Cox proportional hazards - Study Attrition

Identify participant features that predict study attrition

Data pre-processed by amha_data_preprocessing.ipynb

Retention Analysis - One model with Age, Sex, Race, Education, Ethnicity, Gina, AgeOnset

Response Rate - logit(transform) 1) whole study period 2) survey / days in study

Table of Contents

'Cleaned Data'
'EQ Survey Counts'
'Weekly Survey Counts'
'Daily Survey Counts'
'Cohorts'
'Cox Proportional Hazards Analysis'
'Figure 2d: Study Retention By Enrollment Month'
'Figure 2e: Study Retention By Age'
'Suppl. Figure 2a: Predictor Correlation Heatmap '

'Suppl. Figure 2b: Response Rate Histogram'
'Supplementary Table 4a: univariate'
'Supplementary Table 4b: multivariate'
'Supplementary Table 4c: OLS'
'Supplementary Figure 3: Heatmaps'

Libraries

In [2]:
```
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=1)

rcParams['text.color'] = 'black'
rcParams['ytick.color'] = 'black'
rcParams['xtick.color'] = 'black'
rcParams['axes.facecolor'] = 'white'
rcParams['axes.edgecolor'] = 'white'
```

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

Statistical Functions

In [3]:
```
def construct_contingency_table(df, x, y):
    '''
    This function creates a contingency table
    using values from columns x and y in df
    '''
    return df.groupby([x,y])['conting_count'].sum().unstack()

def cramersV_from_df(df, x, y):
    '''
    This function returns cramersV using
    values from columns x and y in df
    '''
    from scipy import stats
    df['conting_count'] = 1
    contingency_df = df.groupby([x,y])['conting_count'].sum().unstack().value
s
    chi2, p, ddof, expected = stats.chi2_contingency(contingency_df)
    CV = np.sqrt(chi2 / len(df[x]) * (min(len(df[x].unique()), len(df[y].uniq
ue())) -1 ))
    return CV

def cramersV(contingency_df, x_sum, x_unique, y_unique):
    '''
    This function calculates cramersV
    from the values of a N x 2 contingency table table
    '''
    from scipy import stats
    chi2, p, ddof, expected = stats.chi2_contingency(contingency_df)

    CV = np.sqrt(chi2 / (x_sum * (min(x_unique, y_unique) -1 )))
    return CV
```

Functions

In [4]:
```
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 [5]:
```python
def get_count(vector):
    '''
    This function counts booleans
    '''
    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 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
```

In [7]:
```
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)]
```

In [8]:
```python
def select_date(df, date_col, year_month_day):
    '''
    This function returns a subset of the data
    with the date_col < year_month_day
    '''

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

Data

In [9]:
```python
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
```

In [10]:
```
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_columns[table]].dropna(how='all')

table_columns
```

```
Out[10]: {'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',
           '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',
           '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',
           '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',
           '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',
           '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',
           'recordId',
           'createdOn',
           'appVersion',
           'ROW_VERSION',
           'uploadDate',
           'aqiResponse.json.state_code',
           'phoneInfo',
           'ROW_ID']}
```

In [11]:
```
#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[11]:
```
[('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)]
```

Cohorts

'Table of Contents'

In [12]:
```
# 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'
```

Clean Synapse Data

'Table of Contents'

In [13]:
```
#SAGE DATA
df_filtered = pd.read_table('../analysis/sage_data_df_filtered_2015_12_04.txt
.gz',sep='\t', compression='gzip',\
                           index_col=['table', 'healthCode'])

df_filtered['study_entry'] = pd.to_datetime(df_filtered.study_entry)
df_filtered['date'] = pd.to_datetime(df_filtered.date)

#study entry for each healthcode
study_entry = select_date(df_filtered.ix['AsthmaDailyPrompt'], 'date', (2015,
9,10)).study_entry.dt.date
study_entry = study_entry.reset_index().drop_duplicates().set_index('healthCo
de')

#df_filtered = select_date(df_filtered, 'date', (2015,9,10))
#df_filtered = df_filtered[df_filtered.index.get_level_values(1).isin(baselin
e.healthCode)]
```

```
/Users/ers/anaconda2/lib/python2.7/site-packages/IPython/core/interactiveshel
l.py:2717: DtypeWarning: Columns (2,6,7,8,14,17,18,20,21,23,24,25,26,31,35,37
,42,43,48,49,55,56,58,59,62,66,67,68,74,75,77,80,81,83,84,85,89,91,98,99,100,
101,107,110,112,124,125,126,127,129,131,132,133) have mixed types. Specify dt
ype option on import or set low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

In [14]:
```
#Setting entry data to 3-23
entry_date = df_filtered.groupby(level=1)['date'].min()
entry_date= entry_date.reset_index().set_index(pd.DatetimeIndex(entry_date))
entry_date_daterange = entry_date['2015-3-23':'2015-3-29'].healthCode.unique(
)
entry_date_daterange.shape
```

Out[14]:
```
(310,)
```

In [15]:
```
#Baseline Data

baseline_data = pd.read_table('../analysis/baseline_first_survey_nicole_2015_
12.tsv.gz',sep='\t',\
                              compression='gzip', index_col='healthCode')
baseline_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 6363 entries, 000152e4-c14d-4e0e-8bc4-d96e53acc868 to ffee35de-8a9e-41
05-a3fa-b9cd870cabe5
Data columns (total 71 columns):
doc_times                               2183 non-null float64
nights                                  6192 non-null float64
emergency                               6191 non-null object
times_hospitalized                      6187 non-null object
miss_work                               6186 non-null float64
emergency_times                         701 non-null float64
age_when_diagnosed                      6055 non-null float64
symptoms                                6194 non-null float64
hospitalized_times                      394 non-null float64
intubated                               6187 non-null float64
createdOn_His_data_2015_12_08           6240 non-null object
limited activity                        6193 non-null float64
seen_doc                                6188 non-null object
oral steroids                           6162 non-null float64
past_month_quick_relief                 5861 non-null float64
flovent_hfa_dose                        361 non-null float64
advair_diskus_dose                      950 non-null float64
symbicort_dose                          662 non-null float64
controlmed                              3420 non-null float64
createdOn_Med_data_2015_12_08           5898 non-null object
quick_relief                            5651 non-null object
advair_hfa_dose                         211 non-null float64
qvar_dose                               345 non-null float64
asmanex_dose                            148 non-null float64
steroid_dose                            515 non-null float64
daily_controller_medication             3887 non-null object
daily_yes                               5699 non-null float64
alvesco_dose                            48 non-null float64
daily_inhaled_medicine                  3435 non-null float64
```

```
flovent_diskus_dose                     72 non-null float64
prescribed_asthma_control_medication    5860 non-null float64
pulmicort_dose                          171 non-null float64
dulera_dose                             362 non-null float64
steroid_which                           3879 non-null float64
chronic_pulmonary_diesease              3749 non-null object
heart_attack                            3767 non-null object
createdOn_MedH_data_2015_12_08          3775 non-null object
peripheral                              3753 non-null object
arthritis                               3751 non-null object
leukemia                                3748 non-null object
tumor                                   3759 non-null float64
tested                                  3753 non-null object
stroke                                  3757 non-null float64
tissue                                  3758 non-null object
other_lung_disease                      3753 non-null object
malignant_lymphoma                      3758 non-null object
ulcer                                   3757 non-null object
Congestive                              3770 non-null object
dementia                                3761 non-null object
allergic_to                             3584 non-null object
kidney                                  3750 non-null object
liver                                   3756 non-null float64
smoking_status                          4244 non-null float64
health_insurance                        4227 non-null float64
race                                    4249 non-null object
smoking_years                           930 non-null float64
ethnicity                               4185 non-null float64
Income                                  4231 non-null float64
createdOn_You_data_2015_12_08           4274 non-null object
education                               4234 non-null float64
avg_cigarettes                          920 non-null float64
asthma_gets_worse_with                  4472 non-null object
```

```
lung_function                          4467 non-null float64
troubles_about_asthma                  4460 non-null object
peak_flow                              765 non-null float64
flu_prompt                             2223 non-null object
aap_prompt                             3538 non-null object
flushot                                4468 non-null object
createdOn_YouA_data_2015_12_08         4494 non-null object
asthma_control                         4416 non-null object
plan                                   4471 non-null object
dtypes: float64(39), object(32)
memory usage: 3.5+ MB
```

In [16]:
```
#Age & Sex df
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.sort(columns='createdOn', inplace=True)
age_sex.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 3408 entries, 5288ddee-c1a6-479c-8dc2-307a96ca26df to 094e479f-0d0e-41
9a-9ddf-308c225105b8
Data columns (total 14 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
dtypes: float64(6), object(8)
memory usage: 399.4+ KB
```

```
/Users/ers/anaconda2/lib/python2.7/site-packages/ipykernel/__main__.py:7: Fut
ureWarning: sort(columns=....) is deprecated, use sort_values(by=.....)
```

In [17]:

```
#Biological Sex df
biosex = age_sex[['healthCode', 'BiologicalSex', 'createdOn']].dropna(how='al
l').drop_duplicates(subset=['healthCode', 'BiologicalSex'])
biosex.sort('createdOn', inplace=True)
biosex = biosex.groupby('healthCode')['BiologicalSex'].value_counts().reset_i
ndex().drop_duplicates('healthCode').set_index('healthCode')['BiologicalSex']
```

```
/Users/ers/anaconda2/lib/python2.7/site-packages/ipykernel/__main__.py:3: Fut
ureWarning: sort(columns=....) is deprecated, use sort_values(by=.....)
  app.launch_new_instance()
```

In [18]:
```
#Age df
age = age_sex[['healthCode', 'CurrentAge', 'createdOn']].dropna(how='all').dr
op_duplicates(subset=['healthCode', 'CurrentAge'])
age = age[~age.CurrentAge.isnull()]
age.sort('createdOn', inplace=True)
age = age.groupby('healthCode')['CurrentAge'].value_counts().reset_index().dr
op_duplicates('healthCode').set_index('healthCode')['CurrentAge']
print age.shape
```

```
(2109,)
```

```
/Users/ers/anaconda2/lib/python2.7/site-packages/ipykernel/__main__.py:4: Fut
ureWarning: sort(columns=....) is deprecated, use sort_values(by=.....)
```

Daily & Weekly Surveys Collapsed

In [19]:
```
#SAGE DATA
daily_weekly_df = pd.read_table('../analysis/daily_weekly_prompt_data_collapsed_2015_12_04.tsv.gz',\
                            sep='\t', compression='gzip', index_col=['healthCode', 'study_day'])

daily_weekly_df = daily_weekly_df.join(study_entry, how='left')

daily_weekly_df['dateTime'] = daily_weekly_df.dateTime.map(pd.to_datetime)  # entered study
daily_weekly_df['date'] = pd.to_datetime(daily_weekly_df.dateTime).dt.date
daily_weekly_df['study_entry'] = pd.to_datetime(daily_weekly_df.study_entry, unit='ms')
```

```
/Users/ers/anaconda2/lib/python2.7/site-packages/IPython/core/interactiveshell.py:2717: DtypeWarning: Columns (40,42,47,48,52,53,54) have mixed types. Specify dtype option on import or set low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

In [20]:
```
len(daily_weekly_df[daily_weekly_df.date>pd.datetime(2015,3,3).date()].index.get_level_values(0).unique())
```

Out[20]:
```
6515
```

Cox Proportinal Hazards

'Table of Contents'

In [21]:
```
days_till_2015_09_09 = pd.datetime(2015,9,8).date() - daily_weekly_df.reset_i
ndex().drop_duplicates(subset=['healthCode', 'study_entry']).set_index('healt
hCode')['study_entry'].dropna()
days_till_2015_09_09 = days_till_2015_09_09.dt.days.groupby(level=0).max().dr
opna()
days_till_2015_09_09.name = 'days_till_2015_09_09'
days_till_2015_09_09.shape

#restricting analysis to those in study for at least 180 days
days_till_2015_09_09_180days = days_till_2015_09_09[days_till_2015_09_09>179]
days_till_2015_09_09_180days.name = 'days_till_2015_09_09'
```

In [22]:
```
#More efficient way to find censored healthCodes
daily_weekly_df_ts = daily_weekly_df.reset_index().set_index(pd.DatetimeIndex
(daily_weekly_df.dateTime))
censored_healthcodes = daily_weekly_df_ts['2015-8-27':'2015-9-8'].healthCode.
unique()
print censored_healthcodes.shape, 'censored healthCodes'

entered_before_june = daily_weekly_df_ts[:'2015-6-8'].healthCode.unique()
print entered_before_june.shape, 'entered before June'
```

```
(260,) censored healthCodes
(5768,) entered before June
```

```
/Users/ers/anaconda2/lib/python2.7/site-packages/pandas/indexes/base.py:3027:
FutureWarning: In the future, 'NAT <= x' and 'x <= NAT' will always be False.
  result = func(np.asarray(other))
```

Creation of dataframes for Multivariate Cox Analysis

In [23]:
```
import lifelines
from lifelines import CoxPHFitter
from lifelines.datasets import load_regression_dataset
from lifelines.utils import k_fold_cross_validation
pd.set_option('display.precision',4)

def study_entry_month_calendar(date):
    if date.day > 8:  #march 10
        return date.month - 3
    return date.month - 4

regression_dataset = load_regression_dataset()
cf = CoxPHFitter(normalize=False)



aboutyou = baseline_data.copy()
study_entry_month = df_filtered.study_entry.map(study_entry_month_calendar).r
eset_index(level=0, drop=True)
study_entry_month = study_entry_month.reset_index().drop_duplicates().set_ind
ex('healthCode')
study_entry_month.columns = ['study_entry_month']



study_entry_week = df_filtered.date - pd.datetime(2015,03,9)
study_entry_week = study_entry_week.dt.days
study_entry_week = study_entry_week.map(lambda x: 180 if x > 180 else x)
study_entry_week = pd.cut(study_entry_week,bins=range(0,365,14), retbins=Fals
e, labels=False)
study_entry_week.name = 'study_entry_week'
study_entry_week.reset_index(0, drop=True, inplace=True)
study_entry_week = study_entry_week.groupby(level=0).min().astype(float)
```

```
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_survey_standardqs = [ 'day_symptoms', 'night_symptoms', 'peakflow', \
                           'get_worse', 'use_qr']


cox_cols = [ 'gina','last_survey_censored',   \
            'event_observed', 'education', \
             'age_when_diagnosed', 'Income', \
             'health_insurance', 'ethnicity',\
            'CurrentAge', 'BiologicalSex',\
           'study_entry_month']


print 'Columns in multivariate CoxPH Model\n', cox_cols


#int values
last_study_day = daily_weekly_df[daily_survey_standardqs].dropna(how='all').r
eset_index().groupby('healthCode')['study_day'].max()
last_study_day = last_study_day.map(lambda x: 180 if x > 179 else x)
last_study_day.name = 'last_survey_censored'
```

```
#restricting to entrants before June
last_study_day = last_study_day[last_study_day.index.isin(entered_before_june
)]


#print last_study_day.shape
last_study_day = aboutyou.join(last_study_day)  #joining about you data
last_study_day = last_study_day.join(study_entry_month)  #joining study entry
month data
last_study_day = last_study_day.join(biosex)  #joining biological sex data
last_study_day = last_study_day.join(age, how='inner') #joining age data
#print last_study_day.shape, 'after age sex joining'

last_study_day = last_study_day[last_study_day.index.isin(days_till_2015_09_0
9.index)]  #restricting data up to 9/9/2015
#print last_study_day.shape
last_study_day = last_study_day.join(days_till_2015_09_09, how='inner')


#restricting to robust users
last_study_day = last_study_day.join(robust.drop_duplicates().set_index('heal
thCode')['gina'], how='inner')


#adding censoring data
last_study_day['event_observed'] = [False if i in censored_healthcodes else T
rue for i in last_study_day.index]

#print last_study_day.shape
cox_df = last_study_day[cox_cols].dropna(how='any')

