PBMCs Example

```
library(org.Hs.eg.db)
library(Seurat)
source('cell_type_identification4_clean.R')
```

We start by reading in our training data, which is downloaded from 10X Genomics: CD4, CD8, CD14, and NK cells. For the purposes of this example, we will withhold 100 cells of each to serve as test data. After loading in the data, we convert the gene symbols to Ensembl IDs (required).

```
set.seed(6619)

### Preparation for symbol->ENSEMBL conversion
Hs_symbol <- org.Hs.egSYMBOL
mapped_Hs_genes.symbol <- mappedkeys(Hs_symbol)
Hs_symbol.df <- as.data.frame(Hs_symbol[mapped_Hs_genes.symbol])
Hs_ensembl <- org.Hs.egENSEMBL
mapped_Hs_genes.ensembl <- mappedkeys(Hs_ensembl)
Hs_ensembl.df <- as.data.frame(Hs_ensembl[mapped_Hs_genes.ensembl])
Hs_mapping <- merge(Hs_symbol.df,Hs_ensembl.df)

## CD4: https://support.10xgenomics.com/single-cell-gene-expression/datasets/1.1.0/cd4_t_helper?
cd4_facs.data <- Read10X(data.dir = "../cd4_singlecell/")
cd4_facs.data <- as.matrix(cd4_facs.data)
rownames(cd4_facs.data) <- Hs_mapping$ensembl_id[match(rownames(cd4_facs.data),
                                                       Hs_mapping$symbol)]
cd4_facs.data <- cd4_facs.data[!is.na(rownames(cd4_facs.data)),]
cd4_facs.data <- na.omit(cd4_facs.data)
cd4.test <- cd4_facs.data[,1:100] # CD4 withheld cells
cd4_facs.data <- cd4_facs.data[,101:5101] # CD4 training cells

## CD8: https://support.10xgenomics.com/single-cell-gene-expression/datasets/1.1.0/cytotoxic_t
cd8_facs.data <- Read10X(data.dir = '../filtered_matrices_cd8/hg19/')
cd8_facs.data <- as.matrix(cd8_facs.data)
rownames(cd8_facs.data) <- Hs_mapping$ensembl_id[match(rownames(cd8_facs.data),
                                                       Hs_mapping$symbol)]
cd8_facs.data <- cd8_facs.data[!is.na(rownames(cd8_facs.data)),]
cd8.test <- cd8_facs.data[,1:100] # CD8 withheld cells
cd8_facs.data <- cd8_facs.data[,101:5101] # CD8 training cells

## CD14: https://support.10xgenomics.com/single-cell-gene-expression/datasets/1.1.0/cd14_monocytes
cd14_facs.data <- Read10X(data.dir = '../filtered_matrices_cd14/hg19/')
cd14_facs.data <- as.matrix(cd14_facs.data)
rownames(cd14_facs.data) <- Hs_mapping$ensembl_id[match(rownames(cd14_facs.data),
                                                        Hs_mapping$symbol)]
cd14_facs.data <- cd14_facs.data[!is.na(rownames(cd14_facs.data)),]
cd14.test <- cd14_facs.data[,1:100] # CD14 withheld cells
```

```
cd14_facs.data <- cd14_facs.data[,101:ncol(cd14_facs.data)] # CD14 training cells

## NK: https://support.10xgenomics.com/single-cell-gene-expression/datasets/1.1.0/cd56_nk
nk_facs.data <- Read10X(data.dir = '../filtered_matrices_nk/hg19/')
nk_facs.data <- as.matrix(nk_facs.data)
rownames(nk_facs.data) <- Hs_mapping$ensembl_id[match(rownames(nk_facs.data),
                                                      Hs_mapping$symbol)]
nk_facs.data <- nk_facs.data[!is.na(rownames(nk_facs.data)),]
nk.test <- nk_facs.data[,1:100] # NK withheld cells
nk_facs.data <- nk_facs.data[,101:5101] # NK training cells
```

