In [3]:
```
#I recommend installing Anaconda for ease:
#https://www.continuum.io/downloads
#certain packages i.e. seaborn, do not come with the standard installation. A simple search for
#how to 'install seaborn with anaconda' will easily find the installation instructions.
#i.e. in this case:
#http://seaborn.pydata.org/installing.html
#conda install seaborn
```

In [ ]:
```
#With regards to the data, and the study during which this data was collected, rights belong to:
#Friedmann, Peter. Criminal Justice Drug Abuse Treatment Studies (CJ-DATS): Step 'N Out, 2002-2006 [Unit
ed States].
#ICPSR30221-v1. Ann Arbor, MI: Inter-university Consortium for Political and Social Research [distributo
r], 2011-07-27.
#They made the entire dataset openly available at:
#http://doi.org/10.3886/ICPSR30221.v1
```

In [4]:
```
#indicate the directory where your data folder, in this case the file 'stepnout.csv', is located
#in this case, it sits in a folder on the dropbox cloud.
cd Dropbox/Data visualization/Course 5/_32ec92f8b8c6d83a374ae0989bec1447_StepNOut
```

```
/Users/RI/Dropbox/Data visualization/Course 5/_32ec92f8b8c6d83a374ae0989bec1447_StepNOut
```

In [252]:
```
#import the packages we will use
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import statsmodels.formula.api as smf
import statsmodels.stats.multicomp as multi
import statsmodels.api as sm
import scipy.stats
from sklearn.cross_validation import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report
import sklearn.metrics
from sklearn import datasets
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn import preprocessing
from sklearn.linear_model import LassoLarsCV
from sklearn.cluster import KMeans
%matplotlib inline
sns.set(style="whitegrid", color_codes=True)
```

In [192]:
```
#load the full dataset
full_dataset = pd.read_csv('stepnout.csv')
```

In [231]:
```
#show the first ten rows of the dataset to get an idea of the variables and their values
full_dataset.head(10)
```

Out[231]:

| | cid | AGE | CHISP | CHETHN | cblack | casian | cnative | cwhite | cpacisl | cothrac | ... | arst24tlfb_9 | arst25tlfb_9 | alctlfb_9 | alc5tlfb_ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 2180 | 30.0 | 0.0 | NaN | NaN | NaN | NaN | 1.0 | NaN | NaN | ... | 1.0 | 0.0 | 0.0 | 0.0 |
| **1** | 2181 | 26.0 | 1.0 | 7.0 | NaN | NaN | NaN | NaN | NaN | 1.0 | ... | 0.0 | 0.0 | 1.0 | 1.0 |
| **2** | 2183 | 46.0 | 1.0 | 5.0 | NaN | NaN | NaN | NaN | NaN | 1.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 |
| **3** | 2185 | 26.0 | 1.0 | 5.0 | NaN | NaN | NaN | NaN | NaN | 1.0 | ... | 0.0 | 0.0 | 1.0 | 1.0 |
| **4** | 2187 | 37.0 | 1.0 | 7.0 | NaN | NaN | NaN | NaN | NaN | 1.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 |
| **5** | 2188 | 41.0 | 0.0 | NaN | 1.0 | NaN | NaN | NaN | NaN | NaN | ... | 0.0 | 0.0 | 1.0 | 0.0 |
| **6** | 2189 | 45.0 | 0.0 | NaN | NaN | NaN | NaN | 1.0 | NaN | NaN | ... | 0.0 | 0.0 | 0.0 | 0.0 |
| **7** | 2191 | 30.0 | 0.0 | NaN | 1.0 | NaN | NaN | NaN | NaN | NaN | ... | 0.0 | 0.0 | 1.0 | 0.0 |
| **8** | 2192 | 29.0 | 1.0 | 7.0 | NaN | NaN | NaN | NaN | NaN | 1.0 | ... | 0.0 | 0.0 | 1.0 | 0.0 |
| **9** | 2193 | 35.0 | 1.0 | 5.0 | NaN | NaN | NaN | NaN | NaN | 1.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 |

10 rows × 691 columns

In [8]:
```
#rename all columns to small caps
full_dataset.rename(columns = lambda x: x.lower(), inplace=True)
```

In [9]:
```
#same result, different way of doing it
#full_dataset.columns = map(str.lower, full_dataset.columns)
```

In [10]:
```
#same result, different way of doing it
#df = df.rename(columns=lambda x: x.replace('$', ''))
```

In [11]:
```
#try to get all columns, one sees there are 691
full_dataset.columns
```

Out[11]:
```
Index(['cid', 'age', 'chisp', 'chethn', 'cblack', 'casian', 'cnative',
       'cwhite', 'cpacisl', 'cothrac',
       ...
       'arst24tlfb_9', 'arst25tlfb_9', 'alctlfb_9', 'alc5tlfb_9', 'allarrests',
       'anyarrest', 'alldrugs', 'anydrugs', 'allcrimes', 'anycrime'],
      dtype='object', length=691)
```

In [12]:
```
#print all columns/variables
print(full_dataset.describe().to_string())
```

```
/Users/RI/Anaconda/anaconda/lib/python3.5/site-packages/numpy/lib/function_base.py:3834: RuntimeWarning
: Invalid value encountered in percentile
  RuntimeWarning)
```

```
               cid         age       chisp     chethn   cblack   casian   cnative   cwhite   cpacisl   cothra
c     jailcnm      rtprrc      rtarrc      tcu001      tcu002      tcu003     tcu004a     tcu004b
tcu005     tcu006a     tcu006b     tcu006c      tcu007      tcu008      tcu009      tcutot      lcsf001
lcsf002     lcsf003     lcsf004     lcsf005     lcsf006     lcsf007     lcsf008     lcsf009     lcsf010
lcsf011     lcsf012     lcsf013     lcsfsc1     lcsfsc2     lcsfsc3     lcsf014     lcsf015        rctr
csex    dy30free   dy180free       clive      lngliv    homeless      suprliv       livsp       jobsp
numch     minorch       livch       supch      fostch      depend      cmstat      lngmst       grade
hs      votech         ged     workged     lngvtch     lngeduc      drvlic          job     finsup1
amtsup1     finsup2     finsup3     finsup4     finsup5     amtsup5     finsup6     amtsup6     finsup7
amtsup7     finsup8      amtsup8     finsup9    finsup10    finsup11      amtsup11    finsup12    finsu
p13      majsup      fmrel1      fmrel2      fmrel3      fmrel4      fmrel5      fmrel6      fmrel7
fmrel8      fmrel9     fmrel10      prrel1      prrel2      prrel3      prrel4      prrel5      prrel6
prrel7      prrel8      prrel9     prrel10      sbrel1      sbrel2      sbrel3      sbrel4      sbrel5
sbrel6      sbrel7      sbrel8      sbrel9     sbrel10      frchr1      frchr2      frchr3      frchr4
frchr5      frchr6      frchr7      frchr8      frchr9     frchr10     frchr11      gangev       gangc
arstlf      arstdrg      age1ar      arstjv      arst6m     arst30d     jail30d      c1cmlf      c1cm6m
c1cm30      c2cmlf      c2cm6m      c2cm30      c2cmin      c2cmdt      c3cmlf      c3cm6m      c3cm30
c4cmlf      c4cm6m      c4cm30      c5cmlf      c5cm6m      c5cm30      c6cmlf      c6cm6m      c6cm30
```

```
c7cmlf     c7cm6m     c7cm30     c8cmlf     c8cm6m     c8cm30     c9cmlf     c9cm6m     c9cm30
c10cmlf     c10cm6m     c10cm30     c11cmlf     c11cm6m     c11cm30     c12cmlf     c12cm6m     c12cm30
c13cmlf     c13cm6m     c13cm30     c14cmlf     c14cm6m     c14cm30     c14cmin  c14cmdt     c15cmlf
c15cm6m     c15cm30     c15cmin     c15cmdt     c16cmlf     c16cm6m     c16cm30     c16cmin     c16cmdt
c17cmlf     c17cm6m     c17cm30     c17cmin     c17cmdt     c18cmlf     c18cm6m     c18cm30     c18cmin
c18cmdt     c19cmlf     c19cm6m     c19cm30     c19cmin     c19cmdt     c20cmlf     c20cm6m     c20cm30
c20cmin     c20cmdt     c21cmlf     c21cm6m     c21cm30     c22cmlf     c22cm6m     c22cm30     c23cmlf
c23cm6m     c23cm30     c23cmin     c23cmdt     c24cmlf     c24cm6m     c24cm30     c25cmlf  c25cm6m  c2
5cm30     jailc     jailyr     jailmo     jaildy     jailcny     jailcnd     dgtxjlc     dgtxc
ldgtxcm     ldgtxcd     jhumed  jhuemot     jhumsa     jresprg     jipsy     jgpsy     jisa
jgsa     jaillf     age1jl     totjltm     jail6m     dyjl6m     prob6m     parol6m     legstat
health     painlf     dprslf     anxlf     hallulf     conclf     viollf     sidelf     sattlf
pain30d     dprs30d     anx30d     hallu30d     conc30d     viol30d     side30d     satt30d     pypr30d
abuse6m     argue6m     nbirth     brthprb     phosplf     hhosplf     htxphys  htxphdy     htxment     h
txmndy     htxsubs     htxsudy     hdetox     hresprg     otxphys     otxphdy     otxment     otxmndy
otxgpsy     otxsubs     otxsudy     otxgsu     etxphys     etxphdy     etxment     etxmndy     etxsubs
etxsudy     msdp1     msdp2     msdp3     age1alc     alc6m     alc30d     age1tob     tob6m
tob30d     age1inh     inh6m     inh30d     age1mj     mj6m     mj30d     age1hlc     hluc6m
hluc30d     age1crk     crck6m     crck30d     age1coc     coc6m     coc30d     ag1coin     coinj30
age1hco     hcoc6m     hcoc30d     ag1hcin     hcinj30     age1hmo     hmth6m     hmth30d     ag1hmin
hminj30     age1her     her6m     her30d     ag1hein     heinj30     age1mth     mthd6m     mthd30d
ag1mtin     mtinj30     age1opi     opiat6m     opiat30d     ag1opin     opinj30     age1met     meta6m
meta30d     ag1mein     meinj30     age1amp     amph6m     amph30d     ag1amin     aminj30     age1lib
lib6m     lib30d     ag1lbin     lbinj30     age1bar     barb6m     barb30d     ag1bain     bainj30
age1sed     sed6m     sed30d  ag1sdin  sdinj30     age1ghb     ghb6m  ghb30d  ag1gbin     gbinj30
age1ket     ket6m     ket30d     ag1ktin     ktinj30     age1oth     othd6m     othd30d     ag1otin
othinj30     alcdys     beerdys     amtbeer     szbeer     maltdys     amtmalt     szmalt     winedy
s  amtwine1  amtwine2     szwine     fwinedys     amtfwin1     amtfwin2     szfwine     liqrdys     amtliq1
amtliq2     szliquor     typeliq     alcprb1     alcprb2     alcprb3     alcprb4     alcprb5     alcprb6
alcprb7     alcprb8     drgprb1     drgprb2     drgprb3     drgprb4     drgprb5     drgprb6     drgprb7
drgprb8     drugod     lstdrod     intdrod     suprob1     suprob2     suprob3     qtsubs     qtctrk
qttx     qtjlntx     qtjltx     qtother     lngqt     satximp     ndrgtx     ninptx     ag1intx
mosintx     nrestx     ag1rstx     mosrstx     nontx     ag1ontx     mosontx     nodftx     ag1dftx
mosdftx     nomtx     ag1mstx     mosmstx     nothtxa     ag1otxa     mosotxa     resptx     slfhlpm
age1shm     nshlpm     mosshm     shlp30d     medins     aidsr1     aidsr2     aidsr3     aidsr4
```

```
aidsr5      aidsr6      aidsr7      aidsr8      aidsr9      aidsr10     aidsr11     aidsr12     aidsr13
aidsr14     aidsr15     aidsr16     aidsr17     aidsr18      aidsr19     aidsr20     aidsr21     aidsr22
aidsr23     aidsr24     aidsr25     aidsr26     aidsr27      aidsr28     aidsr29     aidsr30        cond
cesi001     cesi002     cesi003     cesi004     cesi005      cesi006     cesi007     cesi008     cesi009
cesi010     cesi011     cesi012     cesi013     cesi014      cesi015     cesi016     cesi017     cesi018
cesi019     cesi020     cesi021     cesi022     cesi023      cesi024     cesi025     cesi026     cesi027
cesi028     cesi029     cesi030     cesi031     cesi032      cesi033     cesi034     cesi035     cesi036
cesi037     cesi038     cesi039     cesi040     cesi041      cesi042     cesi043     cesi044     cesi045
cesi046     cesi047     cesi048     cesi049     cesi050      cesi051     cesi052     cesi053     cesi054
cesi055     cesi056     cesi057     cesi058     cesi059      cesi060     cesi061     cesi062     cesi063
cesi064     cesi065     cesi066     cesi067     cesi068      cesi069     cesi070     cesi071     cesi072
cesi073     cesi074     cesi075     cesi076     cesi077      cesi078     cesi079     cesi080     cesi081
cesi082     cesi083     cesi084     cesi085     cesi086      cesi087     cesi088     cesi089     cesi090
cesi091     cesi092     cesi093     cesi094     cesi095  violent_charge  property_charge  drug_charge
parole_violation  other_charge  sex_charge  arrest_9mo  reincarc_9mo  num_arrest  num_reincarc  numtobd
ays   nummjdays  numdrdays  numdr5days  c1cmtlfb_9  c2cmtlfb_9  c3cmtlfb_9  c4cmtlfb_9  c5cmtlfb_9  c6
cmtlfb_9  c7cmtlfb_9  c8cmtlfb_9  c9cmtlfb_9  c10cmtlfb_9  c11cmtlfb_9  c12cmtlfb_9  c13cmtlfb_9  c14cm
tlfb_9  c15cmtlfb_9  c16cmtlfb_9  c17cmtlfb_9  c18cmtlfb_9  c19cmtlfb_9  c20cmtlfb_9  c21cmtlfb_9  c22c
mtlfb_9  c23cmtlfb_9  c24cmtlfb_9  c25cmtlfb_9   tobtlfb_9   inhtlfb_9    mjtlfb_9  hluctlfb_9  crcktlf
b_9   coctlfb_9  hcoctlfb_9  hmethtlfb_9   hertlfb_9  mthdtlfb_9  opiattlfb_9  methtlfb_9   amptlfb_9
libtlfb_9  barbtlfb_9  sedtlfb_9  ghbtlfb_9   kettlfb_9  othdrugtlfb_9  arst1tlfb_9  arst2tlfb_9  arst3
tlfb_9  arst4tlfb_9  arst5tlfb_9  arst6tlfb_9  arst7tlfb_9  arst8tlfb_9  arst9tlfb_9  arst10tlfb_9  ars
t11tlfb_9  arst12tlfb_9  arst13tlfb_9  arst14tlfb_9  arst15tlfb_9  arst16tlfb_9  arst17tlfb_9  arst18tl
fb_9  arst19tlfb_9  arst20tlfb_9  arst21tlfb_9  arst22tlfb_9  arst23tlfb_9  arst24tlfb_9  arst25tlfb_9
alctlfb_9  alc5tlfb_9  allarrests   anyarrest    alldrugs    anydrugs    allcrimes    anycrime
count   476.000000  474.000000  474.000000  71.000000   244.0     0.0     13.0   161.0      0.0     52.
0  178.000000  452.000000  453.000000  473.000000  473.000000  473.000000  472.000000  472.000000  473.
000000  473.000000  473.000000  469.000000  473.000000  473.000000  473.000000  473.000000  473.000000
473.000000  473.000000  473.000000  473.000000  473.000000  473.000000  473.000000  473.000000  471.000
000  473.000000  473.000000  473.000000  473.000000  471.000000  473.000000  472.000000  473.000000  47
6.000000  476.000000  475.000000  475.000000  476.000000  467.000000  474.000000  473.000000  474.00000
0  223.000000  475.000000  360.000000  351.000000  351.000000  351.000000  475.000000  476.000000  175.
000000  474.000000  475.000000  474.000000  346.000000  342.000000  475.000000  474.000000  472.000000
473.000000  474.000000   267.000000  467.000000  474.000000  473.000000  471.000000   115.000000  472.0
00000  92.000000  471.000000   102.000000  476.000000   164.000000  473.000000  470.000000  472.000000
```

359.000000 474.000000 466.000000 467.000000 348.000000 347.000000 344.000000 300.000000 346.000
000 348.000000 347.000000 346.000000 348.000000 348.000000 371.000000 370.000000 368.000000 27
8.000000 368.000000 371.000000 370.000000 368.000000 370.000000 370.000000 367.000000 365.00000
0 362.000000 318.000000 364.000000 365.000000 365.000000 365.000000 365.000000 365.000000 411.
000000 395.000000 406.000000 395.000000 398.000000 411.000000 410.000000 408.000000 405.000000
407.000000 411.000000 474.000000 474.000000 473.000000 474.000000 473.000000 419.000000 475.00
0000 475.000000 472.000000 473.000000 468.000000 467.000000 474.000000 463.000000 463.000000 4
57.000000 456.000000 472.000000 470.000000 471.000000 474.000000 457.000000 456.000000 473.0000
00 465.000000 466.000000 474.000000 435.000000 436.000000 474.000000 453.000000 454.000000 474
.000000 442.000000 441.00000 473.000000 452.000000 450.000000 474.000000 426.000000 426.000000
475.000000 410.000000 413.000000 475.000000 435.000000 434.000000 473.000000 445.000000 444.000
000 475.000000 435.000000 434.000000 407.000000 406.0 474.000000 427.000000 425.000000 405.0
00000 405.000000 475.000000 439.000000 440.000000 415.000000 416.000000 475.000000 414.000000
414.000000 387.000000 384.000000 474.000000 409.000000 411.000000 384.000000 384.0 473.000000
413.000000 412.000000 387.000000 387.000000 474.000000 413.000000 414.000000 386.000000 386.000
000 474.000000 436.000000 438.000000 475.000000 432.000000 434.000000 475.000000 411.000000 41
1.000000 384.000000 384.000000 475.000000 458.000000 458.000000 78.000000 177.0 176.0 474.
000000 401.000000 405.000000 406.000000 171.000000 175.000000 401.000000 399.000000 164.000000
157.000000 181.000000 179.0 180.000000 180.000000 181.000000 181.000000 180.000000 180.000000
476.000000 472.000000 467.000000 474.000000 460.000000 468.000000 465.000000 475.000000 476.000
000 476.000000 476.000000 476.000000 476.000000 476.000000 476.000000 475.000000 476.000000 47
6.000000 474.000000 476.000000 476.000000 476.000000 476.000000 476.000000 476.000000 227.00000
0 476.000000 475.000000 93.000000 92.000000 476.000000 476.000000 181.000000 94.0 145.00000
0 91.000000 161.000000 100.000000 38.000000 39.000000 161.000000 102.000000 142.000000 93.0000
00 19.000000 175.000000 111.000000 44.000000 179.000000 108.000000 143.000000 89.000000 149.00
0000 95.000000 469.000000 459.000000 440.000000 476.000000 475.000000 474.000000 475.000000 47
6.000000 475.000000 475.000000 476.000000 476.000000 476.000000 474.000000 472.000000 475.00000
0 475.000000 473.000000 473.000000 474.000000 473.000000 475.000000 473.000000 472.000000 336.
000000 444.000000 475.000000 475.000000 475.000000 221.000000 417.000000 473.000000 474.000000
476.000000 206.000000 409.000000 475.000000 476.000000 475.000000 297.000000 427.000000 474.000
000 474.000000 475.000000 189.000000 407.000000 474.000000 472.000000 473.000000 278.000000 42
0.000000 474.000000 473.000000 472.000000 283.000000 425.000000 474.000000 473.000000 472.00000
0 185.000000 410.00000 474.000000 473.000000 472.000000 235.000000 414.000000 474.000000 473.0
00000 473.000000 203.000000 403.000000 474.000000 473.000000 473.000000 176.0 403.0 474.00
0000 474.000000 474.0 185.0 405.000000 474.000000 474.000000 474.000000 171.000000 402.0000

00 457.000000 460.000000 460.000000 187.000000 186.000000 476.000000 327.000000 243.000000 242
.000000 324.000000 125.000000 124.000000 324.000000 65.000000 53.000000 61.000000 323.000000 8
0.000000 78.000000 86.000000 378.000000 154.000000 155.000000 147.000000 178.000000 465.000000
464.000000 466.000000 464.000000 462.000000 464.000000 464.000000 463.000000 470.000000 469.000
000 468.000000 469.000000 469.000000 469.000000 469.000000 469.000000 475.000000 121.000000 13
0.000000 462.000000 462.000000 462.000000 474.000000 429.000000 425.000000 433.000000 429.00000
0 414.000000 458.000000 473.000000 473.000000 404.000000 202.000000 203.000000 401.000000 292.
000000 288.000000 408.000000 258.000000 257.000000 405.000000 243.000000 241.000000 399.000000
175.00000 173.000000 396.000000 155.000000 156.000000 474.000000 474.000000 379.000000 367.0000
00 371.000000 451.000000 476.000000 471.000000 126.000000 124.000000 125.000000 129.000000 118
.000000 117.000000 118.000000 139.000000 133.000000 464.000000 419.000000 420.000000 421.000000
436.000000 348.000000 399.000000 399.000000 423.000000 331.000000 329.000000 330.000000 330.000
000 331.000000 337.000000 334.000000 333.000000 468.000000 350.000000 349.000000 476.000000 47
1.000000 470.000000 472.000000 472.000000 470.000000 472.000000 468.000000 473.000000 473.00000
0 466.000000 463.000000 470.000000 472.000000 473.000000 473.000000 473.000000 473.000000 473.
000000 473.000000 471.000000 470.000000 471.000000 470.000000 473.000000 472.000000 470.000000
473.000000 473.000000 470.000000 471.000000 473.000000 473.000000 473.000000 472.000000 473.000
000 473.000000 469.000000 470.000000 473.000000 473.000000 472.000000 473.000000 472.000000 47
0.000000 471.000000 472.000000 473.000000 473.000000 471.000000 473.000000 473.000000 473.00000
0 472.000000 473.000000 473.000000 471.000000 470.000000 469.000000 469.000000 471.000000 472.
000000 471.000000 472.000000 472.000000 473.000000 473.000000 471.000000 473.000000 473.000000
473.000000 473.000000 472.000000 472.000000 473.000000 473.000000 473.000000 473.000000 473.000
000 473.000000 470.000000 473.000000 472.000000 473.000000 473.000000 473.000000 472.000000 47
3.000000 473.000000 473.000000 471.000000 467.000000 473.000000 469.000000 472.000000 472.00000
0 476.000000 476.000000 476.000000 476.000000 476.000000 476.000000 476.000000
476.000000 476.000000 476.000000 441.000000 441.000000 441.000000 441.000000 441.000000 441.0
00000 441.000000 441.000000 441.000000 441.000000 441.000000 441.000000 441.000000 441.000000
441.0 441.000000 441.000000 441.000000 441.0 441.000000 441.000000 441.0
441.0 441.0 441.000000 441.000000 441.000000 441.000000 441.000000 441.000000 441.00
0000 441.000000 441.000000 441.000000 441.000000 441.000000 441.000000 441.000000 441.000000
441.000000 441.000000 441.000000 441.000000 441.0 441.0 441.0 441.000000 441.00
0000 441.000000 441.000000 441.000000 441.000000 441.000000 441.000000 441.000000 441.0
00000 441.000000 441.0 441.0 441.000000 441.000000 441.000000 441.0
441.000000 441.0 441.0 441.0 441.0 441.000000 441.000000 441.0
00000 441.000000 441.000000 441.000000 441.000000 441.000000 441.000000 441.000000 441.0000

```
00  441.000000  441.000000
mean   5255.586134   34.118143    0.135021   4.661972     1.0      NaN       1.0      1.0       NaN      1.
0    9.073034    0.674779    0.865342    0.877378    0.723044    0.862579    0.584746    0.447034    0.
680761    0.649049    0.887949    0.454158    0.773784    0.613108    0.598309    7.517970    0.416490
0.670190    0.938689    1.632135    0.319239    0.794926    0.207188    0.619450    0.120507    0.22080
7    1.739958    0.961945    0.541226    9.194503    1.849257   11.038055    9.375000    2.323467    5.
073529    1.170168   28.856842  163.886316    4.531513   41.571734    1.415612    3.553911    0.440928
2.147982    1.880000    1.944444    0.695157    1.165242    0.133903    1.115789    1.764706  104.28571
4   10.886076    0.315789    0.251055    0.401734    0.152047    6.818947    1.974684    0.345339    3.
786469    2.356540  1341.535581    0.447537    0.675105    1.133192    0.127389    95.939130    0.01483
1   0.032609    0.186837   193.539216    0.789916   159.146341    2.606765    0.295745    1.078390   32
32.370474    0.219409    0.070815    6.340471    2.890805    3.014409    1.008721    0.826667    1.2861
27    2.712644    2.936599    2.543353    2.545977    1.675287    3.234501    3.186486    0.317935    0
.280576    0.122283    2.377358    2.654054    2.010870    2.045946    1.037838    3.106267    3.038356
0.908840    0.833333    0.909341    2.328767    2.616438    1.646575    2.013699    1.153425    1.82968
4    2.217722    2.371921    2.531646    1.701005    2.214112    2.534146    2.313725    2.246914    0.
439803    1.948905    0.059072    0.014768   13.107822    15.164557   17.638478    2.534606    1.277895
0.738947    1.184322  654.604651  150.664530   22.304069  460.491561   92.250540   13.816415    0.12035
0    0.024123  816.959746  261.023404   33.594480  550.063291  176.398249   23.337719  666.376321  194.
217204   26.566524  102.578059   29.354023    3.623853  568.339662  191.275938   24.011013   51.419831
10.567873    2.08390  226.029598   47.553097    7.060000  103.457806   27.664319    3.147887   43.06736
8   11.792683    1.874092   57.574737   13.457471    2.377880  206.515856   37.197753    5.567568   34.
454737    4.560920    0.698157    0.041769       0.0   23.289030    2.742389    1.115294    0.404938
0.412346   24.482105    0.797267    0.268182    5.571084    0.012019    0.208421    0.014493    0.02415
5    0.157623    0.002604    7.554852    0.809291    0.272506    0.231771       0.0    0.057082    0.021
792    0.116505    0.403101    0.180879    0.103376    0.297821    0.002415    0.020725    0.005181    4
5.797468    8.768349    1.511416   31.926316    4.386574    0.753456    0.067368    0.143552    0.03163
0    0.013021    0.023438   37.743158    5.877729    1.349345    0.076923       0.0       0.0    0.371308
0.057357    2.822222    4.305419    1.152047    8.908571    0.391521    0.340852    5.548780   13.43949
0    0.121547       0.0    0.094444   93.077778    0.370166    5.292818    3.911111  160.961111   10.123
950   18.889831   69.989293    0.628692   15.669565   95.675214   17.195699    5.065263    3.638655
0.287815    0.258403    0.252101    0.031513    0.271008    0.113445    0.069474    0.069328    0.12395
0    0.124473    0.182773    0.012605    0.155462    0.025210    0.006303    0.010504    2.431718    0.
121849    0.313684    2.075269    0.293478    0.296218    1.605042    0.011050       0.0    0.020690   0.1
64835    0.086957    0.390000    0.578947    2.538462    0.105590    0.921569    0.056338    0.247312    0.
```

```
789474    0.165714    3.801802   3.704545    0.134078    0.324074    0.006993   0.022472    0.026846
0.368421    6.132196    4.084967    2.620455   11.371849    2.865263    2.854430   10.526316    5.09453
8    5.221053    1.305263    0.044118    0.054622   11.720588    2.818565    2.622881    5.835789    0.
301053    0.300211   11.188161    1.886076    1.917548   11.385263    1.213531    1.184322    4.464286
0.362613    5.166316    0.625263    0.534737    5.393665    0.426859    1.084567    0.132911    0.14285
7    1.179612    0.083130    9.696842    1.733193    1.713684    6.878788    0.761124    1.751055    0.
082278    0.069474    0.312169    0.036855    5.966245    0.495763    0.501057    0.647482    0.054762
7.116034    0.854123    0.777542    3.448763    0.364706    1.411392    0.097252    0.006356    0.04324
3    0.05122    3.827004    0.228330    0.186441    0.323404    0.074879    1.086498    0.065539    0.0
50740    0.502463    0.014888    0.835443    0.008457    0.025370     0.0      0.0    0.299578    0.00
2110    0.0      0.0    0.237037    0.491561    0.021097    0.012658    0.076023    0.044776    0.5010
94    0.058696    0.021739    0.021390    0.032258    9.760504   10.217125    5.374486   13.462810    3
.657407    2.952000   12.346774    0.524691    0.738462    0.283019    4.540984    1.219814   1.337500   0
.910256    6.965116    5.515873    3.318182    1.070968   13.095238    1.308989    0.733333    0.969828
1.068670    1.088362    0.943723    0.961207    0.885776    0.907127    1.768085    2.117271    2.17307
7    2.136461    1.989339    2.268657    1.601279    2.189765    0.509474   43.347107    0.200000    2.
577922    2.463203    2.162338    4.784810    1.515152    0.616471    2.256351    1.125874    0.026570
37.100437    3.059197    2.484144    0.591584    0.856436    1.123153    1.024938   17.339041    7.5347
22    0.901961   18.468992    5.229572    0.693827   16.032922    3.734440    0.205514    8.16000    3.
236994    0.179293    0.077419    0.096154    4.333333    0.776371   25.245383    3.850136   35.137466
0.090909    0.336134    1.188960    0.507937    0.959677    0.672000   40.922481    2.508475    4.64957
3   18.966102    0.834532    2.624060    5.635776    0.758950    0.245238    0.209026    2.754587    1.
364943    1.799499   35.982456   27.661939    8.145015    6.018237   11.215152   17.342424    5.419940
21.700297   16.715569    5.543544    6.049145    0.120000    0.134670    0.510504    3.785563    3.7042
55    4.059322    2.345339    3.404255    3.243644    3.653846    3.879493    3.663848    3.935622    2
.345572    3.987234    2.455508    4.093023    4.285412    4.475687    3.403805    2.938689    3.513742
3.411890    2.274468    2.721868    3.691489    2.520085    3.688559    4.072340    4.543340    2.69344
6    3.155319    2.925690    2.270613    1.792812    3.860465    3.495763    2.437632    3.761099    3.
942431    3.563830    4.414376    3.247357    2.434322    2.782241    3.110169    3.489362    3.165605
2.040254    2.682875    2.985201    3.078556    3.665962    2.467230    3.712474    2.675847    3.54545
5    2.917548    3.949045    2.634043    2.944563    4.076759    4.065817    2.012712    2.719745    2.
824153    3.474576    2.640592    2.581395    2.273885    3.479915    1.536998    3.340381    3.632135
4.021186    1.976695    2.923890    2.496829    3.367865    3.433404    4.257928    2.391121    3.50425
5    3.380550    3.519068    2.560254    4.672304    2.879493    2.796610    2.826638    2.378436    2.
687104    3.337580    2.304069    3.750529    3.601279    2.796610    2.480932        0.027311
```

```
0.058824    0.100840        0.199580    0.105042    0.008403    0.422269     0.361345    0.628151
0.455882   38.376417    7.287982   10.641723    4.514739    0.081633    0.040816    0.188209    0.05442
2    0.079365    0.009070    0.049887    0.009070    0.009070    0.013605         0.0    0.011338
0.015873    0.004535         0.0    0.011338    0.002268         0.0         0.0         0.0
0.011338    0.006803    0.004535    0.158730    0.043084    0.263039    0.011338    0.185941    0.0
40816    0.113379    0.120181    0.018141    0.009070    0.099773    0.011338    0.054422    0.029478
0.006803    0.004535         0.0         0.0         0.0    0.002268     0.004535    0.011338    0.01
5873    0.068027    0.024943    0.022676    0.002268    0.018141    0.002268    0.002268
0.0         0.0    0.009070    0.009070    0.002268         0.0    0.011338         0.0
0.0         0.0         0.0    0.002268    0.002268    0.002268    0.156463    0.036281
0.376417    0.181406    0.399093    0.251701    0.712018    0.442177    0.723356    0.335601
std    2166.587787    8.836234    0.342107    2.048997    0.0    NaN    0.0    0.0    NaN    0.
0    8.394258    0.468977    0.917849    0.328350    0.447968    0.344655    0.493289    0.497714    0.
466675    0.477773    0.315763    0.498426    0.418823    0.487554    0.490759    3.291649    0.493499
0.470642    0.840225    0.647668    0.530420    0.675083    0.405721    0.732856    0.325899    0.41523
2    0.493635    0.732587    0.498825    3.247589    0.572968    3.374667   18.672995    2.540148    2.
884597    0.376176    4.978491   38.277409    1.048836   78.613172   10.999628   21.401869    0.497023
1.270094    1.859041    2.077126    1.108901    1.404499    0.651825    1.656487    1.376224   99.10489
4    1.875540    0.465320    0.434078    0.490959    0.359592   51.717025   17.392843    0.475983    2.
831958    2.701008  1426.043230    1.493523    1.784338    2.192345    0.825597  306.572572    0.27988
5   0.312772    1.016417  933.495885    1.931997  795.463790    2.793325    1.235484    2.192488   33
29.188366    0.873840    0.605820    3.932604    1.086796    1.076453    1.274296    1.197805    1.6193
34    1.248935    1.181022    1.273873    1.049584    1.191114    1.061104    1.123936    0.801568    0
.819692    0.570619    1.358825    1.308425    1.330789    1.147897    1.158766    1.112345    1.173732
1.181566    1.131938    1.277058    1.305681    1.292733    1.306585    1.166063    1.196846    1.49640
0    1.330314    1.220038    1.184025    1.073401    1.290829    1.363174    1.466952    1.420595    1.
005251    1.356041    0.236008    0.120750   20.158036  115.833054    5.896562    4.214990    1.543544
0.622418    4.784743  457.098914  232.232542   25.414032  476.472629  189.650367   20.998732    0.73206
7    0.320489  367.191083  332.926965   32.676153  482.471579  302.786791   31.822101  458.756852  293.
043590   30.932408  297.542207  149.281470   15.755919  479.105073  317.588788   33.127653  195.128035
75.068351   11.24418  394.866920  156.045787   18.307773  290.552421  130.972211   12.905419  193.83834
4   80.128036    9.682014  217.080738   87.977557   11.723531  389.829401  141.019778   16.521617  165.
268505   50.192375    5.803053    0.391693     0.0  139.268182   48.374412    8.021985    5.009790
4.496461  131.672931    7.579255    2.710124   19.038840    0.129316    3.082675    0.208262    0.33285
9    2.553766    0.051031   80.171870   10.020705    2.767177    2.983143     0.0    0.354925    0.201
```

