Summary

Clearly, features such as “Instant Book” and, to a lesser extent, “Book it” are preferable to “Contact Me” as they reduce frictions. Beyond that, bookings have a significantly higher chance of occurring when the initial outbound message is above 400 chars in length. There is no observable evidence from this data that adding in the 140 char minimum reduces the propensity of users to contact hosts, though it also does not appear to directly affect the booking rate. The experiment could be rolled out, though it is suggestible that more experiments are run to ensure that the character limit does not lower contact/pageview conversion rate.

Definitions
-Booking rate: % of time when booking occurs within 30 days of first contact
-Acceptance rate: % of time when host accepts within 30 days of first contact
-during my analysis, I exclude Instant Book from calculations for both of these, as Instant Book should have booking and acceptance rates of 100% by design.

Experimental Results

The experiment appears to be a mixed group one, with users appearing in both the treatment and control groups. We do not have evidence as to which contact instances apply to treatment and control for users who appear in both groups. Also, some users have multiple simultaneous contacts with different hosts altering the probability of success. For the purposes of my analysis identify the following groups:

1) Treatment and control I - where users have exactly one contact in the dataset and therefore sit in one of either treatment or control. This is where I will focus my experimental analysis
2) Treatment and control II - users who appear in treatment or control, but who may appear multiple times.
3) Mixed - users who appear in both treatment and control

I decided to focus primarily on the first set, as the second set’s booking rate would likely be lower on a contact level as multiple bookings could prevent a given contact from being successfully completed. If I were to use group 2 to try to determine if installing a char limit will raise bookings, I would likely look at the acceptance rate, as this could control for the guest choosing a different booking and it’s unlikely that the length of the first message is direct contributor to the acceptance/booking rate.

At a high level there was not a noticeable difference in group 1 between acceptance or booking rates:

Figure 1: booking and acceptance rate for Treatment and Control 1

	is_booked			accepted		
	avg	stderr	n	avg	stderr	n
ab						
control	0.458187	0.005796	7390	0.603383	0.005691	7390
treatment	0.457826	0.005792	7398	0.600297	0.005695	7398

With just this information, there would not be enough information to reject a null hypothesis that the control and treatment have significantly different booking and acceptance rates. That said, it's not clear from this that there's no difference between message length and booking rates: 1) because of where the data was cutoff at 140 chars, 2) this information does not tell us whether people were dissuaded from sending messages by the character limit and 3) it seems from the user assignments that this experiment was not entirely randomized. With this in mind, in the second part of the analysis, I will attempt to build a model that isolates the relationship between booking and acceptance rates and message length.

While it's hard to draw conclusions on whether the char limit dissuades guests from booking without pageview data, group 3, the Mixed group, can possibly shed some light, depending on how control and treatment were assigned. If they were assigned randomly by pageview, we would expect people in the mixed group to show up more often in the control than in the treatment. There are 13755 treatment contacts vs 13771 control contacts from the 6554 users who are assigned to both. If the 140 char limit was a deterrent to sending messages, you could expect there to be significantly fewer treatment contacts. A better way to test the overall effect would be to assign users randomly on a user_id level and then monitor conversion rate from pageviews or sessions to booking, but if users are being assigned randomly on a message level, the evidence points to a lack of change in behaviour between the control and treatment versions of the page.

Model

Excluding "Instant Book" users, who definitionally have a booking_rate of 100%, I was able to construct a logistic regression model which was able to predict whether a contact would convert to a booking within 30 days with an accuracy of 84.2%, an ROC AUC of 84%, and a psuedo R^2 of 33%.

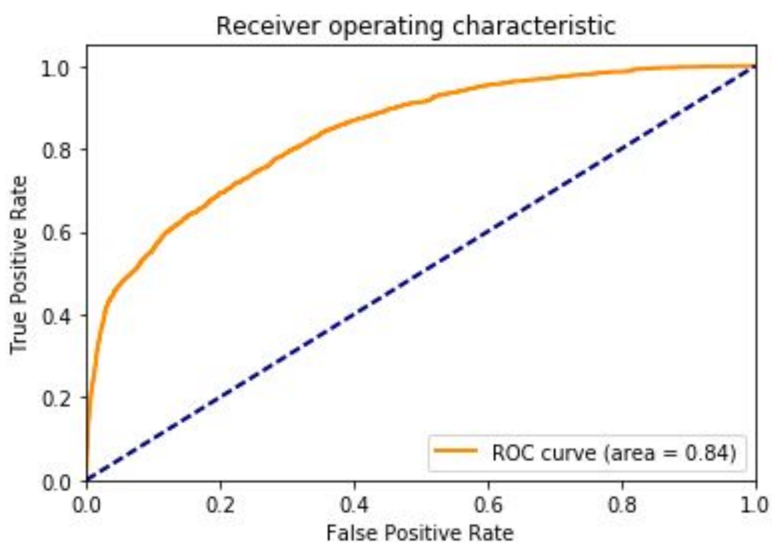
Assumptions

I assumed historical knowledge of monthly and listing level booking data (created from training data in this case). Outside of that, I assumed that the model would be given only data that was available within a day of predicting, e.g. verified guest = true only if verification date < contact date, and number of messages between guests and host would not be available as it’s a future variable.

