Figure 4.3 depicts the distribution of the duration of bursts for all bursts across all the episodes and patients. There are a few anomalous bursts with duration greater than 5 seconds which are cut out from this histogram. Most bursts are less than one second long, with a median burst duration of 0.125 seconds.

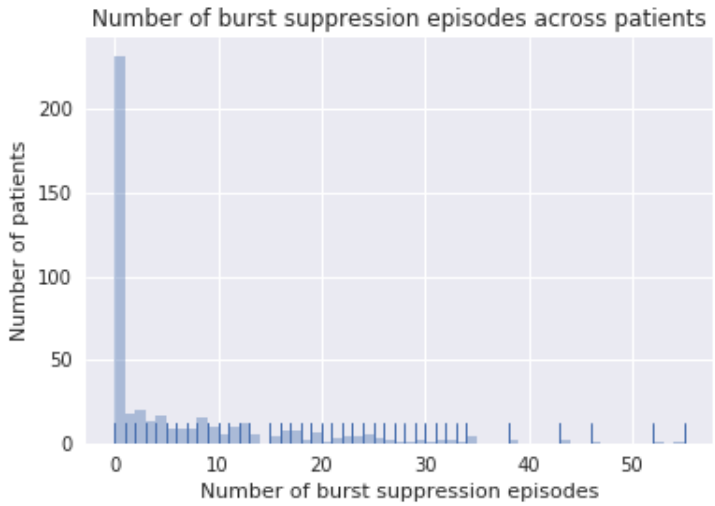


Figure 4.1: A histogram of the number of burst suppression episodes each patient displayed. There are a total of 460 patients plotted here.

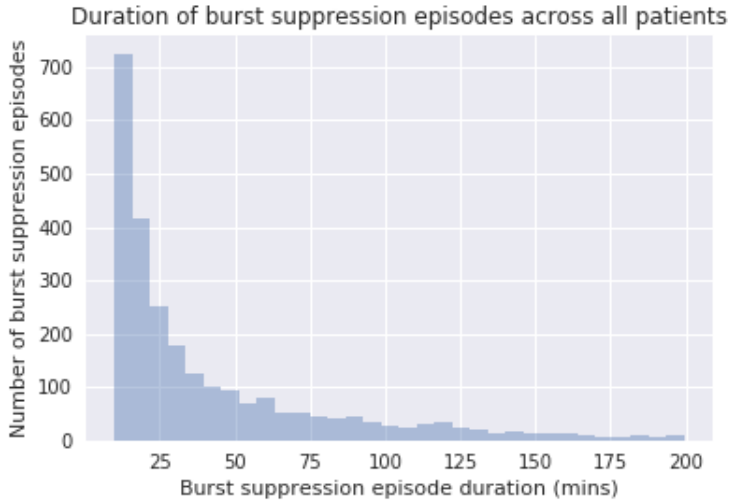


Figure 4.2: A histogram of the duration of burst suppression episodes across all patients. There are a total of 2717 episodes plotted here.

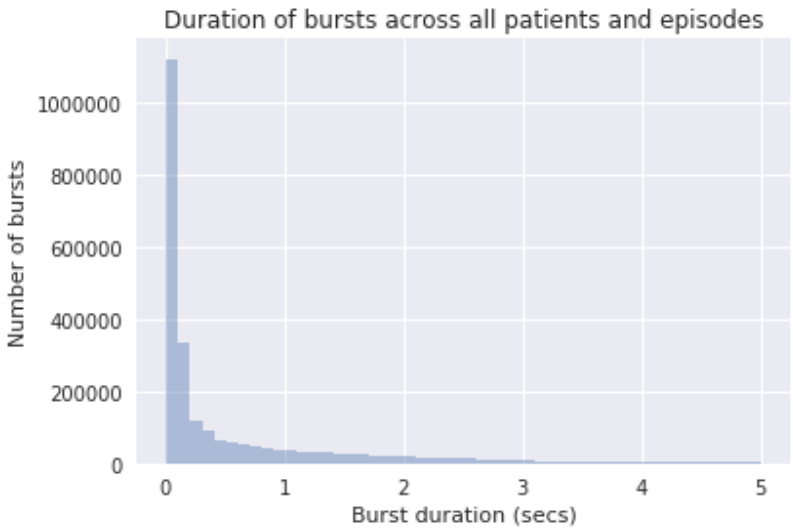
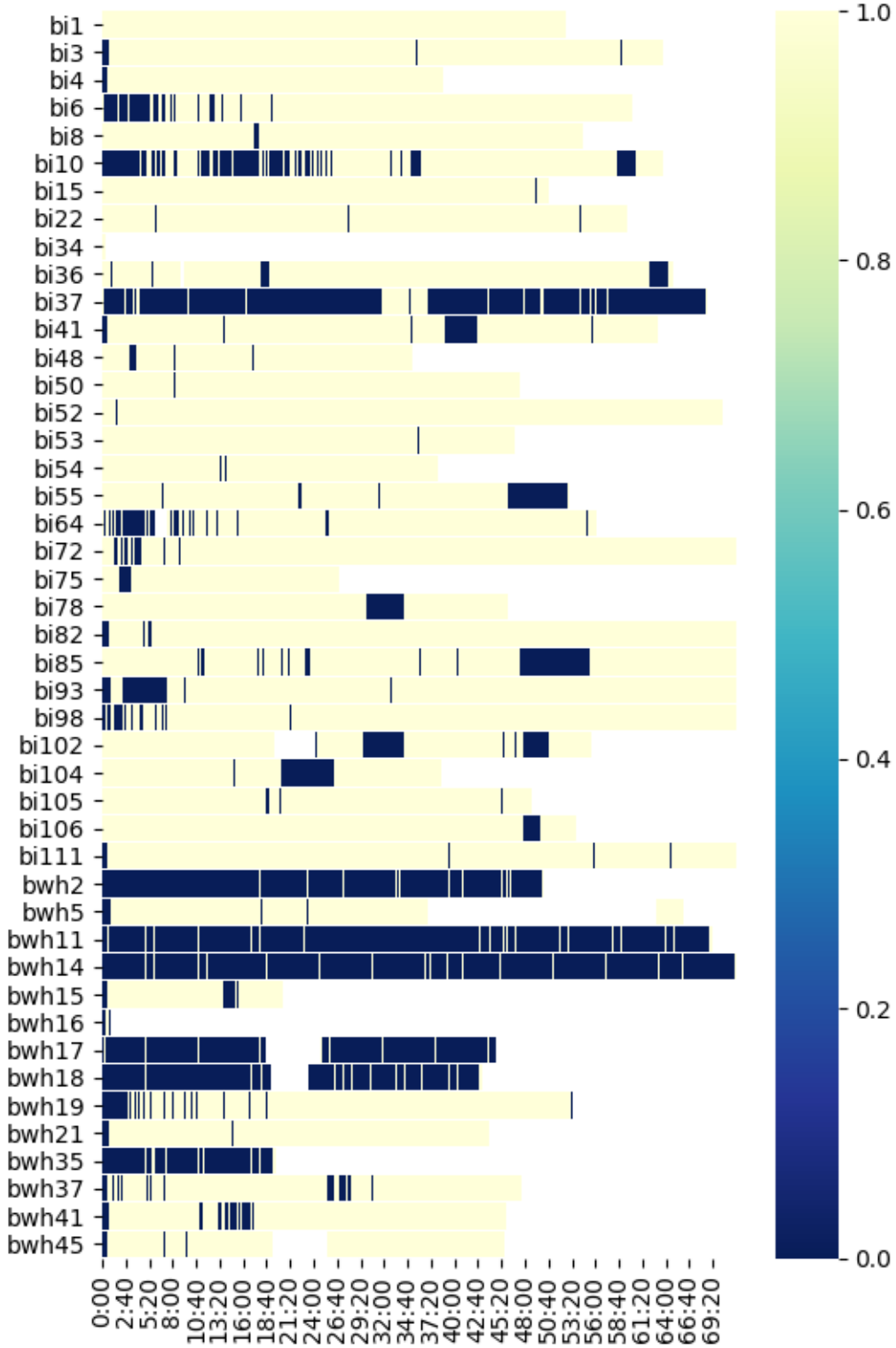


Figure 4.3: A histogram of the duration of bursts across all burst suppression episodes and patients. There are a total of 2503147 bursts plotted here.

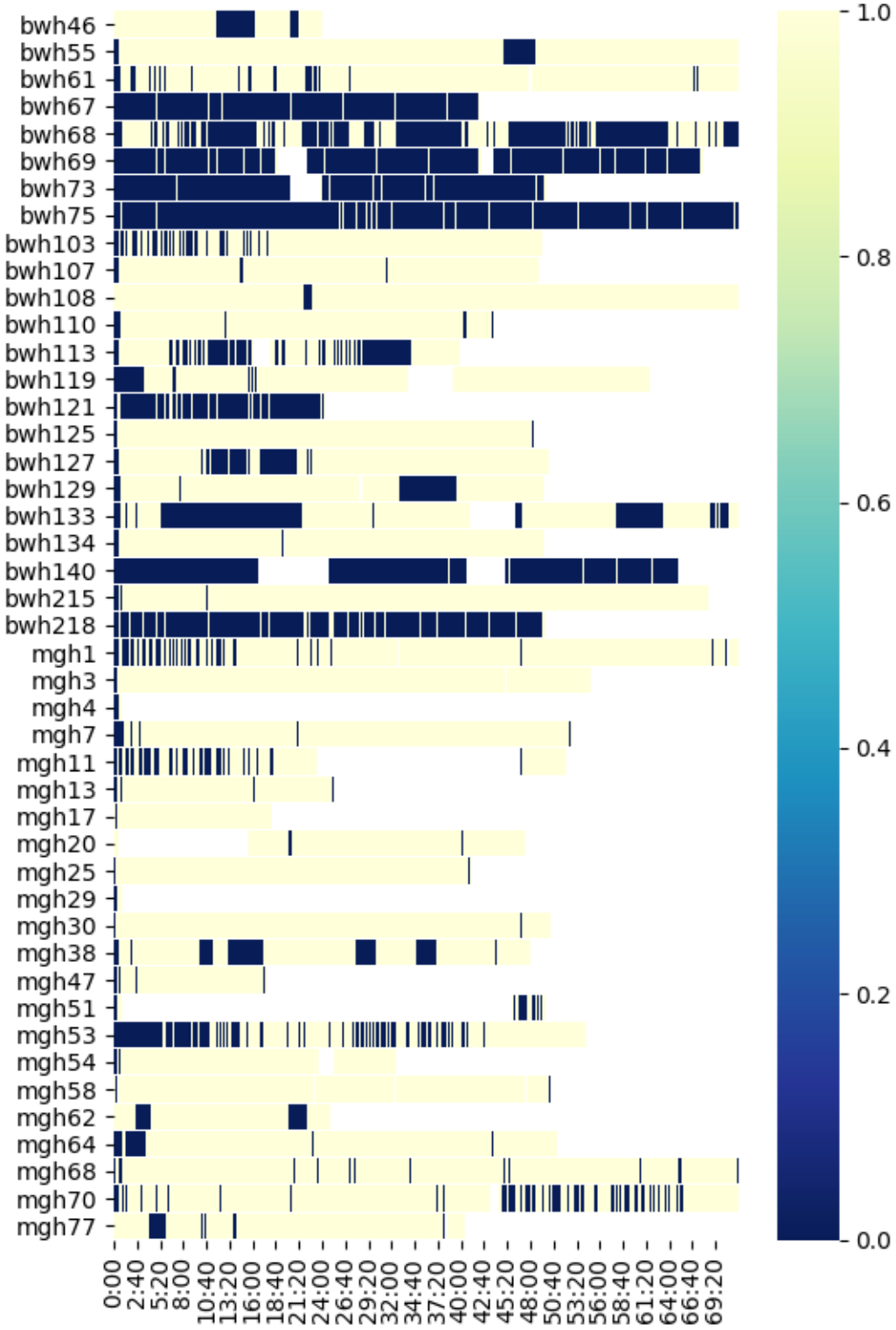
4.1.2 Plots of Global Zs and BSR Over Time

In addition to these histograms, we created colored plots to visualize the z^{GLOBAL} signal and the burst suppression ratio over time for each patient over their 72 hours of recording. We separated the plots into one for the patients with good outcome, and one for the patients with bad outcome. The plots of z^{GLOBAL} have two colors, with dark blue representing suppression periods. The plots of the BSR have a range of colors, with darker colors representing higher BSR values, or higher ratios of suppression. From these plots, we can see that the patients with bad outcomes tend to have more periods of suppression, as well as more periods of burst suppression, while the patients with good outcomes gradually recover from having suppression periods to having mostly bursts.

Figure 4.4


bi1
bi3
bi4
bi6
bi8
bi10
bi15
bi22
bi34
bi36
bi37
bi41
bi48
bi50
bi52
bi53
bi54
bi55
bi64
bi72
bi75
bi78
bi82
bi85
bi93
bi98
bi102
bi104
bi105
bi106
bi111
bwh2
bwh5
bwh11
bwh14
bwh15
bwh16
bwh17
bwh18
bwh19
bwh21
bwh35
bwh37
bwh41
bwh45
0:00
2:40
5:20
8:00
10:40
13:20
16:00
18:40
21:20
24:00
26:40
29:20
32:00
34:40
37:20
40:00
42:40
45:20
48:00
50:40
53:20
56:00
58:40
61:20
64:00
66:40
69:20
1.0
0.8
0.6
0.4
0.2
0.0

Continuation of Figure 4.4



Continuation of Figure 4.4

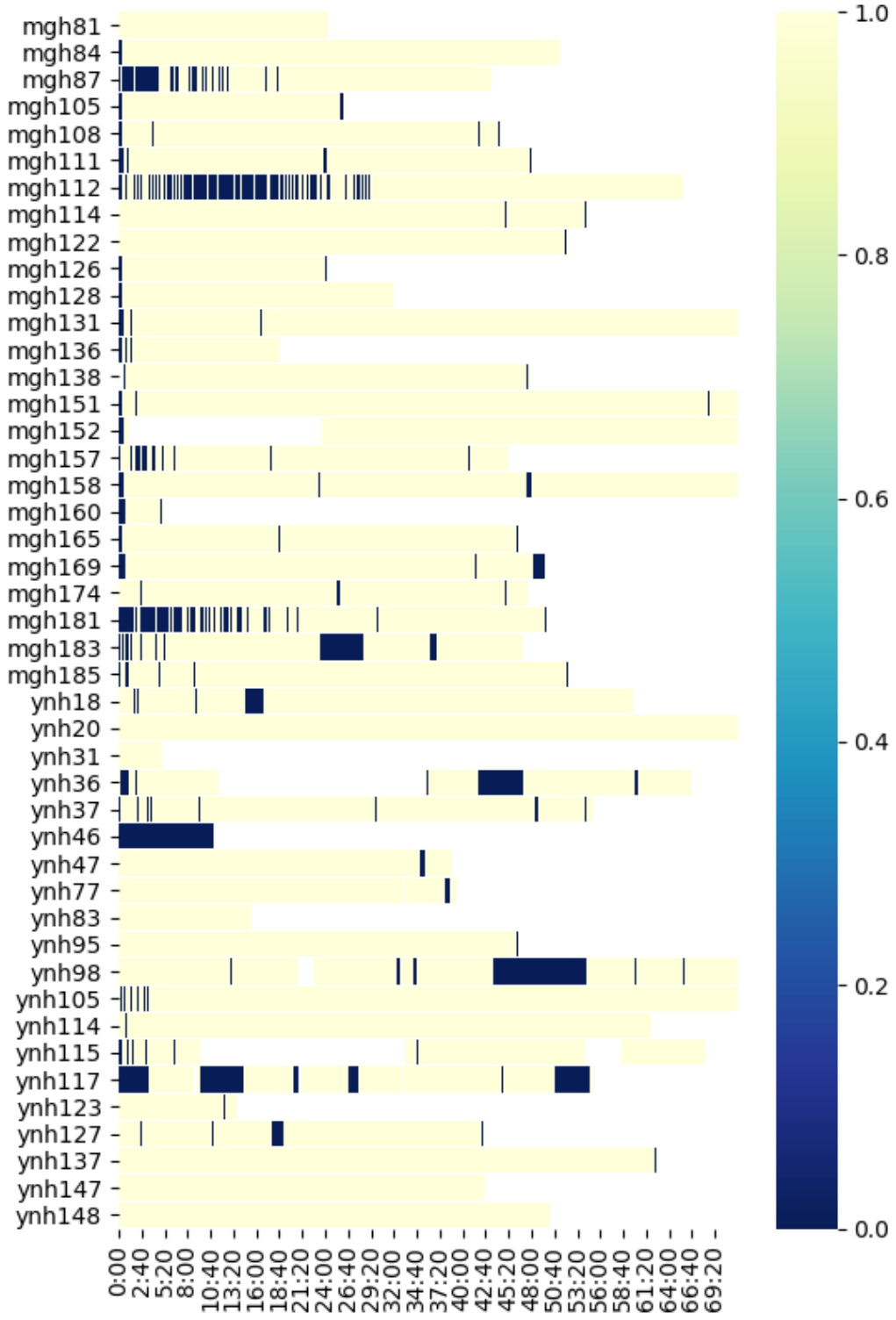
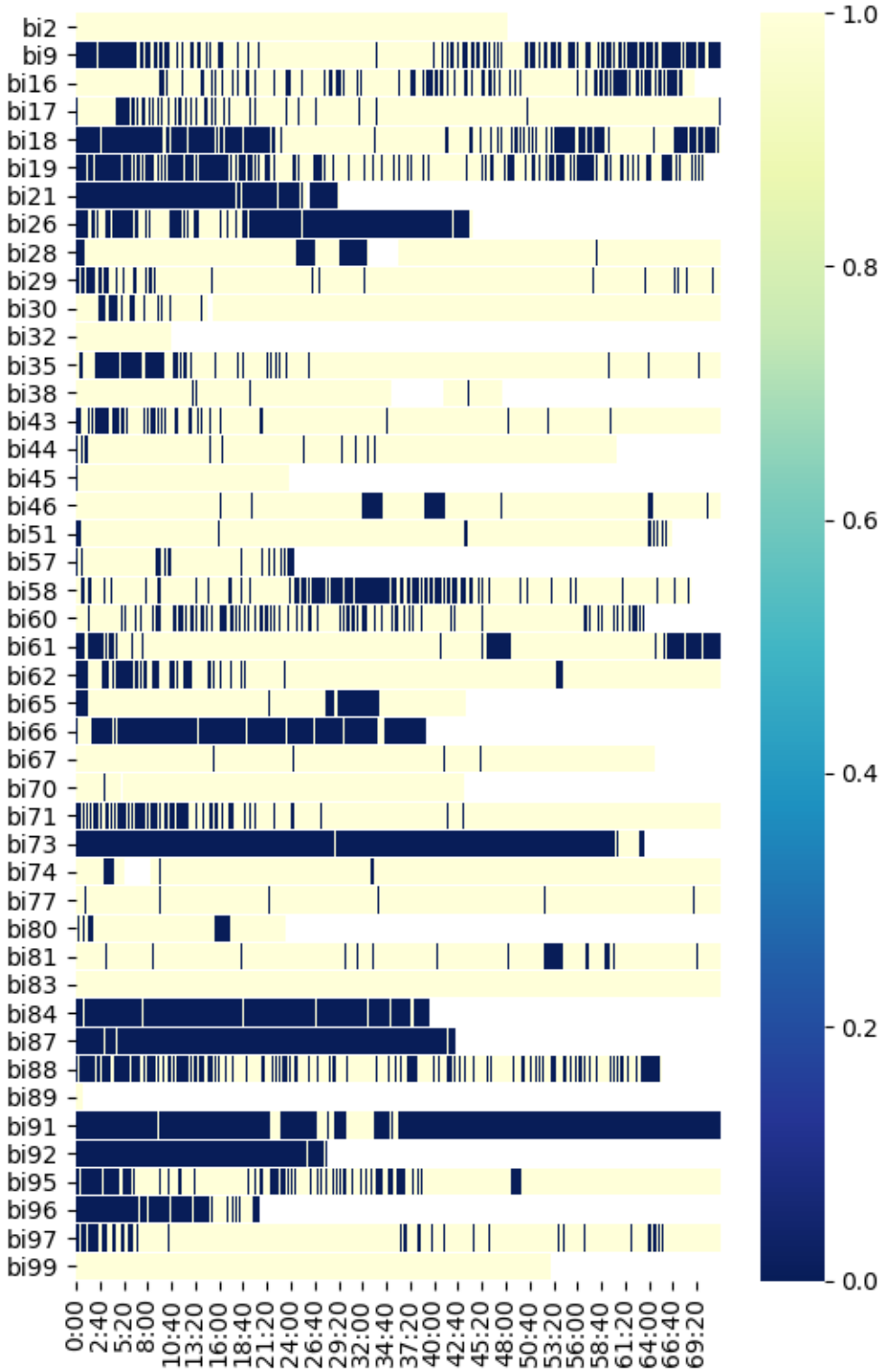
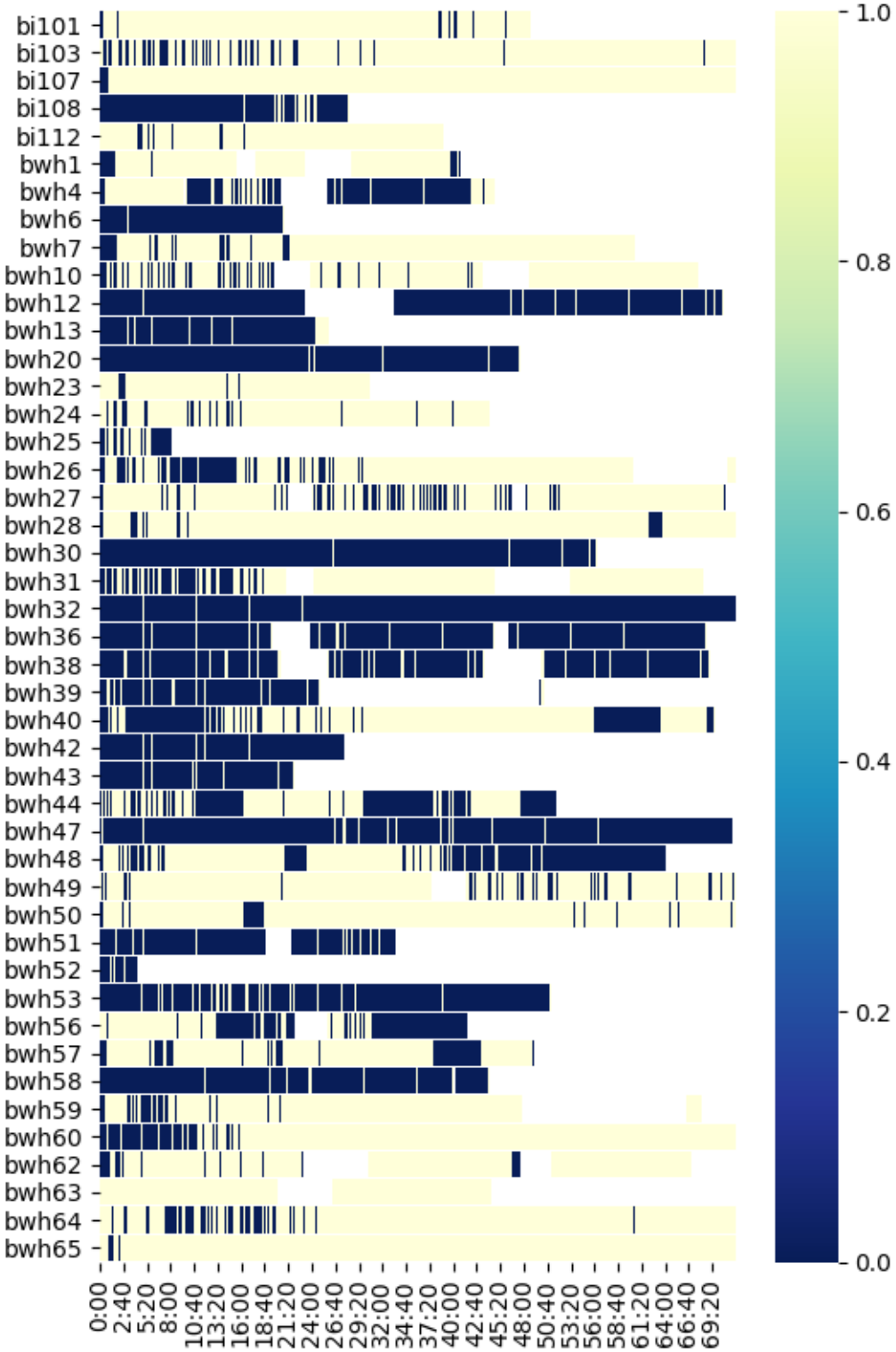