956    1.567933    5.079451    2.565887    1.011493    4.172369    0.049147    0.296447    0.101797  19
2.356031   56.428606    7.304656  162.646159   50.355037    6.320285    0.562252    2.497086    0.44334
9    0.210277    0.459279  176.102978   50.988932    6.195591   0.503732       0.0       0.0    0.483665
0.586687    6.394459    8.708181    2.333952    8.684951    0.488700    0.474591    6.865317   17.76532
8    0.764947       0.0    1.267105   78.239816    2.178428   65.634234    5.659114  125.599228   19.366
041    6.604102   63.954835    1.047144   37.340912   87.904935   51.548462    2.405592    1.015510
0.453221    0.438217    0.434676    0.174882    0.444948    0.317470    0.254526    0.254278    0.32987
1   0.330469    0.386887    0.111680    0.362726    0.156928    0.079221    0.102058    1.343110    0.
327455    0.464479    1.929387    0.763710    1.088923    3.937338    0.104825       0.0    0.142837   1.0
35629    0.282650    1.136137    1.426233    7.465169    0.308271    4.420220    0.231390   1.017840   2.
070398    0.372891    8.645356    6.739607    0.341692    0.993706    0.083624   0.149052    0.162177
3.084114    3.866303    3.796599    3.700525    6.058866    2.949425    3.021752    7.158309    3.65534
8    5.271888    4.459082    0.463564    0.536224    5.683557    3.211500    3.239510    8.140277    1.
261194    1.317556   12.168333    3.074772    3.156841   10.169294    2.407366    2.489453    9.251312
1.486295    9.942860    1.869868    1.795551   10.255985    1.640839    5.190159    0.875587    0.89240
7    5.054027    0.746189   11.779474    3.046695    3.051962   11.306784    2.171970    6.751184    0.
616529    0.596593    2.558447    0.433928   10.127252    1.558479    1.619694    3.997948    0.577503
10.151768    2.173090    2.109347    8.473992    1.520668    5.208299    1.888596    0.102836    0.4523
73    0.58907    8.088806    1.020282    0.974835    2.710711    0.779083    4.644543    0.591193    0.
553843    4.709769    0.298881    3.976454    0.183920    0.410909       0.0       0.0    2.485246    0.0
45932       0.0       0.0    3.085917    3.270234    0.311136    0.275589    0.485142    0.517027    3.068
033    0.551123    0.301711    0.206278    0.327203   12.083281   11.583729    5.558450    8.803565
8.152375    4.319603   12.322921    2.884142    1.278145    0.661506    9.999290    4.026077    1.861969
1.311150   7.670337    9.365041    5.101204    1.284825   12.139787    0.487155    1.204279    1.322327
1.367678    1.333549    1.403850    1.357566    1.216292    1.305173    1.514480    1.482483    1.49587
6    1.472879    1.663424    1.604048    1.385548    1.544297    1.809904   63.013849    1.137112    1.
225802    1.229495    1.180853    5.472601    3.144923    1.330418    4.100741    1.755634    0.188711
52.565919    1.114563    3.316667    3.510216    1.231503    2.094218    1.623076   14.068492   10.3232
85    2.754527   15.099959    6.987001    1.285905   14.506444    5.561401    0.920383   15.77819    9.
155404    3.031593    0.477214    0.760126    1.803997    0.417116    8.883160    1.289679   87.138132
0.287799    0.472883    2.629692    1.451873    2.069520    2.089822   99.246034   10.879877   21.65494
1  43.749663    5.180226   10.735527   29.269266    1.153997    0.756421    0.752149   10.987427   11.
437923    2.533177   80.265590   63.349715   33.108684   59.018390   69.439784   63.311655   58.272729
59.011216   58.164311   56.337211   47.305872    0.861398    0.913926    0.500416    1.341119    1.2928
30    1.007761    1.140557    1.246908    1.172021    1.220737    1.114362    1.187463    1.034947    1

```
.083833    1.232055    1.233869    0.922766    0.779055    0.657374    1.263799    1.432982    1.216205
1.298193    1.132570    1.194460    1.237531    1.166317    1.186637    1.021663    0.694045    1.23383
6    1.320384    1.217251    1.036783    0.872964    0.876712    1.293452    1.056066    0.878177    0.
984328    1.210812    0.696210    1.278952    1.156505    1.228692    1.145721    1.287505    1.316897
1.044890    1.200663    1.261302    1.360611    1.001294    1.049356    0.916996    1.118655    1.11923
9    1.250241    0.767390    1.145172    1.247698    1.061402    0.802867    0.983865    1.267437    1.
178726    1.234410    1.103635    1.078602    1.081438    1.083446    0.755772    0.943531    0.967689
0.895599    0.940641    1.196092    1.033594    1.061650    1.159113    0.757075    1.093744    1.33470
3    1.187417    1.167344    1.159461    0.556562    1.168197    1.171562    1.112129    1.304539    1.
208805    1.255901    0.977013    0.897929    1.304528    1.117772    1.218948        0.163159
0.235542    0.301434        0.400105    0.306930    0.091380    0.494441      0.480896    0.928137
0.704980   75.427893   28.308545   29.310927   18.863394    0.274115    0.198089    0.391323    0.22710
6    0.270615    0.094913    0.217958    0.094913    0.094913    0.115978        0.0     0.105994
0.125126    0.067267        0.0     0.105994    0.047619        0.0         0.0         0.0
0.105994    0.082291    0.067267    0.365839    0.203277    0.440783    0.105994    0.389501    0.1
98089    0.317415    0.325543    0.133611    0.094913    0.300038    0.105994    0.227106    0.169335
0.082291    0.067267        0.0         0.0         0.0    0.047619     0.067267    0.105994    0.12
5126    0.252079    0.156129    0.149037    0.047619    0.133611    0.047619    0.047619
0.0         0.0     0.094913    0.094913    0.047619        0.0     0.105994        0.0
0.0         0.0         0.0     0.047619    0.047619    0.047619    0.363706    0.187201
0.485037    0.385792    0.931470    0.434483    1.064325    0.497209    1.460269    0.472736
min    2180.000000   18.000000    0.000000    0.000000     1.0     NaN     1.0     1.0     NaN     1.
0    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.
000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000
0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.00000
0    0.000000    0.000000    0.000000    1.000000    0.000000    1.000000    0.000000    0.000000    1.
000000    1.000000    0.000000    0.000000    1.000000    1.000000    0.000000    0.000000    0.000000
1.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    1.000000    0.00000
0    4.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    1.
000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.00000
0    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000
0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.00000
0    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.
000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000
0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.00000
```

```
0    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.
000000    0.000000    0.000000    0.000000    0.000000     0.000000    0.000000    0.000000    0.000000
0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.00000
0    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.
000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000
0.000000    0.00000    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000
0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.00000
0    0.000000    0.000000    0.000000     0.0    0.000000    0.000000    0.000000    0.000000    0.000
000    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000
0.000000    0.000000    0.000000    0.000000    0.000000    0.000000     0.0    0.000000    0.000000
0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.00000
0    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.
000000    0.000000    0.000000    0.000000    0.000000    0.000000     0.0     0.0    0.000000    0.00
0000    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000
0.000000     0.0    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000
7.000000    0.000000    0.000000    0.000000    0.000000    0.000000    1.000000    1.000000    0.00000
0    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.
000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    1.000000    0.000000
0.000000   0.000000   0.000000    0.000000    0.000000    0.000000     0.0    0.000000   0.000000    0
.000000    0.000000   0.000000   0.000000    0.000000    0.000000    0.000000   0.000000   0.000000
0.000000    0.000000   0.000000    0.000000    0.000000    0.000000   0.000000    0.000000   0.000000
0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.00000
0    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.
000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000
0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.00000
0    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.
000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000
0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.00000
0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.00000
0    0.000000    0.000000    0.000000    0.000000     0.0     0.0    0.000000    0.000000     0.0
0.0    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000
0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.00000
0    0.000000    0.000000   0.000000   0.000000   0.000000    0.000000   0.000000    0.000000   0.000000
0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.00000
0    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.
```

```
000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    1.000000    1.000000
1.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    1.00000
0    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.
000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.00000    0.000000    0.000000
0.000000    0.000000    1.000000    0.000000    0.000000    1.000000    0.000000    0.000000    0.00000
0    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.
000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000
0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.00000
0    0.000000    0.000000    0.000000    0.000000    0.000000    1.000000    1.000000    1.000000    1.
000000    1.000000    1.000000    1.000000    1.000000    1.000000    1.000000    1.000000    1.000000
1.000000    1.000000    1.000000    1.000000    1.000000    1.000000    1.000000    1.000000    1.00000
0    1.000000    1.000000    1.000000    1.000000    1.000000    1.000000    1.000000    1.000000    1.
000000    1.000000    1.000000    1.000000    1.000000    1.000000    1.000000    1.000000    1.000000
1.000000    1.000000    1.000000    1.000000    1.000000    1.000000    1.000000    1.000000    1.00000
0    1.000000    1.000000    1.000000    1.000000    1.000000    1.000000    1.000000    1.000000    1.
000000    1.000000    1.000000    1.000000    1.000000    1.000000    1.000000    1.000000    1.000000
1.000000    1.000000    1.000000    1.000000    1.000000    1.000000    1.000000    1.000000    1.00000
0    1.000000    1.000000    1.000000    1.000000    1.000000    1.000000    1.000000    1.000000    1.
000000    1.000000    1.000000    1.000000    1.000000    1.000000    1.000000    1.000000    1.000000
1.000000    1.000000    1.000000    1.000000    1.000000        0.000000        0.000000    0.000000
0.000000      0.000000    0.000000    0.000000      0.000000    0.000000      0.000000    0.000000    0
.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000
0.000000    0.000000    0.000000     0.000000        0.0     0.000000     0.000000     0.000000
0.0     0.000000     0.000000        0.0         0.0          0.0     0.000000     0.000000     0.00
0000     0.000000     0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000
0.000000     0.000000    0.000000    0.000000     0.000000    0.000000    0.000000    0.000000
0.0        0.0        0.0    0.000000      0.000000     0.000000     0.000000     0.000000     0.00000
0     0.000000     0.000000     0.000000     0.000000     0.000000          0.0          0.0      0.0
00000      0.000000     0.000000         0.0     0.000000          0.0          0.0          0.0
0.0     0.000000     0.000000     0.000000     0.000000     0.000000    0.000000    0.000000    0.
000000    0.000000    0.000000    0.000000    0.000000    0.000000
25%    4023.250000        NaN         NaN        NaN      NaN     NaN       NaN     NaN       NaN      Na
N        NaN         NaN         NaN         NaN         NaN         NaN         NaN         NaN
NaN         NaN         NaN         NaN         NaN         NaN         NaN         NaN         NaN
NaN         NaN         NaN         NaN         NaN         NaN         NaN         NaN         NaN
```

```
NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN    3.000000
1.000000          NaN          NaN    4.000000          NaN          NaN          NaN          NaN          Na
N          NaN          NaN          NaN          NaN          NaN          NaN    1.000000          NaN
NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN
NaN           NaN          NaN          NaN          NaN          NaN           NaN          NaN          NaN
NaN           NaN    0.000000          NaN          NaN          NaN          NaN           NaN          NaN
NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN
NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN
NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN
NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN
NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN
NaN           NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN
NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN
NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN
NaN          NaN          NaN          NaN          NaN         NaN          NaN          NaN          NaN
NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN
NaN          NaN          NaN          NaN          NaN          NaN          NaN      NaN          NaN
NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN
NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN
NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN
NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN
NaN          NaN          NaN          NaN          NaN          NaN          NaN         NaN       NaN       Na
N          NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN
NaN          NaN          NaN      NaN          NaN          NaN          NaN          NaN          NaN
NaN    3.000000          NaN          NaN          NaN          NaN          NaN          NaN          NaN
3.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000          NaN    0.00000
0    0.000000          NaN    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000
NaN    0.000000          NaN          NaN          NaN    0.000000    0.000000          NaN       NaN
NaN         NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN
NaN         NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN
NaN         NaN          NaN          NaN          NaN    9.000000          NaN          NaN          NaN    0
.000000          NaN          NaN    0.000000    0.000000    11.000000          NaN          NaN          NaN
NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN
NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN    0.000000
NaN          NaN          NaN    0.000000          NaN          NaN          NaN          NaN          NaN
```

```
NaN NaN NaN NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN NaN NaN
NaN NaN 0.000000 NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN NaN NaN N
aN NaN NaN NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN 0.000000 NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN NaN NaN
0.000000 NaN NaN NaN NaN NaN NaN NaN Na
N NaN NaN NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN 0.000000 0.0
00000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 NaN NaN NaN NaN NaN NaN NaN Na
N NaN NaN NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN NaN NaN
```

```
NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN
NaN           NaN           NaN           NaN           NaN           NaN           NaN           NaN
NaN           NaN           NaN           NaN           NaN           NaN          NaN          NaN
NaN         NaN         NaN         NaN         NaN         NaN
50%    4453.500000         NaN         NaN         NaN      NaN      NaN       NaN      NaN       NaN       Na
N         NaN         NaN         NaN         NaN         NaN         NaN         NaN         NaN
NaN         NaN         NaN         NaN         NaN         NaN         NaN         NaN         NaN
NaN         NaN         NaN         NaN         NaN         NaN         NaN         NaN         NaN
NaN         NaN         NaN         NaN         NaN         NaN         NaN         NaN    5.000000
1.000000         NaN         NaN    5.000000         NaN         NaN         NaN         NaN         Na
N         NaN         NaN         NaN         NaN         NaN         NaN    1.000000         NaN
NaN         NaN         NaN         NaN         NaN         NaN         NaN         NaN         NaN
NaN          NaN         NaN         NaN         NaN         NaN          NaN         NaN         NaN
NaN          NaN    0.000000          NaN         NaN         NaN         NaN          NaN          NaN
NaN         NaN         NaN         NaN         NaN         NaN         NaN         NaN         NaN
NaN         NaN         NaN         NaN         NaN         NaN         NaN         NaN         NaN
NaN         NaN         NaN         NaN         NaN         NaN         NaN         NaN         NaN
NaN         NaN         NaN         NaN         NaN         NaN         NaN         NaN         NaN
NaN         NaN         NaN         NaN         NaN         NaN         NaN         NaN         NaN
NaN          NaN         NaN         NaN         NaN         NaN         NaN         NaN         NaN
NaN         NaN         NaN         NaN         NaN         NaN         NaN         NaN         NaN
NaN         NaN         NaN         NaN         NaN         NaN         NaN         NaN         NaN
NaN         NaN         NaN         NaN         NaN        NaN         NaN         NaN         NaN
NaN         NaN         NaN         NaN         NaN         NaN         NaN         NaN         NaN
NaN         NaN         NaN         NaN         NaN         NaN         NaN     NaN         NaN
NaN         NaN         NaN         NaN         NaN         NaN         NaN         NaN         NaN
NaN         NaN         NaN         NaN         NaN         NaN         NaN         NaN         NaN
NaN         NaN         NaN         NaN         NaN         NaN         NaN         NaN         NaN
NaN         NaN         NaN         NaN         NaN         NaN         NaN         NaN         NaN
NaN         NaN         NaN         NaN         NaN         NaN         NaN      NaN      NaN       Na
N         NaN         NaN         NaN         NaN         NaN         NaN         NaN         NaN
NaN         NaN         NaN     NaN         NaN         NaN         NaN         NaN         NaN
NaN    5.000000         NaN         NaN         NaN         NaN         NaN         NaN         NaN
4.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000         NaN    0.00000
0    0.000000         NaN    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000
```

```
NaN    0.000000         NaN         NaN         NaN    0.000000    1.000000         NaN       NaN
NaN          NaN         NaN         NaN         NaN         NaN         NaN         NaN         NaN
NaN          NaN         NaN         NaN         NaN          NaN         NaN         NaN         NaN
NaN          NaN         NaN         NaN          NaN   13.000000          NaN         NaN         NaN    8
.000000          NaN          NaN    0.000000    0.000000   13.000000          NaN          NaN          NaN
NaN           NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN
NaN           NaN          NaN          NaN          NaN          NaN          NaN          NaN    0.000000
NaN           NaN          NaN    0.000000          NaN          NaN          NaN          NaN          NaN
NaN           NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN
NaN           NaN          NaN          NaN          NaN          NaN          NaN          NaN         NaN
NaN           NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN
NaN           NaN          NaN          NaN         NaN        NaN          NaN          NaN       NaN       NaN
NaN           NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN
NaN           NaN     3.000000          NaN          NaN          NaN          NaN          NaN          NaN
NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN          N
aN           NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN
NaN           NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN
NaN           NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN
NaN           NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN
NaN           NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN
NaN           NaN          NaN         NaN          NaN          NaN          NaN          NaN          NaN
NaN           NaN          NaN          NaN          NaN    0.000000          NaN          NaN          NaN
NaN           NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN
NaN           NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN
NaN           NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN
1.000000          NaN          NaN          NaN          NaN          NaN          NaN          NaN          Na
N          NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN
NaN           NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN
NaN           NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN
NaN           NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN
NaN           NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN
NaN           NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN
NaN           NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN
NaN           NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN
NaN           NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN
```

```
NaN          NaN          NaN          NaN          NaN          NaN          NaN        0.000000        0.0
00000     0.000000          0.000000      0.000000     0.000000     0.000000      0.000000     0.000000
0.000000         NaN          NaN          NaN          NaN          NaN          NaN          NaN          Na
N         NaN          NaN          NaN          NaN          NaN           NaN           NaN           NaN
NaN           NaN           NaN           NaN           NaN           NaN           NaN           NaN
NaN           NaN           NaN           NaN           NaN          NaN          NaN          NaN          NaN
NaN          NaN          NaN           NaN          NaN          NaN           NaN          NaN          NaN
NaN          NaN         NaN          NaN          NaN            NaN           NaN           NaN           NaN
NaN           NaN           NaN           NaN           NaN           NaN            NaN            NaN
NaN            NaN            NaN            NaN            NaN            NaN            NaN            NaN
NaN            NaN            NaN            NaN            NaN            NaN          NaN          NaN
NaN          NaN          NaN          NaN          NaN          NaN
75%     7020.500000         NaN          NaN         NaN       NaN       NaN        NaN       NaN        NaN       Na
N         NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN
NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN
NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN
NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN    7.000000
1.000000         NaN          NaN    5.000000          NaN          NaN          NaN          NaN          Na
N         NaN          NaN          NaN          NaN          NaN          NaN    2.000000          NaN
NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN
NaN           NaN          NaN          NaN          NaN          NaN           NaN          NaN          NaN
NaN           NaN    0.000000          NaN          NaN          NaN           NaN            NaN          NaN
NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN
NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN
NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN
NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN
NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN
NaN           NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN
NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN
NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN
NaN          NaN          NaN          NaN          NaN         NaN          NaN          NaN          NaN
NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN
NaN          NaN          NaN          NaN          NaN          NaN          NaN        NaN          NaN
NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN
NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN
```

```
NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN
NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN
NaN          NaN          NaN          NaN          NaN          NaN          NaN         NaN        NaN        Na
N          NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN
NaN          NaN          NaN       NaN          NaN          NaN          NaN          NaN          NaN
NaN   10.000000          NaN          NaN          NaN          NaN          NaN          NaN          NaN
4.000000    1.000000    1.000000    1.000000    0.000000    1.000000    0.000000          NaN    0.00000
0    0.000000          NaN    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000
NaN    0.000000          NaN          NaN          NaN    0.000000    2.000000          NaN       NaN
NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN
NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN
NaN          NaN          NaN          NaN          NaN   15.000000          NaN          NaN          NaN    8
.000000          NaN          NaN    0.000000    0.000000   15.000000          NaN          NaN          NaN
NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN
NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN    0.000000
NaN          NaN          NaN    3.000000          NaN          NaN          NaN          NaN          NaN
NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN
NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN
NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN
NaN          NaN          NaN          NaN       NaN       NaN          NaN          NaN       NaN       NaN
NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN
NaN          NaN   20.000000          NaN          NaN          NaN          NaN          NaN          NaN
NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN          N
aN          NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN
NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN
NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN
NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN
NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN
NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN
NaN          NaN          NaN          NaN          NaN    1.000000          NaN          NaN          NaN
NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN
NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN
NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN
1.000000          NaN          NaN          NaN          NaN          NaN          NaN          NaN          Na
N          NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN
```

```
NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN
NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN
NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN
NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN
NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN
NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN
NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN
NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN          NaN
NaN          NaN          NaN          NaN          NaN          NaN          NaN       0.000000        0.0
00000     0.000000          0.000000     0.000000     0.000000     1.000000      1.000000     1.000000
1.000000         NaN          NaN          NaN          NaN          NaN          NaN          NaN         Na
N          NaN          NaN          NaN          NaN          NaN           NaN           NaN           NaN
NaN           NaN           NaN           NaN           NaN           NaN           NaN           NaN
NaN           NaN           NaN           NaN           NaN          NaN          NaN          NaN          NaN
NaN          NaN          NaN           NaN          NaN          NaN           NaN          NaN          NaN
NaN          NaN         NaN          NaN          NaN               NaN           NaN          NaN           NaN
NaN           NaN           NaN           NaN           NaN           NaN            NaN            NaN
NaN            NaN            NaN            NaN            NaN            NaN            NaN            NaN
NaN            NaN            NaN            NaN            NaN            NaN          NaN          NaN
NaN          NaN          NaN          NaN          NaN          NaN
max    9246.000000   61.000000    1.000000   7.000000     1.0     NaN      1.0     1.0      NaN      1.
0   54.000000    1.000000    2.000000    1.000000    1.000000    1.000000    1.000000    1.000000    1.
000000    1.000000    1.000000    1.000000    1.000000    1.000000    1.000000   12.000000    1.000000
1.000000    2.000000    2.000000    2.000000    2.000000    1.000000    2.000000    1.000000    1.00000
0    2.000000    2.000000    1.000000   18.000000    4.000000   20.000000  300.000000   20.000000   11.
000000    2.000000   30.000000  180.000000    8.000000  733.000000  160.000000  180.000000    1.000000
4.000000    9.000000   18.000000    6.000000    6.000000    6.000000   10.000000    6.000000  602.00000
0   17.000000    1.000000    1.000000    1.000000    1.000000  800.000000  180.000000    1.000000   11.
000000    6.000000  9999.000000    6.000000    6.000000    6.000000    6.000000  2000.000000    6.00000
0   3.000000    6.000000  9000.000000    6.000000  9999.000000    6.000000    6.000000   13.000000  150
00.000000    6.000000    6.000000   13.000000    4.000000    4.000000    4.000000    4.000000    4.0000
00    4.000000    4.000000    4.000000    4.000000    4.000000    4.000000    4.000000    4.000000    4
.000000    4.000000    4.000000    4.000000    4.000000    4.000000    4.000000    4.000000    4.000000
4.000000    4.000000    4.000000    4.000000    4.000000    4.000000    4.000000    4.000000    4.00000
0    4.000000    4.000000    4.000000    4.000000    4.000000    4.000000    4.000000    4.000000    4.
```

```
000000    4.000000    1.000000    1.000000  300.000000  2500.000000   46.000000   40.000000   17.000000
6.000000   30.000000  999.000000  999.000000   99.000000  999.000000  999.000000   99.000000   10.00000
0    6.000000  999.000000  999.000000   99.000000  999.000000  999.000000   99.000000  999.000000  999.
000000   99.000000  999.000000  999.000000   99.000000  999.000000  999.000000   99.000000  999.000000
999.000000   99.00000  999.000000  999.000000   99.000000  999.000000  999.000000   99.000000  999.0000
00  999.000000   99.000000  999.000000  999.000000   99.000000  999.000000  999.000000   99.000000  999
.000000  999.000000   99.000000    5.000000       0.0  999.000000  999.000000   99.000000   99.000000
70.000000  999.000000  150.000000   55.000000   99.000000    2.000000   66.000000    4.000000    6.0000
00   50.000000    1.000000  999.000000  180.000000   39.000000   50.000000       0.0    6.000000    3.00
0000   23.000000   80.000000   50.000000   15.000000   60.000000    1.000000    5.000000    2.000000  9
99.000000  999.000000   99.000000  999.000000  999.000000   99.000000    8.000000   50.000000    8.0000
00    4.000000    9.000000  999.000000  999.000000   99.000000   4.000000       0.0       0.0    1.000000
9.000000   54.000000   90.000000   12.000000   30.000000    1.000000    1.000000   60.000000   90.00000
0    7.000000       0.0   17.000000  358.000000   24.000000  880.000000   35.000000  600.000000  300.000
000   51.000000  360.000000   10.000000  180.000000  365.000000  180.000000    9.000000    5.000000
1.000000    1.000000    1.000000    1.000000    1.000000    1.000000    1.000000    1.000000    1.00000
0    1.000000    1.000000    1.000000    1.000000    1.000000    1.000000    1.000000    5.000000    1.
000000    1.000000    9.000000    6.000000   10.000000   50.000000    1.000000       0.0    1.000000   7.0
00000    1.000000    7.000000    7.000000   30.000000    1.000000   30.000000    1.000000    8.000000   8.
000000    1.000000   30.000000   30.000000    1.000000    9.000000    1.000000    1.000000    1.000000  3
0.000000   20.000000   16.000000   20.000000   35.000000    8.000000    8.000000   40.000000    9.00000
0   88.000000   30.000000    8.000000    8.000000   29.000000    8.000000    8.000000   30.000000    8.
000000    8.000000   54.000000    8.000000    8.000000   38.000000    9.000000    9.000000   37.000000
8.000000   47.000000    8.000000    8.000000   40.000000    8.000000   40.000000    8.000000    8.00000
0   30.000000    9.000000   47.000000    8.000000    8.000000   44.000000    8.000000   39.000000    8.
000000    8.000000   27.000000    6.000000   50.000000    8.000000    8.000000   41.000000    8.000000
40.000000    8.000000    8.000000   42.000000    8.000000   40.000000   41.000000    2.000000    6.0000
00    9.00000   35.000000    8.000000    8.000000   30.000000   10.000000   35.000000    8.000000    8.
000000   60.000000    6.000000   27.000000    4.000000    8.000000       0.0       0.0   25.000000    1.0
00000       0.0       0.0   45.000000   29.000000    6.000000    6.000000    5.000000    6.000000   27.000
000    8.000000    5.000000    2.000000    4.000000   30.000000   30.000000   40.000000   40.000000    3
0.000000   30.000000   40.000000   30.000000    6.000000    3.000000   48.000000   30.000000    7.000000
8.000000   34.000000   30.000000   43.000000   10.000000   68.000000    2.000000    4.000000    4.000000
4.000000    4.000000    4.000000    4.000000    4.000000    4.000000    4.000000    4.000000    4.00000
0    4.000000    4.000000    4.000000    4.000000    4.000000   25.000000  298.000000   12.000000    4.
```

```
000000    4.000000    4.000000   50.000000   30.000000   11.000000   46.000000   15.000000    2.000000
811.000000    4.000000   45.000000   66.000000    4.000000    9.000000   11.000000   59.000000   60.000
000   50.000000   52.000000   42.000000   10.000000   60.000000   42.000000   11.000000   80.00000   60
.000000   60.000000    3.000000    9.000000    7.000000    1.000000   59.000000    5.000000  999.000000
1.000000    1.000000    8.000000    8.000000    8.000000   20.000000  999.000000  105.000000  210.00000
0  280.000000   60.000000   90.000000  540.000000    4.000000    4.000000    4.000000  180.000000  180.
000000   25.000000  999.000000  999.000000  500.000000  999.000000  999.000000  999.000000  999.000000
999.000000  999.000000  999.000000  999.000000   10.000000   10.000000    1.000000    5.000000    5.000
000    5.000000    5.000000    5.000000    5.000000    5.000000    5.000000    5.000000    5.000000
5.000000    5.000000    5.000000    5.000000    5.000000    5.000000    5.000000    5.000000    5.00000
0    5.000000    5.000000    5.000000    5.000000    5.000000    5.000000    5.000000    5.000000    5.
000000    5.000000    5.000000    5.000000    5.000000    5.000000    5.000000    5.000000    5.000000
5.000000    5.000000    5.000000    5.000000    5.000000    5.000000    5.000000    5.000000    5.00000
0    5.000000    5.000000    5.000000    5.000000    5.000000    5.000000    5.000000    5.000000    5.
000000    5.000000    5.000000    5.000000    5.000000    5.000000    5.000000    5.000000    5.000000
5.000000    5.000000    5.000000    5.000000    5.000000    5.000000    5.000000    5.000000    5.00000
0    5.000000    5.000000    5.000000    5.000000    5.000000    5.000000    5.000000    5.000000    5.
000000    5.000000    5.000000    5.000000    5.000000    5.000000    5.000000    5.000000    5.000000
5.000000    5.000000    5.000000    5.000000    5.000000    5.000000    5.000000        1.000000
1.000000     1.000000         1.000000      1.000000     1.000000     1.000000       1.000000    7.000000
5.000000  276.000000  243.000000  269.000000  269.000000    1.000000    1.000000    1.000000    1.00000
0    1.000000    1.000000    1.000000    1.000000    1.000000     1.000000          0.0     1.000000
1.000000     1.000000          0.0     1.000000     1.000000          0.0         0.0         0.0
1.000000     1.000000     1.000000     1.000000     1.000000     1.000000     1.000000     1.000000    1.0
00000    1.000000    1.000000    1.000000     1.000000    1.000000    1.000000     1.000000    1.000000
1.000000    1.000000         0.0         0.0         0.0    1.000000      1.000000     1.000000    1.00
0000     1.000000     1.000000     1.000000     1.000000     1.000000     1.000000     1.000000
0.0          0.0      1.000000      1.000000      1.000000          0.0      1.000000          0.0
0.0          0.0          0.0      1.000000      1.000000      1.000000      1.000000      1.000000
1.000000    1.000000    9.000000    1.000000    9.000000    1.000000   13.000000    1.000000
```

In [13]:
```
#initial exploratory data analysis
#frequency distributions for categorical variables
#graphical representations
#calculations of center and spread for quantitative variables
```

In [14]:
```
#we select 17 variables of interest
reduced_dataset = full_dataset[['cond', 'csex', 'age', 'clive', 'majsup',  'allarrests', 'anyarrest', 'a
lldrugs', 'anydrugs', 'allcrimes', 'anycrime',
                               'arrest_9mo', 'reincarc_9mo', 'num_arrest', 'num_reincarc', 'violent_char
ge',
                               'property_charge']]
```

In [15]:
```
#we take a look at the first 30 rows
reduced_dataset.head(30)
```

Out[15]:

| | cond | csex | age | clive | majsup | allarrests | anyarrest | alldrugs | anydrugs | allcrimes | anycrime | arrest_9mo | reincarc_9mo | nur |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 1 | 30.0 | 5 | 11.0 | 2.0 | 1.0 | 2.0 | 1.0 | 2.0 | 1.0 | 1 | 1 | 2 |
| **1** | 1 | 1 | 26.0 | 5 | 11.0 | 0.0 | 0.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1 | 1 | 1 |
| **2** | 1 | 1 | 46.0 | 4 | 9.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0 | 0 | 0 |
| **3** | 0 | 1 | 26.0 | 4 | 9.0 | 0.0 | 0.0 | 1.0 | 1.0 | 2.0 | 1.0 | 0 | 0 | 0 |
| **4** | 0 | 1 | 37.0 | 4 | 6.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0 | 0 | 0 |
| **5** | 1 | 1 | 41.0 | 4 | 9.0 | 0.0 | 0.0 | 1.0 | 1.0 | 1.0 | 1.0 | 0 | 0 | 0 |
| **6** | 1 | 1 | 45.0 | 5 | 11.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0 | 0 | 0 |
| **7** | 0 | 1 | 30.0 | 4 | 9.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0 | 0 | 0 |
| **8** | 0 | 1 | 29.0 | 5 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0 | 0 | 0 |
| **9** | 0 | 1 | 35.0 | 4 | 9.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0 | 0 | 0 |

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **10** | 0 | 1 | 35.0 | 4 | 9.0 | NaN | NaN | NaN | NaN | NaN | NaN | 0 | 0 | 0 |
| **11** | 0 | 1 | 37.0 | 4 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0 | 0 | 0 |
| **12** | 0 | 1 | 24.0 | 4 | 9.0 | 0.0 | 0.0 | 1.0 | 1.0 | 0.0 | 0.0 | 0 | 0 | 0 |
| **13** | 1 | 1 | 24.0 | 4 | 9.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0 | 0 | 0 |
| **14** | 1 | 1 | 32.0 | 4 | 9.0 | 0.0 | 0.0 | 1.0 | 1.0 | 0.0 | 0.0 | 0 | 0 | 0 |
| **15** | 0 | 1 | 54.0 | 4 | 9.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0 | 0 | 0 |
| **16** | 1 | 1 | 24.0 | 4 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0 | 0 | 0 |
| **17** | 1 | 1 | 25.0 | 5 | 9.0 | 4.0 | 1.0 | 1.0 | 1.0 | 4.0 | 1.0 | 1 | 1 | 1 |
| **18** | 1 | 1 | 29.0 | 4 | 9.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 1.0 | 1 | 1 | 1 |
| **19** | 1 | 1 | 41.0 | 4 | 1.0 | 0.0 | 0.0 | 1.0 | 1.0 | 1.0 | 1.0 | 0 | 0 | 0 |
| **20** | 1 | 1 | 30.0 | 5 | 5.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0 | 0 | 0 |
| **21** | 0 | 1 | 21.0 | 5 | 9.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0 | 0 | 0 |
| **22** | 1 | 1 | 37.0 | 4 | 1.0 | 0.0 | 0.0 | 3.0 | 1.0 | 1.0 | 1.0 | 1 | 1 | 2 |
| **23** | 1 | 1 | 39.0 | 5 | 1.0 | 0.0 | 0.0 | 1.0 | 1.0 | 0.0 | 0.0 | 0 | 0 | 0 |
| **24** | 0 | 1 | 28.0 | 4 | 9.0 | 0.0 | 0.0 | 1.0 | 1.0 | 0.0 | 0.0 | 0 | 0 | 0 |
| **25** | 0 | 1 | 34.0 | 4 | 9.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0 | 0 | 0 |
| **26** | 1 | 1 | 19.0 | 4 | 9.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 1.0 | 0 | 0 | 0 |
| **27** | 1 | 1 | 41.0 | 4 | 1.0 | 0.0 | 0.0 | 9.0 | 1.0 | 0.0 | 0.0 | 0 | 0 | 0 |
| **28** | 1 | 1 | 24.0 | 5 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0 | 0 | 0 |
| **29** | 0 | 1 | 35.0 | 4 | 1.0 | NaN | NaN | NaN | NaN | NaN | NaN | 0 | 0 | 0 |

In [16]:
```
#we check the length of the dataset
print(len(reduced_dataset))
```

```
476
```

In [17]:
```
#we confirm the number of columns
print(len(reduced_dataset.columns))
```

```
17
```

In [18]:
```
#we rename some of the columns to give them more meaningful names
reduced_dataset.rename(columns = {'csex': 'sex', 'clive': 'living_situation', 'majsup': 'support'}, inpl
ace=True)
```

```
/Users/RI/Anaconda/anaconda/lib/python3.5/site-packages/pandas/core/frame.py:2754: SettingWithCopyWarni
ng:
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/stable/indexing.html#indexin
g-view-versus-copy
  **kwargs)
```

