Proposal

Zhuohui Liang/zl2974

Introduction

A biology phenotypes can be represent in a number of layers information: clinical(eletronic health record), methylation level, miRNA expression and etc. Some biological process might be identifiable in one type of biological data but not identifiable in another level.And thus multi-view integration is proposed to enhance the signal in different layers of data. Similarity Network Fusion(SNF)[**?**] are proposed to fuse information from different omics data via interactively fusing between similarity matrix and draw cluster result on the fused similarity matrix. Multiview Diversity Matrix put a independent constrain terms on the optimization forcing learned self-representation matrix contains complementary information from different view of data and cluster on the combined complementary similarity matrix. Multiview Non-negative Matrix Factorization is proposed to recursively optimize learned factor loads, factors and consensus cluster result to reach consensus result from different view.

However, besides SNF is build for biological data, all other methods come from computation vision, thus all has an underlying assumption that all data contains information of same number of clusters to achieve signal boosting. This assumption tends not to hold in biological data. The goal of this report is to test if multiview data integration method can boost clustering result in biological settings. In specific SNF will be compared with the other two methods with on different scenarios which one data only have information to separate some of the clusters but not all. Information from all data are need to find the true clusters. And also test the clustering performance of each methods when data contribute differently on clustering results.

Method

Similarity Network Fusion

Similarity Network based method use the similarity calcuate with some kernel apply to some distance measure. In this report, we will use gaussian kernel and eucludian distance.

$$P(i,j) = exp(\frac{dist(X_i - X_j)}{\sigma})$$

After we calculated the similarity matrix, one can perform spectural clustering on the similarity matrix to get clustering result. The spectural clustering question solves for a lower rank indicator matrix f, that $minTr(f(I-P)f^t)$, $s.t.\ f^tf = I$.

SNF instead of using a single similarity matrix, it first create another set of similarity matrix only keeping the k-nearest neighbour's edges.

$$S = knn(P)$$

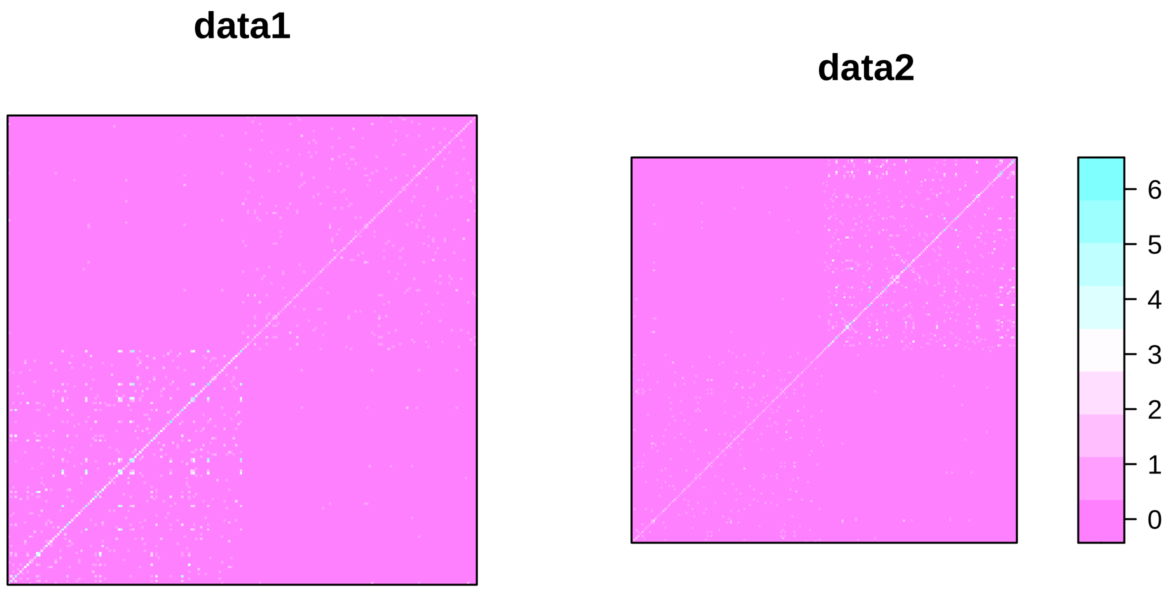
And use the local similarity matrix S of other data, iteratively updating current data's similarity matrix P, until converge.

$$P_1^{t+1} = S_2^t P_1^t S_2^t$$

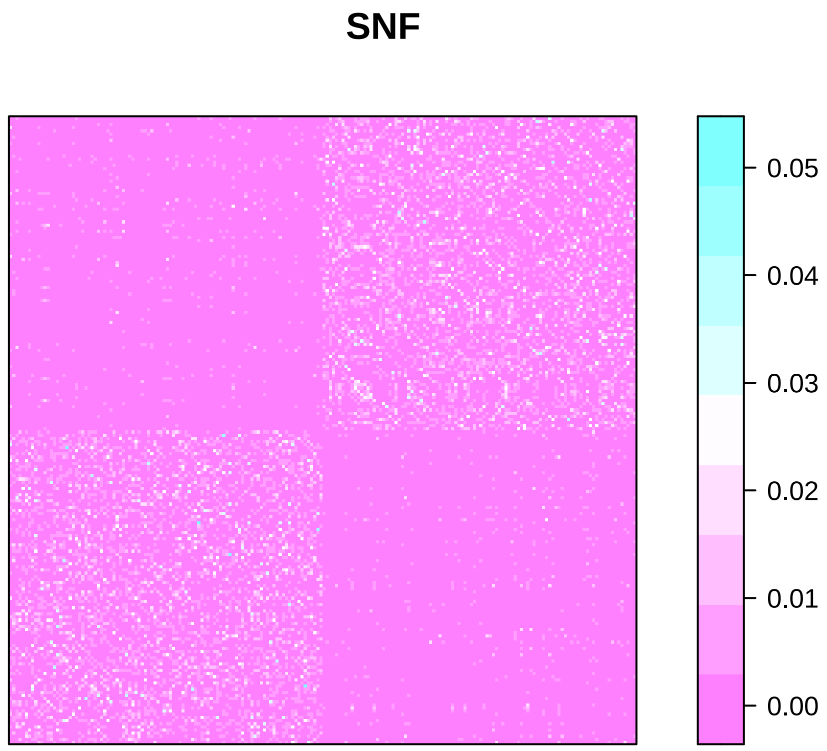
By iteratively updating, the infomation of other data fused together.

```
data("Data1")
data("Data2")

Data1 = as.matrix(Data1)
Data2 = as.matrix(Data2)
```

```r
s1 = affinityMatrix(as.matrix(dist(Data1))) %>% as.matrix()
s2 = affinityMatrix(as.matrix(dist(Data2))) %>% as.matrix()
snf = SNF(list(s1,s2)) %>% as.matrix()
diag(snf) = 0

g1 = levelplot(
  s1,
  main = "data1",
  xlab = NULL,
  ylab = NULL,
  cuts = 8,
  colorkey = FALSE,
  scales = list(x = list(at = NULL),
                y = list(at = NULL))
)
g2 = levelplot(
  s2,
  main = "data2",
  xlab = NULL,
  ylab = NULL,
  cuts = 8,
  colorkey = T,
  scales = list(x = list(at = NULL),
                y = list(at = NULL))
)
g3 = levelplot(
  snf,
  main = "SNF",
  xlab = NULL,
  ylab = NULL,
  cuts = 8,
  colorkey = T,
  scales = list(x = list(at = NULL),
                y = list(at = NULL))
)

grid.arrange(g1,g2,ncol = 2)
```



```
plot(g3)
```