Figure 4.4: z_t^{GLOBAL} for patients with good outcome. Each row is labelled with the patient id and depicts the burst (yellow) versus suppression (blue) label over time (x-axis in hours). White portions represent gaps or artifacts in the signal.

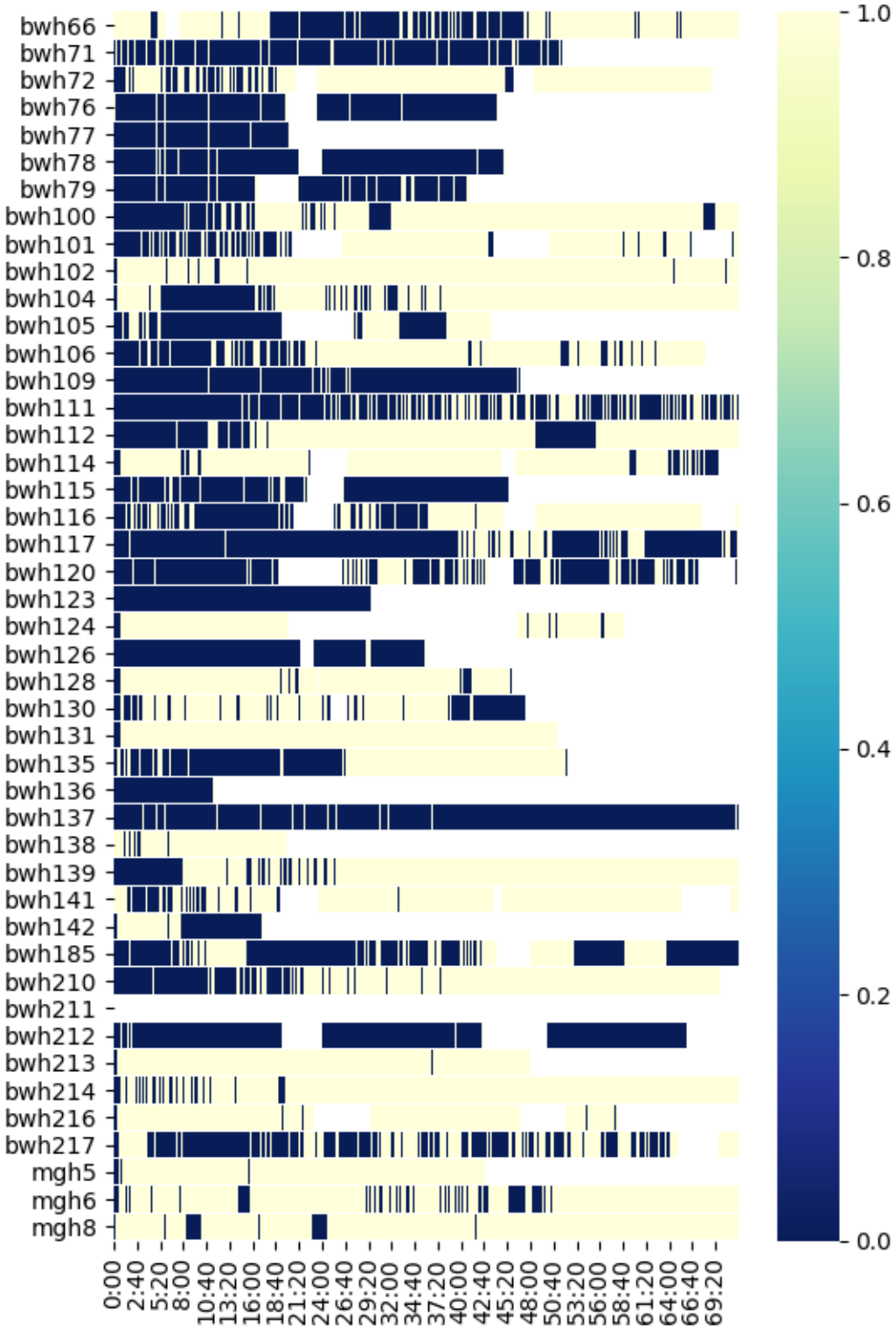
Figure 4.5


bi2
bi9
bi16
bi17
bi18
bi19
bi21
bi26
bi28
bi29
bi30
bi32
bi35
bi38
bi43
bi44
bi45
bi46
bi51
bi57
bi58
bi60
bi61
bi62
bi65
bi66
bi67
bi70
bi71
bi73
bi74
bi77
bi80
bi81
bi83
bi84
bi87
bi88
bi89
bi91
bi92
bi95
bi96
bi97
bi99
0:00
2:40
5:20
8:00
10:40
13:20
16:00
18:40
21:20
24:00
26:40
29:20
32:00
34:40
37:20
40:00
42:40
45:20
48:00
50:40
53:20
56:00
58:40
61:20
64:00
66:40
69:20
1.0
0.8
0.6
0.4
0.2
0.0

