Example workflow for exonmap

MichałJ Okoniewski, Tim Yates, Crispin J Miller

March 26, 2007

Contents

1 Typical workflow

The probeset level expression may be calculated using appropriate CDF files. In the following example, we first identify differentially expressed probesets, then remove any multiply targeted probesets before identifying which genes are targeted by the remaining probeset list.

Mean and variance of the fold changes for well-behaved (i.e. not multiply targeted) exon probesets are then used to identify DE probesets and to prioritise those that have changes in fold change over their length (i.e. putitatively differentially spliced) and those that have consistent fold changes over the entire gene.

```
> library(exonmap)
> raw.data <- read.exon()
> raw.data@cdfName <- "exon.pmcdf"
> x.rma <- rma(raw.data)
> pc.rma <- pc(x.rma, "group", c("a", "b"))
> sigs <- featureNames(x.rma)[(abs(fc(pc.rma)) > 1) & (tt(pc.rma) <
+     1e-04)]
> xmapDatabase("Human")
> sigs.exonic <- select.probewise(sigs, filter = "exonic")
> sigs.intronic <- select.probewise(sigs, filter = "intronic")
> sigs.intergenic <- select.probewise(sigs, filter = "intergenic")
> sigs.mt <- select.probewise(sigs, filter = "multitarget")
> length(sigs.exonic)
> length(sigs.intronic)
> length(sigs.intergenic)
> length(sigs.mt)
> sig.exons.exonic <- probeset.to.exon(sigs.exonic)
> sig.exons.mt <- probeset.to.exon(sigs.mt)
> sig.genes.e <- probeset.to.gene(sigs.exonic)
> sig.genes.mt <- probeset.to.gene(sigs.mt)
> length(sig.genes.e)
> length(sig.genes.mt)
> exmed <- function(e) {
+     this.exon <- exon.to.probeset(e)
+     r <- NA
+     if (length(this.exon) > 0) {
+         print(this.exon)
+         r <- median(fc(pc.rma)[this.exon])
+     }
+     r
+ }
> sumstats <- function(g) {
+     cat(paste("Doing:", g, ".\n"))
+     exons <- gene.to.exon(g)
+     r <- c(NA, NA, NA, NA, NA)
+     if (length(exons > 0)) {
+         meds <- sapply(exons, exmed)
+         meds <- meds[!is.na(meds)]
+         if (length(meds > 0)) {
+             r <- c(min(meds), max(meds), mean(meds), median(meds),
+                 var(meds))
+         }
+     }
+     r
+ }
> gene.e.stats <- sapply(sig.genes.e[1:10], sumstats)
> o <- order(gene.e.stats[5, ], decreasing = FALSE)
> g <- sig.genes.e[o][1:10]
> plot.gene(g, x.rma, 1:4, 4:6, draw.exon.border = F, colour.transcript = F,
+     scale.to.gene = F)
> o <- order(gene.e.stats[5, ], decreasing = TRUE)
> g <- sig.genes.e[o][1:10]
> plot.gene(g, x.rma, 1:4, 4:6, draw.exon.border = F, colour.transcript = F,
+     scale.to.gene = F)
> plot.gene.graph(g, x.rma, 1:4, 4:6, draw.exon.border = F, scale.to.gene = F)
> si <- splicing.index(x.rma, sig.genes.e[1:50], "group", c("a",
+     "b"))
> plot(exprs(x.rma)[names(si), 1], abs(si), cex = 0.3)
> gene.strip(probeset.to.gene(names(which(si > 0.6))), x.rma, 1:3,
+     4:6, type = "mean-fc")
```