Multiple Co-inertia Analysis of Multiple OMICS Data using *omicade4*

Chen Meng and Amin Moghaddas Gholami

October 17, 2016

Contents

1 Introduction

Modern "omics" technologies enable quantitative monitoring of the abundance of various biological molecules in a high-throughput manner, accumulating an unprecedented amount of quantitative information on a genomic scale. Systematic integration and comparison of multiple layers of information is required to provide deeper insights into biological systems.

Multivariate approaches have been applied successfully in the analysis of high throughput "omics" data. Principal component analysis (PCA) has been shown to be useful in exploratory analysis of linear trends in biological data [**?**]. Culhane and colleagues employed a two table coupling method (co-inertia analysis, CIA) to examine covariant gene expression patterns between microarray datasets from two different platforms [**?**]. Although PCA is available in several R packages, the *ade4* and *made4* contain many additional multivariate statistical methods including methods for analysis of one data table, coupling of two data tables or multi-table analysis [**?** **?**]. These methods for integrating multiple datasets make these particular packages very attractive for analysis of multi-omics data. *omicade4* is developed as an extension to *ade4* and *made4* to facilitate input and analysis of more than two omics datasets.

omicade4 provides functions for multiple co-inertia analysis and for graphical representation, so that the general similarity of different datasets could be easily interpreted. The method could be applied when several set of variables (genes, transcripts, proteins) are measured on the same set of individuals (cell lines, patients). This vignette provides a case study on a toy NCI-60 dataset to show the usage of this package. In addition, the package provides methods for S3 class *cia*, which encapsulates results from the co-inertia analysis by `cia` function from *made4*. Therefore, functions from *made4* and *ade4* are also called in this vignette. For more information please refer to [**?**] and several recent reviews.

2 Quick Start

The package includes example data from four different microarray platforms (i.e., Agilent, Affymetrix HGU 95, Affymetrix HGU 133 and Affymetrix HGU 133plus 2.0) on the NCI-60 cell lines. The package and datasets are loaded by the commands:

```
> library(omicade4)
> data(NCI60_4arrays)
```

`NCI60_4arrays` is a list containing the NCI-60 microarray data with only few hundreds of genes randomly selected in each platform to keep the size of the Bioconductor package small. However, the full datasets are available in [**?**].

2.1 Data Overview

MCIA links the individuals (samples in column) in different datasets and thus the columns will be linked between the multiple datasets. Thus we have to ensure that the order of samples (the columns) in all datasets is the same before performing MCIA. The number of variables (genes) does not need to be the same. We can check the dimension of each dataset in the list by

```
> sapply(NCI60_4arrays, dim)
```

```
     agilent hgu133 hgu133p2 hgu95
[1,]     300    298       268   288
[2,]      60     60        60    60
```

And check whether samples are ordered correctly

```
> all(apply((x <- sapply(NCI60_4arrays, colnames))[,-1], 2, function(y)
+ identical(y, x[,1])))
```

```
[1] TRUE
```

Before performing the MCIA, we can use hierarchical clustering to have a general idea about similarity of cell lines, which can be done with the following command. We will compare the clustering result with MCIA.

```
> layout(matrix(1:4, 1, 4))
> par(mar=c(2, 1, 0.1, 6))
> for (df in NCI60_4arrays) {
+   d <- dist(t(df))
+   hcl <- hclust(d)
+   dend <- as.dendrogram(hcl)
+   plot(dend, horiz=TRUE)
+ }
```

2.2 Data Exploration with Multiple Co-inertia Analysis

The main function `mcia` can be used to perform multiple co-inertia analysis:

```
> mcoin <- mcia(NCI60_4arrays, cia.nf=10)
> class(mcoin)
```

```
[1] "mcia"
```

It returns an object of class *mcia*. There are several methods that could be applied on this class. To visualize the result, one can use `plot` directly. However, because there are nine cancer types, we want to distinguish the cell lines by their original cancer type. This can be done by defining a phenotype factor in `plot`. The following commands create a vector to indicate the cell line groups.

