The REIDS Package: Vignette

Marijke Van Moerbeke

July 27, 2018

Contents

1 Introduction

This vignette describes how to use the REIDS package starting from the raw microarray data contained in .CEL files. For the computation of the REIDS model we do recommend the use of a supercomputer as it requires memory and time for the data to be processed. All other methods can be conducted in reasonable time on an ordinary laptop but those who have a high-performance computing (HPC) service freely available have the advantage to use this service for the regular analysis as well. We will demonstrate an example step by step after giving a short introduction of the used methodology. The package accompagnies the papers by Van Moerbeke et al. (2017) and Van Moerbeke et al. (2018) which introduce the REIDS model.

2 Methodology

The Random Effects for Identification of Differential Splicing (REIDS) model is situated in a mixed model framework. Alternative splicing detection can now be formulated as variance decomposition in a random effects model with a random intercept per exon taking the gene expression into account. The following can be said. The *between array variability* of an alternatively spliced exon would be higher than the *within array variability* among the exons of the same transcript cluster. A non-alternatively spliced exon would have a between array variability that is at most the within array variability across all exons of the same transcript cluster. These hypotheses can be formulated as a two-stage mixed effect model for alternative splicing detection.

The model is fitted on the observed PM probe intensities levels as:

$$log2(PM_{ijk}) = p_j + c_i + b_{ik} + \epsilon_{ijk}, \tag{1}$$

Here, parameter p_j denotes the effect of probe j, c_i is the overall gene effect of array i and b_{ik} is an exon specific deviation from this overall effect. The background noise $\epsilon_{ijk(j)} \sim N(0, \sigma^2)$ captures the within array variability with σ^2. Differential expression between the arrays is expected to be negated by incorporating the gene effect parameter c_i. By consequence, the exon specific parameters show the deviation of a particular exon from its corresponding gene level. If there is only a small deviation, it can be said that the exon is present in the sample. A large deviation however shows that the exon likely absent. Note that the exon specific deviations are random effects assumed to be normally distributed, $b_{ik} \sim N(\mathbf{0}, \mathbf{D})$. The $K \times K$ covarianc matrix D contains the exon specific signals with τ_k^2 on its diagonal and K the number of exons in a transcript cluster.

The advantage of a mixed model formulation for alternative splicing detection is the existence of a standard score to quantify the trade-off between signal and noise. We term this score an "exon score" but it is also known as the intra-cluster correlation (ICC) of linear mixed models. The score is defined as the ratio of the exon specific signal to the noise across all exons of a transcript cluster. For a probeset k of a transcript cluster this is formulated as:

$$\rho_k = \tau_k^2/(\sigma^2 + \tau_k^2),$$

where σ^2 is the same for all probesets belonging to the same transcript cluster. It intuitively follows from the definition of the exon score that an equity threshold between exon specific signal and transcript level noise is 0.5. This could be used as a threshold for identifying an alternatively spliced probeset among the probesets of a transcript cluster. A value of $\rho_k > 0.5$ puts more weight in favor of probeset k to be alternatively spliced. Note that the threshold for the exon score could be adjusted to the relative amount of signal in the data. Given that probeset k has been identified to have substantial between array variability, we propose to use the estimated random effects b_{ik} as an "array score" for identifying arrays in which the alternatively spliced exon is expressed. Tissues with an enrichment of probeset k are expected to have array scores greater than zero as they resisted shrinkage towards the overall gene level effects.

The parameters of the proposed mixed effects model are estimated within the Bayesian framework with vague proper priors since the full conditional posterior distributions for the parameters of interest are known. More information can be found in Van Moerbeke et al. (2017) and Van Moerbeke et al. (2018).

