1 Examination of HIV Variation

In an effort to examine the amount of genetic variation from longitudinal visits of well-controlled patients. We are determining the number of mutations that occur in the LTR over a set of consecutive visits in which the patient has maintained a Viral-Load <100 copies/mL and a CD4 count > 250 cells/mL.

```
from __future__ import division
from pandas import *
import os, os.path
import sys
import numpy as np

sys.path.append('/home/will/HIVReportGen/AnalysisCode/')
sys.path.append('/home/will/PySeqUtils/')
os.chdir('/home/will/HIVVariation/')
```

```
from GeneralSeqTools import call_muscle
```

1.1 Data Extraction

Using the Redcap and sequence data up until 1/16/2013.

```
store = HDFStore('/home/will/HIVReportGen/Data/SplitRedcap/2013-01-16/EntireCohort.hdf')
redcap_data = store['redcap']
seq_data = store['seq_data']

t = redcap_data['Event Name'].dropna().apply(lambda x: x.split(' - ')[0])
t.unique()
redcap_data['Patient visit number'] = redcap_data['Patient visit number'].combine_first(
    t)
```

```
wanted_cols = ['Patient ID', 'Patient visit number', 'Date of visit', 'Latest CD4 count
    (cells/uL)', 'Latest viral load', 'Current ART status']
wanted_redcap = redcap_data[wanted_cols]

data = merge(wanted_redcap, seq_data[['LTR']],
           left_on = ['Patient ID', 'Patient visit number'],
           right_index = True, how = 'inner').dropna()
data = data.rename(columns= {
                            'Patient visit number':'VisitNum',
                            'Date of visit':'Date',
                            'Latest CD4 count (cells/uL)':'CD4',
                            'Latest viral load':'VL',
                            'Current ART status':'ART'
                          })
data['WellControlled'] = (data['VL'] <= 100) & (data['CD4'] >= 250)
data = data[data['ART'] != 'naive']
```

```
print 'Valid samples from Redcap/Sequencing'
print data
print data.head().to_string()
```

```
Valid samples from Redcap/Sequencing
<class 'pandas.core.frame.DataFrame'>
Int64Index: 890 entries, 1 to 1397
Data columns:
Patient ID        890  non-null values
VisitNum          890  non-null values
Date              890  non-null values
CD4               890  non-null values
VL                890  non-null values
ART               890  non-null values
LTR               890  non-null values
WellControlled    890  non-null values
dtypes: bool(1), float64(2), object(5)
   Patient ID VisitNum                 Date  CD4  VL ART
1       A0001      R01  2007-08-15 00:00:00  724  80  on  TACACACCAGGGCCAGGAGTCAGATATCCACTGACCTTTGGATGGT
4       A0001      R04  2009-11-10 00:00:00  689  48  on  CTAGTACCAGTTGAGCCAGAGAAGTTAGAAGAAGCCAACAAAGGAC
7       A0002      R00  2006-09-12 00:00:00  505  50  on  GGTCAGATATCCACTGACCTTTGGATGGTGCTACAAGCTAGTACCA
8       A0002      R01  2007-07-11 00:00:00  737  50  on  CAGGGCCAGGGGTCAGATATCCACTGACCTTTGGATGGTGCTACAA
10      A0002      R03  2008-11-12 00:00:00  734  48  on                                TTGTTACACCCTGTGAGCCTGC
```

```
from copy import deepcopy

def filter_to_runs(df, controlled = True):

    min_run = 3

    ndf = df.copy()
    ndf.sort('Date')
    ndf['RunLen'] = np.nan
    ndf['DaysFromControlled'] = np.nan

    crun = []
    mrun = []
    for ind, row in ndf.iterrows():
        if row['WellControlled'] == controlled:
            crun.append(ind)
        else:
            if len(crun) > len(mrun):
                mrun = deepcopy(crun)
                crun = []

    if len(mrun) >= min_run:
        ndf['RunLen'][mrun] = len(mrun)
        ndf['DaysFromControlled'][mrun] = (ndf['Date'][mrun] - ndf['Date'][mrun[0]]).map(
            lambda x:x.days)

    return ndf
```

```
controlled_data = data.groupby('Patient ID', as_index = False).apply(filter_to_runs).
    dropna()
wild_data = data.groupby('Patient ID', as_index = False).apply(filter_to_runs,
    controlled = False).dropna()
```

```
def align_pat_seq(df):

    df['AlnSeq'] = np.nan
    df['NumCompare'] = np.nan
    df['NumMut'] = np.nan
    seqs = []
    for ind, row in df.iterrows():
        seqs.append((str(ind), row['LTR']))

    aln_seqs = call_muscle(seqs)
    t_seqs = [aln_seqs[0]] + aln_seqs
    new_aln_seqs = []
    num_compare_l = []
    num_mut_l = []
    for (_, row), (_, seq), (_, n_seq) in zip(df.iterrows(), aln_seqs, t_seqs):
        new_aln_seqs.append(seq)
        num_compare = 0
        num_mut = 0
        for s1, s2 in zip(seq, n_seq):
            num_compare += 1
            if (s1 == '-') or (s2 == '-'):
                num_compare -= 1
                continue
            elif s1 != s2:
                num_mut += 1
        num_compare_l.append(num_compare)
        num_mut_l.append(num_mut)


    df['AlnSeq'] = new_aln_seqs
    df['NumCompare'] = num_compare_l
    df['NumMut'] = num_mut_l

    return df



controlled_data = controlled_data.groupby('Patient ID', as_index = False).apply(
    align_pat_seq)
wild_data = wild_data.groupby('Patient ID', as_index = False).apply(align_pat_seq)
```

```
controlled_data['MutPer100bp'] = 100*(controlled_data['NumMut']/controlled_data['
    NumCompare'])
controlled_data['StdMonthsOfControl'] = controlled_data['DaysFromControlled']/30
```

```
wild_data['MutPer100bp'] = 100*(wild_data['NumMut']/wild_data['NumCompare'])
wild_data['StdMonthsOfControl'] = wild_data['DaysFromControlled']/30
```

