# Experimental evidence of lack of correlation between psychological evaluation and sexual aggression in inmates

Brian Parks

***Abstract*—It is an unsurprising preconcieved notion that certain psychological traits will lend themselves toward more sexually aggressive or meek behavior, causing certain individuals to become repeat sexual aggressors or repeat sexual victims when introduced to the corrective system. Well-devloped psychological batteries have been developed that are designed to evaluate the psychological aspects of individuals and the record of sexual activity is also well-recorded. Thus, it is the aim of this study to determine whether or not there is a correlation between the two. We begin by evaluating an approach suggested by an expert in the field, making slight changes to compare the stability of the method. Determining it to be extremely unstable, we attempt to explain why this is the case, proposing and evaluating several alternatives to the original method. Failing to provide a robust correlation, we turn to concepts in machine learning to discuss how the problem is ill-posed.**

## I. Introduction

WHEN introduced to the correctional system in the state of Colorado, all inmates take the Coolidge Correctional Inventory [1] to assess their psychological characteristics. This test, at least in theory, provides a concrete representation of the psychological nature of the individual, which can then be used to objectively compare the individual with others. This could have many applications and implications, one of which is the determination of whether or not a new inmate will become a sexual aggressor (predator) or victim when in the company of other inmates.

This seems a reasonable enough assumption; however, since we can't be certain about this fact, it makes sense to jump into the problem from a pattern analysis standpoint rather than a strict classification standpoint, which is the eventual goal. When all is said and done, the prison system would like to ascertain, based on data they already have, whether or not a particular inmate will become a repeated sexual predator or victim within some reasonable degree of certainty.

To analyze the numeric data (which can be thought of as points in $n$-dimensional space), we choose clustering techniques to divide the data up into several groups that appear (based on distance/correlation metrics we choose) to be more or less homogeneous with respect to the components of each data vector. Once these clusters have been formed by various algorithms, we assign each their respective label (whether they correspond to a victim or a predator) and analyze the homogeneity of the clusters formed.

B. Parks is with the University of Colorado at Colorado Springs, Colorado Springs, CO, USA

The remainder of the paper is as follows: we describe our experimental setup in Section II and analyze the results thus obtained in Section III. We follow that with a discussion of alternate approaches to the same problem in Section IV. Failing to find a proper correlation, we turn to the literature in Section V and conclude in Section VI.

## II. Experimental Setup

For our experiments, we implement both a partitional and an agglomerative clustering algorithm. Our partitional algorithm is a simple k-means approach, decribed below with a brief description of our agglomerative approach. To determine which vectors belonged to which clusters, we implemented several distance metrics, including the $L_2$, $L_1$, and $L_\infty$ norms, as well as a cosine correlation and Pearson's product-moment correlation coefficient [2]. Since distance metrics are defined as

$$D : (\mathbf{u}, \mathbf{v}) \rightarrow \{x \in \mathbb{R} | x \geq 0\} \tag{1}$$

while correlations are defined as

$$C : (\mathbf{u}, \mathbf{v}) \rightarrow \{x \in \mathbb{R} | -1 \leq x \leq 1\} \tag{2}$$

we transform the latter into the former via the following equation:

$$d = 1 - r \tag{3}$$

for $r$ a correlation and $d$ the corresponding distance.

For those unfamiliar, the $L_2$ norm is commonly referred to as Euclidean distance when used as a distance metric computed over the difference of two vectors (it is simply the square root of the sum of squares of the components in the difference vector), as popularized by Pythagoras's Theorem. Likewise, the $L_1$ norm is also known as "Manhattan distance" when used as a distance metric and is the sum of the absolute values of the components of the difference vector. $L_\infty$ is also known as the supremum norm, as it is eqivalent to the supremum (maximum) of the absolute values of a vector.

These distance metrics were chosen to provide a varied range of possibilities, as some problems lend themselves better to particular distance measures. Unfortunately, there is little analysis that can be done to avoid making the choice of distance metric *a priori*. By comparing many, we can reduce the chance that an *a priori* assumption will unfairly bias the results.

