

IDP im Sommersemester 2013

QualityControlTool

Sarah Barton und Sonja Vogl
Technische Universität München

05. Juli 2013

Zusammenfassung

Diese Ausarbeitung soll dem Benutzer einen Einblick in die Bedienbarkeit des QualityControlTools geben. Ausgehend von der ersten Version der beiden Entwickler Simon und bla werden die Erweiterungen des QualityControlTools vorgestellt. Zudem geht diese Arbeit auf die Funktionen ein, die das QualityControlTool zur Analyse der Biosensoren GeneActive, Somnowatch, Shimmer und GT3X+ bereitstellt.

Inhaltsverzeichnis

1	Einleitung	3
2	Motivation	3
3	Ausgangslage - die erste Version des Tools	4
4	Das erweiterte Tool	6
4.1	Der Browse-Button	7
4.1.1	Aktion des Benutzers	7
4.1.2	Aktionen des QualityControlTool	7
4.1.3	Beteiligte JAVA- und MATLAB- Funktionen	8
4.2	Der Load-Button	9
4.2.1	Möglichkeiten für den Benutzer	9
4.2.2	Programmvorgänge	9
4.3	PlotAllData + Cut -Button	10
4.3.1	Implementierung	10
4.3.2	Cut Data	11
4.4	PlotRelevantData-Button	11
4.4.1	Aktionen des QualityControlTools	11
4.5	Wearing time van Hees-Button	11
4.5.1	Aktionen des QualityControlTool	12
4.6	Raw to Counts/METs-Button	12
4.6.1	Implementierung	13
5	Beispielablauf einer Analyse von Shimmer Daten	14
5.1	Starten des Programms	14
5.2	Laden einer Shimmer Datei	14
5.3	Umwandeln von Rohdaten in Counts	15
5.4	Plotten aller Daten	15
5.5	Plotten der relevanten Rohdaten	15
5.6	Berechnung der Tragezeit nach van Hees	15
5.7	?	15

1 Einleitung

Diese Arbeit gibt dem Benutzer des QualityControlTools einen Überblick über die Möglichkeiten, die das Programm zur Analyse der Biosensoren GeneActive, Somnowatch, Shimmer und GT3X+ dem User bietet. Zudem wird dem Benutzer nahegebracht, wie das QualityControlTool zu bedienen ist. Hierzu wird zunächst die Benutzeroberfläche (siehe Kapitel 4), wie auch die Funktionsweise der einzelnen Buttons dargestellt. Zusätzlich wird grob auf die Implementierung eingegangen.

2 Motivation

Das Institut für Medizinische Statistik und Epidemiologie (IMSE) der TU München beschäftigt sich mit der Analyse der Daten der KORA Studien. KORA steht für die *kooperative Gesundheitsforschung in der Region Augsburg*.

Das Ziel der *Feasibility*-Studie der Nationalen Kohorten war das Testen der Realisierbarkeit mono- und multifunktioneller Apparaturen, die Körperbewegungen messen. So sollten Körperleistung (PA) und der Energieverbrauch analysiert werden. Im speziellen wurden vier verschiedene Biosensoren betrachtet, die *raw acceleration data* für mindestens 7 Tage lieferten.

Das QualityControlTool soll das Auswerten der csv-Dateien von Biosensoren erleichtern. Dafür existieren zahlreiche Algorithmen, welche in MATLAB implementiert sind. Allerdings besitzen viele in der Medizin nur ein eingeschränktes Informatikwissen. Für diese ist eine Auswertung mit einem hohen zeitlichen und hohem Arbeitsaufwand, sich in Matlab einzuarbeiten, verbunden.

Mit dem QualityControlTool ist eine einfache Bedienbarkeit ohne informatische bzw. mathematische Kenntnisse gegeben. Sobald man mit der Benutzeroberfläche vertraut ist, übernimmt der PC das Auswerten der Daten. Zusätzlich werden die ausgewerteten Daten so gespeichert, dass sie schnell auffindbar sind und jederzeit erneut abgerufen werden können.