#print cox_df.shape
```

```
replacement_dict = {'gina':{'Uncontrolled':2, 'PartlyControlled':1, 'WellCont
rolled':0},\
                    'education':{'1':1, '2':1, '3':1, '4':2, '5':2, '6':3, '7
':3, '8':np.NaN},\
                    'BiologicalSex':{'Female':0, 'Male':1},\
                   'health_insurance':{'3':0, '1':1, '2':1, '4':np.NaN},}
    #Nicoles: 'education':{'1':1, '2':1, '3':2, '4':3, '5':3, '6':4, '7':4, '
8':np.NaN}

# replacement_dict = {'gina':{'Uncontrolled':2, 'PartlyControlled':1, 'WellCo
ntrolled':0},\
#                      'BiologicalSex':{'Female':0, 'Male':1}}

cox_df['education'] = cox_df['education'].astype(int).astype(str)
cox_df['health_insurance'] = cox_df['health_insurance'].astype(int).astype(st
r)
cox_df.replace(to_replace=replacement_dict, inplace=True)
cox_df['education'] = cox_df['education'].astype(float)




print
print 'Sample size before filtering on missing education, income, health insu
rance, ethnicity, and current age\n', cox_df.shape
print
cox_df = cox_df[(cox_df.education < 8) & \
                (cox_df.Income < 6) & \
                (cox_df.health_insurance < 4) & \
               (cox_df.ethnicity <3) & \
               (cox_df.CurrentAge > 0)] #

print 'Sample size after filtering on missing education, income, health insur
ance, ethnicity, and current age\n', cox_df.shape
```

```
print


cox_df['Current_Age_binned'] = pd.cut(cox_df['CurrentAge'], range(0,100,10),
labels=False, retbins=False)
#cox_df = cox_df[cox_df.age_when_diagnosed<20]

cox_df[list( set(cox_df.columns) - set(cox_cols)  )]



#cox_df = cox_df.join(study_entry_week, how='inner')
#cox_cols.append('study_entry_week')

scores = k_fold_cross_validation(cf, cox_df, 'last_survey_censored', event_co
l='event_observed', k=5)

#Five fold cross validation iteration scores
# print
# print scores
# print mean(scores)
# print std(scores)
```

```
Columns in multivariate CoxPH Model
['gina', 'last_survey_censored', 'event_observed', 'education', 'age_when_dia
gnosed', 'Income', 'health_insurance', 'ethnicity', 'CurrentAge', 'Biological
Sex', 'study_entry_month']

Sample size before filtering on missing education, income, health insurance,
ethnicity, and current age
(620, 11)

Sample size after filtering on missing education, income, health insurance, e
thnicity, and current age
(537, 11)
```

In [24]:
```
#Fitting Multivariate Cox PH model

m = cf.fit( cox_df[cox_cols], \
               'last_survey_censored', event_col='event_observed',\
              ) #
m_summary = m.summary
m_summary['abs_z'] = m_summary['z'].map(np.abs)
m_summary.sort(columns=['abs_z'], ascending=False, inplace=True)
del m_summary['abs_z']
#m_summary = m_summary.apply(np.round, args=[3])

#exp(coef) is the hazard ratio
#coef is the increase in having the event for every 1 unit increase in X
#    therefore if coef is negative then the predictor decreases the hazard of
an outcome
```

```
/Users/ers/anaconda2/lib/python2.7/site-packages/ipykernel/__main__.py:6: Fut
ureWarning: sort(columns=....) is deprecated, use sort_values(by=.....)
```

```
In [25]: #CramersV Correlations between categorical variables
         cox_cols_categ = ['health_insurance', \
                           'ethnicity', 'BiologicalSex']
         cramersV_df = []
         for categ1 in cox_cols_categ:
             cramersV_df.append([])
             for categ2 in cox_cols_categ:
                 cramersV_df[-1].append(cramersV_from_df(cox_df, categ1, categ2))
         cramersV_df =  pd.DataFrame(cramersV_df, columns=cox_cols_categ, index=cox_co
         ls_categ)
         cramersV_df
```

Out[25]:

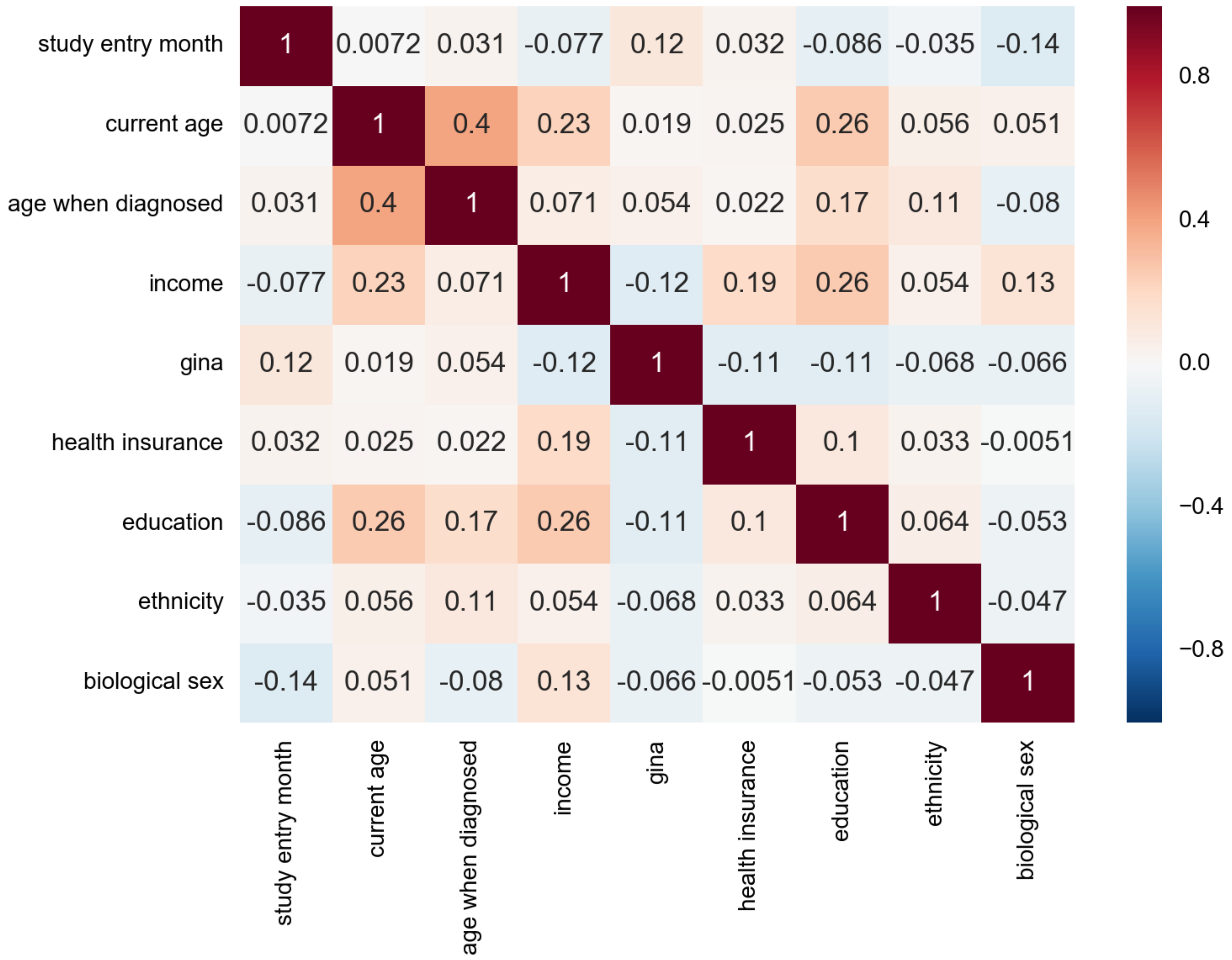
	health_insurance	ethnicity	BiologicalSex
health_insurance	NaN	0.0167	0.0061
ethnicity	0.0167	NaN	0.0413
BiologicalSex	0.0061	0.0413	NaN

Supplementary Figure 2a

Predictor Correlation Heatmaps

'Table of Contents'

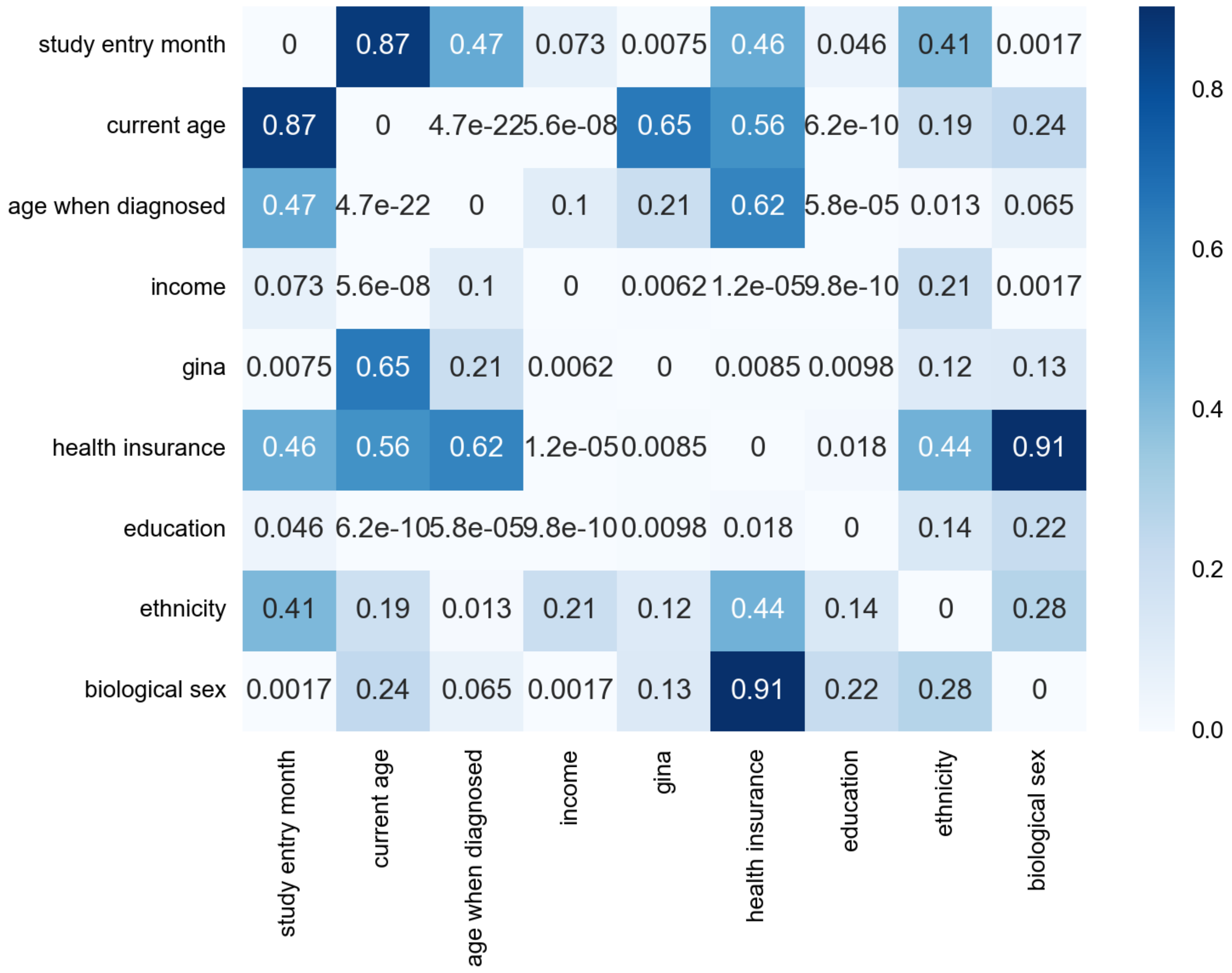
In [26]:
```
#Covariance Matrix for Numeric Predictors (Standardized)
#Pearson's correlation
cov_df_numeric = cox_df[['study_entry_month', 'CurrentAge', 'age_when_diagnos
ed', 'Income', 'gina', 'health_insurance','education', 'ethnicity', 'Biologic
alSex']]
cov_df_numeric = cov_df_numeric.corr()
cov_df_numeric.columns = [i.replace('_', ' ').replace('CurrentAge', 'current
age').replace('BiologicalSex', 'biological sex').lower() for i in cov_df_nume
ric.columns]
cov_df_numeric.index = [i.replace('_', ' ').replace('CurrentAge', 'current ag
e').replace('BiologicalSex', 'biological sex').lower() for i in cov_df_numeri
c.columns]

cov_df_numeric_heatmap = seaborn.heatmap(cov_df_numeric, annot=True)
```



	study entry month	current age	age when diagnosed	income	gina	health insurance	education	ethnicity	biological sex
study entry month	1	0.0072	0.031	-0.077	0.12	0.032	-0.086	-0.035	-0.14
current age	0.0072	1	0.4	0.23	0.019	0.025	0.26	0.056	0.051
age when diagnosed	0.031	0.4	1	0.071	0.054	0.022	0.17	0.11	-0.08
income	-0.077	0.23	0.071	1	-0.12	0.19	0.26	0.054	0.13
gina	0.12	0.019	0.054	-0.12	1	-0.11	-0.11	-0.068	-0.066
health insurance	0.032	0.025	0.022	0.19	-0.11	1	0.1	0.033	-0.0051
education	-0.086	0.26	0.17	0.26	-0.11	0.1	1	0.064	-0.053
ethnicity	-0.035	0.056	0.11	0.054	-0.068	0.033	0.064	1	-0.047
biological sex	-0.14	0.051	-0.08	0.13	-0.066	-0.0051	-0.053	-0.047	1

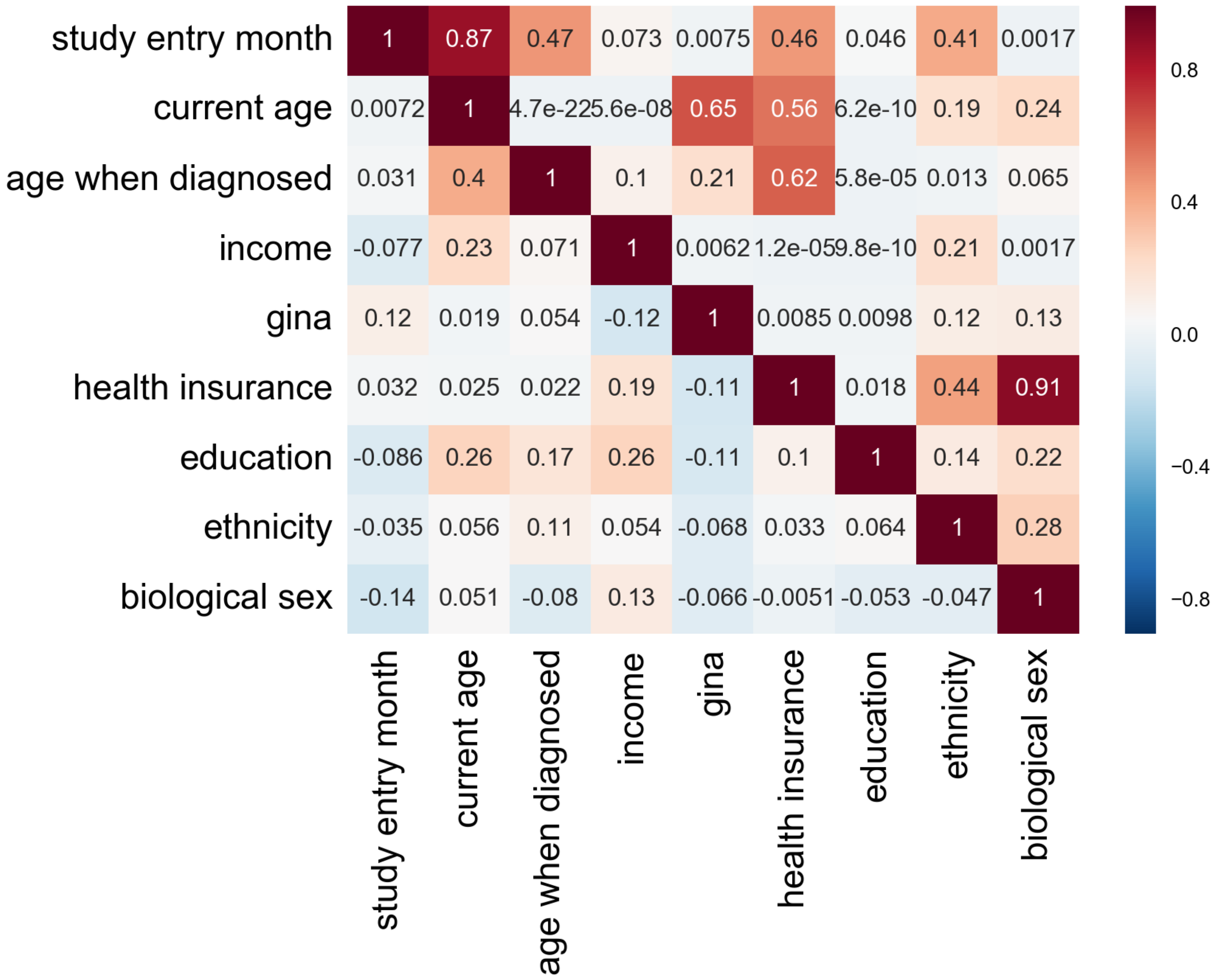
In [27]:
```
#Pearson's p-value
from scipy import stats
cov_df = cox_df[['study_entry_month', 'CurrentAge', 'age_when_diagnosed', 'In
come', 'gina', 'health_insurance','education', 'ethnicity', 'BiologicalSex']]

