humanyze

Statistical Analysis Task

Human Interaction and Performance Analysis

Index

Section 0: References

1) “Python Cheatsheets”, https://drive.google.com/folderview?id=0ByIrJAE4KMTtaGhRcXkxNHhmY2M
2) “Comparison with SQL”, http://pandas.pydata.org/pandas-docs/stable/comparison_with_sql.html
3) Seaborn Statistical Visualization, https://stanford.edu/~mwaskom/software/seaborn/index.html
4) “Useful Pandas Snippets”, http://www.swegler.com/becky/blog/2014/08/06/useful-pandas-snippets/
5) “Python and Pandas: Top 10”, http://manishamde.github.io/blog/2013/03/07/pandas-and-python-top-10/
6) ‘Statistical analysis made easy in Python with SciPy and pandas DataFrames’, http://www.randalolson.com/2012/08/06/statistical-analysis-made-easy-in-python/
7) StackOverflow, http://www.stackoverflow.com
8) “Introduction to Data Science”, https://www.udacity.com/course/intro-to-data-science--ud359

Section 1: Introduction

We have three files namely:

1) Bluetooth.csv with columns **sender.id**, **detected.id**, **date.time** which tells you about the interaction between the **28 employees** using the sociometric badges developed at MIT labs. Even though the technology supports more ways of detecting the interaction between employees, we are given the interactions using Bluetooth in **bluetooth.csv**

2) Participants.csv with columns **badge.id**, **team** contains the information about the teams of employees i.e. whether the employee belongs to **Configuration** or **Pricing**.

3) **Transactions.csv** with columns **assigned.to**, **duration**, **complexity**, **assign.date**, **close.date**.
This file is important because this is going to give us the performance about employees. complexity values for employees can be **Basic**, **Advanced**, **Complex**.

We need to answer the question "**How do employee interaction patterns relate to performance**?"

By the data we have, we can form the following Hypothesis to answer the above question:

Hypothesis:

H0: Interaction does not affect significantly the performance of employees.

Ha: Interaction significantly affect performance, i.e. Employee who interact more, have increased performance or People who interact more have decreased performance.

To perform the above hypothesis, we need to have some data which has simpler form of performance measure and interaction measure. We will do some data aggregation and data exploration to come to a simpler form of data which can be used to do statistical tests.

Section 1: Data exploration, modification and aggregation

Lets start with exploring the three files we have and see if we what we can do to make it more useful for the statistical tests and plotting the graphs.

1. Bluetooth.csv

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 167288 entries, 0 to 167287
Data columns (total 3 columns):
sender.id       167288 non-null int64
detected.id     167288 non-null int64
date.time       167288 non-null object
dtypes: int64(2), object(1)
memory usage: 5.1+ MB
```

Fig 1: BluetoothDF.info()

	sender.id	detected.id
count	167288.000000	167288.000000
mean	243.252343	246.582086
std	69.479742	64.614103
min	82.000000	82.000000
25%	258.000000	258.000000
50%	272.000000	267.000000
75%	281.000000	280.000000
max	298.000000	298.000000

Fig 2: BluetoothDF.describe()

	sender.id	detected.id	date.time
0	273	82	3/26/2007 14:13
1	280	82	3/26/2007 14:14
2	258	82	3/26/2007 14:14
3	281	82	3/26/2007 14:14
4	280	82	3/26/2007 14:15

Fig 3: BluetoothDF.head()

From the above snapshots it seem logical to modify the DataFrame to include **team** of the sender as well as team of detecter along with other fields. So we modify the BluetoothDF to modifiedBluetoothDF which includes **team** of sender as well as detecter.

***Process**:* Joined two tables Bluetooth.csv and Participants.csv on sender.id=badge.id to get the team of sender. Again joined Bluetooth.csv and Participants.csv on detected.id=badge.id to get the team of detected person.
Removed the duplicates to get the modifiedBluetoothDF

	sender.id	detected.id	date.time	sender_team	detected_team
0	273	82	3/26/2007 14:13	Configuration	Configuration
1	273	82	3/26/2007 14:27	Configuration	Configuration
2	273	82	3/26/2007 14:29	Configuration	Configuration
3	273	82	3/26/2007 14:32	Configuration	Configuration
4	273	82	3/26/2007 14:33	Configuration	Configuration

Fig 4: moodifiedBluetoothDF.head()

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 167288 entries, 0 to 167287
Data columns (total 5 columns):
sender.id        167288 non-null int64
detected.id      167288 non-null int64
date.time        167288 non-null object
sender_team      167288 non-null object
detected_team    167288 non-null object
dtypes: int64(2), object(3)
memory usage: 7.7+ MB
```

