# Assignment 6

### Statistics and Data Science 365/565

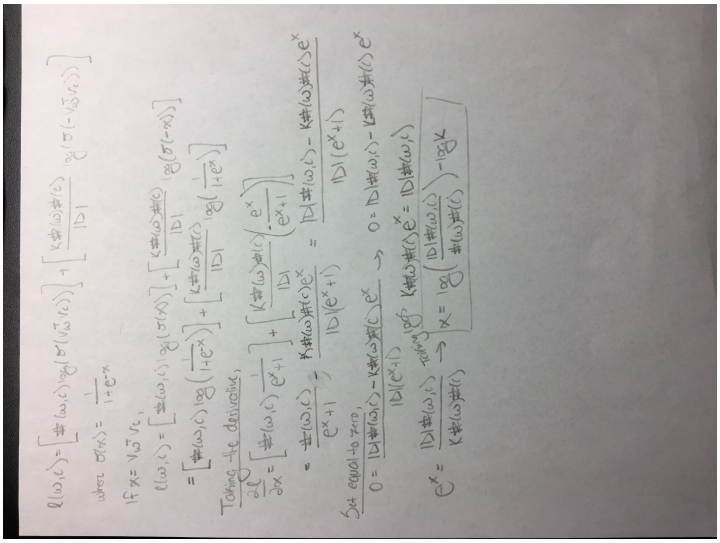
### Due: April 19th (before 11:59 pm)

```
setwd("/Users/mark/Documents/Spring2019Classes/S&DS365/Homework/Homework6")
```

# Problem 1

## Problem 1 Part a

```
question1Path <- "/Users/mark/Documents/Spring2019Classes/S&DS365/Homework/Homework6/Que
stion1a.JPG"
knitr::include_graphics(path = question1Path)
```



## Problem 1 Part b

By constructing our word embeddings as the rank-d SVD of a |V| x |V| matrix M, calculations can be made more computationally efficient. In particular, since we can rewrite $v_w^T v_c$ as a function of the count data (w, c), w, and c, our procedure is much more efficient on large amounts of words. Moreover, using SVD on the matrix M

Writing our local objective in the form from part (a) poses some computational challenges, since the matrix in question is dense and very high-dimensional. However, we can instead choose to construct embeddings as a rank-d SVD of a |V| x |V| matrix M, thereby approximating the high-dimensional original matrix with a more computationally efficient one instead. Specifically, the matrix that we would use is

$$M = U * (\sum) * V^T$$

, which upon simplification equals

$$U(\sum)^{\frac{1}{2}}(\sum)^{\frac{1}{2}}V^T$$

where $(\sum)_d$ is the diagonal matrix formed from the top d singular values and $U_d$ and $V_d$ are the matrices produced by selecting the corresponding coumns from U and V from the SVD of M.

# Problem 2

```
library(text2vec)
library(Matrix)
text8_file <- "/Users/mark/Documents/Spring2019Classes/S&DS365/Homework/Homework6/text8"
wiki <- readLines(text8_file, n = 1, warn = FALSE)
tokens <- space_tokenizer(wiki)
it <- itoken(tokens, progressbar = FALSE)
vocab <- create_vocabulary(it)
vocab <- prune_vocabulary(vocab, term_count_min = 100L)
vectorizer <- vocab_vectorizer(vocab)
```

PMI Embeddings:

```
tcm <- create_tcm(it, vectorizer, skip_grams_window = 5L, skip_grams_window_context = "s
ymmetric", weights = 1 / rep(1, 5L))

co <- tcm + t(tcm)

dimnames(co) <- dimnames(tcm)

# the cooccurence matrix looks at how often a combination of word + context appears

# rownames(co)
# colnames(co)
```

I will create the matrix M in the form of log(#$(w_i, w_i) * |D|$) - log(#$(w_i$ * #$(w_j)$)) - log k$

```
# To find the first term:

D <- length(co)
term1 <- log((co + 1) * D)

# To find the second term:
rowWords <- rowSums(co)
columnWords <- colSums(co)
productRowColumnSums <- rowWords %*% t(columnWords)
term2 <- log(productRowColumnSums)

# Our third term, log(k), is 0 since k = 1

### Calculating the matrix M:
M <- term1 - term2
```

```
# Now to take the rank-D SVD:
library(irlba)
svd <- irlba(M, 50)
```

```
# Now to construct W:
U <- svd$u
sigma <- diag(sqrt(svd$d))

W <- U %*% sigma

# Set rownames to match actual words
rownames(W) <- rownames(co) # These are the learned embeddings
```

Local GloVe Embeddings

```
# Run stochastic gradient descent to obtain GloVe word embeddings stored in the g.embed
 variable
set.seed(1987)
glove_local <- GlobalVectors$new(word_vectors_size = 50,
                           vocabulary = vocab,
                           x_max = 10)
g.embed <- fit_transform(tcm, glove_local, n_iter = 20)
```

```
## INFO [2019-04-17 22:49:43] 2019-04-17 22:49:43 - epoch 1, expected cost 0.0974
## INFO [2019-04-17 22:49:47] 2019-04-17 22:49:47 - epoch 2, expected cost 0.0672
## INFO [2019-04-17 22:49:52] 2019-04-17 22:49:52 - epoch 3, expected cost 0.0613
## INFO [2019-04-17 22:49:57] 2019-04-17 22:49:57 - epoch 4, expected cost 0.0585
## INFO [2019-04-17 22:50:01] 2019-04-17 22:50:01 - epoch 5, expected cost 0.0567
## INFO [2019-04-17 22:50:07] 2019-04-17 22:50:07 - epoch 6, expected cost 0.0555
## INFO [2019-04-17 22:50:12] 2019-04-17 22:50:12 - epoch 7, expected cost 0.0546
## INFO [2019-04-17 22:50:17] 2019-04-17 22:50:17 - epoch 8, expected cost 0.0540
## INFO [2019-04-17 22:50:22] 2019-04-17 22:50:22 - epoch 9, expected cost 0.0534
## INFO [2019-04-17 22:50:26] 2019-04-17 22:50:26 - epoch 10, expected cost 0.0530
## INFO [2019-04-17 22:50:33] 2019-04-17 22:50:33 - epoch 11, expected cost 0.0527
## INFO [2019-04-17 22:50:39] 2019-04-17 22:50:39 - epoch 12, expected cost 0.0524
## INFO [2019-04-17 22:50:45] 2019-04-17 22:50:45 - epoch 13, expected cost 0.0521
## INFO [2019-04-17 22:50:52] 2019-04-17 22:50:52 - epoch 14, expected cost 0.0519
## INFO [2019-04-17 22:51:00] 2019-04-17 22:51:00 - epoch 15, expected cost 0.0517
## INFO [2019-04-17 22:51:10] 2019-04-17 22:51:10 - epoch 16, expected cost 0.0515
## INFO [2019-04-17 22:51:18] 2019-04-17 22:51:18 - epoch 17, expected cost 0.0514
## INFO [2019-04-17 22:51:25] 2019-04-17 22:51:25 - epoch 18, expected cost 0.0512
## INFO [2019-04-17 22:51:30] 2019-04-17 22:51:30 - epoch 19, expected cost 0.0511
## INFO [2019-04-17 22:51:35] 2019-04-17 22:51:35 - epoch 20, expected cost 0.0510
```

Pre-trained GloVe Embeddings

```
# Load embeddings
load("/Users/mark/Documents/Spring2019Classes/S&DS365/Homework/Homework6/pre-trained-glo
ve.RData")
```

# Problem 2 Part a

```
library(tidyverse)
```

For the PMI embeddings

```
PMIClosest <- sim2(W)

#### Looking at Yale:
PMIClosestYale <- PMIClosest[which(rownames(PMIClosest) == "yale"), ] %>% sort(decreasin
g = TRUE)

# Selecting the top 5 words:
PMIClosestYale[1:5]
```

```
##      yale princeton graduated   cornell dartmouth
## 1.0000000 0.9803101 0.9650239 0.9648130 0.9590048
```

```
#### Looking at physics:
PMIClosestPhysics <- PMIClosest[which(rownames(PMIClosest) == "physics"), ] %>% sort(dec
reasing = TRUE)

# Selecting the top 5 words:
PMIClosestPhysics[1:5]
```

```
##      physics    mechanics      quantum    chemistry mathematical
##    1.0000000    0.8785929    0.8660823    0.8386913    0.8014284
```

```
### Looking at republican:
PMIClosestRepublican <- PMIClosest[which(rownames(PMIClosest) == "republican"), ] %>% so
rt(decreasing = TRUE)

# Selecting the top 5 words:
PMIClosestRepublican[1:5]
```

```
##   republican presidential    democrats    coalition    candidate
##    1.0000000    0.9323793    0.9084006    0.8885913    0.8874529
```

```
### Looking at einstein:
PMIClosestEinstein <- PMIClosest[which(rownames(PMIClosest) == "einstein"), ] %>% sort(d
ecreasing = TRUE)

# Selecting the top 5 words:
PMIClosestEinstein[1:5]
```

```
##   einstein     newton relativity    maxwell experiment
##  1.0000000  0.9485829  0.9399612  0.9112443  0.8996901
```

```
### Looking at algebra:
PMIClosestAlgebra <- PMIClosest[which(rownames(PMIClosest) == "algebra"), ] %>% sort(dec
reasing = TRUE)

# Selecting the top 5 words:
PMIClosestAlgebra[1:5]
```

```
##   algebra   theorem    finite algebraic  calculus
## 1.0000000 0.9512161 0.9362144 0.9354340 0.9139258
```

```
### Looking at fish:
PMIClosestFish <- PMIClosest[which(rownames(PMIClosest) == "fish"), ] %>% sort(decreasin
g = TRUE)

# Selecting the top 5 words:
PMIClosestFish[1:5]
```

```
##      fish     fruit    plants      meat     plant
## 1.0000000 0.9121648 0.8843248 0.8831505 0.8782498
```

For the local GloVe embeddings

```
localGloveClosest <- sim2(g.embed)
#### Looking at Yale:
localGloveClosestYale <- localGloveClosest[which(rownames(localGloveClosest) == "yale"),
] %>% sort(decreasing = TRUE)

# Selecting the top 5 words:
localGloveClosestYale[1:5]
```

```
##       yale    harvard  princeton      press university
##  1.0000000  0.6663194  0.6355988  0.6245500  0.5945910
```

```
#### Looking at physics:
localGloveClosestPhysics <- localGloveClosest[which(rownames(localGloveClosest) == "phys
ics"), ] %>% sort(decreasing = TRUE)

# Selecting the top 5 words:
localGloveClosestPhysics[1:5]
```

```
##     physics   chemistry   mechanics     quantum mathematics
##   1.0000000   0.7646421   0.7315312   0.6821681   0.6640930
```

```
### Looking at republican:
localGloveClosestRepublican <- localGloveClosest[which(rownames(localGloveClosest) == "r
epublican"), ] %>% sort(decreasing = TRUE)

# Selecting the top 5 words:
localGloveClosestRepublican[1:5]
```

```
##   republican presidential   democratic     election      senator
##    1.0000000    0.7203823    0.6945851    0.6607051    0.6423452
```

```
### Looking at einstein:
localGloveClosestEinstein <- localGloveClosest[which(rownames(localGloveClosest) == "ein
stein"), ] %>% sort(decreasing = TRUE)

# Selecting the top 5 words:
localGloveClosestEinstein[1:5]
```

```
##   einstein relativity  mechanics    quantum    physics
##  1.0000000  0.8155460  0.6521713  0.6119580  0.6052684
```

```
### Looking at algebra:
localGloveClosestAlgebra <- localGloveClosest[which(rownames(localGloveClosest) == "alge
bra"), ] %>% sort(decreasing = TRUE)

# Selecting the top 5 words:
localGloveClosestAlgebra[1:5]
```

```
##     algebra   algebraic      finite mathematics dimensional
##   1.0000000   0.6948000   0.6407171   0.6312096   0.6025251
```

```
### Looking at fish:
localGloveClosestFish <- localGloveClosest[which(rownames(localGloveClosest) == "fish"),
] %>% sort(decreasing = TRUE)

# Selecting the top 5 words:
localGloveClosestFish[1:5]
```

```
##      fish      food   animals     birds      meat
## 1.0000000 0.6791619 0.6719372 0.6719370 0.6659808
```

For the pre-trained GloVe embeddings

```
pretrainedGloveClosest <- sim2(pt.glove)

#### Looking at Yale:
pretrainedGloveClosestYale <- pretrainedGloveClosest[which(rownames(pretrainedGloveClose
st) == "yale"), ] %>% sort(decreasing = TRUE)

# Selecting the top 5 words:
pretrainedGloveClosestYale[1:5]
```

```
##      yale   harvard princeton   cornell  graduate
## 1.0000000 0.9597434 0.9377335 0.8834298 0.8726720
```

```
#### Looking at physics:
pretrainedGloveClosestPhysics <- pretrainedGloveClosest[which(rownames(pretrainedGloveCl
osest) == "physics"), ] %>% sort(decreasing = TRUE)

# Selecting the top 5 words:
pretrainedGloveClosestPhysics[1:5]
```

```
##     physics   chemistry mathematics theoretical     science
##   1.0000000   0.8995857   0.8771122   0.8480029   0.8314355
```

```
### Looking at republican:
pretrainedGloveClosestRepublican <- pretrainedGloveClosest[which(rownames(pretrainedGlov
eClosest) == "republican"), ] %>% sort(decreasing = TRUE)

# Selecting the top 5 words:
pretrainedGloveClosestRepublican[1:5]
```

```
##  republican    democrat  democratic republicans   democrats
##   1.0000000   0.9142889   0.9113237   0.9098939   0.9068174
```

```
### Looking at einstein:
pretrainedGloveClosestEinstein <- pretrainedGloveClosest[which(rownames(pretrainedGloveC
losest) == "einstein"), ] %>% sort(decreasing = TRUE)

# Selecting the top 5 words:
pretrainedGloveClosestEinstein[1:5]
```

```
##   einstein relativity       bohr    physics      freud
##  1.0000000  0.7438953  0.7286732  0.7018453  0.6971669
```

```
### Looking at algebra:
pretrainedGloveClosestAlgebra <- pretrainedGloveClosest[which(rownames(pretrainedGloveCl
osest) == "algebra"), ] %>% sort(decreasing = TRUE)

# Selecting the top 5 words:
pretrainedGloveClosestAlgebra[1:5]
```

```
##     algebra    geometry   algebraic    algebras associative
##   1.0000000   0.8046252   0.7837221   0.7446803   0.7439568
```

```
### Looking at fish:
pretrainedGloveClosestFish <- pretrainedGloveClosest[which(rownames(pretrainedGloveClose
st) == "fish"), ] %>% sort(decreasing = TRUE)

# Selecting the top 5 words:
pretrainedGloveClosestFish[1:5]
```

```
##      fish    salmon      meat     birds      bird
## 1.0000000 0.8340227 0.7995401 0.7755683 0.7638516
```

I will choose the following query words: card, computer, cotton, child, corn. I will look at the 5 closest words for each:

Using PMI:

```
#### Looking at card:
PMIClosestCard <- PMIClosest[which(rownames(PMIClosest) == "card"), ] %>% sort(decreasin
g = TRUE)

# Selecting the top 5 words:
PMIClosestCard[1:5]
```

```
##      card     cards     match       box     extra
## 1.0000000 0.9266516 0.7883943 0.7830625 0.7771155
```

```
#### Looking at computer:
PMIClosestComputer <- PMIClosest[which(rownames(PMIClosest) == "computer"), ] %>% sort(d
ecreasing = TRUE)

# Selecting the top 5 words:
PMIClosestComputer[1:5]
```

```
##   computer  computers   software    machine technology
##  1.0000000  0.8131871  0.7798565  0.7527352  0.7517043
```

```
### Looking at cotton:
PMIClosestCotton <- PMIClosest[which(rownames(PMIClosest) == "cotton"), ] %>% sort(decre
asing = TRUE)

# Selecting the top 5 words:
PMIClosestCotton[1:5]
```

```
##    cotton   tobacco     wheat      corn    timber
## 1.0000000 0.9749465 0.9713322 0.9683395 0.9678838
```

```
### Looking at child:
PMIClosestChild <- PMIClosest[which(rownames(PMIClosest) == "child"), ] %>% sort(decreas
ing = TRUE)

# Selecting the top 5 words:
PMIClosestChild[1:5]
```

```
##     child    mother     woman  children   parents
## 1.0000000 0.8886911 0.8670071 0.8588512 0.8571085
```

```
### Looking at corn:
PMIClosestCorn <- PMIClosest[which(rownames(PMIClosest) == "corn"), ] %>% sort(decreasin
g = TRUE)

# Selecting the top 5 words:
PMIClosestCorn[1:5]
```

```
##      corn     wheat    fruits     dairy   tobacco
## 1.0000000 0.9880161 0.9805438 0.9796897 0.9779329
```

Using local GloVe

```
#### Looking at card:
localGloveClosestCard <- localGloveClosest[which(rownames(localGloveClosest) == "card"),
] %>% sort(decreasing = TRUE)

# Selecting the top 5 words:
localGloveClosestCard[1:5]
```

```
##      card     cards      game     magic     round
## 1.0000000 0.8608121 0.6254737 0.5998769 0.5799826
```

```
#### Looking at computer:
localGloveClosestComputer <- localGloveClosest[which(rownames(localGloveClosest) == "com
puter"), ] %>% sort(decreasing = TRUE)

# Selecting the top 5 words:
localGloveClosestComputer[1:5]
```

```
##   computer  computers   hardware technology    digital
##  1.0000000  0.7726389  0.7571411  0.7433941  0.6744975
```

```
### Looking at cotton:
localGloveClosestCotton <- localGloveClosest[which(rownames(localGloveClosest) == "cotto
n"), ] %>% sort(decreasing = TRUE)

# Selecting the top 5 words:
localGloveClosestCotton[1:5]
```

```
##    cotton      rice    coffee   tobacco     sugar
## 1.0000000 0.6518935 0.6429017 0.6318107 0.5973048
```

```
### Looking at child:
localGloveClosestChild <- localGloveClosest[which(rownames(localGloveClosest) == "child"
), ] %>% sort(decreasing = TRUE)

# Selecting the top 5 words:
localGloveClosestChild[1:5]
```

```
##     child  children   parents     birth    mother
## 1.0000000 0.8209387 0.7440984 0.7352427 0.7305161
```

```
### Looking at corn:
localGloveClosestCorn <- localGloveClosest[which(rownames(localGloveClosest) == "corn"),
] %>% sort(decreasing = TRUE)

# Selecting the top 5 words:
localGloveClosestCorn[1:5]
```

```
##      corn     maize     wheat      rice  potatoes
## 1.0000000 0.7643162 0.6608211 0.6493447 0.6170008
```

Using pretrained GloVe

```
#### Looking at card:
pretrainedGloveClosestCard <- pretrainedGloveClosest[which(rownames(pretrainedGloveClose
st) == "card"), ] %>% sort(decreasing = TRUE)

# Selecting the top 5 words:
pretrainedGloveClosestCard[1:5]
```

```
##          card         cards         check         picks automatically
##     1.0000000     0.8952802     0.7321196     0.7084092     0.7067509
```

```
#### Looking at computer:
pretrainedGloveClosestComputer <- pretrainedGloveClosest[which(rownames(pretrainedGloveC
losest) == "computer"), ] %>% sort(decreasing = TRUE)

# Selecting the top 5 words:
pretrainedGloveClosestComputer[1:5]
```

```
##   computer  computers   software technology electronic
##  1.0000000  0.9165045  0.8814994  0.8525559  0.8125868
```

```
### Looking at cotton:
pretrainedGloveClosestCotton <- pretrainedGloveClosest[which(rownames(pretrainedGloveClo
sest) == "cotton"), ] %>% sort(decreasing = TRUE)

# Selecting the top 5 words:
pretrainedGloveClosestCotton[1:5]
```

```
##    cotton      wool     wheat      corn     grain
## 1.0000000 0.8465082 0.8045771 0.7738308 0.7348589
```

```
### Looking at child:
pretrainedGloveClosestChild <- pretrainedGloveClosest[which(rownames(pretrainedGloveClos
est) == "child"), ] %>% sort(decreasing = TRUE)

# Selecting the top 5 words:
pretrainedGloveClosestChild[1:5]
```

```
##     child  children   parents  pregnant    mother
## 1.0000000 0.8989084 0.8658787 0.8233451 0.8211389
```

```
### Looking at corn:
pretrainedGloveClosestCorn <- pretrainedGloveClosest[which(rownames(pretrainedGloveClose
st) == "corn"), ] %>% sort(decreasing = TRUE)

# Selecting the top 5 words:
pretrainedGloveClosestCorn[1:5]
```

```
##      corn     wheat     beans     maize     sugar
## 1.0000000 0.9115993 0.8328870 0.8185488 0.8178044
```

Based on my findings, it seems like the words that are closest in the embedding space (both in the example and in the query words that I chose) are the ones that aren't necessarily synonyms, but are often part of a similar "category", or are used in similar contexts. For words that have a pretty clear, singular meaning (i.e. corn), the words that it was closest to were clearly part of one category (in this case, crops). However, for words that are typically used in multiple contexts, such as "card", the words that were closest in Euclidean distance didn't necessarily fit into one category, since cards can be used in the context of card and casino games (hence "game" and "check"), but card can also be used in reference to money (hence why it's close to "credit").

# Problem 2 Part b

Using PMI

```
# For france:paris::england:?
missingEnglandPMI <- PMIClosest[which(rownames(PMIClosest) == "paris"), ] - PMIClosest[w
hich(rownames(PMIClosest) == "france"), ] + PMIClosest[which(rownames(PMIClosest) == "en
gland"), ]


missingEnglandPMI_sorted <- missingEnglandPMI %>% sort(decreasing = TRUE)
missingEnglandPMI_sorted[1:5]
```

```
##     boston     london manchester       hall     palace
##  0.8983286  0.8937987  0.8548091  0.8529208  0.8394330
```

```
# For france:paris::germany:?
missingGermanyPMI <- PMIClosest[which(rownames(PMIClosest) == "paris"), ] - PMIClosest[w
hich(rownames(PMIClosest) == "france"), ] + PMIClosest[which(rownames(PMIClosest) == "ge
rmany"), ]


missingGermanyPMI_sorted <- missingGermanyPMI %>% sort(decreasing = TRUE)
missingGermanyPMI_sorted[1:5]
```

```
##    berlin     paris    moscow    vienna      nazi
## 1.0275896 0.9203063 0.8609722 0.8420555 0.8248016
```

```
# For queen:woman::king:?
missingKingPMI <- PMIClosest[which(rownames(PMIClosest) == "woman"), ] - PMIClosest[whic
h(rownames(PMIClosest) == "queen"), ] + PMIClosest[which(rownames(PMIClosest) == "king"
), ]


missingKingPMI_sorted <- missingKingPMI %>% sort(decreasing = TRUE)
missingKingPMI_sorted[1:5]
```

```
##    father   himself     woman       man       son
## 0.7613524 0.7561699 0.7495879 0.7310811 0.7207726
```

Using localGlove

```
# For france:paris::england:?
missingEnglandlocalGlove <- localGloveClosest[which(rownames(localGloveClosest) == "pari
s"), ] - localGloveClosest[which(rownames(localGloveClosest) == "france"), ] + localGlov
eClosest[which(rownames(localGloveClosest) == "england"), ]


missingEnglandlocalGlove_sorted <- missingEnglandlocalGlove %>% sort(decreasing = TRUE)
missingEnglandlocalGlove_sorted[1:5]
```

```
##       paris      london     england   edinburgh westminster
##   0.8226719   0.8128809   0.7869483   0.7228363   0.6860992
```

```
# For france:paris::germany:?
missingGermanylocalGlove <- localGloveClosest[which(rownames(localGloveClosest) == "pari
s"), ] - localGloveClosest[which(rownames(localGloveClosest) == "france"), ] + localGlov
eClosest[which(rownames(localGloveClosest) == "germany"), ]


missingGermanylocalGlove_sorted <- missingGermanylocalGlove %>% sort(decreasing = TRUE)
missingGermanylocalGlove_sorted[1:5]
```

```
##    berlin     paris    munich   leipzig frankfurt
## 0.8674624 0.8389730 0.7489285 0.6729073 0.6577241
```

```
# For queen:woman::king:?
missingKinglocalGlove <- localGloveClosest[which(rownames(localGloveClosest) == "woman"
), ] - localGloveClosest[which(rownames(localGloveClosest) == "queen"), ] + localGloveCl
osest[which(rownames(localGloveClosest) == "king"), ]


missingKinglocalGlove_sorted <- missingKinglocalGlove %>% sort(decreasing = TRUE)
missingKinglocalGlove_sorted[1:5]
```

```
##     woman       man      whom   himself     young
## 0.8374149 0.7933721 0.7630275 0.7395119 0.7263903
```