1.1.1 Well Controlled Samples

```
print 'Well Controlled Patients:', controlled_data['Patient ID'].unique()
print controlled_data
print controlled_data.drop(['LTR', 'AlnSeq'], axis = 1).head().to_string()
print controlled_data.describe()
```

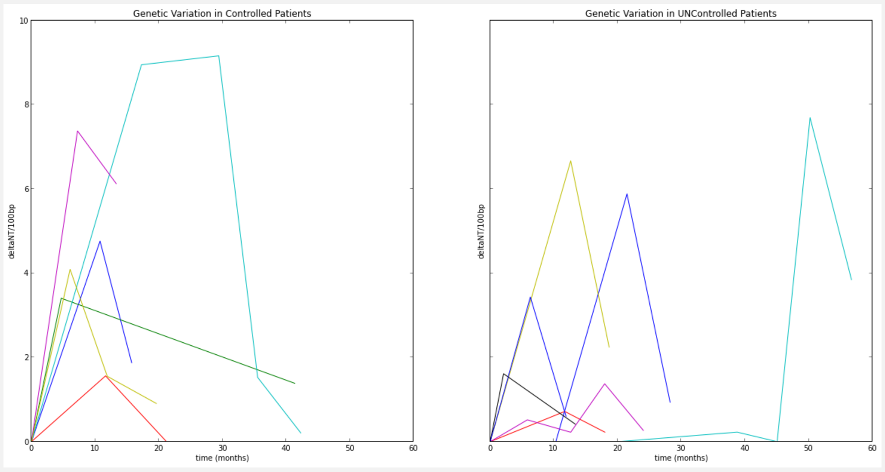
```
Well Controlled Patients: [A0008 A0013 A0025 A0117 A0192 A0305]
<class 'pandas.core.frame.DataFrame'>
Int64Index: 21 entries, 34 to 1027
Data columns:
Patient ID            21  non-null values
VisitNum              21  non-null values
Date                  21  non-null values
CD4                   21  non-null values
VL                    21  non-null values
ART                   21  non-null values
LTR                   21  non-null values
WellControlled        21  non-null values
RunLen                21  non-null values
DaysFromControlled    21  non-null values
AlnSeq                21  non-null values
NumCompare            21  non-null values
NumMut                21  non-null values
MutPer100bp           21  non-null values
StdMonthsOfControl    21  non-null values
dtypes: bool(1), float64(6), int64(2), object(6)
   Patient ID VisitNum                 Date  CD4   VL ART WellControlled  RunLen  DaysFromControlled  Num
34      A0008      R00  2006-09-19 00:00:00  412  59  on           True       3                   0
35      A0008      R01  2007-08-08 00:00:00  372  50  on           True       3                 323
36      A0008      R02  2008-01-04 00:00:00  370  96  on           True       3                 472
50      A0013      R01  2008-06-24 00:00:00  789  48  on           True       3                   0
51      A0013      R02  2008-11-11 00:00:00  624  48  on           True       3                 140
             CD4         VL     RunLen  DaysFromControlled  NumCompare     NumMut  MutPer100bp  StdMonth
count   21.00000  21.000000  21.000000           21.000000   21.000000  21.000000    21.000000
mean   513.52381  51.190476   3.666667          411.238095  425.333333  10.333333     2.518361
std    190.24080  13.507846   0.856349          409.881801   82.682727  12.780193     3.059774
min    256.00000  20.000000   3.000000            0.000000  262.000000   0.000000     0.000000
25%    370.00000  48.000000   3.000000            0.000000  361.000000   0.000000     0.000000
50%    454.00000  48.000000   3.000000          349.000000  447.000000   5.000000     1.526718
75%    678.00000  48.000000   4.000000          588.000000  488.000000  18.000000     4.090909
max    859.00000  96.000000   5.000000         1268.000000  534.000000  40.000000     9.160305
```

1.1.2 Wild Patients

```
print 'Wild Patients:', wild_data['Patient ID'].unique()
print wild_data
print wild_data.drop(['LTR', 'AlnSeq'], axis = 1).head().to_string()
print wild_data.describe()
```

```
Wild Patients: [A0004 A0067 A0093 A0095 A0145 A0188 A0209 A0284]
<class 'pandas.core.frame.DataFrame'>
Int64Index: 30 entries, 19 to 983
