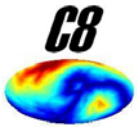


C8 Documentation

Timothy J. Herron

Human Cognitive Neurophysiology Laboratory, David L. Woods, Director
Neurology, Research Service, United States Veterans Affairs
Martinez, California, USA;

C8: May 5, 2011 Version

Overview of C8

C8 is a MatLab (www.mathworks.com) package that helps you take high-resolution (~1mm on a side) T1 human brain images and make many different measurements on the corpus callosum cross sections along the brain's midline. The C8 package takes as input a good-quality white matter segmentation image that has been reasonably well-normalized to MNI space.

C8 main quality is that it leverages the great amount of work that has been previously done in automatically segmenting and normalizing brain images, and uses it to make measurements in standard (i.e. easy to communicate) ways that have appeared in the literature. It is also fast (multiple callosa segmentations are processed per minute) and is easily modifiable having been written in the popular high-level language MatLab. For reasonably good quality T1 images for subjects/patients without too much brain matter missing, C8 can facilitate generating corpus callosum measurements in a fully automated fashion.

C8 can also be used in a semi-automated fashion for patient populations who require manual normalization and/or lower quality images that require manual white matter segmentation.

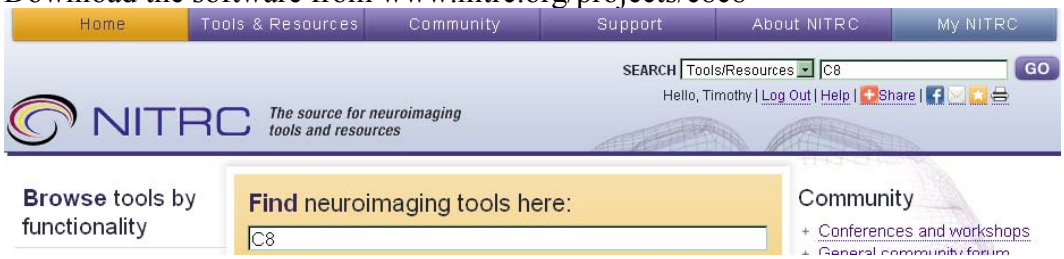
C8 can operate on standard .nii/.img NIFTI or Analyze image format files, using the NIFTI MatLab toolbox created by Jimmy Shen (these files are included).

Installation of C8

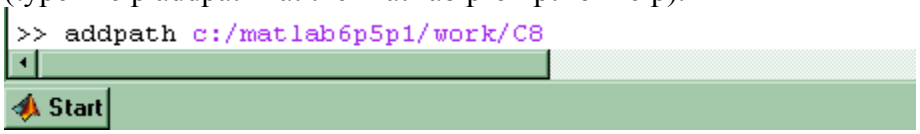
First off, you will need to run Matlab version 5.3 or greater. No other packages are needed.

Installation of C8 itself is fairly simple save for, perhaps, one step.

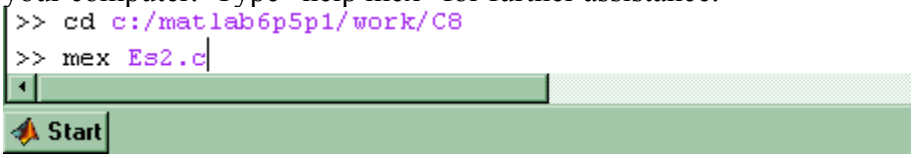
1) Download the software from www.nitrc.org/projects/c8c8



2) Unzip the package into a new directory, say …./C8, which your unzipper might do for you. You can add the new directory to your MatLab path for convenience (type "help addpath" at the MatLab prompt for help).



