Influence of parameters

Morgane Pierre-Jean

2019-09-12

```
library(ggplot2)
library(tibble)
library(dplyr)
library(CrIMMix)
```

In this document we show the influence of sparsity parameters (for MoCluster and SGCCA), number of latent profiles (for MoCluster and SGCCA) and τ parameter for RGCCA. The first section simulates three heterogeneous blocks with 4 unbalanced groups composed of 10, 20, 5 and 25 individuals.

Simulations

Define paramaters for simulations

```
means <- c(2,2,2,2)
sds <- c(1,1,1,1)
params <- mapply(function (m, sd) return(c(mean=m, sd=sd)), means, sds, SIMPLIFY=FALSE)
params_beta <- list(c(mean1=-2, mean2=2, sd1=0.5, sd2=0.5))
S <- 50
nclust=4
n_byClust=c(10,20,5,25)
```

```
noiseD1=c(0.2)
noiseD2=c(0.1)/10
noiseD3=c(0.1)*3
props <- c(0.005, 0.01, 0.02)
```

Simulations of data sets

Here, we simulate three blocks (`dat1`, `dat2`, `dat3`).

```
dat1 <- simulateY(nclust=nclust,n_byClust=n_byClust, J=1000,
                  prop=props[1],params=params, noise=noiseD1)
Y1 <- dat1$data


dat2 <- simulateY(nclust=nclust,n_byClust=n_byClust, J=500, flavor="binary",
                  params=list(c(p=0.6)), prop=props[2], noise=noiseD2)

Y2 <- dat2$data

dat3 <- simulateY(nclust=nclust,n_byClust=n_byClust, J=5000,
                  flavor="beta", params=params_beta, prop=props[3], noise=noiseD3)
Y3 <- dat3$data
```

```
sim <- list(data= list(dat1=Y1, dat2= Y2,dat3=Y3),
            biomark = list(dat1=dat1$positive,
                           dat2=dat2$positive,
                           dat3=dat3$positive),
            true.clust = dat1$true.clusters)

truth <- lapply(lapply(sim$biomark, unlist), unique)
```

Influence parameter

In this section, we evaluate the influence of the number of the latent variables for Mocluster and SGCCA. We define a grid for the value of the number of latent variables.

Influence of number of latent profiles

MoCluster

```
k.grid=c(0.05,0.4, 0.1)%*%t(c(0.1, 0.2,0.5,1,2,5,10))
ncomp.grid <- 1:8
```

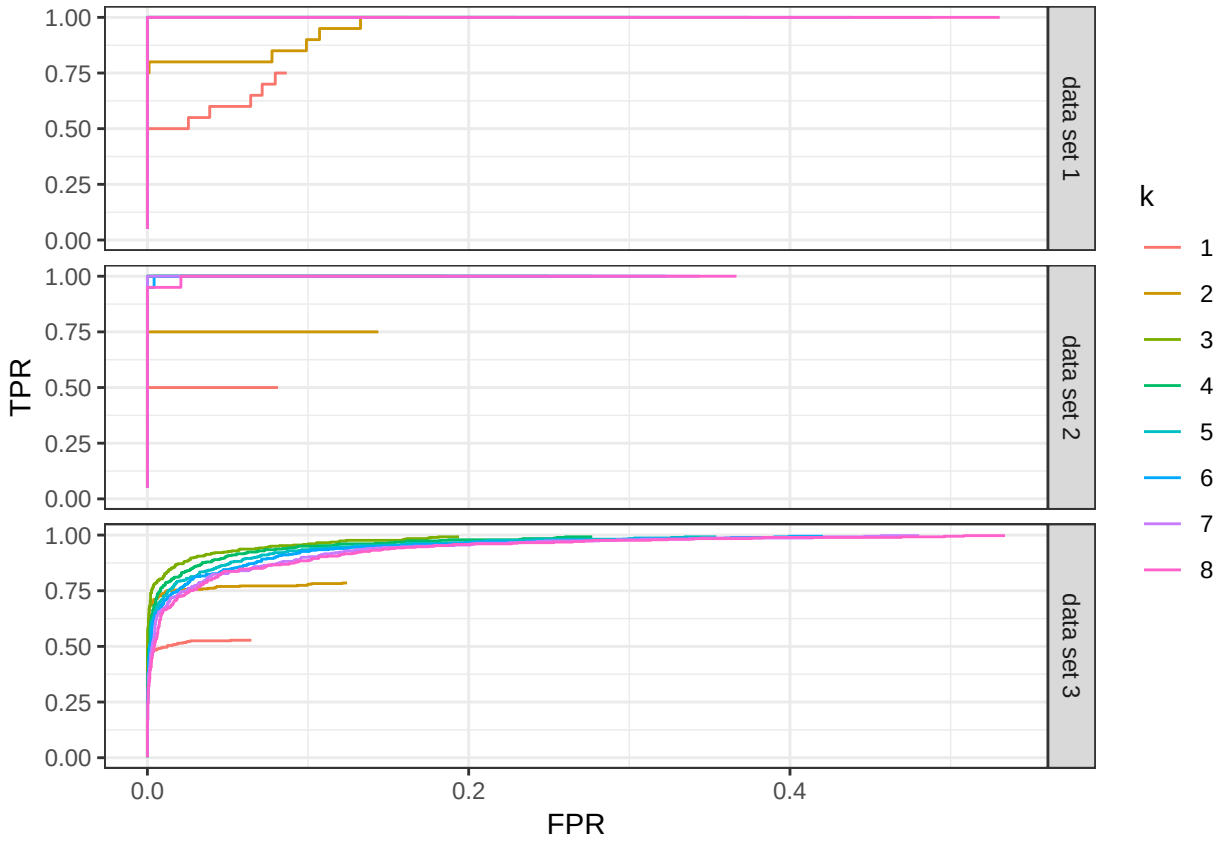
```
Moaresults <- lapply(ncomp.grid,  IntMultiOmics, method="Mocluster",
                     data=sim$data, k=c(0.05*2, 0.05*2, 0.1), K=4 )
```

