

Supplementary Material

Shudong Sun

2022-09-13

Introduction

SSD can determine the sample size for our experiment when we only have a few pilot data. Firstly, it can generate different synthetic training datasets and test datasets. Then it will calculate the corresponding different test classification error/ARI/AMI. The final step is to construct the plots in the terms of the sample size and the different metrics (test classification error/ARI/AMI). Users can determine the sample size by the above plots.

Preparations

Before we dive into the main task, we need to load the package and an example dataset for our task. The dataset we use is the **pbmc_68k** dataset from 10x Genomics.

We pre-processed the dataset: In this dataset, *phenoid* is the y label which has 10 classes. We sampled 15 observations from the original dataset for each class and assemble them as the pilot data. we normalize and scale the pilot data at first and then run principal component analysis (PCA) and keep 18 PCs according to *JackStrawPlot* and *ElbowPlot* mentioned in Seurat - Guided Clustering Tutorial. The *JackStrawPlot* and *ElbowPlot* are shown below:

For the whole dataset, we pre-processed it using the same strategies and keep 23 PCs according to *JackStrawPlot* and *ElbowPlot*.

We put the pre-processed data into our package and we can load them directly.

```
library(SSD)

# load data
pilot_data <- read.csv(system.file("extdata", "data_pbmc68k_pilot_18pc.csv",
                                  package = "SSD"), row.names=1)

print(table(pilot_data$phenoid))
#>
#>          CD14+_Monocyte          CD19+_B
#>                15                15
#>    CD4+/CD25+_T_Reg  CD4+/CD45RA+/CD25+_Naive_T
#>                15                15
#>    CD4+/CD45RO+_Memory  CD4+_T_Helper2
#>                15                15
#>    CD56+_NK  CD8+/CD45RA+_Naive_Cytotoxic
#>                15                15
#>    CD8+_Cytotoxic_T  Dendritic
#>                15                15
```