DIMSC

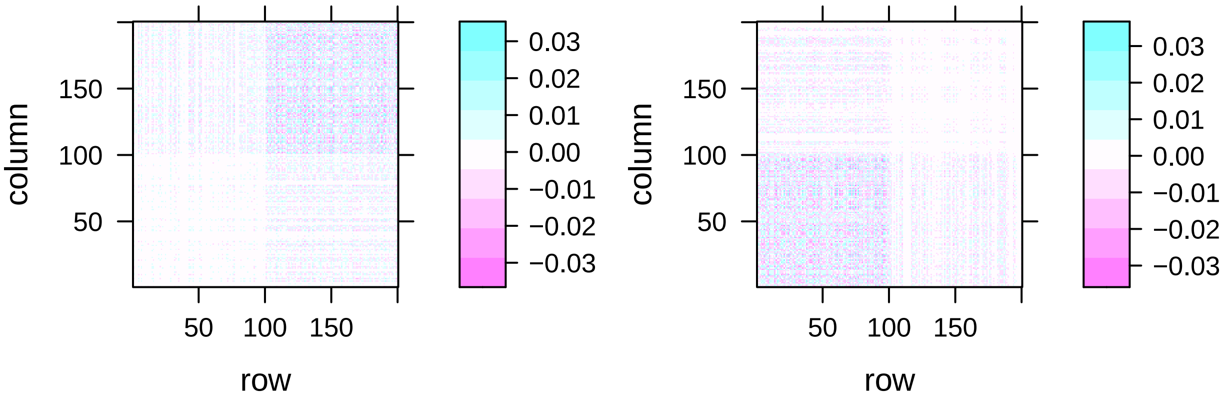
Diversity Induce Mulitview Subspace clustering also perform clustering from similarity matrix. It use so-call “self-expressiveness”

$$||X - XZ||$$

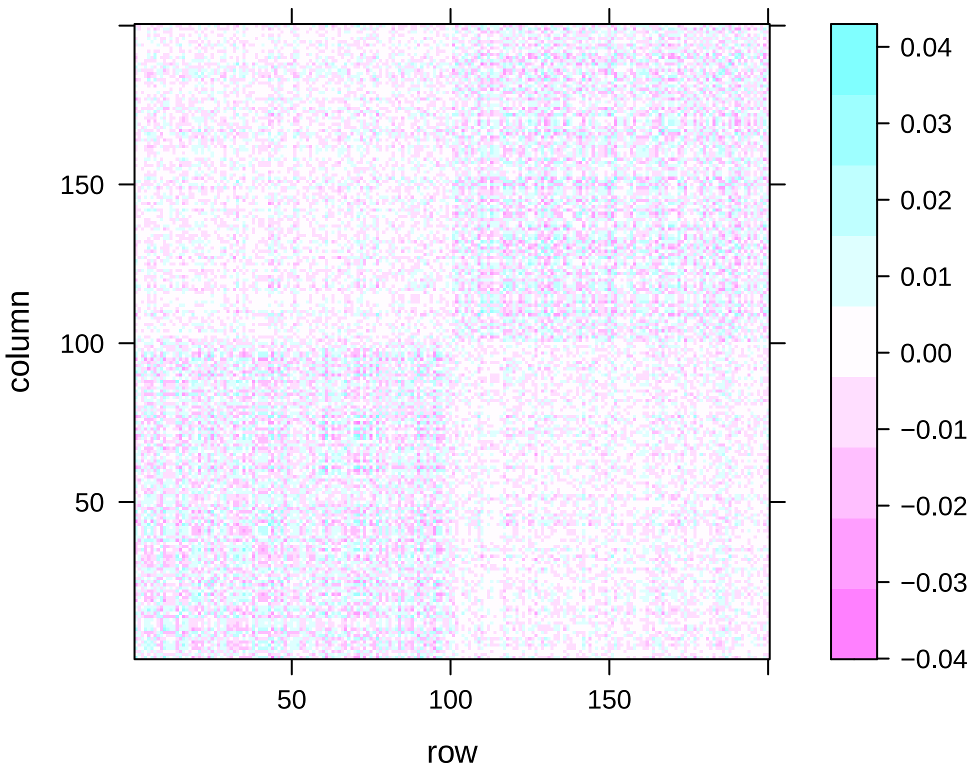
to calculate the similarity matrix Z, and perform spectural clustring. In DIMSC, the algorithm simultaneously updating similarity matrix of each data, and also introduce a penalty term penalizes correlation between the similarity from different data view. Such that each similarity matrix learn some information not in the other data. And suming all similarity matrix in the end as a single similarity matrix for other task.

$$argmin_Z \sum_v ||X^v - X^v Z^v|| + \alpha Tr(Z^v L^v Z^v) + \beta \text{Ind}(Z^1, Z^2, ...Z^v)$$

```
a = DiMSC(list(Data1,Data2),w_function = cor,K=2)
grid.arrange(levelplot(a$Z[[1]],cuts=8),
             levelplot(a$Z[[2]],cuts=8),ncol=2)
```



```
levelplot(a$A,cuts=8)
```



PMSC

Partition level multiview subspace clustering is similar to above methods and solve for following loss fuction.

$$argmin_{Z,F,Y} \sum_v ||X^v - X^v Z^v|| + \alpha Tr(F^v Z^v F^{v^T}) + \beta ||YY^T - F^v F^{v^T}|| + \gamma Sparsity$$

But instead of perform clustering on a single similarity in the end, the method solve for clustering result of each data types during updating and also introduces a common clustering result Y and use the clustering result to help learn similarity matrix. By iteratively updating similarity matrix, clustering of each data types and the common clustering result, the Data information is integrated together.

Simulation

Scenario I This is a simple simulation to demonstrate how the above mention data integration method works. In this scenario, we have two data, each data have 3 clusters and each cluster has 100 subjects. Each subjects has 500 features, and first 12 features are signal to separate subjects from different clusters. First 4 features from data 1 are sample from N(1,2) for subjects from cluster 1 and N(0,2) for cluster 2,3. Next 4 features cluster 2 from data 1 are sample from N(1,2) and cluster 1,3 are sample from N(0,2) and etc. such that data using the first 4 features, we can separate cluster 1 from 2 and 3 and using next 4 to separate cluster 2 from 3 and so on. The rest features are sample from N(0,2) as noise. Data 2 are generated with similar framework and the same effect size.

We compare K-means with SNF, DiMSC and PMSC.

scenario II Three data are generated to represent 4 clusters, in each cluster we have 100 subjects and each subject records 100 features but only the first 10 are signal features that provide information to separate different clusters and the rest of the features are noise sample from N(0,1). For data 1, the signal features for cluster 1 are sample from N(1,1) and the other clusters from N(0,1), such that data 1 have 2 separable clusters and cluster 1 can be separated from 2,3 and 4. The other data follow similar design such that data 2 can separate cluster 2 from 1,2,4 and data 3 separate cluster 3 from 1,2 and 4.

In the second settings, we follow the same design but set all the noise feature to 990 decrease the signal to noise ratio.

I will compare k-means with data concatenation against SNF with spectral clusting and DIMSC with spectral clustering

Data

TCGA's Kidney Renal Cell Carcinoma subset including 291 tumor samples. [Net16] has found experiment result of 2 subtypes of Renal Cell Carcinoma that related to patients survival. We will use patient survival to see the clustering result of the comparing models. In this report, to integrate data to boost signal, we include: Methylation at Gene level(Illumina HM450K platform), gene mutation(Somatic mutations at gene level SNV) and RNAseq(HiSeq platform) are selected for the study for both types of RCC. Missing observation in the methylation data is assumed to be Missing-at-random and use K-nearest-neighbour method for imputation.

