Theory

Pooled or bulk RNA-seq measurements represent averaged transcriptional states of potentially many transcriptionally distinct cell-types at differing proportions. In order to estimate the proportions of these different cell-types, many computational deconvolution approaches have been developed. These deconvolution approaches have generally relied on the existence of a 'pure' cell-type reference, often generated from an external single-cell RNA-seq dataset.

The general intuition is that given the transcriptional profiles of cell-type A `x1` and cell-type B `x2`, then if bulk RNA-seq measurement `y` that represent a mix of cell-type A and cell-type B can be represented as `y = alpha * x1 + beta * x2` where `alpha` and `beta` can be solved. We can train a supervised machine learning classifier such as an `svm` classifier based on a set of simulated `y`s where `alpha` and `beta` are known. This is the basis of CIBERSORT and other similar deconvolution methods.

A simple simulation is shown below:

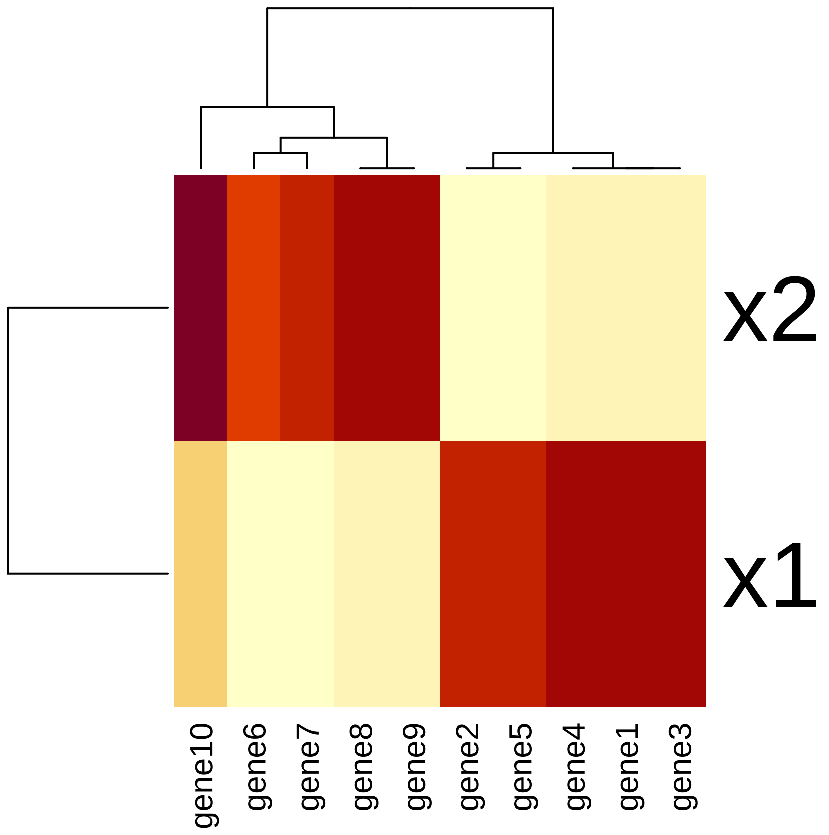
```r
## baseline expression
set.seed(0)
base = abs(round(rnorm(10, 10)))
names(base) <- paste0('gene', 1:10)
head(base)
```

```
## gene1 gene2 gene3 gene4 gene5 gene6
##    11    10    11    11    10     8
```

```r
## cell-type A upregulates genes 1 to 5
x1 = base
x1[1:5] = base[1:5] + 10

## cell-type B upregulates genes 6 to 10
x2 = base
x2[6:10] = base[6:10] + 10

## visualize transcriptional distinctness
heatmap(rbind(x1, x2))
```



```r
## train classifier to predict proportions of cell-type A
## given a mixture of cell-type A and cell-type B
train.data <- do.call(rbind, lapply(seq(1,100, by=5), function(i) {
  ground.truth <- c(i, 100-i)
  names(ground.truth) <- c('ctA', 'ctB')
  y = ground.truth[1]*x1 + ground.truth[2]*x2
  c(y, ground.truth)
}))
rownames(train.data) <- paste0('sim', 1:nrow(train.data))
head(train.data)
```

