Assignment_2_EH

October 17, 2018

1 Assignment 2

1.0.1 MACS 3000, Dr. Evans

1.0.2 Ellen Hsieh

```
In [1]: import numpy as np
        from sklearn import linear_model
        import pandas as pd
        import matplotlib.pyplot as plt
        from sklearn.metrics import mean_squared_error, r2_score
```

1.0.3 1. Imputing age and gender

(a) My proposed strategy to impute age and gender into BestIncome.txt: First, we build two linear regression models respect to age and gender based on total income and weight in dataset SurvIncome.txt respectively. Then, we use those two models to predict the value of age and gender in Bestincone.txt. However, there is no feature called total income in BestIncome.txt, therefore, we can create the value of total income by adding labor and capital income in BestIncome.txt. Finally, we use the total income and weight in BestIncome.txt to calculate the value of age and gender for Bestlincome.txt based on those two regression models. The equations are as following:

$$Age_i = \beta_0 + \beta_1 Total\_income_i + \beta_2 Weight_i + \epsilon_i$$

$$Gender_i = \beta_0 + \beta_1 Total\_income_i + \beta_2 Weight_i + \epsilon_i$$

```
In [2]: # open and rename the column names of the imported files
        best = pd.read_csv('BestIncome.txt', sep=',', header=None)
        survey = pd.read_csv('SurvIncome.txt', sep=',', header=None)
        best = best.rename(index=int, columns={0:'lab_inc', 1:'cap_inc', 2:'hgt', 3: 'wgt'})
        survey = survey.rename(index=int, columns={0:'tot_inc',1:'wgt',2:'age',3:'gender'})
```

```
In [3]: best.describe()
```

```
Out[3]:             lab_inc       cap_inc           hgt           wgt
        count  10000.000000  10000.000000  10000.000000  10000.000000
        mean   57052.925133   9985.798563     65.014021    150.006011
        std     8036.544363   2010.123691      1.999692      9.973001
```

```
min     22917.607900   1495.191896     58.176154    114.510700
25%     51624.339880   8611.756679     63.652971    143.341979
50%     56968.709935   9969.840117     65.003557    149.947641
75%     62408.232277  11339.905773     66.356915    156.724586
max     90059.898537  19882.320069     72.802277    185.408280
```

In [4]:
```
survey.describe()
```

Out[4]:
```
            tot_inc          wgt          age      gender
count   1000.000000  1000.000000  1000.000000  1000.00000
mean   64871.210860   149.542181    44.839320     0.50000
std     9542.444214    22.028883     5.939185     0.50025
min    31816.281649    99.662468    25.741333     0.00000
25%    58349.862384   130.179235    41.025231     0.00000
50%    65281.271149   149.758434    44.955981     0.50000
75%    71749.038000   170.147337    48.817644     1.00000
max    92556.135462   196.503274    66.534646     1.00000
```

(b) Here is where I'll use my proposed method from part (a) to impute variables.

In [5]:
```
# create the total income value by adding labor income and capital income
tot_inc_best = best.lab_inc + best.cap_inc
x_best = np.column_stack((tot_inc_best, best.wgt))
```

In [6]:
```
# with the common features: tot_inc and wgt
# age ~ tot_inc and wgt
x = np.column_stack((survey.tot_inc, survey.wgt))
y_age = survey.age.values
y_gender = survey.gender.values

# simple linear regression: age ~ tot_inc + wgt
regr_age = linear_model.LinearRegression()
regr_age.fit(x, y_age)

# logistic regression: gender ~ tot_inc + wgt
regr_gender = linear_model.LogisticRegression()
regr_gender.fit(x, y_gender)
```

Out[6]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
          penalty='l2', random_state=None, solver='liblinear', tol=0.0001,
          verbose=0, warm_start=False)
```

In [7]:
```
# compute the predicted age and gender based on te information in best
pred_age = regr_age.predict(x_best)
pred_gender = regr_gender.predict(x_best)
```

In [8]:
```
# simple append the predictions onto the best dataset
imputed_best = np.column_stack((best, pred_age, pred_gender))
```

(c) Here is where I'll report the descriptive statistics for my new imputed variables.

