Asthma Survey Data must be preprocessed to ensure accurate results

1) Daily/Weekly Surveys require collapsing boolean, taking max of quant, and union of set features 2) Baseline information about medical history, asthma history, medication history, etc must also be preprocessed Must decide how to deal with multiple answers to the same question over time

Pre-processing class Should clone MySql DB to get schema, types, etc Use preprocessing script to generate views of MySql pre-processed data Create a DB for each participant which records and qc's each feature Build feedback into the app Change of education/race/ethnicity/etc should ping the user to clarify their answers

Libraries

In [1]:
```
import pandas as pd
import seaborn as sns
import os, sys
%config InlineBackend.figure_format = 'retina'
%matplotlib inline
%pylab inline
import seaborn
seaborn.set(rc={'axes.facecolor':'white', 'figure.facecolor':'white', 'grid.c
olor':'lightgrey'}\
            , font_scale=2.5)
```

```
Populating the interactive namespace from numpy and matplotlib
```

PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

Functions

In [2]:
```
def get_duplicates(df, col, date, id_name='healthCode'):
    '''
    This function returns a dataframe with duplicate
    entries ordered by time

    '''
    df.sort(date, inplace=True)

    dups = df.reset_index()[[id_name, col]].dropna().drop_duplicates()[id_nam
e].value_counts()
    dups = dups[dups>1]
    df = df.ix[dups.index].set_index(date, append=True)[col]
    return df
```

PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

In [3]:
```
def strip_brackets(series):

    def bracket_repalce(x):
        if x == '[]':
            return np.NaN
        else:
            return x.rstrip(']').lstrip('[')

    series =  [bracket_repalce(i) if type(i)==str else i for i in series]
    try:
        return map(int, series)
    except:
        return series
```

In [4]:
```
metadata_cols = ['recordId', 'createdOn', 'appVersion', 'ROW_VERSION', 'uploa
dDate', 'ROW_ID']
set(metadata_cols)

def data_columns(cols, metadata_cols=['recordId', 'createdOn', 'appVersion',
\
                                      'ROW_VERSION', 'uploadDate', 'ROW_ID',\
                                     'phoneInfo']):

    return list( set(cols) - set(metadata_cols))



def get_data(df, cols):

    return df[data_columns(cols)]
```

PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

In [5]:
```
def get_count(vector):
    if True in vector or False in vector:
        return [1 if i in [True, False] else 0 for i in vector]

    vector_counts = []
    for i in vector:
        if i == '[]':
            vector_counts.append(0)
            continue
        if i > -1:
            vector_counts.append(1)
            continue
        vector_counts.append(0)

    return vector_counts
```

In [6]:
```
def flatten(set_items):
        '''
        This function flattens a set of
        comma-delimited strings
        '''
        union = set()
        try:
            for s in set_items:
                union.update(s.split(','))
            return ','.join(list(union))
        except TypeError:
            return np.NaN
```

PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

In [7]:
```
def join_df(df_list):

    master_df = df_list[0]
    for df in df_list[1:]:
        master_df = master_df.join(df, how='outer')
    return master_df
```

Specific Fixes

In [8]:
```
def fix_asthma_gets_worse_with(ans):
    '''
    This function fixes issues with
    the asthma_gets_worse_with values which have
    malformed Other values
    '''
    answers = set()
    for i in str(ans).split(','):
        if 'Other' in i:
            answers.add('other')
            continue
        if 'nan' in i:
            continue
        answers.add(i)

    ans_str = ','.join([i for i in list(answers) if len(i) > 0])
    if type(ans_str) != float:
        return ans_str
    else: return np.NaN
```

PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

Function for Baseline Survey Collapsing

PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

In [9]:
```
def baseline_survey_preprocess(df_date_value, date_col, static=True, first=True):

    '''
    This function collapses duplicate baseline survey responses

    df_date_value is a pandas dataframe with a date column and the column of interest

    date_col is the name of the column containing the date for the column of interest

    static
        True: values that shouldn't change, e.g. age_when_diagnosed, sex
            The most frequent value, ties are broken by the first/last date


        False: values that may change over time, e.g. age, education, income
            first response for things that can change

    first
        determines whether ties are broken by the earliest date (first==True) or the last date (first==False)

    '''



    df_date_value = df_date_value.sort(columns=date_col, ascending=first)  #sorting by date


    index_cols = df_date_value.index.names
    df_date_value = df_date_value.reset_index().drop_duplicates()
```

PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

In [10]:
```
def get_first_response(df, id_col, date_col, value_col, keep='first'):
    '''
    This function returns the first or last chronological response for each
    non-nan column value
    '''
    if type(value_col) != list:
        value_col = [value_col]  #transform value_col into list
    df = df.dropna(subset=value_col)  #drop na responses to value_col
    df.sort(date_col, inplace=True)  #sorting by date_col
    return df.drop_duplicates(subset=[id_col], keep=keep).set_index(id_col)
#dropping duplicates responses for each id, value_col

#returning response indexed on id_col
```

PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

In [11]:
```
#WORKING FUNCTION BUT SLOW, get_first_response is faster

def get_first_by_date(participant_df, date_col='dateTime', nan_dict={}, date_
threshold='7D', keep='first'):
    '''
    This function returns the first or last chronological response for each
    non-nan column value

    participant_df : single participant's data indexed on healthCode

    eg df.xs('4718cc70-0d6e-4ea0-a9a9-9933b88f4a19', level=1)

    nan_dict : transforming values to nan if one wants to not consider them

    date_threshold : consider responses within the specified time spane 7D eq
uals first 7 days

    keep : whether to keep the first value or the last value in the sorted da
ta after date_threshold
           filtering
    '''

    def return_value(x):
        return x

    participant_df.set_index(date_col, inplace=True)  #setting timestamp inde
x

    if nan_dict != {}:
        participant_df.replace(to_replace=nan_dict, inplace=True)  #removing
values that should be redacted

    participant_df.sort_index(level=0, inplace=True, ascending=True)  #sort b
```

PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

```
y timestamp

    participant_df = participant_df.first(date_threshold)  #consider only use
r-specified timespan of data collection, e.g. first 7 days

    participant_df = participant_df.stack().reset_index(level=1).drop_duplica
tes(subset='level_1', keep=keep)   #taking first or last answer ordered by ti
me

    participant_df.columns = ['column', 'value']  #renaming columns

    return participant_df#.reset_index().pivot_table(index='dateTime', column
s='column', values='value', aggfunc=return_value)



# user_info_surveys = ['AsthmaHistory', 'AsthmaMedication', 'YourAsthma', 'Ab
outYou', 'MedicalHistory']
# user_info_cols = [n for c in user_info_surveys for n in data_columns(table_
columns[c]) if n not in baseline_set_qs]
# user_info_df = df_filtered[user_info_cols + ['dateTime', 'date', 'study_day
']].dropna(subset=user_info_cols, how='all')

# na_dict = {'intubated':{'3':np.NaN}, 'prescribed_asthma_control_medication'
:{'3':np.NaN}, 'steroid_which':{'4':np.NaN},\
#           'daily_controller_medication':{'9':np.NaN}, 'asthma_gets_worse_w
ith':{'22':np.NaN}, 'lung_function':{'3':np.NaN},\
#           'health_insurance':{'4':np.NaN}, 'race':{'7':np.NaN}, 'ethnicity
':{'3':np.NaN}, 'Income':{'7':np.NaN}, \
#           'education':{'8':np.NaN}}

# # collapsed_baseline = {}
# counter = 0
```

PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

```
# for i,n in user_info_df.replace(to_replace=na_dict).groupby(level=1):
#     collapsed_baseline[i] = get_first_by_date(user_info_df.xs(i, level=1),
date_threshold='14D', keep='last')
#     counter +=1
#     if counter % 200 == 0:
#         print counter
# user_info_df = pd.concat(collapsed_baseline).reset_index('dateTime', drop=T
rue).set_index('column', append=True).unstack().T.reset_index(level=0, drop=T
rue).T

# user_info_df = user_info_df.join(baseline_set_df, how='inner')
# user_info_df

# user_info_df.to_csv('../analysis/user_info_baseline_df_2015_12_04_14D_last.
tsv',sep='\t')
# !gzip ../analysis/user_info_baseline_df_2015_12_04_14D_last.tsv

#!!!!TODO: FILL NA WITH na_dict values for each healthCode!!!!
```

