
quantiphyse Documentation

Release 0.31

Ben Irving, Martin Craig

Jan 23, 2019

Contents

1	License	3
2	Bugs/Issues	5
3	User Guide	7
3.1	Basic functions	7
3.2	Generic analysis and processing tools	14
3.3	Current included plugins	26
3.4	Advanced Tools	55

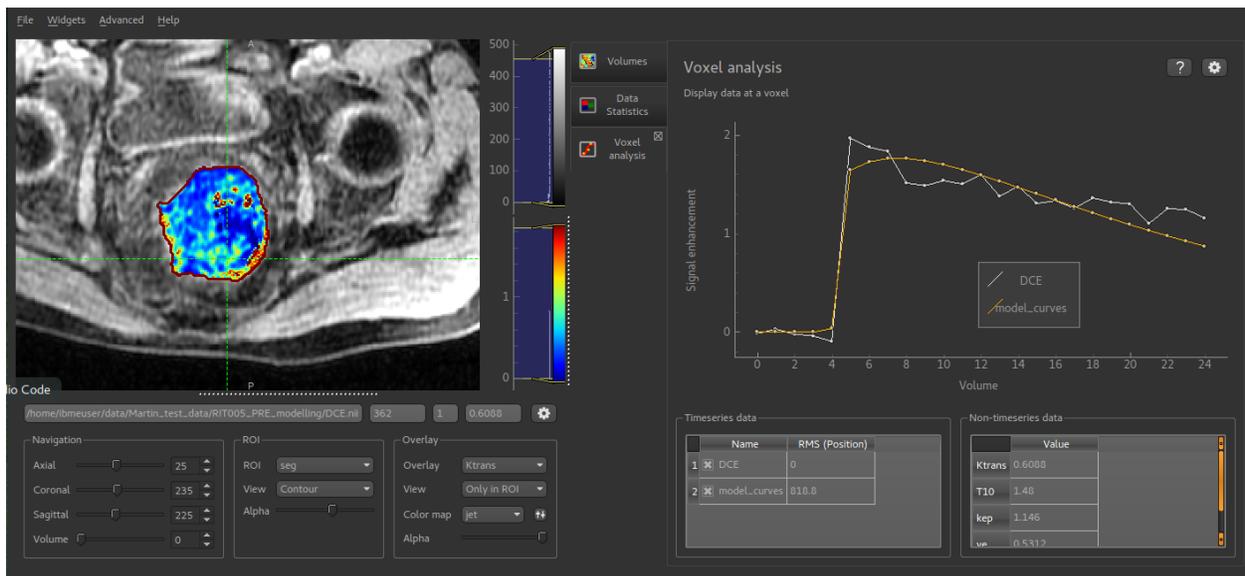
Contributors

- [Martin Craig](#) (Current maintainer)
- [Ben Irving](#) (Original author)
- [Michael Chappell](#)
- [Paula Croal](#)

Acknowledgements

- [Julia Schnabel](#)
- [Sir Mike Brady](#)

Quantiphyse is a viewing and analysis tool for 3D and 4D biomedical data. It is particularly suited for physiological or functional imaging data comprised of multi volumes in a 4D (time-) series and/or multimodal imaging data. Quantiphyse is built around the concept of making spatially resolved measurements of physical or physiological processes from imaging data using either model-based or model-free methods, in a large part exploiting Bayesian inference techniques. Quantiphyse can analyse data both voxelwise or within regions of interest that may be manually or automatically created, e.g. supervoxel or clustering methods.



Features include:

- 2D orthographic viewing and navigation of data, regions of interest (ROIs) and overlays
- Universal analysis tools including clustering, supervoxel generation and curve comparison
- Tools for CEST-MRI analysis and modelling
- Tools for DCE-MRI analysis and modelling
- Tools for ASL-MRI analysis and modelling
- Tools for ROI generation
- Registration and motion correction
- Extensible via plugins, in the future to include ASL MRI data and Bayesian modelling and more

Quantiphyse is available via the [Oxford University Innovation Software Store](#)

This documentation is based on version 0.6

CHAPTER 1

License

Quantiphyse is available free under an academic (non-commercial) license. See the [OUI Software Store](#) for more details.

CHAPTER 2

Bugs/Issues

Please report bug, issues, feature requests or other comments to the [current maintainer](#):

3.1 Basic functions

3.1.1 Overview

Quantiphyse is a visual tool for quantitative analysis of medical images. The aim is to bring advanced analysis tools to users via an easy-to-use interface rather than focusing on the visualisation features themselves. The software is also designed to support advanced usage via non-GUI batch processing and direct interaction with the Python code.

Quantiphyse works with two types of data:

- 3D / 4D data sets.
- Regions of interest (ROIs). These must be 3D and contain integer data only. Voxels with the value zero are taken to be outside the region of interest, nonzero values are inside. ROIs with more than one nonzero value describe multi-level regions of interest which are handled by some of the tools.

One data set is used as the main data. This defaults to the first 4D data to be loaded, or the first data to be loaded, however you can set any data set to be the main volume.

Data orientation

Quantiphyse keeps all data loaded from files in its original order and orientation. For display purposes, it takes the following steps to display data consistently:

- A display grid is derived from the grid on which the main data is defined. This is done by flipping and transposing axes only so the resulting grid is in approximate RAS orientation. This is the *display grid* and ensures that the right/left/anterior/posterior/superior/inferior labels are in a consistent location in the viewer. Note that the main data does *not* need resampling on to the display grid as only transpositions and flips have occurred.
- Data which is defined on a different grid will be displayed relative to the display grid. If possible this is done by taking orthogonal slices through the data and applying rotations and translations for display. In this case no resampling is required for display.

- If the data cannot be displayed without taking a non-orthogonal slice, this is done by default using nearest neighbour interpolation. This is fast, and ensures that all displayed voxels represent ‘real’ raw data values. However, display artifacts may be visible where the nearest neighbour changes from one slice to another.
- To avoid this, the viewer options allow for the use of linear interpolation for slicing. This is slightly slower but produces a more natural effect when viewing data items which are not orthogonally oriented.
- The main data is, by default, the first data loaded, however any data set can be set as the main data.

It is important to reiterate that these steps are done for *display* only and do not affect the raw data which is always retained.

Analysis processes often require the use of multiple data items all defined on the same grid. When this occurs, typically they will resample all the data onto a single grid (usually the grid on which the main data being analysed is defined). For example if fitting a model to an ASL dataset using a T1 map defined on a different grid, the T1 would be resampled to the grid of the ASL dataset. Normally this would be done with linear interpolation however cubic resampling is also available. This is the decision of the analysis process. The output data would then typically be defined on the same grid, however again this is the choice of the analysis process.

Special cases

Quantiphyse will try to handle some special cases which would otherwise prevent data being loaded and processed properly.

Multi-volume 2D data

Some data files may be 3 dimensional, but must be interpreted as multiple 2D volumes (e.g. a time series) rather than a single static 3D volume. When a 3D data set is loaded, an option is available to interpret the data as 2D multi-volumes. To access this option, click the *Advanced* checkbox in the data choice window.

Note: In Nifti files the first 3 dimensions are required to be spatial, so where this occurs with a Nifti file it implies that the file is incorrectly formed and you should ideally correct the acquisition step which produced it.

3.1.2 Loading and Saving Data

File Formats

This software package works with NIFTI volumes. Some builds may contain experimental support for folders of DICOM files, however this is not well tested.

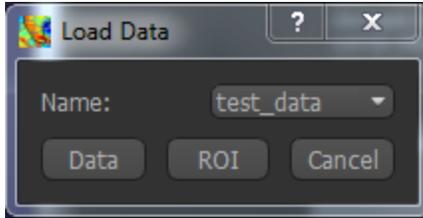
Alternative packages which are able to convert DICOM files to NIFTI include the following:

- `itk-snap`
- `dcm2nii`
- Or the batch version which allows a number of volumes to be converted `dcm2niibatch`

Loading data

Drag and drop

You can drag and drop single or multiple files onto the main window to load data. You will be prompted to choose the type of data:



The suggested name is derived from the file name but is modified to ensure that it is a valid name (data names must be valid Python variable names) and does not clash with any existing data.

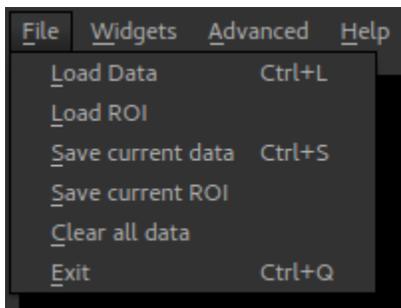
If you choose a name which is the same as an existing data set, you will be asked if you wish to overwrite the existing data.

When dropping multiple files you will be asked to choose the type of each one. If you select *cancel* the data file will not be loaded.

Menu

The following menu option can be used to load data files:

- File -> Load



You will be prompted to choose the file type (data or ROI) and name in the same was as drag/drop.

Saving Data

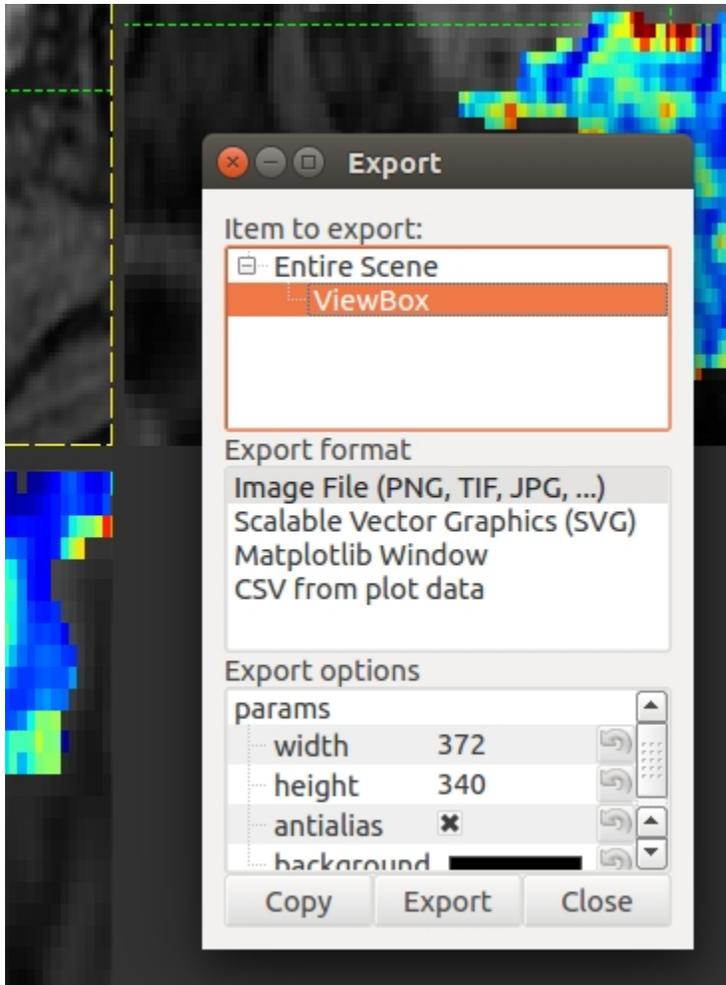
The following menu options are used for saving data:

- File -> Save current data
- File -> Save current ROI

So, to save a data set you need to make it the current data, using the Overlay menu or the Volumes widget. Similarly to save an ROI you need to make it the current ROI. Saving the main data can be done by selecting it as the current overlay.

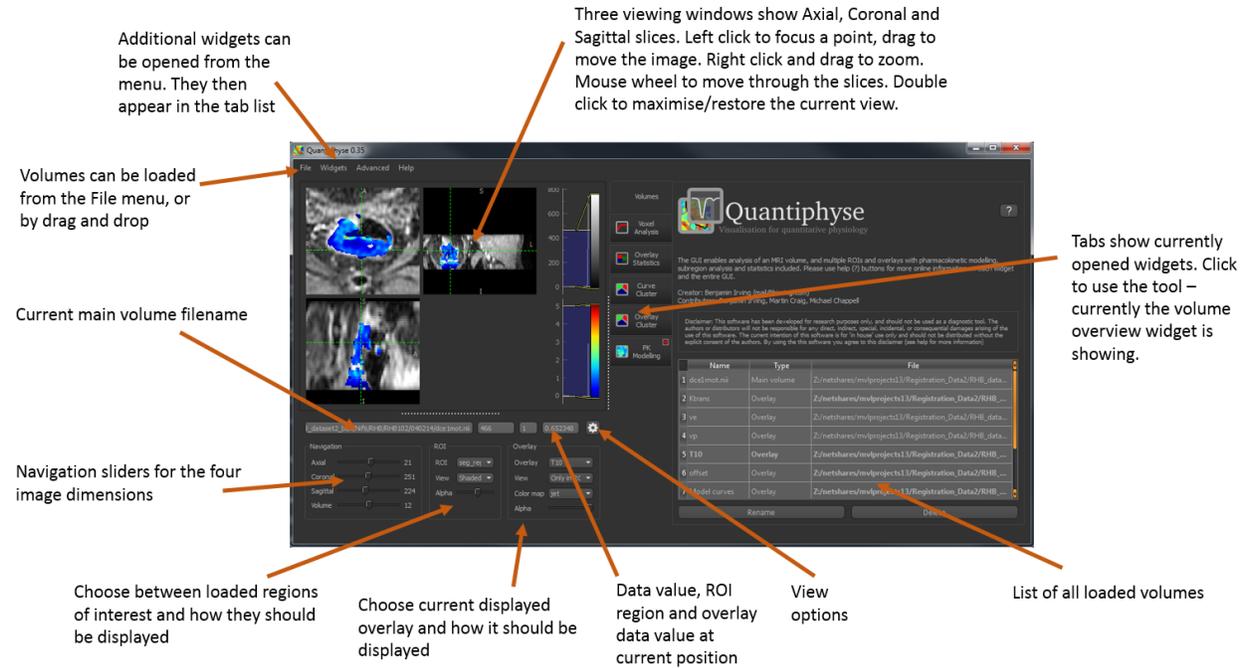
Save a screen shot or plot

- Right click on an image or plot
- Click *Export*
- A view box will appear with the various format options.
- *svg* format will allow editing of the layers and nodes in inkscape or another vector graphics viewer.

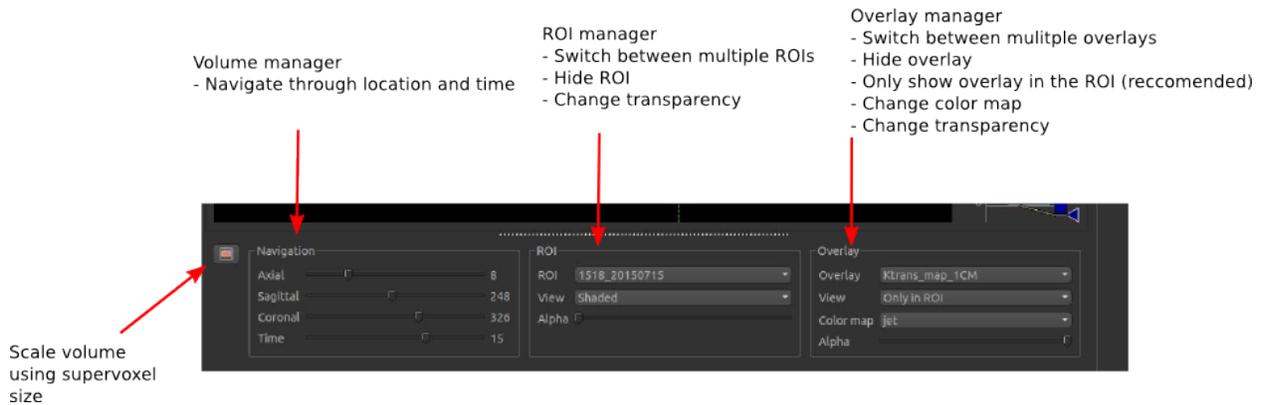


3.1.3 General Interaction

The Main Window



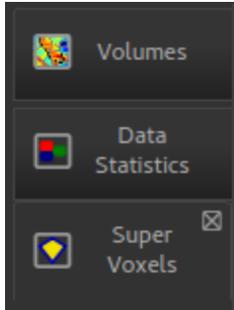
The Navigation Bar



Using Widgets

Widgets appear to the right of the viewer window. Most widgets are accessed from the 'Widgets' menu above the viewer.

When selected, a widget will appear with a tab to the right of the viewer. You can switch between opened widgets by clicking on the tabs. A widget opened from the menu can be closed by clicking on the X in the top right of its tab.



Widgets may have very different user interfaces depending on what they do, however there are a number of common elements:

Help button



This opens the online documentation page relevant to the widget. Internet access is required.

Options button



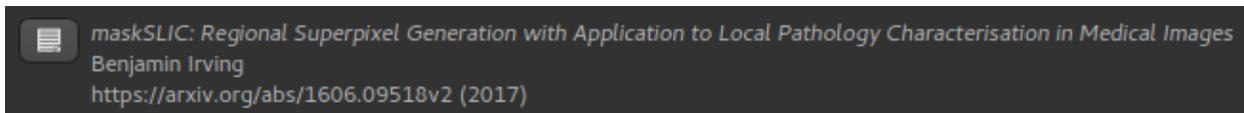
This shows any extended options the widget may have. It is typically used by widgets which display plots as that limits the space available for options.

Batch button



This displays the batch code required to perform the widget's processing, using the currently selected options. This can be useful when building batch files from interactive exploration. It is only supported by widgets which provide image processing functions.

Citation



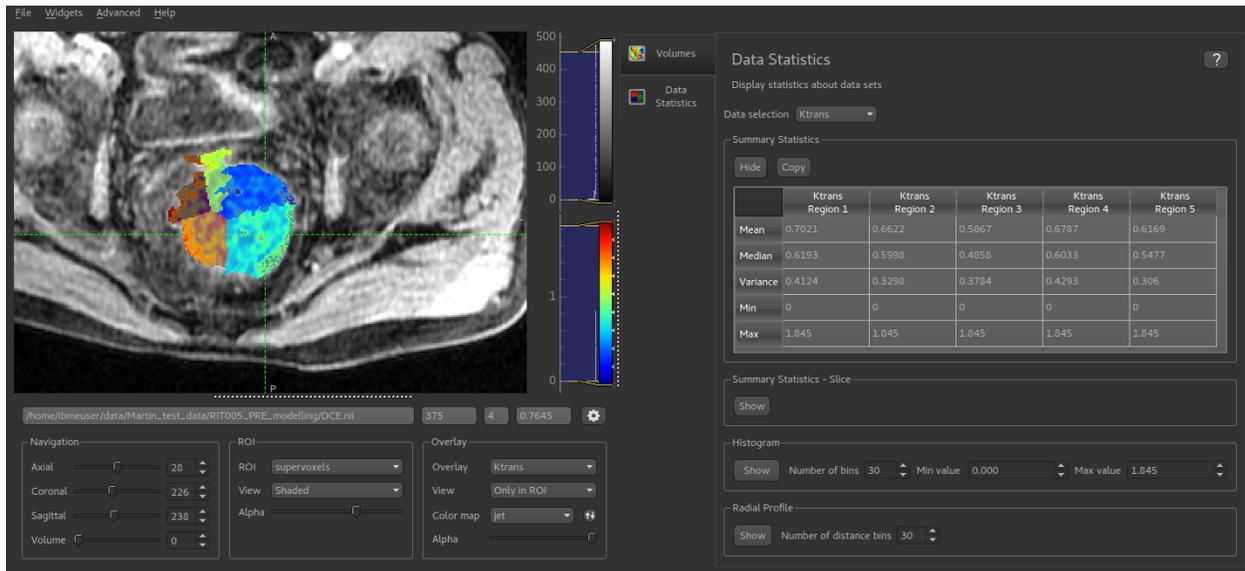
Many widgets are based around novel data processing techniques. The citation provides a reference to a published paper which can be used to find out more information about the underlying method. If you publish work using a widget with a citation, you should at the very least reference the paper given.

Clicking on the citation button performs an internet search for the paper.

3.1.4 Data Statistics

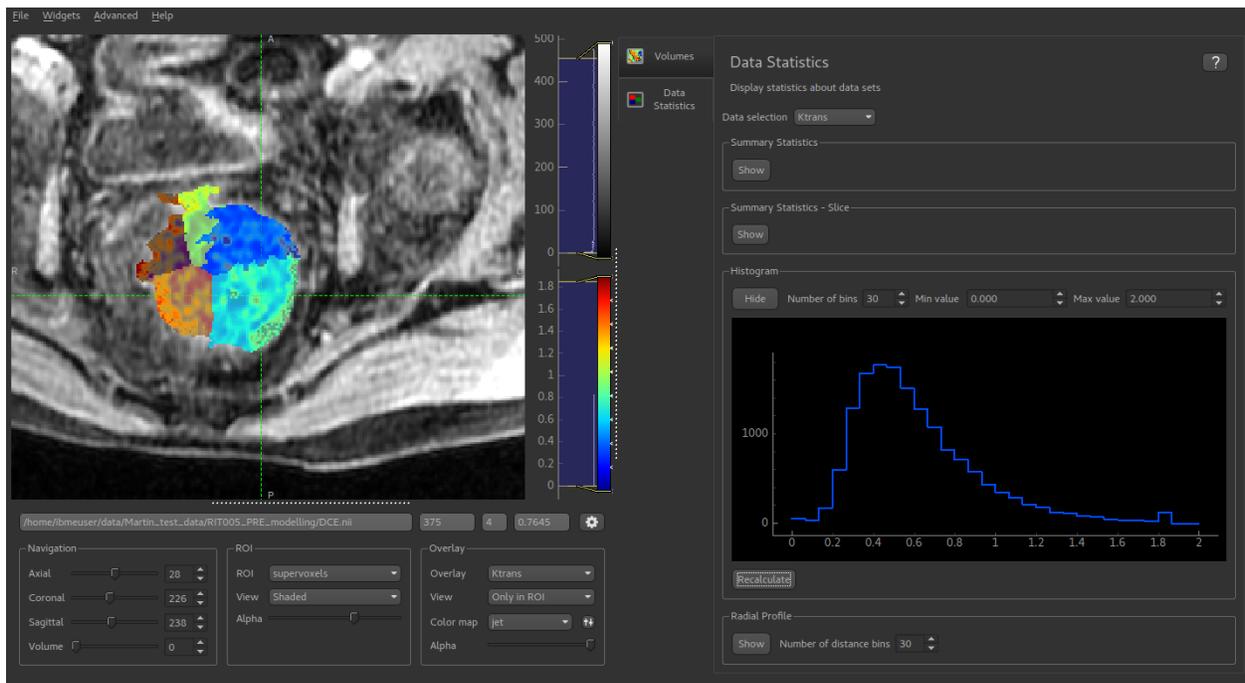
This widget displays data statistics for each region in the current ROI. You can choose to either display statistics for the current data item or for all data items.

The top table displays global statistics across the whole overlay volume - Mean, Median, Variance, Min and Max. The 'Single Slice' table shows the same statistics for the current slice in the selected direction.



The 'Copy' button for each table copies the data to the clipboard in a tab-separated form which should be suitable for pasting into spreadsheets such as Excel.

In addition, a radial profile and histogram can be plotted for the current data item.



