**Homework 10**

**Due Nov 11 (Sunday)**

In this homework, we will consider the IRA twitter dataset but with a focus on the users.

1. The file ira\_users\_csv\_hashed.csv contains all users that have tweets in the ira\_tweets\_csv\_hashed.csv file. Each user is uniquely identified by their userid. For the 20 users in the ira\_users\_csv\_hashed.csv file with the most tweets in ira\_tweets\_csv\_hashed.csv, create a csv file of the users tweets by filtering the ira\_tweets\_csv\_hashed.csv file for the given userid. Put the 20 user tweet files in a separate directory with file names that distinguish the different users. (Use table() and sort() to determine the top 20 tweeters.)
2. Create a User object as follows.
   1. Write a constructor function **User**(userid, dirpath), where userid is the user id of the given user and dirpath is the directory path in which all the files created in Problem 1 are located.
   2. Write a function **get\_userid.User**(u) that returns the userid of the user object.
   3. Write a function **get\_tweet\_dynamics.User**(u) that returns a vector of the tweet times for the given user object.
   4. Write a function **compute\_spline\_regression.User**(u, show\_plot=T) that returns spline ()object created by stats::smooth.spline. If show\_plot is True, then a plot of the spline regression is shown, otherwise no plot is shown. Apply a spline regression by grouping the tweet times by month. This method is a utility method and should not directly access the internal structure of u, but instead should use the get functions.
   5. Write a function **compute\_kernel\_density.User**(u, show\_plot=T) that returns a density object created by the density function. The density function builds a kernel density estimator, see R help and other resources for details. (A density estimator allows us to “smooth” the data without grouping as in part c). If show\_plot is True, then a plot of the estimated density is shown. The method is a utility method.
3. Create a UserCollection object that abstractly is a collection of user objects
   1. Write a constructor **UserCollection**()
   2. Write a function **get\_all\_userids.UserCollection**(uc) that returns all userid’s of user objects in the uc object.
   3. Write a function **get\_user.UserCollection**(uc, i) that returns a user object for the ith user in the UserCollection object uc.
   4. Write a function **add\_user.UserCollection**(uc, u) that adds a user object, u, to the UserCollection object, uc. This is a set function that changes the uc object. Recall that everything inside a function is lost once the function exists, so the function must return a UserCollection object and would typically be called as follows:

**uc <- add\_user.UserCollection(uc, u)**

* 1. Write a function **remove\_user.UserCollection**(uc, u) that removes a user object, u, from the UserCollection object, uc. The is also a set function.
  2. Write a function **correlation\_matrix.UserCollection**(uc) that returns a correlation matrix for the tweet dynamics of the users. The correlations should be computed based on the kernel density estimator of the user object.

1. Build a UserCollection object with the 20 users you considered in problem 1. Then, build the correlation matrix and identify the two users with tweet times that are most correlated.