DECO

DEcomposing heterogeneous Cohorts from Omic profiling

F. J. Campos-Laborie, J. M. Sanchez-Santos and J. De Las Rivas
Bioinformatics and Functional Genomics Group
Cancer Research Centre (CiC-IBMCC, USAL/CSIC/IBSAL)
Salamanca (Spain)

Abstract

Here we present a tutorial to use **DECO**, a method to explore and find differences in heterogeneous large datasets usually produced in biological or biomedical omic-wide studies. The method makes a comprehensive analysis of multidimensional datasets (usually consisting on a collection of samples where hundreds or thousands of features have been measured with a large-scale high-throughput technology, for example, a genomic or proteomic technique). The method finds the differences in the profiles of the features along the samples and identifies the associations between them, showing the features that best mark a given class or category as well as possible sample outliers that do not follow the same pattern of the majority of the corresponding cohort. The method can be used in a supervised or unsupervised mode, it allows the discovery of multiple classes or categories and is quite adequate for patients stratification.

Contents

1. Variability and heterogeneity in high-dimensional data

Individual diversity and variability is one of the most complex issues to deal within high-dimensional studies of large populations, as the ones currently performed in biomedical analyses using omic technologies. DECO is a method that combines two main computational procedures: **(i)** a **Recursive Differential Analysis (RDA)** that performs combinatorial sampling without replacement to select multiple sample subsets followed by differential analysis; and **(ii)** a **Non-Symmetrical Correspondence Analysis (NSCA)** of differential events that allow the characterization and assigment of features and samples in a common multidimensional space, combining in a single statistic parameterization both the feature-sample changes detected and a predictor-response information.

The statistical procedure followed in both parts of the method are detailed in the original publication [1], but this brief **vignette** explains how to use **DECO** to analyze multidimensional datasets that may include heterogeneous samples. The aim is to improve characterization and stratification of complex sample series, mostly focusing on large patient cohorts, where the existence of outlier or mislabeled samples is quite possible.

In this way, **DECO** can reveal exclusive associations between features and samples based in specific differential signal and provide a better way for the stratifycation of populations using multidimensional large-scale data. The method is applied to data derived from different **omic technologies**, for example: genome-wide expression data obtained with microarrays or with RNA-seq (either for genes, miRNAs, ncRNAs, etc).

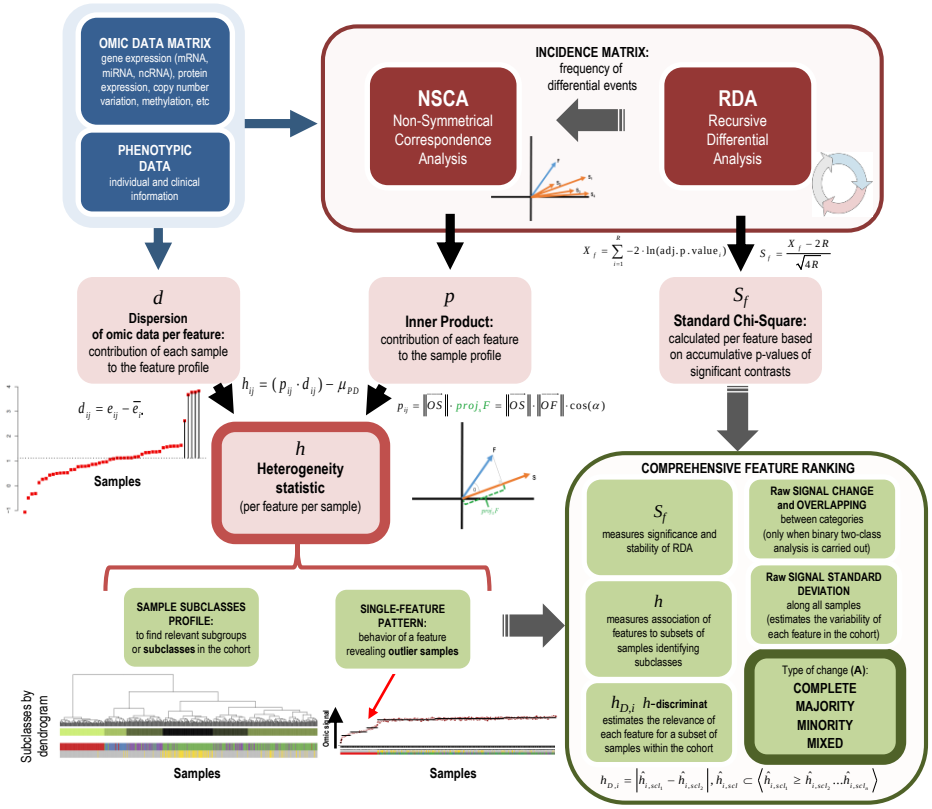


Figure 1: Workflow of DECO algorithm.

