Extrayendo los datos de los informes de MRI FACE

Obteniendo los PatientID

Lo primero es sacar los *Patient ID* de cada MRI. Primero saco los sujetos del proyecto y los asocio a los *IDs* almacenados como *label* del experimento.

```
[osotolongo@brick03 mri_face]$ xnatapic list_subjects --project_id unidad --
label > xnat_subjects.list
[osotolongo@brick03 mri_face]$ while read -r line; do sbj=$(echo ${line} |
awk -F"," '{print $1}'); slbl=$(echo ${line} | awk -F"," '{print $2}');
xpr=$(xnatapic list_experiments --project_id unidad --subject_id ${sbj} --
modality MRI --label); echo "${slbl},${xpr}"; done < xnat_subjects.list |
awk -F"," '{print $1","$3}' > sbj_ids.csv
[osotolongo@brick03 mri_face]$ head sbj_ids.csv
20211269,D18290542
20211480,D21572561
20211176,D14074193
20151338,D21587192
20211523,D21587226
20211401,D21587147
20211281,D21580074
20210716,D21587448
20081210,D21582920
20211482,D21583488
```

Bajar PDFs

1. Entrar en https://impax.corachan.com/
2. Buscar por PatientID
3. Salvar el report como pdf con el numero interno

Nota:

Si vas sacando y te pierdes siempre pues hacer, de vez en cuando,

```
[osotolongo@brick03 mri_face]$ ls reports/*.pdf | sed
's/reports\/\(.*\)\.pdf/\1/' > sbjs_done.txt
[osotolongo@brick03 mri_face]$ grep -v "^`cat sbjs_done.txt`" sbj_ids.csv >
sbj_ids_todo.csv
```

para limpiar la lista de los que faltan

Sacar info

Los convierto a *txt*,

```
[osotolongo@brick03 mri_face]$ for x in reports/*.pdf ; do pdftotext ${x};
done
[osotolongo@brick03 mri_face]$ ls reports/
20081210.pdf  20140947.pdf  20151338.pdf  20201297.pdf  20211384.pdf
20211455.pdf  20211475.pdf  20211480.pdf  20211505.pdf  20211523.pdf
20211524.pdf  20211527.pdf  20211611.pdf
20081210.txt  20140947.txt  20151338.txt  20201297.txt  20211384.txt
20211455.txt  20211475.txt  20211480.txt  20211505.txt  20211523.txt
20211524.txt  20211527.txt  20211611.txt
```

Y voy a hacer un parser rapidito a ver que sale,

parser0.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' => 'ACG',
        ('parietal' => 'Koedam'),
        ('tempor' => 'Scheltens'),
        ('frontal' => 'Kipps')
);
my $parse_dir = shift;
my @txts = find(file => 'name' => "*.txt", in => $parse_dir);
my %info;
dump @txts;
foreach my $report (sort @txts){
        my $name = basename $report;
        $name =~ s/\.txt$//;
        foreach my $roi (sort keys %rois){
                if($roi eq 'global'){
                        $info{$name}{$rois{$roi}} = "NA";
                }elsif($roi eq 'frontal'){
                        $info{$name}{$rois{$roi}.'_F'} = "NA";
                        $info{$name}{$rois{$roi}.'_A'} = "NA";
                        $info{$name}{$rois{$roi}.'_P'} = "NA";
                }else{
```

```
                        $info{$name}{$rois{$roi}.'_I'} = "NA";
                        $info{$name}{$rois{$roi}.'_D'} = "NA";
                }
        }
        open IDF, "<$report";
        my $this_line;
        while(<IDF>) {
                if (/^Atr/){
                        foreach my $roi (sort keys %rois){
                        if (/$roi/){
                                if (/gra.*\d\/\d\/\d/ and /frontal/){
                                        my ($fb, $ab, $pb) =
/gra.*(\d)\/(\d)\/(\d)/;
                                        $info{$name}{$rois{$roi}.'_F'}
= $fb if defined $fb;
                                        $info{$name}{$rois{$roi}.'_A'}
= $ab if defined $ab;
                                        $info{$name}{$rois{$roi}.'_P'}
= $pb if defined $pb;
                                }elsif (/\(D\/I\).*\d*-*\d\/\d*-*\d/){
                                        my ($gd,$gi) = /\(D\/I\).*\d*-
*(\d)\/\d*-*(\d)/;
                                        $info{$name}{$rois{$roi}.'_I'}
= $gi if defined $gi;
                                        $info{$name}{$rois{$roi}.'_D'}
= $gd if defined $gd;
                                }else{
                                        (my $gb) = /\d*-*(\d)/;
                                        unless (/global/) {
$info{$name}{$rois{$roi}.'_I'} = $gb if defined $gb;
$info{$name}{$rois{$roi}.'_D'} = $gb if defined $gb;
                                        }else{
$info{$name}{$rois{$roi}} = $gb if defined $gb;
                                        }
                                }
                        }
                        }
                }
        }
        close IDF;
}
print "Subject";
foreach my $roi (sort keys %rois){
        my $ch;
        if ($roi eq 'global'){
                $ch = ','.$rois{$roi};
        }elsif($roi eq 'frontal'){
                $ch =
','.$rois{$roi}.'_F,'.$rois{$roi}.'_A,'.$rois{$roi}.'_P';
        }else{
                $ch = ','.$rois{$roi}.'_I,'.$rois{$roi}.'_D';
```

```
        }
        print "$ch";
}
print "\n";
foreach my $subject (sort keys %info){
        print "$subject";
        foreach my $roi (sort keys %rois){
                if ($roi eq 'global'){
                        my $ag =
exists($info{$subject}{$rois{$roi}})?$info{$subject}{$rois{$roi}}:"NA";
                        print ",$ag";
                }elsif ($roi eq 'frontal'){
                        my $af =
exists($info{$subject}{$rois{$roi}.'_F'})?$info{$subject}{$rois{$roi}.'
_F'}:"NA";
                        my $aa =
exists($info{$subject}{$rois{$roi}.'_A'})?$info{$subject}{$rois{$roi}.'
_A'}:"NA";
                        my $ap =
exists($info{$subject}{$rois{$roi}.'_P'})?$info{$subject}{$rois{$roi}.'
_P'}:"NA";
                        print ",$af,$aa,$ap";
                }else{
                        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";
                }
        }
        print "\n";
}
```

Procedimiento antiguo

y entonces,

```
[osotolongo@brick03 mri_face]$ ./parse0.pl reports
Subject,Kipps_F,Kipps_A,Kipps_P,ACG,Koedam_I,Koedam_D,Scheltens_I,Scheltens_
D
20081210,NA,NA,NA,1,1,1,0,0
20140947,1,1,1,2,2,2,0,0
20151338,1,1,1,2,2,2,2,2
20201297,1,1,2,2,2,2,2,2
20211384,NA,NA,NA,1,1,1,3,3
20211455,2,2,2,1,2,2,3,3
```

```
20211475,NA,NA,NA,1,1,1,1,1
20211480,2,1,2,2,2,2,1,1
20211505,NA,NA,NA,1,1,1,1,2
20211523,NA,NA,NA,1,1,1,2,2
20211524,NA,NA,NA,2,2,2,1,0
20211527,2,1,1,1,2,2,1,1
20211611,NA,NA,NA,0,0,0,0,0
```

Delete!

Y cuando casi eres feliz, **te cambian la estructura de los informes** 😆

Ahora la estructura es mas parseable pero ojo, hay que incluir tambien los infromes con el formato antiguo. **¿A que la vida es maravillosa?**

Entonces lo que he hecho es rehacer toda la logica desde el principio

parser1.pl

```
#!/usr/bin/perl

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

my @rinf = ('GCA_D', 'GCA_I',
        'Koedam_D', 'Koedam_I',
        'Kipps_F_D', 'Kipps_F_I',
        'Kipps_A_D', 'Kipps_A_I',
        'Kipps_P_D', 'Kipps_P_I',
        'Scheltens_D', 'Scheltens_I',
        'Fazekas'
);

my $project = 'unidad';
my $ofile = 'reports_data.xls';
my $parse_dir = shift;
my @txts = find(file => 'name' => "*.txt", in => $parse_dir);
my %info;
#dump @txts;
foreach my $report (sort @txts){
        my $name = basename $report;
        $name =~ s/\.txt$//;
        foreach my $iinf (@rinf){
                if($iinf eq 'Fazekas'){
                        $info{$name}{$iinf} = 0;
                }else{
```

