ExonModelStrain: simple linear modeling to detect exon-specific strain differences in Affymetrix Exon Array data

Ted Laderas

February 14, 2011

Contents

1 Introduction

The package *ExonModelStrain* applies two linear models to Affymetrix Exon Array data in order to detect exon-specific differences in expression between two strains. Currently, only Mouse Exon Array 1.0 core probesets are supported.

The first linear model applies to multiple-exon transcripts. For a given transcript or gene, we fit the following linear model:

Expression $\bar{S}$train + Exon + Subject%in%Strain + Exon Strain

Note the Exon:Strain interaction term. For a single exon, the model reduces to:

Expression $\bar{S}$train + Subject%in%Strain

ExonModelStrain applies ANOVA to the fit linear models in an attempt to detect differential alternative splicing between strains. In the case of a multiple-exon transcript, we are interested in finding significant p-values for the interaction term Exon Strain. In this case, expression is significantly different between the two strains for one or more exons.

ExonModelStrain also provides tools and metrics based on the Expression Data that allow for filtering and stratification such as the position of the exon with the largest expression difference between strains as well as visualization of the interaction plots between the two strains.

2 Implementation Details

In order to achieve this, an Ensembl-Exon Array mapping (*mouseexonensembl.db*) was built, mapping Ensembl Transcript IDs, Gene IDs, and Exon IDs to probeset boundaries. A probeset was mapped to a particular entity (exon, transcript, etc) if either the start or end of that probeset was within the boundary of that entity. Note that the mapping is many-to-many: that is, a probeset may be a member of multiple exons, due to transcriptional structure (for example, two separate transcripts belonging to a gene may contain slightly different but overlapping exons).

Note this approach is similar to that taken by the *exonmap* project. However, our database package differs in that we only map exonic (not intronic) regions, and in implementation (we use a SQLite based database rather than a MySQL based database). The current version of ExonModelStrain allows the user to connect with exonmap database installations (Mouse, Human, and Rat) in order to utilize their updated mapping. For more details on installing exonmap, please refer to the *exonmap* documentation.

3 Preprocessing Exon Array Data

ExonModelStrain uses as input an ExpressionSet and a list of Ensembl Transcript or Gene IDs to do the analysis. It also requires a phenoData object that contains a column called table.

Exon Array CEL files can be loaded by `ReadAffy`, Normalized and Summarized using current Bioconductor tools (use of the annotation packages available at `http://xmap.picr.man.ac.uk/download/` are recommended. We do recommend SNP masking any SNPs that are different between the two strains. For large datasets and computers with limited memory, use of the *aroma.affymetrix* is suggested.

The following script will preprocess, normalize, SNP mask and produce probeset-level summaries for a set of CEL files in the current working directory.

(Note that the `ReadExon` function from the *exonmap* could also be used as well, provided that an annotation file called *covdesc* exists in the working directory. This file should be a space delimited file with a line for each array, and a column called Strain where each sample is labeled either 1 or 2. For more details, please refer to *exonmap*.)

```
> library(mouseexonpmcdf)
> library(mouseexonensembl.db)
> library(affy)
> raw.data <- ReadAffy()
> raw.data@cdfName <- "mouseexonpmcdf"
> abatch1 <- bg.correct.rma(raw.data)
> abatch2 <- normalize.AffyBatch.quantiles(abatch1)
> mask3 <- function(x, maskfile = "b6vsd2snpmask.txt") {
+     mask <- read.delim(maskfile)
+     probes <- mask[, 2]
+     intensity(x)[probes, ] <- NA
+     return(x)
+ }
> abatch3 <- mask3(abatch2)
> eset <- computeExprSet(abatch3, pmcorrect = "pmonly", summary.method = "medianpolis
+     summary.param = list(na.rm = TRUE))
> library(convert)
> eset <- as(eset, "ExpressionSet")
> coreprobesets <- getCoreProbesets()
> eset <- eset[coreprobesets, ]
```