ROC evaluation

```
auc_eval_moclust <- sapply (Moaresults, function(mm) {
  roc_eval(truth= truth, fit = mm$fit, method = "Mocluster")
}, simplify = FALSE)
```

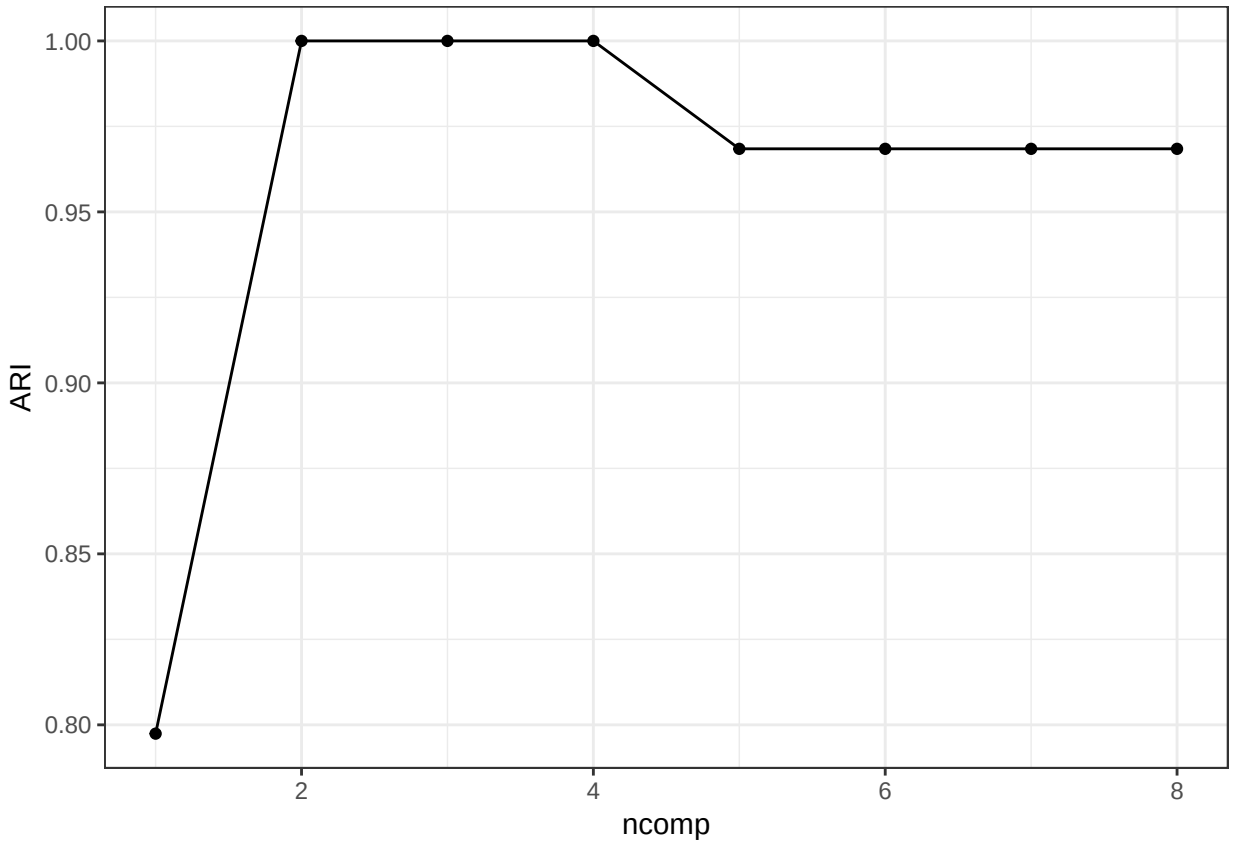
```
g_moclust <- do.call(rbind, lapply(1:length(auc_eval_moclust), function (ss) {
  dd <- auc_eval_moclust[[ss]]
  n_by_data_set <- sapply(dd$TPR, length)
  tprs <- dd$TPR %>% unlist
  fprs <- dd$FPR %>% unlist
  data.frame(TPR=tprs, FPR=fprs,
             dataSet= sprintf("data set %s", rep(1:3, times=n_by_data_set)),
             k=as.factor(ss))
}))
```

```
g_moclust %>% ggplot(aes(x=FPR, y=TPR, color=k))+geom_line()+facet_grid(dataSet~.)+theme_bw()
```



ARI evaluation

```
ari_moclust <- sapply (Moaresults, function(mm) {
  adjustedRIComputing(mm,sim$true.clust)
}, simplify = TRUE)

df_ari <- data.frame(ncomp=ncomp.grid, ARI=ari_moclust)
df_ari %>% ggplot(aes(x=ncomp, y=ARI))+geom_point()+geom_line()+theme_bw()
```



SGCCA

```
ncomp.grid <- rep(1:8,length(sim$data)) %>% matrix(ncol=3)
SGCCAresults <- apply(ncomp.grid,  1, IntMultiOmics, method="SGCCA",
                      data=sim$data, C=1-diag(length(sim$data)),c1=c(0.3, 0.3,0.4), K=4)
```

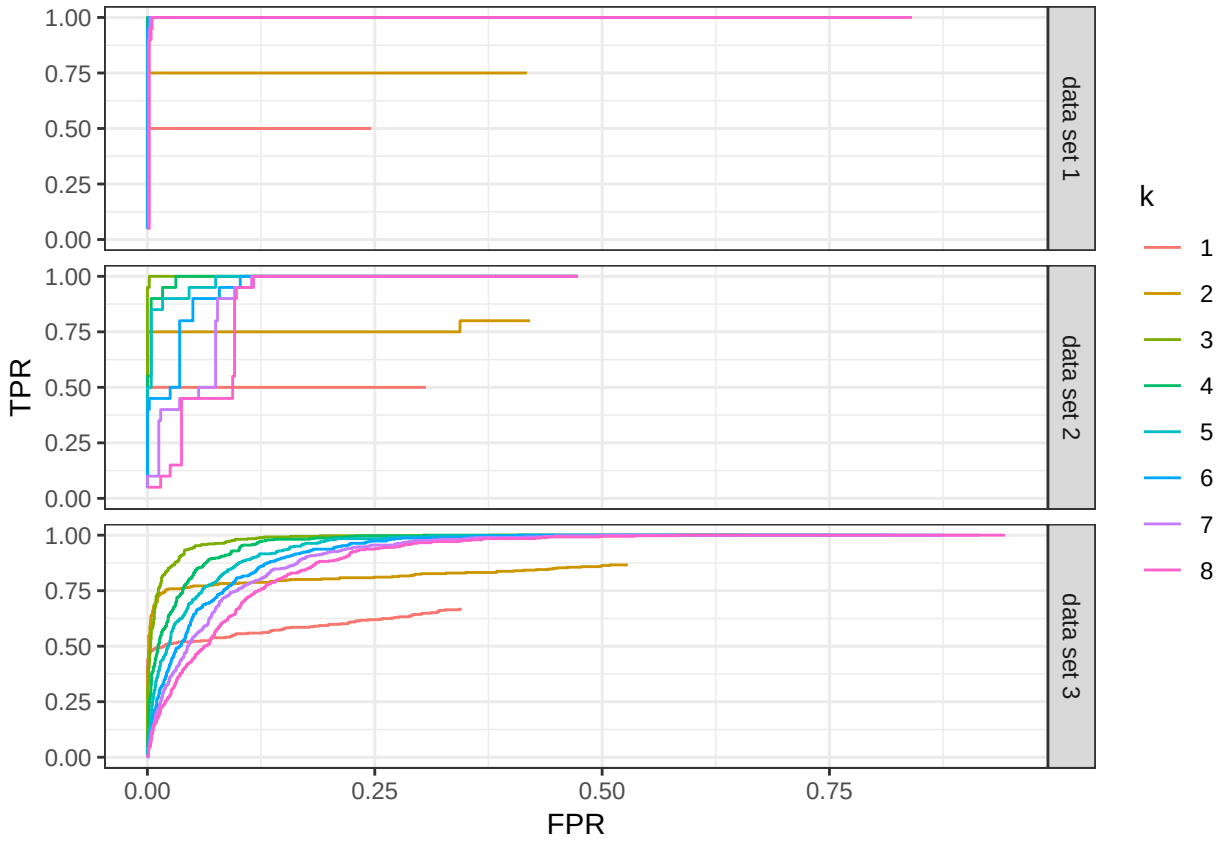
```
## Warning in cor(A[[j]], Y[[j]]): l'écart type est nulle

## Warning in cor(A[[j]], Y[[j]]): l'écart type est nulle

## Warning in cor(A[[j]], Y[[j]]): l'écart type est nulle

## Warning in cor(A[[j]], Y[[j]]): l'écart type est nulle

## Warning in cor(A[[j]], Y[[j]]): l'écart type est nulle

## Warning in cor(A[[j]], Y[[j]]): l'écart type est nulle

## Warning in cor(A[[j]], Y[[j]]): l'écart type est nulle

## Warning in cor(A[[j]], Y[[j]]): l'écart type est nulle
```

ROC evaluation

```
auc_eval_SGCCA <- sapply (SGCCAresults, function(mm) {
  roc_eval(truth= truth, fit = mm$fit, method = "SGCCA")
}, simplify = FALSE)
```

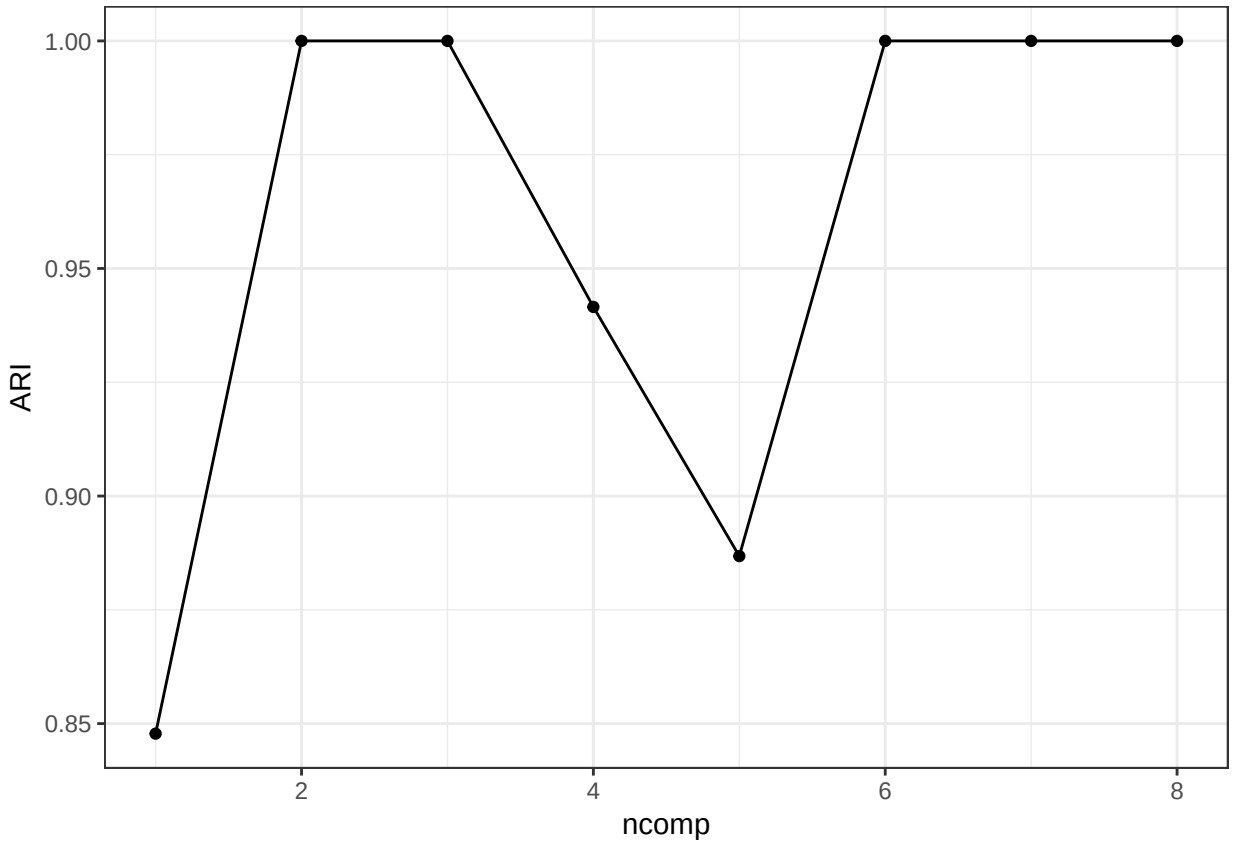
```
g_sgcca <- do.call(rbind, lapply(1:length(auc_eval_SGCCA), function (ss) {
  dd <- auc_eval_SGCCA[[ss]]
  n_by_data_set <- sapply(dd$TPR, length)
  tprs <- dd$TPR %>% unlist
  fprs <- dd$FPR %>% unlist
  data.frame(TPR=tprs, FPR=fprs,
             dataSet= sprintf("data set %s", rep(1:3, times=n_by_data_set)),
             k=as.factor(ss))
}))
```

```
g_sgcca %>% ggplot(aes(x=FPR, y=TPR, color=k))+geom_line()+facet_grid(dataSet~.)+theme_bw()
```



