IEEG-Portal Documentation

Collaborative Research in the Cloud

University of Pennsylvania*
Mayo Clinic

April 15, 2014

Abstract

This document describes the functionality of the IEEG-Portal and its exposed API for Java and Matlab. It contains multiple example tutorials and a thorough explanation of the terminology used on the IEEG-Portal. As this project evolves, this document will be updated and relevant information will be added.

*Supported by: United States National Institutes of Health Grant #1 U24 NS063930-01

Contents

1. Introduction

The need for a robust management system for scientific data sharing and analysis has increased significantly over the last decade. Despite multiple efforts, there is currently no single platform that is widely used to share and interact with large biomedical scientific datasets.

A platform that facilitates these needs requires a secure way to access data, and provide an intuitive way to manage sharing and permissions per dataset. It needs to provide easy access to the data using 3rd party software, be able to handle large amounts of data, and include capabilities to scale up in size when demand requires this.

The IEEG-Portal (`http://www.ieeg.org`) is a cloud based platform that is developed around these requirements. At this time, it specifically targets the epilepsy research community, but it can easily be extended to other research areas. Its flexible platform can be used for data-sharing, and collaborative data analysis on projects containing large time-series datasets. The portal provides a highly efficient way to provide random access to very large time-series data in the cloud, enabling users to use the cloud-based data as their primary data-source.

Portal development

The IEEG-Portal is developed using the Google Web Toolkit in Java, and JavaScript. It is hosted on Amazon's EC2 service and the data is stored on Amazon's S3 service with reduced redundancy. All meta-information is associated with the datasets as well as user-information and portal-state information is provided by Amazon Relational Database Storage (RDS).

The IEEG-Portal provides webservices to interact with the portal for data-fetching and for uploading annotations to the cloud. A client Java library is available with methods to access the webservices and a Matlab Toolbox is provided for users who want to access the IEEG-Portal through Matlab.

2. The Webconsole

The webconsole of the IEEG-Portal provides access to the datasets, the data-management tools, an EEG-Viewer and serves as the primary hub to the user-community. The console consist of a tabular workspace with default tabs for 1)Updates, 2)Work, 3)Datasets, and 4)Tools. In the top right, there are three buttons which allow you to search for datasets, save changes that you made to a dataset, and share datasets that you created with colleagues.

The Datasets panel shows datasets that were returned by your search-request, as well as datasets that are currently opened in the web-console. When you click on the arrow next to the name of the dataset, additional datasets that are derived from the dataset are shown. These could be datasets that you created, or datasets that are shared with you by a collaborator.

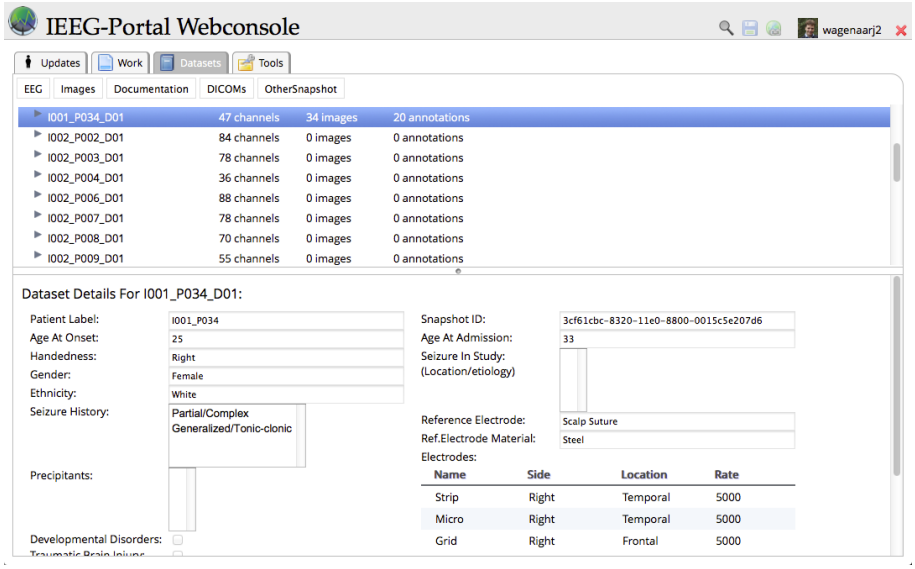

Figure 2.1: The "Datasets" panel of the IEEG-Portal. Here, you can browse through datasets and see detailed meta-information.

Once you select a dataset, you can show the EEG, Documentation, or Images by selecting the option in the tab-menu bar. The "Datasets"-tab menu also contains a button to search for specific datasets by their name. Finally, the menu contains a button to download the DICOM files that are associated with the dataset if these are available.

Dataset privileges

The IEEG-Portal has three levels of authorization for datasets. This allows a user to create a new dataset and make it private, read-only, or open to collaborators. The default privilege level for datasets on the portal is "Read-Only", but users can derive new datasets for which they will have "Owner" privileges. The various levels of privileges are described below:

- **Read-Only:**A user with Read-Only permissions on a dataset can read the data and annotations contained in the dataset but cannot alter it in any way.
- **Editor:** A user with Edit permissions has all the permissions of Read-Only and can add/modify annotation-layers and annotations, can add and delete channels in a dataset, and rename the dataset
- **Owner:** A user with Owner permissions has all the permissions of an Editor, can delete the dataset and change permissions for other users for this dataset.