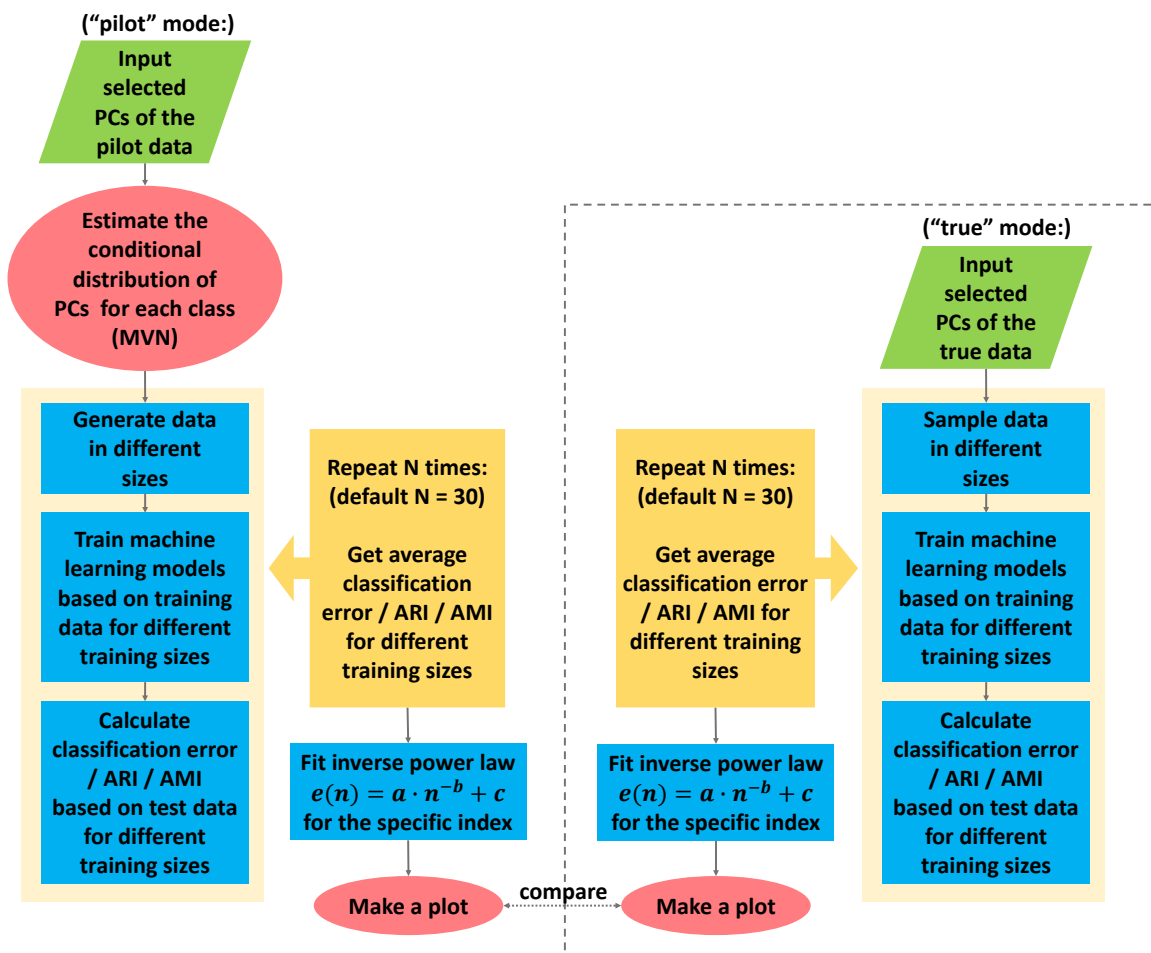


Figure 1: Workflow of the SSD package

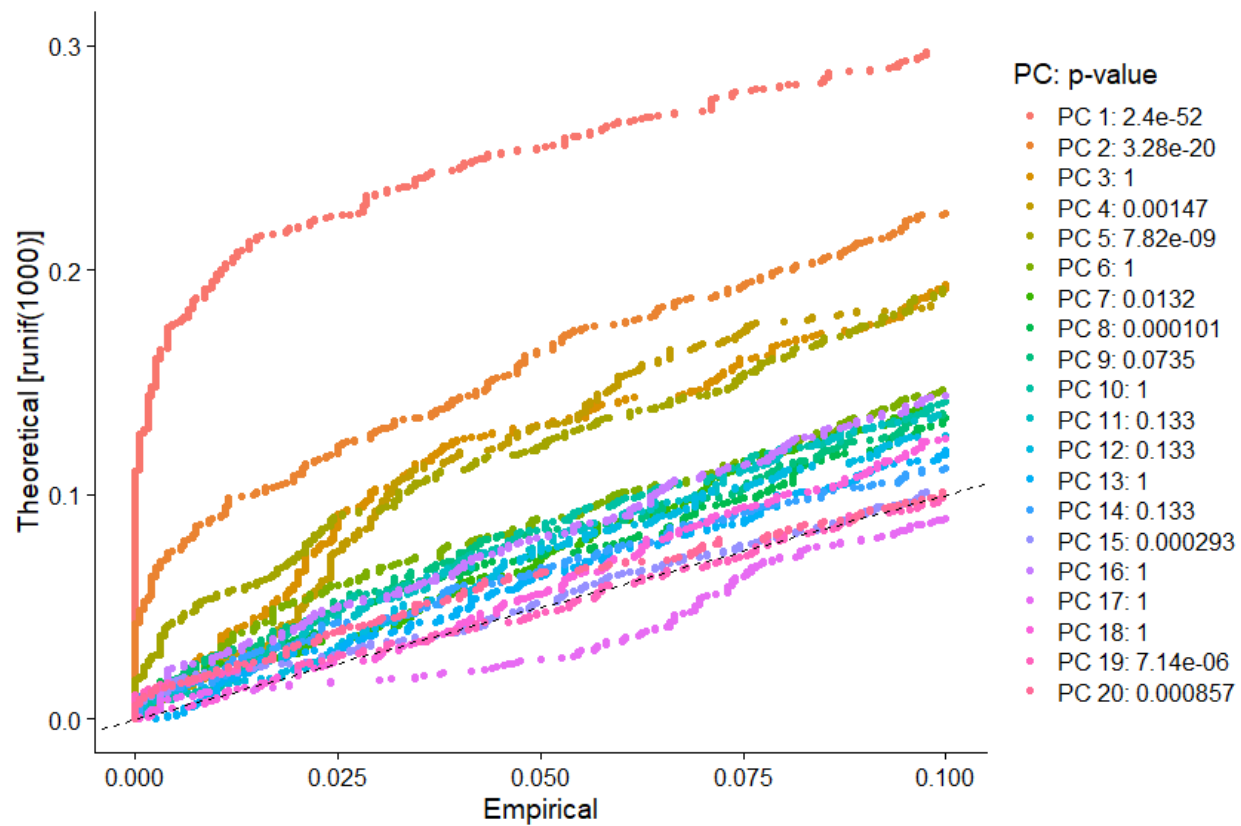


Figure 2: JackStrawPlot of the pilot data

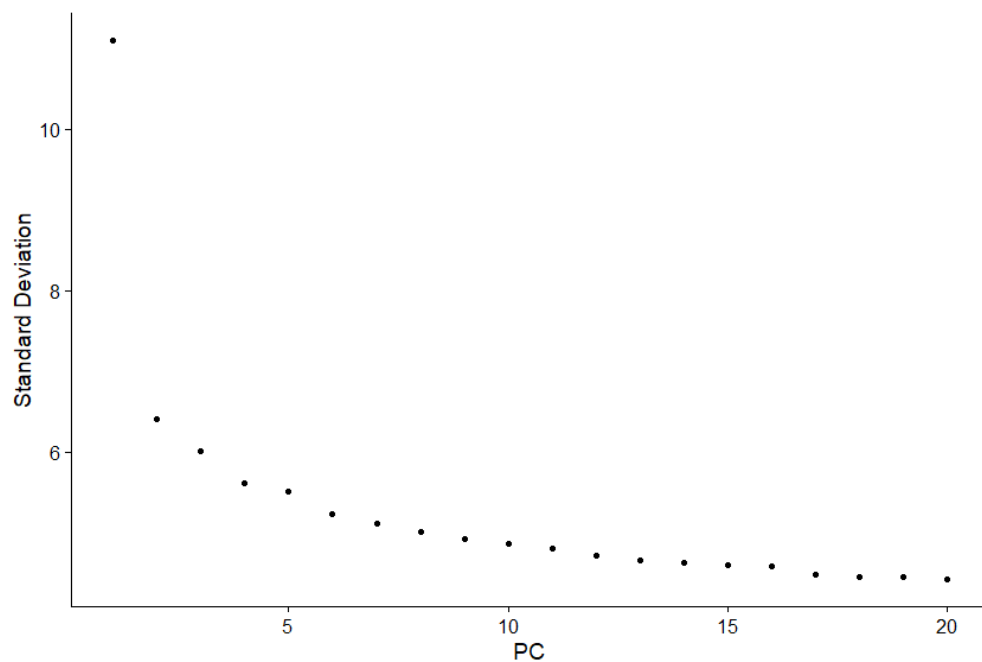


Figure 3: ElbowPlot of the pilot data

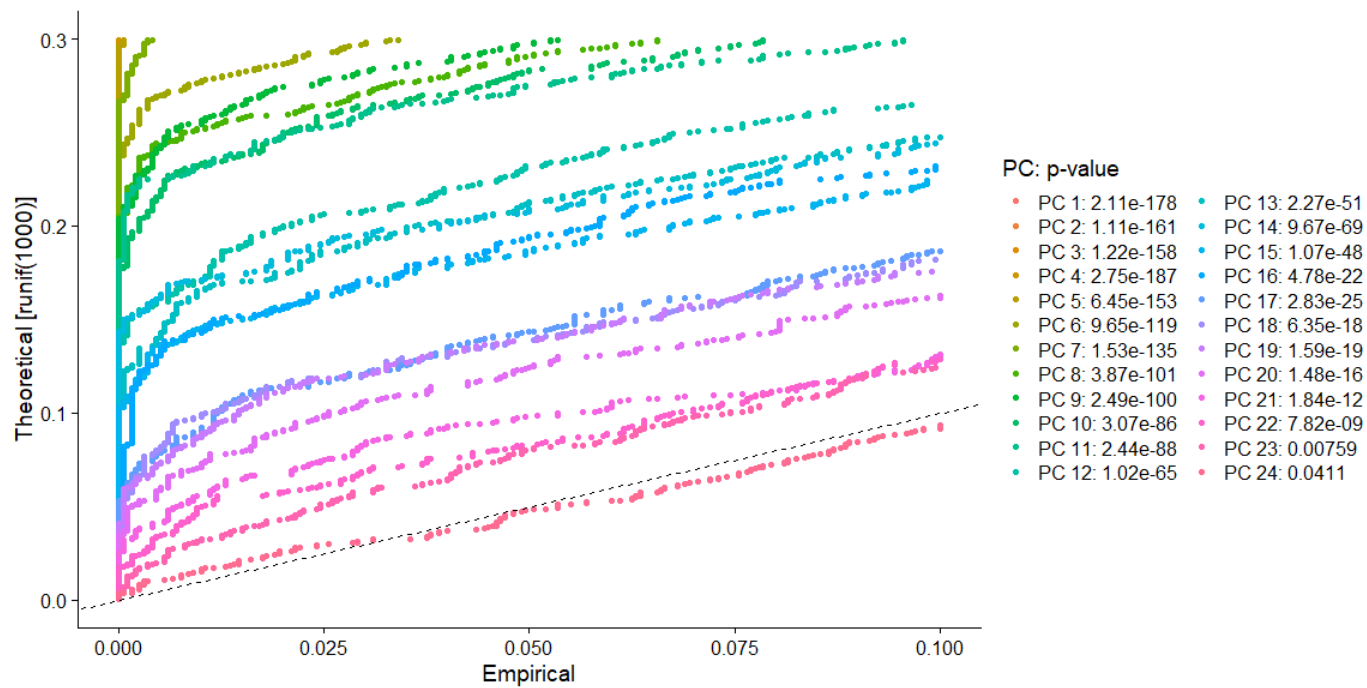


Figure 4: JackStrawPlot of the true training data

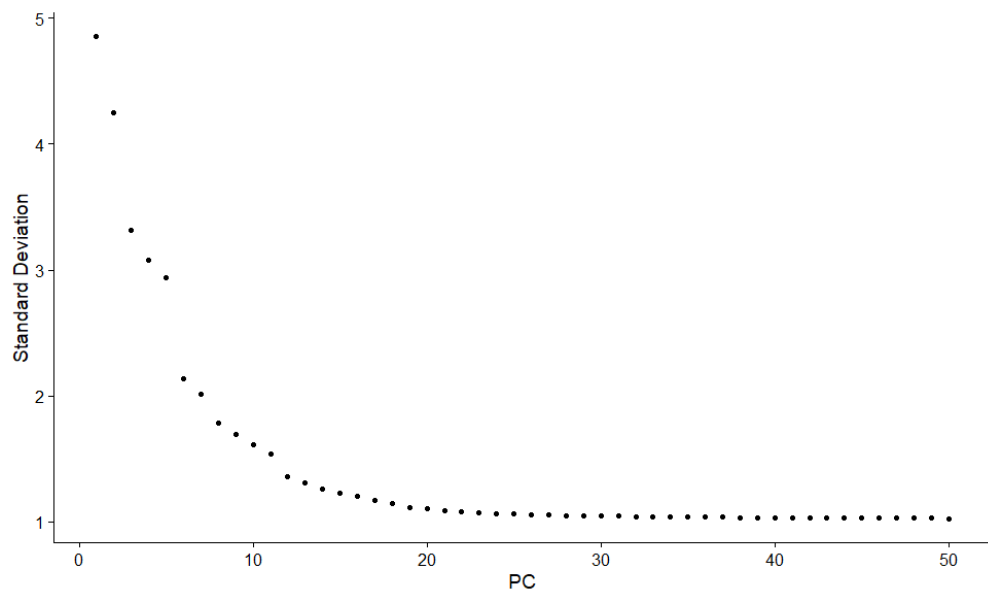


Figure 5: ElbowPlot of the true training data

```

train_data <- read.csv(system.file("extdata", "data_pbmc68k_train_23pc.csv",
                                package = "SSD"), row.names=1)
test_data <- read.csv(system.file("extdata", "data_pbmc68k_test_23pc.csv",
                                package = "SSD"), row.names=1)

print(table(train_data$phenoid))
#>
#>          CD14+_Monocyte          CD19+_B
#>          3717          3206
#>          CD4+/CD25_T_Reg  CD4+/CD45RA+/CD25-_Naive_T
#>          2712          3026
#>          CD4+/CD45RO+_Memory  CD4+_T_Helper2
#>          5759          11345
#>          CD56+_NK  CD8+/CD45RA+_Naive_Cytotoxic
#>          14012          21875
#>          CD8+_Cytotoxic_T  Dendritic
