Proyecto BIOFACE

Conversion DICOMs

Las imagenes contienen los siguientes protocolos,

```
[osotolongo@detritus bioface]$ for x in `find /nas/bioface_raw/B002/ -type
f`; do dckey -k "SeriesDescription" ${x} 2>&1; done | uniq
AAhead_scout
AAhead_scout_MPR_sag
AAhead_scout_MPR_cor
AAhead_scout_MPR_tra
t1_mprage_sag_p2_iso
t2_space_dark-fluid_sag_p2_iso
pd+t2_tse_tra_p2_3mm
Mag_Images
Pha_Images
mIP_Images(SW)
SWI_Images
asl_3d_tra_iso_3.0_highres
asl_3d_tra_iso_3.0_highres_Mosaics
Perfusion_Weighted
Perfusion_Weighted_Mosaics
DTIep2d_diff_mddw_48dir_p3_AP
DTIep2d_diff_mddw_48dir_p3_AP_Mosaics
DTIep2d_diff_mddw_48dir_p3_AP_ADC
DTIep2d_diff_mddw_48dir_p3_AP_TRACEW
DTIep2d_diff_mddw_48dir_p3_AP_FA
DTIep2d_diff_mddw_48dir_p3_AP_TENSOR
DTIep2d_diff_mddw_48dir_p3_AP_TENSOR_B0
DTIep2d_diff_mddw_4b0_PA
PhoenixZIPReport
MPR sag 3mm Range
MPR cor 3mm Range
MPR tra 3mm Range
MPR sag 3mm Range[2]
MPR cor 3mm Range[2]
MPR tra 3mm Range[2]
```

De aquí en principio nos interesan *t1_mprage_sag_p2_iso* para procesar las T1 con Freesurfer, *DTIep2d_diff_mddw_48dir_p3_AP* y *DTIep2d_diff_mddw_4b0_PA* para el procesamiento DTI.

Lo primero es preparar una base de datos anonimizada para el procesamiento. Los sujetos ya estan anonimizados pero es mejor tenerlos separados con los codigos de neuroimagen.

El script es simple, basado en MOPEAD

updatedb.pl

```
#!/usr/bin/perl

use strict; use warnings;
use NEURO qw(load_study);
use Data::Dump qw(dump);

my $study = "bioface";
my %std = load_study($study);
my $src_dir = '/nas/bioface_raw/';
my $ids_file = $std{'DATA'}.'/ids.csv';
my $names_file =$std{'DATA'}.'/ni_names.csv';
my $proj_file = $std{'DATA'}.'/bioface.csv';

#Leo los que han subido
my @orig_str = qx/ls $src_dir/;
chomp @orig_str;

my %idsinfo;
my $idscount;
open IDS, "<$ids_file" or die $!;
while(<IDS>){
        chomp;
        my ($key, $value) = split(/,\s?/);
        $idsinfo{$key} = $value;
        $idscount = $value;
}
$idscount++;
close IDS;
my @not_in_db = grep !$idsinfo{$_}, @orig_str;
foreach my $guy (@not_in_db) {
        $idsinfo{$guy} = $idscount;
        $idscount++;
}
open IDS, ">$ids_file" or die $!;
open NDS, ">$names_file" or die $!;
open PDS, ">$proj_file" or die $!;
foreach my $guy (sort {$idsinfo{$a} <=> $idsinfo{$b}} keys %idsinfo) {
        print IDS $guy,",",$idsinfo{$guy},"\n";
        printf NDS "%s,%04d\n", $guy,$idsinfo{$guy};
        printf PDS "%04d;bfa\n", $idsinfo{$guy};
}
close PDS;
close NDS;
close IDS;
```

El output son tres archivos, los IDs de sujeto del proyecto y de neuroimagen,

```
[osotolongo@detritus bioface]$ cat ids.csv
B002,1
B003,2
B006,3
B008,4
B009,5
...
```

identico pero con la cadena completa,

```
[osotolongo@detritus bioface]$ cat ni_names.csv
B002,0001
B003,0002
B006,0003
B008,0004
B009,0005
...
```

y una lista de los directorios procesados,

```
[osotolongo@detritus bioface]$ cat converted_t1.csv
B002
B003
B006
B008
B009
...
```