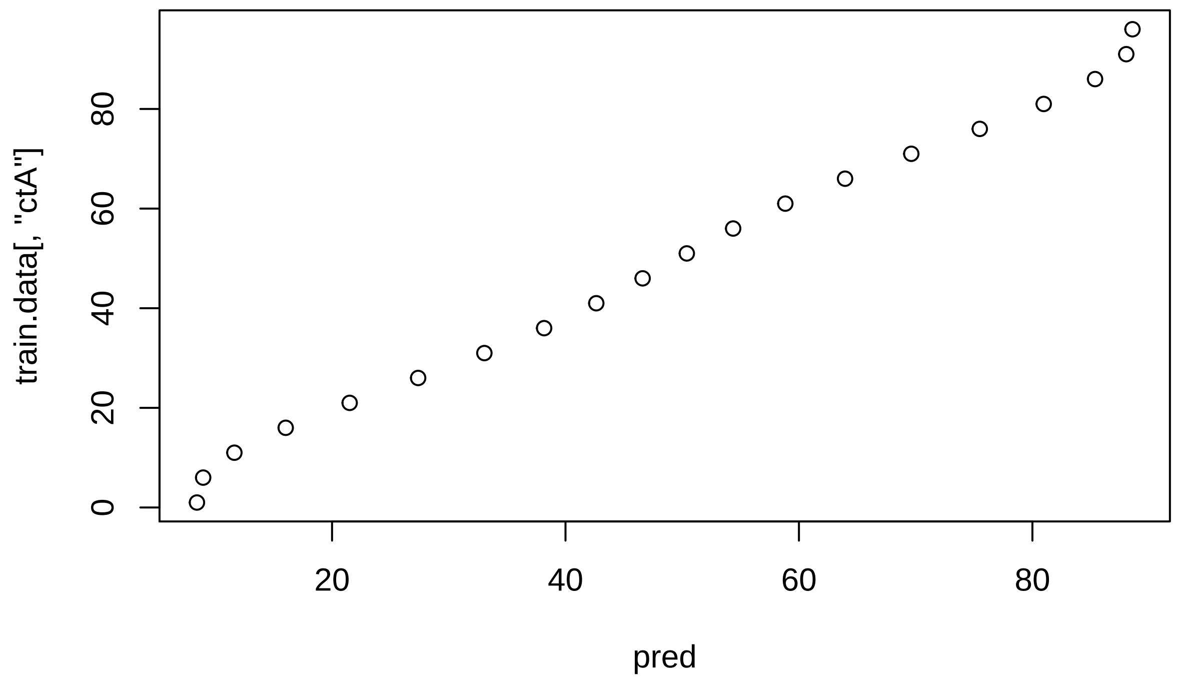
```
##      gene1 gene2 gene3 gene4 gene5 gene6 gene7 gene8 gene9 gene10 ctA ctB
## sim1  1110  1010  1110  1110  1010  1790  1890  1990  1990   2190   1  99
## sim2  1160  1060  1160  1160  1060  1740  1840  1940  1940   2140   6  94
## sim3  1210  1110  1210  1210  1110  1690  1790  1890  1890   2090  11  89
## sim4  1260  1160  1260  1260  1160  1640  1740  1840  1840   2040  16  84
## sim5  1310  1210  1310  1310  1210  1590  1690  1790  1790   1990  21  79
## sim6  1360  1260  1360  1360  1260  1540  1640  1740  1740   1940  26  74
```

```r
## remove truth from training
train.data.sub <- as.data.frame(train.data[, names(x1)])
library(e1071)
model <- e1071::svm(train.data.sub, train.data[,'ctA'])
print(model)
```

```
##
## Call:
## svm.default(x = train.data.sub, y = train.data[, "ctA"])
##
```

```
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  radial
##        cost:  1
##       gamma:  0.1
##     epsilon:  0.1
##
##
## Number of Support Vectors:  4
```

```
# test with train data
pred <- predict(model, train.data.sub)
plot(pred, train.data[, 'ctA'])
```



```
cor(pred, train.data[, 'ctA'])
```

```
## [1] 0.9977975
```

```
## now create real test
ground.truth <- c(30, 70)
y = ground.truth[1]*x1 + ground.truth[2]*x2
## hack to fix error
## https://stackoverflow.com/questions/24829674/r-random-forest-error-type-of-predictors-in-new-data-do
ytest <- rbind(y, train.data.sub)
y <- ytest[1,]
print(y)
```

```
##   gene1 gene2 gene3 gene4 gene5 gene6 gene7 gene8 gene9 gene10
## 1  1400  1300  1400  1400  1300  1500  1600  1700  1700   1900
```

```
pred <- predict(model, y)
print(pred)
```

```
##        1
## 31.95239
```

```
## pretty close to 30 as expected
```