#>          1765          162
print(table(test_data$phenoid))
#>
#>          CD14+_Monocyte          CD19+_B
#>          100          100
#>          CD4+/CD25_T_Reg  CD4+/CD45RA+/CD25-_Naive_T
#>          100          100
#>          CD4+/CD45RO+_Memory  CD4+_T_Helper2
#>          100          100
#>          CD56+_NK  CD8+/CD45RA+_Naive_Cytotoxic
#>          100          100
#>          CD8+_Cytotoxic_T  Dendritic
#>          100          100

```

Task

With pilot data, draw the plot and determine sample size using the built-in model

In the default setting, we use the built-in *random forest* to train the model. The `index` we use is Adjusted Rand Index (ARI). By default, the size of training data for each class is (30, 60, 90, 120, ..., 540, 570, 600) and the size of test data for each class is 300.

```

x_pilot = pilot_data[,-length(pilot_data)]
y_pilot = pilot_data[,length(pilot_data)]

result_pilot = ssd(x_pilot, y_pilot)

```

The plot is drawn only based on the pilot data and we could use the plot to determine the sample size if we don't have large enough true. We should focus on the trends of the plots because the results produced by synthetic data are usually better than true data, but the trends are pretty similar.

With pilot data, draw the plot and determine sample size using the self-defined model

If you want to use the model defined by yourself. Then you need to write a "predict_model" function including your model. The function should take *train_data_x* and *train_data_y* as the first two inputs to train the model and then take *test_data_x* as the third input and return the predicted result of *test_data_x*. Then you could set `model` to *self* and set `func` to *predict_model*, and run the model using your self-defined function.

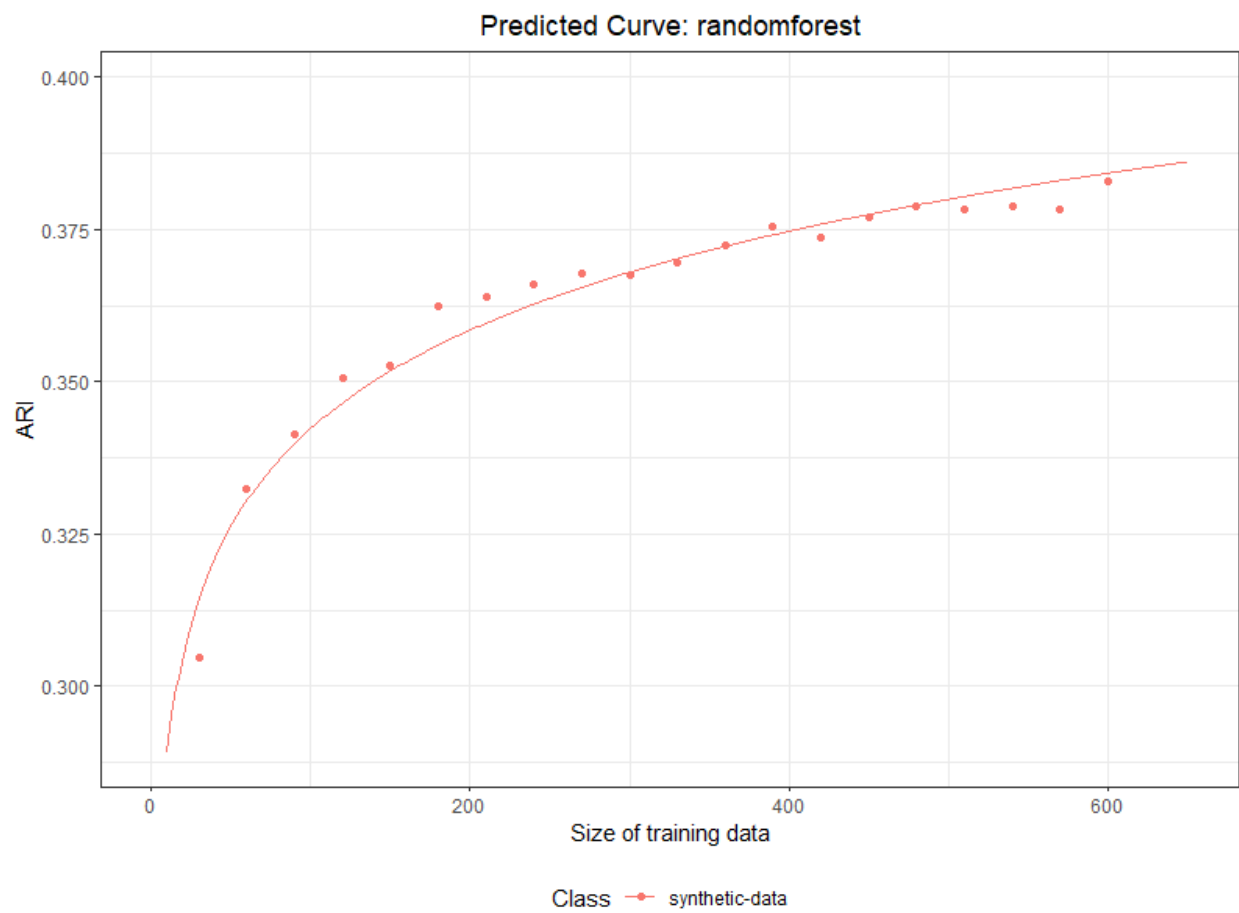


Figure 6: the result plot of the default setting

```
library(e1071)

predict_model <- function(train_data_x, train_data_y, test_data_x){
  train_data = data.frame(train_data_x, as.factor(train_data_y))
  names(train_data)[length(train_data)] = "class"
  fit_svm<-svm(class~.,data=train_data,probability=TRUE)
  pred <- predict(fit_svm, test_data_x)
  return(pred)
}

result_pilot_self = ssd(x_pilot, y_pilot, model="self", func=predict_model)
```

