SPEED DATING: WHAT PREDICTS A MATCH?

Miranda Remmer
DS-GA-23 Final Project

BACKGROUND

What makes a good first impression ?

What influences dating decisions?

How does gender influence selection?

How do perceptions play into choices?

THE DATA SET

Data pulled from a experimental speed dating event, from 2002-2004, sheds insight into dating decisions and outcomes, particularly with first impressions.

Participants were asked a series of questions throughout the experiment; (roughly) 1/3 in the signup survey, 1/3 during the experiment, and 1/3 at the conclusion of the experiment. To standardized observations, six attributes were used throughout: attractiveness, sincerity, intelligence, fun, ambition, and shared interests.

Scores assigned to each attribute at different points in the experiment can be used to predict matches.

THE PROBLEM

Can one’s perception of self predict his/her dating outcome?

Does this differ by gender?

THE HYPOTHESIS

Individuals with lower self-esteem (i.e. give themselves lower scores during self-evaluation), receive fewer dates (i.e. matches) than those who provide more positive ratings.

THE DATA BREAKDOWN

- 195 features
- 8378 observations

a few features -

- IID#
- Gender (F=0 | M=1)
- Dec: decision (yes = 1 | no = 0)
- Dec_o: decision of partner at event (yes = 1 | no = 0)
- Match: partner and subject both said 'yes' = 1
- Met_Count: number of people subject met with
- Match_es: number of matches a participant expects

THE DATA BREAKDOWN cont.

Variable	Description
attr	Attractive
sinc	Sincere
intel	Intelligent
fun	Fun
amb	Ambitious
shar	Shared Interests/Hobbies

Aside from demographic questions (e.g. age, religion) or questions that are only asked once (e.g. match_es); each survey question that uses the 6 attributes, are coded with numbers to indicate when during the survey they are administered.

Example:

A. How do you think others perceive you? (signup survey)

Rate each attribute on a scale of 1-10

attr3_1

sinc3_1

intel3_1

fun3_1

amb3_1

B. How do you think others perceive you? (after experiment)

Rate each attribute on a scale of 1-10

attr3_2

sinc3_2

intel3_2

fun3_2

amb3_2

THE DATA BREAKDOWN *cont.*

Variable	Description
attr	Attractive
sinc	Sincere
intel	Intelligent
fun	Fun
amb	Ambitious
shar	Shared Interests/Hobbies

C. What do you think the opposite sex looks for in a date? (signup survey)

Rate each attribute on a scale of 1-10

attr2_1

sinc2_1

intel2_1

fun2_1

amb2_1

D. Based on what you think the opposite sex looks for in a data, how do you measure up? (signup survey)

Rate each attribute on a scale of 1-10

attr3_1

sinc3_1

intel3_1

fun3_1

amb3_1

THE DATA BREAKDOWN cont.

Variable	Description
attr	Attractive
sinc	Sincere
intel	Intelligent
fun	Fun
amb	Ambitious
shar	Shared Interests/Hobbies

***Scorecards** are also administered to each participant asking him/her to rate the partners he/she meets with (filled out after each 'date'):*

- Rate their attributes on a scale of 1-10: (1=awful, 10=great)
  - Attractive
  - Sincere
  - Intelligent
  - Fun
  - Ambitious
  - Shared Interests/Hobbies
- Decision: yes/no
- Like: overall how much do you like this person (1=don't like at all, 10=like a lot)
- Prob: how probable do you think it is that this person will say 'yes' to you? (1=not probable, 10=extremely probable)

Partner's rating of subject also coded as a feature: attr_o, sinc_o, intel_o, prob_o....etc.

THE PROCESS

- Selecting which questions pertain to research question
  - Some questions are scaled at 100pts across all attributes; others at 10pts per attribute
- View & Clean the data
  - Logistic regression: classification (match vs. no match)
- IID#s – what to do…

```
df_raw
```

	iid	id	gender	idg	condtn	wave	round	position	positin1	order	...	attr3_3	sinc3_3	intel3_3	fun3_3	amb3_3	attr5_3
0	1	1.0	0	1	1	1	10	7	NaN	4	...	5.0	7.0	7.0	7.0	7.0	NaN
1	1	1.0	0	1	1	1	10	7	NaN	3	...	5.0	7.0	7.0	7.0	7.0	NaN
2	1	1.0	0	1	1	1	10	7	NaN	10	...	5.0	7.0	7.0	7.0	7.0	NaN
3	1	1.0	0	1	1	1	10	7	NaN	5	...	5.0	7.0	7.0	7.0	7.0	NaN
4	1	1.0	0	1	1	1	10	7	NaN	7	...	5.0	7.0	7.0	7.0	7.0	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
8373	552	22.0	1	44	2	21	22	14	10.0	5	...	8.0	5.0	7.0	6.0	7.0	9.0
8374	552	22.0	1	44	2	21	22	13	10.0	4	...	8.0	5.0	7.0	6.0	7.0	9.0
8375	552	22.0	1	44	2	21	22	19	10.0	10	...	8.0	5.0	7.0	6.0	7.0	9.0
8376	552	22.0	1	44	2	21	22	3	10.0	16	...	8.0	5.0	7.0	6.0	7.0	9.0
8377	552	NaN	1	44	2	21	22	2	10.0	15	...	8.0	5.0	7.0	6.0	7.0	9.0

8378 rows × 195 columns

THE PROCESS cont.

