Gene expression analysis comparison with other methods

Amal Thomas

Contents

1 Introduction

Joint matrix factorization facilitates the comparison of expression profiles from different species without using gene mapping. Transforming gene expression profiles into reduced eigengene space using singular value decomposition (SVD) has been shown to capture meaningful biological information (Alter, Brown, and Botstein 2000). Tamayo et al. (2007) used a non-negative matrix factorization approach to learn a low-dimensional approximation of the microarray expression datasets and used the reduced space for comparisons. Matrix factorization-based methods are commonly used for gene expression analysis (Alter, Brown, and Botstein 2000; Tamayo et al. 2007). An orthology independent matrix factorization framework based on generalized singular value decomposition [GSVD; Van Loan (1976)] was used by Alter, Brown, and Botstein (2003) to compare gene-expression profiles from two species. This framework was later extended to develop higher-order generalized singular value decomposition (HO GSVD) to analyze data from more than two species (Ponnapalli et al. 2011). Using cell-cycle gene expression datasets, these approaches have shown examples of genes with highly conserved sequences across species but with significantly different cell-cycle peak times. Although these methods have shown the potential advantages of orthology-independent comparisons, the steps involved in estimating the shared factor and comparing the expression profiles using these methods require complex procedures. When estimating the shared factor, the pairwise quotients and their arithmetic mean involve the computation of inverses. As a result, the biological interpretation of the shared factor is difficult in HO GSVD. Similarly, to place new datasets to the space defined by the shared factor requires the computation of generalized inverses. Moreover, the independence of the columns of the species-specific factors and the shared factor is not guaranteed, making it challenging to differentiate the

contribution of genes/features across different dimensions of the shared factor. These limitations restrict the application of these methods in cross-species studies.

This study developed a joint diagonalization approach called Orthogonal Shared Basis Factorization (OSBF) for cross-species expression comparisons. In this workflow, we will compare the gene expression analysis results of OSBF with the HO GSVD factorization. We will use the same gene expression datasets that we used for the `GeneExpressionAnalysis` workflow. We will also perform similar analysis steps demonstrated in that workflow.

```
library(SBF)
```

Additional packages required for the workflow

```
# install packages
pkgs <- c("data.table", "dplyr", "matrixStats")
require_install <- pkgs[!(pkgs %in% row.names(installed.packages()))]
if (length(require_install))
  install.packages(require_install)
suppressPackageStartupMessages({
  library(data.table)
  library(dplyr)
  library(matrixStats)
})
```

2 Species and organs

In this workflow, we will work with gene expression profile of six tissues from eight species.

```
species <- c("Homo_sapiens", "Pan_troglodytes", "Macaca_mulatta",
             "Mus_musculus", "Rattus_norvegicus", "Bos_taurus",
             "Sus_scrofa", "Gallus_gallus")
species_short <- sapply(species, getSpeciesShortName)
species_short
#>      Homo_sapiens   Pan_troglodytes    Macaca_mulatta      Mus_musculus
#>        "hsapiens"    "ptroglodytes"        "mmulatta"       "mmusculus"
#> Rattus_norvegicus        Bos_taurus        Sus_scrofa     Gallus_gallus
#>     "rnorvegicus"         "btaurus"         "sscrofa"         "ggallus"
# common tissues present in all 8 species
tissues <-  c("brain", "heart", "kidney", "liver", "lung", "testis")
```

3 Load gene expression profiles

Download the processed RNA-Seq counts file ("counts.tar.gz") from https://doi.org/10.6084/m9.figshare.20216858 Uncompress the .tar.gz file and add it to the working directory.

```
# set the path to the working directory. Change this accordingly
path <- "~/Dropbox/0.Analysis/0.paper/"
counts_list <- metadata_list <- list()
require(data.table)
for (sp in species) {
  # read logTPM counts for each species
  counts <- data.table::fread(paste0(path, "counts/", sp, "_logTPM.tsv"),
                              sep = "\t", header = TRUE, data.table = FALSE,
                              nThread = 4)
  row.names(counts) <- counts$V1
```

```
  counts$V1 <- NULL
  col_fields <- data.table::tstrsplit(colnames(counts), "_")
  metadata <- data.frame(
      project = col_fields[[1]],
      species = col_fields[[2]],
      tissue = col_fields[[3]],
      gsm = col_fields[[4]],
      name = colnames(counts),
      stringsAsFactors = FALSE)
  metadata_sel <- metadata[metadata$tissue %in% tissues, , drop = FALSE]
  counts_sel <- counts[, colnames(counts) %in% metadata_sel$name, drop = FALSE]
  metadata_sel$ref <- seq_len(nrow(metadata_sel))
  metadata_sel$key <- paste0(metadata_sel$species, "_", metadata_sel$tissue)

  counts_list[[sp]] <- counts_sel
  metadata_list[[sp]] <- metadata_sel
}
```

The dimensions and the number of RNA-Seq profiles for different species.

```
sapply(counts_list, dim)
#>      Homo_sapiens Pan_troglodytes Macaca_mulatta Mus_musculus Rattus_norvegicus
#> [1,]        58676           31373          30807        54446              32623
#> [2,]          111              61            124          107                 64
#>      Bos_taurus Sus_scrofa Gallus_gallus
#> [1,]      24596      25880         19152
#> [2,]         53        165           117
```

3.1 Compute mean expression profiles

Now, for each species, let us compute the mean expression profile for each tissue. We will use `calcAvgCounts` function from the `SBF` package.

```
avg_counts <- list()
for (sp in species) {
    avg_counts[[sp]] <- calcAvgCounts(counts_list[[sp]],
                                      metadata_list[[sp]])
}
# check tissue columns are matching in each species
c_tissues <- as.data.frame(sapply(avg_counts, function(x) {
            data.table::tstrsplit(colnames(x), "_")[[2]]
  }))
if (!all(apply(c_tissues, 1, function(x) all(x == x[1])))) {
        stop("Error! tissues not matching")
}
```

The dimension of mean expression profiles

```
sapply(avg_counts, dim)
#>      Homo_sapiens Pan_troglodytes Macaca_mulatta Mus_musculus Rattus_norvegicus
#> [1,]        58676           31373          30807        54446              32623
#> [2,]            6               6              6            6                  6
#>      Bos_taurus Sus_scrofa Gallus_gallus
#> [1,]      24596      25880         19152
#> [2,]          6          6             6
```

Remove genes not expressed.

```
# remove empty rows
removeZeros <- function(df) {
    return(df[rowSums(df) > 0, ])
}

avg_counts <- lapply(avg_counts, removeZeros)
sapply(avg_counts, dim)
#>      Homo_sapiens Pan_troglodytes Macaca_mulatta Mus_musculus Rattus_norvegicus
#> [1,]        47301           26173          27762        39097             23004
#> [2,]            6               6              6            6                 6
#>      Bos_taurus Sus_scrofa Gallus_gallus
#> [1,]      20028      21448         17810
#> [2,]          6          6             6
# update counts_list
counts_list_sub <- list()
for (sp in names(avg_counts)) {
  counts_list_sub[[sp]] <- counts_list[[sp]][row.names(avg_counts[[sp]]), ,
                                             drop = FALSE]
}
```

