Making synthetic array data

October 18, 2010

1 Introduction

This vignette describes how to build a complete synthetic microarray dataset for expression data. Situations in which this can be useful are:

- dataset of arbitrary size for demonstration purposes (minimizing the need for data packages)
- dataset for test / monitoring purposes (memory usage, etc)
- simulations

The example here is simple, if not simplistic, and supplementary rules governing the creation of data such as sequence-dependent characteristics of the probe signal or behavior of the signal across the dynamic range can be added.

```
> library("biocDatasets")
```

2 Generating a random array design

2.1 Probe design

We first consider the number of transcripts our synthetic array will be designed to measure signal for and generate sequences for each transcript.

```
> n_transcripts <- 250
> transcripts <- randomDNASequences(n_transcripts, sample(300:10000,
+     n_transcripts, replace = TRUE))
```

Each transcript represents the biological entity that the array was designed to measure.

This section outlines the creation of a short oligonucleotide array. Features or probes in the array are created as random subsequences of transcripts.

```
> n_probes <- 10
> len_probe <- 25
> ir <- vector("list", length = n_transcripts)
> features <- data.frame(seq = I(rep("", n_transcripts * n_probes)),
+     target = rep(as.integer(NA), n_transcripts * n_probes))
> for (i in 1:n_transcripts) {
```

```
+     s <- transcripts[[i]]
+     ir[[i]] <- randomIRanges(n_probes, len_probe, 1, length(s),
+         replace = FALSE)
+     features$seq[(1:n_probes) + (i - 1) * n_probes] <- as.character(msubseq(s,
+         ir[[i]]))
+     features$target[(1:n_probes) + (i - 1) * n_probes] <- i
+ }
```

Obtaining duplicated probe sequences from the procedure above is very unlikely, but it is nevertheless preferable to verify it:

```
> summary(duplicated(features$seq))

   Mode   FALSE    NA's 
logical    2500       0 
```

2.2 Probe positions

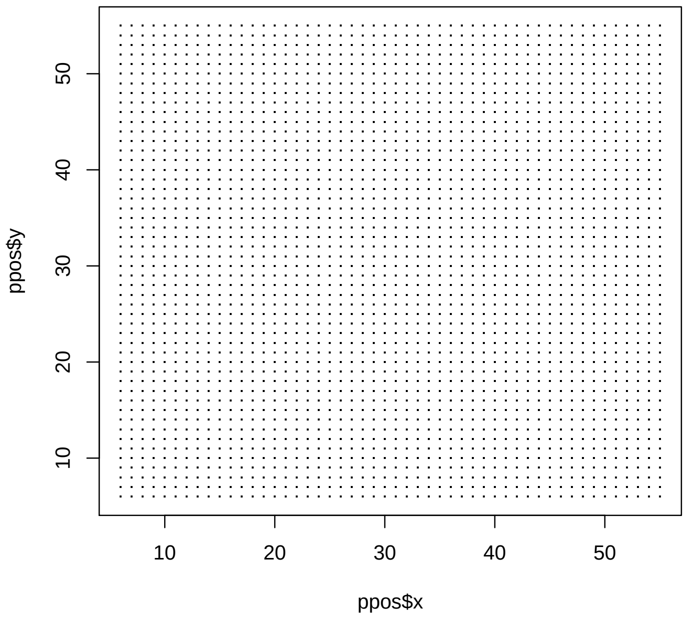
Probes on a microarray are deposited onto 'spots', at given locations on the surface of the slide.

The layout is often a grid, and creating such a layout is easy:

```
> ppos <- createProbeCoords(50, 50)
```

The layout obtained can then be visualized easily:

```
> plot(ppos$x, ppos$y, pch = ".")
```



2.3 Assemble the layout

We can now assemble the feature information (see Section 2.1) with the probe positions.

```
> scramble_i <- sample(seq(along = ppos$x))
> array_layout <- cbind(features, ppos[scramble_i, ])
> row.names(array_layout) <- seq(along = array_layout[[1]])
```

3 Generating random expression-array signal

Here we focus on generating synthetic signal for arrays.

3.1 Replicate the same array

Creating expression array signal for an array is straightforward:

```
> x <- expression_arraywide(50 * 50)
```

Creating synthetic replicates, that is simulate replicate arrays for the same experiment, is equally easy:

```
> x2 <- replicate_arraywide(x)
```

Both array are stored in a matrix, which will constitute the expression matrix at the probe level:

```
> m <- cbind(x, x2)
> colnames(m) <- c(1, 2)
> rownames(m) <- seq(1, 50 * 50)
```

Creating an `ExpressionSet` is now straightforward:

```
> library(Biobase)
> eset <- new("ExpressionSet", featureData = new("AnnotatedDataFrame",
+     array_layout), exprs = m)
```

We are now ready to see how to create a slightly more complex synthetic object.

3.2 Replicates of different arrays

```
> eset2dataframe <- function(eset) {
+     pdataf <- pData(eset)
+     dataf <- lapply(pdataf, function(col) rep(col, rep(nrow(exprs(eset)),
+         nrow(pdataf))))
+     dataf <- as.data.frame(dataf)
+     dataf <- cbind(exprs = c(exprs(eset)), array = rep(rownames(pdataf),
+         each = nrow(exprs(eset))), dataf)
+     return(dataf)
+ }
```

We start by creating two separate array signals, each representing arbitrarily different experimental conditions.

```
> sample_a_seed <- expression_arraywide(50 * 50)
> sample_b_seed <- expression_arraywide(50 * 50)
```

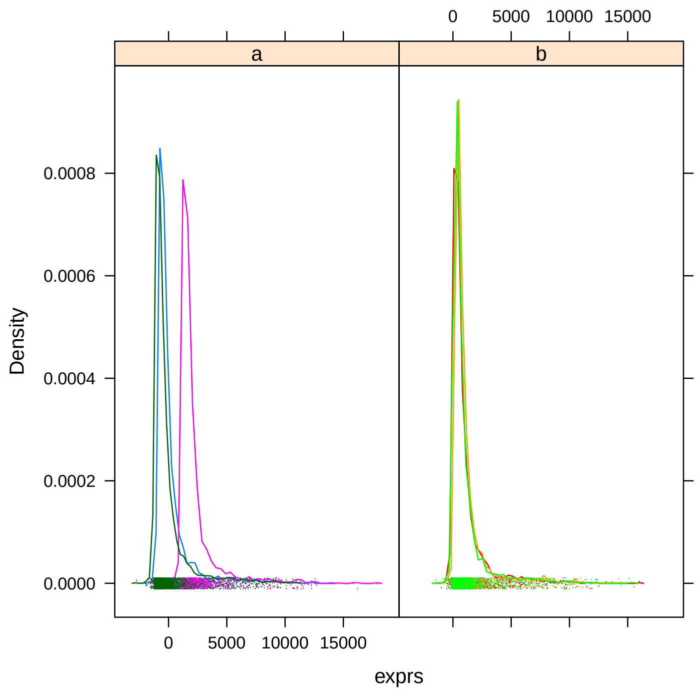
We can then set up a two-sample experiment by replicating each one of the two arrays.

```
> m <- vector("list", length = 6)
> for (i in 1:3) {
+     m[[i]] <- replicate_arraywide(sample_a_seed)
+     m[[i + 3]] <- replicate_arraywide(sample_b_seed)
+ }
> m <- matrix(unlist(m), ncol = 6)
> sample_data <- data.frame(grp = rep(c("a", "b"), c(3, 3)))
> rownames(sample_data) <- 1:6
```

Creating an ExpressionSet is again obvious:

```
> eset <- new("ExpressionSet", featureData = new("AnnotatedDataFrame",
+     array_layout), exprs = m, phenoData = new("AnnotatedDataFrame",
+     sample_data))
```

```
> dataf <- eset2dataframe(eset)
> library(lattice)
> p <- densityplot(~exprs | grp, groups = dataf$array, data = dataf,
+     pch = ".")
> print(p)
```


a
b
Density
exprs