Using pretrained GloVe

```
# For france:paris::england:?
missingEnglandpretrainedGlove <- pretrainedGloveClosest[which(rownames(pretrainedGloveCl
osest) == "paris"), ] - pretrainedGloveClosest[which(rownames(pretrainedGloveClosest) ==
"france"), ] + pretrainedGloveClosest[which(rownames(pretrainedGloveClosest) == "englan
d"), ]


missingEnglandpretrainedGlove_sorted <- missingEnglandpretrainedGlove %>% sort(decreasin
g = TRUE)
missingEnglandpretrainedGlove_sorted[1:5]
```

```
## melbourne    london   england edinburgh   glasgow
## 0.8989643 0.8972782 0.8736394 0.8440709 0.8382895
```

```
# For france:paris::germany:?
missingGermanypretrainedGlove <- pretrainedGloveClosest[which(rownames(pretrainedGloveCl
osest) == "paris"), ] - pretrainedGloveClosest[which(rownames(pretrainedGloveClosest) ==
"france"), ] + pretrainedGloveClosest[which(rownames(pretrainedGloveClosest) == "german
y"), ]


missingGermanypretrainedGlove_sorted <- missingGermanypretrainedGlove %>% sort(decreasin
g = TRUE)
missingGermanypretrainedGlove_sorted[1:5]
```

```
##    berlin frankfurt    vienna    munich   germany
## 0.9528735 0.8491039 0.8471175 0.8439681 0.8384415
```

```
# For queen:woman::king:?
missingKingpretrainedGlove <- pretrainedGloveClosest[which(rownames(pretrainedGloveClose
st) == "woman"), ] - pretrainedGloveClosest[which(rownames(pretrainedGloveClosest) == "q
ueen"), ] + pretrainedGloveClosest[which(rownames(pretrainedGloveClosest) == "king"), ]


missingKingpretrainedGlove_sorted <- missingKingpretrainedGlove %>% sort(decreasing = TR
UE)
missingKingpretrainedGlove_sorted[1:5]
```

```
##       man       who    father     woman     death
## 0.8803014 0.8382553 0.8346130 0.8110273 0.8070209
```

My analogies will be: (1) Harvard:Yale::Dartmouth:?

```
# With PMI:
missingDartmouthPMI <- PMIClosest[which(rownames(PMIClosest) == "yale"), ] - PMIClosest
[which(rownames(PMIClosest) == "harvard"), ] + PMIClosest[which(rownames(PMIClosest) ==
"dartmouth"), ]


missingDartmouthPMI_sorted <- missingDartmouthPMI %>% sort(decreasing = TRUE)
missingDartmouthPMI_sorted[1:5]
```

```
##   gemini courtesy   ascent     lynx      tex
## 1.221807 1.220454 1.219366 1.218155 1.217720
```

```
# With local glove
missingDartmouthlocalGlove <- localGloveClosest[which(rownames(localGloveClosest) == "ya
le"), ] - localGloveClosest[which(rownames(localGloveClosest) == "harvard"), ] + localGl
oveClosest[which(rownames(localGloveClosest) == "dartmouth"), ]


missingDartmouthlocalGlove_sorted <- missingDartmouthlocalGlove %>% sort(decreasing = TR
UE)
missingDartmouthlocalGlove_sorted[1:5]
```

```
##      dartmouth rehabilitation           yale          jonah       seminary
##      0.9222706      0.7118334      0.6932949      0.6646625      0.6207895
```

```
# With pretrained glove
missingDartmouthpretrainedGlove <- pretrainedGloveClosest[which(rownames(pretrainedGlove
Closest) == "yale"), ] - pretrainedGloveClosest[which(rownames(pretrainedGloveClosest) =
= "harvard"), ] + pretrainedGloveClosest[which(rownames(pretrainedGloveClosest) == "dart
mouth"), ]


missingDartmouthpretrainedGlove_sorted <- missingDartmouthpretrainedGlove %>% sort(decre
asing = TRUE)
missingDartmouthpretrainedGlove_sorted[1:5]
```

```
## dartmouth princeton      yale    ithaca  syracuse
## 1.0669644 0.8518230 0.8289406 0.8071399 0.8015608
```

2. Good:Bad::Fast:?

```
# With PMI:
missingFastPMI <- PMIClosest[which(rownames(PMIClosest) == "bad"), ] - PMIClosest[which
(rownames(PMIClosest) == "good"), ] + PMIClosest[which(rownames(PMIClosest) == "fast"),
 ]


missingFastPMI_sorted <- missingFastPMI %>% sort(decreasing = TRUE)
missingFastPMI_sorted[1:5]
```

```
##    crash  driving   driven   stroke shooting
## 1.270165 1.252606 1.247730 1.236682 1.231067
```

```
# With local glove
missingFastlocalGlove <- localGloveClosest[which(rownames(localGloveClosest) == "bad"),
 ] - localGloveClosest[which(rownames(localGloveClosest) == "good"), ] + localGloveClose
st[which(rownames(localGloveClosest) == "fast"), ]


missingFastlocalGlove_sorted <- missingFastlocalGlove %>% sort(decreasing = TRUE)
missingFastlocalGlove_sorted[1:5]
```

```
##      fast      slow       bad     sided dangerous
## 0.9724773 0.6956018 0.6723515 0.6267641 0.6197117
```

```
# With pretrained glove
missingFastpretrainedGlove <- pretrainedGloveClosest[which(rownames(pretrainedGloveClose
st) == "bad"), ] - pretrainedGloveClosest[which(rownames(pretrainedGloveClosest) == "goo
d"), ] + pretrainedGloveClosest[which(rownames(pretrainedGloveClosest) == "fast"), ]


missingFastpretrainedGlove_sorted <- missingFastpretrainedGlove %>% sort(decreasing = TR
UE)
missingFastpretrainedGlove_sorted[1:5]
```

```
##      fast      slow       bad   trouble     worse
## 0.8812637 0.8786136 0.8163926 0.7903674 0.7777667
```

3. China:Asia::Italy:?

```
# With PMI:
missingItalyPMI <- PMIClosest[which(rownames(PMIClosest) == "asia"), ] - PMIClosest[whic
h(rownames(PMIClosest) == "china"), ] + PMIClosest[which(rownames(PMIClosest) == "italy"
), ]


missingItalyPMI_sorted <- missingItalyPMI %>% sort(decreasing = TRUE)
missingItalyPMI_sorted[1:5]
```

```
##       italy    portugal       lands netherlands        asia
##   0.9084428   0.8725906   0.8614756   0.8516800   0.8506548
```

```
# With local glove
missingItalylocalGlove <- localGloveClosest[which(rownames(localGloveClosest) == "asia"
), ] - localGloveClosest[which(rownames(localGloveClosest) == "china"), ] + localGloveCl
osest[which(rownames(localGloveClosest) == "italy"), ]


missingItalylocalGlove_sorted <- missingItalylocalGlove %>% sort(decreasing = TRUE)
missingItalylocalGlove_sorted[1:5]
```

```
##     italy    europe      asia   belgium    france
## 0.9972776 0.9011250 0.7395665 0.7331674 0.7322643
```

```
# With pretrained glove
missingItalypretrainedGlove <- pretrainedGloveClosest[which(rownames(pretrainedGloveClos
est) == "asia"), ] - pretrainedGloveClosest[which(rownames(pretrainedGloveClosest) == "c
hina"), ] + pretrainedGloveClosest[which(rownames(pretrainedGloveClosest) == "italy"), ]


missingItalypretrainedGlove_sorted <- missingItalypretrainedGlove %>% sort(decreasing =
TRUE)
missingItalypretrainedGlove_sorted[1:5]
```

```
##     italy     spain    europe  portugal   italian
## 1.0207828 0.8500086 0.8390121 0.8134104 0.8133671
```

4. Man:Doctor::Woman:?

```
# With PMI:
missingWomanPMI <- PMIClosest[which(rownames(PMIClosest) == "doctor"), ] - PMIClosest[wh
ich(rownames(PMIClosest) == "man"), ] + PMIClosest[which(rownames(PMIClosest) == "woman"
), ]


missingWomanPMI_sorted <- missingWomanPMI %>% sort(decreasing = TRUE)
missingWomanPMI_sorted[1:5]
```

```
##   doctor    girls  teacher    adult    guest
## 1.264472 1.211536 1.197492 1.190824 1.183197
```

```
# With local glove
missingWomanlocalGlove <- localGloveClosest[which(rownames(localGloveClosest) == "docto
r"), ] - localGloveClosest[which(rownames(localGloveClosest) == "man"), ] + localGloveCl
osest[which(rownames(localGloveClosest) == "woman"), ]


missingWomanlocalGlove_sorted <- missingWomanlocalGlove %>% sort(decreasing = TRUE)
missingWomanlocalGlove_sorted[1:5]
```

```
##    doctor qualified  prolific  hospital   priests
## 1.0120818 0.6340437 0.6163007 0.5880426 0.5840008
```

```
# With pretrained glove
missingWomanpretrainedGlove <- pretrainedGloveClosest[which(rownames(pretrainedGloveClos
est) == "doctor"), ] - pretrainedGloveClosest[which(rownames(pretrainedGloveClosest) ==
"man"), ] + pretrainedGloveClosest[which(rownames(pretrainedGloveClosest) == "woman"), ]


missingWomanpretrainedGlove_sorted <- missingWomanpretrainedGlove %>% sort(decreasing =
TRUE)
missingWomanpretrainedGlove_sorted[1:5]
```

```
##    doctor     nurse     child  pregnant    mother
## 1.0133156 0.9413814 0.8583411 0.8479099 0.8420102
```

5. party:fun::work:?

```
# With PMI:
missingWorkPMI <- PMIClosest[which(rownames(PMIClosest) == "fun"), ] - PMIClosest[which
(rownames(PMIClosest) == "party"), ] + PMIClosest[which(rownames(PMIClosest) == "work"),
]


missingWorkPMI_sorted <- missingWorkPMI %>% sort(decreasing = TRUE)
missingWorkPMI_sorted[1:5]
```

```
## studying  leibniz creative  publish lectures
## 1.118536 1.108776 1.093096 1.090862 1.088549
```

```
# With local glove
missingWorklocalGlove <- localGloveClosest[which(rownames(localGloveClosest) == "fun"),
 ] - localGloveClosest[which(rownames(localGloveClosest) == "party"), ] + localGloveClos
est[which(rownames(localGloveClosest) == "work"), ]


missingWorklocalGlove_sorted <- missingWorklocalGlove %>% sort(decreasing = TRUE)
missingWorklocalGlove_sorted[1:5]
```

```
##         fun        work         job   technique assumptions
##   1.3335545   1.0041070   0.9015347   0.8858900   0.8761312
```

```
# With pretrained glove
missingWorkpretrainedGlove <- pretrainedGloveClosest[which(rownames(pretrainedGloveClose
st) == "fun"), ] - pretrainedGloveClosest[which(rownames(pretrainedGloveClosest) == "par
ty"), ] + pretrainedGloveClosest[which(rownames(pretrainedGloveClosest) == "work"), ]


missingWorkpretrainedGlove_sorted <- missingWorkpretrainedGlove %>% sort(decreasing = TR
UE)
missingWorkpretrainedGlove_sorted[1:5]
```

```
##       fun wonderful     stuff     doing       lot
##  1.302527  1.261724  1.231698  1.200596  1.198335
```

My first impression is that these comparisons in 2b, which are looking at the nearest embedding vector, don't give responses that are as accurate (i.e. what a person would say) as the results from 2a, which calculates "closest" with regards to Euclidean distance. From the analogies, I see that the comparisons aren't as accurate as a person would be, but they're pretty darn good. It's impressive that the program can learn such associations just through samples of text. At the very least, the program was able to output guesses that were similar to the "correct" answer in the analogies. For example, instead of just guessing the "correct" capital city, the program gives, as its top 5 answers, 5 different capital cities, so at the very least it showed that it could understand that the "category" was capital cities. In one of my examples, I chose to compare Harvard:Yale to Dartmouth:?, seeing what the program would recommend. The program guessed that the analogy was looking at Ivy League colleges, hence why it guessed words pertaining to Ivy League schools.