Figure 2: Logit summary

```
                           Logit Regression Results
==============================================================================
Dep. Variable:                      y   No. Observations:                38975
Model:                          Logit   Df Residuals:                    38855
Method:                           MLE   Df Model:                          119
Date:                Sun, 02 Apr 2017   Pseudo R-squ.:                  0.3332
Time:                        16:04:58   Log-Likelihood:                -13875.
converged:                       True   LL-Null:                       -20809.
                                        LLR p-value:                     0.000
==============================================================================
```

Figure 3: ROC for logistic regression



Notable findings from model:

1) Controlling for all other factors, the coefficient of log(length of the initial message) was greater than zero at 95% confidence, and was one of the larger positive contributors to the probability of a contact converting to a booking (figure 4). This is not surprising, given the high level relationship between message length and acceptance_rate (figure 5).
2) In addition to log(length of first message), log(number of reviews) was a strong contributor to booking_rate. This was also quite clear from high level data views and could be because guests are more comfortable booking when a place has many reviews.
3) Certain listings had much higher acceptance rates than other (coef = .74), leading that listings difference from the mean to be a good predictor of booking rates.
4) Certain other coefficients were significant but not actionable by Airbnb: number of requests sent for a given trip (negatively correlated because user can only book one trip

for a given date), length of stay for shared rooms was negative (not surprising as private rooms and homes would likely be more open to long stays.

Suggestions for Improving Booking Rate

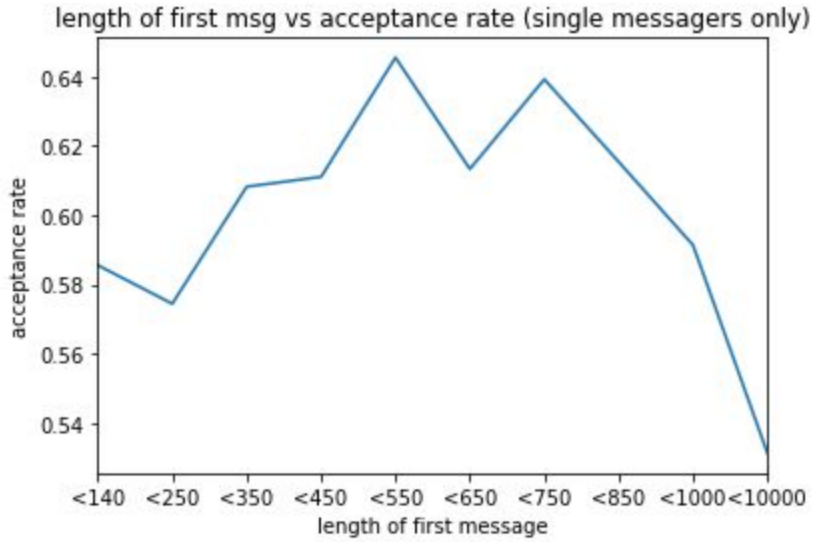
1) Given that Instant_book =100% and Book_it is significantly higher than Contact_Me in terms of booking rate, these features should be encouraged to be adopted by hosts.
2) Encourage guests to write moderate length messages to hosts, perhaps by putting suggested text in the message box or by reminding them that messages lead to booking.
3) Encourage guests and hosts to leave reviews
4) Include response and acceptance rate of listing in determining placement of listings on search pages. Deprioritize listings with low acceptance rates as they lead to potential guests wasting time and a poorer experience.

Figure 4: selected coefficients of logit model

Significant Coefficients

	coef	lb	ub	p
log_rvws^1	0.391311	0.339576	0.443046	1.013175e-49
listing^1	0.742989	0.689716	0.796262	1.609344e-164
log_first_msg^1	0.179512	0.115864	0.243160	3.241070e-08
log_num_reqs_sent^1	-0.827962	-0.893929	-0.761994	1.279452e-133
bool_book_it^1xlisting^1	0.261458	0.224490	0.298427	1.078041e-43
bool_book_it^1xlog_num_reqs_sent^1	-0.165131	-0.259100	-0.071163	5.726282e-04
bool_home^1xlog_length_of_stay^1	0.325567	0.107642	0.543493	3.410807e-03
bool_private_room^1xlog_length_of_stay^1	0.254736	0.039579	0.469893	2.031360e-02
log_rvws^2	-0.274272	-0.324093	-0.224451	3.842347e-27
log_rvws^1xlisting^1	-0.197660	-0.259626	-0.135693	4.056109e-10
listing^2	-0.164888	-0.198854	-0.130922	1.824093e-21
log_length_of_stay^2	-0.160683	-0.200414	-0.120953	2.249152e-15

Figure 5: High level acceptance rate vs length of first message (x-axis not to scale)



length of first msg vs acceptance rate (single messagers only)
acceptance rate
0.64
0.62
0.60
0.58
0.56
0.54
<140
<250
<350
<450
<550
<650
<750
<850
<1000
<10000
length of first message