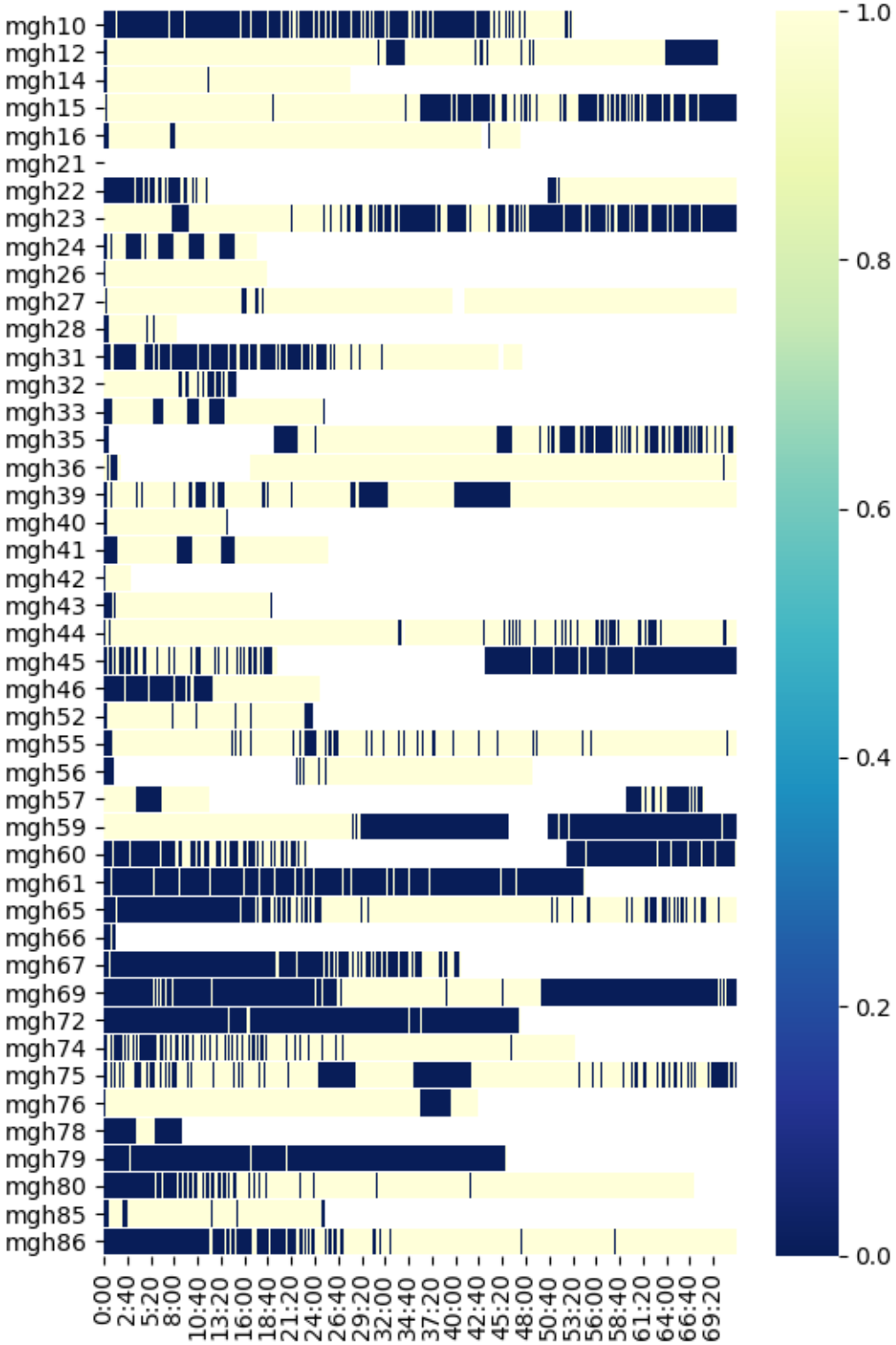
Continuation of Figure 4.5



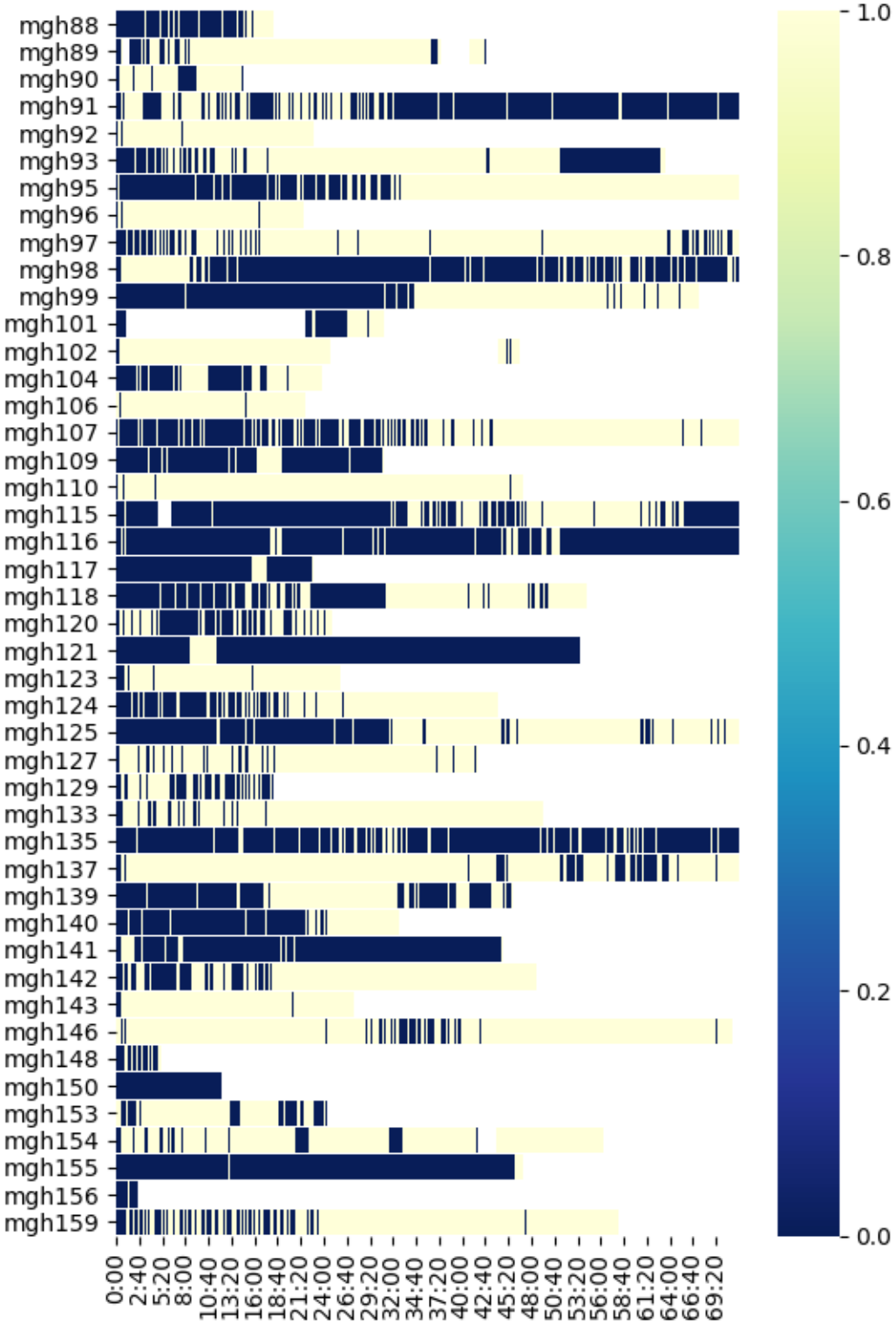
Continuation of Figure 4.5



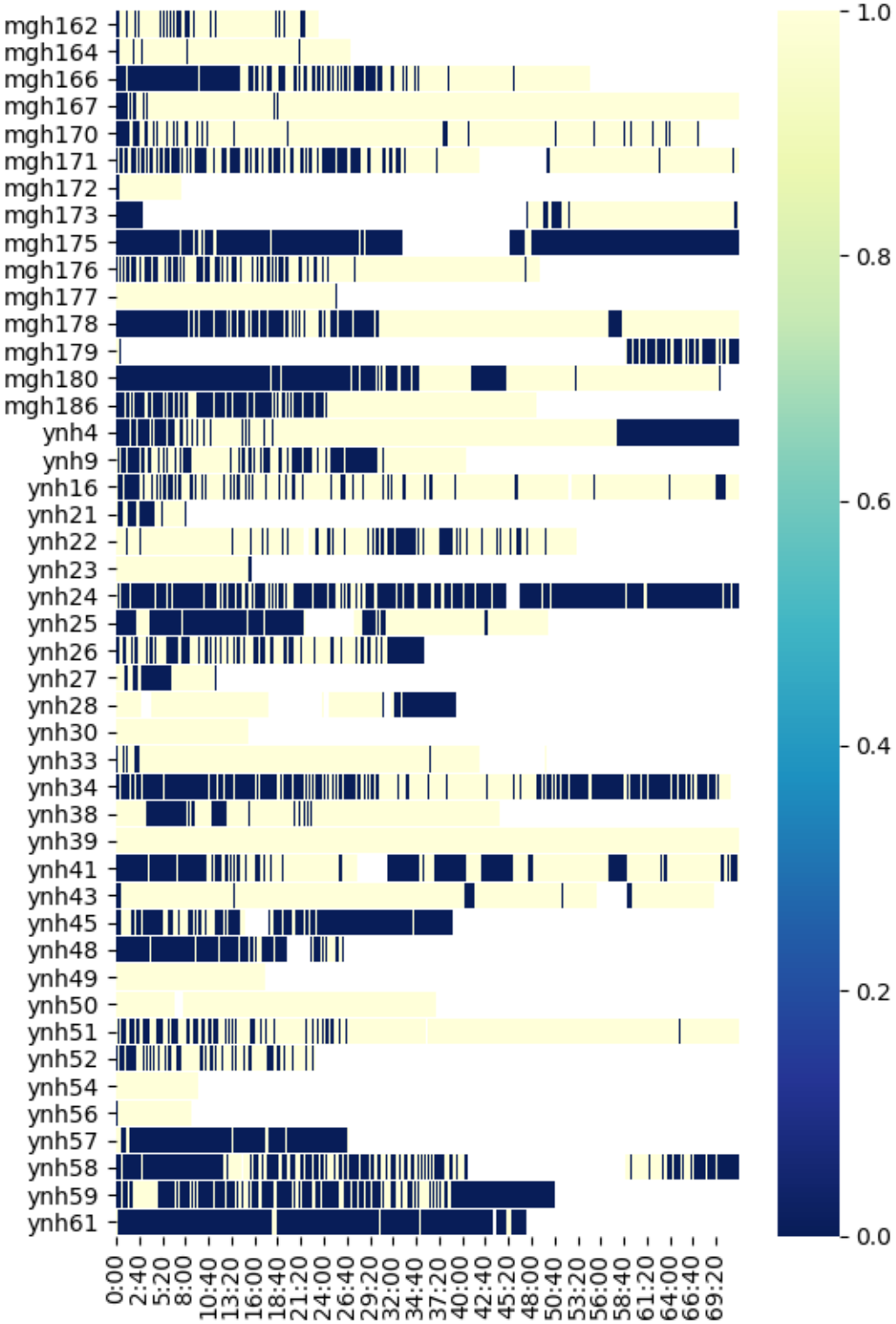
Continuation of Figure 4.5



Continuation of Figure 4.5



Continuation of Figure 4.5



Continuation of Figure 4.5

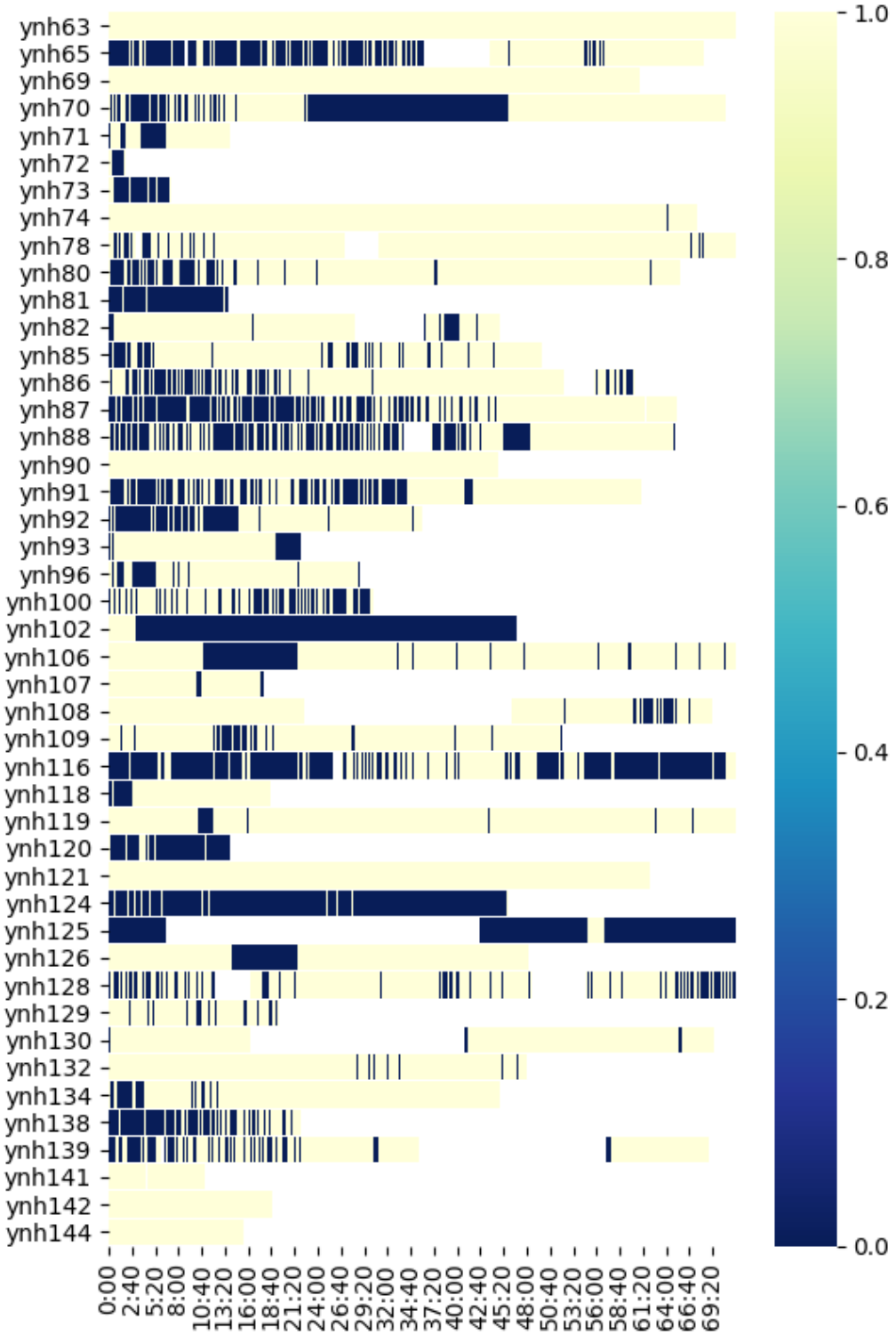
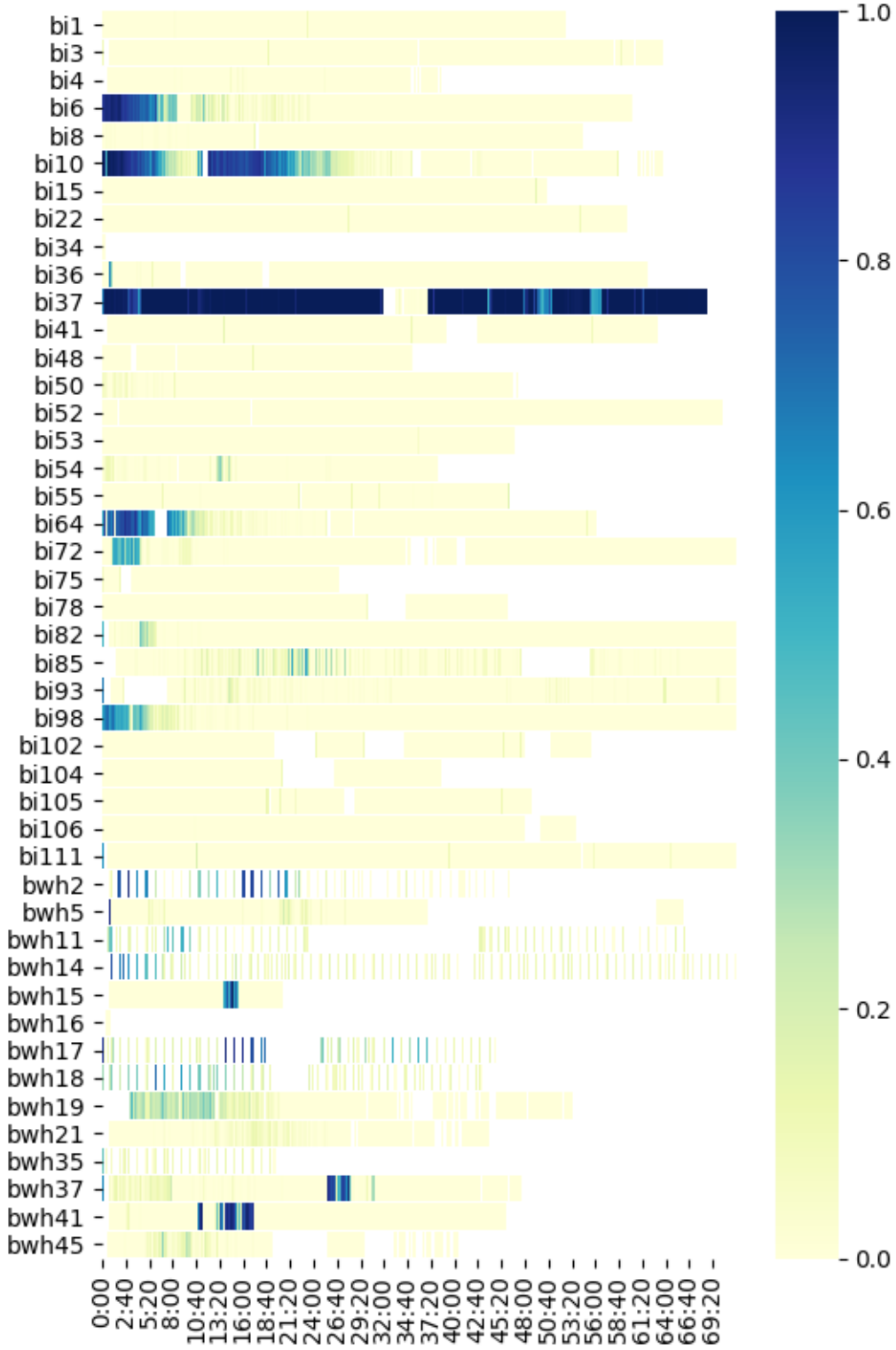
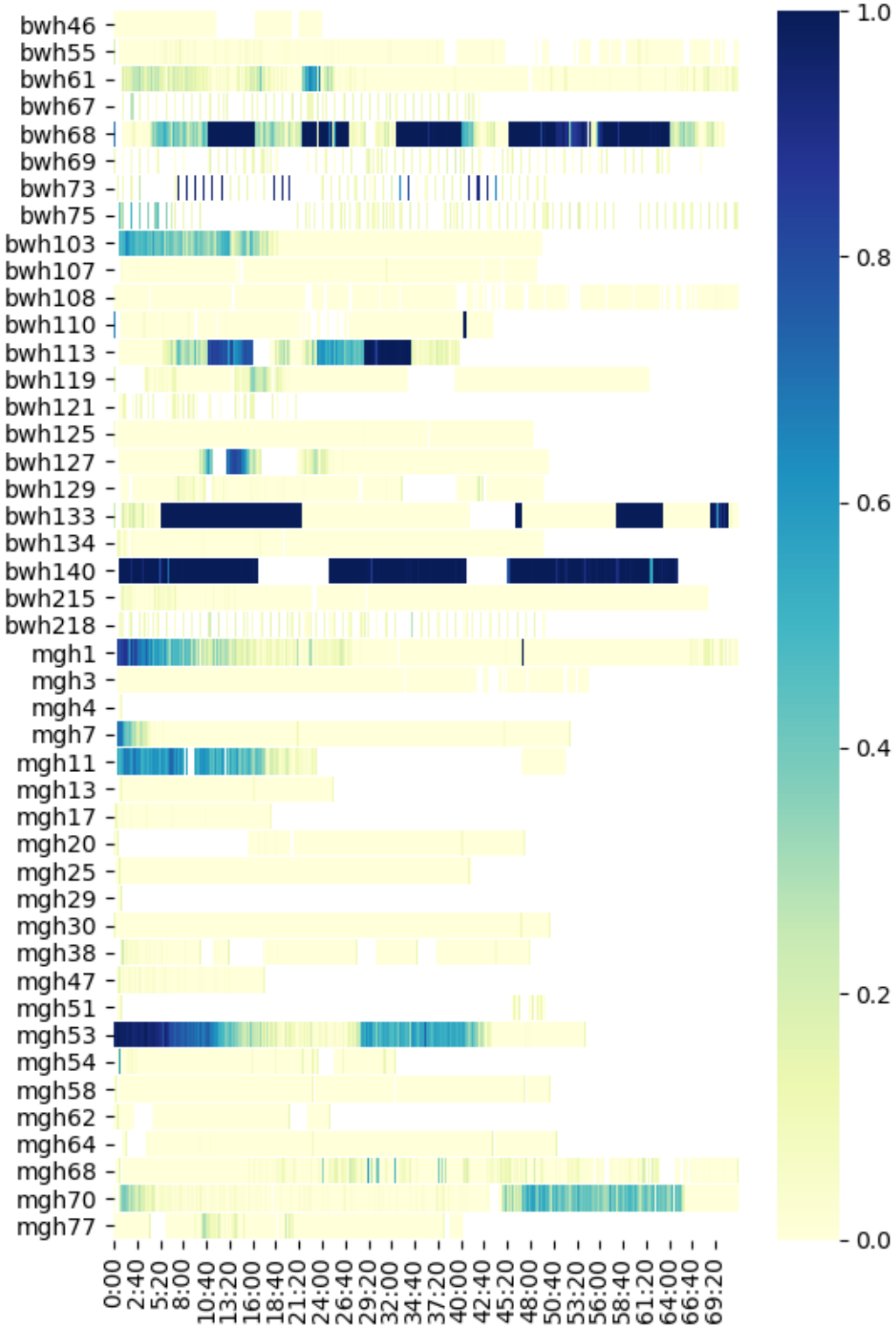


Figure 4.5: z_t^{GLOBAL} for patients with bad outcome. Each row is labelled with the patient id and depicts the burst (yellow) versus suppression (blue) label over time (x-axis in hours). White portions represent gaps or artifacts in the signal.

Figure 4.6


bi1
bi3
bi4
bi6
bi8
bi10
bi15
bi22
bi34
bi36
bi37
bi41
bi48
bi50
bi52
bi53
bi54
bi55
bi64
bi72
bi75
bi78
bi82
bi85
bi93
bi98
bi102
bi104
bi105
bi106
bi111
bwh2
bwh5
bwh11
bwh14
bwh15
bwh16
bwh17
bwh18
bwh19
bwh21
bwh35
bwh37
bwh41
bwh45
0:00
2:40
5:20
8:00
10:40
13:20
16:00
18:40
21:20
24:00
26:40
29:20
32:00
34:40
37:20
40:00
42:40
45:20
48:00
50:40
53:20
56:00
58:40
61:20
64:00
66:40
69:20
1.0
0.8
0.6
0.4
0.2
0.0

Continuation of Figure 4.6



Continuation of Figure 4.6

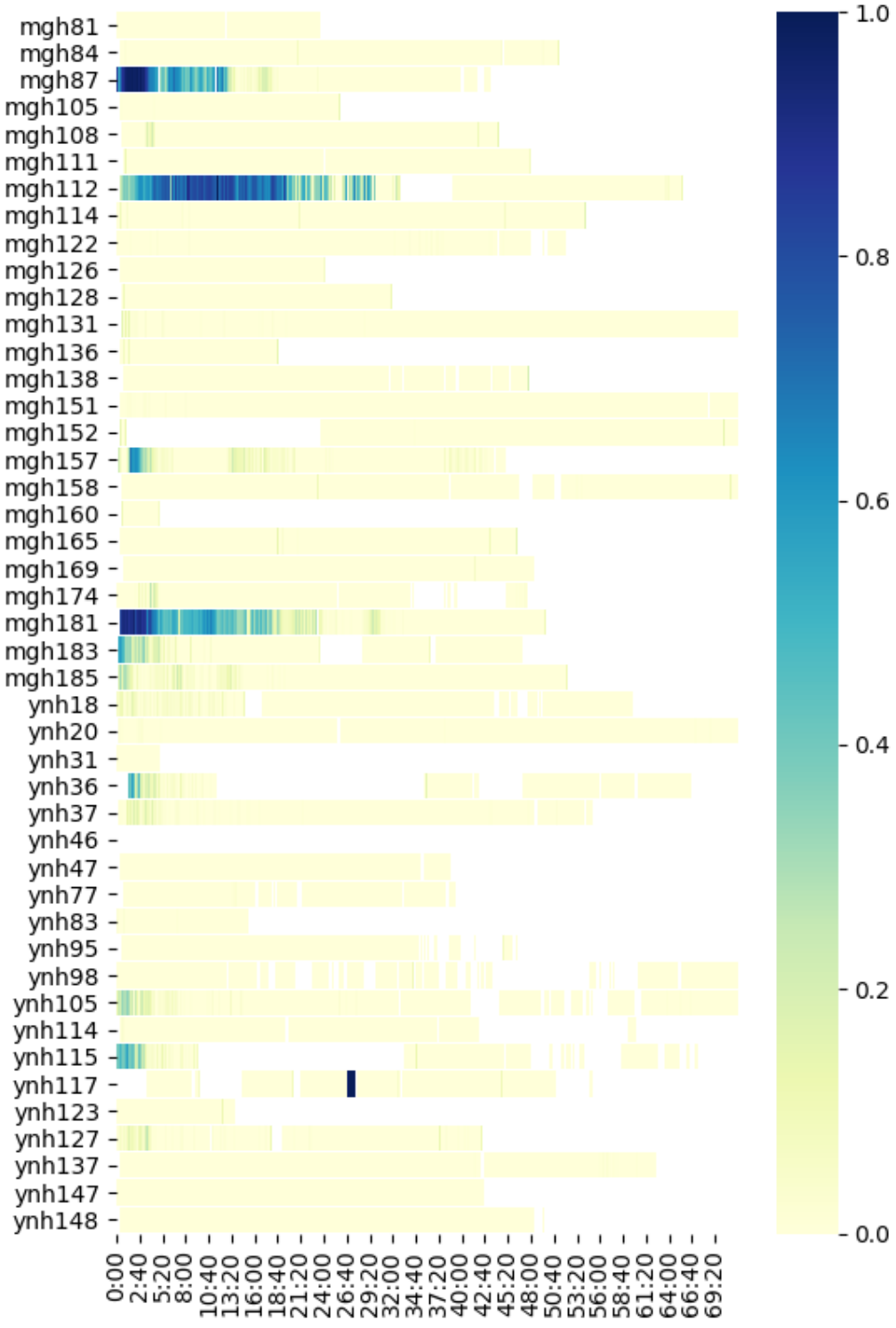
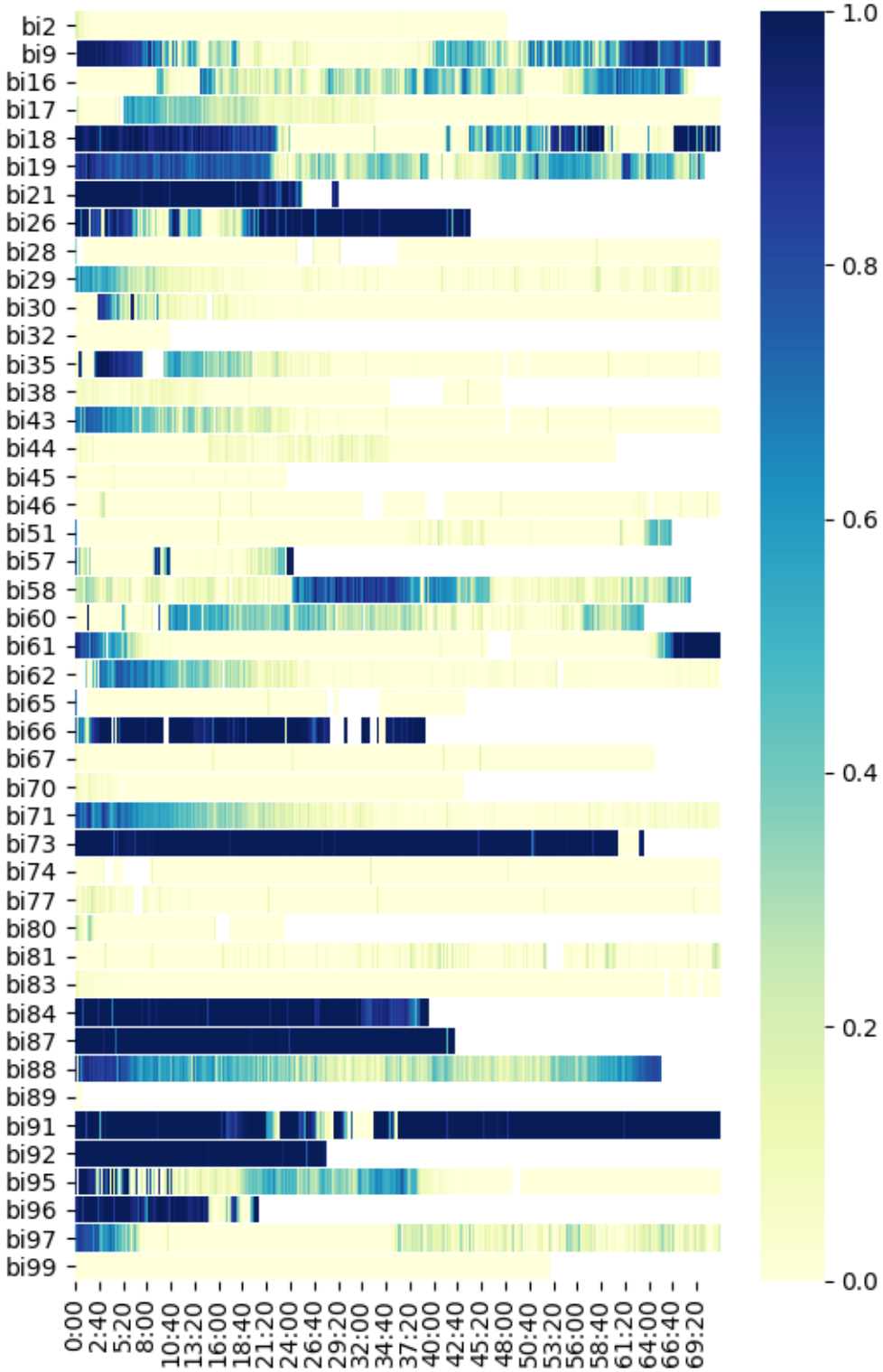
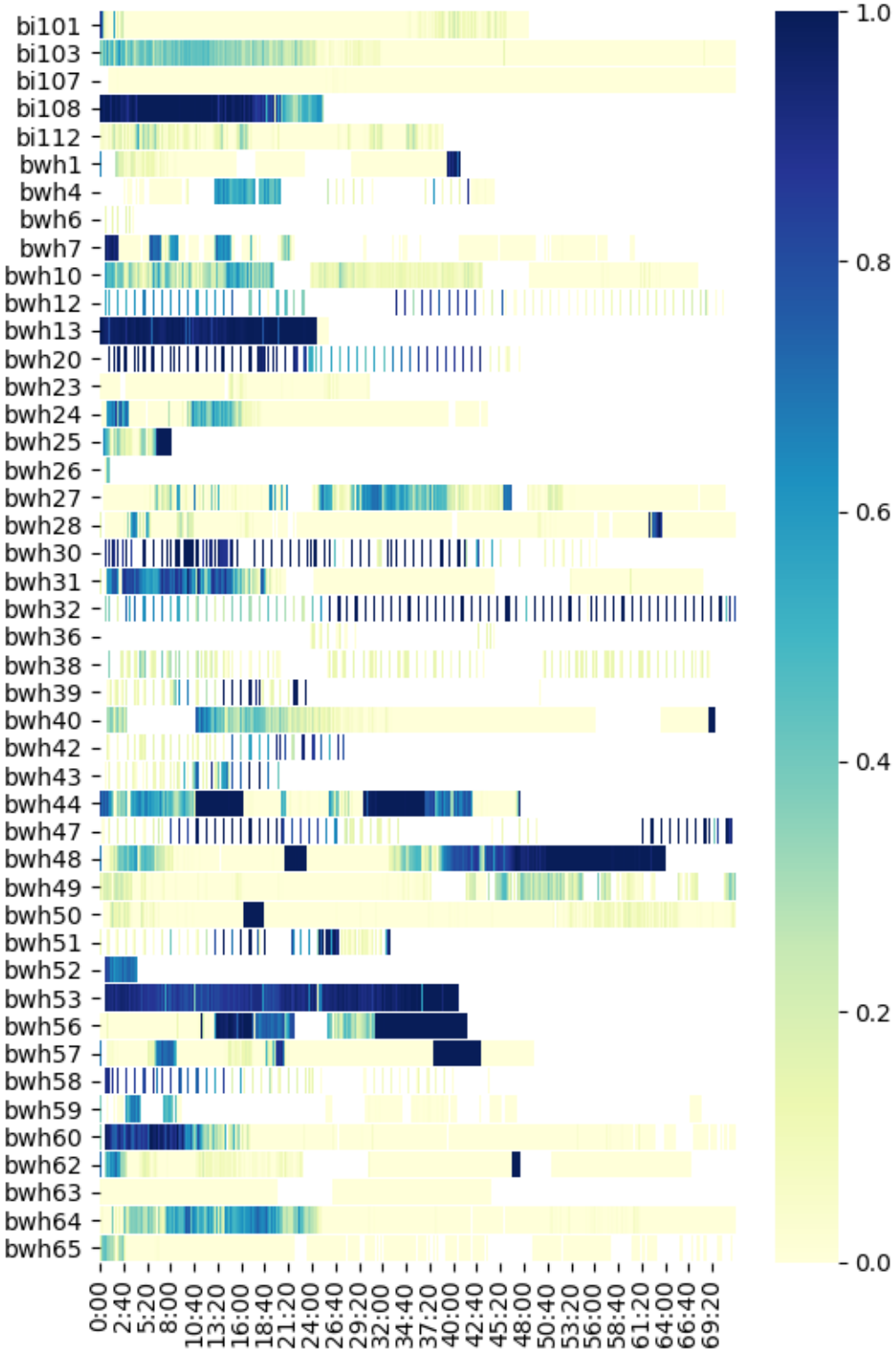


Figure 4.6: BSR for patients with good outcome. Each row is labelled with the patient id and depicts the BSR signal for that patient. White portions represent gaps or artifacts in the signal. X-axis is hours.