However, I found great differences in the accuracy of the analogies, depending on the word. Going back to the Harvard:Yale to Dartmouth:? analogy again, the pretrained gloVe was pretty accurate in guessing that the analogy referred to Ivy League schools, but the PMI was pretty far off. From looking at the results, I would say that the GloVe embeddings were more accurate, in the sense that their responses are more like what a human would say. I can't explain why the GloVe embeddings seem to give results like this, but I do know that based off lecture in class, the GloVe embeddings are a computationally efficient heuristic that seems to give pretty accurate results (although in terms of accuracy I don't know how it "objectively" compares to PMI). However, qualitatively it does seem like the GloVe embeddings outperform the PMI embeddings, since the GloVe embeddings give responses that seem to be more in line with what a person would guess.

Another qualitative observation that I'd like to make is that the programs seem to give results that display our stereotypes (e.g. associating women with teachers or nurses), which is not the fault of the program but rather our own biases. Again, in this case the GloVe embeddings seem to be more "accurate", in that they'd give a response that a person is more likely to give, than the PMI embeddings.

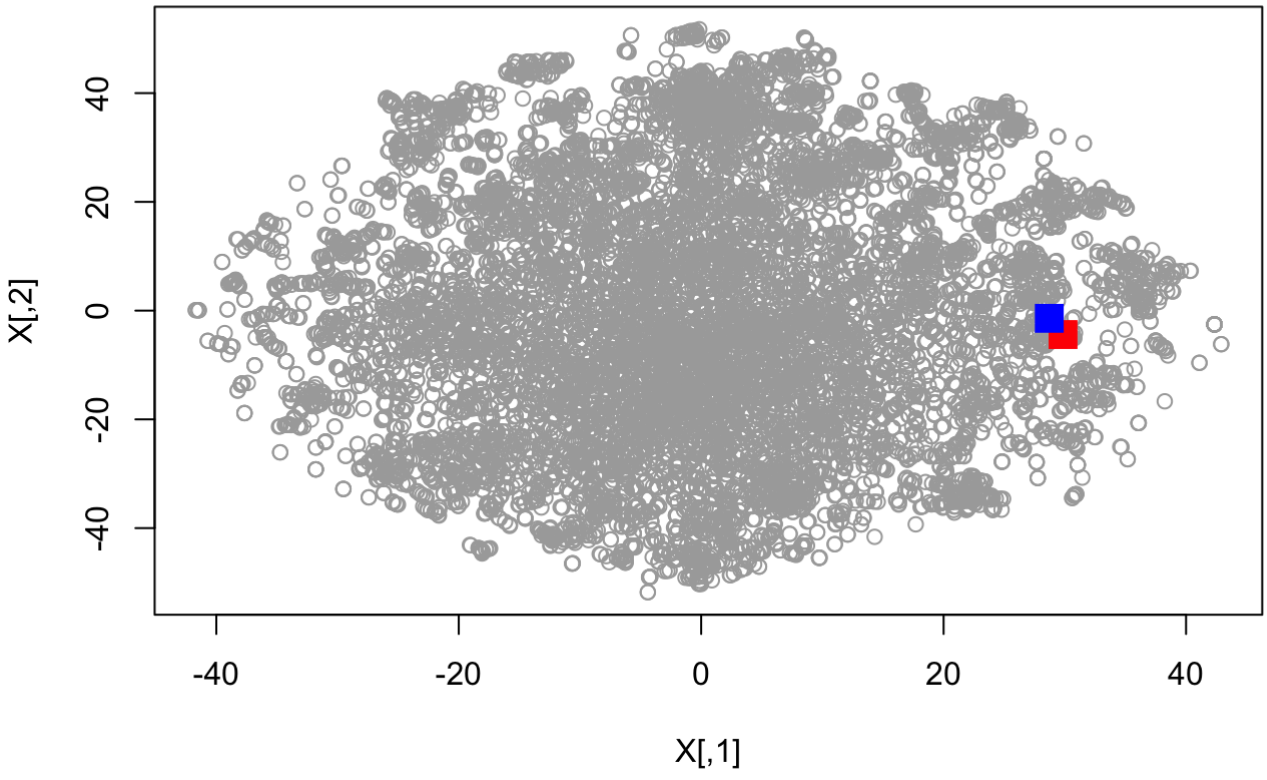
With even more training, I'm sure the models could be improved, but at the moment it seems impressive that (for the most part), the program can "guess" at least the correct category of the word in the analogy, even if it can't always guess what the answer is supposed to be.

# Problem 2 Part c

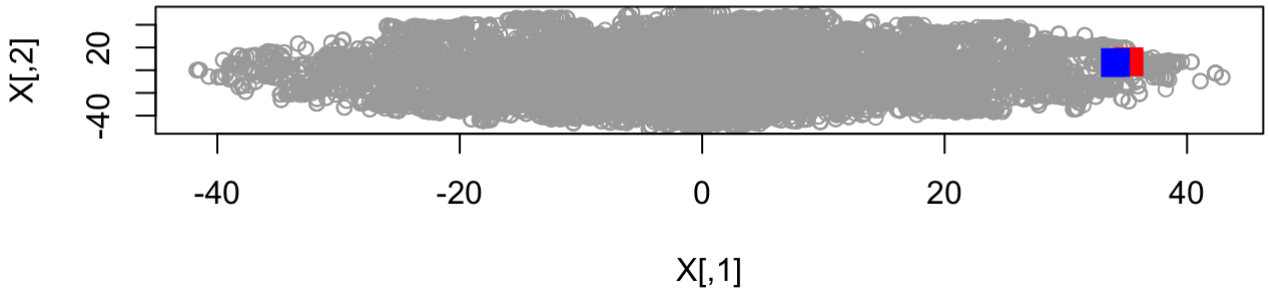
```
# Perform t-SNE method, store 2-D points in variable t.sne

library(Rtsne)
set.seed(1987)
tt <- Rtsne(pt.glove)
t.sne <- tt$Y
rownames(t.sne) <- rownames(pt.glove)
```

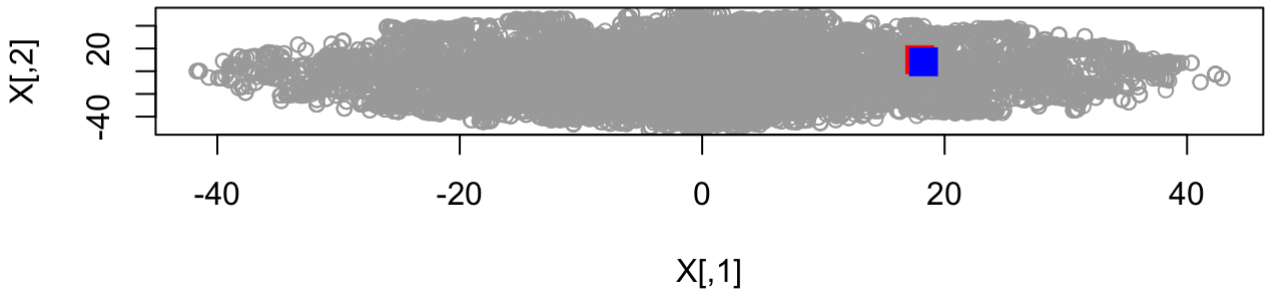
```
# Plot:

plot.pts <- function(X, w1, w2) {
  plot(X, col = "darkgrey")
  points(X[w1, 1], X[w1, 2], col = "red", pch = 15, cex = 2)
  points(X[w2, 1], X[w2, 2], col = "blue", pch = 15, cex = 2)
}
```

```
# Apply example:
plot.pts(t.sne, "democrat", "politics")
```

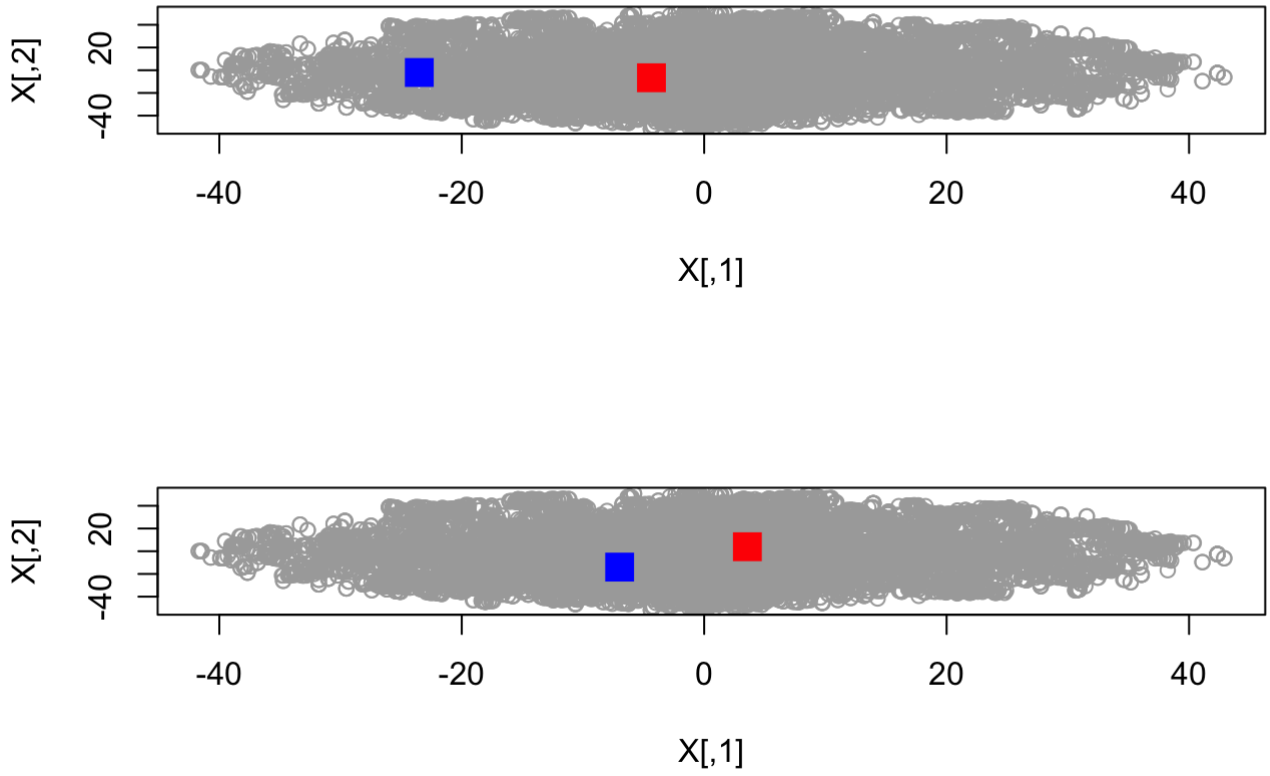



```
# Examples that produce expected results
par(mfrow = c(2, 1))
plot.pts(t.sne, "sports", "athlete")
plot.pts(t.sne, "school", "student")
```




```
# Examples that produce surprising results
par(mfrow = c(2, 1))
plot.pts(t.sne, "good", "upright")
plot.pts(t.sne, "show", "performance")
```




The similarity between good and upright isn't as close as I expected, even though they're synonyms. This is probably because, even though they're synonyms, the word "good" can be used in many contexts, while "upright" is typically used only in a moralistic context. Similarly, "show" and "performance" aren't quite as close as I expected, but this seems to be because even though there is some overlap in when they are used (e.g. "I went to see a show" vs. "I went to see a performance"), they're not necessarily interchangeable.