3.2 Generic analysis and processing tools

3.2.1 Voxel analysis

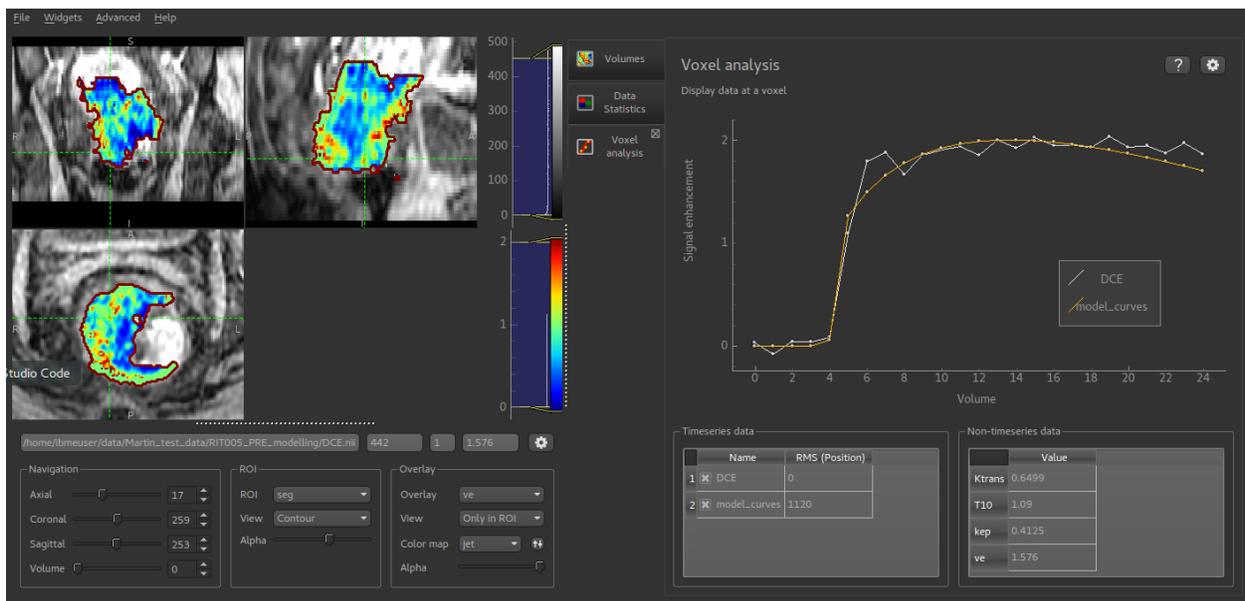
- *Widgets -> Analysis -> Voxel Analysis*

This widget shows data at the selected voxel. This includes a plot of time-series data, and the point values of 3D data sets. Selecting voxels in the viewer window updates the displayed data.

Time series data is often generated by a modelling processes, for example the PK modelling widget, so this widget enables enable the model prediction to be compared to the original data. Below the plot a table shows all the 4D data sets loaded. The checkboxes control whether a given data set is included in the plot or not.

A second table shows the value of each 3D data set at the focus point. This is useful after running modelling as the model parameters can be viewed at the same time as comparing the overall quality of the model fit.

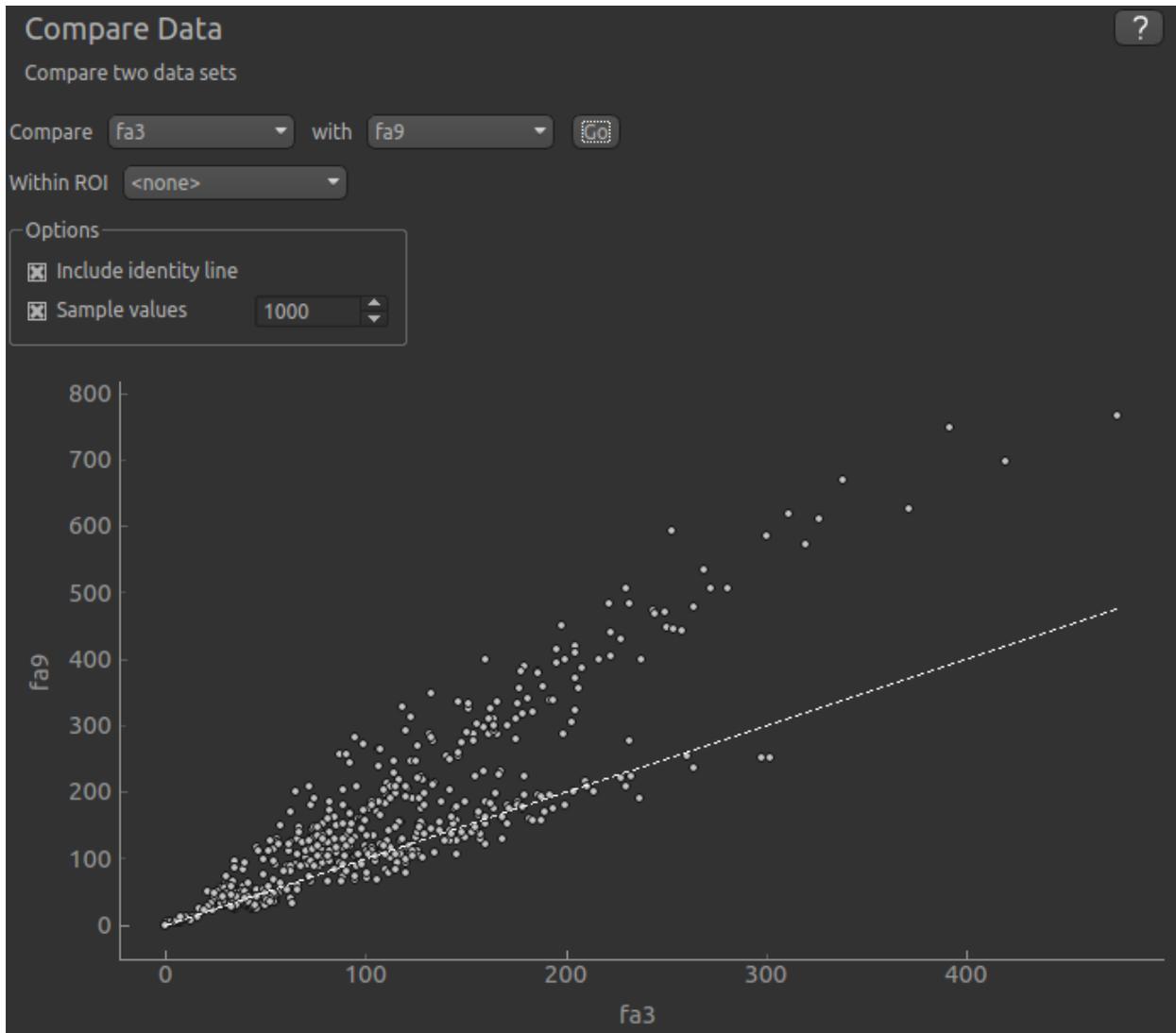
The options button allows the behaviour of the plot to be changed. You can choose to plot either the raw data or to transform the timeseries data to signal enhancement curves. This uses the selected number of volumes as ‘baseline’ and scales the remainder of the data such that the mean value of the baseline volumes is 1.



3.2.2 Compare data

- *Widgets -> Analysis -> Compare Data*

This widget shows a comparison between two data sets. Select the two data sets you are interested in from the menus and click Go to display a scatter plot of corresponding voxel values.



Options

You can choose to restrict the comparison to voxels within a specified ROI.

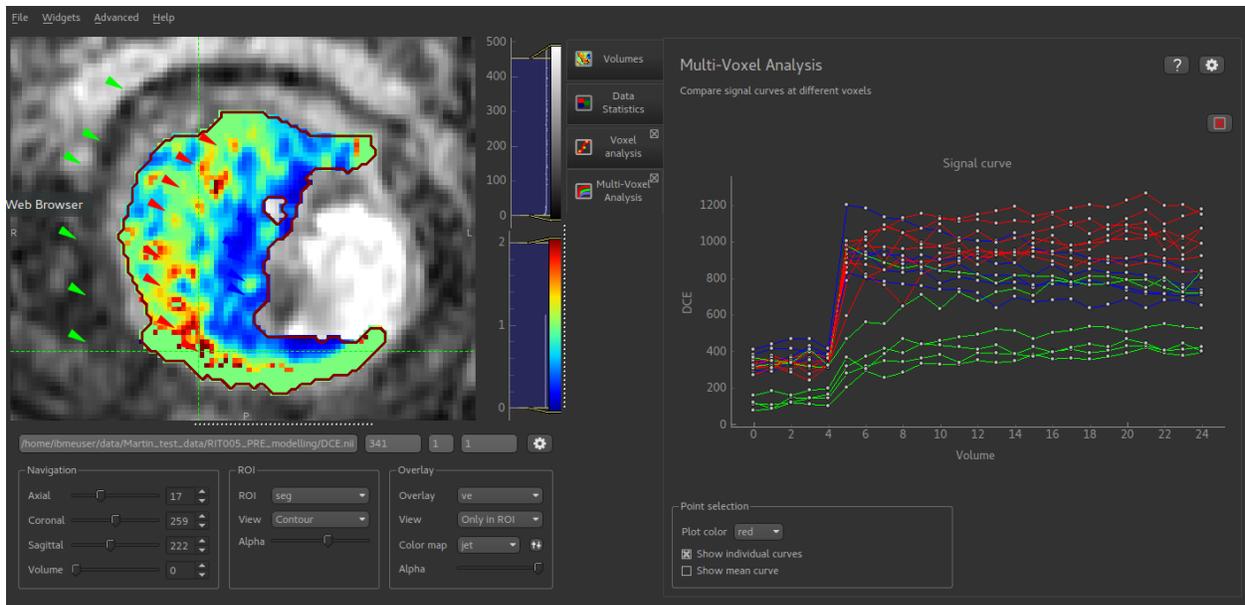
By default only a random sample of 1000 points is displayed. This is because the scatter plot can take a long time to generate otherwise. You can choose the number of points in the sample. If you want to use all values in comparison, turn off the sample, but be aware that the plot may take some time to generate for large or 4D data sets.

An identity line can be shown in cases where you want to compare the data for equality.

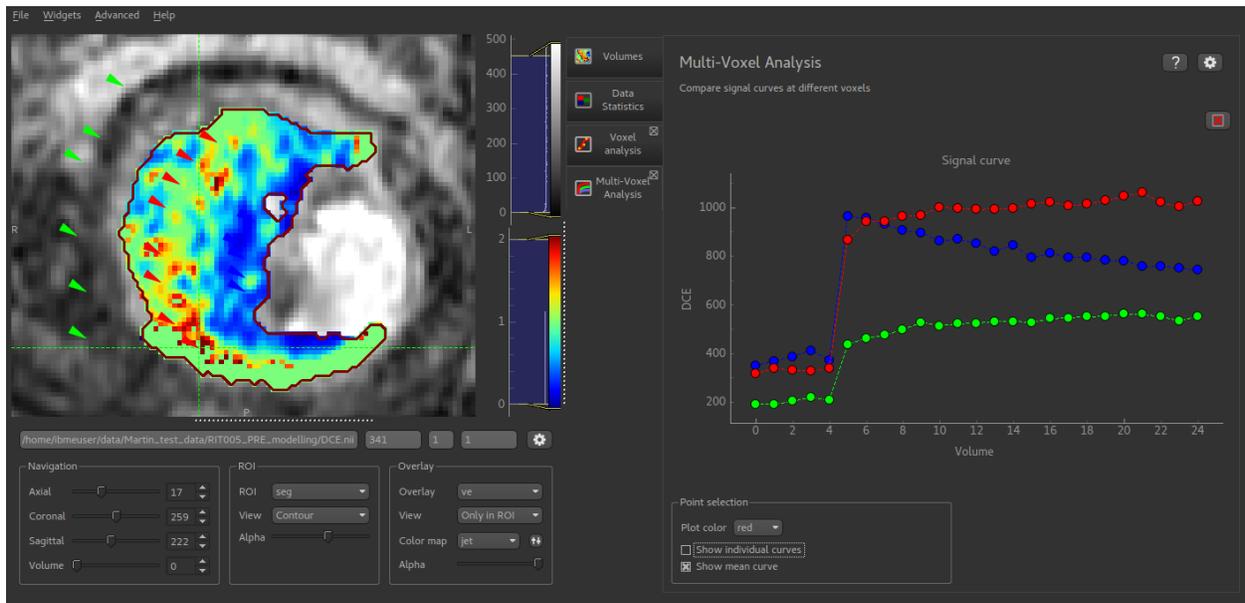
3.2.3 Multi-voxel analysis

Widgets -> Analysis -> Multi-voxel Analysis

This widget shows the signal-time curve at multiple locations.



- Each click on the image adds a new curve to the plot. By changing the colour, a series of curves can be plotted enabling different parts of the image to be compared
- The plot can be cleared by clicking on the red X at the top right of the window
- The mean curve for each color can also be displayed. This is shown with large circular markers and a dotted line. This can be displayed with the individual curves, or on its own (as below)



Additional plot options are available by clicking the Options button in the top right.

3.2.4 Simple maths

Widgets -> Processing -> Simple Maths

This widget is a simplified version of the console and allows new data to be created from simple operations on existing data.

The text entered must be a valid Python expression and can include the names of existing ROIs and overlays which will be Numpy arrays. Numpy functions can be accessed using the `np` namespace.

Examples

Add Gaussian noise to some data

```
newdata = mydata + np.random.normal(0, 100)
```

Calculate the difference between two data sets

```
newdata = mydata1 - mydata2
```

Scale data to range 0-1

```
newdata = (mydata - mydata.min()) / (mydata.max() - mydata.min())
```

3.2.5 Registration and Motion Correction

Widgets -> Processing -> Registration

This widget enables registration and motion correction using various methods. Currently implemented methods are:

- DEEDS - a nonlinear fully deformable registration method
- MCFLIRT - a linear affine/rigid body registration method

Not all methods may be included in all builds.

In Registration mode you must select the data to register and the reference data. The reference data must be a static volume, if multi-volume data is selected you have to choose which volume to use. The options are:

- Middle volume, i.e. if there are 5 volumes, use the third.
- Mean volume, i.e. take the mean of all volumes in the series and use that
- Specified volume, allowing the user to choose which volume to use. Note that the first volume is numbered 0.

The registration data can be single or multi-volume. If it is multi-volume, each volume is registered to the reference data.

Motion correction is implemented as self-registration. Multi-volume data is selected and each volume is registered to a specific volume within the same series, or the mean of the series. The options for this are as listed above.

The registered data can be saved under a specified name, or can be set to replace the original data.

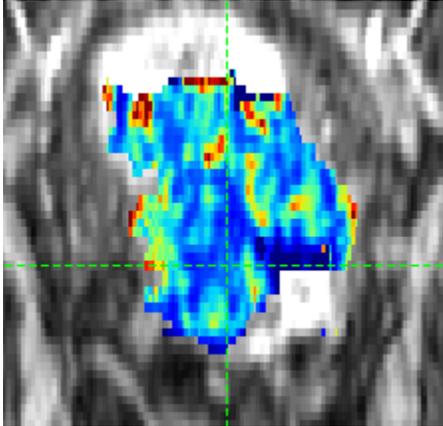
Each registration method has its own set of options which are available when it is selected.

3.2.6 Smoothing

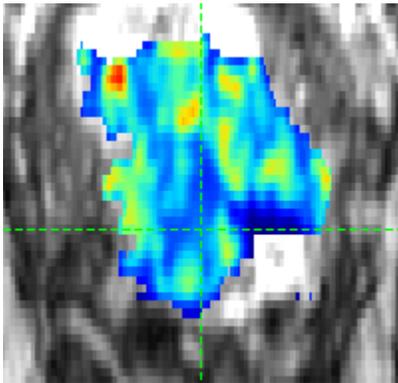
Widgets -> Processing -> Smoothing

This widget provides simple Gaussian smoothing.

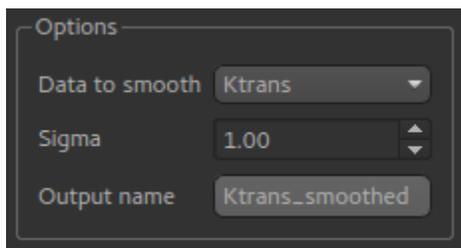
Sample input



Sample output



Options



Sigma is the standard deviation of the Gaussian used in the convolution.

3.2.7 K-Means Clustering

Widgets -> Clustering -> KMeans Clustering

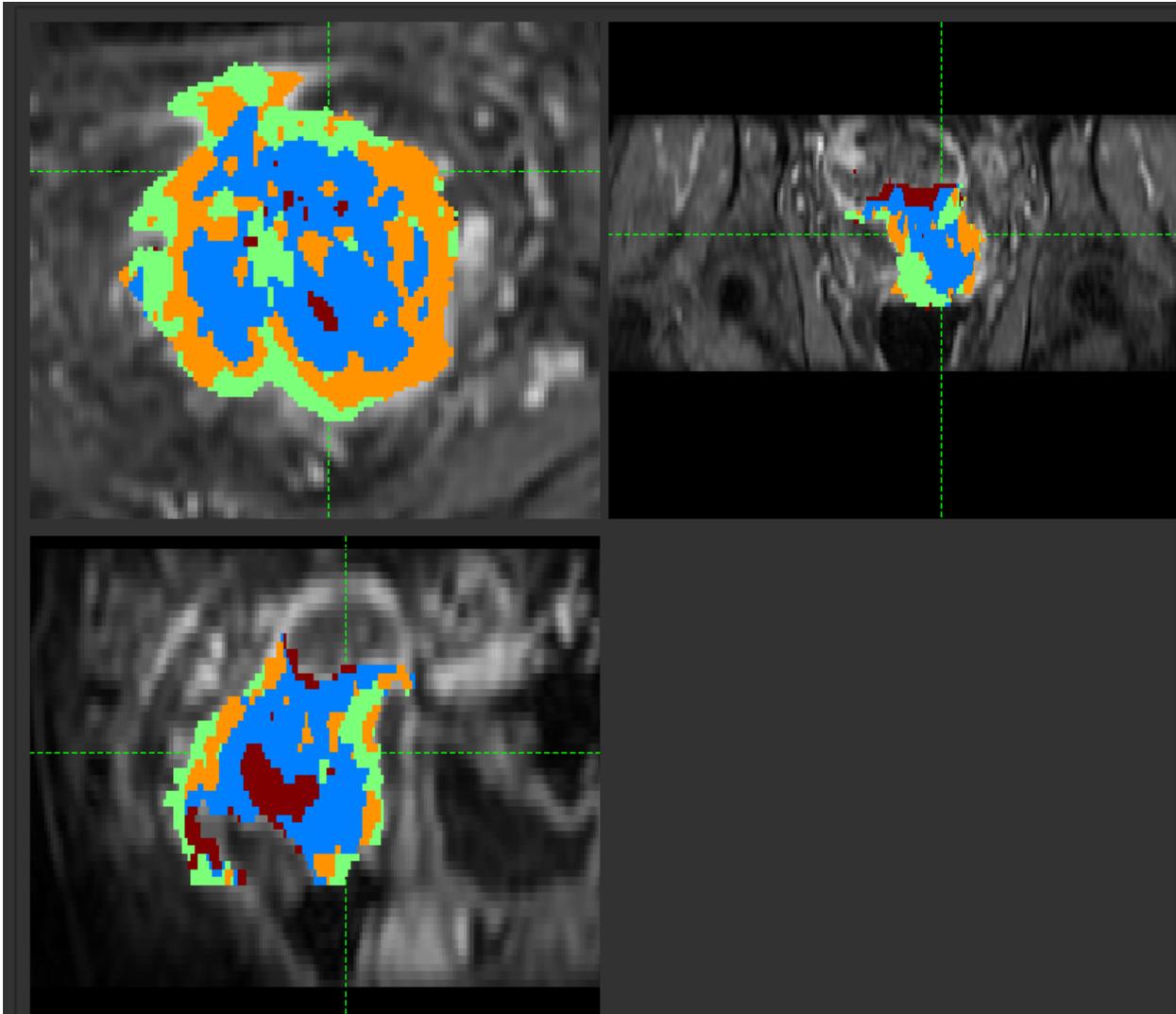
Clustering uses the K-Means algorithm to cluster 3D or 4D data into discrete regions.

When used with 4D data, PCA reduction is used to convert the volume sequence into 3D data before K-Means is applied.

Options

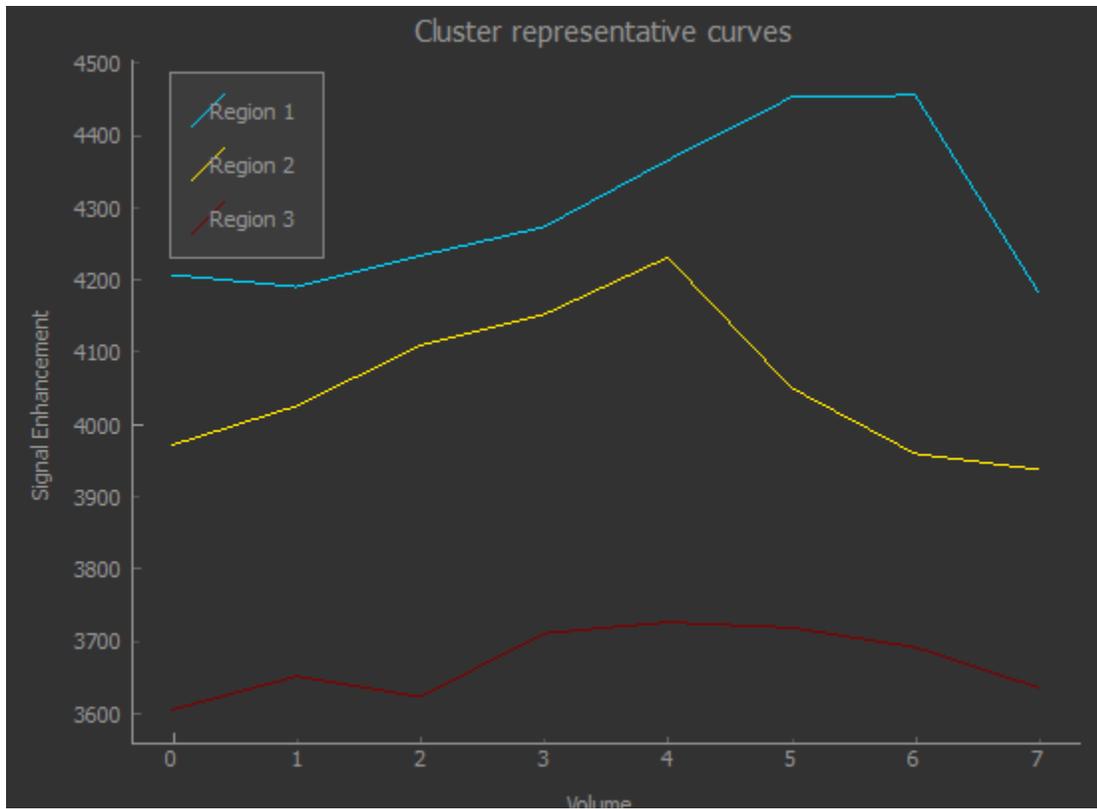
- Data set to use for clustering (defaults to current overlay)
- ROI to cluster within (optional, but recommended for most data)
- Name of the output ROI data set
- Number of clusters, i.e. how many subregions the ROI will be split into
- For 4D data, the number of PCA modes to use for reduction to 3D.

On clicking `Run`, a new ROI is produced with each cluster assigned to an integer ID.



Show representative curves

This option is available when clustering 4D data, and displays the mean time-series curves for each cluster.



In this case the clusters correspond to two distinct phase offsets of the signal curve, with a third cluster picking up voxels with weak or no signal.

Show voxel counts

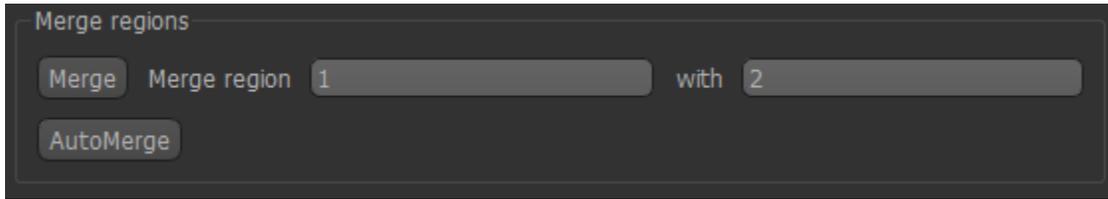
This option shows the number of voxels in each cluster, overall and within the current slice.