Extraccion de fechas

1,2,3.. probando

```
[osotolongo@detritus bioface]$ for x in /nas/bioface_raw/*; do d=$(for y in ${x}/*; do dckey -k "SeriesDate" ${y} 2>&1; done | head -n 1); n=$(echo ${x} | awk -F"/" '{print $4}'); echo ${n},${d}; done
B001,20180719
B002,20180907
.....
```

Es lento pero seguro,

```
[osotolongo@detritus bioface]$ for x in /nas/bioface_raw/*; do d=$(for y in ${x}/*; do dckey -k "SeriesDate" ${y} 2>&1; done | head -n 1); n=$(echo ${x} | awk -F"/" '{print $4}'); echo ${n},${d}; done > dcm_dates.csv
```

bien hecho, solo pregunto al primer file,

```
[osotolongo@detritus bioface]$ for x in /nas/bioface_raw/*; do d=$(for y in
```

```
`ls ${x}/* | head -n 1`; do dckey -k "SeriesDate" ${y} 2>&1; done | head -n 1); n=$(echo ${x} | awk -F"/" '{print $4}'); echo ${n},${d}; done > dcm_dates.csv
```

Conversion

NIfTI

Igual que en MOPEAD. Convertimos todo primero. Lo almacenamos y despues escogemos los archivos necesarios para el procesamiento.

Codigo aqui

update_nifti.pl

```
#!/usr/bin/perl

use strict; use warnings;
use NEURO qw(load_study check_or_make);
use Data::Dump qw(dump);
use File::Find::Rule;

my $study = "bioface";
my %std = load_study($study);
my $src_dir = '/nas/bioface_raw/';
my $conv_file = $std{'DATA'}.'/converted_nifti.csv';
my $ids_file = $std{'DATA'}.'/ids.csv';
my $names_file =$std{'DATA'}.'/ni_names.csv';
my $proj_file = $std{'DATA'}.'/bioface.csv';

#Leo la DB
my %idsinfo;
open IDS, "<$ids_file" or die $!;
while(<IDS>){
        chomp;
        my ($key, $value) = split(/,\s?/);
        $idsinfo{$key} = $value;
}
close IDS;
#Leo los que ya he descomprimido
my @orig_str = qx/ls $src_dir/;
chomp @orig_str;
#Leo los que se han convertido
my @conv_str;
open CONV, "<$conv_file" or die $!;
chomp (@conv_str = <CONV>);
close CONV;
#Y a ver cuantos faltan por convertir!
```

```
my %conv_str = map { $_ => 1 } @conv_str;
my @not_conv = grep !$conv_str{$_}, @orig_str;
# y a convertirlos!
foreach my $guy (@not_conv){
        if(exists($idsinfo{$guy})){
                my $odir = $std{'DATA'}.'/niftis/'.$guy;
                check_or_make($odir);
                my $order = 'dcm2niix -z i -b y -o '.$odir.'
'.$src_dir.'/'.$guy;
                print "$order\n";
                system($order);
        }
}
#actualizo los convertidos
push(@conv_str, @not_conv);
open CONV, ">$conv_file" or die $!;
print CONV "$_\n" foreach @conv_str;
close CONV;
```

T1

Aqui almacenamos en la carpeta *mri* las imagenes con etiqueta *t1_mprage_sag_p2_iso*. Voy a utilizar el mismo archivo *update_t1.pl* que su uso para el proyecto MOPEAD.

