Preprocessing and Genotyping Illumina Arrays for Copy Number Analysis

Rob Scharpf

March 25, 2011

Abstract

This vignette illustrates the steps required prior to copy number analysis for Infinium platforms. Specifically, we require construction of a container to store processed forms of the raw data, preprocessing to normalize the arrays, and genotyping using the CRLMM algorithm. After completing these steps, users can refer to the `copynumber` vignette.

1 Setup

The following codechunk declares a directory for saving `ff` files that will contain the normalized intensities and the genotype calls.

```
> library(ff)
> if (getRversion() < "2.13.0") {
      rpath <- getRversion()
  } else rpath <- "trunk"
> outdir <- paste("/thumper/ctsa/snpmicroarray/rs/ProcessedData/crlmm/",
      rpath, "/illumina_vignette", sep = "")
> ldPath(outdir)
> dir.create(outdir, recursive = TRUE, showWarnings = FALSE)
```

We will also store cached computations in the directory `outdir`.

We declare that crlmm should process 150,000 markers at a time and/or 500 samples at a time (when possible) to reduce the memory footprint. As our example dataset in this vignette contains fewer than 500 samples, all samples will be processed simultaneously.

```
> ocProbesets(150000)
> ocSamples(500)
```

2 Initializing a container for storing processed data

This section will initialize a container for storing processed forms of the data, including the normalized intensities for the A and B alleles and the CRLMM genotype calls and confidence scores. In addition, the container will store information on the markers (physical position, chromosome, and a SNP indicator), the batch, and the samples (e.g., gender). To construct this container for Infinium platforms, several steps are required.

We begin by specifying the path containing the raw IDAT files for a set of samples from the Infinium 370k platform.

```
> datadir <- "/thumper/ctsa/snpmicroarray/illumina/IDATS/370k"
```

For Infinium platforms, an Illumina sample sheet containing information for reading the raw IDAT files is required. Please refer to the BeadStudio Genotyping guide, Appendix A, for additional information. The following code reads in the samplesheet for the IDAT files on our local server.

```
> samplesheet = read.csv(file.path(datadir, "HumanHap370Duo_Sample_Map.csv"),
     header = TRUE, as.is = TRUE)
```

For the purposes of this vignette, we indicate that we only wish to process a subset of the arrays. For our dataset, the file extensions are 'Grn.dat' and 'Red.idat'. We store the complete path to the filename without the file extension in the `arrayNames` and check that all of the green and red IDAT files exists.

```
> samplesheet <- samplesheet[-c(28:46, 61:75, 78:79), ]
> arrayNames <- file.path(datadir, unique(samplesheet[,
     "SentrixPosition"]))
> all(file.exists(paste(arrayNames, "_Grn.idat", sep = "")))

[1] TRUE

> all(file.exists(paste(arrayNames, "_Red.idat", sep = "")))

[1] TRUE

> arrayInfo <- list(barcode = NULL, position = "SentrixPosition")
```

All supported platforms have a corresponding annotation package. The appropriate annotation package is specified by the platform identifier without the `Crlmm` postfix.

```
> cdfName <- "human370v1c"
```

Next, we construct a character vector that specifies the batch for each of the 43 arrays. Here, we have a small dataset and process the samples in a single batch. Processing the samples as a single batch is generally reasonable if the samples were processed at similar times (e.g., within a few weeks).

```
> batch <- rep("1", nrow(samplesheet))
```

Finally, we initialize an object of class `CNSet` using the function `constructInf`.

```
> cnSet <- constructInf(sampleSheet = samplesheet, arrayNames = arrayNames,
     batch = batch, arrayInfoColNames = arrayInfo, cdfName = cdfName,
     verbose = TRUE, saveDate = TRUE)
```

A concise summary of the object's contents can be viewed with the `print` function.

```
> print(cnSet)

CNSet (storageMode: lockedEnvironment)
assayData: 370024 features, 43 samples
  element names: alleleA, alleleB, call, callProbability
protocolData
  rowNames: 4019585367_A 4019585376_B ... 4030186434_B (43
    total)
  varLabels: ScanDate DecodeDate
  varMetadata: labelDescription
phenoData
  sampleNames: 4019585367_A 4019585376_B ... 4030186434_B (43
    total)
  varLabels: gender SNR SKW
  varMetadata: labelDescription
featureData
  featureNames: rs12354060 rs6650104 ... cnv13957p106 (370024
    total)
  fvarLabels: chromosome position isSnp
  fvarMetadata: labelDescription
experimentData: use 'experimentData(object)'
Annotation: human370v1c
batch:    1 43
batchStatistics:  29  elements,  370024  features,  1  batches
```

