Paper_code

Ziyue Zheng

2025-07-22

Install and load the package

You need to load all the related packages as well.

```r
# devtools::install('../FSPmix')
library(tidyverse)
library(FSPmix)
library(caret)
library(patchwork)
```

Load the data

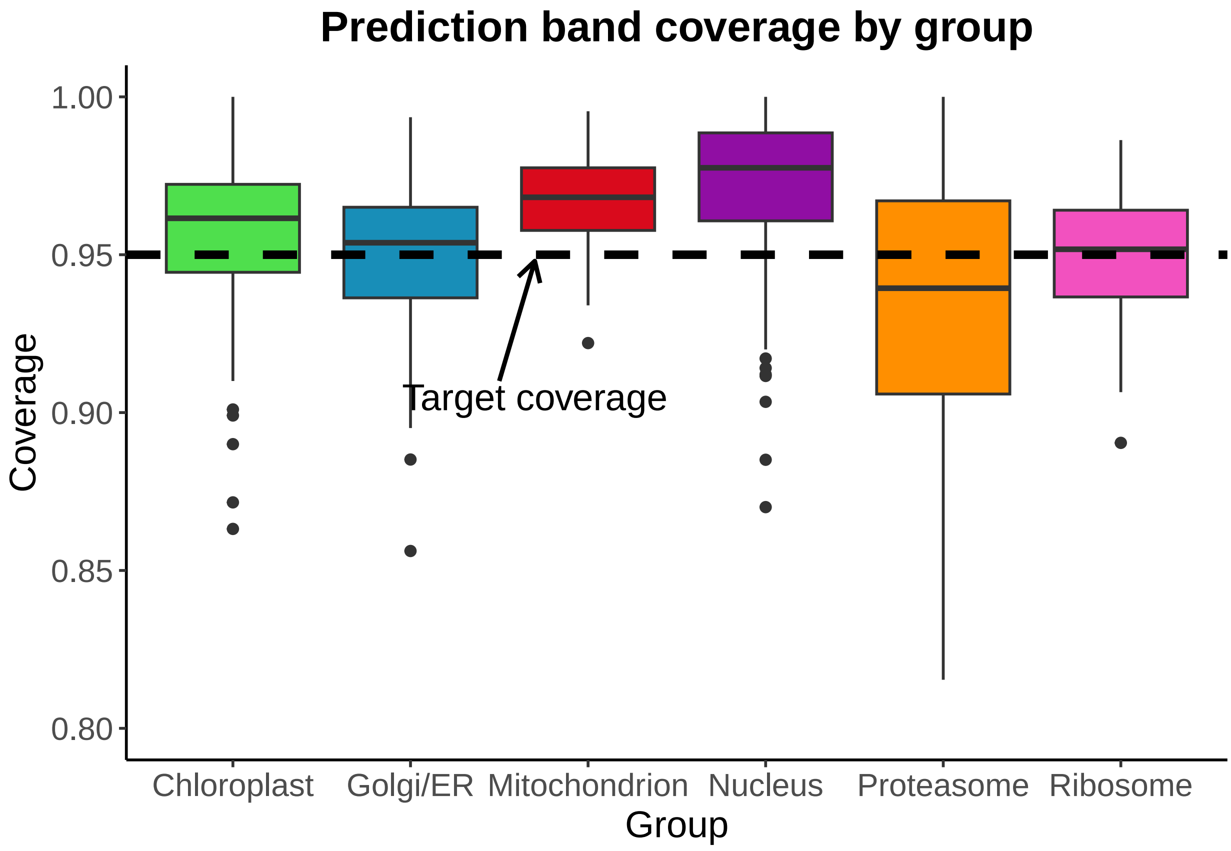
NOTE: The experimental dataset analyzed in this preprint is not publicly available at this time. The data will be made available upon journal publication. In the meantime, researchers who wish to access the data may contact Loay J. Jabre(ljabre@mta.ca). Once the raw data frame is read, *prepare_data* function will prepare the data into forms can be directly used by *fspmix* function.

```r
load("../data/Thaps2024.rda")
x <- prepare_data(Thaps2024)
data = x$data
label = x$labels
```

Code for Figure 2: Coverage of prediction bands

```r
library(caret)
# Step 1 Fit the full data set. Treat them as "True groups".
noK0_result = fspmix(data = data, label = label, num_clust = 6,
                     bandwidth = 1.5, nrep = 20)
data_matrix = data

exp_cov = matrix(0,ncol = 6, nrow = 100)
for(t in 1:100){
  classes <- prediction(noK0_result)$predict_label
  summary(factor(classes))
  # Step 2 Resample half of the data set. Fit the model again
  resample <- caret::createDataPartition(classes, p = 0.6, list = FALSE)
  labeled <- caret::createDataPartition(classes[resample], p = 0.2, list = FALSE)
  #how much labeled data should be
  mimic_lab <- data.frame(index = labeled,
                          label = classes[resample[labeled]])

  colnames(mimic_lab) <- c("index","label")
  # Step 3 Use the rest half to check coverage
  res = fspmix(data = data_matrix[resample,] , label =  mimic_lab , num_clust = 6,
                           bandwidth = 1.5, max_iter = 1000 ,
                           min_gap   = 1 , nrep      = 50)

  pred = prediction(res)
  bands = array(dim = c(2 , 11 , 6))
  for(group in 1:6){
    x = data_matrix[resample[which(pred$predict_label == group)],]
    for(i in 1:11){
      h = res$result$sigma[group,i] * length(x[,i])^(-0.2)
      pad = 10*h
      rng = apply(x, 2, range)
      probs = c(1 - 0.99767, 0.99767)
      interval <- matrix(c(rng[1,] - pad , rng[2,] + pad) , nrow = 2, byrow = TRUE)

      Fhat <- function(z) mean(pnorm((z - x[,i]) / h))
      qfun <- function(p) uniroot(function(z) Fhat(z) - p, interval = interval[,i])$root
      bands[,i,group] <- vapply(probs, qfun, numeric(1))
    }
  }

  # Step 4 Check coverage
  test_sample <- caret::createDataPartition(classes, p = 0.6, list = FALSE)
  cover = rep(0,6)
  for(i in 1:length(test_sample)){
    profile = data_matrix[test_sample[i],]
    cur_pred_lab = classes[test_sample[i]]
    upper = bands[2,,cur_pred_lab]
    lower = bands[1,,cur_pred_lab]
    cover[cur_pred_lab] = cover[cur_pred_lab] +
      prod(profile < upper) * prod(profile > lower)
  }
  exp_cov[t,] = cover/summary(factor(classes[test_sample]))
}
```

```r
load(file = "./paper_exp_dat/exp_cov.Rdata")

exp_cov_long <- as.data.frame(exp_cov) %>%
  setNames(1:6) %>%
  pivot_longer(cols = everything(),
               names_to = "Group",
               values_to = "Coverage")

exp_cov_long$Group <- factor(exp_cov_long$Group)

ggplot(exp_cov_long, aes(x = Group, y = Coverage, fill = Group) ) +
  geom_boxplot() +
  geom_hline(yintercept = 0.95, color = "black", linetype = "dashed", linewidth = 1.5) +
  #annotate("text", x = 0.3, y = 0.95, label = "95%",
  #color = "red", hjust = 1.2,vjust = 0, size = 4) +
  labs(title = "Prediction band coverage by group",) +
  scale_fill_manual(values = c( "#4FDF4D" ,"#188EB8" ,  "#D90A1C" ,
                                "#900EA3" ,  "#FF8F00" , "#F251BF" ), drop = FALSE) +
  scale_x_discrete(labels = c("Chloroplast","Golgi/ER", "Mitochondrion","Nucleus",
                              "Proteasome","Ribosome" ))+
  theme_classic()+
  theme(plot.title = element_text(size = 16, hjust = 0.5, face = "bold"),
        axis.title.x = element_text(size = 14),
        axis.title.y = element_text(size = 14),
        axis.text.x = element_text(size = 12),
        axis.text.y = element_text(size = 12),
        legend.position = "none")+
  annotate("text", x = 2.7, y = 0.905, label = "Target coverage",
           size = 5 , col = "black") +
  annotate("segment",
           x = 2.5, y = 0.91, xend = 2.7, yend = 0.948,
           arrow = arrow(length = unit(3, "mm")),
           linewidth = 0.8,
           col = "black") +
  coord_cartesian(ylim = c(0.8, 1), clip = "off")
```



Code for Figure 3: Choosing the hyper-parameters

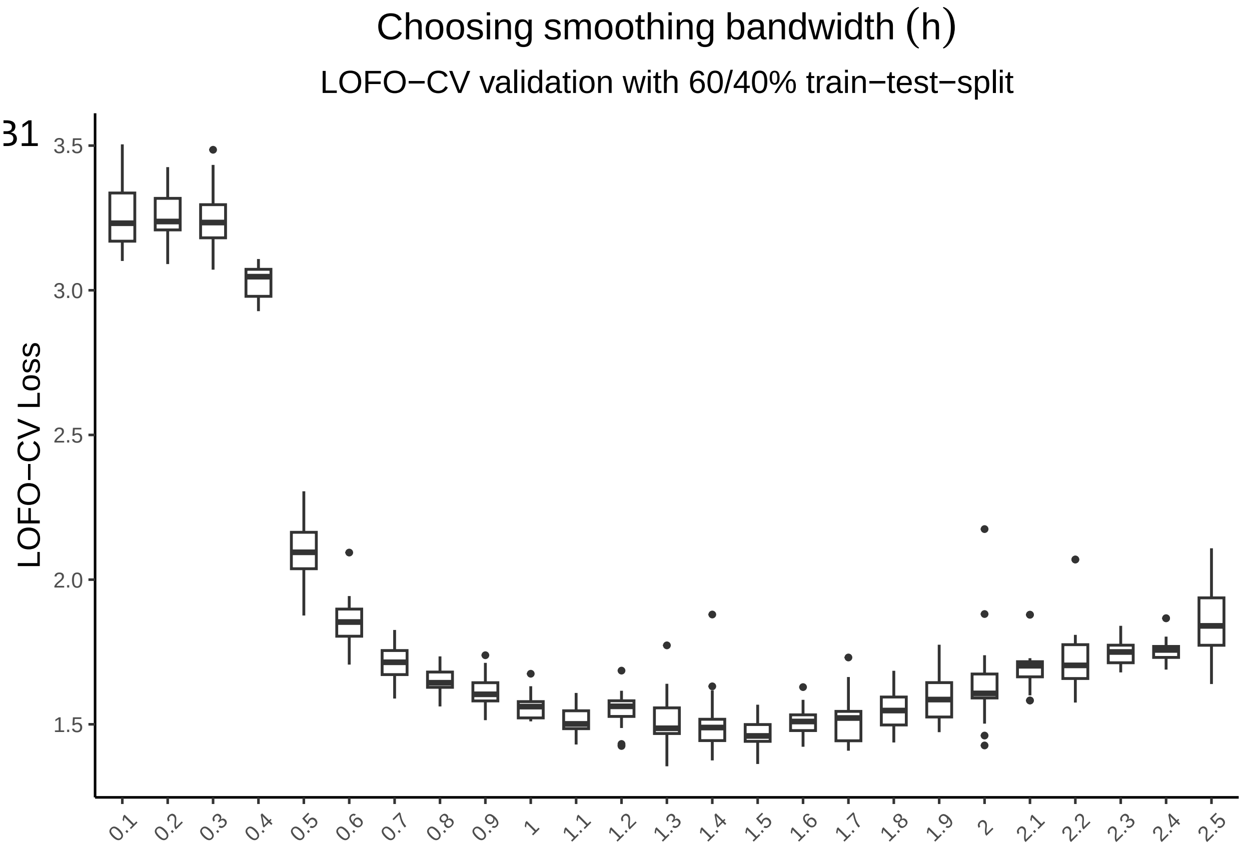
(Top-Right panel) Using Leave-one-out cross-validation(LOO-CV) to choose h

The following chunk is how to use *cv_fspmix* function to do cross validation to choose bandwidth **h**. This can take a long time, so it is not run in the document. We save our result in *cv_res* object and will directly read it in the next chunk to do visualization.

```r
h_grid = seq(0.1,2.5,0.1)
x = cv_fspmix(data = data, label = label, grid = h_grid ,
              num_clust = 10 , max_iter = 1000, min_gap = 0.1, nrep = 20)
x$all_lost   # contian all the lost at every fraction for every h
x$average_lost   #contain the average loss for every h

times = 100
cv_res = vector("list",length(h_grid))
for(t in 1:times){
  res = cv_fspmix(data = data, label = label, grid = h_grid ,
                  num_clust = 10 , max_iter = 1000, min_gap = 0.1, nrep = 20)
  cv_res = Map(c, cv_res, res$average_lost)
}

# save(cv_res , file = "./paper_exp_dat/cv_res.RData")
```

Load the cv experiment

```r
load("./paper_exp_dat/cv_res.RData")
h_grid = seq(0.1,2.5,0.1)

cv_long <- tibble(h = h_grid, CVloss = cv_res) %>%
  unnest_longer(CVloss)

ggplot(cv_long, aes(x = factor(h), y = CVloss)) +
  geom_boxplot(width = .55, outlier.size = .7) +
  # annotate("segment",
  #          x = 16.5, xend = 15,
  #          y = 2, yend = 1.6,
  #          colour = "blue",
  #          arrow = arrow(length = unit(0.25, "cm"), type = "closed"),
  #          linewidth = 1) +
  # annotate("text",
  #          x = 13, y = 2.1,
  #          label = "Minimum loss",
  #          colour = "blue", fontface = "bold", hjust = 0) +
  labs(y = "LOFO-CV Loss",
       x = NULL,
       title = expression(Choosing~smoothing~bandwidth~(h)),
       subtitle = "LOFO-CV validation with 60/40% train-test-split",
       tag = "B1") +
  theme_classic(base_size = 10) +
  theme(axis.title = element_text(size=12),
        axis.text.x = element_text(angle = 45, vjust = 0.5, size = 8),
        plot.title = element_text(size = 14, hjust = 0.5),
        plot.subtitle = element_text(size = 12,hjust = 0.5),
        plot.tag.position = c(0,0.85),
        plot.tag = element_text(size = 14))
```