Next, we fit our model to these training data. We combine the training data into a single matrix and call the `trainAllReference` function on this matrix, alongside a vector of cell-type labels. By default, this function will only consider the subset of discriminative genes needed for classification. For now, to illustrate the uses of our method additional to classification, we will consider all genes by setting `discrim_only=FALSE`. (Note that regardless of whether this parameter is set to `TRUE`, as it is by default, or `FALSE`, the classification function will still only use the subset of discriminative genes; setting this parameter to `TRUE` just has the effect of speeding up the training process by only fitting to that set of genes.)

```
## Input a single matrix with a vector of labels
common_pbmcs_genes <- Reduce(intersect,list(rownames(cd4_facs.data),
                                            rownames(cd14_facs.data),
                                                rownames(cd8_facs.data),
                                            rownames(nk_facs.data)))
pbmcs_reference <- as.matrix(cbind(cd4_facs.data[common_pbmcs_genes,],
                                   cd14_facs.data[common_pbmcs_genes,],
                                   cd8_facs.data[common_pbmcs_genes,],
                                   nk_facs.data[common_pbmcs_genes,]))
pbmcs_reference_labels <- c(rep('CD4',dim(cd4_facs.data)[2]),rep('CD14',dim(cd14_facs.data)[2]),
                            rep('CD8',dim(cd8_facs.data)[2]),rep('NK',dim(nk_facs.data)[2]))
pbmcs.d <- trainAllReference(pbmcs_reference,pbmcs_reference_labels,discrim_only=FALSE)
```

The list `pbmcs.d` contains one table for each inputted cell-type, which indicates each gene's empirical rate, probability of belonging to the off-low latent state, and probability of belonging to the off-high latent state. This is used as input into the `classifyTarget` function. However, if our goal is to examine each cell-type's barcode, we could pass this list into the `getBarcode` function, which will present the probability that each gene is on in each cell-type. This could then be used beyond the cell-type annotation context to study gene expression within a single cell-type, to compare genes across cell-types, and to identify markers.

As an example, we show below how we can find the barcodes for each cell-type, then identify genes with a high probability of being on in NK cells, but a low probability of being on in the others (i.e. potential markers for this cell-type). Finally, we show how we can identify markers for NK cells compared to global patterns of gene expression, rather than the other cell-types under consideration here.

```
## Obtain barcodes
barcodes <- getBarcode(pbmcs.d)

## Identify genes likely to be on in NK cells, but likely to be off in the others
findMarkers('NK',barcodes,thresh_up=0.95,thresh_below=0.05)
```

```
##                          CD4         CD14          CD8        NK
## ENSG00000143184 9.035083e-06 3.708215e-05 9.331617e-03 0.9512755
## ENSG00000117560 1.623782e-06 6.891325e-07 7.614545e-05 0.9566056
```

```
## ENSG00000115607 1.684319e-11 9.289070e-11 4.401534e-05 0.9661884
## ENSG00000180871 1.802118e-10 1.739938e-04 3.966463e-11 0.9981462
## ENSG00000171596 1.600303e-11 2.432488e-09 1.725424e-08 0.9541501
## ENSG00000086200 5.839253e-03 2.990801e-03 1.462233e-03 0.9872516
## ENSG00000104343 1.375974e-05 4.928969e-05 2.015292e-03 0.9999558
## ENSG00000150045 1.321569e-04 3.698147e-03 3.590472e-02 0.9979354
## ENSG00000240403 3.364929e-10 1.002669e-11 4.126948e-08 0.9773942
```

```
## Identify genes likely to be on in NK cells, and not in global patterns
findMarkers('NK',barcodes,thresh_up=0.99,thresh_below=0.01,relative=F)
```

```
##                         NK
## ENSG00000158062 0.9999977
## ENSG00000126088 1.0000000
## ENSG00000173660 1.0000000
## ENSG00000162402 0.9994523
## ENSG00000162607 1.0000000
## ENSG00000077254 1.0000000
## ENSG00000143569 1.0000000
## ENSG00000160714 1.0000000
## ENSG00000143222 1.0000000
## ENSG00000173960 1.0000000
## ENSG00000115464 1.0000000
## ENSG00000169764 1.0000000
## ENSG00000115446 1.0000000
## ENSG00000136731 0.9995615
## ENSG00000144224 1.0000000
## ENSG00000170035 1.0000000
## ENSG00000180871 0.9981462
## ENSG00000184182 1.0000000
## ENSG00000170142 1.0000000
## ENSG00000153560 0.9994523
## ENSG00000010256 1.0000000
## ENSG00000144744 1.0000000
## ENSG00000163714 1.0000000
## ENSG00000163960 0.9973481
## ENSG00000078140 1.0000000
## ENSG00000033178 1.0000000
## ENSG00000138768 1.0000000
## ENSG00000145337 0.9915257
## ENSG00000109332 1.0000000
## ENSG00000164405 1.0000000
## ENSG00000119048 1.0000000
## ENSG00000131508 1.0000000
## ENSG00000137288 1.0000000
## ENSG00000024048 1.0000000
## ENSG00000198833 1.0000000
## ENSG00000014123 1.0000000
## ENSG00000152818 1.0000000
## ENSG00000186591 0.9999996
## ENSG00000157741 0.9931891
## ENSG00000124486 1.0000000
## ENSG00000130985 1.0000000
## ENSG00000102226 1.0000000
```

```
## ENSG00000126756 1.0000000
## ENSG00000188021 1.0000000
## ENSG00000241343 1.0000000
## ENSG00000077721 1.0000000
## ENSG00000125351 1.0000000
## ENSG00000169139 1.0000000
## ENSG00000215114 0.9999998
## ENSG00000104343 0.9999558
## ENSG00000156467 1.0000000
## ENSG00000104529 1.0000000
## ENSG00000107341 1.0000000
## ENSG00000137073 0.9958586
## ENSG00000135018 1.0000000
## ENSG00000167118 1.0000000
## ENSG00000151461 1.0000000
## ENSG00000072401 1.0000000
## ENSG00000188690 1.0000000
## ENSG00000170242 1.0000000
## ENSG00000254772 0.9988525
## ENSG00000162191 1.0000000
## ENSG00000175567 1.0000000
## ENSG00000110344 1.0000000
## ENSG00000135655 1.0000000
## ENSG00000177889 1.0000000
## ENSG00000151148 1.0000000
## ENSG00000150991 1.0000000
## ENSG00000122042 1.0000000
## ENSG00000120686 1.0000000
## ENSG00000134882 1.0000000
## ENSG00000169062 1.0000000
## ENSG00000133983 0.9999959
## ENSG00000213145 0.9999122
## ENSG00000114062 1.0000000
## ENSG00000159459 0.9999998
## ENSG00000138592 1.0000000
## ENSG00000140455 1.0000000
## ENSG00000138629 1.0000000
## ENSG00000140367 1.0000000
## ENSG00000103275 1.0000000
## ENSG00000187555 0.9999692
## ENSG00000182108 0.9999973
## ENSG00000140740 1.0000000
## ENSG00000103353 0.9999469
## ENSG00000103005 1.0000000
## ENSG00000103194 0.9999997
## ENSG00000132388 1.0000000
## ENSG00000124422 1.0000000
## ENSG00000109103 1.0000000
## ENSG00000159202 1.0000000
## ENSG00000108622 1.0000000
## ENSG00000132478 1.0000000
## ENSG00000257949 0.9976273
## ENSG00000101557 1.0000000
## ENSG00000265681 1.0000000
```

```
## ENSG00000198258 1.0000000
## ENSG00000076662 0.9999916
## ENSG00000221983 1.0000000
## ENSG00000169021 1.0000000
## ENSG00000105176 1.0000000
## ENSG00000126261 1.0000000
## ENSG00000105698 1.0000000
## ENSG00000167578 1.0000000
## ENSG00000185651 1.0000000
## ENSG00000240972 1.0000000
## ENSG00000184076 1.0000000
## ENSG00000156256 1.0000000
## ENSG00000160201 0.9999999
## ENSG00000184787 1.0000000
```