cor_pval = []
for c1 in cov_df.columns:
    cor_pval.append([])
    for c2 in cov_df.columns:
        cor_pval[-1].append(stats.pearsonr(cov_df[c1], cov_df[c2])[-1])
cor_pval = pd.DataFrame(cor_pval, columns=cov_df.columns, index=cov_df.column
s)
cor_pval.columns = [i.replace('_', ' ').replace('CurrentAge', 'current age').
replace('BiologicalSex', 'biological sex').lower() for i in cor_pval.columns]
cor_pval.index = [i.replace('_', ' ').replace('CurrentAge', 'current age').re
place('BiologicalSex', 'biological sex').lower() for i in cor_pval.columns]
seaborn.heatmap(cor_pval, annot=True, cmap='Blues')
```

Out[27]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x120a56d90>
```


study entry month 0 0.87 0.47 0.073 0.0075 0.46 0.046 0.41 0.0017
current age 0.87 0 4.7e-22 5.6e-08 0.65 0.56 6.2e-10 0.19 0.24
age when diagnosed 0.47 4.7e-22 0 0.1 0.21 0.62 5.8e-05 0.013 0.065
income 0.073 5.6e-08 0.1 0 0.0062 1.2e-05 9.8e-10 0.21 0.0017
gina 0.0075 0.65 0.21 0.0062 0 0.0085 0.0098 0.12 0.13
health insurance 0.46 0.56 0.62 1.2e-05 0.0085 0 0.018 0.44 0.91
education 0.046 6.2e-10 5.8e-05 9.8e-10 0.0098 0.018 0 0.14 0.22
ethnicity 0.41 0.19 0.013 0.21 0.12 0.44 0.14 0 0.28
biological sex 0.0017 0.24 0.065 0.0017 0.13 0.91 0.22 0.28 0
study entry month
current age
age when diagnosed
income
gina
health insurance
education
ethnicity
biological sex
0.8
0.6
0.4
0.2
0.0

In [28]:
```
#Merged Figure

def triangle_merge(df_triu, df_tril):
    '''
    Combines values from upper triangle
    and lower triangle from dfs
    '''
    triu = pd.DataFrame(np.triu(df_triu.values))
    triu.replace(0.0, np.NaN, inplace=True)
    tril = pd.DataFrame(np.tril(df_tril))
    tril.replace(0.0, np.NaN, inplace=True)

    combined_df = triu.combine_first(tril)
    combined_df.columns = df_triu.columns
    combined_df.index = df_triu.index
    return combined_df

combined_df = triangle_merge(cor_pval, cov_df_numeric)
seaborn.heatmap(combined_df.replace(0.0, 1.0), annot=True,  center=0.05)
plt.xticks(fontsize='x-large')
plt.yticks(fontsize='x-large')
#plt.tight_layout()
#plt.savefig('/Users/ers/Dropbox/feasibility_figures_ERS/CorrelationMatrix.sv
g', dpi=600, format='svg')
```

Out[28]:
```
(array([ 0.5,  1.5,  2.5,  3.5,  4.5,  5.5,  6.5,  7.5,  8.5]),
 <a list of 9 Text yticklabel objects>)
```


study entry month 1 0.87 0.47 0.073 0.0075 0.46 0.046 0.41 0.0017
current age 0.0072 1 4.7e-22 5.6e-08 0.65 0.56 6.2e-10 0.19 0.24
age when diagnosed 0.031 0.4 1 0.1 0.21 0.62 5.8e-05 0.013 0.065
income -0.077 0.23 0.071 1 0.0062 1.2e-05 9.8e-10 0.21 0.0017
gina 0.12 0.019 0.054 -0.12 1 0.0085 0.0098 0.12 0.13
health insurance 0.032 0.025 0.022 0.19 -0.11 1 0.018 0.44 0.91
education -0.086 0.26 0.17 0.26 -0.11 0.1 1 0.14 0.22
ethnicity -0.035 0.056 0.11 0.054 -0.068 0.033 0.064 1 0.28
biological sex -0.14 0.051 -0.08 0.13 -0.066 -0.0051 -0.053 -0.047 1
study entry month
current age
age when diagnosed
income
gina
health insurance
education
ethnicity
biological sex
0.8
0.4
0.0
−0.4
−0.8

Response Rate Histogram

Supplementary Figure 2b

Construction of dataframe for Ordinary Least Squares Model

Data also used for Supplemental Table 4c

'Table of Contents'

In [30]:
```
#DATAFRAME CONSTRUCTION FOR ORDINARY LEAST SQUARES MODEL


import numpy as np
import statsmodels.api as sm
import statsmodels.formula.api as smf
from scipy.special import logit

daily_survey_standardqs = [ 'day_symptoms', 'night_symptoms', 'peakflow', \
                           'get_worse', 'use_qr']




#number of days in study fory each healthcode
days_in_study_2015_9_9 = select_date(daily_weekly_df, 'date', (2015,9,8))

days_in_study_2015_9_9 = days_in_study_2015_9_9.study_entry.dt.date
days_in_study_2015_9_9 = pd.to_datetime(days_in_study_2015_9_9, unit='ms')
days_in_study_2015_9_9 = pd.datetime(2015,9,8) - days_in_study_2015_9_9
days_in_study_2015_9_9 = days_in_study_2015_9_9.dt.days
days_in_study_2015_9_9.name = 'days_in_study_2015_9_9'
days_in_study_2015_9_9.sort(inplace=True, ascending=False)
days_in_study_2015_9_9 = days_in_study_2015_9_9.reset_index().drop_duplicates
().set_index('healthCode')


days_till_2015_09_09 = pd.datetime(2015,9,8).date() - daily_weekly_df['study_
entry'].dropna()
days_till_2015_09_09 = days_till_2015_09_09.reset_index('healthCode').drop_du
plicates().set_index('healthCode')
days_till_2015_09_09_180days = days_till_2015_09_09[days_till_2015_09_09.stud
y_entry.dt.days>90]
days_till_2015_09_09_180days.columns = ['days_till_2015_09_09']
```

```
days_till_2015_09_09_180days['days_till_2015_09_09'] = 184 - days_till_2015_0
9_09_180days['days_till_2015_09_09'].dt.days
days_till_2015_09_09_180days.shape

#DAILY SURVEYS, AT LEAST 1 QUESTION ANSWERED
daily_weekly_df_response = select_date(daily_weekly_df, 'date', (2015,9,8))
#restrict to data before 9/19/2015

#calculate responses per healthcode
daily_survey_response_days = daily_weekly_df_response.query('study_day < 181'
).dropna(subset=[daily_survey_standardqs], how='all').groupby(level=0)['value
'].sum()

#join days in study
daily_survey_response_days = pd.DataFrame(daily_survey_response_days).join(da
ys_in_study_2015_9_9, how='inner')

#calculate rate #days responding / #days enrolled
daily_survey_response_days['ratio'] = daily_survey_response_days['value'].ast
ype(float).divide(daily_survey_response_days['days_in_study_2015_9_9'].values
, axis=0)
daily_survey_response_days = daily_survey_response_days.reset_index().drop_du
plicates(['healthCode', 'ratio']).set_index('healthCode')
daily_survey_response_days.head()



cohort_response_rate = daily_survey_response_days.join(aboutyou)
cohort_response_rate = cohort_response_rate.join(robust.set_index('healthCode
')['gina'], how='inner')

cohort_response_rate['intercept'] = 1
```

```
cohort_response_rate = cohort_response_rate.join(days_till_2015_09_09_180days
, how='inner')



replacement_dict = {'gina':{'Uncontrolled':2, 'PartlyControlled':1, 'WellCont
rolled':0},\
                    'education':{'1':1, '2':1, '3':1, '4':2, '5':2, '6':3, '7
':3, '8':np.NaN},\
                    'BiologicalSex':{'Female':0, 'Male':1},\
                   'health_insurance':{'3.0':0, '1.0':1, '2.0':1, '4.0':np.Na
N},}

cohort_response_rate = cohort_response_rate.dropna(subset=['health_insurance'
])
cohort_response_rate['education'] = cohort_response_rate['education'].astype(
str)
cohort_response_rate['health_insurance'] = cohort_response_rate['health_insur
ance'].astype(str)
cohort_response_rate.replace(to_replace=replacement_dict, inplace=True)
cohort_response_rate['education'] = cohort_response_rate['education'].astype(
float)#.astype(str)


cohort_response_rate = cohort_response_rate[(cohort_response_rate.education<8
) ]
cohort_response_rate = cohort_response_rate.join(study_entry_month)

#DATA NORMALIZATION
cohort_response_rate['logit_response_rate'] = cohort_response_rate['ratio'].m
ap(logit)
cohort_response_rate['logit_response_rate'] = cohort_response_rate['logit_res
ponse_rate'].map(lambda x: 1.0 if x > 1.0 else x)

def std_scale(x):

    mean_x = np.mean(x)
    std_x = np.std(x)
    return [(i - mean_x) / std_x for i in x]

cohort_response_rate = cohort_response_rate[cohort_response_rate.index.isin(c
ox_df.index)]
cohort_response_rate = cohort_response_rate.join(age)
cohort_response_rate['Current_Age_binned'] = pd.cut(cohort_response_rate['Cur
rentAge'], range(0,100,10), labels=False, retbins=False)
cohort_response_rate = cohort_response_rate.join(biosex)

for c in ['education', 'age_when_diagnosed', 'gina', 'CurrentAge', 'ethnicity
', 'Income', 'health_insurance','study_entry_month']:
    cohort_response_rate[c + '_stdscale'] = cohort_response_rate[[c]].apply(s
td_scale)



results = smf.ols('logit_response_rate ~ I(study_entry_month) + I(CurrentAge)
+ C(health_insurance) + C(ethnicity) + C(BiologicalSex) + I(education_stdscal
e) ', \
                                  data=cohort_response_rate[cohort_response_r
ate.index.isin(cox_df.index)]).fit()

results.summary()
```

```
/Users/ers/anaconda2/lib/python2.7/site-packages/ipykernel/__main__.py:21: Fu
tureWarning: sort is deprecated, use sort_values(inplace=True) for INPLACE so
rting
/Users/ers/anaconda2/lib/python2.7/site-packages/ipykernel/__main__.py:29: Se
ttingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/st
able/indexing.html#indexing-view-versus-copy
```

Out[30]:

OLS Regression Results

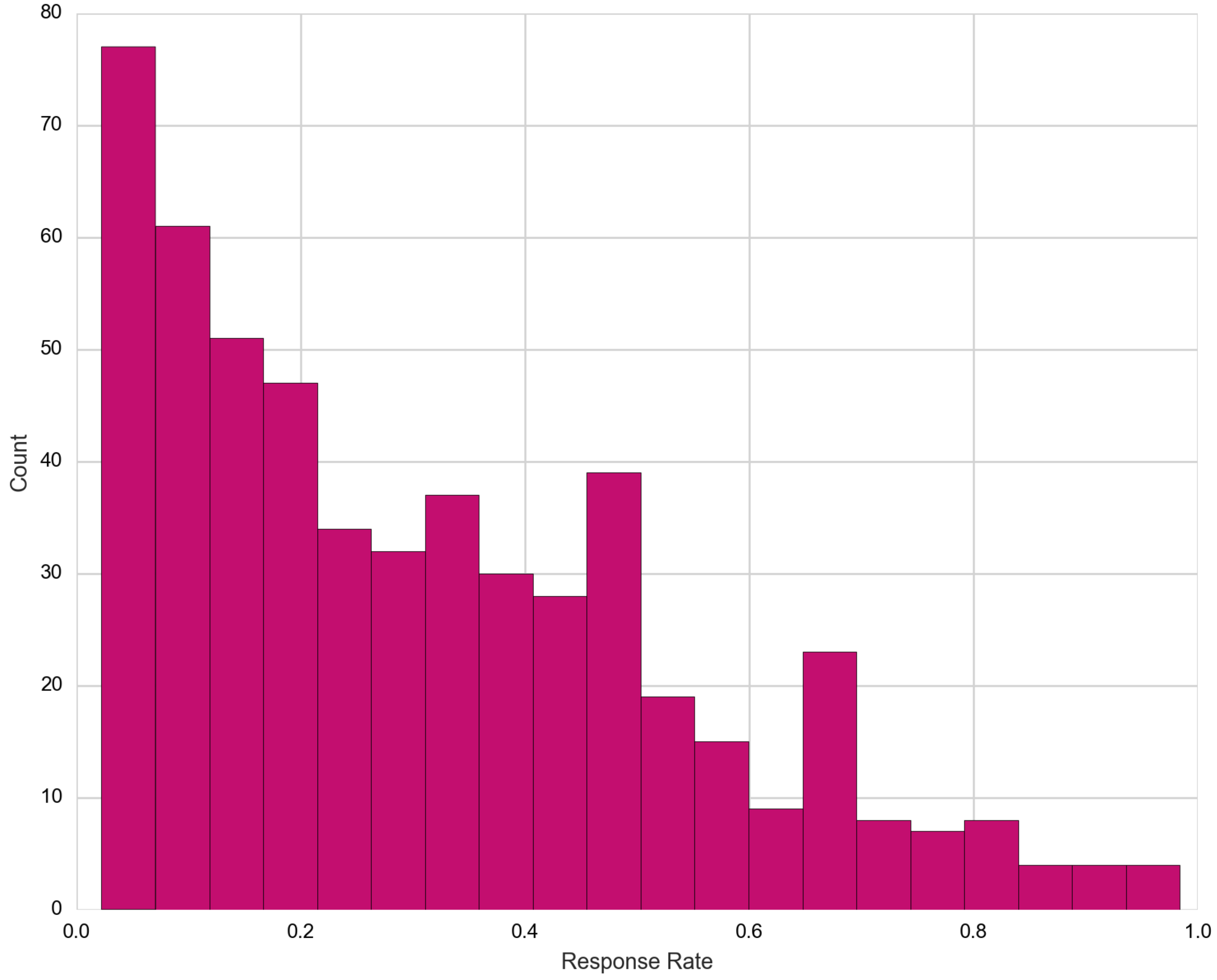
Dep. Variable:	logit_response_rate	**R-squared:**	0.117
Model:	OLS	**Adj. R-squared:**	0.107
Method:	Least Squares	**F-statistic:**	11.70
Date:	Sun, 29 Jan 2017	**Prob (F-statistic):**	2.48e-12
Time:	23:27:08	**Log-Likelihood:**	-835.29
No. Observations:	537	**AIC:**	1685.
Df Residuals:	530	**BIC:**	1715.
Df Model:	6		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[95.0% Conf. Int.]**
Intercept	-2.1320	0.354	-6.021	0.000	-2.828 -1.436
C(health_insurance)[T.1.0]	0.6100	0.295	2.068	0.039	0.030 1.190
C(ethnicity)[T.2.0]	-0.0958	0.152	-0.631	0.528	-0.394 0.202
C(BiologicalSex)[T.Male]	-0.2541	0.104	-2.449	0.015	-0.458 -0.050
I(study_entry_month)	-0.4409	0.076	-5.802	0.000	-0.590 -0.292
I(CurrentAge)	0.0230	0.004	5.702	0.000	0.015 0.031
I(education_stdscale)	-0.0539	0.053	-1.020	0.308	-0.158 0.050

Omnibus:	18.452	**Durbin-Watson:**	1.993
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	9.191
Skew:	-0.078	**Prob(JB):**	0.0101
Kurtosis:	2.378	**Cond. No.**	345.

