Assignment3-Part2

Mengtong Zhang

4/3/2020

Question 3

```
library(wordVectors)
library(Rtsne)
library(tidytext)
library(tidyverse)
```

```
if (!file.exists("cookbooks.zip")) {
  download.file("http://archive.lib.msu.edu/dinfo/feedingamerica/cookbook_text.zip","coo
kbooks.zip")
}
unzip("cookbooks.zip",exdir="cookbooks")
if (!file.exists("cookbooks.txt")) prep_word2vec(origin="cookbooks",destination="cookboo
ks.txt",lowercase=T,bundle_ngrams=1)

# Training a Word2Vec model
if (!file.exists("cookbook_vectors.bin")) {
  model = train_word2vec("cookbooks.txt","cookbook_vectors.bin",
                         vectors=100,threads=4,window=6,
                         min_count = 10,
                         iter=5,negative_samples=15)
} else{
    model = read.vectors("cookbook_vectors.bin")
    }
```

```
##
  |
  |                                                                      |   0%
  |
  |                                                                      |   1%
  |
  |=                                                                     |   1%
  |
  |=                                                                     |   2%
  |
  |==                                                                    |   2%
  |
  |==                                                                    |   3%
  |
  |==                                                                    |   4%
  |
  |===                                                                   |   4%
  |
  |===                                                                   |   5%
  |
  |====                                                                  |   5%
  |
  |====                                                                  |   6%
  |
  |====                                                                  |   7%
  |
  |=====                                                                 |   7%
  |
  |=====                                                                 |   8%
  |
  |======                                                                |   8%
  |
  |======                                                                |   9%
  |
  |======                                                                |  10%
  |
  |=======                                                               |  10%
  |
  |=======                                                               |  11%
  |
  |=======                                                               |  12%
  |
  |========                                                              |  12%
  |
  |========                                                              |  13%
  |
  |=========                                                             |  13%
  |
  |=========                                                             |  14%
  |
  |=========                                                             |  15%
  |
  |==========                                                            |  15%
```

```
|
|==========                                                            |  16%
|
|===========                                                           |  16%
|
|===========                                                           |  17%
|
|===========                                                           |  18%
|
|============                                                          |  18%
|
|============                                                          |  19%
|
|=============                                                         |  19%
|
|=============                                                         |  20%
|
|=============                                                         |  21%
|
|==============                                                        |  21%
|
|==============                                                        |  22%
|
|===============                                                       |  22%
|
|===============                                                       |  23%
|
|===============                                                       |  24%
|
|================                                                      |  24%
|
|================                                                      |  25%
|
|=================                                                     |  25%
|
|=================                                                     |  26%
|
|=================                                                     |  27%
|
|==================                                                    |  27%
|
|==================                                                    |  28%
|
|===================                                                   |  28%
|
|===================                                                   |  29%
|
|===================                                                   |  30%
|
|====================                                                  |  30%
|
|====================                                                  |  31%
|
|====================                                                  |  32%
```

```
|
|=====================                                                 |  32%
|
|=====================                                                 |  33%
|
|======================                                                |  33%
|
|======================                                                |  34%
|
|======================                                                |  35%
|
|=======================                                               |  35%
|
|=======================                                               |  36%
|
|========================                                              |  36%
|
|========================                                              |  37%
|
|========================                                              |  38%
|
|=========================                                             |  38%
|
|=========================                                             |  39%
|
|==========================                                            |  39%
|
|==========================                                            |  40%
|
|==========================                                            |  41%
|
|===========================                                           |  41%
|
|===========================                                           |  42%
|
|============================                                          |  42%
|
|============================                                          |  43%
|
|============================                                          |  44%
|
|=============================                                         |  44%
|
|=============================                                         |  45%
|
|==============================                                        |  45%
|
|==============================                                        |  46%
|
|==============================                                        |  47%
|
|===============================                                       |  47%
|
|===============================                                       |  48%
```

```
|
|================================                                |  48%
|
|================================                                |  49%
|
|================================                                |  50%
|
|=================================                               |  50%
|
|=================================                               |  51%
|
|=================================                               |  52%
|
|==================================                              |  52%
|
|==================================                              |  53%
|
|===================================                             |  53%
|
|===================================                             |  54%
|
|===================================                             |  55%
|
|====================================                            |  55%
|
|====================================                            |  56%
|
|=====================================                           |  56%
|
|=====================================                           |  57%
|
|=====================================                           |  58%
|
|======================================                          |  58%
|
|======================================                          |  59%
|
|=======================================                         |  59%
|
|=======================================                         |  60%
|
|=======================================                         |  61%
|
|========================================                        |  61%
|
|========================================                        |  62%
|
|=========================================                       |  62%
|
|=========================================                       |  63%
|
|=========================================                       |  64%
|
|==========================================                      |  64%
```

```
  |
  |==========================================                        |  65%
  |
  |===========================================                       |  65%
  |
  |===========================================                       |  66%
  |
  |===========================================                       |  67%
  |
  |============================================                      |  67%
  |
  |============================================                      |  68%
  |
  |=============================================                     |  68%
  |
  |=============================================                     |  69%
  |
  |=============================================                     |  70%
  |
  |==============================================                    |  70%
  |
  |==============================================                    |  71%
  |
  |==============================================                    |  72%
  |
  |===============================================                   |  72%
  |
  |===============================================                   |  73%
  |
  |================================================                  |  73%
  |
  |================================================                  |  74%
  |
  |================================================                  |  75%
  |
  |=================================================                 |  75%
  |
  |=================================================                 |  76%
  |
  |==================================================                |  76%
  |
  |==================================================                |  77%
  |
  |==================================================                |  78%
  |
  |===================================================               |  78%
  |
  |===================================================               |  79%
  |
  |====================================================              |  79%
  |
  |====================================================              |  80%
  |
  |====================================================              |  81%
```

```
  |
  |=====================================================            |  81%
  |
  |=====================================================            |  82%
  |
  |======================================================           |  82%
  |
  |======================================================           |  83%
  |
  |======================================================           |  84%
  |
  |=======================================================          |  84%
  |
  |=======================================================          |  85%
  |
  |========================================================         |  85%
  |
  |========================================================         |  86%
  |
  |========================================================         |  87%
  |
  |=========================================================        |  87%
  |
  |=========================================================        |  88%
  |
  |==========================================================       |  88%
  |
  |==========================================================       |  89%
  |
  |==========================================================       |  90%
  |
  |===========================================================      |  90%
  |
  |===========================================================      |  91%
  |
  |===========================================================      |  92%
  |
  |============================================================     |  92%
  |
  |============================================================     |  93%
  |
  |=============================================================    |  93%
  |
  |=============================================================    |  94%
  |
  |=============================================================    |  95%
  |
  |==============================================================   |  95%
  |
  |==============================================================   |  96%
  |
  |===============================================================  |  96%
  |
  |===============================================================  |  97%
```

```
  |
  |=================================================================  |  98%
  |
  |================================================================== |  98%
  |
  |================================================================== |  99%
  |
  |===================================================================|  99%
  |
  |===================================================================| 100%
```

