Analysis Baby Gut

Frontiers Final Version

AB

29 Mars 2019

Contents

Preliminary

```
library(tidyverse)
library(mixOmics)
walk(dir("../Rscripts/", pattern = ".R$", full.names = TRUE),source)
```

Data

Data comes from *(Development of the Human Infant Intestinal Microbiota, Palmer et al. 2007)*. In this paper, authors studied the gastrointestinal microbiome development of babies during the first year of life.

We focus here on the first 100 days because gut almost reached an "adult-like" composition and we also removed the baby data from the babies who received an antibiotic treatment during that period.

Our final design consists in an average of 21 time points for each of the 11 selected babies.

The figure below illustrates the sampling points per baby according to the delivery mode.

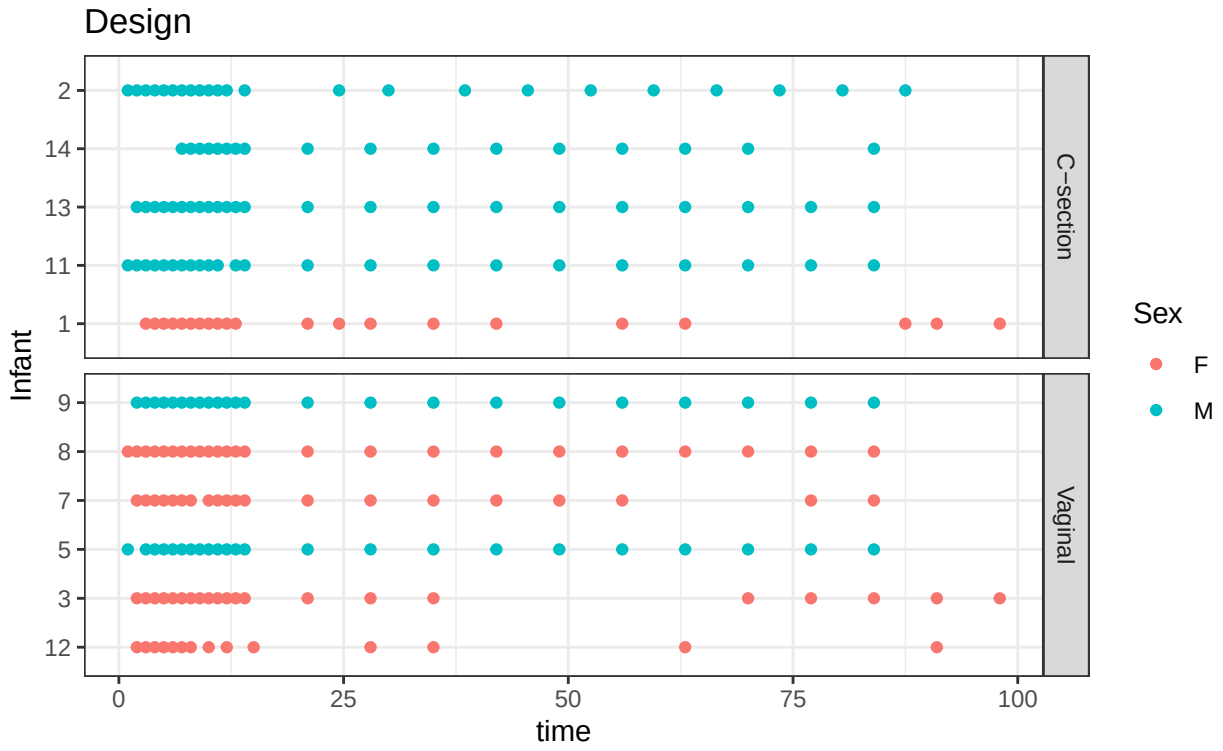


Figure 1: experimental design with sampling time points (in x axis) for each baby (y axis)

Data Transformation

Pre-processing

We perform standard pre-processing steps for microbiome data:

- Low Count Removal: to keep OTU of which abundance is greater than 1% at least in 1 time point.
- Total Sum Scaling: to calculate the relative abundance after the Low Count Removal filter
- Centered Log Ratio Transformation: to project the microbiome compositional data into an Euclidean space

```
# the norm_OTU performs the 3 steps
OTU_norm <- norm_OTU(OTU, AR = T)
```

The following figure shows the evolution of the OTUs as a function of time. Each block corresponds to a baby.

Figure 2: OTU evolution across time by baby

For the following analysis, we have separated the babies according to the mode of delivery.

Split data by mode of delivery

```
# C-section data : OTU_norm.C
dim(OTU_norm.C)
```

```
## [1] 107 107
```

```
# Vaginal data : OTU_norm.V
dim(OTU_norm.V)
```

```
## [1] 125 117
```

We now have 2 separate datasets. Microbiome data for babies born by **vaginal delivery** with 107 OTUs left after pre-processings filters. And the data of the babies born by **C-section** composed of 125 OTUs remaining.

Modelling

Spline smoothing

The modelling of each of the OTUs is performed by the Linear Mixed Model Splines framework which tests 4 different models for each OTU (package `lmms`).

As a reminder, the LMMS modeling step tests 4 different models for each OTUs. 0 = linear model, 1 = linear mixed effect model spline (LMMS) with defined basis, 2 = LMMS taking subject-specific random intercept, 3 = LMMS with subject specific intercept and slope.

Note: in the article, liner model is model 1, LMMS is model 2, LMMS taking subject-specific random intercept is model 3, and LMMS with subject specific intercept and slope is model 4.

```
# C-section
# numeric vector of time (number of days) per sample
time_lmms.C <- rownames(OTU_norm.C) %>% str_split("_") %>% map_chr(~.x[2]) %>% as.numeric

# data is a numeric matrix
# type ?lmms::lmmSpline for help
spline.MILK.C.pspline = lmms::lmmSpline(data = OTU_norm.C, time = time_lmms.C,
                                        sampleID = rownames(OTU_norm.C),
                                        basis = 'p-spline', keepModels = T,
                                        numCores = 2 )

# To visualize the number of OTUs modelled by each model
table(spline.MILK.C.pspline@modelsUsed)
```

```
##
##  0  1
## 78 29
```

For C-section data, we have 78 OTUs modelled with a straight line and 29 modelled with a linear mixed effect model spline and a *p-spline* basis.

```
# Vaginal
# numeric vector of time (number of days) per sample
time_lmms.V <- rownames(OTU_norm.V) %>% str_split("_") %>% map_chr(~.x[2]) %>% as.numeric

# data is a numeric matrix
# type ?lmms::lmmSpline for help
spline.MILK.V.pspline = lmms::lmmSpline(data = OTU_norm.V, time = time_lmms.V,
                                        sampleID = rownames(OTU_norm.V),
                                        basis = 'p-spline', keepModels = T,
                                        numCores = 2 )

# To visualize the number of OTUs modelled by each model
table(spline.MILK.V.pspline@modelsUsed)
```

```
##
##  0  1
## 95 22
```

For vaginal data, we have 95 OTUs modelled with a straight line and 22 modelled with a linear mixed effect model spline and a *p-spline* basis.

Profile filtering

Straight line modelling can occur when the inter-individual variation is too high. To remove the noisy profiles, we first use the Breusch-Pagan test, which tests the homo-sedasticity of the residues. We then add a filter on the mean squared error to reduce the dispersion of the residues around the line.

Note : please note that `raw_data` rownames must be of the form Sample_Time

```
# rownames of raw_data : sample_time
head(rownames(OTU_norm.C))
```

```
## [1] "1_10"   "1_11"   "1_12"   "1_13"   "1_21"   "1_24.5"
```

```
# C-section
# The next function, takes as arguments both the raw data and lmms object.
# It performs Brehsch-Pagan test, applies a filter on the MSE,
# and a list of OTUs to keep for the next part of the analysis
filter.spline.C.res <- wrapper.filter.splines(raw_data = OTU_norm.C,
                                              LMMSObject = spline.MILK.C.pspline)
# Then we filter the modelled data according to the list
index.filter.C <- which(rownames(spline.MILK.C.pspline@predSpline) %in%
                          filter.spline.C.res$to_keep)
spline.data.C <- as.data.frame(t(spline.MILK.C.pspline@predSpline[index.filter.C,]))
```

36 noisy profiles were removed from C-section data. In the figure below, the expression of OTUs is modelled as a function of time for babies born by C-section after filtering of the data.

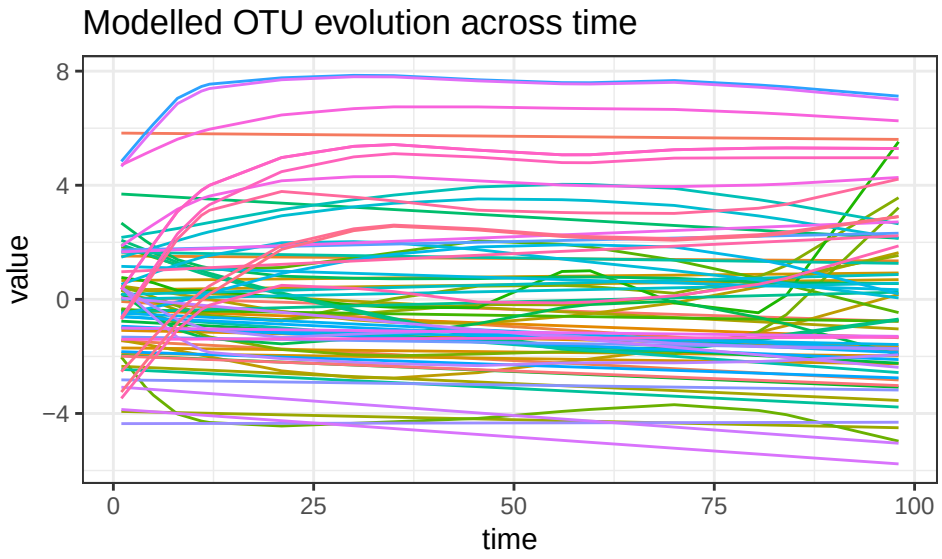


Figure 3: Modelled OTU evolution across time

Vaginal

The same filters have been applied to vaginal data, 27 noisy profiles were removed.

Clustering of time profile

PCA longitudinal clustering

From the modelled data, we use a PCA to cluster OTUs that have the same expression profile over time.

First we have to identify the number of component to select. In the PCA, to build the new components, each original variable will have a positive or negative (or zero) contribution to create these components.

We assign for each molecule to a cluster according to its maximum contribution to one of the components. Then, the number of clusters will be two times the number of components. Some clusters may be empty.

C-section

In the following graph, we have the evolution of the average silhouette coefficient as a function of the number of components added to the PCA. Since the average silhouette coefficient indicates the quality of the clustering, we try to maximize it or detect the drop point.