Data columns:
Patient ID             30  non-null values
VisitNum               30  non-null values
Date                   30  non-null values
CD4                    30  non-null values
VL                     30  non-null values
ART                    30  non-null values
LTR                    30  non-null values
WellControlled         30  non-null values
RunLen                 30  non-null values
DaysFromControlled     30  non-null values
AlnSeq                 30  non-null values
NumCompare             30  non-null values
NumMut                 30  non-null values
MutPer100bp            30  non-null values
StdMonthsOfControl     30  non-null values
dtypes: bool(1), float64(6), int64(2), object(6)
    Patient ID VisitNum                 Date  CD4   VL ART WellControlled  RunLen  DaysFromControlled  N
19       A0004      R00  2006-09-12 00:00:00  400  276  on          False       4                   0
20       A0004      R01  2007-07-18 00:00:00  546  276  on          False       4                 309
21       A0004      R02  2008-06-17 00:00:00  470  280  on          False       4                 644
22       A0004      R03  2009-01-06 00:00:00  473  450  on          False       4                 847
264      A0067      R02  2008-09-03 00:00:00  240  560  on          False       3                   0
              CD4             VL     RunLen  DaysFromControlled  NumCompare     NumMut  MutPer100bp  Std
count   30.000000      30.000000  30.000000           30.000000    30.00000  30.000000    30.000000
mean   478.100000   19636.166667   4.066667          442.666667   430.30000   5.400000     1.229757
std    174.024245   36042.841278   1.229896          470.830325    73.46599   9.761289     2.128571
min    177.000000      48.000000   3.000000            0.000000   260.00000   0.000000     0.000000
25%    368.500000     276.000000   3.000000           15.750000   386.00000   0.000000     0.000000
50%    478.000000     541.000000   3.500000          368.000000   442.00000   1.000000     0.225735
75%    568.750000   25393.500000   5.000000          592.250000   488.50000   4.750000     1.265449
max    873.000000  144930.000000   6.000000         1701.000000   533.00000  34.000000     7.692308
```

```
fig, axes = plt.subplots(1,2, figsize = (20,10), sharey = True, sharex = True)

plt.sca(axes.flatten()[0])
plt.hold(True)
for pat, df in controlled_data.groupby('Patient ID'):
    plt.plot(df['StdMonthsOfControl'], df['MutPer100bp'])
plt.hold(False)
plt.title('Genetic Variation in Controlled Patients')
plt.xlabel('time (months)')
plt.ylabel('deltaNT/100bp')

plt.sca(axes.flatten()[1])
plt.hold(True)
for pat, df in wild_data.groupby('Patient ID'):
    plt.plot(df['StdMonthsOfControl'], df['MutPer100bp'])
plt.hold(False)
plt.title('Genetic Variation in UNControlled Patients')
plt.xlabel('time (months)')
plt.ylabel('deltaNT/100bp')

plt.savefig('variation_figure.png')
```