In [31]:
```
cohort_response_rate[cohort_response_rate.index.isin(cox_df.index)]['ratio'].
hist(bins=20, color='#C30E6F', figsize=(10,8))
plt.ylabel('Count')
plt.xlabel('Response Rate')
```

Out[31]:
```
<matplotlib.text.Text at 0x1209d1b10>
```


Count
Response Rate
0
10
20
30
40
50
60
70
80
0.0
0.2
0.4
0.6
0.8
1.0

Univariate Model

Supplementary Table 4a

Estimated Cox proportional hazards models for predicting retention time based on univariate analysis.

'Table of Contents'

```
In [64]: model_summary_univariate = []
         for c in cox_cols:
             print c
             if c in ['last_survey_censored', 'event_observed']: continue
             cf.fit( cox_df[['last_survey_censored', 'event_observed', c]], \
                        'last_survey_censored', event_col='event_observed',\
                       )

             model_summary = cf.summary
             model_summary.sort(columns=['p'], inplace=True)
             model_summary.index = [i.replace('_', ' ').upper().replace('BIOLOGICALSEX
         ', 'BIOLOGICAL SEX').lower() for i in model_summary.index]
             model_summary['lower 0.95'] = model_summary['lower 0.95'].map(np.exp)
             model_summary['upper 0.95'] = model_summary['upper 0.95'].map(np.exp)

             model_summary_univariate.append( model_summary)

         pd.set_option('display.precision',3)
         univariate_models = pd.concat(model_summary_univariate)
         univariate_models['abs_z'] = univariate_models['z'].map(np.abs)
         univariate_models.sort(columns=['abs_z'], ascending=False, inplace=True)
         del univariate_models['abs_z']
         univariate_models = univariate_models.rename(index={'currentage':'current age
         '})
         univariate_models
```

```
gina
last_survey_censored
event_observed
education
age_when_diagnosed
Income
health_insurance
```

```
/Users/ers/anaconda2/lib/python2.7/site-packages/ipykernel/__main__.py:8: Fut
ureWarning: sort(columns=....) is deprecated, use sort_values(by=.....)
```

```
ethnicity
CurrentAge
BiologicalSex
study_entry_month
```

```
/Users/ers/anaconda2/lib/python2.7/site-packages/ipykernel/__main__.py:18: Fu
tureWarning: sort(columns=....) is deprecated, use sort_values(by=.....)
```

Out[64]:

	coef	exp(coef)	se(coef)	z	p	lower 0.95	upper 0.95
study entry month	0.637	1.891	0.075	8.523	1.559e-17	1.633	2.189
current age	-0.020	0.980	0.004	-4.907	9.230e-07	0.972	0.988
age when diagnosed	-0.009	0.991	0.004	-2.423	1.541e-02	0.984	0.998
ethnicity	-0.293	0.746	0.144	-2.029	4.243e-02	0.562	0.990
income	-0.089	0.915	0.045	-1.995	4.606e-02	0.838	0.998
health insurance	-0.462	0.630	0.272	-1.694	9.025e-02	0.369	1.075
education	-0.131	0.877	0.083	-1.586	1.128e-01	0.746	1.031
gina	0.083	1.086	0.072	1.153	2.488e-01	0.944	1.250
biological sex	-0.017	0.983	0.103	-0.164	8.694e-01	0.803	1.204

Multivariate Cox PH Model

Supplemental Table 4b

'Table of Contents'

In [51]:
```
#Multivariate Cox model
# outcome measurement: number of days until last daily survey completion

pd.set_option('display.precision',4)
m_summary.index = [i.replace('_', ' ').replace('CurrentAge', 'current age').r
eplace('BiologicalSex', 'biological sex').lower() for i in m_summary.index]
m_summary['lower 0.95'] = m_summary['lower 0.95'].apply(np.exp)
m_summary['upper 0.95'] = m_summary['upper 0.95'].apply(np.exp)

m_summary
```

Out[51]:

	coef	exp(coef)	se(coef)	z	p	lower 0.95	upper 0.95
study entry month	0.6976	2.0089	0.0762	9.1562	5.3751e-20	281.9345	29840.2139
current age	-0.0226	0.9776	0.0046	-4.9031	9.4336e-07	13.9411	14.6123
health insurance	-0.4605	0.6310	0.2816	-1.6355	1.0194e-01	4.2126	19.9167
ethnicity	-0.2060	0.8138	0.1464	-1.4075	1.5928e-01	6.3092	19.2461
biological sex	0.0996	1.1048	0.1077	0.9250	3.5497e-01	11.5421	50.1001
age when diagnosed	-0.0037	0.9963	0.0041	-0.8985	3.6890e-01	14.6815	15.3366
gina	0.0609	1.0628	0.0733	0.8305	4.0623e-01	12.3108	30.3105
education	0.0471	1.0483	0.0889	0.5301	5.9603e-01	11.1591	32.5612
income	-0.0139	0.9862	0.0489	-0.2849	7.7571e-01	11.5836	19.3139

Ordinary Least Squares Model

Supplemental Table 4c

'Table of Contents'

```
In [54]: results.summary()
```

Out[54]:

OLS Regression Results

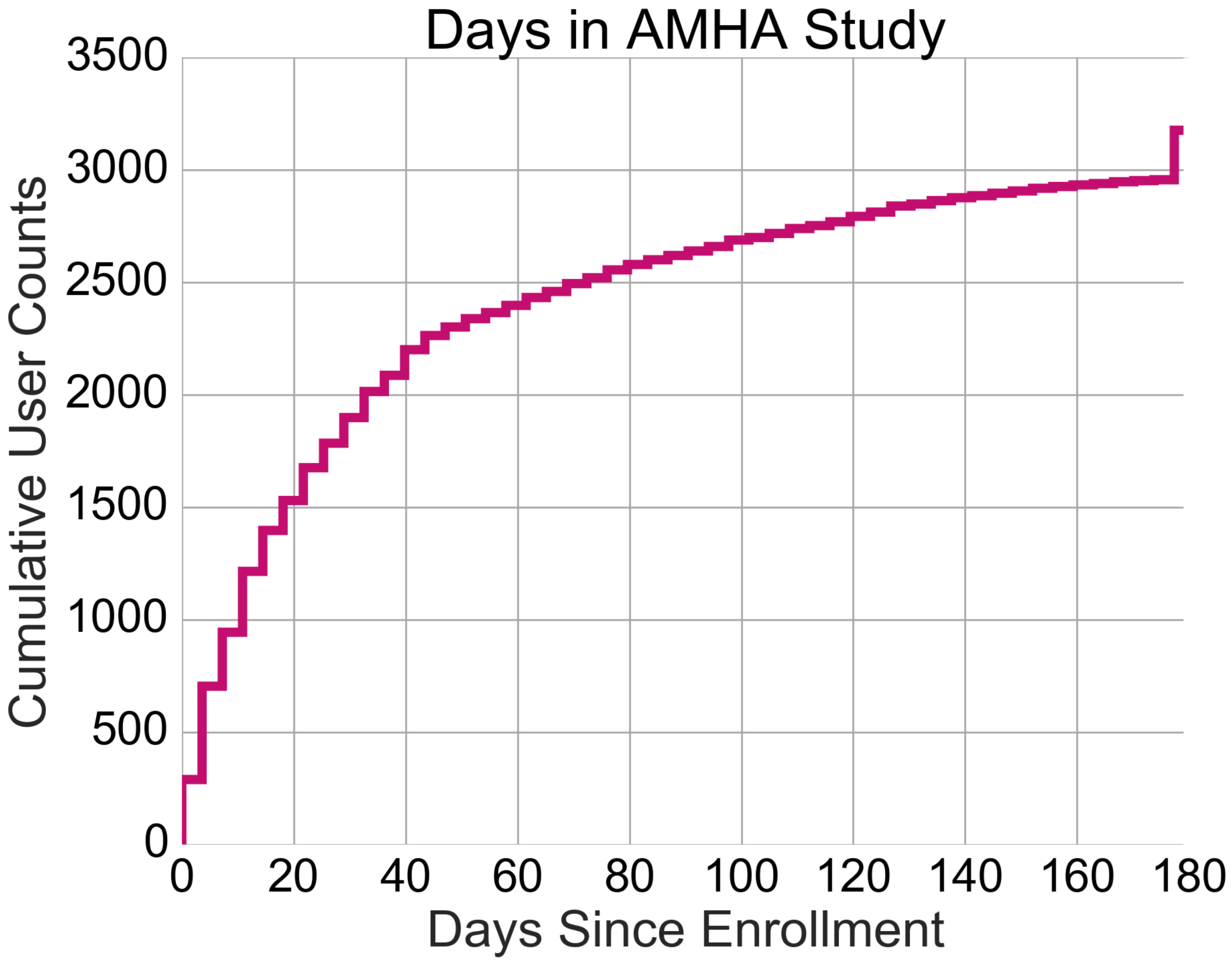
Dep. Variable:	logit_response_rate	**R-squared:**	0.117
Model:	OLS	**Adj. R-squared:**	0.107
Method:	Least Squares	**F-statistic:**	11.70
Date:	Fri, 27 Jan 2017	**Prob (F-statistic):**	2.48e-12
Time:	14:37:23	**Log-Likelihood:**	-835.29
No. Observations:	537	**AIC:**	1685.
Df Residuals:	530	**BIC:**	1715.
Df Model:	6		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[95.0% Conf. Int.]**
Intercept	-2.1239	0.352	-6.026	0.000	-2.816 -1.431
C(health_insurance)[T.1.0]	0.6100	0.295	2.068	0.039	0.030 1.190
C(ethnicity)[T.2.0]	-0.0958	0.152	-0.631	0.528	-0.394 0.202
C(BiologicalSex)[T.Male]	-0.2541	0.104	-2.449	0.015	-0.458 -0.050
I(study_entry_month)	-0.4409	0.076	-5.802	0.000	-0.590 -0.292
I(CurrentAge)	0.0230	0.004	5.702	0.000	0.015 0.031
I(education_stdscale)	-0.0564	0.055	-1.020	0.308	-0.165 0.052

Omnibus:	18.452	**Durbin-Watson:**	1.993
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	9.191
Skew:	-0.078	**Prob(JB):**	0.0101
Kurtosis:	2.378	**Cond. No.**	344.

CDF of Days in AMHA Study

In [55]:
```
last_study_day = daily_weekly_df[daily_survey_standardqs].dropna(how='all').r
eset_index().groupby('healthCode')['study_day'].max()
last_study_day = last_study_day.map(lambda x: 181 if x > 179 else x)
last_study_day.name = 'last_survey_censored'

last_study_day.ix[baseline.healthCode].hist(bins=50,  histtype='step', cumula
tive=True,  linewidth=5, color='#C30E6F', figsize=(10,8))
#plt.xlabel('Enrollment Day')
plt.ylabel('Cumulative User Counts', rotation=90)
plt.xlabel('Days Since Enrollment')
plt.grid(color='darkgray')
plt.title('Days in AMHA Study')
plt.xlim(0,180)
plt.vlines(x=180,ymin=0,ymax=3500, color='white', linewidth=6)
plt.tight_layout()
#plt.savefig('/Users/ers/Dropbox/feasibility_figures_ERS/CumulativeHist_Reten
tion.svg',dpi=600, format='svg')
plt.show()
```


Days in AMHA Study
Cumulative User Counts
0
500
1000
1500
2000
2500
3000
3500
0
20
40
60
80
100
120
140
160
180
Days Since Enrollment

In [56]:
```
cox_df.last_survey_censored.hist(bins=50, histtype='step', linewidth=5, color
='#C30E6F', figsize=(16,5))
plt.ylabel('Participant Count', rotation=90)
plt.grid(color='darkgray')
plt.title('Days in AMHA Study')
plt.tight_layout()
```

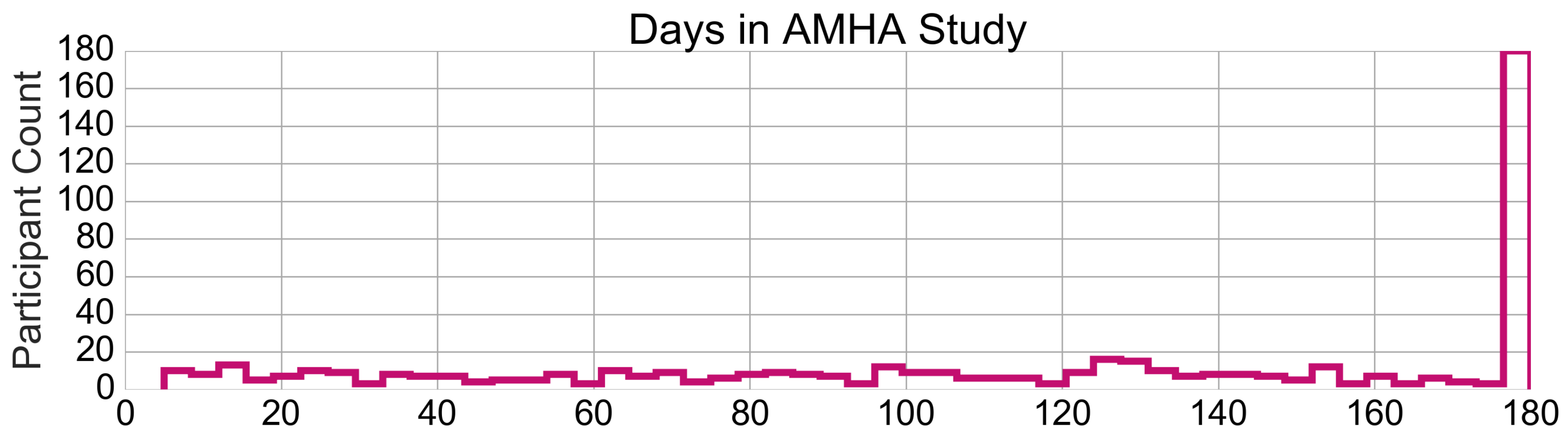
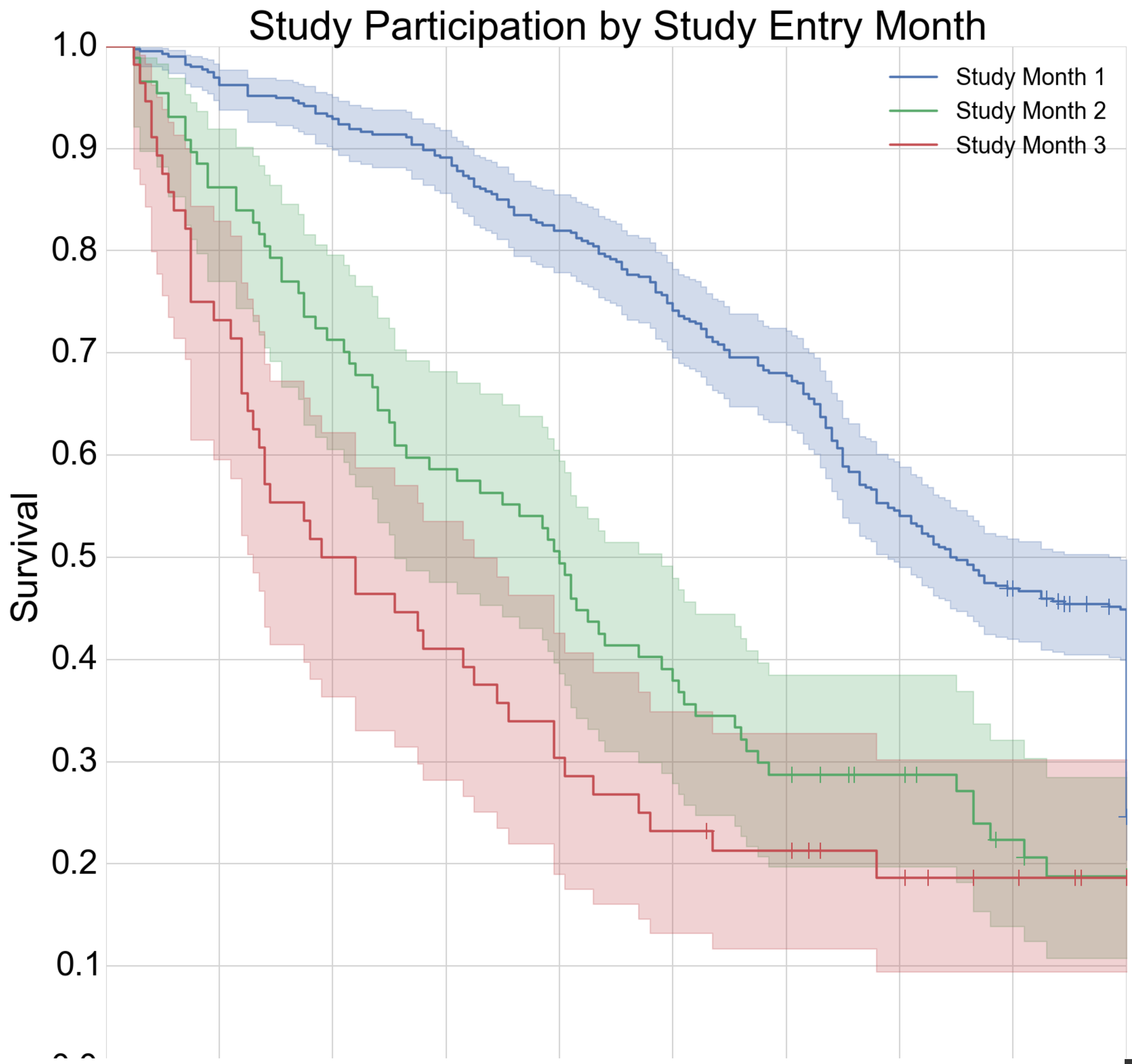


