Instituto Tecnológico y de Estudios Superiores de Monterrey

Actividad

Avance del Reto - Semana 2

BI2007B Procesamiento de imágenes médicas para el diagnóstico (Gpo 201)

Equipo #4

Maxine Annel Pacheco Ramírez	A01551933
Dacia Martínez Díaz	A01733799
Kristen Aideé Pérez Alvarez	A00829551
Allison Hernández Sánchez	A01366543
René Jahaziel Rangel Yáñez	A00826919

Asesor: Dr. José Tamez

Monterrey, NL. Abril 8, 2022

FSL

```
[MacBook-Air-de-Dacia-4:FSL_101 daciamtz$ FSLDIR=/usr/local/fsl
[MacBook-Air-de-Dacia-4:FSL_101 daciamtz$ . ${FSLDIR}/etc/fslconf/fsl.sh
[MacBook-Air-de-Dacia-4:FSL_101 daciamtz$ PATH=${FSLDIR}/bin:${PATH}
[MacBook-Air-de-Dacia-4:FSL_101 daciamtz$ export FSLDIR PATH
[MacBook-Air-de-Dacia-4:FSL_101 daciamtz$ echo $FSLDIR
/usr/local/fsl
[MacBook-Air-de-Dacia-4:FSL_101 daciamtz$ fast -B sub-101_ses-BL_anat_sub-101_ses-BL_T1w.nii.gz
[MacBook-Air-de-Dacia-4:FSL_101 daciamtz$ run_first_all -i sub-101_ses-BL_anat_sub-101_ses-BL_T1w.nii.gz -o 101_scSeg
49013
[MacBook-Air-de-Dacia-4:FSL_101 daciamtz$ bet sub-101_ses-BL_anat_sub-101_ses-BL_T1w.nii.gz 101_bet.nii.gz -m -f .4
```

Para usar los comandos de FSL primero se tienen que crear las variables de entorno en la terminal con los siguientes comandos:

```
FSLDIR=/usr/local/fsl
. ${FSLDIR}/etc/fslconf/fsl.sh
PATH=${FSLDIR}/bin:${PATH}
export FSLDIR PATH
echo $FSLDIR
```

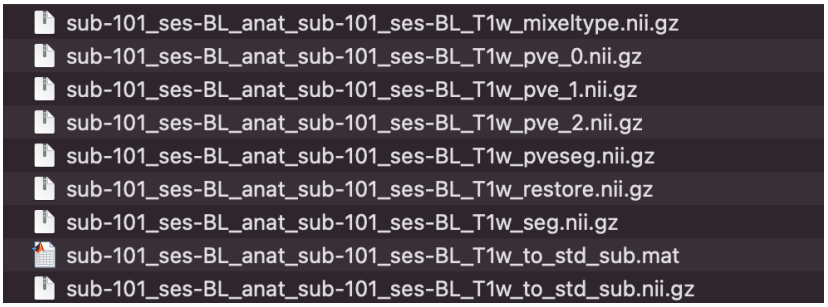
Para las segmentaciones se usó el archivo:
“sub-101_ses-BL_anat_sub-101_ses-FBL_T1w.nii.gz”

FAST (FMRIB's Automated Segmentation Tool)

