Data Integration using Unsupervised Multiple Kernel Learning

Introduction

The TARA Oceans expedition facilitated the study of plankton communities by providing oceans metagenomic data combined with environmental measures to the scientific community. This study focuses on 139 prokaryotic-enriched samples collected from 68 stations and spread across three depth layers: the surface (SRF), the deep chlorophyll maximum (DCM) layer and the mesopelagic (MES) zones. Samples were located in 8 different oceans or seas: Indian Ocean (IO), Mediterranean Sea (MS), North Atlantic Ocean (NAO), North Pacific Ocean (NPO), Red Sea (RS), South Atlantic Ocean (SAO), South Pacific Ocean (SPO) and South Ocean (SO). In this vignette, we consider a subset of the original data. The data include 1% of the 35,650 prokaryotic OTUs and of the 39,246 bacterial genes that were randomly selected. **The aim is to integrate prokaryotic abundances and functional processes to environmental measure with an unsupervised method**.

To run the following code, install the latest versions of mixOmics and phyloseq.

Install and load the mixOmics and mixKernel libraries (note that mixKernel will soon be included in mixOmics!).

```r
# install.packages("mixKernel")
library(mixOmics)
library(mixKernel)
```

Loading TARA Ocean datasets

The (previously normalized) datasets are provided as matrices with matching sample names (rownames):

```r
data(TARAoceans)
# more details with: ?TARAOceans
# we check the dimension of the data:
lapply(list("phychem" = TARAoceans$phychem, "pro.phylo" = TARAoceans$pro.phylo,
            "pro.NOGs" = TARAoceans$pro.NOGs), dim)
```

```
## $phychem
## [1] 139  22
##
## $pro.phylo
## [1] 139 356
##
## $pro.NOGs
## [1] 139 638
```

Multiple kernel computation

Individual kernel computation

For each input dataset, a kernel is computed using the function `compute.kernel` with the choice of linear, phylogenic or abundance kernels. A user defined function can also be provided as input(argument `kernel.func`, see `?compute.kernel`).

The results are lists with a 'kernel' entry that stores the kernel matrix. The resulting kernels are symmetric matrices with a size equal to the number of observations (rows) in the input datasets.

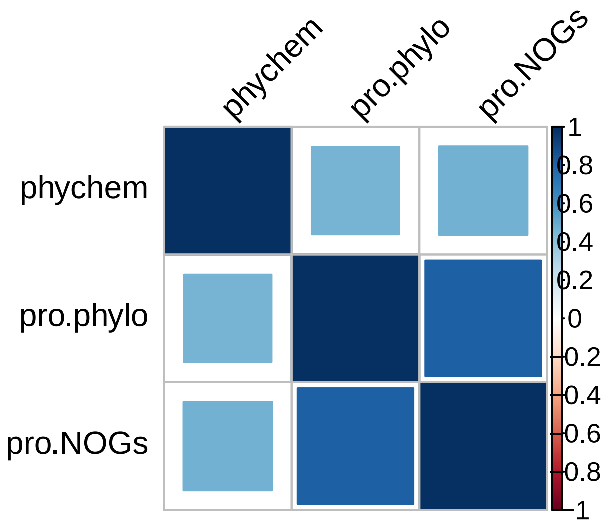
```
phychem.kernel <- compute.kernel(TARAoceans$phychem, kernel.func = "linear")
pro.phylo.kernel <- compute.kernel(TARAoceans$pro.phylo, kernel.func = "abundance")
pro.NOGs.kernel <- compute.kernel(TARAoceans$pro.NOGs, kernel.func = "abundance")

# check dimensions
dim(pro.NOGs.kernel$kernel)
```

```
## [1] 139 139
```

A general overview of the correlation structure between datasets is obtained as described in Mariette and Villa-Vialaneix (2017) and displayed using the function `cim.kernel`:

```
cim.kernel(phychem = phychem.kernel,
           pro.phylo = pro.phylo.kernel,
           pro.NOGs = pro.NOGs.kernel,
           method = "square")
```



The figure shows that `pro.phylo` and `pro.NOGs` is the most correlated pair of kernels. This result is expected as both kernels provide a summary of prokaryotic communities.

Combined kernel computation

The function `combine.kernels` implements 3 different methods for combining kernels: STATIS-UMKL, sparse-UMKL and full-UMKL (see more details in Mariette and Villa-Vialaneix, 2017). It returns a meta-kernel that can be used as an input for the function `kernel.pca` (kernel PCA). The three methods bring complementary information and must be chosen according to the research question.

