Table of Contents

[Creating a Study Structure 2](#_Toc3276655)

[Introduction 2](#_Toc3276656)

[Study Description File 2](#_Toc3276657)

[Creating the study Description File using the GUI Program 3](#_Toc3276658)

[Step 1. Define the name of the study 3](#_Toc3276659)

[Step 2. Number of different Groups in the study 3](#_Toc3276660)

[Step 3. Number of MRI protocols used in the study 4](#_Toc3276661)

[Step 4. Enter the initials of the people who will have created ROIs 4](#_Toc3276662)

[Step 5. Enter the names of the groups, number of sessions and a list of imaged regions 5](#_Toc3276663)

[Step 6. Add the list of participant names/id codes for each defined group 5](#_Toc3276664)

[Step 7. Definition of ROI/imaged region types 5](#_Toc3276665)

[Step 8. Definition of the labels for individual ROIs 6](#_Toc3276666)

[Step 9. Slice index for the different imaged regions 6](#_Toc3276667)

[Step 10. Saving the study template file 7](#_Toc3276668)

[Creating the Study Directory Structure 8](#_Toc3276669)

[Analyzing Data in the Study Structure 8](#_Toc3276670)

[Copying image data into the study directory structure 8](#_Toc3276671)

[Modifying the ROI files for the data 9](#_Toc3276672)

[Set-up of the fitModeldata yaml file 13](#_Toc3276673)

[Description of the main parameters in the fitModeldData file 15](#_Toc3276674)

[Fitting T2 Data with fitEPGazz.py 19](#_Toc3276675)

[Fitting T2 Data with fitEPGazz.py on named single data files 19](#_Toc3276676)

[Fitting T2 Data with fitEPGazz.py in study directory structures 20](#_Toc3276677)

[Output from T2 Fitting Program: fitEPGazz.py 20](#_Toc3276678)

[Naming Conventions for ROI File-names 21](#_Toc3276679)

[Appendix A. Model Parameter Files and Fitting Equations 23](#_Toc3276680)

## Creating a Study Structure

### Introduction

Most clinical MRI studies are well defined before they are started and follow a basic design pattern based on the following criteria:

* number of participants
* number of visits
* control/participant groups
* MRI protocols
* persons working on the study, radiographers, principal investigator, post-docs
* data analysis
* final report generation

Therefore, one can design a data directory structure based on this knowledge and fill it in beforehand with place-holder files for the subsequent data analysis. This idea has been developed by people studying the brain and aids with data sharing and multi-site projects as data can be expected to be in certain directories. For more information see their web-site <http://bids.neuroimaging.io/>. We have attempted to follow their ideas for use in analysing T2 data via EPG methods.

The principle idea is to describe the study and data directory structure in a study template file which can be used by the data analysis software to verify the study and the results. The study template file is typically a YAML or JSON file that is a human readable text file that is structured in a way that it can be read in by programs written in python for example to create a dictionary structure that describes the project and data directory.

Advantages to this approach are many and become more numerous and beneficial with time.

* controlled vocabulary of description words for use in file names
* automated generation of file names
* automated generation of template files, for example Region of Interest files
* defined directory structure to store original data and result files
* similar studies use the same template directory structure and file-naming conventions
* long time studies can be be easily understood in terms of where things are
* processing scripts can be simplified and used between similar but different studies
* changes in personnel working on the study can be tracked over time

### Study Description File

The study description file contains different types of information:

* details of the study
  + study name
  + number of participants in the study
  + number of visits
  + participants grouped in terms of healthy controls and patients and sub-groups based on the number of visits
  + information on staff involved in the study, name, email, job description
* the MRI protocols used
  + protocols used, T2, dixon, diffusion
  + imaged regions of the body, leg, arm, head
* analysis of the images
  + slices used in the analysis
  + ROIs associated with which imaged region
  + ROI label names

The order of the different parts of the study description file is not important. Related information is grouped together, however since most of the file is generated by a program, it is saved in alphabetical order in terms of description items.

## Creating the study Description File using the GUI Program

The study description file contains all the information about the study. This file can be created by hand, but it is easy to introduce errors, so a simple GUI program has been written that asks for the information in a series of dialogs. Below we show the dialog boxes for a simple study that has a single participant, session, imaged region and ROI file.

The program is started from the command line and has no command-line arguments

-> python studyDescriptionGUI.py

### Step 1. Define the name of the study

The first step is define the name of the study and where it will be saved using a directory dialog. If the directory does not already exist, it can be created using the dialog.

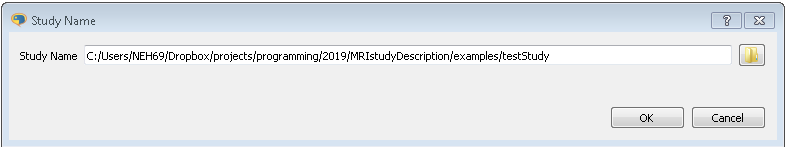


Figure 1 Define the name of the study

### Step 2. Number of different Groups in the study

In the study the number of different participant groups is defined. This is based on simple criteria such as healthy controls and participants with the condition. Within these groups, sub-groups can be defined based on the number of visits/sessions, the different protocols that they have.

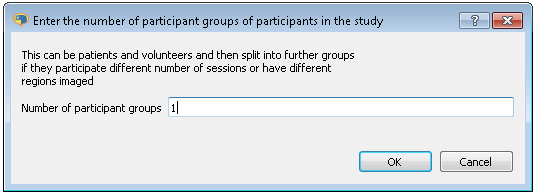


Figure 2 Number of Different Groups

### Step 3. Number of MRI protocols used in the study

The different MRI protocols that will be performed in the study should be listed next. Examples could be based on diffusion, T2, T1, and Dixon. The MRI protocols are added to the field separated by a space. In the example one MRI protocol is given.