4 Attaching phenoData to the ExpressionSet

To this data file, we need to attach an appropriate phenoData file. This file is an annotation file where each sample is represented and the strain is annotated. This annotation file called *covdesc* should exists in the working directory. This file should be a space delimited file with a line for each array, and a column called strain where each sample is labeled either 1 or 2. For more details, please refer to *exonmap*.)

```
> pData <- read.delim("covdesc")
> phen <- new("AnnotatedDataFrame", data = pData)
> phenoData(eset) <- phen
```

Alternatively, we can also build a data frame directly in R, based on the sample names. For example, say the first ten samples correspond to strain 1 and the next 10 samples correspond to strain 2.

```
> n <- sampleNames(eset)
> Strain <- c(rep(1, 10), rep(2, 10))
> names(Strain) <- n
> pData <- as.data.frame(Strain)
> phen <- new("AnnotatedDataFrame", data = pData)
> phenoData(eset) <- phen
```

5 Connecting/Disconnecting to a Mapping Database

Currently, two databases mapping ensembl transcripts to Exon probesets are supported by the *ExonModelStrain* package: *mouseexonenesmbl.db*, a portable SQLite database and *exonmap*, a MySQL database available at: (http://xmap.picr.man.ac.uk/) Note that a local installation of the Xmap database is highly recommended, as *ExonModelStrain* makes many queries of the database.

We will connect to the *mouseexonenesmbl.db* database in our example using the following

```
> library(ExonModelStrain)
> mapConnect(dbPackage = "mouseexonensembl.db")
```

In order to connect to the Xmap database, you would instead use the following lines. dbName is derived from the name of the database that is setup when you specify a databases.txt file in your home directory.

```
> library(ExonModelStrain)
> mapConnect(dbPackage = "xmapcore", dbName = "mouse")
```

Note that if you would like to switch databases, first disconnect from the previous database by using `mapDisconnect` and then connect to the new database using `mapConnect`.

6 Analysing the Expression Set for Differential Exon Expression

Now that we have an appropriate ExpressionSet and we are connected to a database mapping, the Core Expression probesets can now be analysed for differential exon expression using `RunExonModelWorkflow`.

First we examine the list of Transcript IDs.

```
> data(exontestdata)
> testTrans
```

```
 [1] "ENSMUST00000086675" "ENSMUST00000025403" "ENSMUST00000079776"
 [4] "ENSMUST00000089419" "ENSMUST00000062893" "ENSMUST00000079749"
 [7] "ENSMUST00000026901" "ENSMUST00000100498" "ENSMUST00000008733"
[10] "ENSMUST00000100538" "ENSMUST00000027090" "ENSMUST00000043863"
[13] "ENSMUST00000022742" "ENSMUST00000103506" "ENSMUST00000061437"
[16] "ENSMUST00000079465" "ENSMUST00000081945" "ENSMUST00000060522"
[19] "ENSMUST00000086552" "ENSMUST00000026743"
```

