Voilà Michael,

J’ai terminé de développé le module ICA pour ta toolbox EEGpal.

J’ai créé une branche ICA sur ton repository GitHub. Je te propose de fusionné cette branche avec la branche main. Je te laisse regarder pour valider les changement.

Voici une liste des modifications que j’ai apportées à la toolbox :

# Logbook des modification apporté à EEGpal du 26.12.2023-01.05.2024

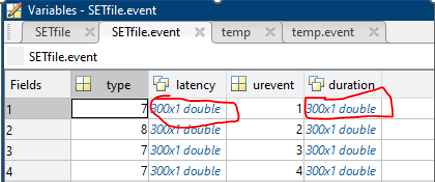
**1. Integration de EEGlab complet dans la toolbox**

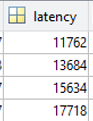
Pour faire de l’ICA, j’ai été obligé d’intégrer le code d’eeglab2023.1 complet dans le répertoire .\eeglab\_plugins. En effet, il était plus simple et plus safe de travailler comme ça car le traitement ICA nécessite beaucoup et sous-fonctions qu’il était difficile à identifier.   
La version eeglab intégrer ne contient ni *cleanline*, ni *clean\_rawdata* pour être sûr que ce sont les version intégrer par MichaelDP qui sont utilisés par le module de Filtring+.

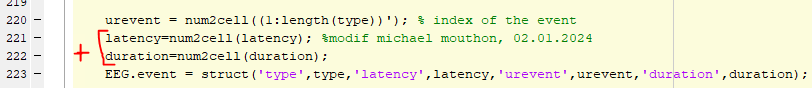
Dans le but d’avoir une situation propre j’ai supprimé les répertoires ou fichier suivant du répertoire .\eeglab\_plugins pour éviter les doublons par rapport au nouveau répertoire .\eeglab\_plugins\eeglab2023.1  :

* .\adminfunc (setdiff\_bc.m)
* .\ eeglab\_guifunc (finputcheck.m ; icadefs.m ; inputgui.m ; pophelp.m ; supergui.m)
* .\popfunc (eeg\_decodechan.m ; eeg\_eegrej.m ; eeg\_lat2point.m ; eeg\_point2lat.m ; pop\_select.m)

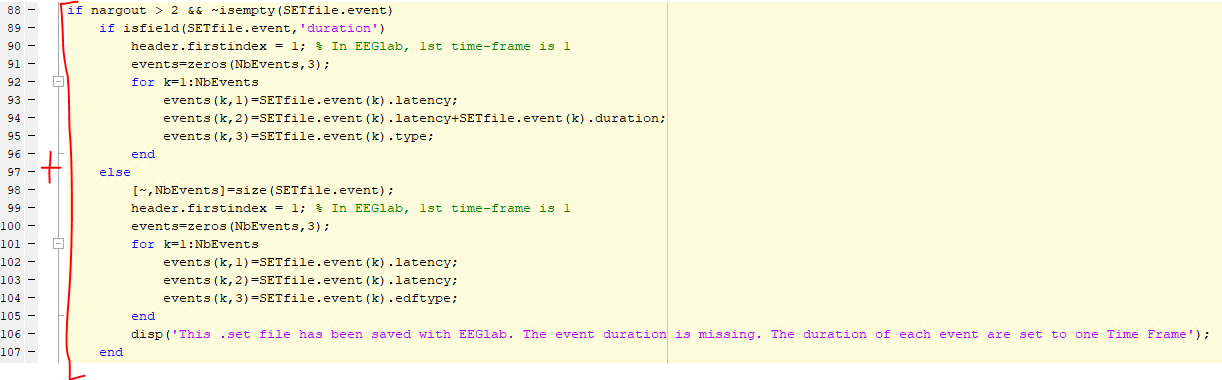
**2. Modification des script save\_eeglab.m et open\_eeglab.m pour mieux gérer les events**

J’ai corrigé une erreur de programmation dans le script *save\_eeglab.m* qui sauvait *EEG.event.lateny* et *EEG.event.duration* sous forme de celle comme on peut le voir sur cette image :   


J’ai modifié pour que ces variables soit sous forme array  


save\_eeglab.m

Du coup, j’ai adapté *open\_eeglab.m* en conséquence en ajoutant la fonctionnalité de lecture pour des fichiers qui ont été enregistré avec EEGLAB (par de variable EEG.event.duration => considéré que c’est 0 par défaut).