### A. $k$-Means

$k$-means is a straighforward partitional algorithm. It starts by choosing $k$ means, which can eaither reflect the data being clustered or be completely arbitrary. In our implementation, we randomized the list of vectors and chose the first $k$ in the resulting list. For purpose of generating meaningful results, we varied $k$ from 2 to 10 and ran the algorithm 10 times for each to reduce the potential effect of an unfairly good or bad randomization.

Once $k$ initial means have been chosen, the remaining data points (vectors) can be sorted into clusters around those means so as to minimize the distance between each vector and its associated mean. The means are then recomputed for each cluster based on the vectors it contains and the vectors are re-sorted to cluster around these new means. This step is then repeated until some stopping condition is met. For our purposes, we performed the re-sorting step 10 times.

### B. Agglomerative

As an agglomerative clustering implementation, we started by creating $n$ clusters, each containing one of our $n$ input vectors. Given a threshold $t$, we iterated through all the clusters, one by one, and aggregated it with any and all clusters whose mean was closer than $t$ as determined by the chosen distance metric. The procedure was iterated, incrementing $t$ by the initial value of $t$ each time, so as to create larger and larger clusters, until the number of clusters was less than 10.

## III. Results and Analysis

In order to analyze the quality of the clusters, we define a "homogeneity" metric $H$ of a set $V$ as follows:

$$H(V) = \frac{1}{\|V\|^2} \sum_{i=1}^{\|C\|} (\|V_{C_i}\|)^2 \tag{4}$$

where $C$ represents the set of all classes $c_1, c_2, \ldots$, $V$ represents the set of all data vectors, and the $\|x\|$ notation denotes the size of each set. $V_{c_i}$ indicates the set of all vectors belonging to class $c_i$. Thus, the homogeneity of a cluster will only ever be 1 (the maximum) if all of the examples are of the same class ($\frac{1}{N^2} \cdot (N^2 + 0 + \ldots) = 1$). Likewise, the homogeneity will decrease appropriately as the number of examples not belonging to this class increases. For example,

$$H = \frac{1}{N^2} \cdot \left( \left(\frac{N}{2}\right)^2 + \left(\frac{N}{2}\right)^2 \right) = \frac{1}{2} \tag{5}$$

for a cluster evenly divided between two classes and

$$H = \frac{1}{N^2} \cdot \left( \left(\frac{N}{k}\right)^2 + \left(\frac{N}{k}\right)^2 + \ldots \right) = \frac{1}{N^2} \cdot \frac{kN^2}{k^2} = \frac{1}{k} \tag{6}$$

in the general case.

In order to take into account all clusters, a weighted average was computed such that larger clusters affect the quality metric linearly with their size. Note that, since the number of vectors per class is skewed, with 75 predators and 25 victims, a cluster with 75% predators and 25% victims (and having a homogeneity of $\frac{3^2+1}{4^2} = \frac{5}{8} = 0.625$) is pure chance.

| | k-means | Agg |
|---|---|---|
| $L_2$ | 0.6466 | 0.7012 |
| $L_1$ | 0.6568 | 0.6862 |
| $L_\infty$ | 0.6605 | 0.6640 |
| $cos$ | 0.6631 | 0.7062 |
| Pearson | 0.6546 | 0.6614 |

TABLE I
HOMOGENEITY MEASURES FOR THE CLUSTERS GENERATED BY TWO ALGORITHMS AND FIVE DISTANCE MEASURES

Table I shows the homogeneity scores for each distance metric for both algorithms computed from a representative experiment. It is not difficult to see that performance is only slightly better than chance for any algorithm or distance metric.

In order to successfully cluster, especially with the eventual goal of classification, there must be a strong correlation by means of some function between the input data (in this case, the psychological evaluation) and the resulting labels (whether the inmate is a repeated predator or victim). The result that clustering yields results that are only slightly above chance leads to the conclusion that this correlation is very close to zero; in other words, there is barely any correlation.