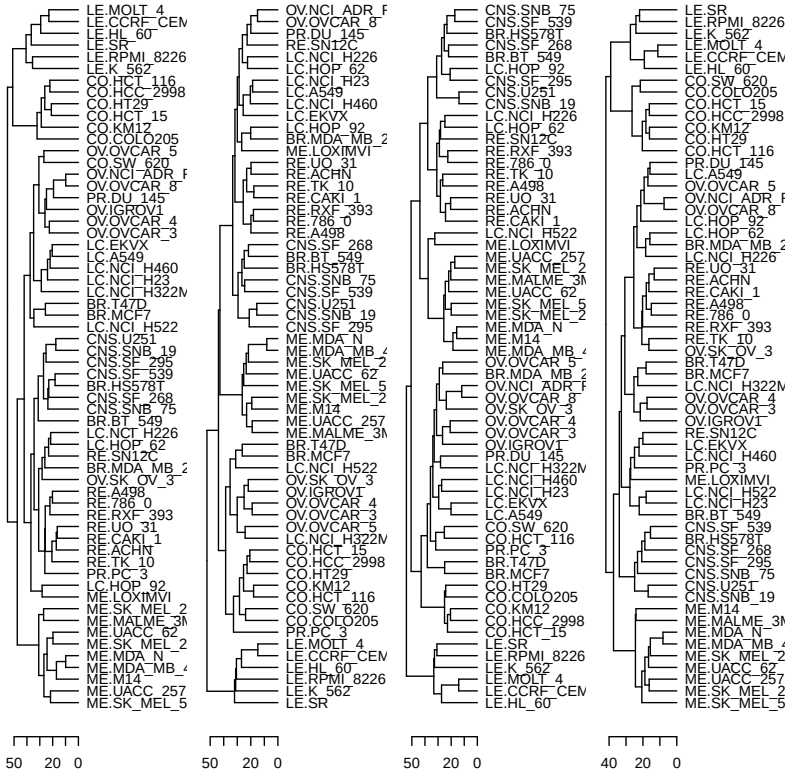

Figure 1: The hierarchical clustering of NCI-60 cell lines

```
> cancer_type <- colnames(NCI60_4arrays$agilent)
> cancer_type <- sapply(strsplit(cancer_type, split="\\."), function(x) x[1])
> cancer_type
```

```
 [1] "BR"  "BR"  "BR"  "BR"  "BR"  "CNS" "CNS" "CNS" "CNS" "CNS" "CNS" "CO"  "CO"  "CO"
[15] "CO"  "CO"  "CO"  "CO"  "LE"  "LE"  "LE"  "LE"  "LE"  "LE"  "ME"  "ME"  "ME"  "ME"
[29] "ME"  "ME"  "ME"  "ME"  "ME"  "ME"  "LC"  "LC"  "LC"  "LC"  "LC"  "LC"  "LC"  "LC"
[43] "LC"  "OV"  "OV"  "OV"  "OV"  "OV"  "OV"  "OV"  "PR"  "PR"  "RE"  "RE"  "RE"  "RE"
[57] "RE"  "RE"  "RE"  "RE"
```

Next, we plot the result for the first two principal components

```
> plot(mcoin, axes=1:2, phenovec=cancer_type, sample.lab=FALSE, df.color=1:4)
```

This command produces a 4-panel figure as shown in figure **??**. The top left panel is the sample space, where each cell line is projected. Shapes represent samples in different platforms. Same cell lines are linked by edges. The shorter the edge, the better the correlation of samples in different platforms. In our sample plot, a relatively high correlation of all microarray datasets is depicted by the short edges. Furthermore, in most cancer types except lung cancer and breast cancer, cell lines having the same origin are closely projected, which indicates high homogeneity of these cancer types. This agrees with the hierarchical clustering (figure **??**).