```
                        $info{$name}{$iinf} = "NA";
                }
        }
        open IDF, "<$report";
        my $this_line;
        while(<IDF>) {
                if (/Fazekas/i){
                        my ($fv) = /Fazekas\s*\d*-*(\d)/i;
                        $info{$name}{'Fazekas'} = $fv if defined $fv;
                }
                if (/.*GCA.*/ and not /^Temporal/i){
                        my ($ad, $ai) = /.*\d*-*(\d)\s*\/*.*\d*-
*(\d).*GCA.*/;
                        $info{$name}{'GCA_D'} = $ad if defined $ad;
                        $info{$name}{'GCA_I'} = $ai if defined $ai;
                }elsif(/Kipps/){
                        if(/frontal/i and /anterior/i and
/posterior/i){
                                my ($af, $aa, $ap) = /.*\d*-
*(\d)\/*\d*-*(\d)\/\d*-*(\d).*/;
                                $info{$name}{'Kipps_F_D'} = $af if
defined $af;
                                $info{$name}{'Kipps_F_I'} = $af if
defined $af;
                                $info{$name}{'Kipps_A_D'} = $aa if
defined $aa;
                                $info{$name}{'Kipps_A_I'} = $aa if
defined $aa;
                                $info{$name}{'Kipps_P_D'} = $ap if
defined $ap;
                                $info{$name}{'Kipps_P_I'} = $ap if
defined $ap;
                        }else{
                                my ($ad, $ai) = /.*\d*-
*(\d)\s*\/*.*\d*-*(\d).*Kipps.*/i;
                                if(/Frontal/i){
                                        $info{$name}{'Kipps_F_D'} = $ad
if defined $ad;
                                        $info{$name}{'Kipps_F_I'} = $ai
if defined $ai;
                                }elsif(/anterior/i){
                                        $info{$name}{'Kipps_A_D'} = $ad
if defined $ad;
                                        $info{$name}{'Kipps_A_I'} = $ai
if defined $ai;
                                }elsif(/posterior/i){
                                        $info{$name}{'Kipps_P_D'} = $ad
if defined $ad;
                                        $info{$name}{'Kipps_P_I'} = $ai
if defined $ai;
```

```
                                }
                        }
                }elsif(/Koedam/i){
                        if(/.*\d\s*\/*.*\d*-*\d.*/){
                                my ($ad, $ai) = /.*\d*-
*(\d)\s*\/*.*\d*-*(\d).*Koedam.*/i;
                                $info{$name}{'Koedam_D'} = $ad if
defined $ad;
                                $info{$name}{'Koedam_I'} = $ai if
defined $ai;
                        }else{
                                my ($ag) = /.*\d*-*(\d).*/;
                                $info{$name}{'Koedam_D'} = $ag if
defined $ag;
                                $info{$name}{'Koedam_I'} = $ag if
defined $ag;
                        }
                }elsif(/Scheltens/i){
                        if(/bilateral/i){
                                my ($ag) = /.*\d*-*(\d).*/;
                                $info{$name}{'Scheltens_D'} = $ag if
defined $ag;
                                $info{$name}{'Scheltens_I'} = $ag if
defined $ag;
                        }else{
                                if(/.*\d.*Scheltens.*/i){
                                        my ($ad, $ai) = /.*\d*-
*(\d)\s*\/*.*\d*-*(\d).*Scheltens.*/i;
                                        $info{$name}{'Scheltens_D'} =
$ad if defined $ad;
                                        $info{$name}{'Scheltens_I'} =
$ai if defined $ai;
                                }else{
                                        if(/.*\d*-*\d.*\d*-*\d.*/){
                                                my ($ad, $ai) =
/.*Scheltens.*\d*-*(\d).*\d*-*(\d).*/i;
$info{$name}{'Scheltens_D'} = $ad if defined $ad;
$info{$name}{'Scheltens_I'} = $ai if defined $ai;
                                        }else{
                                                my ($ag) =
/.*Scheltens.*\d*-*(\d)/i;
$info{$name}{'Scheltens_D'} = $ag if defined $ag;
$info{$name}{'Scheltens_I'} = $ag if defined $ag;
                                        }
                                }
                        }
                }elsif(/global/i and /cortical/i){
                        my ($ab) = /.*\d*-*(\d).*/;
                        $info{$name}{'GCA_D'} = $ab if defined $ab;
                        $info{$name}{'GCA_I'} = $ab if defined $ab;
                }elsif(/tempor/i and /mesial/i){
```

```
                        if(/bilateral/i){
                                my ($ag) = /.*\d*-*(\d).*/;
                                $info{$name}{'Scheltens_D'} = $ag if
defined $ag;
                                $info{$name}{'Scheltens_I'} = $ag if
defined $ag;
                        }else{
                                my ($ad, $ai) = /.*\d*-*(\d).*\d*-
*(\d).*/;
                                $info{$name}{'Scheltens_D'} = $ad if
defined $ad;
                                $info{$name}{'Scheltens_I'} = $ai if
defined $ai;
                        }
                }
        }
        close IDF;
}
################################################
# Este trozo de aqui es para sacar las fechas #
################################################
foreach my $subject (sort keys %info){
        my $xorder = 'xnatapic list_experiments --project_id
'.$project.' --subject_id '.$subject.' --label --date';
        my ($xdata) = qx/$xorder/;
        $xdata =~ s/.*,(.*),.*/$1/;
        chomp $xdata;
        $info{$subject}{'date'} = $xdata;
}
################################################
################################################
################################################
# Ahora, en lugar de ir imprimiendo
# voy a armar las filas del output
my @rows;
$rows[0] = "Subject,Date";
foreach my $iinf (sort @rinf){
        $rows[0] .= ",$iinf";
}
my $count = 1;
foreach my $subject (sort keys %info){
        $rows[$count] = "$subject,$info{$subject}{'date'}";
        foreach my $iinf (sort @rinf){
                $rows[$count] .= ",$info{$subject}{$iinf}";
        }
        $count++;
}
# y ahora intento escribir la filas en un xls
my $workbook = Spreadsheet::Write->new(file => $ofile, sheet =>
'DATA');
```

```
foreach my $row (@rows){
        # Fabrico un array temporal porque es lo que entiende el modulo
        my @arow = split /,/,$row;
        $workbook->addrow(@arow);
        # y tambien saco la fila por STDOUT por si quiero generar un
CSV
        # hacer un grep o cualquier otro tipo de postproc
        print "$row\n";
}
$workbook -> close();
```

y entonces ejecutando algo como,

```
$ ./parse1.pl reports/ > mri_face_reports.csv
```

Obtengo los valores en un *xls* y en un *csv*,

```
Subject,Date,Fazekas,GCA_D,GCA_I,Kipps_A_D,Kipps_A_I,Kipps_F_D,Kipps_F_I,Kip
ps_P_D,Kipps_P_I,Koedam_D,Koedam_I,Scheltens_D,Scheltens_I
20050456,2022-03-06,3,3,3,2,2,2,3,2,2,1,3,2,2
20081210,2021-12-02,1,1,1,NA,NA,NA,NA,NA,NA,1,1,0,0
20090461,2022-01-28,1,1,1,1,2,1,1,1,2,0,3,2,2
20100678,2022-01-16,0,2,2,2,2,2,2,2,2,2,3,3,3
20140947,2021-12-11,1,2,2,1,1,1,1,1,1,2,2,0,0
20150926,2022-01-26,2,2,2,2,2,2,2,3,3,2,3,2,3
20151338,2021-12-04,1,2,2,1,1,1,1,1,1,2,2,2,2
20160418,2021-12-11,3,2,2,2,2,2,2,2,2,1,1,3,3
20170735,2021-12-22,2,1,1,1,1,0,0,1,1,1,1,1,1
```

Nota: Se han tomado en cuenta algunas consideraciones extra, por peticion popular,

1. Si no está *Fazekas* ponemos 0
2. Añadimos la fecha de la MRI a la tabla

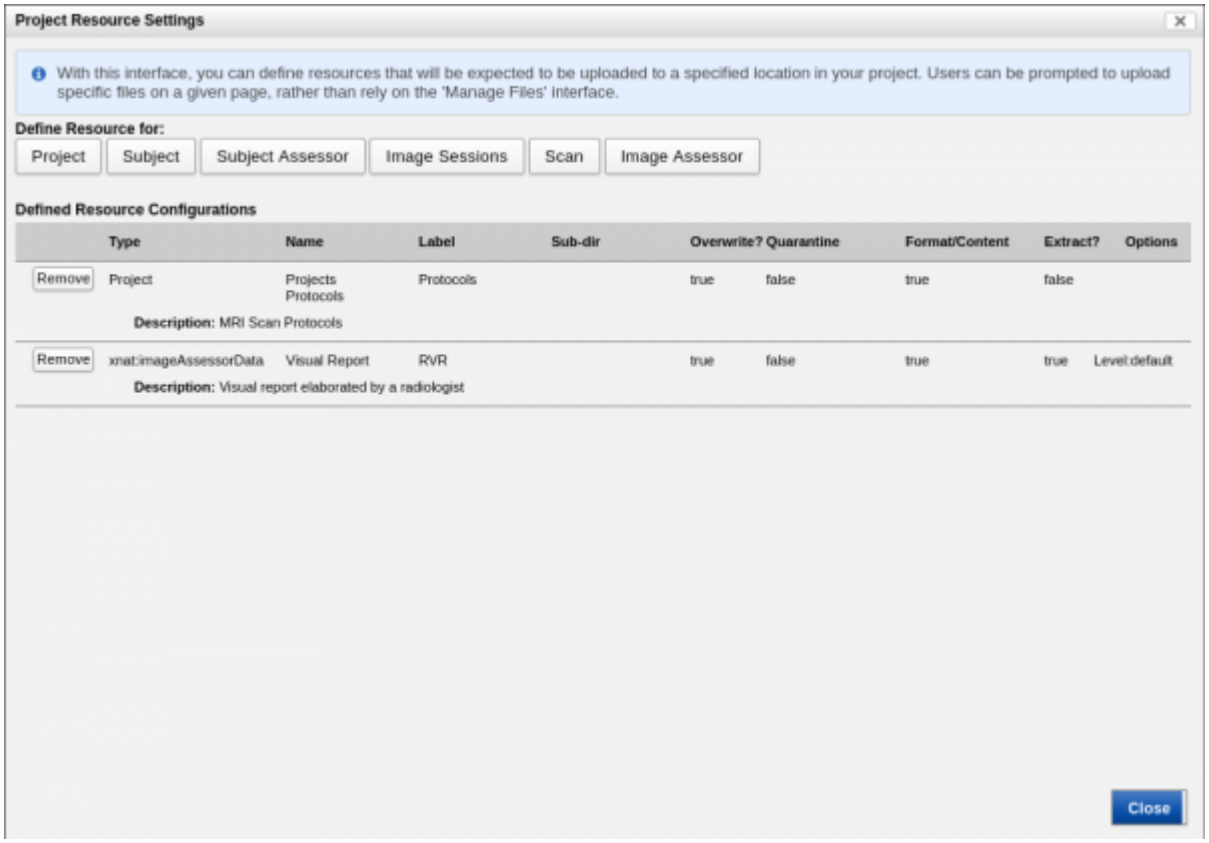
Guardando todo en XNAT

La logica es la siguiente. Este proyecto continuara hasta alcanzar numeros grandes. Si corro el parser cada vez sobre toda la muestra puede llegar a demorar y va ser difcil de revisar.

Asi que voy a guardar todo lo que pueda en XNAT para almacenarlo y podre ir haciendolo a trozos después.

Informes

Lo primero que todo es guardar los informes en PDF para que siempre esten listos para consulta. Para ello creamos un *Resource* en XNAT para el *visual report*,



y entonces subimos el informe con algo asi,

