crlmm to downstream data analysis

VJ Carey, B Carvalho

March, 2012

1 Running CRLMM on a nontrivial set of CEL files

To use the `crlmm` algorithm, the user must load the *crlmm* package, as described below:

```
> library(crlmm)
```

We work with the 90 CEU samples hybridized to Affy 6.0 chips. When CEL files are available, they must be identified and passed to `crlmm`, as shown below. In this example, we assume that the results are stored in a variable called `crlmmResult`.

```
> celFiles <- list.celfiles()
> crlmmResult <- crlmm(celFiles)
```

Alternatively, the data aforementioned are available through the *hapmapsnp6* package (required minimum version 1.3.6) and can be loaded by using:

```
> suppressPackageStartupMessages(library(hapmapsnp6))
> data(crlmmResult)
```

This is currently a *SnpSet* object.

```
>   class(crlmmResult)

[1] "SnpSet"
attr(,"package")
[1] "Biobase"
```

2 Adding information to a *SnpSet*

We will use the *GGdata* package to obtain extra information on the samples. This will be later used when building an *eSet* extension to store the genotyping results.

```
>   suppressPackageStartupMessages(library(GGdata))
>   hmceuB36 <- getSS('GGdata', as.character(1:22))
>   pd <- phenoData(hmceuB36)
>   ggn <- sampleNames(pd)
>   preSN <- sampleNames(crlmmResult)
>   simpSN <- gsub("_.*", "", preSN)
>   if (!all.equal(simpSN, ggn)) stop("align GGdata phenoData with crlmmResult read")
```

The additional information obtained from *GGdata* can be easily combined to what is already available on `crlmmResult`.

```
>   sampleNames(crlmmResult) <- simpSN
>   phenoData(crlmmResult) <- combine(pd, phenoData(crlmmResult))
>   dim(calls(crlmmResult))

[1] 906600     90

>   dim(confs(crlmmResult, FALSE))

[1] 906600     90

>   calls(crlmmResult)[1:10, 1:2]

             NA06985 NA06991
SNP_A-2131660       2       2
SNP_A-1967418       3       3
SNP_A-1969580       3       3
SNP_A-4263484       2       1
SNP_A-1978185       1       1
SNP_A-4264431       1       1
SNP_A-1980898       3       3
SNP_A-1983139       1       1
SNP_A-4265735       2       2
SNP_A-1995832       2       3

>   confs(crlmmResult, FALSE)[1:10, 1:2]

             NA06985 NA06991
SNP_A-2131660   10561   11574
SNP_A-1967418   12517   14866
SNP_A-1969580    7632    7606
SNP_A-4263484   15621   20059
SNP_A-1978185   14030   18021
SNP_A-4264431   17792   17235
SNP_A-1980898    7640    7642
SNP_A-1983139   14127    8974
SNP_A-4265735    8976    9153
SNP_A-1995832   10336   17920
```

3 Coercing to SnpMatrix as a prelude to a GWAS

From this point on, we will use only the genotype calls. Therefore, to reduce memory requirements, we will recode the *crlmm* genotype calls, so the *snpStats* package can be used, and delete the remaining `crlmm` results.

```
> theCalls <- t(calls(crlmmResult))-1L
> rm(crlmmResult)
```

```
            used  (Mb) gc trigger   (Mb)  max used   (Mb)
Ncells   7631568 407.6   11161382  596.1   8125770  434.0
Vcells 109939505 838.8  295584944 2255.2 295028076 2250.9
```

SNP's for which all the samples have the same genotype are not informative for association studies. Therefore, we remove such SNP's prior to fitting the models.

```
> gtypeCounts <- rbind(AA=colSums(theCalls == 0L),
+                      AB=colSums(theCalls == 1L),
+                      BB=colSums(theCalls == 2L))
> gtypeCounts[, 1:5]
```

```
   SNP_A-2131660 SNP_A-1967418 SNP_A-1969580 SNP_A-4263484 SNP_A-1978185
AA             1             3             0             8            90
AB            32            14             2            40             0
BB            57            73            88            42             0
```

```
> toRemove <- which(colSums(gtypeCounts == 0) == 2L)
> gtypeCounts[, toRemove[1:4]]
```

```
   SNP_A-1978185 SNP_A-1983139 SNP_A-1997689 SNP_A-1997709
AA            90            90             0            90
AB             0             0             0             0
BB             0             0            90             0
```

```
> theCalls <- theCalls[, -toRemove]
```

The *snpStats* provides tools to simplify the analysis of GWAS. The snippet below shows how to load the package and convert the genotype calls to a format that *snpStats* is able to handle.

```
> suppressPackageStartupMessages(library(snpStats))
> crlmmSM <- new("SnpMatrix", theCalls)
```

```
coercing object of mode  numeric  to SnpMatrix
```

```
> crlmmSM
```

```
A SnpMatrix with  90 rows and  774475 columns
Row names:  NA06985 ... NA12892
Col names:  SNP_A-2131660 ... SNP_A-8573964
```