3 Annotation

Two supplementary files are needed to make optimal use of the REIDS functionality: an annotation file between probe sets, exons and isoforms and a mapping file between probe sets and junctions. A file with a mapping between the probe sets , exon and isoform annotations is required to have the following format:

```
TC0200480_PROM2 PSR020003983 ENSE00003585208 ENST00000431567 1 ENSG00000155066
TC0200480_PROM2 PSR020003983 ENSE00003601191 ENST00000403131 1 ENSG00000155066
TC0200480_PROM2 PSR020003983 ENSE00003601191 ENST00000317668 1 ENSG00000155066
TC0200480_PROM2 PSR020003983 ENSE00003601191 ENST00000317620 1 ENSG00000155066
TC0200480_PROM2 JUC0200032386 ENSE00003546149 3.0 ENST00000403131 1 ENSG00000155066
TC0200480_PROM2 JUC0200032386 ENSE00003546149 3.0 ENST00000317668 1 ENSG00000155066
```

A mapping file between the probe sets and exons is required to have the following format:

```
TC01000001 PSR01000005 JUC01000001 3
TC01000001 PSR01000008 JUC01000001 5
TC01000001 PSR01000007 JUC01000001 exclusion
TC01000001 PSR01000005 JUC01000002 3
TC01000001 PSR01000007 JUC01000002 5
TC01000001 PSR01000002 JUC01000003 3
```

These files can be generated in differently, depending on the preference of the user. We outline three procedures to retrieve the required annotations.

- **The manufacturer's information**

  A mapping between the probe sets and the annotated exons and isoforms can be retrieved from the manufacturer's information. The python script `ExonAndTranscriptAnnotations.py` collects all necessary information from a file with exon and junction probe set annotations. The function was designed according to the lay-out of an Affymetrix file "HTA-2_0.na35.hg19.probeset.csv". Two files are returned to the user. The first output file is given a user specified name and contains the exon and isoform annotations. the second output file ("LineIndexing_EAnnot_TrAnnot.txt") is an indexing file used in the R code to prevent the loading of the entire annotation file.

  ```
  $ python ExonAndTranscriptAnnotations.py HTA-2_0.na35.hg19.probeset
      .csv HTA_2_0_ExonAndTranscriptAnnotations.txt
  ```

  The mapping between the probe sets and the junctions can be retrieved from the exon annotation. The function `JunctionAssociations.py` creates the corresponding mapping which can be given any name and a line indexing file ("LineIndexing_JAnnot.txt").

  ```
  $ python JunctionAnnotations.py HTA_2_0_ExonAndTranscriptAnnotations
      .txt HTA_2_0_JunctionAnnotations.txt
  ```

  Alternatively, the mapping between the probe sets and junctions can be performed internally by the functions in the package as long as the annotation file is provided. Therefore, it is not a requirement to execute `JunctionAnnotations.py`.

- **The Direct Ensembl Connection**

  A direct Ensembl annotation can be achieved by making an remote connection with the Ensembl database while performing the REIDS function. The connection is established with the `biomaRt` package. The required information consist of the gene names for the transcript ID's and the probe set position. This can be found in the manufacturer's provided data files.

The files are "HTA-2_0.na35.2.hg19.transcript.csv" and "HTA-2_0.na35.2.hg19.probeset.csv" for the Affymetrix microarrays.

```
> library(data.table)
> transcripts<-fread("HTA-2_0.na35.2.hg19.transcript.csv",
    skip=19,header=TRUE,sep=",")
> transcripts=as.data.frame(transcripts)
> transcriptData=transcripts[,c(1,4,8)]
> save(transcriptData,file="transcriptData.RData")

> probesets <- fread("HTA-2_0.na35.hg19.probeset.csv",
    skip=19,header=TRUE,sep=",")
> probesets =as.data.frame(probesets)
> positionData <- probesets[probesets[,13] == "main",c(1,2,4,5,6,7)]
> save(positionsData,file="positionsData.RData")
```

The Ensembl information is retrieved based on the gene name. A mapping of the probe sets to exons is based on the positions of the probe sets. This allows to create the necessary annotations between probe sets, exons and isoforms. However, since no chromosome location is available for the junctions it is still necessary to perform `ExonAndTranscriptAnnotations.py` function in order to achieve a mapping between the probe sets and junctions. This output file can be passed to the internal REIDS functions and a mapping will be deduced from the manufacturer's exon annotations.

