Y-permutation Plot

Saw Simeon, Nuttapat Anuwongcharoen, Watshara Shoombuatong, Aijaz Ahmad Malik, Virapong Prachayasittikul, Jarl E. S. Wikberg and Chanin Nantasenamat

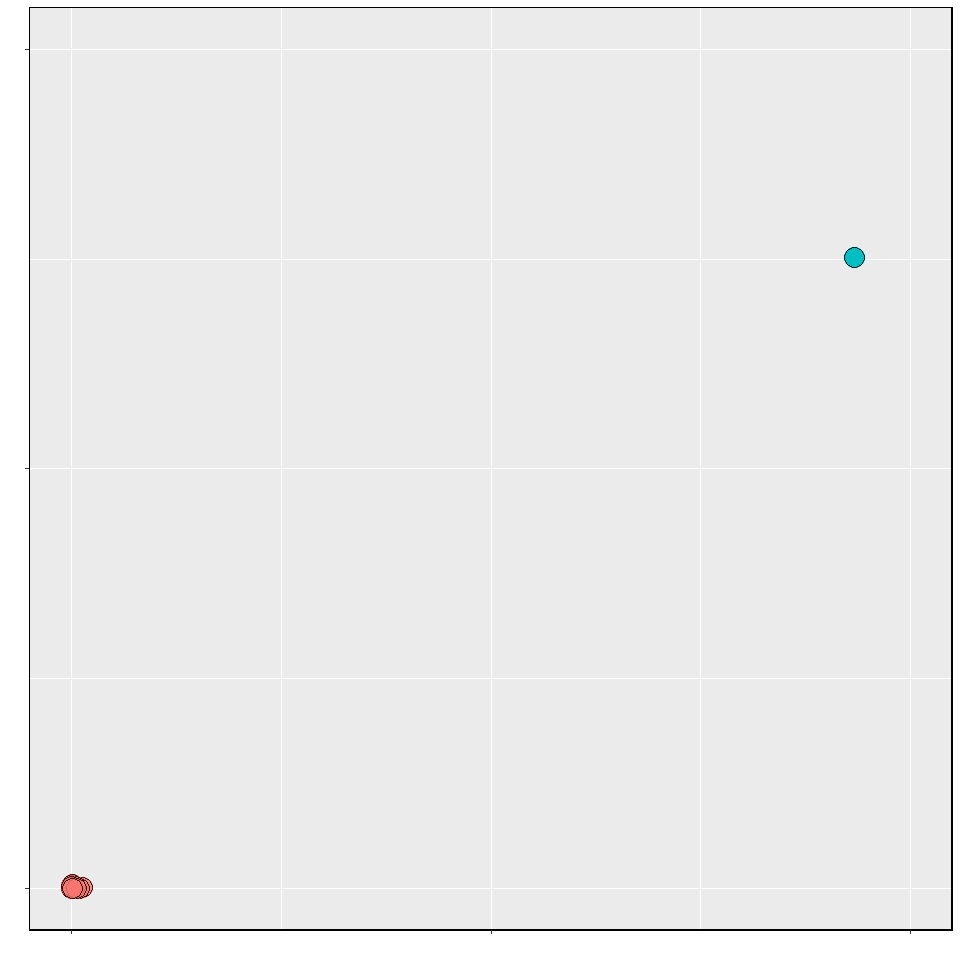
June 10, 2016

## Function For Creating the Scrambling Plot

scrambling\_R2 <- function(x) {  
 library(doSNOW)  
 library(foreach)  
 library(parallel)  
 cl <- makeCluster(8)  
 registerDoSNOW(cl)  
   
 results <- list(50)  
 results <- foreach (i = 1:50) %dopar% {  
 x <- na.omit(x)  
 para <- dplyr::sample\_n(x, size = 2186, replace = TRUE)  
 in\_train\_para <- sample(nrow(para),  
 size = as.integer(nrow(para) \* 0.8),  
 replace = FALSE)  
 Train <- para[in\_train\_para, ]  
 Test <- para[-in\_train\_para, ]  
 pIC50 <- gtools::permute(Train$pIC50)  
 pIC50 <- data.frame(pIC50)  
 fake\_data <- cbind(pIC50, Train[, 2:ncol(Train)])  
 #ctrl <- caret::trainControl(method = "repeatedcv", number = 10, repeats = 1)  
 #tune <- caret::train(pIC50~., data = fake\_data, method = "rf",  
 # trControl = ctrl, tuneLength = 10)  
 fit <- ranger::ranger(pIC50~., data = fake\_data, write.forest = TRUE, save.memory = TRUE)  
 #actual <- train$Activity  
 prediction <- predict(fit, Train)  
 prediction <- prediction$predictions  
 value <- data.frame(obs = Train$pIC50, pred = prediction)  
   
 rm(fit)  
 rm(prediction)  
 labeling <- c("obs", "pred")  
 colnames(value) <- labeling  
 results[[i]] <- caret::defaultSummary(value)  
 }  
 R2 <- data.frame(results)  
 R2 <- t(R2)  
 R2 <- as.numeric(R2[,2])  
 R2 <- round(R2, digits = 5)  
 return(R2)  
 stopCluster(cl)  
}  
  
real\_R2 <- function(x) {  
 x <- na.omit(x)  
 para <- dplyr::sample\_n(x, size = 2186, replace = TRUE)  
 in\_train\_para <- sample(nrow(para),  
 size = as.integer(nrow(para) \* 0.8),  
 replace = FALSE)  
 Train <- para[in\_train\_para, ]  
 Test <- para[-in\_train\_para, ]  
 #ctrl <- caret::trainControl(method = "repeatedcv", number = 10, repeats = 1)  
 #tune <- train(pIC50~., data = x, method = "rf",  
 # trControl = ctrl, tuneLength = 10)  
 fit <- ranger::ranger(pIC50~., data = Train, write.forest = TRUE, save.memory = TRUE)  
 #actual <- train$Activity  
 prediction <- predict(fit, Train)  
 prediction <- prediction$predictions  
 value <- data.frame(obs = Train$pIC50, pred = prediction)  
 labeling <- c("obs", "pred")  
 colnames(value) <- labeling  
 result <- caret::defaultSummary(value)  
 R2 <- as.data.frame(result)  
 R2 <- R2[2, ]  
 R2 <- round(R2, digits = 5)  
 return(R2)  
}  
  