Figure 2d - Enrollment and retention over time for Robust users By Study Entry Month

['Table of Contents']

['cox_df used below created here']

In [34]:
```
#Baseline
#Kaplan Meier Curve
from lifelines import KaplanMeierFitter
# kmf_control = KaplanMeierFitter()
# kmf_control.fit(last_daily.last_daily.values, label='Asthma App').plot()

#Baseline
from lifelines import KaplanMeierFitter
from lifelines.plotting import add_at_risk_counts

rcParams['text.color'] = 'black'
rcParams['ytick.color'] = 'black'
rcParams['xtick.color'] = 'black'
rcParams['axes.facecolor'] = 'white'
rcParams['axes.edgecolor'] = 'white'

fig, ax = plt.subplots(1,1, figsize=(12,12))
fig.patch.set_facecolor('white')


kmfs = []
row_counter = 0
col_counter = 0
for i,n in cox_df[cox_df.index.isin(robust.healthCode)].groupby(['study_entry
_month']):



    kmf_control = KaplanMeierFitter()
    kmf_df = n
    label_dict = {0:'Study Month 1', 1:'Study Month 2', 2:'Study Month 3', 3:
'Study Month 4'}
    label_i = label_dict[i]
    kmf_control.fit(kmf_df.last_survey_censored.values, event_observed=kmf_df
.event_observed, label=label_i)


    kmf_control.plot(label=i, ax=ax, cmap='jet', show_censors=True)
    kmfs.append(kmf_df)
    ax.set_axis_bgcolor('white')
    ax.set_title('Study Participation by Study Entry Month' )


    ax.set_xlabel('Enrollment Day', color='black')

    ax.set_ylabel('Survival', color='black')

plt.legend(fontsize='xx-large')
plt.tight_layout()
plt.yticks(arange(0,1.1,.1))
plt.savefig('../analysis/feasibility_figures_ERS/KM_by_study.svg', format='sv
g', dpi=600)
plt.show()
```



Study Participation by Study Entry Month
Survival
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
Study Month 1
Study Month 2
Study Month 3

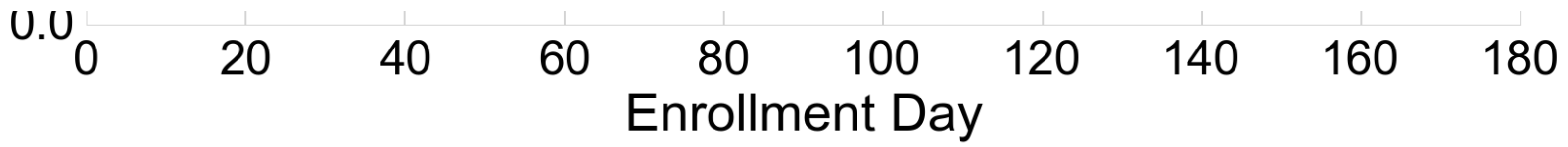
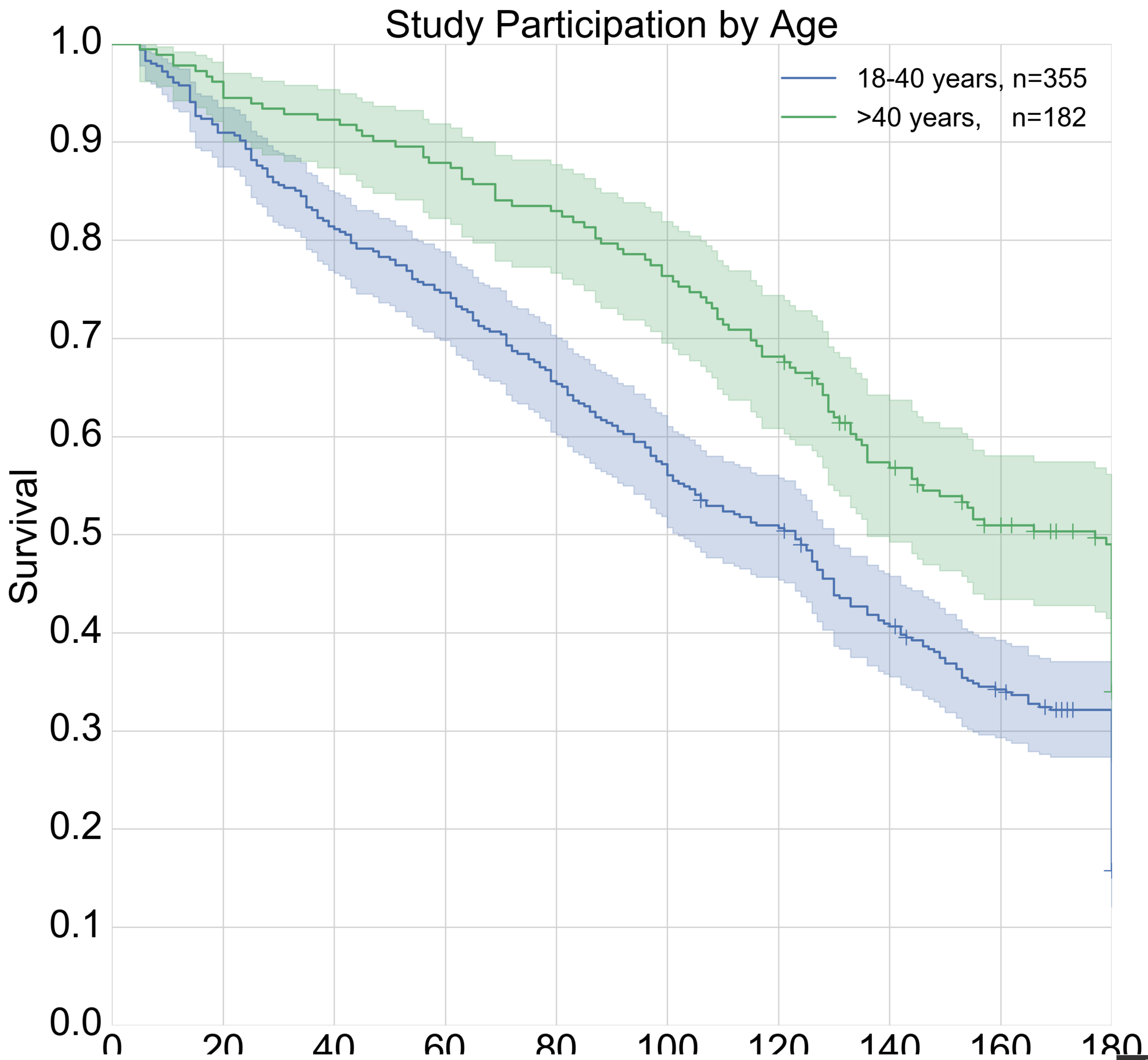


Figure 2e Enrollment and retention over time for Robust users By Age

'Table of Contents'

In [64]:
```
#Baseline
#Kaplan Meier Curve
from lifelines import KaplanMeierFitter

#Baseline
from lifelines import KaplanMeierFitter
from lifelines.plotting import add_at_risk_counts


rcParams['text.color'] = 'black'
rcParams['ytick.color'] = 'black'
rcParams['xtick.color'] = 'black'
rcParams['axes.facecolor'] = 'white'
rcParams['axes.edgecolor'] = 'white'

fig, ax = plt.subplots(1,1, figsize=(12,12))
fig.patch.set_facecolor('white')


kmfs = []
row_counter = 0
col_counter = 0


cox_df_age = cox_df.copy()

cox_df_age['Current Age binned'] = pd.cut(cox_df['CurrentAge'], [0,40,100], l
abels=False, retbins=False)
for i,n in cox_df_age[cox_df_age.index.isin(baseline.healthCode)].groupby(['C
urrent Age binned']):

    print i
    if n.shape[0] < 10: continue
    kmf_control = KaplanMeierFitter()
    kmf_df = n

    if i ==0:
        label_i = '18-40 years, n=' + str(n.shape[0])
    else:
        label_i = '>40 years,    n=' + str(n.shape[0])
    kmf_control.fit(kmf_df.last_survey_censored.values, event_observed=kmf_df
.event_observed, label=label_i)



    kmf_control.plot(label=i, ax=ax, cmap='jet', show_censors=True, ci_show=T
rue)
    kmfs.append(kmf_df)
    ax.set_axis_bgcolor('white')
    ax.set_title('Study Participation by Age', fontsize=30)
    ax.set_ylabel('Survival', fontsize=30)
    ax.set_xlabel('Enrollment Day', fontsize=30)



    ax.set_xlabel('Enrollment Day', color='black')

    ax.set_ylabel('Survival', color='black')
plt.legend(fontsize='xx-large')
plt.tight_layout()
plt.yticks(arange(0,1.1,.1), fontsize=30)
plt.xticks(range(0,200, 20), fontsize=30)
plt.show()
```

```
0
1
```



Study Participation by Age
18-40 years, n=355
>40 years, n=182
Survival
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0
20
40
60
80
100
120
140
160
180

Heatmaps

Supplemental Figure 3

'Table of Contents'