2. Installation

The **deco** R source package can be downloaded from R CRAN repository or from our lab website (http://bioinfow.dep.usal.es/deco/). It can be installed by downloading and executing in UNIX-like terminal:

```
R CMD INSTALL /path/to/deco_1.0.tar.gz
```

or first choosing your nearest CRAN mirror using `chooseCRANmirror()` and executing in R:

```
install.packages("deco", type="source", dependencies = TRUE)
```

If any problem with dependencies are reported, user can download R packages directly from Bioconductor repository and repeat previous instruction to install **deco** R package:

```
# Bioconductor dependencies
source("https://bioconductor.org/biocLite.R")
biocLite(c("AnnotationDbi", "foreign", "plsdof", "limma", "Biobase",
         "gdata", "cluster", "gplots", "RColorBrewer", "locfit",
         "snowfall", "scatterplot3d", "made4", "fmsb",
         "lisp", "ade4", "sfsmisc"))

# Loading package in R
library(deco)
```

It can be also done using `devtools` R package to install dependencies from local directory (in this case the user has to decompress .tar.gz previously):

```
# Loading devtools R package...
library(devtools)

# Installing dependencies. Path to directory containing decompressed R package
install_deps("/path/to/deco/",dependencies="logical")

# Loading package in R
library(deco)
```

This R package contains a experimental dataset as example and all functions needed to run an analysis.

3. Experimental data

At presernt, the method directly supports two types of data from transcriptomic technologies: microarrays and RNA-seq platforms. In case of microarrays, robust normalization of raw signal is needed for correct application of *eBayes* method (done for example with the **RMA** method or with **normalizeBetweenArrays** method from LIMMA package [2]). Notwithstanding, RNA-seq read counts matrix (genes or transcripts as rows and samples as columns) can be the input; and in this case then user should apply voom normalization method [2]. Below, we show two different examples of both types of datasets obtained with such platforms.

3.1 Microarrays dataset: study on lymphoma subtypes

Here, a normalized microarray gene expression matrix from clinical samples is used as example taken from Scarfo et al.[3]. **Anaplastic Large Cell Lymphoma** (**ALCL**) is an heterogeneous disease with two well differentiated forms based on ALK gene expression: *ALK(-)* and *ALK(+)*. The dataset, obtained in GSE65823 from GEO database, corresponds to genome-wide expression profiles of human T-cell samples hybridizated on *Affymetrix* HGU133Plus2.0 platform, which were mapped to ENSEMBL genes with *genemapperhgu133plus2cdf*

CDF package from GATExplorer [4]. The mapping from *Affymetrix* probes to genes can be also done using BrainArray CDF packages.

The main interest to include this dataset, to be analysed with DECO, is because using this sample set Scarfo et al. [3] identified of **a subset of patients within the ALK(-) class** discovering high ectopic expression of several gene markers. To do so, the authors applied one of the most used methods for detection of outliers and heterogeneous behavior, that is COPA [5]. Further comparisons between **COPA** and other related methods can be found in our paper about DECO [1]. The phenotypic information about this sample set provided by GEO database were included in an *ExpressionSet* object. This R object called `ALCLdata` could be directly loaded as follows:

```
data(ALCLdata)
# to see the ExpressionSet object
ALCLdata
```

```
# to see the phenotypic information
pData(ALCLdata)
```

Classes vector to run a *supervised* analysis (explained in following section) to compare both ALCL classes: *positive*ALK and *negative*ALK.

```
classes.ALCL <- pData(ALCLdata)[,"Alk.positivity"]
names(classes.ALCL) <- sampleNames(ALCLdata)
```

3.2 RNA-seq dataset: study on breast cancer subtypes

Here, we show a RNAseq dataset analysed using DECO. The dataset was downloaded from The Cancer Genome Atlas (TCGA). It is composed by 878 clinical samples from patients with different subtypes of Breast Cancer [6], that include the standard classes (given by markers ESR1, PGR and HER2) and two classes associated to the cell-type, called: **Invasive Ductal Carcinoma** (IDC) and **Invasive Lobular Carcinoma** (ILC). The genes of the dataset are mapped to HGNC symbol IDs. The dataset can be loaded directly in R or downlad from the TCGA data portal:

```
# Load required R package to download data.
library(TCGA2STAT)
# Download complete Breast Cancer RNAseq counts matrix from TCGA database.
BRCA_count <- getTCGA(disease = "BRCA", data.type = "RNASeq",
                      type = "count", clinical=TRUE)

# Different slots included in the data (count matrix, clinical data and a merged matrix).
names(BRCA_count)
[1] "dat"        "clinical"   "merged.dat"

# Apply 'voom' normalization from LIMMA R package to read count matrix
# It would return a log2_counts_per_million (logCPM) inside 'E' slot.
BRCA_count$dat <- limma::voom(BRCA_count$dat, normalize = "quantile")$E
```

Then, user can run *voom* normalization method provided in LIMMA R package to calculates matrix of **logCPMs**. Further information about **voom** normalization and its properties can be found in LIMMA R package [2]. The normalized matrix is then analysed using the RDA method of DECO.

DECO is also able to analyse other RNAseq data types (RPKMs, FPKMs or TPMs values). The data are usually log scaled. Here, we shown an example using data type RPKMs from TCGA database.

```
# Load required R package to download data.
library(TCGA2STAT)
```

```
# Download complete Breast Cancer RNAseq RPKMs matrix from TCGA database.
BRCA_rpkm <- getTCGA(disease = "BRCA", data.type = "RNASeq2", clinical=TRUE)

# Different slots included in the data (count matrix, clinical data and a merged matrix).
names(BRCA_rpkm)
[1] "dat"        "clinical"   "merged.dat"

# Conversion to log2 scale, required to apply LIMMA.
BRCA_rpkm$dat <- log2(BRCA_rpkm$dat + 1)
```

3.3 Use of other *omic* platforms

Together with RNA-seq or microarray platforms, DECO algorithm can be applied to datasets obtained with other **omic platforms** (as far as a correct normalization of the data per sample can be achieved).