In the above figure I determined the number of mutations between consecutive visits in both well controlled (left) and uncontrolled patients (right). Each line represents a single patient.

From these figures it looks like there is a roughly equal amount of variation when you look at well controlled and uncontrolled patients. We can also guess that in general there are bursts of genetic variation which wanes over time. In the Controlled patient figure it looks like all patients eventually return to a no-variation state but it takes 2-4 years of well controlled viral parameters for this to occur. To examine this I'm going to look at consecutive pairs of visits (instead of requiring 3+ visits) and then compare the results of consecutive well-controlled visits to consecutive un-controlled visits.

```
odata = data.groupby('Patient ID', as_index=False).apply(align_pat_seq)
```

```
def pick_consecutive_visits(df):

    ndf = df.copy()
    ndf['ConsecutiveID'] = np.nan
    ndf['ConsecutiveType'] = np.nan

    idx = list(ndf.index)

    wc = list(ndf['WellControlled'])
    gp_ind = 0
    tmp = []
    for (k_a, k_b), (wc_a, wc_b) in zip(zip(idx, idx[1:]), zip(wc, wc[1:])):
        if wc_a == wc_b:
            ndf['ConsecutiveID'].ix[[k_a, k_b]] = gp_ind
            ndf['ConsecutiveType'].ix[[k_a, k_b]] = wc_a
            tmp.append(ndf.ix[[k_a, k_b]].copy())
            gp_ind += 1

    if tmp:
        return concat(tmp, axis = 0, ignore_index = True)
    else:
        return None

cdata = odata.groupby('Patient ID', as_index=False).apply(pick_consecutive_visits).
    reset_index(drop = True).dropna()
print cdata
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 712 entries, 0 to 711
Data columns:
Patient ID         712  non-null values
VisitNum           712  non-null values
Date               712  non-null values
CD4                712  non-null values
VL                 712  non-null values
ART                712  non-null values
LTR                712  non-null values
WellControlled     712  non-null values
AlnSeq             712  non-null values
NumCompare         712  non-null values
NumMut             712  non-null values
MutPer100          712  non-null values
ConsecutiveID      712  non-null values
ConsecutiveType    712  non-null values
dtypes: bool(1), float64(5), int64(2), object(6)
```

```
cdata['MutRate'] = cdata['NumMut']/cdata['NumCompare']
print cdata.drop(['LTR', 'AlnSeq'], axis = 1).head(n=20).to_string()
```

