ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΣΥΣΤΗΜΑΤΩΝ ΜΕΤΑΔΟΣΗΣ ΠΛΗΡΟΦΟΡΙΑΣ
ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ ΥΛΙΚΩΝ

ΕΝΙΣΧΥΣΗ ΔΥΝΑΤΟΤΗΤΩΝ OPEN SOURCE ΕΡΓΑΛΕΙΟΥ ΑΠΕΙΚΟΝΙΣΗΣ ΝΕΥΡΟΦΥΣΙΟΛΟΓΙΚΩΝ ΔΕΔΟΜΕΝΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Ελευθέριος Πέτρου Γεωργοπετρέας

Επιβλέπων : Δημήτριος Ασκούνης

Αθήνα, Φεβρουάριος 2024

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΣΥΣΤΗΜΑΤΩΝ ΜΕΤΑΔΟΣΗΣ ΠΛΗΡΟΦΟΡΙΑΣ
ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ ΥΛΙΚΩΝ

ΕΝΙΣΧΥΣΗ ΔΥΝΑΤΟΤΗΤΩΝ OPEN SOURCE ΕΡΓΑΛΕΙΟΥ ΑΠΕΙΚΟΝΙΣΗΣ ΝΕΥΡΟΦΥΣΙΟΛΟΓΙΚΩΝ ΔΕΔΟΜΕΝΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Ελευθέριος Πέτρου Γεωργοπετρέας

Επιβλέπων : Δημήτρης Ασκούνης

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 1η Μαρτίου 2024.

.....	.....	.....
Δημήτριος Ασκούνης	Ιωάννης Ψαράς	Χρυσόστομος Δούκας

Αθήνα, Φεβρουάριος 2024

.....

Ελευθέριος Πέτρου Γεωργοπετρέας

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

1. ΕΙΣΑΓΩΓΗ

1.1 ΑΝΤΙΚΕΙΜΕΝΟ / ΣΤΟΧΟΣ ΔΙΠΛΩΜΑΤΙΚΗΣ

Η μαγνητοεγκεφαλογραφία (MEG) και η ηλεκτροεγκεφαλογραφία (EEG), μεταξύ άλλων, αποτελούν απαραίτητα εργαλεία για την νευροαπεικόνιση, καθόσον παρέχουν σύνθετες πληροφορίες για την εγκεφαλική δραστηριότητα του εξεταζόμενου. Η παρούσα διατριβή ασχολείται με τον ρόλο του λογισμικού ανοικτού κώδικα, MNE-Python (Minimum Norm Estimation), στην εξερεύνηση, οπτικοποίηση και ανάλυση των ανθρώπινων νευροφυσιολογικών δεδομένων. Η MNE-Python, ένα από τα πιο ισχυρά εργαλεία της αγοράς, παρέχει σήμερα κρίσιμα χαρακτηριστικά στην οπτικοποίηση δεδομένων. Ωστόσο, το παρόν έργο αποσκοπεί στην υπέρβαση συγκεκριμένων περιορισμών που σχετίζονται με την κλιμάκωση, ανοίγοντας το δρόμο για διευρυμένα περιθώρια επιλογών, προσαρμοσμένα στις απαιτήσεις των διαφοροποιημένων αναγκών των ιατρών και των ερευνητών.

Το έργο επικεντρώνεται σε τρεις βασικές βελτιώσεις στο πλαίσιο της MNE-Python. Πρώτον, επιτρέπει την κλιμάκωση σε προκαθορισμένα είδη καναλιών, αντί για το σύνολο των δεδομένων. Δεύτερον, παρέχει στους χρήστες μεγαλύτερο έλεγχο υλοποιώντας συντελεστή κλίμακας προσαρμόσιμο σε οποιαδήποτε τιμή. Τέλος, εισάγει λειτουργίες ευαισθησίας, δηλαδή διασφαλίζει τον έλεγχο πάνω στις αποστάσεις που εμφανίζονται στην οθόνη, καθώς αυτό αποτελεί κοινή πρακτική σε αυτό το επιστημονικό πεδίο.

Οι προτεινόμενες βελτιώσεις κλιμάκωσης είναι καίριας σημασίας για τις ιατρικές και ερευνητικές κοινότητες, που χρησιμοποιούν την MNE-Python. Με την αντιμετώπιση αυτών των προκλήσεων, το λογισμικό γίνεται πιο προσαρμόσιμο στις υπάρχουσες ερευνητικές πρακτικές, και φέρνει πρωτοπορία στην ακρίβεια και την αποτελεσματικότητα της ερμηνείας των νευροφυσιολογικών δεδομένων και την εξαγωγή πληροφορίας.

Στα επόμενα κεφάλαια, γίνεται επισκόπηση του υφιστάμενου τοπίου των εργαλείων ανοιχτού κώδικα, διατύπωση της αρχιτεκτονικής και τεχνολογιών που χρησιμοποιούνται, διεύρυνση των μεθοδολογιών για κάθε βελτίωση και, τέλος, παρουσίαση των αποτελεσμάτων καθώς και παράθεση προβληματισμών σχετικά με μελλοντικές συνεισφορές και ερευνητικούς δρόμους.

Λέξεις Κλειδιά

Νευροφυσιολογικά δεδομένα, MNE-Python, ηλεκτροεγκεφαλογραφία, ΕΕΓ, μαγνητοεγκεφαλογραφία, ΜΕΓ, ανοιχτού κώδικα, Python, PyQt, ευαισθησία, κλιμάκωση

1.2 ABSTRACT

Magnetoencephalography (MEG) and electroencephalography (EEG), among others, are essential tools for neuroimaging, as they provide complex information about the subject's brain activity. This thesis addresses the role of the open-source software, MNE-Python (Minimum Norm Estimation), in the exploration, visualization and analysis of human neurophysiological data. MNE-Python, one of the most powerful tools on the market today, provides critical features in data visualization. However, this project aims to overcome specific limitations related to scalability, paving the way for expanded options tailored to the requirements of the diversified needs of physicians and researchers.

The project focuses on three key improvements within MNE-Python. First, it allows scaling to predefined types of channels, rather than the entire data set. Second, it provides users with greater control by implementing a scale factor adaptable to any value. Finally, it introduces sensitivity functions, i.e., it ensures control over the distances displayed on the screen, as this is a common practice in this scientific field.

The proposed scaling improvements are of key importance for the medical and research communities using MNE-Python. By addressing these challenges, the software becomes more adaptable to existing research practices, and brings leadership in the accuracy and efficiency of neurophysiological data interpretation and information extraction.

In the following chapters, we review the current landscape of open-source tools, articulate the architecture and technologies used, expand the methodologies for each improvement, and finally present the results and provide reflections on future contributions and research avenues.

Keywords

Neurophysiological data, MNE-Python, electroencephalography, EEG, MEG, magnetoencephalography, MEG, open source, Python, PyQt, sensitivity, scaling

1.3 ΠΕΡΙΕΧΟΜΕΝΑ

1.4 ΠΙΝΑΚΑΣ ΕΙΚΟΝΩΝ

Κεφάλαιο 2

Κεφάλαιο 3

2. ΣΥΝΕΙΣΦΟΡΑ ΣΕ ΛΟΓΙΣΜΙΚΑ ΑΝΟΙΧΤΟΥ ΚΩΔΙΚΑ

2.1 ΛΟΓΙΣΜΙΚΑ ΑΝΟΙΧΤΟΥ ΚΩΔΙΚΑ ΓΙΑ ΑΝΑΛΥΣΗ ΚΑΙ ΟΠΤΙΚΟΠΟΙΗΣΗ ΝΕΥΡΟΦΥΣΙΟΛΟΓΙΚΩΝ ΔΕΔΟΜΕΝΩΝ

2.1.1 ΟΡΙΣΜΟΣ ΤΩΝ ΝΕΥΡΟΦΥΣΙΟΛΟΓΙΚΩΝ ΔΕΔΟΜΕΝΩΝ

Τα νευροφυσιολογικά δεδομένα αναφέρονται στις καταγραφές της ηλεκτρομαγνητικής δραστηριότητας που παράγεται από το νευρικό σύστημα (Plum, 1960), οι οποίες συνήθως καταγράφονται μέσω διαφόρων τεχνικών, όπως η ηλεκτροεγκεφαλογραφία (EEG), η μαγνητοεγκεφαλογραφία (MEG) ή η λειτουργική απεικόνιση μαγνητικού συντονισμού (fMRI) και άλλες (Niedermeyer & da Silva, 2005). Οι καταγραφές αυτές παρέχουν πολύτιμες πληροφορίες για τη λειτουργία του εγκεφάλου και προσφέρουν ένα παράθυρο στις γνωστικές διαδικασίες, την αισθητηριακή αντίληψη, τις κινητικές λειτουργίες και τις νευρολογικές διαταραχές.

Η ηλεκτροεγκεφαλογραφία καταγράφει την ηλεκτρική δραστηριότητα του εγκεφάλου χρησιμοποιώντας ηλεκτρόδια που τοποθετούνται στο τριχωτό της κεφαλής, μετρώντας τις διακυμάνσεις της τάσης που προκύπτουν από τις ροές ιοντικού ρεύματος εντός των νευρώνων. Πραγματοποιεί τον εντοπισμό των πηγών δραστηριότητας του εγκεφάλου με μεγάλη ακρίβεια και ανάλυση, καθιστώντας την από τα πολυτιμότερα εργαλεία της νευροφυσιολογίας (Michel, et al., 2004).

Η μαγνητοεγκεφαλογραφία, από την άλλη πλευρά, καταγράφει τα μαγνητικά πεδία που παράγονται από τη νευρική δραστηριότητα χρησιμοποιώντας ευαίσθητους αισθητήρες. Διακρίνεται για την εξειδικευμένη απόδοση στην υψηλή χρονική ανάλυση (Hämäläinen, Hari, Ilmoniemi, Knuutila, & Lounasmaa, 1993). Επιπλέον, η λειτουργική απεικόνιση μαγνητικού συντονισμού είναι μια τεχνική νευροαπεικόνισης που μετρά την εγκεφαλική δραστηριότητα μέσω της ανίχνευσης αλλαγών στη ροή του αίματος. Με τη χαρτογράφηση αυτών των αλλαγών, η fMRI επιτρέπει στους ερευνητές να συσχετίζουν συγκεκριμένες περιοχές του εγκεφάλου με γνωστικές εργασίες ή ερεθίσματα, προσφέροντας πληροφορίες για τους νευρωνικούς μηχανισμούς που διέπουν διάφορες νοητικές διαδικασίες (Luck, 2014).

Τα νευροφυσιολογικά δεδομένα, τα οποία δεν περιορίζονται στα παραπάνω παραδείγματα, χαρακτηρίζονται από την εξαιρετική πολυπλοκότητα τους, απαιτώντας εξελιγμένες τεχνικές ανάλυσης για την εξαγωγή ουσιαστικών πληροφοριών (Hämäläinen, Hari, Ilmoniemi, Knuutila, & Lounasmaa, 1993). Οι ερευνητές χρησιμοποιούν διάφορες μεθόδους, όπως την ανάλυση συχνότητας χρόνου, την ανάλυση δυναμικού που σχετίζεται με γεγονότα (ERP), τον εντοπισμός πηγής και την ανάλυση συνδεσιμότητας, για να ερμηνεύσουν αυτά τα δεδομένα και να αποκαλύψουν τους υποκείμενους νευρωνικούς μηχανισμούς (Niedermeyer & da Silva, 2005).

Αυτή η πολυπλοκότητα λοιπόν καλεί για ιδιαίτερες λύσεις. Η χρήση λογισμικού, κυρίως ανοικτού κώδικα, αποτελεί κρίσιμο στοιχείο για την ανάλυση των νευροφυσιολογικών δεδομένων. Η δημιουργία και η ελεύθερη διάθεση λογισμικών ανοικτού κώδικα συμβάλλει στην προώθηση της ανοικτής επιστημονικής πρακτικής και την ανταλλαγή γνώσεων στον τομέα της νευροφυσιολογίας.

2.1.2 ΛΟΓΙΣΜΙΚΑ ΑΝΟΙΧΤΟΥ ΚΩΔΙΚΑ

Τα λογισμικά ανοιχτού κώδικα (open-source) διαδραματίζουν καθοριστικό ρόλο στη σφαίρα της ανάλυσης νευροφυσιολογικών δεδομένων, λόγω της προσφοράς προσβάσιμων και προσαρμόσιμων λύσεων τόσο για τους ερευνητές όσο και για τους κλινικούς γιατρούς. Αυτό το κεφάλαιο διατυπώνει τον ορισμό των προγραμμάτων αυτών και τις βασικές αρχές τους και αναδεικνύει διάφορα αξιοσημείωτα παραδείγματα που αφορούν συγκεκριμένα την ανάλυση νευροφυσιολογικών δεδομένων.

Τα προγράμματα ανοικτού κώδικα είναι εφαρμογές λογισμικού που χαρακτηρίζονται από την ελεύθερη διάθεση του πηγαίου τους κώδικα, επιτρέποντας στους χρήστες έτσι να συμμετέχουν στην σχεδίαση, τροποποίηση, επιθεώρηση και διανομή του (Kogut & Metiu, 2001). Τα λογισμικά αυτά αναπτύσσονται εθελοντικά και χωρίς τους περιορισμούς της ιδιοκτησίας. Λειτουργούν με μεγαλύτερο προτέρημα το ήθος της συνεργασίας, της διαφάνειας και της ανοικτής ανταλλαγής γνώσεων. Αυτός ο δημόσιος χαρακτήρας τους αποσκοπεί στην παγκοσμιοποίηση της επιστημονικής κοινότητας και τεχνογνωσίας, στην αξιολόγηση των νέων ιδεών, από οποιαδήποτε πηγή κι αν προέρχονται, και στην αξιοποίηση του μέγιστου διαθέσιμου δυναμικού. Εκμεταλλευόμενα τη συλλογική ικανότητα μιας ποικιλόμορφης κοινότητας προγραμματιστών, τα προγράμματα ανοικτού κώδικα εξελίσσονται ταχύρρυθμα, προσαρμοζόμενα στις αναδυόμενες ανάγκες και τις τεχνολογικές εξελίξεις (Kogut & Metiu, 2001).

Η ανάπτυξη του κώδικα ακολουθεί μια διαδικασία που καθοδηγείται από την κοινότητα, όπου οι χρήστες, οι προγραμματιστές και οι συνεισφέροντες συμμετέχουν ενεργά στη διαμόρφωση της εξέλιξης του λογισμικού. Η διαδικασία αυτή περιλαμβάνει συνήθως διάφορα βασικά στοιχεία:

Κατευθυντήριες γραμμές συνεισφοράς (contribution guidelines): Τα προγράμματα ανοικτού κώδικα συχνά θεσπίζονται από κατευθυντήριες γραμμές συνεισφοράς που περιγράφουν τις προτιμώμενες μεθόδους για την μορφοποίηση και υποβολή κώδικα, την αναφορά προβλημάτων και την υποβολή προτάσεων βελτίωσης. Αυτές οι κατευθυντήριες γραμμές διασφαλίζουν τη συνοχή, τη σαφήνεια και την τήρηση των προτύπων του έργου.

Συστήματα ελέγχου εκδόσεων (version control): Τα προγράμματα ανοικτού κώδικα χρησιμοποιούν συστήματα ελέγχου εκδόσεων, όπως το Git (Website: Git, n.d.), για την παρακολούθηση των αλλαγών, τη διαχείριση της συνεργασίας και τη διευκόλυνση της ενσωμάτωσης των αλλαγών από πολλούς συνεισφέροντες. Τα συστήματα ελέγχου εκδόσεων επιτρέπουν στους προγραμματιστές να εργάζονται συνεργατικά σε διάφορες

πτυχές του έργου, διατηρώντας παράλληλα μια συνεκτική και σταθερή βάση κώδικα. Επιπλέον προσφέρουν μία πληθώρα από άλλες λειτουργίες, ζωτικές για ένα έργο λογισμικού, όπως η συνεχής ενσωμάτωση και συνεχής παράδοση (CI-CD).

Ανασκόπηση κώδικα (code review): Πριν από τη συγχώνευση των συνεισφορών στην κύρια βάση κώδικα, τα έργα ανοικτού κώδικα υποβάλλονται σε αυστηρές διαδικασίες αναθεώρησης κώδικα. Έμπειροι προγραμματιστές εξετάζουν τις προτεινόμενες αλλαγές, παρέχουν ανατροφοδότηση και διασφαλίζουν την ποιότητα, την αποτελεσματικότητα και τη συμβατότητα του κώδικα. Όταν κριθεί πως οι αλλαγές καλύπτουν τις απαιτήσεις, τότε το αίτημα συγχώνευσης γίνεται αποδεκτό.

Άδειες χρήσης λογισμικού (software licenses): Οι άδειες χρήσης λογισμικού ανοικτού κώδικα διαδραματίζουν καθοριστικό ρόλο στη διαμόρφωση του συνεργατικού τοπίου της ανάπτυξης λογισμικού, καθώς περιγράφουν το νομικό πλαίσιο που διέπει τον διαμοιρασμό, τροποποίηση και διανομή του λογισμικού ανοικτού κώδικα, διασφαλίζοντας ότι οι προγραμματιστές έχουν την ελευθερία πρόσβασης, μελέτης, τροποποίησης και διανομής του πηγαίου κώδικα. Οι συνήθεις άδειες ανοικτού κώδικα, όπως η άδεια BSD 3-Clause (Website: The BSD 3-Clause License, n.d.), η Γενική Άδεια Δημόσιας Χρήσης GNU (GNU GPL) (Website: GNU General Public License, version 3, 2007), η Άδεια MIT (Website: The MIT License, n.d.) και η Άδεια Apache (Website: Apache License, Version 2.0, 2004), παρέχουν ποικίλους βαθμούς δικαιωμάτων και περιορισμών, αντιμετωπίζοντας θέματα όπως η αναδιανομή, η αναγνώριση και τα παράγωγα έργα. Οι άδειες αυτές οχυρώνουν την καινοτομία, ενθαρρύνουν τη διαφάνεια και επιτρέπουν τη δημιουργία ζωντανών κοινοτήτων γύρω από κοινές βάσεις κώδικα, προωθώντας τελικά την τεχνολογική πρόοδο με τρόπο συνεργατικό και χωρίς αποκλεισμούς.

Δέσμευση στην κοινότητα: Τα έργα ανοικτού κώδικα ευδοκιμούν με την εμπλοκή της κοινότητας, με τους χρήστες και τους συνεισφέροντες να συμμετέχουν ενεργά σε συζητήσεις, να μοιράζονται ιδέες και να παρέχουν υποστήριξη. Οι χώροι συζητήσεων, τα συστήματα ελέγχου εκδόσεων και οι διαδικτυακές πλατφόρμες λειτουργούν ως κόμβοι συνεργασίας, ανταλλαγής γνώσεων και επίλυσης προβλημάτων.

2.1.3 ΛΟΓΙΣΜΙΚΑ ΑΝΟΙΧΤΟΥ ΚΩΔΙΚΑ ΓΙΑ ΤΗΝ ΑΝΑΛΥΣΗ ΝΕΥΡΟΦΥΣΙΟΛΟΓΙΚΩΝ ΔΕΔΟΜΕΝΩΝ

Τα λογισμικά ανοιχτού κώδικα για την ανάλυση νευροφυσιολογικών δεδομένων προσφέρουν συνήθως ένα εύρος λειτουργιών, συμπεριλαμβανομένων εργαλείων προεπεξεργασίας δεδομένων, οπτικοποίησης, στατιστικής ανάλυσης και ερμηνείας. Χρησιμοποιούν αλγορίθμους και τεχνικές προσαρμοσμένες στις ειδικές απαιτήσεις της νευροφυσιολογικής έρευνας, επιτρέποντας στους χρήστες να εξάγουν σημαντικές πληροφορίες από πολύπλοκα σύνολα δεδομένων. Επιπλέον, τα προγράμματα αυτά συχνά ενσωματώνονται με υπάρχουσες βιβλιοθήκες και εργαλειοθήκες διαφόρων λειτουργιών, οι οποίες διασφαλίζουν την αποδοτικότητα και αξιοπιστία τους.