Searching for data

You can search for specific datasets on the portal by clicking on the magnifying glass in the top-right corner of the webconsole. A separate window will allow you to choose between human and non-human datasets and specify search-criteria. You can choose to either replace the results in the Dataset-browser or append previously returned results. If you do not specify any restrictions in the search window, all datasets will be returned.

Saving and Sharing

Datasets that have been modified in the webconsole by the user can be saved using the "save" button in the top-right corner of the webconsole. Depending on the level of access that the user has for the dataset, the changes are either saved in the dataset, or the user is asked to derive a new dataset from the read-only dataset in order to save the changes. In this case, the user is asked to provide a new unique name for the derived dataset after which the changes are saved. Both saving and sharing buttons are only enabled in the IEEG-Viewer panel.

By default, new datasets are not shared with anybody on the portal. In order to provide access to collaborators, you have to add the collaborator to the list of users that can read/edit the dataset. You can do this using the "share" button in the top-right of the webconsole This brings up a window (Figure 2.2) which can be used to manage access to the dataset for other users.

2.1 Updates and contributions

The "Updates" tab shows your user account information and latest updates on the portal. It shows how many people have accessed data and tools that you provided and a history of interactions with the portal that are related to your account. These updates will also include messages from other users, notifications of new releases etc.

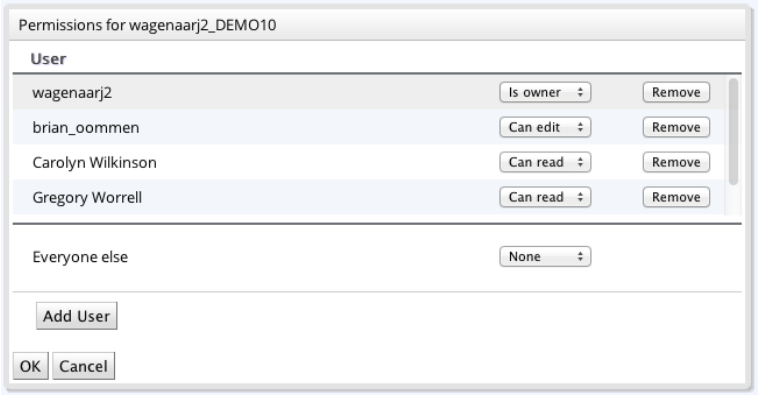

Figure 2.2: Manage permissions for the dataset on the portal. You can provide custom privileges for individual users, or/and set default permissions for other users on the IEEG-Portal.

The update feed will show any updates about datasets and tools that you added to your favourites. This allows you to quickly see changes and messages associated with these items. As the IEEG-Portal matures, we expect that this page will increase in functionality and provide a comprehensive summary of the activity on the portal related to the user.

2.2 Collaborating in projects

The work tab provides information about recently used datasets, and shows your favourite datasets. You can also create project groups in this tab. A project contains a group of users and datasets. You can use this to specify a set of datasets that should be analyzed together. This feature is currently fairly limited but will be extended in future releases of the IEEG-Portal.

2.3 The tool exchange

The tools tab can be used to share code and analysis. Here, you will be able to find analysis scripts provided by the IEEG-Portal as well as user-provided tools. Each tool has a unique tool-identifier which can be associated with a dataset.

Associating a tool with a derived dataset was required in previous versions of the IEEG-Portal but is now optional. However, it is good practice to upload your code once you share results, and we recommend that you associate any shared dataset with the tool that generate results in that dataset. Previous versions of the portal also used the Tool tab-panel to initiate jobs on the job-server. We depreciated the job-server as it was not used by many people in favor of providing random access to datasets using the web-services.

We are working on making the tool-exchange more versatile by allowing the tools to point to 3rd party code-repositories or external web-pages. This would allow users to share their analysis tools using repositories such as Google code and Github

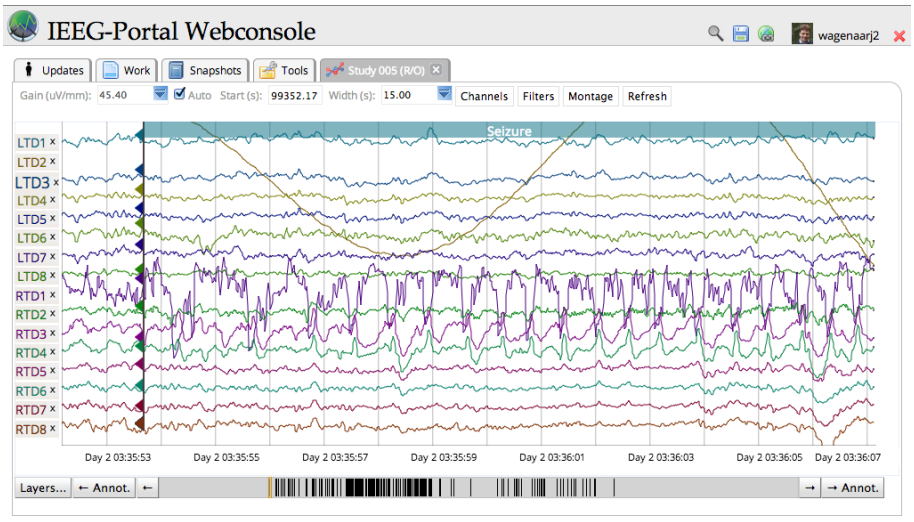

Figure 2.3: The EEG-Viewer pane of the IEEG-Portal

2.4 Viewing EEG datasets

The EEG viewer in the portal can be used to view large time-series datasets. It provides functionality similar to off-line EEG viewers including, re-montaging, filtering and annotations. The EEG viewer allows the user to create annotate layers and manually add annotations to the dataset.