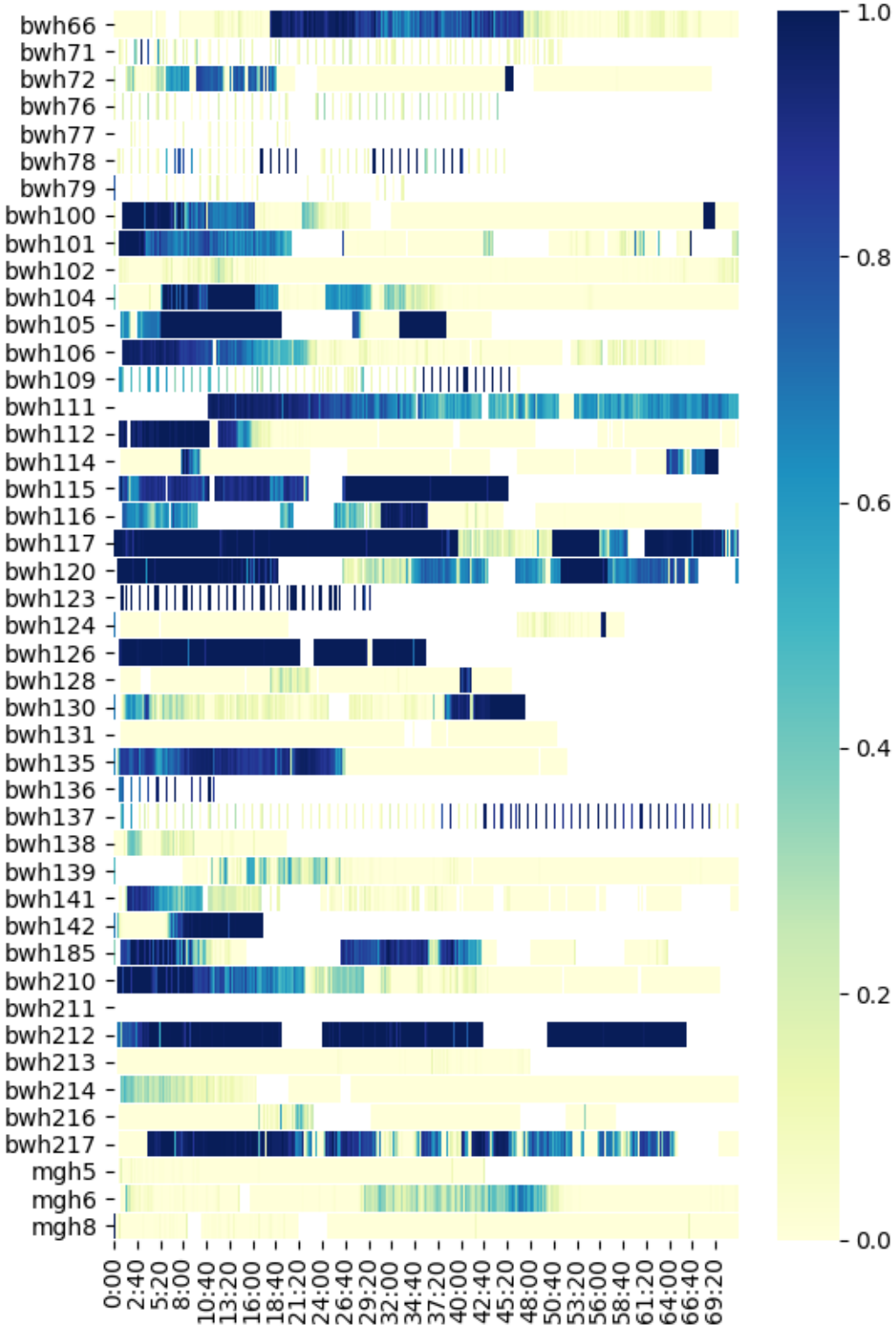
Figure 4.7


bi2
bi9
bi16
bi17
bi18
bi19
bi21
bi26
bi28
bi29
bi30
bi32
bi35
bi38
bi43
bi44
bi45
bi46
bi51
bi57
bi58
bi60
bi61
bi62
bi65
bi66
bi67
bi70
bi71
bi73
bi74
bi77
bi80
bi81
bi83
bi84
bi87
bi88
bi89
bi91
bi92
bi95
bi96
bi97
bi99
0:00
2:40
5:20
8:00
10:40
13:20
16:00
18:40
21:20
24:00
26:40
29:20
32:00
34:40
37:20
40:00
42:40
45:20
48:00
50:40
53:20
56:00
58:40
61:20
64:00
66:40
69:20
1.0
0.8
0.6
0.4
0.2
0.0

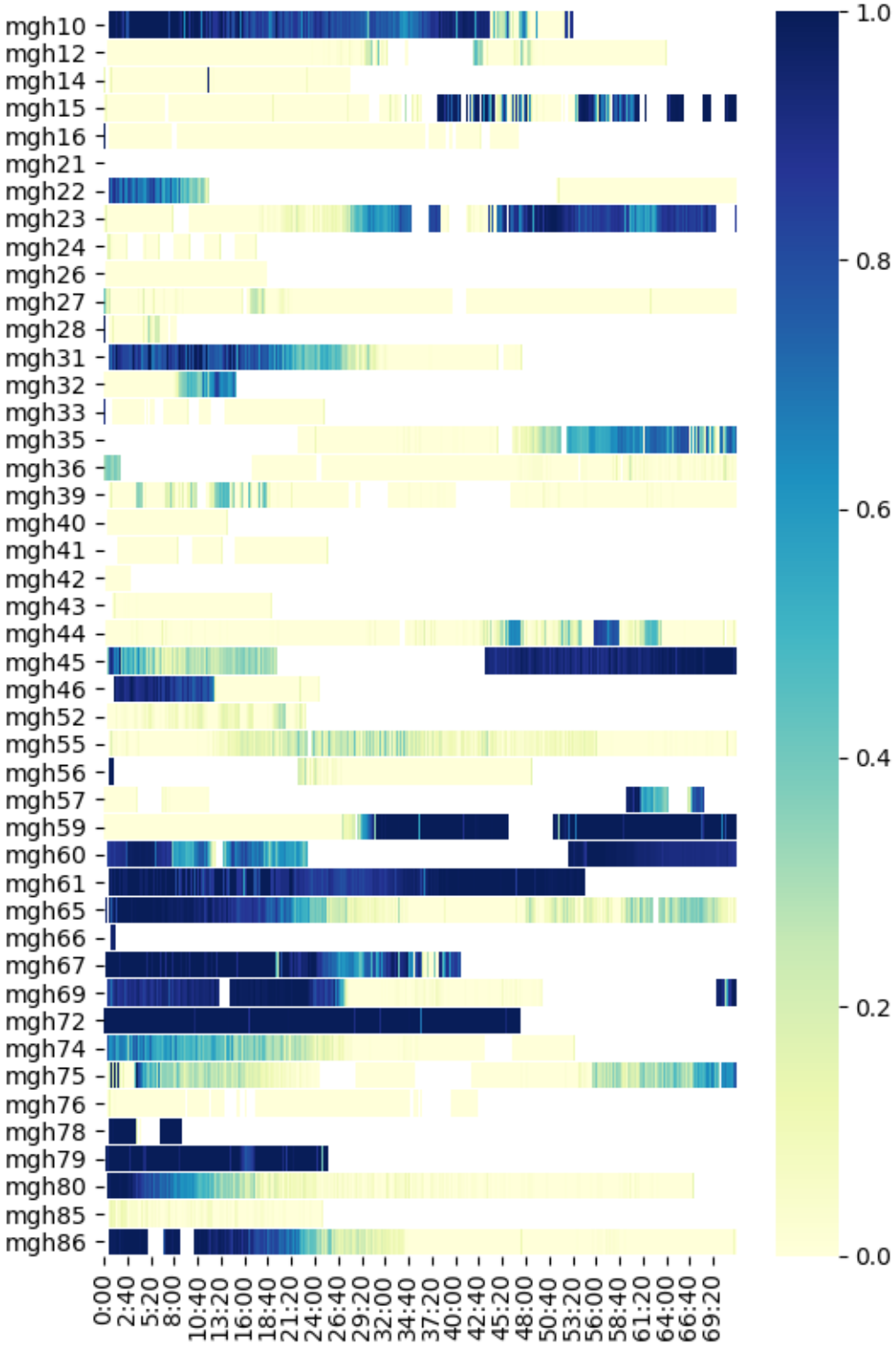
Continuation of Figure 4.7



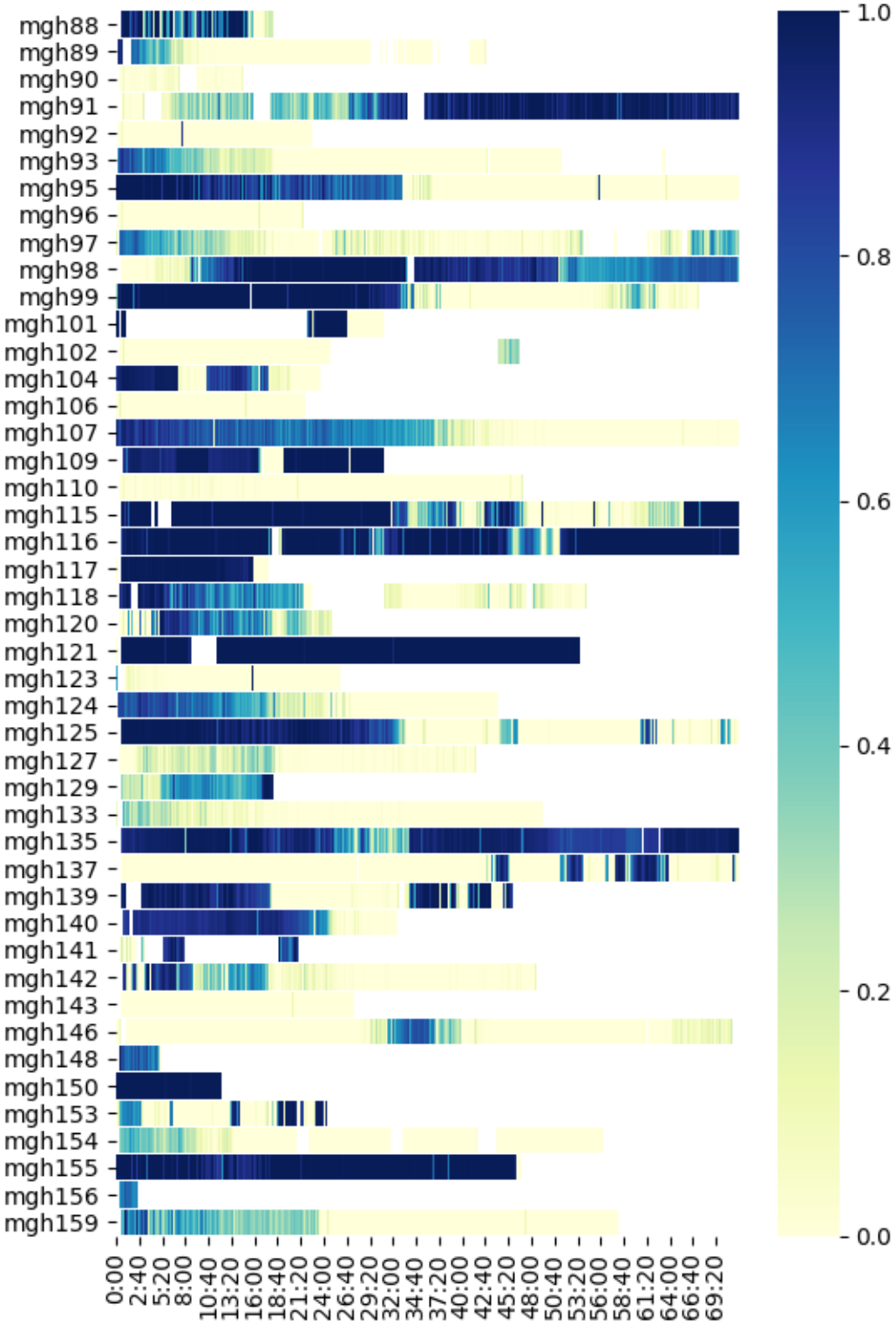
Continuation of Figure 4.7



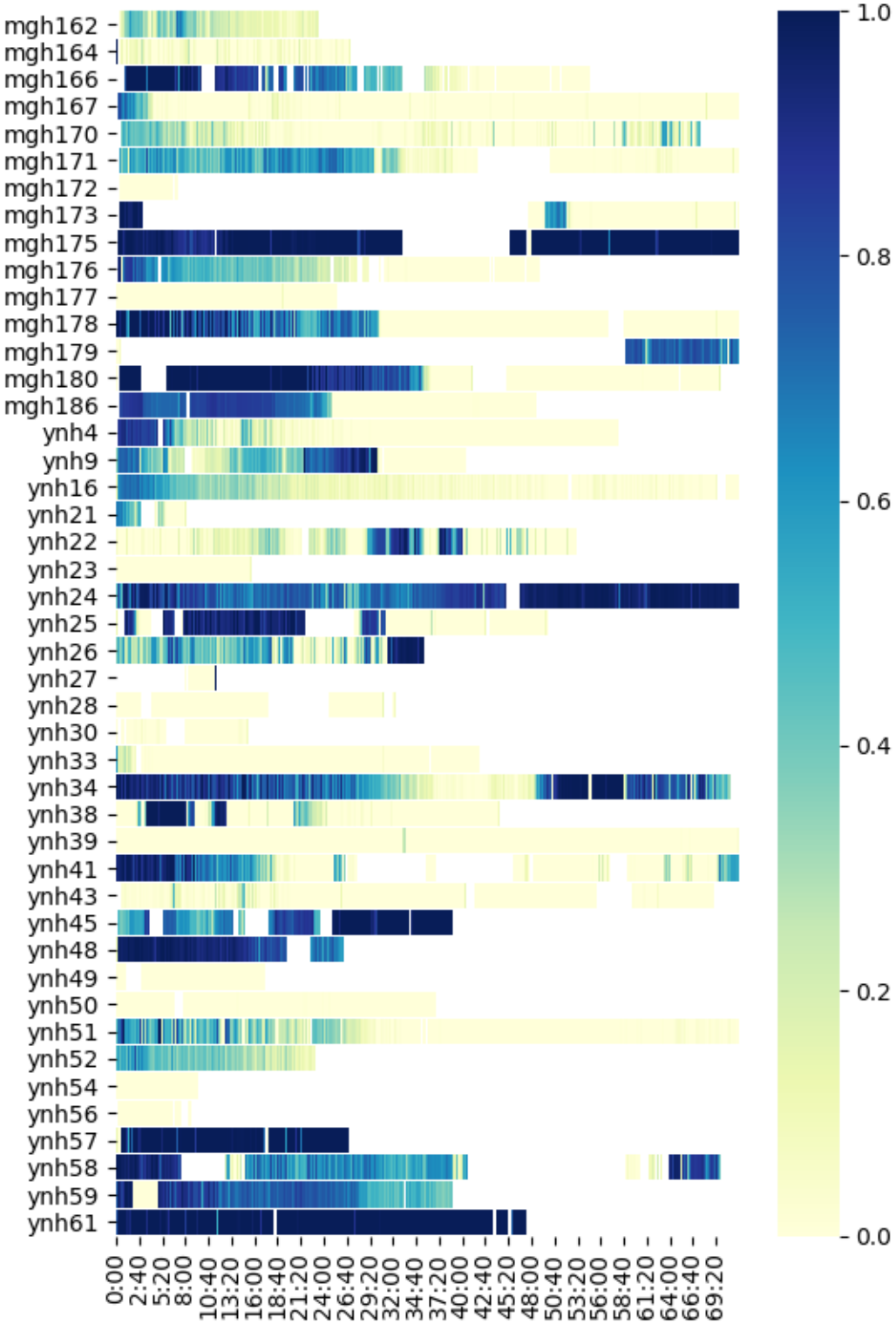
Continuation of Figure 4.7



Continuation of Figure 4.7



Continuation of Figure 4.7



Continuation of Figure 4.7

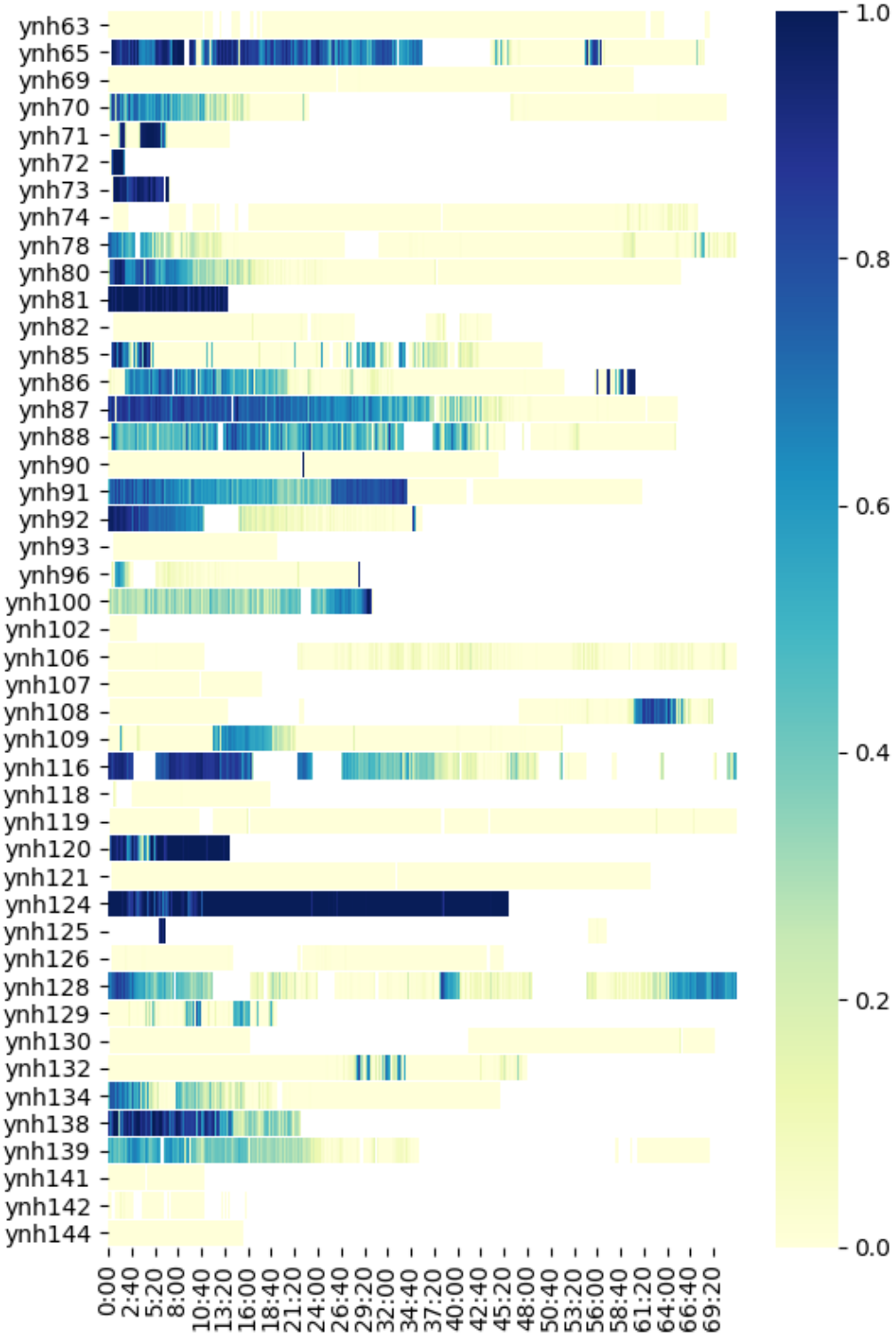


Figure 4.7: BSR for patients with bad outcome. Each row is labelled with the patient id and depicts the BSR signal for that patient. White portions represent gaps or artifacts in the signal. X-axis is hours.