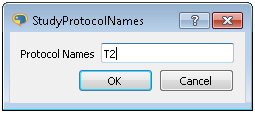


Figure 3 Names of the protocols used in the study

### Step 4. Enter the initials of the people who will have created ROIs

In the data analysis of a study many people work on the project, especially with the data analysis. Many projects require regions of interest to be drawn on the images for use in the analysis. This task is often performed by different people so it is useful for quality control reasons to keep track of who created the ROIs and this is done by giving a list of initials of the different people who will draw the ROIs. This is added to the ROI name so that this aspect of the data analysis may be audited. In the dialog one set of initials has been input. Further sets of initials could be entered separated by a space

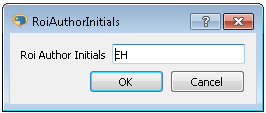


Figure 4 List of author initials who have created ROIs

### Step 5. Enter the names of the groups, number of sessions and a list of imaged regions

The names of the different groups of participants are entered in this dialog. The names must be unique. For each group the number of sessions/visits that the participants in the group attend during the study is entered. Finally, the imaged regions of the body are entered. In this field, if a phantom is used throughout the study a place name for it may be entered.

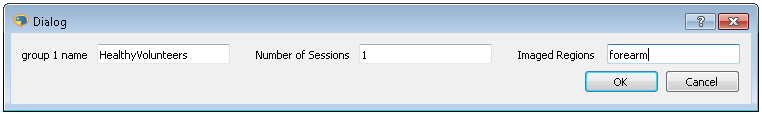


Figure 5 For each group defined previously, enter a name for the group the number of visits/sessions the participants will take part in and the imaged regions that will be looked at

### Step 6. Add the list of participant names/id codes for each defined group

For each defined group the participants id-code is entered in this dialog. The id-code is entered one per line. If more than one group is defined in the study then then it will appear along side the other groups in the dialog window.

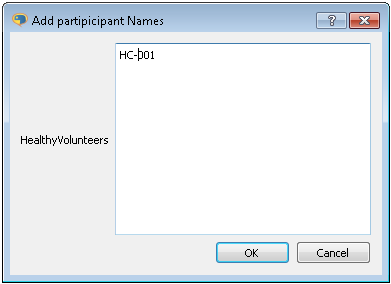


Figure 6 List of participant names/ID codes for each defined group

### Step 7. Definition of ROI/imaged region types

In this step the ROI/Imaged Region types are entered for each imaged region, forearm, upperarm,... This dialog is repeated separately for each group in the study. The group name appears in the dialog title in the border. Each ROI/Imaged Region type is entered on a separate line.

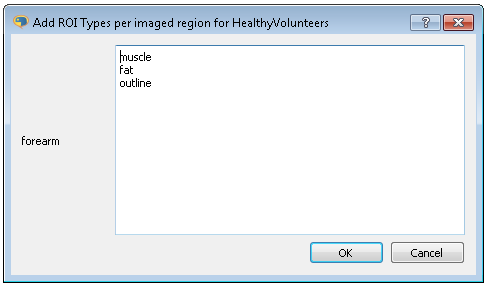


Figure 7 ROI/imaged region types being entered for Healthy volunteers group and the foream imaged region

### Step 8. Definition of the labels for individual ROIs

The names of the regions of interest for the different ROI/imaged region types are entered in this dialog. The names can be words or numbers, but must be unique with the same category

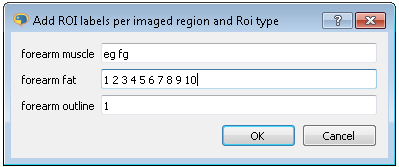


Figure 8 ROI labels entered for the different imaged regions and imaged region types

### Step 9. Slice index for the different imaged regions

The slices used in the study are dependent on the MRI protocol, the imaged region and imaged region/ROI type. Imaged slices are numbered based on a starting index of 1.

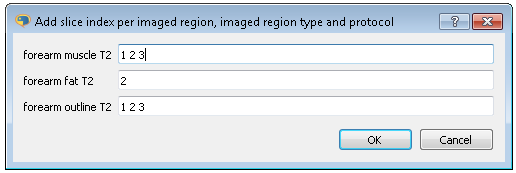


Figure 9 Slices used in the analysis of the data are defined in this dialog.

### Step 10. Saving the study template file

When the final dialog is closed the study template file is saved in the study directory. A number of fields within the study template file remain blank, these should be filled in by the user, using a text editor. The fields requiring updating by the user are:

* principal investigators
* research associates
* students

Below is the study description file produced by the previous dialogs

studyName: testStudy  
studyRootDir: MRIstudyDescription/examples  
  
HealthyVolunteers:  
 imagedRegions: [forearm]  
 name: HealthyVolunteers  
 numSessions: 1  
 participants: [HC-001]  
 rois:  
 forearm:  
 fat: ['1', '2', '3', '4', '5']  
 muscle: [fg, eg]  
 outline: ['1']  
 sessions: [sess-1]  
 slices:  
 forearm:  
 fat:  
 T2: ['2']  
 muscle:  
 T2: ['1', '2', '3']  
 outline:  
 T2: ['1', '2', '3']  
  
groupNames: [HealthyVolunteers]  
protocols: [T2]  
roiAuthors: [EH]  
  
principalInvestigators:  
- {address: '', email: '', initials: '', name: ''}  
  
researchAssociates:  
- {address: '', email: '', initials: '', name: ''}  
  
students:  
- {address: '', email: '', initials: '', name: ''}

## Creating the Study Directory Structure

Once the study description file has been created, the directory structure of the sttudy can be created by issuing the following command.