Paging through data

The EEG viewer contains two views; the multi-channel time-trace window and the global event timeline. The latter represents the entire length of the dataset and can be used for quickly paging through events and provides a reference for the timespan visualized in the time-trace window. There are various ways to page through the data in the EEG-Viewer:

- **Fix step paging:** Use the left and right arrow keys to page forward and backward. You can also use 'n' to page forward.
- **Click and drag:** You can left-click and drag the EEG-traces to page forward and backward.
- **Using paging buttons:** You can page forward and backward using the paging buttons below the EEG-traces. You can also jump to the next/previous annotation.

Selecting Channels

Even though the portal can show many channels simultaneously, it is often useful to limit the viewer to a subset of channels. There are a couple of ways that you can use to select a subset of channels:

- **Using "Channels" button:** Clicking on the "Channels" button will display a new window from which you can select a subset of channels in the dataset.
- **Removing single channel:** You can remove a single channel by clicking on the "x" button next to the channel label.
- **Shift-Click:** Multi-scale zooming: You can select a subset of channels and time by holding the shift-key while click-and-dragging the selected zoom level. Use the backspace key to go back to the previous zoom level.

Filtering

The portal provides various filter settings for the data. By default, the data is filtered using linear interpolation without an anti-aliasing filter. In order to be able to present the data in a fast and efficient way, the portal takes into account the resolution of the display that the data is visualized on. In some cases, this can slightly change the appearance of the data compared to external EEG-Viewers.

Remontaging

The portal currently has limited support for re-montaging. You can manually specify a reference for each channel by clicking on the "Montage" button in the EEG-Viewer. This feature will be improved in the future releases of the IEEG-Portal.

Annotation layers

Each dataset has one or more annotation layers. Each of these layers can have multiple annotations with different types. However, it is recommended to create a separate annotation-layer for each type of annotation (i.e. Seizures, Artifacts, Spikes). You can select which layers you want to show in the EEG-Viewer by clicking on the "Layers..." button.

Manually annotating data

You can manually annotate dataset and share your annotations with collaborators. Unless you are the owner of a dataset (e.g. you created it), you'll have to save the annotations to a new dataset that is derived from the original. This way, everybody can create their own annotations without changing the original dataset. Once saved, you can decide to share your new dataset with collaborators or provide read-access for everybody.

To annotate the data, you first select a set of channels by double clicking at the start-time of the annotation. Then move the cursor to the last channel that you want to annotate and double- click again. This will bring up an annotation widow that allows you to select an annotation type, description, etc. You can also refine the annotated channels at this point. Once you click "Okay" the annotation is created.

3. IEEG-Webservices

One of the strengths of the IEEG-Portal is that there is an extensive API to interact with the portal using external clients. The IEEG-Services are accessible by a Java library on the client side and an easy to use Matlab Toolbox for users that are less familiar with Java. Figure 3.1 shows the dataflow between the cloud and the local client when using the API. The data is stored and send to the client using lossless compression to maximize efficiency and to lower storage and data-access costs for the Amazon services.

We are currently working on implementing a local persistent cache using MongoDB to store downloaded snippets of data locally. Using a local, or shared persistent cache can provide significant efficiency improvements as previously accessed data does not have the be fetched from the cloud more than once. This feature is available but is not fully supported yet. Please contact the developers if you want to test this feature.

Future releases will provide support for local data-files using the same toolbox that is used to access the portal. This is currently not implemented but users are encouraged to have a look at the SFR-Toolbox for Matlab which provides the foundation for this implementation (`http://jwagenaar.github.io/SFR-Toolbox/`).

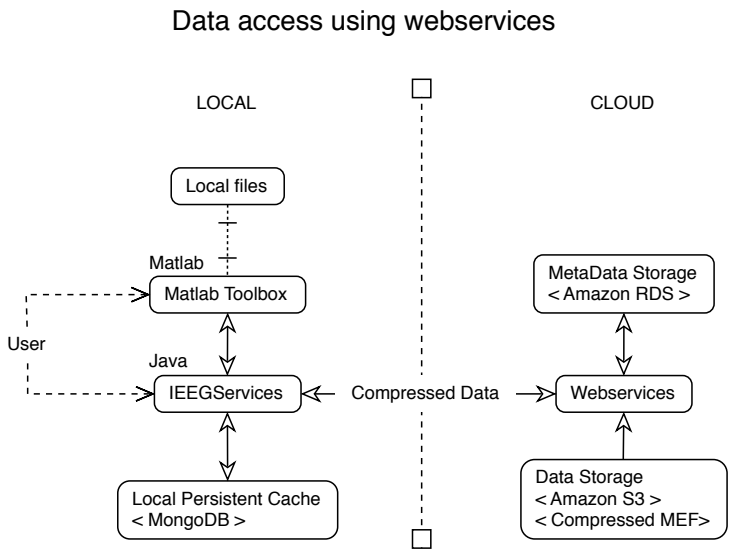

Figure 3.1: The IEEG-Services provide extensive tools to interact with the IEEG-Portal. Local file access from the IEEG-Toolbox for Matlab is currently not implemented, and the local persistent cache is in beta-testing.