3) Now you may need to compile one file, Es2.c, if your system is not compatible with the Windows Es2.dll (or Linux Es2.mexgl or Es2.mexa64) included executables. (you'll know this if when you run C8, it complains about a call to "Es2.xxx" failing).
   a. To compile the code, start MatLab and type "mex Es2.c" at the command line: if you are lucky that will do it, but if not, then type "mex -setup" first and follow the prompts to try and configure any available C compiler on your computer. Type "help mex" for further assistance.



   b. If none of the above works to get you a working Es2 function, then as a last resort you can just change the name of the file "Es2.m.slow" to Es2.m" and then C8 will run, but the 3D permutation tests now run slowly.
4) Then you are ready to run – fire up MatLab and go….

Sample Analysis – Colin's Corpus Callosum

For the sample analysis, we want to measure the corpus callosum of the standard, single subject MNI Colin brain ("ColinAtlas") that we have included in the package.

Normalizing and Segmenting

First, we need to generate a white matter segmentation for the brain and then (affine) normalize it to MNI space: this normalized white matter segmentation will be the input that C8 will operate on.

In this case – see `segNormColin.m` - we have used the MatLab package SPM5 to normalize the brain to MNI space, segment the T1 image into white matter (plus gray and CSF), and then finally map the white matter segment into MNI space. Although this necessary part of callosum measurement is not part of the C8 package, I will walk you through it because the quality of the segmentation determines in large part the quality of the callosum measurements. Similar advice should apply to using any other clustering-based segmentation algorithms.

When doing the MNI normalization of the T1 image, it is important that one only do an affine (linear) normalization only so that measurements performed in MNI space can easily be back-transformed into equivalent measurements in original image space. The parameter in SPM5's function to specify affine-only is 'nits' as follows:

```
14      % choose files to normalize to
15 -    VWF = '';
16 -    VG = spm_vol([spmDir 'templates/T1.nii']);
17 -    VWG = spm_vol([spmDir 'apriori/brainmask.nii']);
18      %
19      % choose normalization parameters
20 -    aflags=struct('smosrc',8,'smoref',0,'regtype','mni','cutoff',70, ...
21            'nits',0,'reg',1);
22      % normalize to MNI
23 -    spm_normalise(VG,VF,matname,VWG,VWF,aflags);
```

The 'nits' parameter in `segNormColin.m` tells SPM5 how many non-linear normalization iterations to do – we want 0 (i.e. to turn it off).

Second, the trickier issue of segmentation. The main idea here is to do the best-quality segmentation that your computer can handle in the time that you have available. In the case of SPM5, the following 3 flags (between the red

bars) to input to the segmentation routine are the most important to consider altering from their default values:

```
sflags.estimate.reg    = 0.005; % 0.01 default
sflags.estimate.cutoff = 20; % 30 default
sflags.estimate.samp   = 2; %3 default
sflags.estimate.bb     =  [[-88 88]' [-122 86]' [-60 95]'];
sflags.estimate.affreg.smosrc = 8;
%sflags.estimate.affreg.regtype = 'mni';
sflags.estimate.affreg.weight = fullfile(spm('Dir'),'apriori',
sflags.estimate.affreg.weight = '';
sflags.write.cleanup   = 1;
sflags.write.wrt_cor   = 1;
sflags.write.wrt_brV   = 1;
spm_segment(VF,VG,sflags);
```

`sflags.estimate.samp` is a flag that tells SPM5 the spacing in *mm* of samples to use for its clustering algorithm – the lower this is the better, however the clustering algorithm uses much more memory as one goes lower (using less than around 1.5 is very difficult, e.g.).

`sflags.estimate.cutoff` is a flag that sets the lower frequency cutoff in mm of the routine that smooths out variations in T1 image intensity. Lowering this a little bit helps make sure that some T1 intensity "divots" don't reduce your regional callosal measurements.

`sflags.estimate.cutoff` is a flag that controls the amount of penalty that segmentation incurs for doing more 'aggressive' clustering. The default value is usually OK here, but moving it around a bit can help in certain cases – e.g. to prevent false minima from occuring.