4.2 Similarity

In this section, we describe our main results on running similarity for the bursts. For each patient, we had as output all the similarity scores for all pairs of bursts within each burst suppression episode. For our analysis, we concatenated the vectors of similarities across all the episodes that met a minimum duration time, so that each patient had only one flattened vector of similarities. We set the minimum duration of episodes we considered to be 30 minutes; anything shorter than this was discarded as being too short to be a true burst suppression episode. We chose this parameter based on the recommendation of the neurologists who collaborated with us. We did this with the similarity numbers produced by both cross-correlation and dynamic time warping (DTW).

In addition, we scaled all the similarity measures so that they would be between 0 and 1 using an inverse exponential function:

$$f(dtw) = e^{-dtw/c}$$

$$f(xcorr) = 1 - e^{-xcorr/c}$$

where c is a parameter that controls how quickly the function goes to 0 or 1. We set $c = 50$, since the raw, unscaled similarity numbers for both DTW and cross-correlation were largely between 0 and 200, with a long right tail of outliers up to a maximum similarity score of about 1000. Scaling in this manner thus allowed us to amplify the differences for raw scores in the range of common scores, and at the same time diminish the differences for the raw scores which were outliers.

In general, the results using similarity scores computed with only the first 500ms of bursts were better than those computed on the entire burst. In the remainder of this section, unless otherwise specified, all the results and figures pertain to the similarities numbers on the clipped bursts.

4.2.1 Histograms for patients with good versus bad outcome

For each patient, we computed a histogram of the similarity scores vector created as described above. We normalized these histograms so that they represented probability densities. We then divided the patients into those with good outcomes, and those with bad outcomes. For each outcome group, we created a histogram of the overall similarity distributions by mixing uniformly over the distributions of the patients in that outcome group. The resulting histograms are compared in Figures 4.8 and 4.9.

4.2.2 Example with identical versus non-identical bursts

We give an example of identical burst and non-identical bursts in our dataset in Figures 4.10 and 4.12. Following these, we also plot the full similarities histogram for the patients who have these bursts in Figures 4.11 and 4.13. We see that the patient with identical bursts has a histogram with a large peak close to one, while the patient with non-identical bursts has a histogram with a peak near 0.2. Following our hypothesis, the patient with identical bursts had poor outcome, and the patient with non-identical bursts had good outcome.

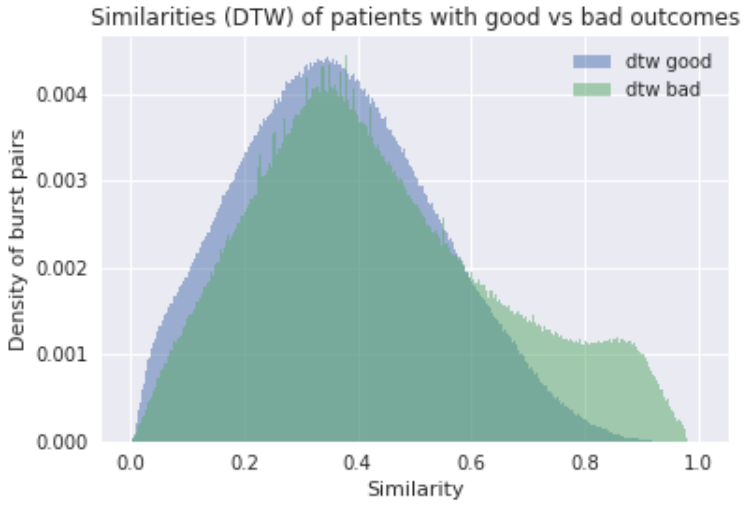


Figure 4.8: Histograms of the DTW similarities vectors of the patients with good and bad outcomes

Figure 4.9: Histograms of the cross-correlation similarities vectors of the patients with good and bad outcomes

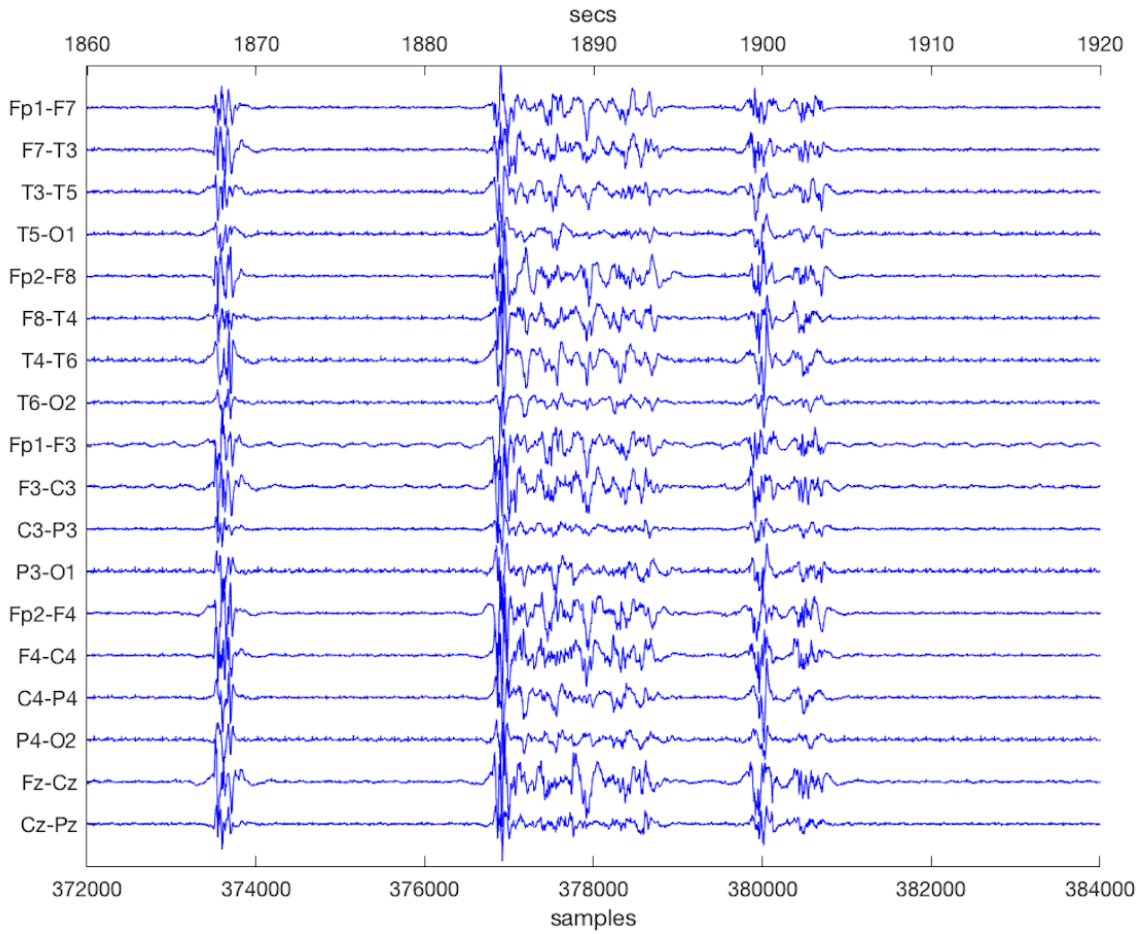


Figure 4.10: An example of bursts which are not identical in our dataset. This EEG comes from patient with id mgh112, who has good outcome.

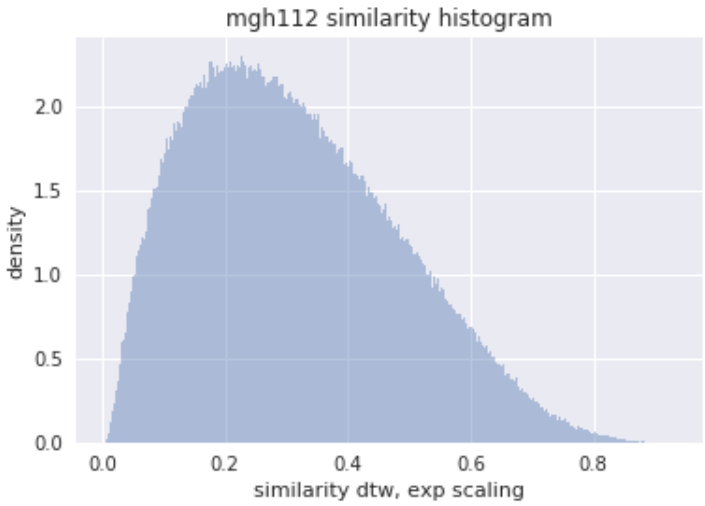


Figure 4.11: The similarities histogram for the patient with id mgh112, who has the non-identical bursts depicted in Figure 4.10. This patient has good outcome.

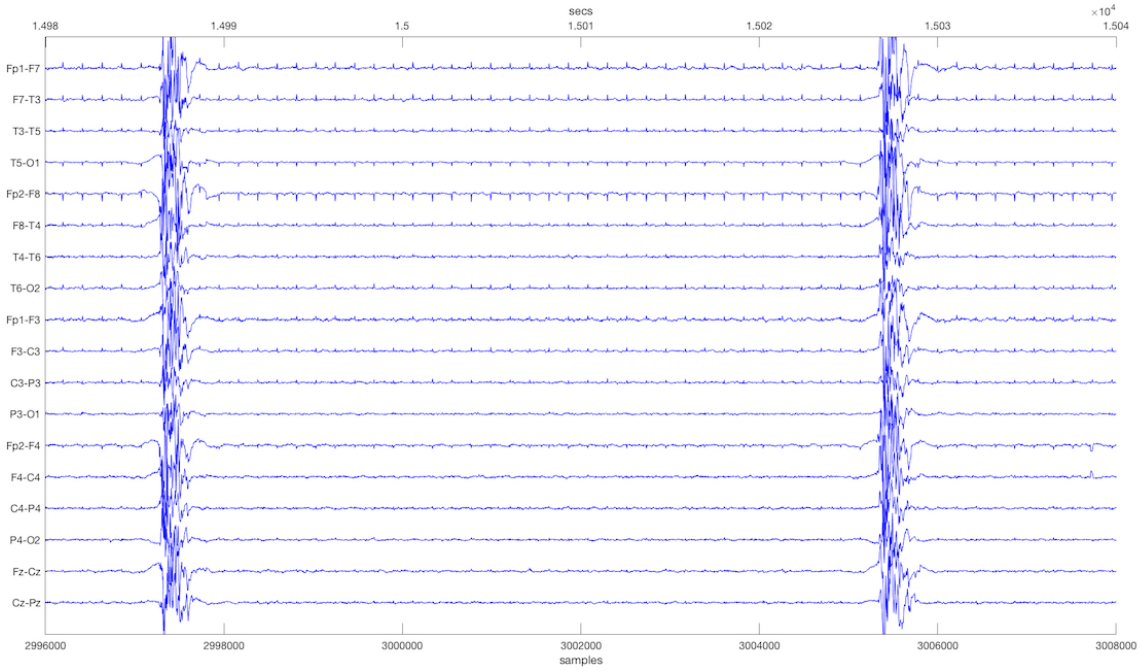


Figure 4.12: An example of bursts which are identical in our dataset. This EEG comes from patient with id ynh34, who has good outcome

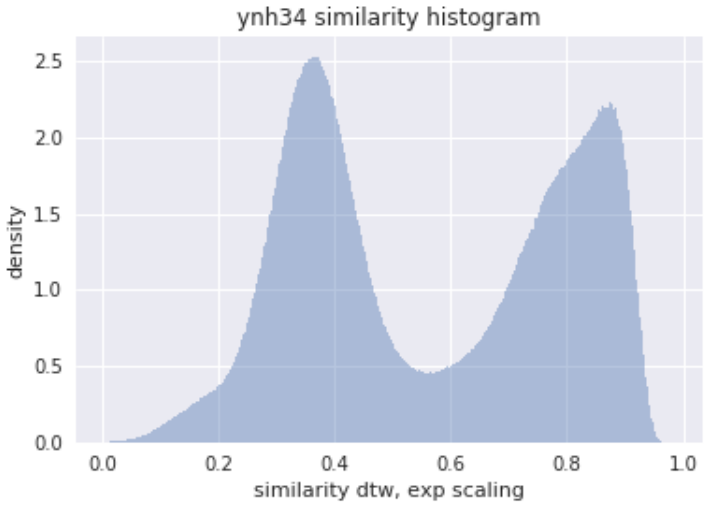


Figure 4.13: The similarities histogram for the patient with id ynh34, who has the identical bursts depicted in Figure 4.12. This patient has poor outcome.

4.2.3 ROC values

To more quantitatively analyze the ability of the similarity scores to predict outcome, we summarized each patient's similarities vector with one number. We tried a variety of summarizing statistics, including the mean and deciles of the similarities scores. In this manner, we ended up with one single similarity score and outcome for each patient. This allowed us to create ROC curves by thresholding the similarity score to predict good versus bad outcome. The ROC curve created from using the 20th percentile DTW value as the summary similarity score is depicted in Figure 4.14, and the ROC curve using the minimum cross-correlation value is depicted in Figure 4.15. Note that in these curves, the "positive" represents bad outcome, and "negative" represents good outcome. We chose these summary statistics by taking the best-performing summary statistic for each similarity function. From Figure 4.14, we see a long vertical line where the false positive rate is 0. From here, we see that using DTW we can correctly identify nearly 40% of the patients with bad outcome with 0

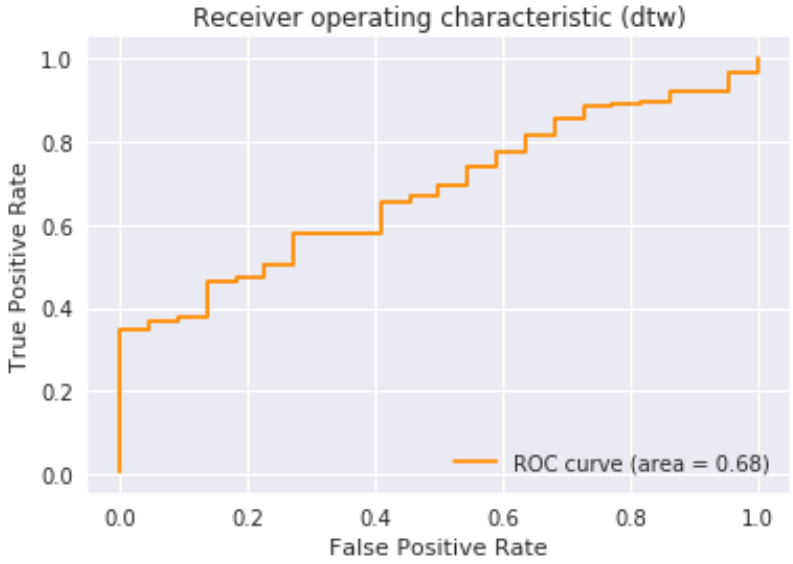


Figure 4.14: ROC curve, where the input is 20th percentile value of each patient's full DTW similarities vector, and the output is whether or not the patient had good outcome

incorrect predictions. We also plot the ROC curve for the similarity scores computed without clipping of the bursts in Figures 4.16 and 4.17. We again chose the summary statistics by taking the best-performing one for each similarity function. We can see that these ROC curves are worse than those computed on the clipped bursts. The cross-correlation similarities are especially effected, as the AUC goes to nearly 0.5, which is the AUC of a random guess.

4.2.4 Bar plots of similarity versus outcomes

Beyond looking at the ROC score, we also visualized how predictive similarity is by creating bins of similarity scores and plotting the percentage of each outcome in each bin. We binned these similarities scores by percentiles so that an equal number of patients ended up in each bin, and plotted the number of patients within each bin

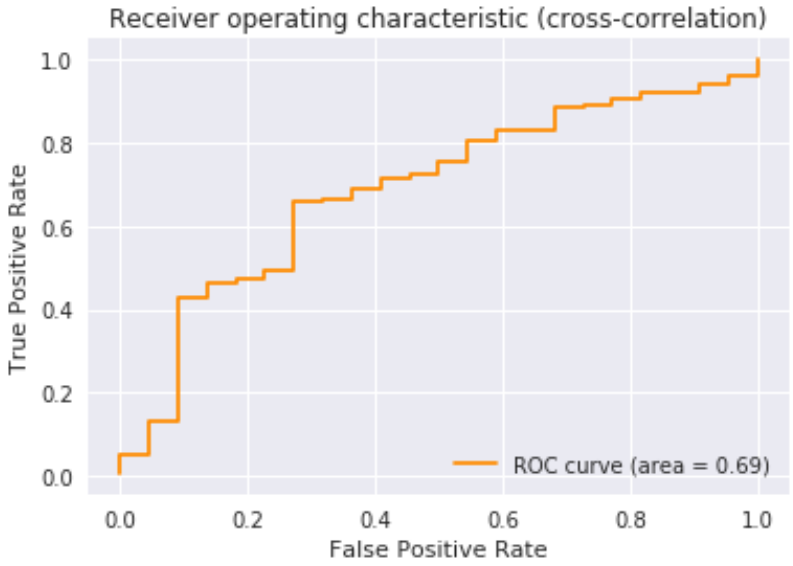


Figure 4.15: ROC curve, where the input is minimum value of each patient's full cross-correlation similarities vector, and the output is whether or not the patient had good outcome

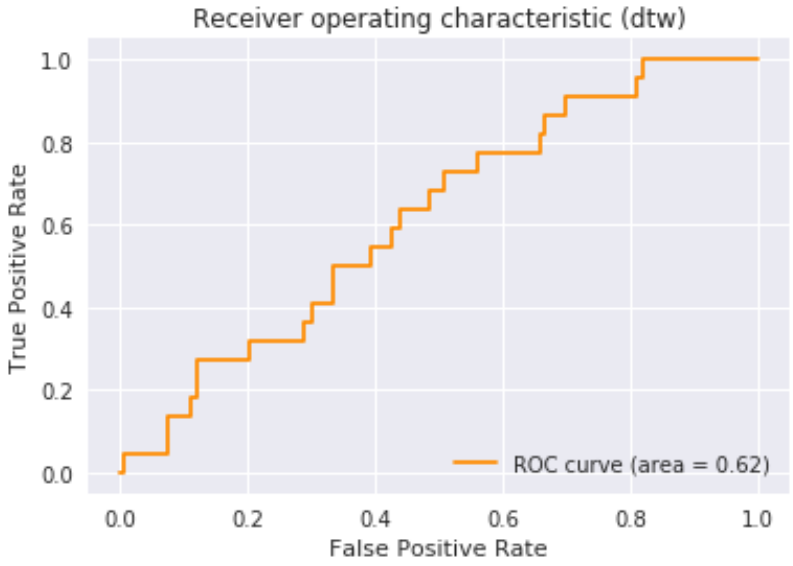


Figure 4.16: ROC curve, where the input is median value of each patient's full DTW similarities vector calculated on the full burst rather than the clipped 500ms burst, and the output is whether or not the patient had good outcome

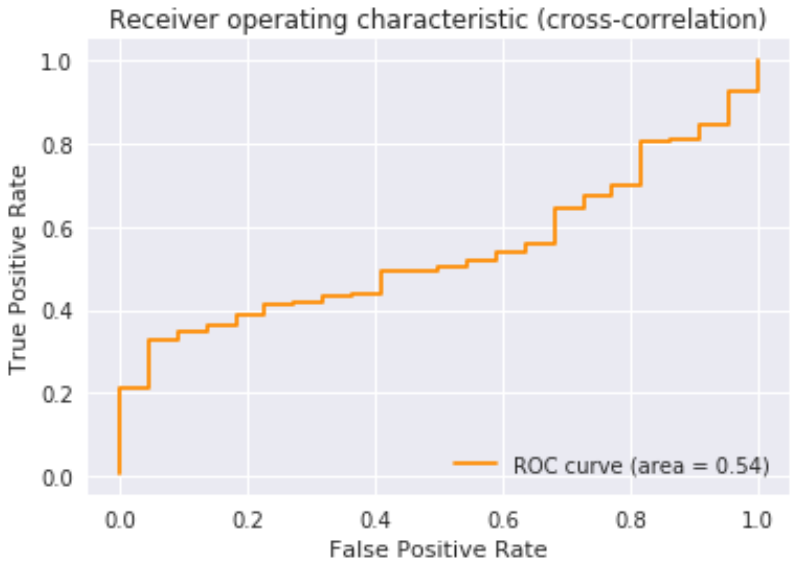


Figure 4.17: ROC curve, where the input is minimum value of each patient's full cross-correlation similarities vector calculated on the full burst rather than the clipped 500ms burst, and the output is whether or not the patient had good outcome

with each outcome. By binning by percentile rather than with equally spaced values, we make this plot robust to the type of scaling used, as long our scaling function is monotonic. We used the same summary statistics for these plots as for the ROC curves. Figure 4.18 shows this plot where the similarity score for each patient is their 20th percentile DTW value, and Figure 4.19 shows this plot where the similarity score for each patient is their minimum cross-correlation value. We can see in these plots that as the similarity score increases, the percentage of patients with good outcomes decreases. The decrease is quite pronounced in the cross-correlation plot; however, having a high DTW score seems to be more strongly indicative of poor outcome, since none of the patients in the highest DTW bin had good outcome.

