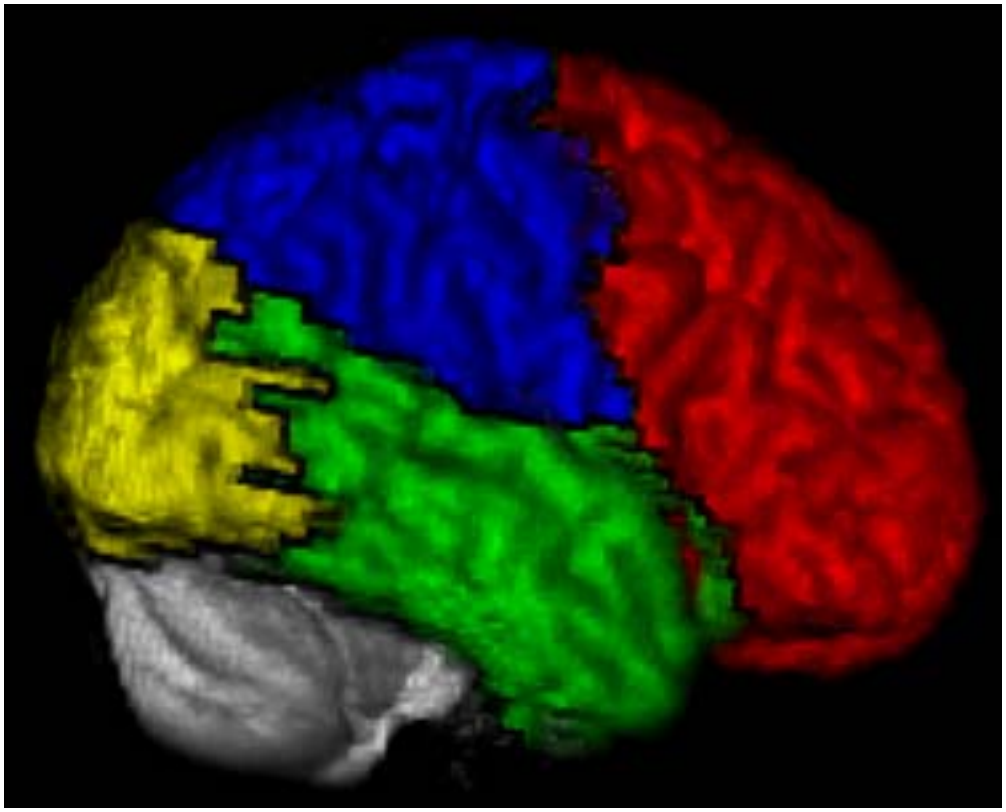
WFU PickAtlas Developers Manual v3.0

ANSIR Laboratory
Wake Forest University School of Medicine



WAKE FOREST
UNIVERSITY
SCHOOL *of* MEDICINE

WFU PickAtlas version 3.0 Developers Manual
Joseph Maldjian, MD

1. Creating a new atlas

Creating a new atlas for use with the WFU PickAtlas is composed of 3 separate tasks: image compliance, atlas descriptions creation, and creation/insertion of the lookup files. The instructions below assume a working installation of SPM5 or better, basic Matlab knowledge, and basic familiarity with the operating system on which Matlab is running. Earlier SPM versions will have to change assorted commands, especially when dealing with the image headers. Additionally, the sample files are in NiFTI format which cannot be read below SPM5.

1.1 Sample atlas information and starting off

It is recommended that you place your source images and any image description texts into a new directory under the wfu_pickatlas directory. This example will take the directory from the samples dataset in the example_atlas directory and eventually place them in wfu_pickatlas/ sampleAtlas.

Using your preferred method of filesystem manipulation, create a directory under the wfu_pickatlas directory called sampleAtlas. Place a copy of the files MNI_T1.nii and brod-double-odd.nii in this new directory.

This atlas currently has the following voxel values to regions mapping:

Voxel	Mapping
1	Inter-hemispheric
2.7	Brainstem
2	Cerebellum
3	Cerebrum

1.2 Image Compliance

All images used in a pickatlas atlas must satisfy the following requirements:

- ✓ be of a int/uint data type;
- ✓ be in the same orientation (radiological or neurological) ;
- ✓ and, be in the same voxel space.

1.2.1 Data Type Checking and Conversion

When the pickatlas is referencing the atlases images in an atlas, it expects them to be of an INT/UINT type. This is because Matlab often will not find a float whose displayed value of 1.0000 equal to the integer 1. Using spm_vol will allow one to view the data type which is stored in the dt field. Spm_type will let you know what the datatype is in generic terms as opposed to a number. Also, one should check that all the values in the volume are reasonable. Below is an example of checking the brod example atlas.