Third, you will need to normalize the segmentation that you've created into MNI space. It is recommended that you use tri-linear interpolation in order to preserve the total amount of segmentation partial volume weight other than that which is altered by the affine registration matrix (C8 accounts for the affine alterations in CC area/thickness):

```
% choose normalization application parameters
wflags=struct('wrap',[0 0 0],'vox',[2 2 2],'bb',Inf*[1 1 1;1 1 1],'interp',1);
spm_write_sn(VF,matname,wflags);
```

'interp',1 tells SPM5 to use trilinear interpolation to rewrite the WM segmentation in MNI space.

Measuring Callosa with C8

For an example of calling the C8 package in order to process the ColinAtlas white matter segmentation, we look at the file `getColin.m`:

```
thisDir='./Subjects/';
aa={'ColinAtlas'};
for p=1:length(aa)
 str=char(aa(p));
 aain2{p,1}=[thisDir aa{p} '/' aa{p} '256FS.img'];
 aain2{p,2}=[thisDir aa{p} '/wc2' aa{p} '256FS.img'];
 aain2{p,3}=[thisDir aa{p} '/' aa{p} '256FS_sn.mat'];
 aain2{p,4}=[thisDir aa{p} '/w' aa{p} '256FS.img'];
% aain2{p,5}=[thisDir aa{p} '/anat/wOther.nii'];
end
[Thickness2 Area2 clustHg02 T1s2]=getCC(aain2);
```

This first half of the file processes the normalized, WM segmentation that is in 2x2x2 mm^3 voxel format (it's faster). The program getCC has only one required input: a cell structure ("aain2") listing the name of the files to process. The first cell entry contains the original T1 image file name ("ColinAtlas256FS.img"), the second one contains the normalized white matter segmentation ("wc2ColinAtlas256FS.img"), and a third containing the affine normalization matrix/structure used to normalize the WM image ("ColinAtlas256FS_sn.mat"). The last file only needs to have a 4x4 affine transformation matrix in it as variable "M" or "Affine", but if the file was generated by SPM, then C8 can interpret it without any alteration.

The 4^{th} and 5^{th} entries in the input cell structure are for optional file names of coregistered images that C8 can sample callosal values. In this case, we sample the coregistered T1 image values from the normalized T1 image ("wColinAtlas256FS.img").

The second call to C8 in the `getColin.m` file redoes the processing directly on the original Colin brain image (a 1x1x1 mm^3 image) because it already lies in MNI space to begin with.

```
[Thickness1 Area1 clustHg01 T1s1]=getCC(aain1,'thresh=0.6;');
```

The second entry in the getCC command shows how to alter the internal parameters of C8: by simply including a string that directly changes the values when executed internally.

C8 Output Variables

In the previous case:

```
[Thickness1 Area1 clustHg01 T1s1]=getCC(aain1,'thresh=0.6;');
```

The first output variable has the following substructures:

Thickness1.**EqualAngle** – Thicknesses around the CC (1=Ant->Pos=50) measured at 50 points spaced by equal angles from a specified centroid
Thickness1.**EqualAngleyMNI** – MNI y coordinate of previous points
Thickness1.**EqualAnglezMNI** – MNI z coordinate of previous points

Thickness1.**EqualDist** – Thicknesses along 50 points spaced by equal distances along a median line
Thickness1.**EqualDistyMNI** – MNI y coordinate of previous points
Thickness1.**EqualDistzMNI** – MNI z coordinate of previous points

Thickness1.**EqualArea** – Thicknesses along 50 median points spaced by equal adjacent areas.
Thickness1.**EqualAreayMNI**– MNI y coordinate of previous points
Thickness1.**EqualAreazMNI**– MNI z coordinate of previous points

Each of the above 3D (non MNI coordinate) variables is n x 50 x 3, where n is the number of subjects that were processed, 50 is the number of thicknesses output per CC, and there are 3 types of thickness measurements Thickness1.XXX(:,:,1) are thickness measurements made from the superior surface, Thickness1.XXX(:,:,2) are thickness measurements made from the inferior surface, and Thickness1.XXX(:,:,3) are thickness measurements made from the medial line through the CC. The medial line [Thickness1.XXX(:,:,3)] thickness measurements are the most accurate and robust to variations in image quality and CC shape/oddities.