# Problem 3

```
# Construct artist embeddings with GloVe
library(text2vec)
playlists_file <- "/Users/mark/Documents/Spring2019Classes/S&DS365/Homework/Homework6/pl
aylists.txt"
playlists <- readLines(playlists_file, warn = FALSE)
tokensPlaylist <- space_tokenizer(playlists)
itPlaylist <- itoken(tokensPlaylist, progressbar = FALSE)
vocabPlaylist <- create_vocabulary(itPlaylist)
vocabPlaylist <- prune_vocabulary(vocabPlaylist, term_count_min = 50L)
vectorizerPlaylist <- vocab_vectorizer(vocabPlaylist)
tcmPlaylist <- create_tcm(itPlaylist, vectorizerPlaylist, skip_grams_window = 750L,
                          skip_grams_window_context = "symmetric",
                          weights = 1 / rep(1, 750L))
glovePlaylist <- GlobalVectors$new(word_vectors_size = 100,
                             vocabulary = vocabPlaylist, x_max = 10)
a.embed <- fit_transform(tcmPlaylist, glovePlaylist, n_iter = 50)
```

```
## INFO [2019-04-17 22:54:39] 2019-04-17 22:54:39 - epoch 1, expected cost 0.6386
## INFO [2019-04-17 22:54:39] 2019-04-17 22:54:39 - epoch 2, expected cost 0.1793
## INFO [2019-04-17 22:54:40] 2019-04-17 22:54:40 - epoch 3, expected cost 0.0832
## INFO [2019-04-17 22:54:40] 2019-04-17 22:54:40 - epoch 4, expected cost 0.0578
## INFO [2019-04-17 22:54:40] 2019-04-17 22:54:40 - epoch 5, expected cost 0.0450
## INFO [2019-04-17 22:54:40] 2019-04-17 22:54:40 - epoch 6, expected cost 0.0372
## INFO [2019-04-17 22:54:40] 2019-04-17 22:54:40 - epoch 7, expected cost 0.0319
## INFO [2019-04-17 22:54:40] 2019-04-17 22:54:40 - epoch 8, expected cost 0.0282
## INFO [2019-04-17 22:54:40] 2019-04-17 22:54:40 - epoch 9, expected cost 0.0253
## INFO [2019-04-17 22:54:40] 2019-04-17 22:54:40 - epoch 10, expected cost 0.0231
## INFO [2019-04-17 22:54:41] 2019-04-17 22:54:41 - epoch 11, expected cost 0.0213
## INFO [2019-04-17 22:54:41] 2019-04-17 22:54:41 - epoch 12, expected cost 0.0198
## INFO [2019-04-17 22:54:41] 2019-04-17 22:54:41 - epoch 13, expected cost 0.0185
## INFO [2019-04-17 22:54:41] 2019-04-17 22:54:41 - epoch 14, expected cost 0.0175
## INFO [2019-04-17 22:54:41] 2019-04-17 22:54:41 - epoch 15, expected cost 0.0165
## INFO [2019-04-17 22:54:41] 2019-04-17 22:54:41 - epoch 16, expected cost 0.0157
## INFO [2019-04-17 22:54:41] 2019-04-17 22:54:41 - epoch 17, expected cost 0.0150
## INFO [2019-04-17 22:54:41] 2019-04-17 22:54:41 - epoch 18, expected cost 0.0144
## INFO [2019-04-17 22:54:42] 2019-04-17 22:54:42 - epoch 19, expected cost 0.0138
## INFO [2019-04-17 22:54:42] 2019-04-17 22:54:42 - epoch 20, expected cost 0.0133
## INFO [2019-04-17 22:54:42] 2019-04-17 22:54:42 - epoch 21, expected cost 0.0128
## INFO [2019-04-17 22:54:42] 2019-04-17 22:54:42 - epoch 22, expected cost 0.0124
## INFO [2019-04-17 22:54:42] 2019-04-17 22:54:42 - epoch 23, expected cost 0.0120
## INFO [2019-04-17 22:54:42] 2019-04-17 22:54:42 - epoch 24, expected cost 0.0116
## INFO [2019-04-17 22:54:42] 2019-04-17 22:54:42 - epoch 25, expected cost 0.0113
## INFO [2019-04-17 22:54:42] 2019-04-17 22:54:42 - epoch 26, expected cost 0.0110
## INFO [2019-04-17 22:54:42] 2019-04-17 22:54:42 - epoch 27, expected cost 0.0107
## INFO [2019-04-17 22:54:43] 2019-04-17 22:54:43 - epoch 28, expected cost 0.0104
## INFO [2019-04-17 22:54:43] 2019-04-17 22:54:43 - epoch 29, expected cost 0.0102
## INFO [2019-04-17 22:54:43] 2019-04-17 22:54:43 - epoch 30, expected cost 0.0099
## INFO [2019-04-17 22:54:43] 2019-04-17 22:54:43 - epoch 31, expected cost 0.0097
## INFO [2019-04-17 22:54:43] 2019-04-17 22:54:43 - epoch 32, expected cost 0.0095
## INFO [2019-04-17 22:54:43] 2019-04-17 22:54:43 - epoch 33, expected cost 0.0093
## INFO [2019-04-17 22:54:43] 2019-04-17 22:54:43 - epoch 34, expected cost 0.0091
## INFO [2019-04-17 22:54:44] 2019-04-17 22:54:44 - epoch 35, expected cost 0.0090
## INFO [2019-04-17 22:54:44] 2019-04-17 22:54:44 - epoch 36, expected cost 0.0088
## INFO [2019-04-17 22:54:44] 2019-04-17 22:54:44 - epoch 37, expected cost 0.0086
## INFO [2019-04-17 22:54:44] 2019-04-17 22:54:44 - epoch 38, expected cost 0.0085
## INFO [2019-04-17 22:54:44] 2019-04-17 22:54:44 - epoch 39, expected cost 0.0083
## INFO [2019-04-17 22:54:44] 2019-04-17 22:54:44 - epoch 40, expected cost 0.0082
## INFO [2019-04-17 22:54:44] 2019-04-17 22:54:44 - epoch 41, expected cost 0.0081
## INFO [2019-04-17 22:54:44] 2019-04-17 22:54:44 - epoch 42, expected cost 0.0080
## INFO [2019-04-17 22:54:45] 2019-04-17 22:54:45 - epoch 43, expected cost 0.0078
## INFO [2019-04-17 22:54:45] 2019-04-17 22:54:45 - epoch 44, expected cost 0.0077
## INFO [2019-04-17 22:54:45] 2019-04-17 22:54:45 - epoch 45, expected cost 0.0076
## INFO [2019-04-17 22:54:45] 2019-04-17 22:54:45 - epoch 46, expected cost 0.0075
## INFO [2019-04-17 22:54:45] 2019-04-17 22:54:45 - epoch 47, expected cost 0.0074
## INFO [2019-04-17 22:54:45] 2019-04-17 22:54:45 - epoch 48, expected cost 0.0073
## INFO [2019-04-17 22:54:45] 2019-04-17 22:54:45 - epoch 49, expected cost 0.0072
## INFO [2019-04-17 22:54:45] 2019-04-17 22:54:45 - epoch 50, expected cost 0.0071
```

```
# change rownames of a.embed to artist names

artists_file <- "/Users/mark/Documents/Spring2019Classes/S&DS365/Homework/Homework6/artists.txt"
artist.hash <- readLines(artists_file, warn = FALSE)
rownames(a.embed) <- artist.hash[as.numeric(rownames(a.embed)) + 1]
```

# Problem 3 Part a

```
artistGloveClosest <- sim2(a.embed)

#### Looking at The Beatles:
artistGloveClosestBeatles <- artistGloveClosest[which(rownames(artistGloveClosest) == "The Beatles"), ] %>% sort(decreasing = TRUE)

# Selecting the top 5 words:
artistGloveClosestBeatles[1:5]
```

```
##                  The Beatles        The Steve Miller Band
##                    1.0000000                    0.5827470
## Creedence Clearwater Revival The Electric Light Orchestra
##                    0.5307554                    0.5088066
##          Grand Funk Railroad
##                    0.4984370
```

```
#### Looking at Lady Gaga:
artistGloveClosestLadyGaga <- artistGloveClosest[which(rownames(artistGloveClosest) == "Lady Gaga"), ] %>% sort(decreasing = TRUE)

# Selecting the top 5 words:
artistGloveClosestLadyGaga[1:5]
```

```
##       Lady Gaga      Katy Perry      Bruno Mars            Pink
##       1.0000000       0.7761772       0.7610474       0.6604128
## Black Eyed Peas
##       0.6486953
```

```
### Looking at Nirvana:
artistGloveClosestNirvana <- artistGloveClosest[which(rownames(artistGloveClosest) == "Nirvana"), ] %>% sort(decreasing = TRUE)

# Selecting the top 5 words:
artistGloveClosestNirvana[1:5]
```

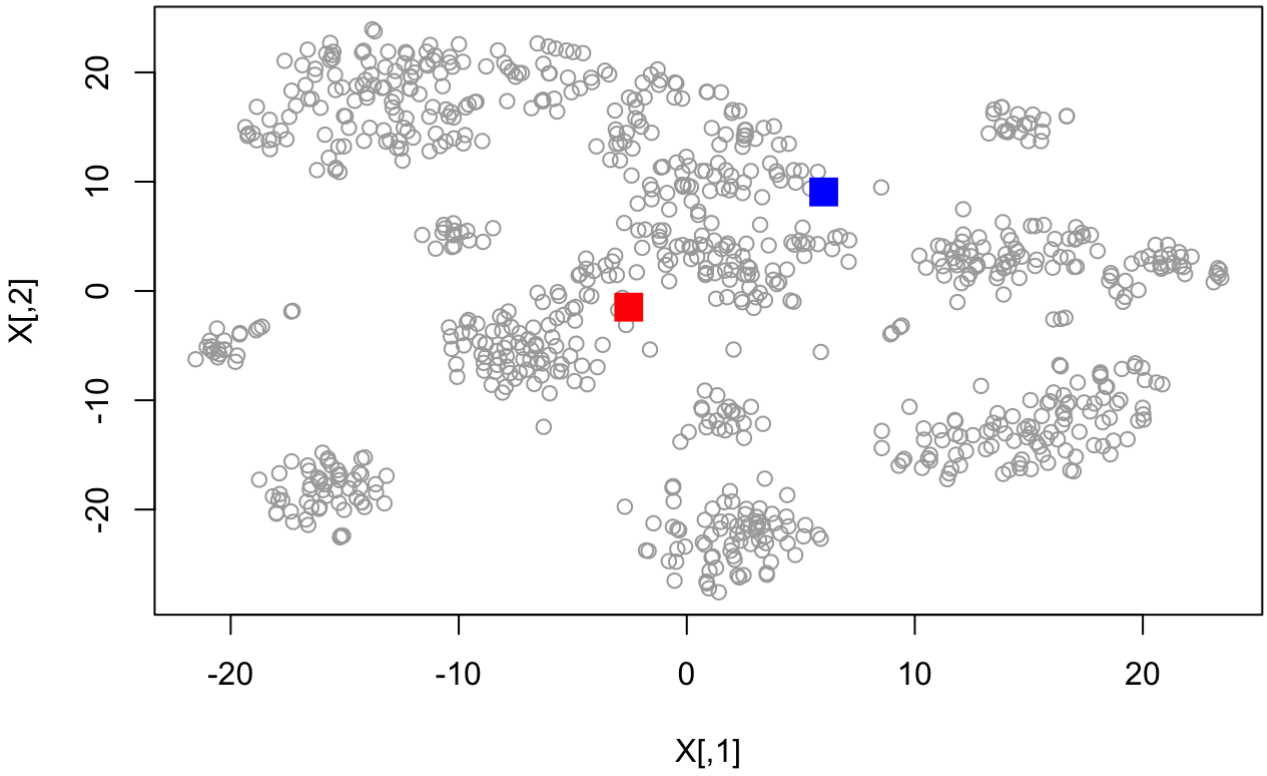
```
##                   Nirvana           Alice In Chains
##                 1.0000000                 0.8647947
##                 Metallica       Stone Temple Pilots
##                 0.7933414                 0.7908844
## The Red Hot Chili Peppers
##                 0.7833540
```

It's interesting, because the 5 closest artist embeddings for each of the artists are pretty much who you'd expect: they're contemporaries who are in the same genre as that singer. For example, in the case of The Beatles, you see older bands whose style were similar to that of The Beatles (I guess? I don't know much about them, but from what I know, then yes). The same goes for Lady Gaga (where we see pop artists who are contemporaries with Lady Gaga) and Nirvana (where we see bands whose genres are similar to Nirvana, I think?).