```
$ curl -f -X PUT -u "user:password"
"http://detritus.fundacioace.com:8088/data/experiments/XNAT4_EXXXX/resources
/RVR/files/report_XUSER.pdf?overwrite=true" -F
file="@/path/to/report/XXXXX.pdf"
```

Agregando datos del informe

Ahora quiero guardar los datos extra de cada informe como otro *resource*. Digamos que tengo un CSV de este estilo,

```
Subject,ATM_I,ATM_D,ACG,Fazekas
F001,NA,NA,NA,1
F002,2,2,NA,1
F003,0,0,NA,1
F005,NA,0,NA,1
F006,1,0,NA,1
F007,0,1,NA,0
F008,1,0,NA,NA
F009,0,0,NA,2
F010,0,NA,NA,1
```

pero los campos pueden variar. Lo que hago es importar el csv automagicamente con *Text::CSV* y

crear un archivo *json* para copiar en cada *RVR*.

Por partes, importo el csv,

```
my $ref_vr = csv(in => $vrfile, headers => "auto");
```

y creo el *json* en un archivo temporal con todos los campos,

```
     foreach my $mrdata (@$ref_vr){
                my $rep_body = '{"ResultSet":{"Result":[{';
                my @rep_arr;
                foreach my $rk (sort keys %$mrdata){
                        #if ($rk ne 'Subject' and $rk ne 'Date'){
                        push @rep_arr, '"'.$rk.'":"'.${$mrdata}{$rk}.'"';
                        #}
                }
                $rep_body .= join ',', @rep_arr;
                $rep_body .= '}]}}';
                my $tvrf = mktemp($tmp_dir.'/rvr_data.XXXXX');
                open TDF, ">$tvrf";
                print TDF "$rep_body\n";
                close TDF;
      }
```

Ahor para subir ese archivo temporal tengo que hacer algo como,

```
$ curl -f -X PUT -u "user:pass"
"connection_site/data/experiments/experimento_evaluado/resources/RVR/files/r
eport_data.json?overwrite=true" -F file="@archivo_temporal"
```

Asi que dentro del sitio he de hacer,

```
                my $xcurl = 'curl -f -X PUT -u
"'.$xconf{'USER'}.':'.$xconf{'PASSWORD'}.'"
"'.$xconf{'HOST'}.'/data/experiments/'.$vrdata{${$mrdata}{'Subject'}}.'/reso
urces/RVR/files/report_data.json?overwrite=true" -F file="@'.$tvrf.'"';
                print "$xcurl\n";
                system($xcurl);
                unlink $tvrf;
```

Subiendo todo

mirar sesiones aqui:
https://wiki.xnat.org/documentation/how-to-use-xnat/generating-and-reusing-a-jsession-id-for-scripted-interactions

Lo que hago es unir todo esto en un solo script Perl

xnat_up_rvr.pl

```
#!/usr/bin/perl
# Put the Visual readings into XNAT resources
# This is made for MRIFACE project but is intended to be use everywhere
# if lucky!
#
# Copyleft 2022 O. Sotolongo <asqwerty@gmail.com>
#
use strict;
use warnings;
use File::Find::Rule;
use File::Basename qw(basename);
use Text::CSV qw(csv);
use File::Temp qw( :mktemp tempdir);
use Data::Dump qw(dump);
my $vrfile;
my $rep_dir;
my $xprj;
while (@ARGV and $ARGV[0] =~ /^-/) {
    $_ = shift;
    last if /^--$/;
    if (/^-i/) {$vrfile = shift; chomp($vrfile);}
    if (/^-d/) {$rep_dir = shift; chomp($rep_dir);}
    if (/^-x/) {$xprj = shift; chomp($xprj);}
}
die "Should supply reports directory" unless $rep_dir;
die "Should supply XNAT project" unless $xprj;
my $xconf_file = $ENV{'HOME'}.'/.xnatapic/xnat.conf';
my %xconf;
open IDF, "<$xconf_file";
while (<IDF>){
        if (/^#.*/ or /^\s*$/) { next; }
        my ($n, $v) = /(.*)=(.*)/;
        $xconf{$n} = $v;
}
#dump %xconf;
close IDF;
my $tmp_dir = $ENV{TMPDIR};
my %vrdata;
my %rdata;
# get the session ID
my $q = 'curl -f -X POST -u "'.$xconf{'USER'}.':'.$xconf{'PASSWORD'}.'"
"'.$xconf{'HOST'}.'/data/JSESSION"';
my $jid = qx/$q/;
my @pdfs = find(file => 'name' => "*.pdf", in => $rep_dir);
foreach my $report (@pdfs) {
        my $bnreport = basename $report;
        (my $xsbj = $bnreport) =~ s/.pdf//;
        my $xorder = 'xnatapic list_experiments --project_id '.$xprj.'
--subject_id '.$xsbj.' --modality MRI';
```

```
        my ($xdata) = qx/$xorder/;
        chomp $xdata;
        $xdata =~ s/"//g;
        $vrdata{$xsbj} = $xdata;
        my $xcurl = 'curl -f -X PUT -b JSESSIONID='.$jid.'
"'.$xconf{'HOST'}.'/data/experiments/'.$xdata.'/resources/RVR"
2>/dev/null';
        system($xcurl);
        $xcurl = 'curl -f -X PUT -b JSESSIONID='.$jid.'
"'.$xconf{'HOST'}.'/data/experiments/'.$xdata.'/resources/RVR/files/rep
ort_'.$xsbj.'.pdf?overwrite=true" -F file="@'.$report.'"';
        print "$xcurl\n";
        system($xcurl);
}
if ($vrfile) {
        my $ref_vr = csv(in => $vrfile, headers => "auto");
        foreach my $mrdata (@$ref_vr){
                my $rep_body = '{"ResultSet":{"Result":[{';
                my @rep_arr;
                foreach my $rk (sort keys %$mrdata){
                        #if ($rk ne 'Subject' and $rk ne 'Date'){
                        push @rep_arr,
'"'.$rk.'":"'.${$mrdata}{$rk}.'"';
                        #}
                }
                $rep_body .= join ',', @rep_arr;
                $rep_body .= '}]}}';
                my $tvrf = mktemp($tmp_dir.'/rvr_data.XXXXX');
                open TDF, ">$tvrf";
                print TDF "$rep_body\n";
                close TDF;
                my $xcurl = 'curl -f -X PUT -b JSESSIONID='.$jid.'
"'.$xconf{'HOST'}.'/data/experiments/'.$vrdata{${$mrdata}{'Subject'}}.'
/resources/RVR/files/report_data.json?overwrite=true" -F
file="@'.$tvrf.'"';
                print "$xcurl\n";
                system($xcurl);
                unlink $tvrf;
        }
}
```

que ejecuto como,

```
[osotolongo@brick03 mri_face]$ ./xnat_up_rvr.pl -i
/nas/osotolongo/parsing/atm_v2_rev_again.csv -d
/nas/osotolongo/parsing/facehbi_informes_mri/v2/pdfs/ -x f2cehbi
  % Total    % Received % Xferd  Average Speed   Time    Time     Time
Current
                                 Dload  Upload   Total   Spent    Left
```

```
Speed
100    32  100    32    0     0    307      0 --:--:-- --:--:-- --:--:--
310
curl -f -X PUT -b JSESSIONID=EAD2EA07CDEE33D3D5178C11FF4EF514
"http://detritus.fundacioace.com:8088/data/experiments/XNAT5_E00001/resource
s/RVR/files/report_F001.pdf?overwrite=true" -F
file="@/nas/osotolongo/parsing/facehbi_informes_mri/v2/pdfs/F001.pdf"
curl -f -X PUT -b JSESSIONID=EAD2EA07CDEE33D3D5178C11FF4EF514
"http://detritus.fundacioace.com:8088/data/experiments/XNAT5_E00087/resource
s/RVR/files/report_F002.pdf?overwrite=true" -F
file="@/nas/osotolongo/parsing/facehbi_informes_mri/v2/pdfs/F002.pdf"
curl -f -X PUT -b JSESSIONID=EAD2EA07CDEE33D3D5178C11FF4EF514
"http://detritus.fundacioace.com:8088/data/experiments/XNAT5_E00088/resource
s/RVR/files/report_F003.pdf?overwrite=true" -F
file="@/nas/osotolongo/parsing/facehbi_informes_mri/v2/pdfs/F003.pdf"
curl -f -X PUT -b JSESSIONID=EAD2EA07CDEE33D3D5178C11FF4EF514
"http://detritus.fundacioace.com:8088/data/experiments/XNAT5_E00089/resource
s/RVR/files/report_F005.pdf?overwrite=true" -F
file="@/nas/osotolongo/parsing/facehbi_informes_mri/v2/pdfs/F005.pdf"
...
...
...
curl -f -X PUT -b JSESSIONID=EAD2EA07CDEE33D3D5178C11FF4EF514
"http://detritus.fundacioace.com:8088/data/experiments/XNAT5_E00697/resource
s/RVR/files/report_data.json?overwrite=true" -F
file="@/old_nas/osotolongo/tmp/rvr_data.BgByl"
curl -f -X PUT -b JSESSIONID=EAD2EA07CDEE33D3D5178C11FF4EF514
"http://detritus.fundacioace.com:8088/data/experiments/XNAT5_E00698/resource
s/RVR/files/report_data.json?overwrite=true" -F
file="@/old_nas/osotolongo/tmp/rvr_data.SzT2s"
curl -f -X PUT -b JSESSIONID=EAD2EA07CDEE33D3D5178C11FF4EF514
"http://detritus.fundacioace.com:8088/data/experiments/XNAT5_E00699/resource
s/RVR/files/report_data.json?overwrite=true" -F
file="@/old_nas/osotolongo/tmp/rvr_data.vyeh8"
curl -f -X PUT -b JSESSIONID=EAD2EA07CDEE33D3D5178C11FF4EF514
"http://detritus.fundacioace.com:8088/data/experiments/XNAT5_E00700/resource
s/RVR/files/report_data.json?overwrite=true" -F
file="@/old_nas/osotolongo/tmp/rvr_data.zJvbo"
curl -f -X PUT -b JSESSIONID=EAD2EA07CDEE33D3D5178C11FF4EF514
"http://detritus.fundacioace.com:8088/data/experiments/XNAT5_E00701/resource
s/RVR/files/report_data.json?overwrite=true" -F
file="@/old_nas/osotolongo/tmp/rvr_data.55aWS"
curl -f -X PUT -b JSESSIONID=EAD2EA07CDEE33D3D5178C11FF4EF514
"http://detritus.fundacioace.com:8088/data/experiments/XNAT5_E00702/resource
s/RVR/files/report_data.json?overwrite=true" -F
file="@/old_nas/osotolongo/tmp/rvr_data.5m_vZ"
curl -f -X PUT -b JSESSIONID=EAD2EA07CDEE33D3D5178C11FF4EF514
"http://detritus.fundacioace.com:8088/data/experiments/XNAT5_E00703/resource
s/RVR/files/report_data.json?overwrite=true" -F
file="@/old_nas/osotolongo/tmp/rvr_data.KIa5m"
```

Nota: El script es lo suficientemente abstracto pra ser incluido en el pipeline!!!!!

Datos demograficos

Como he de sacar los datos demograficos de la DB de Ekon, el primer paso es extraer los ID de los sujetos del proyecto de XNAT, estos deben corresponder al *NHC* en la DB. Esta parte es sencilla,

```
[osotolongo@brick03 mri_face]$ curl -f -X GET  -u "user:pass"
"http://detritus.fundacioace.com:8088/data/projects/unidad/subjects?format=c
sv" > all_subjects.csv
  % Total    % Received % Xferd  Average Speed   Time    Time     Time
Current
                                 Dload  Upload   Total   Spent    Left
Speed
100 10597    0 10597    0     0  56014      0 --:--:-- --:--:-- --:--:--
56367

[osotolongo@brick03 mri_face]$ head all_subjects.csv
ID,project,label,insert_date,insert_user,URI
XNAT05_S00035,unidad,20211269,2021-12-18
20:49:07.368,osotolongo,/data/subjects/XNAT05_S00035
XNAT05_S00042,unidad,20211480,2021-12-18
23:30:44.428,osotolongo,/data/subjects/XNAT05_S00042
XNAT05_S00062,unidad,20211176,2021-12-21
09:59:53.012,osotolongo,/data/subjects/XNAT05_S00062
XNAT05_S00027,unidad,20151338,2021-12-17
16:24:22.916,osotolongo,/data/subjects/XNAT05_S00027
XNAT05_S00028,unidad,20211523,2021-12-17
16:34:33.423,osotolongo,/data/subjects/XNAT05_S00028
XNAT05_S00030,unidad,20211401,2021-12-18
08:08:35.051,osotolongo,/data/subjects/XNAT05_S00030
XNAT05_S00031,unidad,20211281,2021-12-18
09:22:07.411,osotolongo,/data/subjects/XNAT05_S00031
XNAT05_S00032,unidad,20210716,2021-12-18
14:02:44.009,osotolongo,/data/subjects/XNAT05_S00032
XNAT05_S00033,unidad,20081210,2021-12-18
18:36:35.746,osotolongo,/data/subjects/XNAT05_S00033
```

Ahora para cada uno de los *label*, he de ejecutar,

```
[osotolongo@brick03 mri_face]$ sqlcmd ${connection_chain} -s "," -W -Q
"SELECT his_interno, xfecha_nac, xsexo_id FROM
[UNIT4_DATA].[imp].[vh_pac_gral] WHERE his_interno = '"${nhc}"';" | grep
${nhc}
20200484,1945-05-27 00:00:00.000,2
```

Nota: En la DB de Ekon, 1 → male, 2 → female

Entonces con estos datos he de construir un CSV para importar en XNAT.

make_csv.pl

```
#!/usr/bin/perl
#
# Copyleft 2022 O. Sotolongo <asqwerty@gmail.com>
#
use strict;
use warnings;
use File::Temp qw(:mktemp tempdir);
use Data::Dump qw(dump);
# Get input
my $xprj = 'unidad';
my $ilist;
while (@ARGV and $ARGV[0] =~ /^-/) {
        $_ = shift;
        last if /^--$/;
        if (/^-x/) {$xprj = shift; chomp($xprj);}
        if (/^-i/) {$ilist = shift; chomp($ilist);}
}
die "Should supply XNAT project" unless $xprj;

#Read config files
my $xconf_file = $ENV{'HOME'}.'/.xnatapic/xnat.conf';
my %xconf;
open IDF, "<$xconf_file";
while (<IDF>){
        if (/^#.*/ or /^\s*$/) { next; }
        my ($n, $v) = /(.*)=(.*)/;
        $xconf{$n} = $v;
}
close IDF;

my $sqlconf_file = $ENV{'HOME'}.'/.sqlcmd';
my %sqlconf;
open IDF, "<$sqlconf_file";
while (<IDF>){
        if (/^#.*/ or /^\s*$/) { next; }
        my ($n, $v) = /(.*)=(.*)/;
        $sqlconf{$n} = $v;
}
close IDF;

my $tmp_dir = tempdir(TEMPLATE => $ENV{TMPDIR}.'/xnat_data.XXXXXX',
CLEANUP => 1);
my $sbj_file = $tmp_dir.'/all_subjects.csv';
# me conecto y genero mi JSESSIONID
my $q = 'curl -f -X POST -u "'.$xconf{'USER'}.':'.$xconf{'PASSWORD'}.'"
"'.$xconf{'HOST'}.'/data/JSESSION" 2>/dev/null';
my $jid = qx/$q/;
# Saco los sujetos del proyecto
$q = 'curl -f -b JSESSIONID='.$jid.'
```

