Introduction to CytoDx

Zicheng Hu

2019-01-04

Contents

Introduction

CytoDx is a method that predicts clinical outcomes using single cell data without the need of cell gating. It first predicts the association between each cell and the outcome using a linear statistical model (Figure 1). The cell level predictions are then averaged within each sample to represent the sample level predictor. A second model is used to make prediction at the sample level (Figure 1). Compare to traditional gating based methods, CytoDX have multiple advantages.

1. Robustness. CytoDx is able to robustly predict clinical outcomes using data acquired by different cytometry platforms in different research institutes.
2. Interpretability. CytoDx identifies cell markers and cell subsets that are most associated with the clinical outcome, allowing researchers to interpret the result easily.
3. Simplicity. CytoDx associates cytometry data with clinical features without any cell gating steps, therefore simplifying the diagnostic process in clinical settings.

In section 2, we demonstrate how to install CytoDx.

In section 3, we show an example where CytoDx is used to diagnose acute myeloid leukemia (AML). In addition to perform diagnosis, we also show that CytoDx can be used to identify the cell subsets that are associated with AML.

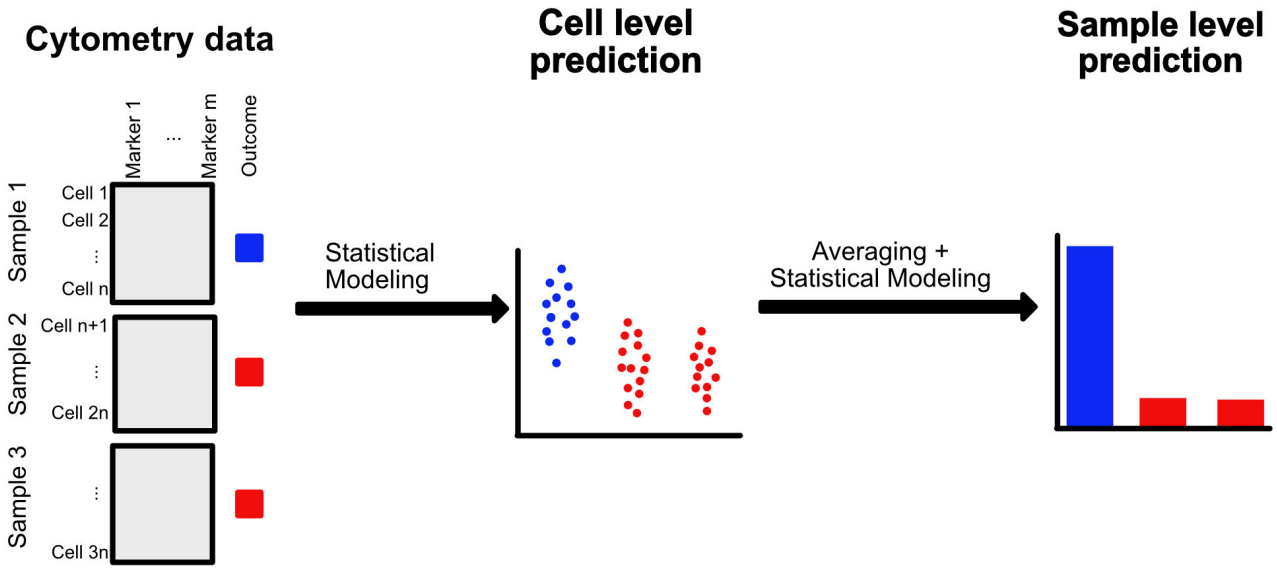


Figure 1

Installation

You can install the stable version of CytoDx from Bioconductor:

```
BiocManager::install("CytoDx")
```

You can also install the latest develop version of CytoDx from github:

```
devtools::install_github("hzc363/CytoDx")
```

Example: diagnosing AML using flow cytometry

In this example, we build a CytoDx model to diagnose acute myeloid leukemia (AML) using flow cytometry data. We train the model using data from 5 AML patients and 5 controls and test the performance in a test dataset.