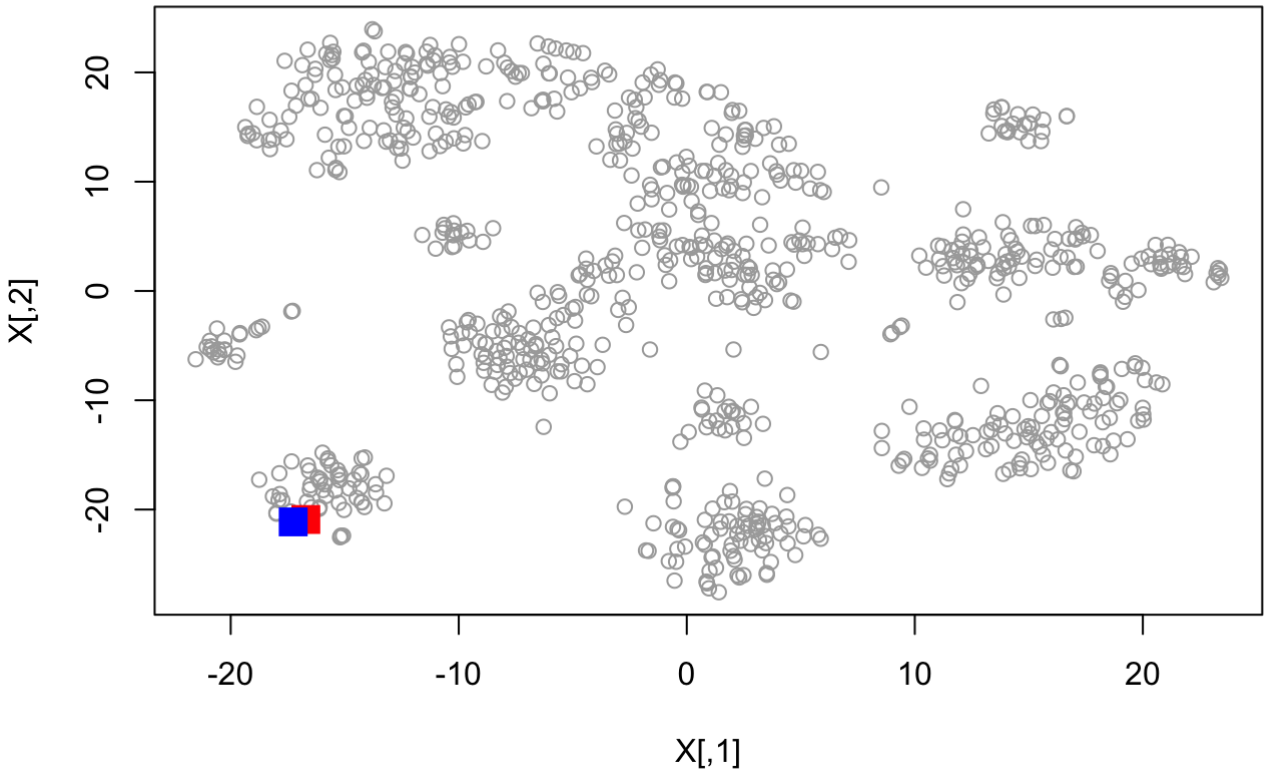
# Problem 3 Part b

```
# Set up t-SNE
set.seed(1987)
tt_artist <- Rtsne(a.embed)
t.sne_artist <- tt_artist$Y
rownames(t.sne_artist) <- rownames(a.embed)
```

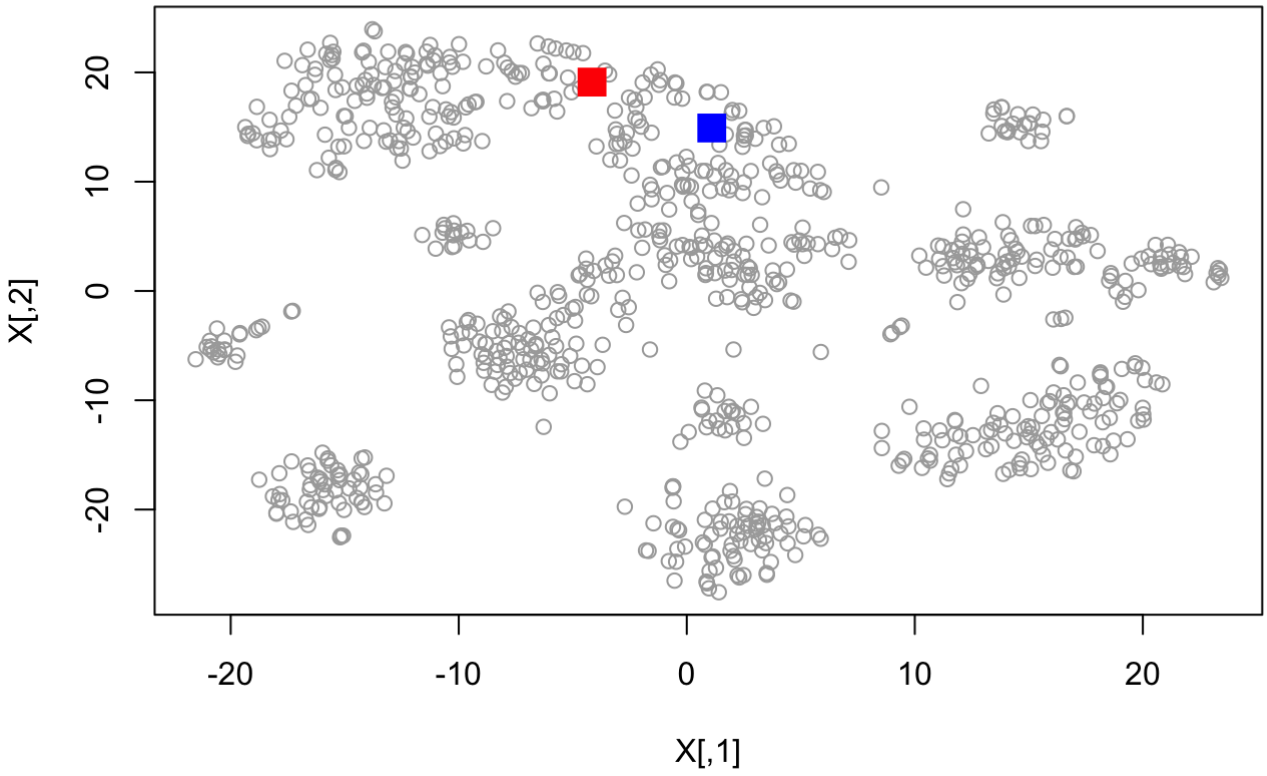
```
# Use t-SNE:

# Example:
plot.pts(t.sne_artist, "The Temptations", "The Supremes")
```



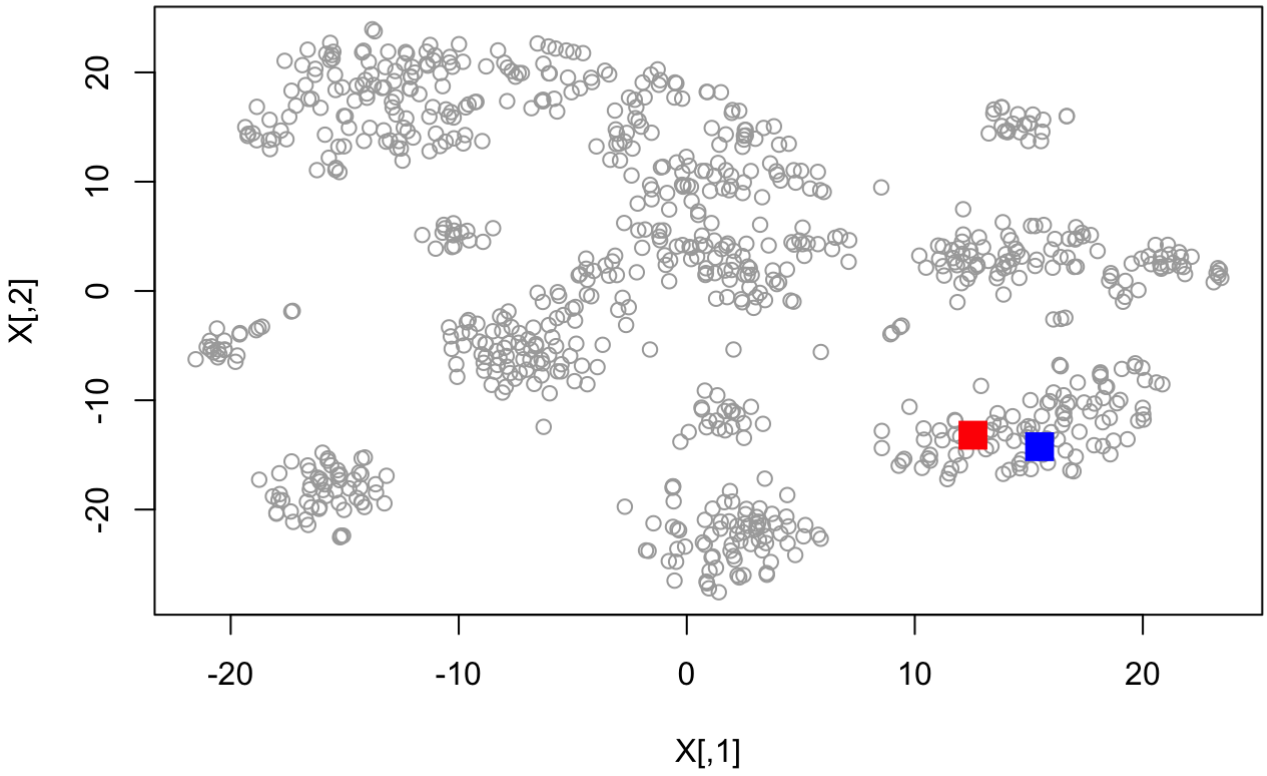

```
# Some of the ones that I thought were cool (first artist is red, second is blue):

# Christian music
plot.pts(t.sne_artist, "Chris Tomlin", "MercyMe")
```



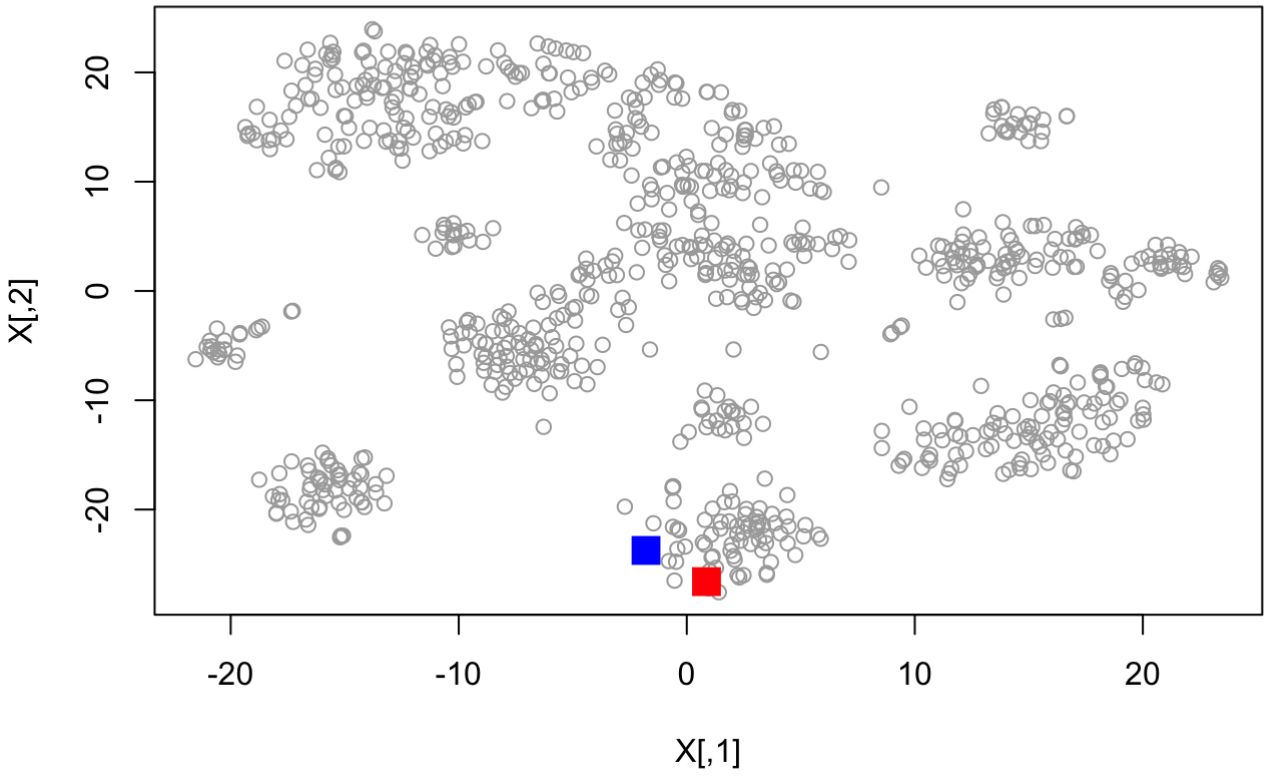

```
# Bands
plot.pts(t.sne_artist, "Queen", "The Beatles")
```



