Using the AffyTiling package

CharlesDanko[‡*]

April 15, 2011

[‡]March 11, 2008
Department of Pharmacology
SUNY Upstate Medical University
Syracuse, NY

Contents

*dankoc@gmail.com

1 Overview of AffyTiling

Affymetrix tiling arrays are becoming more widely used as the technology and coverage improves. The AffyTiling package addresses the needs of biologists who want to analyze Affymetrix tiling arrays at the level of individual probes. A standard analysis using this package will require using at least two files (listed below). Running the analysis on these files is covered in section 2.1.A. Affymetrix .bpmap file - Contains information about the chip design; can be acquired for your array from the Affymetrix website: www.affymetrix.com.

Affymetrix .CEL file(s) - Contains probe-specific intensity information. Should be the GCOS output from your experiment.

Since tiling arrays contain millions of probes, it is sometimes preferable to limit the number of probes depending on the capabilities of the PC that you are using to run the analysis. In the event that you can't run the analysis on the entire chip, or if you are only interested in certain genomic positions, AffyTiling allows you to easily extract the intensity for individual probes given compatible genomic coordinates for the intervals of interest. Limiting the analysis to certain probes is covered in section 2.1.B. Acquiring the proper table using the UCSC genome browser is covered in section 3. Further analysis of these arrays often requires associating probes with a transcription start site. This functionality is covered in section 2.2.

2 Getting Started

2.1 Import and normalize a group of CEL files.

2.1.1 Basic extraction of all probes on human promoter tiling array.

To load AffyTiling, use the following:

```
> library(AffyTiling)
```

To obtain RMA normalized (background corrected, quantile normalized, log-2 transformed) intensity values for every probe on the human tiling array, use the following command:

```
> EXP <- AnalyzeTilingCelFiles(dir(pattern = ".cel|.CEL"), "Hs_PromPR_v02-3_NCBI
```

This function makes use of the excellent "affy" package in Bioconductor **?**. It takes 12 minutes to extract and normalize all probes from 4 CEL files on a 1.3GHz Pentium M laptop with 1GB of RAM. Due to the size of these files (44MB each) it is not possible to distribute them as data.

```
> colnames(EXP)
> EXP[1:10, 1:5]
```

By default, the data matrix contains a (row 1) unique ID for each probe, (row 2 - 3) the start and chromosome of the probe, and (row 4 - end) normalized intensity values. The amount of data the function returns can be regulated by boolean flags. For example, the following command will return probe sequences (up to 25bp in length) instead of a UniqueID for each probe. For additional details, check the documentation for the AnalyzeTilingCelFiles function.

```
> EXP <- AnalyzeTilingCelFiles(dir(pattern = ".cel|.CEL"), "Hs_PromPR_v02-3_NCBI
+     makeUniqueID = FALSE, readProbeSeq = TRUE)
```

2.1.2 Limiting probe extraction to intervals of particular interest.

To return probes only inside an interval of interest, we just have to specify 4 additional arguments to the AnalyzeTilingCelFiles function: iID, iCHR, iSTART, and iEND. These arguments are vectors representing: an ID for each interval, the chromosome, start, and end coordinates of the interval, respectively. Depending on the size of the intervals, this analysis can potentially speed up the data collection substantially. Gathering data for all probes in the first 1MB of chromosome 1-3, for instance, takes much less than 1 minute on the same machine described above. This can be accomplished using the following commands:

```
> iID <- c(1:3)
> iCHR <- c("chr1", "chr2", "chr3")
```

```
> iSTART <- rep(1, 3)
> iEND <- iSTART + 1e+06
> Einter <- AnalyzeTilingCelFiles(dir(pattern = ".cel|.CEL"), "Hs_PromPR_v02-3_N
+     iID = iID, iCHR = iCHR, iSTART = iSTART, iEND = iEND)
```

Note that passed coordinates MUST reference the same genome build as the BPMAP file. There is no way to error check for this and maintain flexibility of the package. Consequently, the onus is on the users to know what data they are putting into the program! For information on obtaining the position of your favorite genomic regions for the proper genome build, see section 3.