-> python create\_study\_directory\_structure.py

This script will create the following study directory structure and also create ROI template files

.  
└── testStudy  
 ├── HC-001  
 │   └── sess-1  
 │   └── forearm  
 │   ├── dicom  
 │   ├── rois  
 │   │   └── EH  
 │   │   ├── EH\_HC-001\_sess-1\_forearm\_fat.zip  
 │   │   ├── EH\_HC-001\_sess-1\_forearm\_muscle.zip  
 │   │   └── EH\_HC-001\_sess-1\_forearm\_outline.zip  
 │   └── T2  
 └── study\_description\_file.yml

## Analyzing Data in the Study Structure

### Copying image data into the study directory structure

To start with, copy the image data into the testStudy data directory. The image data is in ANALYZE format and can be found in the ***simplemodelData*** directory. It needs to be copied into the the T2 directory, see the tree structure above. The two image files are

WIP\_Forearm\_T2\_CLEAR.hdr  
WIP\_Forearm\_T2\_CLEAR.img

### Modifying the ROI files for the data

ROI template files were created automatically in the previous section. They now need to be edited using imageJ to correspond correctly to the data in the image file. An example is shown in Figure 10 below. The ROI file is loaded into imageJ by dragging the zip file on to the toolbar of the imageJ main window.

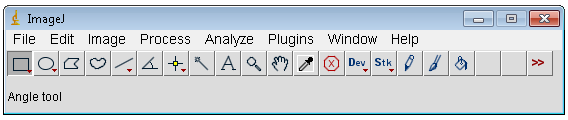


Figure 10 Drag files onto this window to load them into imageJ

The image is loaded in a similar manner, drag the hdr or img file onto the toolbar of the main imageJ window. The image needs to be flipped in the vertical direction as in Figure 11, before editing the ROIs.

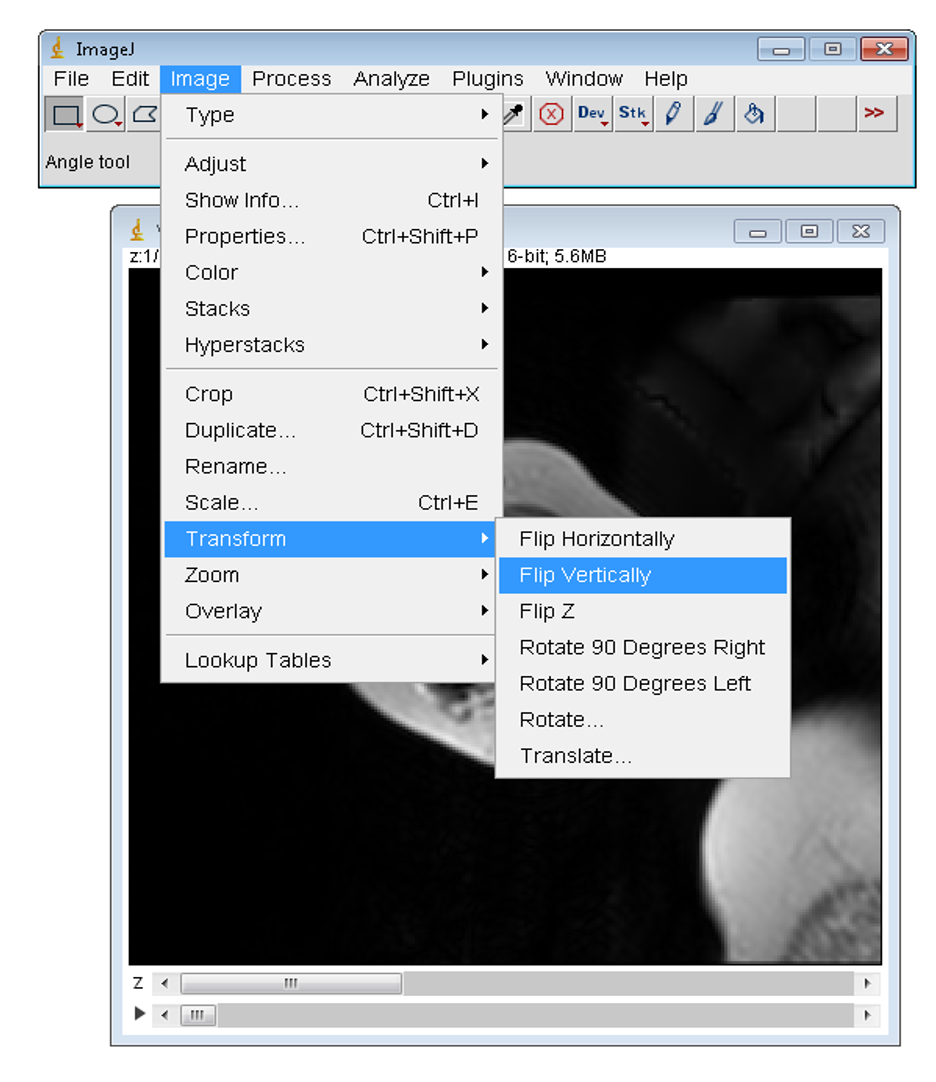


Figure 11 Flip vertically the data set before working with the ROIs

In Figure 12, the images have been flipped and the ROI file for the outlines have been loaded. One can see on the image the outline of the template ROI that will be edited. The ROI is edited by clicking the polygon button in the main imageJ window. Then clicking outside the template ROI and then clicking again at the position where you want to start drawing the ROI