**3. Modification de tous les modules existant**

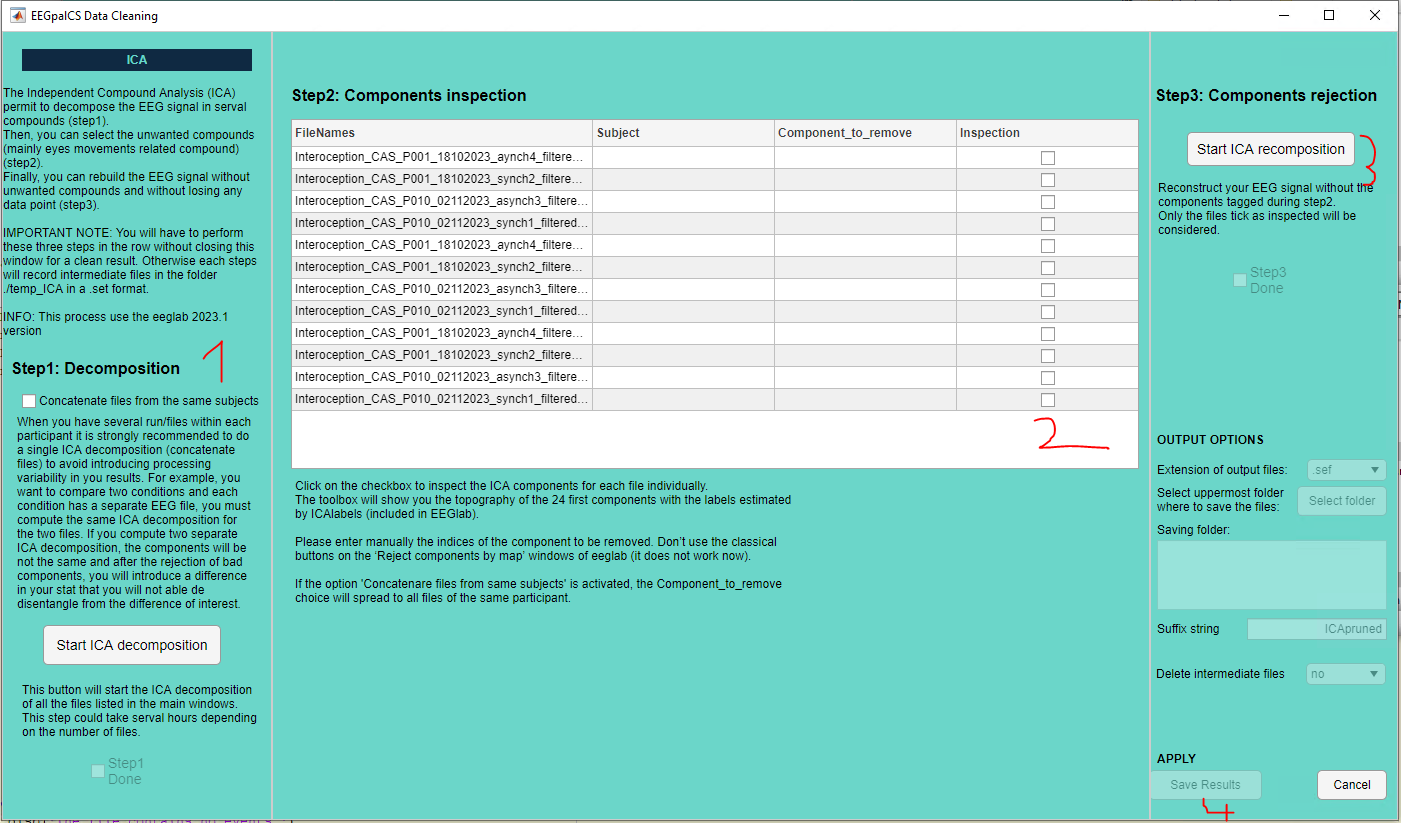
Modification de la lecture/écriture de fichier .set dans tous les modules pour prendre en considérations les modifications apportées au point 2