scrambling\_Q2 <- function(x) {  
 library(doSNOW)  
 library(foreach)  
 library(parallel)  
 cl <- makeCluster(8)  
 registerDoSNOW(cl)  
   
 results <- list(50)  
 results <- foreach (i = 1:50) %dopar% {  
 para <- dplyr::sample\_n(x, size = 2186, replace = TRUE)  
 in\_train\_para <- sample(nrow(para),  
 size = as.integer(nrow(para) \* 0.8),  
 replace = FALSE)  
 Train <- para[in\_train\_para, ]  
 Test <- para[-in\_train\_para, ]  
 pIC50 <- gtools::permute(Train$pIC50)  
 pIC50 <- data.frame(pIC50)  
 myData <- cbind(pIC50, Train[, 2:ncol(x)])  
 k = 10  
 index <- sample(1:k, nrow(myData), replace = TRUE)  
 folds <- 1:k  
 myRes = data.frame()  
 for (j in 1:k) {  
 training <- subset(myData, index %in% folds[-j])  
 testing <- subset(myData, index %in% c(j))  
 #pIC50 <- gtools::permute(training$pIC50)  
 #pIC50 <- data.frame(pIC50)  
 #fake\_data <- cbind(pIC50, training[2:ncol(training)])  
 #ctrl <- caret::trainControl(method = "repeatedcv", number = 10, repeats = 1)  
 #tune <- caret::train(pIC50~., data = training, method = "rf",  
 # trControl = ctrl, tuneLength = 10)  
 fit <- ranger::ranger(pIC50~., data = training, write.forest = TRUE, save.memory = TRUE)  
 #actual <- train$Activity  
 prediction <- predict(fit, testing)  
 prediction <- prediction$predictions  
 value <- data.frame(obs = testing$pIC50, pred = prediction)  
   
 #fit <- randomForest::randomForest(pIC50~., data = training, mtry = tune$bestTune[[1]])  
 #prediction <- predict(fit, testing)  
 #value <- data.frame( obs = testing$pIC50, pred = prediction)  
 myRes <- rbind(myRes, value)}  
 value <- myRes  
 labeling <- c("obs", "pred")  
 rm(tune)  
 rm(fit)  
 rm(prediction)  
 rm(ctrl)  
 colnames(value) <- labeling  
 results[[i]] <- caret::defaultSummary(value)  
 }  
 Q2 <- data.frame(results)  
 Q2 <- t(Q2)  
 Q2 <- as.numeric(Q2[,2])  
 Q2 <- round(Q2, digits = 5)  
 return(Q2)  
 stopCluster(cl)  
}  
  
  
real\_Q2 <- function(x) {  
 para <- dplyr::sample\_n(x, size = 2570, replace = TRUE)  
 in\_train\_para <- sample(nrow(para),  
 size = as.integer(nrow(para) \* 0.8),  
 replace = FALSE)  
 Train <- para[in\_train\_para, ]  
 Test <- para[-in\_train\_para, ]  
 myData <- Train  
 k = 10  
 index <- sample(1:k, nrow(myData), replace = TRUE)  
 folds <- 1:k  
 myRes <- data.frame()  
 for (j in 1:k) {  
 training <- subset(myData, index %in% folds[-j])  
 testing <- subset(myData, index %in% c(j))  
 #ctrl <- caret::trainControl(method = "repeatedcv", number = 10, repeats = 1)  
 #tune <- train(pIC50~., data = training, method = "rf",  
 # trControl = ctrl, tuneLength = 10)  
 fit <- ranger::ranger(pIC50~., data = training, write.forest = TRUE, save.memory = TRUE)  
 #actual <- train$Activity  
 prediction <- predict(fit, testing)  
 prediction <- prediction$predictions  
 value <- data.frame(obs = testing$pIC50, pred = prediction)  
 #fit <- randomForest(pIC50~., data = training, mtry = tune$bestTune[[1]])  
 #prediction <- predict(fit, testing)  
 #value <- data.frame(obs = testing$pIC50, pred = prediction)  
 myRes <- rbind(myRes, value)  
 }  
 value <- myRes  
 labeling <- c("obs", "pred")  
 colnames(value) <- labeling  
 result <- caret::defaultSummary(value)  
 Q2 <- as.data.frame(result)  
 Q2 <- Q2[2, ]  
 Q2 <- round(Q2, digits = 5)  
 return(Q2)  
}  
  
data\_pre <- function(x) {  
 fake\_R2 <- scrambling\_R2(x)  
 real\_R2 <- real\_R2(x)  
 fake\_Q2 <- scrambling\_Q2(x)  
 real\_Q2 <- real\_Q2(x)  
 fake\_R2 <- as.data.frame(fake\_R2)  
 fake\_R2$Label <- "Fake"  
 real\_R2 <- as.data.frame(real\_R2)  
 real\_R2$Label <- "Real"  
 fake\_Q2 <- as.data.frame(fake\_Q2)  
 fake\_Q2$Label <- "Fake"  
 real\_Q2 <- as.data.frame(real\_Q2)  
 real\_Q2$Label <- "Real"  
 fake <- cbind(fake\_R2, fake\_Q2)  
 real <- cbind(real\_R2, real\_Q2)  
 combine <- data.frame(fake, real)  
 return(combine)  
}  
  
plot\_scrambling <- function(x,y){  
 ok <- data\_pre(x)  
 colnames(ok) <- c("R2", "Label", "Q2", "Label", "R2", "Label", "Q2", "Label")  
 R2 <- ok[c(1, 5)]  
 R2 <- reshape2::melt(R2)  
 R2 <- R2$value  
 R2 <- data.frame(R2)  
 Q2 <- ok[c(3, 7)]  
 Q2 <- reshape2::melt(Q2)  
 Q2 <- Q2$value  
 Q2 <- data.frame(Q2)  
 Label <- c("Fake", "Fake", "Fake", "Fake", "Fake", "Fake", "Fake", "Fake",  
 "Fake", "Fake", "Fake", "Fake", "Fake", "Fake", "Fake", "Fake",  
 "Fake", "Fake", "Fake", "Fake", "Fake", "Fake", "Fake", "Fake",  
 "Fake", "Fake", "Fake", "Fake", "Fake", "Fake", "Fake", "Fake",  
 "Fake", "Fake", "Fake", "Fake", "Fake", "Fake", "Fake", "Fake",  
 "Fake", "Fake", "Fake", "Fake", "Fake", "Fake", "Fake", "Fake",  
 "Fake", "Fake", "Real", "Real", "Real", "Real", "Real", "Real",  
 "Real", "Real", "Real", "Real", "Real", "Real", "Real", "Real",  
 "Real", "Real", "Real", "Real", "Real", "Real", "Real", "Real",  
 "Real", "Real", "Real", "Real", "Real", "Real", "Real", "Real",  
 "Real", "Real", "Real", "Real", "Real", "Real", "Real", "Real",  
 "Real", "Real", "Real", "Real", "Real", "Real", "Real", "Real",  
 "Real", "Real", "Real", "Real")  
 data <- cbind(R2, Q2, Label)  
 write.csv(data,y)  
 library(ggplot2)  
 plot <- ggplot(data, aes(x = R2, y = Q2, colour = Label)) +  
 geom\_point(size = 7, colour = "black", aes(fill = factor(Label)), pch = 21, alpha = 0.8) +  
 theme(  
 legend.position = ("none"),  
 axis.text = element\_blank(),  
 panel.border = element\_rect(linetype = "solid", colour = "black", fill = NA, size = 1)) +  
 xlab("") + ylab("") +   
 #labs(y = expression(paste(italic(Q^2)))) +  
 #labs(x = expression(paste(italic(R^2)))) +  
 scale\_x\_continuous(limits = c(0, 1), breaks = seq(from = 0, to = 1, by = 0.5)) +  
 scale\_y\_continuous(limits = c(0, 1), breaks = seq(from = 0, to = 1, by = 0.5))   
 coord\_cartesian(ylim = c(0, 1), xlim = c(0, 1))  
 return(plot)  
}

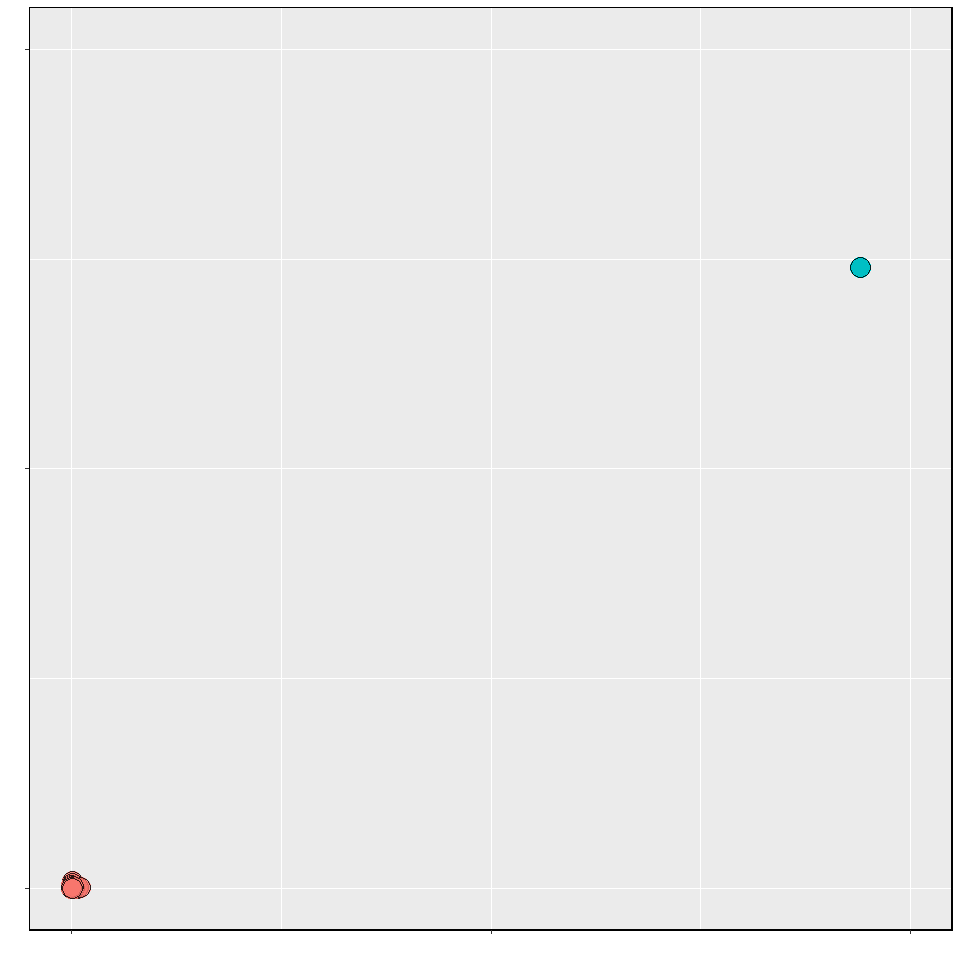
### CDK

input <- readRDS("fpdata.Rds")  
CDK\_Fingerprint <- input$FingerPrinter  
plot\_scrambling(CDK\_Fingerprint,"CDKS.csv")