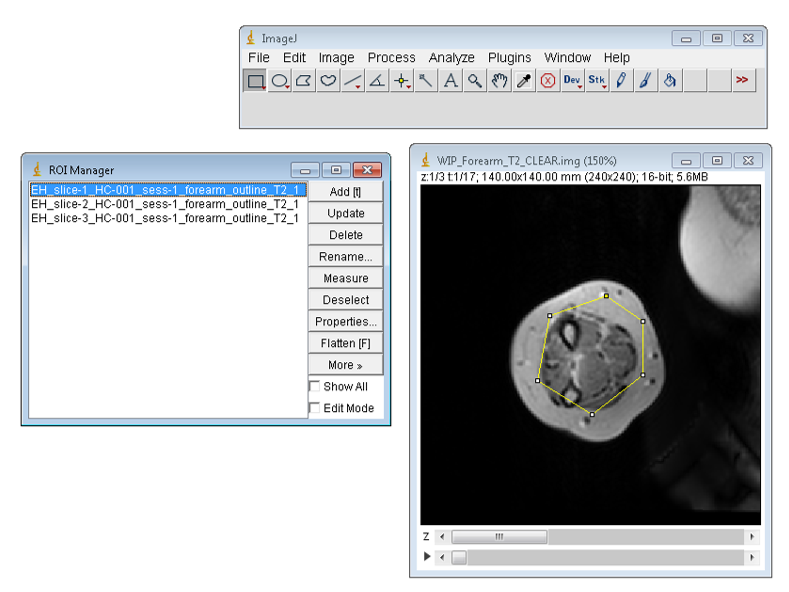


Figure 12 Outline template ROI prior to setting the polygon button and editing

Upon completion of the drawing of the ROI the update button is pressed in the ROI manager window to save the modifications. An example of a edited ROI is shown in Figure 13. When all ROIs have been drawn, making sure to increment the slice to correspond with the ROI name, all the ROIs are highlighted in the ROI manager window and the ***more*** button is clicked and then the ***save*** button. The user should then overwrite the corresponding zip ROI file.

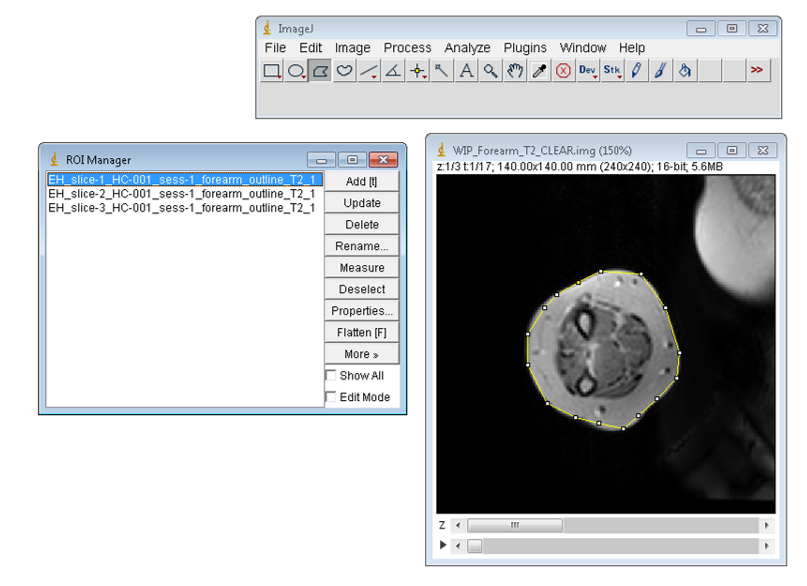


Figure 13 Newly drawn ROI ouline the muscle and fat of the forearm

The rois for the fat are best drawn using the rectangle tool in imageJ. Figure 14 shows the fat ROIs about to be saved after they have been edited

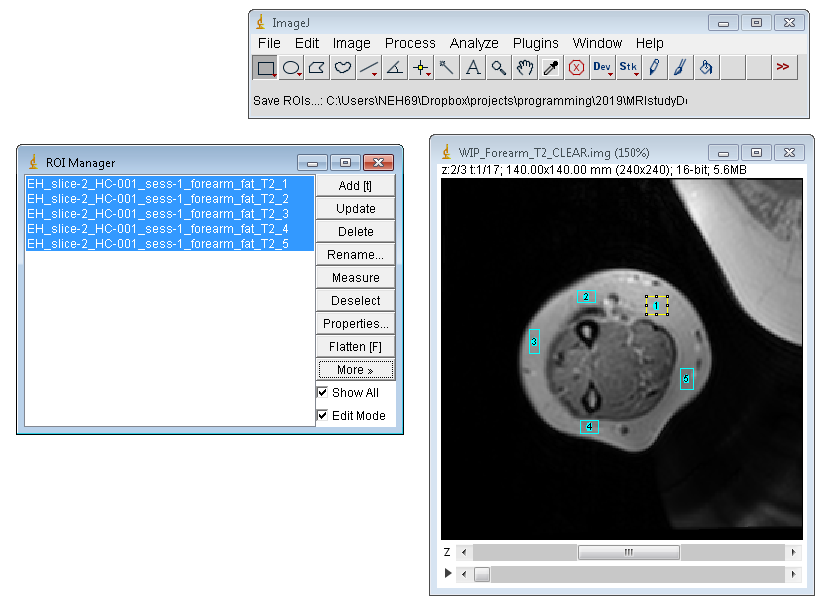


Figure 14 Fat ROIs using rectangle tool ready to be saved

### Set-up of the fitModeldata yaml file

Information to process the data in the study is provided to the fitting program by a fitmodelData yaml file. examples of these files can be found in the ***fitModelDataYAMLfiles*** directory for fitting single data sets and in the ***examples/fitModelDataYAMLfiles*** directory for fitting data in a study directory structure.

* fitStudyDataEPGAzz
* fitStudyDataFatEPG2
* fitStudyDataMuscleEPG1

The fitting program is capable of fitting a number of different models to the T2 image data based on the model structure found in the Azzabou paper. Muscle, fat and phantom data can be fit by the program using a simple exponential decay model or an extended phase graph model (EPG). This document will focus on fitting the T2 image data using the EPG model.

The example taken will be to fit the T2 image data using the **fitStudyDataEPGAzzAnalyze** yaml file.

