Using *crlmm* to genotype data from Illumina's Infinium BeadChips

Matt Ritchie

September 18, 2012

1 Getting started

In this user guide we read in and genotype data from 40 HapMap samples which have been analyzed using Illumina's 370k Duo BeadChips. This data is available in the *hapmap370k* package. Additional chip-specific model parameters and basic SNP annotation information used by CRLMM is stored in the *human370v1cCrlmm* package. The required packages can be installed in the usual way using the `biocLite` function.

```
> source("http://www.bioconductor.org/biocLite.R")
> biocLite(c("crlmm", "hapmap370k", "human370v1cCrlmm"))
```

2 Reading in data

The function `readIdatFiles` extracts the Red and Green intensities from the binary `idat` files output by Illumina's scanning device. The file `samples370k.csv` contains information about each sample.

```
> library(Biobase)
> library(crlmm)
> library(hapmap370k)
> data.dir = system.file("idatFiles", package="hapmap370k")
> # Read in sample annotation info
> samples = read.csv(file.path(data.dir, "samples370k.csv"), as.is=TRUE)
> samples[1:5,]

> # Read in .idats using sampleSheet information
> RG = readIdatFiles(samples, path=data.dir,
+ arrayInfoColNames=list(barcode=NULL,position="SentrixPosition"),saveDate=TRUE)
```

Reading in this data takes approximately 100 seconds and peak memory usage was 0.8 GB of RAM on our linux system. If memory is limiting, load the *ff* package and run the same command. When this package is available, the objects are stored using disk rather then RAM. The `RG` object is an *NChannelSet* which stores the Red and Green intensities, the number of beads and standard errors for each bead-type. The scanning date of each array is stored in `protocolData`.

```
> class(RG)

[1] "NChannelSet"
attr(,"package")
[1] "Biobase"

> dim(RG)

Features  Samples
  381079       40

> slotNames(RG)

[1] "assayData"         "phenoData"         "featureData"
[4] "experimentData"    "annotation"        "protocolData"
[7] ".__classVersion__"

> channelNames(RG)

[1] "G"    "R"    "zero"

> exprs(channel(RG, "R"))[1:5,1:5]

      4030186347_A 4030186263_B 4019585415_B 4031058127_B
10008          321          170         2961         3468
10010         1738         3702         3105         3425
10025           80          101          145           29
10026         5043         1856         6519         8304
10039         4905         2464         9080         9788
      4031058211_B
10008          262
10010           70
10025           21
10026         9872
10039        10867

> exprs(channel(RG, "G"))[1:5,1:5]
```

```
      4030186347_A 4030186263_B 4019585415_B 4031058127_B
10008         4183         4484         3765         3558
10010         2593           51         3824         3528
10025         2768         2322         3435         3471
10026          216         2840          211          164
10039          297         3016          345          361
      4031058211_B
10008         6502
10010         6154
10025         3608
10026          188
10039          380
```

```
> pd = pData(RG)
> pd[1:5,]
```