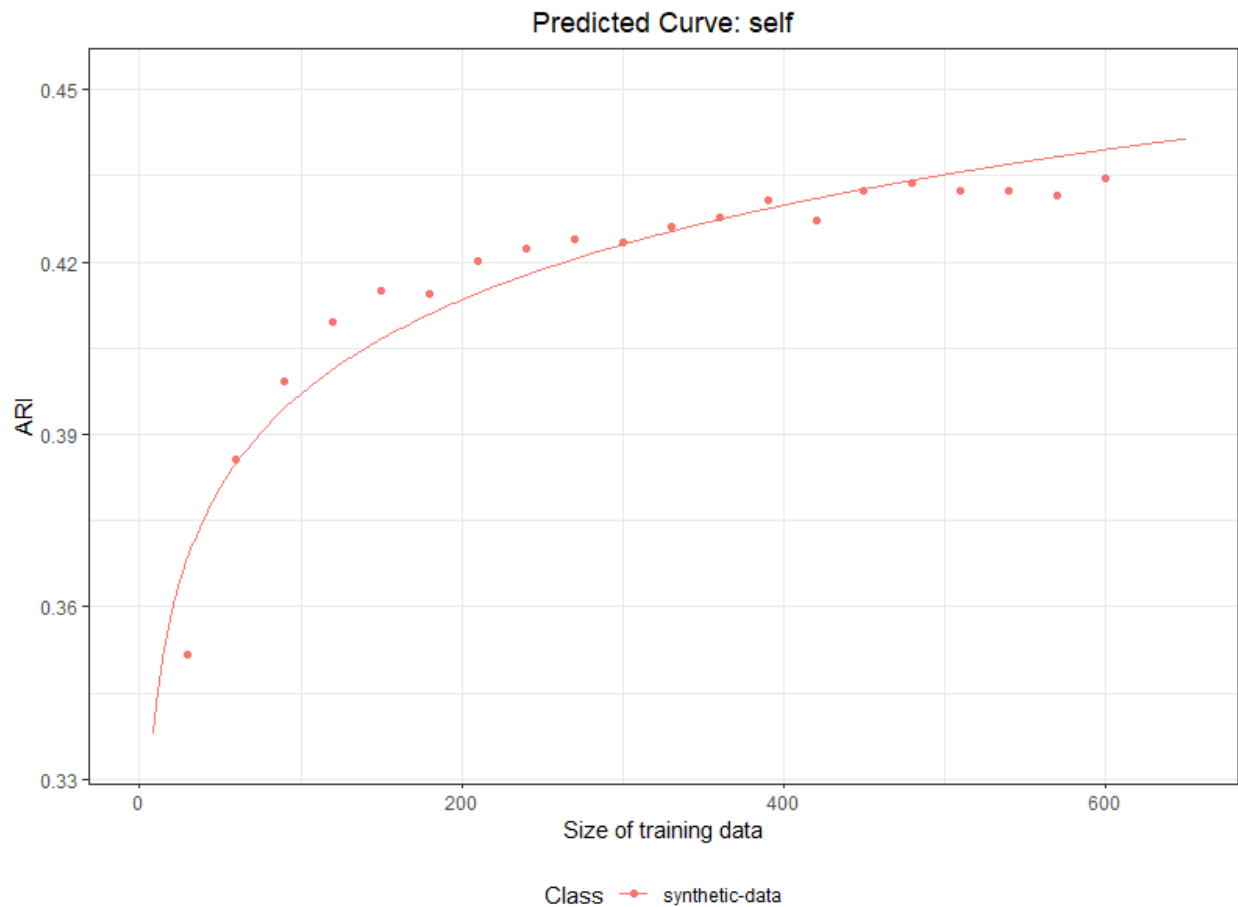


Figure 7: the result plot of the self-defined function setting

You could use the following code to check the function you defined. The result has to be the predicted value of *test_data_x*.

```
num_class=10
n_train=60
n_test=100

for(i in 1:num_class){
  class_i_ids = which(train_data$phenoid == names(table(train_data$phenoid))[i])
```

```

train_test_i_ids = sample(class_i_ids, (n_train+n_test))
train_i_data = train_data[train_test_i_ids[1:n_train],]
test_i_data = train_data[train_test_i_ids[(n_train+1):(n_train+n_test)],]

if(i == 1){
  train_data_sample = train_i_data
  test_data_sample = test_i_data
}else{
  train_data_sample = rbind(train_data_sample, train_i_data)
  test_data_sample = rbind(test_data_sample, test_i_data)
}
}

train_data_x = train_data_sample[,-length(train_data_sample)]
train_data_y = train_data_sample$phenoid
test_data_x = test_data_sample[,-length(test_data_sample)]

result = predict_model(train_data_x, train_data_y, test_data_x)

```

With pilot data and large true data, draw the plot and compare the result

If we have large enough true data and try to compare the plot drawn based on pilot data and the plot drawn based on true data, we could change mode to *true* and compare the results.

```

# prepare large true data
x_true_train = train_data[,-length(train_data)]
y_true_train = train_data[,length(train_data)]

table(train_data$phenoid)
x_true_test = test_data[,-length(test_data)]
y_true_test = test_data[,length(test_data)]
table(test_data$phenoid)

# run the model using the previous pilot data, index using 'ARI'
result_pilot_11 = ssd(x_pilot, y_pilot,model = "randomforest",index = "ARI",
  n_train_list=seq(from=30, to=600,by=30), n_test=100)
# run the model using the true data, index using 'ARI'
result_true_11 = ssd(x=x_true_train, y=y_true_train, model = "randomforest", index="ARI",
  n_train_list=seq(from=30, to=600,by=30),mode="true",
  test_x=x_true_test, test_y=y_true_test, n_test=100)

```

From the results above,even though the values of ARI are different, we can see that the two plots have almost the same trend, which could help us determine the sample size.