1.
To improve the quality of embedding, I would do stop words removal, stemming(lemmatization) and text normalization.

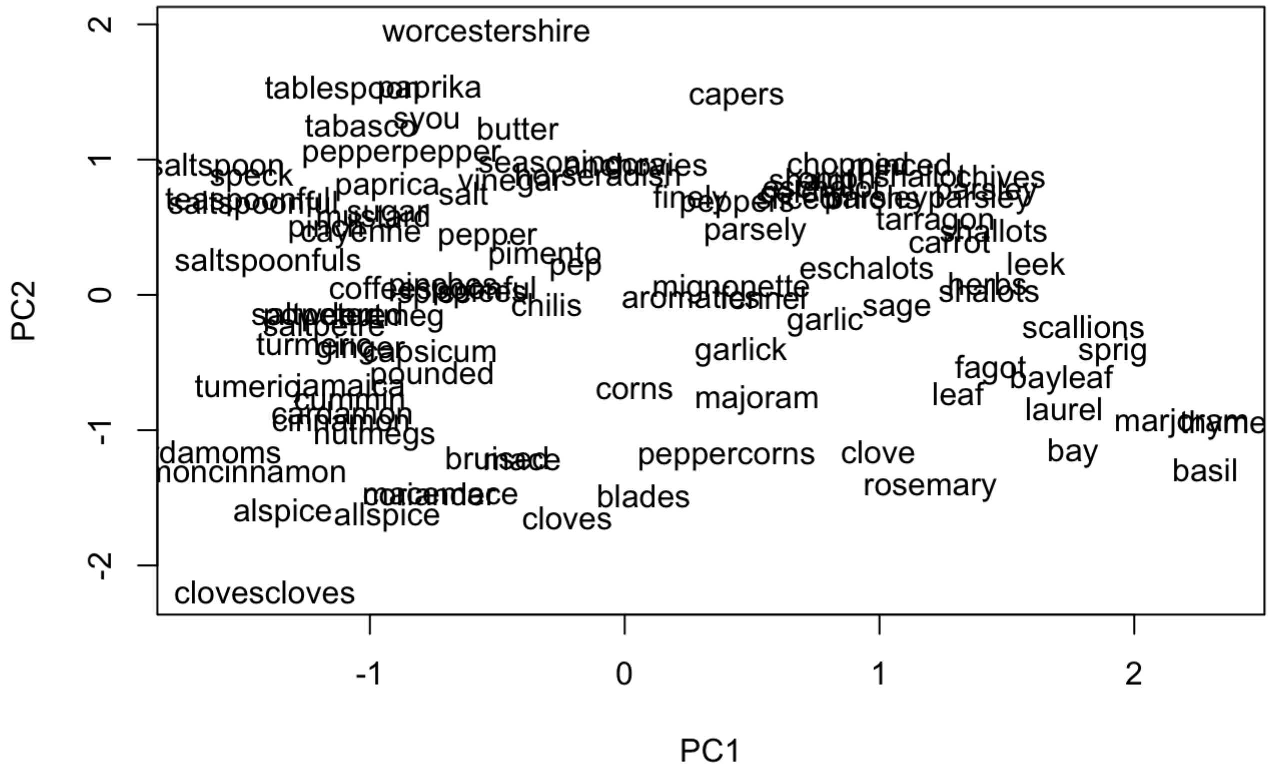
2.
I choose ‘sugar’,‘pepper’ and ‘garlic’ as three ingresients. And pepper, garlic, mace, cayenne and cloves are top 5 similar ingredients.

```
# -- Select ingredient and cuisine --
ingredient = 'sugar'
ingredient_2 = 'pepper'
ingredient_3 = 'garlic'
list_of_ingredients = c(ingredient, ingredient_2, ingredient_3)
# Set of closest words to "sage", "thyme","basil"
model %>% closest_to(model[[list_of_ingredients]],5)
```

```
##      word similarity to model[[list_of_ingredients]]
## 1  pepper                                  0.8521250
## 2  garlic                                  0.8285670
## 3    mace                                  0.7766267
## 4 cayenne                                  0.7758408
## 5  cloves                                  0.7603776
```

3.
Here we use t-SNE to see the relationships between set of words related with the three ingredients in question2.

```
n_words = 100
closest_ingredients = closest_to(model,model[[list_of_ingredients]], n_words)$word
surrounding_ingredients = model[[closest_ingredients,average=F]]
plot(surrounding_ingredients,method="pca")
```




```
embedding = Rtsne(X = surrounding_ingredients, dims = 2,
                  perplexity = 4,
                  theta = 0.5,
                  eta = 10,
                  pca = TRUE, verbose = TRUE,
                  max_iter = 2000)
```