Now we run the model on the 20 transcripts. `RunExonModelStrain` is smart enough to know to run the single-exon model on single-exon transcripts, and the multiple-exon model on multiple-exon transcripts.

```
> results <- RunExonModelWorkflow(TestSetTrans, testTrans)

[1] "running 1 ENSMUST00000086675"
Analysis of Variance Table

Response: Expression
               Df  Sum Sq Mean Sq F value Pr(>F)
Strain          1 0.87696 0.87696
Strain:Subject 20 1.88471 0.09424
Residuals       0 0.00000
[1] "running 2 ENSMUST00000025403"
Analysis of Variance Table

Response: Expression
                Df  Sum Sq Mean Sq F value Pr(>F)
Strain           1   0.085   0.085  0.1384 0.7102
Exon             7 290.618  41.517 67.7111 <2e-16 ***
Strain:Subject  20   4.895   0.245  0.3992 0.9908
Strain:Exon      7   0.710   0.101  0.1654 0.9917
Residuals      228 139.797   0.613
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
[1] "running 3 ENSMUST00000079776"
Analysis of Variance Table

Response: Expression
                Df  Sum Sq Mean Sq F value    Pr(>F)
Strain           1   4.522  4.5220  7.2983 0.0073142 **
Exon            11 212.319 19.3017 31.1519 < 2.2e-16 ***
Strain:Subject  20   4.002  0.2001  0.3230 0.9978354
Strain:Exon     11  21.689  1.9717  3.1822 0.0004308 ***
```

```
Residuals      286 177.205  0.6196
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
[1] "running 4 ENSMUST00000089419"
Analysis of Variance Table

Response: Expression
               Df  Sum Sq  Mean Sq F value Pr(>F)
Strain          1  0.1440 0.143992  0.4761 0.4938
Strain:Subject 20  1.3222 0.066108  0.2186 0.9997
Residuals      44 13.3062 0.302414
[1] "running 5 ENSMUST00000062893"
Analysis of Variance Table

Response: Expression
                Df  Sum Sq Mean Sq F value  Pr(>F)
Strain           1   0.567  0.5672  4.2889 0.03867 *
Exon            41 298.366  7.2772 55.0284 < 2e-16 ***
Strain:Subject  20   4.561  0.2280  1.7243 0.02513 *
Strain:Exon     41   5.247  0.1280  0.9678 0.53003
Residuals      820 108.441  0.1322
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
[1] "running 6 ENSMUST00000079749"
Analysis of Variance Table

Response: Expression
               Df  Sum Sq Mean Sq F value Pr(>F)
Strain          1   0.416  0.4162  0.0756 0.7859
Strain:Subject 20   4.438  0.2219  0.0403 1.0000
Residuals      22 121.032  5.5014
[1] "running 7 ENSMUST00000026901"
[1] "running 8 ENSMUST00000100498"
Analysis of Variance Table

Response: Expression
                Df  Sum Sq Mean Sq F value Pr(>F)
Strain           1   0.107  0.1069  0.1884 0.6645
Exon            11 306.517 27.8652 49.1295 <2e-16 ***
Strain:Subject  20   8.551  0.4276  0.7538 0.7691
Strain:Exon     11   1.163  0.1057  0.1864 0.9982
Residuals      418 237.081  0.5672
```

```
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
[1] "running 9 ENSMUST00000008733"
[1] "running 10 ENSMUST00000100538"
[1] "running 11 ENSMUST00000027090"
Analysis of Variance Table

Response: Expression
               Df Sum Sq  Mean Sq F value Pr(>F)
Strain          1 0.0726 0.072635  0.4997 0.4871
Strain:Subject 20 1.2102 0.060509  0.4163 0.9733
Residuals      22 3.1979 0.145357
[1] "running 12 ENSMUST00000043863"
Analysis of Variance Table

Response: Expression
               Df  Sum Sq  Mean Sq F value Pr(>F)
Strain          1 0.05350 0.053495
Strain:Subject 20 0.33921 0.016960
Residuals       0 0.00000
[1] "running 13 ENSMUST00000022742"
Analysis of Variance Table

Response: Expression
                Df  Sum Sq Mean Sq  F value    Pr(>F)
Strain           1    3.36  3.3577  24.7179 8.463e-07 ***
Exon            32 1005.96 31.4363 231.4227 < 2.2e-16 ***
Strain:Subject  20   22.27  1.1137   8.1983 < 2.2e-16 ***
Strain:Exon     32    8.58  0.2681   1.9736  0.001247 **
Residuals      662   89.93  0.1358
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
[1] "running 14 ENSMUST00000103506"
[1] "running 15 ENSMUST00000061437"
[1] "running 16 ENSMUST00000079465"
Analysis of Variance Table

Response: Expression
               Df Sum Sq Mean Sq F value Pr(>F)