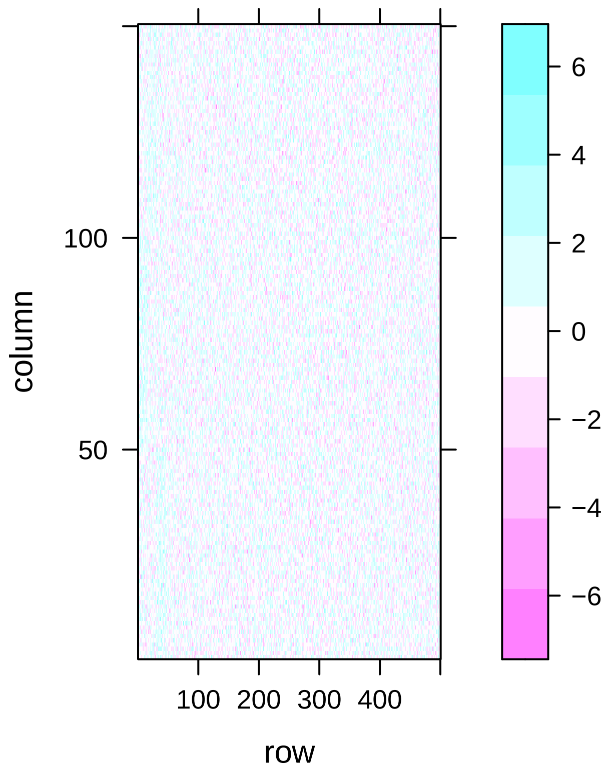
Result

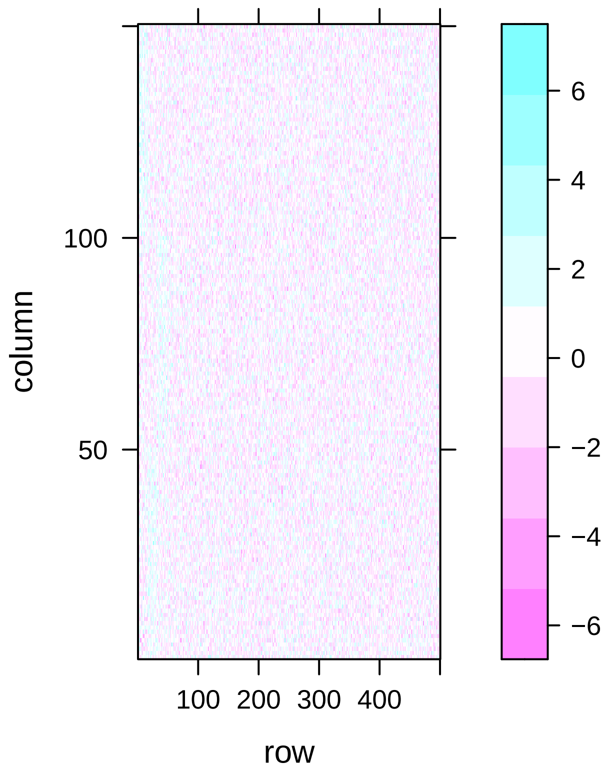
Simulation

Scenario I

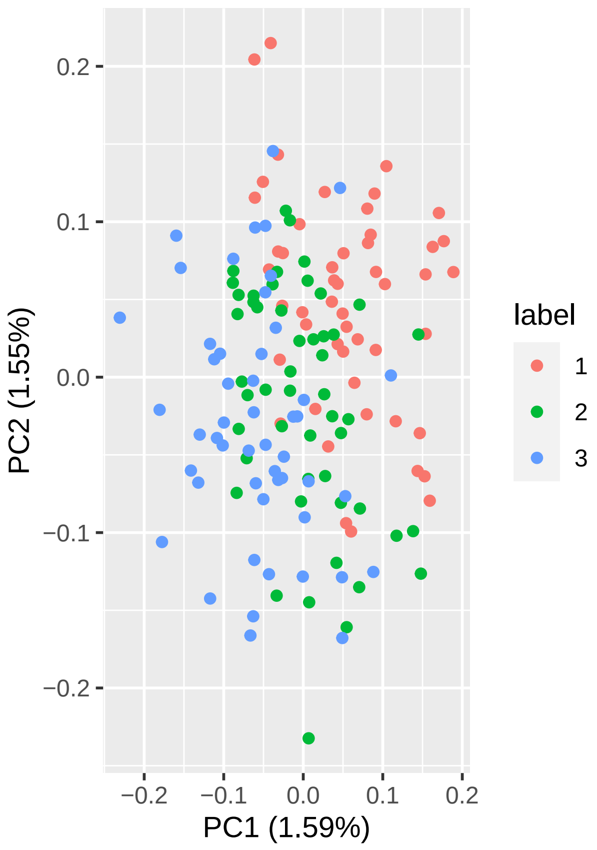
We deplay the data designed and the simulation result.

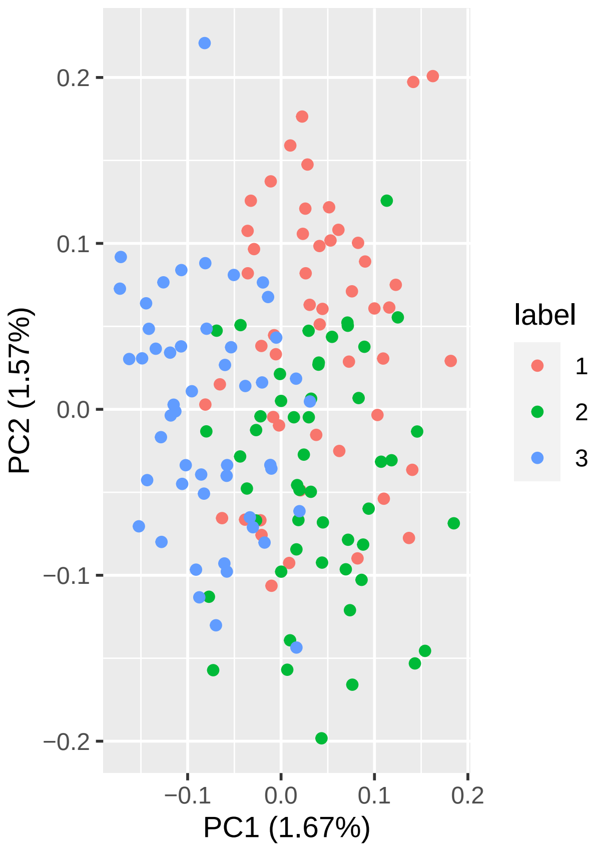
```
s1 = rbind(
  simulate_data(c(0, 0, 1), n_var = 48,sigma = 2),
  simulate_data(c(1, 0, 0), n_var = 48,sigma = 2),
  simulate_data(c(0, 1, 0), n_var = 48,sigma = 2)
)


s2 = rbind(
  simulate_data(c(0, 1, 0), n_var = 48,sigma = 2),
  simulate_data(c(0, 0, 1), n_var = 48,sigma = 2),
  simulate_data(c(1, 0, 0), n_var = 48,sigma = 2)
)

data_list = list(s1, s2)

data_list = lapply(data_list, function(x) add_noise(x,500-48,2))

true_label = as.factor(sort(rep(1:3, 50)))

do.call(grid.arrange, c(lapply(data_list, function(x)
  levelplot(t(x), aspect = "fill",cuts = 8)), ncol = 2))
```