The contents of the file are given below. Line numbers have been added to help with the explanation of the file, but the user should be aware that the sequential order of the different groups within the file is not important. The example YAML file has been set up to be used on the study directory structure defined earlier. The YAML input file has been designed to be used on data in a study directory structure or on individual files which have been defined within the file. In the following paragraphs the different parts of the file will be discussed.

1 fitModel: AzzEPG  
2   
3 # use a relative directory path when fitting data in study directory structure  
4 # when fitting individual files it can be set to a complete path  
5   
6 resultsDir: T2/results/muscle/AzzEPG  
7   
8 # Can only be used when fitting data in study directory structure  
9 # must be set as a relative directory path  
10   
11 fatResultsDir: T2/results/fat/fatEPG2  
12 setParamValsIndividually: True  
13   
14 # EPGAZZ model parameters to update  
15 # Always include the echo value  
16   
17 ParamVals:  
18 echo:  
19 value: 8  
20 min: 0  
21 max: 10  
22 vary: False  
23   
24 paramsModelName:  
25 AzzEPG: fitModelParameterJsonFiles/azz\_fatmuscle\_epg\_model\_params.json  
26 fatEPG2: fitModelParameterJsonFiles/two\_fat\_epg2\_model\_params.json  
27 fatEPG1: fitModelParameterJsonFiles/one\_fat\_epg1\_model\_params.json  
28 muscleEPG1: fitModelParameterJsonFiles/one\_fatmuscle\_epg1\_model\_params.json  
29 muscleEPG2 : fitModelParameterJsonFiles/two\_fatmuscle\_epg2\_model\_params.json  
30 phantomEPG1: fitModelParameterJsonFiles/oneParamEPGphantom\_model\_params.json  
31 phantomEPG2: fitModelParameterJsonFiles/twoParamEPGphantom\_model\_params.json  
32 Azzabou: fitModelParameterJsonFiles/azz\_params.json  
33   
34 roiFitModel:  
35 AzzEPG: muscle  
36 fatEPG2: fat  
37 fatEPG1: fat  
38 muscleEPG1: muscle  
39 muscleEPG2 : muscle  
40 phantomEPG1: phantom  
41 phantomEPG2: phantom  
42 Azzabou: muscle  
43   
44 fitSubject: [HC-001]  
45 fitSession: [sess-1]  
46 fitImagedRegions: [forearm]  
47 fitSlices: [1,2,3]  
48   
49 useRoiOutline: False  
50   
51 roiAuthorPreference: [EH]  
52 imageDataFormat: Analyze  
53   
54 #roisIndividual: simpleModelData/DMD\_001\_FOREARM\_RoiSet.zip  
55   
56 roiOutline: simpleModelData/EH\_DMDT\_001\_1\_foreArm\_outline.zip  
57   
58 analyzeHdr: simpleModelData/WIP\_Forearm\_T2\_CLEAR.hdr  
59 analyzeImg: simpleModelData/WIP\_Forearm\_T2\_CLEAR.img  
60   
61 # minimum step size across profile = 1  
62 # maximum step size across profile = 19  
63   
64 pulseProfileSteps: 10  
65   
66 # pulse profile should correspond to the protocol that was used to collect  
67 # the data  
68   
69 p90pulseProfile: simpleModelData/flip\_angle90\_clairewood.mat  
70 p180pulseProfile: simpleModelData/flip\_angle180\_clairewood.mat  
71   
72 # integration factor, might need to be altered based on value of  
73 # pulseProfileSteps and fit model being used  
74   
75 dx: 0.45

### Description of the main parameters in the fitModeldData file

#### Model Definition

The first line of the file states which model will be used by the fitting program

1 fitModel: AzzEPG

The model chosen must come from the eight fit models defined in lines 24 to 32. For each model there is a corresponding parameter file defined. These lines in the file, together with lines 34 to 42 should not be altered.

24 paramsModelName:  
25 AzzEPG: fitModelParameterJsonFiles/azz\_fatmuscle\_epg\_model\_params.json  
26 fatEPG2: fitModelParameterJsonFiles/two\_fat\_epg2\_model\_params.json  
27 fatEPG1: fitModelParameterJsonFiles/one\_fat\_epg1\_model\_params.json  
28 muscleEPG1: fitModelParameterJsonFiles/one\_fatmuscle\_epg1\_model\_params.json  
29 muscleEPG2 : fitModelParameterJsonFiles/two\_fatmuscle\_epg2\_model\_params.json  
30 phantomEPG1: fitModelParameterJsonFiles/oneParamEPGphantom\_model\_params.json  
31 phantomEPG2: fitModelParameterJsonFiles/twoParamEPGphantom\_model\_params.json  
32 Azzabou: fitModelParameterJsonFiles/azz\_params.json

##### Study Data Directory Structure Used or Not.

It was stated earlier that this file can be used to work on data found in a study directory structure or directly on files in a single directory. This is achieved by either putting a list of participants in the **fitSubject** list or setting it to empty.

44 fitSubject: [HC-001]

In this configuration we will use data found in study directory structure. If the line had been set in the following manner

44 fitSubject: []

then the program would work on named files within the fitModeldata file defined in lines 54 to 59

54 #roisIndividual: simpleModelData/DMD\_001\_FOREARM\_RoiSet.zip  
55   
56 roiOutline: simpleModelData/EH\_DMDT\_001\_1\_foreArm\_outline.zip  
57   
58 analyzeHdr: simpleModelData/WIP\_Forearm\_T2\_CLEAR.hdr  
59 analyzeImg: simpleModelData/WIP\_Forearm\_T2\_CLEAR.img

#### Defining sessions, imaged regions and slices

If the fitting program is taking data from the Data Directory Structure then the sessions and imaged regions to be used are defined as lists in lines 45 and 47. If defined data is used then the ***fitSession*** and ***fitImagedRegions*** fields are ignored.