For the C-section data, 2 components (4 clusters) give the best clustering according to this coefficient.

```
# number of component to select
# this function returns the silhouette coefficient for each ncomp
res.ncomp <- wrapper.pca.ncomp(spline.data.C, ncomp = 8, scale = T, center = T)
```

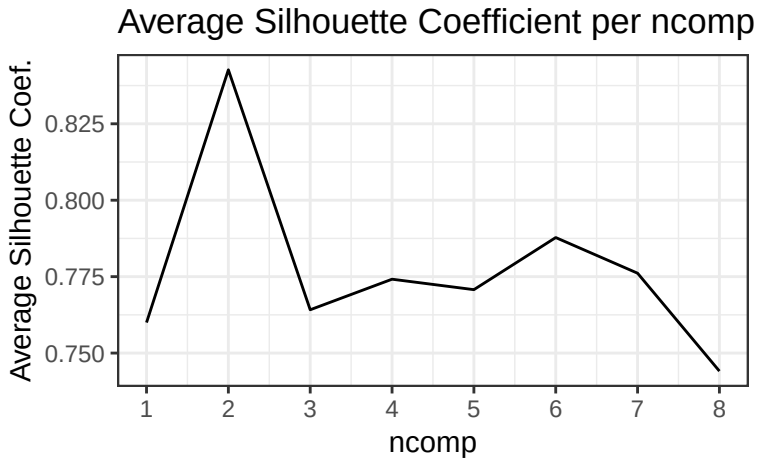


Figure 4: Average Silhouette Coefficient per number of component

```
# let's run mixOmics's pca
pca.res.C <- pca(spline.data.C, ncomp = 2, scale = T, center = T)
```

In the following graph, time points are represented as points placed according to their projection in the smaller subspace spanned by the components of the PCA. They allow to visualize the similarities (the points are grouped together) and the dissimilarities between the times.

```
plotIndiv(pca.res.C)
```

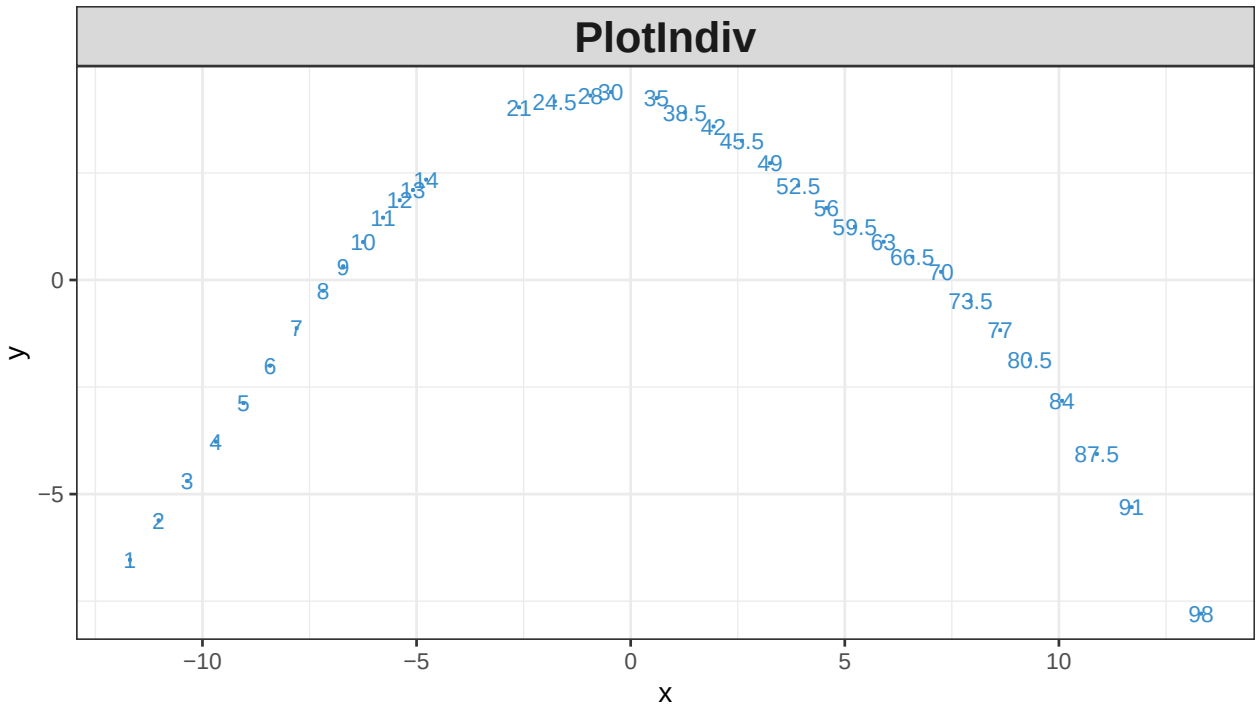


Figure 5: Individual (time points) graph

In the following graph, time points are represented as points placed according to their projection in the smaller subspace spanned by the components of the PCA. They allow to visualize the similarities (the points are grouped together) and the dissimilarities between the times.

The contribution of each OTU in the construction of the new components can be displayed on the circle of correlations plot. On this graph, the strongly correlated OTUs are projected in the same direction. We use this information to build trajectory clusters.

```
plotVar(pca.res.C, cex = 3)
```

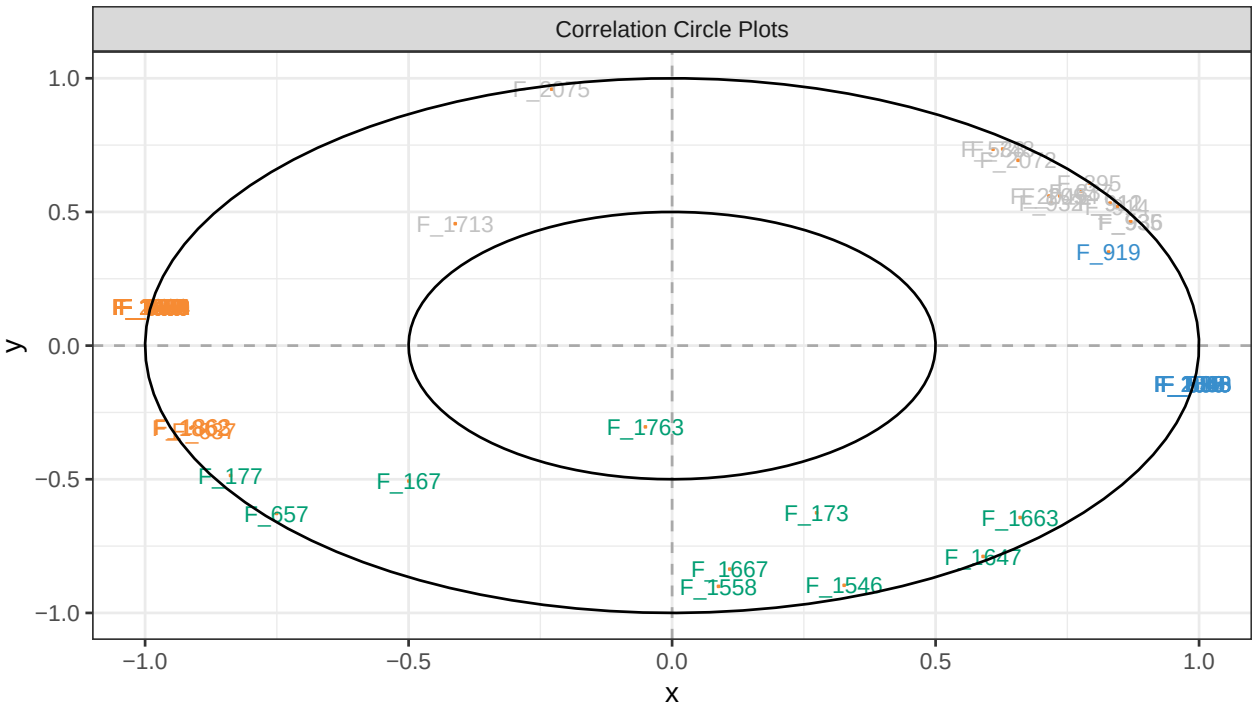


Figure 6: Variable (OTUs) graph

In the previous graph, we have OTUs that contribute positively to the first component (right), OTUs with a negative contribution on the first component (left), OTUs with a positive contribution on the second component (top) and OTUs with a negative contribution on the second component (bottom).

After assigning each OTU to a cluster, we can display the trajectories by cluster. In the following figure, each curve represents the modelled expression of each OTU over time. Each block represents a cluster. We can observe clusters according to their contribution (in row) per component (in column). The expression is centered and scaled.

```
# this function takes as argument the result of the PCA
# and plot the trajectories per cluster
pca.plot(pca.res.C, title = "C-section PCA Clusters, scale = T")
```

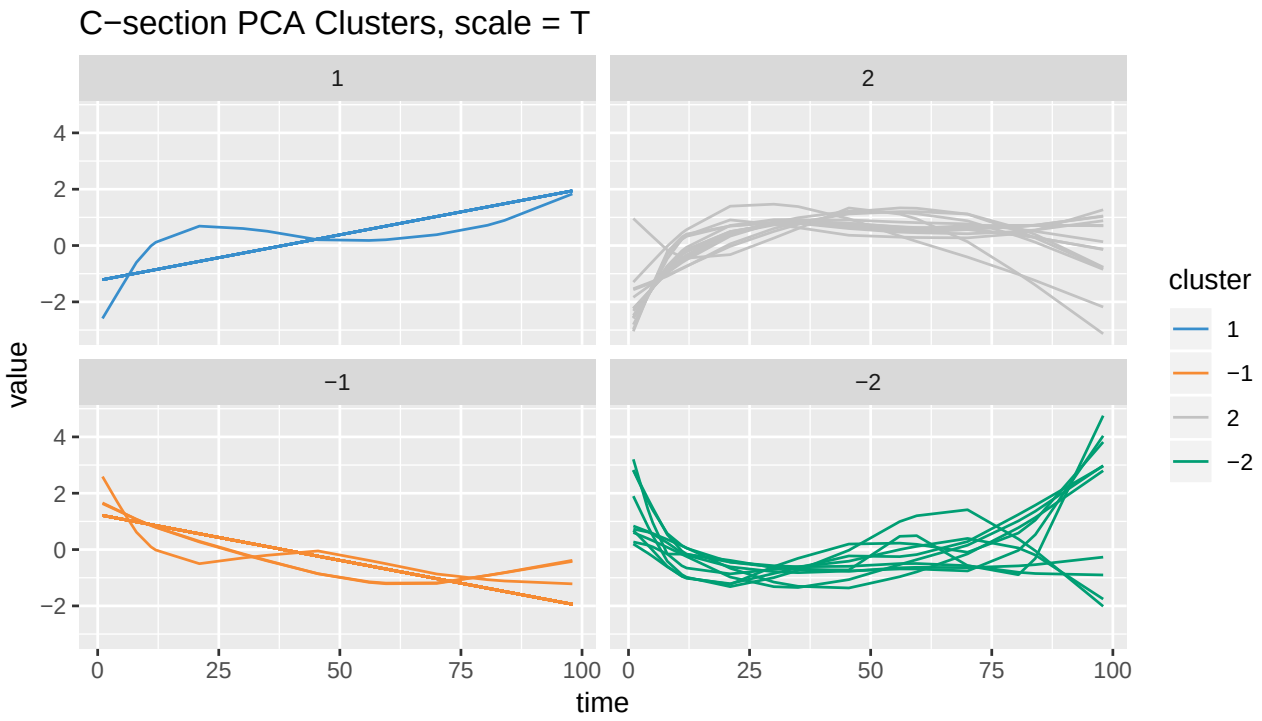


Figure 7: OTU modelled across time by cluster for C-section data

We then have 4 clusters. The cluster labelled `"1"` corresponds to the OTUs having a positive contribution on component 1, the cluster labelled `"-1"` with a negative contribution on component 1. The same applies to clusters labelled `"2"` and `"-2"` with a positive and negative contribution respectively on component 2.

