Using the ACME package

Sean Davis[‡*]

September 23, 2007

[‡]Genetics Branch
National Cancer Institute
National Institutes of Health

Contents

1 Overview of ACME

Data obtained from high-density oligonucleotide tiling arrays present new computational challenges for users. ACME (Algorithm for Capturing Microarray Enrichment) is a method for determing genomic regions of enrichment in the context of tiling microarray experiments. ACME identifies signals or "peaks" in tiled array data using a user-defined sliding window of n-base-pairs and a threshold (again, user-defined) strategy to assign a probability value (p-value) of enrichment to each probe on the array. This approach has been applied successfully to at least two different genomic applications involving tiled arrays: ChIP-chip and DNase-chip. However, it can potentially be applied to tiling array data whenever regions of relative enrichment are expected.

The ACME algorithm is quite straightforward. Using a user-defined quantile of the data, called the threshold, any probes in the data that are above that threshold are considered positive probes. For example, if a user chooses a threshold of 0.95, then, of course, 5 percent of the total data are going to be positive probes. To look for enrichment, a sliding window of fix number of base pairs (the chosen window size) is examined centered on each probe. Enrichment is calculated using a chi-square of the number of expected positive probes in the window as compared to the expected number. A p-value is then assigned to each probe. Note that these p-values are not corrected for multiple comparisons and should be used as a guide to determining regions of interest rather than a strict statistical significance level.

[*]sdavis2@mail.nih.gov

2 Getting Started using ACME

```
> library(ACME)
```

This loads the ACME library.

Now, we will generate three chromosomes of random data, with probe spacing at 100 base pairs, on three samples.

```
> samps <- data.frame(SampleID = 1:3, Sample = paste("Sample",
+     1:3))
> dat <- matrix(rnorm(90000), nc = 3)
> colnames(dat) <- 1:3
> pos <- rep(seq(1, 10000 * 100, 100), 3)
> chrom <- rep(paste("chr", 1:3, sep = ""), each = 10000)
> annot <- data.frame(Chromosome = chrom, Location = pos)
> a <- new("aGFF", data = dat, annotation = annot, samples = samps)
```

Now, `a` is an R data structure (of class *aGFF*) that contains the data from two test GFF files.

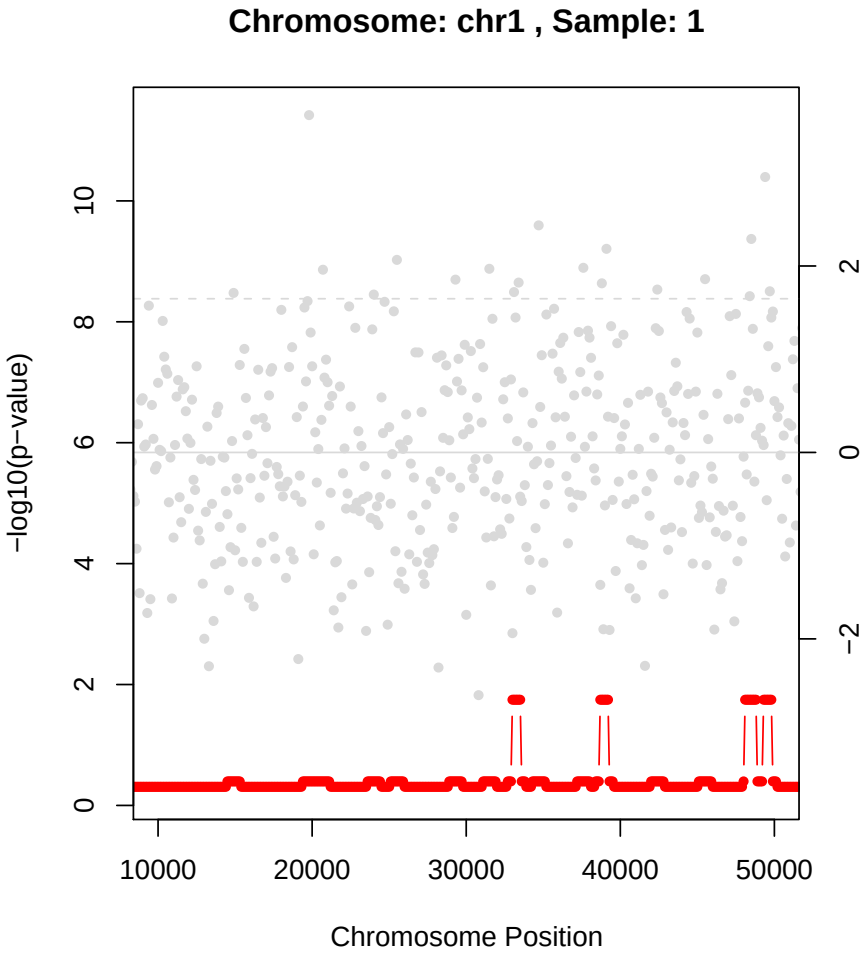
```
> calc <- do.aGFF.calc(a, window = 1000, thresh = 0.95)

Working on sample 1
Working on chromosome chr1
Working on chromosome chr2
Working on chromosome chr3
Working on sample 2
Working on chromosome chr1
Working on chromosome chr2
Working on chromosome chr3
Working on sample 3
Working on chromosome chr1
Working on chromosome chr2
Working on chromosome chr3
```