Note that the above object does not yet contain any processed data (only `NA`'s). As the elements of the `assayData` slot are `ff` objects (not matrices), several `.ff` files now appear in the `outdir`. The `.ff` files should not be removed and can be listed using the R function `list.files`.

```
> sapply(assayData(cnSet), function(x) class(x)[1])

callProbability            call         alleleA         alleleB
    "ff_matrix"     "ff_matrix"     "ff_matrix"     "ff_matrix"

> list.files(outdir, pattern = ".ff")[1:5]

[1] "A582f24e2.ff"    "A72b95e7b.ff"    "B328c7218.ff"
[4] "B330d252e.ff"    "call3479a74a.ff"
```

3 Preprocessing

The raw intensities from the Infinium IDAT files are read and normalized using the function `preprocessInf`. The function `preprocessInf` returns a `ff` object containing the parameters for the mixture model used by the CRLMM genotyping algorithm.

```
> mixtureParams <- preprocessInf(cnSet = cnSet, sampleSheet = samplesheet,
     arrayNames = arrayNames, arrayInfoColNames = arrayInfo)
> invisible(open(mixtureParams))
> str(mixtureParams[])
> invisible(close(mixtureParams))
```

Note that the normalized intensities for the A and B alleles are no longer `NA`s and can be inspected using the methods `A` and `B`, respectively.

```
> invisible(open(A(cnSet)))
> invisible(open(B(cnSet)))
> as.matrix(A(cnSet)[1:5, 1:5])

           4019585367_A 4019585376_B 4019585413_A 4019585415_B
rs12354060         2134         2693         2982         2388
rs6650104          6496         5476         8149         6934
rs12184279          699         2332         2004         1724
rs12564807        19324        20850        20055        20860
rs3115860          7294         7472         7774         6985
           4019585422_A
rs12354060         1928
rs6650104         10635
rs12184279         1804
rs12564807        21730
rs3115860          9236

> as.matrix(B(cnSet)[1:5, 1:5])
```

```
           4019585367_A 4019585376_B 4019585413_A 4019585415_B
rs12354060        17555        17793        17681        18274
rs6650104           606          529          989          566
rs12184279         6088         5920         6017         6685
rs12564807          361          427          468          311
rs3115860           791         4611         5141          568
           4019585422_A
rs12354060        18019
rs6650104           711
rs12184279         7826
rs12564807          360
rs3115860           806
```

```
> invisible(close(A(cnSet)))
> invisible(close(B(cnSet)))
```

4 Genotyping

CRLMM genotype calls and confidence scores are estimated using the function `genotypeInf`.

```
> updated <- genotypeInf(cnSet, mixtureParams = mixtureParams)
> print(updated)
```

The posterior probabilities for the genotype calls in the `callProbability` element of the `assayData` are stored as integers to reduce the file size on disk. The scores can be transformed to the probability scale using the `i2p` function as illustrated in the following code chunk.

```
> invisible(open(snpCallProbability(cnSet)))
> callProbs <- as.matrix(snpCallProbability(cnSet)[1:5,
     1:5])
> i2p(callProbs)
```

```
           4019585367_A 4019585376_B 4019585413_A 4019585415_B
rs12354060    0.9821008    0.6617601    0.6785779    0.9457417
rs6650104     0.9994915    0.9994833    0.9989811    0.9995005
rs12184279    0.9971459    0.9785064    0.9780722    0.7230726
rs12564807    0.0000000    0.0000000    0.0000000    0.0000000
rs3115860     0.9995192    0.9961281    0.9884145    0.9995201
           4019585422_A
rs12354060    0.9905335
rs6650104     0.9993081
rs12184279    0.6740463
rs12564807    0.0000000
rs3115860     0.9995201
```

```
> invisible(close(snpCallProbability(cnSet)))
```

Wrapper: As an alternative to calling the functions `constructInf`, `preprocessInf` and `genotypeInf` in sequence, a convenience function called `genotype.Illumina` is a wrapper for the above functions and produces identical results.

```
> cnSet2 <- genotype.Illumina(sampleSheet = samplesheet,
     arrayNames = arrayNames, arrayInfoColNames = arrayInfo,
     cdfName = "human370v1c", batch = batch)
```

```
> invisible(open(calls(cnSet)))
> invisible(open(calls(cnSet2)))
> snp.index <- which(isSnp(cnSet))
> identical(calls(cnSet)[snp.index, 1:20], calls(cnSet2)[snp.index,
     1:20])

[1] TRUE

> invisible(close(calls(cnSet)))
> invisible(close(calls(cnSet2)))
```

To fully remove the data associated with the `cnSet2` object, one should use the `delete` function in the `ff` package followed by the `rm` function. The following code is not evaluated is it would change the results of the cached computations in the previous code chunk.

```
> lapply(assayData(cnSet2), delete)
> lapply(batchStatistics(cnSet2), delete)
> delete(cnSet2$gender)
> delete(cnSet2$SNR)
> delete(cnSet2$SKW)
> rm(cnSet2)
```

Users may now proceed to the CopyNumber vignette for copy number analyses.