```
"'.$xconf{'HOST'}.'/data/projects/unidad/subjects?format=csv" >
'.$sbj_file.' 2>/dev/null';
system($q);
my @slist;
if ($ilist) {
        open IDF, "<$ilist";
        @slist = <IDF>;
        close IDF;
}
my %subjects;
open IDF, "<$sbj_file";
while (<IDF>){
        if(/^XNAT/){
                my ($xid,$xlabel) = /(XNAT.*),.*,(\d*),.*,.*,.*$/;
                #print "$xid -> $xlabel\n";
                if(!$ilist or grep( /^$xlabel$/, @slist)){
                        $subjects{$xid}{'label'} = $xlabel;
                        print "$xlabel\n";
                }
        }
}
close IDF;
#dump %subjects;
foreach my $subject (sort keys %subjects){
        my $sconn = 'sqlcmd -U '.$sqlconf{'USER'}.' -P
'.$sqlconf{'PASSWORD'}.' -S '.$sqlconf{'HOST'}.' -s "," -W -Q "SELECT
his_interno, xfecha_nac, xsexo_id FROM [UNIT4_DATA].[imp].[vh_pac_gral]
WHERE his_interno = \'"'.$subjects{$subject}{'label'}.'"\';" | grep
'.$subjects{$subject}{'label'};
        my $rdata = qx/$sconn/;
        my ($xdob, $xgender) = $rdata =~
/${subjects{$subject}{'label'}}\s*,\s*(\d{4}-\d{2}-\d{2}).*,(\d)$/;
        if($xdob and $xgender){
                $subjects{$subject}{'dob'} = $xdob;
                $subjects{$subject}{'gender'} =
$xgender==1?'male':'female';
        }
}
my $ofile = $xprj.'_dob_gender.csv';
my $sbj_header = 'ID,label,dob,gender';
open ODF, ">$ofile";
print ODF "$sbj_header\n";
foreach my $subject (sort keys %subjects){
        if (exists($subjects{$subject}{'dob'}) and
exists($subjects{$subject}{'gender'})){
                print ODF
"$subject,$subjects{$subject}{'label'},$subjects{$subject}{'dob'},$subj
ects{$subject}{'gender'}\n";
        }
}
```

```
close ODF;
```

Para construir el CSV con los datos demograficos de todo el proyecto basta hacer,

```
$ ./make_csv.pl -x proyecto
```

(si no se pone el nombre de proyecto toma como default *unidad*)

Si solo quiero hacer unos pocos,

```
$ cat newones.list
20220534
20210104
20210474
20171653
$ ./make_csv.pl -i newones.list
```

Y ahora puedo usar este metodo para subir el CSV

Obteniendo los datos de XNAT

Ahora voy a intentar rescatar desde XNAT los valores de RVR que he subido. Esto es, no hay una estructura fija, en principio, asi que tengo que obtener cualesquiera de los datos que haya definidos en el proyecto.

Para empezar se accede a XNAT y se abre la sesion,

```
my $xconf_file = $ENV{'HOME'}.'/.xnatapic/xnat.conf';
my %xconf;
open IDF, "<$xconf_file";
while (<IDF>){
        if (/^#.*/ or /^\s*$/) { next; }
        my ($n, $v) = /(.*)=(.*)/;
        $xconf{$n} = $v;
}
my $q = 'curl -f -X POST -u "'.$xconf{'USER'}.':'.$xconf{'PASSWORD'}.'"
"'.$xconf{'HOST'}.'/data/JSESSION"';
my $jid = qx/$q/;
```

y ahora vamos a tomar la lista de sujetos,

```
my %subjects;
$q = 'curl -f -b JSESSIONID='.$jid.' -X GET
```

```
"'.$xconf{'HOST'}.'/data/projects/'.$xprj.'/subjects?format=csv&columns=ID,label"';
my @sbj_res = split '\n', qx/$q/;
```

Esta respuesta de XNAT no es precisamente la lista de sujetos pero es bastante similar a lo que queremos y podemos ir recorriendo el array y construyendo nuestro output desde ahi.

```
foreach my $sbj_prop (@sbj_res){
        if ($sbj_prop =~ /^XNAT/){
                my ($sid,$slabel) = $sbj_prop =~ /^(XNAT.+),(\S+),(.*)$/;
                $subjects{$sid}{'label'} = $slabel;
```

asi, separamos el valor de *sid* que es el ID asignado por XNAT y el *label* que no es mas que el ID que hemos asignado en nuestro proyecto. Ahora si hacemos,

```
                my $qe = 'curl -f -b JSESSIONID='.$jid.' -X GET "'.$xconf{'HOST'}.'/data/projects/'.$xprj.'/subjects/'.$sid.'/experiments?format=json&xsiType=xnat:mrSessionData"';
                my $json_res = qx/$qe/;
```

Obtenemos un *json* con los experimentos del sujeto (tipicamente uno solo). Y usando el modulo JSON meto los contenidos en un *AoH*.

```
                my $exp_prop = decode_json $json_res;
                foreach my $experiment (@{$exp_prop->{'ResultSet'}{'Result'}}){
                                $subjects{$sid}{'experimentID'} = $experiment->{'ID'};
                }
```

y ahora voy a obtener el archivo RVR que haya guardado y convertirlo en una fila del CSV.

```
                if (exists($subjects{$sid}{'experimentID'}) and $subjects{$sid}{'experimentID'}){
                        my $qr = 'curl -f -b JSESSIONID='.$jid.' -X GET "'.$xconf{'HOST'}.'/data/projects/'.$xprj.'/experiments/'.$subjects{$sid}{'experimentID'}.'/resources/RVR/files?format=json"';
                        $json_res = qx/$qr/;
                        #print "$json_res\n";
                        my $rvr_prop = decode_json $json_res;
                        my $report_uri;
                        foreach my $rvr_res (@{$rvr_prop->{'ResultSet'}{'Result'}}){
                                if ($rvr_res->{'Name'} eq 'report_data.json'){
                                        $report_uri = $rvr_res->{'URI'};
                                }
                        }
                        if ($report_uri){
                                $qr = 'curl -f -b JSESSIONID='.$jid.' -X GET
```

```
"'.$xconf{'HOST'}.$report_uri.'"';
                                $json_res = qx/$qr/;
                                #print "$json_res\n";
                                my $report_data = decode_json $json_res;
                                foreach my $var_data
(@{$report_data->{'ResultSet'}{'Result'}}){
                                        my @akeys;
                                        my @adata;
                                        foreach my $kdata (sort keys
%{$var_data}){
                                                unless ($dhead){
                                                        push @akeys, $kdata
unless $kdata eq 'Subject';
                                                }
                                                push @adata,
${$var_data}{$kdata} unless $kdata eq 'Subject';
                                        }
                                        $dhead = 'Subject,'. join ',',
@akeys unless $dhead;
                                        $dbody .=
${$var_data}{'Subject'}.','. join ',', @adata;
                                        $dbody .= "\n";
                                }
                        }
                }
```

Ojo, que como no se que estructura va a tener esto, solo veo de poner la columna *Subject* al inicio y lo demas lo intento adivinar automagicamente.

Pues esto despues de escupe a un archiv de output y ya esta.

Para lanzarlo ha de hacerse algo como,

```
$ ./xnat_get_rvr.pl -x unidad -o rvr_output.csv
```

y si no se especifica el nombre de archivo output se creara uno con nombre como *proyecto_rvr_data.csv* que deberia esperarse que fuera algo como,

```
[osotolongo@brick03 mri_face]$ head f5cehbi_rvr_data.csv
Subject,ACG,ACP_D,ACP_I,ATM_D,ATM_I,Fazekas
F005,1,2,2,0,1,1
F006,NA,NA,NA,0,1,1
F007,NA,NA,NA,1,0,0
F009,2,2,2,1,1,2
F010,NA,NA,NA,NA,NA,1
F014,NA,NA,NA,1,0,2
F015,1,1,1,1,1,NA
F023,NA,NA,NA,0,1,2
F024,1,1,1,2,2,1
```

Eso segun elproyecto que sea y las variables que se hayan guardado en el *resource* RVR.

Sacando edad de XNAT

Para sacar la lista de sujetos del proyecto hago simplemente,

```
[osotolongo@brick03 ~]$ curl -b JSESSIONID=1709B38DD3BD206D00C3389592A4FC9E
-X GET
"http://detritus.fundacioace.com:8088/data/projects/unidad/subjects?format=c
sv&columns=ID,label"
```

La forma correcta de parsear el *json* para sacar la fecha de nacimiento es mas o menos asi,