Voxel count			
	Region 1	Region 2	Region 3
Slice	203	181	143
Volume	1578	1660	1196

Show merge options

Having generated clusters, it may be desirable to merge some of the subregions - for example if two are very similar, or one contains very few voxels. The Merge tool allows you to do this by specifying the two regions to be merged.

An `Auto Merge` tool is also provided to automatically identify subregions for merging.



3.2.8 Supervoxel widget

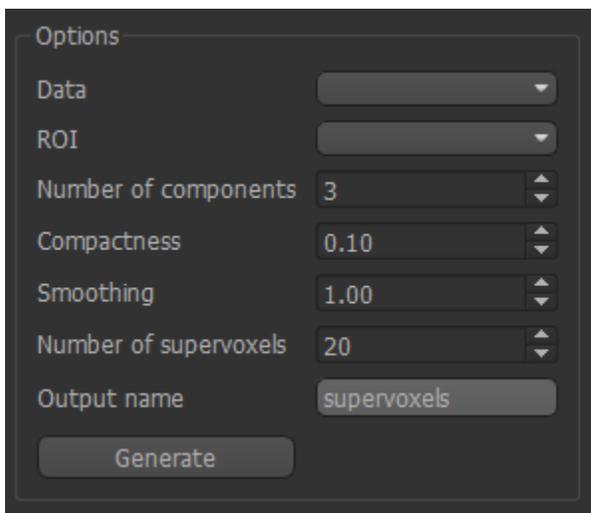
Widgets -> Clustering -> Supervoxels

This widget create supervoxels based a selected data map and a selected ROI.

Supervoxels are collections of voxels which are similar in terms of both data and also spatial location. So, unlike clusters, supervoxels are intended to be connected and localised.

Quantiphyse uses a novel supervoxel method based on SLIC, but modified so that it can be applied sensibly to data within an ROI. For full method details see <https://arxiv.org/abs/1606.09518v2>

Options



The following options are available:

- `Data` can be 3D or 4D data
- `ROI` Select the ROI within which the supervoxels will be generated
- `Number of components` PCA analysis is initially performed to reduce 4D data to a 3D volume, as with the clustering widget. This option controls the number of PCA components and is only visible for 4D data.
- `Compactness` This takes values between 0 and 1 and balances the demands of spatial regularization with similarity of data. A high value will produce supervoxels dominated by spatial location, a low value will produce results similar to clustering with irregular and possibly disconnected supervoxel regions.
- `Smoothing` Degree of smoothing to apply to the data prior to supervoxel generation
- `Number of supervoxels` This is the number of seed points which are placed within the ROI, each of which will define a supervoxel.

- `Output name` The data will be output as a new ROI with this name where each supervoxel is a separate numbered region.

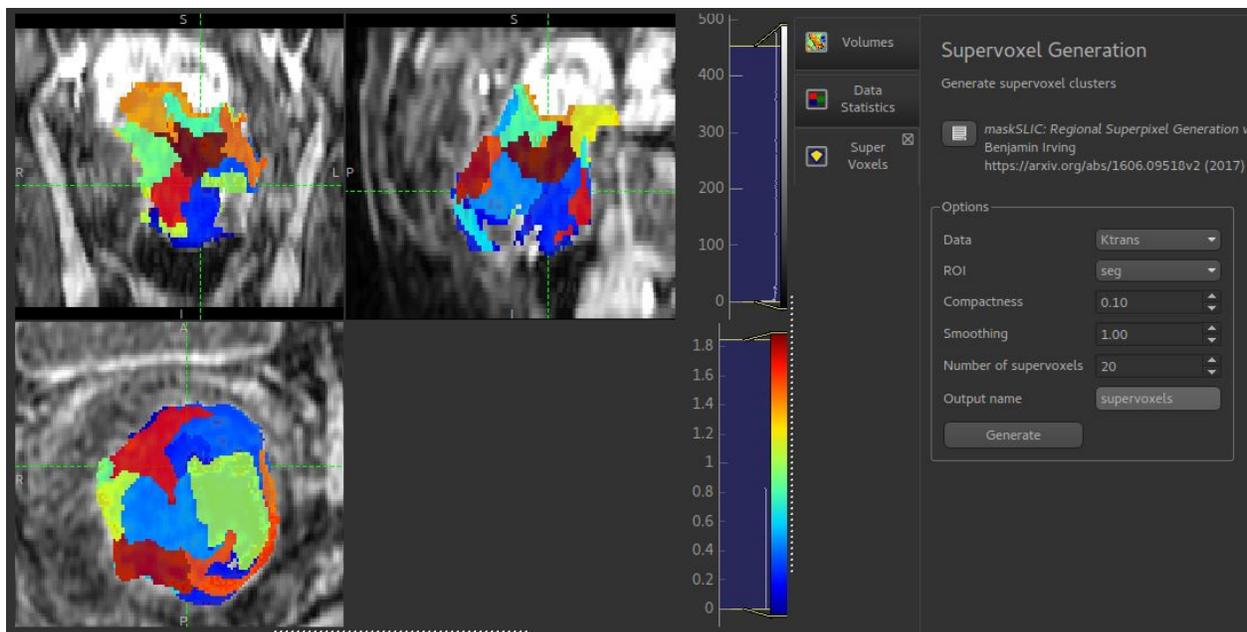
Method

Seed points are placed within the ROI - one for each supervoxel - with initial positions determined by the need to be within the ROI and maximally separated from other seed points and the boundary.

For 4D data, PCA analysis is initially performed as described above. For 3D data, the only preprocessing is a scaling of the data to the range 0-1 to enable parameters such as compactness to have consistent meaning.

The output is an ROI in which each supervoxel is an ROI region. This enables use with for example, the mean values widget, which can replace the data in an overlay with a single value for each supervoxel.

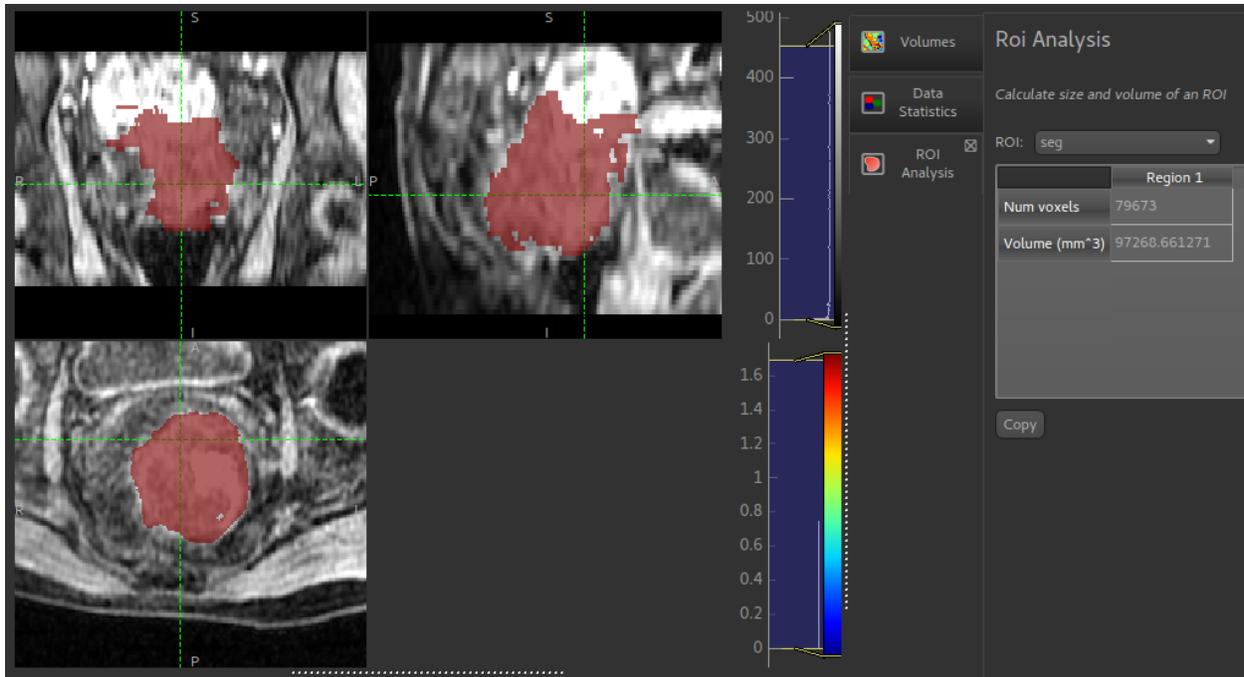
Sample output



3.2.9 ROI Analysis

Widgets -> ROIs -> ROI Analysis

This widget performs a simple calculation of volume and number of voxels within each ROI region.



The output table can be copied and pasted into most spreadsheet applications (e.g. Excel)

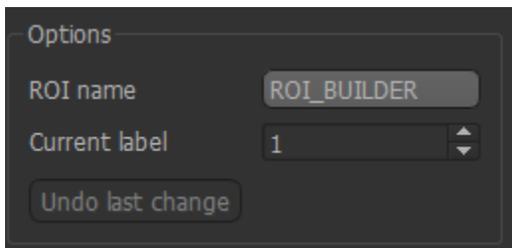
3.2.10 ROI Builder

- *Widgets -> ROIs -> ROI Builder*

This widget is designed for simple construction of regions of interest and manual segmentation. It is not designed to be a replacement for sophisticated semi-automatic segmentation tools! However it is very helpful when running intensive analysis processes as you can easily define a small ROI to run test analyses within before you process the full data set.

Basic concept

At the top of the widget, you can choose the name of the ROI you are building. You can also select the ID of the region you are currently constructing - ROIs can have multiple region IDs, for example to identify two distinct tumours.



Below these options is the toolbox.



Each tool allows you to modify the ROI region - typically (but now always) on a single slice of the image. Another use of the ROI tool is for quick definition of simple ROIs for testing. For example you could define a small region of a few voxels to test a long-running analysis procedure before running it on the full image. In this case, defining a region on a single slice may be sufficient.

Tools



Crosshairs

This tool is used to revert to the use of mouse clicks to select points/slices of focus rather than select an ROI region. This is helpful in selecting the slice you are working on without accidentally defining a new ROI region.



Pen

This is a typical tool for manual segmentation. Click and drag to draw a boundary around the region you want to select. Clicking Add adds the interior of the region to the ROI. Generally with manual segmentation, you work slice by slice, drawing around the regions as you go. If you are doing this, you may want to maximise one of the viewing windows first.



Walker

This provides simple automatic segmentation using the random walk algorithm. Mouse clicks select points known to be inside (red flags) or outside (white flags) the region of interest - a menu allows you to change between these modes. When some points have been selected, the `Segment` button will generate an ROI which includes the red flags and excludes the white flags.

This process can be carried out on a slice-by-slice basis, or across the whole 3D volume - the `segmentation Mode` menu allows you to choose which.



Rectangle

Simple click-and-drag to select a rectangular region. When you are happy, click Add to add it to the ROI, or click Discard to ignore it.



Ellipse

Identical to the `Rectangle` tool, but selects an elliptical region



Polygon

In this tool, each click on the image adds a vertex of a polygon region. When you click `Add` the last node is connected to the first node to close the polygon, and the interior is selected. Clicks within a different slice window are ignored.



Choose Region

This tool allows you to choose a region of an existing ROI - for example to isolate a particular cluster or supervoxel. Using the menu, select the existing ROI and then click on a point to choose the region it lies within. The region will be displayed in isolation and you can choose to 'Accept' or 'Cancel' the selection.



Eraser

With this tool, click on individual voxels to remove them from the ROI.

Undo

Most changes can be undone by clicking on the `Undo` button. Generally the last 10 additions or removals can be undone.

3.2.11 Mean value widget

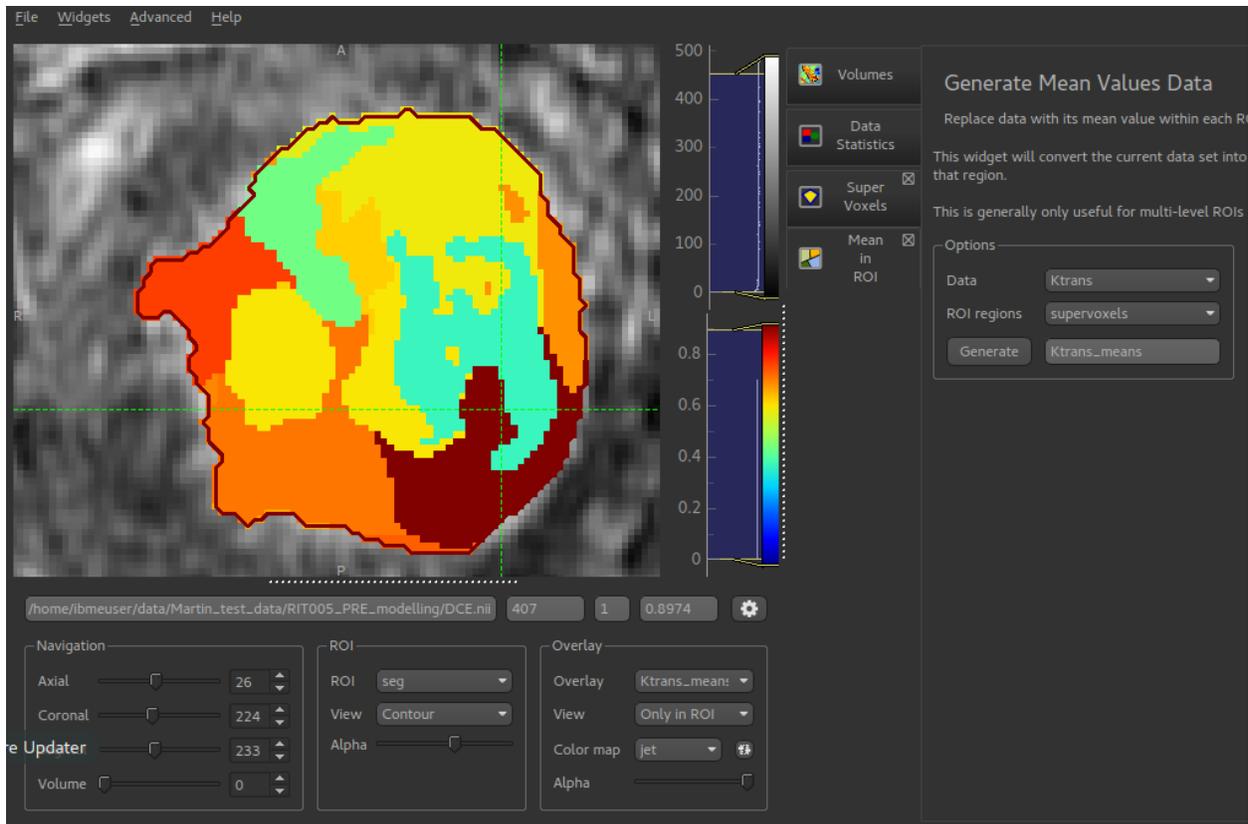
Widgets -> ROIs -> Mean in ROI

The mean values widget takes an overlay and an ROI and outputs a new overlay. Within each ROI region, the new overlay contains the mean value of the original overlay within that region.

This is particularly useful with ROIs generated by clustering or supervoxel methods as it enables the generation of a simplified version of the data where each supervoxel/cluster region has a single value.

The mean values widget also works with 4D data and will output the mean volume series for each ROI region.

Example showing mean K_{trans} value of each supervoxel



3.3 Current included plugins

3.3.1 T1 map from VFA images

Widgets -> T1 -> VFA T1

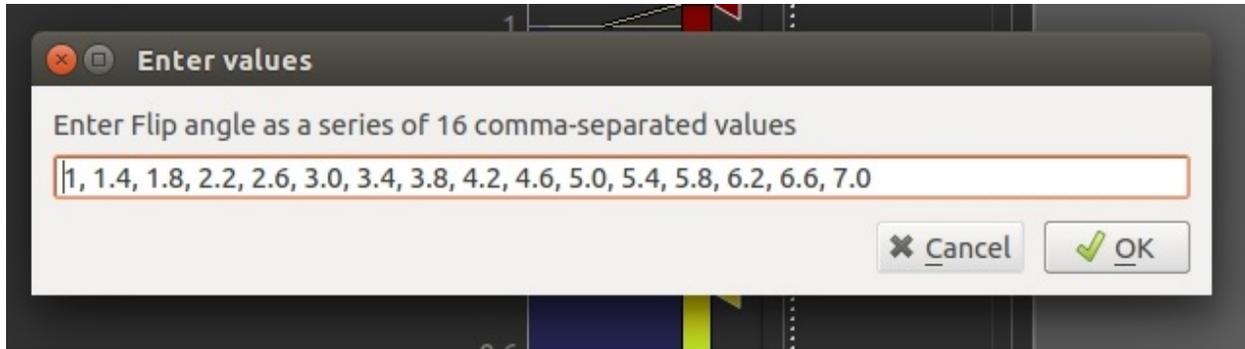
This widget generates T1 maps from variable flip angle images. This is often used as a preprocessing step for kinetic modelling. The VFA scans can be loaded either as separate volumes, one for each flip angle, or a single multi-volume file containing all the flip angles.

The main VFA method is described in¹.

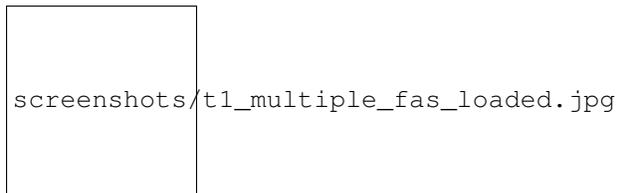
Using a single 4D volume with multiple flip angles

Click Add to and select the data file containing the VFA images. You will then need to enter the flip angles as a comma separated list which must match the number of volumes in the data set.

¹ Fram et al., Magn Reson Imaging 5(3): 201-208 (1987)

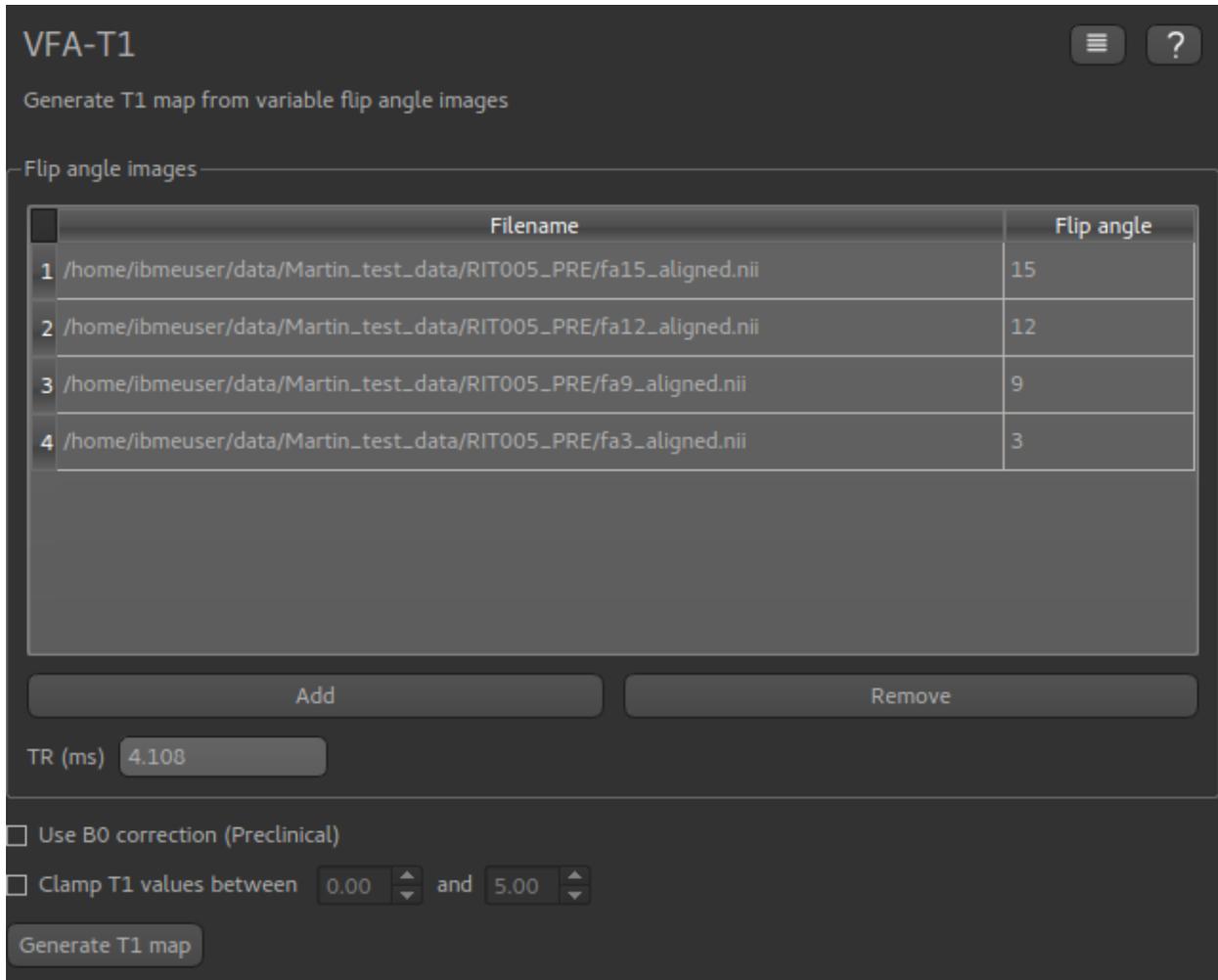


Once loaded, the file will be added to the list:



Loading multiple flip angle volumes

It is also common for the images at different flip angles to be stored in separate files. In this case, simply load each one separately, entering the flip angle for each (the widget will try to guess the flip angle if it is part of the file name, but ensure it has guessed correctly!)

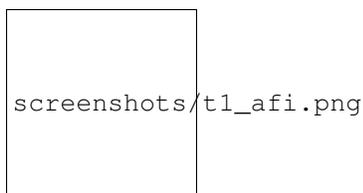


Using a B1 correction

This option allows for correction of field inhomogeneity and B1 effects using Actual Flip Angle Imaging (AFI) data sets. This is particularly common where high field strengths are used, for example in preclinical applications. The method is described in²

These are loaded in the same way as the VFA data described above, however in this case you must enter the TR value for each file in ms (or a sequence of TR values if the AFI data is stored in a single 4D data set).

The flip angle used for the AFI sequence is also required.



² Yarnykh, V. L. (2007), Actual flipangle imaging in the pulsed steady state: A method for rapid three-dimensional mapping of the transmitted radiofrequency field. Magn. Reson. Med., 57: 192-200. doi:10.1002/mrm.21120

Applying Gaussian smoothing to the T1 map

An optional postprocessing step is to apply smoothing to the output T1 map. The `sigma` value is standard deviation of the Gaussian used as the smoothing kernel, and is measured in voxels.

The `Processing->Smoothing` widget can also be used to apply smoothing to the output. This allows the kernel size to be specified in physical units (mm).

Clamping the T1 values

The output T1 values may be clamped between limits if required - this may be useful to eliminate unrealistic values in a small number of voxels.