In [19]:
```
#convert all variables to numeric ones
```

In [193]:
```
#mark all variables as numeric data, and signify, for the relevant ones, that they are categorical
#rather than quantitative variables
#errors='coerce' tells pandas to return invalid values as NaN rather than as the input values themselves
reduced_dataset['cond'] = pd.to_numeric(reduced_dataset['cond'], errors='coerce').astype('category')
reduced_dataset['sex'] = pd.to_numeric(reduced_dataset['sex'], errors='coerce').astype('category')
reduced_dataset['age'] = pd.to_numeric(reduced_dataset['age'], errors='coerce')
reduced_dataset['living_situation'] = pd.to_numeric(reduced_dataset['living_situation'], errors='coerce'
).astype('category')
reduced_dataset['support'] = pd.to_numeric(reduced_dataset['support'], errors='coerce').astype('category
')
reduced_dataset['allarrests'] = pd.to_numeric(reduced_dataset['allarrests'], errors='coerce')
reduced_dataset['anyarrest'] = pd.to_numeric(reduced_dataset['anyarrest'], errors='coerce').astype('cate
gory')
reduced_dataset['alldrugs'] = pd.to_numeric(reduced_dataset['alldrugs'], errors='coerce')
reduced_dataset['anydrugs'] = pd.to_numeric(reduced_dataset['anydrugs'], errors='coerce').astype('catego
ry')
reduced_dataset['allcrimes'] = pd.to_numeric(reduced_dataset['allcrimes'], errors='coerce')
reduced_dataset['anycrime'] = pd.to_numeric(reduced_dataset['anycrime'], errors='coerce').astype('catego
ry')
reduced_dataset['arrest_9mo'] = pd.to_numeric(reduced_dataset['arrest_9mo'], errors='coerce').astype('ca
tegory')
reduced_dataset['reincarc_9mo'] = pd.to_numeric(reduced_dataset['reincarc_9mo'], errors='coerce').astype
('category')
reduced_dataset['num_arrest'] = pd.to_numeric(reduced_dataset['num_arrest'], errors='coerce')
reduced_dataset['num_reincarc'] = pd.to_numeric(reduced_dataset['num_reincarc'], errors='coerce')
reduced_dataset['violent_charge'] = pd.to_numeric(reduced_dataset['violent_charge'], errors='coerce').as
type('category')
reduced_dataset['property_charge'] = pd.to_numeric(reduced_dataset['property_charge'], errors='coerce').
astype('category')
```

In [21]:
```
#now let's look at the counts and frequency distributions for these variables
#for this, we will aim to get an idea of the shape, center, and spread of these variables
#we will analyze the shape visually by checking for modality and skewness
#we will check for measure of center such as mean, median and mode
#we will check the spread through the standard deviation
```

In [194]:
```
reduced_dataset['cond'].value_counts(sort=False, normalize = True)
```

Out[194]:
```
0    0.489496
1    0.510504
Name: cond, dtype: float64
```

In [148]:
```
reduced_dataset['cond'].describe()
```

Out[148]:
```
count     476
unique      2
top         1
freq      243
Name: cond, dtype: int64
```

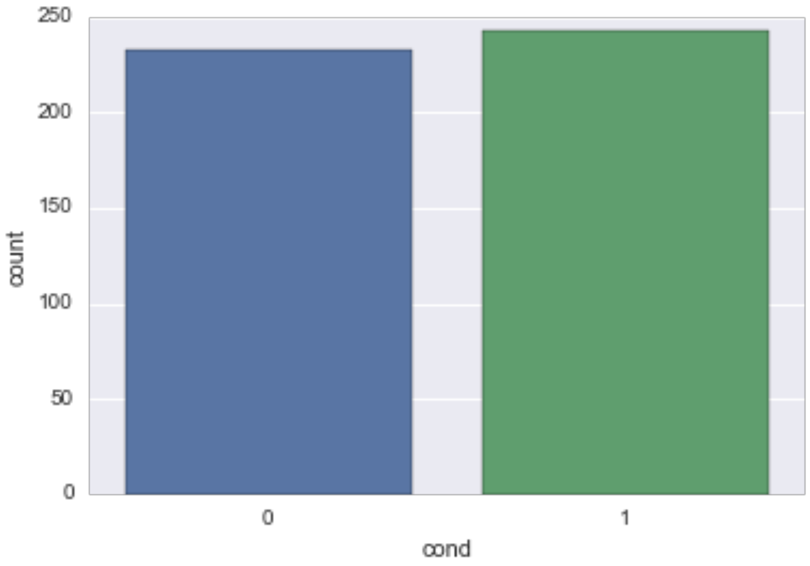
In [24]:
```
#rules for visualizing data:

#for visualizing a variable:
#if it is categorical we use a bar chart i.e. sns's countplot function
#if it is quantitative, we can combine a kernel density estimate and a histogram with sns's
#distplot function

#for visualizing two variables:
# C-C: bivariate bar graph with sns factorplot
# C-Q: bivariate bar graph with sns factorplot
# Q-Q: scatterplot with sns regplot
```

In [25]:
```
#given that the study condition is a categorical variable, we use a count plot to visualize it.
sns.countplot(x='cond', data=reduced_dataset)
```

Out[25]: <matplotlib.axes._subplots.AxesSubplot at 0x1182c91d0>




In [26]:
```
#we check the counts per each value of the variable. sort=False tells pandas not to sort the results
#by values. normalize = True tells it to return the relative frequencies rather than the absolute counts
reduced_dataset['sex'].value_counts(sort=False, normalize=True)
```

Out[26]:
```
1    0.829832
2    0.170168
Name: sex, dtype: float64
```

In [27]:
```
#print all age values as a list to look through them
lis = [x for x in reduced_dataset['age']]
print(lis)
```

```
[30.0, 26.0, 46.0, 26.0, 37.0, 41.0, 45.0, 30.0, 29.0, 35.0, 35.0, 37.0, 24.0, 24.0, 32.0, 54.0, 24.0,
25.0, 29.0, 41.0, 30.0, 21.0, 37.0, 39.0, 28.0, 34.0, 19.0, 41.0, 24.0, 35.0, 27.0, 27.0, 40.0, 35.0, 2
3.0, 36.0, 36.0, 40.0, 24.0, 29.0, 22.0, 34.0, 23.0, 35.0, 36.0, 34.0, 40.0, 23.0, 44.0, 23.0, 28.0, 29
.0, 45.0, 42.0, 41.0, 39.0, 35.0, 30.0, 31.0, 24.0, 26.0, 32.0, 24.0, 25.0, 34.0, 28.0, 39.0, 27.0, 31.
0, 33.0, 27.0, 36.0, 33.0, 31.0, 24.0, 22.0, 33.0, 22.0, 35.0, 33.0, 29.0, 31.0, 45.0, 24.0, 39.0, 38.0
, 44.0, 29.0, 36.0, 33.0, 46.0, 27.0, 22.0, 25.0, 31.0, 22.0, 38.0, 36.0, 35.0, nan, 22.0, nan, 36.0, 3
8.0, 43.0, 36.0, 38.0, 42.0, 44.0, 35.0, 44.0, 24.0, 27.0, 23.0, 41.0, 35.0, 25.0, 34.0, 45.0, 34.0, 24
.0, 35.0, 59.0, 36.0, 30.0, 40.0, 20.0, 35.0, 32.0, 26.0, 27.0, 25.0, 28.0, 27.0, 30.0, 38.0, 41.0, 25.
0, 40.0, 42.0, 34.0, 27.0, 36.0, 29.0, 27.0, 50.0, 43.0, 29.0, 44.0, 42.0, 25.0, 28.0, 41.0, 22.0, 59.0
, 36.0, 32.0, 40.0, 25.0, 21.0, 38.0, 34.0, 40.0, 28.0, 25.0, 37.0, 41.0, 40.0, 24.0, 30.0, 37.0, 29.0,
29.0, 49.0, 37.0, 21.0, 26.0, 40.0, 28.0, 37.0, 38.0, 40.0, 28.0, 42.0, 40.0, 39.0, 24.0, 21.0, 46.0, 3
2.0, 26.0, 24.0, 31.0, 34.0, 26.0, 27.0, 26.0, 26.0, 28.0, 31.0, 50.0, 29.0, 35.0, 21.0, 22.0, 38.0, 29
.0, 42.0, 29.0, 45.0, 28.0, 42.0, 38.0, 23.0, 29.0, 41.0, 39.0, 37.0, 26.0, 22.0, 29.0, 30.0, 26.0, 47.
0, 47.0, 25.0, 32.0, 28.0, 38.0, 29.0, 24.0, 42.0, 22.0, 26.0, 46.0, 34.0, 44.0, 30.0, 23.0, 23.0, 30.0
, 23.0, 20.0, 46.0, 30.0, 24.0, 44.0, 34.0, 40.0, 24.0, 37.0, 37.0, 43.0, 48.0, 44.0, 29.0, 41.0, 22.0,
52.0, 42.0, 28.0, 51.0, 26.0, 43.0, 22.0, 46.0, 47.0, 43.0, 46.0, 23.0, 36.0, 24.0, 25.0, 47.0, 39.0, 3
1.0, 33.0, 35.0, 25.0, 38.0, 36.0, 42.0, 26.0, 44.0, 50.0, 36.0, 32.0, 47.0, 36.0, 29.0, 33.0, 44.0, 27
.0, 29.0, 46.0, 22.0, 37.0, 25.0, 39.0, 47.0, 30.0, 26.0, 40.0, 25.0, 27.0, 45.0, 22.0, 23.0, 28.0, 28.
0, 46.0, 44.0, 35.0, 41.0, 27.0, 43.0, 38.0, 38.0, 30.0, 40.0, 35.0, 33.0, 51.0, 32.0, 41.0, 28.0, 28.0
, 26.0, 31.0, 35.0, 33.0, 33.0, 31.0, 31.0, 43.0, 44.0, 29.0, 35.0, 36.0, 28.0, 56.0, 40.0, 25.0, 39.0,
29.0, 49.0, 22.0, 34.0, 20.0, 46.0, 42.0, 23.0, 24.0, 23.0, 22.0, 36.0, 50.0, 32.0, 24.0, 24.0, 30.0, 2
7.0, 34.0, 30.0, 28.0, 34.0, 24.0, 42.0, 36.0, 52.0, 20.0, 40.0, 37.0, 25.0, 24.0, 40.0, 33.0, 26.0, 25
.0, 27.0, 40.0, 47.0, 27.0, 35.0, 31.0, 51.0, 28.0, 36.0, 28.0, 28.0, 39.0, 40.0, 37.0, 36.0, 38.0, 48.
0, 18.0, 25.0, 32.0, 27.0, 43.0, 44.0, 27.0, 44.0, 19.0, 43.0, 27.0, 34.0, 34.0, 40.0, 34.0, 48.0, 49.0
, 44.0, 43.0, 33.0, 26.0, 32.0, 25.0, 39.0, 43.0, 42.0, 40.0, 44.0, 23.0, 31.0, 42.0, 34.0, 45.0, 57.0,
18.0, 23.0, 21.0, 21.0, 25.0, 29.0, 43.0, 59.0, 25.0, 22.0, 38.0, 41.0, 21.0, 49.0, 34.0, 48.0, 48.0, 4
6.0, 55.0, 50.0, 54.0, 54.0, 61.0, 50.0, 44.0, 21.0, 45.0, 45.0, 43.0, 51.0, 45.0, 19.0, 51.0, 24.0, 37
.0, 18.0, 33.0, 42.0, 42.0, 42.0, 24.0, 37.0, 31.0, 41.0, 39.0, 42.0]
```

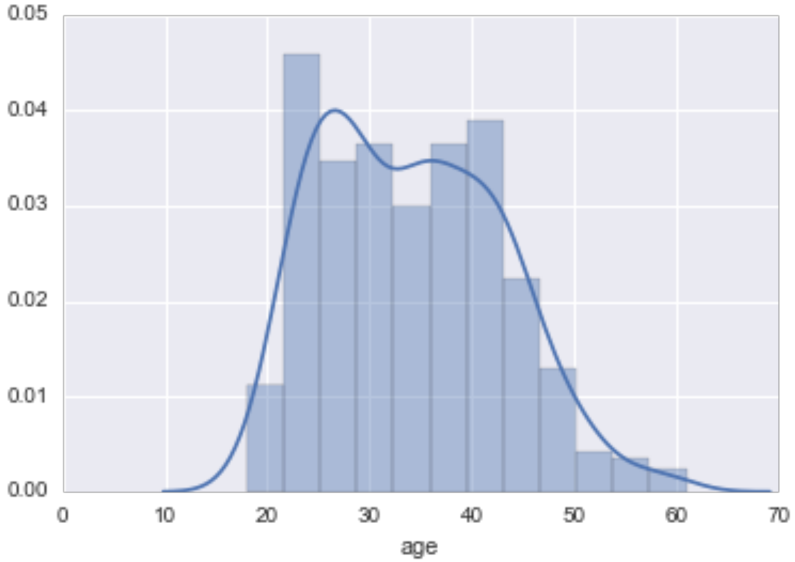
In [28]:
```
#the describe request gives us the count, mean, std, min, max, as well as the quartiles for the
#respective value distribution
reduced_dataset['age'].dropna().describe()
```

Out[28]:
```
count    474.000000
mean      34.118143
std        8.836234
min       18.000000
25%       27.000000
50%       34.000000
75%       41.000000
max       61.000000
Name: age, dtype: float64
```

In [29]:
```
#given that age is a quantitative variable, we use a distplot to visualize it.
sns.distplot(reduced_dataset['age'].dropna())
```

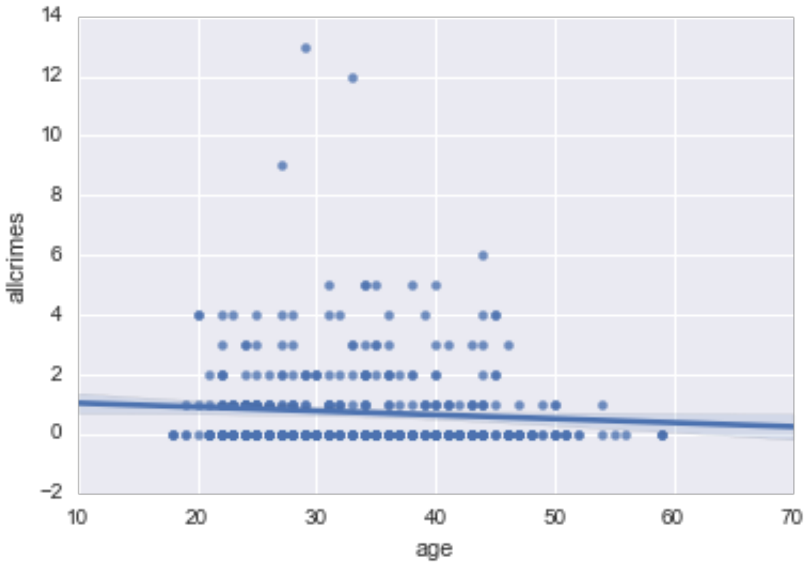
```
/Users/RI/Anaconda/anaconda/lib/python3.5/site-packages/statsmodels/nonparametric/kdetools.py:20: Visib
leDeprecationWarning: using a non-integer number instead of an integer will result in an error in the f
uture
  y = X[:m/2+1] + np.r_[0,X[m/2+1:],0]*1j
```

Out[29]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1182729e8>
```



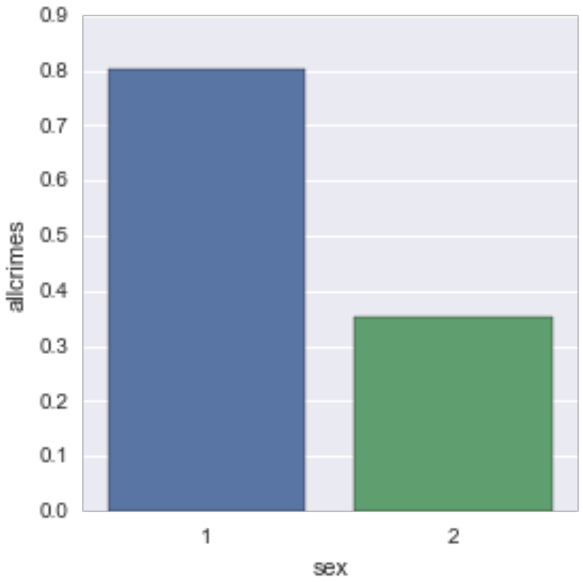
In [32]:
```
#We use a regplot to plot two quantitative variables, age and allcrimes, while also having a regression line suggesting
#any association present
sns.regplot(x='age', y='allcrimes', data=reduced_dataset)
```

Out[32]: <matplotlib.axes._subplots.AxesSubplot at 0x1187067f0>



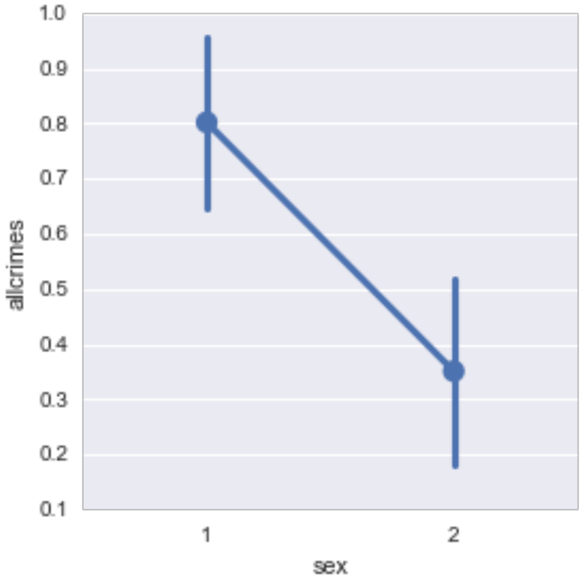
In [33]:
```
#categorical explanatory variable 'sex' and quantitative response variable 'allcrimes'
sns.factorplot(x='sex', y='allcrimes', data=reduced_dataset, kind='bar', ci=None)
```

Out[33]: <seaborn.axisgrid.FacetGrid at 0x11bd14d68>



In [38]:
```
sns.factorplot(x='sex', y='allcrimes', kde='bar', data=reduced_dataset)
```

Out[38]: <seaborn.axisgrid.FacetGrid at 0x11c365978>




In [39]:
```
#before starting to manipulate the dataset itself, we make a copy, and will work on the copy
#rather than the original reduced dataset
rdc = reduced_dataset.copy()
```

In [40]:
```
ct1 = rdc.groupby('anyarrest').size()*100/len(rdc['anyarrest'])
print(ct1)
```

```
anyarrest
0.0    69.327731
1.0    23.319328
dtype: float64
```

In [41]:
```
rdc['anyarrest'].value_counts(sort=False, dropna=False, normalize=True)
```

Out[41]:
```
 0.0    0.693277
 1.0    0.233193
NaN     0.073529
Name: anyarrest, dtype: float64
```

In [42]:
```
rdc['cond'].isnull().value_counts()
```

Out[42]:
```
False    476
Name: cond, dtype: int64
```

In [310]:
```
rdc['anyarrest'].isnull().sum()
```

Out[310]:
```
35
```

In [46]:
```
rdc['anyarrest'] = rdc['anyarrest'].replace(11, np.nan)
```

In [47]:
```
rdc['anyarrest'].value_counts()
```

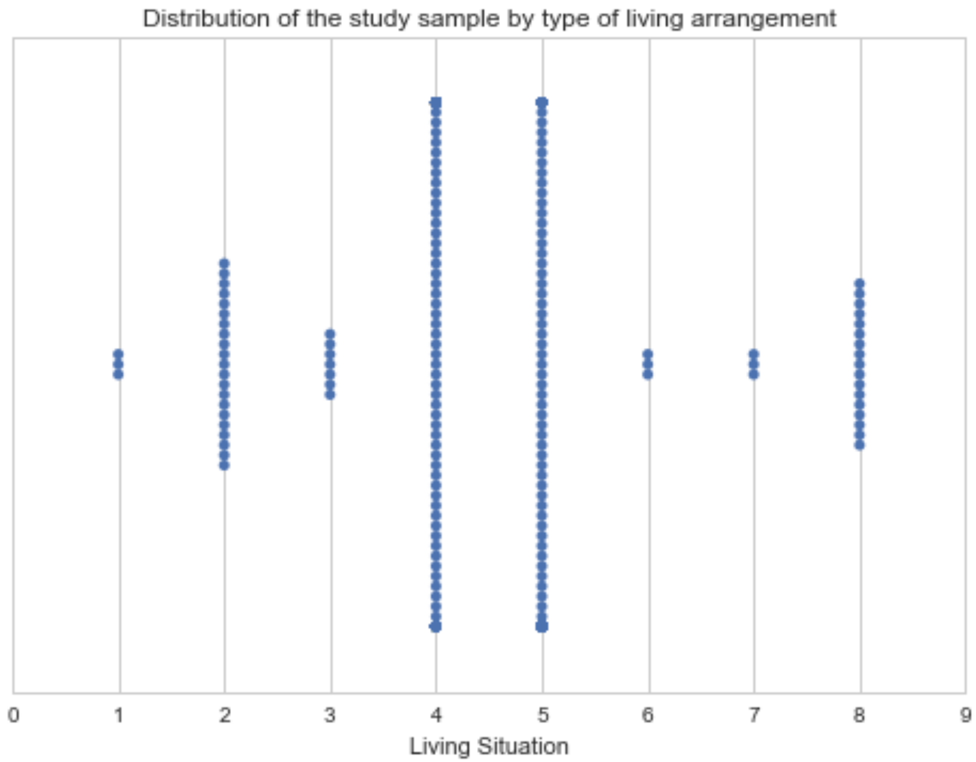
Out[47]:
```
0.0    330
1.0    111
Name: anyarrest, dtype: int64
```

In [48]:
```
living_dic = {1: 1, 2:2, 3:1, 4:1, 5:1, 6:1, 7:1, 8:2}
```

In [49]:
```
rdc['living_situation'] = rdc['living_situation'].map(living_dic)
```

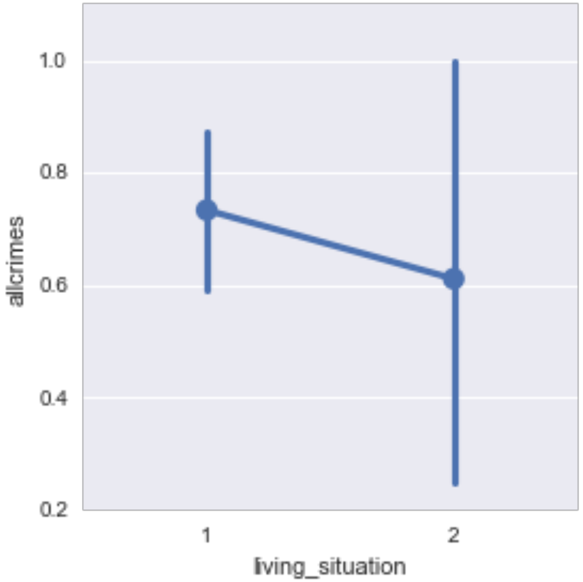
In [152]:
```
sns.swarmplot('living_situation', data=reduced_dataset)
plt.xlabel('Living Situation')
plt.title('Distribution of the study sample by type of living arrangement')
```

Out[152]: <matplotlib.text.Text at 0x11dc74ba8>



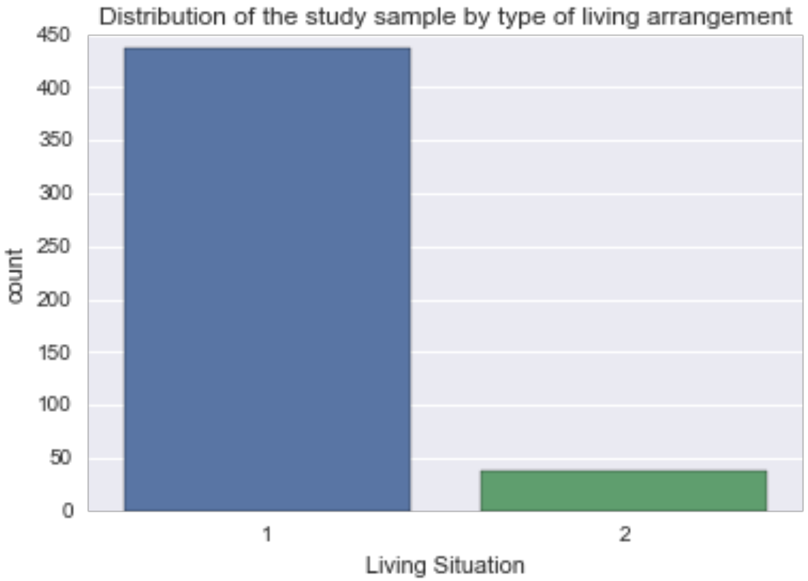
In [142]:
```
sns.factorplot(x='living_situation', y='allcrimes', data=rdc)
```

Out[142]: <seaborn.axisgrid.FacetGrid at 0x11d796c50>



In [149]:
```
sns.countplot('living_situation', data=rdc)
plt.xlabel('Living Situation')
plt.title('Distribution of the study sample by type of living arrangement')
```

Out[149]: <matplotlib.text.Text at 0x11dac9080>




In [140]:
```
rdc['age'].describe()
```

Out[140]:
```
count     473.0
unique      4.0
top        20.0
freq      178.0
Name: age, dtype: float64
```

In [51]:
```
def age_group(x):
    if x < 30:
        return 20
    elif x < 40:
        return 30
    elif x < 50:
        return 40
    elif x < 61:
        return 50
```

In [52]:
```
rdc['age'] = rdc['age'].map(age_group)
```

In [53]:
```
rdc['age']
```

Out[53]:
```
0      30.0
1      20.0
2      40.0
3      20.0
4      30.0
5      40.0
6      40.0
7      30.0
8      20.0
9      30.0
10     30.0
11     30.0
12     20.0
13     20.0
14     30.0
15     50.0
16     20.0
17     20.0
18     20.0
19     40.0
```

