PREDA tutorial

October 31, 2025

Abstract

PREDA (Position RElated Data Analysis) tool is a novel R package for integrative analyses of functional genomics data. PREDA implements a procedure to analyze the relationships between data and physical genomic coordinates along chromosomes with the final aim of identifying chromosomal regions with likely relevant functional role. The procedure for position related data analysis is highly flexible and can be applied on data obtained with different technologies. In principle, it can analyze different types of quantitative functional genomics data, e.g., gene expression, copy number, methylation levels. In particular, the underlying algorithm so far has been successfully adopted for the analysis of gene expression and copy number data obtained with various microarray platforms on different organisms. This tool can also integrate analysis results from different types of functional genomics data (e.g. integrated analysis of copy number and gene expression).

Contents

1 PREDA overview

The relationships between gene expression and genomics coordinates can be addressed as a regression problem as firstly proposed by Toedling et al. [1]. A further refinement of this concept was proposed by Callegaro et Al. [2] adopting a method based on non linear kernel regression with adaptive bandwidth, so as to affectively take into account the extreme variability in data density along the genome: LAP (Locally Adaptive Procedure). This method was subsequently further extended in order to address different biological problems, considering different organisms, and different types of data and high throughput technologies [3, 4, 5, 6, 7]. PREDA implements these methodologies in a flexible framework thus allowing its adoption in a broad spectrum of genomics studies. The potential utilizations cover the study of physiological mechanisms affecting genome utilization, such as cellular differentiation, as well as the study of pathological processes involving genome structure modifications, such as chromosomal translocations in cancer. See also the supplementary material about PREDA method for more details about the analysis algorithm.

Core of the underlying algorithm. The core of the underlying algorithm is an improved version of the LAP procedure [2] which consists of three main steps:

1. computation of a statistic on each gene (or other genomic features);
2. non linear regression for smoothing the statistic along genomic coordinates;
3. permutations of gene related statistics followed by smoothing to empirically estimate the local significance of observed smoothed statistics along the genome.

Key features. The method characteristics allow the adoption of the method for position related analysis in a broad variety of applications. In principle, PREDA can be adopted for the analysis of high throughput genomic data obtained with different technologies. Moreover, it can analyze different types of quantitative functional genomics data, e.g., gene expression, copy number, methylation levels. Some of its key features improving the method flexibility are:

- Smoothing method accounting for variable density in genomics data
- No assumptions on the distribution of genes (or other genomic features)
- No assumptions on the distribution of statistics computed on genes (or other genomic features)
- Different scores can be adopted for different applications
- Data from different technologies can be adopted

Modular framework. The basic computational framework can be easily further extended if custom analytical pipelines are required for more complex or specialized purposes. Custom S4-classes have been defined to manage data and genomic information required for the analyses thus facilitating further implementation of custom analytical workflows. See also the vignette about PREDA S4-classes.

Parallel computing. Since the analytical procedure is time consuming, a parallelized version of the algorithm has been also implemented, based on Rmpi, to speed up the analyses. The parallel implementation allows to take advantage both of High Performance Computing systems and of modern multi-core processors that are currently available in common desktop computers.

2 PREDA step by step

The first part of PREDA tutorial is focused on a sample analysis aiming at identifying differentially expressed genomic regions. The PREDA analysis workflow is described taking into account every individual step of the analysis. In subsequent sections some wrapper functions performing the whole analyis with one single command will be describe as well. The adoption of wrapper functions certainly constitute a simpler and more user friendly solution for PREDA analysis. Nevertheless it's worth describing more in details what's going on in the background of each PREDA step.

Sample gene expression dataset. Gene expression microarray data from a previously described dataset [5, 8] concerning clear renal cell carcinoma (RCC) and normal kidney cell samples are used in the following analysis. In particular this dataset is constituted by a subset of samples derived from ArrayExpress dataset E-TABM-282: the same subset used for the analyses described in [5]. The sample gene expression dataset includes 12 samples of clear cell renal carcinoma and 11 samples from normal kidney tissue. The table 1 reports the complete list of selected samples including sample classes, sample names adopted in the following analyses and original raw data files names (.CEL files) that are freely available for download from ArrayExpress repository. The same table is reported in the tab delimited TXT file (`sampleinfoGE_PREDA.txt`) used to collect samples information.

First of all we have to load the required libraries: PREDAsampledata, providing the sample dataset and the PREDA package.

```
> require("PREDAsampledata")
> require("PREDA")
```

Then the variables defining the path to the directory containing the raw CEL files is defined, as well as the path to the sampleinfo file containing in a tab delimited TXT file the same information reported in table 1. The information from the infofile are loaded into R with a `read.table()` command. The use of an "infofile" to hold information abut samples (raw data file, samplename and sample classes) is a very common solution for microarray data analysis, therefore the same convention is adopted here.

Arrayname	Samplename	Class
27CG_03i16741_K K Two Cycle IVT 06mar06.CEL	27CG	RCC
28RA_04i3579_K K Two Cycle IVT 06mar06.CEL	28RA	RCC
33K_04i13776 K Two Cycle IVT 27oct05.CEL	33BV	RCC
36K_04i18916K Two Cycle IVT 27oct05.CEL	36MML	RCC
37K 04i19473 K Two Cycle IVT 12.10.05.CEL	37BA	RCC
40K_04i20257 K Two Cycle IVT 27oct05.CEL	40RR	RCC
45DM_05i5902_K K Two Cycle IVT 25gen06.CEL	45DM	RCC
46SA_05i6348_K K Two Cycle IVT 26gen06.CEL	46SA	RCC
47CA_04i3579_K K Two Cycle IVT 06mar06.CEL	47CA	RCC
49CA_05i6348_K K Two Cycle IVT 25gen06.CEL	49CA	RCC
50PC_05i9837_K K Two Cycle IVT 06mar06.CEL	50PC	RCC
51MI_05i10081_K K Two Cycle IVT 26gen06.CEL	51MI	RCC
28RA_04i3579_C K Two Cycle IVT 06mar06.CEL	Norm1	normal
32GM_04i12879_C K Two Cycle IVT 25gen06.CEL	Norm2	normal
33N_04i13776 K Two Cycle IVT 27oct05.CEL	Norm3	normal
35PA_04i18143_C K Two Cycle IVT 25gen06.CEL	Norm4	normal
36N_04i18916K Two Cycle IVT 27oct05.CEL	Norm5	normal
37N 04i19473 K Two Cycle IVT 12.10.05.CEL	Norm6	normal
40N_04i20257 K Two Cycle IVT 27oct05.CEL	Norm7	normal
41SG_04i20655_C K Two Cycle IVT 26gen06.CEL	Norm8	normal
44DE_05i3989_C K Two Cycle IVT 25gen06.CEL	Norm9	normal
50PC_05i9837_C K Two Cycle IVT 26gen06.CEL	Norm10	normal
51MI_05i10081_C K Two Cycle IVT 26gen06.CEL	Norm11	normal

