TEAPhysio documentation

Table of Contents

Introduction to the toolbox

How to load TEAPhysio

In order to load TEAPhysio, you have two possibilities, but they all use the same principle

TEAPhysio for multiple small projects

If you want to use TEAP for multiple small projects, the best way is to extract TEAP somewhere and add-it to your matlab's (or octave's) classpath. This way, TEAP's functions will always be accessible.

TEAPhysio for a standalone project

If you want to build a project that uses TEAP and that should work everywhere, the best way is to copy TEAPhysio in your project's folder. Then, you'll simply have to call the `init.m` scrip to load TEAP.

Signal data structure of the TEAP toolbox

Description

Signals are organised in a OOP sort of way: each signal (ex. Galvanic Skin Response) has their own folder (in our case: `GSR`). Once in this folder, the functions are ordered by folder (eg: *acquisition*, *examples*, *features*, *tests*). For example, if you want to capture a Galvanic Skin Response signal, you would have to call a function in the `GSR/acquisition` folder (for ex: `GSR_aqn_file()`).

OOP structure

Signals of the TEAP toolbox are in fact structures. The signal has the following attributes embedded:

- `TEAPhysio`: the signal type. Will always be **S** for a signal.

- samprate: the sampling rate of the signal.
- raw: the raw data of the signal. It's a 1D horizontal vector.
- unit: the unit of the signal (ex: *Ohm*).
- name: the signal's name (ex: *GSR*)
- isAbsolute: indicates if the signal was baselined/relatived.
- preprocessing: a vector of pre-processing features (see below).

You should **always** access to the signal's attributes via the signal's functions. These functions are fail-safed, and guarantee a semantic coherence across the variables: you can't call GSR_feat_median(Sig1) when Sig1 is of type ECG.

pre-processing structure

This structure indicates the possible pre-processing features that the signal has. Actually, only lowpass = 1 is sometimes used.

Complete example

Here is a little example of a complete signal showing features and pre-processing markers: GSRsig =

```
 scalar structure containing the fields:

   TEAPhysio = S
   samprate =  512
   raw = SNIP
   unit = Ohm
   name = GSR
   isAbsolute = 0
   preprocessing =

     scalar structure containing the fields:

       lowpass =  1
```

How to use signals in TEAPhysio

We will now see how to use these signals in a working environnement.

BULK SIGNALS

What are Bulk Signals ?

Sometimes, when you research something, you may have to record many channels (EEG, GSR, etc.) for each experiment. It would be fine to have a variable per channel, but it would be cumbersome if you had 10 participants. One way would be to deal with a cell array or something, but that's work you shouldn't have to do.

Bulk Signals are simply structures with all the channels embedded within. Let's say you want to import a recording of a participant (loaded from a .mat file).
You only have to type something like:

```
load Participant01_t01.mat
P01t01 = Bulk_load_mat(REC);
```

and the variable P01t01 would then be a bulk signal.

If you are curious about how a Bulk signal works: it's simply a structure. Each signal has it's own field (the **GSR** signal is embedded in the .GSR field, etc.). There is also an extra 'validity-check' field, named .TEAPhysio, which consists of a single value: **'B'**, to indicate that this variable is a Bulk signal.

How to use bulk signals, you would ask ?

Well, we think it's fairly simple. Let's take an example. Imagine that you have a **GSR** signal and you want to calculate it's peaks. With this single signal, you only have to write: GSR_feat_peaks(GSRSig).

Now, as you have to deal with multiple signals, you have a bulk signal (let it be named aBS (for 'a Bulk Signal', or 'Anti â€¦' if you are humorous). You would think you would have to type GSR_feat_peaks(aBS.GSR) (that's a BAD practice, see down below why) to calculate the feature, and it would work, but TEAP is clever: the function knows that it needs a **GSR** signal. So if you give it a Bulk signal, it will search inside it, and, if a **GSR** signal is embedded within, will actually take-it. So GSR_feat_peaks(aBS) will work too.

PROGRAMMING GOOD PRACTICES

Taking an embedded signal

You could use `Bulk.SIG`, but it's actually a bad practice. Use instead this simple function:

```
Signal = Bulk_get_signal(BulkSig, SSS_get_signame())
```

`SSS_get_signame()` gives the name of a signal (ex: `GSR_get_signame()` gives back **'GSR'**).

Getting the list of the signals of a bulk signal

Well, the function `Bulk_get_signals()` gives a list of all the signals inside this bulk signal (for you techies, it lists the fields, and takes back the `TEAPhysio` control field.

Adding a signal to a bulk signal

If you want to add a signal to a bulk signal, you may want to do:

```
Bulk.SSS = SSSsig;
```

but that is bad :(. Use:

```
Bulk = Bulk_add_signal(Bulk, SSS_get_name(), SSSsig);
```

instead ;)