3 Ausgangslage - die erste Version des Tools

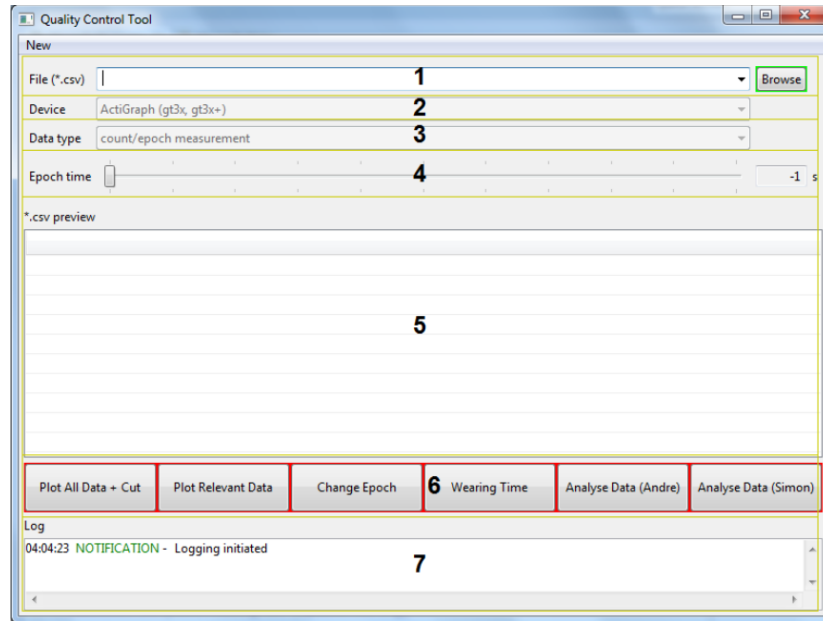


Abbildung 1: Gui des QualityControlTools 1. Version

Die erste Version des QualityControlTools war bislang nur auf den Biosensor GT3X+ Actigraph ausgerichtet. Für diesen Sensor ließen sich csv-Dateien des Datentyps *count data* auswerten. Hierzu konnte der Benutzer einen Pfad mit dem entsprechenden Dateinamen übergeben. Beim Betätigen des BROWSE-Buttons (siehe Abb. 1: 1) wurden die Daten der gewählten Datei geladen. Diese wurden dann als TIMESERIES-Format als MATLAB-Datei unter dem Namen *Kora1.mat* gespeichert.

Für den Sensortyp und den Datentyp war bereits die Vorrichtung für eine weitere Auswahl des Benutzers vorhanden. (siehe Abb. 1: 2 und 3) Diese blieb aber in der ersten Version dem Benutzer gesperrt. Die Anzeige blieb auf *Actigraph(gt3x,gt3x+)* sowie *count/epoch measurement* gesetzt.

Zudem ermöglichte es das Tool durch das Betätigen des PLOT ALL DATA + CUT-Button, die eingelesenen Daten zu *plotten*. Der resultierende Graph wurde unter dem Namen *KoraFirstPlot.png* im Dateisystem gespeichert.

In einem nächsten Schritt konnten die gegebenen Daten zugeschnitten werden. Der Benutzer trug hierzu einen Starttag und die Anzahl der Folgetage in ein vorgegebenes Feld ein. Alle Daten, die sich nicht in diesem Zeitraum

befanden, wurden weggeschnitten. Die resultierende TIMESERIES überschrieben *Kora1.mat*. Aus den entstandenen Daten wurde wiederum ein Graph erstellt. Dieser wurde unter dem Namen *KoraSecondPlot.png* im Dateisystem abgespeichert.

Als ein so genannter Zwischenschritt war das Verändern der Epochenlänge zu sehen. Dafür musste zunächst die gewünschte Epochenlänge, als ein Vielfaches der bisherigen, eingestellt werden. (siehe Abb. 1: 4) Mit Hilfe von Matrizenberechnungen wurden die Werte eines Intervalls aufsummiert. Das Resultat überschreibt *Kora1.mat*.