Table 1: Sample gene expression dataset: samples derived from ArrayExpress dataset E-TABM-282.

```
> # sample info file for the gene expression dataset
> infofile <- system.file("sampledata", "GeneExpression", "sampleinfoGE_PREDA.txt", packag
> sampleinfo<-read.table(infofile, sep="\t", header=TRUE)
> head(sampleinfo)

                                Arrayname Samplename
1 27CG_03i16741_K_Two_Cycle_IVT_06mar06.CEL       27CG
2  28RA_04i3579_K_Two_Cycle_IVT_06mar06.CEL       28RA
3    33K_04i13776_Two_Cycle_IVT_27oct05.CEL       33BV
4     36K_04i18916Two_Cycle_IVT_27oct05.CEL      36MML
5   37K_04i19473_Two_Cycle_IVT_12.10.05.CEL       37BA
6    40K_04i20257_Two_Cycle_IVT_27oct05.CEL       40RR
  Class
1   RCC
2   RCC
3   RCC
4   RCC
5   RCC
6   RCC
```

The current version of PREDAsampledata doesn't contain all of the CEL files, due to package size constraints. The PREDAsampledata package contains

instead the raw CEL files loaded into an AffyBatch object. This AffyBatch object can be loaded with the following command:

```
> data(AffybatchRCC)
```

The origial CEL files can be downloaded from ArrayExpress (accession number E-TABM-282). After decompressing the CEL files archive in a local directory they can be used with the sampleinfo file table provided in the PREDAsampledata package.

```
> CELfilesPath <- "/path/to/local/CEL/files/directory"
```

2.1 Input data

This sample analysis aims at identifying differentially expressed genomic regions in a group of tumor samples (clear cell renal carcinoma - RCC) when compared with a group of normal kidney cell samples. The input data is constituted by the raw gene expression data from Affymetrix GeneChip described above. Raw Affymetrix GeneChips data files (.CEL files) can be preprocessed in R using Bioconductor libraries: see also the Bioconductor documentation for further details, and in particular the documentation for package affy. In this sample analysis the preprocessing steps are taken into account as well beacuse wrapper functions of PREDA package allow performing also raw data preprocessing with a user friendly procedure. Please note that PREDA package can manage gene expression data obtained with every type of microarray or other high throughput technologies, including next generation sequencing. Functions in the PREDA package can import gene expression data (and other types of genomics data) from txt files, for R data.frame objects and from R ExpressionSet objects.

Similarly, for what concerns genomic annotations, i.e. basically the genomic position of each gene, these data can be easily retrieved from Bioconductor libraries or from user provided txt files or data.frame objects. Both cases will be taken into account in the examples of this tutorial.

2.2 Wrapper functions for input data with one step

Since PREDA is a flexible procedure that can actually be adopted for the analysis of different types of genomic data, addressing a variety of biological problems, wrapper functions performing multiple steps of PREDA analysis can be implemented and adoptd to facilitate end user work, especially for non-expert R users.

Differentially expressed genomic regions. In the following example we obtain all of the data required as input for PREDA analysis of differentially expressed genomic regions with one single function. The raw gene expression data are preprocessed, normalized and statistics for differential gene expression are computed to be used as PREDA input statistics.

```
> GEDataForPREDA<-preprocessingGE(SampleInfoFile=infofile,
+ CELfiles_dir=CELfilesPath,
+ custom_cdfname="hgu133plus2",
+ arrayNameColumn=1,
```

```
+ sampleNameColumn=2,
+ classColumn="Class",
+ referenceGroupLabel="normal",
+ statisticType="tstatistic",
+ optionalAnnotations=c("SYMBOL", "ENTREZID"),
+ retain.chrs=1:22
+ )
```

Alternatively we can run the preprocessing steps using the raw data preloaded in the AffyBatch object from PREDAsampledata package.

```
> GEDataForPREDA<-preprocessingGE(
+ AffyBatchInput=AffybatchRCC,
+ custom_cdfname="hgu133plus2",
+ classColumn="Class",
+ referenceGroupLabel="normal",
+ statisticType="tstatistic",
+ optionalAnnotations=c("SYMBOL", "ENTREZID"),
+ retain.chrs=1:22
+ )
```

The `GEDataForPREDA` object contains all of the information and data required for PREDA analysis. In the following sections we can see more in details the individual steps for input data preprocessing and the available options. The preprocessing of genomic data (gene expression data in this case) and genomic annotations are described in distinct sections.

2.2.1 Genomic data

Statistics for PREDA from CEL files. In this sample analysis, we will generate a `statisticsForPREDA` object, that is the S4-class used in PREDA for managing genomic data, directly from raw Affymetrix .CEL files. Since the analysis aim at identifying differentially expressed genomic regions, in the following example the `statisticsForPREDA` object will contain statistics accounting for differential expression of each individual gene.

Raw gene expression data can be preprocessed and normalized using standard procedures from affy package, such as RMA, to obtain an ExpressionSet object: the common data structure used in Bioconductor to manage gene expression data. Here we show just an example generating an ExpressionSet object from raw Affymetrix CEL files. The same object can be obtained from expression data coming other platforms (see also Bioconductor documentation).

The adoption of reliable annotations expression data is a crucial issue for position related analysis, because proper association of expression level to genomic postions is required. In particular, for Affymetrix GeneChips, the adoption of up do date probes annotations, and possibly custom probesets definitions, proved to improve the gene expression analysis results in a number of publications [9, 10, 11, 12]. For this reason we most strongly suggest to adopt custom probesets definition (custom CDF) for data preprocessing with justRMA function: the "cdfname" parameter can be used to specify that a custom CDF [9, 10] must be use instead of standard Affymetrix CDF (i.e. probeset definitions).

As of April 2017 release of Bioconductor, there will be no more any custom CDF package available within Bioconductor repository. For this reason in the working examples provided in this tutorial we are going to use the standard probeset definitions. Custom gene-level centered probeset definitions are still available here http://www.xlab.unimo.it/GA_CDF/ for GeneAnnot based definitions [10] or here http://brainarray.mbni.med.umich.edu/Brainarray/Database/customcdf/genomic_curate for a broader set of options based on different databases [9].

```
> # generate ExpressionSet from raw  CEL files
> ExpressionSetRCC <-justRMA(filenames=sampleinfo[,"Arrayname"], celfile.path=CELfilesPath
```

Again, as alternative to the raw CEL files, for the sample dataset used here we can generate the EpressionSet object starting from the raw data preloaded in the AffyBatch object.

```
> AffybatchRCC@cdfName<-"hgu133plus2"
> annotation(AffybatchRCC)<-"hgu133plus2"
> ExpressionSetRCC <- rma(AffybatchRCC)
```

Then the function `statisticsForPREDAfromEset` can be used to compute statistics for differential expression on ExpressionSet object data to generate a statisticForPREDA object. In this example a t-statistic is computed comparing each group of samples with the specified reference group: in this case the reference group is identified by samples with Class label "normal" ans the only alternative value for Class label is "RCC". Therefore in this example only one comparison is taken into account, i.e. the comparison of RCC samples VS normal samples, because the classVector parameter has just two distinct values. In case multiple classes of samples are available, the selected statistic (in this case the "t-statistic") is repeatedly computed taking into account each group VS reference group comparison.

```
> GEstatisticsForPREDA<-statisticsForPREDAfromEset(ExpressionSetRCC, statisticType="tstati
```

Users can verify the available statistics (just one in our example) using the `analysesNames()` function.

```
> analysesNames(GEstatisticsForPREDA)

[1] "RCC_VS_normal"
```

statistics for PREDA from ExpressionSet. The ExpressionSet S4-class is the generic class used in Bioconductor for managing gene expression data. This data structure can actually be obtained from every gene expression analysis platform. Therefore the procedure above described can actually be adopted to generate statisticsForPREDA objects from every ExpressionSet object, containing expression data from any source.

2.2.2 Genomic annotations

The genomic annotations required for PREDA analysis are managed using `GEnomicAnnotations` and `GEnomicAnnotationsForPREDA` S4-Classes. Easy to use functions are provided to generate these data structures from Bioconductor libraries. Alternative functions for generating as well GenomicAnnotations data structure from R dataframe objects or from txt files are available as well.

GenomicAnnotations from ExpressionSet. The microarray platform used for the expression profiles of the sample gene expression dataset (ArrayExpress dataset E-TABM-282; table 1) is Affymetrix GeneChip HG-U133Plus2.0. The genomic annotations for the probe IDs associated to this platform can be retrieved from the corresponding Bioconductor libraries: see also the Bioconductor documentation for further details, and in particular the documentation for packages affy, annotationDBI and annotate. The information concerning the microarray platform and the associated Bioconductor annotation library are included into a specific slot of the ExpressionSet object as well.

```
> GEGenomicAnnotations<-eset2GenomicAnnotations(ExpressionSetRCC, retain.chrs=1:22)
```

We suggest not to run Position RElated Analysis on sex chromosomes because usually a dataset composition in term of male and female subjects could be unbalanced. That's why the "retain.chrs" parameter is set to retain only autosomal chromosomes, i.e. from 1 to 22 in Human.

GenomicAnnotations from generic annotation library. Alternatively the same information can be obtained directly from a Bioconductor annotation package. The following example creates a `GEnomicAnnotations` object from the Bioconducor library containing the data of EntrezGene database for human.

```
> GEGenomicAnnotations<-GenomicAnnotationsFromLibrary(annotLibrary="org.Hs.eg.db", retain.
```

GenomicAnnotations with optional annotations columns. Finally the above described functions can be used to collect as well additional (optional) annotation columns from the Bioconductor annotation libraries. These optional annotation columns are not required for PREDA analysis but they can be useful for final results annotation. In the following example "SYMBOL" and "ENTREZID" annotation fileds are retrieved from the annotation library for hgu133plus2 GeneChips and included into the output `GEnomicAnnotations` object.

```
> GEGenomicAnnotations<-GenomicAnnotationsFromLibrary(annotLibrary="hgu133plus2.db", retai
```

GenomicAnnotationsForPREDA. The `GenomicAnnotations` S4-class provides an R representation of genomics annotations with a biological meaning: the start and end positions identify he chromosomal localization of each gene locus (or other genomic feature) under investigation. Moreover these annotation are familiar concepts commonly handled by molecular biologists to describe genomic data annotations. However, smoothing analysis, that is the core of PREDA analysis procedure, requires a unique position associated to each data point. For this reason, genomic annotations must be enriched by specifying which exact reference position will be associated to each gene when performing the PREDA smoothing analysis. For this purpose the `GenomicAnnotationsForPREDA` objects are implemented in the PREDA package: they contain an additional annotation field with reference position used for each feature for PREDA analysis. They can be very easily obtained from `GenomicAnnotations` objects using the `GenomicAnnotations2GenomicAnnotationsForPREDA` function. In the following example the "median" position for each gene is used as reference position: i.e. the median position between start and end coordinates of each gene.

```
> GEGenomicAnnotationsForPREDA<-GenomicAnnotations2GenomicAnnotationsForPREDA(GEGenomicAnn
```

Additional available options for "reference_position_type" parameter are: "start", the start coordinate of each gene is used as reference position; "end", the end coordinate of each gene is used as reference position; "strand.start", the start or end coordinate of each gene is used as reference position if the gene is mapped respectively on positive or negative strand; "end.start", the end or start coordinate of each gene is used as reference position if the gene is mapped respectively on positive or negative strand. The user can chose the reference positions according with the data under investigation and analysis purpose. Nevertheless in most cases the reference postion is not expected to dramatically change the final results and usually the "median" position is expected to be a proper choice.

2.2.3 DataForPREDA objects

Before runnning the PREDA analysis, genomic annotations and data are merged into one single object of class `DataForPREDA`. One single function is used for this step performs as well data filtering to remove unmatched data or annotations: a message reporting the number of unmatched ids is printed. Moreover the "sortAndCleanNA" parameter forces the output data and annotation to be sorted according to chromosomal coordinates for each chromosome. Please note that in case "sortAndCleanNA" is set to FALSE (default) the sorting of `DataForPREDA` object is performed as initial step of PREDA analysis (see next section).

```
> GEDataForPREDA<-MergeStatisticAnnotations2DataForPREDA(GEstatisticsForPREDA, GEGenomicAn
```

The output object "GEDataForPREDA", in this example, contains all of the data and annotations required for performing PREDA analysis, i.e. to identify differentially expressed genomic regions. This informatic infrastructure has the clear advantage of taking care of data and annotations consistency, thus facilitating end user work.

2.3 Core of positional analysis

The core of PREDA analysis is composed by non linear smoothing of observed input statistic along chromosomal positions, followed by repeated permutations of input statistic with novel smoothing of permuted data to assess the significance of observed peaks in smoothed statistics. The core of the analysis is performed using the `PREDA_main()` function. The basic input to this function is just a `DataForPREDA` object: this data structure contains all of the data and annotations required for the analysis. Therefore the simplest way to run PREDA analysis is just by using `PREDA_main()` function with default analysis option.

```
> GEanalysisResults<-PREDA_main(GEDataForPREDA)
```

The PREDA_main function is actually a complex function with many options that can be specified by end users. In the following paragraphs some of the main options are described. Nevertheless, for a deeper knowledge of the method we strongly we strongly recommend to carefully read the supplementary material about PREDA method details as well as the `PREDA_main()` function documentation.

Smoothing methods The deafault smoothing metod used in the PREDA_main function is lokern smoothing with scaled bandwidth, using a scaling factor equal to 2. This means that bandwidth estimated by lokern function is divided by 2 so as to reduce the bandwidth. Indeed as discussed in the supplementary method about PREDA, the down-scaling of lokern estimated bandwidths is expected to improve sensitivity and reduce false discovery rate. Scaling factor for lokern bandwidth can be modified with parameter `lokern_scaledBandwidthFactor`. Alterantively, the function used for data smoothing can be modified as well with the parameter `smoothMethod`. Possible values are "lokern", for standard lokern smoothing, "quantsmooth", "spline" and "runningmean.x", where x is a user defined value for the number of adjacent data points using for running mean smoothing.

Permutations Data permutations are used to estimate the significance of extreme values in smoothed statistic. The number of permutations is set with parameter "nperms", with default value equal to 10000. The higher number of permutations is selected, the higher reliability is achieved in estimating statistics significance. Nevertheless an increased number of permutations will result in increased computation time.