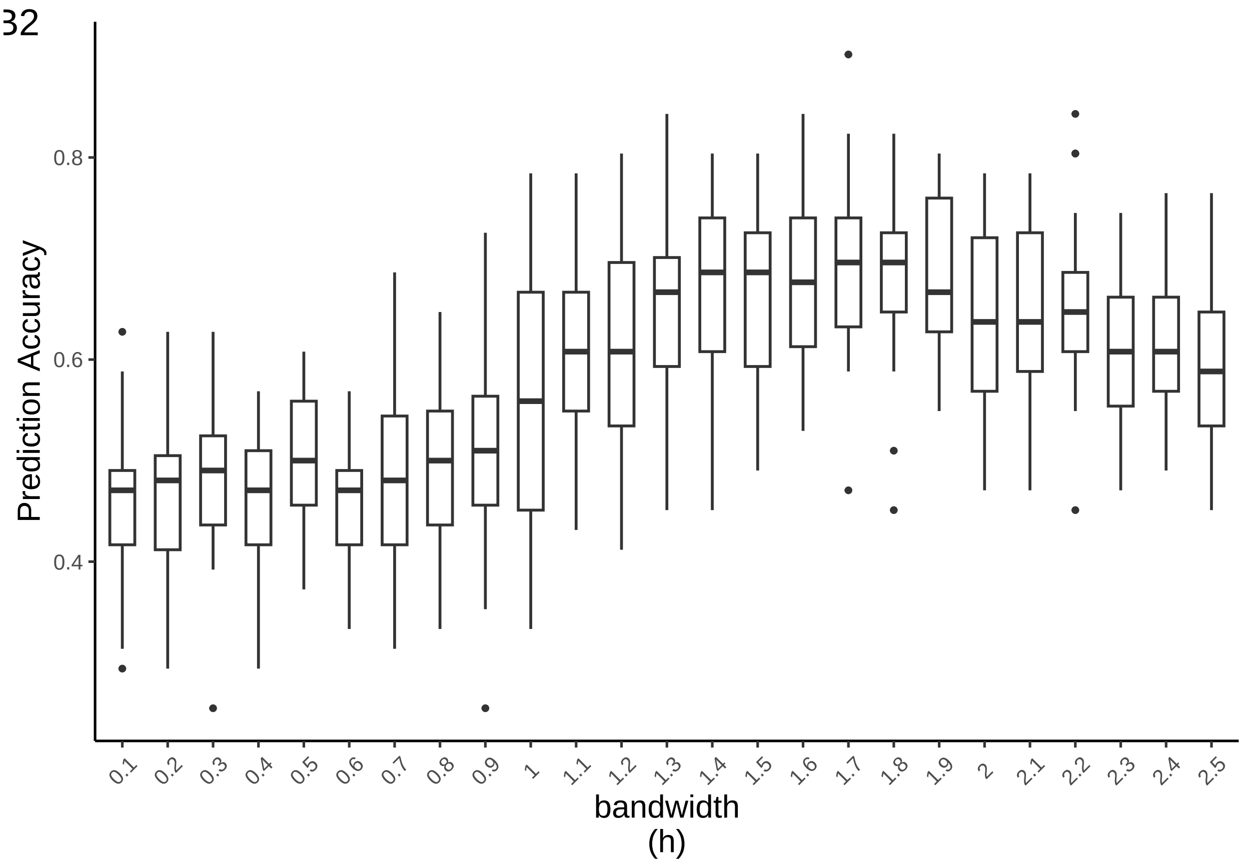
(Bottom-Right panel) Performance based on different h

We perform an experiment to check the model prediction accuracy on different values of **h** using train-test split. We save our result in *htest_acc* object.

```
h_grid <- seq(0.1,2.5,0.1)
times <- 100
# Run on the entire dataset
htest_acc <- matrix(0 , nrow = length(h_grid) , ncol = times)
for(j in 1:times){
  # Train-Test split
  train_size = 0.6
  train_id <- createDataPartition(label$label, p = train_size, list = FALSE)
  train_set <- label[train_id,]
  test_set <- label[-train_id,]
  # Fit the model
  for(i in 1:length(h_grid)){
    res <- fspmix(data = data, label = train_set, num_clust = 6,
                  bandwidth = h_grid[i], max_iter = 1000, min_gap = 0.1, nrep = 50)
    pred <- prediction(res)
    # accuracy
    htest_acc[i,j] <- sum(pred[test_set$index,]$predict_label ==
                            test_set$label)/length(test_set$label)
  }
}

# save(htest_acc, file = "./paper_exp_dat/htest_final.RData")
```

Load the accruacy experiment

```
load("./paper_exp_dat/htest_final.RData")

h_grid <- seq(0.1,2.5,0.1)
htest_acc_long = as.data.frame(t(htest_acc)) %>%
  setNames(h_grid) %>%
  mutate(run = row_number()) %>%
  pivot_longer(-run, names_to = "h",
               values_to = "Accuracy", names_transform = list(h = as.numeric))


ggplot(htest_acc_long, aes(x = factor(h), y = Accuracy)) +
  geom_boxplot(width = .55, outlier.size = .7) +
  # annotate("segment", x = 14, xend = 18, y = 0.42, yend = 0.42,
  #          colour = "blue", linewidth = 1.2) +
  # annotate("segment", x = 14, xend = 14, y = 0.40, yend = 0.44,
  #          colour = "blue", linewidth = 1.2) +
  # annotate("segment", x = 18, xend = 18, y = 0.40, yend = 0.44,
  #          colour = "blue", linewidth = 1.2) +
  # annotate("text", x = 16, y = 0.37, label = "High-accuracy region",
  #          colour = "blue", fontface = "bold") +
  labs(y = "Prediction Accuracy",
       x = "bandwidth\n(h)",
       tag = "B2") +
  theme_classic(base_size = 10) +
  theme(axis.title = element_text(size=12),
        axis.text.x = element_text(angle = 45, vjust = 0.5, size = 8),
        plot.tag.position = c(0,1),
        plot.tag = element_text(size = 14))
```



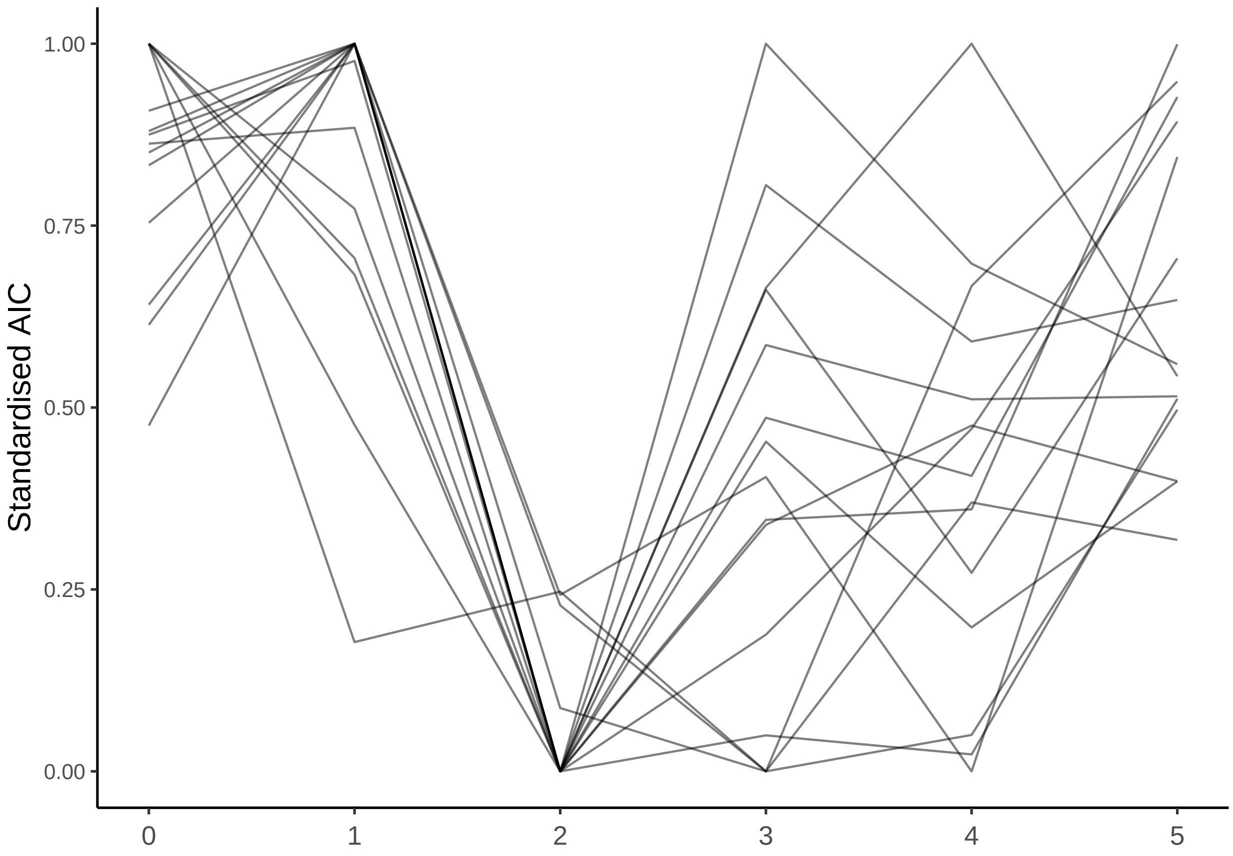
(Left two panels) Validating AIC by hidden two annotated groups.

We design an experiment to validate the procedure to choose number of new group *K_0* with AIC: Fit FSPmix model only on the label dataset by hiding two groups and see if AIC always indicates the correct number of hidden groups. Again, we save our result in *AIC_stand* object and skip the experiment here.

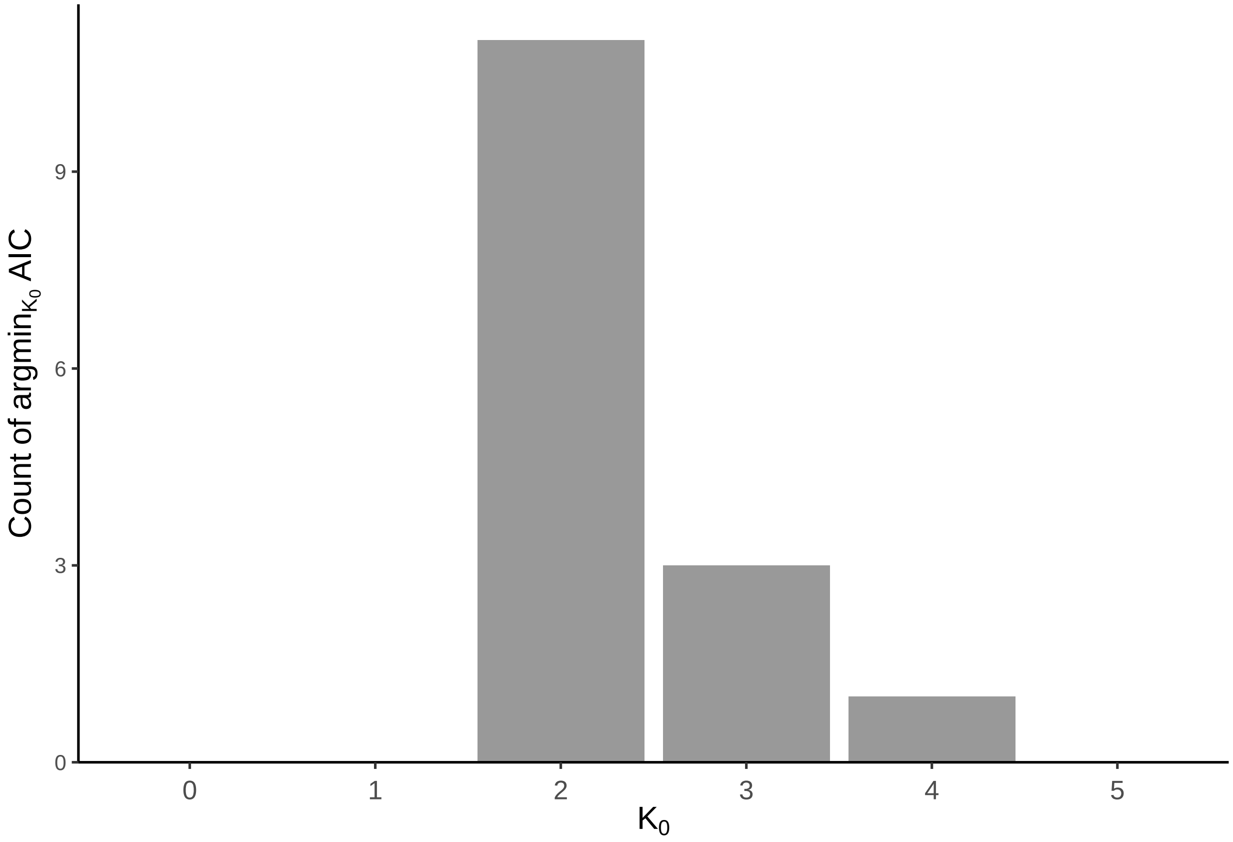
```r
set.seed(3516854)
K_0 = 5
mute_list = combn(6,2)
AIC_validation <- matrix(0,nrow = ncol(mute_list), ncol = 1 + K_0)

for(j in 1:ncol(mute_list)){
  mute <- mute_list[,j]
  label_mute2 <- label[!label$label %in% mute , ]

  label_mute2$index <- which(label$index %in% label_mute2$index) # reindex
  label_mute2$label <- as.integer(factor(label_mute2$label)) # relabel from 1 to K-2
  label_mute2 <- label_mute2[createDataPartition(label_mute2$label,
                                                 p = 0.5,list = FALSE),]
  # only save 50% of the label

  AIC_valid <- aic_fspmix(data = data[label$index,],
                          label = label_mute2, max_new_clust = K_0, bandwidth = 1)
  # the bandwidth is selected using cv only with labeled dataset
  AIC_validation[j,] <- AIC_valid$AIC
}

stand <- function(v){
  mini <- min(v)
  maxi <- max(v)
  (v - mini)/(maxi - mini)
}
AIC_stand = apply(AIC_validation,1,stand) #standardize AIC

plot(seq(0,K_0),AIC_stand[,1],type="l",xlab = "K",
     main = "AIC validation",ylim=c(0,1))
for(i in 2:ncol(mute_list)){
  lines(seq(0,K_0),AIC_stand[,i])
}
```