Symptom Heatmaps

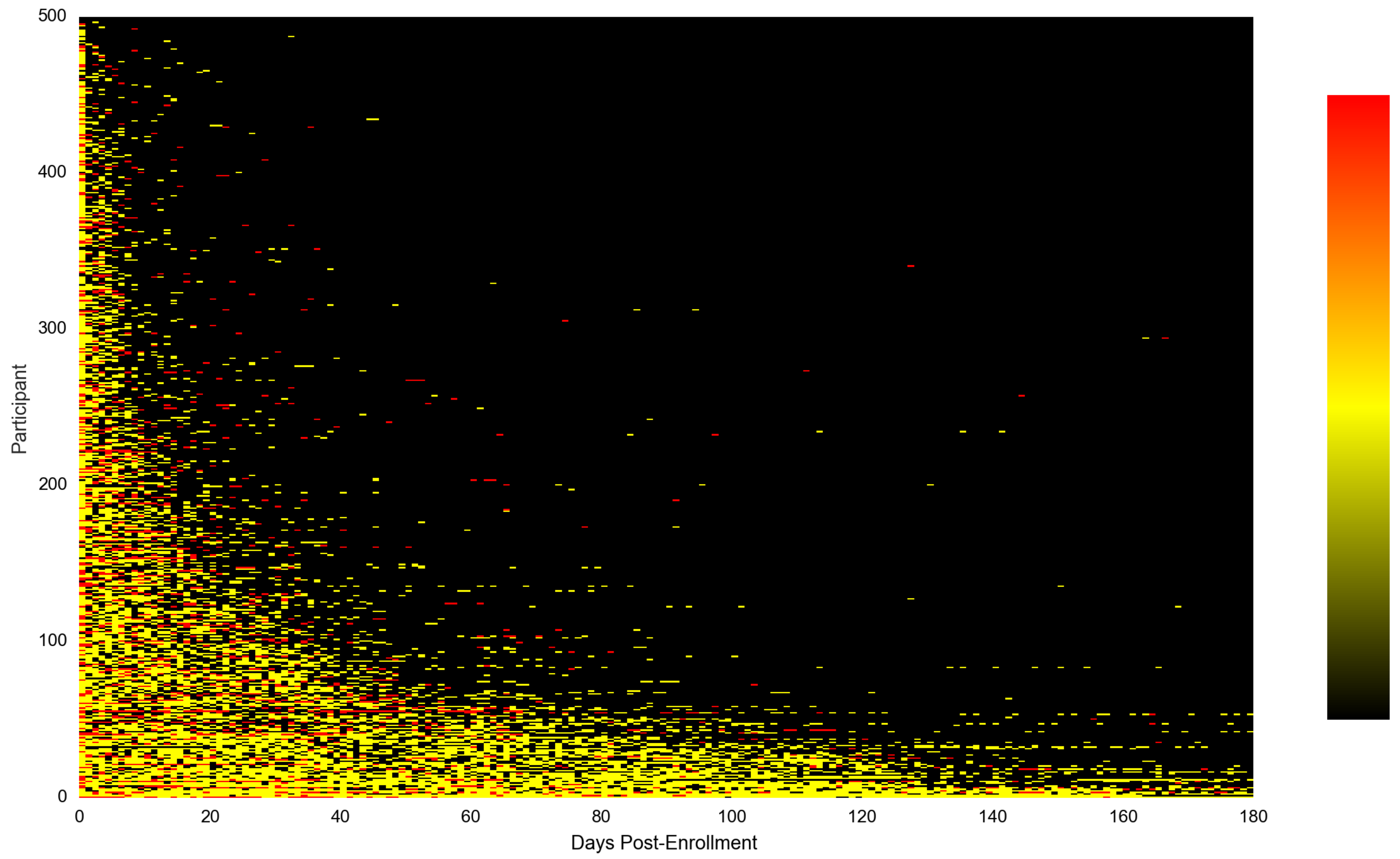
day_symptoms, night_symptoms, use_qr

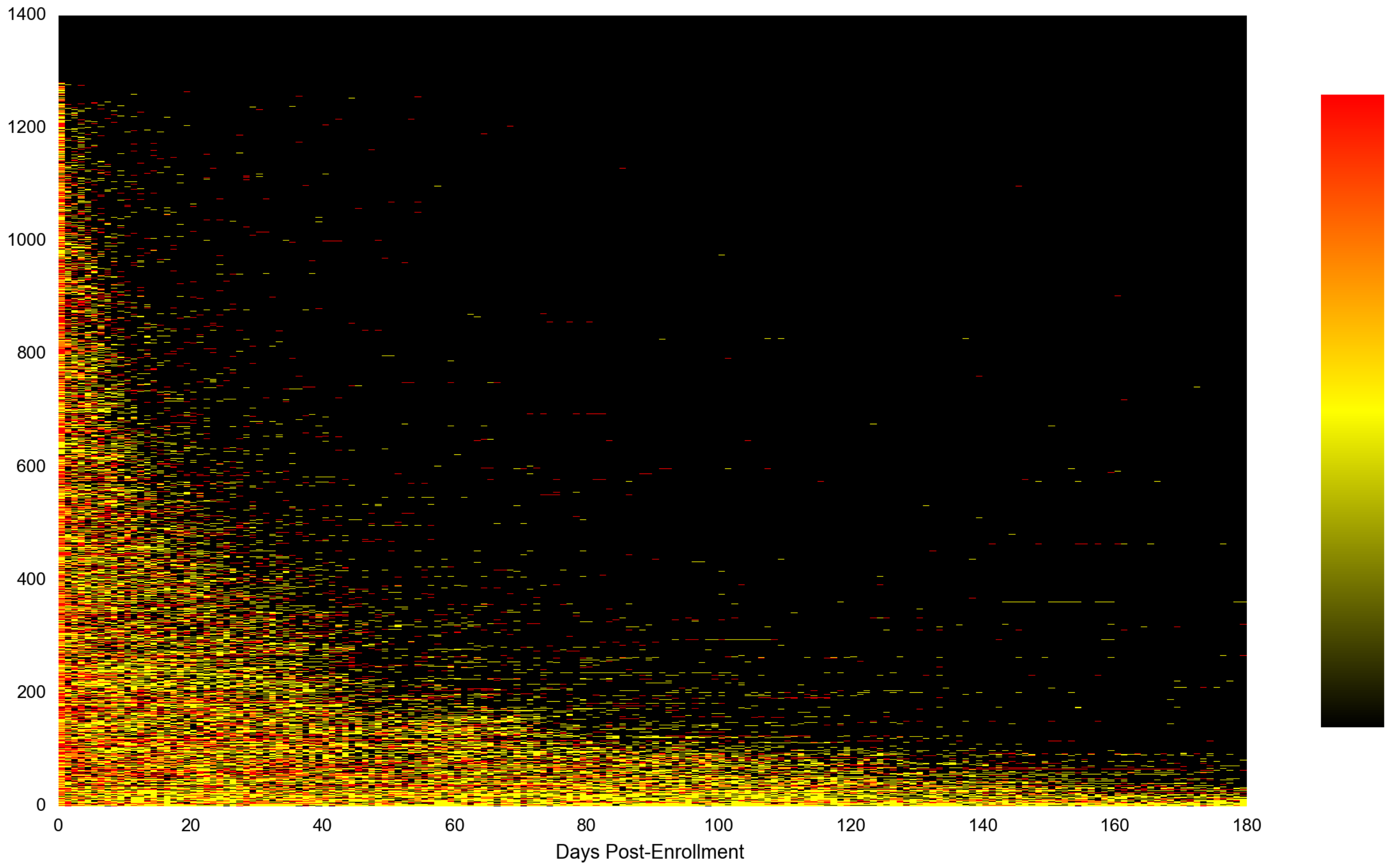
In [65]:
```
rcParams['text.color'] = 'black'
rcParams['ytick.color'] = 'black'
rcParams['xtick.color'] = 'black'



from matplotlib.colors import LinearSegmentedColormap
vmax = 2.0
cmap = LinearSegmentedColormap.from_list('mycmap', [(0 / vmax, 'black'),
                                                    (1 / vmax, 'yellow'),
                                                    (2 / vmax, 'red')]
                                         )



def heatmap_plot(heatmap_df, title, ylabel=True):
    fig, ax = plt.subplots(1,1, figsize=(15,8))

    heatmap_df = heatmap_df[range(0,180)]

    #Order participants by daily survey completion counts
    heatmap_df_order = heatmap_df.apply(lambda x: sum([1 if i in set([0,1]) e
lse 0 for i in x]), axis=1)
    heatmap_df_order.sort(inplace=True, ascending=False)
    heatmap_df_order


    p = ax.pcolormesh(heatmap_df.ix[heatmap_df_order.index].values, cmap=cmap
)


    cbar = fig.colorbar(p, aspect=10,  shrink=0.8, ticks=[-1,0,1])
    cbar.set_ticklabels([''])
    ax.set_xlabel('Days Post-Enrollment ', color='black')
    ax.set_axis_bgcolor('black')


    if ylabel:
        ax.set_ylabel('Participant')


    plt.show()


ids_180days_of_study = days_till_2015_09_09[days_till_2015_09_09>179].index

for n,symptom in enumerate(['day_symptoms', 'night_symptoms', 'use_qr']):

    symptom_title = symptom.replace('_', ' ').upper()
    print symptom_title

    if symptom != 'medicine':
        heatmap_df = daily_weekly_df[symptom].map(sum).reset_index().pivot_ta
ble(index='healthCode', columns='study_day', values=symptom)

    else:
        heatmap_df = daily_weekly_df[[symptom]]
        heatmap_df[symptom] = heatmap_df[symptom].map(med_multi_parse)
        heatmap_df = heatmap_df.reset_index().pivot_table(index='healthCode',
columns='study_day', values=symptom)

    #LIMITS PLOTS TO PEOPLE IN STUDY FOR AT LEAST 180 DAYS BY 9/9/2015
    heatmap_df = heatmap_df[heatmap_df.index.isin(ids_180days_of_study)]

    wellcontrolled = heatmap_df[heatmap_df.index.isin(baseline[baseline.gina=
='WellControlled']['healthCode'].values)]
```

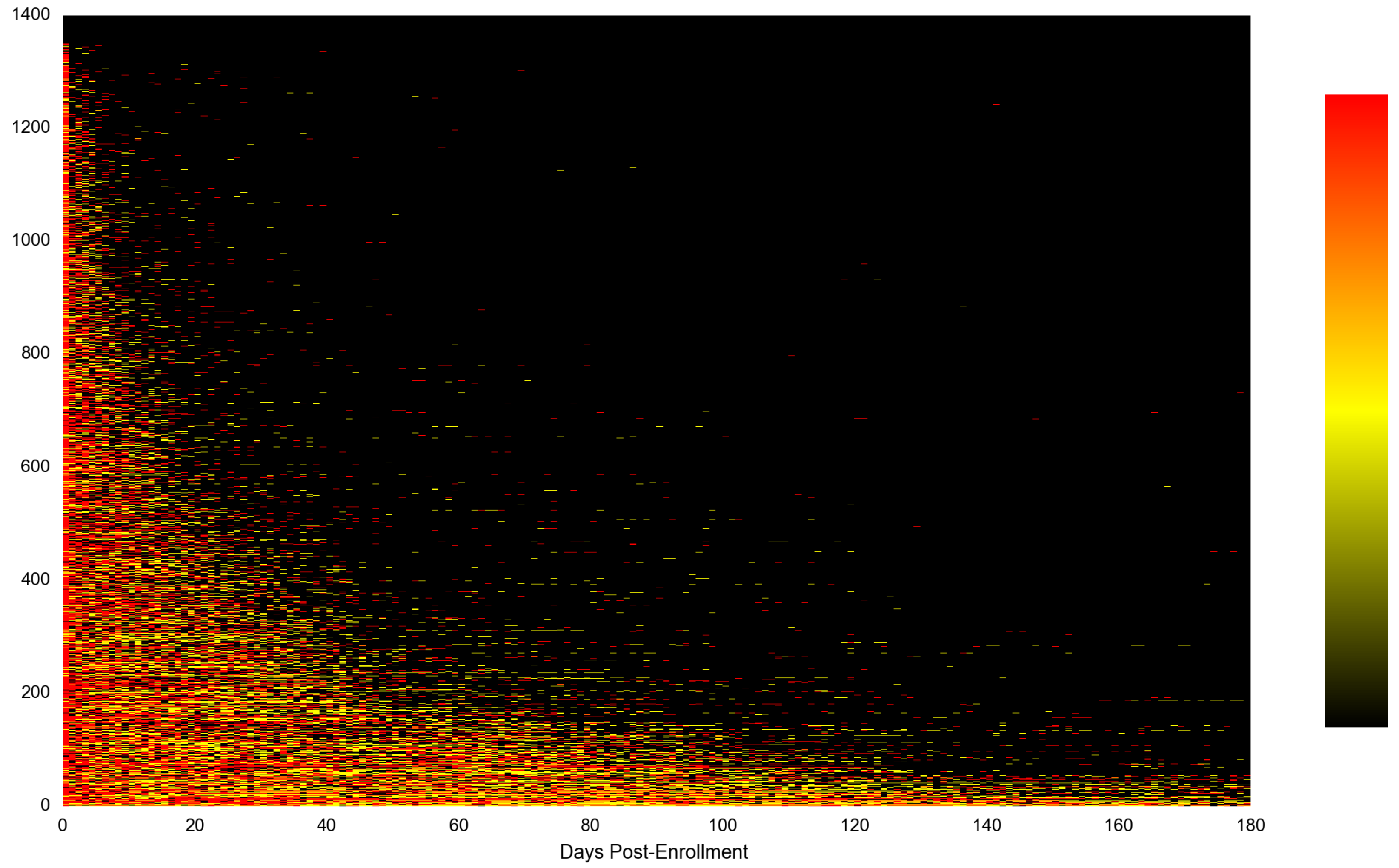
```
    heatmap_plot(wellcontrolled, 'Well Controlled,  n='+ str(wellcontrolled.s
hape[0]), True)

    partlycontrolled = heatmap_df[heatmap_df.index.isin(baseline[baseline.gin
a=='PartlyControlled']['healthCode'].values)]
    heatmap_plot(partlycontrolled, 'Partly Controlled,  n=' + str(partlycontr
olled.shape[0]), False)

    uncontrolled = heatmap_df[heatmap_df.index.isin(baseline[baseline.gina=='
Uncontrolled']['healthCode'].values)]
    heatmap_plot(uncontrolled, 'Uncontrolled,  n=' + str(uncontrolled.shape[0
]), False)
```

```
DAY SYMPTOMS
```

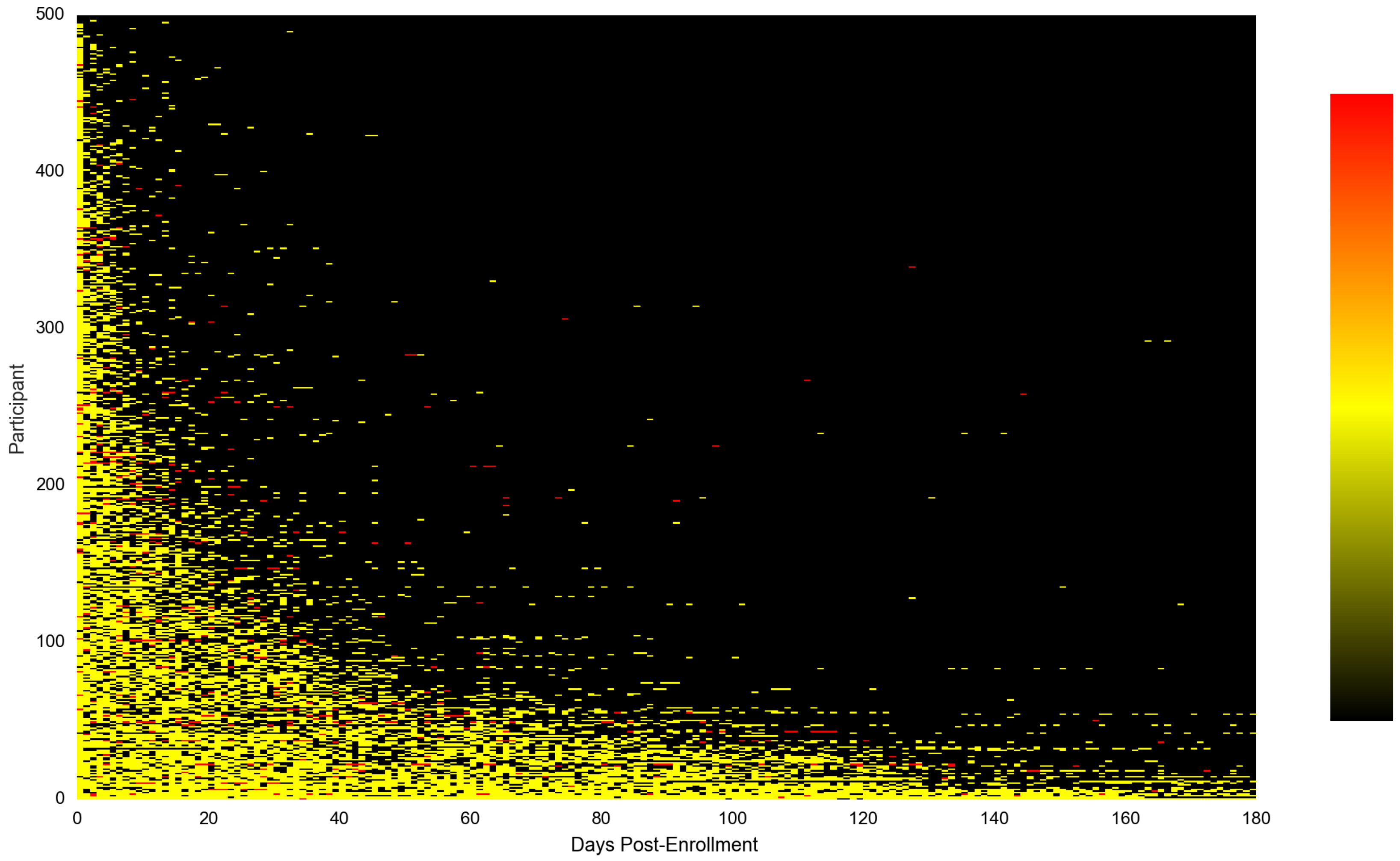
```
/Users/ers/anaconda2/lib/python2.7/site-packages/ipykernel/__main__.py:22: Fu
tureWarning: sort is deprecated, use sort_values(inplace=True) for INPLACE so
rting
```


500
400
300
200
100
0
Participant
0
20
40
60
80
100
120
140
160
180
Days Post-Enrollment

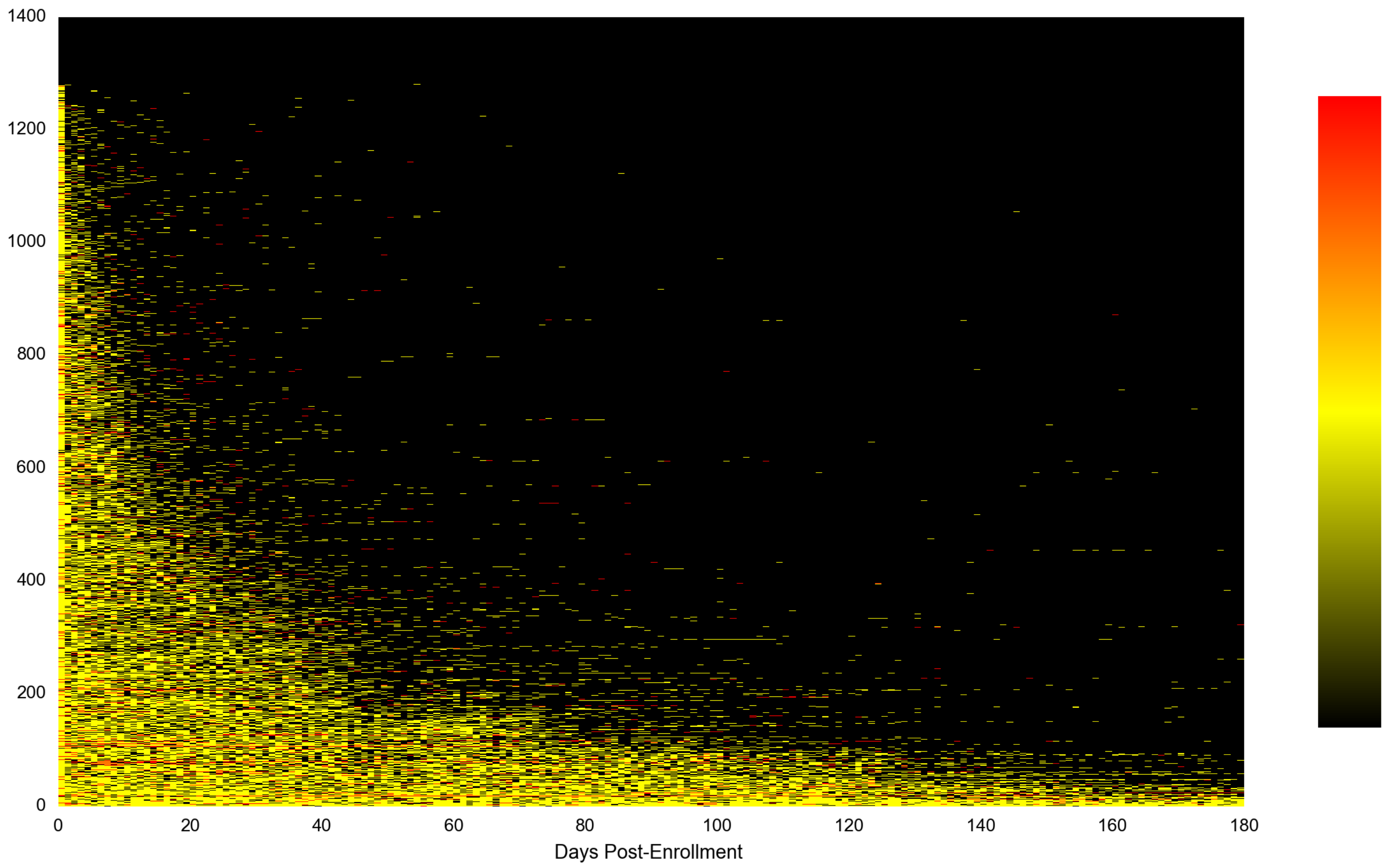

1400
1200
1000
800
600
400
200
0
0
20
40
60
80
100
120
140
160
180
Days Post-Enrollment