Die Daten der csv-Datei ließen sich sowohl mit Hilfe von dem Algorithmus von Andre wie auch nach dem Algorithmus von Simon analysieren. Bei dem erstgenannten wurden für jeden Tag der Eingabedaten eine bestimmte Anzahl an Werte berechnet, welche in einem Vektor gespeichert wurden. Der Vektor ließ sich anschließend in der Datei *finaldata.csv* auffinden.

Auch ließ sich die Tragezeit (siehe Abb. 1: 6) eines Sensors mit Hilfe des Algorithmus von Hecht al 98 berechnen. Resultierend geht daraus ein mit 0 oder 1 befüllten Vektor hervor. Dabei steht 0 für nichtgetragen und 1 für getragen. Aufzufinden war der Vektor sowohl als MATLAB-Datei *wearingTime.mat* und als csv-Datei *wearingTime.csv* vorzufinden. Die csv-Datei bestand aus einer Zeitspalte sowie einer Spalte mit den errechneten Werten.

Abschließend ermöglichte das Tool eine Auswertung der Daten nach Simon.(siehe Abb. 1) Es wurden pro Tag als auch für den gesamten Datensatz eine bestimmte Anzahl an Daten bestimmt. Die Ergebnisse waren in den Dateien *sum.table_[Klassenname]* und *sum.filt_[Klassenname]* abgelegt. Um Problemen in Java vorzubeugen, wurde am Ende das Workspace geleert.

Für weiteres Interesse verweisen wir Sie auf den Praktikumsbereich *Quality Control Tool for Different Accelerometer Data* von Johannes Eder und Martin Attenkofer.

4 Das erweiterte Tool

Bei der Erweiterung des gerade vorgestellten QualityControlTools kam es, neben den Änderungen in der Funktionalität, auch zu einigen Änderungen in der Oberfläche. Diese werden im Folgenden genauer erläutert.

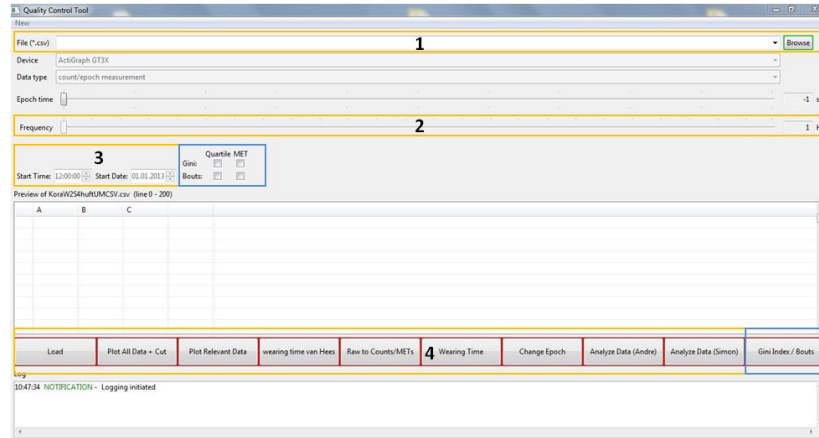


Abbildung 2: Gui des QualityControlTools 2. Version

In 2 sind alle Änderungen zur vorherigen Version gekennzeichnet. Dabei wird in dieser Arbeit auf die blau Markierten Felder nicht eingegangen, da diese Gegenstand einer anderen, parallel laufenden, Arbeit sind.

In Feld 1 der Abbildung 2 hat sich für den Betrachter nichts geändert. Allerdings verbirgt sich hinter dem *Browse*-Button eine andere Funktionalität. Statt die Datei zu laden wie in der ersten Version des Tools, wird hier nur noch die Datei ausgewählt und der entsprechende Dateipfad in die Java-Umgebung geladen. Das eigentliche Laden in die Matlab-Umgebung findet erst mit drücken des *Load*-Button (siehe Abb. 2: 4) statt. Da die wesentlichen Änderungen des Tools darin liegen, dass mehrere Sensorarten und auch Rohdaten ausgewertet werden können musste die Funktionalität des *Load*-Buttons ebenfalls erweitert werden. Die genaue Dokumentation dafür ist in Kapitel 4.2 beschrieben.