Load experiment data

```
load("./paper_exp_dat/validate_K0.Rdata")

aic_long <- AIC_stand %>%
  as_tibble(.name_repair = ~ paste0("K", 0:5)) %>%
  mutate(run = row_number()) %>%
  pivot_longer(-run,
               names_to  = "K0",
               values_to = "AIC") %>%
  mutate(K0 = readr::parse_number(K0))
aic_freq <- aic_long %>%
  group_by(run) %>%
  slice_min(AIC, n = 1, with_ties = FALSE) %>%
  count(K0) %>%
  mutate(prop = n / sum(n))
aic_freq <- rbind(aic_freq,data.frame(run = c(15,16,17),
                   K0 = c(0,1,5),
                   n = c(0,0,0),
                   prop = c(0,0,0)) )
# Just to add three 0 to make the barplot show x-axis with 0 value
ggplot(aic_long, aes(x = K0, y = AIC, group = run)) +
  geom_line(alpha = .5, linewidth = .4) +
  scale_x_continuous(breaks = 0:5) +
  labs(y = "Standardised AIC", x = NULL) +
  theme_classic(base_size = 10) +
  theme(axis.title = element_text(size=12),
        axis.text.x = element_text(vjust = 0, size = 10))
```



```
ggplot(aic_freq, aes(x = factor(K0), y = prop)) +
  geom_col(fill = "grey60") +
  scale_y_continuous(expand = expansion(mult = c(0, .05))) +
  labs(y = expression(Count~of~argmin[K[0]]~AIC), x = expression(K[0])) +
  theme_classic(base_size = 10) +
  theme(axis.title = element_text(size=12),
        axis.text.x = element_text(vjust = 0, size = 10))
```



Prediction accuracy with benchmark models in different datasets(Figure 4 and Appendix figure)

All the datasets in *pRolocdata* are stored in *MSnSet* type. In order to fit them with *FSPmix* model, we first need to transfrom them into the function-required data type(More illustration of required data type is in the introduction of FSPmix package).

```r
mylist <- c("yeast2018",
            "moloneyTbBSF",
            "lopitdcU2OS2018",
            "hirst2018",
            "E14TG2aR")
data(list = mylist)


## Preparing data ##
Thaps2024_MSN <- MSnSet(
  exprs = as.matrix(Thaps2024[,1:11])
)
markers <- Thaps2024$annotation_level1
markers[is.na(markers)] <- "unknown"
Thaps2024_MSN@featureData@data[["markers"]] = factor(markers)
fvarMetadata(Thaps2024_MSN) <-
  data.frame(labelDescription = rep(NA_character_, ncol(fData(Thaps2024_MSN))),
             row.names        = colnames(fData(Thaps2024_MSN)))

d1 <- yeast2018@assayData[["exprs"]] %>%
  data.frame() %>%
  mutate(label = yeast2018@featureData@data[["markers"]])
d2 <- moloneyTbBSF@assayData[["exprs"]] %>%
  data.frame() %>%
  mutate(label = moloneyTbBSF@featureData@data[["markers"]])
d3 <- lopitdcU2OS2018@assayData[["exprs"]] %>%
  data.frame() %>%
  mutate(label = lopitdcU2OS2018@featureData@data[["markers"]])
d4 <- hirst2018@assayData[["exprs"]] %>%
  data.frame() %>%
  mutate(label = hirst2018@featureData@data[["markers"]])
d5 <- E14TG2aR@assayData[["exprs"]] %>%
  data.frame() %>%
  mutate(label = E14TG2aR@featureData@data[["markers"]])


lab1 <- data.frame(index = which(d1$label != "unknown"),
                   label = as.factor(as.numeric(
                     droplevels(d1$label[which(d1$label != "unknown")]))))
lab2 <- data.frame(index = which(d2$label != "unknown"),
                   label = as.factor(as.numeric(
                     as.factor(d2$label[which(d2$label != "unknown")]))))
lab3 <- data.frame(index = which(d3$label != "unknown"),
                   label = as.factor(as.numeric(
                     droplevels(d3$label[which(d3$label != "unknown")]))))
lab4 <- data.frame(index = which(d4$label != "unknown"),
                   label = as.factor(as.numeric(
                     as.factor(d4$label[which(d4$label != "unknown")]))))
lab5 <- data.frame(index = which(d5$label != "unknown"),
                   label = as.factor(as.numeric(
                     droplevels(d5$label[which(d5$label != "unknown")]))))
```

Prediction accuracy with all benchmark models in candidate datasets

```
## Helper functions
train_test_split <- function(labelset,train_size){
  stratify <- caret::createDataPartition(labelset$label, p = train_size, list = FALSE)
  train <- labelset[stratify,]
  test <- labelset %>%
    filter(index %in% setdiff(labelset$index,train$index))
  return(list(train_set = train, test_set = test))
}

get_accuracy <- function(mat){  # input a prediction matrix
  diag_sum   <- sum(diag(mat))
  total_sum  <- sum(mat)
  return(diag_sum / total_sum)
}
```

```
## Supervised (SVM)
svm_yeast <- svmOptimisation(yeast2018, test.size = 0.4)
svm_moloney <- svmOptimisation(moloneyTbBSF,test.size = 0.4)
svm_U2OS <- svmOptimisation(lopitdcU2OS2018 , test.size = 0.4)
svm_hirst <- svmOptimisation(hirst2018 ,test.size = 0.4)
svm_E14 <- svmOptimisation(E14TG2aR,test.size = 0.4)
svm_Thaps2024 <- svmOptimisation(Thaps2024_MSN , test.size = 0.4)


svmc_yeast <- sapply(svm_yeast@cmMatrices, get_accuracy)
svmc_moloney <- sapply(svm_moloney@cmMatrices, get_accuracy)
svmc_U2OS <- sapply(svm_U2OS@cmMatrices, get_accuracy)
svmc_hirst <- sapply(svm_hirst@cmMatrices, get_accuracy)
svmc_E14 <- sapply(svm_E14@cmMatrices, get_accuracy)
svmc_Thaps2024 <- sapply(svm_Thaps2024@cmMatrices, get_accuracy)



## Supervised (KNN)
k_yeast <- knnOptimisation(yeast2018,test.size = 0.4)
k_moloney <- knnOptimisation(moloneyTbBSF,test.size = 0.4)
k_U2OS <- knnOptimisation(lopitdcU2OS2018,test.size = 0.4)
k_hirst <- knnOptimisation(hirst2018,test.size = 0.4)
k_E14 <- knnOptimisation(E14TG2aR,test.size = 0.4)
k_Thaps <- knnOptimisation(Thaps2024_MSN,test.size = 0.4)


kc_yeast <- sapply(k_yeast@cmMatrices, get_accuracy)
kc_moloney <- sapply(k_moloney@cmMatrices, get_accuracy)
kc_U2OS <- sapply(k_U2OS@cmMatrices, get_accuracy)
kc_hirst <- sapply(k_hirst@cmMatrices, get_accuracy)
kc_E14 <- sapply(k_E14@cmMatrices, get_accuracy)
kc_Thaps <- sapply(k@cmMatrices, get_accuracy)



## TAGM
times = 100
tagmc_yeast <- rep(0,times)
tagmc_moloney <- rep(0,times)
tagmc_U2OS <- rep(0,times)
tagmc_hirst <- rep(0,times)
tagmc_E14 <- rep(0,times)
for(i in 1:times){
  label1 <- data.frame(index = which(d1$label != "unknown"),
                       label = d1$label[which(d1$label != "unknown")])
  o <- train_test_split(label1,0.6)
  my_yeast2018 <- yeast2018
  my_yeast2018@featureData@data[["markers"]][o$test_set$index] = "unknown"
  mappers <- tagmMapTrain(my_yeast2018)
  predicted <- tagmMapPredict(my_yeast2018,mappers)
  table(fData(predicted)[o$test_set$index,]$tagm.map.allocation , o$test_set$label)
  tagmc_yeast[i] <- sum(fData(predicted)[o$test_set$index,]$tagm.map.allocation ==
                          o$test_set$label)/length(o$test_set$label)

  label2 <- data.frame(index = which(d2$label != "unknown"),
                       label = d2$label[which(d2$label != "unknown")])
  o <- train_test_split(label2,0.6)
  my_moloneyTbBSF <- moloneyTbBSF
  my_moloneyTbBSF@featureData@data[["markers"]][o$test_set$index] = "unknown"
  mappers <- tagmMapTrain(my_moloneyTbBSF)
  predicted <- tagmMapPredict(my_moloneyTbBSF,mappers)
  table(fData(predicted)[o$test_set$index,]$tagm.map.allocation , o$test_set$label)
  tagmc_moloney[i] <- sum(fData(predicted)[o$test_set$index,]$tagm.map.allocation ==
                            o$test_set$label)/length(o$test_set$label)

  label3 <- data.frame(index = which(d3$label != "unknown"),
                       label = d3$label[which(d3$label != "unknown")])
  o <- train_test_split(label3,0.6)
  my_lopitdcU2OS2018 <- lopitdcU2OS2018
  my_lopitdcU2OS2018@featureData@data[["markers"]][o$test_set$index] = "unknown"
  mappers <- tagmMapTrain(my_lopitdcU2OS2018)
  predicted <- tagmMapPredict(my_lopitdcU2OS2018,mappers)
  table(fData(predicted)[o$test_set$index,]$tagm.map.allocation , o$test_set$label)
  tagmc_U2OS[i] <- sum(fData(predicted)[o$test_set$index,]$tagm.map.allocation ==
                         o$test_set$label)/length(o$test_set$label)

  label4 <- data.frame(index = which(d4$label != "unknown"),
                       label = d4$label[which(d4$label != "unknown")])
  o <- train_test_split(label4,0.6)
  my_hirst2018 <- hirst2018
  my_hirst2018@featureData@data[["markers"]][o$test_set$index] = "unknown"
  mappers <- tagmMapTrain(my_hirst2018)
  predicted <- tagmMapPredict(my_hirst2018,mappers)
  table(fData(predicted)[o$test_set$index,]$tagm.map.allocation , o$test_set$label)
  tagmc_hirst[i] <- sum(fData(predicted)[o$test_set$index,]$tagm.map.allocation ==
                          o$test_set$label)/length(o$test_set$label)

  label5 <- data.frame(index = which(d5$label != "unknown"),
                       label = d5$label[which(d5$label != "unknown")])
  o <- train_test_split(label5,0.6)
  my_E14TG2aR <- E14TG2aR
  my_E14TG2aR@featureData@data[["markers"]][o$test_set$index] = "unknown"
  mappers <- tagmMapTrain(my_E14TG2aR)
  predicted <- tagmMapPredict(my_E14TG2aR,mappers)
  table(fData(predicted)[o$test_set$index,]$tagm.map.allocation , o$test_set$label)
  tagmc_E14[i] <- sum(fData(predicted)[o$test_set$index,]$tagm.map.allocation ==
                        o$test_set$label)/length(o$test_set$label)
}


# TAGM
tagmc_Thaps <- rep(0,times)
for(i in 1:times){
  markers <- full_data$annotation_level1
  markers[is.na(markers)] <- "unknown"
  label <- data.frame(index = which(markers != "unknown"),
                      label = markers[which(markers != "unknown")])
  o <- train_test_split(label,0.6)
  markers[o$test_set$index] = "unknown"
  my_Thaps2024 <- Thaps2024_MSN
  my_Thaps2024@featureData@data[["markers"]] = factor(markers)
  mappers <- tagmMapTrain(my_Thaps2024)
  predicted <- tagmMapPredict(my_Thaps2024,mappers)
  tagmc[i] <- sum(fData(predicted)[o$test_set$index,]$tagm.map.allocation ==
                    o$test_set$label)/length(o$test_set$label)
}
```