- Re-Selecting questions pertaining to research question
  - Get all attributes scaled 1-10pts
- CLEAN THE DATA ☺
  - Drop columns with incomplete data
  - NaN values? Two options..
    - Remove NaN values
    - Replace NaN values with ‘0’

Create functions

Example of dropping data: participant and partner didn't score – most likely didn't participate

In [8]:
```
df_nan_exp_ratings_all = subset_df_clean[(subset_df_clean.attr_o.isnull()) & (subset_df_clean.sinc_o.isnul
l()) & (subset_df_clean.fun_o.isnull()) &
                   (subset_df_clean.intel_o.isnull()) &  (subset_df_clean.amb_o.isnull()) & (subset_df_cle
an.shar_o.isnull()) & (subset_df_clean.attr.isnull()) & (subset_df_clean.sinc.isnull()) &
(subset_df_clean.fun.isnull()) &
                   (subset_df_clean.intel.isnull()) &  (subset_df_clean.amb.isnull()) & (subset_df_clean.s
har.isnull())]

len(df_nan_exp_ratings_all)
```

Out[8]: 132

In [9]:
```
len(subset_df_clean) #test
```

Out[9]: 8378

In [10]:
```
# Dropping Data
subset_df_clean = dropData(subset_df_clean, df_nan_exp_ratings_all)
```

In [11]:
```
len(subset_df_clean) #test
```

Out[11]: 8246

Example of dropping data: participant didn't score partner but said 'yes'

Look at data where subject didn't rate partner:

In [15]:
```
# Grabbing subject data with no scores for partner
##(partner has scored subject)
df_atr_null = subset_df_clean[(subset_df_clean.attr.isnull()) & (subset_df_clean.sinc.isnull())
                              & (subset_df_clean.fun.isnull()) & (subset_df_clean.intel.isnull())
                               &  (subset_df_clean.amb.isnull()) & (subset_df_clean.shar.isnull())]

# View data
df_atr_null[['iid', 'pid', 'like', 'prob', 'like_o', 'prob_o']]
```

Out[15]:

	iid	pid	like	prob	like_o	prob_o
245	23	53.0	NaN	NaN	7.0	6.0
705	50	32.0	7.0	7.0	6.0	3.0
711	50	38.0	7.0	8.0	7.0	6.0
712	50	39.0	8.0	NaN	7.0	5.0
920	67	58.0	NaN	NaN	6.0	7.0
...	...	...	...	...	...	...
8002	535	529.0	NaN	NaN	5.0	5.0
8003	535	530.0	NaN	NaN	8.0	5.0
8045	537	528.0	NaN	NaN	1.0	1.0
8067	538	528.0	NaN	NaN	2.0	2.0
8337	551	512.0	NaN	NaN	5.0	5.0

58 rows × 6 columns

In [16]:
```
# Look at data where subject didnt rate any attributes of parter but subject said 'yes'
##looking at values where dec = 1 from above DF (can drop)

df_atr_null_dec1 = df_atr_null[(df_atr_null.dec == 1)]

# View data
df_atr_null_dec1 [['iid', 'pid', 'dec', 'prob', 'like']]
#includes'like' = NaN but subject said 'yes' (dec = 1)

##remove; doesn't makse sense to late put '0' in attribute ratings
```

Out[16]:

	iid	pid	dec	prob	like
705	50	32.0	1	7.0	7.0
711	50	38.0	1	8.0	7.0
7216	488	476.0	1	NaN	NaN

In [17]:
```
subset_df_clean = dropData(subset_df_clean, df_atr_null_dec1)
```

Example of adding 0 for NaN values : participant scored for at least 1 of the other attributes but left others blank (esp. if dec=0)

Look where all values for attribute_ratings_other are null (aka partner's ratings)

```
In [58]: #pulling data for NaN values for partner rating of subject (all attribute_o ratings that are blank)
df_atr_o_null = subset_df_clean_edit[(subset_df_clean_edit.attr_o.isnull()) &
(subset_df_clean_edit.sinc_o.isnull())
                             & (subset_df_clean_edit.fun_o.isnull()) &
(subset_df_clean_edit.intel_o.isnull())
                             &  (subset_df_clean_edit.amb_o.isnull()) & (subset_df_clean_edit.shar_o.isnul
l())]

df_atr_o_null
```

Out[58]:

	iid	pid	gender	age	round	match	dec	dec_o	exphappy	expnum	...	intel	fun	amb	shar	attr_o	sinc_o	intel_o	fun
739	52	28.0	1	21.0	19	0	0	0	5.0	1.0	...	7.0	5.0	6.0	5.0	NaN	NaN	NaN	Nal
753	53	23.0	1	28.0	19	0	1	0	6.0	9.0	...	8.0	7.0	8.0	6.0	NaN	NaN	NaN	Nal
1755	122	NaN	1	22.0	10	0	0	0	6.0	10.0	...	8.0	8.0	8.0	8.0	NaN	NaN	NaN	Nal
1765	123	NaN	1	18.0	10	0	0	0	5.0	1.0	...	5.0	4.0	5.0	5.0	NaN	NaN	NaN	Nal
1775	124	NaN	1	22.0	10	0	1	0	6.0	10.0	...	7.0	6.0	7.0	7.0	NaN	NaN	NaN	Nal
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
8366	552	519.0	1	25.0	22	0	0	0	10.0	NaN	...	7.0	6.0	6.0	0.0	NaN	NaN	NaN	Nal
2451	178	187.0	0	35.0	10	0	0	0	5.0	NaN	...	7.0	5.0	0.0	0.0	NaN	NaN	NaN	Nal
5965	390	409.0	0	30.0	19	0	1	0	6.0	NaN	...	8.0	6.0	0.0	0.0	NaN	NaN	NaN	Nal
7854	529	535.0	0	22.0	22	0	1	0	5.0	NaN	...	5.0	5.0	0.0	0.0	NaN	NaN	NaN	Nal
911	66	59.0	1	29.0	10	0	0	0	5.0	3.0	...	0.0	0.0	0.0	0.0	NaN	NaN	NaN	Nal

65 rows × 52 columns

Created a function to drop the index of where all the feature info is blank; use this to fill other attribute ratings with null values as ‘0’

```
def cleanDF(main_df, df_all_feature_null, feature):
atr_nan = main_df[main_df[feature].isnull()] #look where partner did not rate subject on feature
atr_cleaned = atr_nan.drop(df_all_feature_null.index) #returning just rows with NaN data in feature but have other columns with data within feature set
 fillNaN([feature], atr_cleaned) ##calling function fillNaN to replace NaN values with 0
main_df = main_df.drop(atr_cleaned.index) #removing old values
main_df = pd.concat([main_df, atr_cleaned]) #adding cleaned data back into df
return main_df
```