```
[osotolongo@brick03 ~]$ curl -b JSESSIONID=1709B38DD3BD206D00C3389592A4FC9E
-X GET
'http://detritus.fundacioace.com:8088/data/subjects/XNAT05_S00052?format=jso
n' | jq '.items[0].children[] | select(.field=="demographics") |
.items[0].data_fields.dob'
```

y para sacar la fecha del primer MRI asignado a sujeto es,

```
[osotolongo@brick03 ~]$ curl -b JSESSIONID=1709B38DD3BD206D00C3389592A4FC9E
-X GET
'http://detritus.fundacioace.com:8088/data/projects/unidad/subjects/XNAT05_S
00052/experiments?format=json' | jq '.ResultSet.Result[0].date'
```

o si se conoce el experimento,

```
[osotolongo@brick03 ~]$ curl -b JSESSIONID=1709B38DD3BD206D00C3389592A4FC9E
-X GET
'http://detritus.fundacioace.com:8088/data/projects/unidad/subjects/XNAT05_S
00052/experiments/XNAT05_E00065?format=json' | jq
'.items[0].data_fields.date'
```

Asi que esto es un poco ir engranado todo

xnat_get_age.pl

```
#!/usr/bin/perl
#
# This is for getting age at MRI date
# Intended for MRIFACE protocol
# All info is XNAT stored.
#
# Copyleft 2022 <asqwerty@gmail.com>
#
use strict;
use warnings;
```

```
use JSON;
use Date::Manip;
use Math::Round;

my $xprj;
my $oxfile;
while (@ARGV and $ARGV[0] =~ /^-/) {
        $_ = shift;
        last if /^--$/;
        if (/^-o/) {$oxfile = shift; chomp($oxfile);}
        if (/^-x/) {$xprj = shift; chomp($xprj);}
}
die "Should supply XNAT project" unless $xprj;
$oxfile = $xprj.'_age_data.csv' unless $oxfile;
#Read xnat user config
my $xconf_file = $ENV{'HOME'}.'/.xnatapic/xnat.conf';
my %xconf;
open IDF, "<$xconf_file";
while (<IDF>){
        if (/^#.*/ or /^\s*$/) { next; }
        my ($n, $v) = /(.*)=(.*)/;
        $xconf{$n} = $v;
}
#get the jsessionid
my $q = 'curl -f -X POST -u "'.$xconf{'USER'}.':'.$xconf{'PASSWORD'}.'"
"'.$xconf{'HOST'}.'/data/JSESSION" 2>/dev/null';
my $jid = qx/$q/;
my %subjects;
$q = 'curl -f -b JSESSIONID='.$jid.' -X GET
"'.$xconf{'HOST'}.'/data/projects/'.$xprj.'/subjects?format=csv&columns
=ID,label" 2>/dev/null';
my @sbj_res = split '\n', qx/$q/;

foreach my $sbj_prop (@sbj_res){
        if ($sbj_prop =~ /^XNAT/){
                my ($sid,$slabel) = $sbj_prop =~
/^(XNAT.+),(\S+),(.*)$/;
                $subjects{$sid}{'label'} = $slabel;
                my $qe = 'curl -f -b JSESSIONID='.$jid.' -X GET
"'.$xconf{'HOST'}.'/data/projects/'.$xprj.'/subjects/'.$sid.'/experimen
ts?format=json&xsiType=xnat:mrSessionData" 2>/dev/null';
                my $json_res = qx/$qe/;
                my $exp_prop = decode_json $json_res;
                foreach my $experiment
(@{$exp_prop->{'ResultSet'}{'Result'}}){
                        $subjects{$sid}{'experimentID'} =
$experiment->{'ID'};
                }
                my $mri_date;
                if (exists($subjects{$sid}{'experimentID'}) and
```

```
$subjects{$sid}{'experimentID'}){
                        my $qr = 'curl -f -b JSESSIONID='.$jid.' -X GET
"'.$xconf{'HOST'}.'/data/projects/'.$xprj.'/experiments/'.$subjects{$si
d}{'experimentID'}.'?format=json" 2>/dev/null | jq
".items[0].data_fields.date"';
                        $mri_date = qx/$qr/; chomp $mri_date;
                        $mri_date =~ s/.*(\d{4})-(\d{2})-
(\d{2}).*/$2\/$3\/$1/;
                        #print "MRI date: $mri_date\n";
                }
                my $qs = 'curl -f -b JSESSIONID='.$jid.' -X GET
"'.$xconf{'HOST'}.'/data/projects/'.$xprj.'/subjects/'.$sid.'?format=js
on" 2>/dev/null | jq ".items[0].children[] |
select(.field==\"demographics\") | .items[0].data_fields.dob"';
                #print "$qs\n";
                my $dob = qx/$qs/; chomp $dob;
                $dob =~ s/.*(\d{4})-(\d{2})-(\d{2}).*/$2\/$3\/$1/;
                #print "DoB: $dob\n";
                if ($mri_date and $dob){
                        #my $d1 = ParseDate($dob);
                        #my $d2 = ParseDate($mri_date);
                        #print "$d1 - $d2\n";
                        my $ddif =
Delta_Format(DateCalc(ParseDate($dob),ParseDate($mri_date)),2,"%hh")/(2
4*365.2425);
                        $subjects{$sid}{'age'} = nearest(0.1, $ddif);
                        print "$subjects{$sid}{'label'},
$subjects{$sid}{'age'}\n";
                }
        }
}
open ODF, ">$oxfile";
print ODF "Subject,Age\n";
foreach my $subject (sort keys %subjects){
        if(exists($subjects{$subject}{'age'})){
                print ODF
"$subjects{$subject}{'label'},$subjects{$subject}{'age'}\n";
        }
}
close ODF;
```

Ejem, me ha costado un poco darme cuenta que debo poner las echas como *MM/DD/YYYY* para que *ParseDate()* las agarre bien, pero en fin,que no es tan complicado.

luego se hace algo como

```
$./xnat_get_age.pl -x unidad
```

y el archivo de ouput es obviamente

```
$ head unidad_age_data.csv
Subject,Age
20151338,87.7
20211523,70.3
20211401,63.1
20211281,79.4
20210716,74.2
20081210,82.9
20211524,77.1
20211269,65
20211482,75
```

Haciendo Update de la DB

Cuando he subido los sujetos nuevos, no me interesa subir todos los datos. Solo de los nuevos. Xnat permite bajarse un CSV con los datos fundamentales de los sujetos. Ejemplo,

```
[osotolongo@brick03 mri_face]$ head osotolongo_9_9_2022_14_20_45.csv
Subject,M/F,YOB,MR Sessions,Inserted
20040526,F,1943,1,2022-08-15 14:50:34.733
20050456,F,1940,1,2022-03-21 10:07:58.79
20050604,U,,1,2022-09-09 11:03:42.296
20071018,F,1937,1,2022-06-01 11:26:55.319
20081210,F,1939,1,2021-12-18 18:36:35.746
20090461,M,1944,1,2022-02-02 10:18:49.364
20090567,F,1951,1,2022-08-16 11:46:09.795
20100147,F,1943,1,2022-08-16 10:24:22.267
20100678,M,1943,1,2022-02-02 10:53:02.929
```

Lo primero es mirar aqui quienes no tienen Genero o Fecha de nacimiento,

```
[osotolongo@brick03 mri_face]$ awk -F"," '{if ($2=="U") print $1 }'
osotolongo_9_9_2022_14_20_45.csv | sort -n > newones.list
[osotolongo@brick03 mri_face]$ cat newones.list
20050604
20181621
20191940
20220120
20220144
20220205
20221002
20221079
20221124
2021020051
2021020062
2021020098
```

Por supuesto que aqui habra tambien los errores previos (*ver las ultimas lineas*) pero no importa

porque estos no se procesaran. Hago,

```
[osotolongo@brick03 mri_face]$ ./make_csv.pl -i newones.list
```

y el archivo resultante,

```
[osotolongo@brick03 mri_face]$ cat unidad_dob_gender.csv
ID,label,dob,gender
XNAT_S00611,20050604,1946-11-18,female
XNAT_S00612,20220144,1948-04-10,female
XNAT_S00613,20220120,1950-12-31,female
XNAT_S00614,20220205,1938-10-06,female
XNAT_S00615,20221002,1955-02-19,female
XNAT_S00616,20181621,1941-10-23,female
XNAT_S00617,20221079,1949-07-25,female
XNAT_S00618,20191940,1943-11-09,female
XNAT_S00619,20221124,1969-03-17,male
```

se utiliza para hacer un update subiendo el spreadsheet directamente a XNAT.

ahora, quiero ver los reports que faltan, asi que hago,