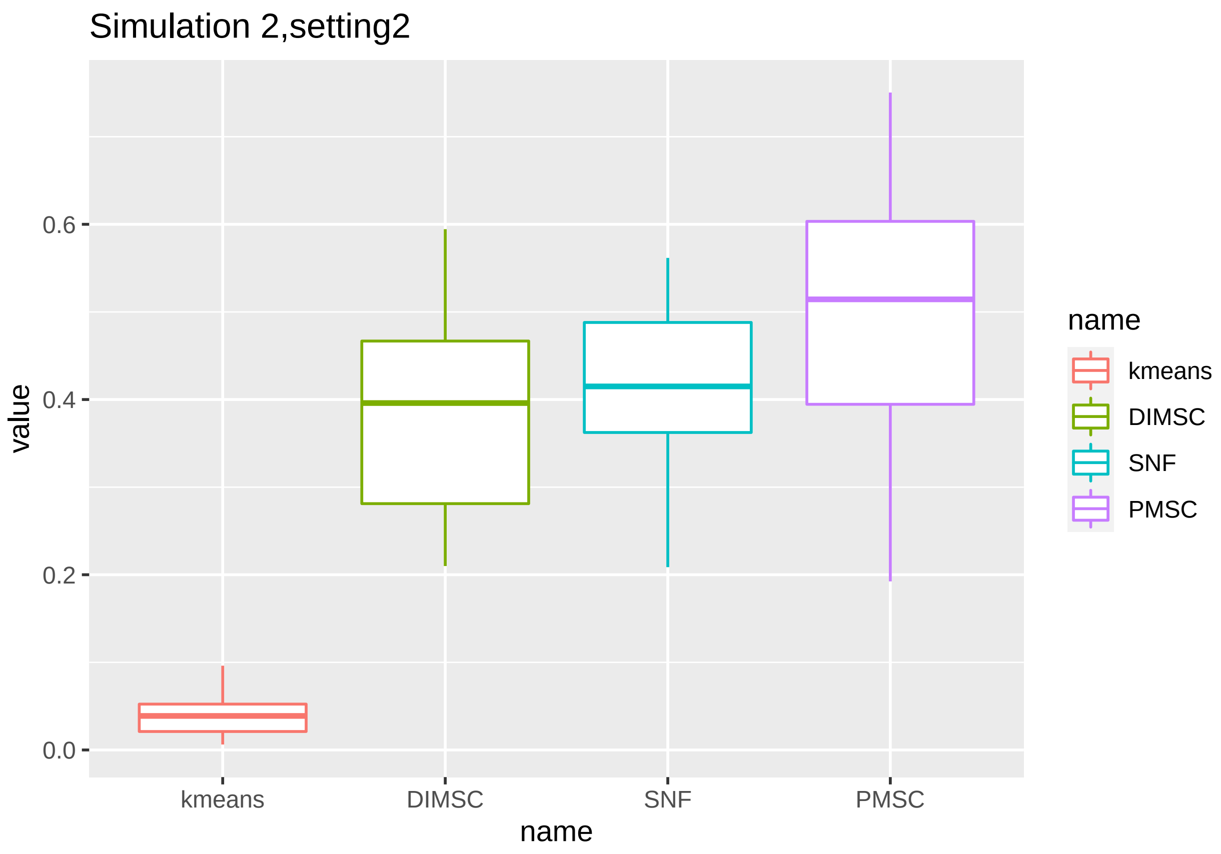
```
do.call(grid.arrange, c(lapply(data_list, function(x) {
  x = as_tibble(x)
  x$label = true_label
  autoplot(prcomp(x[,-ncol(x)]), data = x, colour = "label")
}), ncol = 2))
```





```r
set.seed(123123)

# pretrain_1 = SNF_tuning(data_list,K=3)
# pretrain_1$NMI = sapply(pretrain_1$cluster,function(x) calNMI(true_label,x))
# pretrain_1[which.max(pretrain_1$NMI),]

sim_1 =
  foreach(i = 1:20,
          .combine = "rbind",
          .packages = c("SNFtool","tidyverse")) %dopar% {

            s1 = rbind(
              simulate_data(c(0, 0, 1), n_var = 48, sigma = 2),
              simulate_data(c(1, 0, 0), n_var = 48, sigma = 2),
              simulate_data(c(0, 1, 0), n_var = 48, sigma = 2)
            )


            s2 = rbind(
              simulate_data(c(0, 1, 0), n_var = 48, sigma = 2),
              simulate_data(c(0, 0, 1), n_var = 48, sigma = 2),
              simulate_data(c(1, 0, 0), n_var = 48, sigma = 2)
            )

            data_list = list(s1, s2)

            data_list = lapply(data_list, function(x)
              add_noise(x, 500 - 48, 2))
            result_kmeans = kmeans(do.call(cbind, data_list), 3,
                                   algorithm = "MacQueen")$cluster

            result_snf = spectralClustering(SNF(lapply(data_list, function(x) {
              affinityMatrix(as.matrix(dist(x)),
                             K = 10,
                             sigma = 0.45)
            }),
            K = 20,
            t = 20),
            K = 3)

            result_PMSC = PMSC(
              data_list,
              K = 3,
              alpha = 100,
              beta = 0.01,
              gamma = 0.001
            )$Y

            result_DIMSC = DiMSC(data_list,3,w_function = cor)$cluster

            tibble(
              kmeans =calNMI(true_label, result_kmeans),
              SNF = calNMI(true_label, result_snf),
              PMSC = calNMI(true_label, result_PMSC),
              DIMSC = calNMI(true_label,result_DIMSC)
            )
          }
```

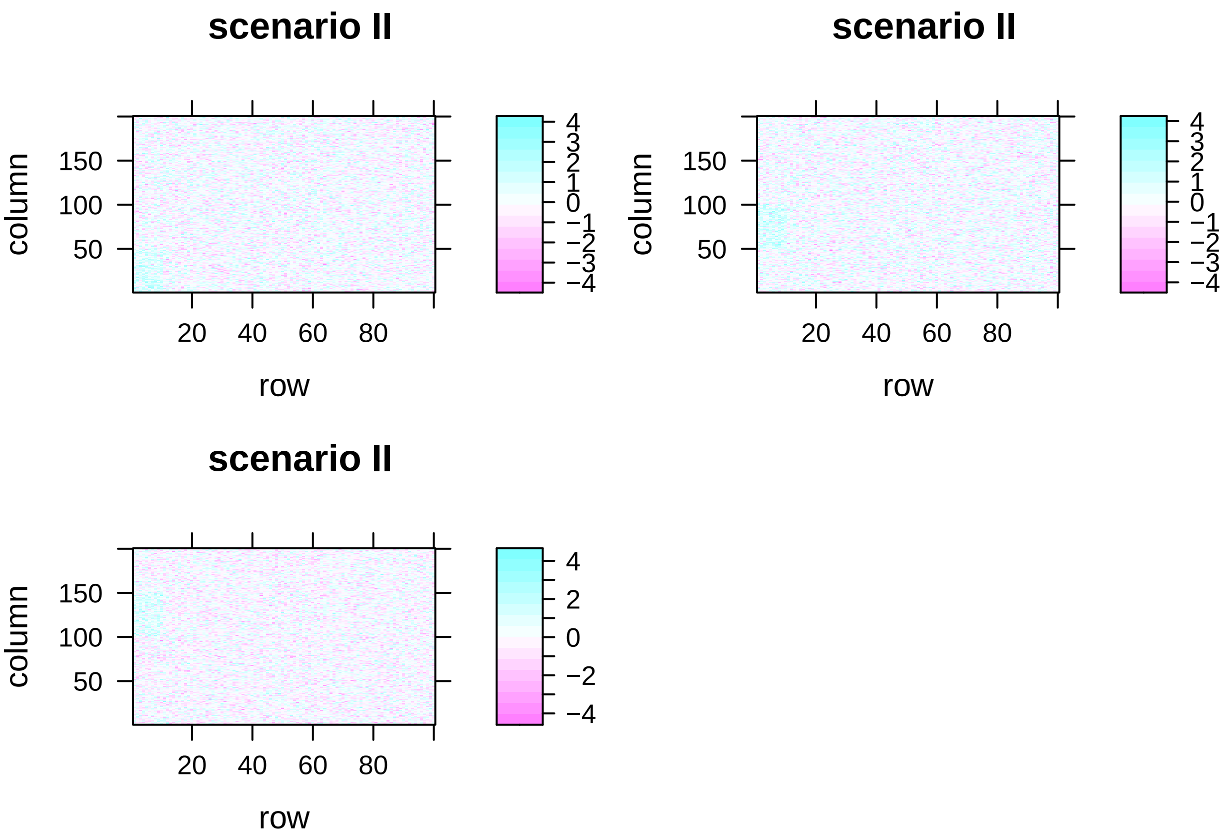
```
sim_1 %>%
  pivot_longer(everything()) %>%
  mutate(name = forcats::fct_reorder(name,value)) %>%
  ggplot(aes(x = name,y = value,colour = name))+
  geom_boxplot()+
  labs(title = "Simulation 2,setting2")
```



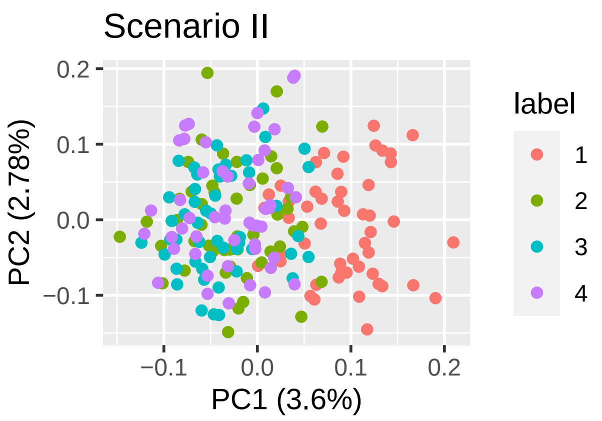
From simulation result, we can see that in this settings, data integration methods are better than the data concatenation and k-means method. PMSC is outperform all other methods.