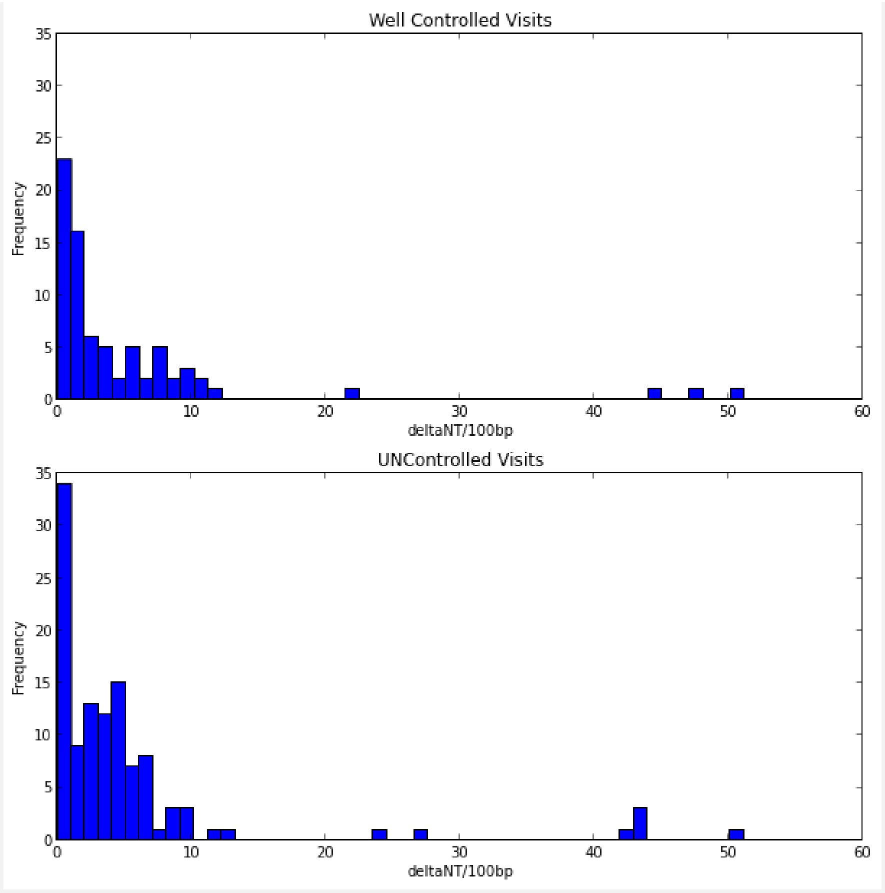
```
Patient ID VisitNum                  Date  CD4   VL ART WellControlled  NumCompare  NumMut  MutPer100  Co
0       A0001      R01  2007-08-15 00:00:00  724   80  on           True         512       0  0.000000
1       A0001      R04  2009-11-10 00:00:00  689   48  on           True         457      14  3.063457
2       A0002      R00  2006-09-12 00:00:00  505   50  on           True         470       0  0.000000
3       A0002      R01  2007-07-11 00:00:00  737   50  on           True         293      28  9.556314
4       A0002      R01  2007-07-11 00:00:00  737   50  on           True         293      28  9.556314
5       A0002      R03  2008-11-12 00:00:00  734   48  on           True         293      22  7.508532
6       A0002      R03  2008-11-12 00:00:00  734   48  on           True         293      22  7.508532
7       A0002      R04  2009-11-03 00:00:00  814   48  on           True         483       3  0.621118
8       A0002      R04  2009-11-03 00:00:00  814   48  on           True         483       3  0.621118
```

```
9        A0002      R05   2010-04-10 00:00:00  764    48   on            True          433        8   1.847575
10       A0002      R05   2010-04-10 00:00:00  764    48   on            True          433        8   1.847575
11       A0002      R09   2012-04-03 00:00:00  926    20   on            True           25        0   0.000000
12       A0004      R00   2006-09-12 00:00:00  400   276   on           False          529        0   0.000000
13       A0004      R01   2007-07-18 00:00:00  546   276   on           False          484        0   0.000000
14       A0004      R01   2007-07-18 00:00:00  546   276   on           False          484        0   0.000000
15       A0004      R02   2008-06-17 00:00:00  470   280   on           False          440       20   4.545455
16       A0004      R02   2008-06-17 00:00:00  470   280   on           False          440       20   4.545455
17       A0004      R03   2009-01-06 00:00:00  473   450   on           False          438        2   0.456621
18       A0004      R04   2009-07-21 00:00:00  427    48   on            True          446        2   0.448430
19       A0004      R05   2010-01-05 00:00:00  491    48   on            True          533        5   0.938086
```

```
tmp = cdata.groupby(['Patient ID', 'ConsecutiveType', 'ConsecutiveID']).last()
ntmp = tmp.groupby(level = ['ConsecutiveType', 'Patient ID']).agg({'MutRate':'mean'})
```

```
fig, axes = plt.subplots(2,1, sharey = True, figsize = (10,10))
plt.sca(axes.flatten()[0])
plt.hist((100*ntmp.ix[0]).values, bins = 50)
plt.title('Well Controlled Visits')
plt.ylabel('Frequency')
plt.xlabel('deltaNT/100bp')

plt.sca(axes.flatten()[1])
plt.hist((100*ntmp.ix[1]).values, bins = 50)
plt.title('UNControlled Visits')
plt.ylabel('Frequency')
plt.xlabel('deltaNT/100bp')
```

```
<matplotlib.text.Text at 0x114f3650>
```