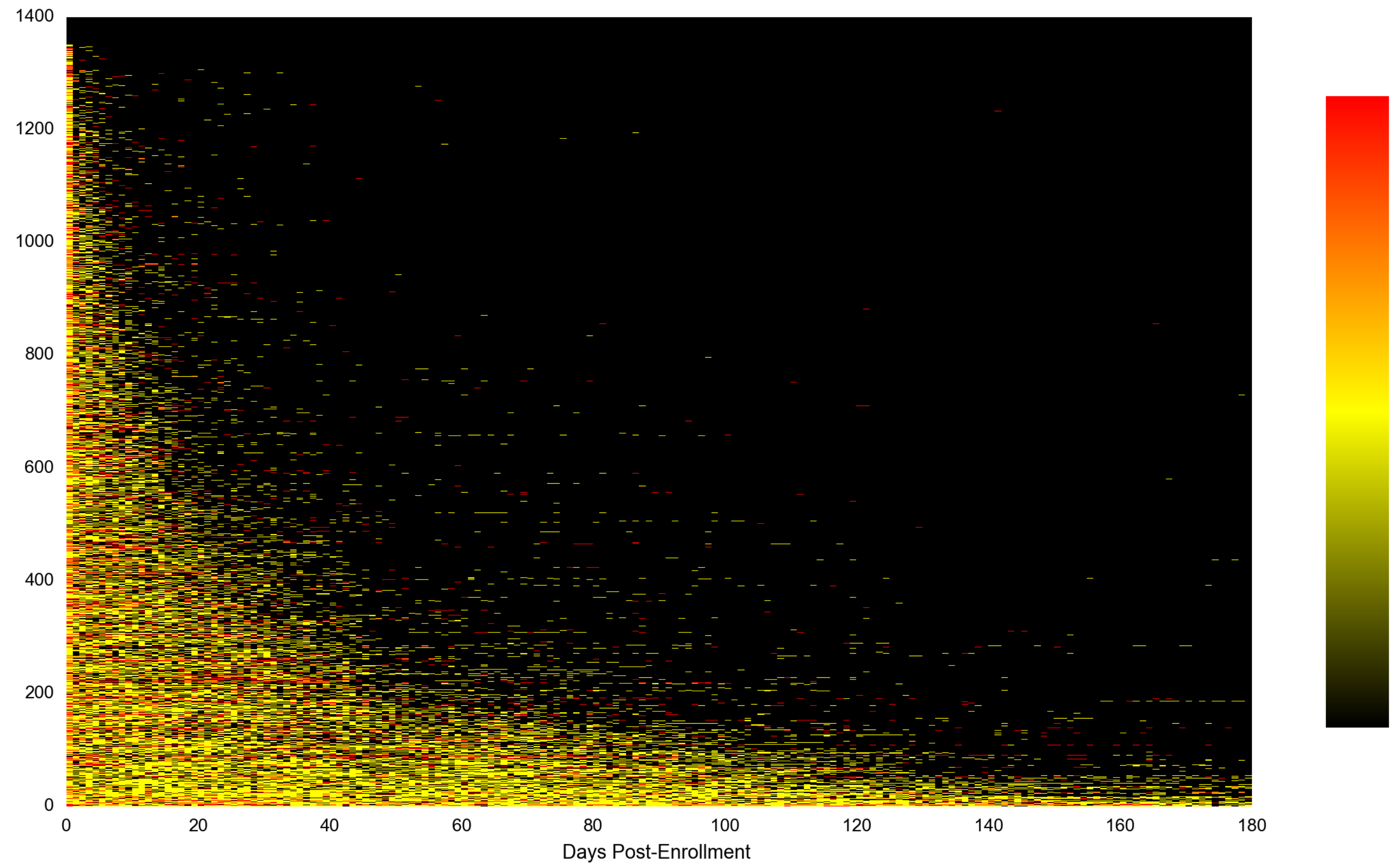
NIGHT SYMPTOMS



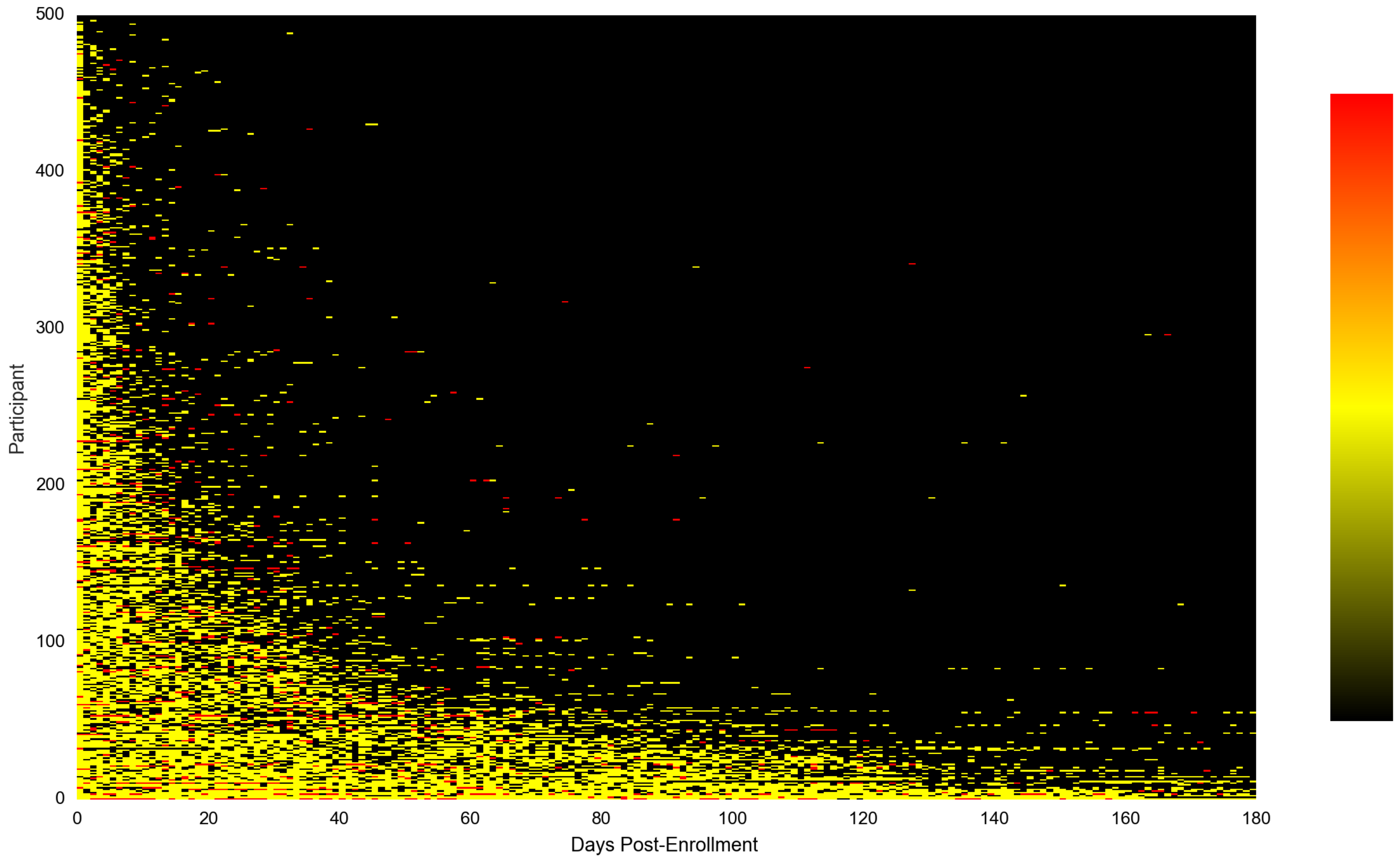
500
400
300
200
100
0
Participant
0
20
40
60
80
100
120
140
160
180
Days Post-Enrollment

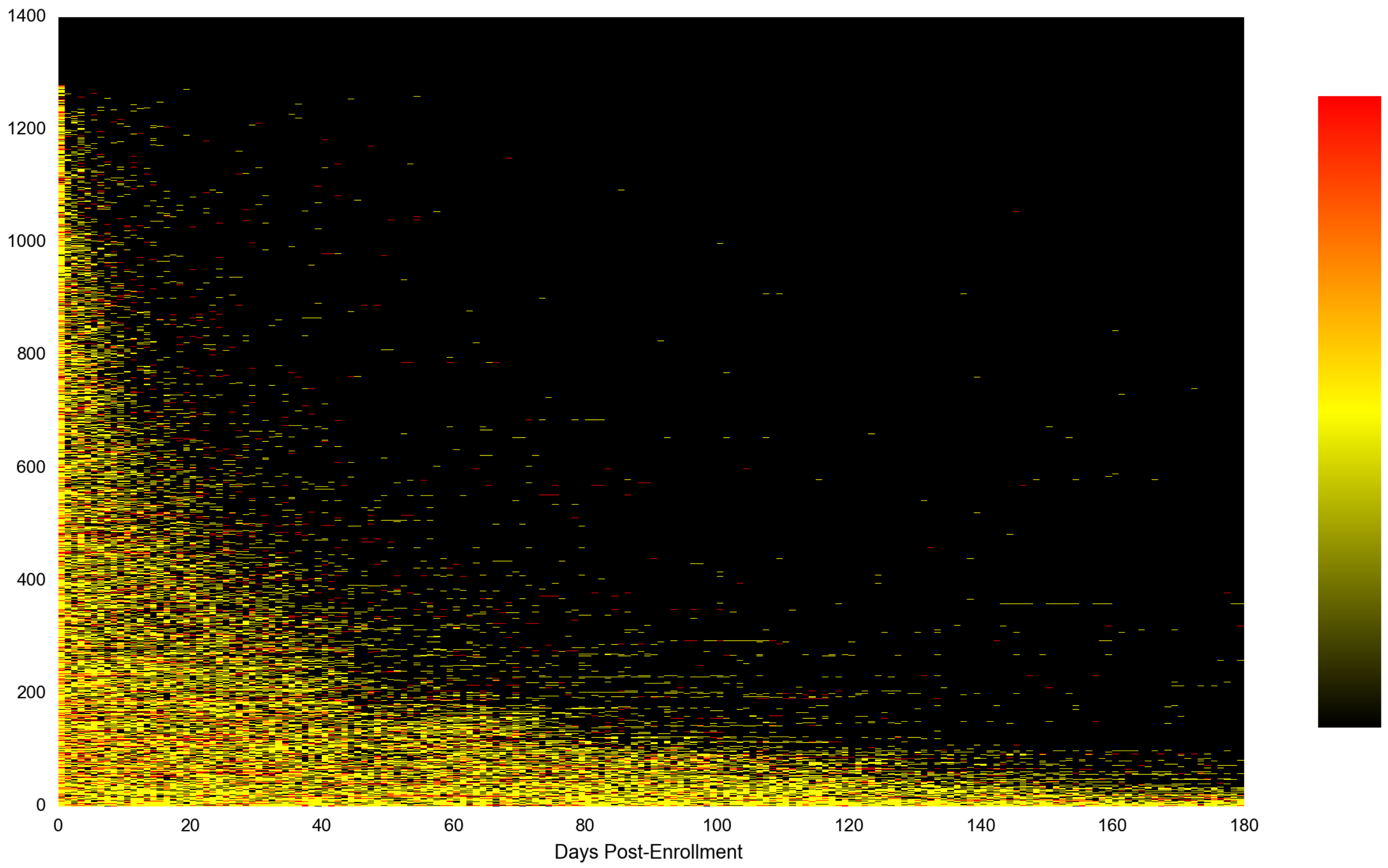


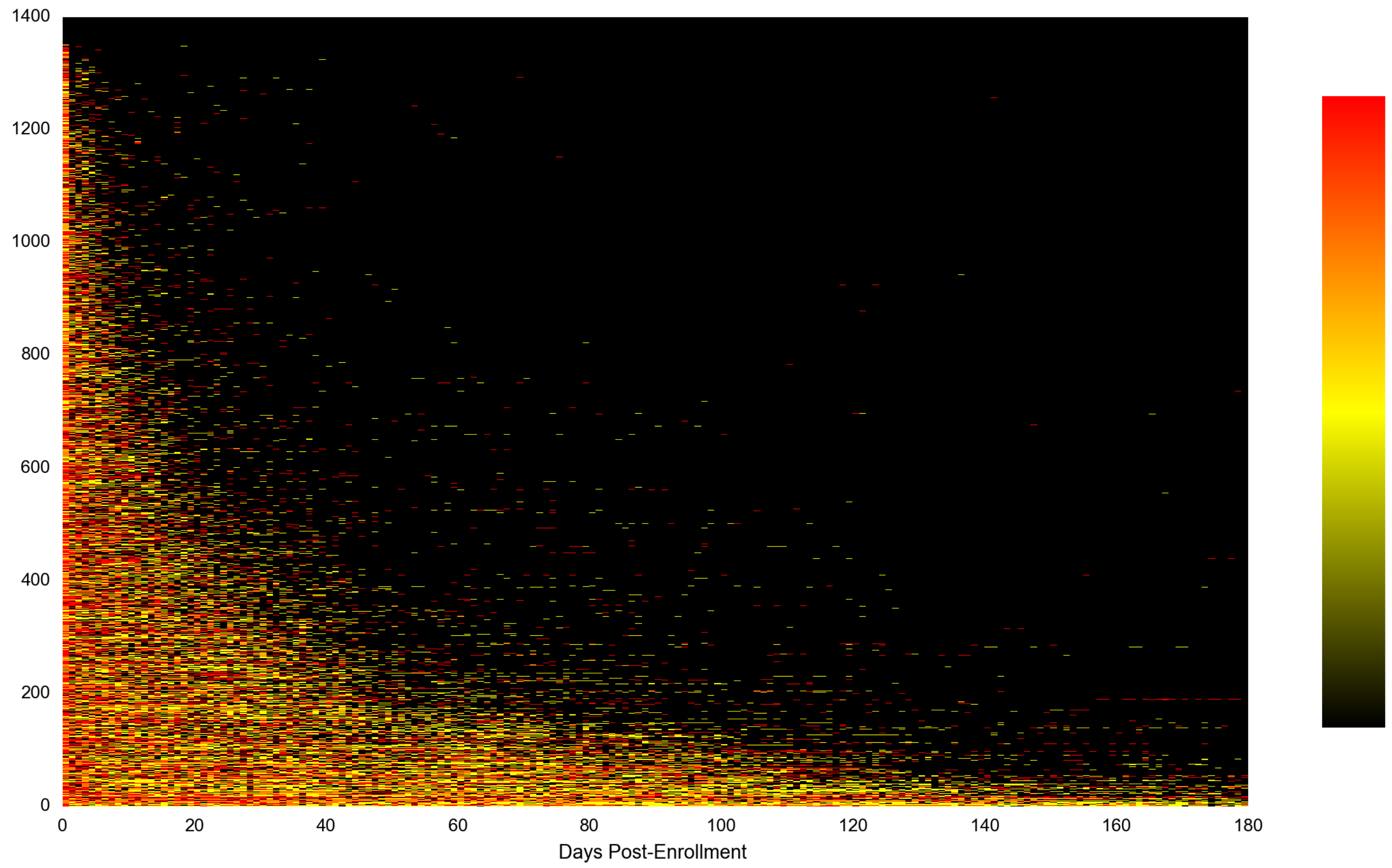
1400
1200
1000
800
600
400
200
0
0
20
40
60
80
100
120
140
160
180
Days Post-Enrollment