Strain          1 0.1748 0.17478  0.5169 0.4797
Strain:Subject 20 5.8596 0.29298  0.8665 0.6242
Residuals      22 7.4383 0.33811
```

```
[1] "running 17 ENSMUST00000081945"
Analysis of Variance Table

Response: Expression
              Df  Sum Sq Mean Sq F value    Pr(>F)
Strain         1   1.517   1.517  11.014  0.001541 **
Exon           3 261.788  87.263 633.539 < 2.2e-16 ***
Strain:Subject 20   3.256   0.163   1.182  0.300859
Strain:Exon    3   1.191   0.397   2.882  0.043142 *
Residuals     60   8.264   0.138
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
[1] "running 18 ENSMUST00000060522"
Analysis of Variance Table

Response: Expression
               Df  Sum Sq Mean Sq F value    Pr(>F)
Strain          1   4.700  4.6996  8.7149 0.0033717 **
Exon           14 245.877 17.5627 32.5680 < 2.2e-16 ***
Strain:Subject  20   5.218  0.2609  0.4838 0.9718319
Strain:Exon     14  23.236  1.6597  3.0778 0.0001593 ***
Residuals      346 186.585  0.5393
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
[1] "running 19 ENSMUST00000086552"
Analysis of Variance Table

Response: Expression
               Df Sum Sq Mean Sq  F value  Pr(>F)
Strain          1   0.25   0.247   0.4331 0.51105
Exon           10 779.78  77.978 136.5538 < 2e-16 ***
Strain:Subject  20   6.67   0.333   0.5839 0.92238
Strain:Exon     10  11.05   1.105   1.9342 0.04091 *
Residuals      266 151.90   0.571
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
[1] "running 20 ENSMUST00000026743"
Analysis of Variance Table

Response: Expression
               Df Sum Sq Mean Sq  F value    Pr(>F)
Strain          1   1.50  1.5018   5.2705 0.0223619 *
```

```
Exon            11 345.63 31.4207 110.2680 < 2.2e-16 ***
Strain:Subject  20  14.63  0.7316   2.5674 0.0003226 ***
Strain:Exon     11   1.34  0.1215   0.4264 0.9438613
Residuals      308  87.76  0.2849
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
> multres <- results$multi
> multres
```

```
                  ID      pStrain         pExon  pExonStrain Strain1mean
1 ENSMUST00000025403 7.102346e-01  3.459671e-52 0.9916593093   3.5752386
2 ENSMUST00000079776 7.314176e-03  8.482543e-43 0.0004307697   1.1273036
3 ENSMUST00000062893 3.867432e-02 2.436248e-205 0.5300290535   0.5617597
4 ENSMUST00000100498 6.644750e-01  2.065729e-68 0.9982457453   5.4915030
5 ENSMUST00000022742 8.463380e-07  0.000000e+00 0.0012467615   4.9007424
6 ENSMUST00000081945 1.540824e-03  2.304691e-45 0.0431422921   2.2731364
7 ENSMUST00000060522 3.371663e-03  1.013017e-54 0.0001592529   1.0145143
8 ENSMUST00000086552 5.110532e-01  1.187603e-98 0.0409138114   5.5789351
9 ENSMUST00000026743 2.236194e-02 8.160432e-100 0.9438613049   8.5145395
  Strain2mean multprobeflag            maxexon maxexondelta position
1   3.5392338             1 ENSMUSE00000143043    0.2437343        7
2   1.3623970             1 ENSMUSE00000125970    2.0971999        6
3   0.6115172             0 ENSMUSE00000497210    0.3844675       22
4   5.5220461             1 ENSMUSE00000648227    0.2111199        5
5   4.7661873             1 ENSMUSE00000480826    1.0104467       30
6   2.5368238             0 ENSMUSE00000549804    0.5789176        1
7   1.2332985             1 ENSMUSE00000125970    2.0971999        8
8   5.5220272             1 ENSMUSE00000416166    0.8925565        2
9   8.3833594             1 ENSMUSE00000152363    0.2271807        5
  positionflag numexonsmapped trueexonnum missingexonflag strand
1            0              8           8               0      1
2            0             12          12               0     -1
3            0             42          46               1      1
4            0             12          12               0     -1
5            0             33          35               1      1
6            1              4           7               1     -1
7            0             15          15               0     -1
8            0             11          11               0     -1
9            0             12          13               1      1
```