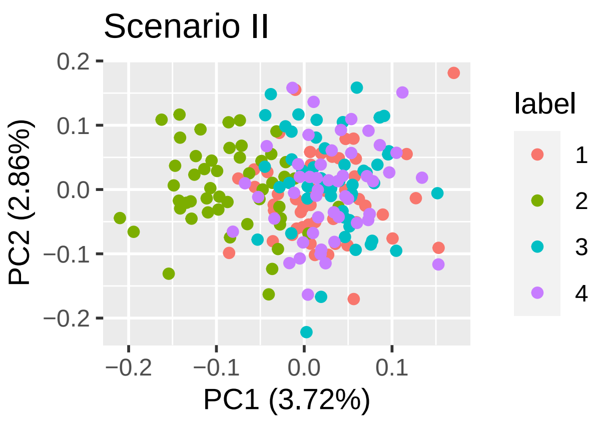
Scenario II

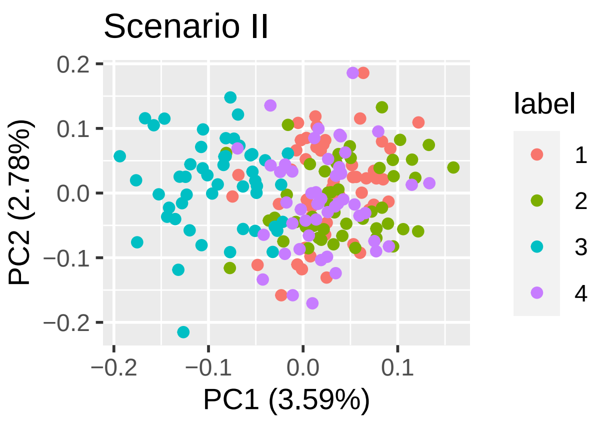
```
s1 = rbind(
  simulate_data(c(1), n_var = 10),
  simulate_data(c(0), n_var = 10),
  simulate_data(c(0), n_var = 10),
  simulate_data(c(0), n_var = 10)
)


s2 = rbind(
  simulate_data(c(0), n_var = 10),
  simulate_data(c(1), n_var = 10),
  simulate_data(c(0), n_var = 10),
  simulate_data(c(0), n_var = 10)
)


s3 = rbind(
  simulate_data(c(0), n_var = 10),
  simulate_data(c(0), n_var = 10),
  simulate_data(c(1), n_var = 10),
  simulate_data(c(0), n_var = 10)
)

data_list = list(s1, s2, s3)
data_list = lapply(data_list, function(x) add_noise(x,90))

true_label = as.factor(sort(rep(1:4, 50)))

do.call(grid.arrange, c(lapply(data_list, function(x)
  levelplot(t(x), aspect = "fill",main = "scenario II")), ncol = 2))
```



```
do.call(grid.arrange, c(lapply(data_list, function(x) {
  x = as_tibble(x)
  x$label = true_label
  autoplot(prcomp(x[,-ncol(x)]), data = x, colour = "label",main = "Scenario II")
}), ncol = 2))
```

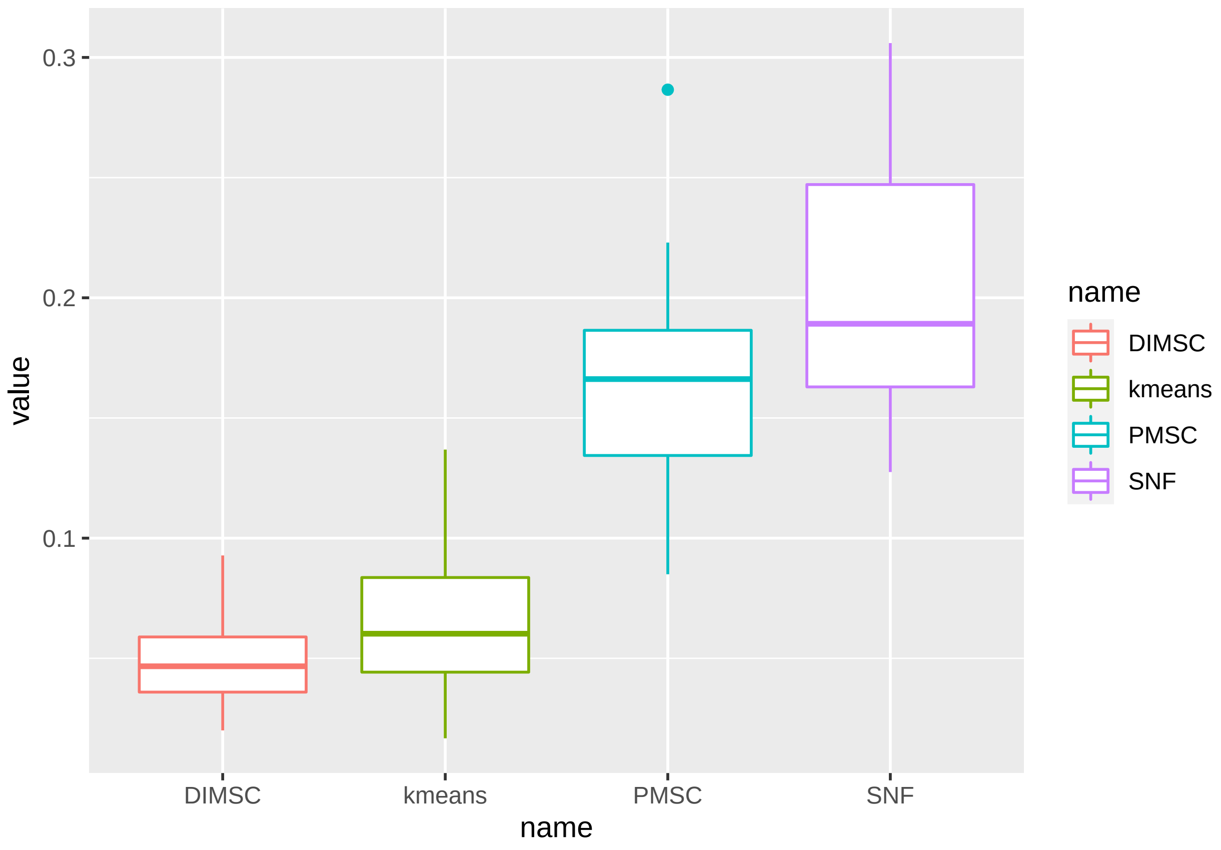






```
sim_2 = foreach(
  i = 1:20,
  .combine = "rbind",
  .packages = c("SNFtool", "tidyverse")
) %dopar% {

  s1 = rbind(
    simulate_data(c(1), n_var = 10),
    simulate_data(c(0), n_var = 10),
    simulate_data(c(0), n_var = 10),
    simulate_data(c(0), n_var = 10)
  )


  s2 = rbind(
    simulate_data(c(0), n_var = 10),
    simulate_data(c(1), n_var = 10),
    simulate_data(c(0), n_var = 10),
    simulate_data(c(0), n_var = 10)
  )


  s3 = rbind(
    simulate_data(c(0), n_var = 10),
    simulate_data(c(0), n_var = 10),
    simulate_data(c(1), n_var = 10),
    simulate_data(c(0), n_var = 10)
  )

  data_list = list(s1, s2, s3)
  data_list = lapply(data_list, function(x)
    add_noise(x, 90))

  true_label = as.factor(sort(rep(1:4, 50)))

  result_kmeans = kmeans(do.call(cbind, data_list), 4,
                         algorithm = "MacQueen")$cluster

  result_snf = spectralClustering(SNF(lapply(data_list, function(x) {
    affinityMatrix(as.matrix(dist(x)),
                   K = 10,
                   sigma = 0.65)
  }),
  K = 20,
  t = 20),
  K = 4)

  result_PMSC = PMSC(
    data_list,
    K = 4,
    alpha = 100,
    beta = 0.01,
    gamma = 0.001
  )$Y

  result_DIMSC = DiMSC(data_list, 4, w_function = cor)$cluster

  tibble(
    kmeans = calNMI(true_label, result_kmeans),
    SNF = calNMI(true_label, result_snf),
    PMSC = calNMI(true_label, result_PMSC),
    DIMSC = calNMI(true_label, result_DIMSC)
  )
}
```

```
sim_2 %>%
  pivot_longer(everything()) %>%
  mutate(name = forcats::fct_reorder(name,value)) %>%
  ggplot(aes(x = name,y = value,colour = name))+
  geom_boxplot()
```