To further test this conclusion, the data set was run through a Support Vector Machine (SVM) [3], the going trend in machine learning classification problems. SVMs are particularly good at learning tranforms that will further separate spatially intertwined classes to yield recognition results much better than their less-sophisticated cousins, such as nearest-neighbor. On this particular dataset, the SVM could not get above 75% successful classification, indicating that this data does not very well determine the labels given. (While chance on this problem is 50%, an SVM should perform in the 90% range for a simple classification problem such as this.)

## IV. Alternate Approaches

One of the nice properties of numeric data is that it can always be treated as a Digital Signal Processing problem. This is further motivated by the fact that Dr. Coolidge mentioned (as one of his *a priori* assumptions) that the contour of the vectors was more important than the specific values themselves. To that end, we performed a simple discrete Fourier transform on the data before attempting to divide it or aggregate it into clusters. This met with the same rate of failure as the original experiments.

In addition, we created balanced subsets of the original data, in case the considerably greater amount of predator data had skewed the results. This did not help a bit, as the reuslting clusters then had an even distribution of both predators and victims. All of these approaches ended with less-than-ideal results, which begged questions.

## V. Discussion

The first issue that is brought to light here is that clustering is not intended as a classification technique. It is primarily a pattern analysis technique; that is, input data is sorted into groups based on patterns that the computer discovers, not

based on *a priori* assumptions. The fact that these clusters thus formed do not correspond to any reasonable separation between classes is a clear indication that either the representation of the data is inadequate for the problem at hand or that there is no function $F$ that can be learned or discovered to map a given input vector $v$ to a specific class $c$.

In order to perform classification, either by analyzing the clusters formed or through the use of a true classification algorithm such as Nearest Neighbor or Support Vector Machines, two properties must hold true. First, there must be a function $F$ as described above. This is shown to likely not be the case by the SVM experiments and the clustering experiments. However, this alone is not sufficient to show that the data poorly determines the class labels, as there is always a function (akin to a lookup table) that will map training data to the correct labels but provide horrid classification results.

However, given a function $F$ as above (even if it is the lookup table example), a correlation between two input vectors should correspond to a correlation between labels, for some definition of correlation. This could be a correlation in the mathematical sense or simply a distance. Both approaches were attempted in this study, with equivalent failure rates.

## VI. Conclusion

In this study, we have attempted to discover patterns in psychological evaluation data of Colorado inmates with the goal of using said data and patterns to construct some sort of classification model to determine the expected behavior of future inmates. We implemented several approaches and compared the results, ultimately concluding that either the data was insufficient for the expected classification or the aspects described by the data did not uniquely determine a future behavior. We also presented reasons why this might be the case in terms of machine learning techniques.

## Appendix

The following is an example of the output from one run of our clustering code base using $k$-means with $k$ = 10 using the $L_2$ norm as a distance measure.

```
--- k: 10   Distance Metric: l2 Trial 1 ---
['P10', 'P13', 'P16', 'P18', 'P21', 'P27',
'P29', 'P32', 'P33', 'P34', 'P35', 'P38',
'P39', 'P43', 'P45', 'P46', 'P48', 'P49',
'P50', 'P52', 'P56', 'P57', 'P58', 'P59',
'P63', 'P67', 'P73', 'P74', 'P8', 'V1',
'V10', 'V13', 'V17', 'V18', 'V19', 'V2',
'V22', 'V23', 'V24', 'V5', 'V8', 'V9']
Class P: 29/42 (69.05%)
Class V: 13/42 (30.95%)
['P11', 'P15', 'P2', 'P25', 'P36', 'P37',
'P40', 'P41', 'P5', 'P53', 'P55', 'P6',
'P72', 'P75', 'V11', 'V12', 'V16', 'V21',
'V25', 'V3', 'V4']
Class P: 14/21 (66.67%)
Class V: 7/21 (33.33%)
['P1', 'P12', 'P20', 'P26', 'P28', 'P3',
'P4', 'P42', 'P44', 'P47', 'P51', 'P65',
'P69', 'P7', 'V14', 'V15', 'V20', 'V7']
Class P: 14/18 (77.78%)
Class V: 4/18 (22.22%)
['P14', 'P17', 'P19', 'P24', 'P54', 'P62',
'P64', 'P66', 'P68', 'P70', 'P71', 'V6']
Class P: 11/12 (91.67%)
Class V: 1/12 (8.33%)
['P23', 'P30']
Class P: 2/2 (100.00%)
['P31']
Class P: 1/1 (100.00%)
['P22']
Class P: 1/1 (100.00%)
['P61']
Class P: 1/1 (100.00%)
['P9']
Class P: 1/1 (100.00%)
['P60']
Class P: 1/1 (100.00%)
```