We can also look at the single exon results.

```
> sing <- results$singles
> sing
```

```
                  ID   pStrain Strain1mean Strain2mean numprobesets
1 ENSMUST00000086675       NaN  4.34236794  3.94139946           22
2 ENSMUST00000089419 0.4937983  0.14176213  0.04795627           66
3 ENSMUST00000079749 0.7858535  4.50880208  4.31348647           44
4 ENSMUST00000027090 0.4870519  0.04899534  0.13059337           44
5 ENSMUST00000043863       NaN 10.09785501  9.99882233           22
6 ENSMUST00000079465 0.4797143  0.49858350  0.37200612           44
             maxexon maxexondelta numexonsmapped trueexonnum missingexonflag
1 ENSMUSE00000657256   0.40096848              1           1               0
2 ENSMUSE00000558095   0.09380586              1           1               0
3 ENSMUSE00000464585   0.19531561              1           1               0
4 ENSMUSE00000354677   0.08159804              1           1               0
5 ENSMUSE00000352751   0.09903269              1           1               0
6 ENSMUSE00000464194   0.12657738              1           1               0
  strand
1     -1
2     -1
3     -1
4     -1
5     -1
6     -1
```

Finally, we can get a list of those transcripts that were not modeled. These transcripts may not have the representative probesets that exist in our data.

```
> results$notrun
```

```
[1] "ENSMUST00000026901" "ENSMUST00000008733" "ENSMUST00000100538"
[4] "ENSMUST00000103506" "ENSMUST00000061437"
```

7 Running the Probeset-Level Model

For datasets that require higher sensitivity in the comparison of AEU events, a probeset-level model is also supplied. This model uses the following formula:

Expression $\bar{\text{S}}$train + Probeset + Subject%in%Strain + Probeset Strain

```
> results2 <- RunExonModelWorkflow(TestSetTrans, testTrans, analysisUnit = "probeset"
```

Note that `PlotExonResults` also has the option of specifying the analysisUnit, which will plot the interaction plot sorted 5' to 3' by probeset. The Ensembl Exon ID is also appended to the probeset name under the plot.

```
> PlotExonResults(results2, analysisUnit = "probeset")
```

8 Adjusting for Multiple Comparisons

A convenience function, `RunQVals` is provided to adjust the p-values for multiple comparison. `RunQVals` utilizes the *qvalue* in order to adjust the p-values for False Discovery Rate (FDR). In our example, qvalue cannot estimate the pi0 term of the raw Exon p-values (because there are so few transcripts in our example) and thus returns a null column for the qvalues.

```
> multqv <- RunQVals(multres)

[1] "ERROR: The estimated pi0 <= 0. Check that you have valid p-values or use another l
[1] "qvalues could not be calculated for pExon"

> multqv
```

	resultframe[, 1]	pStrain	pExon	pExonStrain	qStrain	qExon
1	ENSMUST00000025403	7.102346e-01	3.459671e-52	0.9916593093	8.269903e-03	NA
2	ENSMUST00000079776	7.314176e-03	8.482543e-43	0.0004307697	1.916225e-04	NA
3	ENSMUST00000062893	3.867432e-02	2.436248e-205	0.5300290535	6.754800e-04	NA
4	ENSMUST00000100498	6.644750e-01	2.065729e-68	0.9982457453	8.269903e-03	NA
5	ENSMUST00000022742	8.463380e-07	0.000000e+00	0.0012467615	8.869210e-08	NA
6	ENSMUST00000081945	1.540824e-03	2.304691e-45	0.0431422921	8.073541e-05	NA
7	ENSMUST00000060522	3.371663e-03	1.013017e-54	0.0001592529	1.177780e-04	NA
8	ENSMUST00000086552	5.110532e-01	1.187603e-98	0.0409138114	7.650840e-03	NA
9	ENSMUST00000026743	2.236194e-02	8.160432e-100	0.9438613049	4.686845e-04	NA