References

3.3.2 DCE PK modelling

Widgets -> DCE-MRI -> PK Modelling

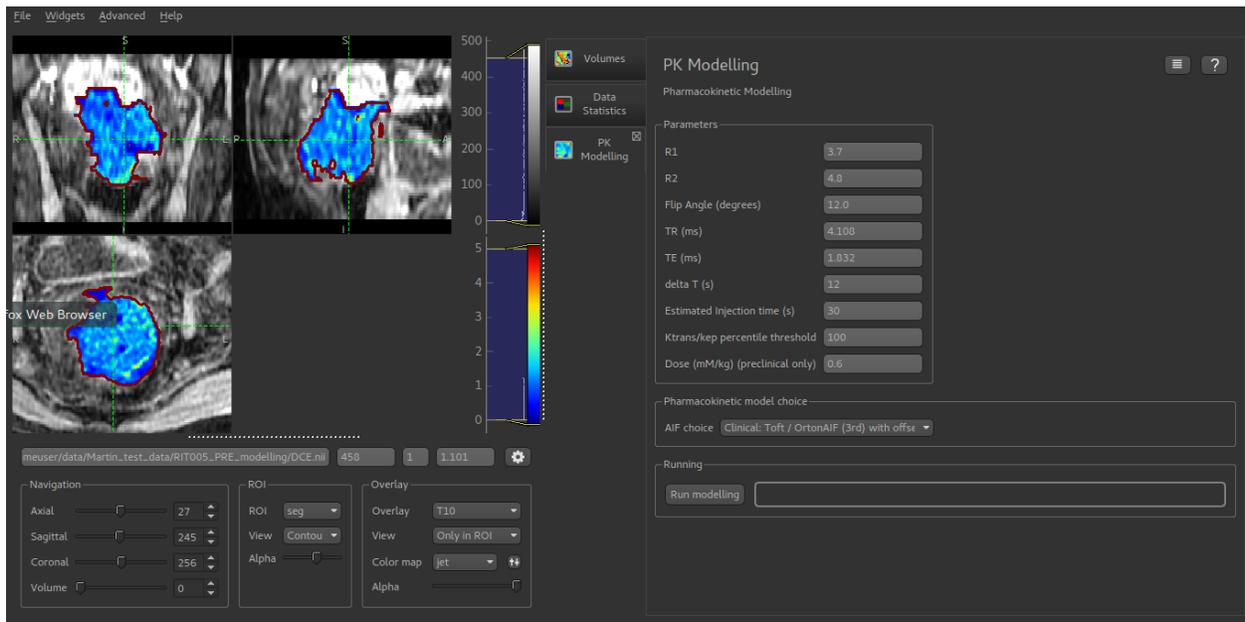
The DCE modelling widget performs pharmacokinetic modelling for Dynamic Contrast-Enhanced MRI (DCE) using the Tofts model.

To use the tool you must first load a DCE-MRI volume, an ROI and a T10 overlay (named 'T10').

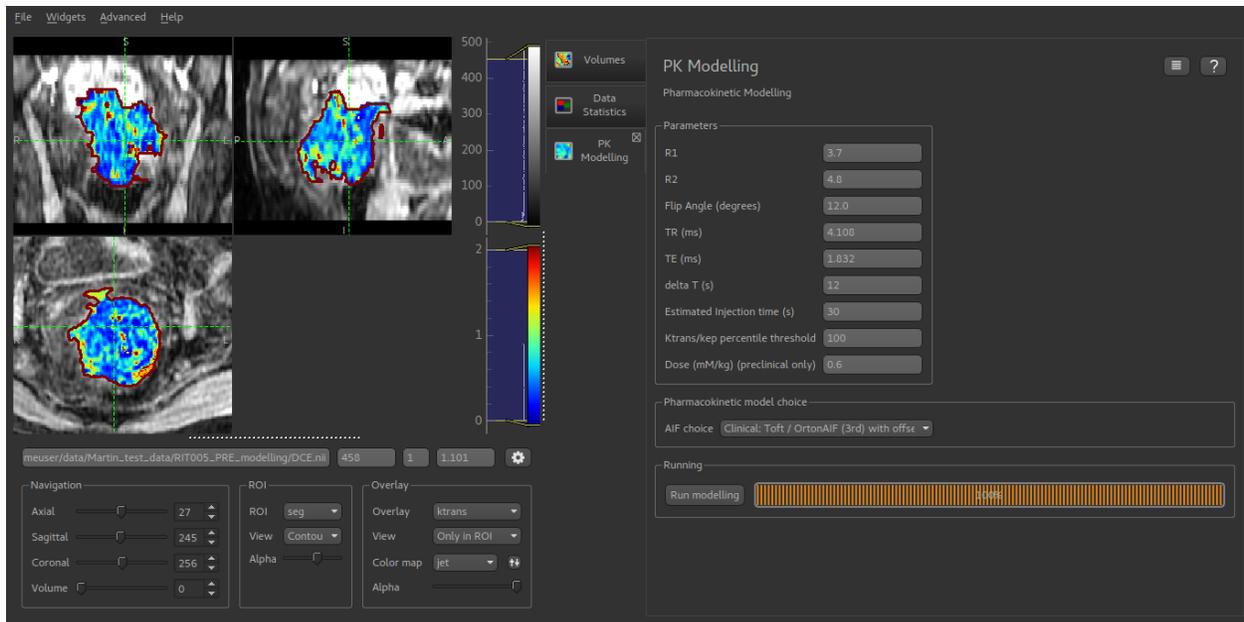
Next you must select the model, which determines the AIF (Arterial Input Function) in use. The Orton AIF is provided for clinical data, and a Weinmann AIF for preclinical applications.

Image acquisition parameters are set using the GUI controls. Once these are correct, click 'Run' to start the modelling process. This could take some time, depending on your data, the progress bar is updated in chunks.

Start of modelling, showing loaded T10 map



Modelling complete with newly generated Ktrans map



Still required:

- Inclusion of Contrast-to-noise ratio restriction

3.3.3 QuantiCEST

- Widgets -> CEST -> QuantiCEST

This widget provides CEST analysis using the Fabber Bayesian model fitting framework.

To do CEST analysis you will need to know the following:

- The frequencies in the z-spectrum you sampled at. The number of frequencies corresponds to the number of volumes in your data
- The field strength - default pool data is provided for 3T and 9.4T, but you can specify a custom value provided you provide the relevant pool data
- The saturation field strength in μT
- For continuous saturation, the duration in seconds
- For pulsed saturation details of the pulse magnitudes and durations for each pulse segment, and the number of pulse repeats
- What pools you want to include in your analysis
- If default data is not available for your pools at your field strength, you will need the ppm offset of each pool, its exchange rate with water and T1/T2 values

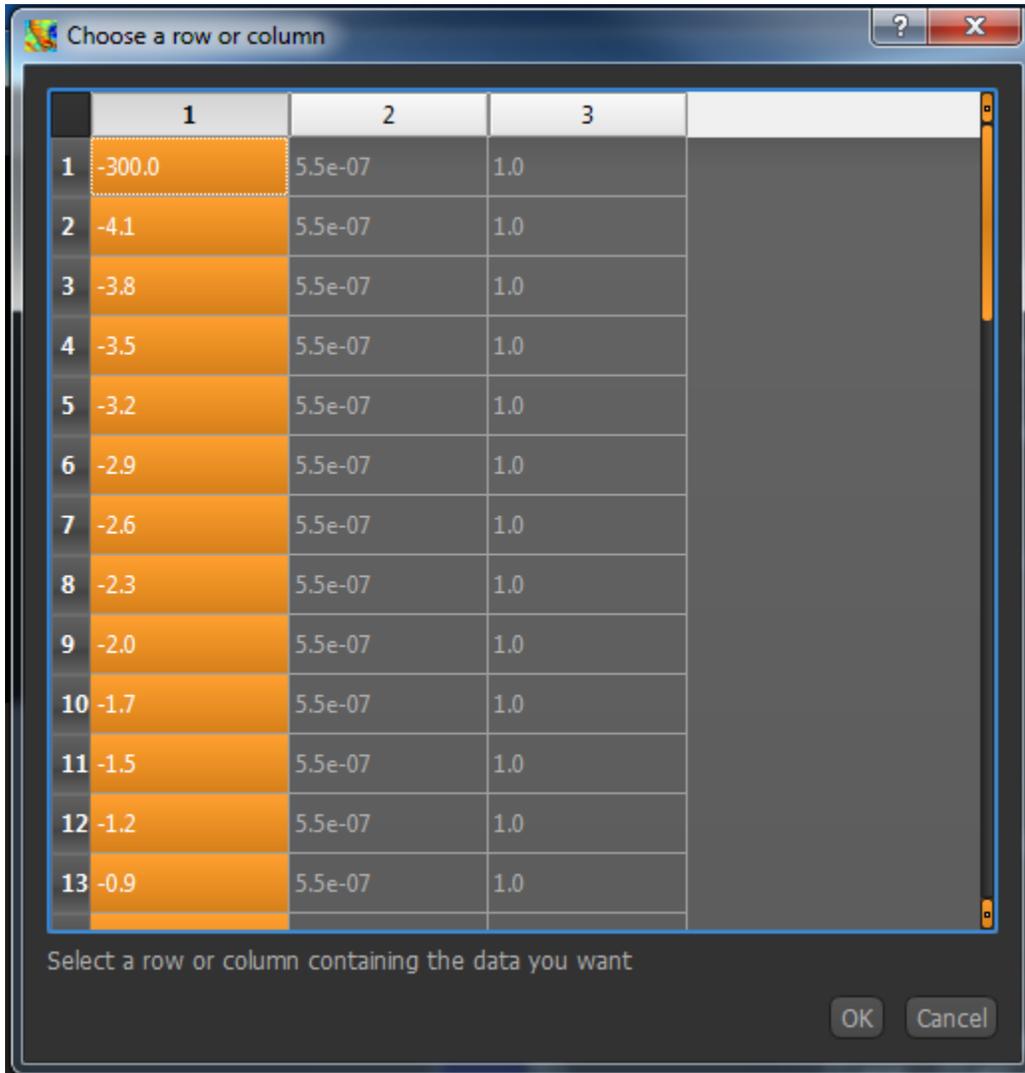
Setting the sampling frequencies

The frequencies are listed horizontally:

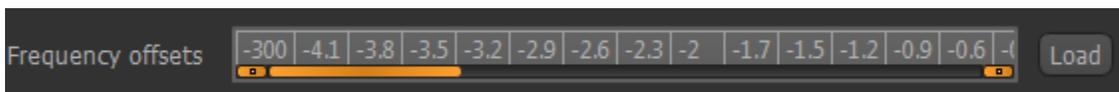


You can type in your frequencies manually - the list will automatically grow as you add numbers.

However, you may have your frequencies listed in an existing text file - for example the `dataspec` file if you have been using Fabber for your analysis. To use this, either drag the file onto the list or click the `Load` button and select the file. Quantiphyse will assume the file contains ASCII numeric data in list or matrix form and will display the data found.



Click on the column or row headers to select the column/row your frequencies are listed in. In this case, we have a Fabber `dataspec` file and the frequencies are in the first column, so I have selected the first column of numbers. Click `OK` to enter this into your frequency list.



Setting the field strengths

Choose the `B0` field strength from the menu. If none of the values are correct, select `Custom` and enter your field strength in the spin box that appears

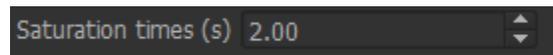


Note that you are being warned that the default pool data will not be correct for a custom field strength and you will need to edit them.

The saturation field strength is set using the `B1` (μT) spin box below.

Continuous saturation

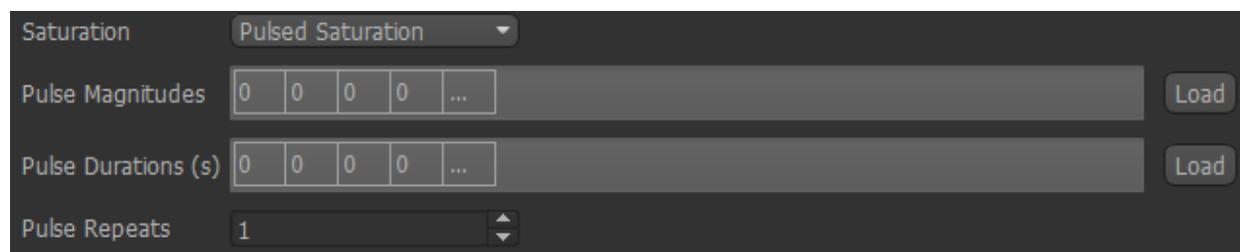
Select `Continuous Saturation` from the menu, and enter the duration in seconds in the spin box



Saturation times (s) 2.00

Pulsed saturation

Select `Pulsed Saturation` from the menu.



Saturation Pulsed Saturation

Pulse Magnitudes 0 0 0 0 ... Load

Pulse Durations (s) 0 0 0 0 ... Load

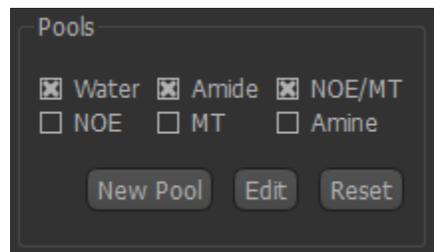
Pulse Repeats 1

The pulse magnitudes and durations can be set in the same way as the sampling frequencies, so if you have them in a text file (for example a Fabber `ptrain` file), drag it onto the list and choose the appropriate row/column.

The number of magnitudes must match the number of durations! Repeats can be set in the spin box at the bottom.

Choosing pools

Six built-in pools are provided, with data at 3T and 9.4T, you can choose which to include using the checkboxes.



Pools

Water Amide NOE/MT

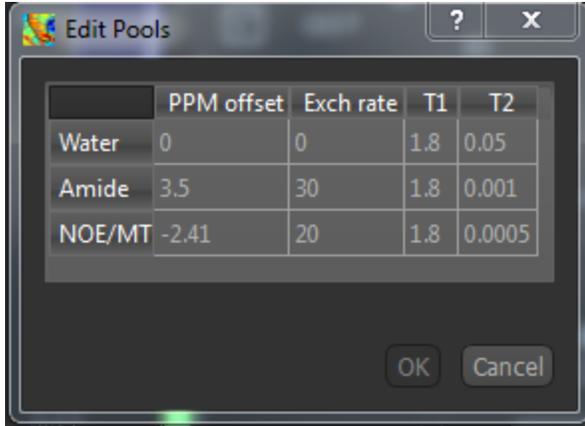
NOE MT Amine

New Pool Edit Reset

Each pool is characterized by four parameters:

- The ppm offset relative to water (by definition this is zero for water)
- The exchange rate with water
- The T1 value at the specified field strength
- The T2 value at the specified field strength

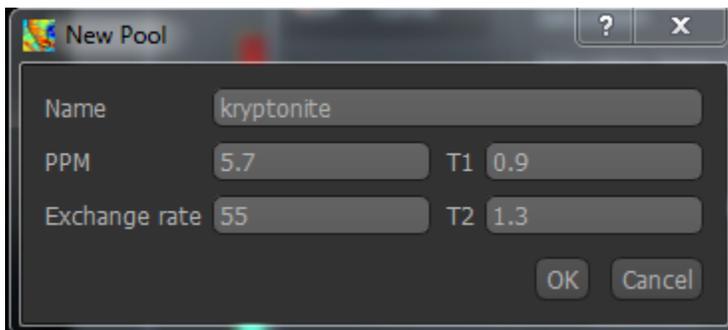
To view or change these values, click the `Edit` button.



A warning will appear if you change the values from the defaults. Obviously this will be necessary if you are using a custom field strength. If you want to return to the original values at any point, click the `Reset` button. This does not affect what pools you have selected and will not remove custom pools

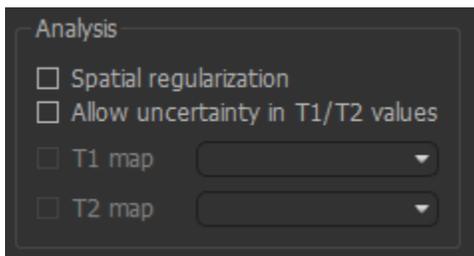
Custom pools

If you want to use a pool which is not built-in, you can use the *New Pool* button to add it. You will need to provide the four parameters above, and your new pool will then be selected by default.



Warning: Currently custom pools, and custom pool values are not saved when you exit Quantiphyse

Analysis options



These affect how Fabber performs the model fitting

- `Spatial regularization` - if enabled, adaptive smoothing will be performed on the parameter maps, with the degree of smoothing determined by the variation of the data

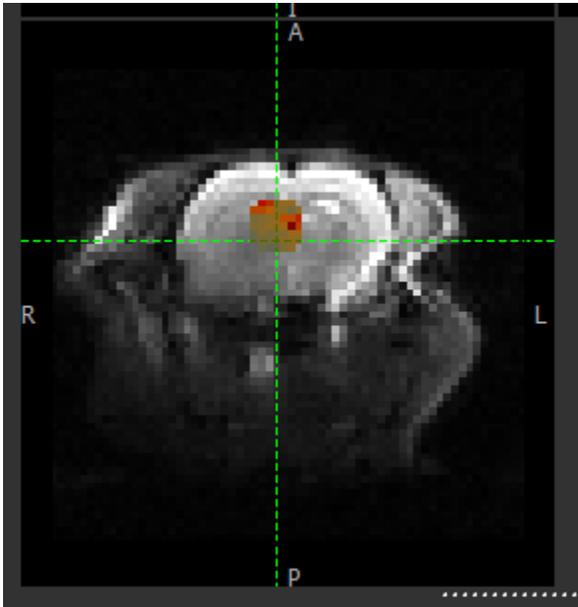
- Allow uncertainty in T1/T2 values - T1/T2 will be inferred, using the pool-specified values as initial priors
- T1 map/T2 map - If inferring T1/T2, an alternative to using the pool-specified values as priors you may provide existing T1/T2 maps for the water pool.

Warning: Spatial regularization prevents Fabber from processing voxels in parallel, hence the analysis will be much slower on multi-core systems.

Run model-based analysis

This will perform the model fitting process.

CEST analysis is computationally expensive, and it is recommended to run on a small ROI before attempting your full data set. The ROI Builder tool is an easy way to define a small group of voxels to act as a test ROI, e.g. as below



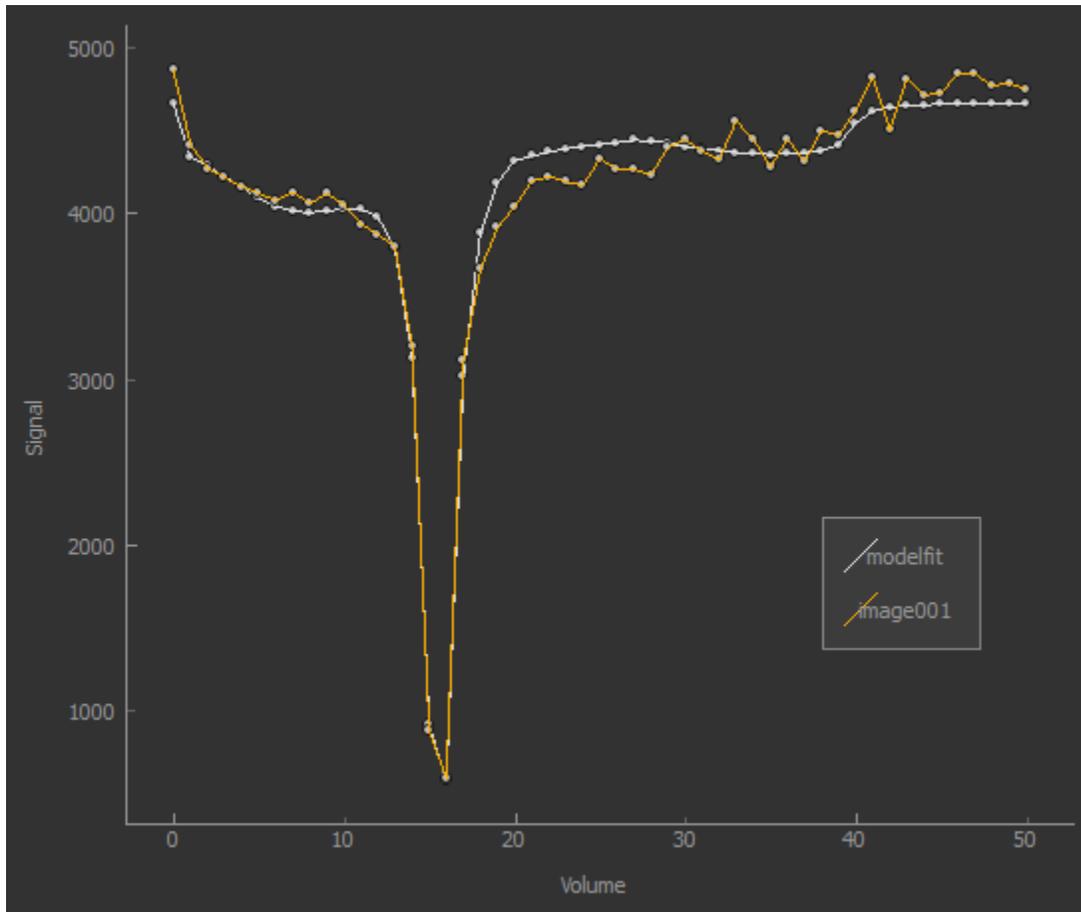
The output of the model-based analysis is a set of data overlays as follows:

- mean_B1_off - Model-inferred correction to the specified B1 value
- mean_ppm_off - Model-inferred correction to the ppm values in the z-spectrum.
- modelfit - Model z-spectrum prediction, for comparison with raw data
- mean_M0_Water - Inferred magnetization of the water pool
- mean_M0_Amine_r, mean_M0_NOE_r, ..etc - Inferred magnetization of the other pools, relative to M0_Water
- mean_exch_Amine, mean_exch_NOE, ..etc - Inferred exchange rates of non-water pools with water
- mean_ppm_Amine, mean_ppm_NOE, ..etc - Inferred ppm frequencies of non-water pools
- cest_rstar_Amine, cest_rstar_NOE, ..etc - Calculation of R* for non-water pools - see below for method

If T1/T2 values are being inferred (Allow uncertainty in T1/T2 values is checked), there will be additional outputs:

- mean_T1_Water, mean_T1_Amine, ..etc - Inferred T1 values for each pool
- mean_T2_Water, mean_T2_Amine, ..etc - Inferred T2 values for each pool

The screenshot below (from the Voxel Analysis widget) shows the model fitting to the z-spectrum.



CEST R* calculation

The R* calculation is performed as follows:

- After the model fitting process, for each non-water pool, two separate z-spectrum predictions are evaluated at each voxel: - The spectrum based on the water pool only - The spectrum based on the water pool and each other pool individually
- The parameters used for this evaluation are those that resulted from the fitting process, except that: - T1 and T2 are given their prior values - The water ppm offset is zero
- Each spectrum is evaluated at the pool ppm resonance value and the normalized difference to water is returned:

$$R_{pool}^* = \frac{Signal_{water} - Signal_{water+pool}}{M_0}$$

Lorentzian difference analysis

This is a quicker alternative to model-based analysis, however less information is returned.

The calculation is performed using the Fabber fitting tool as previously, in the following way:

- Only the water pool is included, i.e. just fitting a single Lorentzian function to the z-spectrum
- Only data points close to the water peak and unsaturated points are included. Currently this means points with ppm between -1 and 1 are included as are points with ppm > 30 or < -30
- The raw data is subtracted from the resulting model prediction at all sampled z-spectrum points

The output of the LDA calculation is provided as a multi-volume overlay `lorenz_diff`.