Sections:

- Data import and clean
- AB test view
- Booking prediction model
- Appendix - Exploratory Analysis

In [1]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.utils import shuffle
from sklearn.metrics import confusion_matrix as cm,roc_curve,auc
from sklearn.preprocessing import
FunctionTransformer,PolynomialFeatures, OneHotEncoder, StandardScaler
from sklearn.pipeline import Pipeline
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from datetime import timedelta

%matplotlib inline
import matplotlib.pyplot as plt
```

In [2]:
```
#helper function for plotting roc

def plot_roc(y,y_hat,title_=''):
    fpr, tpr, thresholds = roc_curve(y,y_hat)
    roc_auc = auc(fpr, tpr)
    lw = 2
    plt.plot(fpr, tpr, color='darkorange',
             lw=lw,label='ROC curve (area = %0.2f)' % roc_auc)
    plt.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver operating characteristic '+ title_)
    plt.legend(loc="lower right")
    plt.show()
```

Import data

- do some conversions
- identify members of control, treatment

In [3]:
```
# import data
users_raw = pd.read_csv('data/users.csv',header=0)
listings_raw =pd.read_csv('data/listings.csv',header=0)
contacts_raw =pd.read_csv('data/contacts.csv',header=0,date_parser = Tru
e)
assignments_raw = pd.read_csv('data/assignments.csv',header=0)#.set_inde
x('id_user_anon')
```

In [4]:
```
# convert to datetimes
ts_cols = [i for i in contacts_raw if (i[:2]=='ts')]
ds_cols = [i for i in contacts_raw if i[:2]=='ds']

for i in ts_cols+ds_cols:
    contacts_raw[i]=pd.to_datetime(contacts_raw[i], errors='coerce')

users_raw['ts_first_verified']=pd.to_datetime(users_raw.ts_first_verifie
d,errors='coerce')
```

In [5]:
```
# check # of unique user, host, listing contacts
for col in ['id_guest_anon','id_host_anon','id_listing_anon']:
    print col +' unique entries in contacts: '+ str(contacts_raw[col].nu
nique())
```

```
id_guest_anon unique entries in contacts: 25209
id_host_anon unique entries in contacts: 3467
id_listing_anon unique entries in contacts: 5215
```

In [6]:
```
# clean up ab table: if user is assigned to more than one group, do not
 include
assignments = assignments_raw.groupby('id_user_anon').filter(lambda x: l
en(x)==1)##x.ab.nunique()==1)
assignments.groupby('ab').count()
```

Out[6]:

	id_user_anon
ab	
control	7909
treatment	7849

In [7]:
```
#add treatment, control to contacts
contacts = pd.merge(contacts_raw, assignments,left_on= 'id_guest_anon',r
ight_on='id_user_anon')

#remove instant book
contacts =
contacts[contacts.dim_contact_channel_first!='instant_booked']
```

Q1 Define success, see if difference between control, treatment

In [8]:
```
# define success
contacts["is_booked"] = ((contacts.ts_booking_at-contacts.ts_interaction
_first)<timedelta(days=30))*1.0
contacts["accepted"]=((contacts.ts_accepted_at_first-contacts.ts_interac
tion_first)<timedelta(days=30))*1.0

# per user
#user_level = contacts[['id_guest_anon','ab','is_booked']].groupby(['id_
guest_anon','ab']).agg(np.max).reset_index()
```

In [9]:
```
# success in group treatment v control
basic_stats = {'avg':np.mean,
              'n':'count',
               'stderr':lambda x: np.std(x)/(len(x)**.5)
               }
aggregation = {'is_booked':basic_stats,'accepted':basic_stats}

contacts.groupby('ab').agg(aggregation)
```

Out[9]:

	is_booked			**accepted**		
	avg	**stderr**	**n**	**avg**	**stderr**	**n**
ab						
control	0.458187	0.005796	7390	0.603383	0.005691	7390
treatment	0.457826	0.005792	7398	0.600297	0.005695	7398

In [10]:
```
# there are messages with len<140 in the treatment group, this should no
t be
contacts[contacts.m_first_message_length_in_characters<140].groupby('ab')
gg(aggregation)
```

Out[10]:

	is_booked			**accepted**		
	avg	**stderr**	**n**	**avg**	**stderr**	**n**
ab						
control	0.469037	0.011950	1744	0.607225	0.011694	1744
treatment	0.627596	0.018622	674	0.722552	0.017246	674

Q2 build model to predict successful booking

- construct helper tables, transform function
- test, fit LR model

In [11]:

```
# take successful to mean trip booked by user, trip defined as anon_id&c
heckout&checkin are unique



raw_df =contacts_raw
# create trips table, mapping onto listings, hosts, dropping instant boo
ks since they ==1 by definition
# the purpose of this table is to determine if a user is sending out mul
tiple messages for the same trip simultaneously
trips = pd.merge(raw_df,listings_raw,on = 'id_listing_anon')
trips = pd.merge(trips,users_raw,left_on='id_guest_anon',right_on='id_us
er_anon')
trips = pd.merge(trips,users_raw,left_on='id_host_anon',right_on='id_use
r_anon')
trips['ds_interaction_first']=trips.ts_interaction_first.dt.date
trips=trips[trips.dim_contact_channel_first!='instant_booked']
trips['accepted'] = ((trips.ts_accepted_at_first-trips.ts_interaction_fi
rst)<timedelta(days =30))*1.0