	qExonStrain	Strain1mean	Strain2mean	multprobeflag	maxexon
1	0.998245745	3.5752386	3.5392338	1	ENSMUSE00000143043
2	0.001938464	1.1273036	1.3623970	1	ENSMUSE00000125970
3	0.795043580	0.5617597	0.6115172	0	ENSMUSE00000497210
4	0.998245745	5.4915030	5.5220461	1	ENSMUSE00000648227
5	0.003740284	4.9007424	4.7661873	1	ENSMUSE00000480826
6	0.077656126	2.2731364	2.5368238	0	ENSMUSE00000549804
7	0.001433276	1.0145143	1.2332985	1	ENSMUSE00000125970
8	0.077656126	5.5789351	5.5220272	1	ENSMUSE00000416166
9	0.998245745	8.5145395	8.3833594	1	ENSMUSE00000152363

	maxexondelta	position	positionflag	numexonsmapped	trueexonnum	missingexonflag
1	0.2437343	7	0	8	8	0
2	2.0971999	6	0	12	12	0
3	0.3844675	22	0	42	46	1
4	0.2111199	5	0	12	12	0
5	1.0104467	30	0	33	35	1
6	0.5789176	1	1	4	7	1
7	2.0971999	8	0	15	15	0

```
8    0.8925565        2            0             11          11               0
9    0.2271807        5            0             12          13               1
  strand
1      1
2     -1
3      1
4     -1
5      1
6     -1
7     -1
8     -1
9      1
```

9 Filtering the Result Sets

Say we are interested in finding Strain-specific Exon interactions. We can find possible interactions by filtering our resulting dataframe.

```
> attach(multqv)
> sig <- multqv[qExonStrain < 0.05, ]
> detach(multqv)
> sig
```

```
    resultframe[, 1]      pStrain        pExon  pExonStrain      qStrain qExon
2 ENSMUST00000079776 7.314176e-03 8.482543e-43 0.0004307697 1.916225e-04    NA
5 ENSMUST00000022742 8.463380e-07 0.000000e+00 0.0012467615 8.869210e-08    NA
7 ENSMUST00000060522 3.371663e-03 1.013017e-54 0.0001592529 1.177780e-04    NA
  qExonStrain Strain1mean Strain2mean multprobeflag            maxexon
2 0.001938464    1.127304    1.362397             1 ENSMUSE00000125970
5 0.003740284    4.900742    4.766187             1 ENSMUSE00000480826
7 0.001433276    1.014514    1.233299             1 ENSMUSE00000125970
  maxexondelta position positionflag numexonsmapped trueexonnum missingexonflag
2     2.097200        6            0             12          12               0
5     1.010447       30            0             33          35               1
7     2.097200        8            0             15          15               0
  strand
2     -1
5      1
7     -1
```

Here we see two transcripts with significant exon/strain interaction terms.

There are other flags that exist in the resulting data frame that can be used to further stratify/filter the data. For example, `multprobeflag` is a flag that indicates

whether there is at least one exon in the transcript that has multiple mapped probesets. `missingexonflag` indicates whether there are exons in the transcript that are not mapped to the data. `MissingExons` will return a list of exons not currently mapped to the data.

For more details, refer to `RunExonModelWorkflow`.