To know which OTU belongs to which cluster, we can apply the following code. The result is an table where each line corresponds to an OTU and its associated cluster.

```
# see OTUs per clusters # only first 6 with head function
head(pca.get_cluster(pca.res.C))
```

```
## # A tibble: 6 x 2
## # Groups:   molecule [6]
##   molecule cluster
##   <chr>      <dbl>
## 1 F_1003        -1
## 2 F_119         -1
## 3 F_1285        -1
## 4 F_1334        -1
## 5 F_1451        -1
## 6 F_1452        -1
```

We can display the silhouette graph. The silhouette coefficient (x-axis) for each OTU is represented by a horizontal line. OTUs belonging to the same cluster appear in the same color. Clusters are labelled here according to their contribution to the component. The average silhouette coefficient is represented by a black vertical line.

```
# silhouette coefficient for this clustering
# this function takes as argument modelled data and the arguments of the pca
# such as scale, center, ?pca for help
wrapper.silhouette.pca(spline.data.C, ncomp = 2, scale = T, center=T, plot.t = TRUE)
```

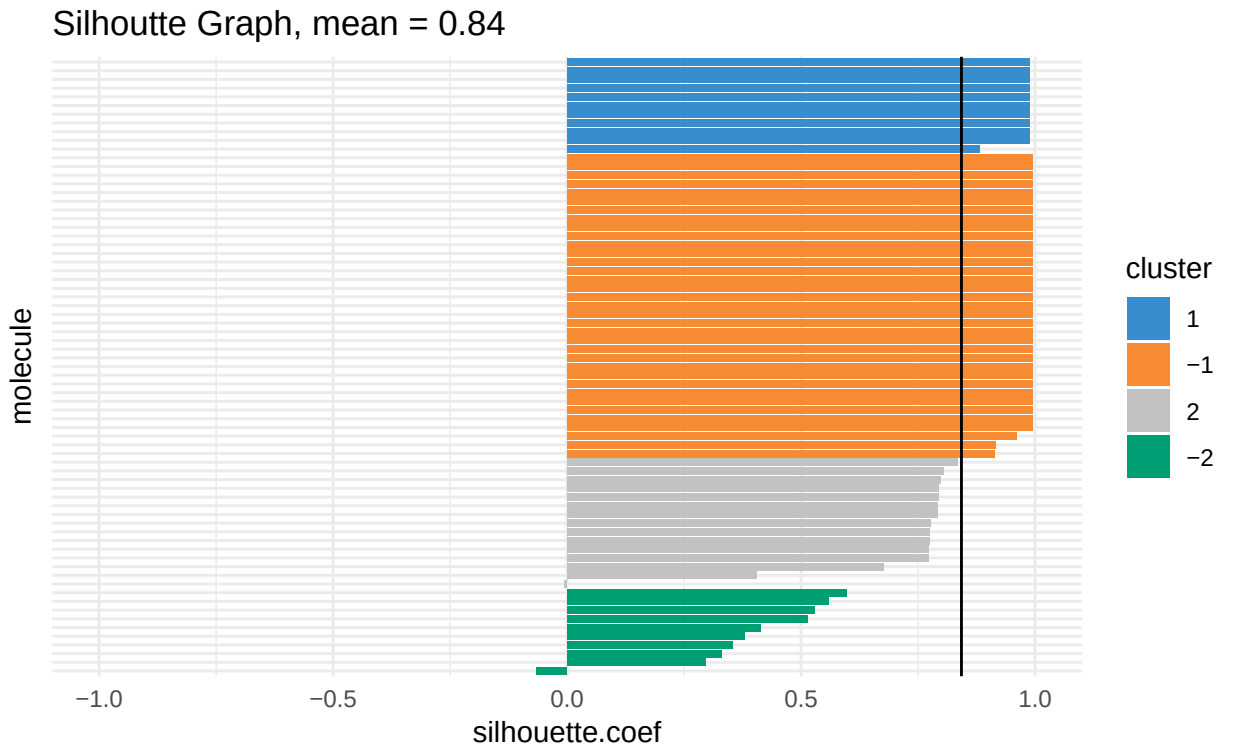


Figure 8: Silhouette Graph

```
## [1] 0.8426561
```

Vaginal

For vaginal data, we applied the same method to identify the number of components and thus the number of clusters. The average silhouette coefficient for 4 clusters is 0.87

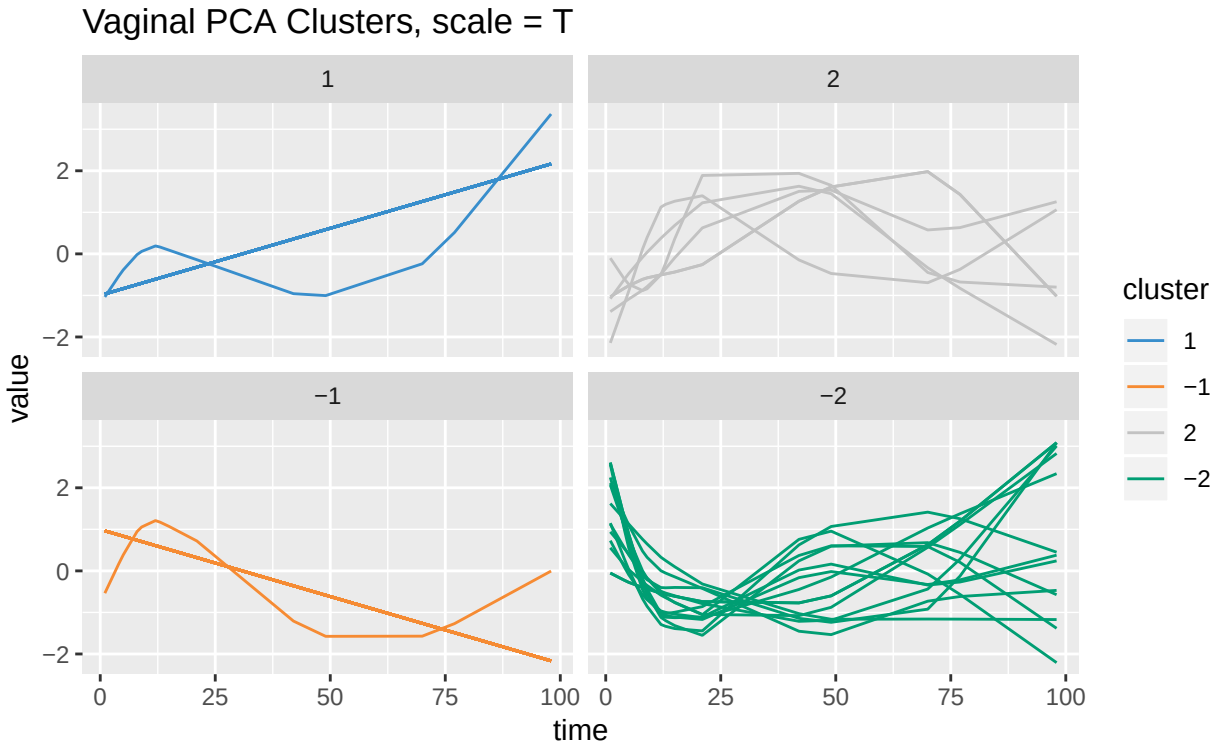
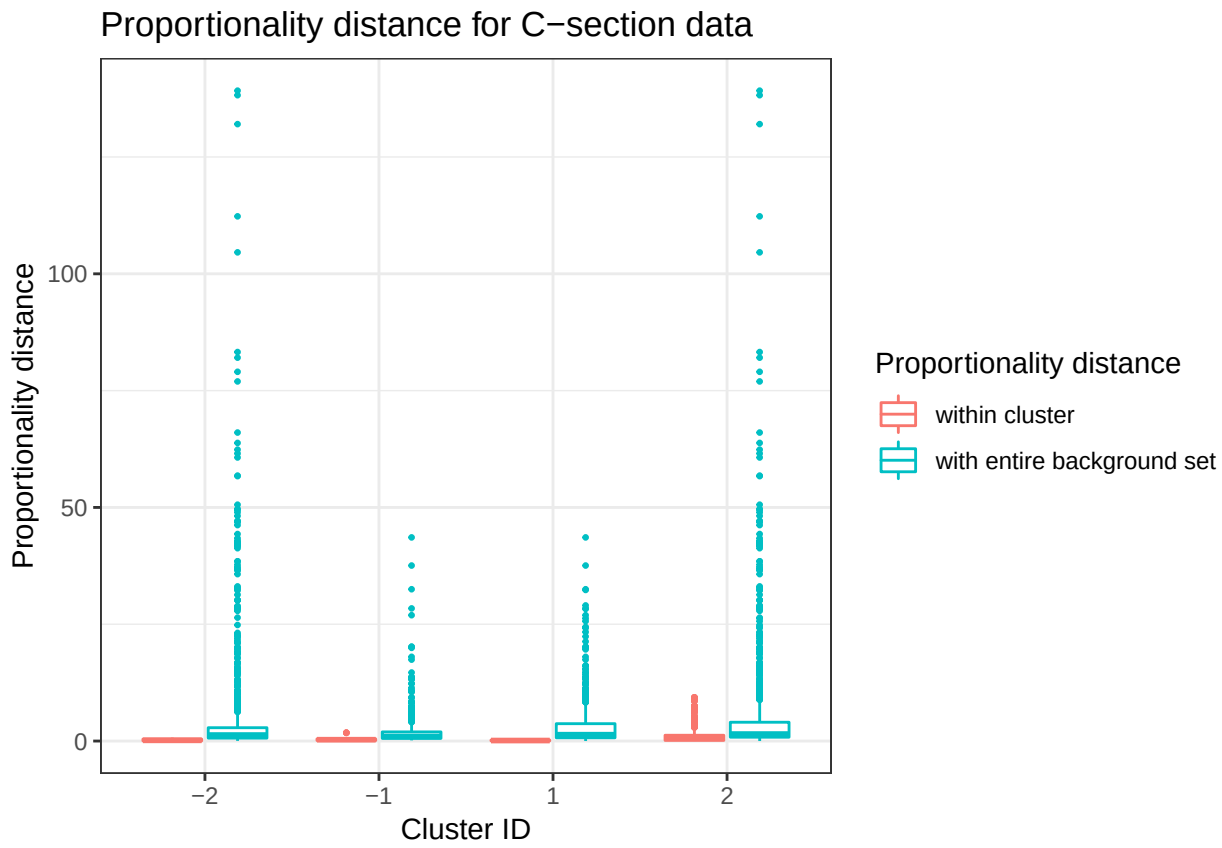


Figure 9: OTU modelled across time by cluster for vaginal data

Measure of association for compositional data

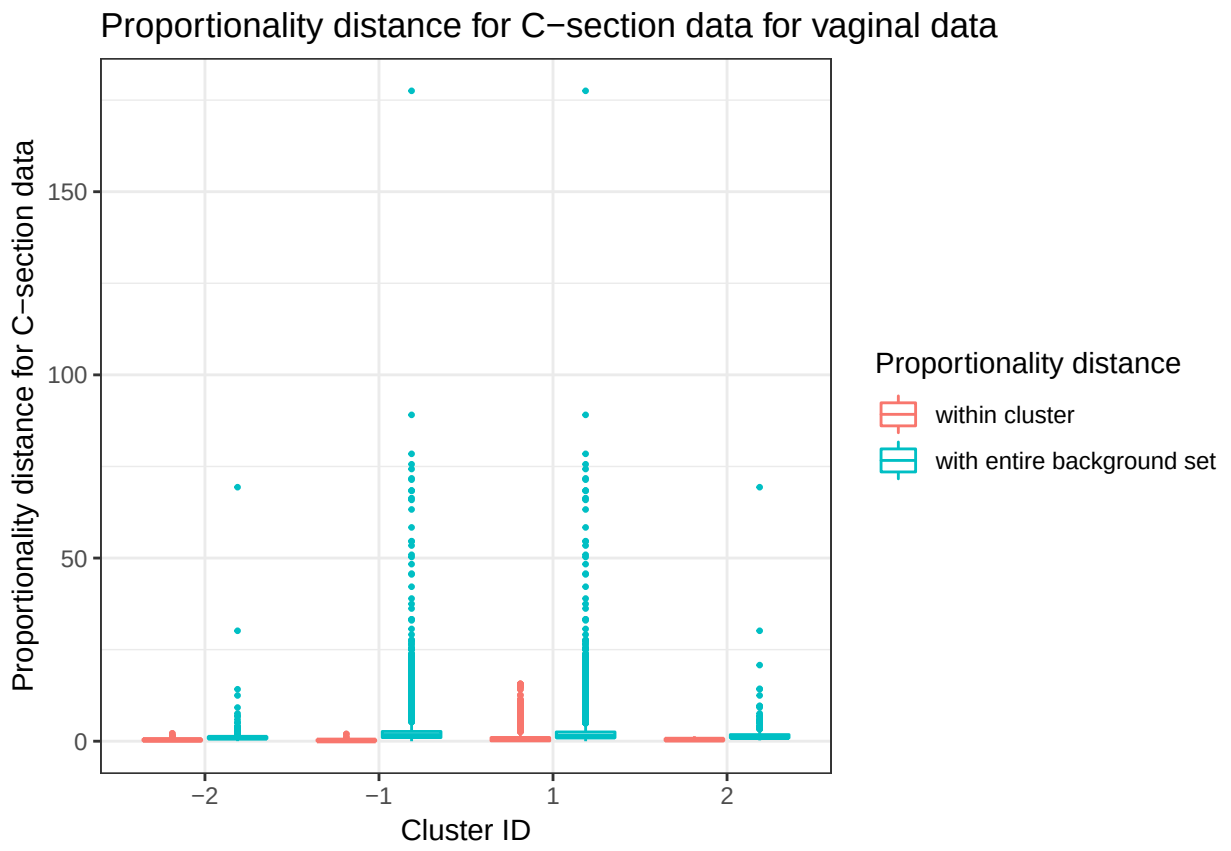
Interpretation based on correlations between profiles must be made with caution as it is highly likely to be spurious. Proportional distances has been proposed as an alternative to measure association.