- **The REMAP Annotations**

  If the REMAP tool was executed, the remapping of probe sets and junctions can be used as input. In order to extract the REMAP information in the correct format the function `REMAP_ExtractInfo.py` can be performed on the output files "GeneInfo.csv" and GeneAllInfo.csv". It will create a line indexing file ("REMAP_Indices") as well as an output file with a user specified name containing the retrieved annotations. Further, if a transcript ID is annotated to multiple genes and the probe sets are divided across the genes a file "TC_Gene_SplitFile.txt" is created containing this information. The file can be used later on in the analysis to make a correct distinction between the genes. A mapping between the probe sets and the junctions will be retrieved later internally by the function.

```
$ python REMAP_ExtractInfo.py REMAP/GeneAllInfo.txt REMAP/GeneInfo
    .csv HTA_2_0_ExonAndTranscriptAnnotations.txt
```

Note that all functions were written to the example of the HTA-2.0 Affymetrix' annotation files. Therefore, for a different type of array of manufacturer it is recommended to study a small example to see if the information to be collected is at the same position in the lines.

4 Creating the .CDF file

The .CDF file necessary to process the array with the R package aroma.affymetrix is created by combining the .clf en .pgf file of the Affymetrix library. Credit goes to the arome.affymetrix maintainers and writes of the necessary function which I bundle here. The following steps are required to create the .CDF file:

1. Combine the .pgf and .clf information with "combineProbeInfo.pl":

```
$ perl combineProbeInfo HTA-2_0.r3
```

The output consists of two files: "HTA-2_0.r3.probeflat" and "HTA-2_0.r3.psr".

2. Run "ID_Name_TC.py" to combine probe set ID, probe set name and TC ID. This function retrieves information of "HTA-2_0.na35.hg19.probeset.csv" and the "HTA-2_0.r3.psr" file.

```
$ python ID_Name_TC.py
```

   The output is one file: "PSID_Name_TCID.txt".

3. Create an empty file .flat file and run "addGeneId_2.pl" on the "HTA-2_0.r3.probeflat" file and "PSID_Name_TCID.txt" with the empty file as third argument.

```
$ perl addGeneId_2 HTA-2_0.r3.probeflat PSID_Name_TCID.txt HTA-2_0.r3.flat
```

   The output will be written to "HTA-2_0.r3.flat".

4. Run "Flat2CDF.R" in R on the created outfile to obtain the .CDF file. The output file is "HTA-2_0,r3.cdf".

(Note: I had to rename "HTA-2_0,r3.cdf" to "HTA-2_0,r.cdf" in order for aroma.affymetrix to recognize the .cdf file and process the .CEL files)

5 Preprocessing

The preprocessing of the information uses functions of the `aroma.affymetrix` (Bengtsson et al., 2008) package. In order to obtain data for the REIDS analysis the raw .CEL files are background corrected with the rma background correction and normalization is performed with the quantile normalization. The result is a data frame with a row for every individual probe. The first column contains the gene IDs and the second column the probe set IDs. All other columns contain the sample values of the probes. It is necessary to have the correct folder structure for the functions of `aroma.affymetrix`. In the current working directory two folders have to be set up. The first folder is a rawData folder. This folder contains another folder with the name of your data set. In our case this will be "HTAData". This "HTAData" folder should contain a last folder with as name the chipType used to produce the data (eg "HTA-2_0") and herein the .CEL files should be placed. The second folder in the working directory is the "annotationData" folder. Herein a folder with name "chipTypes" should be made which contains folders for each chip type with the corresponding names. In the folder of each chip type the .cdf file should be saved. Figure 1 shows the structure of the directories with as working directory a folder named "REIDS_Preprocessing". An R script