+ Modification de EEGpalCS\_main\_Data.mlapp pour obliger l’utilisateur à entré un coordinate file avant de démarrer le module Filtering+ (j’ai constaté que dans le cas contraire, il y a un bug)

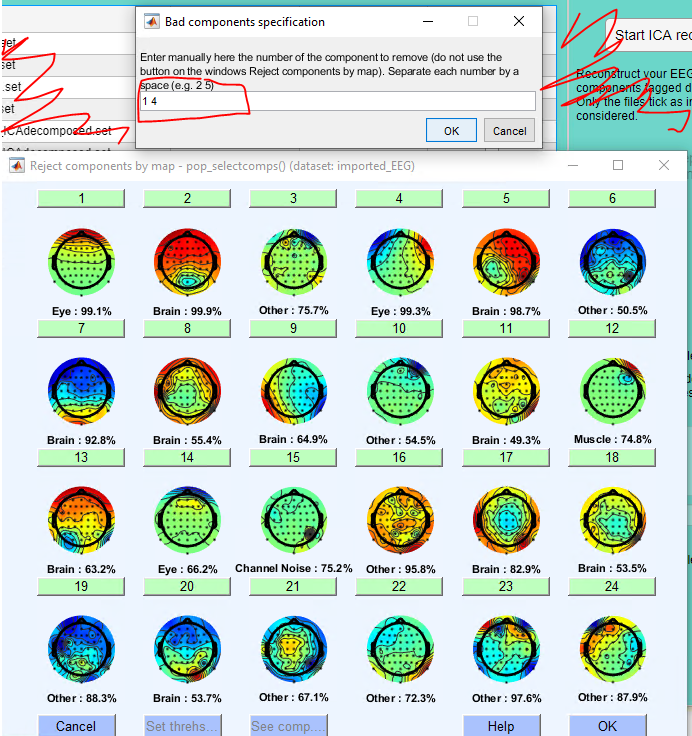
**4. Création du module EEGpalCS\_mod\_ICA.mlapp**

L’appel de ce module nécessite deux pre-requis :

* Un fichier de coordonné xyz dans ‘Electrode Setting’ soit rentré
* Les fichiers input soit obligatoirement en format .set ; Comme je travaille avec eeglab, il est impératif de travailler avec des fichiers en format .set en input. Le module permettra d’exporter les résultats en n’importe quel format (.sef ; .ep ; .eph ; .set)



Le module est composé de 4 partie qu’il faudra exécuter manuellement l’une après l’autre. Les résultats pout les étape 1 et 3 sont sauvegarder dans des fichiers temporaire en .set qui sont enregistré de le répertoire .\ICAtemp dans le répertoire input

1. D’abord la décomposition en composante (c’est la partie la plus longue). Il va utiliser la fonction pop\_runica de Matlab. J’ai ajouté une fonctionnalité supplémentaire : la concaténation de fichier pour un même sujet. Lorsque l’utilisateur aura spécifié les sujets (grâce à ton interface principale très bien faite)+option ‘concatenate files from the same subject’, les fichiers d’un même sujet seront concaténé automatiquement avant décomposition puis le résultat ICA de cette décomposition sera sauvegardé dans chaque fichier individuellement. Sinon la décomposition sera faite pour chaque fichier de manière séparée.   
   Après avoir pressez sur le bouton ‘Start ICA decomposition’, il sera nécessaire de spécifier l’emplacement du fichier des coordonnées des électrodes \*.locs créé par eeglab qui correspond au fichier xyz. En effet, j’ai des problème avec la visualisation des topographie à l’étape 2 (topo tourné de 90° en utilisant les coordonné founi dans le fichier xyz). Le remplacement des coordonné est temporaire et les coordonné xyz seront restauré à l’épape3.   
   Les fichier sont sauvé dans le répertoires .\ICAtemp\decomposed
2. La matrice montre la liste des fichiers en cours. Elle contient d’abord le nom des fichier originaux puis est mis-à-jour après exécution de l’étape 1.   
   L’utilisateur doit cliquer sur la checkbox pour chaque fichier ce qui va automatiquement affiché la topographie des 24 permières composante ICA avec le résultat du classificateur ICALabel inclus par défaut dans eeglab2023.   
    

