Student **CISC4900 / Spring2017**
Name: Donat Vucetaj

Advisor
Name: Professor Theodore Muth / Professor Jessica Joyner
Address: 2900 Bedford Ave, Brooklyn, NY 11210
Brooklyn College Biology Department 3291 Ingresoll
Phone: (718)951-5396

B) I did not work with any other students on this project.

C) I have not met Professor Theodore Muth nor Professor Jessica Joyner before this project, and I have not worked with them on any previous projects.

D) The general problem lies in the classification of large data. Professor Joyner works with many students as a Micro-Biology Professor on a project that requires students to fill out a survey that asks them about things such as their previous knowledge of biology and what they expect from the class. The second survey acts as a post class survey which is compares against the initial surveys. The problem comes in when it comes time to manage and analyze all the survey data so that the information can be used to make future classes better for both students and professors.

The solution to the problem lies in automation and organization. To classify large amounts of data and make it meaningful, the data needs to be organized so that it can be easily parsed. One of the biggest issues that I had while working on the project was dealing with 'dirty data' (improper inputs of data ie strings where numbers where expected, empty inputs) and no standard format for data sheets. The Excel spreadsheets had the columns labeled by what the question on the survey was regarding, so many times the title of the column would be a whole sentence ("AbleAskQuestionsGetHelpfulResponses"). Although this may be more helpful for the person inputting the data from student survey to data sheet, it makes organization and manipulation of data more difficult. Since the primary concern was the change in student response of the paired surveys, a simple fix around this was to label the questions from the first paired survey A(A1, A2,…,A25) and the second was B(B1, B2,…,B22). This may make inputting the data from the physical student survey to the data sheet a bit more tedious if the person inputting the data loses track of cells, but even this issue can be fixed with a supplemented sheet pairing the question to the corresponding survey and question number(B1 = "Clarification of a career path"). The major benefit to this kind of organization is that when processing the data through a script, I was able to specify the columns needed much easier.

Although the organization of data was a crucial part of the process, it was not solved in a single instance. I first realized that the data needed to be made into more concise format if I was going to be able to import and manipulate it. This is where the idea for column names with A#/B# questions came about. This allowed me to select the columns needed in R much easier than before.

```
Fall2015Pre <- read_excel("/AllSemesters.xlsx",sheet="Fall2015-PreSurveyData", ski
p = 2, na = "0")
There were 50 or more warnings (use warnings() to see the first 50)
```

The data being read from is the "Fall2015-PreSurveyData", the workbook is "AllSemesters.xlsx", skip is set to 2 so that 2 lines are skipped and then the data is read into the Fall2015Pre data frame, and the na = 0 stores a NA value as 0. This was how the data was read in, but as you can see by the next line in red, there was an issue with the data being read in.

```
> warnings()

Warning messages:
1: In read_xlsx_(path, sheet, col_names = col_names, col_types = col_types,  ... :
  [133, 30]: expecting numeric: got '1 and 4'
2: In read_xlsx_(path, sheet, col_names = col_names, col_types = col_types,  ... :
  [215, 62]: expecting numeric: got '1 and 5'
```

The issue of ‘dirty data’ came into effect here once I began to import the preferred columns:

("ProjectSemester","Campus","StudentName","Major","A1","A2","A3","A4","A5","A6",
"A7","A8","A9","A10","A11","A12","A13","A14","A15","A16","A17","A18",
"A19","A20","A21","A22","A23","A24","A25","B1","B2","B3","B4","B5","B6",
"B7","B8","B9","B10","B11","B12","B13","B14","B15","B16","B17","B18",
"B19","B20","B21","B22")