4 OSBF and HO GSVD

OSBF call

```
t1 <- proc.time()
# decrease tol to minimize error
osbf <- SBF(avg_counts, transform_matrix = TRUE, orthogonal = TRUE,
                    optimizeV = TRUE, tol = 1e-3)#, tol = 1e-10)
#>
#> OSBF optimizing factorization error
t2 <- proc.time()
cat("Time taken:\n")
#> Time taken:
t2 - t1
#>    user  system elapsed
#>  15.347   0.060  15.410
```

HO GSVD factorization call. We will use the `HOGSVD` function from the `SBF` package for this.

```
hogsvd <- HOGSVD(avg_counts)
```

```
names(hogsvd)
#> [1] "v"      "lambda" "u"      "delta"  "s"
```

Let us compare the factorization error of these two approaches. We can use the `calcDecompError` function from the `SBF` package.

```
calcDecompError(avg_counts, osbf$u, osbf$delta, osbf$v)
#> [1] 23483.37
```

This is same as that stored in the `osbf$error`. Now, for HO GSVD, we have

```
calcDecompError(avg_counts, hogsvd$u, hogsvd$delta, hogsvd$v)
#> [1] 5.781512e-25
```

HO GSVD is an exact factorization while that is not the case for OSBF.

4.1 Properties of different factors

Now we will check the properties of U_i's, V, and Δ_i's.

```
zapsmall(osbf$v %*% t(osbf$v))
#>      [,1] [,2] [,3] [,4] [,5] [,6]
#> [1,]    1    0    0    0    0    0
#> [2,]    0    1    0    0    0    0
#> [3,]    0    0    1    0    0    0
#> [4,]    0    0    0    1    0    0
#> [5,]    0    0    0    0    1    0
#> [6,]    0    0    0    0    0    1
```

Estimated V is orthogonal in OSBF.

```
zapsmall(t(hogsvd$v) %*% hogsvd$v)
#>            [,1]       [,2]       [,3]       [,4]       [,5]       [,6]
#> [1,]  1.0000000  0.6191501 -0.8645097 -0.5590745 -0.7776132 -0.1967525
#> [2,]  0.6191501  1.0000000 -0.7570312 -0.5242897 -0.6051307  0.2058670
#> [3,] -0.8645097 -0.7570312  1.0000000  0.6394191  0.8201746  0.1024071
#> [4,] -0.5590745 -0.5242897  0.6394191  1.0000000  0.5242836  0.0444741
#> [5,] -0.7776132 -0.6051307  0.8201746  0.5242836  1.0000000  0.0077950
#> [6,] -0.1967525  0.2058670  0.1024071  0.0444741  0.0077950  1.0000000
```

The shared V is not orthogonal in HO GSVD.

Now let us compare the U_i factors.

```
# get u factor for fist species
osbf_u <- osbf$u[[names(osbf$u)[1]]]
zapsmall(t(osbf_u) %*% osbf_u)
#>      [,1] [,2] [,3] [,4] [,5] [,6]
#> [1,]    1    0    0    0    0    0
#> [2,]    0    1    0    0    0    0
#> [3,]    0    0    1    0    0    0
#> [4,]    0    0    0    1    0    0
#> [5,]    0    0    0    0    1    0
#> [6,]    0    0    0    0    0    1
```

```
# get u factor for fist species
hogsvd_u <- hogsvd$u[[names(hogsvd$u)[1]]]
zapsmall(t(hogsvd_u) %*% hogsvd_u)
#>            [,1]       [,2]       [,3]       [,4]       [,5]       [,6]
#> [1,]  1.0000000  0.0146547  0.1043939  0.0296987  0.0606258 -0.0128124
#> [2,]  0.0146547  1.0000000 -0.3079538  0.0783326  0.0725241 -0.0853503
#> [3,]  0.1043939 -0.3079538  1.0000000 -0.0572603 -0.0227645  0.1058862
#> [4,]  0.0296987  0.0783326 -0.0572603  1.0000000 -0.0088362 -0.0877172
#> [5,]  0.0606258  0.0725241 -0.0227645 -0.0088362  1.0000000  0.0198586
#> [6,] -0.0128124 -0.0853503  0.1058862 -0.0877172  0.0198586  1.0000000
```

The columns of U_i factors are orthonormal in OSBF and not in HO GSVD.

Information represented by different dimensions. For OSBF, the percentage of correlation information (p_{ij}) represented by a common space dimension is defined as $p_{ij} = \delta_{ij}^2 / \sum_{j=1}^{6} \delta_{ij}^2 \times 100$, where $\Delta_i = \text{diag}(\delta_{i1}, \ldots, \delta_{i6})$.

```
percentInfo_osbf <- calcPercentInfo(osbf)
for (i in names(osbf$delta)) {
  cat("\n", sprintf("%-25s:", i), sprintf("%8.2f", percentInfo_osbf[[i]]))
}
#>
#>  Homo_sapiens              :    87.23     5.05     2.99     2.13     1.41     1.19
#>  Pan_troglodytes           :    88.86     4.64     2.70     1.60     1.24     0.96
#>  Macaca_mulatta            :    87.77     4.87     2.98     1.79     1.36     1.22
#>  Mus_musculus              :    81.64     7.99     4.43     2.67     1.67     1.59
#>  Rattus_norvegicus         :    84.10     6.48     3.86     2.51     1.62     1.44
#>  Bos_taurus                :    87.37     4.94     2.96     1.78     1.71     1.24
#>  Sus_scrofa                :    85.53     5.39     3.38     2.41     1.82     1.47
#>  Gallus_gallus             :    88.21     4.15     2.98     1.86     1.56     1.24
```

```
percentInfo_hogsvd <- calcPercentInfo(hogsvd)
for (i in names(hogsvd$delta)) {
  cat("\n", sprintf("%-25s:", i), sprintf("%8.2f", percentInfo_hogsvd[[i]]))
}
#>
#>  Homo_sapiens              :    16.12     6.33    65.17     2.40     6.31     3.66
#>  Pan_troglodytes           :    18.06     4.14    65.63     2.57     6.13     3.46
#>  Macaca_mulatta            :    21.24     5.50    60.84     2.54     6.37     3.51
#>  Mus_musculus              :    29.53     6.95    49.96     2.78     7.18     3.60
#>  Rattus_norvegicus         :    29.19     6.62    50.60     2.75     7.35     3.49
#>  Bos_taurus                :    17.77     4.97    64.86     2.96     6.00     3.44
#>  Sus_scrofa                :    28.68     5.80    52.73     2.94     6.70     3.16
#>  Gallus_gallus             :    16.43     6.42    64.83     2.75     6.35     3.23
```

In OSBF, dimension 1 captures most of inter-tissue correlation relationship as it represents basic cellular functions shared by different species. In HO GSVD dimension 3 captures most of the information. We will first compare these two dimensions.