From the above table we can see that number of rows before the merge and after the merge is same i.e. 167288, this shows we have properly merged the team names into the modifiedBluetoothDF

Fig 5: modifiedBluetoothDF.info()

	sender.id	detected.id
count	167288.000000	167288.000000
mean	243.252343	246.582086
std	69.479742	64.614103
min	82.000000	82.000000
25%	258.000000	258.000000
50%	272.000000	267.000000
75%	281.000000	280.000000
max	298.000000	298.000000

Fig 6: modifiedBluetoothDF.describe()

2. Participants.csv

	badge.id	team
0	82	Configuration
1	99	Configuration
2	101	Configuration
3	104	Configuration
4	105	Configuration

Fig 8:ParticipantsDF.head()

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 28 entries, 0 to 27
Data columns (total 2 columns):
badge.id    28 non-null int64
team        28 non-null object
dtypes: int64(1), object(1)
memory usage: 672.0+ bytes
```

Fig7: ParticipantsDF.info()

Fig 9: ParticipantsDF.describe()

	badge.id
count	28.000000
mean	230.714286
std	77.430905
min	82.000000
25%	215.500000
50%	265.500000
75%	280.250000
max	298.000000

3) Transactions.csv

The process we did to join team to the bluetoothDF is applied to the transactionDF to make

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 911 entries, 0 to 910
Data columns (total 6 columns):
assigned.to              911 non-null int64
duration (in minutes)    911 non-null int64
complexity               911 non-null object
assign.date              911 non-null object
close.date               911 non-null object
team                     911 non-null object
dtypes: int64(2), object(4)
memory usage: 49.8+ KB
```

Fig 10: modifiedTransactionDF.info()

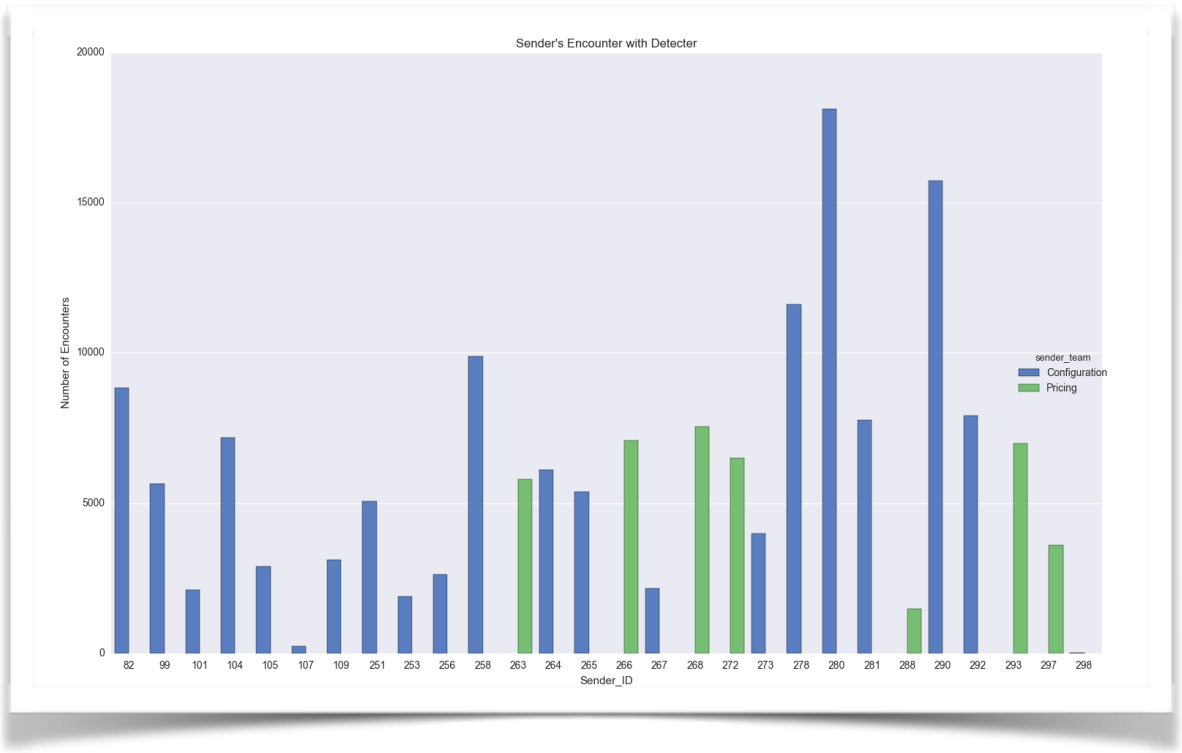
	assigned.to	duration (in minutes)
count	911.000000	911.000000
mean	249.501647	925.911087
std	66.106638	2210.875157
min	82.000000	1.000000
25%	258.000000	44.000000
50%	272.000000	137.000000
75%	293.000000	992.500000
max	298.000000	28241.000000