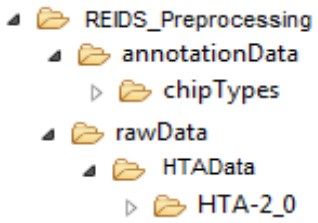

Figure 1: The folder structure for the `aroma.affymetrix` package.

can now be created in the "REIDS_Preprocessing" directory such that the functions have access to the created folders. The function used to perform the data processing is `DataProcessing.R` and is a wrapper of several functions of the `aroma.affymetrix` package. If requested, also gene and exon level summarization is performed with the rma summarization method. Further, the option is available to compute the FIRMA (Purdom et al., 2008) values as well.

```
> library(REIDS)
> DataProcessing(chipType="HTA-2_0",tags="*,r",Name="HTAData",
    ExonSummarization=TRUE,GeneSummarization=TRUE,FIRMA=FALSE,
    location="PreProcessed_HTAData",verbose=TRUE)
```

The tags parameter refers to a tag that was added to the .cdf file. Make sure to have created a folder named "PreProcessed_HTAData" in your working directory for the results to be saved.

In order to avoid the loading of the entire data frame and taking up a substantial amount of memory we create a file in which every line corresponds to one gene and contains all the information of that gene. The creation of the accompanying indexing file will avoid the loading of the entire data frame and allows to process several genes simultaneously by distributing these over multiple cores (eg on a HPC server). This is performed by the function `PivotTransformation.R`. First, a pivot transformation will be performed in order to convert the data into a file with one line per gene. All information concerning one gene is thus gathered into this line. The first column of the returned file contains the gene ID, the second column contains the exon IDs of all the exons of that gene. The third colum indicates the number of probes per exon, the fourth contains the values of those probes per sample and the last column contains the sample names. If a REMAP "TC_Gene_SplitFile.txt" is available, it can be used in order to make a correct distinction between genes. A table with the suffix "GeneTable" is also created which contains the gene ID's. The table is used in a later stage.

```
> data(TC1500264)
> PivotTransformData(Data=TC1500264,GeneID=NULL,ExonID=NULL,REMAPSplitFile=
    "TC1500264_Gene_SplitFile.txt",Location="Output/",Name="TC1500264_Pivot")
```

A final file is created with the begin positions and lengths of the lines of the previous file with the help of `Line_Indexer_Pivot.py`.

```
$ python Line_Indexer_Pivot.py --output_file HTAData_LineIndex.csv
    HTAData_Pivot.csv
```

After output_file the first name is the name to be given to the line indexing file. The second file is the file to be transformed. Make sure that the function is run in the same folder where the file to be transformed is saved. The REIDS function will now work from these begin positions, read a line from the pivot transformed file and process the corresponding gene with the REIDS model.

Finally, we compute the DABG (Affymetrix, 2007) p-values with Affymetrix Power Tool (APT) which is to be installed separately.

```
$ apt-probeset-summarize -a dabg -p HTA-2_0.r3.pgf -c HTA-2_0.r3.clf -b
    HTA-2_0.r3.antigenomic.bgp-o ./APT_DABG Path/To/Folder/*.CEL
```

In order to connect the information of the DABG analysis to our REIDS model a mapping file between the probe set ID, probe set name and transcript ID is necessary. The python script `ID_Name_TC.py` reads information from the "HTA-2_0.na35.hg19.probeset.csv" and "HTA-2_0.r3.psr" files and creates an output file with the required information.

```
$ python ID_Name_TC.py HTA-2_0.na35.hg19.probeset.csv HTA-2_0.r3.psr
    PSID_Name_TCID.txt
```