2.2 Finding the transcription start site nearest to each probe.

If the tiling array is a promoter array measuring ChIP-chip binding, etc. it may be useful to identify the gene(s) whose transcription start site is nearest to each probe. This process is easy to complete using the AffyTiling package.

2.2.1 Obtain transcript coordinate table for the applicable genome build.

Obtain the desired table of genes/ transcripts corresponding to the proper genome build, as described in section 2.1. When introducing the functions, we will assume that this data is already read into R under the variable KG.

2.2.2 Calling AffyTiling AssociateWithGenes function.

The following code identifies the nearest gene for each probe.

```
> data(KnownGenes)
> colnames(KG)

[1] "name"    "chrom"   "strand"  "txStart" "txEnd"
```

KG contains the position of known genes in the first 2 MB of human chromosomes 1-3.

```
> NearestGenes <- AssociateWithGenes(KG[, 1], KG[, 2], KG[, 3],
+     KG[, 4], KG[, 5], Einter[, 1], Einter[, 3], (as.integer(Einter[,
+         2]) + 13))
> head(NearestGenes)

     ProbeID      GeneID     pgDistance
[1,] "chr1-17008" "AK024448" "2179"
[2,] "chr1-17038" "AK024448" "2149"
[3,] "chr1-17070" "AK024448" "2117"
[4,] "chr1-17101" "AK024448" "2086"
[5,] "chr1-17385" "AK024448" "1802"
[6,] "chr1-17421" "AK024448" "1766"
```

The function returns 3 columns: unique probe Ids, the ID of the nearest gene, and the distance. Distance can be either positive (upstream of the gene) or negative (downstream). Say that instead of the gene nearest to each probe ID, we want all potential transcription start sites within 1kb of each probe. We just have to specify the distance by modifying the previous command as follows (note the addition of D). Notice that by specifying D, multiple genes will be returned for many of the unique probe IDs:

```
> NearestGenes <- AssociateWithGenes(KG[, 1], KG[, 2], KG[, 3],
+     KG[, 4], KG[, 5], Einter[, 1], Einter[, 3], (as.integer(Einter[,
+         2]) + 13), D = 1000)
```

To get all probes close to the Gene AY358517:

```
> GeneOfInterest <- NearestGenes[which(NearestGenes[, 2] == "AY358517"),
+     ]
```

The output is 51 probes within 1kb (either upstream or downstream) of the start of AY358517. To get the expression of these probes, we can use the following:

```
> lapply(GeneOfInterest[, 1], function(x) {
+     Einter[which(Einter[, 1] == x), ]
+ })[1:10]
```

```
[[1]]
         UniqueID              START                CHR   19-103-10009.CEL
     "chr2-276851"           "276851"             "chr2" "3.41716750194584"
  19-104-10013.CEL   19-107-10027.CEL   19-108-10031.CEL
"4.83716078032796" "5.26890598294119" "6.17477099058936"

[[2]]
         UniqueID              START                CHR   19-103-10009.CEL
     "chr2-276887"           "276887"             "chr2" "3.35555657931247"
  19-104-10013.CEL   19-107-10027.CEL   19-108-10031.CEL
"4.27163502997186"  "2.9939887387185" "3.54460227689018"

[[3]]
         UniqueID              START                CHR   19-103-10009.CEL
     "chr2-276922"           "276922"             "chr2" "3.41716750194584"
  19-104-10013.CEL   19-107-10027.CEL   19-108-10031.CEL
"5.28209659154275" "5.97802593648148" "3.18615954013395"

[[4]]
         UniqueID              START                CHR   19-103-10009.CEL
     "chr2-276958"           "276958"             "chr2" "2.94253090681253"
  19-104-10013.CEL   19-107-10027.CEL   19-108-10031.CEL
"2.51454598706153" "5.69863463087323" "2.83309380589835"

[[5]]
         UniqueID              START                CHR   19-103-10009.CEL
     "chr2-276994"           "276994"             "chr2" "3.89015557897744"
  19-104-10013.CEL   19-107-10027.CEL   19-108-10031.CEL
"2.96742361721706" "5.14951274457662" "3.48021197632888"

[[6]]
         UniqueID              START                CHR   19-103-10009.CEL
     "chr2-277027"           "277027"             "chr2" "3.62745879314865"
  19-104-10013.CEL   19-107-10027.CEL   19-108-10031.CEL
"3.62745879314865" "5.26890598294119"  "4.6114146068305"

[[7]]
         UniqueID              START                CHR   19-103-10009.CEL
```

```
     "chr2-277063"           "277063"             "chr2" "3.35555657931247"
  19-104-10013.CEL   19-107-10027.CEL   19-108-10031.CEL
"2.68224601970800" "2.51454598706153" "2.68925629886268"

[[8]]
          UniqueID              START                CHR   19-103-10009.CEL
     "chr2-277100"           "277100"             "chr2" "4.28874828769977"
  19-104-10013.CEL   19-107-10027.CEL   19-108-10031.CEL
  "5.159637781214" "4.35473010707964" "5.84792616854573"

[[9]]
          UniqueID              START                CHR   19-103-10009.CEL
     "chr2-277138"           "277138"             "chr2" "3.80369944083995"
  19-104-10013.CEL   19-107-10027.CEL   19-108-10031.CEL
"2.68224601970800" "6.21365680833106"  "5.6524599619435"

[[10]]
          UniqueID              START                CHR   19-103-10009.CEL
     "chr2-277174"           "277174"             "chr2" "6.17957306559713"
  19-104-10013.CEL   19-107-10027.CEL   19-108-10031.CEL
"2.89690448653305" "5.02076759759244" "3.35555657931247"
```