3.3.4 ASL Analysis

- *Widgets -> ASL*

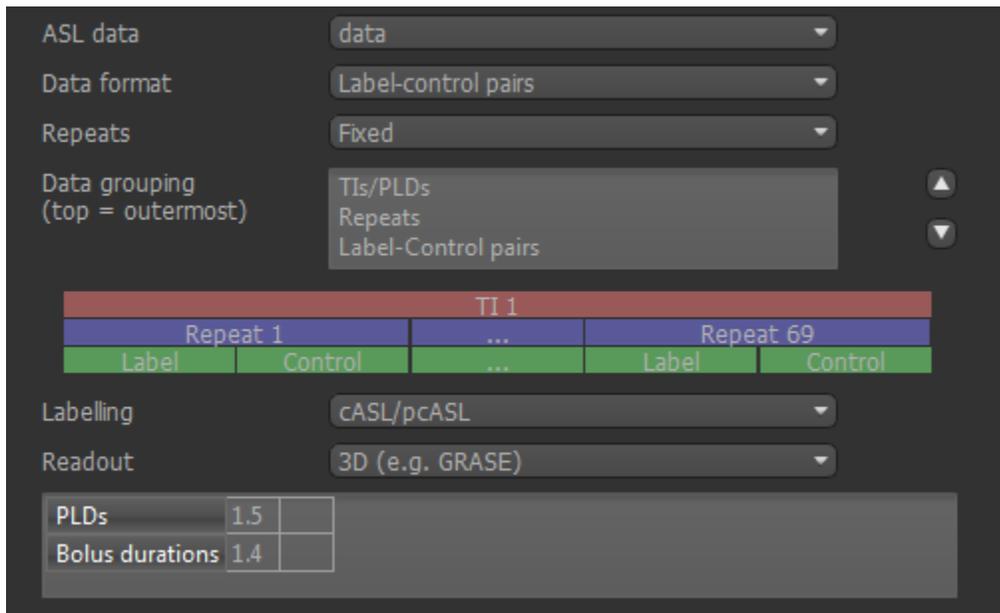
A number of widgets are provided for use in processing Arterial Spin Labelling MRI analysis using the Fabber Bayesian model fitting framework.

ASL Data Structure

All the ASL widgets share a general purpose structure definition widget which allows you to describe the structure and acquisition parameters of your ASL data, for example:

- Whether your data is tag-control pairs, already subtracted or multi-phase
- The ordering of the volume sequence in your data (e.g. whether tag-control pairs occur together, or whether all the tags come first)
- Number of TIs/PLDs used and their values, bolus durations, number of repeats etc
- Additional acquisition details, such as 2d/3d readout

In this section, we describe how to describe the structure of ASL data using this widget.



ASL data

This selects the data set whose structure you are defining. Once you define the structure of a data set in one ASL widget, others will automatically pick up the same structure when using that data set. In addition, if you save the data set to a Nifti file, the structure information is saved as optional metadata and will be recognised when you load the data back into Quantiphyse.

Labelling format

The labelling can be described as Label-control pairs, Control-Label pairs, already tag-control subtracted or multiphase. Note that not all labelling methods are supported by all ASL widgets, for example to work with multiphase data you will need to preprocess it using the [Multiphase ASL widget](#).

Data grouping/order

This describes the sequence of volumes contained in the ASL data set, and what each volume contains. For example the sequence of volumes in your data might be as follows:

```

1. Tag for first TI
2. Control for first TI
3. Tag for second TI
4. Control for second TI
   ... as above for remaining TIs
11. Repeat of Tag for first TI
12. Repeat of Control for first TI
    ... etc

```

The data grouping box enables you to choose how the Tag-Control pairs, the repeats and the TIs are ordered in your data. You can use the Up/Down buttons to move the grouping order. To make this easier, a visualisation is shown below. For example the ordering described above would be described as TC pairs (innermost), TIs, Repeats (outermost), and would be shown as follows:

Repeat 1					Repeat 2					...	
TI 1		TI 2		...	TI 1		TI 2		
Tag	Control	Tag	Control	...	Tag	Control	Tag	Control	

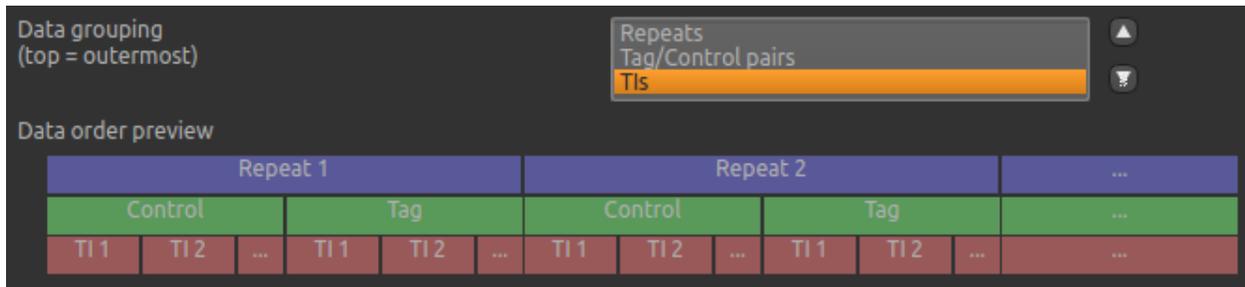
Alternatively the same data might be in a different order:

```

1. Control for first TI
2. Control for second TI
   ... as above for remaining TIs
6. Tag for first TI
7. Tag for second TI
   ... as above for remaining TIs
11. Repeat of Control for first TI
12. Repeat of Control for second TI
    ... etc

```

This would be described as TIs (innermost), CT pairs, Repeats (outermost), and would be shown as follows:



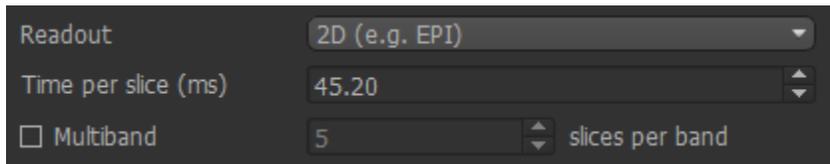
For examples of the signal curves expected for different types of ordering, see the [ASL preprocessing widget](#)

Labelling method

The labelling method is either cASL/pcASL or pASL. In cASL/pcASL, the effective TI for each volume is determined by adding the post-labelling delay (PLD) to the bolus duration. In pASL, the TIs are specified directly.

Readout options

If the readout was 2D, the delay time per slice can be specified to enable the timing of each slice to be determined more accurately. It is also possible to specify a multiband readout.

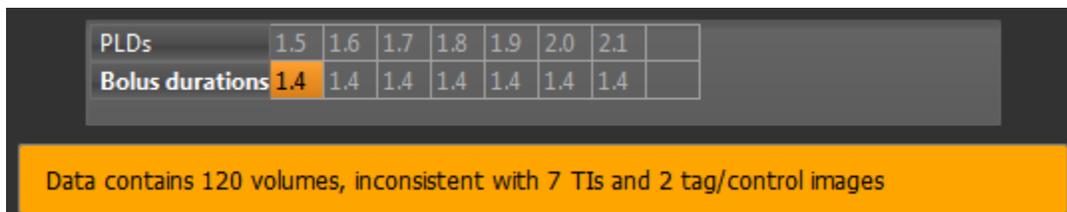


TI/PLD options

The TIs or PLDs recorded in the ASL data must be specified, with the corresponding bolus durations. Initially data is interpreted as single-TI, however additional TIs can be added by typing their values into the blank space at the end of the TIs list. The list will automatically enlarge to fit the number of TIs and provide a new blank space to enter any further values.



If the number of PLDs specified is not consistent with the number of data volumes, a warning is displayed:



Here we have specified a label-control dataset with 7 PLDs - this means the number of volumes should be a multiple of 14.

ASL Preprocessing

- *Widgets -> ASL -> Preprocess*

This widget provides simple preprocessing for ASL data.

ASL data structure

The structure of the ASL data is defined using the ASL structure widget

The screenshot shows the 'ASL structure widget' interface with two tabs: 'Data Structure' and 'Analysis Options'. The 'Data Structure' tab is active. It contains several configuration options:

- ASL data:** mpld_asltc
- Data format:** Label-control pairs
- Repeats:** Fixed
- Data grouping (top = outermost):** TIs/PLDs, Repeats, Label-Control pairs
- Labelling:** cASL/pcASL
- Readout:** 2D (e.g. EPI)
- Time per slice (ms):** 45.20
- Multiband:** (checked)
- Slices per band:** 5

Below these options is a table defining the data structure:

	TI 1		...	TI 6	
Repeat 1	...	Repeat 8	...	Repeat 1	Repeat 8
L	C	...	L	C	...

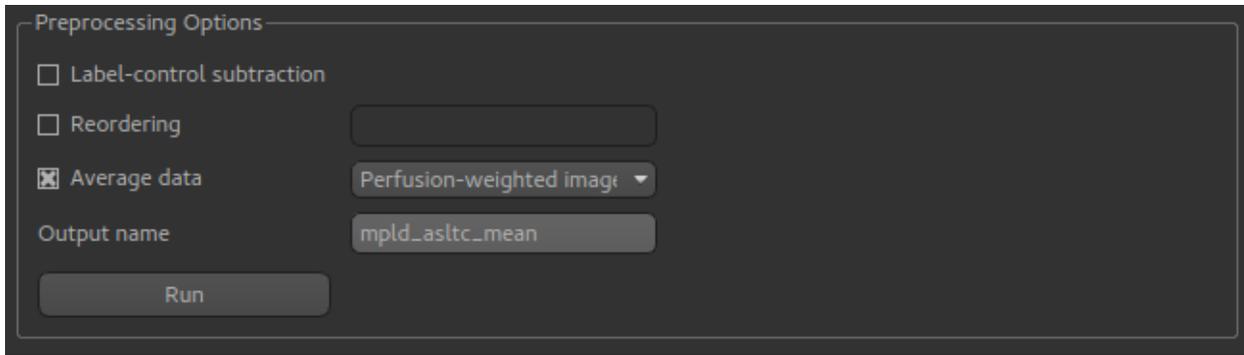
At the bottom, there is a table for PLDs and Bolus durations:

PLDs	0.25	0.5	0.75	1	1.25	1.5	
Bolus durations	1.4	1.4	1.4	1.4	1.4	1.4	

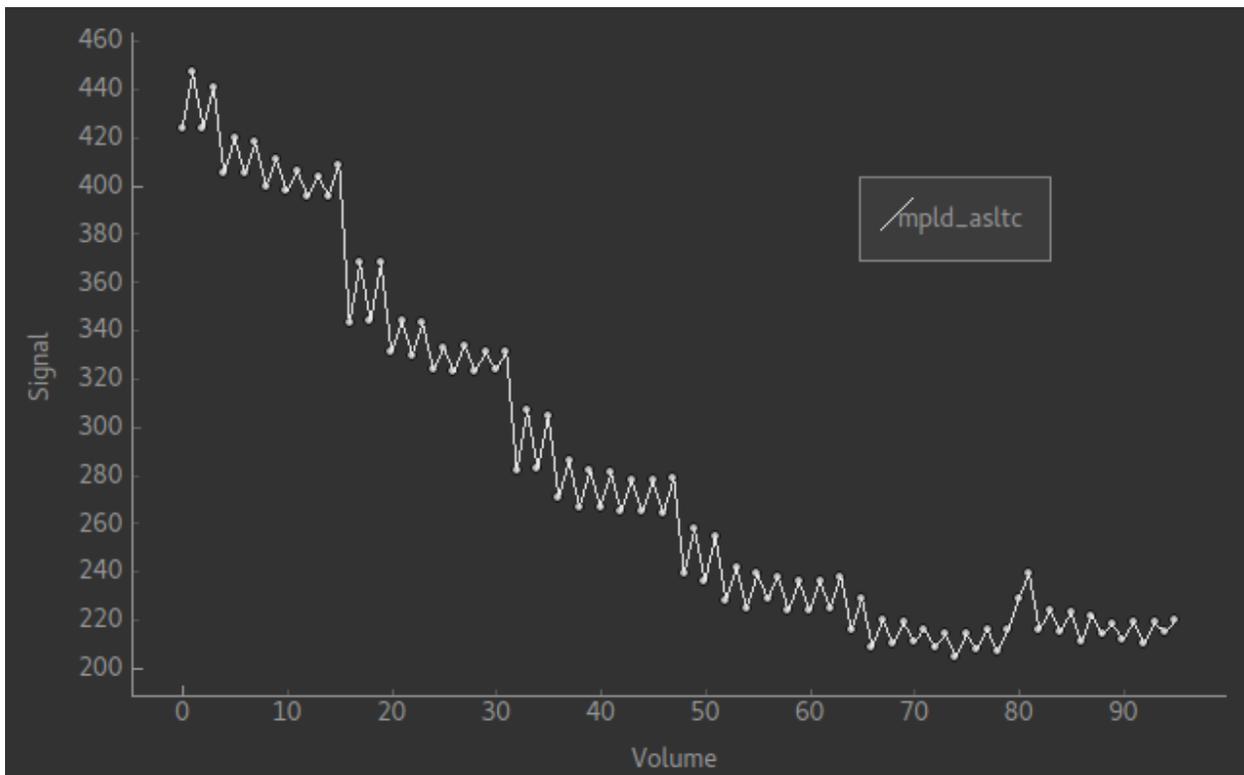
This example shows a multi-PLD PCASL data set with 2D readout.

Preprocessing options

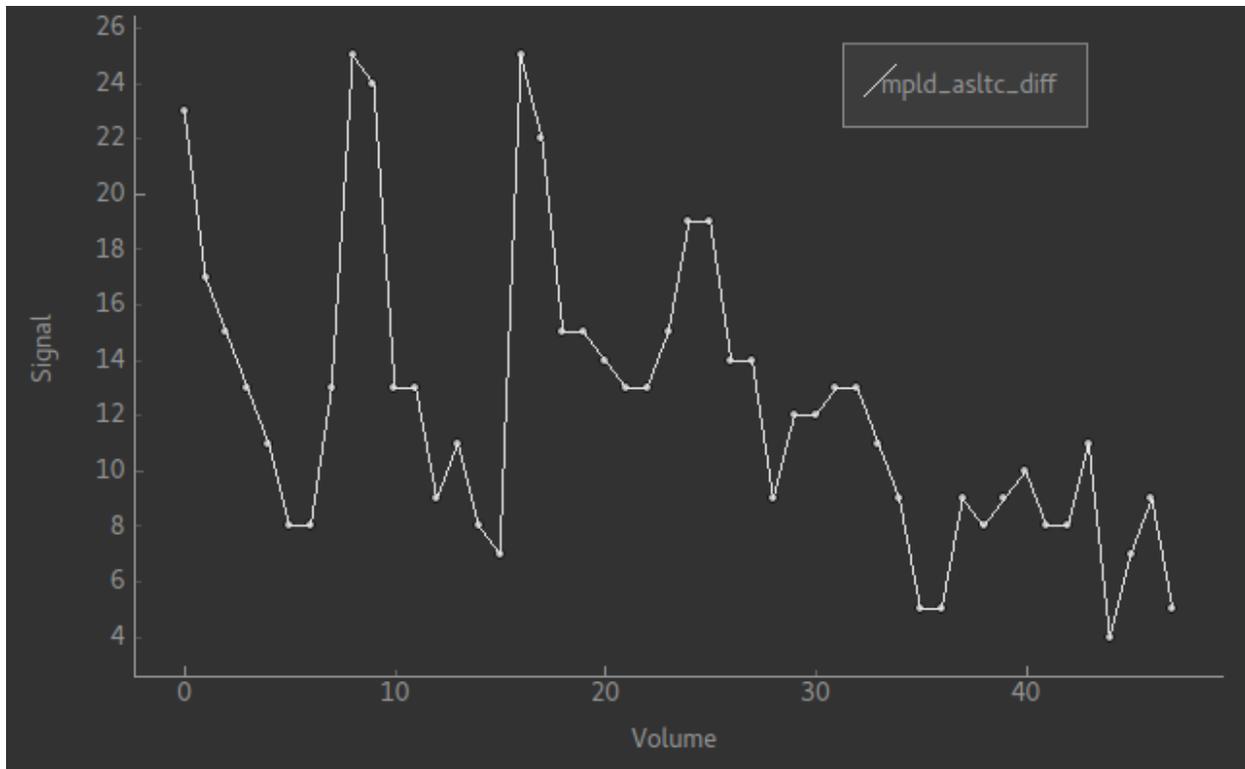
Pre-processing options are shown below the structure definition:



Label-control subtraction will convert the data into differenced ASL data, passing on other details of the ASL structure (PLDs, etc) to the output data set. If we view the raw signal using the `Voxel Analysis` widget we see a pattern like this:



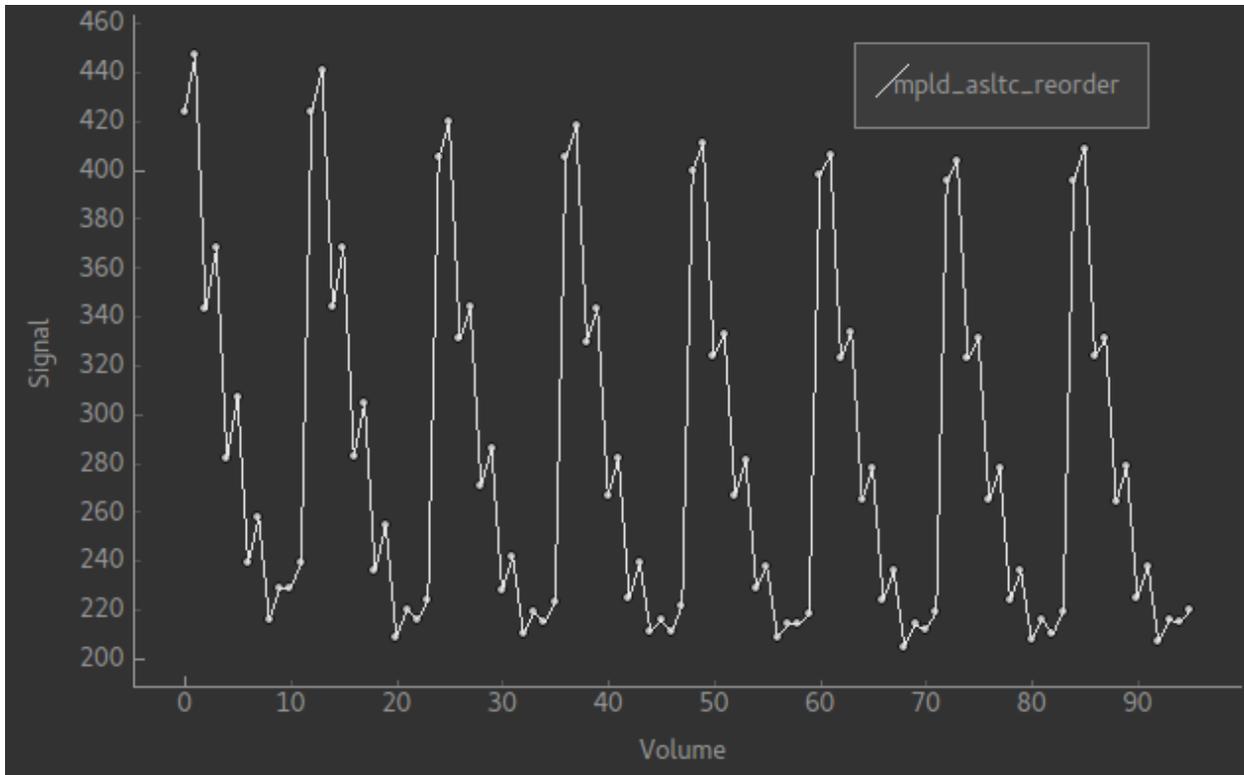
The alternating pattern of the tag-control pairs is clearly visible. After differencing we see the following:



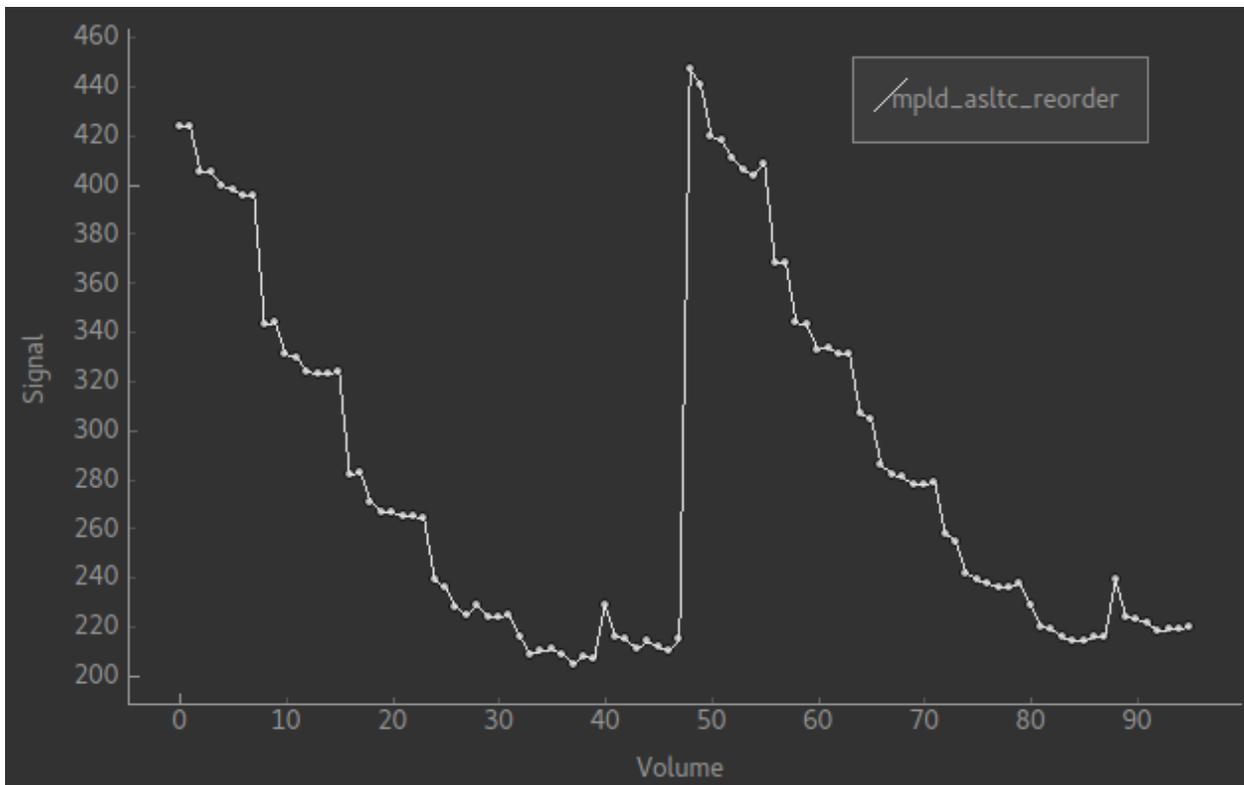
This is harder to read as the data is quite noisy, however note that there are half the number of volumes and the alternating pattern is gone.

Reordering changes the grouping order of the ASL data. The re-ordering string consists of a sequence of characters with the innermost grouping first. `p` represents Label-Control pairs, (capital `P` is used for Control-Label pairs), `r` is for repeats and `t` is for TIs/PLDs. For example, the data above is in order `p r t` and in the signal pattern above you can see a series of repeat measurements at six PLDs.