Parallel computing The overall algorithm adopted for the integrated analysis of gene expression data and genomic positions is computationally intensive. This is mainly due to the analytical procedure that requires a high number of data re-sampling to empirically estimate on every sample the statistical significance of observed values. This can be considered a typical "embarrassingly parallel" problem, therefore a parallel implementation of the software has been already developed, so as to allow effectively exploiting the computational resources of either an HPC system or common computer desktops with modern multi-core processors. To enable parallel computations during PREDA_main execution, end user has just to set the "parallelComputations" parameters equal to `TRUE`. Please note that in order to run PREDA on a parallel computing environment proper installation and configuration of R packages "Rmpi" and "rsprng" is required: these packages are among suggested packages for PREDA but not among PREDA dependencies, as unfortunately their proper configuration might required expert users or system administrators, depending on the computing system that is used.

2.4 Significant genomic regions

The `PREDADataAndResults` or `PREDAResults` can store all of the output data from PREDA: i.e. the output of `PREDA_main()` function, that performs the core of the analysis. Please see the vignette about PREDA classes as well as the documentation pages about each of these S4-classes for more details. The output objects of `PREDA_main()` results actually contain several statistics computed during the PREDA analysis. Genomic regions with significant variation in the input statistics can be extracted from this objects using the `PREDAResults2GenomicRegions` function. Among the most important function parameter there are the "qval.threshold", that is used to decide what threshold must be used on PREDA q-values (adjusted p-values) for filtering results; the

"smoothStatistic.tail" that is used to decide if we are interested in the upper or lower tail of the statistic values (i.e., in this case, in the up or down regulated genomic regions); the "smoothStatistic.threshold", because in order to reduce the false discovery rate in PREDA results, we suggest to filter the results also on smoothed statistic values (using this threshold) and not only using the qvalues.

```
> genomic_regions_UP<-PREDAResults2GenomicRegions(GEanalysisResults, qval.threshold=0.05,
> genomic_regions_DOWN<-PREDAResults2GenomicRegions(GEanalysisResults, qval.threshold=0.05
```

Then selected significant genomic regions, that are stored in a `GEnomicRegions` object, can be visualized as a dataframe listing chromosomal coordinates of significant regions using the `GenomicRegions2dataframe` function. Since the output of `PREDAResults2GenomicRegions` function is actually a list of `GEnomicRegions` objects, a list subselection is required to visualize data from the first element. A list of objects is generated because the PREDA output can actually store the analysis results concerning multiple input statistics (e.g. multiple comparisons).

```
> dataframe_UPregions<-GenomicRegions2dataframe(genomic_regions_UP[[1]])
> head(dataframe_UPregions)

  chr     start       end
1   1  30717430  30745151
2   1  88760199  92343569
3   1  98711089 103540596
4   1 159936317 160316075
5   1 178545842 199005776
6   1 209430683 218394951
```

2.5 Plot the results

The most interesting visualization of significant genomic regions (in this example differentially expressed regions) can be obtained with the `genomePlot` function. See fig 1 for the output results.

```
> checkplot<-genomePlot(GEanalysisResults, genomicRegions=c(genomic_regions_UP, genomic_re
> legend(x=140000000, y=22, legend=c("UP", "DOWN"), fill=c("red","blue"))
```

The basic input parameters of this function is an object of class `GenomicAnnotationsForLAP` or any other class extending this one. In this example we are actually using the `GEanalysisResults` object, i.e. the output or PREDA analysis, because it contains as well the genomic annotations data. Then a set of user selected significant genomic regions must be provided. This second argument can be actually provided as a `list` of `GEnomicRegions` objects, that is the standard output of `PREDAResults2GenomicRegions` function. Then the user must specify the set of colors to be used for representing boxes for each input `GEnomicRegions` object. Finally, the "grouping" parameter is used to plot multiple sets of genomic regions on a single chromosome: in the example shown above, we have two set of genomic regions as input (UP and DOWN regulated regions) that are plotted together on the chromosomes, buth with two distinct colors (red and blue). If the grouping parameter is not specified the red and blue boxes are plotted on two parallel copies of the chromosomes (see fig 2).

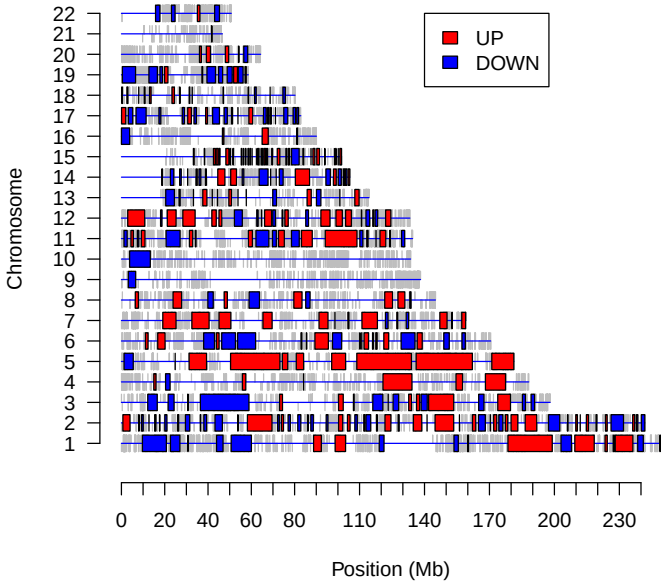

Figure 1: Genome plot: differentially expresssed genomic regions. Blue boxes represent down-regulated regions and red boxes are up-regulated genomic regions.

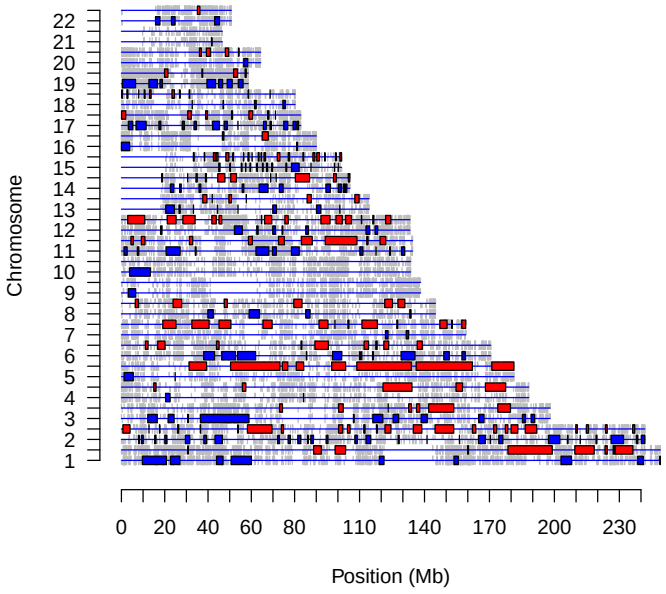

Figure 2: Genome plot: differentially expresssed genomic regions. Blue boxes represent down-regulated regions and red boxes are up-regulated genomic regions.

```
> checkplot<-genomePlot(GEanalysisResults, genomicRegions=c(genomic_regions_UP, genomic_re
```

The `genomePlot` function is actually a very complex function allowing to draw also complex plots. We strongly suggest to carefully read the function documentation.