Wie gerade erwähnt kann das erweiterte Tool auch Rohdaten auswerten. Diese haben keine Epochenlänge in Minuten oder Sekunden wie *count/epoch measurement*, sondern werden mit einer bestimmten Frequenz aufgenommen. Da nicht in allen Rohdaten diese Frequenz angegeben ist, muss der User diese wissen und per Hand eingeben. Dies ist vor Allem bei Shimmer und Somnowatch Daten nötig. Bei ActiGraph GT3X+ und GeneActive Daten wird die Frequenz automatisch ausgelesen (vgl. Kapitel 4.2). Jedoch sollte der User hier immer die Frequenz überprüfen. Damit der User also zugriff auf die

Frequenz hat, diese einstellen und überprüfen kann ist die Leise Frequency (siehe Abb. 2: 2) hinzugekommen.

In der ersten Version des Tools mussten die auszuwertenden Daten immer um 12:00 Uhr beginnen. Da die jetzigen Daten auch zu anderen Zeitpunkten anfangen können wurde das Tool um die Felder *StartTime* und *StartDate* erweitert (siehe Abb. 2: 3). Diese muss der User für Somnowatch und Shimmer Daten per Hand eingeben, da in den entsprechenden Dateien weder Informationen über Anfangszeit noch über Anfangsdatum gespeichert sind. Für Daten der anderen Sensoren werden diese Informationen automatisch eingelesen (vgl. Kapitel 4.2).

Im Bereich der Buttons (siehe Abb. 2: 4) sind ebenfalls einige Neuerungen gemacht worden. Zum einen wurde der oben erwähnte *Load*-Button hinzugefügt. Zum Anderen wurde für die Auswertung von Rohdaten der Button *wearing Time van Hees* hinzugefügt (vgl. Kapitel 4.5). Um es dem User zu ermöglichen Rohdaten in Epochdaten umzuwandeln wurde zudem der Button *Raw to Counts/METs* neu angelegt (vgl. Kapitel 4.6).

4.1 Der Browse-Button

4.1.1 Aktion des Benutzers

Um mit dem QualityControlTool arbeiten zu können, muss zunächst der *Browse*-Button betätigt werden. Im Zuge dessen muss die zu analysierende csv-Datei aus dem entsprechenden Ordner auf dem Computer des Users ausgewählt werden.

4.1.2 Aktionen des QualityControlTool

Ist die auszuwertende Datei gewählt, stellt das QualityControlTool vorläufige Berechnungen an. Es wird aus der csv-Datei der Name des jeweiligen Sensors ausgelesen und in die Gui des Tools eingetragen. Anhand der Datei wird ebenfalls die Art der Daten eingetragen. Handelt es sich um Rohdaten, erscheint auf der Anzeige *raw data measurement*, andernfalls *count/ epoch measurement*. Zusätzlich wird die Frequenz der Sensordaten bestimmt und ebenfalls dem Benutzer dargelegt.