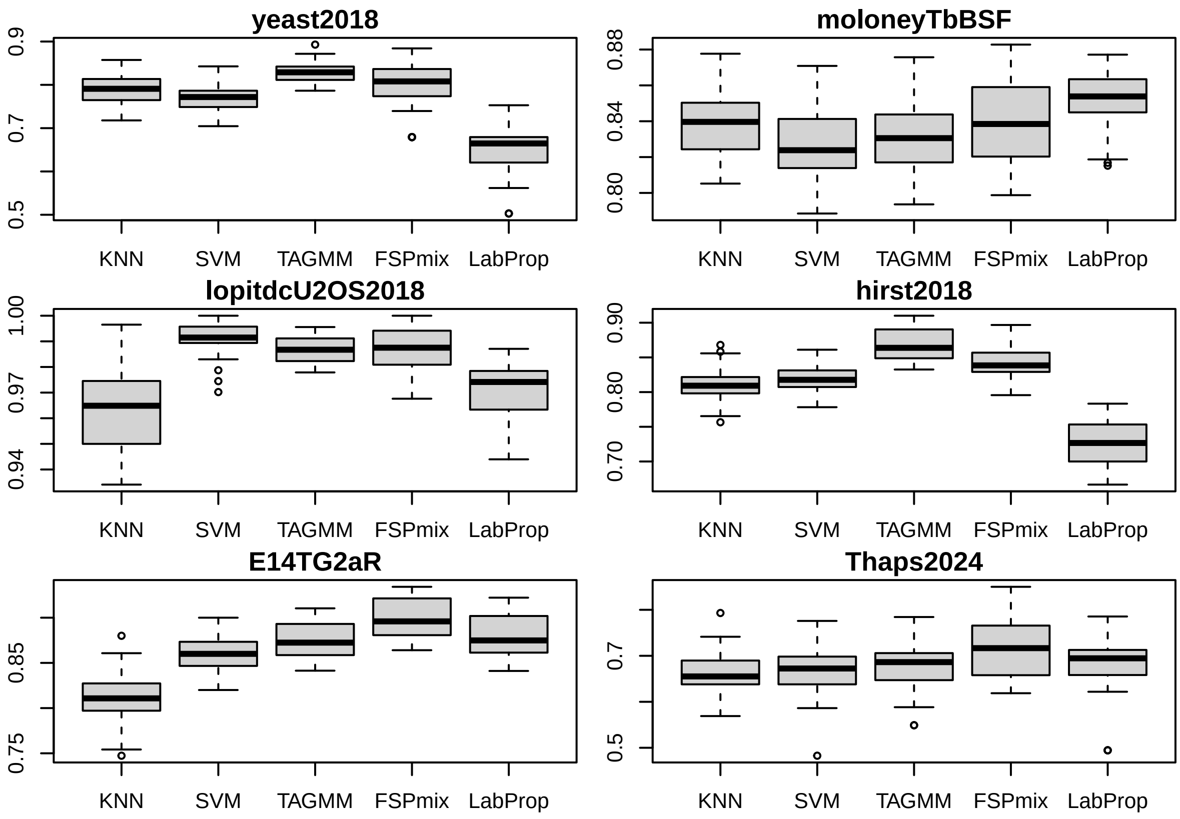
Prediction accuracy with FSPmix in candidate datasets

```
times = 100
yeast2018_res = rep(0,times)
moloneyTbBSF_res = rep(0,times)
lopitdcU2OS2018_res = rep(0,times)
hirst2018_res = rep(0,times)
E14TG2aR_res = rep(0,times)
for(i in 1:times){
  # yeast2018
  yeast2018_split <- train_test_split(labelset = lab1 , train_size = 0.6)
  res_1 <- fspmix(d1[,1:40] ,label = yeast2018_split$train_set ,
                  num_clust = 12, bandwidth = 0.3 ,max_iter = 1000 ,
                  min_gap = 0.1 ,nrep = 30)
  pred_1 <- prediction(res_1)
  yeast2018_res[i] = sum(pred_1[yeast2018_split$test_set$index,]$predict_label ==
            yeast2018_split$test_set$label)/length(yeast2018_split$test_set$label)
  # moloneyTbBSF
  moloneyTbBSF_split <- train_test_split(labelset = lab2 , train_size = 0.6)
  res_2 <- fspmix(d2[,1:33] ,label = moloneyTbBSF_split$train_set ,
                  20, bandwidth = 0.3, max_iter = 1000 ,min_gap = 0.1 ,nrep = 30)
  pred_2 <- prediction(res_2)
  moloneyTbBSF_res[i] = sum(pred_2[moloneyTbBSF_split$test_set$index,]$predict_label ==
             moloneyTbBSF_split$test_set$label)/length(moloneyTbBSF_split$test_set$label)
  # lopitdcU2OS2018
  lopitdcU2OS2018_split <- train_test_split(labelset = lab3 , train_size = 0.6)
  res_3 <- fspmix(d3[,1:30] ,label = lopitdcU2OS2018_split$train_set ,
                  10, bandwidth = 0.4,max_iter = 1000 ,min_gap = 0.1 ,nrep = 30)
  pred_3 <- prediction(res_3)
  lopitdcU2OS2018_res[i] =
    sum(pred_3[lopitdcU2OS2018_split$test_set$index,]$predict_label ==
           lopitdcU2OS2018_split$test_set$label)/
    length(lopitdcU2OS2018_split$test_set$label)
  # hirst2018
  hirst2018_split <- train_test_split(labelset = lab4 , train_size = 0.6)
  res_4 <- fspmix(d4[,1:45] ,label = hirst2018_split$train_set , 12, bandwidth = 0.4 ,
                  max_iter = 1000 ,min_gap = 0.1 ,nrep = 30)
  pred_4 <- prediction(res_4)
  hirst2018_res[i] =
    sum(pred_4[hirst2018_split$test_set$index,]$predict_label ==
          hirst2018_split$test_set$label)/length(hirst2018_split$test_set$label)
  # E14TG2aR
  E14TG2aR_split <- train_test_split(labelset = lab5 , train_size = 0.6)
  res_5 <- fspmix(d5[,1:8] ,label = E14TG2aR_split$train_set , 10, bandwidth = 0.5 ,
                  max_iter = 1000 ,min_gap = 0.1 ,nrep = 30)
  pred_5 <- prediction(res_5)
  E14TG2aR_res[i] =
    sum(pred_5[E14TG2aR_split$test_set$index,]$predict_label ==
          E14TG2aR_split$test_set$label)/length(E14TG2aR_split$test_set$label)
}
```

Draw the accuracy boxplot (Appendix)

Read all the experiment results and visualize the accuracy.

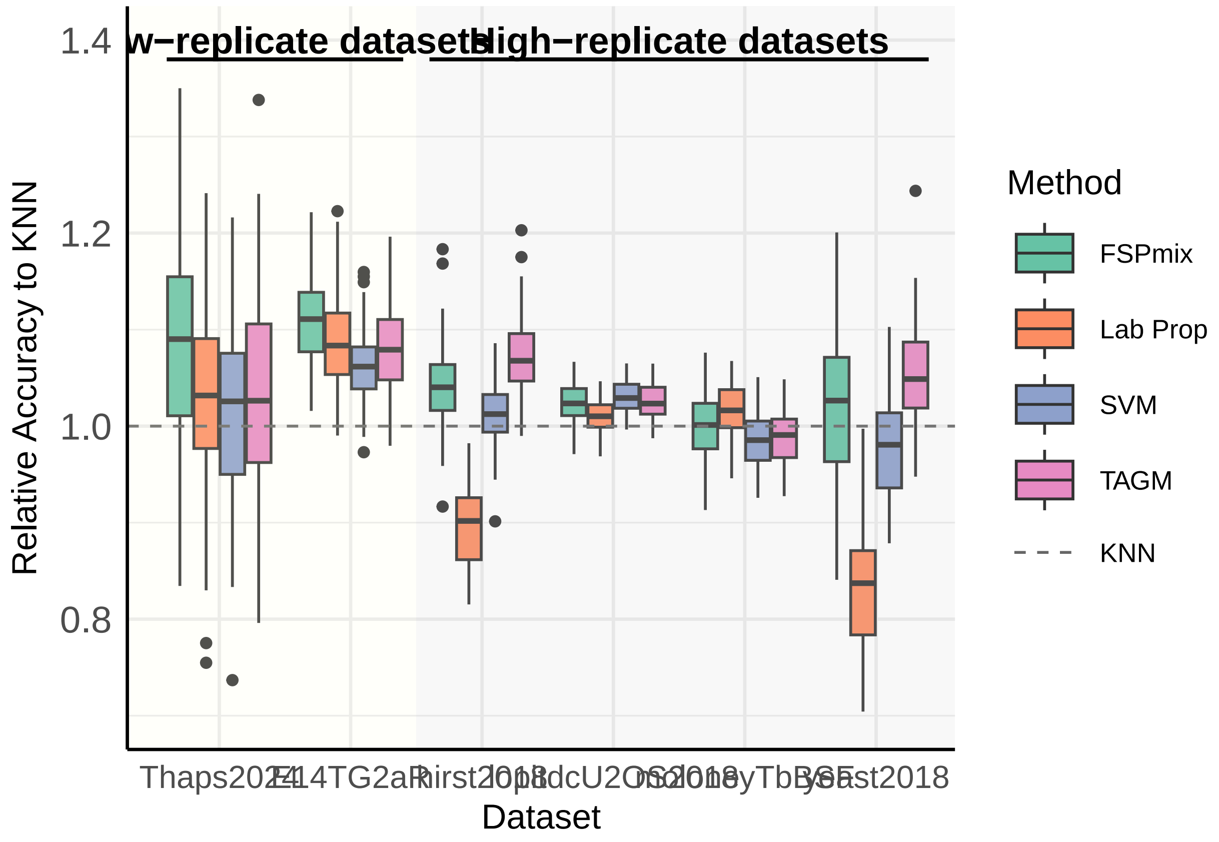
```
a <- load('./paper_exp_dat/accuracy_comparison_result.Rdata')

par(mfrow=c(3,2),
    mar   = c(2, 2.5, 1.5, 0.5),
    oma   = c(1, 1, 1, 1))
boxplot(yeast2018_all , main="yeast2018")
boxplot(moloneyTbBSF_all, main="moloneyTbBSF")
boxplot(lopitdcU2OS2018_all , main = "lopitdcU2OS2018")
boxplot(hirst2018_all, main = "hirst2018")
boxplot(E14TG2aR_all , main = "E14TG2aR")
boxplot(Thaps2024_all, main = "Thaps2024")
```



Draw the relative accuracy boxplot (Figure 4)

We calculate the ratio of candidate models prediction to KNN prediction accuracy(so called relative accuracy). Then we can visualize all the results across datasets in one figure.

```
yeast_compare <- yeast2018_all[,2:5]/yeast2018_all[,1]
yeast_compare <- yeast_compare %>%
  rename_with(~ paste0(.x,'_yeast2018'))
moloney_compare <- moloneyTbBSF_all[,2:5]/moloneyTbBSF_all[,1]
moloney_compare <- moloney_compare %>%
  rename_with(~ paste0(.x,'_moloneyTbBSF'))
U2OS_compare <- lopitdcU2OS2018_all[,2:5]/lopitdcU2OS2018_all[,1]
U2OS_compare <- U2OS_compare %>%
  rename_with(~ paste0(.x,'_lopitdcU2OS2018'))
hirst_compare <- hirst2018_all[,2:5]/hirst2018_all[,1]
hirst_compare <- hirst_compare %>%
  rename_with(~ paste0(.x,'_hirst2018'))
E14_compare <- E14TG2aR_all[,2:5]/E14TG2aR_all[,1]
E14_compare <- E14_compare %>%
  rename_with(~ paste0(.x,'_E14TG2aR'))
Thaps_compare <- Thaps2024_all[,2:5]/Thaps2024_all[,1]
Thaps_compare <- Thaps_compare %>%
  rename_with(~ paste0(.x,'_Thaps2024'))

entire_compareKNN <- cbind(yeast_compare,moloney_compare,U2OS_compare,
                           hirst_compare,E14_compare,Thaps_compare)

df_long <- entire_compareKNN |>
  pivot_longer(
    cols = everything(),
    names_to   = c("method", "dataset"),   # split the name at the underscore
    names_sep  = "_",
    values_to  = "Relative_Accuracy_to_KNN"
  )


df_long$dataset <- factor(df_long$dataset ,
                          levels = c("Thaps2024" ,"E14TG2aR", "hirst2018",
                                     "lopitdcU2OS2018", "moloneyTbBSF", "yeast2018"))
# Sort the boxplot


ggplot(df_long, aes(x = dataset, y = Relative_Accuracy_to_KNN, fill = method)) +
  geom_boxplot(position = position_dodge(width = .8)) +
  labs(x = "Dataset",
       y = expression(Relative~Accuracy~to~KNN),
       fill = "Method") +
  theme_minimal(base_size = 13) +
  ylim(0.7,1.4) +
  theme(
    axis.line = element_line(),
    axis.text.x = element_text(size = 12),
    axis.text.y = element_text(size = 14),
    legend.key.size = unit(1, "cm"),
    legend.spacing.y = unit(-0.5, "cm")
  )  +
  scale_fill_manual(labels = c("FSPmix","Lab Prop","SVM","TAGM"),
                    values = c("#66C2A5","#FC8D62","#8DA0CB", "#E78AC3"),
                    guide = guide_legend(order = 1)
                    ) +
  geom_hline(aes(yintercept = 1,
                 linetype   = "KNN"),
             linewidth  =0.5,
             colour     = "grey40")+
  scale_linetype_manual(name = NULL,
                        values = c("KNN" = "dashed"),
                        guide = guide_legend(order = 2)
                        ) +
  annotate( "rect", xmin  = 0.3,  xmax  = 2.5, ymin  = -Inf, ymax = Inf,
            fill  = "lightyellow",  alpha = 0.15) +
  annotate( "rect", xmin  = 2.5,  xmax  = Inf, ymin  = -Inf, ymax = Inf,
            fill  = "lightgray",  alpha = 0.15) +
  annotate("segment", x = 0.6, xend = 2.4, y = 1.38, yend = 1.38, size = .7) +
  annotate("segment", x = 2.6, xend = 6.4, y = 1.38, yend = 1.38, size = .7) +
  annotate("text",    x = 1.5,   y = 1.4, label = "Low-replicate datasets",
           fontface = "bold", size = 5) +
  annotate("text",    x = 4.5,   y = 1.4, label = "High-replicate datasets",
           fontface = "bold", size = 5)
```