We could also try when index is *classification error* or *AMI* for this dataset.

```

# run the model using the previous pilot data, index using 'classification error'
result_pilot_12 = ssd(x_pilot, y_pilot,model="randomforest",index="classification error",
  n_train_list=seq(from=30, to=600,by=30), n_test=100)
# run the model using the true data, index using 'classification error'
result_true_12 = ssd(x=x_true_train, y=y_true_train, model = "randomforest",
  index="classification error", n_train_list=seq(from=30, to=600,by=30),
  mode="true", test_x=x_true_test, test_y=y_true_test, n_test=100)

```

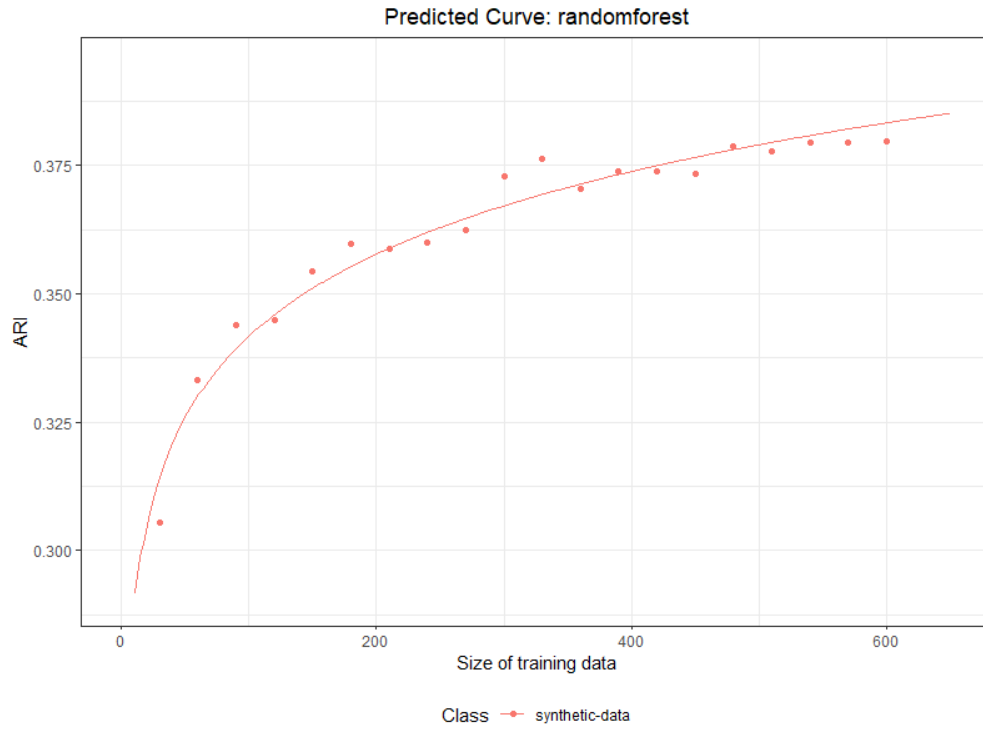



Figure 8: the result plots of comparison using 'ARI' as the index

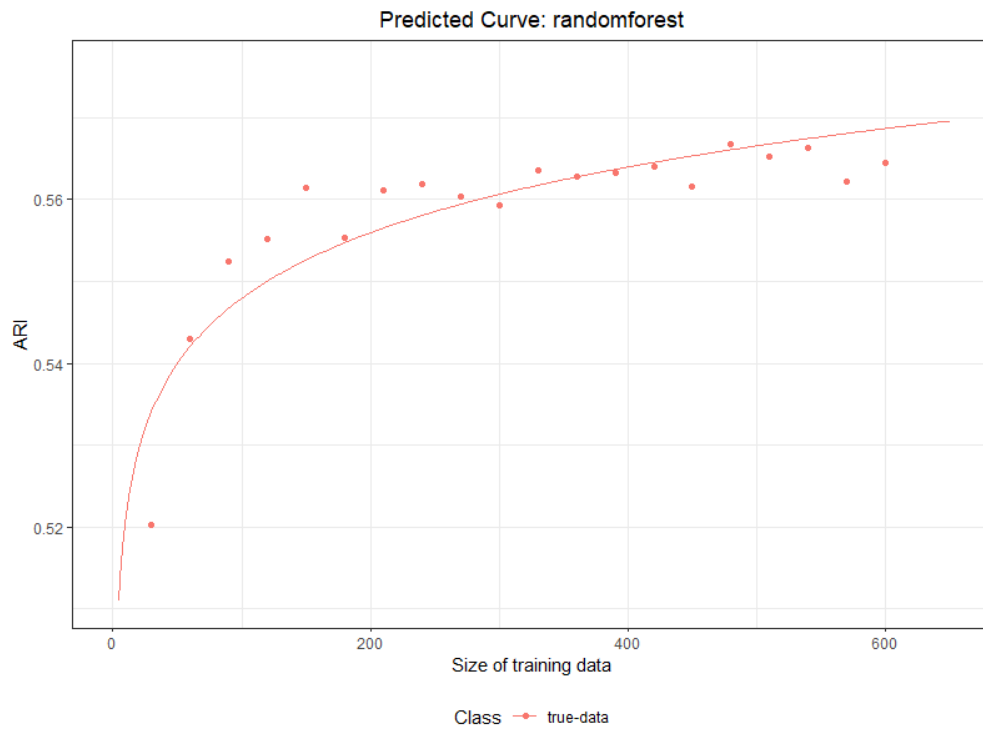


Figure 9: the result plots of comparison using 'ARI' as the index