ARI

```
ari_SGCCA <- sapply (SGCCAresults, function(mm) {
  adjustedRIComputing(mm,sim$true.clust)
}, simplify = TRUE)

df_ari <- data.frame(ncomp=ncomp.grid[,1], ARI=ari_SGCCA)
df_ari %>% ggplot(aes(x=ncomp, y=ARI))+geom_point()+geom_line()+theme_bw()
```



Influence of sparsity parameters

As for the number of latent profiles, we evaluate the influence of the sparsity parameters for MoCluster and SGCCA by defining a grid of values.

MoCluster

```
k.grid=c(0.05,0.4, 0.1)%*%t(c(0.1, 0.2,0.5,1,2,5,10))
```

```
Moaresults_k <- apply(k.grid,2,  IntMultiOmics, method="Mocluster",
                      data=sim$data, K=4, ncomp=3)
```

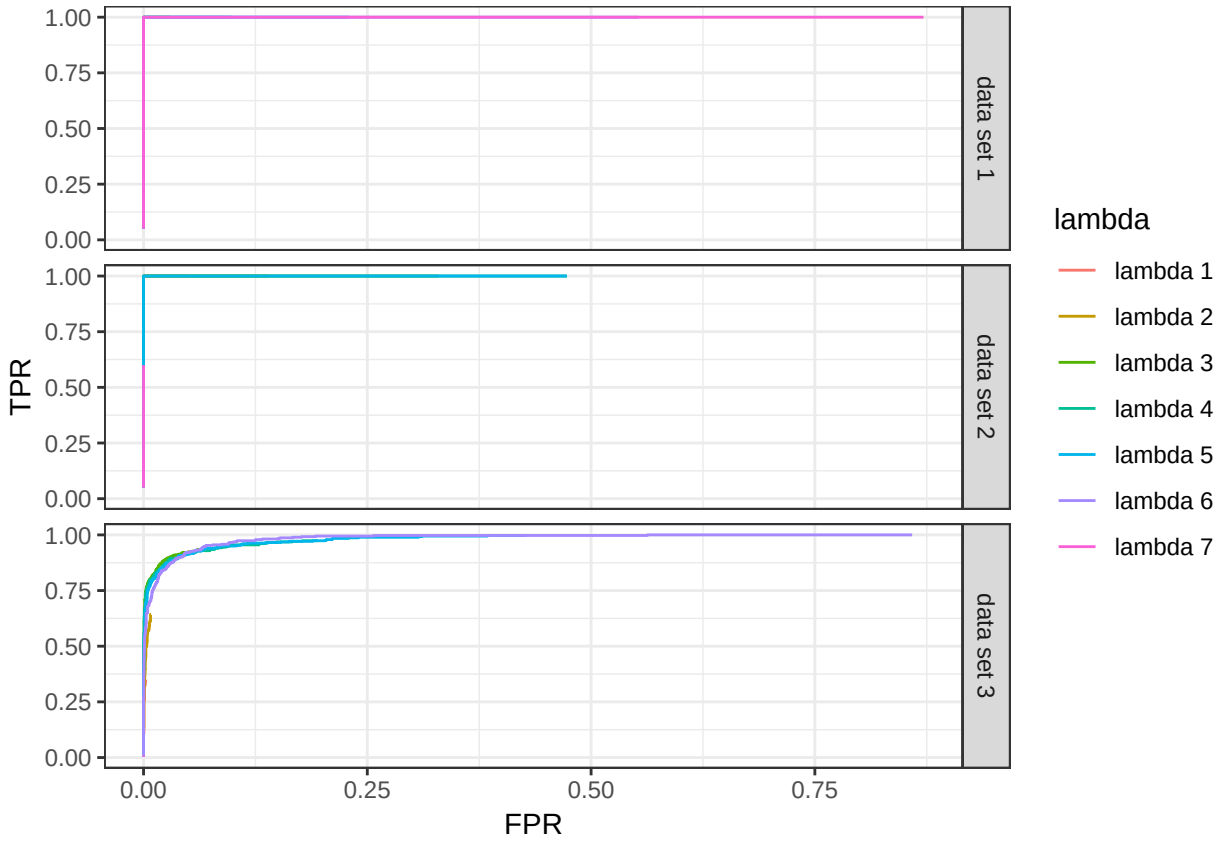
ROC evaluation

```
auc_eval_moclust_k <- sapply (Moaresults_k, function(mm) {
  roc_eval(truth= truth, fit = mm$fit, method = "Mocluster")
}, simplify = FALSE)
```

```
g_moclust_k <- do.call(rbind, lapply(1:length(auc_eval_moclust_k), function (ss) {
  dd <- auc_eval_moclust_k[[ss]]
  n_by_data_set <- sapply(dd$TPR, length)
  tprs <- dd$TPR %>% unlist
  fprs <- dd$FPR %>% unlist
  data.frame(TPR=tprs, FPR=fprs,
```