As of April 2017 release of Bioconductor, there will be no more any custom CDF package available within Bioconductor repository. As can be observed in fig 1 and fig 2 the use of standard probeset definitions, incorporating multiple probeset per gene locus, with variable specificity, yield more noisy position related data analysis results than what achieved with custom gene-centered probeset definitions. For examples of these results the users can refer to PREDA tutorial vignette up to Bioc release 3.4, i.e. the latest release containing custom CDF definition packages. Custom gene-level centered probeset definitions are still available here http://www.xlab.unimo.it/GA_CDF/ for GeneAnnot based definitions [10] or here http://brainarray.mbni.med.umich.edu/Brainarray/Database/customcdf/genomic_curate for a broader set of options based on different databases[9].

3 Predefined workflows

The above described steps of PREDA analysis can actually be customized by end users by defining a set of custom analysis parameters for PREDA functions. Moreover novel functions, such as different smoothing methods, can be incorporated in the analysis workflow, even if multiple options are already provided in the PREDA package (see PREDA functions documentation for more details). In the following sections the use of PREDA pacakge for performing the specific workflow of SODEGIR analysis is described.

3.1 Combined analysis of gene expression and copy number data: SODEGIR

The SODEGIR procedure allows identifying Significant Overlap of Differentially Expressed and Genomic Imbalanced regions described in the paper by Bicciato et al [5]. Figure 3 reports the schema of SODEGIR workflow as described in [5]. Basically, the SODEGIR procedure is composed of two distinct steps of position related data anlysis on gene expression and copy number data: these analysis is performed on each individual sample on gene expression and copy number data. Then the overlap between differentially expressed genomic regions and regions with significant alterations of copy number values is computed to define SODEGIR regions. Finally, the recurrence of specific SODEGIR regions across multiple samples is examined to compute a dataset "signature" of recurrent alterations.

Sample dataset. The sample dataset analyzed with SODEGIR procedure is contained in the PREDAsampledata package. The gene expression dataset is the same described above: the dataset of clear cell renal carcinoma (RCC): ArrayExpress dataset E-TABM-282. The paired copy number data come from ArrayExpress datasets E-TABM-283/E-TABM-284.

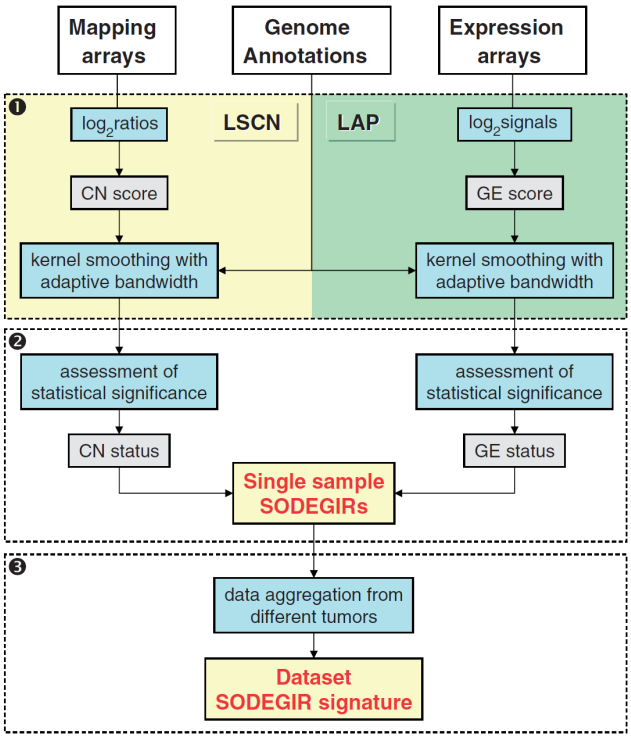

Figure 3: SODEGIR workflow: workflow for the identification of Significant Overlap of Differentially Expressed and Genomic Imbalanced Regions, as described in [5].

3.1.1 Gene expression data analysis

The function `SODEGIRpreprocessingGE` allows performing SODEGIR preprocessing of gene expression data starting from RAW cel files with one single function. This function actually performs the same steps of function `preprocessingGE` but a statistic for each sample is computed: each individual tumor sample is compared with the group of reference normal cells.

```
> # preprocess raw data files
> SODEGIRGEDataForPREDA<-SODEGIRpreprocessingGE(
+ SampleInfoFile=infofile,
+ CELfiles_dir=CELfilesPath,
+ custom_cdfname="hgu133plus2",
+ arrayNameColumn=1,
+ sampleNameColumn=2,
+ classColumn="Class",
+ referenceGroupLabel="normal",
+ statisticType="tstatistic",
+ optionalAnnotations=c("SYMBOL", "ENTREZID"),
+ retain.chrs=1:22
+ )
```

As mentioned in the previous sections, for the sample dataset used in this tutorial, raw gene expression data are alternatively available as an AffyBatch object.

```
> data(AffybatchRCC)
> # preprocess raw data files
> SODEGIRGEDataForPREDA<-SODEGIRpreprocessingGE(
+ AffyBatchInput=AffybatchRCC,
+ custom_cdfname="hgu133plus2",
+ classColumn="Class",
+ referenceGroupLabel="normal",
+ statisticType="tstatistic",
+ optionalAnnotations=c("SYMBOL", "ENTREZID"),
+ retain.chrs=1:22
+ )
```

The resulting DataForPREDA object can be immediately analyzed with PREDA_main function.

```
> # run PREDA analysis on GE data
> SODEGIRGEanalysisResults<-PREDA_main(SODEGIRGEDataForPREDA)
```

Then from PREDA analysis results, we can extract the list of genomic regions with significant UP or DOWN regulation of gene expression levels.

```
> SODEGIR_GE_UP<-PREDAResults2GenomicRegions(SODEGIRGEanalysisResults, qval.threshold=0.05
> SODEGIR_GE_DOWN<-PREDAResults2GenomicRegions(SODEGIRGEanalysisResults, qval.threshold=0.
```

A list of GenomicRegions objects is obtained: with one GenomicRegions object for each sample of the dataset. Therefore the significant regions detected

in each individual sample can be plotted using genomePlot function: see figure 4.

As of April 2017 release of Bioconductor, there will be no more any custom CDF package available within Bioconductor repository. As can be observed in fig 4 the use of standard probeset definitions for gene expression microarrays, incorporating multiple probesets per gene locus, with variable specificity, yield more noisy position related data analysis results than what achieved with custom gene-centered probeset definitions. For examples of these results the users can refer to PREDA tutorial vignette up to Bioc release 3.4, i.e. the latest release containing custom CDF definition packages. Custom gene-level centered probeset definitions are still available here http://www.xlab.unimo.it/GA_CDF/ for GeneAnnot based definitions [10] or here http://brainarray.mbni.med.umich.edu/Brainarray/Database/custor for a broader set of options based on different databases [9].

```
> # plot all the chromosomes for one sample
> checkplot<-genomePlot(SODEGIRGEanalysisResults, genomicRegions=c(SODEGIR_GE_UP[1], SODEG
> title(paste("Sample", names(SODEGIR_GE_UP[1])))
```

