Small example of use of OmicABEL

Yurii Aulchenko for the OmicABEL developers

July 1, 2013

Contents

1 Important note on data format for OmicABEL .......................... 1
2 Outline of the example .................................................................. 2
3 Prepare the data for analysis ............................................................. 3
   3.1 Load the libraries and necessary data ........................................... 3
   3.2 Compute genomic relationship ................................................. 3
   3.3 Simulate some traits .................................................................. 4
   3.4 Export the data in DatABEL format ............................................. 5
   3.5 Export the data in format for FaST-LMM ....................................... 6
4 Run OmicABEL analysis ................................................................. 8
   4.1 Extract the data in text format .................................................. 10
5 Explore the results .......................................................................... 10
   5.1 Load data into R ....................................................................... 11
   5.2 Explore the results ..................................................................... 11
6 Run alternative analysis and compare the results ............................ 11
   6.1 Run FaST-LMM analysis ........................................................... 11
   6.2 Compare the results ................................................................... 13

1 Important note on data format for OmicABEL

The example of use provided below make use of the data provided together with the GenABEL-package. Note that when working with your data you
do NOT need to have the data in the GenABEL-package format, and you
do not need to follow this procedure to get data usable for OmicABEL;
this is a simple example explaining the input and output files and usage of
OmicABEL.

The major fact you need to remember is that OmicABEL makes use of
the ‘filevector’ (aka ‘DatABEL’) “DOUBLE” format for input. Hence, again,
you do NOT have to have your data in GenABEL format, but you do need
to get files into filevector/DatABEL “DOUBLE”.

The major issue is getting the (usually vast amounts of) genotypic data
in right format. To get to right format, in real life we recommend that you
use one of the GenABEL conversion procedures to convert your data from
IMPUTE, MACH, or MiniMac to DatABEL/filevector. The correspond-
ing GenABEL-package functions are impute2databel, mach2databel, and
minimac2databel. Make, however, sure that you use the dataOutType =
“DOUBLE” argument when argument when doing the genotype data con-
version! For example (in R),

```
mach2databel(imputedgenofile = "f1.mldose", mlinfofile = "f1.mlinfo",
        outfile="f1", dataOutType = "DOUBLE")
```

This argument is added to the GenABEL-package since version >=1.7-7.

If you have already have genotypic data in filevector/DatABEL format,
but these are in the “FLOAT” format (default option for the xxx2databel
procedures), you can use the float2double utility provided together with
OmicABEL to convert to “DOUBLE”. For example, if you have your geno-
typic data in (filevector-FLOAT) files myData.fvi and myData.fvd, you can
convert them to DOUBLE by using (from command line):

```
float2double myData myDataDouble
```

After which you will get files myDataDouble.fvi and myDataDouble.fvd
(note that the size of these files is roughly double the size of floats - take
care you have enough HDD space).

2 Outline of the example

In this example, we will use the data set distributed with GenABEL to
show the use of OmicABEL. Hence you need to have GenABEL package
installed on your system. You will also need the DatABEL package for data
manipulations and mvtnorm package (simulation of traits) installed.
For conversion of files to the FaST-LMM format, you will need `PLINK` installed, you will also need `FaST-LMM` if you’d like to run the comparison.

3 Prepare the data for analysis

Note that you do NOT need to follow this procedure to get data usable for OmicABEL. The trick is to get your data in tables and then dump these tables in 'filevector' (aka 'DatABEL') format. You do NOT need to have your data in GenABEL format for that!

Please start R and ...

3.1 Load the libraries and necessary data

We will load the ‘ge03d2.clean’ data set

```r
library(mvtnorm)
library(DatABEL)
library(GenABEL)
data(ge03d2.clean)
df <- ge03d2.clean[,autosomal(ge03d2.clean)]
```

DatABEL v.0.9-4 (March 12, 2013) loaded
Loading required package: MASS
GenABEL v. 1.7-4 (February 22, 2013) loaded

and check how many individuals and SNPs are there

```r
nids(df)
nsnps(df)
```

[1] 822
[1] 6826

3.2 Compute genomic relationship

Compute genomic KINSHIP

```r
gkin <- ibs(df,weight="freq")
gkin[1:3,1:3]
```
Now transform this into relationship matrix

grel <- gkin
grel[upper.tri(grel)] <- t(grel)[upper.tri(grel)]
grel <- 2*grel
grel[1:3,1:3]

```
id4   id10   id25
id4  0.89455901  0.01571251 -0.01585682
id10 0.01571251  0.93140902  0.01376398
id25 -0.01585682  0.01376398  0.96302789
```