The mapping between the probe set ID's and probe set names can be linked to the DABG analysis.

```
> ProbesetID_ProbesetName=read.table("PSID_Name_TCID.txt")
> DABG=read.table("APT_DABG/dabg.summary.txt",header=TRUE,sep="\t")

> #DABG in all samples
> Low=apply(DABG,1,function(x) all(x>=0.05))
> LowUIDs=DABG[,1][Low]
> LOW_Probesets_All=unique(ProbesetID_ProbesetName[which(
> ProbesetID_ProbesetName[,1]%in%LOWUIDs),2])
> save(LOW_Probesets_All,file="Low_AllSamples.RData")

> #Per Group
> Low_GSamples=list()
> for(g in 1:length(Groups)){
    Low=apply(DABG,1,function(x) all(x[Groups[[g]]]>=0.05))
    LowUIDs=DABG[,1][Low]
    LOW_Probesets_Samples=unique(ProbesetID_ProbesetName[which(
    ProbesetID_ProbesetName[,1]%in%LOWUIDs),2])
    Low_GSamples[[g]]=LOW_Probesets_Samples
 }
> save(Low_GSamples,file="Low_GSamples.RData")
```

After this final function, the data is ready for the REIDS model.

6 The REIDS Model

The data is now in an adequate format for the REIDS model. The `REIDSFunction.R` function consists of two subfunctions. The first is the `iniREIDS.R` which filters out non-informative probe sets with the I/NI calls model (Kasim et al., 2010). The option is available whether the filtering should be performed or not. The second function is the `REIDSmodel_intern.R` and performs the actual REIDS model. The minimal returned elements are the exon scores and array scores of the probe sets Optional elements are the outcome of the I/NI calls, summarized gene level values and summarized exon level values. Each element is written to a corresponding .txt file. We illustrate the function for a single gene.

```
> load("Low_AllSamples.RData")

> REIDSFunction(ASPSR=c(),Indices="Output/
    TC1500264_LineIndex.csv",DataFile="Output/TC1500264_Pivot.csv",
    nsim=5000,informativeCalls=FALSE,Summarize=c("WeightedAll","EqualAll"),
    rho=0.5,Low_AllSamples=Low_AllSamples,Groups=list(c(1:3),c(4:6)),
    Location="Output",Name="TC1500264")
```

The non-HPC version of the REIDS model is recommended to be used only on a small data set. The `REIDSFunction_HPCVersion` is an adaptation of the REIDS model for use on a HPC server.

In order to identify exons that are alternatively spliced, their exon scores and array scores should be investigated. The array scores can be tested between the groups of interest. If the data consists of paired samples, the mean paired differences can be investigated. We have written a function `ASExons.R` that performs a simple t-test or F-test on the array scores. The p-values are adjusted for multiplicity across all genes. The function has the option to already filter out probe sets that do not pass the exon score threshold. If also a significance level is specified, non-significant p-values will be left out of the results. A data frame with a line per probe set is returned. The columns contain the gene ID, the exon ID, the test statistic, a p-value and an adjusted p-value. If the groups are paired also the mean paired difference is given.

```
> TC1500264_1vs2=ASExons(ExonScores="Output/TC1500264_ExonScores.txt",
> ArrayScores="Output/TC1500264_ArrayScores.txt",Exonthreshold=0.5,
> Groups=list(c(1:3),c(4:6)),paired=FALSE,significancelevel=0.05)
> save(TC1500264_1vs2,file="TC1500264_1vs2.RData")
```