45 fitSession: [sess-1]  
46 fitImagedRegions: [forearm]  
47 fitSlices: [1,2,3]

The slices to be used in the fitting are defined in line 47. The slices defined in this list are also used when the program works on defined data.

#### Updating the values in the LMFIT parameter structures.

Initial starting values for parameters can be set within the file by updating the fields starting at lines 18 to 22

14 # EPGAZZ model parameters to update  
15 # Always include the echo value  
16   
17 ParamVals:  
18 echo:  
19 value: 8  
20 min: 0  
21 max: 10  
22 vary: False

As shown in the example the echo spacing time must always be set in the file. Further parameters within the fitting model can be set in a similar manner. Parameters that might want to be changed could be the the T2 values of the fat in the model, or the initial guess of the muscle T2.

As an example, further items may be added to the file in the following manner to set the long and short T2 components of the fat.

ParamVals:  
 echo:  
 value: 8.0  
 t2\_l:  
 value: 250.0  
 min: 0.0  
 max: 300.0  
 vary: False  
 t2\_s:  
 value: 80.0  
 min: 0.0  
 max: 300.0  
 vary: False

In this example all the attributes of the two T2 fat parameters are set, i.e. value,minimum, maximum and is it varied in the model. The minimum and maximum are set so that they cover the new value for the parameter. If the value is beyond the range present in the original parameter file then it will be incorrectly set. The vary attribute could be omitted from the definition. If is is changed from its original setting within the original parameter file then the model will be changed as more or less parameters are varied during the fit. It is best not to change the ***vary*** attribute from its original setting.

The parameters that can be altered are listed below:

A\_f  
A\_m  
B1  
T1\_f  
T1\_m  
c\_l  
c\_s  
echo  
t2\_fl  
t2\_fs  
t2\_m

The flag ***setParamValsIndividually*** is set to ***True*** in the file in this example. In this case, then fat values are calculated from previous model data in the study directory structure for each participant in the ***fitSubject*** list. This flag can only be set to ***True*** when data is being fit from a study directory structure. When this is the case the path to the fat data should be indicated using the ***fatResultsDir*** flag

11 fatResultsDir: T2/results/fat/fatEPG2

When the flag ***setParamValsIndividually*** is set to ***False*** then all the fat parameters for certain models are set from within the file if they are present, for all participants.

#### ROI Author Preferences :: roiAuthorPreference

The field ***roiAuthorPreferences*** at line 51 gives a list of author initials that are used to choose which ROI files are used.

51 roiAuthorPreference: [EH]

The order of the list gives the priority, if a certain author cannot be found, then the next author initials will be used to choose the correct ROI files. This option is only acted upon when the fitting scripts are used within a study dierctory structure.

#### Use Roi Outline :: useRoiOutline

This option is set to ***True*** when an outline ROI is used to define which part of the image is to be fit.

38 useRoiOutline: False

Outline ROI filenames must include the word ***outline*** in their name for the programs to function correctly

#### Image Data Format :: imageDataFormat

The fitting progams can now read in both analyze and nifti data formats. This must be indicated in the file

52 imageDataFormat: Analyze

For Analyze format the program expects to find the image data as a file, therefore when the fitting program is fitting individual data then a file name with a relative or complete path should be given.

58 analyzeHdr: simpleModelData/WIP\_Forearm\_T2\_CLEAR.hdr  
59 analyzeImg: simpleModelData/WIP\_Forearm\_T2\_CLEAR.img

Nifti data can come in the form of a single file, either zipped or not, or as a series of files, perhaps corresponding to individual echo times for the T2 data. If this is the case then the directory where the data can be found must be entered.

imageDataFormat: nifti  
  
# when individual files   
  
niftiData: C:\Users\NEH69\Dropbox\projects\programming\2019\MRIstudyDescription\simpleModelData\WIP\_Forearm\_T2\_CLEAR.nii.gz  
  
# when a series of nifti files  
  
niftiData: C:\Users\NEH69\Dropbox\projects\programming\2019\MRIstudyDescription\simpleModelData\niffti

## Fitting T2 Data with fitEPGazz.py

The python program has been designed/developed to work within a study data directory structure or with single files. The program can fit a number of different models to the data and works with a fitModelData yaml file for input.

Example fitModelData yaml files can be found in the following directories

mristudydescription/  
│  
└── examples/  
 │  
 │  
    ├── fitSingleModelDataYAMLfiles/  
 │ │  
    │   ├── fitSingleDataMuscleEPG1nifti.yml  
    │   ├── fitSingleDataMuscleAzzabouNifti.yml  
    │   ├── fitSingleDataFatEPG2nifti.yml  
    │   ├── fitSingleDataEPGAzzAnalyzeOutline.yml  
    │   └── fitSingleDataEPGAzzAnalyzeIndividual.yml  
 │  
    └── fitStudyModelDataYAMLfiles/  
 │  
       ├── fitStudyDataEPGAzzAnalyze.yml  
       ├── fitStudyDataEPGAzzAnalyze.yml  
       ├── fitStudyDataFatEPG2analyze.yml  
       ├── fitStudyDataMuscleEPG1analyze.yml  
       └── fitStudyDataMuscleEPG1nifti.yml

The files in the ***fitSingleModelDataYAMLfiles*** directory correspond to fitModelData example yaml files that work on single specified datasets.

The files in the ***fitStudyModelDataYAMLfiles*** directory correspond to fitModelData example yaml files that work on data in a study data directory structure. In this case, example study data directory structures that work with the yaml files are found in

mristudydescription/  
│  
└── examples/  
 │  
    └── studyDirectoyExamples  
 │  
    ├── testStudyAnalyze  
    ├── testStudyNiftDIR  
    └── testStudyNiftiFile

### Fitting T2 Data with fitEPGazz.py on named single data files

To use the program in this manner one would type the following on the command line after moving to the directory where the fitEPGazz.py is located.

-> python fitEPGazz.py examples\fitSingleModelDataYAMLfiles\fitSingleDataFatEPG2nifti.yml

or just the program name on its own and enter the yaml file using the interactive dialog that pops up.

-> python fitEPGazz.py

### Fitting T2 Data with fitEPGazz.py in study directory structures

The program can fit data from within a study structure directory. This can be achieved by calling the program from the command line without arguments and interactive dialogs will pop up to ask for the yaml file and then the study directory name.

-> python fitEPGazz.py

The fitModelData yaml file can be supplied on the command line. If the ***fitSubject*** parameter list is not empty then an interactive dialog will pop up asking for the path of the study structure directory.

-> python fitEPGazz.py examples\fitStudyModelDataYAMLfiles\fitStudyDataAzzabouAnalyze.yml

Finally, the program can be called with two command line arguments specifying the yaml fitModelData file and the study directory structure path.

-> python fitEPGazz.py examples\fitStudyModelDataYAMLfiles\fitStudyDataAzzabouAnalyze.yml   
 examples\studyDirectoyExamples\testStudyAnalyze

## Output from T2 Fitting Program: fitEPGazz.py

The fitModelData YAML file and three data files are produced from the fitting program in CSV and Excel format:

* A results file listing all the fitted data at a pixel level in one table based on slices and ROIs. \_results.csv(.xls)
* A Summary file of the mean and standard deviation for each ROI in each slice. \_summary.csv(.xls)
* An aggregated summary file giving the weighted mean of each parameter for each ROI averaged over slices and pixels in each ROI. \_summaryAgg.csv(.xls)

The YAML file and the three results files are produced for single data sources and data present in a study directory structure.

When the data is derived from a study directory structure, the filename includes the subject, the session, the imaged region, and the model used together with the appropriate descriptor to indicate the type of data in the file. The files are best saved in a *T2/results/imagedRegionType/model* directory at the appropriate point in study directory structure for the fitted data. This should be set in the ***fitModelData*** file.

3 # use a relative directory path when fitting data in study directory structure  
4 # when fitting individual files it can be set to a complete path  
5   
6 resultsDir: T2/results/muscle/AzzEPG

When data is derived from a single file, the filename includes the model used and appropriate descriptor for the file. The directory where the data is stored is given in the fitModelData file. A relative or absolute path can be used.

When the T2 fitting program is used on single file data, then plots of the ROIs overlayed over the imaged region are displayed in in a single plot (Figure 15) and a representative plot of the fitting results (Figure 16) for a single pixel data set is displayed in a second independent plot taken from data from the middle of the data set.

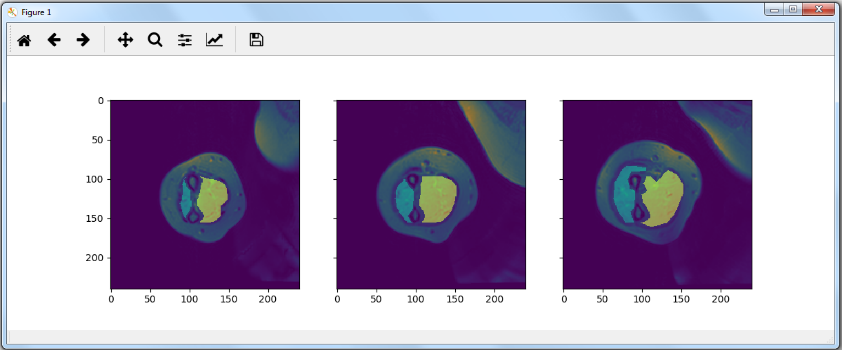


Figure 15 Overlay of ROIs on fitted Image

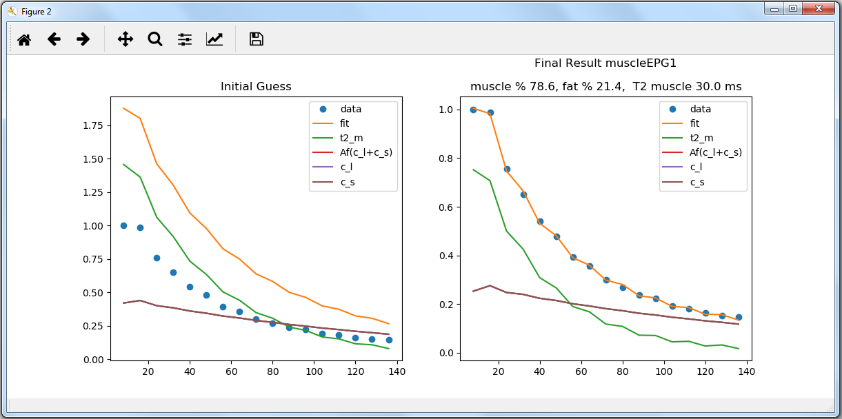


Figure 16 Quality of fit to data

## Naming Conventions for ROI File-names

One of the aims of the scripts produced during this work was to minimize the variation in filenames for such things as ROIs or results files. This is why it is stressed that researchers should use the stury directory approach even for small data sets or when a study is in development and being scoped. However, the authors are aware that on occasion it may be more convenient to work on single datasets in a less structured manner. However, for the fitting programs to work, the ROI files must follow a minimum naming structure. The details are outlined below.

* If the ROIs are outline ROIs then this must be present in the zipped file-name of the set of ROIs encased in underscores, \_outline\_.
* It is a good idea to add what the imagedType the ROIs are of. For example, \_muscle\_, \_fat\_, \_phantom\_, \_brain\_.