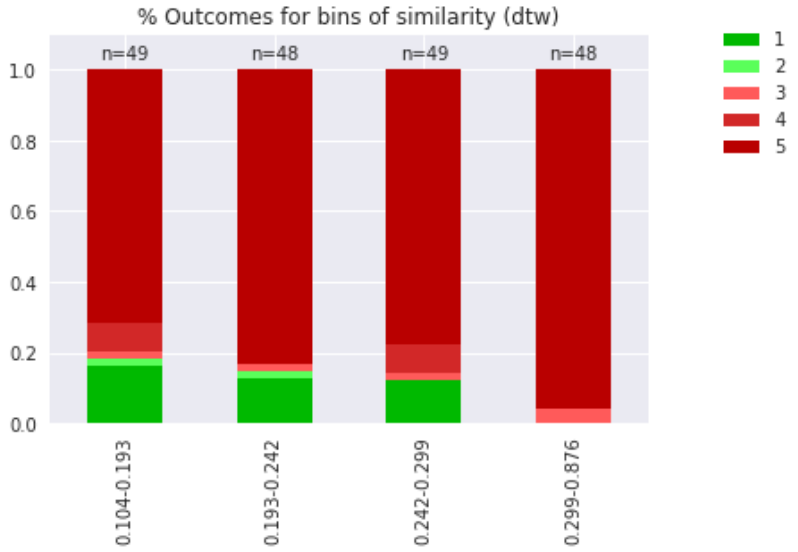


Figure 4.18: Bar plots of the percentage of patients with each outcome in each similarity bin, where the summary similarity score of each patient is the 20th percentile value of their full DTW similarities vector, using the clipped bursts. Note that outcomes 1 and 2 are considered good, and 3-5 are considered bad.

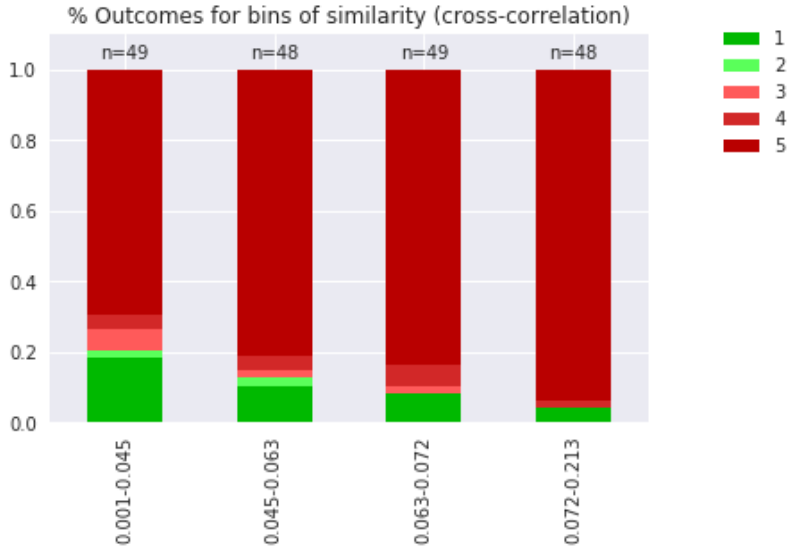


Figure 4.19: Bar plots of the percentage of patients with each outcome in each similarity bin, where the summary similarity score of each patient is the minimum value of their full cross-correlation similarities vector, using the clipped bursts. Note that outcomes 1 and 2 are considered good, and 3-5 are considered bad.

4.2.5 Similarity over time

In order to investigate whether the timing of the burst similarity had any effect on the outcome, we plotted the similarity over time for each patient. For each patient, we created a vector x representing the similarity over time in the following manner:

1. For each burst b_i, we calculated a single similarity score s_i by averaging across the similarities with its neighboring bursts:

$$s_i = \frac{1}{2W} \sum_{j=i-W, j \neq i}^{i+W} \text{similarity}(b_i, b_j).$$

   We set $W = 10$

2. We let x be a vector of length equal to the number of samples in the patient's EEG recording. For each burst b_i whose starting index is I_i, we set the I_i-th index of x equal to s_i:

$$x[I_i] = s_i$$

3. For each burst suppression episode which has starting index n_s and end index n_e, we filled in the values of x between indices n_s and n_e by linearly interpolating between the indices of x which were set by the previous step.

4. For indices which do not belong to a burst suppression episode, or which have no EEG recording data, we leave the x values at those indices blank.

Figures 4.20 and 4.21 depict a heatmap of the resulting x vector for each patient with burst supression, separated into patients with good outcome, and those with bad outcome.

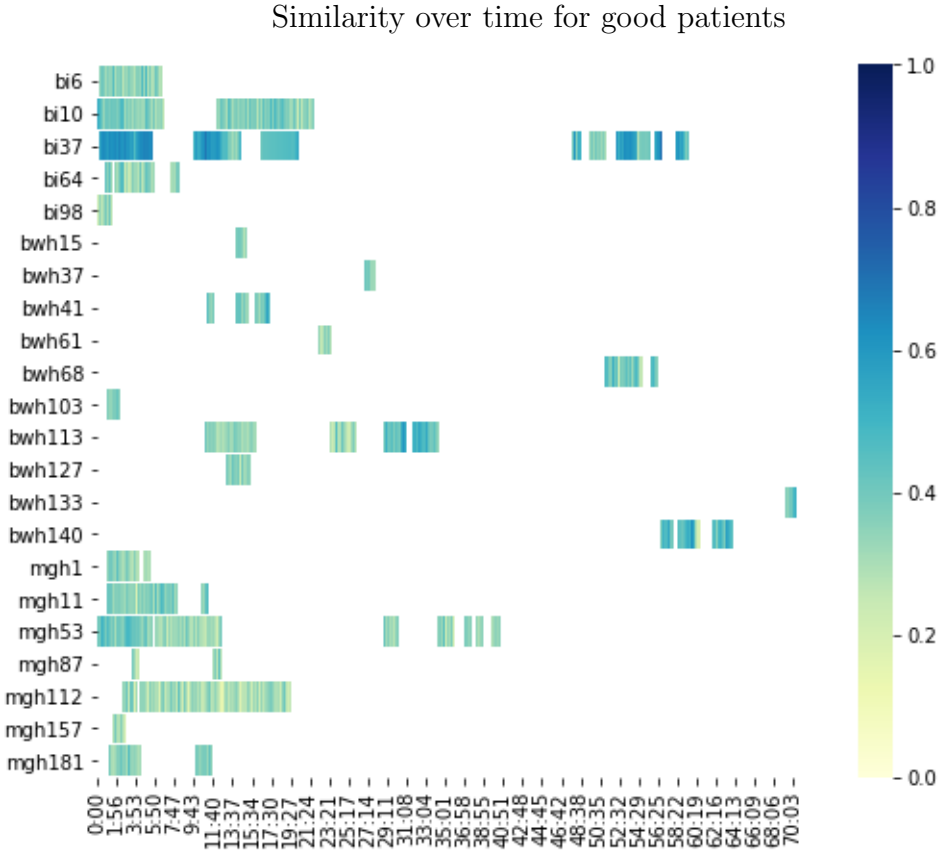
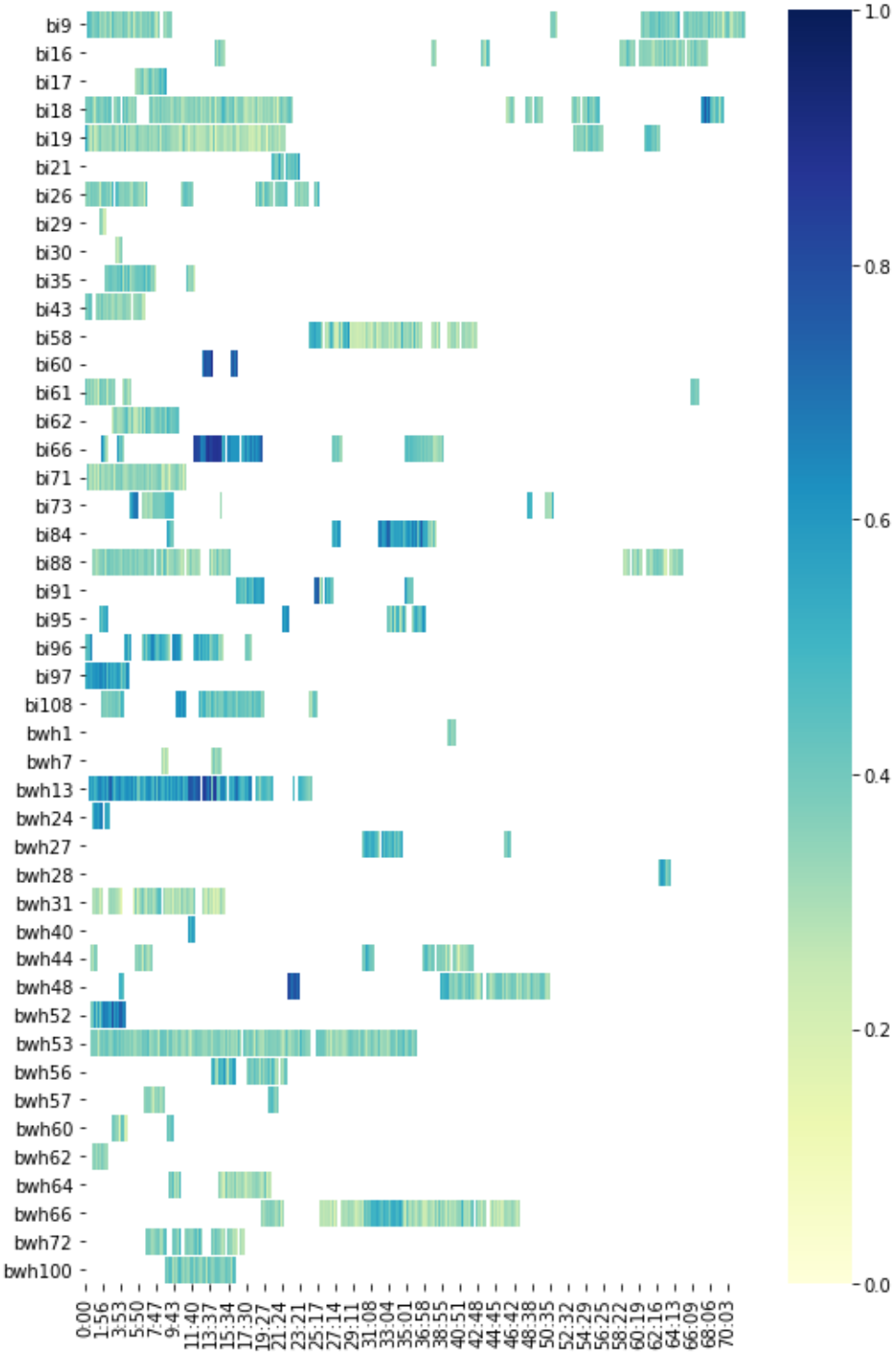
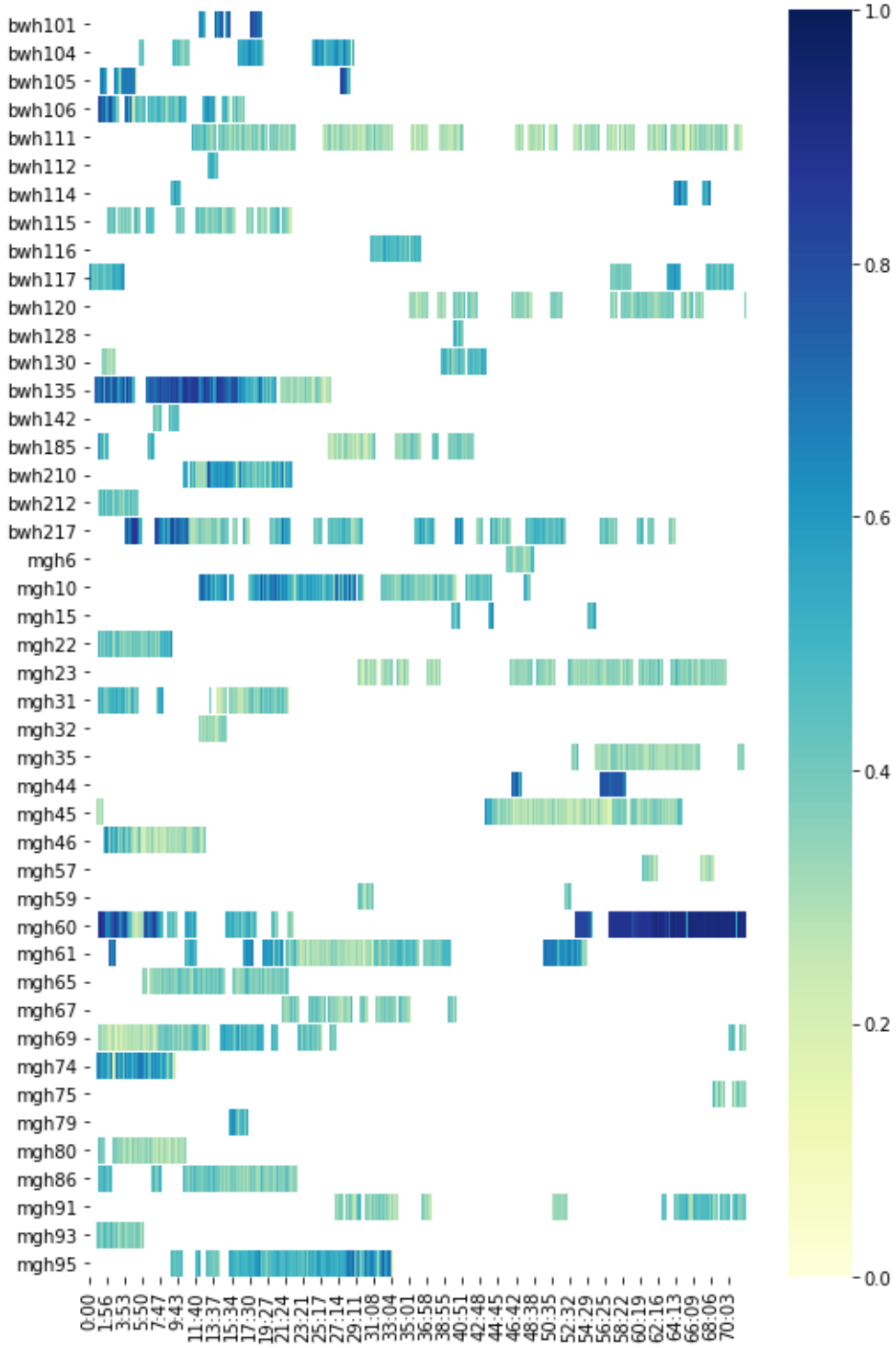


Figure 4.20: Heatmap plot of the similarity over time for each patient with good outcome who has burst suppression. White coloring indicates that the time sample was not labeled as part of a burst suppression episode. X-axis is hours.

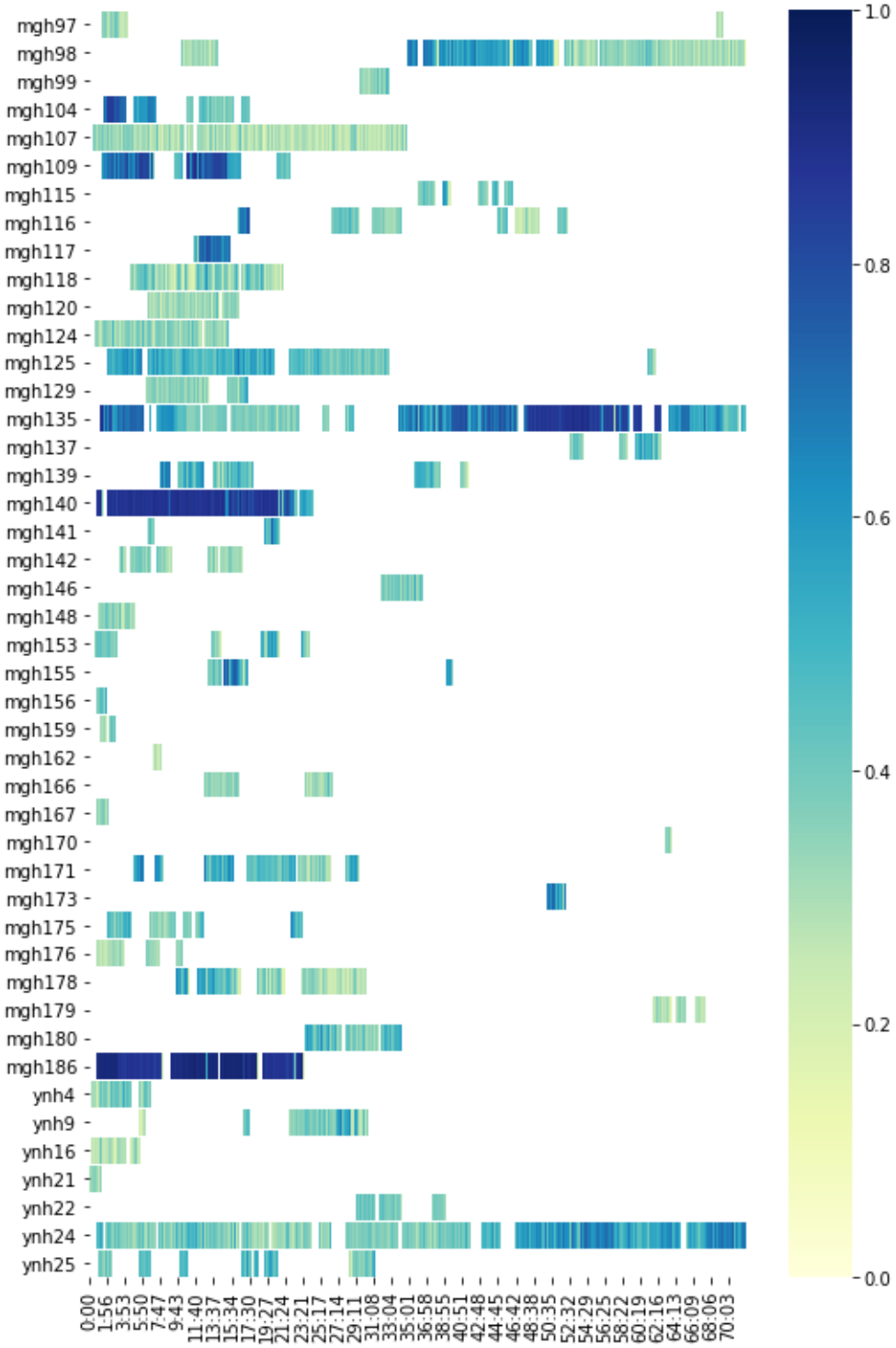
Figure 4.21


bi9
bi16
bi17
bi18
bi19
bi21
bi26
bi29
bi30
bi35
bi43
bi58
bi60
bi61
bi62
bi66
bi71
bi73
bi84
bi88
bi91
bi95
bi96
bi97
bi108
bwh1
bwh7
bwh13
bwh24
bwh27
bwh28
bwh31
bwh40
bwh44
bwh48
bwh52
bwh53
bwh56
bwh57
bwh60
bwh62
bwh64
bwh66
bwh72
bwh100
0:00
1:56
3:53
5:50
7:47
9:43
11:40
13:37
15:34
17:30
19:27
21:24
23:21
25:17
27:14
29:11
31:08
33:04
35:01
36:58
38:55
40:51
42:48
44:45
46:42
48:38
50:35
52:32
54:29
56:25
58:22
60:19
62:16
64:13
66:09
68:06
70:03
1.0
0.8
0.6
0.4
0.2
0.0

