Fall 2015 Biostat 615 Homework #4 (Total 30 pts)

Due by Thursday Novemeber 12th, 2015 noon as a compressed file "hw4.tar.gz" containing the required C++ source code files for the following three problems. Please read the instruction carefully before you start to work on your homework.

Instruction

- The additional grading will be executed on every day from November 4th to November 11th. The final grading will be executed exactly at 12:10pm, Thursday November 12th, 2015.
- Your program might get input from `cin` or `argv`, according to the specific description given each problem.
- Make your algorithm as efficient as you can, as some of the test cases may be computationally challenging and your program will be terminated if it does not finish after running for 10 seconds and you will lose the points for those test cases. However, you should always turn in a program even if it is not very efficient so that you can at least get partial points.
- All your programs (e.g., if it is named as `program.cpp`) will be compiled and tested on the server `scs.itd.umich.edu` using the following command:

```
g++ -O -o program -I ~jiankang/Public/include/ program.cpp
```

  where "program" will be replaced by the corresponding program name for each problem.
- For this homework, you are allowed to use `boost` library in your code.

Problem 1 - Benjamini-Hochberg-Yekutieli FDR control method (10 pts)

The false discover rate (FDR) is the expected number of rejections when the null hypothesis is true. Suppose m hypothesis tests have been conducted that produced p-values $p_1, \ldots, p_m$. Then, the Benjamini/Hochberg/Yekutieli method bounds the FDR by a specified value $\alpha \in (0, 1)$. The calculations proceed according to the following Algorithm

Algorithm 1 Benjamini-Hochberg-Yekutieli FDR control method

1: Arrange input p-values $p_1, \ldots, p_m$ in numerically ascending order as $p_{(1)} \leq \ldots p_{(m)}$
2: Set $q = \alpha / \sum_{j=1}^{m} 1/j$.
3: Set $k = \max\{1 \leq i \leq m : p_{(i)} \leq q(i/m)\}$.
4: **if** k exists **then**
5: Reject the null hypotheses corresponding to $p_{(1)}, \ldots, p_{(k)}$
6: **return** the indices of the null hypotheses, i.e. $(1), \ldots, (k)$
7: **else**
8: Reject nothing
9: **return** 0.
10: **end if**

q-value(i)=p(i)*length(p)/rank(p)

Write a C++ program `fdrBHY.cpp` to implement the above algorithm. The argument of the program is α (`argv`) and the input values are the m p-values (`cin`). The output is the indices of

null hypotheses that should be rejected after the FDR control, separated by a white space. If no hypothesis should be rejected, then the output is 0. Example runs of valid input arguments and values are

```
user@host:~/Private/biostat615/hw4$ ./fdrBHY 0.01
0.05 0.05 0.005 0.005 0.0001
5
user@host:~/Private/biostat615/hw4$ ./fdrBHY 0.01 < test.input
0
```

where "`test.input`" contains 20,000 of 1e-3 that are separated by white spaces.

Problem 2. Read CSV file and Determine Data Type (10 pts)

Suppose there is a CSV file with n columns ($n > 1$). In `R`, you read this file into a data frame using function `read.csv()` with the default settings and use function `class()` to show the data type of each column. Write a C++ program `readCSV` to achieve the same goal and you can use the `boost` library. The input argument is the CSV file name. The output is the data types (the same as the output of `R class()` function) of all columns that are separated by white spaces. Example runs of valid input arguments are

```
user@host:~/Private/biostat615/hw4$./readCSV test.csv
numeric factor  factor  factor  integer logical logical
```

where "test.csv" contains the following data values

```
"X1","X2","X3","X4","X5","X6","X7"
"5.1","a","a","N/A","1","TRUE","T"
"6","b","b","2","2","FALSE","F"
"7","c","N/A","3","3","TRUE","NA"
```

Problem 3. Simple Classifier (10 pts)

Consider a binary classification problem where the outcome y only takes 0 and 1 and predictors $x_1, \ldots, x_p$ are continuous real numbers. The following `R` function `simpleClassifier` implements a simplified version of a machine learning algorithm to solve this problem.

```
simpleClassifier = function(train_dat_filename, test_dat_filename){
  train_dat = read.csv(train_dat_filename)
  test_dat = read.csv(test_dat_filename)
  id = (train_dat[ , 1] == 1)
  center_1 = apply(train_dat[id, -1], 2, mean)
  center_0 = apply(train_dat[!id, -1], 2, mean)
  s_1 = apply(train_dat[id, -1], 2, var)
  s_0 = apply(train_dat[!id, -1], 2, var)
  dim(center_1) = c(1, length(center_1))
  dim(center_0) = c(1, length(center_0))
  dim(s_1) = c(1, length(s_1))
  dim(s_0) = c(1, length(s_0))
  ones = matrix(1, nrow = nrow(test_dat), ncol = 1)
  dist_1 = apply((test_dat[ , -1]-ones%*%center_1)^2/(ones%*%s_1),1,sum)
```

```
  dist_0 = apply((test_dat[ , -1]-ones%*%center_0)^2/(ones%*%s_0),1,sum)
  y_pred = as.numeric(dist_1 < dist_0)
  accuracy = mean(test_dat[ , 1] == y_pred)
  return(accuracy)
}
```

This function will train the classifier using the data in the CSV file *train_dat_filename* and compute the classification accuracy using the testing data in the CSV file *test_dat_filename*. Please read this `R` function carefully and figure out the input data format and understand the algorithm, then write a more efficient C++ program `simpleClassifier.cpp` to implement the algorithm in this `R` function. The input arguments are the names of training dataset and testing dataset and the output is the prediction accuracy (`setprecision(4)`). You can assume that in the training dataset there exist at least two observations with their outcome variables taking values 0 and 1, respectively. There are no missing values in both training and testing datasets. Example runs of valid input arguments are

```
user@host:~/Private/biostat615/hw4$ ./simpleClassifier train.csv test.csv
1
```

where "`train.csv`" is

```
y,x1,x2,x3,x4,x5
1,-0.194,-0.731,2.922,0.04,-0.266
0,-2.301,-1.649,-2.322,-0.395,0.317
1,0.439,0.519,1.232,-0.309,0.583
0,-0.357,-1.117,-0.464,-1.461,-1.04
0,-0.686,-1.235,-0.546,-1.666,-0.115
```

and "`test.csv`" is

```
y,x1,x2,x3,x4,x5
1,0.568,1.124,1.557,1.143,2.417
0,-2.266,-1.635,-3.574,-2.157,-1.561
1,1.031,0.337,0.647,1.369,0.419
1,2.535,0.507,3.115,1.812,1.176
0,-2.234,-0.648,0.36,-2.209,-1.829
```