Vignette 1 - Understanding the Results of MCIA

Max Mattessich Joaquin Reyna Anna Konstorum

9/1/2022

Introduction

In the mini-lectures we covered in detail the math behind MCIA as well as a generic pipeline for using MCIA. In this vignette we will cover the most important functions within the nipalsMCIA packages as well as downstream analyses that help can interpret the MCIA decomposition. MCIA is applicable to any kind of multi-block data. For this vignette we are using a cancer data set from (Meng et al., 2016) and includes 21 subjects with three data blocks. The data blocks include mRNA levels (12895 features), microRNA levels (537 features) and protein levels (7016 features). Without a multi-block method, researchers may try to use feature reduction methods on each block individually but this strategy ignores relationships between different blocks or under-appreciates signals which are specific to one block. In the context of the NCI60 data set and this vignette, we will show you the power of MCIA to find important relationships between mRNA, microRNA and proteins. More specifically, we will show you how to interpret the global factor scores in Part 1 and global loadings in Part 2.

Preview of the NCI60 dataset

The NCI60 data set has been included with the nipalsMCIA package and is easily available as show below:

```
# load the dataset, uses the name data_blocks
data(NCI60)
data_blocks$mrna[1:5,1:3]
```

```
##            5-HT3C2_1_mrna A1BG-AS1_2_mrna A2LD1_3_mrna
## CNS.SF_268           0.53            0.35        -0.05
## CNS.SF_295          -0.42            0.54        -1.04
## CNS.SF_539           0.00            0.80         0.85
## CNS.SNB_19           0.50           -0.24         0.12
## CNS.SNB_75          -0.27           -0.88        -0.36
```

```
data_blocks$miRNA[1:5,1:3]
```

```
##            MI0000060_miRNA 1 MI0000061_miRNA 2 MI0000061_miRNA 3
## CNS.SF_268             11.91              6.71             13.11
## CNS.SF_295             11.94              7.13             12.86
## CNS.SF_539             11.50              5.79             12.10
## CNS.SNB_19             13.16              6.23             13.65
## CNS.SNB_75             12.63              6.62             12.97
```

```
data_blocks$prot[1:5,1:3]
```

```
##            STAU1_1_prot NRAS_2_prot HRAS_3_prot
## CNS.SF_268     5.712331    7.385177    5.758845
## CNS.SF_295     0.000000    6.327175    0.000000
## CNS.SF_539     0.000000    6.597432    0.000000
## CNS.SNB_19     0.000000    6.891811    0.000000
## CNS.SNB_75     0.000000    6.125612    0.000000
```

Running and Reviewing the MCIA output
We can compute the MCIA decomposition with the first 10 global factors by using:

```
mcia_results <- nipals_multiblock(data_blocks, preprocMethod='colprofile',
                                  num_PCs = 10, tol=1e-12, plots = 'none')
```

The results is a list of various decomposition matrices for MCIA and few other values:

```
names(mcia_results)
```

```
## [1] "global_scores"       "global_loadings"     "block_score_weights"
## [4] "block_scores"        "block_loadings"      "eigvals"
## [7] "preprocMethod"
```

`global_scores` refers to $f^{(j)}$'s, `global_loadings` to $a^{(j)}$'s, `block_score_weights` to ?, `block_scores` to $f_k^{(j)}$'s, `block_loadings` $a_k^{(j)}$'s, *eigvals* to $?", `preprocMethod` to the pre-processing method used and `metadata` to any sample meta-data that has been passed. More details on each of the matrices are covered as part of the math mini-lecture however we will briefly review the `global_scores` and `global_loadings`.

The `global_scores` matrix is represented by $\mathbf{F}$ that is $n \times r$, where n is the number of samples and r is the number of factors chosen by the user with the `num_PCs = r` argument. Each column of this matrix represents one of the orders of global factors computed, i.e.

$$\mathbf{F} = \begin{pmatrix} | & | & & | \\ \mathbf{f}^{(1)} & \mathbf{f}^{(2)} & \dots & \mathbf{f}^{(r)} \\ | & | & & | \end{pmatrix} \in \mathbb{R}^{n \times r}$$

This matrix encodes a low-dimensional representation of the data set, with the i-th row representing a set of r-dimensional coordinates for the i-th sample.