### 3.3 Simulate some traits

```r
set.seed(10)
nTraits <- 10
model.h2 <- runif(nTraits,min=0.3,max=0.7)
myPhenos <- matrix(NA,ncol=nTraits,nrow=nids(df))
s2 <- 3
j <- 1
for (current.h2 in model.h2) {
  Sigma <- s2*(current.h2*grel + (1-current.h2)*diag(rep(1,nids(df))))
  myPhenos[,] <- rmvnorm(1,sigma=Svar)
  j <- j + 1
}
colnames(myPhenos) <- paste("tra",1:dim(myPhenos)[2],sep="")
myPhenos[1:2,]
```

```
tra1  tra2  tra3  tra4  tra5  tra6  tra7
[1,]  0.137185 -1.507003  3.299568 -1.181792  0.2607997 -0.8527273 -2.3883019
[2,] -2.177526 -0.442744  0.1585556  1.784114  1.5286974  2.5535508 -0.9764274
```

```
tra8  tra9  tra10
[1,] -0.3908143 -1.0160779  2.773482
[2,]  2.0045780  0.1445038  2.448823
```

Let us also compute all ratios between the traits:
tmp <- myPhenos
for (i in 1:(nTraits-1)) {
  for (j in (i+1):nTraits) {
    cRatioIJ <- tmp[,i]/tmp[,j]
    cRatioJI <- tmp[,j]/tmp[,i]
    tmp <- cbind(tmp, cRatioIJ)
    colnames(tmp)[dim(tmp)[2]] <- paste("rat_", i, ",", j, sep="")
    tmp <- cbind(tmp, cRatioJI)
    colnames(tmp)[dim(tmp)[2]] <- paste("rat_", j, ",", i, sep="")
  }
}
dim(tmp)
myPhenos <- tmp

[1] 822 100

3.4 Export the data in DatABEL format

Make sure you always use the DOUBLE type!

    Need to make sure there are no ‘older’ filevector files around

killFiles <- c("pheno?.??","genos.??","covars.??","grel.??")
unlink(killFiles)

    Export traits. We first export one trait

tmp <- matrix2databel(from=myPhenos[,1,drop=FALSE],filename="pheno1",type="DOUBLE")

    and then all traits – this is handy to show the use of different options

tmp <- matrix2databel(from=myPhenos,filename="phenos",type="DOUBLE")

    Export covariates - note you need ‘1’ as the first covariate!

myCovariates <- phdata(df)[,c("sex","age")]
myCovariates <- cbind(1,myCovariates)
myCovariates[1:3,]
tmp <- matrix2databel(from=as.matrix(myCovariates),filename="covars",type="DOUBLE")

    1 sex   age
id4   1 0 51.63771
id10  1 1 53.73938
id25  1 0 66.01148
Export relationship

tmp <- matrix2databel(from=grel,filename="grel",type="DOUBLE")

Export genotypes. Note that you do NOT need to have genotypes in the GenABEL format to do this operation.

tmp <- matrix2databel(from=as.numeric(gtdata(df)),filename="genos",type="DOUBLE")

Remove the ‘tmp’ object and ‘gc’ to keep things clean:

rm(tmp)

gc()

<table>
<thead>
<tr>
<th>used (Mb)</th>
<th>gc trigger (Mb)</th>
<th>max used (Mb)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ncells</td>
<td>364098</td>
<td>19.5</td>
</tr>
<tr>
<td>Vcells</td>
<td>3105774</td>
<td>23.7</td>
</tr>
</tbody>
</table>

3.5 Export the data in format for FaST-LMM

We are going to compare the OmicABEL results with FaST-LMM, and therefore will export the results in a format usable for FaST-LMM as well. Note this part is NOT necessary to run the OmicABEL!