Παρακάτω παρατίθενται επιγραμματικά παραδείγματα τέτοιων προγραμμάτων που εξειδικεύονται στην ανάλυση νευροφυσιολογικών δεδομένων:

1. MNE-Python (Website: MNE-Python - Home, n.d.)

Η βιβλιοθήκη MNE-Python (Minimum Norm Estimation) είναι ένα ολοκληρωμένο πρόγραμμα ανοικτού κώδικα προσαρμοσμένο για την ανάλυση δεδομένων MEG και EEG. Η αρθρωτή αρχιτεκτονική του προσφέρει ένα ευρύ φάσμα εργαλείων για προεπεξεργασία, εντοπισμό πηγής, ανάλυση συνδεσιμότητας, οπτικοποίηση και ανάλυση με χρήση μηχανικής μάθησης. Με εκτενή τεκμηρίωση και ενεργούς χώρους συζήτησης χρηστών, η MNE-Python απευθύνεται τόσο σε αρχάριους όσο και σε προχωρημένους χρήστες στη νευροφυσιολογική έρευνα. (Gramfort, et al., 2013)

2. EDFbrowser (Website: EDFbrowser, n.d.)

Το EDFbrowser είναι ένα ευέλικτο εργαλείο ανοικτού κώδικα, ειδικά σχεδιασμένο για την οπτικοποίηση και την ανάλυση νευροφυσιολογικών δεδομένων που είναι αποθηκευμένα στο European Data Format (EDF). Παρέχει μια φιλική προς το χρήστη διεπαφή για την περιήγηση, τον σχολιασμό, το φιλτράρισμα και την εξαγωγή καταγραφών EEG. Η εστίαση του EDFbrowser στη συμβατότητα με τυποποιημένες μορφές αρχείων και η ευκολία χρήσης το καθιστούν ανεκτίμητο πόρο για ερευνητές και κλινικούς γιατρούς που εργάζονται με δεδομένα EEG.

3. EEGLAB (Website: EEGLAB, n.d.)

Το EEGLAB είναι ένα ευρέως χρησιμοποιούμενο πρόγραμμα ανοικτού κώδικα που υλοποιείται σε MATLAB για την ανάλυση δεδομένων EEG. Προσφέρει μια ολοκληρωμένη σουίτα εργαλείων για την επεξεργασία σήματος, την αφαίρεση τεχνουργημάτων, την ανάλυση του δυναμικού βασισμένη σε γεγονότα και τον εντοπισμό πηγής σήματος. Το πλεονέκτημα του EEGLAB έγκειται στο εκτεταμένο οικοσύστημα πρόσθετων προγραμμάτων, επιτρέποντας στους χρήστες να προσαρμόζουν και να επεκτείνουν τη λειτουργικότητά του ώστε να ανταποκρίνεται στις συγκεκριμένες ερευνητικές τους ανάγκες. (Delorme & Makeig, 2004)

4. FieldTrip (Website: FieldTrip, n.d.)

Το FieldTrip είναι μια εργαλειοθήκη MATLAB σχεδιασμένη για προηγμένη ανάλυση νευροφυσιολογικών δεδομένων, συμπεριλαμβανομένων των EEG, MEG και ενδοκρανιακών καταγραφών. Η έμφαση που δίνει στην ευελιξία και την προσαρμογή του επιτρέπει στους χρήστες να εφαρμόζουν εξελιγμένες στατιστικές μεθόδους, αλγόριθμους ανακατασκευής πηγής και εργαλεία ανάλυσης συνδεσιμότητας. Η διαφανής διαδικασία ανάπτυξης του FieldTrip και η ενεργή κοινότητα χρηστών προωθούν τη συνεργασία, την ανταλλαγή γνώσεων και τη συνεχή βελτίωση της νευροφυσιολογικής έρευνας. (Oostenveld, Fries, Maris, & Schoffelen, 2011)

2.2 Η ΒΙΒΛΙΟΘΗΚΗ MNE-PYTHON

2.2.1 ΠΕΡΙΓΡΑΦΗ ΤΗΣ ΒΙΒΛΙΟΘΗΚΗΣ ΚΑΙ ΤΩΝ ΔΥΝΑΤΟΤΗΤΩΝ ΤΗΣ

Η βιβλιοθήκη MNE-Python (Website: MNE-Python - Home, n.d.), η οποία ξεκίνησε το 2011 και είναι γραμμένη στην Python3 (Website: Python, n.d.), αποτελεί πολύτιμο εργαλείο για την ανάλυση νευροφυσιολογικών δεδομένων, και είναι ευρέως γνωστή για την αποτελεσματικότητα και την προσβασιμότητά της στην εξέταση δεδομένων MEG και EEG. Κεντρικό ρόλο στη χρησιμότητα της παίζει η ολοκληρωμένη εργαλειοθήκη προεπεξεργασίας της, που περιλαμβάνει, μεταξύ άλλων, την αφαίρεση τεχνουργημάτων, το φιλτράρισμα σήματος και την εποχικοποίηση. Με αυτόν τον τρόπο εξασφαλίζεται η ακεραιότητα, ποιότητα και αναγνωσιμότητα των δεδομένων στις επακόλουθες αναλύσεις. Επιπλέον, προηγμένοι αλγόριθμοι όπως η εκτίμηση ελάχιστης νόρμας (Minimum Norm Estimation) και η δυναμική στατιστική παραμετροποιημένη χαρτογράφηση (dSPM) επιτρέπουν τον ακριβή εντοπισμό της πηγής, υποδεικνύοντας έτσι την προέλευση της νευρικής δραστηριότητας. Η ανάλυση συνδεσιμότητας, που διευκολύνεται από μέτρα όπως η συνοχή και η τιμή κλειδώματος φάσης (PLV), είναι απαραίτητη για την εμβάθυνση στις διαπεριφερειακές αλληλεπιδράσεις του εγκεφάλου, και συνεπώς την αποκάλυψη της δυναμικής των νευρωνικών δικτύων. Παράλληλα με όλες τις λειτουργίες, η βιβλιοθήκη παρέχει και πληθώρα συνόλων δεδομένων (datasets), για εξοικείωση με το λογισμικό και δοκιμές (Gramfort, et al., 2013).

Πέραν των ανωτέρω, η MNE-Python ενσωματώνεται και με άλλες βιβλιοθήκες Python, όπως οι NumPy (Website: NumPy, n.d.) και SciPy (Website: SciPy, n.d.), οι οποίες ενισχύουν τις δυνατότητες χειρισμού και ανάλυσης δεδομένων, συνεπώς και την αποδοτικότητα του λογισμικού. Στο πεδίο της μηχανικής μάθησης, η MNE-Python παρέχει επίσης μια γέφυρα μεταξύ της παραδοσιακής νευροφυσιολογικής ανάλυσης και των προηγμένων τεχνικών που βασίζονται σε δεδομένα. Οι ερευνητές μπορούν να αξιοποιήσουν αλγορίθμους μηχανικής μάθησης για εργασίες όπως η εξαγωγή χαρακτηριστικών, η ταξινόμηση και η πρόβλεψη. Μέσω τεχνικών όπως αυτών αποκαλύπτονται πολύπλοκα μοτίβα και σχέσεις εντός των νευροφυσιολογικών δεδομένων. Αυτή η ενσωμάτωση αλγορίθμων τεχνητής νοημοσύνης όχι μόνο ενισχύει τις αναλυτικές δυνατότητες της MNE, αλλά ανοίγει επίσης νέους δρόμους για τη διερεύνηση των περιπλοκών της λειτουργίας και της δυσλειτουργίας του εγκεφάλου, οι οποίοι είναι απρόσιτοι με κλασσικές μεθόδους ανάλυσης.

Ωστόσο, η πραγματική διαφορά έγκειται στην οπτικοποίηση. Κάνοντας χρήση των βιβλιοθηκών PyQT5 (Website: PyQT5, n.d.), PyQtGraph (Website: PyQtGraph, n.d.) και Matplotlib (Website: MatPlotLib, n.d.), η MNE-Python προσφέρει μια πληθώρα εργαλείων κατάλληλων για την αποτελεσματική ερμηνεία δεδομένων και εξαγωγή συμπερασμάτων. Από διαδραστικές λειτουργίες σχεδίασης έως τοπογραφικούς χάρτες και τρισδιάστατες απεικονίσεις εγκεφαλικών επιφανειών, παρέχεται πλέον στους χρήστες ένα ευρύ φάσμα από ευέλικτα μέσα που θα τους δώσουν τη δυνατότητα να εξερευνήσουν και να διαφωτίσουν

νευροφυσιολογικές γνώσεις με σαφήνεια και ακρίβεια. Αυτές οι τεχνικές οπτικοποίησης διευκολύνουν την εξερεύνηση των δεδομένων και κατά συνέπεια μπορούν να ανοίξουν παράθυρα στη γνώση τα οποία διαφορετικά δεν θα ήταν προσβάσιμα στους ερευνητές και τους ιατρούς.

Αναφορικά με τις άδειες χρήσης, η MNE χρησιμοποιεί την άδεια BSD 3-Clause (Website: The BSD 3-Clause License, n.d.). Αυτή η άδεια χρήσης λογισμικού περιλαμβάνει τρεις ρήτρες. Οι πρώτες δύο καθορίζουν το πλαίσιο για την αναδιανομή και τροποποίηση του υλικού, και την αναγνώρισή του από παράγωγα έργα. Η τρίτη αποτελεί τη ρήτρα αποποίησης εγγύησης, δηλαδή δηλώνει ότι το λογισμικό παρέχεται χωρίς καμία εγγύηση, ρητή ή σιωπηρή.

Τέλος, το οικοσύστημα MNE περιλαμβάνει όλες τις απαραίτητες σελίδες για την οργάνωση και την επικοινωνία των μελών της κοινότητας. Αυτές περιλαμβάνουν την εκτενή τεκμηρίωση του λογισμικού (Website: MNE-Python - Documentation, n.d.), τον οδηγό συνεισφοράς (Website: MNE-Python - Contributing Guide, 2024), τον οργανισμό στο Github mne-tools (Website: Github organisation: mne-tools, n.d.) και διαδικτυακούς χώρους συζήτησης (Website: MNE Forum, n.d.).

2.2.2 ΕΝΙΣΧΥΣΗ ΤΗΣ MNE-PYTHON ΜΕ ΓΝΩΜΟΝΑ ΤΟ EDFBROWSER

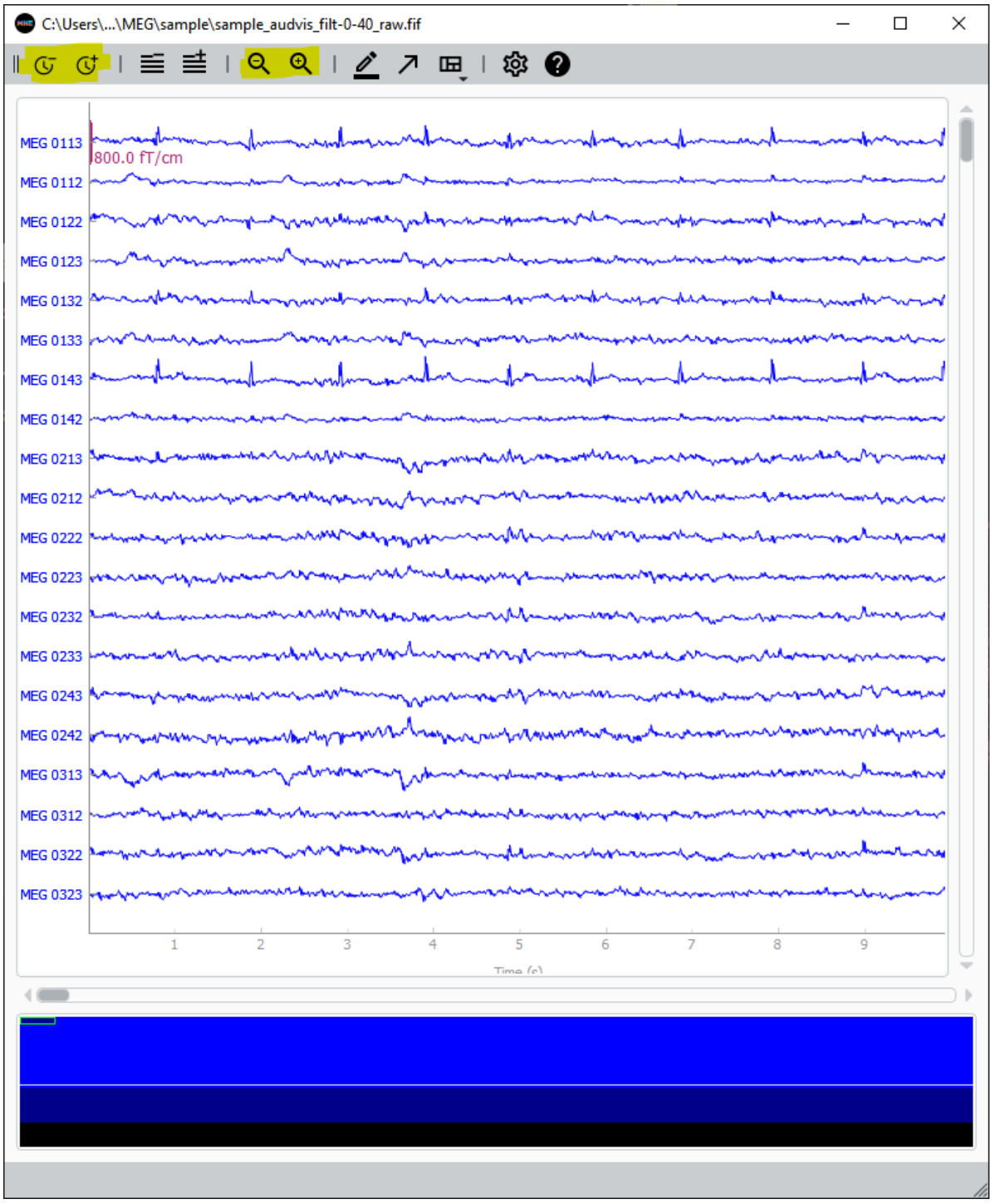
Η βιβλιοθήκη MNE-Python, αν και ισχυρή ως προς τις δυνατότητές της για την ανάλυση νευροφυσιολογικών δεδομένων, παρουσιάζει ορισμένους περιορισμούς σε σύγκριση με εργαλεία όπως το EDFBrowser. Ένας σημαντικός τομέας στον οποίο η MNE-Python υστερεί είναι οι δυνατότητες κλιμάκωσης και η εμφάνιση πραγματικών αποστάσεων στην οθόνη.

Το EDFBrowser προσφέρει στους χρήστες την ευελιξία να κλιμακώνουν είτε τον άξονα πλάτους είτε τον άξονα χρόνου και μάλιστα με ένα ευρύ φάσμα ευελιξίας στις επιλογές αυτές. Αυτό το επίπεδο προσαρμογής πραγματοποιείται είτε σε προκαθορισμένες και δυναμικές τιμές, είτε σε τιμές που εισάγει ο χρήστης (βλ. Εικόνες 2.2, 2.3). Αντίθετα, η διεπαφή της MNE-Python δεν διαθέτει επί του παρόντος αυτή την ευελιξία, περιορίζοντας τους χρήστες στην ικανότητά κλιμάκωσης όλων των δεδομένων ταυτόχρονα, και όχι συγκεκριμένων ειδών, καθώς και μόνο με προκαθορισμένους συντελεστές κλίμακας (βλ. Εικόνα 2.1).

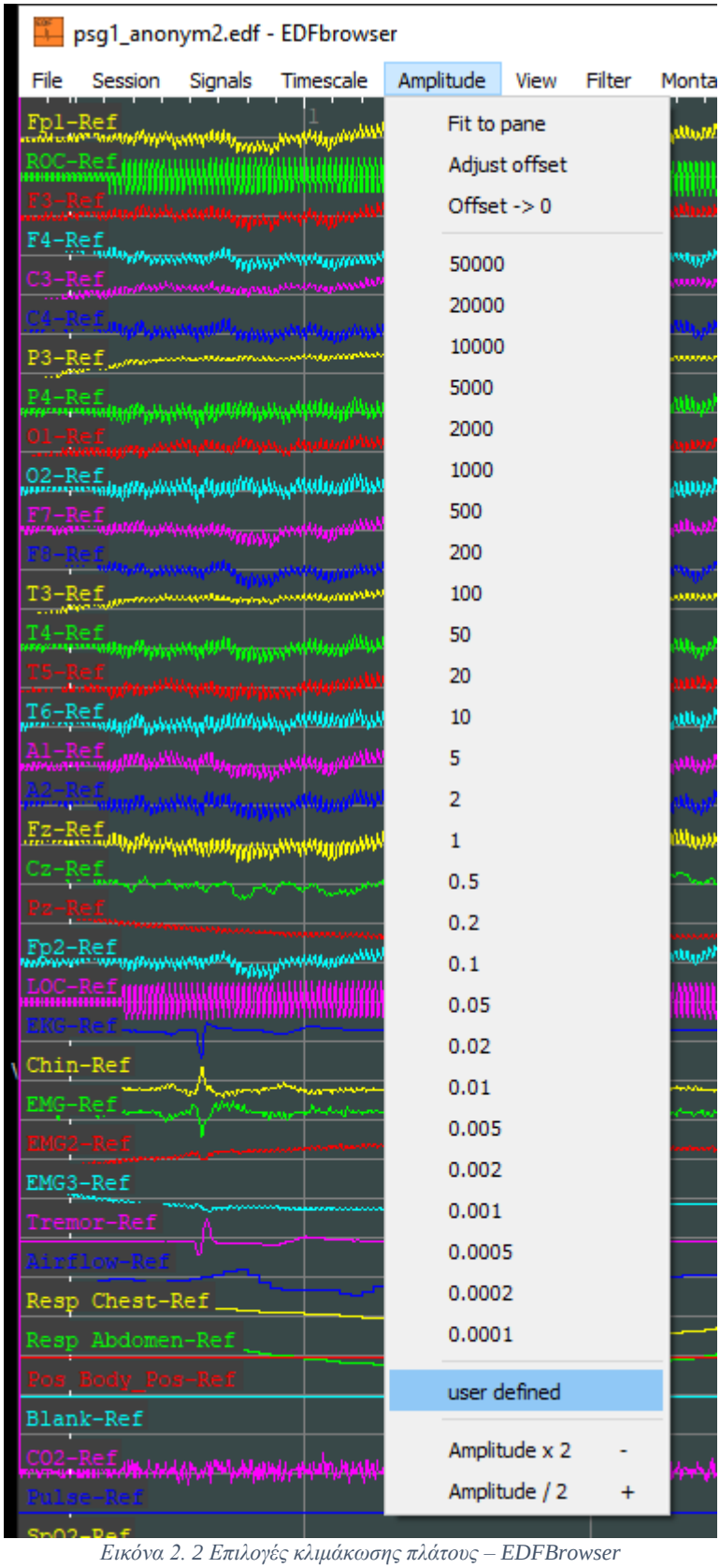
Επιπλέον, η απεικόνιση πραγματικών αποστάσεων στην οθόνη είναι απαραίτητη για την ακριβή ερμηνεία και ανάλυση των νευροφυσιολογικών δεδομένων. Οι ερευνητές και οι ιατροί είναι εκπαιδευμένοι να εντοπίζουν μοτίβα σε γραφήματα αρχικά εκτυπωμένα, βασιζόμενοι σε συνεπείς μονάδες γραφήματος ανά μονάδα μήκους για ακριβή ερμηνεία. Στο EDFBrowser υπάρχει αυτή η λειτουργικότητα μετά από τη βαθμονόμηση οθόνης από το χρήστη (βλ. Εικόνα 2.4). Ωστόσο, η MNE-Python δεν διαθέτει επί του παρόντος τη δυνατότητα να διασφαλίζει ότι τα μήκη που εμφανίζονται στην οθόνη αντιστοιχούν σε

μετρήσεις του πραγματικού κόσμου (βλ. Εικόνα 2.1). Η προσθήκη αυτής της δυνατότητας είναι σίγουρο πως θα παρέχει ένα πανίσχυρο εργαλείο για τον χρήστη που εξερευνά τα δεδομένα.