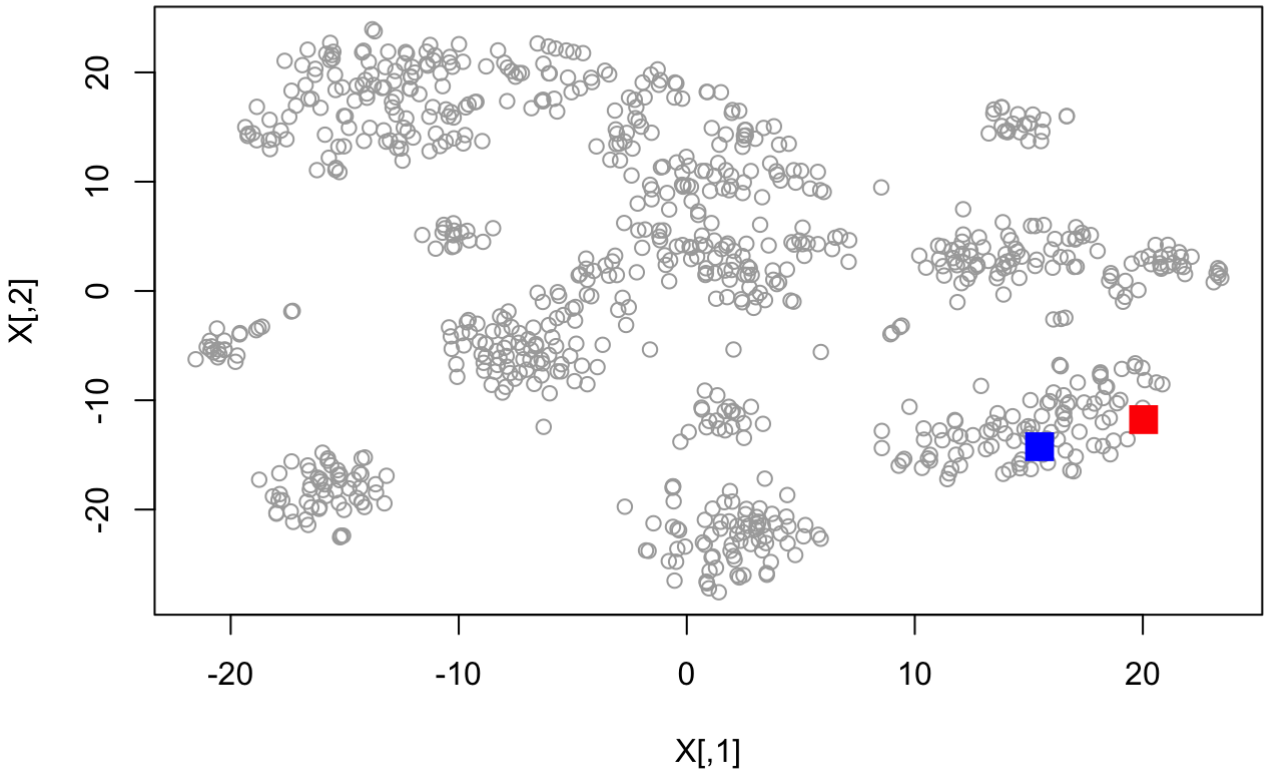

```
# Rap
plot.pts(t.sne_artist, "Ludacris", "Kanye West")
```



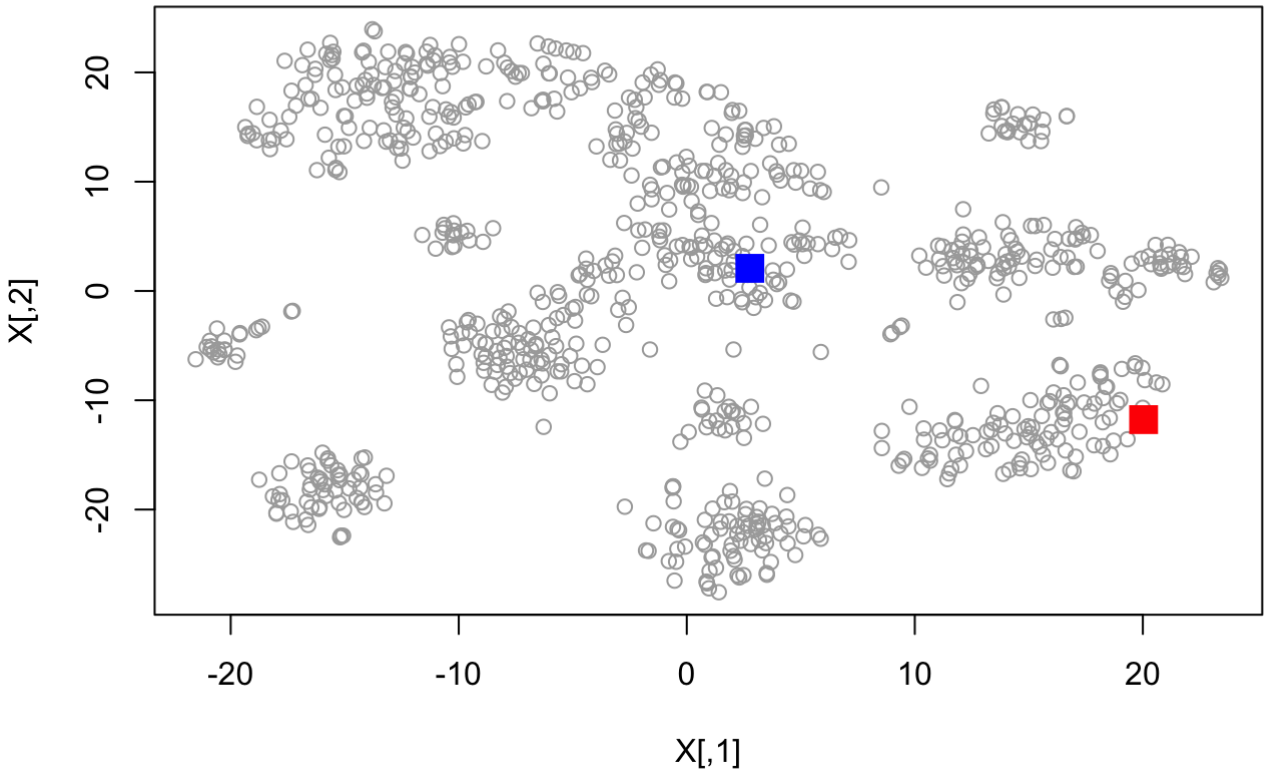

```
# Country
plot.pts(t.sne_artist, "Keith Urban", "Taylor Swift")
```



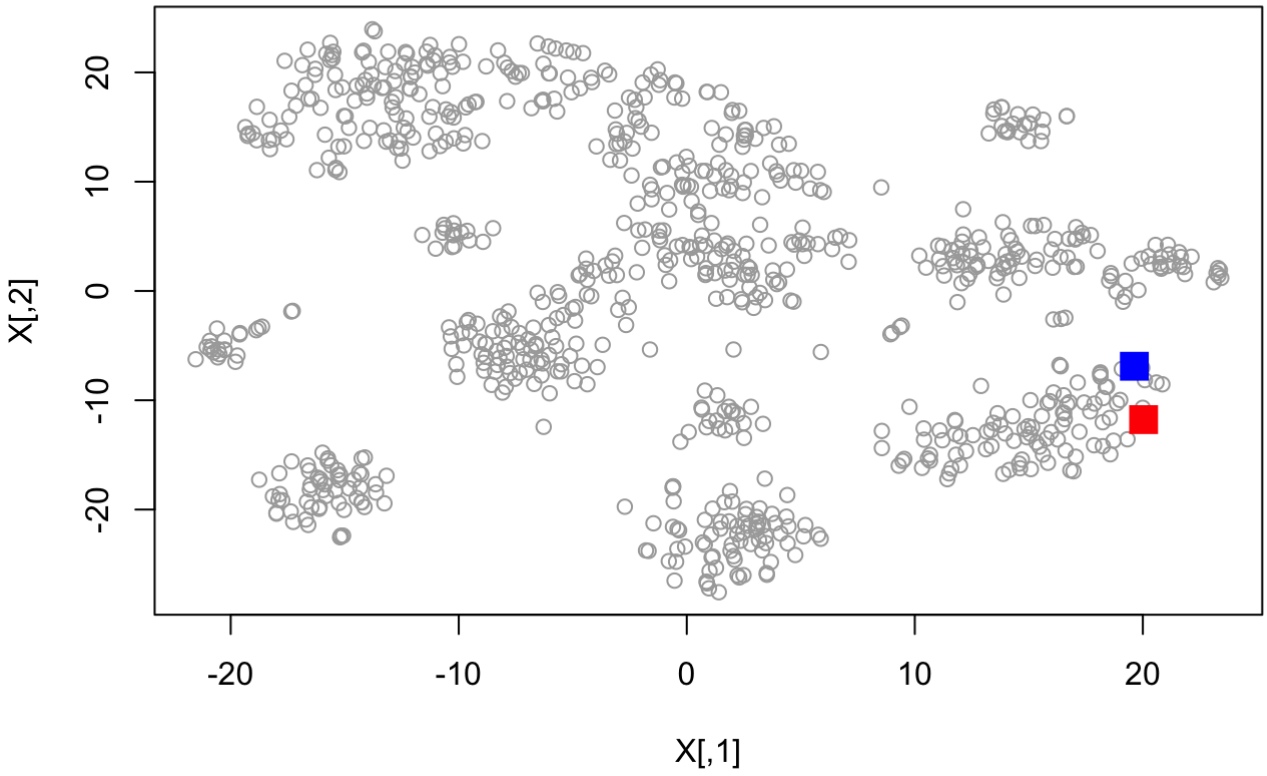

```
#---------

# What genre is Chris Brown? Is he rap? Pop? R&B? Let's see:

# Rap
plot.pts(t.sne_artist, "Chris Brown", "Kanye West")
```



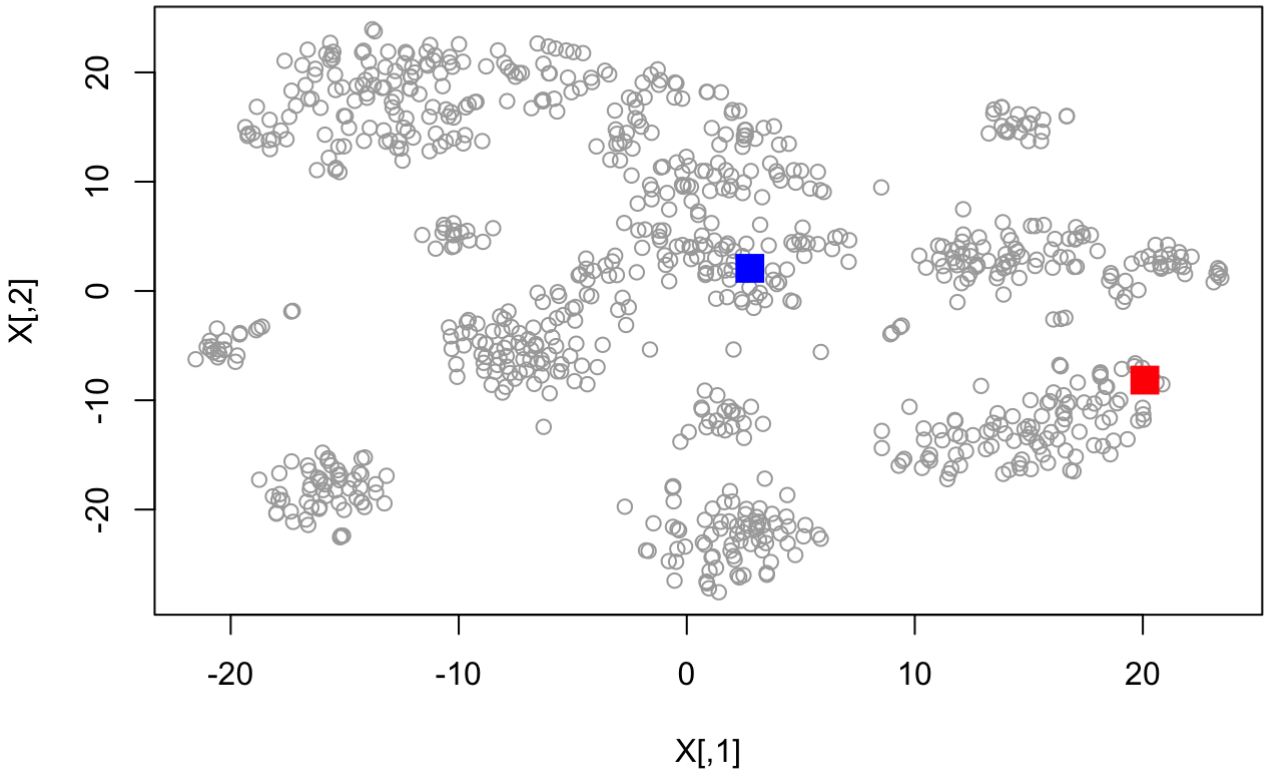

```
# R&B
plot.pts(t.sne_artist, "Chris Brown", "Fergie")
```