5 Exploring properties of dimensions

We will first investigate dimension 1 of OSBF and dimension 3 of HO GSVD.

Functions to compute tissue specificity (τ) and scaled average expression profile.

```
# function to compute Tau
calc_tissue_specificity <- function(a) {
    a <- as.matrix(a)
    b <- a / matrixStats::rowMaxs(a)
    return(rowSums(1 - b) / (ncol(b) - 1))
}
Tau <- lapply(avg_counts, function(x) { calc_tissue_specificity(x)})
avg_counts_scaled <- lapply(avg_counts, function(x) { t(scale(t(x)))})

combine_expr <- list()
for (sp in names(avg_counts_scaled)) {
  x <- as.data.frame(avg_counts_scaled[[sp]])
  x[["Tau"]] <- Tau[[sp]]
  combine_expr[[sp]] <- x
}
```

Lets plot the OSBF U_1 loadings vs expression specificity for human

```r
# install packages
pkgs <- c("grid", "ggthemes", "ggplot2")
require_install <- pkgs[!(pkgs %in% row.names(installed.packages()))]
if (length(require_install))
  install.packages(require_install)
suppressPackageStartupMessages({
  library(grid)
  library(ggthemes)
  library(ggplot2)
})
```

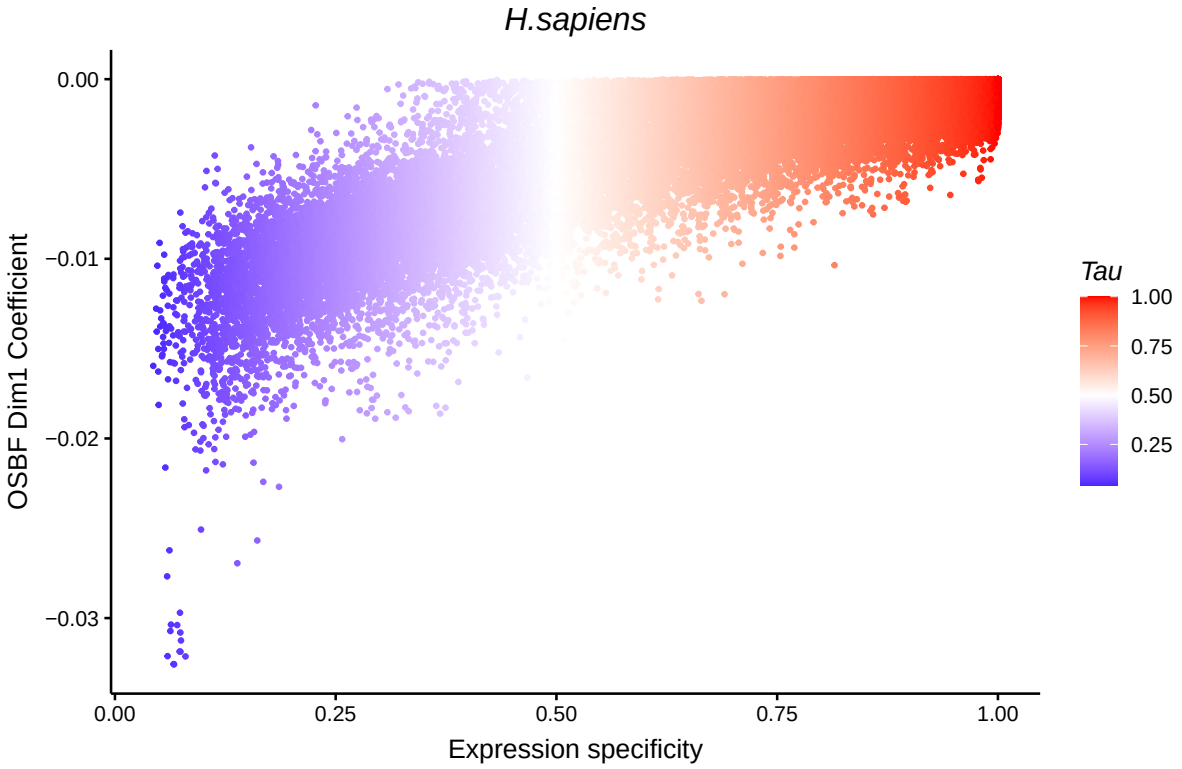
We will use the following custom theme for the ggplots.

```r
# custom theme function for ggplot2
customTheme <- function(base_size = 10, base_family = "helvetica") {
   require(grid)
   require(ggthemes)
   (ggthemes::theme_foundation(base_size = base_size)
   + ggplot2::theme(plot.title = element_text(face = "bold",
                                     size = rel(1.2), hjust = 0.5),
         text = element_text(),
         panel.background = element_rect(colour = NA),
         plot.background = element_rect(colour = NA),
         panel.border = element_rect(colour = NA),
         axis.title = element_text(size = rel(1)),
         axis.title.y = element_text(angle = 90, vjust = 2),
         axis.title.x = element_text(vjust = -0.2),
         axis.text = element_text(),
         axis.line = element_line(colour = "black"),
         axis.ticks = element_line(),
         panel.grid.major = element_blank(),
         panel.grid.minor = element_blank(),
         legend.key = element_rect(colour = NA),
         legend.position = "top",
         legend.direction = "horizontal",
         legend.key.size = unit(0.2, "cm"),
         legend.spacing = unit(0, "cm"),
         legend.title = element_text(face = "italic"),
         plot.margin = unit(c(10, 5, 5, 5), "mm"),
         strip.background = element_rect(colour = "#f0f0f0", fill = "#f0f0f0"),
         strip.text = element_text(face = "bold")))
}
```

```r
sel_dim <- 1
species <- "Homo_sapiens"
expr <- combine_expr[[species]]
osbf_coef <- osbf$u[[species]]
expr[["coef"]] <- osbf_coef[, sel_dim, drop = TRUE]

# plot scatter
mid <- 0.5
p1 <- ggplot2::ggplot(expr, aes(x = Tau, y = coef, col = Tau)) +
  theme_bw() +
  geom_point(size = 0.5) + xlab("Expression specificity") +
  ylab(paste0("OSBF Dim", sel_dim, " Coefficient")) +
```

```
  scale_color_gradient2(midpoint = mid, low = "blue", mid = "white",
                        high = "red", space = "Lab") +
  customTheme() +  theme(legend.position = "right",
                         legend.direction = "vertical") +
  labs(title = getScientificName(species), color = "Tau") +
  theme(legend.key.size = unit(0.5, "cm"),
        plot.title = element_text(face = "italic"))
p1
```