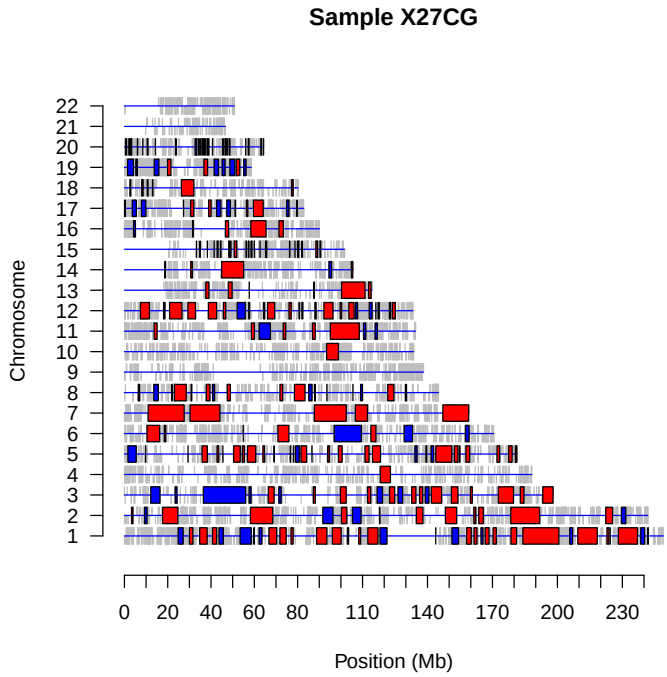

Figure 4: Genome plot for SODEGIR results on gene expression data from one sample.

Alternatively, a plot showing one single chromosome on multiple samples can be drwan. Figure 5 report a plot for chromsomome 5 across all of the RCC samples: this chromosome is frequently ampliifed in this type of cancer

as previuosly discussed [5]. The custom.labels parameter allow modifying the default chromosomes label on vertical axis.

```
> # plot chromosome 5 for all of the samples
> checkplot<-genomePlot(SODEGIRGEanalysisResults, genomicRegions=SODEGIR_GE_UP, scale.posi
```

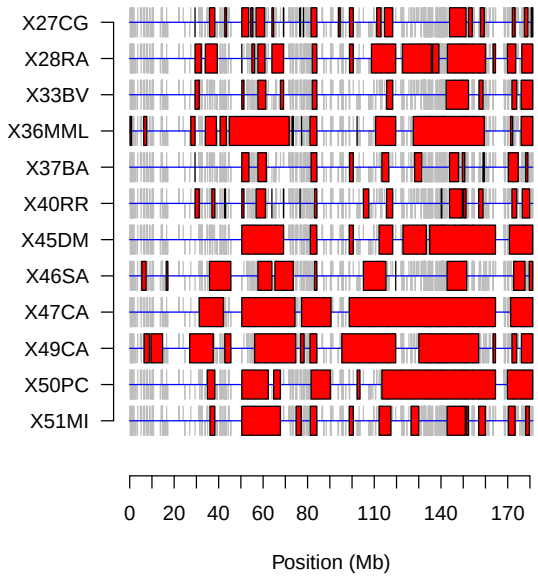

Figure 5: Genome plot for SODEGIR results on gene expression data for chromosome 5 from all samples.

3.1.2 Copy number data analysis

Copy number data were obtained with Human Mapping 100K SNP Affymetrix arrays. In particular copy number microarrays data were obtained from a previously described dataset [5] concerning clear renal cell carcinoma and paired normal diploid cell samples from blood. The initial input copy number data are log-ratio copy number values estimated with CNAG 2.0 software [13] comparing each tumor samples with paired normal reference from blood. Copy number data from paired samples are loaded directly from a text file, as well as corresponding annotation, in order to show the general procedure for importing genomics data into PREDA objects from generic sources. First of all the path to the Copy Number data file and annotations is obtained from PREDAsampledata package.

```
> # path to copy number data files
> CNdataPath <- system.file("sampledata", "CopyNumber", package = "PREDAsampledata")
```

```
> CNdataFile <- file.path(CNdataPath , "CNAG_data_PREDA.txt")
> CNannotationFile <- file.path(CNdataPath , "SNPAnnot100k.csv")
```

Then `StatisticsForPREDAFromfile` function is used to import genomic data from a tab delimited text file into a `StatisticsForPREDA` object.

```
> # read copy number data from file
> CNStatisticsForPREDA<-StatisticsForPREDAFromfile(file=CNdataFile, ids_column="Affymetrix
```

Similarly the `GenomicAnnotationsForPREDAFromfile` function is used to import genomic annotations from a csv file. Please note that different parameters for reading text files can be specified in both functions for reading data from text files, including "header", "sep", "quote", "na.strings": these are common parameers used in R function `read.table()`.

```
> # read genomic annotations
> CNGenomicsAnnotationsForPREDA<-GenomicAnnotationsForPREDAFromfile(
+ file=CNannotationFile,
+ ids_column=1,
+ chr_column="Chromosome",
+ start_column=4,
+ end_column=4,
+ strand_column="Strand",
+ chromosomesLabelsInput=1:22,
+ MinusStrandString="-", PlusStrandString="+", optionalAnnotationsColumns=c("Cytoband", "E
+ header=TRUE, sep=",", quote="\"", na.strings = c("NA", "", "---"))
```

Then genomic data (copy number data) and annotations are merged in to DataForPREDA object.

```
> # merge data and annotations
> SODEGIRCNDataForPREDA<-MergeStatisticAnnotations2DataForPREDA(CNStatisticsForPREDA, CNGe
```

A specific aspect of the SODEGIR procedure, is the integration of copy number analysis output with output from GeneExpression data based on the computation of PREDA statistics on the same set of reference positions used for gene expression data. This integration is achieved in the PREDA package by simply providing the annotations for gene expression data as "outputGenomicAnnotationsForPREDA" parameter. Please note that genomic annotations for gene expression data are actually included as well into the "SODEGIRGEDataForPREDA" object.

```
> # run preda analysis
> SODEGIRCNanalysisResults<-PREDA_main(SODEGIRCNDataForPREDA, outputGenomicAnnotationsForP
```

Then a list of GenomicRegions objects describing chromosomal regions with altered copy number (gain or loss) can be extracted using `PREDAResults2GenomicRegions()` function.

```
> SODEGIR_CN_GAIN<-PREDAResults2GenomicRegions(SODEGIRCNanalysisResults, qval.threshold=0.
> SODEGIR_CN_LOSS<-PREDAResults2GenomicRegions(SODEGIRCNanalysisResults, qval.threshold=0.
```

Figure 6 report the genome plot of regions with copy number gain on chromosome 5 across all of the dataset samples.

```
> # plot chromosome 5 for all of the samples
> checkplot<-genomePlot(SODEGIRGEanalysisResults, genomicRegions=SODEGIR_CN_GAIN, scale.po
```

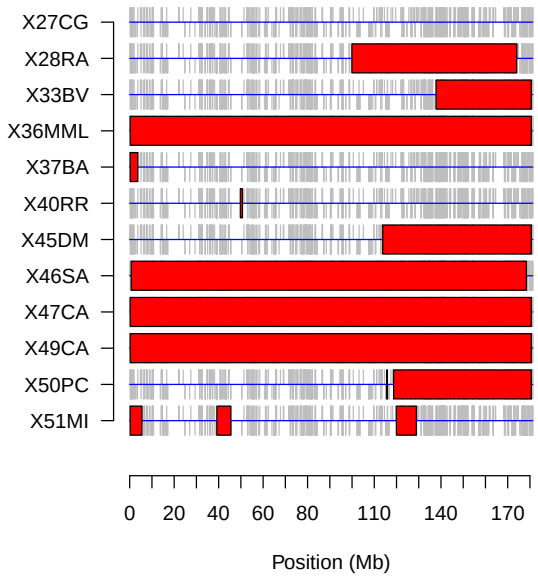

Figure 6: Genome plot for SODEGIR results on copy number data for chromosome 5 from all samples.

3.1.3 SODEGIR procedure

The final integration between gene expression and copy number position related data analysis is achieved by computing the overlap between genomic regions with significant alterations of both types of data. For this purpose it's crucial to analyze the GE and CN data using the same order of samples.

```
> analysesNames(SODEGIRCNanalysisResults)

 [1] "X27CG"  "X28RA"  "X33BV"  "X36MML" "X37BA"  "X40RR"
 [7] "X45DM"  "X46SA"  "X47CA"  "X49CA"  "X50PC"  "X51MI"

> analysesNames(SODEGIRGEanalysisResults)

 [1] "X27CG"  "X28RA"  "X33BV"  "X36MML" "X37BA"  "X40RR"
 [7] "X45DM"  "X46SA"  "X47CA"  "X49CA"  "X50PC"  "X51MI"
```

```
> all(analysesNames(SODEGIRCNanalysisResults) == analysesNames(SODEGIRGEanalysisResults))
```

```
[1] TRUE
```

Then the overlap between up (or down) regulated regions and regions with copy number gain (or loss) is computed for each sample using `GenomicRegionsFindOverlap()` function.

```
> SODEGIR_AMPLIFIED<-GenomicRegionsFindOverlap(SODEGIR_GE_UP, SODEGIR_CN_GAIN)
> SODEGIR_DELETED<-GenomicRegionsFindOverlap(SODEGIR_GE_DOWN, SODEGIR_CN_LOSS)
> names(SODEGIR_AMPLIFIED)<-names(SODEGIR_GE_UP)
> names(SODEGIR_DELETED)<-names(SODEGIR_GE_DOWN)
```

Figure 7 reports a plot of chromosome 5 SODEGIR for all of the samples.

```
> # plot chromosome 5 for all of the samples
> checkplot<-genomePlot(SODEGIRGEanalysisResults, genomicRegions=SODEGIR_AMPLIFIED, scale.
```

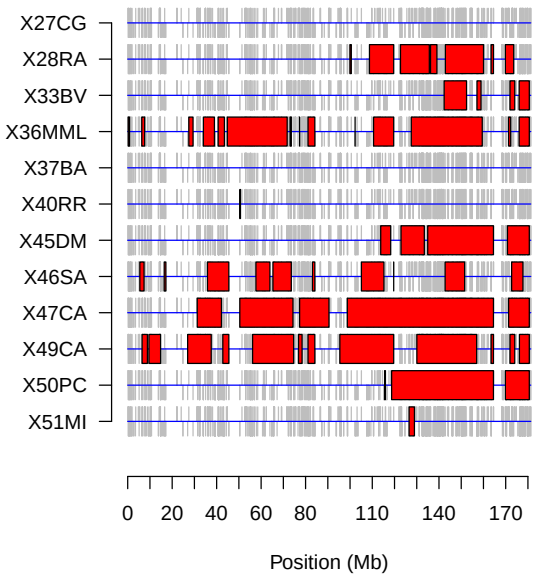

Figure 7: Genome plot for SODEGIR results on chromosome 5 for all samples.

Alterantively we can plot the significant also the regions with significant alterations of gene expression, copy number or SODEGIR from one individual sample: see Figure 8. First of all the selected set of regions is selected from lists containing GenomiRegions objects.

```
> # plot all regions from one sample
>   regions_forPlot<-c(
```

```
+    SODEGIR_GE_UP[[1]],SODEGIR_CN_GAIN[[1]],SODEGIR_AMPLIFIED[[1]],
+    SODEGIR_GE_DOWN[[1]],SODEGIR_CN_LOSS[[1]],SODEGIR_DELETED[[1]]
+    )
```

Then a plot with three copies of each chromosome is drawn (fig.8): each chromosome contains genomic regions of distinct data types as decribed in figure legend.

```
> checkplot<-genomePlot(SODEGIRGEanalysisResults, genomicRegions=regions_forPlot, grouping
> legend(x=140000000, y=22*3, legend=c("GeneExpression UP","CopyNumber gain","SODEGIR ampl
> title(paste("Sample",names(SODEGIR_GE_UP[[1]])))
```

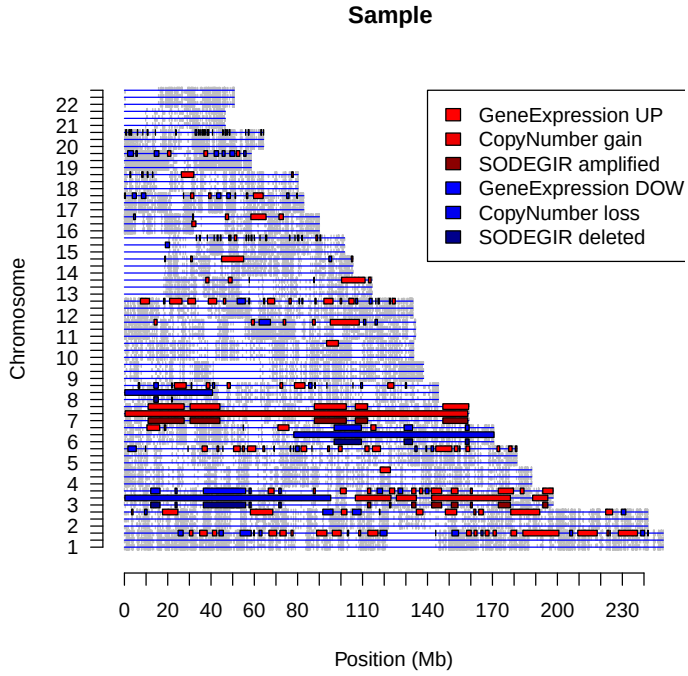

Figure 8: Genome plot for SODEGIR results on one sample, including copy number and gene expression results on distinct lines.

3.1.4 Dataset signature

The final step of SODEGIR analysis is the evaluation of a dataset level signature for recurrent SODEGIRs across multiple samples. This step is performed using function `computeDatasetSignature()` that requires as input a set of reference genomic annotations (derived from "GEDataForPREDA" object) and the list of SODEGIRs from all of hte dataset samples.

```
> SDGsignature_amplified<-computeDatasetSignature(SODEGIRGEDataForPREDA, genomicRegionsLis
> SDGsignature_deleted<-computeDatasetSignature(SODEGIRGEDataForPREDA, genomicRegionsList=
```

The significance of recurrent amplified and deleted regions is computed independently as above shown. The significantly recurrent SODEGIR (i.e. the dataset SDG signature) can be plotted on the genome as well.

```
> # dataset signature
> checkplot<-genomePlot(SODEGIRGEanalysisResults, genomicRegions=c(SDGsignature_amplified,
```

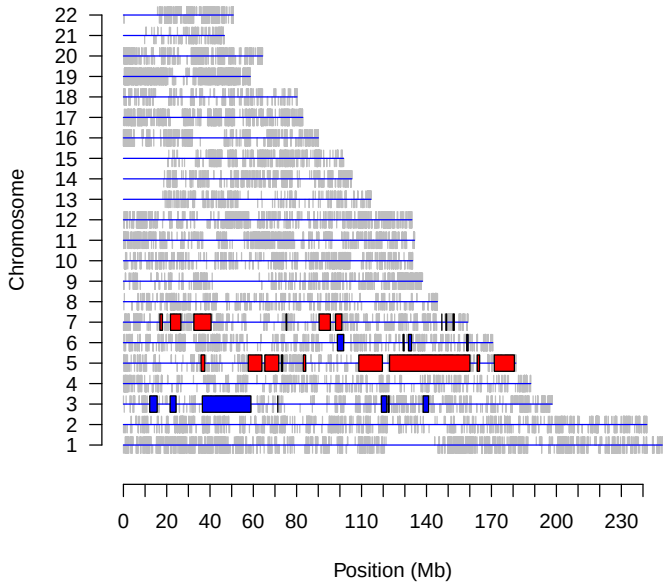

Figure 9: Genome plot for SODEGIR signature over the entire dataset.