```
[osotolongo@brick03 mri_face]$ xnat_get_rvr.pl -x mriface
[osotolongo@brick03 mri_face]$ head mriface_rvr_data.csv
Subject_ID,Date,Fazekas,GCA_D,GCA_I,Kipps_A_D,Kipps_A_I,Kipps_F_D,Kipps_F_I,
Kipps_P_D,Kipps_P_I,Koedam_D,Koedam_I,Scheltens_D,Scheltens_I
20151338,2021-12-04,1,2,2,1,1,1,1,1,2,2,2,2
20211523,2021-12-05,2,1,1,NA,NA,NA,NA,NA,NA,1,1,1,2
2021020098,2021-12-04,1,2,2,2,2,2,2,1,1,1,2,2,2
20211401,2021-12-04,1,1,1,1,1,1,1,2,2,1,1,0,1
20211281,2021-12-01,1,3,3,NA,NA,NA,NA,NA,NA,2,3,3,3
20210716,2021-12-16,1,1,1,1,1,1,1,1,1,1,1,1,1
20081210,2021-12-02,1,1,1,NA,NA,NA,NA,NA,NA,1,1,0,0
20211524,2021-12-02,2,2,2,1,1,3,3,1,1,2,2,1,1
20211482,2021-12-02,2,2,2,3,3,2,2,2,2,2,2,4,4
[osotolongo@brick03 mri_face]$ awk -F"," '{print $1}' mriface_rvr_data.csv >
reports_done.list
[osotolongo@brick03 mri_face]$ awk -F"," '{print $1}'
osotolongo_9_9_2022_14_20_45.csv > all.list
[osotolongo@brick03 mri_face]$ grep -v "`cat reports_done.list`" all.list >
noreports.list
[osotolongo@brick03 mri_face]$ while read -r line; do xpr=$(xnatapic
list_experiments --project_id unidad --subject_id ${line} --modality MRI --
label); echo "${line},${xpr}"; done < noreports.list | awk -F"," '{print
$1","$3}' > sbj_ids.csv
[osotolongo@brick03 mri_face]$ head sbj_ids.csv
20050604,D22362317
20181621,D22407522
20191940,D22402332
20211269,D18290542
20220053,D22038623
```