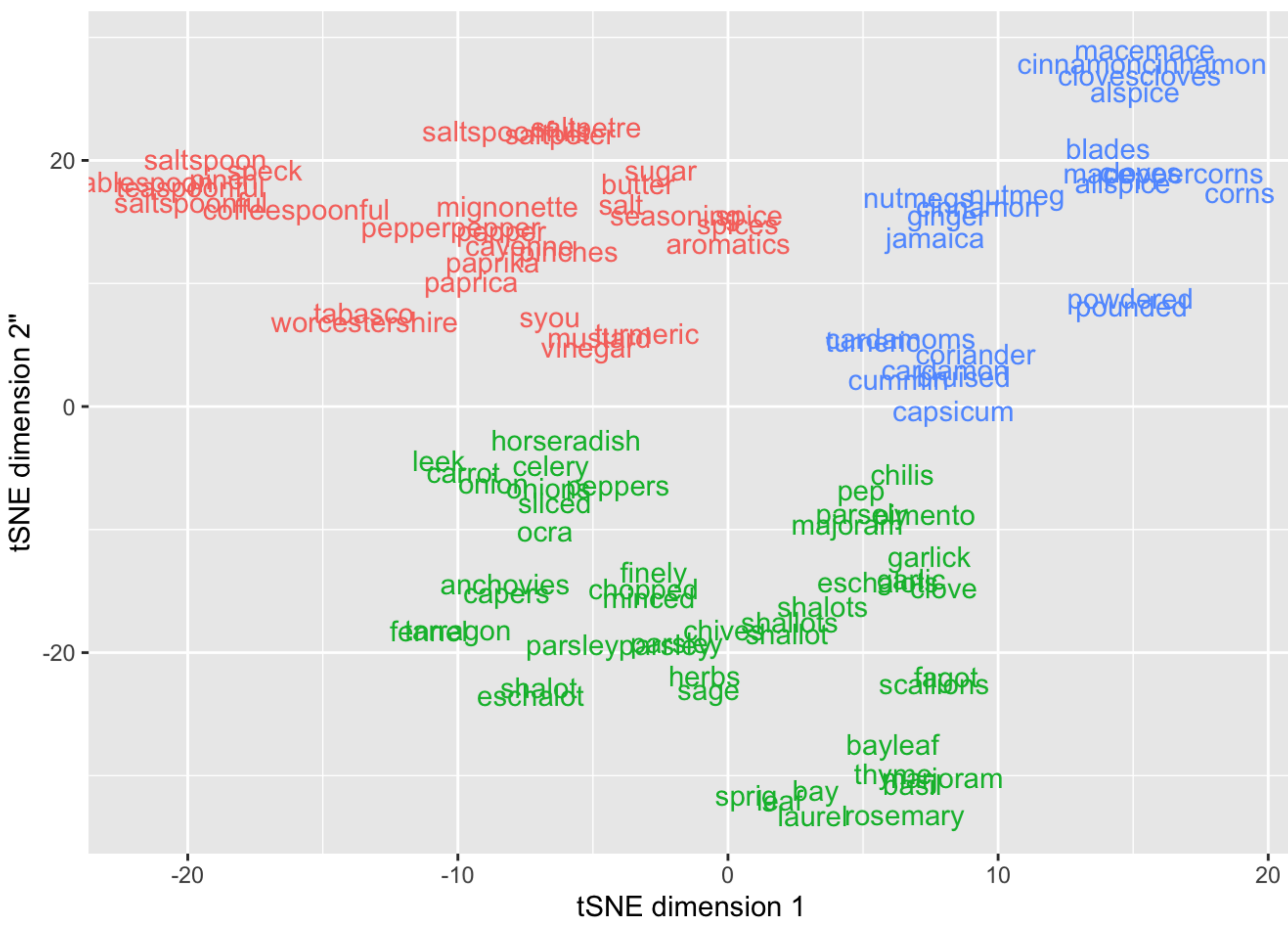
```
## Performing PCA
## Read the 100 x 50 data matrix successfully!
## OpenMP is working. 1 threads.
## Using no_dims = 2, perplexity = 4.000000, and theta = 0.500000
## Computing input similarities...
## Building tree...
## Done in 0.00 seconds (sparsity = 0.172400)!
## Learning embedding...
## Iteration 50: error is 65.275095 (50 iterations in 0.02 seconds)
## Iteration 100: error is 63.217453 (50 iterations in 0.01 seconds)
## Iteration 150: error is 62.399830 (50 iterations in 0.01 seconds)
## Iteration 200: error is 62.399312 (50 iterations in 0.02 seconds)
## Iteration 250: error is 62.399754 (50 iterations in 0.01 seconds)
## Iteration 300: error is 1.062996 (50 iterations in 0.01 seconds)
## Iteration 350: error is 0.852102 (50 iterations in 0.01 seconds)
## Iteration 400: error is 0.785261 (50 iterations in 0.01 seconds)
## Iteration 450: error is 0.756731 (50 iterations in 0.01 seconds)
## Iteration 500: error is 0.742521 (50 iterations in 0.01 seconds)
## Iteration 550: error is 0.731709 (50 iterations in 0.01 seconds)
## Iteration 600: error is 0.729983 (50 iterations in 0.01 seconds)
## Iteration 650: error is 0.724740 (50 iterations in 0.01 seconds)
## Iteration 700: error is 0.721232 (50 iterations in 0.01 seconds)
## Iteration 750: error is 0.721598 (50 iterations in 0.01 seconds)
## Iteration 800: error is 0.719504 (50 iterations in 0.01 seconds)
## Iteration 850: error is 0.718240 (50 iterations in 0.01 seconds)
## Iteration 900: error is 0.718058 (50 iterations in 0.01 seconds)
## Iteration 950: error is 0.717283 (50 iterations in 0.01 seconds)
## Iteration 1000: error is 0.716359 (50 iterations in 0.01 seconds)
## Iteration 1050: error is 0.714786 (50 iterations in 0.01 seconds)
## Iteration 1100: error is 0.713002 (50 iterations in 0.01 seconds)
## Iteration 1150: error is 0.710023 (50 iterations in 0.01 seconds)
## Iteration 1200: error is 0.709658 (50 iterations in 0.01 seconds)
## Iteration 1250: error is 0.708672 (50 iterations in 0.01 seconds)
## Iteration 1300: error is 0.707916 (50 iterations in 0.01 seconds)
## Iteration 1350: error is 0.705931 (50 iterations in 0.01 seconds)
## Iteration 1400: error is 0.705427 (50 iterations in 0.01 seconds)
## Iteration 1450: error is 0.705919 (50 iterations in 0.01 seconds)
## Iteration 1500: error is 0.705767 (50 iterations in 0.01 seconds)
## Iteration 1550: error is 0.703833 (50 iterations in 0.01 seconds)
## Iteration 1600: error is 0.703393 (50 iterations in 0.01 seconds)
## Iteration 1650: error is 0.702515 (50 iterations in 0.01 seconds)
## Iteration 1700: error is 0.701669 (50 iterations in 0.01 seconds)
## Iteration 1750: error is 0.703509 (50 iterations in 0.01 seconds)
## Iteration 1800: error is 0.701852 (50 iterations in 0.01 seconds)
## Iteration 1850: error is 0.700117 (50 iterations in 0.01 seconds)
## Iteration 1900: error is 0.700510 (50 iterations in 0.01 seconds)
## Iteration 1950: error is 0.701443 (50 iterations in 0.01 seconds)
## Iteration 2000: error is 0.700518 (50 iterations in 0.01 seconds)
## Fitting performed in 0.43 seconds.
```

```
embedding_vals = embedding$Y
rownames(embedding_vals) = rownames(surrounding_ingredients)
```

```
embedding_vals = embedding$Y
rownames(embedding_vals) = rownames(surrounding_ingredients)
set.seed(10)
n_centers = 3
clustering = kmeans(embedding_vals,centers=n_centers,
                    iter.max = 5)

# Setting up data for plotting
embedding_plot = tibble(x = embedding$Y[,1],
                        y = embedding$Y[,2],
                        labels = rownames(surrounding_ingredients)) %>%
  bind_cols(cluster = as.character(clustering$cluster))

# Visualizing TSNE output
ggplot(aes(x = x, y=y,label = labels, color = cluster), data = embedding_plot) +
  geom_text() +xlab('tSNE dimension 1') +ylab('tSNE dimension 2"')+theme(legend.position
= 'none')
```



