Homework 1

Jessica Temporal 7547611

August 15, 2016

Contents

1) Load the Golub ALL/AML training set data file into R. Set the row names to the first column values (Affymetrix fragment names) and remove the first column. Look at the dimensions and verify that you have 38 arrays and 7,129 genes.

```
file <- "golubTrain.txt"
golub_data <- read.table(file, header = T)
rownames(golub_data) <- as.character(golub_data[,1])
golub_data$Gene <- NULL
dim(golub_data)
```

```
## [1] 7129   38
```

2) Download and load the 3 class annotation file (golubTrainClass2.txt) for this data set into R.

```
file2 <- "golubTrainClass2.txt"
class_data <- read.table(file2)
```

3) Cast the data to a data frame.

```
golub_df <- as.data.frame(golub_data)
```

4) Make the names of the data frame the annotation file labels (remember that depending on how you read in the annotation file, it may be a dataframe with 1 column.

```
names(golub_df) <- class_data[,1]
```

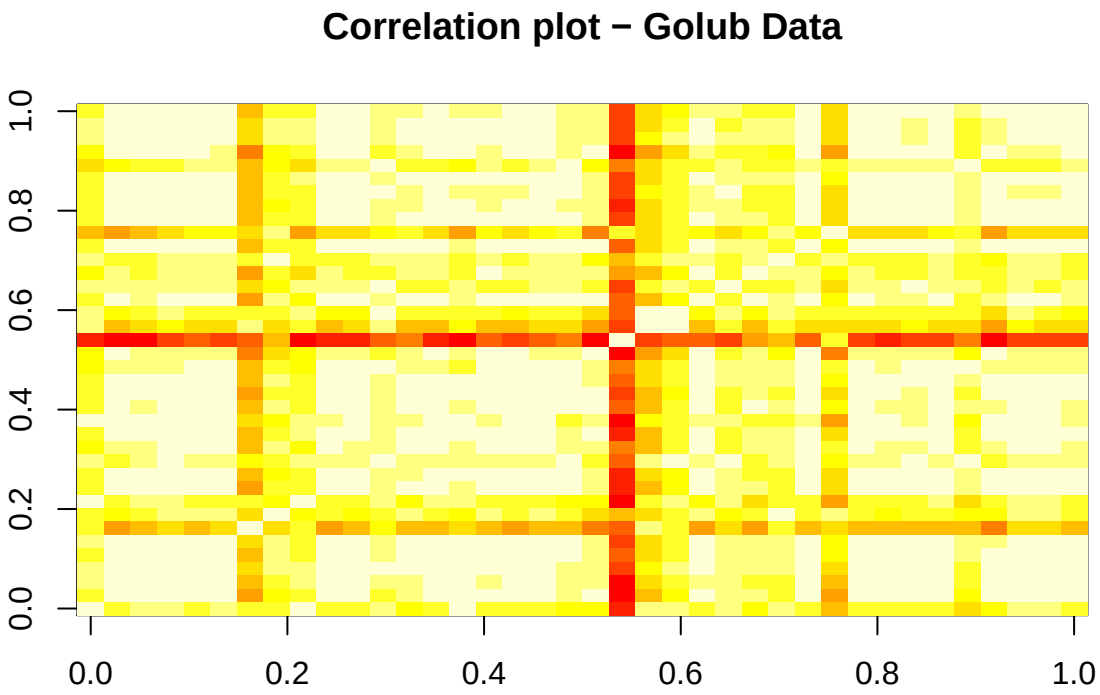
5) Subset the data by the first 100 genes.

```
golub_sub <- golub_df[1:100,]
```

6) When utilizing only the first 100 genes, there exists one aberrant outlier sample. Identify this outlier sample using the following visual plots:

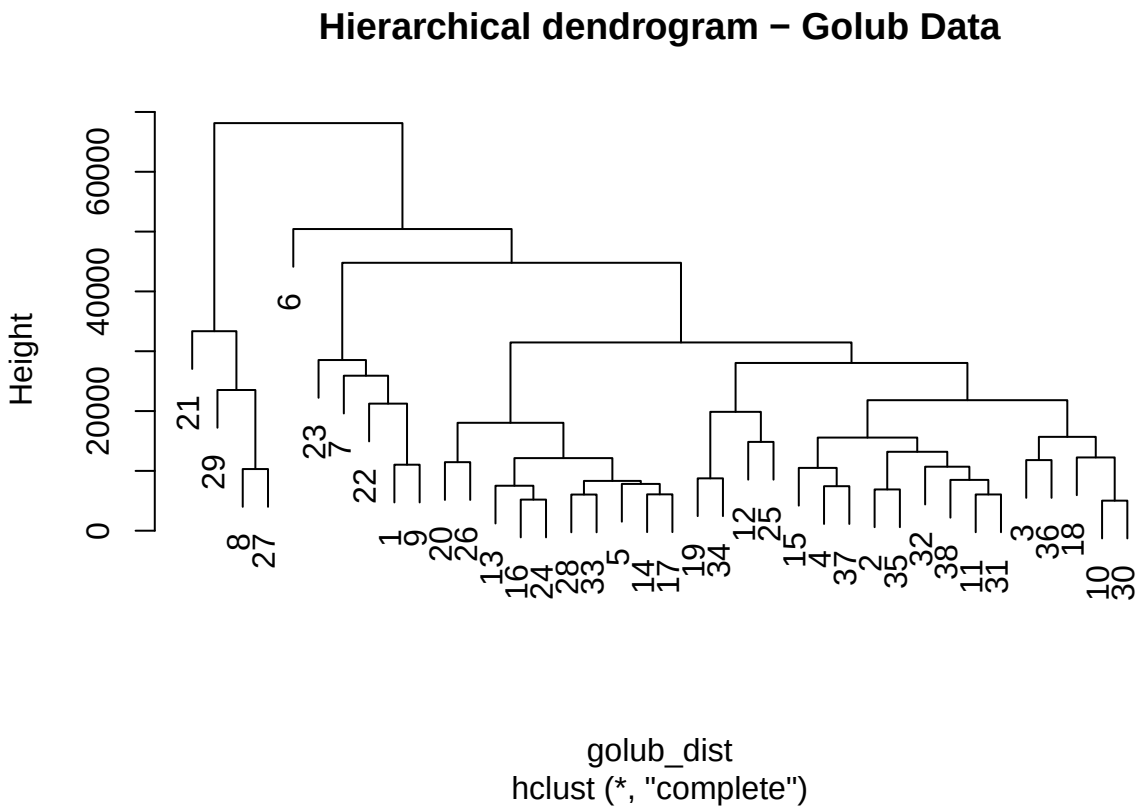
6.1) Correlation plot (heat map)

```
golub_cor <- cor(golub_sub)
image(golub_cor, main = "Correlation plot - Golub Data")
```



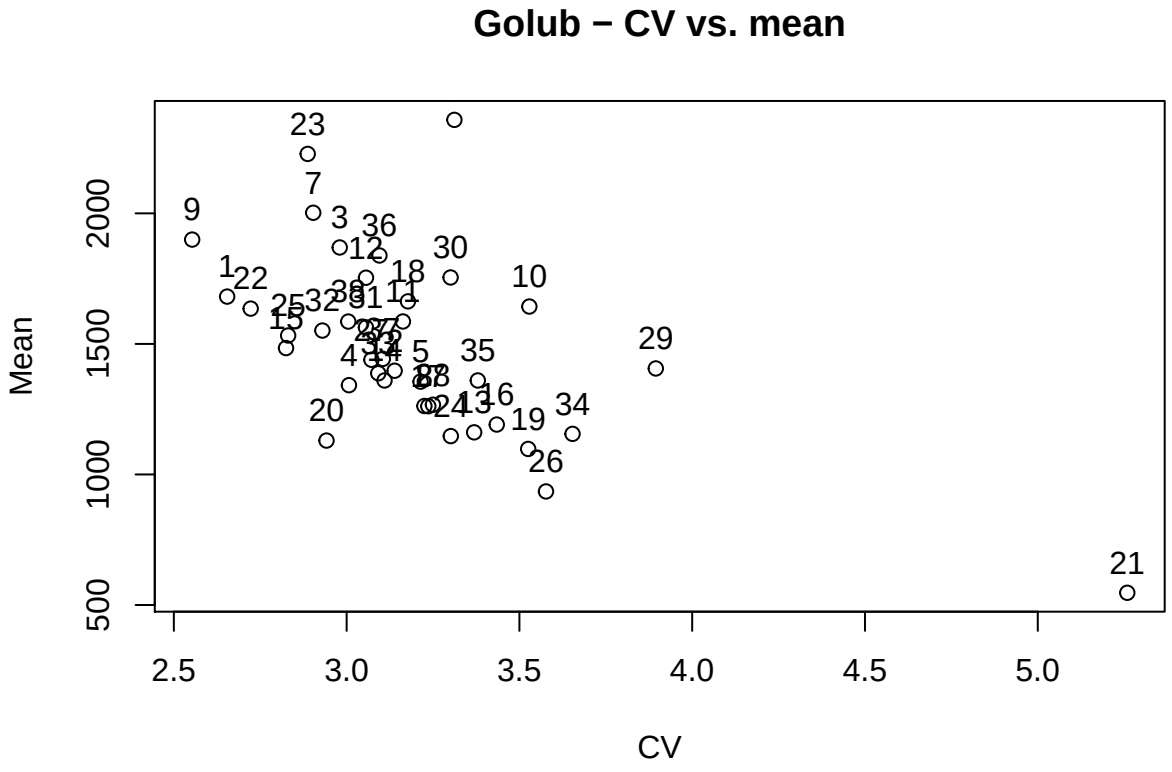
6.2) Hierarchical clustering dendrogram

```
golub_dist <- dist(t(golub_sub))
golub_hclust <- hclust(golub_dist)
plot(golub_hclust, main = "Hierarchical dendrogram - Golub Data", labels = c(1:38))
```