```
In [9]: # convert the numpy array into a pandas dataframe
        new_best = pd.DataFrame({'lab_inc':imputed_best[:,0],'cap_inc':imputed_best[:,1],
                                 'hgt':imputed_best[:,2],'wgt':imputed_best[:,3],
                                 'age':imputed_best[:,4],'gender':imputed_best[:,5]})
```

```
In [10]: new_best.describe()
```

```
Out[10]:              lab_inc        cap_inc           hgt           wgt           age  \
         count  10000.000000   10000.000000  10000.000000  10000.000000  10000.000000
         mean   57052.925133    9985.798563     65.014021    150.006011     44.890828
         std     8036.544363    2010.123691      1.999692      9.973001      0.219150
         min    22917.607900    1495.191896     58.176154    114.510700     43.976495
         25%    51624.339880    8611.756679     63.652971    143.341979     44.743776
         50%    56968.709935    9969.840117     65.003557    149.947641     44.886944
         75%    62408.232277   11339.905773     66.356915    156.724586     45.038991
         max    90059.898537   19882.320069     72.802277    185.408280     45.703819

                      gender
         count  10000.000000
         mean       0.471700
         std        0.499223
         min        0.000000
         25%        0.000000
         50%        0.000000
         75%        1.000000
         max        1.000000
```

(d) Correlation matrix for the now six variables

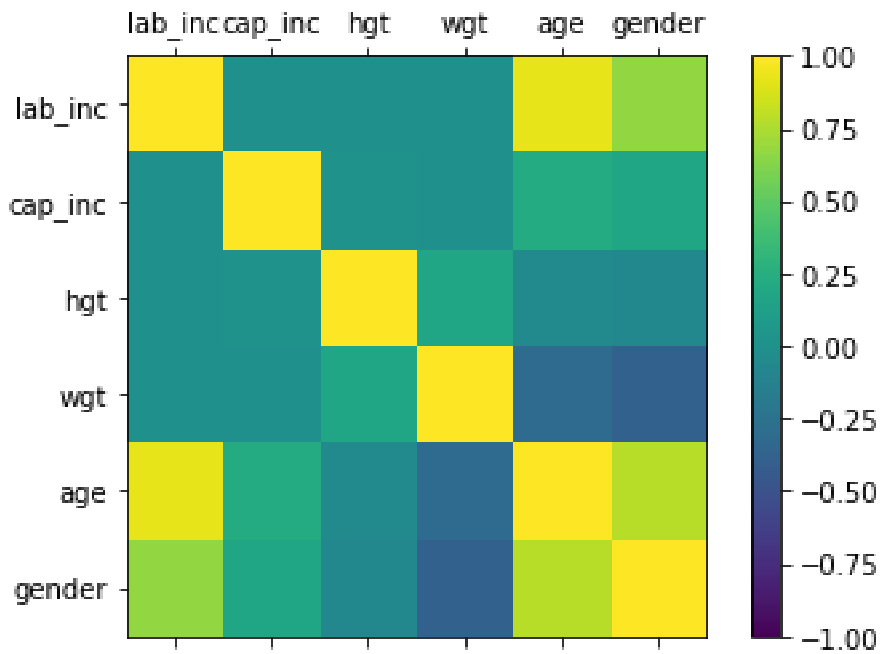
```
In [11]: # Correlation matrix form
         corr = new_best.corr()
         corr.style.background_gradient()
```

```
Out[11]: <pandas.io.formats.style.Styler at 0x1a162981d0>
```

```
In [12]: # Correction Matrix Plot
         def corr_plot(df):

             names = df.columns
             N = len(names)

             correlations = df.corr()
             fig = plt.figure()
             ax = fig.add_subplot(111)
             cax = ax.matshow(correlations, vmin=-1, vmax=1)
             fig.colorbar(cax)
             ticks = np.arange(0,N,1)
```

```
        ax.set_xticks(ticks)
        ax.set_yticks(ticks)
        ax.set_xticklabels(names)
        ax.set_yticklabels(names)
        plt.show()
```

In [13]:
```
corr_plot(new_best)
```



1.0.4 2. Stationarity and data drift

(a) Estimate by OLS and report coefficients

In [14]:
```
# open and rename the column names of the imported files
inc_intel = pd.read_csv('IncomeIntel.txt', sep=',', header=None)
inc_intel = inc_intel.rename(index=int, columns={0:'grad_year', 1:'gre_qnt', 2:'salary
```

In [15]:
```
# Define Outcome and Independent Variables
outcome  = 'salary_p4'
features = ['gre_qnt']