# create trips table, break into train,test
y =((trips.ts_booking_at-trips.ts_interaction_first)<timedelta(days
=30))*1.0
x_train,x_test,y_train,y_test = train_test_split(trips,y, test_size=0.2,
 random_state=777)
```

In [12]:

```
# build acceptance rate table from training set
# this will be used to normalize for listing and time of year acceptance
 rates (alternatively use oneHotEncoder)

mu = np.mean(x_train.accepted)

def normalize_acceptance(x):
    return (np.sum(x)+10*mu)/(10+len(x))-mu

accepted_deviance_by_listing =
x_train.groupby('id_listing_anon').agg({'accepted':normalize_acceptance})
eset_index()
accepted_deviance_by_listing.columns = ['id_listing_anon','listing_devia
nce']
accepted_deviance_by_listing.describe()

accepted_deviance_by_toy = x_train.groupby(x_train['ds_checkin_first'].d
t.month).\
            agg({'accepted':normalize_acceptance}).reset_index()
accepted_deviance_by_toy.columns=['int_month_booking','time_deviance']
```

In [13]:

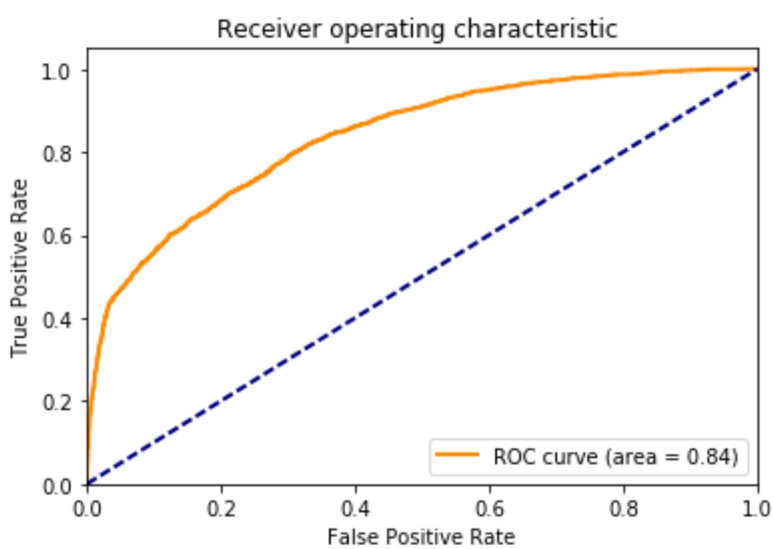
```
# transformations to construct features

#features: time of year (month of booking, categorical)
#   verified
#   2 hosts contacted, >3; is this not the first host contacted for that
 date?
#   normalized length of message (log?)
```

```
#   speaks english, is american, is canadian, other countries
#   type of booking
#   type of reservation/room_type
#  dim_person_capacity
#   # of ppl on booking
#   capacaity v ppl on booking
#   dim_total_reviews--> log, normalize, is_0
#   m_guests_first, m_interactions
# no bathrooms
# len of stay
# time of day of initial
# cross terms


# check to see if user is doing multiple bookings, this will help normal
ize for lower booking rate
# among multi-booker. this data must be extracted from training set, or
 data available at or near time
# of booking
trip_id = ['id_guest_anon','ds_interaction_first','ds_checkin_first']
trip_counter =pd.merge(x_train,x_train,how='left',
            on=trip_id).groupby(trip_id).count().reset_index()
trip_counter=trip_counter[trip_id+['id_host_anon_x']]
trip_counter.columns=[trip_id+['num_reqs_sent']]

# create categories
def transform_(df):
    df['int_month_booking']=df.ds_checkin_first.dt.month
    df= pd.merge(df,trip_counter,on=trip_id,how='left')
    df= pd.merge(df,accepted_deviance_by_listing,on='id_listing_anon',ho
w='left')
    df=
pd.merge(df,accepted_deviance_by_toy,on='int_month_booking',how='left')

    # booleans: types of bookings, matchings
    df['bool_book_it']=1.0*(df.dim_contact_channel_first=='book_it')
    df['bool_match_language']= 1.0*
(df.dim_language_x==df.dim_language_y)
    df['bool_match_country']=1.0*(df.dim_country_x==df.dim_country_y)
    df['bool_last_min']=1.0*((df.ds_checkin_first-df.ts_interaction_firs
t)<timedelta(days =7))
    df['bool_home']=(df.dim_room_type== 'Entire home/apt')*1.0
    df['bool_private_room']=1.0*(df.dim_room_type == 'Private room')
    df['bool_verified']=1.0*((df.ts_interaction_first > df.ts_first_veri
fied_x)&(~pd.isnull(df.ts_first_verified_x)))