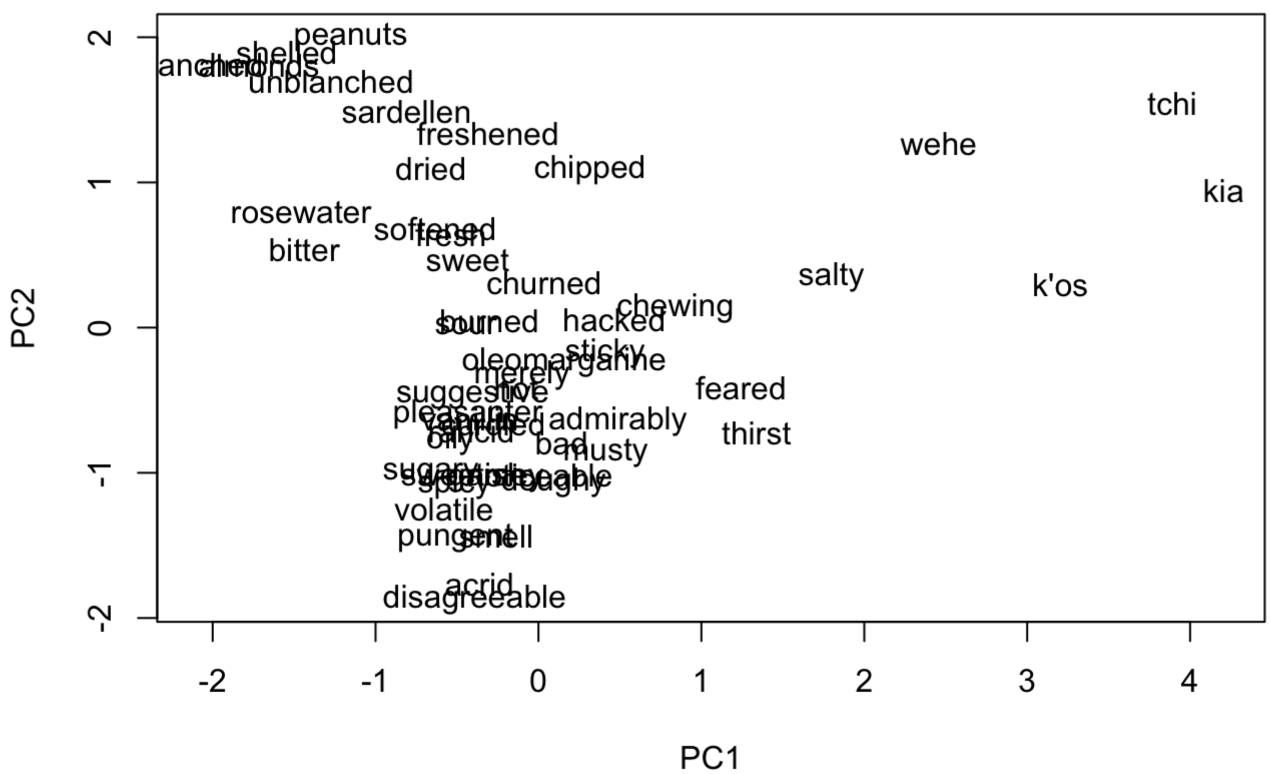

```
# Topics produced by the top 3 words
sapply(sample(1:n_centers,n_centers),function(n) {
  names(clustering$cluster[clustering$cluster==n][1:10])
})
```

```
##       [,1]       [,2]      [,3]
##  [1,] "nutmeg"   "chopped" "butter"
##  [2,] "powdered" "parsley" "sugar"
##  [3,] "cinnamon" "onion"   "salt"
##  [4,] "cloves"   "onions"  "pepper"
##  [5,] "mace"     "celery"  "teaspoonful"
##  [6,] "ginger"   "sliced"  "vinegar"
##  [7,] "pounded"  "finely"  "mustard"
##  [8,] "allspice" "herbs"   "tablespoon"
##  [9,] "bruised"  "minced"  "cayenne"
## [10,] "blades"   "bay"     "pinch"
```

4.

Here I replace three ingredients by three tastes: ‘sweet’, ‘salty’ and ‘bitter’.

```
list_of_ingredients = c('salty','salty','bitter')
closest_ingredients = closest_to(model,model[[list_of_ingredients]], 50)$word
surrounding_ingredients = model[[closest_ingredients,average=F]]
plot(surrounding_ingredients,method="pca")
```



Not quite make sense. There are supposed to be 3 obvious different clusters because they are words of opposite meanings. Then we perform t-SNE. The result quite makes sense.