solo hay que retocarlo un poco

update_t1.pl

```
#!/usr/bin/perl

use strict; use warnings;
use NEURO qw(load_study);
use Data::Dump qw(dump);
use File::Find::Rule;

my $study = "bioface";
my %std = load_study($study);
my $src_dir = '/nas/bioface_raw/';
my $conv_file = $std{'DATA'}.'/converted_t1.csv';
my $ids_file = $std{'DATA'}.'/ids.csv';
my $names_file =$std{'DATA'}.'/ni_names.csv';
my $proj_file = $std{'DATA'}.$study.'.csv';
my @other_exts = ("nii.gz","json");
my $series = 5;
my $tag = 't1_mprage_sag_p2_iso';
#Leo la DB
my %idsinfo;
open IDS, "<$ids_file" or die $!;
```

```
while(<IDS>){
        chomp;
        my ($key, $value) = split(/,\s?/);
        $idsinfo{$key} = $value;
}
close IDS;
#Leo los que ya he descomprimido
my @orig_str = qx/ls $src_dir/;
chomp @orig_str;
#Leo los que se han convertido
my @conv_str;
open CONV, "<$conv_file" or die $!;
chomp (@conv_str = <CONV>);
close CONV;
#Y a ver cuantos faltan por convertir!
my %conv_str = map { $_ => 1 } @conv_str;
my @not_conv = grep !$conv_str{$_}, @orig_str;
# y a convertirlos!
foreach my $guy (@not_conv){
        if(exists($idsinfo{$guy})){
                my @conv_files = find(file => 'name' =>
"*$tag*.nii.gz", in => $std{'DATA'}.'/niftis/'.$guy.'/');
                my $count = $series;
                foreach my $nii_file (@conv_files){
                        my $order = 'fslval '.$nii_file.' dim1';
                        my $xinfo = qx/$order/;
                        if ($xinfo == '176'){
                                my $file_name = $nii_file;
                                foreach my $ext (@other_exts){
                                        ($file_name = $nii_file) =~
s/nii.gz$/$ext/;
                                        $order = sprintf "cp %s
%s/bfa%04ds%04d.%s", $file_name, $std{'MRI'}, $idsinfo{$guy}, $count,
$ext;
                                        print "$order\n";
                                        system($order);
                                }
                        $count++;
                        }
                }
        }
}
#actualizo los convertidos
push(@conv_str, @not_conv);
open CONV, ">$conv_file" or die $!;
print CONV "$_\n" foreach @conv_str;
close CONV;
```

Aqui tambien se genera el archivo *converted_t1.csv* que almacena los T1 que ya se han convertido.

De esta manera las imagenes que ya se han copiado no se reprocesan de nuevo y se ahorra mucho tiempo.

El output son las T1 en el directorio *mri* del proyecto, listas para el procesamiento,

```
[osotolongo@detritus bioface]$ ls mri
bfa0001s0012.json    bfa0004s0012.json    bfa0007s0012.json
bfa0010s0012.json    bfa0013s0012.json    bfa0016s0012.json
bfa0019s0012.json    bfa0022s0012.json
bfa0001s0012.nii.gz  bfa0004s0012.nii.gz  bfa0007s0012.nii.gz
bfa0010s0012.nii.gz  bfa0013s0012.nii.gz  bfa0016s0012.nii.gz
bfa0019s0012.nii.gz  bfa0022s0012.nii.gz
bfa0002s0012.json    bfa0005s0012.json    bfa0008s0012.json
bfa0011s0012.json    bfa0014s0012.json    bfa0017s0012.json
bfa0020s0012.json    bfa0023s0012.json
bfa0002s0012.nii.gz  bfa0005s0012.nii.gz  bfa0008s0012.nii.gz
bfa0011s0012.nii.gz  bfa0014s0012.nii.gz  bfa0017s0012.nii.gz
bfa0020s0012.nii.gz  bfa0023s0012.nii.gz
bfa0003s0012.json    bfa0006s0012.json    bfa0009s0012.json
bfa0012s0012.json    bfa0015s0012.json    bfa0018s0012.json
bfa0021s0012.json    bfa0024s0012.json
bfa0003s0012.nii.gz  bfa0006s0012.nii.gz  bfa0009s0012.nii.gz
bfa0012s0012.nii.gz  bfa0015s0012.nii.gz  bfa0018s0012.nii.gz
bfa0021s0012.nii.gz  bfa0024s0012.nii.gz
```

DTI

La conversion es similar.