For many columns, the expected type was a value between 1 and 6 corresponding to a value in the survey. The value of 6 was “NA” or “Not applicable” for both surveys, survey A ranged from 1 = “No experience” to 5 = “Extensive experience or mastered this element”, and survey B ranged from 1 = “Strongly disagree” to 5 = “Strongly agree”. The reason for the data values input as “X and Y” is because the students did not always clearly indicate which value was their answer if they changed their mind or just made stray marks, so the input was at the discretion of the person inputting the data to the data sheet. The error made itself known as a problem when the comparison of mismatched types were being compared. I could not subtract a numeric from a string, so the cell value would end up being NA, leaving entire columns as NA. The reason for this loss of data in an entire column is because when the data is imported form Excel, the column type is defined by what type of data exists in the column. For example, if I imported the data for the Fall2015Pre survey as is, any column with dirty data in it would convert the everything in the column to a string rather than a numeric value.

```
> sapply(Fall2015Pre, class)
…
$A20                        $A21                      $A22
[1] "numeric"               [1] "character"           [1] "numeric"
…
```

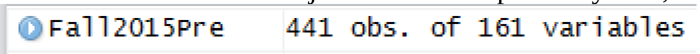
The sapply() method here find the class for every element in the object Fall2015Pre. The columns A20 and A22 are numeric because they contain only numbers, whereas A21 also contains numbers but also a character. To make sure the character is read into the object, the entire class type for that column is converted to “character”. The reason this distinction between types is crucial to data analysis is because if it is not the expected type, then the expected value will be incorrect and the data being worked on is incorrect by default. The mismatch of data types came in when comparing values of paired questions (A1 Post vs A1 Pre).

```
 > if(!is.na(PostDF[i,k]) && !is.na(PreDF[j,k]) && PostDF[[i,k]] < 6 && PreDF[[j,k]] < 6)
       PostDF[i,k] <- PostDF[i,k] - PreDF[j,k]
```

To resolve the issue so that the surveys could be compared, I created a function called CleanDF to be called after the data is imported from the spreadsheet. One line of code is called before this function is called to remove any empty rows in the data as a form of optimization.

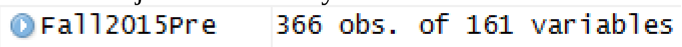
```
> Fall2015Pre <- Fall2015Pre[rowSums(is.na(Fall2015Pre)) != ncol(Fall2015Pre),]
> Fall2015Pre <- CleanDF(Fall2015Pre)

> CleanDF <- function(DFToClean){
      for(i in 1:length(QuestionNames)){
              DFToClean[[QuestionNames[i]]] <- as.numeric(DFToClean[[QuestionNames[i]]])
      }
      return(DFToClean)
}
```

Before the Fall2015Pre object is cleaned up of dirty data, it contains 441 rows.



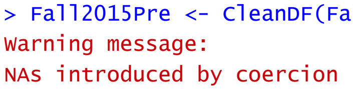
Fall2015Pre   441 obs. of 161 variables

After the line including rowsums is called, Fall2015Pre contains only 366 rows, which means that 75 empty rows were erased. This alone makes the data much easier to read through as empty rows are erased from the object because they are of no value.



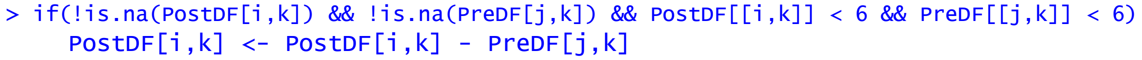
```
Fall2015Pre    366 obs. of 161 variables
```

The CleanDF function works by using coercion to force a data type. Since we are not sure of the actual value of an input like “1 and 3”, we cannot efficiently use that data in the analysis.



```
> Fall2015Pre <- CleanDF(Fall2015Pre)
Warning message:
NAs introduced by coercion
```

NA’s replace the value of the element in each column if they are not numeric. In turn, this cleans out data that is not numeric for that column and changes the data type for the column into a numeric. This means that we are able to compare values now from pre and post surveys since they are numeric or they are NA. In the case of NA, the function used to compare values checks to make sure that the values being compared are not NA (blank values) and they are less than 6 (6 = “Not Applicable” in the survey. This is not meaningful to the analysis).



```
> if(!is.na(PostDF[i,k]) && !is.na(PreDF[j,k]) && PostDF[[i,k]] < 6 && PreDF[[j,k]] < 6)
     PostDF[i,k] <- PostDF[i,k] - PreDF[j,k]
```

