fmriprep

Alternativa: CPAC.

El objetivo final sera utilizar algoritmos de analisis simples para extraer la conectividad en diferentes redes de las imagenes fMRI. Ahora bien, para ello debemos tener las imagenes preprocesadas, con todas las correcciones *standard*. Lo mas sencillo es usar un soft que hayan echo otros. Voy a intentarlo con fmriprep.

Con Singularity

En esta pagina recomiendan usar la versión empaquetada en docker pero he tenido varios problemas (el HPC no va bien) y no me termina de funcionar. Afortunadamente puede convertirse la imagen a un container de Singularity. Asi que he convertido el container y he logrado que funcione de alguna manera.

Construyendo el container, (**No hacer esto en detritus, que no hay espacio**)

```
[root@brick01 singularity]# export SINGULARITY_CACHEDIR=/nas/cache
[root@brick01 singularity]# singularity build
/nas/usr/local/opt/bin/fmriprep-latest.simg
docker://poldracklab/fmriprep:1.5.9
Building into existing container: /nas/usr/local/opt/bin/fmriprep-
latest.simg
Docker image path: index.docker.io/poldracklab/fmriprep:1.5.9
Cache folder set to /home/cache/docker
Importing: base Singularity environment
Exploding layer:
sha256:c832082614738021cf51d68e84457a45a3f85dbe0ddbb93000c5dc5aae77fe93.tar.
gz
Exploding layer:
sha256:6e1a85c1d66a18674bdb5cee0a7d6b294aa4f1921d3a13074004d00485868051.tar.
gz
Exploding layer:
sha256:f1320ef45e201689247f36b8e037ed8de21532f952be224586cb96b43a495f36.tar.
gz
Exploding layer:
sha256:5a6ab6e6fbf62623bce0db5c75f3775f287c7b86cd0152b01ab09275f1d9f540.tar.
gz
Exploding layer:
sha256:6fd240c277673683f0b16a0d7903c036c307954976e19e2f82a6873327d1d69b.tar.
gz
Exploding layer:
sha256:4910036a866945ed7db59ab8db389a7e785a0815c9afe59a1b5626172c0b86be.tar.
gz
Exploding layer:
sha256:552902834964a2b08552128518d10859785fee27b3f8c0e26b53ee856385d096.tar.
```

```
gz
Exploding layer:
sha256:571048daac396c1f5f3b2b9a12089dedea0d7ec9553c67bf3a6e5a1b19c6888c.tar.
gz
Exploding layer:
sha256:1756b6b7c22209f53ff961e0e2696491042f5f36193f6a349c6abc154da72049.tar.
gz
Exploding layer:
sha256:65aa5a2ca8f5712cfe1f4df1f8f2da8f97b12f7c33068bb8eb015f13ade5c522.tar.
gz
Exploding layer:
sha256:143350156437caa631672b3524e7dd4964470cb737f350cf3efe8692fc5989ad.tar.
gz
Exploding layer:
sha256:02a71d7211a274b3a7165d910334bfdce42f85e5a5fb1df1152d32a3cbbc3a50.tar.
gz
Exploding layer:
sha256:cc948187fffce024a2df30038319901da4e6e9956ae618030f7901218f4c19a4.tar.
gz
Exploding layer:
sha256:307e84f37ee94c46357c458c09d6df57df5fff4aaf1be74fbcb601b4a450a5f8.tar.
gz
Exploding layer:
sha256:7f2993d16ccc01a5551dcf3932a89157c1c8d9d0647a1d59fb7706668ff8cec5.tar.
gz
Exploding layer:
sha256:56ed4f8babc15c9779457865603ea70fc3e5f109f17de42625ada1c1fb67630d.tar.
gz
Exploding layer:
sha256:2af115090e565ca47aaa9fbca37d8a442ad41ba0effc8ec1e20bb442d01667ca.tar.
gz
Exploding layer:
sha256:104185b69d0203aa1751588349db7b79a7f14bd5e67ccee8502f5a05fdf0f991.tar.
gz
Exploding layer:
sha256:37434e18a86a3e265f5ce7152ac1da0ddaeb5ec8ad96daaac39b61220357e9a5.tar.
gz
Exploding layer:
sha256:49b8c1f8e6403aa68daf6d580a5f73a6f3e182f3ee774480c456ecaf0152d1da.tar.
gz
Exploding layer:
sha256:0c8d27c764ae53a6c2a1990c6d79981716b841f8ddd4979a6ea024c983f1ae03.tar.
gz
Exploding layer:
sha256:d7e8af36992ff0a13fd804760b414c6fc1ee1fa9bc7a9fefb3380a0dca84114c.tar.
gz
Exploding layer:
sha256:8f2beb8a17f1a38d2ecf836857a18e07cc6f8da373632cd599c96986d0083859.tar.
gz
Exploding layer:
sha256:f7e7732dbf02ecd8abd1a31ba8739cc477b418baf65f43066409408d8f82ed82.tar.
```

```
gz
Exploding layer:
sha256:e317a86a0fd4e465a91100b548c2616b50903ace77eac30f692f406e7a64ca40.tar.
gz
Exploding layer:
sha256:5e088dc61c8ef5e49411fb25f519679c8707753af576e5cdf7236bb09338130f.tar.
gz
Exploding layer:
sha256:bf93bcca5bc9ea661b8929ba47d6c615e0c63dd69459c498fee7aad33f086420.tar.
gz
Exploding layer:
sha256:b47efeea289424acbae0b32f2950b8616f268546e943cc2f40b5861b320a3fc9.tar.
gz
Exploding layer:
sha256:10960f811026f82718ac9905a442d3ec1b06c3591bc661ba9adab336cf4d2a02.tar.
gz
Building Singularity image...
Singularity container built: /nas/usr/local/opt/bin/fmriprep-latest.simg
Cleaning up...
[root@brick01 singularity]# ls -lh /nas/usr/local/opt/bin/fmriprep-
latest.simg
-rwxr-xr-x 1 root root 4.5G Feb 26 10:17 /nas/usr/local/opt/bin/fmriprep-
latest.simg
```

El primer paso es convertir las imagenes a formato BIDS. Luego ha de procesarse cada sujeto por separado asi,

```
[osotolongo@brick02 ~]$ singularity run --cleanenv -B /nas:/nas
/nas/software/fmriprep-latest.simg /nas/data/mopead/bids
/nas/data/mopead/fmriprep participant --participant-label "sub-0001" --
ignore slicetiming --skip_bids_validation --fs-license-file
/nas/usr/local/opt/freesurfer/.license
```

Nota: Esto es eterno, asi que debe enviarse en background cuando este OK. De momento estoy probando un caso, despues sera necesario integrarlo al cluster de alguna manera pero con cuidado porque consume muchisimo.

```
top - 11:35:25 up 209 days, 23:39,  2 users,  load average: 5.48, 5.77, 4.43
Tasks: 663 total,   3 running, 660 sleeping,   0 stopped,   0 zombie
%Cpu(s):  9.0 us,  0.0 sy,  0.0 ni, 91.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0
st
KiB Mem : 26372729+total,  6548484 free,  9591132 used, 24758768+buff/cache
KiB Swap:        0 total,        0 free,        0 used. 25275507+avail Mem

   PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
126411 osotolo+  20   0 2396520 1.698g  19412 R 571.9  0.7  86:27.98
antsRegistratio
 88479 osotolo+  20   0  938292 199044  33236 S   6.3  0.1   1:12.70
fmriprep
......
```

```
......
```

Usando SLURM

El proceso pareceiniciarsin problemas pero deora bastantte, asi que voy a intentar lanzarlos con el *schedule manager*. El primer paso en instalar Singularity en todos los nodos. Luego voy a hacer un script de prueba,

```
[osotolongo@detritus mopead]$ cat test.sh
#!/bin/bash
#SBATCH -J fmri_prep
#SBATCH --time=72:0:0
#SBATCH -o /nas/data/mopead/slurm/test_fmriprep-%j
#SBATCH -c 8
#SBATCH --mail-user=osotolongo
srun singularity run --cleanenv -B /nas:/nas /nas/software/fmriprep-
latest.simg /nas/data/mopead/bids /nas/data/mopead/fmriprep participant --
participant-label "sub-0001" --ignore slicetiming --skip_bids_validation --
fs-license-file /nas/usr/local/opt/freesurfer/.license
```

y lo lanzo normalmente

```
[osotolongo@detritus mopead]$ sbatch test.sh
[osotolongo@detritus mopead]$ squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES
NODELIST(REASON)
......
             16610     devel fmri_pre osotolon  R       8:02      1 brick01
```

Ademas, se puede ir controlando el avance en el archivo de output y recibire los emails en caso de fallo o finalizacion.