The `STATIS-UMKL` approach gives an overview on the common information between the different datasets. The `full-UMKL` computes a kernel that minimizes the distortion between all input kernels. `sparse-UMKL` is a sparse variant of `full-UMKL` but also selects the most relevant kernels.

```
meta.kernel <- combine.kernels(phychem = phychem.kernel,
                               pro.phylo = pro.phylo.kernel,
                               pro.NOGs = pro.NOGs.kernel,
                               method = "full-UMKL")
```

Exploratory analysis: Kernel Principal Component Analysis (KPCA)

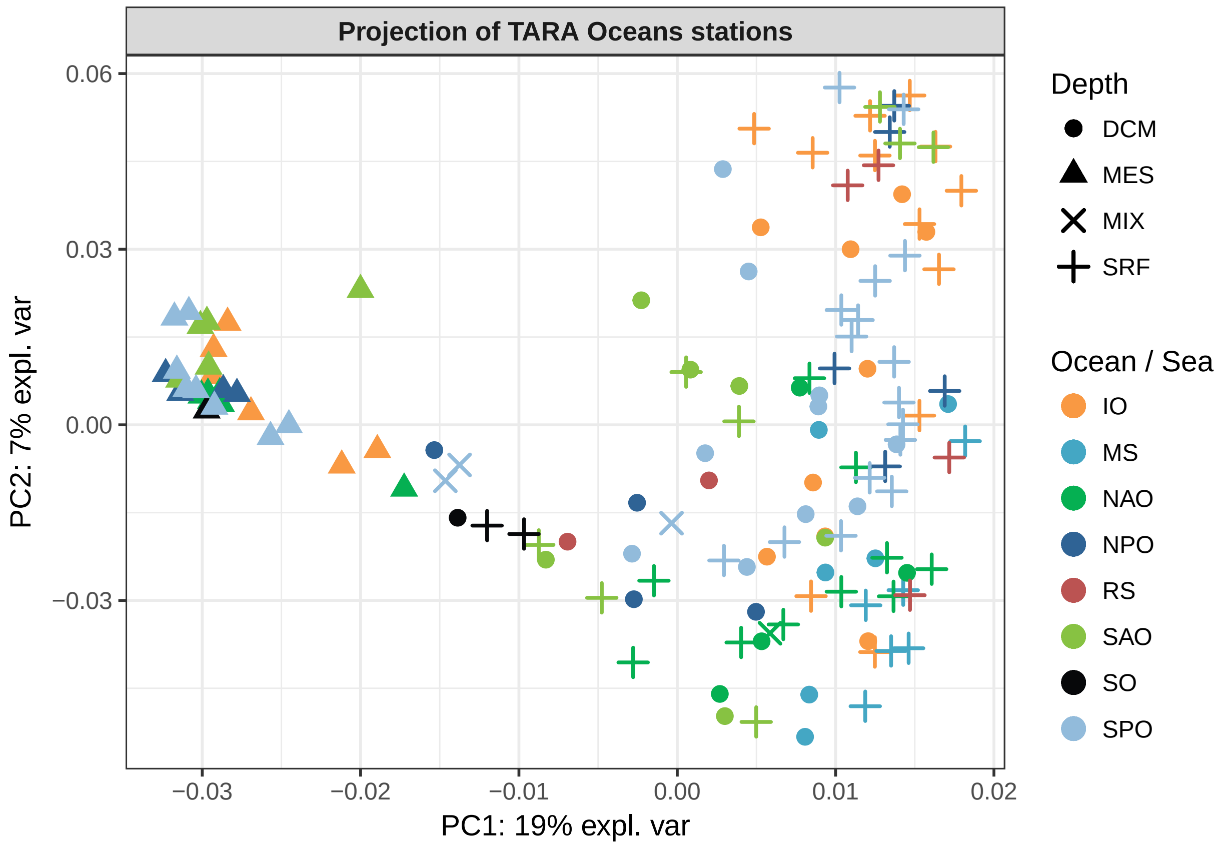
Perform KPCA

A kernel PCA can be performed from the combined kernel with the function `kernel.pca`. The argument `ncomp` allows to choose how many components to be extracted from KPCA.

```
kernel.pca.result <- kernel.pca(meta.kernel, ncomp = 10)
```

