# Data Challenge 2-Copy2

February 27, 2019

## 1 Data Challenge 2

Aim: Investigate why there has been a decrease in user engagment in the app Yammer

User engagement is defined as users login into the app, this definition comes from the SQL query that is used to generate the graph in the question.

Hypothesis: Users are on holiday (it is the summer) Test: Look at users locations - are they logging in from different countries less frequently?

```
In [1]: import pandas as pd
        import numpy

In [2]: #read in lookup time data to set up my index

        df = pd.read_csv('dimension_rollup_periods.csv')

In [3]: df_1007=df.loc[df['period_id'] == 1007]

In [4]: df_1007.loc[:,'pst_end']=pd.to_datetime(df_1007['pst_end'], dayfirst=True)
        df_1007.loc[:,'pst_start']=pd.to_datetime(df_1007['pst_start'], dayfirst=True)

/anaconda3/envs/insight/lib/python3.7/site-packages/pandas/core/indexing.py:543: SettingWithCopy
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#
  self.obj[item] = s


In [5]: week1=pd.to_datetime('28/04/2014 00:00', dayfirst=True)

        weeks=[week1]
        for i in list(range(16)):
            weeki=weeks[i]
            weekn=weeki + pd.to_timedelta(7,'d')
            weeks.append(weekn)

In [6]: #only include weeks that are relevant
        df_1007=df_1007.loc[df_1007['pst_end'].isin(weeks)]
```

```
In [7]: df_1007.head()

Out[7]:       period_id           time_id  pst_start    pst_end         utc_start  \
        1734       1007  28/04/2014 00:00 2014-04-21 2014-04-28  21/04/2014 07:00
        1741       1007  05/05/2014 00:00 2014-04-28 2014-05-05  28/04/2014 07:00
        1748       1007  12/05/2014 00:00 2014-05-05 2014-05-12  05/05/2014 07:00
        1755       1007  19/05/2014 00:00 2014-05-12 2014-05-19  12/05/2014 07:00
        1762       1007  26/05/2014 00:00 2014-05-19 2014-05-26  19/05/2014 07:00

                       utc_end
        1734  28/04/2014 07:00
        1741  05/05/2014 07:00
        1748  12/05/2014 07:00
        1755  19/05/2014 07:00
        1762  26/05/2014 07:00

In [ ]: #lets look at events now

In [8]: df2=pd.read_csv('yammer_events.csv')

In [9]: df2['occurred_at']=pd.to_datetime(df2['occurred_at'], dayfirst=True)

In [ ]: #when do the events in our data set start and stop?

In [10]: df2['occurred_at'].min()

Out[10]: Timestamp('2014-05-01 00:54:00')

In [11]: df2['occurred_at'].max()

Out[11]: Timestamp('2014-08-31 23:03:00')

In [ ]: #seems to be before and after the window we are interested in so cut data set down

In [12]: df2 = df2[df2.occurred_at > weeks[0]]
         df2 = df2[df2.occurred_at <= weeks[-1]]

In [13]: def lookup_time(t):
             match= (df_1007['pst_start'] <= t) & (df_1007['pst_end'] > t)
             t_id=df_1007['time_id'][match]
             if len(t_id) ==0:
                 print (t)
             return pd.to_datetime(t_id.values[0])

         df2['occurred_wk']=df2['occurred_at'].apply(lookup_time)

In [15]: #lets groupby location and count unique locations to see if get an increase in n.o of c

         grouped=df2.groupby(['occurred_wk'])['location'].nunique().reset_index()
         grouped.sort_values(by=['occurred_wk'])
```

Out[15]:

| | occurred_wk | location |
|---|---|---|
| 0 | 2014-02-06 | 47 |
| 1 | 2014-04-08 | 47 |
| 2 | 2014-05-05 | 47 |
| 3 | 2014-05-19 | 47 |
| 4 | 2014-05-26 | 47 |
| 5 | 2014-06-16 | 47 |
| 6 | 2014-06-23 | 47 |
| 7 | 2014-06-30 | 47 |
| 8 | 2014-07-07 | 47 |
| 9 | 2014-07-14 | 47 |
| 10 | 2014-07-21 | 47 |
| 11 | 2014-07-28 | 47 |
| 12 | 2014-08-18 | 47 |
| 13 | 2014-09-06 | 47 |
| 14 | 2014-11-08 | 47 |
| 15 | 2014-12-05 | 47 |

Test fails, maybe hypothesis fails.
I need to do more investigative work but have run out of time.

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: