RchyOptimyx: Gating Hierarchy Optimization for Flow Cytometry

Nima Aghaeepour and Adrin Jalali

May 3, 2016

naghaeep@gmail.com

Contents

1 Licensing

Under the Artistic License, you are free to use and redistribute this software.

2 Introduction

This document demonstrates the functionality of the RchyOptimyx package, a tool for cellular hieraRCHY OPTIMization for flow cytometry data (named after Archeopteryx).

RchyOptimyx models all possible gating strategies and marker panels that can be generated using a high-color assay, and uses dynamic programing and optimization tools from graph-theory to determine the minimal sets of markers that can identify a target population to a desired level of purity. A cellular hierarchy is a directed acyclic graph (DAG), embedded in a plane as a top-down diagram, with one node on the top most level representing all cells (or a major component therefore, such as T-cells) and nodes further down showing more specific cell populations. All the intermediate cell populations are placed in the hierarchy using parent-child relationships. The graph starts from level 0 to level m including i-marker phenotypes on i^{th} level. The phenotype with 0 markers is the top most phenotype with all cells and the phenotype with m markers is the cell population of interest.

The required input phenotypes and their respective scores (target values of the optimization) can be generated either using manual gates or automated gating algorithms (see the *flowType* package in Bioconductor for examples).

The flowType and RchyOptimyx algorithms are described elsewhere [**?**, **?**] and examples are available through [**?**, **?**, **?**].

3 Example: Preparing Raw Data for RchyOptimyx

In this example, we start from a raw flowSet and generated the required materials to produce an RchyOptimyx graph. The dataset consists of a flowSet *HIVData* with 18 HIV$^+$ and 13 normals and a matrix *HIVMetaData* which consists of FCS filename, tube number, and patient label. In this example, we are interested in the second tube only. For more details please see the *flowType* package.

```
> library(flowType)
> data(HIVData)
> data(HIVMetaData)
> HIVMetaData <- HIVMetaData[which(HIVMetaData[,'Tube']==2),];
```

We convert the subject labels so that HIV$^+$ and normal subjects are labeled 2 and 1, respectively.

```
> Labels=(HIVMetaData[,2]=='+')+1;
```

3.1 Processing using flowType

We start by calculating the cell proportions using flowType:

```
> library(flowCore)
> library(RchyOptimyx)
> ##Markers for which cell proportions will be measured.
> PropMarkers <- 5:10
> ##Markers for which MFIs will be measured.
> MFIMarkers <- PropMarkers
> ##Marker Names
> MarkerNames <- c('Time', 'FSC-A','FSC-H','SSC-A',
+                  'IgG','CD38','CD19','CD3',
+                  'CD27','CD20', 'NA', 'NA')
> ##Apply flowType
> ResList <- fsApply(HIVData, 'flowType', PropMarkers,
+                    MFIMarkers, 'kmeans', MarkerNames);
> ##Extract phenotype names
> phenotype.names=unlist(lapply(ResList[[1]]@PhenoCodes,function(x){return(decodePhenotype(x
> names(ResList[[1]]@PhenoCodes)=phenotype.names
```

Then we extract all cell proportions from the list of flowType results and normalize them by the total number of cells in each sample (which is stored in the first cell population - the one with no markers included) to create the all.proportions matrix.

```
> all.proportions <- matrix(0,length(ResList[[1]]@CellFreqs),length(HIVMetaData[,1]))
> for (i in 1:length(ResList))
+   all.proportions[,i] = ResList[[i]]@CellFreqs / ResList[[i]]@CellFreqs[1]
```

We use a t-test to select the phenotypes that have a significantly different mean across the two groups of patients (FDR=0.05). Note that in real world use-cases the assumptions of a t-test must be checked or a resampling-based alternative (e.g., a permutation test) should be used. Sensitivity analysis (e.g., bootstrapping) is also necessary. Eight phenotypes are selected as statistically significant.

```
> Pvals <- vector();
> EffectSize <- vector();
> for (i in 1:dim(all.proportions)[1]){
+
+   ##If all of the cell proportions are 1 (i.e., the phenotype
+   ##with no gates) the p-value is 1.
+   if (length(which(all.proportions[i,]!=1))==0){
+     Pvals[i]=1;
+     EffectSize[i]=0;
+     next;
+   }
+   temp=t.test(all.proportions[i, Labels==1],
+     all.proportions[i, Labels==2])
+   Pvals[i] <- temp$p.value
+   EffectSize[i] <- abs(temp$statistic)
+ }
> Pvals[is.nan(Pvals)]=1
> names(Pvals)=phenotype.names
> ##Bonferroni's correction
> ## TODO: no good results shows up and there is no adjusted p-value <0.6
> selected <- which(p.adjust(Pvals)<0.65);
> print(names(selected))
```

```
 [1] "CD27-"                      "CD27+"
 [3] "CD38-CD27-"                 "CD3+CD27-"
 [5] "IgG-CD27+"                  "CD38-CD27+"
 [7] "CD19-CD27+"                 "CD3-CD27+"
 [9] "CD27+CD20-"                 "CD38-CD3+CD27-"
[11] "CD38+CD3+CD27-"             "CD19-CD3+CD27-"
[13] "IgG-CD38-CD27+"             "IgG-CD19-CD27+"
[15] "CD38-CD19-CD27+"            "CD38-CD3-CD27+"
[17] "CD19-CD3-CD27+"             "IgG-CD3+CD27+"
[19] "CD3+CD27-CD20-"             "IgG-CD27+CD20-"
[21] "CD38-CD27+CD20-"            "CD19-CD27+CD20-"
[23] "CD3-CD27+CD20-"             "CD38-CD19-CD3+CD27-"
[25] "CD38+CD19-CD3+CD27-"        "IgG-CD38-CD19-CD27+"
```