Alternatively the significant regions can be visualized as a dataframe using the function `GenomicRegions2dataframe()`.

```
> GenomicRegions2dataframe(SDGsignature_amplified[[1]])
```

```
   chr     start       end
1    5  35868238  37565965
2    5  57660824  63961169
3    5  65315327  71720105
4    5  73023251  73574558
5    5  83065149  84162730
6    5 108728042 119632690
7    5 122927330 160037322
8    5 163441293 164506630
9    5 171310708 180590795
10   7  16870697  18027319
11   7  21722391  26535972
```

```
12   7  32542083  40497871
13   7  75121186  75413556
14   7  90409643  95590054
15   7  97987402 100881871
16   7 147089273 147089273
17   7 148988475 149291296
18   7 152285464 152807563
```

```
> GenomicRegions2dataframe(SDGsignature_deleted[[1]])
```

```
   chr     start       end
1    3  12156209  15581277
2    3  21586160  24305987
3    3  36464246  58895999
4    3  71297805  71297805
5    3 119189365 121646172
6    3 122333565 122705832
7    3 138457257 140960881
8    6  98836746 101734533
9    6 129199852 129643659
10   6 131692322 133097328
11   6 158585935 159220752
```

4 General remarks

The PREDA package implements an informatic infrastructure to perform position related analysis of genomics data. The sample analyses reported in this tutorial just considered few samples possible applications of this procedure. The availability of multiple parameters for data smoothing and for all of the other steps of the procedure allows the adoption of PREDA to address several distinct biological problems. The SODEGIR procedure itself could be actually used to integrate analyses of different combinations of genomics data. PREDA package is therefore built to constitute a generalized approach for position related data analysis for functional genomics applications.

References

[1] Joern Toedling, Sebastian Schmeier, Matthias Heinig, Benjamin Georgi, and Stefan Roepcke. Macat–microarray chromosome analysis tool. *Bioinformatics*, 21(9):2112–3, May 2005.

[2] A Callegaro, D Basso, and S Bicciato. A locally adaptive statistical procedure (lap) to identify differentially expressed chromosomal regions. *Bioinformatics*, 22(21):2658–66, November 2006.

[3] Francesco Ferrari, Stefania Bortoluzzi, Alessandro Coppe, Dario Basso, Silvio Bicciato, Roberta Zini, Claudia Gemelli, Gian Antonio Danieli, and Sergio Ferrari. Genomic expression during human myelopoiesis. *BMC Genomics*, 8:264, 2007.

[4] Clelia Peano, Silvio Bicciato, Giorgio Corti, Francesco Ferrari, Ermanno Rizzi, Raoul Jp Bonnal, Roberta Bordoni, Alberto Albertini, Luigi Rossi Bernardi, Stefano Donadio, and Gianluca De Bellis. Complete gene expression profiling of saccharopolyspora erythraea using genechip dna microarrays. *Microb Cell Fact*, 6:37, 2007.

[5] Silvio Bicciato, Roberta Spinelli, Mattia Zampieri, Eleonora Mangano, Francesco Ferrari, Luca Beltrame, Ingrid Cifola, Clelia Peano, Aldo Solari, and Cristina Battaglia. A computational procedure to identify significant overlap of differentially expressed and genomic imbalanced regions in cancer datasets. *Nucleic Acids Res*, 37(15):5057–70, August 2009.

[6] Alessandro Coppe, Francesco Ferrari, Andrea Bisognin, Gian Antonio Danieli, Sergio Ferrari, Silvio Bicciato, and Stefania Bortoluzzi. Motif discovery in promoters of genes co-localized and co-expressed during myeloid cells differentiation. *Nucleic Acids Res*, 37(2):533–49, February 2009.

[7] Haisheng Nie, Pieter Bt Neerincx, Jan van der Poel, Francesco Ferrari, Silvio Bicciato, Jack Am Leunissen, and Martien Am Groenen. Microarray data mining using bioconductor packages. *BMC Proc*, 3 Suppl 4:S9, 2009.

[8] Ingrid Cifola, Roberta Spinelli, Luca Beltrame, Clelia Peano, Ester Fasoli, Stefano Ferrero, Silvano Bosari, Stefano Signorini, Francesco Rocco, Roberto Perego, Vanessa Proserpio, Francesca Raimondo, Paolo Mocarelli, and Cristina Battaglia. Genome-wide screening of copy number alterations and loh events in renal cell carcinomas and integration with gene expression profile. *Mol Cancer*, 7:6, 2008.

[9] Manhong Dai, Pinglang Wang, Andrew D Boyd, Georgi Kostov, Brian Athey, Edward G Jones, William E Bunney, Richard M Myers, Terry P Speed, Huda Akil, Stanley J Watson, and Fan Meng. Evolving gene/transcript definitions significantly alter the interpretation of genechip data. *Nucleic Acids Res*, 33(20):e175, 2005.

[10] Francesco Ferrari, Stefania Bortoluzzi, Alessandro Coppe, Alexandra Sirota, Marilyn Safran, Michael Shmoish, Sergio Ferrari, Doron Lancet, Gian Antonio Danieli, and Silvio Bicciato. Novel definition files for human genechips based on geneannot. *BMC Bioinformatics*, 8:446, 2007.

[11] Jun Lu, Joseph C Lee, Marc L Salit, and Margaret C Cam. Transcript-based redefinition of grouped oligonucleotide probe sets using aceview: high-resolution annotation for microarrays. *BMC Bioinformatics*, 8:108, 2007.

[12] Scott L Carter, Aron C Eklund, Brigham H Mecham, Isaac S Kohane, and Zoltan Szallasi. Redefinition of affymetrix probe sets by sequence overlap with cdna microarray probes reduces cross-platform inconsistencies in cancer-associated gene expression measurements. *BMC Bioinformatics*, 6:107, 2005.

[13] Yasuhito Nannya, Masashi Sanada, Kumi Nakazaki, Noriko Hosoya, Lili Wang, Akira Hangaishi, Mineo Kurokawa, Shigeru Chiba, Dione K Bailey, Giulia C Kennedy, and Seishi Ogawa. A robust algorithm for copy number detection using high-density oligonucleotide single nucleotide polymorphism genotyping arrays. *Cancer Res*, 65(14):6071–9, July 2005.