X, y = inc_intel[features], inc_intel[outcome]
```

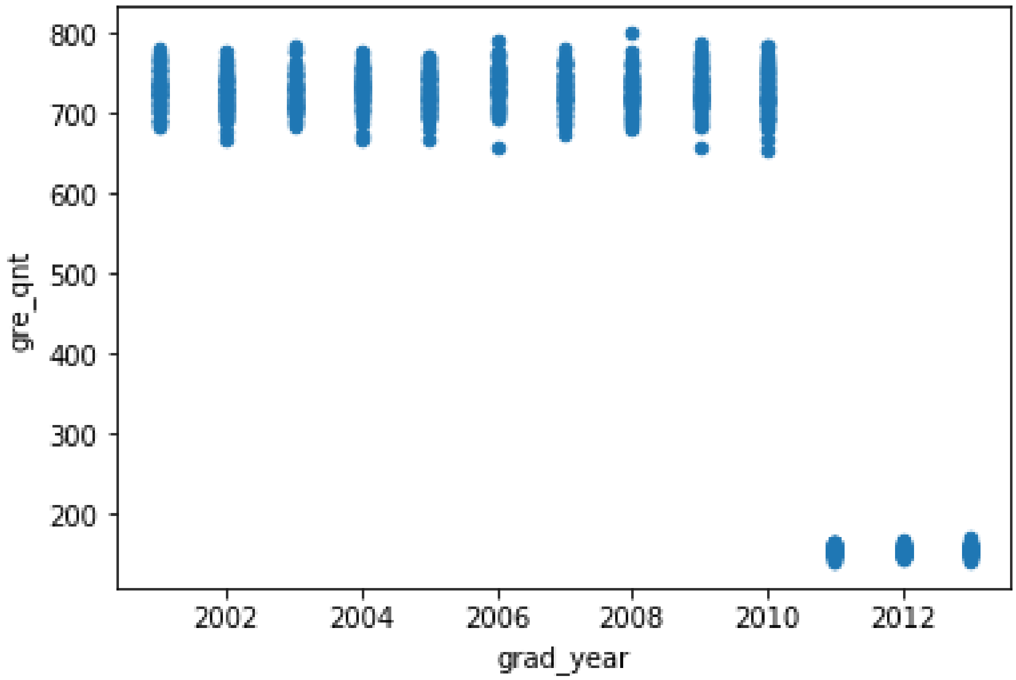
In [16]:
```
# run ols regression and report the result
import statsmodels.api as sm

X_vars = X[['gre_qnt']]
```

```
X_vars = sm.add_constant(X_vars, prepend=False)
X_vars.head()

m = sm.OLS(y, X_vars)

res = m.fit()
print(res.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:              salary_p4   R-squared:                       0.263
Model:                            OLS   Adj. R-squared:                  0.262
Method:                 Least Squares   F-statistic:                     356.3
Date:                Tue, 16 Oct 2018   Prob (F-statistic):           3.43e-68
Time:                        23:33:29   Log-Likelihood:                -10673.
No. Observations:                1000   AIC:                         2.135e+04
Df Residuals:                     998   BIC:                         2.136e+04
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
gre_qnt      -25.7632      1.365    -18.875      0.000     -28.442     -23.085
const       8.954e+04    878.764    101.895      0.000    8.78e+04    9.13e+04
==============================================================================
Omnibus:                        9.118   Durbin-Watson:                   1.424
Prob(Omnibus):                  0.010   Jarque-Bera (JB):                9.100
Skew:                           0.230   Prob(JB):                       0.0106
Kurtosis:                       3.077   Cond. No.                     1.71e+03
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.71e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

(b) Create a scatterplot of GRE score and graduation year.