The next interesting question is which genes are responsible for defining the coordinates of samples. The top right panel is the variable (gene) space, e.g., genes from different platforms, which are distinguished by colors

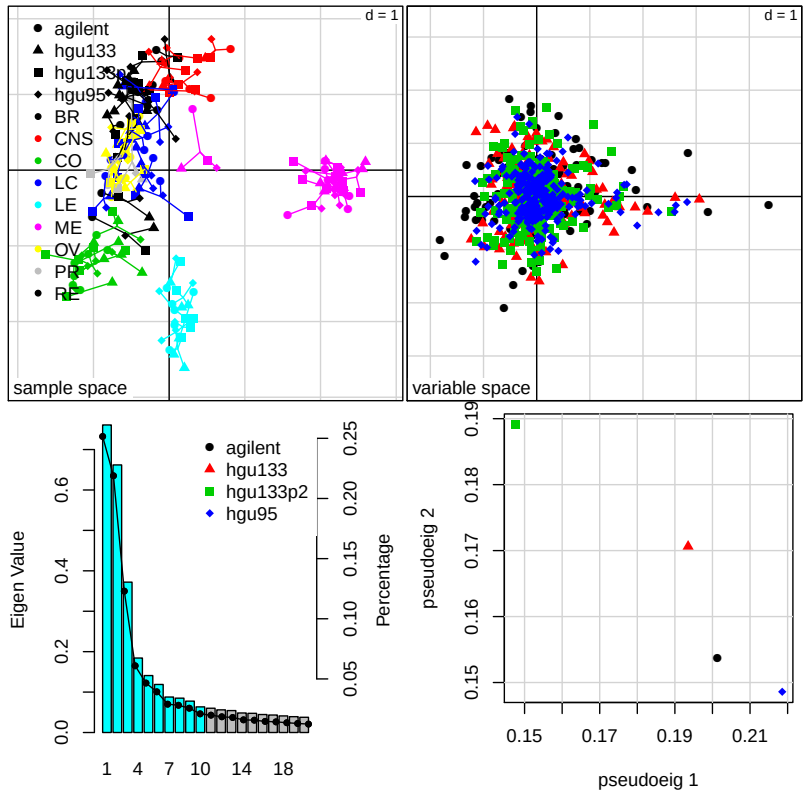

Figure 2: The MCIA plot of NCI-60 data

and shapes, are projected on this space. In this panel, a gene that is particularly highly expressed in a certain cell line will be located on the direction of this cell line. The farther away towards the outer margin, the stronger the association is. Equally, genes projected on the opposite direction from the origin indicate that they are lost or down regulated in those cell lines. From this sense, since the melanoma cell lines are highly weighted on the positive side of the horizontal axis in the first panel, the corresponding melanoma highly expressed genes are on the same direction. The following command could be used to select melanoma associated genes according to the coordinate of genes in that space

```
> melan_gene <- selectVar(mcoin, a1.lim=c(2, Inf), a2.lim=c(-Inf, Inf))
> melan_gene
```

```
       var agilent hgu133 hgu133p2 hgu95 stat
1  ST8SIA1    TRUE  FALSE    FALSE FALSE    1
2   S100A1    TRUE   TRUE    FALSE  TRUE    3
3 C10orf90    TRUE  FALSE    FALSE FALSE    1
4    S100B    TRUE   TRUE    FALSE FALSE    2
5    GPNMB   FALSE   TRUE    FALSE FALSE    1
6 C6orf218   FALSE  FALSE     TRUE FALSE    1
7     ACP5   FALSE  FALSE    FALSE  TRUE    1
8     PLP1   FALSE  FALSE    FALSE  TRUE    1
9    SOX10   FALSE  FALSE    FALSE  TRUE    1
```

The first column represents gene names, the subsequent columns indicate which genes are identified in which platforms, and the last column is a statistic of the total number of platforms identifying the corresponding

gene in the selected region.