Nota: *fmriprep* es bastante iinestable, a veces falla, las tareas se completan en tiempos disimiles, puede no devolver correctamente la salida. De cualquier manera tenemos una herramienta que puede usarse en el cluster con un poco de *tunning*.

Ya sabemos suficiente para hacer un wrapper

cfmriprep.pl

```
#!/usr/bin/perl
# Copyright 2019 O. Sotolongo <asqwerty@gmail.com>
use strict; use warnings;

use File::Find::Rule;
use NEURO qw(print_help get_pair load_study achtung shit_done get_lut
check_or_make centiloid_fbb);
use Data::Dump qw(dump);
use File::Remove 'remove';
```

```
use File::Basename qw(basename);

my $fmri_prep_img = '/nas/software/fmriprep-latest.simg';
my $nas_dir = '/nas';
my $fslic = '/nas/usr/local/opt/freesurfer/.license';
my $cfile;

@ARGV = ("-h") unless @ARGV;

while (@ARGV and $ARGV[0] =~ /^-/) {
    $_ = shift;
    last if /^--$/;
    if (/^-cut/) { $cfile = shift; chomp($cfile);}
    if (/^-h$/) { print_help $ENV{'PIPEDIR'}.'/doc/cfmriprep.hlp';
exit;}
}
my $study = shift;
unless ($study) { print_help $ENV{'PIPEDIR'}.'/doc/cfmriprep.hlp';
exit;}
my %std = load_study($study);
my $w_dir = $std{'WORKING'};
my $data_dir = $std{'DATA'};
my $bids_dir = $data_dir.'/bids';
my $fmriout_dir = $data_dir.'/fmriprep_out';
check_or_make($fmriout_dir);
my $outdir = "$std{'DATA'}/slurm";
check_or_make($outdir);

my @subjects;
if($cfile){
    open DBF, $cfile or die "No such file\n";
    while(<DBF>) {
        chomp;
        push @subjects, $_;
    }
    close DBF;
}else{
    opendir DBD, $bids_dir or die "Cold not open dir\n";
    while (my $thing = readdir DBD){
        if ($thing eq '.' or $thing eq '..') {
            next;
        }
        if ($thing =~ /sub-*/) {
            push @subjects, $thing;
        }
    }
    closedir DBD;
}
foreach my $subject (@subjects) {
    my $orderfile = $outdir.'/'.$subject.'_fmriprep.sh';
    open ORD, ">$orderfile";
```

```
    print ORD '#!/bin/bash'."\n";
    print ORD '#SBATCH -J fmriprep_'.$study."\n";
    print ORD '#SBATCH --time=120:0:0'."\n"; #si no ha terminado en X
horas matalo
    print ORD '#SBATCH --mail-type=FAIL,TIME_LIMIT,STAGE_OUT'."\n"; #no
quieres que te mande email de todo
    print ORD '#SBATCH -o '.$outdir.'/fmriprep-%j'."\n";
    print ORD '#SBATCH -c 20'."\n";
    print ORD '#SBATCH -p fast'."\n";
    print ORD '#SBATCH --mail-user='."$ENV{'USER'}\n";
    print ORD 'srun singularity run --cleanenv -B
'.$nas_dir.':'.$nas_dir.' '.$fmri_prep_img.' '.$bids_dir.'
'.$fmriout_dir.' participant --participant-label "'.$subject.'" --
ignore slicetiming --skip_bids_validation --fs-license-file
'.$fslic."\n";
    close ORD;
    system("sbatch $orderfile");
    sleep(20);
}
my $orderfile = $outdir.'/fmriprep_end.sh';
open ORD, ">$orderfile";
print ORD '#!/bin/bash'."\n";
print ORD '#SBATCH -J fmriprep_'.$study."\n";
print ORD '#SBATCH --mail-type=END'."\n"; #email cuando termine
print ORD '#SBATCH --mail-user='."$ENV{'USER'}\n";
print ORD '#SBATCH -p fast'."\n";
print ORD '#SBATCH -o '.$outdir.'/fmriprep_end-%j'."\n";
print ORD ":\n";
close ORD;
my $order = 'sbatch --dependency=singleton '.$orderfile;
exec($order);
```

Esto basta para ejecutar las imagenes de singularity en el cluster pero hay un inconveniente, fmriprep es muy **inestable** falla continuamente y con errores tontos. Para correr varias veces el programa se han de limpiar (o mover) todos los archivos que crea puede dar erroresporque no encuentra los archivos. He añadido 20 segundos entre cada launch para intentar evitar algun solapamiento almontar la particion de trabajo. Esto hace algo mas lenta la ejecucion, pero no demasiado.

```
[osotolongo@detritus mopead]$ cfmriprep.pl mopead
Submitted batch job 16875
Submitted batch job 16876
Submitted batch job 16877
Submitted batch job 16878
Submitted batch job 16879
Submitted batch job 16880
Submitted batch job 16881
Submitted batch job 16882
Submitted batch job 16883
Submitted batch job 16884
```

```
Submitted batch job 16885
Submitted batch job 16886
Submitted batch job 16887
Submitted batch job 16888
Submitted batch job 16889
Submitted batch job 16890
Submitted batch job 16891
Submitted batch job 16892
Submitted batch job 16893
Submitted batch job 16894
Submitted batch job 16895
Submitted batch job 16896
Submitted batch job 16897
Submitted batch job 16898
[osotolongo@detritus mopead]$ squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES
NODELIST(REASON)
             16883      fast fmriprep osotolon PD       0:00      1
(Resources)
             16884      fast fmriprep osotolon PD       0:00      1
(Priority)
             16885      fast fmriprep osotolon PD       0:00      1
(Priority)
             16886      fast fmriprep osotolon PD       0:00      1
(Priority)
             16887      fast fmriprep osotolon PD       0:00      1
(Priority)
             16888      fast fmriprep osotolon PD       0:00      1
(Priority)
             16889      fast fmriprep osotolon PD       0:00      1
(Priority)
             16890      fast fmriprep osotolon PD       0:00      1
(Priority)
             16891      fast fmriprep osotolon PD       0:00      1
(Priority)
             16892      fast fmriprep osotolon PD       0:00      1
(Priority)
             16893      fast fmriprep osotolon PD       0:00      1
(Priority)
             16894      fast fmriprep osotolon PD       0:00      1
(Priority)
             16895      fast fmriprep osotolon PD       0:00      1
(Priority)
             16896      fast fmriprep osotolon PD       0:00      1
(Priority)
             16897      fast fmriprep osotolon PD       0:00      1
(Priority)
             16898      fast fmriprep osotolon PD       0:00      1
(Dependency)
             16875      fast fmriprep osotolon  R       9:29      1 brick03
             16876      fast fmriprep osotolon  R       9:09      1 brick03
```

```
             16877      fast fmriprep osotolon  R       8:49      1 brick01
             16878      fast fmriprep osotolon  R       8:29      1 brick01
             16879      fast fmriprep osotolon  R       8:09      1 brick01
             16880      fast fmriprep osotolon  R       7:49      1 brick02
             16881      fast fmriprep osotolon  R       7:29      1 brick02
             16882      fast fmriprep osotolon  R       7:09      1 brick02
```