Il faut entré manuellement le vecteur des composantes à supprimé dans la fenêtre supplémentaire ‘Bad components specification’. Malheureusement, je n’ai pas réussi à utiliser les boutons pour affiché le powerspectrum des composantes (mais c’est pas un drama car pas nécessaire pour sélectionné les composantes lié aux eye-blink).

Une fois valider, les composantes seront stocké dans la table sous la colonne Component\_to\_remove. Le tick dans la colonne Inspection permet de valider le fait qu’il y a eu inspection pour ce fichier.

Dans le cas particulier où l’option ‘Concatenate files from the same subjects’ est activé, alors le choix des composantes et le tick de validation sera automatiquement appliqué à tous les fichier pour le même participant

1. Le bouton ‘Start ICA recomposition’ permet de supprimer les composantes ICA spécifier dans le tableau de l’étape 2. Seul les fichiers qui ont le tick dans la colonne ‘Inspection’ seront considéré (les autres fichiers seront ignorer). Le résultat intermédiaire sera sauvé dans le répertoire .\ICAtemp\recomposed sous un format .set
2. Le bouton ‘Save Results’ (disponible seulement après l’étape 3) permet d’exporter le résultat de l’étape 3 dans le répertoire sélectionné et au format désirée. Le paramètre ‘Delete intermediate files’ permet de supprimer ou non le répertoire .\ICAtemp qui contient les fichier temporaire

Il est conseillé de faire les 4 étapes sans fermé le module. Néanmoins, il est possible de faire seulement les étape 2-4 (sans l’étape 1) si la décomposition a été fait préalablement. Il faut pour ça mettre en input les fichiers temporaires de decomposition situé dans .\ICAtemp\decomposed\\*.decomposed.set

Ce développement a nécessité l’ajout de EEGlab complet dans le répertoire eeglab\_plugins. Les script doulons on été retirer

**5. Création du module EEGpalCS\_mod\_File\_Cut.mlapp**

Module qui permet de couper et/ou concatené les fichier EEG basé sur leurs triggers.

A screenshot of a computer

Description automatically generated

**6. Ajout de la détection des ponts dans le module interpolation**

Création d’un bouton « Bridge detection » dans le module d’interpolation. Celui-ci va appelé eBridge (script de EEGLAB) et va automatiquement remplir le tableau interpolation avec les bridge qu’il a détecté. Ceci est compatible avec les détection de mauvaise électrodes de CleanRawData.

**7. Création de la fonction export\_in\_set.m pour convertir tout type de fichier en .set**

L’utilisation de module EEGLAB dans le ICA et la détection des ponts oblige à travailler avec des fichiers .set.

Pour simplifier la vie de l’utilisateur, j’ai créé le script export\_in\_set.m qui fait automatiquement cette conversion. Le code est basé sur celui de MichaelDP dans le module Export.

Il est utiliser pour la detection de bridge dans l’interpolation ainsi que le module ICA

**8. Création d’une méthode global dans la fenêtre principale pour sauvegarder le path de la toolbox : app.AppPath**

J’ai ajouté un répertoire *Resources* dans la toolbox pour y mettre les fichiers de coordonné EEG.   
Grâce à l’ajout de la variable/method global app.AppPath.value, les ui qui demande d’entré les fichier des coordonnés pointrons par défaut dans ce répertoire (module EEGcoordinate + ICA)

Idée pour prochaine amélioration :

Améliorer le module Frequencing afin qu’il enregistre automatiquement un fichier Excel avec la moyenne des tous les fichiers au lieu d’avoir un output par fichier.