The `global_scores` matrix is represented by $\mathbf{A}$ tht is $p \times r$, where p is the number of features across all omics and r is as before. Each column of this matrix represents one of the orders of global loadings computed, i.e.

$$\mathbf{A} = \begin{pmatrix} | & | & & | \\ \mathbf{a}^{(1)} & \mathbf{a}^{(2)} & \dots & \mathbf{a}^{(r)} \\ | & | & & | \end{pmatrix} \in \mathbb{R}^{p \times r}$$

This matrix encodes the contribution of each feature to the low-dimensional representation.

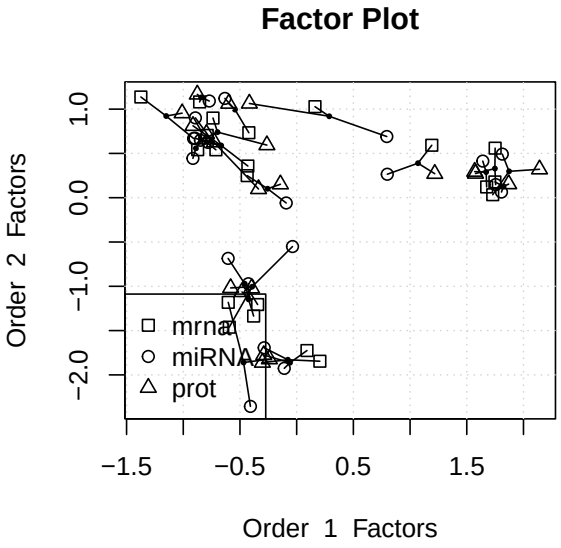
The remainder of this vignette will be broken down into two sections, **Part 1: Interpreting Global Factor Scores** and **Part 2: Interpreting Global Loadings** where we show how to interpret $\mathbf{F}$ and $\mathbf{A}$, respectively.

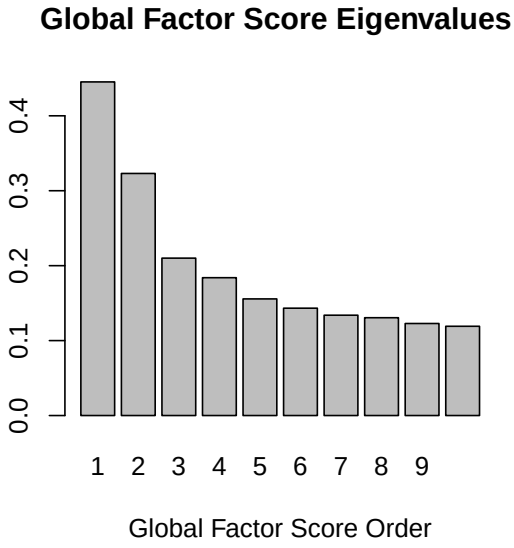
Part 1: Interpreting Global Factor Scores

nipals_multiblock() Generates Basic Visualizations

In the introduction we showed how to calculate the MCIA decomposition using `nipals_multiblock()` but used the parameter `plots='none'`. By default, this function will generate two plots which help establish an initial intuition for the MCIA decomposition. Here we will re-run `nipals_multiblock()` and omit the `plots (default=all)` parameter:

```
mcia_results <- nipals_multiblock(data_blocks, preprocMethod='colprofile',
                                  num_PCs = 10, tol=1e-12)
```





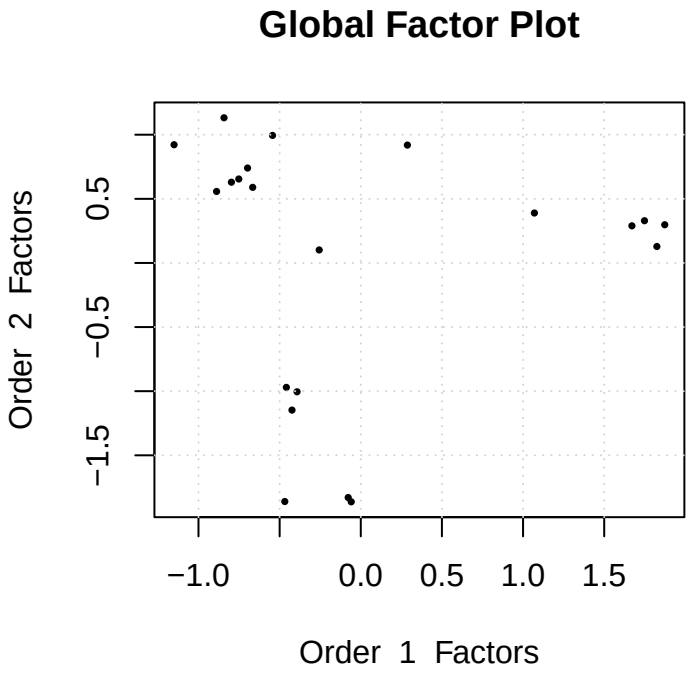
The first plot returned visualizes the first two global factors, with the block factors plotted as shapes connected to the global factors. If a block (in this case, one omics type) is plotted far from its corresponding global factor, this is an indication that the block does not agree with the whole-dataset trend. This may indicate batch effects in data collection, or indicate some underlying difference between blocks.