    # relative popularity of listing, time
    df['listing']=df.listing_deviance.fillna(0)
    df['time']=df.time_deviance.fillna(0)

    # month of first interaction
    df['int_month_interaction']=df.ts_interaction_first.dt.month

    # logarithms of numerical features, with limits for free forms
    df['log_rvws']=np.log1p(df.dim_total_reviews)

df['log_first_msg']=np.log1p(df.m_first_message_length_in_characters.app
```

```
ly(lambda x: min(x,1000)))
    df['log_num_reqs_sent']=np.log1p(df.num_reqs_sent.fillna(1))
    df['log_guests_first']= np.log1p(df.m_guests_first.fillna(2))
    df['log_length_of_stay']=np.log1p((df.ds_checkout_first-df.ds_checki
n_first).apply(lambda x: min(x.days,30)))

    output_cats = ['bool_book_it','bool_match_language','bool_match_coun
try',
                  'bool_last_min','bool_home','bool_private_room','bool_
verified',
                   'log_rvws','listing','time',

'log_first_msg','log_num_reqs_sent','log_guests_first','log_length_of_st
ay']
                  #'int_month_interaction','int_month_booking']
    return df[output_cats]
```

In [17]:
```
#apply transformations
x_train_=transform_(x_train)
x_test_=transform_(x_test)

# OHE replaced by making difference from means/month a factor
#m=x_train_.shape[1]
#enc=OneHotEncoder(categorical_features=[m-2,m-1],
#    handle_unknown='error', n_values='auto', sparse=False)

poly = PolynomialFeatures()

model = Pipeline([
  #('enc',enc),
    ('ss',StandardScaler()),
  ('poly',poly),
    ('clf',LogisticRegression(C=.01,penalty='l1'))
  #('clf', RandomForestClassifier(n_estimators=100,min_samples_leaf=50))
])

# cross-validation...if I had more time gridsearchcv on (l1,l2),C
# Logistic Regression > RF for ease of interpretability

#fit and score model
model.fit(x_train_,y_train)
print model.score(x_train_,y_train)
print model.score(x_test_,y_test)
plot_roc(y_test,model.predict_proba(x_test_)[:,1])
```

```
/Users/Mikebo/miniconda2/lib/python2.7/site-packages/ipykernel/__main_
_.py:32: SettingWithCopyWarning:
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-d
ocs/stable/indexing.html#indexing-view-versus-copy

0.848646568313
0.843596059113
```



In [18]:
```
#use statsmodels to get confidence bands for LR
import statsmodels.discrete.discrete_model as sm

pipe = Pipeline([
    ('ss',StandardScaler()),
      ('poly',poly)
])


target_feature_names = ['x'.join(['{}^{}'.format(pair[0],pair[1]) for pa
ir in tuple if pair[1]!=0]) for tuple in [zip(x_train_.columns,p) for p
in poly.powers_]]
x=pd.DataFrame(data= pipe.fit_transform(x_train_,y_train),columns = targ
et_feature_names)


y=y_train
logit = sm.Logit(y_train.values,x)

alpha =0.01
L1_wt =1

# fit the model
result = logit.fit_regularized(maxiter=10000,
                               tol=1e-4,
                               alpha=alpha,
                               L1_wt=L1_wt, random_state =42)
```

```
Optimization terminated successfully.    (Exit mode 0)
            Current function value: 0.356004255405
            Iterations: 1010
            Function evaluations: 1013
            Gradient evaluations: 1010
QC check did not pass for 65 out of 120 parameters
Try increasing solver accuracy or number of iterations, decreasing alph
a, or switch solvers
Could not trim params automatically due to failed QC check.  Trimming u
sing trim_mode == 'size' will still work.
```

In [19]:
```
#book_it,log_rvws,listing,log_first_msg,log_num_reqs_sent
print(result.summary())
```

```
                           Logit Regression Results

======================================================================
=======
Dep. Variable:                      y   No. Observations:
  38975
Model:                          Logit   Df Residuals:
  38855
Method:                           MLE   Df Model:
    119
Date:                Sun, 02 Apr 2017   Pseudo R-squ.:
0.3332
Time:                        18:32:47   Log-Likelihood:
 -13875.
converged:                       True   LL-Null:
 -20809.
                                        LLR p-value:
  0.000
======================================================================
=======================================
                                                   coef    std err
   z      P>|z|      [0.025      0.975]
----------------------------------------------------------------------
---------------------------------------
                                                -0.3658        nan
nan        nan         nan         nan
bool_book_it^1                                   0.3575        nan
nan        nan         nan         nan
bool_match_language^1                            0.1178        nan
nan        nan         nan         nan
bool_match_country^1                             0.1403        nan
nan        nan         nan         nan
bool_last_min^1                                  0.1949   1.39e+05   1.41
e-06      1.000   -2.72e+05    2.72e+05
bool_home^1                                     -0.0264        nan
nan        nan         nan         nan
bool_private_room^1                              0.2243        nan
nan        nan         nan         nan
bool_verified^1                                  0.0027        nan
nan        nan         nan         nan
log_rvws^1                                       0.3913      0.026     1
4.825      0.000       0.340       0.443
listing^1                                        0.7430      0.027     2
7.335      0.000       0.690       0.796
time^1                                           0.0778      0.025
 3.161      0.002       0.030       0.126
log_first_msg^1                                  0.1795      0.032
 5.528      0.000       0.116       0.243
log_num_reqs_sent^1                             -0.8280      0.034    -2
4.599      0.000      -0.894      -0.762
log_guests_first^1                              -0.0548      0.026     -
2.097      0.036      -0.106      -0.004
log_length_of_stay^1                            -0.0273      0.027     -
1.010      0.313      -0.080       0.026
bool_book_it^2                                   0.2041        nan
nan        nan         nan         nan
bool_book_it^1xbool_match_language^1             0.0603      0.020
```

```
 2.991      0.003      0.021      0.100
bool_book_it^1xbool_match_country^1          0.0383      0.021
 1.864      0.062     -0.002      0.079
bool_book_it^1xbool_last_min^1              -0.0579      0.017     -
3.497      0.000     -0.090     -0.025
bool_book_it^1xbool_home^1                   0.0079      0.058
 0.135      0.892     -0.107      0.122
bool_book_it^1xbool_private_room^1          -0.0336      0.058     -
0.582      0.560     -0.147      0.080
bool_book_it^1xbool_verified^1              -0.0253      0.015     -
1.655      0.098     -0.055      0.005
bool_book_it^1xlog_rvws^1                    0.0097      0.021
 0.467      0.640     -0.031      0.050
bool_book_it^1xlisting^1                     0.2615      0.019     1
3.862      0.000      0.224      0.298
bool_book_it^1xtime^1                        0.0322      0.016
 1.984      0.047      0.000      0.064
bool_book_it^1xlog_first_msg^1              -0.0399      0.016     -
2.458      0.014     -0.072     -0.008
bool_book_it^1xlog_num_reqs_sent^1          -0.1651      0.048     -
3.444      0.001     -0.259     -0.071
bool_book_it^1xlog_guests_first^1           -0.0148      0.019     -
0.791      0.429     -0.051      0.022
bool_book_it^1xlog_length_of_stay^1          0.0750      0.023
 3.229      0.001      0.029      0.121
bool_match_language^2                       -0.2737        nan
nan        nan        nan        nan
bool_match_language^1xbool_match_country^1   0.0320      0.019
 1.679      0.093     -0.005      0.069
bool_match_language^1xbool_last_min^1        0.0159      0.021
 0.770      0.442     -0.025      0.056
bool_match_language^1xbool_home^1           -0.0222      0.074     -
0.301      0.764     -0.167      0.122
bool_match_language^1xbool_private_room^1   -0.0471      0.073     -
0.647      0.517     -0.190      0.095
bool_match_language^1xbool_verified^1       -0.0119      0.019     -
0.638      0.524     -0.049      0.025
bool_match_language^1xlog_rvws^1            -0.0251      0.027     -
0.935      0.350     -0.078      0.028
bool_match_language^1xlisting^1              0.0193      0.024
 0.805      0.421     -0.028      0.066
bool_match_language^1xtime^1                -0.0156      0.020     -
0.771      0.441     -0.055      0.024
bool_match_language^1xlog_first_msg^1        0.0033      0.020
 0.168      0.867     -0.036      0.042
bool_match_language^1xlog_num_reqs_sent^1    0.0454      0.030
 1.518      0.129     -0.013      0.104
bool_match_language^1xlog_guests_first^1    -0.0332      0.023     -
1.453      0.146     -0.078      0.012
bool_match_language^1xlog_length_of_stay^1  -0.0489      0.024     -
2.035      0.042     -0.096     -0.002
bool_match_country^2                        -0.2150        nan
nan        nan        nan        nan
bool_match_country^1xbool_last_min^1         0.0226      0.020
 1.109      0.268     -0.017      0.063
bool_match_country^1xbool_home^1             0.0643      0.084
 0.765      0.444     -0.100      0.229
```