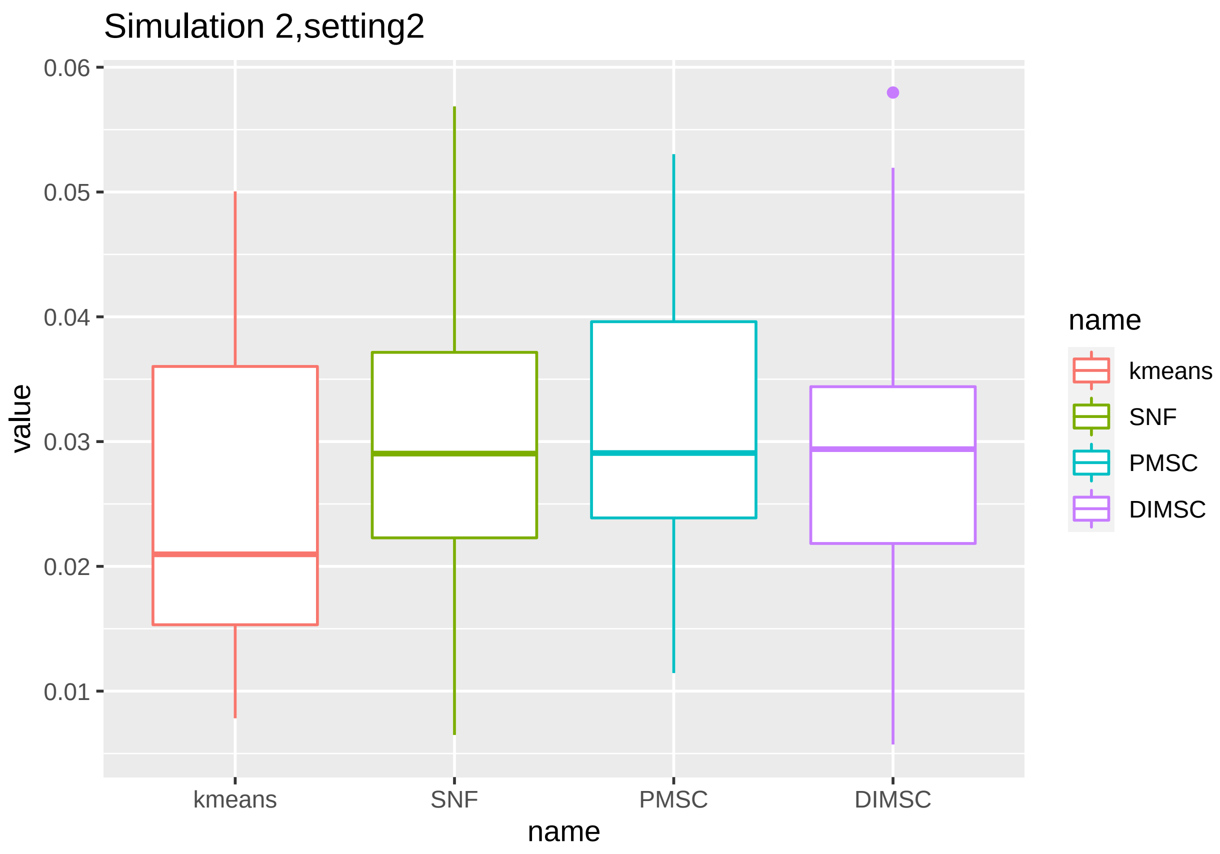


In the second simultion, we can see that the data fit our design, and comparing these methods, we can see that both data integration methods designs for computer vision fail in this specific settings. but SNF and PMSC still out-perform data concatenation and k-means. But as we reduce the signal to noise ratio, although they still slightly out-perform kmeans, but the overall NMI is negligible to consider.

```
sim_3 = foreach(
  i = 1:20,
  .combine = "rbind",
  .packages = c("SNFtool", "tidyverse")
) %dopar% {

  s1 = rbind(
    simulate_data(c(1), n_var = 10),
    simulate_data(c(0), n_var = 10),
    simulate_data(c(0), n_var = 10),
    simulate_data(c(0), n_var = 10)
  )


  s2 = rbind(
    simulate_data(c(0), n_var = 10),
    simulate_data(c(1), n_var = 10),
    simulate_data(c(0), n_var = 10),
    simulate_data(c(0), n_var = 10)
  )


  s3 = rbind(
```

```r
    simulate_data(c(0), n_var = 10),
    simulate_data(c(0), n_var = 10),
    simulate_data(c(1), n_var = 10),
    simulate_data(c(0), n_var = 10)
  )

  data_list = list(s1, s2, s3)

  data_list = lapply(data_list, function(x)
    add_noise(x, 990))

  true_label = as.factor(sort(rep(1:4, 50)))

  result_kmeans = kmeans(do.call(cbind, data_list), 4,
                         algorithm = "MacQueen")$cluster

  result_snf = spectralClustering(SNF(lapply(data_list, function(x) {
    affinityMatrix(as.matrix(dist(x)),
                   K = 50,
                   sigma = 0.8)
  }),
  K = 20,
  t = 20),
  K = 4)

  result_PMSC = PMSC(
    data_list,
    K = 4,
    alpha = 100,
    beta = 0.01,
    gamma = 0.001
  )$Y

  result_DIMSC = DiMSC(data_list, 4, w_function = cor)$cluster

  tibble(
    kmeans = calNMI(true_label, result_kmeans),
    SNF = calNMI(true_label, result_snf),
    PMSC = calNMI(true_label, result_PMSC),
    DIMSC = calNMI(true_label, result_DIMSC)
  )
}
```

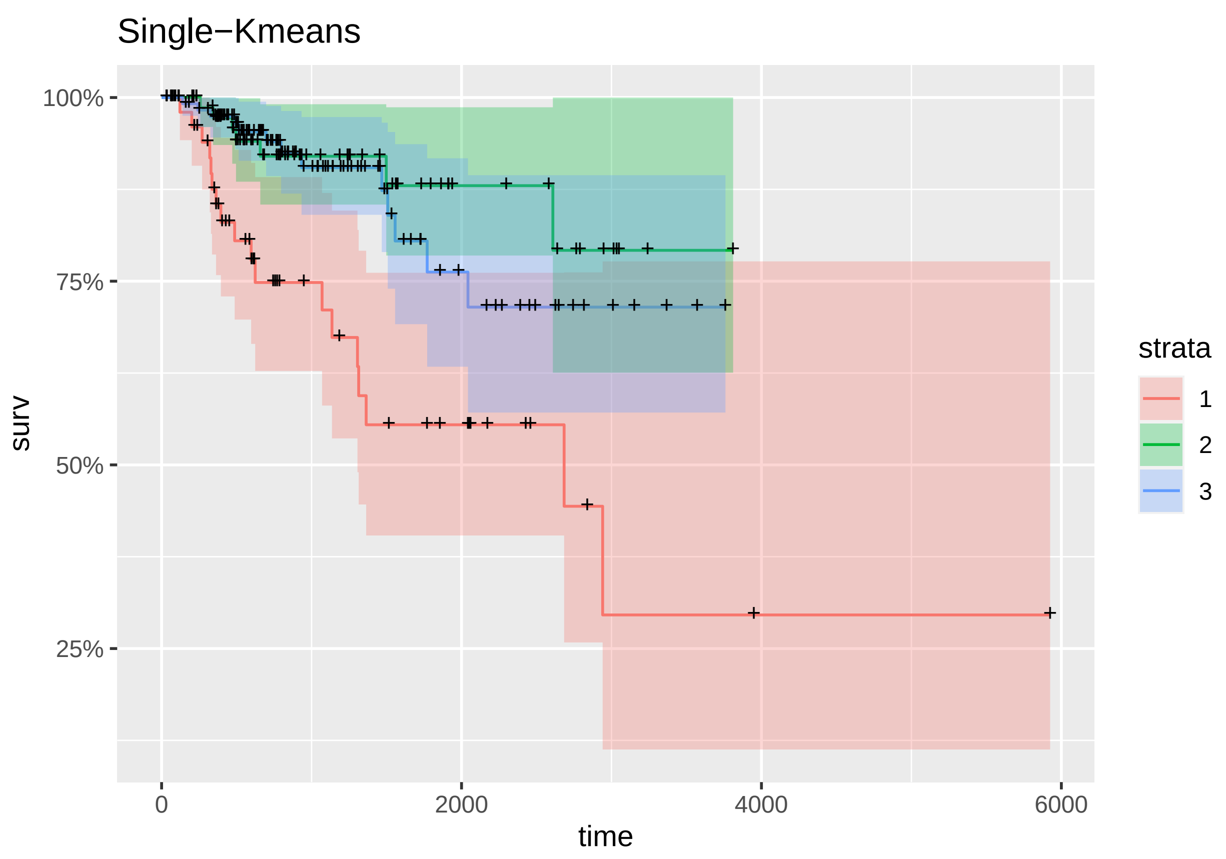
```r
sim_3 %>%
  pivot_longer(everything()) %>%
  mutate(name = forcats::fct_reorder(name,value)) %>%
  ggplot(aes(x = name,y = value,colour = name))+
  geom_boxplot()+
  labs(title = "Simulation 2,setting2")
```