```
20     30.0
21     20.0
22     30.0
23     30.0
24     20.0
25     30.0
26     20.0
27     40.0
28     20.0
29     30.0
       ...
446    40.0
447    40.0
448    50.0
449    50.0
450    50.0
451    50.0
452     NaN
453    50.0
454    40.0
455    20.0
456    40.0
457    40.0
458    40.0
459    50.0
460    40.0
461    20.0
462    50.0
463    20.0
464    30.0
465    20.0
466    30.0
467    40.0
468    40.0
469    40.0
```

```
470    20.0
471    30.0
472    30.0
473    40.0
474    30.0
475    40.0
Name: age, dtype: float64
```

In [141]:
```
#describing the dataset/the variables of the dataset, after we group the values in the dataset
#by age category
rdc.groupby('age').describe()
```

```
/Users/RI/Anaconda/anaconda/lib/python3.5/site-packages/numpy/lib/function_base.py:3834: RuntimeWarning
: Invalid value encountered in percentile
  RuntimeWarning)
```

Out[141]:

| | | age_binned | age_unprocessed | allarrests | allcrimes | alldrugs | living_situation | num_arrest | num_reincarc |
|---|---|---|---|---|---|---|---|---|---|
| age | | | | | | | | | |
| 20.0 | count | 0.0 | 178.000000 | 168.000000 | 168.000000 | 168.000000 | 178.000000 | 178.000000 | 178.000000 |
| | mean | NaN | 25.044944 | 0.494048 | 0.767857 | 0.708333 | 1.073034 | 0.786517 | 0.573034 |
| | std | NaN | 2.787827 | 1.110629 | 1.555262 | 0.864151 | 0.260926 | 0.962247 | 0.749955 |
| | min | NaN | 18.000000 | 0.000000 | 0.000000 | 0.000000 | 1.000000 | 0.000000 | 0.000000 |
| | 25% | NaN | 23.000000 | NaN | NaN | NaN | 1.000000 | 0.000000 | 0.000000 |
| | 50% | NaN | 25.000000 | NaN | NaN | NaN | 1.000000 | 1.000000 | 0.000000 |
| | 75% | NaN | 27.000000 | NaN | NaN | NaN | 1.000000 | 1.000000 | 1.000000 |
| | max | NaN | 29.000000 | 9.000000 | 13.000000 | 4.000000 | 2.000000 | 4.000000 | 4.000000 |
| | count | 0.0 | 153.000000 | 142.000000 | 142.000000 | 142.000000 | 153.000000 | 153.000000 | 153.000000 |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **30.0** | **mean** | NaN | 34.568627 | 0.429577 | 0.866197 | 0.767606 | 1.052288 | 0.699346 | 0.483660 |
| | **std** | NaN | 2.747642 | 0.886446 | 1.612139 | 1.089378 | 0.223337 | 1.026523 | 0.708079 |
| | **min** | NaN | 30.000000 | 0.000000 | 0.000000 | 0.000000 | 1.000000 | 0.000000 | 0.000000 |
| | **25%** | NaN | 32.000000 | NaN | NaN | NaN | 1.000000 | 0.000000 | 0.000000 |
| | **50%** | NaN | 35.000000 | NaN | NaN | NaN | 1.000000 | 0.000000 | 0.000000 |
| | **75%** | NaN | 37.000000 | NaN | NaN | NaN | 1.000000 | 1.000000 | 1.000000 |
| | **max** | NaN | 39.000000 | 5.000000 | 12.000000 | 4.000000 | 2.000000 | 7.000000 | 3.000000 |
| **40.0** | **count** | 0.0 | 120.000000 | 110.000000 | 110.000000 | 110.000000 | 120.000000 | 120.000000 | 120.000000 |
| | **mean** | NaN | 43.316667 | 0.245455 | 0.581818 | 0.700000 | 1.125000 | 0.400000 | 0.308333 |
| | **std** | NaN | 2.573241 | 0.706281 | 1.183850 | 1.324089 | 0.332106 | 0.737928 | 0.645551 |
| | **min** | NaN | 40.000000 | 0.000000 | 0.000000 | 0.000000 | 1.000000 | 0.000000 | 0.000000 |
| | **25%** | NaN | 41.000000 | NaN | NaN | NaN | 1.000000 | 0.000000 | 0.000000 |
| | **50%** | NaN | 43.000000 | NaN | NaN | NaN | 1.000000 | 0.000000 | 0.000000 |
| | **75%** | NaN | 45.000000 | NaN | NaN | NaN | 1.000000 | 1.000000 | 1.000000 |
| | **max** | NaN | 49.000000 | 6.000000 | 6.000000 | 9.000000 | 2.000000 | 5.000000 | 5.000000 |
| **50.0** | **count** | 0.0 | 22.000000 | 19.000000 | 19.000000 | 19.000000 | 22.000000 | 22.000000 | 22.000000 |
| | **mean** | NaN | 53.000000 | 0.263158 | 0.157895 | 0.473684 | 1.090909 | 0.181818 | 0.181818 |
| | **std** | NaN | 3.207135 | 0.561951 | 0.374634 | 0.841191 | 0.294245 | 0.394771 | 0.394771 |
| | **min** | NaN | 50.000000 | 0.000000 | 0.000000 | 0.000000 | 1.000000 | 0.000000 | 0.000000 |
| | **25%** | NaN | 50.250000 | NaN | NaN | NaN | 1.000000 | 0.000000 | 0.000000 |
| | **50%** | NaN | 51.500000 | NaN | NaN | NaN | 1.000000 | 0.000000 | 0.000000 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **75%** | NaN | 54.750000 | NaN | NaN | NaN | 1.000000 | 0.000000 | 0.000000 |
| **max** | NaN | 59.000000 | 2.000000 | 1.000000 | 3.000000 | 2.000000 | 1.000000 | 1.000000 |

In [55]:
```
#quartile split
rdc['age_unprocessed'] = reduced_dataset['age']
print('Age - 4 categories - quartiles')
age_binned = rdc['age_binned'] = pd.qcut(rdc['age_unprocessed'], 4, labels = ["1=0%tile", "2=25%tile",
                                                     "3=50%tile", "4=75%tile"]).value_counts(sort=False, dro
pna=True)
print(age_binned)
```

```
Age - 4 categories - quartiles
1=0%tile     135
2=25%tile    115
3=50%tile    116
4=75%tile    108
Name: age_unprocessed, dtype: int64

/Users/RI/Anaconda/anaconda/lib/python3.5/site-packages/pandas/indexes/category.py:118: RuntimeWarning:
Values and categories have different dtypes. Did you mean to use
'Categorical.from_codes(codes, categories)'?
  data = Categorical(data, categories=categories, ordered=ordered)
/Users/RI/Anaconda/anaconda/lib/python3.5/site-packages/pandas/indexes/category.py:118: RuntimeWarning:
None of the categories were found in values. Did you mean to use
'Categorical.from_codes(codes, categories)'?
  data = Categorical(data, categories=categories, ordered=ordered)
```

In [56]:
```
#Hypothesis testing:
#1. specify the null hypothesis and the alternate hypothesis
#2. choose a sample
#3. assess the evidence
#4. draw conclusions
#Definition: assessing evidence provided by the data in favor or against each hypothesis
#about the population

#a result is statistically significant if it is unlikely to have occurred by chance
#p value is also the type 1 error rate: the number of times we would be wrong in rejecting the
#null hypothesis when it is true
#p=0.03: if we reject the null hypothesis, we would be correct 97/100 times.

#Bivariate statistical tools:
#ANOVA; chi-square; correlation coefficient

#ANOVA F Test: are the differences among the sample means due to true differences among the
#population means, or merely due to sampling variability
#F is the variation among samples means divided by the variation within groups
#for explanatory variables with multiples levels, F test and p value do not tell us why the
#group means are not equal, or how. there are many ways in which this can be the case.

#before performing these analyses, one needs to use the .dropna() function to include only
#valid data
```

In [57]:
```
#we will test the null hypothesis that the study condition (subject or control) and number of crimes are not related.
test1_data = rdc[['cond', 'allcrimes']].dropna()
len(test1_data)
```

Out[57]: 441

In [58]:
```
test1 = smf.ols(formula='allcrimes ~ C(cond)', data=test1_data).fit()
print(test1.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:              allcrimes   R-squared:                       0.005
Model:                            OLS   Adj. R-squared:                  0.002
Method:                 Least Squares   F-statistic:                     2.019
Date:                Thu, 09 Mar 2017   Prob (F-statistic):              0.156
Time:                        07:11:26   Log-Likelihood:                -791.21
No. Observations:                 441   AIC:                             1586.
Df Residuals:                     439   BIC:                             1595.
Df Model:                           1
Covariance Type:            nonrobust
================================================================================
                   coef    std err          t      P>|t|      [95.0% Conf. Int.]
--------------------------------------------------------------------------------
Intercept        0.8241      0.099      8.304      0.000         0.629     1.019
C(cond)[T.1]    -0.1974      0.139     -1.421      0.156        -0.470     0.076
==============================================================================
Omnibus:                      382.656   Durbin-Watson:                   1.930
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             8977.014
Skew:                           3.695   Prob(JB):                         0.00
Kurtosis:                      23.831   Cond. No.                         2.64
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

In [59]:
```
#now we examine the means and stds
grouped1_mean = test1_data.groupby('cond').mean()
print(grouped1_mean)
```

```
      allcrimes
cond
0      0.824074
1      0.626667
```

In [60]:
```
grouped1_std = test1_data.groupby('cond').std()
print(grouped1_std)
```

```
      allcrimes
cond
0      1.710861
1      1.166190
```

In [61]:
```
#we repeat the same analysis with arrests
test2_data = rdc[['cond', 'allarrests']].dropna()
test2 = smf.ols(formula = 'allarrests ~ C(cond)', data=test2_data).fit()
print(test2.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:             allarrests   R-squared:                       0.001
Model:                            OLS   Adj. R-squared:                 -0.002
Method:                 Least Squares   F-statistic:                    0.2401
Date:                Thu, 09 Mar 2017   Prob (F-statistic):              0.624
Time:                        07:11:26   Log-Likelihood:                -593.82
No. Observations:                 441   AIC:                             1192.
Df Residuals:                     439   BIC:                             1200.
Df Model:                           1
Covariance Type:            nonrobust
================================================================================
                   coef    std err          t      P>|t|      [95.0% Conf. Int.]
--------------------------------------------------------------------------------
Intercept        0.4213      0.063      6.642      0.000         0.297     0.546
C(cond)[T.1]    -0.0435      0.089     -0.490      0.624        -0.218     0.131
==============================================================================
Omnibus:                      408.823   Durbin-Watson:                   1.861
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            11117.974
Skew:                           4.036   Prob(JB):                         0.00
Kurtosis:                      26.236   Cond. No.                         2.64
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

In [62]:

```
grouped2_mean = test2_data.groupby('cond').mean()
print(grouped2_mean)
```

```
      allarrests
cond
0       0.421296
1       0.377778
```

In [63]:
```
grouped2_std = test2_data.groupby('cond').std()
print(grouped2_std)
```

```
      allarrests
cond
0       1.013082
1       0.847499
```

In [64]:
```
#we now try to use the grouped age variable as well
rdc['age'] = rdc['age'].astype('category')
test3_data = rdc[['age', 'allcrimes']].dropna()
test3 = smf.ols(formula = 'allcrimes ~ C(age)', data=rdc).fit()
print(test3.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:              allcrimes   R-squared:                       0.012
Model:                            OLS   Adj. R-squared:                  0.005
Method:                 Least Squares   F-statistic:                     1.802
Date:                Thu, 09 Mar 2017   Prob (F-statistic):              0.146
Time:                        07:11:26   Log-Likelihood:                -786.67
No. Observations:                 439   AIC:                             1581.
Df Residuals:                     435   BIC:                             1598.
Df Model:                           3
Covariance Type:            nonrobust
==================================================================================
                     coef    std err          t      P>|t|      [95.0% Conf. Int.]
----------------------------------------------------------------------------------
Intercept          0.7679      0.113      6.823      0.000         0.547     0.989
C(age)[T.30.0]     0.0983      0.166      0.591      0.555        -0.228     0.425
C(age)[T.40.0]    -0.1860      0.179     -1.040      0.299        -0.538     0.166
C(age)[T.50.0]    -0.6100      0.353     -1.728      0.085        -1.304     0.084
==============================================================================
Omnibus:                      382.848   Durbin-Watson:                   1.914
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             9225.000
Skew:                           3.709   Prob(JB):                         0.00
Kurtosis:                      24.196   Cond. No.                         5.66
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

In [65]:
```
#given that we have an explanatory categorical variable with multiple levels, we use the
#tuckey hsd test
tuckey1 = multi.MultiComparison(test3_data['allcrimes'], test3_data['age'])
res1 = tuckey1.tukeyhsd()
print(res1.summary())
```

```
Multiple Comparison of Means - Tukey HSD,FWER=0.05
============================================
group1 group2 meandiff  lower  upper  reject
--------------------------------------------
 20.0   30.0   0.0983  -0.3305 0.5272 False 
 20.0   40.0   -0.186  -0.6475 0.2754 False 
 20.0   50.0   -0.61   -1.5206 0.3007 False 
 30.0   40.0  -0.2844  -0.7622 0.1935 False 
 30.0   50.0  -0.7083  -1.6274 0.2108 False 
 40.0   50.0  -0.4239  -1.3586 0.5108 False 
--------------------------------------------
```

In [66]:
```
#We will now test two other hypotheses:
#Hypothesis(0)(a): the study condition (0 or 1) and the committing of a crime are independent
#i.e. there is no relationship between them
#Hypothesis(0)(b): there is no relationship between age and being arrested during the study
#period
```

In [67]:
```
#contingency table of observed counts
#when creating contingency tables, we put the response variable first (therefore vertical in
#table), and the explanatory variable second, therefore horizontal at the top of the table.
ct1 = pd.crosstab(rdc['anycrime'], rdc['cond'])
print(ct1)
```

```
cond        0    1
anycrime          
0.0       139  154
1.0        77   71
```

In [68]:
```
#column percentages
colsum = ct1.sum(axis=0)
colpct = ct1/colsum
print(colpct)
```

```
cond              0         1
anycrime
0.0        0.643519  0.684444
1.0        0.356481  0.315556
```

In [69]:
```
#chi square test
print('chi-square value, p value, expected counts')
cs1 = scipy.stats.chi2_contingency(ct1)
print(cs1)
```

```
chi-square value, p value, expected counts
(0.65446000714878627, 0.41852260938554675, 1, array([[ 143.51020408,  149.48979592],
       [  72.48979592,   75.51020408]]))
```

In [70]:
```
#now the second hypothesis test
rdc['age'] = rdc['age'].astype('category')
rdc['anyarrest'] = rdc['anyarrest'].astype('category')
```

In [71]:
```
#contingency table of observed counts
ct2 = pd.crosstab(rdc['anyarrest'], rdc['age'])
print(ct2)
```

```
age         20.0  30.0  40.0  50.0
anyarrest
0.0          118   105    90    15
1.0           50    37    20     4
```

In [72]:
```
#column percentages
colsum2 = ct2.sum(axis=0)
colpct2 = ct2/colsum2
print(colpct2)
```

```
age              20.0      30.0      40.0      50.0
anyarrest
0.0         0.702381  0.739437  0.818182  0.789474
1.0         0.297619  0.260563  0.181818  0.210526
```

In [73]:
```
#chi square test
print('chi-square value, p value, expected counts')
cs2 = scipy.stats.chi2_contingency(ct2)
print(cs2)
```

```
chi-square value, p value, expected counts
(4.9451035874289762, 0.17586134722188973, 3, array([[ 125.52164009,  106.09567198,   82.18678815,   14.
19589977],
       [  42.47835991,   35.90432802,   27.81321185,    4.80410023]]))
```

In [74]:
```
#We therefore cannot reject the null hypothesis, but given that the explanatory variable has several lev
els,
#we cannot know why the null hypothesis was not rejected
#we therefore perform a 2 by 2 comparison
```

In [75]:
```
sub1 = rdc.copy()
recode1 = {20:20, 30:30}
sub1['comp1v'] = sub1['age'].map(recode1)
```

In [153]:
```
ct3 = pd.crosstab(sub1['cond'], sub1['comp1v'])
print(ct3)
```

```
comp1v  20.0  30.0
cond
0         95    71
1         83    82
```

In [ ]:
```
#column percentages
colsum3 = ct3.sum(axis=0)
colpct3 = ct3 / colsum3
print(colpct3)
```

In [ ]:
```
#chi square test
print('chi square value, p value, expected values')
cs3 = scipy.stats.chi2_contingency(ct3)
print(cs3)
```

In [ ]:
```
recode2 = {20:20, 40:40}
sub1['comp2v'] = sub1['age'].map(recode2)
```

In [ ]:
```
ct4 = pd.crosstab(sub1['anyarrest'], sub1['comp2v'])
print(ct4)
```

In [ ]:
```
colsum4 = ct4.sum(axis=0)
colpct4 = ct4/colsum4
print(colpct4)
```

In [ ]:
```
print('chi square value, p value, expected values')
cs4 = scipy.stats.chi2_contingency(ct4)
print(cs4)
```

In [ ]:
```
recode3 = {20:20, 50:50}
sub1['compv3'] = sub1['age'].map(recode3)
```

In [ ]:
```
ct5 = pd.crosstab(sub1['anyarrest'], sub1['compv3'])
print(ct5)
```

In [ ]:
```
colsum5 = ct5.sum(axis=0)
colpct5 = ct5/colsum5
print(colpct5)
```

In [ ]:
```
print('chi square value, p value, expected values')
cs5 = scipy.stats.chi2_contingency(ct5)
print(cs5)
```

In [ ]:
```
recode4 = {30:30, 40:40}
sub1['compv4'] = sub1['age'].map(recode4)
```

In [ ]:
```
ct6 = pd.crosstab(sub1['anyarrest'], sub1['compv4'])
print(ct6)
```

In [ ]:
```
colsum6 = ct6.sum(axis=0)
colpct6 = ct6 / colsum6
print(colpct6)
```

In [ ]:
```
print('chi square value, p value, expected values')
cs6 = scipy.stats.chi2_contingency(ct6)
print(cs6)
```

In [ ]:
```
recode6 = {30:30, 50:50}
sub1['compv6'] = sub1['age'].map(recode6)
```

In [ ]:
```
ct7 = pd.crosstab(sub1['anyarrest'], sub1['compv6'])
print(ct7)
```

In [ ]:
```
colsum7 = ct7.sum(axis=0)
colpct7 = ct7/colsum7
print(colpct7)
```

In [ ]:
```
print('chi square value, p value, expected values')
cs7 = scipy.stats.chi2_contingency(ct7)
print(cs7)
```

In [ ]:
```
recode7 = {40:40, 50:50}
sub1['compv7'] = sub1['age'].map(recode7)
```

In [ ]:
```
ct8 = pd.crosstab(sub1['anyarrest'], sub1['compv7'])
print(ct8)
```

In [ ]:
```
colsum8 = ct8.sum(axis=0)
colpct8 = ct8 / colsum8
print(colpct8)
```

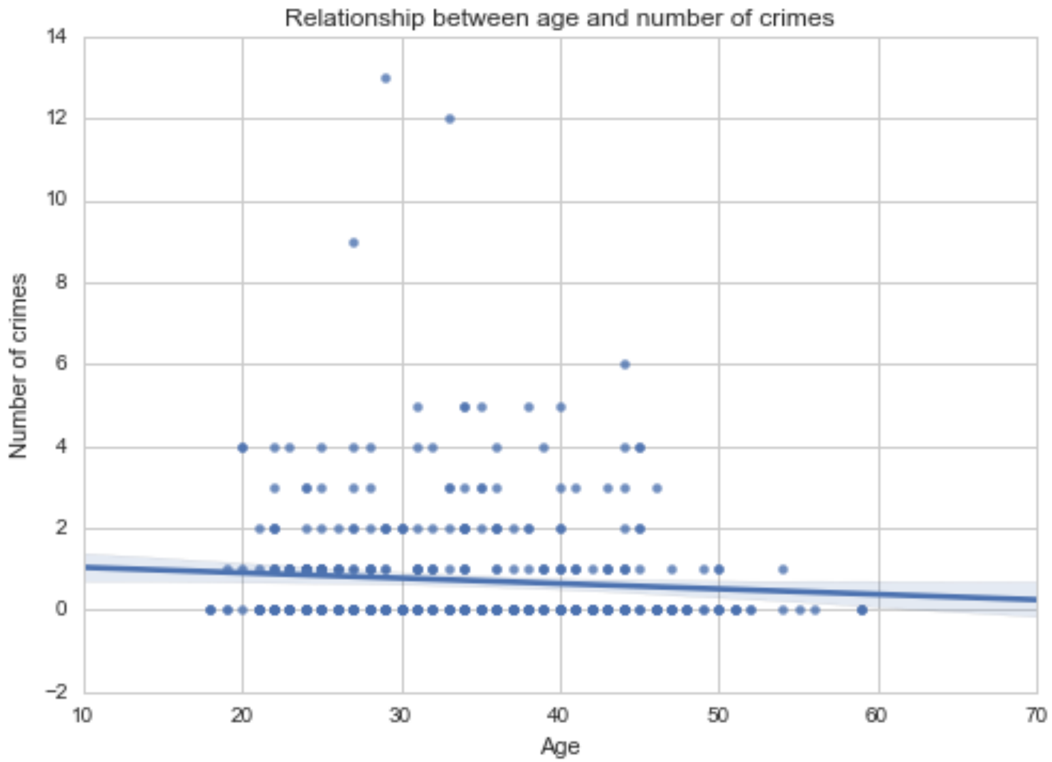
In [ ]:
```
print('chi square value, p value, expected values')
cs8 = scipy.stats.chi2_contingency(ct8)
print(cs8)
```

In [ ]:
```
#now we will test whether there is a relationship between two quantitative variables, age_unprocessed an
d allcrimes
#for this we use the pearson correlation test
#r, going from -1 to 1 only tells us whether the two variables are linearly related. they may be related
in nonlinear ways
#therefore it's always important to look at r in parallel with a scatterplot of the two variables
#r squared is a measure of how much variability in one variable can be explained by the other variable
#to calculate the pearson coefficient we need to remove all missing values
```