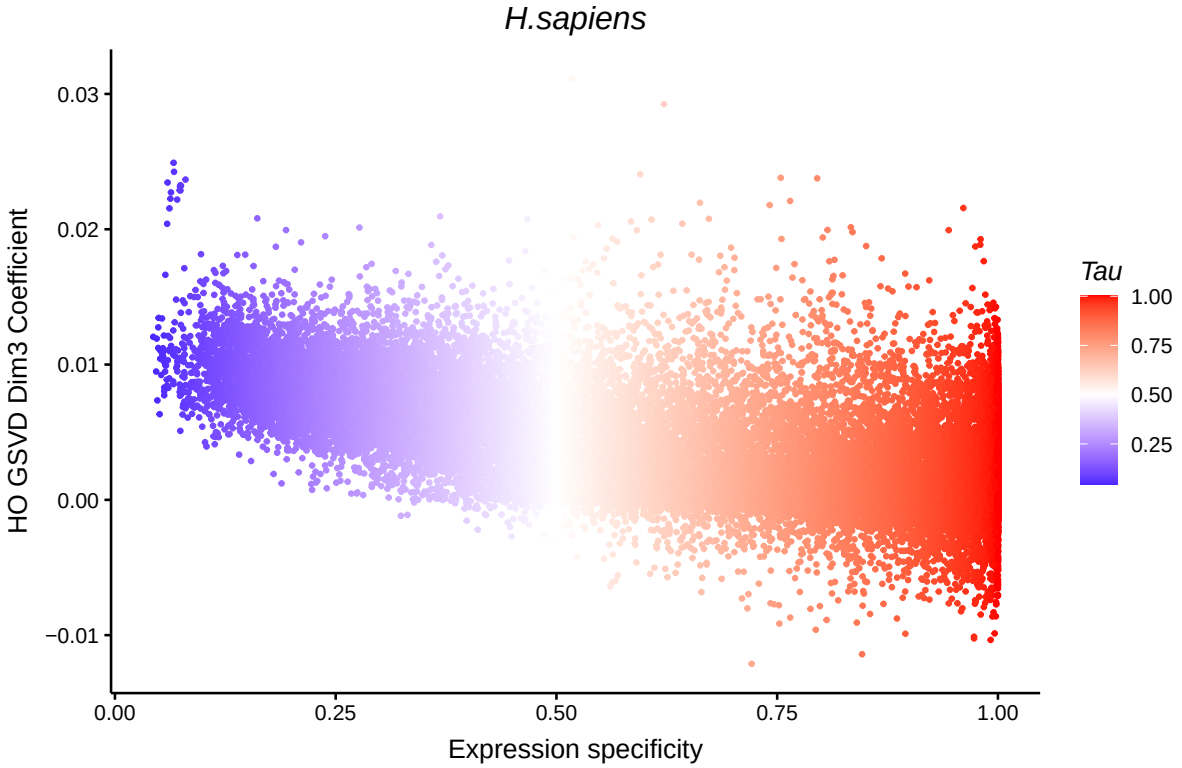
In OSBF, we see that genes with high loading have low expression specificity and are generally house-keeping genes (see Gene expression analysis vignettes).

Now let us plot for the loading vs expression specificity for dimension 3 of HO GSVD

```
sel_dim <- 3
species <- "Homo_sapiens"
expr <- combine_expr[[species]]
hogsvd_coef <- hogsvd$u[[species]]
expr[["coef"]] <- hogsvd_coef[, sel_dim, drop = TRUE]

# plot scatter
mid <- 0.5
p1 <- ggplot2::ggplot(expr, aes(x = Tau, y = coef, col = Tau)) +
  theme_bw() +
  geom_point(size = 0.5) + xlab("Expression specificity") +
  ylab(paste0("HO GSVD Dim", sel_dim, " Coefficient")) +
  scale_color_gradient2(midpoint = mid, low = "blue", mid = "white",
                        high = "red", space = "Lab") +
  customTheme() +  theme(legend.position = "right",
                         legend.direction = "vertical") +
```

```
  labs(title = getScientificName(species), color = "Tau") +
  theme(legend.key.size = unit(0.5, "cm"),
        plot.title = element_text(face = "italic"))
p1
```



We do not see any a trend of high coefficient for low expression specificity here. Thus no direct interpretation is available from this.

Now let us explore properties of other dimensions.

6 Project libraries into common space

In OSBF, the columns of U_i's are orthonormal. This allows us to easily project individual expression profiles to the common space by computing $D_i^T U_i \Delta^{-1}$. We will `projectCounts` function from the `SBF` package for this.

```
df_proj <- projectCounts(counts_list_sub, osbf)
meta1 <- data.table::tstrsplit(row.names(df_proj), "_")
df_proj$tissue <- factor(meta1[[3]])
df_proj$species <- factor(meta1[[2]])
df_proj <- df_proj %>% mutate(species = factor(species,
                              levels = species_short))
```

We will use the same projection strategy ($D_i^T U_i \Delta^{-1}$) for HO GSVD.

```
ho_proj <- projectCounts(counts_list_sub, hogsvd)
meta1 <- data.table::tstrsplit(row.names(ho_proj), "_")
ho_proj$tissue <- factor(meta1[[3]])
ho_proj$species <- factor(meta1[[2]])
ho_proj <- ho_proj %>% mutate(species = factor(species,
```

```
                                levels = species_short))
```