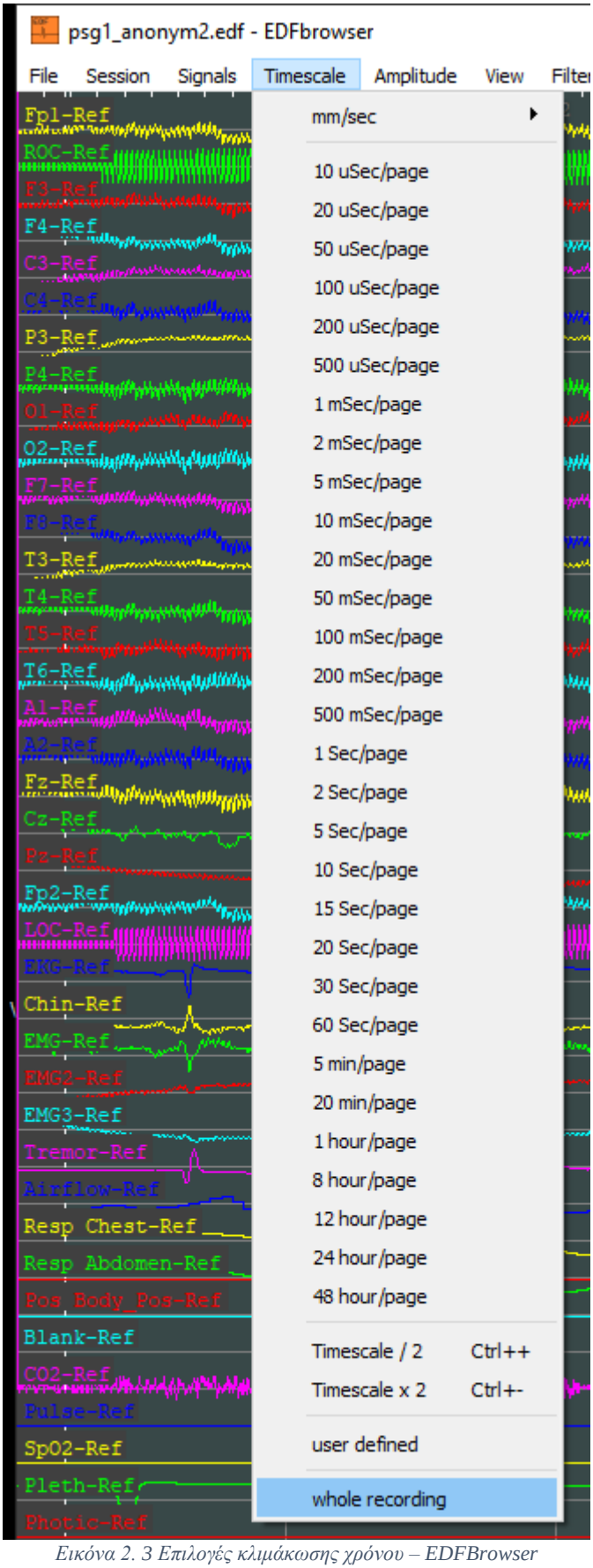
Συμπερασματικά, ενώ η MNE-Python προσφέρει μια ολοκληρωμένη πλατφόρμα για την ανάλυση νευροφυσιολογικών δεδομένων, υφίστανται ορισμένοι τομείς όπου υπάρχουν σαφή περιθώρια βελτίωσης, ιδίως όσον αφορά τις δυνατότητες κλιμάκωσης και την απεικόνιση ευαισθησίας. Εφόσον αντιμετωπισθούν οι περιορισμοί αυτοί με επιτυχία, η MNE-Python θα μπορεί να ανταποκριθεί καλύτερα στις ανάγκες των ερευνητών και των κλινικών ιατρών, παρέχοντάς τους τα εργαλεία που χρειάζονται για τη διεξαγωγή ακριβών και διορατικών αναλύσεων νευροφυσιολογικών δεδομένων.



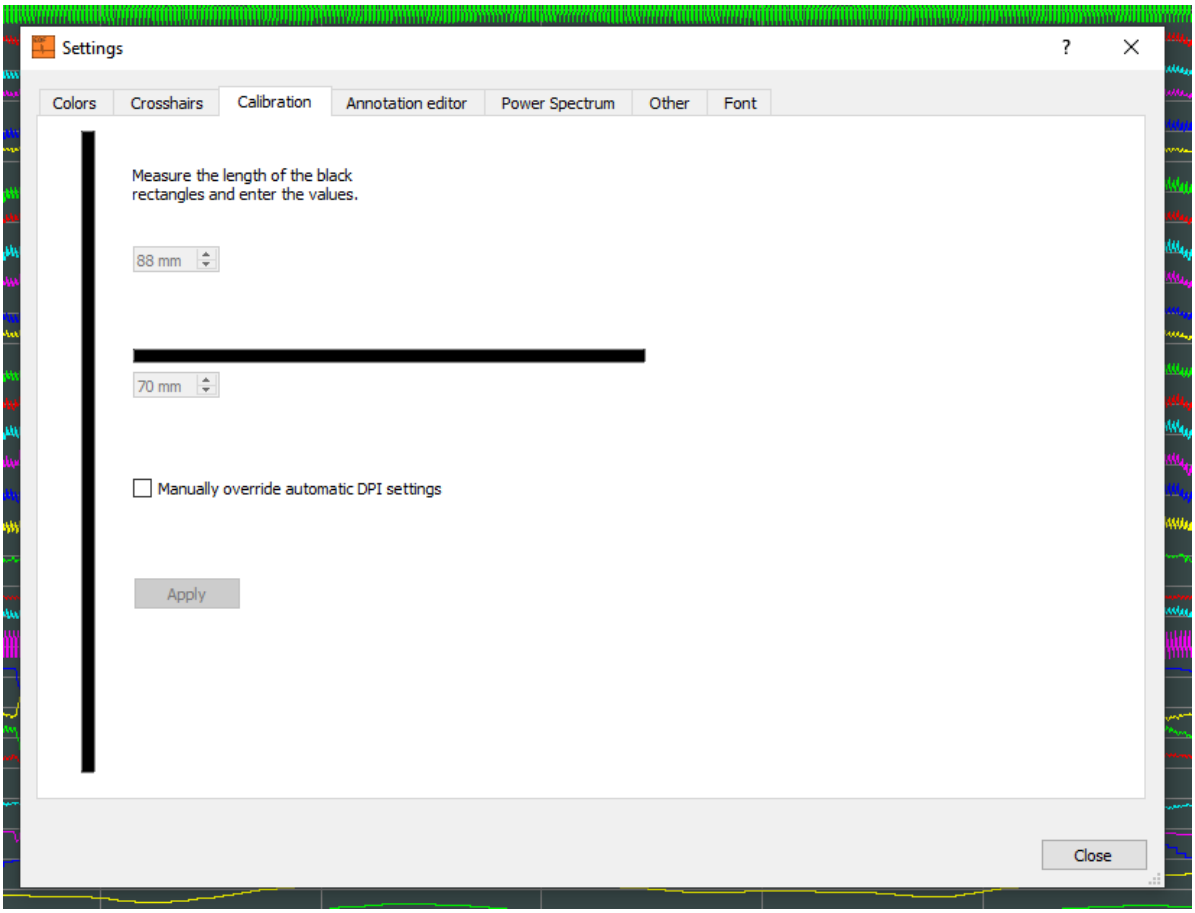
Εικόνα 2. 1 Η αρχική μορφή στην MNE-Python



Εικόνα 2. 2 Επιλογές κλιμάκωσης πλάτους – EDFBrowser



Εικόνα 2. 3 Επιλογές κλιμάκωσης χρόνου – EDFBrowser



Εικόνα 2. 4 Βαθμονόμηση οθόνης – EDFBrowser

3. ΥΛΟΠΟΙΗΣΗ ΤΩΝ ΝΕΩΝ ΛΕΙΤΟΥΡΓΙΩΝ

https://github.com/Grifunf/mne-qt-browser-dev

3.1 ΣΧΕΤΙΚΕΣ ΤΕΧΝΟΛΟΓΙΕΣ

Η βιβλιοθήκη MNE-Python είναι δομημένη στον οργανισμό mne-tools του Github (Website: Github organisation: mne-tools, n.d.), ο οποίος περιλαμβάνει το κύριο αποθετήριο (repository), mne-python (Website: Github repository: mne-python, n.d.), μαζί με πολλά βοηθητικά αποθετήρια που εξυπηρετούν διάφορους σκοπούς, όπως το αποθετήριο προς επεξεργασία, mne-qt-browser (Website: Github repository: mne-qt-browser, n.d.). Αυτή η οργανωτική προσέγγιση διευκολύνει μια αρθρωτή και κλιμακούμενη αρχιτεκτονική για την ανάλυση νευροφυσιολογικών δεδομένων στο πλαίσιο του οικοσυστήματος MNE.

Η βιβλιοθήκη είναι αναπτυγμένη στη γλώσσα προγραμματισμού Python 3 (Website: Python, n.d.), μία από τις πιο δημοφιλείς, φιλικές προς το χρήστη και δυναμικές γλώσσες. Ως σύστημα διαχείρισης πακέτων χρησιμοποιείται το Conda (Website: Conda, n.d.), επιτρέποντας την απρόσκοπτη διαχείριση του περιβάλλοντος και των επιπρόσθετων βιβλιοθηκών. Για την διαχείριση των μεγάλων αριθμητικών συνόλων δεδομένων χρησιμοποιείται η NumPy (Website: NumPy, n.d.), η οποία προσθέτει στην Python τις δυνατότητες αριθμητικών πινάκων με μεγάλες χρονικές αποδόσεις στις πράξεις τους.

Για τη σχεδίαση και την οπτικοποίηση, η MNE-Python χρησιμοποιεί την βιβλιοθήκη PyQt5, έκδοσης 5.15 (Website: PyQT5, n.d.). Η PyQt5, μία Python παραλλαγή για το πλαίσιο εφαρμογών Qt, αρχικά στη C++ (Website: Qt, n.d.), επιτρέπει τη δημιουργία διαδραστικών γραφικών διεπαφών χρήστη (GUI) και προσφέρει μια ολοκληρωμένη εργαλειοθήκη για την ανάπτυξη και τον εμπλουτισμό τους. Η PyQt5 προσφέρει εκτεταμένες λειτουργίες γραφικών παραστάσεων, επιτρέποντας τη δημιουργία διαφόρων ειδών οπτικοποιήσεων προσαρμοσμένων στην ανάλυση νευροφυσιολογικών δεδομένων. Αυτές ποικίλουν από διαγράμματα χρονοσειρών με πολλαπλές περαιτέρω επιλογές, τοπογραφικούς χάρτες, τρισδιάστατες απεικονίσεις εγκεφαλικών επιφανειών και άλλα. Ο σχετικός κώδικας βρίσκεται στο αποθετήριο mne-qt-browser, το οποίο έχει ονομαστεί από την βιβλιοθήκη αυτή.

Στα πλαίσια της PyQt5, το MNE-Python ενσωματώνει το PyQtGraph (Website: PyQtGraph, n.d.) για διαδραστικές γραφικές παραστάσεις υψηλής απόδοσης. Το PyQtGraph εξειδικεύεται στην ανάλυση και απεικόνιση δεδομένων σε πολλές επιστημονικές εφαρμογές. Για αυτό το λόγο χρησιμοποιεί επιτάχυνση GPU για γρήγορη απόδοση, διευκολύνοντας την ομαλή απεικόνιση μεγάλων συνόλων νευροφυσιολογικών δεδομένων με ελάχιστη καθυστέρηση.

Οι δοκιμές αποτελούν αναπόσπαστο μέρος της διαδικασίας ανάπτυξης λογισμικού, με το Pytest (Website: Pytest, n.d.) να χρησιμεύει ως το πλαίσιο για τις αυτοματοποιημένες δοκιμές. Εγκατεστημένο σε έκδοση 7.4, παρέχει μια ευέλικτη και φιλική προς το χρήστη πλατφόρμα για τη συγγραφή και την εκτέλεση περιπτώσεων δοκιμών, διασφαλίζοντας τη

σταθερότητα και την αξιοπιστία του λογισμικού σε διάφορα σενάρια και ακραίες περιπτώσεις.

Συνοπτικά, τα τροποποιημένα στοιχεία της MNE βασίζονται στην Python, PyQt5, PyQtGraph και Pytest. Στην πορεία θα αναπτυχθούν αναλυτικά οι αλλαγές που υλοποιήθηκαν, καθώς και η πορεία των ενεργειών που οδήγησε σε αυτή την υλοποίηση.

3.2 ΟΔΗΓΟΣ ΣΥΝΕΙΣΦΟΡΑΣ

Η συνεισφορά στη βιβλιοθήκη MNE-Python περιλαμβάνει διάφορα βήματα που περιγράφονται στον οδηγό συνεισφοράς, ο οποίος βρίσκεται στην ιστοσελίδα της (Website: MNE-Python - Contributing Guide, 2024). Ακολουθεί μια επισκόπηση των βασικών βημάτων:

1. Εξοικείωση με τις οδηγίες συνεισφοράς: Πριν συνεισφέρει κάποιος στην MNE-Python, είναι σημαντικό να μελετήσει τις κατευθυντήριες γραμμές συνεισφοράς που περιγράφονται στο αποθετήριο του έργου. Εκεί παρέχονται λεπτομερείς οδηγίες για την αποτελεσματική συνεισφορά, συμπεριλαμβανομένων των προτύπων κωδικοποίησης, δοκιμών και τεκμηρίωσης, του κώδικα συμπεριφοράς στους διαδικτυακούς χώρους και των διαδικασιών υποβολής.

2. Δημιουργία του περιβάλλοντος εργασίας: Στην πορεία ο προγραμματιστής πρέπει να ρυθμίσει το git, με το να συνδεθεί στο λογαριασμό του. Κατόπιν, χρησιμοποιώντας στο Conda, δημιουργεί ένα περιβάλλον Python όπου εγκαθίστανται όλα τα απαραίτητα πακέτα του αρχείου environment.yml του αποθετηρίου mne-python.

```
$curl --remote-name https://raw.githubusercontent.com/mne-tools/mne-
python/main/environment.yml
$conda env create --file environment.yml --name mnedev
$conda activate mnedev
```

3. Διακλάδωση του αποθετηρίου: Ο προγραμματιστής δημιουργεί μία διακλάδωση του αρχικού αποθετηρίου στο δικό του λογαριασμό. Με αυτό τον τρόπο μπορεί να εργαστεί τοπικά και να καταγράφει τις αλλαγές στη δική του διακλάδωση του κώδικα, αξιοποιώντας τις λειτουργίες του συστήματος διαχείρισης εκδόσεων git, ιδιαίτερα τους αυτοματισμούς που διαθέτει για τις δοκιμές και βελτιώσεις του κώδικα. Αφού πραγματοποιήσει την διακλάδωση στο Github, εκτελεί εντολές για να συνδέσει το διακλαδωμένο αποθετήριο με το επίσημο.

```
$git remote add upstream https://github.com/mne-tools/mne-python.git
$git fetch --all
$git config --local blame.ignoreRevsFile .git-blame-ignore-revs
```

4. Αίτημα συγχώνευσης (Pull Request): Όταν ο προγραμματιστής έχει ολοκληρώσει τις επιθυμητές αλλαγές, τότε δημιουργεί ένα αίτημα συγχώνευσης από την διακλάδωσή του προς το αρχικό αποθετήριο. Όλες οι προτεινόμενες αλλαγές ελέγχονται από άλλα μέλη του οργανισμού και, μετά από ενδεχόμενη ανατροφοδότηση και τις επιθυμητές βελτιώσεις, είτε απορρίπτονται, είτε γίνονται δεκτές και ενσωματώνονται στην MNE-Python και γίνονται μέρος του επίσημου έργου.

Ακολουθώντας αυτά τα βήματα, οι συνεισφέροντες μπορούν να συμβάλλουν αποτελεσματικά στη συνεχή ανάπτυξη και βελτίωση της βιβλιοθήκης MNE-Python,

συμβάλλοντας στην πρόοδο του πεδίου της ανάλυσης και της έρευνας νευροφυσιολογικών δεδομένων.

3.3 ΥΛΟΠΟΙΗΣΗ

3.3.1 ΧΡΟΝΙΚΟ ΣΥΝΕΙΣΦΟΡΑΣ

Η διαδικασία συμβολής στο έργο MNE εκτυλίχθηκε σε διάφορα βασικά στάδια, καθένα από τα οποία περιλαμβάνει σημαντικές δράσεις και αλληλεπιδράσεις εντός της κοινότητας. Ακολουθεί ένα λεπτομερές χρονοδιάγραμμα των εν λόγω σταδίων:

1. Ενημέρωση για το θέμα και αρχική επαφή: Το ταξίδι ξεκίνησε με την επίγνωση του ζητήματος που συζητήθηκε στους χώρους συζήτησης της MNE-Python (Website: MNE Forum - Scaling & Sensitivity, 2022) και του Github (Website: Github issue: Scaling & Sensitivity, 2022), συνοδευόμενο από πληροφορίες σχετικά με τις απαιτήσεις και τις ανάγκες του έργου. Αναγνωρίζοντας την ευκαιρία να συνεισφέρω, επικοινώνησα με μέλη του οργανισμού MNE, εκφράζοντας την πρόθεσή μου να εργαστώ στο έργο και ζητώντας καθοδήγηση σχετικά με τις συγκεκριμένες απαιτήσεις και την κατάλληλη πορεία δράσης.

2. Δέσμευση και επιμόρφωση: Μετά την αρχική επαφή με τα μέλη της οργάνωσης MNE-Python, συμμετείχα σε συζητήσεις στους παραπάνω χώρους για να αποκτήσω βαθύτερη κατανόηση των στόχων και των σκοπών του έργου, της υπάρχουσας υποδομής και των λειτουργικών απαιτήσεων. Μέσω αυτών των αλληλεπιδράσεων, έλαβα πολύτιμη επίγνωση σχετικά με το πώς να προχωρήσω στην αντιμετώπιση του συγκεκριμένου ζητήματος και την ευθυγράμμιση των προσπαθειών μου με τα πρότυπα και τις συμβάσεις του έργου.

3. Πρόοδος και συνεχής ανατροφοδότηση: Με σαφή κατανόηση των απαιτήσεων και λεπτομερή σχεδίαση, ξεκίνησα το έργο της υλοποίησης των απαραίτητων αλλαγών. Με την πάροδο του χρόνου, άνοιξα ένα προσωρινό αίτημα συγχώνευσης (Website: Github Pull Request 1, 2023) για να προτείνω τις συνεισφορές μου στο αποθετήριο mne-qt-browser. Καθ' όλη αυτή τη διάρκεια, λάμβανα συνεχή ανατροφοδότηση από την κοινότητα της MNE-Python στη σελίδα του αιτήματος, επισημαίνοντας τομείς προς βελτίωση και προτείνοντας τροποποιήσεις για να διασφαλιστεί η ευθυγράμμιση με τις κατευθυντήριες γραμμές του έργου.

4. Προσαρμογή και επαναληπτική ανάπτυξη: Λόγω της σπονδυλωτής φύσης του έργου, καθώς και από το γεγονός ότι ορισμένες πτυχές του αρχικού αιτήματος θεωρήθηκαν ότι είχαν ξεπεραστεί, δημιουργήθηκε ένα δεύτερο προσωρινό αίτημα συγχώνευσης (Website: Github Pull Request 2, 2023), το οποίο συνοδεύτηκε από περαιτέρω συζητήσεις και διαβουλεύσεις εντός της κοινότητας. Αυτή η επαναληπτική διαδικασία επέτρεψε την προσαρμογή και περαιτέρω βελτίωση των προτεινόμενων αλλαγών ώστε να ταιριάζουν καλύτερα στους γενικούς στόχους του έργου.

5. Τελική υποβολή και αναθεώρηση: Καθώς το έργο έφτασε στο πέρας, έγινε η τελική επίσημη υποβολή, η οποία περιλάμβανε όλες τις απαραίτητες τροποποιήσεις και

βελτιώσεις (Website: Github Final Pull Request, 2024). Αυτή η τελική εκδοχή θα υποβληθεί σε διεξοδική εξέταση και δοκιμή από μέλη της κοινότητας MNE-Python, διασφαλίζοντας την ποιότητα και τη συμβατότητα των προτεινόμενων αλλαγών με την υπάρχουσα βάση κώδικα. Αναμένεται ότι, μετά τη διαδικασία αναθεώρησης, οι συνεισφορές θα ενσωματωθούν στο επίσημο αποθετήριο της MNE-Python, ενισχύοντας περαιτέρω τις δυνατότητες και τη λειτουργικότητα του.