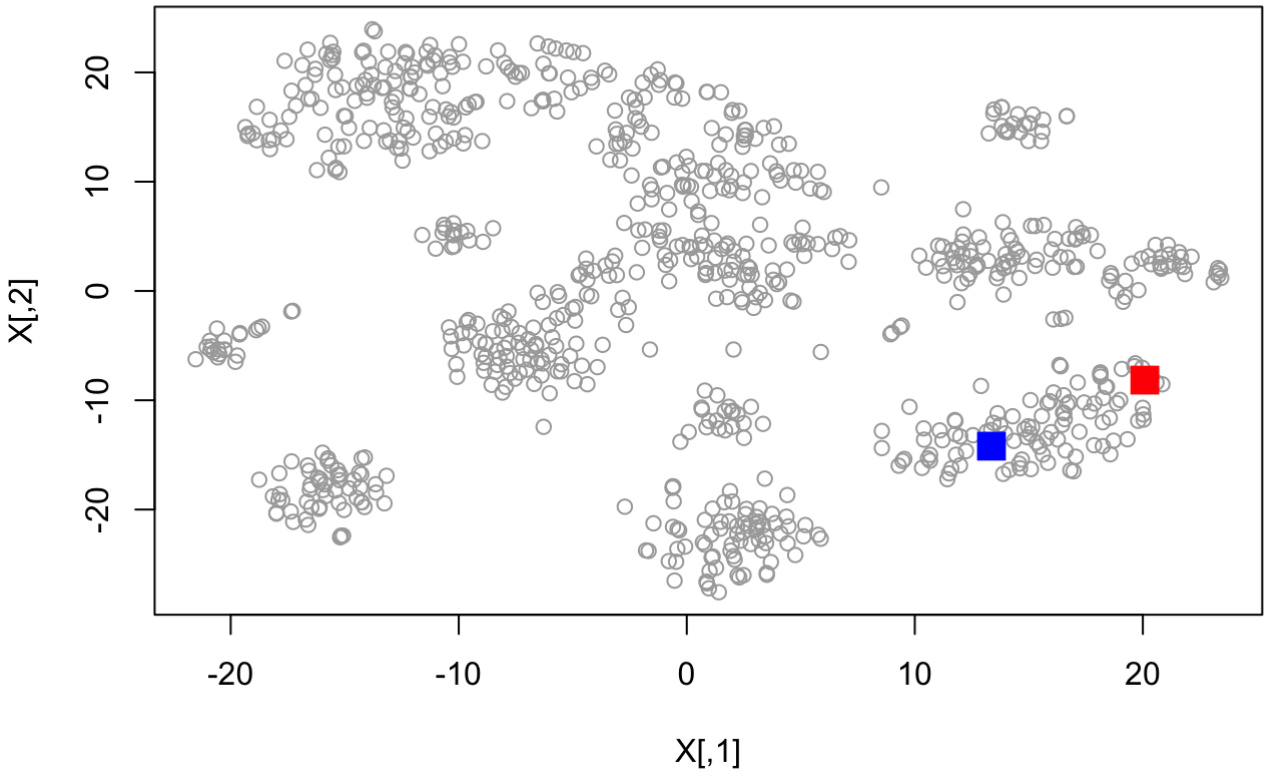

```
# Pop
plot.pts(t.sne_artist, "Chris Brown", "Lady Gaga")
```



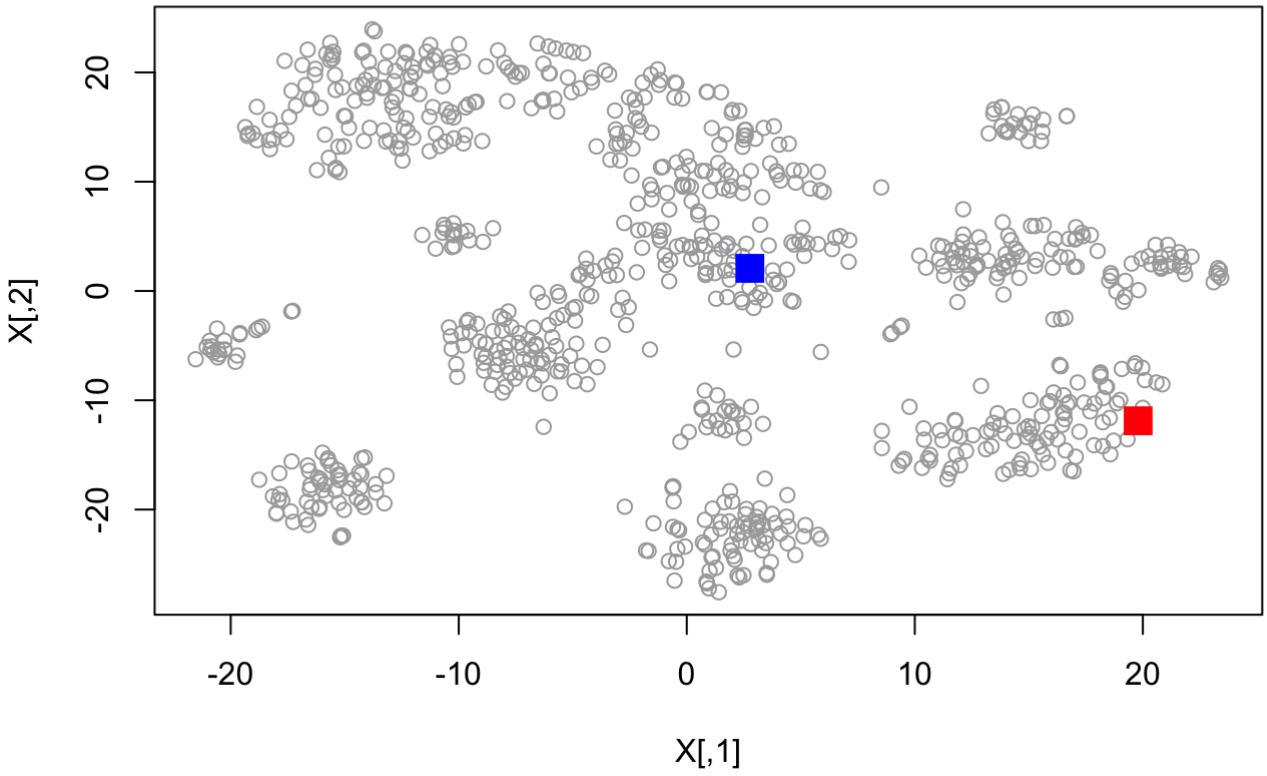

```
#------------
# Comparing Rihanna to Fergie. I think of them as similar, and I think of Rihanna as mor
e pop.
# Does that turn out to be true?
plot.pts(t.sne_artist, "Rihanna", "Fergie") # Not R&B?
```



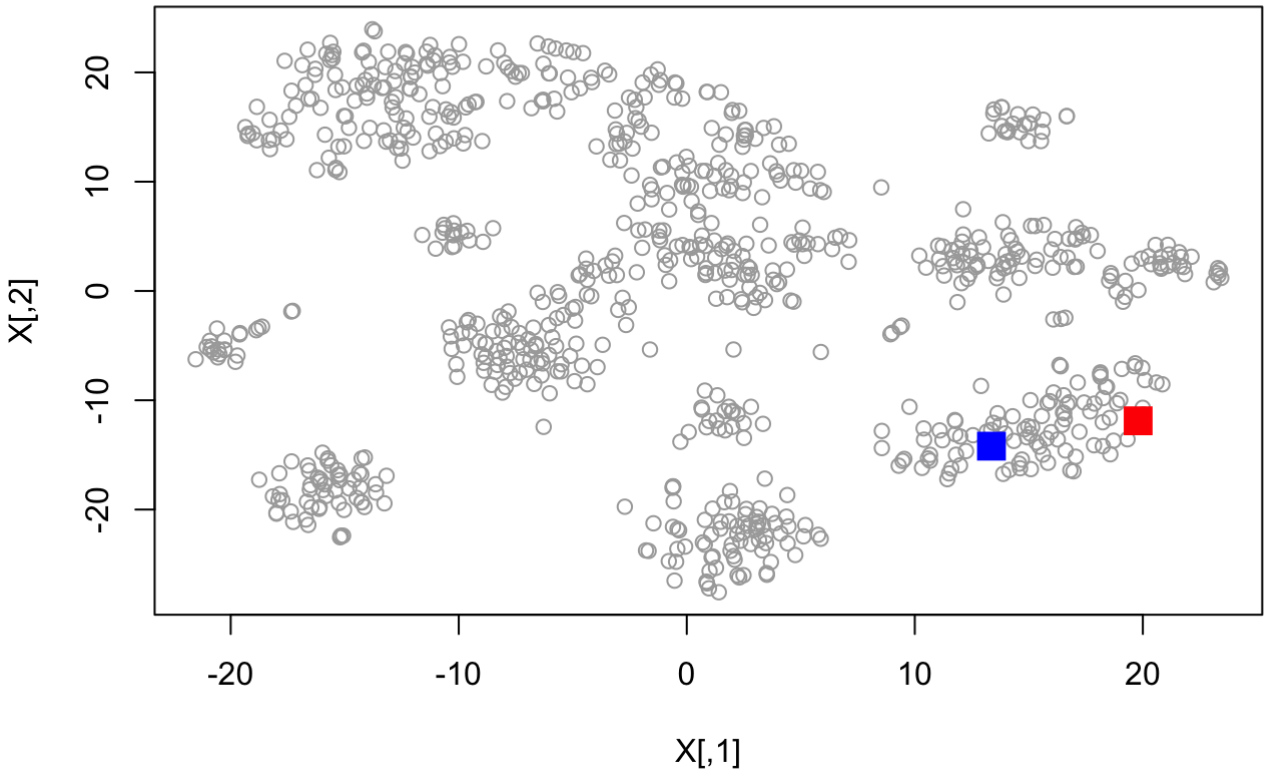

```
plot.pts(t.sne_artist, "Rihanna", "Rick Ross") # Huh, look at that
```




```
#-----------
# Settling a debate: what is Drake's deal? Is he a rapper or a singer
plot.pts(t.sne_artist, "Drake", "Fergie") # Not a singer
```




```
plot.pts(t.sne_artist, "Drake", "Rick Ross") # Yes a rapper
```




The plots seem to associate singers who work in similar “genres”, whether it be pop or rap or Christian music. It’s interesting because from looking at the plots, the clusters do seem to be defined by the genres of music that the performers are within. I was interested to see if some of the clusters would be because of differences in solo vs. band groups, but that didn’t seem to be a factor. I could clearly see the clusters of different genres, such as Christian music in the bottom, country music to the far right, and rap in the right as well. I was interested in studying Chris Brown and Drake, because they both seem to get their start in rap, but seem to dabble in pop. From these plots, it seems like they’re much closer to other rappers than they are to other pop singers (and they are clearly within the “rap” cluster). I also wanted to see how Rihanna would fare - I expected her to be a blend of R&B and pop, but I was very surprised to see that she was clearly in the “rap cluster”.