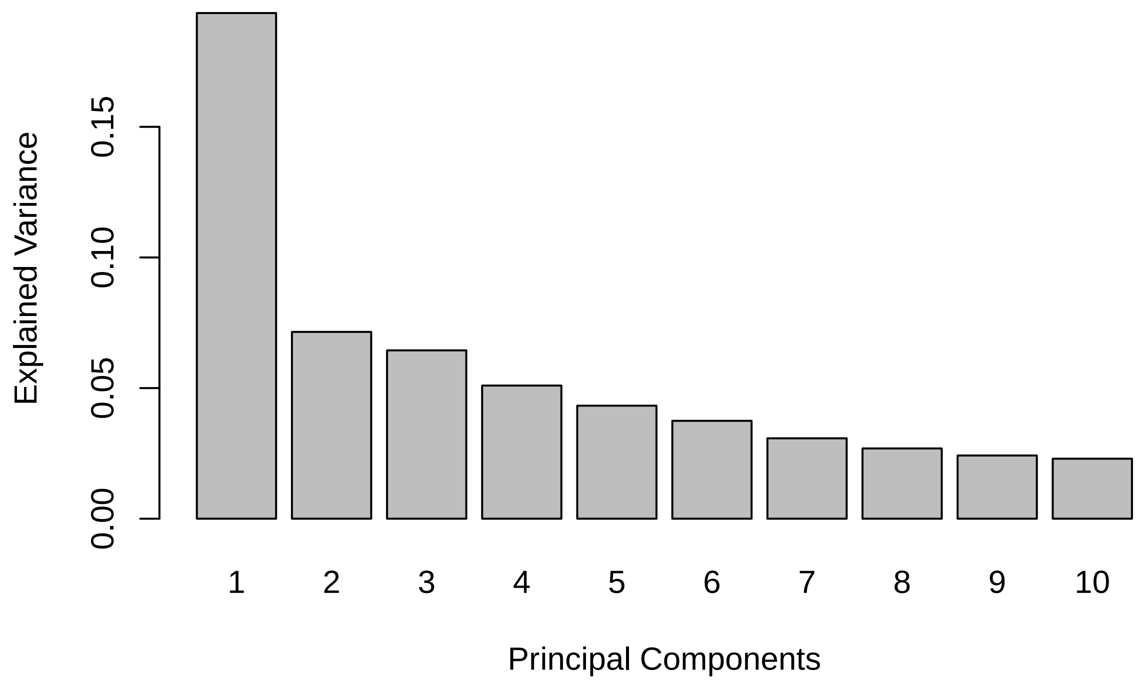
Sample plots using the `plotIndiv` function from `mixOmics`:

```
all.depths <- levels(factor(TARAoceans$sample$depth))
depth.pch <- c(20, 17, 4, 3)[match(TARAoceans$sample$depth, all.depths)]
plotIndiv(kernel.pca.result,
          comp = c(1, 2),
          ind.names = FALSE,
          legend = TRUE,
          group = as.vector(TARAoceans$sample$ocean),
          col.per.group = c("#f99943", "#44a7c4", "#05b052", "#2f6395",
                            "#bb5352", "#87c242", "#07080a", "#92bbdb"),
          pch = depth.pch,
          pch.levels = TARAoceans$sample$depth,
          legend.title = "Ocean / Sea",
          title = "Projection of TARA Oceans stations",
          size.title = 10,
          legend.title.pch = "Depth")
```



The explained variance supported by each axis of KPCA is displayed with the `plot` function, and can help choosing the number of components in KPCA.

```r
plot(kernel.pca.result)
```



The first axis summarises ~ 20% of the total variance.

Assessing important variables

Here we focus on the information summrised on the first component. Variables values are randomly permuted with the function `permute.kernel.pca`.