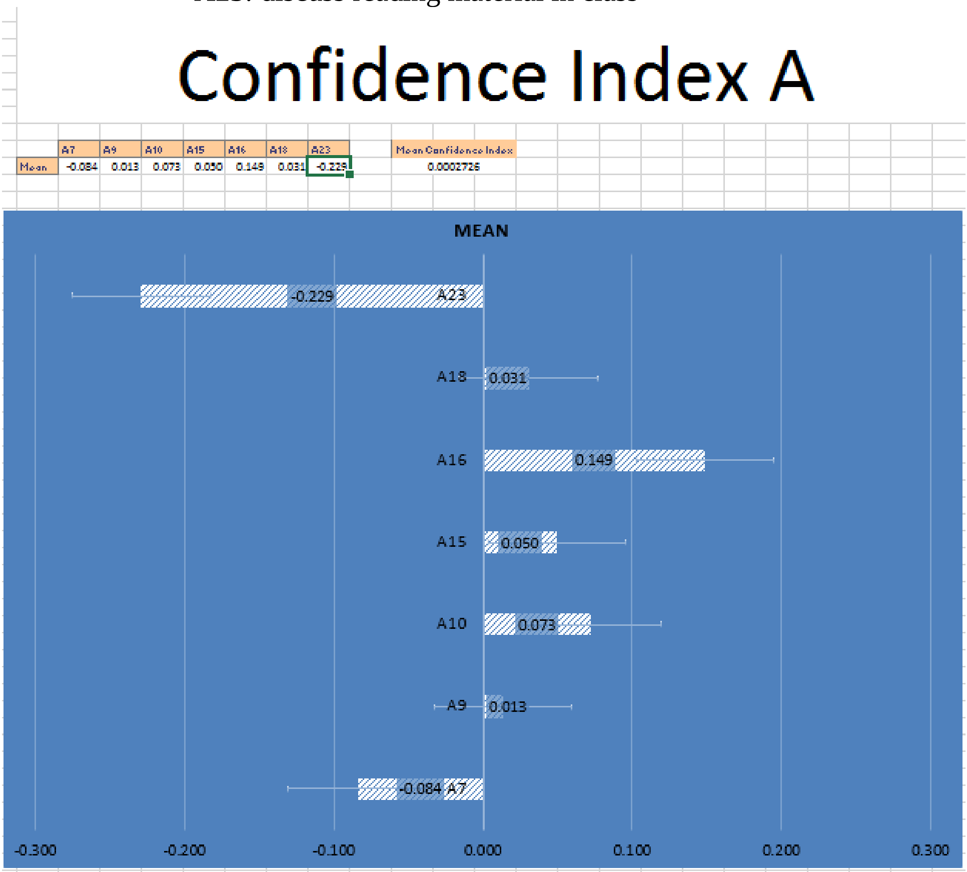
The bulk of the work involved cleaning up data, which unfortunately meant that a lot of inputs were thrown out due to error while inputting or due to invalid inputs from the students. Although there was a large amount of data lost, the data that remained was data that was cleaned up and useable for comparison.

The next step was to use the data that was cleaned up to present meaningful analysis from the values. An important part of presenting the analyzed data included how the data was presented. Because certain values were sometimes negative or low values, certain visual representations of the data harbored immediate bias. If someone was to look at a 2 quadrant vertical bar graph (+/-) and saw bars in the negative part of the chart, it would immediately give off the idea of “BAD” or “LOSS”. This bias, although not conscious, is not always incorrect, but in the case of the data was used was incorrect. A low value would represent a loss, but a sliding scale. For example, the questions in survey A ask about the students skill level in a certain context: A19 -> A25 are things not covered in the course, so the Post survey would report how their gain in skill while the Pre survey would report their skill coming into the class. This will inevitably yield a negative compared mean value because if the student was familiar with the skills not covered in the course (Reading from textbooks, maintaining a lab notebook, in class tests) before, and there was no gain in experience, comparing the change goes from a high value to a low (5 – 1 in the worst case). Although this issue may come off as an insignificant nuance, it is highly important. It is much more difficult to change someone’s mind on a negative thought than it is to change it on a positive one, so if their initial reaction to presented data was a negative one, then it would affect how they understand the rest of the data. Below is an example of a confidence rating for compared values of a certain semester. Since the values are presented horizontally rather than vertically, the initial interpretation is less of a loss but rather more of a current standing. The questions in particular are ones that relate to student confidence based on the context of the question. Sets A and B have to be processed differently because their responses do not hold the same values (Experience level for A / Agree or Disagree for B). The questions for A are as follows:

If you are expected to do the following course elements…

- A7: work individually
- A9: work in small groups
- A10: become responsible for a part of the project
- A15: present results orally

- A16: present results in written papers or reports
- A18: critique the work of other students
- A23: discuss reading material in class

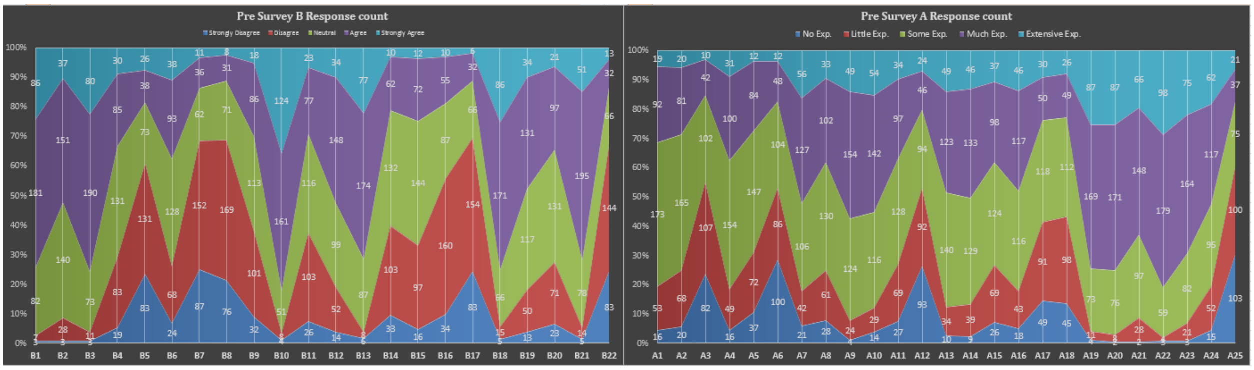


Confidence Index A

	A7	A9	A10	A15	A16	A18	A23		Mean Confidence Index
Mean	-0.084	0.013	0.073	0.050	0.149	0.031	-0.229		0.0002726

There are a few important things to note about this representation. The values are the mean of the compared responses for one data set, the data sets in particular still contain ‘dirty data’, and the data set contains many 0’s. As touched on before, there are questions that will yield negative values due to the nature of the course. Students did not listen to lectures, read textbooks, work on problem sets, take tests in class, discuss reading materials in class (A23 listed above), maintain lab notebooks, or do any computer modeling. If a student had experience with these before the course, there would be a ‘negative’ change because they gained no experience in these things in the course work. The plot and whisker chart along with the color scheme are deliberate because it gives insight into what the range was for the values, what the mean value was, the blue color is immediately associated with positivity as opposed to a color like red, and they are all represented horizontally to show a standing rather than a loss (vertical).

Another example of the importance of char specifics is shown by frequency of response. These two 100% area graphs clearly and effectively depict how the data variance changed from more sporadic results to a more consistent one. The 100% aspect of the graph indicates a set of values out of 100% and this rigidness of the graph gives a better scope for someone to understand the values. The graphs shown in the top photo Pre Survey response frequencies of ranges 1-5 (No experience to extensive experience), and the bottom graphs show the Post Survey response frequencies of ranges 1-5.

