Northeastern University

Course:	CS6020
Assignment:	Module 2 - Programming in R
Total Points:	100
Date Due:	Posted on Blackboard

Learning Objectives

In this assignment, you will learn how to:

- read and write simple text files
- load data selectively
- control conditional and iterative execution
- create data frames and vectors

Tasks

Before diving into the programming problems, study the data files that are provided for the assignment:

1. (5 Points) Load the data file into an appropriate data object of your choice. The file is compressed, so either use a tool to uncompress the file or look up how to read compressed files in R. Look at the other questions to determine what object is most appropriate: data frame versus vector? Figure out how to deal with missing values. Add your strategy as a comment to the functions that you write in the next few tasks.

2. (30 Points) Write a function called `TotalNumDelays(Carrier)` that finds and returns the total number of delays of a carrier. You must calculate the number of delays, not the total number of minutes all flights were late in aggregate.

3. (30 Points) Write a function called `TotalDelaysByOrigin(Origin)` that finds and returns the total number of delays for a particular airport.

4. (25 Points) Write a function called `AvgDelay(Carrier,Dest)` that calculates and returns the average arrival delay for a carrier flying into the specified destination airport.

5. (10 Points) Improve your program and functions so that the data file is not loaded repeatedly, but only once or when needed. This means that the data frame should not

be loaded repeatedly but rather only if it hasn't been loaded before. You will need to a check to see if the data frame has been loaded.

Clarifications

Part of the assignment is to practice good coding standards: consistent naming, formatting with indentation, testing input values, dealing with errors. You need to submit an R script, which is a text file having a .R extension, i.e., the file you are loading into R.

The functions must return the result as a return value which is then print from the calling function or main script. This makes the function truly a "function" versus a "procedure". Functions return values, procedures do work and do not return values. Returning a value also makes it easier to deal with error conditions. For example, you could return -1 if the data could not be found. The calling function can then test for that condition.

The file can either be uncompressed after it's downloaded or you can use various R functions to load a zipped (compressed) file.