Fig 11: modifiedTransactionDF.describe()

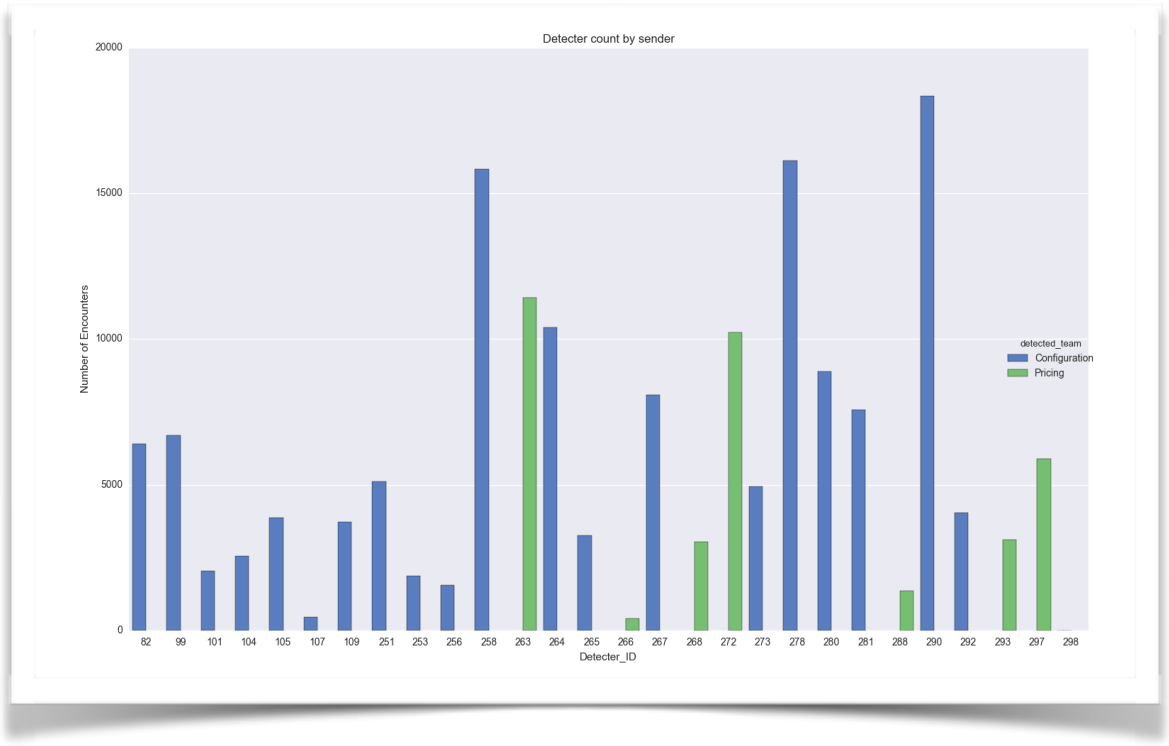
	assigned.to	duration (in minutes)	complexity	assign.date	close.date	team
0	267	221	Complex	3/23/2007 7:09	3/23/2007 10:50	Configuration
1	267	206	Basic	3/23/2007 11:06	3/23/2007 14:32	Configuration
2	267	369	Basic	3/26/2007 7:57	3/26/2007 14:06	Configuration
3	267	286	Basic	3/26/2007 10:00	3/26/2007 14:46	Configuration
4	267	55	Basic	3/27/2007 7:17	3/27/2007 8:12	Configuration

Fig 12: modifiedTransactionDF.head()

I used the seaboarn python library to plot the graphs.

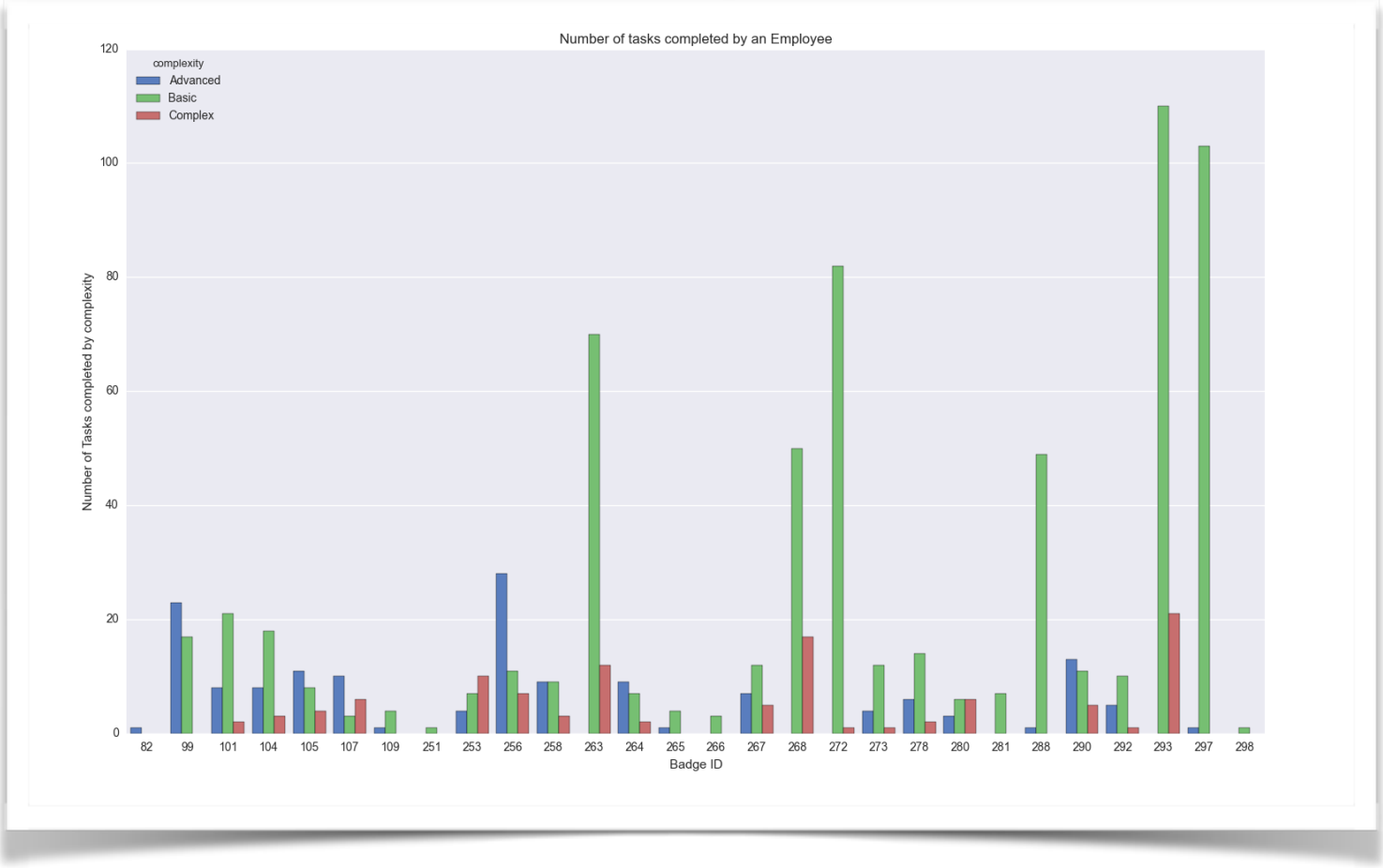


Plot 1: Sender Encounter with Detecter (Group by sender.id)

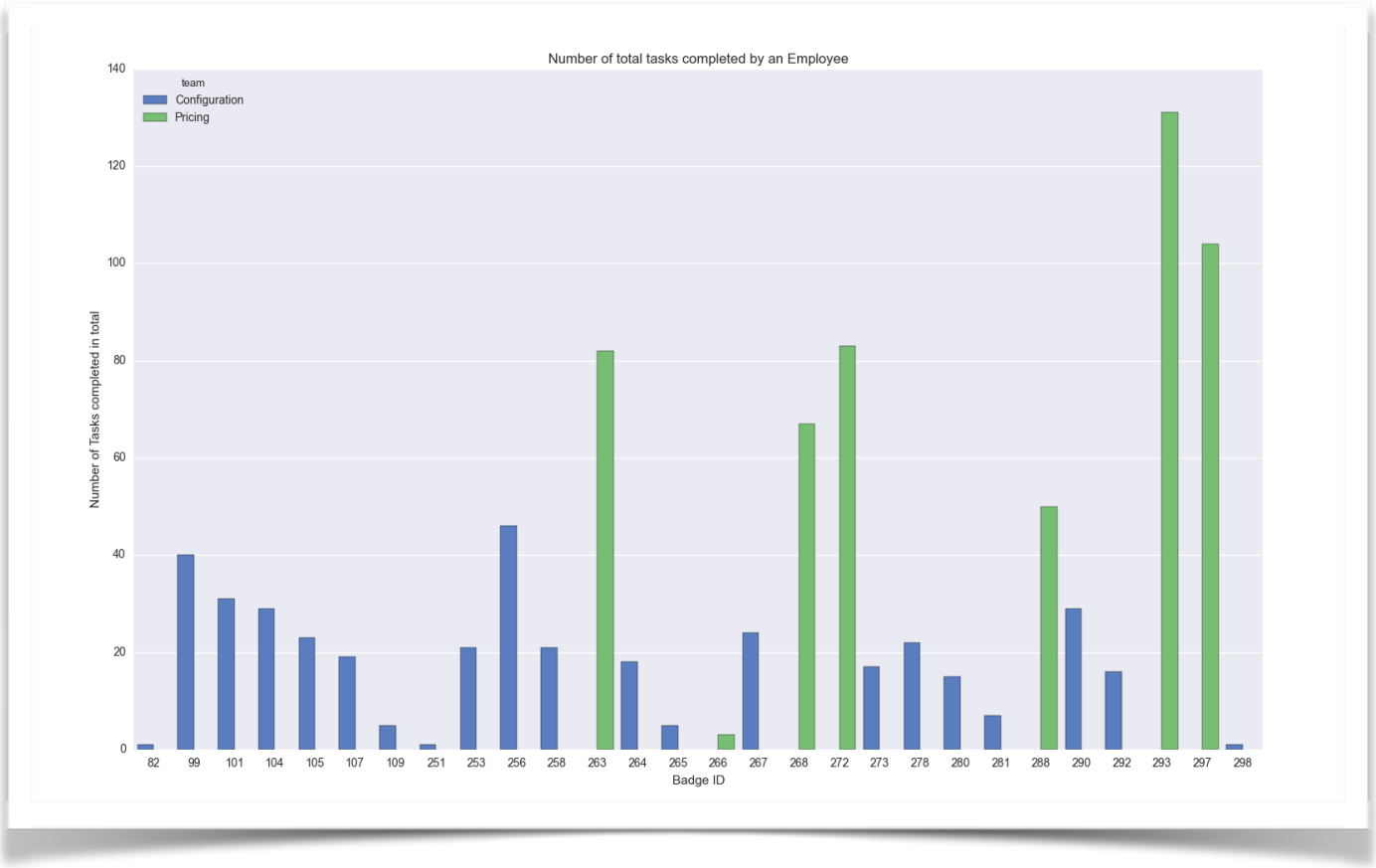


Plot 2: Detecter Count by sender (Group by detecter)

As mentioned in the problem statement, the Sociometric badge is not so perfect. So the count of sender.id is not equal to count of detected.id from Plot 1 and Plot 2.