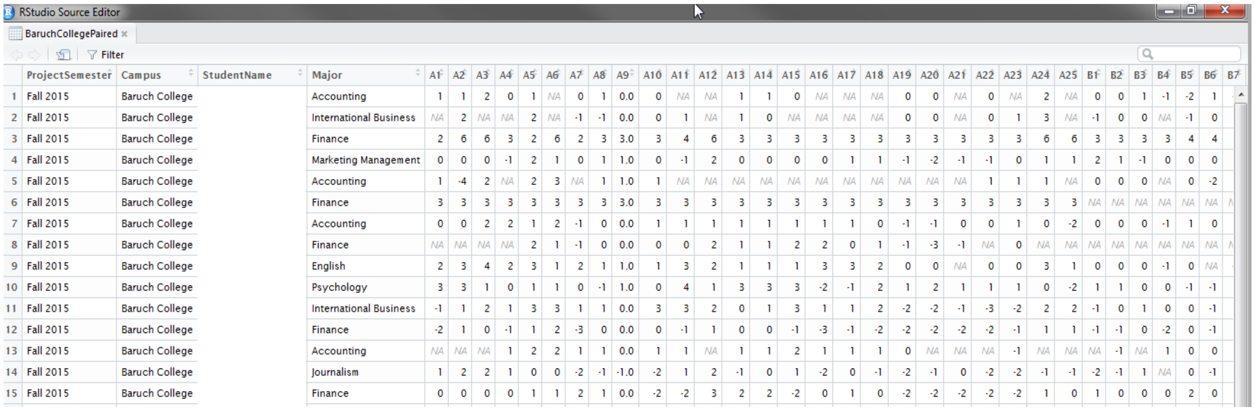


https://www.linkedin.com/in/dvucetaj/

The values in the second graph set (Post Surveys) show less jagged lines and more even ones, which shows us that students were more aware of their answers collectively AFTER having taken the course than before. This type of visualization is does not have the same effect as vertical bar graphs because they do not flow into one another, and a line graph does not effectively show the difference in response frequency as effectively as the 100% area graph does. These little differences make a huge difference when talking about data and presenting data. If the representation of that data is made more accessible to more individuals with less of an immediate visual bias, then talking about the information presented becomes simpler.

The project was completed to the point where the data was compared and the values were presented in a clear, concise, and standardized format so that manipulation of the data was much easier and much more effective. As seen below, with the names of students withheld, this is an example of one data frame.

```
BaruchCollegePre <- addSurveys(Fall2015Pre,"Baruch College",BaruchCollegePre)
>   BaruchCollegePre <- addSurveys(Spring2016Pre,"Baruch College",BaruchCollegePre)
>   BaruchCollegePre <- addSurveys(Fall2016Pre,"Baruch College",BaruchCollegePre)
> BaruchCollegePost <- addSurveys(Fall2015Post,"Baruch College",BaruchCollegePost)
>   BaruchCollegePost <- addSurveys(Spring2016Post,"Baruch College",BaruchCollegePost)
>   BaruchCollegePost <- addSurveys(Fall2016Post,"Baruch College",BaruchCollegePost)
> BaruchCollegePaired <- CreatePairedSheet(BaruchCollegePost, BaruchCollegePre)
```