Settings 2

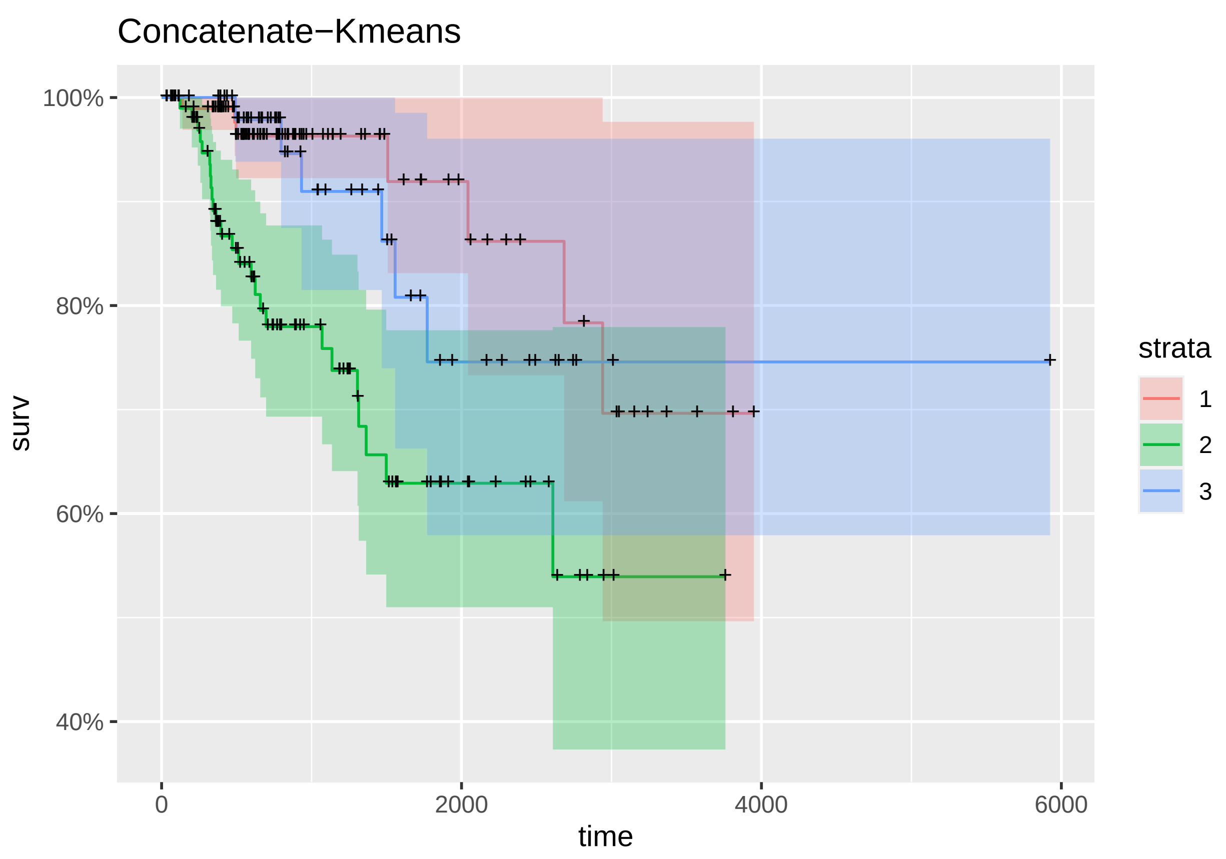
TCGA result

```
result_kmeans = kmeans(mirna, 3, algorithm = "MacQueen")$cluster

survival$cluster = result_kmeans

autoplot(survfit(Surv(time, event) ~ cluster, data = survival), main = "Single-Kmeans")
```



```
sur_skmean = survdiff(Surv(time, event) ~ cluster, data = survival)
```

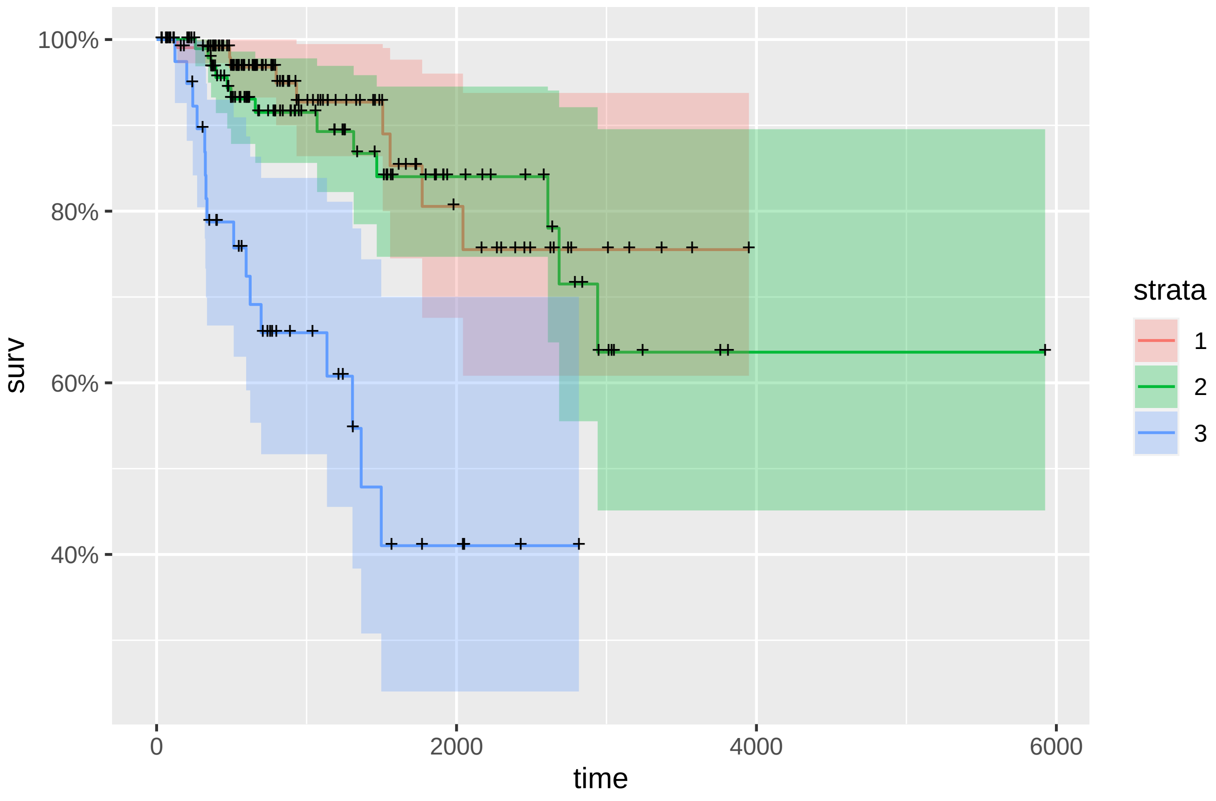
```
result_kmeans = kmeans(cbind(mirna, methylation, rnaseq, mutation), 3, algorithm = "MacQueen")$cluster

survival$cluster = result_kmeans

autoplot(survfit(Surv(time, event) ~ cluster, data = survival), main = "Concatenate-Kmeans")
```



```
sur_ckmean = survdiff(Surv(time, event) ~ cluster, data = survival)
```