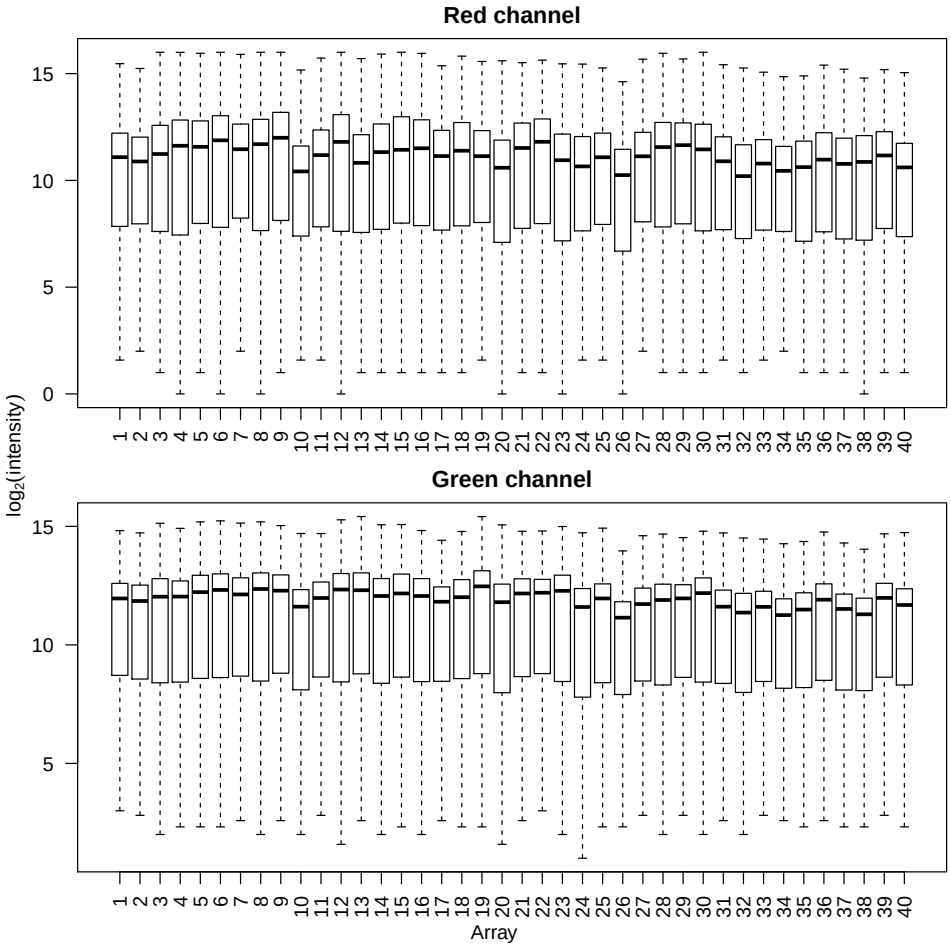
```
             HapMap.Name Gender         Plate Well SentrixPosition
4030186347_A     NA06991 Female WG1000442-DNA  E11    4030186347_A
4030186263_B     NA07000 Female WG1000442-DNA  D08    4030186263_B
4019585415_B     NA10859 Female WG1000453-DNA  B02    4019585415_B
4031058127_B     NA11882 Female WG1000453-DNA  D08    4031058127_B
4031058211_B     NA06993   Male WG1000447-DNA  D11    4031058211_B
```

```
> scandatetime = strptime(protocolData(RG)[["ScanDate"]], "%m/%d/%Y %H:%M:%S %p")
> datescanned = substr(scandatetime, 1, 10)
> scanbatch =  factor(datescanned)
> levels(scanbatch) = 1:16
> scanbatch = as.numeric(scanbatch)
```

If GenCall output is available instead of idat files, the function `readGenCallOutput` can be used to read in the data. This function assumes the GenCall output is formatted to have samples listed one below the other, and that the columns 'X Raw' and 'Y Raw' are available in the file. The resulting `NChannelSet` from this function can be used as input to `crlmmIllumina` via the `XY` argument (instead of the usual `RG` argument used when the data has been read in from idat files).

Plots of the summarised data can be easily generated to check for arrays with poor signal.

```
> par(mfrow=c(2,1), mai=c(0.4,0.4,0.4,0.1), oma=c(1,1,0,0))
> boxplot(log2(exprs(channel(RG, "R"))), xlab="Array", ylab="", names=1:40,
+ main="Red channel",outline=FALSE,las=2)
> boxplot(log2(exprs(channel(RG, "G"))), xlab="Array", ylab="", names=1:40,
+ main="Green channel",outline=FALSE,las=2)
> mtext(expression(log[2](intensity)), side=2, outer=TRUE)
> mtext("Array", side=1, outer=TRUE)
```



3 Genotyping

Next we use the function `crlmmIllumina` which performs preprocessing followed by genotyping using the CRLMM algorithm.

```
> crlmmResult = crlmmIllumina(RG=RG, cdfName="human370v1c", returnParams=TRUE)
```

This analysis took 3 minutes to complete and peak memory usage was 1.9 GB on our system. The output stored in `crlmmResult` is a *SnpSet* object.

```
> class(crlmmResult)
```

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

```
> dim(crlmmResult)
```

```
Features  Samples
  346451       40
```

```
> slotNames(crlmmResult)
```