We now add the junction information to discover more on the transcript composition and the possible AS events. The `REIDS_JunctionAssesment.R` returns four files. The first file has name "Name_ASInfo.txt" and contains a line per probe set. It shows the reached decision regarding the probe set (Const/AS/not DABG),its linking and exclusion junctions, the fold change, the AS type and its annotated exons. The second file, "Name_Compositions.txt", is a list of all found transcripts for a particular transcript ID. The third file,"Name_GroupTranscripts.txt" indicates whether a specific transcript is present or absent in a group. The fourth file "Name_NovelConnections.txt" contains junctions which are showing an undocumented connection between probe sets. A last file, "Name_NotAssessedTranscripts.txt", contains transcript of which the isoform information was not available or could not be processed. The "Name_ASInfo.txt" contains information regarding the AS events.

```
> load("TC1500264_1vs2.RData")
> load("Low_GSamples.RData")
> load("Low_AllSamples.RData")

> ASPSR_PSR=unique(TC1500264_1vs2[,2])
> length(unique(ASPSR_PSR))

> REIDS_JunctionAssesment(Indices="Output/TC1500264_LineIndex.csv",
    DataFile="Output/TC1500264_Pivot.csv",ASProbeSets=ASPSR_PSR,
    EandTrAnnotI="Output/REMAP_Indices.txt",EandTrAnnot="Output/
    HJAY_REMAP.txt",Groups=list(c(1:3),c(4:6)),Low_AllSamples=,
    Low_AllSamples,Low_GSamples=Low_GSamples,Location="Output",
    Name="TC1500264")
```

The `REIDS_IsoformAssesment.R` is an experimental function to analyze the isoform information based on the exon level values and the isoform composition. The function returns three files. The first file has name "Name_IsoformIndication.txt" and contains an assessment of the relative expression levels of the isoforms. A second file,"Name_ExonTesting.txt", shows information regarding the differential expression of exon. A final file,"Name_PossibelDEIsoforms" lists isoforms which might be differentially expressed between the groups.

```
> REIDS_IsoformAssesment(geneIDs=GeneTable,IsoformInfo="Output/
    TC1500264_Compositions.txt",ExonLevel="Output/TC1500264_ExonLevel.txt",
    Groups=list(c(1:3),c(4:6)),Location="Output",Name="TC1500264")
```

Finally, it is possible to perform the REIDS method, AS detection and summarization in one function. The function `REIDAnalysis.R` is a wrapper function of the previously introduced functions. The REIDS model will be performed with,if specified, summarization based on all probe sets, followed by an AS identification step and, if specified, a summarization on the constituitive probe sets. If annotation and junction information is available, a junction and isoform assessment will be performed.

```
> data(TC1500264)
> PivotTransformData(Data=TC1500264,GeneID=NULL,ExonID=NULL,
    REMAPSplitFile="TC1500264_Gene_SplitFile.txt",Location=
    "Output",Name="TC1500264_Pivot")

> load("Low_GSamples.RData")
> load("Low_AllSamples.RData")

> REIDS_Analysis(Indices="Output/TC1500264_LineIndex
    .csv",DataFile="OutputTC1500264_Pivot.csv",nsim=100,Summarize=c(
    "WeightedAll","EqualAll","WeightedConst","EqualConst"),rho=0.5,
    Exonthreshold=0.5,significancelevel=0.05,Groups=Groups,paired=FALSE,
    Low_AllSamples=Low_AllSamples,Low_GSamples=Low_GSamples,EandTrAnnotI
    ="Output/REMAP_Indices.txt",EandTrAnnot="Output/HJAY_REMAP.txt",
    Location="Output",Name="TC1500264")
```

7 Visualization

Information of the probe sets can be visualized by the means of plots. `ExpressionLevelPlot.R` plots a specific probe set with its probe set intensities, summarized exon level and gene level values.

```
> load("HTAData.RData")
> load("ExonLevel.RData")
> load("GeneLevel.RData")
> ExpressionLevelPlot(GeneID="TC12000010",ExonID="PSR12000150",
    Data=HTAData,GeneLevelData=GeneLevel,ExonLevelData=ExonLevel,
    groups=list(c(1:3),c(4:6)),ylabel="",title="PSR12000150")
```