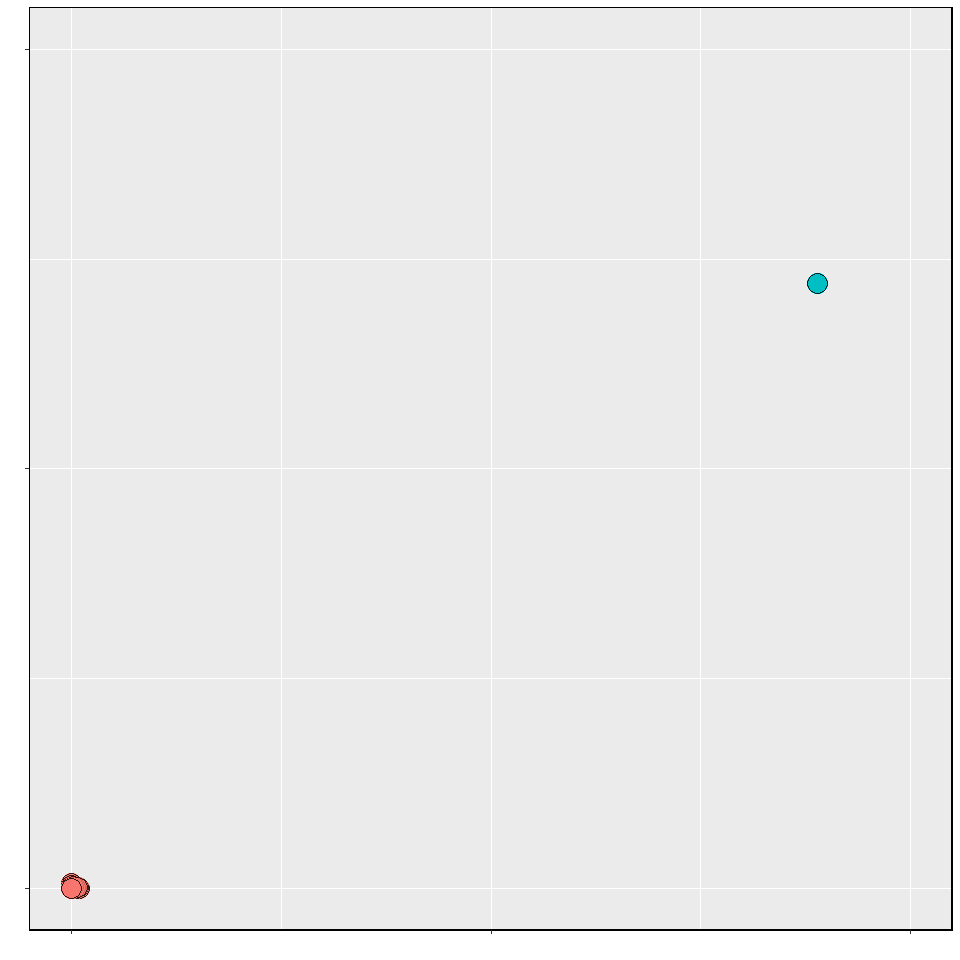
### CDK Extended

input <- readRDS("fpdata.Rds")  
Extended\_CDK\_Fingerprint <- input$Extended\_finterPrinter  
plot\_scrambling(Extended\_CDK\_Fingerprint,"ExtendS.csv")



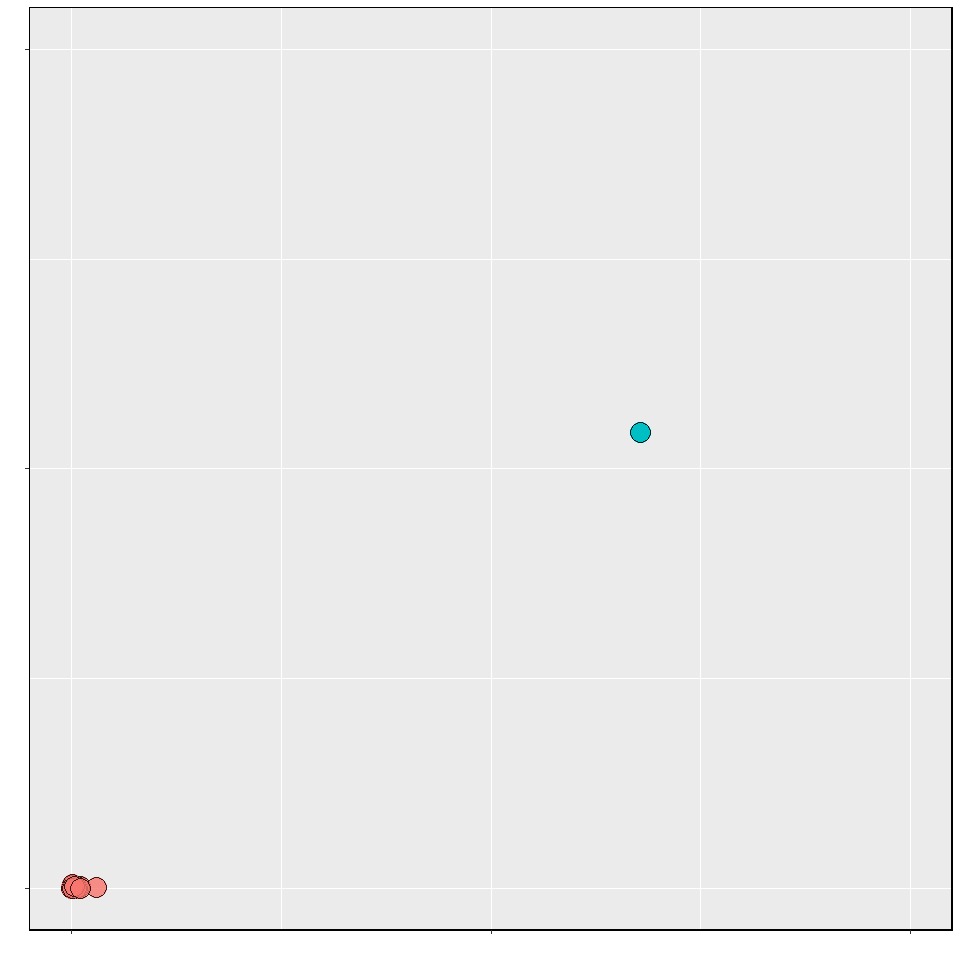
### CDK Graph Only

input <- readRDS("fpdata.Rds")  
Graph\_CDK\_Fingerprint <- input$GraphOnly\_FingerPrinter  
plot\_scrambling(Graph\_CDK\_Fingerprint,"GraphS.csv")