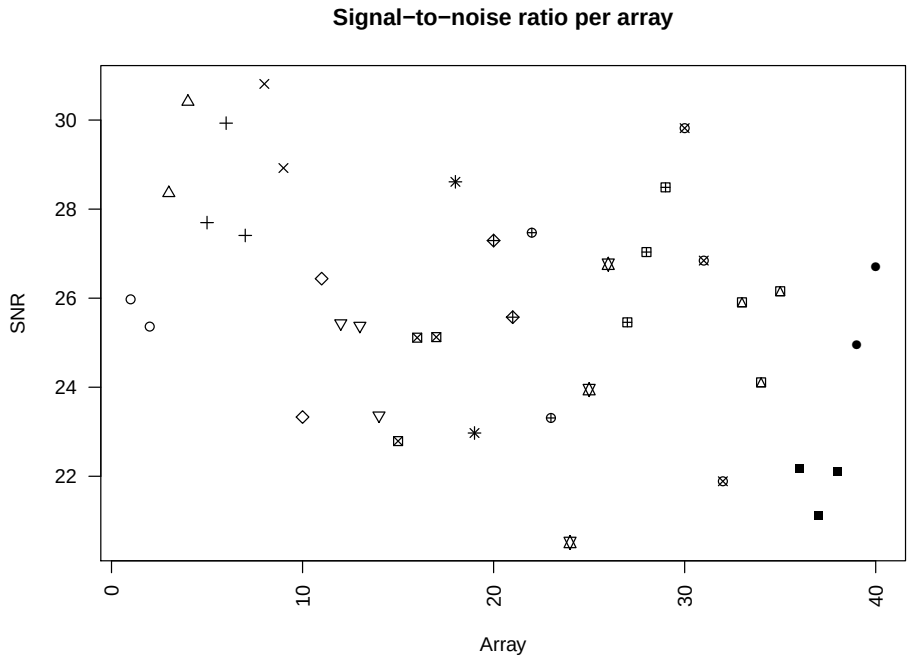
```
[1] "assayData"         "phenoData"         "featureData"
[4] "experimentData"    "annotation"        "protocolData"
[7] ".__classVersion__"
```

```
> calls(crlmmResult)[1:10, 1:5]
```

```
           4030186347_A 4030186263_B 4019585415_B 4031058127_B
rs12354060            3            3            3            3
rs6650104             1            1            1            1
rs12184279            1            1            1            1
rs12564807            1            1            1            1
rs3115860             2            1            1            2
rs3115850             1            2            2            1
rs7515489             3            3            1            1
rs12124819            1            2            2            1
rs17160939            1            1            1            1
rs12086311            3            3            3            3
           4031058211_B
rs12354060            3
rs6650104             1
rs12184279            1
rs12564807            1
rs3115860             2
rs3115850             1
rs7515489             1
rs12124819            1
rs17160939            1
rs12086311            3
```

Plotting the *SNR* reveals no obvious batch effects in this data set (different symbols are used for arrays scanned on different days).

```
> plot(crlmmResult[["SNR"]], pch=scanbatch, xlab="Array", ylab="SNR",
+      main="Signal-to-noise ratio per array",las=2)
```



An all-in-one function named `crlmmIlluminaV2` that combines reading of idat files with genotyping is also available.

```
> crlmmResult2 <- crlmmIlluminaV2(samples, path=data.dir,
+                                 arrayInfoColNames=list(barcode=NULL,position="Sentr
+                                 saveDate=TRUE, cdfName="human370v1c", returnParams=
```

4 System information

This analysis was carried out on a linux machine with 32GB of RAM using the following packages:

```
> sessionInfo()

R version 2.15.1 Patched (2012-07-01 r59713)
Platform: x86_64-unknown-linux-gnu (64-bit)

locale:
 [1] LC_CTYPE=en_US.iso885915       LC_NUMERIC=C
 [3] LC_TIME=en_US.iso885915        LC_COLLATE=en_US.iso885915
 [5] LC_MONETARY=en_US.iso885915    LC_MESSAGES=en_US.iso885915
 [7] LC_PAPER=C                     LC_NAME=C
 [9] LC_ADDRESS=C                   LC_TELEPHONE=C
[11] LC_MEASUREMENT=en_US.iso885915 LC_IDENTIFICATION=C
```

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

other attached packages:
[1] human370v1cCrlmm_1.0.2 crlmm_1.15.28
[3] hapmap370k_1.0.1       oligoClasses_1.19.42
[5] Biobase_2.16.0         BiocGenerics_0.2.0
[7] BiocInstaller_1.4.7

loaded via a namespace (and not attached):
 [1] affyio_1.24.0         annotate_1.34.1       AnnotationDbi_1.18.1
 [4] Biostrings_2.24.1     bit_1.1-8             codetools_0.2-8
 [7] DBI_0.2-5             ellipse_0.3-7         ff_2.2-7
[10] foreach_1.4.0         genefilter_1.38.0     GenomicRanges_1.8.7
[13] grid_2.15.1           IRanges_1.14.4        iterators_1.0.6
[16] lattice_0.20-6        mvtnorm_0.9-9992      preprocessCore_1.18.0
[19] RSQLite_0.11.1        splines_2.15.1        stats4_2.15.1
[22] survival_2.36-14      tools_2.15.1          XML_3.9-4
[25] xtable_1.7-0          zlibbioc_1.2.0
```