```
bool_match_country^1xbool_private_room^1        0.0720       0.083
 0.864       0.388      -0.091       0.235
bool_match_country^1xbool_verified^1            0.0262       0.018
 1.441       0.149      -0.009       0.062
bool_match_country^1xlog_rvws^1                -0.0156       0.028      -
0.564       0.572      -0.070       0.039
bool_match_country^1xlisting^1                  0.0069       0.025
 0.277       0.782      -0.042       0.056
bool_match_country^1xtime^1                    -0.0050       0.020      -
0.247       0.805      -0.045       0.035
bool_match_country^1xlog_first_msg^1         4.733e-05       0.020
 0.002       0.998      -0.040       0.040
bool_match_country^1xlog_num_reqs_sent^1        0.0050       0.031
 0.161       0.872      -0.056       0.066
bool_match_country^1xlog_guests_first^1         0.0227       0.023
 0.973       0.330      -0.023       0.069
bool_match_country^1xlog_length_of_stay^1       0.0365       0.024
 1.492       0.136      -0.011       0.085
bool_last_min^2                                -0.1478    1.43e+05   -1.03
e-06      1.000   -2.81e+05    2.81e+05
bool_last_min^1xbool_home^1                     0.0424       0.064
 0.662       0.508      -0.083       0.168
bool_last_min^1xbool_private_room^1             0.0091       0.063
 0.143       0.886      -0.115       0.133
bool_last_min^1xbool_verified^1                 0.0169       0.015
 1.104       0.270      -0.013       0.047
bool_last_min^1xlog_rvws^1                     -0.0669       0.023      -
2.869       0.004      -0.113      -0.021
bool_last_min^1xlisting^1                       0.0347       0.021
 1.664       0.096      -0.006       0.076
bool_last_min^1xtime^1                          0.0020       0.018
 0.107       0.915      -0.034       0.038
bool_last_min^1xlog_first_msg^1                 0.0400       0.016
 2.577       0.010       0.010       0.070
bool_last_min^1xlog_num_reqs_sent^1             0.0430       0.026
 1.675       0.094      -0.007       0.093
bool_last_min^1xlog_guests_first^1             -0.0076       0.022      -
0.348       0.728      -0.050       0.035
bool_last_min^1xlog_length_of_stay^1           -0.0027       0.023      -
0.117       0.907      -0.047       0.042
bool_home^2                                    -0.3312         nan
nan         nan         nan         nan
bool_home^1xbool_private_room^1                -0.0136         nan
nan         nan         nan         nan
bool_home^1xbool_verified^1                    -0.0994       0.059      -
1.689       0.091      -0.215       0.016
bool_home^1xlog_rvws^1                         -0.1111       0.088      -
1.269       0.204      -0.283       0.060
bool_home^1xlisting^1                          -0.1275       0.152      -
0.838       0.402      -0.426       0.171
bool_home^1xtime^1                             -0.0065       0.073      -
0.090       0.928      -0.149       0.136
bool_home^1xlog_first_msg^1                     0.0977       0.065
 1.498       0.134      -0.030       0.226
bool_home^1xlog_num_reqs_sent^1                -0.1150       0.121      -
0.953       0.341      -0.351       0.121
bool_home^1xlog_guests_first^1                  0.1117       0.059
```

```
 1.884      0.060      -0.004      0.228
bool_home^1xlog_length_of_stay^1                   0.3256      0.111
 2.928      0.003       0.108      0.543
bool_private_room^2                               -0.2010        nan
nan        nan        nan        nan
bool_private_room^1xbool_verified^1               -0.0800      0.058     -
1.379      0.168      -0.194      0.034
bool_private_room^1xlog_rvws^1                    -0.1299      0.087     -
1.494      0.135      -0.300      0.041
bool_private_room^1xlisting^1                     -0.1042      0.151     -
0.689      0.491      -0.400      0.192
bool_private_room^1xtime^1                         0.0610      0.072
 0.843      0.399      -0.081      0.203
bool_private_room^1xlog_first_msg^1                0.0616      0.065
 0.953      0.341      -0.065      0.188
bool_private_room^1xlog_num_reqs_sent^1           -0.0646      0.120     -
0.540      0.589      -0.299      0.170
bool_private_room^1xlog_guests_first^1            -0.0254      0.061     -
0.418      0.676      -0.144      0.094
bool_private_room^1xlog_length_of_stay^1           0.2547      0.110
 2.321      0.020       0.040      0.470
bool_verified^2                                   -0.0023        nan
nan        nan        nan        nan
bool_verified^1xlog_rvws^1                        -0.0286      0.023     -
1.270      0.204      -0.073      0.016
bool_verified^1xlisting^1                          0.0159      0.021
 0.749      0.454      -0.026      0.057
bool_verified^1xtime^1                            -0.0123      0.014     -
0.877      0.380      -0.040      0.015
bool_verified^1xlog_first_msg^1                   -0.0046      0.013     -
0.341      0.733      -0.031      0.022
bool_verified^1xlog_num_reqs_sent^1               -0.0029      0.024     -
0.118      0.906      -0.050      0.045
bool_verified^1xlog_guests_first^1                -0.0025      0.020     -
0.126      0.900      -0.042      0.037
bool_verified^1xlog_length_of_stay^1               0.0204      0.021
 0.962      0.336      -0.021      0.062
log_rvws^2                                        -0.2743      0.025    -1
0.790      0.000      -0.324     -0.224
log_rvws^1xlisting^1                              -0.1977      0.032     -
6.252      0.000      -0.260     -0.136
log_rvws^1xtime^1                                 -0.0286      0.023     -
1.220      0.222      -0.075      0.017
log_rvws^1xlog_first_msg^1                        -0.0239      0.019     -
1.243      0.214      -0.062      0.014
log_rvws^1xlog_num_reqs_sent^1                    -0.0921      0.036     -
2.580      0.010      -0.162     -0.022
log_rvws^1xlog_guests_first^1                      0.0169      0.025
 0.668      0.504      -0.033      0.066
log_rvws^1xlog_length_of_stay^1                   -0.0925      0.029     -
3.238      0.001      -0.148     -0.037
listing^2                                         -0.1649      0.017     -
9.515      0.000      -0.199     -0.131
listing^1xtime^1                                  -0.0250      0.020     -
1.261      0.207      -0.064      0.014
listing^1xlog_first_msg^1                          0.0510      0.018
 2.826      0.005       0.016      0.086
```