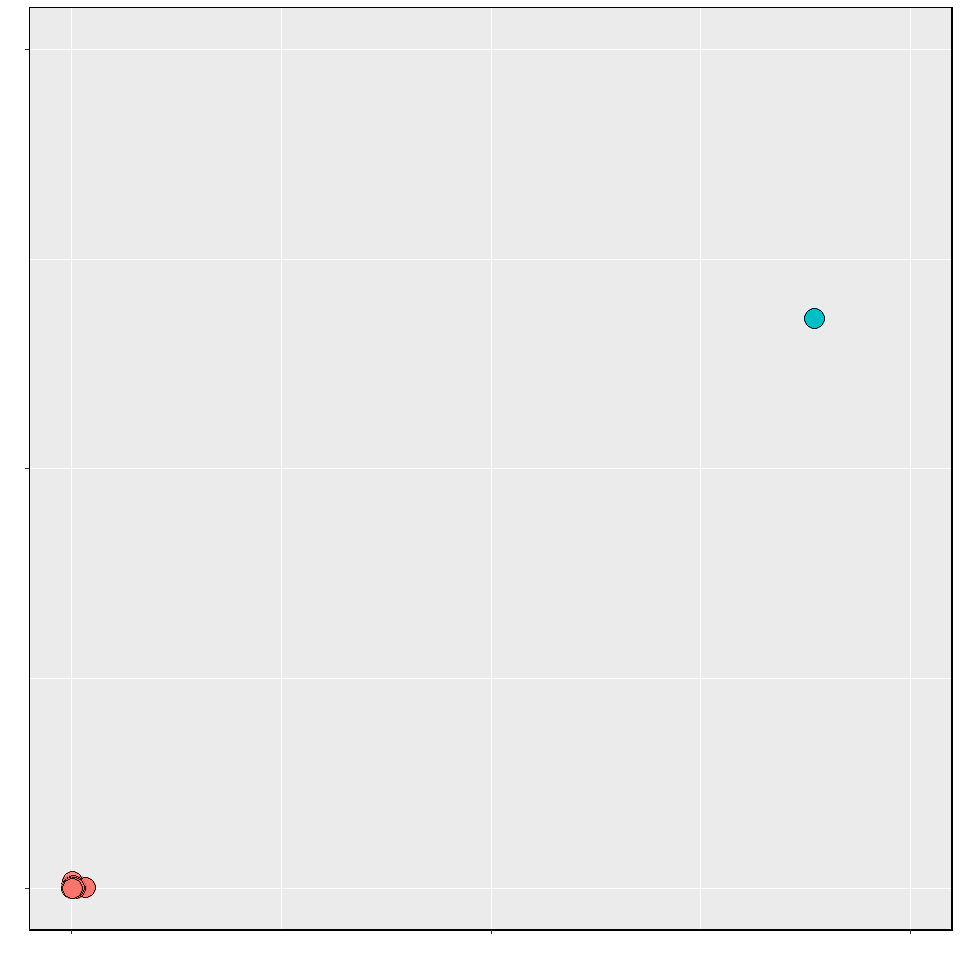
### E-State

input <- readRDS("fpdata.Rds")  
E\_state\_Fingerprint <- input$Estate\_FingerPrinter  
plot\_scrambling(E\_state\_Fingerprint,"EstateS.csv")