## References


[1] F. L. Coolidge, D. L. Segal, K. J. Klebe, B. S. Cahill, and J. M. Whitcomb, "Psychometric properties of the coolidge correctional inventory in a sample of 3,962 prison inmates," *Behavior Sciences and the Law*, vol. 27, no. 5, pp. 713–726, 2009. 1

[2] K. Pearson, "Mathematical contributions to the theory of evolution. III. regression, heredity and panmixia," *Philosophical Transactions of the Royal Society of London, Series A*, vol. 187, pp. 253–318, 1896. 1

[3] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995. 2

```
import numpy as N
import sys, random

from math import sqrt

def l2(v):
    return sqrt(sum(N.multiply(v, v)))

def l1(v):
    return sum(abs(v))

def linf(v):
    return max(abs(v))

def cosc(a, b):
    l2a = l2(a)
    l2b = l2(b)
    if l2a == 0 or l2b == 0:
        return 0
    return N.dot(a, b)/(l2(a)*l2(b))

def pcc(a, b):
    x = a
    y = b
    # copied from wikipedia page
    sum_sq_x = 0
    sum_sq_y = 0
    sum_coproduct = 0
    mean_x = x[1]
    mean_y = y[1]
    for i in range(1, len(x)):
        sweep = (i - 1.0) / i
        delta_x = x[i] - mean_x
        delta_y = y[i] - mean_y
        sum_sq_x += delta_x * delta_x * sweep
        sum_sq_y += delta_y * delta_y * sweep
        sum_coproduct += delta_x * delta_y * sweep
        mean_x += delta_x / i
        mean_y += delta_y / i
    pop_sd_x = sqrt( sum_sq_x/len(x) )
    pop_sd_y = sqrt( sum_sq_y/len(x) )
    cov_x_y = sum_coproduct/len(x)
    if pop_sd_x == 0 or pop_sd_y == 0: # not copied
        return 0                       # not copied
    correlation = cov_x_y / (pop_sd_x * pop_sd_y)
    # end copied from wikipedia page
    return correlation

class Cluster:
    def __init__(self):
        self.vectors = []
        self.labels = []
```

```
    def mean(self):
        if len(self.vectors) == 0:
            return N.zeros(len(vectors[0])-1, float).astype(list)
        vsum = N.zeros(len(self.vectors[0]), float)
        for i in range(len(self.vectors)):
            vsum = N.add(vsum.astype(list), self.vectors[i])
        return N.divide(vsum, len(self.vectors))

def kmeans(vectors, k, distance, iter=10):
    labels = [ v[0] for v in vectors]
    vectors = [ v[1:] for v in vectors]

    # Convert numerics to numeric
    vectors = [ [ float(x) for x in v ] for v in vectors ]

    if len(vectors) < k:
        print "Must be > k vectors."
        return

    # Initialize clusters
    c = []
    for i in range(k):
        c += [ Cluster() ]
        c[i].vectors += [ vectors[i] ]
    for i in range(k, len(vectors)):
        d = [distance(vectors[i], c[j].mean()) for j in range(k)]
        a = N.argmax(d)
        c[a].vectors += vectors[i]

    # Run k-means
    stats = []
    for i in range(iter):
        means = [x.mean() for x in c]
        c = [ Cluster() for j in range(k)]
        for j in range(len(vectors)):
            d = [distance(vectors[j], c[l].mean()) for l in range(k)]
            a = N.argmax(d)
            c[a].vectors += [ vectors[i] ]
            if i == iter-1: #last pass
                while len(stats) <= a:
                    stats += [ [] ]
                stats[a] += [ labels[j] ]
                #print "Vector %d of class %s is in cluster %d" % (j,