In [ ]:
```
rdc.columns
```

In [154]:
```
scat1 = sns.regplot(x='age_unprocessed', y='allcrimes', fit_reg=True, data=rdc)
plt.xlabel('Age')
plt.ylabel('Number of crimes')
plt.title('Relationship between age and number of crimes')
scat1
```

Out[154]: <matplotlib.axes._subplots.AxesSubplot at 0x11dc61860>




In [ ]:
```
data_pearson_test = rdc[['age_unprocessed', 'allcrimes']].dropna()
```

In [ ]:
```
print('association between age and number of crimes')
print(scipy.stats.pearsonr(data_pearson_test['age_unprocessed'],
                                            data_pearson_test['allcrimes']))
```

In [ ]:
```
#a moderator is a third variable that affects the direction and/or strength between your explanatory and
response variables
#the question is, is our response variable associated with our explanatory variable, for each level of o
ur third variable?
```

In [ ]:
```
#let's see if gender is a moderator variable for test group -> allcrimes
```

In [ ]:
```
sub11 = rdc[['cond', 'sex', 'allcrimes']].dropna()
```

In [ ]:
```
rdc['sex'].value_counts()
```

In [ ]:
```
sub12 = sub11[sub11['sex']==1]
sub13 = sub11[sub11['sex']==2]
```

In [ ]:
```
model_12 = smf.ols(formula='allcrimes ~ C(cond)', data=sub12).fit()
print(model_12.summary())
```

In [ ]:
```
model_13 = smf.ols(formula='allcrimes ~ C(cond)', data=sub13).fit()
print(model_13.summary())
```

In [ ]:
```
sub12.groupby('cond').mean()
```

In [ ]:
```
sub12.groupby('cond').std()
```

In [ ]:
```
sub13.groupby('cond').mean()
```

In [ ]:
```
sub13.groupby('cond').std()
```

In [ ]:
```
sns.factorplot(x='cond', y='allcrimes', kind='bar', data=sub12)
```

In [ ]:
```
sns.factorplot(x='cond', y='allcrimes', kind='bar', data=sub13)
```

In [ ]:
```
#we would test for moderator variables with the chi square test the same way
#divide the population into the sublevels of the third variables
#conduct a chi square test for each to see if the relationship is statistically significant for each lev
el
#we would visualize it with a linegraph factorplot(kind='point')
```

In [ ]:
```
#to know if your data is obersevational or experimental, you ask if the explanatory variable was manipul
ated or not
#data reporting tells you what is happening, but data analysis tells you why it is happening
```

In [ ]:
```
#randomization works best as your sample size approaches infinity. for small sizes, imbalances in the gr
oups
#can occur. if you check randomized studies, one of first steps is to check for imbalances between group
s
#on covariates. this is also why we can conclude that variables are associated, but hardly that one caus
es the other
#statistical control: include unbalanced covariates as additional explanatory variables in the study.
#In a true experiment, 3 conditions:
#1. only one variable is manipulated
#2. we have a control group
#3. random assignment
#In theory, in this case one can determine causality
#Quasi experiment:
#1. only one variable is manipulated
#2. control group
#3. no random assignment; groups pre selected. i.e. drug users study.
#To improve a quasi experimental design: add confounding variables; have a control group; use a pre-test
/post-test design
#confounder=control variable=covariate=third variable=lurking variable
```

In [ ]:
```
#identifying a confounding variable does not allow to establish causation, just to get closer to a causa
l connection.
#due to infinite number of possible lurking variables, observational studies cannot rly establish causat
ion
#a lurking of confounding variable is a third variable that is associated with both the explanatory and
response
#variables.
#i.e. x=firefighters; y=damage caused by a fire. plot would suggest more firefighters cuases more fire d
amage.
#in reality there is a third lurking variable that influences both, seriousness of the fire.
#In a study we want to demonstrate that our statistical relationship is valid even after controlling for
confounders.
```

In [ ]:
```
#Linear regression:
#multivariate linear regression for quantitative response variable
#logistic regression for binary categorical response variable
#Assumptions:
#Normality: residuals from our linear regression model are normally distributed. if they are not,
#our model may be misspecified.
#Linearity: association between explanatory and response variable is linear
#Homoscedasticity (or assumption of constant variance): variability in the response variable is the same
at all levels
#of the explanatory variable. i.e. if spread of residuals values increases as you move along x axis, ass
umption is
#false.
#Independence: observations are not correlated with each other. Longitudinal data can violate this assum
ption, as well
#as hierarchical nesting/clustering data i.e. looking at students by classes. this assumption is the mos
t serious
#to be violated, and also cannot be fixed by modifying the variables. the data structure itself is the p
roblem.
#We have to contend with:
#Multicollinearity: explanatory variables are highly correlated with each other. this can mess up your p
arameter estimates
#or make them highly unstable. Signs: 1. highly associated variable not significant. 2. negative regress
ion coefficient
#that should be positive. 3. taking out an explanatory variable drastically changes the results.
#Outliers: can affect your regression line, meaning it will not fit the data as well as it should.
```

In [ ]:
```
#multiple regression model allows us to find the relationship between one explanatory variable and the
#reponse variable, while controlling (holding constant at 0) all the other variables.
#categorical sex (1 and 2); age restricted to 18-25 group: each variable needs to include a meaningful
#value of 0, so as to make it easier to interpret the coefficients
#for a categorical variable, we can just recode one of the values to be 0
#for a quantitative variable, we have to center it. Centering = subtracting the mean of a variable
#from the value of the variable. We are therefore recoding it so that its mean=0.
#Note: do not center the response variable.
```

In [ ]:
```
#We will create a multiple regression model, investigating the relationship between the study group 'con
d', 'age', 'sex',
#and the quantitative response variable 'allcrimes'. We will then do the same for 'allarrests'.
#we will first center the explanatory variables. for categorical variables, one of the categories needs
to be 0, for
#quantitative variables, we need to subtract the mean from each value.
```

In [81]:
```
rdc_linear = rdc[['cond', 'age_unprocessed', 'sex', 'allcrimes', 'allarrests']].dropna()
len(rdc_linear)
```

Out[81]:
```
439
```

In [83]:
```
rdc_linear['age_unprocessed_c'] = rdc['age_unprocessed']-rdc['age_unprocessed'].mean()
print(rdc_linear['age_unprocessed_c'].mean())
```

```
-0.24115029362859228
```

In [84]:
```
rdc_linear['sex'].value_counts()
```

Out[84]:
```
1    362
2     77
Name: sex, dtype: int64
```

In [85]:
```
recode20 = {1:1, 2:0}
rdc_linear['sex_c'] = rdc_linear['sex'].map(recode20)
```

In [92]:
```
model20 = smf.ols(formula = 'allcrimes ~ C(cond)', data=rdc_linear).fit()
print(model20.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:              allcrimes   R-squared:                       0.004
Model:                            OLS   Adj. R-squared:                  0.002
Method:                 Least Squares   F-statistic:                     1.890
Date:                Thu, 09 Mar 2017   Prob (F-statistic):              0.170
Time:                        07:20:13   Log-Likelihood:                -788.43
No. Observations:                 439   AIC:                             1581.
Df Residuals:                     437   BIC:                             1589.
Df Model:                           1
Covariance Type:            nonrobust
================================================================================
                   coef    std err          t      P>|t|      [95.0% Conf. Int.]
--------------------------------------------------------------------------------
Intercept        0.8241      0.099      8.288      0.000         0.629     1.019
C(cond)[T.1]    -0.1918      0.140     -1.375      0.170        -0.466     0.082
==============================================================================
Omnibus:                      380.446   Durbin-Watson:                   1.931
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             8874.157
Skew:                           3.688   Prob(JB):                         0.00
Kurtosis:                      23.754   Cond. No.                         2.64
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

In [93]:
```
model21 = smf.ols(formula = 'allcrimes ~ C(cond)+age_unprocessed_c', data=rdc_linear).fit()
print(model21.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:              allcrimes   R-squared:                       0.010
Model:                            OLS   Adj. R-squared:                  0.006
Method:                 Least Squares   F-statistic:                     2.260
Date:                Thu, 09 Mar 2017   Prob (F-statistic):              0.106
Time:                        07:20:16   Log-Likelihood:                -787.12
No. Observations:                 439   AIC:                             1580.
Df Residuals:                     436   BIC:                             1592.
Df Model:                           2
Covariance Type:            nonrobust
=====================================================================================
                        coef    std err          t      P>|t|      [95.0% Conf. Int.]
-------------------------------------------------------------------------------------
Intercept             0.8186      0.099      8.243      0.000         0.623     1.014
C(cond)[T.1]         -0.1871      0.139     -1.344      0.180        -0.461     0.087
age_unprocessed_c    -0.0129      0.008     -1.619      0.106        -0.029     0.003
==============================================================================
Omnibus:                      380.213   Durbin-Watson:                   1.935
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             8859.859
Skew:                           3.685   Prob(JB):                         0.00
Kurtosis:                      23.738   Cond. No.                         20.0
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

In [94]:
```
model22 = smf.ols(formula = 'allcrimes ~ C(cond)+age_unprocessed_c+C(sex)', data=rdc_linear).fit()
print(model22.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:              allcrimes   R-squared:                       0.023
Model:                            OLS   Adj. R-squared:                  0.017
Method:                 Least Squares   F-statistic:                     3.471
Date:                Thu, 09 Mar 2017   Prob (F-statistic):             0.0162
Time:                        07:20:38   Log-Likelihood:                -784.19
No. Observations:                 439   AIC:                             1576.
Df Residuals:                     435   BIC:                             1593.
Df Model:                           3
Covariance Type:            nonrobust
=====================================================================================
                        coef    std err          t      P>|t|      [95.0% Conf. Int.]
-------------------------------------------------------------------------------------
Intercept             0.8988      0.104      8.627      0.000         0.694     1.104
C(cond)[T.1]         -0.1921      0.139     -1.387      0.166        -0.464     0.080
C(sex)[T.2]          -0.4412      0.182     -2.417      0.016        -0.800    -0.082
age_unprocessed_c    -0.0116      0.008     -1.459      0.145        -0.027     0.004
==============================================================================
Omnibus:                      377.881   Durbin-Watson:                   1.975
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             8785.690
Skew:                           3.650   Prob(JB):                         0.00
Kurtosis:                      23.665   Cond. No.                         23.7
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

In [95]:
```
model23 = smf.ols(formula = 'allcrimes ~ C(cond)+age_unprocessed_c + I(age_unprocessed_c**2)+C(sex)', da
ta=rdc_linear).fit()
print(model23.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:              allcrimes   R-squared:                       0.033
Model:                            OLS   Adj. R-squared:                  0.024
Method:                 Least Squares   F-statistic:                     3.673
Date:                Thu, 09 Mar 2017   Prob (F-statistic):            0.00590
Time:                        07:21:30   Log-Likelihood:                -782.07
No. Observations:                 439   AIC:                             1574.
Df Residuals:                     434   BIC:                             1595.
Df Model:                           4
Covariance Type:            nonrobust
=============================================================================================
                              coef    std err          t      P>|t|      [95.0% Conf. Int.]
---------------------------------------------------------------------------------------------
Intercept                   1.0387      0.124      8.362      0.000         0.795     1.283
C(cond)[T.1]               -0.2112      0.138     -1.527      0.128        -0.483     0.061
C(sex)[T.2]                -0.4521      0.182     -2.485      0.013        -0.810    -0.095
age_unprocessed_c          -0.0072      0.008     -0.876      0.382        -0.023     0.009
I(age_unprocessed_c ** 2)  -0.0017      0.001     -2.050      0.041        -0.003  -6.9e-05
==============================================================================
Omnibus:                      374.130   Durbin-Watson:                   1.982
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             8525.279
Skew:                           3.602   Prob(JB):                         0.00
Kurtosis:                      23.351   Cond. No.                         320.
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

In [97]:
```
model24 = smf.ols(formula = 'allarrests ~ C(cond)', data=rdc_linear).fit()
print(model24.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:             allarrests   R-squared:                       0.000
Model:                            OLS   Adj. R-squared:                 -0.002
Method:                 Least Squares   F-statistic:                    0.2025
Date:                Sat, 11 Mar 2017   Prob (F-statistic):              0.653
Time:                        17:20:34   Log-Likelihood:                -591.96
No. Observations:                 439   AIC:                             1188.
Df Residuals:                     437   BIC:                             1196.
Df Model:                           1
Covariance Type:            nonrobust
================================================================================
                   coef    std err          t      P>|t|      [95.0% Conf. Int.]
--------------------------------------------------------------------------------
Intercept        0.4213      0.064      6.629      0.000         0.296     0.546
C(cond)[T.1]    -0.0401      0.089     -0.450      0.653        -0.215     0.135
==============================================================================
Omnibus:                      406.395   Durbin-Watson:                   1.862
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            10979.833
Skew:                           4.027   Prob(JB):                         0.00
Kurtosis:                      26.138   Cond. No.                         2.64
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

In [99]:
```
model25 = smf.ols(formula = 'allarrests ~ C(cond)+age_unprocessed_c', data=rdc_linear).fit()
print(model25.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:             allarrests   R-squared:                       0.016
Model:                            OLS   Adj. R-squared:                  0.011
Method:                 Least Squares   F-statistic:                     3.519
Date:                Sat, 11 Mar 2017   Prob (F-statistic):             0.0305
Time:                        17:21:10   Log-Likelihood:                -588.55
No. Observations:                 439   AIC:                             1183.
Df Residuals:                     436   BIC:                             1195.
Df Model:                           2
Covariance Type:            nonrobust
=====================================================================================
                        coef    std err          t      P>|t|      [95.0% Conf. Int.]
-------------------------------------------------------------------------------------
Intercept             0.4157      0.063      6.580      0.000         0.292     0.540
C(cond)[T.1]         -0.0354      0.089     -0.399      0.690        -0.209     0.139
age_unprocessed_c    -0.0133      0.005     -2.614      0.009        -0.023    -0.003
==============================================================================
Omnibus:                      404.546   Durbin-Watson:                   1.877
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            10991.230
Skew:                           3.995   Prob(JB):                         0.00
Kurtosis:                      26.175   Cond. No.                         20.0
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

In [103]:
```
model26 = smf.ols(formula = 'allarrests ~ C(cond)+age_unprocessed_c+C(sex)', data=rdc_linear).fit()
print(model26.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:             allarrests   R-squared:                       0.021
Model:                            OLS   Adj. R-squared:                  0.015
Method:                 Least Squares   F-statistic:                     3.179
Date:                Sat, 11 Mar 2017   Prob (F-statistic):             0.0239
Time:                        17:22:42   Log-Likelihood:                -587.30
No. Observations:                 439   AIC:                             1183.
Df Residuals:                     435   BIC:                             1199.
Df Model:                           3
Covariance Type:            nonrobust
=====================================================================================
                        coef    std err          t      P>|t|      [95.0% Conf. Int.]
-------------------------------------------------------------------------------------
Intercept             0.4490      0.067      6.749      0.000         0.318     0.580
C(cond)[T.1]         -0.0374      0.088     -0.423      0.673        -0.211     0.136
C(sex)[T.2]          -0.1834      0.117     -1.574      0.116        -0.412     0.046
age_unprocessed_c    -0.0127      0.005     -2.504      0.013        -0.023    -0.003
==============================================================================
Omnibus:                      402.755   Durbin-Watson:                   1.884
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            10846.200
Skew:                           3.970   Prob(JB):                         0.00
Kurtosis:                      26.020   Cond. No.                         23.7
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

In [102]:
```
model27 = smf.ols(formula = 'allarrests ~ C(cond)+age_unprocessed_c + I(age_unprocessed_c**2)+C(sex)', d
ata=rdc_linear).fit()
print(model27.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:             allarrests   R-squared:                       0.022
Model:                            OLS   Adj. R-squared:                  0.013
Method:                 Least Squares   F-statistic:                     2.409
Date:                Sat, 11 Mar 2017   Prob (F-statistic):             0.0487
Time:                        17:22:34   Log-Likelihood:                -587.24
No. Observations:                 439   AIC:                             1184.
Df Residuals:                     434   BIC:                             1205.
Df Model:                           4
Covariance Type:            nonrobust
=============================================================================================
                              coef    std err          t      P>|t|      [95.0% Conf. Int.]
---------------------------------------------------------------------------------------------
Intercept                   0.4640      0.080      5.822      0.000         0.307     0.621
C(cond)[T.1]               -0.0395      0.089     -0.445      0.657        -0.214     0.135
C(sex)[T.2]                -0.1846      0.117     -1.582      0.114        -0.414     0.045
age_unprocessed_c          -0.0123      0.005     -2.325      0.021        -0.023    -0.002
I(age_unprocessed_c ** 2)  -0.0002      0.001     -0.342      0.733        -0.001     0.001
==============================================================================
Omnibus:                      402.301   Durbin-Watson:                   1.880
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            10785.618
Skew:                           3.965   Prob(JB):                         0.00
Kurtosis:                      25.951   Cond. No.                         320.
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

In [105]:
```
#group means and sd
print('Mean')
ds1 = rdc_linear.groupby('cond').mean()
print(ds1)
```

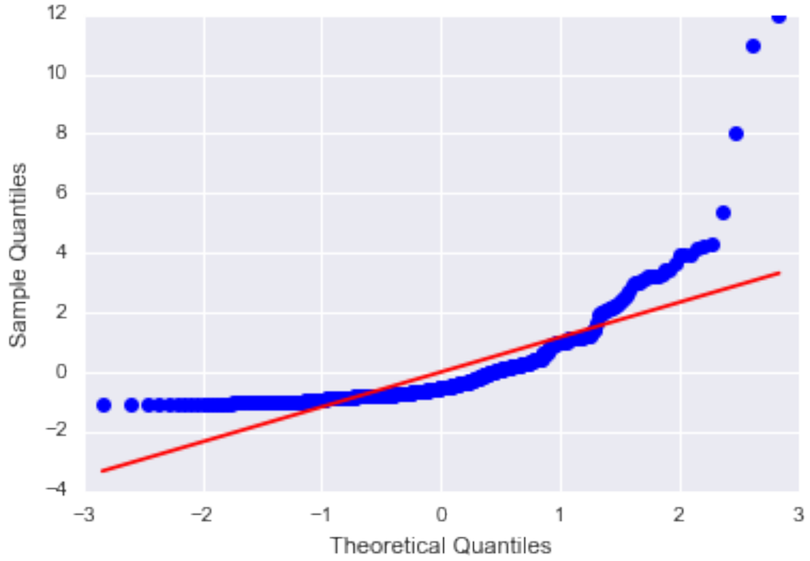
```
Mean
      age_unprocessed  allcrimes  allarrests  age_unprocessed_c     sex_c
cond
0           33.694444   0.824074    0.421296          -0.423699  0.819444
1           34.053812   0.632287    0.381166          -0.064332  0.829596
```

In [106]:
```
print('Standard Deviation')
ds2 = rdc_linear.groupby('cond').std()
print(ds2)
```

```
Standard Deviation
      age_unprocessed  allcrimes  allarrests  age_unprocessed_c     sex_c
cond
0            9.026533   1.710861    1.013082           9.026533  0.385543
1            8.438798   1.169907    0.850545           8.438798  0.376833
```

In [ ]:
```
#For each response variable, allcrimes and allarrests, we choose the model that gave us the highest over
all
#explanatory power, and run further tests to check for evidence of model misspecification.
#If model is correctly specified, residuals are not correlated with explanatory variables.
#If data fails to meet regression assumptions, or misses explanatory variables, we have model misspecifi
cation.
#Intercept = value of response variable when all explanatory variables are held constant at 0
```

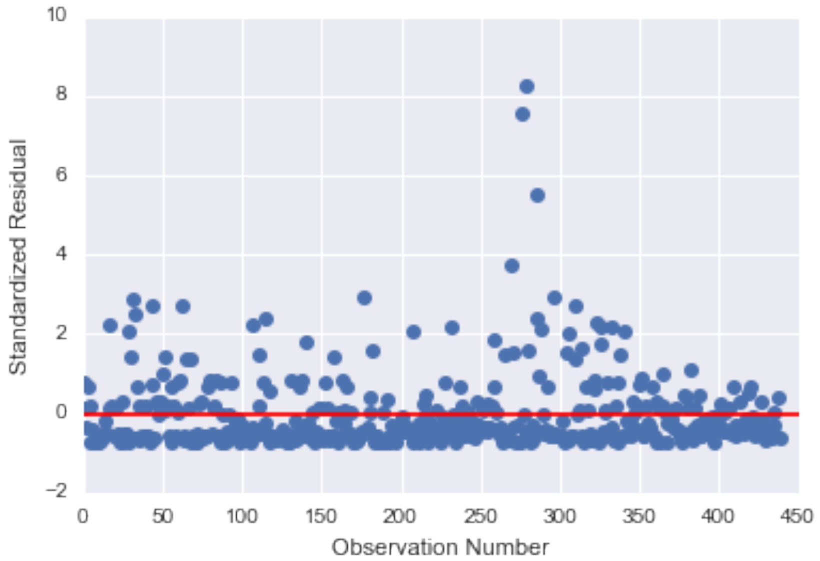
In [110]:
```
#Q-Q plot for normality
fig4 = sm.qqplot(model23.resid, line='r')
#red line represents residuals we would expect if model residuals were normally distributed
#our residuals below deviate significantly from red line, especially at lower and higher quantiles, mean
ing they do not
#follow a normal distribution. This means curvilinear association we saw is not fully explained by our m
odel. We could add
#more explanatory variables.
```



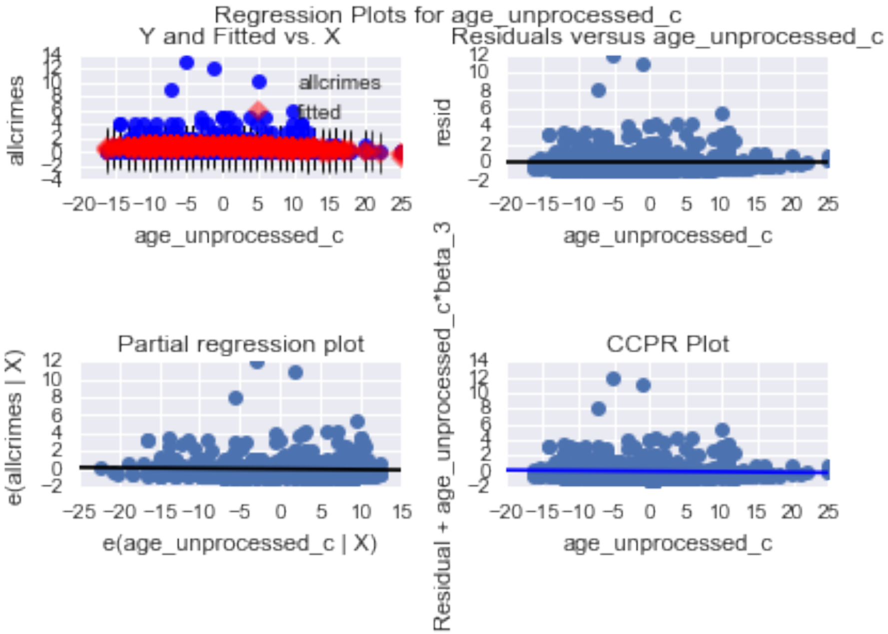

In [ ]:
```
#normalizing or standardizing values makes them fit a standard normal distribution
```

In [111]:
```
#simple plot of residuals
stdres = pd.DataFrame(model23.resid_pearson)
plt.plot(stdres, 'o', ls='None')
l = plt.axhline(y=0, color='r')
plt.ylabel('Standardized Residual')
plt.xlabel('Observation Number')
#resid_pearson normalizes our model's residuals
#ls='none' means points will not be connected
#we expect most residuals to fall within 2sd of the mean. More than 2 are outliers, and more than 3 extr
eme outliers.
#if more than 1% of our observations have standardized residuals with an absolute value greater than 2.5
, or more than 5%
#have one greater than or equal to 2, there is evidence that the fit of the model is poor. largest cause
of this is ommission
#of important explanatory variables in our model.
```