Continuation of Figure 4.21



Continuation of Figure 4.21



Continuation of Figure 4.21

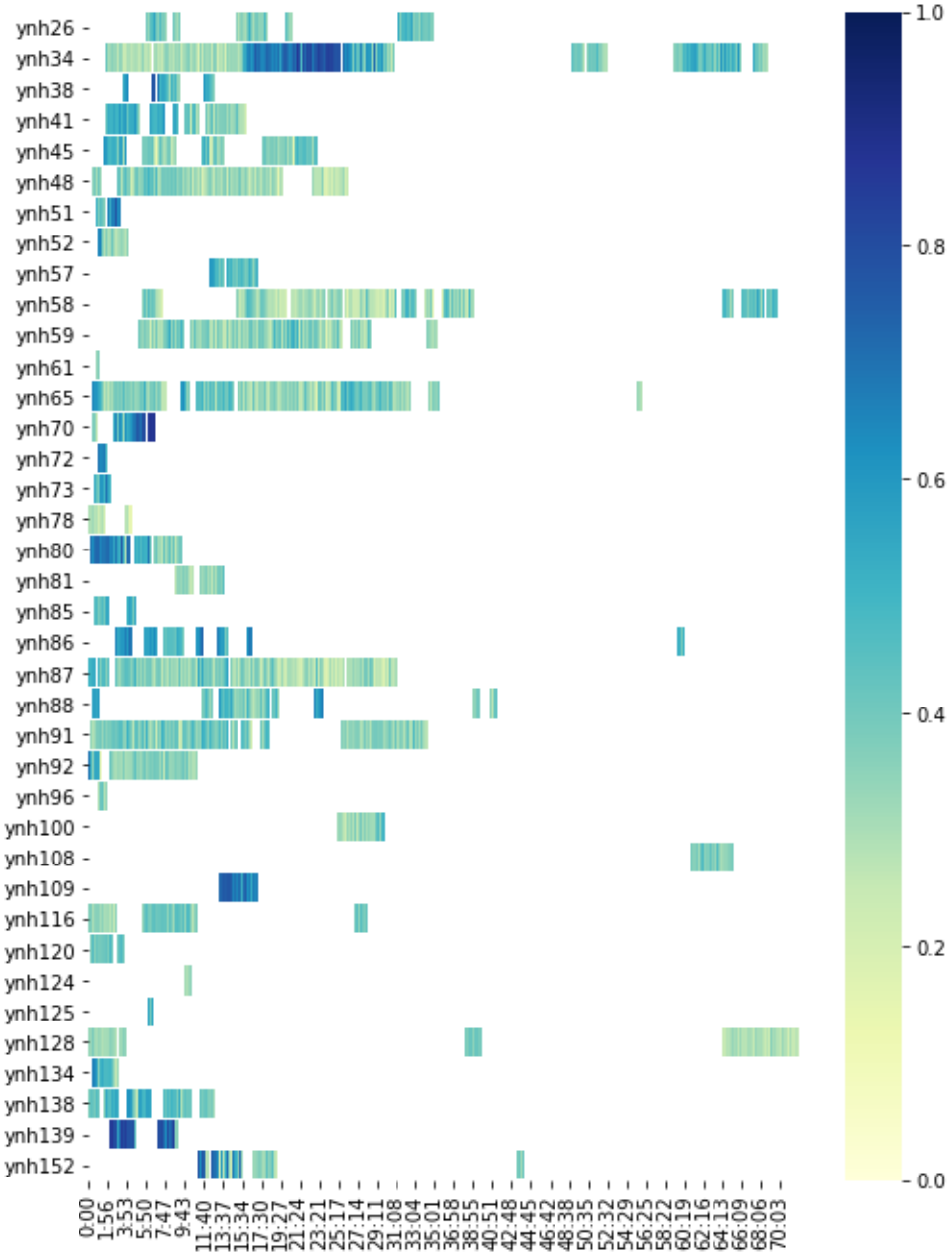


Figure 4.21: Heatmap plots of the similarity over time for each patient with bad outcome who has burst suppression. White coloring indicates that the time sample was not labeled as part of a burst suppression episode. X-axis is hours.

Appendix A

Summary of Parameters

Table A.1 gives a summary of all the parameters used in all steps of our algorithms.

Parameter	Description	Value
Preprocessing		
Downsampling target	common target frequency for downsampling	200 Hz
Low frequency cutoff	cutoff frequency for high-pass filter	0.5 Hz
High frequency cutoff	cutoff frequency for high-pass filter	50 Hz
Bandstop frequency	filtered frequency for notch filter	60 Hz
Artifact Detection		
High amplitude cutoff	max amplitude cutoff for too much signal	500 uV
Low variance cutoff	cutoff in variance for no signal	0.0001 uV^2
Burst Suppression Detection		
Labelling $z^{(i)}$ and z^{GLOBAL}		
Forgetting time	inverse weight of past on local mean/variance	0.1
Burst threshold	min variance for burst label	1.75 Hz
Agree percent	% channels to agree for global suppression label	60%
Min suppression time	min duration of a suppression	60 secs
Calculating burst suppression ratio		
BSR low cutoff	min BSR value for burst suppression	0.5
BSR high cutoff	max BSR value for burst suppression	1
BSR window	length of window for BSR computation	1 min
Defining burst suppression episodes		
Min bs time	min duration of a burst suppression episode	5 secs
Bs episode smoothing	smoothing amount for time ranges of bs	1 sec
Similarity Quantification		
Min burst time	min duration of burst used for calculation	0.1 secs
Max burst time	max duration of a burst used for calculation	5 secs
Burst trim time	trimmed duration of burst used	500 ms and 5 secs
DTWmaxsamp	max warping width from straight-line	200

Table A.1: Summary of all parameters used

Appendix B

Dataset Details

We use a dataset of 532 patients across four different hospitals. Each patient has a number of hours of EEG recording, split into multiple segments of EDF files of various lengths. There are a total of 4762 EDF files, totalling 1.4 terabytes in size. Each EDF segment file contains a recording whose duration ranges from several minutes to tens of hours. Most patients have under 72 hours of recording. For patients who have more than 72 hours, we analyze only the data from their first 72 hours to mirror the amount of information physicians would have when they make their prognosis at the 72 hour mark. Each recording typically contains 19 data channels, corresponding to locations of EEG electrode placements on the head (Figure B.1).

We use a separate CSV (Comma-Separated Values) file containing the outcomes of all the patients. The file lists the hospital, patient id, and patient outcome score on the Cerebral Performance Categories (CPC) scale at the time of discharge, 3 months after discharge, and 6 months after discharge Safar and Grenvik [1981]. We take the

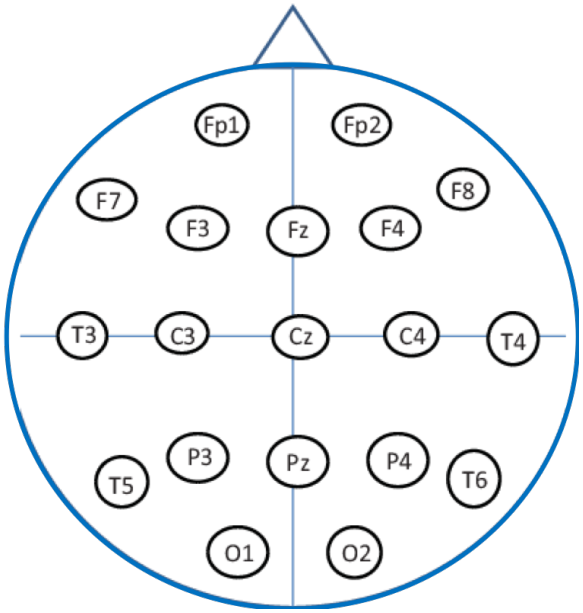


Figure B.1: Figure depicting the 19 electrode locations used in each EEG recording

CPC Outcome Score	Percentage of patients
1	26%
2	5%
3	2%
4	3%
5	64%

Table B.1: The breakdown of outcomes across patients

best score by 6 months as the final patient outcome score. We consider a CPC score of 1 or 2 to be a good outcome, and a score of 3 to 5 a bad outcome. Table B.1 summarizes the percentages of patients in each outcome category.

The patient files are named containing the hospital, the patient ID, the segment number, and the timestamp of the start of the recording. Within the EDF file, the header contains information about the starting timestamp, the number of samples in the record, the number of channels, channel labels, and the units of the recording.

B.1 Dataset Cleaning

There were a few data sanitation issues that needed to be resolved before we could use the dataset. After our dataset was acquired, the releasers of the dataset created a new version of the dataset, changing the file naming scheme. This new dataset version included a CSV of patient outcomes as well as a list of patients who needed to be excluded, either because their outcome information was missing, or because their clinical situation did not match the type of analysis we wanted. In order to use these lists, we needed to match up our old patient names with the new ones. In most cases, the patient id did not change. However, due to the dataset going through multiple iterations, the version we had contained a mix of various naming schemes, some of whose had patient id's that did not match the new version.

Furthermore, we had realized that a number of the timestamps in the names of our EDFs, as well as the timestamps in the headers, were incorrect. The new dataset version had EDFs with correct timestamps, so another task was to match up our old EDF names to the proper new EDF name containing the correct timestamp. This presented a challenge–since we did not know for sure the patient id mapping or the timestamp mapping, it was difficult to map any old names with new ones.

To resolve this, we ran a script on the new dataset to get a list of all the new EDF names and their headers. We assumed that the timestamps in the headers would match the timestamps in the headers in our dataset nearly exactly, with the exception that they might be off by a few seconds. In addition, we were given a table (which we will refer to as the master lookup table) containing the names of all the new EDFs and the names of the new patient id's and their corrresponding old id's

taken from another version of the dataset. In some cases, this old id seemed to match our old id, and by seeing that the number of EDFs matched, and that the timestamps were similar, we could produce mappings from old to new namings. We thus went about mapping in the following manner:

1. First, we attempted to match up EDF names purely based on the headers. For any EDFs which were matched in this manner, we extracted the patient id from the old EDF name and the new EDF name, and added these patient id's to our id mapping list.

2. In some cases, the master lookup table listed that two old id's mapped to the same new id. These were manually added into the old to new id mapping at this point.

3. There were a number of cases where an old EDF header matched with the header of multiple new EDFs. Since there seemed to be no systematic pattern here for matching, we manually reviewed these cases and chose the correct matching among the candidates.

4. Next, we attempted to match EDFs that had similar timestamps to EDFs listed in the master lookup table. Part of the reason for this was that the header dataset seemed to not match the master lookup table exactly, with each containing EDFs or patients that the other did not list, so it was necessary to use both to perform a full mapping.

    (a) For each old EDF that was still unampped at this point, we guessed the new patient id corresponding to the old patient id. We did this by first attempting to get the mapping from our existing old to new id mapping;

if it was not in there, we assumed that the old to new id mapping listed in the master lookup table was correct.

(b) We searched through the new EDFs listed in the master lookup corresponding to this guessed new patient id. If any of them has timestamps that matched our EDF (again, ignoring the seconds), then we assumed that we found a match.

(c) If exactly one EDF match was found, we assume this confirms our guessed old to new patient id mapping, and we add this to our mapping list as well.

5. At this point, most of the id's that were not excluded were matched up. We manually reviewed the rest and matched up the rest. We saved this into `old_to_new_id.csv`

6. We created a file `old_to_new_edf_mapping.xlsx` summarizing the result of this process. For each old patient id, we listed all the old EDF names associated with it. If the old EDF had been mapped to a new EDF name, we list that next to it. We also list the new patient id, as well as the names of all the EDFs in the header dataset, and the names of all the EDFs in the master lookup table, which corresponded to this new patient id. Some patients had EDFs which all matched up perfectly from this process. These were placed into "included_mapped_without_errors" sheet. Others had EDFs where the columns did not all match, but at least every old EDF had a new EDF mapping. These were placed into "included_mapped_with_errors" sheet. Other patients had EDFs which were unmapped; these were placed into "included_unmapped". Finally, a fourth sheet "excluded_mapped_and_unmapped" contained all the patients

whose new id's were in the exclude list, whether or not their EDFs could be mapped.

7. We created a mapping of the old EDFs to the correct starting timestamp. Most of these timestamps were extracted using the old to new EDF name mapping. For old EDFs which had no mapping, we guessed that the timestamp was correct and made a note of it.

8. We made another file `merge_info.csv` which listed for each unexcluded old patient id all the old EDF names and their correct starting timestamps. We noted in a separate column if the timestamp was guessed from the old EDF name. We used the EDF duration information in `eeg_data_size_info.csv` to get the end timestamps. We listed these old EDFs, sorted by starting timestamp, and also listed the time between each ending timestamp and the next starting timestamp. There were cases where timestamps overlapped, in which case this number was negative; there were also cases in which the gap between EDF times was on the order or months, or even years. We then manually reviewed the "guessed timestamp" cases. If the gaps were all short, then we assumed that the guessed timestamp was correct. Otherwise, we marked that that EDF had unreliable timestamp, and we excluded it in our analysis.

Note that all the CSVs mentioned above can be found in the `patient_outcome_info` subdirectory of our code repository. After excluding patients with no outcomes, or whose timestamps were unrecoverable, we were left with 460 total patients whose data we used for our analysis.

Appendix C

The Repository

C.1 The Repository: General Overview

In this section, we present an overview of the repository. The repository can be accessed on Github at `https://github.com/ALFA-group`. There are two primary sections: the MATLAB code and the Python code, contained within the `matlab` and python directories, respectively, of the `src` folder of the repository. The primary portion of the pipeline is written in MATLAB, including reading the EDF data files, preprocessing, burst suppression detection, and calculating burst pair similarity. There are scripts included in the repository which run this pipeline on a given list of EDFs and output the results to a file. The Python code is used to analyze these results and generate figures and insights. The organization of the repository is summarized following.

```
matlab/
└── config_and_params/ - local configuration information and pipeline parameters
```

- `libraries/` - installation of various libraries used in the repository
- `pipeline/` - functions which form the main processing pipeline
  - `prep_and_artifact/` - preprocessing and artifact detection
    - `prep_and_artifact.m` - main function of the directory
  - `detect_bs/` - burst suppression detection
    - `detect_bs.m` - main function of the directory
  - `similarity/` - similarity calculation on burst pairs
    - `similarity.m` - main function of the directory
- `pipeline_wrappers/` - functions which wrap portions of the above processing pipeline
- `plot/` - functions to visualize the EEG
- `save_and_load/` - functions which save and load output created by running the pipeline
- `scripts/` - scripts which run the pipeline on input EDFs
- `util/` - various utility functions

`python/`

- `archives/` - various functions and scripts which are no longer used or updated
  - `data_clean/` - various functions and scripts used to sanitize our specific dataset
- `ml/` - Notebooks and functions for training a recurrent neural network to encode bursts
- `plotting_notebooks/` - scripts which generate the figures and analysis given in the results section
- `readers/` - helper classes which read in output files from MATLAB scripts or patient information CSV files
- `run_matlab_scripts/`
  - `run_matlab_script.py` python wrapper which starts a process to run a MATLAB script

└── `utils.py` - various utility functions

C.2 The Repository: Usage guide

This section will discuss the usage of the repository, describing the process from running the pipeline to generating all the figures and analyses included in this paper. We walk through all the steps an imaginary user Bob would take in order to reproduce the results of this paper.

C.2.1 Preliminaries

We assume Bob has MATLAB and Python installed. We used MATLAB version 9.3.0.713579 (R2017b) and Python version 2.7.14. We performed our processing on a 64-bit Linux machine running Ubuntu 14.04. We also assume Bob has the repository cloned, and that he has a dataset of EEG recordings following the European Data Format (EDF). The `src/matlab/libraries` contains all the libraries needed to run the MATLAB portion of the pipeline. However, some Python packages are required to be installed. These requirements are given in the `src/environment.yml` file. We used Anaconda as our Python package manager; the easiest way to replicate our environment is to use Anaconda and create a new environment from this `environment.yml` file.

Details specific to running in CSAIL environment

We performed our processing on instances created by CSAIL Openstack. In CSAIL Openstack, our instances were created by launching an instance booting from the CSAIL-Ubuntu-14.04LTS image with 32 GB of memory and 16 VCPUs. By adding our ssh keys during creation, we can ssh into the instance as the root user `ubuntu`. Upon instance creation, we login as `ubuntu` and run `src/setup.sh` contained in the repository, which installs git and NFS-common, and mounts the NFS directory containing our EEG dataset into our AFS space. If a new user Bob is using this script, he must first replace the '/afs/csail.mit.edu/u/t/tzhan/' path in the last line with the path to his own AFS space. After running this script, we run `src/make_xcorr_installations.sh`. This gives us access to MATLAB on the instance and installs some libraries necessary to run the 'xcorr' command in MATLAB. At this point, when we log in with our CSAIL username, we have the NFS directory mounted into our home directory. We clone the repository here as well. At this point, there should be no need to log in as the root user.

C.2.2 Running the MATLAB pipeline

To run the MATLAB pipeline, there are two main directories Bob should be familiar with: `src/matlab/pipeline_wrappers` and `src/matlab/scripts/run`. We describe the functions and scripts in these two locations and how to use them.

util/get_pt_from_fname.m

Before Bob can use any of the functions or scripts in the repository, he must ensure that the `src/matlab/util/get_pt_from_fname.m` function matches his specific dataset. If Bob is using the same CSAIL dataset we used, he will not need to change this function. Otherwise, he needs to edit this function so that it takes in the name of an EDF in his dataset and returns the patient id associated with it. The pipeline scripts call this function when saving output so that they can organize the output for all the EDFs corresponding to a given patient into the same directory. We rely on having this specific output directory structure in later analysis when we read these results.

pipeline_wrappers

There are two functions in this directory: `pipeline_up_to_detect_bs` and `pipeline_up_to_similarity`. These take as input a path to an EDF file, and return the results of running the pipeline up to a certain step.

`pipeline_up_to_detect_bs` The input to this function is the path to an EDF file. The function reads the EDF, runs preprocessing and artifact detection, and then runs the burst suppression detection algorithm on the resulting EEG. There are several outputs of this function:

- `eeg` - A matrix of size *num_channels* by *num_samples*. This is the matrix containing the EEG data, after preprocessing.
- `bs_ranges` - A matrix of size *num_bs_episodes* by 2 that lists the start index

and end index of every burst suppression episode detected in the EDF.

- `global_zs` - A binary vector of length *num_samples* containing the z^{GLOBAL} signal.

- `bsr` - A vector of length *num_samples* containing the burst suppression ratio signal.

- `burst_ranges_cell` - A cell of length *num_bs_episodes* where element i is a matrix of length *num_bursts_i* by 2 containing the start index and end index of every burst within burst suppression episode i.

- `local_zs` - A binary matrix of length *num_channels* by *num_samples* where row i is the local $z^{(i)}$ signal for channel i.

`pipeline_up_to_similarity` The input to this function is the path to an EDF file, as well as the name of the similarity function to use (either "dtw" or "cross-correlation"). The function first gets the results of burst suppression detection. If the `use_saved_detect_bs_output` parameter is set and the there exist saved results, the function loads these results. Otherwise, it runs `pipeline_up_to_detect_bs`, and if the `save_detect_bs_output` parameter is set, it saves these results for future use. The function returns two output variables:

- `similarities_cell` - A cell of length *num_bs_episodes* where element i is the vector of similarities v_i for all pairs of bursts in burst suppression episode i. The length of v_i is equal to the number of burst pairs in the episode, or $n_i(n_i - 1)/2$, where n_i is the number of bursts in episode i.