4. The IEEG-Toolbox for Matlab

Download the IEEG-Toolbox for Matlab to interact with the portal from your local Matlab session, or from Matlab clients running in the cloud. The IEEG-Toolbox provides an intuitive way to connect to the IEEG-Portal with minimal installation effort. It provides methods to download data from the IEEG-Portal, view the data within a local Matlab session and upload results to the portal for subsequent sharing with collaborators.

The IEEGToolbox is developed to be used with a local Matlab session or with a cloud based Matlab Compiler Runtime (MCR) engine. It can easily be implemented in code that currently runs with local data. This section outlines the structure and functionality of the IEEG-Toolbox.

4.1 Requirements

The IEEGToolbox requires Matlab version 2012A or later. It also requires the Java Virtual Machine, so it will not work if you run Matlab with the "-nojava" flag.

4.2 Installing the IEEG Toolbox

Installing the IEEGToolbox is very easy. First, you need to add the IEEGToolbox folder to the MATLAB Path. Second, you need to create a password file that is associated with your username.

Authentication for the portal is verified using a username/password combination. In order to prevent the user from hard-coding them into an m-file and subsequently sharing them with everybody, we let the user store their password in a file which is used when the IEEGToolbox connects to the server. To create a password-file, you can use the static CREATEPWDFILE method of the IEEGSESSION class. This will create a file with a binary representation of your password. No encryption is used to store the password.

```
>> yourPath = IEEGSession.createPwdFile('userName','yourPassword');
-- -- IEEG password file saved -- --
>>
```

Advanced Installation (optional)

We highly recommend installing MongoDB to locally cache downloaded data. This step is optional but will vastly improve the user experience of the portal, and only takes a couple of minutes to set up. You can choose to install MongoDB on your local machine or a local server, which can be shared by multiple IEEG-Portal users.

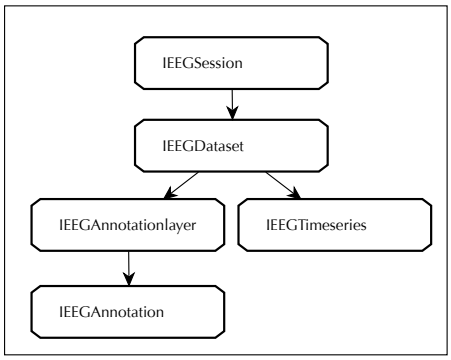

Figure 4.1: The IEEGToolbox classes and their relationships.

After MongoDB is installed and configured for the IEEG-Portal, data will automatically be stored in your local database once you access the data from the cloud using the web-services. The next time you access the same data, it will be fetched from your local MongoDB instead of Amazon's S3. This is significantly faster and reduces costs related to accessing data in the cloud.

Detailed instructions on how to install MongoDB and prepare it for use with the IEEG-Portal can be found on our Google-code site: `http://code.google.com/p/braintrust/wiki/MongoDBGuide`

4.3 IEEG-Toolbox classes

The IEEG-Toolbox is implemented using classes that have a strict hierarchical structure (see fig: 4.1). If you are not familiar with object oriented programming, please refer to the Matlab documentation for background information. Interaction with the IEEG-Portal is initiated when an object of class IEEGSession is created. A session is initiated using your username and password and the name of the dataset you want to access. Although a single dataset is associated with a session initially, additional datasets can be opened within a single session.

The classes in the IEEGToolbox are functionally organized according to the structure in Figure 1. This figure shows that an object of class IEEGSESSION can contain one or more objects of class IEEGDATASET, and an IEEGDATASET object can contain multiple objects of type IEEGANNOTATION and so on. You can navigate through the objects in a single session in a similar way that you would navigate through nested structures in Matlab.

You can view documentation on each of these classes by typing "help <ClassName>", or click on the links that are displayed in the Matlab console when you show an object.

```
>> help IEEGSession
 IEEGSession  IEEG-Portal class that contains session information.

  OBJ = IEEGSession('SnapshotName', 'userName', 'pwdFile') initiates
  an IEEG-Portal session and provides access to the snapshot with
  name 'SnapshotName'. The 'userName' and 'pwdFile' are used to
  authenticate the user on the portal. See CREATEPWDFILE method for
  instructions on how to create a password-file.

  OBJ = IEEGSession(... SERVER) allows the user to use a different server
  for access. Options include 'ieeg.org', 'qa', and 'local'. This option
  should only be used by IEEG-Portal developers.

  For example:
        out = IEEGSession('I001_P034_D01', 'userName', '/pwdfile.bin');

  see also: IEEGSession.createPwdFile
```

4.4 Interacting with the IEEGSession

The IEEGSession class serves as the starting point for communication with the IEEG-Portal. When you create an IEEGSession object, it will contact the server and verify your username and password and will provide the interface for the connection.

When you initiate a session, you will have to provide the name of the dataset that you want to access, your username, and the location of the password-file that you created (see above). Once initiated, a typical session object looks like:

```
>> session = IEEGSession('I001_P034_D01','your_username','pwdfile.bin')
session =

  IEEGSession:

      server: 'ieeg.org'
    userName: 'your_username'
        data: [1x1 IEEGDataset]

  Methods, ieeg.org
>>
```

Clicking on the ieeg,org link will open an external browser and the IEEG-Portal site. The Methods link will show you which methods can be used on the current object.

```
IEEGSession Methods:
  createPwdFile       Creates a pwd-file to be used with the IEEG Portal
  getSnapID           Provides snapshot ID for current Dataset.
  getTSI              Returns the Java TimeSeriesInterface object.
  IEEGSession         Creates an object of class IEEGSESSION.
  methods             Shows all methods associated with the object.
  openDataSet         Adds new dataset to current session.
  openPortalSite      Opens the portal in an external browser.
  up                  Returns parent object in hierarchy.
```

Clicking on the names of the methods will display the help-section for the individual method.