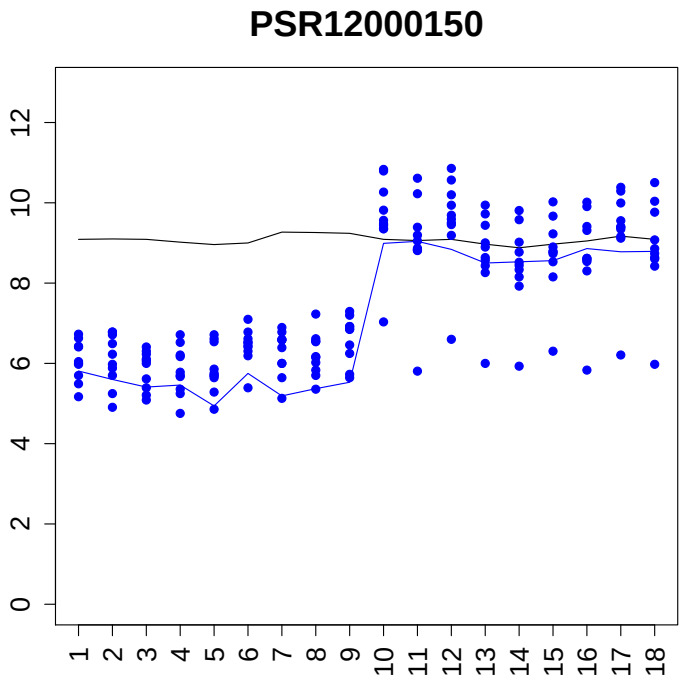

Figure 2: The observed probe intensities of PSR12000150 relative to the summarized gene level values of TC12000010. The black and blue lines indicate the mean profiles of the gene and exon level data respectively. The blue points show the probe level data.

Transcript isoform composition can be retrieved with `TranscriptsPlot.R`. We show the composition of "TC12000010" and highlight PSR12000150.

```
> load("HTAData.RData")
> ExonLevel=read.table("Output_ExonLevel.txt",header=TRUE,sep="\t")

> library(data.table)
> transcripts<-fread("HTA-2_0.na35.2.hg19.transcript.csv",skip=19,
    header=TRUE,sep=",")
> transcripts=as.data.frame(transcripts)
> transcriptData=transcripts[,c(1,4,8)]

> probesets <- fread("HTA-2_0.na35.hg19.probeset.csv",skip=19,
    header=TRUE,sep=",")
> probesets=as.data.frame(probesets)
> positionData <- probesets[probesets[,13] == "main",c(1,2,4,5,6,7)]

> TranscriptsPlot(trans="TC12000010", positions=positionData,
    transcriptinfo=transcriptData,display.probesets=TRUE,
    Data=ExonLevel,groups=list(c(1:3),c(4:6)),Start=NULL,
    Stop=NULL,Highlight="PSR12000150")
```