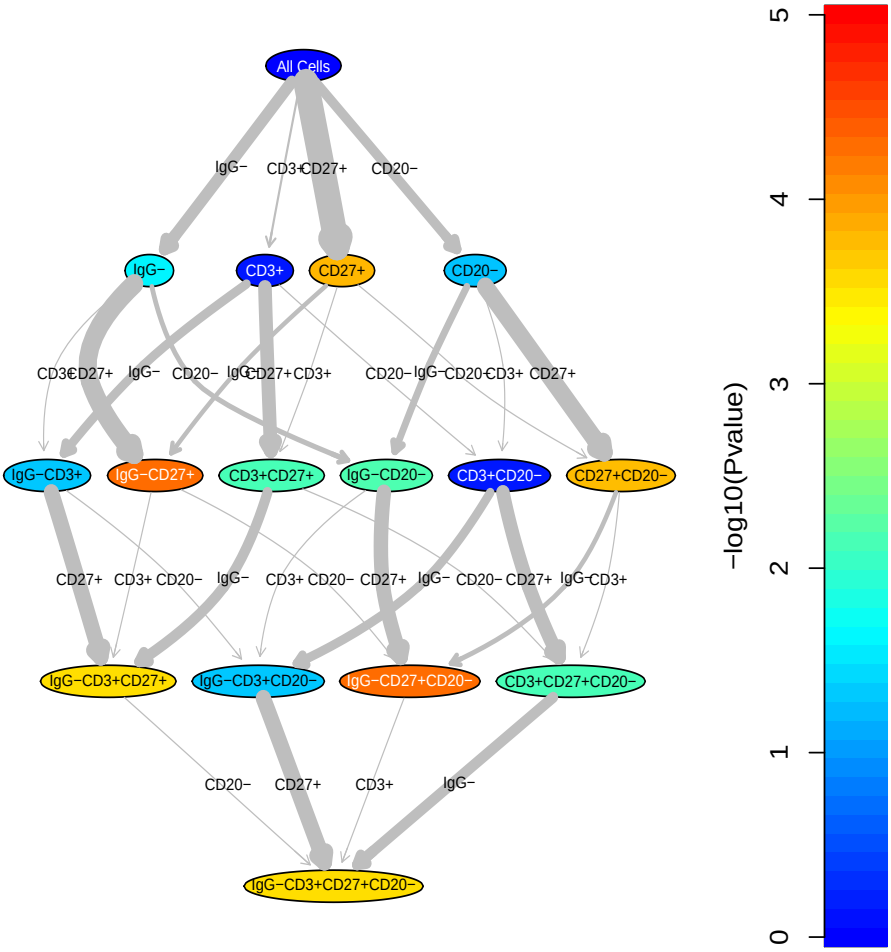
```
[27] "CD38-CD19-CD3-CD27+"          "IgG-CD38-CD3+CD27+"
[29] "IgG-CD19-CD3+CD27+"           "CD38-CD3+CD27-CD20-"
[31] "CD38+CD3+CD27-CD20-"          "CD19-CD3+CD27-CD20-"
[33] "IgG-CD38-CD27+CD20-"          "IgG-CD19-CD27+CD20-"
[35] "CD38-CD19-CD27+CD20-"         "CD38-CD3-CD27+CD20-"
[37] "CD19-CD3-CD27+CD20-"          "IgG-CD3+CD27+CD20-"
[39] "IgG-CD38-CD19-CD3+CD27+"      "CD38-CD19-CD3+CD27-CD20-"
[41] "CD38+CD19-CD3+CD27-CD20-"     "IgG-CD38-CD19-CD27+CD20-"
[43] "CD38-CD19-CD3-CD27+CD20-"     "IgG-CD38-CD3+CD27+CD20-"
[45] "IgG-CD19-CD3+CD27+CD20-"      "IgG-CD38-CD19-CD3+CD27+CD20-"
```

3.2 Basic RchyOptimyx Functionality

We select the longest one (*IgG-CD38-CD19-CD27+CD20-*) for further analysis using RchyOptimyx.