Finally, we can use the original object `pbmcs.d` to annotate our withheld test cells, which should be placed in a single matrix. If we want the possibility of detecting cells that do not belong to any of the cell-types in the reference, we can set `other=T`. Further, we can specify `return.probs=T` if we want a matrix of probabilistic assignments rather than just a vector of highest-probability cell-type labels. We show both below:

```
## Put withheld test cells in one matrix
pbmcs_withheld <- as.matrix(cbind(cd4.test[common_pbmcs_genes,],
                                  cd14.test[common_pbmcs_genes,],
                                    cd8.test[common_pbmcs_genes,],
                                  nk.test[common_pbmcs_genes,]))
true_labels <- c(rep('CD4',100),rep('CD14',100),rep('CD8',100),rep('NK',100))

## Annotate!
annotation <- classifyTarget(pbmcs_withheld,pbmcs.d,other=T)
table(annotation,true_labels)
```

```
##           true_labels
## annotation CD14 CD4 CD8 NK
##       CD14   99   0   0  0
##       CD4     0  94   4  0
##       CD8     1   6  96  1
##       NK      0   0   0 99
```

```
## Return probabilistic assignments instead
probabilistic_annotation <- classifyTarget(pbmcs_withheld,pbmcs.d,other=T,
                                           return.probs=T)
head(round(probabilistic_annotation,2))
```

```
##       CD4 CD14  CD8 NK other
## [1,] 1.00    0 0.00  0     0
## [2,] 0.98    0 0.02  0     0
## [3,] 0.94    0 0.06  0     0
## [4,] 0.50    0 0.50  0     0
## [5,] 0.14    0 0.86  0     0
## [6,] 0.98    0 0.02  0     0
```