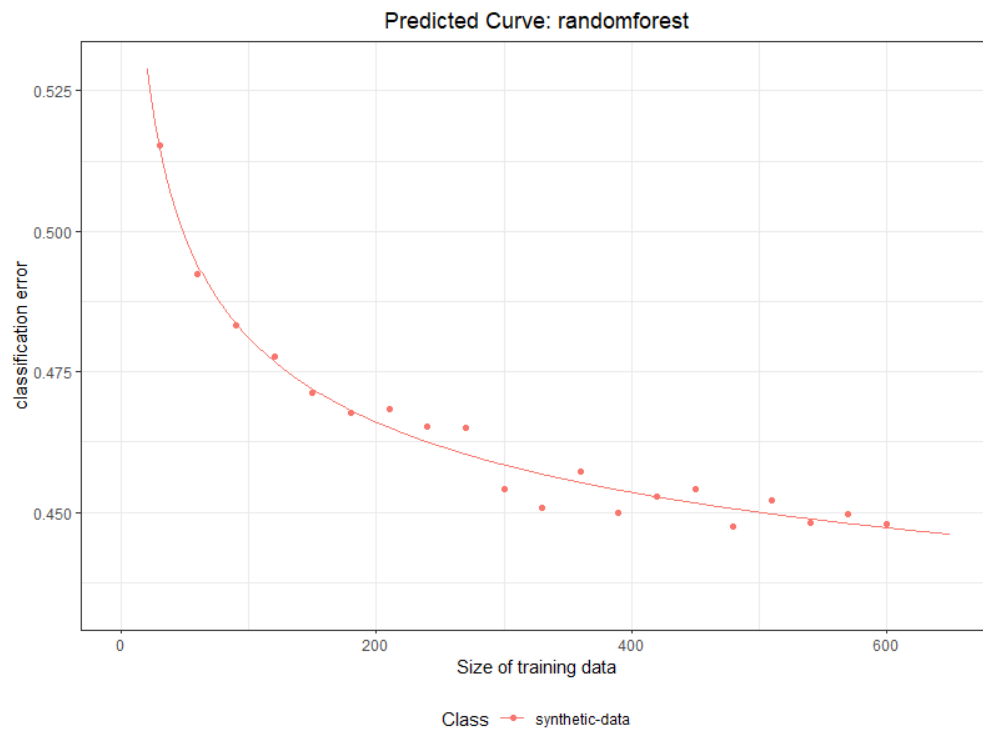


Figure 10: the result plots of comparison using 'classification error' as the index

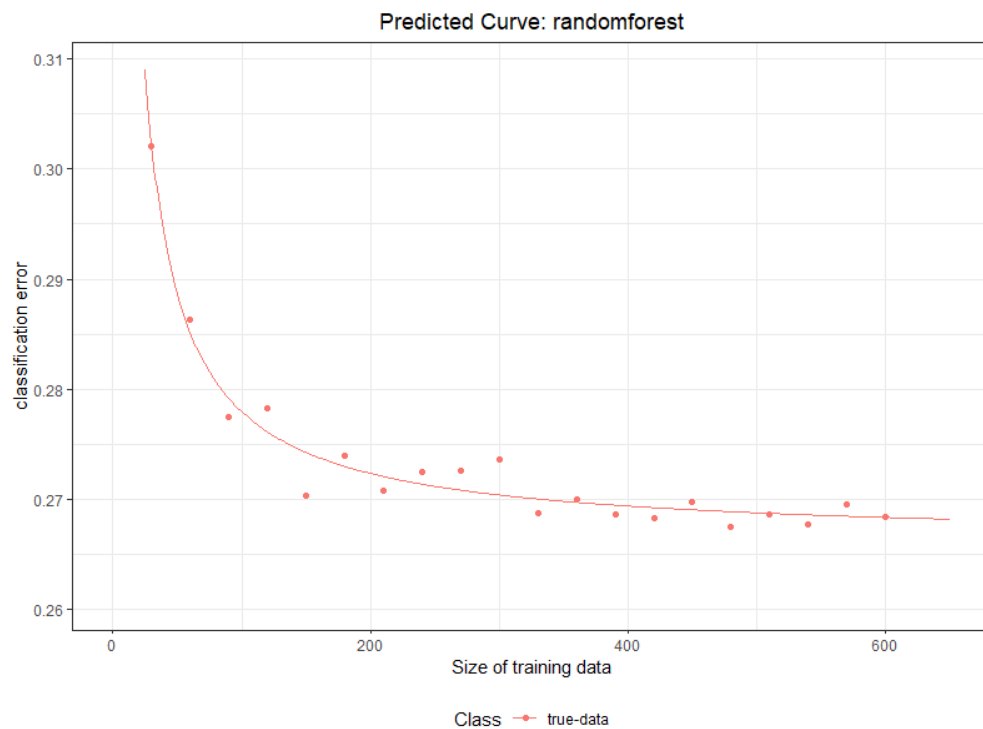


Figure 11: the result plots of comparison using 'classification error' as the index

```

# prepare large true data
x_true = data[,-length(data)]
y_true = data[,length(data)]

# run the model using the previous pilot data, index using 'AMI'
result_pilot_13 = ssd(x_pilot, y_pilot,model = "randomforest",index = "AMI",
                      n_train_list=seq(from=30, to=600,by=30), n_test=100)
# run the model using the true data, index using 'AMI'
result_true_13 = ssd(x=x_true_train, y=y_true_train, model = "randomforest", index="AMI",
                     n_train_list=seq(from=30, to=600,by=30),mode="true",
                     test_x=x_true_test,test_y=y_true_test, n_test=100)