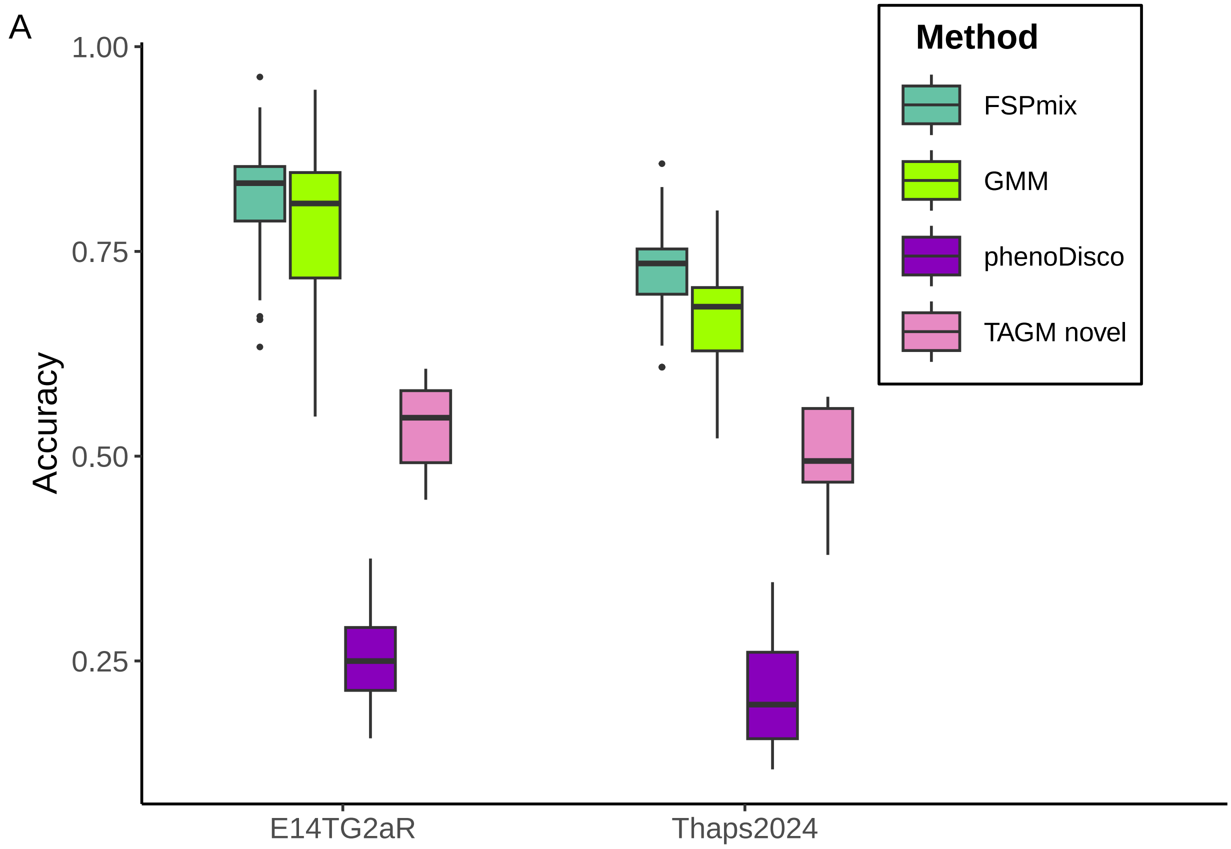
New group discovery (Figure 5)

```
## Helper function of mute some classes of label
mute_labs <- function(label_set , mute_num){
  mute <- sample(levels(label_set$label), mute_num)

  train_label <- label_set[!label_set$label %in% mute,]
  muted_label <- label_set[label_set$label %in% mute,]

  return(list(train_label = train_label , muted_label = muted_label))
}


## E14TG2aR
levels(lab5$label) ## see how many annotated groups we have
new_pred_E14 <- matrix(0,100,3)
for(i in 1:100){
  # FSPmix
  a1 <- mute_labs(lab5,2) ## Make E14 mute 2 groups
  a1$train_label$label <- as.integer(factor(a1$train_label$label))
  # rearrange the label number
  bandw <- cv_fspmix(d5[,1:8] ,label = a1$train_label , grid = seq(0.1,2.5,0.1),
          num_clust = 12 , max_iter = 1000, min_gap = 0.1, nrep = 10) # choosing h
```

```r
  n_clust <- aic_fspmix(d5[,1:8] ,label = a1$train_label , max_new_clust = 6,
           bandwidth = 0.4, max_iter = 1000, min_gap = 0.1, nrep = 20) # choosing K_0
  res_new_E14 <- fspmix(d5[,1:8] ,label = a1$train_label , num_clust = 10,
                        bandwidth = 0.4 ,max_iter = 1000 ,min_gap = 1 ,nrep = 20)
  t1 <- table(apply(res_new_E14$result$resp,1,which.max)[a1$muted_label$index],
              a1$muted_label$label)# result
  new_pred_E14[i,1] <- sum(apply(t1, 2, max))/sum(t1)
  # Pure GMM
  GMM_E14 <- fspmix(d5[,1:8] ,label = a1$train_label , num_clust = 10, bandwidth = 0.01,
                    max_iter = 1000 ,min_gap = 1 ,nrep = 20)
  t2 <- table(apply(GMM_E14$result$resp,1,which.max)[a1$muted_label$index],
              a1$muted_label$label) # result
  new_pred_E14[i,2] <- sum(apply(t2, 2, max))/sum(t2)
  # Pheno Discover
  mute2_E14TG2aR <- E14TG2aR
  mute_label <- levels(
    mute2_E14TG2aR@featureData@data[["markers"]])[unique(a1$muted_label$label)]
  mute2_E14TG2aR@featureData@data[["markers"]][mute2_E14TG2aR@featureData@data[["markers"]]
                                               %in% mute_label] <- "unknown"
  mute2_E14TG2aR@featureData@data[["markers"]] <-
    droplevels(mute2_E14TG2aR@featureData@data[["markers"]]) # droplevels
  pheno_E14 <- phenoDisco(mute2_E14TG2aR, GS = 8, times = 10, fcol = "markers")
  #run phenoDisco
  t3 <- table(fData(pheno_E14)[a1$muted_label$index,]$pd, a1$muted_label$label)
  #result
  new_pred_E14[i,3] <- sum(apply(t3[-nrow(t3),], 2, max))/sum(t3)
}

## Thaps2024
label$label <- as.factor(label$label)
levels(label$label) ## see how many annotated groups we have
new_pred_Loay <- matrix(0,100,3)
for(i in 1:100){
  # FSPmix
  a2 <- mute_labs(label,2)
  ## Make Thaps2024 mute 2 groups (Let ABC[the 1 group]/Nitro[the 5 group] not count in)
  a2$train_label$label <- as.integer(factor(a2$train_label$label))
  # rearrange the label number
  bandw <- cv_fspmix(data = data ,label = a2$train_label , grid = seq(0.1,2.5,0.1),
          num_clust = 12 , max_iter = 1000, min_gap = 0.1, nrep = 10) # choosing h
  n_clust <- aic_fspmix(data = data ,label = a2$train_label , max_new_clust = 6,
           bandwidth = 1.5, max_iter = 1000, min_gap = 0.1, nrep = 20) # choosing K_0
  res_new_loay <- fspmix(data = data ,label = a2$train_label , num_clust = 6,
                         bandwidth = 1.5 ,max_iter = 1000 ,min_gap = 0.1 ,nrep = 50)
  t2 <- table(apply(res_new_loay$result$resp,1,which.max)[a2$muted_label$index],
              a2$muted_label$label) # result
  new_pred_Loay[i,1] <- sum(apply(t2, 2, max))/sum(t2)
  # Pure GMM
  res_new_GMM <- fspmix(data = data ,label = a2$train_label , num_clust = 6,
                        bandwidth = 0.01 ,max_iter = 1000 ,min_gap = 0.1 ,nrep = 50)
  t3 <- table(apply(res_new_GMM$result$resp,1,which.max)[a2$muted_label$index],
              a2$muted_label$label) # result
  new_pred_Loay[i,2] <- sum(apply(t3, 2, max))/sum(t3)
  # Pheno Discover
  mute2_Thaps2024 <- Thaps2024_MSN
  mute_label2 <- levels(mute2_Thaps2024@featureData@data[["markers"]])[
    unique(a2$muted_label$label)]
  mute2_Thaps2024@featureData@data[["markers"]][
    mute2_Thaps2024@featureData@data[["markers"]] %in% c(mute_label2)] <- "unknown"
  mute2_Thaps2024@featureData@data[["markers"]] <-
    droplevels(mute2_Thaps2024@featureData@data[["markers"]]) # droplevels
  pheno_Loay <- phenoDisco(mute2_Thaps2024, GS = 5, times = 10,
                           fcol = "markers", verbose = TRUE) #run phenoDisco
  t4 <- table(fData(pheno_Loay)[a2$muted_label$index,]$pd, a2$muted_label$label) #result
  new_pred_Loay[i,3] <- sum(apply(t4, 2, max))/sum(t4)
}
```

New group discovery accuracy(left panel)

```
load("./paper_exp_dat/new_group_discovery.Rdata")
load('./paper_exp_dat/new_group_tagmnovel.Rdata')

new_discover_all <- tibble(dataset = c("E14TG2aR","Thaps2024",
                                       "E14TG2aR","Thaps2024",
                                       "E14TG2aR","Thaps2024",
                                       "E14TG2aR","Thaps2024"),
                           Method = c("FSPmix","FSPmix",
                                      "GMM","GMM",
                                      "phenoDisco","phenoDisco",
                                      "TAGM_novel","TAGM_novel"),
                           acc = list(fspmix_new_E14,fspmix_new_Thaps,
                                      gmm_new_E14,gmm_new_Thaps,
                                      pheno_new_E14,pheno_new_Thaps,
                                      TAGM_novel_E14,TAGM_novel_Thaps)) %>%
  unnest_longer(acc)

ggplot(new_discover_all, aes(x=dataset,
                                    y=acc,
                                    fill = Method)) +
  geom_boxplot(width = .55,
               outlier.size = 0.5) +
  labs(x = "Dataset",
       y = "Accuracy",
       tag = "A") +
  theme_classic()+
  scale_x_discrete(expand = expansion(mult = c(0.5, 1.2))) +
  theme(
    plot.title = element_text(size = 15, face = "bold", hjust = 0.5),
    axis.text.x = element_text(size = 11),
    axis.title.x = element_blank(),
    axis.text.y = element_text(size = 11),
    axis.title.y = element_text(size = 13),
    legend.text      = element_text(size = 10),
    legend.title     = element_text(size = 13, face = "bold", hjust = 0.2),
    legend.key.size    = unit(1, "cm"),
    legend.position = c(0.8, 0.8),
    legend.spacing.y    = unit(0.1, "cm"),
    legend.background = element_rect(fill = "white", colour = "black")
  ) +
  scale_fill_manual(labels = c("FSPmix","GMM","phenoDisco","TAGM novel"),
                    values = c("#66C2A5","#9FFF00","#80B","#E78AC3"),
                    guide = guide_legend(order = 1)
  )
```

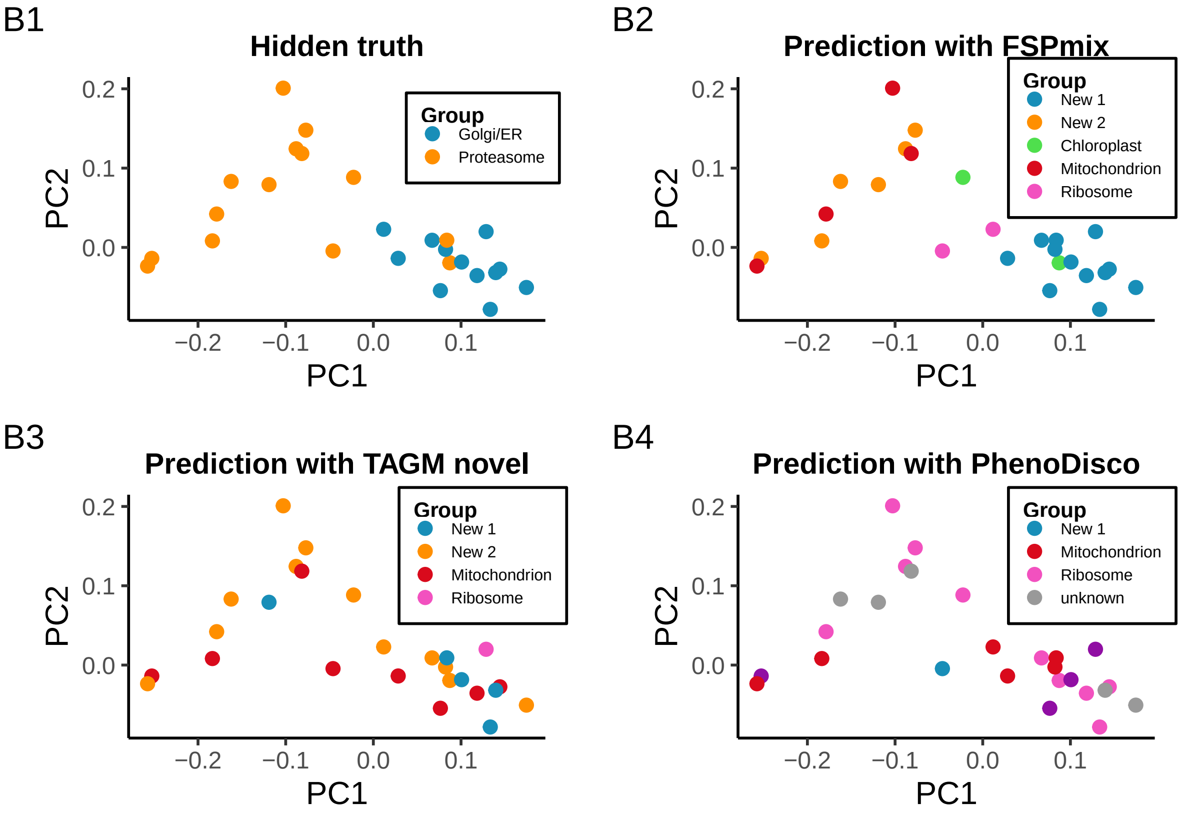


One time experiment(right panel)