6.1 Clustering in the common space

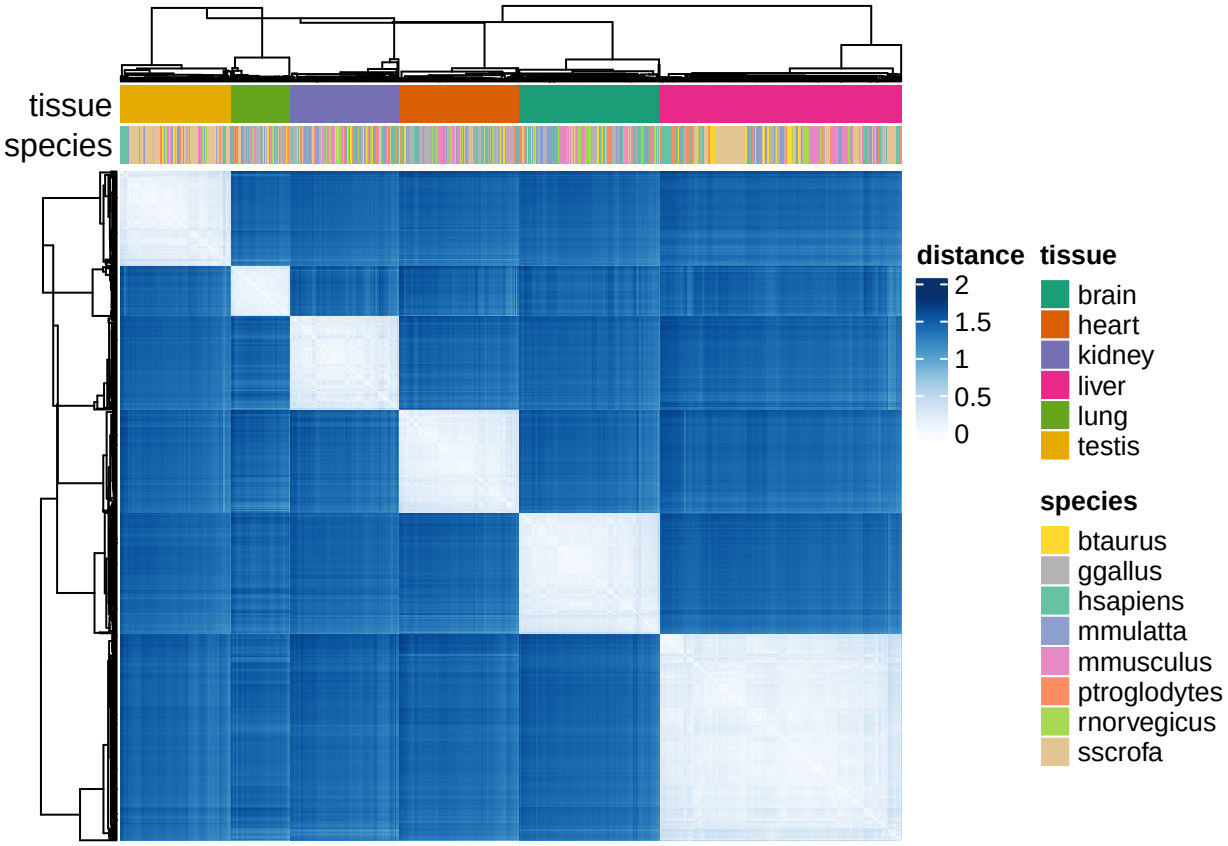
```
# install packages
pkgs <- c("RColorBrewer")
require_install <- pkgs[!(pkgs %in% row.names(installed.packages()))]
if (length(require_install))
  install.packages(require_install)
pkgs <- c("ComplexHeatmap")
require_install <- pkgs[!(pkgs %in% row.names(installed.packages()))]
if (length(require_install)) {
  if (!require("BiocManager", quietly = TRUE))
      install.packages("BiocManager")
  BiocManager::install("ComplexHeatmap")
}
suppressPackageStartupMessages({
  library(ComplexHeatmap)
  library(RColorBrewer)
})
```

Clustering of profiles in OSBF's common space V.

```
data <- df_proj
data$tissue <- NULL
data$species <- NULL
data <- as.matrix(data)
data_dist <- as.matrix(dist(data, method = "euclidean"))
meta <- data.table::tstrsplit(colnames(data_dist), "_")
ht <- ComplexHeatmap::HeatmapAnnotation(tissue = meta[[3]], species = meta[[2]],
                        col = list(tissue = c("brain" = "#1B9E77",
                                              "heart" = "#D95F02",
                                              "kidney" = "#7570B3",
                                              "liver" = "#E7298A",
                                              "lung" = "#66A61E",
                                              "testis" = "#E6AB02"),
                                   species = c("hsapiens" = "#66C2A5",
                                               "ptroglodytes" = "#FC8D62",
                                               "mmulatta" = "#8DA0CB",
                                               "mmusculus" = "#E78AC3",
                                               "rnorvegicus" = "#A6D854",
                                               "btaurus" = "#FFD92F",
                                               "sscrofa" = "#E5C494",
                                               "ggallus" = "#B3B3B3")),
                        #show_annotation_name = F,
                        annotation_name_side = "left")
mypalette <- RColorBrewer::brewer.pal(9, "Blues")
morecolors <- colorRampPalette(mypalette)

myheatmap <- ComplexHeatmap::Heatmap(as.matrix(data_dist), cluster_rows = TRUE,
                                     clustering_method_rows = "centroid",
                                     cluster_columns = TRUE,
                                     clustering_method_columns = "centroid",
                                     top_annotation = ht, col = morecolors(50),
```

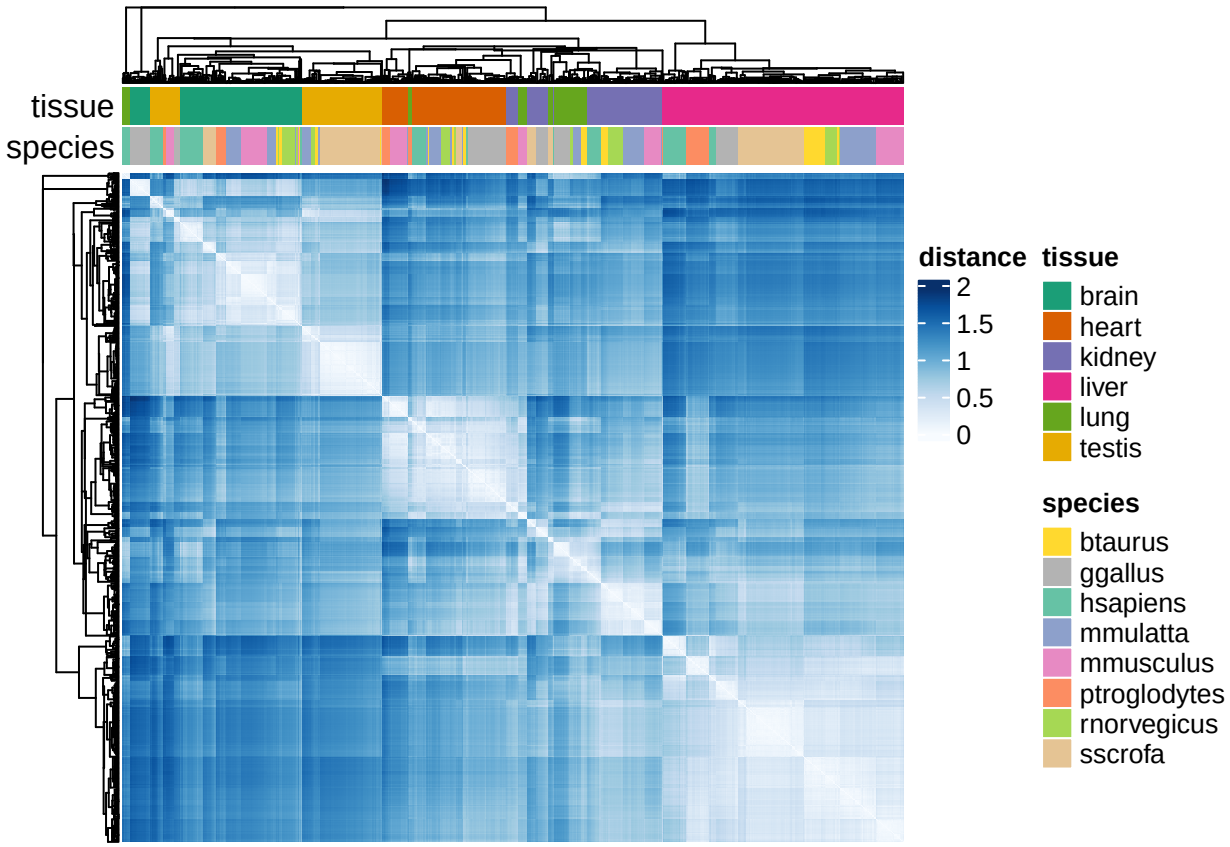
```
                                    show_row_names = FALSE,
                                    show_column_names = FALSE,
                                    name = "distance")
myheatmap
```