Se usa un script parecido,

update_dti.pl

```
#!/usr/bin/perl

use strict; use warnings;
use NEURO qw(load_study);
use File::Find::Rule;
use Data::Dump qw(dump);

my $study = "bioface";
my %std = load_study($study);
#unless ($itype) {$itype='t1';}
#my %dtype = ("t1" => "mri", );
my $src_dir = '/nas/bioface_raw';
#my $raw_dir = $std{'DATA'}.'/raw';
my $conv_file = $std{'DATA'}.'/converted_dti.csv';
my $ids_file = $std{'DATA'}.'/ids.csv';
my $names_file =$std{'DATA'}.'/ni_names.csv';
```

```
my $proj_file = $std{'DATA'}.'/bioface.csv';
my @other_exts = ("nii.gz","bval","bvec","json");
my @b0_exts = ("nii.gz","json");
my $tag = "DTIep2d_diff_mddw_48dir_p3_AP";
my $b0_tag = "DTIep2d_diff_mddw_4b0_PA";
#Leo la DB
my %idsinfo;
open IDS, "<$ids_file" or die $!;
while(<IDS>){
        chomp;
        my ($key, $value) = split(/,\s?/);
        $idsinfo{$key} = $value;
}
close IDS;
#Leo los que ya he descomprimido
my @orig_str = qx/ls $src_dir/;
chomp @orig_str;
#Leo los que se han convertido
my @conv_str;
open CONV, "<$conv_file" or die $!;
chomp (@conv_str = <CONV>);
close CONV;
#Y a ver cuantos faltan por convertir!
my %conv_str = map { $_ => 1 } @conv_str;
my @not_conv = grep !$conv_str{$_}, @orig_str;
# y a convertirlos!
foreach my $guy (@not_conv){
        if(exists($idsinfo{$guy})){
#DTIep2d_diff_mddw_48dir_p3_AP
                my $order = 'dcm2niix -z i -b y -o
'.$std{'DATA'}.'/tmp/ '.$src_dir.'/'.$guy;
                print "$order\n";
                system($order);
                my @conv_files = find(file => 'name' =>
"*$tag*.nii.gz", in => $std{'DATA'}.'/tmp/');
                #dump @conv_files;
                foreach my $nii_file (@conv_files){
                        $order = 'fslinfo '.$nii_file;
                        my %xinfo;
                        foreach (qx/$order/){
                                my ( $key, $value ) =
/(\S+)\s+(\S+)\s*.*/;
                                $xinfo{$key} =$value;
                        }
                        if($xinfo{"dim4"}>1){
                                print "Choosing and moving files\n";
                                my $file_name = $nii_file;
                                foreach my $ext (@other_exts){
                                        ($file_name = $nii_file) =~
s/nii.gz$/$ext/;
                                        if (-e $file_name) {
```

```
                                                $order = sprintf "mv
\"%s\" %s/dti/bfa%04ds0020.%s", $file_name, $std{'DATA'},
$idsinfo{$guy}, $ext;
                                                print "$order\n";
                                                system($order);
                                        }
                                }
                        }
                }
                   print "Getting P>>A B0\n";
#DTIep2d_diff_mddw_4b0_PA
                @conv_files = find(file => 'name' =>
"*$b0_tag*.nii.gz", in => $std{'DATA'}.'/tmp/');
                foreach my $nii_file (@conv_files){
                        foreach my $ext (@b0_exts){
                                (my $file_name = $nii_file) =~
s/nii.gz$/$ext/;
                                $order = sprintf "mv \"%s\"
%s/dti/bfa%04d_p2a_b0.%s", $file_name, $std{'DATA'}, $idsinfo{$guy},
$ext;
                                print "$order\n";
                                system($order);
                        }
                }
                print "Cleaning house\n";
                $order = "rm $std{'DATA'}/tmp/*";
                system($order);
        }
}


#actualizo los convertidos
push(@conv_str, @not_conv);
open CONV, ">$conv_file" or die $!;
print CONV "$_\n" foreach @conv_str;
close CONV;
```

~~Los DTI que se obtienen no son correctos. De los 22 obtenidos hay solo 5 que tienen los *bval* y *bvec*. Algun problema hay con el protocolo.~~ Esto era un problema del dcm2niix. He actualizado el dcm2niix y ahora todo procesa OK.