Finally, we can plot the intensity of the first sample in the region of AY358517 as follows.

```
> ExpSample1 <- as.real(lapply(GeneOfInterest[, 1], function(x) {
+     Einter[which(Einter[, 1] == x), 4]
+ }))
> GenomicPos <- as.integer(lapply(GeneOfInterest[, 1], function(x) {
+     Einter[which(Einter[, 1] == x), 2]
+ }))
> plot(GenomicPos, ExpSample1)
```

3 Acquiring interval data table

If you want to limit your analysis to certain genomic intervals, you will need coordinates for these intervals in the same genome build as the BPMAP file. In most cases, the BPMAP files were created using older genome builds.

Fortunately, coordinates for common genomic features (e.g. genes, exons, promoter regions, CpG islands, etc.) can be obtained for older genome builds using the UCSC genome browser. This section covers downloading a data table for older genome builds and reading this table into R.

3.1 Determining the genome build for a BPMAP file.

Generally, BPMAP files will give the applicable genome build as part of the file name. For instance, the name of the human promoter tiling array BPMAP file used in previous examples assumes that NCBIv34 coordinates are used throughout the analysis.

3.2 A table for the proper genome assembly.

A wide variety of information for older genome builds can be obtained directly using the UCSC genome browser's "Table Browser" feature. UCSC the genome references build number by release date, rather than NCBI version. A table giving the release date of a particular NCBI version can be found here: http://genome.ucsc.edu/FAQ/FAQreleases. On this table, we can find that NCBI build 34 (for the BPMAP example outlined in section 2.1.A) was released July 2003.

Next, we can obtain interval information about our topic of interest. To obtain information about CpG islands in the July 2003 build: 1. Go to the UCSC genome browser web page: http://genome.ucsc.edu. 2. Click the "Table Browser" link. 3. Under "assembly:", change to July 2003. 4. Under "group", change to "Expression and Regulation". 5. Under "track", change to "CpG Islands". 6. Click "get output" 7. Wait for the page to finish downloading. 8. Save the table to your local drive, perhaps "CpG-Table.tsv".

Examining the file in the browser, we see that chromosome, start, and end of each CpG island can be found in columns 1-3, respectively.

3.3 Read data table in R.

Use the following command to read the data table into R.

```
> CpG <- read.table("CpG-Table.tsv", header = TRUE, sep = "\t")
```

Now, the chromosome, start, and end of each CpG island are stored in CpG[,1:3], respectively.

3.4 Convert coordinates to newer genome builds.

This is possible, but beyond the scope of this document. The UCSC genome browser has some tools to make this less painful. You can find more information here: http://genome.ucsc.edu/FAQ/FAQdownloads

4 Further reading.

Users may also want to consult the packages ACME or tilingArray for additional analysis options.

5 References.