In [12]:
```
# user_info_surveys = ['AsthmaHistory', 'AsthmaMedication', 'YourAsthma', 'Ab
outYou', 'MedicalHistory']
# for s in user_info_surveys:
#     for n in data_columns(table_columns[s]):
#         print s+'\t'+n
```

PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

In [13]:
```
#asthma_history - all questions about last month of 6 months so use a study_d
ay cutoff of 14 days
#
# asthma_med = df_filtered[df_filtered.study_day < 14].ix['AsthmaMedication']
[data_columns(table_columns['AsthmaMedication'] + ['date'])].reset_index().dr
op_duplicates()

#AboutYou -
    #static
        #ethnicity 1 or 2 over 3, race 1-5 over 6,7 union,
    #dynamic
        #Income 1-5 over 6 or 7, education 1-7 over 8, smoking_status 2 over
1 or 3, avg_cig int, smoking_years int
        #health_insurance 1-3 over 4

#AsthmaMed -
    #dynamic
        #prescribed_asthma_control_medication 1 or 2 over 3, daily_inhaled_me
dicine categ, controlmed 1-4 over 5 or 6
        #daily_controller_medication 1-7 over 8 or 9, steroid_which categ, st
eroid_dose int, quick_relief categ,
        #past_month_quick_relief
```

Function for Daily and Weekly Survey Collapsing

PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

In [14]:
```
weekly_bool_qs = ['limitations', 'oral_steroids',  'prednisone', 'missed_work
', 'asthma_doc_visit', \
                  'admission', 'emergency_room', 'asthma_medicine']

weekly_quant_qs = ['limitations_days']

weekly_set_qs = ['side_effects', 'oral_steroids_when', 'prednisone_when', 'ad
mitted_when', \
                  'er_when', 'missed_work_days', 'admitted_end']


def handle_multiple_survey_answers(df, bool_qs, quant_qs, set_qs):
    '''
    This function combines information from surveys submitted on the same stu
dy day

    df is expected to have healthCode and study_day as indices

    Boolean Questions:
    Any True answer to questions a boolean question is used

    Quantitative Questions:
    The maximum value is taken for peakflow or number of quick relief puffs

    Set Questions:
    The union of the answers to the get_worse trigger questions is used
    '''
    #bool_qs, quant_qs, set_qs = arg_list

    def join_df(df_list):
        '''
        This function left joins a list of pandas
        on the index
```

PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

```
    '''
    master_df = df_list[0]
    for df in df_list[1:]:
        master_df = master_df.join(df, how='outer')
    return master_df


def flatten(set_items):
    '''
    This function flattens a set of
    comma-delimited strings
    '''
    union = set()
    for s in set_items:
        union.update(s.split(','))
    return ','.join(list(union))


df = df.reset_index()
df = df.set_index(pd.DatetimeIndex(df['dateTime']))
df.index.name = 'dateTime'
df.sort_index(inplace=True, ascending=True)


results = []
na_results = []
for c in bool_qs + quant_qs + set_qs:


    if c in bool_qs:  #boolean questions, take True
        bool_df = df[[c, 'healthCode', 'study_day']].copy()
        bool_df = bool_df.dropna(subset=[c])  #dropping na values
```

PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

```
            if len(bool_df) == 0: #handles empty dataframes
                na_results.append(c)
                continue

            bool_true = bool_df[bool_df[c]==True]
            bool_true = bool_true.drop_duplicates(subset=['healthCode', 'stud
y_day'])  #much faster than inplace
            bool_true = bool_true.reset_index(drop=False).set_index(['healthC
ode', 'study_day'])  #reset dateTime

            bool_false = bool_df[bool_df[c]==False]
            bool_false = bool_false.drop_duplicates(subset=['healthCode', 'st
udy_day'])
            bool_false = bool_false.reset_index(drop=False).set_index(['healt
hCode', 'study_day'])  #reset dateTime

            bool_true = bool_true.combine_first(bool_false)  #using True if p
resent for healthCode, study_day

            bool_true.rename(columns={'dateTime':'dateTime_' + c }, inplace=T
rue)  #renaming dateTime
            results.append(bool_true)  #saving results
            continue



        if c in quant_qs:  #quant questions, take Max for each healthcode and
studyday
            quant_df = df[[c, 'healthCode', 'study_day']].dropna(subset=[c]).
copy()
            quant_df = quant_df.reset_index(drop=False)  #reset dateTime
            quant_df.sort(columns=[c], ascending=False, inplace=True)  #sorti
ng by column value
```

PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

```
            quant_df = quant_df.drop_duplicates(subset=['healthCode', 'study_
day'])  #dropping lower values
            quant_df.rename(columns={'dateTime':'dateTime_' + c }, inplace=Tr
ue)  #renaming dateTime column
            results.append(quant_df.set_index(['healthCode', 'study_day']))
#saving results indexed on healthCode & study_day
            continue



        if c in set_qs:  #set questions, take set union

            index_name = 'dateTime_' + c
            df.index.name = index_name


            set_pivot = df[[c, 'healthCode', 'study_day']].dropna(subset=[c])
.pivot_table(index='healthCode', columns='study_day', values=c, aggfunc=set)
            set_pivot = set_pivot.stack().reset_index()
            set_pivot[c] = [flatten(map(str, i)) for i in set_pivot[0]]
            del set_pivot[0]

            set_pivot_date = df[[c, 'healthCode', 'study_day']].reset_index()
.pivot_table(index='healthCode', columns='study_day', values=index_name, aggf
unc=set)
            set_pivot_date = set_pivot_date.stack()
            set_pivot_date.name = index_name

            set_df = set_pivot.set_index(['healthCode', 'study_day']).join(se
t_pivot_date, how='left')

            results.append(set_df)
            continue
```

PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

```
        #catch issues
        if c == 'dateTime':
            results.append(','.join(map(str, df[c].unique())))
            continue
        assert False, c


    joined_df = join_df(results)
    for c in na_results:
        joined_df[c] = np.NaN


    return joined_df
```

Time Functions

In [15]:
```
def get_last_day(dataframe, cols):
    '''
    This function returns the last study day entry for any entry
    with at least one value in the cols
    '''
    dataframe_last = dataframe.copy()
    for c in cols:  #handling empty responses
        dataframe_last[c] = dataframe_last[c].replace('[]', np.NaN)

    dataframe_last = dataframe_last.dropna(subset=cols, how='all')

    return dataframe_last.groupby(level=1)['study_day'].max()
```

PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

In [16]:
```
#Date Creation and Parsing

def add_dateTime(timestamp):
    '''
    This function converts the timestamp into dateTime.
    Some timestamps are erroneous
    '''
    try:
        return pd.to_datetime(timestamp, unit='ms', infer_datetime_format=Tru
e)
    except:
        return -999  #indicates error


def add_study_dates(df_study_dates):
    '''
    This function adds study_entry, study_day, last_entry, last_study_day
    to dataframe

    study_entry: first createdOn row for a user
    study_day: days enrolled for that row
    last_entry: most recent createdOn row for a user
    last_study_day: last_entry - study_entry

    '''

    first_created_on = add_dateTime(df_study_dates.groupby(level=1)['createdO
n'].min())
    first_created_on.name = 'study_entry'
    df_study_dates = df_study_dates.join(first_created_on)  #adding study_ent
ry
    df_study_dates['study_entry_month'] = df_study_dates.study_entry.dt.month
```

PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

```
    df_study_dates['study_day'] = df_study_dates.date.subtract(df_study_dates
.study_entry.dt.date).dt.days  #adding study day

    last_entry = add_dateTime(df_study_dates.groupby(level=1)['createdOn'].ma
x())
    last_entry.name = 'last_entry'

    df_study_dates = df_study_dates.join(last_entry)
    last_study_day = last_entry.subtract(df_study_dates.groupby(level=1)['stu
dy_entry'].max()).dt.days
    last_study_day.name = 'last_study_day'
    df_study_dates = df_study_dates.join(last_study_day)
    return df_study_dates


def select_date(df, date_col, year_month_day):
    '''
    This function returns a subset of the data
    with the date_col < year_month_day

    date_col must be dt.date format
    '''

    year, month, day = year_month_day
    return df[df[date_col] < pd.datetime(year, month, day).date()]
```

PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

In [17]:
```
#Filtering criteria proposed by Steve
def filter_criteria(df):
    df = df[df.appVersion.map(lambda x: True if 'YML' not in str(x) else Fals
e)]
    #df = df[df.dateTime!= -999]
    #simulator_ids = df[df.phoneInfo.isin(['iPhone8,1','iPhone8,2'])].index.g
et_level_values(1).unique()
    #df = df[~df.index.get_level_values(1).isin(simulator_ids)]
    return df
```

Table Columns

PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

In [18]:
```
def get_table_columns(dir_path='./test_data/'):
    '''
    This function creates a dictionary of table columns
    indexed on table name
    '''

    csvs = [i for i in os.listdir(dir_path) if i.endswith('csv')]

    table_columns = dict()
    for i in csvs:
        table_columns[i.rstrip('.csv')] = list(set(pd.read_table(dir_path + i
, sep=',', nrows=1).columns.tolist()) - set(['healthCode','externalId']))
    return table_columns

table_columns = get_table_columns()


def get_table_responses(df, table, table_cols=table_columns):
    '''
    This function returns table rows with at least
    one non-Na value
    '''
    return df.ix[table][table_cols].dropna(how='all')

table_columns
```

PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

```
Out[18]: {'AboutYou': ['education',
           'recordId',
           'createdOn',
           'appVersion',
           'smoking_status',
           'ROW_VERSION',
           'race',
           'smoking_years',
           'uploadDate',
           'ethnicity',
           'Income',
           'phoneInfo',
           'ROW_ID',
           'health_insurance',
           'avg_cigarettes'],
          'AsthmaDailyPrompt': ['get_worse',
           'quick_relief_puffs',
           'day_symptoms',
           'createdOn',
           'appVersion',
           'medicine_change',
           'recordId',
           'ROW_VERSION',
           'uploadDate',
           'night_symptoms',
           'medicine',
           'phoneInfo',
           'ROW_ID',
           'use_qr',
           'peakflow'],
          'AsthmaHistory': ['doc_times',
           'nights',
           'intubated',
```

PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

```
 'emergency',
 'times_hospitalized',
 'recordId',
 'limited activity',
 'createdOn',
 'appVersion',
 'age_when_diagnosed',
 'symptoms',
 'hospitalized_times',
 'ROW_VERSION',
 'emergency_times',
 'uploadDate',
 'miss_work',
 'phoneInfo',
 'ROW_ID',
 'oral steroids',
 'seen_doc'],
'AsthmaMedication': ['other_meds',
 'alvesco_dose',
 'ROW_ID',
 'flovent_diskus_dose',
 'prescribed_asthma_control_medication',
 'dulera_dose',
 'past_month_quick_relief',
 'nebulizer_meds',
 'steroid_dose',
 'appVersion',
 'symbicort_dose',
 'daily_yes',
 'phoneInfo',
 'pulmicort_dose',
 'steroid_which',
 'controlmed',
```

PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

```
 'advair_hfa_dose',
 'daily_controller_medication',
 'uploadDate',
 'control_puffs',
 'breo_dose',
 'nebulize_daily',
 'advair_diskus_dose',
 'qvar_dose',
 'daily_inhaled_medicine',
 'asmanex_dose',
 'recordId',
 'ROW_VERSION',
 'quick_relief',
 'flovent_hfa_dose',
 'createdOn',
 'use_nebulizer'],
'AsthmaWeeklyPrompt': ['asthma_doc_visit',
 'appVersion',
 'oral_steroids_when',
 'emergency_room',
 'asthma_medicine',
 'ROW_ID',
 'missed_work_days',
 'missed_work',
 'recordId',
 'admission',
 'prednisone_when',
 'admitted_end',
 'er_when',
 'phoneInfo',
 'limitations',
 'admitted_when',
 'prednisone',
```

PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

```
 'oral_steroids',
 'uploadDate',
 'createdOn',
 'ROW_VERSION',
 'limitations_days',
 'side_effects'],
'EQ_5D': ['health_today',
 'slider_instructions',
 'pain',
 'mobility',
 'recordId',
 'createdOn',
 'EQ5Instructions',
 'appVersion',
 'ROW_VERSION',
 'intro',
 'uploadDate',
 'phoneInfo',
 'selfcare',
 'ROW_ID',
 'usual_activities',
 'depression'],
'HealthKitDataCollector': ['recordId',
 'createdOn',
 'appVersion',
 'ROW_VERSION',
 'uploadDate',
 'phoneInfo',
 'ROW_ID',
 'data.csv'],
'MedicalHistory': ['peripheral',
 'appVersion',
 'stroke',
```

PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

```
 'tissue',
 'other_lung_disease',
 'ulcer',
 'Congestive',
 'ROW_ID',
 'dementia',
 'kidney',
 'recordId',
 'ROW_VERSION',
 'malignant_lymphoma',
 'phoneInfo',
 'arthritis',
 'leukemia',
 'tested',
 'uploadDate',
 'heart_attack',
 'createdOn',
 'tumor',
 'chronic_pulmonary_diesease',
 'allergic_to',
 'liver'],
'Milestone': ['weight',
 'alleviate_troubles',
 'use_aap',
 'symptoms2',
 'met_goal',
 'current_goal',
 'nights2',
 'ROW_ID',
 'prevent_ed',
 'appVersion',
 'phoneInfo',
 'current_troubles',
```

PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

```
 'recordId',
 'make_aap',
 'uploadDate',
 'height',
 'cause_visit',
 'prevent_visit',
 'gender',
 'age',
 'daily_yes2',
 'best_peakflow',
 'ROW_VERSION',
 'past_month_quick_relief2',
 'got_spirometry',
 'createdOn',
 'limited activity2'],
'YourAsthma': ['asthma_gets_worse_with',
 'lung_function',
 'plan',
 'recordId',
 'createdOn',
 'appVersion',
 'flu_prompt',
 'peak_flow',
 'troubles_about_asthma',
 'ROW_VERSION',
 'aap_prompt',
 'flushot',
 'phoneInfo',
 'asthma_control',
 'ROW_ID',
 'uploadDate'],
'aqiResponse': ['aqiResponse.json.reporting_area',
 'aqiResponse.json.reports',
```

PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

```
 'recordId',
 'createdOn',
 'appVersion',
 'ROW_VERSION',
 'uploadDate',
 'aqiResponse.json.state_code',
 'phoneInfo',
 'ROW_ID']}
```

In [19]:
```
#Common columns across tables
from collections import Counter
c = Counter([n for i in table_columns.keys() for n in table_columns[i] ])
c.most_common(10)
```

Out[19]:
```
[('recordId', 11),
 ('appVersion', 11),
 ('phoneInfo', 11),
 ('createdOn', 11),
 ('ROW_VERSION', 11),
 ('ROW_ID', 11),
 ('uploadDate', 11),
 ('quick_relief_puffs', 1),
 ('alleviate_troubles', 1),
 ('emergency_times', 1)]
```