In the following example, physical variable are permuted at the variable level (kernel `phychem`), OTU abundances from `pro.phylo` kernel are permuted at the phylum level (OTU phyla are stored in the second column, named `Phylum`, of the taxonomy annotation provided in `TARAoceans` object in the entry `taxonomy`) and gene abundances from `pro.NOGs` are permuted at the GO level (GO are provided in the entry `GO` of the dataset):

```r
head(TARAoceans$taxonomy[ ,"Phylum"], 10)
```

```
##  [1] Actinobacteria   Proteobacteria   Proteobacteria   Gemmatimonadetes
##  [5] Actinobacteria   Actinobacteria   Proteobacteria   Proteobacteria
##  [9] Proteobacteria   Cyanobacteria
## 56 Levels: Acidobacteria Actinobacteria aquifer1 ... WCHB1-60
```

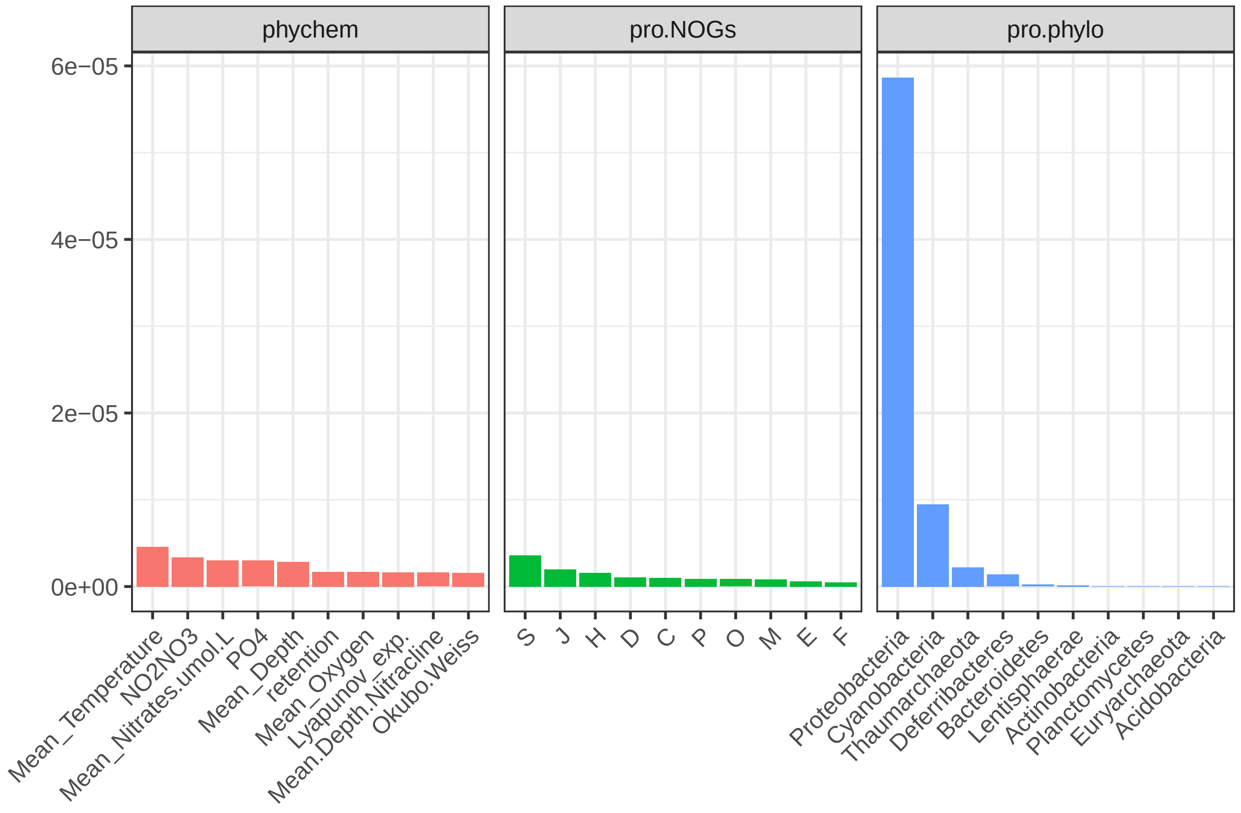
```r
head(TARAoceans$GO, 10)
```

```
##  [1] NA  NA  "K" NA  NA  "S" "S" "S" NA  "S"
```

```r
# here we set a seed for reproducible results with this tutorial
set.seed(17051753)
kernel.pca.result <- kernel.pca.permute(kernel.pca.result, ncomp = 1,
                                        phychem = colnames(TARAoceans$phychem),
                                        pro.phylo = TARAoceans$taxonomy[ ,"Phylum"],
                                        pro.NOGs = TARAoceans$GO)
```

Results are displayed with the function `plotVar.kernel.pca`. The argument `ndisplay` indicates the number of variables to display for each kernel:

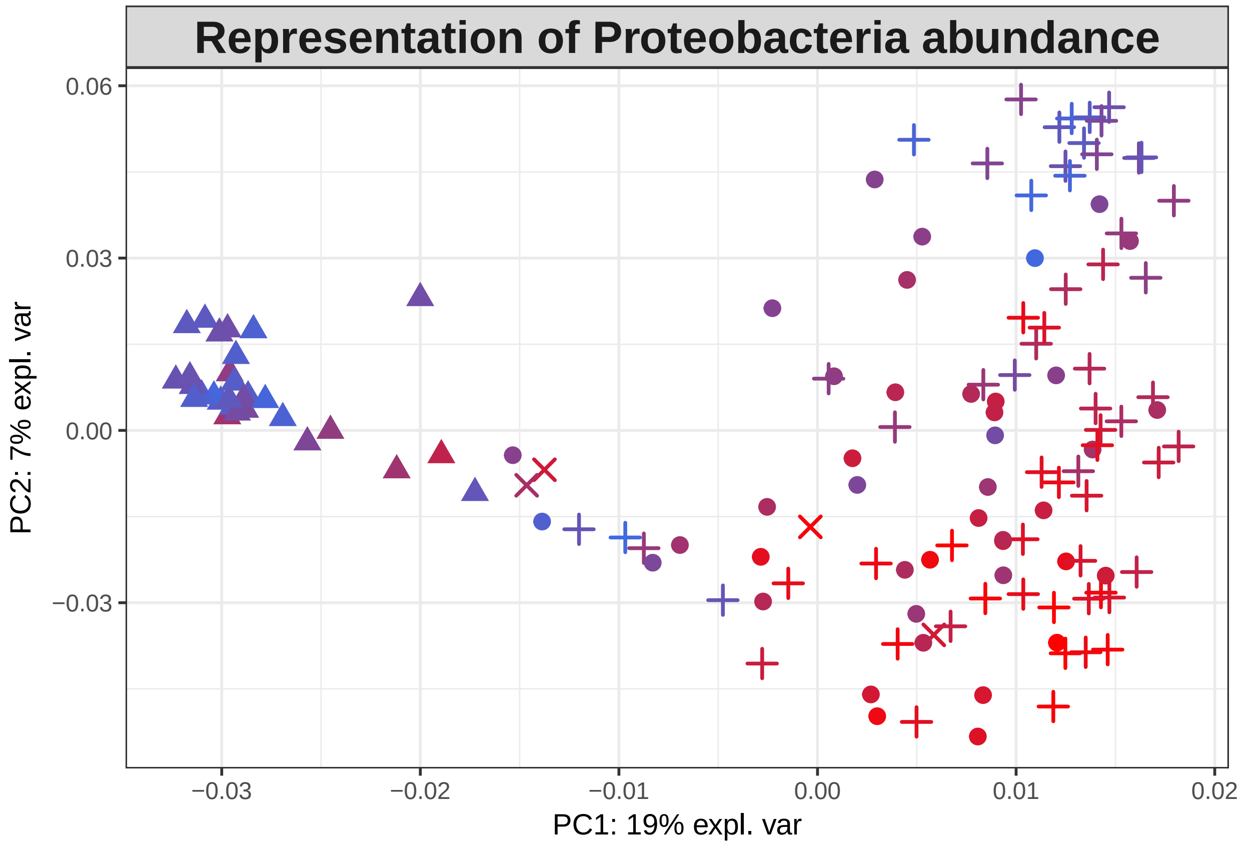
```
plotVar.kernel.pca(kernel.pca.result, ndisplay = 10, ncol = 3)
```



`Proteobacteria` is the most important variable for the `pro.phylo` kernel.