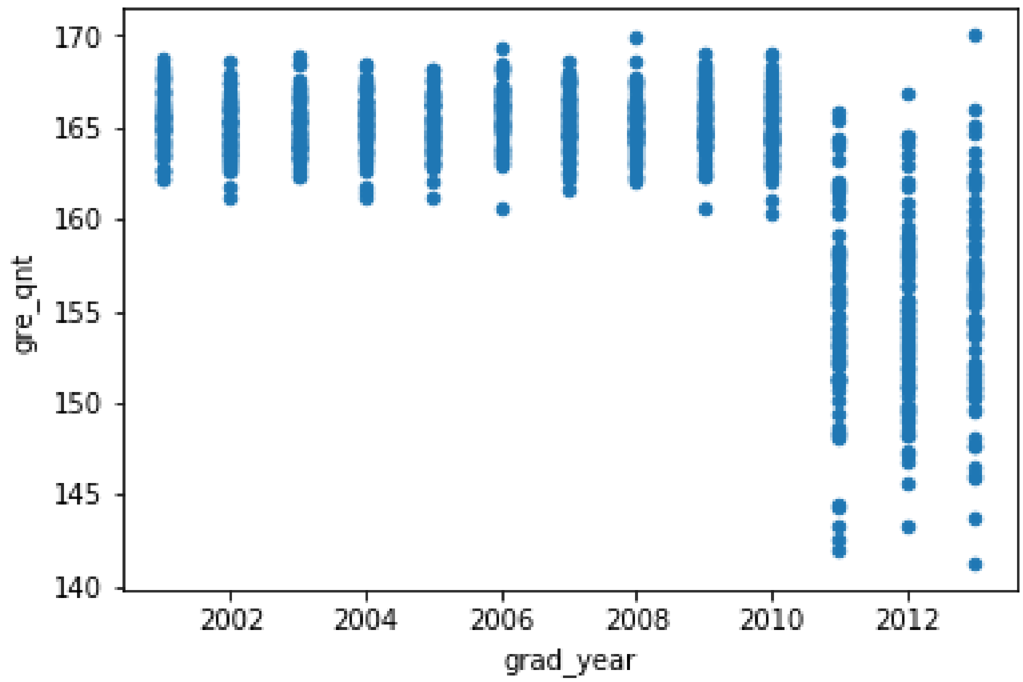
In [17]:
```
# make a scatterplot for gre score and graduation year
grad_year = inc_intel['grad_year']
gre_qnt = inc_intel['gre_qnt']
inc_intel.plot(x='grad_year', y='gre_qnt', kind='scatter')
plt.show()
```



The problem here is that there is a obvious jump down on this scatterplot due to the different scales of GRE score. Before 2011, the score scale for GRE quatative was 200-800 and after that it became 130-170. Therefore, if we use this dataset to build a regression model to predict the salary, it would distort the result of linear regression. In order to solve this problem, we can convert all the GRE scores to the latter scale(130-170), so that it could become consistent. And then, we could get a more precise regression model based on the modified dataset. (Note: The conversion used is not based on percentile so the value might not be precise enough).

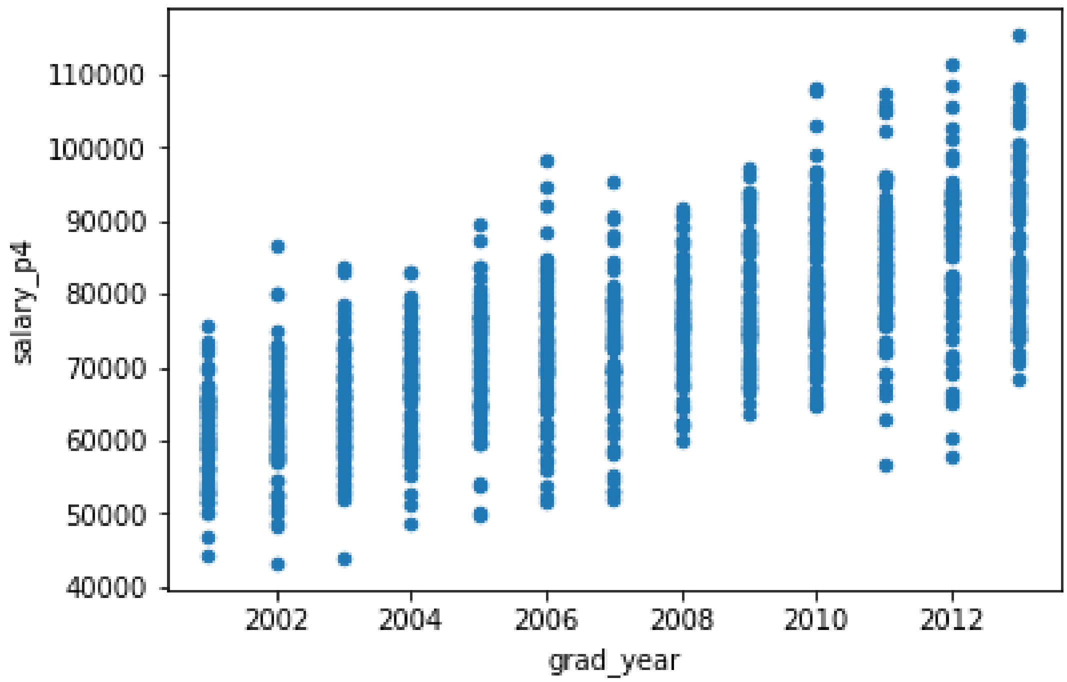
```
In [18]: # convert gre_qnt score
         converted_gre_qnt = inc_intel.gre_qnt[:]
         for i in range(len(inc_intel)):
             if inc_intel.grad_year[i] <= 2010.0:
                 converted_gre_qnt[i] = 130+ ((inc_intel.gre_qnt[i]-200)*40/600)
             else:
                 pass
```

```
In [19]: # a new scatterplot for converted gre score
         grad_year = inc_intel['grad_year']
         gre_qnt = converted_gre_qnt
         inc_intel.plot(x='grad_year', y='gre_qnt', kind='scatter')
         plt.show()
```