                    labels[j], a)
    return stats

def agg(vectors, step, distance, stop=10):
    labels = [ v[0] for v in vectors]
    vectors = [ v[1:] for v in vectors]

    # Convert numerics to numeric
    vectors = [ [ float(x) for x in v ] for v in vectors ]
```

```
    c = []
    for i in range(len(vectors)):
        x = Cluster()
        x.vectors += [ vectors[i] ]
        x.labels += [ labels[i] ]
        c += [ x ]
    thresh = step

    while len(c) > stop:
        print len(c)
        cnew = []
        for i in xrange(len(c)):
            if c[i] is None:
                continue
            m = c[i].mean()
            a = c[i].vectors
            b = c[i].labels
            c[i] = None
            for j in xrange(i+1, len(c)):
                if c[j] is None:
                    continue
                if distance(c[j].mean(), m) < thresh:
                    a += c[j].vectors
                    b += c[j].labels
                    c[j] = None
            x = Cluster()
            x.vectors = a
            x.labels = b
            cnew += [ x ]
        c = cnew
        thresh += step

    stats = []
    for x in c:
        stats += [ x.labels ]
    return stats

def printstats(stats):
    for s in stats:
        s.sort()
        print s

        s = [ x[0] for x in s ]
        uniq = []
        for x in s:
            if x not in uniq:
                uniq += [ x ]
        uniq.sort()
        for u in uniq:
            count = sum([ 1 for x in s if x == u ])
            print "Class %s: %d/%d (%.2f%%)" % (u, count, len(s), (count*
                100.0/len(s)))
```

```
# -----------------------------------------------------------
if __name__ == "__main__":
    try:
        alg = sys.argv[1]
        dist = sys.argv[2]
        arg = sys.argv[3]
        ifile = sys.argv[4]
    except IndexError:
        print "USAGE: python main.py <algorithm> <distance metric> <
            argument> <datafile>"
        print "    algorithm is one of 'kmeans' or 'agg'"
        print "    distance metric is one of 'l2', 'l1', 'linf', 'cosc', or
            'pcc'"
        print "    argument is the necessary argument to specified
            algorithm:"
        print "        kmeans - value of k"
        print "        agg - threshold step"
        sys.exit(1)

    # Read in vectors
    fp = open(ifile, 'r')
    vectors = []
    for line in fp:
        if line[0] == '#':
            continue
        v = line[:-1].split(',')
        vectors += [ v ]
    fp.close()

    #randomize vectors
    random.shuffle(vectors)

    if dist == 'l2':
        distance = lambda a, b: l2(a - b)
    elif dist == 'l1':
        distance = lambda a, b: l1(a - b)
    elif dist == 'linf':
        distance = lambda a, b: linf(a - b)
    elif dist == 'cosc':
        distance = lambda a, b: 1 - cosc(a, b)
    elif dist == 'pcc':
        distance = lambda a, b: 1 - pcc(a, b)
    else:
        print "Unknown distance metric"
        sys.exit(1)

    if alg == 'kmeans':
        stats = kmeans(vectors, int(arg), distance)
    elif alg == 'agg':
        stats = agg(vectors, float(arg), distance)
    else:
        print "Unknown algorithm"
```

```
            sys.exit(1)

        printstats(stats)
```