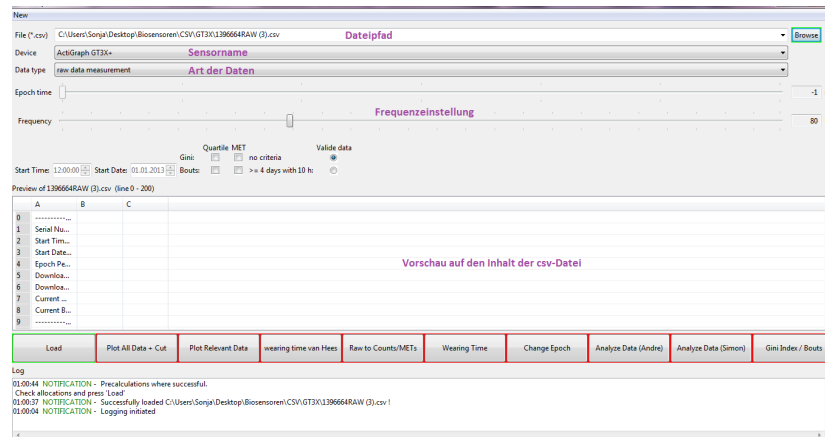


Abbildung 3: Gui nach Betätigen des *Browse*-Buttons

4.1.3 Beteiligte Java- und Matlab- Funktionen

Im JAVA-Code werden in der Klasse *KoraSteps* die Voreinstellungen ausgewählt. Dies findet in der Methode *public static boolean precalculation(String currentPath, String currentFile, String currentDevice)* statt. Hier wird auf die entsprechenden MATLAB-Funktionen zugegriffen.

Den MATLAB-Funktionen wird dabei der Dateipfad und der Dateiname der zuvor ausgewählten csv-Datei übergeben. Es wird die Funktion *FindSensorName(1, file_path)* aufgerufen. Entsprechend des Funktionsnamens liest diese aus der Kopfzeile der Datei den Namen des behandelten Sensors. Der Name des Sensors wird schließlich in JAVA auf den ermittelten Namen gesetzt und die Gui dementsprechend eingestellt. Misslingt die Suche, handelt JAVA dies mit einem Auffangen der *Exception*. Zusätzlich wird der User mit der Mitteilung “Unknown Device Type” benachrichtigt.

Auch die Frequenz des Sensor kann mit Hilfe der MATLAB-Funktion *FindSensorFrequency(1, file_path)* bestimmt werden. Auch dieses Ergebnis aktualisiert in JAVA die aktuelle *Frequeny* und ist für den Benutzer auf der Gui ersichtlich gemacht. Ein einschreitender Fehler wird wiederum mit einer *Exceptionbehandlung* abgefangen und dem User mit der Nachricht “Unknown Frequency!” kenntlich gemacht.

Erfolgreiche Vorberechnungen werden mit “Precalculations where successful. Check allocations and press “Load”” bestätigt.

Daneben erscheint die csv-Datei auf der dem Display der Benutzeroberfläche. Dafür ist die die Funktion *private void setupCsvTable(CsvFileLoader csvFile)* in der JAVA-Klasse *MainControlsBackend* zuständig.

Zum Schluss wird der *Load*-Button zur Nutzung freigeschalten.

4.2 Der Load-Button

Der *Load*-Button wird dem Benutzer erst nach dem Ausführen des *Browse*-Vorgangs freigeschaltet.

4.2.1 Möglichkeiten für den Benutzer

Die Trennung des *Browse*-Buttons und des *Load*-Buttons erlaubt dem User vom Programm selbstständig vorgenommene Einstellungen zu überprüfen und gegebenenfalls zu verifizieren. So lassen sich nach den Voreinstellungen sowohl der Sensorname, als auch die Frequenz und die Datenart einstellen.

4.2.2 Programmvorgänge

Die nun endgültigen Einstellungen werden nun erstmals in die MATLAB-Umgebung geladen. In der JAVA-Klasse *KoraSteps* wird nun die Methode *KoraStep1* ausgeführt. Abhängig von dem jeweiligen Sensor und dem *measurement* werden von hier aus die entsprechenden MATLAB-Funktionen aufgerufen. Für das *count/epoch measurement* existiert in diesem Tool nur eine Implementation für den Sensor Actigraph GT3X. Im Falle von falschen User-Einstellungen im Bezug auf die zugrundeliegende Datenart wird dies als Fehler behandelt. Dabei erscheint die Meldung “No code implemented for [sensorname] count/epoch measurement!”. Das Laden im *raw data measurement* ist für jeden Sensor erlaubt.

Für jeden dieser Fälle existiert eine MATLAB-Funktion. Diese weichen in Sensor- und Datentypspezifischen Details voneinander ab. Grund dafür ist beispielsweise der Unterschied im Aufbau der csv-Dateien. So besitzen die GeneActive Dateien einen Kopf mit den jeweiligen Informationen, wohingegen Somnowatch Dateien direkt mit den Daten starten. Auch die Anzahl der Spalten weichen voneinander ab.