Resultados

De 23 sujetos procesados han terminado con exito 10.

```
[osotolongo@detritus mopead]$ find fmriprep_out/fmriprep/ -name
"*preproc_bold.nii.gz"
fmriprep_out/fmriprep/sub-0002/func/sub-0002_task-rest_run-1_space-
MNI152NLin2009cAsym_desc-preproc_bold.nii.gz
fmriprep_out/fmriprep/sub-0003/func/sub-0003_task-rest_run-1_space-
MNI152NLin2009cAsym_desc-preproc_bold.nii.gz
fmriprep_out/fmriprep/sub-0007/func/sub-0007_task-rest_run-1_space-
MNI152NLin2009cAsym_desc-preproc_bold.nii.gz
fmriprep_out/fmriprep/sub-0008/func/sub-0008_task-rest_run-1_space-
MNI152NLin2009cAsym_desc-preproc_bold.nii.gz
fmriprep_out/fmriprep/sub-0009/func/sub-0009_task-rest_run-1_space-
MNI152NLin2009cAsym_desc-preproc_bold.nii.gz
fmriprep_out/fmriprep/sub-0012/func/sub-0012_task-rest_run-1_space-
MNI152NLin2009cAsym_desc-preproc_bold.nii.gz
fmriprep_out/fmriprep/sub-0014/func/sub-0014_task-rest_run-1_space-
MNI152NLin2009cAsym_desc-preproc_bold.nii.gz
fmriprep_out/fmriprep/sub-0018/func/sub-0018_task-rest_run-1_space-
MNI152NLin2009cAsym_desc-preproc_bold.nii.gz
fmriprep_out/fmriprep/sub-0020/func/sub-0020_task-rest_run-1_space-
MNI152NLin2009cAsym_desc-preproc_bold.nii.gz
fmriprep_out/fmriprep/sub-0023/func/sub-0023_task-rest_run-1_space-
MNI152NLin2009cAsym_desc-preproc_bold.nii.gz
```

Podemos consultar los errores en los archivos de output guardados en el directorio *slurm*.

Algunos sujetos no tienen imagenes *BOLD*,

```
Exception: No BOLD images found for participant 0016 and task <all>. All
workflows require BOLD images.
srun: error: brick01: task 0: Exited with exit code 1
```

Hay errores de procesamiento de freesurfer pero lo raro es que estos archivos ha sido procesados con freesurfer satisfactoriamente en nuestra pipeline.

```
[osotolongo@detritus mopead]$ grep -i error slurm/fmriprep-16875
         [Node] Error on "_autorecon30"
(/nas/data/mopead/work/fmriprep_wf/single_subject_0001_wf/anat_preproc_wf/su
rface_recon_wf/autorecon_resume_wf/autorecon3/mapflow/_autorecon30)
```

```
190515-12:17:20,62 nipype.workflow ERROR:
190515-12:17:20,250 nipype.workflow ERROR:
RuntimeError: Command:
recon-all -s sub-0001 exited with ERRORS at Wed May 15 12:17:19 UTC 2019
Standard error:
Preprocessing did not finish successfully. Errors occurred while processing
data from participants: 0001 (1). Check the HTML reports for details.
    raise RuntimeError("".join(result['traceback']))
RuntimeError: Traceback (most recent call last):
RuntimeError: Command:
recon-all -s sub-0001 exited with ERRORS at Wed May 15 12:17:19 UTC 2019
Standard error:
/usr/local/miniconda/lib/python3.7/site-packages/scipy/fftpack/basic.py:160:
FutureWarning: Using a non-tuple sequence for multidimensional indexing is
deprecated; use `arr[tuple(seq)]` instead of `arr[seq]`. In the future this
will be interpreted as an array index, `arr[np.array(seq)]`, which will
result either in an error or a different result.
srun: error: brick03: task 0: Exited with exit code 1
```

La conclusion, de momento, es que este software es demasiado inestable tal y como esta empaquetado para usarse satisfactoriamente

Con Docker

Tras dar muchas vueltas he conseguido echar a andar docker en las maquinas de la particion *fast*. Esto deberia bastar para procesar todo si tengo cuidado con lo que hago. Basicamente tod es muy parecido asi que me basare en el trabajo hecho anteriormente.

Voy asuponer que tengo todo en BIDS. La orden para procesar un solo sujeto es,

```
fmriprep-docker /nas/data/mopead/bids/ /nas/data/mopead/fmriprep --
participant-label "sub-0003" --ignore slicetiming --skip_bids_validation --
fs-license-file /nas/usr/local/opt/freesurfer/.license
```

Nota: Esto lo he hecho en *brick03* pues no puedo hacerlo en *detritus*.

El resultado,

```
[osotolongo@brick03 mopead]$ ls fmriprep/fmriprep/sub-0003/func/
sub-0003_task-rest_run-1_desc-confounds_regressors.json
sub-0003_task-rest_run-1_space-MNI152NLin2009cAsym_desc-aseg_dseg.nii.gz
sub-0003_task-rest_run-1_desc-confounds_regressors.tsv
sub-0003_task-rest_run-1_space-MNI152NLin2009cAsym_desc-brain_mask.json
sub-0003_task-rest_run-1_space-fsaverage5_hemi-L.func.gii
sub-0003_task-rest_run-1_space-MNI152NLin2009cAsym_desc-brain_mask.nii.gz
sub-0003_task-rest_run-1_space-fsaverage5_hemi-R.func.gii
sub-0003_task-rest_run-1_space-MNI152NLin2009cAsym_desc-preproc_bold.json
sub-0003_task-rest_run-1_space-MNI152NLin2009cAsym_boldref.nii.gz
sub-0003_task-rest_run-1_space-MNI152NLin2009cAsym_desc-preproc_bold.nii.gz
```

```
sub-0003_task-rest_run-1_space-MNI152NLin2009cAsym_desc-
aparcaseg_dseg.nii.gz
[osotolongo@brick03 mopead]$ ls fmriprep/fmriprep/sub-0003/anat/
sub-0003_desc-aparcaseg_dseg.nii.gz                          sub-0003_from-
T1w_to-MNI152NLin2009cAsym_mode-image_xfm.h5  sub-0003_label-
GM_probseg.nii.gz
sub-0003_desc-aseg_dseg.nii.gz                               sub-0003_hemi-
L_inflated.surf.gii                           sub-0003_label-
WM_probseg.nii.gz
sub-0003_desc-brain_mask.json                                sub-0003_hemi-
L_midthickness.surf.gii                       sub-0003_space-
MNI152NLin2009cAsym_desc-brain_mask.json
sub-0003_desc-brain_mask.nii.gz                              sub-0003_hemi-
L_pial.surf.gii                               sub-0003_space-
MNI152NLin2009cAsym_desc-brain_mask.nii.gz
sub-0003_desc-preproc_T1w.json                               sub-0003_hemi-
L_smoothwm.surf.gii                           sub-0003_space-
MNI152NLin2009cAsym_desc-preproc_T1w.json
sub-0003_desc-preproc_T1w.nii.gz                             sub-0003_hemi-
R_inflated.surf.gii                           sub-0003_space-
MNI152NLin2009cAsym_desc-preproc_T1w.nii.gz
sub-0003_dseg.nii.gz                                         sub-0003_hemi-
R_midthickness.surf.gii                       sub-0003_space-
MNI152NLin2009cAsym_dseg.nii.gz
sub-0003_from-MNI152NLin2009cAsym_to-T1w_mode-image_xfm.h5  sub-0003_hemi-
R_pial.surf.gii                               sub-0003_space-
MNI152NLin2009cAsym_label-CSF_probseg.nii.gz
sub-0003_from-orig_to-T1w_mode-image_xfm.txt                 sub-0003_hemi-
R_smoothwm.surf.gii                           sub-0003_space-
MNI152NLin2009cAsym_label-GM_probseg.nii.gz
sub-0003_from-T1w_to-fsnative_mode-image_xfm.txt             sub-0003_label-
CSF_probseg.nii.gz
```

He de intentar ahora organizarlo en el cluster

Usando SLURM

Esto es tan parecido que hay que hacer cambios minimos,

dfmriprep.pl

```
#!/usr/bin/perl
# Copyright 2019 O. Sotolongo <asqwerty@gmail.com>
use strict; use warnings;

use File::Find::Rule;
use NEURO qw(print_help get_pair load_study achtung shit_done get_lut
check_or_make centiloid_fbb);
use Data::Dump qw(dump);
```

```
use File::Remove 'remove';
use File::Basename qw(basename);

my $fslic = '/nas/usr/local/opt/freesurfer/.license';
my $cfile;

@ARGV = ("-h") unless @ARGV;

while (@ARGV and $ARGV[0] =~ /^-/) {
    $_ = shift;
    last if /^--$/;
    if (/^-cut/) { $cfile = shift; chomp($cfile);}
    if (/^-h$/) { print_help $ENV{'PIPEDIR'}.'/doc/cfmriprep.hlp';
exit;}
}
my $study = shift;
unless ($study) { print_help $ENV{'PIPEDIR'}.'/doc/cfmriprep.hlp';
exit;}
my %std = load_study($study);
my $w_dir = $std{'WORKING'};
my $data_dir = $std{'DATA'};
my $bids_dir = $data_dir.'/bids';
my $fmriout_dir = $data_dir.'/fmriprep_out';
check_or_make($fmriout_dir);
my $outdir = "$std{'DATA'}/slurm";
check_or_make($outdir);

my @subjects;
if($cfile){
        open DBF, $cfile or die "No such file\n";
        while(<DBF>) {
                chomp;
                push @subjects, $_;
        }
        close DBF;
}else{
        opendir DBD, $bids_dir or die "Cold not open dir\n";
        while (my $thing = readdir DBD){
                if ($thing eq '.' or $thing eq '..') {
                        next;
                }
                if ($thing =~ /sub-*/) {
                        push @subjects, $thing;
                }
        }
        closedir DBD;
}
foreach my $subject (@subjects) {
        my $orderfile = $outdir.'/'.$subject.'_fmriprep.sh';
        open ORD, ">$orderfile";
        print ORD '#!/bin/bash'."\n";
```

```
        print ORD '#SBATCH -J fmriprep_'.$study."\n";
        print ORD '#SBATCH --time=72:0:0'."\n"; #si no ha terminado en
X horas matalo
        print ORD '#SBATCH --mail-type=FAIL,TIME_LIMIT,STAGE_OUT'."\n";
#no quieres que te mande email de todo
        print ORD '#SBATCH -o '.$outdir.'/fmriprep-%j'."\n";
        print ORD '#SBATCH -c 20'."\n";
        print ORD '#SBATCH -p fast'."\n";
        print ORD '#SBATCH --mail-user='."$ENV{'USER'}\n";
        print ORD 'srun docker run --rm -v
'.$fslic.':/opt/freesurfer/license.txt:ro -v '.$bids_dir.':/data:ro -v
'.$fmriout_dir.':/out poldracklab/fmriprep:1.5.0 /data /out participant
--participant-label '.$subject.' --ignore slicetiming --
skip_bids_validation'."\n";
        #print ORD 'srun fmriprep-docker '.$bids_dir.' '.$fmriout_dir.'
--participant-label "'.$subject.'" --ignore slicetiming --
skip_bids_validation --fs-license-file '.$fslic."\n";
        close ORD;
        system("sbatch $orderfile");
        sleep(20);
}
my $orderfile = $outdir.'/fmriprep_end.sh';
open ORD, ">$orderfile";
print ORD '#!/bin/bash'."\n";
print ORD '#SBATCH -J fmriprep_'.$study."\n";
print ORD '#SBATCH --mail-type=END'."\n"; #email cuando termine
print ORD '#SBATCH --mail-user='."$ENV{'USER'}\n";
print ORD '#SBATCH -p fast'."\n";
print ORD '#SBATCH -o '.$outdir.'/fmriprep_end-%j'."\n";
print ORD ":\n";
close ORD;
my $order = 'sbatch --dependency=singleton '.$orderfile;
exec($order);
```

Nota: La llamada por defecto de *fmriprep-docker* lanza *docker* en un *TTY*. Es necesario tomar la orden y eliminar *-ti* para poder ejecutar la tarea desatendida. de otr manera no es posible.

```
[osotolongo@detritus mopead]$ cat slurm/fmriprep-113968
the input device is not a TTY
srun: error: brick02: task 0: Exited with exit code 1
RUNNING: docker run --rm -it -e DOCKER_VERSION_8395080871=19.03.5 -v
/nas/usr/local/opt/freesurfer/.license:/opt/freesurfer/license.txt:ro -v
/nas/data/mopead/bids:/data:ro -v /nas/data/mopead/fmriprep_out:/out
poldracklab/fmriprep:1.5.0 /data /out participant --participant-label
sub-0001 --ignore slicetiming --skip_bids_validation
fMRIPrep: Please report errors to
https://github.com/poldracklab/fmriprep/issues
```

Let's see what happens,

```
[osotolongo@detritus mopead]$ dfmriprep.pl mopead
Submitted batch job 114000
Submitted batch job 114001
Submitted batch job 114002
Submitted batch job 114003
Submitted batch job 114004
Submitted batch job 114005
Submitted batch job 114006
Submitted batch job 114007
Submitted batch job 114008
Submitted batch job 114009
Submitted batch job 114010
Submitted batch job 114011
Submitted batch job 114012
Submitted batch job 114013
Submitted batch job 114014
Submitted batch job 114015
Submitted batch job 114016
Submitted batch job 114017
Submitted batch job 114018
Submitted batch job 114019
Submitted batch job 114020
Submitted batch job 114021
Submitted batch job 114022
Submitted batch job 114023
```

```
[osotolongo@detritus mopead]$ squeue | grep fmri
             114004      fast fmriprep osotolon PD       0:00      1
(Resources)
             114005      fast fmriprep osotolon PD       0:00      1
(Priority)
             114006      fast fmriprep osotolon PD       0:00      1
(Priority)
             114007      fast fmriprep osotolon PD       0:00      1
(Priority)
             114008      fast fmriprep osotolon PD       0:00      1
(Priority)
             114009      fast fmriprep osotolon PD       0:00      1
(Priority)
             114010      fast fmriprep osotolon PD       0:00      1
(Priority)
             114011      fast fmriprep osotolon PD       0:00      1
(Priority)
             114012      fast fmriprep osotolon PD       0:00      1
(Priority)
             114013      fast fmriprep osotolon PD       0:00      1
(Priority)
             114014      fast fmriprep osotolon PD       0:00      1
(Priority)
             114015      fast fmriprep osotolon PD       0:00      1
(Priority)
```

```
             114016      fast fmriprep osotolon PD       0:00      1
(Priority)
             114017      fast fmriprep osotolon PD       0:00      1
(Priority)
             114018      fast fmriprep osotolon PD       0:00      1
(Priority)
             114019      fast fmriprep osotolon PD       0:00      1
(Priority)
             114020      fast fmriprep osotolon PD       0:00      1
(Priority)
             114021      fast fmriprep osotolon PD       0:00      1
(Priority)
             114022      fast fmriprep osotolon PD       0:00      1
(Priority)
             114023      fast fmriprep osotolon PD       0:00      1
(Dependency)
             114000      fast fmriprep osotolon  R       7:36      1 brick02
             114001      fast fmriprep osotolon  R       7:33      1 brick02
             114002      fast fmriprep osotolon  R       7:30      1 brick02
             114003      fast fmriprep osotolon  R       7:27      1 brick03
```

Errores y resultados

Hay errores en el procesamiento que empiezan a llegar antes que termine la cola de SLURM.

sub-0005

```
[osotolongo@detritus mopead]$ head slurm/fmriprep-114004
191123-21:06:29,704 nipype.workflow IMPORTANT:

    Running fMRIPREP version 1.5.0:
      * BIDS dataset path: /data.
      * Participant list: ['0005'].
      * Run identifier:
20191123-210621_6749c988-7df8-424b-80f3-5036ecb87d47.

191123-21:06:35,138 nipype.workflow IMPORTANT:
```

```
         Creating bold processing workflow for
"/data/sub-0005/func/sub-0005_task-rest_run-1_bold.nii.gz" (0.03 GB / 200
TRs). Memory resampled/largemem=0.12/0.19 GB.
191123-21:06:35,179 nipype.workflow IMPORTANT:


[osotolongo@detritus mopead]$ tail slurm/fmriprep-114004

191124-02:05:03,381 nipype.workflow INFO:
         [Node] Finished "_autorecon30".
191124-02:05:07,344 nipype.workflow ERROR:
         could not run node:
fmriprep_wf.single_subject_0005_wf.anat_preproc_wf.surface_recon_wf.autoreco
n_resume_wf.autorecon3
fMRIPrep failed: Workflow did not execute cleanly. Check log for details
Preprocessing did not finish successfully. Errors occurred while processing
data from participants: 0005 (1). Check the HTML reports for details.
/usr/local/miniconda/lib/python3.7/site-packages/scipy/fftpack/basic.py:160:
FutureWarning: Using a non-tuple sequence for multidimensional indexing is
deprecated; use `arr[tuple(seq)]` instead of `arr[seq]`. In the future this
will be interpreted as an array index, `arr[np.array(seq)]`, which will
result either in an error or a different result.
  z[index] = x
srun: error: brick02: task 0: Exited with exit code 1
```

Mirando los logs veo que es un error de Freesurfer al procesar el **T2**.

```
 mri_normalize -sigma 0.5 -nonmax_suppress 0 -min_dist 1 -aseg
/out/freesurfer/sub-0005/mri/aseg.presurf.mgz -surface
/out/freesurfer/sub-0005/surf/rh.white identity.nofile -surface
/out/freesurfer/sub-0005/surf/lh.white identity.nofile
/out/freesurfer/sub-0005/mri/T2.prenorm.mgz
/out/freesurfer/sub-0005/mri/T2.norm.mgz

mghRead(/out/freesurfer/sub-0005/mri/T2.prenorm.mgz): could not read 262144
bytes at slice 5
using Gaussian smoothing of bias field, sigma=0.500
disabling nonmaximum suppression
retaining  points that are at least 1.000mm from the boundary
using segmentation for initial intensity normalization
reading from /out/freesurfer/sub-0005/mri/T2.prenorm.mgz...
mri_normalize: could not open source file
/out/freesurfer/sub-0005/mri/T2.prenorm.mgz
Linux d4b2670a891a 3.10.0-514.26.2.el7.x86_64 #1 SMP Tue Jul 4 15:04:05 UTC
2017 x86_64 x86_64 x86_64 GNU/Linux

recon-all -s sub-0005 exited with ERRORS at Sun Nov 24 00:59:32 UTC 2019

For more details, see the log file /out/freesurfer/sub-0005/scripts/recon-
all.log
To report a problem, see
```

```
http://surfer.nmr.mgh.harvard.edu/fswiki/BugReporting

Standard error:

Return code: 1
```

sub-0011 (no BOLD images)

```
[osotolongo@detritus mopead]$ head slurm/fmriprep-114010
191124-03:26:53,956 nipype.workflow IMPORTANT:

    Running fMRIPREP version 1.5.0:
      * BIDS dataset path: /data.
      * Participant list: ['0011'].
      * Run identifier:
20191124-032649_5a197f26-695d-4dd9-9880-3e1bda481ae9.

Although ``--fs-no-reconall`` was not set (i.e., FreeSurfer is to be run),
no FreeSurfer output space (valid values are: fsnative, fsaverage,
fsaverage5, fsaverage6) was selected. Adding default "fsaverage5" to the
list of output spaces.
Process Process-2:
Traceback (most recent call last):

[osotolongo@detritus mopead]$ tail slurm/fmriprep-114010
  File "/usr/local/miniconda/lib/python3.7/multiprocessing/process.py", line
99, in run
    self._target(*self._args, **self._kwargs)
  File "/usr/local/miniconda/lib/python3.7/site-
packages/fmriprep/cli/run.py", line 674, in build_workflow
    work_dir=str(work_dir),
  File "/usr/local/miniconda/lib/python3.7/site-
packages/fmriprep/workflows/base.py", line 259, in init_fmriprep_wf
    use_syn=use_syn,
  File "/usr/local/miniconda/lib/python3.7/site-
packages/fmriprep/workflows/base.py", line 474, in init_single_subject_wf
    subject_id, task_id if task_id else '<all>'))
Exception: No BOLD images found for participant 0011 and task <all>. All
workflows require BOLD images.
srun: error: brick02: task 0: Exited with exit code 1
```

sub-0012 (igual que sub-0005?)

```
[osotolongo@detritus mopead]$ head slurm/fmriprep-114011
191124-03:27:22,854 nipype.workflow IMPORTANT:

    Running fMRIPREP version 1.5.0:
      * BIDS dataset path: /data.
```

```
      * Participant list: ['0012'].
      * Run identifier: 20191124-032718_af0a1ce7-0a7d-44c8-
bf5d-3b9dbff4c5fe.

191124-03:27:26,787 nipype.workflow IMPORTANT:
         Creating bold processing workflow for
"/data/sub-0012/func/sub-0012_task-rest_run-1_bold.nii.gz" (0.03 GB / 200
TRs). Memory resampled/largemem=0.12/0.18 GB.
191124-03:27:26,813 nipype.workflow IMPORTANT:

[osotolongo@detritus mopead]$ tail slurm/fmriprep-114011

191124-09:00:17,558 nipype.workflow INFO:
         [Node] Finished "_autorecon31".
191124-09:00:21,198 nipype.workflow ERROR:
         could not run node:
fmriprep_wf.single_subject_0012_wf.anat_preproc_wf.surface_recon_wf.autoreco
n_resume_wf.autorecon3
fMRIPrep failed: Workflow did not execute cleanly. Check log for details
Preprocessing did not finish successfully. Errors occurred while processing
data from participants: 0012 (1). Check the HTML reports for details.
/usr/local/miniconda/lib/python3.7/site-packages/scipy/fftpack/basic.py:160:
FutureWarning: Using a non-tuple sequence for multidimensional indexing is
deprecated; use `arr[tuple(seq)]` instead of `arr[seq]`. In the future this
will be interpreted as an array index, `arr[np.array(seq)]`, which will
result either in an error or a different result.
  z[index] = x
srun: error: brick02: task 0: Exited with exit code 1
```

Mirando los logs veo que es el mismo error de procesamiento con el Freesurfer y el T2.

```
 mri_normalize -sigma 0.5 -nonmax_suppress 0 -min_dist 1 -aseg
/out/freesurfer/sub-0012/mri/aseg.presurf.mgz -surface
/out/freesurfer/sub-0012/surf/rh.white identity.nofile -surface
/out/freesurfer/sub-0012/surf/lh.white identity.nofile
/out/freesurfer/sub-0012/mri/T2.prenorm.mgz
/out/freesurfer/sub-0012/mri/T2.norm.mgz

mghRead(/out/freesurfer/sub-0012/mri/T2.prenorm.mgz): could not read 262144
bytes at slice 40
using Gaussian smoothing of bias field, sigma=0.500
disabling nonmaximum suppression
retaining  points that are at least 1.000mm from the boundary
using segmentation for initial intensity normalization
reading from /out/freesurfer/sub-0012/mri/T2.prenorm.mgz...
mri_normalize: could not open source file
/out/freesurfer/sub-0012/mri/T2.prenorm.mgz
Linux 7ad12bcc9f58 3.10.0-514.26.2.el7.x86_64 #1 SMP Tue Jul 4 15:04:05 UTC
2017 x86_64 x86_64 x86_64 GNU/Linux

recon-all -s sub-0012 exited with ERRORS at Sun Nov 24 07:55:47 UTC 2019
```

```
For more details, see the log file /out/freesurfer/sub-0012/scripts/recon-
all.log
To report a problem, see
http://surfer.nmr.mgh.harvard.edu/fswiki/BugReporting

Standard error:

Return code: 1
```

sub-0014 (igual que sub-0012)

```
[osotolongo@detritus mopead]$ head slurm/fmriprep-114013
191124-04:28:31,129 nipype.workflow IMPORTANT:

    Running fMRIPREP version 1.5.0:
      * BIDS dataset path: /data.
      * Participant list: ['0014'].
      * Run identifier:
20191124-042826_fdabd91c-4fe9-4aba-9e07-436863f2664e.

191124-04:28:33,859 nipype.workflow IMPORTANT:
         Creating bold processing workflow for
"/data/sub-0014/func/sub-0014_task-rest_run-1_bold.nii.gz" (0.03 GB / 200
TRs). Memory resampled/largemem=0.13/0.19 GB.
191124-04:28:33,888 nipype.workflow IMPORTANT:


[osotolongo@detritus mopead]$ tail slurm/fmriprep-114013

191124-09:45:38,729 nipype.workflow INFO:
         [Node] Finished "_autorecon31".
191124-09:45:42,379 nipype.workflow ERROR:
         could not run node:
fmriprep_wf.single_subject_0014_wf.anat_preproc_wf.surface_recon_wf.autoreco
n_resume_wf.autorecon3
fMRIPrep failed: Workflow did not execute cleanly. Check log for details
Preprocessing did not finish successfully. Errors occurred while processing
data from participants: 0014 (1). Check the HTML reports for details.
/usr/local/miniconda/lib/python3.7/site-packages/scipy/fftpack/basic.py:160:
FutureWarning: Using a non-tuple sequence for multidimensional indexing is
deprecated; use `arr[tuple(seq)]` instead of `arr[seq]`. In the future this
will be interpreted as an array index, `arr[np.array(seq)]`, which will
result either in an error or a different result.
  z[index] = x
srun: error: brick02: task 0: Exited with exit code 1
```

```
 mri_normalize -sigma 0.5 -nonmax_suppress 0 -min_dist 1 -aseg
/out/freesurfer/sub-0014/mri/aseg.presurf.mgz -surface
/out/freesurfer/sub-0014/surf/rh.white identity.nofile -surface
```

```
/out/freesurfer/sub-0014/surf/lh.white identity.nofile
/out/freesurfer/sub-0014/mri/T2.prenorm.mgz
/out/freesurfer/sub-0014/mri/T2.norm.mgz

znzTAGskip: tag=1111638594, failed to calloc 1111639040 bytes!

using Gaussian smoothing of bias field, sigma=0.500
disabling nonmaximum suppression
retaining  points that are at least 1.000mm from the boundary
using segmentation for initial intensity normalization
reading from /out/freesurfer/sub-0014/mri/T2.prenorm.mgz...
Cannot allocate memory
Linux 16c0f722c6f0 3.10.0-514.26.2.el7.x86_64 #1 SMP Tue Jul 4 15:04:05 UTC
2017 x86_64 x86_64 x86_64 GNU/Linux

recon-all -s sub-0014 exited with ERRORS at Sun Nov 24 08:55:41 UTC 2019

For more details, see the log file /out/freesurfer/sub-0014/scripts/recon-
all.log
To report a problem, see
http://surfer.nmr.mgh.harvard.edu/fswiki/BugReporting

Standard error:

Return code: 1
```

Nota: Revisando los logs veo que Freesurfer falla por problemas de memoria, (**Cannot allocate memory**). Han de revisarse las opciones de *fmriprep*.

Existe la opcion *–fs-no-reconall* que se salta el procesamiento de Freesurfer. Habria que probar los resultados y comparar.

Probando sin Freesurfer

Voy a tomar *sub-0005* e intentar correr el pipeline sin Freesurfer. Esta puede ser una alternativa al procesamiento de los sujetos que fallen. (*incluirla en el wrapper*)

```
[osotolongo@brick01 mopead]$ docker run --rm -v
/nas/usr/local/opt/freesurfer/.license:/opt/freesurfer/license.txt:ro -v
/nas/data/mopead/bids:/data:ro -v /nas/data/mopead/fmriprep_out:/out
poldracklab/fmriprep:1.5.0 /data /out participant --participant-label
sub-0005 --ignore slicetiming --skip_bids_validation --fs-no-reconall

191124-11:13:51,496 nipype.workflow IMPORTANT:

    Running fMRIPREP version 1.5.0:
      * BIDS dataset path: /data.
      * Participant list: ['0005'].
      * Run identifier: 20191124-111340_2f180886-582f-4eef-
```

```
ac91-030ef42c3e7b.

191124-11:13:53,371 nipype.workflow IMPORTANT:
        Creating bold processing workflow for
"/data/sub-0005/func/sub-0005_task-rest_run-1_bold.nii.gz" (0.03 GB / 200
TRs). Memory resampled/largemem=0.12/0.19 GB.
191124-11:13:53,404 nipype.workflow IMPORTANT:
        No single-band-reference found for sub-0005_task-
rest_run-1_bold.nii.gz
191124-11:13:53,844 nipype.workflow WARNING:
        SDC: no fieldmaps found or they were ignored
(/data/sub-0005/func/sub-0005_task-rest_run-1_bold.nii.gz).
191124-11:13:59,954 nipype.workflow IMPORTANT:
        Works derived from this fMRIPrep execution should include the
following boilerplate:
........
```

Esto funciona. Ha de modificarse el wrapper para añadir la opción (*-nofs*) y que se interprete como lanzar *fmriprep* con *–fs-no-reconall*. Asi que hay que lanzar el wrapper, anotar los errores y usar las opciones *-cut* y *-nofs* para relanzarel proceso solo sobre los que hayan salido mal.

Y ya que estamos. Si no existe el archivo fMRI se podria evitar lanzar el proceso.

dfmriprep.pl

```
#!/usr/bin/perl
# Copyright 2019 O. Sotolongo <asqwerty@gmail.com>
use strict; use warnings;

use File::Find::Rule;
use NEURO qw(print_help get_pair load_study achtung shit_done get_lut
check_or_make centiloid_fbb);
use Data::Dump qw(dump);
use File::Remove 'remove';
use File::Basename qw(basename);

my $fslic = '/nas/usr/local/opt/freesurfer/.license';
my $cfile;
my $fs = 1;
@ARGV = ("-h") unless @ARGV;

while (@ARGV and $ARGV[0] =~ /^-/) {
    $_ = shift;
    last if /^--$/;
    if (/^-cut/) { $cfile = shift; chomp($cfile);}
    if (/^-nofs/) { $fs = 0};
    if (/^-h$/) { print_help $ENV{'PIPEDIR'}.'/doc/cfmriprep.hlp';
exit;}
}
my $study = shift;
```

```
unless ($study) { print_help $ENV{'PIPEDIR'}.'/doc/cfmriprep.hlp';
exit;}
my %std = load_study($study);
my $w_dir = $std{'WORKING'};
my $data_dir = $std{'DATA'};
my $bids_dir = $data_dir.'/bids';
my $fmriout_dir;
my $nofscall;
if($fs){
        $fmriout_dir = $data_dir.'/fmriprep_out';
        $nofscall = "\n";
}else{
        $fmriout_dir = $data_dir.'/fmriprep_nofs_out';
        $nofscall = " --fs-no-reconall\n";
}
check_or_make($fmriout_dir);
my $outdir = "$std{'DATA'}/slurm";
check_or_make($outdir);

my @subjects;
if($cfile){
        open DBF, $cfile or die "No such file\n";
        while(<DBF>) {
                chomp;
                push @subjects, $_;
        }
        close DBF;
}else{
        opendir DBD, $bids_dir or die "Cold not open dir\n";
        while (my $thing = readdir DBD){
                if ($thing eq '.' or $thing eq '..') {
                        next;
                }
                if ($thing =~ /sub-*/ && (-e "$bids_dir/$thing/func" &&
-d "$bids_dir/$thing/func")) {
                        push @subjects, $thing;
                }
        }
        closedir DBD;
}

foreach my $subject (@subjects) {
        my $orderfile = $outdir.'/'.$subject.'_fmriprep.sh';
        open ORD, ">$orderfile";
        print ORD '#!/bin/bash'."\n";
        print ORD '#SBATCH -J fmriprep_'.$study."\n";
        print ORD '#SBATCH --time=72:0:0'."\n"; #si no ha terminado en
X horas matalo
        print ORD '#SBATCH --mail-type=FAIL,TIME_LIMIT,STAGE_OUT'."\n";
#no quieres que te mande email de todo
        print ORD '#SBATCH -o '.$outdir.'/fmriprep-%j'."\n";
```

```
        print ORD '#SBATCH -c 20'."\n";
        print ORD '#SBATCH -p fast'."\n";
        print ORD '#SBATCH --mail-user='."$ENV{'USER'}\n";
#docker run --rm -it -e DOCKER_VERSION_8395080871=19.03.5 -v
/nas/usr/local/opt/freesurfer/.license:/opt/freesurfer/license.txt:ro -
v /nas/data/mopead/bids:/data:ro -v /nas/data/mopead/fmriprep_out:/out
poldracklab/fmriprep:1.5.0 /data /out participant --participant-label
sub-0001 --ignore slicetiming --skip_bids_validation
        print ORD 'srun docker run --rm -v
'.$fslic.':/opt/freesurfer/license.txt:ro -v '.$bids_dir.':/data:ro -v
'.$fmriout_dir.':/out poldracklab/fmriprep:1.5.0 /data /out participant
--participant-label '.$subject.' --ignore slicetiming --
skip_bids_validation'.$nofscall;
        close ORD;
        system("sbatch $orderfile");
        sleep(2);
}
my $orderfile = $outdir.'/fmriprep_end.sh';
open ORD, ">$orderfile";
print ORD '#!/bin/bash'."\n";
print ORD '#SBATCH -J fmriprep_'.$study."\n";
print ORD '#SBATCH --mail-type=END'."\n"; #email cuando termine
print ORD '#SBATCH --mail-user='."$ENV{'USER'}\n";
print ORD '#SBATCH -p fast'."\n";
print ORD '#SBATCH -o '.$outdir.'/fmriprep_end-%j'."\n";
print ORD ":\n";
close ORD;
my $order = 'sbatch --dependency=singleton '.$orderfile;
exec($order);
```

Asi se evitan los errores de no existencia del archivo fMRI y se pueden relanzar todos los procesos erroneos, esperando que los problemas hayan sido solo con Freesurfer. Para esto basta añadir,

```
&& (-e "$bids_dir/$thing/func" && -d "$bids_dir/$thing/func")
```

cuando se chequea que exista el *BIDS*. Se ve que elimina los conflictivos (*sub-0011*, *sub-0013*, etc).

```
(
  "sub-0001",
  "sub-0002",
  "sub-0003",
  "sub-0004",
  "sub-0005",
  "sub-0006",
  "sub-0007",
  "sub-0008",
  "sub-0009",
  "sub-0010",
  "sub-0012",
```

```
  "sub-0014",
  "sub-0015",
  "sub-0017",
  "sub-0018",
  "sub-0019",
  "sub-0020",
  "sub-0021",
  "sub-0022",
  "sub-0023",
)
```

Ahora, en aras de que todas las imagenes tengan el mismo analisis, se podria lanzar el analisis sin Freesurfer para todas. He separado los analisis en directorios distintos previendo esta eventualidad.

```
[osotolongo@detritus mopead]$ dfmriprep.pl -nofs mopead
Submitted batch job 114025
Submitted batch job 114026
Submitted batch job 114027
Submitted batch job 114028
Submitted batch job 114029
Submitted batch job 114030
Submitted batch job 114031
Submitted batch job 114032
Submitted batch job 114033
Submitted batch job 114034
Submitted batch job 114035
Submitted batch job 114036
Submitted batch job 114037
Submitted batch job 114038
Submitted batch job 114039
Submitted batch job 114040
Submitted batch job 114041
Submitted batch job 114042
Submitted batch job 114043
Submitted batch job 114044
Submitted batch job 114045
[osotolongo@detritus mopead]$ squeue | grep fmri
             114029      fast fmriprep osotolon PD       0:00      1
(Resources)
             114030      fast fmriprep osotolon PD       0:00      1
(Priority)
             114031      fast fmriprep osotolon PD       0:00      1
(Priority)
             114032      fast fmriprep osotolon PD       0:00      1
(Priority)
             114033      fast fmriprep osotolon PD       0:00      1
(Priority)
             114034      fast fmriprep osotolon PD       0:00      1
(Priority)
             114035      fast fmriprep osotolon PD       0:00      1
(Priority)
```

```
             114036      fast fmriprep osotolon PD       0:00      1
(Priority)
             114037      fast fmriprep osotolon PD       0:00      1
(Priority)
             114038      fast fmriprep osotolon PD       0:00      1
(Priority)
             114039      fast fmriprep osotolon PD       0:00      1
(Priority)
             114040      fast fmriprep osotolon PD       0:00      1
(Priority)
             114041      fast fmriprep osotolon PD       0:00      1
(Priority)
             114042      fast fmriprep osotolon PD       0:00      1
(Priority)
             114043      fast fmriprep osotolon PD       0:00      1
(Priority)
             114044      fast fmriprep osotolon PD       0:00      1
(Priority)
             114045      fast fmriprep osotolon PD       0:00      1
(Dependency)
             114025      fast fmriprep osotolon  R       1:05      1 brick02
             114026      fast fmriprep osotolon  R       1:02      1 brick02
             114027      fast fmriprep osotolon  R       0:59      1 brick02
             114028      fast fmriprep osotolon  R       0:59      1 brick03
```

Y ahi tenemos el inicio de la estructura,

```
[osotolongo@detritus mopead]$ ls fmriprep_nofs_out/fmriprep/
logs  sub-0001  sub-0002  sub-0003  sub-0004
```

Con slice timing

Si el archivo BIDS se obtiene directamente del DICOM (Convertir DICOMs a BIDS) resulta que la informacion de SliceTiming si que queda registrada en el archivo *.json*. Asi que voy aintentar que se haga la *slicetiming correction*. Despues tocara probar tambien con *freesurfer*.

Veamos el comando,

```
[osotolongo@brick01 ~]$ fmriprep-docker /nas/data/mopead/bids/
/nas/data/mopead/fmriprep --participant-label "sub-0003" --
skip_bids_validation --fs-license-file
/nas/usr/local/opt/freesurfer/.license --fs-no-reconall
```

El resultado,

```
[osotolongo@detritus mopead]$ tree fmriprep/fmriprep/sub-0003/
fmriprep/fmriprep/sub-0003/
├── anat
│   ├── sub-0003_desc-aparcaseg_dseg.nii.gz
```

```
│   ├── sub-0003_desc-aseg_dseg.nii.gz
│   ├── sub-0003_desc-brain_mask.json
│   ├── sub-0003_desc-brain_mask.nii.gz
│   ├── sub-0003_desc-preproc_T1w.json
│   ├── sub-0003_desc-preproc_T1w.nii.gz
│   ├── sub-0003_dseg.nii.gz
│   ├── sub-0003_from-MNI152NLin2009cAsym_to-T1w_mode-image_xfm.h5
│   ├── sub-0003_from-orig_to-T1w_mode-image_xfm.txt
│   ├── sub-0003_from-T1w_to-fsnative_mode-image_xfm.txt
│   ├── sub-0003_from-T1w_to-MNI152NLin2009cAsym_mode-image_xfm.h5
│   ├── sub-0003_hemi-L_inflated.surf.gii
│   ├── sub-0003_hemi-L_midthickness.surf.gii
│   ├── sub-0003_hemi-L_pial.surf.gii
│   ├── sub-0003_hemi-L_smoothwm.surf.gii
│   ├── sub-0003_hemi-R_inflated.surf.gii
│   ├── sub-0003_hemi-R_midthickness.surf.gii
│   ├── sub-0003_hemi-R_pial.surf.gii
│   ├── sub-0003_hemi-R_smoothwm.surf.gii
│   ├── sub-0003_label-CSF_probseg.nii.gz
│   ├── sub-0003_label-GM_probseg.nii.gz
│   ├── sub-0003_label-WM_probseg.nii.gz
│   ├── sub-0003_space-MNI152NLin2009cAsym_desc-brain_mask.json
│   ├── sub-0003_space-MNI152NLin2009cAsym_desc-brain_mask.nii.gz
│   ├── sub-0003_space-MNI152NLin2009cAsym_desc-preproc_T1w.json
│   ├── sub-0003_space-MNI152NLin2009cAsym_desc-preproc_T1w.nii.gz
│   ├── sub-0003_space-MNI152NLin2009cAsym_dseg.nii.gz
│   ├── sub-0003_space-MNI152NLin2009cAsym_label-CSF_probseg.nii.gz
│   ├── sub-0003_space-MNI152NLin2009cAsym_label-GM_probseg.nii.gz
│   └── sub-0003_space-MNI152NLin2009cAsym_label-WM_probseg.nii.gz
├── figures
│   ├── sub-0003_desc-reconall_T1w.svg
│   ├── sub-0003_dseg.svg
│   ├── sub-0003_space-MNI152NLin2009cAsym_T1w.svg
│   ├── sub-0003_task-rest_run-1_desc-bbregister_bold.svg
│   ├── sub-0003_task-rest_run-1_desc-carpetplot_bold.svg
│   ├── sub-0003_task-rest_run-1_desc-compcorvar_bold.svg
│   ├── sub-0003_task-rest_run-1_desc-confoundcorr_bold.svg
│   ├── sub-0003_task-rest_run-1_desc-flirtbbr_bold.svg
│   └── sub-0003_task-rest_run-1_desc-rois_bold.svg
└── func
    ├── sub-0003_task-rest_run-1_desc-confounds_regressors.json
    ├── sub-0003_task-rest_run-1_desc-confounds_regressors.tsv
    ├── sub-0003_task-rest_run-1_space-fsaverage5_hemi-L.func.gii
    ├── sub-0003_task-rest_run-1_space-fsaverage5_hemi-R.func.gii
    ├── sub-0003_task-rest_run-1_space-MNI152NLin2009cAsym_boldref.nii.gz
    ├── sub-0003_task-rest_run-1_space-MNI152NLin2009cAsym_desc-
aparcaseg_dseg.nii.gz
    ├── sub-0003_task-rest_run-1_space-MNI152NLin2009cAsym_desc-
aseg_dseg.nii.gz
    ├── sub-0003_task-rest_run-1_space-MNI152NLin2009cAsym_desc-
brain_mask.json
```

```
    ├── sub-0003_task-rest_run-1_space-MNI152NLin2009cAsym_desc-
brain_mask.nii.gz
    ├── sub-0003_task-rest_run-1_space-MNI152NLin2009cAsym_desc-
preproc_bold.json
    └── sub-0003_task-rest_run-1_space-MNI152NLin2009cAsym_desc-
preproc_bold.nii.gz

3 directories, 50 files
```