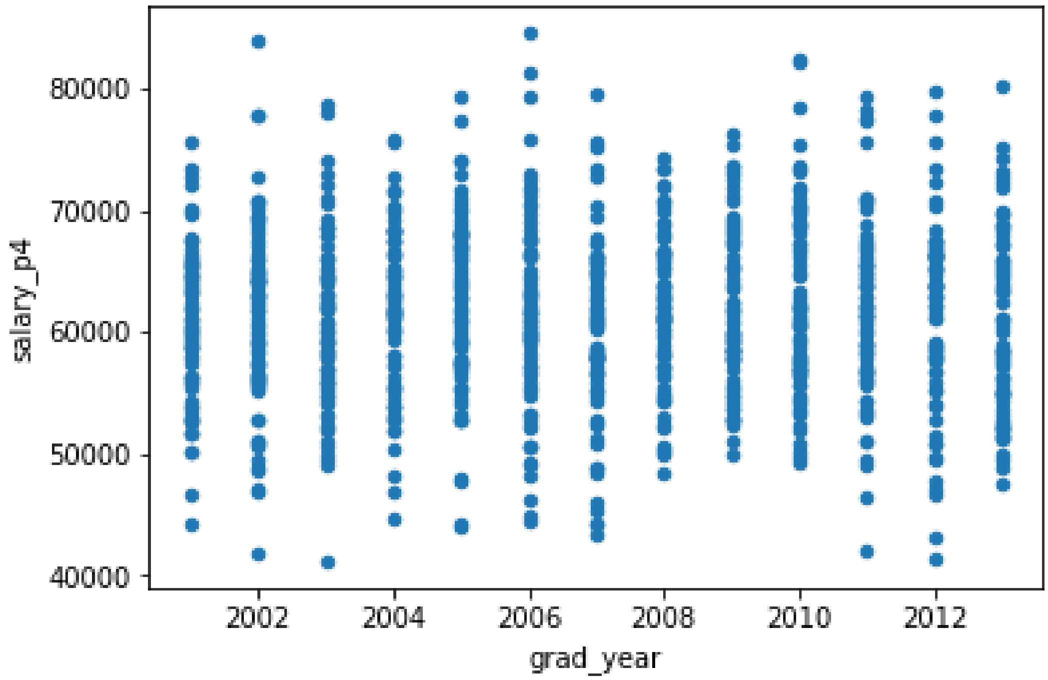
(c) Create a scatterplot of income and graduation year

```
In [20]: # create a scatterplot for income and graduation year
         grad_year = inc_intel['grad_year']
         salary_p4 = inc_intel['salary_p4']
         inc_intel.plot(x='grad_year', y='salary_p4', kind='scatter')
         plt.show()
```



The problem here is that the income is increasing over years. This is because we didn't take the inflation rate into consideration. To solve this problem, we can convert all the incomes into the present value by divided them by the inflation rate regarding to different year.

```
In [21]: # calculate the average income by year
         avg_inc_by_year = inc_intel['salary_p4'].groupby(inc_intel['grad_year']).mean().values

         # calculate the average growth rate
         avg_growth_rate = ((avg_inc_by_year[1:] - avg_inc_by_year[:-1]) / avg_inc_by_year[:-1]

In [22]: # calcualte the present value of all the incomes
         new_salary_p4 = salary_p4[:]

         for i in range(len(salary_p4)):
             if inc_intel.grad_year[i] == 2001.0:
                 new_salary_p4[i] = inc_intel.salary_p4[i]
             else:
                 new_salary_p4[i] = inc_intel.salary_p4[i] / ((1 + avg_growth_rate)**(inc_intel

In [23]: # a new scatterplot for new income value
         grad_year = inc_intel['grad_year']
         salary_p4 = new_salary_p4
         inc_intel.plot(x='grad_year', y='salary_p4', kind='scatter')
         plt.show()
```