Note in the IEEGSession methods that you can also open additional IEEGDatasets within the same session using the "OpenDataSet" method. This will increase the number of Datasets in an IEEGSession.

Getting data

Once you have initiated an IEEGSession, you can browse to the IEEGDataset object with the dataset-name that you provided. Accessing objects within the session utilizes the same syntax as accessing properties in a Matlab structure. Getting values with object will require methods that slightly deviate from the default behavior of Matlab structures. Be sure to view the methods for each class in the IEEG-Toolbox to see what methods are available

```
>> session.data

ans =

  IEEGDataset:

    snapName: 'I001_P034_D01'
    channels: [1x47 IEEGTimeseries]
    annLayer: [1x1 IEEGAnnotationlayer]
      filter: 'No Filter'
    resample: 'Not resampled'
      values: [579016087x47 double]
  sampleRate: 5000

  Methods, ieeg.org
>>
```

The IEEGDataset object shows the name of the snapshot, an array of channels, an array of annotations layers and a "virtual" array of values. The values are not downloaded to the local machine at this point but are easily accessible using the GETVALUES method. Note in the following example that we do not directly access the property but in fact call a method to get the values. This allows us to get the values from the server on demand.

You can click on the "Methods" link to see additional methods available for object of the IEEGDataset class. For example, you can set the filter, and resample properties to automatically filter and resample the data when you fetch it from the cloud. The toolbox will take care of correctly padding the data to remove any filter boundary artifacts.

```
>> values = session.data.getvalues(1:10000, 1:47);
>> values(1:10,1:10)

ans =

     7    11    48    64   -89  -321  -111   -82   -59   -11
     6    11    49    64   -88  -320  -110   -79   -58   -13
     5    12    51    65   -87  -318  -108   -76   -56   -15
     4    12    53    66   -85  -315  -107   -74   -55   -17
     4    12    54    67   -83  -313  -106   -72   -54   -18
     5    13    55    68   -82  -311  -105   -70   -53   -20
     6    13    56    68   -81  -309  -105   -68   -52   -23
     6    13    58    69   -81  -307  -104   -65   -51   -26
     7    14    61    70   -80  -305  -103   -62   -50   -28
     8    15    63    72   -79  -303  -102   -60   -49   -29
>>
```

Note that class methods in Matlab have the object itself as the first argument of the method. The following two method-calls are identical when "session" is an object of class IEEGSession:

```
>> openDataSet(session, 'I001_P002_D01');
>> session.openDataSet('I001_P002_D01');
```

Getting annotations

The IEEGAnnotationlayer has three methods to getannotations from the portal: GETEVENTS, GETNEXTEVENTS, and GETPREVIOUSEVENTS. Please read the help section for each of these methods for detailed information about their specifics.

```
>> help IEEGAnnotationlayer.getEvents

  getEvents  Returns an array of events by time.

    ANNARRAY = getEvents(OBJ, STARTTIME) returns an
    array of IEEGANNOTATION objects following the STARTTIME in usec.
    The default maximum number of returned objects is used (1000
    objects). See alternative method calls to specify this number
    manually.

    ...
```

As you can see, the standard arguments for the getEvents method are the object and a start-time in microsecond. By default, the portal will return up to 250 annotations. If the dataset contains more annotations, the METHOD can be called again with a different start-time, or the user can choose to use the GETNEXTEVENTS method with the last returned annotation object as the input argument.

```
>> anns = out.data.annLayer.getEvents(0)
anns =

  1x20 IEEGAnnotation:

           type
    description
       isGlobal
    hasDuration
          start
           stop
       channels

  Methods, ieeg.org
>>
```

4.5 Creating annotations

The easiest way to create annotation objects is to use the static "createAnnotations" method of the IEEGAnnotation class. Using this method, you can quickly create an array of annotations at various time-points. The method accepts multiple input variations in order to create single-event annotations, dual-event annotations, on all or a limited set of channels.

```
>> help IEEGAnnotation.CreateAnnotations

CreateAnnotations  Factory to create multiple annotations.

  OBJS = createAnnotations(STARTTIMES, TYPEARR) returns an array of
  IEEGAnnotation objects. STARTTIMES is a vector of timestamps
  indicating the time at which the event takes place in usec. The
  startTimes are referenced to the beginning of the dataset, where
  t=0 is the first sample of the dataset. TYPEARR is a string or a
  cellarray of strings indicating the type of events that are
  created. When no channels are supplied, the annotation is
  considered a global annotation and includes all channels once added
  to a dataset.

  ...
```

When you create annotations, the annotation layer is not specified. Once you have created the annotations, you can add them to an annotation-layer using the ADD method. This

will add the annotations to the annotation-layer and upload the annotations to the portal so you can view the annotations in the webconsole EEG-Viewer.

```
>> testLayer = out.data.addAnnLayer('testLayer')

testLayer =

  IEEGAnnotationlayer:

     name: 'testLayer'
    evnts: [1x0 IEEGAnnotation]

  Methods, ieeg.org

>> anns = IEEGAnnotation.createAnnotations([1 2],  {'Event','Event'},...
  {'Marker 1' 'Marker 2'})

anns =

  1x2 IEEGAnnotation:

           type
    description
       isGlobal
    hasDuration
          start
           stop
       channels

  Methods, ieeg.org

>> testLayer.add(anns)

ans =

  IEEGAnnotationlayer:

     name: 'testLayer'
    evnts: [1x2 IEEGAnnotation]

  Methods, ieeg.org
```