C-section



cluster	median inside	median outside	Wilcoxon test Pval
2	0.67	1.50	0
-2	0.16	1.28	0
-1	0.29	0.96	0
1	0.11	1.36	0

Vaginal



cluster	median inside	median outside	Wilcoxon test Pval
-1	0.20	1.44	0
1	0.64	1.30	0
2	0.44	1.11	0
-2	0.35	0.95	0

Feature selection by cluster

The previous clustering used all OTUs. Sometimes we are interested in a cluster signature. We then use the sparse PCA to extract this key signature.

To find the right number of OTUs to keep per component and thus per cluster, we evaluate the silhouette for a list of selected molecules on each component. For this example, we tested 6 iterations on the first component from 14 to 29 in steps of 3 and 6 iterations on the second component from 9 to 15.

We do not recommend using a parameter that is too small since it will tend to pull the silhouette coefficient upwards and bias the interpretation regarding the number of OTUs to be selected.

We will then follow the evolution of the silhouette coefficient of each cluster (component and contribution). The main idea here is to detect a significant decrease in the evolution of the silhouette for each component. In other words, if we add 1 OTU, will the cluster be distorted?

C-section

```
# with tune.spca, we need to give a list of numeric vector of size ncomp
keepX = list(seq(14,29, 3), seq(9,15,1))
# this function takes as arguments modelled data, pca's arguments, and the list of keepX
res.tune.spca.C <- tune.spca(X = spline.data.C, ncomp = 2, keepX = keepX)
# get tuning plot from tuning result
gg <- tune.spca.choice.keepX(res.tune.spca.C, draw = T)
```

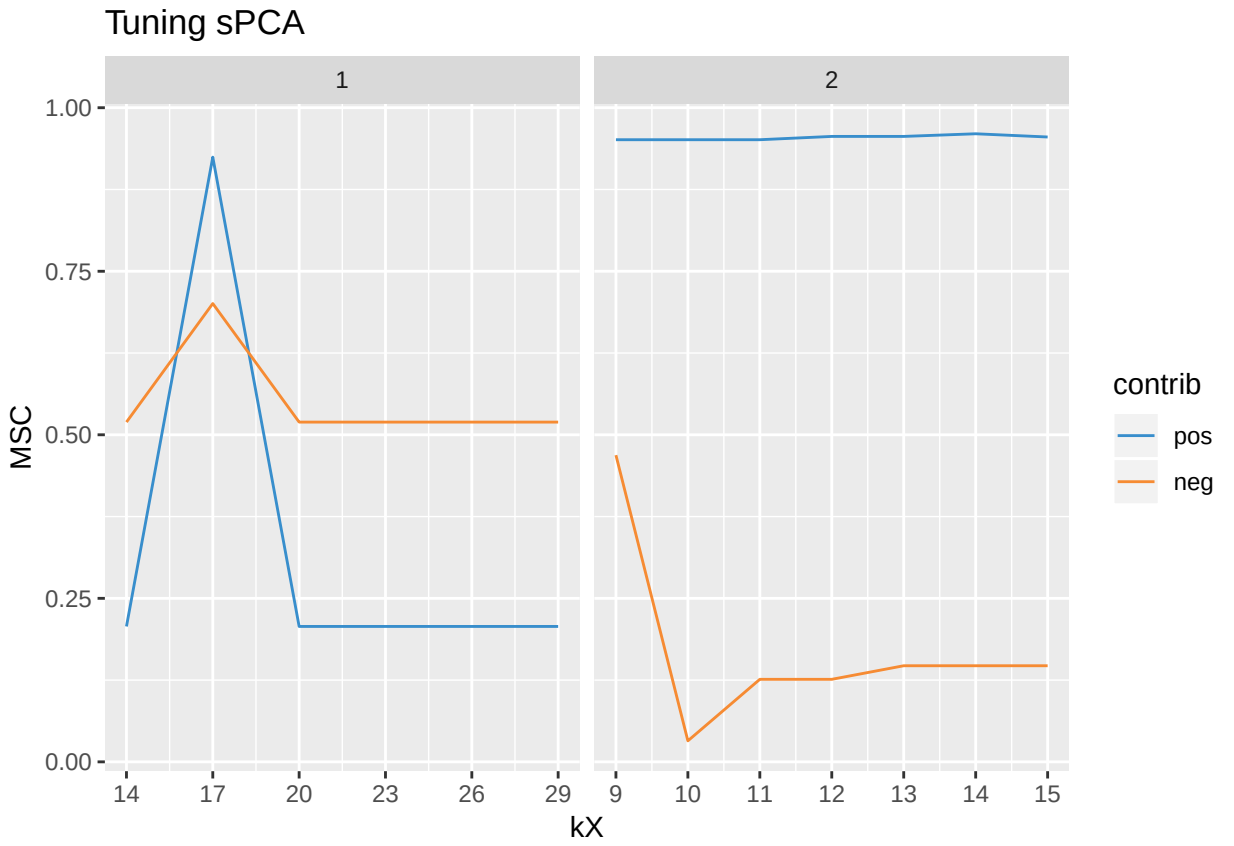


Figure 10: sPCA tuning plot

To detect the optimal number of OTUs to keep here, we can look at the graph above. On this graph, we represent the evolution of the silhouette coefficient by cluster and by component according to the number of OTUs selected.

Here, we will therefore choose 17 trajectories on the first component because the silhouette coefficient for both the positive and negative clusters is maximum and falls from this value. The trajectories will then be separated according to their contribution on component 1, positive or negative.

For the second component, we selected 9 OTUs since the positive cluster does not decrease but for the negative cluster, the silhouette coefficient is maximum at this value. We could have tested a smaller value for the second component but we wanted to select a minimum number of OTUs.

We run the sparse PCA again with these new parameters and we can link each OTU to its cluster with the following code.

```
# mixOmics's spca, ?spca for help
spca.res_f.C <- spca(spline.data.C, ncomp = 2, keepX = c(17,9))
```

```
head(pca.get_cluster(spca.res_f.C))
```

```
## # A tibble: 6 x 2
## # Groups:   molecule [6]
##   molecule cluster
##   <chr>      <dbl>
## 1 F_1003        -1
## 2 F_1285        -1
## 3 F_1334        -1
## 4 F_1563        -1
## 5 F_1632        -1
## 6 F_1714        -1
```

With the sparse PCA and the selection of the most representative trajectories by cluster, we have improved the average silhouette coefficient (0.95).

```
# wrapper to get average silhouette coefficiant for spca clustering
wrapper.silhouette.spca(spline.data.C, keepX = c(17,9),  ncomp = 2,
                            scale = T, center=T, plot.t = T)
```

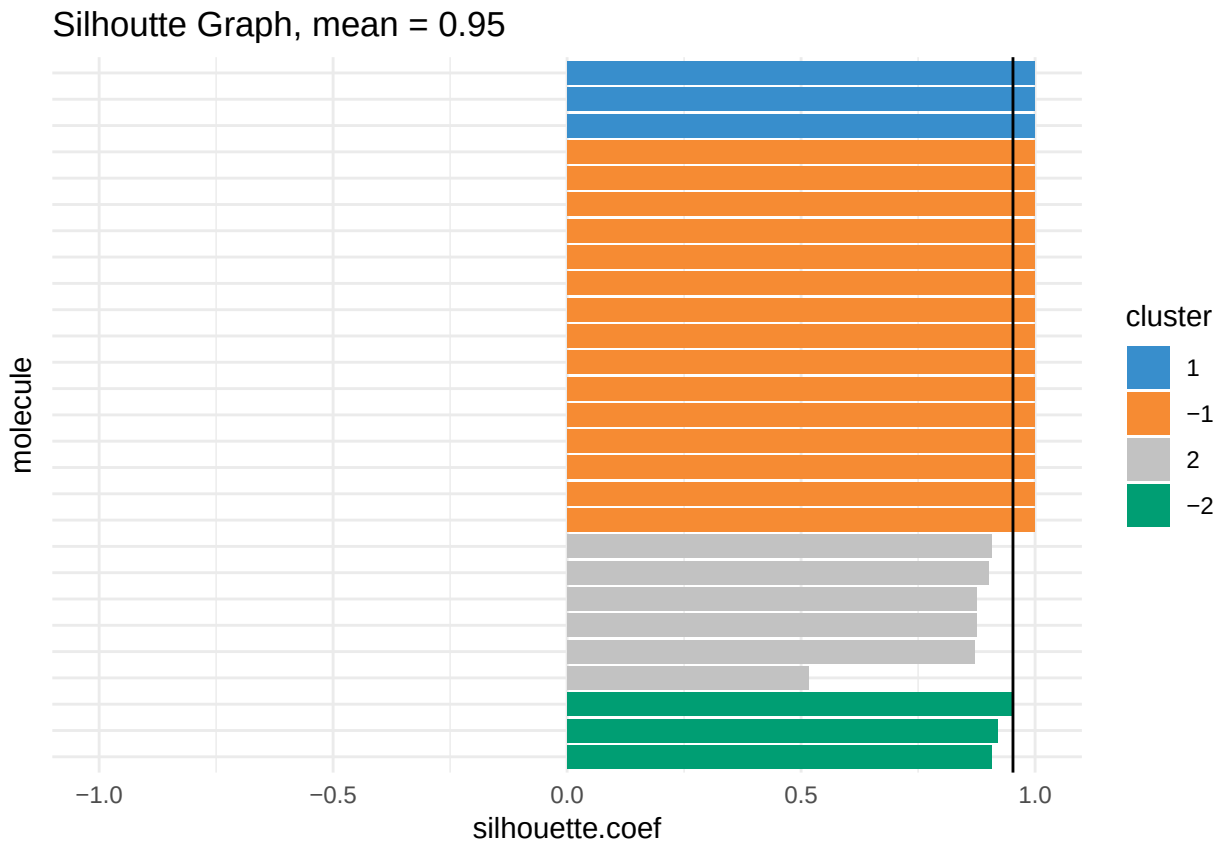


Figure 11: Silhouette Graph for C-section data clustering

```
## [1] 0.9527507
```

```
# plot trajectories per cluster
spca.plot(spca.res_f.C, title = "C-section sparse PCA Clusters")
```