Checking image data types (SPM5)

```
>> cd wfu_pickatlas\sampleAtlas\
>> f='brod-double-odd.nii';
>> h=spm_vol(f)  %read in the images header information

h =

      fname: 'brod-double-odd.nii'
        mat: [4x4 double]
        dim: [91 109 91]
         dt: [16 0]
      pinfo: [3x1 double]
          n: [1 1]
    descrip: 'spm - 3D normalized'
    private: [1x1 nifti]

>> spm_type(16)  %show the data type

ans =

float32

>> v=spm_read_vols(h);   %read in the volume.  USE A SEMI-COLON HERE!
>> unique(v)

ans =

         0
    1.0000
    2.7000
    3.0000
    4.0000
```

The above image has three areas of concern (bold red highlight): it is of a float data type and the unique values do not roughly form integers. The last item, rounding to integers is important as when this volume is converted to a UINT type, all values will be rounded. The 2.7 will round to 3 and thus those mappings would be lost unless care was taken to modify the volume. The order of resolution will be to first move the 2.7 voxels to the available value of 2, change the data type of the volume and then change the data type listed in the header.

First attempt at resolving image to integer values

```
>> v(find(v(:)==2.7))=2;    %use semi-colon

>> unique(v)

ans =

         0
    1.0000
    2.7000
    3.0000
    4.0000

>> find(v(:)==2.7)

ans =

   Empty matrix: 0-by-1

>> format long
>> unique(v)

ans =

                   0
   1.000000000000000
   2.700000047683716
   3.000000000000000
   4.000000000000000
```

The first command should set the values in v of 2.7 to 2. However, the unique command shows that no voxels were set to 2. This is because the targeted voxel values are not actually 2.7, but 2.700000047683716. One could change the initial command to this value, but it is possible that this value is also rounded when shown on screen. As the values are well separated, a multi-part find will work.

Resolving image to integer values

```
>> v(find(2.69 < v(:) & v(:) < 2.71))=2;  %use semi-colon
>> unique(v)

ans =

     0
     1
     2
     3
     4
```

The data should now be converted to the INT/UINT data type. As the background image and this atlas image are known to contain only positive values and 0, a UINT type has been selected. If negative values were included, an INT type should be selected. Also, select a bit selection that will encompass all values in your images. This information is summarized in table 1. In general, a data type of UINT8 will suffice for most atlas images.

Table 1

Matlab Data Type	Min Value	Max Value	spm_type
UINT8	0	255	2
INT16	-32,768	32,767	4
INT32	-2,147,483,648	2,147,483,647	8
SINGLE (spm classifies as float32)	-3.4028235e+038	3.4028235e+038	16
DOUBLE (spm classifies as float64)	-1.797693134862316e+308	1.797693134862316e+308	64
INT8	-128	127	256
UINT16	0	65,535	512
UINT32	0	4,294,967,295	768

Items that have been grayed out are not compatible when generating images for WFU_PickAtlas atlases.

Next transform the volume into the chosen data type using Matlab's internal classes (data type names).

Changing volume to acceptable data type

```
>> class(v)  %this shows the 'data type' of the given variable

ans =

double

>> v=uint8(v);  %use semi-colon
>> class(v)

ans =

uint8
```

The data type and name will be changed next and the image will be saved. Change the name unless the original image is to be overwritten.

Changing the volume header

```
>> h

h =

      fname: 'brod-double-odd.nii'
        mat: [4x4 double]
        dim: [91 109 91]
         dt: [16 0]
      pinfo: [3x1 double]
          n: [1 1]
    descrip: 'spm - 3D normalized'
    private: [1x1 nifti]

>> h.fname = 'brod-double-compliant.nii';  %change filename
>> h.dt(1) = spm_type('uint8')  %change file's type.

h =

      fname: 'brod-double-compliant.nii'
        mat: [4x4 double]
        dim: [91 109 91]
         dt: [2 0]
      pinfo: [3x1 double]
          n: [1 1]
    descrip: 'spm - 3D normalized'
    private: [1x1 nifti]

>> spm_write_vol(h,v)  %save the volume

ans =

      fname: 'brod-double-compliant.nii'
        mat: [4x4 double]
        dim: [91 109 91]
         dt: [2 0]
      pinfo: [3x1 double]
          n: [1 1]
    descrip: 'spm - 3D normalized'
    private: [1x1 nifti]
```

1.2.2 Checking Orientation and Voxel Space

The background image and all images in an atlas must have the same orientation and voxel space. The first item to check is the image headers. Using spm_select, select the background image and all other images being used in the new atlas. Then check the mat and dim.