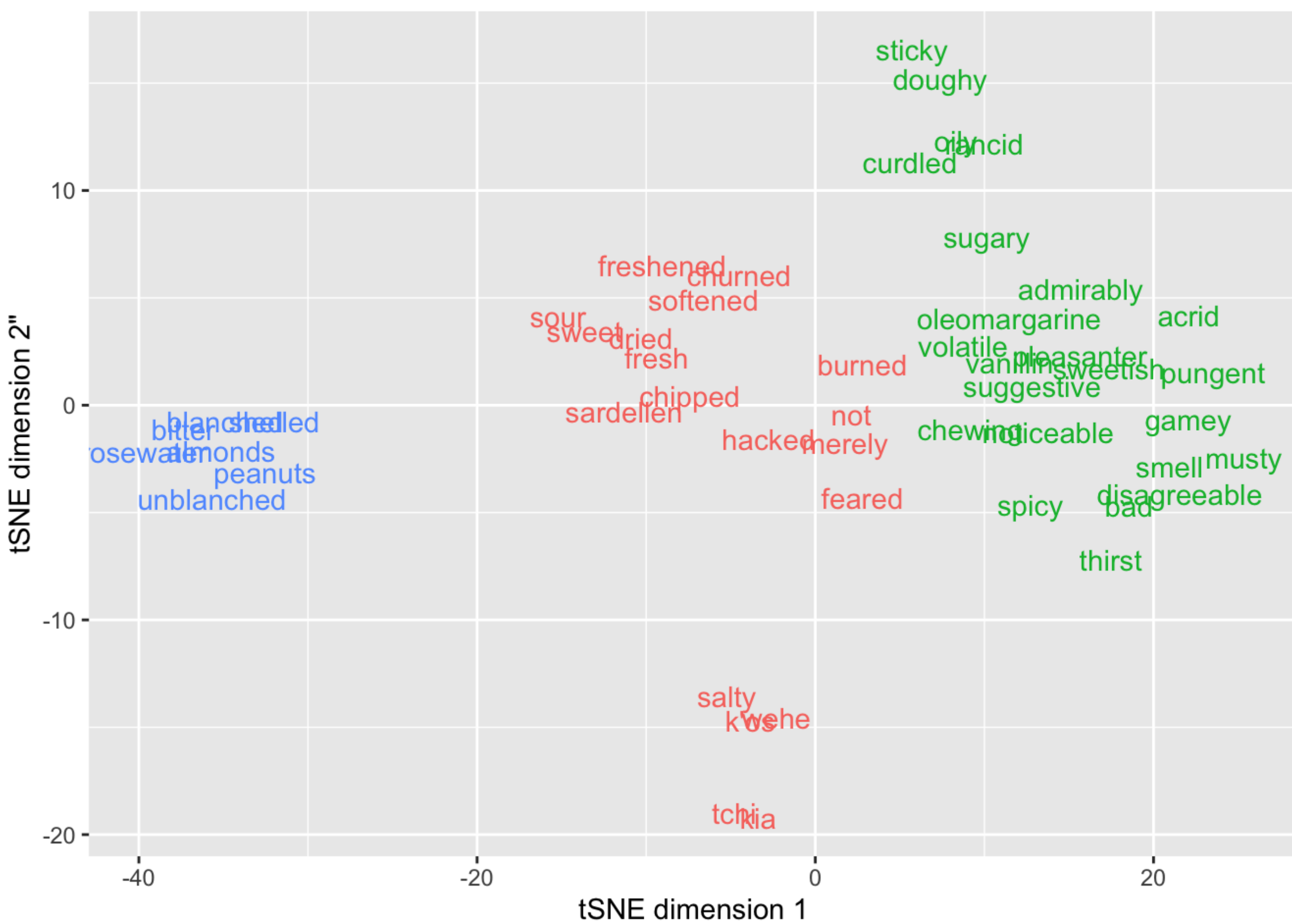
```
embedding = Rtsne(X = surrounding_ingredients, dims = 2,
                  perplexity = 4,
                  theta = 0.5,
                  eta = 10,
                  pca = TRUE, verbose = TRUE,
                  max_iter = 2000)
```

```
## Performing PCA
## Read the 50 x 50 data matrix successfully!
## OpenMP is working. 1 threads.
## Using no_dims = 2, perplexity = 4.000000, and theta = 0.500000
## Computing input similarities...
## Building tree...
## Done in 0.00 seconds (sparsity = 0.363200)!
## Learning embedding...
## Iteration 50: error is 55.881346 (50 iterations in 0.01 seconds)
## Iteration 100: error is 55.362904 (50 iterations in 0.01 seconds)
## Iteration 150: error is 54.614748 (50 iterations in 0.01 seconds)
## Iteration 200: error is 54.609181 (50 iterations in 0.01 seconds)
## Iteration 250: error is 54.611379 (50 iterations in 0.01 seconds)
## Iteration 300: error is 0.669413 (50 iterations in 0.01 seconds)
## Iteration 350: error is 0.606075 (50 iterations in 0.00 seconds)
## Iteration 400: error is 0.589214 (50 iterations in 0.00 seconds)
## Iteration 450: error is 0.581179 (50 iterations in 0.00 seconds)
## Iteration 500: error is 0.573920 (50 iterations in 0.00 seconds)
## Iteration 550: error is 0.566777 (50 iterations in 0.00 seconds)
## Iteration 600: error is 0.567513 (50 iterations in 0.00 seconds)
## Iteration 650: error is 0.566883 (50 iterations in 0.01 seconds)
## Iteration 700: error is 0.560738 (50 iterations in 0.00 seconds)
## Iteration 750: error is 0.560202 (50 iterations in 0.00 seconds)
## Iteration 800: error is 0.559227 (50 iterations in 0.00 seconds)
## Iteration 850: error is 0.557340 (50 iterations in 0.01 seconds)
## Iteration 900: error is 0.561653 (50 iterations in 0.00 seconds)
## Iteration 950: error is 0.562083 (50 iterations in 0.00 seconds)
## Iteration 1000: error is 0.559981 (50 iterations in 0.00 seconds)
## Iteration 1050: error is 0.560380 (50 iterations in 0.00 seconds)
## Iteration 1100: error is 0.553925 (50 iterations in 0.00 seconds)
## Iteration 1150: error is 0.552658 (50 iterations in 0.00 seconds)
## Iteration 1200: error is 0.552017 (50 iterations in 0.01 seconds)
## Iteration 1250: error is 0.552644 (50 iterations in 0.00 seconds)
## Iteration 1300: error is 0.549182 (50 iterations in 0.01 seconds)
## Iteration 1350: error is 0.543661 (50 iterations in 0.00 seconds)
## Iteration 1400: error is 0.539814 (50 iterations in 0.00 seconds)
## Iteration 1450: error is 0.537028 (50 iterations in 0.00 seconds)
## Iteration 1500: error is 0.535264 (50 iterations in 0.00 seconds)
## Iteration 1550: error is 0.537285 (50 iterations in 0.00 seconds)
## Iteration 1600: error is 0.537641 (50 iterations in 0.00 seconds)
## Iteration 1650: error is 0.536899 (50 iterations in 0.00 seconds)
## Iteration 1700: error is 0.533520 (50 iterations in 0.00 seconds)
## Iteration 1750: error is 0.533917 (50 iterations in 0.00 seconds)
## Iteration 1800: error is 0.532094 (50 iterations in 0.00 seconds)
## Iteration 1850: error is 0.531644 (50 iterations in 0.00 seconds)
## Iteration 1900: error is 0.534046 (50 iterations in 0.00 seconds)
## Iteration 1950: error is 0.530506 (50 iterations in 0.00 seconds)
## Iteration 2000: error is 0.531859 (50 iterations in 0.00 seconds)
## Fitting performed in 0.17 seconds.
```

```
embedding_vals = embedding$Y
rownames(embedding_vals) = rownames(surrounding_ingredients)
set.seed(10)
n_centers = 3
clustering = kmeans(embedding_vals,centers=n_centers,
                    iter.max = 5)

# Setting up data for plotting
embedding_plot = tibble(x = embedding$Y[,1],
                        y = embedding$Y[,2],
                        labels = rownames(surrounding_ingredients)) %>%
  bind_cols(cluster = as.character(clustering$cluster))

# Visualizing TSNE output
ggplot(aes(x = x, y=y,label = labels, color = cluster), data = embedding_plot) +
  geom_text() +xlab('tSNE dimension 1') +ylab('tSNE dimension 2"')+theme(legend.position
= 'none')
```




```
# Topics produced by the top 3 words
sapply(sample(1:n_centers,n_centers),function(n) {
  names(clustering$cluster[clustering$cluster==3][1:10])
})
```