Plot 3: Total Number of Tasks Completed By an employee



Plot 4: Number of tasks completed by Employee based on complexity

Section 2: Notable changes in Dataset and assumptions

The final dataset needed to be a simple which, we can pass into the python method for stats test. So I did the following changes and assumption:

1) Added a new **score** column to modifiedTransactionDF based on the **complexity** column. So I filled score value of 3,2 and 1 for **Complex, Advanced and Basic** respectively
2) Summed up the scores by **Assigned.to** for example: employee with id=267 did two **Complex** task, three **Advanced** task and five **Basic** tasks, his score will be 2*3+3*2+5*1=17
3) Counted the interactions grouped by **sender.id** in one data frame and and grouped by **detected.id** in another. Since for some BadgeID's sender.id count may differ from detected.id count, I have taken **a**verage of **s**ender.id **c**ount and **d**etected.id count for every **Badge.ID,** that column is named as **ASCD** in the final DataFrame.
4) I have joined the two table for Interaction and Performance on Badge ID to be used for statistical test.

The final DataFrame looks like below:

	ID	score	coes	coed	ACSD
0	82	2	8833	6414	7623.5
1	99	63	5658	6699	6178.5
2	101	43	2113	2043	2078.0
3	104	43	7179	2552	4865.5
4	105	42	2890	3872	3381.0

Fig 13: JoinedInteractionPerformance.head()

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 21 entries, 0 to 27
Data columns (total 5 columns):
ID       21 non-null int64
score    21 non-null int64
coes     21 non-null int64
coed     21 non-null int64
ACSD     21 non-null float64
dtypes: float64(1), int64(4)
memory usage: 1008.0 bytes
None
```

Fig 14: JoinedInteractionPerformance.info()

	ID	score	coes	coed	ACSD
count	21.000000	21.000000	21.000000	21.000000	21.000000
mean	214.904762	31.238095	6109.238095	6276.857143	6193.047619
std	83.574461	22.516449	4765.455066	5189.495113	4655.784998
min	82.000000	1.000000	15.000000	4.000000	9.500000
25%	107.000000	7.000000	2619.000000	2552.000000	3381.000000
50%	258.000000	32.000000	5380.000000	4951.000000	5094.000000
75%	278.000000	43.000000	7916.000000	8091.000000	7671.000000
max	298.000000	88.000000	18110.000000	18335.000000	17028.000000

Fig 15:JoinedInteractionPerformance.describe()

5) Columns names

ID is the Badge ID of the employee

score is the performance of the Employee calculated by the complexity and the number of tasks completed by the employee
coes is the count of encounters of sender.
coed is the count of encounters of detecter
ACSD is the (**coed**+**coes**)/2

6) I also removed, the rows with **Team** "**Pricing"**, since we are considering the Performance of configuration only.

Section 3: Statistical tests and Hypothesis Test

Coming back to **Hypothesis test**

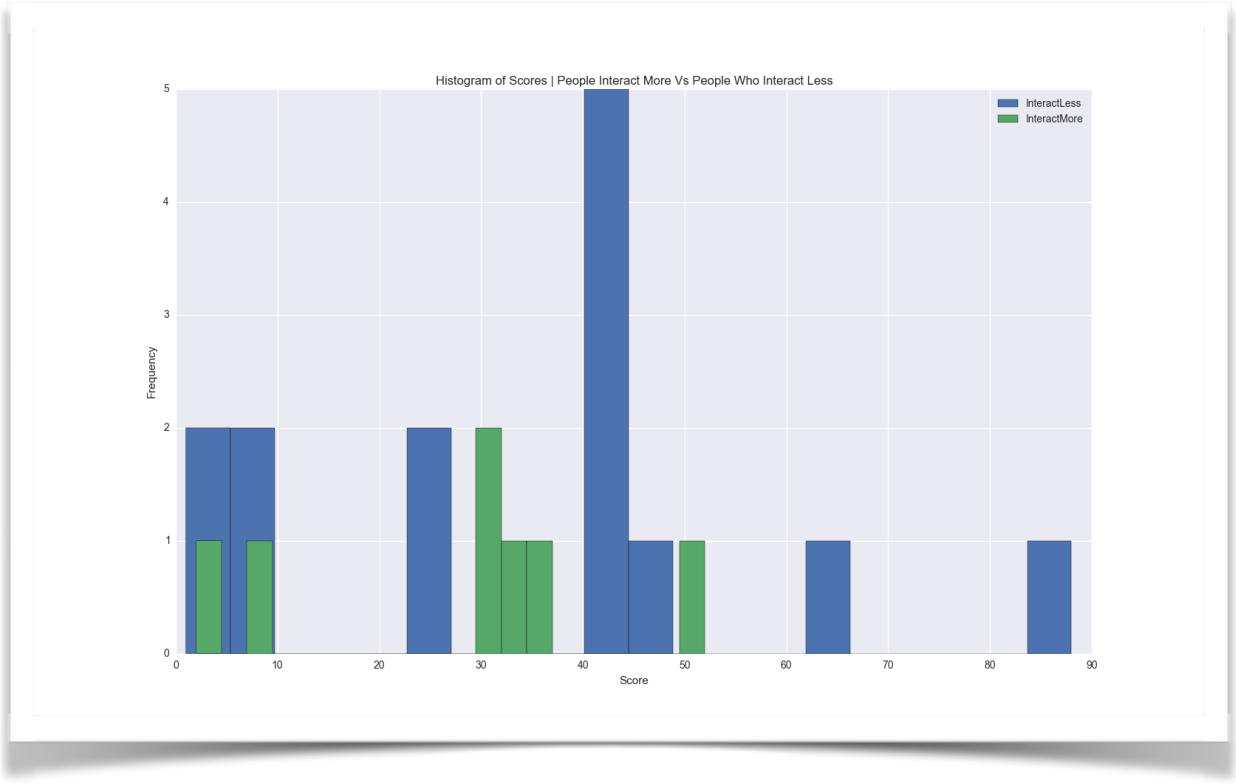
H0: Interaction doesn't have affect on performance.

Ha: Interaction does affect performance, i.e. Interacting people have increase in their performance or interacting people have decrease in their performance.

Mean of the performance score is: 31.238095
Mean of the performance_score of employee interacting more than average is: 27.142857
Mean of the performance_score of people who interacting less than average people : 33.285714
p-critical: 0.05 (alpha level)

I am using two tailed test with p-critical of 0.05 (alpha level). We will have to check if the distribution is normal for the populations we are working on and depending on that we will decide the test to be used.

We have choice of welch's t-test and ManWhitneyU test. Welch's t-test requires the populations to be normally distributed.
We will plot the histograms of the two population and see how it looks.



Plot 5: Histogram of the two populations

As the two populations need to be normally distributed, which in this case is not. So we use ManWhitneyU test instead of using Welch's T test.

Assumptions for ManwhitneyU test:

- The population group considered for comparison should be either ordinal or continuous but not normally distributed
- The number of observations>20 and you have independent samples rank

The reported p-value is for a one-sided hypothesis, to get the two-sided p value multiply the returned p-value by 2.

Results:

- p value: 0.31358293578247343*2= 0.627 (p value is multiplied by 2 since its a two tailed test)
- PeopleWhoInteractLess mean: 33.285714
- PeopleWhoInteractMore mean: 27.142857
- U statistics: 42.0

Interpretation of the test:

- Even though the mean score of PeopleWhoInteractLess (33.29) is greater than mean score of PeopleWhoInteractMore(27.14), the ManWhitney U test at p-critical 0.05 does not show significant change in the score of performance, since p-value(0.627) is greater than p-critical (0.05).

- Based on ManWhitneyU test result, we retain the null hypothesis i.e. interaction does not have effect on performance.

Conclusion:

Mean score of people who interact more: 27.142857
Mean score of people who interact less: 33.285714
I also did Mann Whitney U test which has p-value(0.627) is greater than p critical (0.05).

Even though, the mean score of people who interact more is less than people who interact less, the ManWhitneyU test does not show significant change in the score of performance people who interact less and people who interact less.

Section 4: Shortcoming and possible improvements

1) As mentioned in section 2: Notable changes in Dataset and assumptions, we assumed the performance score depends on the complexity and the number of tasks completed by an employee. We can consider assign.date and close.date to calculate the time taken by an employee and consider that in calculating the score.

For example: If a person completes a **Complex** task within 3 hours we can add one more point to his score, if a person completes **Advanced** task within 2 hours we can add one more point, and **Basic task** within 1 hour we can add one more point.

This would lead to more precise result for the null hypothesis to reject or retain.

2) We used ManWhitney U test to compare two independent populations. ManWhitney U test works better if the number of observations are more than 20. We have 21 observations, the results would be better than this if we have more number of observations.