Vamos a probar con *FS*,

```
[osotolongo@brick01 ~]$ fmriprep-docker /nas/data/mopead/bids/
/nas/data/mopead/fmriprep --participant-label "sub-0004" --
skip_bids_validation --fs-license-file
/nas/usr/local/opt/freesurfer/.license
```

Aprovechando la segmentación ya hecha

https://fmriprep.readthedocs.io/en/stable/workflows.html#surface-preprocessing

> Surface processing will be skipped if the outputs already exist.
>
> In order to bypass reconstruction in fMRIPrep, place existing reconstructed subjects in <output dir>/freesurfer prior to the run, or specify an external subjects directory with the –fs-subjects-dir flag. fMRIPrep will perform any missing recon-all steps, but will not perform any steps whose outputs already exist.

Basicamente cambiamos la orden de ejecucion,

```
'srun docker run --rm -v '.$fslic.':/opt/freesurfer/license.txt:ro -v
'.$bids_dir.'/sub-'.$subject.':/data:ro -v '.$fmriout_dir.':/out
poldracklab/fmriprep:'.$fmriprep_version.' /data /out participant --
participant-label '.$subject.' '.$noslicetiming.' --skip_bids_validation --
fs-subjects-dir '.$fssubsdir;
```

Claro que antes hay que copiar los archivos de FS a un nuevo sitio, etc

dfmriprep.pl

```
#!/usr/bin/perl

# Copyright 2019 O. Sotolongo <asqwerty@gmail.com>

# This program is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation; either version 2 of the License, or
# (at your option) any later version.
#
```

```
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
#
use strict; use warnings;

use File::Find::Rule;
use NEURO4 qw(print_help load_project shit_done cut_shit check_subj
check_or_make);
use Data::Dump qw(dump);
use File::Remove 'remove';
use File::Basename qw(basename);
use File::Copy::Recursive qw(dircopy);

my $fslic = '/nas/usr/local/opt/freesurfer/.license';
my $fmriprep_version = '1.5.6';
my $cfile="";
#my $fs = 0;
my $st = 0;
@ARGV = ("-h") unless @ARGV;

while (@ARGV and $ARGV[0] =~ /^-/) {
    $_ = shift;
    last if /^--$/;
    if (/^-cut/) { $cfile = shift; chomp($cfile);}
#    if (/^-fs/) { $fs = 1};
    if (/^-st/) {$st = 1};
    if (/^-h$/) { print_help $ENV{'PIPEDIR'}.'/doc/dfmriprep.hlp';
exit;}
}
my $study = shift;
unless ($study) { print_help $ENV{'PIPEDIR'}.'/doc/dfmriprep.hlp';
exit;}
my %std = load_project($study);
my $w_dir = $std{'WORKING'};
my $data_dir = $std{'DATA'};
my $bids_dir = $std{'BIDS'};
my $fsdir = $ENV{'SUBJECTS_DIR'};
my $db = $std{'DATA'}.'/'.$study.'_mri.csv';
my $fmriout_dir = $data_dir.'/fmriprep_out';
my $fssubsdir = $fmriout_dir.'/freesurfer';
my $noslicetiming;
if($st){
        $noslicetiming = "";
}else{
        $noslicetiming = "--ignore slicetiming";
}
check_or_make($fmriout_dir);
check_or_make($fssubsdir);
my $outdir = "$std{'DATA'}/slurm";
```

```
check_or_make($outdir);

my @subjects = cut_shit($db, $data_dir."/".$cfile);

foreach my $subject (@subjects) {
        my %nifti = check_subj($std{'DATA'},$subject);
        if($nifti{'func'}){
                my $fssd = $fssubsdir.'/sub-'.$subject;
                my $rfssd = $fsdir.'/'.$study.'_'.$subject;
                check_or_make($fssd);
                dircopy($rfssd,$fssd);
                my $orderfile = $outdir.'/'.$subject.'_fmriprep.sh';
                open ORD, ">$orderfile";
                print ORD '#!/bin/bash'."\n";
                print ORD '#SBATCH -J fmriprep_'.$study."\n";
                print ORD '#SBATCH --time=72:0:0'."\n"; #si no ha
terminado en X horas matalo
                print ORD '#SBATCH --mail-
type=FAIL,TIME_LIMIT,STAGE_OUT'."\n"; #no quieres que te mande email de
todo
                print ORD '#SBATCH -o '.$outdir.'/fmriprep-%j'."\n";
                print ORD '#SBATCH -c 20'."\n";
                print ORD '#SBATCH -p fast'."\n";
                print ORD '#SBATCH --mail-user='."$ENV{'USER'}\n";
#docker run --rm -it -e DOCKER_VERSION_8395080871=19.03.5 -v
/nas/usr/local/opt/freesurfer/.license:/opt/freesurfer/license.txt:ro -
v /nas/data/mopead/bids:/data:ro -v /nas/data/mopead/fmriprep_out:/out
poldracklab/fmriprep:1.5.0 /data /out participant --participant-label
sub-0001 --ignore slicetiming --skip_bids_validation
                print ORD 'srun docker run -t -d --rm -v
'.$fslic.':/opt/freesurfer/license.txt:ro -v '.$bids_dir.'/sub-
'.$subject.':/data:ro -v '.$fmriout_dir.':/out
poldracklab/fmriprep:'.$fmriprep_version.' /data /out participant --
participant-label '.$subject.' '.$noslicetiming.' --
skip_bids_validation --fs-subjects-dir out/freesurfer';
                close ORD;
                system("sbatch $orderfile");
                #sleep(2);
        }
}
my $orderfile = $outdir.'/fmriprep_end.sh';
open ORD, ">$orderfile";
print ORD '#!/bin/bash'."\n";
print ORD '#SBATCH -J fmriprep_'.$study."\n";
print ORD '#SBATCH --mail-type=END'."\n"; #email cuando termine
print ORD '#SBATCH --mail-user='."$ENV{'USER'}\n";
print ORD '#SBATCH -p fast'."\n";
print ORD '#SBATCH -o '.$outdir.'/fmriprep_end-%j'."\n";
print ORD ":\n";
close ORD;
my $order = 'sbatch --dependency=singleton '.$orderfile;
```

```
exec($order);
```

Nota: Ahora mismo hay algun problema de comunicación entre *docker* y *slurm* (mas probablemente de *docker*).*scancel* termina la tarea pero el container sigue procesando. He de probar si cuando termina naturalmente el container cae. Esto quiere decir que **aunque el proceso de TIME_OUT puede haber terminado satisfactoriamente**.Se ha de comprobar los archivos de output.

Upgrade fmriprep

Como root en cada nodo,

```
$ docker pull poldracklab/fmriprep:1.5.6
```

En este caso **1.5.6** es la ultima version. Despues editar */nas/fotware/neuro.dev/bin/dfmriprep.pl* . Buscar la linea,

```
my $fmriprep_version = '1.5.3';
```

y cambiar a la ultima versión,

```
my $fmriprep_version = '1.5.6';
```