	ProjectSemester	Campus	StudentName	Major	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16	A17	A18	A19	A20	A21	A22	A23	A24	A25	B1	B2	B3	B4	B5	B6	B7
1	Fall 2015	Baruch College		Accounting	1	1	2	0	1	NA	0	1	0.0	0	NA	NA	1	1	0	NA	NA	NA	0	0	NA	0	NA	2	NA	0	0	1	-1	-2	1	
2	Fall 2015	Baruch College		International Business	NA	2	NA	NA	2	NA	-1	-1	0.0	0	1	NA	1	0	NA	NA	NA	NA	0	0	NA	0	1	3	NA	-1	0	0	NA	-1	0	
3	Fall 2015	Baruch College		Finance	2	6	6	3	2	6	2	3	3.0	3	4	6	3	3	3	3	3	3	3	3	3	3	3	6	6	3	3	3	3	4	4	
4	Fall 2015	Baruch College		Marketing Management	0	0	0	-1	2	1	0	1	1.0	0	-1	2	0	0	0	0	1	1	-1	-2	-1	-1	0	1	1	2	1	-1	0	0	0	
5	Fall 2015	Baruch College		Accounting	1	-4	2	NA	2	3	NA	1	1.0	1	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	1	1	1	NA	0	0	0	NA	0	-2	
6	Fall 2015	Baruch College		Finance	3	3	3	3	3	3	3	3	3.0	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	NA	NA	NA	NA	NA	NA	
7	Fall 2015	Baruch College		Accounting	0	0	2	2	1	2	-1	0	0.0	1	1	1	1	1	1	1	1	0	-1	-1	0	0	1	0	-2	0	0	0	-1	1	0	
8	Fall 2015	Baruch College		Finance	NA	NA	NA	NA	2	1	-1	0	0.0	0	0	2	1	1	2	2	0	1	-1	-3	-1	NA	0	NA	NA	NA	NA	NA	NA	NA	NA	
9	Fall 2015	Baruch College		English	2	3	4	2	3	1	2	1	1.0	1	3	2	1	1	1	3	3	2	0	0	NA	0	0	3	1	0	0	0	-1	0	NA	
10	Fall 2015	Baruch College		Psychology	3	3	1	0	1	1	0	-1	1.0	0	4	1	3	3	3	-2	-1	2	1	2	1	1	1	0	-2	1	1	0	0	-1	-1	
11	Fall 2015	Baruch College		International Business	-1	1	2	1	3	3	1	1	0.0	3	3	2	0	1	3	1	1	2	-2	-2	-1	-3	-2	2	2	-1	0	1	0	0	-1	
12	Fall 2015	Baruch College		Finance	-2	1	0	-1	1	2	-3	0	0.0	0	-1	1	0	0	-1	-3	-1	-2	-2	-2	-2	-2	-1	1	1	-1	-1	0	-2	0	-1	
13	Fall 2015	Baruch College		Accounting	NA	NA	NA	1	2	2	1	1	0.0	1	1	NA	1	1	2	1	1	1	0	NA	NA	NA	-1	NA	NA	NA	-1	NA	1	0	0	
14	Fall 2015	Baruch College		Journalism	1	2	2	1	0	0	-2	-1	-1.0	-2	1	2	-1	0	1	-2	0	-1	-2	-1	0	-2	-2	-1	-1	-2	-1	1	NA	0	-1	
15	Fall 2015	Baruch College		Finance	0	0	0	0	1	1	2	1	0.0	-2	-2	3	2	2	-2	0	1	0	-2	-2	-2	-2	-2	1	0	1	0	0	0	2	0	

This data does not contain any strings, not does it contain any empty rows, but the loss of data due to improper inputs is apparent. The values show above are sorted with the CreatePairedValues function after the addSurveys function filters through data frames containing all the data for all students in a semester but filters the data by matching for a certain campus.

E) The things that I was not able to complete with the assignment is an automation of the entire process from filled data sheet in Excel to visualizations. My goal was to create a script that the user could manipulate to include certain parameter (ie sort by major or sort by campus, etc) so the data they wanted was readily available. A big reason for this was that this was the first time I have worked extensively in Excel and the first time I have ever used R. I explored a few other languages such as VBA, SQL, and

Python, but worked within R and Excel since that is what Prof. Joyner and Prof. Muth use and there was not enough time for a full integration into new languages/systems in the time frame, also considering the fact that I was learning them for the first time too.

Since this was my first time working with data sets and data analysis, I did not realize the importance of clean data. As mentioned before, if a column contained a single string instead of a number, the entire column would be defined as "character" rather than "numeric". For this reason, among others, much of my time was spent researching methods of cleaning up data and standardizing formats for the data. If I did not define exactly what I was working with and what I wanted to do with it, then the program would fail to process.