Data

PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

In [20]:
```
#READING SAGE DATA
df = pd.read_table('../data/sage_data_unfiltered_2015_12_04.tsv.gz', sep='\t'
, compression='gzip')
df.set_index(['Unnamed: 0', 'healthCode'], inplace=True)
df.index.names = ['table', 'healthCode']
df['dateTime'] = df.createdOn.map(add_dateTime)
df = df[df.dateTime!=-999]
df['dateTime'] = df.dateTime.astype('datetime64[ms]')
df['date'] = df.dateTime.dt.date
```

```
/Users/ers/anaconda/lib/python2.7/site-packages/IPython/core/interactiveshell
.py:2902: DtypeWarning: Columns (2,3,6,7,8,10,11,14,15,16,17,18,20,21,22,23,2
4,25,26,29,30,31,33,35,36,37,38,42,43,44,46,47,48,49,52,53,54,55,56,57,58,59,
60,62,65,66,67,68,70,71,72,73,74,75,76,77,78,79,80,81,82,83,84,85,86,87,88,89
,91,93,94,95,98,99,100,101,103,104,105,106,107,109,110,112,113,114,115,118,11
9,120,121,122,124,125,126,127,128,129,131,132,133,134) have mixed types. Spec
ify dtype option on import or set low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

In [27]:
```
#raw_daily_nicole_check = select_date( df.ix['AsthmaDailyPrompt'], 'date', (2
015,9,10))
#raw_daily_nicole_check.dropna(subset=daily_survey_q, how='all').shape
```

PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

In [21]:
```
#Adding date columns
df_filtered = filter_criteria(df)  #removing irrelevant data, e.g. iphone sim
ulator
#df_filtered['dateTime'] = df_filtered.dateTime.astype('datetime64[ms]')  #ch
anging dateTime to pandas dateTime
#df_filtered['date'] = df_filtered.dateTime.dt.date  #collecting date
df_filtered = add_study_dates(df_filtered)  #adding multiple study dates
df_filtered['value'] = 1  #for pivot tables
df_filtered['study_entry'] = pd.to_datetime(df_filtered.study_entry)

#REMOVING BRACKETS
df_filtered = df_filtered.apply(strip_brackets, axis=0)

#CLEANING MALFORMED RESPONSES
df_filtered['asthma_gets_worse_with'] = df_filtered.asthma_gets_worse_with.ma
p(fix_asthma_gets_worse_with)
df_filtered.shape
```

Out[21]:
```
(221379, 142)
```

Baseline Survey Data

PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

In [22]:

```
baseline_set_qs = ['nebulizer_meds', 'daily_controller_medication', 'quick_re
lief', 'asthma_gets_worse_with',\
                   'troubles_about_asthma', 'asthma_control', 'race', 'allergi
c_to']

baseline_set_df = df_filtered[df_filtered.study_day<15].dropna(subset=baselin
e_set_qs, how='all')
baseline_set_df = baseline_set_df.reset_index(level=0, drop=True)#.set_index(
'study_day', append=True)
del baseline_set_df['study_day']
baseline_set_df = baseline_set_df[baseline_set_qs].stack().reset_index().pivo
t_table(index='healthCode', columns='level_1', values=0, aggfunc=set)

baseline_set_df = pd.DataFrame([baseline_set_df[c].map(flatten) for c in base
line_set_df.columns]).T



user_info_surveys = ['AsthmaHistory', 'AsthmaMedication', 'YourAsthma', 'Abou
tYou', 'MedicalHistory']
user_info_cols = [col for table in user_info_surveys for col in data_columns(
table_columns[table]) if col not in baseline_set_qs]

user_info_df = df_filtered[user_info_cols + ['dateTime', 'date', 'study_day']
].dropna(subset=user_info_cols, how='all')



user_info_df_collapsed = []
for col in user_info_cols:
    user_info_df_collapsed.append(get_first_response(user_info_df[['dateTime'
, col, 'study_day']].reset_index(), 'healthCode', 'dateTime', col)[[col]])



user_info_df_collapsed = join_df(user_info_df_collapsed)
```

PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

```
/Users/ers/anaconda/lib/python2.7/site-packages/ipykernel/__main__.py:9: Futu
reWarning: sort(columns=....) is deprecated, use sort_values(by=.....)
/Users/ers/anaconda/lib/python2.7/site-packages/pandas/core/frame.py:3167: Se
ttingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/st
able/indexing.html#indexing-view-versus-copy
  inplace=inplace, kind=kind, na_position=na_position)
```

In [23]:
```
user_info_df_collapsed = user_info_df_collapsed.join(baseline_set_df, how='ou
ter')
```

In [24]:
```
user_info_df_collapsed.shape
```

Out[24]: (7143, 72)

Age & Sex

In [25]:
```
age_sex = pd.read_table('../data/raw_agesex_2015_12_4.tsv',sep='\t')
age_sex.set_index('healthCode', drop=False, inplace=True)
col_dict = dict((i, i.replace('NonIdentifiableDemographics.json.patient', '')
) for i in age_sex.columns)
age_sex.rename(columns=col_dict, inplace=True)
age_sex = age_sex.drop_duplicates()
age_sex['dateTime'] = age_sex.createdOn.map(add_dateTime)
age_sex.info()
```

PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

```
<class 'pandas.core.frame.DataFrame'>
Index: 3408 entries, 30495a1f-6524-4a07-b543-0cf53e64de21 to 68fc6399-4f53-48
d8-a4d4-208d876ff145
Data columns (total 15 columns):
recordId                                3408 non-null object
healthCode                              3408 non-null object
externalId                              0 non-null float64
uploadDate                              3408 non-null object
createdOn                               3408 non-null object
appVersion                              3408 non-null object
phoneInfo                               3408 non-null object
WeightPounds                            3405 non-null float64
BiologicalSex                           3252 non-null object
HeightInches                            3405 non-null float64
WakeUpTime                              0 non-null float64
CurrentAge                              3253 non-null float64
GoSleepTime                             0 non-null float64
NonIdentifiableDemographics.json.item   3405 non-null object
dateTime                                3408 non-null datetime64[ns]
dtypes: datetime64[ns](1), float64(6), object(8)
memory usage: 426.0+ KB
```

In [ ]:
```
sex_dups = age_sex[['healthCode', 'BiologicalSex']].dropna().drop_duplicates(
)['healthCode'].value_counts()
age_sex.ix[sex_dups.head(6).index].set_index('dateTime', append=True)[['healt
hCode', 'BiologicalSex']]
```

In [ ]:
```
sex_dups = age_sex[['healthCode', 'CurrentAge']].dropna().drop_duplicates()['
healthCode'].value_counts()
age_sex.ix[sex_dups.head(6).index].set_index('dateTime', append=True)[['healt
hCode', 'CurrentAge']]
```

PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

Cohorts

In [29]:
```
# Gina classifications

cohort = pd.read_table('../analysis/cohorts_first.tsv',sep='\t')
cohort.columns = ['cohort','healthCode', 'gina']
cohort['gina'] = cohort['gina'].str.replace(' ', '')

baseline = cohort[cohort.cohort=='baseline']
baseline.name = 'baseline'

robust = cohort[cohort.cohort=='robust']
robust.name = 'robust'

milestone = cohort[cohort.cohort=='milestone']
milestone.name = 'milestone'
```

Iphone8 Check

In [35]:
```
#ALL DIFF HEALTHCODES ASSOCIATED WITH IPHONE8
nicole = pd.read_table('/Users/ers/Downloads/diff_data_healthCodes.tsv',sep='
')
nicole.head()
print len(df_filtered.ix['AsthmaDailyPrompt'][df_filtered.ix['AsthmaDailyProm
pt'].index.get_level_values(0).isin(nicole.healthCode)][daily_survey_q].index
.get_level_values(0).unique())
n = df_filtered.ix['AsthmaDailyPrompt'][df_filtered.ix['AsthmaDailyPrompt'].i
ndex.get_level_values(0).isin(nicole.healthCode)]
n.phoneInfo.value_counts()
```

PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