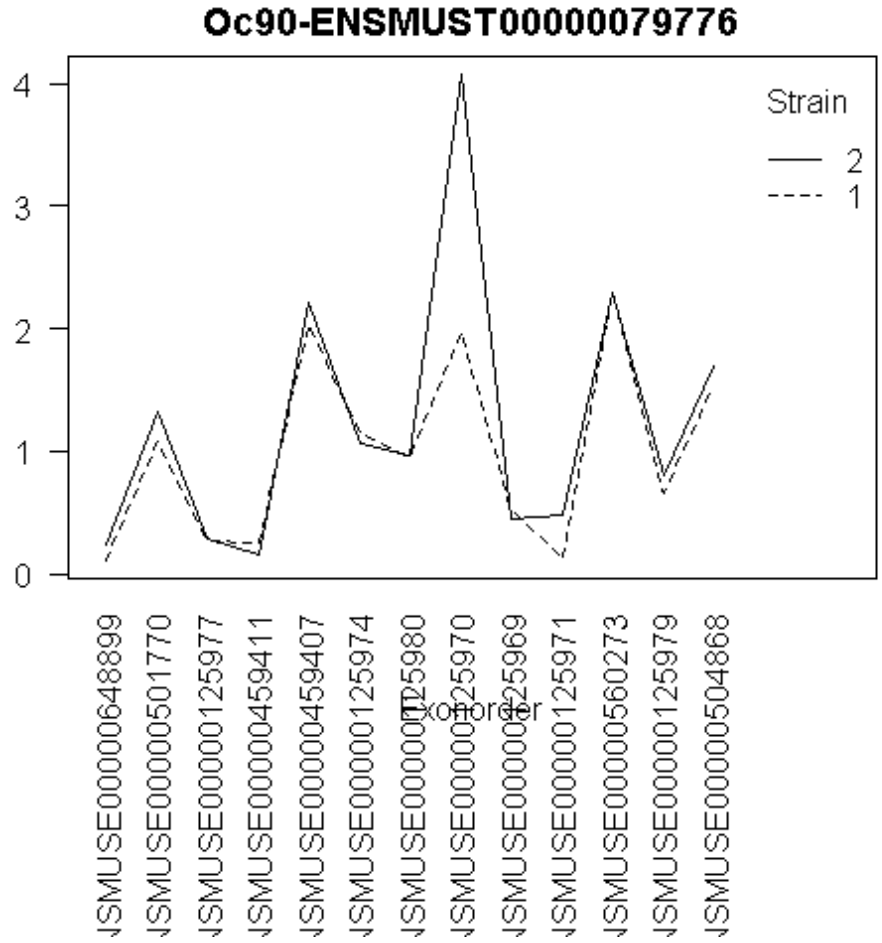
10 Plotting the Result Sets

We have two possible transcripts with exon/strain interactions. We should now examine the interaction plots to see whether these exon specific differences.

Note that these plots may not be representative of all exons in a transcript. That is, if no probesets exist in the ExpressionSet that map to an exon, that exon is not represented in the graph. To get a list of missing exons for a transcript, please refer to `MissingExons`.

There are two ways to plot these results. The first is to plot by id.

```
> PlotExon("ENSMUST00000079776", ExpSet = TestSetTrans)
```



Note the large expression difference between the two strains at Exon ENMUSE00000125970.

Or, if we have a large number of significant results, we can save the plots in the working directory for later examination. These plots are named automatically, by gene symbol + ID (for example,

```
> PlotExonResults(sig, ExpSet = TestSetTrans, savePlot = TRUE)

[1] "ENSMUST00000079776"
[1] "plot is available as c:/Documents and Settings/Ted Laderas/My Documents/My Dropbox
[1] "ENSMUST00000022742"
[1] "plot is available as c:/Documents and Settings/Ted Laderas/My Documents/My Dropbox
[1] "ENSMUST00000060522"
[1] "plot is available as c:/Documents and Settings/Ted Laderas/My Documents/My Dropbox
```

For probeset-level modeling, the option to plot by probeset-level is also provided:

```
> PlotExonResults(sig, ExpSet = TestSetTrans, savePlot = TRUE,
+     analysisUnit = "probeset")

[1] "ENSMUST00000079776"
[1] "plot is available as c:/Documents and Settings/Ted Laderas/My Documents/My Dropbox
[1] "ENSMUST00000022742"
[1] "plot is available as c:/Documents and Settings/Ted Laderas/My Documents/My Dropbox
[1] "ENSMUST00000060522"
[1] "plot is available as c:/Documents and Settings/Ted Laderas/My Documents/My Dropbox
```