Clustering of profiles in HO GSVD's common space V.

```
data <- ho_proj
data$tissue <- NULL
data$species <- NULL
data <- as.matrix(data)
data_dist <- as.matrix(dist(data, method = "euclidean"))
meta <- data.table::tstrsplit(colnames(data_dist), "_")
ht <- ComplexHeatmap::HeatmapAnnotation(tissue = meta[[3]], species = meta[[2]],
                        col = list(tissue = c("brain" = "#1B9E77",
                                              "heart" = "#D95F02",
                                              "kidney" = "#7570B3",
                                              "liver" = "#E7298A",
                                              "lung" = "#66A61E",
                                              "testis" = "#E6AB02"),
                                   species = c("hsapiens" = "#66C2A5",
                                               "ptroglodytes" = "#FC8D62",
                                               "mmulatta" = "#8DA0CB",
                                               "mmusculus" = "#E78AC3",
                                               "rnorvegicus" = "#A6D854",
                                               "btaurus" = "#FFD92F",
                                               "sscrofa" = "#E5C494",
```

```
                                                "ggallus" = "#B3B3B3")),
                        #show_annotation_name = F,
                        annotation_name_side = "left")
myheatmap <- ComplexHeatmap::Heatmap(as.matrix(data_dist), cluster_rows = TRUE,
                                     clustering_method_rows = "centroid",
                                     cluster_columns = TRUE,
                                     clustering_method_columns = "centroid",
                                     top_annotation = ht, col = morecolors(50),
                                     show_row_names = FALSE,
                                     show_column_names = FALSE,
                                     name = "distance")
myheatmap
```



We do not see a clear clustering by tissue type in this case.

6.2 2 D plots

Next, we will explore the 2D projection plots in the common space.

```
# install packages
pkgs <- c("gridExtra")
require_install <- pkgs[!(pkgs %in% row.names(installed.packages()))]
if (length(require_install))
  install.packages(require_install)
suppressPackageStartupMessages({
  library(gridExtra)
})
```

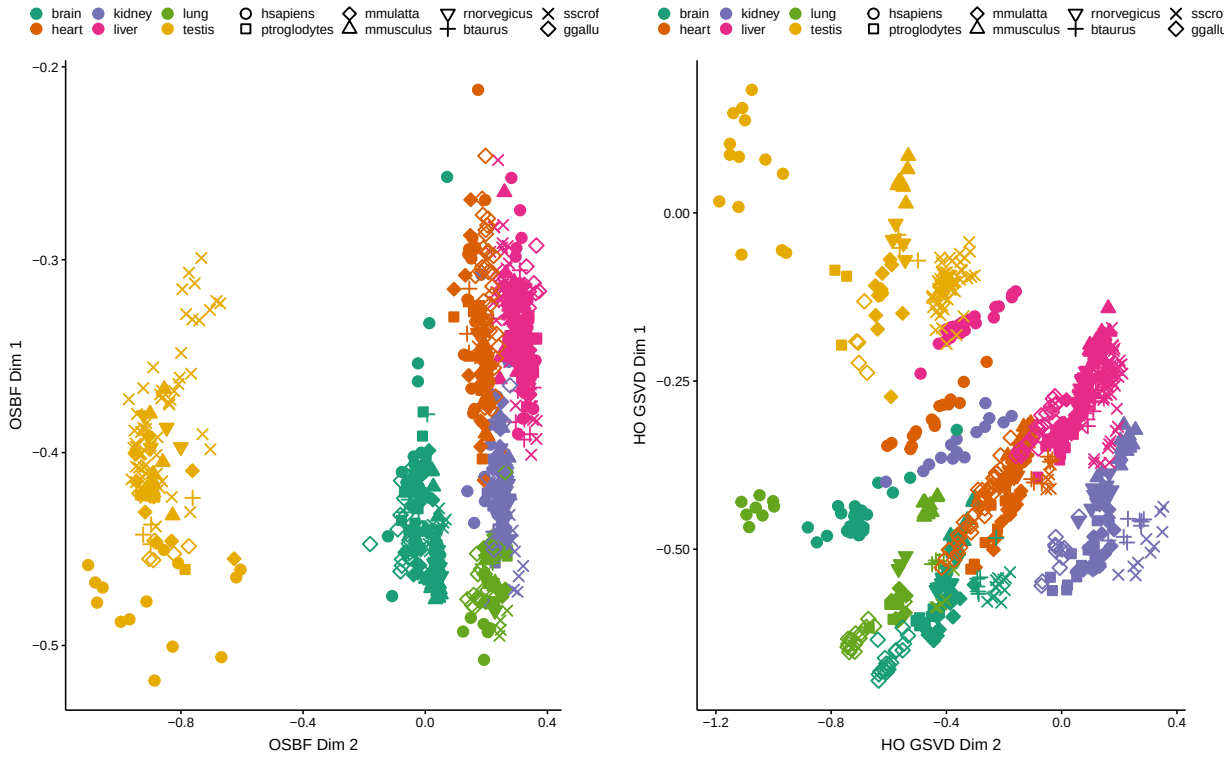
2 D plots for dimension 2-6

```
sel_colors <- RColorBrewer::brewer.pal(8, "Dark2")[seq_len(length(unique(df_proj$tissue)))]

p_list <- lapply(c(1, 3:6), function(k) {
  ggplot2::ggplot(df_proj, aes(x = df_proj[, 2], y = df_proj[, k], col = tissue,
                               shape = species, fill = tissue)) +
  xlab(paste("OSBF Dim", 2)) + ylab(paste("OSBF Dim", k)) +
  geom_point(size = 1.5) + scale_color_manual(values = sel_colors) +
  scale_shape_manual(values = c(21:25, 3:7)) +
  scale_fill_manual(values = sel_colors) +
  customTheme(base_size = 6) +
  theme(legend.title = element_blank())
})
h_list <- lapply(c(1, 3:6), function(k) {
  ggplot2::ggplot(ho_proj, aes(x = ho_proj[, 2], y = ho_proj[, k], col = tissue,
                               shape = species, fill = tissue)) +
  xlab(paste("HO GSVD Dim", 2)) + ylab(paste("HO GSVD Dim", k)) +
  geom_point(size = 1.5) + scale_color_manual(values = sel_colors) +
  scale_shape_manual(values = c(21:25, 3:7)) +
  scale_fill_manual(values = sel_colors) +
  customTheme(base_size = 6) +
  theme(legend.title = element_blank())
})
```