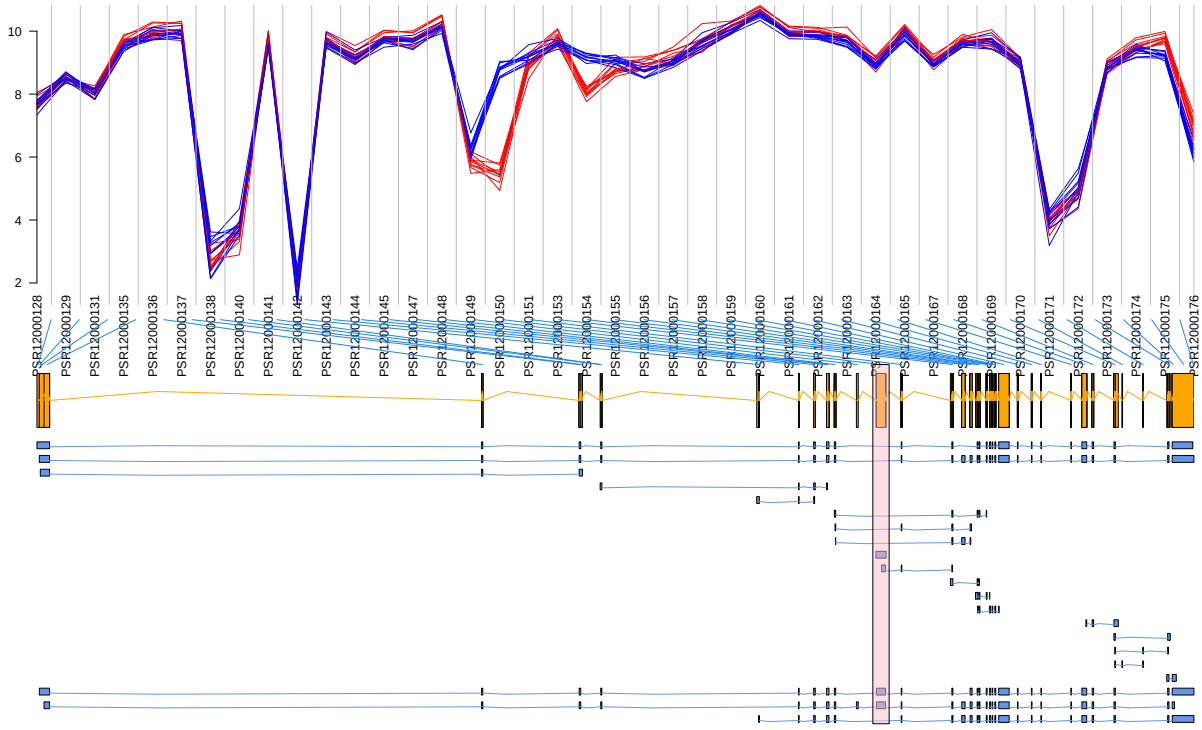

Figure 3: The isoform composition of transcript cluster TC12000010 (WNK1). Probe set PSR12000150 is highlighted as an alternative last event exon. The red lines represent the probe set expression levels of the SCR samples. The blue lines show the probe set expression levels for the siRNA samples.

8 Software used

- R Under development (unstable) (2018-01-28 r74175), `x86_64-w64-mingw32`
- Locale: `LC_COLLATE=C`, `LC_CTYPE=Dutch_Belgium.1252`, `LC_MONETARY=Dutch_Belgium.1252`, `LC_NUMERIC=C`, `LC_TIME=Dutch_Belgium.1252`
- Running under: `Windows 7 x64 (build 7601) Service Pack 1`
- Matrix products: default
- Base packages: base, datasets, grDevices, graphics, methods, stats, utils
- Loaded via a namespace (and not attached): compiler 3.5.0, tools 3.5.0

References

Affymetrix (2007), "Identifying and validating alternative splicing events," *Affymetrix Technical Notes*.

Bengtsson, H., Irizarry, R., Carvalho, B., and Speed, T. P. (2008), "Estimation and assessment of raw copy numbers at the single locus level." *Bioinformatics*, 24, 759–767.

Kasim, A., Lin, D., Van Sanden, S., Clevert, D., Bijnens, L., Goehlmann, H. W. H., Amaratunga, D., Hochreiter, S., Shkedy, Z., and Talloen, W. (2010), "Informative or noninformative calls for gene expression: a latent variable approach." *Stat Appl Genet Mol Biol*, 9, Article4.

Purdom, E., Simpson, K. M., Robinson, M. D., Conboy, J. G., Lapuk, A. V., and Speed, T. P. (2008), "FIRMA: a method for detection of alternative splicing from exon array data," *Bioinformatics*, 24, 1707–1714.

Van Moerbeke, M., Kasim, A., and Shkedy, Z. (2018), "The Usage of Exon-Exon Splice Junctions in the Detection of Alternative Splicing using the REIDS model," *Scientific Reports*, 8, 8331.

Van Moerbeke, M., Kasim, A., Talloen, W., Reumers, J., , Göhlmann, H. W. H., and Shkedy, Z. (2017), "A Random Effectiveects Model for the Identifcation of Differenterential Splicing (REIDS) Using Exon and HTA Arrays," *BMC Bioinformatics*, 18, 273.