```
20220120,D22363608
20220144,D22402314
20220205,D99365341
20220460,D99297208
20220581,D22362247
```

y ahora regresamos a bajar los PDFs y procesarlos

Evaluando neurodegeneración

Ahora, como determinar la N.

Primero obtener los datos,

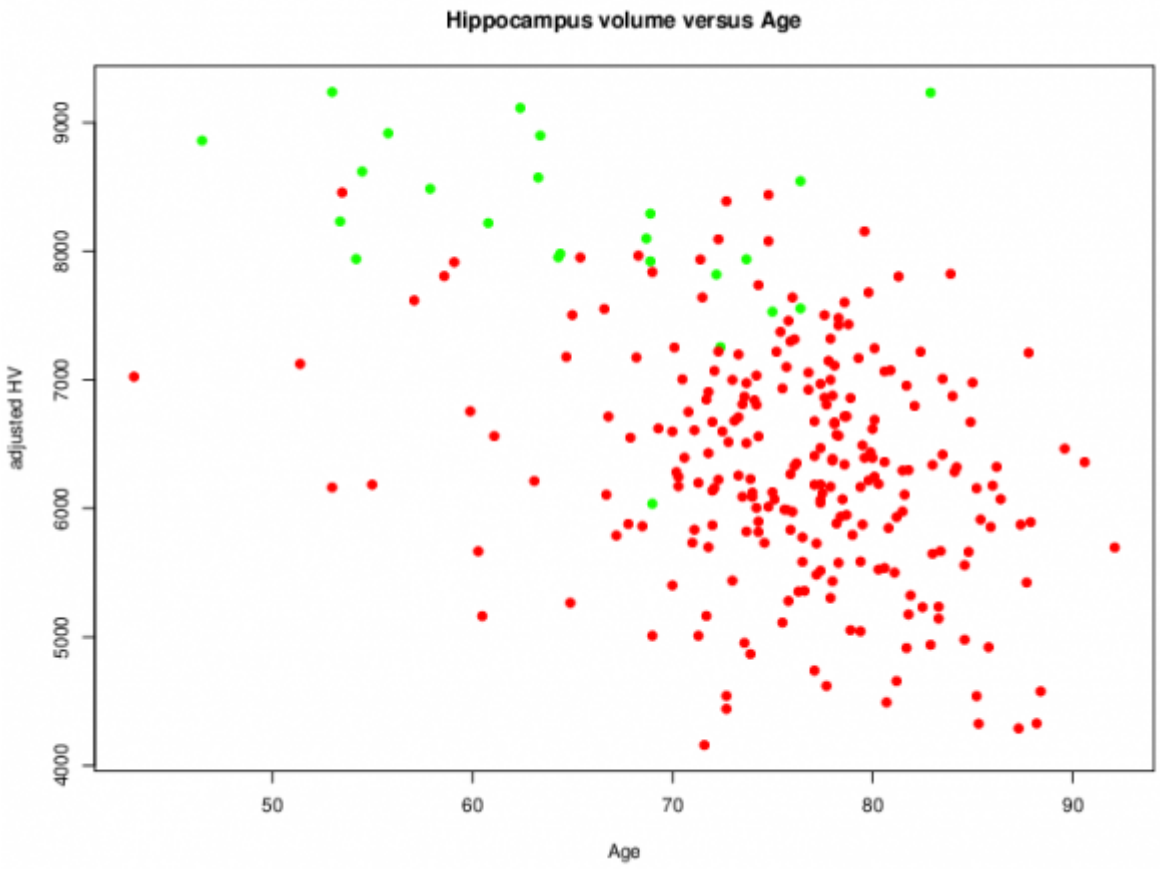
```
[osotolongo@brick03 mri_face]$ xnat_pullfs.pl -s aseg -p unidad -o
base_aseg.csv
[osotolongo@brick03 mri_face]$ xnat_pullfs.pl -s aparc -p unidad -o
base_aparc.csv
[osotolongo@brick03 mri_face]$ join -t, base_aseg.csv base_aparc.csv >
base_full.csv
[osotolongo@brick03 mri_face]$ ./xnat_get_age.pl -x unidad
[osotolongo@brick03 mri_face]$ sort -t, -n unidad_age_data.csv >
unidad_age_data_sorted.csv
[osotolongo@brick03 mri_face]$ join -t, base_full.csv
unidad_age_data_sorted.csv > input_data.csv
```

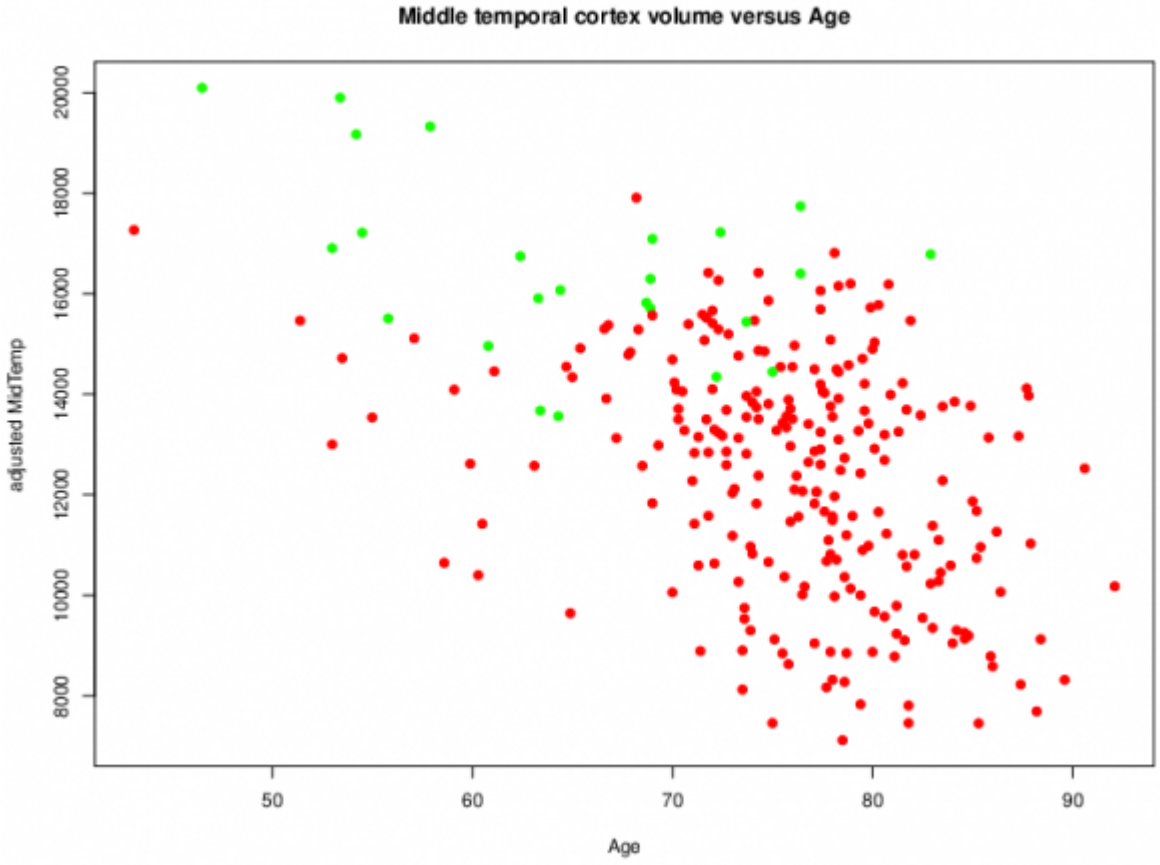
y ahora me bajo el script para calcular la N y,

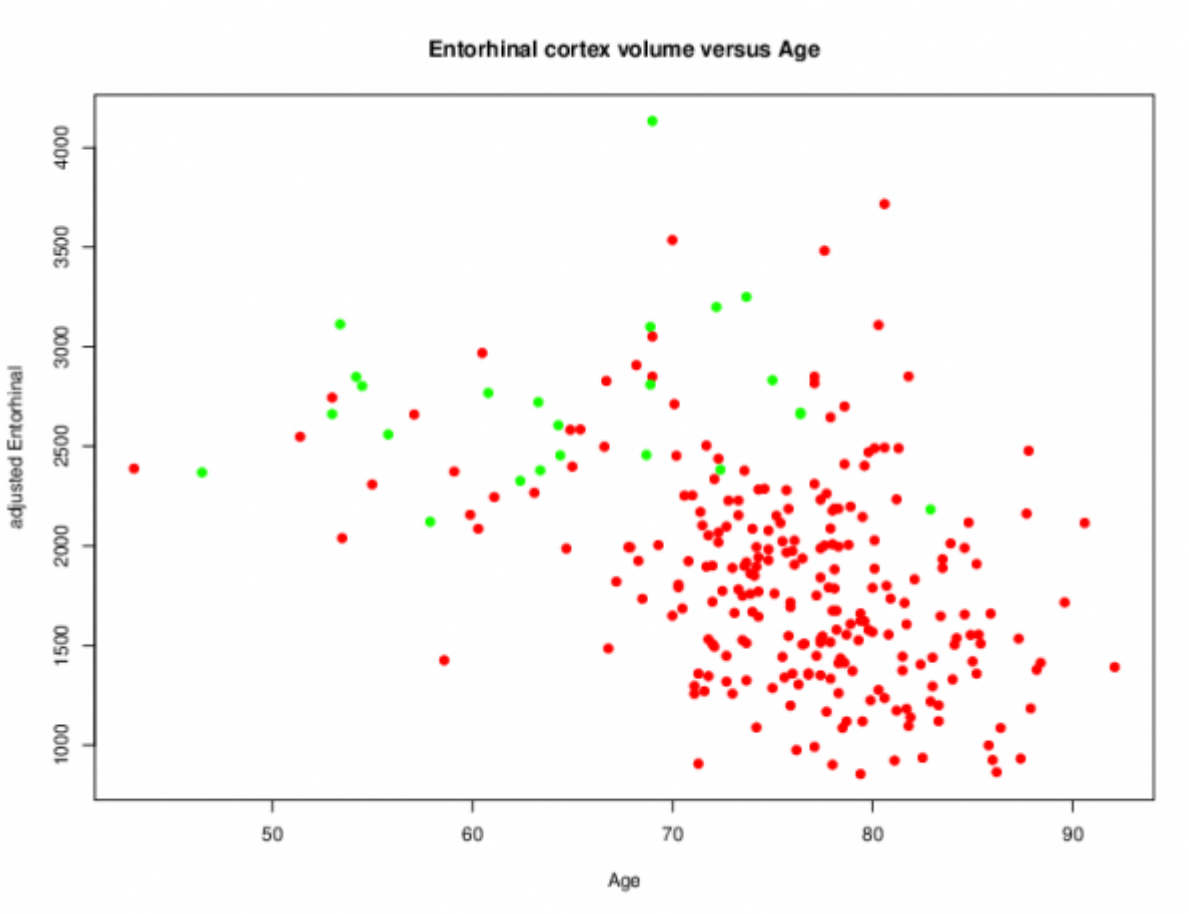
```
[osotolongo@brick03 mri_face]$ Rscript nplus.r
```

Y tenemos como resultado la estimacion de presencia de neurodegeneracion asi como la probailidad de pertenecer a cada grupo.

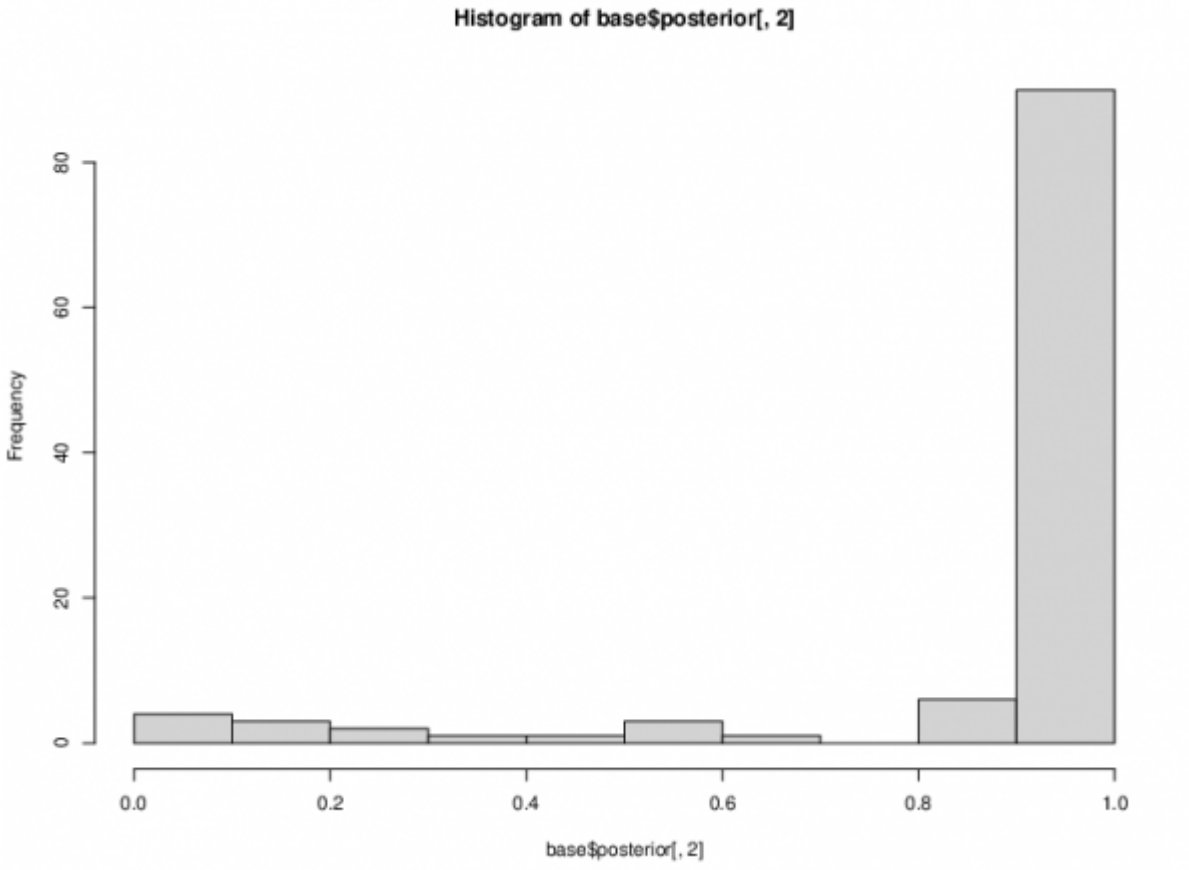
```
[osotolongo@brick03 mri_face]$ head classifier_output.csv
Subject_ID,ND,posterior.0,posterior.1
20050456,1,1.535472e-03,9.984645e-01
20081210,0,7.683998e-01,2.316002e-01
20090461,1,2.417535e-03,9.975825e-01
20100678,1,1.268098e-04,9.998732e-01
20140947,1,6.505165e-02,9.349483e-01
20150926,1,3.967107e-07,9.999996e-01
20151338,1,2.415657e-04,9.997584e-01
20160418,1,2.258146e-07,9.999998e-01
20170735,1,4.166497e-03,9.958335e-01
```


Hippocampus volume versus Age
adjusted HV
4000
5000
6000
7000
8000
9000
50
60
70
80
90
Age


Middle temporal cortex volume versus Age
adjusted MidTemp
8000
10000
12000
14000
16000
18000
20000
50
60
70
80
90
Age



y la densidad de probabilidades de presentar neurodegeneracion perteneciendo a este grupo sería algo como,



Juntando todo en un mismo output