The bottom left panel in figure **??** shows the eigenvalue for each eigenvector. The barplot represents the absolute eigenvalues. The dots linked by lines indicate the proportion of variance for the eigenvectors. Cyan bars indicate the eigenvectors kept in the analysis. In this case, we kept 10 eigenvectors, and the top three axes have a relative large eigenvalue according to the scree plot. Therefore, not only the top two axes, but also the third one could lead to some interesting findings. Different axes could be explored by changing the `axes` argument in `plot`, such as:

```
> plot(mcoin, axes=c(1, 3), phenovec=cancer_type, sample.lab=FALSE, df.color=1:4)
> plot(mcoin, axes=c(2, 3), phenovec=cancer_type, sample.lab=FALSE, df.color=1:4)
```

Finally, the bottom right panel in figure **??** shows the pseudo-eigenvalues space of all datasets, which indicates how much variance of an eigenvalue is contributed by each dataset. In this example, the HGU 95 is highly weighted on the first axis. Therefore, this dataset contributes the most variance on this axis among four datasets. However, the HGU 133 plus 2.0 data highly contribute to the second axis. Note that we selected some melanoma related genes by limiting the first axis using `selectVar` function, where we identified four genes in Agilent and HGU 95 platforms comparing to only one gene in the HGU 133 plus 2.0 platform, which is in agreement with the result suggested by this plot.

In addition, the function `plotVar` could be used to visualize the gene space, given a list of genes of interest. Let's get back to the melanoma genes again, we know that *S100B* and *S100A1* are detected in more than one dataset. Now, we want to know where these genes are projected on the gene space. This could be visualized by

```
> geneStat <- plotVar(mcoin, var=c("S100B", "S100A1"), var.lab=TRUE)
```

The output plot is shown in figure **??**.

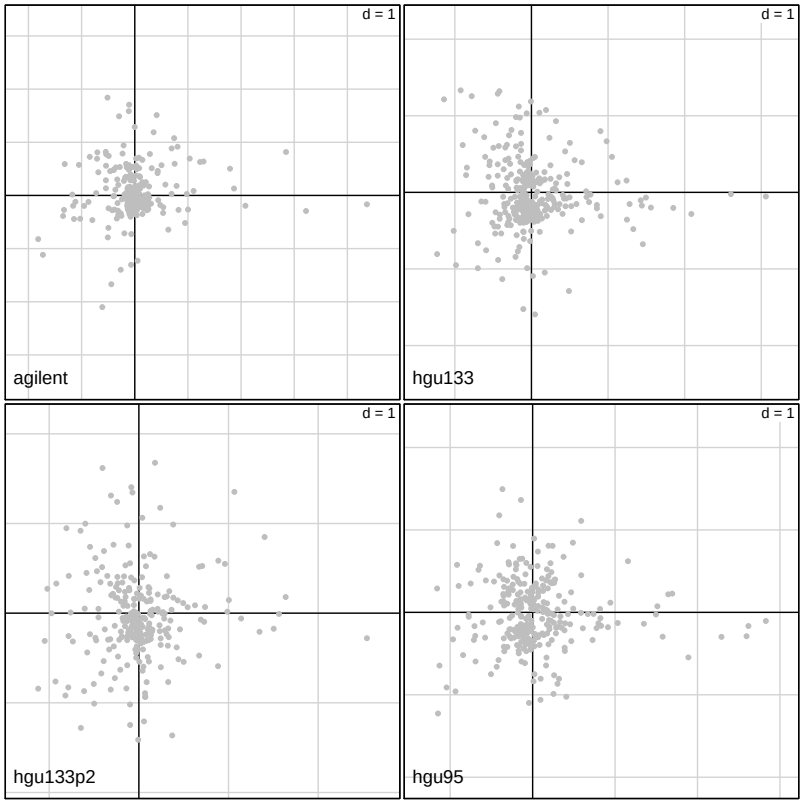

Figure 3: visualization of genes of interest in CIA