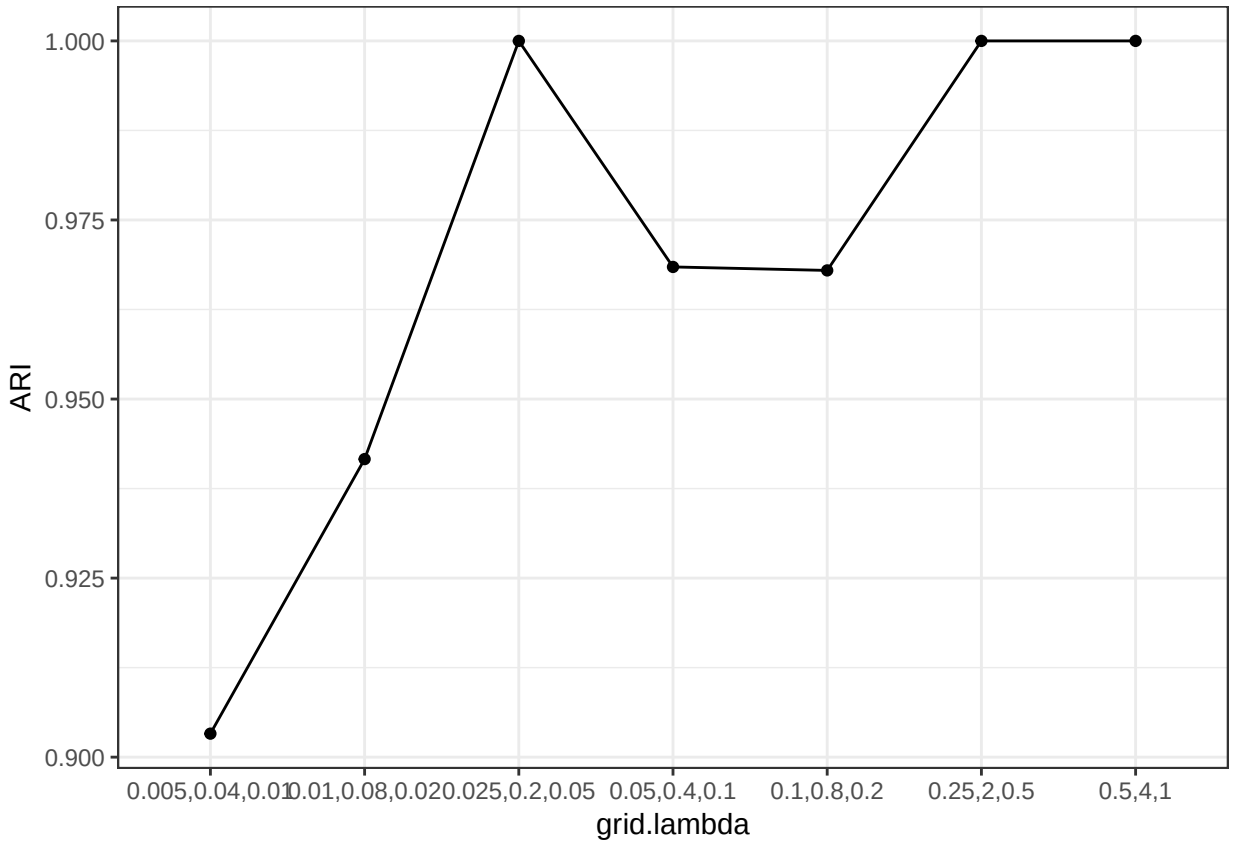
```
             dataSet= sprintf("data set %s", rep(1:3, times=n_by_data_set)),
            lambda=sprintf("lambda %s", as.factor(ss)))
}))
```

```
g_moclust_k %>% ggplot(aes(x=FPR, y=TPR, color=lambda))+geom_line()+facet_grid(dataSet~.)+theme_bw()
```



ARI evaluation

```
ari_moclust <- sapply (Moaresults_k, function(mm) {
  adjustedRIComputing(mm,sim$true.clust)
}, simplify = TRUE)


df_ari <- data.frame(grid.lambda=apply(k.grid, 2, paste,collapse=","), ARI=ari_moclust)
df_ari %>% ggplot(aes(x=grid.lambda, y=ARI, group=1))+geom_point()+geom_line()+theme_bw()
```



SGCCA

```
c1.grid <-c(0.3, 0.3,0.4)%*%t(c(0.2,0.5,1,2))
SGCCAresults_k <- apply(c1.grid,  2, IntMultiOmics, method="SGCCA",
                    data=sim$data, K=4, C=1-diag(length(sim$data)), ncomp=rep(3,3))
```

```
## Warning in cor(A[[j]], Y[[j]]): l'écart type est nulle

## Warning in cor(A[[j]], Y[[j]]): l'écart type est nulle

## Warning in cor(A[[j]], Y[[j]]): l'écart type est nulle

## Warning in cor(A[[j]], Y[[j]]): l'écart type est nulle
```