In order to provide an example of other platform, we show an example of a **miRNAs dataset** from same TCGA database used above:

```
library(TCGA2STAT)
# Download complete Breast Cancer miRNAseq rpmmms matrix from TCGA database.
BRCA_mirna <- getTCGA(disease = "BRCA", data.type = "miRNASeq",
                      type = "rpmmm", clinical=TRUE)

# Different slots included in the data.
names(BRCA_mirna)
[1] "dat"      "clinical"   "merged.dat"

# Convert 'rpmmm' to log2 scale.
BRCA_mirna$dat <- log2(BRCA_mirna$dat + 1)
```

Additionally, more information about different data platforms available for direct download to R environment can be queried on `TCGA2STAT` R package vignette.

4. RDA, Recursive Differential Analysis: *Standard.Chi.Square*

We proposed a recursive subsampling strategy which selects subsets of samples (from the different classes) and compares all against all (in an exhaustive search). When the number of combinations is very large a random selection of all possible subsets is done. In order to obtain the best possible results, three parameters should be taken into consideration before running the analysis: **(i)** the subsampling size called `r` (DECO method calculates an optimal size of subsampling subsets if the user does not define it); **(ii)** number of subsets or combinations, called `iterations`, to compare in this subsampling step; and **(iii)** adjusted.p.value threshold for the differential tests or contrasts, called `q.val` and computed using eBayes from LIMMA [2].

Aiming to summarize all positive differential events (DE) for each feature (combinations with a lower *adj.p.value* than threshold), Fisher's combined probability test is applied to each final feature vector of *adj.p.values* to obtain a **Standard.Chi.Square**, which will is not affected by type of analysis (*supervised* or *unsupervised*) because it only takes into account number of positive DE events.

4.1 Supervised analysis

Depending on *classes* input vector, a *supervised* analysis compares just two types of samples (i.e. healthy donors *versus* patients in a typical biomedical study). The `decoRDA()` RDA function will adjust the optimal subsampling size *r* (that the user can modify) to explore all DE signal, and UP and DOWN events will be taken into account for posterior **NSCA**. Here, an example of *decoRDA* function using `ALCLdata` dataset:

```
# example of SUPERVISED design with Affymetrix microarrays data
sub <- decoRDA(data = exprs(ALCLdata), classes = classes.ALCL, q.val = 0.01,
                    rm.xy = T, r = NULL, cpus = 6, parallel = T, temp.path = getwd(),
                    control = "pos", annot = T, id.type = "ENSEMBL",
                    pack.db = "org.Hs.eg.db")
```

This **RDA** procedure generates an `incidenceMatrix` which counts differential events per gene (feature) per sample. Thus, this matrix would contain just differential genes as rows and samples as columns with one differential event at least.

```
dim(sub$incidenceMatrix)
```

The `incidenceMatrix` produced after the RDA, can reveal the important changes that mark an entire subclass (grey boxes in **Figure 1**), as well as specific signal changes that mark a subclass of samples (red boxes in **Figure 1**). As we can see in a simple example (**Figure 1**), both *Gene 1* and *Gene 2* seem to mark two subclasses (or subtypes) inside each compared class, while *Gene 3* and *Gene 4* reflect the behaviour of control and case classes respectively. Following the RDA step, the NSCA step analyses the numbers of the `incidenceMatrix`. The NSCA analysis is also done splitting UP and DOWN changes when the algorithm is run in *supervised* mode.

4.2 Unsupervised analysis

If *classes* input vector is empty, a *unsupervised* analysis is run comparing all against all samples taking different subsets (each combination of samples is unique) and looking for UP events. Then, those samples which show any differential change with statistical significance will be counted. In order to clarify final results of NSCA analysis, it is important to underline that just UP regulated events will be assigned to samples, while both UP and DOWN regulation events are counted in the *supervised* analyisis explain above.

```
# example of UNSUPERVISED design with RNA-seq data (log2[RPKM])
sub <- decoRDA(data = BRCA_rpkm$dat, q.val = 0.01, r = NULL, cpus = 6,
                   parallel = T, temp.path = getwd(), annot = T, rm.xy = T,
                   id.type = "SYMBOL", pack.db = "org.Hs.eg.db")
```

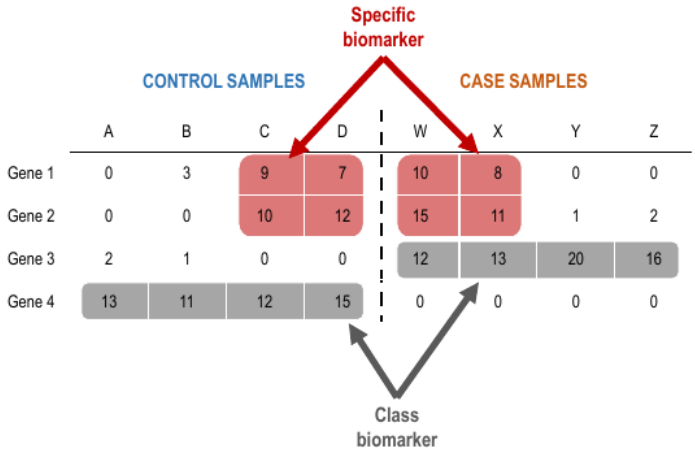


Figure 2: Example of an "incidence matrix" obtained after RDA for a SUPERVISED analysis.