Thickness1.**Witelson** – mean thicknesses within a geometrically-defined 7 partition scheme of Witelson, **Brain**, 112(3) 1989.
Thickness1.**WitelsonMNI** – ditto, but for the CC in MNI space, rather than in original space

Thickness1.**Hofer** - mean thicknesses within a geometrically-defined 5 partition scheme of Hofer & Frahm, **NeuroImage**, 32(3) 2006;
Thickness1.**HoferMNI** - ditto, but for CC in MNI space, rather than in original space

Thickness1.**Chao** - mean thicknesses within a geometrically-defined 5 partition scheme of Chao et al, **Hum Brain Mapp**, 30:3172 2009;
Thickness1.**ChaoMNI** - ditto, but for CC in MNI space, rather than in original space

Thickness1.**thresh** – the threshold value actually used defining CC clusters

Each of the above 3D variables is n x m x 3, where n is the number of subjects that were processed, m is the number of partitions per CC [5 or 7], and there are 3 types of thickness measurements Thickness1.XXX(:,:,1) are thickness measurements made from the superior surface, Thickness1.XXX(:,:,2) are thickness measurements made from the inferior surface, and Thickness1.XXX(:,:,3) are thickness measurements made from the medial line through the CC. The medial line [Thickness1.XXX(:,:,3)] thickness measurements are the most robust to variations in image quality and CC shape/oddities.

The second output variable has the following substructures:

Area1.**Witelson** - – areas within geometrically-defined 7 partition scheme of Witelson, **Brain**, 112(3) 1989.
Area1.**WitelsonMNI** - ditto, but for CC in MNI space, rather than in original space
Area1.**WitelsonBounds** – MNI y coordinates defining the vertical boundaries of the above CC partitions

Area1.**Hofer** - - area within geometrically-defined 5 partition scheme of Hofer & Frahm, **NeuroImage**, 32(3) 2006;
Area1.**HoferMNI** - ditto, but for CC in MNI space, rather than in original space
Area1.**HoferBounds** - MNI y coordinates defining the vertical boundaries of the above CC partitions

Area1.**Chao** - areas within geometrically-defined 5 partition scheme of Chao et al, **Hum Brain Mapp**, 30:3172 2009;
Area1.**ChaoMNI** - ditto, but for CC in MNI space, rather than in original space
Area1.**ChaoBounds** – MNI y coordinates defining the vertical boundaries of the above CC partitions

Note that in the above there is always 1 more vertical boundary in the "Bounds" variables than there are partitions.

Area1.**ACC** – anterior most MNI coordinate of the CC
Area1.**PCC** – posterior most MNI coordinate of the CC

The third output variable "clustHg01" is a 5-D variable that shows the outline and shape of the CC within the (peri)midline cross-section(s) of the image. It has the following dimensions:
1 – image in y (MNI) dimension
2 – image in z (MNI) dimension

3 – different CC cross-sections computed (from –"sidesl" to +"sidesl" in mm in the x (MNI) direction)
4 – subject #
5 – value 1's image contains WM segmentation values for the CC, value 2's image shows the connected interior components of the CC (+ values) and the boundaries and non-CC WM (- values)

The fourth output variable has the following substructures:

T1s1.**EqualAngle** – auxiliary image values along the equally angle spaced points
T1s1.**EqualDist** – auxiliary image values along the equal distance spaced points
T1s1.**EqualArea** – auxiliary image values along the equally area spaced points

T1s1.**Witelson –** mean auxiliary image values within the Witelson partition
T1s1.**WitelsonMed –** median auxiliary image values within the Witelson partition
T1s1.**WitelsonQuart1 –** lower quartile auxiliary image values within the Witelson partition
T1s1.**WitelsonQuart3 –** upper quartile auxiliary image values within the Witelson partition