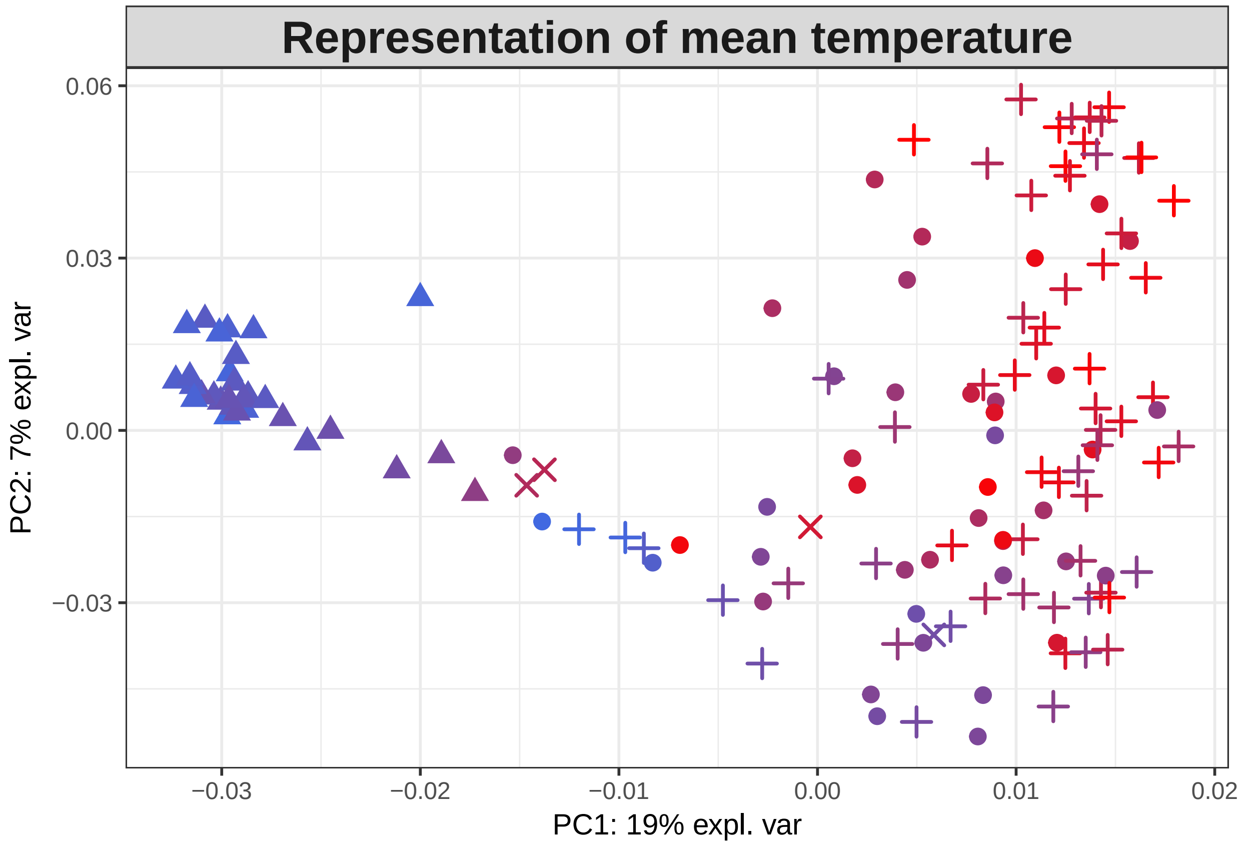
The relative abundance of `Proteobacteria` is then extracted in each of our 139 samples, and each sample is colored according to the value of this variable in the KPCA projection plot:

```
selected <- which(TARAoceans$taxonomy[ ,"Phylum"] == "Proteobacteria")
proteobacteria.per.sample <- apply(TARAoceans$pro.phylo[ ,selected], 1, sum) /
  apply(TARAoceans$pro.phylo, 1, sum)
colfunc <- colorRampPalette(c("royalblue", "red"))
col.proteo <- colfunc(length(proteobacteria.per.sample))
col.proteo <- col.proteo[rank(proteobacteria.per.sample, ties = "first")]
plotIndiv(kernel.pca.result,
          comp = c(1, 2),
          ind.names = FALSE,
          legend = FALSE,
          group = c(1:139),
          col = col.proteo,
          pch = depth.pch,
          pch.levels = TARAoceans$sample$depth,
          legend.title = "Ocean / Sea",
          title = "Representation of Proteobacteria abundance",
          legend.title.pch = "Depth")
```



Similarly, the temperature is the most important variable for the `phychem` kernel. The temperature values can be displayed on the kernel PCA projection as follows:

```r
col.temp <- colfunc(length(TARAoceans$phychem[ ,4]))
col.temp <- col.temp[rank(TARAoceans$phychem[ ,4], ties = "first")]
plotIndiv(kernel.pca.result,
          comp = c(1, 2),
          ind.names = FALSE,
          legend = FALSE,
          group = c(1:139),
          col = col.temp,
          pch = depth.pch,
          pch.levels = TARAoceans$sample$depth,
          legend.title = "Ocean / Sea",
          title = "Representation of mean temperature",
          legend.title.pch = "Depth")
```



