Metanalisis de modelos de plink

Corriendo los modelos

Para sacar el analisis de modelos que esta implementado en plink se ejecuta algo como

```
$ plink --bfile archivo --model --allow-no-sex --out archivo
```

Esto deja el resultado en un archivo llamado *archivo.model*. Como aqui el objetivo final es un meta analisi vamos a ahcerlo para todas las DB del Variomics,

```
$ for x in `ls ../Variomics/*.bed`; do plink --bfile ${x%.bed} --model --
allow-no-sex --out $(basename ${x%.bed}); done
```

Esto deja una serie de archivos de resultado,

```
[osotolongo@detritus models]$ ls -lah *.model
-rw-rw-r-- 1 osotolongo osotolongo 479M May 15 13:18 ADMURimpQC2.model
-rw-rw-r-- 1 osotolongo osotolongo 831M May 15 13:20 ADNIimpQC2.model
-rw-rw-r-- 1 osotolongo osotolongo 692M May 15 13:22 GenADA_impQC2.model
-rw-rw-r-- 1 osotolongo osotolongo 805M May 15 13:25 NIA_AD_impQC2.model
-rw-rw-r-- 1 osotolongo osotolongo 597M May 15 13:26 TGEN_impQC2.model
```

Calculando los odds ratio y demas

Quiero extraer un determinado modelo (e.g.: recesivo → REC) de estos archivos y calcular los *odds ratio*, *standard error* y *p-value*. Para calcular las *p-value* voy a usar el Fisher's Exact Test dado que hay alelos de muy baja frecuencia a los que no puedo hacer un χ^2.

Hice un scriptcillo para ayudarme,

models.pl

```
#!/usr/bin/perl

# Copyright 2013 O. Sotolongo <osotolongo@fundacioace.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::Slurp qw(read_file);
use Text::NSP::Measures::2D::Fisher::twotailed;

my $plist = shift;
my $model = shift;
my $bimfile = shift;
die "usage: models.pl model_file model_tla bim_file\n" unless ($plist
&& $model && $bimfile);

my $ofile = $plist;
$ofile =~ s/(.*)\.(.*)/$1\.$model\.recalc2\.$2/;
# cambiar a clustering con (?:PATTERN)
my %input_data = reverse map
{/^(\s*\d+\s+(\S+)\s+[A,T,C,G]+\s+[A,T,C,G]+\s+$model\s+.*)$/} grep
{/^(.*\s+$model\s+.*)$/} read_file $plist;
my %bim_data = map
{/^\d+\s+(.*)\s+\d+\.*\d*\s+(\d+)\s+[A,T,C,G]+\s+[A,T,C,G]+\s*$/}
read_file $bimfile;
my %cells;
foreach my $marker (sort keys %input_data){
    (@{$cells{$marker}} {qw/chr allele1 allele2 affa1 affa2 unaffa1
unaffa2 chi2 df pval0/}) = $input_data{$marker} =~
/^\s*(\d+)\s+$marker\s+([A,T,C,G])+\s+([A,T,C,G])+\s+$model\s+(\d+)\/(\
d+)\s+(\d+)\/(\d+)\s+(\d+\.*\d*e*-*\d*|NA)\s+(\d+\.*\d*e*-
*\d*|NA)\s+(\d+\.*\d*e*-*\d*|NA)\s*$/;
    if(exists($cells{$marker}{affa1}) && exists($cells{$marker}{affa2})
&& exists($cells{$marker}{unaffa1}) &&
exists($cells{$marker}{unaffa2})){
        my $affa1 = $cells{$marker}{affa1};
        my $affa2 = $cells{$marker}{affa2};
        my $unaffa1 = $cells{$marker}{unaffa1};
        my $unaffa2 = $cells{$marker}{unaffa2};
        my $t1 = $affa1 + $affa2;
        my $t2 = $affa1 + $unaffa1;
        my $t3 = $t1 + $unaffa1 + $unaffa2;
        $cells{$marker}{pvalue} = calculateStatistic(n11=>$affa1,
n1p=>$t1, np1=>$t2, npp=>$t3);
        $affa1 = 0.1 unless ($affa1);
        $affa2 = 0.1 unless ($affa2);
        $unaffa1 = 0.1 unless ($unaffa1);
        $unaffa2 = 0.1 unless ($unaffa2);
        $cells{$marker}{afreq} = $affa1/($affa1+$affa2);
        $cells{$marker}{ufreq} = $unaffa1/($unaffa1+$unaffa2);
        $cells{$marker}{oddsratio} =
($affa1*$unaffa2)/($affa2*$unaffa1);
        $cells{$marker}{stderr} =
sqrt((1/$affa1)+(1/$affa2)+(1/$unaffa1)+(1/$unaffa2));
    }
}
```

```
my $head = "CHR\tSNP\tBP\tA1\tA2\tF_A\tF_U\tCHISQ\tDF\tP0\tOR\tSE\tP";
open ODF, ">$ofile" or die "Could not open output file\n";
print ODF "$head\n";

foreach my $marker (sort {($cells{$a}->{chr} <=> $cells{$b}->{chr}) or
($a cmp $b)} keys %input_data){
    if(exists($bim_data{$marker})){
        print ODF
"$cells{$marker}{chr}\t$marker\t$bim_data{$marker}\t$cells{$marker}{all
ele1}\t$cells{$marker}{allele2}\t$cells{$marker}{afreq}\t$cells{$marker
}{ufreq}\t$cells{$marker}{chi2}\t$cells{$marker}{df}\t$cells{$marker}{p
val0}\t$cells{$marker}{oddsratio}\t$cells{$marker}{stderr}\t$cells{$mar
ker}{pvalue}\n";
    }
}

close ODF;
```