Changing the order to `p t r` will keep the tag/pair alternation but change the data to repeats of sets of all six PLDs:

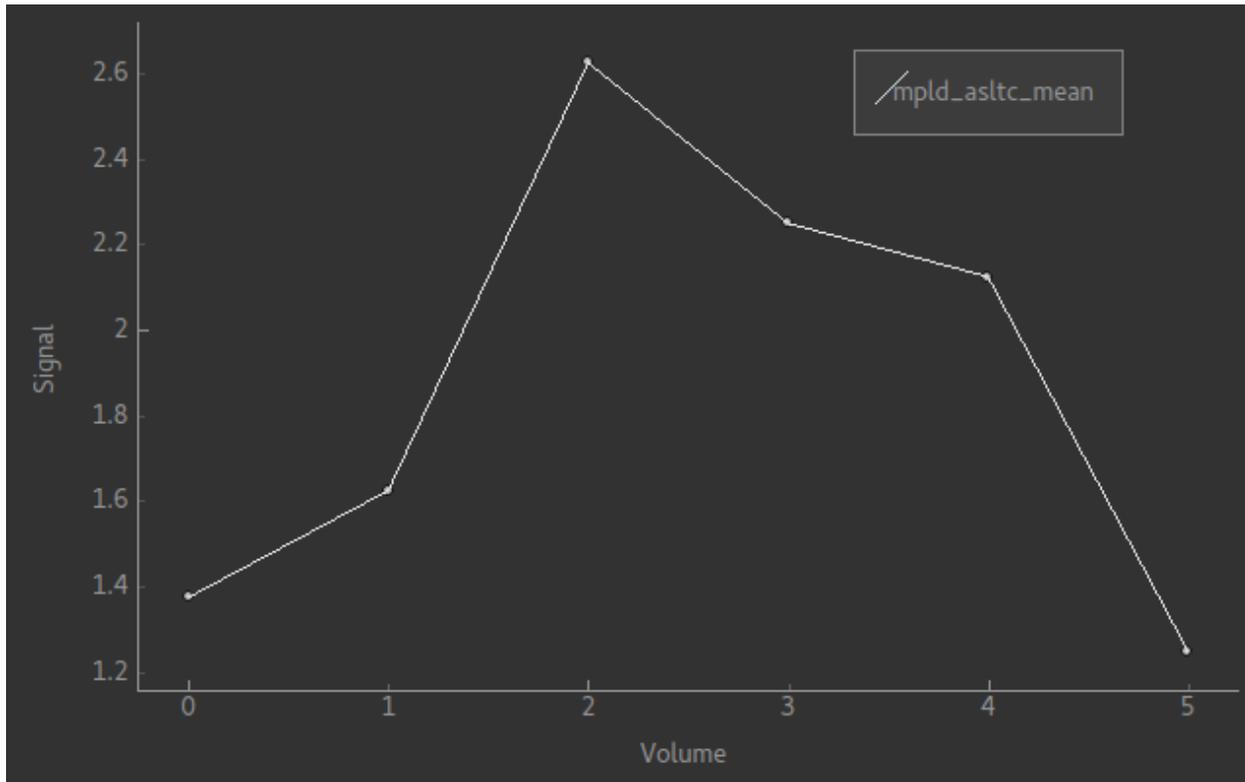


If you prefer, you could have the original ordering but all the tags first and all the control images afterwards. That would be an ordering of `rtb` and looks like this:



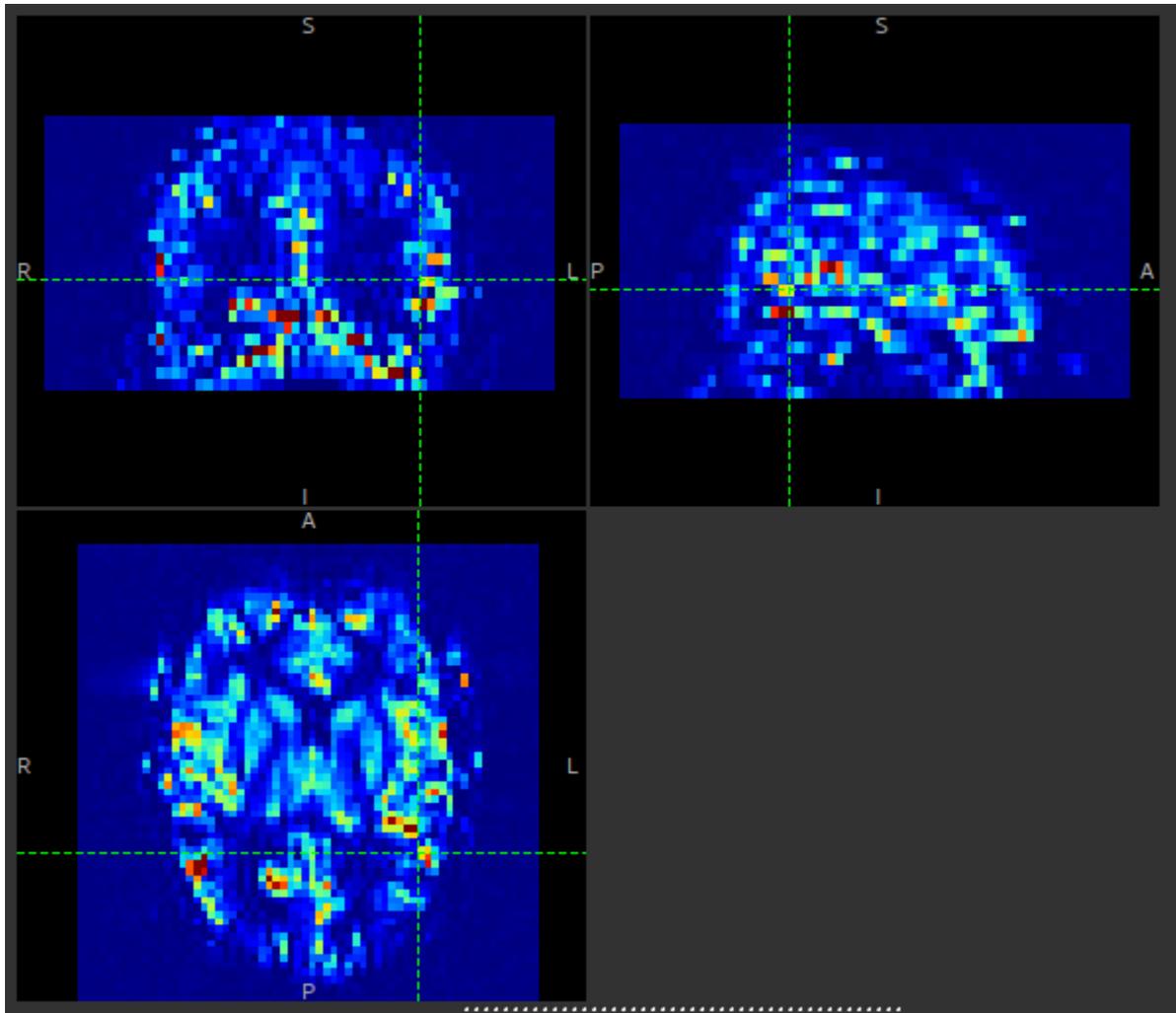
Average data can do one of two things. It can take the mean over the repeats at each PLD, resulting in a new

multi-PLD data set. This enables the ASL signal to be seen more clearly in multi-repeat data, for example this is a typical signal from the data shown above:



This shows the ASL signal more clearly than the noisy differenced data signal shown above.

Alternatively, for a multi-PLD data set you can take the mean over all PLDs as well to generate a Perfusion-weighted image. This is usually similar to the output from model fitting and provides a quick way to check the perfusion. The result of this operation is always a single-volume image. For the data above it looks as follows:



This can be compared to the output shown at the bottom of the [ASL model fitting](#) page.

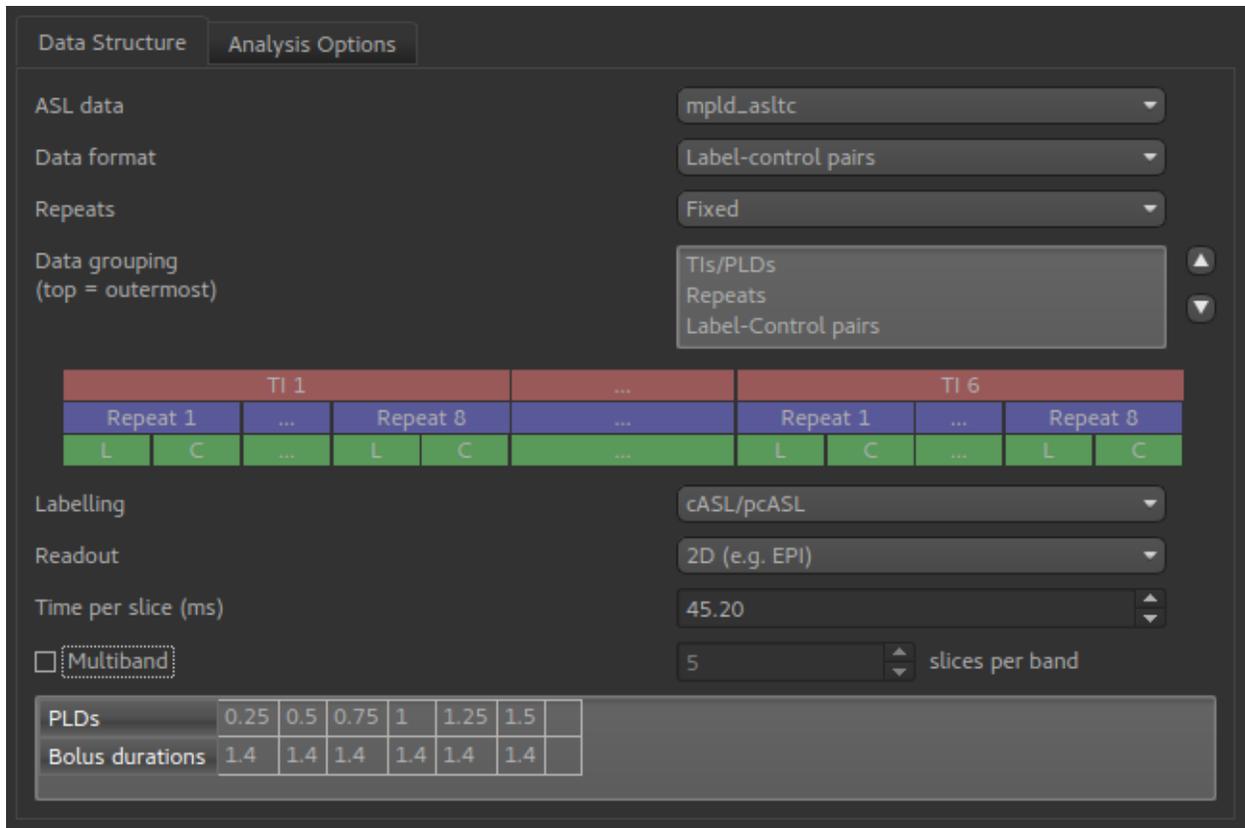
ASL Analysis

- *Widgets -> ASL -> Model fitting*

This widget provides Arterial Spin Labelling MRI analysis using the Fabber Bayesian model fitting framework.

ASL data structure

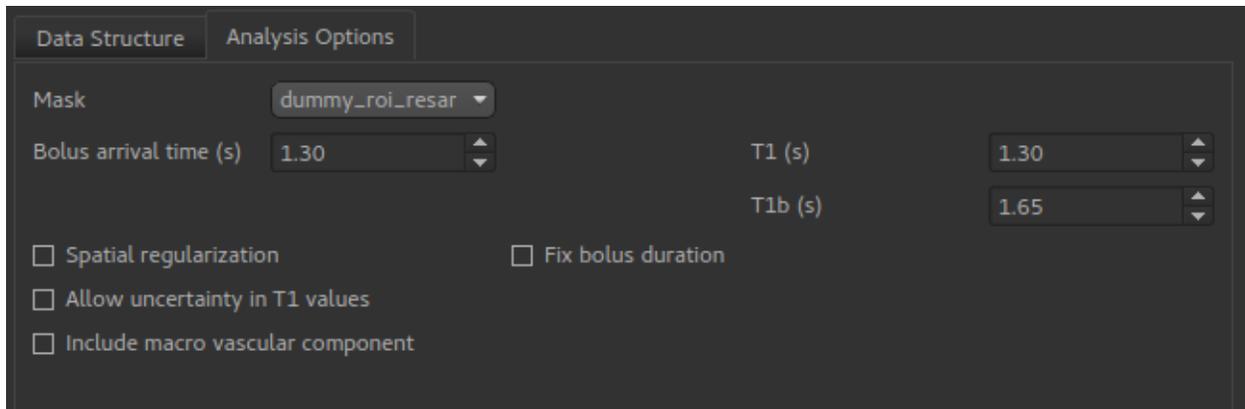
To do ASL model fitting you will need to define the structure of your ASL data, using the [ASL structure](#) widget



This example shows a multi-PLD PCASL data set with 2D readout.

Model fitting options

In addition, a number of options may be specified for the model fitting process on the `Analysis` tab.



- `Bolus arrival time` is an estimate - with multi-PLD data it's actual value will be estimated based on the data.
- `T1` and `T1b` are estimated T1 values for tissue and blood. By default these are fixed, however the `Allow uncertainty in T1 values` option will cause them to be estimated from the data, using the specified values as Bayesian priors.

- `Spatial regularization` will smooth the data using an adaptive method which depends on the degree of variation in the data.
- `Include macro vascular component` will allow an additional arterial component in the fitting process.
- `Fix bolus duration` Normally the bolus duration is allowed some variation to better fit the data. Selecting this option will fix it to the user specified value.

Warning: `Spatial regularization` prevents Fabber from processing voxels in parallel, hence this part of the analysis will be much slower than the other steps on multi-core systems.

Click `Run` to start the analysis. Typically the analysis occurs in multiple stages which are displayed in the `Run ASL modelling` box. Note that the final spatial step does not currently give progress updates, so it may appear to have hung while this is running.

Output data

The following data items are returned:

- `perfusion` Tissue perfusion
- `duration` Bolus duration, unless this is fixed
- `arrival` Inferred bolus arrival time
- `modelfit` Model prediction for comparison with the tag-control differenced data

If `Include macro vascular component` is specified:

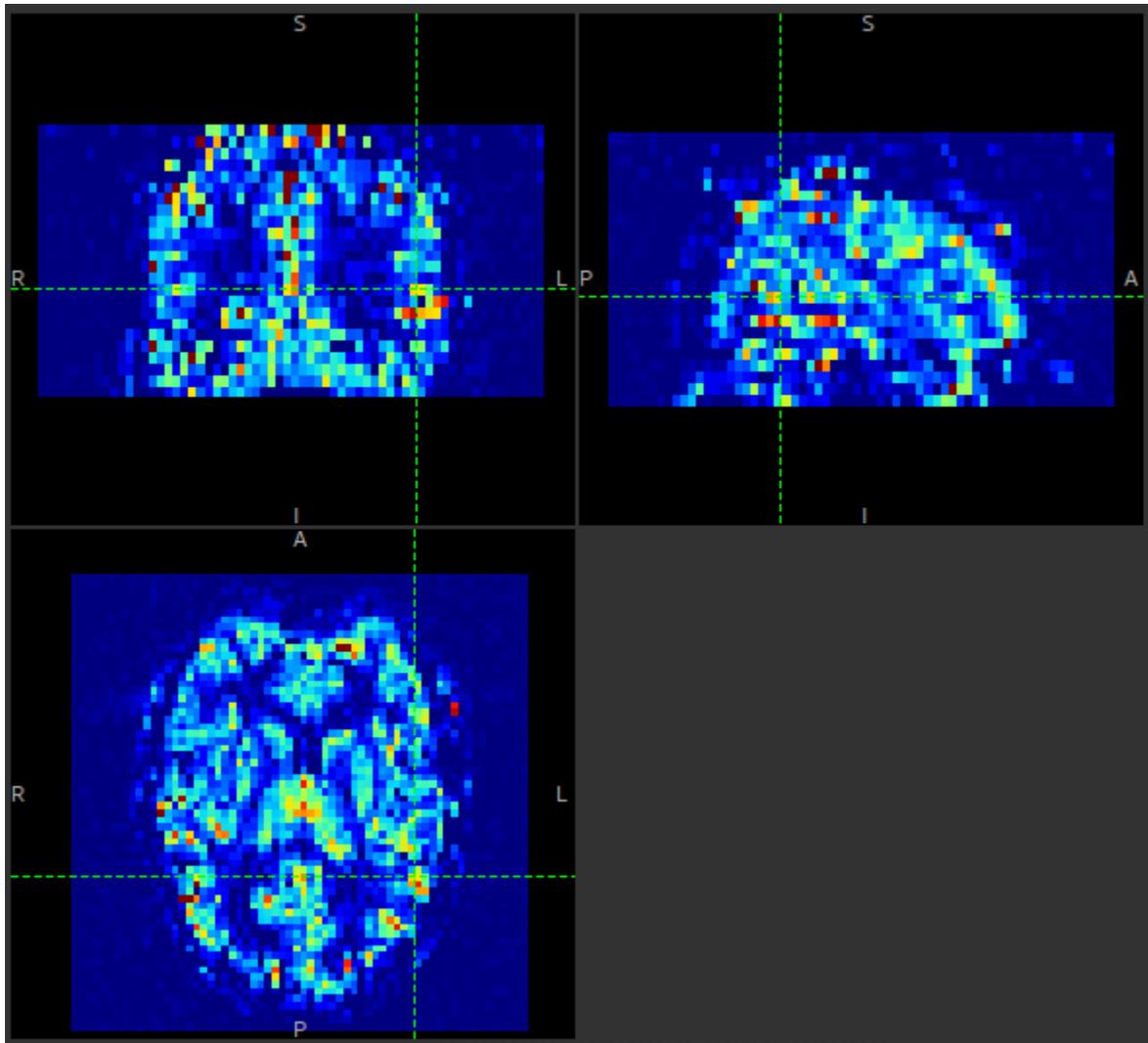
- `aCBV`

If `Allow uncertainty in T1 values` is specified:

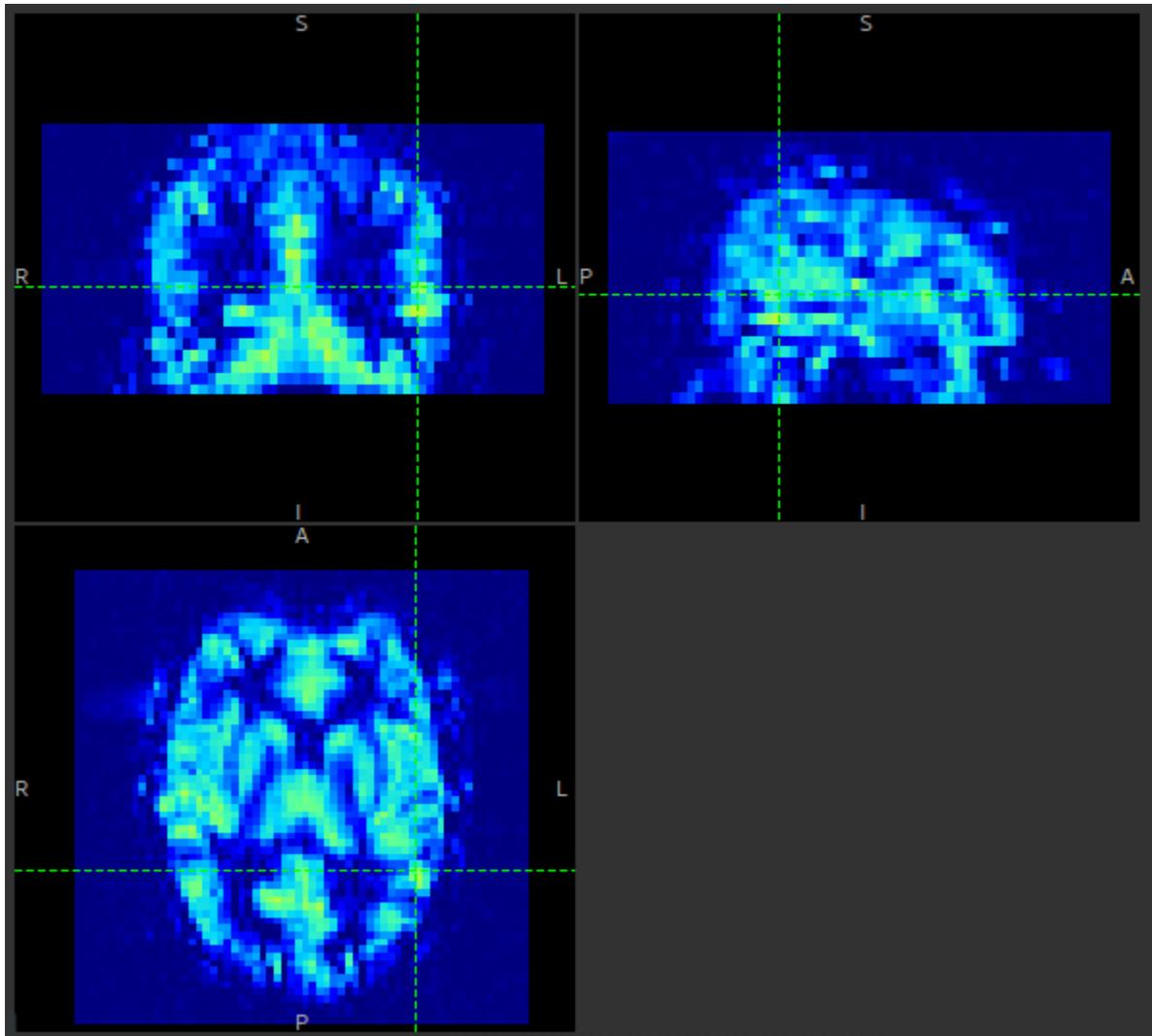
- `mean_T_1` Tissue T1 value
- `mean_T_1b` Blood T1 value

For each of these data items, a standard deviation map will also be produced with the suffix `_std`. This reflects the output of the Bayesian inference - the value of a parameter at a given voxel is represented as a Gaussian probability distribution with a mean value (the parameter map) and a standard deviation (`_std` map).

An example perfusion map without spatial regularization might look like this:



With spatial regularization turned on, the same data set produced the following perfusion map:



The effect is similar to what you would get by applying a smoothing algorithm to the output, however in this case the degree of smoothing is determined by the the variation in the data itself.

ASL Calibration

- *Widgets -> ASL Calibration*

The ASL Calibration widget is required to turn the perfusion images from ASL model fitting into physical units (ml/100g/min).

Basic data specification

Data to calibrate	
Data	perfusion
Data type	Perfusion
Data ROI	dummy_roi_resampled
Calibration method	Voxelwise

To do calibration you specify a data set containing a perfusion image. If you want to calibrate an image containing perfusion variance, the `Data type` selection must be changed (because the scaling factors must be squared in this case).

Calibration methods

Two calibration methods are provided:

- `Voxelwise` calibration, in which an `M0` correction factor is determined for each voxel from the calibration image.
- `Reference region` calibration, in which a single `M0` correction factor is determined for the whole image, by analysing a region of the data containing a single tissue type (typically CSF).

Calibration data

Certain parameters are required regardless of which calibration method you use:

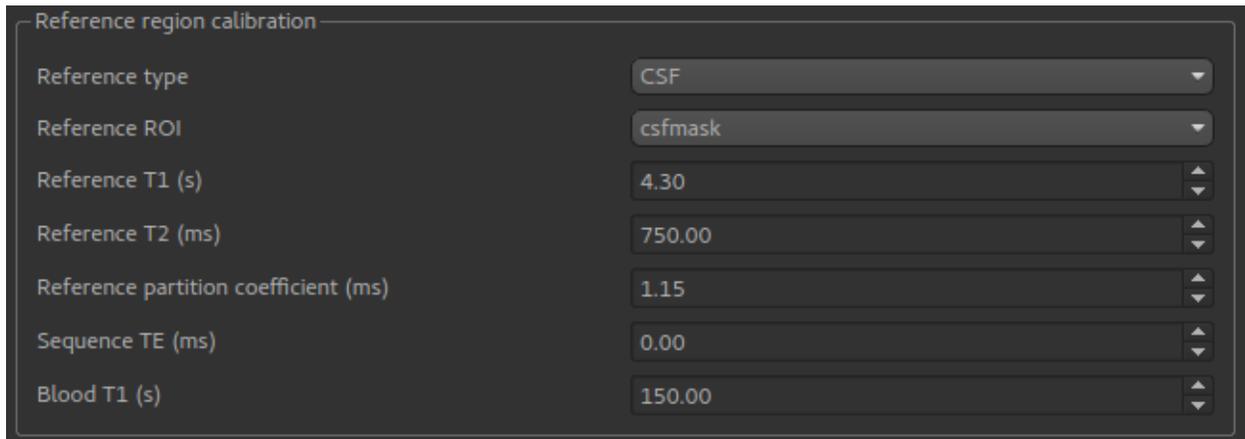
Calibration Data	
Calibration image	aslcalib
Sequence TR (s)	3.20
Calibration gain	1.00
Inversion efficiency	0.98

Voxelwise calibration

Voxelwise calibration	
Tissue T1	1.30
Tissue partition coefficient	0.90

Voxelwise calibration requires a generic estimate of the T1 and the partition coefficient which will be applied to all voxels.