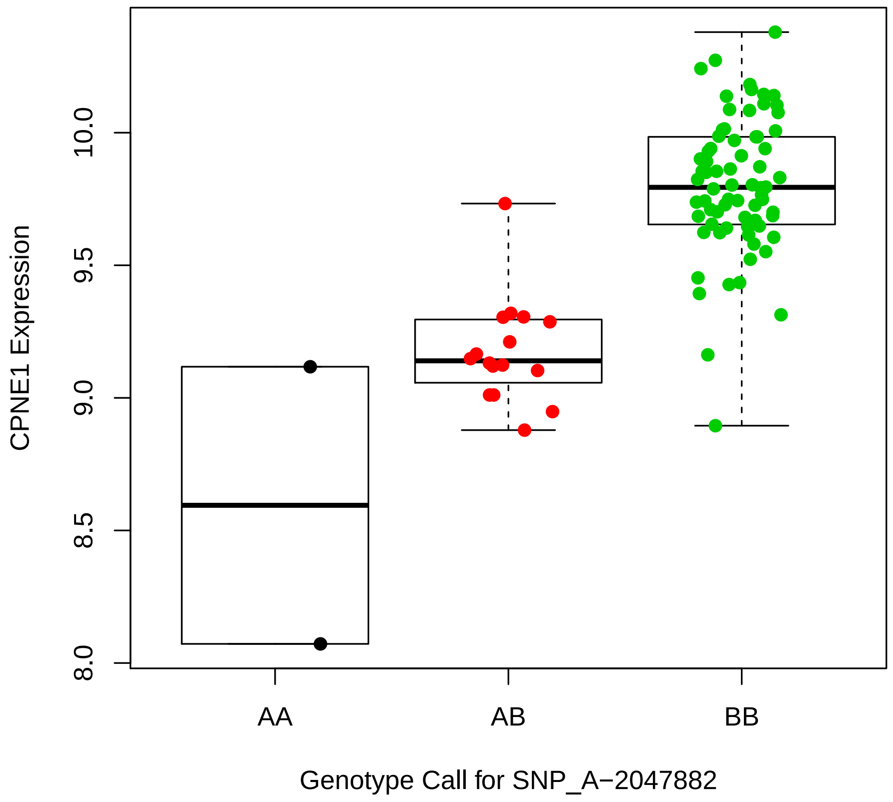
4 Conducting a GWAS

We want to find SNP for which genotype is predictive of expression of CPNE1. We will use expression data available from GGdata, using a naive analysis.

```
> suppressPackageStartupMessages(library(illuminaHumanv1.db))
> rmm <- revmap(illuminaHumanv1SYMBOL)
> mypr <- get("CPNE1", rmm)
> ex <- as.numeric(exprs(hmceuB36)[mypr[1],])
> subjdata <- pData(hmceuB36)
> subjdata[["ex"]] <- ex
> head(subjdata)

        famid persid mothid fathid  sampid isFounder  male        ex
NA06985  1341     14      0      0 NA06985      TRUE FALSE  9.654887
NA06991  1341      2     14     13 NA06991     FALSE FALSE  9.551434
NA06993  1341     13      0      0 NA06993      TRUE  TRUE 10.083945
NA06994  1340      9      0      0 NA06994      TRUE  TRUE  9.930053
NA07000  1340     10      0      0 NA07000      TRUE FALSE  9.645724
NA07019  1340      2     12     11 NA07019     FALSE FALSE  9.788195
```

With the expression data now available in `subjdata`, we can use the tools from *SnpMatrix* to fit models that will be used to evaluate the association between the genotypes of each available SNP and the expression levels of CPNE1.

```
> gwas <- snp.rhs.tests(ex~male, data=subjdata, snp.data=crlmmSM, family="gaussian")
> ok <- which(p.value(gwas) < 1e-10)
> gwas[ok,]

             Chi.squared Df      p.value
SNP_A-2047882    41.82453  1 9.984311e-11
SNP_A-2216659    41.82453  1 9.984311e-11
SNP_A-2220183    46.38761  1 9.702689e-12
SNP_A-2231469    46.38761  1 9.702689e-12
SNP_A-2275065    46.38761  1 9.702689e-12
SNP_A-1890801    42.67888  1 6.450512e-11

> snp <- names(gwas[ok,])[1]
> gtypes <- theCalls[,snp]+1L
> boxplot(ex~gtypes, xlab=paste("Genotype Call for", snp),
+         ylab="CPNE1 Expression", xaxt="n", range=0)
> points(ex~jitter(gtypes), col=gtypes, pch=19)
> axis(1, at=1:3, labels=c("AA", "AB", "BB"))
```



5 Session Info

This vignette was created using the following packages:

```
> sessionInfo()
```

```
R version 2.15.0 beta (2012-03-20 r58793)
Platform: x86_64-apple-darwin9.8.0/x86_64 (64-bit)

locale:
[1] en_GB.UTF-8/en_GB.UTF-8/en_GB.UTF-8/C/en_GB.UTF-8/en_GB.UTF-8

attached base packages:
[1] splines   stats     graphics  grDevices datasets  utils     methods
[8] base
```

```
other attached packages:
 [1] GGdata_1.0.18            illuminaHumanv1.db_1.12.2
 [3] org.Hs.eg.db_2.7.1       RSQLite_0.11.1
 [5] DBI_0.2-5                AnnotationDbi_1.17.27
 [7] GGBase_3.16.5            snpStats_1.5.5
 [9] Matrix_1.0-6             lattice_0.20-6
[11] survival_2.36-12         Biobase_2.15.4
[13] BiocGenerics_0.1.14      hapmapsnp6_1.3.6
[15] crlmm_1.13.16            oligoClasses_1.17.36
[17] RColorBrewer_1.0-5       BiocInstaller_1.1.28

loaded via a namespace (and not attached):
 [1] affyio_1.23.2        annotate_1.33.8      Biostrings_2.23.6
 [4] bit_1.1-9            codetools_0.2-8      ellipse_0.3-7
 [7] ff_2.2-5             foreach_1.3.5        genefilter_1.37.1
[10] grid_2.15.0          IRanges_1.13.32      iterators_1.0.5
[13] mvtnorm_0.9-9992     preprocessCore_1.17.7 stats4_2.15.0
[16] tools_2.15.0         xtable_1.7-0         zlibbioc_1.1.1
```