Start with genotypic files, to be exported in TPED format

export.plink(df,transpose=TRUE)

Export covariates

falmmCov <- cbind(1:nids(df),idnames(df),phdata(df)[,c("sex","age")])
falmmCov[1:3,]

write.table(falmmCov,file="plink.cov",col.names=FALSE,row.names=FALSE,quote=FALSE)

<table>
<thead>
<tr>
<th>1:nids(df)</th>
<th>idnames(df)</th>
<th>sex</th>
<th>age</th>
</tr>
</thead>
<tbody>
<tr>
<td>id4</td>
<td>1</td>
<td>id4</td>
<td>0 51.63771</td>
</tr>
<tr>
<td>id10</td>
<td>2</td>
<td>id10</td>
<td>1 53.73938</td>
</tr>
<tr>
<td>id25</td>
<td>3</td>
<td>id25</td>
<td>0 66.01148</td>
</tr>
</tbody>
</table>

Export phenotypes
falmmPhe <- cbind(1:nids(df), idnames(df), myPhenos)
write.table(falmmPhe, file = "plink.phe", col.names = FALSE, row.names = FALSE, quote = FALSE)

Export genomic relationship
falmmRel <- grel
nms <- paste(1:nids(df), idnames(df))
colnames(falmmRel) <- nms
falmmRel <- cbind(var = nms, falmmRel)
write.table(falmmRel, file = "plink.sim", col.names = TRUE, row.names = FALSE, quote = FALSE, sep = "\t")

Now, you can leave R.
Let us switch to the shell and convert the TPED into binary PLINK format (so our 'timing' comparison is more fair)

plink --tfile plink --make-bed

@----------------------------------------------------------@
| PLINK! | v1.07 | 10/Aug/2009 |
|---------------------------|------------------|
| (C) 2009 Shaun Purcell, GNU General Public License, v2 |
| For documentation, citation & bug-report instructions: |
| http://pngu.mgh.harvard.edu/purcell/plink/ |
@----------------------------------------------------------@
Web-based version check ( --noweb to skip )
Recent cached web-check found... OK, v1.07 is current

Writing this text to log file [ plink.log ]
Analysis started: Fri Mar 15 01:40:55 2013

Options in effect:
--tfile plink
--make-bed

Reading pedigree information from [ plink.tfam ]
822 individuals read from [ plink.tfam ]
0 individuals with nonmissing phenotypes
Assuming a disease phenotype (1=unaff, 2=aff, 0=miss)
Missing phenotype value is also -9
0 cases, 0 controls and 822 missing
438 males, 384 females, and 0 of unspecified sex
6826 (of 6826) markers to be included from [ plink.tped ]
Before frequency and genotyping pruning, there are 6826 SNPs
822 founders and 0 non-founders found
Total genotyping rate in remaining individuals is 0.990041
1
0 SNPs failed frequency test ( MAF < 0 )
After frequency and genotyping pruning, there are 6826 SNPs
After filtering, 0 cases, 0 controls and 822 missing
After filtering, 438 males, 384 females, and 0 of unspecified sex
Writing pedigree information to [ plink.fam ]
Writing map (extended format) information to [ plink.bim ]
Writing genotype bitfile to [ plink.bed ]
Using (default) SNP-major mode

Analysis finished: Fri Mar 15 01:40:59 2013

4 Run OmicABEL analysis

Let us check that all files are present in the current directory:

ls *.fv?
covars.fvd
covars.fvi
genos.fvd
genos.fvi
grel.fvd
grel.fvi
pheno1.fvd
pheno1.fvi
phenos.fvd
phenos.fvi

Make sure that the executables ‘CLAK-GWAS’ and ‘reshuffle’, or make sure they are in your path. We can now run the OmicABEL analysis with (also using ‘time’ to time the run).
We first run analysis of single trait using the option ‘chol’

**CLAK-GWAS -var chol -nths 3 -cov covars -phi grel \ -snp genos -pheno pheno1 -out myres1**

Running a Genome-Wide Association Study of the following size:
- sample size: 822
- # of covariates: 2
- # of SNPs: 6826
- # of phenotypes: 1

Will use CLAK-Chol with the following parameters:
- $x_b$: 3072
- $y_b$: 1
- $o_b$: 3072
- $x_{tile}$: 160
- $y_{tile}$: 160
- # of threads: 3