Συνοπτικά, η πορεία της συνεισφοράς στο έργο MNE-Python περιλάμβανε μια σειρά συντονισμένων ενεργειών, αλληλεπιδράσεων και επαναλήψεων, με αποκορύφωμα την υποβολή προτεινόμενων αλλαγών για αναθεώρηση και ενσωμάτωση. Μέσω της συνεργασίας και της τήρησης των κατευθυντήριων γραμμών συνεισφοράς, η υλοποίηση των επιθυμητών αλλαγών έγινε με ευκολία και αξιοπιστία.

3.3.2 ΑΡΧΙΚΕΣ ΕΝΕΡΓΕΙΕΣ

Παρακάτω περιγράφονται τα πρώτα βήματα που ακολουθήθηκαν για την ρύθμιση του περιβάλλοντος ανάπτυξης κώδικα, για την υλοποίηση του έργου. Αρχικά δημιουργήθηκε μία διακλάδωση του αποθετηρίου mne-qt-browser ονομαζόμενη mne-qt-browser-dev (Website: Github forked repository: mne-qt-browser-dev, 2023), που αφορά τις λειτουργικότητες διεπαφών και οπτικοποίησης της βιβλιοθήκης PyQt5 στο περιβάλλον MNE-Python. Το αρχείο mne_qt_browser/_pg_figure.py είναι αυτό που αφορά τη συγκεκριμένη διεπαφή η οποία τροποποιήθηκε.

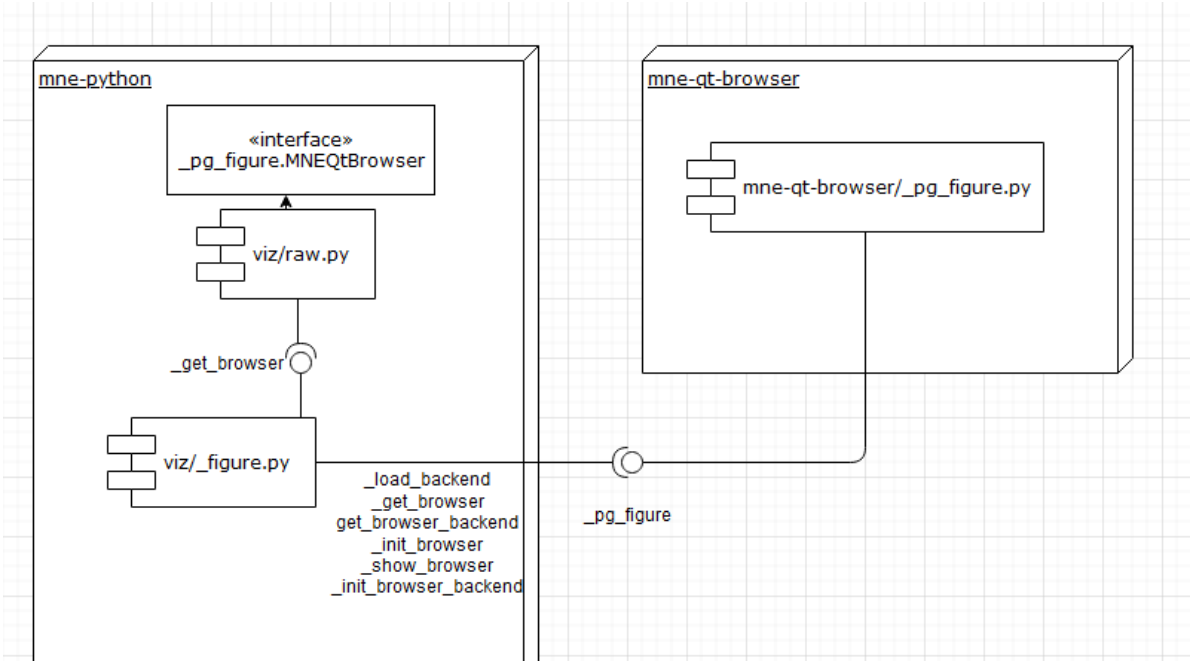
Ακολουθήθηκαν όλα τα βήματα του οδηγού συνεισφοράς, με την εξής διαφορά. Επειδή το αρχείο προς το τροποποίηση είναι στο αποθετήριο mne-qt-browser και όχι στο mne-python, εγκαταστάθηκε και αυτό για ανάπτυξη. Συγκεκριμένα, έγινε τοπικά λήψη της προσωπικής διακλάδωσης του αποθετηρίου και εγκαταστάθηκε με το ειδικό όρισμα “-e”. Αυτό, που σημαίνει editable (επεξεργάσιμος), επιτρέπει την συνεχή αποτύπωση των αποτελεσμάτων της επεξεργασίας κατά τη χρήση του λογισμικού. Με εντολές σε τερματικό:

```
git clone https://github.com/$GITHUB_USERNAME/mne-qt-browser.git
cd mne-qt-browser
conda remove --force mne-qt-browser
pip install -e .
```

Επίσης απαραίτητη ήταν και η εγκατάσταση ενός επεξεργαστή κώδικα, στην προκειμένη περίπτωση του Visual Studio Code (Website: Visual Studio Code, n.d.). Αξιοποιήθηκαν στο μέγιστο οι δυνατότητες και επεκτάσεις του σχετικά με μορφοποίηση, Python Notebooks και ενσωμάτωση των περιβαλλόντων που διαχειρίζεται το Conda για εύκολη ανάπτυξη και δοκιμή του κώδικα.

3.3.3 ΔΟΜΗ ΤΟΥ ΥΠΑΡΧΟΝΤΟΣ ΚΩΔΙΚΑ

Η συνάρτηση της βιβλιοθήκης MNE-Python που οπτικοποιεί νευροφυσιολογικά δεδομένα αποθηκευμένα σε αντικείμενα κλάσης “Raw” (Website: MNE-Python - Documentation - Raw, n.d.) είναι η plot_raw() (Website: MNE-Python - Documentation - plot_raw(), n.d.), η οποία φιλοξενείται στο mne/viz χώρο του αποθετηρίου mne-python. Με μία σειρά από εσωτερικές κλήσεις, η διαδικασία καταλήγει στο αρχείο mne_qt_browser/_pg_figure.py του αποθετηρίου mne-qt-browser (βλ. Εικόνα 3.1). Αυτό το αρχείο περιέχει τον κώδικα για τη δημιουργία ενός GUI παραθύρου, στο οποίο οπτικοποιούνται τα δεδομένα και υπάρχουν διάφορα εργαλεία για την εξυπηρέτηση του χρήστη.



Εικόνα 3. 1 Component Diagram για την κλήση της συνάρτησης plot_raw()

Η φιλοσοφία προγραμματισμού που χρησιμοποιείται τόσο στη βιβλιοθήκη MNE-Python όσο και την PyQt5 είναι ο Αντικειμενοστραφής Προγραμματισμός (Object Oriented Programming). Σε αυτή γίνεται σχεδίαση λογισμικού που περιστρέφεται γύρω από δεδομένα, αντικείμενα και τις αλληλεπιδράσεις τους. Ο αντικειμενοστραφής προγραμματισμός χαρακτηρίζεται από πολλά θετικά μεταξύ των οποίων η ανακυκλωσιμότητα, η κληρονομικότητα και ο πολυμορφισμός που επιτρέπουν την επίτευξη ευέλικτων και αρθρωτών σχεδίων (Gamma, Helm, Johnson, & Vlissides, 1994).

Στο αρχείο _pg_figure.py και στο κέντρο της υλοποίησης βρίσκεται η κλάση “MNEQtBrowser”. Αποτελεί το βασικό αντικείμενο του προγράμματος, καθώς και το βασικό παράθυρο με τα οπτικοποιημένα δεδομένα και τα διάφορα σχετικά εργαλεία σε ένα

μενού (βλ. Εικόνα 2.1). Επίσης αυτή η κλάση διαδραματίζει ρόλο ενορχηστρωτή, καλώντας λειτουργίες από τα υπόλοιπα αντικείμενα όποτε αυτό χρειάζεται, ή διατηρώντας κεντρικές λειτουργίες για να καλούνται από τα αντικείμενα αυτά. Για το σκοπό της οπτικοποίησης εντός της κλάσης χρησιμοποιείται το PyQtGraph και η αρχιτεκτονική γραφικών «θέας» (Website: PyQt5 Documentation - Graphics View Framework, n.d.) που αποτελείται από μία σκηνή (scene), μία θέα (view) και ένα παράθυρο (viewbox), με αποστολή τη αποδοτική διαχείριση και προβολή των γραφημάτων.

Όλες οι κλάσεις και τα αντικείμενα μοιράζονται μεταξύ τους, είτε δηλωμένα ρητά είτε μέσω κληρονομικότητας, την μεταβλητή "mne". Αυτή, που στην πραγματικότητα είναι ένας χώρος πολλών μεταβλητών, είναι ένας εξαιρετικά αποδοτικός τρόπος για τον διαμοιρασμό πληροφοριών μεταξύ των αντικειμένων, χωρίς τις δαπανηρές μεταφορές αλλά μέσω της πρόσβασης σε ένα κοινό σημείο αποθήκευσης. Αυτή χρησιμοποιήθηκε κατά την υλοποίηση του παρόντος έργου, αποθηκεύοντας εκεί τις νέες πληροφορίες που χρησιμεύουν σε πολλαπλά αντικείμενα, ή και μεταφέροντας ήδη υπάρχουσες που ήταν αποθηκευμένες αλλού (βλ. Παρ. 1, Γρ. 3137).

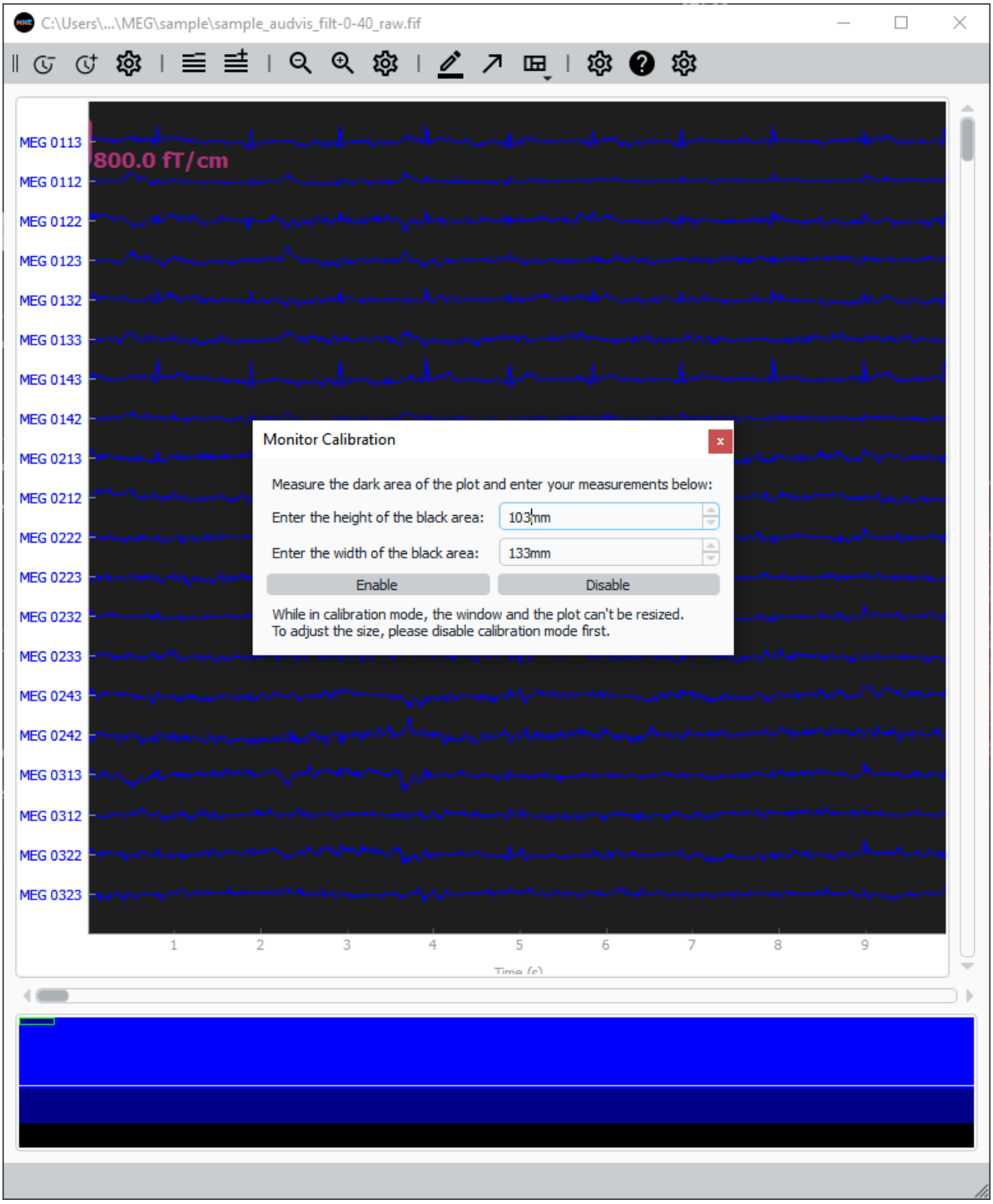
Άλλα είδη κλάσεων που χρησιμοποιούνται ποικίλουν στη λειτουργικότητα και το είδος τους. Υπάρχει μία κλάση για κάθε δευτερεύον παράθυρο που μπορεί να ανοίξει ο χρήστης, όπως ρυθμίσεις, σχολιασμούς και διάλογο βοήθειας, κλάσεις που αφορούν στοιχεία της διεπαφής όπως οι γραμμές των γραφημάτων, οι άξονες και τα κείμενά τους, οι γραμμές κύλισης και άλλες λειτουργικότητες που υπάρχουν στο παρασκήνιο. Επίσης στο αρχείο βρίσκονται και κλάσεις που υπηρετούν ως βάση για άλλες κλάσεις, επιτρέποντας την ελαχιστοποίηση του κώδικα και την επαναχρησιμοποίησή του μέσω της κληρονομικότητας.

Οι αλλαγές που υλοποιήθηκαν και αναλύονται στα επόμενα κεφάλαια αναπτύχθηκαν με βάση τις ανωτέρω φιλοσοφίες και πρακτικές. Τα νέα παράθυρα που δημιουργήθηκαν αποτελούν νέες κλάσεις, η πληροφορία μεταξύ των αντικειμένων μοιράζεται μέσω του κοινόχρηστου χώρου "mne" και ο ρόλος ενορχήστρωσης της "MNEQtBrowser" συμβάλλει στο συντονισμό και τη μετάδοση σημάτων (βλ. Παράρτημα: Υποβληθέντας Κώδικας).

3.3.3 ΠΑΡΑΘΥΡΟ ΚΑΙ ΛΕΙΤΟΥΡΓΙΚΟΤΗΤΑ ΒΑΘΜΟΝΟΜΗΣΗΣ ΟΘΟΝΗΣ

Για τη λειτουργικότητα της βαθμονόμησης της οθόνης και την παρουσία ορθών αποστάσεων επί της οθόνης ακολουθήθηκε η προσέγγιση του EDFBrowser (βλ. Εικόνα 2.4). Η ιδέα είναι να ζητηθεί από το χρήστη να μετρήσει δύο σχετικά μήκη πάνω στην οθόνη του. Αφού εισάγει τις πληροφορίες που του ζητούνται στο πρόγραμμα, ενεργοποιείται η δυνατότητα υπολογισμού όλων των σχετικών μηκών των στοιχείων του γραφήματος, συνεπώς και η πληροφορία της ευαισθησίας, δηλαδή των μονάδων ανά μονάδα μήκους. Αυτή η έννοια εφαρμόζεται και στην κάθετη κλιμάκωση πλάτους αλλά και στην οριζόντια κλιμάκωση χρόνου.

Σε αυτά τα πλαίσια αναπτύχθηκε μία καινούργια κλάση, η “CalibrationDialog” (βλ. Παρ. 1, Γρ. 1958) που αντιστοιχεί στο αντίστοιχο παράθυρο “Monitor Calibration” (βλ. Εικόνα 3.2). Όταν ο χρήστης ανοίγει με οποιονδήποτε τρόπο αυτό το παράθυρο, το διάγραμμα στο παρασκήνιο γίνεται το αντίθετο χρώμα (στο παράδειγμα από άσπρο μαύρο) κατόπιν ζητείται από τον χρήστη η μέτρηση και καταγραφή των διαστάσεών του.



Εικόνα 3. 2 Παράθυρο Βαθμονόμησης οθόνης

Στο παράθυρο υπάρχουν δύο πεδία εισόδου ακέραιου αριθμού (Spinboxes), ένα για το ύψος και ένα για το πλάτος του γραφήματος. Σαν μονάδα μέτρησης επιλέχθηκαν τα

χιλιοστόμετρα, τα οποία είναι και η πιο φυσική για ένα παράθυρο οθόνης και διαδεδομένη μονάδα στο πεδίο της ευαισθησίας. Με τα κουμπιά Ενεργοποίηση-Απενεργοποίηση (Enable-Disable) ο χρήστης εισάγει τις μετρήσεις του και ενεργοποιεί την Βαθμονομημένη Λειτουργία (calibration mode). Κατά την παραμονή στην λειτουργία αυτή, αφενός ενεργοποιούνται όλες οι επιμέρους λειτουργικότητες που αφορούν την ευαισθησία, οι οποίες θα καλυφθούν αναλυτικότερα στα επόμενα κεφάλαια, αφετέρου το μέγεθος του παραθύρου και του διαγράμματος κλειδώνουν στην τιμή που βρίσκονται, για τη διατήρηση των μετρήσεων του χρήστη. Επίσης είναι πλέον ορατή και η τιμή της ευαισθησίας του κάθε καναλιού, πέρα από το πλάτος του.

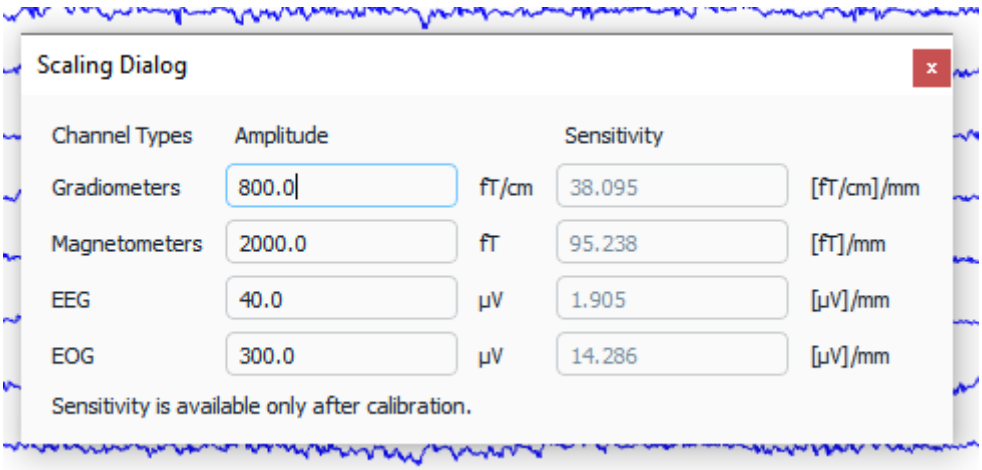
Από αυτή τη διαδικασία προκύπτουν το ύψος και το πλάτος του γραφήματος, τα οποία και αποθηκεύονται στον κοινόχρηστο χώρο “mne”. Στην πορεία πραγματοποιούνται υπολογισμοί είτε για το ύψος του κάθε καναλιού και κατόπιν την ευαισθησία του, λαμβάνοντας υπόψη τον αριθμό των καναλιών που προβάλλονται τη συγκεκριμένη στιγμή, είτε για το μήκος που καλύπτει κάθε δευτερόλεπτο στην οθόνη, λαμβάνοντας υπόψη τη συνολική διάρκεια που προβάλλεται τη συγκεκριμένη στιγμή.

Στο εγχείρημα αυτό αντιμετωπίσθηκαν ορισμένες δυσκολίες και προκλήσεις. Αφενός διαπιστώθηκε πως η αλλαγή του μεγέθους του παραθύρου άλλαζε τις διαστάσεις του γραφήματος, καθιστώντας τις πληροφορίες σχετικές με την ευαισθησία άκυρες. Η διόρθωση που υλοποιήθηκε ήταν το «κλείδωμα» του μεγέθους του παραθύρου για τη διάρκεια που ο χρήστης βρίσκεται σε Βαθμονομημένη Λειτουργία. Υπάρχει σύντομο κείμενο για σχετική ενημέρωση στο κάτω μέρος του παραθύρου, προτρέποντας το χρήστη να απενεργοποιήσει πρώτα τη λειτουργία για να προβεί σε αλλαγή του μεγέθους.