Step 1: Prepare data

The CytoDx R package contains the fcs files and the ground truth (AML or normal) that are needed for our example. We first load the ground truth.

```
library(CytoDx)

# Find data in CytoDx package
path <- system.file("extdata",package="CytoDx")

# read the ground truth
fcs_info  <- read.csv(file.path(path,"fcs_info.csv"))

# print out the ground truth
knitr::kable(fcs_info)
```

fcsName	Label	dataset
sample1.fcs	aml	test
sample2.fcs	aml	test
sample3.fcs	aml	test
sample4.fcs	aml	test
sample5.fcs	aml	test
sample6.fcs	normal	test
sample7.fcs	normal	test
sample8.fcs	normal	test
sample9.fcs	normal	test
sample10.fcs	normal	test
sample11.fcs	aml	train
sample12.fcs	aml	train
sample13.fcs	aml	train
sample14.fcs	aml	train
sample15.fcs	aml	train
sample16.fcs	normal	train
sample17.fcs	normal	train
sample18.fcs	normal	train
sample19.fcs	normal	train
sample20.fcs	normal	train

We then read the cytometry data for training samples using the fcs2DF function.

```
# Find the training data
train_info <- subset(fcs_info,fcs_info$dataset=="train")

# Specify the path to the cytometry files
fn <- file.path(path,train_info$fcsName)

# Read cytometry files using fcs2DF function
```

```
train_data <- fcs2DF(fcsFiles=fn,
                     y=train_info$Label,
                     assay="FCM",
                     b=1/150,
                     excludeTransformParameters=
                       c("FSC-A","FSC-W","FSC-H","Time"))
```

The CytoDx is flexible to data transformations. It can be applied to rank transformed data to reduce batch effects. Here, we transform the original data to rank data.

```
# Perfroms rank transformation
x_train <- pRank(x=train_data[,1:7],xSample=train_data$xSample)

# Convert data frame into matrix. Here we included the 2-way interactions.
x_train <- model.matrix(~.*.,x_train)
```

Step 2: Build CytoDx model

We use training data to build a predictive model.

```
# Build predictive model using the CytoDx.fit function
fit <- CytoDx.fit(x=x_train,
                  y=(train_data$y=="aml"),
                  xSample=train_data$xSample,
                  family = "binomial",
                  reg = FALSE)
```

Step 3: Predict AML using testing data

We first load and rank transform the test data.

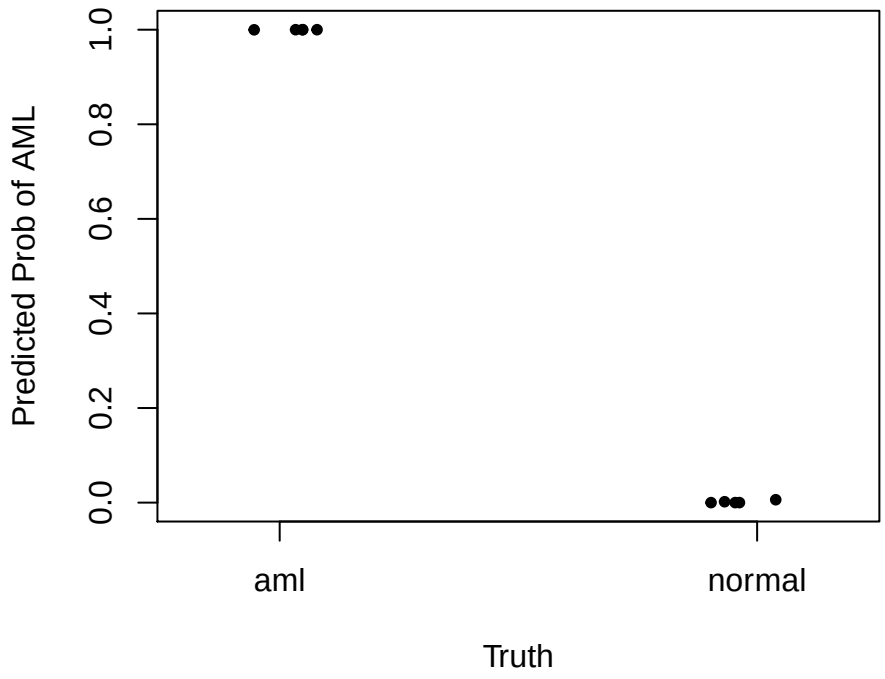
```
# Find testing data
test_info <- subset(fcs_info,fcs_info$dataset=="test")

# Specify the path to cytometry files
fn <- file.path(path,test_info$fcsName)

# Read cytometry files using fcs2DF function
test_data <- fcs2DF(fcsFiles=fn,
                    y=NULL,
                    assay="FCM",
                    b=1/150,
                    excludeTransformParameters=
                      c("FSC-A","FSC-W","FSC-H","Time"))
# Perfroms rank transformation
x_test <- pRank(x=test_data[,1:7],xSample=test_data$xSample)

# Convert data frame into matrix. Here we included the 2-way interactions.
x_test <- model.matrix(~.*.,x_test)
```

We use the built CytoDx model to predict AML.

```
# Predict AML using CytoDx.ped function
pred <- CytoDx.pred(fit,xNew=x_test,xSampleNew=test_data$xSample)
```

We plot the prediction. In this example, CytoDx classifies the sample into AML and normal perfectly.