Available memory 3.107723 GBs (out of 3.873131 GBs)

Estimating GWAS parameters: heritability and variance... Done (took 0.287 secs)
Performing the study... Done (took 0.888 secs)

(this takes about 3 seconds)

Now, for analysis of multiple traits it is suggested to use the option ‘eigen’:

**CLAK-GWAS -var eigen -nths 3 -cov covars -phi grel \ -snp genos -pheno phenos -out myres**

Running a Genome-Wide Association Study of the following size:
- sample size: 822
- # of covariates: 2
- # of SNPs: 6826
- # of phenotypes: 100

Will use CLAK-Eig with the following parameters:
- $x_b$: 3200
- $y_b$: 100
Estimating GWAS parameters: heritability and variance... Done (took 2.088 secs)
Performing the study... Done (took 5.996 secs)

(this takes about 20 seconds)
Easy, ergh?! Note that time does not add up - doing $N$ phenotypes is much faster then doing $N$ time one phenotype!

4.1 Extract the data in text format
This command will dump ALL results into single large text file (danger! danger! - check the reshuffle options for more targeted extracts)

```
reshuffle myres --chi2
```

Let us have a look at the first few lines of the output:

```
head -n 3 chi_data.txt
```

```
SNP    Trait  beta_1  beta_sex  beta_age  beta_SNP  se_1  se_sex
rs1646456  tral   0.0915423184633255 0.122388660907745 0.122388660907745 -0.000707811850588769 -0.000707811850588769
rs7950586  tral  -0.0256201047450304 0.11798419803381 0.11798419803381 0.00100125558674335 -0.00388470059260726 -0.00388470059260726
```

5 Explore the results
Start R again and …
5.1 Load data into R

```r
myRes <- read.table("chi_data.txt",head=TRUE,stringsAsFactors=FALSE)
myRes$Pvalue <- pchisq(myRes$Chi2,1,low=FALSE)
dim(myRes)
myRes[1:2,]
```

```
[1] 682600 18
SNP Trait beta_1 beta_sex beta_age beta_SNP se_1
1 rs1646456 tra1 0.09154232 0.1223887 -0.0007078119 -0.2248352 0.2267392
2 rs7950586 tra1 -0.02562010 0.1179842 -0.0008740884 -0.1955265 0.2193964
se_sex se_age se_SNP cov_sex_1 cov_age_1 cov_SNP_1
1 0.1071139 0.004135856 0.1023784 -0.006057684 -0.0008482659 -0.006096948
2 0.1071970 0.004135147 0.2361242 -0.006151666 -0.0008531595 -0.003884701
cov_age_sex cov_SNP_sex cov_SNP_age Chi2 Pvalue
1 -5.950255e-07 -4.163674e-05 -1.022416e-05 4.822943 0.02808336
2 -9.072596e-07 1.001256e-03 -1.512502e-05 0.685694 0.40763289
```

5.2 Explore the results

Let us check GC $\lambda$’s and max($\chi^2$) for all the traits:

```r
for (cTrait in unique(myRes$Trait)) {
  condition <- which(myRes$Trait==cTrait)
  lambda <- median(myRes[condition,"Chi2"])/qchisq(.5,1)
  maxChi2 <- max(myRes[condition,"Chi2"])
  cat(cTrait,"'s Lambda = ",lambda,"; max chi2 = ",maxChi2,"\n")
}
```

```
tra1 's Lambda = 1.040671 ; max chi2 = 11.51143
tra2 's Lambda = 1.001553 ; max chi2 = 14.37402
tra3 's Lambda = 1.005891 ; max chi2 = 12.72925
[OUTPUT TRUNCATED]
```

6 Run alternative analysis and compare the results

6.1 Run FaST-LMM analysis

Let us analyze trait number 1 (and on the way we will also store the eigen-decomposition)
fastlmmc -bfile plink -sim plink.sim -covar plink.cov \ 
  -pheno plink.phe -mpheno 1 \ 
  -REML -simLearnType ONCE -MaxThreads 3 \ 
  -eigenOut flmmEigenRes -out flmmOutT1.txt

(this takes about 3 seconds)