```
27
```

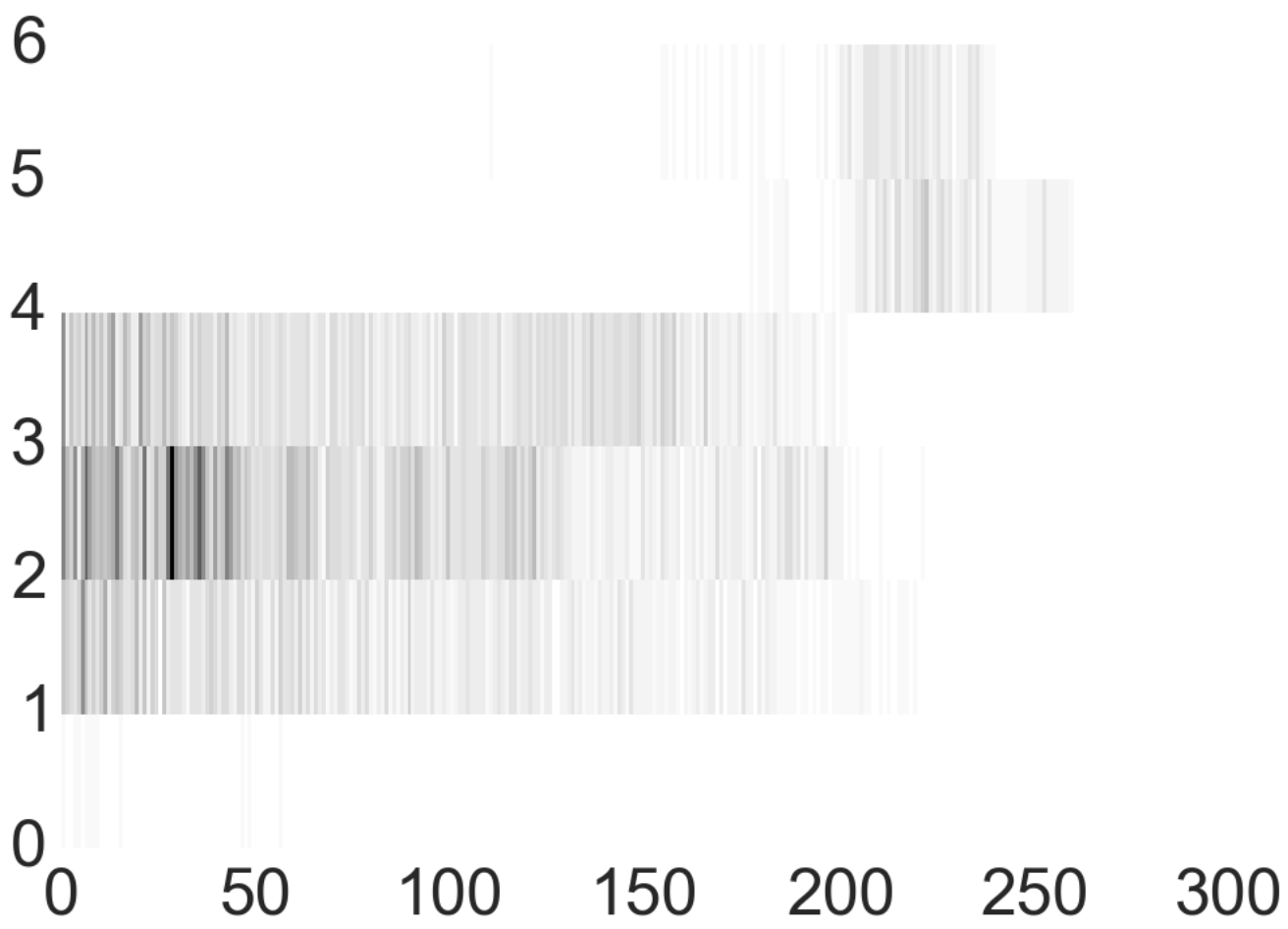
Out[35]:
```
iPhone 6           992
iPhone 6 Plus      778
iPhone 5s (GSM)    605
iPhone8,1          164
iPhone8,2          130
iPhone 5c (GSM)     11
Name: phoneInfo, dtype: int64
```

In [36]:
```
plt.pcolormesh(n.groupby('phoneInfo')['study_day'].value_counts().unstack().fillna(0).values)
```

Out[36]:
```
<matplotlib.collections.QuadMesh at 0x10d2d3210>
```

```
/Users/ers/anaconda/lib/python2.7/site-packages/matplotlib/collections.py:590: FutureWarning: elementwise comparison failed; returning scalar instead, but in the future will perform elementwise comparison
  if self._edgecolors == str('face'):
```

PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com



In [37]:
```
#Number of unique healthcodes for daily surveys
len(select_date(df_filtered.dropna(subset=[daily_survey_q], how='all'), 'date
', (2015,9,10)).ix['AsthmaDailyPrompt'].index.get_level_values(0).unique())
```

Out[37]: 6023

Baseline

Consensus, randomization

PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

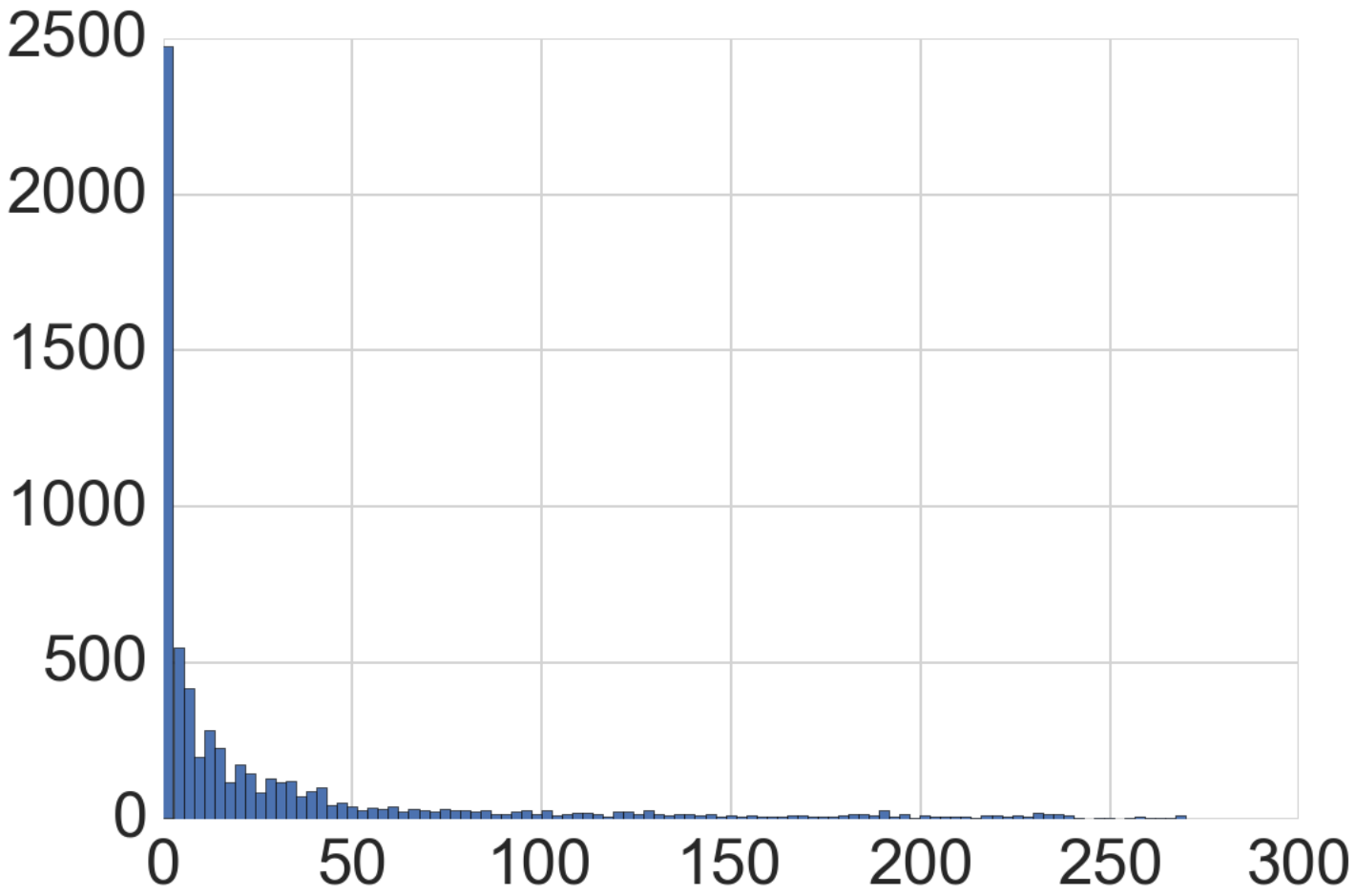
Daily Surveys

Daily Survey Functions

In [33]:
```
daily_survey_q = ['quick_relief_puffs', 'medicine', 'day_symptoms', \
                  'night_symptoms', 'peakflow', 'get_worse', \
                  'use_qr']

daily_prompt_data_cols = ['quick_relief_puffs', 'medicine', 'get_worse', \
                          'day_symptoms', 'night_symptoms', 'peakflow', \
                          'value', 'study_day', 'createdOn', 'dateTime',\
                         'use_qr']

table_columns['AsthmaDailyPrompt']


daily_survey_q_df = df_filtered.ix['AsthmaDailyPrompt'].dropna(subset=daily_s
urvey_q, how='all')