We store one of the experiment results in

```
load("./paper_exp_dat/novel_onetime_prediction.Rdata")
load('./paper_exp_dat/novel_onetime.Rdata')
load('./paper_exp_dat/phenodisco_Loay.Rdata')
load('./paper_exp_dat/tagm_newpred.Rdata')
load(file.path("./paper_exp_dat/umap_scores.Rdata"))
load(file.path("./paper_exp_dat/pca_scores.Rdata"))
full_data = Thaps2024 %>%
  mutate(UMAP1 = umap_scores$X1,
         UMAP2 = umap_scores$X2,
          PC1 = pr[,1],
          PC2 = pr[,2])
```

```r
one_time_dat <- list(train_label = label[label$label %in% c(2,5),],
                     muted_label = label[label$label %in% c(2,5),])


g1 <- full_data[one_time_dat$muted_label$index,] %>%
  ggplot() +
  geom_point(aes(x = PC1, y = PC2, col = annotation_level1), size=2) +
  labs(title = "Hidden truth",
       color = "Group",
       tag = "B1") +
  theme_classic()+
  theme(plot.title = element_text(size = 11, face = "bold", hjust = 0.5),
        axis.title.x = element_text(size = 12),
        axis.title.y = element_text(size = 12),
        legend.title = element_text(size = 8, face = "bold",
                                    margin = margin(t = -1, r = 0, b = -1, l = 0) ),
        legend.position = c(0.85, 0.75),
        legend.text = element_text(size = 6),
        legend.key.size   = unit(0.6, "lines"),
        legend.background = element_rect(fill = "white", colour = "black"),
        plot.margin = margin(t = 5, r = 20, b = 5, l = 5))+
  scale_colour_manual(values = c('Golgi/ER' = '#188EB8', 'Proteasome' = '#FF8F00'))
g2 <- full_data[one_time_dat$muted_label$index,] %>%
  ggplot() +
  geom_point(aes(x = PC1, y = PC2, col = c("Ribosome" ,"Chloroplast", "New 1",
                                           "New 2", "Nucleus", "Mitochondrion")
                 [loay_newpred$predict_label[one_time_dat$muted_label$index]]),size=2) +
  labs(title = "Prediction with FSPmix",
       color = "Group",
       tag = "B2") +
  theme_classic()+
  theme(plot.title = element_text(size = 11, face = "bold", hjust = 0.5),
        legend.title = element_text(size = 8, face = "bold",
                                    margin = margin(t = -1, r = 0, b = -1, l = 0)),
        legend.position = c(0.85, 0.75),
        legend.text = element_text(size = 6),
        axis.title.x = element_text(size = 12),
        axis.title.y = element_text(size = 12),
        legend.key.size   = unit(0.6, "lines"),
        legend.background = element_rect(fill = "white", colour = "black"),
        plot.margin = margin(t = 5, r = 20, b = 5, l = 5))+
  scale_colour_manual(values = c("Chloroplast" = '#4FDF4D',"Mitochondrion" = '#D90A1C',
                                 "Ribosome" = '#F251BF',
                                 "New 1" = "#188EB8", "New 2" = '#FF8F00'),
                      breaks = c("New 1", "New 2",
                                 "Chloroplast", "Mitochondrion", "Ribosome"))
levels(tagm_newpred)[levels(tagm_newpred) == "Phenotype 1"] = "New 2"
# Just change the group names to align
levels(tagm_newpred)[levels(tagm_newpred) == "Phenotype 2"] = "New 1"
# Just change the group names to align
g3 <- full_data[one_time_dat$muted_label$index,] %>%
  ggplot() +
  geom_point(aes(x = PC1, y = PC2, col = tagm_newpred), size=2) +
  labs(title = "Prediction with TAGM novel",
       color = "Group",
       tag = "B3") +
  theme_classic()+
  theme(plot.title = element_text(size = 11, face = "bold", hjust = 0.5),
        axis.title.x = element_text(size = 12),
        axis.title.y = element_text(size = 12),
        legend.title = element_text(size = 8, face = "bold",
                                    margin = margin(t = -1, r = 0, b = -1, l = 0)),
        legend.position = c(0.85, 0.75),
        legend.text = element_text(size = 6),
        legend.key.size   = unit(0.6, "lines"),
        legend.background = element_rect(fill = "white", colour = "black"),
        plot.margin = margin(t = 5, r = 20, b = 5, l = 5))+
  scale_colour_manual(values = c("Mitochondrion" = '#D90A1C', "Ribosome" = '#F251BF',
                                 "New 1" = "#188EB8", "New 2" = '#FF8F00'),
                      breaks = c("New 1", "New 2", "Mitochondrion", "Ribosome"))

g4 <- full_data[one_time_dat$muted_label$index,] %>%
  ggplot() +
  geom_point(aes(x = PC1, y = PC2, col = pheno_newpred), size=2) +
  labs(title = "Prediction with PhenoDisco",
       color = "Group",
       tag = "B4") +
  theme_classic()+
  theme(plot.title = element_text(size = 11, face = "bold", hjust = 0.5),
        axis.title.x = element_text(size = 12),
        axis.title.y = element_text(size = 12),
        legend.title = element_text(size = 8, face = "bold",
                                    margin = margin(t = -1, r = 0, b = -1, l = 0)),
        legend.text = element_text(size = 6),
        legend.position = c(0.85, 0.75),
        legend.key.size   = unit(0.6, "lines"),
        legend.background = element_rect(fill = "white", colour = "black"),
        plot.margin = margin(t = 5, r = 20, b = 5, l = 5))+
  scale_colour_manual(values = c("Mitochondrion" = '#D90A1C', "Ribosome" = '#F251BF',
                                 "Nucleus" = "#900EA3","New 1" = "#188EB8",
                                 "New 2" = '#FF8F00', "unknown" = "#999999"),
                      breaks = c("New 1", "New 2",
                                 "Mitochondrion", "Ribosome", "unknown"))

(g1|g2) / (g3|g4)
```

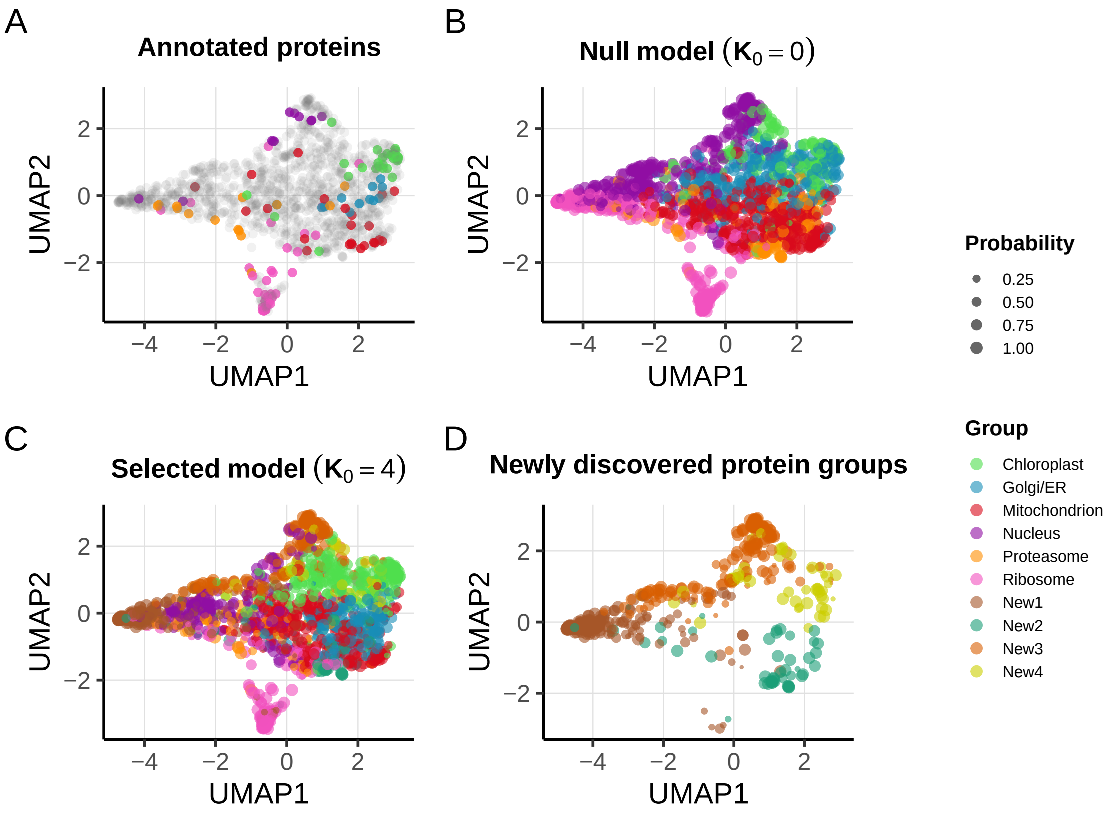


Visualize the result (Figure 6)

In this section, we are going to visualize the model result of Thaps2024 dataset. All the detailed explanation of following figures are in the paper(refer the paper). ## UMAP prediction plot (Top 4 panel)

```
res1 = fspmix(data = data, label = label, num_clust = 6, bandwidth = 1.5, nrep = 50)
res2 = fspmix(data = data, label = label, num_clust = 10, bandwidth = 1.5, nrep = 50)
pred1 = prediction(res1)
pred2 = prediction(res2)

# (Optional) You can use the visualize_res_UMAP function built within package
#visualize_res_UMAP(data = data, label = label , res = res)
```

```
pal <- c("#4FDF4D","#188EB8","#D90A1C","#900EA3","#FF8F00",
         "#F251BF","#A65628","#1B9E77","#D95F02","#CCCF00","#999999")
groups <- c("Chloroplast", "Golgi/ER", "Mitochondrion", "Nucleus",
            "Proteasome", "Ribosome", "New1", "New2", "New3", "New4")

vis_data <- data.frame(UMAP1 = umap_scores$X1,
                       UMAP2 = umap_scores$X2,
                       anno1 = Thaps2024$annotation_level1,
                       pred1 = factor(groups[pred1$predict_label],levels = groups),
                       prob1 = pred1$prob,
                       pred2 = factor(groups[pred2$predict_label],levels = groups),
                       prob2 = pred2$prob)

p1 <- vis_data %>% ggplot() +
  geom_point(aes(x = UMAP1, y = UMAP2, col = anno1), size=1,
             alpha = ifelse(is.na(vis_data$anno1),0.1,0.8)) +
  scale_colour_manual(values = pal) +
  theme_classic() +
  theme(legend.title = element_blank(),
        plot.subtitle = element_text(hjust = 0.5,face = "bold",size = 10),
        legend.position = "none",
        panel.grid.major = element_line(color = "gray88", size = 0.2)) +
  labs(subtitle = "Annotated proteins",
       tag = "A")
p2 <- vis_data %>%
  ggplot() +
  geom_point(aes(x = UMAP1, y = UMAP2, col = pred1, size = prob1) , alpha = .6) +
  scale_size(range = c(0.01, 1.5)) +
  scale_colour_manual(values = pal) +
  theme_classic() +
  theme(legend.title = element_blank(),
        plot.subtitle = element_text(hjust = 0.5,face = "bold",size = 10),
        legend.position = "none",
        panel.grid.major = element_line(color = "gray88", size = 0.2)) +
  labs(subtitle = expression(bold(Null~model~(K[0]==0))),
       tag = "B")
p3 <- vis_data %>%
  ggplot() +
  geom_point(aes(x = UMAP1, y = UMAP2, col = pred2, size = prob2) , alpha = .6) +
  scale_size(range = c(0.01, 1.5), guide = guide_legend(order = 2)) +
  scale_colour_manual(values = pal) +
  theme_classic() +
  theme(plot.subtitle = element_text(hjust = 0.5,size = 10),
        panel.grid.major = element_line(color = "gray88", size = 0.2),
        legend.title = element_text(size = 8, face = "bold"),
        legend.text = element_text(size = 6),
        legend.position = c(3, 1.2),
        legend.key.size = unit(0.6, "lines")
        ) +
  labs(colour = "Group",
       size = "Probability",
       subtitle = expression(bold(Selected~model~(K[0]==4))),
       tag = "C")
p4 <- vis_data %>%
  dplyr::filter(pred2 %in% c("New1", "New2", "New3", "New4")) %>%
  ggplot() +
  #facet_wrap(~pred2) +
  geom_point(aes(x = UMAP1, y = UMAP2, col = pred2, size=prob2), alpha = .6) +
  scale_size(range = c(0.01, 1.5)) +
  ylim(-3,3) +
  xlim(-5,3) +
  scale_colour_manual(values = pal[c(7,8,9,10)]) +
  theme_classic() +
  theme(legend.title = element_blank(),
        plot.subtitle = element_text(hjust = 0.5,face = "bold",size = 10),
        legend.position = "none",
        panel.grid.major = element_line(color = "gray88", size = 0.2)) +
  labs(subtitle = "Newly discovered protein groups",
       tag = "D")

p_blank <- plot_spacer()

(p1|p2|p_blank) / (p3 |p4|p_blank) + plot_layout(widths = c(1, 1, 0.2))
```



Protein-wise responsibilities heatmap (Figure 7)

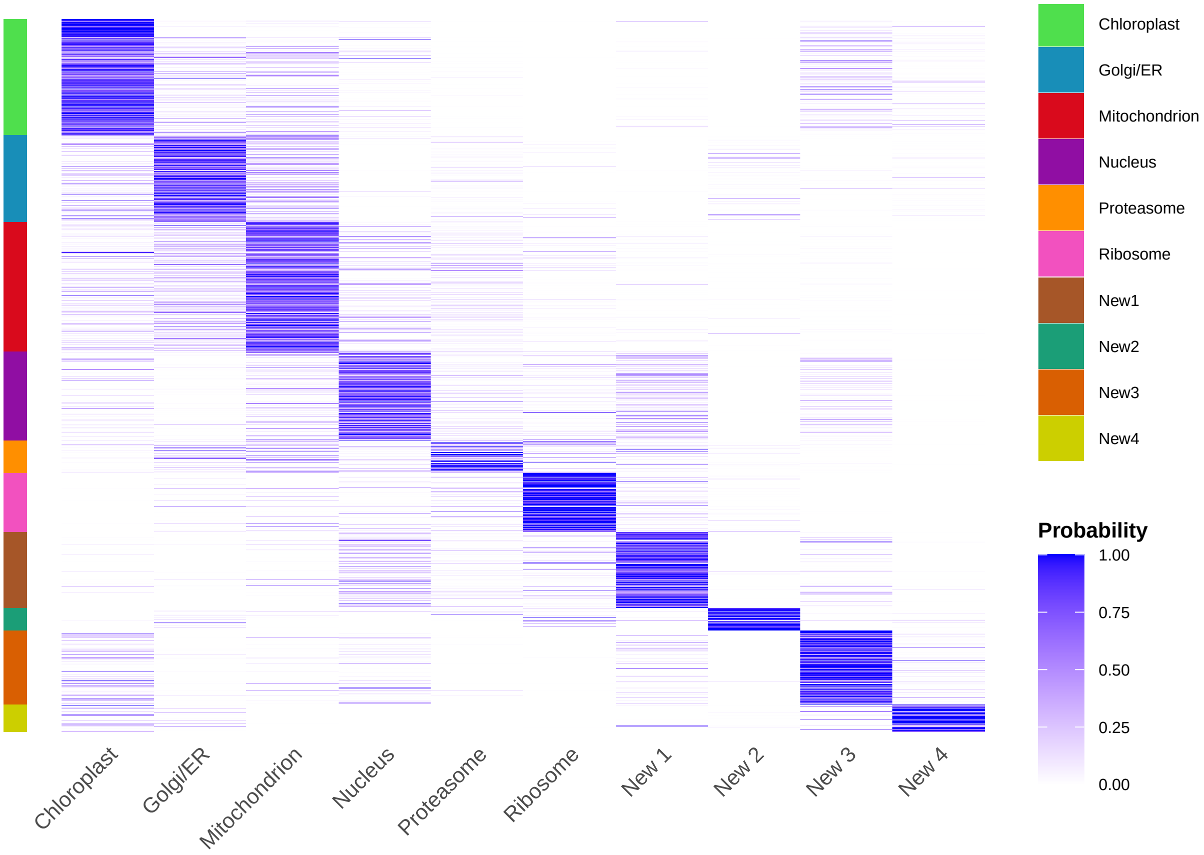
You can directly use the bulit-in function *draw_heatmap* in FSPmix package to visualize the membership. But you need to setup the label and the dataset.

```
draw_heatmap(res1)
```