The individual ROIs within the zipped file should also follow a certain naming convention. This should include the following:

* The slice number should be included, starting from an index of one and be identified by adding the word slice and connecting the number with a hyphon, for example, slice-1\_, or \_slice-2\_ and separated from other parts of the filename with an underscore.
* The word or number identifying the ROI should be placed at the end of filename before the extension descriptor. For example slice-1\_muscle\_roi1.roi or slice-1\_fat\_1.roi or slice1\_muscle\_extensor.roi. The ROI identifier must be unique within a set of ROIs.
* It is also a good idea to add the initials of the person who drew the ROI. For example, KGH\_slice-1\_muscle\_extensor.roi. This is useful for book-keeping purposes.

When data is derived from a single file, the filename includes the model used and appropriate descriptor for the file. The directory where the data is stored is given in the fitModelData file. A relative or absolute path can be used.

## Appendix A. Model Parameter Files and Fitting Equations

Table A.1 EPGAzz :: azz\_fatmuscle\_epg\_model\_params.json

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Name | Value | Min | Max | Vary |  |  |
| A\_f | 0.2 | 0 | 2 | True |  | (A.1) |
| A\_m | 0.8 | 0 | 2 | True |
| T1\_m | 1400.0 | 0 | 4000 | False |
| T1\_f | 365.0 | 0 | 1000 | False |
| t2\_m | 40.0 | 0 | 100 | True |
| c\_l | 0.6 | 0 | 2 | False |
| c\_s | 0.5 | 0 | 2.0 | False |
| t2\_fl | 250.0 | 0 | 2000 | False |
| t2\_fs | 50.0 | 0 | 100 | False |
| B1 | 1.0 | 0 | 2 | True |
| echo | 10.0 | 0 | 30 | False |

Table A.2 fatEPG2 :: two\_fat\_epg2\_model\_params.json

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Name | Value | Min | Max | Vary |  | (A.2) |
| A\_f | 1.0 | 0 | 2 | False |
| A\_m | 0.0 | 0 | 2 | False |
| t2\_m | 40.0 | 0 | 100 | False |
| T1\_m | 1400.0 | 0 | 4000 | False |
| T1\_f | 365.0 | 0 | 1000 | False |
| c\_l | 0.2 | 0 | 2 | True |
| c\_s | 0.8 | 0 | 2.0 | True |
| t2\_fl | 500.0 | 0 | 2000 | True |
| t2\_fs | 100.0 | 0 | 2000 | True |
| B1 | 1.0 | 0 | 2 | True |
| echo | 10.0 | 0 | 30 | False |

Table A.3 fatEPG1 :: one\_fat\_epg1\_model\_params.json

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Name | Value | Min | Max | Vary |  | (A.3) |
| A\_f | 1.0 | 0 | 2.0 | False |
| A\_m | 0.0 | 0 | 2.0 | False |
| t2\_m | 40.0 | 0 | 100 | False |
| T1\_m | 1400.0 | 0 | 4000 | False |
| T1\_f | 365.0 | 0 | 1000 | False |
| c\_l | 1.0 | 0 | 2.0 | True |
| c\_s | 0.0 | 0 | 2.0 | False |
| t2\_fl | 250.0 | 0 | 2000 | True |
| t2\_fs | 50.0 | 0 | 100 | False |
| B1 | 1.0 | 0 | 2 | True |
| echo | 10.0 | 0 | 30 | False |

Table A.4 muscleEPG1 :: one\_fatmuscle\_epg1\_model\_params.json

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Name | Value | Min | Max | Vary |  | (A.4) |
| A\_f | 0.2 | 0 | 2 | True |
| A\_m | 0.8 | 0 | 2 | True |
| T1\_m | 1400.0 | 0 | 4000 | False |
| T1\_f | 365.0 | 0 | 1000 | False |
| t2\_m | 40.0 | 0 | 100 | True |
| c\_l | 1.0 | 0 | 2 | False |
| c\_s | 0.0 | 0 | 2.0 | False |
| t2\_fl | 140.0 | 0 | 2000 | False |
| t2\_fs | 50.0 | 0 | 100 | False |
| B1 | 1.0 | 0 | 2 | True |
| echo | 10.0 | 0 | 30 | False |

Table A.5 muscleEPG2 :: two\_fatmuscle\_epg2\_model\_params.json

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Name | Value | Min | Max | Vary |  | (A.5) |
| A\_f | 0.2 | 0 | 2 | True |
| A\_m | 0.8 | 0 | 2 | True |
| T1\_m | 1400.0 | 0 | 4000 | False |
| T1\_f | 365.0 | 0 | 1000 | False |
| t2\_m | 40.0 | 0 | 100 | True |
| c\_l | 1.0 | 0 | 2 | False |
| c\_s | 0.0 | 0 | 2.0 | False |
| t2\_fl | 250.0 | 0 | 2000 | False |
| t2\_fs | 50.0 | 0 | 100 | False |
| B1 | 1.0 | 0 | 2 | True |
| echo | 10.0 | 0 | 30 | False |

Table A.6 phantomEPG1 :: oneParamEPGphantom\_model\_params.json

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Name | Value | Min | Max | Vary |  | (A.6) |
| A\_f | 0.0 | 0 | 2.0 | False |
| A\_m | 1.5 | 0 | 1.75 | True |
| t2\_m | 100.0 | 0 | 1000 | True |
| T1\_m | 1400.0 | 0 | 4000 | False |
| T1\_f | 0.0 | 0 | 1000 | False |
| c\_l | 0.0 | 0 | 2.0 | False |
| c\_s | 0.0 | 0 | 2.0 | False |
| t2\_fl | 0.0 | 0 | 2000 | False |
| t2\_fs | 0.0 | 0 | 100 | False |
| B1 | 0.8 | 0.5 | 1.1 | True |
| echo | 8.0 | 0 | 30 | False |