Reference region calibration



The screenshot shows a dark-themed control panel titled "Reference region calibration". It contains seven rows of controls, each with a label on the left and a value on the right. The first two rows are dropdown menus, and the remaining five are numeric input fields with up/down arrow buttons on the right side.

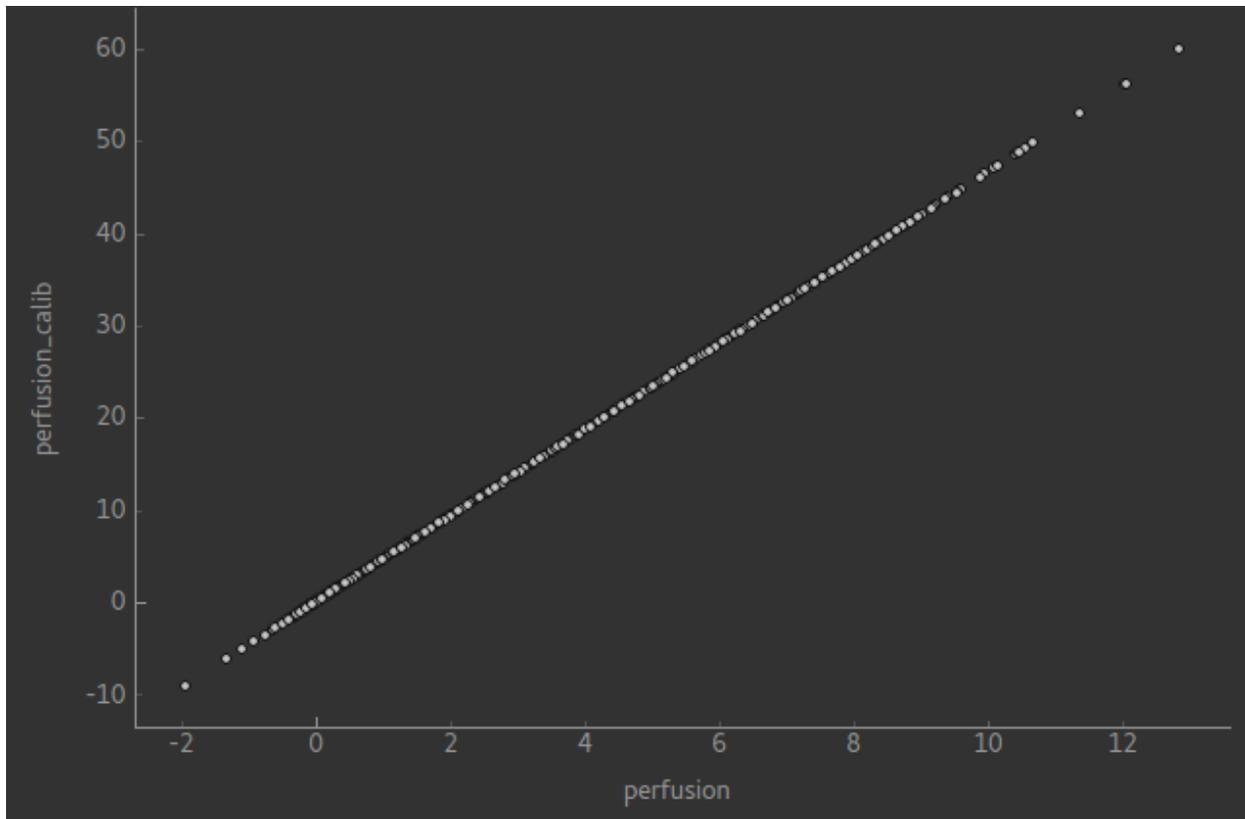
Parameter	Value
Reference type	CSF
Reference ROI	csfmask
Reference T1 (s)	4.30
Reference T2 (ms)	750.00
Reference partition coefficient (ms)	1.15
Sequence TE (ms)	0.00
Blood T1 (s)	150.00

The reference region method requires a `Reference ROI` which identifies a particular tissue type. This would normally be created by a segmentation tool such as FAST, however you could also use the `ROI Builder` to identify a region of the calibration image of a known tissue type. CSF is the most common.

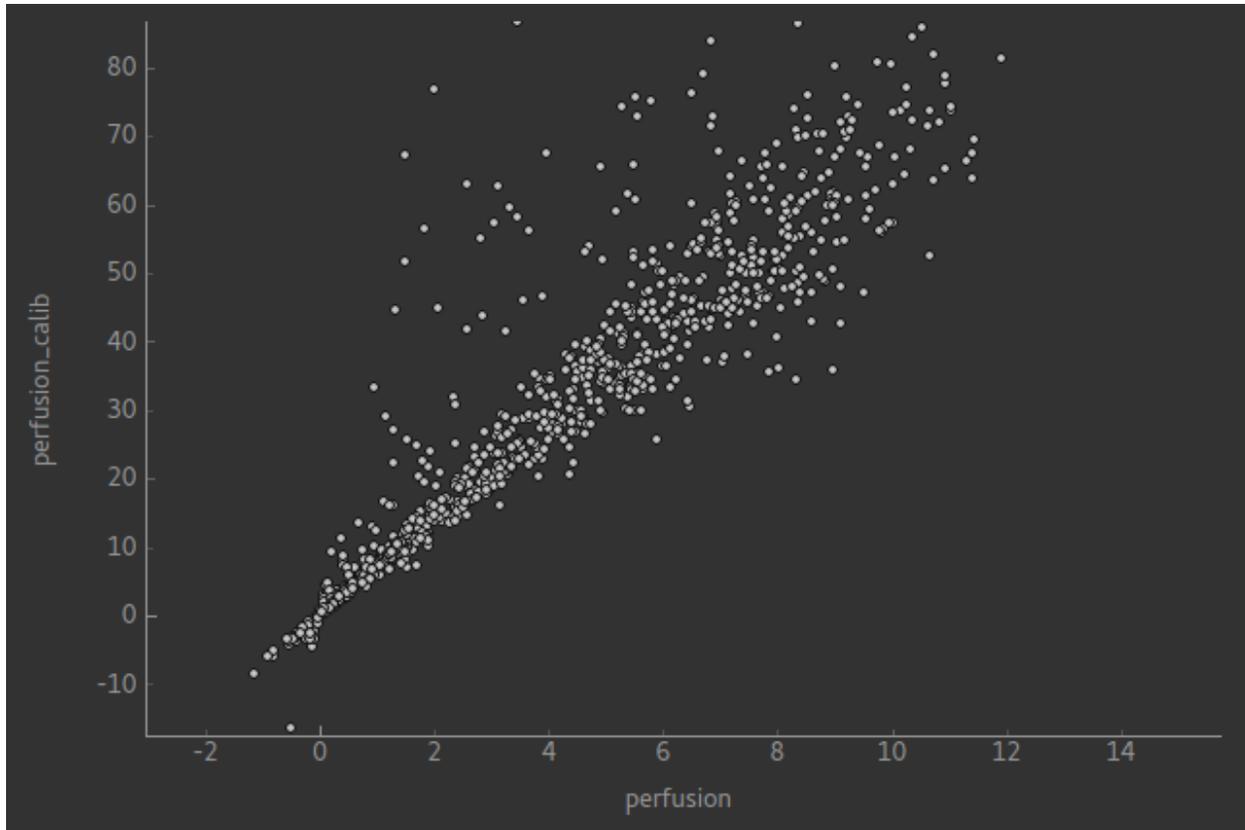
The T1, T2 and partition coefficient for this tissue type must be specified. Default values are provided for CSF, WM and GM. In addition the sequence TE and blood T1 estimates are required.

Output

Calibration returns a new data set with the suffix `_calib`, for example `perfusion_calib`. In the case of reference region calibration, the calibrated image will simply be a scaled copy of the original perfusion image - as shown if we use the `Compare Data` widget:



In voxelwise calibration this is not the case although there should still be an approximate linear relationship in the areas of interest - the comparison below is within a brain mask:



Multiphase ASL

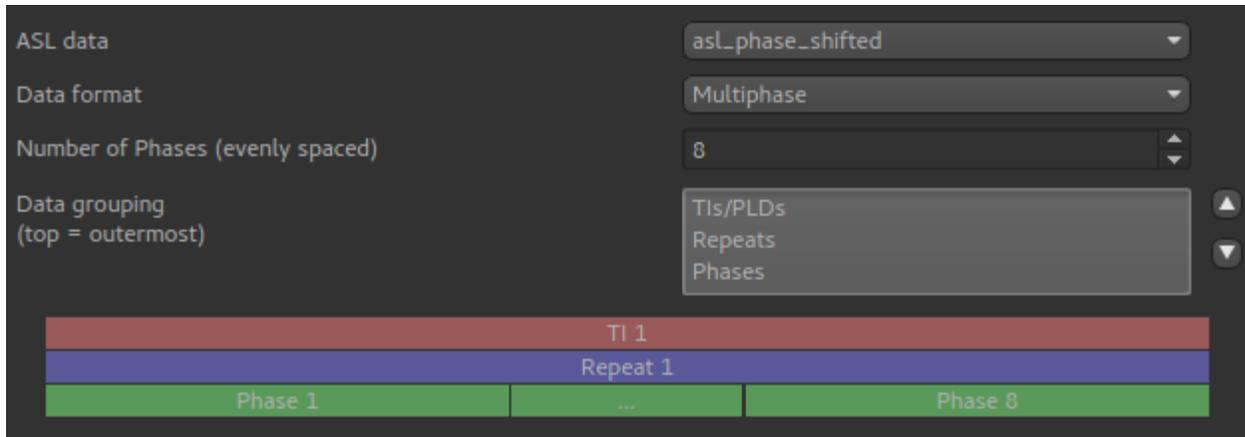
The Multiphase ASL widget is designed for single PLD multiphase ASL data. It performs a model-based fitting process which results in a net magnetisation map which can be treated as differenced ASL data.

Defining the structure

To begin you must define the structure of your ASL data, in the same way as with the other ASL widgets. Multiphase modelling is currently possible only for single PLD data, hence the PLDs entry is not visible. The main things to set are:

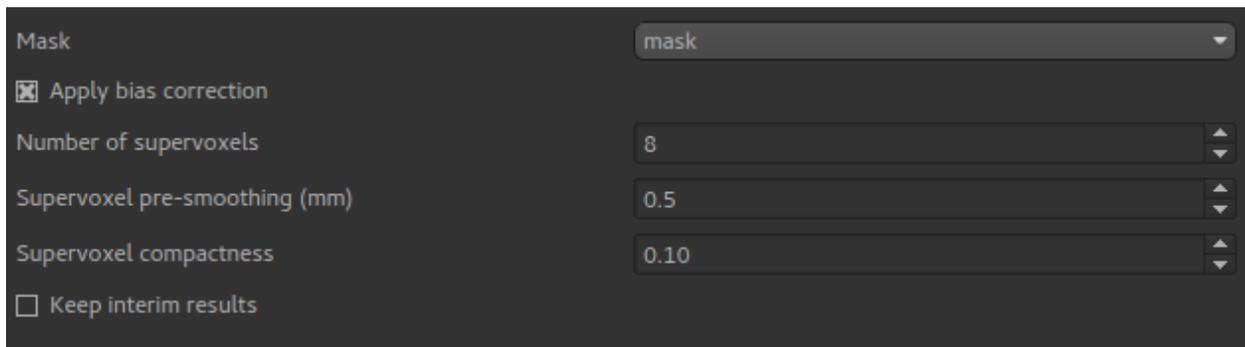
1. The number of phases, which are assumed to be evenly spaced
2. The data ordering, i.e. whether repeats of each phase are together, or whether the full set of phases is repeated multiple times. This is not relevant if the data is single-repeat

This data structure shows a simple single-repeat multiphase data set with 8 phases.



Analysis options

Bias correction



One issue with multiphase modelling is that the estimates of the magnetisation are biased in the presence of noise (in general a lower signal:noise ratio causes an overestimate of the magnetisation). The bias correction option (enabled by default) performs a series of steps to reduce this bias using a supervoxel-based method to first estimate the phase offset in distinct regions of the data. This phase offset is then fixed for the final modelling step, which eliminates the bias which only occurs when magnetisation and phase are both free to vary.

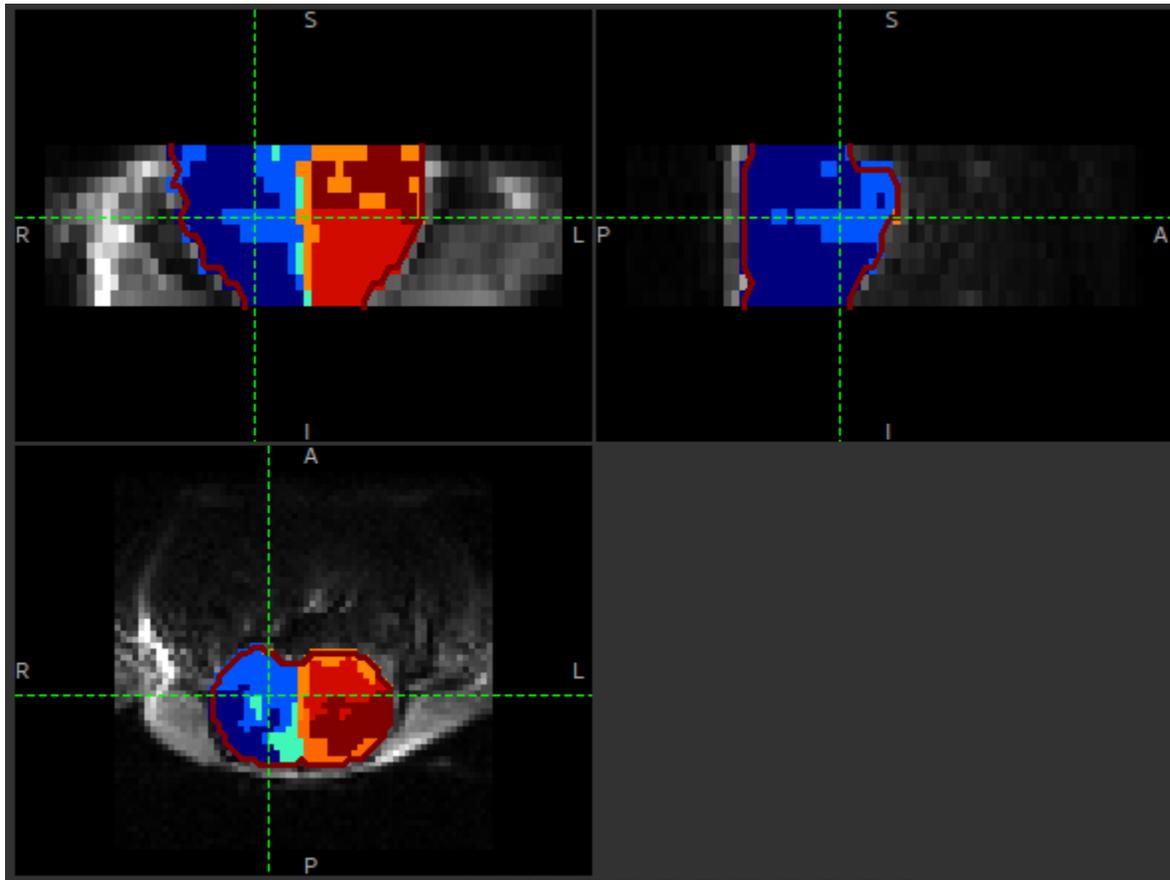
The full set of steps for bias correction are as follows:

1. An initial modelling run is performed, allowing both magnetisation and phase to vary.
2. A set of supervoxels are created based on the output *phase* map only.
3. The signal is averaged in each supervoxel, and a second modelling step is performed. The averaging increases the SNR in each supervoxel to give an improved estimate of the phase in each supervoxel region.
4. A final modelling step is performed with the phase in each supervoxel region fixed to the value determined in step 3. However the magnetisation and overall signal offset are free to vary independently at each voxel (i.e. are not constant within each supervoxel region). With the phase held constant, the biasing effects of noise are eliminated.

The supervoxels step requires a choice for the total number of supervoxels, the compactness and the degree of pre-smoothing. See the [Supervoxels widget](#) for more information about how these options are used.

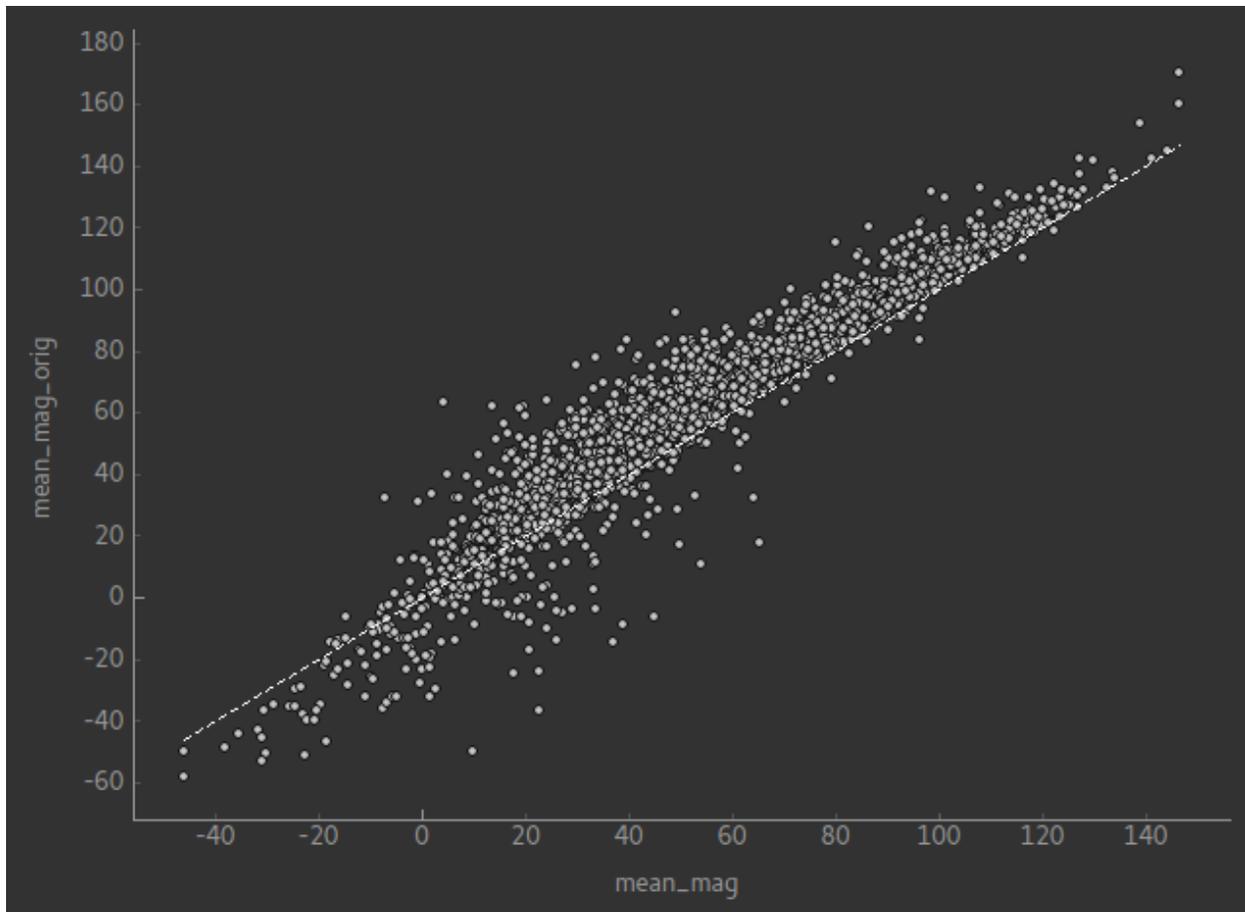
The justification for assuming a constant phase in distinct regions is that this is related to distinct vascular territories (although there is no assumption of a 1:1 mapping between supervoxels and territories). For example, the following

image shows the phase map for a data set with a significant left/right phase difference:



The supervoxels used in the phase mapping can be seen clearly.

This image shows a comparison between the magnetisation map with and without the bias correction. The systematic over-estimation of the magnetisation is clear



By default, only the resulting magnetisation map is returned by the process. By enabling the `Keep interim results` option all the data generated during the process can be preserved. This includes the original uncorrected outputs (suffix `_orig`), the averaged outputs within the supervoxels (suffix `_sv`) and the supervoxels ROI itself (`sv`)

3.4 Advanced Tools

3.4.1 Batch processing

Often it is useful to be able to run a set of analysis / processing steps on a whole set of files, without needing to manually load and save the files separately within the GUI. Quantiphyse provides a simple batch processing system which gives access to most of the processing steps available from the GUI.

Batch files are written in YAML syntax. Below is a simple example.

```
# Example config file for running Fabber

OutputFolder: out
Debug: True

Processing:
  - Load:
    data:
      mri.nii:
```

(continues on next page)

(continued from previous page)

```

    rois:
      roi.nii: mask

- Fabber:
  method: vb
  max-iterations: 30
  model: poly
  noise: white
  degree: 2
  save-mean:

- Save:
  mean_c0:
  mean_c1:
  mean_c2:
  mask:
  mri:

Cases:
  Subj0001:
    InputFolder:  c:\mydata\0001
  Subj0003:
    InputFolder:  c:\mydata\0003
  Subj0003:
    InputFolder:  c:\mydata\0003

```

The batch file is divided into three main sections. First we have statements to set up default options which will apply to all cases. In this example we are putting all output data in the `out` folder and enabling Debug messages.

The next section defines a series of processing steps. This usually starts with a Load statement and typically ends with Save. In this case we are loading a data file and an ROI which have the same filename for each of our cases, however this file name is interpreted relative to the individual case folder so different data is loaded each time.

After loading the data we run the Fabber modelling tool. Options to provide to the tool are given here.

Finally we save the three output data sets generated by the Fabber process, as well as the main data and ROI. These files will be saved in a subdirectory of the output folder specific to the case.

The third section of the batch file is a list of `Cases`. The processing steps will be run separately on each case and the output saved in separate subdirectories of the output folder.