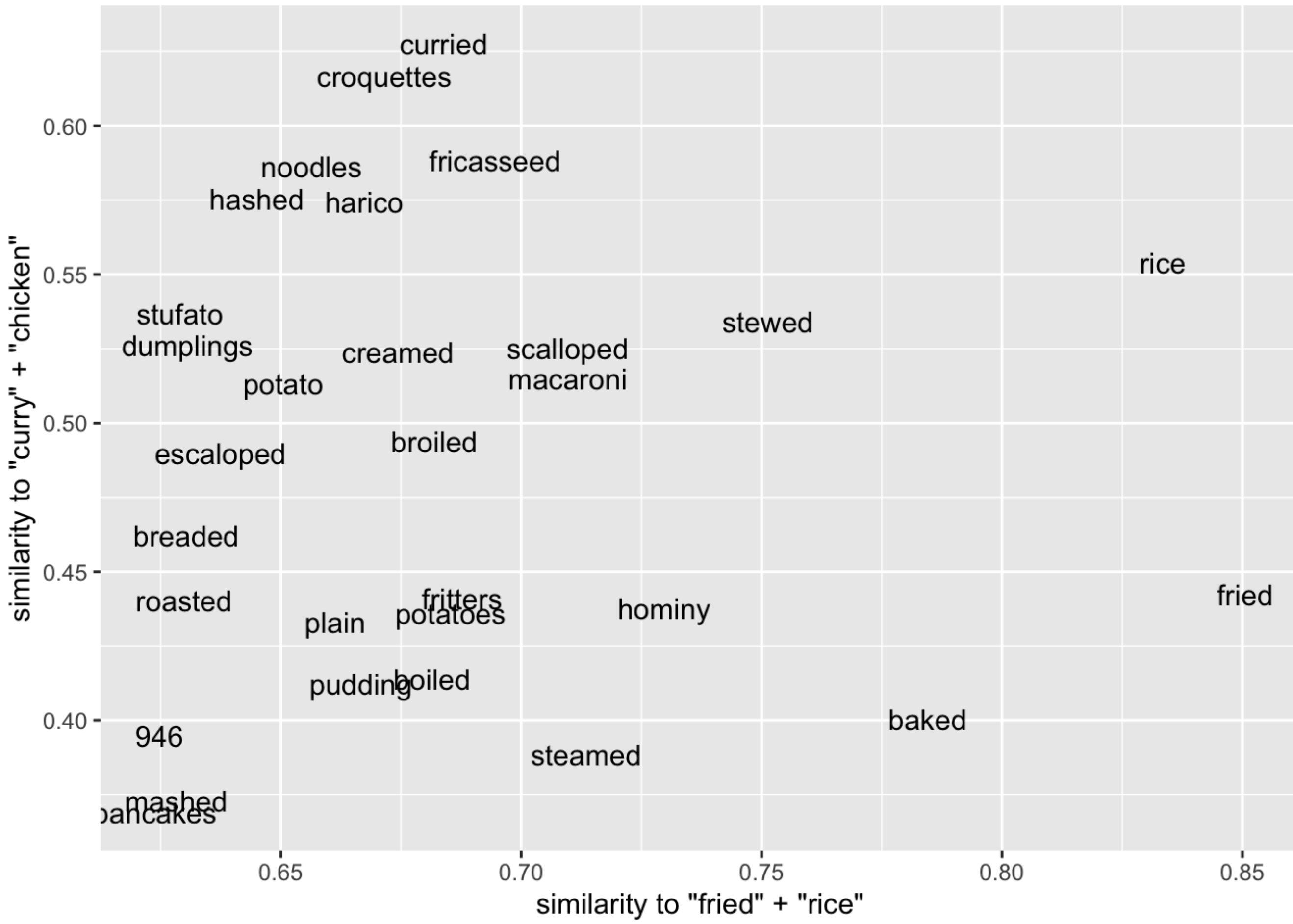
```
##       [,1]         [,2]         [,3]        
##  [1,] "almonds"    "almonds"    "almonds"   
##  [2,] "blanched"   "blanched"   "blanched"  
##  [3,] "bitter"     "bitter"     "bitter"    
##  [4,] "shelled"    "shelled"    "shelled"   
##  [5,] "peanuts"    "peanuts"    "peanuts"   
##  [6,] "rosewater"  "rosewater"  "rosewater" 
##  [7,] "unblanched" "unblanched" "unblanched"
##  [8,] NA           NA           NA          
##  [9,] NA           NA           NA          
## [10,] NA           NA           NA          
```

5.

```
top_evaluative_words = model %>%
  closest_to(~ "fried"+"rice",n=30)
goodness = model %>%
  closest_to(~ "fried"-"rice",n=Inf)
taste = model %>%
  closest_to(~ "curry" + "chicken", n=Inf)

top_evaluative_words %>%
  inner_join(goodness) %>%
  inner_join(taste) %>%
  ggplot() +
  geom_text(aes(x=`similarity to "fried" + "rice"`,
                y=`similarity to "curry" + "chicken"`,
                label=word))
```

```
## Joining, by = "word"
## Joining, by = "word"
```