4.6 Matlab client viewer

The IEEGToolbox for Matlab provides a simple EEG-Viewer that can be used to view EEG-Traces and annotations for a given IEEGDataset object. Each annotation layer can be

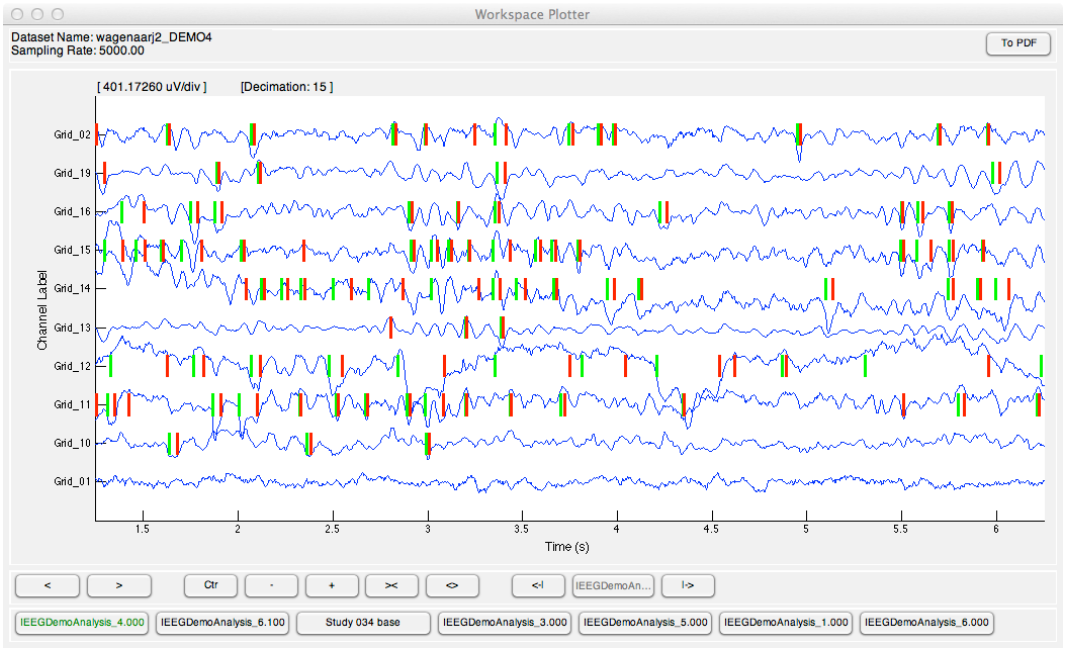

Figure 4.2: The Matlab viewer provides an easy way to page through EEG-data using the IEEGToolbox for Matlab. It supports multiple annotation layers and basic functionality to scale and scroll through the data.

activated independently and the user can jump between annotations of a single annotation layer. The viewer is implemented as a method of the IEEGDataset class and can be invoked as follows:

```
>> help IEEGDataset.viewer

viewer  GUI which displays raw data and events.
     viewer(OBJ, RANGE, INDECES) creates a GUI which shows the
     electrodes INDICES over the range RANGE=[first_index last_index]
     for the IEEGDATASET object.
     ...
```

You can specify the range and the channel indices that should be displayed in the viewer. You cannot change the channels once the viewer is opened, but you can increase, or decrease the range.

5. Example Data Analysis

In this tutorial, we walk through a simple demonstration where we analyze data and share the results with a collaborator using the IEEG-Toolbox. It is assumed that you have installed the toolbox at this time and created a password-file. You can type along during the tutorial or download the entire sourcecode from the portal we console (Toolname = IEEG-Tutorial).

5.1 Connecting to the portal and opening a dataset.

We will be using dataset "I001_P034_D01" as the starting dataset for this tutorial. If you look on the IEEG-Portal web console, you can see that this is a study with 47 channels and 20 seizure annotations. In Matlab, create a session object by copy-pasting the following command, replacing the parameters between brackets with your username and path to your password file.

```
>> session = IEEGSession( 'I001_P034_D01', <your_username>, <your_pwdfile>)
...
session =

    IEEGSession:

      server: 'ieeg.org'
    userName: <your_username>'
        data: [1x1 IEEGDataset]

    Methods, ieeg.org

>>
```

Session is an object that provides access to the portal data in the cloud. As you can see, the session currently has a single data set associated with it; I001_P034_D01. It is possible to open additional datasets in a single session. For example, you can open dataset "I001_P002_D01" by typing:

```
>> openDataSet(session, 'I001_P002_D01')
ans =
  IEEGSession:

      server: 'qa'
    userName: 'wagenaarj2'
        data: [1x2 IEEGDataset]

  Methods, ieeg.org

>>
```

Notice how the "data"-property now is a 1x2 array of IEEGDataset objects. Open the data property to inspect both objects:

```
>> dataset_1 = session.data(1);
>> dataset_2 = session.data(2)

dataset_2 =

IEEGDataset:

    snapName: 'I001_P002_D01'
    channels: [1x15 IEEGTimeseries]
    annLayer: [1x1 IEEGAnnotationlayer]
      values: [2318257553x15 double]

  Methods, ieeg.org

>>
```

The variables 'dataset_1' and 'dataset_2' are references to the objects stored in the data property of the IEEGSession object. This means that any changes to 'dataset_1' also change the session.data(1) object (IEEG objects are handles). Please refer to the Matlab manual to read more about handle objects if you want to know more about this.