Out[111]: <matplotlib.text.Text at 0x11d0630b8>

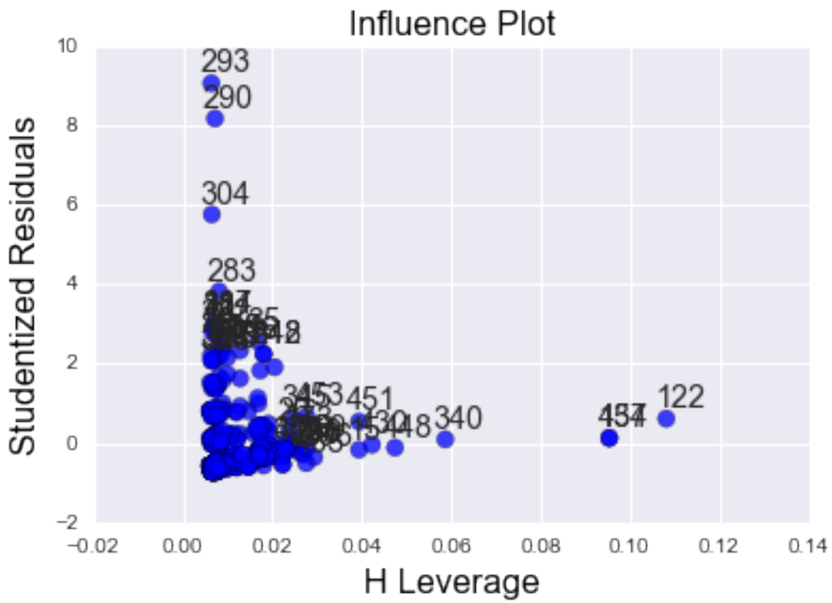


In [113]:
```
#additional regression diagnostic plots
#fig1 = plt.figure(figsize(12,8))
fig1 = sm.graphics.plot_regress_exog(model23, 'age_unprocessed_c')
```



In [114]:
```
#leverage plot
fig3 = sm.graphics.influence_plot(model23, size=8)
print(fig3)
#we see that we have extreme outliers, but they are low leverage, meaning they do not have an undue infl
uence on our
#estimation of the regression model.
#we also have high leverage observations, but they are not outliers.
#we have no observations that are both high leverage and outliers.
```

```
Figure(480x320)
```




In [ ]:
```
#now we will apply logistic regression models to the binary response variables anycrime and anyarrest
#the data management has already been performed
```

In [158]:
```
rdc_logistic['anycrime'] = pd.to_numeric(rdc_logistic['anycrime'], errors='coerce')
rdc_logistic['anyarrest'] = pd.to_numeric(rdc_logistic['anyarrest'], errors='coerce')
```

In [159]:
```
rdc_logistic['cond'] = rdc_logistic['cond'].astype('category')
lreg1 = smf.logit(formula='anyarrest ~ C(cond)', data=rdc_logistic).fit()
print(lreg1.summary())
#equation would be anyarrest = -1.0259-0.1268*cond
```

```
Optimization terminated successfully.
         Current function value: 0.563817
         Iterations 5
                           Logit Regression Results
==============================================================================
Dep. Variable:              anyarrest   No. Observations:                  441
Model:                          Logit   Df Residuals:                      439
Method:                           MLE   Df Model:                            1
Date:                Sun, 12 Mar 2017   Pseudo R-squ.:               0.0006709
Time:                        15:21:29   Log-Likelihood:                -248.64
converged:                       True   LL-Null:                       -248.81
                                        LLR p-value:                    0.5634
================================================================================
                   coef    std err          z      P>|z|      [95.0% Conf. Int.]
--------------------------------------------------------------------------------
Intercept       -1.0259      0.154     -6.645      0.000        -1.328    -0.723
C(cond)[T.1]    -0.1268      0.220     -0.578      0.563        -0.557     0.303
================================================================================
```

In [164]:
```
#however, for logistic regression it makes much more sense to calculate the odds ratio
#if OR=1, the model is not statistically significant
#if OR<1, the response variable becomes less likely as the explanatory one increases
#if OR>1, the response variable becomes more likely as the explanatory one increases
print('Odds Ratios')
print(np.exp(lreg1.params))
#Study subjects in the treatment group (cond=1) are 0.88 times as likely to have had an arrest since rel
ease on parole
#as study subjects in the control group
```

```
Odds Ratios
Intercept        0.358491
C(cond)[T.1]     0.880886
dtype: float64
```

In [163]:
```
# odd ratios with 95% confidence intervals
params = lreg1.params
conf = lreg1.conf_int()
conf['OR'] = params
conf.columns = ['Lower CI', 'Upper CI', 'OR']
print (np.exp(conf))
#we have 95% confidence that the population odds ratio will be between 0.57 and 1.35.
```

```
              Lower CI  Upper CI        OR
Intercept     0.264892  0.485161  0.358491
C(cond)[T.1]  0.572854  1.354554  0.880886
```

In [ ]:
```
#Machine learning encompasses a wide range of statistical methods. These can be used for:
#1. Describe associations
#2. Search for patterns
#3. Make predictions
#We typically do not use ML with hypotheses in mind. instead we learn from the data
#We learn from the test set.
#Accuracy = test error rate. The rate at which it correctly classifies or estimates.
#Goal is to minimize test error rate.
#Linear regression: accuracy = mean squared error
#Variance = change in parameter estimates across different data sets
#Bias = how far off model estimated values are from true values
#ideally we want low variance and low bias, but they are negatively associated. As one decreases, the ot
her increases.
#Generally, complexity of model leads to high variance and low bias
#Simple models will have lower variance, but also be more biased.
#Logistic regression: accuracy = how well the model classifies observations
```

In [ ]:
```
#Supervised Prediction includes:
#Linear regression
#Pattern recognition
#Discriminant analysis
#Multivariate function estimation
#Supervised ML techniques
#Decision trees
#Like linear regression, decision trees are designed for supervised prediction problems.
#Root node, and terminal nodes or leaves.
#Growing the tree process: binary splits maximize correct classification; all cut points are tested; sub
groups showing
#similar outcomes are generated
#Validating the tree: cross validation guards against overfit. A random subset is tested and only 'branc
hes' that improve
#the classification are retained
#Selected sub tree is the lowest probability of misclassification
#Trees allow the handling of many variables that cannot be done as efficiently in linear regression. The
y can also uncover
#constellations of variables that can predict high or low rates of the response variable.
```

In [ ]:
```
#Strengths of decision trees
#Can select from a large number of variables those and their interactions that are most important in det
ermining the
#target or response variable to be explained
#They are easy to interpret and visualize, especially when the tree is small
#Can handle large data sets well and can predict both binary, categorical target variables and also quan
titative ones
#Limitations: small changes in the data can lead to different splits and this can undermine the interpre
tability of the model
#and decision trees are not very reproducible on future data
```

In [201]:
```
reduced_dataset_clean = reduced_dataset.dropna()
len(reduced_dataset_clean)
```

Out[201]: 433

In [205]:
```
predictors = reduced_dataset_clean.ix[:, reduced_dataset_clean.columns != 'allcrimes']
```

In [206]:
```
predictors
```

Out[206]:

| | cond | sex | age | living_situation | support | allarrests | anyarrest | alldrugs | anydrugs | anycrime | arrest_9mo | reincarc_9mo | nun |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 1 | 30.0 | 5 | 11.0 | 2.0 | 1.0 | 2.0 | 1.0 | 1.0 | 1 | 1 | 2 |
| **1** | 1 | 1 | 26.0 | 5 | 11.0 | 0.0 | 0.0 | 1.0 | 1.0 | 1.0 | 1 | 1 | 1 |
| **2** | 1 | 1 | 46.0 | 4 | 9.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0 | 0 | 0 |
| **3** | 0 | 1 | 26.0 | 4 | 9.0 | 0.0 | 0.0 | 1.0 | 1.0 | 1.0 | 0 | 0 | 0 |
| **4** | 0 | 1 | 37.0 | 4 | 6.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0 | 0 | 0 |
| **5** | 1 | 1 | 41.0 | 4 | 9.0 | 0.0 | 0.0 | 1.0 | 1.0 | 1.0 | 0 | 0 | 0 |
| **6** | 1 | 1 | 45.0 | 5 | 11.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0 | 0 | 0 |
| **7** | 0 | 1 | 30.0 | 4 | 9.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0 | 0 | 0 |
| **8** | 0 | 1 | 29.0 | 5 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0 | 0 | 0 |
| **9** | 0 | 1 | 35.0 | 4 | 9.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0 | 0 | 0 |
| **11** | 0 | 1 | 37.0 | 4 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0 | 0 | 0 |
| **12** | 0 | 1 | 24.0 | 4 | 9.0 | 0.0 | 0.0 | 1.0 | 1.0 | 0.0 | 0 | 0 | 0 |
| **13** | 1 | 1 | 24.0 | 4 | 9.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0 | 0 | 0 |
| **14** | 1 | 1 | 32.0 | 4 | 9.0 | 0.0 | 0.0 | 1.0 | 1.0 | 0.0 | 0 | 0 | 0 |

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **15** | 0 | 1 | 54.0 | 4 | 9.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0 | 0 | 0 |
| **16** | 1 | 1 | 24.0 | 4 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0 | 0 | 0 |
| **17** | 1 | 1 | 25.0 | 5 | 9.0 | 4.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1 | 1 | 1 |
| **18** | 1 | 1 | 29.0 | 4 | 9.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 1 | 1 | 1 |
| **19** | 1 | 1 | 41.0 | 4 | 1.0 | 0.0 | 0.0 | 1.0 | 1.0 | 1.0 | 0 | 0 | 0 |
| **20** | 1 | 1 | 30.0 | 5 | 5.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0 | 0 | 0 |
| **21** | 0 | 1 | 21.0 | 5 | 9.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0 | 0 | 0 |
| **22** | 1 | 1 | 37.0 | 4 | 1.0 | 0.0 | 0.0 | 3.0 | 1.0 | 1.0 | 1 | 1 | 2 |
| **23** | 1 | 1 | 39.0 | 5 | 1.0 | 0.0 | 0.0 | 1.0 | 1.0 | 0.0 | 0 | 0 | 0 |
| **24** | 0 | 1 | 28.0 | 4 | 9.0 | 0.0 | 0.0 | 1.0 | 1.0 | 0.0 | 0 | 0 | 0 |
| **25** | 0 | 1 | 34.0 | 4 | 9.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0 | 0 | 0 |
| **26** | 1 | 1 | 19.0 | 4 | 9.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0 | 0 | 0 |
| **27** | 1 | 1 | 41.0 | 4 | 1.0 | 0.0 | 0.0 | 9.0 | 1.0 | 0.0 | 0 | 0 | 0 |
| **28** | 1 | 1 | 24.0 | 5 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0 | 0 | 0 |
| **30** | 0 | 1 | 27.0 | 4 | 9.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0 | 0 | 0 |
| **31** | 0 | 1 | 27.0 | 4 | 9.0 | 1.0 | 1.0 | 2.0 | 1.0 | 1.0 | 1 | 1 | 1 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **438** | 1 | 1 | 25.0 | 4 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0 | 0 | 0 |
| **439** | 1 | 1 | 22.0 | 4 | 9.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1 | 1 | 1 |
| **440** | 0 | 1 | 38.0 | 5 | 9.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1 | 1 | 1 |
| **441** | 0 | 2 | 41.0 | 5 | 9.0 | 0.0 | 0.0 | 1.0 | 1.0 | 0.0 | 1 | 1 | 1 |

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **442** | 0 | 1 | 21.0 | 5 | 8.0 | 0.0 | 0.0 | 1.0 | 1.0 | 0.0 | 0 | 0 | 0 |
| **443** | 0 | 1 | 49.0 | 8 | 3.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0 | 0 | 0 |
| **445** | 0 | 1 | 48.0 | 5 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0 | 0 | 0 |
| **446** | 1 | 1 | 48.0 | 2 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0 | 0 | 0 |
| **447** | 0 | 1 | 46.0 | 7 | 4.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0 | 0 | 0 |
| **448** | 0 | 1 | 55.0 | 4 | 7.0 | 0.0 | 0.0 | 3.0 | 1.0 | 0.0 | 0 | 0 | 0 |
| **449** | 0 | 1 | 50.0 | 5 | 1.0 | 0.0 | 0.0 | 1.0 | 1.0 | 0.0 | 0 | 0 | 0 |
| **451** | 0 | 1 | 54.0 | 5 | 11.0 | 1.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0 | 0 | 0 |
| **453** | 0 | 2 | 50.0 | 4 | 2.0 | 1.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0 | 0 | 0 |
| **454** | 1 | 2 | 44.0 | 5 | 7.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0 | 0 | 0 |
| **455** | 0 | 1 | 21.0 | 5 | 9.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0 | 0 | 0 |
| **457** | 0 | 1 | 45.0 | 5 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0 | 0 | 0 |
| **458** | 0 | 2 | 43.0 | 5 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0 | 0 | 0 |
| **459** | 0 | 1 | 51.0 | 4 | 7.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0 | 0 | 0 |
| **460** | 1 | 1 | 45.0 | 4 | 1.0 | 1.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0 | 0 | 0 |
| **461** | 0 | 1 | 19.0 | 5 | 4.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0 | 0 | 0 |
| **463** | 0 | 1 | 24.0 | 5 | 9.0 | 0.0 | 0.0 | 1.0 | 1.0 | 0.0 | 0 | 0 | 0 |
| **464** | 0 | 2 | 37.0 | 5 | 7.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0 | 0 | 0 |
| **465** | 0 | 1 | 18.0 | 5 | 6.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0 | 0 | 0 |
| **467** | 1 | 1 | 42.0 | 4 | 9.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0 | 0 | 0 |
| **468** | 0 | 1 | 42.0 | 5 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0 | 0 | 0 |

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **469** | 0 | 2 | 42.0 | 2 | 3.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0 | 0 | 0 |
| **470** | 0 | 1 | 24.0 | 5 | 4.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 0 | 0 | 0 |
| **471** | 1 | 1 | 37.0 | 5 | 9.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0 | 0 | 0 |
| **472** | 1 | 2 | 31.0 | 4 | 2.0 | 1.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0 | 0 | 0 |
| **473** | 0 | 1 | 41.0 | 4 | 1.0 | 0.0 | 0.0 | 1.0 | 1.0 | 0.0 | 0 | 0 | 0 |

433 rows × 16 columns

In [207]:
```
targets = reduced_dataset_clean['allcrimes']
```

In [208]:
```
pred_train, pred_test, tar_train, tar_test = train_test_split(predictors, targets, test_size= 0.4)
```

In [209]:
```
print(pred_train.shape, pred_test.shape, tar_train.shape, tar_test.shape)
```

```
(259, 16) (174, 16) (259,) (174,)
```

In [210]:
```
classifier = DecisionTreeClassifier()
classifier = classifier.fit(pred_train, tar_train)
```

In [213]:
```
predictions = classifier.predict(pred_test)
```

In [215]:
```
sklearn.metrics.confusion_matrix(tar_test, predictions)
```

Out[215]:
```
array([[112,   0,   0,   0,   0,   0,   0,   0],
       [  0,  19,   8,   1,   1,   2,   1,   0],
       [  0,   7,   3,   1,   1,   0,   1,   0],
       [  0,   3,   4,   1,   2,   0,   0,   0],
       [  0,   0,   2,   0,   2,   0,   0,   1],
       [  0,   1,   0,   1,   0,   0,   0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0]])
```

In [216]:
```
sklearn.metrics.accuracy_score(tar_test, predictions)
```

Out[216]:
```
0.78735632183908044
```

In [ ]:
```
#Random forests
#Forests of trees
#Splits on only ONE variable in each node. Variable with largest association with Target among
#candidate variables. Only among variables randomly selected to be tested for that node.
#First a subset of explanatory variables is selected at random
#Next the node is split with the Best variable of the subset. After this node is split, a new list of su
bset variables
#is selected at random to split on the next node.
#typical k fold values: 5 or 10
```

In [219]:
```
classifier2 = RandomForestClassifier(n_estimators=25)
classifier2 = classifier.fit(pred_train, tar_train)
```

In [221]:
```
predictions2 = classifier2.predict(pred_test)
```

In [222]:
```
sklearn.metrics.confusion_matrix(tar_test, predictions)
```

Out[222]:
```
array([[112,   0,   0,   0,   0,   0,   0],
       [  0,  20,   7,   2,   1,   2,   0],
       [  0,   7,   3,   1,   1,   0,   1],
       [  0,   3,   3,   1,   3,   0,   0],
       [  0,   0,   2,   1,   2,   0,   0],
       [  0,   1,   0,   0,   1,   0,   0],
       [  0,   0,   0,   0,   0,   0,   0]])
```

In [223]:
```
sklearn.metrics.accuracy_score(tar_test, predictions)
```

Out[223]:
```
0.7931034482758621
```

In [224]:
```
#fit an extra Trees model to the data
model = ExtraTreesClassifier()
model.fit(pred_train, tar_train)
#display the relative importance of each attribute
print(model.feature_importances_)
```

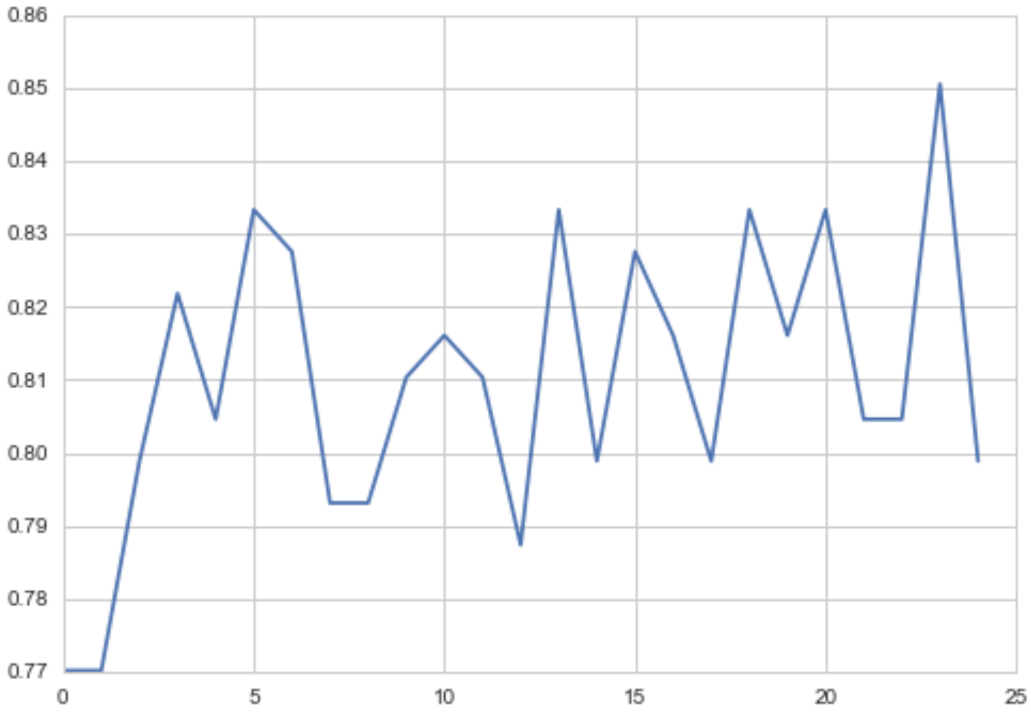
```
[ 0.04472255  0.01493451  0.05962542  0.04225549  0.06900537  0.05969145
  0.03740676  0.04925347  0.04173209  0.46073854  0.00651952  0.02053235
  0.03376509  0.02278637  0.01254948  0.02448154]
```

In [227]:
```
trees = range(25)
accuracy = np.zeros(25)
```

In [228]:
```
for idx in range(len(trees)):
    classifier = RandomForestClassifier(n_estimators = idx + 1)
    classifier = classifier.fit(pred_train, tar_train)
    predictions = classifier.predict(pred_test)
    accuracy[idx] = sklearn.metrics.accuracy_score(tar_test, predictions)
```

In [229]:
```
plt.cla()
plt.plot(trees, accuracy)
```

Out[229]: [<matplotlib.lines.Line2D at 0x11e9f2320>]



In [230]:
```
#Lasso regression
#penalized regression method
#supervised learning method
#shrinkage and selection method
#shrinkage = constraints on parameters that shrinks coefficients to 0
#selection = identifies most imp. variables associated with response variable
#Can increase prediction accuracy and improve model interpretability vs standard OLS
#When lambda=0, it becomes OLS regression
#Bias increases and variance decreases as lambda increases
#in Lasso regression, penalty is not fair if variables are not on the same scale
#standardize all predictor variables to have means equal to 0 and sd = 1
#Lasso regression has several algorithms, among them LAR (least angle regression-)
#sklearn library refers to the penalty term as 'alpha'
```

In [ ]:
```
#Limitations of lasso regression
#1. Selection of variables is 100% statistically driven
#2. If predictors are correlated, lasso arbitrarily selects one
#3. Estimating p values is not straightforward
#4. Different selection methods or statistical softwares can provide different results
#5. No guarantee that selected model is not overfitted nor that it's the best model
#All regression models can produce meaningless models without human intervention
#Best approach is a combination of ML, human intervention, and independent application
```

In [233]:
```
predictors.columns
```

Out[233]:
```
Index(['cond', 'sex', 'age', 'living_situation', 'support', 'allarrests',
       'anyarrest', 'alldrugs', 'anydrugs', 'anycrime', 'arrest_9mo',
       'reincarc_9mo', 'num_arrest', 'num_reincarc', 'violent_charge',
       'property_charge'],
      dtype='object')
```