Again, even looking at the variation from consecutive visits I don't see any difference between Uncontrolled visits and Well controlled visits.

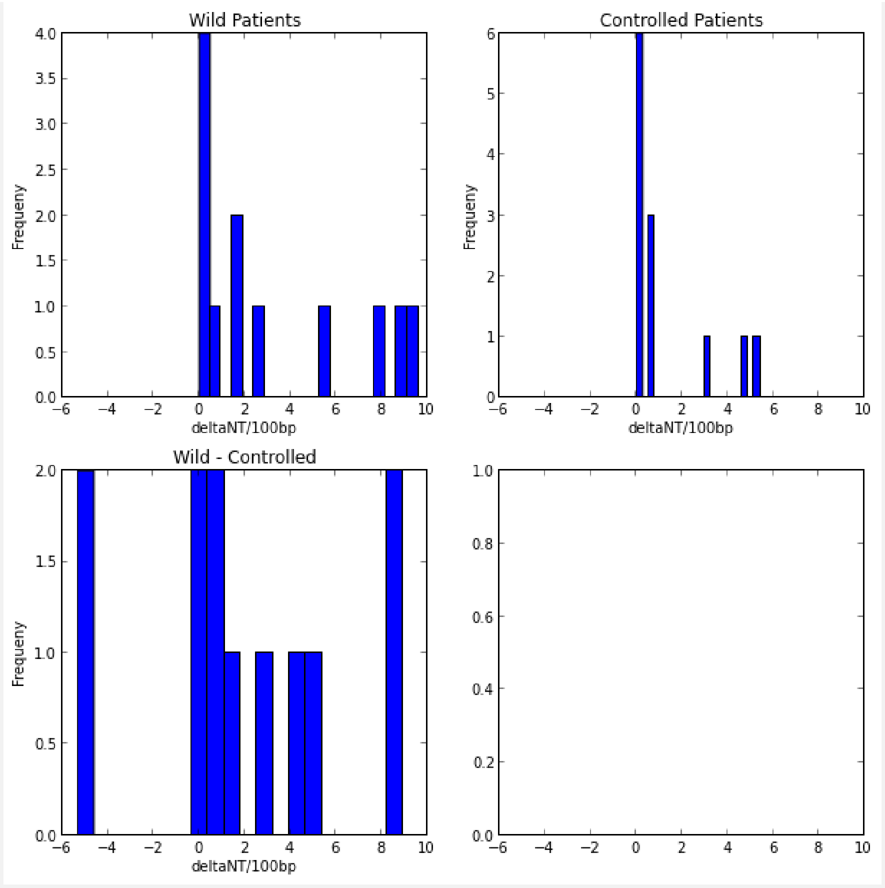
```
paired_data = 100*merge(ntmp.ix[0], ntmp.ix[1],
                  left_index = True, right_index = True,
                  suffixes = ('_Wild', '_Controled'))
paired_data['Difference'] = paired_data['MutRate_Wild'] - paired_data['MutRate_Controled
    ']
print paired_data
```

```
MutRate_Wild  MutRate_Controled  Difference
Patient ID
A0004           1.667359           0.767254    0.900105
A0015           7.823961           3.188438    4.635523
A0019           9.621993           0.647948    8.974045
A0037           1.617251           0.201207    1.416043
A0044           0.000000           4.694264   -4.694264
A0062           5.517704           0.804839    4.712866
A0067           2.621723           0.000000    2.621723
A0096           9.042553           0.236407    8.806147
A0113           0.000000           0.236967   -0.236967
A0145           0.232077           0.267380   -0.035302
```

```
A0162            0.531915            0.130548    0.401367
A0284            0.100806            5.432159   -5.331352
```