Look where all values from the null_all_df where dec_o =o

In [59]:
```
#looking at values where dec_o = 0 from above DF; can make attribute ratings 0
df_atr_o_null_deco0 = df_atr_o_null[df_atr_o_null.dec_o ==0]
df_atr_o_null_deco0
```

Out[59]:

	iid	pid	gender	age	round	match	dec	dec_o	exphappy	expnum	...	intel	fun	amb	shar	attr_o	sinc_o	intel_o	fun
739	52	28.0	1	21.0	19	0	0	0	5.0	1.0	...	7.0	5.0	6.0	5.0	NaN	NaN	NaN	Nal
753	53	23.0	1	28.0	19	0	1	0	6.0	9.0	...	8.0	7.0	8.0	6.0	NaN	NaN	NaN	Nal
1755	122	NaN	1	22.0	10	0	0	0	6.0	10.0	...	8.0	8.0	8.0	8.0	NaN	NaN	NaN	Nal
1765	123	NaN	1	18.0	10	0	0	0	5.0	1.0	...	5.0	4.0	5.0	5.0	NaN	NaN	NaN	Nal
1775	124	NaN	1	22.0	10	0	1	0	6.0	10.0	...	7.0	6.0	7.0	7.0	NaN	NaN	NaN	Nal
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
8366	552	519.0	1	25.0	22	0	0	0	10.0	NaN	...	7.0	6.0	6.0	0.0	NaN	NaN	NaN	Nal
2451	178	187.0	0	35.0	10	0	0	0	5.0	NaN	...	7.0	5.0	0.0	0.0	NaN	NaN	NaN	Nal
5965	390	409.0	0	30.0	19	0	1	0	6.0	NaN	...	8.0	6.0	0.0	0.0	NaN	NaN	NaN	Nal
7854	529	535.0	0	22.0	22	0	1	0	5.0	NaN	...	5.0	5.0	0.0	0.0	NaN	NaN	NaN	Nal
911	66	59.0	1	29.0	10	0	0	0	5.0	3.0	...	0.0	0.0	0.0	0.0	NaN	NaN	NaN	Nal

65 rows × 52 columns

Within this DF, look at where like_o has no value; added a 'o' to NaN values

(no ratings for partner, decision = no, like _o = NaN)

In [59]:
```
#looking at values where dec_o = 0 from above DF; can make attribute ratings 0
df_atr_o_null_deco0 = df_atr_o_null[df_atr_o_null.dec_o ==0]
df_atr_o_null_deco0
```

Out[59]:

	iid	pid	gender	age	round	match	dec	dec_o	exphappy	expnum	...	intel	fun	amb	shar	attr_o	sinc_o	intel_o	fun
739	52	28.0	1	21.0	19	0	0	0	5.0	1.0	...	7.0	5.0	6.0	5.0	NaN	NaN	NaN	Nal
753	53	23.0	1	28.0	19	0	1	0	6.0	9.0	...	8.0	7.0	8.0	6.0	NaN	NaN	NaN	Nal
1755	122	NaN	1	22.0	10	0	0	0	6.0	10.0	...	8.0	8.0	8.0	8.0	NaN	NaN	NaN	Nal
1765	123	NaN	1	18.0	10	0	0	0	5.0	1.0	...	5.0	4.0	5.0	5.0	NaN	NaN	NaN	Nal
1775	124	NaN	1	22.0	10	0	1	0	6.0	10.0	...	7.0	6.0	7.0	7.0	NaN	NaN	NaN	Nal
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
8366	552	519.0	1	25.0	22	0	0	0	10.0	NaN	...	7.0	6.0	6.0	0.0	NaN	NaN	NaN	Nal
2451	178	187.0	0	35.0	10	0	0	0	5.0	NaN	...	7.0	5.0	0.0	0.0	NaN	NaN	NaN	Nal
5965	390	409.0	0	30.0	19	0	1	0	6.0	NaN	...	8.0	6.0	0.0	0.0	NaN	NaN	NaN	Nal
7854	529	535.0	0	22.0	22	0	1	0	5.0	NaN	...	5.0	5.0	0.0	0.0	NaN	NaN	NaN	Nal
911	66	59.0	1	29.0	10	0	0	0	5.0	3.0	...	0.0	0.0	0.0	0.0	NaN	NaN	NaN	Nal

65 rows × 52 columns

Function to Convert Values from Old DF to New Condensed DF:

In [12]:
```
def ConvertDF(DF, df):  #DF = old dataframe    #df = new data frame
    df['iid'] = DF.iid.unique()  #returning iid# (1 unique value per person - 1 row per subject)
    df['gender'] = getValueSet('gender', DF) #returning 1 row per iid with subject's gender
    df['age'] = getValueSet('age', DF) #returning 1 row per iid with subject's age
    df['met_count'] = getValueSet('met_count', DF) #returning 1 row per iid with met_count info (how many
 people each person met with)
    df['exphappy'] = getValueSet('exphappy', DF) #returning rating for exphappy per iid
    df['expnum'] = getValueSet('expnum', DF) #returning expnum per iid (1 value)
    df['match_es'] = getValueSet('match_es', DF)

    df['attr_iMeasUp_1'] = getValueSet('attr_iMeasUp_1', DF)
    df['sinc_iMeasUp_1'] = getValueSet('sinc_iMeasUp_1', DF)
    df['intel_iMeasUp_1'] = getValueSet('intel_iMeasUp_1', DF)
    df['fun_iMeasUp_1'] = getValueSet('fun_iMeasUp_1', DF)
    df['amb_iMeasUp_1']= getValueSet('amb_iMeasUp_1', DF)

    df['attr_iMeasUp_2'] = getValueSet('attr_iMeasUp_2', DF)
    df['sinc_iMeasUp_2'] = getValueSet('sinc_iMeasUp_2', DF)
    df['intel_iMeasUp_2'] = getValueSet('intel_iMeasUp_2', DF)
    df['fun_iMeasUp_2'] = getValueSet('fun_iMeasUp_2', DF)
    df['amb_iMeasUp_2']= getValueSet('amb_iMeasUp_2', DF)

    df['attr_oPercveMe_1'] = getValueSet('attr_oPercveMe_1', DF)
    df['sinc_oPercveMe_1'] = getValueSet('sinc_oPercveMe_1', DF)
    df['intel_oPercveMe_1'] = getValueSet('intel_oPercveMe_1', DF)
    df['fun_oPercveMe_1'] = getValueSet('fun_oPercveMe_1', DF)
    df['amb_oPercveMe_1'] = getValueSet('amb_oPercveMe_1', DF)

    df['attr_oPercveMe_2'] = getValueSet('attr_oPercveMe_2', DF)
    df['sinc_oPercveMe_2'] = getValueSet('sinc_oPercveMe_2', DF)
    df['intel_oPercveMe_2'] = getValueSet('intel_oPercveMe_2', DF)
    df['fun_oPercveMe_2'] = getValueSet('fun_oPercveMe_2', DF)
    df['amb_oPercveMe_2'] = getValueSet('amb_oPercveMe_2', DF)

    df['attr_iRateMe_exp'] = getValueSet('attr_iRateMe_exp', DF)
    df['sinc_iRateMe_exp'] = getValueSet('sinc_iRateMe_exp', DF)
    df['intel_iRateMe_exp'] = getValueSet('intel_iRateMe_exp', DF)
    df['fun_iRateMe_exp'] = getValueSet('fun_iRateMe_exp', DF)
    df['amb_iRateMe_exp'] = getValueSet('amb_iRateMe_exp', DF)

    df['match_sum'] = getSum('match', DF)
    df['dec_sum'] = getSum('dec', DF) #sum of subject's decisions (num of 'yes'')
    df['dec_o_sum'] = getSum('dec_o', DF) #sum of parnter's decisions (num of 'yes'')

    df['match_es_ave'] = getAve('match_es', DF)  #MATCH_ES_AVE  = % of people they think they'll match wit
h
```

- Recount met_count
- Rename features
- Separate between male and female datasets
- Condense the data (combine iids)
  - **dec_sum**: sum of subject's decisions (# of 'yes')
  - **dec_o_sum:** sum of partner's decisions (# of 'yes')
  - **attr_ave:** average 'attractive' rating subject gave partners he/she met with (sum of 'attractive' ratings/met_count)
  - **attr_o_ave:** average attractive rating partners gave subject

DATA EXPLORATION & ANALYSIS

```
In [124]: #returning an array of correlation values that are the highest

          c = df_female_condensed.corr()

          s = c.stack().nlargest(115)

          s.sort(ascending=False, inplace=True); s.iloc[59:]

/Users/Miranda/anaconda2/lib/python2.7/site-packages/ipykernel/__main__.py:7: FutureWarning: sort is depre
cated, use sort_values(inplace=True) for INPLACE sorting

Out[124]: exphappy          exphappy_ave        1.000000
          exphappy_ave      exphappy            1.000000
          match_sum         match_ave           0.885822
          match_ave         match_sum           0.885822
          match_es          match_es_ave        0.875759
          match_es_ave      match_es            0.875759
          like_o_ave        attr_o_ave          0.856744
          attr_o_ave        like_o_ave          0.856744
          dec_sum           dec_ave             0.840868
          dec_ave           dec_sum             0.840868
          amb_iRateMe_exp   amb_iMeasUp_2       0.830971
          amb_iMeasUp_2     amb_iRateMe_exp     0.830971
          attr_oPercveMe_2  attr_iMeasUp_2      0.815613
          attr_iMeasUp_2    attr_oPercveMe_2    0.815613
          dec_o_ave         attr_o_ave          0.811872
          attr_o_ave        dec_o_ave           0.811872
          attr_iMeasUp_2    attr_iRateMe_exp    0.808513
          attr_iRateMe_exp  attr_iMeasUp_2      0.808513
          dec_o_sum         dec_o_ave           0.798661
          dec_o_ave         dec_o_sum           0.798661
          fun_iMeasUp_2     fun_oPercveMe_2     0.797956
          fun_oPercveMe_2   fun_iMeasUp_2       0.797956
          like_o_ave        dec_o_ave           0.794109
```

match_sum correlations	
match_ave	0.885822
dec_sum	0.600592
dec_match_ave	0.58332
dec_o_sum	0.558067
dec_ave	0.486358
match_es	0.459059
dec_o_ave	0.434573
like_o_ave	0.389363
attr_o_ave	0.379035
fun_o_ave	0.350454
match_es_ave	0.315399

Correlations on female DF:

like_o_ave | attr_o_ave : 0.856744

dec_o_ave | attr_o_ave : 0.811872

like_o_ave | fun_o_ave : 0.779505

like_ave | attr_ave : 0.730203

like_ave | fun_ave : 0.727753

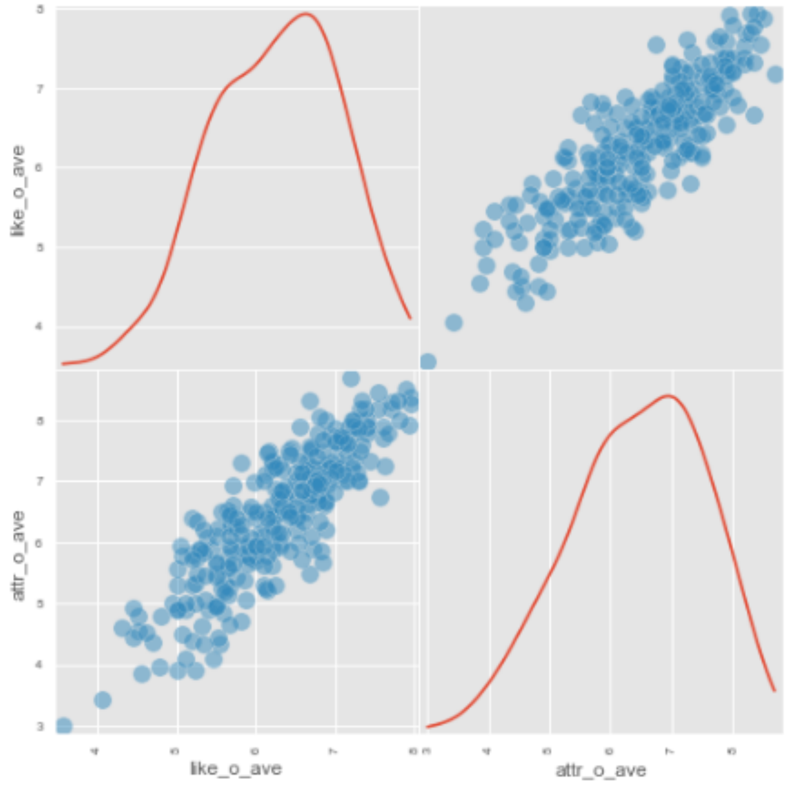
dec_match_ave | attr_o_ave : 0.620261

like_o & attr_o_ave

(had a correlation of: 0.856744 for females)

In [136]:
```python
#female DF
pd.tools.plotting.scatter_matrix(df_female_condensed[ ['like_o_ave', 'attr_o_ave'] ], diagonal = 'kde', s
= 500, figsize = (8, 8))
```

Out[136]:
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x121372790>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x121e7ac10>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x12201d350>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x12207f650>]], dtype=object)
```



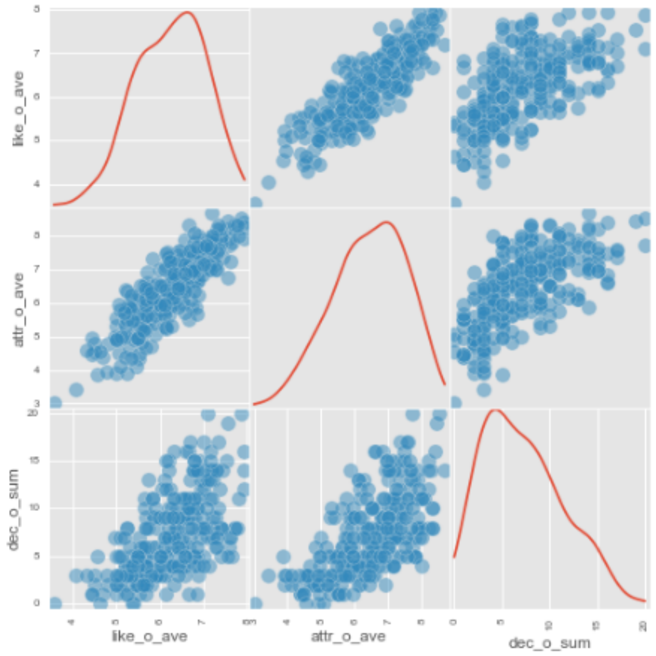
Observations: the more attractive a man found a woman he met with, the higher 'like' rating he gave her

like_o & attr_o_ave & dec_o_sum

On female dataset; as other, these results refer to what men think of the women they met with

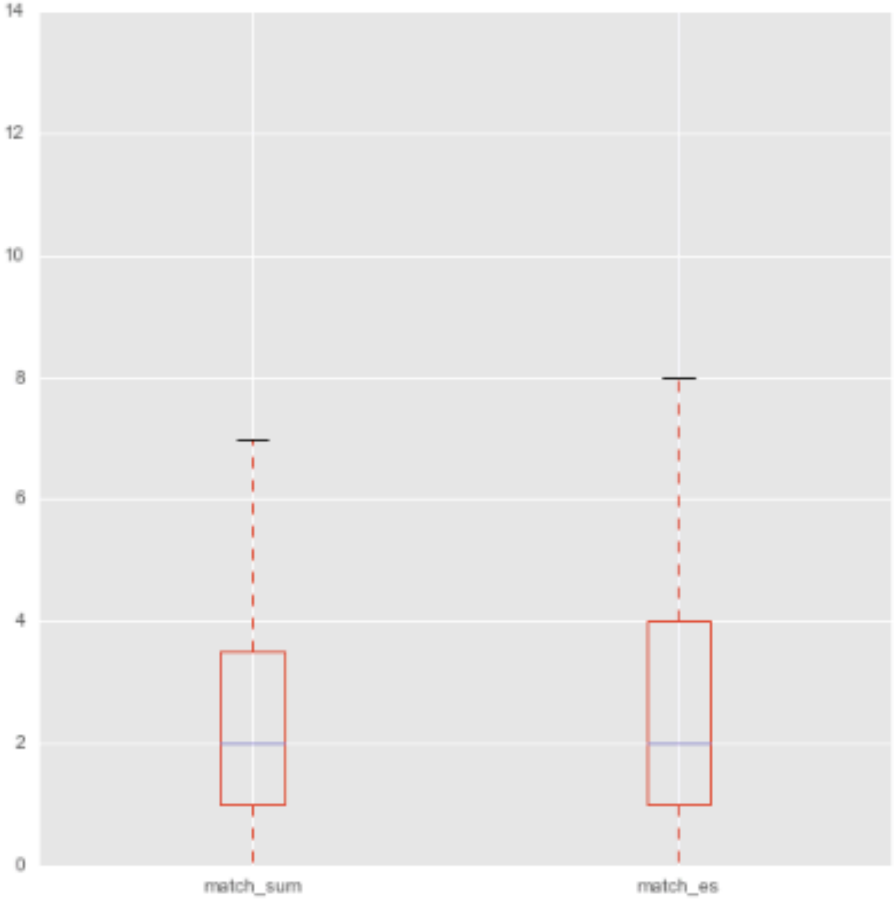
In [137]:
```
#female DF
pd.tools.plotting.scatter_matrix(df_female_condensed[ ['like_o_ave', 'attr_o_ave', 'dec_o_sum'] ], diagona
l = 'kde', s = 500, figsize = (8, 8))
```

Out[137]:
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x12217d590>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x12231e7d0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1223c1710>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x122422f10>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1224a4e90>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x12250bc90>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x122594dd0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x12274ed50>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x12179e190>]], dtype=object)
```