In Matlab wird in der Funktion *FileLoader_[sensorname]Kora* bzw. für *count/epochmeasurement* *FileLoaderKora_new* die Messdaten aus der csv-Datei gefiltert und zeitlich angepasst. Das Ergebnis wird unter dem *[Filename]_RAW* bzw. *[Filename]_MAT* gespeichert und für den User jederzeit wieder aufrufbar.

Ist das Laden abgeschlossen, wird sowohl der *PlotAllData + Cut*-Button als auch der *RawToCounts/METs*-Button auf “grün” geschalten.

4.3 PlotAllData + Cut -Button

Die geladenen Daten lassen sich in einer Grafik darstellen. In der Grafik sind drei verschiedene Graphen zu sehen. (siehe Abb. 4) Die dazugehörige Legende beschreibt, was der jeweilige Graph darstellt. Zusätzlich ist die Grafik mit dem Sensorname, dem Datentyp und der Frequenz beschriftet.

In dem Fenster des *Plots* kann der Benutzer nun Einstellungen bezüglich Startzeit und Dauer der relevanten Daten treffen.

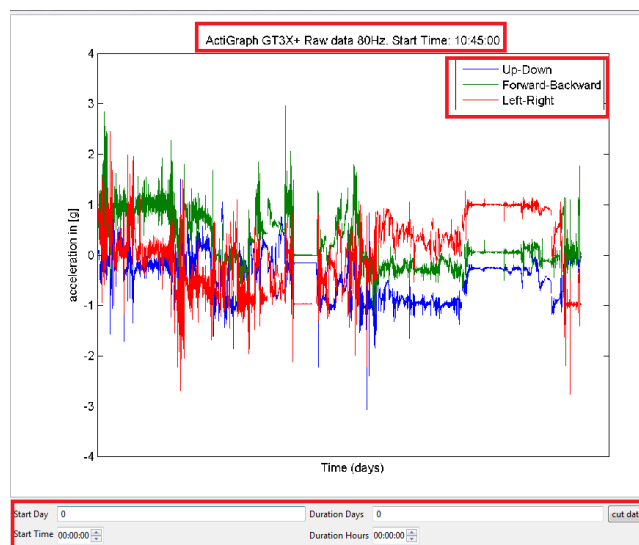


Abbildung 4: Beispiel: *Plot* einer GT3X+ Datei

4.3.1 Implementierung

Auch bei dem *Plotten* wird zunächst von der *KoraSteps.class* auf MATLAB zugegriffen. Von *step2_plotFile* wird je nach *measurement cutFirst1_Raw* bzw. *cutFirst1_new* aufgerufen.

Im Fall von *count/epoch measurement* berechnet die Funktion lediglich ein Schaubild zur einfacheren Visualisierung der Daten, wie schon in der ersten Version des Tools. Die einzigen Änderungen zur vorherigen Version sind eine ausführlichere Titulierung des erstellten Diagramms und eine Anpassung der Beschriftung auf der x-Achse. Auf dieser sind sowohl bei *count/epoch measurement* als auch bei *raw data measurement* die aufgezeichnete Zeit dargestellt. Dabei wird ein Tick immer genau um 12:00 Uhr eines Tages eingezeichnet. Da die Daten im Unterschied zur ersten Version aber auch zu anderen Zeiten starten können musste hier eine zusätzliche Berechnung eingefügt werden.

Bei *raw data measurement* zeigt das Diagramm statt nur einer Achse alle drei gemessenen Achsen. Diese werden, neben allen anderen Berechnungen, ebenfalls in der Methode *cutFirst1_Raw*, abhängig von dem jeweiligen Sensortyp, beschriftet.

4.3.2 Cut Data

Wie oben erwähnt gibt es bei den auszuwertenden Daten verschiedene Startzeiten. Damit der User möglichst flexibel ist, was das abschneiden der irrelevanten Daten angeht wurden die zwei Felder *Start Time* und *Duration Hours* hinzugefügt. Neben der schon vorhandenen Funktion im nächsten Schritt nur bestimmte Tage zu plotten, können jetzt auch bestimmte Zeiten ausgewählt werden. So ist es beispielsweise möglich nur die Daten eines Tages von 6:00 Uhr bis 17:00 Uhr zu betrachten.