6.
I found that:
– noodles has strong relations with fried rice and curried chicken
– pancakes has not really strong relationship with curry chicken
– steamed is related with fried rice, probably because rice need to be steamed first

7.
Here we need the package ngram.

```
library(ngram)
text = readLines('/Users/zmt/Desktop/Assignment3/cookbooks.txt')
txt = ''
for (i in text){txt = concatenate(txt,i)}
```

```
txt = preprocess (txt ,case ="lower", remove.punct = TRUE )
```

```
ng <- ngram (txt , n =2)
```

The list is as shown below:

```
cat('Top ten are','\n')
```

```
## Top ten are
```

```
get.phrasetable(ng)[1:10,]
```

```
##        ngrams  freq        prop
## 1      of the  61141 0.005835815
## 2      in the  46763 0.004463457
## 3        in a  45059 0.004300812
## 4      with a  29628 0.002827947
## 5      to the  24058 0.002296299
## 6       it is  23983 0.002289141
## 7    a little  23460 0.002239221
## 8        of a  22043 0.002103970
## 9    with the  20208 0.001928823
## 10      and a  20000 0.001908969
```

Question 4

Part1

1. Here we choose $\theta = 1$ for the first fitting process. Due to the limitation of R calculation, we only sample 10 points as test sets.

```
dat = read.csv('/Users/zmt/Desktop/Assignment3/kernel_regression_1.csv')
set.seed(129)
idx = 985:995
x_observed = dat$x[-idx]
f = dat$y[-idx]
x_prime = dat$x
K = function(x,x_prime,l){
  d = sapply(x,FUN=function(x_in)(x_in-x_prime)^2)
  return(t(exp(-1/(2*l)*d)))
}
mu=0
mu_star=0
l=10
K_f = K(x_observed,x_observed,l)
for (i in 1:dim(K_f)[1]){K_f[i,i]=K_f[i,i]+0.000001}
K_star = K(x_observed,x_prime,l)
K_starstar = K(x_prime,x_prime,l)
mu_star = mu_star + t(K_star)%*%solve(K_f)%*%(f-mu)
Sigma_star = K_starstar - t(K_star)%*%t(solve(K_f))%*%K_star
```

2. Note that here we only take the kernel of log likelihood, which means we will ignore some constant terms. So the value we calculated might be positive.

$$L(y_1, \ldots, y_n) = \frac{1}{(2\pi)^n |\Sigma|} e^{-(y-\mu)^T \Sigma^{-1} (y-\mu)}$$

$$logLL \propto -log(|\Sigma|) - (y-\mu)^T \Sigma^{-1} (y-\mu)$$

```
for (ll in c(1,0.001,0.0001,0.00001)){
mu=0
mu_star=0
l=ll
K_f = K(x_observed,x_observed,l)
# Add little perbutation to make K_f inversable
for (i in 1:dim(K_f)[1]){K_f[i,i]=K_f[i,i]+0.000001}
K_star = K(x_observed,x_prime,l)
K_starstar = K(x_prime,x_prime,l)
mu_star = mu_star + t(K_star)%*%solve(K_f)%*%(f-mu)
Sigma_star = K_starstar - t(K_star)%*%t(solve(K_f))%*%K_star
Sigma_star_test = Sigma_star[idx,idx]
#for (i in 1:dim(Sigma_star)[1]){Sigma_star[i,i]=Sigma_star[i,i]+1}
# Here only take the kernel of likelihood
#logLL = log(dmvnorm(dat$y[idx],mean = mu_star[idx], sigma=Sigma_star_test))
logLL = -log(det(Sigma_star_test)) - t(dat$y[idx]-mu_star[idx])%*%solve(Sigma_star_tes
t)%*%(dat$y[idx]-mu_star[idx])
cat('theta=',ll,'Kernel of Negative Log Likelihood is',-logLL,'\n')}
```

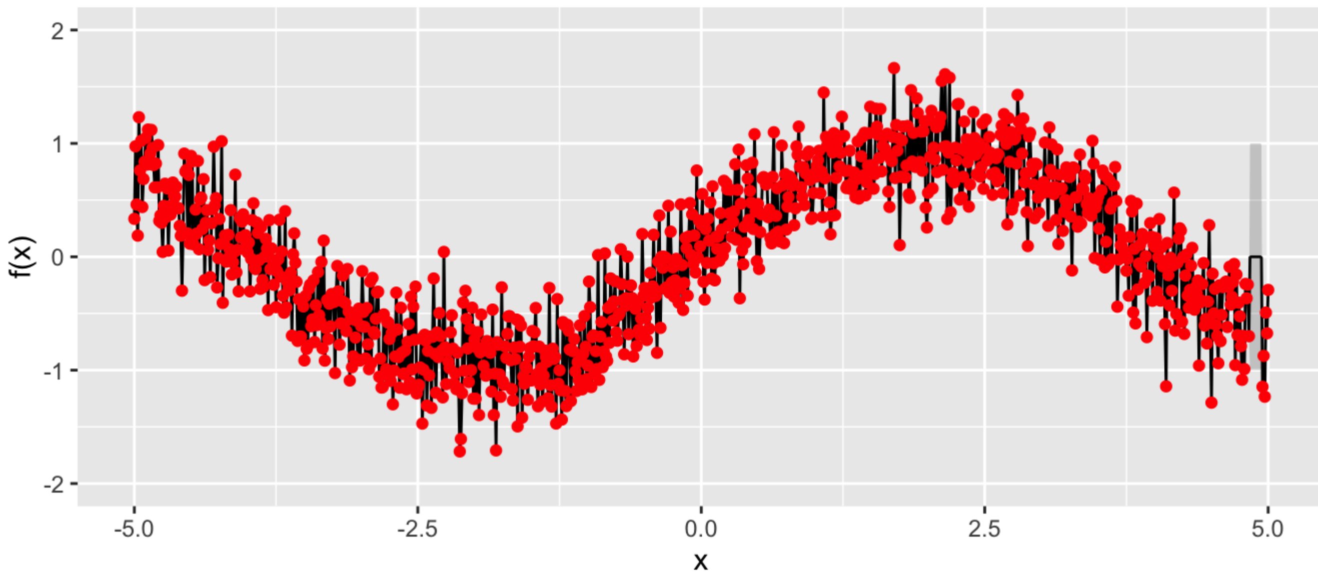
```
## theta= 1 Kernel of Negative Log Likelihood is -7.220478e+15
## theta= 0.001 Kernel of Negative Log Likelihood is 4457304556
## theta= 1e-04 Kernel of Negative Log Likelihood is -1.33645
## theta= 1e-05 Kernel of Negative Log Likelihood is 6.050314
```

$\theta = 1$ is the best choice.