The second plot returned is a scree plot of factor singular values, where higher values indicate a given order of factor is more important. This can be interpreted exactly the same as an eigenvalue scree plot in principal component analysis.

Visualizing only global factor scores

For clustering, it is useful to only look at global factors (Without block factors). The `MCIA_plots()` function can be used to generate this plot with the `projection_global` argument:

```
MCIA_plots(mcia_results,'projection_global',orders = c(1,2), legend_loc = "bottomleft")
```



It may be helpful to color points by some external label. The `MCIA_plots()` function can do this with a `metadata` argument. The `metadata` object is a dataframe of labels for each sample, possibly containing multiple types of external data. For instance, each of the 21 samples in the NCI-60 dataset relates to one of three cancer types: CNS, Leukemia, or Melanoma. Thus we create the metadata object with the `cancerType` column:

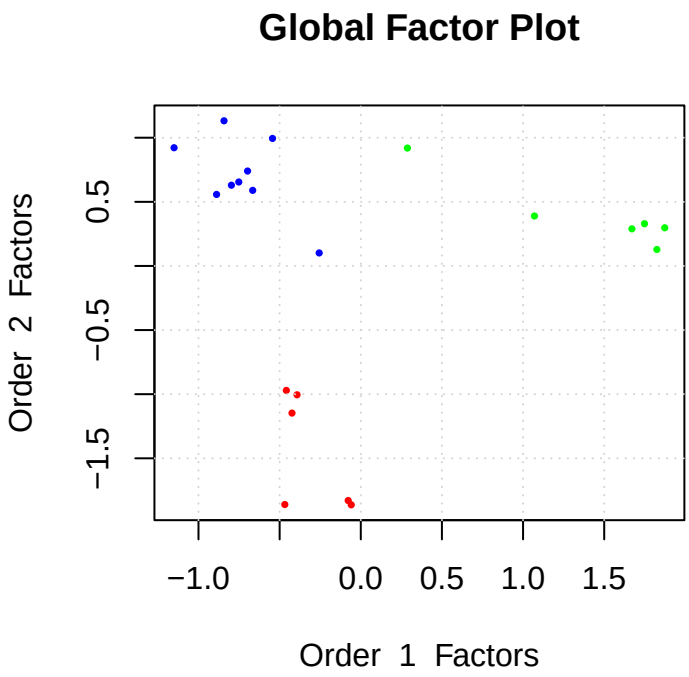
```
CNS = 1:6; LEU = 7:12; ME = 13:21;
nameslist <- list(1:21)
nameslist[CNS] <- "CNS"
nameslist[LEU] <- "Leukemia"
nameslist[ME] <- "Melanoma"
metadata_NCI60 <- data.frame(cancerType = unlist(nameslist))
row.names(metadata_NCI60) <- rownames(data_blocks[[1]])

#View(metadata_NCI60)
```

This object can be passed into the `metadata` argument of `nipals_multiblock()` or added directly to `mcia_results`.

```
mcia_results <- nipals_multiblock(data_blocks, preprocMethod='colprofile',
                                  metadata = metadata_NCI60,
                                  plots = 'none',num_PCs = 10, tol=1e-12)
# Alternative method for adding metadata:
mcia_results$metadata <- metadata_NCI60
```