In ST data, each ST spot represents the averaged transcriptional state of 100s of cells that may represent many transcriptionally distinct cell-types at differing proportions. However, unlike in bulk RNA-seq data where we only have 1 pooled measurement, in ST data, we have 100s of spots, each representing a different mixture of the same set of cell-types within the tissue. We should be able to take advantage of this natural variation to deconvolve expression without an external single cell reference using unsupervised machine learning approaches such as latent direchlet allocation.

Latent direchlet allocation (LDA) is a machine learning approach often used in topic modeling. Given a set of articles, each with many many words, LDA will learn the underlying set of topics, which are defined by different words, and assess the proportional representation of each topic in each article. An analogy can be made with ST data where given a set of spots, each with many many genes, LDA can learn the underlying set of cell types, which are defined by expression of different genes, and assess the proportional representation of each cell-type in each spot.

Consider if `train.data.sub` was the transcriptional profiles of each spot. Note, in LDA, we do have to define the number of topics (or cell-types). Here, we already know that there are two cell-types. However, we do not have to provide any ground truth cell-type proportion labels.

```
library(topicmodels)
library(slam)
library(Matrix)
K = 2
ap_lda <- topicmodels::LDA(slam::as.simple_triplet_matrix(as.matrix(train.data.sub)),
                           k=K, control = list(seed = 0))
ap_lda
```

```
## A LDA_VEM topic model with 2 topics.
```

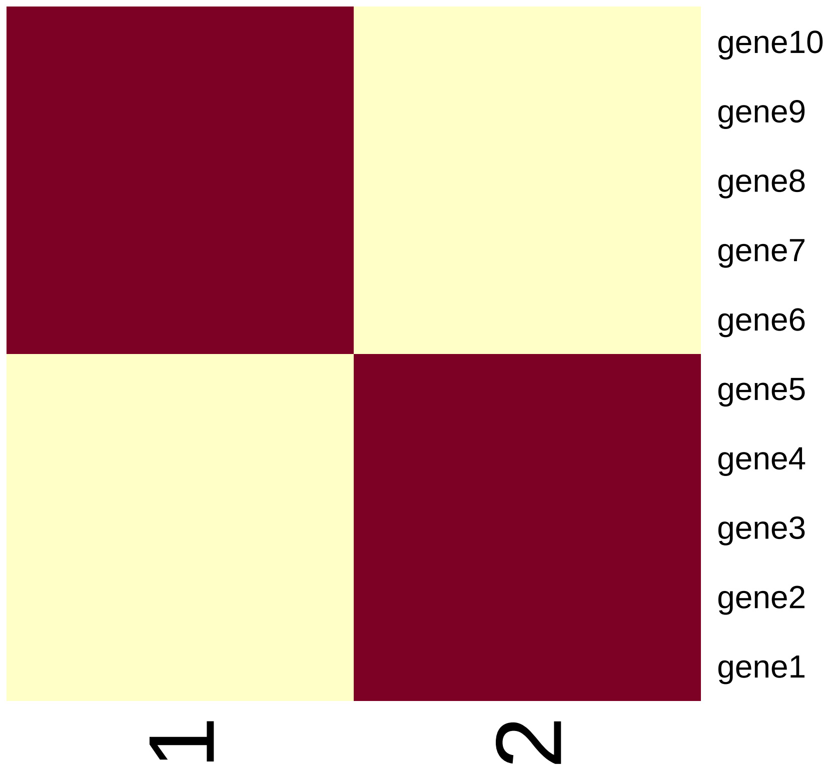
```
## association of each gene with each cell-type
library(tidytext)
ap_topics <- tidy(ap_lda, matrix = "beta")
ap_topics
```

```
## # A tibble: 20 x 3
##    topic term    beta
##    <int> <chr>  <dbl>
##  1     1 gene1  0.0780
##  2     2 gene1  0.140
##  3     1 gene2  0.0713
##  4     2 gene2  0.134
##  5     1 gene3  0.0762
##  6     2 gene3  0.143
##  7     1 gene4  0.0770
##  8     2 gene4  0.142
##  9     1 gene5  0.0703
## 10     2 gene5  0.135
## 11     1 gene6  0.117
## 12     2 gene6  0.0447
## 13     1 gene7  0.119
## 14     2 gene7  0.0580
## 15     1 gene8  0.125
## 16     2 gene8  0.0650
## 17     1 gene9  0.128
## 18     2 gene9  0.0611
## 19     1 gene10 0.138
## 20     2 gene10 0.0780
```

```
ap_topics.mat <- t(cast_sparse(ap_topics, topic, term, beta))
dim(ap_topics.mat)
```

```
## [1] 10  2
```

```
heatmap(as.matrix(ap_topics.mat), scale='row', Rowv=NA, Colv=NA)
```