RDA procedure generates in this case an `incidenceMatrix` which counts just UP events per gene per sample.

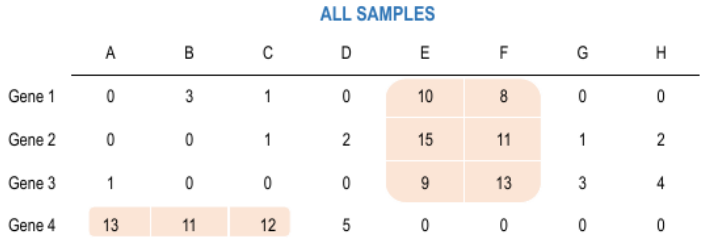


Figure 3: Example of an "incidence matrix" obtained after RDA for a UNSUPERVISED analysis

4.3 Multiclass analysis

Together with *supervised* or *unsupervised* analyses, the method can be run for *multiclass comparison*, taking subsets of samples from several classes identified *a priori* and forcing them to be compared. Then, we would count differential events per feature per sample but there will not be mix between different classes. Here, we show an example of a breast cancer dataset (from Ciriello et al. [6], log2(RPKM+1) scaled) that uses the well-defined PAM50 classes:

```
# Loading phenoData for TCGA paper (Cell 2012)
data(phenoData.BRCA.TCGA)
# Selecting samples with complete phenoData.
BRCA_rpkm$dat <- BRCA_rpkm$dat[, colnames(BRCA_rpkm$dat) %in%
                               phenoData.BRCA.TCGA[,"mRNA"]]
# Selecting IDC and ILC samples.
BRCA_rpkm$dat <- BRCA_rpkm$dat[, rownames(phenoData.BRCA.TCGA[
  !phenoData.BRCA.TCGA[,"Final.Pathology"] %in% c("Other","Mixed.IDC.ILC"),])]
# Creating a multiclass vector with PAM50 subclasses.
classes.PAM <- phenoData.BRCA.TCGA[colnames(BRCA_rpkm$dat), "PAM50"]
names(classes.PAM) <- rownames(phenoData.BRCA.TCGA[colnames(BRCA_rpkm$dat),])
# example of MULTICLASS design with RNA-seq data (log2[RPKM])
sub <- decoRDA(data = BRCA_rpkm$dat, classes = classes.PAM, q.val = 0.01,
```

```
                r = NULL, cpus = 6, parallel = T,
                temp.path = getwd(), annot = T, rm.xy = T,
                id.type = "SYMBOL", pack.db = "org.Hs.eg.db")
```

4.4 Running the RDA function: `decoRDA()`

This vignette presented some examples of `decoRDA()` subsampling function for *supervised* and *unsupervised* analyses (if user has two classes of samples or not). Now, details about all the input parameters which control the RDA procedure are indicated:

- `data` input corresponds to our expression matrix with features as rows and samples as columns.
- `q.val` is the threshold imposed to the *adjusted.p.value* from *LIMMA* method in each iteration.
- `r` is the resampling size.
- `temp.path` defines a location in your computer where `decoRDA()` would save temporary results.
- `classes` is a character vector or factor indicating to which class each sample belongs.
- `control` is a character indicating which label has to be set as control class in a *supervised* analysis.
- `rm.xy` is a logical indicating if X or Y chromosome placed genes/proteins/features should be removed before run RDA (requires `id.type` and `annot` inputs).

All the rest of parameters are used to annotate features or to establish a parallel computation of processes, so they are explained within a longer and more detailed vignette included in the DECO R package.

5. NSCA, Non-Symmetrical Correspondence Analysis: *h* statistic

Once the frequency matrix of DE events or *incidenceMatrix* has been produced, DECO follows applying a NSCA [7] procedure. **NSCA** allows analyse all dependencies and covariances between differential features and samples placing them in the same relational space. Further information can be found in a more detailed vignette included in the DECO R package and also in our original publication [1].

As a measure of this significant association, NSCA function returns a *inner product* matrix relating feature-sample dependencies in the differential context. After the *inner product* matrix is generated, samples with similar profiles (using all the genes that gave DE events over a threshold: `pos.rep`) are grouped together using a hierarchical clustering based on Pearson correlation distances between samples: $dist_{ij} = 1 - corr(p_i, p_j)$.

Additionally, all different *agglomeration methods* to creates a dendrogram (see further information in `hclust` R function) are assessed looking for the method that shows highest *cophenetic correlation* [8]. Thus, we identify the best clustering procedure to make subclasses, choosing an optimal number of subclasses depending on the best *Hubber's Pearson* γ cutting this dendrogram.