Now, we can fairly time how long does it take to analyze all 10 phenotypes. For this, we will arrange an ‘sh’ file which will run all phenotypes analysis (and time it). This perl script generates the ‘runFaST-LMM.sh’:

    open OUF,">runFaST-LMM.sh" or die $!
    for ($i=1;$i<=100;$i++) {
      print OUF "
fastlmmc -bfile plink -sim plink.sim -covar plink.cov \ 
  -pheno plink.phe -mpheno $i \ 
  -REML -simLearnType ONCE -MaxThreads 3 \ 
  -eigen flmmEigenRes -out flmmOutT$i.txt"
    }

Let us run it and check few first lines if the batch file:

head runFaST-LMM.sh

fastlmmc -bfile plink -sim plink.sim -covar plink.cov \ 
  -pheno plink.phe -mpheno 1 \ 
  -REML -simLearnType ONCE -MaxThreads 3 \ 
  -eigen flmmEigenRes -out flmmOutT1.txt

fastlmmc -bfile plink -sim plink.sim -covar plink.cov \ 
  -pheno plink.phe -mpheno 2 \ 
  -REML -simLearnType ONCE -MaxThreads 3 \ 
  -eigen flmmEigenRes -out flmmOutT2.txt

Finally, let us run (and time) the analysis of all 10 traits:

sh runFaST-LMM.sh &> runFaST-LMM.out

(this takes about 220 seconds)
6.2 Compare the results

Time-wise, you can easily see that OmicABEL outperforms other implementations. Let us now check how close are the results.

Let us first extract OmicABEL results for a specific trait, “tra3”, so we do not need to load everything into R:

```r
head -n 1 chi_data.txt > chi_data_tra3.txt
grep "tra3" chi_data.txt >> chi_data_tra3.txt
```

Load OmicABEL results

```r
myRes <- read.table("chi_data_tra3.txt",head=TRUE,stringsAsFactors=FALSE)
myRes$Pvalue <- pchisq(myRes$Chi2,1,low=FALSE)
dim(myRes)
myRes[1:2,]
```

```
[1] 6826 18

SNP Trait beta_1 beta_sex beta_age beta_SNP se_1
1 rs1646456 tra3 -0.03245008 0.01006570 -0.0006363846 -0.0116124 0.2320724
2 rs7950586 tra3 -0.08013462 0.02045402 -0.0007954766 0.5805020 0.2243592

se_sex se_age se_SNP cov_sex_1 cov_age_1 cov_SNP_1
1 0.1095037 0.004230591 0.1063008 -0.006363481 -0.0008876185 -0.006569172
2 0.1095915 0.004229751 0.2449815 -0.006463339 -0.0008930187 -0.004231980

cov_age_sex cov_SNP_sex cov_SNP_age Chi2 Pvalue
1 -4.002032e-09 -4.097003e-05 -1.114071e-05 0.01193359 0.91301138
2 -3.186624e-07 1.078362e-03 -1.526427e-05 5.61488639 0.01780854
```

and results from FaST-LMM - say, trait 3:

```r
falmmRes <- read.table("flmmOutT3.txt",head=TRUE,stringsAsFactors=FALSE)
falmmRes$Chi2 <- qchisq(falmmRes$Pvalue,1,low=FALSE)
dim(falmmRes)
falmmRes[1:2,]
```

```
[1] 6826 19

SNP Chromosome GeneticDistance Position Pvalue Qvalue N
1 rs245122 1 0 366653 0.0003820629 0.9383752 822
2 rs547541 3 0 10955677 0.0004055348 0.9383752 822

NullLogLike AltLogLike SNPWeight SNPWeightSE WaldStat NullLogDelta
1 -1579.886 -1586.850 0.2664040 0.07468547 12.72358 -0.08197409
```
The results are perfectly correlated:

rownames(falmmRes) <- falmmRes$SNP
table(rownames(falmmRes) %in% myRes[,"SNP"])
falmmRes <- falmmRes[myRes[,"SNP"],]
cor(falmmRes$Chi2,myRes[,"Chi2"])^2

TRUE
6826
[1] 0.9998198

Cross-plot the results:

jpeg("OmicAvsFaST.jpeg")
plot(falmmRes$Chi2,myRes[,"Chi2"],xlab="FaST-LMM",ylab="OmicABEL")
abline(a=0,b=1)
dev.off()
	null device

1