Checking Orientation and Voxel Space through SPM

```
>> f=spm_select()

f =

MNI_T1.nii
brod-double-compliant.nii

>> h=spm_vol(f);
>> h.mat   %these matrices should be the same

ans =

     2     0     0   -92
     0     2     0  -128
     0     0     2   -74
     0     0     0     1


ans =

     2     0     0   -92
     0     2     0  -128
     0     0     2   -74
     0     0     0     1

>> h.dim   %these dimensions should be the same

ans =

    91   109    91


ans =

    91   109    91

>> v=spm_read_vols(h);
```

Generally, if you can get through checking images with the above commands, especially the last one, then the images are probably in good shape. If you receive a message similar to the one below, then you should reslice your images to the template. Be sure to use nearest neighbor interpolation in your reslice.

SPM error message

```
>> v=spm_read_vols(h);

    ** The images do not all have the same dimensions. **
The function assumes that a voxel in one image corresponds with
the same  voxel in another.   This  is not a safe assumption if
the image dimensions differ.   Please  ensure  that  you  have
processed all the image data in the same way (eg. check spatial
normalisation bounding-boxes, voxel-sizes etc).
Here are the dimensions of the image volumes.  This list can be
used to determine which file(s) are causing the problem.

[91 109 91]   example_atlas\MNI_T1.nii
[256 256 124] AdultRhesus_T1.nii
```

1.3 Lookup files

All lookup tables are simple text files. They can be modified using matlab's editor or your favorite text editor. Be sure they are saved as simple text as formatting characters are not allowed and may cause the atlas not to load.

In our example, let's rename (or copy) the brod-double-compliant.nii to major_brain_labels.nii.

1.3.1 Image labels

The image labels are a mapping of voxel values to alphanumeric names. The file is generally named the same as the atlas image. It contains a small header and then the listing of voxel name mappings. The header is the naming of our region group. The list is: voxel value TAB Name. Our dataset would have the following file:

major_brain_labels.txt

```
[ Example Regions]
1 Inter-hemispheric
2 Brainstem
3 Cerebellum
4 Cerebrum
```

1.3.2 Atlas Regions

An atlas may have many different naming regions (for example the TD_labels, TD_brodmann, etc) that are incorporated into one atlas. The atlas regions file lists these regions, their respective image, image label files, and an offset. The offset is used to maintain unique values for each region in the PickAtlas. The Atlas Regions from the MNI atlas looks similar to this:

master_lookup.txt

```
TD brodmann areas+,   TD_brodmann.img,     TD_brodmann.txt,      1000
TD Lobes,             TD_lobe.img,         TD_lobe.txt,          2000
TD Hemispheres,       TD_hemisphere.img,   TD_hemisphere.txt,    3000
```

For ours we will create a master lookup file that looks like:

master_lookup.txt

```
Example Regions, major_brain_labels.nii, major_brain_labels.txt,  1000
```

1.3.3 Master File

The last file to be modified is the Atlas_types.txt in the PickAtlas directory. This contains a list of all available atlases in a coma separated form. Insert the directory (sampleAtlas) and associated information as such:

Atlas_types.txt

```
%atlas name,     sub directory,        lookup file,         display image
HUMAN ATLAS,     MNI_atlas_templates,  master_lookup.txt,   MNI_T1.img
Monkey Atlas,    monkey_templates,     master_lookup.txt,   monkey_T1.hdr
Rat Atlas,       rat_templates,        master_lookup.txt,   rat_axial.hdr
Mouse Atlas,     mouse_templates,      master_lookup.txt,   mouse.hdr
Sample Atlas,    sampleAtlas,          master_lookup.txt,   MNI_T1.nii
```

1.3.4 Optional additions to the Master File

The master file may also have two more fields: an atlas info txt/html document and a conversion file. The atlas info is a text/html document loaded in PickAtlas when the Atlas Information button is selected. If the file does not exist, or is blank, the Atlas Information button does not show. This same documentation may also be brought up in the Results Viewer through the Help menu, selecting Atlas Information.

The atlas conversion file indicates how to map voxel space to some other space. These functions should allow multiple XYZ voxels as input. If these functions currently exist in the path, they can be referenced. If they do not exist in the path, then they should be placed in the directory of the atlas (sampleAtlas in our example). The format is: Descriptor, voxel to space function, space

to voxel function, and isTalairarch setting. The isTalairarch setting allows the PickAtlas to know if the “Integrate Talairach Daemon” button should be active.

Atlas_space_conversion.txt

```
% Voxel to Space conversion functions
% The functions should take as inputs an Nx3 or 3xN array in voxel,
% and the template image MAT, and the wfu_pickatlas handles. These
% functions are evaluated as part of the cmdGoN_Callback's in
% wfu_pickatlas.m
%
% Format: reference_number, display_name, mm_to_space_function,
%         space_to_cc_function, isTalairarch
%
% isTalairarch is bool (0 or 1) with 1 representing the output
% from this function is in Talairarch space.  This allows the
% PickAtlas to know if it can integrate the Talairarch Daemon
MNI, wfu_cub2mni, wfu_mni2cub, 0
Tal, wfu_cub2tal, wfu_tal2cub, 1
```