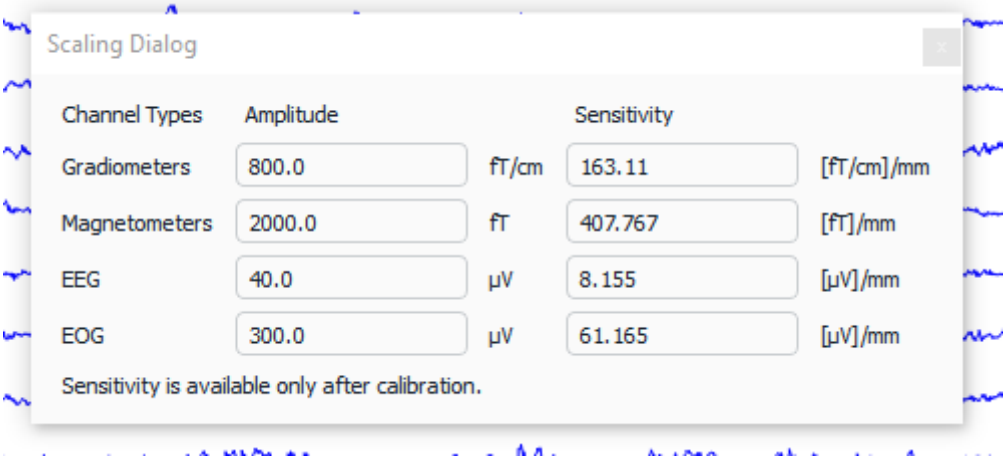
Η άλλη πρόκληση ήταν η επιλογή της διαδικασίας βαθμονόμησης και του αντικειμένου το οποίο θα υποβληθεί σε μέτρηση από το χρήστη. Το συμπέρασμα που εξήχθη είναι πως τα πραγματικά χρήσιμα μεγέθη για τους υπολογισμούς της ευαισθησίας είναι οι διαστάσεις του γραφήματος. Αποφασίστηκε λοιπόν η βαθμονόμηση να γίνει απευθείας επί τούτου και όχι μέσω άλλων γραμμών, όπως στο EDFBrowser (βλ. Εικόνα 2.4), γεγονός που καθιστά και την ανάπτυξη αλλά και τους αριθμητικούς υπολογισμούς ευκολότερους.

3.3.4 ΠΑΡΑΘΥΡΟ ΚΑΙ ΛΕΙΤΟΥΡΓΙΚΟΤΗΤΑ ΚΛΙΜΑΚΩΣΗΣ ΠΛΑΤΟΥΣ

Η επόμενη λειτουργικότητα που υλοποιήθηκε αφορά την κλιμάκωση του πλάτους των καναλιών. Στα πλαίσιά της αναπτύχθηκε η κλάση “ScalingDialog” (βλ. Παρ. 1, Γρ. 1958), που αντιστοιχεί στο παράθυρο “Scaling Dialog” (βλ. Εικόνα 3.3). Σε αυτό ο χρήστης μπορεί να πραγματοποιήσει επισκόπηση αλλά και επεξεργασία των πλατών του κάθε είδους καναλιού ξεχωριστά, αλλά και της ευαισθησίας τους, δηλαδή των μονάδων πλάτους ανά χιλιοστόμετρο. Υπενθυμίζεται πως οι λειτουργίες ευαισθησίας είναι απενεργοποιημένες έως ότου ο χρήστης κάνει βαθμονόμηση οθόνης και για όσο είναι ανενεργή αυτή η λειτουργία (βλ. Εικόνα 3.4).



Εικόνα 3. 3 Παράθυρο κλιμάκωσης πλάτους - Εκτός Βαθμονομημένης Λειτουργίας



Εικόνα 3. 4 Παράθυρο κλιμάκωσης πλάτους - Εντός Βαθμονομημένης Λειτουργίας

Στο παράθυρο αυτό υπάρχουν τρεις στήλες. Στη πρώτη στήλη φαίνονται τα ονόματα για κάθε είδος καναλιού που υπάρχει στο αρχείο που προβάλλεται, το οποίο λαμβάνεται αυτόματα μέσω της συνάρτησης "_handle_default()" (βλ. Παρ. 1, Γρ. 42, 1958). Στη δεύτερη στήλη φαίνονται συγκεντρωμένα τα πλάτη των καναλιών (Amplitude), ενώ στην τρίτη στήλη φαίνονται οι αντίστοιχες ευαισθησίες (Sensitivity). Χρησιμοποιήθηκαν πεδία κειμένου (Textboxes) για τη φιλοξενία των τιμών, καθώς είναι ορατές και οι εκάστοτε μονάδες μέτρησης στα πλάτη και οι ίδιες μονάδες ανά χιλιοστόμετρο στις ευαισθησίες. Για τα πεδία αυτά χρησιμοποιήθηκε μάσκα Κανονικής Έκφρασης (Regular Expression), που διασφαλίζει ότι η είσοδος του χρήστη είναι είτε δεκαδικός αριθμός, είτε χρησιμοποιεί επιστημονική σημειογραφία.

Η βιβλιοθήκη μέχρι τώρα υποστήριζε μόνο την ταυτόχρονη κλιμάκωση πλάτους όλων των καναλιών, και όχι επιμέρους ειδών. Αυτό ήταν δυνατό με την υπάρχουσα μεταβλητή πραγματικού αριθμού "mne.scale_factor", η οποία φιλοξενούσε τον συντελεστή κλίμακας του γραφήματος για όλα τα κανάλια. Για να προστεθούν οι νέες λειτουργίες, αυτή αντικαταστάθηκε με ένα λεξικό (dictionary), ονομαζόμενο "mne.scale_factors" το οποίο

αρχικοποιείται με μία εγγραφή για κάθε είδος καναλιού το οποίο εντοπίζεται στο αρχείο που προβάλλεται (βλ. Παρ. 1, Γρ. 3137), προκειμένου να υπάρχει η ζητούμενη λειτουργία. Ταυτόχρονα με αυτή την αλλαγή, χρειάστηκε να διαμορφωθεί κάθε σημείο του αρχείου που αναφερόταν σε αυτή την μεταβλητή, ώστε να πραγματοποιείται πλέον επανάληψη επί των ειδών καναλιών που υπάρχουν (βλ. Παρ. 1, Γρ. 392, 1618, 3137, 3803, 4030, 4354).

Το πλάτος για κάθε είδος καναλιού υπολογίζεται ως εξής:

$$amplitude = \frac{scaler * norms\_dict[ch\_type]}{scale\_factors[ch\_type]}$$

όπου scaler είναι ίσο με 2 αν το διάγραμμα είναι σε μορφή πεταλούδας, αλλιώς είναι 1, norms_dict είναι ένα λεξικό που περιέχει τις βασικές νόρμες κάθε είδους καναλιού, scale_factors είναι το λεξικό που περιέχει τους συντελεστές κλίμακας κάθε είδους καναλιού, και ch_type είναι το είδος καναλιού.

Η ευαισθησία του καναλιού είναι ουσιαστικά το πλάτος του διαιρεμένο με το μήκος του ύψος του καναλιού. Λαμβάνοντας υπόψη πως στο γράφημα χρησιμοποιείται ένα επιπλέον κενό κανάλι για λόγους επένδυσης του, η ευαισθησία υπολογίζεται ως εξής:

$$sensitivity = \frac{amplitude}{channel\_height} = \frac{amplitude * (n\_channels + 1)}{height}$$

όπου amplitude είναι το πλάτος που υπολογίσθηκε πριν, n_channels είναι ο αριθμός των καναλιών που προβάλλονται αυτή τη στιγμή και height είναι το συνολικό ύψος του διαγράμματος, που έχει μετρήσει ο χρήστης στη διαδικασία βαθμονόμησης.

Η κλάση αυτή είναι εξοπλισμένη με κατάλληλες μεθόδους για να διαχειρίζονται την αλλαγή κάποιας τιμής από τον χρήστη και τον υπολογισμό του νέου συντελεστή κλίμακας. Συγκεκριμένα για τα πλάτη, και δεδομένου ότι οι αλλαγές στον συντελεστή κλίμακας είναι αντιστρόφως ανάλογες με τις αλλαγές του πλάτους, προκύπτει ότι:

$$new\_scale\_factor = old\_scale\_factor * \frac{old\_amplitude}{new\_amplitude}$$
$$= \frac{scaler * norms\_dict[ch\_type]}{new\_amplitude}$$

όπου new_scale_factor είναι η νέα τιμή του συντελεστή κλίμακας για το συγκεκριμένο κανάλι, old_amplitude είναι το πλάτος που είχε πριν την αλλαγή και new_amplitude η νέα τιμή που εισήγαγε ο χρήστης.

Διαφορετικά, αν η αλλαγή αφορά τις ευαισθησίες, οι υπολογισμοί είναι οι ίδιοι αλλά σύμφωνα με τον ορισμό της ευαισθησίας, με κάποιους επιπρόσθετους συντελεστές. Δηλαδή:

$$new\_scale\_factor = \frac{scaler * norms\_dict[ch\_type] * (n\_channels + 1)}{new\_sensitivity * height}$$

όπου new_sensitivity είναι η τιμή που εισήγαγε ο χρήστης για την ευαισθησία του συγκεκριμένου είδους καναλιού.

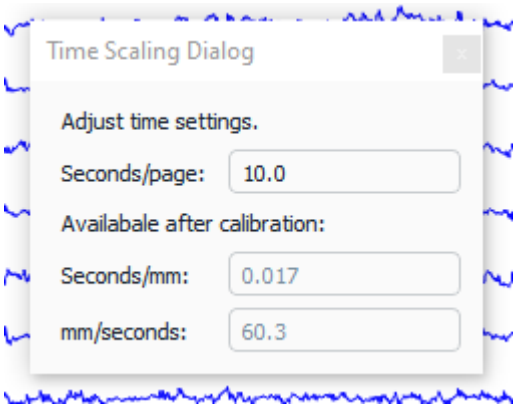
Αφού γίνει κάποια αλλαγή από το χρήστη και υπολογισθούν οι νέοι συντελεστές κλίμακας, πρέπει να γίνουν και οι απαραίτητες αλλαγές. Η εφαρμογή των νέων τιμών γίνεται στο λεξικό των συντελεστών κλίμακας, ακολουθεί περικοπή (clipping) των δεδομένων αν είναι αναγκαία, κλιμάκωση των γραμμών και ανανέωση των σχετικών κειμένων με τις τιμές πλάτους και ευαισθησίας, είτε στο γράφημα, είτε στο ανοιχτό παράθυρο κλιμάκωσης (βλ. Παρ. 1, Γρ. 1958).

Προκλήσεις αντιμετωπίσθηκαν στην ανάπτυξη και αυτής της λειτουργίας. Σχεδιαστική ανάγκη που αποκαλύφθηκε εκ των υστέρων, ήταν η ανάγκη για δυναμική ανανέωση των τιμών των πεδίων σε πολλές διαφορετικές περιπτώσεις. Για το λόγο αυτό συστάθηκε και συγκεκριμένη μέθοδος "_update_boxes()" (βλ. Παρ. 1, Γρ. 1958), όπως και στα υπόλοιπα παράθυρα. Κάθε φορά λοιπόν που αλλάζει κάτι σχετικό είτε με το πλάτος είτε με την ευαισθησία, γίνεται κλήση της μεθόδου αυτής για να γίνει και η σχετική ανανέωση των τιμών στο παράθυρο, εφόσον αυτό είναι ανοιχτό (βλ. Παρ. 1, Γρ. 2080, 3815, 3933, 4102, 4624). Παραδείγματα αποτελούν τα υπάρχοντα κουμπιά μεγέθυνσης-σμίκρυνσης (Zoom in-Zoom out), ή η αλλαγή του προβαλλόμενου αριθμού καναλιών, που επηρεάζει την ευαισθησία.

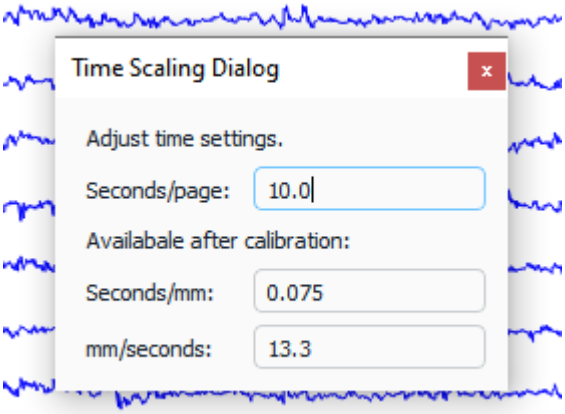
Σημαντική δυσκολία επίσης αντιμετωπίσθηκε στο κομμάτι της περικοπής (βλ. Παρ. 1, Γρ. 4354). Αυτή προηγουμένως γινόταν πολλαπλασιάζοντας κάθε τιμή δεδομένου με τον, προηγουμένως οικουμενικό, συντελεστή κλίμακας. Αυτό δεν ήταν πλέον δυνατόν, λόγω του ότι ο συντελεστής κλίμακας έγινε λεξικό και το αποτέλεσμα εξαρτάται στο εξής από το είδος του εκάστοτε καναλιού. Το πρόβλημα λύθηκε δημιουργώντας έναν κατάλληλο NumPy πίνακα, ώστε να γίνει σωστά από τη βιβλιοθήκη η ερμηνεία του σχήματος και η πράξη του πολλαπλασιασμού όλων των δεδομένων με τους αντίστοιχους συντελεστές τους. Αυτού του είδους η διανυσματική προσέγγιση ήταν απαραίτητη, διότι κατά αυτόν τον τρόπο λειτουργεί βέλτιστα η βιβλιοθήκη NumPy. Σε μία δοκιμή με κλασσική επανάληψη "for loop", το πρόγραμμα έγινε τόσο αργό που δεν ήταν πλέον εύχρηστο, καθώς κάθε βήμα κύλισης και οι απαραίτητοι υπολογισμοί διαρκούσαν περίπου δύο δευτερόλεπτα, σε ένα ενδεικτικό αρχείο μέτριου μεγέθους.

3.3.5 ΠΑΡΑΘΥΡΟ ΚΑΙ ΛΕΙΤΟΥΡΓΙΚΟΤΗΤΑ ΚΛΙΜΑΚΩΣΗΣ ΧΡΟΝΟΥ

Η τελευταία βασική λειτουργικότητα που προστέθηκε ήταν αυτή που αφορά την κλιμάκωση του χρόνου. Για την ανάγκη αυτή δημιουργήθηκε μία νέα κλάση "TimeScalingDialog" (βλ. Παρ. 1, Γρ. 1958), που αντιστοιχεί στο παράθυρο "Time Scaling Dialog" (βλ. Εικόνα 3.5). Αυτή η νέα κλάση παρουσιάζει πολλά κοινά χαρακτηριστικά με την προηγούμενη, αφού η πρώτη αφορά την κάθετη κλιμάκωση και η δεύτερη την οριζόντια. Παρά ταύτα, η καθεμία είχε τις δικιές της ιδιαιτερότητες και ανάγκες.



Εικόνα 3. 5 Παράθυρο κλιμάκωσης χρόνου - Εκτός Βαθμονομημένης Λειτουργίας



Εικόνα 3. 6 Παράθυρο κλιμάκωσης χρόνου - Εντός Βαθμονομημένης Λειτουργίας

Συγκεκριμένα, παρά τις ομοιότητές τους, για την κλιμάκωση χρόνου αφενός έγινε η σύμβαση (σε κοινή συνεννόηση με μέλη της κοινότητας MNE-Python) πως δεν έχει νόημα η διαφορετική χρονική κλιμάκωση διαφορετικών ειδών καναλιών. Αντιθέτως έχει νόημα όλα τα κανάλια να έχουν τον ίδιο χρονικό άξονα, ούτως ώστε να αναδεικνύονται οι ενδεχόμενες αλληλεξαρτήσεις μεταξύ τους. Επίσης, η σχετική ποσότητα που τροποποιείται δεν είναι πλέον οι συντελεστές κλίμακας, αλλά η συνολική διάρκεια του αρχείου που προβάλλεται εκείνη τη στιγμή, για να πραγματοποιηθούν οι εκάστοτε ανάγκες. Αυτή είναι αποθηκευμένη στην μεταβλητή πραγματικού αριθμού “mne.duration”, η οποία δεν χρειάστηκε κάποια μετατροπή σε αντίθεση με τους συντελεστές κλίμακας.

Στο παράθυρο αυτό υπάρχουν τρία πεδία κειμένου (Textboxes). Το πρώτο περιέχει την συνολική διάρκεια που προβάλλεται εκείνη τη στιγμή σε δευτερόλεπτα (Seconds/page), το δεύτερο τα δευτερόλεπτα ανά χιλιοστόμετρο (Seconds/mm) και το τρίτο τα χιλιοστόμετρα ανά δευτερόλεπτο (mm/seconds). Το δεύτερο και το τρίτο πεδίο αφορούν αποστάσεις πάνω στην οθόνη, και συνεπώς είναι απενεργοποιημένα έως ότου ο χρήστης κάνει βαθμονόμηση οθόνης και για όσο είναι ανενεργή αυτή η λειτουργία (βλ. Εικόνα 3.6). Υπάρχει και σχετική ενημέρωση με κείμενο μέσα στο παράθυρο. Για τα πεδία αυτά χρησιμοποιήθηκε η ίδια μάσκα Κανονικής Έκφρασης που προαναφέρθηκε, που διασφαλίζει

ότι η είσοδος του χρήστη είναι είτε δεκαδικός αριθμός, είτε χρησιμοποιεί επιστημονική σημειογραφία.

Για τις αλλαγές μετά την επεξεργασία από το χρήστη, δημιουργήθηκε κατάλληλη μέθοδος. Το μόνο σχετικό μέγεθος που επιδέχεται τροποποίησης είναι η διάρκεια που προβάλλεται και για αυτό το σκοπό πρέπει σε κάθε περίπτωση να βρεθεί το απαιτούμενο ποσοστιαίο βήμα της αλλαγής της. Αυτό είναι απλό και υπολογίζεται παρακάτω.

Για το πρώτο κουτί, της συνολικής διάρκειας:

$$new\_duration = old\_duration + step * old\_duration \Leftrightarrow$$

$$step = new\_duration / old\_duration - 1$$

Για το δεύτερο κουτί, δευτερόλεπτα ανά χιλιοστόμετρο:

$$step = new\_duration / old\_duration - 1 \Leftrightarrow$$

$$step = new\_seconds\_mm * width / old\_duration - 1$$

Τέλος, για το τρίτο κουτί, χιλιοστόμετρα ανά δευτερόλεπτο:

$$step = new\_duration / old\_duration - 1 \Leftrightarrow$$

$$step = width / (old\_duration * new\_mm\_seconds) - 1$$

Στους παραπάνω υπολογισμούς, step είναι το ποσοστιαίο βήμα της αλλαγής της διάρκειας προς υπολογισμό, old_duration και new_duration είναι η παλιά και η νέα διάρκεια αντίστοιχα, width είναι το πλάτος του γραφήματος που εισήγαγε ο χρήστης στη διαδικασία της βαθμονόμησης, new_seconds_mm είναι η νέα τιμή που εισήχθη στο δεύτερο πεδίο και new_mm_seconds είναι η νέα τιμή που εισήχθη στο τρίτο πεδίο. Αφού υπολογισθεί το step, καλείται η εσωτερική μέθοδος “change_duration()” της κλάσης “MNEQtBrowser” η οποία αλλάζει την διάρκεια που προβάλλεται στο διάγραμμα κατάλληλα.