5.1 Running the NSCA function: `decoNSCA()`

Here, we show an example of how user can run the second step of **DECO**:

```
# It can be applied to any subsampling design (SUPERVISED or UNSUPERVISED) and
# any transcriptomic platform (microarray, RNAseq, methylation, etc)
deco_results_ma <- decoNSCA(sub = sub, v = 80, method = "ward.D",
                            k.control = 3, k.case = 3, samp.perc = 0.05, rep.thr = 5,
                            parallel = T, cpus = 6, rm.complete = F, overlap = T)
```

Several important **input parameters** of this `decoNSCA()` function can be set up by user. Further information could be found in `?decoNSCA` help page. Finally, this function will return an output R object `deco` that is described in the following section.

6. Description of output results

```
slotNames(deco_results_ma)
[1] "featureTable"    "NSCAcluster"     "incidenceMatrix" "classes"
"pos.iter"        "control"         "q.val"         "subsampling.call"
"deco.call"
```

After running NSCA function *decoNSCA*, the method produces an R object of *deco* class. The main slots with relevant information inside this object are:

- `featureTable` is the main output table with the **feature statistics** and rankings.
- `NSCAcluster` contains the NSCA information and sample subclasses. It will be duplicated if a *supervised* analysis is run.
- `incidenceMatrix` is the **Absolute frequency matrix** with DE events per sample used in the NSCA.
- Vector of *classes* with labels per sample. For *unsupervised* analysis it will be **NA**.
- Label set as *control*.
- `q.val` is the adjusted.p.val threshold previously defined.
- `subsampling.call` and `deco.call` correspond to both `decoRDA` and `decoNSCA` function calls.

Feature ranking and statistics

The main output table with relevant feature information from RDA, NSCA and subclasses searching corresponds to *featureTable*.

```
dim(deco_results_ma@featureTable)
# Statistics of top-10 features
deco_results_ma@featureTable[1:10,]
```

The most relevant statistic derived from RDA technique is the *Standard.Chi.Square*. The amount of *differential events* or *Repeats* that each gene (each feature) appears differentially changed among classes or samples is also very important, and it is summarized in *Standard.Chi.Square* since this parameter weights the significance of the DE. Genes with similar *Repeats* values which correspond to lower *adj.p.value* resemble higher *Standard.Chi.Square* values, meanwhile genes with higher *adj.p.value*, or near *q.val* threshold imposed by user, give lower *Standard.Chi.Square* values.

IDs	Standard.Chi.Square	Repeats	adj.p.values	h.Range	Dendrogram.group
DE feature 1	250	100	~ 0.01	3.26	2
DE feature 2	150	100	~ 0.05	12.65	5

Moreover, for *supervised* analysis the *exprsUpDw* character indicates if case class shows UP or DOWN regulation of each feature. In some cases, several genes could follow deregulation in both classes for some subgroup of samples, which we called change-type **MIXED**. This kind of change pattern could explain some hidden characteristic of the samples and allows finding outliers: a subgroup of samples that only change in a subset of genes. In this cases there are not differences between the mean or median for the whole classes, and so classical methods like **SAM** or **LIMMA** do not find these patterns.

After RDA and NSCA analysis, the statistics referred to sample subclasses found is used to rank DE features properly. In this way, h statistic obtained per feature is used to determine how each feature discriminates each subclass found. As we mentioned above, this statistic combines both the DE changes and the predictor-response relationship between features and samples, so it refers to feature's discriminant ability. Furthermore, `Dendrogram.group` helps to identify to which pattern belongs each feature and each sample within the h statistic heatmap (`decoReport()` PDF report).

Sample subclasses membership

To see how samples are grouped into different *subclasses* within class:

```
# If SUPERVISED analysis
sampleSubclass <- rbind(deco_results_ma@NSCAcluster$Control$samplesSubclass,
                        deco_results_ma@NSCAcluster$Case$samplesSubclass)
# If UNSUPERVISED analysis
sampleSubclass <- deco_results_ma@NSCAcluster$All$samplesSubclass
## Sample subclass membership
head(sampleSubclass)
                  Subclass
GSM1607016 "pos Subclass 1"
GSM1607017 "pos Subclass 1"
GSM1607018 "pos Subclass 1"
GSM1607019 "pos Subclass 1"
GSM1607021 "pos Subclass 1"
GSM1607024 "pos Subclass 2"
```

Additionally, we can print a brief summary of DECO analysis using `summary` or `print` native R functions.

```
## Example of summary of a 'deco' R object (ALCL supervised/binary example)

summary(deco_results_ma)
# Decomposing Heterogeneous Cohorts from Omic profiling: DECO
# Summary:
# Analysis design: Supervised
# Classes compared:
# neg pos
#  20  11
#            RDA.q.value Minimum.repeats Percentage.of.affected.samples NSCA.variability
# Thresholds        0.01          10.00                           5.00               86
# Number of features out of thresholds: 297
# Feature profile table:
# Complete Majority Minority    Mixed
#       12       87      197        1
# Number of samples affected: 31
# Number of positive RDA comparisons: 1999
# Number of total RDA comparisons: 10000
```

An **extended report** (as PDF file) including more detailed information of the analysis and several plots illustrating all the results (as the bi-clustering approach to `h` statistic matrix) can be also produced with the `decoReport()` R function. Information about the extended report is included in longer and more detailed vignette in the DECO R package.