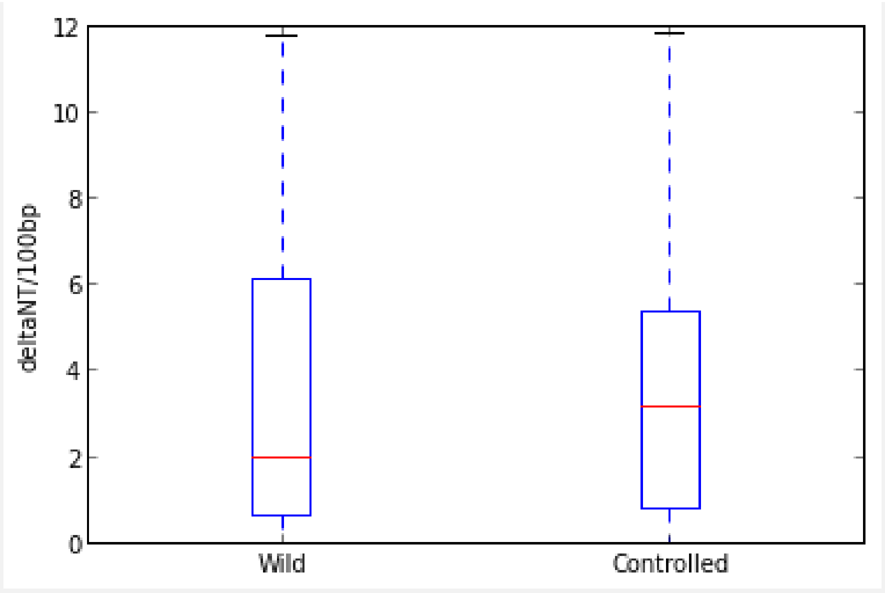
```
fig, axes = plt.subplots(2,2, figsize = (10,10), sharex = True)

plt.sca(axes.flatten()[0])
plt.hist(paired_data['MutRate_Wild'], bins = 20)
plt.title('Wild Patients')
plt.ylabel('Frequeny')
plt.xlabel('deltaNT/100bp')

plt.sca(axes.flatten()[1])
plt.hist(paired_data['MutRate_Controled'], bins = 20)
plt.title('Controlled Patients')
plt.ylabel('Frequeny')
plt.xlabel('deltaNT/100bp')


plt.sca(axes.flatten()[2])
plt.hist(paired_data['Difference'], bins = 20)
plt.title('Wild - Controlled')
plt.ylabel('Frequeny')
plt.xlabel('deltaNT/100bp')



#paired_data[['MutRate_Wild', 'MutRate_Controled', 'Difference']].hist(bins = 20, ax =
    plt.gca(), sharex = True, sharey = True)
```

```
<matplotlib.text.Text at 0x112a4ed0>
```