The `coloring` argument of `MCIA_plots()` can be used to determine which column of `metadata` is used for coloring the projection plots. In this case the column is `cancerType`:

```
MCIA_plots(mcia_results,'projection_global',orders = c(1,2),coloring = 'cancerType',
           legend_loc = "bottomleft")
```

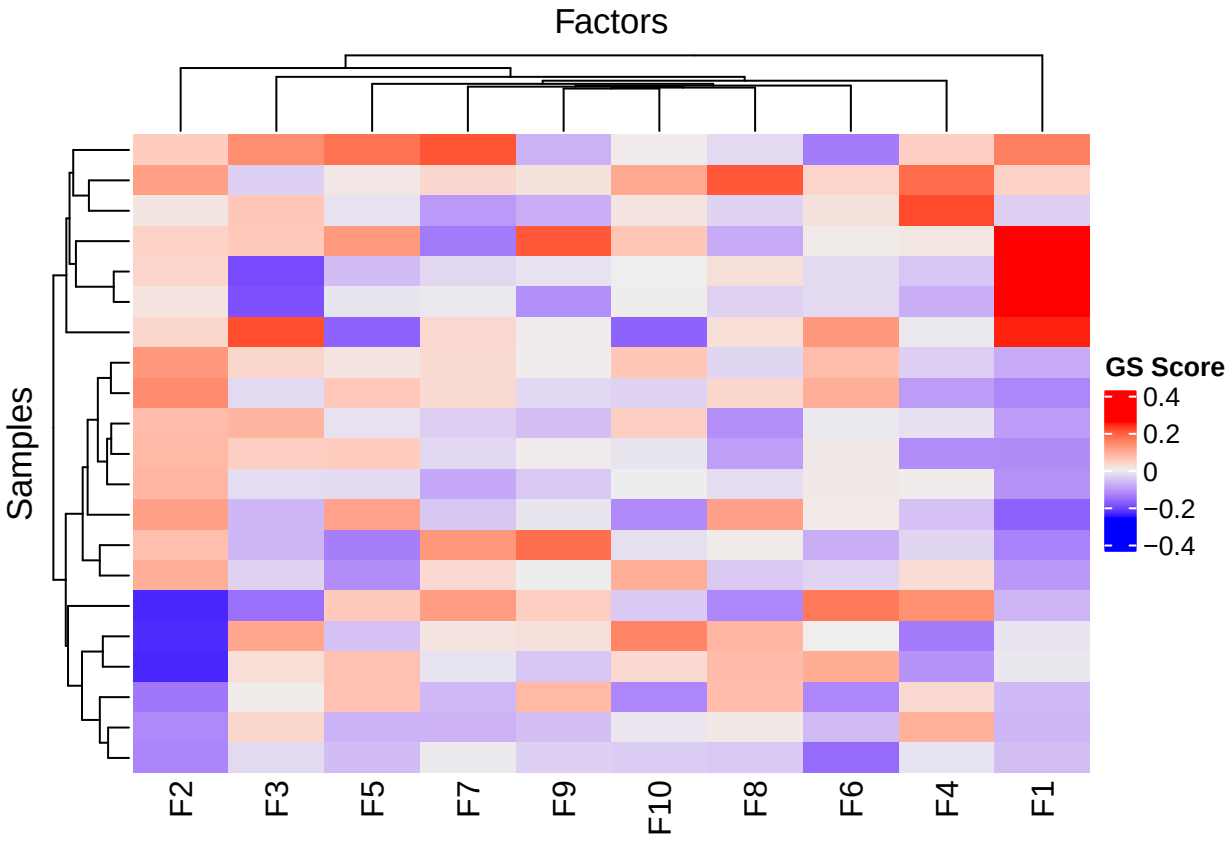


Clusters of samples can be computed from the global factors using any method, such as k-means clustering.

