README asl\_analysis

% function T=asl\_analysis(src,subj)

[cvr\_default, dims, scales, bpp, endian]= read\_avw([src subj '/func/' subj '\_cvr\_default']);

[cvr\_paramA, dims, scales, bpp, endian] = read\_avw([src subj '/func/' subj '\_cvr\_paramA']);

[cvr\_paramB, dims, scales, bpp, endian] = read\_avw([src subj '/func/' subj '\_cvr\_paramB']);

# read\_avw

* comes with fsl

Builds up the directory for source subj no – need to update these to derivatives

• - fills in the variables -

• **Dataset** (CVR\_MRI parameter)

• **dims** = dimension of the matrix – 64 rows, cols and so many slices , and so many volumes

• **scales** = list of dimesons e.g. 3.5x3.5.4 x time

• **bpp** = byte per pixel – so that if you can save out with same ‘byte’

• r**ndian** = how numbers stored - determine the number, some read from big end some read second then first big endian.

• Path read in = sourcedata folder/subjectnumber/func/subject\_cvr\_MRIparameter

• **Inputs**: sourcedata folder/subjectnumber/func/subject\_cvr\_MRIparameter

• **Outputs**: Loads in default, paramA & paramB cvr maps and assigns variables

## gm\_cortex

gm\_cortex=read\_avw([src subj '/func/' subj '\_GM\_MZeroScan']);

Reads in the cortex map that we created GM\_MZeroScan and assigns it to variable called ‘gm\_cortex’

• **Input**: sourcedata folder/subjectnumber/func/subject\_ GM\_MZeroScan

• **Output**: variable gm\_cortex

## figure

figure; %creates a figure

imagesc(cvr\_default(:,:,6),[0 2]); %displays cvr\_default as a sanity check

colorbar; %colour bar displayed in figure

# reshape

cvr\_defaultr=reshape(cvr\_default,64\*64\*12,1);

cvr\_paramAr=reshape(cvr\_paramA,64\*64\*12,1);

cvr\_paramBr=reshape(cvr\_paramB,64\*64\*12,1);

gm\_cortexr=reshape(gm\_cortex,64\*64\*12,1);;

**Cvr\_defaultr=reshape** - takes cvr default map – put an r on the end of the variable for a ‘reshaped map' – reshape pass through matrix and tell it how to look like – instead of 4D it goes down to 2 – each column is a time point

The last variable on the thing

**Input**: cvr\_MRIparameter, dimensions of the file

**Output**: reshaped file called ‘cvr\_MRI parameterr

cvr\_default\_vals=cvr\_defaultr(gm\_cortexr>0,:);

cvr\_paramA\_vals=cvr\_paramAr(gm\_cortexr>0,:);

cvr\_paramB\_vals=cvr\_paramBr(gm\_cortexr>0,:);

Gone from matrix with a stack – but also in time (not in this case) to each time point you have a column

Time is subsequent columns so that if you have a cvr map and another column w. GM mask

• For the GM column you can say GM >0 ===> another col - > has a 1 or 0 for voxel w. any datapoint >>0 so it creates a mask

• Then takes cvr map and put the mask into the argument it will give values for each point that is picked out and then from there calc the median of the cvrmask

If you have more than one column then change it to go to CVR(mask, ;) so it takes all the things instead of just one column of numbers have a column for even time point and median down the column instead of across their rows

When you do reshape the quickest way is to take the param which has multiple dimensions – have param=pparam(:)

• If you have more than one column then you need to change the number on the end to have the number of time points

• - you could pull these from dimes - > 64 – dims 1 12 – dims3,

Could end up with 30 time points - > have that in the dimensions

**gm\_cortex** will only have one time point

# Set up table inputs

rows={'default'; 'paramA'; 'paramB'};

cvrmeans(1,:)=mean(cvr\_default\_vals);

Cvr\_default \_vals ==> so gm\_cx >>0 – col of 1 and 0s so those w. > 0

• > cvr\_default\_vals is acol of numbers picked out pf cvrmap and from there do stuff to is

• Cvrmeans - numbers into row 1

• Cvr\_default\_vals will only be one number – number that have been picked out to give column of one number

• so it could just say row 1 is the default values

• Row 2 is the parameterA values

• row 3 is he parameter B values

• - made it more confusing by saying that calc the mean and medians in difference ways

• Cvrmeans takes means of num you get

cvrmeansnoZ(1,:)=mean(cvr\_default\_vals(cvr\_default\_vals~=0));

MeansnoZ – removes any numbers that are 0 -> so only actually calculating stuff w.in the mask

• so look at cvr\_default vales – check any are not == to 0

cvrmedians(1,:)=median(cvr\_default\_vals);

cvrmediansnoZ(1,:)=median(cvr\_default\_vals(cvr\_default\_vals~=0));

cvrmeans(2,:)=mean(cvr\_paramA\_vals);

cvrmeansnoZ (2,:)=mean(cvr\_paramA\_vals(cvr\_paramA\_vals~=0));

cvrmedians(2,:)=median(cvr\_paramA\_vals);

cvrmediansnoZ(2,:)=median(cvr\_paramA\_vals(cvr\_paramA\_vals~=0));

cvrmeans(3,:)=mean(cvr\_paramB\_vals);

cvrmeansnoZ(3,:)=mean(cvr\_paramB\_vals(cvr\_paramB\_vals~=0));

cvrmedians(3,:)=median(cvr\_paramB\_vals);

cvrmediansnoZ(3,:)=median(cvr\_paramB\_vals(cvr\_paramB\_vals~=0));

Do same for medians

And mediansnoZ

• For all 3 parameters

## Make Table

T=table(rows,cvrmeans,cvrmeansnoZ,cvrmedians,cvrmediansnoZ);

Makes a table

T=table – names of rows – variables of the columns names - > in command window you have a table w. headings of rows, cvrm, … and numbers listed down the columns

- that table T is produced by the funcn

## Make Histograms

figure;

hist(cvr\_default\_vals,1000)

title('CVR default');

figure;

hist(cvr\_paramA\_vals,1000)

figure;

hist(cvr\_paramB\_vals,1000)

**hist ->** Put it in that histogram –

- do one for each

- so you can look at it and see if mean is better representation to the peak or the median 0

Often the median is better, in some cases – when less noisy data then mean and median is sim

If noisy then mean and median will be far apart but median more likely to be close to the peak