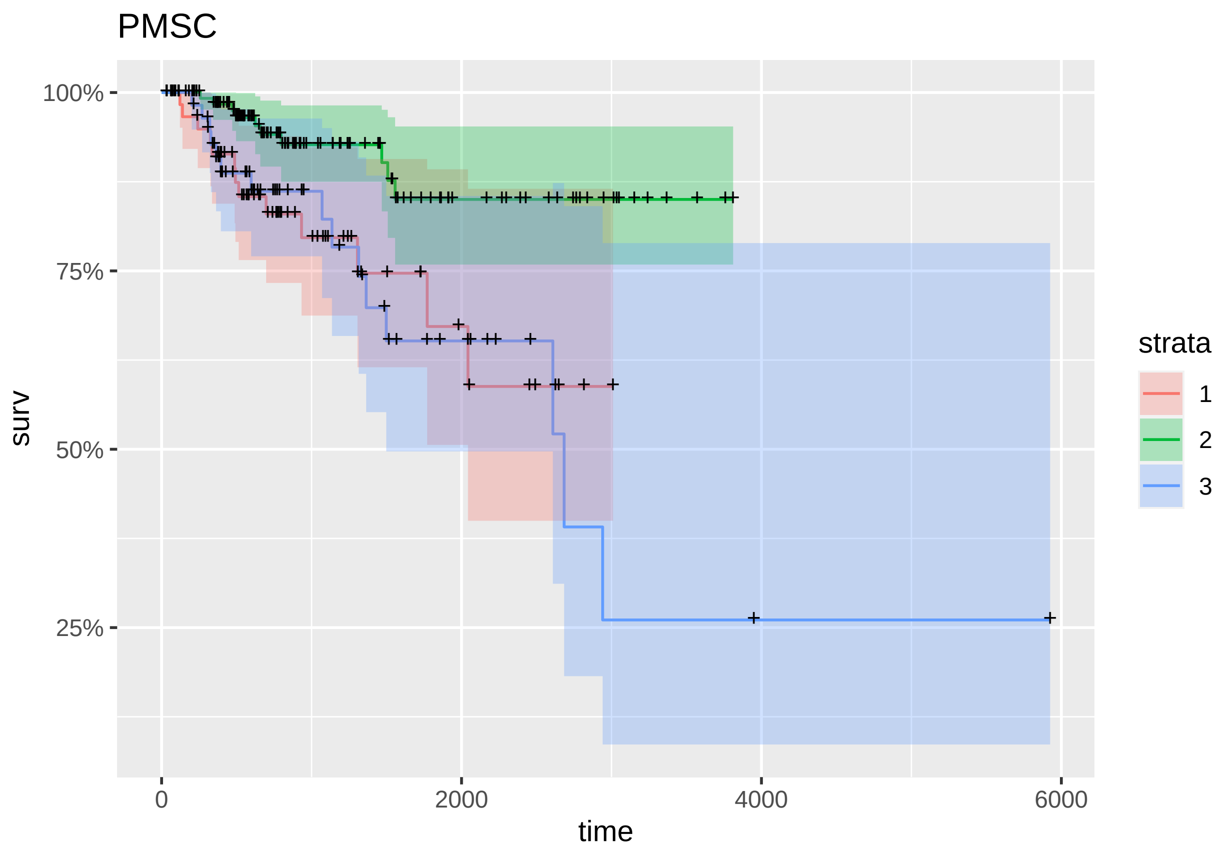
```
affinity_list = lapply(list(mirna, methylation, rnaseq,mutation),
                       function(x) {
                         affinityMatrix(as.matrix(dist(x)),
                                        K = 30,
                                        sigma = 0.5)
                       })

# plot(1:20,eigen(normalize(SNF(affinity_list,K = 20)))$val[1:20])

result_snf = spectralClustering(SNF(affinity_list,K = 30),3)

survival$cluster = result_snf

autoplot(survfit(Surv(time,event)~cluster,data = survival),main = "SNF")
```

SNF



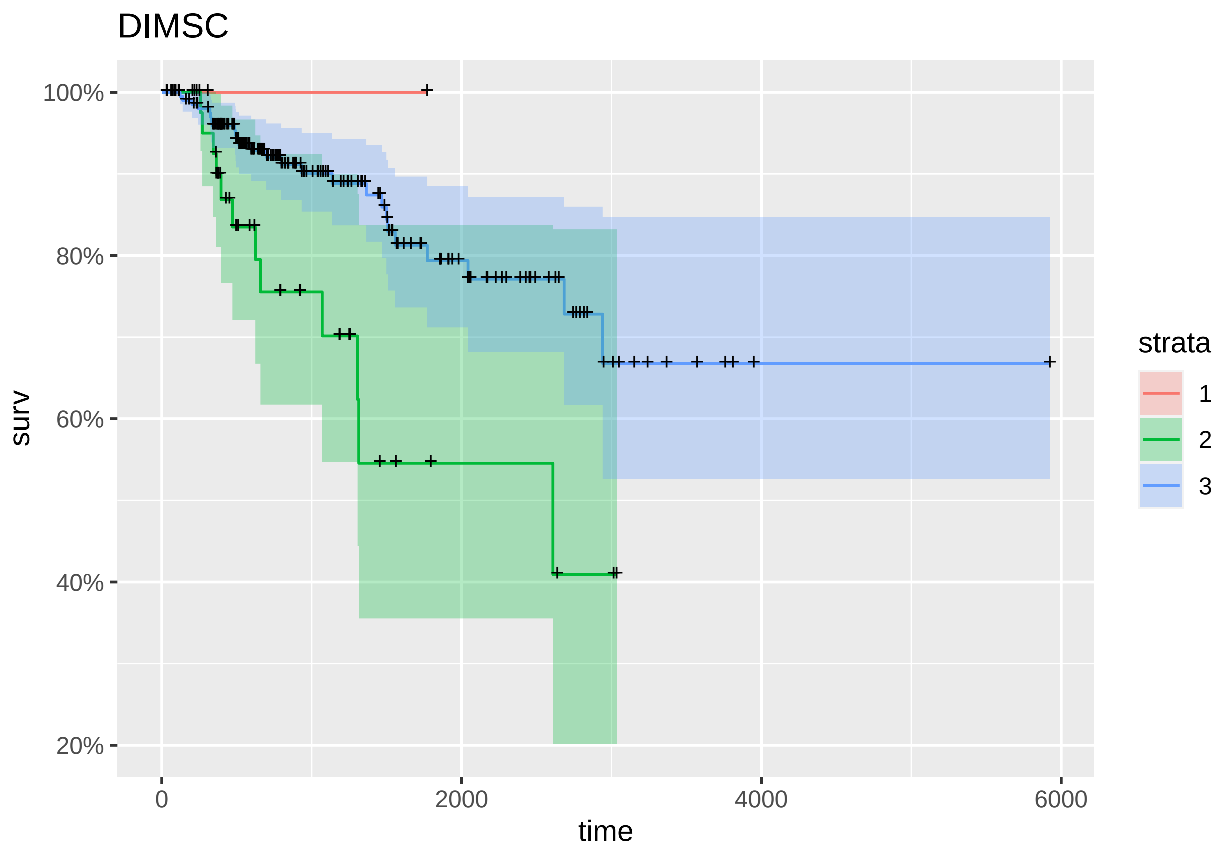
```
sur_SNF = survdiff(Surv(time, event) ~ cluster, data = survival)
```

```
result_PMSC = PMSC(list(mirna,methylation,rnaseq),K=3,beta = 0.01,.max_iter = 20)$Y

survival$cluster = result_PMSC

autoplot(survfit(Surv(time,event)~cluster,data = survival),main = "PMSC")
```



```
sur_PMSC = survdiff(Surv(time, event) ~ cluster, data = survival)
```

```
result_DIMSC = DiMSC(list(mirna,mutation,rnaseq,methylation),K=3)$cluster

survival$cluster = result_DIMSC

autoplot(survfit(Surv(time,event)~cluster,data = survival),main = "DIMSC")
```



```
sur_DIMSC = survdiff(Surv(time, event) ~ cluster, data = survival)
```

```
tibble(
  method = c("single kmean", "concatanate kmean", "SNF", "DISMC", "PMSC"),
  log_rank_test = lapply(list(sur_skmean, sur_ckmean, sur_SNF, sur_DIMSC, sur_PMSC), function(x)
    x$chis)
) %>%
  knitr::kable(caption = "Log-Rank test for survival trend")
```

Table 1: Log-Rank test for survival trend

method	log_rank_test
single kmean	17.78007
concatanate kmean	15.03928
SNF	30.78584
DISMC	9.954825
PMSC	13.41339

We choose 3 subtypes based on the maximum eigen gap of the similarity matrix of all 3 data set, and compare the survival of all methods clustering patient into 3 clusters against selecting using only miRNA to perform the clusters.

We can see that SNF clusters patients into 2 group has better survival and 1 group with worse outcomes. SNF has a more statistically significant survival difference than single kmeans and concatenate-data kmeans.

DISMC perform poorly in identifing the survival function. PMSC cluster 2 group has worse outcome and 1 group with better outcome.

Discussion

Data integration can boost signal when the signal is weak in single data types. But we also see that K-mean with single data types can have similar performance when the signal is strong enough to separate all clusters. DIMSC have high performance in all data have similar data structure settings, but fails when this setting is change. We can see that this method without modification might not suitable for omics data's task.

References

[Net16] Cancer Genome Atlas Research Network. Comprehensive molecular characterization of papillary renal-cell carcinoma. *New England Journal of Medicine*, 374(2):135–145, 2016.