Just for a better annotation purpose, here is the raw code generating figure 6

```
mat <- res2$result$resp
pred <- prediction(res2)

heat_df <- mat %>%
  as.data.frame() %>%
```

```r
  rownames_to_column(var = "id") %>%
  mutate(label = pred$predict_label) %>%
  pivot_longer(
    cols      = -c(id, label),
    names_to  = "fraction",
    values_to = "probability"
  ) %>%
  arrange(label) %>%
  mutate(
    id       = factor(id,       levels = rev(unique(id))),
    fraction = factor(fraction, levels = unique(fraction)),
    label = factor(label)
  )


ggplot(heat_df, aes(x = fraction, y = id, fill = probability)) +
  geom_tile() +
  scale_fill_gradient(high = "blue",
                      low = "white",
                      name = "Probability",
                      guide = guide_colorbar(order = 2) ) +
  ggnewscale::new_scale_fill() +
  geom_tile(aes(x = 0, fill = factor(heat_df$label)), width = 0.25) +
  scale_fill_manual(values = pal,
                    label = c("Chloroplast", "Golgi/ER", "Mitochondrion",
                   "Nucleus", "Proteasome", "Ribosome",
                   "New1", "New2", "New3", "New4"),
                    name = "Predicted group",
                    guide = guide_legend(order = 1)) +
  scale_x_discrete(labels = c("Chloroplast", "Golgi/ER", "Mitochondrion",
                  "Nucleus", "Proteasome", "Ribosome",
                  "New 1", "New 2", "New 3", "New 4")) +
  theme_minimal() +
  theme(axis.title  = element_blank(),
        axis.text.x  = element_text(angle = 45, hjust = 1, size = 8),
        legend.text = element_text(size = 6),
        legend.title = element_text(size = 8, face = "bold"),
        axis.text.y  = element_blank(),
        axis.ticks.y  = element_blank(),
        panel.grid      = element_blank())
```



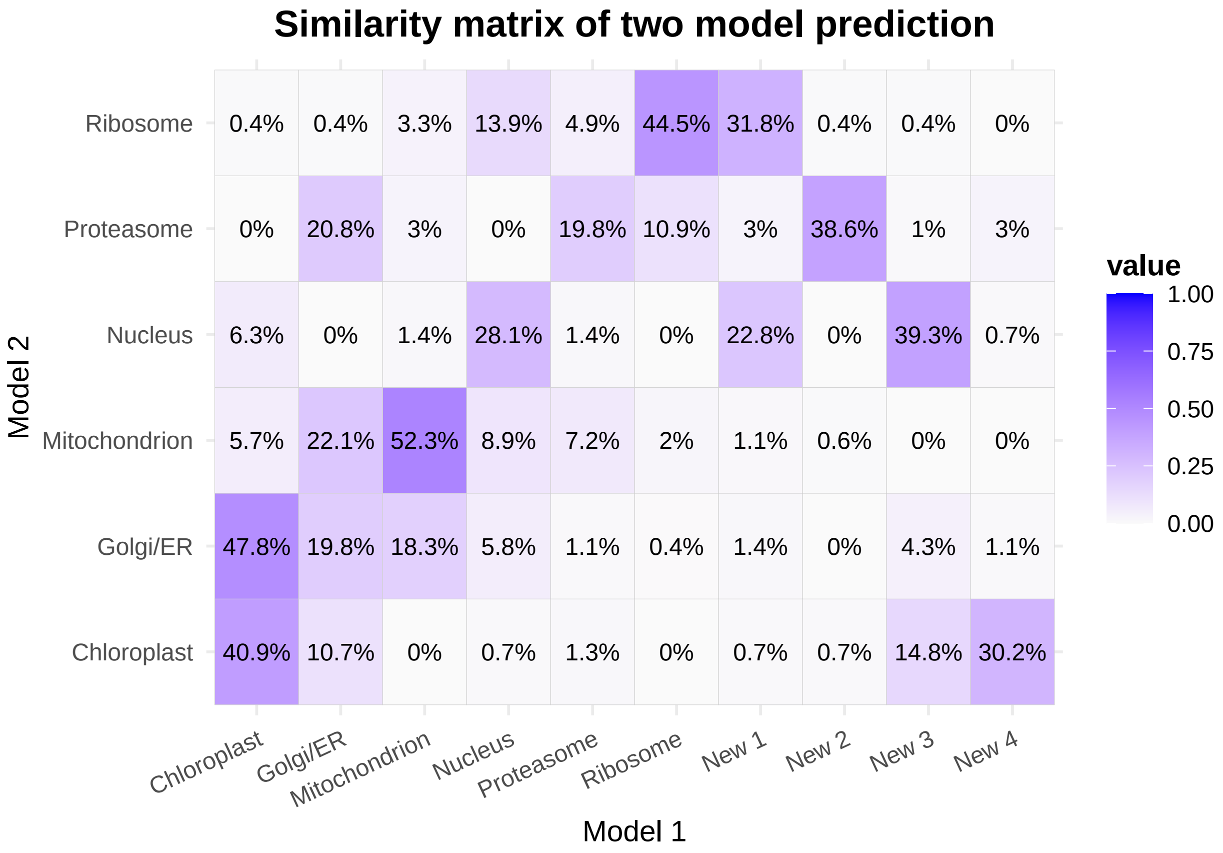
Heatmap of group similarity (Figure 6 bottom panel) and Alluvial graph (Appendix Figure S2)

You can directly use the bulit-in function *comparison_visual* in FSPmix package to compare two model results.

```
library(ggalluvial)
comparison_visual(res1,res2)
```

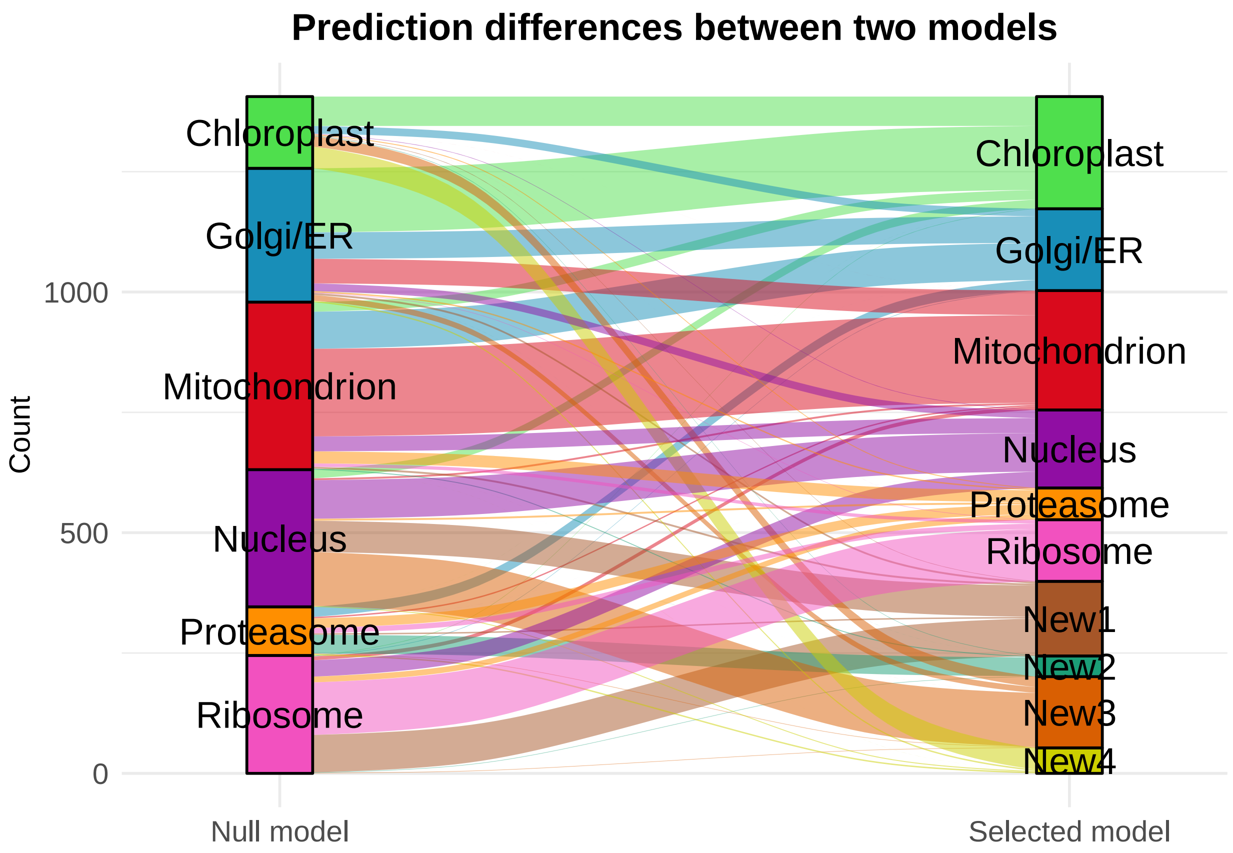
```
library(ggalluvial)
pred1 = prediction(res1)
pred2 = prediction(res2)

# make contingency table
make_contingency_table<-function(mem1, mem2){
  tab = table(mem1, mem2)
  tab = cbind(tab, rowSums(tab))
  tab = rbind(tab, colSums(tab))
  return(tab)
}

tab = make_contingency_table(pred1$predict_label, pred2$predict_label)
tab_clean <- tab[1:(nrow(tab) - 1), 1:(ncol(tab) - 1)]
rtab_clean = tab_clean / rowSums(tab_clean)
```

```
rtab_long <- as.data.frame(rtab_clean) %>%
  mutate(from = rownames(rtab_clean)) %>%
  pivot_longer(cols = -from, names_to = "to", values_to = "value") %>%
  mutate(to = factor(.$to, levels = colnames(tab_clean)))

# Heatmap
heatmap <- ggplot(rtab_long, aes(x = to, y = from, fill = value)) +
  geom_tile(color = "lightgrey") +
  geom_text(aes(label = paste0(round(value * 100, 1), "%")), color = "black", size = 3) +
  scale_fill_gradient(limits = c(0, 1), low = grey(0.98), high = "blue") +
  scale_x_discrete(labels = c("Chloroplast", "Golgi/ER", "Mitochondrion", "Nucleus", "Proteasome", "Ribo
  scale_y_discrete(labels = c("Chloroplast", "Golgi/ER", "Mitochondrion", "Nucleus", "Proteasome", "Ribo
  labs(
    x = "Model 1",
    y = "Model 2",
    title = "Similarity matrix of two model prediction"
  )+
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5,size = 14,face = "bold"),
        axis.text.x = element_text(angle = 25, vjust = 1,hjust = 1),
        plot.margin = margin(t = 5, r = 5, b = 5, l = 5),
        legend.title = element_text(face = "bold"))

heatmap
```



	Chloroplast	Golgi/ER	Mitochondrion	Nucleus	Proteasome	Ribosome	New 1	New 2	New 3	New 4
Ribosome	0.4%	0.4%	3.3%	13.9%	4.9%	44.5%	31.8%	0.4%	0.4%	0%
Proteasome	0%	20.8%	3%	0%	19.8%	10.9%	3%	38.6%	1%	3%
Nucleus	6.3%	0%	1.4%	28.1%	1.4%	0%	22.8%	0%	39.3%	0.7%
Mitochondrion	5.7%	22.1%	52.3%	8.9%	7.2%	2%	1.1%	0.6%	0%	0%
Golgi/ER	47.8%	19.8%	18.3%	5.8%	1.1%	0.4%	1.4%	0%	4.3%	1.1%
Chloroplast	40.9%	10.7%	0%	0.7%	1.3%	0%	0.7%	0.7%	14.8%	30.2%

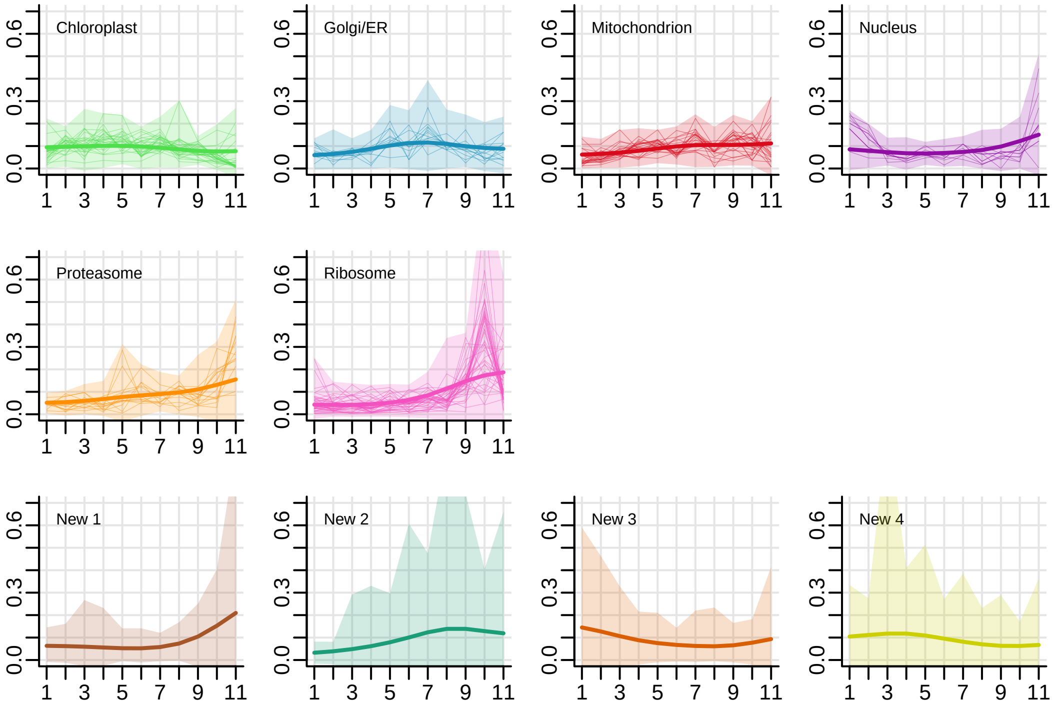
```
# Convert to counts of flows
targets = c("Chloroplast" , "Golgi/ER" ,  "Mitochondrion" ,"Nucleus"  , "Proteasome"   , "Ribosome", "Ne

l1 = targets[pred1$predict_label]
l2 = targets[pred2$predict_label]

df_counts <- as.data.frame(table(l1, l2))
colnames(df_counts) <- c("Model_1", "Model_2", "Freq")

# Alluvial plot
alluvia <- ggplot(df_counts,
       aes(axis1 = Model_1, axis2 = Model_2, y = Freq)) +
  geom_alluvium(aes(fill = Model_2),  width = 1/12) +
  geom_stratum(width = 1/12, aes(fill = after_stat(stratum)), color = "black") +
  geom_text(stat = "stratum",
            aes(label = after_stat(stratum)),
            size = 5) +
  scale_x_discrete(limits = c("Null model", "Selected model"), expand = c(.1, .1)) +
  scale_fill_manual(values = c(
    "Chloroplast" = "#4FDF4D",
    "Golgi/ER" = "#188EB8",
    "Mitochondrion" = "#D90A1C",
    "Nucleus" = "#900EA3",
    "Proteasome" = "#FF8F00",
    "Ribosome" = "#F251BF",
    "New1" = "#A65628",
    "New2" = "#1B9E77",
    "New3" = "#D95F02",
    "New4" = "#CCCF00"
  )) +
  theme_minimal() +
  labs(y = "Count", title = "Prediction differences between two models") +
  theme(legend.position = "none",
        plot.title = element_text(hjust = 0.5,size = 14, face = "bold"),
        axis.text = element_text(size = 11))

alluvia
```