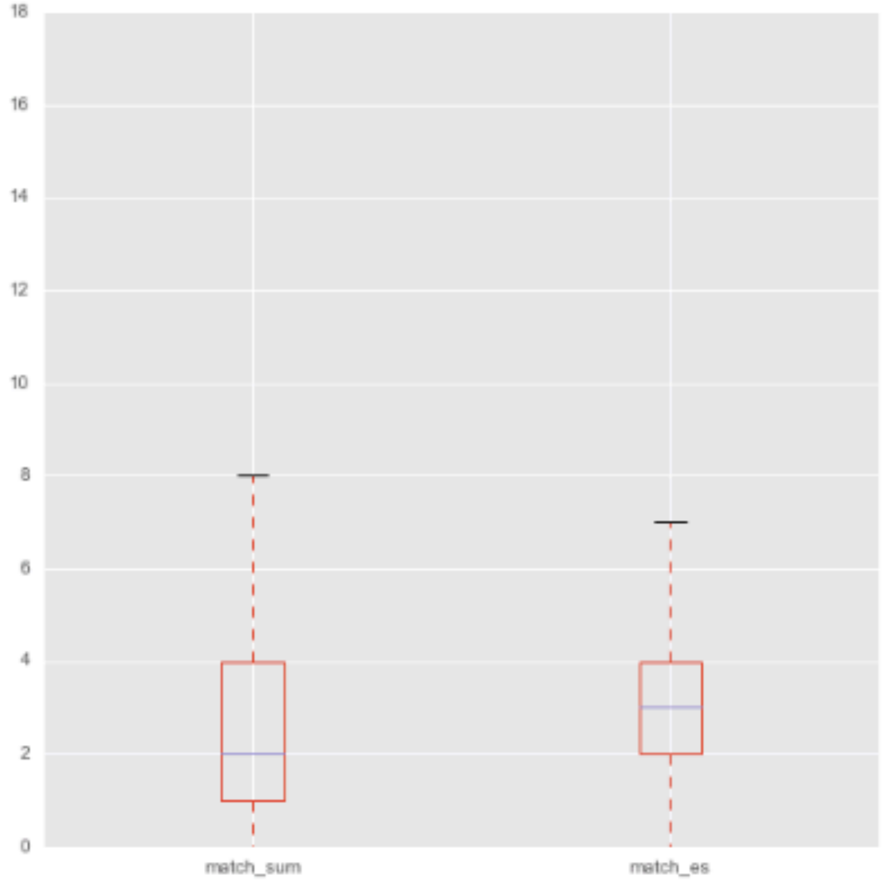
FEMALES

```
In [130]: #female dataset
          df_female_condensed[ ['match_sum', 'match_es'] ].plot(kind = 'b
Out[130]: <matplotlib.axes._subplots.AxesSubplot at 0x11b96cbd0>
```



MALES

```
In [132]: #male dataset
          df_male_condensed[ ['match_sum', 'match_es'] ].plot(kind = 'bo
Out[132]: <matplotlib.axes._subplots.AxesSubplot at 0x11cfdab50>
```



View data match_sum vs. match_es

FEMALES

In [131]:
```
df_female_condensed[ ['match_sum', 'match_es'] ].describe()
```

Out[131]:

	match_sum	match_es
count	271.000000	238.000000
mean	2.535055	2.892857
std	2.359706	2.375613
min	0.000000	0.000000
25%	1.000000	1.000000
50%	2.000000	2.000000
75%	3.500000	4.000000
max	14.000000	12.000000

MALES

In [133]:
```
df_male_condensed[ ['match_sum', 'match_es'] ].describe()
```

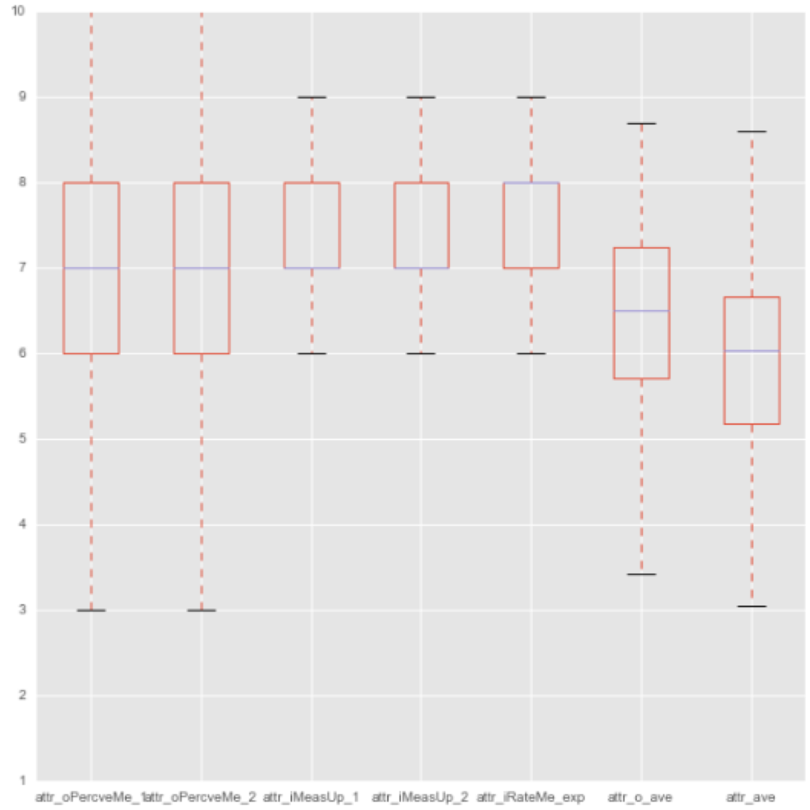
Out[133]:

	match_sum	match_es
count	276.000000	238.000000
mean	2.467391	3.178151
std	2.199757	2.319525
min	0.000000	0.000000
25%	1.000000	2.000000
50%	2.000000	3.000000
75%	4.000000	4.000000
max	11.000000	18.000000

View data match_sum vs. match_es

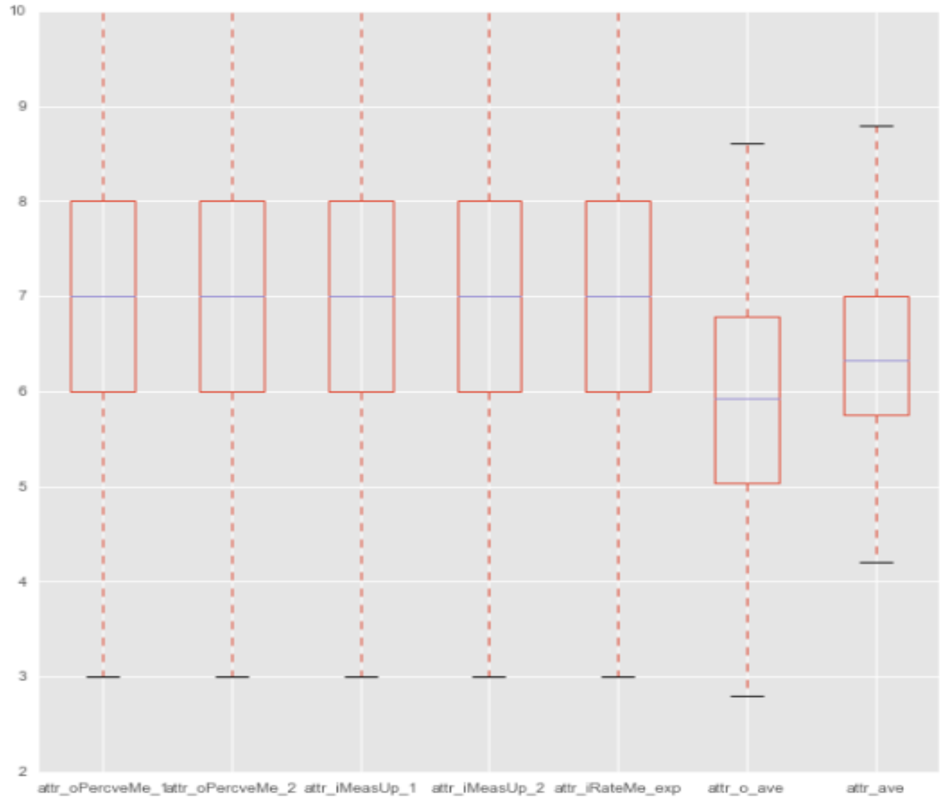
FEMALES

```
In [161]: #female dataset
          df_female_condensed[ ['attr_oPercveMe_1', 'attr_oPercveMe_2', 'attr_iMeasUp_1',  'attr_iMeasUp_2', 'attr_i
          RateMe_exp', 'attr_o_ave', 'attr_ave'] ].plot(kind = 'box', figsize = (10, 10))

Out[161]: <matplotlib.axes._subplots.AxesSubplot at 0x129d33c10>
```



MALES

```
In [164]: #male dataset
          df_male_condensed[ ['attr_oPercveMe_1', 'attr_oPercveMe_2', 'attr_iMeasUp_1',  '
          teMe_exp', 'attr_o_ave', 'attr_ave'] ].plot(kind = 'box', figsize = (10, 10))

Out[164]: <matplotlib.axes._subplots.AxesSubplot at 0x129f71f10>
```



View data for all self-ratings of 'attractiveness' as well as ratings by partners

MODELS

predicting match_sum

In [253]:
```
model = smf.ols(formula = 'match_sum ~ 0 +attr_o_ave + fun_o_ave *  attr_oPercveMe_2', d
model.summary()
```

Out[253]:

OLS Regression Results

Dep. Variable:	match_sum	**R-squared:**	0.624
Model:	OLS	**Adj. R-squared:**	0.603
Method:	Least Squares	**F-statistic:**	28.68
Date:	Tue, 12 Jul 2016	**Prob (F-statistic):**	4.78e-14
Time:	19:00:25	**Log-Likelihood:**	-155.63
No. Observations:	73	**AIC:**	319.3
Df Residuals:	69	**BIC:**	328.4
Df Model:	4		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[95.0% Conf. Int.]
attr_o_ave	0.5058	0.366	1.382	0.171	-0.224 1.236
fun_o_ave	-0.1194	0.422	-0.283	0.778	-0.962 0.723
attr_oPercveMe_2	-0.6597	0.259	-2.544	0.013	-1.177 -0.142
fun_o_ave:attr_oPercveMe_2	0.1023	0.050	2.055	0.044	0.003 0.202

Omnibus:	7.171	**Durbin-Watson:**	1.426
Prob(Omnibus):	0.028	**Jarque-Bera (JB):**	7.092
Skew:	0.763	**Prob(JB):**	0.0288
Kurtosis:	3.053	**Cond. No.**	104.

MODELS

transformations

Transforming Variables

```
In [221]: subsetdf_OPME1[ ['amb_iRateMe_exp_Log', 'amb_iMeasUp_2_Log', 'attr_oPercveMe_1_Log',
                   'sinc_oPercveMe_1_Log', 'intel_oPercveMe_1_Log', 'fun_oPercveMe_1_Log',
                   'amb_oPercveMe_1_Log', 'attr_iRateMe_exp_Log', 'fun_iRateMe_exp_Log' ] ] =
          subsetdf_OPME1[ ['amb_iRateMe_exp', 'amb_iMeasUp_2','attr_oPercveMe_1',
                     'sinc_oPercveMe_1', 'intel_oPercveMe_1', 'fun_oPercveMe_1',
                     'amb_oPercveMe_1', 'attr_iRateMe_exp', 'fun_iRateMe_exp'] ].apply(np.log10)

          subsetdf_OPME1[ ['amb_iRateMe_exp_Sqrt', 'amb_iMeasUp_2_Sqrt', 'attr_oPercveMe_1_Sqrt',
                   'sinc_oPercveMe_1_Sqrt', 'intel_oPercveMe_1_Sqrt', 'fun_oPercveMe_1_Sqrt',
                   'amb_oPercveMe_1_Sqrt', 'attr_iMeasUp_2_Sqrt', 'attr_iMeasUp_1_Sqrt'] ] = subsetdf_
           ['amb_iRateMe_exp', 'amb_iMeasUp_2','attr_oPercveMe_1',
                     'sinc_oPercveMe_1', 'intel_oPercveMe_1', 'fun_oPercveMe_1',
                     'amb_oPercveMe_1', 'attr_iMeasUp_2', 'attr_iMeasUp_1'] ].apply(np.sqrt)

          subsetdf_OPME1[ ['amb_iRateMe_exp_Square', 'amb_iMeasUp_2_Square', 'attr_oPercveMe_1_Square',
                  'sinc_oPercveMe_1_Square', 'intel_oPercveMe_1_Square', 'fun_oPercveMe_1_Square',
                   'amb_oPercveMe_1_Square', 'attr_iMeasUp_2_Square', 'attr_iMeasUp_1_Square'] ] = subsetdf_
          OPME1[ ['amb_iRateMe_exp', 'amb_iMeasUp_2','attr_oPercveMe_1',
                     'sinc_oPercveMe_1', 'intel_oPercveMe_1', 'fun_oPercveMe_1',
                     'amb_oPercveMe_1', 'attr_iMeasUp_2', 'attr_iMeasUp_1'] ].apply(np.square)
```

```
In [257]: formula = 'match_sum ~ attr_iMeasUp_2_Square + attr_iMeasUp_1_Square + attr_oPercveMe_1_Square * dec_o_sum * match_e
          formula += ' + attr_oPercveMe_1_Square * fun_oPercveMe_1_Square * intel_oPercveMe_1_Square + attr_o_ave * fun_o_ave
          formula += ' + attr_oPercveMe_1_Square * sinc_oPercveMe_1_Square * amb_oPercveMe_1_Square * fun_oPercveMe_1_Square +
          #formula += ' + amb_iRateMe_exp_Log * attr_iRateMe_exp_Log * fun_iRateMe_exp_Log'

          smf.ols(formula = formula, data = subsetdf_OPME1).fit().summary()
```

Out[257]:

OLS Regression Results

Dep. Variable:	match_sum	**R-squared:**	0.891
Model:	OLS	**Adj. R-squared:**	0.789
Method:	Least Squares	**F-statistic:**	8.675
Date:	Tue, 12 Jul 2016	**Prob (F-statistic):**	1.06e-09
Time:	19:04:14	**Log-Likelihood:**	-81.806
No. Observations:	73	**AIC:**	235.6
Df Residuals:	37	**BIC:**	318.1
Df Model:	35		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[95 Co Int
Intercept	-70.9941	71.711	-0.990	0.329	-21 74.
attr_iMeasUp_2_Square	0.0045	0.016	0.282	0.779	-0. 0.0
attr_iMeasUp_1_Square	0.0105	0.016	0.647	0.521	-0. 0.0

MODELS

returned better when predicting dec_match_ave

dec_match_ave ~ 0 +attr_oPercveMe_1 + intel_oPercveMe_1 + amb_oPercveMe_1 + fun_oPercveMe_1

forcing intercept to 0

```
model = smf.ols(formula = 'dec_match_ave ~ 0 + attr_oPercveMe_1 + intel_oPercveMe_1 + amb_oPercveMe_1 + fun_oPercveMe_1' , data = subsetdf_OPME1).fit()

model.summary()
```

Dep. Variable:	dec_match_ave	R-squared:	0.655
Model:	OLS	Adj. R-squared:	0.635
Method:	Least Squares	F-statistic:	33.25
Date:	Mon, 11 Jul 2016	Prob (F-statistic):	1.57e-15
Time:	21:47:45	Log-Likelihood:	-22.303
No. Observations:	74	AIC:	52.61
Df Residuals:	70	BIC:	61.82
Df Model:	4		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[95.0% Conf. Int.]
attr_oPercveMe_1	0.0332	0.032	1.032	0.306	-0.031 0.097
intel_oPercveMe_1	0.0368	0.033	1.117	0.268	-0.029 0.103
amb_oPercveMe_1	-0.0192	0.028	-0.676	0.502	-0.076 0.038
fun_oPercveMe_1	0.0083	0.026	0.315	0.753	-0.044 0.061

Omnibus:	8.480	Durbin-Watson:	1.525
Prob(Omnibus):	0.014	Jarque-Bera (JB):	3.229
Skew:	0.167	Prob(JB):	0.199
Kurtosis:	2.032	Cond. No.	15.9

FURTHER RESEARCH & EXPLORATION

- Look at gender perceptions; how does that predict dating outcomes
  - Does one's perception of their gender generalizations differ from their own evaluations of what's important when it comes to selecting mates?
  - E.g.: do men rate 'attractiveness' as less important for their own dating choices but more important for other men?*
    - men will rate 'attractiveness' as less important for their own dating choices but more important for other men's decisions when choosing a partner
- Look at probability rating and score; i.e. if person gave high score but thought the prob of other person choosing them was low, how that affects decision (y/n); ultimately predicting match

ACKNOWLEDGMENTS

Kaggle

Ivan Corneillet

Bob Stark

Tim Payne

Classmates

Google