Matt Dickenson `mcd31`
STA561/CS571 — Fall 2013 **Homework 5** Due: 28 October, 2013

*Homework Notes:* I did not work with anyone else on this homework or refer to resources other than the course notes, textbook, and course Piazza page.

## Problem 1

**A** Figure 1 shows the distribution of eigenvalues for each matrix covariance matrix of $X$ with different values of $\psi$ ($X_i$ uses $\psi_i$). All of the distributions are right-skewed, but the mean of the eigenvalues increases and the variance decreases as the amount of noise ($\psi$) in the original matrix increases. The eigenvalues were normalized using the $\ell^2$ norm.

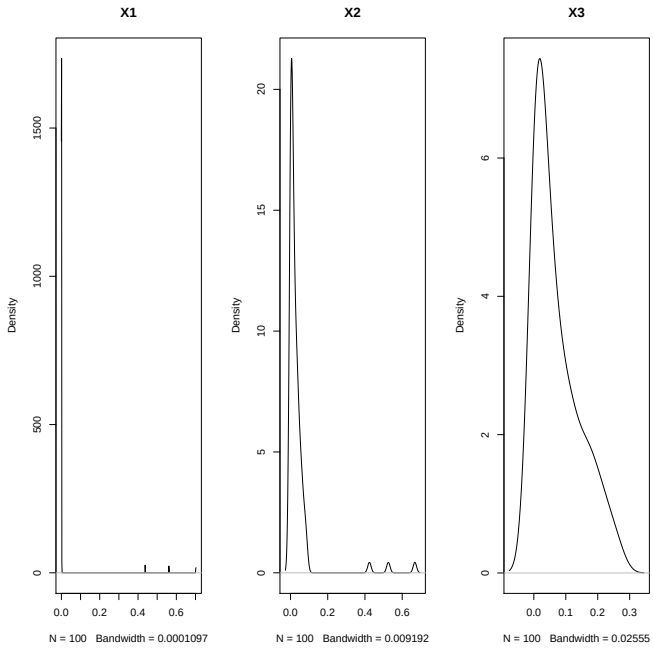


Figure 1: Distribution of Normalized Eigenvalues for Three Values of $\psi$

**B** Table 1 presents the root mean squared error (RMSE) between the covariances of the X matrices and the matrix reconstructions using the first three eigenvectors (and eigen values). Overall, the eigenvectors do a good job of recapitulating the original data matrices. Those with less noise (small $\psi$) do a better job (lower RMSE) than those with more noise.

**C** When we reconstruct the matrices we are able to obtain an estimate of the covariance of $X$, subject to some noise. This could be useful for identifying the number of components that could be used in our analysis, as long as we assume that the level of noise is relatively low.

Table 1: RMSE between Cov(X) and Matrix Reconstructions

| $\psi$ | RMSE |
|---|---|
| 0.2 | 0.0054 |
| 2 | 0.553 |
| 10 | 13.029 |

**Problem 2**

**A and B** Steps for building string kernel:

1. Create all possible "words" up to length three (3) using the letters A, C, G, and T (`combos`)
2. Write a function to count all occurrences of a substring within a string (`substringCount`)
3. Write a string kernel function to loop through the words generated in step 1, count their occurrence within a string using the function in step 2, multiply, and apply a weight proportional to the length of the word/substring (`stringKernel`)

The $(i,j)^{th}$ Gram matrix entry, $K(x_i, x_j)$ was computed as $\Sigma_a w(a)\phi(x_i, a)\phi(x_2, a)$ where $a$ is a word/substring, $\phi(x, a)$ counts the occurrences of $a$ in $x$, and $w(a) = \frac{|a|}{228}$ because there are a total of 228 characters in all the words that can be created as a combination of A, C, G, and T.

Listing 1: R Code for 2A and 2B

```
# Problem 2
train = read.table("training_data_gp.txt", as.is=TRUE, header=FALSE)
dim(train)
n = nrow(train)
test = read.table("test_data_gp.txt", as.is=TRUE, header=FALSE)
dim(test)
names(train) = names(test) = c("sequence", "intensity")

# a
combos1 = c("A", "C", "G", "T")
substrings = expand.grid(first=combos1, second=combos1)
combos2 = paste(substrings$first, substrings$second, sep="")
substrings = expand.grid(first=combos1, second=combos1, third=combos1)
combos3 = paste(substrings$first, substrings$second, substrings$third, sep=
combos3
combos = c(combos1, combos2, combos3)
length(combos)
```

```
18
19 substringCount = function(substring, string) {
20   count = 0
21   i = 1
22   n = nchar(string)
23   found = regexpr(substring, substr(string, i, n), perl=TRUE)
24   while (found > 0) {
25     count = count + 1
26     i = i + found
27     found = regexpr(substring, substr(string, i, n), perl=TRUE)
28   }
29   return(count)
30 }
31
32
33 charsInCombos = 4+(2*16)+(3*64)
34
35 stringKernel = function(x1, x2){
36   s = 0
37   for(combo in combos){
38     phi = substringCount(combo, x1) * substringCount(combo, x2)
39     w = nchar(combo)/charsInCombos
40     s = s + (w*phi)
41   }
42   return(s)
43 }
44
45 n = nrow(train)
46 gram = matrix(NA, nrow=n, ncol=n)
47 start = proc.time()
48 for(i in 1:n){
49   for(j in i:n){
50     gram[i,j] = gram[j,i] =
51       stringKernel(train$sequence[i], train$sequence[j])
52   }
53   print(i)
54 }
55
56
57 # b
58 rbf = function(x1, x2, sigma = 5){
59   dist = stringKernel(x1, x2)
60   k = exp(-dist/(2*sigma^2))
```

```
61    return(k)
62  }
63
64  complete.data = rbind(train, test)
65  head(complete.data)
66  n = nrow(complete.data)
67  rbfMatrix = matrix(NA, nrow=n, ncol=n)
68  for(i in 1:n){
69    for(j in i:n){
70      rbfMatrix[i,j] = rbfMatrix[j,i] =
71        rbf(complete.data$sequence[i], complete.data$sequence[j])
72    }
73    print(i)
74  }
```