Interence. Making prediciton bands (Figure 8)

Again, function *create_hpb* was a bulit-in function to show the prediction bands in FSPmix package, and here we present the raw code.

```
# Getting the bands
res = res2
pred = prediction(res2)
K <- 10
p <- 11
alpha = 0.05
bands = array(dim = c(2 , p , K))
pro = c("Chloroplast", "Golgi/ER", "Mitochondrion",
                  "Nucleus", "Proteasome", "Ribosome",
        "New 1", "New 2", "New 3", "New 4")

for(k in 1:K){
  x = res$data[which(pred$predict_label == k),]
  for(d in 1:p){
    h = res$result$sigma[k,d] * length(x[,d])^(-0.2)
    pad = 10*h
    rng = apply(x, 2, range)
    probs = c( (alpha/p)/2 , 1 - (alpha/p)/2 )
    interval <- matrix(c(rng[1,] - pad , rng[2,] + pad) , nrow = 2, byrow = TRUE)
    Fhat <- function(z) mean(pnorm((z - x[,d]) / h))
    qfun <- function(p) uniroot(function(z) Fhat(z) - p, interval = interval[,d])$root
    bands[,d,k] <- vapply(probs, qfun, numeric(1))
  }
}

# Draw the mean lines and HPB
par(mfrow=c(3,4),
    mar  = c(1.5, 1.5, 1.5, 1),
    oma  = c(2, 6, 3, 1))
for(i in 1:6){
  mean_line <- res$result$mu[i,]
  var_all <- res$result$sigma[i,]
  x <- 1:11
  par(mgp = c(3, 0.5, 0))
  plot(x, t(mean_line), type = "l", ylim = c(0,0.7),lty = 1,
       lwd = 2, pch = 20, col = pal[i],
       xlab = "",
       ylab = "",
       xaxt="n",
       bty="l",
       panel.first = {
         abline(h = axTicks(2),  col = "grey90", lwd = 1)
         abline(v = 1:11,  col = "grey90", lwd = 1)
       })
  legend(x = -0.1 , y = 0.72,
         pro[i],
         cex = 0.8,
         bty = "n",
         text.font = 0.8)
  upper <- bands[2,,i]
  lower <- bands[1,,i]
  polygon(c(x, rev(x)), c(upper, rev(lower)),
          col = adjustcolor(pal[i], alpha.f = 0.2), border = NA)
  axis(1, at = 1:11, labels = 1:11)
  # Addding the annotated data
  annota <- data[label[label$label==i,]$index,]
  for(j in 1:dim(annota)[1]){
    lines(x,annota[j,],col = adjustcolor(pal[i], alpha.f = 0.5), lwd = 0.3)
  }

}

for(i in 1:2){
  plot.new()
}
for(i in 7:10){
  mean_line <- res$result$mu[i,]
  var_all <- res$result$sigma[i,]
  x <- 1:11
  plot(x, t(mean_line), type = "l", ylim = c(0,0.7),
       lty = 1, lwd = 2, pch = 20, col = pal[i],
       xlab = "",
       ylab = "",
       xaxt="n",
       bty="l",
       panel.first = {
         abline(h = axTicks(2),  col = "grey90", lwd = 1)
         abline(v = 1:11,  col = "grey90", lwd = 1)
       })
  legend(x = -0.1 , y = 0.72,
         pro[i],
         cex = 0.8,
         bty = "n",
         text.font = 0.8)
  upper <- bands[2,,i]
  lower <- bands[1,,i]
  polygon(c(x, rev(x)), c(upper, rev(lower)),
          col = adjustcolor(pal[i], alpha.f = 0.2), border = NA)
  axis(1, at = 1:11, labels = 1:11)
}
```



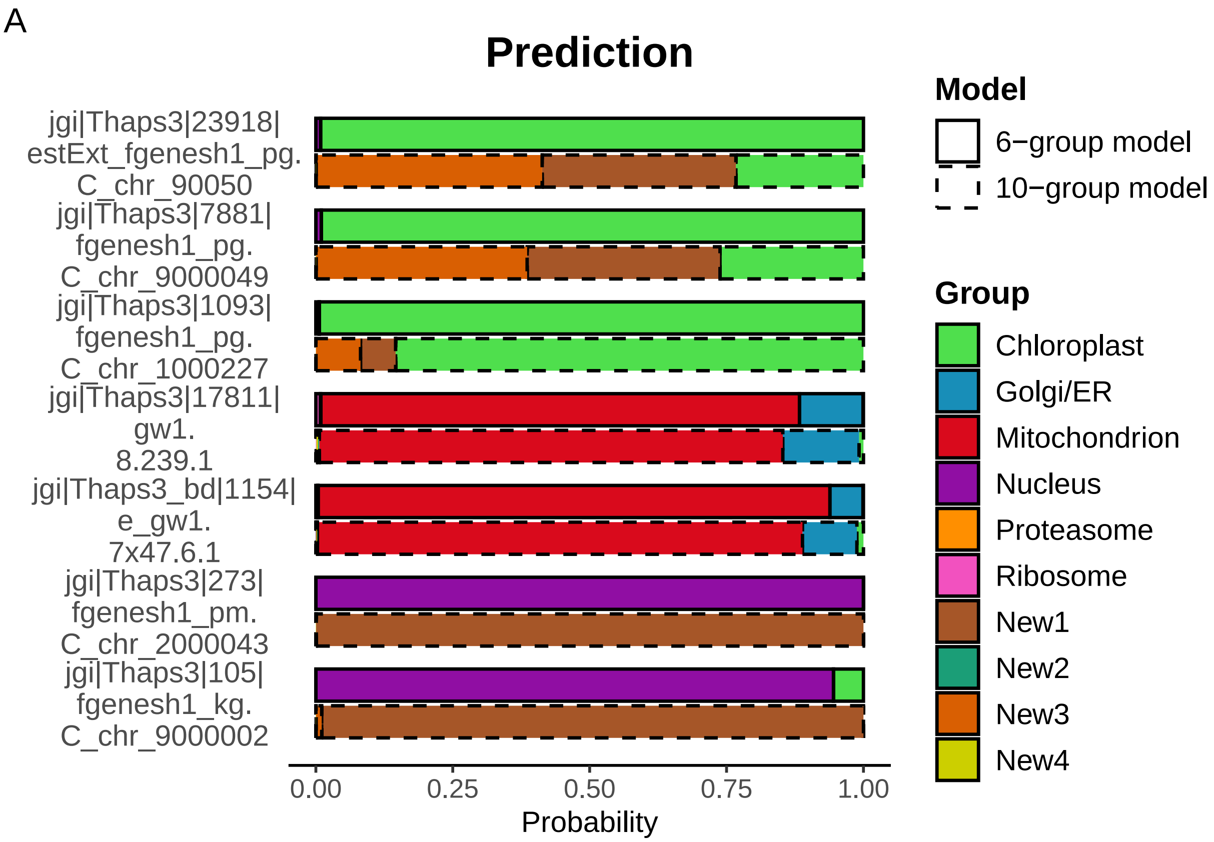
Specific proteins prediction example(Figure 9)

We select a small set of proteins and compare their results on null model and selected model. Also visualize the location of these proteins in UMAP space.

```
library(ggrepel)
load(file.path("./paper_exp_dat//model_result_remove_ABC_NM.RData"), verbose = TRUE)
```

```
## Loading objects:
##   noK0_result
##   bestK0_result
```

```
my_list = c("jgi|Thaps3|105|fgenesh1_kg.C_chr_9000002",
            "jgi|Thaps3|273|fgenesh1_pm.C_chr_2000043",
            "jgi|Thaps3_bd|1154|e_gw1.7x47.6.1",
            "jgi|Thaps3|17811|gw1.8.239.1",
            "jgi|Thaps3|1093|fgenesh1_pg.C_chr_1000227",
            "jgi|Thaps3|7881|fgenesh1_pg.C_chr_9000049",
            "jgi|Thaps3|23918|estExt_fgenesh1_pg.C_chr_90050")
raw_dat <- Thaps2024
my_list_id <- match(my_list,rownames(raw_dat))


prob_mat1 <- noK0_result$result$resp[my_list_id,]
prob_mat2 <- bestK0_result$result$resp[my_list_id,]
colnames(prob_mat1) = groups[1:6]
colnames(prob_mat2) = groups


df_long1 <- prob_mat1 %>%
  as.data.frame() %>%
  mutate(id = factor(1:nrow(.)), model = "6-group model") %>%
  pivot_longer(-c(id, model), names_to = "group", values_to = "prob")
df_long2 <- prob_mat2 %>%
  as.data.frame() %>%
  mutate(id = factor(1:nrow(.)), model = "10-group model") %>%
  pivot_longer(-c(id, model), names_to = "group", values_to = "prob")

df <- bind_rows(df_long1, df_long2)
df <- df %>%
  mutate(bar = interaction(id, model, sep = "_"))%>%
  mutate(bar = factor(bar, levels = c(outer(levels(df$id)[7:1],
                                  c("6-group model","10-group model"),
                                  FUN = paste, sep = "_")))) %>%
  mutate(y = as.numeric(id) + if_else(model == "6-group model", 0.2, -0.2))

bar <- ggplot(df, aes(x = y, y = prob, fill = group)) +
  geom_col(
    aes(linetype = model),
    colour = "black",
    width  = 0.35,
    size   = 0.6
  ) +
  coord_flip() +
  scale_fill_manual(values = pal,
                    breaks = groups) +
  scale_linetype_manual(
    values = c("6-group model" = "solid",
               "10-group model" = "dashed"),
    breaks = c("6-group model", "10-group model"),
    name   = "Model"
  )+
  scale_x_continuous(
    breaks = 1:7,
    labels = sub("\\.", ".\n",
                 sub("^(([^|]*\\|){3})", "\\1\n", my_list)),
    expand = expansion(add = 0.3)
  ) +
  guides(
    linetype = guide_legend(
      override.aes = list(fill = NA)
    )
  )+
  labs(x = NULL,
       y = "Probability",
       fill = "Group",
       title = "Prediction",
       tag = "A") +
  theme_classic() +
  theme(axis.ticks.y = element_blank(),
    plot.title = element_text(size = 16,
                              hjust = 0.5,
                              face = "bold"),
    axis.line.y  = element_blank(),
    axis.text.x = element_text(size = 10),
    axis.text.y = element_text(size = 11,
                               hjust = 0.5,
                               margin = margin(r = -8)),
    legend.text = element_text(size = 11),
    legend.title = element_text(size = 12,face = "bold"),
    legend.spacing.y = unit(0.5, "cm") )



u <- ggplot(full_data) +
  geom_point(
    aes(x = UMAP1, y = UMAP2, colour = annotation_level1,
        alpha  = ifelse(is.na(annotation_level1), 0.1, 0.8) )
  ) +
  geom_point(
    data = function(d) d[rownames(d) %in% my_list_id, ],
    aes(x = UMAP1, y = UMAP2), shape  = 24,colour = "black", size  = 3
  ) +
  scale_colour_manual(values = pal) +
  scale_alpha_identity(guide = "none") +   # keeps the legend tidy
  theme_classic() +
  theme(
    plot.title = element_text(size = 16,
                              hjust = 0.5,
                              face = "bold"),
    legend.title = element_text(size = 11, face = "bold", hjust = 0.3),
    legend.text = element_text(size = 9),
    legend.key.size = unit(0.8, "lines"),
    legend.position = c(0.18, 0.23),
    legend.background      = element_rect(fill = "transparent", colour = NA),
    legend.box.background  = element_rect(fill = "transparent", colour = NA),
    panel.grid.major = element_line(colour = "gray88", linewidth = 0.2)
  ) +
  geom_text_repel(
    data = function(d) {
      row_num <- as.numeric(rownames(d))
      keep <- row_num %in% my_list_id
      ss   <- d[keep, , drop = FALSE]
      ss$lbl <- my_list[match(row_num[keep], my_list_id)]
      ss
    },
    aes(x = UMAP1, y = UMAP2, label = lbl),
    size       = 4,
    colour     = "black",
    box.padding = 1,
    min.segment.length = 0
  ) +
  labs(colour = "Annotated proteins",
      title = "Annotated dataset",
      tag = "B")

bar
```


A
Prediction
jgi|Thaps3|23918|
estExt_fgenesh1_pg.
C_chr_90050
jgi|Thaps3|7881|
fgenesh1_pg.
C_chr_9000049
jgi|Thaps3|1093|
fgenesh1_pg.
C_chr_1000227
jgi|Thaps3|17811|
gw1.
8.239.1
jgi|Thaps3_bd|1154|
e_gw1.
7x47.6.1
jgi|Thaps3|273|
fgenesh1_pm.
C_chr_2000043
jgi|Thaps3|105|
fgenesh1_kg.
C_chr_9000002
0.00
0.25
0.50
0.75
1.00
Probability
Model
6−group model
10−group model
Group
Chloroplast
Golgi/ER
Mitochondrion
Nucleus
Proteasome
Ribosome
New1
New2
New3
New4