La orden,

```
$ models.pl admurcia_model.model REC ../Variomics/ADMURimpQC2.bim
```

extrae el modelo recesivo, calcula los valores deseados y los añade a los ya existentes en el archivo *admurcia_model.REC.recalc.model*. Además usa el archivo **.bim** original para asignar a cada marcador su posicion fisica ya que *plink* la pide para el meta-analisis

Para correr todas las DBs juntas hago,

```
$ for x in `ls *.model | grep -v recalc`; do models.pl $x REC
../Variomics/${x%.model}.bim; done
```

y

```
$ for x in `ls *.model | grep -v recalc`; do models.pl $x DOM
../Variomics/${x%.model}.bim; done
```

dado que me interesan solo los modelos dominate y recesivo.

Meta analisis con plink

Para terminar basta con correr plink sobre todas las bases procesadas,

```
$ a=$(ls *REC*); plink --meta-analysis $a + study --out recesivo
```

y

```
$ a=$(ls *DOM*); plink --meta-analysis $a + study --out dominante
```

Los resultados salen ordenados por cromosoma y SNP. Para ordenar por *p-value* puede hacerse,

```
$ sort -g -k7,7 dominante.meta > dominante.meta.sorted
```

para cortar por un valor especifico de *p-value*,

```
$ awk {'if($7<5e-6 || $7=="P") print'} dominante.meta.sorted >
dominante.meta.sorted.cut
```

y si estamos interesados en algun SNP especifico,

```
$ echo "DB `head -n 1 ADNIimpQC2.REC.recalc.model`" > 20p.rec.cool.txt
$ for x in *REC*; do echo "${x%.REC.recalc.model} `grep -w "rs714235" $x`"
>> 20p.rec.cool.txt; done
```

Alternativa a Fisher: Barnard Test

Como al blandengue no le gusta el test de Fisher, he buscado como hacer que el script me calcule los test de Barnard. No he encontrado ningun modulo en Perl pero como todo ya esta inventado en esta vida hay un paquete en R que lo calcula. Asi que lo que he hecho es usar el modulo **Statistics::R** para calcular el test en R desde Perl. 😉

Esto es un poco enredado pero se resume en sustituir la llamada al test de Fisher con,

```
my $shit = Statistics::R->new();
$shit->run(q`library(Barnard)`);
$shit->run(qq`barnardw.test($affa1, $affa2, $unaffa1, $unaffa2,
verbose=FALSE)->bt`);
$shit->run(q`bt["p.value"][[1]][[2]] -> x`);
$cells{$marker}{pvalue} = shit->get('x');
```

OJO: Esto demora muchisimo. Despues de 5 días de calculo sobre una base de datos y modelo dominate tuve que para la ejecucion. Lo lanzare cuando tenga tiempo y la base de datos a usar este definida.