Παρόμοια με το προηγούμενο παράθυρο, υπήρχαν και εδώ κάποια σημεία που χρειάζονταν προσοχή. Αφενός δημιουργήθηκε κατά τα γνωστά μία μέθοδος “_update_boxes()” η οποία καλείται όχι μόνο στην αρχικοποίηση αλλά και σε διάφορες ενέργειες του χρήστη που οδηγούν στην αλλαγή των τιμών των πεδίων του παραθύρου, όσο αυτό είναι ανοιχτό (βλ. Παρ. 1, Γρ. 3911, 4624). Αυτές είναι η ενεργοποίηση της Λειτουργίας Βαθμονόμησης με νέες μετρήσεις από τον χρήστη, ή η αλλαγή της προβαλλόμενης διάρκειας μέσω κάποιου άλλου τρόπου, όπως τα προϋπάρχοντα κουμπιά του μενού. Χωρίς αυτή τη λειτουργικότητα, ο χρήστης θα μπορούσε να οδηγηθεί σε κάποια κατάσταση όπου τα πεδία θα ανέφεραν λάθος τιμές, κάτι που τώρα δεν είναι δυνατό.

3.3.6 ΔΟΚΙΜΕΣ ΚΩΔΙΚΑ

Αφού ολοκληρωθούν όλες οι αλλαγές των νέων χαρακτηριστικών, ο κώδικας πρέπει να υποβληθεί σε δοκιμές (tests). Εφόσον η φύση των λειτουργιών που προστέθηκαν είναι νέα για τη βιβλιοθήκη, η ήδη υπάρχουσα σουίτα δοκιμών δεν επαρκεί. Παράλληλα λοιπόν με την ανάπτυξη νέου κώδικα, είναι απαραίτητη και η ανάπτυξη νέων δοκιμών (Crispin, 2006).

Για την υλοποίηση των δοκιμών γίνεται χρήση της βιβλιοθήκης Pytest, έκδοσης 7.4.3. Η εν λόγω βιβλιοθήκη προσφέρει μία ποικιλία από εργαλεία για δοκιμές κώδικα, στο προσιτό περιβάλλον της Python. Οι δοκιμές κωδικοποιούνται ως συναρτήσεις, οι οποίες αξιοποιούν τις αλληλεπιδράσεις της βιβλιοθήκης για να εκτελέσουν το πρόγραμμα με δυναμικό χαρακτήρα, εξετάζοντας πολύ πιο εξουθενωτικά το έργο από έναν αντίστοιχο ανθρώπινο παράγοντα.

Η αξία των δοκιμών έγκειται προφανώς στην εγγύηση λειτουργικού και αποδοτικού κώδικα. Οι επιγνώσεις που ενδεχομένως δημιουργηθούν μπορεί να αφορούν την σωστή λειτουργία, εντοπίζοντας λάθη, ή να καταγράφουν μετρικές σχετικά με την απόδοση, όπως ο χρόνος εκτέλεσης ή οι πόροι που χρησιμοποιήθηκαν. Περαιτέρω όμως, η ανάπτυξη δοκιμών παράλληλα με την ανάπτυξη χαρακτηριστικών βοηθάει τον προγραμματιστή να αντιληφθεί ζητήματα πριν αυτά προκύψουν, διευκολύνοντας έτσι τη διαδικασία σχεδιασμού και αποτρέποντας αναδιαμορφώσεις στο μέλλον (Crispin, 2006).

Στην προκειμένη περίπτωση, ακολουθήθηκε το μοντέλο που ήδη υπήρχε. Στο αποθετήριο mne-qt-browser υπάρχει ο φάκελος tests με τα αρχεία test_pg_specific.py και test_speed.py. Το πρώτο εκτελεί δοκιμές για όλα τα στοιχεία της αρχιτεκτονικής, συνεπώς χρειάστηκε να τροποποιηθεί κατάλληλα. Το δεύτερο εκτελεί δοκιμές για την ταχύτητα επεξεργασίας των δεδομένων υπό διαφορετικές ρυθμίσεις καταγράφοντας αποτελέσματα, και δεν χρειάστηκε κάποια αλλαγή, καθώς δεν υπήρχε επικάλυψη των νέων στοιχείων με τις συγκεκριμένες δοκιμές. Παρά ταύτα, κατά την εκτέλεση της σουίτας εκτελέστηκαν όλες οι δοκιμές, και οι νέες αλλά και αυτές που δεν υπέστησαν κάποια αλλαγή.

Στο αρχείο test_pg_specific.py υπάρχει ένα σύνολο συναρτήσεων, καθεμία επιφορτισμένη με την ενδελεχή δοκιμή κάποιου στοιχείου ή και ενέργειας της διεπαφής. Δεδομένου ότι προστέθηκαν νέα στοιχεία στο πρόγραμμα, τα τρία προαναφερθέντα παράθυρα και οι λειτουργικότητές τους, δημιουργήθηκε μία νέα συνάρτηση, η “test_sensitivity_etc()” (βλ. Παρ. 2, Γρ. 22) , υπεύθυνη για τη δοκιμή τους. Γίνεται συνειδητή επιλογή της μίας και όχι περισσότερων συναρτήσεων καθώς, παρόλο που τα τρία στοιχεία είναι διακριτά μεταξύ τους, η λειτουργία τους συνδέεται. Σχεδόν οι μισές λειτουργίες του Παραθύρου Κλιμάκωσης Πλάτους και του Παραθύρου Κλιμάκωσης Χρόνου ενεργοποιούνται μόνο μέσω του Παραθύρου Βαθμονόμησης Οθόνης, οπότε είναι φυσιολογικό ή δοκιμή τους να γίνει ταυτόχρονα.

Ακολουθώντας το πρότυπο των δοκιμών στη βιβλιοθήκη MNE-Python, η συνάρτηση εκτελεί τους εξής ελέγχους:

1. Έλεγχος της λειτουργικότητας των παραθύρων: Αρχικά καλείται το πρόγραμμα και εμφανίζεται η αρχική διεπαφή. Αφού διαπιστώσει πως το παράθυρο είναι ενεργοποιημένο, η συνάρτηση προχωράει στο να ανοίξει και να κλείσει εναλλάξ λίγες

φορές τα προαναφερθέντα τρία παράθυρα, όσο ελέγχει συνέχεια τον αριθμό των ενεργών παραθύρων για να εντοπίσει τυχόν ανωμαλίες.

2. Έλεγχος της λειτουργικότητας της βαθμονόμησης: Στο Παράθυρο Βαθμονόμησης Οθόνης εισάγονται τιμές στα πεδία του ύψους και του πλάτους και γίνεται έλεγχος για την σωστή μετάδοση και αποθήκευσή τους στον κοινόχρηστο χώρο “mne”. Ενεργοποιώντας την λειτουργία βαθμονόμησης γίνεται εφικτός ο πλήρης έλεγχος των επόμενων δύο παραθύρων

3. Έλεγχος της λειτουργικότητας κλιμάκωσης πλάτους: Στο Παράθυρο Κλιμάκωσης Πλάτους εισάγονται διάφορες τιμές σε όλα τα διαθέσιμα πεδία, δηλαδή στο πλάτος και στην ευαισθησία όλων των διαθέσιμων ειδών καναλιών. Σε αυτή τη διάρκεια γίνεται παράλληλος έλεγχος πως οι συντελεστές κλίμακας τροποποιούνται βάσει των σωστών υπολογισμών και αποκτούν τις επιθυμητές τιμές.

4. Έλεγχος της λειτουργικότητας κλιμάκωσης χρόνου: Στο Παράθυρο Κλιμάκωσης Χρόνου εισάγονται διάφορες τιμές σε όλα τα διαθέσιμα πεδία. Κάθε αλλαγή πρέπει να αντικατοπτρίζεται στην αντίστοιχη σωστή αλλαγή της τιμής της προβαλλόμενης διάρκειας, το οποίο και ελέγχεται κατά τις εισαγωγές τιμών στα πεδία του παραθύρου.

5. Τέλος: Τα παράθυρα κλείνουν και γίνεται έλεγχος πως όντως έχουν απενεργοποιηθεί και το πρόγραμμα είναι σε υγιή κατάσταση.

Και κατά τη διάρκεια, για τον εντοπισμό λαθών, αλλά και στο τέλος της ανάπτυξης εκτελείται η σουίτα δοκιμών. Μετά την εκτέλεση, και τη διαπίστωση της επιτυχίας των δοκιμών (βλ. Εικόνα 3.7), η αξιοπιστία και αποδοτικότητα των νέων λειτουργιών είναι πιο πιθανή. Κατά την υποβολή στο git, ο κώδικας υποβάλλεται και σε επιπρόσθετες δοκιμές και βελτιώσεις, αφενός για την μορφοποίησή του αλλά και για την λειτουργία του σε διάφορα οικοσυστήματα. Έτσι διασφαλίζονται και οι συμφωνημένες κατευθυντήριες γραμμές περί μορφοποίησης, αλλά και οι δοκιμές σε περιβάλλοντα τα οποία ο προγραμματιστής δεν είχε πρόσβαση, παρέχοντας μία επιπλέον δικλείδα ασφαλείας και ελέγχου.

```
============================================= test session starts =============================================
platform win32 -- Python 3.11.6, pytest-7.4.3, pluggy-1.3.0
PyQt5 5.15.9 -- Qt runtime 5.15.8 -- Qt compiled 5.15.8
rootdir: D:\Repos\mne-qt-browser-dev
configfile: pyproject.toml
plugins: anyio-4.0.0, cov-4.1.0, harvest-1.10.4, qt-4.2.0, timeout-2.2.0
collected 10 items

mne_qt_browser\tests\test_pg_specific.py ..........                                                  [100%]

------------------------- generated xml file: D:\Repos\mne-qt-browser-dev\junit-results.xml -------------------------
============================================= slowest 10 durations ============================================
9.36s call     mne_qt_browser/tests/test_pg_specific.py::test_pg_settings_dialog
7.10s call     mne_qt_browser/tests/test_pg_specific.py::test_pg_toolbar_time_plus_minus
4.65s call     mne_qt_browser/tests/test_pg_specific.py::test_sensitivity_etc
4.08s call     mne_qt_browser/tests/test_pg_specific.py::test_pg_toolbar_zoom
4.06s call     mne_qt_browser/tests/test_pg_specific.py::test_pg_toolbar_channels_plus_minus
3.64s call     mne_qt_browser/tests/test_pg_specific.py::test_pg_toolbar_actions
3.37s call     mne_qt_browser/tests/test_pg_specific.py::test_pg_help_dialog
2.30s call     mne_qt_browser/tests/test_pg_specific.py::test_pg_toolbar_annotations
1.63s call     mne_qt_browser/tests/test_pg_specific.py::test_annotations_interactions
1.40s call     mne_qt_browser/tests/test_pg_specific.py::test_ch_specific_annot
============================================== 10 passed in 45.55s ============================================
```

Εικόνα 3. 7 Αποτελέσματα Δοκιμών

3.3.7 ΤΕΚΜΗΡΙΩΣΗ ΚΩΔΙΚΑ

Κάθε σωστή διαδικασία ανάπτυξης λογισμικού περιλαμβάνει ενδελεχή τεκμηρίωση (documentation) του κώδικα. Αυτό δεν συμβαίνει μόνο σε επίπεδο ολοκληρωμένου μεγάλου έργου, όπου η τεκμηρίωση υπάρχει διαθέσιμη στο διαδίκτυο ξεχωριστά από την βάση κώδικα, αλλά γίνεται και σε επίπεδο προγραμματιστή και μικρών αλλαγών, συνήθως με τη μορφή σχολιασμών (comments).

Υπάρχει πληθώρα λόγων που αυτό είναι απαραίτητο. Πέραν των κατευθυντήριων γραμμών για την ονοματοδοσία στοιχείων στον κώδικα, χρειάζεται εκτενής περιγραφή προκειμένου αυτός να γίνεται κατανοητός σε τρίτους προγραμματιστές, πέρα από τον δημιουργό. Ένας κώδικας που είναι ευανάγνωστος και κατανοήσιμος και από άλλους εξασφαλίζει την ασφάλεια, επεκτασιμότητα και συντηρησιμότητά του (Khamis, Rilling, & Witte, 2013). Ιδιαίτερα στα εργαλεία ανοιχτού κώδικα παρουσιάζεται αυτή η ανάγκη, πολύ περισσότερο αφού ολόκληρη η κοινότητα εμπλέκεται στην διαδικασία της ανάπτυξης.

Η διεύρυνση, ή και η εκμετάλλευση ενός εκτενώς τεκμηριωμένου κώδικα είναι πολύ εύκολη σε σχέση με έναν κώδικα χωρίς σχολιασμούς. Ο ενδιαφερόμενος προγραμματιστής μπορεί να δει άμεσα τις τεχνικές προδιαγραφές κάθε στοιχείου, όπως για παράδειγμα τις εισόδους και εξόδους κάποιας συνάρτησης και τους υπολογισμούς που πραγματοποιούνται. Επίσης, οι εκτενείς σχολιασμοί συμβάλλουν καθοριστικά στην διαδικασία αξιολόγησης και ελέγχου του κώδικα, κατά την υποβολή αιτήματος συγχώνευσης, δίνοντας επίγνωση στους αναθεωρητές για τις προθέσεις του υποβολέα και τις υλοποιήσεις του.

Με βάση όλα τα παραπάνω η προκειμένη βελτίωση της βιβλιοθήκης MNE-Python είναι πλήρως τεκμηριωμένη. Σε όλη την έκταση του υποβληθέντα κώδικα βρίσκονται σχολιασμοί γραμμών με επεξηγήσεις για το λόγο ύπαρξης των νέων στοιχείων, την αρχιτεκτονική, τις διαδικασίες και υπολογισμούς που εκτελούνται, αιτιολόγηση για αποφάσεις που ελήφθησαν για τη σχεδίαση, προτάσεις και προβληματισμούς.

Πέρα από αυτούς τους σύντομους σχολιασμούς, περιλαμβάνονται και πιο εκτενείς σχολιασμοί σε ορισμούς βασικών οντοτήτων, οι οποίοι μπορούν να προσπελαστούν με αυτοματισμούς για την παραγωγή τεκμηρίωσης (docstrings) (βλ. Εικόνες 3.8, 3.9). Σε αυτούς περιγράφεται με μεγαλύτερη ακρίβεια η ταυτότητα, λειτουργικότητα και τεχνικές προδιαγραφές της συγκεκριμένης οντότητας, με σκοπό αυτή να είναι προσβάσιμη σε τρίτους προγραμματιστές ή και χρήστες της εφαρμογής μέσω της αυτοματοποιημένης τεκμηρίωσης.

```
class TimeScalingDialog(_BaseDialog):
    """
    Dialog for the purposes of time scaling.

    After calibration, the distance-related options become available.
    Relevant measures:
    - duration: The total duration displayed in seconds.
    - width: The plot width in millimeters.
    Derived measures, and editable textboxes:
    - spp: Seconds per page. spp = duration
    - spmm: Seconds per millimeter. spmm = duration/width
    - mmps: Millimeters per second. mmps = width/duration
    """

    def __init__(self, main, title="Time Scaling Dialog", **kwargs):
        super().__init__(main, title=title, **kwargs)
```

Εικόνα 3. 8 Σχολιασμός της κλάσης "TimeScalingDialog"

```
def _edited(self, box):
    """
    Determine the percentile step, with which duration is scaled.

    - Seconds per page:
    new_duration = old_duration + step*old_duration <=>
    step = new_duration/old_duration - 1

    - Seconds per millimeter:
    step = new_duration/old_duration - 1 <=>
    step = new_spmm * width / old_duration - 1

    - Millimeters per second:
    step = new_duration/old_duration - 1 <=>
    step = width / (old_duration * new_mmps) - 1
    """
```

Εικόνα 3. 9 Σχολιασμός της μεθόδου "_edited()"

4. ΕΠΙΛΟΓΟΣ

4.1 ΣΥΜΠΕΡΑΣΜΑΤΑ ΚΑΙ ΑΠΟΤΕΛΕΣΜΑΤΑ

Η ολοκλήρωση του παρόντος έργου σηματοδοτεί σημαντική πρόοδο στη λειτουργικότητα και τη χρηστικότητα του λογισμικού MNE-Python για την ανάλυση νευροφυσιολογικών δεδομένων. Κατά τη διάρκεια του έργου, εφαρμόστηκαν διάφορες βελτιώσεις για την αντιμετώπιση συγκεκριμένων αναγκών και τη βελτίωση της συνολικής εμπειρίας του χρήστη. Αυτές οι βελτιώσεις επικεντρώθηκαν κυρίως στην τελειοποίηση της λειτουργίας κλιμάκωσης, στοχεύοντας σε τρεις βασικούς τομείς: κλιμάκωση συγκεκριμένων ειδών καναλιών, δυνατότητα προσαρμογής του συντελεστή κλίμακας και διασφάλιση της ακριβούς αναπαράστασης των πραγματικών μηκών στην οθόνη.

Κάθε βελτίωση υλοποιήθηκε σχολαστικά, λαμβάνοντας υπόψη τις απαιτήσεις των χρηστών, την τεχνική εφικτότητα και την τήρηση των κατευθυντήριων γραμμών συνεισφοράς και των βέλτιστων πρακτικών ανάπτυξης λογισμικού. Επιτρέποντας στους χρήστες να κλιμακώνουν συγκεκριμένους τύπους καναλιών και να ρυθμίζουν χωρίς περιορισμούς τους συντελεστές κλίμακας, το λογισμικό παρέχει πλέον μεγαλύτερη ευελιξία και έλεγχο στην απεικόνιση των δεδομένων, διευκολύνοντας το έργο των χρηστών. Επιπλέον, η ακριβής αναπαράσταση των πραγματικών μηκών στην οθόνη, δηλαδή η ευαισθησία, ενισχύει την ερμηνευσιμότητα των εμφανιζόμενων δεδομένων, και την αναγνωσιμότητα από τον ανθρώπινο αναλυτή ο οποίος έχει εκπαιδευθεί κατά αυτόν τον τρόπο.

Μέσα από εκτενείς δοκιμές, ανθρώπινες και αυτοματοποιήσιμες, διαπιστώθηκε η επιτυχής υλοποίηση των ζητούμενων λειτουργιών. Με τη χρήση της εφαρμογής φαίνεται εποπτικά η ευκολότερη και ακριβέστερη διαχείριση της κλιμάκωσης των δεδομένων, καθώς και η ορθότητα των λειτουργιών ευαισθησίας, με τις τιμές των μηκών να ανταποκρίνονται στην πραγματικότητα. Οι αυτοματοποιημένες δοκιμές εξασφαλίζουν την ομαλή λειτουργία της εφαρμογής και σε άλλες περιπτώσεις, καθώς και τον διαρκή έλεγχό της, δεδομένου ότι το περιβάλλον αλλάζει συνεχώς.

Το έργο αυτό κατέστη δυνατό χάρη στις προσπάθειες συνεργασίας και την ενεργό εμπλοκή με την κοινότητα της MNE. Η συνεχής ανατροφοδότηση και καθοδήγηση από τα μέλη της κοινότητας διαδραμάτισε καθοριστικό ρόλο στην βελτίωση και τελειοποίηση των προτεινόμενων αλλαγών, διασφαλίζοντας την ευθυγράμμισή τους με τα πρότυπα και τους στόχους του έργου. Αυτή η διαδικασία συνεργατικής ανάπτυξης υπογραμμίζει τη συλλογική δέσμευση για αριστεία στην κοινότητα του MNE-Python και αναδεικνύει την αποτελεσματικότητα των συνεργατικών προσπαθειών ανάπτυξης λογισμικού ανοιχτού κώδικα.

4.2 ΠΡΟΒΛΗΜΑΤΙΣΜΟΙ ΚΑΙ ΠΡΟΤΑΣΕΙΣ ΓΙΑ ΤΟ ΜΕΛΛΟΝ

Καθώς εξελίσσεται το λογισμικό MNE, προκύπτουν διάφορες εκτιμήσεις και προκλήσεις για τη μελλοντική του ανάπτυξη. Η αρχική ανάγκη για ευαισθησία και ευέλικτη κλιμάκωση καλύφθηκε, ωστόσο υπάρχουν και άλλες βελτιώσεις που μπορούν να πραγματοποιηθούν. Ένας βασικός τομέας εστίασης είναι η εφαρμογή αυτόματης βαθμονόμησης οθονών, σε αντίθεση, ή εναλλακτικά με τη βαθμονόμηση από το χρήστη που γίνεται τώρα, για τη διασφάλιση ακριβών αναπαραστάσεων πραγματικών μηκών σε διαφορετικά συστήματα και οθόνες απεικόνισης, και ανεξάρτητα από τον παράγοντα του ανθρώπινου λάθους. Με την ενσωμάτωση δεδομένων λειτουργιών στον τομέα της βιβλιοθήκης PyQt5, η MNE-Python θα μπορούσε να προσαρμόζει δυναμικά τις παραμέτρους απεικόνισης με βάση τα χαρακτηριστικά της οθόνης, βελτιώνοντας τη συνοχή και μειώνοντας την ανάγκη για χειροκίνητες προσαρμογές.

Μια άλλη σημαντική σκέψη είναι η εφαρμογή αλγορίθμων δυναμικής βαθμονόμησης που προσαρμόζουν τις παραμέτρους των διαστάσεων του παραθύρου σε πραγματικό χρόνο, καθώς αυτό αλλάζει μέγεθος. Αυτή τη στιγμή η λειτουργία της ευαισθησίας συνοδεύεται παράλληλα από κλείδωμα των διαστάσεων στην τιμή που βρίσκονται εκείνη τη στιγμή. Αυτό το χαρακτηριστικό εξασφαλίζει αρχικά την δυνατότητα αλλαγής του μεγέθους, αλλά και ότι οι πληροφορίες σχετικές με την ευαισθησία δεν περιέχουν σφάλματα. Η ενσωμάτωση της δυνατότητας αυτής ταυτόχρονα με τη βαθμονόμηση θα προσφέρει στους χρήστες καλύτερο έλεγχο και χρηστικότητα του εργαλείου, ακόμη κι αν είναι μία καθαρά ποιοτική βελτίωση.

Κατά την ανάπτυξη διαπιστώθηκε επίσης μία δυσκολία στην αναγνωσιμότητα των τιμών των πλατών των καναλιών, όπως αυτά διατυπώνονται στην οθόνη από την κλάση “ScaleBarText”. Αυτό το πρόβλημα διογκώθηκε από την προσθήκη της Βαθμονομημένης Λειτουργίας, καθώς αυτή προσθέτει σε αυτά τα κείμενα τις νέες τιμές ευαισθησίας σε μία νέα γραμμή. Ο μεγάλος πλέον όγκος κειμένου όχι μόνο δεν είναι ευανάγνωστος, αλλά κρύβει και ένα μέρος του γραφήματος από πίσω του, εμποδίζοντας την απολύτως σωστή λειτουργία.

Παράλληλα, δίπλα από το κείμενο των πλατών των καναλιών, βρίσκεται μία κάθετη γραμμή, η κλάση “ScaleBar”, η οποία πρακτικά είναι το μοναδικό σημείο αναφοράς για το πλάτος του καναλιού, διαθέσιμο στο χρήστη. Παρόμοια με το κείμενο και αυτή υστερεί στην ορατότητα, όπως επίσης εν μέρει και στη λειτουργικότητά της, καθώς βρίσκεται στο αριστερό σημείο του γραφήματος. Κρίνεται λοιπόν, πως η αισθητική ή και λειτουργική αναβάθμιση αυτών των δύο στοιχείων, του κειμένου του πλάτους και της γραμμής αναφοράς των καναλιών, θα επιφέρει βελτίωση στην αναγνωσιμότητα των τιμών και των δεδομένων, και διευκόλυνση στο έργο των χρηστών.

Εν κατακλείδι, η αντιμετώπιση αυτών των προβληματισμών, αλλά και νέων οι οποίοι προκύπτουν, απαιτεί συνεργατικές προσπάθειες και καινοτομία στο πλαίσιο της κοινότητας MNE. Με την αξιοποίηση των διαδικτυακών χώρων συζήτησης για την διατύπωση και ιεράρχηση των αναγκών των χρηστών, η MNE-Python μπορεί να συνεχίσει να εξελίσσει τις δυνατότητές της και να διατηρήσει τη θέση της ως κορυφαία πλατφόρμα

για την ανάλυση και την έρευνα νευροφυσιολογικών δεδομένων.

4.3 ΚΑΤΑΛΗΚΤΙΚΕΣ ΣΚΕΨΕΙΣ

Οι συνεισφορές που έγιναν μέσω αυτού του έργου, εφόσον γίνουν αποδεκτές από τους υπεύθυνους της κοινότητας και μέρος της επίσημης βάσης κώδικα, αναμένεται να έχουν θετικό αντίκτυπο στο οικοσύστημα λογισμικού MNE, ωφελώντας ερευνητές, κλινικούς ιατρούς και επαγγελματίες στον τομέα της ανάλυσης νευροφυσιολογικών δεδομένων. Καθώς το λογισμικό συνεχίζει να εξελίσσεται και να αναπτύσσεται, υπάρχουν δυνατότητες για περαιτέρω βελτιώσεις και εξελίξεις στο πλαίσιο της βιβλιοθήκης, καλύπτοντας πρόσθετες ανάγκες των χρηστών και προωθώντας την τελευταία λέξη της τεχνολογίας στην έρευνα και την ανάλυση νευροαπεικόνισης.

Η επιτυχής ολοκλήρωση του παρόντος έργου αποτελεί σημαντικό ορόσημο στη συνεχιζόμενη εξέλιξη του λογισμικού MNE-Python, όπως κάθε συνεισφορά σε ένα εργαλείο ανοιχτού κώδικα. Μέσω της συνεργατικής και συγκεντρωτικής προσπάθειας της επιστημονικής κοινότητας από ποικίλους κλάδους, μπορούν να πραγματοποιηθούν διαρκείς πρόοδοι σε κάθε τομέα.

Εν κατακλείδι, οι βελτιώσεις που υλοποιήθηκαν αποσκοπούν στην εξυπηρέτηση των χρηστών του και, εν τέλει, στην βαθύτερη κατανόηση του νευρικού συστήματος από την επιστημονική κοινότητα. Η προώθηση της επιστήμης της νευροφυσιολογίας μπορεί στο μέλλον να καταπολεμήσει ασθένειες αλλά ακόμα και να αποκαλύψει τοπία της άγνωστα στην ανθρωπότητα μέχρι τώρα. Στη φαντασία του καθενός επαφίεται, προς το παρόν, η διερεύνση των δυνατοτήτων της νευροφυσιολογίας στο μέλλον.

ΠΑΡΑΡΤΗΜΑ: ΥΠΟΒΛΗΘΕΙΣ ΚΩΔΙΚΑΣ

Σε αυτό το παράρτημα παρατίθεται ο κώδικας που υποβλήθηκε με αίτημα συγχώνευσης. Οι αλλαγές που βρίσκονται παρακάτω έχουν ανθυποβληθεί με την έκδοση του αποθετηρίου mne-qt-browser (Website: Github forked repository: mne-qt-browser-dev, 2023) με SHA του τελευταίου μηνύματος:

(939cde007dce4c569f54f541a1f0dce4aaa79e24)

1. ΑΡΧΕΙΟ _pg_figure.py

Καταρχάς παρατίθενται οι αλλαγές στο αρχείο mne_qt_browser/_pg_figure.py. Αυτό είναι και το βασικό αρχείο με όλες τις λειτουργικότητες που προστέθηκαν.

Γραμμή 42:

```
+from mne.defaults import _handle_default
```

Γραμμή 69:

```
from qtpy.QtCore import (
+    QRegularExpression,
```

Γραμμή 84:

```
from qtpy.QtGui import (
+    QRegularExpressionValidator,
```

Γραμμή 392:

```
    def update_scale(self):  # noqa: D102
        transform = QTransform()
-        transform.scale(1.0, self.mne.scale_factor)
+        transform.scale(1.0, self.mne.scale_factors[self.ch_type])
        self.setTransform(transform)
```

Γραμμή 1618:

```
class ScaleBarText(BaseScaleBar, TextItem):  # noqa: D101

-        self.setFont(_q_font(10))
+        self.setFont(_q_font(13, bold=True))

    def update_value(self):
```

```
        """Update value of ScaleBarText."""
        scaler = 1 if self.mne.butterfly else 2
        inv_norm = (
            scaler
-            * self.mne.scalings[self.ch_type]
-            * self.mne.unit_scalings[self.ch_type]
-            / self.mne.scale_factor
-        )
-        self.setText(f"{_simplify_float(inv_norm)} "
f"{self.mne.units[self.ch_type]}")
+            * self.mne.norms_dict[self.ch_type]
+            / self.mne.scale_factors[self.ch_type]
+        )
+        # If in calibration mode, also display the sensitivity in a new
line.
+        if self.mne.calibration_mode:
+            sens_norm = inv_norm * (self.mne.n_channels + 1) /
self.mne.height
+            self.setText(
+                f"{_simplify_float(inv_norm)} "
+                f"{self.mne.units[self.ch_type]}\n"
+                f"{_simplify_float(sens_norm)} "
+                f"[{self.mne.units[self.ch_type]}]/mm"
+            )
+        else:
+            self.setText(
+                  f"{_simplify_float(inv_norm)} "
f"{self.mne.units[self.ch_type]}"
+            )
```

Γραμμή 1958:

```
+# RegEx validator for float values. Accepts formats 42, 2.71, .5772, 3e8
+# Used in TimeScalingDialog and ScalingDialog
+rx = QRegularExpression(
+    "([0-9]+([.][0-9]*)?([eE][+-]?[0-9]+)?|[.][0-9]+([eE][+-]?[0-9]+)?)"
+)
+
+class TimeScalingDialog(_BaseDialog):
+    """
+    Dialog for the purposes of time scaling.

+    After calibration, the distance-related options become available.
+    Relevant measures:
+    - duration: The total duration displayed in seconds.
+    - width: The plot width in millimeters.
+    Derived measures, and editable textboxes:
+    - spp: Seconds per page. spp = duration
+    - spmm: Seconds per millimeter. spmm = duration/width
```

```
+    - mmps: Millimeters per second. mmps = width/duration
+    """
+
+    def __init__(self, main, title="Time Scaling Dialog", **kwargs):
+        super().__init__(main, title=title, **kwargs)
+        layout = QFormLayout()
+        layout.addRow(QLabel("Adjust time settings."))
+        # Seconds per page box
+        self.page_box = QLineEdit()
+        self.page_box.setValidator(QRegularExpressionValidator(rx, self))
+        self.page_box.editingFinished.connect(_methpartial(self._edited,
box="page"))
+        layout.addRow("Seconds/page:", self.page_box)
+        # Seconds per millimeter box
+        layout.addRow(QLabel("Availabale after calibration:"))
+        self.seconds_box = QLineEdit()
+        self.seconds_box.setValidator(QRegularExpressionValidator(rx, self))
+        self.seconds_box.editingFinished.connect(
+            _methpartial(self._edited, box="seconds")
+        )
+        layout.addRow("Seconds/mm:", self.seconds_box)
+        # Millimeters per second box
+        self.mm_box = QLineEdit()
+        self.mm_box.setValidator(QRegularExpressionValidator(rx, self))
+        self.mm_box.editingFinished.connect(_methpartial(self._edited,
box="mm"))
+        layout.addRow("mm/seconds:", self.mm_box)
+
+        self._update_boxes()
+        self.setLayout(layout)
+        self.show()
+
+    def _update_boxes(self):
+        # Enable/Disable boxes, depending on calibration mode.
+        # The duration box is irrelevant of width.
+        for box in [self.mm_box, self.seconds_box]:
+            box.setEnabled(self.mne.calibration_mode)
+        # Set texts
+        self.page_box.setText(str(round(self.mne.duration, 3)))
+        self.seconds_box.setText(str(round(self.mne.duration /
self.mne.width, 3)))
+        self.mm_box.setText(str(round(self.mne.width / self.mne.duration,
3)))
+
+    def _edited(self, box):
+        """
+        Determine the percentile step, with which duration is scaled.
+
+        - Seconds per page:
```

```
+        new_duration = old_duration + step*old_duration <=>
+        step = new_duration/old_duration - 1
+
+        - Seconds per millimeter:
+        step = new_duration/old_duration - 1 <=>
+        step = new_spmm * width / old_duration - 1
+
+        - Millimeters per second:
+        step = new_duration/old_duration - 1 <=>
+        step = width / (old_duration * new_mmps) - 1
+        """
+        # Seconds per page
+        if box == "page":
+            step = float(self.page_box.text()) / self.mne.duration - 1
+        # Seconds per millimeter
+        elif box == "seconds":
+            step = (
+                float(self.seconds_box.text()) * self.mne.width /
self.mne.duration - 1
+            )
+        # Millimeters per second
+        else:
+            step = self.mne.width / (self.mne.duration *
float(self.mm_box.text())) - 1
+        # Perform the change in duration and update boxes
+        self.weakmain().change_duration(step=step)
+
+    def closeEvent(self, event):
+        for box in [self.page_box, self.seconds_box, self.mm_box]:
+            _disconnect(box.editingFinished)
+        super().closeEvent(event)
+
+class ScalingDialog(_BaseDialog):
+    """
+    Dialog for the purposes of scaling, either the amplitude or sensitivity.

+    Amplitude of the plot is considered the height of the y axis,
+    for each channel. It is identical with the Scalebar text.
+    Sensitivity is the same, but in regard to the true length on the
monitor.
+    For that reason it is available only while in calibration_mode.
+    Sensitivity for a specific channel type is defined as:
+    sensitivity = amplitude / scalebar_length <=>
+    sensitivity = amplitude * (n_channels + 1) / height
+    where n_channels is the number of channels currently displayed (+1
because of
+    padding) and height is the plot height. Using the n_channels is vital,
+    because otherwise changing it would result in decalibration.
+    """
```

```
+
+    def __init__(self, main, title="Scaling Dialog", **kwargs):
+        super().__init__(main, title=title, **kwargs)
+        layout = QGridLayout()
+        # Initialize
+        self.amplitude_boxes = OrderedDict()
+        self.sensitivity_boxes = OrderedDict()
+        titles = _handle_default("titles")
+        # Titles
+        layout.addWidget(QLabel("Channel Types"), 0, 0)
+        layout.addWidget(QLabel("Amplitude"), 0, 1, 1, 2)
+        slbl = QLabel("Sensitivity")
+        layout.addWidget(slbl, 0, 3, 1, 2)
+        # Amplitude and Sensitivity Boxes
+        row = 1
+        for ch_type in [ct for ct in self.mne.ch_types_ordered if ct !=
"stim"]:
+            layout.addWidget(QLabel(titles.get(ch_type, ch_type.upper())),
row, 0)
+            # Amplitude Box
+            abox = QLineEdit()
+            abox.setValidator(QRegularExpressionValidator(rx, self))
+            abox.editingFinished.connect(
+                _methpartial(self._amplitude_edited, ch_type=ch_type)
+            )
+            self.amplitude_boxes[ch_type] = abox
+            layout.addWidget(abox, row, 1)
+            layout.addWidget(QLabel(self.mne.units[ch_type]), row, 2)
+            # Sensitivity Box
+            sbox = QLineEdit()
+            sbox.setValidator(QRegularExpressionValidator(rx, self))
+            sbox.editingFinished.connect(
+                _methpartial(self._sensitivity_edited, ch_type=ch_type)
+            )
+            self.sensitivity_boxes[ch_type] = sbox
+            layout.addWidget(sbox, row, 3)
+            layout.addWidget(QLabel("[" + self.mne.units[ch_type] + "]/mm"),
row, 4)
+
+            row += 1
+        self._update_boxes()
+        text = QLabel("Sensitivity is available only after calibration.")
+        layout.addWidget(text, row, 0, 1, 5)
+        self.setLayout(layout)
+        self.show()
+
+    def _update_boxes(self):
+        """Set the text of each box, according to definitions."""
```

```
+        for ch_type in [ct for ct in self.mne.ch_types_ordered if ct !=
"stim"]:
+            scaler = 1 if self.mne.butterfly else 2
+            amplitude = (
+                scaler * self.mne.norms_dict[ch_type] /
self.mne.scale_factors[ch_type]
+            )
+            self.amplitude_boxes[ch_type].setText(str(round(amplitude, 3)))
+            sensitivity = amplitude * (self.mne.n_channels + 1) /
self.mne.height
+            self.sensitivity_boxes[ch_type].setText(str(round(sensitivity,
3)))
+            # Disable sensitivity boxes if not in calibration_mode
+
self.sensitivity_boxes[ch_type].setEnabled(self.mne.calibration_mode)
+
+    def _amplitude_edited(self, ch_type):
+        """
+        Determine the new scale factor and scale accordingly.
+
+        Note that the scale factor is inversely proportional to the
amplitude.
+        new_scale_factor = scale_factor * old_amplitude / new_amplitude
+        """
+        new_value = float(self.amplitude_boxes[ch_type].text())
+        if new_value != 0:
+            scaler = 1 if self.mne.butterfly else 2
+            self.mne.scale_factors[ch_type] = (
+                scaler * self.mne.norms_dict[ch_type] / new_value
+            )
+
+            # Reapply clipping if necessary
+            if self.mne.clipping is not None:
+                self.weakmain()._update_data()
+
+            # Scale Traces (by scaling the Item, not the data)
+            for line in [line for line in self.mne.traces if line.ch_type
ch_type]:
+                line.update_scale()
+
+            # Update Scalebars
+            self.weakmain()._update_scalebar_values()
+
+            # Update the corresponding sensitivity box
+            sensitivity = new_value * (self.mne.n_channels + 1) /
self.mne.height
+            self.sensitivity_boxes[ch_type].setText(str(round(sensitivity,
2)))
+
```

```
+    def _sensitivity_edited(self, ch_type):
+        """
+        Determine the new scale factor and scale accordingly.
+
+        Same value with amplitude, but scaled by (n_channels + 1) / height,
+        according to the definition of sensitivity.
+        """
+        new_value = float(self.sensitivity_boxes[ch_type].text())
+        if new_value != 0:
+            scaler = 1 if self.mne.butterfly else 2
+            self.mne.scale_factors[ch_type] = (
+                scaler
+                * self.mne.norms_dict[ch_type]
+                * (self.mne.n_channels + 1)
+                / (new_value * self.mne.height)
+            )
+
+            # Reapply clipping if necessary
+            if self.mne.clipping is not None:
+                self.weakmain()._update_data()
+
+            # Scale Traces (by scaling the Item, not the data)
+            for line in [line for line in self.mne.traces if line.ch_type ==
ch_type]:
+                line.update_scale()
+
+            # Update Scalebars
+            self.weakmain()._update_scalebar_values()
+
+            # Update the corresponding ampplitude box
+            amplitude = new_value * self.mne.height / (self.mne.n_channels +
1)
+            self.amplitude_boxes[ch_type].setText(str(round(amplitude, 2)))
+
+    def closeEvent(self, event):
+        for box in self.amplitude_boxes.values():
+            _disconnect(box.editingFinished)
+        for box in self.sensitivity_boxes.values():
+            _disconnect(box.editingFinished)
+        super().closeEvent(event)
+
+class Spinbox(QSpinBox):
+    """Custom QSpinBox widget, used in the Calibration Dialog."""

+    def __init__(self, **kwargs):
+        super().__init__(**kwargs)
+        self.setMinimum(10)
+        self.setMaximum(10000)
+        self.setSuffix("mm")
```

```
+
+class CalibrationDialog(_BaseDialog):
+    """
+    Monitor Calibration.
+
+    The plot is painted black and the user is asked to measure the visible
black
+    rectangle. From this process, the height and width of the plot are
recorded
+    and the calibration_mode is enabled. While in calibration_mode, the user
has
+    access to sensitivity features, in the Scaling and Time Scaling dialogs,
+    and the sensitivity is displayed below the amplitude in the scalebars.
+    Also, the size of the window is locked, to prevent decalibration.
+    """
+
+    def __init__(self, main, title="Monitor Calibration", **kwargs):
+        super().__init__(main, title=title, **kwargs)
+        layout = QFormLayout()
+        color = "light" if self.mne.dark else "dark"
+        layout.addRow(
+            QLabel(
+                f"Measure the {color} area of the plot"
+                " and enter your measurements below:"
+            )
+        )
+        # Spinboxes for measurements in millimeters
+        self.height_box = Spinbox()
+        self.height_box.setValue(self.mne.height)
+        layout.addRow("Enter the height of the black area:",
self.height_box)
+        self.width_box = Spinbox()
+        self.width_box.setValue(self.mne.width)
+        layout.addRow("Enter the width of the black area:", self.width_box)
+        # Enable/Disable calibration_mode buttons
+        btns = QHBoxLayout()
+        self.enable_btn = QPushButton("Enable")
+        self.enable_btn.clicked.connect(self._enable_mode)
+        btns.addWidget(self.enable_btn)
+        self.disable_btn = QPushButton("Disable")
+        self.disable_btn.clicked.connect(self._disable_mode)
+        btns.addWidget(self.disable_btn)
+        layout.addRow(btns)
+        help_text = QLabel(
+            "While in calibration mode, the window and the plot can't be
resized."
+            "\nTo adjust the size, please disable calibration mode first."
+        )
+        layout.addRow(help_text)
```

```
+         self.setLayout(layout)
+         self.show()
+
+     def _enable_mode(self):
+         # Enable calibration mode
+         self.mne.calibration_mode = True
+         # Record measurements
+         self.mne.height = self.height_box.value()
+         self.mne.width = self.width_box.value()
+         self.weakmain()._toggle_calibration_mode()
+
+     def _disable_mode(self):
+         # Disable calibration mode
+         self.mne.calibration_mode = False
+         self.weakmain()._toggle_calibration_mode()
+
+     def closeEvent(self, event):
+         _disconnect(self.enable_btn.clicked)
+         _disconnect(self.disable_btn.clicked)
+         self.mne.viewbox.setBackgroundColor(_get_color(self.mne.bgcolor,
self.mne.dark))
+         super().closeEvent(event)
```