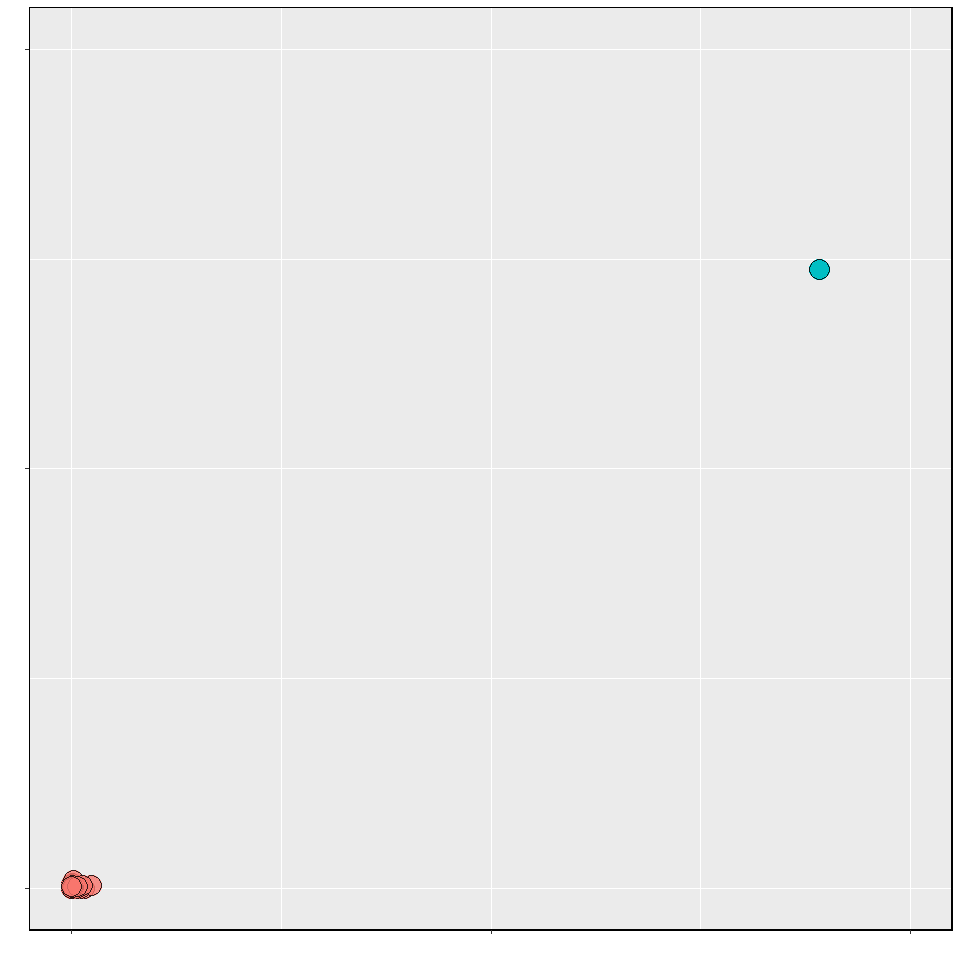
### MACCS

input <- readRDS("fpdata.Rds")  
MACCS\_Fingerprint <- input$MACCS\_FingerPrinter  
plot\_scrambling(MACCS\_Fingerprint,"MACCSS.csv")



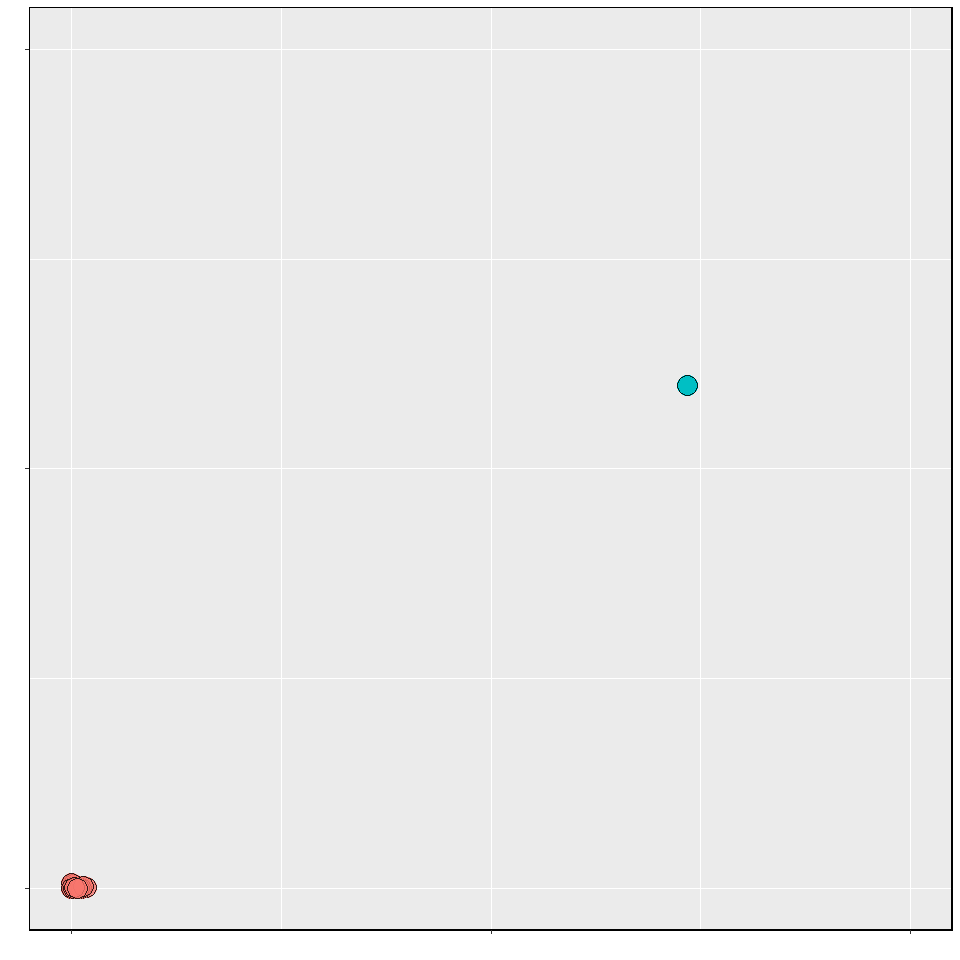
### PubChem

input <- readRDS("fpdata.Rds")  
PubChem\_Fingerprint <- input$Pubchem\_FingerPrinter  
plot\_scrambling(PubChem\_Fingerprint,"PubchemS.csv")