7. Output reports

7.1 Generating a PDF report

DECO R package implements an additional function to help users to view and analyse the output results. It contains a detailed representation of main results (subclasses found, main biomarkers, *h* statistic heatmap, best feature profiles, feature's overlapping signal...). Here, we briefly describe how to run `decoReport()` R function:

```
### microarray results
decoReport(deco_results_ma, sub, pdf.file = "Report.pdf",
           info.sample = pData(ALCLdata)[,"Type", drop = F],
           cex.names = 0.3, print.annot = T)
### rnaseq results
decoReport(deco_results_rnaseq, sub, pdf.file = "Report.pdf",
           info.sample = clinical.data.tcga[, 1:2, drop=F],
           cex.names = 0.3, print.annot = T)
```

A main result of DECO analysis is the *h statistic matrix* derived from both combination of RDA and NSCA information. In this way, `decoReport()` generates the **heatmap** representation of this *h statistic matrix* that includes a double correlation analysis between samples and between features and two derived clustering dendrograms. In this way the **heatmap** reveals subclasses of samples and feature patterns.

Heatmap (based on double correlation and clustering of h matrix) including features found.

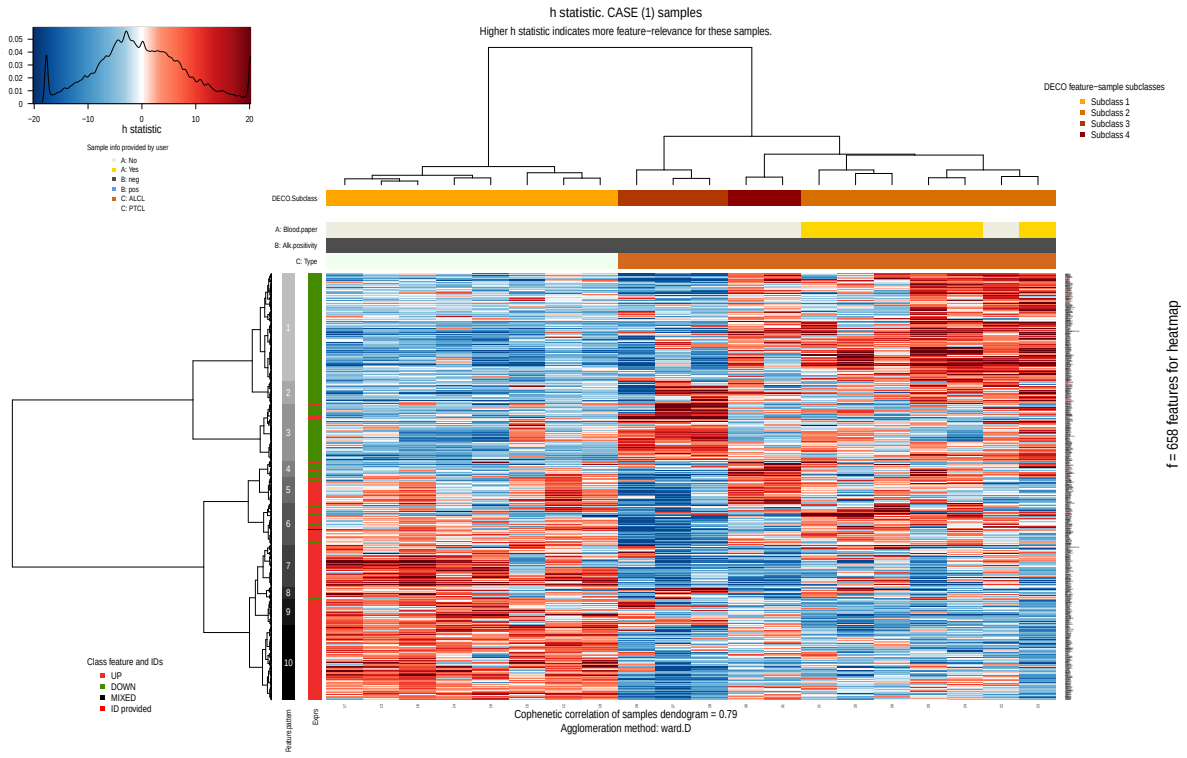


Figure 4: Heatmap of h statistic values from DECO, included in PDF generated by decoReport() function.

Further information about all plots included in the PDF report (`decoReport()`) can be found using *help*.

7.2 Generating plots and profiles for a specific feature (i.e. gene profiling)

Additionally, `plotDECOProfile()` R function provides a way to visualize a single feature profile.