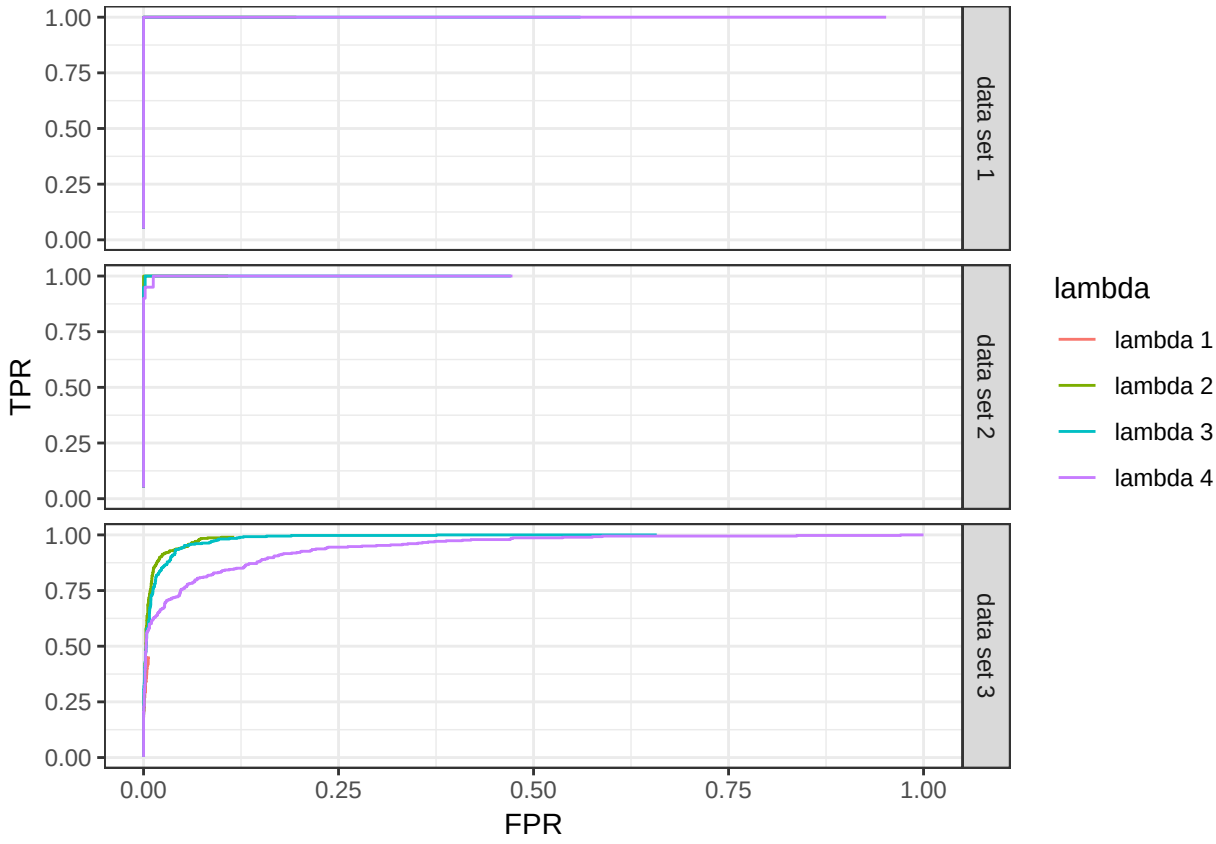
ROC evaluation

```
auc_eval_SGCCA_k <- sapply (SGCCAresults_k, function(mm) {
  roc_eval(truth= truth, fit = mm$fit, method = "SGCCA")
}, simplify = FALSE)
```

```
g_sgcca_k <- do.call(rbind, lapply(1:length(auc_eval_SGCCA_k), function (ss) {
  dd <- auc_eval_SGCCA_k[[ss]]
  n_by_data_set <- sapply(dd$TPR, length)
  tprs <- dd$TPR %>% unlist
  fprs <- dd$FPR %>% unlist
  data.frame(TPR=tprs, FPR=fprs,
```