Γραμμή 2080:

```
+         # If Scaling Dialog is open, update the boxes
+         if self.mne.scaling_fig is not None:
+             self.mne.scaling_fig._update_boxes()
+         # Toggle Scalebar Texts
+         for scalebar_text in self.mne.scalebar_texts.values():
+             scalebar_text.update_value()
```

Γραμμή 3125:

```
+         # Time scaling dialog
+         self.mne.time_scaling_fig = None
+         # Scaling dialog
+         self.mne.scaling_fig = None
+         # Calibration mode and calibration dialog
+         self.mne.calibration_mode = False
+         self.mne.calibration_fig = None
```

Γραμμή 3137:

```
-         # Scale-Factor
-         self.mne.scale_factor = 1
+         # Ordered channel types list, used in multiple instances
+         self.mne.ordered_types = self.mne.ch_types[self.mne.ch_order]
```

```
+        unique_type_idxs = np.unique(self.mne.ordered_types,
return_index=True)[1]
+        self.mne.ch_types_ordered = [
+            self.mne.ordered_types[idx] for idx in sorted(unique_type_idxs)
+        ]
+        # Scale factors dictionary
+        self.mne.scale_factors = dict()
+        # Inverted norms dictionary, used in calculating amplitudes
+        self.mne.norms_dict = dict()
+        for ct in self.mne.ch_types_ordered:
+            self.mne.scale_factors[ct] = 1
+            if ct != "stim":
+                self.mne.norms_dict[ct] = (
+                    self.mne.scalings[ct] * self.mne.unit_scalings[ct]
+                )
```

Γραμμή 3348:

```
+        # Set inital height and width of plot, for sensitivity
+        self.mne.height = int(self.mne.viewbox.height())
+        self.mne.width = int(self.mne.viewbox.width())
```

Γραμμή 3404:

```
+        ascaling_time = QAction(
+            QIcon.fromTheme("settings"), "Time Scaling Dialog", parent=self
+        )
+        ascaling_time.triggered.connect(self._toggle_time_scaling_fig)
+        self.mne.toolbar.addAction(ascaling_time)
```

Γραμμή 3432:

```
+        ascaling = QAction(QIcon.fromTheme("settings"), "Scaling Dialog",
parent=self)
+        ascaling.triggered.connect(self._toggle_scaling_fig)
+        self.mne.toolbar.addAction(ascaling)
```

Γραμμή 3507:

```
+        acalibration = QAction(
+            QIcon.fromTheme("settings"), "Monitor Calibration", parent=self
+        )
+        acalibration.triggered.connect(self._toggle_calibration_fig)
+        self.mne.toolbar.addAction(acalibration)
```

Γραμμή 3735:

```
-        # To keep order (np.unique sorts)
```

```
-         ordered_types = self.mne.ch_types[self.mne.ch_order]
-         unique_type_idxs = np.unique(ordered_types, return_index=True)[1]
-         ch_types_ordered = [ordered_types[idx] for idx in
sorted(unique_type_idxs)]
          for ch_type in [
              ct
-             for ct in ch_types_ordered
+             for ct in self.mne.ch_types_ordered
```

Γραμμή 3803:

```
    def scale_all(self, checked=False, *, step):
        """Scale all traces by multiplying with step."""
-         self.mne.scale_factor *= step
+         for ct in self.mne.scale_factors:
+             self.mne.scale_factors[ct] *= step
```

Γραμμή 3815:

```
+         # If Scaling Dialog is open, update the boxes
+         if self.mne.scaling_fig is not None:
+             self.mne.scaling_fig._update_boxes()
```

Γραμμή 3911:

```
+         # Update the boxes of Time scaling dialog, if it is open
+         if self.mne.time_scaling_fig is not None:
+             self.mne.time_scaling_fig._update_boxes()
```

Γραμμή 3933:

```
+             # Update Scaling dialog boxes, if it is open
+             if self.mne.scaling_fig is not None:
+                 self.mne.scaling_fig._update_boxes()
+
+             # Toggle Scalebar Texts
+             for scalebar_text in self.mne.scalebar_texts.values():
+                 scalebar_text.update_value()
```

Γραμμή 4030:

```
                        scaler = -1 if self.mne.butterfly else -2
                        inv_norm = (
                            scaler
-                           * self.mne.scalings[trace.ch_type]
-                           * self.mne.unit_scalings[trace.ch_type]
-                           / self.mne.scale_factor
+                           * self.mne.norms_dict[trace.ch_type]
```

```
+                            / self.mne.scale_factors[trace.ch_type]
                         )
```

Γραμμή 4102:

```
+                # If Scaling Dialog is open, update the boxes
+                if self.mne.scaling_fig is not None:
+                    self.mne.scaling_fig._update_boxes()
+                # Toggle Scalebar Texts
+                for scalebar_text in self.mne.scalebar_texts.values():
+                    scalebar_text.update_value()
```

Γραμμή 4142:

```
        # Update data of traces outside of yrange (reuse remaining trace-
items)
        for trace, ch_idx in zip(off_traces, add_idxs):
            trace.set_ch_idx(ch_idx)
            trace.update_color()
            trace.update_data()
+            trace.update_scale()
```

Γραμμή 4354:

```
+            # Previously scale_factor, now the scale_factors needs to be
cast as an
+            # array that can be broadcasted correctly by numpy in the
condition below.
+            factors_ordered =
np.atleast_2d(np.empty(np.shape(self.mne.data)[0])).T
+            for i in range(np.shape(self.mne.data)[0]):
+                factors_ordered[i] =
self.mne.scale_factors[self.mne.ordered_types[i]]
            self.mne.data[
-                abs(self.mne.data * self.mne.scale_factor) >
self.mne.clipping
+                abs(self.mne.data * factors_ordered) > self.mne.clipping
```

Γραμμή 4610:

```
+    def _toggle_time_scaling_fig(self):
+        if self.mne.time_scaling_fig is None:
+            TimeScalingDialog(self, name="time_scaling_fig")
+        else:
+            self.mne.time_scaling_fig.close()
+            self.mne.time_scaling_fig = None
+
+    def _toggle_scaling_fig(self):
```

```
+        if self.mne.scaling_fig is None:
+            ScalingDialog(self, name="scaling_fig")
+        else:
+            self.mne.scaling_fig.close()
+            self.mne.scaling_fig = None
```

Γραμμή 4624:

```
+    def _toggle_calibration_fig(self):
+        if self.mne.calibration_fig is None:
+            # Change the background color
+            CalibrationDialog(self, name="calibration_fig")
+            self.mne.viewbox.setBackgroundColor(
+                _get_color(self.mne.bgcolor, not self.mne.dark)
+            )
+        else:
+            self.mne.calibration_fig.close()
+            self.mne.calibration_fig = None
+
+    def _toggle_calibration_mode(self):
+        # Toggle everything that changes in calibration mode
+        # Toggle window size policy
+        if self.mne.calibration_mode:
+            self.setFixedSize(self.size())
+            self.mne.view.setFixedSize(self.mne.view.size())
+        else:
+            self.setMaximumSize(100000, 100000)
+            self.setMinimumSize(0, 0)
+            self.mne.view.setMaximumSize(100000, 100000)
+            self.mne.view.setMinimumSize(0, 0)
+
+        # Update Scaling dialog boxes, if it is open
+        if self.mne.scaling_fig is not None:
+            self.mne.scaling_fig._update_boxes()
+
+        # Update Time Scaling dialog boxes, if it is open
+        if self.mne.time_scaling_fig is not None:
+            self.mne.time_scaling_fig._update_boxes()
+
+        # Toggle Scalebar Texts
+        for scalebar_text in self.mne.scalebar_texts.values():
+            scalebar_text.update_value()
```

2. ΑΡΧΕΙΟ test_pg_specific.py

Εδώ υπάρχουν οι αλλαγές στο αρχείο mne_qt_browser/tests/test_pg_specific.py. Αυτό είναι το αρχείο που είναι υπεύθυνο για τις δοκιμές του κώδικα. Παρακάτω φαίνεται η νεοσύστατη συνάρτηση που ελέγχει τις νέες λειτουργικότητες.

Γραμμή 22:

```
+def test_sensitivity_etc(raw_orig, pg_backend):
+    """Test monitor calibration and sensitivity related settings."""
+    fig = raw_orig.plot()
+    fig.test_mode = True
+    QTest.qWaitForWindowExposed(fig)
+    QTest.qWait(50)
+    # Calibration Dialog
+    assert fig.mne.calibration_fig is None
+    with pytest.raises(ValueError, match="FooAction"):
+        fig._fake_click_on_toolbar_action("FooAction")
+    fig._fake_click_on_toolbar_action("Monitor Calibration", wait_after=100)
+    assert fig.mne.calibration_fig is not None
+    assert pg_backend._get_n_figs() == 2
+    fig._fake_click_on_toolbar_action("Monitor Calibration", wait_after=100)
+    assert fig.mne.calibration_fig is None
+    assert pg_backend._get_n_figs() == 1
+    fig._fake_click_on_toolbar_action("Monitor Calibration", wait_after=100)
+    assert fig.mne.calibration_fig is not None
+    # Check correct value passing
+    fig.mne.calibration_fig.height_box.setValue(100)
+    fig.mne.calibration_fig.width_box.setValue(200)
+    fig.mne.calibration_fig._enable_mode()
+    assert fig.mne.height == 100
+    assert fig.mne.width == 200
+
+    # Scaling Dialog
+    assert fig.mne.scaling_fig is None
+    fig._fake_click_on_toolbar_action("Scaling Dialog", wait_after=100)
+    assert fig.mne.scaling_fig is not None
+    assert pg_backend._get_n_figs() == 3
+    fig._fake_click_on_toolbar_action("Scaling Dialog", wait_after=100)
+    assert fig.mne.scaling_fig is None
+    assert pg_backend._get_n_figs() == 2
+    fig._fake_click_on_toolbar_action("Scaling Dialog", wait_after=100)
+    assert fig.mne.scaling_fig is not None
+    # Check boxes
+    for ch_type, box in fig.mne.scaling_fig.amplitude_boxes.items():
+        # Amplitude / 2
```

```
+        amplitude = float(box.text()) / 2
+        box.setText(str(amplitude))
+        fig.mne.scaling_fig._amplitude_edited(ch_type=ch_type)
+        assert fig.mne.scale_factors[ch_type] == 2
+        sens_box = fig.mne.scaling_fig.sensitivity_boxes[ch_type]
+        sensitivity = amplitude * (fig.mne.n_channels + 1) / 100
+        assert sens_box.text() == str(round(sensitivity, 3))
+        # Sensitivity / 4
+        sens_box.setText(str(sensitivity / 4))
+        fig.mne.scaling_fig._sensitivity_edited(ch_type=ch_type)
+        assert fig.mne.scale_factors[ch_type] == 8
+
+    # Time Scaling Dialog
+    assert fig.mne.time_scaling_fig is None
+    fig._fake_click_on_toolbar_action("Time Scaling Dialog", wait_after=100)
+    assert fig.mne.time_scaling_fig is not None
+    assert pg_backend._get_n_figs() == 4
+    fig._fake_click_on_toolbar_action("Time Scaling Dialog", wait_after=100)
+    assert fig.mne.time_scaling_fig is None
+    assert pg_backend._get_n_figs() == 3
+    fig._fake_click_on_toolbar_action("Time Scaling Dialog", wait_after=100)
+    assert fig.mne.time_scaling_fig is not None
+    # Check boxes
+    value = str(round(fig.mne.duration, 3))
+    assert fig.mne.time_scaling_fig.page_box.text() == value
+    assert fig.mne.time_scaling_fig.seconds_box.text() == str(
+        round(fig.mne.duration / 200, 3)
+    )
+    assert fig.mne.time_scaling_fig.mm_box.text() == str(
+        round(200 / fig.mne.duration, 3)
+    )
+    # Seconds per page
+    fig.mne.time_scaling_fig.page_box.setText(str(7))
+    fig.mne.time_scaling_fig._edited(box="page")
+    assert fig.mne.duration == 7
+    # Seconds per millimeter
+    fig.mne.time_scaling_fig.seconds_box.setText(str(0.01))
+    fig.mne.time_scaling_fig._edited(box="seconds")
+    assert fig.mne.duration == 2
+    # Millimeters per second
+    fig.mne.time_scaling_fig.mm_box.setText(str(50))
+    fig.mne.time_scaling_fig._edited(box="mm")
+    assert fig.mne.duration == 4
+
+    # Close dialogs
+    fig._fake_click_on_toolbar_action("Monitor Calibration", wait_after=100)
+    fig._fake_click_on_toolbar_action("Scaling Dialog", wait_after=100)
+    fig._fake_click_on_toolbar_action("Time Scaling Dialog", wait_after=100)
+    assert pg_backend._get_n_figs() == 1
```

```
+     fig.close()
```

ΒΙΒΛΙΟΓΡΑΦΙΑ

Crispin, L. (2006). Driving software quality: How test-driven development impacts software quality. IEEE software.

Delorme, A., & Makeig, S. (2004). EEGLAB: an open source toolbox for analysis of single-trial EEG dynamics including independent component analysis. Journal of neuroscience methods, 9-21.

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley Professional.

Gramfort, A., Luessi, M., Larson, E., Engemann, D. A., Strohmeier, D., Brodbeck, C., . . . Hämäläinen, M. S. (2013). MEG and EEG data analysis with MNE-Python. Frontiers in Neuroscience, 7. Frontiers in Neuroscience, 1-13.

Hämäläinen, M., Hari, R., Ilmoniemi, R. J., Knuutila, J., & Lounasmaa, O. V. (1993). Magnetoencephalography—theory, instrumentation, and applications to noninvasive studies of the working human brain. Reviews of modern Physics.

Khamis, N., Rilling, J., & Witte, R. (2013). Assessing the quality factors found in in-line documentation written in natural language: The JavadocMiner. Data & Knowledge Engineering, 19-40.

Kogut, B., & Metiu, A. (2001). Open-source Software Development and Distributed Innovation. Oxford review of economic policy, 17.

Luck, S. J. (2014). An introduction to the event-related potential technique. MIT press.

Michel, C. M., Murray, M. M., Lantz, G., Gonzalez, S., Spinelli, L., & Grave de Peralta, R. (2004). EEG source imaging. Clinical Neurophysiology, 115, 2195-2222. doi:https://doi.org/10.1016/j.clinph.2004.06.001

Niedermeyer, E., & da Silva, F. L. (2005). Electroencephalography: basic principles, clinical applications, and related fields. Lippincott Williams & Wilkins.

Oostenveld, R., Fries, P., Maris, E., & Schoffelen, J.-M. (2011). FieldTrip: open source software for advanced analysis of MEG, EEG, and invasive electrophysiological data. Computational intelligence and neuroscience, 1-9.

Plum, F. (1960). Handbook of Physiology. JAMA Neurology.

Website: Apache License, Version 2.0. (2004). Ανάκτηση 2 2024, από https://www.apache.org/licenses/LICENSE-2.0

Website: Conda. (χ.χ.). Ανάκτηση 2 2024, από https://docs.conda.io/en/latest/

Website: EDFbrowser. (χ.χ.). Ανάκτηση 2 2024, από https://www.teuniz.net/edfbrowser/

Website: EEGLAB. (χ.χ.). Ανάκτηση 2 2024, από https://sccn.ucsd.edu/eeglab/index.php

Website: FieldTrip. (χ.χ.). Ανάκτηση 2 2024, από https://www.fieldtriptoolbox.org/

Website: Git. (χ.χ.). Ανάκτηση 2 2024, από https://git-scm.com/

Website: Github Final Pull Request. (2024). Ανάκτηση 2 2024, από https://github.com/mne-tools/mne-qt-browser/pull/230

Website: Github forked repository: mne-qt-browser-dev. (2023). Ανάκτηση 2 2024, από https://github.com/Grifunf/mne-qt-browser-dev

Website: Github issue: Scaling & Sensitivity. (2022, 6 30). Ανάκτηση 2 2024, από https://github.com/mne-tools/mne-python/issues/10888

Website: Github organisation: mne-tools. (χ.χ.). Ανάκτηση 2 2024, από https://github.com/mne-tools

Website: Github Pull Request 1. (2023, 10 20). Ανάκτηση 2 2024, από https://github.com/mne-tools/mne-qt-browser/pull/208

Website: Github Pull Request 2. (2023, 11 4). Ανάκτηση 2 2024, από https://github.com/mne-tools/mne-qt-browser/pull/212

Website: Github repository: mne-python. (χ.χ.). Ανάκτηση 2 2024, από https://github.com/mne-tools/mne-python

Website: Github repository: mne-qt-browser. (χ.χ.). Ανάκτηση 2 2024, από https://github.com/mne-tools/mne-qt-browser

Website: GNU General Public License, version 3. (2007). Ανάκτηση 2 2024, από http://www.gnu.org/licenses/gpl.html

Website: MatPlotLib. (χ.χ.). Ανάκτηση 2 2024, από https://matplotlib.org/

Website: MNE Forum. (χ.χ.). Ανάκτηση 2 2024, από https://mne.discourse.group/

Website: MNE Forum - Scaling & Sensitivity. (2022, 6 1). Ανάκτηση 5 2023, από https://mne.discourse.group/t/scaling-sensitivity-uv-mm/5079

Website: MNE-Python - Contributing Guide. (2024). Ανάκτηση από https://mne.tools/stable/development/contributing.html

Website: MNE-Python - Documentation. (χ.χ.). Ανάκτηση 2 2024, από https://mne.tools/stable/documentation/index.html

Website: MNE-Python - Documentation - plot_raw(). (χ.χ.). Ανάκτηση 2 2024, από https://mne.tools/stable/generated/mne.viz.plot_raw.html

Website: MNE-Python - Documentation - Raw. (χ.χ.). Ανάκτηση 2 2024, από https://mne.tools/dev/generated/mne.io.Raw.html#mne.io.Raw

Website: MNE-Python - Home. (χ.χ.). Ανάκτηση 2 2024, από https://mne.tools/stable/index.html

Website: NumPy. (χ.χ.). Ανάκτηση 2 2024, από https://numpy.org/

Website: PyQT5. (χ.χ.). Ανάκτηση 2 2024, από https://doc.qt.io/qtforpython-5/

Website: PyQt5 Documentation - Graphics View Framework. (χ.χ.). Ανάκτηση 2 2024, από https://doc.qt.io/qtforpython-6/overviews/graphicsview.html

Website: PyQtGraph. (χ.χ.). Ανάκτηση 2 2024, από https://www.pyqtgraph.org/

Website: Pytest. (χ.χ.). Ανάκτηση 2 2024, από https://docs.pytest.org/en/8.0.x/

Website: Python. (χ.χ.). Ανάκτηση 2 2024, από https://www.python.org/

Website: Qt. (χ.χ.). Ανάκτηση 2 2024, από https://www.qt.io/

Website: SciPy. (χ.χ.). Ανάκτηση 2 2024, από https://scipy.org/

Website: The BSD 3-Clause License. (χ.χ.). Ανάκτηση 2 2024, από https://opensource.org/license/bsd-3-clause/

Website: The MIT License. (χ.χ.). Ανάκτηση 2 2024, από https://opensource.org/license/mit/

Website: Visual Studio Code. (χ.χ.). Ανάκτηση 2 2024, από https://code.visualstudio.com/