Here, we show the examples for two genes discovered by Scarfo et al. [3] in the analysis of the ALCL samples: **(i) ALK**, that is the key gene-marker used by doctors to separate the two major subtypes of **Anaplastic Large Cell Lymphomas** (in the analysis done with DECO, this gene shows a change-profile *Complete* which supports the value of the gene to separate the ALCL samples); **(ii) ERBB4**, that was reported by authors as biomarker of a new subclass found inside the *negative* ALCL samples (in the analysis done with DECO, this gene shows a change-profile *Minority* indicating the existence of a subset of *negative* ALCL samples that are separated from the rest).

```
### ALK gene profile
plotDECOProfile(deco = deco, id = "ENSG00000171094",
                data = data, pdf.file = "ALK_ALCLdata.pdf",
                cex.samples = 2, info.sample = pData(ALCLdata)[,c(9,8,10)])
```

ALK gene: profile section

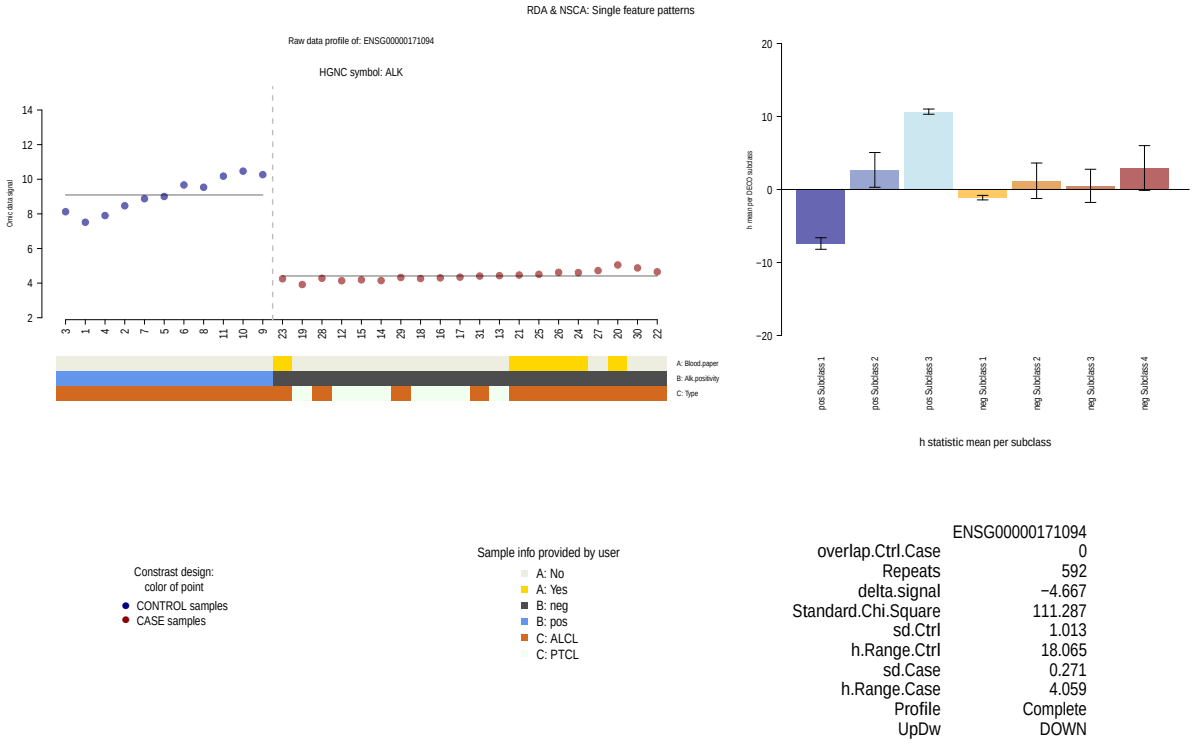
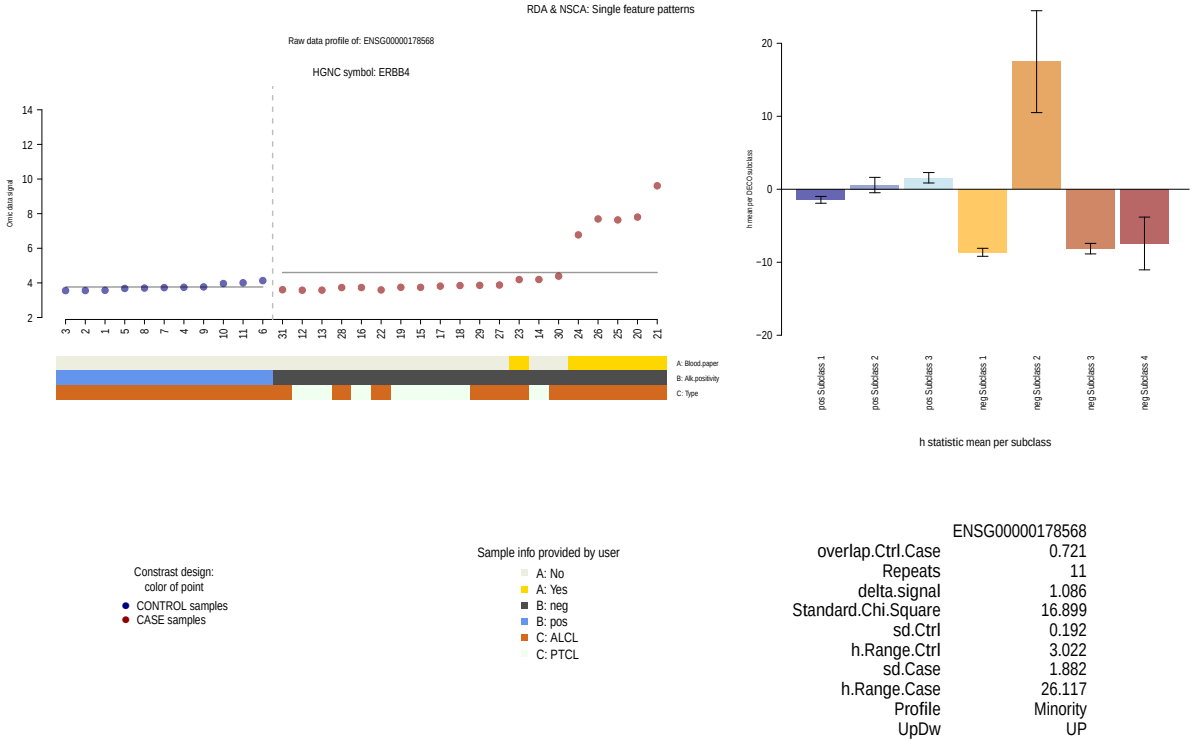