```
listing^1xlog_num_reqs_sent^1                    0.0105      0.031
 0.334      0.738      -0.051       0.072
listing^1xlog_guests_first^1                     0.0126      0.021
 0.612      0.541      -0.028       0.053
listing^1xlog_length_of_stay^1                  -0.0309      0.026     -
1.210      0.226      -0.081       0.019
time^2                                          -0.0180      0.018     -
1.011      0.312      -0.053       0.017
time^1xlog_first_msg^1                           0.0161      0.016
 1.032      0.302      -0.014       0.047
time^1xlog_num_reqs_sent^1                      -0.0977      0.026     -
3.822      0.000      -0.148      -0.048
time^1xlog_guests_first^1                        0.0510      0.018
 2.772      0.006       0.015       0.087
time^1xlog_length_of_stay^1                      0.0048      0.023
 0.207      0.836      -0.040       0.050
log_first_msg^2                                  0.0302      0.009
 3.339      0.001       0.012       0.048
log_first_msg^1xlog_num_reqs_sent^1             -0.0945      0.032     -
2.985      0.003      -0.157      -0.032
log_first_msg^1xlog_guests_first^1              -0.0071      0.018     -
0.395      0.693      -0.042       0.028
log_first_msg^1xlog_length_of_stay^1            -0.0034      0.022     -
0.152      0.879      -0.047       0.040
log_num_reqs_sent^2                              0.0333      0.025
 1.346      0.178      -0.015       0.082
log_num_reqs_sent^1xlog_guests_first^1           0.0617      0.030
 2.070      0.038       0.003       0.120
log_num_reqs_sent^1xlog_length_of_stay^1         0.0435      0.032
 1.378      0.168      -0.018       0.105
log_guests_first^2                              -0.0414      0.014     -
2.870      0.004      -0.070      -0.013
log_guests_first^1xlog_length_of_stay^1         -0.0390      0.027     -
1.453      0.146      -0.092       0.014
log_length_of_stay^2                            -0.1607      0.020     -
7.927      0.000      -0.200      -0.121
========================================================================
=======================================
```

In [20]:

```
# significant factors
sigfacs = (result.pvalues<.05) & (abs(result.params.values)>.1)
sig_model=pd.concat([result.params[sigfacs],result.conf_int()[sigfacs],r
esult.pvalues[sigfacs]],axis=1)
sig_model.columns = ['coef','lb','ub','p']
print
print 'Significant Coefficients'
sig_model
```

```
Significant Coefficients
```

Out[20]:

	coef	lb	ub	p
log_rvws^1	0.391311	0.339576	0.443046	1.013175e-49
listing^1	0.742989	0.689716	0.796262	1.609344e-164
log_first_msg^1	0.179512	0.115864	0.243160	3.241070e-08
log_num_reqs_sent^1	-0.827962	-0.893929	-0.761994	1.279452e-133
bool_book_it^1xlisting^1	0.261458	0.224490	0.298427	1.078041e-43
bool_book_it^1xlog_num_reqs_sent^1	-0.165131	-0.259100	-0.071163	5.726282e-04
bool_home^1xlog_length_of_stay^1	0.325567	0.107642	0.543493	3.410807e-03
bool_private_room^1xlog_length_of_stay^1	0.254736	0.039579	0.469893	2.031360e-02
log_rvws^2	-0.274272	-0.324093	-0.224451	3.842347e-27
log_rvws^1xlisting^1	-0.197660	-0.259626	-0.135693	4.056109e-10
listing^2	-0.164888	-0.198854	-0.130922	1.824093e-21
log_length_of_stay^2	-0.160683	-0.200414	-0.120953	2.249152e-15

Appendix: Exploratory Analysis

```
In [21]: # explore booking vs reviews, seems to be logarithmic increase
pd.concat([x_train.dim_total_reviews,y_train],axis=1).groupby(x_train.di
m_total_reviews//5).agg({0:np.mean}).plot()
plt.title('booking rate v total reviews(/5 for easy bucketing)')
```

```
Out[21]: <matplotlib.text.Text at 0x10edbcb10>
```


```
In [22]: # create length of first msg vs acceptance rate graph
bins = [0, 140, 250,350,450,550,650,750,850,1000, 10000]
group_names = ['<'+str(i) for i in bins[1:]]
contacts['categories'] = pd.cut(contacts.m_first_message_length_in_chara
cters, bins, labels=group_names)

contacts.groupby('categories').agg({'ts_accepted_at_first':{lambda x:
np.mean(~pd.isnull(x))}}).reset_index().plot(legend=False)
plt.xticks(range(len(group_names)),group_names);
plt.xlabel('length of first message')
plt.ylabel('acceptance rate')
plt.title('length of first msg vs acceptance rate (single messagers onl
y)')
```

```
Out[22]: <matplotlib.text.Text at 0x10eea6210>
```