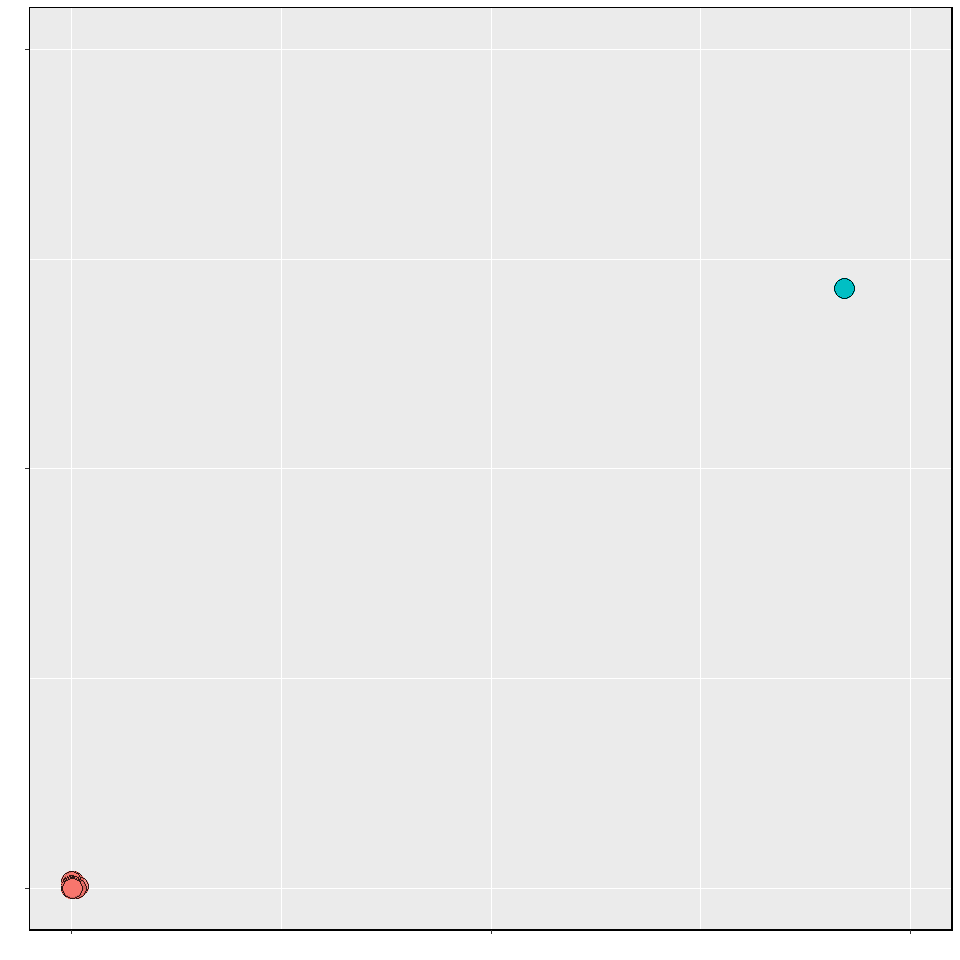
### Substructure

input <- readRDS("fpdata.Rds")  
Substructure <- input$Substructure\_fingerPrinter  
plot\_scrambling(Substructure,"SubsturesS.csv")



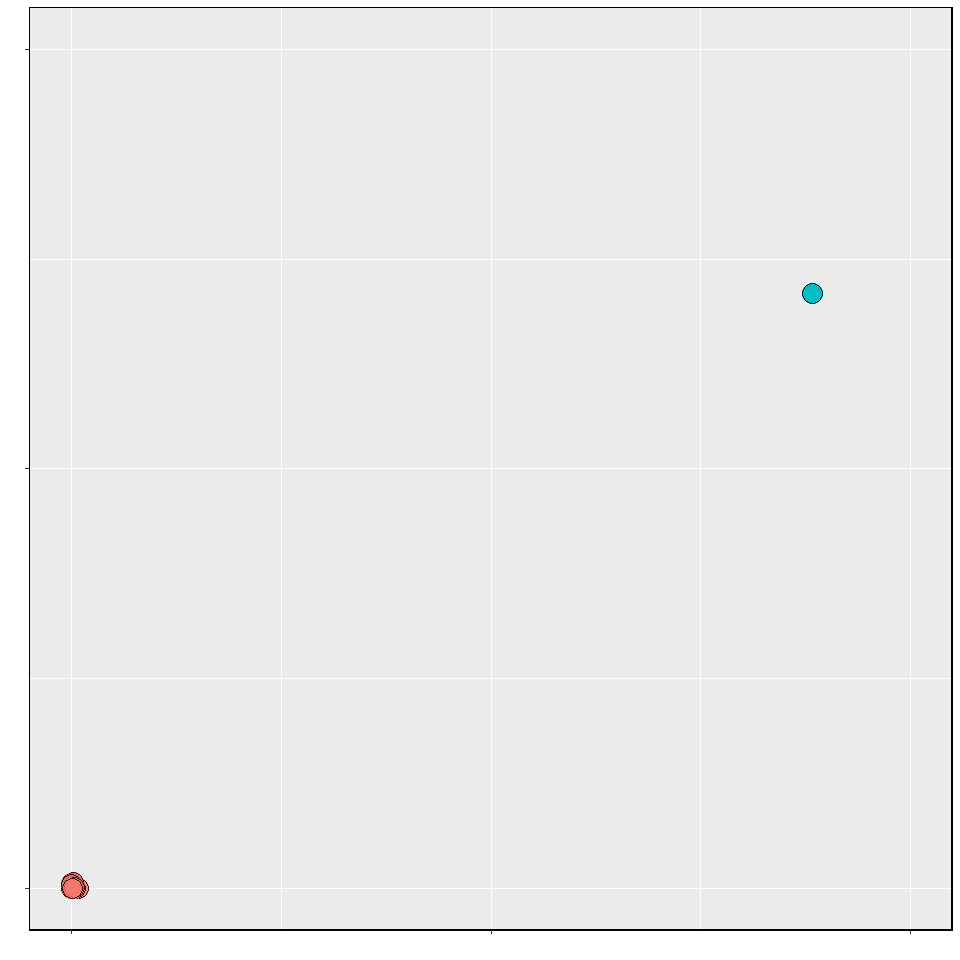
### Substructure Count

input <- readRDS("fpdata.Rds")  
Substructure\_Count <- input$Substructure\_fingerPrintCount  
plot\_scrambling(Substructure\_Count,"Substructure\_CountS.csv")