#LAST DAY FOR DAILY SURVEY
daily_survey_last_day = get_last_day(df_filtered, daily_survey_q)
daily_survey_last_day.hist(bins=100)
```

Out[33]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1126983d0>
```

PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com



PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

In [38]:
```
#DAILY SURVEY DATA ANY RESPONSES


def get_count(vector):
    if True in vector or False in vector:
        return [1 if i in [True, False] else 0 for i in vector]

    vector_counts = []
    for i in vector:
        if i == '[]':
            vector_counts.append(0)
            continue
        if i > -1:
            vector_counts.append(1)
            continue
        vector_counts.append(0)

    return vector_counts



daily_survey_q = ['quick_relief_puffs', 'medicine', 'day_symptoms', \
                  'night_symptoms', 'peakflow', 'get_worse', \
                  'use_qr', 'medicine_change']

daily_prompt_data_cols = ['quick_relief_puffs', 'medicine', 'get_worse', \
                          'day_symptoms', 'night_symptoms', 'peakflow', \
                          'value', 'study_day', 'createdOn', 'dateTime',\
                         'use_qr', 'medicine_change']


daily_bool_qs = ['night_symptoms', 'day_symptoms', 'use_qr', 'medicine_change
']
```

PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

```
daily_quant_qs = ['peakflow', 'quick_relief_puffs']
daily_set_qs = ['get_worse', 'medicine', 'createdOn']


#Drop columns with na value in all daily prompt question fields
#daily_prompt_data = df_filtered.ix['AsthmaDailyPrompt'].ix[gina_parsed.index
] #.dropna(subset=daily_survey_q, how='all')

daily_prompt_data = df_filtered.ix['AsthmaDailyPrompt']#.ix[nicole_clean_dail
y.x.unique()]
print len(daily_prompt_data.index.get_level_values(0).unique()), 1


#restrict data frame to useful columns
daily_prompt_data = daily_prompt_data[daily_prompt_data_cols]

print len(daily_prompt_data.index.get_level_values(0).unique())
daily_prompt_data = daily_prompt_data.dropna(subset=daily_survey_q, how='all'
)
print len(daily_prompt_data.index.get_level_values(0).unique())

%time daily_prompt_data_collapsed = handle_multiple_survey_answers(daily_prom
pt_data.set_index('study_day', append=True)[daily_survey_q + [ 'dateTime', 'c
reatedOn']], daily_bool_qs, daily_quant_qs, daily_set_qs)
len(daily_prompt_data_collapsed.index.get_level_values(0).unique())
daily_prompt_data_collapsed['date'] = pd.to_datetime([list(i)[0] for i in dai
ly_prompt_data_collapsed.dateTime_createdOn.values], unit='ms', infer_datetim
e_format=True)


daily_prompt_data_collapsed['date'] = pd.to_datetime([list(i)[0] for i in dai
ly_prompt_data_collapsed.dateTime_createdOn.values])
daily_prompt_data_collapsed['date'] = daily_prompt_data_collapsed['date'].dt.
```

PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

```
date


daily_prompt_data_collapsed['dailyQ_count'] = daily_prompt_data_collapsed[dai
ly_survey_q].apply(get_count).apply(np.sum, axis=1).values


print daily_prompt_data_collapsed.shape
print daily_prompt_data_collapsed[data_columns(table_columns['AsthmaDailyProm
pt']) + ['date']].reset_index().drop_duplicates().shape


# #daily_survey_q.remove('quick_relief_puffs')
# #daily_survey_q.remove('medicine')


#Create table of daily questions answered for each participant
daily_prompt_q_ans_count_df = daily_prompt_data_collapsed.reset_index().pivot
_table(index='healthCode', columns='study_day', values='dailyQ_count', fill_v
alue=0)


# daily_prompt_data['value'] = 1
# robust_users = daily_prompt_data.groupby('healthCode')['value'].sum()
# #robust_users.head()
# robust_users = robust_users[robust_users>4]
# robust_users.shape
```

PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

```
6529 1
6529
6515
CPU times: user 17.3 s, sys: 626 ms, total: 17.9 s
Wall time: 17.9 s
(83368, 20)
(83368, 11)
```

```
/Users/ers/anaconda/lib/python2.7/site-packages/ipykernel/__main__.py:86: Fut
ureWarning: sort(columns=....) is deprecated, use sort_values(by=.....)
```

In [39]:
```
len(daily_prompt_data.index.get_level_values(0).unique())
```

Out[39]:
```
6515
```

In [40]:
```
len(daily_prompt_data_collapsed.index.get_level_values(0).unique())
```

Out[40]:
```
6515
```

In [41]:
```
daily_prompt_data_collapsed.info()
```

PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

```
<class 'pandas.core.frame.DataFrame'>
MultiIndex: 83368 entries, (000152e4-c14d-4e0e-8bc4-d96e53acc868, 0) to (ffee
35de-8a9e-4105-a3fa-b9cd870cabe5, 224)
Data columns (total 20 columns):
dateTime_night_symptoms        83192 non-null datetime64[ns]
night_symptoms                 83192 non-null object
dateTime_day_symptoms          83132 non-null datetime64[ns]
day_symptoms                   83132 non-null object
dateTime_use_qr                83083 non-null datetime64[ns]
use_qr                         83083 non-null object
dateTime_medicine_change       67966 non-null datetime64[ns]
medicine_change                67966 non-null object
dateTime_peakflow              20326 non-null datetime64[ns]
peakflow                       20326 non-null float64
dateTime_quick_relief_puffs    23989 non-null datetime64[ns]
quick_relief_puffs             23989 non-null float64
get_worse                      70305 non-null object
dateTime_get_worse             70305 non-null object
medicine                       79492 non-null object
dateTime_medicine              79492 non-null object
createdOn                      83368 non-null object
dateTime_createdOn             83368 non-null object
date                           83368 non-null object
dailyQ_count                   83368 non-null int64
dtypes: datetime64[ns](6), float64(2), int64(1), object(11)
memory usage: 13.4+ MB
```

In [42]:
```
#daily_prompt_data_collapsed.to_csv('../analysis/daily_prompt_data_cleaned_20
15_12_04.tsv',sep='\t')
#!gzip ../analysis/daily_prompt_data_cleaned_2015_12_04.tsv
```

PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

In [ ]:
```
last_medicine_by_date = []
counter = 0
for i,n in df_filtered.reset_index('healthCode').groupby('date'):

    l = get_first_response(n, 'healthCode', 'dateTime', 'medicine',keep='last
')[['dateTime', 'medicine', 'study_day']]
    last_medicine_by_date.append(l)

last_medicine_by_date = pd.concat(last_medicine_by_date)
last_medicine_by_date.to_csv('../analysis/Daily_medicine_ans_by_date_last_res
ponse.tsv', sep='\t')
!gzip ../analysis/Daily_medicine_ans_by_date_last_response.tsv
```

Weekly Surveys

In [43]:
```
weekly_q = ['asthma_doc_visit', 'oral_steroids_when','emergency_room','asthma
_medicine',\
            'missed_work_days', 'missed_work', 'admission', 'prednisone_when',
'admitted_end',\
            'er_when', 'limitations', 'admitted_when', 'prednisone', 'oral_ste
roids', 'limitations_days',\
            'side_effects']

weekly_survey_last_day = get_last_day(df_filtered, weekly_q)
```

In [44]:
```
#Weekly Questions
data_columns(table_columns['AsthmaWeeklyPrompt'])
```

PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

```
Out[44]: ['limitations',
          'missed_work_days',
          'admitted_when',
          'prednisone',
          'missed_work',
          'asthma_doc_visit',
          'admission',
          'er_when',
          'prednisone_when',
          'oral_steroids',
          'admitted_end',
          'emergency_room',
          'asthma_medicine',
          'oral_steroids_when',
          'limitations_days',
          'side_effects']
```

PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

In [80]:

```
weekly_q = ['asthma_doc_visit', 'oral_steroids_when','emergency_room','asthma
_medicine',\
           'missed_work_days', 'missed_work', 'admission', 'prednisone_when',
'admitted_end',\
           'er_when', 'limitations', 'admitted_when', 'prednisone', 'oral_ste
roids', 'limitations_days',\
           'side_effects']

weekly_survey_df = df_filtered.ix['AsthmaWeeklyPrompt'][weekly_q + ['study_da
y', 'dateTime', 'createdOn']]

print weekly_survey_df.shape
weekly_survey_df = weekly_survey_df.dropna(subset=weekly_q, how='all')
print weekly_survey_df.shape

weekly_survey_df = weekly_survey_df.reset_index('healthCode').drop_duplicates
()

weekly_survey_df.set_index(['healthCode', 'study_day'], inplace=True)


weekly_bool_qs = ['limitations', 'oral_steroids',  'prednisone', 'missed_work
', 'asthma_doc_visit', \
                  'admission', 'emergency_room', 'asthma_medicine']

weekly_quant_qs = ['limitations_days']

weekly_set_qs = ['side_effects', 'oral_steroids_when', 'prednisone_when', 'ad
mitted_when', \
                  'er_when', 'missed_work_days', 'admitted_end', 'createdOn']
```

PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

```
%time weekly_survey_df = handle_multiple_survey_answers(weekly_survey_df[week
ly_q + [ 'dateTime', 'createdOn']], weekly_bool_qs, weekly_quant_qs, weekly_s
et_qs)
weekly_survey_df.shape
```

```
(13650, 19)
(13633, 19)
CPU times: user 4.88 s, sys: 101 ms, total: 4.98 s
Wall time: 4.99 s
```

```
/Users/ers/anaconda/lib/python2.7/site-packages/ipykernel/__main__.py:86: Fut
ureWarning: sort(columns=....) is deprecated, use sort_values(by=.....)
```

Out[80]:
```
(13007, 34)
```

In [81]:
```
#sanity check
print weekly_survey_df[data_columns(table_columns['AsthmaWeeklyPrompt'])].res
et_index().drop_duplicates().shape
```

```
(13007, 18)
```

In [82]:
```
#Sanity Check # People on Prednisone
len(weekly_survey_df[weekly_survey_df.prednisone==True].index.get_level_value
s(0).unique())
```

Out[82]:
```
474
```

In [386]:
```
df_filtered.xs('00087279-2303-4945-b91d-0736ba920576', level=1)[weekly_q + ['
createdOn']].dropna(how='all').info()
```

PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

```
<class 'pandas.core.frame.DataFrame'>
Index: 309 entries, AboutYou to aqiResponse
Data columns (total 17 columns):
asthma_doc_visit      21 non-null object
oral_steroids_when    1 non-null float64
emergency_room        21 non-null object
asthma_medicine       1 non-null object
missed_work_days      1 non-null object
missed_work           21 non-null object
admission             21 non-null object
prednisone_when       1 non-null float64
admitted_end          0 non-null object
er_when               0 non-null float64
limitations           20 non-null object
admitted_when         0 non-null float64
prednisone            19 non-null object
oral_steroids         1 non-null object
limitations_days      5 non-null float64
side_effects          20 non-null object
createdOn             309 non-null int64
dtypes: float64(5), int64(1), object(11)
memory usage: 43.5+ KB
```

PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

In [ ]:
```
#Weekly Survey Response Count
#weekly_survey_df = df_filtered.ix['AsthmaWeeklyPrompt'][data_columns(table_columns['AsthmaWeeklyPrompt']) + ['study_day', 'createdOn']]
#weekly_survey_df_study_day = weekly_survey_df['study_day']

#weekly_survey_df['study_day'] = weekly_survey_df_study_day
weekly_survey_df['weekly_q_count'] = weekly_survey_df[data_columns(table_columns['AsthmaWeeklyPrompt'])].apply(get_count).apply(np.sum, axis=1)
weekly_survey_df.weekly_q_count.hist(bins=100)


weekly_survey_df_q_count = weekly_survey_df.reset_index().pivot_table(index='healthCode', columns='study_day', values='weekly_q_count').fillna(0)
weekly_survey_df_q_count.head()
```

In [390]:
```
#weekly_survey_df.to_csv('../analysis/weekly_prompt_data_cleaned_2015_12_04.tsv', sep='\t')
#!gzip ../analysis/weekly_prompt_data_cleaned_2015_12_04.tsv
#!gzip ../analysis/weekly_prompt_data_cleaned_2015_11_30.tsv
```

In [84]:
```
weekly_survey_df_q_count_grace = weekly_survey_df_q_count[range(166,194)].apply(sum, axis=1)
weekly_survey_df_q_count_grace = weekly_survey_df_q_count_grace[weekly_survey_df_q_count_grace>0]
weekly_survey_df_q_count_grace.shape
```

Out[84]:
```
(249,)
```

Milestone Survey

PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

In [424]:
```
milestone = df_filtered.ix['Milestone'][data_columns(table_columns['Milestone
']) + ['date']]
print milestone.shape
milestone = milestone.reset_index().drop_duplicates().set_index('healthCode')
print milestone.shape
```

```
(136, 21)
(133, 21)
```

In [428]:
```
milestone.index.get_level_values(0).value_counts().head(3)
```

Out[428]:
```
260aa1c5-3112-45ee-86ee-c48e50c8b4c1    2
6858b8bd-0eae-40f5-abfb-8dc0659d3e82    2
edc5f27c-0bf1-4f7f-a997-7d9631e27025    2
Name: healthCode, dtype: int64
```

In [455]:
```
milestone.to_csv('../analysis/milestone_surveys_2015_12_04.tsv', sep='\t')
!gzip ../analysis/milestone_surveys_2015_12_04.tsv
```