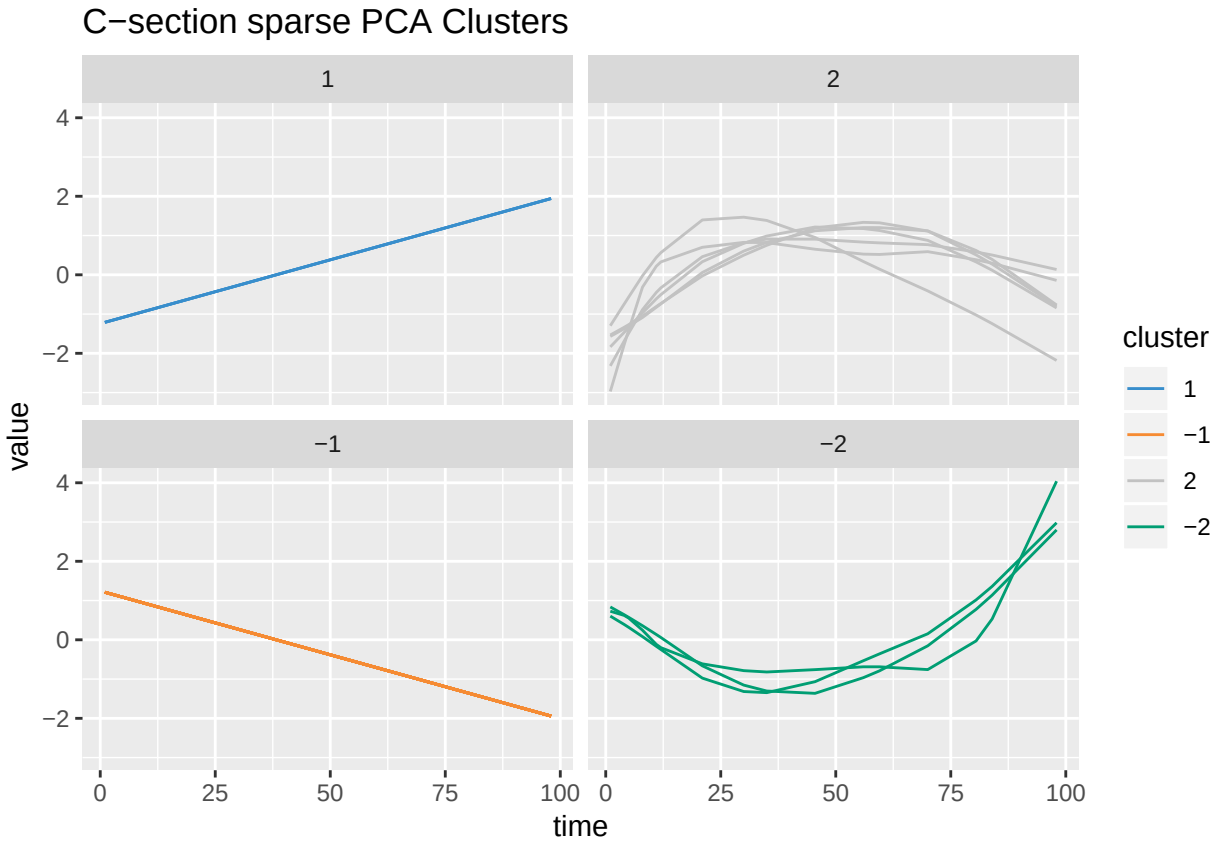
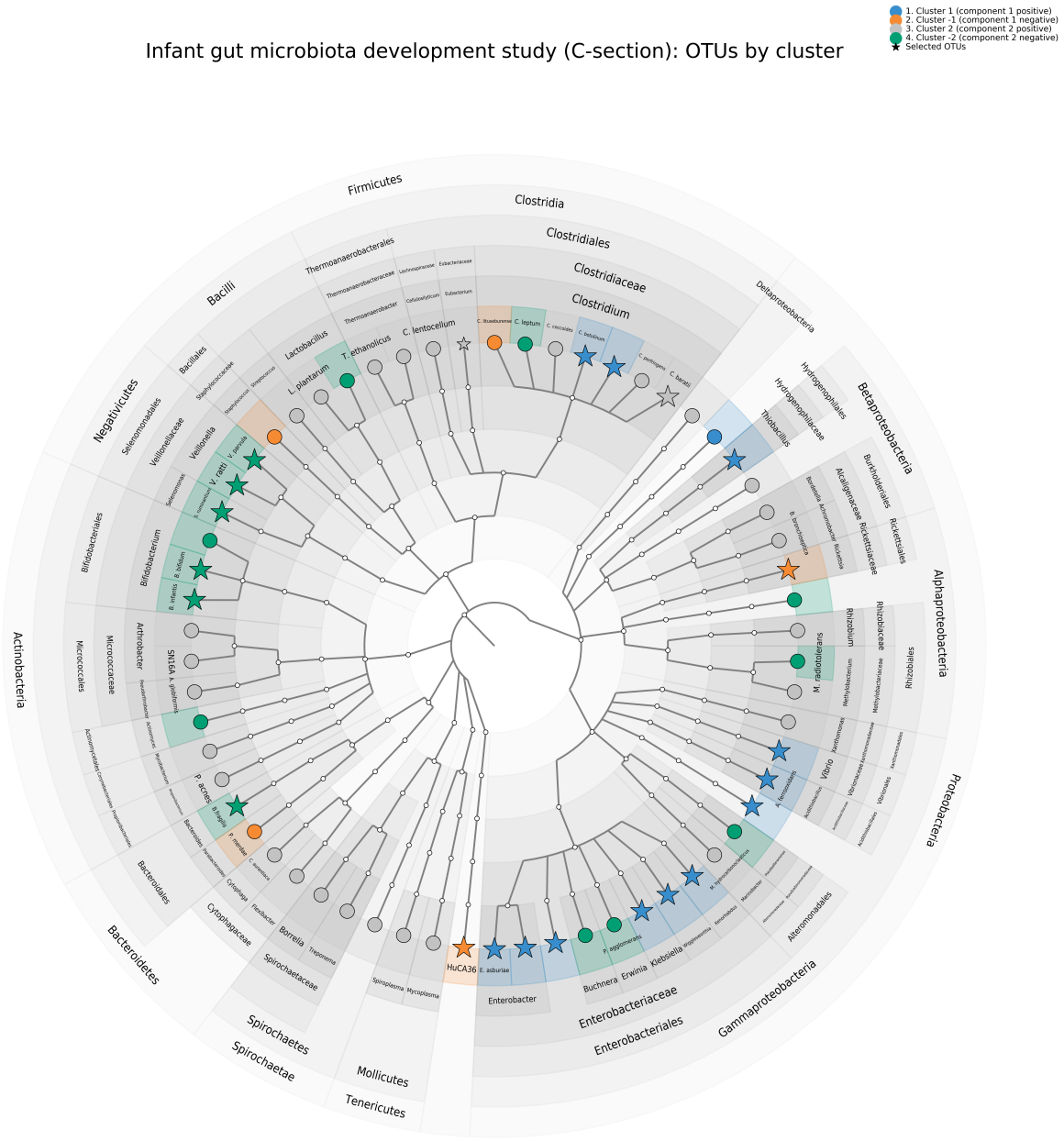


Figure 12: Silhouette Graph for C-section data clustering

We finally display above the trajectories of the selected OTUs.

The phylogenetic tree below were produced using GraPhlAn tools. To create such cladograms, GraPhlAn needs a taxonomy file describing the tree structure as well as an annotation file. The latter was generated partly through R-scripts (`./graphlan_csection.R`) and was finalised by hand. The final annotation file is present here (`../Data/annotation_csection.txt`) and below is the bash cammandes to reproduce the tree.

```
graphlan_annotate.py --annot ../Data/annotation_csection.txt \
  ../Data/tree_csection.txt tree_csection.xml
graphlan.py tree_csection.xml tree_csection.png --dpi 600 --size 10
```



Vaginal

We applied the same selection method on the vaginal data clusters.

We selected 17 OTUs for the first component because the first drop is observed for the positive cluster, after this value. We selected 10 OTUs on the second component because the silhouette coefficient reaches a maximum for the positive and negative cluster. Although the silhouette coefficient remains the same for the negative cluster, it decreases for the positive cluster.

The average silhouette coefficient is 0.86.