**C** Figure 2 presents a scatter plot of predicted intensities versus the measured intensities in the test data. The RMSE was calculated to be 0.654.

**D** There are at least three ways that we would improve the prediction acuracy for our regression:

1. Consider longer substrings of the sequence, such as $|A| = 4, 5, ...$
2. Modify the weights put on shorter substrings, including weighting single character strings at zero
3. Try other values of $\sigma$

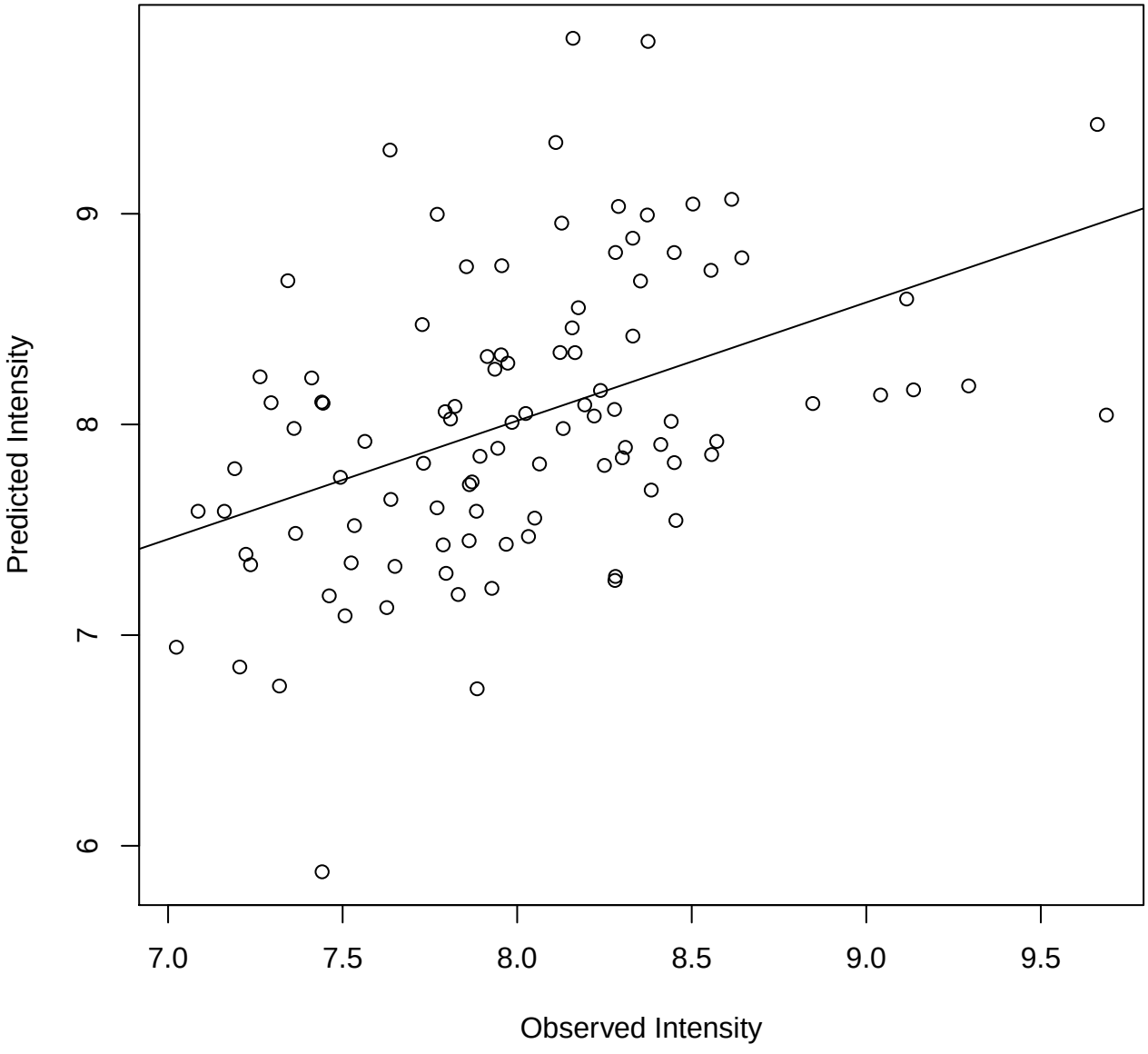


Figure 2: Scatterplot of Predicted versus Measured Intensities