(d) Re-estimate coefficients with updated variables.

```
In [24]: # add the modified features to inc_intel
         inc_intel['converted_gre_qnt'] = converted_gre_qnt
         inc_intel['new_salary_p4'] = new_salary_p4

         # define the outcome and features for regression model
         outcome  = 'new_salary_p4'
         features = ['converted_gre_qnt']

         X, y = inc_intel[features], inc_intel[outcome]

         # run the regression for new variables
         X_vars = X[['converted_gre_qnt']]
         X_vars = sm.add_constant(X_vars, prepend=False)
         X_vars.head()

         m = sm.OLS(y, X_vars)

         res = m.fit()
         print(res.summary())
```

```
                            OLS Regression Results
==============================================================================
```

```
Dep. Variable:          new_salary_p4   R-squared:                       0.000
Model:                            OLS   Adj. R-squared:                 -0.001
Method:                 Least Squares   F-statistic:                   0.05257
Date:                Tue, 16 Oct 2018   Prob (F-statistic):              0.819
Time:                        23:33:30   Log-Likelihood:                -10291.
No. Observations:                1000   AIC:                         2.059e+04
Df Residuals:                     998   BIC:                         2.060e+04
Df Model:                           1
Covariance Type:            nonrobust
=====================================================================================
                        coef    std err          t      P>|t|      [0.025      0.975]
-------------------------------------------------------------------------------------
converted_gre_qnt    -9.9681     43.474     -0.229      0.819     -95.279      75.343
const              6.304e+04   7083.395      8.900      0.000    4.91e+04    7.69e+04
==============================================================================
Omnibus:                        0.757   Durbin-Watson:                   2.026
Prob(Omnibus):                  0.685   Jarque-Bera (JB):                0.668
Skew:                           0.059   Prob(JB):                        0.716
Kurtosis:                       3.048   Cond. No.                     5.11e+03
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 5.11e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

The result of OLS regression for modified GRE score and salary shows that the new estimated coefficents became -9.9681 which is less negative than the former one. From this fact, we can know that GRE score now has less nagtive impact on salary. Before the coversion of GRE score and salary, the larger range of GRE score and abnormally distribution made GRE score have larger negative effect on salary. Therefore, after modified the dataset, the correlation between those variables become more accurate and the negative impact from GRE score on salary become less significant, which supports my hypothesis for the result of the modified regression model.

1.0.5 3. Assessment of Kossinets and Watts.

See attached PDF.

3. Assessment of Kossinets and Watts (2009)

(a) Why do people tend to group up with those of the same race?

(b) The data used in this research is collected from a large university in the U.S., including 30,396 undergraduate and graduate students, faculty, and staff. Three different databases comprise the data set: (1) the logs of email interaction within the university, (2) individual attributes such as status, gender, age, etc. (3) records of course registration. The time period span for one academic year. The description and definition of all the variables are listed in the appendix A.

(c) Since the error-correction strategies are based on longitudinal nature of data set, the potential problem for data cleansing process used in the research might wrongly interpolate some data which might not necessarily relate to time. For example, as the paper mentions that we could use backward interpolation for campus, dormitory, school, etc. However, those attributes are not necessarily related to time. For example, individual's preference might have greater influence on their choice. Therefore, the researchers could get a wrong or biased speculation due to wrongly interpolating the data, which could weaken the hypotheses proposed by the authors.

(d) One weakness of using e-mail logs to observe social relationship is that this data is incomplete. Due to the privacy consideration, the access of the content of emails is limited. Therefore, the authors cannot really analyze the content of those emails and develop a more precise relationship between emails and the characteristics of senders and receivers. The solution proposed by the authors in order to address the problem is informed consent procedure. Through exchanging some benefits, some users might be willing to provide their content and data for the researchers. Then, the researchers can better interpret the relations between those e-mail contents and the characteristics of the individuals.