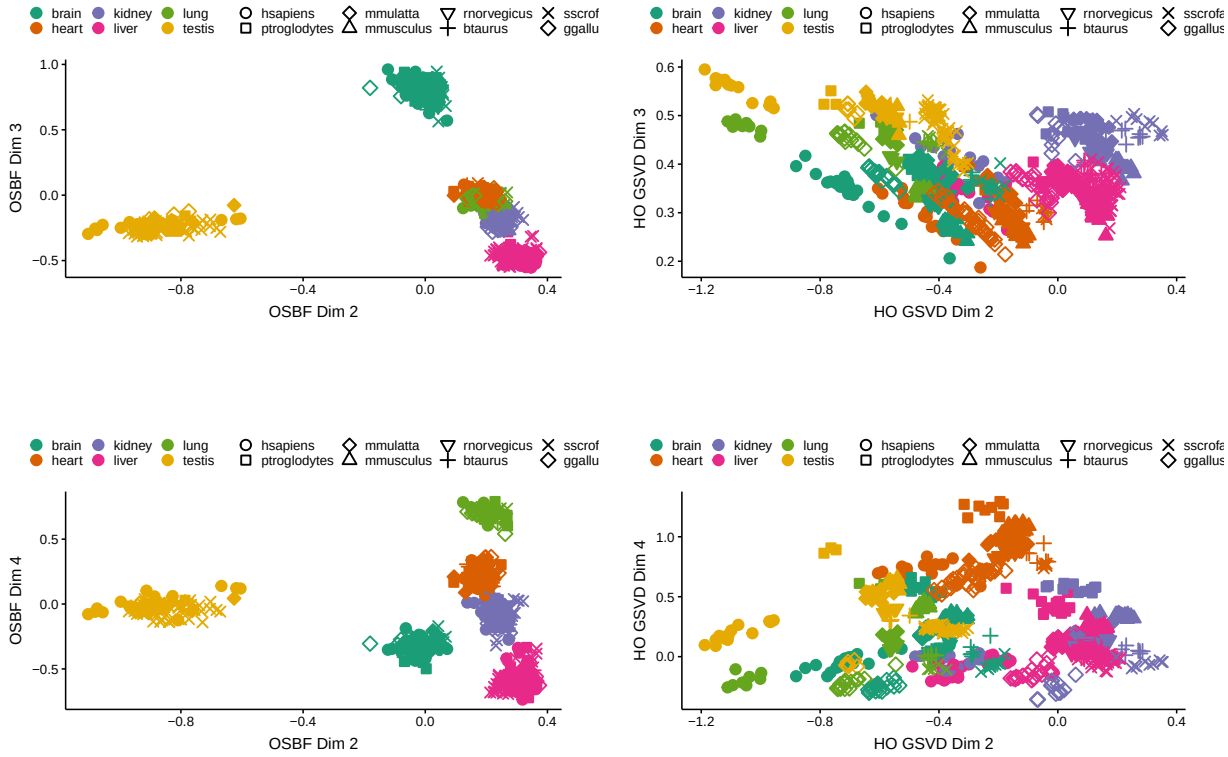
Dimensions 1 vs 2

```
grid.arrange(p_list[[1]], h_list[[1]], ncol = 2)
```



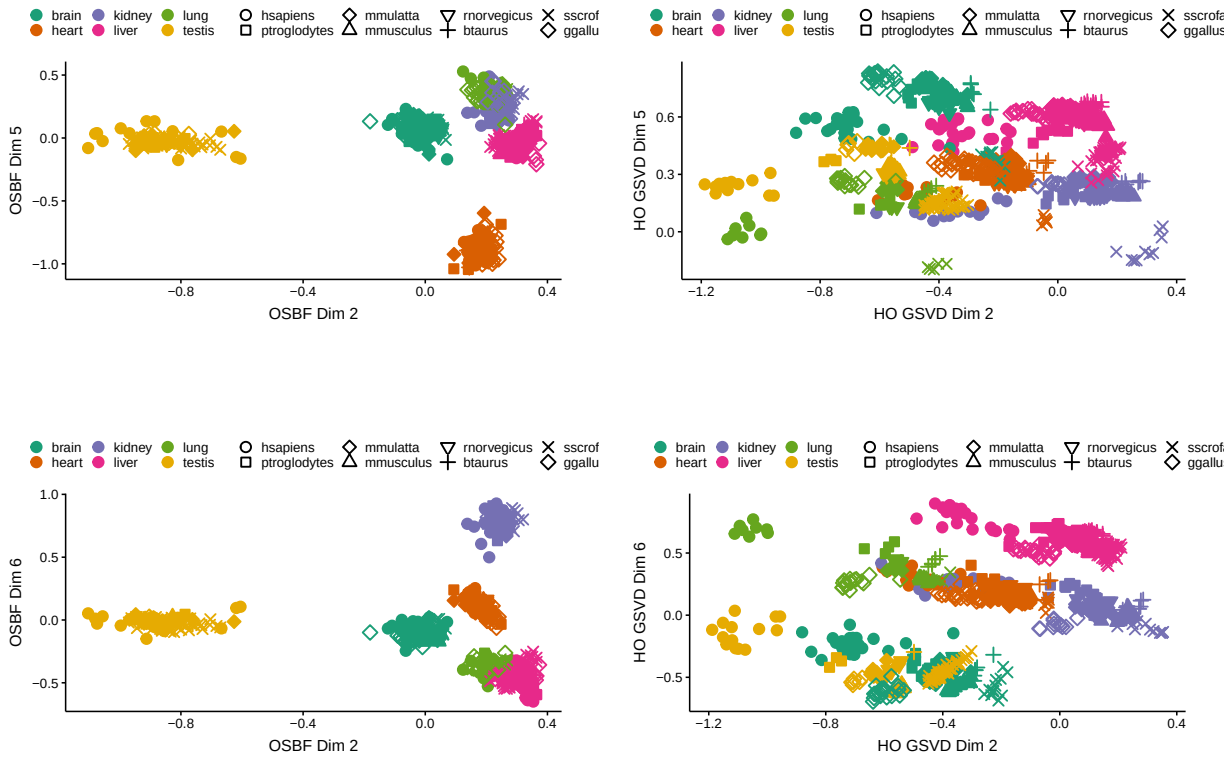
Dimensions 2 vs 3 and 2 vs 4

```
grid.arrange(p_list[[2]], h_list[[2]], p_list[[3]], h_list[[3]], ncol = 2)
```



Dimensions 2 vs 5 and 2 vs 6

```
grid.arrange(p_list[[4]], h_list[[4]], p_list[[5]], h_list[[5]], ncol = 2)
```



Using the same projection strategy as that of the OSBF, we do not see a clear separation of tissues along different dimensions of the HO GSVD shared space V.

7 Session info

```
sessionInfo()
#> R version 4.2.0 (2022-04-22)
#> Platform: x86_64-pc-linux-gnu (64-bit)
#> Running under: Ubuntu 20.04.4 LTS
#>
#> Matrix products: default
#> BLAS:   /usr/lib/x86_64-linux-gnu/blas/libblas.so.3.9.0
#> LAPACK: /usr/lib/x86_64-linux-gnu/lapack/liblapack.so.3.9.0
#>
#> locale:
#>  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
#>  [3] LC_TIME=en_US.UTF-8        LC_COLLATE=en_US.UTF-8
#>  [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
#>  [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
#>  [9] LC_ADDRESS=C               LC_TELEPHONE=C
#> [11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C
#>
#> attached base packages:
#> [1] grid      stats     graphics  grDevices utils     datasets  methods
#> [8] base
#>
#> other attached packages:
#>  [1] gridExtra_2.3          goseq_1.48.0           geneLenDataBase_1.32.0
#>  [4] BiasedUrn_1.07         ggplot2_3.3.6          ggthemes_4.2.4
#>  [7] RColorBrewer_1.1-3     ComplexHeatmap_2.12.0  matrixStats_0.62.0
#> [10] dplyr_1.0.9            data.table_1.14.2      SBF_1.0.0.0
#>
#> loaded via a namespace (and not attached):
#>   [1] colorspace_2.0-3            rjson_0.2.21
#>   [3] ellipsis_0.3.2              rprojroot_2.0.3
#>   [5] circlize_0.4.15             XVector_0.36.0
#>   [7] GenomicRanges_1.48.0        GlobalOptions_0.1.2
#>   [9] fs_1.5.2                    clue_0.3-61
#>  [11] rstudioapi_0.13             farver_2.1.0
#>  [13] remotes_2.4.2               bit64_4.0.5
#>  [15] AnnotationDbi_1.58.0        fansi_1.0.3
#>  [17] xml2_1.3.3                  splines_4.2.0
#>  [19] codetools_0.2-18            doParallel_1.0.17
#>  [21] cachem_1.0.6                knitr_1.39
#>  [23] pkgload_1.2.4               Rsamtools_2.12.0
#>  [25] GO.db_3.15.0                dbplyr_2.1.1
#>  [27] cluster_2.1.3               png_0.1-7
#>  [29] compiler_4.2.0              httr_1.4.3
#>  [31] assertthat_0.2.1            Matrix_1.4-1
#>  [33] fastmap_1.1.0               cli_3.3.0
#>  [35] htmltools_0.5.2             prettyunits_1.1.1
#>  [37] tools_4.2.0                 gtable_0.3.0
#>  [39] glue_1.6.2                  GenomeInfoDbData_1.2.8
```

```
#>  [41] rappdirs_0.3.3              tinytex_0.39
#>  [43] Rcpp_1.0.8.3                Biobase_2.56.0
#>  [45] vctrs_0.4.1                 Biostrings_2.64.0
#>  [47] nlme_3.1-157                rtracklayer_1.56.0
#>  [49] iterators_1.0.14            xfun_0.31
#>  [51] stringr_1.4.0               ps_1.7.0
#>  [53] brio_1.1.3                  testthat_3.1.4
#>  [55] lifecycle_1.0.1             restfulr_0.0.13
#>  [57] devtools_2.4.3              XML_3.99-0.9
#>  [59] zlibbioc_1.42.0             scales_1.2.0
#>  [61] hms_1.1.1                   MatrixGenerics_1.8.0
#>  [63] parallel_4.2.0              SummarizedExperiment_1.26.1
#>  [65] curl_4.3.2                  yaml_2.3.5
#>  [67] memoise_2.0.1               biomaRt_2.52.0
#>  [69] stringi_1.7.6               RSQLite_2.2.14
#>  [71] highr_0.9                   S4Vectors_0.34.0
#>  [73] BiocIO_1.6.0                desc_1.4.1
#>  [75] foreach_1.5.2               filelock_1.0.2
#>  [77] GenomicFeatures_1.48.1      BiocGenerics_0.42.0
#>  [79] pkgbuild_1.3.1              BiocParallel_1.30.2
#>  [81] shape_1.4.6                 GenomeInfoDb_1.32.2
#>  [83] rlang_1.0.2                 pkgconfig_2.0.3
#>  [85] bitops_1.0-7                evaluate_0.15
#>  [87] lattice_0.20-45             purrr_0.3.4
#>  [89] GenomicAlignments_1.32.0    labeling_0.4.2
#>  [91] bit_4.0.4                   processx_3.5.3
#>  [93] tidyselect_1.1.2            magrittr_2.0.3
#>  [95] R6_2.5.1                    IRanges_2.30.0
#>  [97] generics_0.1.2              DelayedArray_0.22.0
#>  [99] DBI_1.1.2                   mgcv_1.8-40
#> [101] pillar_1.7.0                withr_2.5.0
#> [103] KEGGREST_1.36.0             RCurl_1.98-1.6
#> [105] tibble_3.1.7                crayon_1.5.1
#> [107] utf8_1.2.2                  BiocFileCache_2.4.0
#> [109] rmarkdown_2.14              GetoptLong_1.0.5
#> [111] progress_1.2.2              usethis_2.1.6
#> [113] blob_1.2.3                  callr_3.7.0
#> [115] digest_0.6.29               stats4_4.2.0
#> [117] munsell_0.5.0               sessioninfo_1.2.2
```

References

Alter, Orly, Patrick O Brown, and David Botstein. 2000. "Singular Value Decomposition for Genome-Wide Expression Data Processing and Modeling." *Proceedings of the National Academy of Sciences* 97 (18): 10101–6.

———. 2003. "Generalized singular value decomposition for comparative analysis of genome-scale expression data sets of two different organisms." *Proceedings of the National Academy of Sciences* 100 (6): 3351–56.

Ponnapalli, Sri Priya, Michael A Saunders, Charles F Van Loan, and Orly Alter. 2011. "A higher-order generalized singular value decomposition for comparison of global mRNA expression from multiple organisms." *PloS One* 6 (12): e28072.

Tamayo, Pablo, Daniel Scanfeld, Benjamin L Ebert, Michael A Gillette, Charles WM Roberts, and Jill P Mesirov. 2007. "Metagene projection for cross-platform, cross-species characterization of global transcriptional states." *Proceedings of the National Academy of Sciences* 104 (14): 5959–64.

Van Loan, Charles F. 1976. "Generalizing the Singular Value Decomposition." *SIAM Journal on Numerical Analysis* 13 (1): 76–83.