The function do.aGFF.calc takes as input an aGFF object, a window size (usually 2-3 times the expected fragment size from the experiment and large enough to include about 10 probes, at least), and a threshold, which will be used to determine which probes are counted as positive in the chi-square test.

If desired, the results can be plotted in an R graphics window. The raw signal intensities of each oligonucleotide (Chip/total genomic DNA) will be displayed as grey points; corresponding P values will be displayed in red. The dotted horizontal line represents the threshold as defined in the call to `do.aGFF.calc`. In the following example, R plots the results from an arbitrarily chosen region on chromosome 1, genome coordinates 10,000-50,000.

```
> plot(calc, chrom = "chr1", sample = 1, xlim = c(10000, 50000))
```



One can obtain an overview of an aGFF object by typing the variable name like so:

```
> a

Array GFF object
Number of Samples: 3
Number of probes:  30000

===== Data =====
  Chromosome Location          1         2          3
1       chr1        1  0.4189136 1.5219579  0.74396441
2       chr1      101 -0.2248416 0.3314264  0.38168294
3       chr1      201 -1.3924137 0.2005475  0.09781096
4       chr1      301 -0.8916437 1.9878973 -0.31524729
5       chr1      401  0.4305913 1.1034762 -0.74345112
```

```
With 29995 more rows...

=== Samples ===
  SampleID   Sample
1        1 Sample 1
2        2 Sample 2
3        3 Sample 3


== Annotation ==
  Chromosome Location
1       chr1        1
2       chr1      101
3       chr1      201
4       chr1      301
5       chr1      401
With 29995 more rows...
```

2.1 Generating files for the Affymetrix Integrated Genome Browser

The Affymetrix Integrated Genome Browser (IGB) is a very fast, cross-platform (Java-based) genome browser that can display data in many formats. By generating so-called "sgr" files, one can view both the raw data and the calculated p-values in a fully interactive manner. A simple function, `write.sgr`, will generate such files that can then be loaded into that browser. The function also serves as a model for how to generate other file formats (such as those needed by the UCSC Genome Browser, another fantastic way to view results). With minor modifications, other formats can be generated.

```
> write.sgr(calc)
```

```
./1_thresh0.95.sgr
./1_raw.sgr
./NA_thresh0.95.sgr
./NA_raw.sgr
./NA_thresh0.95.sgr
./NA_raw.sgr
```

```
> write.sgr(calc, raw = FALSE)
```

```
./1_thresh0.95.sgr
./NA_thresh0.95.sgr
./NA_thresh0.95.sgr
```

The function also serves as a model for how to generate other file formats (such as those needed by the UCSC Genome Browser, another fantastic way to view results). With minor modifications, other formats can be generated.