Output

The output from processing is stored in `OutputFolder` in a subdirectory named by the case identifier (e.g. `Subj0001`). Processing steps may also generate a log file (e.g. `Fabber.log`) in the same subdirectory. In the above example we would expect the following output structure:

```

out/Subj0001/mri.nii
out/Subj0001/mean_c0.nii
out/Subj0001/mean_c1.nii
out/Subj0001/mean_c2.nii
out/Subj0001/mask.nii
out/Subj0001/Fabber.log
out/Subj0002/mri.nii
out/Subj0002/mean_c0.nii
out/Subj0002/mean_c1.nii

```

(continues on next page)

(continued from previous page)

```
out/Subj0002/mean_c2.nii
out/Subj0002/mask.nii
out/Subj0002/Fabber.log
out/Subj0003/mri.nii
out/Subj0003/mean_c0.nii
out/Subj0003/mean_c1.nii
out/Subj0003/mean_c2.nii
out/Subj0003/mask.nii
out/Subj0003/Fabber.log
```

Overriding processing options within a case

If a particular case needs options to be varied, you can override any of the toplevel options within the case block. For example.

```
Cases:
  Subj0001:
    InputFolder:  c:\mydata\0001
    # This case does not converge in 30 iterations
    Fabber:
      max-iterations: 100
```

Or, if one case is causing problems we might enable debug mode just for that case

```
Debug: False

Cases:
  Subj0005:
    InputFolder:  c:\mydata\0005
    # What's going on here?
    Debug: True
```

Multiple processing steps

The Processing block contains a list of steps, which will be performed in order. For example this example performs motion correction on the main data, followed by PK modelling:

```
Processing:
- Moco:
  method: deeds
  replace-vol: True
  ref-vol: 14
- PkModelling:
  model: 1
  fa: 30 # degrees
  tr: 5.0 # ms
  te: 2.2 # ms
  dt: 0.5 # temporal resolution (s)
  r1: 3.7 # T1 Relaxivity of contrast agent
  r2: 4.8 # T2 Relaxivity of contrast agent
  ve-thresh: 99.8 # Ktrans/kep percentile threshold
  tinj: 60 # Approximate injection time (s)
```

Extras

Extras are data created by processing modules which are not voxel data, but can be saved to a text file. They can be saved in the same way as data using the `SaveExtras` command. For example the `CalcVolumes` process calculates the volume of each region of an ROI and outputs a table extra.

```
OutputFolder: out

Processing:
  - CalcVolumes:
    roi: mask
    output-name: roi_vols

  - SaveExtras:
    roi_vols:

Cases:
  Subject1:
    Folder:  c:\Users\ctsu0221\build\data
    Load:
      data:
        test_data.nii
      rois:
        test_mask.nii : mask
```

In this case, the volume data will be saved in `out/Subject1/roi_vols.txt`. In this case the output is a tab-separated file which can be loaded into a spreadsheet.

Building batch files from the GUI

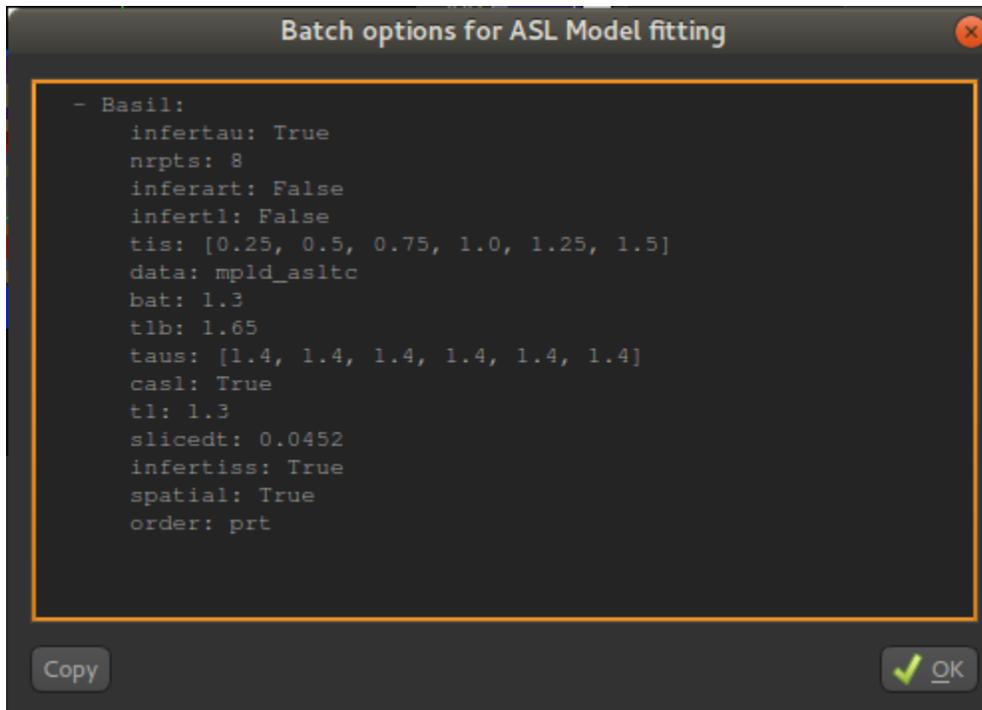
It can be convenient to build up a batch process during the course of an interactive session, for example to try out processing steps on a sample dataset and record the selected steps for later application to a group of cases. Quantiphyse provides some basic features to facilitate this.

The *Batch Button*

Many widgets support a *Batch Button* which is normally located in the top right corner, level with the widget title:



Clicking the batch button pops up a window containing the batch file code for the analysis process currently defined by the widget's GUI controls. For example, here is the result of clicking the batch button on the ASL model fitting widget after we have set up a multi-PLD analysis:



```
- Basil:
  infertau: True
  nrpts: 8
  inferart: False
  infertl: False
  tis: [0.25, 0.5, 0.75, 1.0, 1.25, 1.5]
  data: mp1d_asl1tc
  bat: 1.3
  tlb: 1.65
  taus: [1.4, 1.4, 1.4, 1.4, 1.4, 1.4]
  casl: True
  tl: 1.3
  slicedt: 0.0452
  infertiss: True
  spatial: True
  order: prt
```

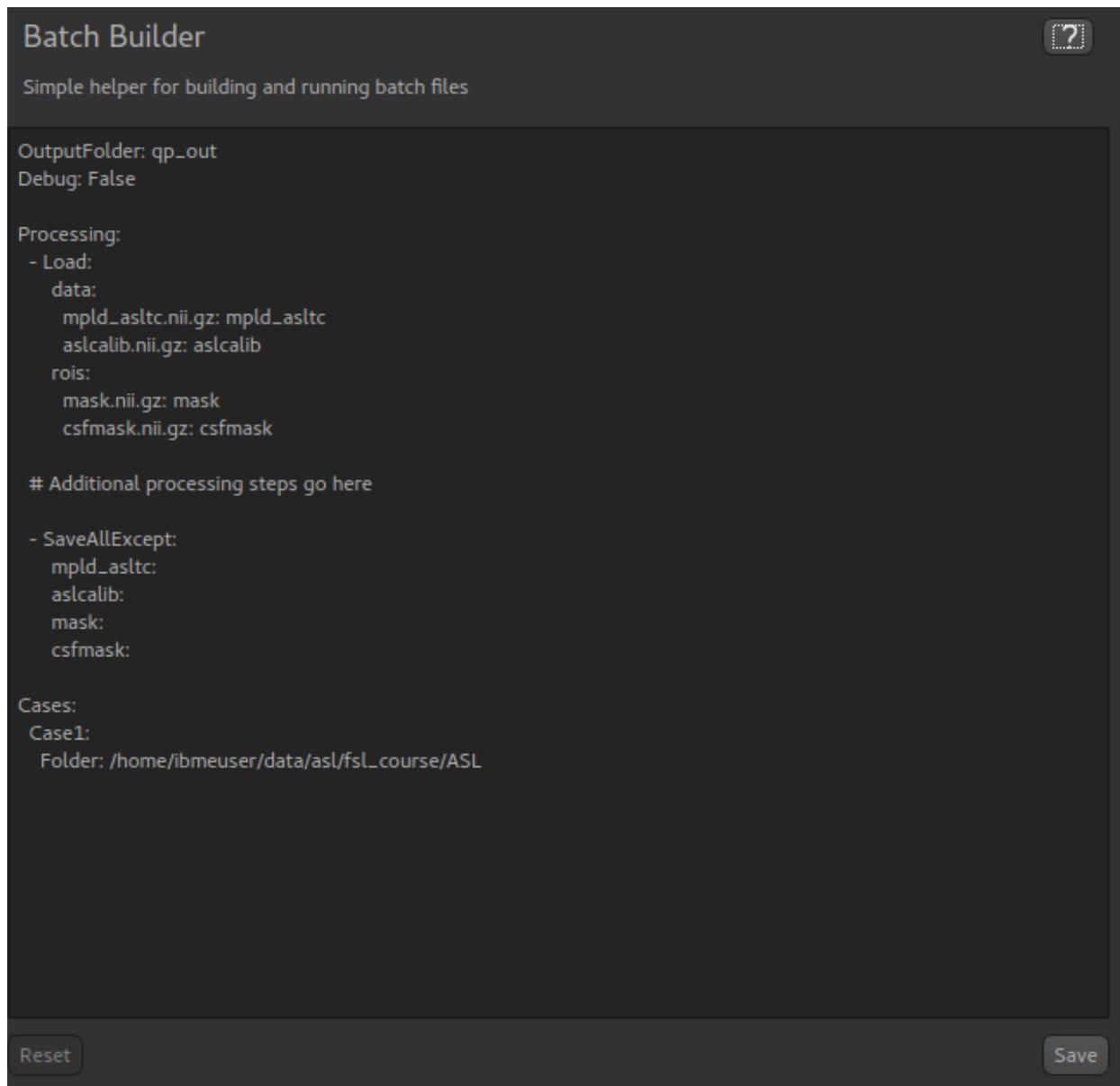
Copy OK

The `Copy` button copies this code to the clipboard where it can be pasted into a batch script that you are creating in a text editor, or using the `Batch Builder` widget (see below).

The *Batch Builder* widget

This widget is available from the ‘Utilities’ menu and gives a simple editor for batch scripts.

When first opened, a skeleton batch script will be generated which loads all currently opened data files and then saves all new data created during the batch script (using the `SaveAllExcept` process). Here’s an example after we’ve loaded some ASL data:



This script will not do anything else, however we can copy batch code from widgets using the batch button and paste it where the script says: `# Additional processing steps go here`. So we could paste the ASL analysis code shown above:

```
OutputFolder: qp_out
Debug: False

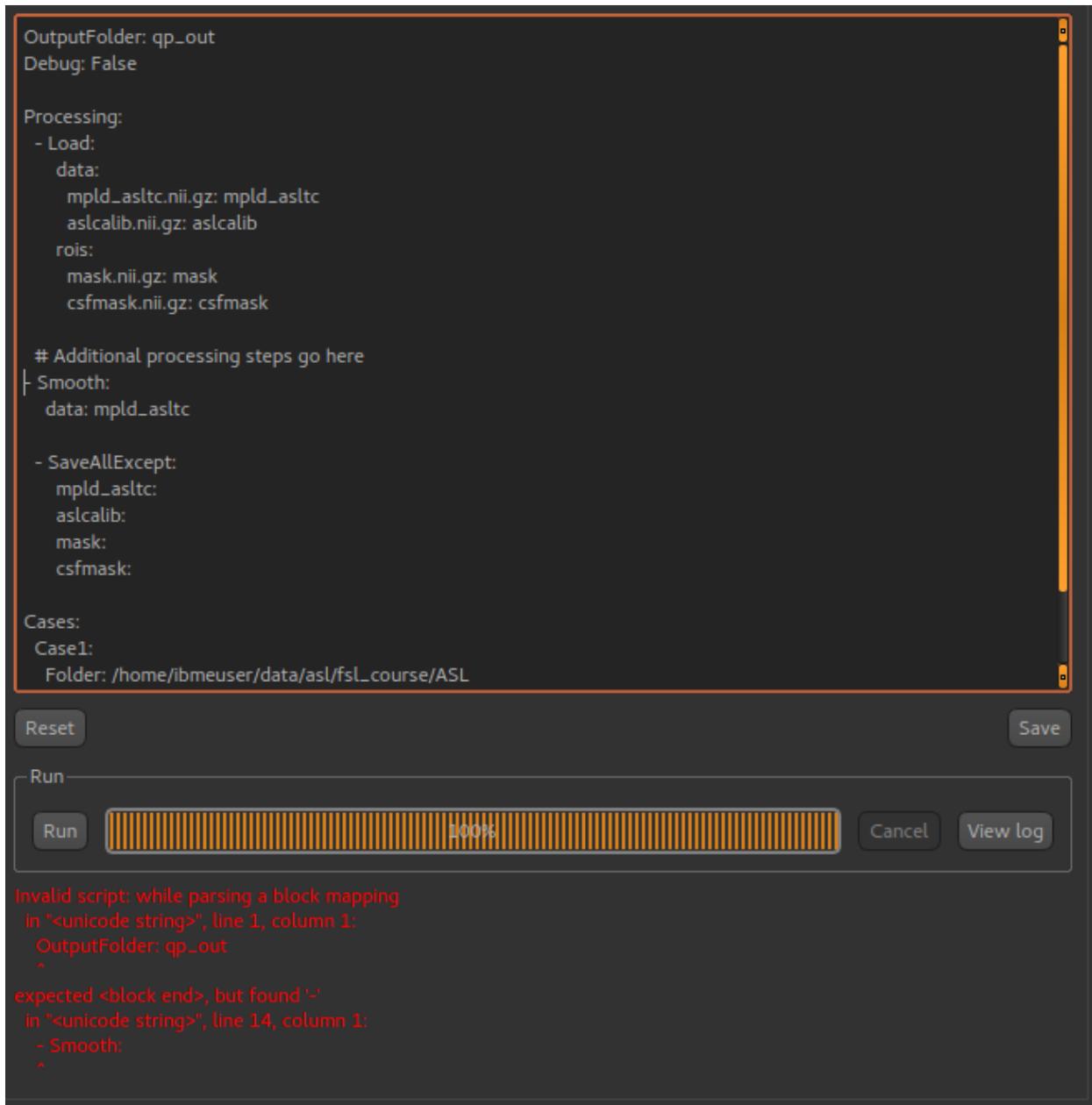
Processing:
- Load:
  data:
    mpld_asltc.nii.gz: mpld_asltc
    aslcalib.nii.gz: aslcalib
  rois:
    mask.nii.gz: mask
    csfmask.nii.gz: csfmask

# Additional processing steps go here
- Basil:
  infertau: True
  nrpts: 8
  inferart: False
  infert1: False
  tis: [0.25, 0.5, 0.75, 1.0, 1.25, 1.5]
  data: mpld_asltc
  bat: 1.3
  t1b: 1.65
  taus: [1.4, 1.4, 1.4, 1.4, 1.4, 1.4]
  casl: True
  t1: 1.3
  slicedt: 0.0452
  infertiss: True
  spatial: True
  order: prt

- SaveAllExcept:
  mpld_asltc:
```

This batch script can be `Run` to test it, and then we use `Save` to save it to a file when we're happy. You can add cases and other processing as required. `Reset` will return to the 'skeleton' batch script with no custom processing.

The batch builder will indicate if your file contains any syntax errors, for example if we don't indent our processing steps correctly:



One common issue is the use of tabs in a batch file which is not allowed but can cause difficult to interpret errors. Therefore, if you use a tab character in the batch builder it will check and simply give a warning of `Tabs detected`.

Future extensions

The batch system may be extended in the future, however it is *not* intended to be a programming language and basic facilities such as loops and conditionals will not be implemented. If your processing pipeline is complex enough to require this the suggested method is to write the process in Python, using Quantiphyse modules directly, for example:

```
from quantiphyse.volumes import ImageVolumeManagement
from quantiphyse.analysis.io import LoadProcess, SaveProcess

ivm = ImageVolumeManagement ()
```

(continues on next page)

(continued from previous page)

```
load = LoadProcess()
load.run({"data" : {"mydata.nii" : "data"}, "rois" : {"mask_43.nii.gz" : "roi"}})

# The std() method returns the data on the standard RAS grid derived from the main_
→data
numpy_data = ivm.data["data"].std()

# ...Do my processing here which may involve running additional Quantiphyse processes
#   alongside custom Python manipulations...

ivm.add_data(output_data, name="output_data")
save = SaveProcess()
save.run({"output_data":"output_data.nii.gz"})
```

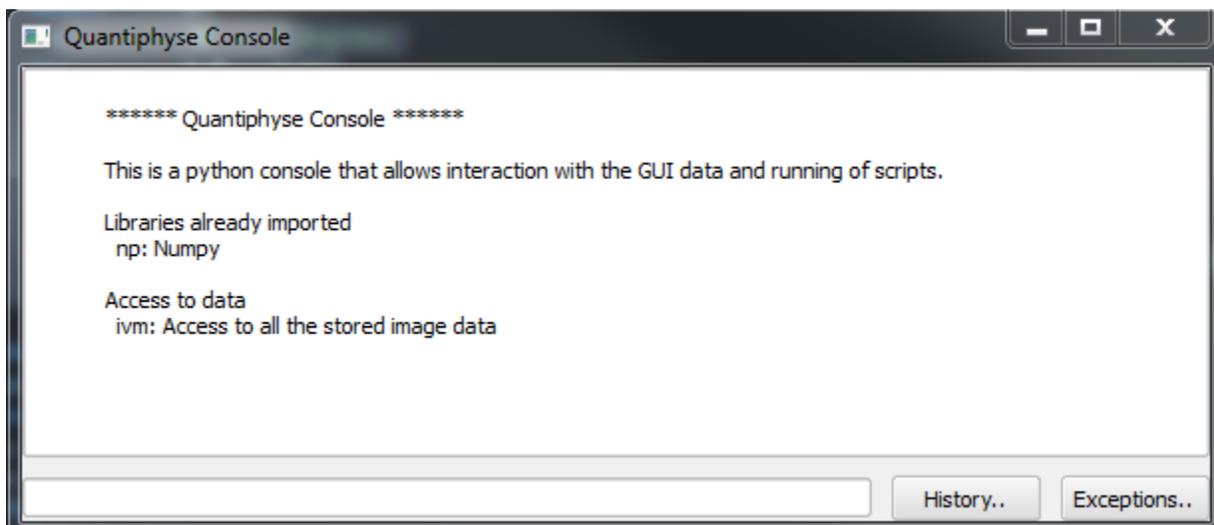
The processing modules available in the batch file are all included in the `quantiphyse.analysis` package. They all operate on data stored in the `ImageVolumeManagement` object. Data can be added to this object using the `add_data` and `add_roi` methods, which can take a Numpy array, provided it's dimensions are consistent with the current main data. This means that you can load data independently or generate it programmatically if this is required.

Warning: The volume management and analysis process APIs are *not* currently stable and you will need to read the code to see how to use them - a stable API may be defined in the future for this purpose.

3.4.2 Using the console

The console is an advanced tool which allows you to interact directly with the data structures within the program. You might use this to perform processing steps which don't have a predefined widget, using the full power of Python and the Numpy and Scipy libraries.

To open the console, select `Console` from the `Advanced` menu.



Objects provided

The main predefined variables are:

- `ivm` - The volume management object. It provides the `add_data` and `add_roi` methods you need to get data into the viewer
- Each existing data item is a named variable - for example if you have an overlay named `T10` there will be a variable `T10` which contains the data.

The following namespaces are predefined:

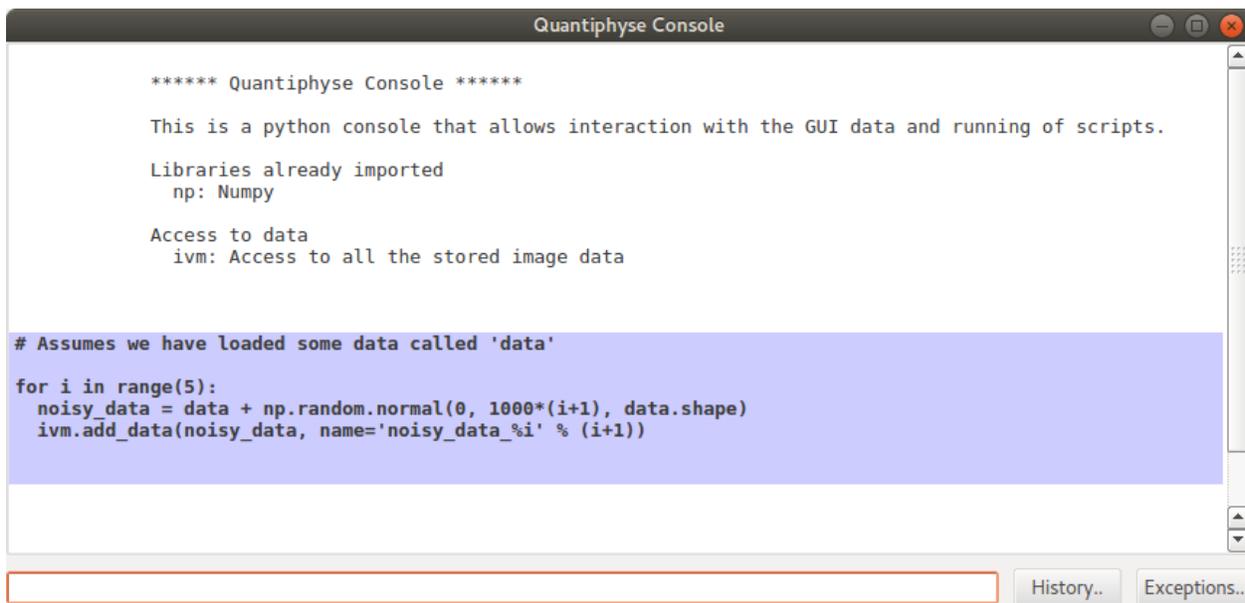
- `np` - The Numpy module

Working with data

Data objects are subclasses of Numpy arrays and can support any operations on them. To add new data into the viewer you use the `add_data()` or `add_roi()` methods.

Examples

- Create a series of data objects by adding varying levels of Gaussian noise to an existing data set



```
Quantiphyse Console

***** Quantiphyse Console *****

This is a python console that allows interaction with the GUI data and running of scripts.

Libraries already imported
  np: Numpy

Access to data
  ivm: Access to all the stored image data

# Assumes we have loaded some data called 'data'
for i in range(5):
    noisy_data = data + np.random.normal(0, 1000*(i+1), data.shape)
    ivm.add_data(noisy_data, name='noisy_data_%i' % (i+1))

History.. Exceptions..
```

This creates 5 new data sets containing the original `test_data` plus random Gaussian noise with mean 0 and standard deviations between 1000 and 5000.

	Name	Type	File
1	data	Data*	/home/lbmeuser/data/Martin_test_data/RIT005_PRE_modelling/Ktrans...
2	noisy_data_1	Data	
3	noisy_data_2	Data	
4	noisy_data_3	Data	
5	noisy_data_4	Data	
6	noisy_data_5	Data	

3.4.3 NIFTI metadata extension

Quantiphyse stores various metadata about its data sets which it would be useful to persist across loading and saving. The NIFTI format provides for this in the form of ‘header extensions’.

Each header extension is identified by a code number so software can choose to pay attention only to header extensions that it knows about. Quantiphyse has been assigned the code 42 for its header extensions.

Quantiphyse extensions will be stored as strings in YAML format for easy serialization/deserialization to Python and because YAML is already used as the basis for the batch format.

There has been suggestion that `nibabel` may add its own metadata as a NIFTI extension. This might enable some of the Quantiphyse metadata to be deprecated, however this is not available at present.

The following set of metadata is an initial proposal, however any widget can save its own metadata by adding a YAML-serializable object to the data sets `metadata` dictionary attribute. Hence this list is not exhaustive.

Generic metadata

```
Quantiphyse:
  roi : True/False           # Whether the data set should be treated as an ROI
  regions :                   # ROI regions (codes and names)
    1 : tumour
    2 : nodes
  raw-2dt : True             # Indicates that 3D data should be interpreted as 2D+time
  dps: 3                      # Suggested number of decimal places to display for values
```

ASL data set structure

```
AslData:
  tis : [1.4, 1.6, ...]      # List of TIs
  plds : [2.5, 2.6, ...]    # List of PLDs, alternative to TIs
```

(continues on next page)

(continued from previous page)

```
rpts : [4, 4, 4, ...]      # Repeats at each TI/PLD
phases : [0, 45, 90, ...] # Phases in degrees for multiphase data
nphases : 8               # Alternatively, number of evenly-spaced phases
```

CEST data set structure

```
CestData:
  freq-offsets : [-300, -30, 0, 10, 20, ...] # Frequency offsets
  b0 : 9.4                                     # Field strength in T
  b1 : 0.55                                   # B1 in microT
  sat-time : 2                               # Continuous saturation time in s
  sat-mags : [1, 2, 3, 4, ...]               # Pulsed saturation magnitudes
  sat-durs : [1, 3, 2, 4, ...]               # Pulsed saturation durations in s
  sat-rpts : 1                               # Pulsed saturation repeats
```

Images copyright 2018 University of Oxford