- `filtered_burst_ranges_cell` - A cell of length *num_bs_episodes* with the same format as `burst_ranges_cell`. The difference is that `burst_ranges_cell` contains all bursts in each episode, while `filtered_burst_ranges_cell` contains only the bursts that were used for the similarity calculation—namely, the bursts that fall between "min burst time" and "max burst time."

scripts/run

The `scripts/run` directory contains two scripts which run the pipeline up to a certain point for a list of EDFs. To use these, Bob first creates a txt file, where each line is the path to an EDF file he wishes to analyze. We refer to this as the "todo files list." The scripts run the pipeline and save output for all the EDFs in the list for which saved output does not already exist. This way, if Bob has started running this script and the process stops for any reason before completion, Bob can easily rerun the script without needing to edit the "todo files list." We found this feature very useful, as the script often failed partway through for a variety of reasons including corrupt EDFs, memory issues, and other unexplained process terminations.

`Config.m` Before running any scripts, our user Bob must now edit `config_and_params/Config.m` to match his local configuration. He sets `repo_dir` to the path to the repository, `output_dir` to the location where he wants the outputs saved, and `todo_files_list`. The config file also contains a variable `save_images_dir` which points to the location where Bob wants any output images saved. This variable is used only for scripts within `scripts/visualize`.

The `scripts/run/describe_bs.m` script runs `pipeline_up_to_detect_bs` and saves the output for all EDFs listed in `todo_files_list`. The `scripts/run/write_similarities.m` runs `pipeline_up_to_similarity` with "dtw" and "cross-correlation" for each EDF and saves the results.

`run_matlab_script.py`. Bob can use the python wrapper script `python/run_matlab_scripts/run_matlab_script.py` to run any of the scripts provided in the MATLAB `scripts` directory. This python wrapper starts a process which runs the MATLAB script in the background; it opens up MATLAB, adds the necessary directories of the repository, and calls the desired scripts. It pipes all output to a log file and also displays the output on the terminal. To use this wrapper, Bob enters into the directory containing the wrapper and calls
`python run_matlab_script.py relative_path_to_script run_tag`,
where `relative_path_to_script` is the path to the MATLAB script Bob wishes to run, relative to the `matlab/scripts` directory of the repository, and `run_tag` is a short tag describing the run. The log-file containing the output of this run will contain the name of the script and the run tag. For example, if Bob wants to run `scripts/run/write_similarities.m` for all the EDFs from the MGH hospital, he can call
`python run_matlab_script.py run/write_similarities.m _mgh`,
and all the output will be written to `write_similarities_mgh.log`.

Details specific to running in MIT CSAIL (Computer Science and Artificial Intelligence Laboratory) environment. If Bob is running the script from one of the CSAIL instances which he created as described in section C.2.1, he first logs in

with his CSAIL username. To start the process and ensure that it continues running after he logs out, he creates a longtmux session which will give his process access to his AFS space for many days. It is important that Bob uses `longtmux` rather than `tmux`, since the latter might lose AFS access before the process is finished. He starts the process from his longtmux session using the same command described previously.

Adjusting Parameters of the Pipeline

The `*Params.m` files in the `matlab/config_and_params/` directory define the parameters which are used in the pipeline. There are three parameter files, each one containing the parameter associated with the portion of the pipeline matching its name. If Bob wishes to run the scripts with any different parameters, he should edit the definitions in these files. The descriptions of each parameter are described in the files themselves; they also correspond roughly with the names of the parameters given in Table A.1.

C.2.3 Visualizing the MATLAB pipeline

We provide a few functions for plotting the EEG and the results of various portions of the pipeline.

plot

The `src/matlab/plot` folder provides a few functions for plotting EEGs. There are three top-level functions, which we describe here.

plot_eeg_noninteractive.m `plot_eeg_noninteractive.m` creates a MATLAB plot for a portion of the EEG signal. The required input is an eeg object; this is the type of object returned by the preprocessing function `pipeline/prep_and_artifact/prep_and_artifact.m` and its subfunction `pipeline/prep_and_artifact/load_eeglab_eeg.m`. There are several optional parameters:

- `start_index` - index at which to start plotting
- `end_index` - index at which to stop plotting
- `winlength` - number of seconds to plot. Only one of `winlength` and `end_index` should be specified.
- `is_shaded` - binary vector of length equal to eeg. Indices where this vector equals 1 are shaded.
- `colors` - cell array of colors to use
- `color_indices` - matrix of indices into colors of size same as eeg. If $color\_indices[i][j] = k$, the sample at $eeg[i][j]$ will be colored $colors\{k\}$.
- `global_zs` - vector of global zs to additionally plot for burst suppression detection visualization. If specified, `bsr` must also be specified.

- `bsr` - vector of bsr to additionally plot. If specified, `global_zs` must also be specified.

plot_eeg_interactive.m `plot_eeg_interactive.m` is similar to `plot_eeg_noninteractive.m`, but rather than creating a static plot, it allows the user to scroll through the EEG. After Bob calls this function, he can press 'f' to go forward and plot the next window of the EEG, 'b' to go back one window, or 'e' to exit the interactive plot. The optional arguments to this function are the same as those of the noninteractive version.

save_eeg_plot.m `save_eeg_plot.m` is similar to `plot_eeg_noninteractive.m`, except that it saves the plot rather than displaying it. In addition to requiring the eeg object for input, it requires the path to the directory (`save_dir_name`) and the filename (`save_filename`) in which to save the plot. Again, it accepts the same optional arguments as `plot_eeg_noninteractive.m`.

run/visualize

We provide three scripts for visualizing preprocessing and burst suppression detection inside `src/matlab/scripts/visualize`: `visualize_preprocessing.m`, `visualize_bs_detection.m`, and `interactive_vis_bs_detection.m`.

visualize_preprocessing.m To run `visualize_preprocessing`, Bob must have his `todo_files_list` and his `config_and_params/Config.m` file set with the proper configuration as described in section C.2.2. When Bob runs the script, it runs prepro-

cessing and artifact detection for all the EDFs in the todo_files_list. For each EDF, it chooses 5 random 15-second portions of the EEG to plot; it makes one plot of the EEG signal before preprocessing (but with bipolar montage applied) and one plot of the EEG signal after the filters, as in Figures 3.2a and 3.2b. If any portion of the snapshotted window is detected as artifact, that portion is shaded blue, as in Figure 3.3. These plots are all saved into the directory specified by `save_images_dir` in the `config.m` file. Using this script, Bob can easily review the preprocessing results on several EDFs at a time.

visualize_bs_detection.m The `visualize_bs_detection.m` script runs the burst suppression detection pipeline for all the EDFs in the todo_files_list and plots the EEG signal, the BSR, and the global zs signal for 5 random 30-second snapshots. This produces a plot as in Figure 3.4.

interactive_vis_bs_detection.m The `interactive_vis_bs_detection.m` visualizes burst suppression detection for a single EDF. Unlike the `visualize_bs_detection.m` script, it does not save any output, but rather is meant to be used interactively. He edits the first line defining the `file_path` variable to be the path to the EDF he wishes to visualize. The result is an interactive eeg plot displaying the EEG signal, BSR, and global zs signal, starting at index 0 of the signal. Bob can navigate through this interactive plot as described in section C.2.3.

C.2.4 Running the Python notebooks

In this section, we describe how Bob re-creates the figures presented in our results section. All the code for these plots is located in `src/python/plotting_notebooks`.

Patient Outcome Info

If Bob is not using the CSAIL dataset, he first needs to change the information in the `patient_outcome_info` directory of the repository to correspond with his own dataset, as the files within that directory in the repository all refer to the CSAIL dataset. There are three files which Bob must change:

1. `exclude_patient_sids.txt` - This must be a text file containing a list of any patients which are to be excluded from analysis. The list should be written so that each line of the file contains the name of an excluded patient, following the current formatting. If there are no patients to exclude, this file should be left blank.

2. `merge_info.csv` - This must a CSV file containing a list of all the EDFs in the dataset, sorted first by patient, and then by starting timestamp. The column "sid" must give the patient name, the "timestamp" column should be the starting timestamp, and the "nsamples" column should give the number of samples within the EDF record. The "duration" column should be derived from the "nsamples" column using the appropriate sample rate, and the "end_timestamp" column should be derived using the duration and the start timestamp. The "time_to_next_edf" column should be derived by computing the time difference between the end timestamp of the EDF on the current

row and the start timestamp of the next EDF for the same patient. The "is_timestamp_guessed_from_old_edf_name" and "is_guessed_timestamp_correct_seeming" columns are mostly irrelevant for Bob. Unless Bob thinks some of the values in the "timestamp" column might be wrong, Bob should simply mark "FALSE" in the entire "is_timestamp_guessed_from_old_edf_name" column, and leave blank the entire "is_guessed_timestamp_correct_seeming" column.

3. `new_outcomes.csv` - This must be a CSV file containing at least two columns: "sids", which gives the patient name, and "bestCpcBy6Mo", which gives the patient outcome as a CPC score taken within six months of the patient's cardiac arrest incident.

Note that there are other subdirectories within the `patient_outcome_info` directory. Bob should ignore these unless he is using the CSAIL dataset and needs to perform further cleaning on it.

utils.py

If Bob is not using the CSAIL dataset, he will likely also need to edit some of the functions within the `src/python/utils.py` file. First, he should edit to update the sample rate. In the file, we assume a sample rate of 200. If this is not the case for Bob, he must edit the `sec_to_samples` and `samples_to_sec` to replace the 200 with the proper sample rate. In addition, there a section of the file labeled "Utilities for parsing edfs, specific to the csail/icare dataset." These are functions which are used for parsing EDF names in the CSAIL dataset, since EDFs in the CSAIL dataset have names containing the patient hospital, the patient id, and the

start timestamp of the recording. Ideally for simplicity, Bob should ensure that his dataset has EDFs containing these pieces of information as well. He must then edit the functions within the section labelled "Utilities for parsing edfs, specific to the csail/icare dataset" to correspond to his dataset naming format. Each method contains comments explaining their expected outputs, so Bob can read these for guidance before editing the methods.

Describe burst suppression

We now move on to discuss how to make the plots seen within Section 4.1. Within the directory `src/python/plotting_notebooks/describe_bs`there are two notebooks which Bob uses.

bsr_zs_over_time_heatmap.ipynb Bob runs `bsr_zs_over_time_heatmap.ipynb` to create figures as in Figures 4.7 or 4.5. Bob edits the `output_dir` variable in the first cell to point to the path where the output from the detect burst suppression pipeline was saved. This should be the `describe_bs` subdirectory of the `output_dir` Bob defined in `Config.m`. Bob defines the input parameters to `get_zs_bsr_over_time_heatmap`:

- `zs_or_bsr` - a string indicating which signal to plot; either "zs" or "bsr"
- `outcome_lambda` - boolean function which returns whether to include a patient in the plot
- `ds_rate` - Downsampling rate for creating the full matrix of global zs or bsr for all the patients

- `max_num_hours` - Maximum number of hours to plot
- `plot_in_parts` - If True, makes multiple plots, each sized to fit a pdf page. Otherwise, makes one plot with all patients.

describe_bs_histograms.ipynb Bob uses `describe_bs_histograms.ipynb` to create the histograms seen in Figures 4.1, 4.2, and 4.3. He again edits the `output_dir` variable as in the `bsr_zs_over_time_heatmap.ipynb` notebook. The first time Bob runs this, he sets `read_output = True`. The notebooks saves some intermediate output objects into pickle files. If Bob wishes to save time by using these output objects instead of re-creating them, he sets `read_output = False`. Running each cell will produce the desired figures. Some cells provide histograms on a per-patient, or even per-episode, basis. These create a set of histograms which Bob can flip through interactively.

Similarity plots

We now describe how Bob create the plots related to the similarity results.

similarity_scaling.py Before creating the plots in this paper, Bob must understand the different scaling functions which are used to scale the raw DTW and cross-correlation similarity numbers to be between 0 and 1, where 0 represents completely dissimilar, and 1 represents identical. We describe in section 4.2 the exponential function. However, there are other options which Bob can use to scale, defined in `python/plotting_notebooks/similarity/similarity_scaling.py`, whose main function is `scale_arr`. We define a few types of scaling, including "noscale", which

does no scaling at all; "linear", which scales everything linearly so that the minimum raw cross-correlation value scales to 0 and the maximum scales to 1, and vice versa for DTW values;"exp", which scales exponentially; and "sigmoid", which scales sigmoidally. To use any of these, Bob must first adjust the parameters of the scaling functions to match his raw values.

Within `src/python/plotting_notebooks/similarity` there are two notebooks which Bob uses.

similarity_over_time.ipynb Bob uses `plotting_notebooks/similarity/similarity_over_time.ipynb` to create the plots as in Figures 4.20 and Figures 4.21. Bob edits the `similarity_output_dir` variable in the first cell after the imporst to point to the path where the output from the similarity pipeline was saved. This should be the `similarity` subdirectory of the `output_dir` Bob defined in `Config.m`. However, at this point, if Bob has run the similarity pipeline several times with different parameters and has each set of results saved in different directories, then he points the variable to the path containing the specific set of results he wishes to analyze. He edits the `output_dir_tag` variable to be a short string describing the specific output directory. This way, when plots are saved, their names include the output directory from which their results are derived. There a few variables which Bob sets in the cell labelled "Define parameters":

- `similarity_fn` - the similarity function to use; either "xcorr" or "dtw"
- `scale_fn` - the type of scaling to apply to the similarity scores; passed to `scale_arr`; this can be "noscale", "linear", "exp", or "sigmoid"

- `scale_param` - parameter of the scale function; passed to `scale_arr`; applies only to `scale_fn = ‘‘exp"`;
- `min_episode_length` - minimum duration in minutes of a burst suppression episode used in analysis
- `neighbor_size` - number of neighbors to use when computing summarized similarity for a burst; this is the W parameter in subsection 4.2.5
- `target_outcome` - string; either “good” or “bad”; plots only patients matching this outcome; used to define `outcome_lambda`
- `ds_rate` - Downsampling rate for creating the full matrix of global zs or BSR for all the patients
- `max_num_hours` - Maximum number of hours to plot
- `plot_in_parts` - If True, makes multiple plots, each sized to fit a pdf page. Otherwise, makes one plot with all patients.

similarity_analysis.ipynb Bob uses `similarity_histogram.ipynb` to create the remaining figures. As before, he defines `similarity_output_dir`. He also defines `pickle_tag`, which serves the exact same function as `output_dir_tag` in `similarity_over_time.ipynb`. He then sets `min_episode_length`, `scale_fn`, and `scale_param`, which serve the same functions as they did in `similarity_over_time.ipynb`. The `run_results` variable is similar to `read_output` in `describe_bs_histograms.ipynb`. The first time Bob runs the notebook, `run_results` should be set to True. The code saves intermediate output, and the following times, Bob can use the saved inter-

mediates with `run_results = False`. In the "Similarity/Distance vs Outcomes" section, Bob edits the cell defining the summary measure, editing the variables `summary_measure` and `summary_percentile_decile` according the comments in the notebook.

Appendix D

Acknowledgements

We would like to acknowledge the many collaborators who contributed to this project, including, but not limited to, Una-May O'Reilly, Abdullah Al-Dujaili, Brandon Westover, Edilberto Amorim, and Alessandro de Palma.

Bibliography

Donald W Marion. Coma due to cardiac arrest: prognosis and contemporary treatment. *F1000 Medicine Reports*, nov 2009. doi: 10.3410/m1-89. URL https://doi.org/10.3410/m1-89.

David E Bateman. Neurological assessment of coma. *Journal of Neurology, Neurosurgery & Psychiatry*, 71(suppl 1):i13–i17, 2001. ISSN 0022-3050. doi: 10.1136/jnnp.71.suppl_1.i13. URL http://jnnp.bmj.com/content/71/suppl_1/i13.

Quentin Noirhomme, Rémy Lehembre, Zulay del Rosario Lugo, Damien Lesenfants, André Luxen, Steven Laureys, Mauro Oddo, and Andrea O. Rossetti. Automated analysis of background EEG and reactivity during therapeutic hypothermia in comatose patients after cardiac arrest. *Clinical EEG and Neuroscience*, 45(1):6–13, jan 2014. doi: 10.1177/1550059413509616. URL https://doi.org/10.1177/1550059413509616.

Jeannette Hofmeijer, Marleen C. Tjepkema-Cloostermans, and Michel J.A.M. van Putten. Burst-suppression with identical bursts: A distinct EEG pattern with poor outcome in postanoxic coma. *Clinical Neurophysiology*, 125(5):947–954, may 2014. doi: 10.1016/j.clinph.2013.10.017. URL https://doi.org/10.1016/j.clinph.2013.10.017.

Marleen C. Cloostermans, Fokke B. van Meulen, Carin J. Eertman, Harold W. Hom, and Michel J. A. M. van Putten. Continuous electroencephalography monitoring for early prediction of neurological outcome in postanoxic patients after cardiac arrest. *Critical Care Medicine*, 40(10):2867–2875, oct 2012. doi: 10.1097/ccm.0b013e31825b94f0. URL https://doi.org/10.1097/ccm.0b013e31825b94f0.

Jun Rho Yoon, Yee Suk Kim, and Tae Kwan Kim. Thiopental-induced burst suppression measured by the bispectral index is extended during propofol administration compared with sevoflurane. *Journal of Neurosurgical Anesthesiology*, 24(2):146–151, apr 2012. doi: 10.1097/ana.0b013e3182429829. URL https://doi.org/10.1097/ana.0b013e3182429829.

Martijn Beudel, Marleen C. Tjepkema-Cloostermans, Jochem H. Boersma, and Michel J. A. M. van Putten. Small-world characteristics of EEG patterns in postanoxic encephalopathy. *Frontiers in Neurology*, 5, jun 2014. doi: 10.3389/fneur.2014.00097. URL https://doi.org/10.3389/fneur.2014.00097.

Peter B. Forgacs, Hans-Peter Frey, Angela Velazquez, Stephanie Thompson, Daniel Brodie, Vivek Moitra, Leroy Rabani, Soojin Park, Sachin Agarwal, Maria Cristina Falo, Nicholas D. Schiff, and Jan Claassen. Dynamic regimes of neocortical activity linked to corticothalamic integrity correlate with outcomes in acute anoxic brain injury after cardiac arrest. *Annals of Clinical and Translational Neurology*, 4(2):119–129, jan 2017. doi: 10.1002/acn3.385. URL https://doi.org/10.1002/acn3.385.

Hana Moshirvaziri, Nima Ramezan-Arab, and Shadnaz Asgari. Prediction of the outcome in cardiac arrest patients undergoing hypothermia using eeg wavelet entropy.

2016 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), 2016. doi: 10.1109/embc.2016.7591550.

Andriy Temko, Orla Doyle, Deirdre Murray, Gordon Lightbody, Geraldine Boylan, and William Marnane. Multimodal predictor of neurodevelopmental outcome in newborns with hypoxic-ischaemic encephalopathy. *Computers in Biology and Medicine*, 63:169–177, aug 2015. doi: 10.1016/j.compbiomed.2015.05.017. URL `https://doi.org/10.1016/j.compbiomed.2015.05.017`.

Marleen C. Tjepkema-Cloostermans, Jeannette Hofmeijer, Albertus Beishuizen, Harold W. Hom, Michiel J. Blans, Frank H. Bosch, and Michel J. A. M. van Putten. Cerebral recovery index. *Critical Care Medicine*, 45(8):e789–e797, aug 2017. doi: 10.1097/ccm.0000000000002412. URL `https://doi.org/10.1097/ccm.0000000000002412`.

Michel J. A. M. van Putten, Jeannette Hofmeijer, Barry J. Ruijter, and Marleen C. Tjepkema-Cloostermans. Deep learning for outcome prediction of postanoxic coma. pages 506–509, 2018.

M. Brandon Westover, Mouhsin M. Shafi, ShiNung Ching, Jessica J. Chemali, Patrick L. Purdon, Sydney S. Cash, and Emery N. Brown. Real-time segmentation of burst suppression patterns in critical care EEG monitoring. *Journal of Neuroscience Methods*, 219(1):131–141, sep 2013. doi: 10.1016/j.jneumeth.2013.07.003. URL `https://doi.org/10.1016/j.jneumeth.2013.07.003`.

Jingzhi An, Durga Jonnalagadda, Valdery Moura, Patrick L. Purdon, Emery N. Brown, and M. Brandon Westover. Spatial variation in automated burst suppression detection in pharmacologically induced coma. In *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Soci-*

ety (EMBC). IEEE, aug 2015. doi: 10.1109/embc.2015.7320109. URL https://doi.org/10.1109/embc.2015.7320109.

Thanawin Rakthanmanon, Bilson Campana, Abdullah Mueen, Gustavo Batista, Brandon Westover, Qiang Zhu, Jesin Zakaria, and Eamonn Keogh. Searching and mining trillions of time series subsequences under dynamic time warping. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '12*. ACM Press, 2012. doi: 10.1145/2339530.2339576. URL https://doi.org/10.1145/2339530.2339576.

Eric W. Weisstein. Cross-correlation. URL http://mathworld.wolfram.com/Cross-Correlation.html.

1924 Safar, Peter and Ake Grenvik. *Brain failure and resuscitation*. New York : Churchill Livingstone, 1981. ISBN 0443081433.