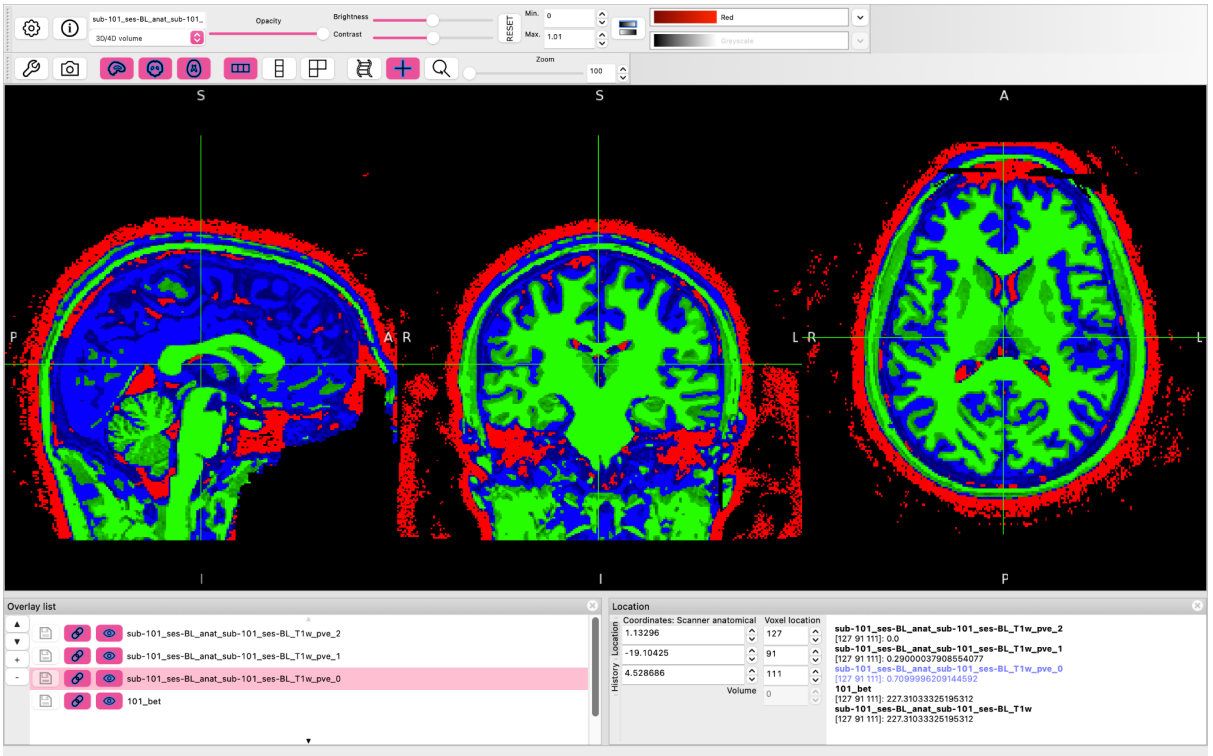
FAST segmenta una imagen 3D del cerebro en diferentes tipos de tejido (materia gris, materia blanca, CSF, etc.), al mismo tiempo que corrige las variaciones de intensidad espacial (también conocidas como campo de polarización o falta de homogeneidad de RF).

Comando:
`fast -B sub-101_ses-BL_anat_sub-101_ses-BL_T1w.nii.gz`

Output files:



Cada imagen PVE representa la estimación de volumen parcial para un tejido en particular. La numeración viene determinada por la intensidad media de cada tejido, ordenada de más oscuro a más claro. En el caso de una imagen potenciada en T1: PVE 0 = CSF; PVE 1 = Sustancia gris; PVE 2 = Sustancia Blanca.