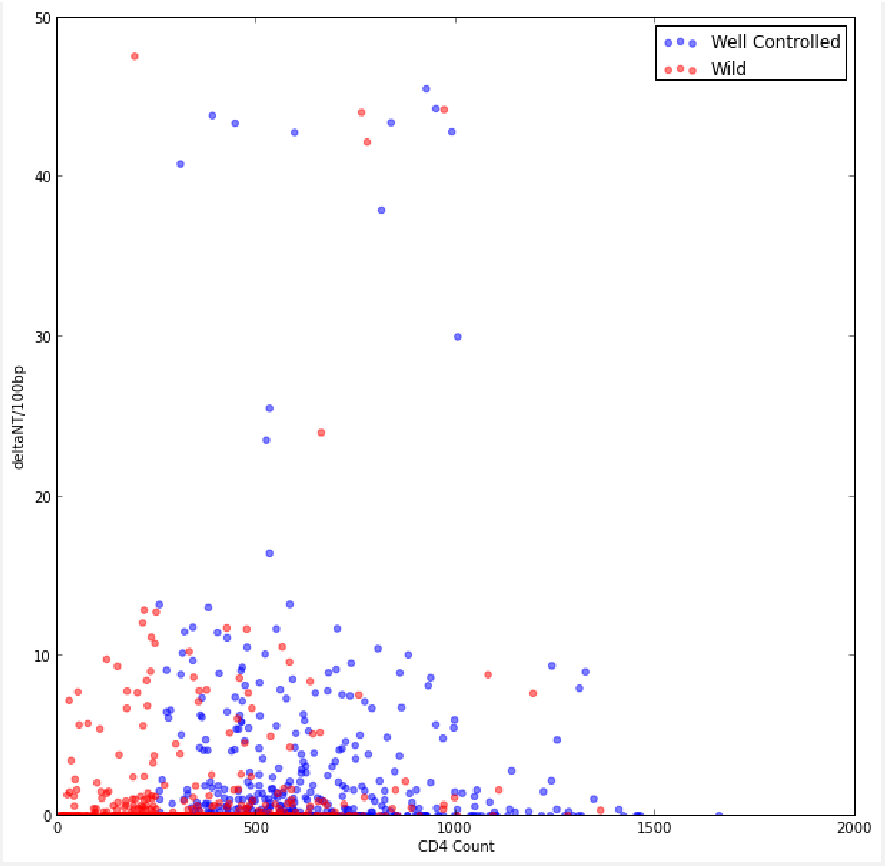
This set of histograms shows the average change when I cluster by patients and ensure that each patient has both a well-controlled and uncontrolled set of visits. The Wild-Controlled histogram shows the difference between the same patient … negative values indicate that the controlled varaition is MORE then the uncontrolled. Maybe this is due to the gouping/pairing? I'll try it by looking at all of data.

```
plt.boxplot([100*ntmp.ix[0], 100*ntmp.ix[1]], sym = '', bootstrap = 1000);
plt.xticks([1,2], ['Wild', 'Controlled']);
plt.ylabel('deltaNT/100bp');
```



Here is a boxplot of the number of mutations/100bp. The box represents the quartile range and the whiskers are the 95% confidence interval. This data is from patients that have both a well-controlled and uncontrolled visit. Here we can see that the Wild and Well Controlled patients are identical.

```
cdict = {True:'b', False:'g'}
odata['MutPer100'] = 100*(odata['NumMut']/odata['NumCompare'])
mask = odata['WellControlled']
plt.figure(figsize = (10,10))
plt.scatter(odata['CD4'].ix[mask].values, odata['MutPer100'].ix[mask].values, color = 'b
    ', alpha = 0.5, s = 20)
plt.hold(True)
plt.scatter(odata['CD4'].ix[~mask].values, odata['MutPer100'].ix[~mask].values, color =
    'r', alpha = 0.5, s = 20)
plt.hold(False)
plt.xlim([0,2000]);
plt.ylim([0,50]);

plt.legend(['Well Controlled', 'Wild']);
plt.xlabel('CD4 Count');
plt.ylabel('deltaNT/100bp');
```



Looking at the correlation between CD4 and number of mutations. I do not really see any relationship between CD4 count and number of mutations. The outliers up there worry me a little, but there are roughly equal numbers of each type.