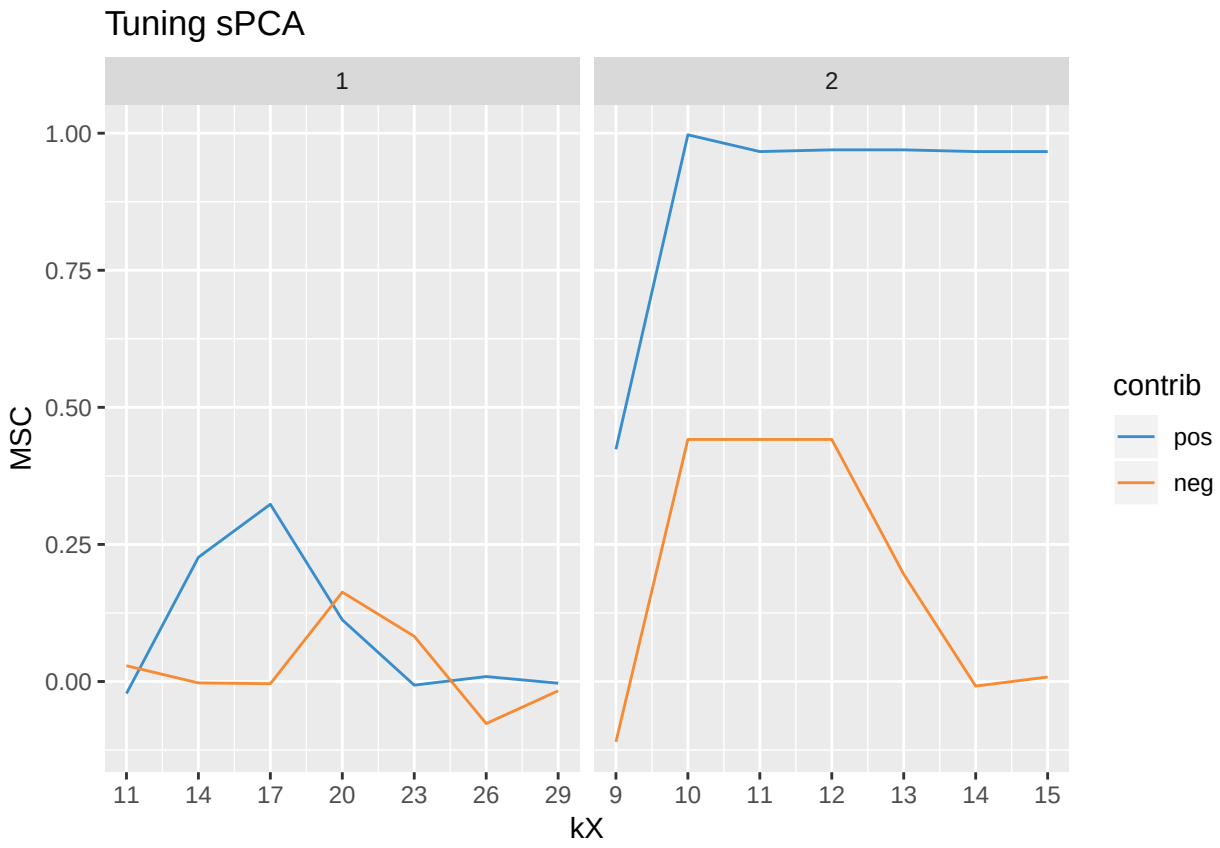


Figure 13: sPCA tuning plot

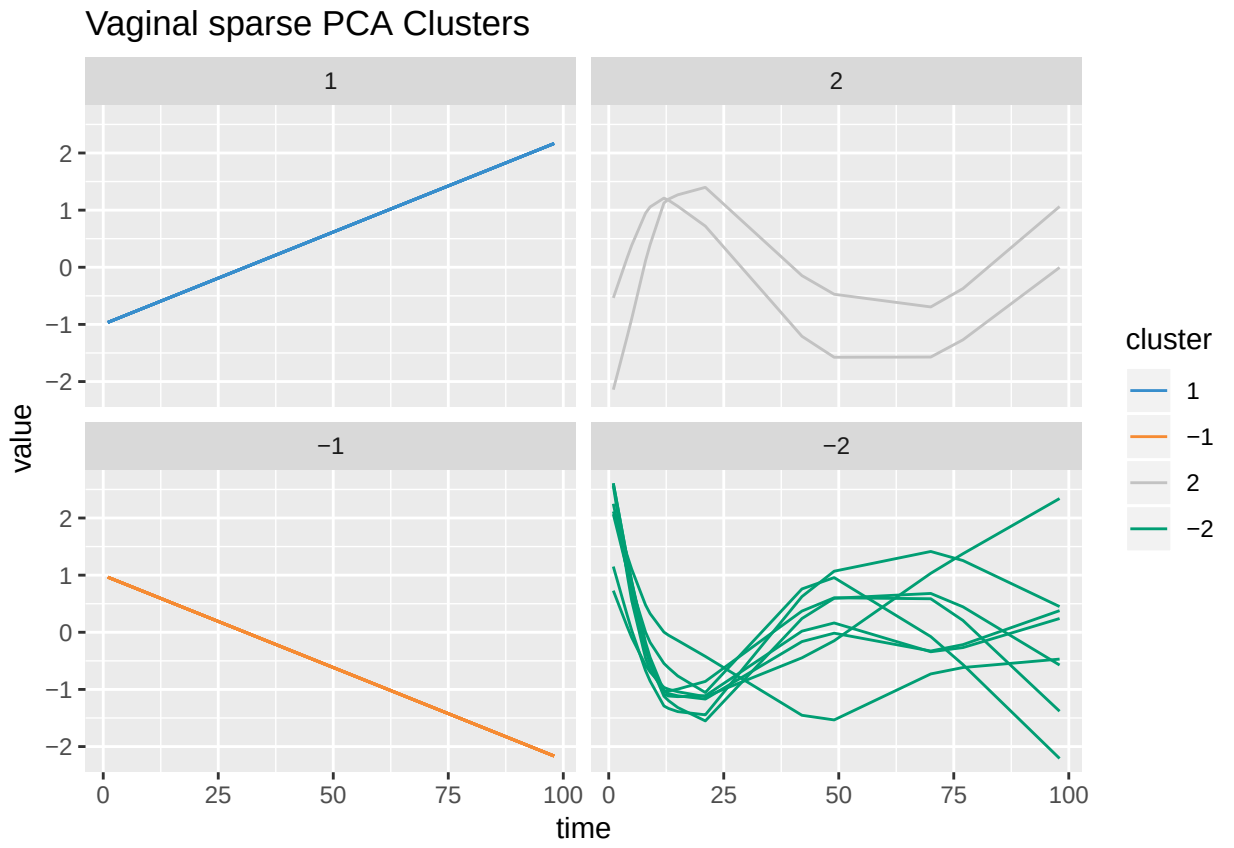


Figure 14: Silhouette Graph for vaginal data clustering

GraPhlAn was used to buid this tree. Annotation file was patialy build with R-scripts (`./graphlan_vaginal.R`) and was finalised by hand. The final annotation file is present here (`../Data/annotation__csection_vaginal.txt`) and below is the bash cammandes to reproduce the tree.

```
graphlan_annotate.py --annot ../Data/annotation_vaginal.txt \
  ../Data/tree_vaginal.txt tree_vaginal.xml
graphlan.py tree_vaginal.xml tree_vaginal.png --dpi 600 --size 10
```

Results

The results are summarized below.

In the following table, we summarize the average silhouette coefficient by method and data set. A higher average silhouette coefficient indicates a better partitioning.

	PCA	sPCA
C-section	0.84	0.95
Vaginal	0.87	0.86

LMMS models each OTU with 4 different models Here only linear (0) and mixed model spline (1) models are used and we summarize the number of OTUs modelled by each model in the following table.

	0	1
C-section	42	29
Vaginal	68	22

Comparison with Functional Principal Component Analysis clustering

Functional Principal Component Analysis (fPCA) is a popular approach to cluster longitudinal data and it extracts 'modes of variation'. It performs functional clustering using k-centres functional Clustering (k-CFC) or model-based clustering using an Expectation-Maximization algorithm (EM).

In this section, we compared, for each dataset, our clustering results with the clustering methods proposed with the fPCA method : EM and kCFC.

Note: we used the `fdapace` R package version `0.4.0`, please install the correct version before using it `devtools::install_version('fdapace', version = '0.4.0')`

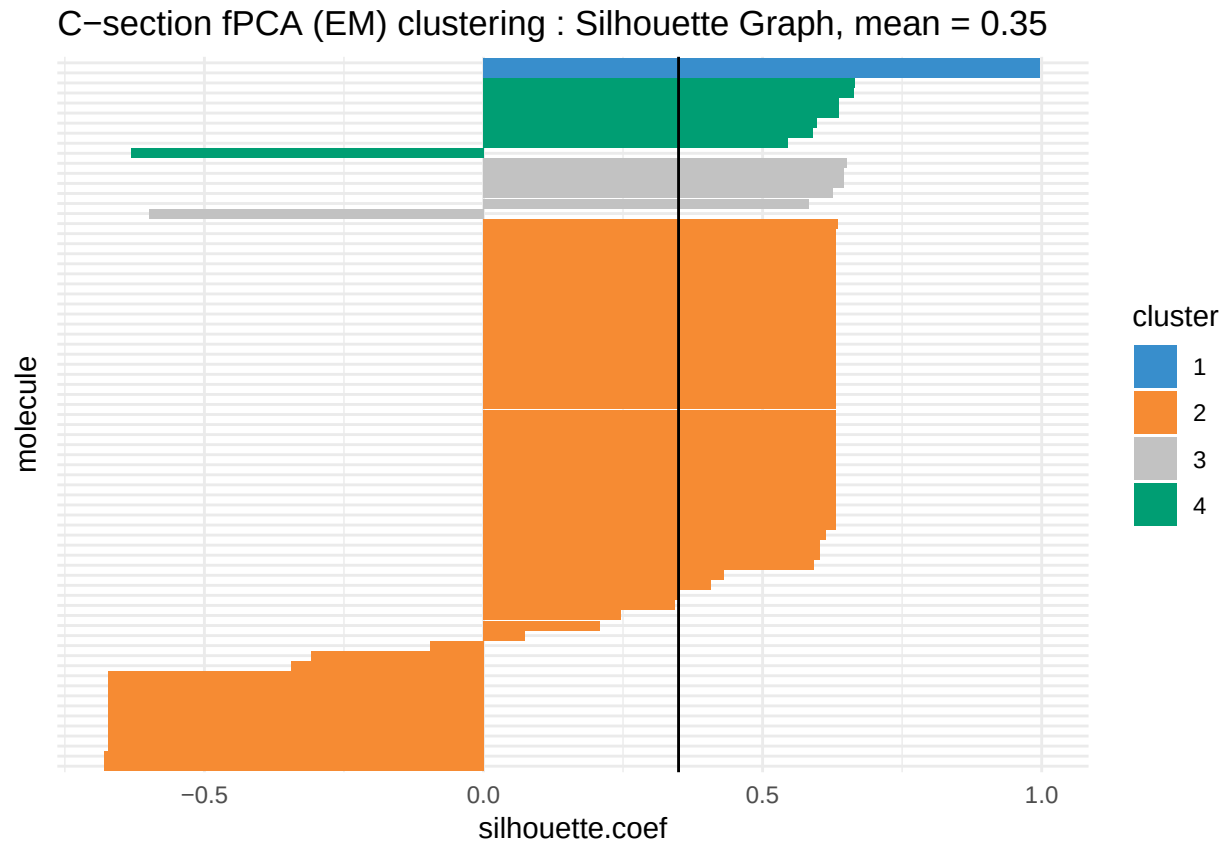
```
library(fdapace)
# check fdapace version
if(packageVersion('fdapace') != '0.4.0'){
    stop("We are using `fdapace` in version 0.4.0, please install the correct version
         devtools::install_version('fdapace', version = '0.4.0')")
}
```

C-section

EM

The figure below shows the silhouette profile for C-section data. With the Expectation-Maximisation clustering method associated to the fPCA, the profile shows us a large proportion of misclassified OTUs since they have a silhouette coefficient below 0.

```
## [1] 0.3494568
```


C−section fPCA (EM) clustering : Silhouette Graph, mean = 0.35
molecule
silhouette.coef
−0.5
0.0
0.5
1.0
cluster
1
2
3
4


C−section fPCA (EM) Clusters
1
2
3
4
value
time
8
4
0
−4
0
25
50
75
100

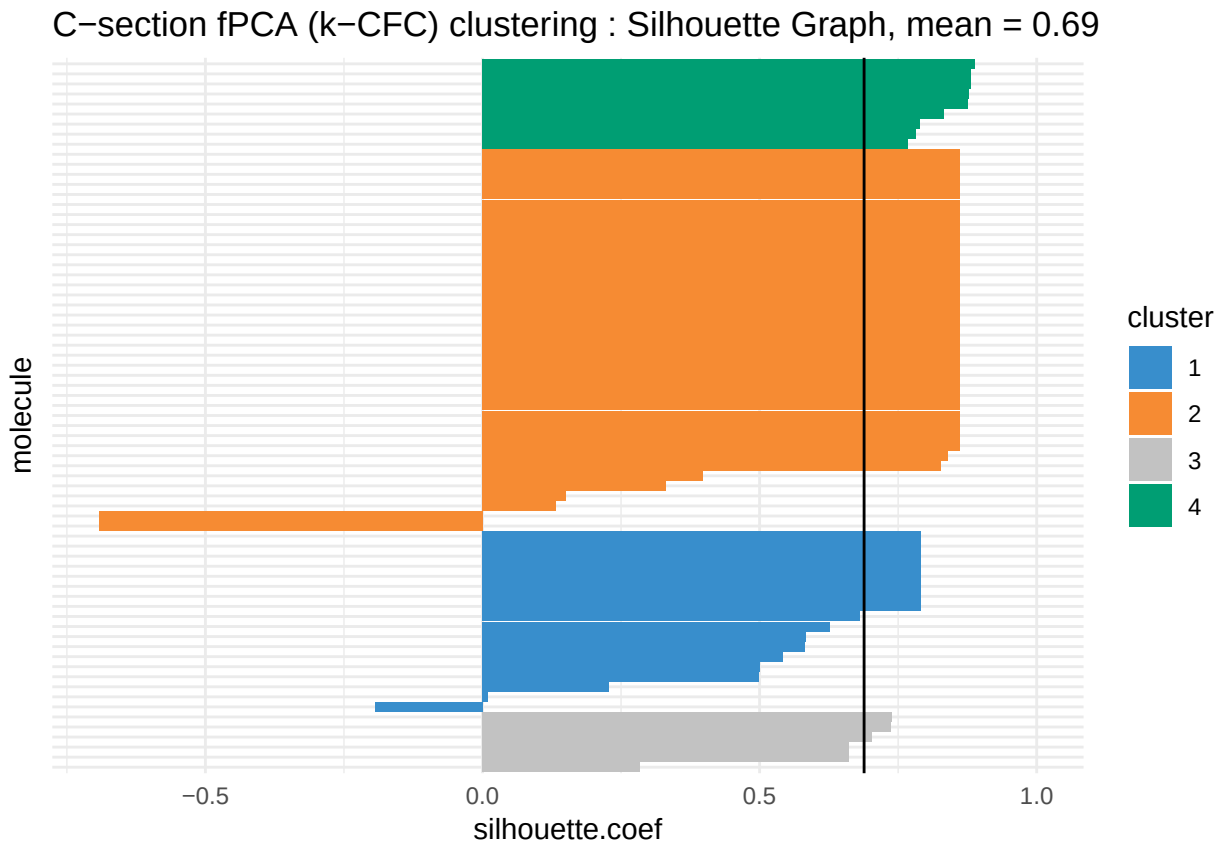
In the figure above which shows the trajectories per cluster, cluster 2 is very noisy.