In [240]:
```
predictors['cond'] = preprocessing.scale(predictors['cond'].astype('float64'))
predictors['sex'] = preprocessing.scale(predictors['sex'].astype('float64'))
predictors['age'] = preprocessing.scale(predictors['age'].astype('float64'))
predictors['living_situation'] = preprocessing.scale(predictors['living_situation'].astype('float64'))
predictors['support'] = preprocessing.scale(predictors['support'].astype('float64'))
predictors['allarrests'] = preprocessing.scale(predictors['allarrests'].astype('float64'))
predictors['anyarrest'] = preprocessing.scale(predictors['anyarrest'].astype('float64'))
predictors['alldrugs'] = preprocessing.scale(predictors['alldrugs'].astype('float64'))
predictors['anydrugs'] = preprocessing.scale(predictors['anydrugs'].astype('float64'))
predictors['anycrime'] = preprocessing.scale(predictors['anycrime'].astype('float64'))
predictors['arrest_9mo'] = preprocessing.scale(predictors['arrest_9mo'].astype('float64'))
predictors['reincarc_9mo'] = preprocessing.scale(predictors['reincarc_9mo'].astype('float64'))
predictors['num_arrest'] = preprocessing.scale(predictors['num_arrest'].astype('float64'))
predictors['num_reincarc'] = preprocessing.scale(predictors['num_reincarc'].astype('float64'))
predictors['violent_charge'] = preprocessing.scale(predictors['violent_charge'].astype('float64'))
predictors['property_charge'] = preprocessing.scale(predictors['property_charge'].astype('float64'))
```

```
/Users/RI/Anaconda/anaconda/lib/python3.5/site-packages/ipykernel/__main__.py:1: SettingWithCopyWarning
:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexin
g-view-versus-copy
  if __name__ == '__main__':
/Users/RI/Anaconda/anaconda/lib/python3.5/site-packages/ipykernel/__main__.py:2: SettingWithCopyWarning
:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexin
g-view-versus-copy
  from ipykernel import kernelapp as app
/Users/RI/Anaconda/anaconda/lib/python3.5/site-packages/ipykernel/__main__.py:3: SettingWithCopyWarning
:
```

```
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexin
g-view-versus-copy
  app.launch_new_instance()
/Users/RI/Anaconda/anaconda/lib/python3.5/site-packages/ipykernel/__main__.py:4: SettingWithCopyWarning
:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexin
g-view-versus-copy
/Users/RI/Anaconda/anaconda/lib/python3.5/site-packages/ipykernel/__main__.py:5: SettingWithCopyWarning
:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexin
g-view-versus-copy
/Users/RI/Anaconda/anaconda/lib/python3.5/site-packages/ipykernel/__main__.py:6: SettingWithCopyWarning
:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexin
g-view-versus-copy
/Users/RI/Anaconda/anaconda/lib/python3.5/site-packages/ipykernel/__main__.py:7: SettingWithCopyWarning
:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexin
g-view-versus-copy
/Users/RI/Anaconda/anaconda/lib/python3.5/site-packages/ipykernel/__main__.py:8: SettingWithCopyWarning
```

```
:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexin
g-view-versus-copy
/Users/RI/Anaconda/anaconda/lib/python3.5/site-packages/ipykernel/__main__.py:9: SettingWithCopyWarning
:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexin
g-view-versus-copy
/Users/RI/Anaconda/anaconda/lib/python3.5/site-packages/ipykernel/__main__.py:10: SettingWithCopyWarnin
g:
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/stable/indexing.html#indexin
g-view-versus-copy
/Users/RI/Anaconda/anaconda/lib/python3.5/site-packages/ipykernel/__main__.py:11: SettingWithCopyWarnin
g:
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/stable/indexing.html#indexin
g-view-versus-copy
/Users/RI/Anaconda/anaconda/lib/python3.5/site-packages/ipykernel/__main__.py:12: SettingWithCopyWarnin
g:
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/stable/indexing.html#indexin
g-view-versus-copy
/Users/RI/Anaconda/anaconda/lib/python3.5/site-packages/ipykernel/__main__.py:13: SettingWithCopyWarnin
```

```
g:
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/stable/indexing.html#indexin
g-view-versus-copy
/Users/RI/Anaconda/anaconda/lib/python3.5/site-packages/ipykernel/__main__.py:14: SettingWithCopyWarnin
g:
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/stable/indexing.html#indexin
g-view-versus-copy
/Users/RI/Anaconda/anaconda/lib/python3.5/site-packages/ipykernel/__main__.py:15: SettingWithCopyWarnin
g:
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/stable/indexing.html#indexin
g-view-versus-copy
/Users/RI/Anaconda/anaconda/lib/python3.5/site-packages/ipykernel/__main__.py:16: SettingWithCopyWarnin
g:
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/stable/indexing.html#indexin
g-view-versus-copy
```

In [243]:

```
#split data into train and test tests
pred_train, pred_test, tar_train, tar_test = train_test_split(predictors, targets,
                                                              test_size=.3, random_state=123)
```

In [246]:

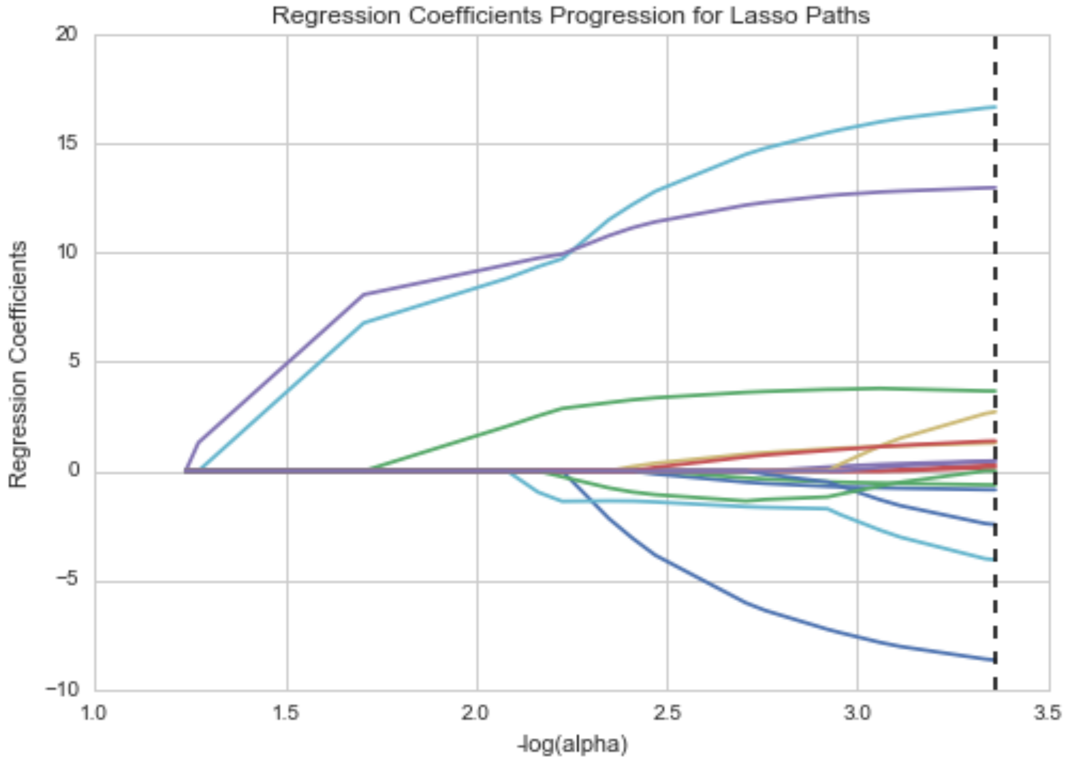
```
# specify the lasso regression model
model=LassoLarsCV(cv=10, precompute=False).fit(pred_train,tar_train)
```

In [247]:
```
# print variable names and regression coefficients
dict(zip(predictors.columns, model.coef_))
```

Out[247]:
```
{'age': 0.00922117951327419,
 'allarrests': 0.97816170401806635,
 'alldrugs': 0.23108994348020051,
 'anyarrest': -0.48346833180493959,
 'anycrime': 0.73524559191125771,
 'anydrugs': 0.01539580168267698,
 'arrest_9mo': 0.15463119466699699,
 'cond': -0.049336041215996017,
 'living_situation': 0.027322802531668331,
 'num_arrest': -0.13464215269435476,
 'num_reincarc': 0.0,
 'property_charge': 0.024425067866711093,
 'reincarc_9mo': -0.23122026644497329,
 'sex': -0.036216582632220544,
 'support': 0.074415953782179037,
 'violent_charge': 0.072286686175727102}
```

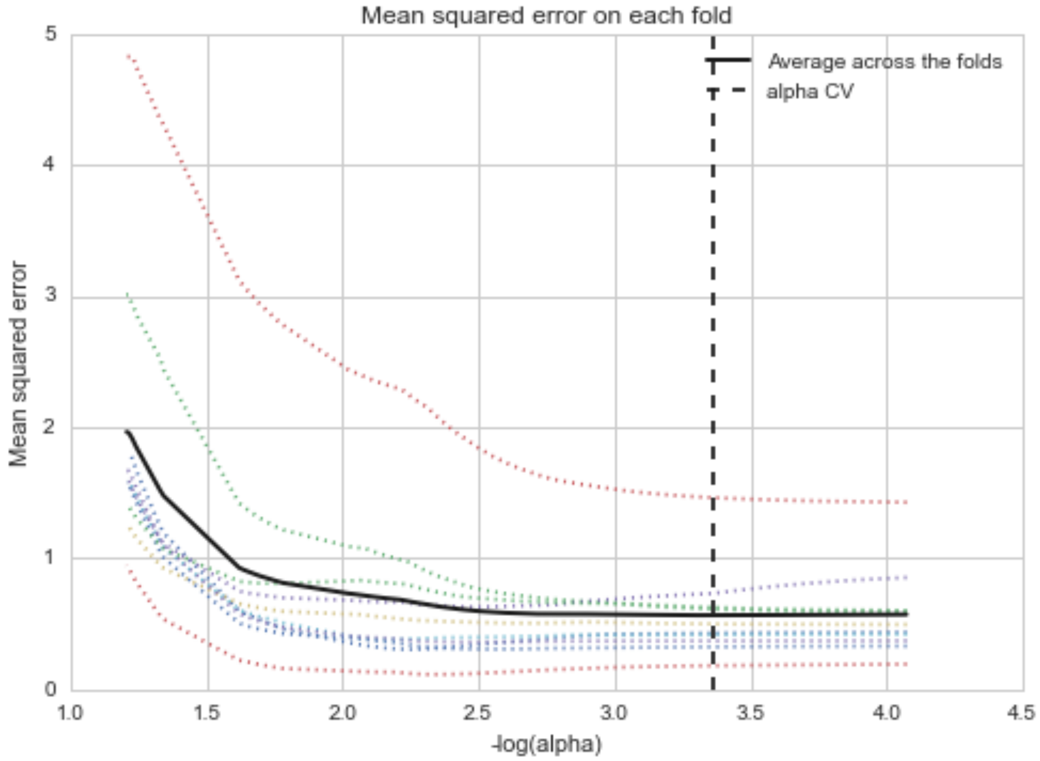
In [248]:
```
# plot coefficient progression
m_log_alphas = -np.log10(model.alphas_)
ax = plt.gca()
plt.plot(m_log_alphas, model.coef_path_.T)
plt.axvline(-np.log10(model.alpha_), linestyle='--', color='k',
            label='alpha CV')
plt.ylabel('Regression Coefficients')
plt.xlabel('-log(alpha)')
plt.title('Regression Coefficients Progression for Lasso Paths')
```

```
Out[248]: <matplotlib.text.Text at 0x11ea7eac8>
```



In [249]:
```
# plot mean square error for each fold
m_log_alphascv = -np.log10(model.cv_alphas_)
plt.figure()
plt.plot(m_log_alphascv, model.cv_mse_path_, ':')
plt.plot(m_log_alphascv, model.cv_mse_path_.mean(axis=-1), 'k',
         label='Average across the folds', linewidth=2)
plt.axvline(-np.log10(model.alpha_), linestyle='--', color='k',
            label='alpha CV')
plt.legend()
plt.xlabel('-log(alpha)')
plt.ylabel('Mean squared error')
plt.title('Mean squared error on each fold')
```

Out[249]: <matplotlib.text.Text at 0x11f00aa90>



In [250]:
```
# MSE from training and test data
from sklearn.metrics import mean_squared_error
train_error = mean_squared_error(tar_train, model.predict(pred_train))
test_error = mean_squared_error(tar_test, model.predict(pred_test))
print ('training data MSE')
print(train_error)
print ('test data MSE')
print(test_error)
```

```
training data MSE
0.460347733345
test data MSE
0.54794067835
```

In [251]:
```
# R-square from training and test data
rsquared_train=model.score(pred_train,tar_train)
rsquared_test=model.score(pred_test,tar_test)
print ('training data R-square')
print(rsquared_train)
print ('test data R-square')
print(rsquared_test)
```

```
training data R-square
0.766331632034
test data R-square
0.789061561182
```

In [ ]:
```
#Cluster analysis
#Unsupervised learning method = no response variable included in the analysis
#Goal: to have less variance within clusters, and more between clusters
#Can also be used as a method of data reduction, to reduce number of variables
#to the number of categorical variables equal to the clusters produced
#Canonical discriminant analysis:
#creates a smaller number of variables
#linear combinations of clustering variables
#canonical variables are ordered by proportion of variance accounted for
#majority of variance is accounted for by first few canonical variables
```

In [255]:
```
clustervar = predictors.copy()
```

In [257]:
```
# split data into train and test sets
clus_train, clus_test = train_test_split(clustervar, test_size=.3, random_state=123)
```

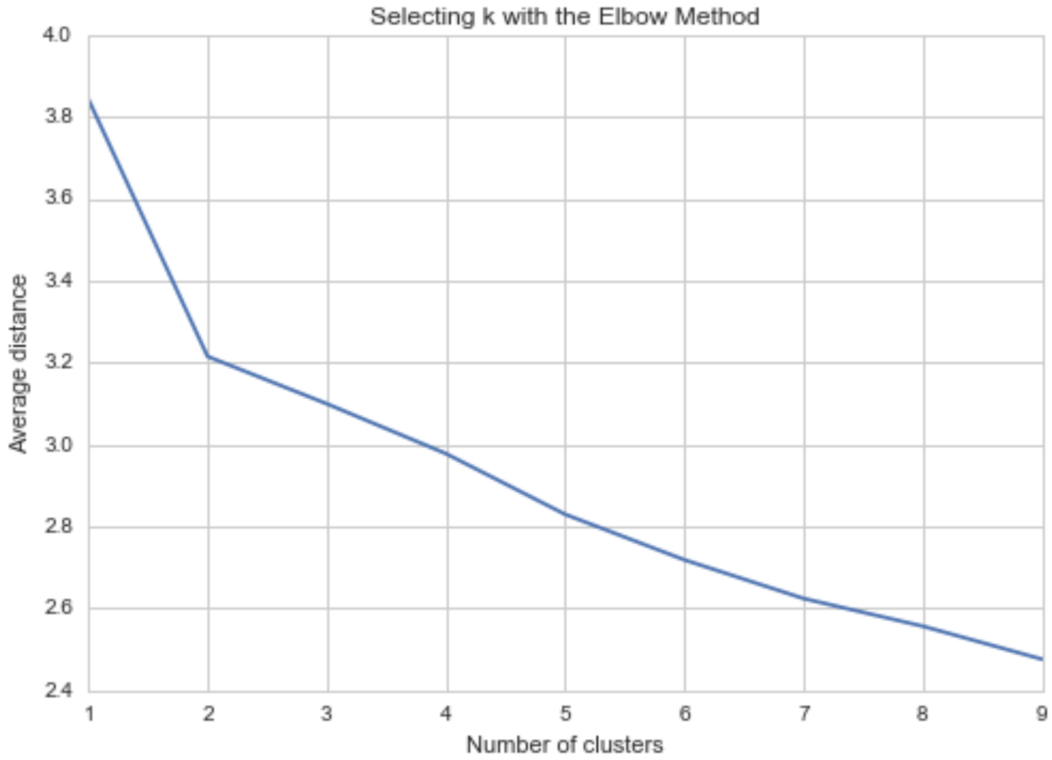
In [258]:
```
# k-means cluster analysis for 1-9 clusters
from scipy.spatial.distance import cdist
clusters=range(1,10)
meandist=[]

for k in clusters:
    model=KMeans(n_clusters=k)
    model.fit(clus_train)
    clusassign=model.predict(clus_train)
    meandist.append(sum(np.min(cdist(clus_train, model.cluster_centers_, 'euclidean'), axis=1))
    / clus_train.shape[0])
```

In [259]:
```
"""
Plot average distance from observations from the cluster centroid
to use the Elbow Method to identify number of clusters to choose
"""

plt.plot(clusters, meandist)
plt.xlabel('Number of clusters')
plt.ylabel('Average distance')
plt.title('Selecting k with the Elbow Method')
```

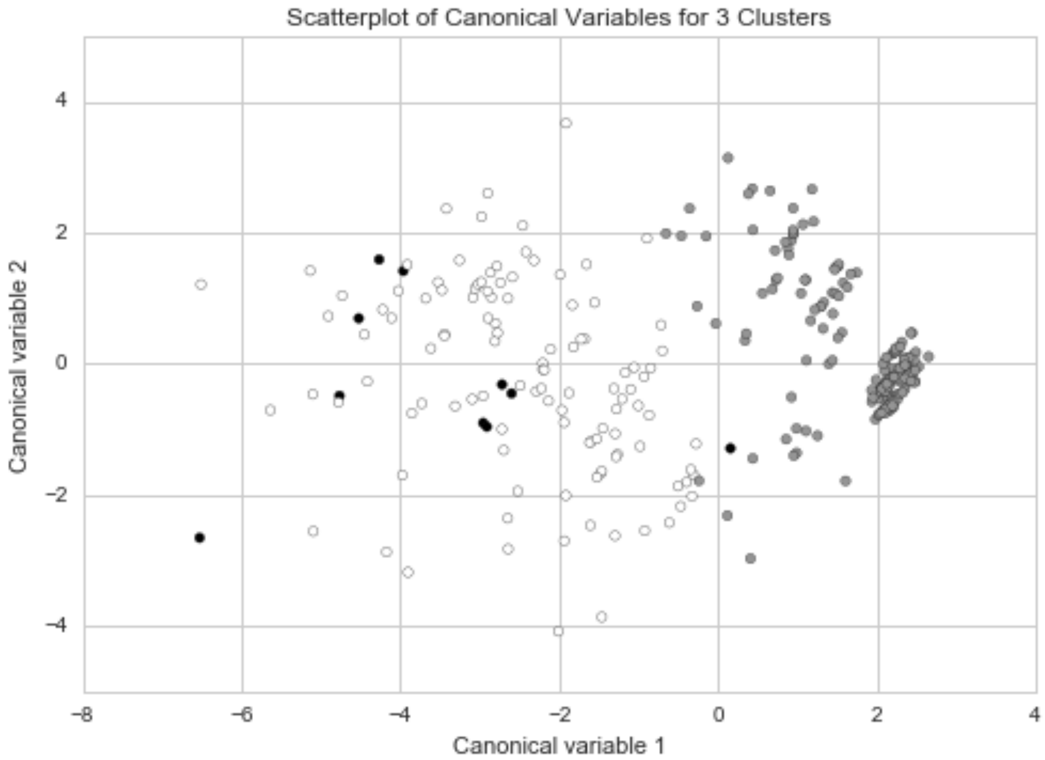
Out[259]: <matplotlib.text.Text at 0x11f661b70>



In [260]:
```
# Interpret 3 cluster solution
model3=KMeans(n_clusters=3)
model3.fit(clus_train)
clusassign=model3.predict(clus_train)
```

In [261]:
```
# plot clusters

from sklearn.decomposition import PCA
pca_2 = PCA(2)
plot_columns = pca_2.fit_transform(clus_train)
plt.scatter(x=plot_columns[:,0], y=plot_columns[:,1], c=model3.labels_,)
plt.xlabel('Canonical variable 1')
plt.ylabel('Canonical variable 2')
plt.title('Scatterplot of Canonical Variables for 3 Clusters')
plt.show()
```



In [288]:
```
"""
BEGIN multiple steps to merge cluster assignment with clustering variables to examine
cluster variable means by cluster
"""
# create a unique identifier variable from the index for the
# cluster training data to merge with the cluster assignment variable
clus_train.reset_index(level=0, drop=True)
# create a list that has the new index variable
cluslist=list(clus_train['index'])
# create a list of cluster assignments
labels=list(model3.labels_)
# combine index variable list with cluster assignment list into a dictionary
newlist=dict(zip(cluslist, labels))
newlist
# convert newlist dictionary to a dataframe
newclus=pd.DataFrame.from_dict(newlist, orient= 'index')
newclus
# rename the cluster assignment column
newclus.columns = ['cluster']

# now do the same for the cluster assignment variable
# create a unique identifier variable from the index for the
# cluster assignment dataframe
# to merge with cluster training data
newclus.reset_index(level=0, drop=True)
# merge the cluster assignment dataframe with the cluster training variable dataframe
# by the index variable
merged_train=pd.merge(clus_train, newclus, left_index=True, right_index=True)
merged_train.head(n=100)
# cluster frequencies
merged_train.cluster.value_counts()
```

Out[288]:
```
1    110
0     80
2      6
Name: cluster, dtype: int64
```

In [289]:
```
"""
END multiple steps to merge cluster assignment with clustering variables to examine
cluster variable means by cluster
"""

# FINALLY calculate clustering variable means by cluster
clustergrp = merged_train.groupby('cluster').mean()
print ("Clustering variable means by cluster")
print(clustergrp)
```

```
Clustering variable means by cluster
            level_0       index      cond       sex       age  \
cluster
0        155.375000  218.900000 -0.125445 -0.132803 -0.115947
1        145.590909  224.527273 -0.061788  0.064352 -0.084813
2         61.666667  210.333333 -0.025412 -0.023272 -0.197488

         living_situation   support  allarrests  anyarrest  alldrugs  \
cluster
0               -0.085034  0.187033    0.126453   0.185858  0.048168
1               -0.063659 -0.101742    0.012498   0.058328 -0.121868
2                0.118736 -0.369018   -0.078019   0.176315  0.114828

         anydrugs  anycrime  arrest_9mo  reincarc_9mo  num_arrest  \
cluster
0        0.264865  0.053554   -0.055090     -0.004977   -0.102726
1       -0.123996  0.094418    0.091929      0.035152    0.101847
2       -0.221593 -0.008142    0.147061      0.280647    0.036514

         num_reincarc  violent_charge  property_charge
cluster
0           -0.043510        0.059618         0.010329
1            0.012591        0.108077         0.091644
2            0.065391       -0.168830        -0.252749
```

In [307]:
```
# validate clusters in training data by examining cluster differences in GPA using ANOVA
# first have to merge GPA with clustering variables and cluster assignment data
allarrests_data['targets'] = pd.DataFrame(targets)
# split GPA data into train and test sets
arrests_train, arrests_test = train_test_split(allarrests_data, test_size=.3, random_state=123)
arrests_train1=pd.DataFrame(arrests_train)
arrests_train1.reset_index(level=0, inplace=True)
merged_train_all=pd.merge(arrests_train1, merged_train, left_index=True, right_index=True)
sub1 = merged_train_all[['targets', 'cluster']].dropna()

import statsmodels.formula.api as smf
import statsmodels.stats.multicomp as multi

allarrests_mod = smf.ols(formula='targets ~ C(cluster)', data=sub1).fit()
print (allarrests_mod.summary())

print ('means for all arrests by cluster')
m1= sub1.groupby('cluster').mean()
print (m1)

print ('standard deviations for all arrests by cluster')
m2= sub1.groupby('cluster').std()
print (m2)

mc1 = multi.MultiComparison(sub1['targets'], sub1['cluster'])
res1 = mc1.tukeyhsd()
print(res1.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                targets   R-squared:                       0.004
Model:                            OLS   Adj. R-squared:                 -0.007
Method:                 Least Squares   F-statistic:                    0.3610
Date:                Tue, 21 Mar 2017   Prob (F-statistic):              0.697
Time:                        06:54:56   Log-Likelihood:                -358.59
```

```
No. Observations:                 196   AIC:                             723.2
Df Residuals:                     193   BIC:                             733.0
Df Model:                           2
Covariance Type:            nonrobust
===================================================================================
                      coef    std err          t      P>|t|      [95.0% Conf. Int.]
-----------------------------------------------------------------------------------
Intercept           0.9000      0.170      5.298      0.000         0.565     1.235
C(cluster)[T.1]    -0.1545      0.223     -0.692      0.490        -0.595     0.286
C(cluster)[T.2]    -0.4000      0.643     -0.622      0.535        -1.668     0.868
==============================================================================
Omnibus:                      178.561   Durbin-Watson:                   2.128
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             3045.652
Skew:                           3.497   Prob(JB):                         0.00
Kurtosis:                      21.001   Cond. No.                         7.01
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
means for all arrests by cluster
          targets
cluster
0        0.900000
1        0.745455
2        0.500000
standard deviations for all arrests by cluster
          targets
cluster
0        1.879941
1        1.222393
2        0.836660
Multiple Comparison of Means - Tukey HSD,FWER=0.05
============================================
group1 group2 meandiff  lower  upper  reject
--------------------------------------------
```

```
  0      1    -0.1545  -0.6819 0.3728 False
  0      2      -0.4   -1.9192 1.1192 False
  1      2    -0.2455  -1.7501 1.2592 False
----------------------------------------------
```

In [ ]:

In [ ]: