**DIN\_Translate**

Funkcja interpretuje znacznie DINów (według zasad, które można jej podać) i update'uje eventy wybranych eeglabowych plików (zmieniając nazwy i dodając pola).

Dla danych zawierających wiele eventów typu 'DIN4', 'DI32' (ale też 'BIMBOM64') funkcje:

**1.** update'ują strukturę EEG - zamieniają DINy występujące w tym samym czasie na **pojedyncze eventy**

**2.** nowe eventy mają dodatkowe pola:

**.type2** z binarną reprezentacją DINów, z których utworzony został event;

**.type10** z wartością pola .type2 w systemie dziesiętnym

**3.** na podstawie pola .type2 oraz macierzy zasad tłumaczenia (o której więcej dalej) tworzone są:

     a)  **nowe nazwy eventów** (np. 'Target\_Easy\_5')

     b)  **nowe pola w strukturze event** z odpowiednimi wartościami (np. pole '.stimtype' o wartości 4)

**===Poniżej więcej informacji (dokumentacja) dla zainteresowanych===**

**Aby zupdate'ować strukturę EEG jednej wczytanej już osoby**

**(np. aby przetestować czy wszystko nam się dobrze tłumaczy) piszemy:**

EEG = **DIN\_Translate**(**Transl**, **EEG**);

gdzie:

**EEG** - struktura EEG z EEGLABa

**Transl** - macierz zasad tłumaczenia (opisana dalej)

**Możemy też zupdate'ować masowo wiele plików, piszemy wtedy:**

EEG = **DIN\_Translate**(Transl, **'mass'**);

wyskoczy nam wtedy okienko, w którym wybieramy pliki do zupdate'owania

otworzy się eeglab i kolejno będą updateowane wybrane przez nas pliki.

**Można też update'ować masowo wszystkie pliki z jakiegoś folderu bez wyboru plików okienkiem, pisząc:**

EEG = **DIN\_Translate**(Transl, 'mass', **'folder'**, 'C:\BADANIA\2012\Telekineza\GrupaA\');

**ZASADY TŁUMACZENIA:**

Jeżeli markery binarne były wysyłane z sensem (tzn. można z nich wydobyć znaczenie markera)

to możemy od razu przekodowywać eventy funkcją DIN\_Translate (patrz punkt 3 na początku).

**=> Co to znaczy, że markery są sensowne?**

Znaczy to, że program nadzorujący eksperyment wysyłał markery powtarzalnie zgodnie z pewnymi zasadami

np.: w markerze 1011 pierwszy bit od lewej zawsze opisuje obecność/nieobecność dystraktora; kolejne dwa bity

typ bodźca ('01' --> ryba; '10' --> twarz; '11' --> przerażający dźwięk); natomiast ostatni bit daje znać o tym

czy osoba badana była równocześnie rażona prądem.

Jeżeli możemy w podobny sposób jak powyżej opisać zasady wysyłania DINów, tworzymy macierz

zasad tłumaczenia nazywaną wcześniej Transl (zasady jej tworzenia parę linijek niżej).

Jeżeli nie ma takich zasad definiujemy Transl jako:

Transl = [];

**=> Jak stworzyć macierz Transl?**

Macierz tłumacząca (nazywamy ją tu **Transl**) to **macierz komórkowa**

Tworzenie tej macierzy nie jest trudne, pomaga też odświeżyć deklarowanie

macierzy komórkowych komendami matlabowymi.

Pamiętać jednak warto, że matlab jest na tyle przyjazny, że można macierz tworzyć klikając w odpowiednie pola i

wpisując w nie wartości, nie trzeba wszystkiego robić z pola komend.

Każda jej **kolumna** macierzy **Transl** definiuje pewną tłumaczoną **własność** (*które bity co znaczą*)

i powinna spełniać taką zależność:

**pierwszy rząd**: liczba bitów opisujących tę własność

**drugi rząd**: nazwa własności (nazwa pola, które zostanie utworzone w evencie)

**trzeci rząd**: opis wartości, jakie obiera własność

(przykłady macierzy tłumaczenia w folderze ‘Translation matrices’)

Jeżeli nie brzmi to na razie zrozumiale, przyda się **przykład**:

**Przykład**

mamy 5 bitów definiujących event:

na przykład: 10110

pierwszy bit od lewej to u nas zawsze info o tym (na przykład)

czy marker oznacza rozpoczęcie czy zakończenie prezentowanego dźwięku

dlatego do macierzy Transl wpisujemy:

**Transl{1,1} = 1;**  % jeden bit definiuje tę własność

**Transl{2,1} = 'prepost';**  % tak nazwiemy tę własność i tak nazywać się będzie dodatkowe pole w eventach

jeden bit daje dwie możliwe wartości:

0, 1

więc w ostatnim rzędzie obecnej kolumny

definiujemy jak tłumaczyć te wartości na nasze własności.

możemy to zrobić na kilka sposobów, ale zrobimy

to w najbogatszy z nich :)

W ostatnim rzędzie naszej macierzy **Transl**, w pierwszej kolumnie (bo opisujemy pierwszą własność)

wrzucimy macierz komórkową informującą jak rekodować wartość DINów definiujących własność na znaczenie.