The variable 'dataset_1' refers to the 'I001_P034_D01' dataset which is Read-Only. In order to upload results to the portal, we need to derive a new dataset reference from the original which we can modify. This does not duplicate the data in the cloud, but only creates a new pointer to the raw data that can be associated with your results.

Let's create a new dataset with 4 channels based on the 'I001_P034_D01' dataset:

```
>> inclCh = dataset_1.channels([1 3 5 7]);
>> myDataset = dataset_1.deriveDataset([session.userName ...
    '_myDataset'], inclCh)
myDataset =

IEEGDataset:

    snapName: '<userName>_myDataset'
    channels: [1x4 IEEGTimeseries]
    annLayer: [1x1 IEEGAnnotationlayer]
      values: [287101314x4 double]

   Methods, ieeg.org
>>
```

Notice that we used a combination of the userName and '_mySnapshot' as the name of the new snapshot. We do this because all datasets should have a unique name. If you provide a name that already exists on the IEEG-Portal, you will not be able to create the dataset. Prepending your username provides an easy way to create a unique name.

5.2 Showing data and annotations

Look at the contents of "myDataset". You can see that the object contains 4 channels and that it has a single annotation-layer. We also see that the data has a very large array of values. To access these values, we use the GETVALUES method. We cannot directly access the values property because the values need to be downloaded from the server first.

```
>> someData = myDataset.getvalues(1:10000, 1:4);
>> size(someData)

ans =
       10000           4

>>
```

The variable "someData" now contains a local copy of the first 10000 values on all channels of this dataset. Similarly, we can get the annotations. They are stored in the "annLayer" property.

```
myAnnotations = myDataset.annLayer(1).getEvents(0)
```

When you check the methods of the IEEGAnnotationlayer class, you can see that there are multiple methods available that retrieve annotations from the server. Depending on your analysis, the other methods might provide easier access to multiple request of annotations.

5.3 Detecting events

In this section, we are going to use one of the tools that you can download from the portal to demonstrate how easy it is to use the IEEG-Toolbox for doing some real analysis. Please download the "IEEGDemoAnalysis" tool from the Tools page in the portal by clicking on the "Download" link. Extract the downloaded zip file, rename the m-file to "IEEGDemoAnalysis.m", and place the m-file in a location where Matlab can run it (on the Matlab path).

The tool contains three methods:

```
function dataset = IEEGDemoAnalysis(dataset, channels, threshold)
    ...
function annotations = detectEvents(values, startTime, sf, threshold, ...
    channels ,init)
    ...
function [runMean, runVar] = runningVariance(newData, init)
    ...
```

The first method is the main method that we will be calling later to run the analysis. This method creates a new annotation layer and iterates over the number of data-blocks that we will be processing. During each iteration, it calls the second method of the tool to detect the events we are interested in. The first method then adds these events to the dataset, which puts the annotations on the portal.

The second method processes each data-block and extracts the event-times for which the values per channel cross a predefined threshold. In this demo-tool, the threshold is calculated by

$$Th = \mu + n \times \sigma, \tag{5.1}$$

where μ is the mean of the channel values, σ is the standard deviation of the values for that channel. and n is a multiplier provided by the user. The method then combines any events that are separated by less than 0.075 seconds. This number is chosen arbitrarily. Finally, it creates an array on annotation objects which are passed back to the first method.

Because we are fetching the data in multiple blocks, we do not know the mean and standard deviation of the entire dataset during the analysis. The third method in the tool updates the mean and standard deviation of the signal every time new data is presented. The mean and standard deviation should stabilize quickly as more data-blocks are fetched. An improvement to this method could include a running buffer instead of including all data in the mean and standard deviation. Please read through the source-code of the three methods

to see how the IEEG-Toolbox is used to fetch data, and to add annotations to the dataset object.

Let's run the tool on our derived dataset with a multiplier of 2. This means that any time the values exceed the mean plus two times the standard deviation, the data is annotated.

```
>> IEEGDemoAnalysis(myDataset,1:4, 2)
Creating 015 annotations on channel 01
Creating 009 annotations on channel 02
Creating 005 annotations on channel 03
Creating 005 annotations on channel 04
Creating 011 annotations on channel 01
Creating 007 annotations on channel 02
Creating 002 annotations on channel 04
Creating 016 annotations on channel 01
Creating 003 annotations on channel 02
...

ans =

  IEEGDataset:

    snapName: '<userName>_myDataset'
    channels: [1x4 IEEGTimeseries]
    annLayer: [1x2 IEEGAnnotationlayer]
      values: [579016087x4 double]

  Methods, ieeg.org
```

You can see that the dataset now contains two annotation layers and if you browse to the second annotation layer, you should see it contains 212 annotations. A good way to get a quick view of the returned annotations is to view the data using the Matlab viewer (Fig. 5.1). You can also see the annotations in the portal by opening the derived dataset and select the appropriate layer in the EEG-Viewer in the portal.

The generated annotations are also visible in the portal and ready to be shared with other IEEG-Portal users. Open the IEEG-Portal in your web-browser and go to the "Datasets" panel. Now, click on "OtheDataset" and provide the name of your newly derived Dataset. This will open the dataset in the EEG-Viewer (Fig. 5.2) panel and should show you the new events (You might have to click on the "Layers" button in the bottom of the screen and select the new annotation layer before the annotations show up).

This dataset is currently only visible to you. To share this dataset with other users, click on the "Share" button in the top-right of the webconsole. Now, find the user "wagenaarj2" and add this user as a collaborator with read-only privileges. Congrats, you just shared your first results on the IEEG-Portal.

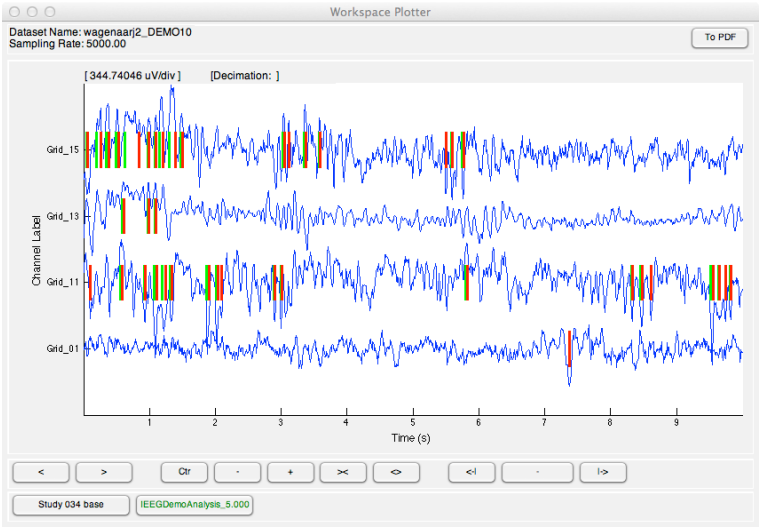

Figure 5.1: The results of the event-detection algorithm displayed in the Matlab IEEG-Viewer. It can be seen that the algorithm detects events where the data surpasses the mean plus five times the standard deviation.

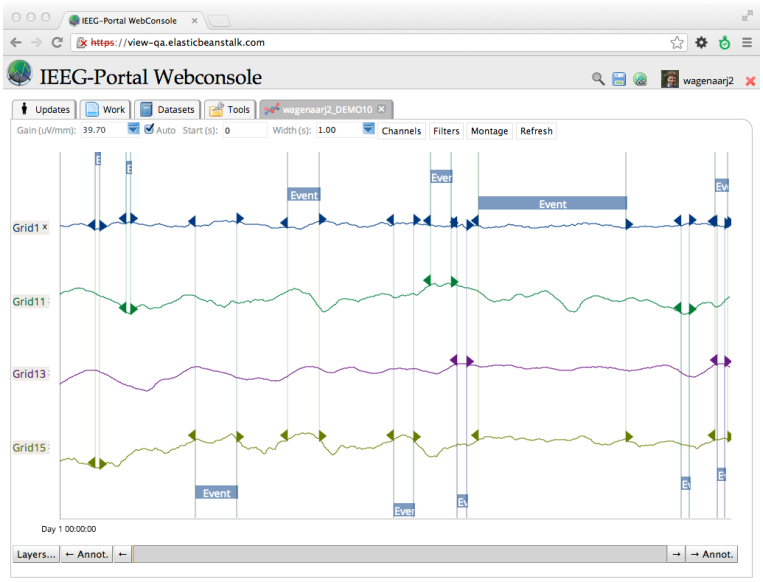

Figure 5.2: The results of the event-detection algorithm displayed in the Matlab IEEG-Viewer. It can be seen that the algorithm detects events where the data surpasses the mean plus five times the standard deviation.

6. Supporting the IEEG-Portal

The best way to support the portal is to use it and provide feedback about the features that you would like to have improved. We found that speed and ease-of-use are the most important requirements to most users and we have been working nonstop to improve the portal to fulfill those needs.

The portal is currently focussed on Epilepsy data but we will be expanding this over the next year. We are working on including ICU monitoring data, and collaborating with multiple sites to include animal datasets. In addition, we are working on improving support for imaging and cross-platform integration with other scientific portals.

If you are interested in working on a project for the portal, or if you have suggestions/feedback about the portal, please contact us at: ieegportal@gmail.com.

7. Terminology

- **Annotation:** An annotation marks an event in the data. The event has a start and stop time and an associated type. The type indicates what kind of event occurred. A single annotation can span all, or a subset of channels in a dataset.
- **Dataset:** A set of time-series data and its associated meta-data. The IEEG-Portal contains static datasets which are associated with uploaded data, and mutable user-defined datasets which can be constructed from static datasets or other user-defined datasets.
- **Snapshot:** The term "Snapshot" was used in the portal instead of "Dataset" but is being phased out as it conflicts with the perceived immutable nature of the term.
- **Webservice:** The method of communication between the third party tool and the IEEG-Portal.
- **Tool:** A tool refers to a specific method (and its dependent methods) that is registered with the IEEG portal. Each tool is assigned a Tool Identification Number (ToolID), which is used by the IEEGToolbox to identify the analysis that is performed on the data. The ToolID should be defined as a constant in the m-file that is associated with the specific tool.
- **Snapshot Permissions:** There are three permission levels for a snapshot; Read-only, Edit, Owner.