```

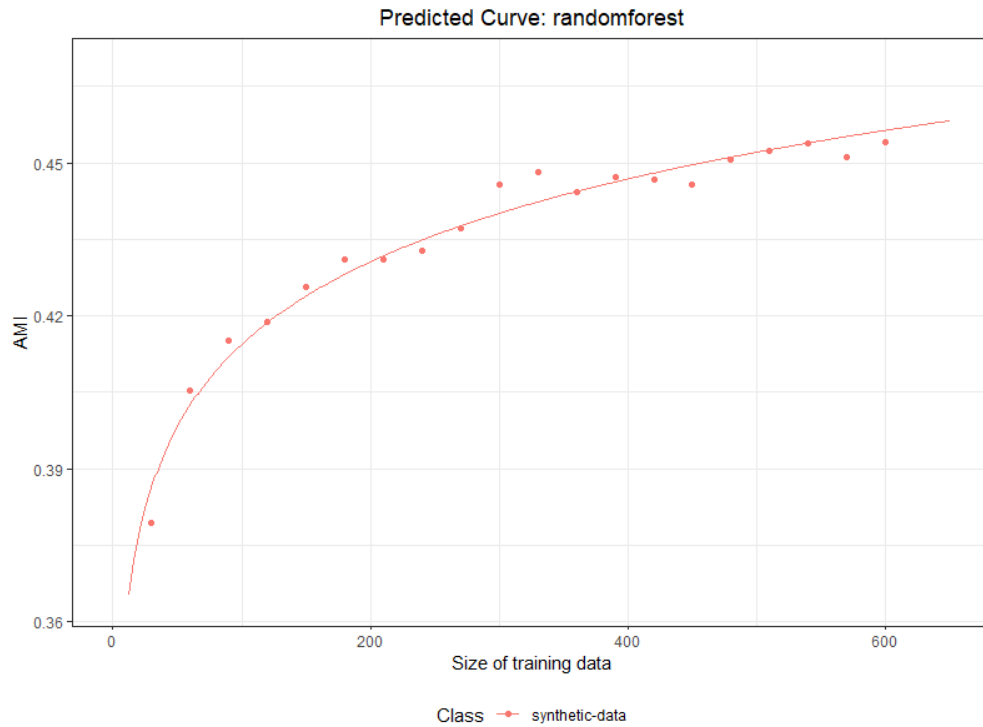


Figure 12: the result plots of comparison using 'AMI' as the index

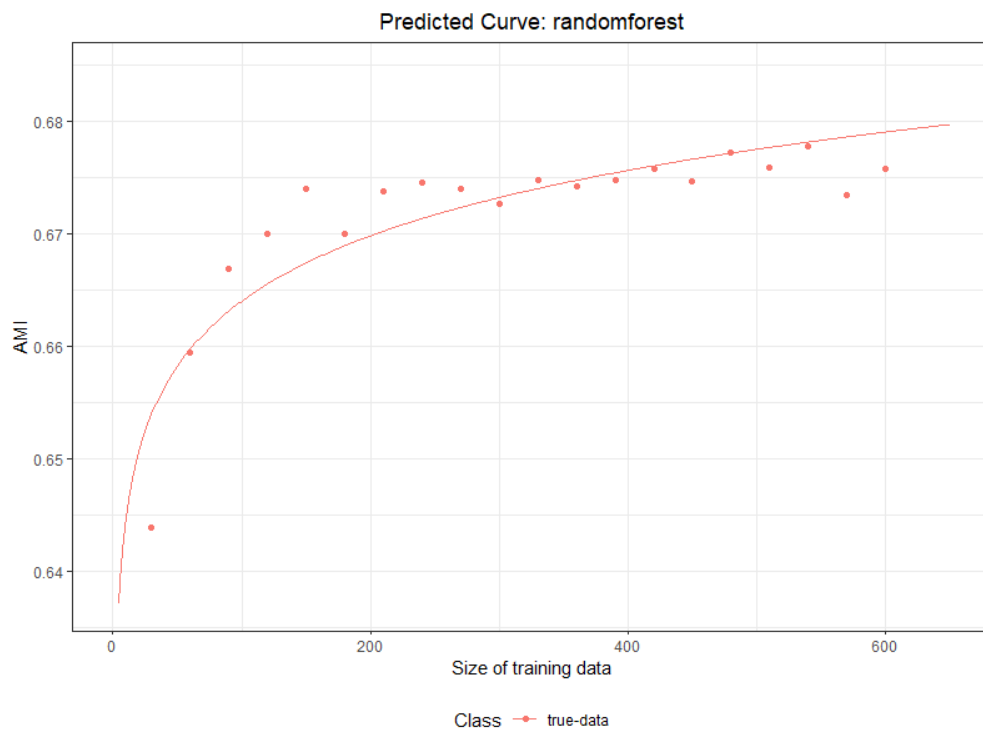


Figure 13: the result plots of comparison using 'AMI' as the index