1400
1200
1000
800
600
400
200
0
0
20
40
60
80
100
120
140
160
180
Days Post-Enrollment

USE QR


500
400
300
200
100
0
Participant
0
20
40
60
80
100
120
140
160
180
Days Post-Enrollment


1400
1200
1000
800
600
400
200
0
0
20
40
60
80
100
120
140
160
180
Days Post-Enrollment



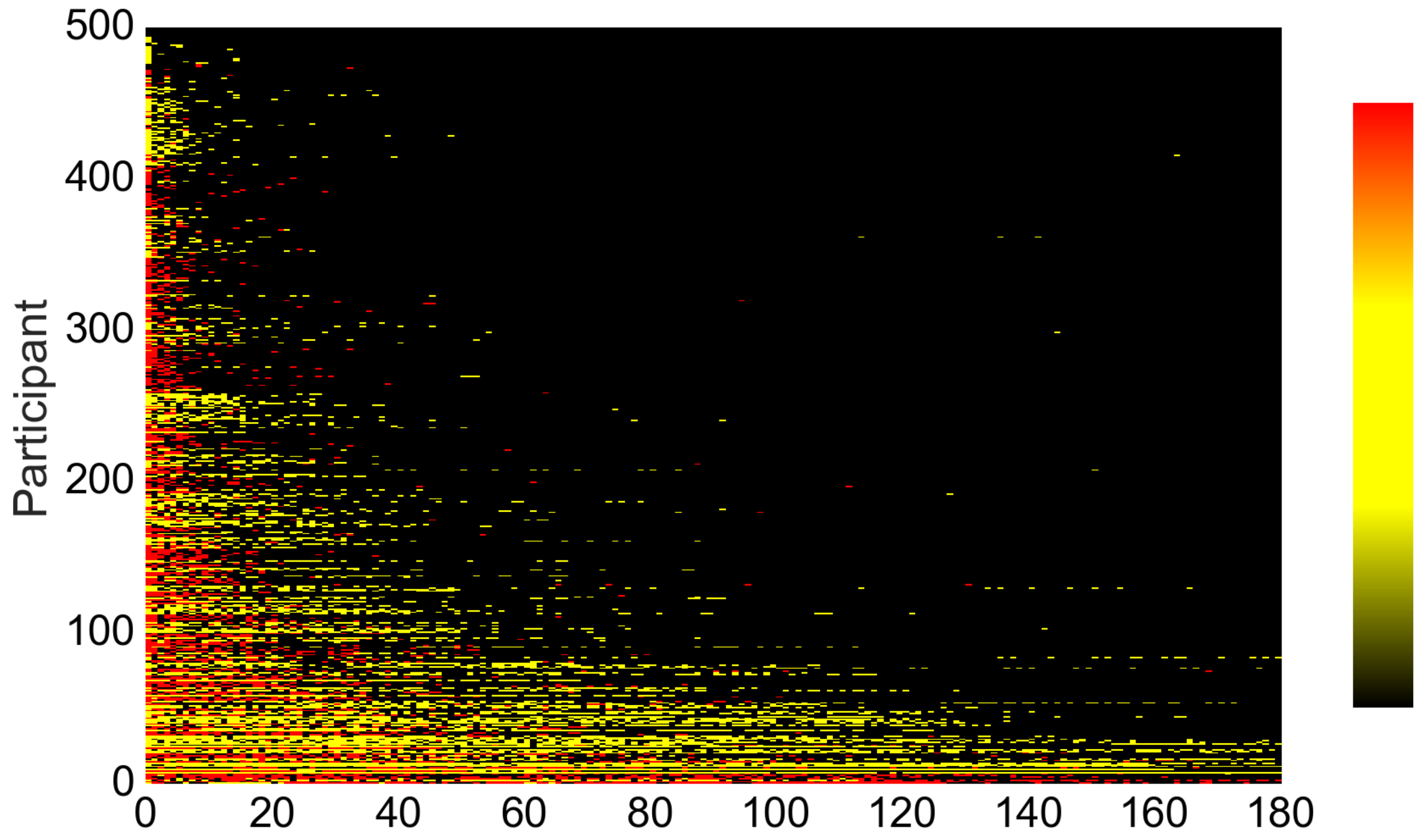
Daily Controller Medication Usage Heatmap

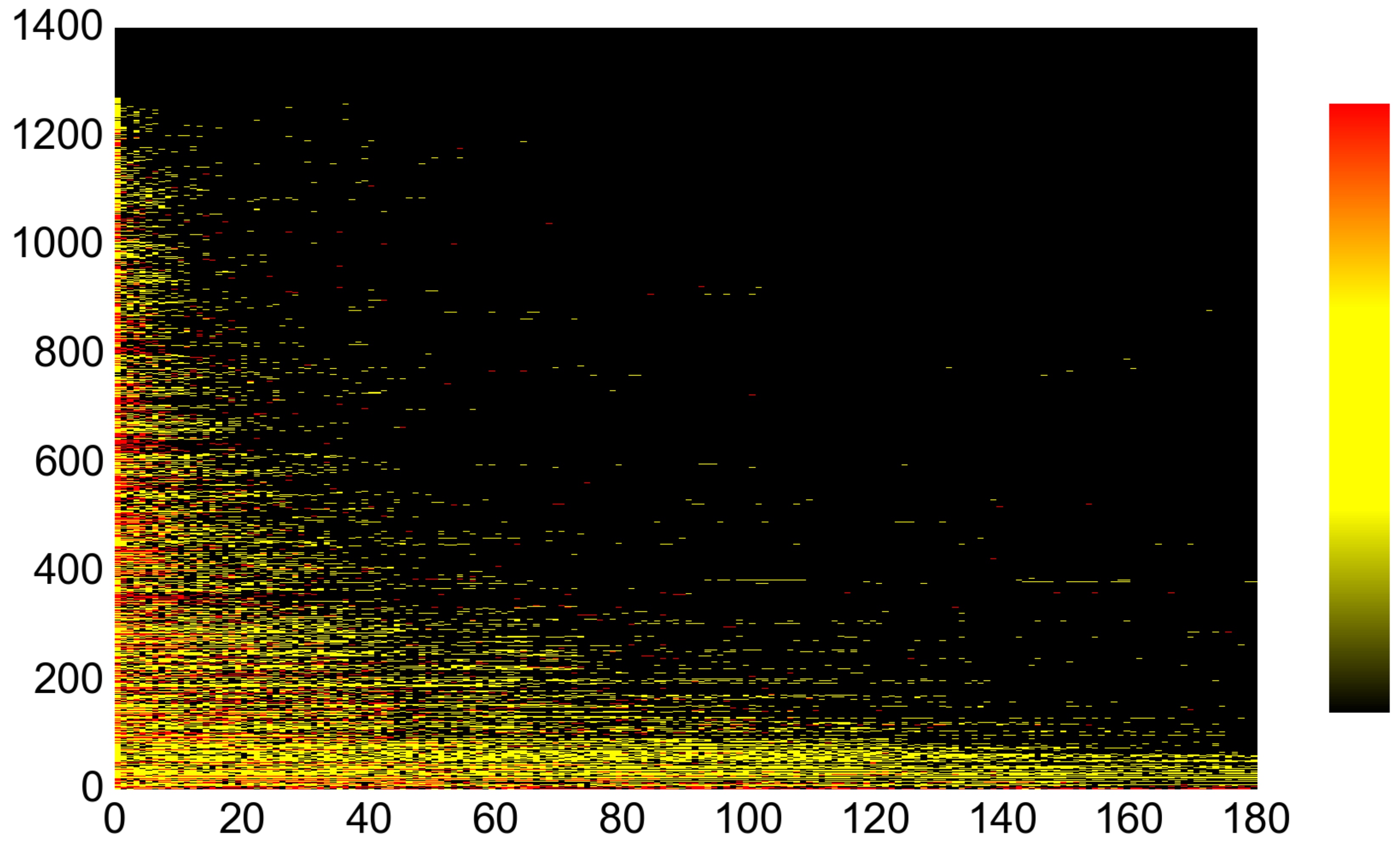
In [37]:
```
#DAILY CONTROLLER MEDICATION USAGE HEATMAP

rcParams['text.color'] = 'black'
rcParams['ytick.color'] = 'black'
rcParams['xtick.color'] = 'black'




from matplotlib.colors import LinearSegmentedColormap
vmax = 3.0
cmap = LinearSegmentedColormap.from_list('mycmap', [(0 / vmax, 'black'),
                                                    (1 / vmax, 'yellow'),
                                                    (2 / vmax, 'yellow'),
                                                    (3 / vmax, 'red')]

                                        )

def heatmap_plot(heatmap_df, title, ylabel=True):
    fig, ax = plt.subplots(1,1, figsize=(15,8))

    heatmap_df = heatmap_df[range(0,180)]

    #Order participants by daily survey completion counts

    heatmap_df_order = heatmap_df.apply(np.sum, axis=1)
    heatmap_df_order.sort(inplace=True, ascending=False)
    heatmap_df_order


    p = ax.pcolormesh(heatmap_df.ix[heatmap_df_order.index].values, cmap=cmap
)
```

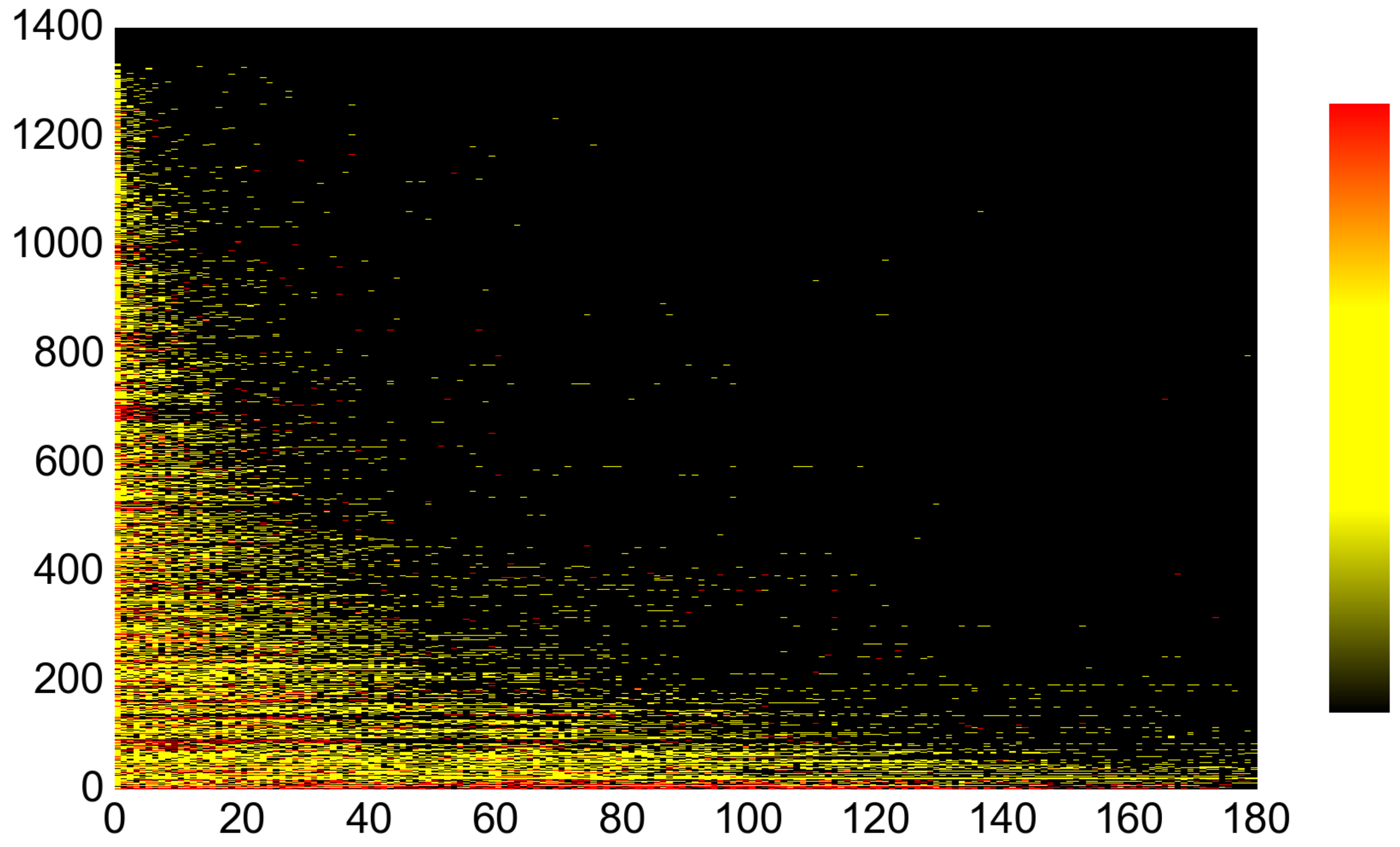
```
    cbar = fig.colorbar(p, aspect=10,  shrink=0.8, ticks=[0, 1, 2, 3])
    cbar.set_ticklabels([''])

    ax.set_axis_bgcolor('black')


    if ylabel:
        ax.set_ylabel('Participant')


    plt.show()


def med_multi_parse(x):
    if ',' in str(x):
        x = map(float, x.split(','))
        return min(x)
    return float(x)


ids_180days_of_study = days_till_2015_09_09[days_till_2015_09_09>179].index

for n,symptom in enumerate(['medicine']):

    symptom_title = symptom.replace('_', ' ').upper()
    print symptom_title

    if symptom != 'medicine':
        heatmap_df = daily_weekly_df[symptom].map(sum).reset_index().pivot_ta
ble(index='healthCode', columns='study_day', values=symptom)

    else:
        heatmap_df = daily_weekly_df[[symptom]]
        heatmap_df[symptom] = heatmap_df[symptom].map(med_multi_parse)
        heatmap_df = heatmap_df[heatmap_df[symptom]!=4]
        heatmap_df = heatmap_df.reset_index().pivot_table(index='healthCode',
columns='study_day', values=symptom)

    #LIMITS PLOTS TO PEOPLE IN STUDY FOR AT LEAST 180 DAYS BY 9/9/2015

    heatmap_df = heatmap_df.fillna(0)

    wellcontrolled = heatmap_df[heatmap_df.index.isin(baseline[baseline.gina=
='WellControlled']['healthCode'].values)]
    heatmap_plot(wellcontrolled, 'Well Controlled,  n='+ str(wellcontrolled.s
hape[0]), True)

    partlycontrolled = heatmap_df[heatmap_df.index.isin(baseline[baseline.gin
a=='PartlyControlled']['healthCode'].values)]
    heatmap_plot(partlycontrolled, 'Partly Controlled,  n=' + str(partlycontr
olled.shape[0]), False)

    uncontrolled = heatmap_df[heatmap_df.index.isin(baseline[baseline.gina=='
Uncontrolled']['healthCode'].values)]
    heatmap_plot(uncontrolled, 'Uncontrolled,  n=' + str(uncontrolled.shape[0
]), False)
```

```
/Users/ers/anaconda/lib/python2.7/site-packages/ipykernel/__main__.py:76: Set
tingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/st
able/indexing.html#indexing-view-versus-copy
/Users/ers/anaconda/lib/python2.7/site-packages/ipykernel/__main__.py:26: Fut
ureWarning: sort is deprecated, use sort_values(inplace=True) for for INPLACE
sorting
```

```
MEDICINE
```


Participant
500
400
300
200
100
0
0
20
40
60
80
100
120
140
160
180


1400
1200
1000
800
600
400
200
0
0
20
40
60
80
100
120
140
160
180


1400
1200
1000
800
600
400
200
0
0
20
40
60
80
100
120
140
160
180