6.3) CV vs. mean plot

```
golub_sub_mean <- apply(golub_sub, 2, mean)
golub_sub_cv <- (apply(golub_sub, 2, sd))/golub_sub_mean
plot(golub_sub_cv, golub_sub_mean, xlab = "CV", ylab = "Mean",
     main = "Golub - CV vs. mean")
text(golub_sub_cv, golub_sub_mean, labels = c(1:ncol(golub_sub)), pos = 3)
```



7) Now download and load the Spellman yeast data set. Remember to set the row names to the first column as you did before with the leukemia dataset.

```
spell <- "spellman.txt"
spell_data <- read.table(spell, header = T)
rownames(spell_data) <- as.character(spell_data$row.names)
spell_data$row.names <- NULL
```

8) Cast the data to a data frame and subset to only work with the cdc28 experiment samples.

```
spell_df <- as.data.frame(spell_data)
cdc28 <- subset(spell_df, select = grep("cdc28", names(spell_df)))
```

9) Use both the function and call to the function below to fill all "NA" values with the computed row means.

```
miss.fill <- function(x) {
    if(sum(is.na(as.numeric(x))) ==17 ) {
        x[is.na(x)] <-0
    }
    if(sum(is.na(as.numeric(x))) > 0 & sum(is.na(as.numeric(x))) <17 ) {
        x[is.na(x)] <- mean(as.numeric(x[!is.na(x)]))
    }
    return(x)
}

cdc28_fill <- as.data.frame(t(apply(cdc28, 1, miss.fill)))
cdc28_fill <- round(cdc28_fill, 2)
```

10) Calculate the kmeans clustering method on all 6,178 genes, using 10 cluster centers and 100 iterations.

```
cdc28_kmeans <- kmeans(cdc28_fill, centers = 10, iter.max = 100)
```

11) Look for gene #2 (YAL002W) and find the cluster that it belongs to. Using these genes, calculate the distance from each gene to gene #2 (use manhattan distance in `dist()` function)

```
groups <- cdc28_kmeans$cluster
head(groups)
```

```
## YAL001C YAL002W YAL003W YAL004W YAL005C YAL007C
##       4       4       3       2       2       6
```

```
groups['YAL002W'] # cluster 4
```

```
## YAL002W
##       4
```

```
cluster <- groups == 4
cluster <- dimnames(cdc28_fill)[[1]][cluster]
dist2genes <- dist(cdc28_fill[cluster,], method = 'man')
```

12) Cast the distance object to a matrix and get the column that gene #2 corresponds to. Hint: If gene #2 is in column #1 of your matrix, get the corresponding cluster member distances with: `gene.dist <- gene.dist[2:nrow(gene.dist),1]` If gene #2 is in column #4 of your matrix, get the corresponding cluster member distances with: `gene.dist <- gene.dist[c(1:3,5:nrow(gene.dist)),4]`

```
dist_mat <- as.matrix(dist2genes)
gene2 <- dist_mat[2:nrow(dist_mat),'YAL002W']
```

13) Get the weights of each gene as a percentage of the sum of distance values. Assuming that the first array (cdc28_0) is missing a value for gene #2, calculate the weighted mean from the gene weight vector for this missing value. Print out this weighted mean value.

```
weights <- as.numeric(gene2/sum(gene2))
weighted_mean <- weighted.mean(x = as.numeric(gene2), w = weights)
weighted_mean
```

```
## [1] 5.527842
```