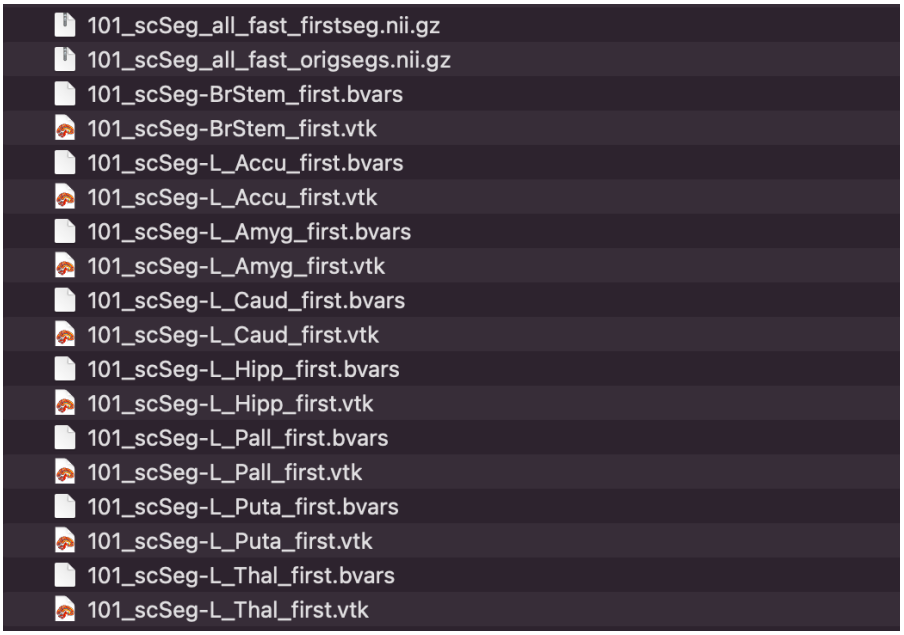
FIRST

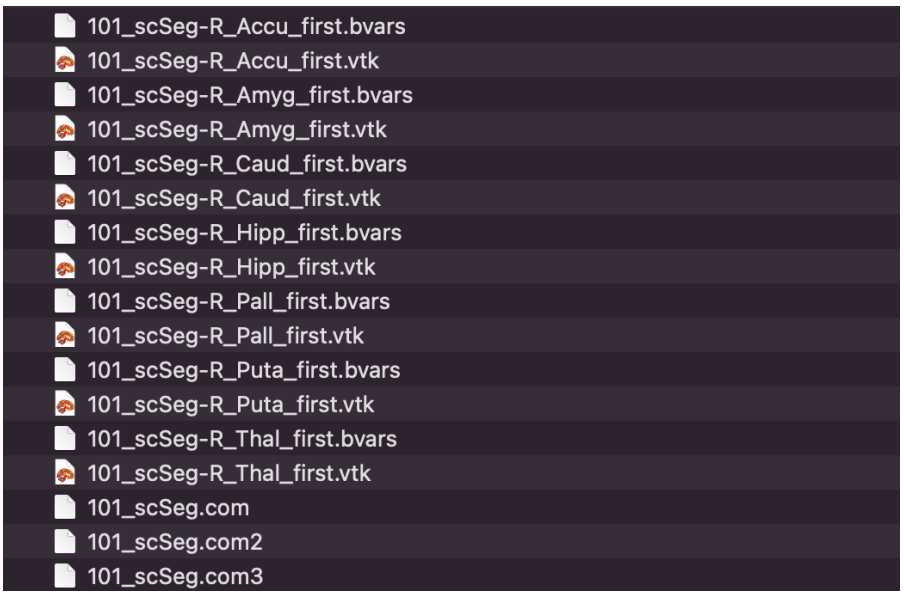
FIRST es una herramienta de segmentación/registro basada en modelos. Los modelos de forma/apariencia utilizados en FIRST se construyen a partir de imágenes segmentadas manualmente proporcionadas por el Centro de Análisis Morfométrico (CMA), MGH, Boston.

Comando:

```
run_first_all -i sub-101_ses-BL_anat_sub-101_ses-BL_T1w.nii.gz -o 101_scSeg
```

Output files:





Reference:

Zhang, Y. and Brady, M. and Smith, S. Segmentation of brain MR images through a hidden Markov random field model and the expectation-maximization algorithm. IEEE Trans Med Imag, 20(1):45-57, 2001.

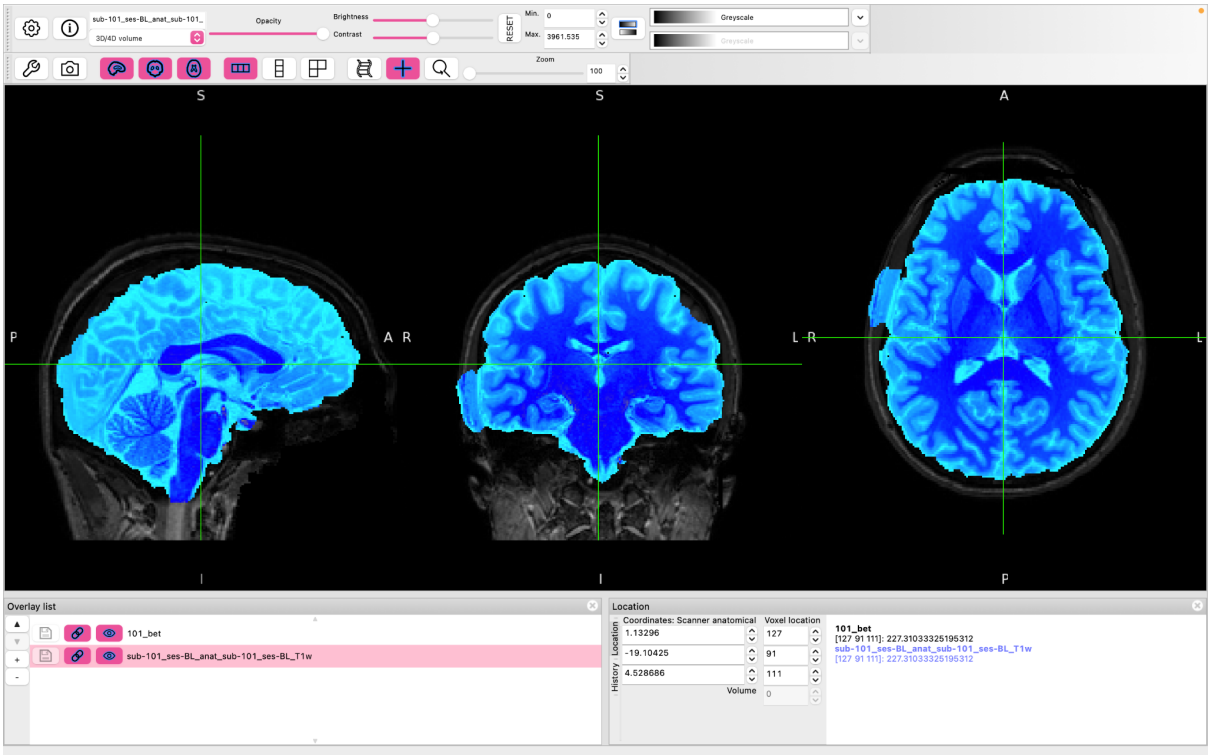
BET (Brain Extraction Tool)

Elimina tejido no cerebral de una imagen de la cabeza completa.

Comando:

```
bet sub-101_ses-BL_anat_sub-101_ses-FBL_T1w.nii.gz 101_bet.nii.gz -m -f .08
```

Output files:



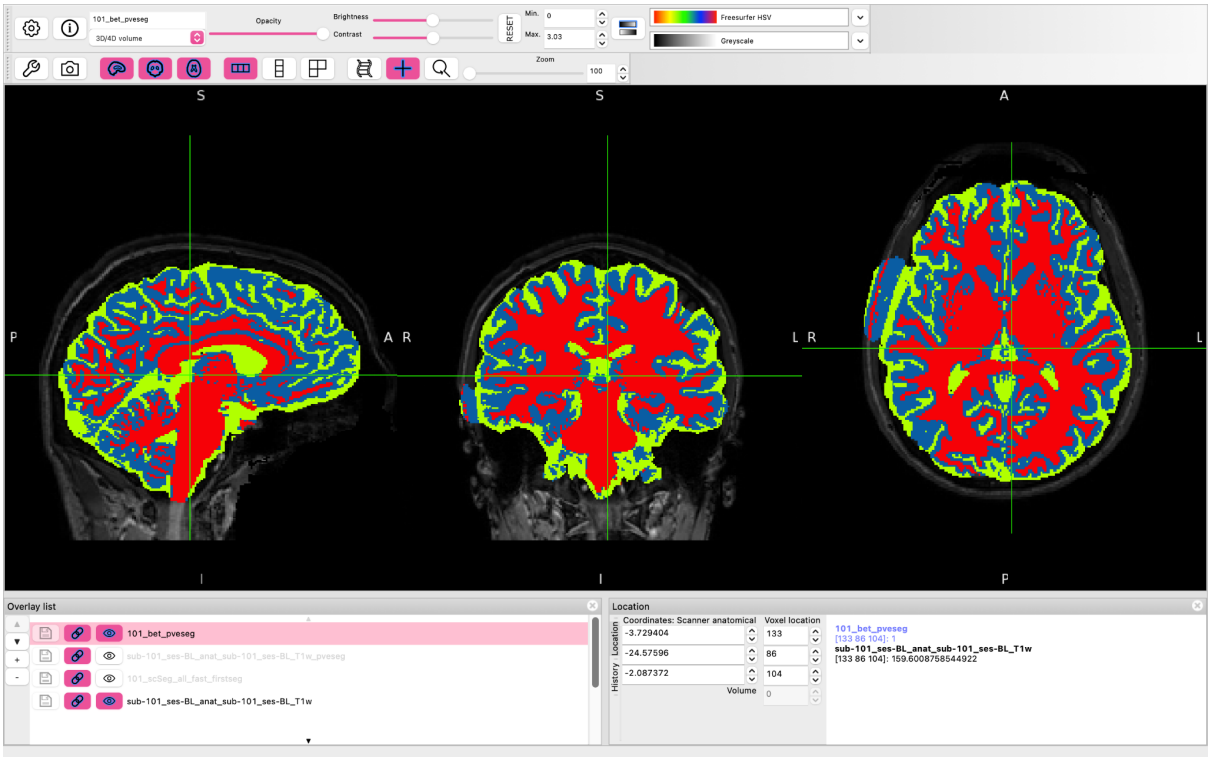
Reference:

S.M. Smith. Fast robust automated brain extraction. Human Brain Mapping, 17(3):143-155, November 2002.

Wellcome Centre for Integrative Imaging. (s.f.). *Example Box: Tissue-Type Segmentation.* https://www.fmrib.ox.ac.uk/primers/intro_primer/ExBox3/IntroBox3.html

Stanford University Wandell Lab. (2015). *Segementation-FSL-Pipeline*. https://web.stanford.edu/group/vista/cgi-bin/wiki/index.php/Segementation-FSL-Pipeline

FAST con el archivo generado de BET:



MRtrix

Se vieron videos de tutoriales para realizar la segmentación.

```
Last login: Fri Apr  8 10:58:02 on ttys000
MacBook-Air-de-Kristen:~ kristenaideeperezalvarez$ mv ~/Desktop/Segmentation
usage: mv [-f | -i | -n] [-v] source target
       mv [-f | -i | -n] [-v] source ... directory
MacBook-Air-de-Kristen:~ kristenaideeperezalvarez$ cd Desktop/Segmentation
MacBook-Air-de-Kristen:Segmentation kristenaideeperezalvarez$ cd sub-125/ses-BL/anat
MacBook-Air-de-Kristen:anat kristenaideeperezalvarez$ mrconvert sub-125_ses-BL_T1w.nii.gz sub-125_T1w.mif -
fslgrad sub-125_ses-BL_T1w.bvec sub-125_ses-BL_T1w.bval
mrconvert: [ERROR] input file "sub-125_ses-BL_T1w.bvec" for option "-fslgrad" not found
MacBook-Air-de-Kristen:anat kristenaideeperezalvarez$
```

5ttgen fsl

5ttgen actúa como un script "maestro" para generar una imagen de tejido segmentado 5TT (five-tissue-type) adecuada para uso en Tractografía Restringida Anatómicamente (ACT), haciendo uso de comandos de FSL.

Comando:

```
FSL_101 daciamtz$ 5ttgen fsl sub-101_ses-BL_anat_sub-101_ses-BL_T1w.nii.gz
5TT.mif -premasked
```

```
MacBook-Air-de-Dacia-4:FSL_101 daciamtz$ 5ttgen fsl sub-101_ses-BL_anat_sub-101_ses-BL_T1w.nii.gz
 5TT.mif -premasked
5ttgen:
5ttgen: Note that this script makes use of commands / algorithms that have relevant articles for
citation; INCLUDING FROM EXTERNAL SOFTWARE PACKAGES. Please consult the help page (-help option)
for more information.
5ttgen:
5ttgen: Generated scratch directory: /Users/daciamtz/Desktop/FSL_101/5ttgen-tmp-OAFHWI/
Command:  mrconvert /Users/daciamtz/Desktop/FSL_101/sub-101_ses-BL_anat_sub-101_ses-BL_T1w.nii.gz
 /Users/daciamtz/Desktop/FSL_101/5ttgen-tmp-OAFHWI/input.mif
5ttgen: Changing to scratch directory (/Users/daciamtz/Desktop/FSL_101/5ttgen-tmp-OAFHWI/)
Command:  mrconvert input.mif T1.nii -strides -1,+2,+3
Command:  fast T1.nii
Command:  run_first_all -m none -s L_Accu,R_Accu,L_Caud,R_Caud,L_Pall,R_Pall,L_Puta,R_Puta,L_Thal
,R_Thal -i T1.nii -o first -b
5ttgen: [100%] Generating partial volume images for SGM structures
Command:  mrmath [mesh2voxel_*.mif (10 items)] sum - | mrcalc - 1.0 -min all_sgms.mif
Command:  mrthreshold T1_pve_2.nii.gz - -abs 0.001 | maskfilter - connect - -connectivity | mrcal
c 1 - 1 -gt -sub remove_unconnected_wm_mask.mif -datatype bit
Command:  mrcalc T1_pve_0.nii.gz remove_unconnected_wm_mask.mif -mult csf.mif
Command:  mrcalc 1.0 csf.mif -sub all_sgms.mif -min sgm.mif
Command:  mrcalc 1.0 csf.mif sgm.mif -add -sub T1_pve_1.nii.gz T1_pve_2.nii.gz -add -div multipli
er.mif
Command:  mrcalc multiplier.mif -finite multiplier.mif 0.0 -if multiplier_noNAN.mif
Command:  mrcalc T1_pve_1.nii.gz multiplier_noNAN.mif -mult remove_unconnected_wm_mask.mif -mult
cgm.mif
Command:  mrcalc T1_pve_2.nii.gz multiplier_noNAN.mif -mult remove_unconnected_wm_mask.mif -mult
wm.mif
Command:  mrcalc 0 wm.mif -min path.mif
Command:  mrcat cgm.mif sgm.mif wm.mif csf.mif path.mif - -axis 3 | mrconvert - combined_precrop.
mif -strides +2,+3,+4,+1
Command:  mrmath combined_precrop.mif sum - -axis 3 | mrthreshold - - -abs 0.5 | mrgrid combined_
precrop.mif crop result.mif -mask -
Command:  mrconvert result.mif /Users/daciamtz/Desktop/FSL_101/5TT.mif
Command:  5ttcheck result.mif
5ttgen: Changing back to original directory (/Users/daciamtz/Desktop/FSL_101)
5ttgen: Deleting scratch directory (/Users/daciamtz/Desktop/FSL_101/5ttgen-tmp-OAFHWI/)
MacBook-Air-de-Dacia-4:FSL_101 daciamtz$
```

Output file:

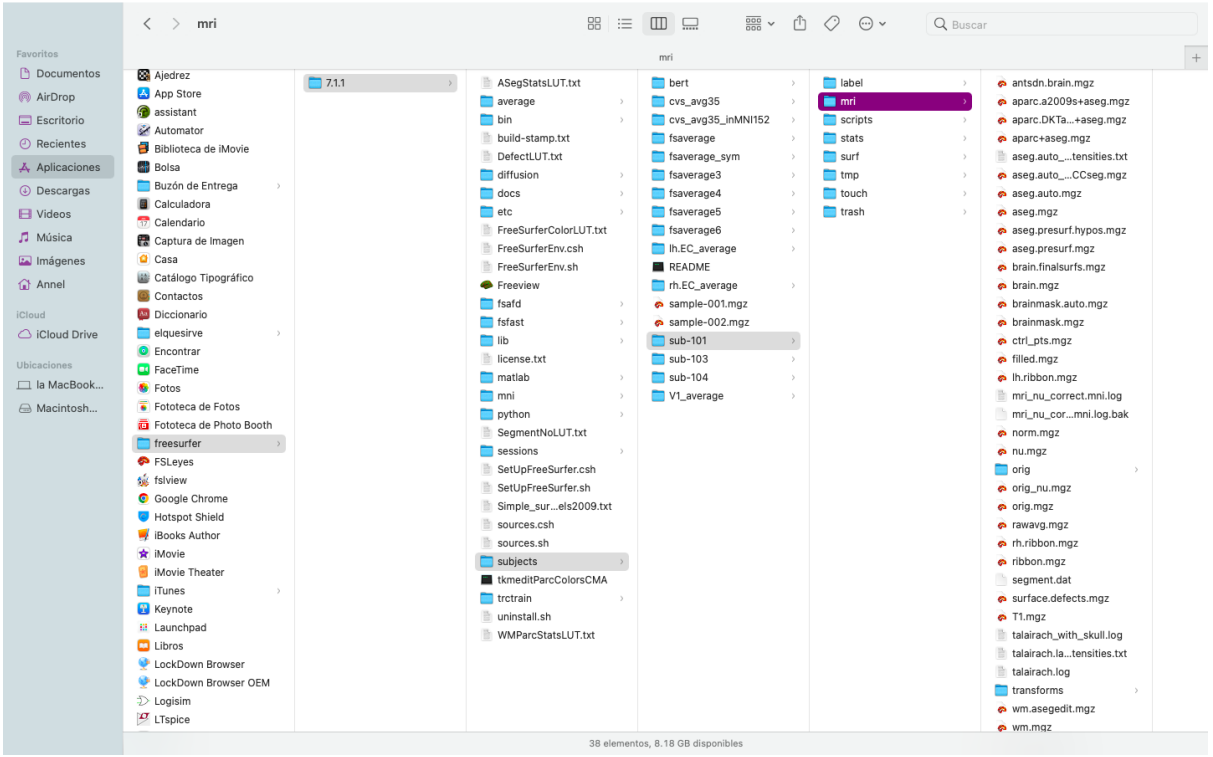
5TT.mif

Documentation:
https://mrtrix.readthedocs.io/en/dev/reference/commands/5ttgen.html

Segmentación con Freesurfer [Recon-all]

Tutorial a seguir:
https://andysbrainbook.readthedocs.io/en/latest/FreeSurfer/FS_ShortCourse/FS_03_ReconAll.html
Archivos a analizar:
https://openneuro.org/datasets/ds000174/versions/1.0.1

Se lograron obtener las carpetas con los MRI's de los tres sujetos a analizar y se colocaron dentro de la carpeta de BERT
sub-101_ses-BL_anat_sub-101_ses-BL_T1w.nii
sub-103_ses-BL_anat_sub-103_ses-BL_T1w.nii
sub-104_ses-BL_anat_sub-104_ses-BL_T1w.nii



Importar un archivo .mgz a MATLAB

Haremos uso de la siguiente función para poder importar un archivo .mgz a matlab.
https://github.com/fieldtrip/fieldtrip/blob/master/external/freesurfer/MRIread.m

```
function mri = MRIread(fstring,headeronly,permuteflag)
% mri = MRIread(fstring,headeronly,permuteflag)
```

```
%
% Reads in a volume based on the fstring. fstring can be:
%  1. A stem, in which case the format and full file name is determined
%     by finding a file on disk called fstring.ext, where ext can be
%     either mgh, mgz, img, bhdr, nii, or nii.gz
%  2. MGH file. Eg, f.mgh or f.mgz
%  3. BVolume HDR file. Eg, f.bhdr
%  4. Analyze, eg, f.img or f.hdr
%  5. NIFTI, eg, f.nii or f.nii.gz
%
% Creates a structure similar to the FreeSurfer MRI struct
% defined in mri.h. Times are in ms and angles are in radians.
% The vox2ras0 matrix is the matrix that converts a 0-based
% column, row, and slice to XYZ. vox2ras1 is the same with
% 1-based indices. The volume is rows, cols, slices frames,
% but the vox2ras expects col, row, slice.
%
% !!!! If you intend to use indices obtained from Matlab in FreeSurfer
programs
%
% bear in mind that mri.vox(j+1, i+1, k+1) = mri(i,j,k)
%
% where mri(i,j,k) refers to indices as they are seen in scuba, tkmedit
% or used in binaries such as mri_convert
%
% This happens because matlab uses row major (ie, the "fasted" dim
% goes from one row to the next), whereas C uses col major.
% So if you simply load in a matrix and view it with imagesc,
% it will appear to be transposed.
%
% Note: you can load unpermuted data with permuteflag=0. If you
% leave out this flag or set it to anything non-zero, then
% it will be permuted. The permuteflag will not affect the vox2ras
% info. If you used permuteflag=0, then use the same when
% running MRIwrite().
%
% If headeronly=1, then the pixel data are not read in.
%
% If the input is a bhdr, then mri.srcbext is set to either bshort
% or bfloat, depending upon the precision of the input. If the
% input is not bhdr, then mri.srcbext will exist but be empty.
% See also MRIwrite() and mri.outbext.
%
% If the input is NIFTI, then mri.niftihdr is the nifti header
% If the input is ANALYZE, then mri.analyzehdr is the analyze header
%



%
% MRIread.m
%
% Original Author: Doug Greve
%
```

```
% Copyright © 2011 The General Hospital Corporation (Boston, MA) "MGH"
%
% Terms and conditions for use, reproduction, distribution and contribution
% are found in the 'FreeSurfer Software License Agreement' contained
% in the file 'LICENSE' found in the FreeSurfer distribution, and here:
%
% https://surfer.nmr.mgh.harvard.edu/fswiki/FreeSurferSoftwareLicense
%
% Reporting: freesurfer@nmr.mgh.harvard.edu
%


mri = [];

if(nargin < 1 | nargin > 3)
  fprintf('mri = MRIread(fstring,headeronly,permuteflag)\n');
  return;
end
if(exist('headeronly') ~=1) headeronly = 0; end
if(exist('permuteflag')~=1) permuteflag = 1; end

[fspec fstem fmt] = MRIfspec(fstring);
if(isempty(fspec))
  err = sprintf('ERROR: cannot determine format of %s
(%s)\n',fstring,mfilename);
  error(err);
  return;
end

mri.srcbext = '';    % empty be default
mri.analyzehdr = []; % empty be default
mri.bhdr = []; % empty be default

%-------------- MGH -------------------------%
switch(fmt)
  case {'mgh','mgz'}
  [mri.vol, M, mr_parms, volsz] = load_mgh(fspec,[],[],headeronly);
  if(isempty(M))
    fprintf('ERROR: loading %s as MGH\n',fspec);
    mri = [];
    return;
  end
  if(~headeronly)
    if(permuteflag) mri.vol = permute(mri.vol,[2 1 3 4]); end
    volsz = size(mri.vol);
  else
    mri.vol = [];
    volsz(1:2) = [volsz(2) volsz(1)];
  end
  if(isempty(mr_parms)) mr_parms = zeros(4,1); end
  tr = mr_parms(1);
  flip_angle = mr_parms(2);
  te = mr_parms(3);
```

```
  ti = mr_parms(4);
%--------------- bshort/bfloat -----------------------%
 case {'bhdr'}
  if(~headeronly)
    [mri.vol bmri] = fast_ldbslice(fstem);
    if(isempty(mri.vol))
      fprintf('ERROR: loading %s as bvolume\n',fspec);
      mri = [];
      return;
    end
    volsz = size(mri.vol);
  else
    mri.vol = [];
    bmri = fast_ldbhdr(fstem);
    if(isempty(bmri))
      fprintf('ERROR: loading %s as bvolume\n',fspec);
      mri = [];
      return;
    end
    [nslices nrows ncols ntp] = fmri_bvoldim(fstem);
    volsz = [nrows ncols nslices ntp];
  end
  [nrows ncols ntp fs ns endian bext] = fmri_bfiledim(fstem);
  mri.srcbext = bext;
  M = bmri.T;
  tr = bmri.tr;
  flip_angle = bmri.flip_angle;
  te = bmri.te;
  ti = bmri.ti;
  mri.bhdr = bmri;
%------- analyze -------------------------------------
 case {'img'}
  hdr = load_analyze(fspec,headeronly);
  if(isempty(hdr))
    fprintf('ERROR: loading %s as analyze\n',fspec);
    mri = [];
    return;
  end
  volsz = hdr.dime.dim(2:end);
  indnz = find(volsz~=0);
  volsz = volsz(indnz);
  volsz = volsz(:)'; % just make sure it's a row vect
  if(~headeronly)
    if(permuteflag) mri.vol = permute(hdr.vol,[2 1 3 4]); end
  else
    mri.vol = [];
  end
  volsz([1 2]) = volsz([2 1]); % Make consistent. No effect when rows=cols
  tr = 1000*hdr.dime.pixdim(5); % msec
  flip_angle = 0;
  te = 0;
  ti = 0;
  hdr.vol = []; % already have it above, so clear it
```

```
  M = vox2ras_1to0(hdr.vox2ras);
  mri.analyzehdr = hdr;
%------- nifti nii -------------------------------------
 case {'nii','nii.gz'}
  hdr = load_nifti(fspec,headeronly);
  if(isempty(hdr))
    fprintf('ERROR: loading %s as analyze\n',fspec);
    mri = [];
    return;
  end
  volsz = hdr.dim(2:end);
  indnz = find(volsz~=0);
  volsz = volsz(indnz);
  volsz = volsz(:)'; % just make sure it's a row vect
  % This handles the case where data has > 4 dims
  % Just puts all data into dim 4.
  if(~headeronly)
    hdr.vol = reshape(hdr.vol,[volsz(1) volsz(2) volsz(3)
prod(volsz(4:end))]);
    if(permuteflag)
      mri.vol = permute(hdr.vol,[2 1 3 4]);
    else
      mri.vol = hdr.vol;
    end
  else mri.vol = [];
  end
  volsz([1 2]) = volsz([2 1]); % Make consistent. No effect when rows=cols
  tr = hdr.pixdim(5); % already msec
  flip_angle = 0;
  te = 0;
  ti = 0;
  hdr.vol = []; % already have it above, so clear it
  M = hdr.vox2ras;
  mri.niftihdr = hdr;
%-----------------------------------------------------
 otherwise
  fprintf('ERROR: format %s not supported\n',fmt);
  mri = [];
  return;
end
%--------------------------------------%

mri.fspec = fspec;
mri.pwd = pwd;

mri.flip_angle = flip_angle;
mri.tr  = tr;
mri.te  = te;
mri.ti  = ti;

% Assumes indices are 0-based. See vox2ras1 below for 1-based.  Note:
% MRIwrite() derives all geometry information (ie, direction cosines,
% voxel resolution, and P0 from vox2ras0. If you change other geometry
```

```
% elements of the structure, it will not be reflected in the output
% volume. Also note that vox2ras still maps Col-Row-Slice and not
% Row-Col-Slice.  Make sure to take this into account when indexing
% into matlab volumes (which are RCS).
mri.vox2ras0 = M;

% Dimensions not redundant when using header only
volsz(length(volsz)+1:4) = 1; % Make sure all dims are represented
mri.volsize = volsz(1:3); % only spatial components
mri.height  = volsz(1);   % Note that height (rows) is the first dimension
mri.width   = volsz(2);   % Note that width (cols) is the second dimension
mri.depth   = volsz(3);
mri.nframes = volsz(4);

%-------------------------------------------------------------------%
% Everything below is redundant in that they can be derivied from
% stuff above, but they are in the MRI struct defined in mri.h, so I
% thought I would add them here for completeness.  Note: MRIwrite()
% derives all geometry information (ie, direction cosines, voxel
% resolution, and P0 from vox2ras0. If you change other geometry
% elements below, it will not be reflected in the output volume.

mri.vox2ras = mri.vox2ras0;

mri.nvoxels = mri.height * mri.width * mri.depth; % number of spatial voxles
mri.xsize = sqrt(sum(M(:,1).^2)); % Col
mri.ysize = sqrt(sum(M(:,2).^2)); % Row
mri.zsize = sqrt(sum(M(:,3).^2)); % Slice

mri.x_r = M(1,1)/mri.xsize; % Col
mri.x_a = M(2,1)/mri.xsize;
mri.x_s = M(3,1)/mri.xsize;

mri.y_r = M(1,2)/mri.ysize; % Row
mri.y_a = M(2,2)/mri.ysize;
mri.y_s = M(3,2)/mri.ysize;

mri.z_r = M(1,3)/mri.zsize; % Slice
mri.z_a = M(2,3)/mri.zsize;
mri.z_s = M(3,3)/mri.zsize;

ic = [(mri.width)/2 (mri.height)/2 (mri.depth)/2 1]';
c = M*ic;
mri.c_r = c(1);
mri.c_a = c(2);
mri.c_s = c(3);
%---------------------------------------------------%

%-------- The stuff here is for convenience --------------

% 1-based vox2ras. Good for doing transforms in matlab
mri.vox2ras1 = vox2ras_0to1(M);
```

```
% Matrix of direction cosines
mri.Mdc = [M(1:3,1)/mri.xsize M(1:3,2)/mri.ysize M(1:3,3)/mri.zsize];

% Vector of voxel resolutions (Row-Col-Slice)
mri.volres = [mri.ysize mri.xsize mri.zsize];

% Have to swap rows and columns back
voldim = [mri.volsize(2) mri.volsize(1) mri.volsize(3)]; %[ncols nrows
nslices]
volres = [mri.volres(2)  mri.volres(1)  mri.volres(3)];  %[dcol drow dslice]
mri.tkrvox2ras = vox2ras_tkreg(voldim,volres);


return;
```