Ändert der Benutzer nichts an den Einstellungen der Felder *Start Time* und *Duration Hours* so werden die Daten jeweils vom gewünschten Tag ab 12:00 Uhr bis zum letzten, gewünschten Tag um 12:00 Uhr dargestellt. Der letzte Tag berechnet sich dabei aus dem Eintrag im Feld *Start Days* plus dem Eintrag im Feld *Duration Days*. Überschreitet dieser Wert den vom Sensor gemessenen Zeitraum werden alle Daten ab dem Startdatum dargestellt.

Falls das Feld *Duration Days* ebenfalls nicht geändert wird und somit der Eintrag auf 0 bleibt bedeutet dies, dass die auszuwertenden Daten eine Länge von 0 Tagen besitzen. Das führt in weiteren Berechnungen zu Fehlern, da kein Datensatz mehr zur Verfügung steht. Deshalb sollte der User immer darauf achten, beim beschneiden der Daten eine sinnvolle und gültige Zeitangabe zu machen.

4.4 PlotRelevantData-Button

Der *PlotRelevantData-Button* wird dem Benutzer erst nach dem Ausführen des *Plots* aller Daten und Schneidevorgang zur Verfügung gestellt. Als Ausgabe erscheint die Graphik der zugeschnittenen relevanten Daten. Das Fenster entspricht dem im Kapitel 4.3 vorgestellten. Allerdings lässt sich diese Graphik hier nicht weiter schneiden. Demnach lässt dieses Fenster dem Benutzer keine weiteren Angaben zur Dauer bzw. Startzeit zu.

4.4.1 Aktionen des QualityControlTools

/TODO!!!

4.5 Wearing time van Hees-Button

Dieser Button stellt dem Benutzer eine Möglichkeit zur Berechnung der Tragezeiten bei *raw data measurement* zur Verfügung. Hierbei wird der Algorithmus

von van Hees (TODO: Referenz zu van Hees Doc einfügen!!!) verwendet. Der Button wird freigeschaltet sobald der Schritt *PlotAllData + Cut* erfolgreich beendet wurde.

4.5.1 Aktionen des QualityControlTool

Sobald der User den Button gedrückt hat wird über den JAVA-Code die Methode *public static boolean KoraStep5VanHees(String device, String file, int frequency)* in der Klasse *KoraSteps* aufgerufen. Über diese wird dann wiederum die MATLAB-Funktion *wearingTimevH_{Raw}(sensor, file, hz)* geöffnet und berechnet. Zur Berechnung werden die Daten aus der Datei *[Filename]_RAW* verwendet. Das heißt, wurden die irrelevanten Daten bereits mit der Funktion *PlotRelevantData* abgeschnitten, fehlen diese auch in der *wearingTime*-Berechnung. Wurde dieser Schritt (vgl. Kapitel 4.4) bisher noch nicht durchgeführt wird die Tragezeit für alle eingelesenen Daten ausgewertet. Dabei betrachtet der Algorithmus immer die eingehenden Daten in Intervallen von 15 Minuten. Aus diesem Grund müssen die auszuwertenden Daten auch eine Länge von mindestens 15 Minuten aufweisen, andernfalls wird der Fehler "Error while performing van Hees Wearing Time calculation! Data to small!" ausgegeben.

Das Ergebnis wird auf zwei verschiedene Arten abgespeichert: Es wird eine *.mat*-Datei erstellt. In dieser befindet sich eine Matrix *B* die in den ersten drei Spalten die gemessenen Daten des Sensors enthält und in der vierten Spalte jeweils die Information, ob der Sensor laut van Hees getragen wurde oder nicht. Wurde der Sensor getragen ist der Eintrag gleich 1, wurde er nicht getragen ist der Eintrag gleich 0.