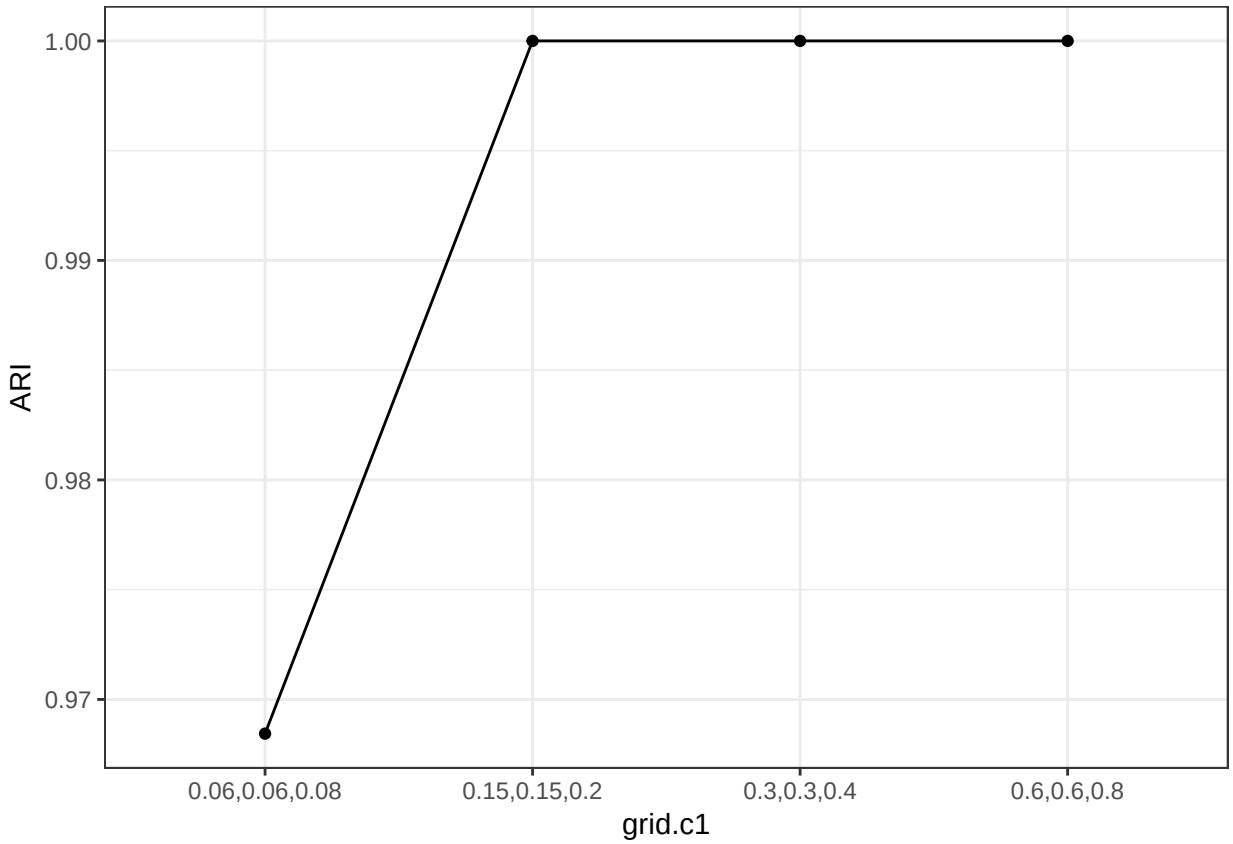
```
            dataSet= sprintf("data set %s", rep(1:3, times=n_by_data_set)),
            lambda=sprintf("lambda %s", as.factor(ss)))
}))
```

```
g_sgcca_k %>% ggplot(aes(x=FPR, y=TPR, color=lambda))+geom_line()+facet_grid(dataSet~.)+theme_bw()
```



ARI evaluation

```
ari_sgcca <- sapply (SGCCAresults_k, function(mm) {
  adjustedRIComputing(mm,sim$true.clust)
}, simplify = TRUE)

df_ari <- data.frame(grid.c1=apply(c1.grid, 2, paste,collapse=","), ARI=ari_sgcca)
df_ari %>% ggplot(aes(x=grid.c1, y=ARI, group=1))+geom_point()+geom_line()+theme_bw()
```



τ parameter influence

RGCCA

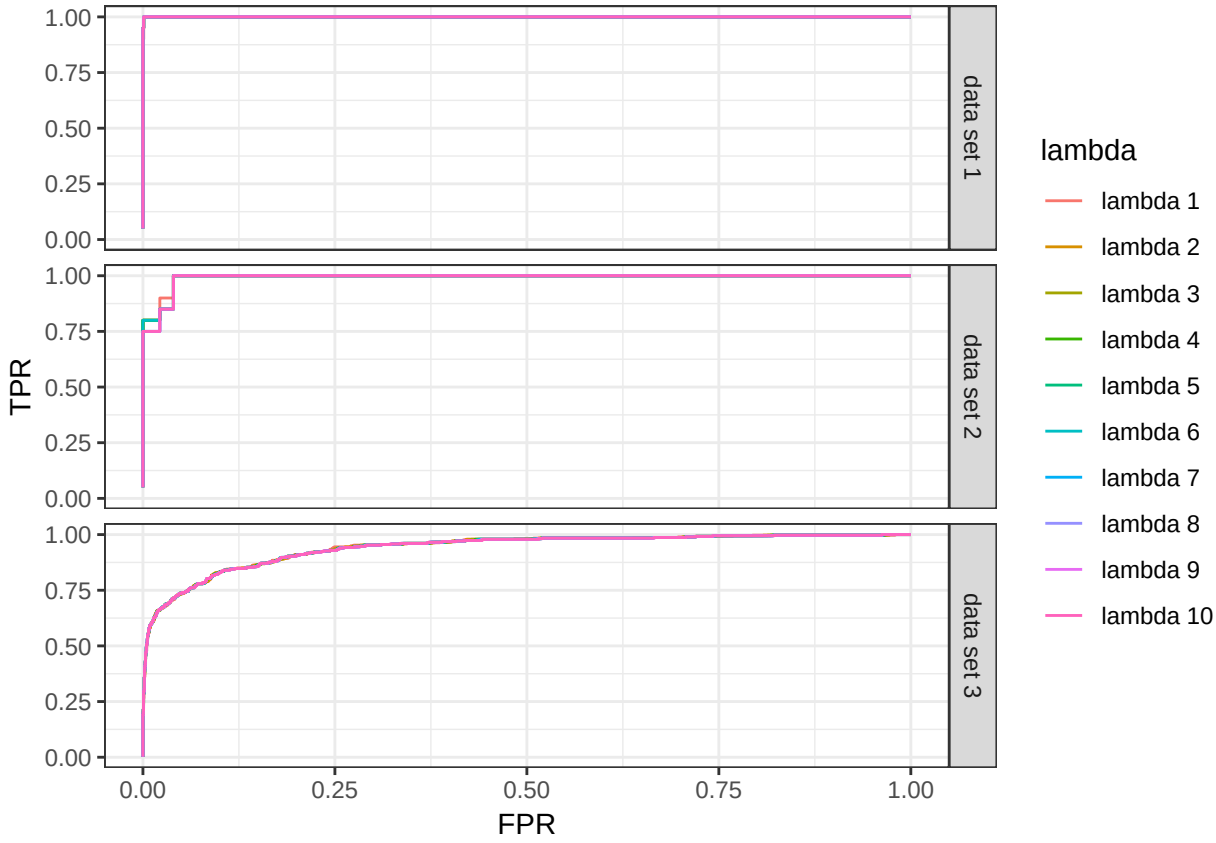
```
dat2 <- sim$data$dat2[,which(sim$data$dat2 %>% colSums()!=0)]
sim$data$dat2 <- dat2
c1.grid <- matrix(rep(seq(from=0.1, to=1, length=10), each=3), ncol=3, byrow=TRUE)
RGCCAresults_k <- apply(c1.grid,  1, IntMultiOmics, method="RGCCA",
                    data=sim$data, K=4, C=1-diag(length(sim$data)), ncomp=rep(3,3), scheme="centroid")
```

ROC evaluation

```
auc_eval_RGCCA_k <- sapply (RGCCAresults_k, function(mm) {
  r <- roc_eval(truth= truth, fit = mm$fit, method = "RGCCA")
}, simplify = FALSE)
```

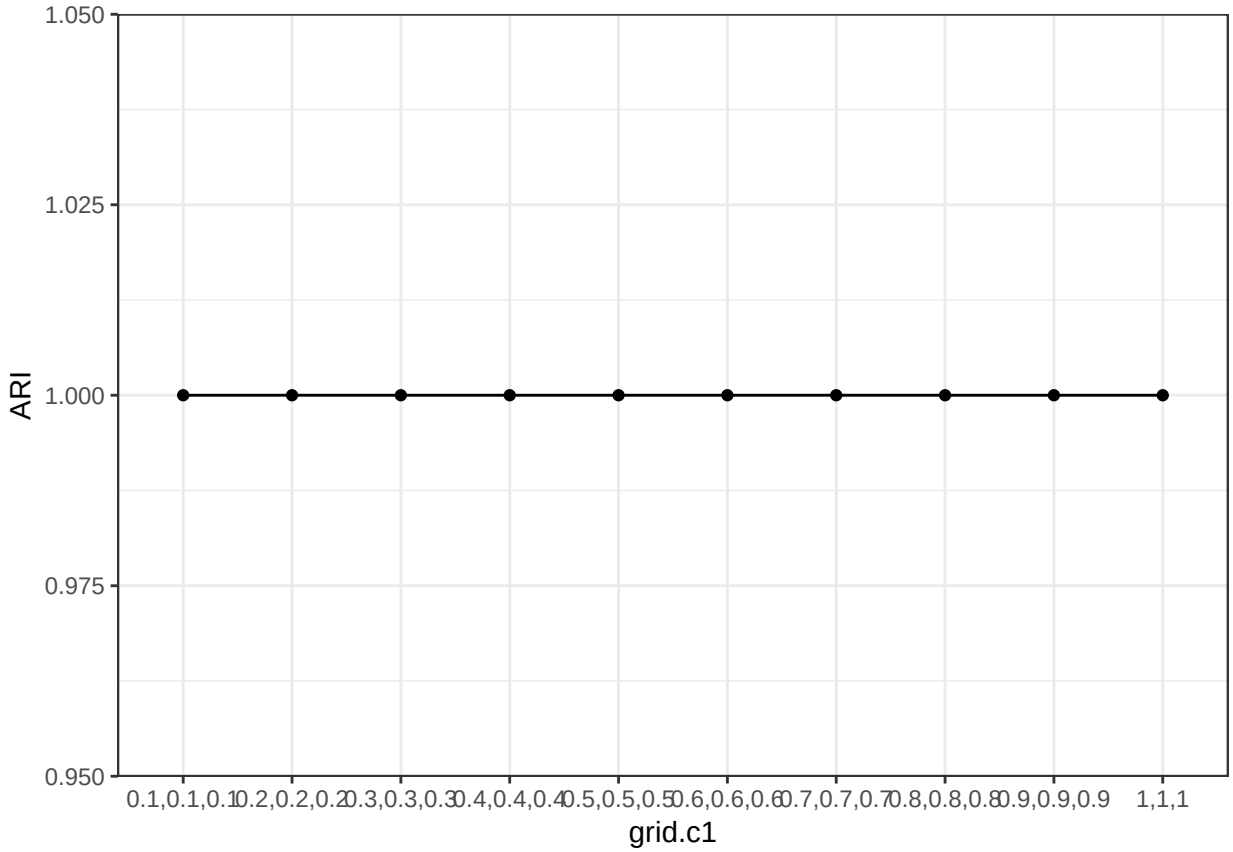
```
g_rgcca_k <- do.call(rbind, lapply(1:length(auc_eval_RGCCA_k), function (ss) {
  dd <- auc_eval_RGCCA_k[[ss]]
  n_by_data_set <- sapply(dd$TPR, length)
  tprs <- dd$TPR %>% unlist
  fprs <- dd$FPR %>% unlist
  data.frame(TPR=tprs, FPR=fprs,
             dataSet= sprintf("data set %s", rep(1:3, times=n_by_data_set)),
             lambda=sprintf("lambda %s", as.factor(ss)))
}))
```

```
g_rgcca_k %>% ggplot(aes(x=FPR, y=TPR, color=lambda))+geom_line()+facet_grid(dataSet~.)+theme_bw()
```



ARI evaluation

```
ari_rgcca <- sapply (RGCCAresults_k, function(mm) {
  adjustedRIComputing(mm,sim$true.clust)
}, simplify = TRUE)

df_ari <- data.frame(grid.c1=apply(c1.grid, 1, paste,collapse=","), ARI=ari_rgcca)
df_ari %>% ggplot(aes(x=grid.c1, y=ARI, group=1))+geom_point()+geom_line()+theme_bw()
```



Conclusion

To conclude, the number of latent variables and the values of sparsity parameters influence the performance of the methods. A too small or a too high number of latent variables could lead to bad performance. We observe similar results with the values of the sparsity parameters.