Figure 5: Search for the specific patterns for a feature within the profiles obtained with DECO. The figures correspond to ALK gene and include a plot of its raw expression along samples and a plot of the h statistic of this gene per subclass. This gene shows a change type COMPLETE. The h statistics per subclass found are large for the controls ("positive" ALCLs), and constant and close to 0 for the "negative" ALCLs.

```
### ERBB4 gene profile
plotDECOProfile(deco = deco, id = "ENSG00000178568",
                data = data, pdf.file = "ERBB4_ALCLdata.pdf",
                cex.samples = 2, info.sample = pData(ALCLdata)[,c(9,8,10)])
```

ERBB4 gene: profile section



	ENSG00000178568
overlap.Ctrl.Case	0.721
Repeats	11
delta.signal	1.086
Standard.Chi.Square	16.899
sd.Ctrl	0.192
h.Range.Ctrl	3.022
sd.Case	1.882
h.Range.Case	26.117
Profile	Minority
UpDw	UP

Figure 6: Search for the specific patterns for a feature within the omic profile derived from the RDA-NSCA results. The figures correspond to ERBB4 gene and include a plot of its raw expression along samples and a plot of the h statistic of this gene per subclasses. This gene shows a profile of change type MINORITY, that reveals a different behaviour for a subset of samples inside the "negative" ALCLs. The h statistics per subclass found in this case do not change for the controls (blue boxes corresponding to "positive" ALCL samples), but change a lot within the "negative" ALCL samples, indicating that is a clear marker of this group (segregating a subtype inside that corresponds to negative subclass 2) (see also Figure 3).

8. References

1: Campos-Laborie FJ, Risueño A, Roson-Burgo B, Droste C, Fontanillo C, Ortiz-Estevez M, Trotter MW and De Las Rivas J (2018). **Decomposing heterogeneous population cohorts for patient stratification and discovery of subclass biomarkers using omic data profiling.** Article submitted.

2: Ritchie ME, Phipson B, Wu D, Hu Y, Law CW, Shi W and Smyth GK (2015). **limma powers differential expression analyses for RNA-sequencing and microarray studies.** *Nucleic Acids Res.*, 43:e47. doi:10.1093/nar/gkv007.

3: Scarfo I, Pellegrino E, Mereu E, Kwee I, Agnelli L, Bergaggio E, Garaffo G, Vitale N, Caputo M, Machiorlatti R, Circosta P, Abate F, Barreca A, Novero D, Mathew S, Rinaldi A, Tiacci E, Serra S, Deaglio S, Neri A, Falini B, Rabadan R, Bertoni F, Inghirami G, Piva R; European T-Cell Lymphoma Study Group (2016). **Identification of a new subclass of ALK-negative ALCL expressing aberrant levels of ERBB4 transcripts.** *Blood*, 127:221-232. doi:10.1182/blood-2014-12-614503.

4: Risueño A, Fontanillo C, Dinger ME, De Las Rivas J (2010). **GATExplorer: genomic and transcriptomic explorer; mapping expression probes to gene loci, transcripts, exons and ncRNAs.** *BMC Bioinformatics*, 11:221. doi:10.1186/1471-2105-11-221.

5: MacDonald JW and Ghosh D (2006). **COPA–Cancer Outlier Profile Analysis.** *Bioinformatics*, 22:2950-2951. doi:10.1093/bioinformatics/btl433

6: Ciriello G, Gatza ML, Beck AH, Wilkerson MD, Rhie SK, Pastore A, Zhang H, McLellan M, Yau C, Kandoth C, Bowlby R, Shen H, Hayat S, Fieldhouse R, Lester SC, Tse GM, Factor RE, Collins LC, Allison KH, Chen YY, Jensen K, Johnson NB, Oesterreich S, Mills GB, Cherniack AD, Robertson G, Benz C, Sander C, Laird PW, Hoadley KA, King TA; TCGA Research Network., Perou CM (2015). **Comprehensive molecular portraits of invasive lobular breast cancer**. *Cell*, 163:505-519. doi:10.1016/j.cell.2015.09.033.

7: Lauro N and D'Ambra L (1984). **L'analyse non symetrique des correspondances.** In: *Data Analysis and Informatics III*, (Diday E, Jambu M, Lebart L, Pages J and Tomassone R Eds.). North Holland, Amsterdam, p.433-446.

8: Sokal RR and Rohlf FJ (1962). **The comparison of dendrograms by objective methods.** *Taxon*, 11, 33.