T1s1.**Hofer –** mean auxiliary image values within the Hofer partition
T1s1.**HoferMed –** median auxiliary image values within the Hofer partition
T1s1.**HoferQuart1 –** lower quartile auxiliary image values within the Hofer partition
T1s1.**HoferQuart3 –** upper quartile auxiliary image values within the Hofer partition

T1s1.**Chao –** mean auxiliary image values within the Chao partition
T1s1.**ChaoMed –** median auxiliary image values within the Chao partition
T1s1.**ChaoQuart1 –** lower quartile auxiliary image values within the Chao partition
T1s1.**ChaoQuart3 –** upper quartile auxiliary image values within the Chao partition

The fifth output variable (when it exists) has the same structure as the fourth output variable

Options and Tips

1) In the case of damaged patient brains where there is still a fairly clear midline cross-sectional corpus callosum (fragment) to be measured.
    a. In this case you may need to move to more sophisticated normalization procedure if there is substantial brain matter missing – e.g. you can use manual brain masks (e.g. "VFW" parameter for SPM5) to tell the routine to ignore those areas when normalizing brains.
    b. It is also possible to do a very simple normalization manually – simply rotate/translate the brain so that the callosum cross-section sits on the image midline (x=0), so that the callosum sits inside a specified box (see parameters "ybox" and "zbox" inside `getCC.m`), and so that the callosum cross-section has the correct orientation inside the box (midbody superior, genu anterior, splenium posterior).

2) `getCC.m` is tuned out-of-the-box to operate on normalized segmentations that are 2x2x2 mm^3. You can move the "thresh" segmentation threshold parameters around a little, probably higher, when using C8 with smaller sized voxels.

3) In the case that you have images with quality too poor to automatically segment into white matter, then it is possible to trace out and fill the callosum cross-section using, e.g., the program MRIcron. All that matters to `getCC.m` is that there is a NIFTI/Analyze format compatible file that has values ranging from 0 to 1 with values indicating what fraction of that voxel is filled with the callosum part.

4) C8 has various methods for getting rid of the fornix that is sometimes connected to the callosum. If C8 cannot get rid of it within a certain subject by modifying thresholds (mainly "thresh" and "downthresh"), then you can manually detach it using medical image modification software.

5) C8 is not setup to use diffusion images to detect medial/lateral directional fibers indicative of the midline callosum. However, such images can be used as a prefilter to the WM segmentation before inputting to C8, thereby eliminating the fornix, e.g.

Limitations of C8

- C8's measurements are limited in accuracy by the quality of the whole brain WM segmentation that you do. So it's best to spend the time and computer memory needed perform the most accurate segmentation you can afford.
- C8 cannot always detach the fornix. In these cases the thickness measurements are generally more accurate than the area measurements because the former knows how to ignore most of the fornix.
- Right now C8 cannot detect the posterior or anterior commissures, and it currently does not even try to find them.
-

Files included in the C8 package

1) Main function and how to call it:
   a. `getCC.m  getTest.m  getColin.m`
2) Files from an older version of the NIFTI MatLab toolbox written by Jimmy Shen: these functions all have "nii" or "rri" in them:
   a. `bipolar_nii.m examples_nii.txt expand_nii_scan.m extra_nii_hdr.m get_nii_frame.m load_nii.m load_nii_hdr.m load_nii_img.m make_nii.m mat_into_hdr_nii.m rri_file_menu.m rri_orient.m rri_orient_ui.m rri_xhair.m rri_zoom_menu.m save_nii.m save_nii_hdr.m unxform_nii.m view_nii.m view_nii_menu.m xform_nii.m`
3) Functions to pull out data from 2D images:
   a. `gridInterp.m  planeNbd2.m  find2Dnbd.m  extractLines.m`
4) Functions to filter images:
   a. `mgso.m  opregset.m  polyFiltImg.m`
5) Sample preprocessing function:
   a. `segNormColin.m  (getColin.m)`
6) Computational functions
   a. `bloat.m  Es2.*  quartile.m  cluster1D.m  convexGradDesc.m`
7) Documentation
   a. `208.17_SfN_2010.pdf  C8 Documentation.pdf`
8) Subjects/ColinAtlas/ - T1 and WM segmenetation
   a. `ColinAtlas256FS.img  ColinAtlas256FS.hdr  ColinAtlas256FS.mat  ColinAtlas256FS_sn.mat  c2ColinAtlas256FS.img  wColinAtlas256FS.img  c2ColinAtlas256FS.hdr  wColinAtlas256FS.hdr  wc2ColinAtlas256FS.img  nothing.mat  wc2ColinAtlas256FS.hdr`
9) Subjects/TestSlab/ - T1 and WM artificial test case
   a. `TestSlab256FS.img  TestSlab256FS.hdr  TestSlab256FS.mat  c2TestSlab256FS.img  c2TestSlab256FS.hdr  c2TestSlab256FS.mat  nothing.mat`