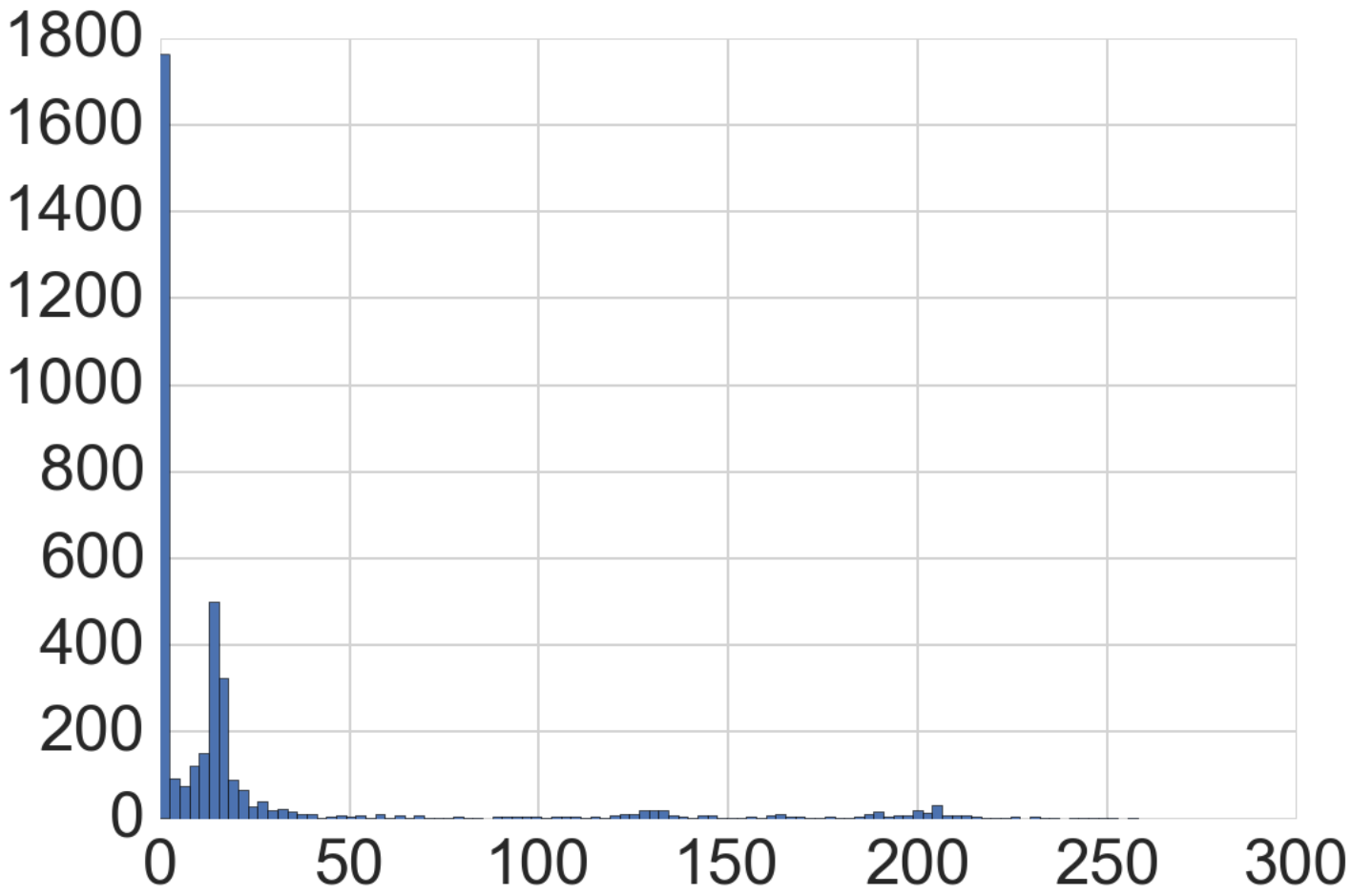
EQ Surveys

PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

In [85]:
```
EQ_5D_qs = ['health_today', 'pain', 'mobility', 'selfcare', 'usual_activities
', 'depression']

table_columns['EQ_5D']


eq_5d_df = df_filtered.ix['EQ_5D'].dropna(subset=EQ_5D_qs, how='all')


eq_5d_df_last_day = get_last_day(df_filtered, EQ_5D_qs)
eq_5d_df_last_day.hist(bins=100)
```

Out[85]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x140f1e6d0>
```

PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com



In [86]:
```
#Number of duplicate surveys on a single study day
sum(eq_5d_df[EQ_5D_qs + ['study_day', 'createdOn', 'value']].reset_index().dr
op_duplicates().groupby(['healthCode', 'study_day'])['value'].sum()>1)
```

Out[86]: 64

In [88]:
```
eq_5d_df[EQ_5D_qs + ['study_day', 'createdOn', 'value']].reset_index().groupb
y(['healthCode', 'study_day'])
```

Out[88]: <pandas.core.groupby.DataFrameGroupBy object at 0x11bd7d390>

PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

In [89]:
```
#Number of EQ Surveys
print eq_5d_df[EQ_5D_qs + ['study_day']].reset_index().groupby(['healthCode',
'study_day']).first().index.get_level_values(0).value_counts().sum()

#Number of Unique Participants Completing ≥ 1 EQ Surveys
print len(eq_5d_df[EQ_5D_qs + ['study_day']].dropna(how='all').index.unique()
)
```

```
4518
3806
```

In [90]:
```
eq_5d_df[EQ_5D_qs + ['study_day']].reset_index().groupby(['healthCode', 'stud
y_day']).first().index.get_level_values(0).value_counts().describe()
```

Out[90]:
```
count    3806.000000
mean        1.187073
std         0.573805
min         1.000000
25%         1.000000
50%         1.000000
75%         1.000000
max         8.000000
Name: healthCode, dtype: float64
```

Daily, Weekly, EQ Joined on healthCode and Study Date

In [ ]:
```
daily_weekly_df = daily_prompt_data_collapsed.join(weekly_survey_df, how='out
er', lsuffix='_daily', rsuffix='_weekly')
daily_weekly_df.head()
```

PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

In [90]:
```
daily_weekly_df.shape
```

Out[90]:
```
(79423, 51)
```

In [474]:
```
daily_weekly_df['value']=1
```

In [475]:
```
max(daily_weekly_df.groupby(level=[0])['value'].sum().value_counts().index)
```

Out[475]:
```
242
```

In [476]:
```
daily_weekly_df.info()
```

PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

```
<class 'pandas.core.frame.DataFrame'>
MultiIndex: 83563 entries, (000152e4-c14d-4e0e-8bc4-d96e53acc868, 0) to (ffee
35de-8a9e-4105-a3fa-b9cd870cabe5, 224)
Data columns (total 55 columns):
dateTime_night_symptoms        83192 non-null datetime64[ns]
night_symptoms                 83192 non-null object
dateTime_day_symptoms          83132 non-null datetime64[ns]
day_symptoms                   83132 non-null object
dateTime_use_qr                83083 non-null datetime64[ns]
use_qr                         83083 non-null object
dateTime_medicine_change       67966 non-null datetime64[ns]
medicine_change                67966 non-null object
dateTime_peakflow              20326 non-null datetime64[ns]
peakflow                       20326 non-null float64
dateTime_quick_relief_puffs    23989 non-null datetime64[ns]
quick_relief_puffs             23989 non-null float64
get_worse                      70305 non-null object
dateTime_get_worse             70305 non-null object
medicine                       79492 non-null object
dateTime_medicine              79492 non-null object
createdOn_daily                83368 non-null object
dateTime_createdOn_daily       83368 non-null object
date                           83368 non-null object
dailyQ_count                   83368 non-null float64
dateTime_limitations           12947 non-null datetime64[ns]
limitations                    12947 non-null object
dateTime_oral_steroids         486 non-null datetime64[ns]
oral_steroids                  486 non-null object
dateTime_prednisone            12665 non-null datetime64[ns]
prednisone                     12665 non-null object
dateTime_missed_work           12892 non-null datetime64[ns]
missed_work                    12892 non-null object
dateTime_asthma_doc_visit      12970 non-null datetime64[ns]
```

PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

```
asthma_doc_visit                12970 non-null object
dateTime_admission              12899 non-null datetime64[ns]
admission                       12899 non-null object
dateTime_emergency_room         12969 non-null datetime64[ns]
emergency_room                  12969 non-null object
dateTime_asthma_medicine        485 non-null datetime64[ns]
asthma_medicine                 485 non-null object
dateTime_limitations_days       2570 non-null datetime64[ns]
limitations_days                2570 non-null float64
side_effects                    12821 non-null object
dateTime_side_effects           12821 non-null object
oral_steroids_when              273 non-null object
dateTime_oral_steroids_when     273 non-null object
prednisone_when                 785 non-null object
dateTime_prednisone_when        785 non-null object
admitted_when                   36 non-null object
dateTime_admitted_when          36 non-null object
er_when                         188 non-null object
dateTime_er_when                188 non-null object
missed_work_days                510 non-null object
dateTime_missed_work_days       510 non-null object
admitted_end                    39 non-null object
dateTime_admitted_end           39 non-null object
createdOn_weekly                13007 non-null object
dateTime_createdOn_weekly       13007 non-null object
value                           83563 non-null int64
dtypes: datetime64[ns](15), float64(4), int64(1), object(35)
memory usage: 35.7+ MB
```

In [295]:
```
daily_weekly_df.createdOn_daily.values
```

PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com

Out[295]:
```
array(['1426100199000,1426040912000', '1426197315000',
       '1426300341000,1426371235000', ..., '1443832915000',
       '1444404443000', '1445618031000'], dtype=object)
```

In [478]:
```
daily_weekly_df['dateTime'] = [pd.to_datetime(int(str(i).split(',')[0]), unit
='ms') if 'nan' not in str(i) else np.NaN  for i in daily_weekly_df.createdOn
_daily.values]
```

In [480]:
```
daily_weekly_df.to_csv('../analysis/daily_weekly_prompt_data_collapsed_2015_1
2_04.tsv',sep='\t', date_format='%Y-%m-%d %H:%M:%S')
!gzip ../analysis/daily_weekly_prompt_data_collapsed_2015_12_04.tsv
```

PRO version Are you a developer? Try out the HTML to PDF API pdfcrowd.com