References

1. Mariette, J. and Villa-Vialaneix, N. (2017) Integrating TARA Oceans datasets using unsupervised multiple kernel learning. bioRxiv 139287
2. Zhuang, J., Wang, J., Hoi, S., and Lan, X. (2011). Unsupervised multiple kernel clustering. Journal of Machine Learning Research: Workshop and Conference Proceedings, 20, 129–144.
3. Lavit, C., Escoufier, Y., Sabatier, R., and Traissac, P. (1994). The act (statis method). Computational Statistics & Data Analysis, 18(1), 97 – 119.

Session information

```
sessionInfo()
```

```
## R version 3.4.3 (2017-11-30)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 16.04.5 LTS
##
## Matrix products: default
## BLAS: /usr/local/lib/R/lib/libRblas.so
## LAPACK: /usr/local/lib/R/lib/libRlapack.so
##
## locale:
```

```
##  [1] LC_CTYPE=fr_FR.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=fr_FR.UTF-8        LC_COLLATE=fr_FR.UTF-8
##  [5] LC_MONETARY=fr_FR.UTF-8    LC_MESSAGES=fr_FR.UTF-8
##  [7] LC_PAPER=fr_FR.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=fr_FR.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] mixKernel_0.3   mixOmics_6.3.2  ggplot2_2.2.1   lattice_0.20-35
## [5] MASS_7.3-50     knitr_1.20
##
## loaded via a namespace (and not attached):
##  [1] Biobase_2.38.0        tidyr_0.8.1           splines_3.4.3
##  [4] jsonlite_1.5          foreach_1.4.4         ellipse_0.4.1
##  [7] shiny_1.1.0           assertthat_0.2.0      stats4_3.4.3
## [10] phyloseq_1.22.3       yaml_2.1.19           corrplot_0.84
## [13] pillar_1.2.3          backports_1.1.2       quadprog_1.5-5
## [16] glue_1.2.0            digest_0.6.15         manipulateWidget_0.9.0
## [19] RColorBrewer_1.1-2    promises_1.0.1        XVector_0.18.0
## [22] colorspace_1.3-2      psych_1.8.4           htmltools_0.3.6
## [25] httpuv_1.4.3          Matrix_1.2-14         plyr_1.8.4
## [28] pkgconfig_2.0.2       zlibbioc_1.24.0       purrr_0.2.5
## [31] xtable_1.8-2          corpcor_1.6.9         scales_0.5.0
## [34] RSpectra_0.13-1       later_0.7.2           tibble_1.4.2
## [37] mgcv_1.8-23           IRanges_2.12.0        BiocGenerics_0.24.0
## [40] lazyeval_0.2.1        mnormt_1.5-5          survival_2.42-3
## [43] magrittr_1.5          mime_0.5              evaluate_0.10.1
## [46] nlme_3.1-137          foreign_0.8-70        vegan_2.5-2
## [49] data.table_1.11.4     tools_3.4.3           matrixStats_0.53.1
## [52] stringr_1.3.1         S4Vectors_0.16.0      munsell_0.4.3
## [55] cluster_2.0.7-1       bindrcpp_0.2.2        Biostrings_2.46.0
## [58] ade4_1.7-11           compiler_3.4.3        rlang_0.2.1
## [61] rhdf5_2.22.0          grid_3.4.3            iterators_1.0.9
## [64] biomformat_1.6.0      htmlwidgets_1.2       crosstalk_1.0.0
## [67] igraph_1.2.2          miniUI_0.1.1.1        labeling_0.3
## [70] rmarkdown_1.9         multtest_2.34.0       gtable_0.2.0
## [73] codetools_0.2-15      rARPACK_0.11-0        reshape2_1.4.3
## [76] R6_2.2.2              gridExtra_2.3         dplyr_0.7.6
## [79] bindr_0.1.1           rprojroot_1.3-2       LDRTools_0.2-1
## [82] permute_0.9-4         ape_5.2               stringi_1.2.2
## [85] parallel_3.4.3        Rcpp_1.0.0            rgl_0.99.16
## [88] tidyselect_0.2.4
```