In this case, LDA correctly identifies two cell-types, where cell-type 1 is marked by upregulation of genes 6 to 10 (our real cell-type B), and cell-type 2 is marked by upregulation of genes 1 to 5 (our real cell-type A). Likewise, we can compare the learned proportional representation to our real simulated presentation.

```
## proportional representation of each topic in each document
## or each cell-type in each spot
ap_documents <- tidy(ap_lda, matrix = "gamma")
ap_documents
```

```
## # A tibble: 40 x 3
##    document topic gamma
##    <chr>    <int> <dbl>
##  1 sim1         1 0.997
##  2 sim2         1 0.987
##  3 sim3         1 0.950
##  4 sim4         1 0.902
##  5 sim5         1 0.852
##  6 sim6         1 0.802
##  7 sim7         1 0.751
##  8 sim8         1 0.700
##  9 sim9         1 0.649
## 10 sim10        1 0.599
## # ... with 30 more rows
```

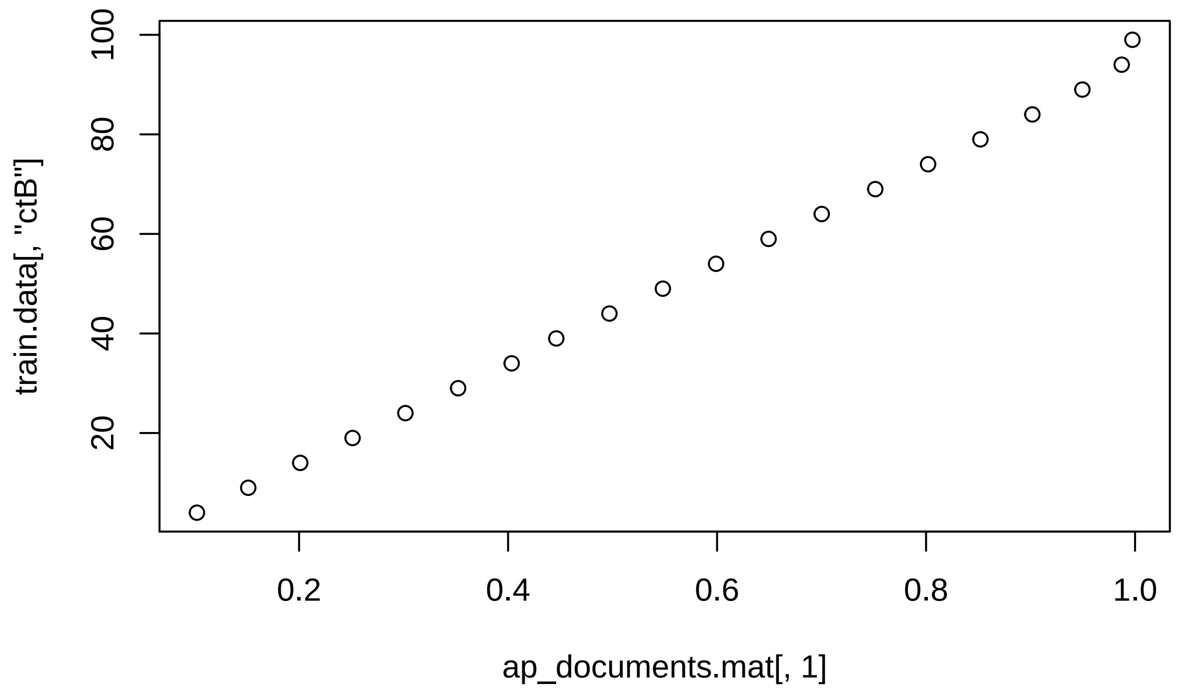
```
ap_documents.mat <- cast_sparse(ap_documents, document, topic, gamma)
dim(ap_documents.mat)
```

```
## [1] 20  2
```

```
head(ap_documents.mat)
```

```
## 6 x 2 sparse Matrix of class "dgCMatrix"
##              1           2
## sim1 0.9974741 0.002525873
## sim2 0.9872026 0.012797438
## sim3 0.9495838 0.050416230
## sim4 0.9017185 0.098281490
## sim5 0.8519966 0.148003436
## sim6 0.8019386 0.198061402
```