Visualizing the clustering of samples by factor scores

We may also be interested in the clustering of samples by their factors scores, to do this analyses we will the the XX function.

```
gs_scores = mcia_results$global_scores
colnames(gs_scores) = paste0('F', seq(1, ncol(mcia_results$global_scores)))
p = ComplexHeatmap::Heatmap(gs_scores,
    name = "GS Score",
    column_title = "Factors",
    row_title = "Samples",
    row_names_gp = grid::gpar(fontsize = 7),
    show_column_names = T,
    show_row_names = T,
    row_names_side = "right"
)
p
```



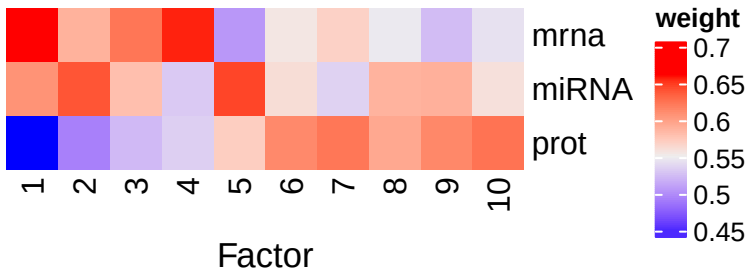
From these heatmap we can see that samples cluster quite strongly by factor 1 signals (to be continued. . . .want to have sample names to explain more if possible)

Part 2: Interpreting Global Loadings

In addition to the global scores matrix, MCIA also calculates a global loadings matrix that is $(m_1 + ... + m_j + ... + m_R) \times k$ where m_j is the number of features within the omics matrix X^j and K is the number of factors calculated. This second matrix provides information as to the contribution

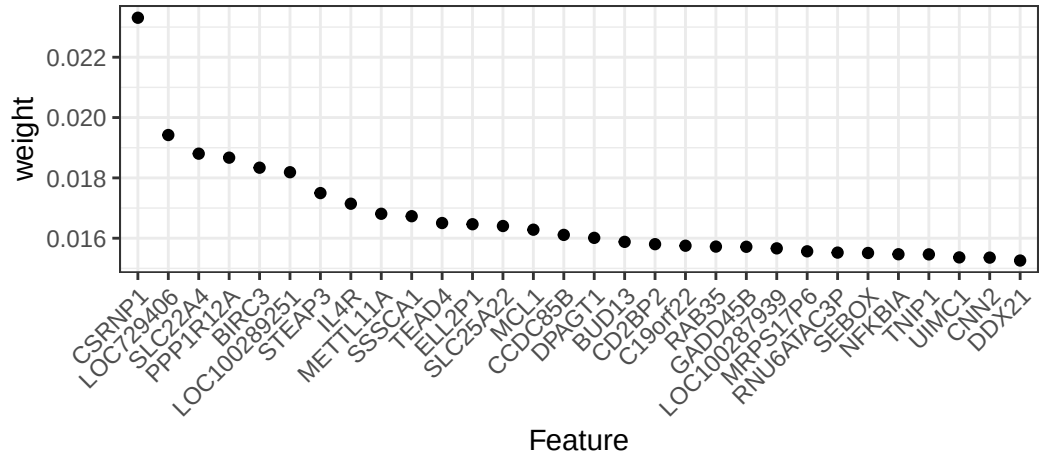
Pseudoeigenvalues representing the contribution of each omic to the global factor score

```
## Should make into a function
bs_weights<-as.matrix(data.frame(mcia_results$block_score_weights))
colnames(bs_weights)<-c(1:10)
Heatmap(bs_weights, cluster_columns=FALSE, cluster_rows=FALSE, name = "weight",
                        column_title_side = "bottom", column_title = "Factor")
```



Scree Plot: Visualizing the top features per factor

```
# make into function
gl<-mcia_results$global_loadings
omic_list<-gsub("^.*_", "", rownames(gl))
factor<-4
gl_f<-data.frame(gl[,factor])
gl_f$omic<-omic_list
colnames(gl_f)<-c("weight",'omic')
gl_f_ord<-gl_f[order(gl_f$weight, decreasing=TRUE),]

# look at mRNA (need to filter since issue with miRNA omic name)
gl_f_ord_mRNA<-gl_f_ord[gl_f_ord$omic=="mrna",]
omic_name<-sub("_.*", "",rownames(gl_f_ord_mRNA) )
gl_f_ord_mRNA$omic_name<-sub("_.*", "",rownames(gl_f_ord_mRNA) )
ggplot(gl_f_ord_mRNA[0:30,], aes(x=factor(omic_name,level=omic_name), y=weight))+
      geom_point()+
      theme_bw()+
      xlab('Feature')+
      theme(axis.text.x=element_text(angle=45, hjust=1))
```



Pathway analysis for the top factors using data from gene-centric omics blocks

. . . (to be continued)