W naszym przypadku pierwsza kolumna tej macierzy zawiera informację o wartości binarnej, którą rekodujemy

druga kolumna - na jaką wartość rekodujemy, czyli co zostanie wpisane do dodatkowego pola eventu

trzecia kolumna - (nie trzeba jej tworzyć) informuje o tym jak zmieniać nazwę eventu w zależności od wartości własności

**Transl{3,1}(1,1:3) = {0, 1, 'Pre\_'};** % wartość binarną zero rekoduj na wartość 1 pola 'prepost' oraz oznaczaj w nazwie eventu dodając fragment 'Pre\_'

**Transl{3,1}(2,1:3) = {1, 2, 'Post\_'};** % wartość binarną jeden rekoduj na wartość 2 pola 'prepost' oraz oznaczaj w nazwie eventu dodając fragment 'Post\_'

Powyższe definicje znaczą tyle, że marker zaczynający się jedynką (w swej wartości binarnej) będzie:

- miał **nazwę** zaczynającą się na 'Post\_', na przykład: 'Post\_Info\_Easy\_2'

- event ten będzie miał **pole** 'prepost' z wartością 2 (nazwę pola definiowaliśmy w drugim rzędzie macierzy Transl)

czyli (np.) EEG.event(5).prepost będzie wynosiło 2

W ten sam sposób definiujemy kolejne bity (po kolei od lewej)

....

Załóżmy jednak, że ostatnie dwa bity (z pięciu w naszym przykładzie)

definiują u nas **jedną własność**

Piszemy:

**Transl{1,4} = 2;**  % dwa bity definiują tę własność (numer kolumny to numer własności)

**Transl{2,4} = 'stimtype'** % tak nazwiemy tę własność, a zarazem dodatkowe pole eventu

(jeżeli nie chcemy, aby tworzone było dodatkowe pole eventu - wpisujemy tu: **''**, czyli pusty wektor znaków)

Dwa bity to cztery możliwe wartości:

**0** (00), **1** (01), **2** (10), **3** (11)

przy czym załóżmy, że ustaliliśmy, że interesują

nas tylko wartości **1 - 3**

Definiujemy to tak:

**Transl{3,4}(1,1:3) = {1, 1, '1'};** % wartość jeden rekoduj na wartość pola jeden, a w nazwie opisuj jako '1'

**Transl{3,4}(2,1:3) = {2, 2, '2'};**% itd.

**Transl{3,4}(3,1:3) = {3, 3, '3'};**

Marker o dwóch ostatnich bitach **10**

będzie więc:

- miał **nazwę** kończącą się na '2', na przykład: 'Post\_Info\_Easy\_2'

- event ten będzie miał **pole** 'stimtype' z wartością 2

czyli (np.) EEG.event(5).stimtype będzie wynosiło 2

W folderze ‘Translation matrices’ są przykładowe macierze tłumaczeń

z których korzysta skrypt DIN\_Translate

(można otworzyć sobie w matlabie i posymulować tłumaczenie)

**==> Symulowanie tłumaczenia**

Możecie poeksperymentować z tłumaczeniem binarnym:

Jeżeli do funkcji DIN\_Translate wrzucimy, oprócz macierzy tłumaczącej

wektor zer i jedynek, potraktuje ją ona jako binarną wartość markera

i zasymuluje efekt przetłumaczenia takiego markera na event.

(można też wrzucić macierz zer i jedynek gdzie każdy kolejny

rząd jest kolejnym markerem).

Przydaje się to, gdy chcemy sprawdzić, czy dobrze zbudowaliśmy

macierz tłumaczącą.

Na przykład

otwieramy w matlabie obecny w folderze ‘Translation matrices’ plik **trsl\_in**

i (zakładając, że mamy potrzebne funkcje dodane do ścieżki dostępu)

piszemy:

>> **EEG = DIN\_Translate(trsl\_in, [1,0,0,1,1]);**

W ten sposób dostajemy zmienną **EEG**, strukturę zawierającą

zasymulowane eventy i ureventy.

Sprawdźmy jak został przetłumaczony nasz marker

>> EEG.event.type

ans =

Post\_Info\_Easy\_3

>> EEG.event.prepost

ans =

     2

>> EEG.event.difficulty

ans =

     1

>> output.event.stimtype

ans =

     3

**==> dodatkowe info**

Pomijanie bitów

W tłumaczeniu markerów binarnych na eeglabowe eventy, możemy pomijać

określone bity, wystarczy, że zrobimy z nich własność, która będzie miała

wyłącznie informację o liczbie definiujących ją bitów, np.:

jeżeli chcemy pominąć pierwsze 3 bity od lewej tworzymy Transl tak:

**Transl{1,1} = 3;**

**Transl{2,1} = [];**

**Transl{3,1} = [];**

Nietworzenie dodatkowego pola w evencie dla własności

Jeżeli nie chcemy, aby tworzone było dodatkowe pole w eventach wpisujemy w

**drugim wierszu** danej własności (kolumny) **''**, czyli pusty wektor znaków

**Transl{2,1} = '';**

======