

Using the ACME package

Sean Davis^{‡*}

April 22, 2010

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

Contents

1	Overview of ACME	1
2	Getting Started using ACME	2
2.1	Generating files for the Affymetrix Integrated Genome Browser	4

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 determining 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.

*sdavis2@mail.nih.gov

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.

2 Getting Started using ACME

```
> library(ACME)
```

This loads the ACME library.

To illustrate the package, we begin by loading some example data from two nimblegen arrays. The arrays were custom-designed to assay HOX genes in a ChIP-chip experiment.

```
> datdir <- system.file("extdata", package = "ACME")
> fnames <- dir(datdir)
> example.agff <- read.resultsGFF(fnames, path = datdir)

[1] "Reading /tmp/Rinst6591046780/ACME/extdata/testsamp1.gff"
[1] "Reading /tmp/Rinst6591046780/ACME/extdata/testsamp2.gff"
```

```
> example.agff
```

```
ACMESet (storageMode: lockedEnvironment)
assayData: 190181 features, 2 samples
  element names: exprs
protocolData: none
phenoData
  sampleNames: testsamp1, testsamp2
  varLabels and varMetadata description:
    fullfnames: NA
featureData
  featureNames: 74065, 74066, ..., 103913 (190181 total)
  fvarLabels and fvarMetadata description:
    chromosome: NA
    source: NA
    ...: ...
    comment: NA
    (8 total)
experimentData: use 'experimentData(object)'
Annotation:
```

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

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

Working on sample 1

Working on chromosome:

chr1 chr10 chr11 chr12 chr13 chr14 chr15 chr16 chr17 chr18 chr19 chr2 chr20

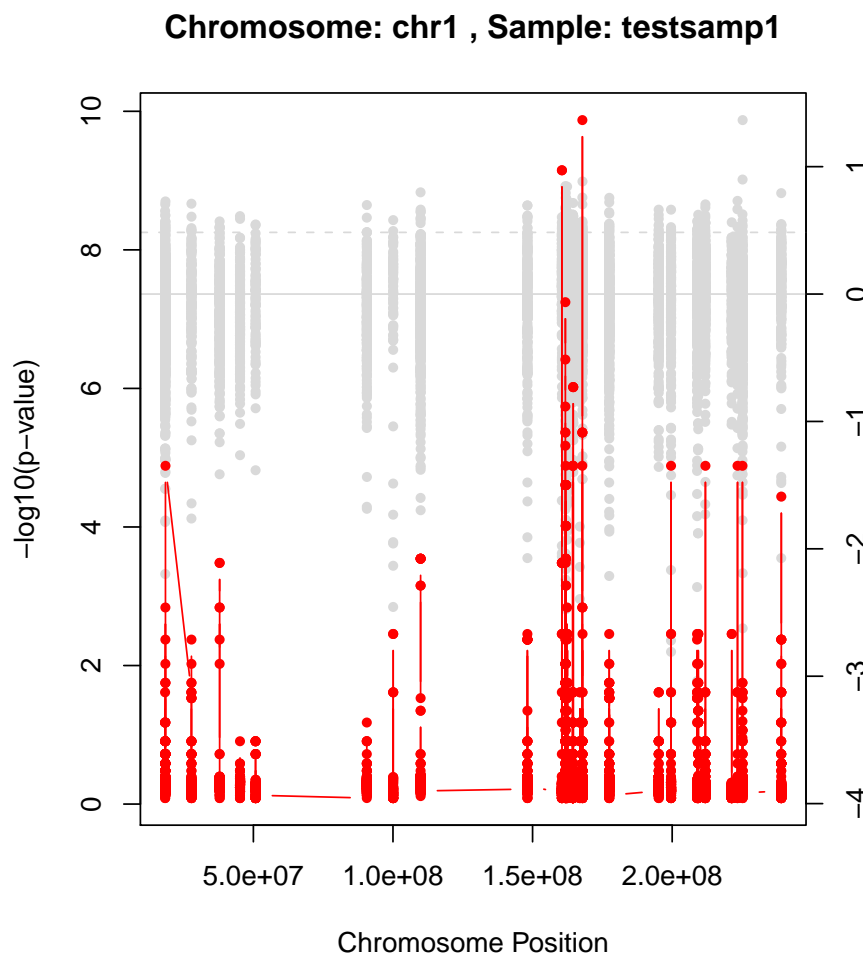
Working on chromosome:

chr1 chr10 chr11 chr12 chr13 chr14 chr15 chr16 chr17 chr18 chr19 chr2 chr20

The function `do.aGFF.calc` takes as input an *ACMESet* 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)
```



And one can find significant regions of interest using:

```
> regs <- findRegions(calc)
> regs[1:5, ]
```

	Length	TF	StartInd	EndInd	Sample	Chromosome	Start
testsamp1.chr1.1	918	FALSE	1	918	testsamp1	chr1	18370933
testsamp1.chr1.2	1	TRUE	919	919	testsamp1	chr1	18515429
testsamp1.chr1.3	1806	FALSE	920	2725	testsamp1	chr1	27803112
testsamp1.chr1.4	2	TRUE	2726	2727	testsamp1	chr1	160510960
testsamp1.chr1.5	183	FALSE	2728	2910	testsamp1	chr1	160512520
	End		Median		Mean		
testsamp1.chr1.1	18514188		5.164139e-01		5.003686e-01		
testsamp1.chr1.2	18515429		1.308413e-05		1.308413e-05		
testsamp1.chr1.3	160504694		4.912989e-01		5.041074e-01		
testsamp1.chr1.4	160511031		7.101277e-10		7.101277e-10		
testsamp1.chr1.5	161743150		6.079601e-01		5.724538e-01		

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)

./testsamp1_thresh0.95.sgr
./testsamp1_raw.sgr
./testsamp2_thresh0.95.sgr
./testsamp2_raw.sgr

> write.sgr(calc, raw = FALSE)

./testsamp1_thresh0.95.sgr
./testsamp2_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.