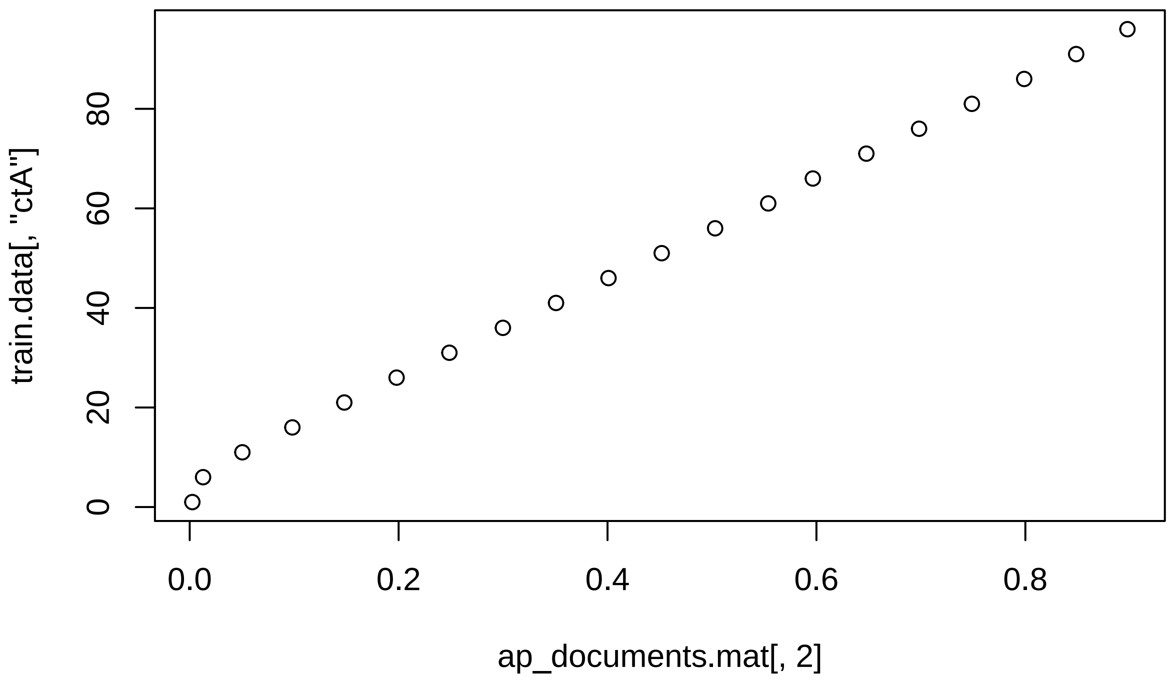
```
plot(ap_documents.mat[,1], train.data[, 'ctB'])
```



```
cor(ap_documents.mat[,1], train.data[, 'ctB'])
```

```
## [1] 0.9993066
```

```
plot(ap_documents.mat[,2], train.data[, 'ctA'])
```



```r
cor(ap_documents.mat[,2], train.data[, 'ctA'])
```

```
## [1] 0.9993066
```

Questions

1. Can we create a more realistic simulation using single-cell RNA-seq data? (test.R can help you get started
2. What happens if we choose a wrong K? Can we detect that the chosen K is wrong?
3. Should we use all words (genes) or only train on words that are known to be variable across topics (genes that are overdispersed across cell-types)?
4. This approach relies on variability of cell-type proportions in the training data. What happens if there is little variability? How will we know if there is enough variability in our data?