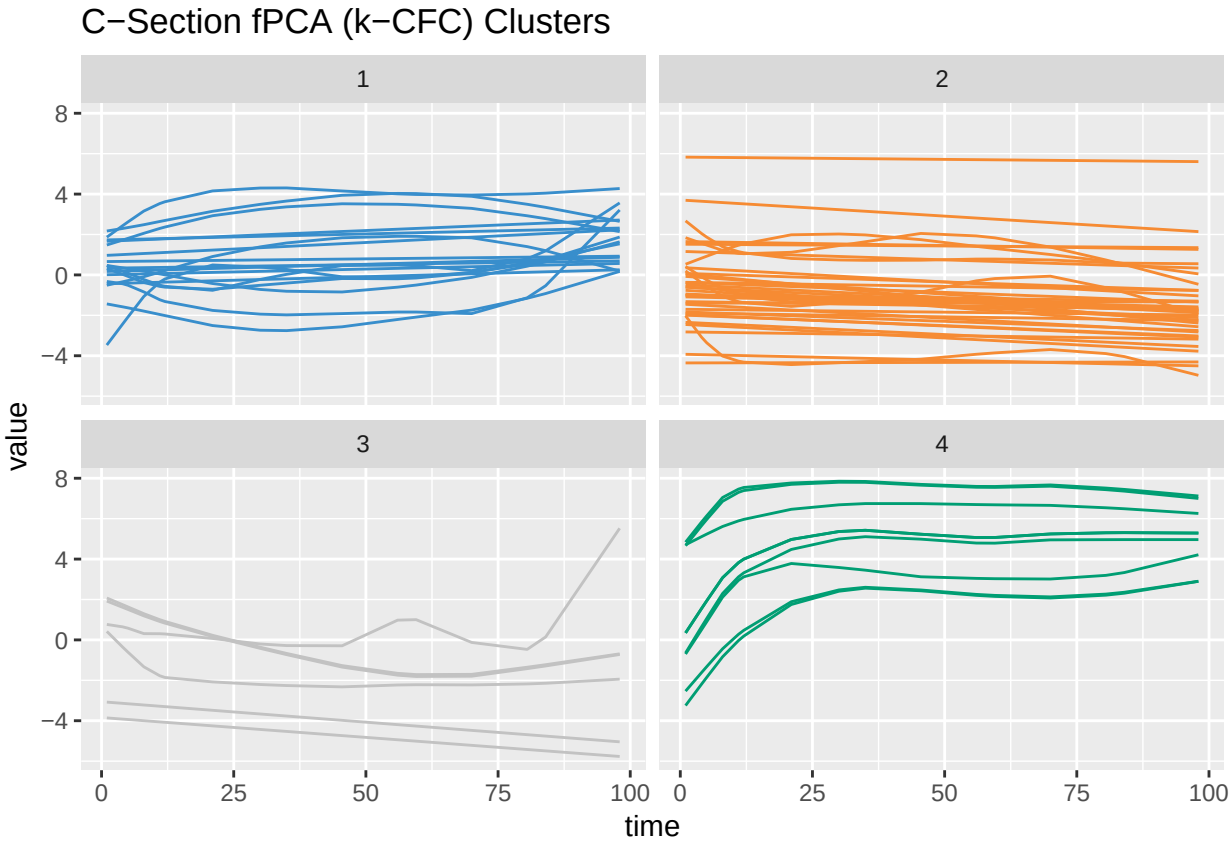
For this clustering, the average silhouette coefficient is very low (0.35) and indicates a poor clustering.

kCFC

We also tested the kCFC clustering method associated to the FPCA for C-section data.

```
## [1] 0.6885726
```





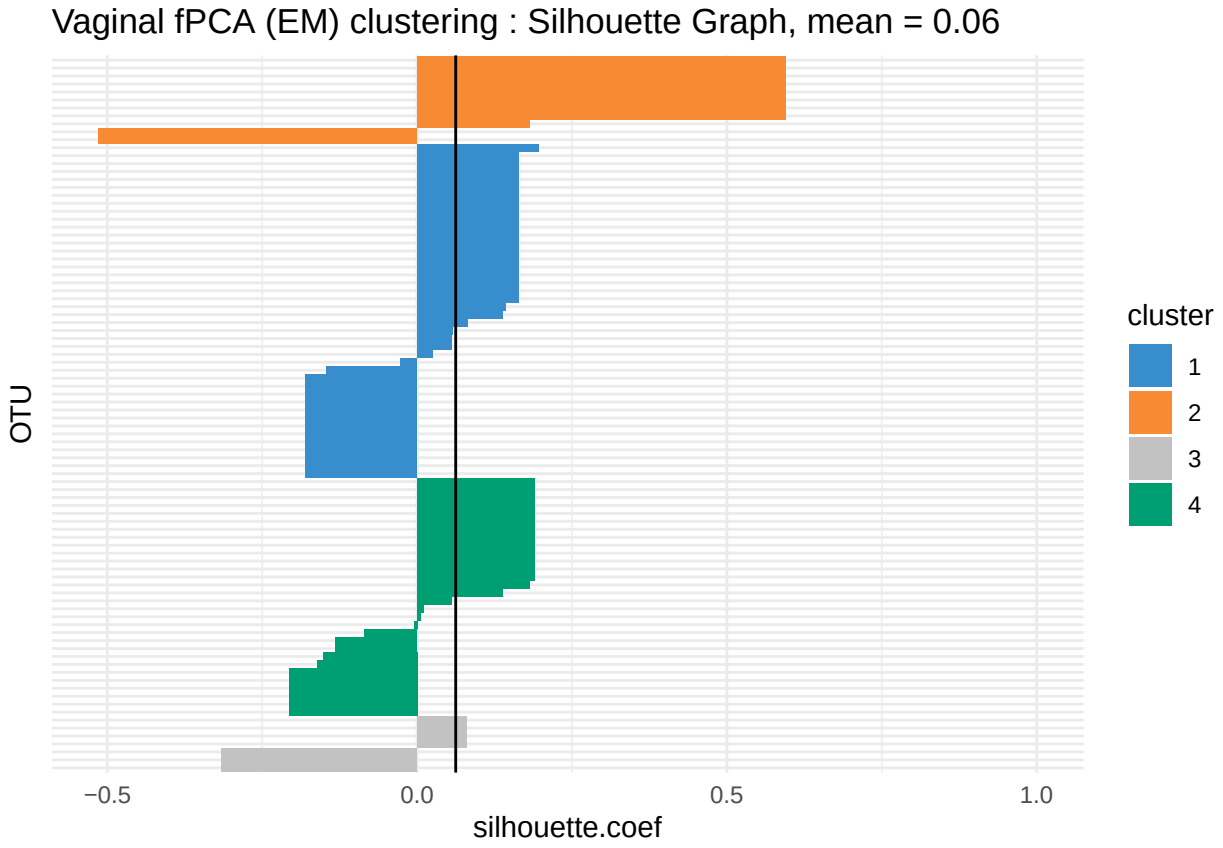
According to the average silhouette coefficient (0.69), clustering with kCFC is better than the EM method. We can corroborate this by observing the trajectories by cluster. Nevertheless, the PCA seems to produce better clustering results for C-section data (0.84) .

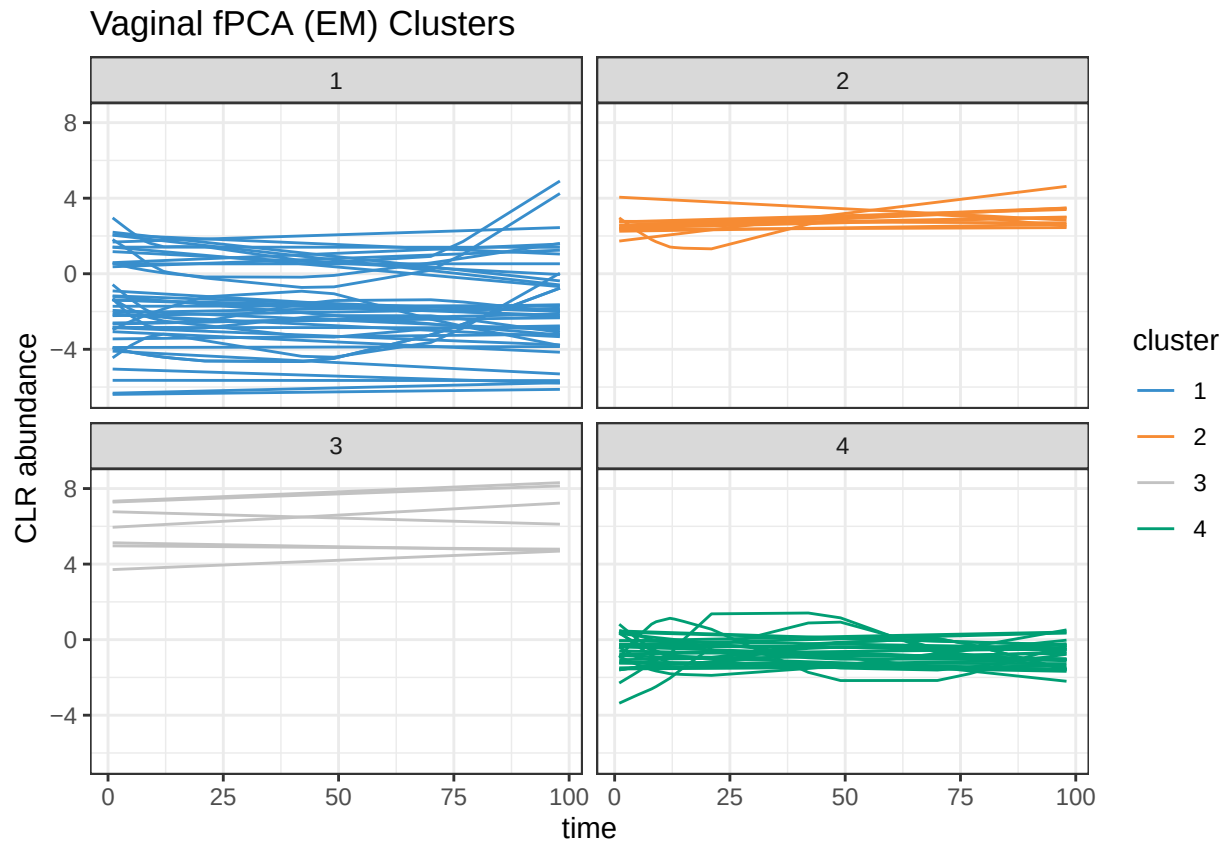
Vaginal

We tested the 2 previous methods with vaginal data.