I was not able to create visualizations for the compared data because I only reached the point of usable data at the end of the semester. This in part is due to starting the code a bit late and due to the amount of work it took to get the usable data. I have a good understanding of which data sets should be represented by which visualization and why, but it came together too late into the process.

Finally, the algorithm that the whole compared data set relies on was wildly inefficient. The actual function worked properly and did what it needed to do, but because the algorithm ran on an order similar to:

Time = ((number of rows in post surveys) x (number of rows in pre surveys)) x (number of columns to be compared)

The time it took to process 366 entries was close to a minute, as shown below. This algorithm at this point does function but it is not properly scalable. Since the students name is not the only discernable value for the row, and each set contains both student name and the semester they attended the course in that college, a hashmap would be very helpful here. The values would be sorted initially by semester so that all groupings of semesters would exist between a certain number of rows. Once the value changes, that semester would be assigned to the hashmap for the starting address of that semester of students. This is similar to how sequential memory allocation is used in an operating system to store data.

```
> BrooklynCollegePre <- addSurveys(Fall2015Pre,"Brooklyn College",BrooklynCollegePre)
>   BrooklynCollegePre <- addSurveys(Spring2016Pre,"Brooklyn College",BrooklynCollegePre)
>   BrooklynCollegePre <- addSurveys(Fall2016Pre,"Brooklyn College",BrooklynCollegePre)
>   BrooklynCollegePre <- BrooklynCollegePre[order(BrooklynCollegePre$StudentName),]
> BrooklynCollegePost <- addSurveys(Fall2015Post,"Brooklyn College",BrooklynCollegePost)
>   BrooklynCollegePost <- addSurveys(Spring2016Post,"Brooklyn College",BrooklynCollegePost)
>   BrooklynCollegePost <- addSurveys(Fall2016Post,"Brooklyn College",BrooklynCollegePost)
> system.time(BrooklynCollegePaired <- CreatePairedSheet(BrooklynCollegePost, BrooklynCollege
Pre))
   user  system elapsed 
  52.96    0.02   53.37 
```

Finally, the last thing I would want to accomplish would be a total automation of the process, namely object creation and naming. The process would involve creating a list of college campus names, parsing that list and creating a series of objects with the "pre" or "post" suffix based on the suffix of the imported data set. This would involve string manipulation and the use of regular expressions.

H) This project has been quite an interesting task to say the least. I have learned more from this project than I have from almost any class I have taken solely because of the amount of times I messed something up because some kind of error popped up that I was unfamiliar with, but needed to solve to get my function running. If I was not able to debug properly, I was unable to create a working set of data and the whole project would be moot. I learned a number of things that I see as very useful now that I overlooked in the past with projects in other classed. I want to mention three that really stood out to me and why.

First off was the use of a timeline. In the "Intro to software engineering class" taught by Professor. Yarmish we learned about the lifecycle of a program, simply put. We learned about the maintenance of software and the creation of software, and it all relied heavily on timing. Because the semester was a busy one, I was not as prudent about the time management as I could have been, nor did I really understand the importance until I had a working data set. In the past I have worked on projects outside of school that utilized a Scrum framework (Agile programing), which keeps track of the tasks at hand (ToDo, InProgress, Completed) through Jira (a task managing online software). Because I was the only one responsible for my tasks, I needed to make sure that everything was within a timeline and what was completed and when it was completed.

Second was the use of documentation. Proper documentation helped me understand the code I wrote and helped Prof. Joyner understand it when I relayed it to her. Without proper documentation, I would have been unable to understand parts of my code that I had just learned to use.

Third and finally, was the use of testing. I did not create test cases for the code and I regret doing so because every time I wanted to test the function, I had to load up every function or data set in R again and again. Had I implemented a test set, I could just run that instead of sorting through the script and running one line at a time.

All in all I think that the class was a great experience into the field of data analysis and it was a pleasure working with a Professor who cared about the work, was supportive, and was present in every step of the way.