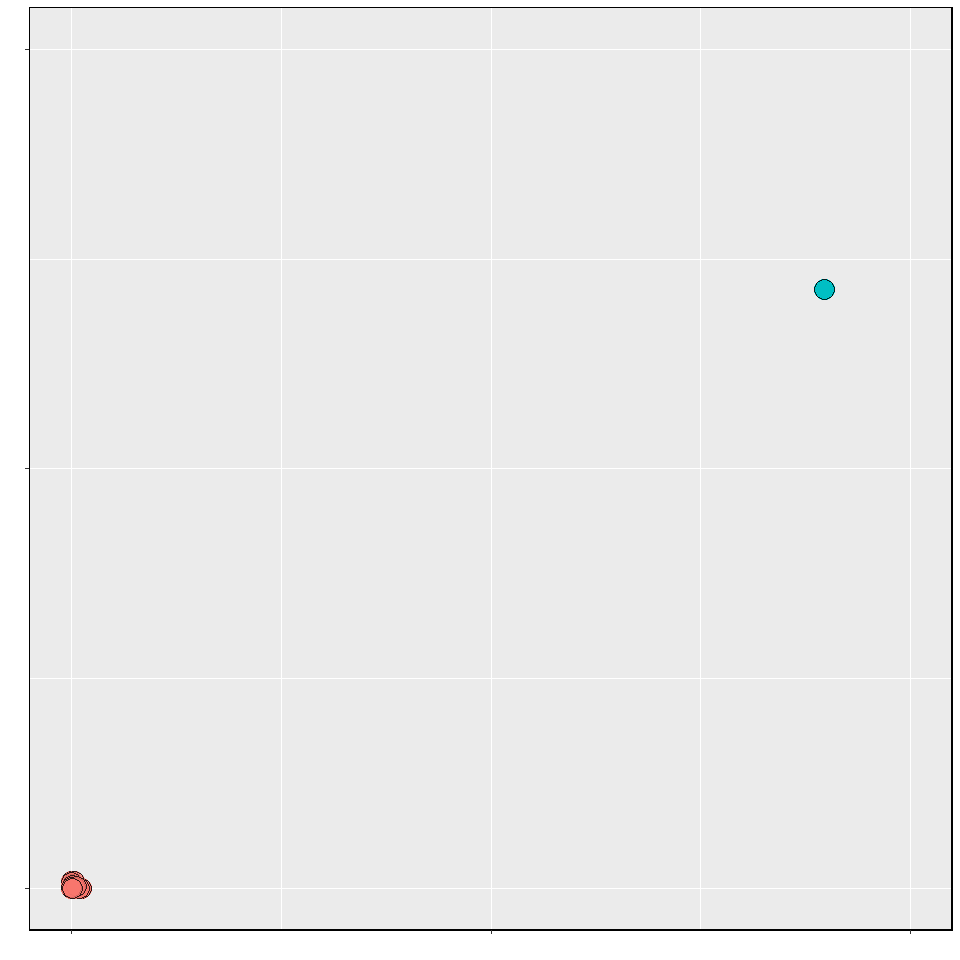
### Klekota-Roth

input <- readRDS("fpdata.Rds")  
Klekota\_Roth <- input$KlekotaRoth\_FingerPrinter  
plot\_scrambling(Klekota\_Roth,"KlekotaS.csv")



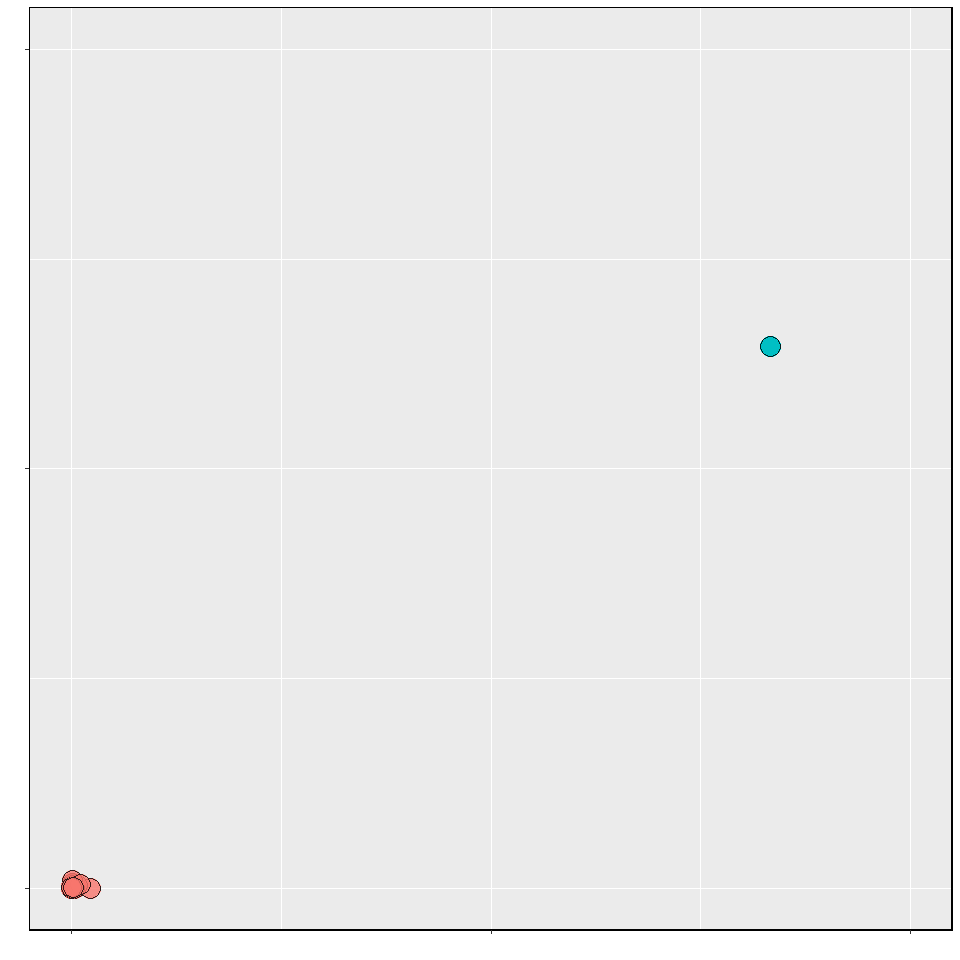
### Klekota-Roth Count

input <- readRDS("fpdata.Rds")  
Klekota\_Roth\_Count <- input$KlekotaRoth\_FingerprintCount  
plot\_scrambling(Klekota\_Roth\_Count,"Klekota\_CountS.csv")



### 2D Atom Pairs

input <- readRDS("fpdata.Rds")  
Atom\_Pairs <- input$AtomPairs2D\_fingerPrinter  
plot\_scrambling(Atom\_Pairs,"AtomS.csv")



### 2D Atom Pairs Count

input <- readRDS("fpdata.Rds")  
Atom\_Pairs\_Count <- input$AtomPairs2D\_fingerPrintCount  
plot\_scrambling(Atom\_Pairs\_Count,"Atom\_CountS.csv")