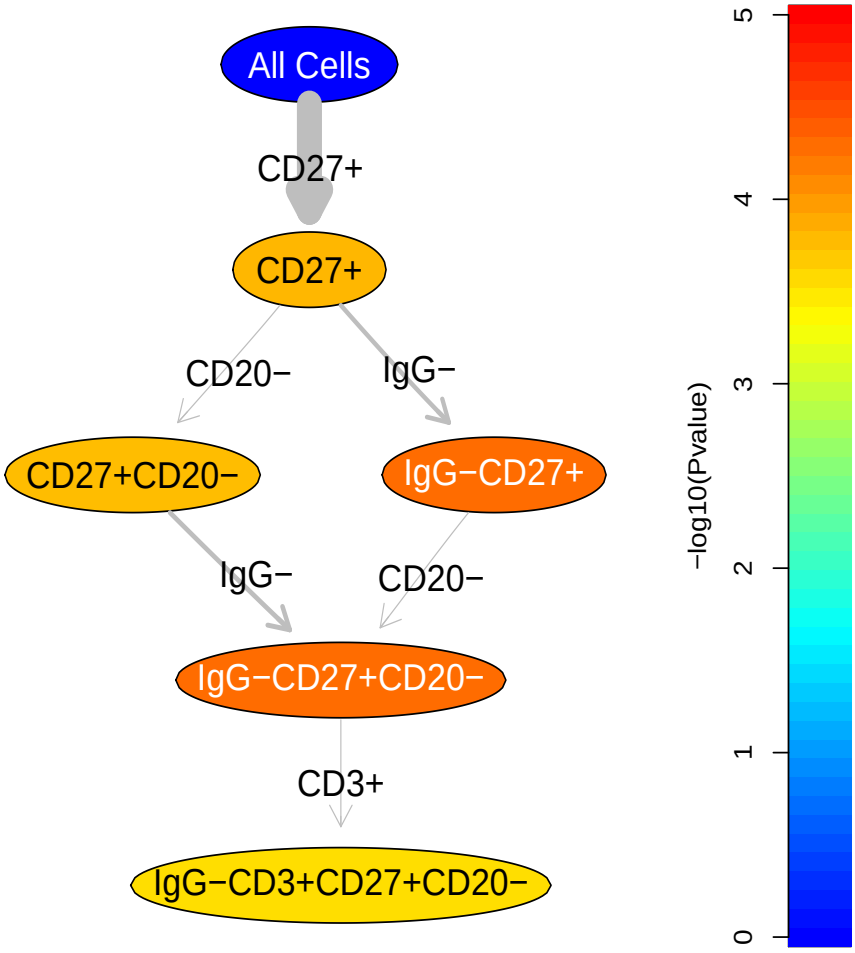
Now we can plot the hierarchy:

```
> res<-RchyOptimyx(ResList[[1]]@PhenoCodes, -log10(Pvals),
+                  ResList[[1]]@PhenoCodes[selected[38]], factorial(6),FALSE)
> plot(res, phenotypeScores=-log10(Pvals), phenotypeCodes=ResList[[1]]@PhenoCodes, marker.na
```



and an optimized hierarchy (with only the top 10 paths):

```
> res<-RchyOptimyx(pheno.codes=ResList[[1]]@PhenoCodes, phenotypeScores=-log10(Pvals),
+                  startPhenotype=ResList[[1]]@PhenoCodes[selected[38]], 2, FALSE)
> plot(res, phenotypeScores=-log10(Pvals), phenotypeCodes=ResList[[1]]@PhenoCodes, marker.na
```



3.3 RchyOptimyx with Multiple Targets

An optmized hierarchy can contain more than one target. For example, we can create a hierarchy for all of the selected cell populations. Because the plot will be too large, we will trim it to 4 levels.

```
> res<-RchyOptimyx(pheno.codes=ResList[[1]]@PhenoCodes, phenotypeScores=-log10(Pvals),
+           startPhenotype=ResList[[1]]@PhenoCodes[selected[1]], 1, FALSE,trim.level=4)
> for (i in 2:length(selected)){
+ temp<-RchyOptimyx(pheno.codes=ResList[[1]]@PhenoCodes, phenotypeScores=-log10(Pvals),
+     startPhenotype=ResList[[1]]@PhenoCodes[selected[i]], 1, FALSE,trim.level=4)
+ res=merge(res,temp)
+ }
```

```
> plot(res, phenotypeScores=-log10(Pvals), phenotypeCodes=ResList[[1]]@PhenoCodes, marker.na
```