EM

```
## [1] 0.06230024
```

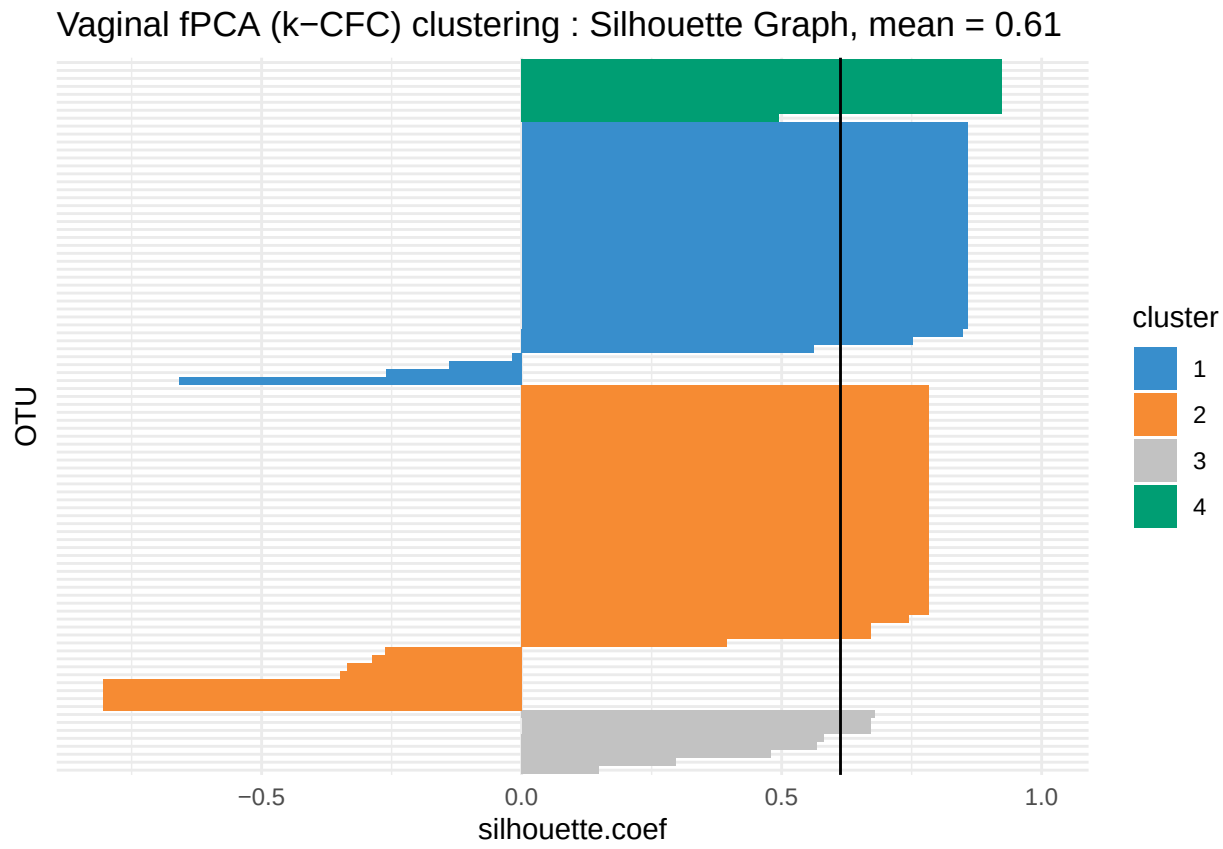

Vaginal fPCA (EM) clustering : Silhouette Graph, mean = 0.06
OTU
−0.5
0.0
0.5
1.0
silhouette.coef
cluster
1
2
3
4

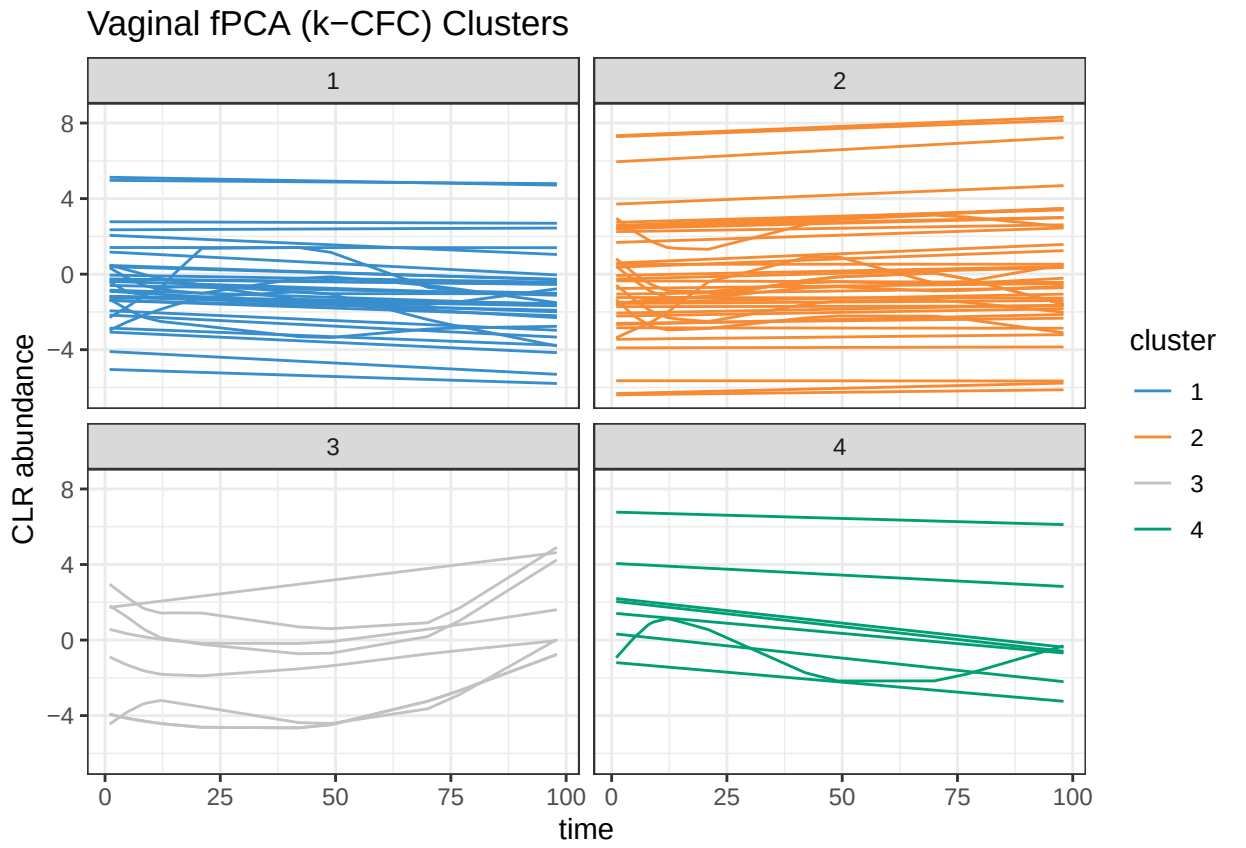

Vaginal fPCA (EM) Clusters
1
2
3
4
CLR abundance
8
4
0
−4
0
25
50
75
100
time
cluster
1
2
3
4

The EM method gives the worst results (0.06).

kCFC

```
## [1] 0.6131915
```





The kCFC method gives better results (0.61)than EM clustering but PCA clustering is much better according to our criterion (0.87) for vaginal data.

Finally, in every situation in this example, PCA clustering gives better results according to the average silhouette coefficient.

Below, you will find the packages and their versions required to reproduce this example.

```
sessionInfo()
```

```
## R version 3.5.3 (2019-03-11)
## Platform: x86_64-redhat-linux-gnu (64-bit)
## Running under: Fedora 29 (Workstation Edition)
##
## Matrix products: default
## BLAS/LAPACK: /usr/lib64/R/lib/libRblas.so
##
## 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] parallel  stats     graphics  grDevices utils     datasets  methods
## [8] base
##
```

```
## other attached packages:
##  [1] fdapace_0.4.0     bindrcpp_0.2.2    pspline_1.0-18
##  [4] geiger_2.0.6      ape_5.2           usethis_1.5.0
##  [7] devtools_2.0.2    clValid_0.6-6     edci_1.1-3
## [10] mclust_5.4.2      cluster_2.0.7-1   dynOmics_1.2
## [13] lmeSplines_1.1-10 nlme_3.1-137      knitr_1.22
## [16] reshape2_1.4.3    lmms_1.3.3        lmtest_0.9-36
## [19] zoo_1.8-4         tseries_0.10-46   mixOmics_6.6.0
## [22] lattice_0.20-38   MASS_7.3-51.1     forcats_0.3.0
## [25] stringr_1.3.1     dplyr_0.7.8       purrr_0.2.5
## [28] readr_1.3.0       tidyr_0.8.2       tibble_1.4.2
## [31] ggplot2_3.1.0     tidyverse_1.2.1
##
## loaded via a namespace (and not attached):
##   [1] colorspace_1.3-2    class_7.3-15        rprojroot_1.3-2
##   [4] htmlTable_1.12      corpcor_1.6.9       base64enc_0.1-3
##   [7] fs_1.2.6            rstudioapi_0.8      remotes_2.0.2
##  [10] fansi_0.4.0         RSpectra_0.13-1     mvtnorm_1.0-11
##  [13] lubridate_1.7.4     xml2_1.2.0          codetools_0.2-16
##  [16] splines_3.5.3       pkgload_1.0.2       Formula_1.2-3
##  [19] jsonlite_1.6        broom_0.5.1         compiler_3.5.3
##  [22] httr_1.4.0          backports_1.1.2     assertthat_0.2.0
##  [25] Matrix_1.2-15       lazyeval_0.2.1      cli_1.0.1
##  [28] acepack_1.4.1       htmltools_0.3.6     prettyunits_1.0.2
##  [31] tools_3.5.3         igraph_1.2.2        coda_0.19-2
##  [34] gtable_0.2.0        glue_1.3.0          Rcpp_1.0.0
##  [37] cellranger_1.1.0    gdata_2.18.0        xfun_0.3
##  [40] ps_1.2.1            testthat_2.0.1      rvest_0.3.2
##  [43] gtools_3.8.1        scales_1.0.0        subplex_1.5-4
##  [46] hms_0.4.2           RColorBrewer_1.1-2  yaml_2.2.0
##  [49] quantmod_0.4-13     curl_3.2            memoise_1.1.0
##  [52] gridExtra_2.3       rpart_4.1-13        latticeExtra_0.6-28
##  [55] stringi_1.2.3       highr_0.7           desc_1.2.0
##  [58] checkmate_1.8.5     TTR_0.23-4          caTools_1.17.1.1
##  [61] pkgbuild_1.0.2      rlang_0.4.0         pkgconfig_2.0.2
##  [64] matrixStats_0.54.0  bitops_1.0-6        pracma_2.2.2
##  [67] evaluate_0.12       bindr_0.1.1         labeling_0.3
##  [70] htmlwidgets_1.3     tidyselect_0.2.5    processx_3.2.1
##  [73] deSolve_1.21        plyr_1.8.4          magrittr_1.5
##  [76] R6_2.3.0            snow_0.4-3          Hmisc_4.1-1
##  [79] gplots_3.0.1        generics_0.0.2      EMCluster_0.2-10
##  [82] foreign_0.8-71      pillar_1.3.0        haven_2.0.0
##  [85] withr_2.1.2         xts_0.11-2          nnet_7.3-12
##  [88] survival_2.43-3     modelr_0.1.2        crayon_1.3.4
##  [91] rARPACK_0.11-0      utf8_1.1.4          KernSmooth_2.23-15
##  [94] ellipse_0.4.1       rmarkdown_1.10.14   grid_3.5.3
##  [97] readxl_1.1.0        data.table_1.11.8   propr_4.1.1
## [100] callr_3.1.0         htmldeps_0.1.1      digest_0.6.18
## [103] numDeriv_2016.8-1   munsell_0.5.0       sessioninfo_1.1.1
## [106] quadprog_1.5-5
```