Außerdem wird eine CSV-Datei abgespeichert. Diese enthält die genaue Zeitangabe jedes berechneten Intervalls und die dazugehörige Trage-Informationen aus der vierten Spalte der Matrix *B*. Zu finden sind diese Dateien jeweils unter dem Namen *[Filename]_wearingTimevHees* mit der entsprechenden Endung für den Datentyp.

4.6 Raw to Counts/METs-Button

Sobald eine Datei mit dem *Load*-Button (vgl. Kapitel 4.2) in die MATLAB-Umgebung geladen wurde und es sich bei den Daten um *raw data measurement* handelt besteht die Möglichkeit diese in *count/epoch measurement* umzuwandeln. Diese Funktionalität stellt der Button *Raw to Counts/METs* für den Benutzer zur Verfügung.

Der Button wird aktiv, das heißt seine Umrandung wird grün, sobald der *Load*-Vorgang erfolgreich abgeschlossen ist und solange im Feld *Data type* der Typ *raw data measurement* ausgewählt ist. Nachdem die Berechnungen durch diesen Button erfolgreich abgeschlossen wurden wird das Feld *Data type* automatisch

auf *count/epoch measurement* gesetzt und alle damit verbundenen Aktionen werden freigegeben. Damit ist der *Raw to Counts/METs*-Button nicht mehr aktiv, also für den User nicht mehr anwählbar.

4.6.1 Implementierung

Drückt der Benutzer den *Raw to Counts/METs*-Button wird in der JAVA-Klasse *KoraSteps* die Methode *public static boolean RawDataToEpoch(String currentPath, String currentFile, String currentDevice, int currentFrequency, String startDate, String startTime)* aufgerufen. Diese öffnet sogleich die MATLAB-Funktion *calculateCounts1(path, file, hz, sensor, d, t)*. Da für die Berechnung der Counts die Daten nochmals direkt aus der CSV-Datei gelesen werden wird hier eine Fallunterscheidung zwischen allen Sensortypen gemacht.

In jedem Fall werden zuerst die Daten aus der angegebenen Datei gelesen und in eine Matrix gespeichert. Dies geschieht jeweils mit einem Sensor-spezifischen Funktionsaufruf. Mit dieser Matrix werden in einem zweiten Schritt jeweils die Counts/METs berechnet. Diese Berechnung ist für die Sensoren Gene-Active, Shimmer und Somnowatch gleich und wird in der MATLAB-Funktion *rawDataToEpoch3(A, hz, DateTimeNum, file)* durchgeführt. Für den Sensor ActiGraphGT3X+ muss die Berechnung angepasst werden. Deshalb wird in diesem Fall nur die Funktion *rawDataToEpoch2(path, file, hz)* aufgerufen. Da diese Funktion also nur für den Sensor ActiGraphGT3X+ verwendet werden kann werden in ihr auch gleich die Daten aus der CSV-Datei eingelesen. Das heißt für diesen Sensor gibt es keinen extra Funktionsaufruf zum Einlesen der Daten.

Bei jedem Sensor werden die Rohdaten automatisch in Counts mit einer Epochenlänge von 60 Sekunden umgewandelt. Die Ergebnisse werden dann auf zwei verschiedene Arten abgespeichert: Als erstes werden sie als timeserie unter dem Namen *[Filename].mat* gespeichert. Mit dieser Datei ist es dann möglich alle weiteren Berechnungen durchzuführen, die auch schon bei der ersten Version des Tools für *count/epoch measurement* implementiert waren.

Außerdem werden die Counts und METs in einer CSV-Datei unter dem Namen *[Filename]-Count_MET.csv* gespeichert. Diese Datei enthält dann zuerst die genaue Zeitangabe der Daten. Dahinter stehen jeweils, durch Kommata getrennt, die zugehörigen Counts und dann METs.

Damit nach der erfolgreichen Umwandlung die Oberfläche auch das Richtige anzeigt wird im JAVA-Code als letzter Schritt in dieser Berechnung die Epochen-Skala aktualisiert und auf 60 Sekunden gesetzt.