3.

```
plot_gp = tibble(x = x_prime,
                 y = mu_star %>% as.vector(),
                 sd_prime = sqrt(diag(Sigma_star)))

# Plotting values
ggplot(aes(x = x, y = y), data = plot_gp) +
  geom_line()+
  geom_ribbon(aes(ymin = y-sd_prime,ymax = y+sd_prime), alpha = 0.2)+
  geom_point(aes(x =x , y= y), data = tibble(x = x_observed, y = f),
             color = 'red') +
  xlim(c(-5,5))+ylim(c(-2,2))+
  coord_fixed(ratio = 1) +ylab('f(x)')
```



Part 2

1.

```
dat = read.csv('/Users/zmt/Desktop/Assignment3/kernel_regression_2.csv')
set.seed(129)
idx = 990:1001
x_observed = dat$x[-idx]
f = dat$z[-idx]
x_prime = dat$x
K = function(x,x_prime,l){
  d = sapply(x,FUN=function(x_in)(x_in-x_prime)^2)
  return(t(exp(-1/(2*l)*d)))
}
mu=0
mu_star=0
l=10
K_f = K(x_observed,x_observed,l)
for (i in 1:dim(K_f)[1]){K_f[i,i]=K_f[i,i]+0.000001}
K_star = K(x_observed,x_prime,l)
K_starstar = K(x_prime,x_prime,l)
mu_star = mu_star + t(K_star)%*%solve(K_f)%*%(f-mu)
Sigma_star = K_starstar - t(K_star)%*%t(solve(K_f))%*%K_star
```

2.

```
for (ll in c(0.1,0.01,0.00001)){
mu=0
mu_star=0
l=ll
K_f = K(x_observed,x_observed,l)
for (i in 1:dim(K_f)[1]){K_f[i,i]=K_f[i,i]+0.000001}
K_star = K(x_observed,x_prime,l)
K_starstar = K(x_prime,x_prime,l)
mu_star = mu_star + t(K_star)%*%solve(K_f)%*%(f-mu)
Sigma_star = K_starstar - t(K_star)%*%t(solve(K_f))%*%K_star
Sigma_star_test = Sigma_star[idx,idx]
logLL = -log(det(Sigma_star_test)) - t(dat$y[idx]-mu_star[idx])%*%solve(Sigma_star_tes
t)%*%(dat$y[idx]-mu_star[idx])
cat('theta=',ll,'Negative Log Likelihood is',-logLL,'\n')}
```

```
## theta= 0.1 Negative Log Likelihood is 1960834402
## theta= 0.01 Negative Log Likelihood is 6756231175
## theta= 1e-05 Negative Log Likelihood is 8570229749
```

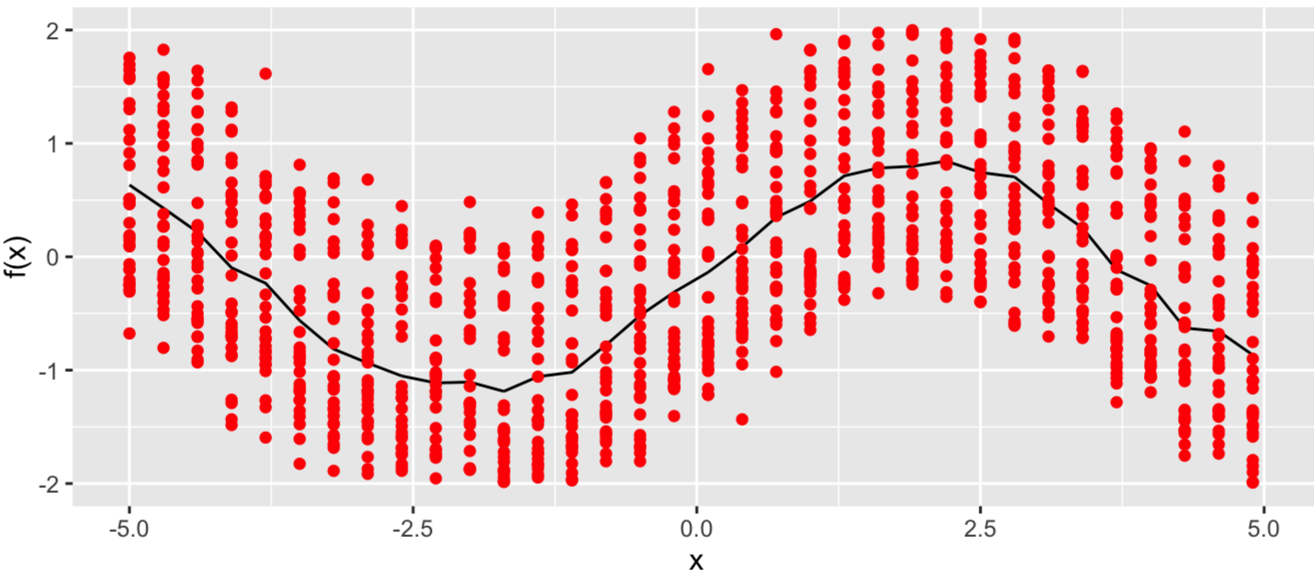
$\theta = 0.1$ is the best choice.

3.

```
plot_gp = tibble(x = x_prime,
                 y = mu_star %>% as.vector(),
                 sd_prime = sqrt(diag(Sigma_star)))

# Plotting values
ggplot(aes(x = x, y = y), data = plot_gp) +
  geom_line()+
  geom_ribbon(aes(ymin = y-sd_prime,ymax = y+sd_prime), alpha = 0.2)+
  geom_point(aes(x =x , y= y), data = tibble(x = x_observed, y = f),
             color = 'red') +
  xlim(c(-5,5))+ylim(c(-2,2))+
  coord_fixed(ratio = 1) +ylab('f(x)')
```

```
## Warning: Removed 38 rows containing missing values (geom_point).
```


2
1
0
-1
-2
f(x)
-5.0
-2.5
0.0
2.5
5.0
x