Para futuras actualizaciones:

```
$ git clone -b development https://github.com/rordenlab/dcm2niix.git
$ cd dcm2niix
$ mkdir build
$ cd build/
$ cmake
$ cp dcm2niix /nas/software/bin/
```

```
$ cp /nas/software/bin/dcm2niix /usr/local/mricron/
$ dcm2niix
Chris Rorden's dcm2niiX version v1.0.20191031  GCC4.4.7 (64-bit Linux)
usage: dcm2niix [options] <in_folder>
......
```

ASL

Para convertir los ASL.

update_asl.pl

```
#!/usr/bin/perl

use strict; use warnings;
use NEURO qw(load_study);
use Data::Dump qw(dump);
use File::Find::Rule;

my $study = "bioface";
my %std = load_study($study);
#unless ($itype) {$itype='t1';}
#my %dtype = ("t1" => "mri", );
my $src_dir = '/nas/bioface_raw';
#my $raw_dir = $std{'DATA'}.'/raw';
my $conv_file = $std{'DATA'}.'/converted_asl.csv';
my $ids_file = $std{'DATA'}.'/ids.csv';
my $names_file =$std{'DATA'}.'/ni_names.csv';
my $proj_file = $std{'DATA'}.'/bioface.csv';
my @other_exts = ("nii.gz","json");
my $tag = "asl_3d_tra_iso";
my $series = 12;
#Leo la DB
my %idsinfo;
open IDS, "<$ids_file" or die $!;
while(<IDS>){
        chomp;
        my ($key, $value) = split(/,\s?/);
        $idsinfo{$key} = $value;
}
close IDS;
#Leo los que ya he descomprimido
my @orig_str = qx/ls $src_dir/;
chomp @orig_str;
#Leo los que se han convertido
my @conv_str;
open CONV, "<$conv_file" or die $!;
chomp (@conv_str = <CONV>);
```

```
close CONV;
#Y a ver cuantos faltan por convertir!
my %conv_str = map { $_ => 1 } @conv_str;
my @not_conv = grep !$conv_str{$_}, @orig_str;
# y a convertirlos!
foreach my $guy (@not_conv){
        if(exists($idsinfo{$guy})){
                my $order = 'dcm2niix -z i -b y -o
'.$std{'DATA'}.'/tmp/ '.$src_dir.'/'.$guy;
                print "$order\n";
                system($order);
                my @conv_files = find(file => 'name' =>
"*$tag*.nii.gz", in => $std{'DATA'}.'/tmp/');
                my $count = $series;
                foreach my $nii_file (@conv_files){
                        $order = 'fslval '.$nii_file.' dim4';
                        my $xinfo = qx/$order/;
                        if ($xinfo > 1){
                                my $file_name = $nii_file;
                                foreach my $ext (@other_exts){
                                        ($file_name = $nii_file) =~
s/nii.gz$/$ext/;
                                        $order = sprintf "mv \"%s\"
%s/asl/bfa%04ds%04d.%s", $file_name, $std{'DATA'}, $idsinfo{$guy},
$count, $ext;
                                        print "$order\n";
                                        system($order);
                                }
                        }
                $count++;
                }
                print "Cleaning house\n";
                $order = "rm $std{'DATA'}/tmp/*";
                system($order);
        }
}
#actualizo los convertidos
push(@conv_str, @not_conv);
open CONV, ">$conv_file" or die $!;
print CONV "$_\n" foreach @conv_str;
close CONV;
```

Ver https://asl-docs.readthedocs.io/en/latest/oxasl_userguide.html

Informes

Tengo los informes de las MRI en pdf asi que voy a intentar sacar los datos de atrofia de ahi.