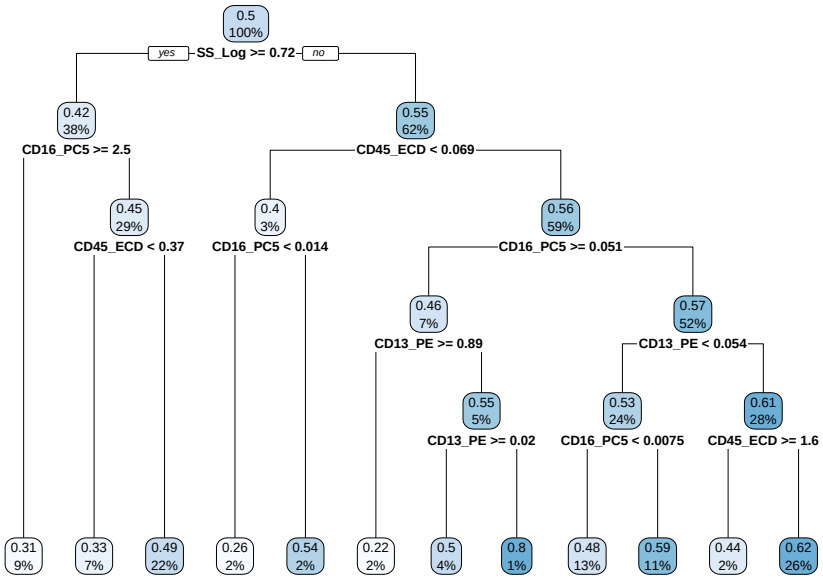
```
# Cmbine prediction and truth
result <- data.frame("Truth"=test_info$Label,
                     "Prob"=pred$xNew.Pred.sample$y.Pred.s0)

# Plot the prediction
stripchart(result$Prob~result$Truth, jitter = 0.1,
           vertical = TRUE, method = "jitter", pch = 20,
           xlab="Truth",ylab="Predicted Prob of AML")
```



Step 4: Find cell subsets associated with AML

We use a decision tree to find cell subsets that are associated the AML. In this step, the original cytometry data should be used, rather than the ranked data.

```
# Use decision tree to find the cell subsets that are associated the AML.
TG <- treeGate(P = fit$train.Data.cell$y.Pred.s0,
               x= train_data[,1:7])
```



Session Infomation

```
sessionInfo()
```

R version 3.5.2 Patched (2018-12-20 r75875) Platform: x86_64-apple-darwin15.6.0 (64-bit) Running under: OS X El Capitan 10.11.6

Matrix products: default BLAS: /Library/Frameworks/R.framework/Versions/3.5/Resources/lib/libRblas.0.dylib LAPACK: /Library/Frameworks/R.framework/Versions/3.5/Resources/lib/libRlapack.dylib

locale: [1] C/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8

attached base packages: [1] stats graphics grDevices utils datasets methods base

other attached packages: [1] bindrcpp_0.2.2 CytoDx_1.2.1

loaded via a namespace (and not attached): [1] pcaPP_1.9-73 Rcpp_1.0.0 highr_0.7
[4] DEoptimR_1.0-8 pillar_1.3.1 compiler_3.5.2
[7] bindr_0.1.1 iterators_1.0.10 tools_3.5.2
[10] rpart_4.1-13 digest_0.6.18 evaluate_0.12
[13] tibble_2.0.0 lattice_0.20-38 pkgconfig_2.0.2
[16] rlang_0.3.0.1 Matrix_1.2-15 foreach_1.4.4
[19] graph_1.60.0 yaml_2.2.0 parallel_3.5.2
[22] mvtnorm_1.0-8 xfun_0.4 dplyr_0.7.8
[25] stringr_1.3.1 knitr_1.21 cluster_2.0.7-1
[28] glmnet_2.0-16 stats4_3.5.2 grid_3.5.2
[31] tidyselect_0.2.5 robustbase_0.93-3 glue_1.3.0
[34] Biobase_2.42.0 R6_2.3.0 rrcov_1.4-7
[37] rmarkdown_1.11 corpcor_1.6.9 purrr_0.2.5
[40] magrittr_1.5 MASS_7.3-51.1 codetools_0.2-16
[43] htmltools_0.3.6 matrixStats_0.54.0 BiocGenerics_0.28.0 [46] rpart.plot_3.0.6 assertthat_0.2.0 flowCore_1.48.1
[49] stringi_1.2.4 doParallel_1.0.14 crayon_1.3.4