C8 Parameters

```
% starting parameters
thisDir=pwd;        % directory to operate in
thresh=0.55;        % threshold that defines a connected CC component
downthresh=0.4;     % get WM segmentation lower threshold that defines clusters of
                    % the non-boundary portion of the CC
numclust=1;         % number of clusters of the CC to look for (ideally)
                    % WM segmentation threshold to look for more CC clusters
minlen=59;          % minimum Ant-Pos (MNI y) length of CC so as not to reduce the
                    % distance in mm to look for CC away from midline - will compute
sidesl=1;           % 2*sidesl+1 CC cross-sections 1 is normal
stand=50;           % Number of thickness, etc to output in standard coordinates
ybox=[-52,42];      % coordinates of box that usually contains the CC in MNI space - used
to
zbox=[-8,40];       % extract CC clusters when the CC is in more clusters than expected
MNIrot=0;           % should we rotate the CC out of MNI space to make it lie flat?
AuxFilt=1;          % should we linearly detrend the auxiliary image values?
pureCoM=0;          % use alternative centroids for angle defs - pure centoer of mass
safeCentroid=0;     % center of mass x and mean of genu splenium minimum
```

Parameter description:

thisDir – directory to put temporary files in

thresh – WM segmentation threshold value above which defined connected components of the CC when C8 is identifying those. Values lower than this on the outer boundary (ies) of the CC are included within thickness and variable measurements. This value can be raised or lowered in order to help automatically detach the fornix from the CC or to have the isthmus connect the CC into one cluster, e.g.

downthresh – If C8 cannot find CC components of total length high enough (as specified by "minlen") with the given "thresh", then C8 will start to lower the threshold, continuing until it hits "downthresh". If that still does not suffice, then C8 looks at all components within the MNI coordinates specified by "ybox" and "xbox" for potential CC parts.

numclust – the number of CC clusters that C8 expects to find. If C8 finds fewer than this number, then the WM threshold is lowered in order to try and fuse together clusters.

minlen – the minimum length in mm that C8 desires the CC to be (see "downthresh for more info")

sidesl – how far out laterally in mm to go to measure the CC cross-section – to increase robustness, C8 measures CC cross-sections in 1 mm steps from

sidesl to +sidesl (x direction). Default is 1, but 2 works well also for normal controls.

stand – specifies the number of thicknesses to measure over the whole CC.

ybox, zbox – defines an MNI cross-section box that C8 expects to find the CC inside (parts of the CC can stick outside the box, however). Strict MNI normalization is not necessary if the WM segmentation locates the CC in this area.

MNIrot – 0 means measure the CC in MNI space, whereas =1 means to rotate the CC back (splenium down, genu up) ~13 degrees in order to have, on average, the genu and splenium bottoms "standing" on a horizontal plane.

AuxFilt – If using auxiliary files whose CC location you wish to sample, then setting this to =1 tells C8 to filter all of the linear gradients out of the image before sampling

safeCentroid - =1 tells C8 to use an alternative centroid for unwrapping the CC, placing a median line within it, and measuring equally spaced things