Primero a convertir los PDF en algo util,

```
[osotolongo@brick03 informes]$ pwd
/nas/data/bioface/informes
[osotolongo@brick03 informes]$ mkdir txts
[osotolongo@brick03 informes]$ for x in *.pdf; do pdftotext ${x}
txts/${x%.pdf}.txt; done
```

Ahora quiero sacar las atrofias que estan informadas estructuradamente,

```
[osotolongo@brick03 informes]$ grep "^Atrofia" txts/B001-20180611.txt
Atrofia global grado 1
Atrofia parietal grado (D/I) 1/1
Atrofia temporal medial grado (D/I) 0/0
```

Me hare un parser en Perl entonces,

parse0.pl

```
#!/usr/bin/perl

use strict;
use warnings;
use File::Find::Rule;
use File::Basename qw(basename);
use Data::Dump qw(dump);

my %rois = ('global' => 'AG',
    ('parietal' => 'AP'),
    ('tempo' => 'ATM')
);
my $parse_dir = shift;
my @txts = find(file => 'name' => "*.txt", in => $parse_dir);
my %info;
foreach my $report (sort @txts){
    my $name = basename $report;
    $name =~ s/\.txt$//;
    foreach my $roi (sort keys %rois){
        unless($rois{$roi} eq 'AG'){
            $info{$name}{$rois{$roi}.'_I'} = "NA";
            $info{$name}{$rois{$roi}.'_D'} = "NA";
        }else{
            $info{$name}{$rois{$roi}} = "NA";
        }
    }
    open IDF, "<$report";
    my $this_line;
    while(<IDF>) {
    if (/^Atrofia/){
        foreach my $roi (sort keys %rois){
            if (/$roi/){
```

```
                if (/\(D\/I\)/){
                    my ($gd,$gi) = /\(D\/I\).*(\d)\/(\d)/;
                    $info{$name}{$rois{$roi}.'_I'} = $gi if defined
$gi;
                    $info{$name}{$rois{$roi}.'_D'} = $gd if defined
$gd;
                }else{
                    (my $gb) = /grado.*\d*-*(\d)/;
                    $info{$name}{$rois{$roi}} = $gb if defined $gb;
                }
            }
        }
    }
    }
    close IDF;
}

print "Subject";
foreach my $roi (sort keys %rois){
    my $ch;
    unless ($rois{$roi} eq 'AG'){
        $ch = ','.$rois{$roi}.'_I,'.$rois{$roi}.'_D';
    }else{
        $ch = ','.$rois{$roi};
    }
    print "$ch";
}
print "\n";
foreach my $subject (sort keys %info){
    print "$subject";
    foreach my $roi (sort keys %rois){
        unless ($rois{$roi} eq 'AG'){
            my $ad =
exists($info{$subject}{$rois{$roi}.'_D'})?$info{$subject}{$rois{$roi}.'
_D'}:"NA";
            my $ai =
exists($info{$subject}{$rois{$roi}.'_I'})?$info{$subject}{$rois{$roi}.'
_I'}:"NA";
            print ",$ai,$ad";
        }else{
            my $ag =
exists($info{$subject}{$rois{$roi}})?$info{$subject}{$rois{$roi}}:"NA";
            print ",$ag";
        }

    }
    print "\n";
}
```

y le digo que me haga el spider donde estan los *txt*,

```
[osotolongo@brick03 informes]$ ./parse0.pl txts/ > parsed_reports.csv
[osotolongo@brick03 informes]$ head parsed_reports.csv
Subject,AG,AP_I,AP_D,ATM_I,ATM_D
B001-20180611,1,1,1,0,0
B002-20170581,1,1,1,0,0
B003-20180364,NA,1,0,1,0
B004-20141541,0,NA,NA,NA,NA
B005-20180246,1,1,1,0,0
B006-20131186,0,NA,NA,NA,NA
B007-20161829,1,1,0,0,0
B008-20180298,0,0,0,0,0
B009-20171174,1,0,1,0,0
```

No saca todos pero si bastantes. 😎

Recepcion V2

Aqui tenemos las imagenes nuevas,

```
[root@brick03 corachan]# ls BIOFACE_V2
B014  B035  B038  B048  B053  B056  B058  B060  B063  B069  B072  B076  B084
B106
B030  B037  B039  B050  B054  B057  B059  B061  B068  B070  B074  B078  B105
```

Asi que las subimos al XNAT,

```
[osotolongo@brick03 bioface]$ for x in $(ls /nas/corachan/BIOFACE_V2); do
xnatapic upload_dicom --project_id bioface21 --subject_id ${x} --pipelines
/nas/corachan/BIOFACE_V2/${x}; done
```

sacamos la lista de experimentos y los enviamos al pipeline,

```
[osotolongo@brick03 bioface]$ for x in $(ls /nas/corachan/BIOFACE_V2); do
xnatapic list_experiments --project_id bioface21 --subject_id ${x} --
modality MRI; done >> experiments.list
[osotolongo@brick03 bioface]$ for x in $(cat experiments.list); do xnatapic
run_pipeline --project_id bioface21 --pipeline RunFreesurfer --experiment_id
${x}; done
[osotolongo@brick03 bioface]$ queue | grep XNAT
  163807      fast          RunFreesurfer.XNAT5_E01331     xnat PD
0:00      1 (Resources)
  163808      fast          RunFreesurfer.XNAT5_E01332     xnat PD
0:00      1 (Priority)
  163809      fast          RunFreesurfer.XNAT5_E01333     xnat PD
0:00      1 (Priority)
  163810      fast          RunFreesurfer.XNAT5_E01334     xnat PD
0:00      1 (Priority)
  163811      fast          RunFreesurfer.XNAT5_E01335     xnat PD
```

```
0:00      1 (Priority)
  163812      fast           RunFreesurfer.XNAT5_E01336     xnat PD
0:00      1 (Priority)
  163786      fast           RunFreesurfer.XNAT5_E01310     xnat  R
1:41      1 brick02
  163787      fast           RunFreesurfer.XNAT5_E01311     xnat  R
1:41      1 brick02
  163788      fast           RunFreesurfer.XNAT5_E01312     xnat  R
1:41      1 brick05
  163789      fast           RunFreesurfer.XNAT5_E01313     xnat  R
1:41      1 brick05
  163790      fast           RunFreesurfer.XNAT5_E01314     xnat  R
1:38      1 brick05
  163791      fast           RunFreesurfer.XNAT5_E01315     xnat  R
1:38      1 brick05
  163792      fast           RunFreesurfer.XNAT5_E01316     xnat  R
1:38      1 brick05
  163793      fast           RunFreesurfer.XNAT5_E01317     xnat  R
1:38      1 brick05
  163794      fast           RunFreesurfer.XNAT5_E01318     xnat  R
1:38      1 brick04
  163795      fast           RunFreesurfer.XNAT5_E01319     xnat  R
1:38      1 brick04
  163796      fast           RunFreesurfer.XNAT5_E01320     xnat  R
1:38      1 brick04
  163797      fast           RunFreesurfer.XNAT5_E01321     xnat  R
1:38      1 brick04
  163798      fast           RunFreesurfer.XNAT5_E01322     xnat  R
1:38      1 brick04
  163799      fast           RunFreesurfer.XNAT5_E01323     xnat  R
1:38      1 brick04
  163800      fast           RunFreesurfer.XNAT5_E01324     xnat  R
1:35      1 brick04
  163801      fast           RunFreesurfer.XNAT5_E01325     xnat  R
1:35      1 brick04
  163802      fast           RunFreesurfer.XNAT5_E01326     xnat  R
1:35      1 brick03
  163803      fast           RunFreesurfer.XNAT5_E01327     xnat  R
1:35      1 brick03
  163804      fast           RunFreesurfer.XNAT5_E01328     xnat  R
1:35      1 brick03
  163805      fast           RunFreesurfer.XNAT5_E01329     xnat  R
1:35      1 brick03
  163806      fast           RunFreesurfer.XNAT5_E01330     xnat  R
1:35      1 brick03
```

comprobando las MRI

Saco la columna de fecha de las mri y el codigo del sujeto del excel y lo savlo, luego lo convierto a algo potable,

```
[osotolongo@brick03 bioface]$ ./xls2csv.pl v2_mris.xlsx | grep -iv
"no\|dropout\|ensayo" | awk -F"," '{if($1 && $2) print}' | sed 's/,$//' >
citados.csv
```

y lo filtro segun la lista de los subidos,

```
[osotolongo@brick03 bioface]$ grep -v "`ls /nas/corachan/BIOFACE_V2/`"
citados.csv
Sujeto ,RM
B009,2021-06-18
B021,2021-06-15
B055,2021-07-14
B094,2021-07-23
```